DIVISION DE EDUCACION CONTINUA
FACULTAD DE INGENIERIA U.N.A.M.

MICROPROCESADORES

Y MICROCOMPUTADORAS

M.EN I.LUIS PEÑARRIETA ECHENIQUE
ING.HÉCTOR CALVARIO MARTÍNEZ
ING.ROLANDO SAMUEL CARRERA SÁNCHEZ
ING.OCTAVIO OROZCO Y OROZCO
ING.DAVID BETANCOURT

O C T U B R E, 1984

Indice

Lista de Figuras

Lista de Tablas

CAPITULO 1
DESARROLLO DE LA MICROCOMPUTACION

1.1. EVOLUCION DE LOS MICROPROCESADORES

En 1947 se inventó el transistor de punto de contacto en Bell Telephone Laboratories, sus inventores fueron los físicos William Shockley, John Bardeen y Walter H. Brattain, los cuales recibieron el premio Nobel de física en 1956 por este descubrimiento. La prensa tomo con indiferencia la noticia, el New York Times dedico escasos 4 parrafos al dia siguiente de la presentación hecha por Bell Labs. en la antepenultima página del periódico en la columna denominada "Las nuevas de la radio". El dia de la presentación del transistor fue el 30 de junio de 1948.

El nombre del transistor deriva de la dualidad que existe entre el dispositivo descubierto y el tubo de vacio. El parámetro importante en el tubo de vacio es la transconductancia, mientras que en el nuevo dispositivo era la transresistencia, por lo que John R. Pierce sugirio "transistor".

William Shockley dejo Bell Labs. en 1954 para instalar su propia compañia, Shockley Semiconductor Lab. en Palo Alto Calif. Entre la gente joven con mucho talento que contrató figuraban: Eugene Kliner, Jay Last, Victor Grinish, Jean Hoerni, Sheldon Roberts Julius Blank, Gordon E. More y Robert Noyce. Estos 8 jovenes, desconformes con el ambiente de laboratorio que reinaba en la compañia y motivados por los ofrecimientos de Fairchild Co., se retiraron y formaron la subsidiaria Fairchild Semiconductor en septiembre de 1957. Esta compañia instaló su planta, también en Palo Alto. El primer producto que sacaron al mercado fue uno que Shockley habia pensado antes, el transistor de difusión por base, basado en el proceso mesa. Fue una muy muy buena decisión, porque la mayoría de los procesos subsiguientes fueron a base de difusión. Posteriormente cambiaron al proceso planar haciendo bastantes mejoras sobre el proceso mesa. Noyce se especializó en como colocar varios transistores en un solo chip y asi sucesivamente fueron produciendo artículos cada vez más atractivos.

En 1967, cuando Fairchild Semiconductors era una de las más importantes compañias de semiconductores se vio seriamente afectada por el éxodo masivo de ejecutivos de alto nivel a National Semiconductors, entre ellos se fue el gerente general y 4 directores de división, lo cual desmanteló practicamente la cabeza de la compañia. Ante esta catastrofe Fairchild Co. nombró ejecutivos muy jóvenes, los cuales ampliaron el clima de inconformidad en el resto del personal. Noyce y Moore ante el panorama tan desalentador que se presentaba, decidieron separarse

de Fairchild y formar su propia compañia que la llamaron Intel Corporation, llevandose algunos colaboradores muy cercanos.

A finales de los sesentas el negocio de las calculadoras electrónicas estaba tomando matices dramáticos, la competencia era tan grande que los fabricantes de calculadoras deseaban obtener la exclusividad de los nuevos chips, en cambio los fabricantes de cnips de calculadoras trataban de no quedar sujetos a un solo tipo de mercado o comprador, por el contrario ellos proponian arquitecturas de calculadoras modulares en donde las partes (chips) de las mismas pudieran ser atractivas a más de un fabricante de calculadoras.

1.1.1. MICROS DE 4 BITS

En agosto de 1969 Intel obtuvo un contrato con Busicom Co., una empresa japonesa dedicada a la fabricación de calculadoras para diseñar un conjunto de chips de calculadoras con caracteristicas muy especiales, Marcian Hoff fue asignado al proyecto el cual estudio los requerimientos de Busicom Co., sus conclusiones fueron que el proyecto tal como estaba presentado era demaciado complejo requiriendo gran cantidad de logica discreta y chips de 30 a 40 patas, tecnologia que en esa epoca no estaba al alacance; por lo que propuso un enfoque mas general, una calculadora de programa almacenado en ROM, la cual podría utilizar una secuencia de instrucciones mas generales no solo aritméticas muy útiles para manejar el teclado, despliegue de la información, impresion de los resultados, etc. Busicom aprobo la propuesta de Intel y de inmediato se dedicaron a afinar las caracteristicas del diseño. El conjunto de chips llamados 4004 fue diseñado por Federico Faggin, actual presidente de Zilog Co., los cuales se componian de 3 chips basicos, el CPU, ROM de 256 bytes y RAM junto con puertos paralelos y un registro de corrimiento. El diseño y desarrollo de estos chips fue realizado en forma conjunta entre Intel y un equipo de personas de Busicom en el valle del silicio (California), entre los japoneses que llegaron sobresalio Masatosni Snima diseñador posteriormente del 8080 de Intel, Z80 y Z8000 de Zilog.

El conjunto de chips 4004 de tecnologia PMOS fue completado en 1971, año en que se inicio su producción masiva, sin embargo, estos chips podian ser vendidos unicamente, debido a las restricciones del contrato, a Busicom. Para entonces, la competencia en el mercado de las calculadoras era sorprendente, por lo que Busicom aunque tenia la exclusividad de los 4004, los compraba muy caros, motivo por el que propuso a Intel reducir los costos a cambio le permitia vender estos chips a clientes con aplicaciones distintas de las calculadoras. De esta manera fue como Intel lanzó su primer anuncio, en el "Electronic News" del 15 de noviembre de 1971, sobre microcomputadoras programables en

un chip. El principal chip del conjunto 4004 fue el CPU con un conjunto de 45 instrucciones, el chip tenia alrededor de 2000 transistores integrados en una sola pastilla al cual se le podia añadir 4K bytes de ROM y 4120 bits de RAM.

El principal rango de aplicación de estos chips, aparte de las calculadoras fue sustituir lógica discreta por lógica programable, sin embargo para llegar a esta conclusión hubo que pasar antes por un estudio de mercado de las minicomputadoras, ya que se pensó en principio que estos podían sustituir a las mismas en aplicaciones de pequeña escala, alrededor de un 10% del mercado de las minicomputadoras lo que representaba 20,000 unidades al año. La Dirección de Intel no se convencio del exito del producto nasta que se contrato a Ed Gelbach ex director del departamento de Mercado de Texas Instruments, el cual propuso insertar inteligencia en muchos productos y equipo de medición y a la vez sustituir logica discreta por estos procesadores pequeños (microprocesadores), los cuales presentaban caracteísticas muy ventajosas en costo, flexibilidad, facilidad de diseño, reducción del consumo de potencia, facilidad de mantenimiento, etc.

A partir de entonces, la nueva propaganda de estos chips se enfoco nacia sustituir lógica discreta y aumentar flexibilidad e inteligencia en los nuevos equipos. Fue tal el éxito de este nuevo enfoque que ni el más óptimista esperaba que para febrero de 1972 ya hubiesen vendido 85,000 Sus. de este nuevo producto. Aunque el 4004 fue el primer producto en su clase, un año despues le siguieron otros microprocesadores, procedentes también de chips de calculadoras entre ellos destacan el PPS-4 de Rockwell anunciado a finales de 1972, el cual fue también de 4 bits, construido con PMOS, que tenia un conjunto de 50 instrucciones, un ciclo de instrucción de 5 microsegundos y un reloj de 0.2 MHz. En 1973 se anunciaron muchos otros microprocesadores entre ellos, el de Texas Instruments TMS 1000, Fairchild, National, Signetics, Tosniba, AMI y American Microsystems.

1.1.2. MICROPROCESADORES DE 8 BITS

Mientras Intel desarrollaba el sistema MCS-4, en paralelo tenia el proyecto de desarrollar un microprocesador de 8 bits, el cual sería, también, el primer dispositivo de 8 bits en el mercado, este chip se llamó 8008. Su origen data desde 1969 cuando Computer Terminals Corporation (ahora Datapoint), contrato a Intel para desarrollar circuitos integrados de alta escala de integración para su nueva terminal inteligente Datapoint 2200. Intel propuso a CTC integrar un procesador completo en una sola pastilla, por lo que se definio un procesador de 8 bits y el diseño del chip fue encargado a Hal Feeney. Poco tiempo después CTC dio las especificaciones a Texas Instrument. el cual

desarrollo , un cnip de 8 bits de 212 x 224 mils (milesimas de pulgada).

El cnip fue demostrado a CTC en marzo de 1971, sin embargo, la terminal no utilizó este chip. Aunque el contrato con CTC nabia terminado, Feeney continuo con el proyecto 8008 y en abril de 1972 concluyó su trabajo, resultando un CPU de 8 bits paralelos con 45 instrucciónes orientadas nacia el manejo de cadenas de caracteres, tenia un ciclo de instrucción promedio de 30 microsegundos, 6 registros de 8 bits de aplicación múltiple. Este procesador fue integrado en una pastilla de 18 patas con tecnología PMOS. Para una aplicación típica requiere de cuando menos 20 chips adicionales para conectarlo con memoria y I/O, puede direccionar hasta 16 K bytes de memoria.

1.1.3. SEGUNDA GENERACION DE MICROPROCESADORES

De 1972 a 1976 muchos fabricantes desarrollaron microprocesadores, componentes de soporte, tarjetas, sistemas y software muy rapidamente con el fin de ganar un nicho dentro de este nuevo mercado. En julio de 1974, 19 microprocesadores ya estaban en el mercado o nabian sido anunciados, un año mas tarde el numero crecio a 40 y en 1976 a 54.

El 8080 de Intel anunciado en abril de 1974 se puede consideapar como el inicio de la segunda generacion de microprocesadores. Este dispositivo sucesor del 8008 y 4004, fue diseñado en base a la experiencia de estos, por lo que su velocidad, capacidad de operación y conjunto de instrucciones lo convirtieron como una norma o punto de referencia para el desarrollo de los siguientes microprocesadores. Fue tal la demanda por este procesador que nubo la necesidad de producirlo por segundas fuentes (otros fabricantes), se cuenta este micro como el que tiene mayor cantidad de segundas fuentes.

Algunas de las características sobresalientes que el 8080 tiene son: un ciclo de instrucción de 2 microsegundos, un conjunto de 30 instrucciones más que el 8008, puede direccionar hasta 64 K bytes directamente, el stack (pila) fue puesto en memoria, se quitaron las restricciones en los anillos anidados, se incrementó el número de puertos a 256, tiene la capacidad de ejecutar operaciones aritméticas en decimal o BCD, un mejor procesamiento de interrupciones, etc. Solo 6 cnips adicionales es necesario para tener un sistema utilizable. El diseño del 8080 está realizado con 5000 transistores en una pastilla de 40 patas.

La rápida aceptación y la increible demanda del 8080, motivaron a otros fabricantes a producir microprocesadores con mejores caracteísticas que posteriormente desplazaron prácticamente, al 8080 del mercado. Uno de estos micros es el

6800 de Motorola, el cual fue anunciado a la venta en 1974; este micro fue el primero en contar con una sola fuente de alimentación de 5 volts, por lo cual resultó muy popular.

Una nueva característica que trajo consigo la aparición de los microprocesadores fue la producción de chips de soporte para los micros, los cuales facilitan en gran medida la interfase de los micros con perifericos de almacenamiento y dispositivos de entrada y salida. Intel, Motorola y cada nuevo fabricante que lanzaba al mercado un nuevo micro, producia también toda una gama de chips adicionales que formaban una familia con características muy particulares. El objetivo de estos nuevos chips es sustituir logica discreta y reducir en lo posible el número de componentes para construir microcomputadoras y controladores inteligentes.

Federico Faggin uno de los principales promotores de los micros de 8 bits de la segunda generación y uno de los más importantes integrantes del proyecto 8080, se separó de Intel en 1974 y bajo el apoyo de Exxon formó la compañia Zilog, la cual entro de lleno al mercado de los microprocesadores con el Z80, el cual fue diseñado por Masatoshi Snima y un equipo de personas muy competente, muchos de los cuales procedian de Intel. El Z80 se anuncio a la venta en 1976 el cual agrupa las mejores características de los micros precedentes. Una muy importante desición a la hora de diseño del mismo fue mantener la compatibilidad de sus instrucciones e inclusive el código de operación del 8080 y además ampliar el conjunto de instrucciones a 158, con lo cual, heredabá automáticamente toda la gran cantidad de programas (software) escrito hasta ese momento para el 8080. Esta estrategia influenció en gran medida para que el Z80 ganará popularidad muy rápidamente y en poco tiempo se convirtiera en el microprocesador de 8 bits más usado, inclusive a la fecha, aún es el micro más popular dentro de los de 8 bits; existen varios fabricantes que también lo producen a nivel de segundas fuentes.

Algunos micros de mediados de los setentas fueron el PPS-8 de Rockwell que aparecio en junio de 1974, otros el 2650 de Signetics y SCAMP de National, ambos de 1975. Un micro importante también en esa epoca, por sus características muy atractivas fue el 6502 de MOS Tecnology que aparecio en 1975. Un micro paralelo al Z80 fue el 8085 de Intel que aparecio, también en 1976, manteniendo la misma compatibilidad con el 8080, sin embargo, sus características propias son de menor alcance que las del Z80, ambos micros tienen un bus (canal paralelo) de datos de 8 bits, sin embargo, internamente pueden procesar datos de 16 bits. Algunos micros que llegaron tarde a la repartición de usuarios y no tuvieron la aceptación que sus fabricantes desearon son el 9980 de Texas Instruments, 8088 de Intel (ahora iAPX 88), el 6809 de Motorola, etc.

A partir del 8080 la tecnología más usada en los micros fue NMOS, la cual ofrece buenas ventajas en densidad y velocidad. La tecnología usada inicialmente fue PMOS por su facilidad y dominio de la técnica, pero se descartó por su lentitud. La tecnología bipolar ofrece alta velocidad pero baja densidad, por lo que quedó relegada a procesadores del tipo "bit slice" (procesadores de ancho incrementable). La tecnología CMOS ofrece el bajo consumo de potencia, pero también baja densidad, el primer microprocesador construido con esta tecnología fue el 1801 de RCA en 1974, al cual posteriormente le siguio el 1802 con características superiores.

1.1.4. MICROCOMPUTADORAS EN UN SOLO CHIP

Se ha dado por llamar micros de una nueva generación a aquellos dispositivos que tienen integrados en un solo chip el CPU, memorias RAM y ROM y puertos de entrada/salida para el manejo de perifericos. Gary Boone y Micnael Cochran de Texas Instruments demostraron en 1971 la factibilidad de integrar en un solo chip toda la circuiteria esencial de una microcomputadora. Recibieron el patente por esto en 1978. Su trabajo lo culminaron con la familia TMS-1000, chips microcomputadoras de 4 bits que tuvieron un enorme exito en juegos, juguetes y aplicaciones de control de bajo nivel. El F8 de Fairchild es una microcomputadora de 2 chips, posteriormente le siguio el 3870 de Mostek que tuvo un gran exito.

En realidad el primer chip microcomputadora de 8 bits fue el 8048 de Intel puesto a la venta en 1976. Posteriormente se anuncio el 8748 el cual es un micro similar al 8048 con la diferencia que en lugar de ROM usa EPROM lo cual le da gran versatilidad al usuario para programar y reprogramar a voluntad sus aplicaciones. Motorola con el 6802, Rockwell y otros fabricantes pronto siguieron con micros similares.

1.1.5. MICROPROCESADORES ANALOGICOS

Muchos fabricantes desidieron integrar interfases analógicas con microprocesadores, Intel por ejemplo, saco primero el micro 8022 con un puerto analogico de entrada y otro de salida, pero el que tiene gran exito es el 2920 (procesador de señales analógicas), el cual tiene la capacidad de efectuar procesamiento digital en tiempo real de señales analógicas. Tiene un conjunto de instrucciones especial para el procesamiento de señales analógicas un ALU de 25 bits, la circuitería digital incluye EPROM para el almacenamiento de programas, RAM, reloj de tiempo real, el ALU y un escalador binario. La circuitería analógica consiste de 4 puertos de entrada analógicos, entrada y salida multiplexada, un mantenedor de nivel (sample and holds) de

entrada, conversores análogo/digital y digital/análogo, 8 puertos analógicos de salida y un mantenedor de nivel para la salida. En resumen este cnip provee capacidad de procesamiento digital de alta velocidad en medios ambientes analógicos.

1.1.6. MICROS DE 16 BITS

El primer micro de 16 bits integrado en un solo chip fue el PACE de National Semiconductors, aparecido en 1974, el cual resultó ser una versión integrada del procesador "bit-slice" IMP-16. El PACE fue de tecnología PMOS con 10 microsegundos de ciclo de instrucción empaquetado en un chip de 40 patas. Posteriormente National continuo con su Super PACE (procesador bipolar considerablemente mas rápido que el PACE original). Otro micro de 16 bits tempranero fue el 9900 de Texas Instruments aparecido en 1975, con un espacio de direccionamiento de 32 K bytes, posteriormente hubieron varias innovaciones de este micro como el 9980, 9981 y 9995, este último aparecido en 1980, finalmente los últimos son el 99110 y 99120. El micro CP1600 de General Instruments es también de los micros tempraneros de 16 bits, este aparecio en 1976. En esa epoca, también existian muchos micros con buses de 8 bits pero con capacidad interna para manejar datos de 16 bits.

1.1.7. TERCERA GENERACION DE MICROPROCESADORES

Los sorprendentes desarrollos de la tecnología de semiconductores han permitido colocar en un solo chip un microprocesador, al menos una orden de magnitud mayor tanto en rendimiento como en complejidad de circuitería que los previamente disponibles. Las intenciones de sus fabricantes fueron combinar las facilidades de desarrollo tecnológico con las técnicas modernas de las ciencias de la computación para obtener micros de 16 bits tan avanzados, que invadan áreas antes privilegiadas para los procesadores e las grandes computadoras. Se tomaron en consideración las facilidades necesarias en el hardware para montar sistemas operativos complejos de múltiples tareas y multiusuario, para desarrollar compiladores de lenguajes de alto nivel, para facilitar el uso de estos micros en medios ambientes de multiprcesamiento, sistemas de procesamiento distribuido, etc.

Los avances del dopado en plasma seco, reducción con rayos laser, transistores HMOS, (transistores NMOS de alta densidad) y las técnicas automatizadas ayudadas por computadora para el diseño de circuitos integrados VLSI provieron una base tecnológica firme para que los ingenieros de mercado tuvieran la suficiente libertad innovativa para diseñar micros fáciles de usar más confiables y más flexibles en sus aplicaciones mientras el rendimiento se incrementaba cada vez más.

Particular énfasis se ha puesto para obtener arquitecturas lo más regular posibles tanto en registros, instrucciones, modos de direccionamiento y tipos de datos; dos modos de operación con instrucciones privilegiadas para el sistema operativo, manejo sofisticado de interrupciones y traps y facilidades para compartir líneas con otros procesadores. De gran importancia para el programador de istemas son las características que algunos micros de 16 bits tienen para detectar la ocurrencia de errores en la programación (bugs). Estas facilidades se presentan a nivel de traps y existen recursos integrados para la depuración de programas que facilitan el seguimiento del mismo instrucción por instrucción. El manejo de memoria na sufrido cambios radicales, mientras que los micros de 8 bits tienen un espacio de direccionamiento directo de 64K bytes, los micros de 16 bits manejan un espacio segmentado de memoria virtula de varias decenas de mega bytes. Sin embargo, el manejo de memoria normalmente no está integrado en el mismo chip del procesador sino que lo realiza otro chip VLSI extra, especialmente diseñado para tal efecto. Este chip manejador de memoria tiene la capacidad de abortar instrucciones, proteger segmentos, checar por la existencia de memoria, etc., aspectos normalmente básicos para el uso de sistemas operativos de múltiple-tarea y multiusuario.

En 1978 fue cuando se entro a esta nueva era de micros de 16 bits, llamados micros de la tercera generación o de alto rendimiento. El primero en aparacer fue el 8086 (llamado ahora iAPX 86) de Intel, el cual ocupa un área de 51,000 mils cuadrados (milesimas de pulgada cuadrada) y contiene aproximadamente 29,000 transistores en un solo chip, posteriormente en 1979 aparecio el Z8000 de Zilog en dos versiones una no segmentada con capacidad de micro tradicional y la otra segmentada con capacidad muy por encima del 8086. En 1980 Motorola cambio de su tradicional familia 6800 a MC68000, el cual es también, un procesador de gran alcance, este procesador contiene aproximadamente 68,000 transistores. En 1981 aparecio la familia de micros de National Semiconductors el NS16008, NS16016 y NS16032, de los cuales este último tiene características muy sobresalientes. En 1982 Zilog anuncio el Z8003, el cual tiene el doble de velocidad que sus predecesores.

1.1.8. FUTUROS MICROS

Actualmente varias compañias están trabajando arduamente en producir sus futuros microprocesadores, los cuales en la mayoria de los casos serán de 32 bits. Estos micros que serán de la cuarta generación competirán directamente con los procesadores de las supercomputadoras, pero a un costo mucho menor. Existen ya algunos anuncios sobre estos micros, tal es el caso de Intel que desde nace más de un año anuncio su iAPX 432, el cual tendrá

integrados alrededor de 200,000 transistores, un direccionamiento directo de 16 Mbytes y un direccionamiento virtual de un trillon de bytes, podrá ejecutar dos millones de instrucciones por segundo cuando se use una configuración de múltiples procesadores, lo cual es comparable con una IBM 370/158. Dos chips componrán el procesador, el CPU y el procesador de I/O (para el manejo de periféricos). Otro procesador recientemente anunciado es el Z80,000 de Zilog, el cual también es de 32 bits con características de un super procesador, las aplicaciones de estos micros son el manejo de recursos y control de sistemas en línea, tales como bases de datos, redes de computadoras, etc.

1.2. DESARROLLO DE MICROCOMPUTADORAS

En 1972 Intel dedico gran parte de sus esfuersos a promover sus nuevos microprocesadores, tanto de 4 como de 8 bits, debido al esceptisismo de la epoca por el futuro de los micros. Las primeras tarjetas que fabricó fueron SIM4-01 y SIM4-02, las cuales las ofreció a sus clientes desde mayo de 1972. Estas tarjetas que usaban el 4004, eran prototipos para el entrenamiento de sus clientes en el manejo de esta nueva tecnología. Estas tarjetas contaban con un generador de reloj de dos fases, circuiteria de "reset" y prueba, interfase para teletipos ASR-33, PROM Y RAM.

Un problema grave por el que se enfrentarón las compañias de micros sobre todo Intel (pionera en la comercialización de los mismos), al principio de la era de los microprocesadores fue la prácticamente imposible posibilidad de conseguir programadores para que trabajen en una compañia de semiconductores en algo que más que procesador era un juguete. Aquella era la epoca de las grandes computadoras y muchos programadores sentian que perdian prestigio si no trabajaban en otra cosa que no fuera una gran computadora. Sin embargo, Intel dada la gran demanda pudo terminar en junio de 1972 un pequeño paquete de software para sus micros, consistente en un cross-ensamblador y un cross-emulador escrito en Fortran IV, que se vendia en cinta de papel perforada o tarjetas perforadas, para usarse en una macro o mini computadora, este paquete lo ofrecia gratis, también, a sus clientes cuyas ordenes eran superiores a los 20,000 dolares anuales. A finales de 1972, Intel saco el primer ensamblador en PROM para el 4004 8008 y el primer prototipo SIM8-01 con el nuevo micro 8008. Los sistemas de desarrollo Intellec 4 e Intellec 8 fueron ofrecidos en 1973, completos con cross-ensambladores y cross-simuladores. El lenguage macro-ensamblador PL/M basado en PL/1 de IBM fue ofrecido por Intel en 1973 en la forma de cross-compilador, pero en 1974 llego a quedar residente en los sistemas de desarrollo Intellec. Con este macro-ensamblador los diseñadores podían desarrollar software modular a través de la

generación de tablas de ligado, en módulos de código objeto relocalizable, de esta manera se podia producir un código más confiable, documentado y en un tiempo mucho menor que con el lenguaje ensamblador.

Los módulos ISIS e ICE fueron dos desarrollos muy importantes, ambos anunciados en 1975. El ISIS (Intel Systems Implementation Supervisor) incorpora recursos de programación modular, tales como macro-ensamblador, ligador, localizador, manejador de directorios y un editor de texto. El ICE (in circuit emulation) remplazó las necesidades de simulación con cross-compiladores y facilitó enormemente a los diseñadores depurar hardware y software concurrentememte.

El desarrollo de las herramientas de software realizadas en los cinco primeros años después de la invención del microprocesador hizo posible incrementar en un orden de magnitud la productividad del programador y a su vez del diseñador de hardware, por las facilidades para detectar fallas de hardware y localizar errores de software.

Microcomputadoras de varias tarjetas proliferaron enormemente en poco tiempo, sus principales fabricantes fueron MITS Altair, Micral, Pro-Log, etc. Digital Equipment Corporatión dio su serie LSI-11 a compañias OEM (integradores de sistemas) compatibles totalmente con la vasta cantidad de software generado por las minicomputadoras PDP-11. Lo mismo sucedio en Data General con su Micronova y en General Automation con su GA-16/110.

El uso de computadoras personales y de computadoras para pequeños negocios fue iniciado en 1975 por la microcomputadora Altair de MITS. El kit (partes listas para ensamblarse) se vendia por 395 dolares, lo cual facilito a muchas personas de tener una microcomputadora en su propia casa. Esta revolución de la computadora fue facilitada por la posibilidad de adquirir un sistema operativo como CP/M a muy bajo costo, por solo 70 dolares era posible adquirir no solo el sistema operativo, sino un ensamblador, un depurador dinámico y un editor. El sistema operativo CP/M, tan popular dentro de los micros de 8 bits, fue escrito por Gary Kildall de Digital Research en 1975. La estructura del bus usada en Altair llego a adoptarse ampliamente como una norma, este se conoce como el bus S-100. Recientemente este bus ha sido modificado, extendido a 16 bits y normalizado por la Sociedad de Computación de la IEEE, y actualmente se conoce como el bus IEEE-696.

Intel entró al mercado de las computadoras en una sola tarjeta en 1976 con su SBC 80/10 (Single Board Computer), la cual costaba en ese entonces 295 dolares y estaba basada en el 8080. Un año después lanzó al mercado una versión mejorada de la tarjeta que la llamó SBC 80/20 en la cual presentó la

arquitectura del sistema "Multibus", con el cual se pueden interconectar 16 SBC 80/20s. El Multibus ha sido extendido a procesadores de 16 bits y se normalizará como el bus IEEE-796.

CAPITULO 2
LOGICAS DE SEMICONDUCTORES

2.1. OPERACION DE LOS TRANSISTORES

En este capítulo se hara un breve repaso de la operación de los transistores bipolares y MOSFET sobre todo cuando trabajan en las zonas de saturación y corte, es decir cuando trabajan como switches. En la segunda parte del capítulo se analizarán las características de las distintas familias lógicas, sobre todo de las más utilizadas en la actualidad.

2.1.1. TRANSISTORES BIPOLARES

Los transistores bipolares tienen tres zonas de operación que son: la zona de saturación donde el transistor se dice que esta prendido, la zona activa, que es la zona donde el comportamiento del transistor es lineal, esta es la zona que importa para las aplicaciones analógicas y finalmente la zona de corte donde se dice que el transistor esta apagado. En aplicaciones digitales las zonas importantes son la de saturación y la de corte.

En la figura 2-1 se muestra un transistor NPN y otro PNP, algunas de las ecuaciones que rigen el comportamiento del transistor NPN son:

$$Ie = Ic + Ib$$

donde:

Ie = corriente de emisor
Ic = corriente de colector
Ib = corriente de base

$$Vce = Vcb + Vbe$$

donde:

Vce = voltaje entre colector y emisor
Vcb = voltaje entre colector y base
Vbe = voltaje entre base y emisor

Se denomina ganancia de corriente en DC a:

$$n(fe) = Ic / Ib$$

Las siguientes son algunas de las características del transistor cuando se encuentra en la región de saturación:

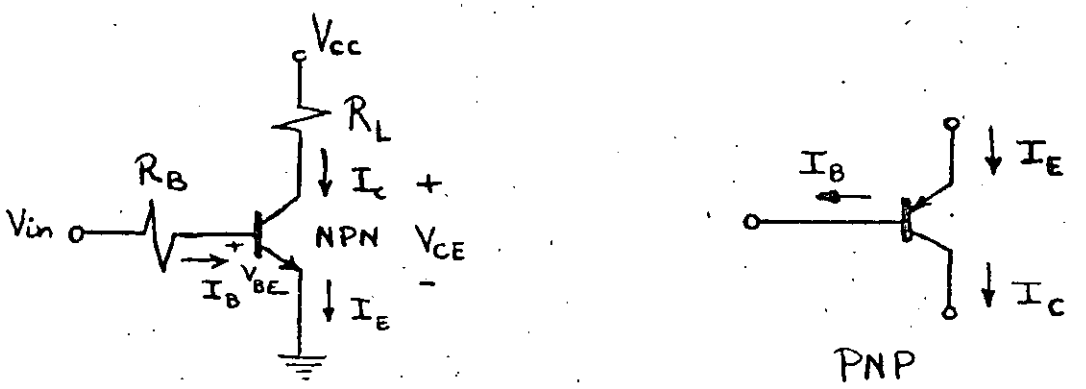


Figura 2-1: Transistores NPN y PNP

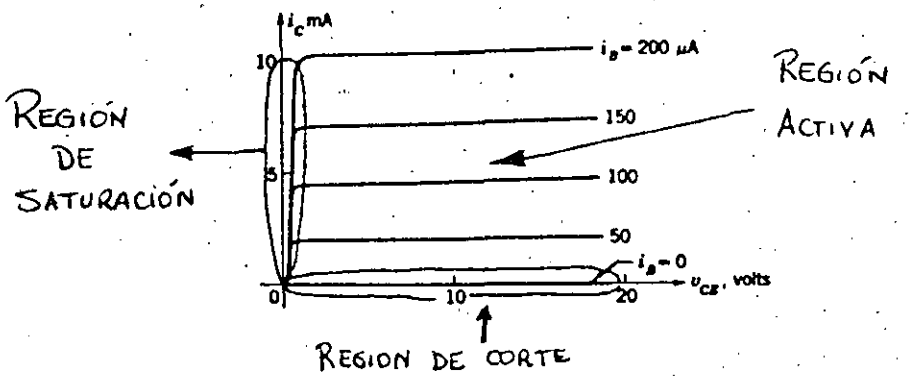


Figura 2-2: Regiones de operación del Transistor Bipolar

- Ib tiende a cero.
- Vce (sat) tiende a cero, generalmente tiene un valor de 0.2 volts.
- Vcb es menor que cero. Esta es la indicación más importante de que el transistor se encuentra en saturación.
- Vbe es aproximadamente igual a 0.75 volts en el caso de los transistores de silicio.
- Ic = (Vcc - Vce(sat))/Rl = (Vcc - 0.2)/Rl, o sea que Ic es casi igual a Vcc/Rl, es decir, la máxima corriente que se puede esperar de Ic.

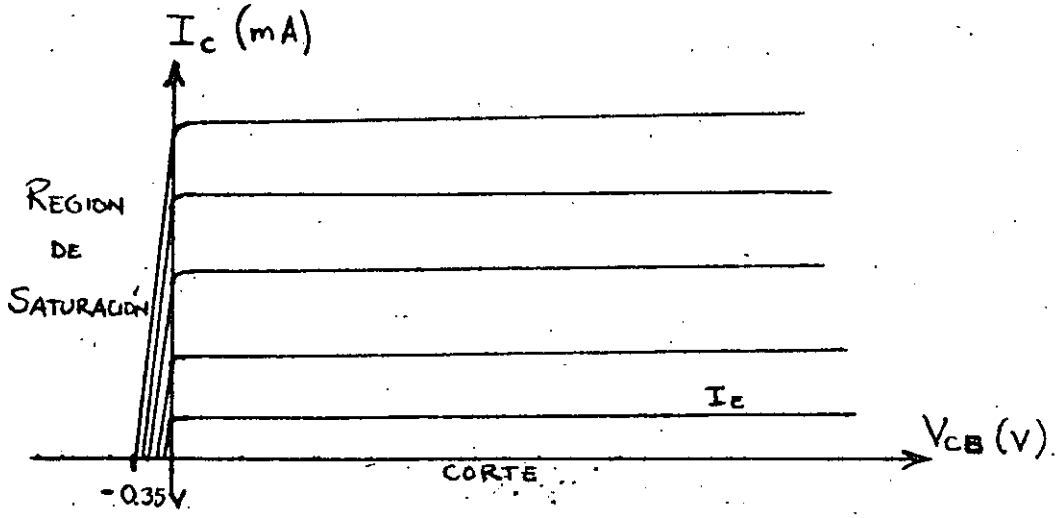


Figura 2-3: Región de saturación muy bien definida

Las siguientes son algunas de las características del transistor cuando opera en la región de corte:

- Ib es mucho mayor que cero.
- Vce es casi igual a Vcc.
- Ic es casi igual a cero.
- Vbe es menor de 0.65 volts en el caso de transistores de silicio.
- Vcb es aproximadamente igual a Vcc - Vbe.

En la figura 2-3 se muestra claramente la región de saturación, o sea es aquella donde las curvas tienen valores negativos para Vcb.

2.1.2. TRANSISTORES MOSFET

Existen dos tipos de transistores MOS, los NMOS en los cuales los portadores de carga son negativos, o sea los electrones y los PMOS donde los portadores de carga son positivos, en este caso los huecos.

Los transistores MOS tienen siempre tres terminaciones que son el "drain" o sumidero, el "source" o fuente y el "gate" o compuerta. Una característica muy importante en los MOS es que los portadores de carga siempre se mueven de la fuente hacia el sumidero. Los electrones tienen aproximadamente 3 veces más

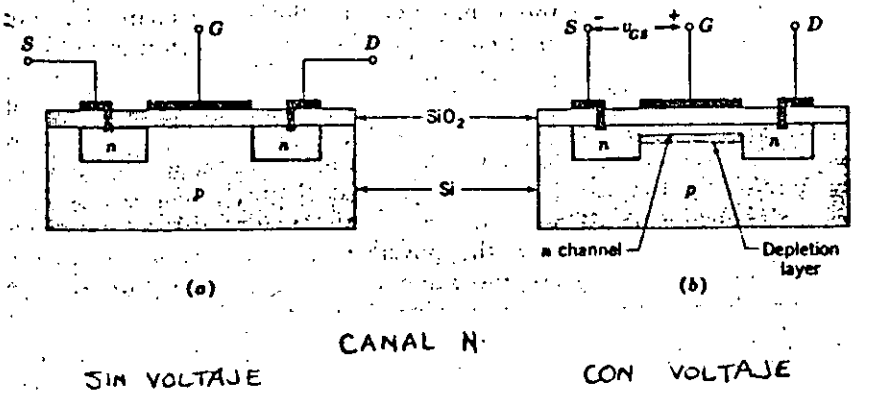


Figura 2-4: Estructura Básica del MOSFET

movilidad que los huecos por lo que los transistores NMOS son, entonces, tres veces más veloces que los PMOS. En el caso de los transistores MOS no es muy facil definir la región de saturación y la región de corte como en los transistores bipolares.

Se denomina V(T) al voltaje de umbral o voltaje de ruptura a partir del cual el transistor empieza a conducir. El voltaje de ruptura varia normalmente entre 2 y 5 volts. Si este voltaje es positivo (entre +2 y +5 volts) el transistor se denomina MOS enriquesido y si el voltaje es negativo (entre -5 y -2 volts) el transistor se denomina MOS empobrecido.

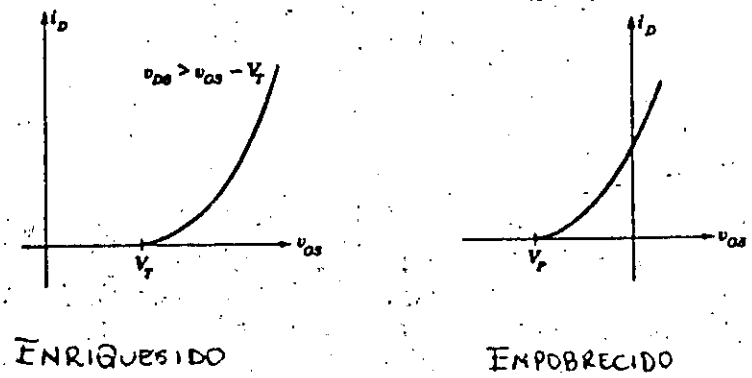


Figura 2-7: Voltajes de ruptura en MOSFETs enriquesidos y empobrecidos

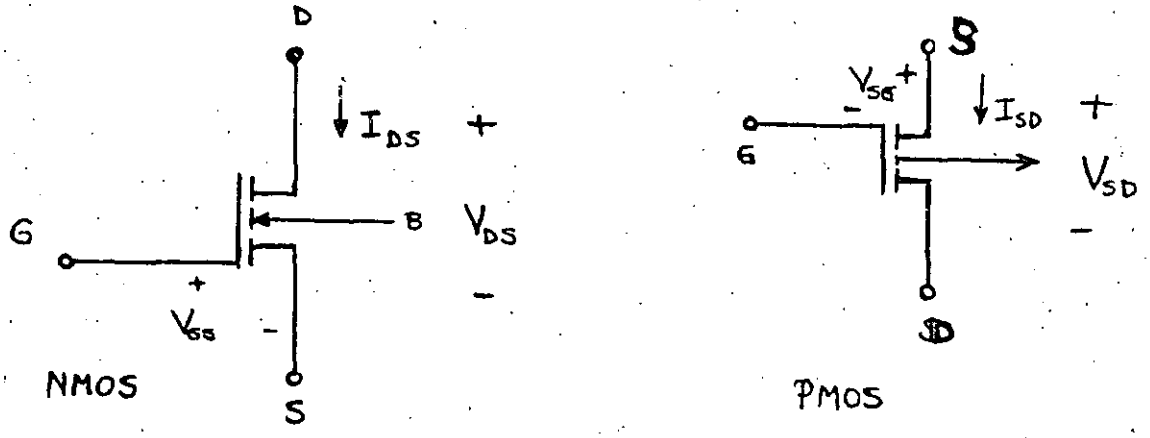


Figura 2-5: Transistores NMOS y PMOS

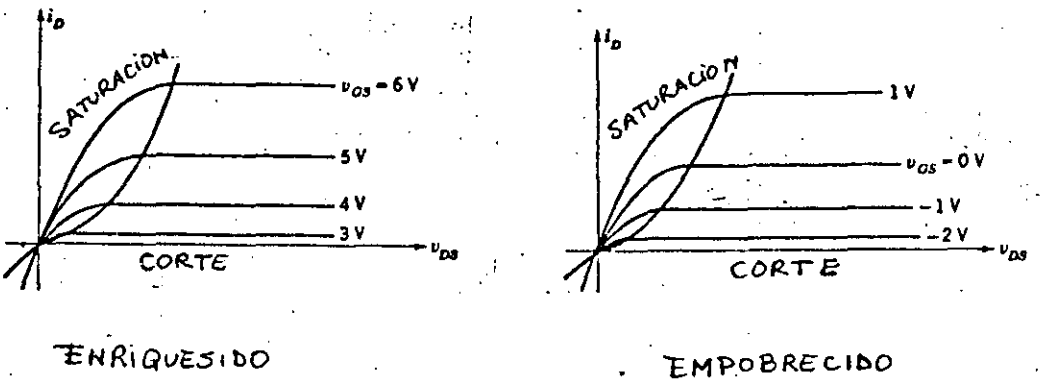


Figura 2-6: Regiones de operación de los Transistores MOSFET

El transistor MOS está en saturación cuando:

$$Vds >= Vgs - V(T)$$

$$Ids = K(Vgs - V(T))^2$$

El transistor MOS está en corte cuando:

$$Vds <= Vgs - V(T)$$

$$Ids = K[2(Vgs - V(T))Vds - Vds^2]$$

donde:

$$K = (\mu e/2t)W/L$$

μ es la movilidad de los portadores en el canal

e es la constante dielectrica de la capa de oxido aislante
t es el grosor del oxido bajo la compuerta
W es el ancho del canal
L es la longuitud del canal

Para NMOS $\mu e/2t$ es aproximadamente 12 microamp./volts
Para PMOS $\mu e/2t$ es aproximadamente 4 microamp./volts

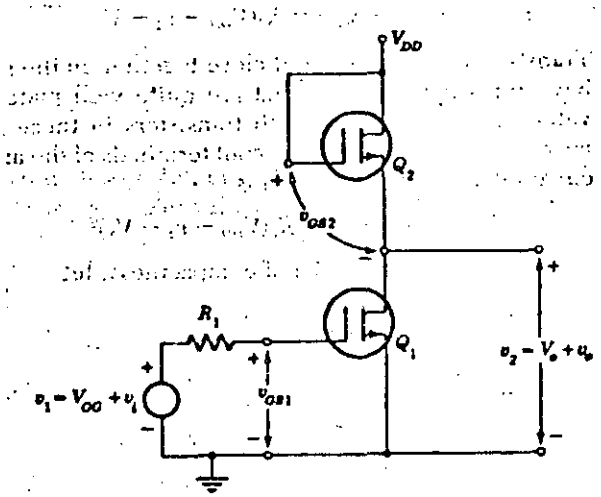


Figura 2-8: El Inversor MOSFET

Para construir un inversor con transistores MOS se requieren dos transistores uno llamado manejador "driver" y otro carga "load". Es recomendable que el transistor de carga sea enriquesido para que cuando la entrada sea cero el transistor este apagado. El transistor de carga debe ser de baja conductancia. El voltaje V(GG) debe ser mayor que V(DD) para que el nivel de salida en alto tienda a ser igual a V(DD). En las compuertas integradas MOS no se usa una resistencia como carga en virtud de que se requieren cargas del orden de 100 Kohms y es más dificil construir una resistencia de esta naturaleza que un transistor con carga equivalente. Por ejemplo una resistencia de 20 Kohms ocupa un área de 20 mils al cuadrado, en cambio un transistor MOS de una carga equivalente a 100 Kohms ocupa un área de 2 mils cuadrados.

El valor W/L para los transistores de carga debe ser de 0.1, mientras que el valor W/L para los transistores manejadores debe estar entre 20 y 40. Esto quiere decir que en el caso de los transistores manejadores el ancho del canal debe ser mucho mayor que su longuitud. En la figura 2-9 se observa que mientras mayor

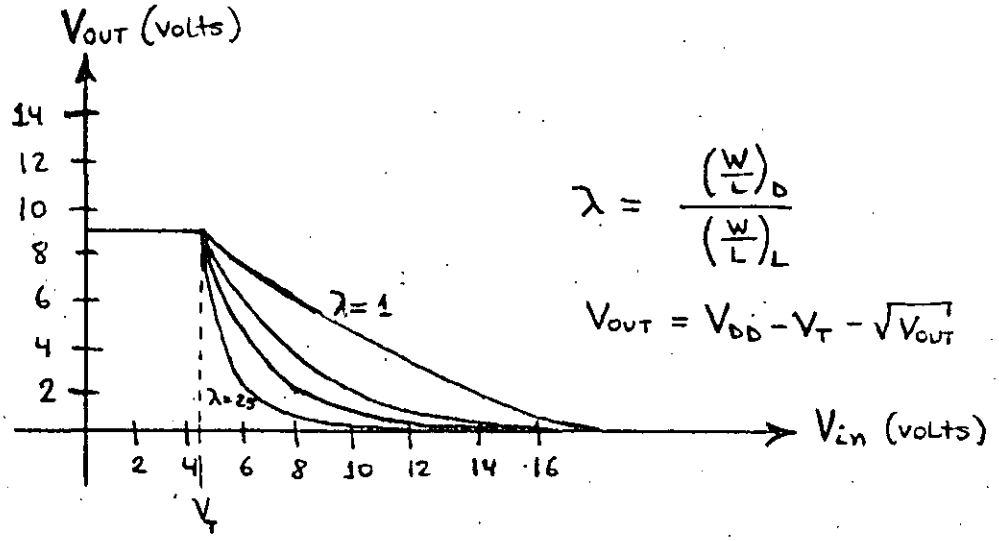


Figura 2-9: Curva de respuesta del Transistor MOSFET

sea el valor de gama es mejor porque el cambio de estado en la señal de salida será más abrupto.

2.1.3. TRANSISTORES CMOS

Los inversores CMOS se componen de 2 transistores complementarios uno es PMOS y el otro es NMOS, ambos enriquesidos.

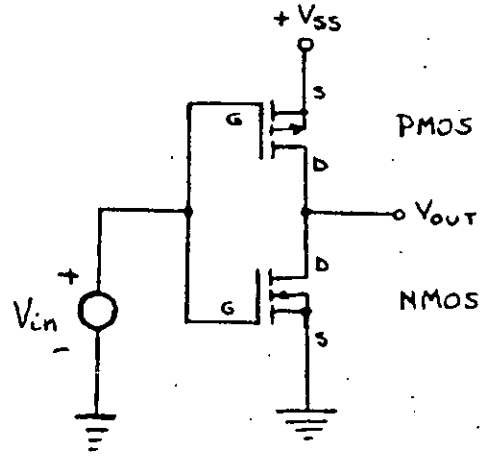


Figura 2-10: El Inversor CMOS

Si el voltaje de entrada es cero el NMOS está apagado (Vgs < V(T)) y el PMOS prendido (Vsg > V(T)), en este caso el voltaje de salida es V(SS). Si el voltaje de entrada es úno V(SS), el NMOS está prendido (Vgs > V(T)) mientras que el PMOS está apagado (Vsg < V(T)), y el voltaje de salida es igual a cero (máximo 10 milivolts). Esto significa que siempre que la señal de entrada se encuentre en alguno de los dos estados, uno de los transistores está prendido y el otro apagado. El transistor que esté apagado ocaciona que no se condusca corriente, por lo que no hay consumo de corriente de la fuente.

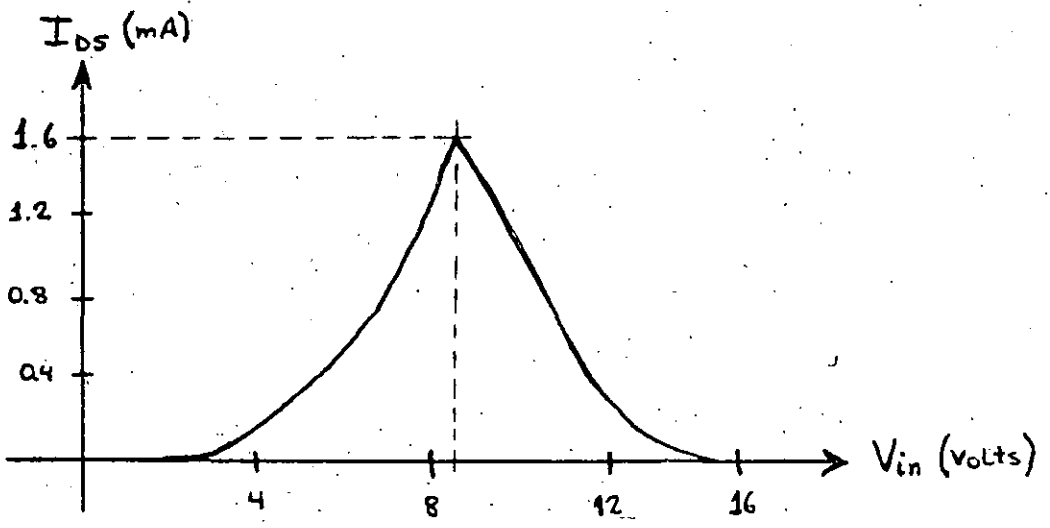


Figura 2-11: Consumo de corriente del CMOS

Solo hay consumo de corriente cuando se está en la transición de uno a otro estado y no así cuando la salida está estable en cualquiera de los dos estados.

2.2. FAMILIAS LOGICAS

Desde la integración de varios transistores en un solo chip, las compañias de semiconductores se han dedicado a elaborar familias lógicas, buscando siempre, la comfiabilidad, facilidad de manejo y una serie de características que las hagan atractivas, tanto para la producción como para la utilidad de las mismas en diferentes aplicacines. Las primeras familias, desde luego, no resultaron con características sobresalientes, por lo que se usaron pocc, por el contrario los investigadores de estas compañias seguian buscando familias lógicas con mejores características. Las primeras familias fueron RTL (lógica de resistencia transistor), DTL (lógica de diodo transistor), para

los casos de lógicas bipolares. En este capítulo se analizarán las características de algunas familias lógicas muy utilizadas en la actualidad.

2.2.1. TTL

James L. Buie fue uno de los diseñadores y el que recibio la patente por el acoplamiento entre etapas a través de transistores, que posteriormente se llamó lógica TTL. Este invento tan trascendental se produjo en una pequeña compañia instalada en Los Angeles Ca. llamada Pacific Semiconductors Co. que posteriormente fue adquirida por TRW y se convirtio en la División de Semiconductores de TRW. Los diseños iniciales los realizaron en 1961, pero fue en la segunda mitad de los sesentas cuando la versión estandar de esta familia se llegó a consolidar totalmente. Muchos fabricantes optarón por esta familia y pronto aprendierón a producir estos chips con un alto grado de eficiencia a un costo muy bajo. La familia TTL estandar se dividió en dos grandes ramas: la comercial, denominada 7400 con un rango de temperatura de 0 a 70 grados centigrados y la militar denominada 5400 con un rango de temperatura de 0 a 125 grados centigrados.

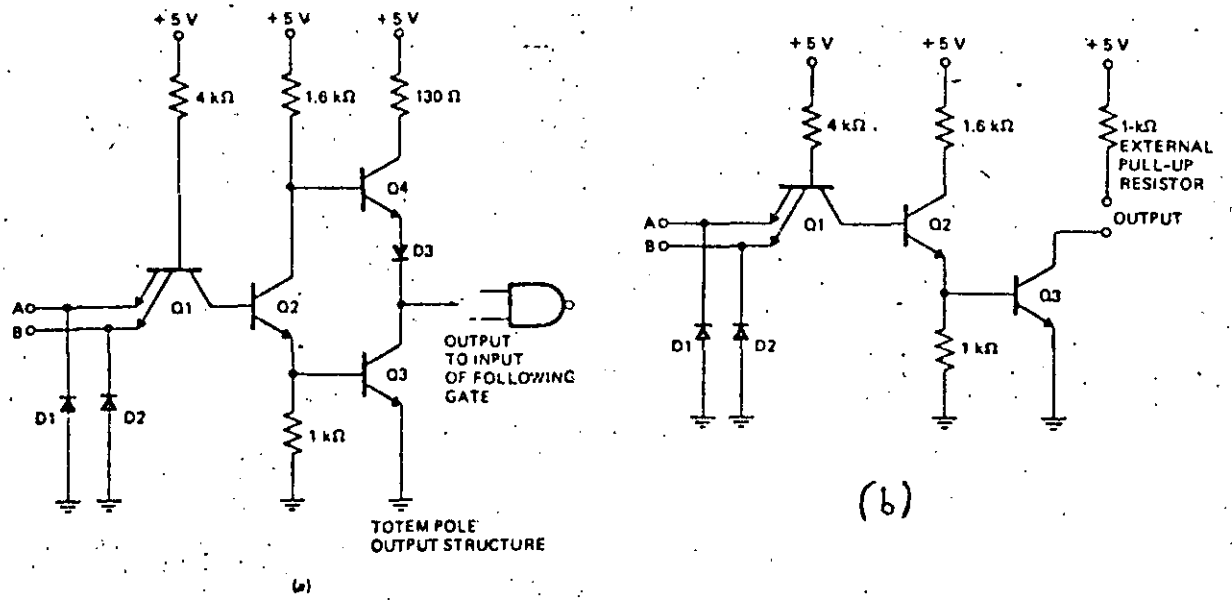


Figura 2-12: Compuertas NAND TTL estandar:
(a) Salida "totem pole"
(b) Salida de colector abierto

La subfamilia TTL estandar presenta, sin embrgo, algunas deficiencias sobre todo en velocidad de transferencia, que es relativamente lenta, y en el alto consumo de potencia, pese a que es la más barata de todas. Debido a estas deficiencias se optó por producir nuevas subfamilias mejorando en cada caso alguno de esos aspectos.

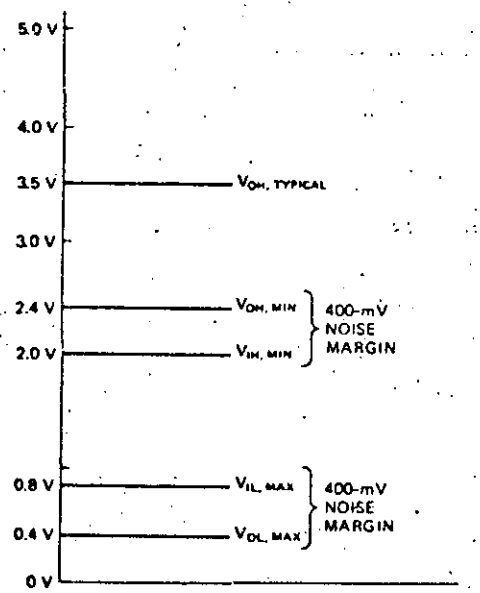


Figura 2-13: Niveles de voltaje del TTL estandar

2.2.1.1. SUBFAMILIA TTL DE ALTA VELOCIDAD "H"

La subfamilia (H) 74H00/54H00, tiene una alta velocidad de propagación, sin embargo, el consumo de potencia también es muy alto. El factor de carga (la corriente que demandan las señales de entrada) es 1.25 veces más que la subfamilia TTL estandar. Esta subfamilia es más cara que la estandar y que la (LS).

2.2.1.2. SUBFAMILIA TTL DE BAJO CONSUMO DE POTENCIA "L"

La subfamilia (L) 74L00/54L00 tiene un muy bajo consumo de potencia, sin embargo, es muy lenta en la velocidad de transferencia. Otra ventaja es el factor de carga que es muy bajo en comparación con la subfamilia estandar 0.25 de esta. Esta subfamilia es más cara que la estandar y que la (LS). Esta subfamilia es la más apropiada para interfasear TTL con circuitos MOS, por su muy baja demanda de energía en las señales de entrada.

2.2.1.3. SUBFAMILIA TTL SCHOTTKY "S"

Posteriormente a las dos subfamilias anteriores, se inventó el diodo Schottky que se conecta entre colector y base de los transistores; esta técnica mejora considerablemente la velocidad inclusive es mayor que la subfamilia (H) y de menor consumo de potencia. Esta subfamilia se denominó Schottky TTL en honor a su inventor y se conoce como la subfamilia (S) 74S00/54S00. El factor de carga es 1.25 veces más que la subfamilia estandar. Esta subfamilia es la más cara de todas, por lo que resulta no popular.

2.2.1.4. SUBFAMILIA TTL SCHOTTKY DE BAJO CONSUMO DE POTENCIA "LS"

Finalmente, se modificó la subfamilia (S) para producir la (LS), en la cual se sacrifica un poco la velocidad de la (S) pero se reduce enormemente el consumo de potencia; la velocidad de propagación de esta subfamilia es ligeramente mayor que la estandar y el consumo de potencia es mucho menor. El factor de carga es la mitad del correspondiente a la subfamilia estandar. Lo cual la hace atractiva para conectarse con salidas de circuitos MOS. Esta subfamilia es un poco más cara que la estandar, pero debido a sus ventajas en el consumo de potencia que es solo una quinta parte del consumo de la estandar, se ha convertido en la subfamilia más utilizada.

Las siguientes son algunas recomendaciones para trabajar con compuertas TTL:

1. Las entradas que no se usen deben conectarse a tierra directamente o a Vcc a través de una resistencia de 1 K ohm, según sea el caso que convenga.

2. Las salidas de compuertas TTL no deben conectarse entre si a menos que sean salidas de colector abierto o salidas de tres estados.

3. El voltaje máximo que se debe aplicar a Vcc es 7 volts.

4. El voltaje máximo de una señal de entrada es 5.5 volts teniendo Vcc a 5 volts.

5. Existe un máximo "fan-out" (número de entradas que puede manejar una salida) de 10 en cada subfamilia TTL.

6. Se debe instalar un capacitor de 0.1 microfaratn o 0.01 microfaratn entre Vcc y tierra cada grupo de 5 chips que se usen.

7. Las distancias entre las conecciones no deben ser mayores 35 cms. para el caso de la subfamilia normal y 10 cms. para la subfamilia (S).

2.2.2. ECL

La familia ECL (lógica acoplada por emisor) es muy distinta de la familia TTL, su principal característica es un muy pequeño retardo de propagación. Los chips típicos de esta familia son los MECL 10,000, los cuales utilizan una alimentación de voltaje de -5.2 volts, esto hace que sea muy dificil conectarlos con compuertas TTL.

La compuerta más simple de esta familia es la OR-NOR, en lugar de la NAND que es para TTL. Esta familia tiene un gran "fan-out", del orden de 90 entradas por salida. Sus desventajas son:

- Muy alto consumo de potencia.
- Niveles de voltaje no compatibles con TTL.
- Transiciones muy rápidas en los tiempos de levantamiento, lo que ocaciona que cualquier conección un poco larga haga que se comporte como una línea de transmisión por las reflecciones que se sucitan. Para evitar las reflecciones hay que terminar estas pequeñas líneas con la impedancia característica de las mismas.

La familia ECL III es muy poderosa debido a la capacidad de operar a frecuencias muy altas, sin embargo, los problemas de reflecciones son dificiles de manejar. En 1968 Motorola saco la familia MECL 10,000 la cual mantiene bajos los retardos de propagación pero incrementa los tiempos de levantamiento de l

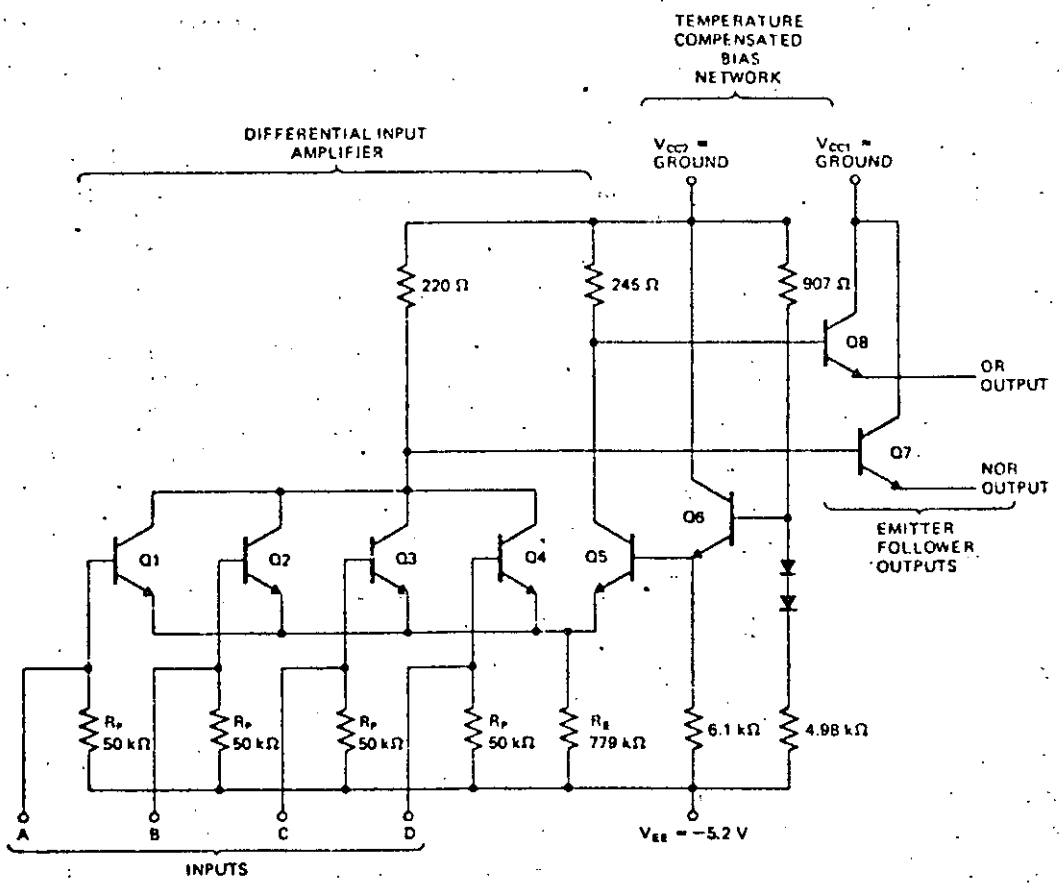


Figura 2-14: Compuerta elemental ECL

nseg. a 3.5 nseg., con esto es posible manejar lineas no terminadas nasta un máximo de 20 cms. Esta familia es muy usada en computadoras de muy alta velocidad e inlusive existen microprocesadores del tipo "bit slice" construidos con esta lógica.

2.2.3. IIL

Esta es otra familia lógica con transistores bipolares, la cual recibe el nombre de lógica de corriente de inyección o lógica de inyección integrada (IIL). Este tipo de logica no es usada para integrar compuertas discretas, como las anteriores dos familias, sino que se usa sobre todo para circuitos integrados que contienen miles de compuertas, tales como un reloj digital completo, o una calculadora en un solo chip, o un

microprocesador, por ejemplo, Texas Instrument tiene una versión del micro TMS9900 con esta lógica.

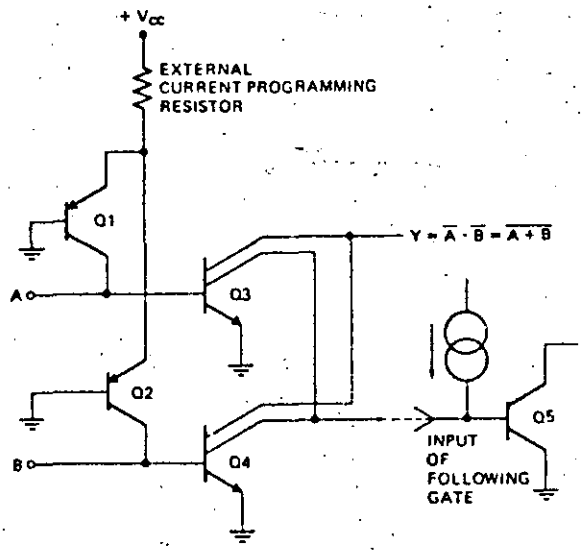


Figura 2-15: Compuerta IIL básica

Algunas ventajas de esta lógica son:

- La simplicidad de las compuertas y su bajo consumo de potencia hace posible integrar una gran cantidad de compuertas en una área muy pequeña.

- Las corrientes que se generan son constantes por lo que usualmente no existen las transientes en la línea de Vcc como sucede en TTL o CMOS.

- Debido a la facilidad de programar la corriente de inyección, se pueden variar los retardos de propagación de las señales y la disipación de potencia sobre un rango muy amplio. Por ejemplo, una corriente de inyección de 10 nanoamp. produce un retardo en la propagación de 100 microseg. y una corriente de inyección de 100 microamp. reduce el retardo en la propagacion a 25 nanoseg. Para aplicaciones lentas se puede reducir el consumo de potencia al mínimo.

- Otra ventaja de esta lógica es que puede quedar facilmente integrado en el mismo chip lógica digital

con circuitos analógicos bipolares tales como ampliticadores operacionales.

Una desventaja de esta lógica es que requiere un paso más en el proceso de manuractura que el MOS, el cual es definitivamente su principal competidor en el mercado de LSI (lógica de gran escala de integración).

2.2.4. FAMILIAS LOGICAS MOS

Los transistores MOS son de baja disipación de potencia y requieren muy pequeñas áreas en los chips, debido a esto las familias lógicas MOS son ampliamente usadas en muchos circuitos LSI tales como memorias, microprocesadores, etc. Existen muchas variaciones de la familia MOS, las más comunes son: PMOS, NMOS, CMOS, DMOS, HMOS, VMOS y SOSMOS. La subfamilia CMOS es la única en el grupo que se usa inclusive para fabricar chips con compuertas o funciones simples como en el caso de TTL.

2.2.4.1. PMOS

PMOS es la primer subfamilia que se ha producido en forma masiva, esta subfamilia usa los transistores MOS canal P en modo enriquesido para formar las compuertas. Las típicas fuentes de voltaje a tierra son -13 o -27 volts.

Las primeras versiones de los transistores PMOS tenian un alto voltaje de umbral, posteriormente, se desarrollaron transistores PMOS con voltajes de umbral relativamente bajos, entre 2 y 4 volts. Tradicionalmente se han necesitado acopladores especiales para conectar transistores MOS con lógica TTL, debido a las diferencias de nivel. Posteriormente, las versiones PMOS de bajo voltaje de umbral sustituyeron las compuertas de metal por compuertas de silicio para los transistores internos. Este enfoque mejoró la velocidad de propagación y proporcionó compatibilidad con TTL tanto a la entrada como a la salida. Las fuentes de voltaje típicas para chips PMOS con bajo voltaje de umbral pueden ser de los siguientes casos:

Vcc = 5 volts, Vdd = -5 volts y Vgg = -12 volts
Vcc = 5 volts, Vdd = -12 volts y Vgg = -12 volts

Este tipo de chips pueden manejar una carga TTL normal en sus salidas y sus entradas aceptan señales de nivel TTL.

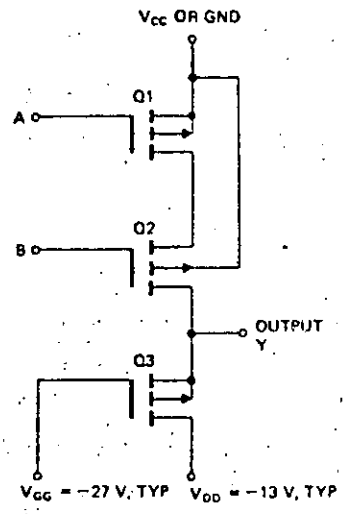


Figura 2-16: Compuerta PMOS básica

2.2.4.2. NMOS

Los chips PMOS fueron los primeros de la familia MOS en fabricarse debido a que el procesamiento del canal P tiene menos problemas de contaminación que el procesamiento de canal N. La tecnología NMOS provee mayor velocidad de propagación que la PMOS debido a que los portadores en NMOS son los electrones, los cuales tienen una movilidad 3 veces superior a los huecos, que son los portadores en los PMOS. Otra ventaja de los NMOS sobre los PMOS es que los chips resultantes ocupan menor área por transistor. Con todas estas ventajas los fabricantes de semiconductores cambiaron a NMOS tan pronto como el procesamiento de la tecnología lo permitio. La mayoría de las actuales memorias y microprocesadores MOS usan alguna variante de la tecnología MOS de canal N.

Entre los primeros circuitos NMOS de gran escala de integración (LSI) fue el microprocesador 8080A, el cual usa Vcc = 5 volts, Vbb = -5 volts y una fuente de alto voltaje Vdd = 12 volts con el fin de mejorar la velocidad interna de los circuitos y hacer que las salidas sean compatibles con señales TTL. Posteriormente salieron los primeros chips NMOS con una sola fuente de alimentación de 5 volts, tal es el caso de la memoria RAM estática de 1 K bit 2102, que usa una tecnología de compuertas de silicio y una estructura "push-pull" en las salidas

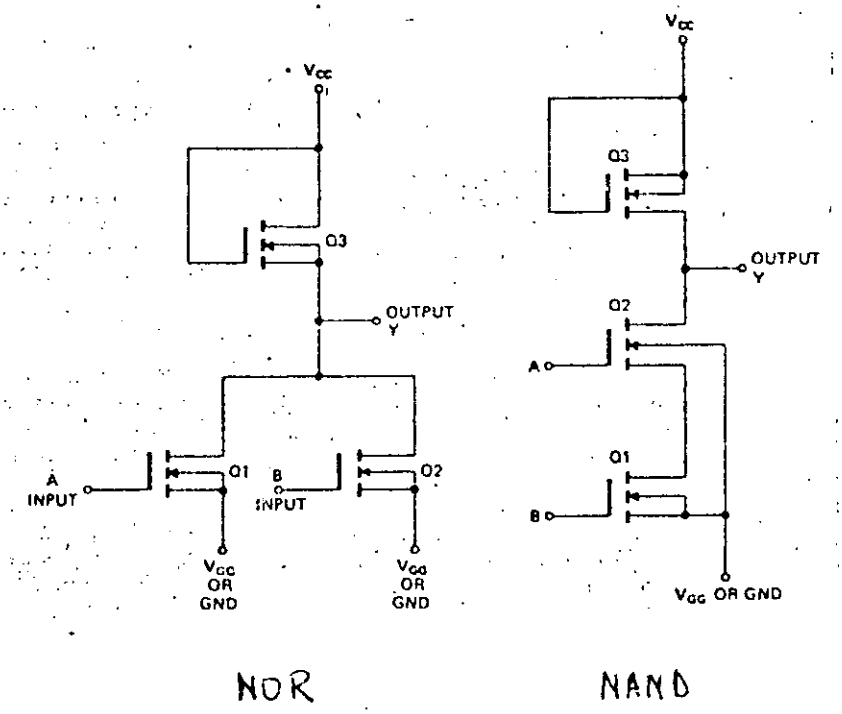


Figura 2-17: Compuertas NOR y NAND del tipo NMOS

con el fin de proveer suficiente corriente a las salidas para manejar cargas TTL normales, manteniendo en todos los casos una sola fuente de 5 volts. Las memorias RAM dinámicas NMOS fueron de las últimas en cambiar a una sola fuente de alimentación de 5 volts, debido a que el estado de las celdas lo representa la carga de un capacitor y mientras más alto sea el voltaje de carga menos tiempo se tardará en descargar. Las actuales memorias de 64 Kbits ya son de una sola fuente de alimentación de 5 volts.

2.2.4.3. VMOS, DMOS Y HMOS

VMOS, DMOS y HMOS son variaciones estructurales de la tecnología MOS de canal N, los cuales producen circuitos con mucho menor tiempo de propagación. Los transistores VMOS deben su nombre a la estructura en V que toman los mismos, y al hecho de que las corrientes fluyen verticalmente del "source" al "drain" y no horizontalmente como sucede en los demás NMOS.

Los transistores VMOS debido a su baja capacitancia y alta velocidad prometen como amplificadores de potencia para radio frecuencia, así como para circuitos de lógica LSI.

DMOS reduce la longitud efectiva del canal con el fin de reducir los tiempos de propagación, dosificando doblemente el dopado en la región del "gate".

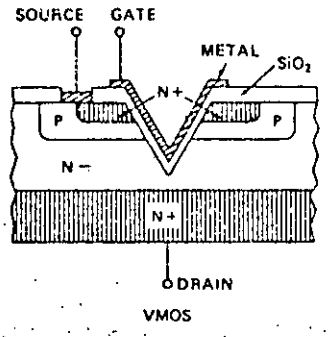


Figura 2-18: Transistor VMOS

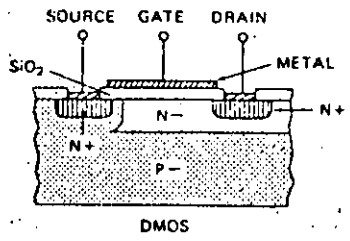


Figura 2-19: Transistor DMOS

HMOS o los MOS de alto rendimiento logran reducir los tiempos de propagación escalando hacia abajo en forma proporcional todas las dimenciones de los transistores NMOS del chip. La única dificultad con HMOS, al parecer, es que para lograr un óptimo rendimiento se requiere una fuente de voltaje menor que el estandart de 5 volts.

Estas tres variantes de lógica NMOS son capaces de lograr velocidades tipo ECL, mientras que requieren mucho menor potencia y área de chips que ECL.

2.2.4.4. MOS COMPLEMENTARIO "CMOS"

Casi al mismo tiempo que la tecnología de canal P habia sido desarrollada para circuitos LSI, la lógica CMOS o MOS complementario fue usada para producir una familia cuyas funciones se podían comparar con aquellas encontradas en TTL, pero con una disipación de potencia mucho menor. Los circuitos CMOS usan un área mucho mayor que el requerido para PMOS, sin embargo, son mucho más rápidos y tienen características de entrada y salida compatibles con la mayoría de las familias lógicas. Una propiedad interesante del CMOS es que se requiere una sola fuente de voltaje que puede tener cualquier valor entre 3 y 15 volts.

Existen cuatro series comunes de CMOS: la serie original 4000A, la serie mejorada de la anterior, conocida como 4000B, la serie de Fairchild 4500 y la serie de National Semiconductor 74C00, la cual es similar en características que la serie 4000B, pero tiene las mismas funciones lógicas y números de patas que los correspondientes chips TTL.

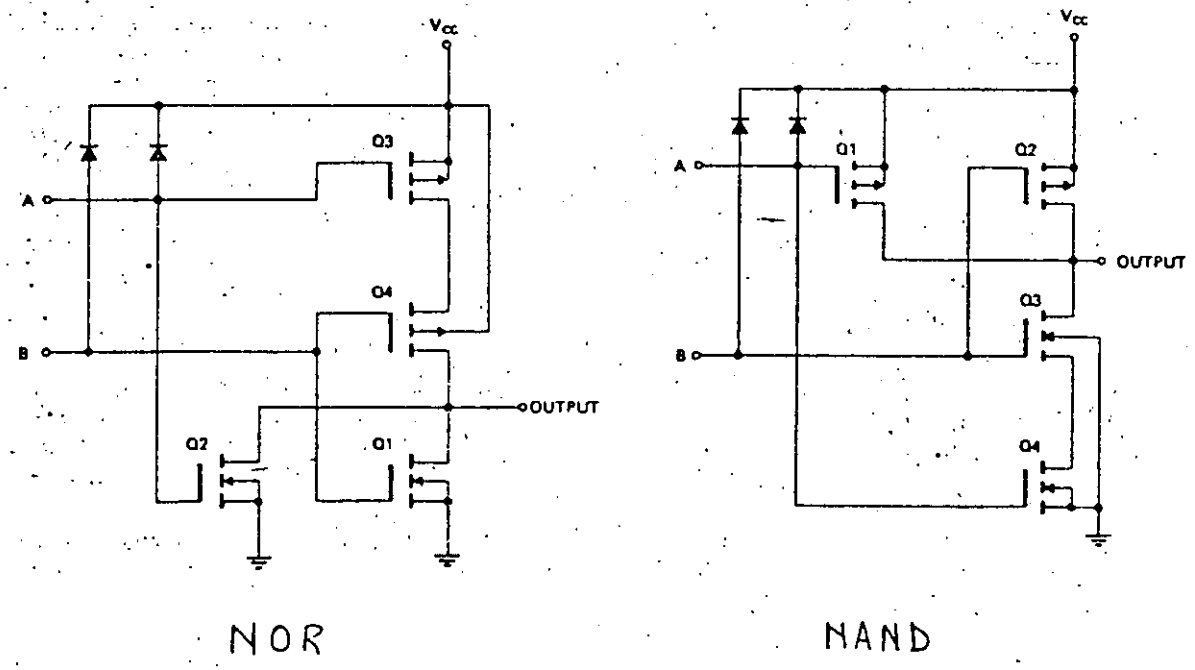


Figura 2-20: Compuertas NOR y NAND del tipo CMOS

La impedancia de entrada es muy alta porque, desde luego la entrada es un transistor MOS, por lo que el consumo de corriente a la entrada es de 10 pamp. tanto en estado alto como en bajo. En los circuitos integrados CMOS modernos es usual incluir en cada señal de entrada un diodo a Vcc como protección para prevenir

daños con cargas estáticas. La capacitancia total en las señales de entrada es de 5 a 7 picofaradios.

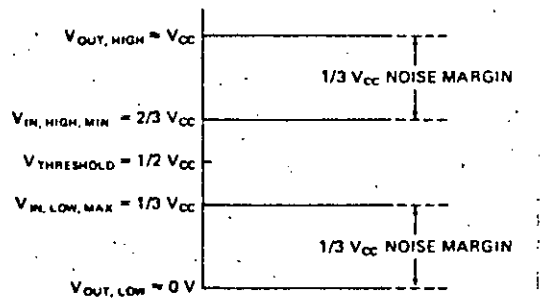


Figura 2-21: Niveles de voltaje y margenes de ruido en CMOS

El peor caso para el nivel de la señal de entrada en bajo es máximo 1/3 de Vcc y en alto es mínimo 2/3 de Vcc, el margen de ruido en ambos casos es 1/3 de Vcc. El peor caso para los niveles de las señales de salida es aproximadamente igual a Vcc menos 10 milivolts en estado altc ; 0 mas 10 milivolts en estado bajo. Por lo tanto esto proporciona un margen de señal a ruido muy grande, por lo que hace que esta familia lógica tenga una alta inmunidad al ruido electrico, pudiendo operar correctamente en medios ambientes ruidosos como plantas electricas, fabricas, etc.

Las compuertas CMOS operando con una fuente de alimentación de 5 volts tienen un tiempo de propagación de más de 100 nseg, lo cual limita su uso a aplicaciones lentas de pocos meganertz, sin embargo, al incrementar el voltaje de alimentación a 15 volts el tiempo de propagación decrece a menos de 100 nseg, lo cual permite que la lógica CMOS pueda ser usadaa en aplicaciones donde se requieren maayores frecuencias. Uno de los mayores problemas del CMOS es precisamente el tiempo de propagación que es directamente proporcional a la carga capacitiva de la salida, si esta carga es de 15 pf (picofaradios) equivale a tener 3 entradas conectadas a esta salida el tiempo de propagación es de 50 a 75 nseg mientras que si la carga se incrementa a 50 pf o equivalente a 10 entradas el tiempo de propagación puede subir hasta un máximo de 360 nseg. Estas variaciones tan grandes en los tiempos de propagación pueden causar serios problemas en algunos circuitos.

Una de las mayores ventajas del CMOS es su muy baja disipación de potencia cuando opera a bajas frecuencias. Una compuerta CMOS típica disipa solo 10 microwatts cuando está

operando a 1 Khz. con 5 volts de alimentación. Surge la duda de porque la frecuencia de operación está relacionada directamente con la disipación de potencia, esto se debe a que las compuertas CMOS no consumen prácticamente energía cuando la salida está en estado estable cualquiera que sea el nivel lógico cero o uno. Solo hay consumo de energía en la transición de los estados de cero a uno o de uno a cero, es por esto que a medida que se incremente el número de transiciones (aumente la frecuencia de operación) el consumo de energia también se incrementará. Esta característica del CMOS lo hace muy atractivo para usarse con pilas o baterias en aplicaciones de relativa baja frecuencia.

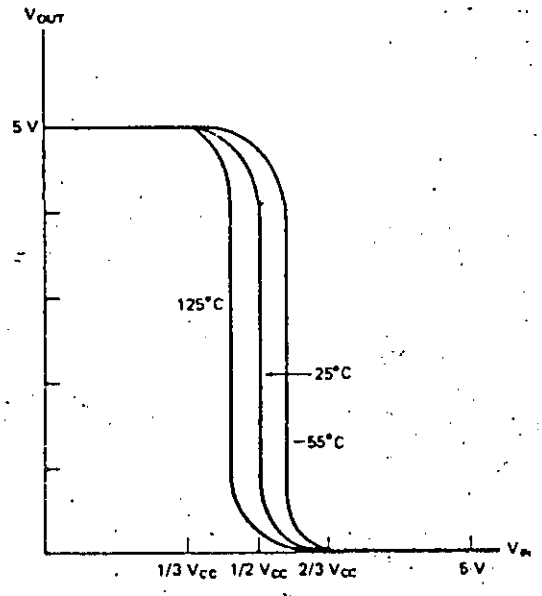


Figura 2-22: Curva de transferencia del CMOS. Inmunidad a la temperatura

Otra gran ventaja del CMOS es la inmunidad a la temperatura, el rango de temperatura para la operación del CMOS es muy grande. La curva de transferencia del inversor CMOS varia muy poco con respecto a los cambios de temperatura, aún considerando los límites militares de temperatura que son -55 y 125 grados centigrados, la curva de transferencia varia en forma insignificante entre esos dos extremos.

Una desventaja del CMOS es la complejidad del proceso tecnológico de fabricación que requiere más pasos que el proceso PMOS o NMOS, lo cual redunda en el costo naciendo que el CMOS resulte más caro que los dos anteriores.

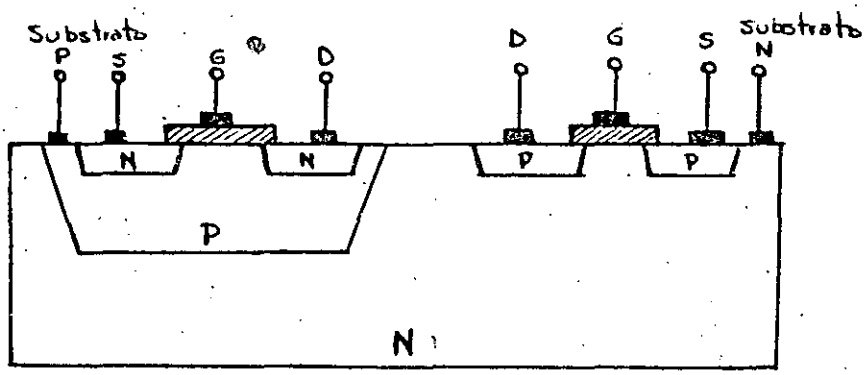


Figura 2-23: Estructura básica del CMOS

2.2.4.5. SOS-CMOS

Una versión de alto rendimiento del CMOS es el SOS-CMOS, la cual consiste en la construcción de transistores CMOS sobre una base o substrato de safiro (sapphire) en lugar del usual substrato de silicio. El substrato de safiro es un buen aislante cuyo efecto en los transistores es reducir la capacitancia que siempre se forma entre estos y el substrato. Además, aumenta considerablemente la frecuencia de operación del CMOS, pudiendose tener frecuencias de operación hasta de 50 MHz.

La tecnología SOS (significa silicio sobre safiro) es usada principalmente para circuitos LSI tales como memorias y microprocesadores.

2.3. COMPARACION ENTRE FAMILIAS LOGICAS

Es dificil, practicamente imposible, decir que familia lógica es mejor, puesto que cada una tiene características sobresalientes en algún aspecto, mientras que dejan mucho que desear en otro(s). La aplicación juega un papel importante en la desición de la familia lógica, ya que marca las politicas que se deben seguir para escoger a la misma. Por ejemplo, si se desea trabajar a muy alta velocidad se puede escoger ECL, o si lo que más importa es el costo puede ser TTL estandar, o bien, si el consumo de potencia es crítico, lo más lógico sería escoger CMOS o bien IIL, etc.

La figura 2-24 muestra las curvas típicas de disipación de potencia contra la frecuencia de entrada, se observa en la lógica

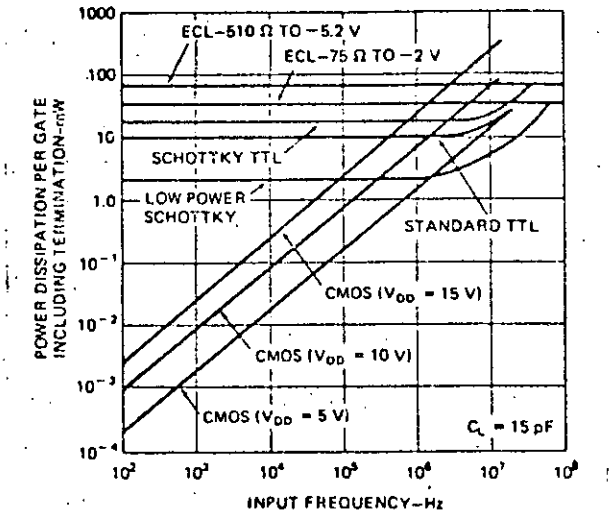


Figura 2-24: Típica disipación de potencia vs. frecuencia de la señal de entrada

TTL que el consumo de potencia es prácticamente independiente de la frecuencia de la señal de entrada, los cambios se presentan solo a muy alta frecuencia, cerca de los límites establecidos por la misma lógica. En cambio CMOS es una lógica que depende directamente de la frecuencia de la señal de entrada, con frecuencias arriba de 1 MHz. el consumo de potencia es comparable con la lógica TTL, sin embargo, a muy bajas frecuencias no consumen practicamente nada de energía.

La siguiente tabla proporciona una visión bastante clara sobre algunas características comunes de las lógicas comerciales más usadas en la actualidad. Puede ser una gran ayuda para decidir que lógica se debe emplear dependiendo de la aplicación.

Tabla 2-1 Comparación de características comunes en varias familias lógicas

LOGIC FAMILY	OPERATING SUPPLY VOLTAGE MIN	OPERATING SUPPLY VOLTAGE MAX	MIN. MUM LOGIC "1" INPUT	MAX. MUM LOGIC "0" INPUT	MIN. MUM LOGIC "1" OUTPUT	MAX. MUM LOGIC "0" OUTPUT	MAX. MUM LOGIC "0" INPUT CURRENT	MAX. MUM LOGIC "1" INPUT CURRENT	MAX. MUM LOGIC "0" OUTPUT CURRENT	MAX. MUM LOGIC "1" OUTPUT CURRENT	TYPICAL PROPAGATION DELAY (ns) C_L = 50 pF T_{PHL}	TYPICAL PROPAGATION DELAY (ns) C_L = 50 pF T_{PLH}	MAX. MUM COUNTER FRE- QUENCY	FANOUT	LOADING FACTOR	TYPICAL POWER DISSI- PATION PER GATE	TYPICAL t_r 10-90%	SPECIFIC DEVICE
7400	4 75	5 25	2 0	0 8	2 4	0 4	1 6 mA	40 µA	16 mA	400 µA	8 ns	13 ns	35 MHz	10	1	10 mW	15 ns	5400
S400	4 5	5 5																
74H	4 75	5 25	2 0	0 8	2 4	0 4	2 0 mA	50 µA	20 mA	500 µA	6 2 ns	5 9 ns	50 MHz	12-24 10-74H	1 25	22 mW	7 ns	74H00
74L	4 75	5 25	2 0	0 8	2 4	0 4	0 18 mA	10 µA	3 6 mA	200 µA	30 ns	60 ns	3 MHz	2-24 10-74L	0 25	1 mW	30 ns	74L00
74S	4 75	5 25	2 0	0 8	2 7	0 5	2 0 mA	50 µA	20 mA	1 mA	5 ns	5 ns	125 MHz	12-24 10-74S	1 25	19 mW	3 ns	74S00
74LS	4 75	5 25	2 0	0 8	2 7	0 4	0 36 mA	20 µA	4 mA	400 µA	8 ns	8 ns	45 MHz	2-24 10-74LS	0 5	2 mW	15 ns	74LS00
PECL IH OR MECL III	-4 7	-5 7	-0 980	-1 600	-0 960	-1 620	350 µA	350 µA	40 mA	40 mA	1 ns	1 ns	1000 MHz	DC — 63 MECL III	NOT TTL-COMPAT-IBLE	60 mW	1 ns	MC1662
MECL 10 000	-4 7	-5 7	-0 980	-1 630	-0 960	-1 650	0 5 µA	265 µA	50 mA	50 mA	2 ns	2 ns	200 MHz	DC — 92 MECL	NOT TTL-COMPAT-IBLE	25 mW	3 5 ns	MC10101
I^2L	1	15	0 7	0 4	V_{CC}	0 4	INJECTOR CURRENT	INJECTOR CURRENT	ZERO WITHOUT RESISTOR TO V_{CC}	20 mA	ADJUSTABLE 25 ns-250 ns		—	DEPENDS ON INJECTOR CURRENT	DEPENDS ON INJECTOR CURRENT	6 nW-70 µW	—	—
LOW THRESH-HOLD PMOS	V_{CC} = +5 V ±5% V_{DD} = -9 V ±5% $V_{DD} = V_{PO}$		3 0	0 65	3 5	V_{CC} - 6 V MOS OR 0 45	1 µA	1 µA	1 8 mA @ 0 45 V	0 2 mA @ 3 5 V			1 MHz	—	—	—	—	1702A ROM
+12, +5, -5 NMOS	V_{CC} = +5 V_{DD} = +12 V_{GG} = -5		3 3	0 8	3 7	0 45	±10 µA	±10 nA	1 9 mA	150 µA	—	—	—	—	—	—	—	8080A
NMOS +5 ONLY	4 5	5 5	2 0	0 8	2 4	0 4	10 µA	10 µA	2 1 mA	100 µA	—	—	—	—	1-24 5-74LS	—	—	2102 RAM
4000A +5 V CMOS	3	15	2/3 V_{CC}	1/3 V_{CC}	V_{CC} - 01	0 01	10 pA	10pA	12 mA @ 5 V	12 mA @ 4 5 V	50 pF 180 ns	50 pF 125 ns	1 MHz CD4040	DC = ∞ AC = 3 - 10 CMOS	DC = ∞ AC = 3 - 10 CMOS	DEPENDS ON OPERAT-ING FRE-QUENCY	175 ns	CD4011
4005B +5 V CMOS	3	18	2/3 V_{CC}	1/3 V_{CC}	V_{CC} - 01	0 01	±1 µA	±1 µA	0 4 mA @ 0 4 V	1 6 mA @ 2 5 V	50 pF 160-320 ns	50 pF 210-420 ns	2 5 MHz 4520	DC = ∞ AC = 3 - 10 CMOS	—		100 ns	CD40518
4030B + 15 V CMOS	3	18	2/3 V_{CC}	1/3 V_{CC}	V_{CC} - 01	0 01	±1 µA	±1 µA	3 mA @ 1 5 V	3 mA @ 13 5 V	50 ns	65 ns	6 MHz 10 V 4520	3-10 CMOS	—		40 ns	CD40818
74C00 +5 V CMOS	3	15	3 5	1 5	4 5	0 5	5 nA	5 nA	0 4 mA @ 0 4 V	0 36 mA @ 2 4 V	50 pF 90 ns	50 pF 90 ns	2 MHz 74C90	3-10 CMOS 2-74L00	—		100 ns	74C00

CAPITULO 3
MICROPROCESADORES

3.1. INTRODUCCION

Uno de los avances tecnológicos de mayor significancia de la década pasada fué el surgimiento de los circuitos con gran escala de integración LSI (Large Scale Integration). Los métodos de fabricación y la tecnologia apropiada permitieron la producción de circuitos muy complejos en una sola tableta de silicio empaquetados llamados "chips". La evolución en el campo de la lógica digitál fue a través de etapas de producción de subunidades lógicas estandares en circuitos integrados IC (Integrated Circuits). En la primera etapa surgieron las compuertas simples (v.g. and,or,inversores) y "flip-flops" en chips de pequeña escala de integración SSI (Small Scale Integration). De esta etapa surguieron los chips con mediana escala de integración MSI (Medium Scale Integration), los cuales contenian registros, contadores, codificadores, decodificadores,etc. El número de los diferentes elementos en un chip está determinado en gran manera por el número requerido de conecciones externas hacia el chip, por lo que es típico encontrar un multiplexor de ocho entradas y una salida o bien cuatro flip-flops en un chip de 16 patas.

Conforme aumentaba la habilidad para construir ICs con gran cantidad de elementos lógicos, se hizo ventajoso considerar circuitos que requirieran un gran número de elementos pero con pocas (relativamente) conecciones externas. El resultado fué la aparición de chips más complejos, tales como, unidades capaces de procesar funciones aritméticas y lógicas sobre datos codificados en cuatro dígitos binarios (bits) en paralelo. Estos pimeros dispositos fueron llamados microprocesadores debido a su relativa baja velocidad de procesamiento y su limitación para operar con datos de tamaño reducido en comparación con las computadoras de la década pasada (actualmente esta distinción es de poco valor). Pronto surgieron chips que operan con datos de 8 bits y mas recientemente de 16 bits.

En el año de 1971, la compañia Intel introdujo al mercado el microprocesador Intel 4004 (de cuatro bits), para un fabricante japonés de calculadoras, desde entonces los microprocesadores han sido motivo de gran uso para las industrias de la electrónica y de procesamiento de datos.

Fué cuando estos dispositivos maravillaron al mundo haciendo posible las calculadoras de bolsillo, que actualmente, son de uso corriente y común.

A pesar del alto precio de entonces (160 dolares), esos dispositivos fueron bien recibidos por el mundo de la electrónica digital. En 1975 cuando los precios se redujeron drásticamente, la popularidad y aplicaciones de los microprocesadores crecieron exponencialmente. Tan solo en un solo periodo de tres meses, los precios de los microprocesadores cayeron en un 50 a 70 por ciento, y no solo en grandes ordenes de cantidades sino también en compras unitarias.

Actualmente, los microprocesadores cuestan, algunos, poco menos de 20 dolares, otros en un rango de 10 dolares y otros cerca de los 5 dolares.

Hoy en dia, esos dispositivos, son usados en computadoras, dispositivos perifericos, automoviles, relojes, máquinas de juegos, sistémas de seguridad, hornos de micro-ondas, juegos de TV, juguetes, comunicaciones y en una amplia variedad de aplicaciones.

3.2. ORGANIZACION DE LAS COMPUTADORAS

Antes de describir y entender a los microprocesadores, es necesario conocer que son las computadoras, cuales son sus unidades funcionales, y en términos generales su funcionamiento.

3.2.1. DEFINICION DE COMPUTADORA

Una computadora digital o simplemente computadora en su mas simple forma, es una máquina electrónica capaz de realizar cálculos con gran rapidez, obedeciendo instrucciones muy específicas y elementales que reflejan su estructura funcional u organización.

Dentro de las computadoras existen dos grandes tipos; aquellas que realizan los cálculos de manera secuencial (tipo Von Neumann) a la cuál pertenecen la mayoria de las computadoras y aquellas que realizan procesos en paralelo, de concepción mas reciente y generalmente están en etapa de investigación y desarrollo.

Para los propósitos del curso nos referiremos a aquellas de tipo Von Neumann o de propósito general que son las comunmente utilizadas ya sea en ambientes cientificos o comerciales.

Con la palabra computadora abarcamos una gran cantidad de máquinas que difieren enormemente en costo, capacidad, velocidad, etc. Sin embargo es comun hablar de "micros", "minis", y máquinas grandes (mainframes). Tradicionalmente se han manejado los

términos capacidad, costo y velocidad del procesador para ubicar a un equipo de cómputo en alguna de las categorias antes mencionadas, sin embargo, actualmente es mas dificil establecer los límites y rangos de cada una de ellas (tal vez la mejor manera de saberlo es preguntar al constructor en que categoría coloca a su equipo).

Si bien existen desde un punto de vista de complejidad enormes diferencias y varaciones entre los equipos grandes y pequeños, funcional y conceptualente son iguales, y estas ideas son las que trataremos de explicar aquí.

3.2.2. UNIDADES FUNCIONALES DE UNA COMPUTADORA

Tradicionalmente se ha dividido a una computadora como se muestra en la figura 3-1.

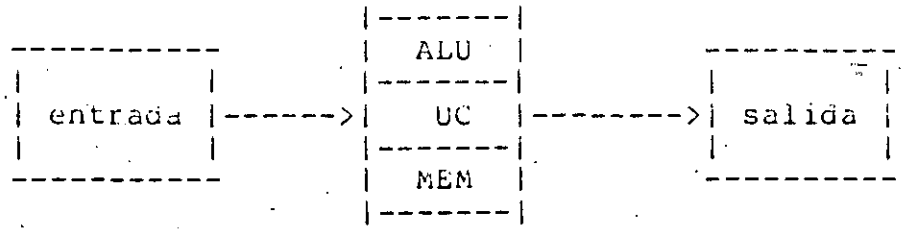


Figura 3-1: Componentes de una computadora.

Las unidades de entrada "aceptan" información codificada, ya sea de humanos o bien de otros dispositivos, esta información se almacena en la memoria (MEM), se procesan por la unidad aritmético lógica (ALU), la cual realiza las funciones deseadas en base a un "programa" almacenado en la misma memoria donde además se pueden encontrar los datos; los resultados se "entregan" al mundo exterior haciendo uso de los dispositivos de salida. Todo esto es coordinado por la unidad de control (UC).

Tambien, se acostumbra representar una computadora como en la figura 3-2, donde la unidad central de proceso (CPU "Central Process Unit"), engloba las funciones de la ALU y de la UC, y se le conoce simplemente como el procesador o CPU.

La mayor parte de los dispositivos de entrada-salida (E/S), tienen posibilidad de realizar funciones tanto de entrada como de salida de información, por lo que se representan en un solo bloque.

Se mencionó que la información codificada, ya sean datos y/o

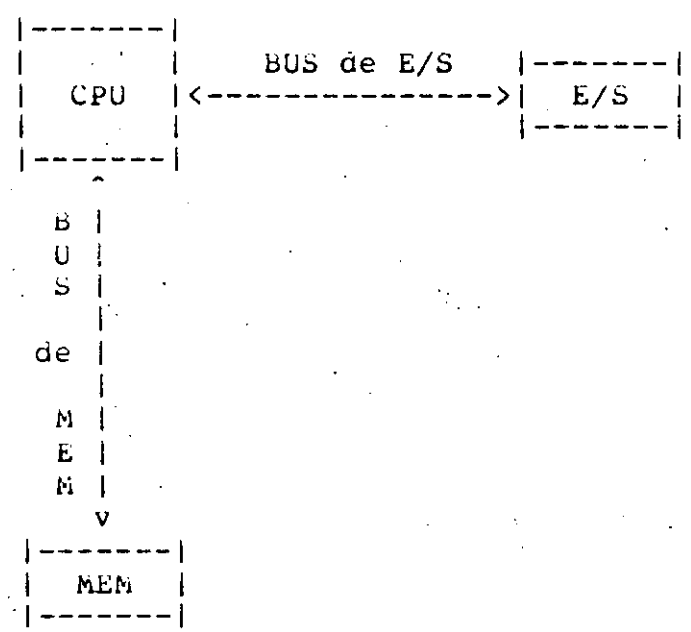


Figura 3-2: Representación de una computadora.

instrucciones se almacenan en la misma memoria. Esto es importante, ya que podemos distinguir entre datos e instrucciones. Ambos coexisten "dentro" de la misma memoria (ocupando diferentes lugares por supuesto). Los programas (conjunto de instrucciones que realizan una función) son comandos que gobiernan el flujo de información dentro del CPU, la memoria y los dispositivos de E/S.

Para que un programa pueda ser ejecutado, deberá estar almacenado en la memoria (aunque no es necesario que lo esté en su totalidad) de donde el CPU obtendrá y ejecutará una a una las instrucciones.

Normalmente la obtención y ejecución se realiza en forma secuencial, aunque es factible tener "brincos" de una instrucción a otra, en base al programa almacenado.

La información manejada dentro de una computadora debe ser codificada (esto es, traducir la información proporcionada por el mundo exterior) de una manera que ésta la entienda. Dado que se emplean circuitos y dispositivos lógicos electrónicos que representan dos posibles estados: encendido (ON) y apagado (OFF), para ello existe el código binario que puede representar el ON como un uno (1) y el OFF con un cero (0). A los unos o ceros se les denomina BITS (que es una contraccion de dos palabras del inglés Binary DigiTS).

Los números usualmente se representan en magnitud y signo o en complemento a dos, los caracteres alfanuméricos se representan básicamente en dos códigos: ASCII, donde cada caracter se representa con 7 bits y EBCDIC donde los caracteres se representan con 8 bits.

3.2.2.1. UNIDADES DE ENTRADA/SALIDA

Las unidades de E/S son dispositivos que nos permiten la comunicación con el medio exterior y la computadora. Existe una gran cantidad de ellos como lo son las terminales de pantalla o CRT (Cathode Ray Tube), los teletipos (TTY), lectoras de tarjetas y de cinta de papel, impresoras, unidades de cinta magnética, unidades de discos magnéticos, graficadoras, digitalizadores, etc.

3.2.2.2. MEMORIA

La función de la memoria consiste en almacenar datos e instrucciones. Podemos distinguir entre dos tipos de memoria: **Memoria Principal**, es aquella que esta formada por dispositivos electrónicos rápidos, capaces de almacenar información, la otra es la llamada **Memoria Secundaria o Masiva**, ésta se encuentra externa a la computadora y está formada de elementos con propiedades magnéticas (discos, floppys, diskettes y cintas) que permiten la grabación y almacenamiento de información.

La memoria principal está organizada en celdas, cada una de ellas es capaz de almacenar un cero o un uno, es decir un bit de información. Estas celdas se pueden manejar en forma individual o bien agruparlas en conjuntos de varios bits, formando **"bytes"** y éstos a su vez formarán **"palabras"** ("words").

La memoria principal puede configurarse de tal manera que el contenido de un bit, byte o palabra pueda ser obtenido o almacenado (depende de la computadora) en una sola operación de lectura o escritura respectivamente. Generalmente el acceso es por palabra y para poder hacerlo, es necesario darle un nombre diferente a cada una de ellas. Estos nombres son números de identificación y se les denomina **Direcciones Físicas o simplemente direcciones.** Una dirección de una localidad de memoria es la identificación dada a una posición de la memoria.

Al número n de bits que forman una palabra, se le conoce como **"longitud de la palabra"**, que varía de acuerdo a la máquina en cuestión. Actualmente las micros tienen entre 8, 16 y 32 bits, las minis entre 16 y 32 bits y las maxis más de 32 bits.

La capacidad o **espacio de memoria**, es un parámetro que

indica el número de localidades de memoria (palabras o bytes) que tiene una computadora, siendo valores típicos desde 4 Kb hasta 20 Mb (1 Kb = 1024 bytes, 1Mb = 1000 Kb).

Las memorias principales pueden o no tener capacidad para poder accesarse ya sea para escribir y o leer información. Aquellas que permiten tanto la lectura como la escritura son llamadas memorias RAM (Random Access Memory),las cuales pueden ser estáticas (SRAM) o dinámicas (DRAM). Existen aplicaciones que requieren la información permanentemente almacenada o raramente alterada (v.g. los programas de control en las calculadoras de bolsillo están usualmente almacenados permanentemente), las memorias que proporcionan éste tipo de acceso (solo lectura) son llamadas memorias ROM (Read Only Memories) y aquellas memorias ROM que pueden ser reprogramadas o reescritas son las PROM (Programmable Read Only Memories). La información almacenada en las ROMs y PROMs es no volatil, esto es, aquella no se pierde cuando dejan de ser energizadas, a diferencia de las RAM que son volátiles.

El tiempo para accesar una localidad de memoria se le conoce como ciclo de memoria y varía, dependiendo de la computadora entre unos 100 ns (un nanosegundo "ns" es una milmillonesima parte de un segundo) a 1 microsegundo.

3.2.2.3. CPU (Unidad Central de Procesamiento)

El procesador está formado por el ALU, la cual hace las veces de una calculadora, realizando funciones aritméticas y lógicas; la Unidad de Control se encarga de organizar y coordinar la operación de los diferentes dispositivos conectados a la máquina. La UC envía las señales de tiempo y sincronía para realizar las diferentes instrucciones. Estas señales de tiempos son realizadas por un circuito de tiempos llamado "reloj del sistema" o simplemente reloj.

El reloj marca secuencias de tiempos repetitivos llamados ciclos de tiempo, de máquina o periodos de reloj. Los ciclos o periodos son cuantificados en unidades de tiempo, a ésta medición se le denomina frecuencia (la frecuencia y el periodo son reciprocos). La frecuencia se mide en términos de "Hertz" cuando la unidad de tiempo utilizada es el segundo, esto es, un Hertz (Hz) es un ciclo por segundo (1Hz = ciclo/s). Las abreviaciones KHz y MHz , denotan respectivamente Kilo (mil) y Mega (millon) de Hertz. Se debe tener cuidado de no confundirlo con el "reloj de tiempo real" o "timer", el cuál es usado para generar "interrupciones" (las cuales se trataran posteriormente) y que es conectado externamente al CPU para provocarselas.

2.2.1.3.1. BLOQUES FUNCIONALES DEL PROCESADOR

En la figura 3-3 se muestra de manera mas detallada la estructura interna del procesador y las conecciones entre el y la memoria principal.

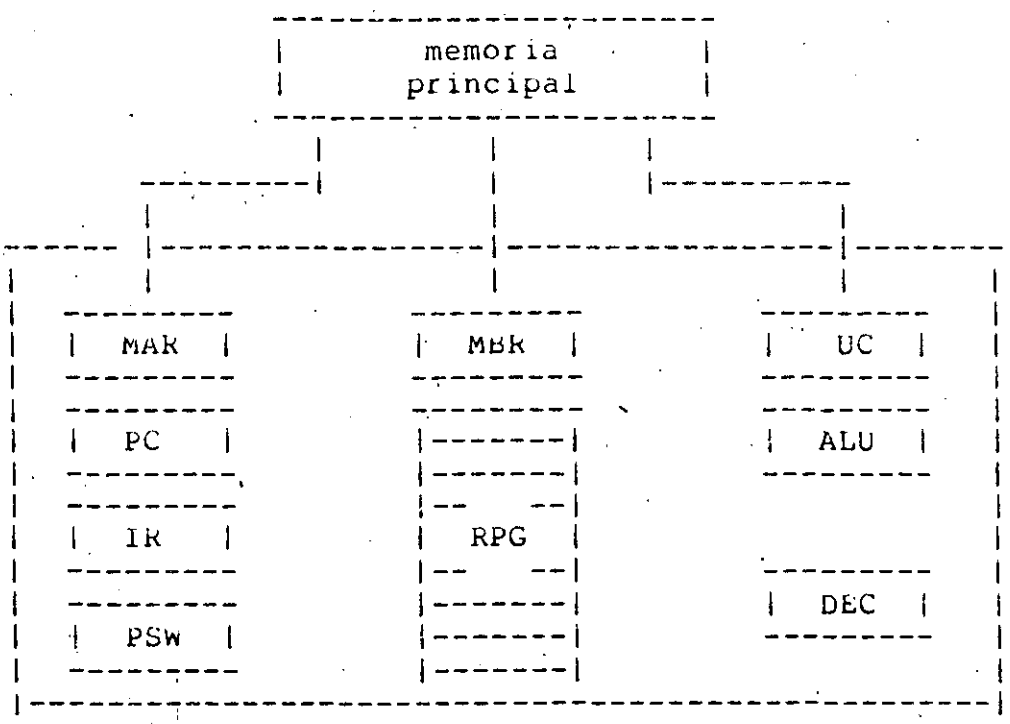


Figura 3-3: Conecciones entre el CPU y la memoria principal.

A excepción de los bloques UC, ALU (vistos anteriormente) y del bloque DEC, los demás son dispositivos de almacenamiento temporal con capacidad de una o dos palabras, similares a las localidades de la memoria principal. Estos dispositivos electrónicos son llamados registros. El bloque DEC es un circuito complejo que puede contener varios registros, decodificadores y lógica asociada.

A continuación se hace una descripción de los diferentes bloques que componen al CPU.

1. PC.- Program Counter. Contiene siempre la dirección de memoria de la siguiente instrucción a ejecutar.

2. MAR.- Memory Address Register. Contiene la dirección de memoria de la instrucción o dato que se va a leer de la memoria.

3. MBR.- Memory Buffer Register. Registro donde se almacena el contenido de la localidad de memoria apuntada por el MAR.

4. IR.- Instruction Register. Contiene la instrucción que se está ejecutando.

5. RPG.- Registros de Propósito General. Registros temporales donde el usuario o el sistema almacenan operandos de carácter temporal.

6. PSW.- Processor Status Word. Registro del CPU que contiene información respecto al estado actual del procesador en base a la operación anteriormente realizada.

7. DEC.- Decodificador. Circuito encargado de interpretar la instrucción que se encuentra en el IR y que fue previamente leida de memoria.

3.2.2.4. EL BUS

Hasta ahora se han discutido las diferentes partes funcionales de una computadora. Para que ésta sea un sistéma funcional, sus unidades deberan estar conectadas de una manera organizada. Existen diferentes maneras de hacerlo y tienen mucho que ver con la velocidad de operación de la computadora.

Para que una computadora tenga una velocidad razonable de operación, esta deberá estar organizada de tál manera que sus unidades puedan manejar completamente una palabra a un tiempo dado. Lo cual también significa que las transferencias sean hechas en un solo tiempo (una palabra a la vez y no bit por bit), esto implica la consideración de un gran número de lineas para establecer las conecciones. Una colección de tales lineas (alambres), que tengan una identidad en común es llamada "bus". El bus que transporta datos es llamado "bus de datos", tambien se vió que para accesar un dato a memoria es necesario una dirección, así que el conjunto de lineas que transportan direcciones se les denomina "bus de direcciones". Además de las lineas que transportan datos y direcciones, es esencial tener algunas lineas para propósitos de control. A menudo se considera al bus como una sola entidad consistente de lineas de control, datos y direcciones.

La configuración de una computadora mostrada en la figura 3-2 muestra dos buses denominados, uno el "bus de memoria", por el cuál el CPU interactúa con la memoria; el otro bus es el llamado "bus de E/S" por el cuál se manejan las funciones de entrada/salida, de tál manera que los datos pasan a través del

CPU en ruta nacia la memoria principal. En esta configuración las transferencias de E/S son bajo el control directo del CPU. Este inicia la transferencia y la monitorea hasta su completa finalización (esta acción es comunmente llamada "entrada salida/programada").

Una configuración un poco diferente a la anterior es una en la que el CPU y la memoria estan invertidos, esto es, la unidad de E/S está directamente conectada con la memoria a través del bus de E/S y el CPU con la memoria por medio del bus de memoria. En este esquema las transferencias de E/S son realizadas directamente desde o nacia memoria. Debido a que la memoria tiene poca o nula circuiteria para controlar tales transferencias (a diferencia del CPU), es necesario introducir un dispositivo que sea capaz de efectuarlas. Este dispositivo es llamado "Canal de E/S", o "Manejador de Acceso Directo a Memoria" (DMA-"Direct Access Memory"). La manera de llevar a cabo la transferencia es haciendo que el CPU indique la información necesaria al canal o DMA, el cuál controla la transferencia.

Las dos descripciones anteriores son representativas de la mayoría de las computadoras y en general de las grandes. Muchas máquinas tienen diferentes buses lo que de hecho las convierte en máquinas "multibus". Sin embargo, su operación está adecuadamente representada por las organizaciones de dos buses. La razón de la inclusión de buses es para mejorar la velocidad de operación en base a un mayor paralelismo de acciones entre las unidades de la computadora.

Una organización significante por su estructura es la que considera todas las unidades funcionales de la computadora (Entrada, Salida, Memoria y CPU) en "un solo bus", el cual se muestra en la figura 3-4. Este provee un medio único de interacción. Debido a que el bus solo puede ser usado para una transferencia a la vez, esto conlleva a que solamente dos unidades pueden estar activas usando el bus a un tiempo. Este tipo de bus contiene las lineas de datos, direcciones y control. La principal virtud de la estructura de un solo bus es su bajo costo y flexibilidad para "colgarle" dispositivos periféricos. Su desventaja es su baja velocidad de operación. La estructura de un solo bus es frecuentemente encontradas en minis y microcomputadoras.

A pesar que las diferencias entre las diferentes estructuras de bus tienen un efecto significante en la eficiencia de las computadoras, esto no afecta escencialmente los principios de operación de las comutadoras de proposito general. Por lo que, para fines del curso, se puede considerar que la estructura del bus es independiente de los principios funcionales de operación.

Por último nay que mencionar que los elementos que conforman

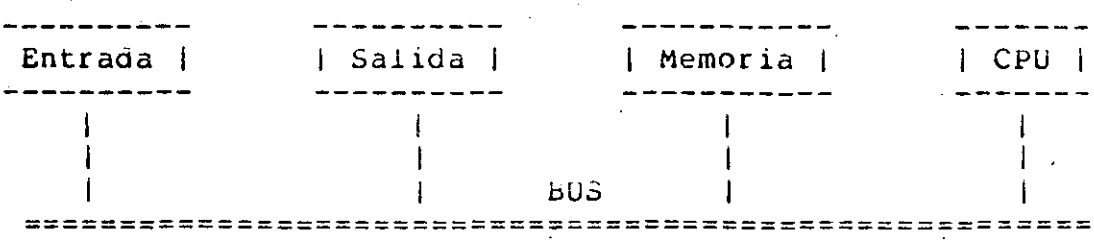


Figura 3-4: Estructura de un solo bus.

al CPU (ver fig. 3-3), tambien estan interconectados por un bus denominado "bus interno".

3.2.3. CONCEPTO DE PROGRAMA ALMACENADO

El concepto de "programa almacenado" fué introducido en 1833 por Cnarles Babbage como una caracteristica de una "máquina calculante", que el propuso para que se construyera. Esa máquina, realizaría aritmética decimal, ayudandose de engranes mecánicos de conteo. La máquina de Babbage, incorporó principios que son básicos para el diseño de las computadoras electrónicas. La forma moderna del programa almacenado fué introducido por John von Neumann, quien le añadió un refinamiento significante. El especificó que tanto los datos a ser procesados como las instrucciones que las procesan debian estar escritas en la misma notación. Con esta contribución las instrucciones pueden ser manipuladas por la máquina como si fueran datos. Por lo que un programa puede alterar a otro programa o a sí mismo.

La idea de que un programa pueda alterar a otro puede extenderse a que el programa puede ser controlado, monitoreado o supervisado por otro mas "inteligente". Este nuevo "superprograma", además de manejar programas puede tener la capacidád de controlar los recursos de la computadora, por ejemplo, la memoria principal, las unidades de entrada y salida y el procesador, y de coordinarlos para un mejor aprovechamiento de la misma computadora así como de los programas que supervisa. A tál "superprograma" se le denomina "sistema operativo".

3.2.4. EL CONJUNTO DE INSTRUCCIONES

Una **instrucción** es una operación ejecutable por la computadora. El hecho de poder ser ejecutable está en función de su organización y caracteristicas propias del CPU, pues desencadenan una serie de acciones muy bien definidas en sus unidades funcionales. Así que, diferentes computadoras podrán ejecutar instrucciones "similares" (nótese que pueden ser similares, ya que, por ejemplo, una instrucción "suma dos valores" puede ser ejecutada por la mayoría de las computadoras y seguramente en diferente forma), pero,. existirán instrucciones que solo puedan ser ejecutadas en cada una de ellas.

La instrucción para ser "entendible" por la computadora deberá estar representada por un patron de bits. El tamaño de ese patrón, es una indicación del número de bits requeridos para construir una instrucción ejecutable. El número de bits o tamaño de la instrucción generalmente es algún múltiplo del tamaño de la palabra de la computadora y pueden ser normalmente de 1 a 3 palabras. Usualmente instrucciones de una palabra requieren menos tiempo de ejecución que una de dos palabras y a su vez ésta menos tiempo que una de tres palabras. Las instrucciones cortas son generalmente aquellas que operan con registros u operandos contenidos en registros, mientras que las instrucciones largas pueden ser capaces de trabajar con uno o dos operandos en memoria principal.

El **conjunto de instrucciones**, es el grupo de instrucciones ejecutables por la computadora(NOTA: no confundir con programa, el cual es un conjunto de instrucciones que le indican a la computadora la realización de una tarea específica y generalmente siguiendo un lineamiento lógico denominado "algoritmo").

El conjunto de instrucciones puede ser dividido en clases bien delimitadas según las acciones que produzcan en la computadora. De manera general estas clases son:

1. Transferencia de datos entre la memoria principal y los registros del CPU.
2. Operaciones aritméticas y lógicas sobre los datos(operandos).
3. Transferencias de control y secuencia de ejecución de instrucciones en un programa.
4. Transferencia de datos desde o hacia dispositivos de E/S.

3.2.5. EVENTOS DE TIEMPO DENTRO DEL CPU

Existe dentro del CPU un mecanismo para producir las acciones de ejecución de las instrucciones en base a una referencia fija en el tiempo. Esta base de tiempo está dada por el circuito del reloj que se encuentra en el CPU.

Para que una instrucción sea ejecutada, deberá apegarse a un cierto tiempo, el cual está en función de las características estructurales de la computadora (en particular el CPU). Este tiempo de ejecución de la instrucción se le denomina **ciclo de instrucción.**

El ciclo de instrucción, está dividido en los llamados ciclos de máquina, los cuales a su vez estan formados por una cantidad fija de ciclos de reloj. Existen diferentes tipos de ciclos de máquina (cada una realizando una serie de acciones básicas), los cuales pueden ser:

- Ciclo para la obtención de una instrucción.
- Ciclo para realizar una lectura o escritura de algún operando o resultado.
- Ciclo para la lectura o escritura en operaciones con dispositivos de E/S.
- Ciclo para manejar la utilización del bus.
- Ciclo para el manejo de interrupciones.
- Ciclo para indicar el fin de acciones del CPU.

Cada ciclo de instrucción tiene un número variable de ciclos de máquina, pues depende del tipo de instrucción a ejecutar. Esto es, no es lo mismo ejecutar una instrucción que indique que se detenga la ejecución de acciones del CPU que una que requiera de accesar un operando de memoria almacenarlo en un registro, accesar otro operando de memoria, guardarlo en otro registro, aplicarles algún operador y posteriormente almacenar el resultado en alguna localidad de memoria; esto evidentemente se llevará un mayor tiempo de ejecución y por consiguiente el ciclo de instrucción es mayor que el que indica el alto de acciones del CPU.

3.2.6. FUNCIONAMIENTO DE UNA COMPUTADORA

A pesar de lo complejo que pueda parecer la computadora, lo único que realiza siempre e ininterrumpidamente es un ciclo de instrucción de tres fases, las cuales a su vez por simplicidad en la explicación pueden ser alguno de los ciclos de máquina antes mencionados o bien pueden estar constituidos por varios ciclos de máquina. Estas fases son identificables en casi todas las computadoras y son:

FETCH Se obtiene la instrucción de memoria principal.

DEFER Decodificación o interpretación de la instrucción.

EXECUTE Fase que ejecuta la instrucción.

Este ciclo de instrucción se realiza repetitivamente, ya sea que la máquina esté ejecutando una parte del sistema operativo o bien el programa del usuario.

El ciclo lo ejecuta la unidad de control, para lo cual genera las señales de tiempo y sincronía necesarias.

Durante la fase de **FETCH** se realizan las siguientes operaciones:

MAR <-- PC
Se copia en el MAR el valor del PC; éste apunta siempre a la siguiente instrucción por ejecutar, así que antes de comenzar la ejecución del programa el PC ya tiene la dirección de la primera instrucción.

PC <-- PC+1
Se incrementa automáticamente el valor del PC para que apunte siempre a la siguiente instrucción por ejecutar.

MBR <-- MEM[MAR]
Se copia en el MBR el contenido de la localidad de memoria apuntada por el MAR. Es decir, en el MBR se coloca la instrucción misma.

IR <-- MBR
El contenido del MBR es copiado en el IR, el cual contendrá la instrucción que se está ejecutando.

A continuación la fase de **DEFER** toma el contenido del IR y lo pasa al circuito decodificador de instrucciones, donde se decide el tipo de instrucción de que se trata.

Finalmente en la fase de **EXECUTE** se realiza la ejecución de la instrucción. Puede suceder que ésta requiera de algun dato (operando) almacenado en memoria, o bien que tenga la necesidad de almacenar algun valor en la misma memoria principal, por lo que se hará uso del MAR y del MBR, los cuales servirán como registros "puente" entre la memoria y el procesador.

Una vez terminado la fase de **EXECUTE** comienza de nueva cuenta la fase de **FETCH** iniciandose el ciclo, se copiará en el MAR el valor del PC que había sido incrementado automáticamente. Ahora el MAR tiene la dirección de la siguiente instrucción ejecutable del programa.

En casi todas las máquinas modernas, las fases del ciclo anterior se traslapan con el fin de ganar velocidad en la ejecución de las instrucciones, esto es, la fase de ejecución de la instrucción puede realizarse simultaneamente con la de decodificación de la siguiente instrucción y esta con la fase de obtención de una tercera.

3.2.7. MODOS DE DIRECCIONAMIENTO

Para que una instrucción pueda accesar los diferentes operandos necesarios para su correcta ejecución, es necesario dotarla de algún mecanismo llamado **método o modo de direccionamiento**; el cual se encuentra indicado dentro de la misma instrucción. Los diferentes modos de direccionamiento se pueden definir como las diferentes formas de accesar los registros de propósito general del procesador o las localidades de memoria.

Si bien en todas las computadoras, estos modos de direccionamiento son diferentes en su forma no lo es así en su función, pues pueden distinguirse cuatro modos básicos:

INMEDIATO En este modo el operando se encuentra dentro de la misma instrucción, esto es, el operando es accesado por el CPU al momento de hacer el FETCH de la instrucción.

DIRECTO o ABSOLUTO
En este modo, la instrucción tiene explícitamente la dirección efectiva de localización del operando.

INDIRECTO o DIFERIDO

La dirección efectiva del operando está en la localidad de memoria principal o registro cuya dirección aparece en la instrucción.

En este modo existe un paso más de direccionamiento que en el directo, pues no se tiene la dirección directamente en la instrucción, sino que, el operando se obtiene indirectamente, esto es, existe una dirección en la instrucción que indica una localidad de memoria o registro donde se encontrara la dirección (la efectiva) donde se encontrará finalmente el operando.

INDEXADO o RELATIVO

En este modo, la dirección efectiva del operando es **generada** por la suma de un valor **índice** con la dirección dada en la instrucción. Esto es, el operando se encuentra en una dirección **relativa** a otra considerada como "base".

El valor índice, o símplemente el índice, está normalmente contenido en un registro del CPU. En algunas computadoras, un registro está dedicado únicamente para este propósito. Este es llamado el **registro índice** y está involucrado implícitamente cuando se especifica este modo. En otras computadoras, el registro índice puede ser uno de los registros de propósito general, en tál caso, el registro deberá ser nombrado explícitamente en la instrución.

3.2.8. INTERRUPCIONES

Una **interrupción** en su amplio sentido, es la acción provocada en el CPU debida a una requisición de atención de un evento externo o interno al procesador.

Las interrupciones debidas a un evento externo están provocadas por los dispósitivos de E/S, los cuales provocan la acción por medio de señales transmitidas por el bus hacia el CPU. La señal de interrupción normalmente es generada por circuitos propios de los dispositivos de E/S, esto es, es una interrupción por "hardware" o símplemente interrupción.

Sin las interrupciones, la lógica externa no tendría manera de controlar las secuencias de ejecución de un programa. Sin las interrupciones, un dispositivo externo que requiera la ejecución de un programa en específicc debe "llamar" la atención del

procesador modificando el estado de algunas banderas de estado del mismo dispositivo. El dispositivo externo debe entonces esperar hasta que el procesador "vea" el estado de esas banderas. A estas acciones se les conoce como **"poleo"**. El poleo no es adecuado para situaciones en las que la lógica externa requiera de atención inmediata y quiza despues de un cierto tiempo el procesador cheque el estado de las banderas del dispositivo para atenderlo y entonces probablemente sea muy tarde para que se realize alguna acción, pues, puede suceder que datos que el dispositivo externo necesite proporcionar al procesador sean perdidos; o bien que la información que tiene el dispositivo no llegue a tiempo al procesador; o tambien puede suceder que el procesador continue efectuando operaciones de entrada/salida sobre el dispositivo una vez que el mismo determine algun error fatal y trate de reportarlo a el. Para evitar que lo anterior suceda, el CPU debe estar entonces "poleando" o monitoreando continuamente al dispositivo, lo cual resulta en consumir mucho tiempo en la acción.

Muchas veces el evento externo está asociado con la transferencia de datos entre los dispositivos de E/S y memoria principal o CPU; tambien con funciones de "tiempo real" provocadas por circuitos externos al CPU que generan intervalos de tiempos para la atención de algún evento en momentos preestablecidos; o bien puede estar asociado con condiciones catastróficas o anormales, típicamente por fallas de potencia o una falla en una porción de la computadora o en un sistéma en tiempo real conectada a ella. Los ejemplos anteriores son situaciones en las que el dispositivo debe tomar un papel activo, forzando al procesador a detenerse y realizar una serie de acciones que atiendan la necesidad en forma precisa.

3.2.8.1. ATENCION DE LA INTERRUPCION

El dispositivo externo debe mandar una serie de señales apropiadas para realizar una **"petición de interrupción"**. El procesador prueba esas lineas de petición de interrupción en la ejecución de cada instrucción. Algunas interrupciones pueden ser "nabilitadas" o "desnabilitadas" bajo el control de programa (esto es, por el usuario), otras no pueden serlo. El procesador ignora aquellas peticiones de interrupción desnabilitadas; este sirve las peticiones de interrupción habilitadas de la siguiente manera:

1. Detiene la ejecución del programa que esté ejecutando en ese momento.

2. Ejecuta un programa especial que cumple con las necesidades del dispositivo externo que provoca la interrupción.

3. Continúa ejecutando el programa a partir del punto donde ocurrió la interrupción.

RECONOCIMIENTO DE LA INTERRUPCION

El paso 1 anterior se le conoce como **"reconocimiento de la interrupción"**. Durante el reconocimiento de la interrupción el procesador debe "salvar" o guardar el PC y el PSW, en alguna area de memoria, a menudo es en un area de RAM denominada "stack". El PC entonces direcciona la siguiente instrucción secuencial; esto es, la instrucción la cual debió ser ejecutada si la interrupción no hubiese ocurrido. Esta es tambien la instrucción que será ejecutada tan pronto como la interrupción haya sido servida en el paso 3. En el paso 2, la ejecución "brinca" a un programa especial dedicado a una interrupción en particular que haya sido reconocida.

RUTINA DE SERVICIO DE LA INTERRUPCION

El programa ejecutado debido a la atención de una interrupción reconocida se le conoce como **"rutina de servicio de la interrupción"**. Esta rutina normalmente comienza salvando información adicional que no es automáticamente salvada durante el proceso de reconocimiento. Por ejemplo, los contenidos de todos los registros son frecuentemente guardados en el stack antes de que cualquier registro sea modificado por la rutina de servicio. Es entonces cuando la rutina de servicio efectúa las operaciones requeridas por la interrupción reconocida.

RETORNO DE LA INTERRUPCION

Finalmente, en el paso 3, ocurre un retorno de la interrupción; esto es exactamente el proceso inverso al del reconocimiento de la interrupción. Si la rutina de servicio salvó información adicional antes de empezar a ejecutarse, entonces esta restaurará esta información antes de efectuarse el retorno de la interrupción presente. Por ejemplo, si la rutina de servicio de interrupción salvó inicialmente todos los registros del CPU en el stack, entonces restaurará todos esos contenidos del stack. Es entonces cuando se ejecuta una instrucción del tipo "Retorno de Interrupción"; esta restaura los contenidos del PC y del PSW que fueron salvados durante el proceso de reconocimiento, causando posteriormente que continúe la ejecución del programa interrumpido a partir del punto donde fué interrumpido (recuerdese que el PC restaurado tiene la dirección de la siguiente instruccion a ejecutarse del programa).

3.2.8.2. INTERRUPCIONES NO MASCARABLES

Algunas interrupciones son tán importantes que no pueden ser desnabilitadas. Estas son llamadas "interrupciones no mascarables". El procesador siempre reconocerá y servirá una interrupción no mascarable. Las interrupciones no mascarables son frecuentemente usadas para indicar una falla de potencia; un procesador usualmente puede ejecutar unos cientos de instrucciones entre el tiempo de detección de la falla de potencia y el tiempo cuando la potencia sea insuficiente para operar al procesador (a la computadora en general). Esas instrucciones pueden "poner en orden" al programa, permitiendo un reinicio cuando la potencia sea reestablecida de nuevo.

3.2.8.3. INTERRUPCIONES MASCARABLES

Las interrupciones que pueden ser habilitadas y deshabilitadas son conocidas como "interrupciones mascarables". Todas las interrupciones encontradas durante la ejecución normal de un programa deberían ser interrupciones mascarables (o enmascarables).

La secuencia del evento de interrupción es por sí misma calificable, cuando la interrupción es de tipo interno al procesador.Por lo que el CPU permiten que la secuencia del evento sea iniciada por cierta lógica interna a el. Esto puede ocurrir en una de las dos formas siguientes:

1. Una condición detectada durante la ejecución de una instrucción puede iniciar una secuencia de eventos similar a una petición de interrupción; esta es llamada **"TRAP DE SOFTWARE"**. Por ejemplo, si en el ciclo de Fetch se obtiene una instrucción con codigo desconocido, el CPU debe responder ejecutando un Trap.

2. Muchos procesadores (la mayoria de los microprocesadores de 16 bits o mas), tienen instrucciones que están diseñadas para que ocurra una secuencia de interrupción. Estas son conocidas como **"INTERRUPCIONES DE SOFTWARE"**.

3.2.8.4. PRIORIDADES DE INTERRUPCION

Un procesador puede recibir diferentes peticiones de interrupción; por lo que puede suceder que dos o mas peticiones de interrupción ocurran al mismo tiempo, y solo una debe ser servida determinada por una "lógica de prioridades de interrupción". Esta lógica de prioridades se aplica unicamente durante la fase de reconocimiento de la interrupción; no se aplica durante el periodo completo de servicio de la interrupción. Por ejemplo, si dos peticiones de interrupción ocurren simultáneamente y se reconoce una petición de interrupción con prioridad alta, entonces, la rutina de servicio de interrupción alta mantendrá la interrupción de baja prioridad deshabilitada. Si la rutina de alta prioridad no mantiene deshabilitada a la de baja, sucederá que la interrupción de baja prioridad sea reconocida dentro de la rutina de alta prioridad.

Una vez, que la rutina de alta prioridad inicia su ejecución, unicamente la petición de interrupción de baja prioridad quedará pendiente. Solo podrá ser reconocida dentro de la rutina de alta prioridad si dentro de ella se habilitan las interrupciones.

3.2.8.5. INTERRUPCIONES VECTORIZADAS

Una vez que la interrupción ha sido reconocida, hay dos maneras con las cuales el CPU puede identificar la fuente de la interrupción. Si la interrupción es vectorizada, entonces no se requiere proceso de identificación, debido a que la secuencia de reconocimiento maneja la identificación. Cada interrupción vectorizada tiene una rutina de servicio y el proceso de reconocimiento de interrupción incluye algun mecanismo para la identificación de esta rutina de interrupción. Este tipo de interrupciones hacen uso de un vector, el cual simplemente es un apuntador a una dirección de memoria que contiene la rutina de servicio apropiada. Los vectores de auto-identificación son proporcionados por los dispositivos que interrumpen y pueden ser tanto la dirección de memoria completa o un número índice que es sumado a una base para desarrollar el vector completo.

3.2.8.6. INTERRUPCIONES NO-VECTORIZADAS

En este tipo de interrupciones, el procesador proporciona directamente la rutina de atención al dispositivo que lo requiere. La manera en que identifica al dispositivo es por medio de una "linea" especial conectada entre el y el procesador sin necesidad de una identificación explícita. Pero pueden existir casos en los que dos o más dispositivos compartan una sola linea de interrupcion. Entonces el microprocesador tiene que "polear"

a los dispositivos que comparten la petición de interrupción. El CPU debe leer el contenido de los registros de estatus de los diferentes dispositivos, lo cual es lo mismo si no hubieran interrupciones (o sea si siempre estuviera poleando para determinar si hay interrupción). La razón de que polee es que **es la interrupción la que inicia ese poleo.** Sin la interupción el procesador tendría que estar poleando continuamente, o bien en otro esquema que de tiempo en tiempo lo haga.

3.3. QUE ES UN MICROPROCESADOR ?

Un **microprocesador** es el CPU de una computadora, reducido a tál escala que cabe en un chip (los primeros a veces ocupaban más de uno), el cuál contiene decenas de miles de transistores, resistencias y elementos de circuitos similares integrados en "gran escala" (LSI).Aunque estrictamente nablando, un microprocesador puede ser implementado con componentes mas convencionales de "mediana escala de integración" (MSI).

El microprocesador **no** es una computadora en su amplio sentido (al menos, no en la mayoria de los casos), pero contiene los elementos lógicos para manipular datos y efectuar operaciones lógicas y aritméticas sobre ellos. Para que sea una computadora **completa**, el microprocesador, generalmente deberá reunir una serie de elementos de **soporte** (que junto con el microprocesador conforman las llamadas **familias**) tales como memoria, circuitos de entrada y salida, fuentes de poder y otros con funciones especializadas. El microprocesador junto con sus circuitos de soporte normalmente están organizados alrededor de una estructura de "un solo bus" (ver fig. 3-4), esto es todos los elementos que "cuelgan" del bus estan conectados en paralelo, con las consabidas ventajas y desventajas de bajo costo y "relativa" lenta velocidad de operación debido, al compartimiento del bus entre sus elementos (solo dos lo pueden utilizar en una acción).

Así que, una computadora contigurada alrededor de un micrprocesador es llamada **microcomputadora.** También existen microcomputadoras integradas en un solo chip, esto es: CPU,memoria principal y circuitos de entrada-salida empaquetados en uno solo.

3.3.1. FUNCIONAMIENTO DE UN MICROPROCESADOR

La tarea de un microprocesador, así como en el CPU de una gran computadora, es:

- Recibir datos en forma de "cadenas" de dígitos binarios, a través de un dispositivo de entrada.
- Almacenar los datos para un procesamiento posterior.
- Realizar operaciones aritméticas y lógicas sobre los datos de acuerdo con las instrucciones previamente almacenadas (programa).
- Y por último otorgar los resultados al usuario a través de mecanismos (dispositivos) de salida.

Así que el diagrama típico de un microprocesador es el mostrado en la figura 3-3, y el funcionamiento de sus unidades se aplica de manera similar al descrito en la sección anterior.

3.3.2. MICROPROCESADORES "BIT SLICED"

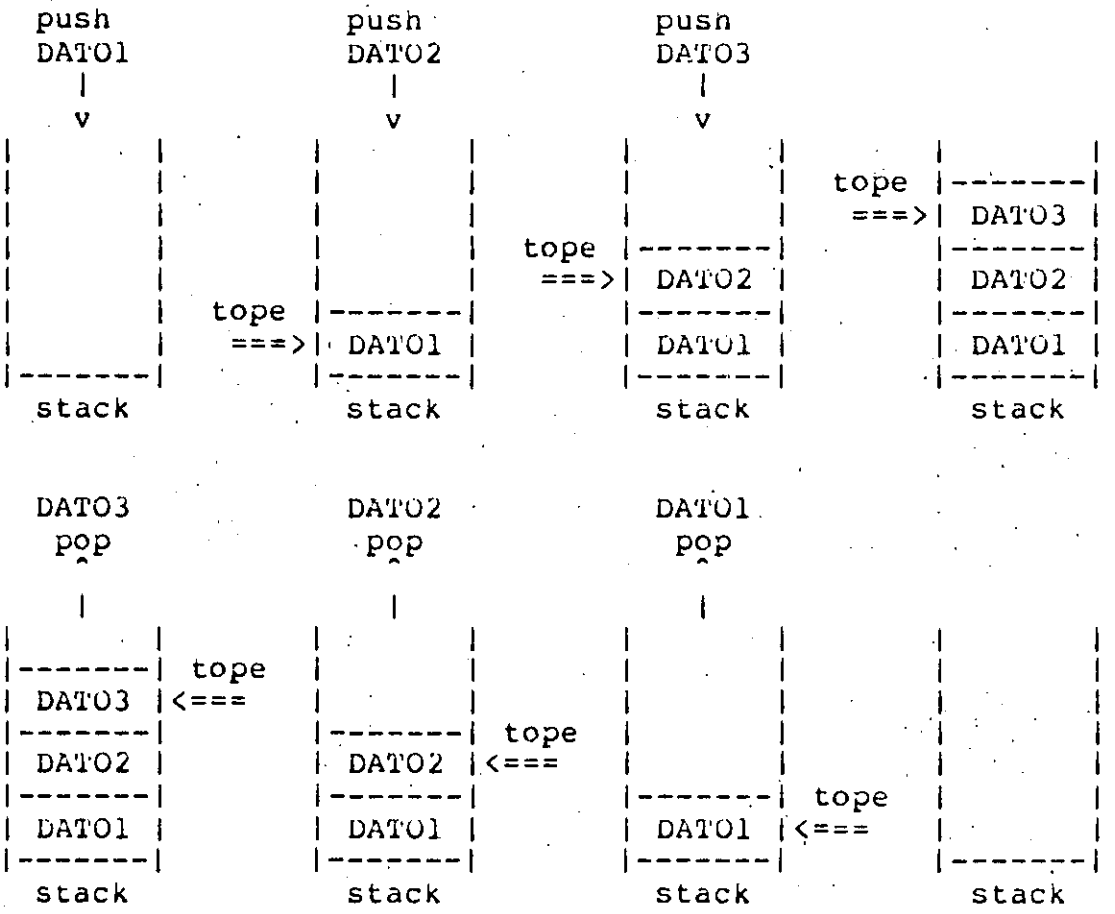
Los microprocesadores generalmente manejan un tamaño fijo de palabra (la cuál puede ser 8, 12, 16 y en ocasiones de 32 bits); también existen aquellos en los cuales sus unidades funcionales, están divididos modularmente en varios chips idénticos que pueden ser ligados en paralelo, el número total de chips depende de la longitud de la "palabra" que el usuario desea procesar: cuatro bits, ocho bits, doce bits o mas. Tal arreglo "multichips" es conocido como una **organización "rebanadas de bits" ("bit sliced")**. Una caracteristica de este tipo de microprocesadores es que ellos son **microprogramables**: permiten al usuario crear grupos específicos de instrucciones, lo cual es una ventaja definitiva en muchas aplicaciones.

3.3.3. MICROPROCESADORES DE 8 BITS

Una vez que se han entendido los conceptos fundamentales de una computadora y por consiguiente de los microprocesadores, será necesario revisar algunos representativos tanto por su caracteristica de manejo de palabras de 8 bits como por lo común que son en el mercado, su marcada preferencia y su trascendencia historica (como lo es el caso del Intel 8080, precursor de los micros de 8 bits). No se revisarán los microprocesadores "bit slice", debido a sus aplicaciones tan especificas, además de ser necesario introducir conceptos nuevos a los vistos como lo es la

"microprogramación", concepto que dista (como pudiera intuirse) de la programación de microprocesadores de propósito general con tamaño fijo de palabra y que pudiesen (de mala manera) asociarlos con estos.

Los microprocesadores seleccionados son el **Intel 8080 y 8085, el Motorola 6800, el Zilog Z80, y el 6502 de Mostek.** Su revisión será en base a sus aspectos funcionales y filosofía estructural.



NOTA: el símbolo ===> representa a el apuntador.

Figura 3-5: Las acciones de PUSH y POP en el STACK

Antes de conocerlos, es necesario mencionar una caracteristica importante que presentan la gran mayoría de los microprocesadores y es la del manejo de un area de memoria a veces interno al CPU y en otras arquitecturas externo a el. Esta area de memoria consiste de localidades consecutivas de palabras

manejadas como un **"stack"**, cuyo funcionamiento es el almacenar datos contiguos. Estos datos van siendo apilados uno encima del otro y la manera de poder obtenerlos es: el último apilado (escrito en el stack) es el primero que se lee. Para poder accesar (leer o escribir) datos en el stack, es necesario contar con un **apuntador** o **"pointer"**, el cuál señala la posición del último dato metido en el.Este apuntador tiene la dirección del **"tope"** del stack.

En la figura 3-5, se muestran las acciones llevadas a cabo en el stack, las cuales son la de "meter" y "sacar" datos (escribir o leer), a estas acciones se les conoce como hacer un **"push"** o hacer un **"pop"** respectivamente.

El apuntador en algunos microprocesadores está contenido en un registro dedicado, denominado generalmente como **registro del apuntador del stack o "stack pointer register".**

El "stack", sirve para almacenar los contenidos del PC, del PSW, de los registros en general, y cualquier dato que pudieran ser afectados durante la ejecución del programa o en los casos de interrupciones y llamadas a "subrutinas".

3.3.3.1. EL MICROPROCESADOR 8080

En la figura 3-6, se muestra el diagrama del CPU del microprocesador 8080. En el se pueden distinguir perfectamente el ALU, el PCW llamado FLAG FLIP-FLOPS, la UC, un arreglo de registros, el MAR llamado "Address Buffer", el MBR o "Data Bus Buffer/Latch", el IR, el circuito de decodificación y ejecución de la instucción y el bus interno de datos.

Este microprocesador (o micro simplemente), tiene las siguientes caracteristicas:

TECNOLOGIA: NMOS, en chip de 40 patas.

FUENTES DE ALIMENTACION:
+5,-5 y +12 Volts.

RELOJ: Externo de dos fases, con un circuito generador (8224), frecuencia de 2 MHz (aunque hay versiones con relojes más rápidos de 2.6MHz y 3MHz).

CICLOS DE MAQUINA:
Cada uno requiere de uno a cinco ciclos de reloj. Existen 10 diferentes ciclos: 1. FETCH, 2. MEMORY READ, 3. MEMORY WRITE, 4.STACK READ, 5. STACK WRITE, 6. INPUT, 7. OUTPUT, 8. INTERRUPT, 9.HALT, 10. HALT&INTERRUPT.

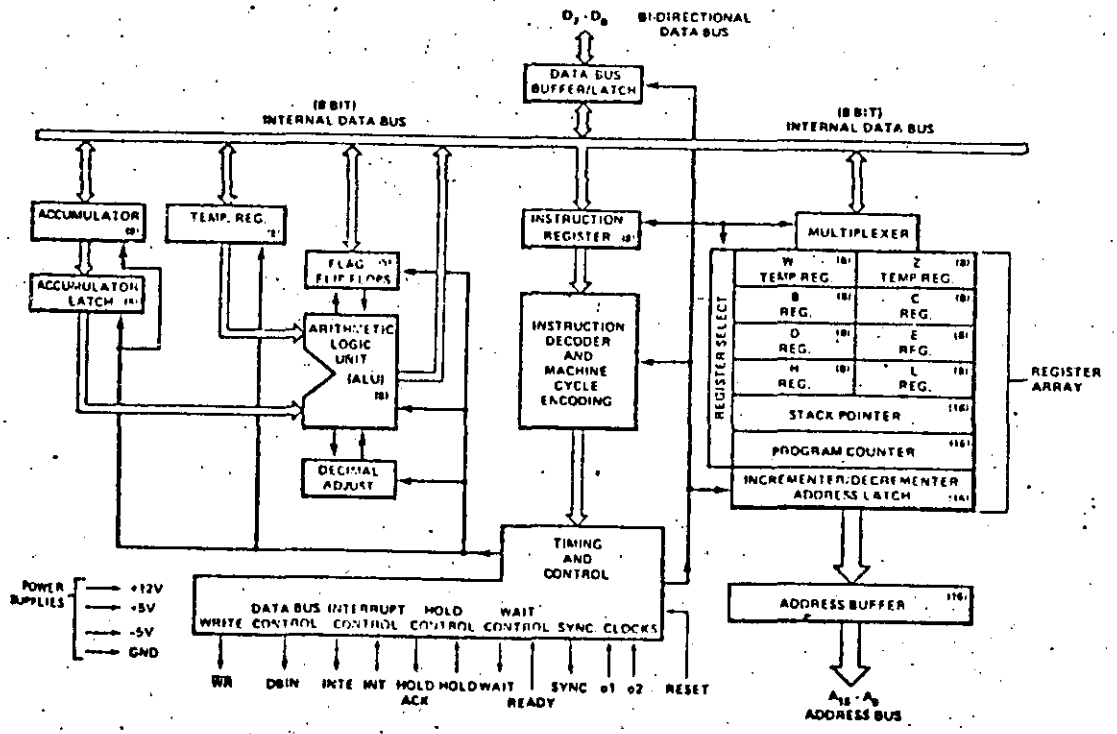


Figura 3-6: Diagrama de Bloques Funcionales del CPU 8080.

CICLO DE INSTRUCCION:
Esta constituido de uno a cuatro ciclos de máquina.

TAMAÑO DE LA INSTRUCCION:
De 8, 16 y 24 bits (1,2 y 3 palabras).

REGISTROS: Hay 6 registros de propósito general arreglados en pares, llamados: B,C; D,E; y HL, con los cuales se pueden manejar datos de 16 bits o de 8 bits utilizandolos por separado, además del acumulador A. Tiene tambien un par de registros temporales W,Z. Y el registro del "stack pointer" de 16 bits.

STACK: Externo, y de tamaño máximo de 64K bytes.

ESPACIO DE MEMORIA:
Puede direccionar 64K bytes con modo de direccionamiento directo.

BANDERAS: Tiene un registro de banderas(flags), las cuales indican el estado de la instrucción previamente ejecutada y son cada una de un bit: No cero, Cero, No acarreo, Acarreo, Paridad par e impar, y Signo positivo y negativo.

ENTRADA/SALIDA: Transferencia de datos entre el acumulador (A) y dispositivos de E/S con tamaño de palabra de 8 bits.

SISTEMA DE INTERRUPCIONES:
Vectorizada con 8 niveles (una por cada rutina de atención).

CONJUNTO DE INSTRUCCIONES:
Consiste de 78 instrucciones, agrupadas en las de movimiento de datos entre registros, registros a memoria al stack, del stack o memoria hacia los registros, de entrada/salida de o hacia el acumulador; de transferencia de control; operaciones aritméticas y lógicas y para manejo de interrupciones. Puede efectuar operaciones de suma y resta entre dos registros en 2 microsegundos.

TIPO DE ARITMETICA:
En complemento a dos y en BCD (4 bits/dígito).

MODOS DE DIRECCIONAMIENTO:
Direccionamiento directo; Direccionamiento por par de registros (se especifica una dirección en uno de los regs. pares); Direccionamiento por el apuntador del stack (una localidad de memoria se puede accesar via el registro del apuntador del stack); y Direccionamiento Inmediato.

Presenta un registro temporal, TMP, el cual recibe información del bus interno y puede mandar todo o porciones de el hacia el ALU, al registro de banderas y hacia el bus interno.

3.3.3.2. EL MICROPROCESADOR 8085

Este micro esta ilustrado en el diagrama de la figura 3-7. Se puede observar que en su estructura es muy similar al 8080 de la figura 3-6, con la diferencia de que el 8085 en primera instancia presenta un circuito para control de interrupciones, un control de E/S seriales para una interfase serial simple. El 8085 utiliza un bus de datos multiplexado (compartido) para datos y direcciones. La dirección es dividida entre el bus de direcciones altas (A15-A8) y los 8 bits bajos de direcciones, son colocados en el bus de Datos/Direcciones (AD7-AD0). Otra diferencia esta en los registros.

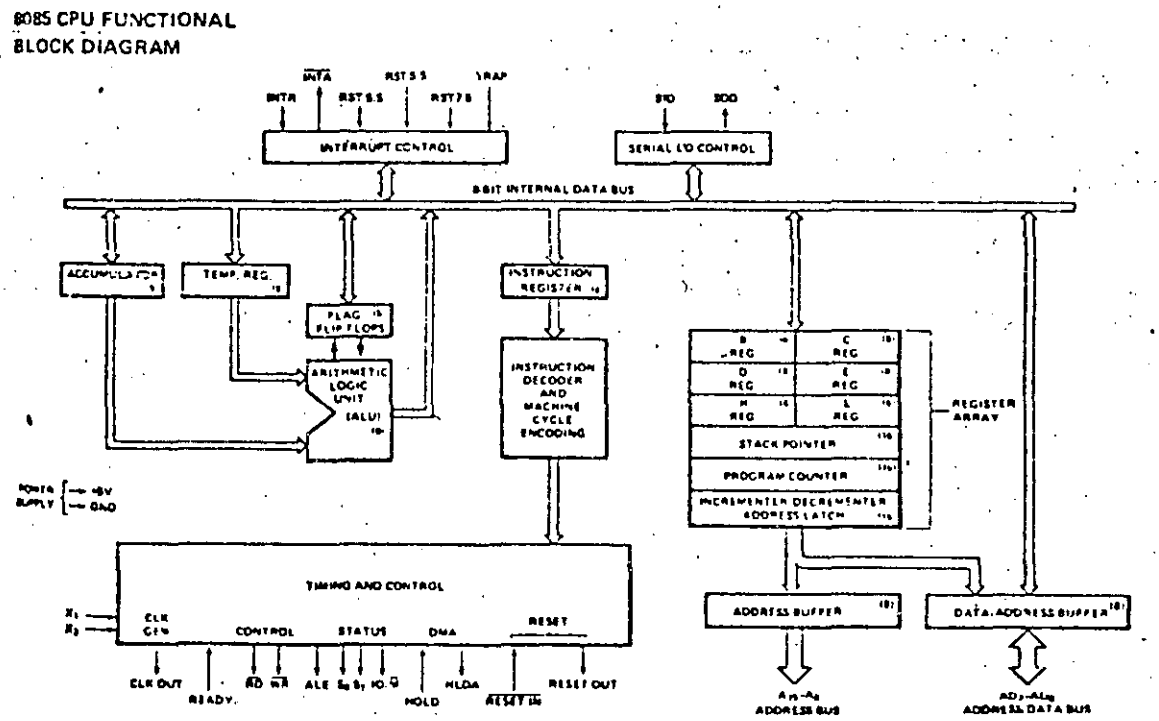


Figura 3-7: Diagrama de los bloques funcionales del CPU 8085

Las caracteristicas principales de este micro estan a continuación:

TECNOLOGIA: NMOS, en chip de 40 patas.

FUENTES DE ALIMENTACION:
Solo una fuente de +5 Volts.

RELOJ: Tiene el generador de reloj incluido y solo necesita un cristal externo o un circuito oscilador RC. Frecuencia de 3Mz.

CICLOS DE MAQUINA:
Consiste de 3, 4 o 6 ciclos de reloj. Existen los siguientes ciclos de máquina: 1. WRITE MEMORY, 2.READ MEMORY, 3.INPUT, 4.OUTPUT, 5.FETCH, 6.INTERRUPT, 7.HALT, 8.HOLD y 9.RESET.

CICLOS DE INSTRUCCION:
De cuatro a 18 ciclos de reloj.

TAMAÑO DE LA INSTRUCCION:
De 8, 16 y 24 bits (1, 2 y 3 palabras).

REGISTROS: Hay 6 registros de 8 bits arreglados en pares BC; DE; y HL. Tambien está el acumulador A de 8 bits el del apuntador al stack de 16 bits y el registro de banderas.

STACK: Externo de 64K bytes.

ESPACIO DE MEMORIA:
Direccionamiento de 64K bytes en modo directo.

BANDERAS: Las mismas que en el 8080.

ENTRADA/SALIDA: Transferencia de datos de 8 bits en paralelo y además cuenta con un puerto serial para entrada y salida.

SISTEMA DE INTERRUPCIONES:
Vectorizada con 12 niveles. Tiene 3 interrupciones con máscara programable, una no enmascarable(TRAP) para condiciones catastróficas,como falla de potencia.

CONJUNTO DE INSTRUCCIONES:
Consiste de 80 instrucciones del tipo de las del 8080 y además algunas para el manejo de interrupciones y para las máscaras de las interrupciones enmascarables. Realiza operaciones de suma y resta entre registros en un tiempo de 1.3 microsegundos.

TIPO DE ARITMETICA:
Decimal (BCD), binaria (en complemento a dos) y de doble precisión (operandos de 16 bits).

MODOS DE DIRECCIONAMIENTO:
Inmediato; Inmediato Extendido (igual que el inmediato pero con operandos de 16 bits); direccionamiento Directo; Directo por Registr; direccionamiento Extendido; direccionamiento de Página Cero (utilizado solo para la instrucción RST).

Este micro es compatible en el conjunto de instrucciones con el 8080, presenta mejoras como control de sistema con información disponible de los ciclos de máquina para control de un sistema mayor. Presenta tambien cuatro entradas de interrupción y además una interrupción compatible con el 8080. Otra diferencia está en el reloj que es de una fase.

3.3.3.3. EL MICROPROCESADOR Z80

Este micro es uno de los mas populares que existen en el mercado, presenta mejoras sustanciales con respecto al 8080 (en realidad es una mejora del 8080) y en otras con el 8085.

En la figura 3-8, se muestra la estructura funcional del CPU Z80. Este micro presenta un conjunto de registros mayor a los del 8080 y 8085 pues se añaden dos registros mas para manejo de índices, un registro para la identificación de dispositivos que pudieran interrumpir y un registro especializado para producir las secuencias de "refrescamiento" en las memorias dinámicas asociadas con este micro. Tiene ademas de los registros A, F, BC, DE, y HL del 8080 otro grupo de registros "espejo" de estos.

Las caracteristicas de este micro son las siguientes:

TECNOLOGIA: NMOS en chip de 40 patas.

FUENTES DE ALIMENTACION:
Solo una de +5 Volts.

RELOJ: Externo de una fase, con un circuito convencional de cristal o RC y frecuencia de 4 MHz.

CICLOS DE MAQUINA:
Cada ciclo de máquina está compuesto de tres a seis ciclos de reloj. Existen siete tipos de ciclos de máquina: 1. FETCH, 2. MEMORY READ o

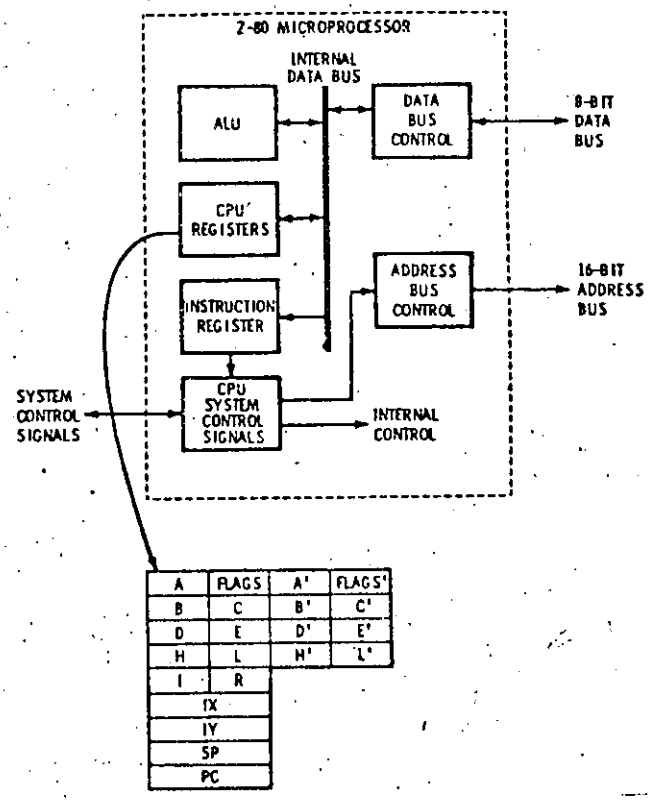


Figura 3-8: Diagrama de los bloques funcionales del CPU Z80.

MEMORY WRITE, 3. INPUT o OUTPUT 4. BUS REQUEST/ACKNOWLEDGE, 5. INTERRUPT REQUEST/ACKNOWLEDGE, 6. NONMASCARABLE INTERRUPT REQUEST/ACKNOWLEDGE, 7. EXIT.

CICLO DE INSTRUCCION:
Consiste de uno a seis ciclos de máquina (con excepción de instrucciones relacionadas con movimiento de datos).

TAMAÑO DE LA INSTRUCCION:
De 8 y 16 bits (1 y 2 palabras).

REGISTROS: Consiste de 14 registros de propósito general de 8 bits llamados A,B,C,D,E,H y L y A',B',C',D',E',H' y L' (espejos), tambien existe el registro de banderas

F y F'. Se pueden utilizar en pares de BC, DE y HL así como sus correspondientes primos (espejos). Estos registros pueden intercambiar sus contenidos con sus correspondientes primos. Tambien hay registros de propósito especial llamados I (usado para formar los 8 bits mas significativos de un apuntador a una dirección en un vector de direcciones de rutinas de servicios); 2 registros IX e IY de 16 bits cada uno (para manipulación de datos organizados en tablas, así como para implementación de código relocalizable); un registro R para el manejo automático de "refresco" en memorias dinámicas; y un registro del apuntador del stack llamado SP.

STACK: De 64K bytes, externo en memoria RAM.

ESPACIO DE MEMORIA:
Se pueden accesar hasta 64K bytes de memoria con modo directo.

BANDERAS: Tiene 7 banderas en un registro de 8 bits llamado F. Estas banderas son la de signo (S), la de cero (Z), bandera de medio carry en operaciones BCD (H), bandera para paridad/sobreflujo (P/V), bandera de substracción en operaciones de resta BCD (N) y la bandera de acarreo del bit de mayor orden del acumulador (C).

ENTRADA/SALIDA: Transferencia de datos de 8 bits entre el acmulador y dispositivos de E/S por medio de instrucciones especiales (IN y OUT).

SISTEMA DE INTERRUPCIONES:
Vectorizada, enmascarable y no enmascarable, con un alto grado de sofisticación en relación con los micros presentados.

CONJUNTO DE INSTRUCCIONES:
Las instrucciones se pueden agrupar en aquellas de carga de datos de 8 y 16 bits; de intercambio, transferencia de bloques de datos y busquedas, de aritmética de 8 y 16 bits; lógicas; orientadas a manejo de bits; de transferencia y control de programa y de entrada/salida.

TIPO DE ARITMETICA:
Binaria,decimal y de doble precisión.

MODOS DE DIRECCIONAMIENTO:
Tiene modos Inmediato; Inmediato Extendido;

Direccionamiento por Registro; Extendido; Indirecto; Relativo; Modificado para Pagina Cero; y direccionamiento por Bit.

Las caracteristicas que lo hacen superior a los micros 8080 y 8085 es su conjunto de registros, un mejor manejo de interrupciones, un mayor número de instrucciones y más modos de direccionamiento (son extensiones de los básicos, pero de manera explícita).

Notese que los micros 8080, 8085 y Z80 tienen la misma filosofía de organización funcional pero con ventajas sustanciales en el último.

3.3.3.4. EL MICROPROCESADOR 6800

Este micro se muestra en la figura 3-9, en el se muestran el MAR o Output Buffers de 16 bits, el MBR o Data Butter, el IR, la UC, el PC y el Stack Pointer compuestos de dos registros de 8 bits cada uno (uno para la parte mas significante y el otro para los bits menos significantes), un par de acumuladores, el ALU y el FCW llamado Condition Code Register. A continuación se muestran las caracteristicas importantes de este microprocesador:

TECNOLOGIA: NMOS, en cnip de 40 patas.

FUENTES DE ALIMENTACION:
Solo una fuente de +5 Volts.

RELOJ: Externo, de dos fases, por medio de un circuito generador de tiempos (MC6870). Y con frecuencia de 1 MHz.

TAMAÑO DE LA INSTRUCCION:
De 8 16 y 24 bits (1, 2 y 3 palabras).

REGISTROS: Tiene solo dos registros de proposito general aritmeticos de 8 bits cada uno llamados A y B. Tiene tambien un registro índice para manejo de datos de 16 bits, y un registro para el apuntador al stack de 16 bits, divididos cada uno de estos en dos de 8 bits.

STACK: Es externo y de tamaño máximo de 64K bytes.

ESPACIO DE MEMORIA:
Se pueden direccionar 64K bytes de memoria con modo directo.

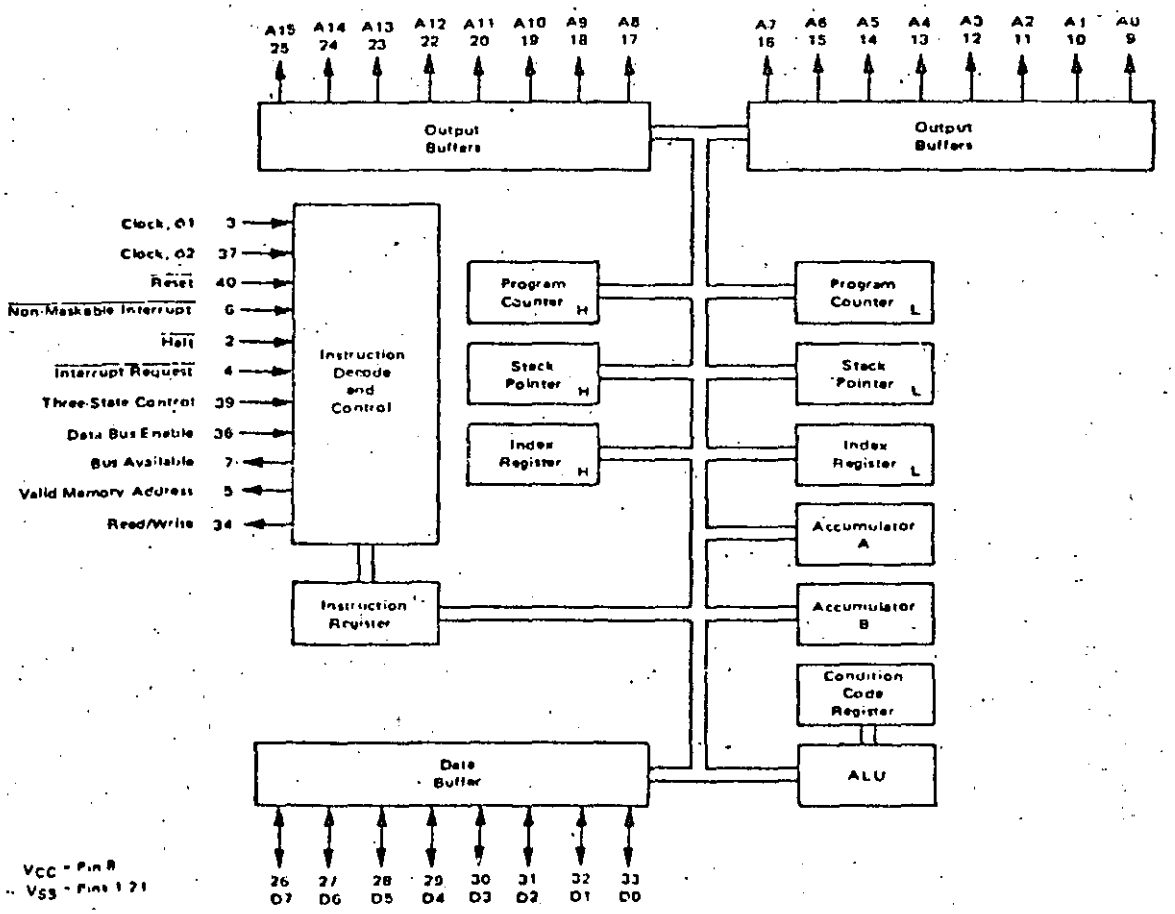


Figura 3-9: Diagrama de los bloques funcionales del CPU 6800.

BANDERAS: Tiene seis banderas, una para el acarreo (C), otra para marcar el sobreflujo (V), para el cero (Z), para indicar resultado negativo (N), para interrupción (I) y para medio acarreo en operaciones BCD (H).

ENTRADA/SALIDA: Transferencia de datos con tamaño de 8 bits.

SISTEMA DE INTERRUPCIONES:
Es Vectorizada enmascarable y una no enmascarable.

CONJUNTO DE INSTRUCCIONES:
Tiene 72 instrucciones agrupadas como aritméticas,logicas, de control y transferencia, para el manejo de interrupciones y manejo del stack.

TIPO DE ARITMETICA:
Puede efectuar aritmética binaria y decimal.

MODOS DE DIRECCIONAMIENTO:
Hay 7 modos: Direccionamiento por el acumulador; Inmediato; Directo; Extendido; Indexado, Relativo e Implicado.

Este micro efectúa operaciones de suma y resta en 5 microsegundos (es mas lento que los anteriores), tiene tambien la deventaja de poseer unicamente dos registros aritméticos (A,B) y solo un registro índice (IX). No presenta capacidad de autoincremento, por lo que es necesario hacerlo con instrucciones separadas. Sin embargo tiene un amplio repertorio de modos de direccionamiento y sus sistema de interrupciones es mas completo que el del 8080.

3.3.3.5. EL MICROPROCESADOR 6502

Este microprocesador es muy semejante al 6800, solo que en lugar de tener dos registros aritméticos y uno índice, tiene solo un acumulador y dos registros índices. El diagrama del micro 6502 se muestra en la figura 3-10. Este microprocesador presenta las siguientes caracteristicas:

TECNOLOGIA: NMOS en chip de 40 patas.

FUENTES DE ALIMENTACION:
Una fuente de +5 Volts.

RELOJ: Tiene un generador de reloj integrado, el cual solo necesita un circuito externo oscilador o una red RC. La frecuencia es de 1 MHz.

CICLO DE MAQUINA:
El numero mínimo de ciclos de reloj para formar un ciclo de máquina es de 2.

TAMAÑO DE LA INSTRUCCION:
Puede ser de 8, 16 y 24 bits (1, 2, o 3 palabras).

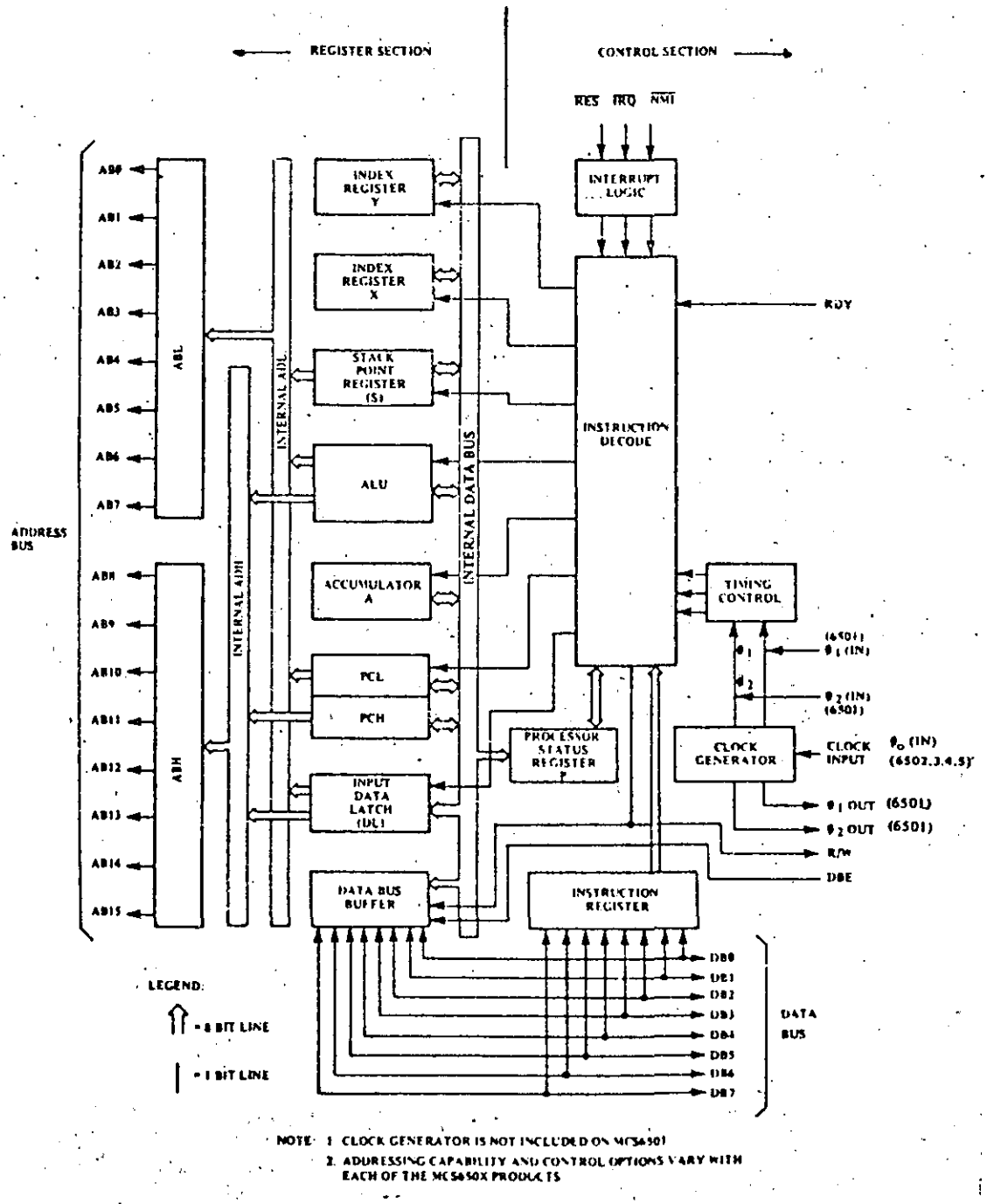


Figura 3-10: Diagrama de los bloques funcionales del CPU 6502.

REGISTROS: Tiene dos registros índices de 8 bits cada uno llamdos X y Y, un acumulador como registro aritmético de propósito general A y el apuntador al stack de 8 bits.

STACK: Este es reducido de tamaño máximo de 256 bytes.

ESPACIO DE MEMORIA:
Puede direccionar hasta 64K bytes de memoria con direccionamiento directo.

BANDERAS: Tiene 7 banderas cada una de un bit. Y son: de acarreo (C), de resultado cero (Z), para desnabilitar interrupciones (I), para aritmética decimal (D), para un "break" (B) (solo es prendida por el micro y es usada para determinar si durante un servicio de interrupción, fué esta causada por el comando break o por una interrupción real), una bandera para sobreflujo (V) y una bandera de resultado negativo (N).

ENTRADA/SALIDA: La transferencia de datos en E/S es de tamaño de 8 bits.

SISTEMA DE INTERRUPCIONES:
Es del tipo "poleadas", con dos lineas, enmascarable y no enmascarables.

CONJUNTO DE INSTRUCCIONES:
Consiste de 64 instrucciones y son del tipo aritmeticas, lógicas, de transferencia y control de programa, pero no hay para manejo de interrupciones.

TIPO DE ARITMETICA:
Binaria y decimal (BCD).

MODOS DE DIRECCIONAMIENTO:
Tiene 9 modos los cuales son: Inmediato; Absoluto; Página cero; Implícito; Relativo; Indexado Absoluto; Indexado en Página cero; Indirecto Indexado y Indexado Indirecto.

Este micro presenta una diferencia sustancial con respecto a los anteriores y es el de su manejo de interrupciones pues lo hace de manera "poleada". El tiempo para efectuar una suma o una resta es de dos microsegundos.

3.3.4. MICROPROCESADORES DE 16 BITS

Ahora se procederá a revisar las caracteristicas de los micros de 16 bits y que no presentan los micros de 8 bits. Por su importancia se describiran de manera general los micros: 8086 de Intel, Z8000 de Zilog, MC68000 de Motorola y el NS16000 de National Semiconductors.

3.3.4.1. ESTRUCTURA INTERNA

El 8086 de Intel, básicamente es una versión mejorada del 8080/8085 a nivel de 16 bits. Las instrucciones son orientadas por byte y se pueden ejecutar todas las instrucciones del 8080 y 8085. Internamente se compone de dos unidades,la unidad de ejecución y la unidad de interfase al bus, ver la figura 3-11. Una de sus mejores caracteristicas es la de manejar una cola de 6 bytes para instrucciones próximas a ejecutarse, esta cola alimenta instrucciones a la unidad de ejecución en segmentos de 8 bits. El bus interno es de 8 bits, mientras que, el del ALU es de 16 bits.

El Z8000 de Zilog no es una mejora del Z80 sino que, tiene una estructura interna totalmente distinta, no son compatibles ni en código ni en instrucciones. Existen dos versiones de este micro, el Z8001 o versión segmentada que permite un manejo sofisticado de memoria y el Z8002 o versión reducida no segmentada, que tiene un espacio de direccionamiento de 64K bytes, por lo que, tiene una capacidad semejante a los micros de 8 bits. Sin embargo, por lo demás son compatibles totalmente, incluso el Z8001 puede ejecutar código de Z8002 operando en modo no segmentado. Sus trayectorias internas de datos e instrucciones son de 16 bits, aunque puede manejar bytes, las instrucciones son orientadas por palabras de 16 bits. En la figura 3-12 se muestra el diagrama de bloques del micro.El 68000 de Motorola es arquitectónicamente distinto de los anteriores y de los micros de 8 bits ya que está totalmente microprogramado. Su estructura es mas simple y se muestra en la figura 3-13, su operación se centra alrededor de una unidad de ejecución de microprograma controlada. El tamaño del almacen de control se minimiza a través del uso de una estructura de control de dos niveles. En el nivel uno, las instrucciones de máquina se producen por secuencias de "microinstrucciones" en el almacen del microcontrol. Estas microinstrucciones son apuntadores a "nano-instrucciones" en el "nano-almacén" del nivel dos. El nao-almacén contiene palabras de control las cuales controlan la unidad de ejecución.

La arquitectura del microprocesador NS16032, se muestra en la figura 3-14, en el se contempla un mecanismo de anticipación el cual pre-almacena la secuencia de la instrucción en una cola de 8 bytes, mientras se extrae la instrucción anterior de la cola

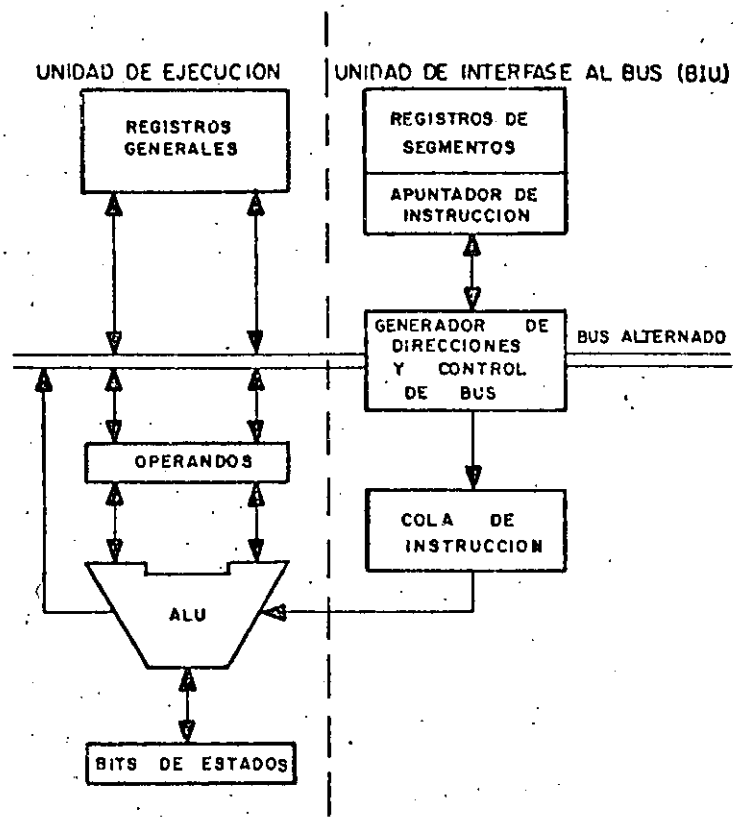


Figura 3-11: Estructura interna del microprocesador Intel 8086 (iAPX 86).

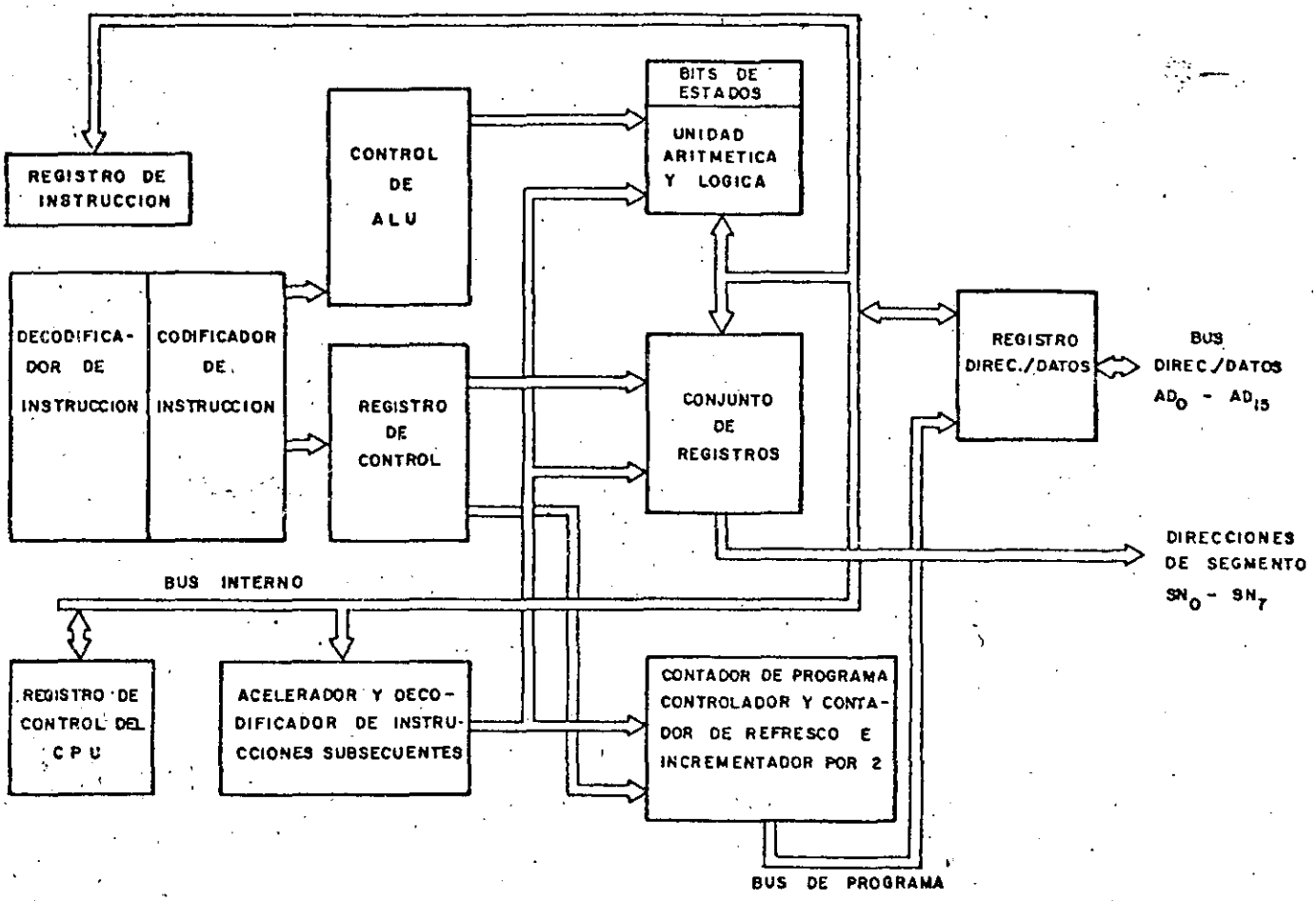


Figura 3-12: Estructura interna del microprocesador Zilog Z8000.

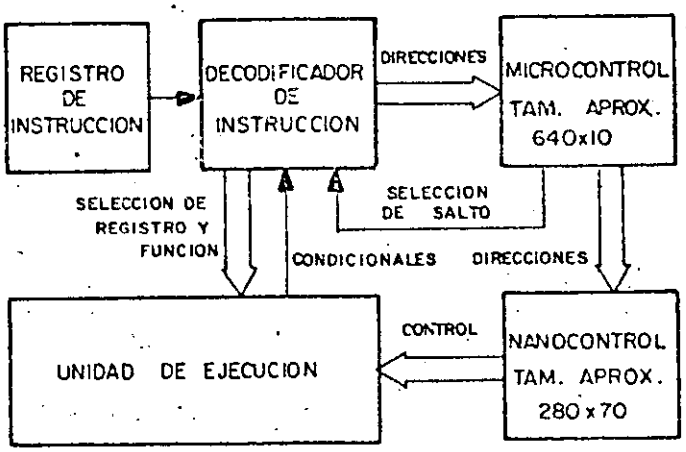


Figura 3-13: Estructura interna del microprocesador Motorola MC68000.

para su decodificación simultáneamente se estia terminando de ejecutar la instrucción más anterior. Por lo tanto, tres instrucciones sucesivas pueden ser procesadas simultáneamente por el procesador. La arquitectura representa una máquina de dós operandos donde cada operando puede ser direccionado por todos los modos de direccionamiento. Se implementado un "microcódigo" con técnica de dos niveles para compartir las partes comunes, rutinas de cálculo de la dirección efectiva, de todas las instrucciones y evitar de esta manera pérdidas de tiempo en llamadas a subrutinas y espacios en memoria al tener que repetir partes del microcódigo. El nivel superior es un preprocesador que controla el cálculo de la dirección efectiva del operando u operandos y las secuencias de ejecución, mientras que, el nivel interior concierne a los detalles de los pasos de ejecución. Internamente el 16032 es una máquina de 32 bits puesto que maneja buses de datos, registros de 32 bits y ALU de 32 bits, con lo que se pueden efectuar operaciones artméticas de 2 operandos de 32 bits ya sea binario o BCD.

3.3.4.2. MODOS DE OPERACION

Una característica específica de los microprocesadores de 16 bits y no encontrada en ningún micro de 8 bits es el manejo de dos modos de operación. Esta característica es muy importante para los sistemas operativos de multitarea y multiusuario ya que uno de los modos está reservado para el sistema operativo de la máquina y el otro para los usuarios. Cada modo tiene sus propios registros, su propio status y su propia memoria; existen algunas

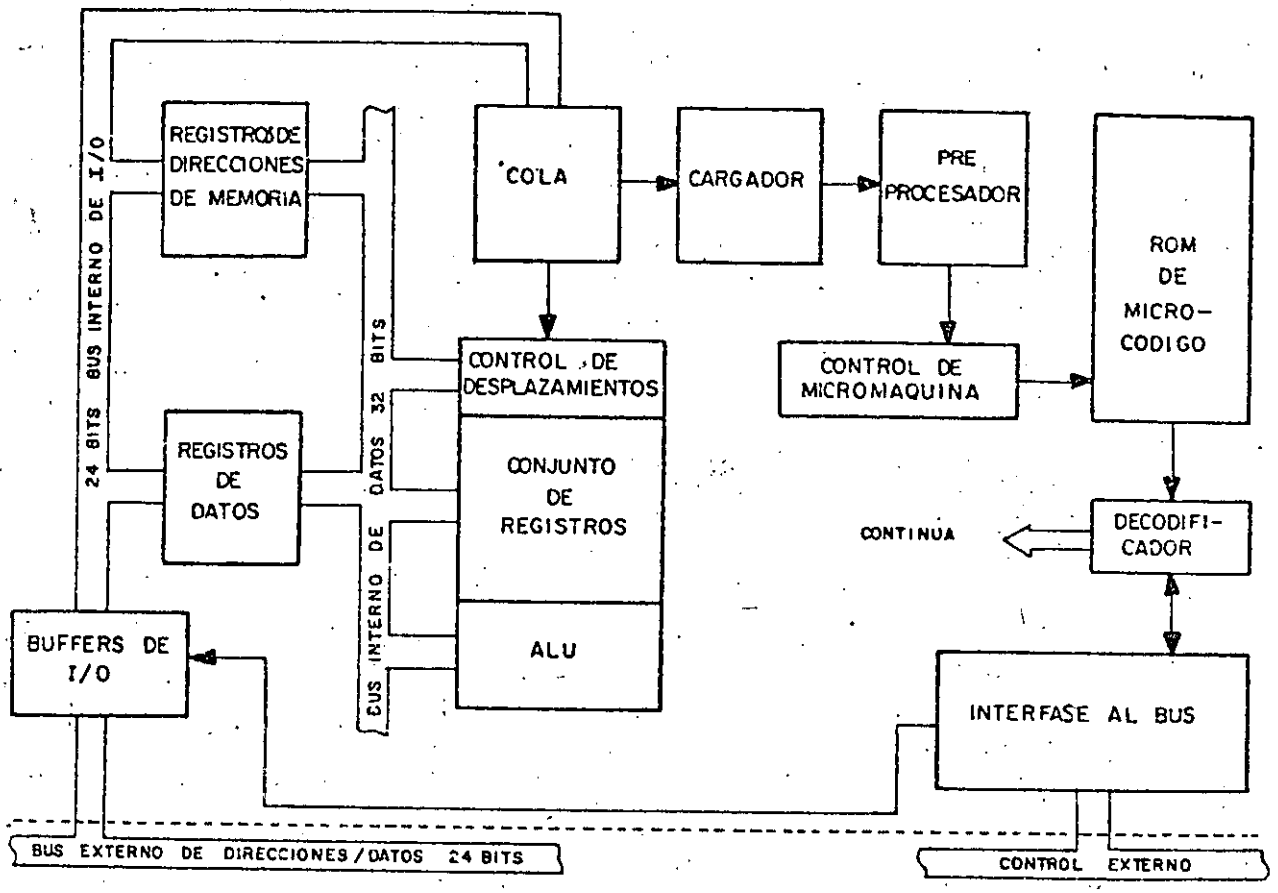


Figura 3-14: Estructura internal del microprocesador National Semiconductor NS16000.

partes privilegiadas del micro (instrucciones, direccionamiento, etc.), las cuales son usadas exclusivamente por el sistema operativo. En los micros de 8 bits no existen recursos específicos de "hardware" para el sistema operativo, todas las medidas de protección y reserva deben tomarse por "software", mientras que en los micros de 16 bits parte de esas medidas están concebidas en el hardware.

Esta es una de las característica más importantes en los micros de 16 bits que ha cambiado totalmente el enfoque de aplicaciones y utilidad de los microprocesadores, ya que de ser unos simples controladores de equipo, computadoras pequeñas monousuarias han pasado ahora a ser usadas en aplicaciones de mayor alcance y a competir con las minicomputadoras. El 8086, sin embargo, tiene un solo modo de operación.

El 28000 tiene dos modos de operación el modo sistema y el modo normal. El sistema operativo se ejecuta en el modo sistema desde donde se pueden manejar todos los recursos del procesador, existen instrucciones privilegiadas que solo se pueden usar en modo sistema, intentar usar estas instrucciones en modo normal ocacionan un trap interno. Por haroware existe una señal que indica el modo en el que se encuentra operando el procesador, usando esta señal se puede duplicar la memoria y usar una parte para sistema y otra para normal.

El 68000 tiene dos modos principales de operación el modo supervisorio y el modo usuario. El sistema operativo se ejecuta en el modo supervisorio. Existe un modo alterno que es el modo de depuración o de seguimiento (modo trace) el cual es una herramienta integrada de ayuda para la depuración de errores. Este modo permite seguir un programa instrucción por instrucción y observar su comportamiento con fines de detección de errores. El modo trace resulta en un trap después de la ejecución de cada instrucción. El modo trace está disponible en modo supervisorio o modo usuario, pero al presentarse el trap se pasa inmediatamente al modo supervisorio.

El 16000 tiene dos modos de operación el modo supervisor y el modo usuario. El sistema operativo se ejecuta en modo supervisor y existen 11 instrucciones privilegiadas que sólo se ejecutan en este modo.

3.3.4.3. REGISTROS INTERNOS

La estructura de los registros internos ha cambiado totalmente de la de los micros de 8 bits; los micros de 16 bits se caracterizan por tener mayor cantidad de registros y una organización regular en estos. Existen registros de propósito general y registros de uso específico, en algunos casos existen registros duplicados, uno para el sistema operativo y otro para los usuarios. Los registros de proposito general pueden ser agrupados para manejar datos de diferente tamaño (8, 16, 32 o 64 bits), cosa que en los micros de 8 bits a lo sumo se llegaban a manejar datos de 16 bits.

El 8086 mantiene la estructura de los registros internos semejante a la del 8080, 8085 y Z80. Tiene 4 registros de 16 bits manejables por byte que pueden considerarse de propósito general aunque uno de ellos, el acumulador, tiene mayores prerrogativas que los otros 3. Los otros 10 registros son de 16 bits y de propósito especial: apuntador al stack, apuntador base, índice fuente, índice destino, apuntador a instrucción, status y 4 para el manejo de memoria segmentada, que son: de código, dato, stack y extra.

El Z8000 tiene 16 registros de próposito general de 16 bits que pueden ser agrupados para manejar datos de 8, 16, 32 o 64 bits. Uno de estos registros se comporta como un registro doble que sirve como apuntador al stack del sistema y apuntador al stack normal. Tiene los siguientes registros de propósito específico: contador de programa, palabra de status y control, apuntador al área del nuevo status del programa (NPSAP) y contador de refresco para memorias dinámicas.

El 68000 tiene 8 registros de 32 bits para datos y 8 registros de 32 bits para direcciones, el último registro de las direcciones sirve como apuntador al stack y es un registro doble uno se usa para el stack del modo usuario y el otro para el stack del modo supervisorio. Además tiene el contador de programa (32 bits) y el registro de status (16 bits). Los registros no pueden agruparse para manejar datos de 64 bits, pero si pueden ser usados para datos de 16 bits o bytes.

La arquitectura del 16000 maneja 8 registros de próposito general de 32 bits cada uno y 8 registros de próposito específico, que son: contador de programa (24 bits), registro de base estática (24 bits), apuntador al bloque de stacks (24 bits), apuntador al stack de usuario (24 bits), apuntador al stack de interrupciones (24 bits), registro base de interrupciones (24 bits), registro de status (16 bits) y registro módulo (16 bits).

3.3.4.4. TIPOS DE DATOS

Se ha puesto mucho énfasis en el manejo de los datos de los micros de 16 bits. Se pueden operar desde bits hasta palabras de 64 bits, las cadenas secuenciales de datos son muy comunes en estos micros.

El 8086 maneja los siguientes tipos de datos: dígitos BCD, conversión automática a/de ASCII, bytes, palabras de 16 bits, y puede manejar cadenas secuenciales de bytes.

El Z8000 maneja los siguientes tipos de datos en memoria:

- bits
  Fija, quita o prueba cualquier bit.

- dígitos BCD (4 bits)
  Para operaciones aritméticas.

- bytes (8 bits)
  Para caracteres o pequeños valores enteros.

- palabras (16 bits)

Para enteros, direcciones no segmentadas e instrucciones.

- palabras dobles (32 bits)
  Para enteros muy grandes y direcciones segmentadas.

- cadenas secuenciales de bytes (máximo 64K bytes).

- cadenas secuenciales de palabras (máximo 64K bytes).

El 68000 puede manejar los siguientes 5 tipos de datos: bits dígitos BCD (4 bits), bytes (8 bits), palabras (16 bits) y palabras grandes (32 bits).

El 16000 por sí solo puede manejar los siguientes tipos de datos: bits, dígitos BCD, bytes, palabras (16 bits), palabras dobles (32 bits) y palabras cuádruples (64 bits). Además, añadiendo un chip adicional, llamado procesador de punto flotante, puede manejar conjuntamente con este chip datos de punto flotante: precisión simple (32 bits) y doble precisión (64 bits). Finalmente tiene todas las facilidades para manejar arreglos de datos de los siguientes tipos:

- Bytes o enteros pequeños.

- Palabras o enteros medianos.

- Palabras dobles o enteros de doble precisión o datos de punto flotante de precisión sencilla.

- Palabras cuadruples o datos de punto flotante doble precisión.

3.3.4.5. MODOS DE DIRECCIONAMIENTO

Existe una mayor cantidad de modos de direccionamiento en los micros de 16 bits que en los de 8 bits. En este caso se manejan dos tipos de modos de direccionamiento los que están explícitos en la misma instrucción y los implícitos. En los micros de 8 bits, prácticamente, se manejan solo los explícitos.

El 8086 tiene 24 modos de direccionamiento de los operandos.

En el Z8000 existen 8 modos de direccionamiento que están explícitamente especificados en la instrucción. Autoincremento y autodecremento son los dos modos de direccionamiento implícitos para instrucciones de bloque o cadenas de datos. Dentro de los modos explícitos hay 5 principales y 3 secundarios. Los modos

principales son profusamente manejados en casi todas las instrucciones y son:

- Registro (R).
- Registro indirecto (IR).
- Directo (DA).
- Inmediato (IM).
- Indicado (X).

Los modos secundarios utilizados solo en algunas instrucciones son:

- Dirección base (BA).
- Base Indicada (BX).
- Dirección relativa (RA).

El 68000 tiene 14 modos de direccionamiento de los cuales 6 son de tipo básico:

- Directo de registro.
- Indirecto de registro.
- Absoluto.
- Imediato.
- Relativo al contador de programa.
- Implicado.

El 16000 tiene la característica novedosa de ser una máquina de 2 operandos, donde cada uno de ellos puede ser direccionado por 9 modos de direccionamiento, 4 son comunes en los otros micros y los 5 restantes son modos de direccionamiento especiales no encontrados en los otros micros de 16 bits que ayudan al desarrollo de software de alto nivel. Los 4 modos comunes son:

3.3.4.6. CONJUNTO DE INSTRUCCIONES

El conjunto de instrucciones de los micros de 16 bits es mayor que el de los micros de 8 bits. Se busca mucho la simetría entre las instrucciones, modos de direccionamiento y códigos de condiciones. Una característica importante en el conjunto de instrucciones de los micros de 16 bits es el manejo escrupuloso de los bits pudiéndose acertar, negar, probar o prueba y pone cualquier bit; esta facilidad es muy importante para el manejo de periféricos y en sistemas de control donde continuamente hay que leer el estatus, monitorear alguna variable o enviar un comando por alguna señal. En operaciones aritméticas se incluyeron la multiplicación y división signadas. Existen instrucciones privilegiadas que sólo se ejecutan en un modo. Otra novedad son las instrucciones o ayudas de software para manejar un micro en medios ambientes de múltiples microprocesadores.

El 8086 tiene 86 instrucciones. Entre ellas ajustes de ASCII para resta, multiplicación y división. Tiene multiplicación y división no signadas e instrucciones de bloque más primitivas que el Z80. El 8086 tiene dos instrucciones para manejar dispositivos externos ESC y LOCK.

El Z8000 tiene 110 instrucciones que combinadas con los modos de direccionamiento resultan un total de 414. Para representar una instrucción se utilizan de 1 a 5 palabras según la complejidad de la instrucción. El espacio de direccionamiento de I/O es distinto al espacio de direccionamiento de memoria, se distinguen por las líneas de status. Existen dos tipos de instrucciones de I/O; cada uno con su propio espacio de direccionamiento:

- Normales, para dispositivos de I/O. Se usa el byte menos significante del bus de datos.
- Especiales, para cargar y descargar los MMUs. Se usa el byte más significante del bus de datos.

El Z8000 tiene las siguientes instrucciones privilegiadas que sólo se ejecutan en modo sistema y ocacionan un trap interno si se tratan de ejecutar en modo normal.

- Todas las instrucciones de entrada/salida.
- Halt.
- Habilitación y desnabilitación de interrupciones.

- Registro.
- Inmediato.
- Absoluto.
- Relativo al registro.

Los 5 modos especiales son:

- Espacio de Memoria, permite que las referencias sean relocalizables a áreas de memoria comúnmente usadas en lenguages de alto nivel.
- Parte superior del stack, cualquier operando puede ser referido al primer dato del stack, el dato es extraído del stack y el apuntador al stack puede o no ser modificado según la necesidad de la instrucción.
- Relativo a memoria, útil para manejar apuntadores, este modo usa dos desplazamientos, el primero es añadido a uno de los registros dedicados, especificado por el modo, y el segundo es un desplazamiento inmediato.
- Indexado escalado, calcula la dirección efectiva añadiendo al contenido de uno de los registros de propósito general la dirección base multiplicada por 1, 2, 4 u 8, muy útil para manejar arreglos de datos de doble precisión o de punto flotante, ya que los elementos del arreglo pueden ser manejados como bytes, palabras, doble palabra o palabras cuadruples directamente.
- Modo de direccionamiento externo, facilita que los módulos sean relocalizados sin necesidad de ligarse. Este modo es usado para referir operandos externos al módulo de ejecución en ese momento. Asociado a cada módulo existe una tabla de ligado que contiene las direcciones absolutas de las variables externas y las direcciones relativas de los operandos a ser accesados por otros módulos. El modo de direccionamiento externo especifica dos desplazamientos: el número ordinario de la variable externa y el corrimiento de la variable referenciada.

- Cargado de la palabra de control.
- Cargado del nuevo status del programa.
- Regreso de interrupción.
- Todas las instrucciones para operación con múltiples micros.

Tiene multiplicación y división signadas, acertar, negar, probar o prueba y pone cualquier bit y una amplia variedad de instrucciones de bloque.

El 68000 tiene 68 instrucciones, algunas de ellas operan con varios modos de direccionamiento y diferentes códigos de condiciones. Tiene multiplicación y división por hardware, varias instrucciones de prueba de bits, como: prueba bit, prueba bit y acierta, prueba bit y niega y prueba bit y cambia. Tiene tres instrucciones para la generación de traps por sortware y 8 instrucciones privilegiadas que son:

- Reset a dispositivos externos.
- RTE retorno de excepciones.
- STOP detener la ejecución de programas.
- OR, AND y EOR del registro de status.
- Mover el apuntador del stack de usuario.
- Cargar un nuevo registro de status.

El conjunto de instrucciones de la micro NS16000 incluye más de 100 tipos de instrucciones básicas codificadas en códigos de máquina de longitud variable. El tamaño de ese código es de uno a 3 bytes de longitud. Existen instrucciones que usan hasta 5 operandos con uno a tres desplazamientos (de uno a cuatro bytes cada uno). Los códigos de instrucción fueron cuidadosamente asignados, de tal forma que las instrucciones usadas con mayor frecuencia tienen códigos muy cortos, mientras que las usadas pocas veces utilizan códigos más largos. En adición a las instrucciones convencionales de todo micro tales como, movimiento de datos, operaciones lógicas y aritméticas, y corrimientos en todas las direcciones (en el caso del 16000 con la capacidad de efectuarlo de memoria a memoria), la arquitectura del 16000 incluye instrucciones avanzadas que son muy útiles en medios ambientes de lenguajes de alto nivel. Algunas de estas instrucciones son:

- CHECK, que determina si el indice de un arreglo está dentro de sus fronteras, caso contrario lo ajusta a su valor cero del arreglo.

- INDEX, implementa pasos de indexado recursivo, útil para arreglos de dimenciones múltiples.

- STRING, maneja cadenas de datos con traducción opcional, verifica si son caracteres <ESC>, <CR>, <LF>, etc.

- CXP, permite llamadas automáticas a rutinas externas por un simple "call external procedure".

- ENTER y EXIT minimizan los procedimientos de llamadas manejando los recursos (registros y stack frame) posicionados al comienzo de un procedure y reclamados al final del mismo.

- INTERLOCKED (prueba y coloca/niega) provee una primitiva para la implementación de semáforos que coordinen sistemas de multitareas y multiprocesamiento.

- PUNTO FLOTANTE, estas instrucciones se manejan con ayuda de un chip adicional y pueden manejar operaciones aritméticas de precisión simple (32 bits) o doble precisión (64 bits).

3.3.4.7. INTERRUPCIONES Y TRAPS

Los micros de 16 bits tienen un esquema sofisticado para el manejo de las interrupciones, a diferencia de los métodos muy simples de los micros de 8 bits. Una característica nueva, no conocida en los micros de 8 bits, es el manejo de traps. Las interrupciones son eventos asíncronos mientras que los traps son eventos síncronos que se generan al presentarse alguna anomalía interna del procesador, en algunos casos puede ser una anomalía externa.

El 8086 tiene dos tipos de interrupciones, la interrupción no enmascarable y la interrupción enmascarable, con 128 niveles distintos para cada interrupción.

El Z8000 tiene tres tipos de interrupciones y dos tipos de traps, interno o externo. El orden de prioridad de las interrupciones y traps para que sean atendidas por el procesador es el siguiente:

- Trap interno. Se genera con cualquier intento de

ejecutar bien sea instrucciones privilegiadas en modo normal, instrucciones ilegales o la instrucción "Call System".

- Interrupción no enmascarable (NMI).
- Trap externo o trap de segmento. Se genera cuando se presenta alguna anomalía en el manejo de memoria.
- Interrupción enmascarable vectorizada (VI).
- Interrupción enmascarable no vectorizada (NVI).

El 68000 provee 7 niveles de prioridades de interrupción. Los dispositivos pueden ser encadenados externamente dentro de cada uno de los niveles de prioridades de interrupción, logrando de esta manera, que un ilimitado número de dispositivos periféricos puedan interrumpir al procesador. El nivel 7 es el de mayor prioridad y el de nivel uno es el de menor prioridad. Inclusive el mismo procesador tiene un nivel de prioridad que por software se programa en el registro de status, todo nivel igual o menor que el procesador se innibe. El 68000 provee dos tipos de traps, el trap de hardware y el trap de software, los traps de hardware se generan cuando se presentan condiciones anormales internas. El micro detecta las siguientes condiciones de error:

- Acceso a una palabra con dirección impar.
- Instrucciones ilegales.
- Instrucciones no existentes.
- Acceso ilegal a memoria (error de bus).
- División por cero.
- Sobreflujo al código de condiciones.
- Registro fuera de condición (instrucción CHK se usa para verificar fronteras de arreglos de datos).
- Interrupción espurea.

Adicionalmente se proveen 16 instrucciones de traps de software, las cuales pueden ser utilizadas por el programador en aplicaciones de detección y corrección de errores.

Las interrupciones en el NS16000 se manejan por medio de un

chip extra, especializado en el manejo de interrupciones NS16202. Los traps pueden ser internos o externos. Los traps internos se refieren a malfunciones del microcódigo, en cambio los traps externos los generan tanto el MMU en todo intento de accesar memoria incorrectamente como el procesador de punto flotante al encontrar incongruencia en las operaciones que trata de ejecutar.

3.3.4.8. ESPACIOS DE DIRECCIONAMIENTO

Una nueva característica de los micros de 16 bits es que tienen varios espacios de memoria totalmente distintos entre sí, lo que automáticamente amplia el espacio de direccionamiento del procesador. Los micros de 8 bits se caracterizan por tener un espacio de direccionamiento directo reducido, máximo 64K bytes, mientras que los micros de 16 bits manejan espacios de 2 o 3 órdenes de magnitud mayores, esto se debe a dos factores básicos:

1. El acelerado avance de la tecnología en memorias de semiconductores permite construir memorias de alta densidad a bajo costo.

2. Los requerimientos de sistemas operativos y compiladores modernos son de un gran espacio de memoria. Por ejemplo, para atender a una gran cantidad de usuarios se requiere un buen espacio de memoria.

El micro 8086 tiene un espacio de direccionamiento de 1M byte y tiene la capacidad de distinguir 4 espacios de memoria distintos que son: código, datos, datos alternos y stack.

El Z8000 tiene la capacidad de distinguir entre instrucciones datos y stack en ambos modos sistema y normal. Esta distinción la realiza por una combinación de estados en las líneas de status. A continuación en la figura 3-15, se mencionan los espacios de direccionamiento que tiene este micro y sus capacidades.

El 68000 puede distinguir referencias a memoria bien sea si son instrucciones o datos en ambos modos supervisor y usuario. Estas 4 áreas de memoria las distingue a través de las líneas de status. Por lo tanto, puede direccionar como máximo 64 M bytes, considerando las cuatro áreas de memoria.

El NS16000, tiene un solo espacio de direccionamiento de 16 M bytes donde efectúa todos los manejos de datos y ejecución de programas.

	Z8001		Z8002
	Directo	Virtual	Directo
Código en modo sistema	8 M	16 M	64 K
Instruc. en modo sistema	8 M	16 M	64 K
Stack en modo sistema	8 M	16 M	64 K
Código en modo normal	8 M	16 M	64 K
Instruc. en modo normal	8 M	16 M	64 K
Stack en modo normal	8 M	16 M	64 K
T O T A L	48 M	96 M	384 K

Figura 3-15: Modos de direccionamiento del Z8000 y sus capacidades en bytes.

3.3.4.9. MANEJO DE MEMORIA

Una de las características más importantes de los micros de 16 bits es la posibilidad de manejar la memoria en forma segmentada a través de un chip adicional llamado unidad manejadora de memoria (MMU). El MMU tiene la capacidad de relocalizar dinámicamente los segmentos con lo cual las direcciones de software del usuario quedan independientes de las direcciones físicas de almacenamiento, liberándolo de tener que especificar en que direcciones físicas se encuentra la información que necesita. Este manejo inteligente de memoria facilita el desarrollo de los sistemas de multi-programación/multi-usuario, la implantación de sistemas manejadores de bases de datos, el despliegue elegante de imagenes, etc.

Las funciones que normalmente realizan los MMUs son las siguientes:

- Validación de accesos a memoria, lo cual protege áreas de memoria de accesos no autorizados o no intencionales, característica muy importante cuando se atienden a varios usuarios.
- Alarma al usuario cuando llega al final del segmento. Los segmentos pueden ser de tamaño variable y cada acceso a memoria se verifica si pertenece al segmento previamente definido.
- Verifica el status del acceso, modo, si es instrucción, datos o stack, etc.

- Hay la posibilidad de proteger segmentos de escrituras, por lo que éstos se convierten en segmentos de lectura solamente, para estos segmentos el MMU verifica si el acceso es de lectura, caso contrario aborta el acceso.

En todo acceso a memoria que se pretenda realizar se verifican estos atributos, si alguno de ellos no se cumple se genera un trap de segmento (trap externo).

El chip 8086 es el único que tiene el sistema manejador de memoria integrado en el mismo chip del procesador, por lo que las direcciones que entrega son direcciones físicas siempre. La memoria puede ser dividida lógicamente en segmentos de código, dato, dato alterno y stack de 64 Kbytes cada uno.

El chip manejador de memoria (MMU) para el micro Z8001 es el Z8010. Tiene la capacidad de manejar como máximo 64 segmentos de tamaño variable múltiplo de 256 bytes, desde 256 bytes hasta 64K bytes. La traducción de direcciones virtuales a físicas se efectúa a través de una tabla donde se programan el tamaño del segmento, el origen o dirección base del segmento y el status de acceso del mismo. El diagrama de bloques del MMU se muestra en la figura 3-16, donde también se observa el procedimiento para calcular la dirección física en base a la dirección virtual compuesta del número de segmento y un desplazamiento dentro del mismo (offset). Tiene la capacidad de efectuar todos los anteriores atributos y el esquema de protección puede operar en cualquiera de las siguientes maneras lectura solamente, lectura/escritura, ejecución solamente o sistema solamente. Las tablas de traducción y protección son cargadas y descargadas como un periférico de I/O con las instrucciones especiales de I/O que usan el byte más significante del bus de datos. Ese cargado y descargado es dinámico y sucede a medida que las tareas son creadas, suspendidas o cambiadas. Dentro de un espacio de direccionamiento varios MMUs se pueden usar para crear varias tablas de traducción.

El MC68451 es el chip manejador de memoria para el MC68000, que entre sus características más sobresalientes están las siguientes:

- Manejar 16 Mbytes de un espacio de direccionamiento lógico.
- Separar los espacios de direccionamiento del usuario y del sistema.
- Separar los espacios de direccionamiento de programas y datos.

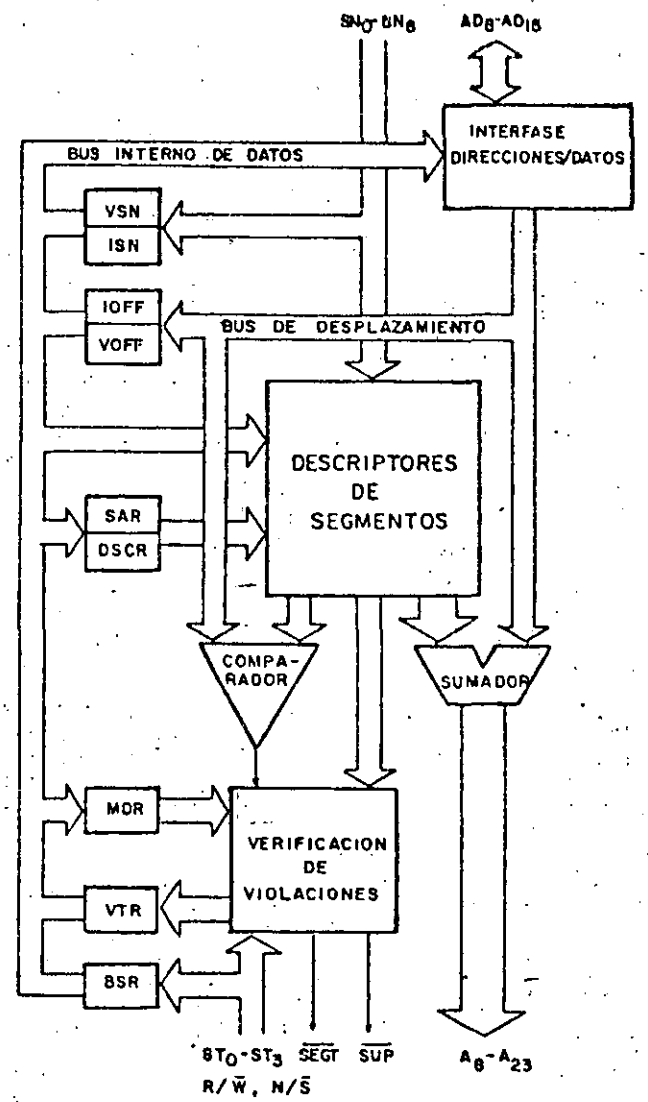


Figura 3-16: Diagrama de bloques de la Unidad de Manejo de Memoria (MMU) para el Z8000.

- Comunicación entre procesos a través de recursos compartidos.
- Capacidad para manejar ambos paginación o segmentación.
- Manejo de memoria virtual y sistemas masters de múltiples buses.
- Protección para segmentos de lectura solamente.
- Provisión para múltiples MMUs en un sistema.

Cada MMU puede manejar 32 descriptores de segmento, cada uno variando desde 256 bytes hasta 16 Mbytes.

Para transformar el micro NS16032 en una máquina de memoria virtual, el MMU 16082 es apareado con el CPU. El MMU opera como esclavo del CPU, se comunican a través de un protocolo complejo pero seguro. El MMU es un sistema de manejo de memoria paginado, con mecanismos de protección en las páginas y tiene una sección especial que facilita la depuración del software. El MMU se programa sólo en modo supervisor, se genera un trap al intentar programarlo en modo usuario. Se pueden manejar, en cualquier momento, uno o dos espacios de memoria con el MMU. En el modo de un solo espacio, cada usuario comparte un espacio de memoria virtual de 16 Mbytes con el sistema operativo. Ellos tienen tablas de traducción comunes. En el modo de doble espacio, cada usuario y el sistema operativo tienen espacios de memoria virtual separados de 16 Mbytes. La estructura del MMU no limita el número de usuarios. La estructura del MMU se muestra en la figura 3-17, cada página es de 512 bytes, existen 256 tablas de 128 apuntadores cada una, por lo tanto existen 256 x 128 = 32 K páginas en 16 M bytes. El acceso a estas tablas se efectúa a través de una tabla maestra de 256 apuntadores. La característica más importante es la capacidad de abortar instrucciones si se trata de accesar páginas protegidas, o páginas de lectura solamente, o páginas del sistema operativo, etc. La forma como el MMU contesta los abortos es durante la comunicación bidireccional que se lleva a cabo durante el protocolo, con lo cual el MMU tiene la posibilidad de indicar por la línea de datos que tipo de aborto se refiere.

3.3.4.10. AYUDAS PARA MULTI-MICROS

Los micros de 16 bits tienen recursos para facilitar la operación de los mismos en medios ambientes de múltiples procesadores. Estos recursos son tanto a nivel de hardware como a nivel de software, en hardware existen líneas para manejar buses de tiempo compartido y en software se dedican ciertas instrucciones para manejar estas líneas.

El Z8000 tiene dos patas para facilitar el manejo de buses de tiempo compartido, el multi-micro-output sirve para enviar un requerimiento por un recurso físico y el multi-micro-input es usado para reconcer el estado de ese recurso. De esta forma cualquier procesador en un sistema de múltiples microprocesadores puede utilizar un recurso compartido arbitrando su acceso por medio de estas ayudas o bien excluir a los otros procesadores de compartir un recurso crítico.

Ninguno de los otros micros 8086, MC68000 ni NS16000 tienen señales de hardware ni instrucciones de software que los ayuden para su operación en medios ambientes de múltiples procesadores.

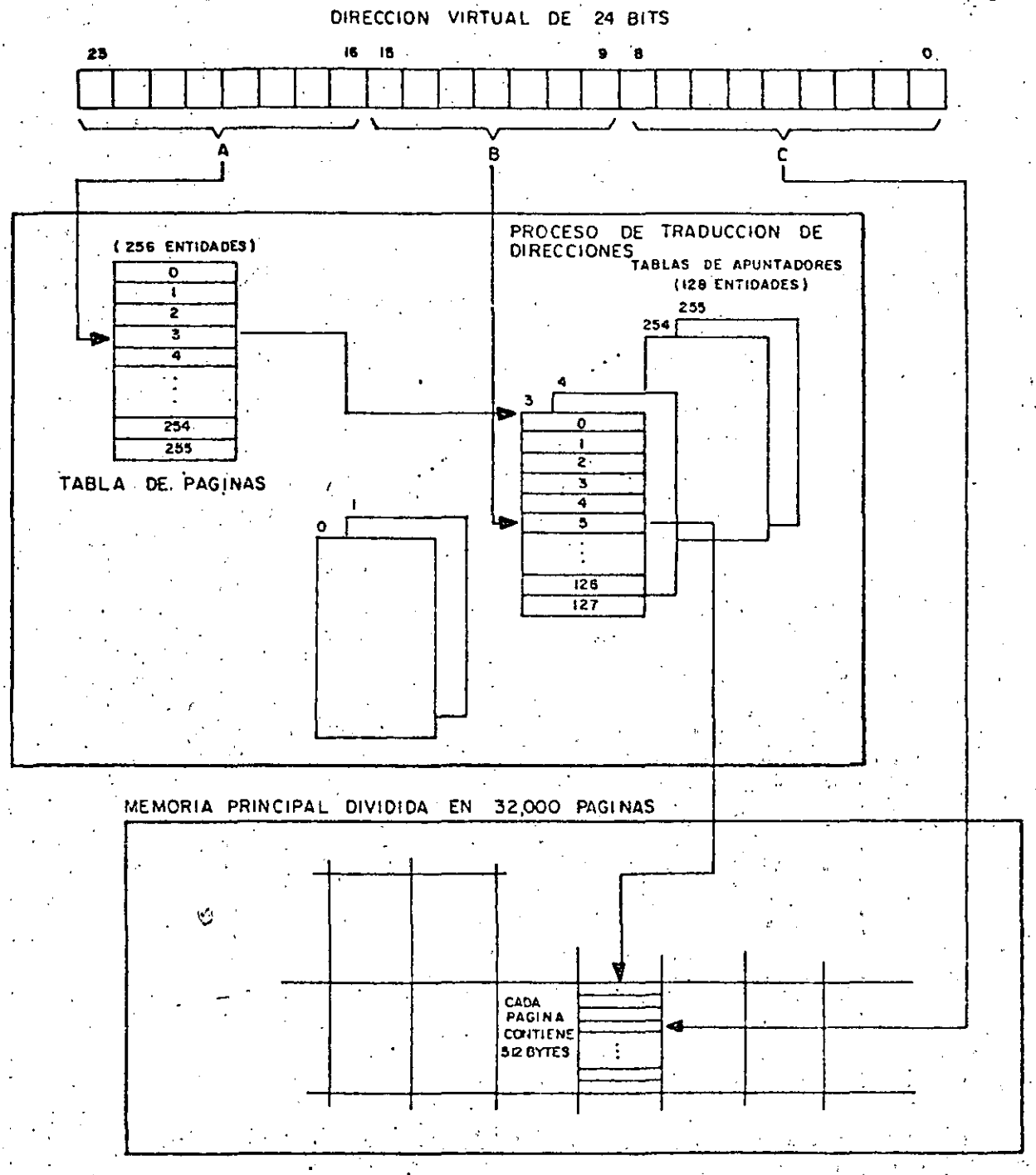


Figura 3-17: Estructura del manejo de paginas (MMU) para el NS16032

3.3.4.11. VELOCIDAD DE OPERACION

Los micros de 16 bits tienen código muy compacto, significa que sus instrucciones tienen un alto contenido de operaciones internas y una amplia gama de instrucciones, lo cual ocaciona que con muy pocas instrucciones se puedan desarrollar rutinas complejas. Existen algunas instrucciones de los micros de 16 bits que tienen que representarse por medio de una rutina de varias instrucciones en los micros de 8 bits, por ejemplo, las instrucciones de bloque, prueba y pone, etc.

Hay dos formas en que la velocidad de operación de los micros de 16 bits se incrementa:

1. En forma indirecta, al contar con código compacto y una amplia variedad de instrucciones.

2. En forma directa al aumentar la velocidad de ejecución de las instrucciones, es decir, al operar con relojes cada vez más rápidos. La unidad básica de tiempo es el ciclo de reloj, ya que un conjunto de ciclos de reloj forma un ciclo de máquina y un conjunto de ciclos de máquina forma un ciclo de instrucción.

El 8086 tiene densidad relativa en sus instrucciones, tarda mucho cuando se trata de datos de doble palabra (32 bits). Sin embargo, opera con relojes de muy alta velocidad, inicialmente 8086 5 MHz, 8086-4 4 MHz, y desde que cambio de nombre iAPX-86/10 tiene la capacidad de operar a 8 MHz.

El 28000 cumple con las dos formas de aumento de velocidad de operación, tiene código compacto y opera con relojes de alta velocidad.

En la tabla 3-1, se muestran los diferentes tiempos requeridos para los diferentes ciclos de reloj, máquina e instrucción, utilizando los modos de direccionamiento por registro y absoluto, en las diferentes versiones del 28000.

Se ha puesto gran énfasis en la densidad del código del 68000 con el propósito de aumentar la velocidad de operación y reducir la longitud de los programas, existen instrucciones de extra rápida ejecución y las operaciones de multiplicación y división normalmente tardadas en su ejecución, en este caso, han sido optimizadas en el microcódigo para que se ejecuten muy rapidamente. La frecuencia del reloj en el 68000 empezó en 5 MHz y actualmente es de 8 MHz.

El microprocesador de 16 bits más rápido de los cuatro

Micro	Frec. Max.	Ciclo reloj	Ciclo máq. Ciclo Instr Direc. Reg.	Ciclo Instr. Direc. absol.
Z8000	4 MHz	250 nseg	1 useg	2.5 useg
Z8000A	6 MHz	167 nseg	668 nseg	1.67 useg
Z8000B	8 MHz	125 nseg	500 nseg	1.25 useg

Tabla 3-1: Tiempos de los diferentes ciclos en el Z8000.

OPERACION	DATO	8086	Z8000	MC68000	NS16000
MOVIMIENTO reg -> reg	Byte/Palab	0.40	0.75	0.50	0.30
	Doble pal.	0.80	1.25	0.50	0.30
MOVIMIENTO mem -> mem	Byte/Palab	7.00	7.00	2.50	1.60
	Doble pal.	14.00	8.50	3.75	2.40
SUMA mem -> mem	Byte/palab	3.60	3.75	1.50	1.10
	Doble pal.	7.20	5.25	2.25	1.50
MULT mem -> mem	Byte	13.00	20.25	8.75	2.80
	Palabra	23.00	16.00	8.75	4.60
	Doble pal.	115.20	85.75	43.00	7.60
Salto Condicional	Efectuado	1.60	1.50	1.25	1.30
	No efect.	0.80	1.50	1.00	0.70
Salto a subrutina		3.80	3.75	2.25	2.50

Tabla 3-2: Tabla de comparación de velocidades de ejecución entre los micros 8086, Z8000, MC68000 y NS16000.

estudiados es el NS16000, ya que tiene un código muy compacto y opera con relojes de 10 MHz. Aún así, las operaciones de multiplicación y división son muy rápidas.

En la tabla 3-2, se muestra una comparación de las velocidades de ejecución en microsegundos de algunas operaciones como son el movimiento de datos entre registros, entre localidades de memoria; sumas y multiplicaciones entre operandos localizados en memoria y registro o bien ambos en memoria; tambien se muestran los tiempos para efectuar saltos (branchs o jumps).

3.3.4.12. CHIPS DE SOPORTE

Para facilitar su utilidad y aplicación de los micros de 16 bits, sus fabricantes ofrecen una serie de chips adicionales de soporte compatibles con las características de sus procesadores. Estos chips facilitan en forma considerable las siguientes tareas:

- Manejo de memoria.
- Manejo de periféricos.
- Ejecución de operaciones aritméticas complejas como operaciones de punto flotante.
- Interfase hacia el mundo real.
- Manejo de comunicaciones (transmisión de datos).

El éxito de un micro depende no solo de la arquitectura del mismo, sino de los chips de soporte que el fabricante pueda ofrecer.

CAPITULO 4
MICROCOMPUTADORAS

4.1. UNIDAD CENTRAL DE PROCESAMIENTO

La unidad central de procesamiento (CPU) de las microcomputadoras es un microprocesador que puede ser de 8 ó 16 bits. Las microcomputadoras monousuario usan un CPU de 8 bits, en cambio el CPU de 16 bits, generalmente, esta reservado para sistemas de múltiples tareas, o de múltiples usuarios.

Además del microprocesador, el CPU involucra, también, la circuitería del reloj central, de inicialización "reset" y amplificadores de corriente "buffers" los cuales van a permitir que las líneas del procesador puedan ser usadas por muchos dispositivos.

En este capítulo no se profundizará más a cerca de la unidad central de procesamiento, en virtud de que el anterior capítulo esta dedicado integramente a los microprocesadores.

4.2. MEMORIA PRINCIPAL

En electrónica, el termino memoria significa la capacidad de almacenar información digital. La memoria principal es uno de los componentes claves de toda computadora, en ella se almacenan programas y datos que se estan usando o serán usados muy proximamente por el procesador o cualquier otro dispositivo que este conectado a la memoria. Dependiendo de la complejidad del sistema, la cantidad de información que puede almacenar una memoria puede variar desde unas cuantas piezas de información hasta billones de estas piezas. Estas piezas de información son referidas como celdas de memoria, cada celda de memoria es un dispositivo o circuito electrónico que tiene dos o más estados estables.

Las memorias más comunes son las binarias que tienen solamente dos estados estables, por lo que sus celdas tienen la capacidad de almacenar digitos binarios o bits. El agrupamiento físico de varias de estas celdas o bits nos proporcionan digitos decimales (4 bits), bytes (8 bits), o palabras (n bits). Los bytes o palabras son referidos como un "quantum" en la memoria principal por lo que todos sus bits son accesados simultaneamente para operaciones de lectura o escritura.

Dos características muy importantes se notan en la memoria principal:

- La memoria principal puede ser de lectura y escritura (RWM), en tal caso la información puede ser almacenada o recuperada en cualquier instante; o bien puede ser una memoria solo de lectura (ROM), en cuyo caso se puede leer información a velocidad comparable con las memorias de lectura/escritura, sin embargo, la operación de escritura esta restringida, en algunos casos a una sola vez y fuera del sistema en un dispositivo aparte.

- La memoria principal es una memoria de acceso aleatorio "random" (RAM), donde el tiempo para tener acceso a cualquier palabra es constante, independiente de la secuencia en la cual las palabras se almacenen o hayan sido almacenadas. Esto contrasta con las memorias seriales tales como discos, cintas, registros de corrimiento, CCDs, memorias de burbuja magnética, etc., en los cuales los datos se almacenan y están disponibles en una secuencia determinada, y el tiempo para tener acceso a estos datos es variable.

La magnitud de la memoria principal se mide como w palabras de b bits cada una, siempre en ese orden.

Existen una serie de requerimientos muy deseables que toda memoria principal debe tener, los más importantes son:

- Gran capacidad de energía a la salida. Significa que las líneas de salida deben tener la capacidad de manejar la mayor cantidad de carga posible.

- Baja captación de energía a la entrada. Significa que las líneas de entrada de la memoria deben consumir la menor cantidad de corriente posible de tal manera que cualquier dispositivo las pueda manejar.

- Bajo consumo de potencia por celda de memoria. Este es un requerimiento muy importante porque mientras menor sea el consumo de energía de la celda, menores serán las condiciones para la fuente de alimentación de la memoria.

- Lectura no destructiva. Algunas memorias pierden la información cuando son leidas, tal es el caso de las memorias magnéticas de ferritas de core, en las cuales hay que restituir el dato que se lee para que puedan ser leidas nuevamente, esta condición retarda el proceso de lectura.

- Insencibilidad de las celdas no seleccionadas. Es decir, que no sufra ninguna alteración la información de aquellas celdas que no se usan.

- No volatilidad. Esto significa que la memoria principal no pierda información cuando haya una falla en la alimentación eléctrica.

- Bajo costo por bit. El costo de cada bit debe ser muy bajo para que el costo total de la memoria no se incremente mucho a medida que se incremente la magnitud de la misma.

- Volumen pequeño de cada celda. La idea es que mientras más pequeño sea el volumen que ocupa cada celda habrá mayor capacidad de almacenamiento en un volumen determinado.

- Menor tiempo de acceso posible. Este es otro de los requerimientos más importantes porque mientras menor sea el tiempo que toma leer o escribir un dato en memoria principal mayor será la velocidad de operación que se logre en el sistema.

- Inmunidad al ruido. Esto significa que la información que guarda la memoria no debe alterarse aunque el medio ambiente en donde se encuentre la misma sea muy ruidoso electricamente.

4.2.1. MEMORIAS DE LECTURA SOLAMENTE

Las memorias de lectura solamente o ROM son un tipo de memorias de semiconductores, las cuales almacenan permanentemente la información, inclusive aún fallando la alimentación de la energía eléctrica. En alguno dispositivos los datos deben ser construidos durante el proceso de manufactura, y en otros los datos pueden ser grabados electricamente. El proceso de alimentar los datos al ROM se lo conoce como programación.

Las memorias ROM juegan un papel importante en las computadoras en general y en las microcomputadoras en particular ya que se usan en aplicaciones donde la información no va a cambiar frecuentemente, por ejemplo, para almacenar programas que quedarán residentes definitivamente en la memoria principal, tales como monitores (programa para interactuar con los recursos del sistema), "bootstraps" (programas que sirven para arrancar la ejecución de un programa mucho más grande), sistemas operativos, funciones especiales, etc.

Existen varios tipos de memorias ROM, la figura 4-1 muestra

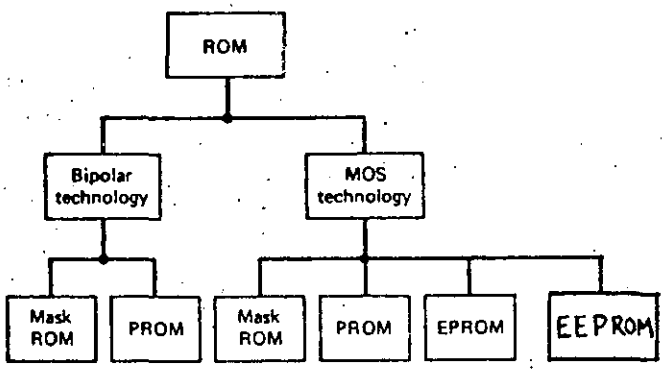


Figura 4-1: Tipos de memorias ROM

las dos tecnologías y los diferentes tipos de ROMs que existen para cada caso.

En terminos generales, los ROMs bipolares se caracterizan por un tiempo de acceso muy pequeño, entre 30 y 90 nseg y una baja capacidad de almacenamiento por chip, desde 256 bits hasta 4K bits. En cambio, los ROM del tipo MOS tienen características contrarias totalmente, un tiempo de acceso grande, entre 200 y 1500 nseg y una gran capacidad de almacenamiento por chip desde 4K bits nasta 128 Kbits.

4.2.1.1. ROM DE MASCARA

El ROM de mascara es un dispositivo en el cual el patrón de datos que se deseé almacenar se programa como una parte del proceso de manufactura. Una vez que el dispositivo ha sido programado el patron de datos no puede ser cambiado nunca más, por lo tanto se requiere una seguridad absoluta con los datos para solicitar la programación. Aparte de este problema, los fabricantes, que son los que efectuan la programación, no programan ROMs en cantidades pequeñas por el alto costo del proceso tecnologíco, sino unicamente cuando se trata de grandes cantidades de chips. Sin embargo, en cantidades muy grandes el costo de cada chip es muy barato.

La programación del patron de datos se efectua en la última etapa del proceso tecnológico, o sea en la etapa de metalización,

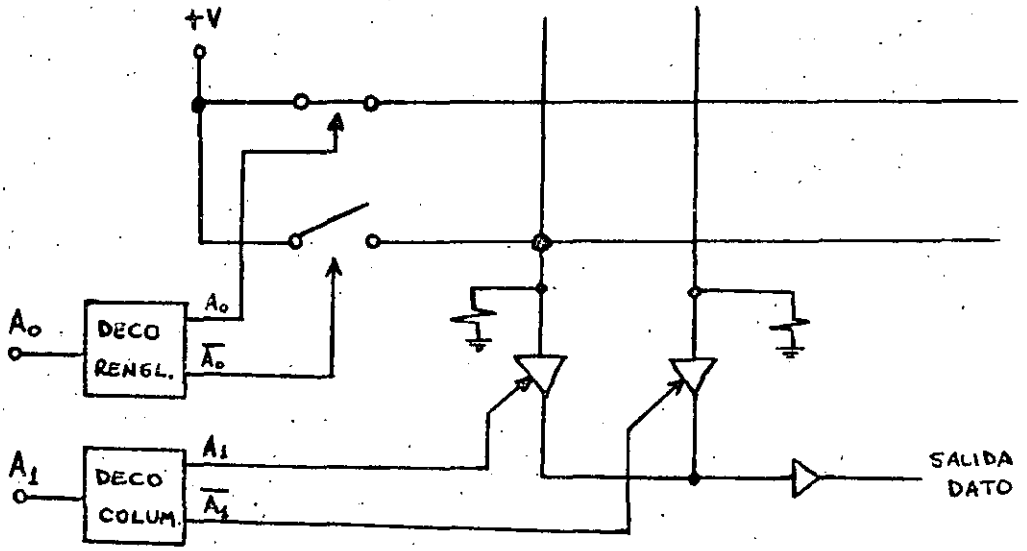


Figura 4-2: Metalizado en las celdas del ROM de mascara.

en la cual se inyecta una capa de aluminio a toda la oblea y por medio de una mascara inyectando una substancia quimica se elimina el aluminio de las partes deseadas. En el ROM de mascara, la programación consiste en quitar o dejar el aluminio en las celdas de memoria (intersecciones de renglones y columnas). Según la figura 4-2 cuando se deja aluminio en la intersección de un renglon y una columna, ambos quedan en corto circuito, produciendo un uno (voltaje alto) a la salida cuando se seleccione dicna intersecci"on. Cuando se quita el aluminio de las intersecciones la selección de alguna de estas proporciona un cero a la salida.

4.2.1.2. ROM PROGRAMABLE "PROM"

El ROM programable electricamente, más conocido como PROM, difiere del ROM de mascara en que el patrón de datos se programa electricamente por el usuario en lugar de que sea un paso del proceso de fabricación del circuito integrado. La programación usualmente se lleva a efecto con un equipo especial conocido como programador de PROMs, y se hace fuera de la tarjeta o del sistema que usará el PROM. Una vez que el patrón de datos ha sido programado es, en la gran mayoría de los casos, imposible cambiarlo. El PROM es usado principalmente en aplicaciones donde la cantidad de estos es pequeña y por lo tanto no amerita usar ROMs de mascara.

La celda básica del PROM, ver figura 4-3, está constituida

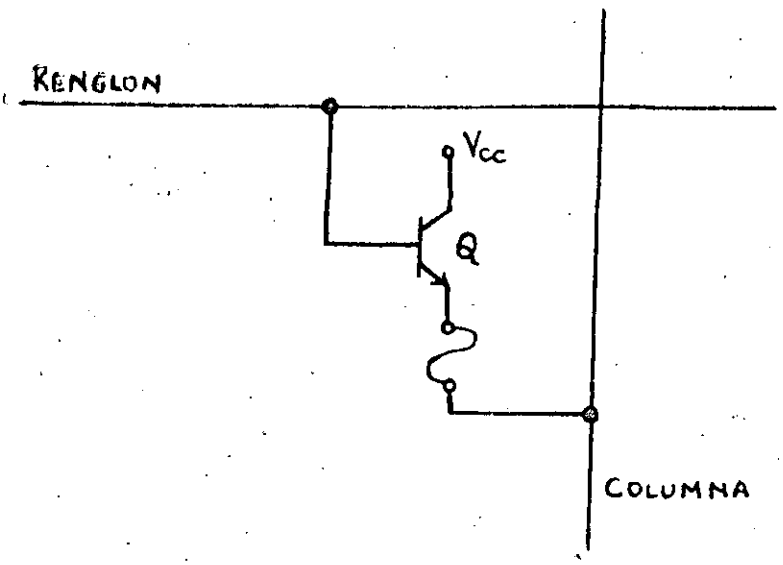


Figura 4-3: Celda básica de un PROM

por un transistor que se conecta entre el renglon y la columna. Entre el emisor y la columna existe un fusible que puede ser de Nicromo (aleación de Niquel y Cromo) o un fusible de silicio policristalino. Haciendo circular una gran cantidad de corriente se rompe o destruye el fusible y queda para siempre abierta esa celda, es decir, la conección en esa intersección entre el renglon y columna respectiva. Si la celda no ha sido programada (fusible sano) se leerá un uno lógico, en cambio, si la celda se ha programado (fusible roto) a partir de ese momento siempre se leerá un cero lógico.

4.2.1.3. ROM PROGRAMABLE Y BORRABLE "EPROM"

Este es un ROM que puede ser programado electricamente por el usuario, sin embargo, su patrón de datos puede ser borrado exponiendo el dispositivo a luz ultravioleta durante un cierto tiempo. Este dispositivo es conocido como EPROM y la gran diferencia con el PROM radica en que el EPROM puede ser reprogramado nuevamente, luego borrado y reprogramado otra vez y así sucesivamente pueden realizarse varios ciclos de este estilo. En los EPROMs modernos se pueden tener del orden de 20 ciclos de borrado y reprogramación.

Los EPROMs se usan sobre todo en las áreas de investigación y desarrollo, ya que en estos casos los programas se prueban una y otra vez nasta lograr la versión definitiva del programa. También, son usados en situaciones en las cuales la información almacenada eventualmente puede cambiar.

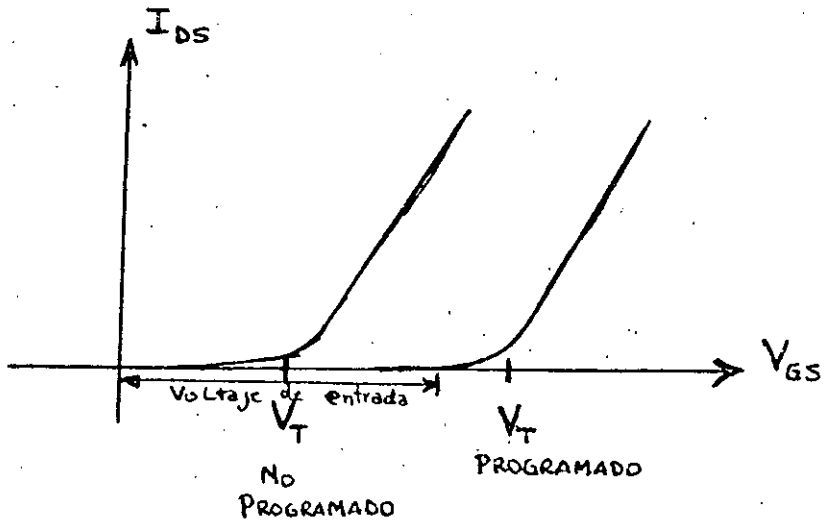


Figura 4-4: Efecto en el voltaje de ruptura al programar el EPROM

En 1971 Intel lanzó al mercado el primer EPROM, fue el 1702 el cual tenia como celda de memoria un transistor llamado FAMOS (floating gate avalancne inyection MOS). El efecto que se produce en el transistor FAMOS al programarse la celda es un corrimiento del voltaje de umbral hacia un valor más alto, tal como se indica en la figura 4-4, de esta manera al seleccionar una celda programada no se llega a prender el transistor, en cambio al seleccionar una celda no programada este se prende inmediatamente.

4.2.1.4. PROM BORRABLE ELECTRICAMENTE "EEPROM"

Se denomina EEPROM al PROM programable eléctricamente y también borrable eléctricamente. La gran diferencia con el EPROM es la facilidad de programarlos y borrarlos por el mismo sistema sin necesidad de quitarlos de la tarjeta y programarlos aparte con un equipo de programación especial. Estos son los últimos dispositivos que han salido al mercado y tienen la ventaja que se pueden realizar cientos de miles de ciclos de borrado y reprogramación. El área de aplicación de estos dispositivos es mucho más amplia que los anteriores tipos de ROMs, ya que pueden servir para las mismas aplicaciones y además para los casos en que los datos van a permanecer constantes un cierto tiempo y luego cambiar, por ejemplo en dispositivos donde se programan datos o parámetros previamente, tal como calculadoras con memoria no volatil, terminales, equipo de medición, aparatos de control, memoria para salvar el estatus de ejecución de las máquinas antes de que se corte la energía eléctrica, etc.

Estos dispositivos tienen las características de las memorias de ferritas de core, porque retienen la información cuando se corta la alimentación de la energía eléctrica y además

tiene la capacidad de cambiar los datos en forma rápida. También, tiene características de RAM y ROM de semiconductores por su costo menor al de las ferritas de core, bajo consumo de potencia y alta densidad.

El EEPROM tiene tres modos principales de operación:

1. Modo de lectura. Similar a una RAM, con tiempos de acceso en el orden de 500 nanosegundos.

2. Modo de escritura básica. Para lo cual se requiere una fuente de 17 volts en una sola pata y un pulso de 100 microseg., y una corriente de escritura de 10 miliamp.

3. Modo de borrado. Borra todas las celdas del chip con un solo pulso de 17 volts en V(DD) y en la misma pata que para la escritura. Este pulso es de 100 microseg. y requiere una corriente de borrado de 10 miliamp.

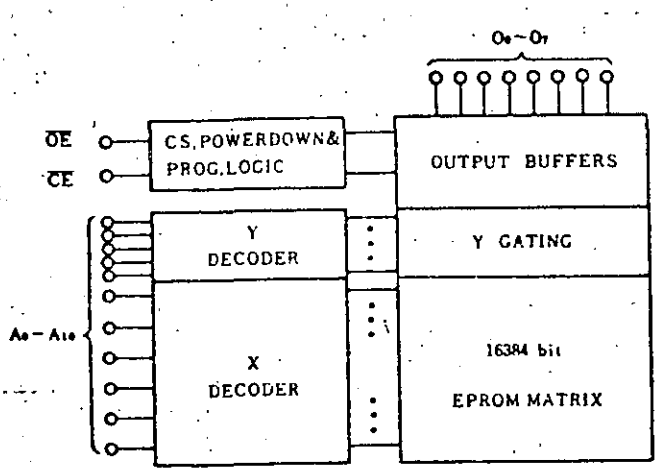
El tiempo que almacenan la información estos chips es del orden de 10 años, aún permaneciendo a 100 grados centigrados.

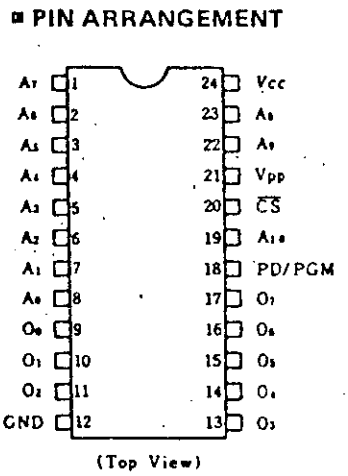
4.2.1.5. EJEMPLO: EPROM 2716

El EPROM 2716 es un chip muy usado que tiene una capacidad de almacenamiento de 16 K bits y está organizado como 2K bytes. Existen dos tipos de 2716, aquellos que tienen 3 fuentes de alimentación +5, +12 y -5 volts, los cuales conservan la misma distribución de fuentes de alimentación que su predecesor el 2708; y los de una sola fuente de alimentación de +5 volts. Los EPROMs sucesores al 2716 tienen una sola fuente de alimentación como son los 2732 y 2764. Se ha tratado de mantener la mayor compatibilidad posible en la distribución de patas entre todos los chips de la familia 2700, con el fin de evitar al máximo las modificaciones de nuevas versiones de sistemas.

El 2716 tiene 11 líneas de dirección, 8 líneas de datos y tres señales de control: CE "chip enable" habilitación del chip, OE "output enable" habilitación de la salida y Vpp pata de programación. Este chip tiene las siguientes formas de operación:

	CE	OE	Vpp	Vcc	SALIDAS
Lectura	0	0	+5	+5	Datos de salida
No seleccion	X	1	+5	+5	Alta impedancia
Reducción					





DESCRIPCION EXTERNA

Figura 4-5: Organización interna del EPROM 2716

de potencia	Ø	1	+5	+5	Alta impedancia
Programacion	Pulso	1	+25	+5	Datos de entrada
Verificacion	Ø	Ø	+25	+5	Datos de salida
No programar	Ø	1	+25	+5	Alta impedancia

Tabla 4-1: Formas de operación del EPROM 2716

Para leer los datos del EPROM se debe mantener en Ø CE y cambiar las direcciones manteniendo en Ø OE cuando las direcciones están estables.

A diferencia de sus ancestros 2708 y 2704, en los cuales nabia que programar todas las palabras de una sola vez, en el 2716 se puede programar palabra por palabra o palabras escogidas aleatoriamente, además, se puede verificar la programación de las palabras inmediatamente después de haber programado la misma sin necesidad de haber cambiado a modo lectura (cambiar Vpp a +5 volts).

El pulso en CE para la programación debe ser de 50 miliseg en cambio el pulso de OE en la verificación debe ser igual al tiempo de acceso (entre 300 y 500 nanoseg). La figura 4-7 muestra las formas de onda de las señales que se deben generar para programar y erificar un dato. Todas las señales que se deben alimentar, excepto Vpp tienen niveles TTL. Programar todos los datos del EPROM secuencialmente (50 miliseg por dato), toma un tiempo de 100 segundos aproximadamente.

READ MODE ($\overline{CE}$ = V_{IL})

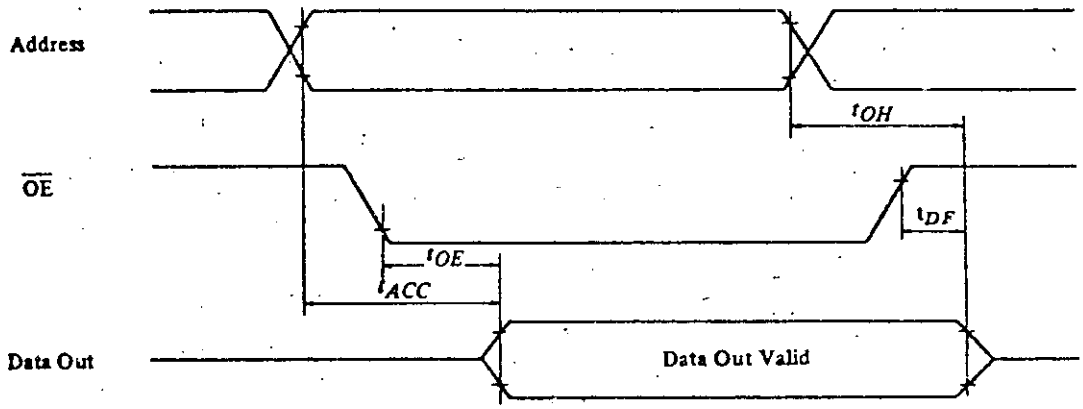


Figura 4-6: Modo de lectura del EPROM 2716

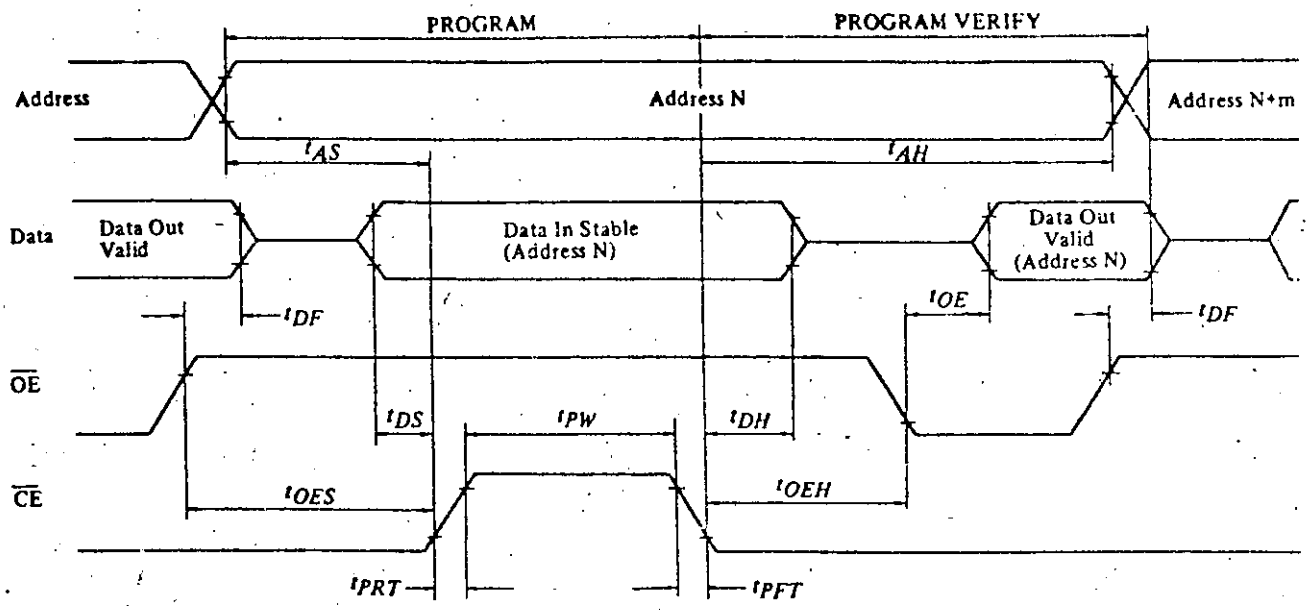


Figura 4-7: Modo de programación y verificación del EPROM 2716

Borrar con luz ultravioleta el chip significa dejar en uno lógico todas las celdas de memoria. Por lo tanto el proceso de programación lo que hace es dejar en 0 lógico las celdas que se van a programar. El modo de no programación permite programar varios chips con diferentes datos a través de un solo bus de datos; este modo impide que la programación afecte a todos los chips sino solo a aquellos que se deseé.

4.2.2. MEMORIA DE LECTURA Y ESCRITURA

En la industria de la electrónica digital se denomina memoria RAM a los dispositivos que tienen la capacidad de almacenar y recuperar información en forma aleaatoria. La memoria RAM a diferencia de la memoria ROM no tiene la capacidad de almacenar la información en forma permanente, mas bien, se puede alterar la información en cualquier momento. El tiempo que se requiere para leer (recuperar) es semejante al tiempo que se requiere para escribir (almacenar) un dato; esto difiere con la memoria ROM, donde en algunos casos se requiere escribir fuera de línea y en el mejor caso (EEPROMS) el tiempo de escritura es 200 veces mayor al tiempo de lectura.

Existen dos tipos de memoria RAM: magnética y de semiconductores. La memoria magnética fue la primera en utilizarse, las celdas de esta memoria están compuestas por unos nucleos o toroides ferromagnéticos muy pequeños por los cuales atraviezan 3 o 4 conductores eléctricos que sirven para seleccionar la celda, leer, escribir o inhibir datos. La gran ventaja de esta tecnología es que resulta una memoria no volatil, precisamente porque se usan materiales magnéticos para la construcción de la memoria. Sin embargo, presenta varias desventajas respecto a las memorias de semiconductores, entre ellas el alto costo, baja densidad, alto consumo de potencia, lectura destructiva, tiempo de acceso bastante grande, etc. En la actualidad estas memorias se usan sobre todo en los equipos y computadoras antiguas. En microcomputadoras no es usual encontrar memorias ferromagnéticas, las memorias que dominan en la actualidad son las memorias de semiconductores.

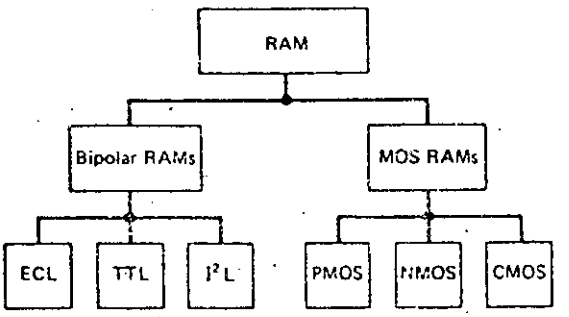


Figura 4-8: Tecnologías en memorias de semiconductores

Existen dos tecnologías básicas para la fabricación de memorias RAM de semiconductores, la bipolar y la MOS. La figura 4-8 muestra las diferentes familias lógicas que se usan para cada caso.

Las memorias RAM de semiconductores tienen 3 buses básicos, direcciones, datos y señales de control.

La RAM se diferencía de la memoria ROM en que tiene líneas para los datos de salida y una señal de control que indica lectura o escritura (R/W). La figura 4-9 muestra la memoria RAM como una caja negra y divide las señales que comunican hacia el exterior a la RAM en tres grandes grupos: direcciones, control y

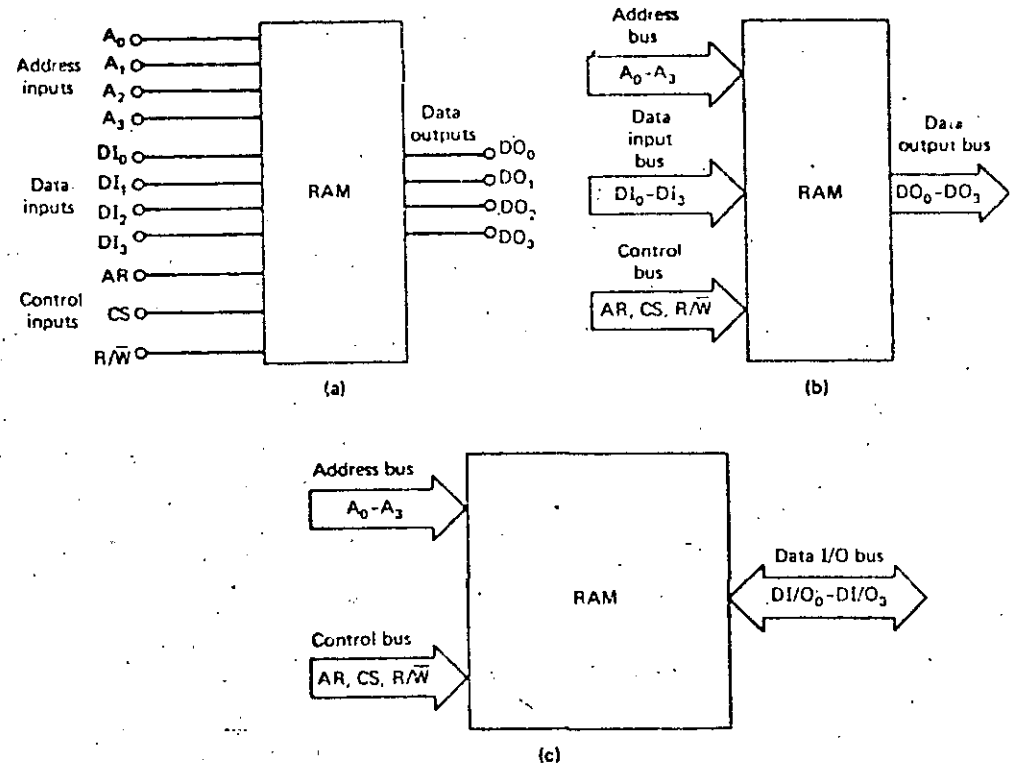


Figura 4-9: Memoria RAM vista desde afuera

datos, los cuales pueden ser unidireccionales o bidireccionales. Las salidas de los datos son del tipo de colector abierto o de tres estados para formar buses de datos en paralelo y tener la opción de expander la memoria a voluntad. Las señales de control normalmente son:

- R/W, la cual indica si se va efectuar la lectura o escritura de un dato.
- CS, esta señal permite colocar chips en paralelo y seleccionar solo aquellos que se requieran.
- AR "address ready", sirve para indicar que las direcciones están estables.

4.2.2.1. ESTRUCTURA INTERNA DE LA MEMORIA RAM

La memoria RAM se compone de varios bloques internos que son: receptor de señales de control, receptor de datos de entrada, receptor de direcciones, decodificador de renglones, decodificador de columnas, arreglo de celdas de memoria, amplificador de sensado y amplificador para las señales de salida.

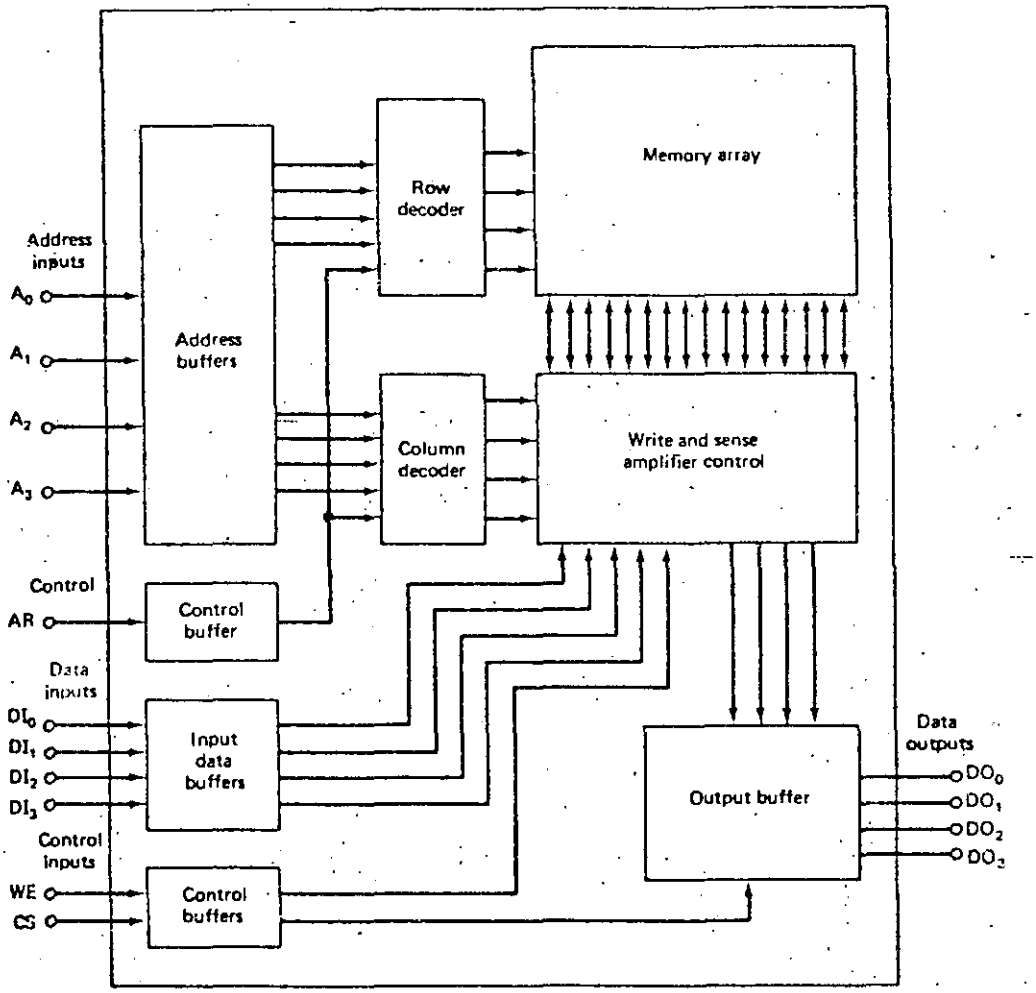


Figura 4-10: Arquitectura típica de una RAM

La figura 4-10 muestra los bloques internos de una memoria RAM típica y las interconecciones entre estos.

El arreglo de las celdas de memoria está constituido por una conjunto de elementos idénticos organizados normalmente en forma de una matriz cuadrada que tiene igual número de renglones que columnas. Existen dos tipos básicos de celdas de memoria que clasifican en dos grandes categorías a las memorias RAM:

1. Las celdas que mientras exista energía eléctrica guardan indefinidamente la información, las memorias que tienen este tipo de celdas son conocidas como RAM estáticas.

2. Y las celdas que aunque haya energía electrica pierden la información a menos que se les recuerde cada cierto tiempo cual es la información que tienen almacenadas, este tipo de celdas tienen las memorias conocidas como RAM dinámicas.

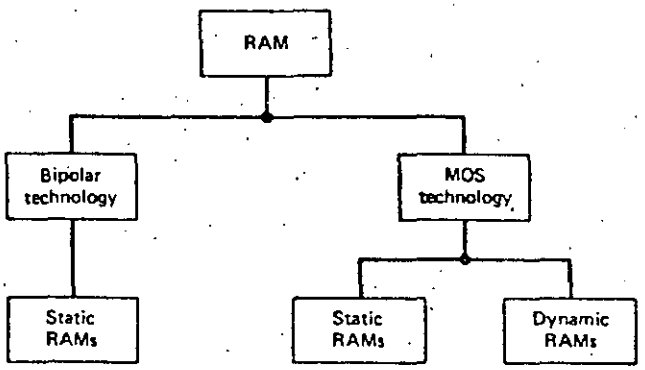


Figura 4-11: Categorías de memorias RAM

4.2.2.2. MEMORIAS MOS RAM ESTATICAS

En las memorias RAM estáticas las celdas pueden estar constituidas por transistores bipolares o MOS. Las celdas en este caso son flip-flops construidos a base de transistores, por lo tanto el dato que almacenan estas celdas corresponden al estado lógico del flip-flop, de esta manera el dato puede permanecer indefinidamente a menos que se corte la alimentación de la energía eléctrica.

Las celdas están organizadas internamente por renglones y columnas, el decodificador de renglones habilita todas las celdas que pertenecen a un renglón determinado, ver figura 4-12, asimismo el decodificador de columnas selecciona la o las columnas respectivas. Si la organización del chip es de 16K bits,

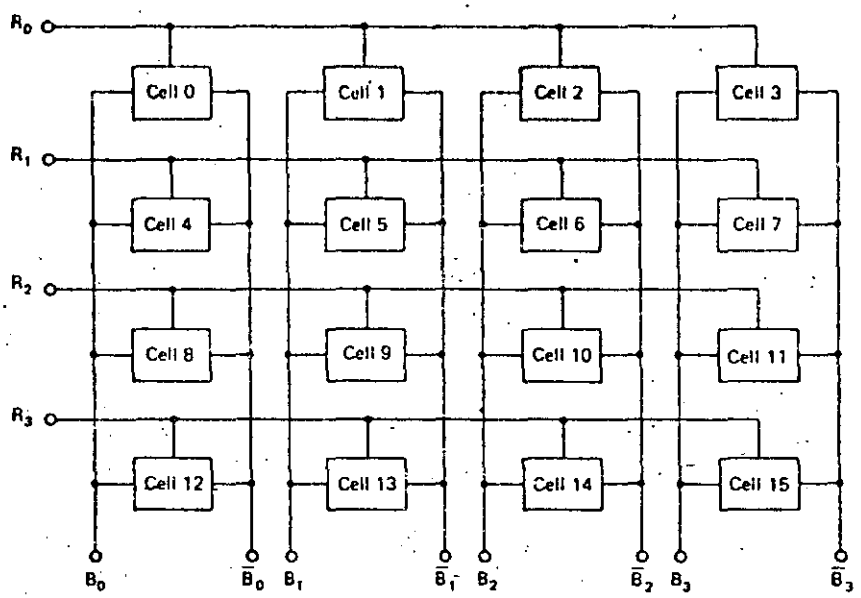


Figura 4-12: Organización interna de las celdas

el decodificador de columnas seleccionará una sola, si es de 2K bytes seleccionrá 8 columnas, etc.

Una desventaja de estas celdas es el gran tamaño que ocupan lo cual reduce la densidad considerablemente, estás celdas en el caso MOS están constituidas por 6 transistores.

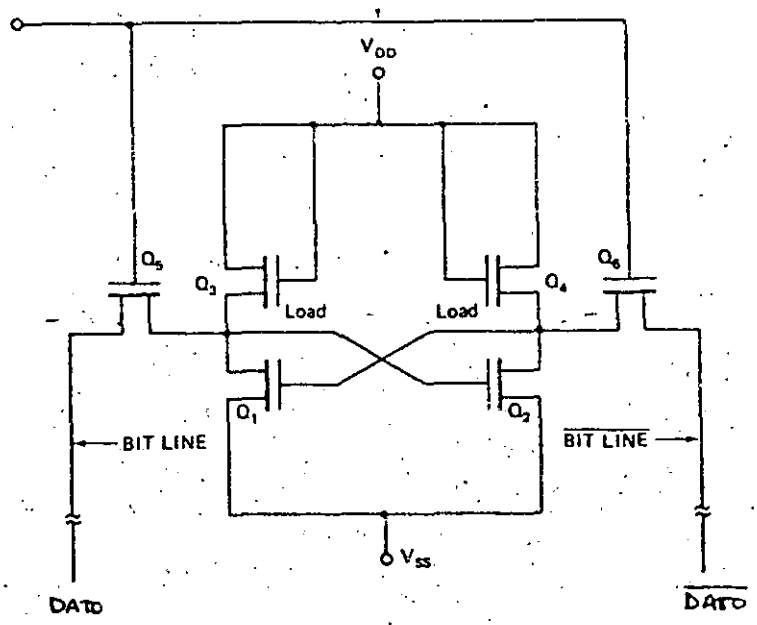


Figura 4-13: Celda de memoria de 6 transistores (estática)

Las líneas de DATO y DATO negado son buses a los cuales se conectan todas las celdas que pertenecen a la columna, pero son manejadas solamente por la celda que tiene los transistores TS1 TS2 prendidos, es decir, la celda que pertenece al renglon seleccionado en ese momento.

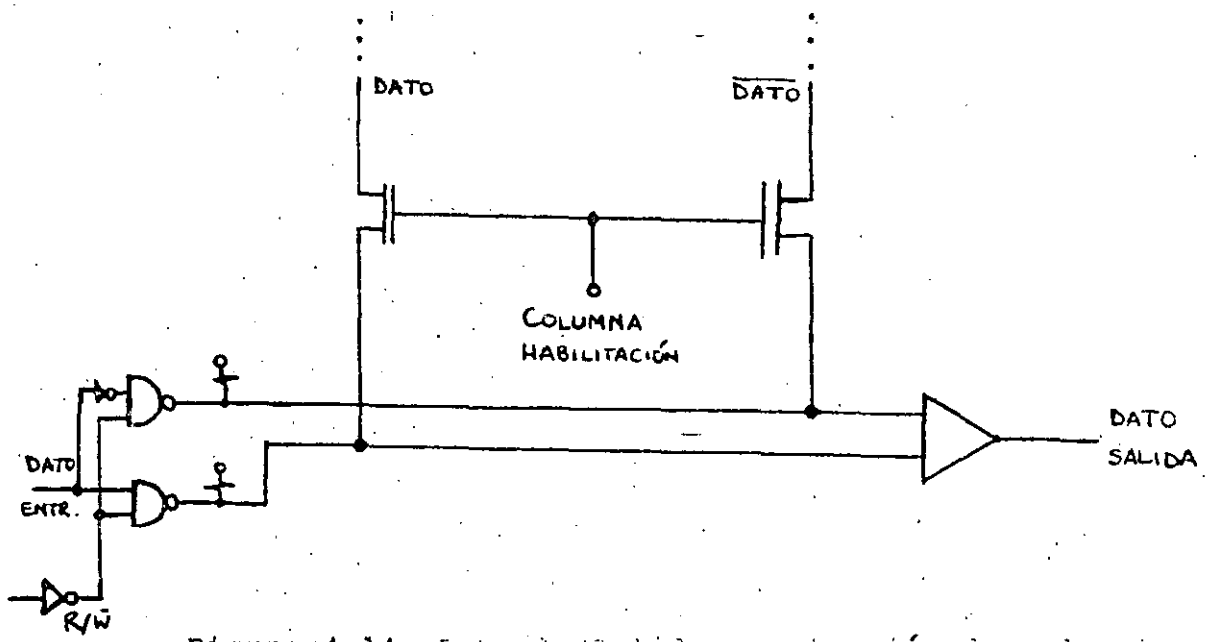


Figura 4-14: Entrada/Salida y selección de columna

La selección de la columna se hace habilitando dos transistores que se encuentran conectados a los buses de DATO y DATO negado. Estos dos transistores pertenecen al bloque de amplificación de sensado y permiten la escritura de nuevos datos y el sensado de la información que se encuentre en DATO y DATO negado, asimismo esa informmación llega a un amplificador diferencial del cual se obtiene los datos a la salida.

4.2.2.3. MEMORIA MOS RAM DINAMICA

Las celdas de memorias RAM dinámicas están constituidas por transistores del tipo MOSFET unicamente, precisamente porque se aprovecha la impedancia de entrada casi infinita del MOSFET, la cual provee un modo de almacenamiento temporal de datos y se puede aprovechar para simplificar la circuitería de las celdas RAM. La juntura PN de la región de la compuerta "gate" y el substrato forman una capacitancia bastante grande que se puede aprovechar para almacenar por un tiempo finito datos en la compuerta del MOS. El tiempo que la información puede permanecer almacenada mientras se descarga el capacitor es de varios milisegundos.

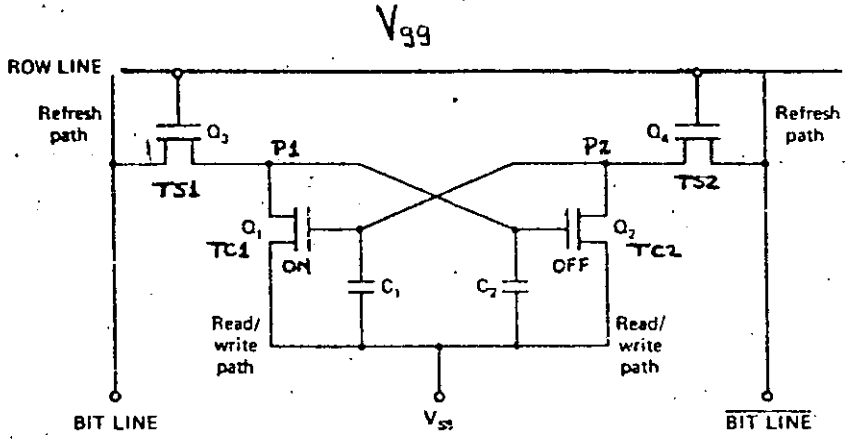


Figura 4-15: Celda de memoria de 4 transistores

La figura 4-15 muestra una celda dinámica de cuatro transistores, dos de ellos sirven como switches TS1 y TS2, en cambio los otros dos son los que almacenan el dato de la celda, por lo que en este caso la información almacenada está representada por la carga de los capacitores de las compuertas de los transistores TC1 y TC2, a diferencia de las memorias estáticas donde los datos son los estados lógicos de los flip-flops de las celdas.

Si Vgg = Vdd, la celda de cuatro transistores de la fig. 4-15 es igual a la estructura básica del flip-flop de la celda de 6 transistores, fig 4-13. En cambio si Vgg = Ø, ya no hay sustento para la alimentación eléctrica de la celda y el capacitor que estuviese cargado C1 o C2 se empieza a descargar exponencialmente. En el ejemplo de la fig 4-16 se supone que TC1 está prendido y TC2 está apagado, por lo que el capacitor C1 está cargado no así el capacitor C2. La señal en la compuerta de TC1 decrece mientras se descarga el capacitor C1, sin embargo, el nivel de esta señal no debe ser nunca menor que V(t) del transistor TC1 para que este no llege a apagarse. Cuando el nivel de esta señal está muy cerca de V(t) es preciso alimentar nuevamente Vgg a Vdd, con lo cual la señal de entrada de TC1 vuelve nuevamente a adquirir su valor inicial, es decir, el capacitor C1 vuelve a cargarse nuevamente. El tiempo que se requiere para recargar nuevamente el capacitor es muy corto e igual al tiempo de ciclo de la memoria. Esta señal de Vgg puede

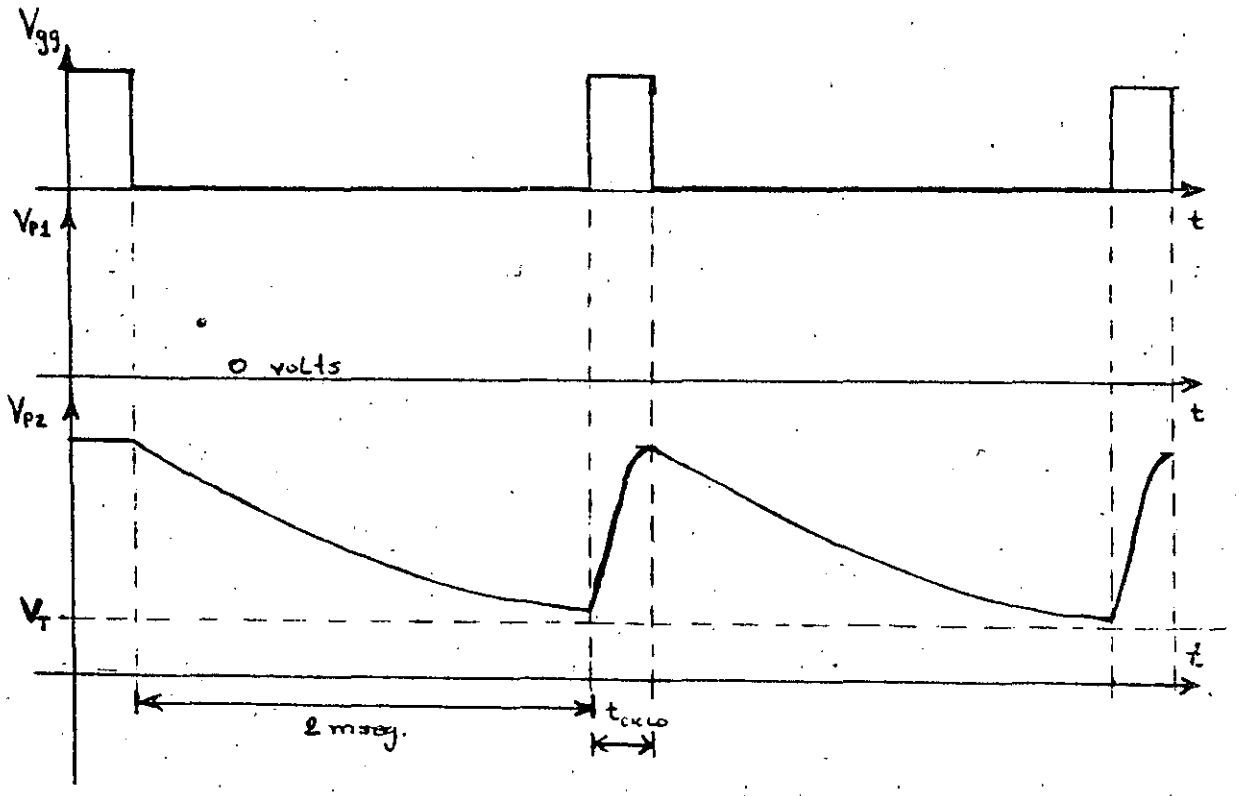


Figura 4-16: Carga y descarga de los capacitores C1 y C2

ser la habilitación del renglón, por lo que entonces cada 2 miliseg (tiempo de descarga) se debe habilitar el renglon para recordar a las celdas que están conectadas a este cual es el dato que tienen almacenado, en otras palabras, cargar nuevamente los capacitores que estabán descargandose.

Este recordatorio que hay que hacer cada 2 miliseg (aproximadamente) se conoce como refrescar la información de las celdas. El refresco hay que efectuarlo, desde luego, en todos los renglones máximo 2 miliseg entre recordatorio y recordatorio. El refresco en las memorias dinámicas las hace más difíciles de usar que las estáticas, asimismo existe un tiempo muerto (durante el refrescamiento) que no se pueden usar. En cambio la gran ventaja de las memorias dinámicas sobre las estáticas es que son de muy alta densidad aproximadamente 4 veces más que las estáticas porque la celda tiene menos transistores, en este caso 4 transistores, sin embargo, existen celdas de 3 transistores y las más comunes son las celdas de un solo transistor y un capacitor, exprofesamente construido para almacenar la carga que representa la información de la celda.

La figura 4-17 muestra una celda de memoria de un solo transistor, esta celda es muy usada en chips de 16K bits y 64K

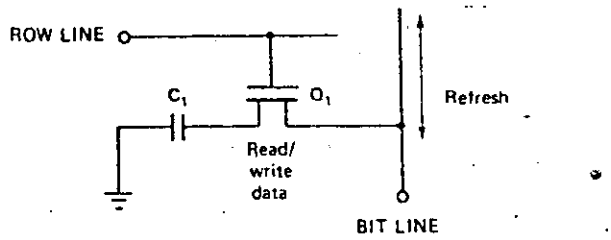


Figura 4-17: Celda dinámica de un solo transistor

bits, esta última es la memoria más densa de la actualidad. En cambio la memoria estática más densa actualmente es la de 2K bytes, o sea 16K bits.

4.2.2.4. EJEMPLOS DE MEMORIAS RAM

La memoria 6116 es una memoria RAM estática de 16K bits con una organización de 2K bytes.

La figura 4-18 muestra una memoria RAM estática del tipo CMOS, la cual tiene 11 líneas de dirección, 8 líneas de datos y 3 señales de control, CS "chip select", OE "output enable" y WE "write enable". La distribución de patas de este chip es semejante al EPROM 2716, de esta manera resultan, prácticamente, compatibles; la única señal que se requiere cambiar es WE, porque en el EPROM esta línea corresponde a Vpp.

Las figuras 4-19 y 4-20 muestran los ciclos de lectura y escritura de la memoria 6116, se observa que para leer CE y OE deben estar en bajo mientras que WE en alto, para escribir OE debe estar en alto, mientas que CS y WE en bajo.

Las memorias RAM dinámicas 4864 o 4164 tienen una capacidad y organización de 64K bits. Tienen una sola fuente de alimentación de 5 volts y todas sus señales son compatibles directamente con TTL. La organización interna de las celdas de memoria es de 8 arreglos de memoria de 128 x 64 bits cada uno, para poder mantener solo 128 renglones de refrescamiento, aunque desde el punto de vista de acceso la memoria se ve como una organización cuadrada de 256 x 256 bits.

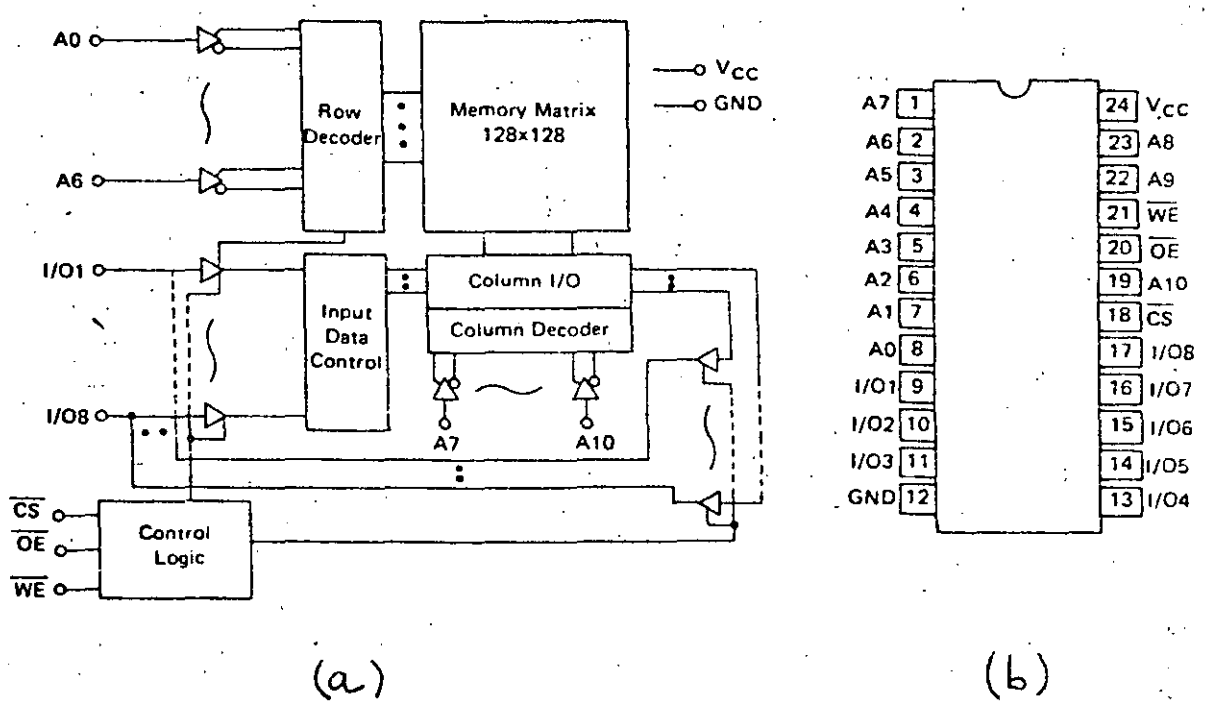


Figura 4-18: RAM CMOS estática de 2K bytes
(a) Diagrama de bloques
(b) Distribución de patas del chip

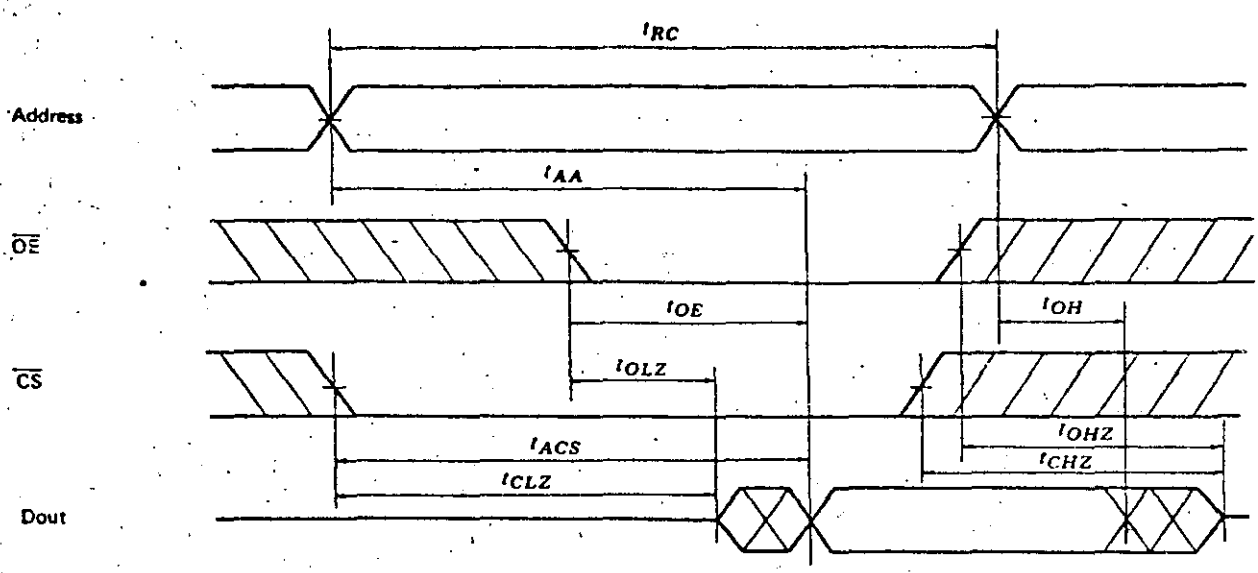


Figura 4-19: Ciclo de lectura de la memoria 6116

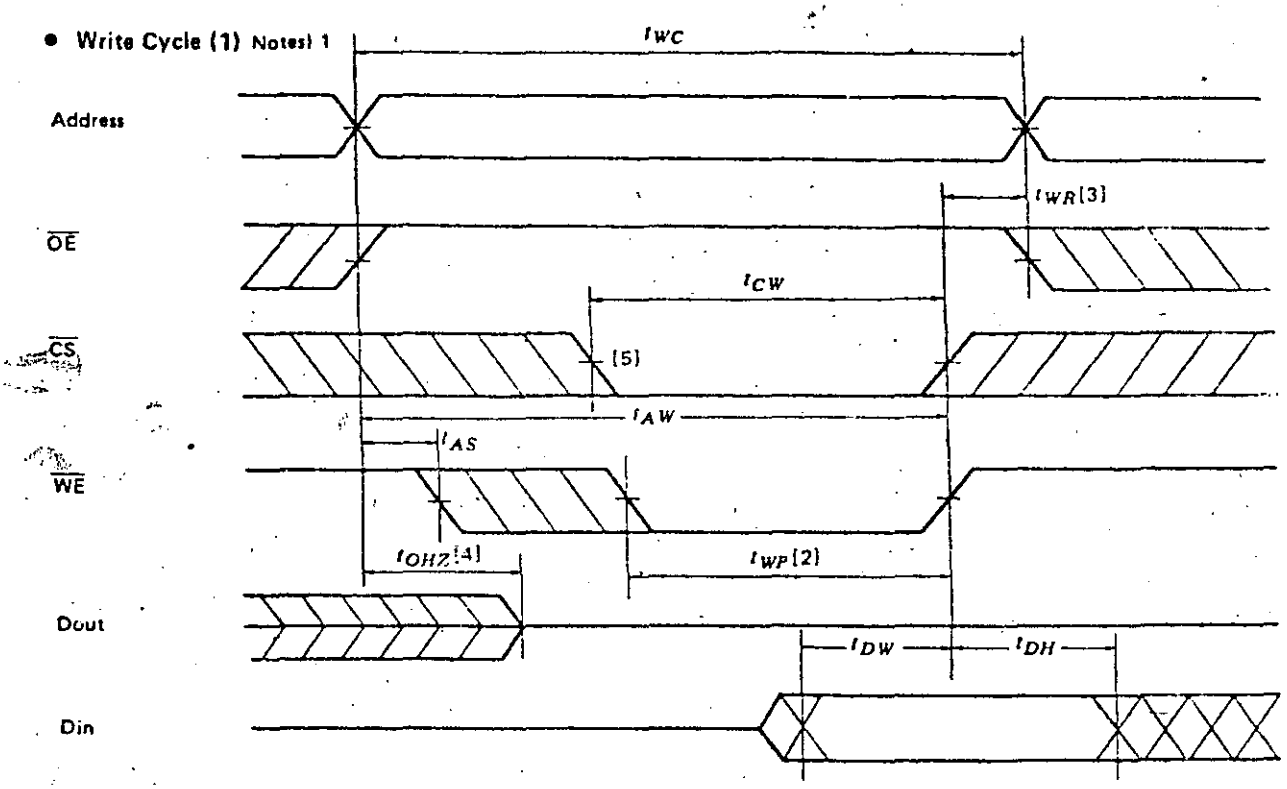


Figura 4-20: Ciclo de escritura de la memoria 6116

• **READ CYCLE**

Item	Symbol	HM6116P-2		HM6116P-3		HM6116P-4		Unit
		min.	max.	min.	max.	min.	max.	
Read Cycle Time	t_{RC}	120	–	150	–	200	–	ns
Address Access Time	t_{AA}	–	120	–	150	–	200	ns
Chip Select Access Time	t_{ACS}	–	120	–	150	–	200	ns
Chip Selection to Output in Low Z	t_{CLZ}	10	–	15	–	15	–	ns
Output Enable to Output Valid	t_{OE}	–	80	–	100	–	120	ns
Output Enable to Output in Low Z	t_{OLZ}	10	–	15	–	15	–	ns
Chip deselection to Output in High Z	t_{CHZ}	0	40	0	50	0	60	ns
Chip Disable to Output in High Z	t_{OHZ}	0	40	0	50	0	60	ns
Output Hold from Address Change	t_{OH}	10	–	15	–	15	–	ns

• **WRITE CYCLE**

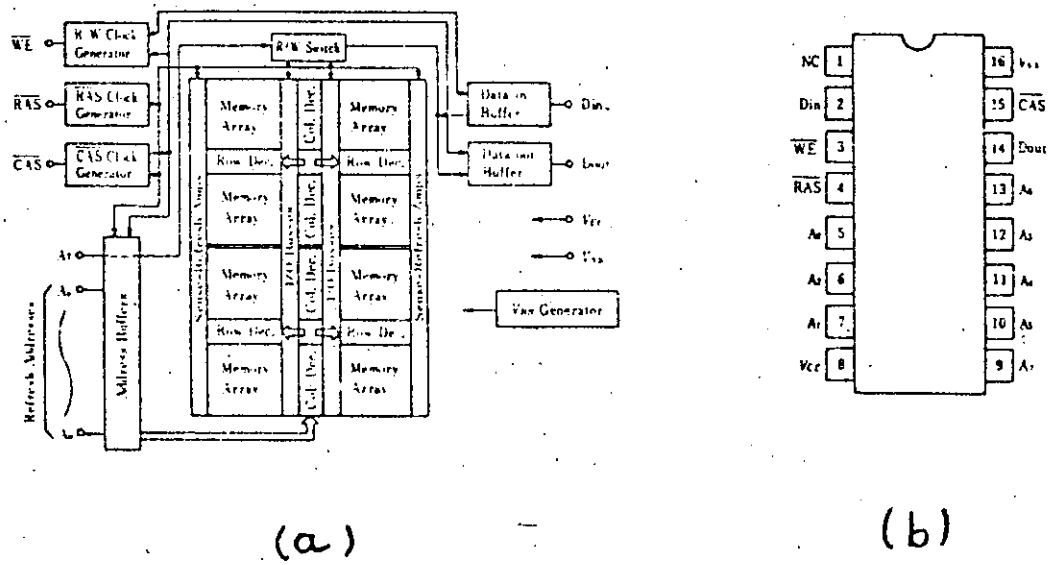
Item	Symbol	HM6116P-2		HM6116P-3		HM6116P-4		Unit
		min.	typ.	min.	max.	min.	max.	
Write Cycle Time	t_{WC}	120	–	150	–	200	–	ns
Chip Selection to End of Write	t_{CW}	70	–	90	–	120	–	ns
Address Valid to End of Write	t_{AW}	105	–	120	–	140	–	ns
Address Set Up Time	t_{AS}	20	–	20	–	20	–	ns
Write Pulse Width	t_{WP}	70	–	90	–	120	–	ns
Write Recovery Time	t_{WR}	5	–	10	–	10	–	ns
Output Disable to Output in High Z	t_{OHZ}	0	40	0	50	0	60	ns
Write to Output in High Z	t_{WHZ}	0	50	0	60	0	60	ns
Data to Write Time Overlap	t_{DW}	35	–	40	–	60	–	ns
Data Hold from Write Time	t_{DH}	5	–	10	–	10	–	ns
Output Active from End of Write	t_{OW}	5	–	10	–	10	–	ns

Tabla 4-2: Límites de tiempos en los ciclos de lectura y escritura

El chip de esta memoria es de 16 patas y tiene 8 líneas de dirección, una línea de datos de entrada otra para datos de salida, y tres señales de control WE "write enable", RAS "row address strobe" y CAS "column address strobe". Las direcciones se deben entregar en dos etapas primero la dirección del renglon (8 bits) y luego por las mismas líneas, la dirección de la columna (8 bits). Las señales de RAS y CAS sirven para diferenciar en las líneas de dirección cuando se entrega la dirección del renglón y la columna.

Las figuras 4-22 y 4-23 muestran la secuencia que se debe realizar para cumplir con los ciclos de lectura y escritura. Es muy importante, en este caso, los instantes en los cuales debe cambiar cada señal, ya que en muchos casos existen tiempos mínimos y máximos que se deben cumplir.

La tabla 4-3 muestra los tiempos mínimos y máximos recomendados para la operación de las memorias HM4864-2 y HM4864-3, algunos lapsos de tiempo son muy críticos, como por ejemplo el retardo de RAS a CAS (tRCD), para cumplir con el tiempo de acceso de RAS (tRAC).



(a) (b)

Figura 4-21: Memoria RAM dinámica 4864
(a) Diagrama de bloques interno
(b) Distribución de patas en el chip

- **READ CYCLE**

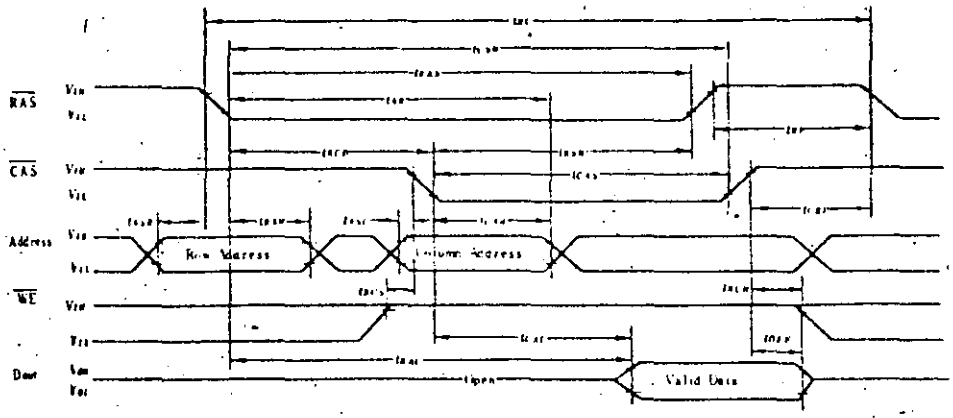


Figura 4-22: Ciclo de lectura de la memoria 4864

- **WRITE CYCLE**

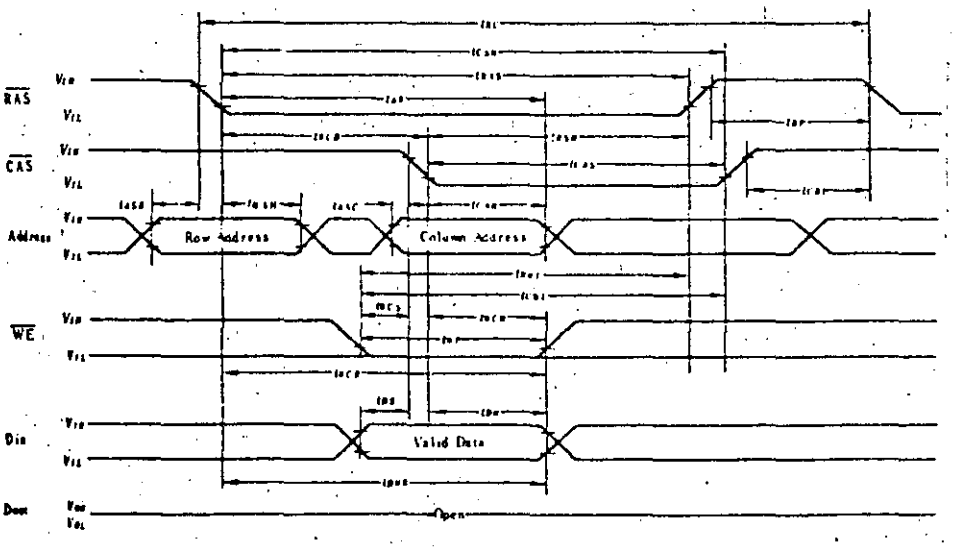


Figura 4-23: Ciclo de escritura de la memoria 4864

Parameter	Symbol	HM4864-2		HM4864-3		Unit
		min.	max.	min.	max.	
Random Read or Write Cycle Time	t_{RC}	270	–	335	–	ns
Read-Write Cycle Time	t_{RWC}	270	–	335	–	ns
Page Mode Cycle Time	t_{PC}	170	–	225	–	ns
Access Time from $\overline{RAS}$	t_{RAC}	–	150	–	200	ns
Access Time from $\overline{CAS}$	t_{CAC}	–	100	–	135	ns
Output Buffer Turn-off Delay	t_{OFF}	0	40	0	50	ns
Transition Time (Rise and Fall)	t_T	3	35	3	50	ns
$\overline{RAS}$ Precharge Time	t_{RP}	100	–	120	–	ns
$\overline{RAS}$ Pulse Width	t_{RAS}	150	10000	200	10000	ns
$\overline{RAS}$ Hold Time	t_{RSH}	100	–	135	–	ns
$\overline{CAS}$ Pulse Width	t_{CAS}	100	–	135	–	ns
$\overline{CAS}$ Hold Time	t_{CSH}	150	–	200	–	ns
$\overline{RAS}$ to $\overline{CAS}$ Delay Time	t_{RCD}	20	50	25	65	ns
$\overline{CAS}$ to $\overline{RAS}$ Precharge Time	t_{CRP}	−20	–	−20	–	ns
Row Address Set-up Time	t_{ASR}	0	–	0	–	ns
Row Address Hold Time	t_{RAH}	20	–	25	–	ns
Column Address Set-up Time	t_{ASC}	−10	–	−10	–	ns
Column Address Hold Time	t_{CAH}	45	–	55	–	ns
Column Address Hold Time referenced to $\overline{RAS}$	t_{AR}	95	–	120	–	ns
Read Command Set-up Time	t_{RCS}	0	–	0	–	ns
Read Command Hold Time	t_{RCH}	0	–	0	–	ns
Write Command Hold Time	t_{WCH}	45	–	55	–	ns
Write Command Hold Time referenced to $\overline{RAS}$	t_{WCR}	95	–	120	–	ns
Write Command Pulse Width	t_{WP}	45	–	55	–	ns
Write Command to $\overline{RAS}$ Lead Time	t_{RWL}	45	–	55	–	ns
Write Command to $\overline{CAS}$ Lead Time	t_{CWL}	45	–	55	–	ns
Data-in Set-up Time	t_{DS}	0	–	0	–	ns
Data-in Hold Time	t_{DH}	45	–	55	–	ns
Data-in Hold Time referenced to $\overline{RAS}$	t_{DHR}	95	–	120	–	ns
$\overline{CAS}$ Precharge Time (for Page-mode Cycle Only)	t_{CP}	60	–	80	–	ns
Refresh Period	t_{REF}	–	2	–	2	ms
$\overline{WE}$ Command Set-up Time	t_{WCS}	−20	–	−20	–	ns
$\overline{CAS}$ to $\overline{RAS}$ Delay	t_{CWD}	60	–	80	–	ns
$\overline{RAS}$ to $\overline{WE}$ Delay	t_{RWD}	110	–	145	–	ns
$\overline{RAS}$ Precharge to $\overline{CAS}$ Hold Time	t_{RPC}	0	–	0	–	ns

Tabla 4-3: Condiciones recomendadas de operación en AC

• "$\overline{\text{RAS}}$-ONLY" REFRESH CYCLE

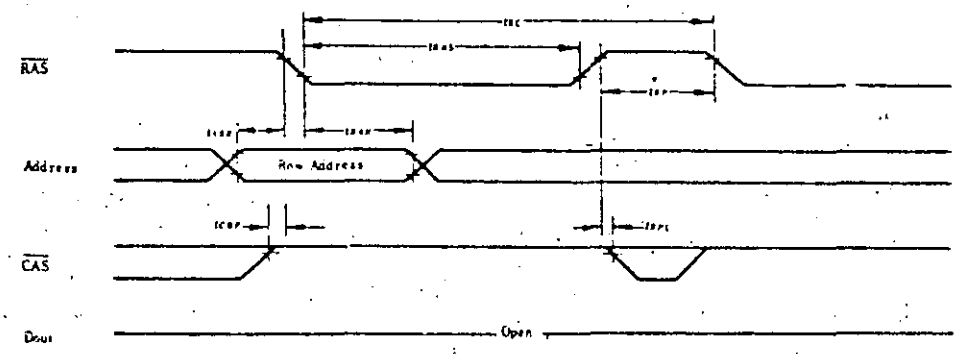


Figura 4-24: Ciclo de refresco para la memoria 4864

La figura 4-24 muestra el ciclo de refresco, en el cual no se habilita CAS, esto hace que no se seleccione una celda en específico sino todo un renglón. La señal de CAS sirve, también para la selección de chips, o sea, que al habilitar CAS solo en los chips que se deseé, los demás permanecerán inhibidos o bien efectuarán un ciclo de refresco si es que todos los chips están conectados a RAS.

4.3. CHIPS DE SOPORTE PARA PERIFERICOS

Un requerimiento importante para el exito de un microprocesador es fabricar chips adicionales que faciliten integrar el micro en las aplicaciones que se deseen. Estos chips normalmente usan las mismas señales que el micro y se conenctan directamente al procesador, reduciendo al mínimo los esfuerzos del diseñador para interfasear el micro con el mundo real. Todos los fabricantes siguen este enfoque y cada uno tiene su propia familia de chips con características muy especiales. Lo que si resulta problematico es utilizar los chips de soporte de otra familia, porque en algunos casos hay que añadir muchos componentes adicionales para que se puedan adaptar y usar en forma óptima.

Algunos aspectos que los fabricantes han tratado de contemplar al diseñar sus familias son:

- Selección simple del chip.

- Acceso directo al bus de datos del procesador.
- Usar las mismas fuentes de alimentación que el procesador.
- Usar el mismo tipo de reloj que el procesador.
- Usar el mismo mecanismo de "reset" que el procesador.
- Manejo de interrupciones directo.
- Incluir el mecanismo de prioridad de las interrupciones.
- Programación directa de los chips usando el conjunto de instrucciones del procesador.
- Lectura o escritura de datos inmediata.

Entre los dispositivos que más se usan para integrarlos en chips se encuentran los siguientes:

- Puertos paralelos. Para manejar líneas de dos estados bien sean de entrada o salida.
- Puertos seriales. Para manejar comunicaciones seriales síncronas o asíncronas.
- Contadores y controladores de eventos. Permite contar eventos que se produscan en forma síncrona o asíncrona, o bien, producir un cierto número de eventos a determinada frecuencia.
- Controladores de tubos de rayos catódicos para la generación de video.
- Manejadores de teclados, que eliminan el rebote y entregan los datos en determinado código.
- Manejadores de DMA, para facilitar la transferencia de datos entre periférico y memoria o viceversa, o entre periféricos o entre zonas de memoria.
- Controladores de discos flexibles para manejar discos flexibles de 8 y 5 y 1/4 de pulgada.
- Unidades manejadoras de memoria, las cuales permiten manejar memoria virtual en forma paginada o segmentada.

- Chips encriptadores de datos para transmitir información codificada que ocultan el texto de la información.

- Manejadores de protocolos para la transmisión y recepción de información.

- Multiplicadores y divisores de alta velocidad

- Unidades para operaciones aritméticas en punto flotante.

De toda la gama enorme de chips de soporte que ya existen en el mercado se van a revisar con cierto detalle los siguientes tipos de chips:

4.3.1. PUERTOS PARALELOS "PIO"

El Z80 Parallel I/O (PIO) es un dispositivo que tiene dos puertos programables y es compatible con TTL, tanto hacia los dispositivos periféricos que se conecte como hacia el CPU. El CPU puede configurar el PIO de tal forma que puede conectarse con un amplio rango de periféricos que no requieren lógica adicional. Dispositivos típicos que pueden usarse con el PIO son: teclados, lectoras de cinta de papel, impresoras, programadores de PROMs, switches, focos, leds, etc. El PIO utiliza tecnología NMOS y esta empquetado en un chip de 40 patas. Las principales características del PIO son:

- Dos puertos bidireccionales independientes cada uno con señales de control.

- Interrupción programada en las líneas de control para un rápida respuesta.

- Cuatro modos de operación de los puertos: salida byte, entrada byte, byte bidireccional, modo de control bit. Todos con interrupción programada y controlada.

- Lógica de interrupción de prioridades tipo cadena "daysi chain". Esto provee un mecanismo de interrupciones vectorizadas automático sin requerir lógica externa.

- Las ocho salidas son capaces de manejar transistores de tipo Darlington.

- Las ocho salidas y entradas de los puertos son totalmente compatibles con TTL.

- Requiere una sola fuente de alimentaci"on de 5 volts y un reloj de una sola fase.

Una de las características del PIO que lo diferencia bastante de otros chips que tienen puertos paralelos es que la transferencia de datos entre el periférico y el CPU se puede efectuar bajo un estricto control de interrupciones. La lógica de interrupciones del PIO permite un óptimo uso de la capacidad de interrupción del CPU durante transferencias de I/O. Toda la lógica necesaria para construir una estructura de interrupciones anidada está incluida en el PIO, por lo que no se requieren circuitos adicionales para tal efecto. Otra característica importante del PIO es que este puede ser programado para interrumpir al CPU cuando ocurran condiciones de estado espcificadas previamente. Por ejemplo, el PIO puede ser programado para interrumpir cuando ocurra una condicion de alarma en el periférico o dispositivo que está manejando. La interrupción evita que el procesador tenga que estar sensando continuamente esa condición de alarma, y por el contrario puede dedicarse despreocupadamente a otras actividades.

4.3.1.1. ARQUITECTURA INTERNA DEL PIO

La estructura interna del PIO consiste de una interfase hacia el CPU, lógica de control interna y bloque de control de interrupciones.

Los dos puertos son prácticamente idénticos y cada uno de ellos está compuesto de 6 registros con lógica de control incluida (handshake), tal como se muestra en la fig 4-26.

4.3.1.2. DESCRIPCION EXTERNA DEL PIO

El chip se compone de 40 patas 18 de las cuales son para conectarse con el CPU, 2 de la fuente de alimentación y 10 de cada puerto.

A continuación se decriben cada una las patas del chip:

D7-D0	Bus de datos del CPU, bidireccional, tipo tres estados.
B/A	Selección del puerto A o B, 0 selección del puerto A, 1 puerto B.
C/D	Selección de datos o control. 0 datos, 1 control (para recibir comandos del CPU).

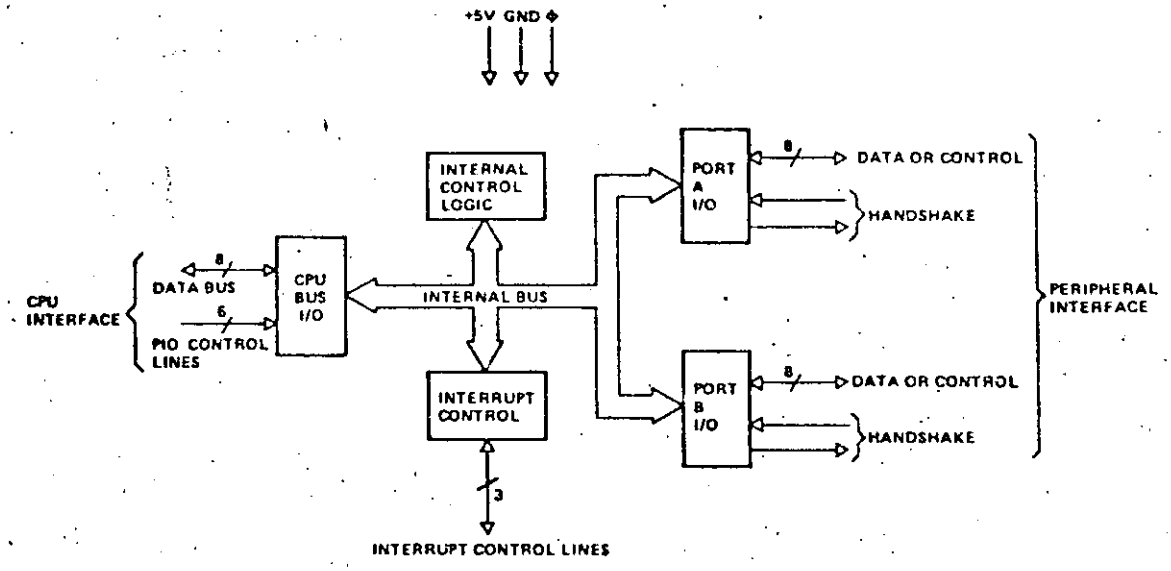


Figura 4-25: Diagrama de bloques interno del PIO

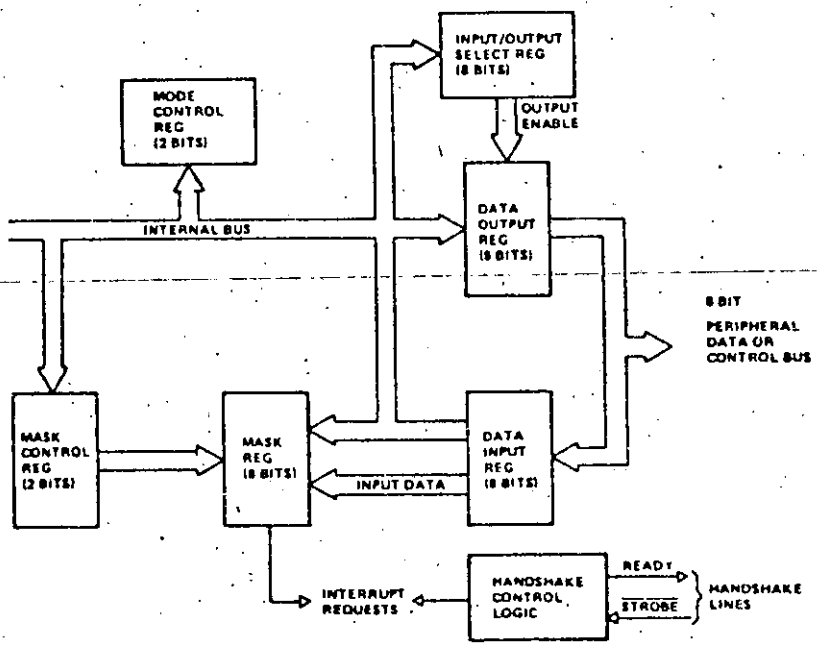


Figura 4-26: Diagrama de bloques del puerto

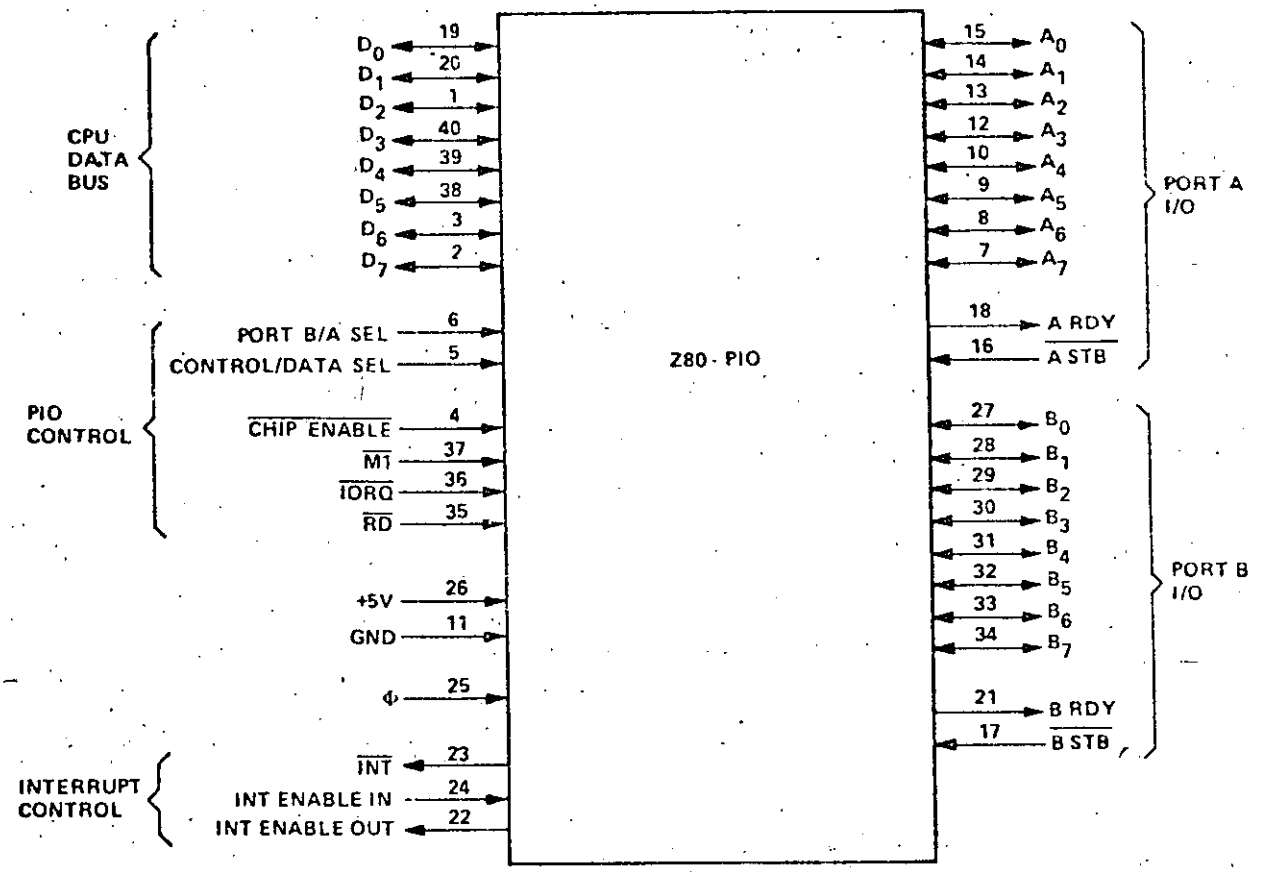


Figura 4-27: Descripción externa del PIO

CE — Habilitación del chip, 0 habilita el chip, 1 deshabilita.

Φ — Reloj del sistema, mismo que el del Z80-CPU.

M1 — Indicación del inicio de un ciclo de instrucción, señal que viene del CPU.

IORQ — Requerimiento de entrada o salida. Esta señal también la genera el CPU y se produce cada que se realiza una instrucción de salida (OUT) o entrada (IN).

RD — Indicación de ciclo de lectura. Esta señal, también, es generada por el CPU y cuando está en bajo indica que se está realizando un ciclo de lectura.

IEI — "Interrupt enable in". Señal de entrada al PIO,

sirve para formar la cadena de dispositivos que van a interrumpir al CPU. Esta señal si es 0 indica que un dispositivo de mayor prioridad está interrumpiendo.

IEO "Interrupt enable out". Señal de salida del PIO, sirve para formar la cadena de dispositivos que van a interrumpir al CPU. Cuando un puerto del PIO interrumpe está señal se va inmediatamente a 0 y permanece en este estado hasta que la rutina de atención de interrupciones haya sido atendida.

INT Señal de interrupción que va al CPU.

A0-A7 Puerto A del PIO, son 8 líneas de proposito general pueden ser entradas o salidas, salidas de tres estados.

ASTB Pulso de "strobe" del puerto A. El significado de esta señal es distinto, según sea el modo de operación del puerto A. En la sección de modos de operación se verá con mayor detalle la utilidad de esta señal.

ARDY Señal de "ready" del puerto A. El significado de esta señal depende del modo de operación seleccinado para el puerto A. En la sección de modos de operación se analizará la utilidad de esta señal.

B0-B7 Puerto B del PIO. Son 8 líneas de proposito general que pueden ser programadas como salidas o entradas. Las señales de salida son de tres estados.

BSTB Pulso de "strobe" del puerto B. Mismo caso que ASTB.

BRDY Señal de "ready" del puerto B. Mismo caso que ARDY.

4.3.1.3. PROGRAMACION DEL PIO

El PIO ha sido diseñado para operar con el Z80-CPU usando el modo de interrupción 2. Este modo requiere que un vector de interrupciones sea proporcionado por el dispositivo que interrumpe al CPU en el momento en que se reconoce esa interrupción. Este vector es usado por el CPU para formar la dirección de la rutina de atención de la interrupción de este puerto. El vector de interrupción es cargado en el PIO

escribiendo un solo byte en el registro de control del puerto deseado del PIO. Este byte debe tener un Ø en el bit menos significativo (dØ).

Existen 4 modos de operación, para programar un puerto. La manera de realizarlo es escribir en el registro de control del puerto un byte de programación, el cual debe tener en los dos bits más significantes indicado en binario el modo, de la siguiente manera:

D7	D6	D5	D4	D3	D2	D1	DØ	MODO	
Ø	Ø	x	x	1	1	1	1	Ø	salida
Ø	1	x	x	1	1	1	1	1	entrada
1	Ø	x	x	1	1	1	1	2	bidireccional
1	1	x	x	1	1	1	1	3	bits independ.

Tabla 4-4: Selección del modo en el PIO

En los modos Ø, 1 y 2 los 8 bits queda programados idénticos. En cambio en el modo 3 los bits se pueden programar en forma independiente algunos de salida y otros de entrada. La manera como indicar al PIO que bits quedan de salida y que otros de entrada es escribiendo en la palabra de control, después de haber programado el modo 3, un Ø en los bits correspondientes que serán de salida y un 1 en los bits que quedarán como entrada.

La manera de como habilitar o deshabiltar interrupciones es escribiendo en la palabra de control un Ø en D7 para deshabilitar y un 1 en D7 para habilitar las interrupciones. En el modo 3 se puede formar una función booleana para permitir la interrupción. Si el bit D6 en la palabra de control de interrupciones está en Ø indica que se forma una OR entre todas las señales de entrada para interrumpir, o sea que con cualquier entrada que tenga estado de interrupción interrumpirá al CPU, si D6 es Ø entonces se forma una AND, o sea que solo que todas las señales de entrada tengan estado de interrupción se interrumpirá al CPU. En el bit D5 de la palabra de control de interrupciones se indica el estado de interrupción, si este bit es Ø el estado de interrupción es bajo y si es 1 es alto. El bit D4 de la palabra de control de interrupciones sirve para indicar que a continuación se escribirá un nuevo dato (mascará) en la palabra de control del puerto, el cual indicará que bits del puerto se van a monitorear para la interrupción. Los otros bits de la palabra de control de interrupciones deben ser D3 = Ø y D2 = D1 = DØ = 1.

La mascara debe tener Ø en aquellos bits que se van a monitorear para generar la interrupción.

4.3.1.4. MODOS DE OPERACION

En el modo salida los 8 bits son señales de salida del puerto y todas se escriben en paralelo.

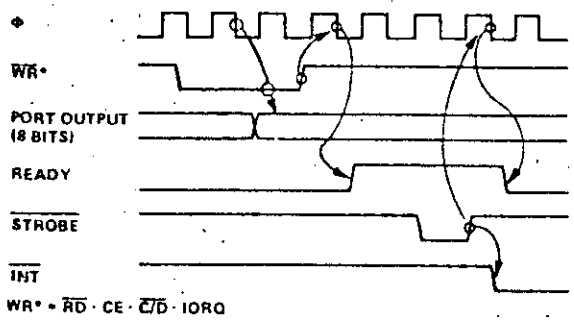


Figura 4-28: Modo 0 salida del puerto

En el modo entrada los 8 bits son señales de entrada al puerto y todas se leen en paralelo.

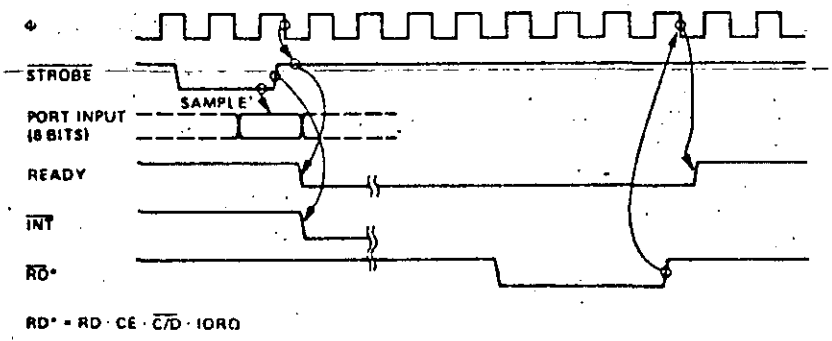


Figura 4-29: Modo 1 entrada al puerto

El modo bidireccional sirve solamente para el puerto A ya que usa las 4 señales de protocolo (nandsnake), las dos del puerto A y las dos del puerto B. Al programar el puerto A en modo bidireccional el puerto B debe estar programado en modo bit ya que es el único modo que no usa las señales de protocolo (nandsnake).

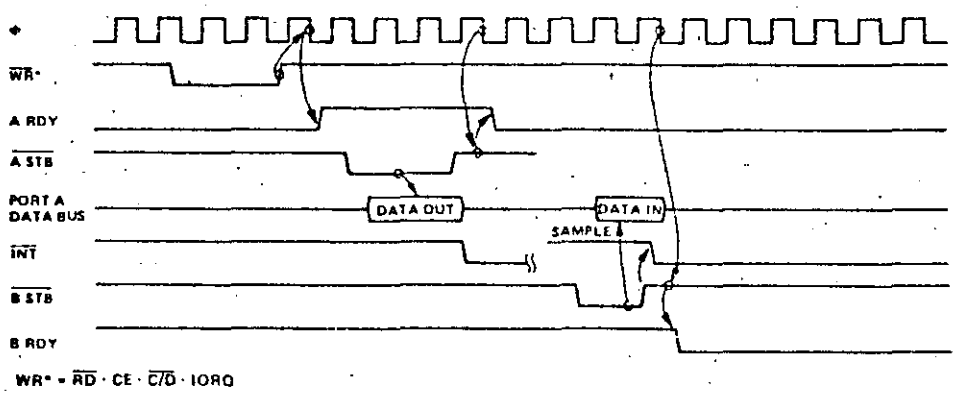


Figura 4-30: Puerto A en modo 2, bidireccional

El modo 3 o modo bit, programa los bits en forma independiente algunos como salida otros como entrada. Este modo no usa las señales de protocolo (handshake) ya que los mismos bits son los que producen la interrupción.

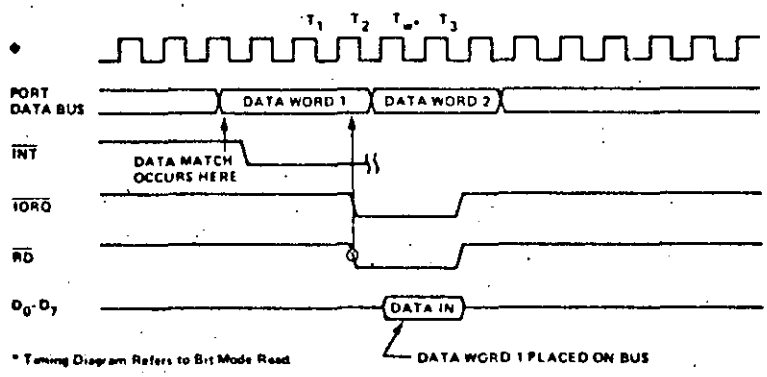


Figura 4-31: Modo 3, bits independientes

4.3.1.5. ATENCION DE INTERRUPCIONES

Poco tiempo después de que un PIO ha solicitado interrupción el CPU atenderá la misma produciendo un ciclo de atención de interrupción, el cual se caracteriza porque tanto IORQ como M1 estan en bajo. Esto hace que INT vuelva a subir y el procesador lee el vector de interrupción que entrega el PIO. Con este vector (parte menos significativa) y el contenido del registro I forma la dirección en la cual se encuentra el apuntador de la rutina de atención de interrupciones de este puerto. El CPU previamente ha salvado en el stack la dirección de retorno a la cual debe volver una vez que salga de la rutina de atención de interrupciones.

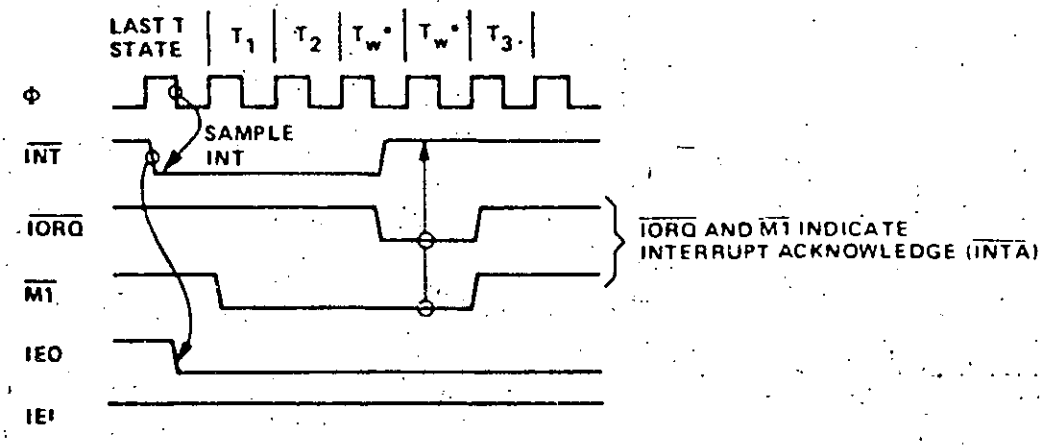


Figura 4-32: Ciclo de reconocimiento de interrupción

La señal de IEO del PIO cambia a Ø desde el momento en que se solicita la interrupción hasta que el procesador sale de la rutina de atención de interrupción. El PIO se da cuenta de esto último porque el procesador ejecuta la instrucción RETI (retorno de interrupción), la cual es decodificada por el PIO.

La manera como se organiza la estructura de prioridades de interrupción es formando una cadena con los puertos llamada "daysi chain". En esta cadena el primer puerto tiene IEI conectado a Vcc, por lo que es el puerto de mayor prioridad (cabeza de la cadena), los puertos que siguen tienen cada uno una prioridad menor a medida la distancia que lo separa de la cabeza de la cadena es mayor.

La figura 4-33 muestra un ejemplo de como un puerto de mayor

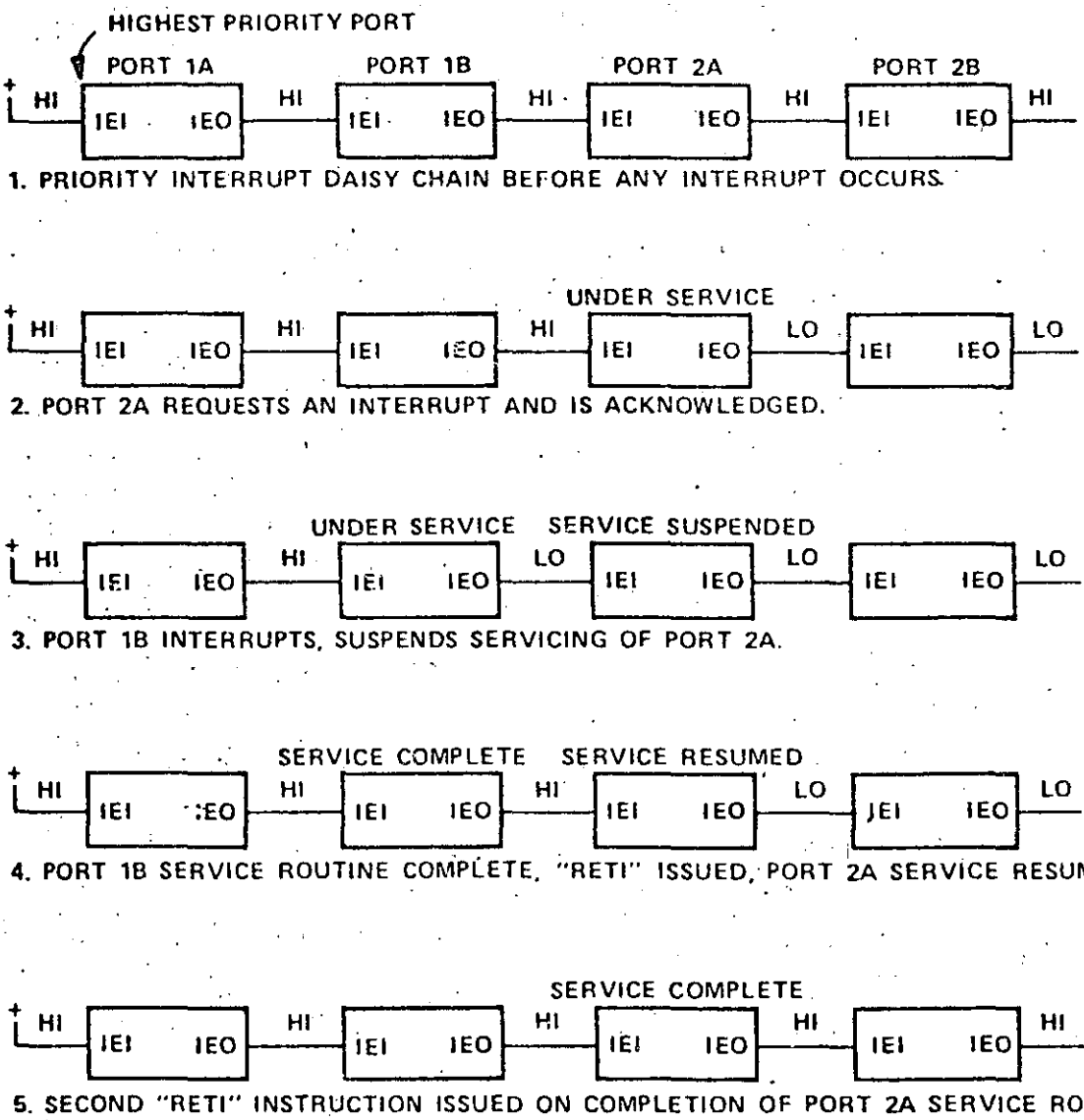


Figura 4-33: Atención de interrupciones en una estructura de prioridades del tipo cadena

prioridad puede interrumpir el servicio de atención de interrupciones de un puerto de menor prioridad.

4.3.2. PUERTOS SERIALES "SIO"

El Z80-SIO (Serial Input/Output) es un dispositivo programable de doble canal el cual provee el formateo de los datos para la comunicación de datos seriales. Es capaz de manejar comunicaciones asíncronas, síncronas y protocolos orientados por bit síncronos tales como IBM BiSync, HDLC y SDLC. Tiene la capacidad de generar código CRC en cualquier modo síncrono y puede ser programado por el CPU para cualquier formato asíncrono tradicional.

Una tecnología NMOS de silicio, el chip es de 40 patas, usa una sola fuente de voltaje de 5 volts y un solo reloj de una sola fase a 5 volts. Los dos canales pueden ser programados para operar en modo comunicación simultánea (full duplex).

Algunas de las características sobresalientes de este chip son:

- Dos canales independientes tipo "full duplex".
- Velocidades de comunicación desde 0 hasta 550 Kbits/seg.
- Registros de datos de recepción "buffereados" 4 veces y los registros de datos de transmisión "buffereados" doblemente.
- Operación asíncrona: 5, 6, 7 u 8 bits por caracter, 1 1y1/2 o 2 bits de fin "stop", paridad par o impar o no paridad, operaciones del reloj de x1, x16, x32 y x64. Generación y detección de "break", y finalmente detección de errores de paridad, "overrun" y "framing".
- Operación síncrona: Internos o externos los caracteres de sincronización, uno o dos caracteres de sincronización en registros separados, inclusión automática del caracter de sincronización. Automática generación y chequeo de CRC.
- Operación HDLC o IBM SDLC: Quitado o inclusión automática de Zero. Inclusión automática de bandera, reconocimiento automático del campo de dirección, manejo del residuo del campo I, mensajes de recepción válidos son protegidos de "overrun", generación y chequeo de CRC.
- Ocho líneas de control de entrada y salida para modem.
- Están implementados dos tipos de CRC, tanto CRC-16 como CRC-CCITT (-0 y -1).

- Se incluye la lógica de interrupción de prioridades del tipo cadena con el fin de proveer un vector de interrupción automático sin necesidad de requerir lógica externa.
- Todas las entradas y salidas son compatibles totalmente con TTL.

98882

4.3.2.1. ESTRUCTURA INTERNA DEL SIO

El SIO tiene 6 bloques básicos, los dos canales A y B, el bloque para el manejo de las señales del modem, lógica interna, lógica de control de interrupciones y la interfase con el CPU.

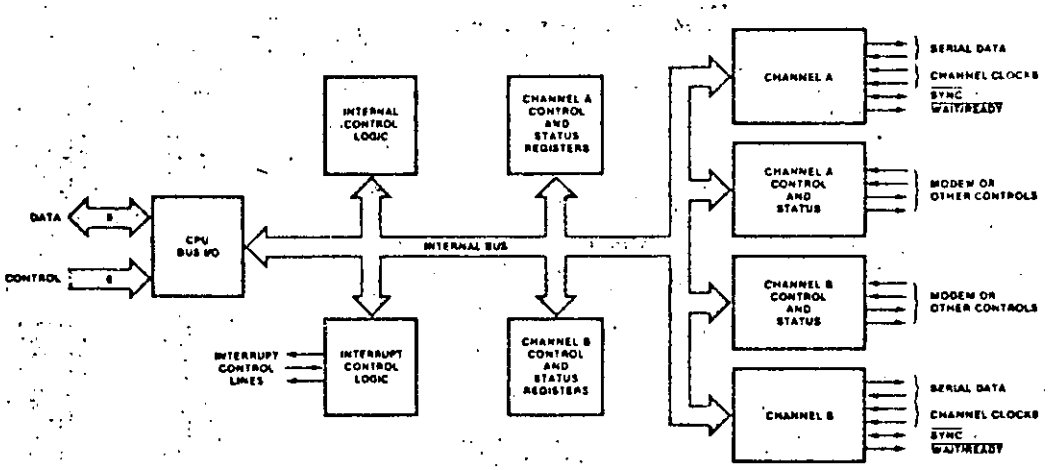


Figura 4-34: Diagrama de bloques del SIO

La prioridad entre los canales y dispositivos dentro del canal es fija y está determinada como sigue: el canal A tiene mayor prioridad que el canal B. Dentro de los dispositivos del canal el receptor tiene mayor prioridad, luego el transmisor y finalmente el status externo.

La lógica interna de cada canal a nivel de bloques está mostrada en la figura 4-35. Cada canal tiene:

- 5 registros de control de 8 bits cada uno.
- 2 registros de estatus de 8 bits cada uno.

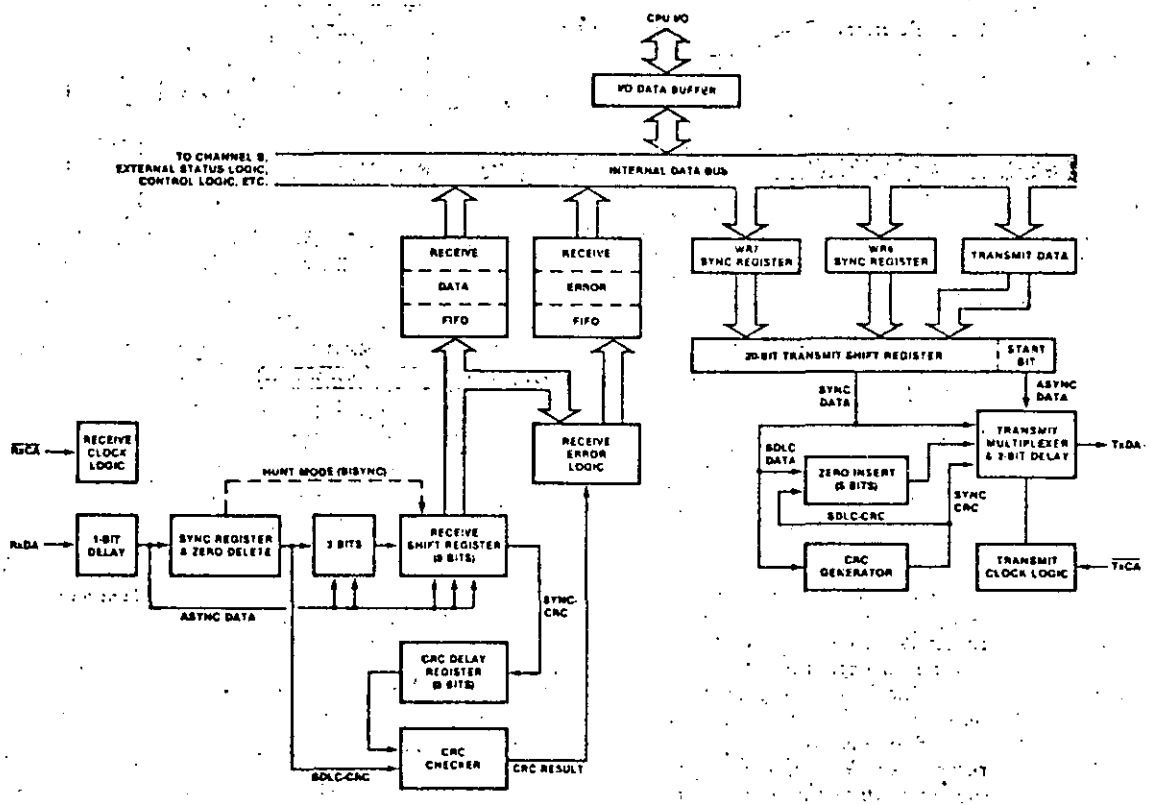


Figura 4-35: Diagrama de bloques interno del canal

- 2 registros para caracteres de sincronía de 8 bits.
- El receptor tiene 3 registros de 8 bits arreglados en forma de FIFO además del registro de corrimiento de entrada de 8 bits.
- El transmisor tiene un registro de 8 bits además del registro de corrimiento de salida de 8 bits.
- El vector de interrupciones es único y se programa en el canal B.
- El CRC generador/checador es un registro de corrimiento de 16 bits con realimentación interna apropiada, programable para dos diferentes códigos CRC.

4.3.2.2. DESCRIPCION EXTERNA DEL SIO

El SIO como el PIO tiene dos zonas, la interfase con el CPU y los puertos para la comunicación.

D0-D7	Bus de datos bidireccional.
B/A	Selección de canal A (0) o B (1).
C/D	Selección de datos (0) o control (1).
CE	Selección del chip, activo bajo.
M1	Inicio del ciclo de instrucción, señal del Z80-CPU.
IORQ	Requerimiento de entrada/salida, señal del Z80-CPU.
RD	Ciclo de lectura, señal del Z80-CPU.
Ø	Reloj del sistema.
RESET	Deshabilita transmisores y receptores, borra todos los registros e inicializa nuevamente todos los dispositivos. Es recomendable dar un RESET (activo abjo) después de prender la máquina.
IEI	Para formar la cadena de prioridade de interrupción.
IEO	Para formar la cadena de prioridades de interrupción.
INT	Requerimiento de interrupción, activa baja.
WAIT/READY A, B	Son dos patas una por cada canal A y B. Se pueden programar para servir como líneas "ready" de un controlador de DMA, o bien pueden servir para detener la ejecución del CPU (wait) para que se sincronice con la velocidad del SIO.
CTS A, B	Son señales de entrada "clear to send", pueden servir para el modem o bien como señales de proposito general. Cuando se programan como auto nabilitación, inniben los transmisores de sus respectivos canales.
DCD A, B	Son señales de entrada, "data carrier detect", sus funciones son similares al CTS, excepto que

ellas son usadas para inhibir los receptores. Se usa, también, con modem y en algunos casos esta señal es conocida como DSR "data set ready".

RxD A, B — Señales de recepción de datos seriales.

TxD A, B — Señales de transmisión de datos seriales.

RxC A, B — Reloj de recepción. El reloj puede ser x1, x16, x32 o x64 la velocidad de recepción de los datos en modo asíncrono.

TxC A, B — Reloj de transmisión. El mismo comportamiento que el reloj de recepción.

RTS A, B — Son señales de salida, "request to send", cuando el bit de programación RTS es 1 la señal respectiva RTS se va a bajo. Cuando el bit RTS es 0 en modo asíncrono la señal RTS se ba a alto cada vez que el buffer de transmisión está vacio. En modo síncrono la señal RTS es una simple salida que sigue estrictamente el estado del bit RTS.

DTR A, B — Son señales de salida "data terminal ready", estas señales siguen esctrictamente la programación del bit DTR.

SYNC A, B — Caracter de sincronización externo. Si el modo de sincronización externa es seleccionado el ensamblado de caracteres comenzará con el próximo flanco de subida de RxC. Si el modo de sincronización de caracteres interno es seleccionado, estas señales son activadas durante la parte de los ciclos de reloj que el caracter de sincronización es reconocido. La condición de sincronización no es almacenada por lo que esta señal será activada cada vez que un patron de sincronización es reconocido y no en los caracteres de frontera. En modo asíncrono estas líneas son señales de entrada al bit Hunt/Sync del registro 0 de estatus y pueden ser usadas como una función de entrada de proposito general.

4.3.2.3. MODELOS DE SIOs

El requerimiento del número de patas del SIO es 41, sin embargo el chip tiene solo 40, por lo que debido a la restricción en el número de patas se decidio fabricar cuatro modelos del chip SIO, tres de ellos tienen los dos canales en los cuales hnubo que hacer ciertos trucos para reducir el número de patas a 40 y el cuarto es un SIO que tiene un solo canal completo, también en 40 patas. Los modelos del SIO son los siguientes:

- El Z80-SIO/0 el cual tiene juntos en un sola pata las señales de rejo de transmisión y recepción del canal B, es decir que en lugar de tener TxCA y TxCB tiene una sola línea que es RxTxCB.
- El Z80-SIO/1 el cual no dispone de la señal DTRB.
- El Z80-SIO/2 el cual no dispone de la señal SYNCB.
- El Z80-SIO/9 el cual tiene un solo canal que es el canal A, no maneja ninguna señal del canal B, sin embargo, la programación del vector de interrupciones que normalmente se realiza en el canal B se sigue haciendo ahí mismo.

4.3.2.4. FORMATOS DE OPERACION DEL SIO

Existe un formato típico para la operación en modo asíncrono, sin embargo, existen varios formatos para la operación en modo síncrono.

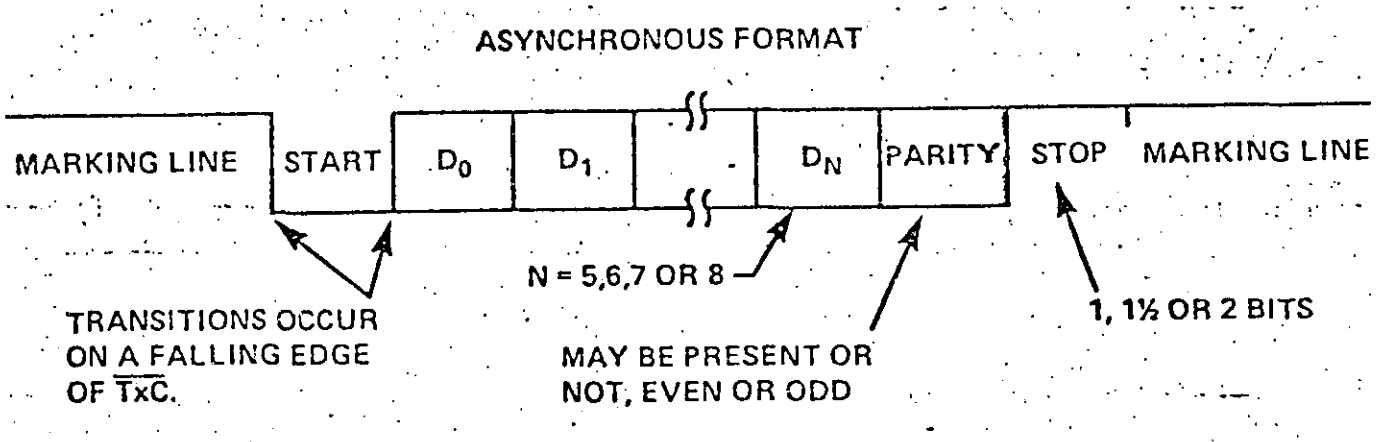
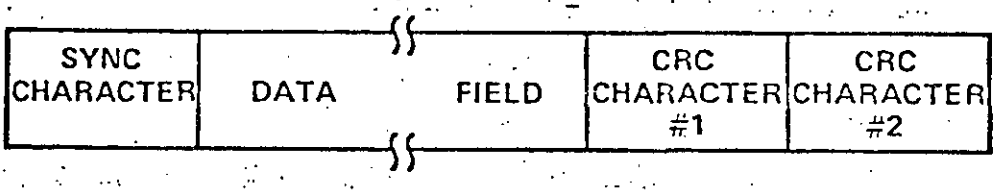


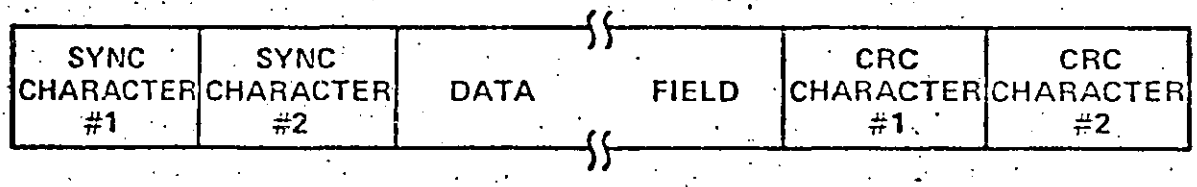
Figura 4-36: Formato de operación en modo asíncrono

La figura 4-36 muestra el formato de operación en modo asíncrono, asi como todas las opciones para este caso.

MONOSYNC MESSAGE FORMAT (Internal Sync Detect)



BISYNC MESSAGE FORMAT (Internal Sync Detect)



EXTERNAL SYNC DETECT FORMAT

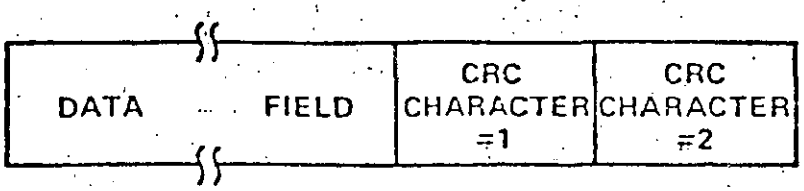


Figura 4-37: Formato de operación en modo síncrono

La figura 4-37 muestra tres formatos para la operación en modo síncrono, el formato monosíncrono, bisíncrono y el formato con sincronía externa.

TRANSMISSION

SDLC/HDLC Message Format

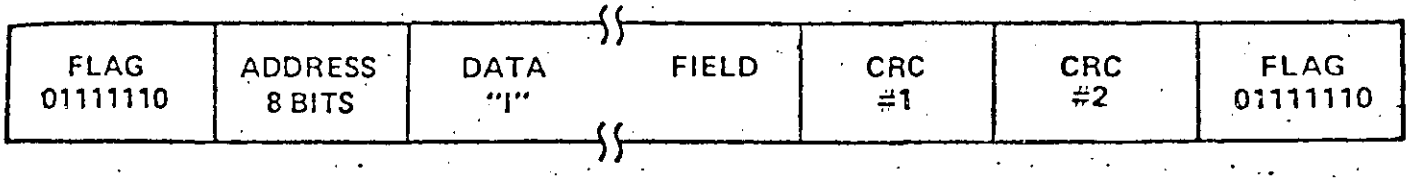


Figura 4-38: Formato para el modo síncrono SDLC/HDLC

4.3.2.5. PROGRAMACION DEL SIO

Para programar el SIO se usan 8 registros, llamados registros de escritura o registros de comando. El direccionamiento de los registros depende de los tres bits que tenga el registro de escritura 0 (WR0).

Siempre que se envie un comando por primera vez al SIO se direcciona el registro WR0, en el cual se indica que registro se direccionará posteriormente. A continuación se envia el comando que se referirá al registro seleccionado previamente. La función en general de cada registro de comando es la siguiente:

1. El registro WR1 es el que controla las interrupciones, o sea, habilita y deshabilita las mismas, tanto de transmisión como de recepción.

2. El registro WR2 sirve para el canal B solamente, en este registro se programa el vector de interrupción.

3. El registro WR3 es el registro para controlar la recepción tanto en modo síncrono como asíncrono.

4. El registro WR4 es un registro de caracter general y sirve para programar algunas características de la comunicación. Este es el registro que se debe programar antes que cualquier otro.

5. El registro WR5 es el registro que sirve para controlar la transmisión tanto en modo síncrono como asíncrono.

6. El registro WR6 sirve para programar el byte menos significante del caracter de sincronía interno o bien el campo de dirección cuando se usa el protocolo SDLC.

7. El registro WR7 sirve para programar el byte más significante del caracter de sincronía interno o bien el campo de bandera cuando se usa el protocolo SDLC.

Los registros de lectura son llamados también registros de estatus y sirven para leer el estado, errores o información referente a la comunicación. Existen tres registros de estatus, los cuales son direccionados de la misma manera que los registros de comando, indicando siempre en el registro WR0 que registro se desea leer. Si no se indica la dirección del registro en WR0, siempre que se lea la palabra de control del puerto se leerá el registro de lectura 0 (RR0).

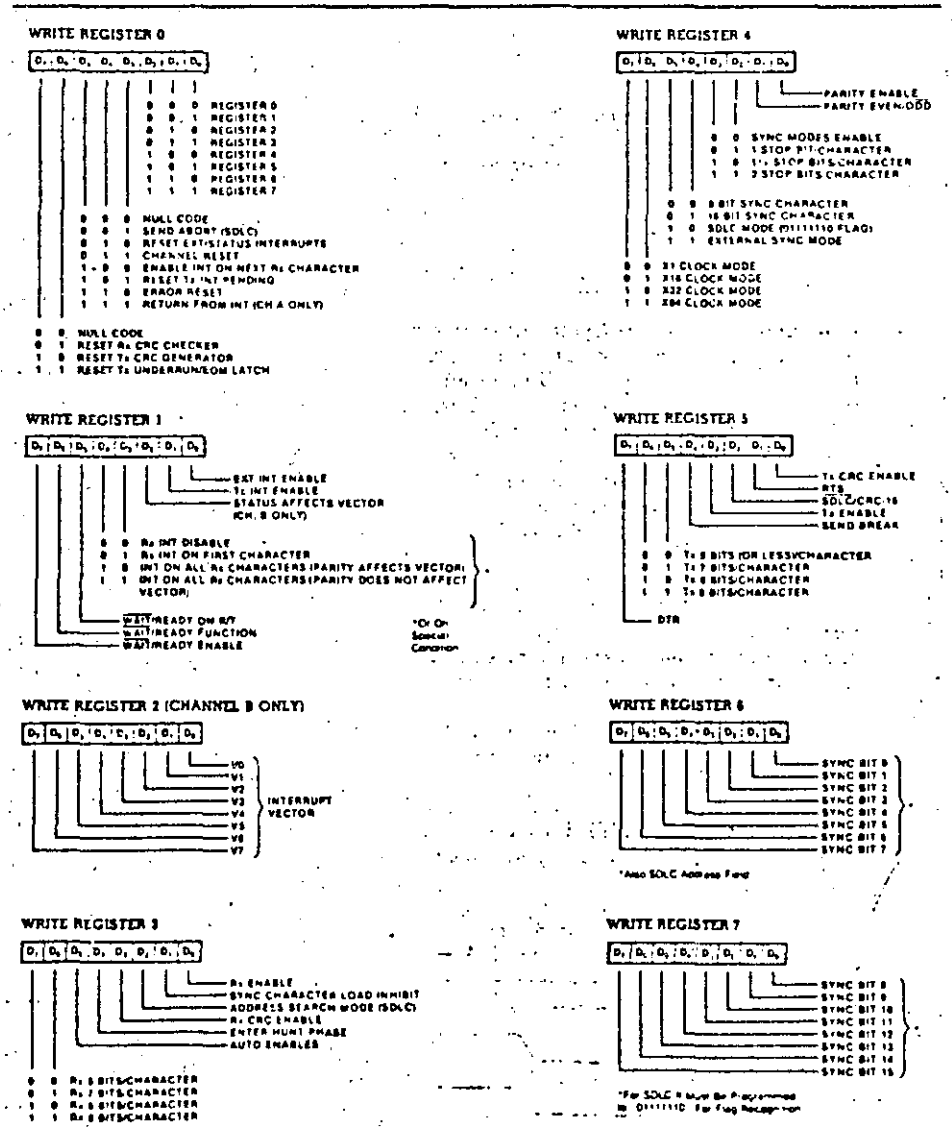


Figura 4-39: Funciones de los registros de comandos

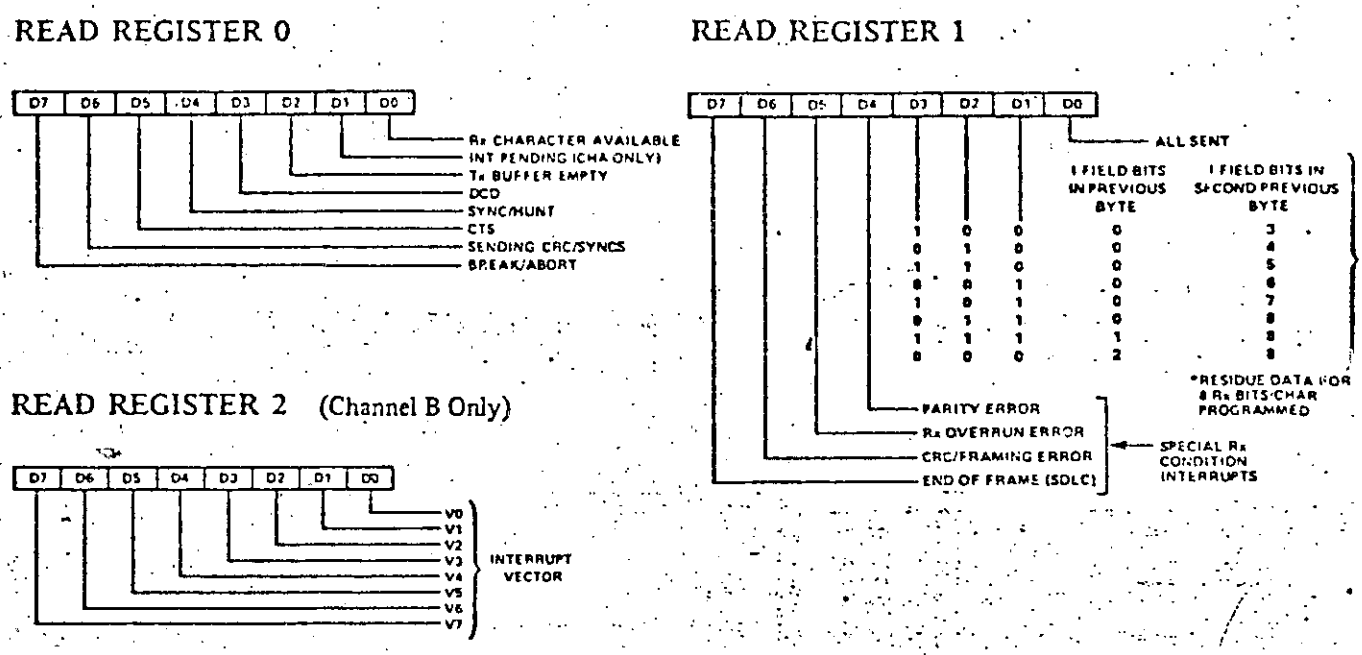


Figura 4-40: Funciones de los bits de los registros de lectura

El registro RR0 es muy importante porque en él se sabe, por ejemplo, si un caracter ya ha sido transmitido (buffer de transmisión vacio) o bien si se recibio un caracter o si existe una interrupción pendiente, etc. El registro RR1 sirve para leer los tipos de errores y el campo "1" del protocolo de comunicación síncrono SDLC/HDLC. El registro RR2 sirve para leer el vector de interrupción que se ha programado, este registro lo tiene únicamente el canal B.

4.3.2.6. MANEJO DE INTERRUPCIONES

En cada SIO se programa un solo vector de interrupción, sin embargo, el SIO tiene la opción de modificar este vector dependiendo del tipo de interrupción que se haya presentado. Sin esta opción resultaría tedioso tener que investigar por software a que se debe la interrupción cuando esta se presenta. Este problema se complica cuando existen varias interrupciones habilitadas, por ejemplo, cuando la interrupción de recepción y transmisión de ambos puertos están habilitadas. Por este motivo se puede escoger en el bit 2 del registro WR1 si se desea un solo vector de interrupción (que sería el programado previamente) o bien dependiendo de la interrupción que se afecte el vector de interrupción. Este bit solamente funciona en el canal B.

La figura 4-41 muestra los 8 casos de interrupciones que se pueden presentar en un SIO. Obsérvese que los bits que se afectan en el vector de interrupción son los bits 1, 2 y 3. El bit 0

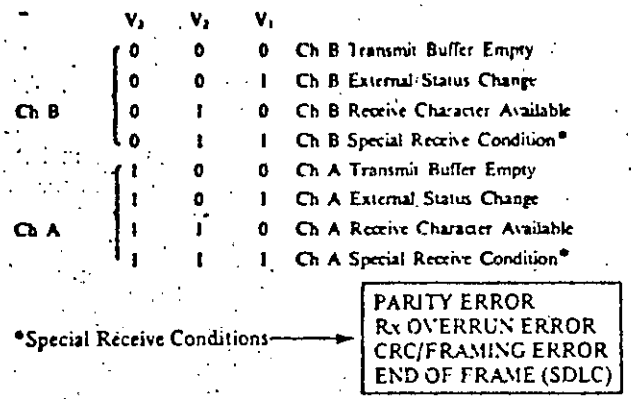


Figura 4-41: Forma en como se afecta el vector de interrupción

desde luego no se afecta para que en la tabla de vectores resulten 8 apuntadores a las rutinas de atención de interrupciónes.

4.4. MEMORIA MASIVA

Toda microcomputadora tiene algún medio para almacenar información en gran cantidad y en forma permanente. En las primeras microcomputadoras el medio más común ha sido la cinta de cassette, por lo barata y accesible. Los problemas que el cassette tiene son un tiempo de acceso muy grande, muy baja densidad y generalmente requiere intervención humana para cualquier actividad. Posteriormente se usó el disco flexible, debido a que mejora en gran medida el tiempo de acceso y la densidad respecto al cassette, además no requiere intervención humana para operar con el disco, sin embargo, es más caro que el cassette, tanto a nivel de la unidad manejadora "drive" como a nivel del medio de grabación. Inicialmente los discos flexibles fueron de 8 pulgadas de diametro, posteriormente aparecieron los de 5 y 1/4 de pulgada (minifloppies) que mejoran el costo de los anteriores pese a que tienen menor densidad. Ultimamente aparecieron los discos flexibles de 3 pulgadas de diametro (microfloppies) con capacidades de almacenamiento semejantes a los de 5 y 1/4 de pulgada. Poco antes de los microfloppies aparecieron los discos duros de 8 pulgadas de diametro, los cuales se comportan de manera muy semejante a los tradicionales discos de computadora de 14 pulgadas de diametro. Tienen un mejor tiempo de acceso y mucha mayor densidad que los discos

flexibles, el hecho de que no sean removibles es una desventaja así como el costo que es mayor que en los discos flexibles. Ultimamente han adquirido mucha popularidad los discos fijos duros llamados discos "wincnester", por su gran capacidad de almacenamiento, disponibilidad de una gran cantidad de fabricantes y costo bajo en comparación con discos de 14 pulgadas.

Todos estos dispositivos almacenan la información en un medio magnético.

4.4.1. FUNDAMENTOS DE GRABACION MAGNETICA

La grabación magnética provee gran capacidad de almacenamiento a bajo costo y con tiempos de acceso bastante razonables. Los requerimientos básicos para la grabación magnética son:

1. Medio de almacenamiento.
2. Mecanismo de escritura (grabado de la información).
3. Mecanismo de lectura y sensado (recuperación de la información).
4. Mecanismo de direccionamiento.

La grabación magnética se concentra en dos objetivos muy importantes, que son:

1. Conseguir una alta densidad de grabación, es decir, almacenar los bits lo más juntos posible.
2. Procurar que el margen de señal a ruido durante el proceso de lectura sea adecuado de tal manera que se pueda diferenciar perfectamente la información del ruido eléctrico. El proceso de lectura es el más delicado y susceptible a falla.

En la práctica el espacio ocupado por muchos bits grabados secuencialmente (densidad) está dictado principalmente por las propiedades del medio de grabación y por la cabeza de lectura.

4.4.1.1. MEDIO DE GRABACION MAGNETICA

Existen dos tipos de medios que son muy usados, el flexible para cintas y discos y el duro para discos. La diferencia entre estos radica en la parte no magnética, la cual determina prácticamente el grosor del medio, ya que la parte magnética es un muy delgada del orden de 0.5 mils (milesimas de pulgada). De esto podemos inducir que todo medio de grabación magnética consiste de dos partes una magnética y otra no magnética, ambas partes afectan la densidad y la razón señal a ruido del medio ya que existen variables magnéticas y no magnéticas que son interdependientes entre si y juntas determinan las propiedades del medio.

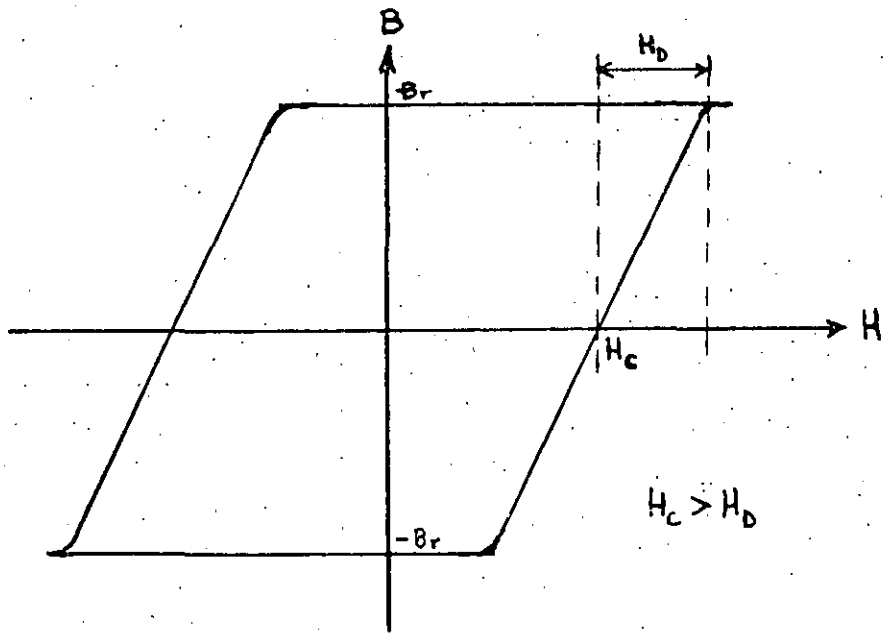


Figura 4-42: Comportamiento de un material ferromagnético, ciclo BH

Las propiedades del medio magnético:

1. Tamaño de la partícula en el oxido y el tamaño del grano del medio no magnético.
2. Fuerza cohercitiva Hc.
3. Densidad de flujo residual Br.
4. Grosor.

5. Rugosidad de la superficie.

6. Uniformidad de características sobre el medio completo.

4.4.1.2. MECANISMO DE ESCRITURA

El transductor de escritura en toda grabación magnética es un dispositivo ferromagnético que tiene una estructura toroidal con un gap (abertura muy pequeña).

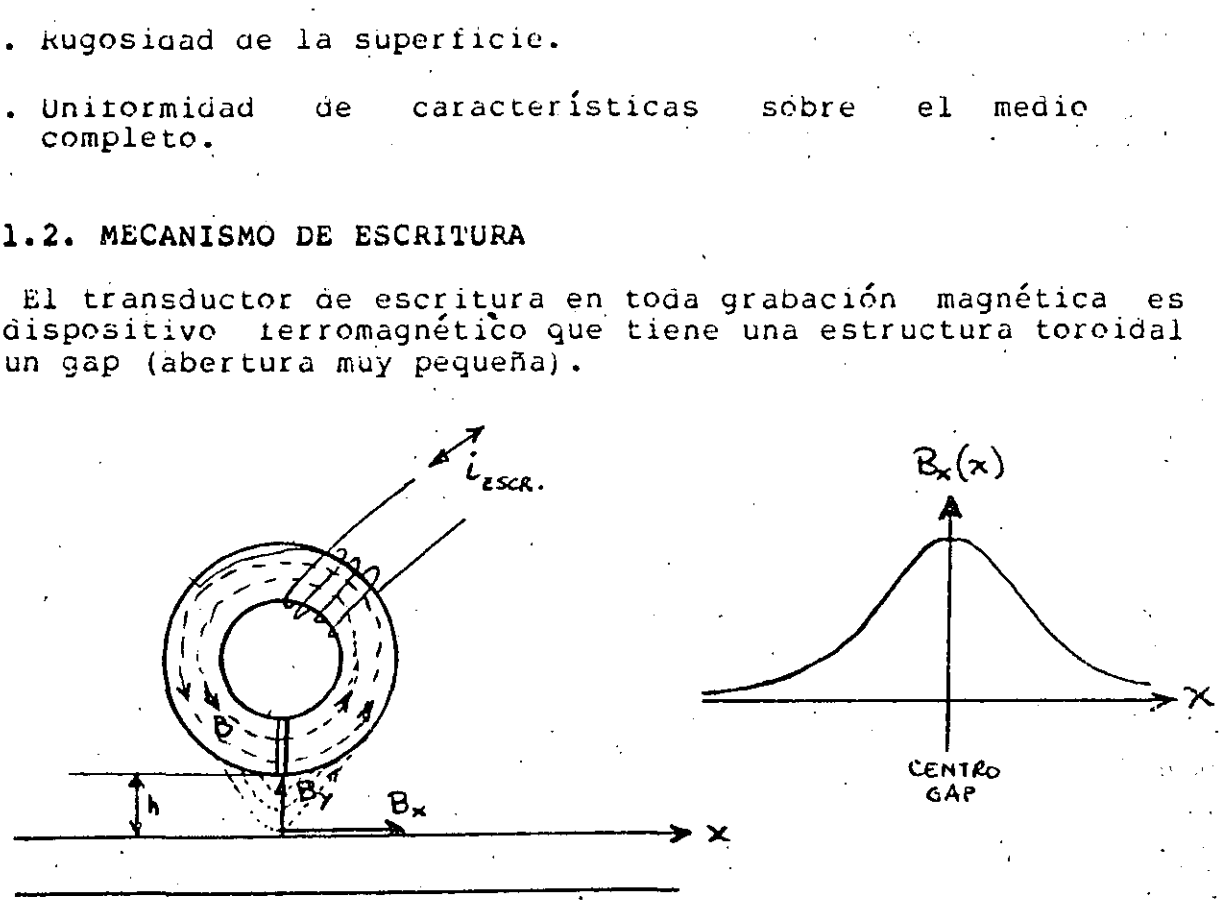


Figura 4-43: Flujo magnético en la cabeza

Al hacer circular una corriente en la bobina se induce un flujo magnético en el toroide de la figura 4-43. Este flujo magnético circula por la estructura del toroide, sin embargo, se expande en el gap hacia los lados en virtud de la descontinuidad del material. Este flujo magnético que rebalsa o salta el gap sobre todo en los bordes del toroide se llama "fringe", es el que efectua la escritura en la superficie magnética que pasa a una altura "h" de la cabeza. Para lograr una mejor escritura de los datos el gap debe ser diseñado de tal manera que se obtenga el máximo rebalse de flujo magnético y la altura h sea la mínima posible.

La manera como se escriben los datos es, el fringe ejerce una fuerza magnética que forza a las particulas del medio magnético a orientarse en terminada dirección de tal manera que se forman dipolos magnéticos en el medio. Estos dipolos

magnéticos son los que representan la información grabada en un medio magnético. La fuerza magnética que se ejerce en el medio adquiere su máximo valor en el centro del gap y decrece a ambos lados en forma de campana.

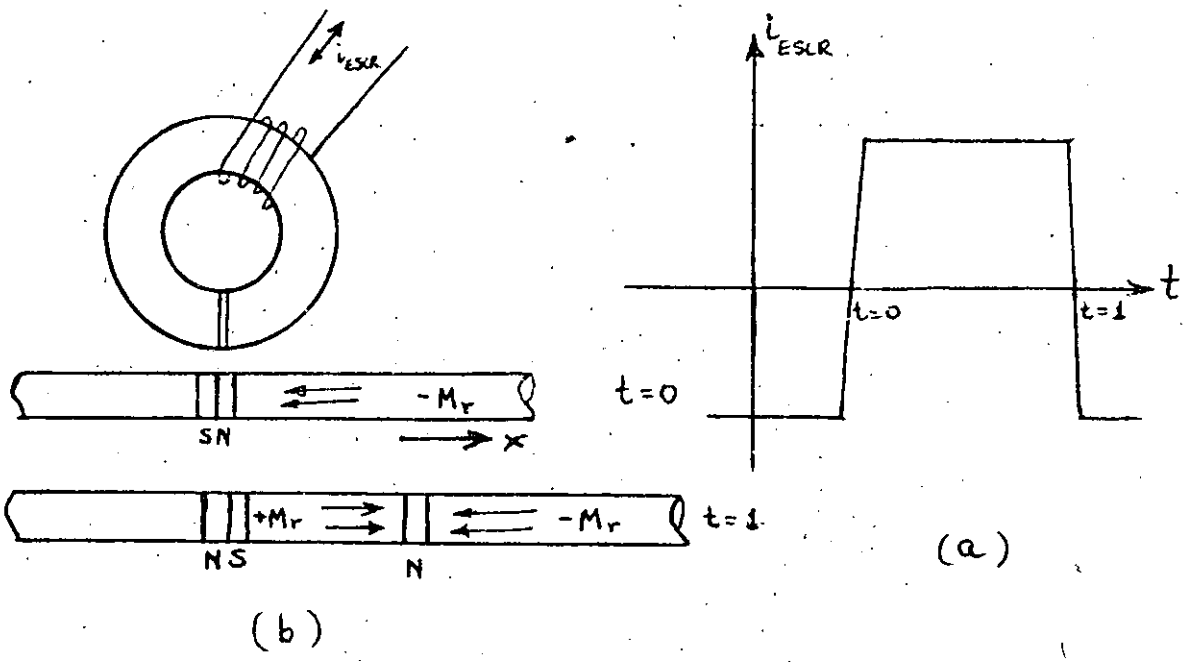


Figura 4-44: Efectos de la grabación magnética en el medio
(a) Corriente de escritura
(b) Dipolos magnéticos en el medio

La figura 4-44 muestra un ejemplo de como se graba la información en el medio magnético, formandose polos nortes y sures cuando la transición de la corriente de escritura cambia de menos a mas y de mas a menos respectivamente. Las particulas que quedan dentro del dipolo se agrupan y se orientan siempre de sur a norte. Las particulas del medio magnético tienen mucha facilidad de orientarse en la dirección X mientras que presentan gran dificultad de polarizarse en la dirección Z, por lo que el ancho de la polarización es igual al ancho de la cabeza.

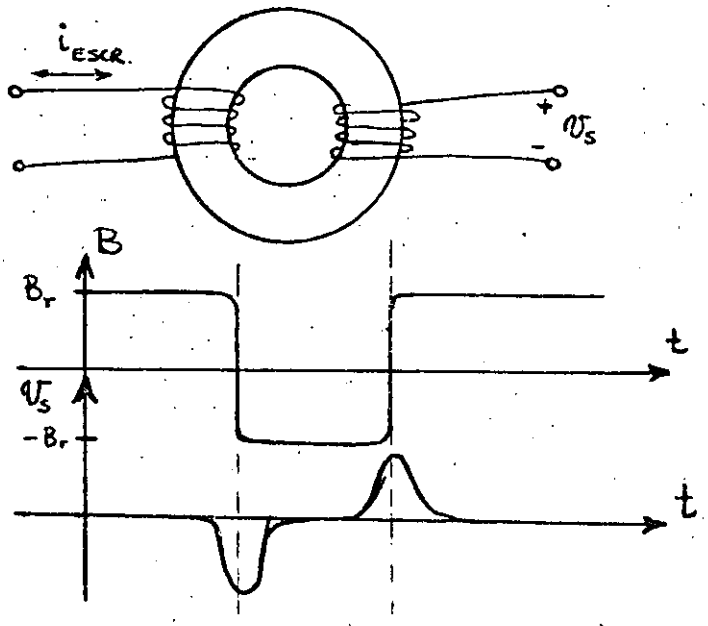
4.4.1.3. MECANISMO DE SENSADO O LECTURA

El problema en este caso es leer confiablemente y en forma precisa la información almacenada. La señal que se obtiene del disco depende de muchos factores, entre ellos, de la abertura del gap en la cabeza lectora, variaciones de velocidad del medio magnético (cinta o disco), de la distancia que separan la cabeza lectora del medio, etc.

En grabación magnética, la información que es escrita por

una cabeza puede ser sensada por la misma u otra cabeza en un tiempo posterior. A diferencia de RAM la posición exacta donde ha sido almacenada la información es desconocida. Si el proceso de lectura es sincronizado con un reloj externo, la frecuencia del mismo debe ser idéntico, dentro de ciertas tolerancias, a la frecuencia del reloj usado durante la escritura.

El proceso de sensado es idéntico al de la escritura, solo que a la inversa, se puede usar la misma cabeza de escritura solo que en lugar de aplicar una corriente al embobinado se obtiene una señal del mismo. Esta señal se genera de acuerdo con los dipolos magnéticos que contiene el disco o la cinta, el agrupamiento y polarización de las particulas que pasan por la cabeza inducen un campo magnético en el toroide el cual a su vez produce una corriente en el embobinado. Un aspecto muy importante que hay que considerar es que la señal de salida en el embobinado es directamente proporcional a los cambios de flujo magnético del toroide y no al flujo magnético mismo.



$$V \propto \frac{dB}{dt}$$

$$B \propto \int V \cdot dt$$

$$V \propto N \frac{d\phi}{dt}$$

Figura 4-45: Señal de sensado en la cabeza magnética

De la figura 4-45 se observa que la señal de sensado es perceptible solo cuando existe un polo magnético, o sea, cuando existe un cambio de flujo magnético en la cinta o disco, por lo tanto lo que se puede leer de la información que se graba son unicamente las transiciones de la señal de escritura.

El proceso de lectura es uno de los factores más limitantes en la densidad, porque aunque si se pueden grabar transiciones muy juntas unas de otras no se pueden leer confiablemente a menos

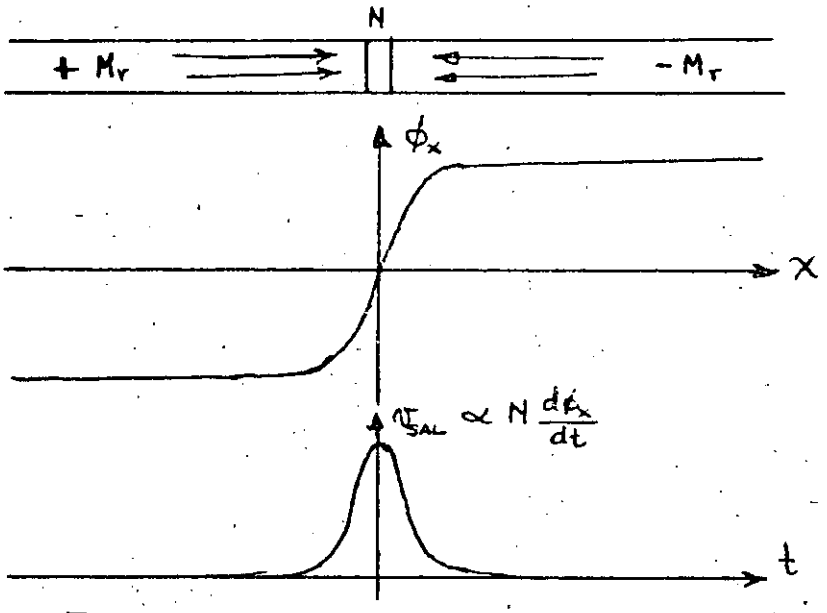


Figura 4-46: Señal de sensado

que estas transiciones respeten una distancia mínima entre ellas. Si existen dos transiciones muy juntas en la señal de lectura estas se perjudican entre si y el efecto es que ambas reducen su amplitud, lo cual es muy peligroso porque el margen de señal a ruido es muy pequeño y eventualmente se podrían considerar ruido.

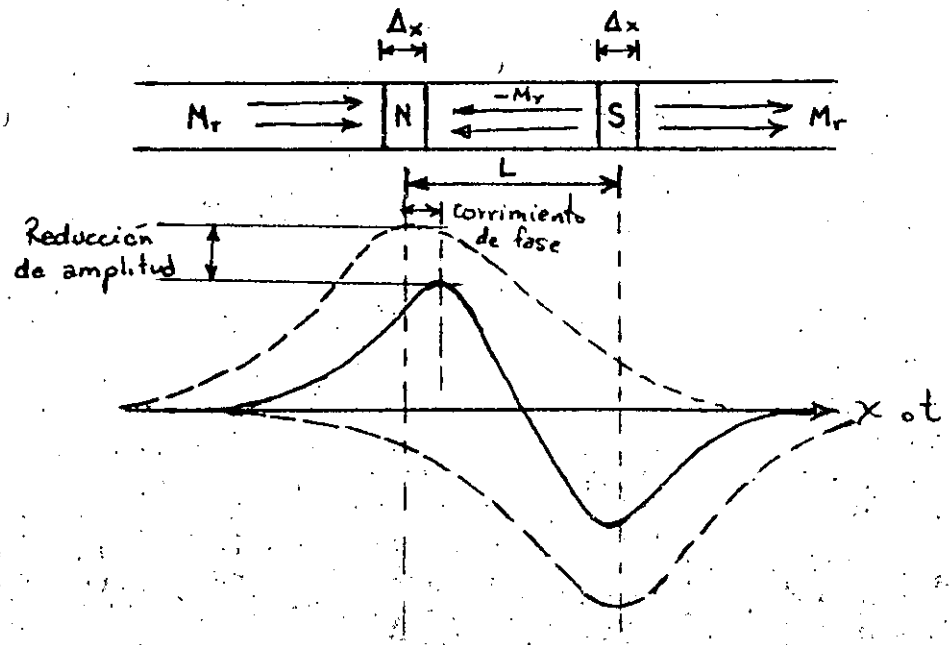


Figura 4-47: Efecto de juntar mucho las transiciones

4.4.1.4. MECANISMO DE DIRECCIONAMIENTO

En memorias RAM ROM, el direccinamiento está alambrado y para localizar una posición determinada, los decodificadores respectivos la ubican y permiten escribirla o leerla según sea la necesidad sin alterar ni pasar por las otras posiciones. La localización es directa y no crea conflictos ni dudas.

En discos o cintas no se conoce con exactitud la posición de los datos requeridos en virtud de que ni el disco y menos la cinta son memorias de acceso directo como la RAM. Son memorias secuenciales y es preciso recorrer un cierto espacio antes de llegar al dato requerido. La cinta y el disco deben tener ciertas marcas que sirven como referencia para localizar los datos.

El mecanismo que se sigue para localizar los datos es sacrificar cierto espacio del disco o cinta (que bien podría servir para almacenar más datos) para guardar la identificación de los datos o señales de referencia y de sincronización. Por lo tanto es usual dividir el espacio de grabación en registros o sectores o bloques de información de determinado tamaño separados por áreas de control, donde se pueden almacenar caracteres de sincronización, señales de referencia o la identificación de los datos. Estas áreas de control reducen la capacidad en bruto de almacenamiento de la cinta o disco a una capacidad neta de almacenamiento, con la ventaja de que se puede localizar con facilidad y confiabilidad la información que se busca. Este proceso de separar áreas de datos entre áreas de control se denomina formatear la superficie de almacenamiento.

La figura 4-48 muestra la forma como se debe formatear una pista de disco, en varios sectores, los sectores físicos se componen de la zona de datos y la zona de control respectiva. Todas las pistas tienen la misma organización de sectores con la diferencia que el número de pista, desde luego, es distinto. Lo mismo sucede con las superficies, donde todas tienen la misma organización de sectores excepto por el número de superficie.

4.4.2. CODIGOS DE GRABACION MAGNETICA

Existen numerosos métodos para almacenar información en superficies magnéticas, estos métodos son conocidos como códigos. Los objetivos de todo código, básicamente, se reducen a conseguir la mayor densidad posible manteniendo una alta confiabilidad durante el proceso de lectura. Las partes del proceso de lectura que son altamente dependientes del código son:

1. Requerimientos de sincronización del código, es decir,

4.4.2.3. CODIGO FM

El código Fm (frecuencia modulada) es una señal del tipo NRZI pero intercala entre los bits un pulso de reloj, por lo que este es el código más representativo de los que tienen reloj implicito.

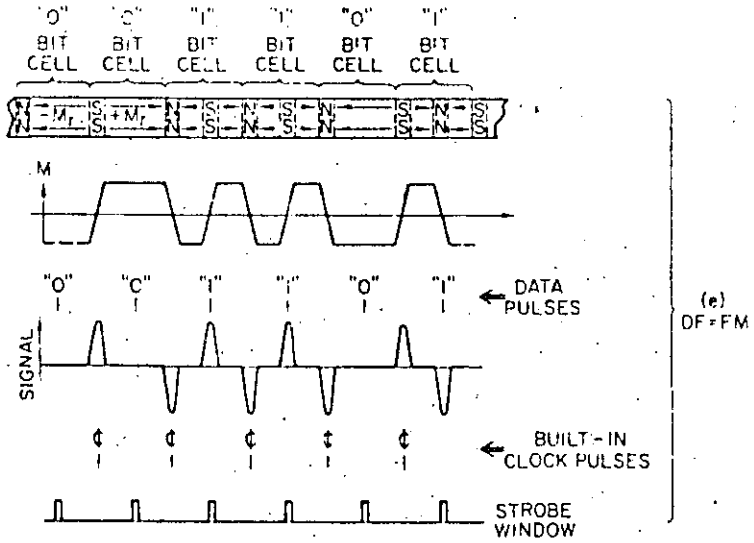


Figura 4-58: Código Fm, Frecuencia Modulada

Este código usan los discos flexibles de densidad sencilla, usa el mismo formato que NRZI con la diferencia que intercala un 1 entre datos. No tiene el problema de errores por tiempos acumulados, cada celda de bit tiene una o dos transiciones. El ancho de banda es conocido y menor que el NRZI, sin embargo, usa el doble de frecuencia que el NRZI para grabar la misma cantidad de datos.

4.4.2.4. CODIGO MFM O CODIGO MILLER

El código MFM (frecuencia modulada modificada) o conocido también como código Miller se deriva del código FM, más bien elimina los pulsos de reloj redundantes, por lo que resulta un código del doble de densidad.

Las únicas transiciones de reloj que realmente valen la pena del código FM son las que se encuentran entre dos ceros, por lo que estás son las únicas que se mantienen, se eliminan todas las otras. Este código dobla la densidad del FM si se mantiene la restricción de mínima distancia entre dos transiciones, sin

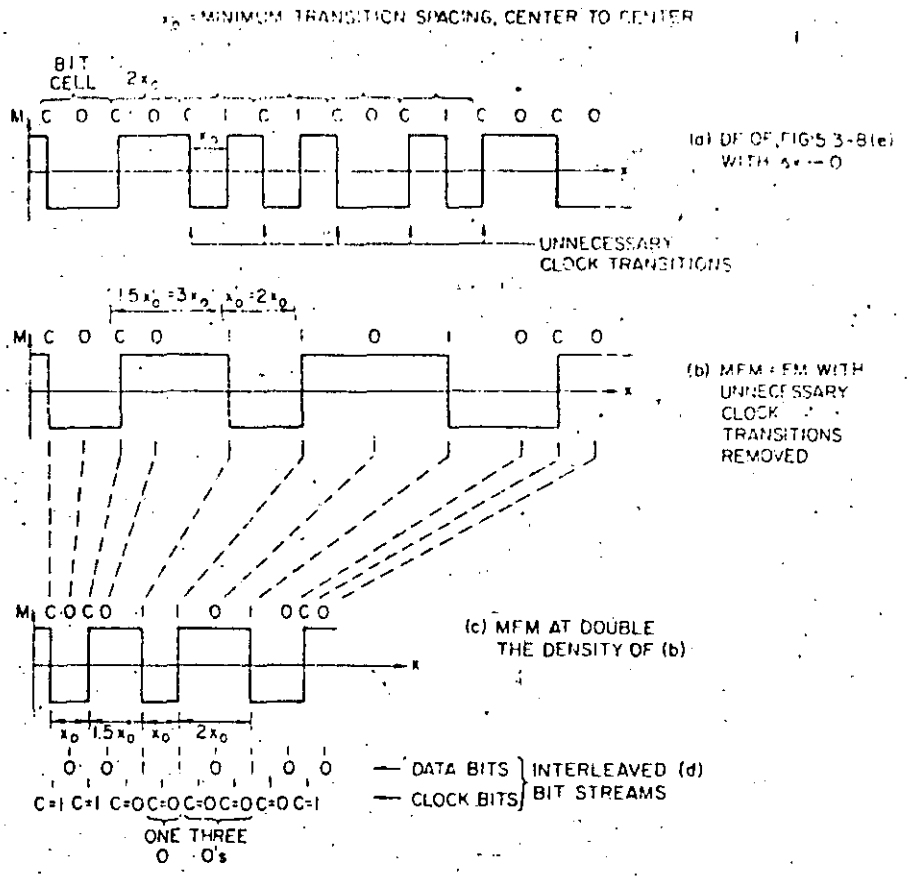


Figura 4-51: Código MFM o código Miller

embargo es mucho más compleja la separación de datos porque en este caso las distancias entre transiciones pueden ser 1x 1.5x y 2x, donde x es la distancia mínima entre dos transiciones. Este código es muy usado en los discos duros tipo Winchester, en los paquetes de discos y en los discos flexibles de doble densidad.

4.4.3. CINTAS TIPO CASSETTE

El medio más barato e inmediato para almacenar información es el cassette, pudiendose usar inclusive cassettes de audio comerciales para tal efecto. Por la calidad del cassette se los puede clasificar de la siguiente manera:

CASSETTE	PRECIO DOLARES	VELOCIDAD BITS/SEG	METODO GRABACION
De audio comun	1	150	FSK
De audio fino	5	1200	FSK, NRZI
Digital	7	9600	NRZI, PE
Limite mecanico		32000	

sectores por marcas grabadas por el mismo controlador del disco, esta estructura es muy flexible ya que se pueden reprogramar esas marcas a voluntad. Se puede programar en el formateo del disco el número de sectores por pista que se deseé.

- Por volumen, existen dos tipos de discos, los de altura estandar, y los de media altura, esto se presenta tanto en los discos de 8 pulgadas como en los de 5 1/4 pulg. Los discos de media altura, generalmente tienen las mismas características de grabación que los discos de altura estandar, sin embargo, ocupan la mitad de volumen, por lo que en el espacio que ocupa un disco de altura estandar pueden caber dos discos de media altura y la ventaja que se obtiene es que por el mismo volumen se logra el doble de densidad. Los discos de media altura son relativamente nuevos.

- Por el movimiento del disco, existen discos que están girando continuamente, mientras que hay otros que se detienen cuando no se usan, es decir, solo giran cuando van a ser utilizados para leer o escribir. Esta característica de discos detenibles la tienen los discos de 5 1/4 pulgadas, ya que usan motores de corriente directa alimentados con una fuente de 12 volts DC. Mientras que los de 8 pulg. usan motores de corriente alterna de 127 VAC.

4.4.4.1. TIEMPO DE ACCESO

El tiempo de acceso en los discos magnéticos esta determinado por varios factores que son:

t(acc) = t(seek) + t(pos) + t(laten) + t(arr)

donde los tiempos son para:

t(seek) mover la cabeza al track adecuado
t(pos) colocar la cabeza en el medio
t(laten) encontrar un dato dentro del track
t(arr) arrancar el motor

El t(arr) sirve solamente para los discos flexibles de 5 1/4 que son detenibles. El tiempo de acceso no es constante como en las memorias RAM o Rom, depende de la posición de la cabeza o del estado que se encuentre el disco, en el peor caso, hay que inicializar el motor, mover la cabeza de un extremo a otro, bajar la cabeza al disco y esperar toda una vuelta para localizar el

dato buscado, el mejor caso es que el motor este girando, la cabeza ubicada en el track adecuado y posicionada en el medio y el menor tiempo de busqueda secuencial "latency". Dentro de estos dos extremos puede haber un sin número de combinaciones.

4.4.4.2. VELOCIDAD DE TRANSFERENCIA

Los bits almacenados en los tracks están sincronizados a un reloj fijo, lo cual asegura que el número de bits por track es el mismo en todos los tracks, por lo tanto la velocidad de transferencia es también la misma en todos los tracks. Sin embargo, la longitud lineal de los tracks no es la misma, ya que los tracks más cortos son los que tienen el menor radio, por lo tanto la densidad lineal varia en forma proporcional a la longuitud de los tracks. Los tracks más internos (de menor radio) son los que tienen mayor densidad. El track cero es el track más externo y es el que tiene menor densidad. Hay dos formas en que los fabricantes especifican, generalmente, la densidad, dando la densidad máxima, o sea la densidad del track más interno o bien dando la densidad promedio, que es la que tiene el track del medio.

La velocidad de transferencia para cualquier track se calcula con la siguiente expresión:

$$V(t) = D \times w \times L$$

donde:

V(t)	Velocidad de transferencia (bits/seg)
D	Densidad del track (bpi) (bits/pulg)
w	Veloc. de rotac. del disco (rev/seg)
L	Longuitud lineal del track (pulg/rev)

4.4.4.3. DIAGRAMA DE BLOQUES

Los discos flexibles son equipos electromecánicos, una parte es mecánica que comprende la velocidad de rotación del disco y el mecanismo de movimiento de la cabeza, que puede ser un motor de pasos. La otra parte es electrónica que comprende la circuitería de lectura, escritura y el control tanto mecánico como electrónico de toda la unidad.

En la figura 4-53 se observan los tres bloques principales de la parte electrónica de la unidad y las señales que comunican con el controlador.

El circuito más delicado es el de lectura porque una parte es analógica y la otra es digital. La parte analógica es la que

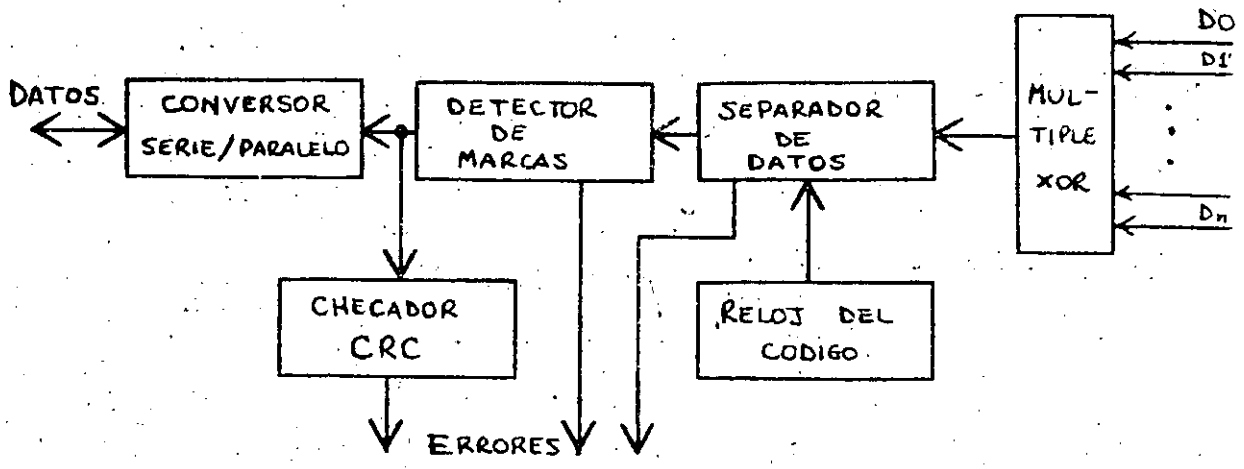


Figura 4-57: Lógica de lectura

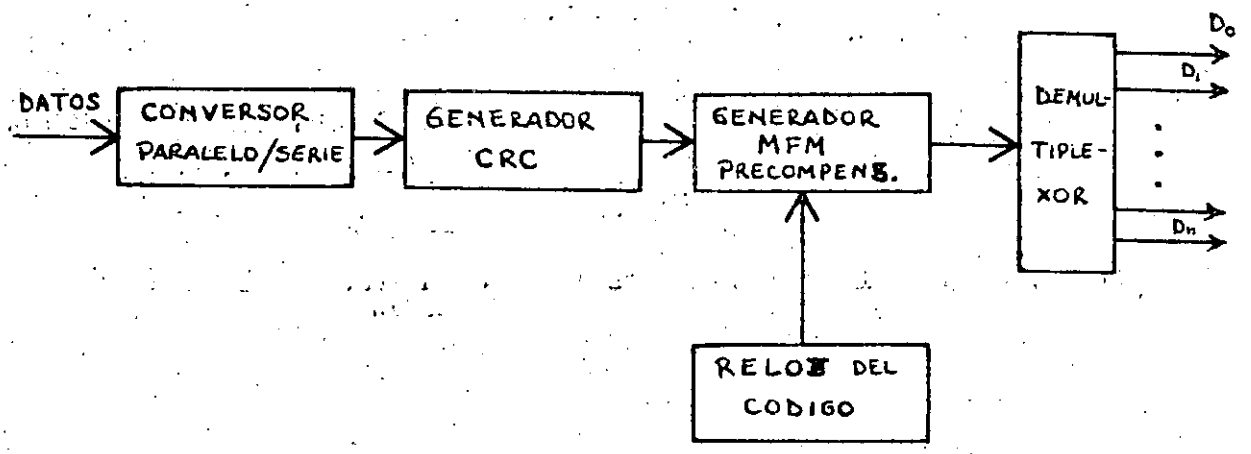


Figura 4-58: Lógica de escritura

4.4.6.2. FUNCIONES DE LOS CONTROLADORES DE DISCOS FLEXIBLES

Los controladores de discos flexibles se encuentran integrados en uno o dos chips, en algunos casos el bloque de separación de los datos se encuentra en un chip y el resto en otro, en cambio, los controladores más nuevos tienen todas las funciones integradas en un solo chip. Debido a la baja velocidad de transferencia que tienen los discos flexibles, gran parte de las rutinas las realiza el mismo procesador, por ejemplo, crear imagenes de los tracks para posteriormente formatearlos, leer y escribir byte a byte la información, etc. Estos controladores cargan con mucho trabajo al procesador, por lo que son usados en sistemas monousuarios.

Las funciones de estos controladores son las siguientes:

- Restore, regresar las cabezas al track cero.
- Seek, mover las cabezas de un track a otro.
- Lectura de uno o múltiples sectores.
- Escritura en uno o múltiples sectores.
- Lectura de indentificadores de sectores.
- Formateo del track, el formateo completo se realiza track por track.
- Lectura de un track completo incluyendo áreas de control.
- Verificación del track en el que se encuentra ubicada la cabeza.
- Aborto de comandos.
- Busqueda y localización de un sector de un track.
- Tamaño del sector programable.
- Generación y chequeo de un código ciclico de error (CRC).
- Manejo y control de errores.

4.4.6.3. FUNCIONES DEL CONTROLADOR DE DISCOS WINCHESTER

Los controladores de discos duros tipo wincnester tienen mayor inteligencia y realizan sus funciones sin intervención del CPU. Estos controladores, generalmente, tienen un procesador dedicado a manejar y controlar sus funciones. No existen chips que realicen todas las funciones del controlador de discos duros, los que existen comercialmente son tarjetas que se conectan al bus del procesador o a un adaptador especial.

La comunicación con el CPU se realiza a nivel de comandos y todas las funciones de bajo nivel (similares a las funciones del controlador de discos flexibles) las realiza en forma automática, asimismo tienen capacidad de recuperación automática en caso de errores no graves.

Las funciones más comunes de estos controladores son las siguientes:

- Seeks traslapados, es decir, tienen capacidad de efectuar seek en varios discos simultáneamente para ahorrar tiempo.

- Seek automático con verificación al leer o escribir un sector.

- Detección de fallas, bien sean del disco, del propio controlador o de la comunicación con el procesador central.

- Cambio automático de cabeza o cilindro en lecturas o escrituras de sectores múltiples al llegar al último sector.

- Corrección y sensado de errores en los datos.

- Espacio para almacenar "buffer" datos de un sector en el controlador.

- Independiza la velocidad de transferencia entre el disco y el CPU.

- Entrelazado de sectores para optimizar la velocidad de transferencia efectiva entre disco y memoria.

- Capacidad de efectuar diagnosticos, tanto del disco como del controlador, fuera de línea.

- Capacidad de DMA a través de un protocolo de hardware inteligente "handshake".

- Formateo automático sin intervención del procesador.

Estos controladores se usan en sistemas de mediana capacidad en los cuales se tiene varios procesos ejecutandose al mismo tiempo o bien varios usuarios.

4.5. TERMINALES DE VIDEO

Las terminales de video también son conocidas como CRTs (tubos de rayos catódicos) que si las unimos con un teclado pueden sustituir un teletipo (TTY), la única diferencia es que no tendríamos una copia en papel de nuestro trabajo. Para solucionar ésto, algunas terminales permiten que se les adapte un dispositivo de impresión.

Las terminales de video operan a velocidades mayores que los

teletipos por lo que tienen muchas aplicaciones como sería la graficación. La imagen en la terminal se forma al hacer que un haz de electrones choque contra una pantalla fluorescente produciendo un punto luminoso éste haz pasa por toda la pantalla por lo hace que sean vistas distintas imágenes en ella. Para ello se requiere de un "hardware" especial que lo haga.

Básicamente existen dos tipos de terminales de video que son las alfanuméricas y las gráficas. En las terminales de video alfanuméricas el haz de electrones recorre la pantalla 60 veces por segundo presentando conjuntos de puntos, cada uno de los cuales representa un caracter ASCII por lo que se necesita contar con una memoria que traduzca la señal enviada por la computadora en su respectiva representación con puntos. Las terminales de video gráficas son aquellas en las que el haz de electrones se ve en forma de líneas por lo que es más usado en aplicaciones como serían : diagramación, control en tiempo real, juegos, etc.. Cada punto en la pantalla necesita una memoria de un bit para ser guardado, ésto es, que cuando el haz de electrones pasa por un punto cualquiera de la pantalla, tiene que consultar la memoria para saber si se debe ver o no.

En la figura 4-59, se muestra un diagrama de bloques de una computadora con una terminal grática. Más adelante se explica su funcionamiento.El procesador de video se comunica con el resto de la computadora a traves de un controlador, de manera semejante a un dispositivo de entrada/salida. Los comandos, indicarán al procesador de video la forma en que debe dibujar la figura, éstos son ensamblados en la memoria principal por un programa del CPU. Este programa manda al controlador una señal de inicio, así como la dirección a partir de la cual, se encuentran del comandos de la figura, y lo hace por el bus de entrada/salida.

4.6. IMPRESORAS

Las impresoras obtienen sus reportes a velocidades mucho más altas que los teletipos (TTYś). Estas impresoras pueden escribir hasta 200 caracteres por línea a una velocidad hasta de unos cuantos miles de líneas por minuto. Esta velocidad depende de su construcción y del mecanismo de impresión que utilice y que puede ser entre otros, con un tambor o con una cadena.

Para analizar una impresora de tambor, consideremos una línea de "n" caracteres. El tambor será dividido entonces en "n" pistas, cada pista tiene un conjunto completo de caracteres. Para hacer la impresión, se tienen "n" martillos (uno por pista) que presionan el papel y la cinta con la tinta, contra el caracter. La línea que se va a imprimir, se encuentra en un "buffer" que es cargado por los circuitos de salida de la computadora. El tambor

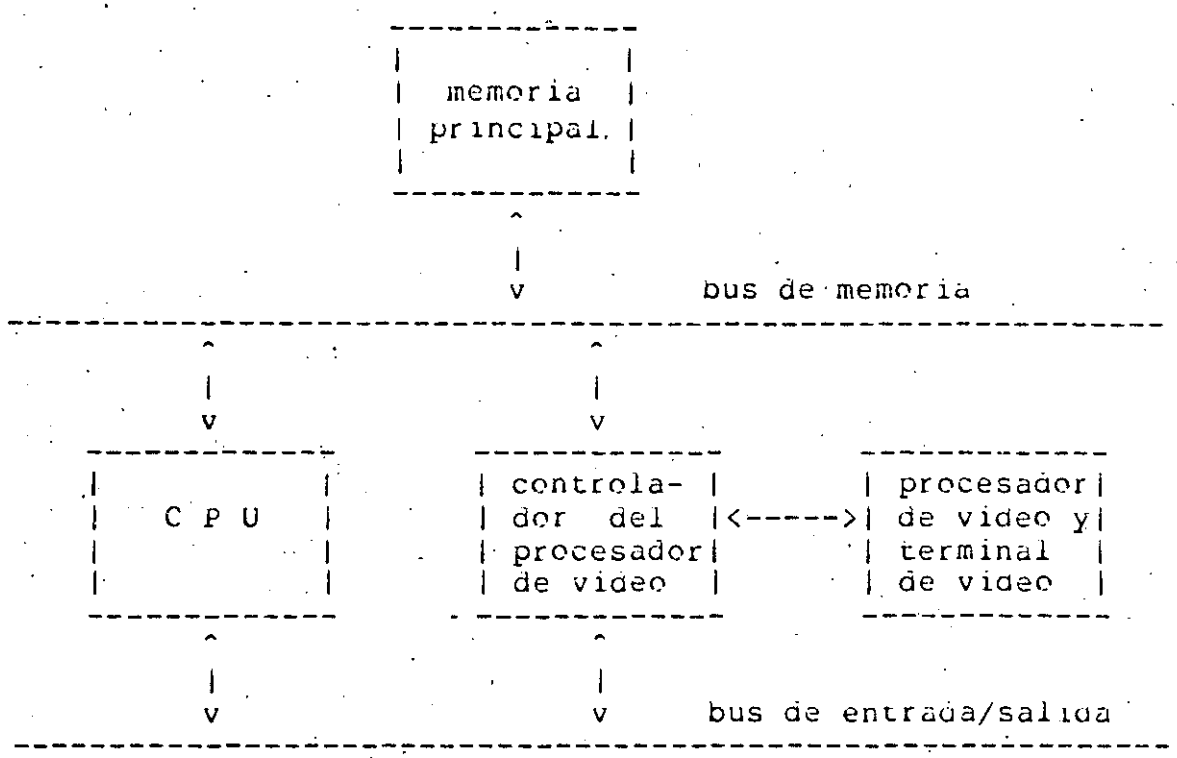


Figura 4-59: Terminal de video en una computadora.

gira y en el momento en que pasa frente al martillo el caracter que se encuentra en el buffer, éste se activa. Como se puede observar, cada línea se imprime con un solo giro el tambor.

Las impresoras de cadena emplean una cinta con un conjunto completo de caracteres. Esta cinta gira a lo largo de toda la línea por lo que, cuando pasa el caracter que se necesita frente a un martillo, éste se activa.

La velocidad de impresión de éstas impresoras mecánicas depende del movimiento de sus partes mecánicas por lo que puede variar de 1000 a 2000 líneas por minuto. Existen impresoras electrostáticas, en las que los caracteres son formados por el control de electrodos ya que el papel contiene material electricamente conductor. Estas impresoras pueden trabajar hasta con una velocidad de 10000 líneas por minuto.

4.7. EJEMPLOS DE MICROCOMPUTADORAS

Como se vió en el capitulo 3, las microcomputadoras, son commputadoras basadas en chips de microprocesadores, los cuáles, debidos al alto nivel de integración alcanzado se han "empaquetados" en un solo chip, junto con memorias RAM, ROM y PROM; así como relojes, interfases de memoria, controladores de interrupciones y puertos de E/S (por ejemplo el Intel 8048). El alto grado de integración de funciones ha dado como resultado que las microcomputadoras contengan menor número de circuitos integrados, liberando espacio en las tarjetas que componen a las microcomputadoras. Se ha utilizado éste espacio para proporcionar unidades funcionales adicionales como: interfases de E/S inteligentes, en particular los chips controladores de "floppys" o discos winchester; circuitos sofisticados para control de interrupciones, circuiteria para conexión con redes de computadoras, controladores de DMA, así como un constante incremento de capacidad en memoria. Con las innovaciones anteriores, las microcomputadoras han tomado caracteristicas de minicomputadoras, de la misma manera en que las "minis" lo empiezan a hacer con las computadoras grandes (mainframes).

Las microcomputadoras tienen una amplia variedad de aplicaciones, como puede ser en negocios, adquisición de datos, como terminales inteligentes de otras computadoras mayores, para control de procesos, en apliaciones científicas, educación, etc.

Existe una gran cantidad de firmas que producen microcomputadoras. Dentro de las más conocidas (en el país), son: Apple, Radio-Shack, Cromemco, Micron, Sundance, Onyx, Alpha-Micro, DEC-LSI-11, Newlett-Packard entre otras. Las cuáles, están configuradas alrededor de diferentes microprocesadores, y capacidades (8, 16 y algunos de 32 bits). Debido a la gran popularidad que ha alcanzado la microcomputadora Apple, se mostrarán algunas de las caracteristicas importantes que presenta esta computadora:

Microcomputadora APPLE II+

-CONSTRUCCION: Una tarjeta con 63 chips, fuente de poder, consola de control y gabinete.

-MICROPROCESADOR: Un 6502 de Synertek, de 8 bits por palabra, con interrupciones del tipo "poleadas" de un solo nivel.

-MEMORIA RAM: Capacidad para 48K bytes, con palabras de 8 bits y ciclo de acceso de 350 ns.

-MEMORIA ROM: Capaciadad de 12K bytes, con tamaño de 8 bits por palabra y ciclo de acceso de 400 ns.

-CONTROL DE E/S:

Tamaõ de la palabra de E/S de 8 bits, con 8 canales (slots) para expansión. Con capaciaad para DMA. Interiase para monitor y para cassette.

-SOFTWARE: Ensamblador, Desensamblador, Monitor, Basic, Pascal, CPM y otros.

-APLICACIONES: Para negocios, como terminal inteligente, para control de procesos, aplicaciones científicas, en educación y recreación.

-OTRAS: Capacidad para graficas a color en alta resolución, unidades de control para juegos, audio. Interfases para impresora, disco floppy y línea RS232. Tiene fuentes de poder para memoria RAM y tarjetas de E/S. Teclado ASCII.

Con las caracteristicas anteriores la microcomputadora, puede ser expandida con diferente equipo como puede ser discos floppys, winchesters, modems, tarjetas de comunicaciones, tabletas digitalizadoras, graficadoras, impresoras, terminales de video entre otros.

CAPITULO 5
LOS LENGUAJES ENSAMBLADORES

5.1. INTRODUCCION

5.1.1. EL SIGNIFICADO DE LAS INSTRUCCIONES.

El conjunto de instrucciones de un microprocesador es simplemente el conjunto de valores binarios que al ser leidos por el microprocesador producen acciones perfectamente definidas durante el ciclo de instrucción.

Una instrucción es, sencillamente, un patrón binario de bits. El cual debe estar disponible en las patas del chip, por las que se transmiten o reciben datos, en el momento preciso, para que pueda ser interpretado como instrucción.

5.1.2. ?QUE ES UN PROGRAMA DE COMPUTADORA?

Un programa es una serie de instrucciones que hacen que una computadora realice cierta tarea en especial.

Cada programa queda cargado en la memoria como un conjunto de números binarios .

Este conjunto de números es el programa en lenguaje de máquina o programa objeto.

5.1.3. VEAMOS EL PROBLEMA DE LA PROGRAMACION

Si tratamos de crear programas en código objeto o lenguaje de máquina vamos a tener que salvar muchas dificultades tales como:

1. Los programas son muy difíciles de entender.
2. Si no es fácil entederlos, entonces su depuración es aun más difícil.
3. El proceso de cargar los programas dentro de la memoria de la computadora es realmente lento, dado que hay que cargar bit por bit.
4. Dicho programa no da ninguna descripción (en alguna forma entendible para cualquier persona) acerca de la tarea que deseamos que la computadora realice.

5. Como tenemos que escribir los programas en números binarios, entonces resultan muy...muy largos y lentos de escribir.

6. Por la razón anterior el escribir los programas resulta una tarea tediosa , aburrida y cansada.

7. Al estar cansado, el programador, entonces es fácil que cometa errores, los cuales son muy difíciles de encontrar en un programa escrito en numeración binaria.

5.1.4. ES MEJOR USAR NUMERACION OCTAL O HEXADECIMAL

Podemos mejorar la situación usando numeración octal o hexadecimal, en vez de la binaria, para escribir los programas.

Pero ahora, ?qué hacemos con un programa escrito en hexadecimal si el micro sólo entiende instrucciones escritas en código binario?

Pues ya escrito en números hexadecimales, ahora hay que convertirlos a binarios y luego cargar el programa en binario a la computadora. Sin embargo, esto aún es muy cansado y sujeto a los errores que cometa el individuo. ?Qué podemos hacer?

La respuesta es: hay que escribir un programa que tome los números en hexadecimal y los convierta en binarios. A este programa lo llamaremos Cargador Hexadecimal.

Pero ahora tenemos que darnos cuenta de que el Cargador Hexadecimal es un programa que debe estar en memoria y por lo tanto ocupa espacio en ella. Este inconveniente debe atrontarse, ya que es mayor el tiempo y esfuerzo que el programador anorra usándolo.

Sin embargo persisten varios de los 7 inconvenientes antes mencionados para los programas escritos en código objeto. !Entonces hagamos algo!.

5.1.5. USEMOS MNEMONICOS PARA LOS CODIGOS DE LAS INSTRUCCIONES

Podemos mejorar la situación al dar nombre a cada código de instrucción. Dicho nombre es llamado mnemónico y debe dar idea de que es lo que hace la instrucción.

Al conjunto de mnemónicos le llamamos Lenguaje de Ensamble o Lenguaje Ensamblador. Y a un programa escrito en este lenguaje le llamamos Programa en Lenguaje Ensamblador.

Para cargar dicho programa en la computadora tenemos que traducirlo a mano a su programa en código de máquina. Esto presenta nuevamente los 7 inconvenientes antes marcados.

5.1.6. EL PROGRAMA ENSAMBLADOR

El hacer tal traducción es una tarea engorrosa la cual es perfecta para que el micro se encargue de ella. Ya que el micro nunca comete errores al traducir códigos; también sabe cuantas palabras necesita para cada instrucción y qué formato tiene cada una.

El programa que hace ese trabajo es llamado Ensamblador. El programa ensamblador traduce el programa del usuario o Programa Fuente, escrito usando los mnemónicos, en un Programa en Lenguaje de Máquina o Programa Objeto.

Este programa usa mucha más memoria y requiere de más tiempo de execución y de más periféricos que el Cargador Hexadecimal, pero hace muy fácil la tarea de escribir programas.

Sin embargo los ensambladores tienen que ser usados bajo sus propias reglas del juego, las cuales tenemos que aprender para poder trabajar con ellos.

Tales reglas del juego las encontramos en el manual de usuario del ensamblador. Algunas de ellas son: el uso de ciertos delimitadores en lugares determinados, la ortografía correcta del lenguaje ensamblador, cierta información de control y en algunos casos hay que sujetarse a poner nombres y números en lugares especícos.

5.1.7. VENTAJAS ADICIONALES DE LOS ENSAMBLADORES

1. Permiten que el usuario asigne nombres a localidades de memoria, a los mecanismos de E/S y a secuencias de instrucciones.

2. Tienen la capacidad para convertir datos o direcciones de diversos sistemas numéricos, tales como el octal , hexadecimal o decimal, a valores binarios.

3. Tienen que realizar algunas funciones aritméticas como parte del proceso de ensamblado.

4. Dicen al programa cargador en que partes de memoria deben de ponerse los datos o el programa.

5. Permiten que el usuario asigne areas de memoria para usarlas como almacenamiento temporal de datos.

6. Proveen toda la información necesaria para incluir Programas de Biblioteca, o programas escritos en alguna otra ocasión , dentro del programa actual del ususario.

7. Y permiten al usuario controlar el formato del listado del programa así como los mecanismos de entrada y salida.

5.1.8. ?CUALES SON LAS DESVENTAJAS DE USAR LENGUAJE ENSAMBLADOR?

Ni el cargador hexadecimal ni aun el programa ensamblador, pueden resolver todos los problemas derivados de programar. Ya que, todavía existe gran distancia entre lo que permite hacer el conjunto de instrucciones de la computadora y los problemas que desea resolver el programador con esa máquina.

Además si se está programando en lenguaje ensamblador, el programador debe conocer detalladamente la computadora que está usando.

Otra desventaja es que los programas en ensamblador no son transportables ya que cada micro tiene su propio conjunto de instrucciones.

5.1.9. LOS LENGUAJES DE ALTO NIVEL

Es por lo anterior que se usan lenguajes de alto nivel, los cuales serán motivo de otros cursos.

5.1.10. FACTORES QUE DETERMINAN EL USO DEL LENGUAJE ENSAMBLADOR

Se puede usar cuando el tamaño de los programas va de pequeño a mediano.

En aplicaciones cuando el costo de memoria es importante.

En control de tiempo real.

Cuando se hace más control o E/S que cálculo

5.2. LOS PROGRAMAS ENSAMBLADORES

5.2.1. QUE HACEN LOS PROGRAMAS ENSAMBLADORES?

La función principal de los ensambladores es traducir los mnemónicos del lenguaje ensamblador a sus correspondientes códigos binarios. Veamos ahora como hacen esta tarea.

Las instrucciones de un programa en lenguaje ensamblador se escriben de acuerdo a un formato especial que generalmente está dividido en cuatro campos, los cuales se pueden apreciar en la Figura 5-1.

```
+-------------------------------------------------------------+
! campo de !  código de  ! operando o !                       !
!   la    ! operación o ! campo de la ! campo de comentarios !
! etiqueta !  mnemónico  ! dirección  !                       !
+-------------------------------------------------------------+
 EJEMPLO:      LD    _       HL,(EJEMPLO) ; CARGA EN HL LA DIR.
                                          ; DE ESTA INSTRUCCION
```

Figura 5-1: ejemplos de los campos

De estos cuatro campos el del código de operación es el único que siempre debe estar presente, y debe de tener o el mnemónico de una instrucción o una directiva para el ensamblador. también llamadas pseudo-instrucción o pseudo-operación (las cuales se discutirán más adelante).

El campo de operación puede tener una dirección, un dato o estar en blanco.

Los campos de cometarios y de la etiqueta pueden o no ser usados, pero facilitan la tarea del programador cuando trata de identificar las secciones de su programa, o cuando quiere poner alguna explicación breve.

??Como sabe entonces el ensamblador donde comienza y acaba cada campo?. La mayoría de los ensambladores usan algún símbolo o caracter especial al principio o final de cada campo, los cuales son conocidos como Delimitatores.

Los delimitadores más usados se muestran en la figura 5-2

DELIMITADOR	EXPLICACION DE SU USO
:	DESPUES DE UNA ETIQUETA
ESPACIO EN BLANCO	ENTRE CODIGO DE OPERACION Y DIRECCION
,	ENTRE LOS OPERANDOS
;	ANTES DE UN COMENTARIO

Figura 5-2: Los Delimitadores y su uso

5.2.2. LAS ETIQUETAS

El campo de la etiqueta es el primer campo de una instrucción en lenguaje ensamblador. Cuando el ensamblador encuentra una etiqueta en este campo, lo que hace es asignarle a la etiqueta el valor de la localidad de memoria en donde se encuentra el primer byte de la presente instrucción.

Cualquier etiqueta puede ser utilizada en el campo de operandos, como datos o como dirección, de cualquier instrucción. Cuando el ensamblador la encuetre, lo que hace es reemplazar la etiqueta por el valor que se le ha asignado.

El uso de las etiquetas presta especial ayuda para realizar saltos a otras partes del programa o para hacer llamadas a subrutinas. También para el uso de bloques contiguos de memoria, los cuales frecuentemente son usados como buffers o arreglos, y se necesita tener la referencia de donde empiezan para poder utilizar cada una de las palabras que los forman.

5.2.2.1. ??QUE FACILIDADES NOS DA EL USO DE ETIQUETAS?

1. Una etiqueta permite que una localidad sea fácil de encontrar y de recordar.

2. Si la etiqueta fué escrita en alguna linea equivocada, podemos moverla a su lugar correcto sin tener que hacer ningún cambio a las instrucciones que la usan, ya que el ensamblador se encarga de esa tarea.

3. El ensamblador o el cargador, pueden relocalizar todo el programa a cualquier lugar de memoria sumando, simplemente, una constante a cada dirección en la que se haya usado una etiqueta.

4. Hace que cualquier programa sea mucho más fácil de entender que uno en código objeto, y por lo tanto más personas pueden hacer uso de él en sus programas.

5. No tenemos que calcular las direcciones de memoria, tarea que se hace especialmente pesada si los códigos objeto de las instrucciones son de longitud variable.

5.2.2.2. ??QUE REGLAS HAY PARA EL USO DE LAS ETIQUETAS?

1. Use etiquetas que den idea del propósito del programa en esa parte.

2. El tamaño máximo va a ser de 5 ó 6 caracteres, empezando por una letra.

3. Se recomienda que no se usen etiquetas idénticas a algunos de los mnemónicos que sean propios de alguna instrucción o pseudo-operaciones del ensamblador.

4. Evite el uso de caracteres especiales y de letras minusculas.

5. Evite también el uso de los números 1, 2 y 0 así como de las letras I O Y Z.

5.2.3. LOS MNEMONICOS O CODIGOS DE OPERACION DEL ENSAMBLADOR

La principal tarea del ensamblador es la traducción de los mnemónicos a sus valores binarios. Sin embargo también debe determinar cuantos operandos requiere una instrucción y de que tipo deben ser.

5.2.4. LAS PSEUDO-OPERACIONES

Hay algunas instrucciones del ensamblador que no son traducidas a alguna de las instrucciones de máquina, sino que son órdenes o directivas al ensamblador y permiten asignar el programa completo a algún area en especial de la memoria, definir símbolos, designar areas de memoria para almacenamiento temporal de datos, designar la localización de tablas o datos en memoria y permitir la referencia a otros programas así como algunas funciones de control.

5.2.4.1. VEAMOS LAS PSEUDO-OPERACIONES MAS COMUNES

1. DATA op1<,op2,...opN> permite al programador almacenar datos fijos en la memoria, los cuales pueden ser numéricos y alfanuméricos. Otros nombres que recibe el DATA son DEFINE BYTE para manejar números de 8 bits y DEFINE WORD para manejar números de 16 bits.

2. etq EQUate op1 asigna el valor numérico en su campo de operandos a la etiqueta que se encuentra en su campo de etiqueta. Es importante resaltar que la pseudo-operación EQU no hace que el ensamblador asigne ningún lugar de memoria, sino que sólamente se inserta en la tabla de símbolos el nombre y el valor de la etiqueta.

3. ORG op1. El ensamblador maneja un contador de localidades que contiene la localidad en memoria donde será guardado el siguiente byte del código objeto que sea generado por el ensamblador. Con el ORG hacemos que el ensamblador cambie el contador de localidades por el valor del operando, con lo cual estamos dirigiendo al ensamblador para que produzca código objeto que será cargado y corrido específicamente en esa localidad de memoria.

   El ORG se usa principalmente para conocer la dirección de inicio, en programas principales, para conocer las direcciones de servicio para interrupciones, en subrrutinas, direcciones de memoria que han sido usadas para mecanismos de E/S, para manejar almacenamiento en la memoria, etc.

4. etq RESERV op1, o etq DEFSpace op1 esta pseudo-op permite asignar nombre a un area de memoria y declarar el tamaño de la misma.

5.2.4.2. PSEUDO-PERACIONES PARA LIGADO DE PROGRAMAS

Cuando queremos usar tablas, variables o rutinas declaradas en algún otro programa necesitamos hacer una referencia EXTerna. Con lo anterior dejamos ciertas declaraciones de la tabla de símbolos sin su valor, el cual será definido hasta el momento de ligado del programa.

Si existe EXT, tiene que existir ENTry el cual avisa al ensamblador que esta declaración debe estar disponible para que sea usada en otros programas. También es conocido como GLOBAL.

El ensamblador al encontrar estos pseudo-operadores, lo que hace es dejar un archivo con la información necesaria para que el programa ligador pueda solucionar todas las referencias externas al momento del ligado de los diferentes módulos.

5.2.4.3. PSEUDO-OPERACIONES DE CONTROL

Este tipo de pseudo operaciones sirve para controlar o afectar la operación del ensamblador y sus archivos de salida,pero no al programa objeto.

1. END marca el punto final del programa fuente de ensamblador.
2. LIST indica al ensamblador que imprima el listado del programa fuente,también se puede solicitar impresiones parciales, por ejemplo sólo la tabla de símbolos con LIST SYMBOL TABLE.
3. NAME o TITLE que indican que nobre se debe imprimir al principio de cada hoja del listado.
4. PAGE o SPACE ordenan el salto a la siguiente página o linea del listado.

5.2.4.4. EL USO DE LAS ETIQUETAS EN LAS PSEUDO-OPERACIONES

1. Las PSEUDO-OPERACIONES DEFB, DEFW, DEFS, o DATA y RESERVE generalmente usan etiquetas, con el fin de identificar la primera localidad de memoria asignada a las mismas.
2. Todas las PSEUDO-OPERACIONES EQUate deben usar etiquetas, ya que precisamente su propósito es el de definir significado a la etiqueta que va en su campo de etq.

5.2.5. EL CAMPO DEL OPERANDO O DE LA DIRECCION

En éste campo, es donde se tiene que describir el operando o la dirección de la instrucción, ahora bien, ?cuales son las formas en que el programa ensamblador nos permite definir estos valores?

5.2.5.1. VALORES NUMERICOS

1. Descripción por medio de números decimales. Cada vez que el programa ensamblador encuentra un número en este campo, assume que está en base decimal a menos que se especifique otra base. La forma en que se pueden usar otras bases numéricas es usando caracteres especiales antes o despues del número que se desea representar

2. Si son números en base binaria, se usan los caracteres B o % para identificarlos.

3. Para números en octal se usan los caracteres O, Q, C o @.

4. Si se trata de números en base hexadecimal, se usan los caracteres $ o H. Es importante que los números escritos en esta base tengan su primer dígito entre el 0 y el 9. Lo anterior es necesario debido a que sabemos que el ensamblador nos permite construir etiquetas que comienzan con letras únicamente, y si el primer dígito del hexadecimal es una letra, el ensamblador lo tratará como un identificador o etiqueta, lo cual provocará errores al momento de ensamblar.

5.2.5.2. ETIQUETAS O NOMBRES SIMBOLICOS

Se pueden poner etiquetas en el campo de operando, pero representarán realmente a los datos o direcciones para los que fueron definidos.

5.2.5.3. REFERENCIA POR EL CONTADOR DE LOCALIDADES

En vez de etiquetas se pueden hacer referencias a localidades de memoria a través del uso del location counter, su uso frecuentemente lo lleva a uno a generar errores involuntarios que se pueden evitar con el uso de las etiquetas.

5.2.5.4. CADENAS DE CARACTES

Se pueden meter en éste campo de operando cadenas de caracteres utilizando las siguientes convenciones:

1. Poner la cadena entre comilla o dobles comillas
2. Poner la cadena antecediendola o precediendola de algún caracter especial tal como A o C.

Generalmente las cadenas se manejan en código ASCII.

5.2.5.5. COMBINACIONES DE LAS FORMAS ANTERIORES USANDO LOS OPERADORES ESPECIALES ARITMETICOS Y LOGICOS

Generalmente los ensambladores permiten el uso de combinaciones aritméticas simples tales como etiq+10. Algunos también permiten realizar multiplicaciones, divisiones, funciones lógicas, etc. Las cuales son llamadas expresiones. Es importante tener en cuenta que el ensamblador evalúa las expresiones en el momento del ensamblado, para no incurrir en el frecuente error de creer que cuando el programa se esté ejecutando, se volverán a evaluar tales expresiones cada vez que el programa pase por esa instrucción.

5.2.5.6. ENSAMBLADO CONDICIONAL

Es frecuente que los ensambladores tengan PSEUDO-OPERACIONES que permitan incluir o no, porciones de programas fuentes, dependiendo del estado de ciertas banderas en el momento de ensamblado. Esta pseudo operación se usa generalmente para:

1. Crear versiones especializadas de programas que fueron escritos contemplando diferentes comportamientos del mismo programa ante diversos perifericos.
2. Incluir o excluir variables adicionales.
3. Incluir porciones de programa que solo sirven para hacer diagnósticos o para probar el comportamiento, durante las corridas de prueba de algún programa en especial.

5.2.6. EL CAMPO DE COMENTARIOS

Todos los ensambladores permiten al usuario poner comentarios en el programa fuente. El único uso de éstos es el auxiliar al programador para entender y documentar sus programas. Al documentar los programas no se está perdiendo el tiempo, sino que se está previendo que en unas horas, o tal vez algunos días, el programador olvidará lo que hace cada sección de su programa , y por tanto si hubo algún error ayuda a que sea fácil corregirlo.

Se hacen las siguientes recomendaciones para el uso de los comentarios:

1. El comentario debe decir que es lo que hace el programa en ese punto, no que es lo que hace la instrucción de ese renglón.

2. Los comentarios deben de ser breves y concisos. Los detalles deben ponerse en la documentación aparte del programa.

3. Hay que poner especial atención en documentar instrucciones que no tengan significado obvio. Por ejemplo, no hay que documentar instrucciones o secuencias de ellas que actualizen contadores o apuntadores.

4. No hay que usar abreviaciones, para que los comentarios sean perfectamente entendibles.

5. Se deben comentar todas las definiciones, tales como: macros, data, defw, defb, etc.describiendo su propósito.

6. Hay que usar la misma terminología a lo largo del programa.

7. Es bueno el dejar notas para uno mismo en los puntos que sean confusos, tales como, RECUERDA QUE EL CARRY FUE MODIFICADO POR LA LLAMADA A LA SUBRUTINA.

5.3. DEFINICION DE MACROS

Los macros le permiten al programador asignar un nombre o etiqueta a alguna secuencia de instrucciones, con lo cual si esta secuencia se repite frecuentemente a lo largo de todo el programa, el programador ya no tendrá que escribir la secuencia completa, sino sólo el nombre del macro. También hay que hacer notar que se pueden definir parámetros que serán pasados al momento de hacer la llamada del macro.

El ensamblador se encargará de reemplazar el nombre del macro por la secuencia de instrucciones. Esto significa que en cada llamada que se hace al macro realmente se está poniendo todo el código, esto lo diferencía de una subrutina, puesto que el código de ésta sólo aparece una vez en todo el programa, y para que se ejecute se realizan saltos a la rutina.

5.3.1. VENTAJAS DE LOS MACROS

1. Los programas quedan más cortos, y por lo tanto se pueden documentar mejor.
2. Ya se está usando una secuencia de instrucciones depurada.
3. Es más fácil hacer cambios, ya que al hacer el cambio en la definición del macro, el ensamblador hace el mismo cambio cada vez que se usa el macro.
4. Se pude usar el macro para extender el conjunto de instrucciones al declarar nuevas funciones que serán entendidas por el programa.
5. también se pueden redefinir instrucciones ya existentes.

5.3.2. DESVENTAJAS DE LOS MACROS

1. Repetición del mismo código en cada llamada del macro ya que estas son expandidas.
2. Un sólo macro puede estar formado por gran cantidad de instrucciones, y ésto se ve reflejado en el tamaño del programa.
3. Posibles efectos laterales en los registros o banderas que no pueden apreciarse claramente.

4. Finalmente hay que hacer notar que las variables que se usan en los macros son únicamete locales a éstos, así que no podremos conocer su valor fuera del macro.

5.4. TIPOS DE ENSAMBLADORES

A pesar de que todos los ensambladores tienen que ralizar las mismas tareas, (ver el principio de este capítulo), la forma como son escritos, y las técnicas que usan son muy diferentes, debido a lo anterior, a continuación sólo se nará la descripción de algunas de sus principales características.

5.4.1. ENSAMBLADORES DE CODIGO CRUZADO

Un ensamblador de código cruzado, es aquel que aunque se ejecuta en determinada computadora, produce código objeto que sólo se puede ejecutar en otra máquina diferente de la primera.

5.4.2. ENSAMBLADORES DE CODIGO PROPIO

Obviamente el ensamblador de código propio, es aquel que corre en la misma computadora para la cual está generando el código, y que se ejecutará en ella misma.

5.4.3. MACROENSAMBLADORES

Este tipo de ensambladores poseen la característica especial de que permiten al usuario el definir secuencias de instrucciones a través de macros, tal como se explicó previamente.

5.4.4. ENSAMBLADORES DE UNA PASADA

Se les llama así a aquellos ensambladores que sólo nacen una lectura del programa fuente para generar el código objeto.

Este tipo de ensambladores no pueden utilizar referencias a etiquetas, si estas no han sido previamente definidas en el programa.

5.4.5. ENSAMBLADORES DE UNA Y MEDIA PASADA

Debido a que el uso de las etiquetas es de gran ayuda, entonces se pensó en tener ensambladores de una pasada pero que si encontraba una etiqueta que no nabia sido definida aún, lo que nacía era guardar la información de la localidad donde se encontró la etiqueta no definida, para que en forma posterior al ya tener definido el valor de la etiqueta, poder regresaar a esos lugarea que nabian sido dejados en blanco y poner ahí el valor correspondiente a la etiqueta.

5.4.6. ENSAMBLADORES DE DOS PASADAS

El ensamblador de dos pasadas es aquel que lee dos veces el programa fuente. Durante la primera lectura lo único que nace es recolectar la información correspondiente a las definiciones de las etiquetas e identificadores , es decir, asignar los valores de todas las etiquetas usadas en el programa. En la segunda pasada el ensamblador ya reemplaza los códigos objeto de todas las instrucciones, pues ya no existe ningún problema respecto a etiquetas o identificadores no definidos.

5.5. CARGADORES

Los cargadores son los programas que permiten depositar en la memoria de la computadora el código objeto generado por el ensamblador y a continuación darle el control al programa para iniciar su ejecución.

5.5.1. CARGADOR RELOCALIZANTE

El cargador relocalizante es aquel puede cargar los programas objeto dejados por el ensamblador en cualquier parte de la memoria siempre y cuando el programa mismo sea relocalizable.

5.5.2. CARGADOR ABSOLUTO

A diferencia del anterior este cargdor siempre cargará el programa en la misma area.

5.5.3. CARGADOR-LIGADOR

Este tipo de cargador carga diferentes módulos o programas que han sido ensamblados por separado, resolviendo las referencias externas (ver PSEUDO-OPERACIONES para ligado).

5.6. LIGADORES

Se pueden separar las funciones de ligado y cargado de programas teniendo que realizar entonces el ligado de los diversos módulos antes de poder cargarlos a memoria para su ejecución.

La función de los ligadores es entonces el tomar uno o varios módulos o programas objeto creados por el enamblador o por los compiladores de lenguajes de alto nivél, produciendo un sólo modulo o programa que ya puede ser ejecutado. Para hacer lo anterior, el ligador tiene que permitir la relocalización de los módulos y resolver todas las referencias intermodulares de forma que se puedan ligar módulos ensamblados por separado.

A continuación se mencionan las principales características de los ligadores que normalmente se tienen en una microcomputadora.

1. Crea un archivo con el código objeto listo para ser corrido.

2. Permite ligar módulos ya sea que estos sean relocalizables o no.

3. También permite asignar, opcionalmete, orígenes absolutos a algunos o todos los módulos que se estan ligando

4. Checa que no haya globales con definiciones múltiples y que no quede sin resolver ningún identificador declarado como externo.

5. Crea un mapa de las globales y la dirección con que quedaron asignadas finalmente.

CAPITULO 6
EDITORES

6.1. INTRODUCCION

Los editores interactivos se han convertido en un componente esencial de cualquier lugar en el que se usen máquinas computadoras. Estos programas usan el poder de la computadora para la creación, adición, borrado y modificación de textos tales como instrucciones de programas, texto manuscrito y datos números. Un editor permite que un texto sea modificado y corregido más rápido y más fácilmente , en muchos ordenes de magnitud, de lo que sería si se tuviese que hacer la corrección manualmente.

A pesar de que los editores siempre han sido herramientas importantes en los sistemas de computación, ha sido hasta ahora que han empezado a convertirse en tema de investigación, conforme se empiezan a convertir en componenetes claves de las oficinas del futuro. Actualmente los editores no son vistos ya como herramientas de uso exclusivo de los programadores, o para secretarias que transcriben lo que les pasa en borrador de papel el autor. Ahora se esta comprendiendo cada vez más que el editor debe ser considerado la interfaz primaria entre la computadora y todo aquel trabajador cuyo quenacer intelectual involucre la composición, organización, estudio, y manipulación de información basada en computadora.

6.2. PANORAMA GENERAL

6.2.1. EL PROCESO DE EDICION

Un editor es un programa de computadora que permite al usuario crear y modificar un documento en forma interactiva.

Editar, entonces, se podrá definir como un diálogo interactivo entre la computadora y el usuario. Este diálogo consiste básicamente en lo siguiente :

1. Selección de la parte del documento que será vista y manipulada.

   - Viajar a través del documento y filtrarlo para controlar lo que se podrá ver y manipular.

2. Determinar el formato con que se presentará el documento en línea y mostrarlo.

   - La presentación deberá ser la misma que se imprima en papel.

3. Especificar y ejecutar operaciones que modifiquen el documento.

   - El proceso de edición. Es decir , el conjunto de operaciones que crean o modifican el documento.

4. Actualizar la presentación adecuadamente.

   - Esto forma parte del proceso de edición.

6.2.2. EL EDITOR DESDE EL PUNTO DE VISTA DEL USUARIO

1. Lo que se le presenta a el usuario es lo que llamaremos **modelo conceptual del sistema**, es la estructura en la cual el editor y el "mundo" en el que opera están basados, y sus principales características deberán ser :

   - Que esté bien articulado.
   - Que provea de una estructura consistente y completa para usar e implenetar el sistema.

2. Interfaz del usuario. Es el conjunto de herramientas y técnicas con las que el usuario se comunica con el editor y consiste básicamente de :

   - Dispositivos de entrada.
   - Dispositivos de salida.
   - Lenguaje interactivo.

3. Documentación adicional. Es deseable que contenga :

- Descripción del modelo conceptual.

- Descripción de la arquitectura del sistema, con terminología a nivel de usuario.

- Una guía de el usuario detallando la sintaxis y semántica del lenguaje de interacción.

- Un tutorial que posea definiciones operacionales y que demuestre situaciones típicas con ejemplos.

6.2.3. EL USUARIO DESDE EL PUNTO DE VISTA DEL EDITOR

Cada individuo forma un **modelo de usuario** personal de un editor y y este modelo puede diferir de el modelo conceptual de usuario en varias formas.

1. Como subconjunto de el modelo conceptual de usuario.

   - Cuando el usuario solo usa un subconjunto de instrucciones del editor.

2. Como una extensión del modelo conceptual de usuario.

   - Cuando el usuario hace, en forma consistente, uso de instrucciones "primitivas" para realizar operaciones comunes en formas que no fueron originalmente abarcadas por el modelo conceptual.

3. Como un equivalente operacional pero lógicamente diferente del modelo conceptual de usuario.

   - Cuando el usuario tiene un modelo conceptual, del sistema, operante pero que sin embargo difiere del modelo conceptual en el que se basó el diseñador del editor.

   - **ES IMPORTANTE NOTAR AQUI QUE EL USUARIO HACE USO DEL EDITOR EN BASE A SU PROPIO MODELO CONCEPTUAL DEL SISTEMA.**

6.2.4. EL EDITOR DESDE EL PUNTO DE VISTA DEL SISTEMA

Los editores en general siguen una arquitectura similar a la que se muestra en la figura 6-1

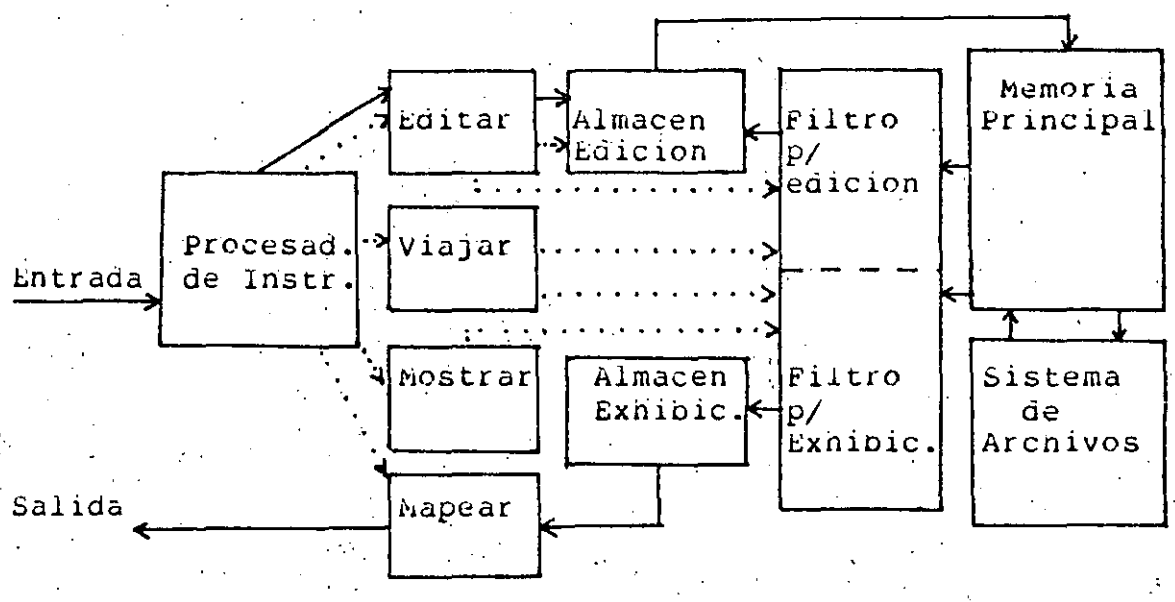


Figura 6-1: Arquitectura general de un editor

1. Procesador de instrucciones.

   - Acepta datos cuya fuente son los dispositivos de entrada de usuario.

   - Analiza el léxico y convierte la cadena de entrada en descriptores (separador , operador, identificador, etc.).

   - Analiza sintácticamente el conjunto de descriptores y si encuentra una composición válida de descriptores, invoca a las rutinas semánticas apropiadas.

   - Las rutinas semánticas invocan a las rutinas que permiten editar, viajar, mostrar y mapear en algún dispositivo de salida el documento.

   Mientras que las operaciones de edición son siempre especificadas explícitamente por el usuario, y las operaciones de mapeo lo son e

forma implícita por las otras tres categorías de operaciones, las operaciones de viajar y mostrar pueden ser explícitamente especificadas o implícitamente invocadas por las operaciones de edición. De hecho la relación entre las tres clases de operaciones puede ser considerablemente más complicada que el simple modelo de la sección 1. (viajar para determinar el LUGAR que se seleccionará , filtrar para seleccionar QUE es lo que se va a mostrar y manipular, formatear para determinar COMO aparecerá el mapa del documento en los dispositivos de salida, y después editar y reformatear).

En particular, no es necesario qe haya una simple relación uno a uno entre lo que se está mostrando en los dispositivos de salida y lo que se puede editar.

Para ilustrar lo anterior observaremos más de cerca los componentes restantes de la figura 1 los cuales pretenden ser entidades conceptuales.

2. Editar

Se encarga de determinar el área que puede ser editada, usando , lo que llamaremos, un apuntador para edición.

Este apuntador puede ser explícita o implícitamente posicionado por el usuario o por el sistema, respectivamente, como resultado de las instrucciones de edición.

3. Viajar

Posiciona los apuntadores siguientes:

El de edición.

El de exhibición.

Por lo tanto este componente marca el inicio de el punto en que el filtrado del documento empieza: filtrado para edición y filtrado para exhibición.

4. Mostrar

Se encarga de determinar el área que puede ser mostrada, usando , lo que llamaremos, un apuntador para exhibición .

Este apuntador puede ser explícita o implícitamente posicionado por el usuario o por el sistema, respectivamente, como resultado de las instrucciones de edición.

5. **Mapear**

Se toma lo que hay en el almacen de exhibición y se mapea uno-a-uno a el dispositivo de salida.

6. **Almacen de edicion**

Es aquí donde se tiene una copia de la parte del documento que se está actualizando.

7. **Almacen para exhibicion**

Aquí se tiene almacenada la copia del documento actualizada que se mostrará al usuario.

8. **Filtro para la edicion**

Este filtro es invocado siempre que el usuario teclea una instrucción. Y lo que hace es poner una parte de el documento, una copia, en el almacen de edición, teniendo como referencia los apuntadores de edición y sus propios parámetros. (Estos parámetros son especificados tanto por el usuario como por el sistema. Y dan información como: Que rango que puede ser afectado por el usuario, es decir, la línea actual en un editor de línea y la pantalla actual en un editor de pantalla.)

9. **Filtro para exhibicion del documento**

Este filtro es invocado siempre que lo que se esta mostrando al ususario necesite ser actualizado.

Este filtro se encarga pues, de filtrar una copia del documento, y pasarla al almacen del exhibición, en función del apuntador actual de exhibición y de sus propios parámetros.

10. **Memoria principal y Sistema de Archivos**

Generalmente se almacena aquí una copia del documento. Sin embargo cuando esto es imposible o no deseable, entonces se mapea el archivo completo en memoria virtual y se deja que el sistema operativo realice de paginación para el editor. Las cuales se almacenarán en memoria principal hasta que una operación del usuario requiera otra página.

En cualquier caso los documetos son frecuentemente representados no como cadenas secuenciales de caracteres, sino como una " estructura de datos para editor " , cuya característica distintiva es permitir la adición, borrado y modificación de caracteres minimizando el uso de las rutinas de entrada/salida asi como el movimiento de caracteres.

6.2.4.1. CONFIGURACIONES

Los editores funcionan en tres tipos básicos de ambientes de computación a saber : Tiempo compartido, dedicado (personal) , y Distribuido.

1. Tiempo compartido. Este tipo de editor debe funcionar adecuadamente en el contexto de la carga impuesta al procesador de la computadora, a la memoria principal y a la memoria secundaria.

2. Dedicado (pesonal). En este caso el editor debe tener acceso a las funciones que él editor de un sistema de tiempo compartido obtiene de el sistema operativo de la computadora anfitrión.

   Estas funciones pueden ser obtenidas en parte de un peguño sistema operativo local o pueden estar implementadas en el mismo editor si el sistema solo se usa para edición.

3. Distribuido. El editor en este tipo de sistemas debe, como en el caso de un editor para un sistema personal, correr en forma independiente en cada una de las máquinas que conforman la red, y además debe competir, como en el caso de un editor de un sistema de tiempo compartido, por recursos tales como archivos.

6.3. DESARROLLO DE LOS EDITORES

Consiste básicamente de un panorama general en el cual se revisan algunos conceptos importantes.

6.3.1. EDICION COMPUTARIZADA NO INTERACTIVA

La unidad básica fué la línea de 80 columnas; el ususario nacía correcccciones línea por línea, reescribiendo las tarjetas erroneas.

6.3.2. EDICION DE TARJETAS

Aquí el conjunto inicial de tarjertas de los usuarios se almacenaba en un archivo imagen, ya sea en cinta o disco magnético. Cada tarjeta se referenciaba por un número de secuencia único. Los cambios se nacían creando un conjunto de tarjetas de edición compuesto de trajetas que tenían las instrucciones de edición y se corría el conjunto de tarjetas usando un programa editor que corría fuera de línea. Los editores de este tipo, que corrian fuera de línea, eliminaron los problemas de tarjetas que resultaban de desecho asi como el de tener que reescribir, y en algunas versiones permitieron operaciones como las de nacer reemplazos globales de un patron dado. Sin embargo, nabía ciertos incovenientes. por ejemplo los programadores debían tener un listado del conjunto de tarjetas completo antes de tratar de nacer algún cambio. Y algunas de las ventajas organizacionales de las tarjetas se perdierón tales como la inspección visual sencilla de las siguientes características:

- Secuencia Adecuada.
- Código de colores.
- Etiquetado adecuado de cajas de tarjetas.

6.3.3. EDITORES DE LINEA INTERACTIVOS

Se diseñaron a mediados de los 60s, con el advenimiento de los sistemas de tiempo compartido.

Las características que vale la pena remarcar en estos editores son las siguientes:

- Permitian crear y modificar arcnivos desde una terminal.
- Las líneas de estos archivos eran de longitud fija, inicialmente de 80 caracteres.
- Muchos de estos editores permitian nacer correcciones, usando mucno de la sintaxis que caracteriza a sus

antecesores. Era típico que muchos de estos editores compartieran la infortunada propiedad de TRUNCAR: si una inserción o un cambio forzaba a la línea a exceder su longitud máxima, ! los caracteres se tiraban por el fin de la línea según se fuese necesitando !. Esta "característica" de implementación, parte de un modelo conceptual de el proceso de edición basado en la simulación de tarjetas perforadas y listados de impresoras de línea,lo cual fué solo marginalmente aceptable para la edición de programas y completamente inaceptable para la creación de manuscritos en serio. (En éste último caso, la creación automática de una línea en el caso de rebosamiento de caracteres hubiese sido una problema cuya solución es realmente trivial).

6.3.4. EDITORES DE LINEA BASADOS EN CONTEXTO

Otro avance que se tuvo en los editores fué la posibilidad de identificar una línea que contenia el "objetivo" de una operación dada, sin tener que especificar el número de línea. Es decir, el poder identificar una linea dado un patrón, un caracter, que el editor tenía que encontrar.

En esta parte de la historia de los editores el usuario erá aún forzado a pensar en términos de entidades de líneas múltiples, tales como parrafos y bloques de programas, usualmente con la imagen de el formato de las tarjetas; no había instrucciones tipo interlínea que pudieran, por ejemplo, borrar texto que fuese e la mitad de una línea a la mitad de otra.

6.3.5. EDITORES DE LINEA DE LONGITUD VARIABLE

Este fué el primer "rompimiento" con los editores de "tarjetas de 80 columnas" . Aún asi el elemento primario de edición fué la línea pero ahora la línea podía ser de longitud "arbitraria" (generalmente límitada a un máximo de, digamos 500 caracteres).

Es importante remarcar que el hecho de haber eliminado la restricción de tener una imagen de una tarjeta al estar editando, dió como resultado un impacto fuerte y benéfico en la versátilidad de el procesamiento de texto.

Otro importante hecho, mucho tiempo después comprendido, es el texto que se está mostrando NO tiene que se una mapa uno-a-uno de la representación interna, sino que puede ser una visión más abstracta y adecuada de los elementos editables.

Sin embargo aún se tenían problemas en lo que a edición de manuscritos se refiere, básicamente :

- Truncado de la línea cuando esta pasaba de el límite impuesto.
- Innabilidad para editar una cadena de caracteres que crusará el límite entre entre dos líneas.
- Innabilidad para buscar una cadena de caracteres que cruce el límite entre dos líneas.

6.3.6. EDITORES DE LINEA "INFINITA"

Estos editores resolvieron los problemas básicos de los editores de línea de longitud variable, al eliminar por completo los límites entre líneas. El texto entero fué considerado como una línea completa la cual es convertida en líneas visibles por las rutinas que muestran la información.

6.3.7. EDITORES DE PANTALLA

Se llaman tambien editores de cursor. Y permiten otra forma de eliminar las limitaciones de los editores de línea/superlínea usando el poder de las pantallas para mostrar varias líneas a la vez. (lo cual permite direccionar el cursor y la posibilidad de tener almacenes locales, de uso transitorio, para edición).

Estos editores pueden trabajar con líneas de longitud variable o "infinita", ofreciendo al usuario una pantalla llena de texto que puede ser editada sin tener que pensar en términos de una línea de longitud fija.

6.3.8. DESARROLLOS RELEVANTES EN EL CAMPO

6.3.8.1. NLS

NLS (oNLine System), es un desarrollo importante debido a que el editor se ve como una HERRAMIENTA PARA EL AUTOR, es decir, un medio interactivo de organizar y revisar información y no como una herramienta mundana para alterar caracteres en un solo archivo. (Fué desarrollado a principios de 1959 en el instituto de investigaciones de Stanford en Estados Unidos de Norteamerica por Douglas Engelbart).

6.3.8.2. EDITOR DIRIGIDO POR SINTAXIS

Los usuarios pueden manipular construcciones lógicas tales como ciclos iterativos y lo que esta anidado en ellos como una sola unidad.

6.3.8.3. DESARROLLO DE UTILERIAS PARA EDITORES

Este es un desarrollo importante que se origino a pricipios de los 70s y básicamente consiste en darle tanto peso a las utilerias para edición como a las utilerias para programación.

Ejemplos de utilerias son: formateadores de texto, de ecuaciones, de tablas, de bases de datos bibliográficas, asi como correctores de ortográfia y de estilo.

6.3.8.4. EDITORES/FORMATEADORES

El archivo del usuario de despliega en una pantalla usando mapeadores de bits en un facsímile con la tipografía y diseño del documento final.

6.4. CONCLUSIONES

Más que ser esto un conjunto de conclusiones terminantes, pretende sugerir un conjunto de criterios para el diseño de un editor "ideal".

- **Modelo Conceptual.** Debe estar bien definido y ser consistente. El usuario debe estar familiarizado y agusto con la filosofía que hay detrás del sistema.

- **Documentacion.** Es importante que haya documentación en línea, una instrucción del editor que sea: AYUDA por ejemplo. y documentación fuera de línea como manuales. Este tipo de documentación debe explicar el modelo conceptual asi como los detalles de la interfaz con el usuario y las funciones del sistema.

- **Interfaz con el Usuario.** Debe ser clara y concisa, fácil de aprender y usar y además debe ser consistente a través de diferentes tipos de documentos tales como: texto, gráficas y voz. De hecho, una buena forma para probar si una interfaz es eficiente y agradable es que los autores usaran el sistema para componer y revisar los manuscritos por ellos mismos. (No deberán

necesitar de expertos en el uso del sistema para que le ayuden, o de secretarias para que hagan cambios, en ninguna fase de la creación o edición del documento).

- **Hacer/Arrepentirse.** Una capacidad infinita para hacer y arrepentirse le permite al autor el **experimentar** sin tener que preocuparse por la pérdida o daño a un documento.

- **Facilidades.** Fácilidades poderosas , es decir con pocas restricciones y excepciones, que permitan al usuario hacer todo lo que se puede hacer con textos de papel con lápiz rojo, calculadora, tijeras y cinta adesiva. Además se debe tomar ventaja de las capacidades de la computadora para compensar las límitaciones humanas. Por ejemplo: Sustituir un patrón dado por otro en forma global a través de un documento; replicación de una frase de uso corriente, de un parrafo, y renumeración automática de secciones o referencias después o mientras el archivo es editado.

- **Acceso a informacion compartida.** Que el usuario pueda accesar información y archivos compartidos bajo situaciones controladas.

- **Mezclar documentos.** Debe tenerse la facilidad para mezclar diferentes documentos tales como texto, gráficas, programas y formas con facilidad.

- **Multiple Contexto.** Múltiple Contexto en la misma superficie de exhibición, permitiendo al usuario el revisar y usar una gran cantidad de utilerias familiares y documentos en una sesión de edición. El editor no debe forzar al usuario a un medio ambiente pequeño y menos poderoso sino que este debe formar parte de un medio ambiente mayor e integrado, permitiendo al usuario, en la mitad de una sesión de edición el obtener información mirando a través del sistema de archivos, el usar una utileria que emule una calculadora u obtener un mensaje de correo eléctronico o una pieza de datos de un sistema de base de datos con regreso transparente a la sesión de edición.

- **Mostrar el documento en su version final.** La habilidad de editar un facsímil muy parecido a la composición , a la disposición de texto, y a la tipografía final del documento sin un impacto significante en el tiempo de respuesta de la computadora.

CAPITULO 7
INTERPRETES

7.1. CARACTERISTICAS

Un lenguaje intérprete, es un programa que acepta como entrada un programa en código fuente y lo ejecuta directamente; a diferencia de un compilador que también acepta como entrada un programa fuente, pero que produce código que posteriormente será ejecutado por un procesador. Esto no es del todo cierto, ya que generalmente los intérpretes emplean dos fases : duraante la primera se traduce el código fuente a un lenguaje intermedio ó forma interna; durante la segunda se enjecutan las acciones representadas por ese código intermedio. Un intérprete de código hilvanado produce una forma interna totalmente analizada , que consiste en una lista de direcciones de formas internas previamente definidas; ésta lista se obtiene durante la primera fase , la cual se llama modo compilación. Durante la ejecución el intérprete ejecuta formas internas consecutivamente, sin llevar a cabo ningún tipo de análisis ó búsquedas, puesto que ya se llevaron a cabo durante la primera etapa.

7.2. BASIC

7.2.1. CARACTERISTICAS PROPIAS

BASIC es un lenguaje de programación muy empleado por las microcomputadoras y muy sencillo de aprender. Aunque existen compiladores de BASIC, la forma más comúnmente empleada para correr programas escritos en éste lenguaje, es utilizar un intérprete de BASIC. El lenguaje es de propósito general y se utiliza tanto para problemas relacionados con soluciones de sistemas de ecuaciones, como para sistemas de información pequeños (inventarios, nóminas, etc.). Su principal desventaja es que no es estructurado, ya que tiene GOTOS, lo que impide hacer programas verdaderamente legibles. Sus ventajas incluyen: simplicidad de instrucciones y entrada/salida relativamente sencilla.

7.2.2. INSTRUCCIONES

El conjunto de instrucciones de BASIC es relativamente pequeño y muy simple, además de ser muy sencillo de utilizar; existen funciones de entrada/salida, de control de flujo, aritméticas, de manejo de cadenas de caracteres y matemáticas.

7.2.2.1. INSTRUCCIONES PARA MANEJAR ARREGLOS

DIM

Reserva localidades de memoria para una variable de tipo arreglo. Por ejemplo, DIM A(20,20), reserva memoria para alojar a una matriz de 20x20 elementos y cuyo nombre es A.

7.2.2.2. FUNCIONES PARA MANEJAR CADENAS DE CARACTERES

- LEN

  Regresa el número de caracteres de una cadena de caracteres.

- STR$

  Convierte una expresión aritmética en una cadena de caracteres que representa ese valor.

- VAL

  Interpreta una cadena de caracteres, como un número real ó entero, regresando el valor de ése numéro.

- CHR$

  Es una función que regresa el caracter ASCII, que corresponde a una expresión aritmética.

- ASC

  Regresa el código ASCII del primer caracter de una cadena de caracteres.

- LEFT$

  Regresa los primeros n caracteres de la izquierda, de una cadena de caracteres.

- RIGHT$

Regresa los n últimos caracteres de la derecha, de una cadena de caracteres.

- MID$

Regresa una subcadena de n caracteres, a partir del i-ésimo caracter de una cadena de caracteres.

7.2.2.3. INSTRUCCIONES DE ENTRADA/SALIDA

Las instrucciones de entrada/salida son aquellas que permiten introducir y obtener datos de la computadora antes, durante y después de la ejecución,

- INPUT

Está instrucción cuyos parámetros son: una sola cadena de caracteres ó una lista de variables de cualquier tipo, excepto cadenas ; permite leer del teclado los datos necesarios y por cada variable que necesita leer, despliega en la pantalla un caracter (p. ej. ?' ó #), significando que requiere un dato a entrar por el teclado.

- DATA

Esta proposición crea una lista de elementos que posteriormente podrán ser usados por la instrucción READ. Dichos elementos pueden ser de cualquier tipo. Cada DATA encontrado, agrega sus elementos a los ya existentes, por la aparición de previos DATAś ; éstas declaraciones pueden aparecer en cualquier lugar dentro del programa.

- READ

Esta instrucción lee datos de la lista conformada por todos los DATA previamente declarados. A cada READ, corresponde un elemento previamente declarado en DATA.

- RESTORE

Esta instrucción, tan sólo mueve el apuntador a la lista de datos, al principio de la lista.

- PRINT

Imprime en la pantalla el ó los valores declarados después de la misma instrucción, que en general puede ser una expresión.

7.2.2.4. FUNCIONES MATEMATICAS

Adicionalmente, BASIC tiene un conjunto de funciones matemáticas, que se enumeran a continuación:

1. SIN (seno)
2. COS (coseno)
3. TAN (tangente)
4. ATN (función inversa de la tangente)
5. INT (entero más grande, menor ó igual)
6. RND (número aleatorio mayor ó igual a 0 y menor que 1)
7. SGN (signo del número)
8. ABS (valor absoluto)
9. SQR (raíz cuadrada)
10. EXP (eleva e (2.718289), a la potencia indicada)
11. LOG (obtiene el logaritmo natural (base e))

7.2.3. MANEJO DE DATOS

Los tipos de datos que BASIC maneja son :

- números enteros,
- números de punto flotante (reales)
- arreglos
- y cadenas de caracteres.

7.2.3.1. ENTEROS

- Los números enteros se encuentran en el rango de -32767 a 32767.
- Las variables enteras se forman con el nombre al que se le agrega al final un signo de "%".

- Un número entero ocupa 2 bytes de memoria.

7.2.3.2. REALES

- Los números reales se encuentran en el rango de -1E38 a 1E38.
- Un número real ocupa 5 bytes de memoria.

7.2.3.3. ARREGLOS

Un arreglo es una tabla de números, cuyo nombre es cualquier nombre de variable válido; para seleccionar uno de los elementos de la tabla, se utiliza un subíndice. El subíndice se pone enseguida del nombre de la variable encerrado entre parentésis redondos. Generalmente los subíndices comienzan a partir del valor 0, aunque ocasionalmente algunos intérpretes generan subíndices a partir del valor 1. Por ejemplo, el tercer elemento del arreglo cuyo nombre es ARREGLO, está denotado por la expresión ARREGLO(2). Aunque las posibilidades de dimensionar variables de tipo arreglo son inmensas, en la realidad la cantidad de espacio que se puede reservar es función de la cantidad de memoria disponible. Esto es debido a que como se dijo líneas atrás ,cada elemento entero de un arreglo ocupa 2 bytes y cada elemento real utiliza 5 bytes en memoria.

7.2.3.4. CADENAS DE CARACTERES

Las cadenas de caracteres ocupan 3 bytes más de la longitud de las mismas, 1 byte es para la longitud y 2 para un apuntador a la cadena. Los nombres de variables tipo cadena, se forman agregando al final el símbolo de "$". Es posible llevar a cabo operaciones lógicas con las cadenas; éso significa que se pueden comparar dos cadenas y dar como resultado una variable lógica, además existe la concatenación, adicionalmente de las operaciones mencionadas en secciones atrás.

7.2.4. CONTROL DE FLUJO DE PROGRAMA

Las instrucciones de control de flujo, son aquellas que permiten romper la ejecución secuencial del programa y dependiendo de alguna condición, pasar el control del programa a otra aréa del mismo.

- GOTO

Salta ó transfiere el control del flujo del programa a la línea que se encuentra enseguida de "GOTO".

- IF.... THEN....

Si a expresión aritmética que se encuentra entre IF y THEN, tiene un valor 1 (se considera que es verdadero), entonces se ejecuta lo que está a continuación de THEN, lo cual puede ser: una instrucción, un número de línea ó un GOTO; en caso contrario, se ejecuta la instrucción que se encuentra en la siguiente línea numerada. Por ejemplo:

```
100 IF A+B = 2 THEN A=1

100 IF A>B GOTO 200

100 IF A<B THEN 200
```

son instrucciones válidas.

- FOR....NEXT

Esta instrucción sirve para generar secuencias repetitivas de instrucciones que se ejecutan un número determinado de veces, hasta que el límite inferior adquiere el valor ó es mayor que el límite superior. El incremento del índice puede fijarse arbitrariamente por medio de la proposición STEP; si ésta no existe, se sobreentiende un incremento unitario del índice. Como ejemplo, la siguiente función obtiene los números impares menores que 20.

```
100 FOR I=1 TO 20 STEP 2

200 PRINT I

300 NEXT
```

Pueden utilizarse FOR....NEXT , unos dentro de otros, (anidamiento); sólo debe cuidarse de que no queden traslapados unos con otros, ya que entonces el flujo del programa no podrá ser controlado y producirá resultados erróneos. El límite de anidamiento (profundidad) es variable y depende de los diferentes sistemas en los cuales se ejecutan programas.

- GOSUB

  Cuando se encuentra ésta instrucción, el control del programa se pasa a la línea cuyo número se encuentra después de GOSUB. El primer RETURN que es encontrado, retorna el control del programa a la instrucción que está inmediatamente abajo de GOSUB.

- RETURN

  Es una instrucción que no tiene parámetros, es un salto a la instrucción que sigue al más reciente GOSUB encontrado .

- ON....GOTO, ON....GOSUB

  Dependiendo del valor de la expresión aritmética que se encuentra entre ON y GOTO ó GOSUB, se transfiere el control del programa a la línea que se encuentra en la posición que corresponde a dicho valor.

7.2.5. SUBRUTINAS

Como a se dijo en la sección anterior, las llamadas a subrutinas se hacen por medio de la instrucción GOSUB; ó condicionalmente por medio de ON..GOSUB. Como las subrutinas se llaman por número de línea, es complicado tratar de hacer programas en una sola pasada, pues no se sabe que cantidad de código habrá de ponerse entre la llamada a la subrutina y el principio de ella. Es aconsejable, utilizar una numeración de tal forma, que si se requiere poner alguna instrucción intermedia, se pueda intercalar entre 2 existentes; por ningún motivo, deberá utilizarse una numeración en la cual los incrementos sean unitarios, pues raramente los programas están bien escritos a la primera vez. Sin embargo, el uso de subrutinas es apropiado y aconsejable cuando se tienen programas largos, y se prestan para emplear muchas rutinas pequeñas. En éste momento es válido aclarar que la longitud máxima de un programa hecho en BASIC es en la mayoría de los sistemas de 10000 líneas, ya que sólo se tienen 4 dgitos para enumerar las líneas, sin embargo, p.ej. el de APPLE, tiene capacidad para direccionar hasta 64000 líneas de programa.

7.2.6. ASPECTOS DE IMPLEMENTACION

En el IIMAS-UNAM, se desarrolló un intérprete de BASIC, escrito totalmente en FORTH (se describe en la siguiente sección), el cual a su vez és también intérprete. Se desarrollo a pedido del grupo ALFA del CCH SUR, para poder implementar sus programas de enseñanza de mátematicas auxiliada por computadora. Una de las características que debería tener éste intérprete, era ser totalmente compatible con el BASIC de las microcomputadoras APPLE. Esto se debe a que todos los programas ya existentes, los cuales eran bastantes por cierto, estaban escritos para correrse en las APPLE. Por lo tanto, es propiamente un intérprete de BASIC de APPLE. Esto significa que cualquier programa escrito para una APPLE, correrá en éste intérprete sin ningún cambio en absoluto. La desventaja principal de correr un intérprete escrito en otro intérprete se observa en el momento de correr programas, especialmente si éstos son grandes: la respuesta es más lenta que la de un intérprete normal.Otra desventaja es la cantidad de memoria que emplea el intérprete: casi 32K bytes; ésto significa casi la mitad de la memoria disponible, lo que deja tan sólo 32K bytes para programas. Esto en la práctica no es problema, pues la aplicación para la cual fué diseñado, requiere de módulos de enseñanza que no ocupan más de la mitad del espacio disponible para programas.

7.2.7. EDITOR INTEGRADO

Una de las características que han hecho muy popular a BASIC, aparte de ser un lenguaje de programación muy simple y fácil de usar, es el hecho de poder contar con un editor integrado al intérprete. El tener editor integrado, significa que el usuario puede:

- crear programas
- modificar alguna de las líneas
- modificar la numeración de alguna de las líneas
- listar el programa creado
- correr el programa

sin necesidad de :

1. llamar a un editor

2. editar el programa

3. salir del editor

4. correr el programa y si tiene errores, regresar al paso 1

como se ve, existe un anorro de tiempo bastante considerable. Este es digno de tomarse en cuenta, sobre todo cuando se quieren correr programas pequeños. Una de las posibles desventajas de los editores integrados, si no se tiene un mecanismo adecuado para almacenar los programas creados, en un medio de almacenamiento masivo, es que los programas se pierden; de modo que cuando el individuo desea correr el mismo programa que hizo antes (suponiendo que alguien ya usó el mismo sistema, ó que fué apagado), tiene que teclearlo de nuevo.

7.3. FORTH

7.3.1. CARACTERISTICAS

FORTH es un lenguaje de programación que tiene ciertas características, que lo hacen preferible en varias aplicacio nes, a lenguajes como BASIC o PASCAL. En general es más preferible que BASIC, porque a pesar de que también es conversacional; y al igual que BASIC permite ejecutar porciones de código y variables sin necesidad de utilizar un editor, el código de FORTH es estructurado y carece de la proposición GOTO; siendo por lo tanto de más alto nivel que aquél. Una de sus ventajas es que, a pesar de que no tiene chequeo de pa- rámetros como PASCAL, es más rico en tipos de datos y el compilador es varias veces más rápido.

El programador de FORTH interactúa con el sistema a través de un intérprete de notación polaca postfija y evalúa las expresiones empleadas utilizando una pila (stack). El intérprete examina las secuencias de instrucciones de izquierda a derecha; las cuales se van introduciendo por medio del buffer de entrada, y si

1. encuentra un número lo mete al stack,

2. encuentra una función la ejecuta.

7.3.2. TIPOS DE DATOS

Los tipos de datos que maneja FORTH son :

- NUMEROS ENTEROS

Los números enteros son convertidos a binario por el intérprete externo, de acuerdo a la base en la cual se trabaja; aunque se puede trabajar en cualquier base, las más usadas son : binario (2), octal (8), decimal (10) y hexadecimal (16). Es muy usual estar cambiando de una base a otra, sobre todo de decimal a hexadecimal y viceversa. El rango para los números depende del microprocesador y para números tratados internamente como de 16 bits es:

números signados -32768<= n <=32767

números no signados 0<= n <=65535

Como también existen valores de tipo bytes, su rango es:

números signados -128<= n <=127

números no signados 0<= n <=255

- BANDERAS LOGICAS

Una bandera lógica es un parámetro con 2 posibles estados Verdadero ó Falso; y de acuerdo a la convención, un 1 es Verdadero y un 0 es Falso. Algunas veces alguna constante ó variable puede ser tomada como una bandera lógica; en éste caso, cualquier valor distinto de cero será considerado como Verdadero.

- CADENAS DE CARACTERES

El manejo de cadenas de caracteres es posible hacerlo mediante ciertos operadores y algunos más que pueden implementarse de acuerdo a las necesidades y deseos del programador. Como se almacena tanto la longitud de la cadena, como todos los caracteres, la manipulación de las cadenas se lleva a cabo de un modo bastante simple. Una de las aplicaciones más comunes, es para desplegar mensajes en la pantalla.

- CONSTANTES

CONSTANT es una función que crea funciones, las cuale al ser llamadas regresan el valor que se les asignó inicialmente; la sintaxis empleada es la siguiente: valor CONSTANT nombre. Por ejemplo, si se crea la función constante CUATRO de la siguienter manera:

```
4 CONSTANT CUATRO
```

al llamar a la función CUATRO, el intérprete regresará a la pila de datos el valor 4.

- VARIABLES

VARIABLE tiene la misma sintaxis que CONSTANT, sólo que, a diferencia de ésta, crea funciones que al ser llamadas regresan la dirección de un valor, el cual puede ser cambiado en cualquier momento, no así el de las constantes. La forma de poder tener acceso a esos valores y poderlos modificar, es por medio de las funciones @, c@, ! y c!. Las 2 primeras sirven para traer el valor de variables de tipo palabra (16 bits) y tipo byte respectivamente a la pila de datos; mientras que las segundas sirven para asignar nuevos valores a las variables cuya dirección se encuentra en la pila, de tipo palabra y byte, respectivamente. las constantes y variables de tipo byte, se crean con las funciones CCONSTANT y CVARIABLE. Los tipos de datos de FORTH, tienen asociada una función a la que se llama prólogo; el prológo determina que se debe hacer con el código que se encuentra enseguida. las funciones que tienen el mismo prológo reciben el nombre de "instancias" de un tipo de dato.

- ARREGLOS

Los arreglos son funciones pasivas que reservan aréa del diccionario que tiene un nombre asociado. Se pueden definir operadores ó funciones que crean arreglos de tipo byte, o de tipo palabra, etc; aún más, es posible, de acuerdo con la flexibilidad que exhibe FORTH, de crear funciones que crean estructuras más complicadas, tales como RECORDS,SETS,etc.. Los elementos de tales estructuras se direccionan relativamente al primer elemento del arreglo; es decir, como se conoce siempre la dirección del primer elemento, basta sumarle el índice ó desplazamiento, para tener el elemento deseado.

7.3.3. MANEJO DE FUNCIONES

Existen 2 tipos de funciones : las primitivas y las secundarias ; las primeras son aquellas que están escritas en el lenguaje en el que se implantó el intérprete de FORTH, y las segundas son aquellas que se encuentran escritas en FORTH mismo. Esto implica que las funciones primitivas son más rápidas al momento de ejecutarse, por lo que sería deseable poder tener todas las funciones requeridas en ROM, ya que de ésta manera, la ejecución ó interpretación de programas escritos en FORTH se volvería muy rápida; sin embargo, representa una gran cantidad de esfuerzo y trabajo, el poder generar código de alto nivel escrito en ensamblador. Lo que normalmente se hace, es tener un pequeño núcleo que ocupa unos 2K bytes, escrito totalmente en ensamblador y a partir de él generar funciones secundarias escritas en FORTH; las cuales podrían estar almacenadas en memoria de sólo lectura, en disco ó inclusive en cassette. Con un poco más de 2K bytes adicionales, los cuales incluyen un número bastante aceptable de funciones secundarias, es posible tener un sistema stand-alone; y a partir de éste se pueden generar ensambladores, editores, compiladores, etc., sumamente transportables y sobre todo fácilmente modificables. Las partes principales del intérprete son:

- DICCIONARIO

  Las funciones se encuentran almacenadas en una parte de memoria que se denomina DICCIONARIO; en él se almacenan nombres de funciones, código y datos. Existen 2 funciones que tranfieren información de la pila al diccionario: una es ",", la cual pasa una palabra del stack al diccionario, y "c,", la cual transfiere tan sólo un byte.

  El dicccionario está organizado en conjuntos de funciones llamados VOCABULARIOS. Al iniciar el sistema existen sólo dos: "CORE", el cual contiene las funciones mediatas (aquellas que se ejecutan en el momento de compilación) y "COMPILER", el cual contiene las funciones inmediatas (pueden ejecutarse en modo intérprete). El diccionario crece hacia las direcciones altas de memoria. Al redefinir una función, las llamadas a esa función hechas antes de de la redefinicion se conservan, y las nuevas llamadas ejecutarán la nueva definición.

- BUFFER DE ENTRADA

  El buffer de entrada es un arreglo de bytes en donde se almacena la información leída por la función INLINE.

- PILA DE DATOS

  Los parámetros de las funciones, se encuentran en una pila llamada "pila de datos" o simplemente "pila"; (se utiliza más en éste texto la palabra "stack"), la cual se encuentra organizada por palabras, de modo que no es posible almacenar un sólo byte y crece nacia las direcciones bajas.

- PILA DE RETORNO

  Para no interferir con los parámetros, el contador de programa se almacena en otra pila que también crece nacia las direcciones bajas.

7.3.4. FUNCIONES PRIMITIVAS

Las funciones primitivas que conforman un núcleo de tamaño aceptable , son alrededor de 120 a 150; con ellas es posible despegar totalmente del lenguaje de máquina; y a partir de aquél construir un lenguaje de alto nivel realmente pooeroso Enseguida se muestran las funciones de FORTH, más comúnmente usadas, con un ejemplo de lo que sucede en la pila de datos, que és de donde toman y dejan los parámetros, dichas funciones. Dependiendo del número de argumentos (1, 2 ó 3), se proporciona una lista de números (1, 2 ó 3) los cuales tienen el siguiente significado :

- el elemento de la extrema derecha representa el elemento que se encuentra en el tope del stack, el siguiente elemento a la derecha corresponde al segundo elemento del stack, y así sucesivamente.

- Si se encuentra algún número entre parentésis, significa contenido de la localidad

- se da enseguida el nombre de la función (los números que se encuentran antes del nombre de la función, representan los valores de los parámetros justo antes de llamar a la función).

- Los números que aparecen con una letra ĥ al final, son números hexadecimales.

- por último, se dan los valores finales de los parámetros después de ejecutar la función; el significado del orden en que se encuentran es el mismo que el dado arriba.

7.3.4.1. FUNCIONES QUE HACEN REFERENCIA A MEMORIA

- !

Almacena el segundo elemento del stack, en la dirección que se encuentra en el tope del stack.

```
(2345)=0
5 2345 !
(2345)=5
```

- C!

Almacena el byte menos significativo del segundo elemento del stack, en la dirección que se encuentra en el tope del stack.

```
(3333)=0
5678 3333 c!
(3333)=78
```

- @

Trae al stack el contenido de la dirección que se encuentra en el tope del stack.

```
(1111)=1234
1111 @
1234
```

- C@

Trae al byte menos significante del tope del stack, el contenido del apuntador que se encuentra en el tope del stack.

```
(1000)=56
1000 c@
0056
```

- BSET

Inicializa a cero la localidad cuya dirección se encuentra en el tope del stack.

```
(2000)=xxxx
2000 0set
(2000)=0
```

- 1SET

Asigna el valor 1 a la localidad cuya dirección se encuentra en el tope del stack.

```
(2000)=00
2000 1SET
(2000)=1
```

- C0SET

Asigna el valor 0 al byte cuya dirección está en el tope del stack.

```
(3000)=10
3000 C0SET
(3000)=0
```

- C1SET

Asigna el valor 1 al byte cuya dirección está en el tope del stack.

```
(3000)=0
3000 C1SET
(3000)=1
```

- +!

Suma el valor que se encuentra en el segundo elemento del stack, al contenido de la dirección que se encuentra en el tope del stack.

```
(3000)=1
5000 3000 +!
```

```
(3000)=5001
```

- C+!

  Suma el bytes menos significativo del segundo elemento del stack, al byte cuya dirección está en el tope del stack.

```
(4000)=23
67 4000 c+!
(4000)=100
```

7.3.4.2. OPERADORES DE STACK

- SWAP

  Intercambia los dos elementos del stack.

```
34 47
SWAP
47 34
```

- CSPLIT Separa los 2 bytes del tope del stack, los expande a 16 bits y pone el más significativo como segundo elemento y al menos significativo en el tope del stack.

```
1234h
CSPLIT
12 34
```

- CJOIN

  Toma los 2 bytes menos significantes del tope y segundo elementos del stack, y forma un número que pone en el tope (es el inverso de CSPLIT).

```
2312 1234
CJOIN
3412
```

- DUP

Duplica el elemento que se encuentra en el tope del stack.

```
78
DUP
78 78
```

- 2DUP

Replica 2 veces el tope del stack.

```
55
2DUP
55 55 55
```

- OVER Pone en el tope del stack una copia del segundo elemento del mismo.

```
34 45
OVER
34 45 34
```

- 2OVER

Pone en el tope del stack el tercer elemento del mismo.

```
11 22 33
2OVER
11 22 33 11
```

7.3.4.3. OPERADORES ARITMETICOS

- ABS

Obtiene el valor absoluto del tope del stack.

```
-1111
ABS
1111
```

- MINUS

Obtiene el complemento a 2 del tope del stack.

```
FFFFh
MINUS
0001
```

- /MOD

Divide el segundo elemento del stack entre el tope del stack, pone en el tope el residuo y el cociente como segundo elemento.

```
1234 100
/MOD
12 34
```

- MOD

Divide el segundo elemento del stack entre el tope, deja solamente el residuo en el tope del stack.

```
1234 100
MOD
100
```

- DIV

Divide el segundo elemento del stack entre el tope, deja en el tope del stack el cociente.

```
1234 100
DIV
12
```

- MAX

Compara el tope y el segundo elemento del stack y deja en el tope el mayor de ellos.

```
234 2345
```

```
MAX
2345
```

- MIN

Compara el tope y el segundo elemento del stack, deja en el tope el menor de ellos.

```
234 2345
MIN
234
```

- 1+

Incrementa el tope del stack en 1.

```
444
1+
445
```

- 1-

Decrementa el tope del stack en 1.

```
444
1-
443
```

- 2-

Decrementa el tope del stack en 2.

```
444
2-
442
```

- 2+

Incrementa el tope del stack en 2.

```
442
```

```
2+
444
```

- 2/

Divide entre 2 el tope del stack, (división entera).

```
121
2/
60
```

- 2*

Multiplica por 2 el tope del stack (equivale a un corrimiento a la izquierda).

```
60
2*
120
```

7.3.4.4. OPERADORES RELACIONALES

- >

Si el segundo elemento del stack es mayor que el tope del stack, deja un 1 en el stack; si no, deja un 0.

```
33 24
 >
1
```

- <

Si el segundo elemento del stack es menor que el tope del mismo, deja un 1; si no, deja un 0.

```
23 10
<
0
```

- =

Si el tope y el segundo elemento del stack son iguales, deja un 1 en el tope, si no deja un 0.

```
23 24
=
0
```

- 0=

Si el tope del stack es igual a 0, deja un 1 en el stack; si no deja un 0,

```
0
0=
1
```

7.3.4.5. OPERADORES LOGICOS

- AND

Efectúa el AND lógico bit a bit, de los 2 elementos del stack, deja el resultado en el stack.

```
1111h  ff00h
AND
1100h
```

- OR

Lleva a cabo el OR lógico bit a bit de los 2 elementos del stack, deja el resultado en el stack.

```
1110h 0101h
OR
1111h
```

- NOT

Obtiene el complemento a 1 del tope del stack, deja el resultado en el stack.

```
1234h
NOT
edcbh
```

- XOR

Efectúa el OR exclusivo del tope y el segundo elemento del stack.

```
llllh eeeeh
XOR
ffffh
```

7.3.4.6. FUNCIONES DE CONTROL DE FLUJO

- IF....THEN

Esta función sirve para ejecutar incondicionalmente una porción de código. IF también utiliza notación polaca postfija; esto significa que la condición debe escribirse antes del IF. Tiene un parámetro de entrada, y si su valor es 0, el control de flujo se transfiere a la instrucción que sigue a THEN. En caso contrario, se ejecuta el código que se encuentra entre IF y THEN.

- IF....THEN....ELSE

Al ejecutarse el IF, se extrae una bandera de la pila y si es 0, se transfiere el control a la instrucción que sigue a ELSE; si por el contrario es 1 el valor, se ejecuta el código que se encuentra entre IF y ELSE, y se ejecuta un brinco incondicional a la instrucción que sigue a THEN.

- DO....LOOP

Una vez compilado, DO tiene 2 argumentos de entrada: el primero se interpreta como el valor final que adquirirá el índice, y el segundo como el valor inicial del mismo. El índice se crea automáticamente. "LOOP" incrementa el valor del índice en 1; en caso de ser igual al valor máximo, el flujo continúa con la instrucción que sigue a "LOOP". En caso contrario, regresa a la instrucción que sigue a "DO". Dentro del ciclo "DO... LOOP", es posible examinar el índice por medio de la función "i>". El ciclo se ejecuta al menos

una vez y existe la posibilidad de anidar varios ciclos "DO ...LOOP"; solamente hay que tener cuidado al manipular los índices, a los cuales se tiene acceso todo el tiempo.

- <DO....OD>

  Esta función crea un ciclo infinito, es decir, "OD>" actúa como un brinco incondicional a la instrucción que sigue a "<DO". La manera más usual de salir de ella es por medio de la función RETURN, que termina la ejecución de la instrucción.

- BEGIN....END

  Esta función se utiliza para generar ciclos repetitivos, que terminan cuando se cumple una condición requerida, el modo de salir de estos es poniendo en el stack una bandera justo antes de la instrucción "END". si la bandera es 1 se continúa con la instrucción que sigue a "END"; se es 0, entonces se regresa el control a la instrucción que sigue inmediatamente después de "BEGIN".

7.3.5. FUNCIONES SECUNDARIAS

Las funciones secundaria son como ya se dijo, funciones que están formadas por funciones primitivas y/o funciones secundarias. La característica principal de éstas, es que dado que se encuentran escritas en el mismo lenguaje FORTH, resultan sumamente transportables ; y por lo tanto, pueden correr en otros sistemas que tienen un intérprete de código hilvanado. Se forman de la siguiente manera:

1. código de definición (:)
2. nombre de la función
3. Nombres de las funciones primitivas ó secundarias, las cuales deberán ser previamente definidas; aunque existe una manera de poser llamar a funciones que aún no se definen en el momento de llamarlas.
4. código de fin de definición (;)

Para llamar a las funciones previamente definidas, basta con poner el nombre de ellas, no existen CALLS. Tienen la ventaja de que pueden definirse de nuevo, de modo que las nuevas llamadas a

esas funciones ejecutarán la nueva definición. ejemplo : suma + ; En éste caso, se está definiendo la suma con el nombre "suma"; de modo que "suma" y "+" son equivalentes; y con la ventaja de que la definición anterior no desaparece.

7.3.6. DICCIONARIOS

En FORTH es posible tener varios diccionarios a la vez, éstos son conjuntos de funciones que se agrupan bajo un mismo nombre y se crean llamando a la función VOCABULARY, que como su nombre lo indica, cre un vocabulario asociado a un nombre, creando asímismo, una rama del diccionario. Lo único que tienen en común todos los posibles diccionarios es el núcleo de FORTH. De ahí en delante, cada uno puede tener sus propias definiciones, sus propias funciones; sólo se puede tener acceso a funciones de un cierto diccionario, cuando se está dentro del apropiado. Una definición de vocabulario podría ser : es un conjunto de funciones que se asocian a un nombre determinado, y que únicamente cuando se llama a ese nombre, es posible tener acceso a esas funciones. Esto abre la posibilidad de tener funciones con el mismo nombre, pero que efectúan cosas diferentes y que no existen simultáneamente en un momento determinado. Se puede decir que se crea un "ambiente" para un cierto tipo de funciones, las cuales sólo existen dentro de ese "ambiente"; y dado que existe la facilidad de cambiar de un diccionario a otro, tiene bastantes ventajas, como puede ser que el usuario no tenga acceso a las funciones del sistema por error ó deliberadamente.

7.3.7. DESARROLLO DE PROGRAMAS

El desarrollo de programas se puede llevar a cabo de dos modos diferentes : el modo intérprete y el modo compilación. En el modo intérprete, recibe instrucciones y datos y los va ejecutando conforme los va encontrando; de modo que la respuesta se puede decir que es instantánea. Por ejemplo, si se teclea en la terminal

```
10 20 * 5 + 50 -
```

el intérprete responderá:

```
155
```

En modo intérprete, como no hay generación ó definición de nuevas funciónes, no se almacena nada en el diccionario. En el modo

compilación, por el contrario; como su nombre lo indica, se recibe una serie de funciones que se definen por primera vez, ó que redefinen alguna otra y se traduce a código nilvanado, agregándose al diccionario. Cuando estas funciones son llamadas, dado que ya existen, simplemente se ejecutarán. Cuando existe la facilidad de diskettes ó cassettes, es posible guardar las nuevas definiciones en esos medios de almacenamiento y la siguiente vez que se desea llamar al intérprete, éste tendrá disponibles permanentemente las funciones definidas con anterioridad; haciendo cada vez más robusto al sistema. Por ejemplo, si queremos definir la función que obtiene el cubo de un número, se haría de la siguiente manera: : cubo dup dup * * ; El intérprete lo que hace en éste caso es "compilar" la nueva definición agregándola al diccionario para uso futuro. Los ":" indican que a continuación viene el nombre de una nueva función que hay que agregar al diccionario, enseguida viene el código, representado por cuatro funciones primitivas y finalmente la terminación de la definición, por medio de el ";". Cuando se llama a ésa función, como sólo requiere como parámetro un número, el cual se teclea antes de llamar a la función, el intérprete responderá con el valor del número elevado a la potencia 3. Por ejemplo, si se teclea

6 CUBO

el intérprete responderá

216

7.3.8. MANEJO DE PARAMETROS

Los parámetros que se utilizan en las rutinas de FORTH, se pasan de unas a otras por medio del stack de datos; ésto significa que no existen llamadas a funciones con el nombre de los parámetros; solamente existen valores de parámetros y direcciones de funciones. Dado que no existe chequeo automático de los parámetros, ni en el intérprete externo, ni en los compiladores, tanto de tipos, como del número correcto de ellos que son llamados por las funciones, es importante verificar que no falten ni sobren valores en la pila, pues ésto puede afectar la correcta interpretación de los programas. En éstos casos se recomienda diseñar ó construir muchas funciones ó rutinas pequeñas, de modo que nunca se pierda de vista que parámetros se hallan en el stack en un momento dado.

SISTEMAS OPERATIVOS Pag 229

CAPITULO 8
SISTEMAS OPERATIVOS

8.1. INTRODUCTION

El equipo de que están compuestas las computadoras modernas es muy poderoso, sin embargo los programas que se necesitan emplear para hacer que todo ese equipo realize una función tan simple como recibir o transmitir información de o hacia una terminal son muy complejos. Por lo anterior es importante disponer de un programa (el Sistema Operativo) que sepa llevar el control de todos los aparatos que forman una computadora y que tenga rutinas que puedan ser usadas por el programador, a las que sólo se tendrá que llamar y pasarle los parámetros apropiados para que realize la misma función que un programa que a nosotros nos llevaría mucho tiempo escribir y probar para hacer esa tarea.

La verdadera importancia del sistema operativo radica en que transforma el hardware inhóspito de la máquina, que es un ambiente verdaderamente hostil, en un medio mucho más sencillo de entender y trabajar con él. Para ilustrar lo anterior hagamos una analogía de la diferencia que existe entre el telégrafo que se usaba en las películas antiguas del oeste y el telex moderno. El primero era realmente difícil de usar, ya que el telegrafista tenía que aprender el código para cada caracter por medio de rayas y puntos, y despues tardaba mucho tiempo en acostumbrarse a oirlo para poder decodificar lo que le transmitían de otro pueblo, mientras que el segundo es una máquina de escribir que al oprimir una tecla la codifica en determinadas señales y la envía al telégrafo antiguo, el cual transmite las rayas y puntos a través de un cable a otro telégrafo antiguo, el cual a su vez comunica al telex el código recibido el cual lo decodifica e imprime el caracter correspondiente sin que el operador tenga que ver con el manejo de las señales que realmente se estan enviando por el cable telefónico.

8.1.1. LA IMPORTANCIA DEL SISTEMA OPERATIVO AL COMPRAR UNA MAQUINA

Al comprar una computadora los factores que intervienen en la decisión son:

1. El tipo de problema ya que si se trata de un problema de control en tiempo real necesitará de un sistema que permita ese tipo de manejo. Mientras que si sólo se necesita un sistema en el que no es relevante el tiempo de respuesta, se usará un sistema operativo de tipo convencional. (Ver NOTA).

2. La facilidad de adaptar el sistema operativo al problema propio.

3. La capacidad que tiene ese sistema para manejar los diferentes periféricos adaptables a la máquina y que nos ayudan a resolver nuestro problema.

4. La facilidad del sistema para ser entendido por nuevos usuarios, ya que es él con quien tienen que verselas, los usuarios, al momento de querer realizar cualquier tarea o programa.

El comprador probablemente encontrará que el vendedor dependiendo de la marca de la máquina le dará diferentes nombres al **sistema operativo** tales como: el **programa controlador**, el **supervisor**, el **ejecutivo** o el **monitor**.

NOTA: Que el sistema sea de tiempo real significa que el tiempo de respuesta es importante y debe ser mínimo. Los sistemas de tiempo real caen dentro de uno de las siguientes casos:

- Control de Procesos: como en el caso de una industria para llevar el control de la temperatura de sus hornos o calderas, o en el caso de una hidroeléctrica para llevar el control de cuando meter o sacar de funcionamiento las turbinas productoras de energía eléctrica de acuerdo a la demanda que de ella exista. El sistema operativo tendrá que dar la mayor confiabilidad posible al proceso de forma que haya poca intervención humana y dé seguridad contra fallas de alguna de las máquinas que estén bajo control.

- En Sistemas de Información: hay sistemas de información o de bases de datos en los que se requiere de respuesta inmediata (unos cuantos segundos) para realizar la transacción que se desea operar. Dichas transacciones pueden ser preguntas o modificaciones hacia la base de datos. Este tipo de sistemas pueden ser un sistema de transacciones bancarias, un sistema de transacciones de una compañía aerea, un sistema de información médica acerca de la historia clínica de algún paciente en un hospital, etc.

Un sistema operativo convencional, será aquel en el que se podrán procesar todos los trabajos sin estar sujetos a restricciones de tiempo en la respuesta del sistema, este tipo de sistemas se pueden usar para procesar la nómina de una compañía,

o para procesar la enorme variedad de problemas en un medio universitario, o para procesar las compras o las órdenes de venta de una compañía, llevar estadísticas, y otras muchas aplicaciones.

Estos sistemas pueden ser clasificados de la siguiente manera:

- **BATCH** En estos sistemas una vez que la computadora comienza a procesar un trabajo, se sigue con él hasta terminarlo, el programador no tiene ninguna interacción con el programa hasta que termina de procesarse y se obtienen los resultados. Estos son los programas que se corren por tajetas.

- **ACCESO MULTIPLE** Es el S.O. en el que el usuario podrá tener interacción con los programas que está corriendo a través de una terminal, desde la cual podra monitoriar su programa, o estar metiendo datos y recibiendo respuestas en forma interactiva.

Es importante resaltar que, sin importar el tipo de sistema operativo ya sea de tiempo real o no, estos sistemas pueden estar corriendo en una sóla computadora o en varias máquinas interconectadas. En este último caso el trabajo total del sistema puede estar distribuido en forma equitativa en todas ellas, o cada una puede estar efectuando una función específica. Por ejemplo, una puede estar dedicada a manejar todo lo que se trate de E/S con los periféricos y otra a realizar la ejecución de todas las demas tareas como la ejecución de la parte número de los programas bajo proceso.

8.1.2. LAS FUNCIONES DEL SISTEMA OPERATIVO

El Sistema Operativo tiene que administrar los recursos de la computadora, distribuyendolos entre los usuarios de la manera más eficiente. Un buen administrador debe realizar las siguientes funciones:

1. Llevar el control de cuales son los recursos del sistema.

2. Controlar la asignación de cada uno de los recursos.

3. Controlar la recuperación de los recursos, una vez que estos ya no son utilizados.

4. Una vez determinada la política de asignación de recursos, tratar de apegarse, en lo posible, a ella en función de su eficiencia.

8.1.2.1. LAS PRINCIPALES FUNCIONES YA APLICADAS A LA COMPUTADORA SON:

1. La asignación del procesador a cada uno de los programas que se estén ejecutando.

2. El control, distribución y recuperación de la memoria entre los distintos programas que estan ejecutandose.

3. El acceso a medios de almacenamiento tales como disco, cinta o disco flexible. Pero es importante resaltar aquí que el acceso a estos medios puede ser al medio físico, como es el caso de un sector y track determinado de un disco, o al medio lógico cuando estamos accesando un archivo y es el sistema operativo el que sabe manejar la existencia de estas entidades lógicas.

4. El acceso a los demás periféricos de entrada salida que posea el equipo tales como:

   - Lectora o perforadora de tarjetas.
   - Lectora óptica.
   - Impresoras.
   - Terminales, y otros periféricos.

5. Funciones de manejo del sistema de archivos.

6. Protección contra fallas del sistema o contra sabotaje.

8.2. KERNEL

El sistema operativo está formado por gran cantidad de rutinas, pero hay que hacer notar que no todas se encuentran presentes en la memoria mientras el sistema está corriendo. El **Kernel** está formado por todas aquellas rutinas del sistema operativo que siempre se encuentran residentes en memoria. Las demás rutinas se encuentran en disco y solo son cargadas en la memoria cuando se les necesita.

Generalmente la mayor parte de estas rutinas pueden ser llamadas desde los programas de los usuarios, pero existen ciertas rutinas especiales tales como:

- Checa la clave del usuario.
- Lista el directorio.
- Borra un archivo.
- Cámbiale el nombre.
- Ejecuta un programa y pasale los siguientes parámetros.
- Cual es el status de mi pograma (elapsed time, i/o time, process time).
- Haz una copia del archivo.
- Modifica la protección del archivo.

Las cuales sirven para interactuar con el usuario y no para ser llamadas por sus programas. Es ahora cuando hay que tener en cuenta que existen ciertas rutinas del sistema operativo que pueden ser invocadas por el usuario directamente desde su terminal, las cuales serán reconocidas por la parte del sistema que llamaremos **El Interprete de Comandos.** El cual en CP/M es conocido como el CCP (procesador de comandos de la consola).

A CONTINUACION SE VA A EXPLICAR UN SISTEMA OPERATIVO PARA UNA MICROCOMPUTADORA

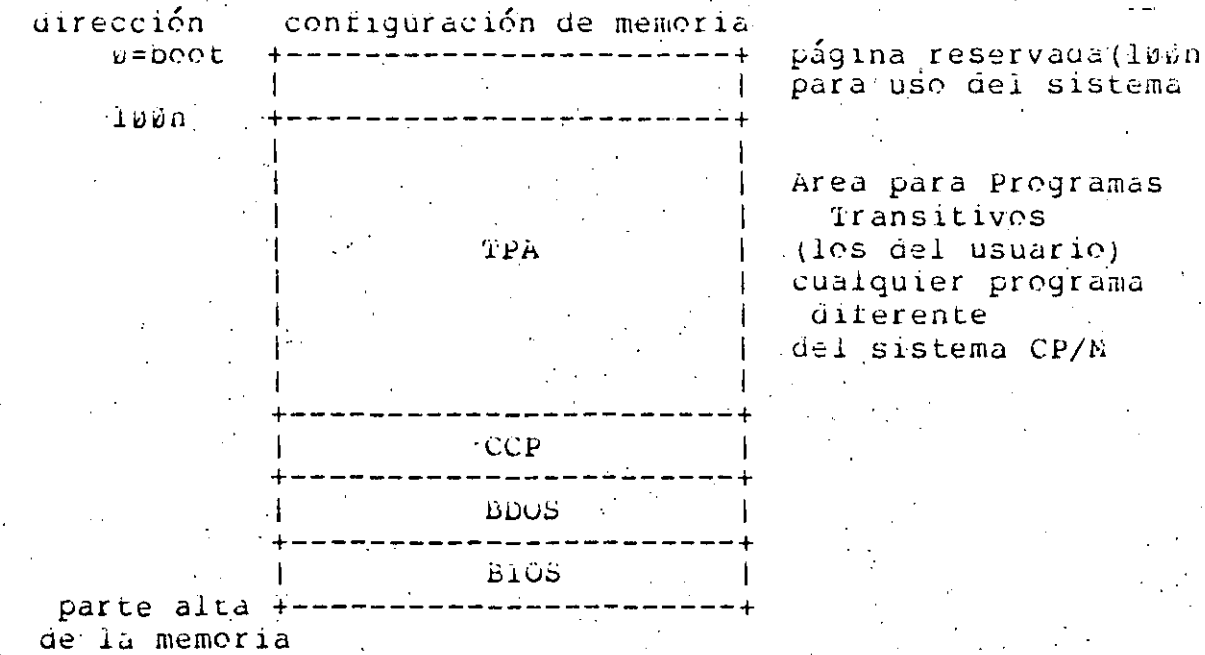
Antes de comenzar este tema deseo hacer notar que un S.O. para una microcomputadora que maneja un sólo usuario y no maneja multiprogramación, es bastante más sencillo en cuanto al control, asignación y recuperación de los recursos de la micro que el S.O. de una supercomputadora que maneja múltiples usuarios y varios procesadores en un ambiente de multiprogramacion y/o multiproceso.

8.3. CP/M CONTROL PROGRAM FOR MICROCOMPUTERS

Es el sistema operativo más comunmente usado en micros basadas en el 8080, 8085 y Z80.

Las partes principales de este sistema son:

1. BIOS - sistema básico de E/S su programación es completamente dependiente de los periféricos usados (diferente para cada marca de periférico).
2. BDOS - sistema básico de operación de disco el cual ya no depende de la configuración específica de los periféricos.
3. CCP - es el procesador de los comandos de consola (o monitor), el cual hace uso del BDOS.
4. TPA - es el area donde corren los programas transitivos.



Mapa de memoria de CP/M.

8.3.1. FUNCIONES DEL BIOS

El BIOS provee de los manejadores de los periféricos necesarios tales como e/s de discos, de tty, de crt, perforadora y lectora de cinta de papel y otros periféricos específicos del usuario, en total 17 funciones diferentes, las cuales son:

1. BOOT hace la carga inicial de CP/M (desde el disco) incluyendo las inicializaciones de puertos y sus velocidades, así como el mensaje de entrada al sistema. También carga las primeras 8 localidades de memoria con la siguiente información:

localidades	contenido
0, 1, 2	JMP WBOOT
3	valor inicial del IOBYTE
4	el número del disco al que entra por default.
5, 6, 7	JMP BDOS que es el punto primario de entrada a CP/M para los programas transitivos

Finalmente el control es transferido al CCP con el registro C=0 para seleccionar el drive A.

2. WBOOT un warm start se realiza cada vez que el programa de un usuario salta a la localidad 0000H o cuando el CPU es reiniciado dandole reset. CP/M será cargado de las primeras dos pistas del disco A pero ahora sin incluir el BIOS, los mismos parámetros y localidades de memoria que son iniciados en el cold boot se reasignan aquí y finalmente se salta al CCP.

3. CONST Prueba el estado de la consola y regresa un 00h en el registro A si no hay ningún caracter listo para ser leido, o un 0FFH si hay un caracter listo para leerse.

4. CONIN Lee un caracter de la consola y lo deja en el registro A, y pone el bit de paridad en cero. Si no hay ningún caracter listo en la consola espera hasta que es tecleado antes de retornar.

5. CONOUT Envía el caracter que esté en el registro C a la consola. El caracter estará en ASCII con el bit de paridad en cero. En esta rutina se pueden filtrar caracteres de control que ordenen a la consola algúnas funciones especiales tales como el clear de la pantalla y otras.

6. LIST Envía el caracter del registro C a la impresora. El caracter está en ASCII con el bit de paridad igual a cero.

7. PUNCH Envía el caracter del registro C a la perforadora. El caracter está en ASCII con el bit de paridad igual a cero.

8. READER Carga el caracter de la lectora al registo A con el bit de paridad en cero, la condición de fin de archivo es reportada enviando un control-Z (ASCII).

9. HOME Regresa la cabeza del disco a la posición de la pista cero.

10. SELDSK selecciónа el disco determinado por el valor del registro C, y regresa en HL la dirección base del area llamada el disk parameter header (DPH). Si se trata de selecciónar un disco no existente, regresa HL=0000h como indicación del error.

11. SETTRK el registro BC contiene el número de pista a la que se harán los subsecuentes accesos.

12. SETSEC el registro BC contiene el número de sector que será accesado en la siguiente operación de E/S.

13. SETDMA el registro BC contiene la dirección de DMA (direct memory address) para las subsecuentes operaciones de E/S.

14. READ Suponiendo que el disco, la pista, el sector y el DMA ya han sido determinados, esta rutina trata de leer el sector especificado, regresando los siguientes valores en el registro A.

0	si terminó sin errores
1	si ocurrió un error irrecuperable

Normalmente se tratará unas 10 veces de realizar alguna operación antes de declararla irrecuperable.

15. WRITE escribirá los datos tomandolos de la dirección de DMA especificada, en el disco, pista y sector seleccionados. Como salida dará los mismos valores que el READ.

16. LISIST regresa los siguientes valores en el registro A:

    00 si la impresora aun no está lista para recibir un caracter.

    FF si se puede enviar a listar un caracter.

17. SECTRAN realiza la traducción del número de sector lógico al físico, la cual se realiza para mejorar la respuesta de CP/M, en la E/S, a través de un skew factor de n, con lo que n sectores son saltados entre cada operación logica de E/S. Dicho factor da, a la mayoria de los programas, el tiempo suficiente para que carguen sus buffers y alcancen el siguiente sector en la misma revolución del disco.

    Esta rutina recibe el número de un sector logico en BC y la dirección a la tabla de traducción en DE. El número de sector es utilizado como el índice para entrar a la tabla y asi cargar el número del sector físico en HL.

8.3.2. FUNCIONES DEL BDOS BASIC DISK OPERATING SYSTEM

Es a través del BDOS que los programas del usuario podrán realizar funciones de E/S de o hacia la consola, la lectora o la perforadora de cinta o la impresora, y es también a través del BDOS que el usuario tiene la facilidad del manejo de archivos en discos. Ya que es esta parte de CP/M la que puede controlar uno o más discos conteniendo directorios de archivos independientes, asi como la construcción dinámica de archivos, tratando de que cumplan con la propiedad de cercanía para minimizar el movimiento de la cabeza durante el acceso.

El nombre de un archivo en disco está constituido por 3 partes principales: el código de selección del disco, el nombre del archivo que puede tener hasta 8 caracteres, y la extención que está formada por 3 caracteres. Ej. A:POLI.TXT

CP/M permite manejar archivos de hasta 64K registros, donde cada registro tiene 128 bytes de longitud, lo que da 8 Mbytes de

capacidad de direccionamiento para un archivo. Los archivos están divididos en segmentos de 16K bytes, y cada segmento esta direccionado por un extent, lo cual significa que si un archivo tiene más de 16K de longitud, ocupara un extent en el directorio del disco por cada 16K o fracción que haya que direccionar.

Casi todas las funciones del sistema de archivos del S.O. reciben en el registro par DE la dirección del FCB File Control Block que halla definido el usuario. Hay sin embargo un area que usa el S.O. y que puede ser usada por el programador, y se encuentra en la dirección BOOT+05Ch. Todas las funciones del sistema que tienen que ver con la lectura o escritura de archivos del disco, utilizan un burfer llamado DMA, el cual se encuentra localizado en la dirección BOOT+80H. Sin embargo la dirección de este buffer puede ser cambiada dinámicamente según las necesidades del usuario.

El FCB esta formado por 36 bytes si se estan realizando funciones de acceso random, o de 33 bytes y si el acceso a los archivos es secuencial. A continuación se muestra un FCB y sus campos:

```
+-----------------------------------------------------+
!dr!n1!n2!..!n8!e1!e2!e3!ex!s1!s2!rc!d0!..!dn!cr!r0!r1!r2!
+-----------------------------------------------------+
```

dr código del disco a usar (0-16)

0 el disco actualmente seleccionado
1 el disco A
...
16 el disco P

n1..n8 Estos 8 bytes llevan el nombre del archivo en código ASCII.

e1..e3 En estos 3 bytes se guarda el tipo del archivo en ASCII. Pero además el bit más significante de estas posiciones da información de la protección.

bit 7,e1 read only
bit 7,e2 archivo del sistema no listarlo

ex Tiene el número del extent del archivo, de 0 a 31.

s1 s2 Están reservados para uso del sistema, pero s2 debe ponerse igual a cero cuando se llame a OPEN, MAKE o SEARCH.

rc Es el contador que indica que registro del extent se está accesando va de 0-128

d0..dn Block allocation map. Es una secuencia de 1 a 16 bytes que indica qué bloques del disco está usando este archivo en este extent.

cr Se refiere al registro actual y solo se usa para acceso secuencial del archivo. 0 a 255

r1..r2 Es el campo opcional para acceso aleatorio a los archivos. de 0 a 65535. r2 indica si hay sobreflujo cuando es diferente de cero

El BDOS permite al usuario hacer llamadas a 37 funciones del S.O. (BDOS). La manera de hacer dichas llamadas es cargar el registro C del cpu con el No. de la función, poner en E o DE el parámetro, si existe, hacer un CALL 05H, y los valores de salida serán regresados en A o en HL. Si HL es igual a cero, significa que la función no estaba definida.

A continuación se enumeran las funciones del BDOS y su número corresponde al no. de función que tiene que ser cargado en C.

- 0. WARM BOOT: Esta rutina carga nuevamente el CCP y el BDOS del Sistema Operativo y reinicializa las variables del sistema dejando seleccionado el disco A y el user 0.

1. READ CONSOLE: Espera hasta leer un caracter ASCII de la terminal, al regresar deja en A el caracter leido de la terminal, y además lo despliega en la pantalla a menos que se trate de un caracter de control. También realiza el chequeo para saber si se está tecleando ctl-S para detener el listado sobre la consola de algún archivo, o ctl-P para mandar a la impresora lo que se está desplegando sobre la terminal.

2. WRITE CONSOLE: Hay que poner en E el caracter, en ASCII, que se desea imprimir en la pantalla. Esta rutina también realiza el chequeo para ver si el usuario ha tecleado ctl-P o ctl-S.

3. READ READER: Espera hasta leer un caracter ASCII de

lectora de cinta de papel y al retornar lo deja en el registro A.

4. WRITE PUNCH: Se pasa en el registro E el caracter ASCII que se desea perforar en la cinta de papel.

5. WRITE LIST: Se le pasa en E el caracter ASCII a ser escrito por la impresora.

6. DIRECT CONSOLE I/O: Esta función permite al usuario leer de la consola si el registro E contiene un 0FFH, sin embargo si no se ha tecleado ningún caracter, la rutina regresa A=0, esto significa que no espera hasta que se teclee algo en la consola. Si E no es igual a 0FFH, entonces lo que hace es escribir a la consola el caracter que está en el registro E asumiendo que es ASCII.

7. RETURN I/O BYTE VALUE: Esta rutina lo que hace es ir a leer el valor que está en la localidad 3 de memoria donde se almacena el I/O BYTE, y lo deja en el registo A.

8. MODIFY I/O BYTE: Esta rutina requiere tener en el Registo E el valor del nuevo I/O byte, y lo que hace es escribir dicho valor en la localidad 3 de memoria.

9. PRINT STRING FROM BUFFER: En el par DE se pasa la dirección de donde comienza el buffer que se desea imprimir. La rutina toma este valor y comienza a mandar a la pantalla todo lo que esté en memoria a partir de la dirección indicada por DE hasta que se encuentra con un caracter $ que indica el fin de la cadena. La rutina también esta pendiente por si el usuario teclea ctl-P o ctl-S.

10. READ STRING TO BUFFER: Para llamar esta función se debe poner en DE la dirección inicial del buffer en memoria, entonces la rutina comienza a leer de 1 a 255 caracteres, la lectura termina cuando se exede el maximo número de caracteres que puede tener el buffer, o cuando se envía un line feed o un carry return.

11. GET CONSOLE STATUS: Esta rutina checa si ya se ha tecleado algún caracter en la consola. Si lo hay regresa A=0FFH, si no A=00H.

12. GET CP/M VERSION NUMBER: Esta función regresa en el par HL el número de versión del S.O. que está corriendo y sirve para saber si un programa puede correr en la versión que tiene el usuario, ya que de

una versión a otra cambian algunas rutinas del sistema operativo.

13. RESET DISKS: Esta rutina permite al programa del usuario reiniciar el sistema de archivos con todos los discos listos para R/W y solamente se tendrá seleccionado el disco A, y la dirección del buffer de E/S de los discos (DMA) será BOOT+80H.

14. SELECT DISK: Se pasa en el registro E el número del disco sobre el que se harán las subsecuentes operaciones con discos, a menos que se indique explísitamente otro disco.

15. OPEN FILE: en DE se pasa la dirección del FCB que es el descriptor del archivo que se quiere habrir. En A se deja un valor de 0 a 3 si pudo abrirse el archivo y 0FFH si no.

16. CLOSE FILE: DE tiene la dirección del FCB a la salida deja A = 0-3 si se encontró un archivo con ese nombre y se pudo cerrar, o A = FF si no.

17. SEARCH FOR FIRST: En DE va la dirección del FCB, y regresa en A 0-3 si encontro algúno o FF si no, y lo que hace es buscar el primer entry del directorio que cace con el FCB y lo lee dejandolo en la dirección del DMA.

18. SEARCH FOR NEXT: Esta función es la continuación de la anterior, y sirve para ir buscando los demas Entries de un archivo, en A regresa FF cuando ya no hay más Entries que chequen.

19. DELETE FILE: En DE se pasa la dirección del FCB del archivo a borrar, en A deja el código de error 0-3 si hubo algún archivo para borrar, o FF si no.

20. READ SEQUENTIAL: En DE va la dirección del FCB, y regresa en A cero si se pudo leer, o un valor diferente de cero si no se pudo. Suponiendo que el archivo direccionado por el FCB ha sido activado (por un open file o make), esta función lee el siguiente registro del archivo (128 bytes) dejandolo en la dirección dada por el DMA.

21. WRITE SEQUENTIAL: En DE va la dirección del FCB, y regresa en A 0 si escribió, u otro valor si el disco ya se llenó. La función escribe 128 bytes que toma del DMA.

22. MAKE NEW FILE: Recibe en DE la dirección del FCB y regresa en A 0-3 si pudo abrirlo, o FF si no.

23. RENAME EXISTING FILE: DE = dirección del FCB regresa A con 0-3 si pudo, FF si no.

24. DETERMINE LOGIN VECTOR: A la salida deja en HL el vector con la información de cuales de los 16 discos están activados.

25. RETURN CURRENT DEFAULT DISK: Esta función regresa en A un valor de 0-15 dependiendo de cual es el disco que está actualmente como el disco seleccionado (ver select disk).

26. SET DMA ADDRESS: Se le pasa en DE la dirección del nuevo DMA para las subsecuentes operaciones de lectura o escritura del disco.

27. GET ALLOCATION VECTOR: A la salida regresa en HL la dirección del vector de memoria asignada del disco que está activo en ese momento.

28. WRITE PROTECT DISK: Esta función permite proteger, temporalmente el disco que se encuentra seleccionado en ese momento hasta que se produzca el siguiente boot o warm start.

29. FIND READ/ONLY VECTOR: Esta rutina deja, al salir, el vector de los discos que estan protegidos temporalmente contra escritura.

30. SET FILE ATRIBUTES: Al entrar DE debe contener la dirección del FCB que tiene los nuevos atributos.

31. GET ADDRES OF DISK PARAMETER BLOCK (DPB): Al salir de esta rutina, HL contiene la dirección del DPB del BIOS. Los parámetros del disco se pueden usar para calcular el espacio del disco.

32. SET/GET USER NUMBER: Se pasa en el registro E el número de usuario 0-31 o se pone en E 0FFH para solicitar que obtenga cual es el usuario actual. A la salida dejará en el registro A el número del usuario si al entrar E fue 0FFH.

33. READ RANDOM: A la entrada DE tiene la direción del FCB y a la salida A contendrá un código de error de 0 a 6. La imformación leida es dejada en la dirección indicada por DMA.

34. WRITE RANDOM: Esta función recibe en DE la dirección del FCB, y a la salida deja en A el código de error de 0 a 6. Esta rutina toma 128 bytes de la dirección que indica DMA y los escribe en el disco en el registro indicado.

35. GET FILE SIZE: Al entrar en DE va la dirección del FCB, a la salida deja en 3 de los bytes del FCB (r0, r1 y r2) la información del tamaño del archivo en número de registros ocupados.

36. SET RANDOM RECORD: A la entrada DE contiene la dirección del FCB, a la salida deja el FCB modificado con la posición random de la posición del registro actual que se ha leído o escrito en forma secuencial.

Además todas estas funciones pueden ser llamadas a través de los programas de usuario.

8.4. DEPURADORES DE PROGRAMAS

Un depurador es un programa que sirve para probar y depurar los errores que pueda tener cualquier programa en ensamblador que el usuario esté desarollando en su computadora.

8.4.1. CARACTERISTICAS DE LOS DEPURADORES

A través del depurador el usuario puede: desplegar el contenido de la memoria en varios formatos (en hexadecimal y ASCII o también desensamblar el código para listar los menmónicos), puede transferir el control al programa, y es aquí donde resulta útil que permitán insertar brakepoints para detener la ejecución del programa en algún punto de especial interés, y una vez ahí desplegar el contenido de los registros y modificarlos si se desea, o modificar también el contenido de la memoria.

Hay otros depuradores que no manejan breakpoints y que entonces manejan la opción de hacer el rastreo del programa diciendo el usuario durante cuantas instrucciones se desea hacer este rastreo, y deteniendose en este punto mostrando los valores de los registros.

Es importante hacer notar que aunque los depuradores permitan desensamblar la información en memoria, algúnas veces fallarán mostrando instrucciones que no existen si dentro del código del programa hay areas de datos, ya que el desensamblador

no tendrá conocimiento de esto y tomará los datos como instrucciones si puede.

CAPITULO 9
APLICACIONES

9.4. NO NUMERICAS

9.4.1. INTRODUCCION

En este momento nos dedicaremos a comprender qué es un sistema para el manejo de una base de datos y para qué sirve. Como primer punto, dejaremos establecido que estos sistemas son de propósito general, y no están hechos para resolver algún problema específico. Lo cual significa que deben de ser flexibles para adaptarse al problema de cada usuario.

Un sistema de manejo de bases de datos (SMBD) es un programa que permite a los usuario crear archivos ligados entre sí, a través de diferentes llaves, sin tener que saber como estan almacenados los archivos ni como se lleva el mantenimiento y actualización de las ligas o apuntadores de las diferentes llaves, ya que de esto se encargará el SMDB. También permite al usuario utilizar dichos archivos para realizar preguntas acerca de los datos, o para actualizar la información que se encuentre en los archivos, dando facilidades para crear reportes y listados organizados de acuerdo a las necesidades de los usuarios.

Para ilustrar lo anterior pensemos en un sistema de facturación, en el cada factura involucra el uso de varios campos de datos que se encuentran relacionados entre sí por el nombre del cliente, el número de la factura y la fecha. En un SMBD al expedir una factura se verán afectados inmediatamente otros archivos además del de facturas, por ejemplo el de inventarios, el de registro de clientes si el cliente aún no habia sido dado de alta, etc. ya que todos los archivos forman parte de la base de datos, y cuando se realiza cualquier transacción contra alguna parte de la base de datos (BD) el sistema automáticamente actualizará todos los datos que estén relacionados con el campo de datos en cuestión.

9.4.2. DEFINICIONES BASICAS

A continuación se dará el significado de algunos de los términos que son utilizados en bases de datos de acuerdo a como fueron definidos por CODASYL:

- DATA ITEM: Es la mínima unidad de datos a la que podemos hacer referencia, y puede tener asignado un

nombre (nombre del campo) y solamente estar definida dentro de un rango de valores. Son ejemplos el número de parte de una pieza, o el número de cliente, o el número de proveedor, etc.

- RECORD TYPE: Es un conjunto de cero o más data items. El usuario le puede dar nombre al record type. Un ejemplo de record type es el que identifica el PROVEEDOR, y consiste de los siguientes data items: nombre del proveedor, dirección, teléfono, partes que provee. Con este record type estoy identificando la clase de datos que habrá en la ocurrencia de un record, pero un record type no tiene datos nacerca de ningún proveedor. Esto significa que sólo me permite definir la clase de datos que deben de estar en cada campo.

- RECORD OCURRENCE: Cada record type definido en la base de datos, tal como PROVEEDOR, puede tener muchas ocurrencias, una para cada proveedor para ser preciso. Es así que se le llama record ocurrence al conjunto de valores que siendo del tipo definido en el record type ocupan un registro detro de algún archivo de la BD.

- SET-TYPE: De la misma manera que los data items se agrupan en records, los records se pueden agrupar dentro de sets o conjuntos. En CODASYL un set consiste generalmente de un record ocurrence que llamaremos el PROPIETARIO y de varias ocurrencias de otro record type que llamaremos los MIEMBROS asociados al propietario.

Los SMBD permiten los usuarios ver la organización lógica de los datos a través de tres estructuras de datos que son:

- JERARQUICA: la base de datos se ve como un árbol o una jerarquia. Sólo hay una forma de llegar a un dato que es a través de su predecesor. En estos sistemas sólo se permite tener relaciones múltiples del padre a los hijos, pero no de los hijos al padre.

- RETICULAR: la BD permite manejar múltiples relaciones entre los records , lo cual produce una red de relaciones.

- RELACIONAL: permite al usuario ver los datos como tablas de dos dimensiones como las que estamos acostumbrados a usar todas las gentes.

9.4.3. CARACTERISTICAS DE UN SISTEMA DE MANEJO DE BASES DE DATOS

Estos sistemas deben ofrecer varias de las características que a continuación se enumeran:

1. LENGUAJE PARA LA DEFINICION DE LOS DATOS: Un SMBD debe tener alguna forma de permitir al usuario definir que tipos de datos habrá en su BD, quien tendrá acceso a que parte, que tipos de registros formarán la BD.

2. PERMITE TENER VISTAS MULTIPLES DE LOS DATOS: Un SMBD manejará estructuras de datos complejas para permitir vistas múltiples de los mismos datos. Por ejemplo el programador que trabaja en el departamento de ventas necesita de algunos datos (como el número de parte vendida) que también necesitan los programadores del departamento de embarques y almacén para mantener las existencias mínimas del almacén, sin embargo cada uno necesita tener diferentes datos asóciados al número de parte. Por ejemplo el vendedor necesita tener el nombre y dirección del cliente, pero el de almacén necesita tenerlo asociado al nombre del proveedor para solicitar nuevamente partes.

3. FLEXIBILIDAD DE LA BASE DE DATOS: El sistema debe dar flexibilidad en el manejo de la definición de qué datos forman la BD, ya que si en un futuro se necesita un nuevo campo, o deja de servir algún otro campo se debe tener la flexibilidad para eliminar o insertar campos de los registros de la base.

4. EVITAR LA REDUNDANCIA DE LOS DATOS: A menudo un tipo de campo, como el nombre del proveedor, aparece en varios archivos de la base. Si el prveedor cambia su nombre, entonces el sistema tendrá que actualizar el nombre en cada uno de los archivos de la base, y si además el archivo está ordenado por el nombre del proveedor, entonces habrá que reordenar el archivo. Si la BD tiene una organización apropiada, entonces un dato aparecerá una sóla vez y habrá referencias a él desde otros archivos, y será fácil hacer la actualización de cualquier dato.

5. SEGURIDAD DE LOS DATOS: Debe proteger la privacidad y la integridad de los datos que forman la base. Por ejemplo algunos usuarios solo podrán leer los datos pero no actualizarlos.

6. DICCIONARIO DE DATOS DE LA BASE DE DATOS: El SMBD debe tener un diccionario en el que se especifiquen que

tipo de datos hay en la base, definiendo el nombre de cada data item, el tipo de dato (caracter, real, entero) su longitud, su nivel de protección lectura/escritura. También tendrá las definiciones de cuales son los data tupes que forman cada record type, así como que record types forman cada set-type. Con esta información se podrán escribir programas generales que son independientes de los archivos particulares de la base, y que podrán accesar la BD al extraer información del diccionario de datos para que de acuerdo a esta información, se accesen los archivos de la BD.

7. PROCESO DE QUERYS O PREGUNTAS: Un SMBD debe permitir accesar y actualizar la información qué mantiene. para esto tiene dos alternativas, contar con un interprete o procesador de peguntas, o contar con la interface hacia algún lenguaje anfitrión. En el primer caso, el procesador de preguntas permitirá agregar, actualizar y desplegar datos de la base. Estos sistemas permiten crear reportes en base a preguntas que se hacen contra la BD en forma interactiva.

8. INTERFACE CON UN LENGUAJE ANFITRION: Esta es la segunda alternativa para trabajar con una BD, y es la forma en la que la mayoría de los SMBD permiten accesar la BD. La interface puede ser a través de llamadas a rutinas y debe permitir:

   a. CREA: crea una ocurrencia de un registro.

   b. ALMACENA: guarda los datos en la BD.

   c. TRAE: saca un dato de la BD.

   d. MODIFICA: actualiza el valor de un data item dentro de un data ocurrence.

   e. INSERTA: agrega un data ocurrence a la BD.

   f. BORRA: borra un data record.

BIBLIOGRAFIA

CAPITULO 1

1.1 - ELECTRONICS, Special Commemorative Issue, Vol. 53, No. 9, abril 17, 1980.

1.2 Noyce, R. y Hoff, M.E.Jr., "A History of Microprocessor Development at Intel", IEEE MICRO, vol. 1, No. 1, pp. 8-21 (febr. 1981).

1.3 - DATAPRO REPORT, "All About Microcomputers", 37 pag. (junio 1978).

1.4 - PROCEEDINGS OF THE IEEE, Special Issue on Microprocessor Technology and Applications, Vol. 64, No. 6 (junio 1976).

1.5 Vacroux, A.G., "Microcomputers", SCIENTIFIC AMERICAN, Vol. 232, No. 5, pp. 32-40 (mayo 1975).

1.6 - COMPUTER, Special Issue on Small Scale Computing, Vol.10, No. 3 (marzo 1977).

CAPITULO 2

2.1 Hall, Douglas, "MICROPROCESSORS AND DIGITAL SYSTEMS", McGraw-Hill International Student Edition, pag. 426, Tokio, Japon (1980).

2.2 Howes, M.J. y Morgan, D.V., "LARGE SCALE INTEGRATION", John Wiley and Sons, pag. 346, Nueva York, EUA (febr. 1980).

2.3 A Scientific American Book, "MICROELECTRONICS", W.H. Freeman and Co., pag. 145, San Francisco, EUA (sept. 1977).

CAPITULO 3

CONCEPTOS GRALES.:

3.1 - "Microelectronics", A Scientific American Book, Freeman and Co., 1977.

3.2 Hamacher, Vranesic, Zaky, "Computer Organization", Computer Science Series, McGraw Hill, 1978.

3.3 Hall, Douglas V. "Microprocessors and Digital Systems", McGraw Hill International Student Edition, 1980.

3.4 Hayes, "Computer Architecture and Organization", Computer Science Series, McGraw Hill.

3.5 Stone, Siewiorek,"Introduction to computer organization and Data Structures: PDP-11 Edition". McGraw Hill.

3.6 Morris Mano, M. "Computer System Architecture", Prentice Hall.

3.7 Chu, "Computer Organization and Microprogramming", Prentice H.

3.8 Katzan, "Microprogramming Primer", McGraw Hill.

3.10 Ullman, "Fundamental Concepts of Programming Systems", Addison Wesley.

MICROS DE 8 BITS :

3.11 - "Micro Procesor Specification", DATAPRO RESEARCH CORPORATION 1978.

3.12 - Intel, "Component Data Catalog 1979", INTEL CO., pp. 10-1 a 10-22 (1979).

3.13 Kony, P y Larsen, D, "The 8080 Bugbook", SAMS, 1979.

3.14 Barden, "The Z80 microcomputer handbook", SAMS, 1979.

MICROS DE 16 BITS y MMUs :

3.15 Orlando, R.V. y Anderson, T.L., "An overview of the 9900 Microprocessor Family", IEEE MICRO, vol. 1, No. 3, pp. 38-44 (agosto 1981).

3.16 - Zilog, "Z8000 CPU Product Specifications", ZILOG INC, (octubre 1979).

3.17 - Motorola, "MC68000 16-bit microprocessor, Users Manual", MOTOROLA INC, (enero 1980).

3.18 - Zilog, "Z8000 Family, Tecnnical Overview", ZILOG INC, (agosto 1979).

3.19 - Zilog, "Z8010 MMU Memory Management Unit, Product Specification", ZILOG INC, (octubre 1979).

3.20 - ELECTRONIC DESIGN, "Annual Microprocessor Special", vol. 29, No. 24, pp. 114-175 (nov. 26, 1981).

3.21 Bal, S., Kaminker, A., Lavi, Y., Menacnem, A. y Sona, Z., "The NS16000 Family, Advances in Architecture and Hardware", COMPUTER, vol. 15, No. 6, pp. 58-67 (junio 1982).

3.22 Toong, H.D. y Gupta, A., "An Architectural Comparison of Contemporary 16-bit Microprocessors", IEEE-MICRO, vol. 1, No. 2, pp. 26-37 (mayo 1981).

3.24 - ELECTRONIC DESIGN, "Microprocessor Data Manual", vol. 28, No. 24, pp. 107-208 (nov. 22, 1980).

3.25 Noyce, R.N. y Hoff, M.E.Jr., "A History of Microprocessor Development at Intel", IEEE MICRO, vol. 1, No. 1, pp. 8-21 (febr. 1981).

3.26 - Zilog, "Segmented vs. Linear Addressing Z8000 vs. 68000, Concept paper", ZILOG INC., 9 pag. (nov. 1980).

3.27 Abraham, J. y Nadur, S.P., "Comparison of the Z8000 and the MC68000 microprocessors (A Soft point of view)", TATA INSTITUTE OF FUNDAMENTAL RESEARCH, BOMBAY-INDIA, Tecnnical Report No. 47, 17 pag. (dic. 1979).

3.28 Fairclough, D.A., "A Unique Microprocessor Instruction Set", IEEE MICRO, vol. 2, No. 2, pp. 8-18 (mayo 1982).

3.29 Best, D.W., Kress, C.E., Mykris, N.M., Russell, J.D. y Smith, W.J., "An Advanced-Architecture CMOS/SOS Microprocessor", IEEE-MICRO, vol.2, No.3, pp. 10-26 (agosto 1982).

3.30 Peñarrieta, L.H. y Lyons, L., "An Image Processing System", 1981 IEEE COMP. SOC. WORKSHOP ON CAPAIDM, pp. 295-300 (nov. 1981).

3.31 Collins, D.L. y Collins, C.A., "Memory-management chip

masters large data bases", ELECTRONIC DESIGN, pp. 115-121 (agosto 20, 1981).

3.32 Mateosian, R., "Segmentation advances uC memory addressing",ELECTRONIC DESIGN, pp. 155-162 (febrero 19, 1981).

3.33 Lavi, Y., Kaminker, A., Menachem, A. y Bal, S., "16-bit microprocessor enters virtual memory domain", ELECTRONICS,pp. 123-129 (abril 24, 1980).

CAPITULO 4

LIBROS

4.1 Triebel, Walter A., y Chu, Alfred E., "Handbook of semiconductor and Bubbles Memories", PRENTICE-HALL INC., Englewood Cliffs, N.J., EUA (1982).

4.2 Proebster, Walter E., "Digital Memory and Storage", VIEWEG, BRAUNSCHWEIG, Boblingen, Alemania Federal (1978).

4.3 Matick, Richard, "Computer Storage Systems and Technology",WILEY INTERSCIENCE, Nueva York (1977).

4.4 Luecke, G., Mize, J.P. y Carr, W.N., "Semiconductor Memory Design and Application", MCGRAW-HILL KOGAKUSHA, Tokyo (1973).

4.5 Marilyn, Bohl, "Introduction to IBM Direct Access Storage Devices", SCIENCE RESEARCH ASSOCIATES, INC., USA (1981).

4.6 Chu Yaohan, "Computer Organization and Microprogramming", PRENTICE-HALL INC., Englewood Cliffs, N.J., EUA (1972), Capitulo 7.

4.7 Kane Jerry, "An introduction to Microcomputers, Some Real Support Devices", Vol. 3, OSBORNE AND ASSOCIATES, INC., Berkeley, CA., USA (1978), Section A.

4.8 - "Memory Design Handbook", INTEL CO., (1975).

MEMORIA RAM

4.9 Sud, K. y Hardee, K.C., "16-K static RAM takes new route

to high speed", ELECTRONICS, pp. 117-123 (septiembre 11, 1980).

4.10 Onzone, T., "64-K static RAM surrounds n-MOS cells with C-MOS circuits", ELECTRONICS, pp. 145-148 (noviembre 6, 1980).

MEMORIA ROM

4.11 Greene, Bob, "Application of the Intel 2708 8K Erasable PROM", INTEL CO., APPLICATION NOTE AP-17 (1976).

4.12 Desrochers, G., "EEPROM eclipses other reprogrammable memories", ELECTRONIC DESIGN, pp. 247-250 (noviembre 22, 1980).

4.13 Bursky, D., "UV EPROMs and EEPROMs crash speed and density limits", ELECTRONIC DESIGN, pp. 55-66 (noviembre 22, 1980).

APLICACIONES RAM-ROM

4.14 Lavi, Y., Kaminker, A., Menachem, A. y Bal, S., "16-bit microprocessor enters virtual memory domain", ELECTRONICS, pp. 123-129 (abril 24, 1980).

4.15 Buzen, J.P. y Goldberg, R.P., "Virtual Machine Techniques for Introducing Peripherals into Computer Systems", COMPCON 74 COMPUTER PERIPHERALS, 8a. IEEE CONF., pp. 157-159 (febrero 1974).

4.16 Pohm, A.V., Agrawal, O.P. y Monroe, R.N., "The Cost and Performance Tradeoffs of Buffered Memories", PROCEEDINGS OF THE IEEE, vol. 63, No. 8, pp. 1129-1135 (agosto 1975).

FUNDAMENTOS DE GRABACION MAGNETICA

4.17 Mallinson, J.C., "Tutorial Review of Magnetic Recording", PROCEEDINGS OF THE IEEE, vol. 64, No. 2, pp. 196-208 (febr. 1976).

4.18 Schneidewind, N. y Syms, G., "Mass Memory System Peripherals", COMPCON 74 COMPUTER PERIPHERALS, 8a. IEEE CONF., pp. 87-91, (febrero 1974).

4.19 Knoll, A.L., "Spectrum Analysis of Digital Magnetic Recording Waveforms", IEEE TRANS. ON ELECTRON. COMPUT., vol LC-16, No.6, pp. 732-743 (diciembre 1967).

4.20 Franchini, R.C. y Wartner, D.L., "A Method of High Density Recording on Flexible Magnetic Discs", COMPUTER DESIGN, pp.106-109 (octubre 1976).

4.21 Patel, A.M., "New Method for Magnetic Encoding Combines Advantages of Older Techniques", COMPUTER DESIGN, pp. 85-91 (agosto 1976).

4.22 Sidnu, P.S., "Group-Coded Recoraing Reliably Doubles Diskette Capacity", COMPUTER DESIGN, pp. 84-88 (diciembre 1976).

CINTA MAGNETICA

4.23 Sallet, H.W., "A Magnetic Tape: A High performer", IEEE SPECTRUM, (julio 1977).

4.24 Rodriguez, J.A., "An Analysis of Tape Drive Technology", PROCEEDINGS OF THE IEEE, vol. 63, No. 8 (agosto 1975).

DISCOS MAGNETICOS

4.25 White, R.M., "Disk-Storage Technology", SCIENTIFIC AMERICAN, pp. 112-121 (noviembre 1980).

4.26 Manuel, Tom, "The Hard-Disk Explosion", BYTE, (agosto 1980).

4.27 Naughton, R.B., "An Overview of Disk Storage Systems", PROCEEDINGS OF THE IEEE, vol. 63, No. 8, pp. 1148-1152 (agosto 1975).

TENDENCIAS EN MEMORIAS

4.28 Bloch, E. y Galage, D., "Component Progress: Its Effect on High-Speed Computer Architecture and Machine Organization" COMPUTER, pp. 64-76 (abril 1976).

4.29 Hodges, D.A., "A Review and Projection of Semiconductor Components for Digital Storage", PROCEEDINGS OF THE

IEEE, Vol. 63, No. 8, pp. 1136-1147 (agosto 1975).

4.30 Bursky, D., "Special Report: Memories pace systems growth", ELECTRONIC DESIGN, pp. 63-76 (septiembre 27, 1980).

4.31 Posa, J.G., "Memories", ELECTRONICS, pp. 132-145 (octubre 23, 1980).

IMPRESORAS

4.32 Hewitt, V. y King, D., "Choosing a line printer", MINI-MICRO SYSTEMS (enero 1981).

CAPITULO 5

5.1 Barden, William. "The Z80 Microcomputer Handbook" Howard W. Sams & Co., Inc. 4th edition 1980.

5.2 Levethal, Lance A. "Z80 ASSEMBLY LANGUAGE PROGRAMMING", OSBORNE/McGraw Hill 1979

5.3 ZILOG, "Assembler Reference Manual", ZILOG.

CAPITULO 6

6.1 Meyeowitz, N., y Van Dam, A. "Interactive Editing Systems: Part I, Part II" en Computing Surveys 14, 3 (Sept. 1982), 321-415.

6.2 Shneiderman, B. "Direct Manipulation: A Step Beyond Programming Languages" en Computer 16, 8 (Agosto 1983), 57-59.

CAPITULO 7

BASIC

7.1 Basic Programming Reference Manual, Apple Computer Inc., Cupertino Cal., 1981.

FORTH

7.2 R.G. Loeliger, Threaded Interpretive Languages, Byte Books, Peterborough NH, 1981.

CAPITULO 8

8.1 Lister, A.M.A "Fundamentals of Operating Systems" Springer-Verlag. 2nd edition.

8.2 Madnick, Stuart E. Donovan, John J. "Operating Systems". McGraw Hill, 1974.

8.3 Miller Alan R. "MASTERING CP/M". SYBEX, 1983.

8.4 Digital Research. "CP/M Reference Manual". Digital Research, 1978.

8.4 Digital Research. "CP/M 2.2 ALTERATION GUIDE". Digital Research, 1979.

CAPITULO 9

9.1 Martin, James. "Computer Data-base Organization". Prentice Hall. 1975

9.2 Gagie, Michael. Konenier, Gary J. Whinston, Andrew. "Data-Base Management Systems: Powerful Newcomers to Microcomputers". BYTE, Nov 1981.

9.3 Heintz, Carl. "Guide to Database System Software". INTERFACE AGE, Feb 1983.

MICROPROCESADORES Y MICROCOMPUTADORAS

ESTADO DEL ARTE Y REVOLUCION COMPUTACIONAL HITOS TECNOLOGICOS

O C T U B R E, 1984

Desarrollo/Tecnología

1

¡Presentado en IEEE COMPUTACION'81 Seminario y exhibición de equipos y sistemas

Estado del arte y revolución computacional hitos tecnológicos

1. Introducción

¿En que sentido podemos hablar de una revolución en la computación?

¿Cuáles son los principales avances en esta rama, cuáles las tendencias que podemos encontrar y cuáles son los problemas que han trabado y traban este desarrollo?

A estas y otras cuestiones trataremos de responder en este trabajo.

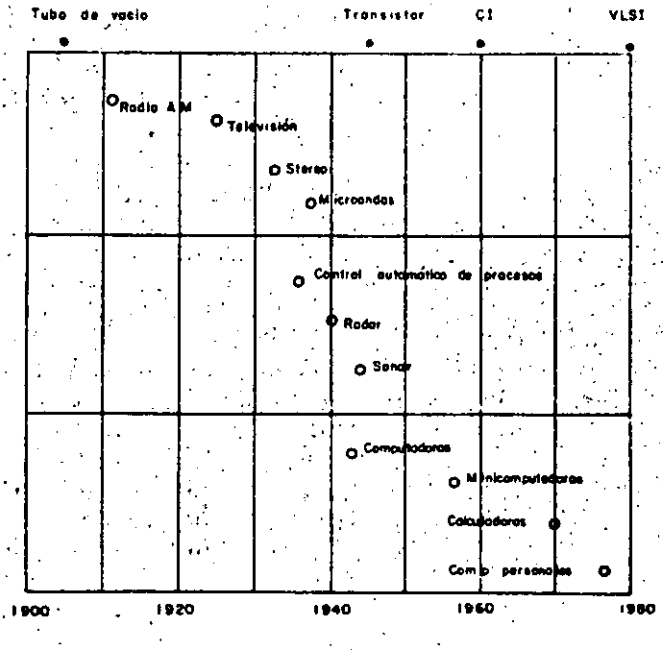
Según algunos economistas (sobre todo Mandel) después de la revolución industrial el mundo moderno ha atravesado por otras dos "revoluciones tecnológicas": aquella inaugurada por el cambio de la fuerza motriz de los motores de vapor a los motores de combustión interna y aquella que se verifica con el descubrimiento de la fusión nuclear controlada. Esta última iría acompañada por el gran desarrollo de la electrónica que hallaría aplicaciones en todo los ambitos de la vida diaria. (1)

Otros han tratado de definir las fases que abarcaría la tercera revolución tecnológica (como lo ha hecho Millman) y han subdividido la revolución electrónica en tres momentos sucesivos: la revolución en las comunicaciones, la revolución del control y la revolución de las computadoras (2). En la fig. 1 se puede apreciar estas tres fases del proceso.

Podemos decir ahora que hablamos de revolución computacional en un sentido más amplio de lo que el término refleja por si mismo: Entendemos por tal revolución un cambio *cualitativo* en el tipo de tecnología que se emplea hoy en día, cambio que va marcado por la penetración de la electrónica y la computación en todos los sectores de la producción y los servicios.

Una rápida mirada a ciertos hechos nos ayudaría a convencernos de lo dicho. En la fig. 2 podemos ver como ha descendido el precio de cierta "capacidad de cómputo constante", desde lo que eran las computadoras que se producian en 1955 y las calculadoras programables de hoy en día. Ambas tienen, aproximadamente, la misma capacidad de cálculo; pero el precio de esa capacidad era, aproximadamente 500 veces mayor en 1955. Nótese la escala logarítmica de la gráfica tres.

En las figs. 3 y 4 podemos ver, de una hojeada, cuál ha sido el desarrollo de los circuitos de las computadoras desde



LA REVOLUCION DE LOS COMPONENTES
Fig. 1

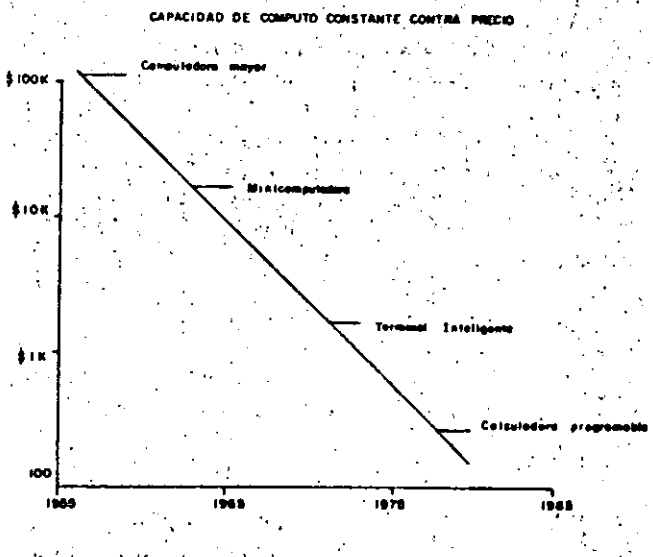


Fig. 2

Raúl Rojas González
Universidad Nacional Autónoma de México
Instituto Nacional de Investigaciones Nucleares

K el mismo bit costaba alrededor de 0.03 centavos de dólar, o sea, cinco veces menos. Nótese que la escala es logarítmica, lo que implica la caida exponencial en el costo por bit de los circuitos. Próximamente aparecerán las pastillas de más de 64 K, que seguramente ocuparán la prolongación de la gráfica.

COSTO DEL BIT DE MEMORIA

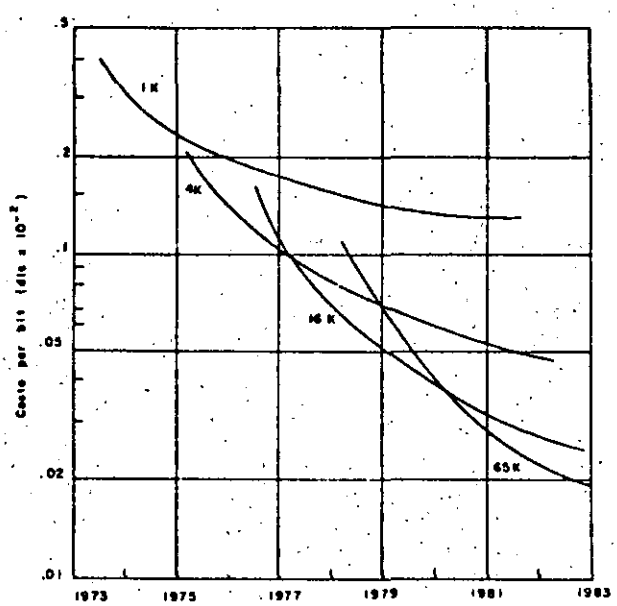


Fig. 13.

(Scientific American)

Cuando la compañía Intel introdujo el microprocesador en 1971 pocos se imaginaron el impacto que esto tendría. Hoy las microcomputadoras pueden ser usadas en todas partes; en el control de los semáforos, en los automóviles, en los instrumentos de medición, etc. Solamente del modelo 8080 de Intel se han vendido en todo el mundo un millón de unidades.

La utilización anual de componentes electrónicos ha crecido en los años posteriores a los sesentas al mismo ritmo que tuvo desde un principio. En la gráfica 14 podemos ver —otra vez a escala logarítmica— la curva de utilización de componentes contra el tiempo. Podemos ver que (aún con el margen de error que se puede suponer) para 1985 el número de componentes utilizados en todo el mundo se habrá multiplicado por 10 cuando menos. Se ha ido convirtiendo en realidad el antiguo sueño de popularizar las computadoras, al grado de que se han vuelto accesibles para gran cantidad de aplicaciones, aún para el uso personal. Se espera, por ejemplo, que la Tandy Corporation (que produce los sistemas TRS-80 de Radio Shack) anuncie próximamente una computadora portátil de 300 dólares (20) y en E.U. ya se vende la Sinclair ZX80 que es la primera microcomputadora completa de menos de 200 dólares (21). Se calcula también que para 1990 se habrán vendido ¡40 millones! de computadoras personales (22). Y muchas de estas pequeñas máquinas no tendrán nada que pedirle a los antiguos sistemas.

UTILIZACION ANUAL DE COMPONENTES ELECTRONICAS

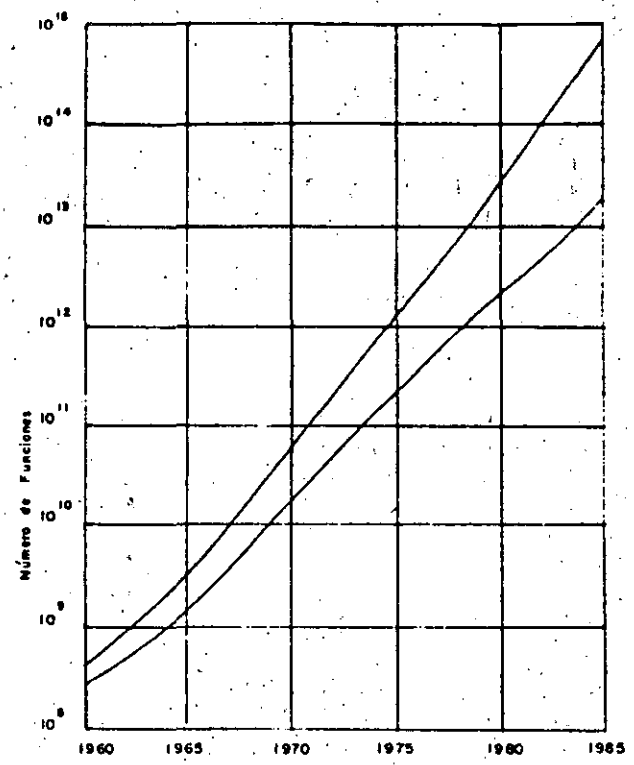


Fig. 14

(Scientific American)

III. Dos obstáculos a la revolución de las computadoras

Frente a esta situación de crecimiento exponencial de la complejidad de los circuitos y de extensión de la utilización de las computadoras hay dos obstáculos principales. El primero es un obstáculo "débil": la tecnología de impresión de los circuitos integrados. El segundo es un obstáculo "fuerte": el sofware asociado a las computadoras.

Con respecto a la primera cuestión el problema que se encuentra actualmente es que, a medida que aumenta la densidad de los circuitos, se requiere construirlos con líneas y elementos más y más finos, de manera que se ha llegado casi al punto en que las técnicas ópticas ya no son utilizables. En los próximos años deberán comenzar a utilizarse otras técnicas: de haz de electrónes o de rayos X.

En la figura 15 se puede ver esta cuestión. El problema sin embargo, no es insuperable ya que las nuevas técnicas se estan ajustando y es posible que en los próximos años comience su utilización en gran escala.

El segundo obstáculo es lo que se ha llamado el "cuello de botella" de la computación: el software. Resulta que mientrás el hardware se ha venido abaratando y haciendo más poderoso de manera exponencial, el software no ha podido ni siquiera acercarse a este ritmo de desarrollo. Es tan dramático el problema que ya actualmente el costo de toda la vida de un sistema de cómputo está constituido en casi un 80% por gastos de programación.

En la Fig. 16 se puede ver como esta situación tendrá a agudizarse en el futuro (23). De los costos de software, además, la mayor parte está constituida por los gastos de

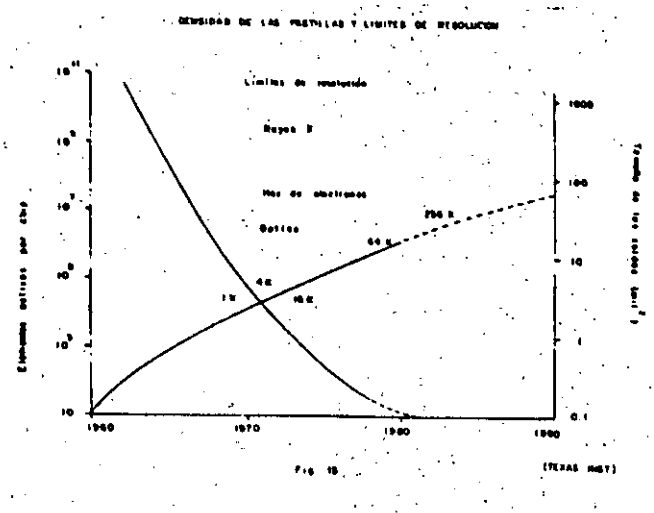


Fig. 15

mantenimiento, es decir, por la constante revisión y actualización de los sistemas.

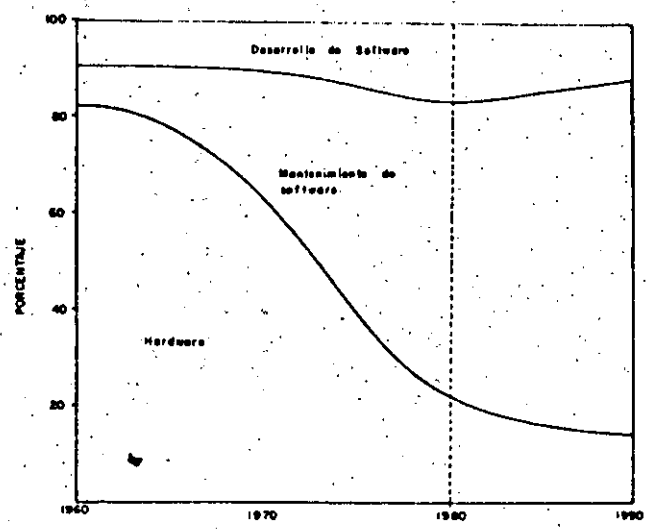


Fig. 16

El problema del software tiene un doble origen. En primer lugar los lenguajes de programación no han avanzado hasta el punto en que se cuente con un lenguaje módular, de alto nivel y que minimice los errores de programación. Pascal es un avance en ese sentido, pero aún falta diseñar, crear nuevas técnicas de programación acordes con el desarrollo actual del hardware. En segundo lugar tenemos la carencia de personal capacitado en la progración. En la Fig. 17 se puede ver como aún en los Estados Unidos, el número de graduados en ciencias de la computación no es ni el 10% del personal total requerido en esta rama. Mientras el personal en software crecerá exponencialmente hasta 1990, llegando a 2.4 millones de personas, el número de graduados en computación crecerá linealmente y a una tasa muy baja. Aún en el caso de que todos los ingenieros que se gradúan de todas las especialidades en E. U. se dedicarán a la computación de aquí a 1990, no se alcanzaría a llenar la brecha. Como se vé, pués este es un problema mayor al que nos enfrentará el futuro.

Sobra decir aquí que la situación en cuanto a software es aún más dramática en nuestro propio país.

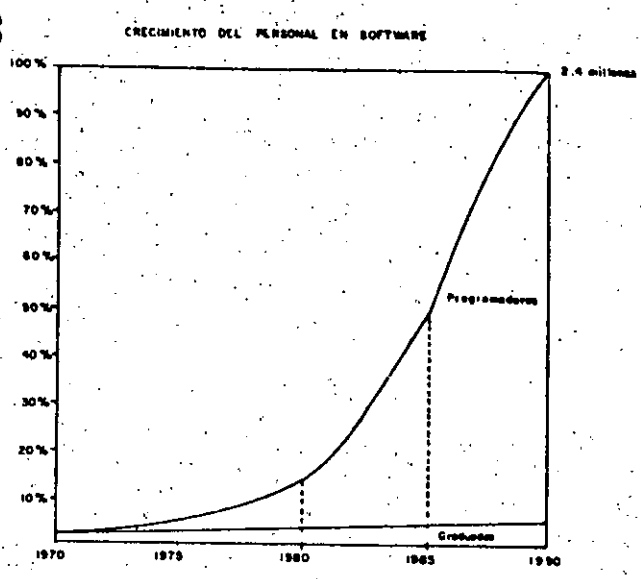


Fig. 17

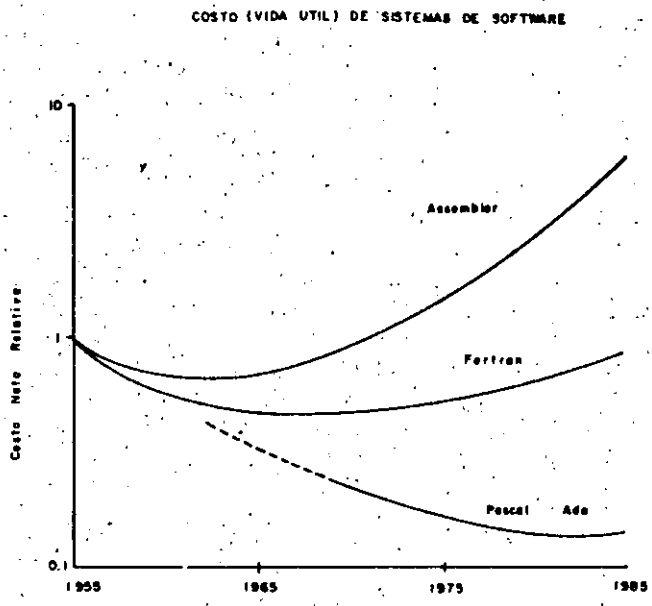


Fig. 18

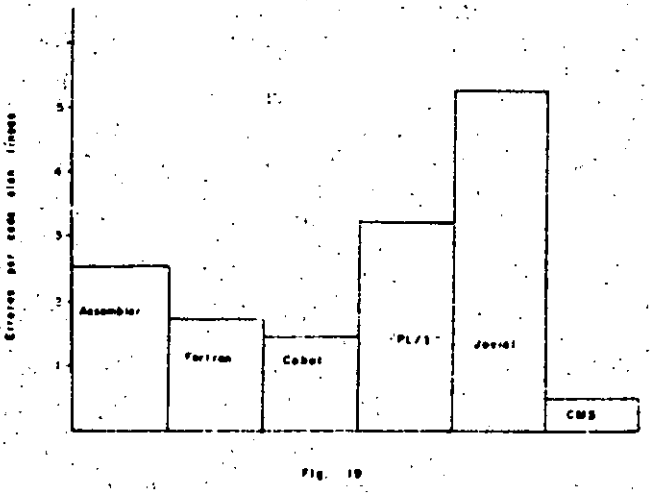
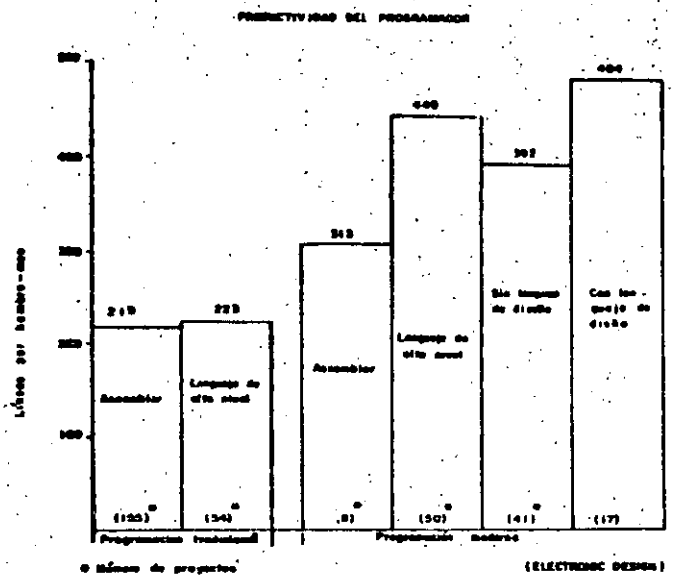


Fig. 19



IV. Los avances del futuro

Es difícil escribir en unas cuantas líneas acerca de los cambios y descubrimientos que podemos esperar en el futuro en el terreno de la computación. Sólo mencionaremos algunos ejemplos de la dirección que está tomando actualmente la investigación y que pueden servir de indicadores acerca de las tendencias del futuro.

Una de las cuestiones más importantes que actualmente se investigan es la forma de fabricar circuitos y microprocesadores más rápidos. El objetivo es producir computadoras digitales con un ciclo de un *nanosegundo*. Pero para lograr esto se necesitan componentes con velocidades mucho mayores a las actúales. Una forma de lograrlo es llevar la tecnología de la fotolitografía a su límite, lo que permitiría anchos de línea en los circuitos integrados de menos de 1 micra. Pero con estas dimensiones de los circuitos se pierde resolución en el método óptico y por eso se ha buscado sustituirlo por otro que realiza la impresión del circuito utilizando un haz de electrones. En la figura 21 se puede observar cómo ha ido disminuyendo el ancho de las líneas de los microcircuitos y la proyección que se hace para 1985, así como los límites de la tecnología óptica y la de haz de electrones. En la figura 22 se puede ver cuáles son los tiempos de ciclo típico en las computadoras actuales, siendo la computadora más veloz que hasta hoy existe la Cray-1 que ha logrado reducir el tiempo de ciclo a 12 nanosegundos (24).

Para alcanzar estos tiempos de ciclo increíbles de 1 nanosegundo en una computadora (que sería 50 veces más rápida que un sistema IBM 370) se está probando con diversos dispositivos. Uno de ellos son los llamados interruptores Josephson, que consisten de dos superconductores separados por una pequeña capa de material aislante que bajo ciertas condiciones permite el paso ("efecto tunel") de los electrones y cierra el circuito. Un interruptor de Josephson abierto representa un cero y uno cerrado representa un uno. La ventaja de estos dispositivos de superconducción es que son al menos diez veces más rápidos que los más rápidos dispositivos de semiconducción, pudiendo cambiar de estado en menos de 6 picosegundos (6×10^{-12} seg). Además los circuitos de superconducción generan menos calor que los de semiconducción de tal manera que se puedan empacar para formar computadoras pequeñísimas sin el peligro de que el calor disipado las funda. Según los investigadores de

4 IBM que más han estudiado este tipo de dispositivo, la primera computadora de superconducción será un cubo de dos pulgadas de arista y tendrá un tiempo de ciclo de dos nanosegundos. (25).

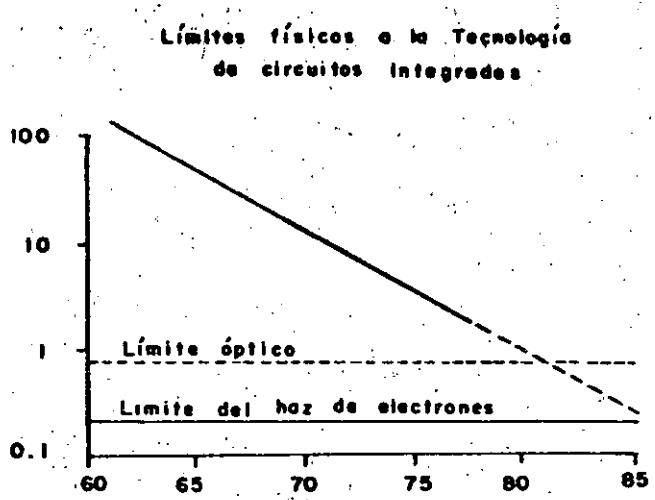


Fig. 21

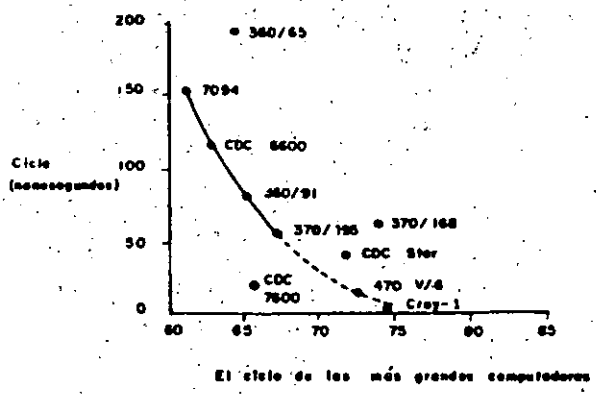


Fig. 22

(Computer)

La desventaja de los circuitos de Josephson es que deben operar a temperaturas cercanas al cero absoluto y para ello se les debe enfriar con helio líquido que resulta costoso. Una alternativa a esta dificultad es la utilización de los recién descubiertos superconductores orgánicos, que alcanzan la superconductividad a temperaturas más altas que los metales y necesitan ser enfriados con nitrógeno líquido, que es mucho más barato que el helio (26). Se continúa investigando para tratar de encontrar super conductores que funcionen a 20° arriba del cero absoluto y que podrían emplearse en las computadoras o en cualquier aparato eléctrico.

Otro intento para producir circuitos más rápidos es el llevado a cabo por Fujitsu, la compañía de computadoras más grande de Japón, que utilizando la tecnología de semiconducción e introduciendo leves modificaciones, afirma haber producido dispositivos, casi tan rápidos como los interruptores de Josephson y que funcionan a la temperatura del nitrógeno líquido (27). Este hecho los haría muy atractivos para los fabricantes de equipo de cómputo.

La tercera alternativa que está siendo estudiada es la tecnología óptica. En la Heriot-Watt-University de Edinburhg en Inglaterra ya se prueban dispositivos digitales que pueden cambiar de estado utilizando medios ópticos, en un pisosegundo. Eso los convertiría en aparatos siete o seis veces más rápidos que los interruptores de Josephson y que no necesitan operar a bajas temperaturas para ser funcionales (28). Hay sin embargo pocas noticias acerca de este tipo de circuitos, aunque hay que destacar que la tecnología óptica ha comenzado a ganar terreno en muchas aplicacio-

Velocidad de las computadoras de monoprocesador

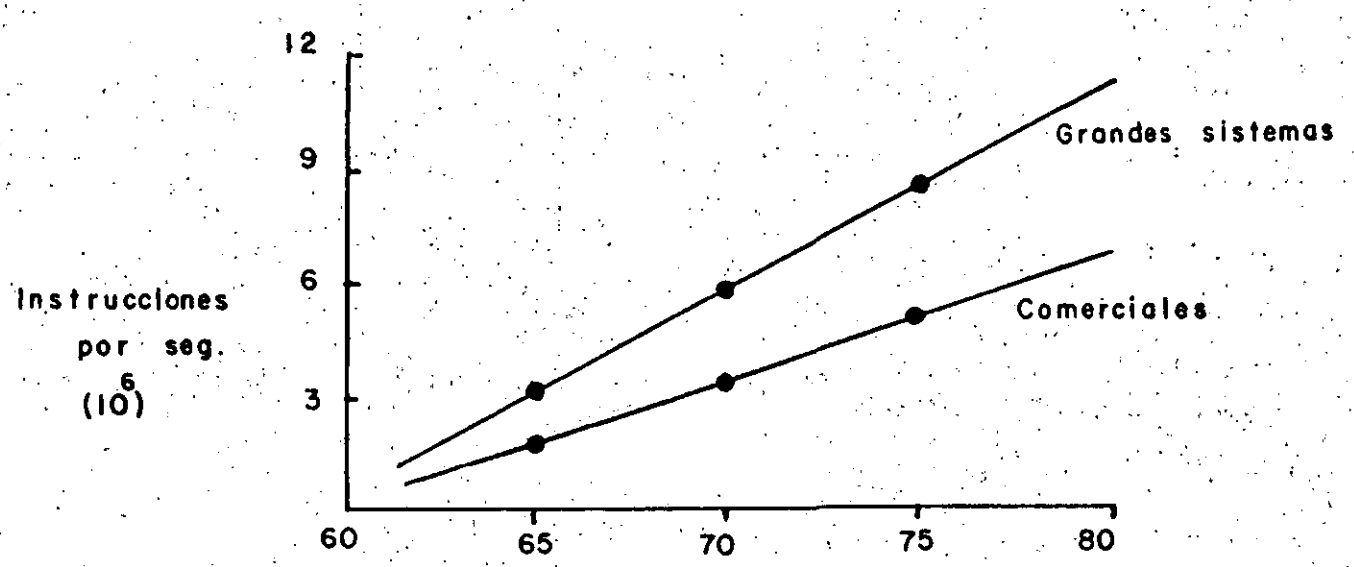


Fig. 23

Velocidad de las computadoras de mono y multiprocesador

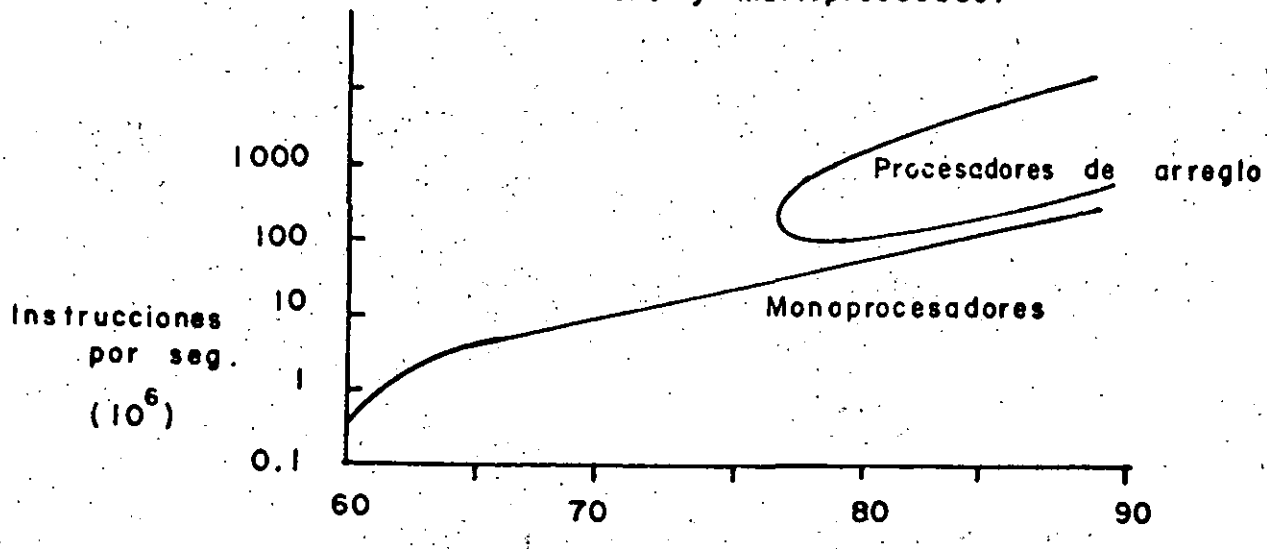


Fig. 24

(Computer)

nes, especialmente las de comunicaciones.

A través de una fibra óptica se pueden enviar muchos más mensajes que por las líneas convencionales.

¿Qué se impondrá? ¿La tecnología de superconducción, la óptica o la de Fujitsu? Los próximos años lo dirán, pero es evidente que cualquiera sea el resultado nos estamos aproximando al sueño de los antiguos investigadores: la construcción de computadoras poderosísimas y además portátiles. Las figuras 23 y 24 muestran como ha progresado la velocidad de las computadoras hasta 1980, y la previsión que se hace para el futuro con la nueva tecnología de los procesadores paralelos y de arreglo (array processors). (29).

Otro terreno en el que se están logrando avances considerables es en el almacenamiento de la información en grandes volúmenes reduciendo a la vez el tiempo de acceso a los dispositivos de almacenamiento masivo (bulk storage). Hasta ahora se ha intentado construir nuevos aparatos que elevan por un factor de 10^4 la capacidad de almacenamiento de la memoria RAM y aunque son más lentas que ésta, son más rápidos que los discos convencionales por un factor de 100 o de 1000. La figura 25 muestra esta situación. Recientemente se ha comenzado a investigar las posibilidades que ofrecen los medios ópticos para el almacenamiento de la información, utilizando rayos laser para grabar y recuperar a ésta. El resultado que se obtiene es un incremento por un factor de 10 en la densidad del empaque de la información sobre los mejores discos magnéticos. (30).

Estos avances son significativos si se piensa que una de las operaciones que más tiempo consumen en la computadora es la lectura y escritura a los dispositivos de memoria masiva. De nada serviría una computadora con un ciclo de

un nanosegundo y que estuviera atada a la lentitud del resto de los dispositivos. La tendencia para los próximos años, en cuanto a la memoria masiva, debe ser incrementar la densidad del almacenamiento a la vez que se reduce el tiempo de acceso. Por ahora la tecnología óptica parece una sólida promesa.

¿Qué utilidad tendrían estas computadoras más veloces y más compactas? ¿Es realmente necesario un procesador con un ciclo de 1 nanosegundo? Hay dos aplicaciones que en el futuro tenderán a ganar terreno y que podrían beneficiarse de los avances en la dirección antérior: la construcción de robots y la elaboración de máquinas con inteligencia artificial.

Hoy en día los robots ya forman parte de las líneas de montaje de numerosas plantas armadoras, como las de Texas instrument o General Motors. En los próximos cinco años, sin embargo, se cuadruplicará la utilización de los robots, hasta llegar a constituir una "fuerza de trabajo" mundial (traducimos literalmente el término workforce) de 60,000 aparatos (31). General Motors puso últimamente a prueba un robot de la compañía Unimate que puede ensamblar el 95% de las piezas de un automóvil; y decidió comenzar a instalar varios miles de ellos (32). En la compañía General Electric hay rumores de que en los próximos años serán sustituidos 15,000 obreros por robots de uso más o menos general. ¿Realidad o ficción? lo más probable es que las condiciones técnicas para una transición de este tipo estén ya dadas o a punto de darse, lo que tal vez pudiera ser un freno a esta tendencia a la automatización serían los factores de tipo económico. Y es que el mayor gasto de

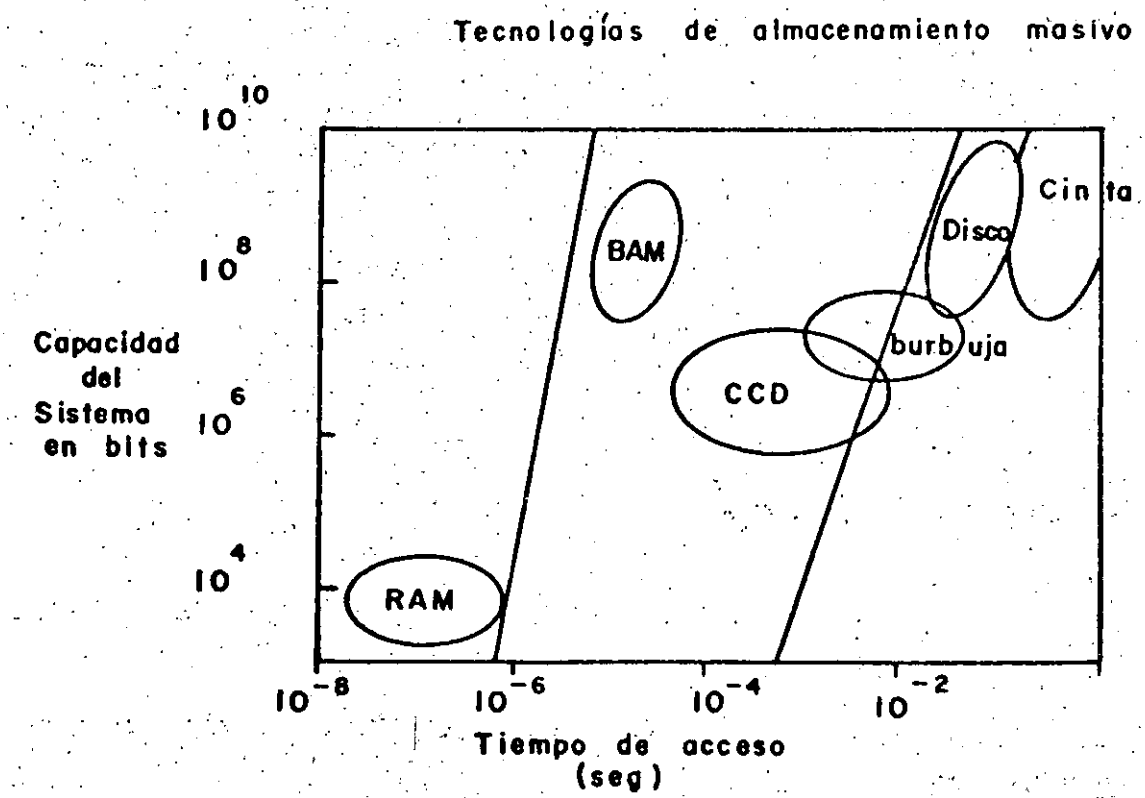


Fig. 25

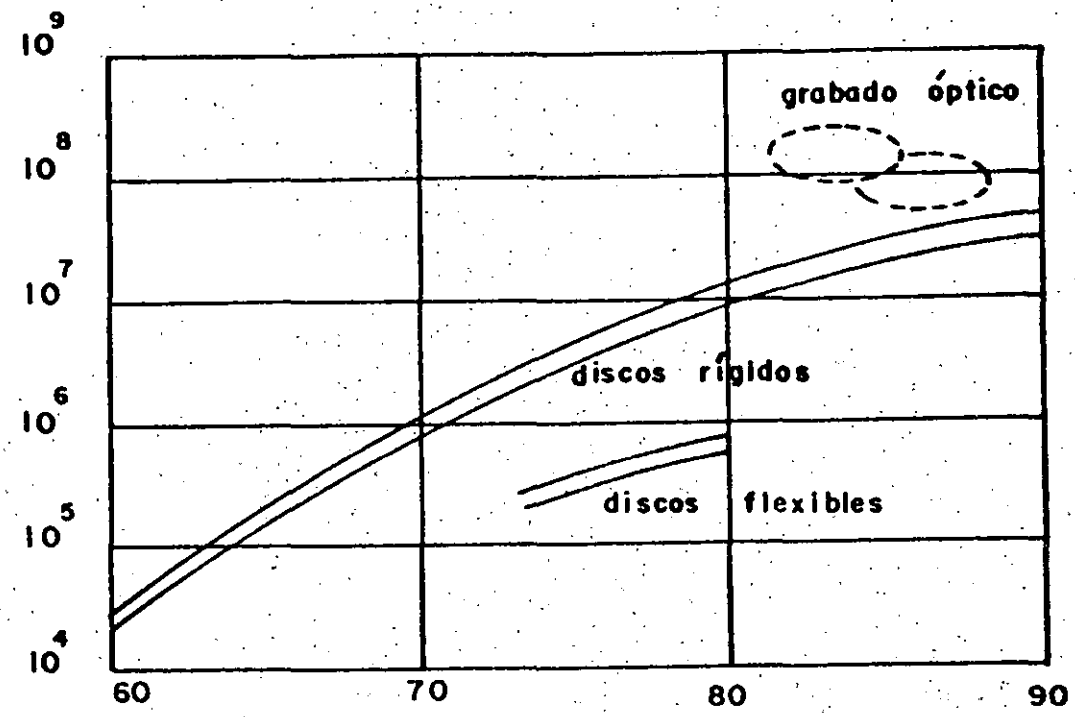


Fig. 26

(Scientific American)

capital involucrado, el aumento de su intensidad es un problema que permanentemente aqueja al capitalismo, que lo conduce a coyunturar cíclicas y que no sólo no es resuelto por la automatización, sino agravado por ella como ha demostrado Henryk Grossmann.

El principal problema con los robots es como enseñarlos a reconocer patrones (es decir objetos) de manera que puedan ejecutar operaciones diversas (33). Este problema es el más general involucrado con los intentos de "crear inteligencia artifical". Una buena parte de las manipulaciones mentales humanas tiene que ver con este proceso de reconocimiento de patrones, lo mismo cuando se percibe una impresión a través de la vista que cuando intenta demostrarse un teorema. Hasta ahora los intentos por producir máquinas inteligentes han logrado algunos resultados que serán obviamente mayores en el futuro a medida que se ponen a punto los algoritmos de reconocimiento de patrones y los procesadores que pueden hacerlos suficientemente rápidos. Cuando esto suceda las computadoras comenzarán a abordar problemas hasta antes sólo solubles mediante la intervención humana. Mucho del trabajo en este sentido está aún por realizarse, pero es éste uno de los campos que más atención han atraido actualmente, al grado de que ya existen microprocesadores capaces de jugar al ajedrez con un muy buen nivel o pequeños robots controlados por computadoras personales que pueden resolver problemas de laberintos.

POBLACION MUNDIAL DE ROBOTS

PAIS	CANTIDAD
JAPON	47,000
ALEMANIA FEDERAL	5,850
ESTADOS UNIDOS	3,255
GRAN BRETAÑA	185
POLONIA	720
BELGICA	20
SUECIA	570
NORUEGA	200
FINLANDIA	130

(SPECTRUM)

Todo esto es cosa del futuro, pero el futuro está a la vuelta de la esquina y a nuestra generación la ha tocado presenciar cómo se llega a él. La revolución electrónica sigue en marcha. ★

DIVISION DE EDUCACION CONTINUA
FACULTAD DE INGENIERIA U.N.A.M.

MICROPROCESADORES Y MICROCOMPUTADORAS

Z8420
Z80 PIO Parallel
Input/Output Controller

O C T U B R E, 1984

Palacio de Minería Calle de Tacuba 5 primer piso Deleg. Cuauhtemoc 06000 México, D.F. Tel.: 521-40-20 Apdo. Postal M-2285

Z8420
Z80® PIO Parallel Input/Output Controller

Zilog

Product Specification

March 1981

Features

- Provides a direct interface between Z-80 microcomputer systems and peripheral devices.
- Both ports have interrupt-driven handshake for fast response.
- Four programmable operating modes: byte input, byte output, byte input/output (Port A only), and bit input/output.
- Programmable interrupts on peripheral status conditions.
- Standard Z-80 Family bus-request and prioritized interrupt-request daisy chains implemented without external logic.
- The eight Port B outputs can drive Darlington transistors (1.5 mA at 1.5 V).

General Description

The Z-80 PIO Parallel I/O Circuit is a programmable, dual-port device that provides a TTL-compatible interface between peripheral devices and the Z-80 CPU. The CPU configures the Z-80 PIO to interface with a wide range of peripheral devices with no other external logic. Typical peripheral devices that are compatible with the Z-80 PIO include most keyboards, paper tape readers and punches, printers, PROM programmers, etc.

One characteristic of the Z-80 peripheral controllers that separates them from other interface controllers is that all data transfer between the peripheral device and the CPU is accomplished under interrupt control. Thus, the interrupt logic of the PIO permits full use of the efficient interrupt capabilities of the Z-80 CPU during I/O transfers. All logic necessary to implement a fully nested interrupt structure is included in the PIO.

Another feature of the PIO is the ability to interrupt the CPU upon occurrence of specified status conditions in the peripheral device. For example, the PIO can be programmed to interrupt if any specified peripheral alarm conditions should occur. This interrupt capability reduces the time the processor must spend in polling peripheral status.

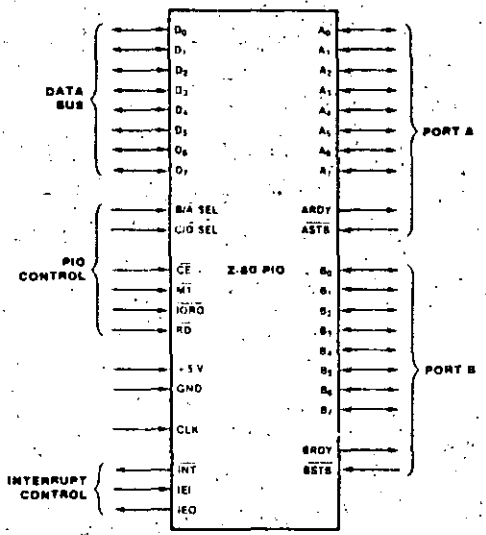


Figure 1. Pin Functions

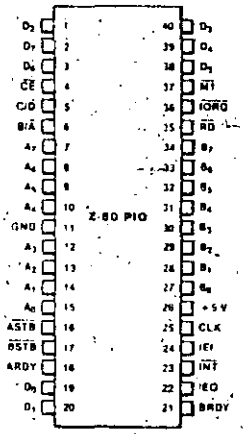


Figure 2. Pin Assignments

General Description (Continued)

The Z-80 PIO interfaces to peripherals via two independent general-purpose I/O ports, designated Port A and Port B. Each port has eight data bits and two handshake signals, Ready and Strobe, which control data transfer. The Ready output indicates to the peripheral that the port is ready for a data transfer. Strobe is an input from the peripheral that indicates when a data transfer has occurred.

Operating Modes. The Z-80 PIO ports can be programmed to operate in four modes: byte output (Mode 0); byte input (Mode 1), byte input/output (Mode 2) and bit input/output (Mode 3).

In Mode 0, either Port A or Port B can be programmed to output data. Both ports have output registers that are individually addressed by the CPU; data can be written to either port at any time. When data is written to a port, an active Ready output indicates to the external device that data is available at the associated port and is ready for transfer to the external device. After the data transfer, the external device responds with an active Strobe input, which generates an interrupt, if enabled.

In Mode 1, either Port A or Port B can be configured in the input mode. Each port has an input register addressed by the CPU. When the CPU reads data from a port, the PIO sets the Ready signal, which is detected by the external device. The external device then places data on the I/O lines and strobes the I/O port, which latches the data into the Port Input Register, resets Ready, and triggers the Interrupt Request, if enabled. The CPU can read the input data at any time, which again sets Ready.

Mode 2 is bidirectional and uses Port A, plus the interrupts and handshake signals from both ports. Port B must be set to Mode 3 and masked off. In operation, Port A is used for both data input and output. Output operation is similar to Mode 0 except that data is allowed out onto the Port A bus only when $\overline{ASTB}$ is Low. For input operation is similar to Mode 1, except that the data input uses the Port B handshake signals and the Port B interrupt (if enabled).

Both ports can be used in Mode 3. In this mode, the individual bits are defined as either input or output bits. This provides up to eight separate, individually defined bits for each port. During operation, Ready and Strobe are not used. Instead, an interrupt is generated if the condition of one input changes, or if all inputs change. The requirements for generating an interrupt are defined during the programming operation; the active level is specified as either High or Low, and the logic condition is specified as either one input active (OR) or all inputs active (AND). For example, if the port is programmed for active Low inputs and the logic function is AND, then all inputs at the specified port must go Low to generate an interrupt.

Data outputs are controlled by the CPU and can be written or changed at any time.

- Individual bits can be masked off.
- The handshake signals are not used in Mode 3; Ready is held Low, and Strobe is disabled.
- When using the Z-80 PIO interrupts, the Z-80 CPU interrupt mode must be set to Mode 2.

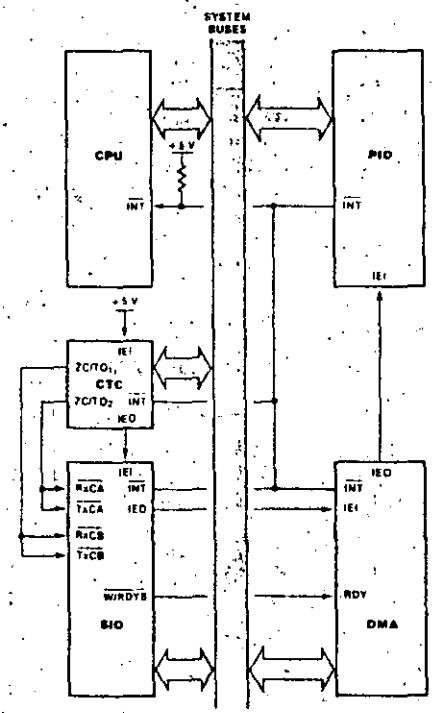


Figure 3. PIO in a Typical Z80 Family Environment

Internal Structure

The internal structure of the Z-80 PIO consists of a Z-80 CPU bus interface, internal control logic, Port A I/O logic, Port B I/O logic, and interrupt control logic (Figure 4). The CPU bus interface logic allows the Z-80 PIO to interface directly to the Z-80 CPU with no other external logic. The internal control logic synchronizes the CPU data bus to the peripheral device interfaces (Port A and Port B). The two I/O ports (A and B) are virtually identical and are used to interface directly to peripheral devices.

Port Logic. Each port contains separate input and output registers, handshake control logic, and the control registers shown in Figure 5. All data transfers between the peripheral unit and the CPU use the data input and output registers. The handshake logic associated with each port controls the data transfers through the input and the output registers. The mode control register (two bits) selects one of the four programmable operating modes.

The control mode (Mode 3) uses the remaining registers. The input/output control register specifies which of the eight data bits in the port are to be outputs and enables these bits; the remaining bits are inputs. The mask register and the mask control register control Mode 3 interrupt conditions. The mask register specifies which of the bits in the port are active and which are masked or inactive.

The mask control register specifies two conditions: first, whether the active state of the input bits is High or Low, and second, whether an interrupt is generated when any *one* unmasked input bit is active (OR condition) or if the interrupt is generated when *all* unmasked input bits are active (AND condition).

Interrupt Control Logic. The interrupt control logic section handles all CPU interrupt protocol for nested-priority interrupt structures. Any device's physical location in a daisy-chain configuration determines its priority. Two lines (IEI and IEO) are provided in each PIO to form this daisy chain. The device closest to the CPU has the highest priority. Within a PIO, Port A interrupts have higher priority than those of Port B. In the byte input, byte output, or bidirectional modes, an interrupt can be generated whenever the peripheral requests a new byte transfer. In the bit control mode, an interrupt can be generated when the peripheral status matches a programmed value. The PIO provides for complete control of nested interrupts. That is, lower priority devices may not interrupt higher priority devices that have not had their interrupt service routines completed by the CPU. Higher priority devices may interrupt the servicing of lower priority devices.

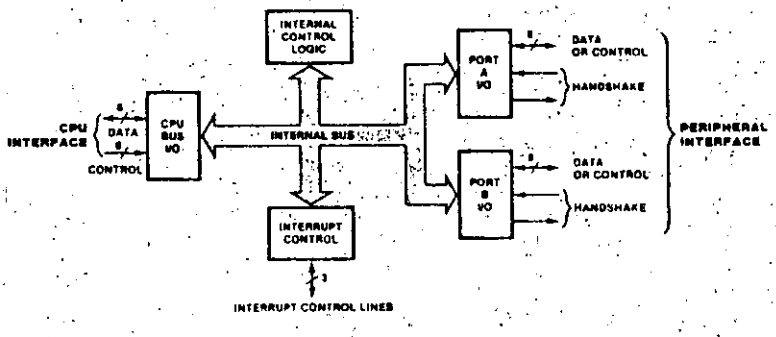


Figure 4. Block Diagram

Internal Structure (Continued)

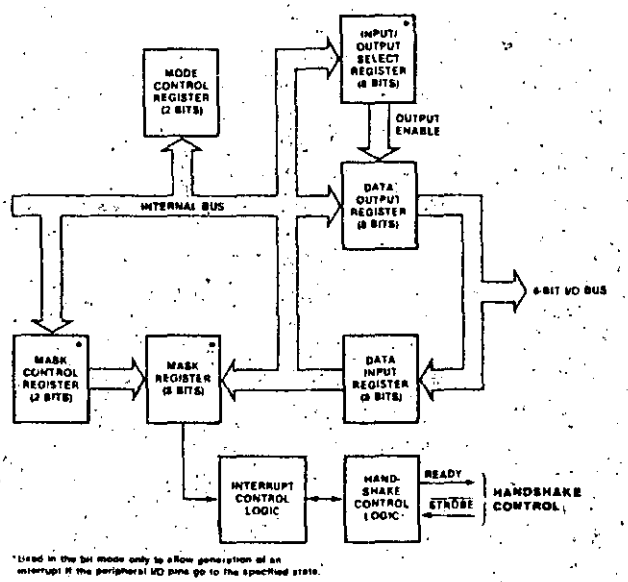
If the CPU (in Interrupt Mode 2) accepts an interrupt, the interrupting device must provide an 8-bit interrupt vector for the CPU. This vector forms a pointer to a location in memory where the address of the interrupt service routine is located. The 8-bit vector from the interrupting device forms the least significant eight bits of the indirect pointer while the I Register in the CPU provides the most significant eight bits of the pointer. Each port (A and B) has an independent interrupt vector. The least significant bit of the vector is automatically set to 0 within the PIO because the pointer must point to two adjacent memory locations for a complete 16-bit address.

Unlike the other Z-80 peripherals, the PIO does not enable interrupts immediately after programming. It waits until $\overline{M1}$ goes Low (e.g., during an opcode fetch). This condition is unimportant in the Z-80 environment but might not be if another type of CPU is used.

The PIO decodes the RETI (Return From Interrupt) instruction directly from the CPU data bus so that each PIO in the system knows at all times whether it is being serviced by the CPU interrupt service routine. No other communication with the CPU is required.

CPU Bus I/O Logic. The CPU bus interface logic interfaces the Z-80 PIO directly to the Z-80 CPU, so no external logic is necessary. For large systems, however, address decoders and/or buffers may be necessary.

Internal Control Logic. This logic receives the control words for each port during programming and, in turn, controls the operating functions of the Z-80 PIO. The control logic synchronizes the port operations, controls the port mode, port addressing, selects the read/write function, and issues appropriate commands to the ports and the interrupt logic. The Z-80 PIO does not receive a write input from the CPU; instead, the $\overline{RD}$, $\overline{CE}$, $C/\overline{D}$ and $\overline{IORQ}$ signals generate the write input internally.



*Used in the bit mode only to allow generation of an interrupt if the peripheral I/O pins go to the specified state.

Figure 5. Typical Port I/O Block Diagram

Programming

Mode 0, 1, or 2. *(Byte Input, Output, or Bidirectional).* Programming a port for Mode 0, 1, or 2 requires two words per port. These words are:

A Mode Control Word. Selects the port operating mode (Figure 6). This word may be written any time.

An Interrupt Vector. The Z-80 PIO is designed for use with the Z-80 CPU in Interrupt Mode 2 (Figure 7). When interrupts are enabled, the PIO must provide an interrupt vector.

Mode 3. *(Bit Input/Output).* Programming a port for Mode 3 operation requires a control word, a vector (if interrupts are enabled), and three additional words, described as follows:

I/O Register Control. When Mode 3 is selected, the mode control word must be followed by another control word that sets the I/O control register, which in turn defines which port lines are inputs and which are outputs (Figure 8).

Interrupt Control Word. In Mode 3, handshake is not used. Interrupts are generated as a logic function of the input signal levels. The interrupt control word sets the logic conditions and the logic levels required for generating an interrupt. Two logic conditions or functions are available: AND (if all input bits change to the active level, an interrupt is triggered), and OR (if any one of the input bits changes to the active level, an interrupt is triggered). Bit D_6 sets the logic function, as shown in Figure 9. The active level of the input bits can be set either High or Low. The active level is controlled by Bit D_5.

Mask Control Word. This word sets the mask control register, allowing any unused bits to be masked off. If any bits are to be masked, then D_4 must be set. When D_4 is set, the next word written to the port must be a mask control word (Figure 10).

Interrupt Disable. There is one other control word which can be used to enable or disable a port interrupt. It can be used without changing the rest of the interrupt control word (Figure 11).

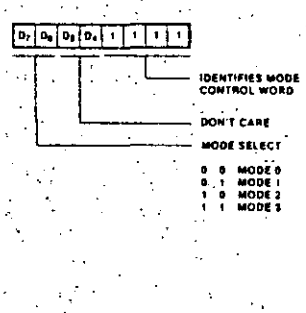


Figure 8. Mode Control Word

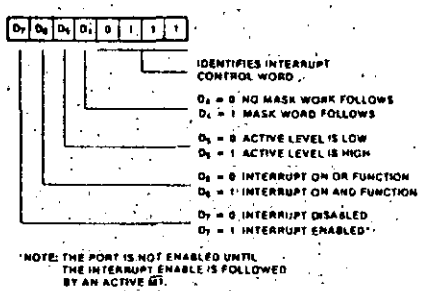


Figure 9. Interrupt Control Word

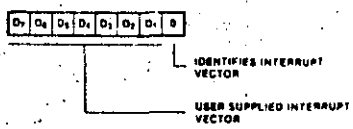


Figure 7. Interrupt Vector Word

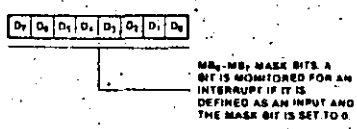


Figure 10. Mask Control Word

Figure 8. I/O Register Control Word

Figure 11. Interrupt Disable Word

Pin Description

A_0-A_7. *Port A Bus* (bidirectional, 3-state). This 8-bit bus transfers data, status, or control information between Port A of the PIO and a peripheral device. A_0 is the least significant bit of the Port A data bus.

ARDY. *Register A Ready* (output, active High). The meaning of this signal depends on the mode of operation selected for Port A as follows:

Output Mode. This signal goes active to indicate that the Port A output register has been loaded and the peripheral data bus is stable and ready for transfer to the peripheral device.

Input Mode. This signal is active when the Port A input register is empty and ready to accept data from the peripheral device.

Bidirectional Mode. This signal is active when data is available in the Port A output register for transfer to the peripheral device. In this mode, data is not placed on the Port A data bus, unless $\overline{ASTB}$ is active.

Control Mode. This signal is disabled and forced to a Low state.

$\overline{ASTB}$. *Port A Strobe Pulse From Peripheral Device* (input, active Low). The meaning of this signal depends on the mode of operation selected for Port A as follows:

Output Mode. The positive edge of this strobe is issued by the peripheral to acknowledge the receipt of data made available by the PIO.

Input Mode. The strobe is issued by the peripheral to load data from the peripheral into the Port A input register. Data is loaded into the PIO when this signal is active.

Bidirectional Mode. When this signal is active, data from the Port A output register is gated onto the Port A bidirectional data bus. The positive edge of the strobe acknowledges the receipt of the data.

Control Mode. The strobe is inhibited internally.

B_0-B_7. *Port B Bus* (bidirectional, 3-state). This 8-bit bus transfers data, status, or control information between Port B and a peripheral device. The Port B data bus can supply 1.5 mA at 1.5 V to drive Darlington transistors. B_0 is the least significant bit of the bus.

B/$\overline{A}$. *Port B Or A Select* (input, High = B). This pin defines which port is accessed during a data transfer between the CPU and the PIO. A Low on this pin selects Port A; a High selects Port B. Often address bit A_0 from the CPU is used for this selection function.

BRDY. *Register B Ready* (output, active High). This signal is similar to ARDY, except that in the Port A bidirectional mode this signal is High when the Port A input register is empty and ready to accept data from the peripheral device.

$\overline{BSTB}$. *Port B Strobe Pulse From Peripheral Device* (input, active Low). This signal is similar to $\overline{ASTB}$, except that in the Port A bidirectional mode this signal strobes data from the peripheral device into the Port A input register.

C/$\overline{D}$. *Control Or Data Select* (input, High = C). This pin defines the type of data transfer to be performed between the CPU and the PIO. A High on this pin during a CPU write to the PIO causes the Z-80 data bus to be interpreted as a *command* for the port selected by the B/$\overline{A}$ Select line. A Low on this pin means that the Z-80 data bus is being used to transfer data between the CPU and the PIO. Often address bit A_1 from the CPU is used for this function.

$\overline{CE}$. *Chip Enable* (input, active Low). A Low on this pin enables the PIO to accept command or data inputs from the CPU during a write cycle or to transmit data to the CPU during a read cycle. This signal is generally decoded from four I/O port numbers for Ports A and B, data, and control.

CLK. *System Clock* (input). The Z-80 PIO uses the standard single-phase Z-80 system clock.

D_0-D_7. *Z-80 CPU Data Bus* (bidirectional, 3-state). This bus is used to transfer all data and commands between the Z-80 CPU and the Z-80 PIO. D_0 is the least significant bit.

IEI. *Interrupt Enable In* (input, active High). This signal is used to form a priority interrupt daisy chain when more than one interrupt-driven device is being used. A High level on this pin indicates that no other devices of higher priority are being serviced by a CPU interrupt service routine.

IEO. *Interrupt Enable Out* (output, active High). The IEO signal is the other signal required to form a daisy chain priority scheme. It is High only if IEI is High and the CPU is not servicing an interrupt from this PIO. Thus this signal blocks lower priority devices from interrupting while a higher priority device is being serviced by its CPU interrupt service routine.

$\overline{INT}$. *Interrupt Request* (output, open drain, active Low). When $\overline{INT}$ is active the Z-80 PIO is requesting an interrupt from the Z-80 CPU.

$\overline{IORQ}$. *Input/Output Request* (input from Z-80 CPU, active Low). $\overline{IORQ}$ is used in conjunction with B/$\overline{A}$, C/$\overline{D}$, $\overline{CE}$, and $\overline{RD}$ to transfer commands and data between the Z-80 CPU and the Z-80 PIO. When $\overline{CE}$, $\overline{RD}$, and $\overline{IORQ}$ are active, the port addressed by B/$\overline{A}$ transfers data to the CPU (a read operation). Conversely, when $\overline{CE}$ and $\overline{IORQ}$ are active but $\overline{RD}$ is not, the port addressed by B/$\overline{A}$ is written into from the CPU with either data or control information, as specified by C/$\overline{D}$. Also, if $\overline{IORQ}$ and $\overline{M1}$ are active simultaneously, the CPU is acknowledging an interrupt; the interrupting port automatically places its interrupt vector on the CPU data bus if it is the highest priority device requesting an interrupt.

Pin Description (Continued)

$\overline{M1}$, *Machine Cycle* (input from CPU, active Low). This signal is used as a sync pulse to control several internal PIO operations. When both the $\overline{M1}$ and $\overline{RD}$ signals are active, the Z-80 CPU is fetching an instruction from memory. Conversely, when both $\overline{M1}$ and $\overline{IORQ}$ are active, the CPU is acknowledging an interrupt. In addition, $\overline{M1}$ has two other functions within the Z-80 PIO: it synchronizes the PIO interrupt logic; when $\overline{M1}$ occurs without an active $\overline{RD}$ or $\overline{IORQ}$ signal, the PIO is reset.

7

$\overline{RD}$, *Read Cycle Status* (input from Z-80 CPU, active Low). If $\overline{RD}$ is active, or an I/O operation is in progress, $\overline{RD}$ is used with B/$\overline{A}$, C/$\overline{D}$, $\overline{CE}$, and $\overline{IORQ}$ to transfer data from the Z-80 PIO to the Z-80 CPU.

Timing

The following timing diagrams show typical timing in a Z-80 CPU environment. For more precise specifications refer to the composite ac timing diagram.

Write Cycle. Figure 12 illustrates the timing for programming the Z-80 PIO or for writing data to one of its ports. No Wait states are allowed for writing to the PIO other than the automatically inserted T_{WA}. The PIO does not receive a specific write signal; it internally generates its own from the lack of an active $\overline{RD}$ signal.

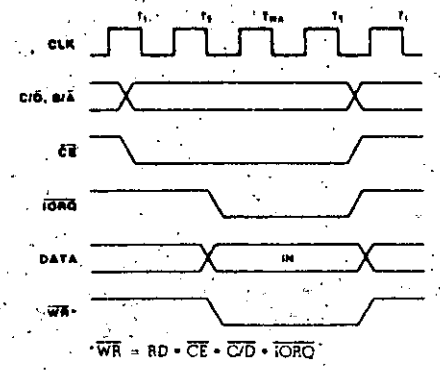


Figure 12. Write Cycle Timing

Read Cycle. Figure 13 illustrates the timing for reading the data input from an external device to one of the Z-80 PIO ports. No Wait states are allowed for reading the PIO other than the automatically inserted T_{WA}.

Output Mode (Mode 0). An output cycle (Figure 14) is always started by the execution of an output instruction by the CPU. The $\overline{WR}$* pulse from the CPU latches the data from the CPU data bus into the selected port's output register. The $\overline{WR}$* pulse sets the Ready flag after a Low-going edge of CLK, indicating data is available. Ready stays active until the positive edge of the strobe line is received, indicating that data was taken by the peripheral. The positive edge of the strobe pulse generates an $\overline{INT}$ if the interrupt enable flip-flop has been set and if this device has the highest priority.

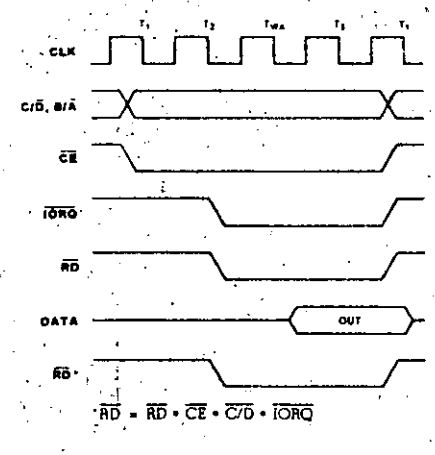


Figure 13. Read Cycle Timing

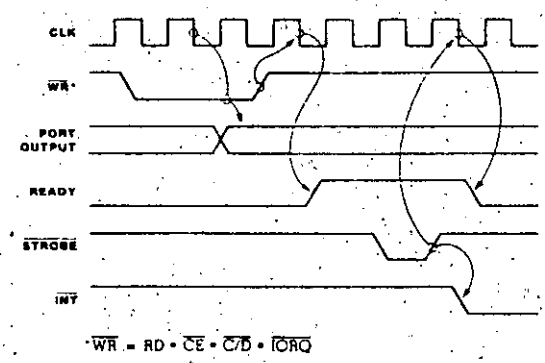


Figure 14. Mode 0 Output Timing

Timing (Continued)

Input Mode (Mode 1). When $\overline{STROBE}$ goes Low, data is loaded into the selected port input register (Figure 15). The next rising edge of strobe activates $\overline{INT}$, if Interrupt Enable is set and this is the highest-priority requesting device. The following falling edge of CLK resets Ready to an inactive state, indicating that the input register is full and cannot accept any more data until the CPU completes a read. When a read is complete, the positive edge of $\overline{RD}$ sets Ready at the next Low-going transition of CLK. At this time new data can be loaded into the PIO.

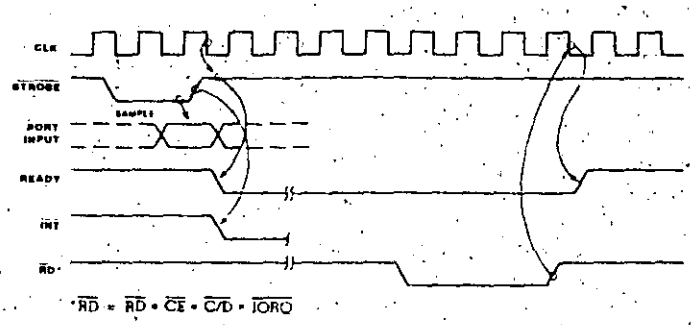


Figure 15. Mode 1 Input Timing

Bidirectional Mode (Mode 2). This is a combination of Modes 0 and 1 using all four handshake lines and the eight Port A I/O lines (Figure 16). Port B must be set to the bit mode and its inputs must be masked. The Port A handshake lines are used for output control and the Port B lines are used for input control. If interrupts occur, Port A's vector will be used during port output and Port B's will be used during port input. Data is allowed out onto the Port A bus only when $\overline{ASTB}$ is Low. The rising edge of this strobe can be used to latch the data into the peripheral.

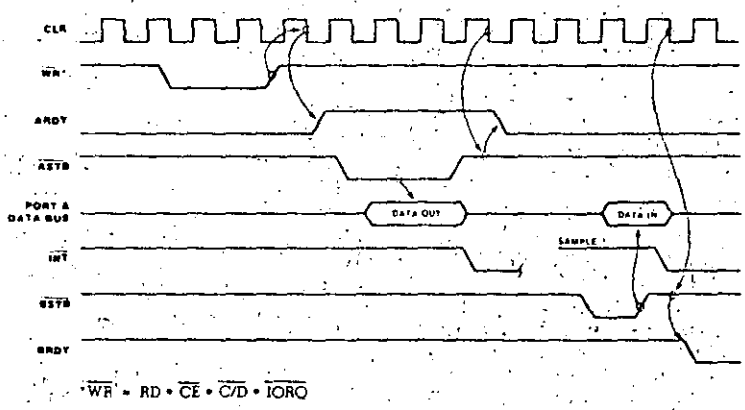


Figure 16. Mode 2 Bidirectional Timing

Timing (Continued)

Bit Mode (Mode 3). The bit mode does not utilize the handshake signals, and a normal port write or port read can be executed at any time. When writing, the data is latched into the output registers with the same timing as the output mode (Figure 17).

When reading the PIO, the data returned to the CPU is composed of output register data from those port data lines assigned as outputs and input register data from those port data lines assigned as inputs. The input register contains data that was present immediately prior to the falling edge of $\overline{RD}$. An interrupt is generated if interrupts from the port are enabled and the data on the port data lines satisfy the logical equation defined by the 8-bit mask and 2-bit mask control registers. However, if Port A is programmed in bidirectional mode, Port B does not issue an interrupt in bit mode and must therefore be polled.

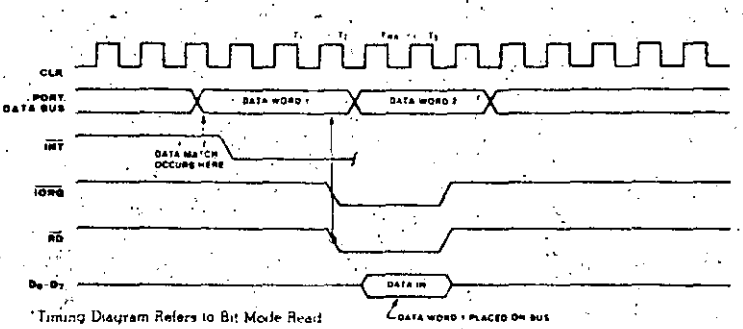


Figure 17. Mode 3 Bit Mode Timing

Interrupt Acknowledge Timing. During $\overline{M1}$ time, peripheral controllers are inhibited from changing their interrupt enable status, permitting the Interrupt Enable signal to ripple through the daisy chain. The peripheral with IEI High and IEO Low during $\overline{INTACK}$ places a preprogrammed 8-bit interrupt vector on the data bus at this time (Figure 18). IEO is held Low until a Return From Interrupt (RETI) instruction is executed by the CPU while IEI is High. The 2-byte RETI instruction is decoded internally by the PIO for this purpose.

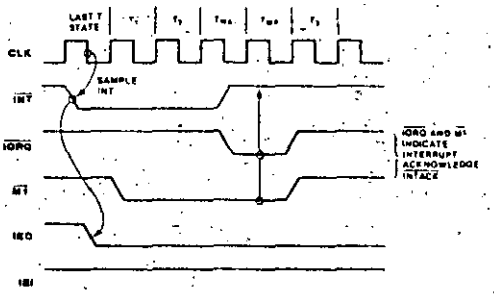


Figure 18. Interrupt Acknowledge Timing

Return From Interrupt Cycle. If a Z-80 peripheral has no interrupt pending and is not under service, then its IEO = IEI. If it has an interrupt under service (i.e., it has already interrupted and received an interrupt acknowledge) then its IEO is always Low, inhibiting lower priority devices from interrupting. If it has an interrupt pending which has not yet been acknowledged, IEO is Low unless an "ED" is decoded as the first byte of a 2-byte opcode (Figure 19). In this case, IEO goes High until the next opcode byte is decoded, whereupon it goes Low again. If the second byte of the opcode was a "4D," then the opcode was an RETI instruction.

After an "ED" opcode is decoded, only the peripheral device which has interrupted and is currently under service has its IEI High and its IEO Low. This device is the highest-priority device in the daisy chain that has received an interrupt acknowledge. All other peripherals have IEI = IEO. If the next opcode byte decoded is "4D," this peripheral device resets its "interrupt under service" condition.

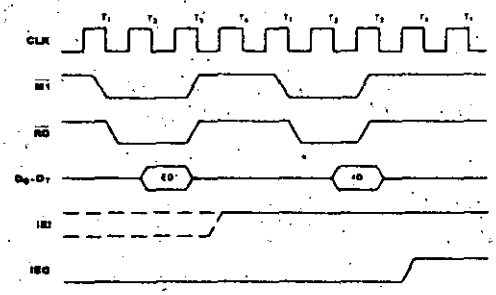
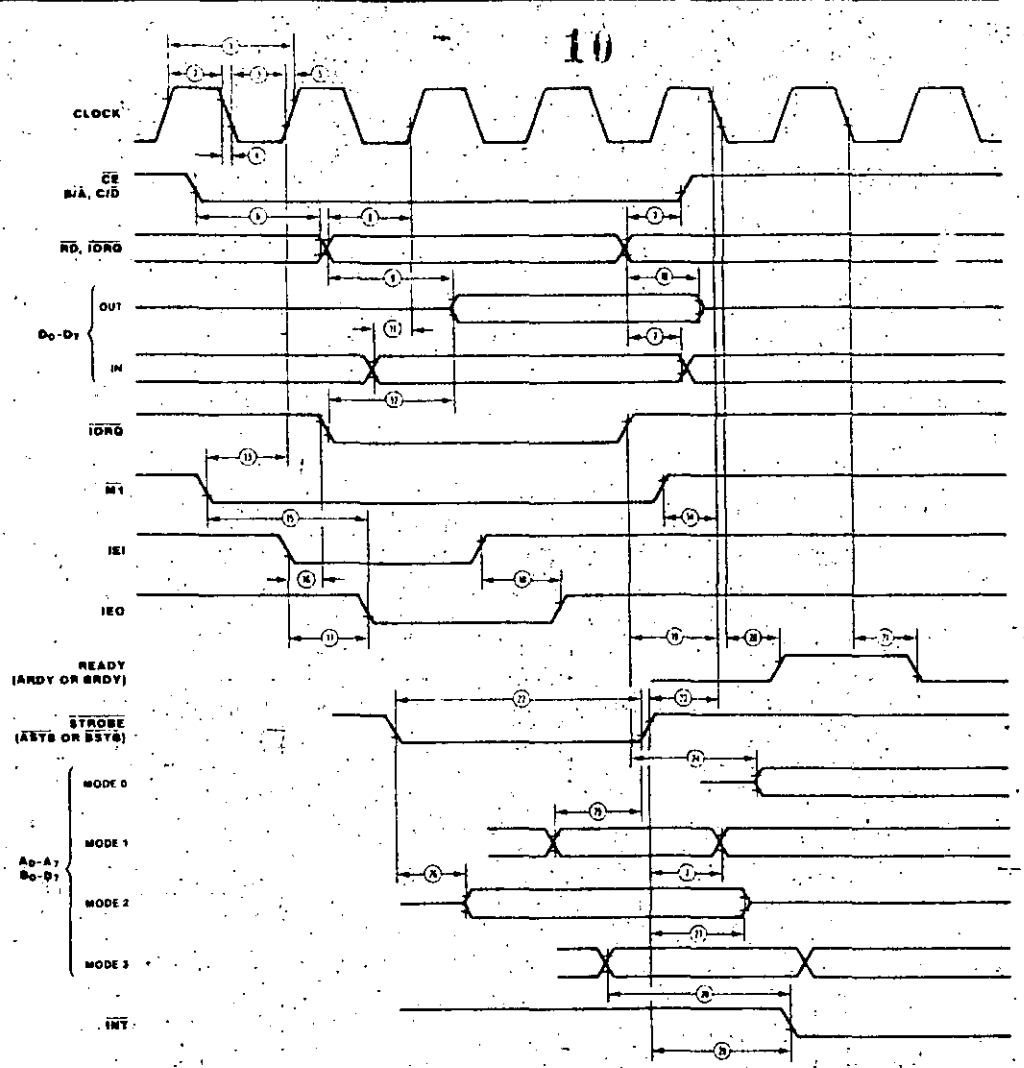


Figure 19. Return From Interrupt


CLOCK
CE
B/A, C/D
RD, IORQ
D0-D7
OUT
IN
IORQ
M1
IEI
IEO
READY
(ARDY OR BRDY)
STROBE
(ASTB OR BSTB)
A0-A7
B0-B7
MODE 0
MODE 1
MODE 2
MODE 3
INT

Number	Symbol	Parameter	Z-80 PIO Min (ns)	Z-80 PIO Max (ns)	Z-80A PIO Min (ns)	Z-80A PIO Max (ns)	Z-80B PIO[9] Min (ns)	Z-80B PIO[9] Max (ns)	Comment
1	TcC	Clock Cycle Time	400	[1]	250	[1]	165	[1]	
2	TwCh	Clock Width (High)	170	2000	105	2000	65	2000	
3	TwCl	Clock Width (Low)	170	2000	105	2000	65	2000	
4	TfC	Clock Fall Time		30		30		20	
5	TrC	Clock Rise Time		30		30		20	
6	TsCS(RI)	$\overline{CE}$, B/$\overline{A}$, C/$\overline{D}$ to $\overline{RD}$, $\overline{IORQ}$ ↓ Setup Time	50		50		50		[6]
7	Th	Any Hold Times for Specified Setup Time	0		0		0	0	
8	TsRI(C)	$\overline{RD}$, $\overline{IORQ}$ to Clock ↑ Setup Time	115		115		70		
9	TdRI(DO)	$\overline{RD}$, $\overline{IORQ}$ ↓ to Data Out Delay		430		380		300	[2]
10	TdRI(DOs)	$\overline{RD}$, $\overline{IORQ}$ ↑ to Data Out Float Delay		160		110		70	
11	TsDI(C)	Data In to Clock ↓ Setup Time	50		50		40		CL = 50 pF
12	TdIO(DOI)	$\overline{IORQ}$ ↓ to Data Out Delay (INTACK Cycle)	340		160		120		[3]
13	TsM1(Cr)	$\overline{M1}$ ↓ to Clock ↑ Setup Time	210		90		70		
14	TsM1(Cf)	$\overline{M1}$ ↓ to Clock ↓ Setup Time (M1 Cycle)	0		0		0		[8]
15	TdM1(IEO)	$\overline{M1}$ ↓ to IEO ↓ Delay (Interrupt Immediately Preceding $\overline{M1}$ ↓)		300		190		100	[5, 7]
16	TsIEI(IO)	IEI to $\overline{IORQ}$ ↓ Setup Time (INTACK Cycle)	140		140		100		[7]
17	TdIEI(IEOf)	IEI ↓ to IEO ↓ Delay		190		130		120	[5] CL = 50 pF
18	TdIEI(IEOr)	IEI ↑ to IEO ↑ Delay (after ED Decode)		210		160		160	[5]
19	TcIO(C)	$\overline{IORQ}$ ↑ to Clock ↑ Setup Time (To Activate READY on Next Clock Cycle)	220		200		170		
20	TdC(RDYr)	Clock ↓ to READY ↑ Delay	200		190		170		[5] CL = 50 pF
21	TdC(RDYf)	Clock ↓ to READY ↓ Delay	150		140		120		[5]
22	TwSTB	$\overline{STROBE}$ Pulse Width	150		150		120		[4]
23	TsSTB(C)	$\overline{STROBE}$ ↓ to Clock ↓ Setup Time (To Activate READY on Next Clock Cycle)	220		220		150		[5]
24	TdIO(PD)	$\overline{IORQ}$ ↑ to PORT DATA Stable Delay (Mode 0)		200		180		160	[5]
25	TsPD(STB)	PORT DATA to $\overline{STROBE}$ ↑ Setup Time (Mode 1)	260		230		190		
26	TdSTB(PD)	$\overline{STROBE}$ ↓ to PORT DATA Stable (Mode 2)		230		210		180	[5]
27	TdSTB(PDr)	$\overline{STROBE}$ ↑ to PORT DATA Float Delay (Mode 2)		200		180		160	CL = 50 pF
28	TdPD(INT)	PORT DATA Match to $\overline{INT}$ ↓ Delay (Mode 3)		540		490		430	
29	TdSTB(INT)	$\overline{STROBE}$ ↑ to $\overline{INT}$ ↓ Delay		490		440		350	

NOTES:
[1] TcC = TwCh + TwCl + TrC + TfC.
[2] Increase TdRI(DO) by 10 ns for each 50 pF increase in load up to 200 pF max.
[3] Increase TdIO(DOI) by 10 ns for each 50 pF increase in loading up to 200 pF max.
[4] For Mode 2: TwSTB > TdSTB(STB).
[5] Increase these values by 2 ns for each 10 pF increase in loading up to 100 pF max.
[6] TsCS(RI) may be reduced. However, the time subtracted from TsCS(RI) will be added to TdRI(DO).
[7] 2.5 TcC > (N-2)TcIEI(IEO) + TdM1(IEO) + TsIEI(IO) + TTL Buffer Delay, if any.
[8] $\overline{M1}$ must be active for a minimum of two clock cycles to reset the PIO.
[9] Z80B PIO numbers are preliminary and subject to change.

Absolute Maximum Ratings

Voltages on all inputs and outputs with respect to GND ..... -0.3 V to +7.0 V

Operating Ambient Temperature ..... As Specified in Ordering Information

Storage Temperature ..... -65°C to +150°C

Stresses greater than those listed under Absolute Maximum Ratings may cause permanent damage to the device. This is a stress rating only; operation of the device at any condition above those indicated in the operational sections of these specifications is not implied. Exposure to absolute maximum rating conditions for extended periods may affect device reliability.

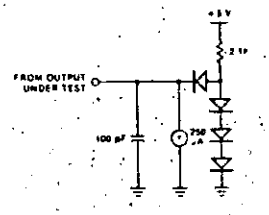
Test Conditions

The characteristics below apply for the following standard test conditions, unless otherwise noted. All voltages are referenced to GND (0 V). Positive current flows into the referenced pin. Available operating temperature ranges are:

- 0° to +70°C, $+4.75\ V \le V_{CC} \le +5.25\ V$
- -40°C to +85°C, $+4.75\ V \le V_{CC} \le +5.25\ V$
- -55° to +125°C, $+4.75\ V \le V_{CC} \le +5.5\ V$

The product number for each operating temperature range may be found in the Ordering Information section.

All ac parameters assume a load capacitance of 100 pF max. Timing references between two output signals assume a load difference of 50 pF max.



DC Characteristics

Symbol	Parameter	Min	Max	Unit	Test Condition
V_{ILC}	Clock Input Low Voltage	-0.3	+0.45	V	
V_{IHC}	Clock Input High Voltage	V_{CC}-0.6	+5.5	V	
V_{IL}	Input Low Voltage	-0.3	+0.8	V	
V_{IH}	Input High Voltage	+2.0	+5.5	V	
V_{OL}	Output Low Voltage		+0.4	V	I_{OL} = 2.0 mA
V_{OH}	Output High Voltage	+2.4		V	I_{OH} = -250 µA
I_{LI}	Input Leakage Current	-10.0	+10.0	µA	$0 \le V_{IN} \le V_{CC}$
I_Z	3-State Output/Data Bus Input Leakage Current	-10.0	+10.0	µA	$0 < V_{IN} < V_{CC}$
I_{CC}	Power Supply Current		100.0	mA	V_{OH} = 1.5V
I_{GHD}	Darlington Drive Current	-1.5	3.8	mA	R_{EXT} = 390 Ω

Over specified temperature and voltage range.

Capacitance

Symbol	Parameter	Min	Max	Unit	Test Condition
C	Clock Capacitance		10	pF	Unmeasured pins returned to ground
C_{IN}	Input Capacitance		5	pF	
C_{OUT}	Output Capacitance		10	pF	

Over specified temperature range; f = 1MHz

56

Ordering Information

Product Number	Package/ Temp	Speed	Description
Z8420	CE	2.5 MHz	Z80 PIO (40-pin)
Z8420	CM	2.5 MHz	Same as above
Z8420	CMB	2.5 MHz	Same as above
Z8420	CS	2.5 MHz	Same as above
Z8420	DE	2.5 MHz	Same as above
Z8420	DS	2.5 MHz	Same as above
Z8420	PE	4.0 MHz	Same as above
Z8420	PS	4.0 MHz	Same as above
Z8420A	CE	4.0 MHz	Z80A PIO (40-pin)
Z8420A	CM	4.0 MHz	Same as above
Z8420A	CMB	4.0 MHz	Same as above
Z8420A	CS	4.0 MHz	Same as above
Z8420A	DE	4.0 MHz	Z80A PIO (40-pin)
Z8420A	DS	4.0 MHz	Same as above
Z8420A	PE	4.0 MHz	Same as above
Z8420A	PS	4.0 MHz	Same as above
Z8420B	CE	6.0 MHz	Z80B PIO (40-pin)
Z8420B	CM	6.0 MHz	Same as above
Z8420B	CMB	6.0 MHz	Same as above
Z8420B	CS	6.0 MHz	Same as above
Z8420B	DE	6.0 MHz	Same as above
Z8420B	DS	6.0 MHz	Same as above
Z8420B	PE	6.0 MHz	Same as above
Z8420B	PS	6.0 MHz	Same as above

NOTES: C = Ceramic, D = Cerdip, P = Plastic; E = -40°C to +85°C, M = -55°C to +125°C, MB = 55°C to +125°C with MIL-STD-883 Class B processing, S = 0°C to +70°C.

Ordering Information

Product Number	Package/ Temp	Speed	Description
Z8400	CE	2.5 MHz	Z80 CPU (40-pin)
Z8400	CM	2.5 MHz	Same as above
Z8400	CMB	2.5 MHz	Same as above
Z8400	CS	2.5 MHz	Same as above
Z8400	DE	2.5 MHz	Same as above
Z8400	DS	2.5 MHz	Same as above
Z8400	PE	2.5 MHz	Same as above
Z8400	PS	2.5 MHz	Same as above
Z8400A	CE	4.0 MHz	Z80A CPU (40-pin)
Z8400A	CM	4.0 MHz	Same as above
Z8400A	CMB	4.0 MHz	Same as above
Z8400A	CS	4.0 MHz	Same as above
Z8400A	DE	4.0 MHz	Z80A CPU (40-pin)
Z8400A	DS	4.0 MHz	Same as above
Z8400A	PE	4.0 MHz	Same as above
Z8400A	PS	4.0 MHz	Same as above
Z8400B	CE	6.0 MHz	Z80B CPU (40-pin)
Z8400B	CM	6.0 MHz	Same as above
Z8400B	CMB	6.0 MHz	Same as above
Z8400B	CS	6.0 MHz	Same as above
Z8400B	DE	6.0 MHz	Same as above
Z8400B	DS	6.0 MHz	Same as above
Z8400B	PE	6.0 MHz	Same as above
Z8400B	PS	6.0 MHz	Same as above

NOTES: C = Ceramic, D = Cerdip, P = Plastic; E = -40°C to +85°C, M = -55°C to +125°C, MB = -55°C to +125°C with MIL-STD 883 Class B processing, S = 0°C to +70°C.

MICROPROCESADORES Y MICROCOMPUTADORAS

Z8400

Z80 CPU Central

Processing Unit

OCTUBRE, 1984.

Z8400
Z80® CPU Central Processing Unit

(3)

Zilog

Product Specification

March 1981

Features

- The instruction set contains 158 instructions. The 78 instructions of the 8080A are included as a subset; 8080A software compatibility is maintained.
- Six MHz, 4 MHz and 2.5 MHz clocks for the Z80B, Z80A, and Z80 CPU result in rapid instruction execution with consequent high data throughput.
- The extensive instruction set includes string, bit, byte, and word operations. Block searches and block transfers together with indexed and relative addressing result in the most powerful data handling capabilities in the microcomputer industry.
- The Z80 microprocessors and associated family of peripheral controllers are linked by a vectored interrupt system. This system may be daisy-chained to allow implementation of a priority interrupt scheme. Little, if any, additional logic is required for daisy-chaining.
- Duplicate sets of both general-purpose and flag registers are provided, easing the design and operation of system software through single-context switching, background-foreground programming, and single-level interrupt processing. In addition, two 16-bit index registers facilitate program processing of tables and arrays.
- There are three modes of high speed interrupt processing: 8080 compatible, non-Z80 peripheral device, and Z80 Family peripheral with or without daisy chain.
- On-chip dynamic memory refresh counter.

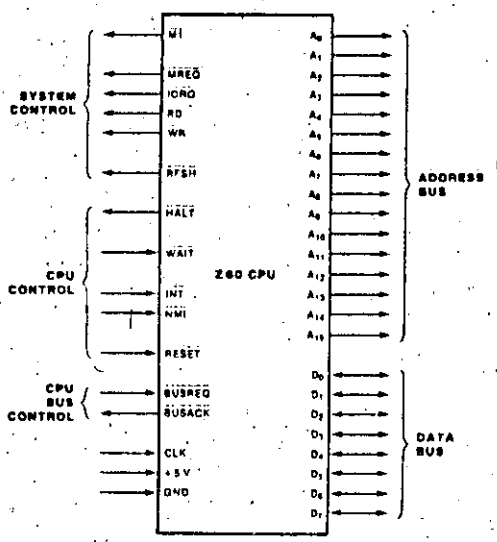


Figure 1. Pin Functions

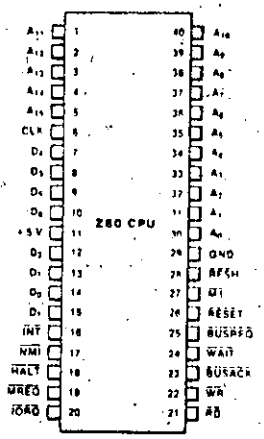


Figure 2. Pin Assignments

General Description

The Z80, Z80A, and Z80B CPUs are third-generation single-chip microprocessors with exceptional computational power. They offer higher system throughput and more efficient memory utilization than comparable second- and third-generation microprocessors. The internal registers contain 208 bits of read/write memory that are accessible to the programmer. These registers include two sets of six general-purpose registers which may be used individually as either 8-bit registers or as 16-bit register pairs. In addition, there are two sets of accumulator and flag registers. A group of "Exchange" instructions makes either set of main or alternate registers accessible to the programmer. The alternate set allows operation in foreground-background mode or it may be reserved for very fast interrupt response.

The Z80 also contains a Stack Pointer, Program Counter, two index registers, a Refresh register (counter), and an Interrupt register. The CPU is easy to incorporate into a system since it requires only a single +5 V power source, all output signals are fully decoded and timed to control standard memory or peripheral circuits, and is supported by an extensive family of peripheral controllers. The internal block diagram (Figure 3) shows the primary functions of the Z80 processors. Subsequent text provides more detail on the Z80 I/O controller family, registers, instruction set, interrupts and daisy chaining, and CPU timing.

(6)

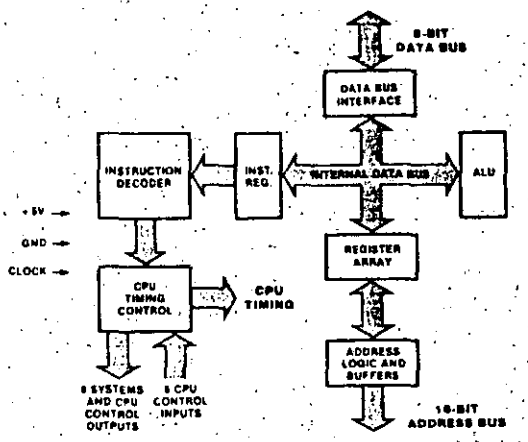


Figure 3. Z80 CPU Block Diagram

Z80 Microprocessor Family

3

The Zilog Z80 microprocessor is the central element of a comprehensive microprocessor product family. This family works together in most applications with minimum requirements for additional logic, facilitating the design of efficient and cost-effective microcomputer-based systems.

Zilog has designed five components to provide extensive support for the Z80 microprocessor. These are:

- The PIO (Parallel Input/Output) operates in both data-byte I/O transfer mode (with handshaking) and in bit mode (without handshaking). The PIO may be configured to interface with standard parallel peripheral devices such as printers, tape punches, and keyboards.
- The CTC (Counter/Timer Circuit) features four programmable 8-bit counter/timers, each of which has an 8-bit prescaler. Each of the four channels may be configured to operate in either counter or timer mode.
- The DMA (Direct Memory Access) controller provides dual port data transfer operations and the ability to terminate data transfer as a result of a pattern match.
- The SIO (Serial Input/Output) controller offers two channels. It is capable of operating in a variety of programmable modes for both synchronous and asynchronous communication, including Bi-Synch and SDLC.
- The DART (Dual Asynchronous Receiver/Transmitter) device provides low cost asynchronous serial communication. It has two channels and a full modem control interface.

Z80 CPU Registers

Figure 4 shows three groups of registers within the Z80 CPU. The first group consists of duplicate sets of 8-bit registers: a principal set and an alternate set (designated by ' [prime], e.g., A'). Both sets consist of the Accumulator Register, the Flag Register, and six general-purpose registers. Transfer of data between these duplicate sets of registers is accomplished by use of "Exchange" instructions. The result is faster response to interrupts and easy, efficient implementation of such versatile programming techniques as background-foreground data processing. The second set of registers consists of six registers with assigned functions. These are the I (Interrupt Register), the R (Refresh Register), the IX and IY (Index Registers), the SP (Stack Pointer), and the PC (Program Counter). The third group consists of two interrupt status flip-flops, plus an additional pair of flip-flops which assists in identifying the interrupt mode at any particular time. Table 1 provides further information on these registers.

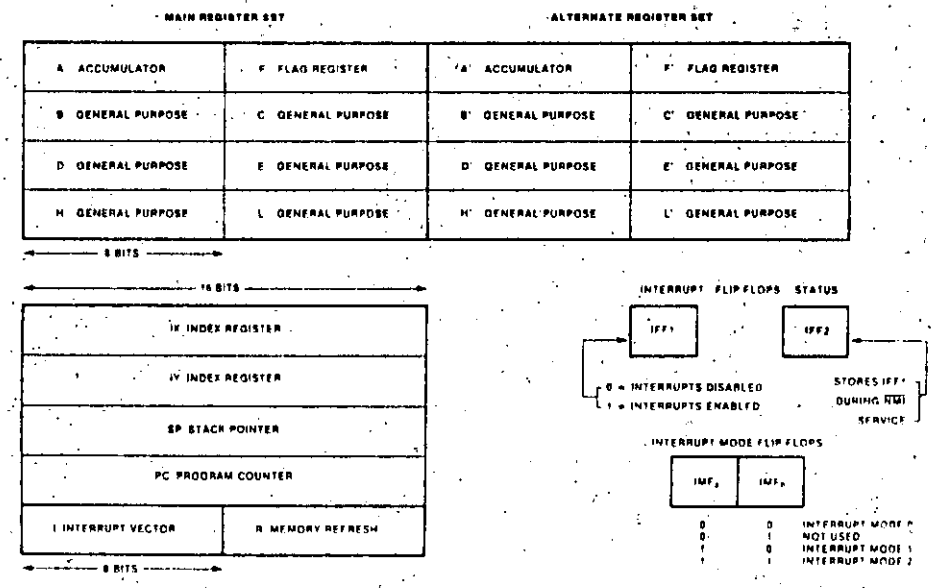


Figure 4. CPU Registers

Z80 CPU Registers (Continued)

Register		Size (Bits)	Remarks
A, A'	Accumulator	8	Stores an operand or the results of an operation.
F, F'	Flags	8	See Instruction Set.
B, B'	General Purpose	8	Can be used separately or as a 16-bit register with C.
C, C'	General Purpose	8	See B, above.
D, D'	General Purpose	8	Can be used separately or as a 16-bit register with E.
E, E'	General Purpose	8	See D, above.
H, H'	General Purpose	8	Can be used separately or as a 16-bit register with L.
L, L'	General Purpose	8	See H, above.
			Note: The (B,C), (D,E), and (H,L) sets are combined as follows: B — High byte C — Low byte D — High byte E — Low byte H — High byte L — Low byte
I	Interrupt Register	8	Stores upper eight bits of memory address for vectored interrupt processing.
R	Refresh Register	8	Provides user-transparent dynamic memory refresh. Automatically incremented and placed on the address bus during each instruction fetch cycle.
IX	Index Register	16	Used for indexed addressing.
IY	Index Register	16	Same as IX, above.
SP	Stack Pointer	16	Stores addresses or data temporarily. See Push or Pop in instruction set.
PC	Program Counter	16	Holds address of next instruction.
IFF_1-IFF_2	Interrupt Enable	Flip-Flops	Set or reset to indicate interrupt status (see Figure 4).
IMFa-IMFb	Interrupt Mode	Flip-Flops	Reflect Interrupt mode (see Figure 4).

Table 1. Z80 CPU Registers

Interrupts: General Operation

The CPU accepts two interrupt input signals: $\overline{NMI}$ and $\overline{INT}$. The $\overline{NMI}$ is a non-maskable interrupt and has the highest priority. $\overline{INT}$ is a lower priority interrupt since it requires that interrupts be enabled in software in order to operate. Either $\overline{NMI}$ or $\overline{INT}$ can be connected to multiple peripheral devices in a wired-OR configuration.

The Z80 has a single response mode for interrupt service for the non-maskable interrupt. The maskable interrupt, $\overline{INT}$, has three programmable response modes available. These are:

- Mode 0 — compatible with the 8080 microprocessor.
- Mode 1 — Peripheral Interrupt service, for use with non-8080/Z80 systems.
- Mode 2 — a vectored interrupt scheme, usually daisy-chained, for use with Z80 Family and compatible peripheral devices.

The CPU services interrupts by sampling the $\overline{NMI}$ and $\overline{INT}$ signals at the rising edge of the last clock of an instruction. Further interrupt service processing depends upon the type of interrupt that was detected. Details on interrupt responses are shown in the CPU Timing Section.

Non-Maskable Interrupt ($\overline{NMI}$). The non-maskable interrupt cannot be disabled by program control and therefore will be accepted at at all times by the CPU. $\overline{NMI}$ is usually reserved for servicing only the highest priority type interrupts, such as that for orderly shutdown after power failure has been detected. After recognition of the $\overline{NMI}$ signal (providing $\overline{BUSREQ}$ is not active), the CPU jumps to restart location 0066H. Normally, software starting at this address contains the interrupt service routine.

Maskable Interrupt ($\overline{INT}$). Regardless of the interrupt mode set by the user, the Z80 response to a maskable interrupt input follows a common timing cycle. After the interrupt has been detected by the CPU (provided that interrupts are enabled and $\overline{BUSREQ}$ is not active) a special interrupt processing cycle begins. This is a special fetch ($\overline{M1}$) cycle in which $\overline{IORQ}$ becomes active rather than $\overline{MREQ}$, as in a normal $\overline{M1}$ cycle. In addition, this special $\overline{M1}$ cycle is automatically extended by two $\overline{WAIT}$ states, to allow for the time required to acknowledge the interrupt request and to place the interrupt vector on the bus.

Mode 0 Interrupt Operation. This mode is compatible with the 8080 microprocessor interrupt service procedures. The interrupting device places an instruction on the data bus, which is then acted on six times by the CPU. This is normally a Restart Instruction, which will initiate an unconditional jump to the selected one of eight restart locations in page zero of memory.

Mode 1 Interrupt Operation. Mode 1 operation is very similar to that for the $\overline{NMI}$. The principal difference is that the Mode 1 interrupt has a vector address of 0038H only.

Mode 2 Interrupt Operation. This interrupt mode has been designed to utilize most effectively the capabilities of the Z80 microprocessor and its associated peripheral family. The interrupting peripheral device selects the starting address of the interrupt service routine. It does this by placing an 8-bit address vector on the data bus during the interrupt acknowledge cycle. The high-order byte of the interrupt service routine address is supplied by the I (Interrupt) register. This flexibility in selecting the interrupt service routine address allows the peripheral device to use several different types of service routines. These routines may be located at any available location in memory. Since the interrupting device supplies the low-order byte of the 2-byte vector, bit 0 (A_0) must be a zero.

Interrupt Priority (Daisy Chaining and Nested Interrupts). The interrupt priority of each peripheral device is determined by its physical location within a daisy-chain configuration. Each device in the chain has an interrupt enable input line (IEI) and an interrupt enable output line (IEO), which is fed to the next lower priority device. The first device in the daisy chain has its IEI input hardwared to a High level. The first device has highest priority, while each succeeding device has a corresponding lower priority. This arrangement permits the CPU to select the highest priority interrupt from several simultaneously interrupting peripherals.

The interrupting device disables its IEO line to the next lower priority peripheral until it has been serviced. After servicing, its IEO line is raised, allowing lower priority peripherals to demand interrupt servicing.

The Z80 CPU will nest (queue) any pending interrupts or interrupts received while a selected peripheral is being serviced.

Interrupt Enable/Disable Operation. Two flip-flops, IFF_1 and IFF_2, referred to in the register description are used to signal the CPU interrupt status. Operation of the two flip-flops is described in Table 2. For more details, refer to the *Z80 CPU Technical Manual* and *Z80 Assembly Language Manual.*

Action	IFF_1	IFF_2	Comments
CPU Reset	0	0	Maskable interrupt $\overline{INT}$ disabled
DI instruction execution	0	0	Maskable interrupt $\overline{INT}$ disabled
EI instruction execution	1	1	Maskable interrupt $\overline{INT}$ enabled
LD A,I instruction execution	•	•	IFF_2 = Parity flag
LD A,R instruction execution	•	•	IFF_2 = Parity flag
Accept $\overline{NMI}$	0	IFF_1	$IFF_1 = IFF_2$ (Maskable interrupt $\overline{INT}$ disabled)
RETN instruction execution	IFF_2	•	$IFF_2 = IFF_1$ at completion of an $\overline{NMI}$ service routine.

Table 2. State of Flip-Flops

Instruction Set

6

The Z80 microprocessor has one of the most powerful and versatile instruction sets available in any 8-bit microprocessor. It includes such unique operations as a block move for fast, efficient data transfers within memory or between memory and I/O. It also allows operations on any bit in any location in memory.

The following is a summary of the Z80 instruction set and shows the assembly language mnemonic, the operation, the flag status, and gives comments on each instruction. The *Z80 CPU Technical Manual* (03-0029-01) and *Assembly Language Programming Manual* (03-0002-01) contain significantly more details for programming use.

The instructions are divided into the following categories:

- 8-bit loads
- 16-bit loads
- Exchanges, block transfers, and searches
- 8-bit arithmetic and logic operations
- General-purpose arithmetic and CPU control
- 16-bit arithmetic operations
- Rotates and shifts
- Bit set, reset, and test operations
- Jumps
- Calls, returns, and restarts
- Input and output operations

A variety of addressing modes are implemented to permit efficient and fast data transfer between various registers, memory locations, and input/output devices. These addressing modes include:

- Immediate
- Immediate extended
- Modified page zero
- Relative
- Extended
- Indexed
- Register
- Register indirect
- Implied
- Bit

8-Bit Load Group

Mnemonic	Symbolic Operation	S	Z	H	P/V	N	C	Opcode 76 543 210	Hex	No. of Bytes	No. of M Cycles	No. of T States	Comments
LD r, r'	r ← r'	•	•	X	•	X	•	•	•	01 r r'		1	1	4	r, r' Reg.
LD r, n	r ← n	•	•	X	•	X	•	•	•	00 r 110		2	2	7	000 B
										— n —					001 C
LD r, (HL)	r ← (HL)	•	•	X	•	X	•	•	•	01 r 110		1	2	7	010 D
LD r, (IX + d)	r ← (IX + d)	•	•	X	•	X	•	•	•	11 011 101	DD	3	5	19	011 E
										01 r 101					100 H
										— d —					101 L
LD r, (IY + d)	r ← (IY + d)	•	•	X	•	X	•	•	•	11 111 101	FD	3	5	19	111 A
										01 r 110					
										— d —					
LD (HL), r	(HL) ← r	•	•	X	•	X	•	•	•	01 110 r		1	2	7	
LD (IX + d), r	(IX + d) ← r	•	•	X	•	X	•	•	•	11 011 101	DD	3	5	19	
										01 110 r					
										— d —					
LD (IY + d), r	(IY + d) ← r	•	•	X	•	X	•	•	•	11 111 101	FD	3	5	19	
										01 110 r					
										— d —					
LD (HL), n	(HL) ← n	•	•	X	•	X	•	•	•	00 110 110	36	2	3	10	
										— n —					
LD (IX + d), n	(IX + d) ← n	•	•	X	•	X	•	•	•	11 011 101	DD	4	5	19	
										00 110 110	36				
										— d —					
										— n —					
LD (IY + d), n	(IY + d) ← n	•	•	X	•	X	•	•	•	11 111 101	FD	4	5	19	
										00 110 110	36				
										— d —					
										— n —					
LD A, (BC)	A ← (BC)	•	•	X	•	X	•	•	•	00 001 010	0A	1	2	7	
LD A, (DE)	A ← (DE)	•	•	X	•	X	•	•	•	00 011 010	1A	1	2	7	
LD A, (nn)	A ← (nn)	•	•	X	•	X	•	•	•	00 111 010	3A	3	4	13	
										— n —					
										— n —					
LD (BC), A	(BC) ← A	•	•	X	•	X	•	•	•	00 000 010	02	1	2	7	
LD (DE), A	(DE) ← A	•	•	X	•	X	•	•	•	00 010 010	12	1	2	7	
LD (nn), A	(nn) ← A	•	•	X	•	X	•	•	•	00 110 010	32	3	4	13	
										— n —					
										— n —					
LD A, I	A ← I	↕	↕	X	0	X	IFF	0	•	11 101 101	ED	2	2	9	
										01 010 111	57				
LD A, R	A ← R	↕	↕	X	0	X	IFF	0	•	11 101 101	ED	2	2	9	
										01 011 111	5F				
LD I, A	I ← A	•	•	X	•	X	•	•	•	11 101 101	ED	2	2	9	
										01 000 111	47				
LD R, A	R ← A	•	•	X	•	X	•	•	•	11 101 101	ED	2	2	9	
										01 001 111	4F				

NOTES: r, r' means any of the registers A, B, C, D, E, H, L.
IFF the content of the interrupt enable flip-flop, (IFF) is copied into the P/V flag.
For an explanation of flag notation and symbols for mnemonic tables, see Symbolic Notation section following tables.

16-Bit Load Group

Mnemonic	Symbolic Operation	Flags S	Z	H	P/V	N	C	Opcode 76 543 210 Hex	No. of Bytes	No. of M Cycles	No. of T States	Comments
LD dd, nn	dd ← nn	•	•	X	X	•	•	00 dd0 001 – n – – n –	3	3	10	dd: 00 BC, 01 DE, 10 HL, 11 SP
LD IX, nn	IX ← nn	•	•	X	X	•	•	11 011 101 DD 00 100 001 21 – n – – n –	4	4	14	
LD IY, nn	IY ← nn	•	•	X	X	•	•	11 111 101 FD 00 100 001 21 – n – – n –	4	4	14	
LD HL, (nn)	H ← (nn + 1) L ← (nn)	•	•	X	X	•	•	00 101 010 2A – n – – n –	3	5	16	
LD dd, (nn)	dd_H ← (nn + 1) dd_L ← (nn)	•	•	X	X	•	•	11 101 101 ED 01 dd1 011 – n – – n –	4	6	20	
LD IX, (nn)	IX_H ← (nn + 1) IX_L ← (nn)	•	•	X	X	•	•	11 011 101 DD 00 101 010 2A – n – – n –	4	6	20	
LD IY, (nn)	IY_H ← (nn + 1) IY_L ← (nn)	•	•	X	X	•	•	11 111 101 FD 00 101 010 2A – n – – n –	4	6	20	
LD (nn), HL	(nn + 1) ← H (nn) ← L	•	•	X	X	•	•	00 100 010 22 – n – – n –	3	5	16	
LD (nn), dd	(nn + 1) ← dd_H (nn) ← dd_L	•	•	X	X	•	•	11 101 101 ED 01 dd0 011 – n – – n –	4	6	20	
LD (nn), IX	(nn + 1) ← IX_H (nn) ← IX_L	•	•	X	X	•	•	11 011 101 DD 00 100 010 22 – n – – n –	4	6	20	
LD (nn), IY	(nn + 1) ← IY_H (nn) ← IY_L	•	•	X	X	•	•	11 111 101 FD 00 100 010 22 – n – – n –	4	6	20	
LD SP, HL	SP ← HL	•	•	X	X	•	•	11 111 001 F9	1	1	6	
LD SP, IX	SP ← IX	•	•	X	X	•	•	11 011 101 DD 11 111 001 F9	2	2	10	
LD SP, IY	SP ← IY	•	•	X	X	•	•	11 111 101 FD 11 111 001 F9	2	2	10	
PUSH qq	(SP – 2) ← qq_L (SP – 1) ← qq_H SP ← SP – 2	•	•	X	X	•	•	11 qq0 101	1	3	11	qq: 00 BC, 01 DE, 10 HL, 11 AF
PUSH IX	(SP – 2) ← IX_L (SP – 1) ← IX_H SP ← SP – 2	•	•	X	X	•	•	11 011 101 DD 11 100 101 E5	2	4	15	
PUSH IY	(SP – 2) ← IY_L (SP – 1) ← IY_H SP ← SP – 2	•	•	X	X	•	•	11 111 101 FD 11 100 101 E5	2	4	15	
POP qq	qq_H ← (SP + 1) qq_L ← (SP) SP ← SP + 2	•	•	X	X	•	•	11 qq0 001	1	3	10	
POP IX	IX_H ← (SP + 1) IX_L ← (SP) SP ← SP + 2	•	•	X	X	•	•	11 011 101 DD 11 100 001 E1	2	4	14	
POP IY	IY_H ← (SP + 1) IY_L ← (SP) SP ← SP + 2	•	•	X	X	•	•	11 111 101 FD 11 100 001 E1	2	4	14	

NOTES: dd is any of the register pairs BC, DE, HL, SP.
qq is any of the register pairs AF, BC, DE, HL.
$(PA)_H$, $(PA)_L$ refer to high order and low order eight bits of the register pair respectively.
e.g., BC_L = C, AF_H = A

Exchange, Block Transfer, Block Search Groups

Mnemonic	Symbolic Operation	Flags S	Z	H	P/V	N	C	Opcode 76 543 210 Hex	No. of Bytes	No. of M Cycles	No. of T States	Comments
EX DE, HL	DE ↔ HL	•	•	X	X	•	•	11 101 011 EB	1	1	4	
EX AF, AF'	AF ↔ AF'	•	•	X	X	•	•	00 001 000 08	1	1	4	
EXX	BC ↔ BC' DE ↔ DE' HL ↔ HL'	•	•	X	X	•	•	11 011 001 D9	1	1	4	Register bank and auxiliary register bank exchange
EX (SP), HL	H ↔ (SP + 1) L ↔ (SP)	•	•	X	X	•	•	11 100 011 E3	1	5	19	
EX (SP), IX	IX_H ↔ (SP + 1) IX_L ↔ (SP)	•	•	X	X	•	•	11 011 101 DD 11 100 011 E3	2	6	23	
EX (SP), IY	IY_H ↔ (SP + 1) IY_L ↔ (SP)	•	•	X	X	•	•	11 111 101 FD 11 100 011 E3	2	6	23	
LDI	(DE) ← (HL) DE ← DE + 1 HL ← HL + 1 BC ← BC – 1	•	•	X	0	X	①	0 •	11 101 101 ED 10 100 000 A0	2	4	16	Load (HL) into (DE), increment the pointers and decrement the byte counter (BC)
LDIR	(DE) ← (HL) DE ← DE + 1 HL ← HL + 1 BC ← BC – 1 Repeat until BC = 0	•	•	X	0	X	0	0 •	11 101 101 ED 10 110 000 B0	2 2	5 4	21 16	If BC ≠ 0 If BC = 0

NOTE: ① P/V flag is 0 if the result of BC – 1 = 0, otherwise P/V = 1.

Exchange, Block Transfer, Block Search Groups (Continued)

Mnemonic	Symbolic Operation	S	Z	H	P/V	N	C	Opcode 76 543 210 Hex	No. of Bytes	No. of M Cycles	No. of T States	Comments
LDD	(DE) ← (HL) DE ← DE − 1 HL ← HL − 1 BC ← BC − 1	•	•	X	①	X	0	•	11 101 101 ED 10 101 000 A8	2	4	16	
LDDR	(DE) ← (HL) DE ← DE − 1 HL ← HL − 1 BC ← BC − 1 Repeat until BC = 0	•	•	X	①	X	0	•	11 101 101 ED 10 111 000 B8	2 2	5 4	21 16	If BC ≠ 0 If BC = 0
CPI	A − (HL) HL ← HL + 1 BC ← BC − 1	↕	↕	X	↕	X	①	↕	11 101 101 ED 10 100 001 A1	2	4	16	
CPIR	A − (HL) HL ← HL + 1 BC ← BC − 1 Repeat until A = (HL) or BC = 0	↕	↕	X	↕	X	①	↕	11 101 101 ED 10 110 001 B1	2 2	5 4	21 16	If BC ≠ 0 and A ≠ (HL) If BC = 0 or A = (HL)
CPD	A − (HL) HL ← HL − 1 BC ← BC − 1	↕	↕	X	↕	X	①	↕	11 101 101 ED 10 101 001 A9	2	4	16	
CPDR	A − (HL) HL ← HL − 1 BC ← BC − 1 Repeat until A = (HL) or BC = 0	↕	↕	X	↕	X	①	↕	11 101 101 ED 10 111 001 B9	2 2	5 4	21 16	If BC ≠ 0 and A ≠ (HL) If BC = 0 or A = (HL)

NOTES: ① P/V flag is 0 if the result of BC − 1 = 0, otherwise P/V = 1.
② Z flag is 1 if A = (HL), otherwise Z = 0.

8-Bit Arithmetic and Logical Group

Mnemonic	Symbolic Operation	S	Z	H	P/V	N	C	Opcode 76 543 210 Hex	No. of Bytes	No. of M Cycles	No. of T States	Comments
ADD A, r	A ← A + r	↕	↕	X	↕	X	V	0	↕	10 000 r	1	1	4	r: Reg. 000 B 001 C 010 D 011 E 100 H 101 L 111 A
ADD A, n	A ← A + n	↕	↕	X	↕	X	V	0	↕	11 000 110 — n —	2	2	7	
ADD A, (HL)	A ← A + (HL)	↕	↕	X	↕	X	V	0	↕	10 000 110	1	2	7	
ADD A, (IX + d)	A ← A + (IX + d)	↕	↕	X	↕	X	V	0	↕	11 011 101 DD 10 000 110 — d —	3	5	19	
ADD A, (IY + d)	A ← A + (IY + d)	↕	↕	X	↕	X	V	0	↕	11 111 101 FD 10 000 110 — d —	3	5	19	
ADC A, s	A ← A + s + CY	↕	↕	X	↕	X	V	0	↕	001				s is any of r, n, (HL), (IX + d), (IY + d) as shown for ADD instruction. The indicated bits replace the 000 in the ADD set above.
SUB s	A ← A − s	↕	↕	X	↕	X	V	1	↕	010				
SBC A, s	A ← A − s − CY	↕	↕	X	↕	X	V	1	↕	011				
AND s	A ← A ∧ s	↕	↕	X	1	X	P	0	0	100				
OR s	A ← A ∨ s	↕	↕	X	0	X	P	0	0	110				
XOR s	A ← A ⊕ s	↕	↕	X	0	X	P	0	0	101				
CP s	A − s	↕	↕	X	↕	X	V	1	↕	111				
INC r	r ← r + 1	↕	↕	X	↕	X	V	0	•	00 r 100	1	1	4	
INC (HL)	(HL) ← (HL) + 1	↕	↕	X	↕	X	V	0	•	00 110 100	1	3	11	
INC (IX + d)	(IX + d) ← (IX + d) + 1	↕	↕	X	↕	X	V	0	•	11 011 101 DD 00 110 100 — d —	3	6	23	
INC (IY + d)	(IY + d) ← (IY + d) + 1	↕	↕	X	↕	X	V	0	•	11 111 101 FD 00 110 100 — d —	3	6	23	
DEC m	m ← m − 1	↕	↕	X	↕	X	V	1	•	101				m is any of r, (HL), (IX + d), (IY + d) as shown for INC. DEC same format and states as INC. Replace 100 with 101 in opcode.

General-Purpose Arithmetic and CPU Control Groups

Mnemonic	Symbolic Operation	S	Z	H	P/V	N	C	Opcode 76 543 210	Hex	No. of Bytes	No. of M Cycles	No. of T States	Comments
DAA	Converts acc. content into packed BCD following add or subtract with packed BCD operands	↕	↕	X	P	•	↕	00 100 111	27	1	1	4	Decimal adjust accumulator.
CPL	$A \leftarrow \overline{A}$	•	•	X	1	X	•	1	•	00 101 111	2F	1	1	4	Complement accumulator (one's complement).
NEG	$A \leftarrow 0 - A$	↕	↕	X	↕	X	V	1	↕	11 101 101 01 000 100	ED 44	2	2	8	Negate acc. (two's complement).
CCF	$CY \leftarrow \overline{CY}$	•	•	X	X	X	•	0	↕	00 111 111	3F	1	1	4	Complement carry flag.
SCF	CY ← 1	•	•	X	0	X	•	0	1	00 110 111	37	1	1	4	Set carry flag.
NOP	No operation	•	•	X	•	X	•	•	•	00 000 000	00	1	1	4	
HALT	CPU halted	•	•	X	•	X	•	•	•	01 110 110	76	1	1	4	
DI *	IFF ← 0	•	•	X	•	X	•	•	•	11 110 011	F3	1	1	4	
EI *	IFF ← 1	•	•	X	•	X	•	•	•	11 111 011	FB	1	1	4	
IM 0	Set interrupt mode 0	•	•	X	•	X	•	•	•	11 101 101 01 000 110	ED 46	2	2	8	
IM 1	Set interrupt mode 1	•	•	X	•	X	•	•	•	11 101 101 01 010 110	ED 56	2	2	8	
IM 2	Set interrupt mode 2	•	•	X	•	X	•	•	•	11 101 101 01 011 110	ED 5E	2	2	8	

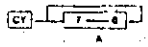
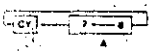
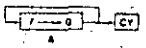
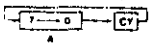
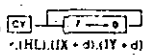
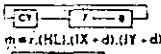
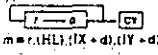
NOTES: IFF indicates the interrupt enable flip-flop.
CY indicates the carry flip-flop.
* indicates interrupts are not sampled at the end of EI or DI.

16-Bit Arithmetic Group

Mnemonic	Symbolic Operation	S	Z	H	P/V	N	C	Opcode 76 543 210	Hex	No. of Bytes	No. of M Cycles	No. of T States	Comments
ADD HL, ss	HL ← HL + ss	•	•	X	X	X	•	0	↕	00 ss1 001		1	3	11	ss Reg 00 BC 01 DE 10 HL 11 SP
ADC HL, ss	HL ← HL + ss + CY	↕	↕	X	X	X	V	0	↕	11 101 101 01 ss1 010	ED	2	4	15	
SBC HL, ss	HL ← HL − ss − CY	↕	↕	X	X	X	V	1	↕	11 101 101 01 ss0 010	ED	2	4	15	
ADD IX, pp	IX ← IX + pp	•	•	X	X	X	•	0	↕	11 011 101 00 pp1 001	DD	2	4	15	pp Reg. 00 BC 01 DE 10 IX 11 SP
ADD IY, rr	IY ← IY + rr	•	•	X	X	X	•	0	↕	11 111 101 00 rr1 001	FD	2	4	15	rr Reg. 00 BC 01 DE 10 IY 11 SP
INC ss	ss ← ss + 1	•	•	X	•	X	•	•	•	00 ss0 011		1	1	6	
INC IX	IX ← IX + 1	•	•	X	•	X	•	•	•	11 011 101 00 100 011	DD 23	2	2	10	
INC IY	IY ← IY + 1	•	•	X	•	X	•	•	•	11 111 101 00 100 011	FD 23	2	2	10	
DEC ss	ss ← ss − 1	•	•	X	•	X	•	•	•	00 ss1 011		1	1	6	
DEC IX	IX ← IX − 1	•	•	X	•	X	•	•	•	11 011 101 00 101 011	DD 2B	2	2	10	
DEC IY	IY ← IY − 1	•	•	X	•	X	•	•	•	11 111 101 00 101 011	FD 2B	2	2	10	

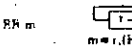
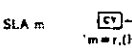
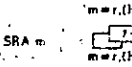
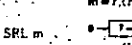
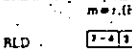
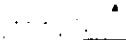
NOTES: ss is any of the register pairs BC, DE, HL, SP.
pp is any of the register pairs BC, DE, IX, SP.
rr is any of the register pairs BC, DE, IY, SP.

Rotate and Shift Group

Mnemonic	Symbolic Operation	S	Z	H	P/V	N	C	Opcode 76 543 210	Hex	No. of Bytes	No. of M Cycles	No. of T States	Comments
RLCA		•	•	X	0	X	•	0	↕	00 000 111	07	1	1	4	Rotate left circular accumulator.
RLA		•	•	X	0	X	•	0	↕	00 010 111	17	1	1	4	Rotate left accumulator.
RRCA		•	•	X	0	X	•	0	↕	00 001 111	0F	1	1	4	Rotate right circular accumulator.
RRA		•	•	X	0	X	•	0	↕	00 011 111	1F	1	1	4	Rotate right accumulator.
RLC r		↕	↕	X	0	X	P	0	↕	11 001 011 00 000 r	CB	2	2	8	Rotate left circular register r.
RLC (HL)		↕	↕	X	0	X	P	0	↕	11 001 011 00 000 110	CB	2	4	15	r Reg. 000 B 001 C 010 D 011 E 100 H 101 L 111 A
RLC (IX + d)	 r,(HL),(IX + d),(IY + d)	↕	↕	X	0	X	P	0	↕	11 011 101 11 001 011 — d — 00 000 110	DD CB	4	6	23	
RLC (IY + d)		↕	↕	X	0	X	P	0	↕	11 111 101 11 001 011 — d — 00 000 110	FD CB	4	6	23	
RL m	 m ≡ r,(HL),(IX + d),(IY + d)	↕	↕	X	0	X	P	0	↕	010					Instruction format and states are as shown for RLC s. To form new opcode replace 000 of RLC s with shown code.
RRC m	 m ≡ r,(HL),(IX + d),(IY + d)	↕	↕	X	0	X	P	0	↕	001					

Rotate and Shift Group (Continued)

10

Mnemonic	Symbolic Operation	S	Z	H	P/V	N	C	Opcode 76 543 210	Hex	No. of Bytes	No. of M Cycles	No. of T States	Comments
RR m	 m = r,(HL),(IX + d),(IY + d)	↕	↕	X	0	X	P	0	↕	00 011 000	18				
SLA m	 m = r,(HL),(IX + d),(IY + d)	↕	↕	X	0	X	P	0	↕	00 100 000	20				
SRA m	 m = r,(HL),(IX + d),(IY + d)	↕	↕	X	0	X	P	0	↕	00 101 000	28				
SRL m	 m = r,(HL),(IX + d),(IY + d)	↕	↕	X	0	X	P	0	↕	00 111 000	38				
RLD		↕	↕	X	0	X	P	0	•	11 101 101 01 101 111	ED 6F	2	5	18	Rotate digit left and right between the accumulator and location (HL). The content of the upper half of the accumulator is unaffected.
RRD		↕	↕	X	0	X	P	0	•	11 101 101 01 100 111	ED 67	2	5	18	

Bit Set, Reset and Test Group

Mnemonic	Symbolic Operation	S	Z	H	P/V	N	C	Opcode 76 543 210	Hex	No. of Bytes	No. of M Cycles	No. of T States	Comments
BIT b, r	$Z = \overline{r_b}$	X	↕	X	1	X	X	0	•	11 001 011 01 b r	CB	2	2	8	r Reg. 000 B 001 C 010 D 011 E 100 H 101 L 111 A b Bit Tested 000 0 001 1 010 2 011 3 100 4 101 5 110 6 111 7
BIT b, (HL)	$Z = \overline{(HL)_b}$	X	↕	X	1	X	X	0	•	11 001 011 01 b 110	CB	2	3	12	
BIT b, (IX + d)$_b$	$Z = \overline{(IX+d)_b}$	X	↕	X	1	X	X	0	•	11 011 101 11 001 011 – d – 01 b 110	DD CB	4	5	20	
BIT b, (IY + d)$_b$	$Z = \overline{(IY+d)_b}$	X	↕	X	1	X	X	0	•	11 111 101 11 001 011 – d – 01 b 110	FD CB	4	5	20	
SET b, r	$r_b = 1$	•	•	X	•	X	•	•	•	11 001 011 11 b r	CB	2	2	8	
SET b, (HL)	$(HL)_b = 1$	•	•	X	•	X	•	•	•	11 001 011 11 b 110	CB	2	4	15	
SET b, (IX + d)	$(IX + d)_b = 1$	•	•	X	•	X	•	•	•	11 011 101 11 001 011 – d – 11 b 110	DD CB	4	6	22	
SET b, (IY + d)	$(IY + d)_b = 1$	•	•	X	•	X	•	•	•	11 111 101 11 001 011 – d – 11 b 110	FD CB	4	6	23	
RES b, m	$m_b = 0$ m = r, (HL), (IX + d), (IY + d)	•	•	X	•	X	•	•	•	10					To form new opcode replace 11 of SET b, s with 10. Flags and time states for SET instruction.

NOTE: The notation m_b indicates bit b (0 to 7) of location m.

Jump Group

Mnemonic	Symbolic Operation	S	Z	H	P/V	N	C	Opcode 76 543 210	Hex	No. of Bytes	No. of M Cycles	No. of T States	Comments
JP nn	PC = nn	•	•	X	•	X	•	•	•	11 000 011 – n – – n –	C3	3	3	10	
JP cc, nn	If condition cc is true PC = nn, otherwise continue	•	•	X	•	X	•	•	•	11 cc 010 – n – – n –		3	3	10	cc Condition 000 NZ non-zero 001 Z zero 010 NC non-carry 011 C carry 100 PO parity odd 101 PE parity even 110 P sign positive 111 M sign negative
JR e	PC = PC + e	•	•	X	•	X	•	•	•	00 011 000 – e – 2 –	18	2	3	12	
JR C, e	If C = 0, continue	•	•	X	•	X	•	•	•	00 111 000 – e – 2 –	38	2	2	7	If condition not met.
	If C = 1, PC = PC + e											2	3	12	If condition is met.
JR NC, e	If C = 1, continue	•	•	X	•	X	•	•	•	00 110 000 – e – 2 –	30	2	2	7	If condition not met.
	If C = 0, PC = PC + e											2	3	12	If condition is met.
JP Z, e	If Z = 0 continue	•	•	X	•	X	•	•	•	00 101 000 – e – 2 –	28	2	2	7	If condition not met.
	If Z = 1, PC = PC + e											2	3	12	If condition is met.
JR NZ, e	If Z = 1, continue	•	•	X	•	X	•	•	•	00 100 000 – e – 2 –	20	2	2	7	If condition not met.
	If Z = 0, PC = PC + e											2	3	12	If condition is met.
JP (HL)	PC = HL	•	•	X	•	X	•	•	•	11 101 001	E9	1	1	4	
JP (IX)	PC = IX	•	•	X	•	X	•	•	•	11 011 101 11 101 001	DD E9	2	2	8	

Jump Group (Continued)

Mnemonic	Symbolic Operation	S	Z	H	P/V	N	C	Opcode	No. of Bytes	No. of M Cycles	No. of T States	Comments
JP (IY)	PC ← IY	•	•	X	•	X	•	•	•	11 111 101 FD 11 101 001 E9	2	2	8	
DJNZ, e	B ← B - 1 If B = 0, continue If B ≠ 0, PC ← PC + e	•	•	X	•	X	•	•	•	00 010 000 10 — e - 2 —	2 2	2 1	8 13	If B = 0. If B ≠ 0

NOTES: e represents the extension in the relative addressing mode
e is a signed two's complement number in the range < -126, 129 >
e - 2 in the opcode provides an effective address of pc + e as PC is incremented by 2 prior to the addition of e.

11

Call and Return Group

CALL nn	(SP - 1) ← PC_H (SP - 2) ← PC_L PC ← nn	•	•	X	•	X	•	•	•	11 001 101 CD — n — — n —	3	5	17	
CALL cc, nn	If condition cc is false continue, otherwise same as CALL nn	•	•	X	•	X	•	•	•	11 cc 100 — n — — n —	3 3	3 5	10 17	If cc is false. If cc is true.
RET	PC_L ← (SP) PC_H ← (SP + 1)	•	•	X	•	X	•	•	•	11 001 001 C9	1	3	10	
RET cc	If condition cc is false continue, otherwise same as RET	•	•	X	•	X	•	•	•	11 cc 000	1 1	1 3	5 11	If cc is false. If cc is true.
RETI	Return from interrupt	•	•	X	•	X	•	•	•	11 101 101 ED 01 001 101 4D	2	4	14	
RETN[1]	Return from non-maskable interrupt	•	•	X	•	X	•	•	•	11 101 101 ED 01 000 101 45	2	4	14	
RST p	(SP - 1) ← PC_H (SP - 2) ← PC_L PC_H ← 0 PC_L ← p	•	•	X	•	X	•	•	•	11 t 111	1	3	11	

cc: Condition
000 NZ non-zero
001 Z zero
010 NC non-carry
011 C carry
100 PO parity odd
101 PE parity even
110 P sign positive
111 M sign negative

t: p
000 00H
001 08H
010 10H
011 18H
100 20H
101 28H
110 30H
111 38H

NOTE: [1] RETN loads IFF_2 → IFF_1

Input and Output Group

IN A, (n)	A ← (n)	•	•	X	•	X	•	•	•	11 011 011 DB — n —	2	3	11	n to A_0 ~ A_7 Acc. to A_8 ~ A_{15}
IN r, (C)	r ← (C) if r = 110 only the flags will be affected	‡	‡	X	‡	X	P	0	•	11 101 101 ED 01 r 000	2	3	12	C to A_0 ~ A_7 B to A_8 ~ A_{15}
INI	(HL) ← (C) B ← B - 1 HL ← HL + 1	①	‡	X	X	X	X	1	•	11 101 101 ED 10 100 010 A2	2	4	16	C to A_0 ~ A_7 B to A_8 ~ A_{15}
INIR	(HL) ← (C) B ← B - 1 HL ← HL + 1 Repeat until B = 0	①	‡	X	X	X	X	1	•	11 101 101 ED 10 110 010 B2	2 2	5 (If B≠0) 4 (If B=0)	21 16	C to A_0 ~ A_7 B to A_8 ~ A_{15}
IND	(HL) ← (C) B ← B - 1 HL ← HL - 1	①	‡	X	X	X	X	1	•	11 101 101 ED 10 101 010 AA	2	4	16	C to A_0 ~ A_7 B to A_8 ~ A_{15}
INDR	(HL) ← (C) B ← B - 1 HL ← HL - 1 Repeat until B = 0	①	‡	X	X	X	X	1	•	11 101 101 ED 10 111 010 BA	2 2	5 (If B≠0) 4 (If B=0)	21 16	C to A_0 ~ A_7 B to A_8 ~ A_{15}
OUT (n), A	(n) ← A	•	•	X	•	X	•	•	•	11 010 011 D3 — n —	2	3	11	n to A_0 ~ A_7 Acc. to A_8 ~ A_{15}
OUT (C), r	(C) ← r	•	•	X	•	X	•	•	•	11 101 101 ED 01 r 001	2	3	12	C to A_0 ~ A_7 B to A_8 ~ A_{15}
OUTI	(C) ← (HL) B ← B - 1 HL ← HL + 1	①	‡	X	X	X	X	1	•	11 101 101 ED 10 100 011 A3	2	4	16	C to A_0 ~ A_7 B to A_8 ~ A_{15}
OTIR	(C) ← (HL) B ← B - 1 HL ← HL + 1 Repeat until B = 0	①	‡	X	X	X	X	1	•	11 101 101 ED 10 110 011 B3	2 2	5 (If B≠0) 4 (If B=0)	21 16	C to A_0 ~ A_7 B to A_8 ~ A_{15}
OUTD	(C) ← (HL) B ← B - 1 HL ← HL - 1	①	‡	X	X	X	X	1	•	11 101 101 ED 10 101 011 AB	2	4	16	C to A_0 ~ A_7 B to A_8 ~ A_{15}

NOTE: ① If the result of B - 1 is zero the Z flag is set, otherwise it is reset.

Input and Output Group (Continued)

Mnemonic	Symbolic Operation	S	Z	H	P/V	N	C	Opcode 76 543 210 Hex	No. of Bytes	No. of M Cycles	No. of T States	Comments
OTDR	(C) ← (HL) B ← B − 1 HL ← HL − 1 Repeat until B = 0	—	X	X	X	X	X	11 101 101 ED 10 111 011 **12**	2	5 (If B ≠ 0) 4 (If B = 0)	21 16	C to $A_0 - A_7$ B to $A_8 - A_{15}$

Summary of Flag Operation

Instruction	D_7 S	Z	H	P/V	N	D_0 C	Comments
ADD A, s; ADC A, s	↕	↕	X	V	0	↕	8-bit add or add with carry
SUB s, SBC A, s; CP s; NEG	↕	↕	X	V	1	↕	8 bit subtract, subtract with carry, compare and negate accumulator.
AND s	↕	↕	X	P	0	0	Logical operations.
OR s, XOR s	↕	↕	X	P	0	0	
INC s	↕	↕	X	V	0	•	8-bit increment.
DEC s	↕	↕	X	V	1	•	8 bit decrement.
ADD DD, ss	•	•	X	•	0	↕	16-bit add.
ADC HL, ss	↕	↕	X	V	0	↕	16-bit add with carry.
SBC HL, ss	↕	↕	X	V	1	↕	16-bit subtract with carry.
RLA; RLCA; RRA; RRCA	•	•	X	•	0	↕	Rotate accumulator.
RL m; RLC m; RR m; RRC m; SLA m; SRA m; SRL m	↕	↕	X	P	0	↕	Rotate and shift locations.
RLD; RRD	↕	↕	X	P	0	•	Rotate right left and right
DAA	↕	↕	X	P	•	↕	Decimal adjust accumulator.
CPL	•	•	X	•	1	•	Complement accumulator.
SCF	•	•	X	•	0	1	Set carry.
CCF	•	•	X	•	0	↕	Complement carry.
IN r (C)	↕	↕	X	P	0	•	Input register indirect.
INI; IND; OUTI; OUTD	X	↕	X	X	1	•	Block input and output. Z = 0 if B ≠ 0 otherwise Z = 0.
INIR; INDR; OTIR; OTDR	X	1	X	X	1	•	
LDI; LDD	X	X	X	↕	0	•	Block transfer instructions. P/V = 1 if BC ≠ 0, otherwise P/V = 0.
LDIR, LDDR	X	X	X	0	0	•	
CPI; CPIR; CPD; CPDR	X	↕	X	↕	1	•	Block search instructions. Z = 1 if A = (HL), otherwise Z = 0. P/V = 1 if BC ≠ 0, otherwise P/V = 0
LD A, I; LD A, R	↕	↕	X	IFF	0	•	The content of the interrupt enable flip-flop (IFF) is copied into the P/V flag.
BIT b, s	X	↕	X	X	0	•	The state of bit b of location s is copied into the Z flag.

Symbolic Notation

Symbol	Operation
S	Sign flag. S = 1 if the MSB of the result is 1.
Z	Zero flag. Z = 1 if the result of the operation is 0.
P/V	Parity or overflow flag. Parity (P) and overflow (V) share the same flag. Logical operations affect this flag with the parity of the result while arithmetic operations affect this flag with the overflow of the result. If P/V holds parity, P/V = 1 if the result of the operation is even, P/V = 0 if result is odd. If P/V holds overflow, P/V = 1 if the result of the operation produced an overflow.
H	Half-carry flag. H = 1 if the add or subtract operation produced a carry into or borrow from bit 4 of the accumulator.
N	Add/Subtract flag. N = 1 if the previous operation was a subtract.
H & N	H and N flags are used in conjunction with the decimal adjust instruction (DAA) to properly correct the result into packed BCD format following addition or subtraction using operands with packed BCD format.
C	Carry/Link flag. C = 1 if the operation produced a carry from the MSB of the operand or result.

Symbol	Operation
↕	The flag is affected according to the result of the operation.
•	The flag is unchanged by the operation.
0	The flag is reset by the operation.
1	The flag is set by the operation.
X	The flag is a "don't care."
V	P/V flag affected according to the overflow result of the operation.
P	P/V flag affected according to the parity result of the operation.
r	Any one of the CPU registers A, B, C, D, E, H, L.
s	Any 8-bit location for all the addressing modes allowed for the particular instruction.
ss	Any 16-bit location for all the addressing modes allowed for that instruction.
ii	Any one of the two index registers IX or IY.
R	Refresh counter.
n, nn	8-bit value in range < 0, 255 >. 16-bit value in range < 0, 65535 >.

A_0-A_{15}. *Address Bus* (output, active High, 3-state). A_0-A_{15} form a 16-bit address bus. The Address Bus provides the address for memory data bus exchanges (up to 64K bytes) and for I/O device exchanges.

$\overline{BUSACK}$. *Bus Acknowledge* (output, active Low). Bus Acknowledge indicates to the requesting device that the CPU address bus, data bus, and control signals $\overline{MREQ}$, $\overline{IORQ}$, $\overline{RD}$, and $\overline{WR}$ have entered their high-impedance states. The external circuitry can now control these lines.

$\overline{BUSREQ}$. *Bus Request* (input, active Low). Bus Request has a higher priority than $\overline{NMI}$ and is always recognized at the end of the current machine cycle. $\overline{BUSREQ}$ forces the CPU address bus, data bus, and control signals $\overline{MREQ}$, $\overline{IORQ}$, $\overline{RD}$, and $\overline{WR}$ to go to a high-impedance state so that other devices can control these lines. $\overline{BUSREQ}$ is normally wire-ORed and requires an external pullup for these applications. Extended $\overline{BUSREQ}$ periods due to extensive DMA operations can prevent the CPU from properly refreshing dynamic RAMs.

D_0-D_7. *Data Bus* (input/output, active High, 3-state). D_0-D_7 constitute an 8-bit bidirectional data bus, used for data exchanges with memory and I/O.

$\overline{HALT}$. *Halt State* (output, active Low). $\overline{HALT}$ indicates that the CPU has executed a Halt instruction and is awaiting either a non-maskable or a maskable interrupt (with the mask enabled) before operation can resume. While halted, the CPU executes NOPs to maintain memory refresh.

$\overline{INT}$. *Interrupt Request* (input, active Low). Interrupt Request is generated by I/O devices. The CPU honors a request at the end of the current instruction if the internal software-controlled interrupt enable flip-flop (IFF) is enabled. $\overline{INT}$ is normally wire-ORed and requires an external pullup for these applications.

$\overline{IORQ}$. *Input/Output Request* (output, active Low, 3-state). $\overline{IORQ}$ indicates that the lower half of the address bus holds a valid I/O address for an I/O read or write operation. $\overline{IORQ}$ is also generated concurrently with $\overline{M1}$ during an interrupt acknowledge cycle to indicate that an interrupt response vector can be placed on the data bus.

$\overline{M1}$. *Machine Cycle One* (output, active Low). $\overline{M1}$, together with $\overline{MREQ}$, indicates that the current machine cycle is the opcode fetch cycle of an instruction execution. $\overline{M1}$, together with $\overline{IORQ}$, indicates an interrupt acknowledge cycle.

$\overline{MREQ}$. *Memory Request* (output, active Low, 3-state). $\overline{MREQ}$ indicates that the address bus holds a valid address for a memory read or memory write operation.

$\overline{NMI}$. *Non-Maskable Interrupt* (input, active Low). $\overline{NMI}$ has a higher priority than $\overline{INT}$. $\overline{NMI}$ is always recognized at the end of the current instruction, independent of the status of the interrupt enable flip-flop, and automatically forces the CPU to restart at location 0066H.

$\overline{RD}$. *Memory Read* (output, active Low, 3-state). $\overline{RD}$ indicates that the CPU wants to read data from memory or an I/O device. The addressed I/O device or memory should use this signal to gate data onto the CPU data bus.

$\overline{RESET}$. *Reset* (input, active Low). $\overline{RESET}$ initializes the CPU as follows: it resets the interrupt enable flip-flop, clears the PC and Registers I and R, and sets the interrupt status to Mode 0. During reset-time, the address and data bus go to a high-impedance state, and all control output signals go to the inactive state. Note that $\overline{RESET}$ must be active for a minimum of three full clock cycles before the reset operation is complete.

$\overline{RFSH}$. *Refresh* (output, active Low). $\overline{RFSH}$, together with $\overline{MREQ}$, indicates that the lower seven bits of the system's address bus can be used as a refresh address to the system's dynamic memories.

$\overline{WAIT}$. *Wait* (input, active Low). $\overline{WAIT}$ indicates to the CPU that the addressed memory or I/O devices are not ready for a data transfer. The CPU continues to enter a Wait state as long as this signal is active. Extended $\overline{WAIT}$ periods can prevent the CPU from refreshing dynamic memory properly.

$\overline{WR}$. *Memory Write* (output, active Low, 3-state). $\overline{WR}$ indicates that the CPU data bus holds valid data to be stored at the addressed memory or I/O location.

CPU Timing

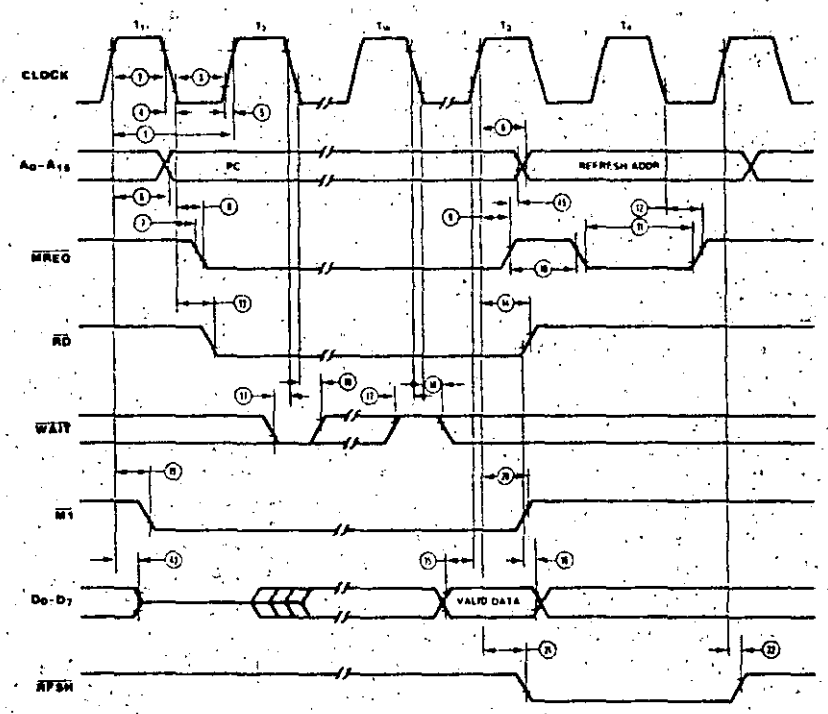
The Z80 CPU executes instructions by proceeding through a specific sequence of operations:

- Memory read or write
- I/O device read or write
- Interrupt acknowledge

The basic clock period is referred to as a T time or cycle, and three or more T cycles make up a machine cycle (M1, M2 or M3 for instance). Machine cycles can be extended either by the CPU automatically inserting one or more Wait states or by the insertion of one or more Wait states by the user.

Instruction Opcode Fetch. The CPU places the contents of the Program Counter (PC) on the address bus at the start of the cycle (Figure 5). Approximately one-half clock cycle later, $\overline{MREQ}$ goes active. The falling edge of $\overline{MREQ}$ can be used directly as a Chip Enable to dynamic memories. When active, $\overline{RD}$ indicates that the memory data can be enabled onto the CPU data bus.

The CPU samples the $\overline{WAIT}$ input with the rising edge of clock state T3. During clock states T3 and T4 of an M1 cycle dynamic RAM refresh can occur while the CPU starts decoding and executing the instruction. When the Refresh Control signal becomes active, refreshing of dynamic memory can take place.



NOTE: T_W-Wait cycle added when necessary for slow ancillary devices.

Figure 5: Instruction Opcode Fetch

CPU Timing (Continued)

Memory Read or Write Cycles. Figure 6 shows the timing of memory read or write cycles other than an opcode fetch ($\overline{M1}$) cycle. The $\overline{MREQ}$ and $\overline{RD}$ signals function exactly as in the fetch cycle. In a memory write cycle, $\overline{MREQ}$ also becomes active when the address bus is stable, so that it can be used directly as a Chip Enable for dynamic memories. The $\overline{WR}$ line is active when the data bus is stable, so that it can be used directly as an R/$\overline{W}$ pulse to most semiconductor memories.

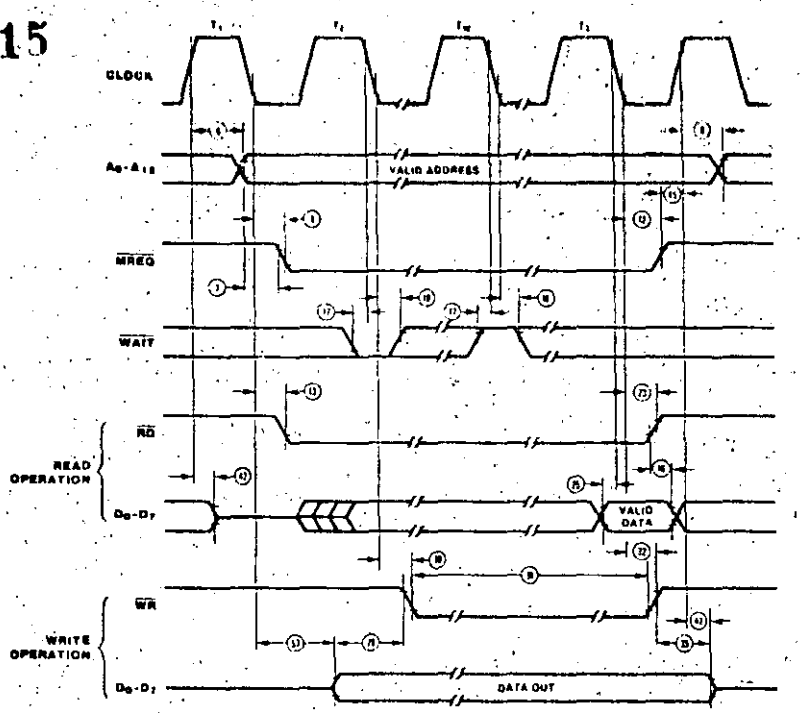
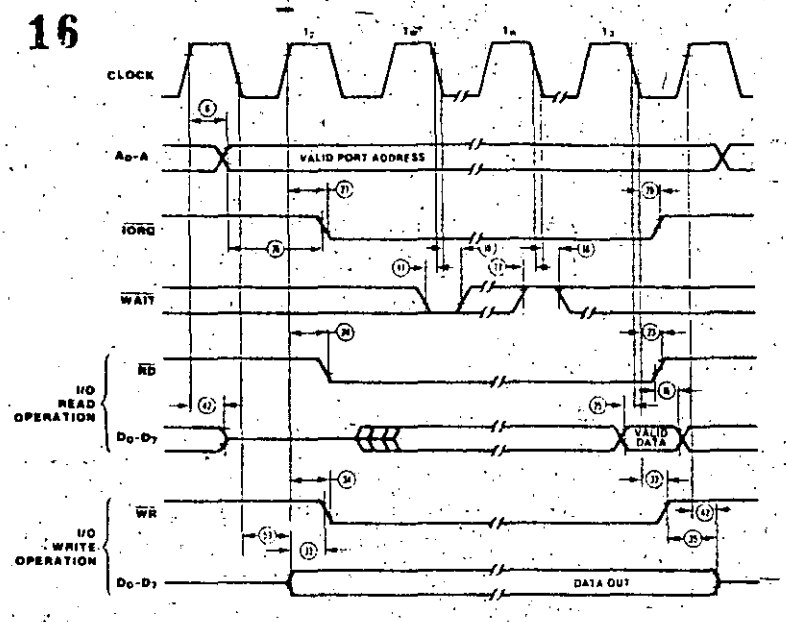


Figure 6. Memory Read or Write Cycles

CPU Timing (Continued)

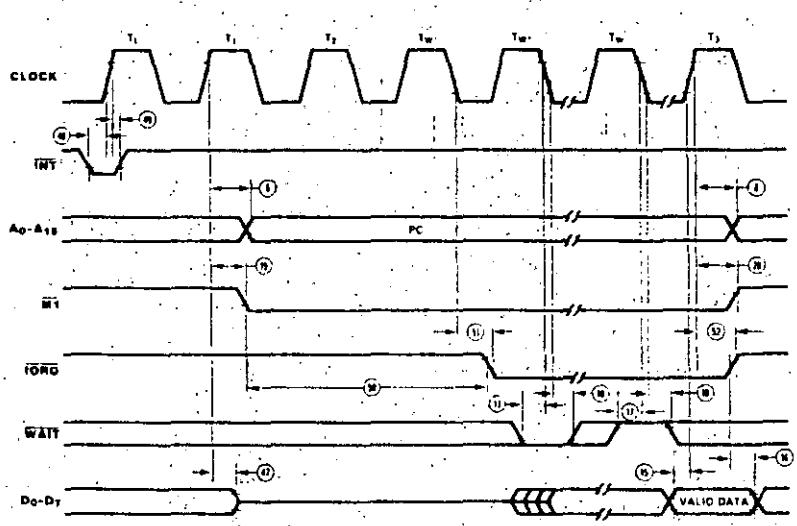
Input or Output Cycles. Figure 7 shows the timing for an I/O read or I/O write operation. During I/O operations, the CPU automatically inserts a single Wait state (T_w). This extra Wait state allows sufficient time for an I/O port to decode the address and the port address lines.



NOTE: T_w = One Wait cycle automatically inserted by CPU.

Figure 7. Input or Output Cycles

Interrupt Request/Acknowledge Cycle. The CPU samples the interrupt signal with the rising edge of the last clock cycle at the end of any instruction (Figure 8). When an interrupt is accepted, a special $\overline{M1}$ cycle is generated. During this $\overline{M1}$ cycle, $\overline{IORQ}$ becomes active (instead of $\overline{MREQ}$) to indicate that the interrupting device can place an 8-bit vector on the data bus. The CPU automatically adds two Wait states to this cycle.



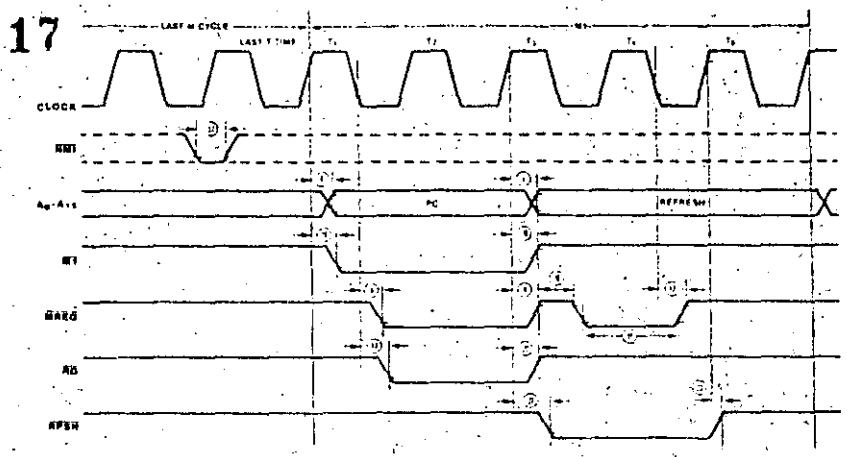
NOTE: 1) T_L = Last state of previous instruction.

2) Two Wait cycles automatically inserted by CPU(*).

Figure 8. Interrupt Request/Acknowledge Cycle

CPU Timing (Continued)

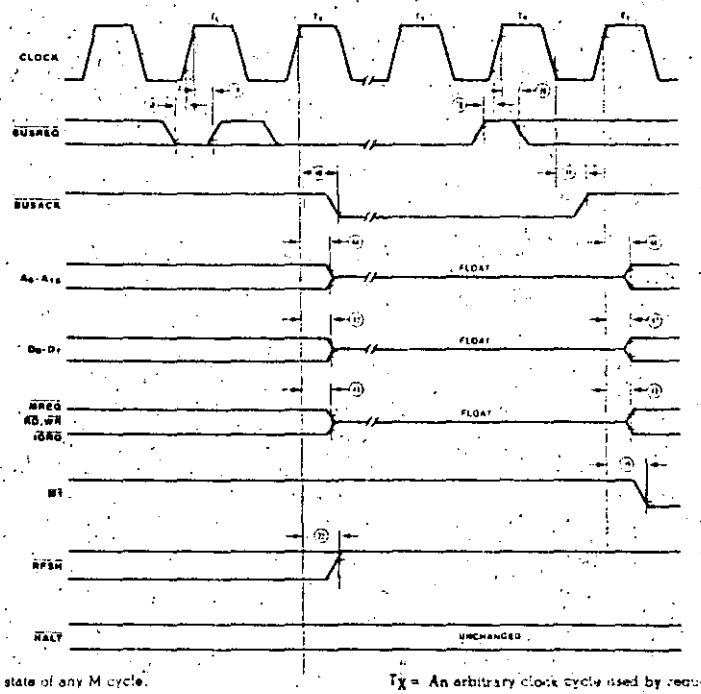
Non-Maskable Interrupt Request Cycle. $\overline{NMI}$ is sampled at the same time as the maskable interrupt input $\overline{INT}$ but has higher priority and cannot be disabled under software control. The subsequent timing is similar to that of a normal memory read operation except that data put on the bus by the memory is ignored. The CPU instead executes a restart (RST) operation and jumps to the $\overline{NMI}$ service routine located at address 0066H (Figure 9).



*Although $\overline{NMI}$ is an asynchronous input, to guarantee its being recognized on the following machine cycle, $\overline{NMI}$'s falling edge must occur no later than the rising edge of the clock cycle preceding T_{LAST}.

Figure 9. Non-Maskable Interrupt Request Operation

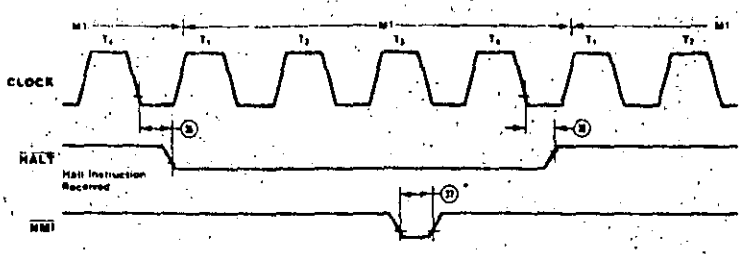
Bus Request/Acknowledge Cycle. The CPU samples $\overline{BUSREQ}$ with the rising edge of the last clock period of any machine cycle (Figure 10). If $\overline{BUSREQ}$ is active, the CPU sets its address, data, and $\overline{MREQ}$, $\overline{IORQ}$, $\overline{RD}$, and $\overline{WR}$ lines to a high-impedance state with the rising edge of the next clock pulse. At that time, any external device can take control of these lines, usually to transfer data between memory and I/O devices.



NOTE: T_L = Last state of any M cycle. T_X = An arbitrary clock cycle used by requesting device.

Figure 10. Bus Request/Acknowledge Cycle

Halt Acknowledge Cycle. When the CPU receives a $\overline{HALT}$ instruction, it executes NOP states until either an $\overline{INT}$ or $\overline{NMI}$ input is received. When in the Halt state, the $\overline{HALT}$ output is active and remains so until an interrupt is processed (Figure 11).



NOTE: INT will also force a Halt exit.

*See note, Figure 9.

Figure 11. Halt Acknowledge Cycle

Reset Cycle. $\overline{RESET}$ must be active for at least three clock cycles for the CPU to properly accept it. As long as $\overline{RESET}$ remains active, the address and data buses float, and the control outputs are inactive. Once $\overline{RESET}$ goes inactive, two internal T cycles are consumed before the CPU resumes normal processing operation. $\overline{RESET}$ clears the PC register, so the first opcode fetch will be to location 0000 (Figure 12).

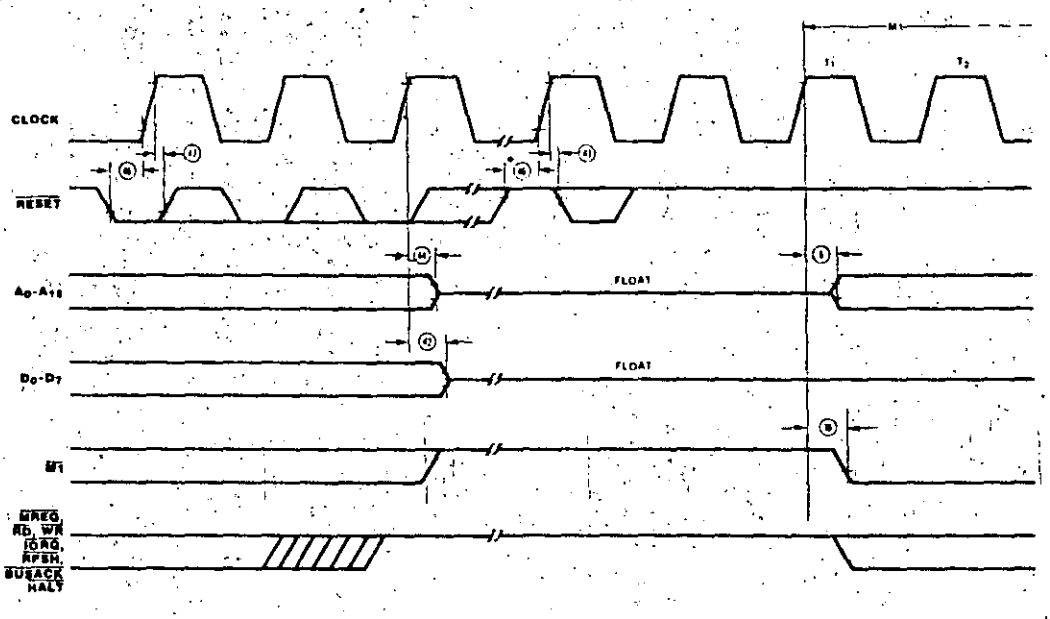


Figure 12. Reset Cycle

AC Characteristics

Number	Symbol	Parameter	Z80 CPU Min (ns)	Z80 CPU Max (ns)	Z80A CPU Min (ns)	Z80A CPU Max (ns)	Z80B CPU Min (ns)	Z80B CPU Max (ns)
1	TcC	Clock Cycle Time	400*		250*		165*	
2	TwCh	Clock Pulse Width (High)	180*		110*		65*	
3	TwCl	Clock Pulse Width (Low)	180	2000	110	2000	65	2000
4	TfC	Clock Fall Time	—	30	—	30	—	20
5	TrC	Clock Rise Time	—	30	—	30	—	20
6	TdCr(A)	Clock ↑ to Address Valid Delay	—	145	—	110	—	90
7	TdA(MREQf)	Address Valid to $\overline{MREQ}$ ↓ Delay	125*	—	65*	—	35*	—
8	TdCf(MREQf)	Clock ↓ to $\overline{MREQ}$ ↓ Delay	—	100	—	85	—	70
9	TdCr(MREQr)	Clock ↑ to $\overline{MREQ}$ ↑ Delay	—	100	—	85	—	70
10	TwMREQh	$\overline{MREQ}$ Pulse Width (High)	170*		110*		65*	
11	TwMREQl	$\overline{MREQ}$ Pulse Width (Low)	360*	—	220*	—	135*	—
12	TdCf(MREQr)	Clock ↓ to $\overline{MREQ}$ ↑ Delay	—	100	—	85	—	70
13	TdCf(RDf)	Clock ↓ to $\overline{RD}$ ↓ Delay	—	130	—	95	—	80
14	TdCr(RDr)	Clock ↑ to $\overline{RD}$ ↑ Delay	—	100	—	85	—	70
15	TsD(Cr)	Data Setup Time to Clock ↑	50		35		30	
16	ThD(RDr)	Data Hold Time to $\overline{RD}$ ↑	—	0	—	0	—	0
17	TsWAIT(Cf)	$\overline{WAIT}$ Setup Time to Clock ↓	70	—	70	—	60	—
18	ThWAIT(Cf)	$\overline{WAIT}$ Hold Time after Clock ↓	—	0	—	0	—	0
19	TdCr(M1f)	Clock ↑ to $\overline{M1}$ ↓ Delay	—	130	—	100	—	80
20	TdCr(M1r)	Clock ↑ to $\overline{M1}$ ↑ Delay	—	130	—	100	—	80
21	TdCr(RFSHf)	Clock ↑ to $\overline{RFSH}$ ↓ Delay	—	180	—	130	—	110
22	TdCr(RFSHr)	Clock ↑ to $\overline{RFSH}$ ↑ Delay	—	150	—	120	—	100
23	TdCf(RDr)	Clock ↓ to $\overline{RD}$ ↑ Delay	—	110	—	85	—	70
24	TdCr(RDf)	Clock ↑ to $\overline{RD}$ ↓ Delay	—	100	—	85	—	70
25	TsD(Cf)	Data Setup to Clock ↓ during M_2, M_3, M_4 or M_5 Cycles	60		50		40	
26	TdA(IORQf)	Address Stable prior to $\overline{IORQ}$ ↓	320*	—	180*	—	110*	—
27	TdCr(IORQf)	Clock ↑ to $\overline{IORQ}$ ↓ Delay	—	90	—	75	—	65
28	TdCf(IORQr)	Clock ↓ to $\overline{IORQ}$ ↑ Delay	—	110	—	85	—	70
29	TdD(WRf)	Data Stable prior to $\overline{WR}$ ↓	190*	—	80*	—	25*	—
30	TdCf(WRf)	Clock ↓ to $\overline{WR}$ ↓ Delay	—	90	—	80	—	70
31	TwWR	$\overline{WR}$ Pulse Width	360*	—	220*	—	135*	—
32	TdCf(WRr)	Clock ↓ to $\overline{WR}$ ↑ Delay	—	100	—	80	—	70
33	TdD(WRf)	Data Stable prior to $\overline{WR}$ ↓	20*	—	-10*	—	-55*	—
34	TdCr(WRf)	Clock ↑ to $\overline{WR}$ ↓ Delay	—	80	—	65	—	60
35	TdWRr(D)	Data Stable from $\overline{WR}$ ↑	120*		60*		30*	
36	TdCf(HALT)	Clock ↓ to $\overline{HALT}$ ↑ or ↓	—	300	—	300	—	260
37	TwNMI	$\overline{NMI}$ Pulse Width	80	—	80	—	70	—
38	TsBUSREQ(Cr)	$\overline{BUSREQ}$ Setup Time to Clock ↑	80	—	50	—	50	—

*For clock periods other than the minimums shown in the table, calculate parameters using the expressions in the table on the following page.

AC Characteristics (Continued)

Number	Symbol	Parameter	Z80 CPU Min (ns)	Z80 CPU Max (ns)	Z80A CPU Min (ns)	Z80A CPU Max (ns)	Z80B CPU Min (ns)	Z80B CPU Max (ns)
39	ThBUSREQ(Cr)	$\overline{BUSREQ}$ Hold Time after Clock ↑	0	—	0	—	0	—
40	TdCr(BUSACKf)	Clock ↑ to $\overline{BUSACK}$ ↓ Delay	—	120	—	100	—	90
41	TdCf(BUSACKr)	Clock ↓ to $\overline{BUSACK}$ ↑ Delay	—	110	—	100	—	90
42	TdCr(Dz)	Clock ↑ to Data Float Delay	—	90	—	90	—	80
43	TdCr(CTz)	Clock ↑ to Control Outputs Float Delay ($\overline{MREQ}$, $\overline{IORQ}$, $\overline{RD}$, and $\overline{WR}$)	—	110	—	80	—	70
44	TdCr(Az)	Clock ↑ to Address Float Delay	—	110	—	90	—	80
45	TdCTr(A)	Address Stable after $\overline{MREQ}$ ↑, $\overline{IORQ}$ ↑, $\overline{RD}$ ↑, and $\overline{WR}$ ↑	160*	—	80*	—	35*	—
46	TsRESET(Cr)	$\overline{RESET}$ to Clock ↑ Setup Time	90	—	60	—	60	—
47	ThRESET(Cr)	$\overline{RESET}$ to Clock ↑ Hold Time	—	0	—	0	—	0
48	TsINT(Cr)	$\overline{INT}$ to Clock ↑ Setup Time	80	—	80	—	70	—
49	ThINTr(Cr)	$\overline{INT}$ to Clock ↑ Hold Time	—	0	—	0	—	0
50	TdM1f(IORQf)	$\overline{M1}$ ↓ to $\overline{IORQ}$ ↓ Delay	920*	—	565*	—	365*	—
51	TdCf(IORQf)	Clock ↓ to $\overline{IORQ}$ ↓ Delay	—	110	—	85	—	70
52	TdCf(IORQr)	Clock ↓ to $\overline{IORQ}$ ↑ Delay	—	100	—	85	—	70
53	TdCf(D)	Clock ↓ to Data Valid Delay	—	230	—	150	—	130

*For clock periods other than the minimums shown in the table, calculate parameters using the following expressions. Calculated values above assumed TrC = TfC = 20 ns.

Footnotes to AC Characteristics

Number	Symbol	Z80	Z80A	Z80B
1	TcC	TwCh + TwCl + TrC + TfC	TwCh + TwCl + TrC + TfC	TwCh + TwCl + TrC + TfC
2	TwCh	Although static by design, TwCh of greater than 200 μs is not guaranteed	Although static by design, TwCh of greater than 200 μs is not guaranteed	Although static by design, TwCh of greater than 200 μs is not guaranteed
7	TdA(MREQf)	TwCh + TfC − 75	TwCh + TfC − 65	TwCh + TfC − 50
10	TwMREQh	TwCh + TfC − 30	TwCh + TfC − 20	TwCh + TfC − 20
11	TwMREQl	TcC − 40	TcC − 30	TcC − 30
26	TdA(IORQf)	TcC − 80	TcC − 70	TcC − 55
29	TdD(WRf)	TcC − 210	TcC − 170	TcC − 140
31	TwWR	TcC − 40	TcC − 30	TcC − 30
33	TdD(WRf)	TwCl + TrC − 180	TwCl + TrC − 140	TwCl + TrC − 140
35	TdWRr(D)	TwCl + TrC − 80	TwCl + TrC − 70	TwCl + TrC − 55
45	TdCTr(A)	TwCl + TrC − 40	TwCl + TrC − 50	TwCl + TrC − 50
50	TdM1f(IORQf)	2TcC + TwCh + TfC − 80	2TcC + TwCh + TfC − 65	2TcC + TwCh + TfC − 50

AC Test Conditions:
V_{IH} = 2.0 V
V_{IL} = 0.8 V
V_{IHC} = V_{CC} −0.6 V
V_{ILC} = 0.45 V
V_{OH} = 2.0 V
V_{OL} = 0.8 V
FLOAT = ±0.5 V

Absolute Maximum Ratings

21

Storage Temperature ..... -65°C to +150°C
Temperature under Bias ..... Specified operating range
Voltages on all inputs and outputs with respect to ground ..-0.3 V to +7 V
Power Dissipation ..... 1.5 W

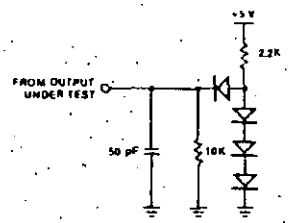
Stresses greater than those listed under Absolute Maximum Ratings may cause permanent damage to the device. This is a stress rating only; operation of the device at any condition above those indicated in the operational sections of these specifications is not implied. Exposure to absolute maximum rating conditions for extended periods may affect device reliability.

Standard Test Conditions

The characteristics below apply for the following standard test conditions, unless otherwise noted. All voltages are referenced to GND (0 V). Positive current flows into the referenced pin. Available operating temperature ranges are:

- 0°C to +70°C, +4.75 V ≤ V_{CC} ≤ +5.25 V
- -40°C to +85°C, +4.75 V ≤ V_{CC} ≤ +5.25 V
- -55°C to +125°C, +4.5 V ≤ V_{CC} ≤ +5.5 V

All ac parameters assume a load capacitance of 50 pF. Add 10 ns delay for each 50 pF increase in load up to a maximum of 200 pF for the data bus and 100 pF for address and control lines.



DC Characteristics

Symbol	Parameter	Min	Max	Unit	Test Condition
V_{ILC}	Clock Input Low Voltage	-0.3	0.45	V	
V_{IHC}	Clock Input High Voltage	V_{CC}-.6	V_{CC} + .3	V	
V_{IL}	Input Low Voltage	-0.3	0.8	V	
V_{IH}	Input High Voltage	2.0	V_{CC}	V	
V_{OL}	Output Low Voltage		0.4	V	I_{OL} = 1.8 mA
V_{OH}	Output High Voltage	2.4		V	I_{OH} = -250 μA
I_{CC}	Power Supply Current				
	Z80		150[1]	mA	
	Z80A		200[2]	mA	
	Z80B		200	mA	
I_{LI}	Input Leakage Current		10	μA	V_{IN} = 0 to V_{CC}
I_{LEAK}	3-State Output Leakage Current in Float	-10	10[3]	μA	V_{OUT} = 0.4 to V_{CC}

1. For military grade parts, I_{CC} is 200 mA.
2. Typical rate for Z80A is 90 mA.
3. A_{15}-A_0, D_7-D_0, $\overline{MREQ}$, $\overline{IORQ}$, $\overline{RD}$, and $\overline{WR}$.

Capacitance

Symbol	Parameter	Min	Max	Unit	Note
C_{CLOCK}	Clock Capacitance		35	pF	Unmeasured pins returned to ground
C_{IN}	Input Capacitance		5	pF	
C_{OUT}	Output Capacitance		10	pF	

T_A = 25°C, f = 1 MHz

MICROPROCESADORES Y MICROCOMPUTADORAS

Z8440
Z80 SIO Serial
Input/Output Controller

O C T U B R E, 1984

Z8440
Z80® SIO Serial Input/Output Controller

Zilog

Product Specification

March 1981

Features

- Two independent full-duplex channels, with separate control and status lines for modems or other devices.
- Data rates of 0 to 500K bits/second in the x1 clock mode with a 2.5 MHz clock (Z-80 SIO), or 0 to 800K bits/second with a 4.0 MHz clock (Z-80A SIO).
- Asynchronous protocols: everything necessary for complete messages in 5, 6, 7 or 8 bits/character. Includes variable stop bits and several clock-rate multipliers; break generation and detection; parity; overrun and framing error detection.
- Synchronous protocols: everything necessary for complete bit- or byte-oriented messages in 5, 6, 7 or 8 bits/character, including IBM Bisync, SDLC, HDLC, CCITT-X.25 and others. Automatic CRC generation/checking, sync character and zero insertion/deletion, abort generation/detection and flag insertion.
- Receiver data registers quadruply buffered, transmitter registers doubly buffered.
- Highly sophisticated and flexible daisy-chain interrupt vectoring for interrupts without external logic.

General Description

The Z-80 SIO Serial Input/Output Controller is a dual-channel data communication interface with extraordinary versatility and capability. Its basic functions as a serial-to-parallel, parallel-to-serial converter/controller can be programmed by a CPU for a broad range of serial communication applications.

The device supports all common asynchronous and synchronous protocols, byte- or bit-oriented, and performs all of the functions traditionally done by UARTs, USARTs and synchronous communication controllers combined, plus additional functions traditionally performed by the CPU. Moreover, it does this on two fully-independent channels, with an exceptionally sophisticated interrupt structure that allows very fast transfers.

Full interfacing is provided for CPU or DMA

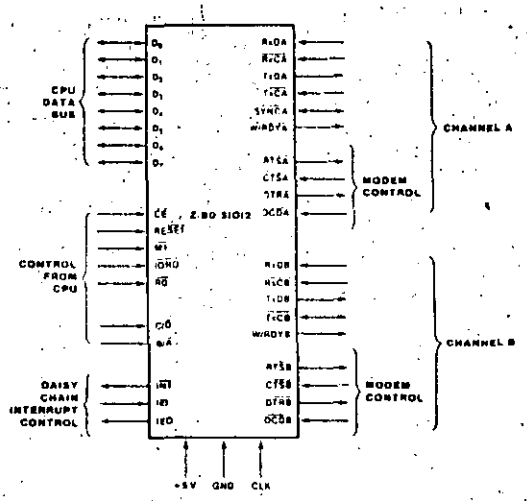


Figure 1. Z-80 SIO/2 Pin Functions

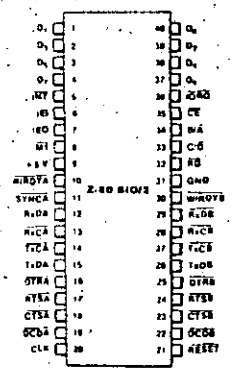


Figure 2. Z-80 SIO/2 Pin Assignments

General Description (Continued)

control. In addition to data communication, the circuit can handle virtually all types of serial I/O with fast (or slow) peripheral devices. While designed primarily as a member of the Z-80 family, its versatility makes it well suited to many other CPUs.

The Z-80 SIO is an n-channel silicon-gate depletion-load device packaged in a 40-pin plastic or ceramic DIP. It uses a single +5 V power supply and the standard Z-80 family single-phase clock.

Pin Description

Figures 1 through 6 illustrate the three pin configurations (bonding options) available in the SIO. The constraints of a 40-pin package make it impossible to bring out the Receive Clock ($\overline{RxC}$), Transmit Clock ($\overline{TxC}$), Data Terminal Ready ($\overline{DTR}$) and Sync ($\overline{SYNC}$) signals for both channels. Therefore, either Channel B lacks a signal or two signals are bonded together in the three bonding options offered:

- Z-80 SIO/2 lacks $\overline{SYNCB}$
- Z-80 SIO/1 lacks $\overline{DTRB}$
- Z-80 SIO/0 has all four signals, but $\overline{TxCB}$ and $\overline{RxCB}$ are bonded together

The first bonding option above (SIO/2) is the preferred version for most applications. The pin descriptions are as follows:

B/$\overline{A}$. *Channel A Or B Select* (input, High selects Channel B). This input defines which channel is accessed during a data transfer between the CPU and the SIO. Address bit A_0 from the CPU is often used for the selection function.

C/$\overline{D}$. *Control Or Data Select* (input, High selects Control). This input defines the type of information transfer performed between the CPU and the SIO. A High at this input during a CPU write to the SIO causes the information on the data bus to be interpreted as a command for the channel selected by B/$\overline{A}$. A Low at C/$\overline{D}$ means that the information on the data bus is data. Address bit A_1 is often used for this function.

$\overline{CE}$. *Chip Enable* (input, active Low). A Low level at this input enables the SIO to accept command or data input from the CPU during a write cycle or to transmit data to the CPU during a read cycle.

CLK. *System Clock* (input). The SIO uses the standard Z-80 System Clock to synchronize internal signals. This is a single-phase clock.

$\overline{CTSA}$, $\overline{CTSB}$. *Clear To Send* (inputs, active Low). When programmed as Auto Enables, a Low on these inputs enables the respective transmitter. If not programmed as Auto Enables, these inputs may be programmed as general-purpose inputs. Both inputs are Schmitt-trigger buffered to accommodate slow-risetime signals. The SIO detects pulses on these inputs and interrupts the CPU on both logic level transitions. The Schmitt-trigger buffering does not guarantee a specified noise-level margin.

D_0-D_7. *System Data Bus* (bidirectional, 3-state). The system data bus transfers data and commands between the CPU and the Z-80 SIO. D_0 is the least significant bit.

$\overline{DCDA}$, $\overline{DCDB}$. *Data Carrier Detect* (inputs, active Low). These pins function as receiver enables if the SIO is programmed for Auto Enables; otherwise they may be used as general-purpose input pins. Both pins are Schmitt-trigger buffered to accommodate slow-risetime signals. The SIO detects pulses on these pins and interrupts the CPU on both logic level transitions. Schmitt-trigger buffer-

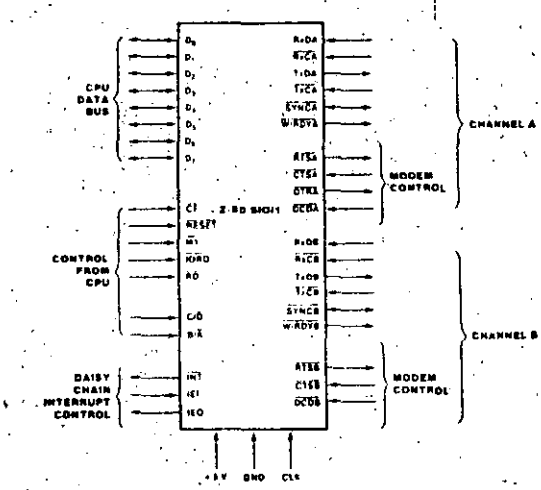


Figure 3. Z-80 SIO/1 Pin Functions

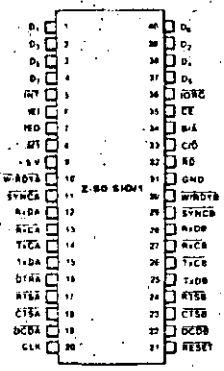


Figure 4. Z-80 SIO/1 Pin Assignments

Pin Description (Continued)

ing does not guarantee a specific noise-level margin.

DTRA, DTRB. *Data Terminal Ready* (outputs, active Low). These outputs follow the state programmed into Z-80 SIO. They can also be programmed as general-purpose outputs.

In the Z-80 SIO/1 bonding option, $\overline{DTRB}$ is omitted.

IEI. *Interrupt Enable In* (input, active High). This signal is used with IEO to form a priority daisy chain when there is more than one interrupt-driven device. A High on this line indicates that no other device of higher priority is being serviced by a CPU interrupt service routine.

IEO. *Interrupt Enable Out* (output, active High). IEO is High only if IEI is High and the CPU is not servicing an interrupt from this SIO. Thus, this signal blocks lower priority devices from interrupting while a higher priority device is being serviced by its CPU interrupt service routine.

$\overline{INT}$. *Interrupt Request* (output, open drain, active Low). When the SIO is requesting an interrupt, it pulls $\overline{INT}$ Low.

$\overline{IORQ}$. *Input/Output Request* (input from CPU, active Low). $\overline{IORQ}$ is used in conjunction with B/$\overline{A}$, C/$\overline{D}$, $\overline{CE}$ and $\overline{RD}$ to transfer commands and data between the CPU and the SIO. When $\overline{CE}$, $\overline{RD}$ and $\overline{IORQ}$ are all active, the channel selected by B/$\overline{A}$ transfers data to the CPU (a read operation). When $\overline{CE}$ and $\overline{IORQ}$ are active but $\overline{RD}$ is inactive, the channel selected by B/$\overline{A}$ is written to by the CPU with either data or control information as specified by C/$\overline{D}$. If $\overline{IORQ}$ and $\overline{M1}$ are active simultaneously, the CPU is acknowledging an interrupt and the SIO automatically places its interrupt vector on the CPU data bus if it is the highest priority device requesting an interrupt.

$\overline{M1}$. *Machine Cycle* (input from Z-80 CPU, active Low). When $\overline{M1}$ is active and $\overline{RD}$ is also active, the Z-80 CPU is fetching an instruction from memory; when $\overline{M1}$ is active while $\overline{IORQ}$ is active, the SIO accepts $\overline{M1}$ and $\overline{IORQ}$ as an interrupt acknowledge if the SIO is the highest priority device that has interrupted the Z-80 CPU.

$\overline{RxCA}$, $\overline{RxCB}$. *Receiver Clocks* (inputs). Receive data is sampled on the rising edge of $\overline{RxC}$. The Receive Clocks may be 1, 16, 32 or 64 times the data rate in asynchronous modes. These clocks may be driven by the Z-80 CTC Counter Timer Circuit for programmable baud rate generation. Both inputs are Schmitt-trigger buffered (no noise level margin is specified).

In the Z-80 SIO/0 bonding option, $\overline{RxCB}$ is bonded together with $\overline{TxCB}$.

$\overline{RD}$. *Read Cycle Status* (input from CPU, active Low). If $\overline{RD}$ is active, a memory or I/O read operation is in progress. $\overline{RD}$ is used with B/$\overline{A}$, $\overline{CE}$ and $\overline{IORQ}$ to transfer data from the SIO to the CPU.

RxDA, RxDB. *Receive Data* (inputs, active High). Serial data at TTL levels.

$\overline{RESET}$. *Reset* (input, active Low). A Low $\overline{RESET}$ disables both receivers and transmitters, forces TxDA and TxDB marking, forces the modem controls High and disables all interrupts. The control registers must be

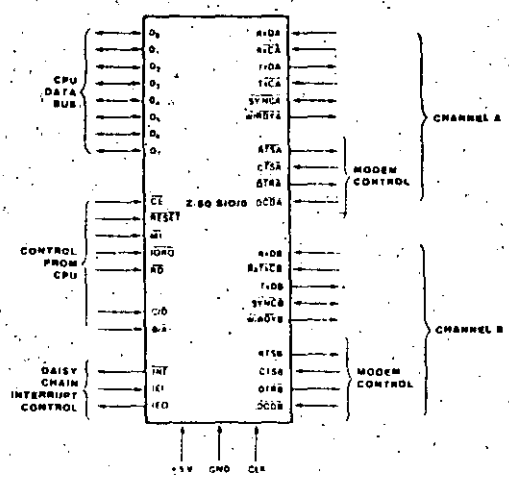


Figure 5. Z-80 SIO/0 Pin Functions

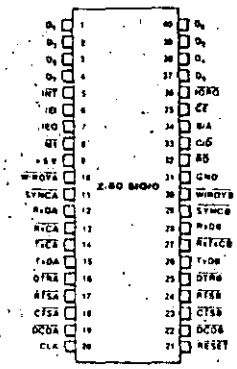


Figure 6. Z-80 SIO/0 Pin Assignments

Pin Description (Continued)

rewritten after the SIO is reset and before data is transmitted or received.

$\overline{RTSA}$, $\overline{RTSB}$. *Request To Send* (outputs, active Low). When the RTS bit in Write Register 5 (Figure 14) is set, the $\overline{RTS}$ output goes Low. When the RTS bit is reset in the Asynchronous mode, the output goes High after the transmitter is empty. In Synchronous modes, the $\overline{RTS}$ pin strictly follows the state of the RTS bit. Both pins can be used as general-purpose outputs.

$\overline{SYNCA}$, $\overline{SYNCB}$. *Synchronization* (inputs/outputs, active Low). These pins can act either as inputs or outputs. In the asynchronous receive mode, they are inputs similar to $\overline{CTS}$ and $\overline{DCD}$. In this mode, the transitions on these lines affect the state of the Sync/Hunt status bits in Read Register 0 (Figure 13), but have no other function. In the External Sync mode, these lines also act as inputs. When external synchronization is achieved, $\overline{SYNC}$ must be driven Low on the second rising edge of $\overline{RxC}$ after that rising edge of $\overline{RxC}$ on which the last bit of the sync character was received. In other words, after the sync pattern is detected, the external logic must wait for two full Receive Clock cycles to activate the $\overline{SYNC}$ input. Once $\overline{SYNC}$ is forced Low, it should be kept Low until the CPU informs the external synchronization detect logic that synchronization has been lost or a new message is about to start. Character assembly begins on the rising edge of $\overline{RxC}$ that immediately precedes the falling edge of $\overline{SYNC}$ in the External Sync mode.

In the internal synchronization mode (Monosync and Bisync), these pins act as outputs that are active during the part of the receive clock ($\overline{RxC}$) cycle in which sync characters are recognized. The sync condition is not latched, so these outputs are active each time a sync pattern is recognized, regardless of character boundaries.

In the Z-80 SIO/2 bonding option, $\overline{SYNCB}$ is omitted.

$\overline{TxCA}$, $\overline{TxCB}$. *Transmitter Clocks* (inputs). In asynchronous modes, the Transmitter Clocks may be 1, 16, 32 or 64 times the data rate; however, the clock multiplier for the transmitter and the receiver must be the same. The Transmit Clock inputs are Schmitt-trigger buffered for relaxed rise- and fall-time requirements (no noise level margin is specified). Transmitter Clocks may be driven by the Z-80 CTC Counter Timer Circuit for programmable baud rate generation.

In the Z-80 SIO/0 bonding option, $\overline{TxCB}$ is bonded together with $\overline{RxCB}$.

TxDA, TxDB. *Transmit Data* (outputs, active High). Serial data at TTL levels. TxD changes from the falling edge of $\overline{TxC}$.

$\overline{W/RDYA}$, $\overline{W/RDYB}$. *Wait/Ready A, Wait/Ready B* (outputs, open drain when programmed for Wait function, driven High and Low when programmed for Ready function). These dual-purpose outputs may be programmed as Ready lines for a DMA controller or as Wait lines that synchronize the CPU to the SIO data rate. The reset state is open drain.

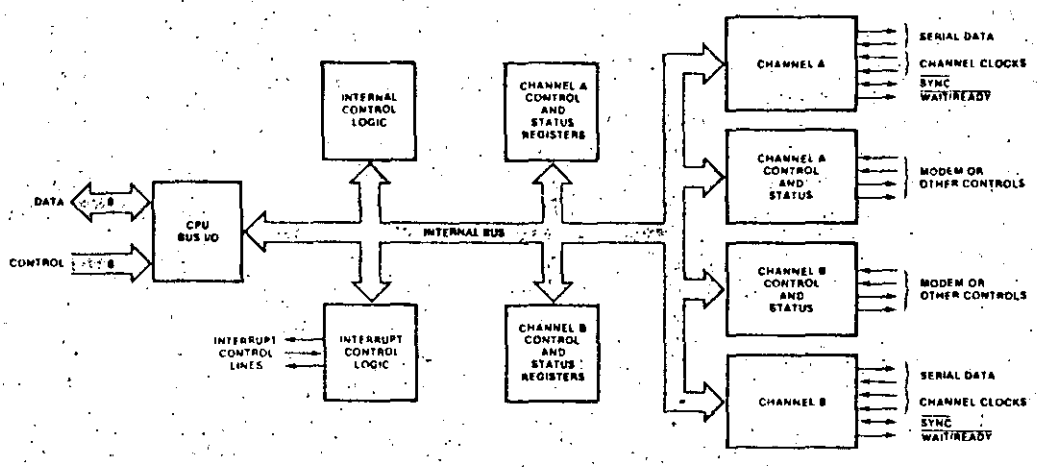


Figure 7. Block Diagram

Functional Description

The functional capabilities of the Z-80 SIO can be described from two different points of view: as a data communications device, it transmits and receives serial data in a wide variety of data-communication protocols; as a Z-80 family peripheral, it interacts with the Z-80 CPU and other peripheral circuits, sharing the data, address and control buses, as well as being a part of the Z-80 interrupt structure. As a peripheral to other microprocessors, the SIO offers valuable features such as non-vectored interrupts, polling and simple handshake capability.

5

Figure 8 illustrates the conventional devices that the SIO replaces.

The first part of the following discussion covers SIO data-communication capabilities; the second part describes interactions between the CPU and the SIO.

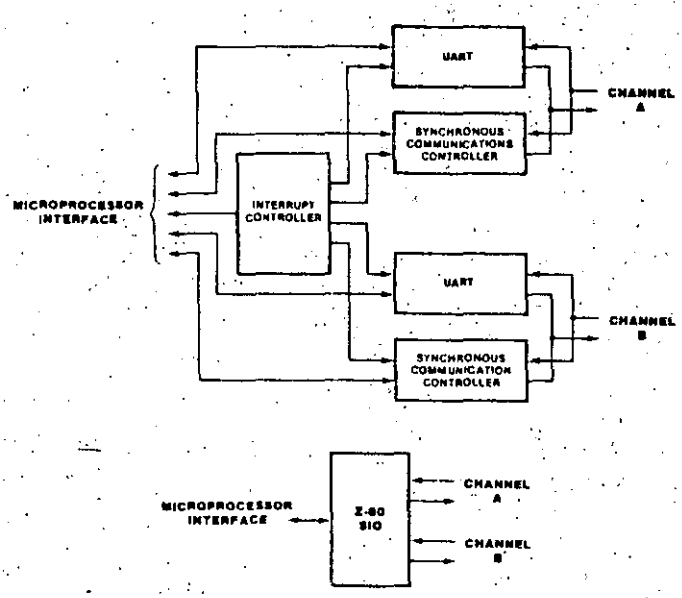


Figure 8. Conventional Devices Replaced by the Z-80 SIO

Data Communication Capabilities

The SIO provides two independent full-duplex channels that can be programmed for use in any common asynchronous or synchronous data-communication protocol. Figure 9 illustrates some of these protocols. The following is a short description of them. A more detailed explanation of these modes can be found in the *Z-80 SIO Technical Manual*.

Asynchronous Modes. Transmission and reception can be done independently on each channel with five to eight bits per character, plus optional even or odd parity. The transmitters can supply one, one-and-a-half or two stop bits per character and can provide a break output at any time. The receiver break-detection logic interrupts the CPU both at the start and end of a received break. Reception is protected from spikes by a transient spike-rejection mechanism that checks the signal one-half a bit time after a Low level is detected on the receive data input (RxDA or RxDB in Figure 5). If the Low does not persist—as in the case of a transient—the character assembly process is not started.

Framing errors and overrun errors are detected and buffered together with the partial character on which they occurred. Vectored interrupts allow fast servicing of error conditions using dedicated routines. Furthermore, a built-in checking process avoids interpreting a framing error as a new start bit: a framing error results in the addition of one-half a bit time to the point at which the search for the next start bit is begun.

The SIO does not require symmetric transmit and receive clock signals—a feature that allows it to be used with a Z-80 CTC or many other clock sources. The transmitter and receiver can handle data at a rate of 1, 1/16, 1/32 or 1/64 of the clock rate supplied to the receive and transmit clock inputs.

In asynchronous modes, the $\overline{\text{SYNC}}$ pin may be programmed as an input that can be used for functions such as monitoring a ring indicator.

Synchronous Modes. The SIO supports both byte-oriented and bit-oriented synchronous communication.

Synchronous byte-oriented protocols can be handled in several modes that allow character synchronization with an 8-bit sync character (Monosync), any 16-bit sync pattern (Bisync), or with an external sync signal. Leading sync

Data Communication Capabilities (Continued)

characters can be removed without interrupting the CPU.

Five-, six- or seven-bit sync characters are detected with 8- or 16-bit patterns in the SIO by overlapping the larger pattern across multiple in-coming sync characters, as shown in Figure 10.

CRC checking for synchronous byte-oriented modes is delayed by one character time so the CPU may disable CRC checking on specific characters. This permits implementation of protocols such as IBM Bisync.

Both CRC-16 ($X^{16} + X^{15} + X^2 + 1$) and CCITT ($X^{16} + X^{12} + X^5 + 1$) error checking polynomials are supported. In all non-SDLC modes, the CRC generator is initialized to 0's; in SDLC modes, it is initialized to 1's. The SIO can be used for interfacing to peripherals such as hard-sectored floppy disk, but it cannot generate or check CRC for IBM-compatible soft-sectored disks. The SIO also provides a feature that automatically transmits CRC data when no other data is available for transmission. This allows very high-speed transmissions under DMA control with no need for CPU intervention at the end of a message. When there is no data or CRC to send in synchronous modes, the transmitter inserts 8- or 16-bit sync characters regardless of the programmed character length.

The SIO supports synchronous bit-oriented protocols such as SDLC and HDLC by performing automatic flag sending, zero insertion and CRC generation. A special command can be used to abort a frame in transmission. At the end of a message the SIO automatically transmits the CRC and trailing flag when the transmit buffer becomes empty. If a transmit underrun occurs in the middle of a message, an external/status interrupt warns the CPU of this status change so that an abort may be issued. One to eight bits per character can be sent, which allows reception of a message with no prior information about the character structure in the information field of a frame.

The receiver automatically synchronizes on the leading flag of a frame in SDLC or HDLC, and provides a synchronization signal on the $\overline{SYNC}$ pin; an interrupt can also be programmed. The receiver can be programmed to search for frames addressed by a single byte to only a specified user-selected address or to a global broadcast address. In this mode, frames that do not match either the user-selected or broadcast address are ignored. The number of address bytes can be extended under software control. For transmitting data, an interrupt on the first received character or on every character can be selected. The receiver automatically deletes all zeroes inserted by the transmitter during character assembly. It also calculates and automatically checks the CRC to validate frame transmission. At the end of transmission, the status of a received frame is available in the status registers.

The SIO can be conveniently used under DMA control to provide high-speed reception or transmission. In reception, for example, the SIO can interrupt the CPU when the first character of a message is received. The CPU then enables the DMA to transfer the message to memory. The SIO then issues an end-of-frame interrupt and the CPU can check the status of the received message. Thus, the CPU is freed for other service while the message is being received.

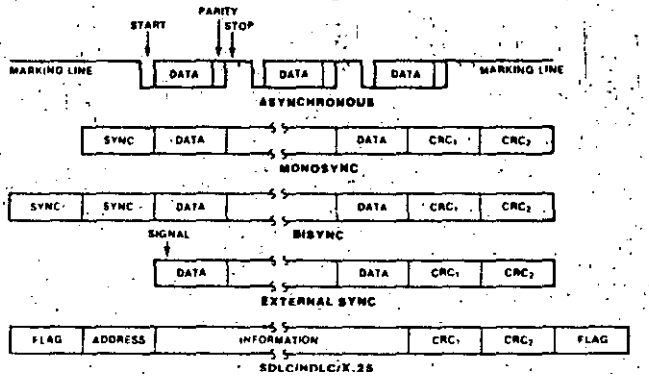


Figure 9. Some Z-80 SIO Protocols

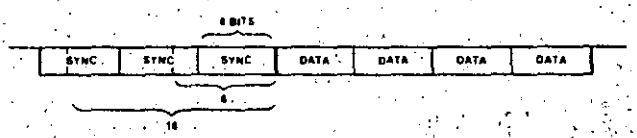


Figure 10.

I/O Interface Capabilities

The SIO offers the choice of polling, interrupt (vectored or non-vectored) and block-transfer modes to transfer data, status and control information to and from the CPU. The block-transfer mode can also be implemented under DMA control.

Polling. Two status registers are updated at appropriate times for each function being performed (for example, CRC error-status valid at the end of a message). When the CPU is operated in a polling fashion, one of the SIO's two status registers is used to indicate whether the SIO has some data or needs some data. Depending on the contents of this register, the CPU will either write data, read data, or just go on. Two bits in the register indicate that a data transfer is needed. In addition, error and other conditions are indicated. The second status register (special receive conditions) does not have to be read in a polling sequence, until a character has been received. All interrupt modes are disabled when operating the device in a polled environment.

Interrupts. The SIO has an elaborate interrupt scheme to provide fast interrupt service in real-time applications. A control register and a status register in Channel B contain the interrupt vector. When programmed to do so, the SIO can modify three bits of the interrupt vector in the status register so that it points directly to one of eight interrupt service routines in memory, thereby servicing conditions in both channels and eliminating most of the needs for a status-analysis routine.

Transmit interrupts, receive interrupts and external/status interrupts are the main sources of interrupts. Each interrupt source is enabled under program control, with Channel A having a higher priority than Channel B, and with receive, transmit and external/status interrupts prioritized in that order within each channel. When the transmit interrupt is enabled, the CPU is interrupted by the transmit buffer becoming empty. (This implies that the transmitter must have had a data character written into it so it can become empty.) The receiver can interrupt the CPU in one of two ways:

- Interrupt on first received character
- Interrupt on all received characters

Interrupt-on-first-received-character is typically used with the block-transfer mode. Interrupt-on-all-received-characters has the option of modifying the interrupt vector in the event of a parity error. Both of these interrupt modes will also interrupt under special receive conditions on a character or message basis (end-of-frame interrupt in SDLC, for example). This means that the special-receive condition can cause an interrupt only if the interrupt-on-first-received-character or interrupt-on-all-received-characters mode is selected. In interrupt-on-first-received-character, an interrupt can occur from special-receive conditions (except parity error) after the first-received-character interrupt (example: receive-overrun interrupt).

The main function of the external/status interrupt is to monitor the signal transitions of the Clear To Send ($\overline{CTS}$), Data Carrier Detect ($\overline{DCD}$) and Synchronization ($\overline{SYNC}$) pins (Figures 1 through 6). In addition, an external/status interrupt is also caused by a CRC-sending condition or by the detection of a break sequence (asynchronous mode) or abort sequence (SDLC mode) in the data stream. The interrupt caused by the break/abort sequence allows the SIO to interrupt when the break/abort sequence is detected or terminated. This feature facilitates the proper termination of the current message, correct initialization of the next message, and the accurate timing of the break/abort condition in external logic.

I/O Interface Capabilities (Continued)

In a Z-80 CPU environment (Figure 11), SIO interrupt vectoring is "automatic": the SIO passes its internally-modifiable 8-bit interrupt vector to the CPU, which adds an additional 8 bits from its interrupt-vector (I) register to form the memory address of the interrupt-routine table. This table contains the address of the beginning of the interrupt routine itself. The process entails an indirect transfer of CPU control to the interrupt routine, so that the next instruction executed after an interrupt acknowledge by the CPU is the first instruction of the interrupt routine itself.

CPU/DMA Block Transfer. The SIO's block-transfer mode accommodates both CPU block transfers and DMA controllers (Z-80 DMA or other designs). The block-transfer mode uses the Wait/Ready output signal, which is selected with three bits in an internal control register. The Wait/Ready output signal can be programmed as a $\overline{\text{WAIT}}$ line in the CPU block-transfer mode or as a $\overline{\text{READY}}$ line in the DMA block-transfer mode.

To a DMA controller, the SIO $\overline{\text{READY}}$ output indicates that the SIO is ready to transfer data to or from memory. To the CPU, the $\overline{\text{WAIT}}$ output indicates that the SIO is not ready to transfer data, thereby requesting the CPU to extend the I/O cycle.

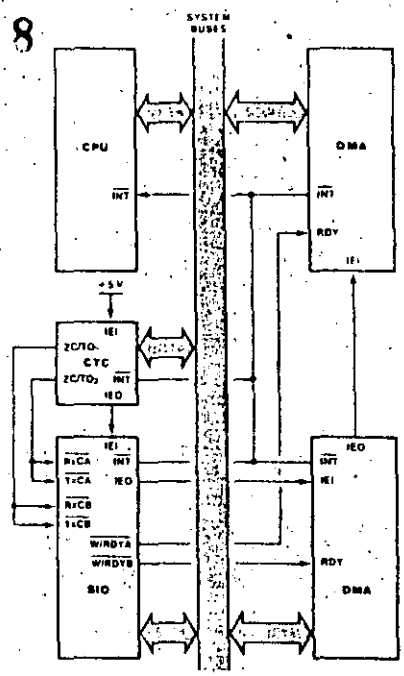


Figure 11. Typical Z-80 Environment

Internal Structure

The internal structure of the device includes a Z-80 CPU interface, internal control and interrupt logic, and two full-duplex channels. Each channel contains its own set of control and status (write and read) registers, and control and status logic that provides the interface to modems or other external devices.

The registers for each channel are designated as follows:

WR0-WR7 — Write Registers 0 through 7
RR0-RR2 — Read Registers 0 through 2

The register group includes five 8-bit control registers, two sync-character registers and two status registers. The interrupt vector is written into an additional 8-bit register (Write Register 2) in Channel B that may be read through another 8-bit register (Read Register 2) in Channel B. The bit assignment and functional grouping of each register is configured to simplify and organize the programming process. Table 1 lists the functions assigned to each read or write register.

Read Register Functions	
RR0	Transmit/Receive buffer status, interrupt status and external status
RR1	Special Receive Condition status
RR2	Modified interrupt vector (Channel B only)

Write Register Functions	
WR0	Register pointers, CRC initialize, initialization commands for the various modes, etc.
WR1	Transmit/Receive interrupt and data transfer mode definition.
WR2	Interrupt vector (Channel B only)
WR3	Receive parameters and control
WR4	Transmit/Receive miscellaneous parameters and modes
WR5	Transmit parameters and controls
WR6	Sync character or SDLC address field
WR7	Sync character or SDLC flag

Internal Structure (Continued)

The logic for both channels provides formats, synchronization and validation for data transferred to and from the channel interface. The modem control inputs, Clear To Send ($\overline{CTS}$) and Data Carrier Detect ($\overline{DCD}$), are monitored by the external control and status logic under program control. All external control-and-status-logic signals are general-purpose in nature and can be used for functions other than modem control.

Data Path. The transmit and receive data path illustrated for Channel A in Figure 12 is identical for both channels. The receiver has three 8-bit buffer registers in a FIFO arrangement, in addition to the 8-bit receive shift register. This scheme creates additional time for the CPU to service an interrupt at the beginning of a block of high-speed data. Incoming data is routed through one of several paths (data or CRC) depending on the selected mode and—in asynchronous modes—the character length.

The transmitter has an 8-bit transmit data buffer register that is loaded from the internal data bus, and a 20-bit transmit shift register that can be loaded from the sync-character buffers or from the transmit data register. Depending on the operational mode, outgoing data is routed through one of four main paths before it is transmitted from the Transmit Data output (TxD).

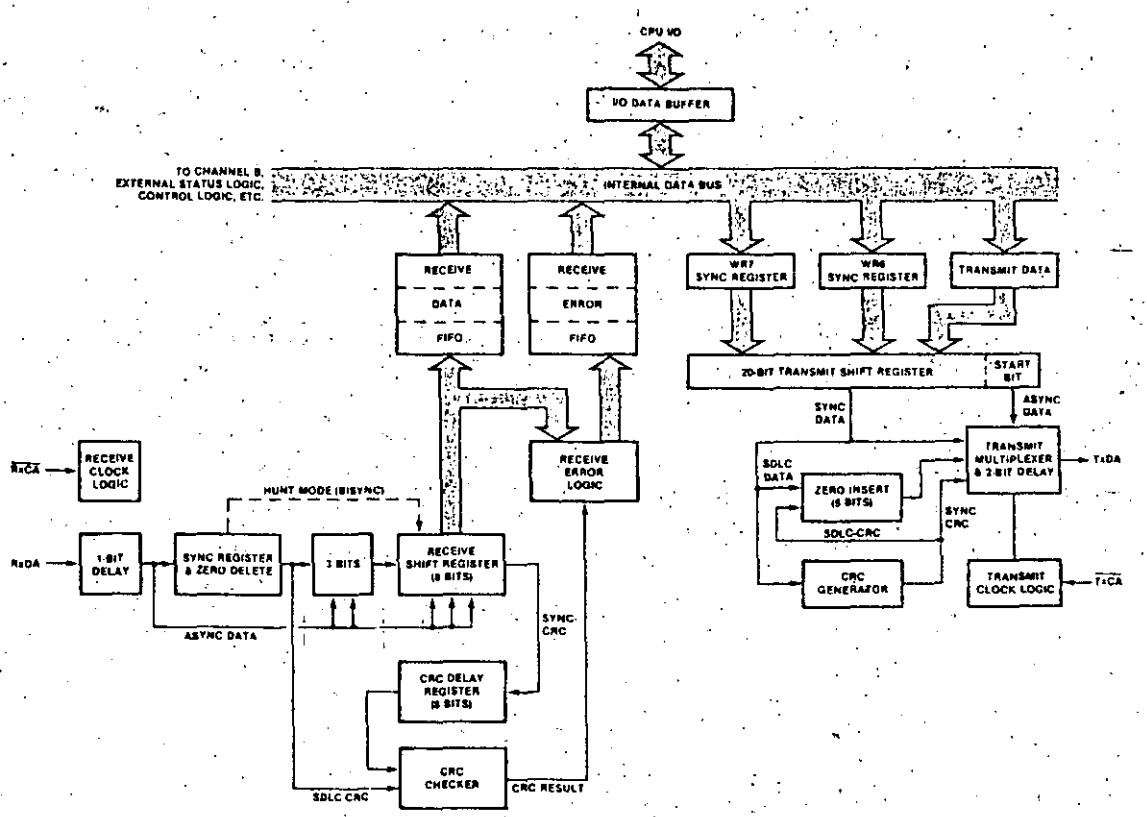


Figure 12. Transmit and Receive Data Path (Channel A)

Programming

The system program first issues a series of commands that initialize the basic mode of operation and then other commands that qualify conditions within the selected mode. For example, the asynchronous mode; character length, clock rate, number of stop bits, even or odd parity might be set first; then the interrupt mode; and finally, receiver or transmitter enable.

Both channels contain registers that must be programmed via the system program prior to operation. The channel-select input (B/$\overline{A}$) and the control/data input (C/$\overline{D}$) are the command-structure addressing controls, and are normally controlled by the CPU address bus. Figures 15 and 16 illustrate the timing relationships for programming the write registers and transferring data and status.

Read Registers. The SIO contains three read registers for Channel B and two read registers for Channel A (RR0-RR2 in Figure 13) that can be read to obtain the status information; RR2 contains the internally-modifiable interrupt vector and is only in the Channel B register set. The status information includes error conditions, interrupt vector and standard communications-interface signals.

To read the contents of a selected read register other than RR0, the system program must first write the pointer byte to WR0 in exactly the same way as a write register operation. Then, by executing a read instruction, the contents of the addressed read register can be read by the CPU.

The status bits of RR0 and RR1 are carefully grouped to simplify status monitoring. For example, when the interrupt vector indicates that a Special Receive Condition interrupt has occurred, all the appropriate error bits can be read from a single register (RR1).

Write Registers. The SIO contains eight write registers for Channel B and seven write registers for Channel A (WR0-WR7 in Figure 14) that are programmed separately to configure the functional personality of the channels; WR2 contains the interrupt vector for both channels and is only in the Channel B register set. With the exception of WR0, programming the write registers requires two bytes. The first byte is to WR0 and contains three bits (D_0-D_2) that point to the selected register; the second byte is the actual control word that is written into the register to configure the SIO.

WR0 is a special case in that all of the basic commands can be written to it with a single byte. Reset (internal or external) initializes the pointer bits D_0-D_2 to point to WR0. This implies that a channel reset must not be combined with the pointing to any register.

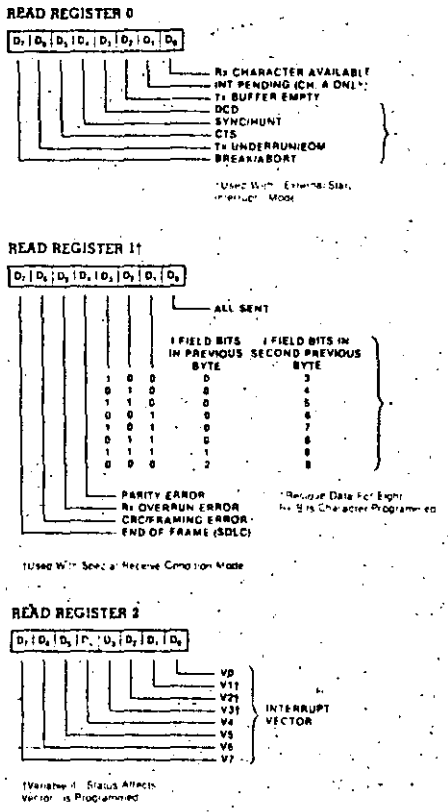


Figure 13. Read Register Bit Functions

Programming (Continued)

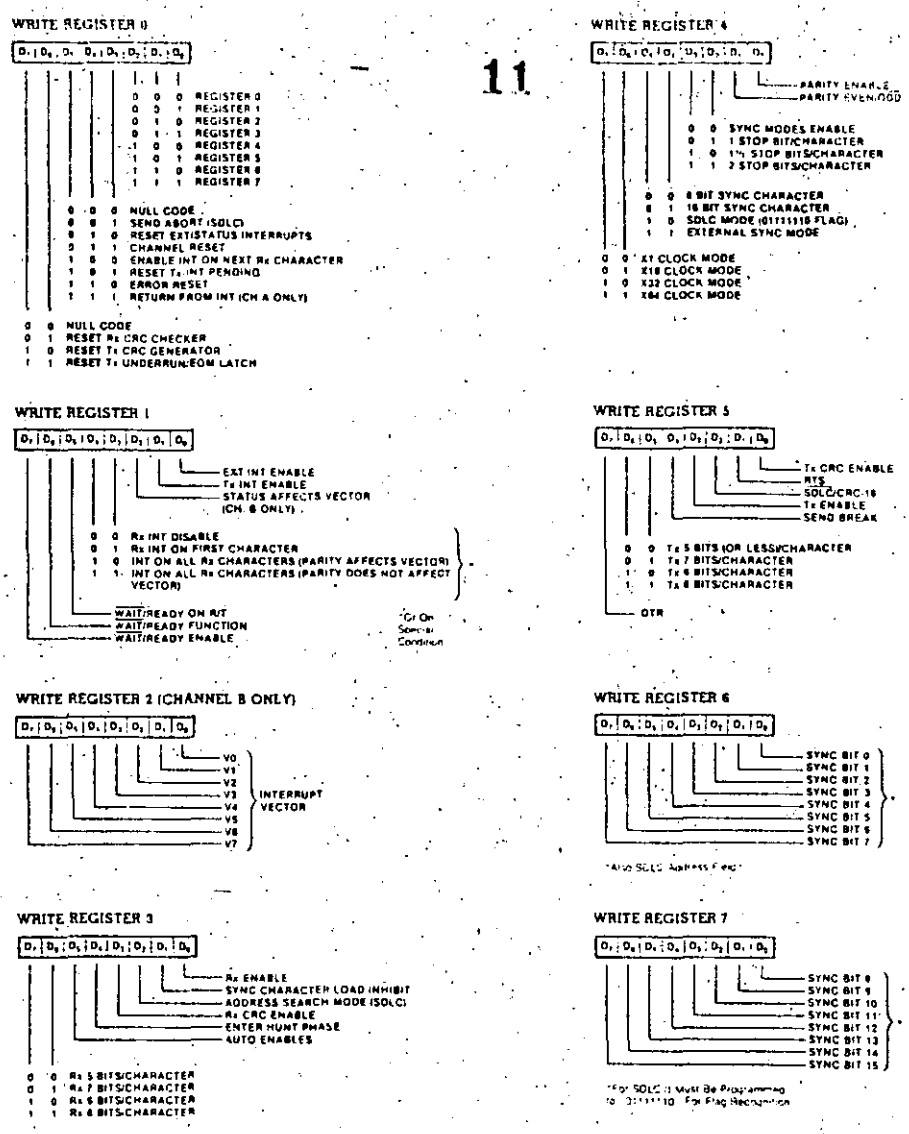


Figure 14. Write Register Bit Functions

The SIO must have the same clock as the CPU (same phase and frequency relationship, not necessarily the same driver).

Read Cycle. The timing signals generated by a Z-80 CPU input instruction to read a data or status byte from the SIO are illustrated in Figure 15.

Write Cycle. Figure 16 illustrates the timing and data signals generated by a Z-80 CPU output instruction to write a data or control byte into the SIO.

Interrupt-Acknowledge Cycle. After receiving an interrupt-request signal from an SIO ($\overline{INT}$ pulled Low), the Z-80 CPU sends an interrupt-acknowledge sequence ($\overline{M1}$ Low, and $\overline{IORQ}$ Low a few cycles later) as in Figure 17.

The SIO contains an internal daisy-chained interrupt structure for prioritizing nested interrupts for the various functions of its two channels, and this structure can be used within an external user-defined daisy chain that prioritizes several peripheral circuits.

The IEI of the highest-priority device is terminated High. A device that has an interrupt pending or under service forces its IEO Low. For devices with no interrupt pending or under service, IEO = IEI.

To insure stable conditions in the daisy chain, all interrupt status signals are prevented from changing while $\overline{M1}$ is Low. When $\overline{IORQ}$ is Low, the highest priority interrupt requestor (the one with IEI High) places its interrupt vector on the data bus and sets its internal interrupt-under-service latch.

Return From Interrupt Cycle. Figure 18 illustrates the return from interrupt cycle. Normally, the Z-80 CPU issues a RETI (Return From Interrupt) instruction at the end of an interrupt service routine. RETI is a 2-byte opcode (ED-4D) that resets the interrupt-under-service latch in the SIO to terminate the interrupt that has just been processed. This is accomplished by manipulating the daisy chain in the following way.

The normal daisy-chain operation can be used to detect a pending interrupt; however, it cannot distinguish between an interrupt under service and a pending unacknowledged interrupt of a higher priority. Whenever "ED" is decoded, the daisy chain is modified by forcing High the IEO of any interrupt that has not yet been acknowledged. Thus the daisy chain identifies the device presently under service as the only one with an IEI High and an IEO Low. If the next opcode byte is "4D," the interrupt-under-service latch is reset.

The ripple time of the interrupt daisy chain (both the High-to-Low and the Low-to-High transitions) limits the number of devices that can be placed in the daisy chain. Ripple time can be improved with carry-look-ahead, or by extending the interrupt-acknowledge cycle. For further information about techniques for increasing the number of daisy-chained devices, refer to the *Z-80 CPU Product Specification*.

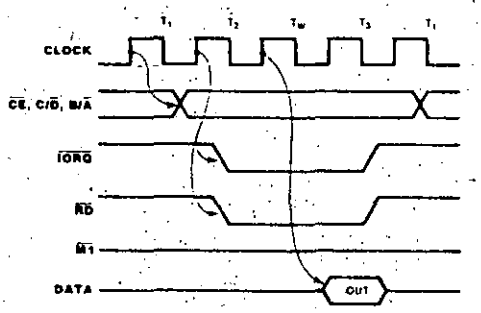


Figure 15. Read Cycle

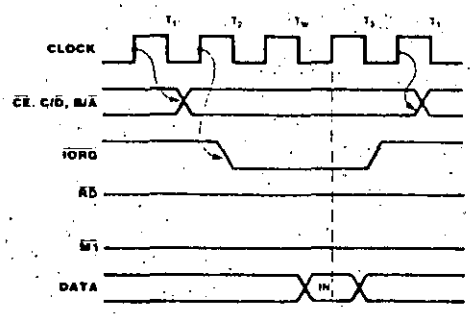


Figure 16. Write Cycle

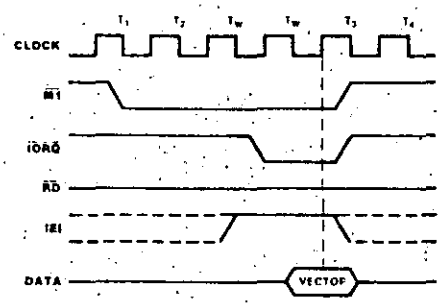


Figure 17. Interrupt Acknowledge Cycle

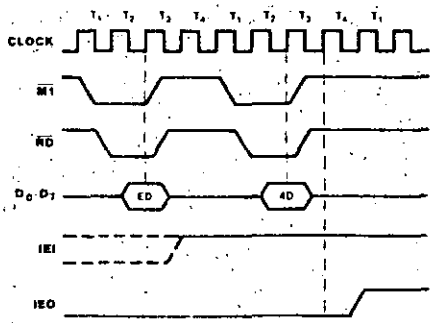


Figure 18. Return from Interrupt Cycle

Absolute Maximum Ratings

Voltages on all inputs and outputs with respect to GND . . . . . -0.3 V to +7.0 V

Operating Ambient Temperature . . . . . As Specified in Ordering Information

Storage Temperature . . . . . -65°C to +150°C

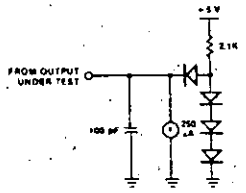
Stresses greater than those listed under Absolute Maximum Ratings may cause permanent damage to the device. This is a stress rating only; operation of the device at any condition above those indicated in the operational sections of these specifications is not implied. Exposure to absolute maximum rating conditions for extended periods may affect device reliability.

Test Conditions

The characteristics below apply for the following test conditions, unless otherwise noted. All voltages are referenced to GND (0 V). Positive current flows into the referenced pin. Available operating-temperature ranges are:

- 0°C to +70°C, $+4.75\ V \le V_{CC} \le +5.25\ V$
- −40°C to +85°C, $+4.75\ V \le V_{CC} \le +5.25\ V$
- −55°C to +125°C, $+4.5\ V \le V_{CC} \le +5.5\ V$

The product number for each operating temperature range may be found in the ordering information section.



DC Characteristics

Symbol	Parameter	Min	Max	Unit	Test Condition
V_{ILC}	Clock Input Low Voltage	-0.3	+0.45	V	
V_{IHC}	Clock Input High Voltage	V_{CC}-0.6	+5.5	V	
V_{IL}	Input Low Voltage	-0.3	+0.8	V	
V_{IH}	Input High Voltage	+2.0	+5.5	V	
V_{OL}	Output Low Voltage		+0.4	V	I_{OL} = 2.0 mA
V_{OH}	Output High Voltage	+2.4		V	I_{OH} = -250 µA
I_{LI}	Input Leakage Current	-10	+10	µA	$0 < V_{IN} < V_{CC}$
I_Z	3-State Output/Data Bus Input Leakage Current	-10	+10	µA	$0 < V_{IN} < V_{CC}$
$I_{L(SY)}$	$\overline{SYNC}$ Pin Leakage Current	-40	+10	µA	$0 < V_{IN} < V_{CC}$
I_{CC}	Power Supply Current		100	mA	

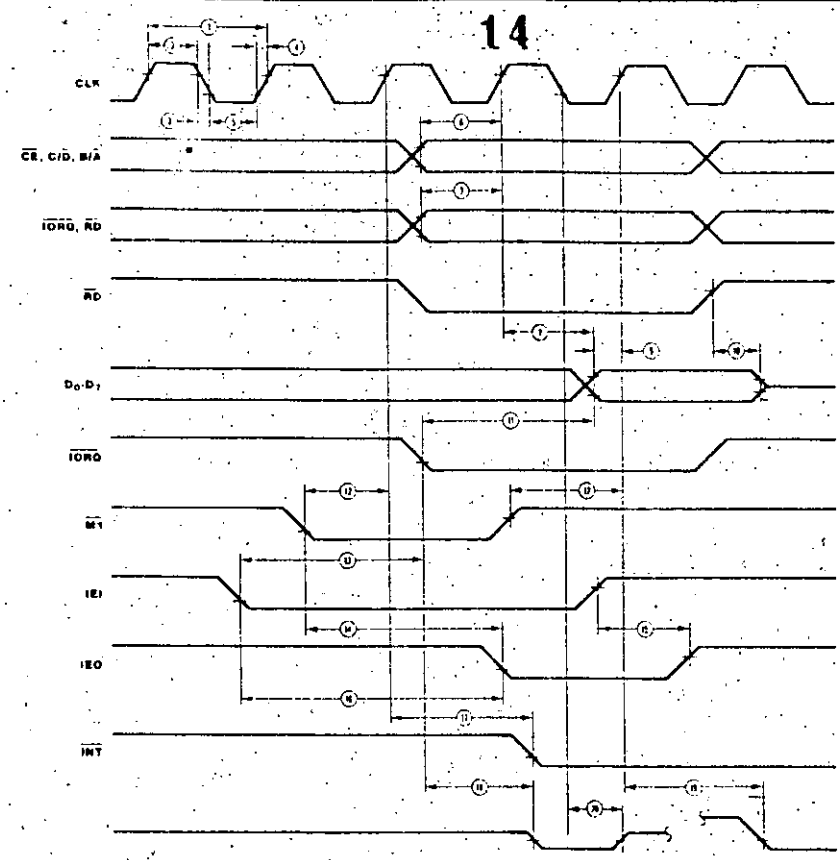
Over specified temperature and voltage range.

Capacitance

Symbol	Parameter	Min	Max	Unit	Test Condition
C	Clock Capacitance		40	pF	Unmeasured
C_{IN}	Input Capacitance		5	pF	pins returned
C_{OUT}	Output Capacitance		10	pF	to ground

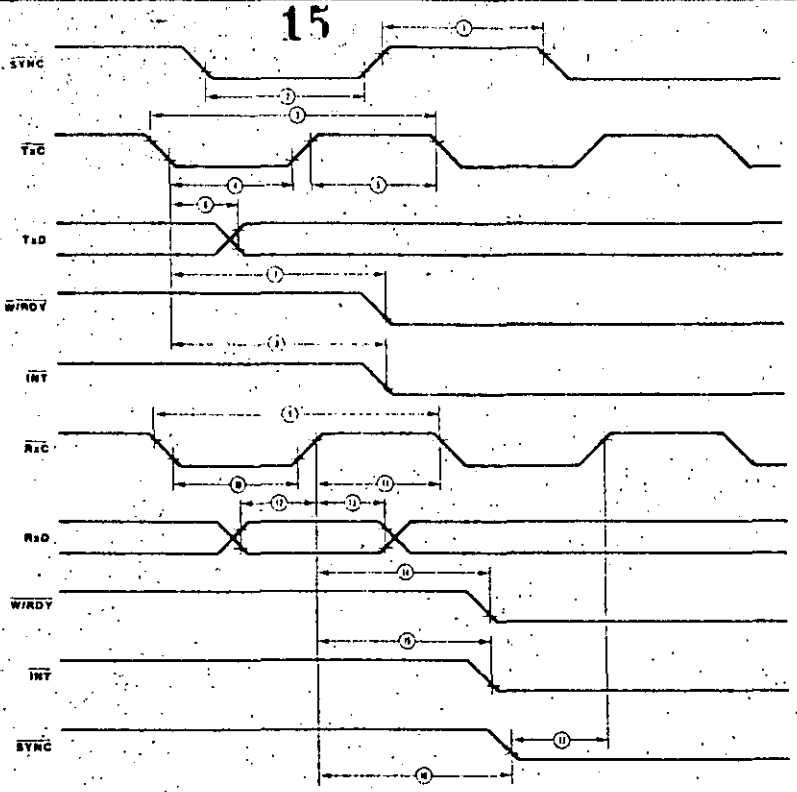
Over specified temperature range; f = 1 MHz

AC Electrical Characteristics



Number	Symbol	Parameter	Z-80 SIO Min	Z-80 SIO Max	Z-80A SIO Min	Z-80A SIO Max	Z-80B SIO Min	Z-80B SIO Max	Unit
1	TcC	Clock Cycle Time	400	4000	250	4000	165	4000	ns
2	TwCh	Clock Width (High)	170	2000	105	2000	70	2000	ns
3	TfC	Clock Fall Time		30		30		15	ns
4	TrC	Clock Rise Time		30		30		15	ns
5	TwCl	Clock Width (Low)	170	2000	105	2000	70	2000	ns
6	TsAD(C)	$\overline{CE}$, C/$\overline{D}$, B/$\overline{A}$ to Clock ↑ Setup Time	160		145		60		ns
7	TsCS(C)	$\overline{IORQ}$, $\overline{RD}$ to Clock ↑ Setup Time	240		115		60		ns
8	TdC(DO)	Clock ↑ to Data Out Delay		240		220		150	ns
9	TsDI(C)	Data In to Clock ↑ Setup (Write or $\overline{M1}$ Cycle)	50		50		30		ns
10	TdRD(DOz)	$\overline{RD}$ ↑ to Data Out Float Delay		230		110		90	ns
11	TdIO(DOI)	$\overline{IORQ}$ ↓ to Data Out Delay (INTACK Cycle)		340		160		100	ns
12	TsM1(C)	$\overline{M1}$ to Clock ↑ Setup Time	210		90		75		ns
13	TsIEI(IO)	IEI to $\overline{IORQ}$ ↓ Setup Time (INTACK Cycle)	200		140		120		ns
14	TdM1(IEO)	$\overline{M1}$ ↓ to IEO ↓ Delay (interrupt before $\overline{M1}$)		300		190	160		ns
15	TdIEI(IEOr)	IEI ↑ to IEO ↑ Delay (after ED decode)		150		100		70	ns
16	TdIEI(IEOf)	IEI ↓ to IEO ↓ Delay		150		100		70	ns
17	TdC(INT)	Clock ↑ to $\overline{INT}$ ↓ Delay		200		200		150	ns
18	TdIO(W/RWf)	$\overline{IORQ}$ ↓ or $\overline{CE}$ ↓ to $\overline{W/RDY}$ ↓ Delay Wait Mode)		300		210		175	ns
19	TdC(W/RR)	Clock ↑ to $\overline{W/RDY}$ ↓ Delay (Ready Mode)		120		120		100	ns
20	TdC(W/RWz)	Clock ↓ to $\overline{W/RDY}$ Float Delay (Wait Mode)		150		130		110	ns
21	Th	Any unspecified Hold when Setup is specified	0		0		0		ns

AC Electrical Characteristics (Continued)



Number	Symbol	Parameter	Z-80 SIO Min	Z-80 SIO Max	Z-80A SIO Min	Z-80A SIO Max	Z-80B SIO Min	Z-80B SIO Max	Unit
1	TwPh	Pulse Width (High)	200		200		200		ns
2	TwPl	Pulse Width (Low)	200		200		200		ns
3	TcTxC	$\overline{TxC}$ Cycle Time	400	∞	400	∞	330	∞	ns
4	TwTxCl	$\overline{TxC}$ Width (Low)	180	∞	180	∞	100	∞	ns
5	TwTxCh	$\overline{TxC}$ Width (High)	180	∞	180	∞	100	∞	ns
6	TdTxC(TxD)	$\overline{TxC}$ ↓ to TxD Delay (x1 Mode)		400		300		220	ns
7	TdTxC(W/RRI)	$\overline{TxC}$ ↓ to $\overline{W/RDY}$ ↑ Delay (Ready Mode)	5	9	5	9	5	9	Clk Periods*
8	TdTxC(INT)	$\overline{TxC}$ ↓ to $\overline{INT}$ ↓ Delay	5	9	5	9	5	9	Clk Periods*
9	TcRxC	$\overline{RxC}$ Cycle Time	400	∞	400	∞	330	∞	ns
10	TwRxCl	$\overline{RxC}$ Width (Low)	180	∞	180	∞	100	∞	ns
11	TwRxCh	$\overline{RxC}$ Width (High)	180	∞	180	∞	100	∞	ns
12	TsRxD(RxC)	RxD to $\overline{RxC}$ ↑ Setup Time (x1 Mode)	0		0		0		ns
13	ThRxD(RxC)	$\overline{RxC}$ ↑ to RxD Hold Time (x1 Mode)	140		140		100		ns
14	TdRxC(W/RRI)	$\overline{RxC}$ ↑ to $\overline{W/RDY}$ ↑ Delay (Ready Mode)	10	13	10	13	10	13	Clk Periods*
15	TdRxC(INT)	$\overline{RxC}$ ↑ to $\overline{INT}$ ↑ Delay	10	13	10	13	10	13	Clk Periods*
16	TdRxC(SYNC)	$\overline{RxC}$ ↑ to $\overline{SYNC}$ ↑ Delay (Output Modes)	4	7	4	7	4	7	Clk Periods*
17	TsSYNC(RxC)	$\overline{SYNC}$ ↑ to $\overline{RxC}$ ↑ Setup (External Sync Modes)	-100		-100			100	ns

In all modes, the System Clock rate must be at least five times the maximum data rate.
$\overline{RESET}$ must be active a minimum of one complete Clock Cycle.
*System Clock

Ordering Information

Product Number	Package/ Temp	Speed	Description
Z8440	CE,CM	2.5 MHz	Z80 SIO/0 (40-pin)
Z8440	CMB,CS	2.5 MHz	Same as above
Z8440	DE,DS	2.5 MHz	Same as above
Z8440	PE,PS	2.5 MHz	Same as above
Z8440A	CE,CM	4.0 MHz	Z80A SIO/0 (40-pin)
Z8440A	CMB,CS	4.0 MHz	Same as above
Z8440A	DE,DS	4.0 MHz	Same as above
Z8440A	PE,PS	4.0 MHz	Same as above
Z8440B	CE,CM	6.0 MHz	Z80B SIO/0 (40-pin)
Z8440B	CMB,CS	6.0 MHz	Same as above
Z8440B	DE,DS	6.0 MHz	Same as above
Z8440B	PE,PS	6.0 MHz	Same as above
Z8441	CE,CM	2.5 MHz	Z80 SIO/1 (40-pin)
Z8441	CMB,CS	2.5 MHz	Same as above
Z8441	DE,DS	2.5 MHz	Same as above
Z8441	PE,PS	2.5 MHz	Same as above
Z8441A	CE,CM	4.0 MHz	Z80A SIO/1 (40-pin)
Z8441A	CMB,CS	4.0 MHz	Same as above
Z8441A	DE,DS	4.0 MHz	Z80A SIO/1 (40-pin)
Z8441A	PE,PS	4.0 MHz	Same as above
Z8441B	CE,CM	6.0 MHz	Z80B SIO/1 (40-pin)
Z8441B	CMB,CS	6.0 MHz	Same as above
Z8441B	DE,DS	6.0 MHz	Same as above
Z8441B	PE,PS	6.0 MHz	Same as above
Z8442	CE,CM	2.5 MHz	Z80 SIO/2 (40-pin)
Z8442	CMB,CS	2.5 MHz	Same as above
Z8442	DE,DS	2.5 MHz	Same as above
Z8442	PE,PS	2.5 MHz	Same as above
Z8442A	CE,CM	4.0 MHz	Z80A SIO/2 (40-pin)
Z8442A	CMB,CS	4.0 MHz	Same as above
Z8442A	DE,DS	4.0 MHz	Same as above
Z8442A	PE,PS	4.0 MHz	Same as above
Z8442B	CE,CM	6.0 MHz	Z80B SIO/2 (40-pin)
Z8442B	CMB,CS	6.0 MHz	Same as above
Z8442B	DE,DS	6.0 MHz	Same as above
Z8442B	PE,PS	6.0 MHz	Same as above

NOTES: C = Ceramic, D = Cerdip, P = Plastic; E = -40°C to +85°C, M = -55°C to +125°C, MB = -55°C to +125°C with MIL-STD-883 with Class B processing, S = 0°C to +70°C.

Z8449
17 Z80® SIO/9 Serial Input/Output Controller

Zilog

Product Specification

March 1981

Features

- One full-duplex channel, with separate control and status lines for a modem or other device.
- Data rates of 0 to 500K bits/second in the x1 clock mode with a 2.5 MHz clock (Z-80 SIO), or 0 to 800K bits/second with a 4.0 MHz clock (Z-80A SIO).
- Asynchronous protocols: everything necessary for complete messages in 5, 6, 7 or 8 bits/character. Includes variable stop bits and several clock-rate multipliers; break generation and detection; parity, overrun and framing error detection.
- Receiver data registers quadruply buffered, transmitter registers doubly buffered.
- Synchronous protocols: everything necessary for complete bit- or byte-oriented messages in 5, 6, 7 or 8 bits/character, including IBM Bisync, SDLC, HDLC, CCITT-X.25 and others. Automatic CRC generation/checking, sync character and zero insertion/deletion, abort generation/detection and flag insertion.
- Highly sophisticated and flexible daisy-chain interrupt vectoring for interrupts without external logic.

General Description

The Z-80 SIO/9 Serial Input/Output Controller is a single-channel data communication interface with extraordinary versatility and capability. Functionally this device is identical to the Z-80 SIO, except that it operates in one channel only. Its basic functions as a serial-to-parallel, parallel-to-serial converter/controller can be programmed by a CPU for a broad range of serial communication applications.

The device supports all common asynchronous and synchronous protocols, byte- or bit-oriented, and performs all of the functions

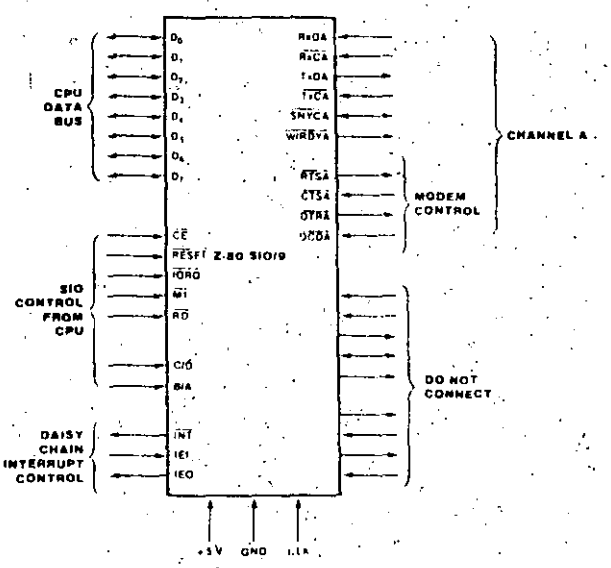


Figure 1. Z-80 SIO/9 Pin Functions

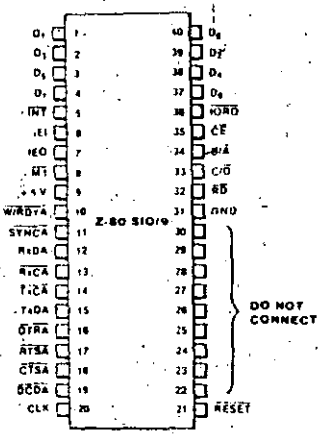


Figure 2. Z-80 SIO/9 Pin Assignments

General Description (Continued)

traditionally done by UARTs, USARTs and synchronous communication controllers combined, plus additional functions traditionally performed by the CPU. Moreover, it does this with an exceptionally sophisticated interrupt structure that allows very fast transfers.

Full interfacing is provided for CPU and DMA control. In addition to data communication, the circuit can handle virtually all types of serial I/O with fast (or slow) peripheral devices. While designed primarily as a member of the Z-80 family, its versatility makes it well-suited to many other CPUs.

The Z-80 SIO/9 is an n-channel silicon-gate depletion-load device packaged in a 40-pin plastic or ceramic DIP. It uses a single +5 V power supply and standard Z-80 family single-phase clock.

Refer to the *Z-80 SIO Product Specification* and the *Z-80 SIO Technical Manual* for detailed functional and electrical descriptions. All functional and electrical descriptions in these publications are applicable to the Z-80 SIO/9, except that Channel B cannot be used for data input or output and pins 22 through 30 must not be connected.

Write Register 2 (interrupt vector) and the Status Affects Vector bit in Write Register 1 are, however, still programmed by selecting Channel B with the B/$\overline{A}$ input. All other bits in Write Register 1 or Channel B must be programmed to 0.

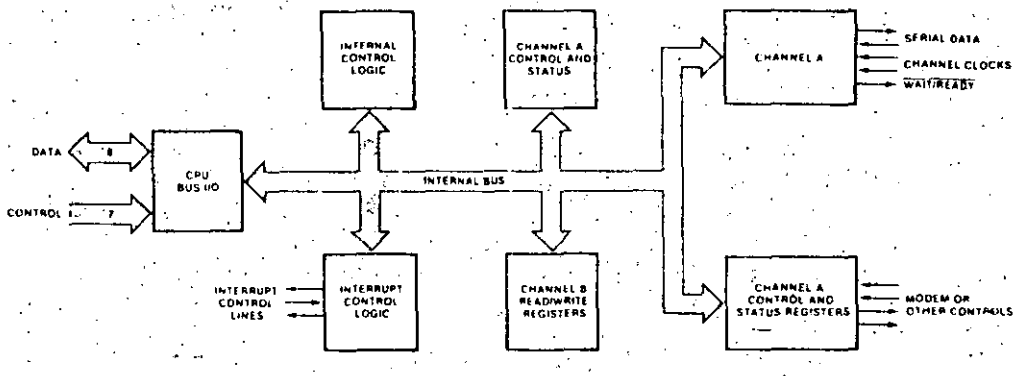


Figure 3. Block Diagram

Ordering Information

Product Number	Package/ Temp	Speed	Description	Product Number	Package/ Temp	Speed	Description
Z8449	CE	2.5 MHz	Z80 SIO/9 (40-pin)	Z8449A	DE	4.0 MHz	Z80A SIO/9 (40-pin)
Z8449	CM	2.5 MHz	Same as above	Z8499A	DS	4.0 MHz	Same as above
Z8449	CMB	2.5 MHz	Same as above	Z8449A	PE	4.0 MHz	Same as above
Z8449	CS	2.5 MHz	Same as above	Z8449A	PS	4.0 MHz	Same as above
Z8449	DE	2.5 MHz	Same as above	Z8449B	CE	6.0 MHz	Z80B SIO/9 (40-pin)
Z8449	DS	2.5 MHz	Same as above	Z8449B	CM	6.0 MHz	Same as above
Z8449	PE	2.5 MHz	Same as above	Z8449B	CMB	6.0 MHz	Same as above
Z8449	PS	2.5 MHz	Same as above	Z8449B	CS	6.0 MHz	Same as above
Z8449A	CE	4.0 MHz	Z80A SIO/9 40-pin)	Z8449B	DE	6.0 MHz	Same as above
Z8449A	CM	4.0 MHz	Same as above	Z8449B	DS	6.0 MHz	Same as above
Z8449A	CMB	4.0 MHz	Same as above	Z8449B	PE	6.0 MHz	Same as above
Z8449A	CS	4.0 MHz	Same as above	Z8449B	PS	6.0 MHz	Same as above

NOTES: C = Ceramic, D = Cerdip, P = Plastic; E = -40°C to +85°C, M = -55°C to +125°C, MB = -55°C to +125°C with MIL-STD-883 Class B processing, S = 0°C to +70°C.

DIVISION DE EDUCACION CONTINUA
FACULTAD DE INGENIERIA U.N.A.M.

MICROPROCESADORES Y MICROCOMPUTADORAS

EDITOR DE PANTALLA

OCTUBRE, 1984

Palacio de Minería Calle de Tacuba 5 primer piso Deleg. Cuauhtemoc 06000 México, D.F. Tel.: 521-40-20 Apdo. Postal M-2285

Introduccion

1. Modelo Conceptual

- Descripción -

Este editor le permite a usted pensar que lo que tiene en frente es un rollo de papel "inmenso",al que llamaremos documento, en el cual puede ir escribiendo lo que desee.

Las características más generales de el documento son las siguientes :

1. Cuando se inicia la sesión el documento esta en blanco y completamente enrollado.(Puede ver el letrero de fin, que ahí aparece. Su posición indica el tamaño actual del documento)

Ejemplo:

Inicie una sesión con el editor usando la instrucción :

edita <nombre del documento>

Presione varias veces la tecla que dice : return y observe como se desenrolla el documento.

Ahora presionne las teclas ctrl y Z al mismo tiempo y vea como al ir borrando líneas el documento se va enrollando de nuevo !

2. Todo lo que usted teclee quedará impreso en el documento con la característica única de que puede modificar en forma muy elegante y no tiene que preocuparse por manejar papel sino hasta el momento en que el documento este completamente terminado.

Ejemplo :

Escriba su poema, algoritmo, canción, pensamiento, etc . favorito presionando la tecla return cada vez que quiera pasar a una línea nueva.

Viaje a través de el documento usando las flechitas !

Búsque una línea que no le guste y reemplazela por otra , reescribiendo sobre ella !

Quisiera poner un comentario entre dos líneas seguidas ? Posicione el cursor en la línea superior y presione la tecla return. Se insertará una línea automáticamente entre las dos líneas !

2. Descripción del sistema

1. Esta descripción pretende darle las herramientas necesarias para que en un futuro sea usted capáz de modificar, para su mayor comodidad, el editor.
2. La representación interna del documento es la forma en que se almacena en memoria todo lo que usted teclea.

Esta representación tiene la siguiente estructura : El inicio de el documento lo da un separador al cual apunta siempre el apuntador cuyo nombre es "tope", de la misma manera el final de el documento esta dado por un separador al cual apunta el letrero de fin.

Las líneas se almacenan en forma compactificada por ejemplo, si usted inicia una sesión y teclea :

```
La Luna
La luz, a la que, lentamente, enamora mi espíritu
fin
```

En la memoria tendremos :

```
&La Luna&La luz, a la que, lentamente, enamora mi espíritu&
↑                                                         ↑
tope                                                     fin
```

3. Lo que nos permite generar esta representación interna, en forma sencilla es el uso de un renglón que está entre el teclado y dicha representación interna y es en el cual se va almacenando todo lo que se teclea, hasta que se presiona la tecla de return.

Por ejemplo, en el caso anterior cuando usted presiona la tecla de return después de teclear "La Luna" la representación interna tendrá la forma siguiente:

```
                    &La Luna&&
                     ↑      ↑
                   tope     fin
```

! Note que al presionar la tecla se inserta el texto más un separador que corresponde a el retorno de carro. !

y en el renglon lo que tendremos es el texto que se va tecleando :

```
 La luz, al que lentamen
 ↑                     ↑
renglon                mx
```

Note que el inicio de el renglón esta dado por el apuntador llamado renglon. Y que la posición, dentro de el renglón, en el que se escribirá el caracter siguiente esta apuntada por el apuntador cuyo nombre es mx .

En el momento que usted presione la tecla return el texto que este en el renglón, más un separador que se le agrega, pasará a la memoria y entonces la representación interna quedará :

```
&La Luna&La luz, a la que, lentamente, enamora mi espíritu&&
↑                                                        ↑
tope                                                    fin
```

Y en la pantalla lo que usted estará viendo será lo siguiente :

```
La Luna
La luz, a la que, lentamente, enamora mi espíritu
        ( Aquí realmente hay una línea en blanco )
fin
```

! Note como cada separador marca el inicio de una línea, excepto por supuesto el de fin !

Como es que se relaciona el contenido de el renglon con lo que tenemos en la representación interna ?, se preguntará, esto lo explico a continuación; suponga que tenemos el texto anterior

La Luna
La luz, a la que, lentamente, enamora mi espiritú

fin

Y que usted con las flechitas posiciona el cursor en el renglon que dice : La Luna

En el renglón tendremos :

```
 La Luna
 ↑
renglon
 ↑
 ax
```

! Con ax apuntado al inicio del renglon listo para modificar el texto !

Y la representación interna se verá de la manera siguiente :

```
&La Luna&La luz, a la que, lentamente, enamora mi espiritú&&
↑↑                                                        ↑
tope my                                                  fin
```

! Y aquí aparece el último apuntador que usamos en la representacion interna y cuyo nombre es my !

Este apuntador es importante ya que a la línea a la que apunta se coria en el renglón, y si se modifica el contenido de el renglón al teclear algo nuevo, my nos sirve como referencia para actualizar la representación interna.

Regresando a el ejemplo, si usted ahora baja con la flechita una línea y no modificó la anterior la representación interna se verá :

```
&La Luna&La luz, a la que, lentamente, enamora mi espíritu&&
 ↑       ↑                                                ↑
lope     my                                              fin
```

Y en el renglón tendremos :

```
  La luz, a la que, lentamente, enamora mi espíritu
   ↑
renglon
   ↑
   mx
```

! Note como my siempre apunta a el primer caracter de la línea !

Si en las condiciones actuales usted presiona la tecla de return entonces tendremos :

La representacion interna se verá :

```
&La Luna&La luz, a la que, lentamente, enamora mi espíritu&&&
 ↑                                                      ↑↑
lope                                                   my fin
```

Y en el renglón tendremos :

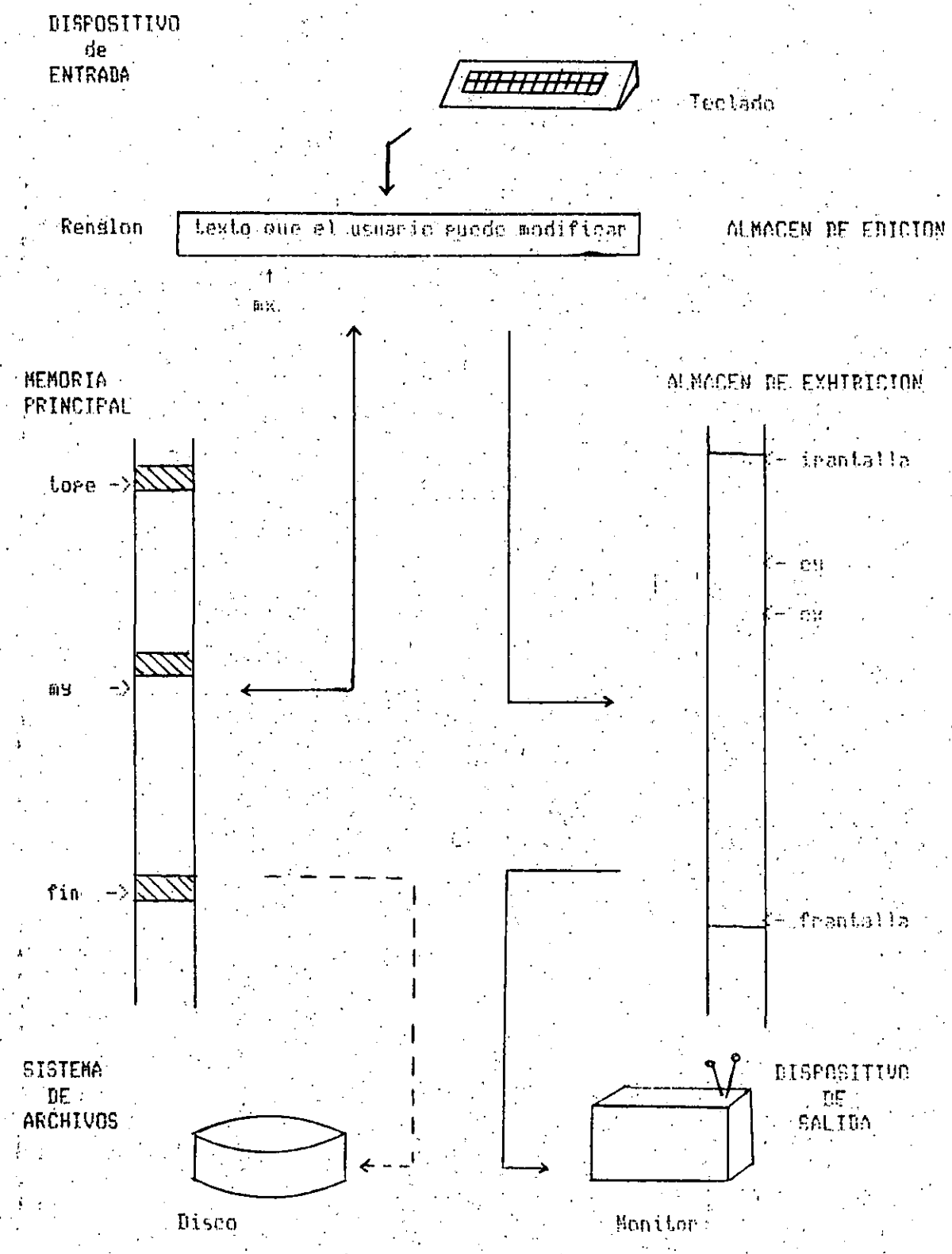
```
 ( línea en blanco )
  ↑
renglon
  ↑
  mx
```

! Note como en la representacion interna dos separadores, uno seguido de otro, representan una línea en blanco !

Y lo que usted estará viendo en la pantalla será lo siguiente :

```
La Luna
La luz, a la que, lentamente, enamora mi espíritu
( línea en blanco )   en esta línea deberá estar el cursor
( línea en blanco )
fin
```

ARQUITECTURA DE EL EDITOR



EDITOR DE PANTALLA

```
RUTINA editor de pantalla ( nombre del archivo )
    ESTABLECE medio ambiente del editor
    SI  existe el archivo
        ENTONCES
            TRAELO
        SINO
            CREALO
    FIN de la pregunta
    INICIALIZA
        memoria principal
            ilore
            imx
            ifin
        almacen de edicion
            ireg
            imx
        almacen de exhibicion
            ilfin
    MUESTRA la primera pagina del documento en la pantalla
    LLAMA al procesador de instrucciones
FIN dela rutina
```

IMPLEMENTACION EN FORTH

```
: edita
  editando
  trae/crea
  imem
  tope 14 refrescar
  leditor
;
```

PROCESADOR DE INSTRUCCIONES

```
RUTINA procesador de instrucciones
   INICIA un ciclo
      ACEPTA DATOS
      caracter <- tecla presionada por el usuario
      SI caracter = tecla de borrar
          ENTONCES
            borra el caracter
          SINO
            SI caracter = caracter de control
              ENTONCES
                BUSCALO
                SI busqueda = exitosa
                    ENTONCES
                      ejecutalo
                    SINO
                      continua
                FIN de la pregunta
              SINO
                documento <- caracter
                se modifico el documento
            FIN de la pregunta
      FIN de la pregunta
   CONTINUA siempre con el ciclo
FIN de la rutina
```

IMPLEMENTACION EN FORTH

```
:: leditor ** car idp & ( SI SE AGREGAN PARAMETROS MODIFICAR to P. F.)
  <do
   key .car
   car 7f = if
     rubout
   else
     car 20 <    % menor q' 20 -> caracter de ctrl
     if
       dp @ .idp
       2 c,
       c# 1 c,
       car          % lee el caracter de  control, hay que prenderle un bit
       40 or        % , el 6 , para convertirlo a caracter imprimible
       c,
       idp dp !
       context @ @  % vocabulario
       search
       not
       if
         execute
       fi
      else
        car texto
        sicambio
      fi
    fi
  od>
;
```

EDITAR

```
RUTINA documento ( caracter )
SI posicion del cursor = # de columnas
    ENTONCES
        LLAMA asigna caracter Z cursor > limite "> no eso
    SINO
        SI  modo insercion = activo
            ENTONCES
                LLAMA inserta caracter
            SINO
                MUESTRA el caracter en la pantalla
                LLAMA asigna caracter
        FIN de la pregunta
FIN de la pregunta
FIN de la rutina
```

IMPLEMENTACION EN FORTH

```
:: texto >> caracter  A
   cx #columnas 1- =
   if
     caracter mvasig
    else
      imodo
      if
       caracter insercion
      else
       caracter echo
       caracter mvasig
      fi
   fi
;
```

ASIGNA CARACTER

```
RUTINA asigna caracter (caracter)
ASIGNA a donde apunta mx el caracter
SI posicion = # de columnas
    ENTONCES
        LLAMA retorno de carro automatico
    SINO
        AVANZA apuntadores mx y cx
FIN de la pregunta
FIN de la rutina
```

IMPLEMENTACION EN FORTH

```
m:: mx mxasig >> val dir &          % asigna un caracter a el cursion en la
   val dir @ c!
    distancia $columnas >=
   if
      rcautomatico
    else
      mx 1+ .mx
   fi
;
```

INSERTA CARACTER

RUTINA inserta caracter (caracter)
    RECORRE los caracteres hacia la 'derecha' de mx
    LLAMA asigna caracter
    MUESTRA el contenido del almacen de edicion
FIN de la rutina.

IMPLEMENTACION EN FORTH

```
m:: renglon insercion >> caracter posicion &
      1 agrandarbuffer                  % agrandar el renglon
      caracter mxasig                   % insertar caracter en mx avanza mx 9 en
      sicambio
      }pantalla                         % enviar renglon a la pantalla
;
```

DIVISION DE EDUCACION CONTINUA
FACULTAD DE INGENIERIA U.N.A.M.

MICROPROCESADORES Y MICROCOMPUTADORAS

ANEXO 4
TERMINALES DE VIDEO E IMPRESORAS

ING. HÉCTOR CALVARIO MARTÍNEZ

OCTUBRE, 1984.

TERMINALES DE VIDEO

- TUBOS DE RAYOS CATODICOS + TECLADO
  (1840 POR WILLIAM CROOKOFT THOMPSONT FARADAY)

- TRABAJAN A MAYOR VELOCIDAD QUE LOS TTY

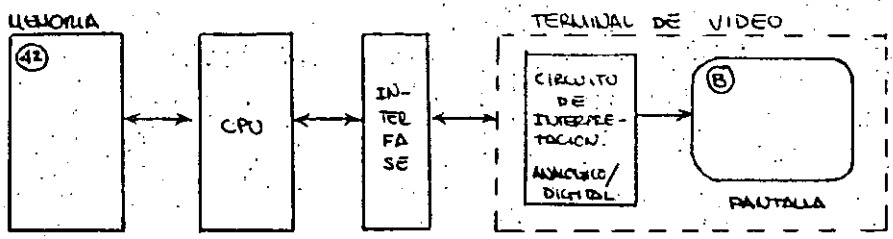
- APLICACIONES MAYORES: - DESPLIEGUE DE IMAGENES
  - GRAFICACION

- LA IMAGEN SE FORMA AL BARRER UNA PANTALLA FLUORESCENTE CON UN HAZ DE ELECTRONES PRODUCIDOS POR UN "CAÑON".

- DOS TIPOS DE TERMINALES DE VIDEO:

  - ALFANUMERICAS
  - GRAFICAS

- ALFANUMERICAS: EL HAZ RECORRE LA PANTALLA 60 VECES/SEG. → CONJUNTOS DE PUNTOS QUE REPRESENTAN UN CARACTER ASCII.



- GRAFICADORAS

CLASIFICACION DE LOS CRT's:

1) POSICION DIRECTA (VECTOR):

SE "DIBUJAN" LAS PARTES DE LA FIGURA EN CUALQUIER SECUENCIA POR MEDIO DE SEGMENTOS DE LINEA (VECTORES). PARA LOGRAR LA PERMANENCIA DE LA IMAGEN, SE DIBUJAN CONTINUAMENTE LOS SEGMENTOS QUE LA COMPONEN MEDIANTE UN PROGRAMA EN UN PROCESADOR ESPECIAL CONECTADO AL SISTEMA DE DESPLIEGUE.

2) DESPLIEGUE POR BARRIDO:

LA IMAGEN SE FORMA POR UNA SECUENCIA FIJA DE EXPLORACION (BARRIDO) DE UN HAZ DE ELECTRONES SOBRE LA PANTALLA DE DERECHA A IZQUIERDA Y DE ARRIBA HACIA ABAJO, VARIANDOSE LA INTENSIDAD DEL HAZ

- REPRESENTACION DE IMAGENES POR T.V.
- DESPLIEGUE DE DATOS PROVENIENTES POR COMPUTADORA.
  - TERMINALES DE CARACTERES PREFIJOS EN MEMORIA PROM.
  - TIPO BIT MAP (C/PUNTO DE LA PANTALLA (PIXEL) TIENE UNA CORRESPONDENCIA BIUNIVOCA CON UNA MEMORIA.

3) TUBOS DE ALMACENAMIENTO:

LAS FIGURAS SE FORMAN AL IGUAL QUE EN LOS DE POSICION DIRECTA, SOLO QUE EN ESTOS NO SE REQUIERE RETRAZAR LA IMAGEN.

4) DESPLIEGUE POR PLASMA:

LA IMAGEN SE FORMA POR LA IONIZACION DE UN GAS, EL CUAL PRODUCE UNA DESCARGA DE LUZ. LA PANTALLA ES UNA MATRIZ DE ELECTRODOS, Y EN CADA INTERSECCION HAY UNA CELDA DE DESPLIEGUE CON MEMORIA QUE CORRESPONDE A UN PUNTO.

APLICACIONES DE LOS SISTEMAS DE DESPLIEGUE

* TIPO POSICION DIRECTA Y TUBOS DE ALMACENAMIENTO.

  EN SISTEMAS DE GRAFICACION:

  - INFORMACION EN DISEÑO INDUSTRIAL
  - INDUSTRIA DE LA CONSTRUCCION
  - DISEÑO ELECTRONICO (CIRCUITOS INTEGRADOS)
  - TARJETAS DE CIRCUITOS IMPRESOS

  EN INSTRUMENTACION: - OSCILOSCOPIOS
  - ANALIZADORES DE ESPECTROS.

* TIPO DESPLIEGUE POR BARRIDO.

  EN SISTEMAS DE CARACTERES PREFIJOS (ALFANUMERICOS):

  - DESPLIEGUE DE CARACTERES A LA SALIDA DE UNA COMPUTADORA.

  EN GRAFICACION

  - REPRESENTACION DE IMAGENES REALES DE T.V.
  - MANEJO DE INFORMACION PARA DISEÑO INDUSTRIAL
  - INDUSTRIA DE LA CONSTRUCCION
  - DISEÑO ELECTRONICO

* TIPO DESPLIEGUE POR PLASMA

  - DESPLIEGUE DE CARACTERES A LA SALIDA DE UNA COMPUTADORA
  - TERMINALES PARA PROCESAMIENTO DE PALABRA

I M P R E S O R A S

(MPI - 88G)

IMPRESION EN SENTIDO UNI Y BIDIRECCIONAL

MODO GRAFICAS

"BUFFER" PARA ALMACENAR LOS CARACTERES Y COMANDOS (1 K BYTE)

CUANDO EL BUFFER CONTIENE UNA LINEA A SER IMPRESA Y UN COMANDO APROPIADO A INICIADO LA IMPRESION, EL BUFFER QUEDA EN UN ESTADO TAL QUE SE PUEDEN MANDAR CARACTERES A LA IMPRESORA MIENTRAS LA IMPRESORA IMPRIME

LA COMPUTADORA PUEDE MANDAR INFORMACION IMPRIMIBLE A LA IMPRESORA Y DESPUES PROCEDER A PREPARAR INFORMACION ADICIONAL MIENTRAS LA IMPRESORA VACIA SU BUFFER.

SECUENCIA DE OPERACION (MPI 88G)

I. A) BUFFER VACIO, CARACTERES HACIA LA IMPRESORA,
   B) LOS CARACTERES SON ALMACENADOS EN EL BUFFER.
   NO SE LLEVA A CABO LA IMPRESION HASTA QUE...

II. UN COMANDO QUE INICIE LA IMPRESION SEA RECIBIDA EN EL FLUJO DE DATOS (CR, LF, UNA LINEA LLENA DE CARACTERES, ETC.).
ESTE CARACTER SE EXAMINA Y TAMBIEN SE ALMACENA EN EL BUFFER. LA IMPRESION SE INICIA, REMOVIENDO UN CARACTER POR TIEMPO DEL BUFFER HASTA QUE EL CARACTER QUE FORZA LA IMPRESION SE ALCANZE.

III. SI EL BUFFER ESTA VACIO, LA IMPRESORA SE DETIENE.

IV. SI HAY CARACTERES COMANDOS, ESTOS SE EJECUTAN

V. SI HAY MENOS DE UNA LINEA COMPLETA EN EL BUFFER, TODOS LOS CARACTERES SON EXAMINADOS PARA VER SI HAY UN COMANDO (CR, LF, ETC.).
SI LO HAY, LA IMPRESORA INICIA A IMPRIMIR EN LA LINEA SIGUIENTE.
SI NO HAY TAL COMANDO LA IMPRESORA SE DETIENE HASTA QUE LLEGUE EL COMANDO.

VI. SI HAY MAS DE UNA LINEA EN EL BUFFER, EL MICRO DE LA IMPRESORA INSERTA AUTOMATICAMENTE CR.

VII. EL BUFFER OPERA EN MODO FIFO. CUANDO LOS CARACTERES SON REMOVIDOS DEL BUFFER, SU ESPACIO ES OCUPADO INMEDIATAMENTE POR NUEVOS CARACTERES.

TIPOS DE INTERFASE IMPRESORAS <—> FUENTE (COMP).

SERIAL: . COMUNICACION ASINCRONA { RS232C / CURRENT LOOP

. TRANSMISION DE DATOS EN DISTANCIAS RELATIVAMENTE LARGAS

. MINIMO DE ALAMBRES USADOS (UN PAR)

. TRANSMISION BIT A BIT A UNA CIERTA VELOCIDAD

. DATOS EN MENSAJES DE 10 ó 11 BITS

. PARA QUE LA IMPRESORA RECIBA LOS BITS APROPIADAMENTE TANTO LA COMPUTADORA COMO LA IMPRESORA DEBEN ESTAR AMBAS A UNA VELOCIDAD DE TRANSMISION ("BAUD RATE")

OPERACION RS232C (SERIAL)

NIVELES: "0" DE +3 A +25 V
"1" DE -3 A -25 V

SEÑAL	FUENTE	DESCRIPCION
(5) RECEIVED DATA	COMP.	DATO SERIAL DEL TRANSMISOR
(7) REQUEST TO SEND	IMPR.	LE INDICA A LA COMP. QUE LA IMPR. ESTA LISTA PARA RECIBIR DATOS.
(14) DATA TERMINAL READY	IMPR.	INDICA A LA COMP. QUE LA IMPR. ESTA ENCENDIDA.
(1) PROTECTIVE GROUND	IMPR.	TIERRA DEL CHASIS, NO CONECTADA CON LA SEÑAL DE TIERRA.
(13) SIGNAL GROUND	IMPR.	TIERRA DE SEÑAL Y LOGICA
(10) CURRENT LOOP POSITIVE	COMP.	ENTRADA POSITIVA PARA UN LOOP DE CORRIENTE DE 20 MA.
(6) CURRENT LOOP NEGATIVE	COMP.	ENTRADA NEGATIVA PARA UN LOOP DE CORRIENTE DE 20 MA.
(21) BUSY	IMPR.	NIVEL QUE INDICA QUE LA IMPRESORA NO PUEDE ACEPTAR DATOS

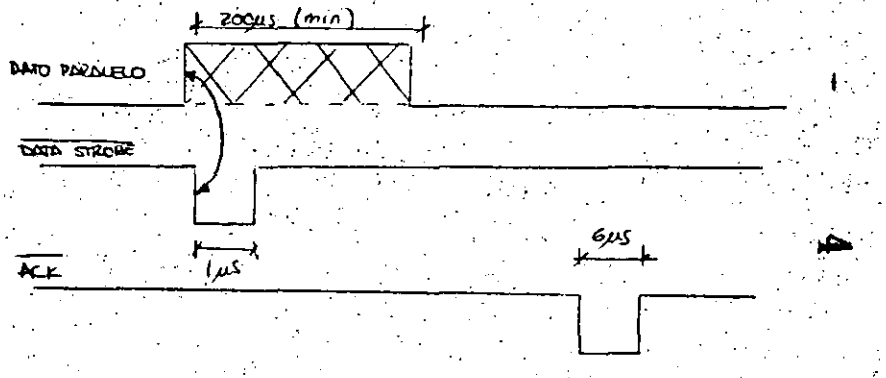
OPERACION POR LOOP DE CORRIENTE (SERIAL)

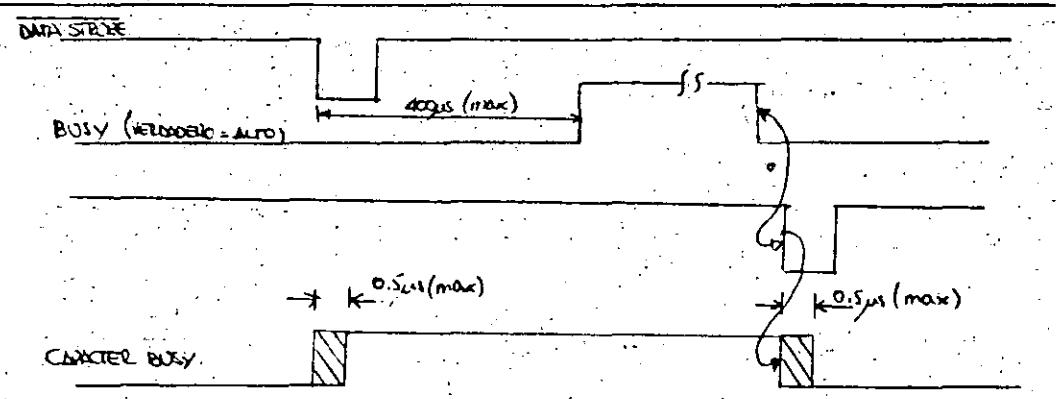
VARIACION DEL MODO DE TRANSMISION SERIAL. EN VEZ DE SER UN SISTEMA MANEJADO POR VOLTAJE ES MANEJADO POR CORRIENTE. SI LA CORRIENTE FLUYE A 20 MA ⇒"1" SI NO ENTONCES "0"

- DEBE EXISTIR EN LA IMPRESORA INTERFASES PARA LOOP DE CORRIENTE.

OPERACION EN PARALELO

- TRANSMISION BUFEREADA, PARALELA POR BIT, SERIAL POR CARACTER.
- SE USAN 3 LINEAS DE DATOS (EL BIT 8 SE IGNORA ELECTRICAMENTE).
- LOS DATOS SON "PUESTOS" (STROBE) POR LA COMPUTADORA Y SON RECIBIDOS Y "RECONOCIDOS" (ACKNOWLEDGED) POR LA IMPRESORA.
- SEÑAL DE BUSY INDICA EL STATUS DE LA IMPRESORA A LA COMPUTADORA. BUSY=1 CUANDO HAY UN CARACTER DE CONTROL DE MOV. DE PAPEL RECIBIDO EN LA IMPRESORA.





DIVISION DE EDUCACION CONTINUA
FACULTAD DE INGENIERIA U.N.A.M.

MICROPROCESADORES Y MICROCOMPUTADORAS

APENDICE CAPITULO 8-INTERPRETES

ING:DAVID BETANCOURT

O C T U B R E, 1984

5 - 10 KBYTES

- ES CONVERSACIONAL COMO LISP, BASIC O APL

- ES ESTRUCTURADO

- PERMITE COMPILACION INCREMENTAL

- ES EXTENSIBLE

- ESTA ESCRITO EN GRAN PARTE SOBRE SI MISMO Y POR LO TANTO ES MUY TRANSPORTABLE

- PUEDE INCLUIR UN SISTEMA DE MEMORIA VIRTUAL CONTROLABLE POR EL USUARIO

- ES BASTANTE RAPIDO

TRANSPORTABLE (NUCLEO = 2K)

EXTENSIBLE

CODIGO DE:
- DEFINICION DE TIPOS DE DATOS
- ACCESO A LOS DATOS

FUNCIONES DE CONTROL DE FLUJO

REDEFINIR PRIMITIVAS

```
A(5, 7, 6) = B(1, 4) + C(10, 30, 50)

1 4 B @ 10 30 50 C @ + 5 7 6 A !

TEMP = LEE1(1, 4) + LEE2(10, 30, 50)
CALL ESC(5, 7, 6,TEMP)
```

VOCABULARIO

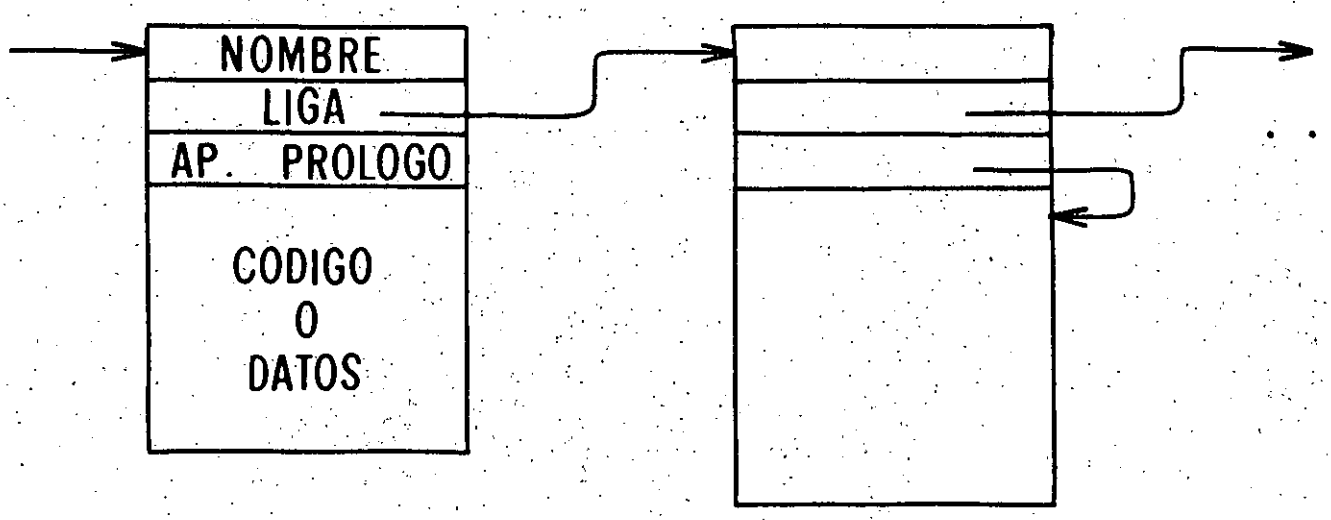
COLECCION DE FUNCIONES QUE PUEDE SEPARARSE Y RECIBIR UN NOMBRE

MEMORIA VIRTUAL

SCREENS

LENGUAJE INTERACTIVO:

DICCIONARIO



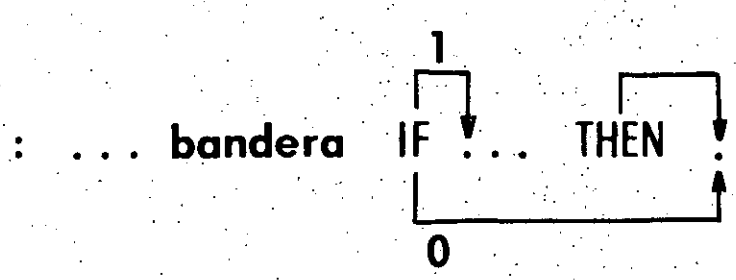
INTERPRETE EXTERNO (INTERFAZ CON EL USUARIO)

- ACEPTA CARACTERES DE LA TERMINAL
- SI RECIBE UN NUMERO LO METE AL STACK
- SI RECIBE UNA FUNCION, LA BUSCA Y LA EJECUTA

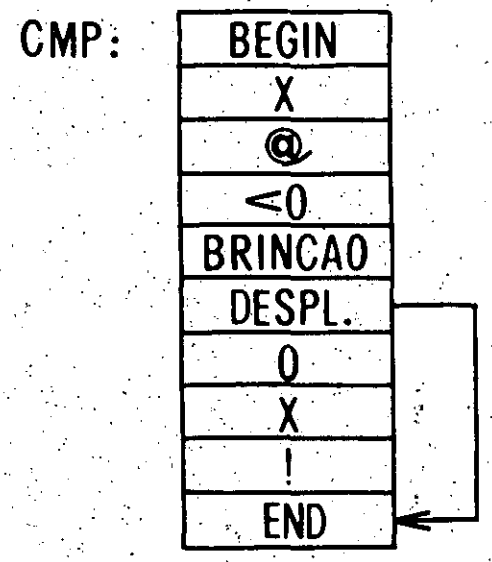
10 20 30 * + PRINT

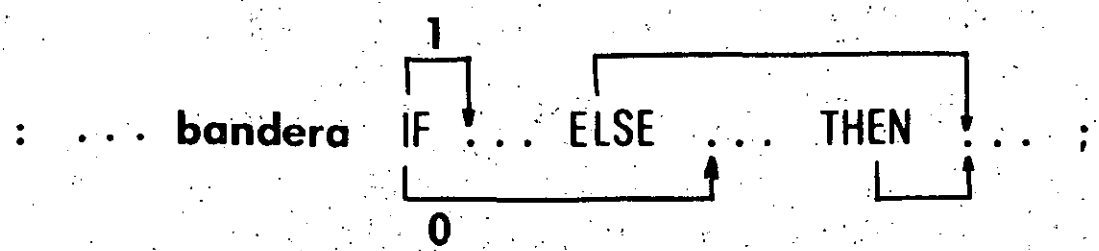
- UN SOLO DELIMITADOR: ESPACIO

- CODIGO ESTRUCTURADO




```
: CMP  X  @  <0  IF  0  !  THEN  ;
```





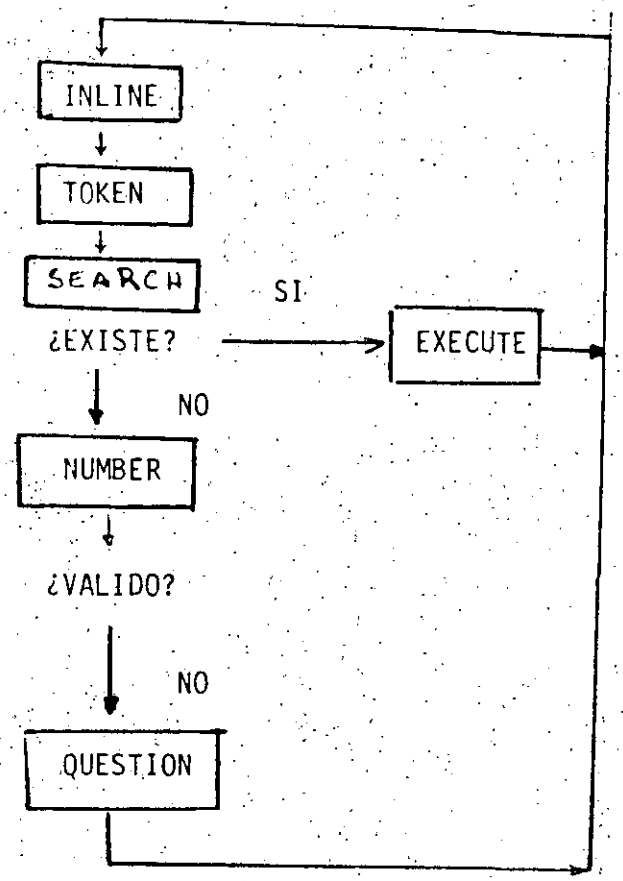
POCO LEGIBLE

PARAMETROS SIN NOMBRE

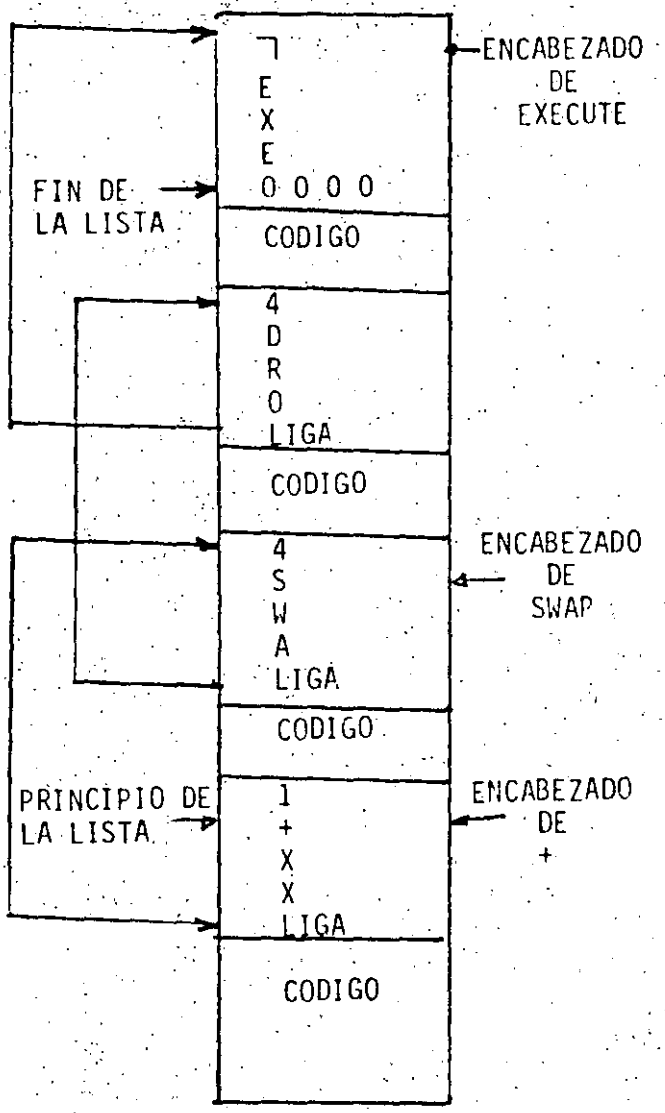
FALTA DE ANALISIS DE TIPOS

FALTA DE ANALISIS DE SINTAXIS

(FUNCIONES PEQUEÑAS)



INTERPRETE EXTERNO



ORGANIZACION DEL DICCIONARIO

PRODUCTO DE MATRICES

```
 0  VARIABLE  SUMA
10  MATRIZ  A
10  MATRIZ  B
10  MATRIZ  C
10    0   DO
   10    0  DO
       SUMA  0  SET
       10  0  DO
          SUMA  @
             K >  I > A
             I >  J > B

             +
             SUMA  !
         LOOP
     LOOP  →  K >  J >  C  SUMA @  SWAP  !
  LOOP
10  0  DO
10  0  DO
    J >  I >  C  C @
     LOOP
    LOOP
```

ORDENAMIENTO DE NUMEROS

```
10  DIM A (10)
**************************
20  FOR  I = 1  TO 10
30  INPUT  A-(I)
40  NEXT  I
**************************
50  F = 0
**************************
60  FOR  I = 1  TO 10
70  IF  A(I) < = A(I+1)  THEN  GOTO  40
80  T = A(I)
90  A(I) = A(I+1)
100  A-(I+1) = T
110  F = 1
120  NEXT I
130  IF F = 1  THEN GOTO  50
*********************************************
140  FOR I = 1  TO 10
150  PRINT  A (I)
160  NEXT I
```

10

PRODUCTO DE 2 MATRICES

```
 10  DIM A (10(10) B (10,10) ((10,10)
 20  FOR I = 1  TO 10
 30  FOR J - 1  TO 10
 40  INPUT A(I,J), B (I,J)
 50  NEXT J
 60  NEXT I
 70  FOR  I = 1  TO 10
 80  FOR  J = 1  TO 10
 90  SUMA = 0
100  FOR K = 1  TO 10
110  SUMA = SUMA + A(I,K) + B(K,J)
120  NEXT K
130  C (I,J) = SUMA
140  NEXT  J
150  NEXT I
160  FOR I = 1  TO 10
170  FOR J = 1  TO 10
180  PRINT C(I,J)
190  NEXT  J
200  NEXT  I
```

ORDENAMIENTO DE CADENAS

```
10  DIM A$ (10)
20  FOR I=1 TO 1 S
30  READ A$ (I)
40  NEXT I
50  F = 0  I = 1
60  IF A$ (I) <= A$ (I+1) THEN GOTO 110
70  T$ = A$ (I+1)
80  A$ (I+1) = A$(I)
90  A$ (I) = T($)
100 F = 1
110 F = I + 1
120 IF I <= 10 THEN GO TO 60
130 IF F = 1 THEN GOTO 50
140 FOR I = 1 TO 10
150 PRINT A$ (I)
160 NEXT (I)
170 DATA UNO DOS TRES CUATRO CINCO
180 DATA SEIS SIETE OCHO NUEVE DIEZ
```

SELECCION Y OPERACION DE BOMBAS DE AGUA Y SISTEMAS DE BOMBEO

TEMA : D E W A T E R I N G

PROF. ING. JUAN JACOBO SCHMITTER.

Septiembre, 1984.

Van Olphen, H.: 1956. "Forces between suspended bentonite particles," *Proceedings, 4th National Conference on Clays and Clay Minerals*, National Academy of Sciences, *National Research Council Publication* 456, p. 204.

Verwey, E. J. W., and J. T. Overbeek: 1948. *Theory of the Stability of Lyophobic Colloids*, Elsevier Press, Inc., Houston, Tex.

Wang, J. H.: 1951. "The effect of ions on the self-diffusion of water in aqueous electrolytic solutions," *Journal of Physical Chemistry*, vol. 58, p. 686.

Whitman, R. V.: 1957. "The behavior of soils under transient loadings," *Proceedings, 4th International Conference on Soil Mechanics and Foundation Engineering* (London), vol. 1, p. 207.

———: 1959. "Testing soils with transient loads," *American Society for Testing Materials, Special Technical Publication* STP 232, p. 242.

Wilson, S. D.: 1953. "Control of foundation settlements by preloading," *Journal, Boston Society of Civil Engineers*, vol. 40, no. 1, p. 10.

Winterkorn, H. W.: 1943. "The condition of water in porous systems," *Soil Science*, vol. 55, p. 109.

Woods, K. B.: 1940. "Design and construction of highway embankments," *Proceedings, Purdue Conference on Soil Mechanics and Its Applications*, p. 355.

Wu, T. H.: 1958. "Geotechnical properties of glacial lake clays," *Journal, Soil Mechanics and Foundations Division, American Society of Civil Engineers*, vol. 84, no. SM3, p. 1732.

Wylie, C. R.: 1951. *Advanced Engineering Mathematics*, McGraw-Hill Book Company, Inc., New York.

Wyllie, M. R. J., and A. R. Gregory: 1955. "Fluid flow through unconsolidated porous aggregates: Effect of porosity and particle shape on Kozeny-Carman constants," *Industrial Engineering Chemistry*, vol. 47, p. 1379.

——— and W. D. Rose: 1950. "Some theoretical considerations related to the quantitative evaluation of the physical characteristics of reservoir rock from electrical log data," *Journal Petroleum Technology*, vol. 2, p. 105.

——— and M. B. Spangler: 1952. "Applications of electrical resistivity measurements to problems of fluid flow in porous media," *Bulletin, American Association of Petroleum Geologists*, vol. 36, p. 359.

Yang, S. H.: 1953. "On the permeability of homogeneous anisotropic soils," *Proceedings, 2nd International Conference on Soil Mechanics and Foundation Engineering* (Rotterdam), vol. 2, p. 317.

Zeevaert, L.: 1957. "Consolidation of Mexico City volcanic clay," *American Society for Testing Materials, Special Technical Publication* 232, pp. 18–32, 381–410.

Chapter 3

DEWATERING

by Charles I. Mansur and Robert I. Kaufman

INTRODUCTION

Construction of buildings, powerhouses, dams, locks, tunnels, and graving docks frequently requires excavation below the water table into water-bearing soils. Such excavations require lowering the water table below the slopes and bottom of the excavation to prevent raveling or sloughing of the slope and to ensure dry, firm working conditions for construction operations. In some cases an excavation may be underlain by a pervious stratum under artesian pressure, which, if not relieved, can rupture the bottom of the excavation, with attendant development of sand boils and loss of material from the foundation beneath the structure.

Ground water can be controlled by means of one or more types of dewatering systems appropriate to the size and depth of the excavation, geological conditions, and characteristics of the soil. A properly designed, installed, and operated dewatering and pressure-relief system will facilitate construction by:

1. Lowering the water table and intercepting seepage, which would otherwise emerge from the slopes or bottom of the excavation
2. Increasing the stability of excavated slopes
3. Preventing loss of material from beneath the slopes or bottom of the excavation
4. Reducing lateral loads on sheeting and bracing
5. Reducing required air pressure in tunneling
6. Improving the excavating and hauling characteristics of sandy soils
7. Preventing rupture or heaving of the bottom of an excavation

Lowering the water table can also be utilized to increase the effective weight of the soil and thereby further consolidate both the soil above and below the lowered water table. Creating a partial vacuum in a sand stratum beneath a soft stratum of overlying soil will help consolidate the

soft material by increasing its effective weight as a result of lowering the water table and will also result in additional consolidation by adding a partial atmospheric pressure to the surface of the soil. Certain permanent structures can also be of less expensive design if the ground water or artesian pressure is permanently lowered or reduced. In many cases the permanent drainage or pressure-relief system can be utilized to dewater the excavation for the structure during construction.

Before the advent of suitable well-installation equipment, techniques, and pumping apparatus, the control of ground water in construction work was accomplished by one or a combination of the following methods:

1. Constructing ditches and dikes to intercept seepage from the slopes combined with a rate of excavation that permitted the slopes to more or less drain as the excavation was carried downward. Water was pumped from sumps in the collector ditches.

2. Sheeting combined with pumping from sumps or French drains in the bottom of the excavation.

3. Pumping from deep sheeted sumps dug outside of the working area.

Present installations for lowering the ground-water table for construction purposes now also include:

1. Wellpoints connected to header pipes installed at one or more elevations in the excavation pumped with combined vacuum and centrifugal pumps

2. Deep, large-diameter wells installed in or around the periphery of the excavation and pumped with deep-well turbine or submersible pumps

3. Vertical sand drains combined with deep wells and/or wellpoints installed in a deeper underlying pervious aquifer

4. Horizontal drainage system connected to a deep well or sump or to a gravity collection pipe

5. Vacuum dewatering for certain fine-grained or stratified silts and sands

6. Wellpoints combined with passage of a direct current through soil to the wellpoints (electroosmotic method)

The applicability of these methods of dewatering and ground-water lowering to various soil conditions, their design, installation, and operation are subsequently described.

3-1. Early Examples of Dewatering

The first recorded lowering of ground water to facilitate construction was that for the Kilsby tunnel for the London and Birmingham Railroad in 1838 (1). The water table for this tunnel was lowered by pumping from large vertical shafts sunk along and adjacent to the tunnel. A total flow of 1,800 gpm was pumped from these shafts. After this beginning, there was little advance in dewatering techniques for almost fifty years.

Use of wellpoints and large deep wells for lowering the water table was really first begun in 1896 in Germany in connection with construction of subways in Berlin (1). Cased deep wells, open at the bottom, were used in connection with the construction of the Esna barrage on the Nile River in 1907 (2). The first deep wells used in Europe for dewatering were large enough (3 or 4 ft) to accommodate a centrifugal pump inside the wells. Wellpoints were first used for dewatering in the United States in 1900. However, it was not until after 1921 that real improvements were made in wellpoint design, methods of installation, and in pumping equipment.

In Europe, the customary method for lowering the water table was to pump from deep wells spaced about 20 to 40 ft apart (3). The wells were generally about 8 in. in diameter; the screen was surrounded with a gravel filter; and the wells were pumped with a centrifugal pump connected to a common header pipe. Beginning in 1925 to 1930, extensive use was made of deep wells in Europe for lowering the water table and reducing artesian pressure for construction of large structures below the water table. Dewatering by pumping from deep wells was facilitated at this time by the development of vertical turbine and submersible pumps.

In the United States most dewatering has been accomplished by means of wellpoints. However, use has been made of deep wells with or without wellpoints to lower the water table or reduce artesian pressure since 1930.

Examples of some of the earliest large-scale dewatering projects in various countries are summarized in the following paragraphs.

Germany. One of the earliest examples (1) of the use of deep wells for dewatering was in connection with the construction of two large locks in the harbor at Bremerhaven in 1928 to 1931. Soil conditions consisted of 50 ft of soft clay underlain by 20 ft of sand under artesian pressure; the sand was underlain by clay. Construction of the locks required that the excavation be carried down to the top of the sand. Fifty-eight deep wells, located outside of the sheet piling to avoid interference with construction, were used to lower the artesian pressure in the sand 50 ft. Approximately 1,500 gpm was pumped from these wells.

Electroosmotic control of ground water was used for the first time in 1939 to stabilize a long railroad cut at Salzgitter, Germany (4). This cut was made in clayey silt where, because of the fine nature of the soil, a wellpoint system had proved ineffective. The cut was stabilized by installing 22-ft-long wellpoints at 30-ft intervals along the top of both sides of the excavation for cathodes, installing ½-in. gas-pipe anodes

midway between the cathodes, and then applying a potential of 90 volts of direct current between the cathodes and anodes. The wellpoints were surrounded with 3 in. of a graded sand-gravel. Water draining into the wellpoints was pumped out through 1-in. suction pipes. The power used amounted to 1.2 to 1.7 kw per cathode. Total consumption of energy per cubic yard of excavation amounted to about one kilowatthour. Before application of voltage to the soil there was practically no flow of water into the wellpoints; after application of voltage approximately 0.5 gpm was obtained from each wellpoint.

This method was also used to dewater and stabilize the slopes of an excavation 40 ft deep for a U-boat pen at Trondheim, Norway, during World War II (4). The soil at this site consisted of a soft, very silty clay with some sand seams. For this installation, the cathodes consisted of two rows of slotted 8-in. pipe, covered with copper mesh, 45 ft apart, with the wells spaced on 30-ft centers. Iron pipes were installed midway between the wells for anodes. A voltage of 40 volts was applied to the system.

United States. Deep wells were first used for dewatering in the United States for construction of the Rutgers Street tunnels in New York City in 1930–1931 (1). Sand existed from the surface to rock at a depth of 115 ft. The bottom of the shaft was 75 ft deep, and the water table was 50 ft deep. The dewatering was accomplished by pumping from five wells 100 ft deep equipped with electrically powered deep-well turbine pumps with a capacity of 500 gpm each. The wells had 10-in. screens 25 ft long surrounded with a gravel filter. Pumping these wells permitted construction of 1,350 ft of the landward section of the tunnels under free air, and it significantly lowered the air pressure required for the tunnels from the main shaft toward the river. Approximately 1,500 gpm was pumped from these wells.

Twelve hundred wellpoints were used to excavate, in the dry, 10 to 15 ft of silt and sand over a 400- by 1,500-ft area prior to placing fill in the Franklin Falls Dam in New Hampshire (1). The wellpoints lowered the water table 17 ft and permitted the use of wheeled excavating equipment. Approximately 14,000 gpm of water was pumped while dewatering this area.

At Tampa, Fla., wellpoints were used to dewater the excavations for three 83- by 1,200-ft shipbuilding basins and one 82- by 520-ft graving dock which penetrated marl overlying pervious limestone rock (5). Two levels or stages of wellpoints were installed. The lower stage was left in place to keep the water table down permanently, which permitted the bottom of the basins to be lined with concrete only 8 in. thick.

Belgium. One of the deepest, largest (30-ft diameter), subaqueous tunnels ever constructed in soft ground is the Antwerp vehicle tunnel driven under the Scheldt River in 1931 to 1933 (6). Wellpoints, deep wells, and freezing were used to cope with ground water. For the first time dewatering wells were sunk in a river to facilitate construction of a structure under the river without protection of cofferdams. The tunnel was 5,800 ft long and 116 ft deep at the deepest point. Soils at the site consisted of 25 ft of silty sand, 10 ft of clay (on the west side to the middle of the river), and 45 to 50 ft of fine sand underlain by clay. The objective of the dewatering system was to lower the water level in the sand in order to facilitate open-cut construction for the tunnel approaches and to reduce the air pressure required for the shield-driven section of the tube.

Wellpoints were installed on each side of the approach excavations. Deep wells (10-in.) with a gravel filter were installed on one side of the tunnel on 15- to 60-ft centers down to the top of the deep clay. More deep wells were installed in the river in two rows on each side of the terminal. Submersible pumps with a capacity of 75 gpm were installed in each well. Pumping these wells lowered the water table 45 ft. Lowering the water table, combined with steel sheet piling driven into clay, permitted construction of 850 to 900 ft of tunnel on each end in open cut.

Lowering the water table 50 ft permitted construction of a portion of the cast-iron section of the tunnel under free air and materially reduced the air pressure required for the remainder of the tube.

The ground-water problem at the ventilation shafts was solved by freezing the ground around each shaft by circulating refrigerated brine through 116 wells around the shaft. The need for bracing was also eliminated by the natural stability of the hollow cylinder of frozen ground 12 to 15 ft thick.

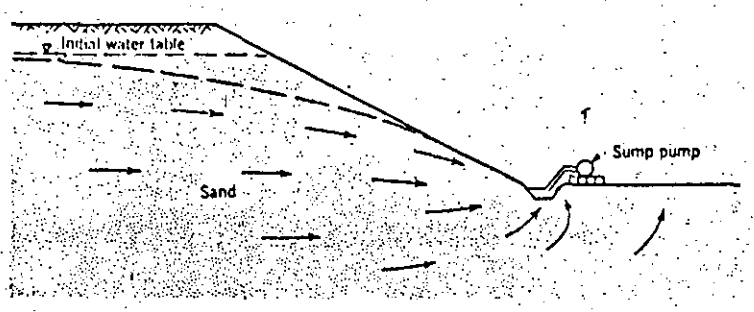
England. The first example of predrainage on an extensive scale by means of deep wells in England was in 1931 to 1933 in connection with construction of the King George V graving dock (1). The dock was 75 ft deep, 135 ft wide, and 1,200 ft long. The soil consisted of clay underlain by fine sand at a depth of 100 ft, the top of which sloped downward toward the outshore end. For two-thirds of the lock, there was danger that the artesian pressure would burst through the overlying clay stratum in the bottom of the excavation. Ten wells 175 ft deep and spaced 200 ft apart and provided with submersible pumps were used to lower the artesian pressure beneath the excavation to safe values. The screens for these wells were surrounded with two concentric rings of gravel filter. The wells were pumped at a rate of 750 gpm. The wells were subsequently connected to horizontal discharge pipes leading into the lock to provide permanent pressure relief beneath the lock.

METHODS OF DEWATERING AND PRESSURE RELIEF

3-2. Sumps and Ditches

For small excavations and in some types of soils (e.g., dense, well-graded, or cemented soils) it is sometimes possible to permit seepage from the slopes to collect in ditches and sumps, from which it can be pumped out of the excavation as illustrated in Fig. 3-1.

Fig. 3-1. Collection of seepage in open ditches and sumps.



Collection of seepage in open ditches and pumping from open sumps without filters has several disadvantages. It tends to cause softening and raveling or sloughing of the lower part of a slope. Where the soil contains lenses of fine sand or silt, springs frequently develop which can cause underground erosion and subsidence of the adjacent ground surface or slumping of the slope. The rate of excavation may also be slowed as a result of having to wait for the slopes and soil to drain. Where the slopes are not too steep or the seepage too heavy, the slopes and bottom may be stabilized by covering with a well-graded sand and gravel.

3-3. Sheeting and Open Pumping

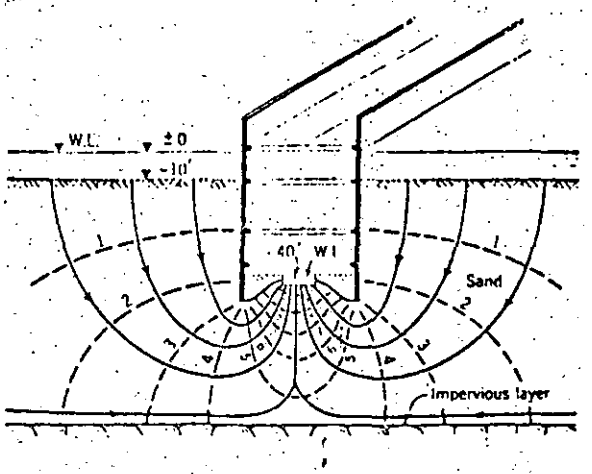
Prior to development of dewatering equipment, most excavations in pervious soils below the natural water table were made by driving wood or steel sheet piling, excavating the earth, and pumping the water out as it seeped into the bottom of the sheeted excavation. In this method of excavation the seepage is forced to enter the enclosed area through the bottom as shown in Fig. 3-2.

As a result of the water table being lowered much more rapidly on the inside than on the outside, the water and seepage will create considerable pressure on the sheeting and at the bottom of the excavation. If the

head is great enough, seepage forces at the bottom can cause the soil to become quick, lose its shearing strength, and place excessive load on the bottom struts, with the possibility of the bracing collapsing. For certain conditions, piping may undermine the sheeting, thereby causing loss of the excavation. Another disadvantage of this method is that the dirt will be wet and difficult to handle.

Sheeting and open pumping can be successfully used to control ground water if the sheeting is designed and braced to take into consideration resulting seepage forces and loss of soil strength. Covering the bottom of the excavation with a properly graded sand-and-gravel filter blanket will facilitate construction and pumping out the seepage water. Appropriate seepage analyses should be made before attempting to use this method of controlling ground water.

Fig. 3-2. Unwatering a sheeted excavation by sump pumping. (*From Tschebotarioff (7).*)



3-4. Deep-well Sumps

Sheeted sumps were one of the earliest methods used to lower the water table. However, such sumps are inefficient, expensive to dig, and work satisfactorily only in relatively coarse materials. Wells with screens, with or without a gravel filter, are now used in lieu of deep sumps.

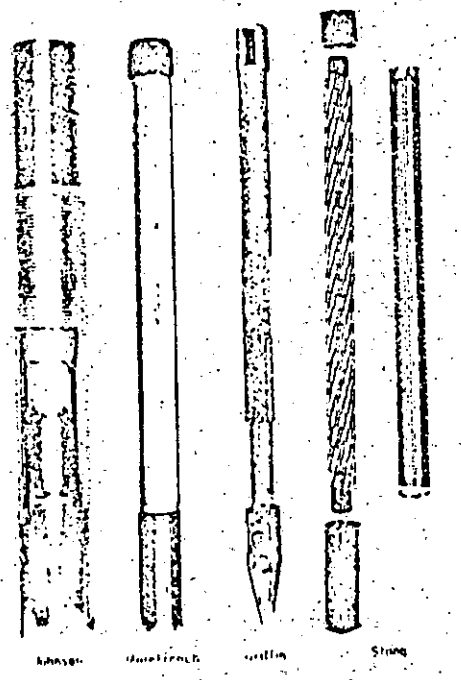
3-5. Wellpoint Systems

Wellpoints are small well screens approximately 2 to 3 in. in diameter and 1 to 3½ ft long. They are manufactured with either brass or stain-

less-steel screens and are made with either closed ends or self-jetting tips (Fig. 3-3).

Lines or rings of these wellpoints installed on 3- to 12-ft centers along or around an excavation and attached to a common header pipe (6 to 12 in. in diameter) and connected to a wellpoint pump (a combined

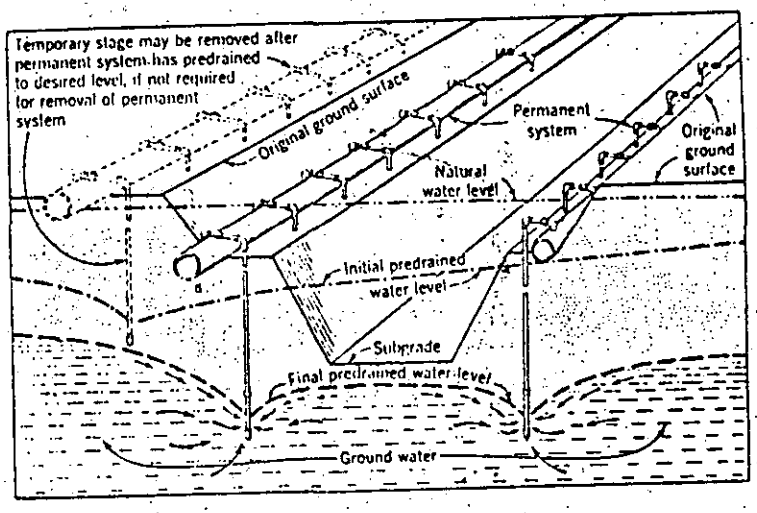
Fig. 3-3. Self-jetting wellpoints.



vacuum and centrifugal pump) are called a wellpoint system. A typical installation of a wellpoint system is shown in Fig. 3-4.

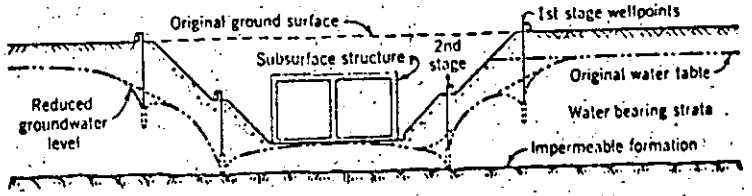
Wellpoints are the most common method for dewatering or lowering the water table for construction purposes in the United States. A wellpoint system is suitable where the site is accessible and where the water-bearing strata to be drained are not too deep. Such a system is usually the most practicable and economical method for dewatering small excavations. Wellpoints are also generally used where the water table does not have to be lowered too much and for open-cut work in water-

Fig. 3-4. Typical installation of a wellpoint system. [*Griffin Wellpoint Corp.* (8).]



bearing soil. For large excavations or where the depth of excavation below the water table is more than 30 or 40 ft, or where artesian pressure in a deep aquifer beneath an excavation must be reduced, it may be more desirable to use deep wells and turbine pumps with or without wellpoints as accessory. Wellpoints are more suitable than deep wells where the submergence available for the well screens is small (Fig. 3-5). Where

Fig. 3-5. Use of wellpoints where submergence is small. [*From Neboline* (1).]

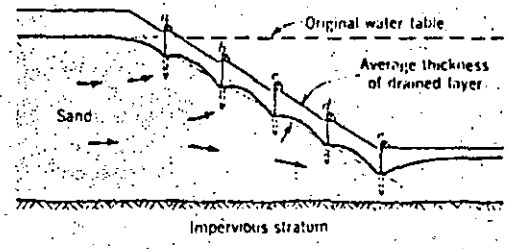


adequate submergence is available and the required rate of pumping is large, deep wells and pumps may be indicated in lieu of wellpoints.

Wellpoints may be used to dewater deep open-cut excavations by installing a row of wellpoints for about every 15 ft of depth as illustrated in Fig. 3-6.

The average thickness of the outer portion of the slope drained by this method is not more than about 15 ft. Beneath this drained layer, the soil is acted on by the seepage pressure of percolating water. If the depth of the cut is more than 40 to 50 ft, the stability of the slope should be checked, taking into account the seepage forces beneath the drained zone.

FIG. 3-6. Drainage of an open deep cut by means of a multiple-stage wellpoint system. [*From Terzaghi and Peck* (3).]



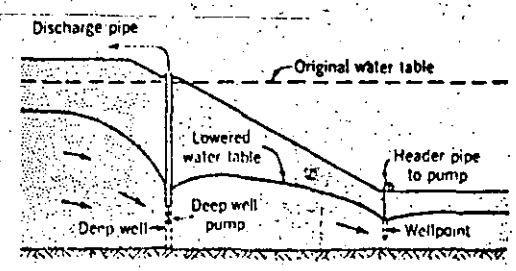
Where the required depth of water-table lowering is more than 15 ft and the rate of pumpage for each wellpoint is relatively small (i.e., less than 10 to 15 gpm), it may be advantageous to install a single-stage system of wellpoints at the top of the excavation or water table and pump each wellpoint with a jet-eductor pump, rather than install a multiple-stage system of wellpoints. As a jet-eductor pump is capable of producing 10 ft or more of vacuum and lifting water over 100 ft, a jet-eductor wellpoint system is capable of lowering the water table 50 to 100 ft below the ground surface.

A jet-eductor wellpoint system consists of a wellpoint attached to the bottom of a jet-eductor pump, with one pressure pipe and a slightly larger return pipe. (The size of these pipes usually varies from 1 to 2 in., depending on the head and capacity requirements of the pump.) The wellpoint, jet-eductor pump, and two pipes are installed in a cased hole and may be surrounded with a filter sand if soil conditions require. Jet-eductor pumps may be powered with individual small high-pressure centrifugal pumps or with one or two large pumps pumping into a single pressure pipe supplying water to each wellpoint with a single return header pipe. With the single-pump setup, the water is usually recirculated through a stilling tank with an overflow for the gain in water from pumping the wellpoints. This type of system has the disadvantage that jet-eductor pumps have an efficiency of only about 30 per cent, and their design is quite complex.

3-6. Deep-well Drainage

Large-diameter deep wells are also suitable for lowering the groundwater table where the soil formation becomes more pervious with depth, the excavation penetrates or is underlain by sand or coarser granular soils, and where there is sufficient depth of pervious materials below the level to which the water table is to be lowered for adequate submergence of well screen and pump. In contrast to wellpoints, a dewatering system consisting of deep wells and submersible or turbine pumps can be installed outside of the zone of construction operations, and drainage effected to the depth of excavation required. Where applicable, deep wells may be used exclusively to lower the water table for an excavation or in combination with a wellpoint system as illustrated in Fig. 3-7. Some wellpoints may be needed at the toe of the slopes to intercept any minor

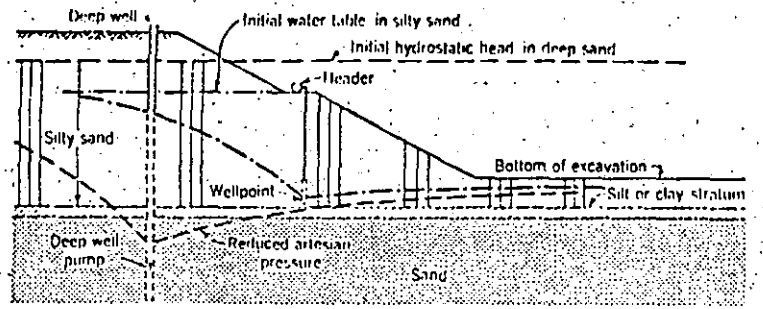
FIG. 3-7. Dewatering a deep excavation by means of deep wells and wellpoints. [*From Terzaghi and Peck* (3).]



seepage that may pass between the wells. By pumping from deep wells installed around the top of an excavation, seepage will be intercepted before it can compromise the stability of the slope.

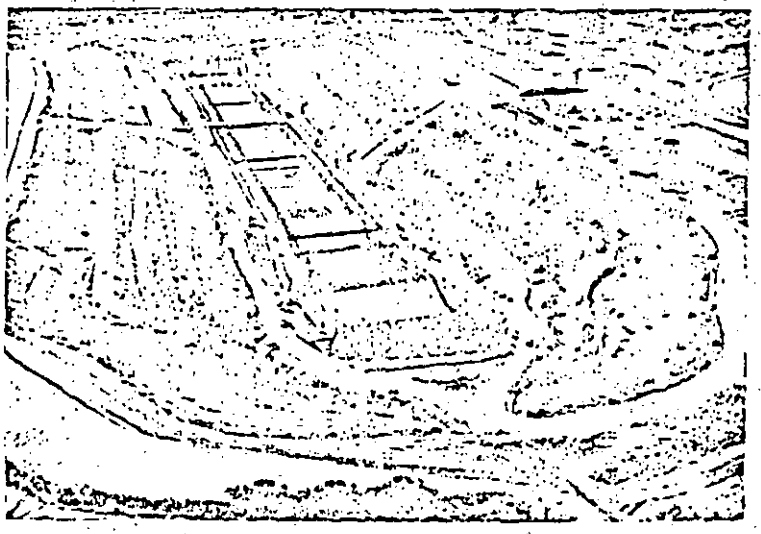
Where the bottom of a large excavation is underlain by a relatively impervious stratum, which in turn is underlain by a pervious stratum under excessive artesian pressure, a heave or blow may occur in spite of drainage achieved by shallow wellpoints on the slopes. If the net hydrostatic pressure on the bottom of the impervious stratum is greater than the *effective* weight of the overlying soil, the bottom of the excavation will heave at a local weak or thin area, with a resultant sand boil. Such an occurrence may result in costly delays or may render the foundation unsuitable for the intended structure. Relief of this artesian pressure can best be achieved by means of deep wells installed at the top of the excavation or at some safe elevation as the excavation is carried down, as illustrated in Fig. 3-8.

FIG. 3-8. Relief of artesian pressure by means of deep wells.



Deep wells for dewatering are usually spaced on 20- to 200-ft centers, depending upon the amount the water table must be lowered, perviousness of the sand, source of seepage, and amount of submergence available. Such wells commonly have a diameter of 6 to 18 in. with a screen 20 to 75 ft long. The screen may consist of a commercial type of water well screen or a perforated metal or wooden screen surrounded with a properly graded sand-gravel filter. A combination system of wellpoints and deep wells for dewatering slopes and relieving artesian pressure beneath a large excavation is shown in Fig. 3-9.

FIG. 3-9. System of deep wells for relieving artesian pressure beneath excavation for Port Allen Lock, La. (*Photograph by U.S. Army Engineer District, New Orleans*, 1958.)



Where the area to be dewatered consists of silts or silty sands underlain by a more pervious stratum, seepage toward the excavation can be intercepted and the water table lowered by a combination of vertical sand drains installed around the top of the excavation and deep wells with screens installed in the deep sand. The sand drains will permit drainage of the upper, less permeable soil down to the deeper sand in which the pressure is kept reduced by pumping from the deep wells.

3-7. Horizontal Drainage

Where it is desirable to avoid open-cut work and submergence is inadequate for deep wells, the ground-water table can be lowered by means of a Ranney drainage system. A Ranney system consists of a number of horizontal perforated pipes projected from one or more reinforced-concrete shafts or wells. These pipes may be extended 200 ft or more in any direction. Ground water flowing into the well is usually pumped out by means of a turbine pump. This type of system is not considered suitable for lowering the water table in stratified soils.

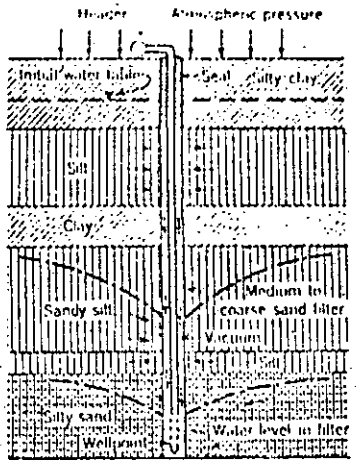
3-8. Vacuum Dewatering Systems

Fine-grained silts ($D_{10} \leqq 0.05$ mm) with a low coefficient of permeability ($k = 0.1 \times$ to 10×10^{-4} cm/sec) cannot be drained successfully by gravity methods because the water is held in the voids of the soil by capillary forces. However, such soils can be stabilized by means of a *vacuum* well or wellpoint system. A vacuum dewatering system consists of wells or wellpoints with the screen and riser pipe surrounded with a free-draining sand filter extending to within a few feet of the surface (Fig. 3-10). The remainder of the hole at the top is sealed with bentonite or impervious soil. By maintaining a vacuum in the well screen and sand filter, the hydraulic gradient producing flow toward the well or wellpoint is increased, particularly in stratified soils, and the soil in the vicinity of the wells or wellpoints is stabilized by atmospheric pressure, which tends to prevent seepage from entering the excavation and increases the effective pressure on the soil grains and thus the shear strength of the soil. In order to dewater this type of soil properly it is usually necessary to install the wells or wellpoints fairly close together.

In the case of a wellpoint system, the net vacuum at the wellpoint and in the filter is the vacuum in the header pipe minus the lift or length of riser pipe. Therefore relatively little vacuum effect can be obtained with a wellpoint system if the lift is more than 15 ft. If there is much air loss, it may be necessary to attach extra vacuum pumps to the wellpoint pump in order to ensure maximum vacuum in the wellpoints. The required pumping capacity is, of course, small.

Dewatering, and the effect of vacuum and atmospheric pressure, can

Fig. 3-10. Vacuum system of drainage.



Note: Soil in vicinity of wellpoint under partial vacuum.

be achieved at considerable depths by using deep wells with long screens and filter except for a section of riser pipe at the top, submersible pumps, and a vacuum line connected to the tops of the wells. The tops of the wells must be sealed airtight as well as the space around the riser pipes.

Grain-size curves of soils for which various types of drainage systems are considered most suitable are shown in Fig. 3-11.

3-9. Drainage by Electroosmosis

Most soils that require dewatering can be dewatered by one or a combination of the methods previously described. However, there are some silts, clayey silts, and fine clayey silty sands that cannot be successfully drained by the previous methods but that can be drained by wells or wellpoints combined with a flow of electricity through the soil to the wells. This method of drainage is known as the electroosmotic or electrical drainage method. The application of electroosmosis to dewatering of soil was largely developed by Dr. Leo Casagrande (4).

If two electrodes are driven into saturated soil and a direct electric current is passed between them, water contained in the soil will migrate through the soil from the positive electrode (anode) to the negative electrode (cathode). By making the cathode a well, the water can be removed by pumping. In this manner, water in the soil which otherwise would tend to seep toward the excavated slope and reduce the stability of the soil mass flows instead toward the wells, thereby increasing the shear strength of the soil and stability of the slope.

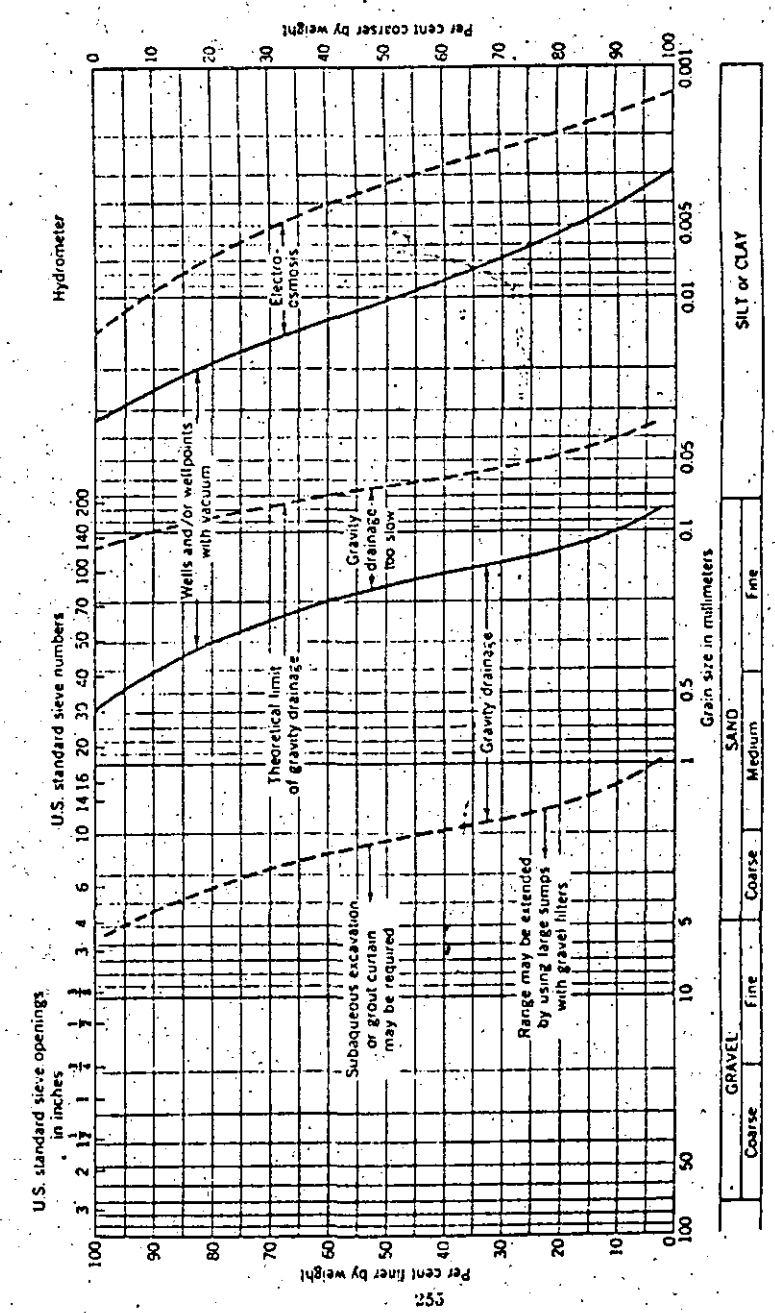
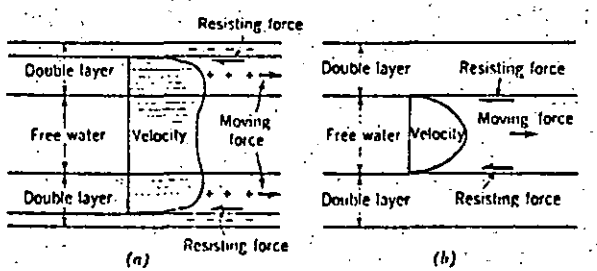


Fig. 3-11. Dewatering systems applicable to different soils. (*Moreltench Corp.*)

An electric current causes water to flow through soil in the following manner. Since the surface of soil particles carries a net negative charge, positive ions (cations) in solution are attracted to the soil particles and concentrate near their surfaces. As the centers of gravity of the positive and negative charges in a water molecule do not coincide, water molecules are attracted to the cations (Chap. 2). Upon application of a d-c voltage between two electrodes, the positive ions adjacent to the soil particles and the water molecules attached to the ions are attracted to the cathode (or well) and are repelled by the anode. The free water in the interior of the void spaces is carried along to the cathode by viscous flow. A comparison of electroosmotic flow with hydraulic flow through a single capillary is illustrated in Fig. 3-12. The actual distribution and average velocity in the voids of soil would vary from those shown because of the variation in the size of voids in soil.

FIG. 3-12. Comparison of electroosmotic flow with hydraulic flow in a single capillary (*a*) Electroosmotic flow; (*b*) hydraulic flow.



According to Casagrande (4), the coefficient of electroosmotic permeability k_e, or the rate of flow for electroosmotic flow, is about the same for either sands, silts, or clay. Casagrande reports that for practical purposes most soils may be assumed to have a k_e of 0.5×10^{-4} cm per sec for a gradient of 1 volt per cm.

3-10. Miscellaneous Methods of Ground-water Control

Where open gravel seams exist, it may be desirable to supplement the dewatering system by first grouting a curtain wall around the area to reduce well and pumpage requirements. A suitable grout mixture consists of bentonite, portland cement, an admixture to reduce surface tension, and water. Silica gels and a commercial product known as AM-9 have also been used for grouting to control ground water. AM-9 has a viscosity approaching that of water and will successfully penetrate fine sand. For grouting to be effective, the voids in the ground must be large enough to take the grout, the grout holes must be close enough

together so that a more or less continuous grout curtain is obtained, and the depth of grouting should penetrate fully the strata being grouted.

Ground water has also been controlled by freezing a zone of soil around the area to be excavated. However, this is a rather expensive procedure and requires expert design and installation. If the ice cofferdam is not completely effective, seepage of water through even a small gap in the frozen barrier after the excavation is opened up may cause rapid enlargement of the fault, with consequent serious trouble, difficult if not almost impossible to remedy.

For small excavations it may be desirable to drive steel sheet piling around the work area, excavate the soil under water, and then tremie in a seal of concrete. This method is frequently used for bridge piers in open water. In restricted areas it may be necessary to use a combination of sheeting and bracing with wells or wellpoints installed just inside or outside of the sheeting.

FIELD INVESTIGATIONS AND SOIL CONDITIONS

In determining the need for and before selecting and designing a dewatering system, a number of factors should be considered or investigated. These include the geological and soil conditions in the immediate vicinity of the site; size and depth of the excavation; time water table is to be lowered; permeability, stratification, and thickness of the pervious strata to be dewatered; water table and hydrostatic pressure beneath the excavation; variation of water table and hydrostatic pressure with season of year or river stage; river stages likely to occur; source of seepage or radius of influence likely to exist for geological and soil conditions at the site; allowable ground-water table or uplift pressure during construction; protection from flooding; effect of ground-water lowering on adjacent structures or wells; power facilities; and chemical characteristics and temperature of the ground water and its variation with the season of year. All these factors should be considered in designing a dewatering system; however, the detail of investigation of the individual items will depend upon the project and the complexity of the dewatering problem.

3-11. Geological and Soil Conditions

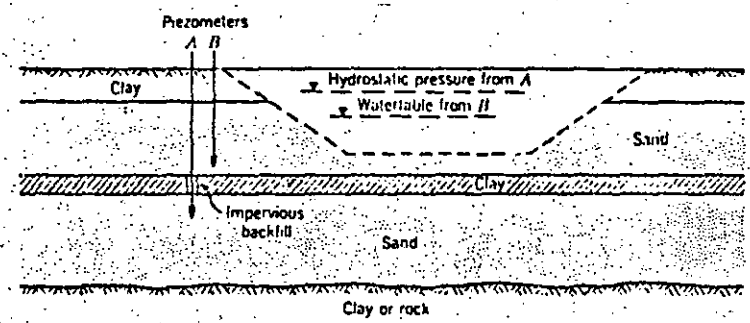
A thorough knowledge of the characteristics of the soils adjacent to and beneath an excavation is of paramount importance in the design and installation of a dewatering system. The type and stratification of the foundation soils should be ascertained from properly made borings. For deep or large excavations several of these borings should extend to the bottom of the excavation plus 1½ times the depth of the excavation. The borings should be spaced sufficiently close together to reveal any significant variation in soil conditions that would have a bearing on the

need for dewatering, seepage flow, or spacing of wells or wellpoints. Samples should be taken at sufficiently frequent intervals to detect changes in soil type. For large excavations and excavations underlain by deep strata of sand, the depth of the sand and its permeability should be ascertained for its full depth. The method of dewatering generally most suitable for various types of soils is shown in Fig. 3-11.

3-12. Water Table and Artesian Pressure

A thorough knowledge of the water table and any substratum pressure at a site is most important in planning an excavation and any dewatering or pressure-relief system. The initial elevation of the water table will generally determine the elevation at which the first stage of wellpoints or wells would have to be installed. Therefore the water table should be determined by borings, or preferably by piezometers, with tips installed in the strata to be dewatered as illustrated in Fig. 3-13. If time permits,

FIG. 3-13. Installation of piezometers to determine water table and artesian hydrostatic pressure.



the water table or artesian pressure should be observed over a period of time since it will frequently vary with the season of the year or with the stage of an adjacent river. The water table or hydrostatic pressure should be correlated with the stage of any adjacent stream. A rapid rise in water table with a rising-river stage indicates that the river may serve as a ready source of seepage. The maximum stage of any nearby river likely to occur during construction should also be ascertained.

For most soils the ground-water table during construction should be maintained at least 2 to 5 ft below the bottom of the excavation in order to ensure "dry" working conditions. It needs to be kept somewhat lower for silts than sands to keep traffic from pumping water to the surface and making the bottom of the excavation wet or spongy.

Where an excavation is underlain by a stratum of silt or clay that is underlain by a pervious stratum of sand under artesian pressure, upward seepage from the deeper stratum may keep the bottom of a large excavation wet, even though wellpoints may be in operation on the slopes or at the toe of the excavation. If this situation exists, it may be necessary to lower the head in the deep sand stratum below the bottom of the excavation by means of deep wells or wellpoints. If the intervening clay stratum, as shown in Fig. 3-13, is tight, the hydrostatic head in the deep sand can be somewhat higher than the bottom of the excavation, but in no event should the net head above the bottom of the excavation exceed 80 per cent of the submerged weight of the soil above the top of the artesian aquifer, to prevent any heave or blowup in the bottom of the excavation.

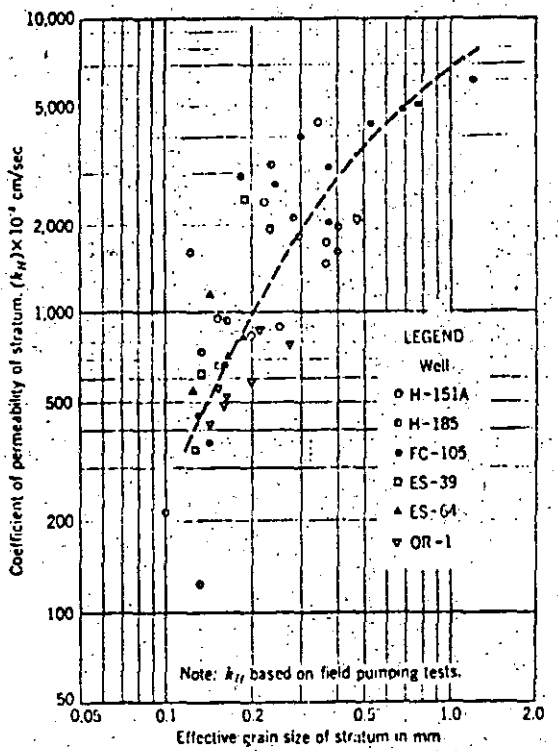
3-13. Coefficient of Permeability of Pervious Strata

The permeability of the sand strata to be dewatered or in which the hydrostatic pressure is to be reduced should be determined prior to designing a dewatering system. Various methods may be used to determine this permeability. The simplest is a visual examination and classification of the sands by someone familiar with soils and by comparison with sand of known permeability. A rough approximation of the permeability may be obtained from the following table.

Type of sand (U.S. Army Engineer classification)	*Coefficient of permeability* $k \times 10^{-4}$ *cm/sec*
Very Fine sand	50
Fine sand	200
Fine to Medium sand	500
Medium sand	1,000
Medium to Coarse sand	1,500
Gravel and Coarse sand	3,000

A better estimate of the permeability can be obtained from a comparison of grain size curves or the D_{10} of the sand samples, with certain empirical relations between D_{10} and k such as shown in Fig. 3-14, as developed from field tests on sands in the middle and lower Mississippi River Valley (9 11). However, these empirical relationships may be misleading unless they are based on reliable information and are known to be valid in the area of the project. Laboratory permeability tests may be used to estimate the permeability of uniform sands, but such tests frequently do not give results indicative of the actual *in situ* permeability of many

FIG. 3-14. D_{10} versus *in situ* coefficient of permeability—Mississippi River Valley. (*Reference* 9.)



sands, particularly well-graded or stratified sands or gravelly sands (9-11). Also, laboratory tests on sand samples which have become segregated or contaminated with drilling mud during sampling operations do not give reliable results.

For large dewatering projects a pumping test on a well that fully penetrates the sand stratum to be dewatered is considered warranted to determine the permeability of the pervious formation. If desired, the permeability of individual strata may be determined by means of a well flow meter such as described in Ref. 11. The drawdown to the test well should be determined by means of piezometers installed with the tips in the sand formation being tested and on one or more lines extending out from the test well a minimum of 500 ft. The coefficient of permeability can be computed from Eq. (3-49) or (3-59). Readings from such piezometers will also give an indication of the effective distance to the source of seepage or *radius of influence* for the soil and well being tested. The water level in the well should not be used to compute the permeability of the sand without accurate evaluation of well entrance losses. It is advisable to pump the well at three different rates of flow for check purposes and to observe any change in the radius of influence with drawdown. A record should be kept of drawdown with time of pumping, and the well should be pumped long enough to develop the full drawdown and radius of influence.

3-14. Source of Seepage

The distance to the "effective" source of seepage has an appreciable effect on the design and operation of a dewatering or pressure-relief system in that it affects both the spacing of wells or wellpoints and flow to the system. The source of seepage may be the bed of a nearby stream or other body of water that may be in contact with the pervious strata to be dewatered; the flow may be from the aquifer being drained, the distance to which is commonly known as the *radius of influence;* or seepage may flow to the area being dewatered from both an adjacent river and from the aquifer itself.

The source of seepage depends to a great extent on the geological features of the area, adjacent streams or bodies of water, on the perviousness of the sand formation, and amount of drawdown. Where the only source of seepage is from the formation being dewatered, the distance, or radius of influence, can be estimated from pumping tests where the drawdown curve is determined by means of piezometers, or from empirical relationships between the radius of influence and permeability of sand developed from field observations such as shown in Fig. 3-39. Generally, the normal radius of influence is greater for coarse or very pervious sands than for finer sands, whether the flow be gravity or artesian. The radius of influence will also become greater with increased drawdown in the area which is being dewatered and with pumping time. The magnitude of these effects is also dependent upon whether the flow is gravity or artesian flow. However, these effects are difficult to estimate numerically, and therefore the radius of influence should be estimated conservatively.

3-15. Miscellaneous Factors

Some ground waters are very corrosive, and others contain iron or various carbonates that form incrustations which will, over an extended period of time, clog or partially clog well screens, gravel filters, or surrounding sand and reduce the efficiency of a well system. Therefore consideration should be given to the chemical properties of ground water in designing and operating a dewatering system.

Corrosion of well screens usually consists of one of the following types: (1) direct chemical, (2) dezincification or selective, or (3) electrolytic (galvanic). The first type can be attributed to destruction or solution of the metal by chemicals in the water. Dezincification consists essentially of one metal of an alloy being removed, leaving the metal in a spongy and weakened condition. It results from the electrochemical difference in potential between the metals in an alloy when submerged in a saline or slightly acid water with the presence of oxygen. Galvanic corrosion occurs when two dissimilar metals are electrically connected in a solution which will conduct electricity.

Incrustation of well screens, gravel filters, and sand around wells results from precipitation of materials carried to the wells in solution such as carbonates of calcium and magnesium, oxidation of iron compounds in solution upon contact with air to form an "ocherous" material, and deposition of soil carried to the wells, and in some cases stoppage results from the presence of iron bacteria or slime-forming organisms.

Chemical characteristics of ground water pertinent to corrosion and incrustation tendencies are:

pH	Hardness	Hydrogen sulfide
Carbon dioxide	Chlorides	Carbonates
Alkalinity	Iron	

A pH below 7.0 shows the water to be acid and that the water will tend to be corrosive. Free carbon dioxide is the chief cause of acidity and low pH values in ground water. Alkalinities in ground water exist in three forms: bicarbonates, carbonates, and hydroxides. They are a measure of potential incrustants in a water. Hardness is due mainly to the presence of calcium and magnesium compounds, usually in the form of bicarbonates, sulfates, and chlorides. Where these mineral salts are dissolved in the ground water, portions of these substances will come out of solution when the pressure in the water is reduced. For example, if the drawdown at a well produced by pumping releases enough carbon dioxide to upset the solution balance, solid particles of carbonates and sulfates will incrust the screen, gravel pack, or sand around the well. Generally, chlorides in ground water in quantities less than 100 ppm are not a problem in dewatering; however, they may be a problem in greater concentrations. Saline waters promote galvanic action in metallic screens and risers of different metals. Iron or iron compounds (usually iron bicarbonate or sulfate) in ground water may precipitate in the gravel pack, well screen, or in the well or pump, particularly where the water has access to air or is subjected to violent turbulent action such as in a pump.

The most common causes of corrosion are:

1. Low pH coupled with low alkalinity, low hardness, and high content of carbon dioxide
2. High dissolved oxygen content
3. Presence of hydrogen sulfide, sulfur dioxide, or similar gases
4. Presence of organic acids or iron sulfate

Corrosion may be minimized by using similar metals in well or wellpoint construction, metals that are resistant to corrosion, materials such as wood or plastics that are not subject to corrosion, or steel coated with asphalt. Metals and alloys graded in order of ability to resist attack of ground water are as follows:

1. Monel
2. Bronze
3. Stainless steel
4. Brass
5. Galvanized iron

The primary cause of incrustation of wells is the presence of incrustants in the ground water and the release of carbon dioxide as a result of reduced pressure at the well caused by pumping and/or the oxidation of iron in the water as a result of the ground water coming in contact with oxygen in the air at the well or a change in solution balance as the water goes through the pump. Where incrustation tends to form, it cannot be entirely prevented. It can be minimized by:

1. Installing wellpoints or wells so that the water can enter the well with the least resistance possible for the flow to be handled
2. Designing the wells so that entrance velocities are not excessive
3. Not drawing the water down at any one well more than necessary
4. Cleaning the well, if necessary, before the incrustation becomes excessive

The temperature of ground water, particularly near streams, may vary with the season of the year, with a resulting effect on required rate of pumpage. A change in the temperature of the ground water of 1°F will change the pumpage rate about 1½ per cent. Thus, for large dewatering jobs, it may be advisable to consider this factor in design and/or operation of a dewatering system.

Where the excavation is near a stream or open water, a check should be made of maximum river or tide stages likely to occur while dewatering so that measures can be taken to protect the equipment against flooding.

Where the area to be dewatered is adjacent to structures founded on or underlain by saturated loose sand, care should be exercised to prevent any detrimental settlement of such structures as a result of lowering the water table. Therefore, prior to the start of any large-scale dewatering operation, the condition, elevation, and water table of structures immedi-

50

ately adjacent to the project should be ascertained. If the excavation and dewatering results in excessive lowering of the water table at such structures, it may be necessary to underpin the structure or install recharge wells into which water is pumped to maintain the original water-table elevation, to prevent detrimental settlement of the structure or structures.

ANALYSES AND DESIGN FORMULAS

Design of a dewatering and/or pressure-relief system requires determination of the number, size, spacing, and penetration of the wellpoints or wells and the rate at which water must be removed from the pervious strata in order to achieve the required ground-water lowering or pressure relief. The size and capacity of the collectors and pumps also will depend upon the required rate of discharge and corresponding drawdown in the wells or wellpoints. Therefore it is necessary to establish the fundamental relationships between discharge from wells and wellpoints and the corresponding drawdown produced in the pervious strata which they penetrate. It is not intended to present detailed derivations of all formulas used in design; however, some of the formulas most commonly used are developed to permit understanding the procedures used in the derivations and the limitations in the formulas.

The rate of discharge necessary to produce the required ground-water lowering is computed from equations which relate the hydraulic head loss in and flow of water through porous soil strata. The basic laws governing the flow of water through porous soils have been presented in Chap. 2; in general, the flow of ground water is laminar and in conformance with Darcy's law. The subsequent design equations pertaining to well flow and drawdown caused by pumping from wells are based on the assumption that Darcy's law is valid and that the flow is continuous. It is further assumed that the pervious strata being unwatered and/or pressure-relieved are either homogeneous and isotropic or are anisotropic but have been transformed into equivalent isotropic strata. Procedures for accomplishing this transformation are given in Chap. 2.

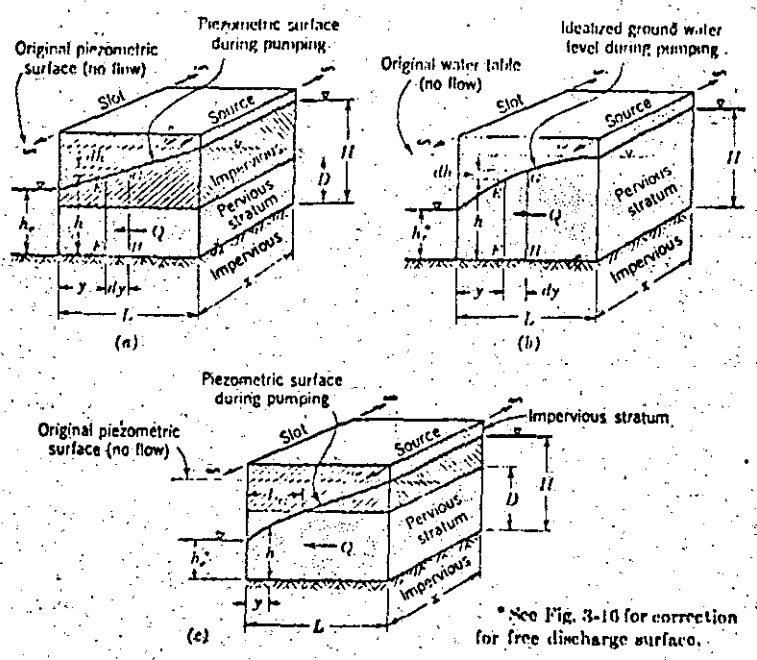
Usually, wells are classed either as *artesian* or *gravity* wells, depending on conditions of flow in the pervious strata in which the well screen is set. Sometimes wells are of the combined *artesian-gravity* type. A description of flow conditions for artesian, gravity, and combined artesian-gravity wells follows, together with appropriate equations for discharge from and drawdown caused by a single well and by multiple wells. To facilitate comprehension of the principles involved in developing equations for well flow and drawdown, it is desirable to consider first the principles of flow to a vertical slot and of flow nets, since both are pertinent to the design of dewatering systems.

3-16. Flow to a Slot from a Single Line Source

Fully Penetrating Slot. Where a dewatering system consists of a single line of closely spaced wells, an approximate solution for the drawdown produced by these wells can be obtained by considering the line of wells equivalent to a drainage slot. The validity of this assumption depends upon the spacing of the wells; as wells are more closely spaced, they tend to approach a continuous line sink or slot. Inasmuch as many cases arise where a system of wells can be approximated by a slot and as the equations for flow to and head reduction caused by a slot are of use in evaluating flow to wells in complex systems, the equations for flow to slots are presented below. Although the subsequent equations are based on the assumption that the slot is infinite in length, i.e., the flow per unit length is constant, in actuality a slot or line of wellpoints or wells will have a finite length. Where a slot of finite length is a distance L from a line source of seepage, the flow to the slot will be the same as if the slot were of infinite length except within a distance of about $0.5L$ from each end of the slot. Near the ends of the slot the flow will be greater and the head reduction less than it would be if the slot were of infinite length. Conditions near the ends of the slot can be evaluated from a plan flow net as described on pages 276ff.

For *artesian* flow, consider a pervious, homogeneous, isotropic stratum of constant thickness D, bounded above and below by horizontal impervious strata as shown in Fig. 3-15*a*. Assume that the seepage enters the stratum from a vertical line source and emerges from the stratum at a vertical line sink or slot that fully penetrates the pervious stratum, and assume that both the source and slot are of infinite length. Further, consider that water is pumped from the slot continuously but that during pumping the water level in the slot is at or above the top of the pervious stratum. Under these conditions the flow is "confined," or "artesian," because the head h at every point in the pervious stratum will be at an elevation equal to or above the top of this stratum. Also assume that no hydraulic head loss occurs at the slot as the ground water flows from the soil into the slot. In nature these conditions would be approximated where a line of very closely spaced wells (the slot) is installed near and parallel to the bank of a river in which the pervious substratum is exposed. The relationship between the head h and rate of discharge Q from the slot per unit length x of the system, after equilibrium is reached, can be obtained by considering the state of flow in the vertical element *EFGH* shown in Fig. 3-15*a*. The flow Q through the element can be expressed

FIG. 3-13 Flow from a line source of seepage to a fully penetrating slot or sink, both of infinite length. (*a*) Artesian flow; (*b*) gravity flow; (*c*) combined artesian-gravity flow.



* See Fig. 3-10 for correction for free discharge surface.

as follows:

$$Q = kiA \tag{3-1}$$

where Q = flow through area A per unit time
k = permeability of pervious stratum in direction of flow
i = hydraulic gradient producing flow

From Fig. 3-15*a* it is apparent that

$$i = \frac{dh}{dy} \quad \text{and} \quad A = Dx \tag{3-2}$$

Substituting values given by Eq. (3-2) into Eq. (3-1) results in the following expression:

$$Q = kDx\frac{dh}{dy} \quad \text{or} \quad dh = \frac{Q\,dy}{kDx} \tag{3-3}$$

which when integrated becomes

$$h = \frac{Qy}{kDx} + c \tag{3-4}$$

The value of c is obtained by inserting into Eq. (3-4) the following boundary conditions: $h = h_s$ where $y = 0$. Therefore $c = h_s$, and Eq. (3-4) becomes

$$h = \frac{Qy}{kDx} + h_s \tag{3-5}$$

The expression for flow Q is obtained by substituting into Eq. (3-5) the boundary conditions at the source of seepage, $h = H$ at $y = L$. Thus Eq. (3-5) becomes

$$Q = \frac{kDx}{L}(H - h_s) \tag{3-6}$$

The drawdown $(H - h)$ at any distance y from the slot is obtained by combining Eqs. (3-5) and (3-6), from which it is found that

$$H - h = \frac{Q}{kDx}(L - y) = \frac{L - y}{L}(H - h_s) \tag{3-7}$$

For *gravity*, or "unconfined," flow, consider a vertical slot of infinite length that fully penetrates a homogeneous, isotropic, pervious stratum containing a horizontal *free* water surface and bounded at its base by a horizontal impervious stratum. Further assume that the pervious stratum is supplied by a vertical line source also of infinite length as shown in Fig. 3-15*b*. For a condition of equilibrium the equation for flow Q per unit time per unit length x of the system can be developed in a manner similar to that for an artesian case. In the development of this equation, it is assumed that on any vertical line below the drawdown curve or free water surface, the hydraulic gradient is constant and equal to the slope of the drawdown curve at the point where the vertical line intersects the drawdown curve. This last assumption commonly is referred to as the *Dupuit-Forchheimer assumption* (12, 13). Considering the flow through the vertical element *EFGH* shown in Fig. 3-15*b* and applying Darcy's law,

$$Q = kiA \tag{3-1}$$

and considering that

$$i = \frac{dh}{dy} \quad \text{and} \quad A = hx \tag{3-8}$$

substituting Eq. (3-8) into Eq. (3-1) gives

$$Q = khx\frac{dh}{dy} \quad \text{or} \quad h(dh) = \frac{Q}{kx}dy \tag{3-9}$$

Integrating Eq. (3-9) and substituting the boundary condition that at $y = 0$, $h = h_o$ and at $y = l_e$, $h = H$, the following expressions are obtained:

$$h^2 = \frac{2Qy}{kx} + h_o^2 \tag{3-10}$$

and

$$\boxed{Q = \frac{kx}{2l_e}(H^2 - h_o^2)} \tag{3-11}$$

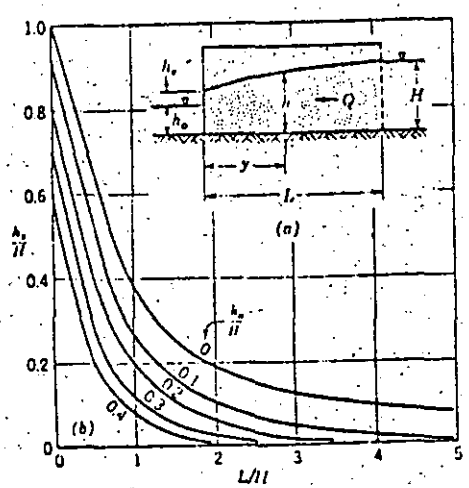
Combining Eqs. (3-10) and (3-11) results in the following equations for head h and the term $H^2 - h^2$:

$$h^2 = \frac{y}{l_e}(H^2 - h_o^2) + h_o^2 \tag{3-12}$$

$$H^2 - h^2 = \frac{2Q}{kx}(l_e - y) = \frac{l_e - y}{l_e}(H^2 - h_o^2) \tag{3-13}$$

The drawdown curve will be at an elevation higher than that corresponding to values of h computed from either Eq. (3-10) or (3-12), as indicated in Fig. 3-16*a* because of vertical drainage at the slot. The

FIG. 3-16. Gravity flow to a fully penetrating slot from a line source of seepage and height of free discharge surface. (*a*) Section through flow system; (*b*) correction factor for height of free discharge surface.



height h_o of this free discharge surface and corresponding drawdown curve can be estimated from Fig. 3-16*b* developed by Chapman (14). Except where l_e/H and/or h_o/H are small, the shape of the drawdown curve can be obtained satisfactorily from Eqs. (3-10) and (3-12). Where such is not the case, the drawdown curve should be determined from the head h as computed from the following expression:

$$H^2 - h^2 = \frac{l_e - y}{l_e}[H^2 - (h_o + h_o)^2] \tag{3-14}$$

From a comparison between Eqs. (3-13) and (3-14) it is apparent that where h_o is very small compared with h_o, the equations are essentially identical. Regardless of the height of free discharge surface h_o, the discharge Q can be computed from Eq. (3-11), using the height of water h_o at the slot as the value of h_o.

In the development of equations for artesian flow to a slot, it was assumed that the rate of pumping was such that the water level in the slot was not lowered below the top of the pervious stratum. However, in practice, it may be necessary to dewater an excavation extending through a relatively impervious top stratum and into a pervious substratum. Therefore it is necessary to consider the effect of lowering the ground-water level below the top of such a pervious substratum. This case is referred to as the *artesian-gravity* case and is illustrated in Fig. 3-15*c*. From this figure it is apparent that near the line source the flow is artesian, whereas gravity flow occurs in the vicinity of the slot. The distance L_a from the slot to the point at which the flow changes from artesian to gravity (the point at which the piezometric surface coincides with the top of the pervious stratum) can be determined as follows: The artesian flow Q_1 can be obtained from Eq. (3-6) by substituting D for h_o and $l_e - L_a$ for L. Similarly, the gravity flow Q_2 can be obtained from Eq. (3-11) by substituting D for H and L_a for L. Accordingly, the following equations for flow are obtained:

$$Q_1 = kDx\frac{H - D}{l_e - L_a} \tag{3-15}$$

and

$$Q_2 = \frac{kx}{2L_a}(D^2 - h_o^2) \tag{3-16}$$

As Q_1 must equal Q_2,

$$L_a = \frac{l_e(D^2 - h_o^2)}{2DH - D^2 - h_o^2} \tag{3-17}$$

Substituting L_a into Eq. (3-16),

$$Q = \frac{kx(2DH - D^2 - h_o^2)}{2L} \tag{3-18}$$

53

The piezometric surface can be obtained from the following two expressions for head which can be derived from Eqs. (3-7) and (3-12):

For $y \leqq L_a$: $$h = \sqrt{\frac{y}{L}(D^2 - h_s^2) + h_s^2} \tag{3-19}$$

For $y \geqq L_a$: $$h = \left(\frac{H - D}{L - L_a}\right)(y - L_a) + D \tag{3-20}$$

It is apparent from the above that the problem of artesian-gravity flow is solved by first determining the point at which the flow changes from the artesian to the gravity state and then considering the zone adjacent to the slot as one of gravity flow. Since Eq. (3-19) does not consider the influence of the height of free discharge at the slot on the resulting drawdown curve, the following expression should be used to compute the head h where y is less than L_a:

$$h = \sqrt{D^2 - \left(\frac{L_a - y}{L_a}\right)[D^2 - (h_s + h_s)^2]} \tag{3-21}$$

where the value of h_s can be obtained from Fig. 3-16*b*, using L_a for L, and D for H.

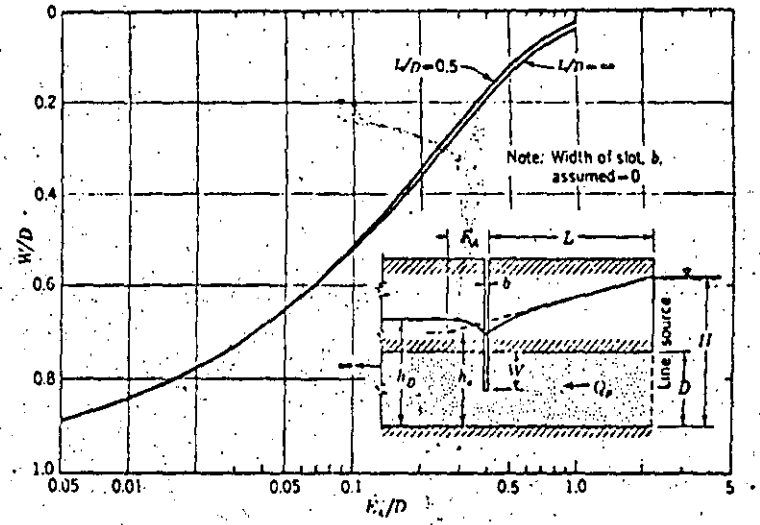
Partially Penetrating Slot. The previous equations for flow to a slot were based on the assumption that the slot fully penetrated the pervious stratum being dewatered. Frequently, the thickness of the pervious stratum is too great to permit the economic use of a fully penetrating dewatering system, and therefore it is necessary to consider the influence of a partially penetrating system on the reduction of hydrostatic pressures in the pervious stratum. Only the artesian- and gravity-flow cases are presented, since relationships for the combined artesian-gravity-flow case can be obtained from the gravity-flow case after first determining the point at which the flow changes from artesian to gravity. This point can be determined by equating the gravity flow to the flow in the artesian zone and then solving for the distance L_a, using procedures similar to those in the preceding paragraph.

For a slot that partially penetrates an *artesian* stratum supplied by a line source as illustrated in Fig. 3-17, the flow Q_p from the slot can be computed from the following expression:

$$Q_p = \frac{kD_x(H - h_s)}{L + E_A} \tag{3-22}$$

where E_A is an "extra-length" factor which depends upon the ratio of slot penetration W to the thickness of the pervious substratum D, and is determined from Fig. 3-17 developed by Barron (15).

FIG. 3-17. Artesian flow to a partially penetrating slot from a line source of seepage.



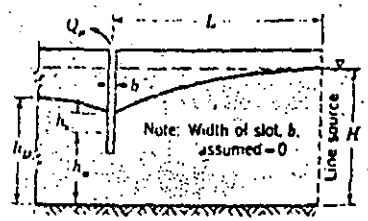
As illustrated in Fig. 3-17, the maximum residual head h_D at a distance downstream from the slot is greater than that at the slot and can be computed from the following equation:

$$h_D = \frac{E_A(H - h_s)}{L + E_A} + h_s \tag{3-23}$$

Thus when wellpoints are spaced sufficiently close as to be considered equivalent to a slot, the head downstream from the slot should be considered since it will exceed that at the slot.

The flow from, and drawdown caused by, pumping a slot which partially penetrates a pervious stratum supplied by a line source under *gravity*-flow conditions, as shown in Fig. 3-18, can be computed from

FIG. 3-18. Gravity flow to a partially penetrating slot from a line source of seepage.



results of model studies conducted by Chapman (16). The flow from the slot Q_s can be computed from

$$Q_s = \left(0.73 + 0.27\frac{H - h_s}{H}\right)\frac{kx}{2L}(H^2 - h_s^2) \qquad (3\text{-}24)$$

with the symbols as shown in Fig. 3-18. The maximum residual head downstream from the slot h_D can be obtained from the following equation:

$$h_D = h_s\left[\frac{1.48}{L}(H - h_s) + 1\right] \qquad (3\text{-}25)$$

As in the case for artesian flow, the head h_D exceeds that at the slot and therefore is of practical significance. Equations (3-24) and (3-25) are valid for values of L/H equal to or greater than 3, which will encompass the range of most field problems.

3-17. Flow to a Slot from Two Line Sources

Fully Penetrating Slot. Generally, the flow to a slot of infinite length will originate on both sides of the slot, instead of only one side. Where the slot is located midway between and parallel to two parallel line sources, the flow for a given head reduction will be twice that computed from Eq. (3-6), (3-11), or (3-18), where the value of L used in these equations corresponds to the distance from the slot to either of the two line sources.

Partially Penetrating Slot. Where a partially penetrating slot of great length installed in a pervious substratum is pumped, the flow to such a slot will be symmetrical with respect to the slot and can be considered as originating from two line sources equidistant from and parallel to the slot. This case is frequently encountered and is discussed below for both artesian- and gravity-flow systems.

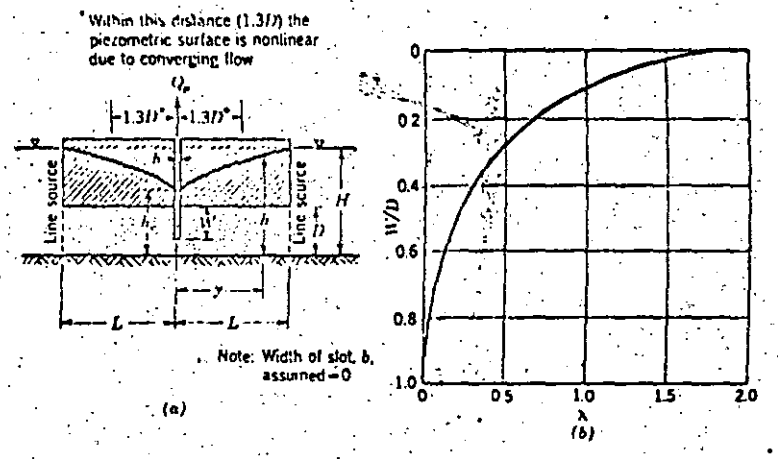
The *artesian* flow to a partially penetrating slot parallel to and midway between two line sources as shown in Fig. 3-19*a* can be expressed as follows:

$$Q_s = \frac{2kDx(H - h_s)}{L + \lambda D} \qquad (3\text{-}26)$$

where L = distance from slot to either of two line sources
λ = a factor which depends upon ratio of slot penetration W/D (Fig. 3-19*b*)

At distances y from the slot, in excess of about $1.3D$, the head h increases linearly as y increases and can be computed as follows:

FIG. 3-19. Artesian flow to a partially penetrating slot midway between and parallel to two line sources of seepage. (*a*) Cross section through flow system; (*b*) factor λ versus ratio W/D.



$$h = h_s + (H - h_s)\frac{y + \lambda D}{L + \lambda D} \qquad (3\text{-}27)$$

In the region close to the slot ($y < 1.3D$), the head h does not vary linearly with y because of the convergence of flow into the slot. Such a flow pattern is shown in Fig. 3-23. In the vicinity of the slot the head h at the top of the pervious stratum can be estimated graphically by drawing a smooth curve between the head h_s at the slot and that at $y = 1.3D$, computed from Eq. (3-27). This curve must be drawn tangent to the slope of the piezometric grade line at $y = 1.3D$, which slope is equal to $(H - h_s)/(L + \lambda D)$, as illustrated in Fig. 3-19*a*. The distance from the slot at which the head is not influenced by converging flow decreases with increasing slot penetration. However, the value of $1.3D$ suggested above usually can be used in design problems even where the percentage of slot penetration is relatively small.

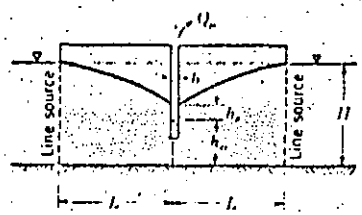
The flow to a partially penetrating slot midway between two line sources under *gravity*-flow conditions as shown in Fig. 3-20 can be estimated from the following equation:

$$Q_s = \left(0.73 + 0.27\frac{H - h_s}{H}\right)\frac{kx}{L}(H^2 - h_s^2) \qquad (3\text{-}28)$$

This equation is based on model studies by Chapman (16) for gravity flow from a line source to a single partially penetrating slot. In the case

of the model, some flow passed beneath the slot and entered it from the downstream side, whereas if the slot had been midway between two line sources, the flow pattern would have been symmetrical with respect to

FIG. 3-20. Gravity flow to a partially penetrating slot midway between and parallel to two line sources of seepage.



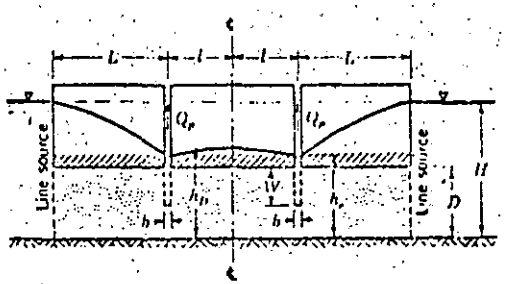
the slot. Therefore Eq. (3-28), which implies that the flow to a partially penetrating slot midway between two line sources is twice the flow from a single line source, is not exact, but merely an approximate means of estimating the flow required to produce a given head reduction. The model results were based on cases where L/H equaled or exceeded 3.

3-18. Flow to Two Partially Penetrating Slots Midway between and Parallel to Two Line Sources.

Consider a case where it is necessary to dewater a long excavation by two lines of closely spaced, partially penetrating wellpoints simulated by two slots as shown in Fig. 3-21. If the flow to both slots is symmetrical about the center line of the excavation, it can be considered as originating from two equidistant line sources as shown in Fig. 3-21.

The *artesian* flow from one source to the closest of the two slots can be computed from Eq. (3-22), using the value of F_A given in Fig. 3-17. Also, the head h_D midway between slots can be estimated from Eq. (3-23).

FIG. 3-21. Artesian flow to two partially penetrating slots from two line sources of seepage.



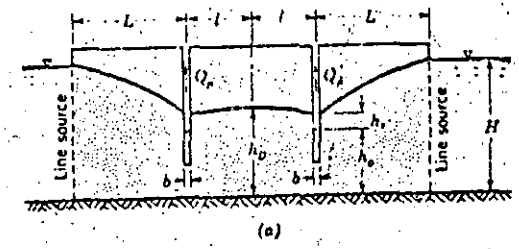
The head so computed will be fairly reliable except where the slots are very close to each other. In this case a slightly conservative estimate will be obtained from Eq. (3-23), which assumes that the slots are sufficiently distant so that neither affects the pressure distribution at or near the other.

The approximate relationship between flow from and drawdown caused by pumping two parallel slots partially penetrating a pervious substratum supplied by equidistant line sources under *gravity*-flow conditions as shown in Fig. 3-22*a* can be obtained from results of model studies conducted by Chapman (16) for values of L/H equal to or greater than 3. The flow from either slot can be computed from Eq. (3-24). The head h_D remaining midway between the two slots can be estimated from the following expression:

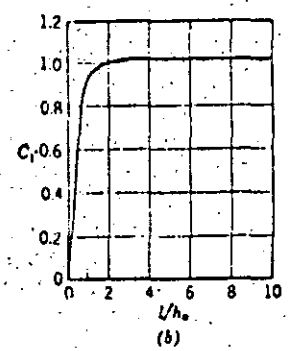
$$h_D = h_s \left[\frac{C_1 C_2}{L} (H - h_s) + 1 \right] \tag{3-29}$$

where C_1 and C_2 are obtained from Figs. 3-22*b* and 3-22*c*, respectively.

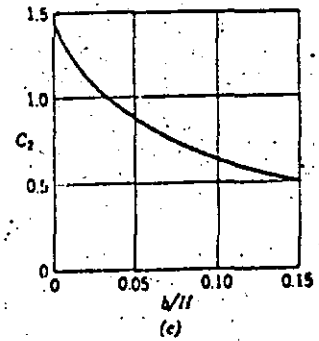
FIG. 3-22. Gravity flow to two partially penetrating slots from two line sources of seepage. (*a*) Cross section; (*b*) factor C_1; (*c*) factor C_2.



(*a*)



(*b*)



(*c*)

3-19. Flow Nets

The use of a graphical representation of flow through soil, and hence the graphical solution to the Laplace equation discussed in Chap. 2, is of great assistance in designing certain types of dewatering and pressure-relief well systems. Furthermore, it is generally easier to obtain a graphical solution (flow net) rather than a mathematical solution to the Laplace equation. The purpose of this section is to discuss briefly the basic relationships to be maintained in a flow net and the equations for seepage flow. Proficiency in constructing flow nets is developed by practice and the study of properly constructed nets. Excellent treatises on flow nets and their construction are given by Casagrande (17), Taylor (18), and Barron (19).

The path followed by a particle of water flowing through a saturated soil mass is called a *flow line*. Each flow line originates at a point of high head and terminates at a point where the net head is zero. The distribution of net head causing the flow through the soil mass is represented by equipotential lines. An *equipotential line* represents a contour of equal head; therefore piezometers installed with their tips located on such a line will register the same height of water.

A flow net may be constructed either to represent the plan view of the seepage pattern or a sectional view, or both, depending upon the needs of the designer. When constructing a flow net, the section must first be transformed into one where the horizontal and vertical permeabilities are equal, using the procedures for transformation described in Chap. 2, and then the following general rules must be observed:

1. Flow lines and equipotential lines intersect at right angles and form curvilinear squares or rectangles.

2. Where the entire section cannot be divided conveniently into squares, a row of rectangles will remain and the ratio of the lengths of the sides of each rectangle shall remain constant.

3. A discharge face under atmospheric pressure (in contact with air) is neither an equipotential nor a flow line, and therefore squares are incomplete; and flow as well as equipotential lines need not intersect such a boundary at right angles.

4. In gravity-flow systems, equipotential lines intersect the seepage line or phreatic surface at equal intervals of elevation, each interval being a constant fraction of the total net head H'.

From the flow net, the discharge q per unit width and the head h at any point can be determined by means of the following equations:

$$q = kH' \frac{N_f}{N_e} = k(H - h_e) \frac{N_f}{N_e} \tag{3-30}$$

$$h = \frac{n_e}{N_e} H' = \frac{n_e}{N_e} (H - h_e) \tag{3-31}$$

where k = coefficient of permeability of soil
N_f = number of flow channels in net
N_e = total number of equipotential drops between full head H and head h_e at point of flow exit
n_e = number of equipotential drops from exit to point at which head h is desired

In the above notational system H' equals $H - h_e$, and the unit discharge q equals the total flow Q divided by the length x of the flow system in a sectional flow net, and q equals Q divided by the effective thickness D of the pervious stratum in a plan flow net.

An example of a sectional flow net is shown in Fig. 3-23. This flow net shows the seepage pattern that would develop in an artesian stratum penetrated to its mid-depth by a slot and where the flow to the slot originates at two line sources equidistant from the slot. Also shown is the discharge q per unit length of slot corresponding to a head reduction $(H - h_e)$ of 30 ft at the slot.

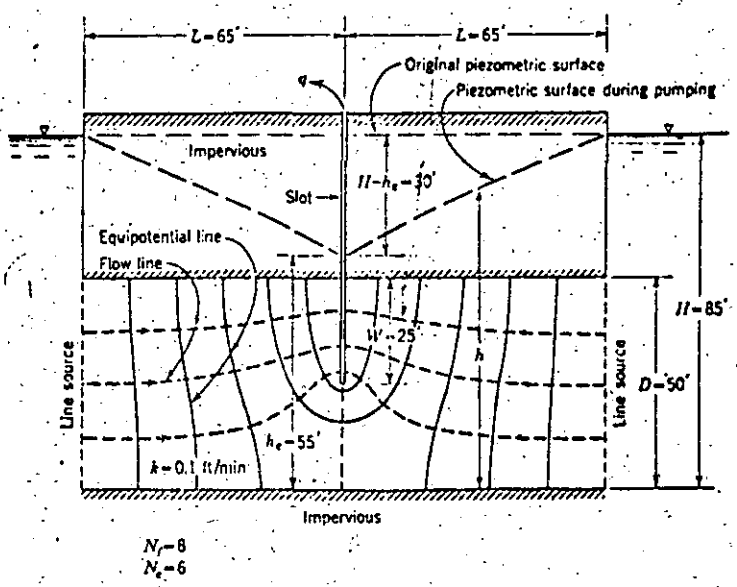


Fig. 3-23. Flow net for artesian flow to a partially penetrating slot midway between two line sources of seepage, all of infinite length.

The flow per ft of slot required for a drawdown of 30 ft is computed from Eq. (3-30) as follows:

$$q = k(H - h_e)N_f/N_e$$
$$q = 0.1 \times 30 \times \tfrac{8}{6} = 4.0 \text{ cfm per ft of slot}$$

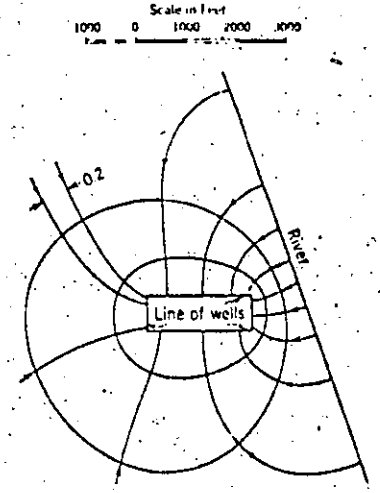
It should be noted that for the case illustrated in Fig. 3-23 the same unit flow could have been obtained more readily from Eq. (3-26). Since $W/D = 0.50$, the value of λ to be used in Eq. (3-26) as obtained from Fig. 3-19*b* is 0.2. Substituting this value of λ and those for the remaining factors shown in Fig. 3-23 into Eq. (3-26) results in the following value of q per unit length ($x = 1$) of slot, which is the same as that shown in Fig. 3-23:

$$q = \frac{2 \times 0.1 \times 50 \times 1 \times (85 - 55)}{65 + 0.2 \times 50} = 4.0 \text{ cfm} = 30 \text{ gpm}$$

An example of a plan flow net is shown in Fig. 3-24. This flow net shows the seepage pattern resulting from pumping a system of pressure relief wells which fully penetrates an artesian stratum and which has been simulated by a rectangular slot. In this flow net it was assumed that the seepage flow originates from a river simulated by a vertical line source of seepage. Also shown in Fig. 3-24 is the discharge Q required to produce a head reduction ($H - h_c$) of 50 ft at the slot. Had this plan flow net been constructed for a gravity- instead of artesian-flow pattern, the "equipotential lines" could be drawn as shown in Fig. 3-24 but would then correspond to $H^2 - h^2$ instead of $H - h$.

I-J

Fig. 3-24. Plan flow net and computation of flow to a relief well system.



Properties of pervious substratum:

$k = 500 \times 10^{-4}$ cm/sec = 0.10 ft/min
$D = 110$ ft
Flow to wells, artesian

From the flow net,

$$N_f = 12.2 \quad \text{and} \quad N_s = 3.0$$

The flow q per foot depth of pervious substratum is

$$q = k(H - h_c)\frac{N_f}{N_s} \tag{3-30}$$

For a 110-ft-thick pervious substratum, the flow Q_T is

$$Q_T = k(H - h_c)110\frac{N_f}{N_s}$$

or

$$\frac{Q_T}{H - h_c} = 0.10 \times 110 \times \frac{12.2}{3.0} = 44.7 \text{ cfm/ft drawdown}$$

To produce a drawdown ($H - h_c$) of 50 ft, the required total flow Q is

$$Q_T = 50 \times 44.7 = 2,235 \text{ cfm} = 16,700 \text{ gpm}$$

As illustrated above, the flow net can be a useful tool when designing dewatering systems, especially where complicated boundary conditions are present. However, such nets are primarily for analysis of two-dimensional flow problems as illustrated in Figs. 3-23 and 3-24 and therefore can give erroneous results if used to analyze problems which are three-dimensional. For example, had the well system and/or slot illustrated in Fig. 3-24 penetrated only 50 per cent of the thickness of the pervious stratum, the flow in the vicinity of the wells would converge upward similarly to that shown in Fig. 3-2*b*, and in this region the equipotential lines would not be vertical, as was assumed in constructing the plan flow net in Fig. 3-24. If, however, the penetration had been 90 per cent, the plan flow net would give essentially the correct pressure distribution. Accordingly, the flow net should be used with caution since plan flow nets cannot be expected to give exact results in the vicinity of partially penetrating well systems, the error being inversely proportional to the percentage of penetration. The plan flow net can give fairly accurate results for partially penetrating artesian well systems if the heads are adjusted, using procedures set forth in Sec. 3-20, to take into account the fact that the wells only partially penetrate the aquifer.

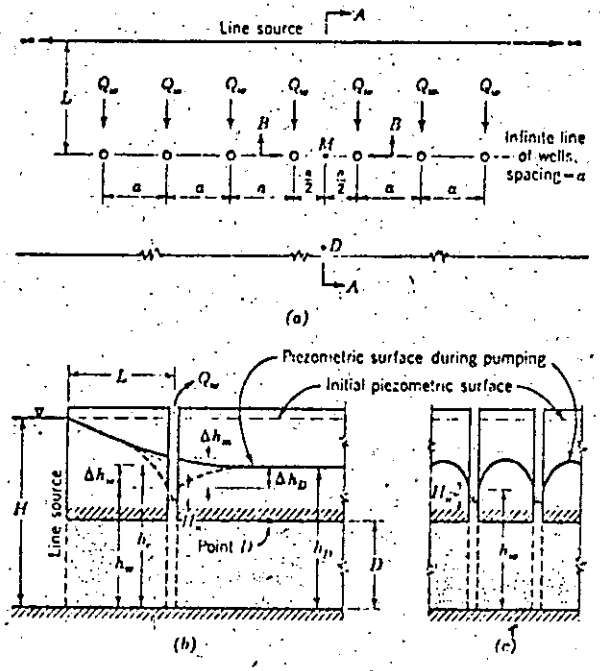
3-20. Head in Vicinity of Wells or Wellpoints Computed from Formulas for Slots or Plan Flow Nets

In the previous equations for flow to slots and resultant head reduction and the example of a plan flow net, it was assumed that the wells were spaced at sufficiently close intervals so that they could be simulated by a continuous slot. Engelund (20) has shown that after first computing the head reduction caused by pumping a continuous line slot used to simulate a well system, a head correction can be applied to this head reduction to account for the fact that the discharge system consists of a finite group of wells instead of a slot. Engelund's procedures were

developed for fully penetrating well systems; they can be extended to apply to partially penetrating systems as described below.

Consider an infinite line of fully penetrating *artesian* wells with spacing a supplied by an infinite-line source of seepage as shown in Fig. 3-25.

FIG. 3-25. Flow to infinite line of fully penetrating artesian wells from an infinite line source of seepage. (*a*) Plan; (*b*) section *A-A*; (*c*) section *B-B*.



Let Q_w equal the discharge per well. If the wells are replaced by a continuous fully penetrating slot of infinite length, the head reduction $(H - h_s)$ at the slot caused by a discharge Q_w to a slot of length a, as given by Eq. (3-6), is

$$H - h_s = \frac{Q_w I_s}{kDa} \tag{3-6}$$

However, an additional head loss Δh_w (measured below the head h_s at the slot) occurs because of the converging flow at the wells. This head loss is a function of well flow, well spacing and penetration, well radius, and thickness and permeability of the aquifer. It can be computed as follows for fully penetrating wells:

$$\Delta h_w = \frac{Q_w}{2\pi kD} \ln \frac{a}{2\pi r_w} \tag{3-32}$$

and from the following equation for partially penetrating wells:

$$\Delta h_w = \frac{Q_w \theta_w}{kD} \tag{3-32a}$$

where r_w = "effective" radius of well (as described in Sec. 3-27)
a = well spacing
θ_w = uplift factor whose value can be obtained from Fig. 3-38

The head reduction $(H - h_w)$ at the well, neglecting hydraulic head losses in the well, is that at the slot plus that due to the well; therefore

$$H - h_w = H - h_s + \Delta h_w \tag{3-33}$$

For fully penetrating wells

$$H - h_w = \frac{Q_w I_s}{kDa} + \frac{Q_w}{2\pi kD} \ln \frac{a}{2\pi r_w} \tag{3-34}$$

For partially penetrating wells

$$H - h_w = \frac{Q_w}{kD}\left(\frac{I_s}{a} + \theta_w\right) \tag{3-34a}$$

The head h_m midway between wells will exceed the head h_w at a well by an amount Δh_w, which can be computed from the following expression for fully penetrating wells

$$\Delta h_m = \frac{Q_w}{2\pi kD} \ln \frac{a}{\pi r_w} \tag{3-35}$$

and from

$$\Delta h_m = \frac{Q_w \theta_m}{kD} \tag{3-35a}$$

for partially penetrating wells where θ_m = midpoint uplift factor whose value can be obtained from Fig. 3-38.

Therefore, for fully penetrating wells

$$H - h_m = \frac{Q_w}{kDa} - 0.11\frac{Q_w}{kD} \tag{3-36}$$

and for partially penetrating wells

$$H - h_m = \frac{Q_w}{kD}\left(\frac{I_s}{a} + \theta_w - \theta_m\right) \tag{3-36a}$$

At a distance downstream from the well system the head will exceed that

at a well by an amount Δh_n, where Δh_n can be computed from the following equation for fully penetrating wells

$$\Delta h_n = \frac{Q_w}{2\pi kD} \ln \frac{a}{2\pi r_w} \tag{3-37}$$

and

$$\Delta h_n = \frac{Q_w \theta_w}{kD} \tag{3-37a}$$

for partially penetrating wells.

It should be noted that when the increment Δh_n is added to h_w, the resulting head h_n is equal to the head h_s that would occur at the slot for the same total discharge. Also, it can be shown that the following equation results for fully penetrating wells when Eqs. (3-6) and (3-34) are combined to eliminate Q_w after substituting h_n for h_s in Eq. (3-6):

$$\frac{h_n - h_w}{H - h_n} = \frac{a}{2\pi L} \ln \frac{a}{2\pi r_w} \tag{3-37b}$$

The method described above also can be used to compute approximately the heads at, midway between, and downstream from wells in cases where a plan flow net is used initially to determine the flow-head-reduction relationship for artesian cases with complex boundary conditions, and the wells are spaced proportionally to the flow lines as shown in Fig. 3-26. In such a case the well system first can be simulated by a continuous slot to determine the total discharge Q_T corresponding to the desired head reduction $(H - h_s)$ at the slot. Dividing this flow by the number of wells n in the system results in the average flow Q_w per well. The head reduction $(H - h_w)$ at a well is that at the slot plus Δh_w computed from Eq. (3-32), or Eq. (3-32a). Thus, for fully penetrating wells

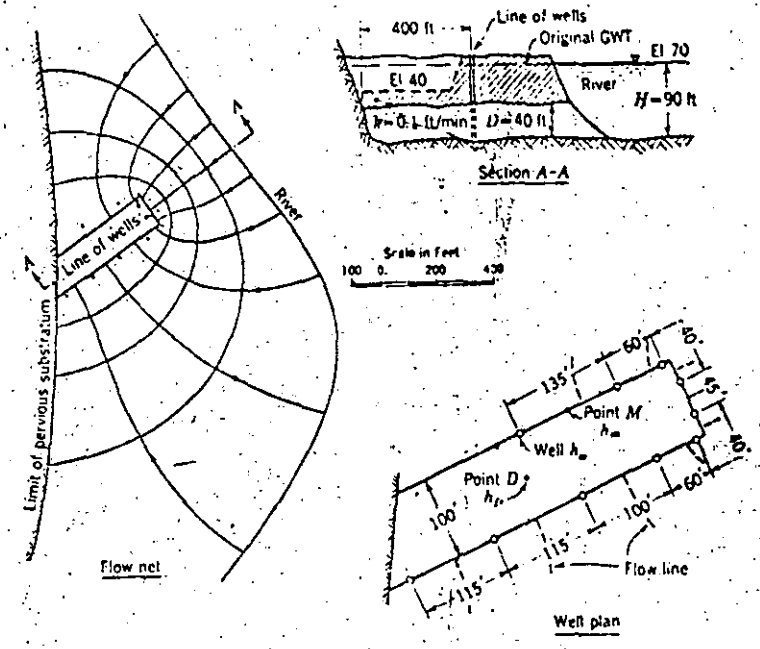
$$H - h_w = \frac{Q_w}{kD}\left(\frac{nN_s}{N_f} + \frac{1}{2\pi} \ln \frac{a}{2\pi r_w}\right) \tag{3-38}$$

and for partially penetrating wells

$$H - h_w = \frac{Q_w}{kD}\left(\frac{nN_s}{N_f} + \theta_w\right) \tag{3-38a}$$

In a similar manner the head midway between wells (point M, Fig. 3-26) can be estimated from either Eqs. (3-35) and (3-38) or Eqs. (3-35a) and (3-38a), and the head at the center of the system (point D, Fig. 3-26) from either Eqs. (3-37) and (3-38) or Eqs. (3-37a) and (3-38a). When the above procedure is used, the average well spacing should be used to compute θ_w and the head at the wells and at the center of the well system, and the maximum spacing is used to compute θ_m and the heads midway between wells, except where wells are spaced proportionally to flow lines, in which case the average well spacing can be used. A typical set of calculations is shown in Fig. 3-26.

Fig. 3-26. Design of a deep-well system for pressure relief using a flow net.



Problem. Given the flow net, the data above, and the plan of wells as shown, compute the well flow required to reduce the head in the sand stratum to el 40 at point D, the corresponding head h_w at the wells, h_m midway between wells, and h_n at the center of the excavation. Assume that wells fully penetrate the pervious stratum and that $D = 40$ ft, $k = 500 \times 10^{-4}$ cm per sec $= 0.1$ ft per min, and $r_w = 1.0$ ft.

Solution. Flow to slot (or wells) from flow net

$$Q_T = k(H - h_s)\frac{N_f}{N_s} D = 0.1(90 - 60)\frac{10.0}{4.0} \times 40 = 300 \text{ cfm} = 2{,}250 \text{ gpm}$$

Assume 10 wells located as shown in "Well plan." Since a well has been spaced at the center of each flow channel, the flow per well is the same for all wells. Thus $Q_w = 225$ gpm or 30 cfm per well.

From Eq. (3-38),

$$H - h_w = \frac{30}{0.1(40)}\left[10\left(\frac{4}{10}\right) + \frac{1}{2\pi} \ln \frac{90}{2\pi(1)}\right] = 33.2 \text{ ft}$$

Since the average well spacing a is approximately 90 ft, compute Δh_m from Eq. (3-35) for $a = 90$ ft.

$$\Delta h_m = \frac{30}{2\pi(0.1)40} \ln \frac{90}{\pi(1)} = 4.0 \text{ ft}$$

Thus $H - h_w = H - h_c - \Delta h_w = 33.2 - 4.0 = 29.2$ ft

From Eq. (3-37), for $a = 90$ ft,

$$\Delta h_D = \frac{30}{2\pi(0.1)40} \ln \frac{90}{2\pi(1)} = 3.2 \text{ ft}$$

Thus $H - h_D = H - h_c - \Delta h_D = 33.2 - 3.2 = 30.0$ ft

The heads h_w, h_m, and h_D in terms of elevation are as follows:

$h_w = 70 - 33.2 = 36.8$ ft MSL
$h_m = 70 - 29.2 = 40.8$ ft MSL
$h_D = 70 - 30.0 = 40.0$ ft MSL

Since the GWT is to be lowered to el 40 at point D and since the computed head at this point is at el 40.0, $Q_w = 30$ cfm, or 225 gpm per well will produce the required head reduction. The values of $(H - h_w)$, Δh_m, and Δh_D also can be computed from Eqs. (3-38a), (3-35a), and (3-37a), respectively, as shown below. Note that the values so obtained are identical to those computed above.

From Fig. 3-38, $\theta_w = 0.42$ and $\theta_m = 0.53$ for $a/r_w = 90$ and $W/D = 100\%$.

From Eq. (3-38a),

$$H - h_w = \frac{30}{0.1(40)}\left[10\left(\frac{4}{10}\right) + 0.42\right] = 33.2 \text{ ft}$$

From Eq. (3-35a),

$$\Delta h_m = \frac{30(0.53)}{0.1(40)} = 4.0 \text{ ft}$$

From Eq. (3-37a),

$$\Delta h_D = \frac{30(0.12)}{0.1(40)} = 3.2 \text{ ft}$$

As was the case for artesian-flow systems, a correction for head reduction for a discharge system of *gravity* wells can be applied to the head reduction computed for a slot to account for the fact that the discharge system consists of a finite system of wells rather than a continuous line slot. The head reduction $(H - h_c)$ caused by pumping a fully penetrating slot in a gravity-flow system supplied by a single line source can be expressed by

$$H^2 - h_c^2 = \frac{2Q_w L}{ka} \tag{3-39}$$

The additional head loss Δh_w^2 due to a fully penetrating gravity well is

$$\Delta h_w^2 = h_c^2 - h_w^2 = \frac{Q_w}{\pi k} \ln \frac{a}{2\pi r_w} \tag{3-40}$$

The head reduction at the well is the sum of the head drops given by the above two equations, or

$$H^2 - h_w^2 = \frac{2Q_w L}{ka} + \frac{Q_w}{\pi k} \ln \frac{a}{2\pi r_w} \tag{3-41}$$

The head h_m midway between wells will exceed the head at a well, indicated by the term $(H^2 - h_w^2)$, by an amount Δh_m^2, which can be computed from the following equation:

$$\Delta h_m^2 = \frac{Q_w}{\pi k} \ln \frac{a}{\pi r_w} \tag{3-42}$$

The head h_m can be computed from the following equation:

$$H^2 - h_m^2 = H^2 - h_w^2 - \Delta h_m^2 \tag{3-43}$$

At a distance downstream from the well system the head h_D can be computed from the term $(H^2 - h_D^2)$, where

$$H^2 - h_D^2 = H^2 - h_w^2 - \Delta h_D^2 \tag{3-44}$$

and where Δh_D^2 is computed as follows:

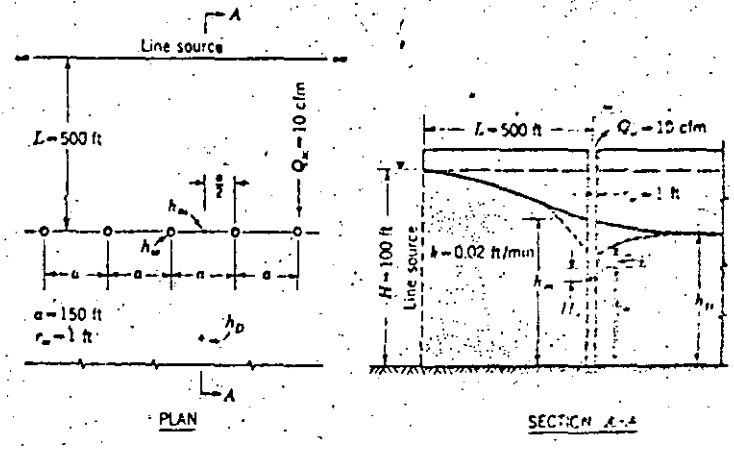
$$\Delta h_D^2 = \frac{Q_w}{\pi k} \ln \frac{a}{2\pi r_w} \tag{3-45}$$

It should be noted that when the increment Δh_D^2 is added to $H^2 - h_w^2$, the resulting head h_D is equal to the head h_c that would occur at the slot for the same total discharge. Also, it can be shown that the following equation results for fully penetrating wells when Eqs. (3-39) and (3-41) are combined to eliminate Q_w after substituting h_D for h_c:

$$\frac{h_D^2 - h_w^2}{H^2 - h_D^2} = \frac{a}{2\pi L} \ln \frac{a}{2\pi r_w} \tag{3-45a}$$

Equations (3-39) to (3-45a) also can be applied to systems of partially penetrating gravity wells provided the discharge Q_w per well is computed from an appropriate equation for gravity flow to a partially penetrating slot. A numerical example illustrating the use of the above equations is given in Fig. 3-27.

FIG. 3-27. Approximate method for computing head near a system of fully penetrating gravity wells with a line source of seepage.



Problem. *From data shown on plan and section above, compute the head h_w at a well, h_m midway between wells, and h_0 downstream from the well system.*

Solution

From Eq. (3-39), $H^2 - h_w^2 = \dfrac{2Q_w L}{ka} = \dfrac{2 \times 10 \times 500}{0.02 \times 150} = 3{,}333$

From Eq. (3-41), $H^2 - h_m^2 = 3{,}333 + \Delta h_w^2$

From Eq. (3-40), $\Delta h_w^2 = \dfrac{10}{\pi(0.02)} \ln \dfrac{150}{2\pi(1)} = 504$

or

$$h_w = \sqrt{H^2 - 3{,}333 - \Delta h_w^2} = \sqrt{100^2 - 3{,}333 - 504} = 78.5 \text{ ft}$$

Also $H^2 - h_w^2 = 3{,}333 + 504 = 3{,}837$

From Eq. (3-43), $H^2 - h_m^2 = 3{,}837 - \Delta h_m^2$

From Eq. (3-42), $\Delta h_m^2 = \dfrac{10}{\pi(0.02)} \ln \dfrac{150}{\pi(1)} = 615$

Therefore

$$h_m = \sqrt{H^2 - 3{,}837 + \Delta h_m^2} = \sqrt{100^2 - 3{,}837 + 615} = 82.3 \text{ ft}$$

From Eq. (3-44), $H^2 - h_0^2 = 3{,}837 - \Delta h_0^2$

From Eq. (3-45), $\Delta h_0^2 = \dfrac{10}{\pi(0.02)} \ln \dfrac{150}{2\pi(1)} = 504$

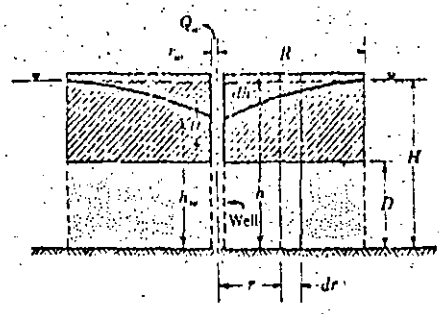
Thus

$$h_0 = \sqrt{H^2 - 3{,}837 + \Delta h_0^2} = \sqrt{100^2 - 3{,}837 + 504} = 81.6 \text{ ft}$$

3-21. Flow to a Single Well—Circular Source

Fully Penetrating Artesian Well. Consider a well installed in a pervious, homogeneous, isotropic stratum having a constant thickness D and permeability k and which is bounded above and below by parallel impervious strata. Further assume that the well screen fully penetrates the pervious stratum and that the well has a radius r_w and in plan is installed in the center of a circular island having a radius R. These conditions are illustrated in Fig. 3-28. The equation for flow to the well

FIG. 3-28. Flow to a fully penetrating artesian well from a circular source of seepage.



Q_w can be developed by considering the flow through a cylindrical element of radius r, thickness dr, and height D. Using Darcy's law and assuming a steady state of flow,

$$i = dh/dr \quad \text{and} \quad A = 2\pi r D \tag{3-46}$$

Substituting Eq. (3-46) into Eq. (3-1) and integrating this equation between the limits ($r = r_w$, $h = h_w$) and ($r = R$, $h = H$) results in the following equation for flow to the well:

$$Q_w = \frac{2\pi k D(H - h_w)}{\ln (R/r_w)} \tag{3-47}$$

The head h at any distance r from the well, where $r_w \leq r \leq R$, can be obtained from the following equation, which results from substituting Eq. (3-46) into Eq. (3-1), integrating the resulting expression, and solving for the constant of integration for the boundary conditions ($r = r_w$, $h = h_w$) and ($r = R$, $h = H$).

$$h = \frac{Q_w}{2\pi k D} \ln \frac{r}{r_w} + h_w \tag{3-48}$$

The drawdown ($H - h$) at r distance from the well can be obtained from either of the following two equations:

$$H - h = \frac{Q_w}{2\pi k D} \ln \frac{R}{r} \tag{3-49}$$

$$H - h = h_w + \frac{H - h_w}{\ln (R/r_w)} \ln \frac{r}{r_w} \tag{3-50}$$

The equations developed above lead to the following significant facts concerning flow to a single, fully penetrating artesian well with a circular source of seepage:

1. The drawdown at the well and at any distance r from the well varies linearly with the well discharge.
2. For a given well discharge the drawdown at a point varies inversely as the logarithm of the distance from the point to the well.
3. A plot of drawdown to an arithmetic scale vs. distance from the well to a logarithmic scale is linear.
4. The ratio of drawdown at any point to that at the well is constant, i.e., independent of the well discharge.

It is important to note that the above equations were developed on the basis that the head at the well, h_w, was equal to the water level in the well. This assumption is valid only when there is no hydraulic head loss

in the well. As some head is required to force the water through the filter and well screen, the water level in the well will be lower than the head in the pervious stratum at the well, h_w, by an amount H_w, which is equal to the hydraulic head loss in the well. This condition is illustrated in Fig. 3-28. The equations developed above for head reduction and discharge from artesian wells are valid provided h_w is considered as the head at the periphery of the well and not the water level in the well itself. Hydraulic head losses in wells and the evaluation thereof are discussed later in this chapter.

It is possible that an artesian well may not be located at the center of the circle of influence and/or that the head at the periphery of the circle of influence is not a constant value H, as assumed in the previous development. These conditions were recognized by Muskat (21), who developed the following relationship for flow to an artesian well located a distance E from the center of the circle of influence:

$$Q_w = \frac{2\pi kD(H - h_w)}{\ln [(R^2 - E^2)/Rr_w]} \tag{3-51}$$

From the above equation it can be shown that for values of E less than about $0.7R$, the computed discharge Q_w is within about 10 per cent of that given by Eq. (3-47), which applies to conditions where the well is at the center of the circle of influence. Thus the eccentricity of the well with respect to circle of influence can be neglected unless the eccentricity is extremely large. Muskat also found that Eq. (3-47) will be valid even when the pressure head H varies along the periphery of the circle of influence provided the average value of H at the periphery is used in this equation. Thus Eq. (3-47) is essentially valid for a wide range of conditions provided the flow to the well is artesian.

Partially Penetrating Artesian Well. If the screen of an artesian well does not fully penetrate the pervious stratum, the pattern of flow in the aquifer in the vicinity of the well will deviate from that for a fully penetrating well. Therefore the flow required to produce a given drawdown at the well depends upon the depth to which the well screen penetrates the pervious stratum. The equation for flow to a partially penetrating artesian well is as follows:

$$Q_{wp} = \frac{2\pi kD(H - h_w)G}{\ln (R/r_w)} \tag{3-52}$$

where G is a correction factor for partial penetration, which is equal to the ratio of flow from the partially penetrating well Q_{wp} to that for a fully penetrating well for the same drawdown $(H - h_w)$ at the periphery of the wells. Reasonable values of G can be obtained from the following equation developed by Kozeny (22):

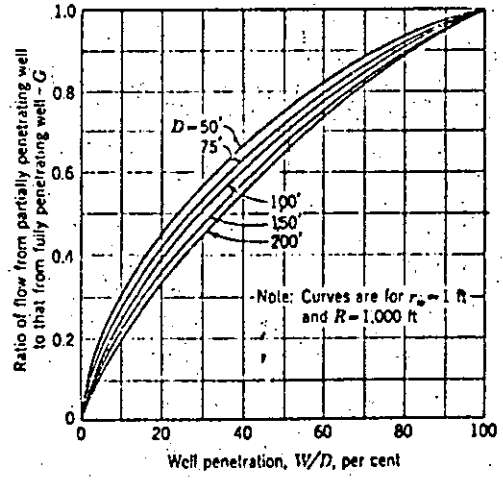
$$G = \frac{W}{D}\left(1 + 7\sqrt{\frac{r_w}{2W}}\cos\frac{\pi W/D}{2}\right) \tag{3-53}$$

where W/D equals the penetration of the well screen into the pervious stratum expressed as a decimal. More exact values of G can be obtained from the following formula developed by Muskat (21):

$$G = \frac{D}{2W}\left[2\ln\frac{4D}{r_w} - \ln\frac{\Gamma(0.875W/D)\Gamma(0.125W/D)}{\Gamma(1 - 0.875W/D)\Gamma(1 - 0.125W/D)}\right] - \ln\frac{4D}{R} \tag{3-54}$$

where the notations in Eqs. (3-53) and (3-54) are identical, and Γ is the gamma function. Values of G for a typical large-diameter well ($r_w = 1$ ft) with a radius of influence of 1,000 ft are shown in Fig. 3-29.

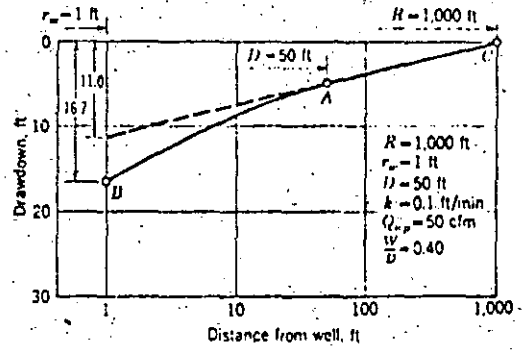
FIG. 3-29. Relation between flow from a partially penetrating artesian well in a homogeneous foundation and that from a fully penetrating well.



Unfortunately, the shape of the drawdown curve for a partially penetrating artesian well cannot be obtained directly from Eq. (3-52) using the value of G from either Eq. (3-53) or (3-54), and, to date, a satisfactory general expression for the drawdown has not been developed. However, a reasonable approximation of the shape of the drawdown curve at the top of the pervious stratum can be obtained by assuming that the effect of well penetration on drawdown is insignificant at distances from the well exceeding the effective thickness of the pervious stratum. There-

fore, to obtain the drawdown curve, it is first necessary to compute the flow Q_{w_p} from the partially penetrating well from Eq. (3-52), using a value of G obtained from either Eq. (3-53) or (3-54). Next, from Eq. (3-49), compute the drawdown at the well ($H - h_w$) that would occur if a fully penetrating well were pumped at a discharge equal to the value of Q_{w_p} computed above. Plot the drawdown for the fully penetrating well versus the log of the distance from the well as shown by line AC in Fig. 3-30 [it is evident from Eq. (3-50) that the drawdown curve will be linear on such a plot]. Draw a curved line from the point (h_w, r_w) point B in Fig. 3-30 for the partially penetrating well to a point A on the drawdown curve for a fully penetrating well located at distance D from the well as shown in Fig. 3-30. This combined curve, which at point A is tangent to the drawdown curve for a fully penetrating well, approximates the desired drawdown curve at the top of the pervious stratum. The drawdown curve also can be established by means of flow nets. However, drawing a flow net for radial flow is quite time-consuming and may not be warranted in most cases.

FIG. 3-30. Method for estimating shape of drawdown curve for a partially penetrating artesian well.



For 40 per cent penetration, $G = 0.66$ from Eq. (3-54), and for $Q_{w_p} = 50$ cfm, the drawdown at the well from Eq. (3-52) is

$$H - h_w = \frac{Q_{w_p} \ln (R/r_w)}{2\pi k D G} = \frac{50 \ln (1{,}000/1)}{2\pi(0.1)(50)(0.66)} = 16.7 \text{ ft}$$

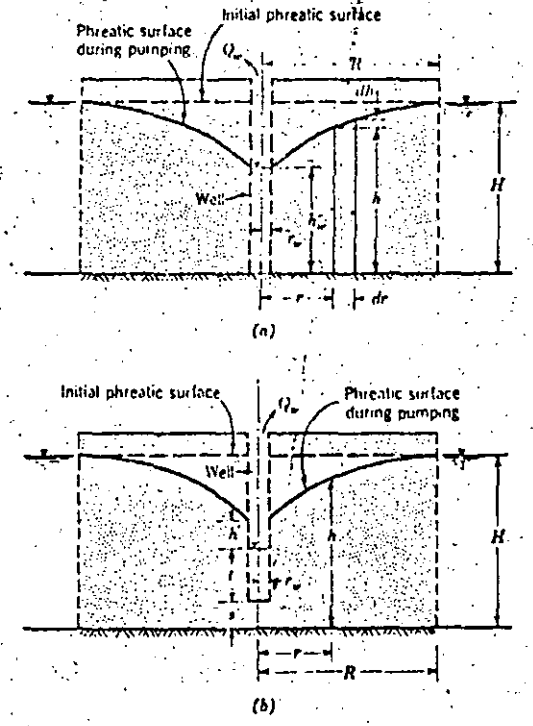
The drawdown at a fully penetrating well for a flow of 50 cfm is

$$H - h_w = \frac{Q_w \ln (R/r_w)}{2\pi k D} = \frac{50 \ln (1{,}000/1)}{2\pi(0.1)50} = 11.0 \text{ ft}$$

Construct drawdown curve BAC for partially penetrating well as indicated. This curve represents the drawdown at the top of the pervious stratum.

Fully and Partially Penetrating Gravity Well. A gravity well is one that penetrates a pervious stratum containing a free water table. Consider a vertical well that fully penetrates a homogeneous, isotropic, pervious stratum containing a horizontal free water table and bounded at its base by a horizontal impervious stratum as shown in Fig. 3-31*a*. Also

FIG. 3-31. Flow to a gravity well from a circular source of seepage. (*a*) Conditions assumed in developing Dupuit equation for a fully penetrating well; (*b*) notations for flow to and drawdown caused by pumping either a partially or fully penetrating well, taking into account height of free discharge.



consider the well to be located in plan in the center of a circular island having a radius R. If the well is pumped until a condition of equilibrium is attained, the equation for well flow can be developed in a manner similar to that for an artesian well. In the development of the equation it is assumed that the hydraulic gradient is constant with depth and equal to the slope of the drawdown curve at any particular point. This last assumption is referred to as the *Dupuit-Forchheimer assumption*.

64

Applying Darcy's law to the flow through a cylinder of radius r, thickness dr, and height h, shown in Fig. 3-31*a*, the flow through the cylinder can be expressed as

$$Q_w = kiA \tag{3-1}$$

and

$$i = \frac{dh}{dr} \quad \text{and} \quad A = 2\pi rh \tag{3-55}$$

Substituting Eq. (3-55) into Eq. (3-1) results in the following expression for flow Q_w:

$$Q_w = k\frac{dh}{dr}2\pi rh \tag{3-56}$$

Integrating Eq. (3-56) and substituting the boundary conditions $h = H$ at $r = R$ and $h = h_w$ at $r = r_w$ gives the following expression for flow from a gravity well:

$$Q_w = \frac{\pi k(H^2 - h_w^2)}{\ln (R/r_w)} \tag{3-57}$$

The head h at a distance r from the well, where $r_w \leq r \leq R$, can be obtained by integrating Eq. (3-56), which results in the following equation:

$$h^2 - h_w^2 = \frac{Q_w}{\pi k}\ln\frac{r}{r_w} \tag{3-58}$$

or

$$h^2 = \frac{Q_w}{\pi k}\ln\frac{r}{r_w} + h_w^2 \tag{3-58a}$$

The drawdown $(H - h)$ at distance r from the gravity well cannot be expressed as readily as that for an artesian well. However, a similar type of term $(H^2 - h^2)$, analogous to the term $(H - h)$ for artesian wells, is of considerable significance in gravity-well systems and can be expressed as follows:

$$H^2 - h^2 = \frac{Q_w}{\pi k}\ln\frac{R}{r} \tag{3-59}$$

From a comparison between Eqs. (3-49) and (3-59) it can be seen that for both artesian and gravity wells supplied by a circular source, the head reduction at a given point is a function of the product $Q_w \ln R/r$. Generally, this product must be evaluated in the design and analysis of all well systems where the source of flow is circular.

Also significant is the fact that in a frictionless gravity well the water level in the well will be lower than the piezometric surface at the periphery of the well because of vertical drainage occurring at the periphery of the well. This is in contrast to a frictionless artesian well where the water level inside the well and piezometric surface at the periphery of the well

are coincident. In the gravity well the difference in the two water levels or height of free discharge surface h' and resulting phreatic surface shown in Fig. 3-31*b* has been the subject of numerous investigations (23-25). The results of these investigations can be summarized as follows:

1. The flow to a fully penetrating gravity well can be estimated accurately from Eq. (3-57), regardless of the depth to which the water in the well is lowered, using the height of water $t + s$ for the term h_w (Fig. 3-31*b*). However, the capacity of the well is limited by the amount of well screen remaining below the ground-water level at the well; accordingly, the submerged portion of the screen must be of sufficient length, size, and capacity to admit the flow.

2. At distances from the well exceeding approximately 1.0 to 1.5 times the height H to the original ground-water table, the drawdown will be equal to that computed from Eq. (3-58).

3. At distances from the well less than about 1.0 to 1.5H, the drawdown will be less than that computed from Eq. (3-58), and the discrepancy between the actual drawdown and that computed from Eq. (3-58) will increase as the drawdown at the well increases.

In those cases where it becomes necessary to compute the drawdown in the immediate vicinity of a gravity well, for example, at distances less than about 1.0H to 1.5H, the following empirical equations developed by Boreli (25) can be used. It should be noted that these equations are applicable to partially as well as fully penetrating gravity wells.

The discharge from any gravity well fed by a circular source of seepage can be expressed as follows:

$$Q_w = \frac{\pi k[(H - s)^2 - t^2]}{\ln (R/r_w)}\left[1 + \left(0.30 + \frac{10r_w}{H}\right)\sin\frac{1.8s}{H}\right] \tag{3-60}$$

where the notations are shown in Fig. 3-31*b*.

Equations for the phreatic surface from which the drawdown can be computed are given below, the value of Q_w being obtained from Eq. (3-60).

For r/h greater than 1.5, $H^2 - h^2 = \dfrac{Q_w}{\pi k}\ln\dfrac{R}{r}$ (3-61)

which is the Dupuit-Forchheimer equation developed previously.

For r/h less than 1.5, $H - h = \dfrac{Q_w P \ln (10R/H)}{\pi kH[1 - 0.8(s/H)^{1.5}]}$ (3-62)

Boreli found that the function P depended upon the r/h ratio and proposed the following two equations for P:

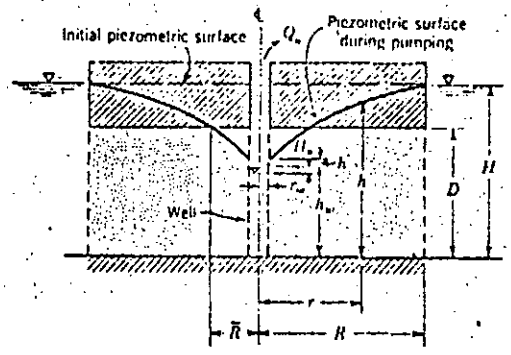
For $0.3 < r/h < 1.5$, $P = 0.13\ln\dfrac{R}{r}$ (3-63)

For $r/h < 0.3$,
$$P = C_r + \Delta C \tag{3-64}$$
where
$$C_r = 0.13 \ln \frac{R}{r} - 0.0123 \ln^2 \frac{R}{10r} \tag{3-65}$$
and
$$\Delta C = \frac{s}{h}\left[\left(\frac{1}{2.3}\ln\frac{R}{10r}\right)\left(1.2\frac{s}{H} - 0.48\right) + 0.113 \ln\frac{2.4H}{R}\ln\frac{R}{34r}\right] \tag{3-66}$$

It is apparent that considerable computation is required to determine the height of the phreatic surface and resulting drawdown in the immediate vicinity of a gravity well (r/h less than 0.3). The drawdown in this zone usually is not of special interest in dewatering systems and seldom needs to be computed. However, it is always necessary to compute the water level in the well for selection and design of the pumping equipment. Also, it should be noted that the above equations pertain to frictionless wells and do not consider the effects of hydraulic head losses on the phreatic surface and elevation of the water surface in the well. These factors are discussed later in this chapter.

Fully Penetrating Artesian-Gravity Well. When pumping from an artesian well installed in a pervious stratum, it may be possible at high pumping rates to lower the water at the well to a level below the top of the pervious stratum as indicated in Fig. 3-32. Under these conditions the flow pattern close to the well is similar to that for a gravity well,

FIG. 3-32. Flow to a combined artesian-gravity well from a circular source of seepage.



whereas at distances farther from the well the flow is artesian. This type of well is referred to as a *combined artesian-gravity well*. If such a well fully penetrates the pervious stratum, the flow from it can be computed from the following expression developed by Muskat (21):

$$Q_w = \frac{\pi k(2DH - D^2 - h_w^2)}{\ln (R/r_w)} \tag{3-67}$$

where the notations used in the above equation are shown in Fig. 3-32. Muskat further shows that the head h at any distance r from the well can be obtained from the following equation:

$$h = \frac{H - D}{\ln (R/r_w)} \ln \frac{r}{r_w} + \sqrt{D^2 - \frac{D^2 - h_w^2}{\ln (R/r_w)} \ln \frac{R}{r}} \tag{3-68}$$

The distance $\bar{R}$, measured from the well, at which the flow changes from artesian to gravity can be computed from the following equation:

$$\ln \bar{R} = \frac{(D^2 - h_w^2) \ln R + 2D(H - D) \ln r_w}{2DH - D^2 - h_w^2} \tag{3-69}$$

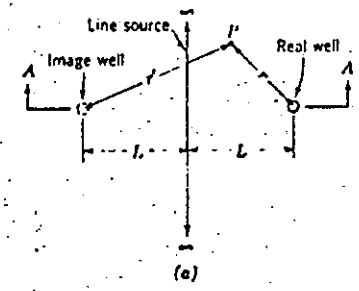
Equations (3-67) and (3-68) are based on the assumption that the head h_w at the well is at the same elevation as the water surface in the well. As shown previously, this will not be true where the drawdown is relatively large. In the latter case, the head at and in the close vicinity of the well can be computed from Eqs. (3-63) to (3-66). In these equations the value of Q_w used is that computed from Eq. (3-67), assuming h_w equal to the height of water in the well, and the value of $\bar{R}$ computed from Eq. (3-69) is used in lieu of R.

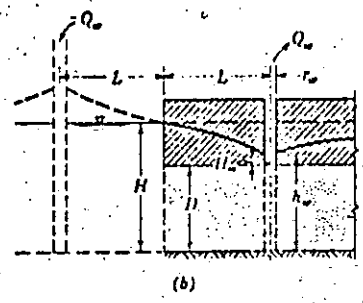
3-22. Flow to a Single Well—Line Source

Conditions may arise where it is necessary to dewater an excavation located close to the bank of a river, canal, or other body of water, where the primary source of water supplying the pervious strata originates from the adjacent body of water. In such cases the bank or shore line can be considered in plan as an infinite line source of seepage. It is of interest to consider the relationship between well flow, drawdown at the well, and drawdown at any point away from the well for this condition. In addition, such considerations are significant since they introduce a valuable procedure for solving dewatering problems, namely, *the method of image wells*.

Consider a pervious stratum bounded above and below by horizontal impervious strata and having a vertical face or bank infinite in extent in plan adjacent to a body of water. Assume that an *artesian* well fully penetrates the pervious stratum and is located at a distance L from the bank, as shown in Figs. 3-33*a* and 3-33*b*. To determine the flow to and drawdown caused by the well, the body of water is replaced by a continuation of the pervious stratum and an imaginary recharge well, or *image well*, supplying the pervious stratum with the same quantity of water as

that being pumped from the real well. The image well is located as the mirror image of the real well with respect to the line source, as shown in Figs. 3-33*a* and 3-33*b*. Using a procedure by Forchheimer (26) and Dachler (27) of computing the drawdown at a point caused by a group of wells as the sum of the drawdowns at the point due to each well in the group [Eq. (3-76)], discussed in Sec. 3-23 on multiple-well systems, the expression for drawdown at a point r distant from the well can be developed as follows. The drawdown at point P under artesian conditions can be expressed by





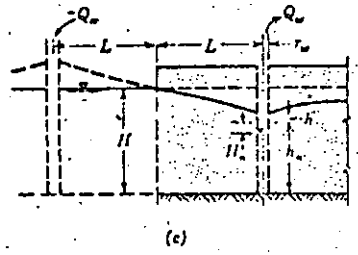


FIG. 3-33. Flow to a fully penetrating well from a line source of seepage. (*a*) Plan; (*b*) section *A-A* for an artesian well; (*c*) section *A-A* for a gravity well.

$$H - h = \frac{1}{2\pi kD} \sum_{i=1}^{i=n} Q_{wi} \ln \frac{R_i}{r_i} \quad (3\text{-}70)$$

Considering the flow from the real well to be positive and the flow into the image well as negative and both wells to have a radius of influence R_i equal to R, Eq. (3-70) reduces to

$$H - h = \frac{1}{2\pi kD} \left(Q_w \ln \frac{R}{r} - Q_w \ln \frac{R}{r'} \right) \quad (3\text{-}70a)$$

or

$$H - h = \frac{Q_w}{2\pi kD} \ln \frac{r'}{r} \quad (3\text{-}70b)$$

where r' and r are the distances from the point at which the drawdown is computed to image well and real well, respectively, as shown in Fig. 3-33*a*. The drawdown at the well can be determined from the above equation by considering that as r approaches the radius of the well r_w, distance r' approaches the value $2L$ and the head h approaches the head at the well h_w. Making the above substitutions in Eq. (3-70*b*) gives the following equations for drawdown at the well

$$H - h_w = \frac{Q_w}{2\pi kD} \ln \frac{2L}{r_w} \quad (3\text{-}71)$$

and flow to the well

$$Q_w = \frac{2\pi kD(H - h_w)}{\ln (2L/r_w)} \quad (3\text{-}72)$$

A comparison between Eq. (3-72) for flow to a well from an infinite line source and Eq. (3-47) for well flow from a circular source indicates that where $2L = R$, the flow per foot of drawdown at the well computed from both equations will be the same. If, however, $2L$ is greater than R, the effect of the line source on flow to a single well can be neglected because the well flow required to produce a given head reduction will be governed by the radius of influence and not the line source of seepage.

The flow to a fully penetrating *gravity* well from an infinite line source can be expressed as

$$Q_w = \frac{\pi k(H^2 - h_w^2)}{\ln (2L/r_w)} \quad (3\text{-}73)$$

for conditions as shown in Fig. 3-33*c*. The head h_p at any point P can be computed from the following equation:

$$H^2 - h_p^2 = \frac{Q_w}{\pi k} \ln \frac{r'}{r} \quad (3\text{-}74)$$

3-23. Multiple-well Systems

Most dewatering systems consist of a number of wells and/or wellpoints which are pumped to provide the required relief of substratum pressures or reduction of ground-water levels. One of the major problems in designing such a system is to determine the number and spacing of wells or wellpoints required to reduce the pressures or ground-water level to the required levels. Therefore it is necessary to determine the relationship between drawdown caused by a group of wells or wellpoints and the flow from such systems. Previously, equations were developed for flow to and drawdown caused by pumping a system of multiple wells by first determining flow to and drawdown at an equivalent slot and then adjusting the head in the vicinity of the slot to allow for the effects of the wells. Since in some cases this procedure is of limited accuracy, more exact methods are presented in the following paragraphs.

General Case. The influence of a system of *artesian* wells on the drawdown $(H - h)$ at any point was first determined by Forchheimer (26). The general equation for drawdown developed by him and later modified by Dachler (27) is as follows for a group of artesian wells:

$$H - h = \frac{1}{2\pi kD} \left(Q_{w1} \ln \frac{R_1}{r_1} + Q_{w2} \ln \frac{R_2}{r_2} + \cdots + Q_{wn} \ln \frac{R_n}{r_n} \right) \quad (3\text{-}75)$$

or

$$H - h = \frac{1}{2\pi kD} \sum_{i=1}^{i=n} Q_{wi} \ln \frac{R_i}{r_i} \quad (3\text{-}75a)$$

where Q_{wi} = the discharge from ith well
R_i = radius of influence for ith well
r_i = distance from ith well to point at which drawdown is computed
n = number of wells in group

The general equation from which drawdown at any point caused by pumping a group of *gravity* wells is as follows:

$$H^2 - h^2 = \frac{1}{\pi k} \sum_{i=1}^{i=n} Q_{wi} \ln \frac{R_i}{r_i} \qquad (3\text{-}76)$$

where the notations are the same as those used in Eqs. (3-75) and (3-75*a*).

A comparison between Eqs. (3-75*a*) and (3-76) indicates that the factor $\Sigma Q_{wi} \ln (R_i/r_i)$ is common to systems of both *artesian* and *gravity* wells. In general, these two equations can be expressed as follows:

Artesian case: $$H - h = \frac{F}{2\pi kD} \qquad (3\text{-}77)$$

Gravity case: $$H^2 - h^2 = \frac{F}{\pi k} \qquad (3\text{-}78)$$

where F is a factor which depends upon the flow from and position of each well in the system and which is independent of the condition of flow – *artesian* or *gravity*. The notational system in Eqs. (3-77) and (3-78) is used extensively in this chapter to avoid repetition.

The head h_{wj} at any well, for example, well j, in a system of n wells can be determined from the equations for *artesian* and *gravity* flow:

Artesian case: $$H - h_{wj} = \frac{1}{2\pi kD} \left(Q_{wj} \ln \frac{R_j}{r_{wj}} + \sum_{i=1}^{i=n-1} Q_{wi} \ln \frac{R_i}{r_{i,j}} \right) \qquad (3\text{-}79)$$

Gravity case: $$H^2 - h_{wj}^2 = \frac{1}{\pi k} \left(Q_{wj} \ln \frac{R_j}{r_{wj}} + \sum_{i=1}^{i=n-1} Q_{wi} \ln \frac{R_i}{r_{i,j}} \right) \qquad (3\text{-}80)$$

where Q_{wj} = the flow from well j
R_j = radius of influence of well j
r_{wj} = effective well radius of well j
$r_{i,j}$ = distance from each well to well j

and other symbols are as defined previously. It can be seen that the above two equations can be written as follows:

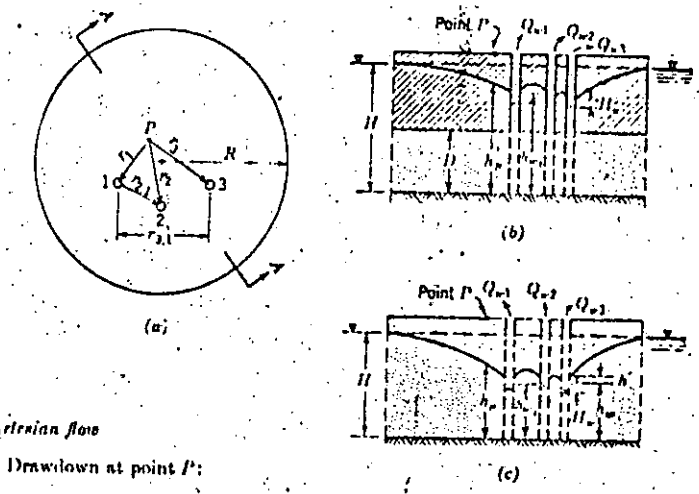
Artesian case: $$H - h_{wj} = \frac{F_w}{2\pi kD} \qquad (3\text{-}79a)$$

Gravity case: $$H^2 - h_{wj}^2 = \frac{F_w}{\pi k} \qquad (3\text{-}80a)$$

where F_w is a factor for drawdown at a well.

In problems where there is no unusual boundary, such as a line source of seepage or discontinuous pervious stratum, the radius of influence for all wells can be assumed constant, as shown in Fig. 3-34, where R_i is assumed equal to R. For this case Eqs. (3-75*a*) and (3-76) for head at any point can be written

Fig. 3-34. Drawdown for group of fully penetrating wells with seepage from a circular source. (*a*) Plan; (*b*) section A-A, artesian case; (*c*) section 1-1, gravity case.



Artesian flow

Drawdown at point P:

$$H - h_p = \frac{1}{2\pi kD} \left(Q_{w1} \ln \frac{R}{r_1} + Q_{w2} \ln \frac{R}{r_2} + Q_{w3} \ln \frac{R}{r_3} \right)$$

or $$H - h_p = \frac{1}{2\pi kD} \sum_{i=1}^{i=n} Q_{wi} \ln \frac{R}{r_i} = \frac{F_p}{2\pi kD}$$

Drawdown at well 1:

$$H - h_{w1} = \frac{1}{2\pi kD} \left(Q_{w1} \ln \frac{R}{r_{w1}} + \sum_{i=2}^{i=3} Q_{wi} \ln \frac{R}{r_{i,1}} \right)$$

or $$H - h_{w1} = \frac{F_{w1}}{2\pi kD}$$

Gravity flow

Head h_p at point P:

$$H^2 - h_p^2 = \frac{F_p}{\pi k}$$

Head h_{w1} at well 1:

$$H^2 - h_{w1}^2 = \frac{F_{w1}}{\pi k}$$

$$H - h = \frac{1}{2\pi kD} \sum_{i=1}^{i=n} Q_{wi} \ln \frac{R}{r_i} \tag{3-75b}$$

and

$$H^2 - h^2 = \frac{1}{\pi k} \sum_{i=1}^{i=n} Q_{wi} \ln \frac{R}{r_i} \tag{3-76a}$$

Likewise the factor F_w in Eqs. (3-79a) and (3-80a) would become equal to

$$F_w = Q_{wj} \ln \frac{R}{r_{wj}} + \sum_{i=1}^{i=n-1} Q_{wi} \ln \frac{R}{r_{ij}} \tag{3-81}$$

Comparison of Eqs. (3-17) and (3-79) reveals an important principle. The drawdown at a given well in a system of artesian wells is equal to that resulting from pumping this well as if no other wells were present plus the drawdown caused at the well due to pumping the remaining wells. Therefore drawdowns at a given point, caused by pumping individual wells, can be added to obtain the total drawdown caused at this point due to all wells. Similarly, a comparison of Eqs. (3-57) and (3-80) indicates that the term $(H^2 - h_{wj}^2)$ at a gravity well is equal to that caused by pumping that well plus the sum of the terms $(H^2 - h^2)$ caused at the well due to pumping the other gravity wells in the system.

Group of Wells Line Source. Where the flow to a group of *artesian* wells originates from a line source as shown in Fig. 3-35, the general equation for drawdown $(H - h_p)$ at any point P as determined from image wells is as follows:

$$H - h_p = \frac{1}{2\pi kD} \sum_{i=1}^{i=n} Q_{wi} \ln \frac{S_i}{r_i} = \frac{F'_p}{2\pi kD} \tag{3-82}$$

where S_i = distance from point P to image well i
r_i = distance from point P to actual well i

The drawdown at a given well, say, well j $(i = j)$, is expressed as follows:

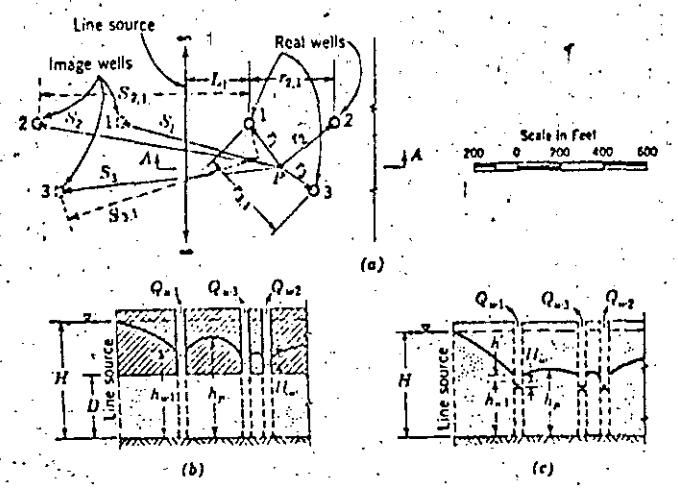


FIG. 3-35. Drawdown for group of fully penetrating wells with seepage from a line source. (*a*) Plan; (*b*) section *A-A*, artesian flow; (*c*) section *A-A*, gravity flow; (*d*) sample computation for F'_p and F'_{w1} for plan in (*a*).

(*b*) *Artesian flow*

Drawdown at point P:

$$H - h_p = \frac{1}{2\pi kD} \sum_{i=1}^{i=n} Q_{wi} \ln \frac{S_i}{r_i} = \frac{F'_p}{2\pi kD}$$

Drawdown at well 1:

$$H - h_{w1} = \frac{1}{2\pi kD} \left(Q_{w1} \ln \frac{2L_1}{r_{w1}} + \sum_{i=2}^{i=n} Q_{wi} \ln \frac{S_{i1}}{r_{i1}} \right)$$

or

$$H - h_{w1} = \frac{F'_{w1}}{2\pi kD}$$

(*c*) *Gravity flow*

Drawdown at point P:

$$H^2 - h_p^2 = \frac{F'_p}{\pi k}$$

Head h_{w1} at well 1:

$$H^2 - h_{w1}^2 = \frac{F'_{w1}}{\pi k}$$

(*d*)

Assume $r_{w1} = 1$ ft; Q_{w1}, Q_{w2}, and Q_{w3} = 40, 35, and 50 cfm, respectively; and $n = 3$. Compute F'_p and F'_{w1} for plan of wells in (*a*).

i	Q_{wi} cfm	r_i, ft	S_i, ft	$Q_{wi} \ln \frac{S_i}{r_i}$	r_{i1}, ft	S_{i1} or $2L_1$, ft	$Q_{wi} \ln \frac{2L_1}{r_{w1}}$ or $Q_{wi} \ln (S_i/r_{i1})$
1	40	250	780	45.6	1*	600	256.0
2	35	320	1,170	45.4	400	1,000	32.1
3	50	190	1,060	86.0	425	950	40.2
				$F'_p = 177.0$			$F'_{w1} = 328.3$

* $r_{w1} = 1$ ft.

$$H - h_{wj} = \frac{1}{2\pi kD}\left(Q_{wj}\ln\frac{2l_{ij}}{r_{wj}} + \sum_{i=1}^{i=n-1} Q_{wi}\ln\frac{S_{ij}}{r_{ij}}\right) \qquad (3\text{-}83)$$

or

$$H - h_{wj} = \frac{F'_w}{2\pi kD} \qquad (3\text{-}84)$$

where $2l_{ij}$ = distance from real well j to image well j
r_{wj} = effective radius of well j
S_{ij} = distance from real well j to image well i
r_{ij} = distance from real well j to real well i

and all other symbols are as defined previously. A comparison between Eqs. (3-71) and (3-83) again indicates that the drawdown at a well in a group equals that due to pumping the given well plus the drawdown caused at the well due to pumping the remaining wells.

Where the flow from the source to the wells is under *gravity* rather than artesian conditions, Eqs. (3-82) and (3-84) are written as follows:

$$H^2 - h_p^2 = \frac{F'}{\pi k} \qquad (3\text{-}85)$$

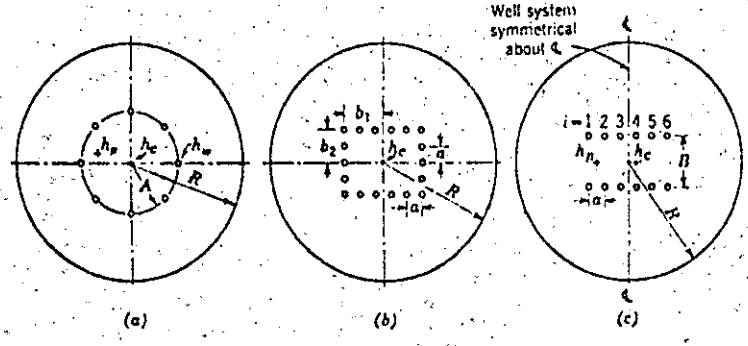
and

$$H^2 - h_{wj}^2 = \frac{F'_w}{\pi k} \qquad (3\text{-}86)$$

where the values of F' and F'_w are the same as those, respectively, in Eqs. (3-82) and (3-84).

Drawdown Factors for Wells in Special Arrays. Where the wells in a system are spaced in a linear, circular, or other orderly array, it is expedient to use either design curves or simple formulas for determining the drawdown factors F_w, F'_w, F, and F'. Curves and formulas for some of the arrays that are used frequently are shown in Figs. 3-36 to 3-38.

FIG. 3-36. Drawdown factors for systems of multiple wells with seepage from a circular source. (*a*) Circular array of equally spaced wells; (*b*) rectangular array of equally spaced wells; (*c*) two parallel lines of equally spaced wells.



(*a*)

$$F_w = Q_w \ln\frac{R^n}{nr_w A^{(n-1)}} \qquad (3\text{-}87)$$

$$F_r = nQ_w \ln\frac{R}{A} \qquad (3\text{-}88)$$

For *artesian* flow:

$$h_p = H - \frac{(H - h_w)\left(n\ln R - \sum_{i=1}^{i=n}\ln r_i\right)}{\ln\frac{R^n}{nr_w A^{(n-1)}}} \qquad (3\text{-}75c)$$

$$h_r = H - \frac{(H - h_w)n\ln(R/A)}{\ln\frac{R^n}{nr_w A^{(n-1)}}} \qquad (3\text{-}75d)$$

For *gravity* flow:

$$H - h_r = H - \sqrt{H^2 - \frac{n(H^2 - h_w^2)\ln(R/A)}{\ln\frac{R^n}{nr_w A^{(n-1)}}}} \qquad (3\text{-}76b)$$

(*b*)

Approximate Method:

Compute equivalent radius A_r of well system from following equation:

$$A_r = \sqrt[n]{b_1 b_2} \qquad (3\text{-}89)$$

Compute F_w and F_r from Eqs. (3-87) and (3-88) using A_r instead of A.

Exact Method:

Compute F_w and F_r from Eqs. (3-81) and (3-75*b*), respectively.

(*c*)

$$F_r = 4Q_w \sum_{i=1}^{i=n/4} \ln\frac{R}{\frac{1}{2}\sqrt{n^2(2i-1)^2 + B^2}} \qquad (3\text{-}90)$$

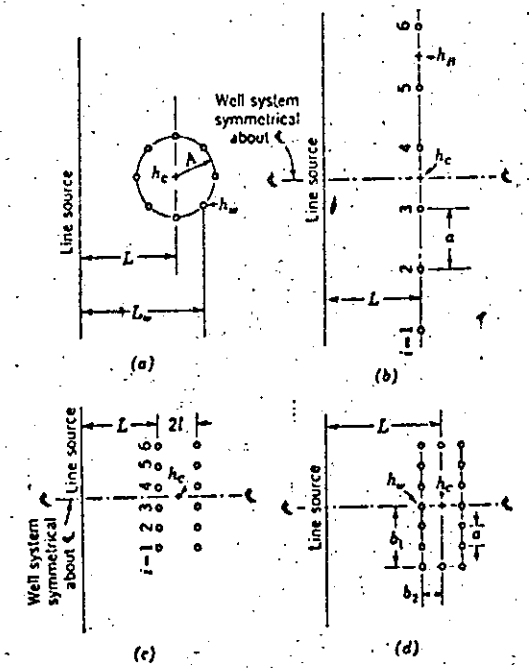
$$F_w = 2Q_w \sum_{i=1}^{i=n/2} \ln\frac{R}{\frac{1}{2}\sqrt{n^2(2i-3)^2 + B^2}} \qquad (3\text{-}91)$$

(*a*), (*b*), and (*c*)

For *artesian* flow:		For *gravity* flow:	
$H - h_r = \frac{F_r}{2\pi kD}$	(3-77)	$H^2 - h_r^2 = \frac{F_r}{\pi k}$	(3-78)
$H - h_w = \frac{F_w}{2\pi kD}$	(3-79*a*)	$H^2 - h_w^2 = \frac{F_w}{\pi k}$	(3-83*a*)

Note: n = number of wells in system
i = well number as shown on Fig. 3-36*c*
Q_w = flow per well
Flows from all wells in system assumed equal. Wells assumed to fully penetrate pervious stratum.

Fig. 3-37. Drawdown factors for systems of multiple wells with seepage from a line source. (*a*) Circular array of equally spaced wells; (*b*) single line of equally spaced wells; (*c*) two parallel lines of equally spaced wells; (*d*) rectangular array of equally spaced wells.



(*a*)

$$F'_s = \frac{Q_w}{2} \sum_{i=1}^{i=n} \ln \left[1 + 4\left(\frac{L}{\lambda}\right)^2 - 4\left(\frac{L}{\lambda}\right) \cos (i-1)\frac{2\pi}{n} \right] \tag{3-92}$$

where

$$\frac{L}{\lambda} \geq 2 \qquad F'_s = Q_w n \ln \frac{2L}{\lambda} \tag{3-92a}$$

$$F'_w = Q_w \left(n \ln \frac{2L_e}{\lambda} + \ln \frac{\Lambda}{nr_w} \right) \tag{3-93}$$

(*b*)

At center of well system,

$$F'_s = 2Q_w \sum_{i=1}^{i=n/2} \ln \sqrt{1 + \left[\frac{2L}{(a/2)(n+1-2i)} \right]^2} \tag{3-94}$$

Midway between the last two wells,

$$F'_s = Q_w \sum_{i=1}^{i=n} \ln \sqrt{1 + \left[\frac{2L}{(a/2)(2i-3)} \right]^2} \tag{3-95}$$

Where $n = \infty$, use graphs on Fig. 3-38.

(*c*)

At center of well system,

$$F'_s = 2Q_w \sum_{i=1}^{i=n/4} \left\{ \ln \sqrt{1 + \left[\frac{2L+l}{(a/4)(n+2-4i)} \right]^2} + \ln \sqrt{1 + \left[\frac{2L+3l}{(a/4)(n+2-4i)} \right]^2} \right\} \tag{3-96}$$

(*d*)

Approximate Method. Compute equivalent radius A_e of well system from Eq. (3-89) on Fig. 3-36, and then compute F'_s and F'_w from Eq. (3-92) or (3-92*a*) and Eq. (3-93), respectively, using A_e instead of A.

Exact Method. Compute F'_s and F'_w from Eqs. (3-82) and (3-83), respectively.

(*a*), (*b*), (*c*), and (*d*)

For *artesian* flow:

$$H - h = \frac{F''}{2\pi kl} \tag{3-77}$$

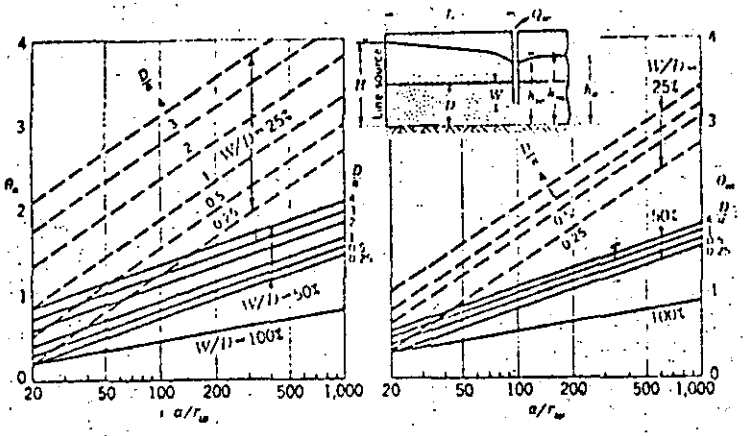
For *gravity* flow:

$$H^2 - h^2 = \frac{F''}{\pi k} \tag{3-78}$$

Note: Q_w = flow per well
n = number of wells in system
i = well number as shown

Flows from all wells in system assumed equal. Wells assumed to fully penetrate pervious stratum.

FIG. 3-38. Factors for computing flow to and head reduction from an infinite line of artesian wells with a line source of seepage.



$$Q_w = \frac{kD(H - h_w)}{L/a + \theta_s} \tag{3-97}$$

$$h_m = h_w + \frac{\theta_m(H - h_w)}{L/a + \theta_s} \tag{3-98}$$

$$h_s = h_w + \frac{\theta_s(H - h_w)}{L/a + \theta_s} \tag{3-99}$$

where r_w = well radius
a = well spacing

Note: Above plots from data in Relief Well Design, *Civil Works Engineer Bulletin* 55-11, Department of the Army, Office Chief of Engineers, June 28, 1955.

Curves and equations pertinent to well systems supplied from a line source of seepage were developed by the image-well method. Where the source of seepage is circular in plan, Eqs. (3-75*a*) and (3-76) were used as the basis.

3-24. Source of Seepage

The equations presented above for single wells and multiple-well systems were based on either an assumed radius of influence R for a circular source of seepage or a distance L from the well system to one (or possibly two) line source(s) of seepage. Since the values of R or L used in the equations affect the computed discharges and drawdowns, it is necessary to estimate the proper values to be used. Procedures therefor are as follows.

In the development of the previous equations for flow from a circular source to a single well, it was assumed for simplicity that the well was located in the center of a circular island. However, usually the well or group of wells will be installed in a general land area and not on a circular island. If the wells are not close to a river or canal, the well flow and piezometric head in the vicinity of the well can be computed from the equations given above for a circular source by considering that the term R is the radius of influence of the well instead of the radius of the island. The *radius of influence* is defined as the radius of the circle beyond which the well has no significant influence on the original ground-water level or piezometric surface. The value of R depends upon geologic and foundation conditions at a site, duration of pumping, properties of the pervious stratum, and amount of drawdown.

From the equations for well flow developed for gravity or artesian wells, it can be shown that the flow Q is not especially sensitive to the value of R, because the radius of influence R will normally be large compared with the radius of the well r_w. For example, assume that from a pumping test on an artesian well with radius $r_w = 1$ ft, it was found that $R = 2{,}000$ ft. Had R been 1,000 ft instead of 2,000 ft, the flow required to produce the same drawdown $(H - h_w)$ at the well would have been only about 10 per cent greater. Thus, when computing the flow from a well for a given drawdown, using formulas for a circular source, reasonably correct flows will be obtained if the estimated value of R is only approximate, provided the actual radius of influence is large compared with the radius of the well.

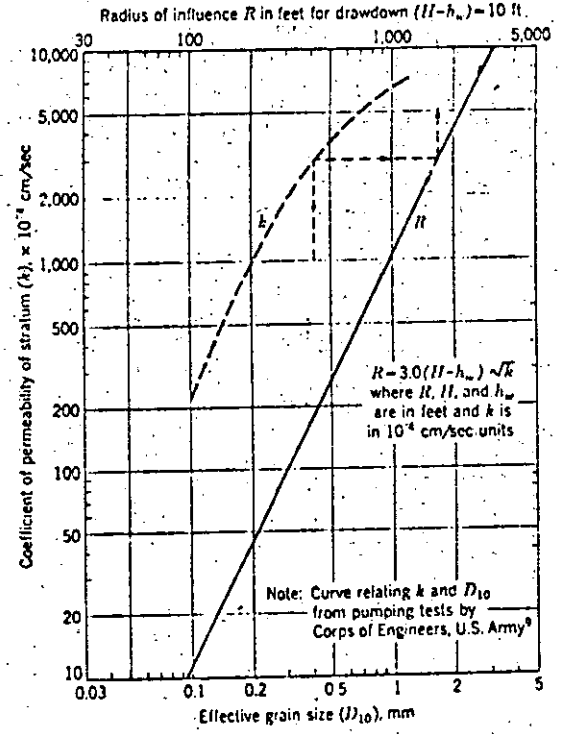
A value of R can be estimated from the following empirical equation proposed by Sichardt (28):

$$R = C'(H - h_w)\sqrt{k} \tag{3-100}$$

where R, H, and h_w are in feet, k is in 10^{-4} cm per sec units, and C' is a dimensionless constant for which a value of 3.0 was proposed by Sichardt for gravity wells. Sichardt's equation gives a fair approximation for values of R observed in pumping tests on artesian wells in the lower Mississippi River Valley by the U.S. Army Corps of Engineers. The Moretrench Corporation has found the above equation applicable for estimating the radius of influence for a single line of wellpoints, but has found C' to range from about 1.5 to 2.0. Since the permeability k is related to grain size of the soil, R can be related to grain size. Figure 3-39 shows the relationship between k measured *in situ* and effective grain size D_{15} obtained from special pumping tests made in the Mississippi River Valley by the Corps of Engineers, U.S. Army (9). The values of R for a drawdown of 10 ft computed from Eq. (3-100) with $C' = 3.0$ also are plotted on this figure versus k. For drawdowns other than 10 ft, the

value of R from Fig. 3-39 should be multiplied by the ratio of the actual to the 10-ft drawdown to obtain the value of R corresponding to the actual drawdown for $C' = 3.0$.

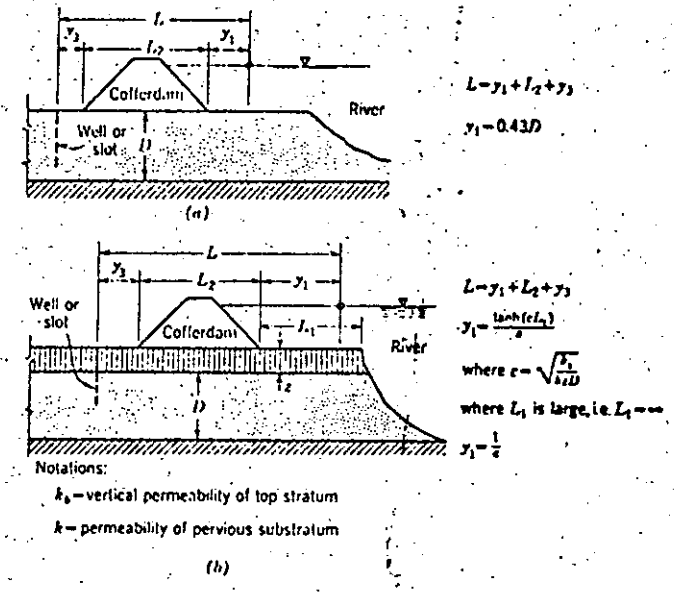
FIG. 3-39. *In situ* horizontal permeability versus grain size, and radius of influence versus permeability.



Where a well or group of wells is close to a river or shore line exposing the pervious strata, the source of seepage generally can be considered as at the river, provided the distance L from the well or wells to the river is less than $R/2$ and provided no seepage enters the pervious stratum between the wells and river. However, where cofferdams are placed around the construction area and the river rises against the cofferdam, seepage may enter the pervious stratum through the ground surface area between the cofferdam and river bank. In such cases the distance L from the well system to the effective line source of seepage will be less than the distance from the wells to the river bank. Typical cases with

corresponding equations for L are shown in Fig. 3-40. If no leakage occurs through the riverside blanket but the most landward wells in the group are at distances greater than $R/2$ from the river and the most riverward wells are closer to the river than $R/2$, the source of seepage can be treated as a circle, using a radius somewhat less than the value of R. If the bank or bed of a river located close to a group of wells is covered with mud, the river may not act as a source of seepage.

FIG. 3-40. Effect of vertical seepage riverside of cofferdam on point of effective seepage entrance. (*a*) No top stratum present; (*b*) semipervious top stratum extends from river to cofferdam.



Where a line (or two parallel lines) of wells is (are) installed in an area not close to a river, the seepage can be considered as originating from a line source on each side of the wells. If wells are to be located in an area containing pervious strata with irregular vertical discontinuities such as those shown in Fig. 3-26, it is generally desirable to draw a flow net and design the well system using Eqs. (3-32) to (3-45) rather than design the system from equations for a circular or line source with estimated values of R or L.

As the flow to a dewatering system near a line source of seepage is inversely proportional to the distance L to the source, the value of L should be determined as accurately as practicable.

DESIGN OF WELLPOINTS AND DEEP WELLS

Wells and wellpoints should be of a type and design that will offer little resistance to water flowing through the screen and riser pipe, prevent infiltration of sand during pumping, and resist corrosion by water and soil.

Most commercial wellpoints are made with either brass or stainless-steel screens mounted over galvanized, tin-dipped, or stainless-steel suction pipe. Where large flows are anticipated, a high-capacity type of wellpoint should be selected. In order to prevent infiltration of sand, the mesh or slot size of a screen should be smaller than the 80 per cent size (D_{80}) or 70 per cent size (D_{70}), respectively, of the sand in which the wellpoint will be installed. Where silty soils are to be drained, the wellpoint should be provided with a graded medium-to-coarse sand filter designed in accordance with filter criteria subsequently set forth. The inner suction pipe of self-jetting wellpoints should be of a size and have openings to permit inflow of water with minimum hydraulic head loss. Self-jetting wellpoints should also be designed so that most of the jet water will go out the tip of the point, with some backflow to keep the screen flushed clean while jetting the point in place. One and one-half- or two-inch steel pipe generally is used for riser pipe for wellpoints, depending upon the length required and anticipated flow.

Screens commonly used for deep, large-diameter wells are slotted steel or wood stave pipe or perforated steel pipe wrapped with galvanized, trapezoidal-shaped wire. Slotted steel or wooden screens are commercially available with $\frac{1}{8}$-, $\frac{3}{16}$-, or $\frac{1}{4}$-in. wide slots. It is usually necessary and desirable to use a properly graded filter gravel around such screens to prevent infiltration of the sand being drained and to improve the efficiency of the well. The riser pipe usually consists of either steel or wood.

The following criteria should be observed in designing or selecting well screens or wellpoints:

Slot width $\leqq D_{70}$ (filter or aquifer sand)
Hole diameter or width $\leqq D_{80}$ (filter or aquifer sand)

A filter must meet the following criteria to prevent infiltration of the sand being drained and ensure the filter being much more permeable than the aquifer sand:

$$\frac{D_{15}\text{ (filter)}}{D_{85}\text{ (aquifer sand)}} \leqq 5.0$$

$$\frac{D_{15}\text{ (filter)}}{D_{15}\text{ (aquifer sand)}} \geqq 4.0$$

3-25. Well Penetration

In a stratified aquifer, the effective well penetration usually differs from that computed from the ratio of the length of well screen to total thickness of the aquifer. To determine the required length of well screen W to achieve an effective screen penetration W' in a stratified aquifer, the following procedure can be used. Each stratum of the pervious substratum with thickness d' and horizontal and vertical permeability coefficients k_h and k_v is first transformed into an isotropic layer of thickness d, where

$$d = d'\sqrt{\frac{k_h}{k_v}}$$

The transformed coefficient of permeability of each stratum is

$$k = \sqrt{k_h k_v}$$

The thickness of the transformed, homogeneous, isotropic aquifer is

$$D = \sqrt{\sum (d'k_h)} \cdot \sum (d'/k_v)$$

and the effective permeability of the transformed aquifer is

$$k = \sqrt{\frac{\sum (d'k_h)}{\sum (d'/k_v)}}$$

The effective well-screen penetration W into the transformed aquifer is

$$W = \frac{\sum_0^n d'k_h}{k}$$

The penetration of the well screen in the transformed aquifer (expressed as a decimal) is

$$\frac{W'}{D} = \frac{\sum_0^n d'k_h}{kD} = \frac{\sum_0^n d'k_h}{\sum_0^b d'k_h}$$

where D is the actual total thickness of the stratified pervious aquifer.

3-26. Hydraulic Head Loss in Wellpoints and Wells

As ground water flows into and within a well, hydraulic head losses occur in the well as a result of such flow. The head losses consist of screen entrance loss H_e; friction loss due to flow up through the well screen, H_s; friction loss due to flow in the riser pipe, H_r; and velocity head

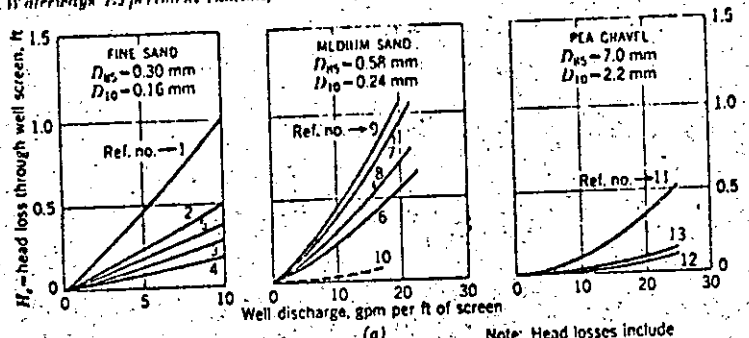
loss H_s. The total head loss in the well, H_w, is the sum of these individual head losses, and therefore

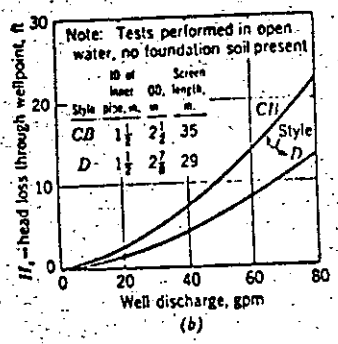
$$H_w = H_e + H_f + H_v + H_s \qquad (3\text{-}101)$$

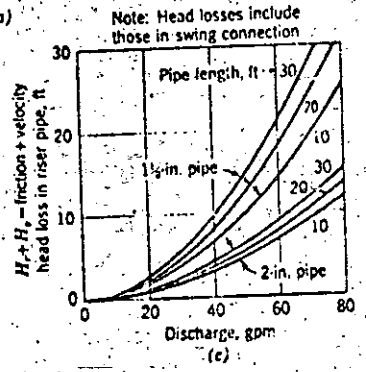
Procedures for estimating these head losses are given below.

Entrance head loss H_e is that required to force the flow through the gravel filter of the well, if the well has a filter, and the well screen. This loss depends on the flow per unit length of well screen, the type of sand adjacent to the screen, permeability of gravel filter, the number, size, and type openings in the well screen, and the diameter of the screen. Entrance head losses for various types of wellpoints installed in fine and medium sand and pea gravel determined from laboratory tests are plotted for various well flows in Fig. 3-11. Entrance head losses for two types of self-jetting wellpoints obtained from tests by the Moretrench Corporation also are shown in Fig. 3-11. Similar data for large-diameter wood-stave well screens with graded gravel filters obtained from labora-

Fig. 3-11. Entrance head loss for typical wellpoints, and velocity and friction head loss in wellpoint riser pipes. (*a*) 2½-in.-ID plain-tip wellpoints; (*b*) self-jetting wellpoints; (*c*) head losses in wellpoint riser pipes, $C = 110$. [(*a*) *U.S. Army Engineer Waterways Experiment Station, Vicksburg, Miss.* (*b*) *Moretrench Corp.*]







Ref.	Wellpoint	Slot or mesh	
		No.	Opening in mm
1	A and B, grooved slot	12	0.30
2	C, wire wrap on perforated pipe	20	0.51
3	D, wire mesh on perforated pipe	28	0.59
4	E, wire mesh on perforated pipe	28	0.59
5	B, grooved slot	25	0.63
6	A, grooved slot	25	0.63
7	D, wire mesh on perforated pipe	28	0.59
8	A, grooved slot	50	1.27
9	B, grooved slot	30	0.76
10	F, perforated pipe with 6-in. pea-gravel filter	$\frac{5}{32}$ in.	3.97
11	A, grooved slot	12	0.30
12	A, grooved slot	100	2.54
13	E, perforated pipe	$\frac{5}{32}$ in.	3.97
CB	Moretrench, commercial bronze, self-jetting, mesh SF	40 × 45	0.31 × 0.38
D	Moretrench, stainless steel style D, self-jetting, mesh E	12 × 68	0.30 × 1.73

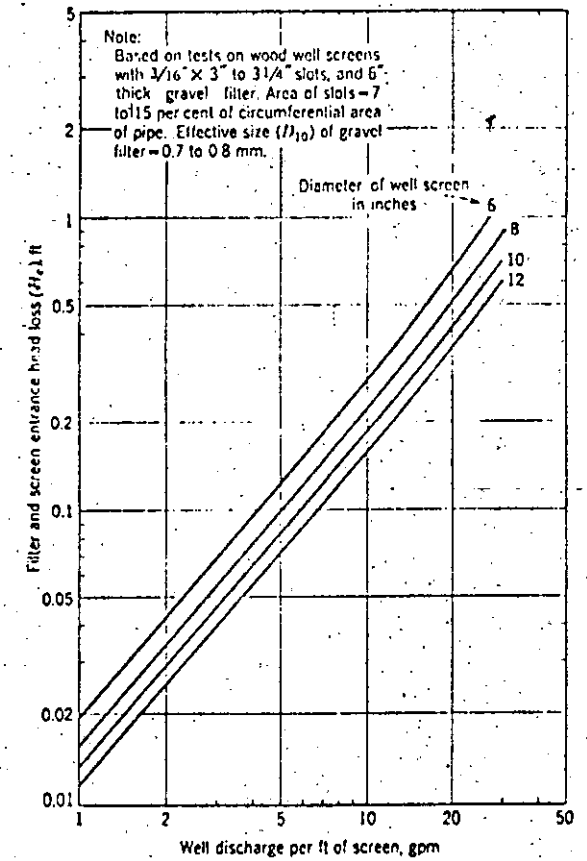
tory tests and pumping tests on wells installed in pervious sand strata in the alluvial valley of the lower Mississippi River are shown in Fig. 3-12. Data for several other types of well screens are presented by Petersen, Rohwer, and Albertson (29). When estimating the entrance head loss, a weighted average flow per foot of screen should be used.

Friction loss in well screens can be estimated from the screen length and distribution of flow in the screen from hydraulic formulas for flow of water through pipes. For example, if a well fully penetrates a uniform pervious artesian stratum, the flow in the screen will increase approximately linearly from zero at the bottom to the full well flow Q_w at the top of the screen. In this case the friction loss can be computed assuming the entire well flow Q_w to be flowing through one-third the length of the well screen. If the pervious stratum grades coarser in grain size with depth or if the well only partially penetrates a uniform stratum, the flow in the lower portion of the screen will be greater than that assumed above, in which case the head loss can be estimated assuming a flow of $0.9Q_w$ through one-half of the length of well screen (30).

The friction head loss H_f in the riser pipe can be computed from the well flow Q_w and the length of riser pipe through which flow takes place using ordinary hydraulic formulas. Head losses in the connecting pipe between the well or wellpoint and collector pipe also should be included. For estimating loss in standard steel swing connections for wellpoints (including that in the cock valve in the swing), the swing can be replaced by the following equivalent lengths of riser pipe:

ID of suing connection, in.	*Equivalent length, ft of pipe with same ID as suing*
1½	30
2	56

FIG. 3-42. Entrance head loss for wood well screens.

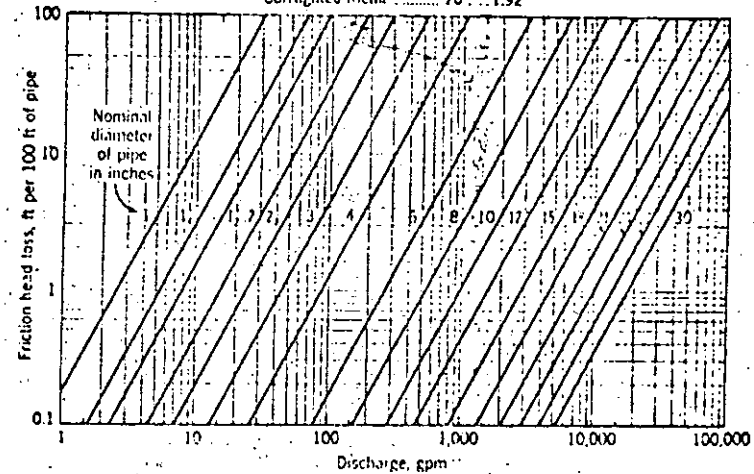


The velocity head loss H_v is equal to $v^2/2g$, where v is the velocity of flow in the riser pipe and g (the acceleration of gravity) = 32.2 ft per sec^2.

Curves for estimating hydraulic head losses in wells and wellpoints are shown in Figs. 3-41 to 3-44.

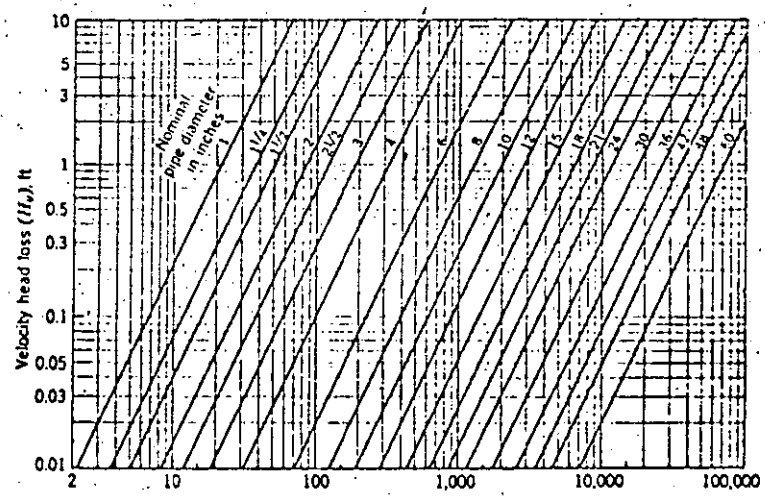
FIG. 3-43. Friction head loss in riser pipes and well screens.

Type of pipe	C	$\left(\frac{100}{C}\right)^{1.85}$
Steel (new)	125	0.67
Steel (avg. condition)	110	0.83
Wood stave riser	130	0.71
Wood stave screen	100	1.00
Corrugated metal	70	1.92



Based on Hazen Williams equation with $C = 100$
Multiply losses by $(100/C)^{1.85}$ for values of C other than 100

FIG. 3-44. Velocity head loss in wells and pipes.



3-27. Effective Well Radius

The effective well radius r_w for a well or wellpoint installed without a gravel or sand filter can be taken as half the outside diameter of the well screen. Where a gravel or sand filter has been placed around the well screen, one-half the outside diameter of the filter can be used for r_w. Where a well screen has been installed without a filter in a pervious stratum but a natural filter around the screen has been developed by surging, the effective well radius will exceed half of the outside diameter of the well screen. However, since the extent of the developed filter is indefinite, generally it will be satisfactory to use a value for r_w corresponding to the outside diameter of the well screen, since conservative designs will result if this is done.

DESIGN OF DEWATERING AND PRESSURE-RELIEF SYSTEMS

A complete dewatering system should be capable of lowering the ground-water table as required, intercepting seepage into the excavation, and removing any surface water that would affect operation of the dewatering system or construction. The type of system selected and its design will depend upon geological, soil, and water-table conditions at the site, the size of excavation, surface-water runoff, and construction requirements. The design of such systems requires a knowledge and careful evaluation of the above factors, a knowledge of seepage and ground-water flow to wells or wellpoints, an understanding of wells, wellpoints, and pumping equipment, and a considerable amount of experience and judgment. The design of dewatering and pressure-relief systems requires (1) computation of the rate of flow required to lower the water table or artesian head; (2) selection of the required spacing, depth, size, and type of wellpoints or wells; (3) selection of collector pipes, pumps, and discharge systems; and (4) selection of pumps, sumps, dikes, etc., for control of surface water.

3-28. Dewatering Slopes and Excavations

Ground water adjacent to excavations can be lowered or controlled by one or a combination of the methods previously described in this chapter. Principles to be considered in the design of certain types of dewatering systems together with numerical examples of designs are given in the following paragraphs.

Wellpoint Systems. Systems of closely spaced wellpoints connected to a header pipe pumped by a wellpoint pump can be used for ground-water lowering where the required lowering is not excessive. Since wellpoints usually are closely spaced, they can be considered as creating a continuous drainage slot and then their spacing determined so that the head along the line of wellpoints is essentially the same as would exist at a slot. First, the head reduction $(H - h_s)$ at a slot required to produce the desired residual head h_D is computed from equations for gravity or artesian-gravity flow on pages 265–275. The wellpoint spacing can then be computed from Eq. (3-15*a*) by assuming in this equation that $h_s = h_D$ and that the head difference $(h_p - h_s)$ is very small. For design purposes it is suggested that $h_s - h_w$ be assumed equal to about $0.001H$. After the wellpoint spacing and the head h_s at a wellpoint have been computed, the flow Q_w per wellpoint can be computed from the equations on pages 282 and 284. The above value of h_s must be equal to or greater than the value of h_s computed from the following equation:

$$h_s = M - V + H_f + H_w \qquad (3\text{-}102)$$

where M = distance from base of pervious substratum to pump intake, ft
V = vacuum at pump intake, ft of water
H_f = average head loss in header pipe up to pump intake, ft
H_w = head loss in wellpoint computed from Eq. (3-101) plus that in swing connecting wellpoint riser to header pipe, ft

The top of the wellpoint screen should be set slightly below $(h_s - H_w)$ to ensure that the wellpoint is submerged; otherwise excessive air may enter the dewatering system and reduce its efficiency.

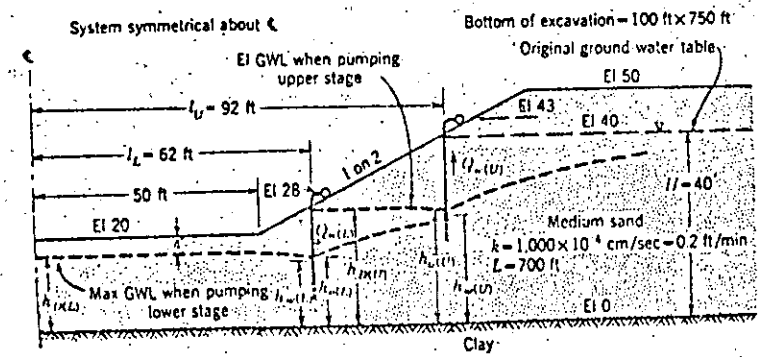
Since wellpoints are generally connected to a collector pipe pumped by a centrifugal pump, a single stage of them can lower the ground water only about 15 to 20 ft. The drawdown attainable per stage is limited by the vacuum that can be developed by the pump, the height of the pump above the initial ground-water level, and hydraulic head losses in the wellpoint and collector systems. For proper efficiency the hydraulic head losses in the system must be small; this requires that the header pipe, pumps, wellpoints, and riser pipes be of adequate size for the flow to be pumped. Where two or more stages of wellpoints are required, it is customary to design each stage capable of producing the total drawdown required by that stage with none of the upper stages functioning. In such cases it may be necessary to maintain these upper stages intact so that they can be pumped during the process of backfilling the excavation. However, if the piezometric grade line cannot be confined within the slope by pumping the lowest stage of wellpoints, it may be necessary to pump one or more of the upper stages to obtain the required ground-water lowering. Where more than a single-stage system is required, it is advisable to observe the ground-water level immediately prior to and while pumping the upper stage and to measure the discharge from the wellpoint system. By comparing the actual discharge and ground-water lowering with that computed, the ade-

quacy of the lower stage can be checked to a degree prior to its installation. Testing and evaluating the performance of a dewatering system as it is installed and operated are advisable, since it frequently is not possible in design to determine accurately the permeability of the pervious strata, the radius of influence or distance to effective source of seepage, and head losses in the wellpoint system. The design of a typical two-stage wellpoint system for lowering the ground-water table below an excavation is illustrated below.

Assume that an excavation 30 ft deep and with a bottom dimension of 100 by 750 ft is to be made in a stratum of medium sand having a thickness of 50 ft, underlain by clay, and that $k = 1{,}000 \times 10^{-4}$ cm per sec, with a ground-water table 10 ft below the ground surface. Design a wellpoint system to lower the ground water 4 ft below the bottom of the excavation. These conditions are illustrated on Fig. 3-45, together with the computations necessary to determine the required number of stages and spacing of the wellpoints and the computed flow per wellpoint.

From Fig. 3-45 it is seen that two stages of wellpoints are required. The first stage, consisting of 2½-in. self-jetting wellpoints 29 in. long with 2-in. 20-ft-long risers on 5-ft centers, would be installed at elevation 43 (3 ft above the initial ground-water level) to lower the ground-water table about 15 ft or to elevation 25. Maintaining an average vacuum in the header pipe of about 20 ft in this stage would permit installation of the second stage 3 ft above the lowered ground water or at elevation 28. The second stage, consisting of similar self-jetting wellpoints attached to 2-in.-diameter riser pipes 20 ft long and on 3-ft centers, could lower the ground water to the desired elevation when the average vacuum in the header pipe is equal to or greater than 20 ft. Although computations on Fig. 3-45 indicate that shallower wellpoints could be used, the wellpoints

FIG. 3-45. Design of a wellpoint system for slope dewatering.



Problem. Determine spacing of 2½-in. self-jetting wellpoints with 2-in. riser pipes required to lower GWL 4 ft below bottom of excavation, assuming vacuum at pump (V) = 24 ft, head loss in collector (H_c) = 2 ft, and intake of pump 2 ft above collector. Assume style 1; Mesh E Moretrench wellpoints are to be used.

Solution. Compute spacing so that water level in wellpoints at a distance below collector equal to or less than available vacuum in collector = 20 ft. Two stages of wellpoints will be required. Assume each stage installed 3 ft above the ground water existing at the time of installation. Also assume $r_w = 0.12$ ft.

Upper Stage. Install upper stage at el 43, and 93 ft from ℄ of the excavation, to temporarily lower the ground water to el 25 or 15 ft to permit installation of the lower stage at el 28. Required $h_p = 25$ ft. Assume $h_w = h_p = 25$ ft or $H - h_p = 15$ ft for first trial.

$l/h_w = {}^{93}/_{25} = 3.7$; from Fig. 3-22 $C_1 = 1.02$. Since b can be assumed zero for wellpoints, $b/H = 0$, and from Fig. 3-22, $C_2 = 1.45$. Compute h_p from Eq. (3-29):

$$h_p = 25\left[1 + \frac{1.02(1.45)(15)}{700}\right] = 25.8 \text{ ft}$$

Since $h_p - h_w = 0.8$ ft, assume $h_w = 24.2$ ft and recompute h_p. Thus $l/h_w = {}^{93}/_{24.2} = 3.8$; $C_1 = 1.02$ from Fig. 3-22.

$$h_p = 24.2\left[1 + \frac{1.02(1.45)(15.8)}{700}\right] = 25.0 \text{ ft}$$

which agrees with the required value.

Thus $h_p = 25.0$ ft, $h_w = 24.2$ ft, and $H - h_w = 15.8$ ft. Assume $h_w - h_c = 0.001H = 0.04$ ft; thus $h_c = 24.16$ ft. Substitute h_c for h_w and h_w, L, H, and r_w in Eq. (3-45*a*):

$$\frac{24.2^2 - 24.16^2}{40^2 - 24.2^2} = \frac{a}{2\pi(700)} \ln \frac{a}{2\pi(0.12)}$$

Solving the above equation gives $a = 4.6$ ft. Use $a = 5$ ft for design. From Eq. (3-24) the flow per wellpoint for an infinite line of wellpoints is

$$Q_w = aQ_s = 5\left(0.73 + 0.27 \times \frac{15.8}{40}\right)\frac{0.20}{2 \times 700}(40^2 - 24.2^2) = 0.61 \text{ cfm} = 4.6 \text{ gpm}$$

From a plan flow net it can be shown that the average flow for the finite line of wellpoints will be about 35 per cent greater than that for an infinite line.

Thus $Q_w = 4.6$ gpm × 1.35 = 6.2 gpm per wellpoint.

For $Q_w = 6.2$ gpm, the hydraulic head losses are as follows:

$H_v = 0.08$ ft from Fig. 3-41*a*, assuming loss given by curve 7 is applicable
$H_e = 0.15$ ft from Fig. 3-41*b*
$H_f + H_v = 0.12$ ft from Fig. 3-41*c* which includes loss in swing
$H_w = 0.35$ ft

Thus $h_w - H_w = 24.16 - 0.35 = 23.8$ ft

From Eq. (3-102) the required effective vacuum $V - H_c$ at the wellpoint is 19.2 ft. Since this value is slightly less than the available 20 ft, a wellpoint spacing of 5 ft with header at el 43 and top of wellpoint screen at el 23 would be used.

Lower Stage. Assume $l = 62$ ft, and for a first trial $h_w = h_p = 16$ ft. $l/h_w = {}^{62}/_{16} = 3.9$; from Fig. 3-22, $C_1 = 1.02$ and $C_2 = 1.45$, since $b = 0$. From Eq. (3-29),

$$h_p = 16\left[1 + \frac{1.02(1.45)(40 - 16)}{700}\right] = 16.8 \text{ ft}$$

Since $h_p - h_w = 0.8$ ft, assume $h_w = 15.2$ ft and recompute h_p. $l/h_w = 63/15.2 = 4.1$, and $C_1 = 1.02$ from Fig. 3-22.

$$h_p = 15.2\left[1 + \frac{1.02(1.45)(40 - 15.2)}{700}\right] = 16.0 \text{ ft}$$

Thus $h_p = 16.0$ ft, $h_w = 15.2$ ft. Assume $h_w - h_w' = 0.001H = 0.04$ ft; $h_w' = 15.2 - 0.04 = 15.16$ ft. Substitute h_w' for h_p and h_w, H, l, and r_w in Eq. (3-15a):

$$\frac{15.2^2 - 15.16^2}{40^2 - 15.2^2} = \frac{a}{2\pi(700)} \ln \frac{a}{2\pi(0.12)}$$

Solving this equation gives $a = 3$ ft. From Eq. (3-24) the flow per wellpoint is

$$Q_w = aQ_s = 3\left(0.73 + 0.27 \times \frac{24.8}{40}\right)\frac{0.20}{2 \times 700}(40^2 - 15.2^2) = 0.53 \text{ cfm}$$

$$= 4.0 \text{ gpm} \quad \text{for an infinite line of wellpoints}$$

or $1.35 \times 4.0 = 5.4$ gpm for the finite line

For this discharge $H_w = 0.26$ ft; thus $h_w - H_w = 15.16 - 0.26 = 14.90$ ft. From Eq. (3-102) the required vacuum at the wellpoint is 13.1 ft. Therefore, if the vacuum in the header is maintained at 13 ft, wellpoints on 3-ft centers would lower the ground water to el 16. However, since the available vacuum in the header is 20 ft and since generally it is not practicable to operate the pump to reduce the vacuum, the wellpoints would be installed with 20-ft-long riser pipes on about 3-ft centers. This would prevent excess air from entering the system which might otherwise occur had 13-ft-long riser pipes been used. With such a system $Q_w = 6.2$ gpm, $H_w = 0.35$ ft, and $h_w - H_w = 8.1$ ft, and the ground water would be lowered to about el 9.

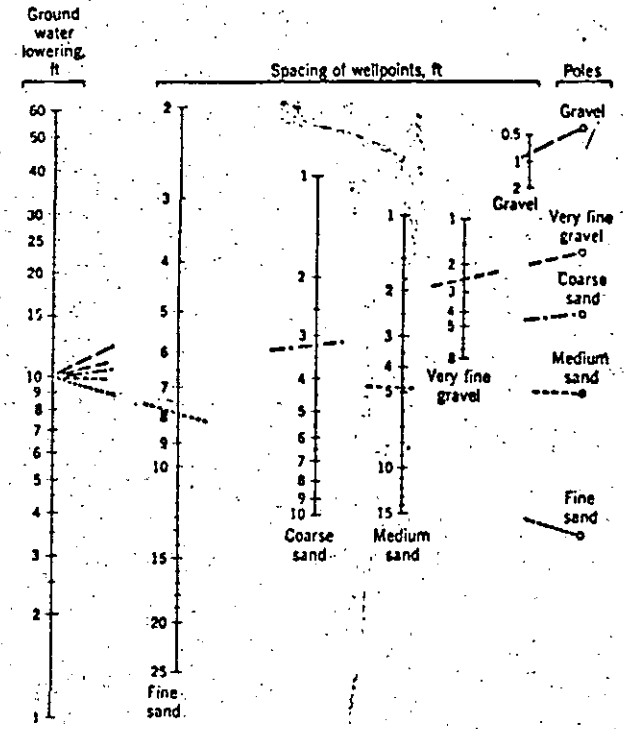
In the above case it would be advisable to observe ground-water levels before and during pumping of the upper stage and to measure the discharge. From these data the design of the lower stage could be adjusted if it is found that the observed flow per foot drawdown differs appreciably from the computed values. Such differences can occur because of limitations in accuracy of k, l, and H_w used in design.

would be installed with 20-ft risers to prevent possible reduction in efficiency of the stage due to excess air entering the system.

In lieu of the detailed computations shown in Fig. 3-15, the approximate spacing of wellpoints required to produce a given ground-water lowering in various soils can be estimated from the nomographs shown in Figs. 3-16 and 3-17. However, these nomographs should be used with caution since they are based on empirical data and are for average conditions. Nevertheless, they may be used as a guide in selecting the spacing of wellpoints.

In some cases, wellpoints alone cannot entirely eliminate seepage into an excavation and supplemental measures are necessary. For example, where a pervious stratum is underlain by rock and it is necessary to lower the ground-water level to the top of rock, some seepage will pass between

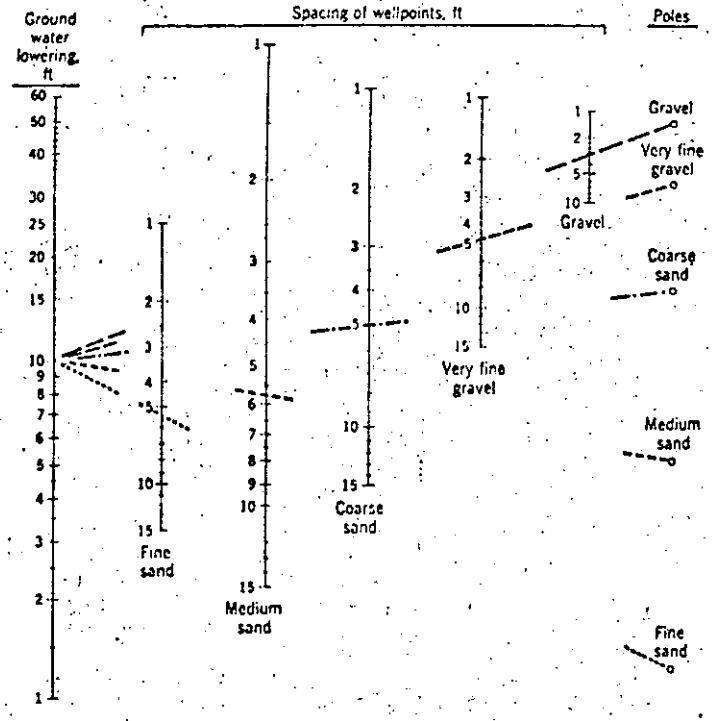
FIG. 3-16. Wellpoint spacing for uniform clean sands and gravels. (*Moretrench Corp.*)



the wellpoints even though they extend to the rock. In such cases it may be necessary to intercept this seepage with ditches or French drains and install automatic "mops" in the bottom of the ditches. In cases where the pervious stratum is immediately underlain by clay, the wellpoints can be installed in holes penetrating 3 or 4 ft into the clay and backfilled with sand so that the water level in the wellpoints during pumping can be maintained at or below the bottom of the pervious stratum. This procedure will reduce or eliminate seepage that would otherwise bypass the wellpoints had they only been installed with their tips at the top of the clay stratum.

Deep Wells. Deep wells can be used effectively for dewatering slopes and excavations where the required lowering is fairly large. Such wells can be located near or at the top of the excavation and pumped by deep-well turbine pumps, or they can be located on the excavation slope and pumped by a centrifugal pump connected to a common header pipe.

FIG. 3-17. Wellpoint spacing for stratified clean sands and gravels. (*Moretrench Corp.*)



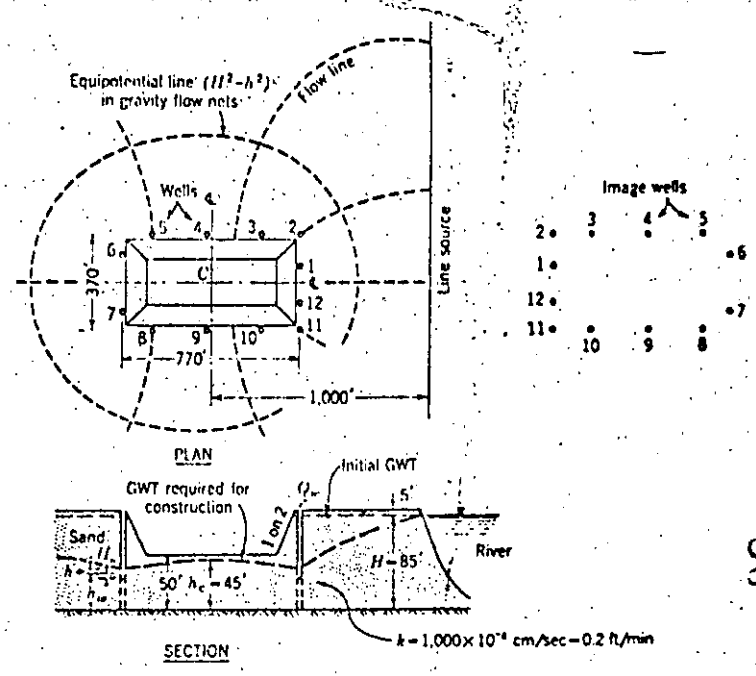
In the latter case it is necessary to lower the header pipe and pumps in about 15-ft stages as the excavation is carried down. Also, the header must be raised in 15-ft stages when "backing out of the excavation" to provide ground-water lowering until construction below the initial ground-water level is completed. The comparative costs of wells at the top of the slope pumped with deep-well pumps and wells located at the toe with a common collector should be considered when selecting the system to be used. Locating the wells at the top of the excavation will eliminate interference with excavation and construction.

Procedures for designing a system of deep wells are somewhat similar to those for wellpoints. As in the case of wellpoints, the tops of screens for deep wells connected to a common collector pumped by a centrifugal pump should be set below the computed water surface in the well. This requirement is not necessary if each well is pumped by a deep-well pump. The bottom of the well should be set to provide sufficient length

of submerged screen to admit the flow without excessive entrance head loss (Fig. 3-42).

Figure 3-18 illustrates the design of a system of deep wells suitable for

FIG. 3-48. Design of a deep-well system for slope dewatering.



Problem. Design a system of 10-in.-diameter wood-stave wells, pumped by deep-well turbine pumps, for lowering the ground-water level 5 ft below the bottom of the excavation. Assume maximum allowable $Q_w = 1{,}200$ gpm, wells located 5 ft from top of slope, well radius $r_w = 1$ ft, and D_{15} of gravel filter = 0.25 mm.

Solution. Estimate total flow required from Eq. (3-73) using radius A_r of an equivalent large-diameter well computed from Eq. (3-89) (see Fig. 3-36).

$$A_r = \frac{1}{\pi}\sqrt{770 \times 370} = 340 \text{ ft}$$

$$Q_T = \frac{\pi(0.2)(85^2 - 45^2)}{\ln [(2 \times 1{,}000)/340]} = 1{,}840 \text{ cfm} = 13{,}800 \text{ gpm}$$

Use 12 wells with $Q_w = 1{,}150$ gpm. Locate wells as shown in plan so as to intercept equal quantity of flow as indicated by flow net and to obtain approximate level drawdown beneath excavation. Compute head h_c at center of excavation and head h_w at a well to check adequacy of system.

HEAD AT POINT C AND WELL 4 COMPUTED BY METHOD OF IMAGES FOR Q_w = 1,150 GPM = 153 CFM

Well	Head at point C			Head at well 4		
	S_i, ft	r_i, ft	$\ln \frac{S_i}{r_i}$	$S_{i,4}$, ft	$r_{i,4}$, ft	$\ln \frac{S_{i,4}}{r_{i,4}}$
1	1,620	390	1.42	1,650	410	1.39
2	1,630	420	1.36	1,640	400	1.41
3	1,800	290	1.82	1,800	240	2.02
4	2,040	180	2.42	2,050	1	7.63
5	2,280	330	1.93	2,300	250	2.22
6	2,400	390	1.82	2,420	370	1.88
7	2,400	390	1.82	2,435	400	1.67
8	2,280	330	1.93	2,330	410	1.67
9	2,040	180	2.42	2,090	370	1.73
10	1,800	290	1.82	1,840	435	1.44
11	1,630	420	1.36	1,675	540	1.13
12	1,620	390	1.42	1,650	480	1.24
	F'_i = 21.54 × 153 = 3,290			F'_w = 25.44 × 153 = 3,890		

From Eq. (3-85), $H^2 - h_c^2 = \frac{3,290}{\pi(0.2)} = 5,250$

From Eq. (3-86), $H^2 - h_w^2 = \frac{3,890}{\pi(0.2)} = 6,200$

$h_c = \sqrt{85^2 - 5,250} = 44.6$ ft $\quad h_w = \sqrt{85^2 - 6,200} = 32.0$ ft

The corresponding flow per foot of well screen is 1,150/32, or 36 gpm per ft.

Compute head loss in well H_w:

$H_r = 0.70$ ft (from Fig. 3-42)

$H_i = 0.35$ (from Fig. 3-44)

$H_r + H_i = 1.25\left(\frac{32}{100} \times \frac{1}{3}\right) = 0.13$ (from Fig. 3-43 and using the flow through one-third the length of screen (see page 313))

$H_w = 1.18$ ft, say, 1.2 ft

Thus $h_w - H_w = 32.0 - 1.2 = 30.8$ ft. Bowls of pump should be set about 2 ft below this level, and the pump provided with a 10-ft suction pipe. With such a suction pipe, $H_r + H_i$ will be slightly less than the value computed above.

Had the approximate method on Fig. 3-37 been used, the following values of F'_i and F'_w would have been obtained from Eq. (3-92a) and Eq. (3-93), respectively:

$$F'_i = 153 \times 12 \ln \frac{2 \times 1,000}{340} = 3,250$$

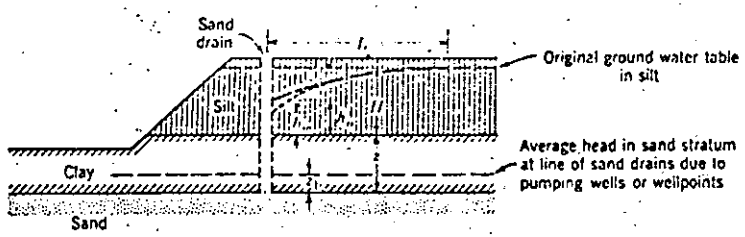
$$F'_w = 153 \times 12 \ln \frac{2 \times 1,025}{340} + 16 \frac{340}{12 \times 1} = 3,800$$

These values agree closely with those computed by the exact method.

dewatering an excavation in sand close to a river. Assume that the wells are located 5 ft from the crown of the excavation slope and that they are provided with 40 ft of 10-in.-ID wood-stave pipe screen perforated with 3/16-in. slots totaling 40 sq in. of open area per foot of screen and that the screens are surrounded by a graded gravel filter 6 in. thick. Lowering the water table as indicated on Fig. 3-48 with a peripheral system of wells requires a total pumping rate of about 13,800 gpm. This pumping rate would suggest, as a tentative design, 10-in. wells and deep-well pumps with a capacity of about 1,000 to 1,200 gpm, thus requiring about 12 wells. For this type of well and flow, H_w would equal about 1 ft, about the maximum desirable. The wells should be located on the basis of a plan flow net so that each well would intercept the same amount of seepage and so that the head at all wells would be the same. With this arrangement and by pumping each well at the same rate, a fairly uniform lowering of the ground-water level beneath the bottom of the excavation should result. The adequacy of the array can be checked by computing the head h_c at the center of the excavation as shown on Fig. 3-48. The head at a selected well and drawdown therein should then be computed to determine the elevation at which the bottom of the pump or pump bowl should be set. Had a fewer number of wells been considered, they would have had to be larger in diameter since a nominal 10-in. pump could not produce the required flow. Note that the ground-water level at each well would be slightly higher than the water level in the well because of the effect of the free discharge surface (see pages 291-294). The true level at the well can be computed from the equations given on the previous pages. Since the head h_c is at a distance from the wells greater than 1.5 times the value of h_c computed on Fig. 3-48, it is affected little by the free discharge surface and needs no correction.

Vertical Sand Drains Combined with Wellpoints or Wells. In cases where a stratified semipervious stratum with low vertical permeability overlies a relatively pervious stratum, and it is necessary to lower the ground-water level in both strata, the ground-water level in the upper stratum can be lowered by means of sand drains. If properly designed and installed, sand drains will intercept the seepage in the upper stratum and conduct it into the lower, more permeable stratum being drained by wellpoints or wells. The drains should be of such size, permeability, and spacing as to conduct the flow to the lower sand stratum with small hydraulic head loss in each drain. Sand- or gravel-filled drains are not effective where installed in highly pervious soils because they do not have enough hydraulic carrying capacity to permit flow to the lower stratum without excessive head loss. An example of vertical drains together with equations for computing the required spacing of drains is given in Fig. 3-49.

FIG. 3-39. Design of sand drains for slope dewatering.



Equations

$$Q_D = \frac{k_D(z - z_1 + h_a)A_D}{z + h_a} \qquad (3\text{-}1a)$$

$$Q_s = \left(0.73 + 0.27\,\frac{H - h_s}{H}\right)\frac{kn}{2L}(H^2 - h_s^2) \qquad (3\text{-}24)$$

$$h_s^2 = h_a^2 + \frac{Q_s}{\pi k}\ln\frac{a}{2\pi r_D} \qquad (3\text{-}40a)$$

Note: To solve the above equations simultaneously it is necessary to assume $h_a = 0$

Notations

Q_D = vertical flow per drain
Q_s = seepage through stratum being drained per length a measured along line of drains
k_D = vertical permeability of drain
A_D = sectional area of drain with radius r_D
k = permeability of stratum being drained
h_s = head at equivalent slot simulating line of drains
h_a = head at sand drain
a = spacing of drains

Other dimensions and symbols are as shown.

Problem. Given a condition as shown, where $k = 5 \times 10^{-4}$ cm per sec, $H = 15$ ft, $z = 12$ ft, $z_1 = 3$ ft, $k_D = 1{,}000 \times 10^{-4}$ cm per sec, $r_D = 0.5$ ft, $A_D = 0.785$ sq ft. Determine spacing of sand drains required to drain upper silt stratum.

Solution. Compute Q_D from Eq. (3-1*a*), assuming $h_a = 0$. Since Q_s must equal Q_D, substitute this value of Q_D in Eq. (3-40*a*) and compute h_s for various values of a assuming $h_a = 0$. Using these values of h_s and a, compute Q_s from Eq. (3-24). The required spacing a is that which makes Q_s from Eq. (3-24) equal to Q_D computed from Eq. (3-1*a*).

From Eq. (3-1*a*)

$$Q_D = \frac{0.20(12 - 3)0.785}{12} = 0.118 \text{ cfm} = 0.89 \text{ gpm}$$

Substituting this value of Q_D in Eq. (3-40*a*) and assuming $h_a = 0$ gives

$$h_s^2 = \frac{0.118}{\pi \times 0.001}\ln\frac{a}{2\pi(0.5)} \qquad (3\text{-}40a)$$

Also, from Fig. 3-39, $L = 100$ ft for $H - h_s = 15$ ft. Substituting this value and the other constants into Eq. (3-24) results in the following equation:

$$Q_s = \left[0.73 + \frac{0.27(15 - h_s)}{15}\right]\frac{0.001a}{2 \times 100}(15^2 - h_s^2) \qquad (3\text{-}24)$$

Compute h_s and Q_s for various values of a from Eq. (3-40*a*) and Eq. (3-24), respectively, which results in the following values:

a, ft	h_s, ft	Q_s	
		cfm	gpm
5	4.17	0.045	0.34
10	6.58	0.086	0.65
15	7.65	0.117	0.88
20	8.33	0.143	1.07

From the above tabulation the required spacing is 15 ft since the corresponding value of $Q_s \approx Q_D$ computed from Eq. (3-1*a*). However, since the above equations do not consider effect of entrance head loss, the drain spacing should be reduced somewhat. Therefore a spacing of about 10 to 12 ft would be used.

Electroosmosis. Some extremely fine grained soils cannot be drained by the previous methods, but can be drained successfully by the electroosmotic method. The required system consists of positive electrodes (anodes) and negative electrodes (cathodes) across which a d-c voltage is impressed, creating flow of pore water to the cathode. The cathode can consist of a well or wellpoint, surrounded by a sand filter, which collects the flow. The discharge is removed by pumping. Since electroosmosis is not used frequently, detailed design procedures are not presented. However, a summary of pertinent features of successful systems is as follows:

Cathodes have been installed in one or more lines and spaced on about 25- to 35-ft centers, with anodes installed midway between the cathodes. Where more than one line of cathodes was installed, the anodes were placed along a line midway between and parallel to the lines of cathodes. Where multiple lines of cathodes were used, the distance between them was about 30 to 45 ft. The proper spacing of electrodes depends mainly on the voltage available at the site. Potential gradients of more than 0.5 volt per cm between electrodes should not be exceeded for long-term applications because higher gradients result in excessive energy losses in the form of heating of the ground. Anodes can consist of any available conductor, such as steel pipe, railroad rail, etc. Cathodes usually consist of small-diameter wells or wellpoints but with sufficient diameter to admit a suction pipe (usually 1-in.-diameter) from a pump. Since the rate of discharge at a cathode is small, say, 4 to 200 gpd, intermittent

pumping may suffice for removing water collected in the well. Anodes and cathodes should extend in depth at least 5 ft below the bottom of the slope or excavation being stabilized. Impressed voltages vary between 30 and 100 volts, the lower voltages being satisfactory where the ground water contains a high concentration of minerals. Current requirements range between 15 and 30 amp per well; over-all power consumption usually is high. Power required per well on successful systems designed by Casagrande (4) ranged from 0.5 to 2.5 kw per well for respective gradients of about 1.5 and 4 volts per ft distance between electrodes.

An estimate of the discharge Q_s to a well can be obtained from the following equation:

$$Q_s = k_e i_e az \tag{3-103}$$

where k_e = coefficient of electroosmotic permeability, which can be assumed to be 0.5×10^{-4} cm per sec per volt per cm
i_e = gradient, volts per cm, between electrodes
z = depth of soil being stabilized
a = effective spacing of wells

The current required can be estimated from the following expression developed by Maclean and Rolfe (31):

$$It = 4.1c - 25 \tag{3-104}$$

where I = current, amp, required per gram of water expelled
t = time, sec
c = "clay" content of soil, per cent (per cent by weight of soil finer than 0.002 mm)

From Eq. (3-104) it can be seen that power requirements will be large since even where $c = 19$ a current of 25 amp will be required to produce a flow of about 37 gpd per well or wellpoint. However, even though power requirements are large, the electroosmotic method may be the only practical means of stabilizing certain soils.

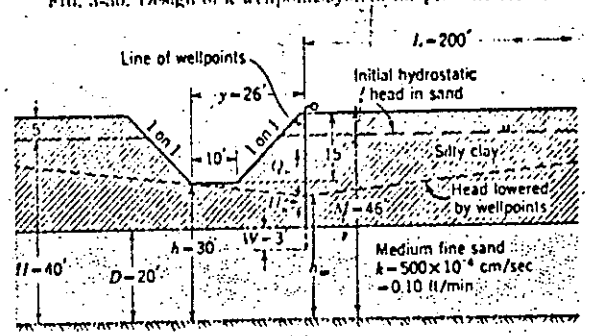
3-29. Pressure Relief beneath Excavations

Where an excavation is underlain by a pervious stratum under artesian head capable of causing heaving or sand boils in the bottom of an excavation, the head should be lowered to safe values by either wellpoints or deep wells. The allowable upward-seepage gradient depends upon the uniformity and permeability of the fine-grained soils overlying the pervious stratum. In uniform plastic clays upward hydraulic gradients as high as 0.5 may be safe, whereas in silty soils it may be necessary to lower the artesian head below the bottom of the excavation in order to control upward seepage and achieve a dry, stable bottom. Stratification of the soil will also affect the allowable uplift pressure.

Pressure-relief systems may be designed using applicable equations for artesian flow presented previously. Since the flow required for a given head reduction depends upon the penetration of the wells or wellpoints, this factor should be considered in design. To avoid excessive pumping, the penetration should be no greater than that necessary to achieve the required drawdown. However, stratification of the sand being drained has an appreciable effect on the penetration required; therefore the well penetration must be based on the transformed dimensions of the system if the pervious substratum is anisotropic, as described previously. Principles to be considered in the design of wellpoint and deep-well systems for pressure relief together with numerical examples are given in the following paragraphs.

Wellpoints. Each stage of a wellpoint system is capable of reducing artesian head only about 15 to 20 ft because of hydraulic limitations of the system as described previously in Sec. 3-28 on dewatering slopes and excavations. However, a wellpoint system can be used effectively for relieving artesian pressure where the length of riser pipes, thickness of pervious substratum, and the discharge required to produce the head reduction are not excessive. The design of a single-stage wellpoint system for relieving excess hydrostatic pressure in a 20-ft-thick sand stratum beneath a long, narrow excavation for a sewer is illustrated in Fig. 3-50.

FIG. 3-50. Design of a wellpoint system for pressure relief.



Problem. Determine required spacing of 2½-in.-ID style CH self-jetting Moretrench wellpoints with 2-in.-ID riser pipes to lower hydrostatic head to bottom of trench. Assume effective vacuum at top of riser pipe = 20 ft, $L = 200$ ft, and $r_w = 0.104$ ft.

Solution. Use a single line of wellpoints at top of excavation, one stage being required. For $W/D = \frac{3}{20} = 0.15$, $\lambda = 0.82$ from Fig. 3-19; therefore $\lambda D = 0.82 \times 20 = 16.4$ ft. Maximum h at trench = 30 ft. Assume this value of h at the far edge of the trench, a distance y of 26 ft from the line of wellpoints. Compute the required h_r from Eq. (3-27) as follows:

$$20 = h_w + (40 - h_w)\frac{26 + 16.4}{200 + 16.4} \quad \text{or} \quad h_w = 27.7 \text{ ft}$$

The flow Q_p per unit length of system as computed from Eq. (3-26) is

$$Q_p = \frac{2 \times 0.1 \times 20 \times 1 \times (40 - 27.6)}{200 + 16.4} = 0.23 \text{ cfm} = 1.7 \text{ gpm per ft of trench}$$

Compute Δh_w from Eq. (3-32), h_w from Eq. (3-33), and H_w from Eq. (3-101), and select a so that $h_w - H_w \geq 20$ ft (M minus the vacuum at the top of the riser pipe).

a, ft	Q_w cfm	Δh_w ft	h_w ft	Head loss in wellpoint, ft				$h_w - H_w$ ft
				H_s*	H_r†	$H_r + H_s$‡	H_w	
10	2.3	0.50	27.1	1.75	0.22	0.87	2.84	24.3
8	1.8	0.36	27.2	1.16	0.17	0.54	1.87	25.3
6	1.4	0.24	27.4	0.74	0.13	0.34	1.21	26.2

* From Fig. 3-41*b*.

† From Fig. 3-41*a*, assuming H_r same as that given by curve 7.

‡ From Fig. 3-41*c*, assuming $C = 110$.

Thus a spacing of 6 ft would be required, since $h_w - H_w$ should not be less than 20 ft. The tops of the wellpoint screens would be set slightly below the top of the sand stratum.

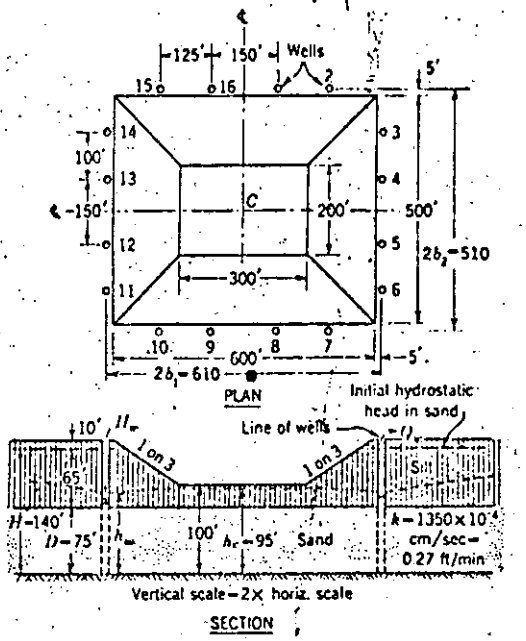
In this case it was assumed that the source of seepage consisted of two vertical lines equidistant from and parallel to the line of wellpoints. The wellpoint spacing was determined so that the drawdown inside the wellpoint, including the effect of hydraulic head losses, would equal the effective vacuum at the top of the wellpoint riser pipe.

Deep Wells. Deep wells can be used to relieve excess hydrostatic pressure in deep strata. They may be pumped individually by deep-well turbine pumps or connected to a collector pipe pumped by a centrifugal wellpoint pump. The well screens should be of sufficient length to admit the flow with small head loss, and the well should be large enough for the pump required and to keep head losses low. Details of design of well screens have been discussed previously. Design of a typical pressure-relief well system is presented below.

Figure 3-51 illustrates the design of a system of deep, large-diameter wells for reducing hydrostatic heads in a thick pervious sand stratum beneath a 200- by 300-ft excavation, 50 ft deep. Sixteen 10-in.-diameter wells pumped by deep-well turbine pumps would be required to lower the water table as indicated. In this case the head in the sand was reduced 5 ft below the bottom of the excavation to prevent upward seepage through the semipervious silt stratum from entering the bottom

of the excavation. Wells were located at the top of the excavation slope so as not to interfere with excavation and construction. The elevation of the water surface in the wells was computed to determine the elevation at which the impellers of the pump should be set and to check on submergence of screen. Had the wells been connected to a common collector pumped with a centrifugal pump, they would have been located along

FIG. 3-51. Design of a deep-well system for pressure relief.



Problem. Determine number of 10-in.-diameter wells with 6-in. gravel filter required to lower head in sand stratum 5 ft below bottom of excavation, for wells located at top of slope and pumped by deep-well turbine pumps. (Thus $r_w = 1.0$ ft.) Use a fully penetrating system of wood-stave wells with $\frac{3}{16}$-in. slots and a gravel filter with D_{15} size = 0.25 mm. Area of slots $\cong$ 10 per cent of circumferential area of well screen. (Geologic and soil conditions indicate a circular source of seepage.)

Solution. Determine equivalent radius A, of well system from Eq. (3-89), with wells located 5 ft from crown of slope.

$$A = \frac{1}{\pi}\sqrt{510 \times 610} = 355 \text{ ft}$$

From Fig. 3-39, $R \cong 4{,}700$ ft for $k = 1{,}350 \times 10^{-4}$ cm per sec and $H - h_w = 45$ ft.

84

Compute total required flow Q_T from Eq. (3-47) for $h_w = h = 95$ ft and $r_w = A_r = 355$ ft.

$$Q_T = \frac{2\pi(0.27)75(140 - 95)}{\ln 4{,}700/355} = 2{,}130 \text{ cfm} = 16{,}000 \text{ gpm}$$

As 1,000 gpm is about the maximum that can be pumped by a nominal 10-in.-deep well pump, 16 wells would be required. Try a spacing as shown on the plan, and compute the residual head at the center of the excavation from Eq. (3-75*b*) as shown below. Since the wells are symmetrical about both center lines of the excavation, the total drawdown can be obtained by computing the drawdown caused by the four wells in one quadrant and multiplying the result by 4.

Well	R, ft	r_n, ft	$\ln \frac{R}{r_n}$
1	4,700	266	2.87
2	4,700	324	2.68
3	4,700	352	2.60
4	4,700	314	2.71
			$\Sigma = 10.86$

$$Q_w = 1{,}000 \text{ gpm} = 133 \text{ cfm} \quad \text{for each well}$$

For 4 wells,
$$H - h_e = \frac{133(10.86)}{2\pi(0.27)75} = 11.4 \text{ ft}$$

For all 16 wells,

$$H - h_e = 45.6 \text{ ft} \quad \text{or} \quad h_e = 140 - 45.6 = 94.4 \text{ ft}$$

Since the maximum allowable h_e is 95 ft, the system shown in plan is adequate.

The approximate head h_w at a well can be computed from Eq. (3-32) as follows, using an average well spacing of $2(510 + 610)/16 = 140$ ft.

$$\Delta h_w = \frac{133}{2\pi(0.27)75} \ln \frac{140}{2\pi(1.0)} = 3.3 \text{ ft}$$

Thus
$$h_w \cong 94.4 - 3.3 = 91.1 \text{ ft}$$

Hydraulic head loss in the wells is as follows, assuming intake of pump about 85 ft above bottom of sand.

$H_r = 0.26$ ft from Fig. 3-44
$H_r = 0.07 = 0.71^*(\frac{10}{100})$ for 10 ft of submerged riser
$H_s = 0.25 = 1.00(\frac{25}{100} \times 75 + 100)$ assuming uniform rate of screen inflow and $C = 100$
$H_s = 0.26$ from Fig. 3-42, with $Q_w = 13.3$ gpm/ft of screen
$H_w = 0.84$ ft

Thus the water surface in the wells would be about $91.1 - 0.84 = 90.3$ ft above the bottom of the sand. The pump bowl should be set about 85 ft above the bottom of the sand and provided with a 10-ft suction pipe.

* From Fig. 3-43, with $C = 130$.

the toe of the excavation slope. However, this would have required moving the header pipes and pumps down about three times as the excavation was made, but the number of wells could have been reduced to 12 or 13, assuming a maximum flow of 1,000 gpm per well. Also, it would have been necessary to compute the elevation of the water surface in the wells for each stage of lowering to determine the elevation at which the collector pipe and pumps should be set in keeping with the vacuum available in the collector pipe. Had the thickness of the pervious stratum been small, for example, 5 or 10 ft, wellpoints instead of large-diameter wells might have been more economical.

3-30. Pumps, Headers, and Discharge Lines

Pumps, headers, and discharge lines must be of sufficient capacity to remove the required flow from the wells or wellpoints and conduct it away from the dewatered area. If the header pipes are too small, excessive hydraulic head losses will develop and prevent lowering the ground water or head to required levels. A brief summary of pertinent features affecting design or selection of centrifugal and deep-well pumps and collector pipes is presented. Detailed data can be found in textbooks on hydraulics and catalogues issued by pump manufacturers.

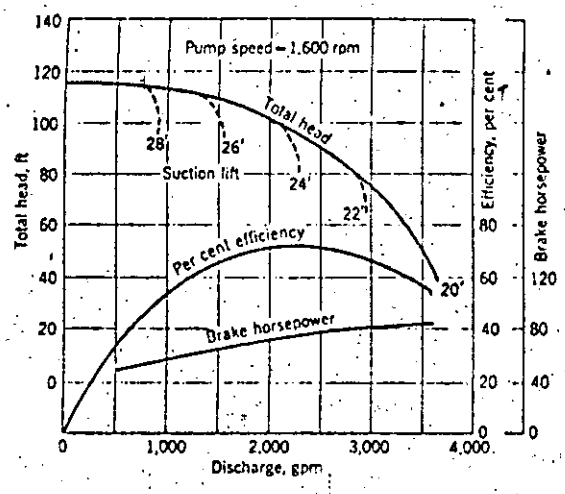
Centrifugal Pumps. Centrifugal pumps are used for pumping collector pipes connected to well or wellpoint systems and for removing surface water from sumps. The former are usually termed *wellpoint pumps*, the latter *sump pumps*.

The selection of a pump and power unit depends on the required discharge, total dynamic head ("suction" lift plus positive head including hydraulic head losses), suction lift, air-handling capacity, power available, fuel economy, and durability of units.

Pumps connected to collector lines for wells or wellpoints should have adequate air-handling capacity and be capable of producing a high vacuum since the amount of dewatering or pressure relief of such systems is affected by the vacuum available at the pump. For these purposes a wellpoint pump consisting of a self-priming centrifugal pump with attached vacuum pump is used, since it can develop a vacuum of about 20 to 25 ft of water. It is usually safe to assume in design that a 20-ft vacuum can be developed by the pump. Since the capacity of a wellpoint pump depends in part on positive pressure head, this factor also must be considered.

The capacity of a wellpoint pump depends upon pump speed, suction lift, and the total dynamic head. The characteristics of a typical 8-in. wellpoint pump, operated at a rated speed of 1,200 rpm, are shown in Fig. 3-52.

Fig. 3-52. Characteristics of 8-in. Griffin wellpoint pump. (*Griffin Wellpoint Corp.*)



Electric-, gasoline-, diesel-, or butane-power units can be used to drive a pump. It is preferable to obtain the pump and motor as a unit. In selecting the power unit, consideration should be given to the initial cost of the unit and its cost of operation, including maintenance and fuel. Where electric motors are supplied by commercial power, gasoline or diesel motor-generator units or pump motors should be provided as stand-by equipment in case of power failure. It is desirable to provide 75 to 100 per cent stand-by equipment where electric motors are used. Where gasoline or diesel motors are used, some stand-by units should be provided; the number will depend on the nature of the dewatering problem and number of units operating. Stand-by equipment should be ready for immediate use during any emergency.

Centrifugal pumps for removal of water collected in ditches and sumps are also necessary on most projects. These sump pumps should be self-priming and capable of developing a vacuum of at least 20 ft. Their capacity depends upon pump speed, total dynamic head, and suction lift.

When selecting a pump and motor unit, a conservative approach should be taken. It must be remembered that computed flows from a dewatering or pressure-relief system are not exact, that for long-time dewatering jobs some loss in pump efficiency will occur, and that it is better to have some excess capacity rather than inadequate capacity.

The location and spacing of wellpoint pumps, as well as their capacity, are important and are affected by the length of header pipe to be pumped, rate of flow, and point of discharge. If a long collector line (say, 500 or 1,000 ft) is pumped by a single pump, the pump should be located at the center of the line to obtain a maximum vacuum in the line. In short lines and where the flow is small, the pump can be located wherever convenient. The intake of the pump should be set as low as practicable. The intake should not be more than 15 to 18 ft above the bottom of the excavation being dewatered. Where the discharge from the system will be large, the pump intake should be set at the same elevation as the collector line and connected with a minimum of fittings. Hydraulic head losses in the wells, header, and discharge line should be taken into account in designing the system.

Deep-well Pumps. Deep-well turbine or submersible pumps are used to pump large-diameter wells and consist of one or more impellers on a vertical shaft driven by a motor. Motors of most large-capacity pumps used in deep wells are located at the top of the pump.

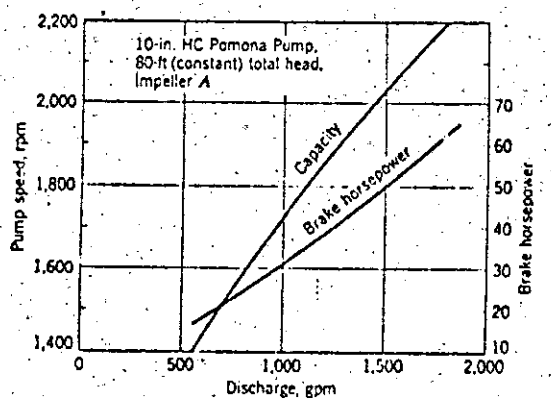
The capacity of a deep-well pump depends upon its size, speed, number of stages (impellers), type and size of impeller, and total dynamic head. When selecting pumps for a given project, manufacturers' representatives should be consulted regarding selection of the unit to ensure that pumps of proper capacity are obtained. This is important since the capacity of a well frequently is limited by the pump, and not the available yield of the pervious substratum in which the screen is set, and since a wide variety of deep-well pumps are available. However, for rough estimating purposes the following table is of use in determining the approximate maximum capacity of deep-well pumps normally available for dewatering:

Pump bowl size, in. (minimum ID of well pump will enter)	Preferred minimum ID of well, in.	Approximate maximum capacity, gpm
4	5	90
5½	6	160
6	8	450
8	10	900
10	12	1,200
12	14	1,800
14	16	2,400
16	18	3,000

A rating curve for a typical 10-in. three-stage pump is shown on Fig. 3-53. The normal speed of this pump is 1,750 rpm, and the capacity at

this speed is about 1,050 gpm. However, the capacity can be increased by operating the pump at a faster speed. Pumps should be selected to operate at their normal rated speeds, realizing that some margin of safety exists because of available additional capacity at speeds greater than normal.

FIG. 3-53. Rating curves for three-stage 10-in.-high-capacity deep-well pump. (*Fairbanks-Morse & Co.*)



Header Pipes. Header pipes should be capable of carrying the flow from wells or wellpoints to the pumps with small head loss, since such loss directly reduces the drawdown obtainable with centrifugal pumps. Head losses to be considered are those due to velocity and friction and to enlargements, tees, elbows, valves, and other discontinuities in the line. Friction and velocity head losses can be determined from Figs. 3-13 and 3-14, respectively. Losses due to irregularities in the line can be estimated from coefficients given in hydraulic textbooks.

Header pipes commonly consist of relatively lightweight steel pipe; headers for wellpoints contain inlets for wellpoint connections at short intervals, permitting a wide range in wellpoint spacing. Locations of header pipes and pumps should be planned well in advance of construction. Headers should be installed slightly above the prevailing ground-water level and where they are accessible. They should be located on the excavation slope or a berm thereon where the depth of ground-water lowering or initial ground water precludes locating them at the top of the slope. The header of a single-stage system or lowest header of a multistage system should be located not more than about 15 ft above subgrade to ensure that proper drawdown of the ground-water level can be achieved with the vacuum available in the line.

The header pipe should be kept reasonably straight and as free as possible from discontinuities.

Discharge lines from pumps can consist of steel or aluminum pipe and should be designed to conduct the required flow with relatively small head loss since these losses add to the total pressure head and thereby reduce the capacity of the pump. In some situations it is possible to dig ditches to conduct the flow from the site. However, such ditches must be kept well back from the top of the excavation to keep from saturating the upper part of the slope; the distance necessary will depend upon the type of soil.

3-31. Control of Surface Water

In laying out a dewatering or pressure-relief system, adequate facilities should be provided for collecting and removing surface water so that the pumps cannot be flooded, resulting in failure of the dewatering system. Uncontrolled runoff also can cause serious erosion of slopes, with consequent costly repair and maintenance. Measures which can be used to control surface water include dikes, ditches, sumps and pumps, and mulching and seeding to minimize slope erosion. Factors to be considered in designing surface-water control measures include duration of construction, frequency of rainfall occurrence, intensity of rainfall and resulting runoff, size of area to be protected, and available sump storage.

The frequency of occurrence used in selecting a design storm will depend upon the duration of construction and the risk involved in failure of the dewatering or pressure-relief system. The intensity and duration of the design storm selected will depend upon the season of the year, geographical location of the site, time of concentration, and available storage. Intensity of rainfall for storms of various frequency and duration can be estimated from charts and tables in Ref. 32 when detailed rainfall data are not available.

After the rainfall rate has been selected, the rate of runoff Q_R in cubic feet per second can be computed from the following equation:

$$Q_R = C_R i A \tag{3-105}$$

where C_R = a coefficient of runoff (the imperviousness of the area expressed as a decimal)
i = maximum average rate of rainfall over drainage area, in. per hr, occurring during the time of concentration
A = area drained, acres

The value of C_R depends on the relative porosity, character, and slope of the surface. For saturated, steep excavations, C_R values of 0.7 to 0.95 are applicable.

Dikes and Ditches. Where an excavation is made in an area which drains nearby areas, a dike should be built around the top of the excavation to eliminate runoff into the excavation from the surrounding area. Sumps or combined ditches and sumps which can be utilized for storage will reduce the capacity of pumping equipment required. Dikes should be high enough to prevent the water from overtopping them and of sufficient section to withstand the head against them. The top of the dike should be at least a foot above the computed elevation of surface water to be impounded. Dikes should have a crown width of 3 to 5 ft and side slopes of 1 on 2 or 2.5.

Dikes combined with ditches can be located on excavation slopes to control runoff and reduce slope erosion. Runoff retained by dikes can be conducted to sumps in the bottom of the excavation by pipes or lined channels and then pumped out of the excavation, or the runoff retained by the dikes can be pumped out where collected on the slope. Ponding runoff on excavation slopes is somewhat risky because any overtopping of a dike could result in overtopping of all dikes at lower elevations, with resultant flooding of the excavation. Where runoff is ponded on and pumped from slopes, the slopes of the excavation should be divided into subareas enclosed by the dikes and sump pumps provided with sufficient capacity to remove the computed runoff into each subarea.

Ditches should contain ample allowance for silting, freeboard, and storage. A ditch with inadequate capacity is potentially dangerous and therefore should be conservatively designed. This is especially true of ditches located on excavation slopes. Velocities of flow must be low enough to prevent erosion, and the ditch should be large enough to reduce the amount of maintenance necessary to keep the ditch unobstructed. In silty erodible soils, ditches and dikes should be seeded to reduce erosion. Even with these precautionary measures, some maintenance can still be expected. Combined dikes and ditches on excavation slopes should be designed so that the cut for the ditch balances the fill in the dike. Dikes and ditches used to collect and store surface runoff in the excavation for Port Allen Lock, La., are shown in Fig. 3-9.

Capacity of Sumps and Pumps. The required capacity of pumps for pumping surface runoff depends upon the volume of storage available in sumps as well as the rate of runoff. For example, if no storage is available, it would be necessary to pump the runoff at the rate it enters the excavation to prevent flooding. This usually is not practical. In large excavations, sumps should be provided where practicable to reduce the required pumping capacity. The volume of sumps and their effect on pump size can be estimated from the following expression:

$$Q_D = Q_R - \frac{V}{T} \tag{3-106}$$

where Q_D = total pump capacity, cfs
Q_R = average rate of runoff, cfs
V = volume of sumps, cu ft
T = duration of rainfall, hr

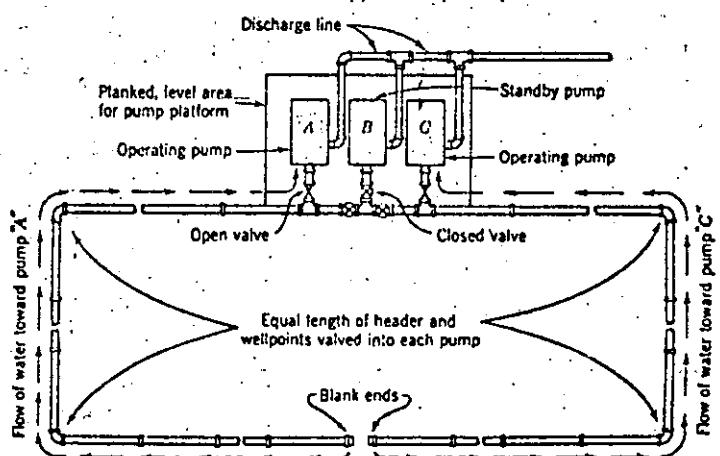
INSTALLATION AND OPERATION OF DEWATERING SYSTEMS

Successful and efficient performance of any dewatering system requires that it be properly installed and operated. Unless the system is correctly installed and operated, it may fail to accomplish the dewatering or pressure relief required, no matter how carefully the system is designed. Equally important is the selection of efficient and dependable equipment for the system being used. Some of the principal features of installation and operation of dewatering systems are presented in the following paragraphs. Additional details regarding installation, operation, and maintenance of wellpoint systems and pumping equipment may be found in sources such as Refs. 8 and 33.

3-32. Wellpoint Systems

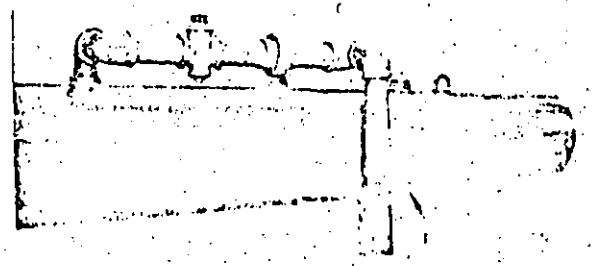
After the plan for the header, wellpoints, and pumps has been developed, the header pipe for the first stage of wellpoints should be laid. Header pipe usually consists of lightweight, plain-end steel pipe with 1½- or 2-in. plugged inlets on 2½- or 3-ft centers. The header is usually joined with *band* or Dresser couplings. The header, valves, tees, and elbows should be laid in accordance with the plan. Care should be taken to lay the header in as straight and level a line as practical. A plan of a typical wellpoint installation is shown in Fig. 3-54.

Fig. 3-54. Plan of a typical wellpoint system.



After the header is laid, half of the swing connection, shown in Fig. 3-55, should be connected to the header pipe on the spacing called for in the design. All fittings and plugs left in the header should be made tight, using a joint compound to prevent leakage.

FIG. 3-55. Swing connection and header pipe. (*Moretrench Corp.*)

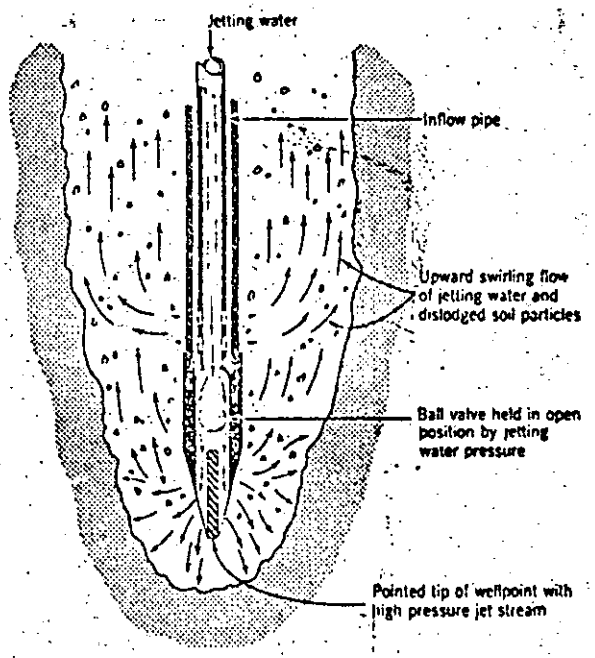


There are two basic types of wellpoints—self-jetting and plain-tip. The self-jetting type is installed by jetting it into the ground with water flowing out of the tip under high pressure. If a hole puncher is used to penetrate the ground or permit placement of a sand filter around the tip and riser, a plain-end type of point can be used. A typical self-jetting wellpoint is illustrated in Fig. 3-56.

Self-jetting wellpoints can be installed in clean medium sand quite easily. After attaching the jetting hose (usually 2-in.-diameter), the wellpoint and riser are picked up by hand or crane, held in a vertical position, and the jet water turned on. A water pressure of 50 to 60 psi is generally adequate to install a wellpoint in ordinary sand. Jet water may be supplied from either a hydrant or a jet pump. [Jet pumps are high-pressure (75- to 125-psi) pumps with capacities of 500 to 1,000 gpm (33).] The wellpoint should be allowed to sink slowly into the soil to ensure washing all fine sand and dirt out of the hole. When the wellpoint reaches final grade and before turning off the water, the two halves of the swing connection should be lined up for easy connection when the jet water is turned off and the jetting hose disconnected. No filter sand or gravel is necessary in medium sand.

Jetting wellpoints in coarse sand with gravel requires considerably more water and water pressure (125 psi) in order to carry out the heavier particles of sand and gravel. Care should be taken to ensure return of jet water to the surface, otherwise the point may "freeze" before it reaches grade. If this tends to occur, the point should be raised until circulation is restored and then slowly lowered. If the soil being penetrated is particularly gravelly, it may be necessary to supplement the jet water with a separate air or water jet at about 125 psi pressure.

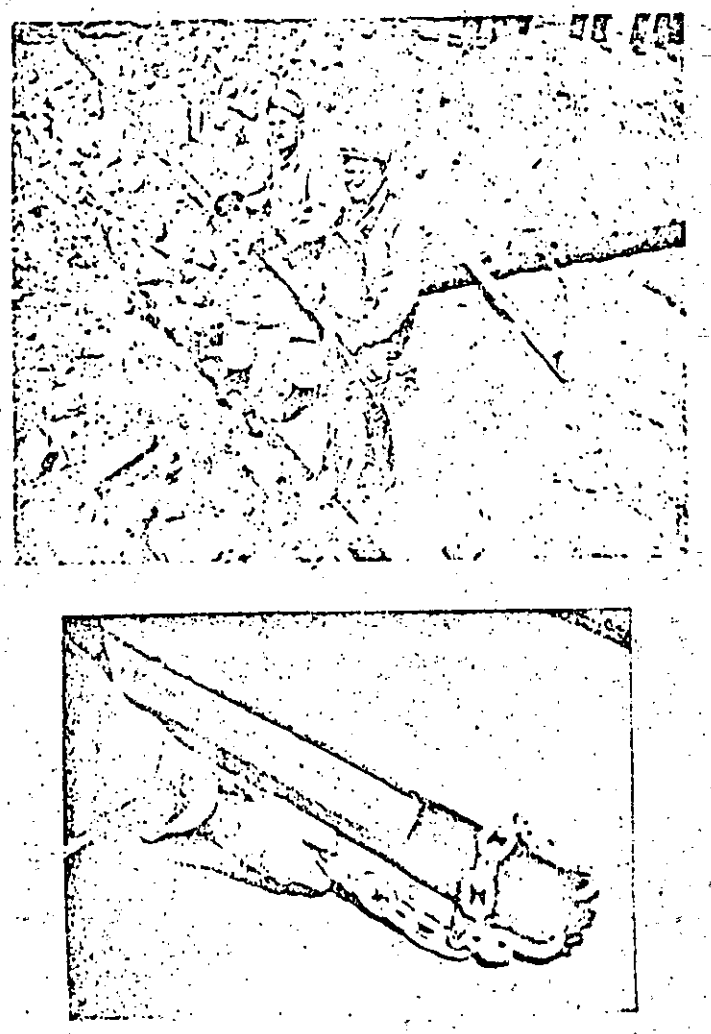
FIG. 3-56. Self-jetting wellpoint. (*Griffin Wellpoint Corp.*)



Proper drainage of fine and dirty sand or sandy soils containing strata of silt or clay requires placement of a filter sand around the tip and riser as shown in Fig. 3-10. This filter permits drainage of overlying pervious strata down to the wellpoint and prevents fine sand or dirt from clogging the wellpoint screen. The sand used for this filter should be clean, reasonably uniform, medium to coarse sand. Wellpoints to be jetted in this type of soil should be equipped with a jetting chain (33) or "star" (8) (Fig. 3-57), so as to create a space around the point and riser for the filter sand. After the wellpoint has been jetted to grade, the jet water should be allowed to run until the return water is practically clear. The chain or star should then be removed from the hole and the jet flow reduced so that there is only a small flow returning to the surface. The filter sand should then be tremied or poured in a *continuous* stream into the hole up to the water table. The remainder of the hole should be filled with silt or clay to minimize air getting into the wellpoint through the sand filter. For this method of installation to be successful, the soil being penetrated must be sufficiently stiff or firm for the hole created to stay open. If the soil is too fluid, the hole will collapse, preventing the sand filter from forming a continuous drain through the various strata to be drained.

The only positive method of placing a sand drain around a wellpoint is to first jet down an 8- to 12-in. heavy-steel casing with cutting teeth at the bottom and a removable head piece on the top. The head piece is generally provided with a 2- or 2½-in. water inlet and a 1¼- or 1½-in. air inlet. The casing is jetted into the ground with water and possibly air pressure, supplied at a pressure of at least 125 psi. Where resistant

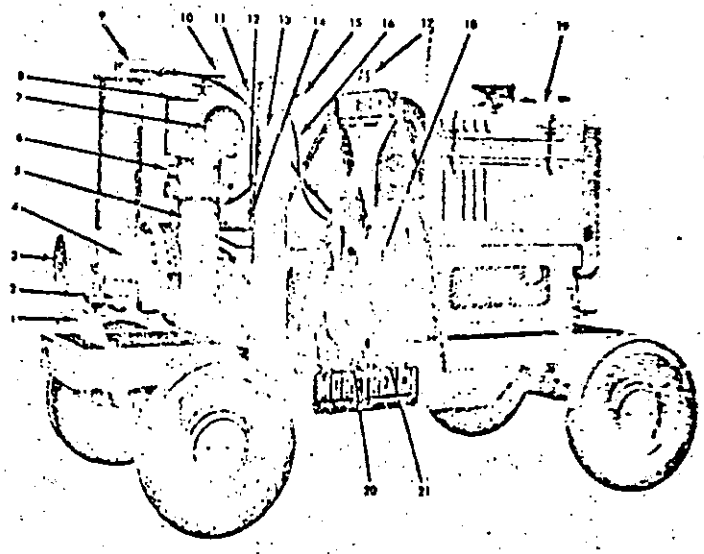
FIG. 3-57. Jetting chain and star. (*Moretrench Corp. and Griffin Wellpoint Corp.*)



strata are encountered, the casing is raised and dropped with the crane or it can be turned down. The casing should be installed to a depth 2 or 3 ft greater than the length of the assembled wellpoint. After the casing or sanding shell is installed to grade, the cap is removed, the wellpoint placed in the casing, the sand filter tremied or poured in, and the casing pulled. Some means should be provided to center the wellpoint in the casing to ensure that filter sand will completely surround the point.

Pumps connected to the header pipe should be regular wellpoint pumps provided with a high-capacity vacuum pump. The pumps should be set level on flooring. It is also desirable to provide some type of shelter for the pumps. Discharge pipe should be supported independently of the pump. Where a large volume of water is to be pumped or the suction lift needed is more than 15 ft, the pump suction should be set level with the header pipe, if practicable. Characteristic parts of a wellpoint pump are illustrated in Fig. 3-58.

FIG. 3-58. Characteristic parts of a wellpoint pump. (1) Alignment support with set screw; (2) clean-out quickcoupling; (3) suction connection; (4) screen box; (5) centrifugal pump; (6) connection for water supply; (7) discharge connection; (8) automatic discharge stop and check valve; (9) automatic float-valve chamber; (10) handwheel for discharge valve; (11) air intake to vacuum pump; (12) clear water lead to packing box; (13) settling tank (drain not visible); (14) water lead to settling tank; (15) air release cock; (16) clear water lead to vacuum pump; (17) balanced lifting bale; (18) V-belt pulley for vacuum pump drive; (19) engine; (20) vacuum pump; (21) double flexible coupling. (*Moretrench Corp.*)



3-33. Vacuum Dewatering Systems

Vacuum wellpoint systems are installed in the same manner as described in Sec. 3-32, except that more care should be taken in obtaining an airtight seal around the upper 5 ft of the riser pipe.

The wells for a deep-well vacuum system are installed as described in Sec. 3-35 except that the top of the well and the space around the riser pipe must be carefully sealed. The submersible pumps should be installed in the bottom of the wells and they should have a capacity greater than the maximum anticipated inflow. These pumps should be designed and manufactured so that they will pump out continuously all the water entering the well without surging or cavitating.

3-34. Operation of Wellpoint System

After all wellpoints have been installed and connected, the header line and all joints should be tested for leaks. Proper performance of a wellpoint system requires continuous maintenance of a steady high vacuum. The tightness of a header line can be checked by closing all pump suction lines and swing-joint shutoff valves and filling the header with water under low pressure (10 to 15 psi maximum) and checking the line for leaks. After the header line has been checked for leaks, the pump can be started. With the gate valve between the header and pump suction closed, the vacuum should rise to a steady 25 to 27 in. of mercury; if not, there must be air leaks in the pump or the moving parts are worn. After testing the pump, the gate valve may be opened and vacuum applied to the header. If the pump creates a steady vacuum of 25 in. or more in the header. If the pump creates a steady vacuum of 25 in. or more in the line, it is tight; if a lower steady vacuum is obtained, there are leaks in the line. If a low, unsteady vacuum develops, it may be that the water table has been lowered near the wellpoints so that air is entering the system through one or more screens. If air is entering the wellpoints, the shutoff valves on these points should be adjusted so that water is not being pulled out of the point faster than it is entering. One method of obtaining maximum vacuum and eliminating air leaking into the well screen is to use riser pipe 25 ft or more in length.

If no air leaks are found in the wellpoints or header line, a thorough check of various parts of the pumps which might leak air should be made.

Where minor amounts of seepage or surface water enter an excavation, it may be desirable to remove this water by means of "mops" attached directly to the header pipe with a suction hose or pipe. The mop should be provided with a float and float valve which will cause the mop to open and close as needed without continuous manual attention. The mop should be surrounded with gravel or crushed stone to keep it from clogging.

3-35. Deep-well Systems

Deep wells for dewatering and relief of artesian pressures can be installed by the *reverse-rotary* method or with a bailer and casing. In the reverse rotary method the hole for the well screen and riser pipe is advanced using a bit usually 20 to 30 in. in diameter, depending on the size of well screen and thickness of filter a bit, drill stem, and a constant head of muddy water in the drill hole. Soil from the hole is removed by suction created by a centrifugal pump; the wall of the hole is stabilized by seepage forces, acting against a thin film of fine-grained soil on the wall of the hole, created by maintaining a head of water in the hole several feet above the water table. The effluent from the suction pump is discharged into a sump pit in which the sand, silt, and gravel settle out and from which muddy drilling water flows back into the hole. (No bentonitic drilling mud is needed, nor should it be used in this method of drilling.) For this method of drilling to be successful, the top of the hole or water level in the sump pit should be at least 7 ft above the water table in the sand strata being penetrated. Where the water table is higher than this amount, it may be necessary to build up for the drill rig and sump pit or lower the water table temporarily around the well location by means of temporary wellpoints jetted into place.

Holes for deep wells can also be made by the bailer-and-casing method. The churn-drill method of drilling is not recommended for dewatering wells. For certain conditions it may be feasible to jet in a 4- or 6-in. well casing to the required depth, install a slightly smaller screen in the casing, and then pull the casing. In this latter method, it is not possible to place a filter around the screen.

Holes for deep wells should be drilled vertically so that the screen and riser can be installed straight and plumb. A vertical straight well is necessary to install a turbine pump properly.

The well screen and riser should be provided with guides which will center the assembly in the hole and hold it in place while the filter is placed. After the screen is lowered to grade, the filter, if one is to be placed, should be tremied in. The tremie pipe should be 4 to 6 in. in diameter and should be provided with $\frac{1}{16}$- to $\frac{3}{32}$-in.-wide slots. The tremie pipe should be filled with filter gravel while resting on the bottom of the hole. The tremie should then be slowly raised, keeping the tremie pipe full of filter gravel at all times, until the filter is 5 to 10 ft above the top of the screen. If a temporary casing is used, the filter gravel should be placed in increments not to exceed 2 ft. The tremie and casing should be raised in increments approximately equal to increments of the filter gravel. After the filter is placed, the well should be promptly surged and pumped until the water is clear and free of sand. The capacity of each

well should also be checked by measuring the flow and corresponding drawdown.

The type of pump used in deep wells will depend on the design of the system. They can be pumped with centrifugal, submersible, or vertical turbine pumps. Of course, the amount of drawdown obtainable with a centrifugal pump is limited by available suction lift. Therefore it is usually considered advisable to use submersible or deep-well turbine pumps in order to obtain more drawdown at the wells.

After the wells are developed and pumped until free of sand and the discharge is clear, they can be connected to a common suction header connected to a centrifugal pump, or the turbine pumps can be installed and connected to a discharge line or ditch outside of the excavation. Care should be taken to set the impellers of turbine pumps in accordance with the manufacturer's instructions.

3-36. Collector and Discharge Systems

Suction header pipe and discharge pipes should be carefully laid and supported where necessary over soft ground and at changes in alignment. Valves should be maintained in good operating condition. All header and discharge pipe should be positively protected against breakage by construction equipment, particularly at ramp and road crossings.

Water from the dewatering and surface-water pumps should be conveyed away from the excavation by means of discharge lines, or by ditches where ground, topography, and surface drainage permit.

3-37. Control of Performance

On large dewatering or pressure-relief jobs, it is worthwhile installing some piezometers to measure the water table and/or any artesian pressure at significant locations so that the system can be operated as necessary for construction conditions and to check the adequacy and efficiency of the dewatering system. Piezometer readings should be plotted and analyzed as the operation proceeds. Gauges are normally installed on the vacuum pumps or header lines, and deep wells provided with some means of measuring the water level in each well. It is also advisable to set up some means to measure the discharge from the wellpoint system or wells. Other information which may be of value to obtain is the radius of influence of the dewatering operation, the temperature of the ground water, and the elevation of any nearby streams. Prior to the start of any large-scale dewatering operation it is desirable to check the level of nearby wells that may be affected and the condition and elevation of any buildings immediately adjacent to the project; these items should also be checked during and at the end of dewatering operations.

Evaluation of collected data will be useful in detecting malfunctioning

pumps; clogged wellpoints or screens; the need for cleaning screens, wells, header, or discharge pipe; and correcting any faulty operation before construction operations are impeded or the excavation or foundation damaged.

NOTATIONS

A area through which seepage is passing, drainage area, or radius of a circular group of wells
A_n cross-section area of sand drain
A_r equivalent radius of a group of wells
a' spacing of wells or sand drains
B distance between two lines of wells
b width of drainage slot
b_1 half width of rectangular array of wells
b_2 half length of rectangular array of wells
C Hazen-Williams factor relating to friction loss in pipes
C_R coefficient of runoff
c constant of integration, or clay content of soil
D thickness of homogeneous isotropic aquifer
D_{10} effective grain size
d' thickness of a pervious stratum
R distance well is located from center of circle of influence
R_A extra-length factor
F, F_p factor for computing drawdown at any point due to a group of wells with circular source of seepage
F_c factor for computing drawdown at center of a group of wells with circular source of seepage
F_w factor for computing drawdown at a well in a group of wells with circular source of seepage
F', F_p' factor for computing drawdown at any point due to a group of wells with line source of seepage
F_c' factor for computing drawdown at center of a group of wells with line source of seepage
F_w' factor for computing drawdown at a well in a group of wells with line source of seepage
G correction factor for a partially penetrating well
GWL ground water level
GWT ground water table
g acceleration of gravity
H height of water table (initial) or piezometric surface
H' drawdown
H_a average head loss in header pipe to pump intake
H_e screen entrance loss, or filter and screen entrance loss
H_f friction head loss in riser pipe
H_s friction head loss in well screen
H_v velocity head loss in well
H_w total hydraulic head loss in well or wellpoint
h head at a specific point
h' height of free discharge above water level in well
h_s maximum head landward from a line of drainage wells

h_a head midway between last two wells in a finite line of wells
h_c head at center of a group of wells
h_D maximum head landward from a drainage slot or line of wells
Δh_H maximum head landward from a line of wells above head at wells
h_e head at a drainage slot, or average head at a line of wells
h_m head midway between wells
Δh_m head midway between wells above that at a well
h_s head in a gravity drainage slot, or at equivalent drainage slot simulating a line of wellpoints or sand drains
h_p head at point P
h_r height of free discharge above water level in drainage slot
h_w head at well or sand drain
Δh_w drawdown at well in a line of wells below head (h_e) at an equivalent drainage slot
h_{wj} head at well j in a system of n wells
I electric current
i hydraulic gradient of water table, or maximum average rate of rainfall over drainage area
i_e electrical gradient between electrodes
k coefficient of permeability of homogeneous isotropic aquifer
$\bar{k}$ transformed coefficient of permeability
k_H vertical coefficient of permeability of a sand drain
k_e coefficient of electroosmotic permeability
k_H horizontal coefficient of permeability
k_V vertical coefficient of permeability
L distance from drainage slot, well, or line of wells or wellpoints to a line source of seepage
L_0 distance from drainage slot to change from gravity to artesian flow
L_j distance from well j to source of seepage
L_1 distance from river to riverside toe of cofferdam
L_2 base width of cofferdam
l half the distance between two parallel drainage slots or lines of wells
M height of pump intake above base of aquifer
MSL mean sea level
N_e number of equipotential drops in a flow net
N_f number of flow channels in a flow net
n number of wells in system or group
n_e number of equipotential drops from drainage exit to point
Q rate of flow to a fully penetrating drainage slot per unit length of slot, or rate of seepage flow through area A
Q_D total pump capacity for surface runoff, or rate of flow to a sand drain
Q_e flow to well in an electroosmotic drainage system
Q_p rate of flow to a partially penetrating drainage slot per unit length of slot
Q_R rate of surface runoff
Q_T total flow to a dewatering system
Q_w flow to a well
Q_{wi} flow to ith well
Q_{wj} flow to well j
Q_{wp} flow to a partially penetrating artesian well
q flow per unit length of vertical flow net
R radius of influence of well
$\bar{R}$ distance from well to change from artesian to gravity flow
R_i radius of influence of ith well

R_j radius of influence of well j
r distance from well to point P
r' distance from image well to point P
r_i distance from ith well to point P
r_{ij} distance from well j to well j
r_w radius (effective) of a well
r_{wj} radius (effective) of well j
S_i distance from point P to image well i
S_{ij} distance from well j to image well i
s height of bottom of well above bottom of aquifer
T duration of rainfall
t depth of water in well, or time
V vacuum at pump intake, or volume of sumps
v velocity of flow in riser pipe
W effective depth of penetration of a drainage slot or well into aquifer
W' penetration of a drainage well or slot required to obtain an effective penetration of W in a stratified aquifer
x length of a drainage slot
y distance from drainage slot to a specific line
y_1 distance from riverside toe of cofferdam to effective source of seepage
y_2 distance from landside toe of cofferdam to drainage slot or line of wells
Γ gamma function
λ extra-length coefficient for flow to a partially penetrating drainage slot
θ_e uplift factor for artesian wells or wellpoints
θ_m midpoint uplift factor for artesian wells or wellpoints

REFERENCES

1. Neboline, Ross: Ground Dewatering for Construction, *Engineering News-Record*, vol. 132, p. 479 (Apr. 6, 1944).
2. Lelinvsky, S.: Modern Dewatering Methods in Irrigation Problems, *The Engineer*, July 27, 1956.
3. Terzaghi, K., and R. Peck: *Soil Mechanics in Engineering Practice*, John Wiley & Sons, Inc., New York, 1948.
4. Casagrande, Leo: Electro-osmotic Stabilization of Soils, *Journal of the Boston Society of Civil Engineers*, January, 1952.
5. Ship Construction Basins of Thin Concrete, *Engineering News-Record*, vol. 131, p. 152 (July 15, 1943).
6. Thoresen, S. A.: Shield Driven Tunnels near Completion under the Scheldt at Antwerp, *Engineering News-Record*, vol. 110, p. 827 (June 29, 1933).
7. Tschebotarioff, G.: *Soil Mechanics, Foundations, and Earth Structures*, McGraw-Hill Book Company, Inc., New York, 1951.
8. *The Wellpoint System in Principle and Practice*, Griffin Wellpoint Corporation, New York, 1950.
9. Investigation of Underseepage and Its Control, Lower Mississippi River Levees, *Waterways Experiment Station* TM 3-424, Vicksburg, Miss., October, 1956.
10. Investigation of Underseepage, Mississippi River Levees, Alton to Gale, Ill., *Waterways Experiment Station* TM 3-430, Vicksburg, Miss., April, 1956.
11. Mansur, Charles I.: Laboratory and In-situ Permeability of Sand, *Transactions, American Society of Civil Engineers*, vol. 123, p. 858 (1958).
12. Dupuit, J.: *Études théoriques et pratiques sur le mouvement des eaux*, 1863.

13. Forchheimer, P.: Über die Ergiebigkeit von Brunnen-Anlagen und Sickerschlitzen, *Der Architekten-und Ingenieure-Verein*, Hannover, vol. 32, no. 7 (1886).
14. Chapman, T. G.: Groundwater Flow through a Bank, *Geotechnique, The Institution of Civil Engineers*, London, March, 1957.
15. Barron, R. A.: The Efficacy of Landside Toe Drains in Controlling Foundation Seepage and Uplift, thesis, Massachusetts Institute of Technology, 1952.
16. Chapman, T. G.: Groundwater Flow to Trenches and Wellpoints, *Journal of the Institution of Engineers*, Australia, October–November, 1956.
17. Casagrande, A.: Seepage through Dams, *Contributions to Soil Mechanics, Boston Society of Civil Engineers*, 1940.
18. Taylor, D. W.: *Fundamentals of Soil Mechanics*, John Wiley & Sons, Inc., New York, 1948.
19. Barron, R. A.: Transformations for Flow Net Construction, *Proceedings, Second International Conference on Soil Mechanics*, Rotterdam, vol. 7 (1948).
20. Engelund, F.: On the Theory of Multiple-well Systems, paper dedicated to Prof. A. E. Bretting, Jan. 17, 1958.
21. Muskat, M.: *The Flow of Homogeneous Fluids through Porous Media*, McGraw-Hill Book Company, Inc., New York, 1937.
22. Kozeny, J.: Theorie und Berechnung der Brunnen, *Wasserkraft und Wasserwirtschaft*, Munich, vol. 28 (1933).
23. Babbitt, H. E., and D. H. Caldwell: The Free Surface around and Interference between Gravity Wells, *Engineering Experiment Station Bulletin Series, no. 374*, University of Illinois, Urbana, Ill., 1948.
24. Hall, H. P.: An Investigation of Steady Flow toward a Gravity Well, *La Houille Blanche*, January–February, 1955.
25. Borelli, M.: Free-surface Flow toward Partially Penetrating Wells, *Transactions, American Geophysical Union*, vol. 36, no. 4 (August, 1955).
26. Forchheimer, P.: *Hydraulik*, Teubner Verlags gesellschaft, mbH, Stuttgart, 1930.
27. Dachler, R.: Einige Bemerkungen zur Grundwasserabsenkung mittels Brunnengruppen, *Wasserkraft und Wasserwirtschaft*, Munich, vol. 19, no. 21 (1924).
28. Sichardt, W., and W. Kyrieleis: *Grundwasserabsenkungen bei Fundierungsarbeiten*, Berlin, 1930.
29. Petersen, J., C. Rohwer, and M. Albertson: Effect of Well Screens on Flow into Wells, *Transactions, American Society of Civil Engineers*, vol. 120 (1955).
30. Control of Underseepage by Relief Wells, Trotters, Mississippi, *Waterways Experiment Station* TM 3-341, Vicksburg, Miss., April, 1952.
31. Maclean, D. J., and D. W. Rolfe: Soil Drainage by an Electrical Method, *Civil Engineering*, London, vol. 40 (1945); *Road Abstracts*, vol. 12, no. 207 (1945).
32. Handbook of Culvert and Drainage Practice, Armco Drainage & Metal Products, Inc., Middletown, Ohio, 1948.
33. *General Instructions for the Installation and Operation of Moretrench Pumps and Wellpoint Systems*, Moretrench Corporation, Rockaway, N.J., 1954.

RESUMEN DE CONDICIONES PARA APLICACION DE FORMULAS

SIMBOLOGIA

FUENTE	Circular (C)	FLUJO	Artesiano (A)	CAPTACION	Ranura simple (R1)	PENETRACION	Total
	Lineal simple (L1)		Gravitacional (G)		Ranura doble (R2)		Parcial
	Lineal doble (L2)		Mixto (M)		Hilera de pozos pta. (H)		
					Pozo único (U)		
					Gpo. asismétrico de pzo. (Ga)		
					Gpo. simétrico de pozos (Gs)		

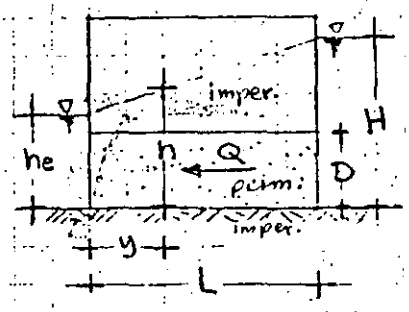
Ranuras

Fórmula No.	Fuente	Flujo	Captación	Penetración
1	L1	A	R1	T
2	"	G	"	"
3	"	M	"	"
4	"	A	"	P
5	"	G	"	"
6	L2	A	"	T
7	"	G	"	"
8	"	M	"	"
9	"	A	"	P
10	"	G	"	"
11	"	A	R2	"
12	"	G	"	"
13	L1	A	H	T
14	"	"	"	P
15	"	G	"	T
16	"	"	"	P

Pozos

Fórmula No.	Fuente	Flujo	Captación	Penetración
17	C	A	U	T
18	"	"	"	P
19	"	G	"	T
20	"	"	"	P
21	"	M	"	T
22	L1	A	U	"
23	"	G	"	"
24	C	A	Ga	"
25	"	G	"	"
26	L1	A	"	"
27	"	G	"	"
28	C	A	Gs	"
29	"	G	"	"
30	L1	A	"	"
31	"	G	"	"

(1) (L1, A, R1, T)



$$Q = \frac{k D x}{L}\left(H - h_c\right) \qquad 1a$$

$$H - h = \frac{L - y}{L}\left(H - h_c\right) \qquad 1b$$

Q = gasto, m^3/seg

K = coef. de permeabilidad, m/seg

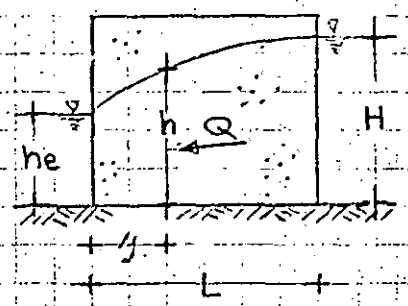
X = ancho del flujo en dirección perpendicular al papel, m = según geometría del problema m.

H, h, he, D, y, L = H

(6) (L2, A, R1, T)

Multiplicar Q de la (1a) por 2

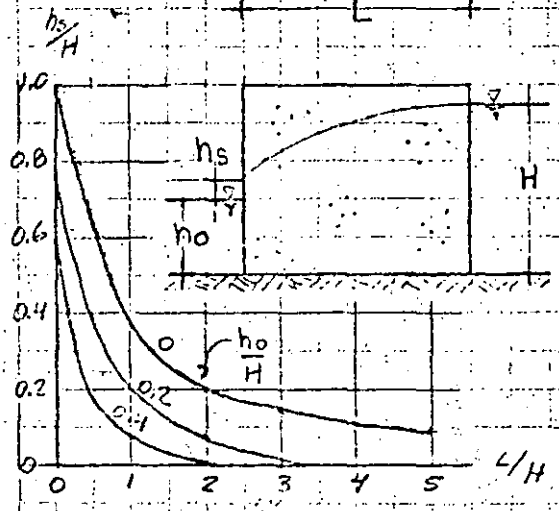
(2) (L1, G, R1, T)



$$Q = \frac{k x}{2L}\left(H^2 - h_c^2\right) \qquad 2a$$

$$H^2 - h^2 = \frac{L - y}{L}\left(H^2 - h_c^2\right) \qquad 2b$$

aplicable para $\frac{L}{H} > 2$ y $\frac{h_e}{H} > 0.3$



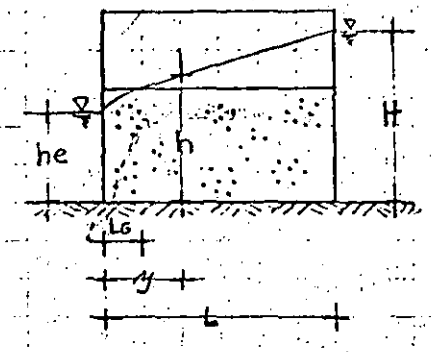
en caso contrario aplicar:

$$H^2 - h^2 = \frac{L - y}{L}\left[H^2 - (h_o + h_s)^2\right] \qquad 2c$$

(7) (L2, G, R1, T)

Multiplicar Q de la (2a) por 2

(3) (L1, M, R1, T)



$$Q = \frac{k x (2DH - D^2 - he^2}{2L} \quad \text{3a}$$

para $y \leq LG$

$$h = \sqrt{\frac{y}{L}(D^2 - he^2) + he^2} \quad \text{3b}$$

para $y \geq LG$

$$h = \left(\frac{H-D}{L-LG}\right)(y - LG) + D \quad \text{3c}$$

y cuando hay corrección por descarga libre:

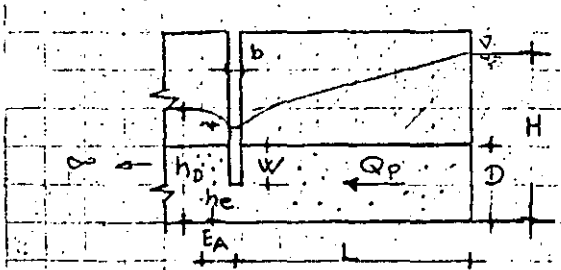
(8) (L2, M, R1, T)

Multiplicar Q de la (3a) por 2.

para $y \leq LG$

$$h = \sqrt{D^2 - \left(\frac{LG - y}{LG}\right)[D^2 - (ho + hs)^2]}$$

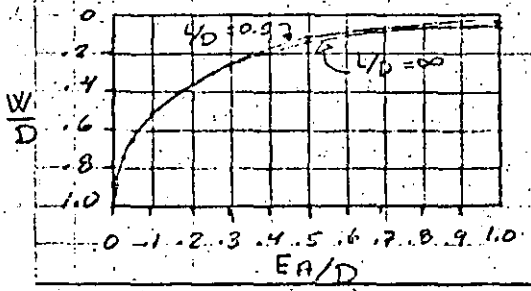
(4) (L1, A, R1, P)



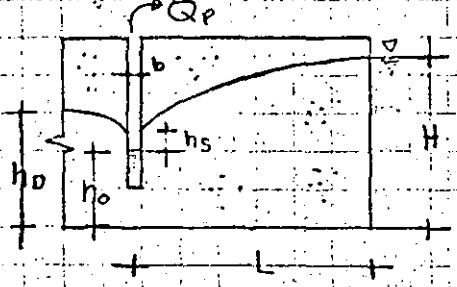
$$Q_p = \frac{k D x (H - he)}{L + E_A} \quad \text{4a}$$

$$h_D = \frac{E_A (H - he)}{L + E_A} + he \quad \text{4b}$$

b, se considera = 0



(5) (L1, G, R1, P)



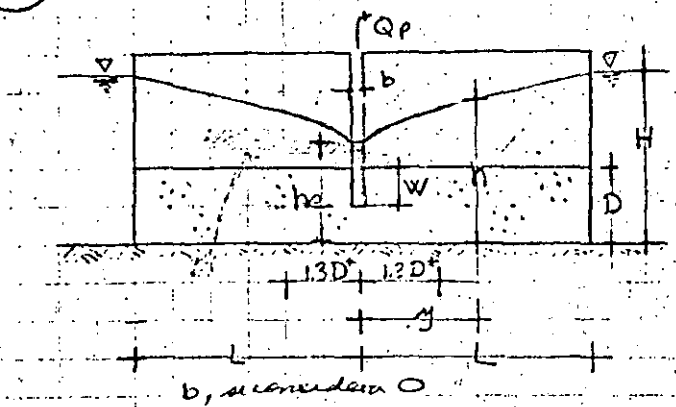
$$Q_p = \left(0.73 + 0.27\,\frac{H - h_o}{H}\right)\frac{kx}{2L}\left(H^2 - h_o^2\right) \quad \text{5a}$$

$$h_D = h_o\left[\frac{1.4B}{L}(H - h_o) + 1\right] \quad \text{5b}$$

ambas ecuaciones son válidas para $L/H > 3$

b, se considera 0

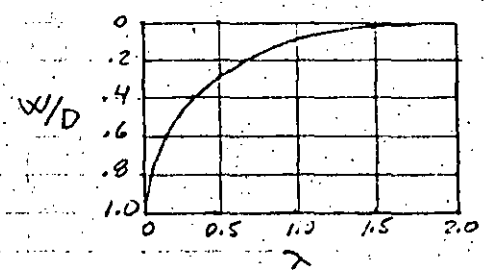
(9) (L2, A, R1, P)



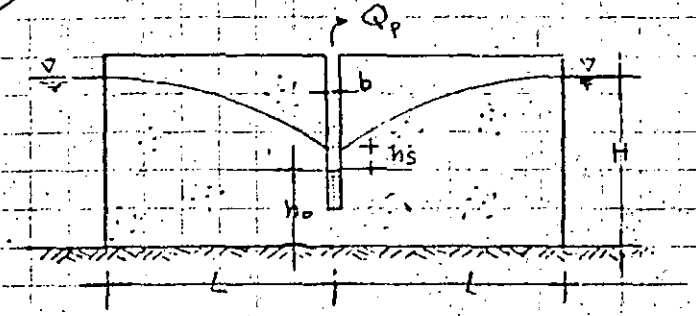
* Dentro del intervalo 1.3D la superficie piezométrica es no lineal

$$Q_p = \frac{2k\,Dx\,(H-he)}{L+\lambda D} \qquad \text{---------- 9a}$$

$$h = he + (H-he)\frac{y+\lambda D}{L+\lambda D} \qquad \text{--- 9b}$$



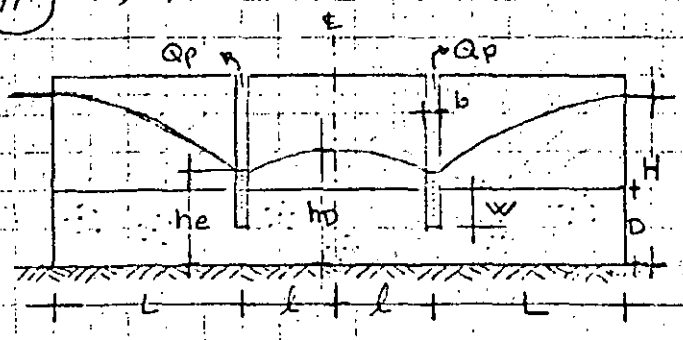
(10) (L2, G, R1, P)



$$Q_p = \left(0.73 + 0.27\frac{H-h_o}{H}\right)\frac{Kx}{L}\left(H^2-h_o^2\right) \qquad \text{--- 10}$$

Ecuación válida para $L/H > 3$

(11) (L2, A, R2, P)

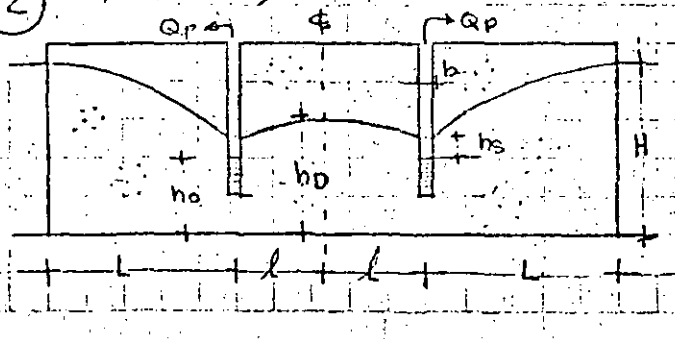


$$Q_p = \frac{K\,Dx\,(H-he)}{L+E_A} \qquad \text{---------- (11a)}$$

$$h_D = \frac{E_A\,(H-he)}{L+E_A} + he \qquad \text{---- (11b)}$$

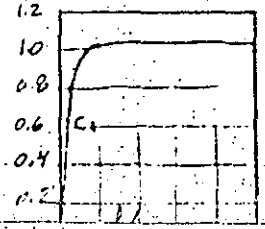
(ver fórmulas 4)

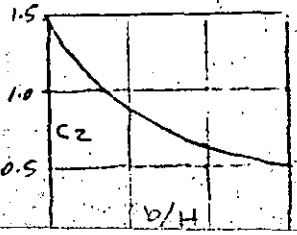
(12) (L2, G, R2, P)



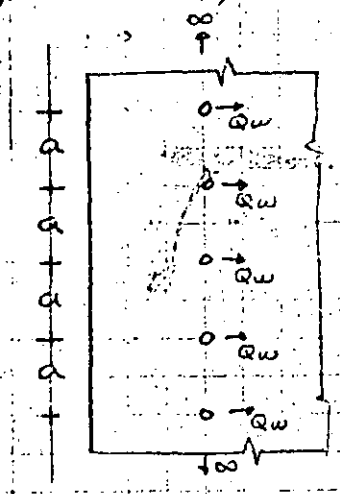
$$Q_p = \left(0.73 + 0.27\frac{H-h_o}{H}\right)\frac{Kx}{2L}\left(H^2-h_o^2\right) \qquad \text{--- (5a)}$$

$$h_D = h_o\left[\frac{C_1 C_2}{L}(H-h_o)+1\right]$$





(13) (L, A, H, T)



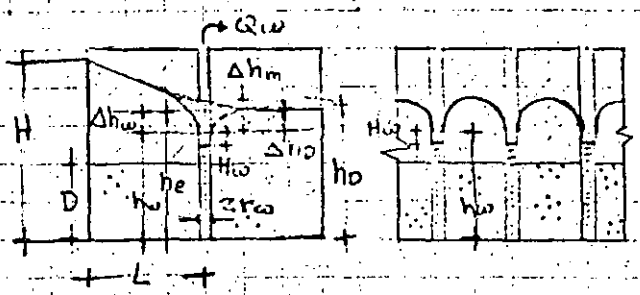
$$H - h_e = \frac{Q_w L}{kDa} \quad \text{(como ranura)} \qquad 13a$$

$$\Delta h_w = \frac{Q_w}{2\pi kD} \ln \frac{a}{2\pi r_w} \text{ (como pozos)} \qquad 13b$$

$$H - h_w = \frac{Q_w}{kD}\left(\frac{L}{a} + \frac{1}{2\pi} \ln \frac{a}{2\pi r_w}\right) \qquad 13c$$

$$\Delta h_m = \frac{Q_w}{2\pi kD} \ln \frac{a}{\pi r_w} \qquad 13d$$

$$H - h_m = \frac{Q_w}{kDa} - 0.11 \frac{Q_w}{kD} \qquad 13d$$



$$\Delta H_D = \frac{Q_w}{2\pi kD} \ln \frac{a}{2\pi r_w} \qquad 13e$$

(14) (L, A, H, P)

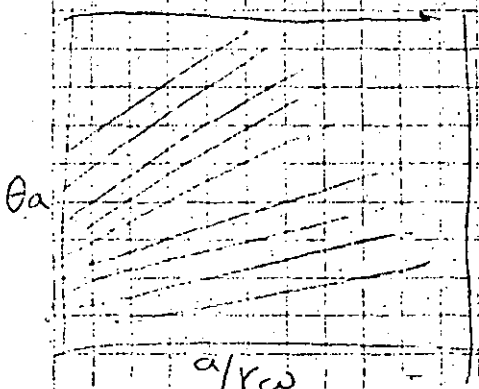
$$\Delta h_w = \frac{Q_w \theta_a}{kD} \qquad 14a$$

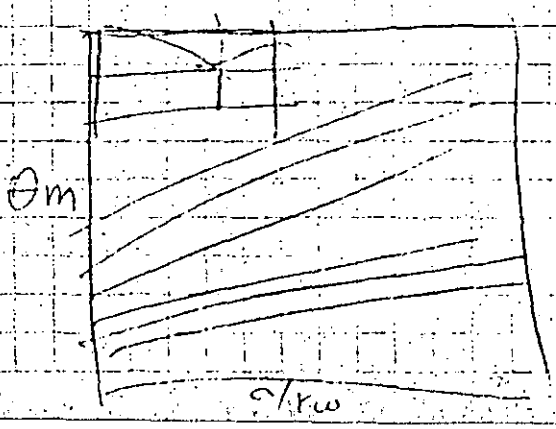
$$H - h_w = \frac{Q_w}{kD}\left(\frac{L}{a} + \theta_a\right) \qquad 14b$$

$$\Delta h_m = \frac{Q_w \theta_m}{kD} \qquad 14c$$

$$H - h_m = \frac{Q_w}{kD}\left(\frac{L}{a} + \theta_a - \theta_m\right) \qquad 14d$$

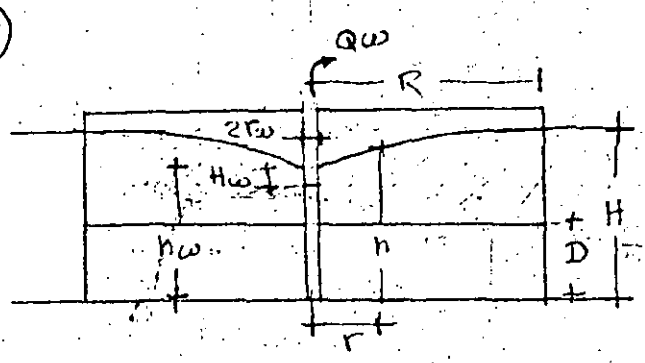
$$\Delta H_D = \frac{Q_w \theta_a}{kD} \qquad 14e$$





17

(17)

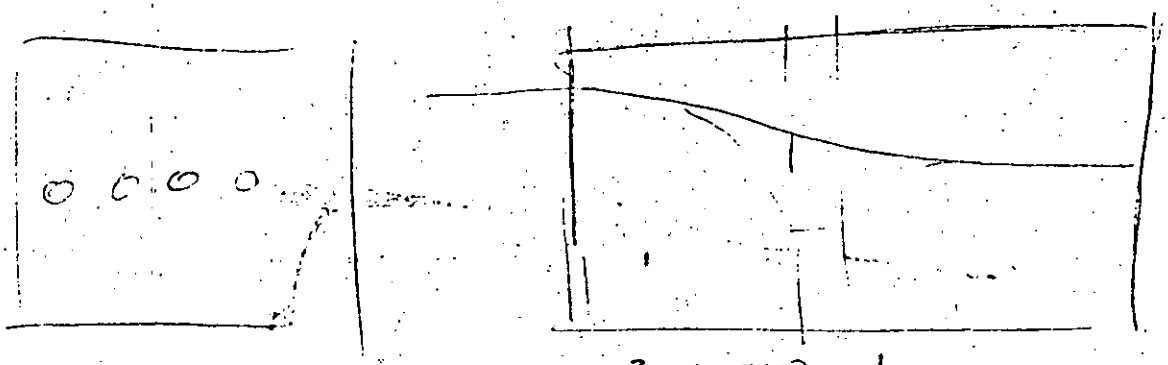


$$Q_w = \frac{2\pi k D (H - h_w)}{\ln (R/r_w)}$$

$$h = \frac{Q_w}{2\pi k D} \ln \frac{r}{r_w} + h_w$$

$$H - h = \frac{Q_w}{2\pi k D} \ln \frac{R}{r}$$

(15) (L1 G 14 1.)



$$H^2 - he^2 = \frac{2Q\omega L}{ka} \qquad \text{15a}$$

$$\Delta h\omega^2 = he^2 - h\omega^2 = \frac{Q\omega}{\pi k} \ln \frac{a}{2\pi r\omega} \qquad \text{15b}$$

$$H^2 - h\omega^2 = \frac{2Q\omega L}{ka} + \frac{Q}{\pi k} \ln \frac{a}{2\pi r\omega}$$

$$\Delta hm^2 = \frac{Q\omega}{\pi k} \ln \frac{a}{\pi r\omega}$$

$$H^2 - hm^2 = H^2 - h\omega^2 - \Delta hm^2$$

$$\Delta h_D^2 = \frac{Q\omega}{\pi k} \ln \frac{a}{2\pi r\omega}$$

$$H^2 - h_D^2 = H^2 - h\omega^2 - \Delta h_D^2$$

DIVISION DE EDUCACION CONTINUA
FACULTAD DE INGENIERIA U.N.A.M.

MICROPROCESADORES Y MICROCOMPUTADORAS

EJEMPLO DE PROGRAMA EN ENSAMBLADORES

ING. ROLANDO CARRERA SANCHEZ

O C T U B R E, 1984

ARCHIVO FUENTE

1

```
                ; ESTE PROGRAMA CALCULA LA SUMA DE UN ARREGLO DE NUMEROS
                ; A CONTINUACION SE EXPLICA EL SIGNIFICADO DE LAS VARIABLES
                ;
                ; LNG = LONGITUD DEL ARREGLO A SUMAR.
                ;
                ; ARGL = DIRECCION INICIAL DEL ARREGLO
                ;
                ; SUMA = DIRECCION DONDE QUEDARA EL RESULTADO DE LA SUMA
                ;
                ; OJO estoy usando la directiva o pseudooperacion EQU
                ; con la que estoy definiendo el tamano del arreglo a 10 bytes
                ; ya que 0AH = 10 decimal.
                ;
LNG:    EQU     0AH
                ;
                ; ahora defino el espacio para almacenar el arreglo
                ; el cual sera igual a la longitud anteriormente declarada.
                ;
ARGL:   DEFS    LNG
                ;
                ; a continuacion declaro 2 bytes de memoria que contendran
                ; el resultado de la suma de los elementos del arreglo.
                ;
SUMA    DEFS    2
                ;
                ; NOTAR que se han omitido los dos puntos (:) de la etiqueta
                ;
                ; YA QUE TENEMOS DEFINIDAS NUESTRAS VARIABLES Y CONSTANTES
                ; COMENZAMOS A ESCRIBIR EL PROGRAMA QUE SUMA EL ARREGLO
                ;
                ;
                ; primero HL tendra la direccion inicial del arreglo
                ;
                ; en B se tendra el numero de elementos a sumar para
                ; controlar el loop con la instruccion DJNZ
                ;
                ; los resultados parciales de la suma se almacenaran
                ; en los registros A y C, siendo A el byte menos
                ; significante y C el mas significante. Por lo tanto
                ; iniciamos los valores de A y C igual a cero
                ;
                ; SE DECLARA LA ETIQUETA GLOBAL PARA QUE PUEDA SER
                ; USADA POR OTROS PROGRAMAS EN DONDE SE DEFINIRA
                ; EXTERNAL LA MISMA ETIQUETA.
```

```
        GLOBAL  SUMARG
        ;
        ;
SUMARG: LD      HL,ARGL         ; DIRECCION DEL ARREGLO
        LD      B,LNG           ; NUMERO DE ELEMENTOS A SUMAR
        SUB     A               ; BRAJO = 0
        LD      C,A             ; BALTO = 0
SUM:    ADD     A,(HL)          ; SUM = SUM + ARGL(HL)
        INC     HL
        JR      NC,SINCY        ; VE SI HUBO CARRY DE LA SUMA DE 8 BITS
        INC     C               ; SUMA EL ACARREO AL BALTO
SINCY   DJNZ    SUM             ; VE SI TODAVIA HAY DATOS PARA SUMAR
        ; SI YA TERMINO LA SUMA GUARDALA EN SUMA
        LD      HL,SUMA
        LD      (HL),A          ; GUARDA EL BRAJO
        INC     HL
        LD      (HL),C          ; GUARDA EL BALTO
        HALT
        END
```

viernes, septiembre 23, 1983
SUM16,L

ARCHIVO ENSAMBLADO

```
                        SUM16                                   PAGE    1
LOC   OBJ CODE M STMT SOURCE STATEMENT                          ASM 5.5

                                3
                          1
                          2             ; ESTE PROGRAMA CALCULA LA SUMA DE UN ARREGLO DE NUMEROS
                          3             ; A CONTINUACION SE EXPLICA EL SIGNIFICADO DE LAS VARIABLES
                          4             ;
                          5             ; LNG = LONGITUD DEL ARREGLO A SUMAR
                          6             ;
                          7             ; ARGL = DIRECCION INICIAL DEL ARREGLO
                          8             ;
                          9             ; SUMA = DIRECCION DONDE QUEDARA EL RESULTADO DE LA SUMA
                         10             ;
                         11             ; OJO estoy usando la directiva o pseudooperacion EQU
                         12             ; con la que estoy definiendo el tamano del arreglo a 10 bytes
                         13             ; ya que OAH = 10 decimal.
                         14  ;
                         15  LNG:   EQU     OAH
                         16  ;
                         17             ; ahora defino el espacio para almacenar el arreglo
                         18             ; el cual sera igual a la longitud anteriormente declarada.
                         19  ;
0000                     20  ARGL:  DEFS    LNG
                         21  ;
                         22             ; a continuacion declaro 2 bytes de memoria que contendran
                         23             ; el resultado de la suma de los elementos del arreglo.
                         24  ;
000A                     25  SUMA    DEFS    2
                         26  ;
                         27             ; NOTAR que se han omitido los dos puntos (:) de la etiqueta
                         28  ;
                         29             ; YA QUE TENEMOS DEFINIDAS NUESTRAS VARIABLES Y CONSTANTES
                         30             ; COMENZAMOS A ESCRIBIR EL PROGRAMA QUE SUMA EL ARREGLO
                         31  ;
                         32  ;
                         33             ; primero HL tendra la direccion inicial del arreglo
                         34  ;
                         35             ; en B se tendra el numero de elementos a sumar para
                         36             ; controlar el loop con la instruccion DJNZ
                         37  ;
                         38             ; los resultados parciales de la suma se almacenaran
                         39             ; en los registros A y C, siendo A el byte menos
                         40             ; significante y C el mas significante. Por lo tanto
                         41             ; iniciamos los valores de A y C igual a cero
                         42  ;
                         43             ; SE DECLARA LA ETIQUETA GLOBAL PARA QUE PUEDA SER
                         44             ; USADA POR OTROS PROGRAMAS EN DONDE SE DEFINIRA
                         45             ; EXTERNAL LA MISMA ETIQUETA.
                         46  ;
```

```
                        46  ;
                        47          GLOBAL  SUMARG
                        48  ;
                        49  ;
0000    210000    R     50  SUMARG: LD      HL,ARGL         ; DIRECCION DEL ARREGLO
0003    060A            51          LD      B,LNG           ; NUMERO DE ELEMENTOS A SUMAR
0005    97              52          SUB     A               ; BBAJO = 0
0006    4F              53          LD      C,A             ; BALTO = 0
0007    86              54  SUM:    ADD     A,(HL)          ; SUM = SUM + ARGL(HL)
0008    23              55          INC     HL
0009    3001            56          JR      NC,SINCY        ; VE SI HUBO CARRY DE LA SUMA DE 8 BITS
000B    0C              57          INC     C               ; SUMA EL ACARREO AL BALTO
000C    10F9            58  SINCY   DJNZ    SUM             ; VE SI TODAVIA HAY DATOS PARA SUMAR
```

4

SUM16 PAGE 2
LOC OBJ CODE M STMT SOURCE STATEMENT ASM 5.8

```
                        59          ; SI YA TERMINO LA SUMA GUARDALA EN SUMA
000E    210A00    R     60          LD      HL,SUMA
0011    77              61          LD      (HL),A          ; GUARDA EL BBAJO
0012    23              62          INC     HL
0013    71              63          LD      (HL),C          ; GUARDA EL BALTO
0014    76              64          HALT
                        65          END
```

Viernes, septiembre 23, 1983
SUM16.MAP
LINK 1.6

ARCHIVO CON LAS DECLARACIONES (MAPA CREADO AL LIGAR EL ARCHIVO ENSAMBLADO)

5

LOAD MAP

MODULE	ORIGIN	LENGTH
SUM16	5000	0021

GLOBAL	ADDRESS	MODULE
SUMARG	500C	SUM16

PROGRAM SUM16 -- 0080 BYTES
ENTRY: 500C

```
;*****************************************************************
; ESTE PROGRAMA ENCUENTRA EL MAYOR VALOR DE UN ARREGLO DE NUMEROS.*
;*****************************************************************
;
; A CONTINUACION SE EXPLICA EL SIGNIFICADO DE LAS VARIABLES
;
; LNG = LONGITUD DEL ARREGLO
;
; ARGL = DIRECCION INICIAL DEL ARREGLO
;
; MAXI = DIRECCION DONDE QUEDARA EL VALOR MAXIMO
;
; OJO estoy usando la directiva o pseudooperacion EQU
; con la que estoy definiendo el tamano del arreglo a 10 bytes
; ya que OAH = 10 decimal.
;
LNG:    EQU     OAH
;
        ; ahora defino el espacio para almacenar el arreglo
        ; el cual sera igual a la longitud anteriormente declarada.
;
ARGL:   DEFS    LNG
;
        ; a continuacion declaro 1 byte de memoria que contendra
        ; el resultado de la busqueda del valor maximo.
;
MAXI    DEFS    1
;
        ; NOTAR que se han omitido los dos puntos (:) de la etiqueta
;
        ; YA QUE TENEMOS DEFINIDAS NUESTRAS VARIABLES Y CONSTANTES
        ; COMENZAMOS A ESCRIBIR EL PROGRAMA BUSCA EL MAXIMO
;
;
        ; primero HL tendra la direccion inicial del arreglo
;
        ; en B se tendra el numero de elementos a comparar
        ; para controlar el loop con la instruccion DJNZ.
;
        ; el resultado parcial se almacenara en el registro A
        ; iniciamos el valor maximo como 0 en el registro A
;
        ; SE DECLARA LA ETIQUETA GLOBAL PARA QUE PUEDA SER
        ; USADA POR OTROS PROGRAMAS EN DONDE SE DEFINIRA
        ; EXTERNAL LA MISMA ETIQUETA.
;
        GLOBAL  MAXARG
;
```

```
;
MAXARR: LD      HL,ARGL         ; DIRECCION DEL ARREGLO
        LD      B,LNG           ; TAMANO DEL ARREGLO
        SUB     A               ; MAXIMO = 0
ESMAX:  CP      (HL)            ; ES EL ELEMENTO ARGL(HL) > MAXI(A)
        JR      NC,SIGMAX       ; VE SI NO HUBO CARRY DE LA COMPARACION
        LD      A,(HL)          ; ACTUALIZA EL MAXIMO MAXI = ARGL(HL)
SIGMAX  INC     HL              ; SIGUIENTE ELEMENTO DEL ARREGLO
        DJNZ    ESMAX           ; VE SI TODAVIA HAY DATOS
        ;
        ; SI YA TERMINO LA BUSQUEDA GUARDA EL MAXIMO EN MAXI
        ;
        LD      HL,MAXI
        LD      (HL),A          ; GUARDA EL MAXIMO

        HALT
        END
```

```
                        1
                        2          ; ***********************************************************
                        3          ; ESTE PROGRAMA ENCUENTRA EL MAYOR VALOR DE UN ARREGLO DE NUME
                        4          ; ***********************************************************
                        5          ;
                        6          ; A CONTINUACION SE EXPLICA EL SIGNIFICADO DE LAS VARIABLES
                        7          ;
                        8          ; LNG = LONGITUD DEL ARREGLO
                        9          ;
                       10          ; ARGL = DIRECCION INICIAL DEL ARREGLO
                       11          ;
                       12          ; MAXI = DIRECCION DONDE QUEDARA EL VALOR MAXIMO
                       13          ;
                       14          ; OJO estoy usando la directiva o pseudooperacion EQU
                       15          ; con la que estoy definiendo el tamano del arreglo a 10 but
                       16          ; ya que 0AH = 10 decimal.
                       17  ;
                       18  LNG:    EQU     0AH
                       19  ;
                       20          ; ahora defino el espacio para almacenar el arreglo
                       21          ; el cual sera igual a la longitud anteriormente declarada.
                       22  ;
0000                   23  ARGL:   DEFS    LNG
                       24  ;
                       25          ; a continuacion declaro 1 byte de memoria que contendra
                       26          ; el resultado de la busqueda del valor maximo.
                       27  ;
000A                   28  MAXI    DEFS    1
                       29  ;
                       30          ; NOTAR que se han omitido los dos puntos (:) de la etiqueta
                       31  ;
                       32          ; YA QUE TENEMOS DEFINIDAS NUESTRAS VARIABLES Y CONSTANTES
                       33          ; COMENZAMOS A ESCRIBIR EL PROGRAMA BUSCA EL MAXIMO
                       34  ;
                       35  ;
                       36          ; primero HL tendra la direccion inicial del arreglo
                       37  ;
                       38          ; en B se tendra el numero de elementos a comparar
                       39          ; para controlar el loop con la instruccion DJNZ.
                       40  ;
                       41          ; el resultado parcial se almacenara en el registro A
                       42          ; iniciamos el valor maximo como 0 en el registro A
                       43  ;
                       44          ; SE DECLARA LA ETIQUETA GLOBAL PARA QUE PUEDA SER
                       45          ; USADA POR OTROS PROGRAMAS EN DONDE SE DEFINIRA
                       46          ; EXTERNAL LA MISMA ETIQUETA.
                       47  ;
                       48          GLOBAL  MAXARG
                       49  ;
```

```
                        50  ;
000B  210000    R       51  MAXARG: LD      HL,ARGL         ; DIRECCION DEL ARREGLO
000E  060A              52          LD      B,LNG           ; TAMANO DEL ARREGLO
0010  97                53          SUB     A               ; MAXIMO = 0
0011  BE                54  ESMAX:  CP      (HL)            ; ES EL ELEMENTO ARG(HL) > MAXI(A
0012  3001              55          JR      NC,SIGMAX       ; VE SI NO HUBO CARRY DE LA COMPARACION
0014  7E                56          LD      A,(HL)          ; ACTUALIZA EL MAXIMO MAXI = ARG(HL)
0015  23                57  SIGMAX: INC     HL              ; SIGUIENTE ELEMENTO DEL ARREGLO
0016  10F9              58          DJNZ    ESMAX           ; VE SI TODAVIA HAY DATOS
```

8

9

```
                                MAXI                                    PAGE  2
LOC   OBJ CODE M STMT SOURCE STATEMENT                                   ASM 5.2

                        59          ;
                        60          ; SI YA TERMINO LA BUSQUEDA GUARDA EL MAXIMO EN MAXI
                        61          ;
0018  210A00    R       62          LD      HL,MAXI
001B  77                63          LD      (HL),A          ; GUARDA EL MAXIMO
001C  76                64          HALT
                        65          END
```

```
OCH
viernes, septiembre 23, 1983
MAXI.MAP
LINK 1.6

LOAD MAP
MODULE      ORIGIN LENGTH

MAXI          5000   001D

GLOBAL      ADDRESS   MODULE

MAXARG        500B    MAXI

PROGRAM MAXI -- 0080 BYTES
ENTRY: 500B
```

MAPA DE MEMORIA DONDE SE CARGARÁ EL PROGRAMA OBJETO

10

```
;
        ; SE DEFINE UN MACRO PARA HACER LA COMPARACION DE
        ; DOS BLOQUES DE MEMORIA
;
COMPARA MACRO   #BLQ1,#BLQ2,#NBYTES
        COND    .NOT.('#NBYTES'='') ; ESTAN LOS 3 PARAMETROS??
                                    ; ENTONCES ENSAMBLA.
        PUSH    HL                  ; SALVA LOS REGISTROS EN EL STACK
        PUSH    DE
        PUSH    BC
        PUSH    AF
        LD      HL,#BLQ1            ;CARGA LOS REGISTROS CON LOS PARAMETROS
        LD      DE,#BLQ2
        LD      C,#BYTES
        CALL    COMPR
        POP     AF
        POP     BC
        POP     DE
        POP     HL

        COND    PRIEXP  ; ENSAMBLA SOLO SI ES LA PRIMERA VEZ QUE SE
                        ; LLAMA A ESTE MACRO.
        JR      FCP$$YM
COMPR   LD      A,(DE)              ; TRAE UN CARACTER
        CP      (HL)                ; SON IGUALES
        RET     NZ                  ; TERMINA SI SON DIFERENTES
        INC     HL
        INC     DE
        DJNZ    COMPR               ; TERMINO DE COMPARAR??
        RET                         ;

PRIEXP  DEFL    0
        ENDC    ; ES LA PRIMERA EXPANCION DE MACRO???
        ENDC    ; '#NBYTES'=''
FCP$$YM ENDM

; TERMINO LA DEFINICION DEL MACRO Y EMPIEZA EL PROGRAMA DE EJEMPLO
;
;

        GLOBAL  EJEMPLO
PRIEXP  DEFL    1       ; BANDERA PARA MANEJAR LA EXPANCION CONDICIONAL
EJEMPLO COMPARA 5000H 6000H 0AH
        COMPARA 5000H 6000H 0BH
        HALT
        END
viernes, septiembre 23, 1983
COMPARA.L
```

10

```
                        1 ;
                        2 ;
                        3         ; SE DEFINE UN MACRO PARA HACER LA COMPARACION DE
                        4         ; DOS BLOQUES DE MEMORIA.
                        5 ;
                        6 COMPARA MACRO   #BLQ1,#BLQ2,#NBYTES
                        7         COND    .NOT.('#NBYTES'='')  ; ESTAN LOS 3 PARAMETROS??
                        8                                      ; ENTONCES ENSAMBLA.
                        9         PUSH    HL                   ; SALVA LOS REGISTROS EN EL STACK
                       10         PUSH    DE
                       11         PUSH    BC
                       12         PUSH    AF
                       13         LD      HL,#BLQ1             ;CARGA LOS REGISTROS CON LOS PARAMETROS
                       14         LD      DE,#BLQ2
                       15         LD      C,#BYTES
*** MACRO DEFINITION ERROR ***
                       16         CALL    COMPR
                       17         POP     AF
                       18         POP     BC
                       19         POP     DE
                       20         POP     HL
                       21 ;
                       22         COND    PRIEXP  ; ENSAMBLA SOLO SI ES LA PRIMERA VEZ QUE SE
                       23                         ; LLAMA A ESTE MACRO.
                       24         JR      FCP$$YM
                       25 COMPR   LD      A,(DE)               ; TRAE UN CARACTER
                       26         CP      (HL)                 ; SON IGUALES
                       27         RET     NZ                   ; TERMINA SI SON DIFERENTES
                       28         INC     HL
                       29         INC     DE
                       30         DJNZ    COMPR                ; TERMINO DE COMPARAR??
                       31         RET                          ;
                       32 ;
                       33 PRIEXP  DEFL    0
                       34         ENDC    ; ES LA PRIMERA EXPANCION DE MACRO???
                       35         ENDC    ; '#NBYTES'=''
                       36 FCP$$YM ENDM
                       37 ;
                       38 ; TERMINO LA DEFINICION DEL MACRO Y EMPIEZA EL PROGRAMA DE EJEMPLO
                       39 ;
                       40 ;
                       41 ;
                       42         GLOBAL  EJEMPLO
                       43 PRIEXP  DEFL    1        ; BANDERA PARA MANEJAR LA EXPANCION CONDICIONAL
                       44 EJEMPLO COMPARA 5000H 6000H 0AH
                                  COND    .NOT.('0AH'='')  ; ESTAN LOS 3 PARAMETROS??
                                                           ; ENTONCES ENSAMBLA.
0000   E5                          PUSH    HL               ; SALVA LOS REGISTROS EN EL STACK
0001   D5                          PUSH    DE
0002   C5                          PUSH    BC
0003   F5                          PUSH    AF
0004   210050                      LD      HL,5000H         ;CARGA LOS REGISTROS CON LOS PARAMETROS
0007   110060                      LD      DE,6000H
000A   CD1300   R                  CALL    COMPR
000D   F1                          POP     AF
000E   C1                          POP     BC
000F   D1                          POP     DE
0010   E1                          POP     HL
```

; SE OMITIÓ LA N DE LA ETIQUETA DEL TERCER PARAMETRO ≠ NBYTES

```
                                COND    PRIEXP  ; ENSAMBLA SOLO SI ES LA PRIMERA VEZ QUE SE
                                                ; LLAMA A ESTE MACRO.
0011    1808                    JR      FCP0000
0013    1A              COMPR   LD      A,(DE)          ; TRAE UN CARACTER
0014    BE                      CP      (HL)            ; SON IGUALES
0015    C0                      RET     NZ              ; TERMINA SI SON DIFERENTES
0016    23                      INC     HL
0017    13                      INC     DE
0018    10F9                    DJNZ    COMPR           ; TERMINO DE COMPARAR??
001A    C9                      RET                     ;

                        PRIEXP  DEFL    0
                                ENDC    ; ES LA PRIMERA EXPANCION DE MACRO???
                                ENDC    ; '0AH'='
                        FCP0000
                    45          COMPARA 5000H 6000H 0AH
                                COND    .NOT.('0AH'='') ; ESTAN LOS 3 PARAMETROS??
                                                        ; ENTONCES ENSAMBLA.
001B    E5                      PUSH    HL              ; SALVA LOS REGISTROS EN EL STACK
001C    D5                      PUSH    DE
001D    C5                      PUSH    BC
001E    F5                      PUSH    AF
001F    215050                  LD      HL,5000H        ;CARGA LOS REGISTROS CON LOS PARAMETROS
0022    116060                  LD      DE,6000H
0025    CD1300  R               CALL    COMPR
0028    F1                      POP     AF
0029    C1                      POP     BC
002A    D1                      POP     DE
002B    E1                      POP     HL

                                COND    PRIEXP  ; ENSAMBLA SOLO SI ES LA PRIMERA VEZ QUE SE
                                                ; LLAMA A ESTE MACRO.
                                JR      FCP0001
                        COMPR   LD      A,(DE)          ; TRAE UN CARACTER
                                CP      (HL)            ; SON IGUALES
                                RET     NZ              ; TERMINA SI SON DIFERENTES
                                INC     HL
                                INC     DE
                                DJNZ    COMPR           ; TERMINO DE COMPARAR??
                                RET                     ;

                        PRIEXP  DEFL    0
                                ENDC    ; ES LA PRIMERA EXPANCION DE MACRO???
                                ENDC    ; '0AH'='
                        FCP0001
*** MULTIPLE DECLARATION ***
002C    76                  46          HALT
                            47          END
```

* Como el ensamblador sólo toma los primeros 6 caracteres, cree que {FCP000|0} son iguales {FCP000|1}

** y protesta.

```
;
        ; SE DEFINE UN MACRO PARA HACER LA COMPARACION DE
        ; DOS BLOQUES DE MEMORIA
;
COMPARA MACRO   #BLQ1,#BLQ2,#NBYTES
        COND    .NOT.('#NBYTES'='') ; ESTAN LOS 3 PARAMETROS??
                                    ; ENTONCES ENSAMBLA.
        PUSH    HL                  ; SALVA LOS REGISTROS EN EL STACK
        PUSH    DE
        PUSH    BC
        PUSH    AF
        LD      HL,#BLQ1            ;CARGA LOS REGISTROS CON LOS PARAMETROS
        LD      DE,#BLQ2
        LD      C,#NBYTES
        CALL    COMPR
        POP     AF
        POP     BC
        POP     DE
        POP     HL

        COND    PRIEXP  ; ENSAMBLA SOLO SI ES LA PRIMERA VEZ QUE SE
                        ; LLAMA A ESTE MACRO.
        JR      FC#$YM
COMPR   LD      A,(DE)              ; TRAE UN CARACTER
        CP      (HL)                ; SON IGUALES
        RET     NZ                  ; TERMINA SI SON DIFERENTES
        INC     HL
        INC     DE
        DJNZ    COMPR               ; TERMINO DE COMPARAR??
        RET                         ;

PRIEXP  DEFL    0
        ENDC    ; ES LA PRIMERA EXPANCION DE MACRO???
        ENDC    ; '#NBYTES'=''
FC#$YM  ENDM

; TERMINO LA DEFINICION DEL MACRO Y EMPIEZA EL PROGRAMA DE EJEMPLO
;
;

        GLOBAL  EJEMPLO
PRIEXP  DEFL    1       ; BANDERA PARA MANEJAR LA EXPANCION CONDICIONAL
EJEMPLO COMPARA 5000H 6000H 0AH
        COMPARA 5000H 6000H 0AH
        HALT
        END
```

13

```
                        1 ;
                        2 ;
                        3         ; SE DEFINE UN MACRO PARA HACER LA COMPARACION DE
                        4         ; DOS BLOQUES DE MEMORIA
                        5 ;
                        6 COMPARA MACRO   #BLQ1,#BLQ2,#NBYTES
                        7         COND    .NOT.('#NBYTES'='') ; ESTAN LOS 3 PARAMETROS??
                        8                                     ; ENTONCES ENSAMBLA.
                        9         PUSH    HL                  ; SALVA LOS REGISTROS EN EL STACK
                       10         PUSH    DE
                       11         PUSH    BC
                       12         PUSH    AF
                       13         LD      HL,#BLQ1            ;CARGA LOS REGISTROS CON LOS PARAMETROS
                       14         LD      DE,#BLQ2
                       15         LD      C,#NBYTES
                       16         CALL    COMPR
                       17         POP     AF
                       18         POP     BC
                       19         POP     DE
                       20         POP     HL
                       21 ;
                       22         COND    PRIEXP  ; ENSAMBLA SOLO SI ES LA PRIMERA VEZ QUE SE
                       23                         ; LLAMA A ESTE MACRO.
                       24         JR      FC#$YM
                       25 COMPR   LD      A,(DE)              ; TRAE UN CARACTER
                       26         CP      (HL)                ; SON IGUALES
                       27         RET     NZ                  ; TERMINA SI SON DIFERENTES
                       28         INC     HL
                       29         INC     DE
                       30         DJNZ    COMPR               ; TERMINO DE COMPARAR??
                       31         RET                         ;
                       32 ;
                       33 PRIEXP  DEFL    0
                       34         ENDC    ; ES LA PRIMERA EXPANCION DE MACRO???
                       35         ENDC    ; '#NBYTES'=''
                       36 FC#$YM  ENDM
                       37 ;
                       38 ; TERMINO LA DEFINICION DEL MACRO Y EMPIEZA EL PROGRAMA DE EJEMPLO
                       39 ;
                       40 ;
                       41
                       42         GLOBAL  EJEMPLO
                       43 PRIEXP  DEFL    1        ; BANDERA PARA MANEJAR LA EXPANCION CONDICIONAL
                       44 EJEMPLO COMPARA 5000H,6000H,0AH
                                  COND    .NOT.('0AH'='') ; ESTAN LOS 3 PARAMETROS??
                                                          ; ENTONCES ENSAMBLA.
0000  E5                                   PUSH    HL                  ; SALVA LOS REGISTROS EN EL STACK
0001  D5                                   PUSH    DE
0002  C5                                   PUSH    BC
0003  F5                                   PUSH    AF
0004  210050                               LD      HL,5000H            ;CARGA LOS REGISTROS CON LOS PARAMETROS
0007  116060                               LD      DE,6000H
000A  0E0A                                 LD      C,0AH
000C  CD1500    R                          CALL    COMPR
000F  F1                                   POP     AF
0010  C1                                   POP     BC
0011  D1                                   POP     DE
0012  E1                                   POP     HL
```

```
                                COND    PRIEXP  ; ENSAMBLA SOLO SI ES LA PRIMERA VEZ QUE SE
                                                ; LLAMA A ESTE MACRO.
0013    1808                    JR      FC0000
0015    1A              COMPR   LD      A,(DE)          ; TRAE UN CARACTER
0016    BE                      CP      (HL)            ; SON IGUALES
0017    C0                      RET     NZ              ; TERMINA SI SON DIFERENTES
0018    23                      INC     HL
0019    13                      INC     DE
001A    10F9                    DJNZ    COMPR           ; TERMINO DE COMPARAR??
001C    C9                      RET

                        PRIEXP  DEFL    0
                                ENDC    ; ES LA PRIMERA EXPANCION DE MACRO???
                                ENDC    ; '0AH'=''
                        FC0000
                    45          COMPARA 5000H 6000H 0AH
                                COND    .NOT.('0AH'='')  ; ESTAN LOS 3 PARAMETROS??
                                                         ; ENTONCES ENSAMBLA.
001D    E5                      PUSH    HL              ; SALVA LOS REGISTROS EN EL STACK
001E    D5                      PUSH    DE
001F    C5                      PUSH    BC
0020    F5                      PUSH    AF
0021    210050                  LD      HL,5000H        ;CARGA LOS REGISTROS CON LOS PARAMETROS
0024    110060                  LD      DE,6000H
0027    0E0A                    LD      C,0AH
0029    CD1500      R           CALL    COMPR
002C    F1                      POP     AF
002D    C1                      POP     BC
002E    D1                      POP     DE
002F    E1                      POP     HL

                                COND    PRIEXP  ; ENSAMBLA SOLO SI ES LA PRIMERA VEZ QUE SE
                                                ; LLAMA A ESTE MACRO.
                                JR      FC0001
                        COMPR   LD      A,(DE)          ; TRAE UN CARACTER
                                CP      (HL)            ; SON IGUALES
                                RET     NZ              ; TERMINA SI SON DIFERENTES
                                INC     HL
                                INC     DE
                                DJNZ    COMPR           ; TERMINO DE COMPARAR??
                                RET

                        PRIEXP  DEFL    0
                                ENDC    ; ES LA PRIMERA EXPANCION DE MACRO???
                                ENDC    ; '0AH'=''
                        FC0001
0030    76                  46          HALT
                            47          END
```

OBSERVAR COMO LA RUTINA * SOLO SE EXPANDIÓ EN LA PRIMERA LLAMADA AL MACRO, PERO YA NO EN LA SEGUNDA (#).

15

```
                    1
                    2 ;
                    3         ; SE DEFINE UN MACRO PARA HACER LA COMPARACION DE
                    4         ; DOS BLOQUES DE MEMORIA
                    5 ;
                    6 COMPARA MACRO   #BL01,#BL02,#NBYTES
                    7         COND    .NOT.('#NBYTES'='')  ; ESTAN LOS 3 PARAMETROS??
                    8                                      ; ENTONCES ENSAMBLA.
                    9         PUSH    HL                   ; SALVA LOS REGISTROS EN EL STACK
                   10         PUSH    DE
                   11         PUSH    BC
                   12         PUSH    AF
                   13         LD      HL,#BL01             ;CARGA LOS REGISTROS CON LOS PARAMETRO
                   14         LD      DE,#BL02
                   15         LD      C,#NBYTES
                   16         CALL    COMPR
                   17         POP     AF
                   18         POP     BC
                   19         POP     DE
                   20         POP     HL
                   21
                   22         COND    PRIEXP  ; ENSAMBLA SOLO SI ES LA PRIMERA VEZ QUE SE
                   23                         ; LLAMA A ESTE MACRO.
                   24         JR      FC#$YM
                   25 COMPR   LD      A,(DE)               ; TRAE UN CARACTER
                   26         CP      (HL)                 ; SON IGUALES
                   27         RET     NZ                   ; TERMINA SI SON DIFERENTES
                   28         INC     HL
                   29         INC     DE
                   30         DJNZ    COMPR                ; TERMINO DE COMPARAR??
                   31         RET                          ;
                   32
                   33 PRIEXP  DEFL    0
                   34         ENDC    ; ES LA PRIMERA EXPANCION DE MACRO???
                   35         ENDC    ; '#NBYTES'=''
                   36 FC#$YM  ENDM
                   37
                   38 ; TERMINO LA DEFINICION DEL MACRO Y EMPIEZA EL PROGRAMA DE EJEMPLO
                   39 ;
                   40 ;
                   41
                   42         GLOBAL  EJEMPLO
                   43 PRIEXP  DEFL    1       ; BANDERA PARA MANEJAR LA EXPANCION CONDICIONA
                   44 EJEMPLO COMPARA 5000H,6000H
                              COND    .NOT.(''='') ; ESTAN LOS 3 PARAMETROS??
                                                   ; ENTONCES ENSAMBLA.
                              PUSH    HL           ; SALVA LOS REGISTROS EN EL STACK
                              PUSH    DE
                              PUSH    BC
                              PUSH    AF
                              LD      HL,5000H     ;CARGA LOS REGISTROS CON LOS PARAMETRO
                              LD      DE,6000H
                              LD      C,
                              CALL    COMPR
                              POP     AF
                              POP     BC
                              POP     DE
                              POP     HL
                              JR      FC''$YM
```

16

```
                                COND    PRIEXP  ; ENSAMBLA SOLO SI ES LA PRIMERA VEZ QUE S
                                                ; LLAMA A ESTE MACRO.
                                JR      FC0000
                        COMPR   LD      A,(DE)          ; TRAE UN CARACTER
                                CP      (HL)            ; SON IGUALES
                                RET     NZ              ; TERMINA SI SON DIFERENTES
                                INC     HL
                                INC     DE
                                DJNZ    COMPR           ; TERMINO DE COMPARAR??
                                RET                     ;

                        PRIEXP  DEFL    0
                                ENDC    ; ES LA PRIMERA EXPANCION DE MACRO???
                                ENDC    ; ''=''
                        FC0000
                    45          COMPARA 5000H 6000H 0AH
                                COND    .NOT.('0AH'='')  ; ESTAN LOS 3 PARAMETROS??
                                                         ; ENTONCES ENSAMBLA.
0000    E5                              PUSH    HL              ; SALVA LOS REGISTROS EN EL STACK
0001    D5                              PUSH    DE
0002    C5                              PUSH    BC
0003    F5                              PUSH    AF
0004    210050                          LD      HL,5000H        ;CARGA LOS REGISTROS CON LOS PARA
0007    116060                          LD      DE,6000H
000A    0E0A                            LD      C,0AH
000C    CD1500  R                       CALL    COMPR
000F    F1                              POP     AF
0010    C1                              POP     BC
0011    D1                              POP     DE
0012    E1                              POP     HL

                                COND    PRIEXP  ; ENSAMBLA SOLO SI ES LA PRIMERA VEZ QUE
                                                ; LLAMA A ESTE MACRO.
0013    1808                            JR      FC0001
0015    1A                      COMPR   LD      A,(DE)          ; TRAE UN CARACTER
0016    BE                              CP      (HL)            ; SON IGUALES
0017    C0                              RET     NZ              ; TERMINA SI SON DIFERENTES
0018    23                              INC     HL
0019    13                              INC     DE
001A    10F9                            DJNZ    COMPR           ; TERMINO DE COMPARAR??
001C    C9                              RET                     ;

                        PRIEXP  DEFL    0
                                ENDC    ; ES LA PRIMERA EXPANCION DE MACRO???
                                ENDC    ; '0AH'=''
                        FC0001
001D    76              46              HALT
                        47              END
```

* OBSERVAR COMO EN * AL FALTAR UNO DE LOS PARAMETROS EN LA LLAMADA AL MACRO NO ENSAMBLA NADA POR LA CONDICIÓN DE ENSAMBLADO EN #

DIVISION DE EDUCACION CONTINUA
FACULTAD DE INGENIERIA U.N.A.M.

MICROPROCESADORES Y MICROCOMPUTADORAS

Z80 - CPU
Z80A- CPU
TECHNICAL MANUAL

NOVIEMBRE 1984

Palacio de Mineria Calle de Tacuba 5 primer piso Deleg. Cuauhtémoc 06000 México, D.F. Tel.: 521-40-20 Apdo. Postal M-2285

1.0 INTRODUCTION

The term "microcomputer" has been used to describe virtually every type of small computing device designed within the last few years. This term has been applied to everything from simple "microprogrammed" controllers constructed out of TTL MSI up to low end minicomputers with a portion of the CPU constructed out of TTL LSI "bit slices." However, the major impact of the LSI technology within the last few years has been with MOS LSI. With this technology, it is possible to fabricate complete and very powerful computer systems with only a few MOS LSI components.

The Zilog Z-80 family of components is a significant advancement in the state-of-the art of microcomputers. These components can be configured with any type of standard semiconductor memory to generate computer systems with an extremely wide range of capabilities. For example, as few as two LSI circuits and three standard TTL MSI packages can be combined to form a simple controller. With additional memory and I/O devices a computer can be constructed with capabilities that only a minicomputer could previously deliver. This wide range of computational power allows standard modules to be constructed by a user that can satisfy the requirements of an extremely wide range of applications.

The major reason for MOS LSI domination of the microcomputer market is the low cost of these few LSI components. For example, MOS LSI microcomputers have already replaced TTL logic in such applications as terminal controllers, peripheral device controllers, traffic signal controllers, point of sale terminals, intelligent terminals and test systems. In fact the MOS LSI microcomputer is finding its way into almost every product that now uses electronics and it is even replacing many mechanical systems such as weight scales and automobile controls.

The MOS LSI microcomputer market is already well established and new products using them are being developed at an extraordinary rate. The Zilog Z-80 component set has been designed to fit into this market through the following factors:

1. The Z-80 is fully software compatible with the popular 8080A CPU offered from several sources. Existing designs can be easily converted to include the Z-80 as a superior alternative.
2. The Z-80 component set is superior in both software and hardware capabilities to any other microcomputer system on the market. These capabilities provide the user with significantly lower hardware and software development costs while also allowing him to offer additional features in his system.
3. For increased throughput the Z80A operating at a 4 MHZ clock rate offers the user significant speed advantages over competitive products.
4. A complete product line including full software support with strong emphasis on high level languages and a disk-based development system with advanced real-time debug capabilities is offered to enable the user to easily develop new products.

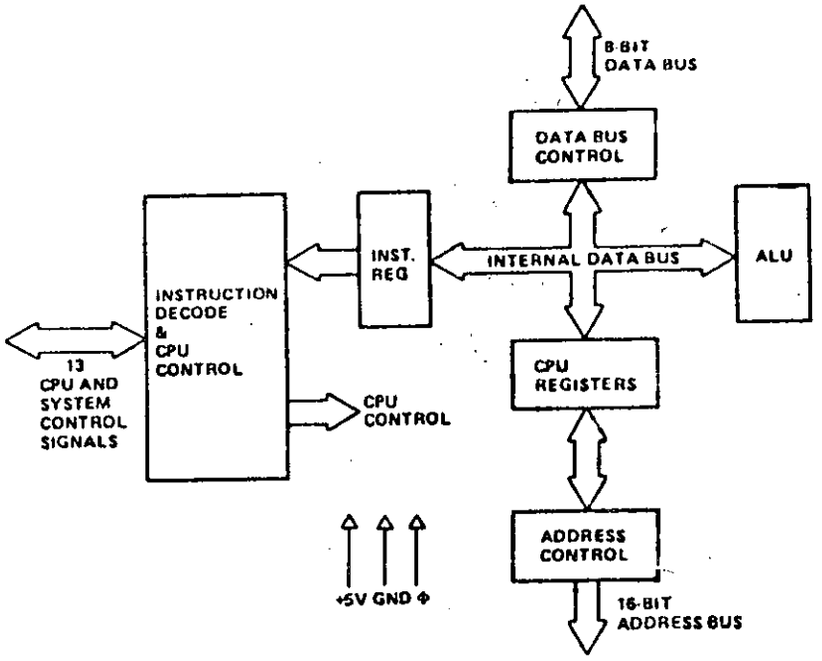
Microcomputer systems are extremely simple to construct using Z-80 components. Any such system consists of three parts:

1. CPU (Central Processing Unit)
2. Memory
3. Interface Circuits to peripheral devices

The CPU is the heart of the system. Its function is to obtain instructions from the memory and perform the desired operations. The memory is used to contain instructions and in most cases data that is to be processed. For example, a typical instruction sequence may be to read data from a specific peripheral device, store it in a location in memory, check the parity and write it out to another peripheral device. Note that the Zilog component set includes the CPU and various general purpose I/O device controllers, while a wide range of memory devices may be used from any source. Thus, all required components can be connected together in a very simple manner with virtually no other external logic. The user's effort then becomes primarily one of software development. That is, the user can concentrate on describing his problem and translating it into a series of instructions that can be loaded into the microcomputer memory. Zilog is dedicated to making this step of software generation as simple as possible. A good example of this is our assembly language in which a simple mnemonic is used to represent every instruction that the CPU can perform. This language is self documenting in such a way that from the mnemonic the user can understand exactly what the instruction is doing without constantly checking back to a complex cross listing.

2.0 Z-80 CPU ARCHITECTURE

A block diagram of the internal architecture of the Z-80 CPU is shown in figure 2.0-1. The diagram shows all of the major elements in the CPU and it should be referred to throughout the following description.



Z-80 CPU BLOCK DIAGRAM
FIGURE 2.0-1

2.1 CPU REGISTERS

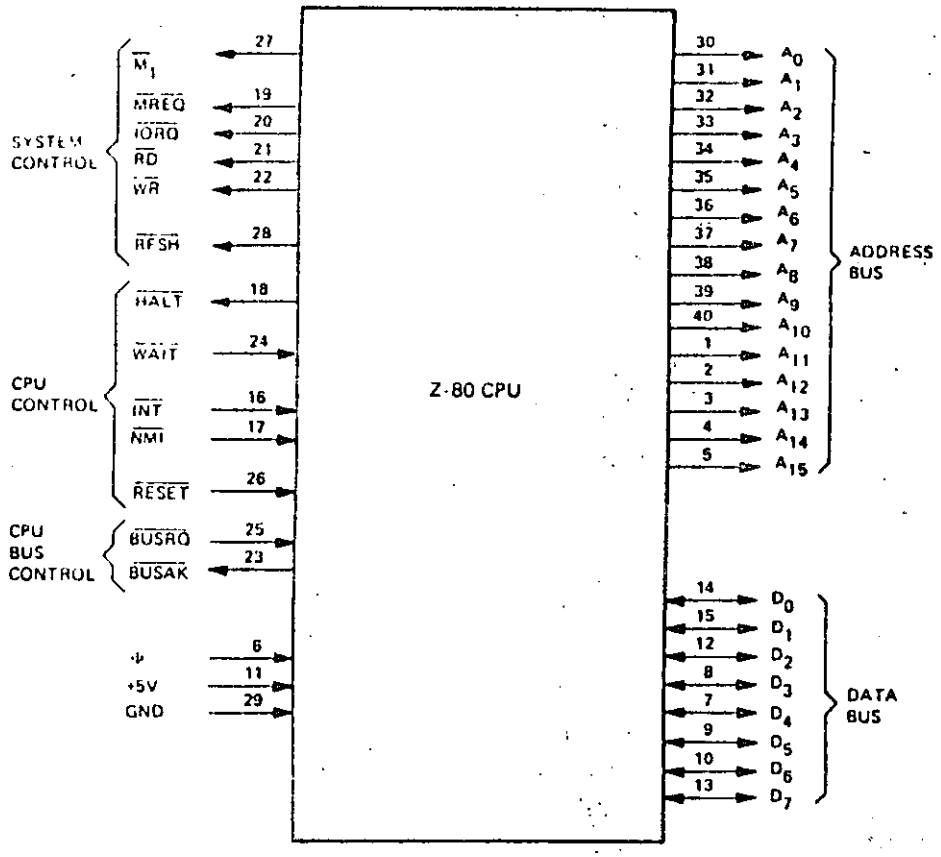
The Z-80 CPU contains 208 bits of R/W memory that are accessible to the programmer. Figure 2.0-2 illustrates how this memory is configured into eighteen 8-bit registers and four 16-bit registers. All Z-80 registers are implemented using static RAM. The registers include two sets of six general purpose registers that may be used individually as 8-bit registers or in pairs as 16-bit registers. There are also two sets of accumulator and flag registers.

Special Purpose Registers

1. Program Counter (PC). The program counter holds the 16-bit address of the current instruction being fetched from memory. The PC is automatically incremented after its contents have been transferred to the address lines. When a program jump occurs the new value is automatically placed in the PC, overriding the incrementer.
2. Stack Pointer (SP). The stack pointer holds the 16-bit address of the current top of a stack located anywhere in external system RAM memory. The external stack memory is organized as a last-in first-out (LIFO) file. Data can be pushed onto the stack from specific CPU registers or popped off of the stack into specific CPU registers through the execution of PUSH and POP instructions. The data popped from the stack is always the last data pushed onto it. The stack allows simple implementation of multiple level interrupts, unlimited subroutine nesting and simplification of many types of data manipulation.

3.0 Z-80 CPU PIN DESCRIPTION

The Z-80 CPU is packaged in an industry standard 40 pin Dual In-Line Package. The I/O pins are shown in figure 3.0-1 and the function of each is described below.



Z-80 PIN CONFIGURATION
FIGURE 3.0-1

A_0-A_{15}
(Address Bus)

Tri-state output, active high. A_0-A_{15} constitute a 16-bit address bus. The address bus provides the address for memory (up to 64K bytes) data exchanges and for I/O device data exchanges. I/O addressing uses the 8 lower address bits to allow the user to directly select up to 256 input or 256 output ports. A_0 is the least significant address bit. During refresh time, the lower 7 bits contain a valid refresh address.

D_0-D_7
(Data Bus)

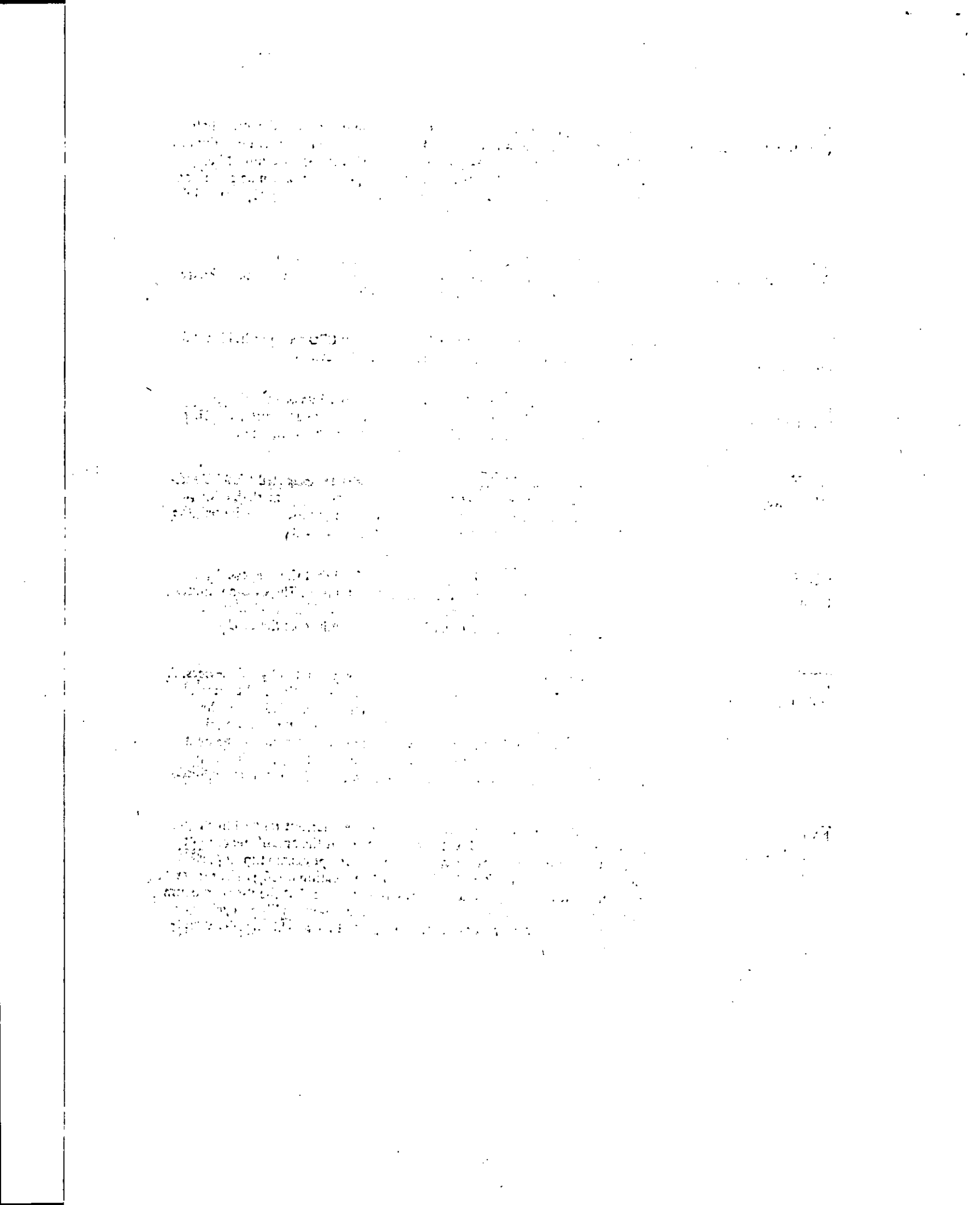
Tri-state input/output, active high. D_0-D_7 constitute an 8-bit bidirectional data bus. The data bus is used for data exchanges with memory and I/O devices.

$\overline{M_1}$
(Machine Cycle one)

Output, active low. $\overline{M_1}$ indicates that the current machine cycle is the OP code fetch cycle of an instruction execution. Note that during execution of 2-byte op-codes, $\overline{M1}$ is generated as each op code byte is fetched. These two byte op-codes always begin with CBH, DDH, EDH or FDH. $\overline{M1}$ also occurs with $\overline{IORQ}$ to indicate an interrupt acknowledge cycle.

$\overline{MREQ}$
(Memory Request)

Tri-state output, active low. The memory request signal indicates that the address bus holds a valid address for a memory read or memory write operation.



$\overline{\text{RESET}}$	Input, active low. $\overline{\text{RESET}}$ forces the program counter to zero and initializes the CPU. The CPU initialization includes: 1) Disable the interrupt enable flip-flop 2) Set Register I = 00_H 3) Set Register R = 00_H 4) Set Interrupt Mode 0 (?) During reset time, the address bus and data bus go to a high impedance state and all control output signals go to the inactive state.
$\overline{\text{BUSRQ}}$ (Bus Request)	Input, active low. The bus request signal is used to request the CPU address bus, data bus and tri-state output control signals to go to a high impedance state so that other devices can control these buses. When $\overline{\text{BUSRQ}}$ is activated, the CPU will set these buses to a high impedance state as soon as the current CPU machine cycle is terminated.
$\overline{\text{BUSAK}}$ (Bus Acknowledge)	Output, active low. Bus acknowledge is used to indicate to the requesting device that the CPU address bus, data bus and tri-state control bus signals have been set to their high impedance state and the external device can now control these signals.
Φ	Single phase TTL level clock which requires only a 330 ohm pull-up resistor to +5 volts to meet all clock requirements.

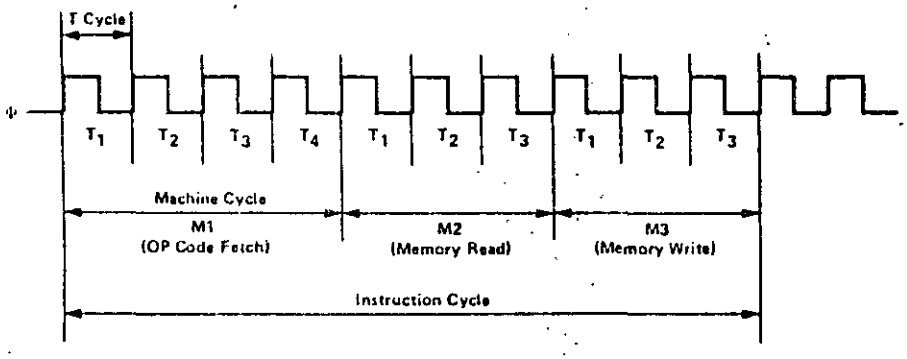
-BLANK- (10)

4.0 CPU TIMING

The Z-80 CPU executes instructions by stepping through a very precise set of a few basic operations. These include:

Memory read or write
I/O device read or write
Interrupt acknowledge

All instructions are merely a series of these basic operations. Each of these basic operations can take from three to six clock periods to complete or they can be lengthened to synchronize the CPU to the speed of external devices. The basic clock periods are referred to as T cycles and the basic operations are referred to as M (for machine) cycles. Figure 4.0-0 illustrates how a typical instruction will be merely a series of specific M and T cycles. Notice that this instruction consists of three machine cycles (M1, M2 and M3). The first machine cycle of any instruction is a fetch cycle which is four, five or six T cycles long (unless lengthened by the wait signal which will be fully described in the next section). The fetch cycle (M1) is used to fetch the OP code of the next instruction to be executed. Subsequent machine cycles move data between the CPU and memory or I/O devices and they may have anywhere from three to five T cycles (again they may be lengthened by wait states to synchronize the external devices to the CPU). The following paragraphs describe the timing which occurs within any of the basic machine cycles. In section 7, the exact timing for each instruction is specified.



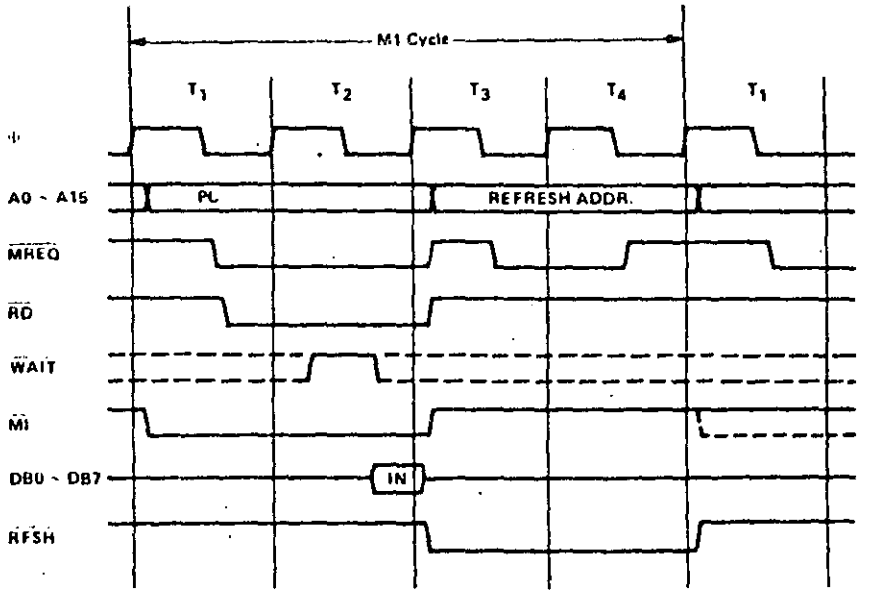
BASIC CPU TIMING EXAMPLE
FIGURE 4.0-0

All CPU timing can be broken down into a few very simple timing diagrams as shown in figure 4.0-1 through 4.0-7. These diagrams show the following basic operations with and without wait states (wait states are added to synchronize the CPU to slow memory or I/O devices).

4.0-1. Instruction OP code fetch (M1 cycle)
4.0-2. Memory data read or write cycles
4.0-3. I/O read or write cycles
4.0-4. Bus Request/Acknowledge Cycle
4.0-5. Interrupt Request/Acknowledge Cycle
4.0-6. Non maskable Interrupt Request/Acknowledge Cycle
4.0-7. Exit from a HALT instruction

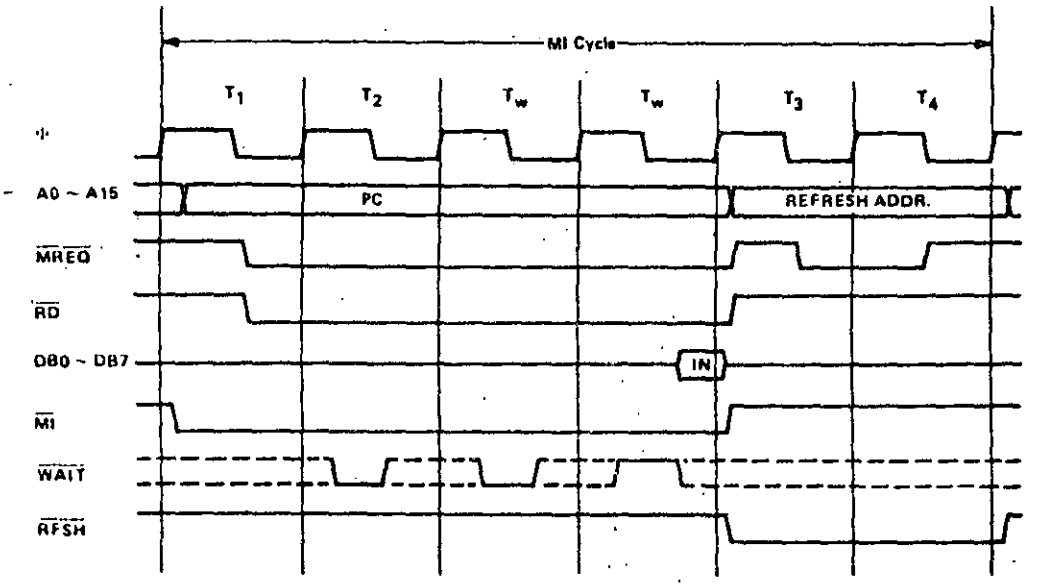
INSTRUCTION FETCH

Figure 4.0-1 shows the timing during an M1 cycle (OP code fetch). Notice that the PC is placed on the address bus at the beginning of the M1 cycle. One half clock time later the $\overline{MREQ}$ signal goes active. At this time the address to the memory has had time to stabilize so that the falling edge of $\overline{MREQ}$ can be used directly as a chip enable clock to dynamic memories. The $\overline{RD}$ line also goes active to indicate that the memory read data should be enabled onto the CPU data bus. The CPU samples the data from the memory on the data bus with the rising edge of the clock of state T3 and this same edge is used by the CPU to turn off the $\overline{RD}$ and $\overline{MRQ}$ signals. Thus the data has already been sampled by the CPU before the $\overline{RD}$ signal becomes inactive. Clock state T3 and T4 of a fetch cycle are used to refresh dynamic memories. (The CPU uses this time to decode and execute the fetched instruction so that no other operation could be performed at this time). During T3 and T4 the lower 7 bits of the address bus contain a memory refresh address and the $\overline{RFSH}$ signal becomes active to indicate that a refresh read of all dynamic memories should be accomplished. Notice that a $\overline{RD}$ signal is not generated during refresh time to prevent data from different memory segments from being gated onto the data bus. The $\overline{MREQ}$ signal during refresh time should be used to perform a refresh read of all memory elements. The refresh signal can not be used by itself since the refresh address is only guaranteed to be stable during $\overline{MREQ}$ time.



INSTRUCTION OP CODE FETCH
FIGURE 4.0-1

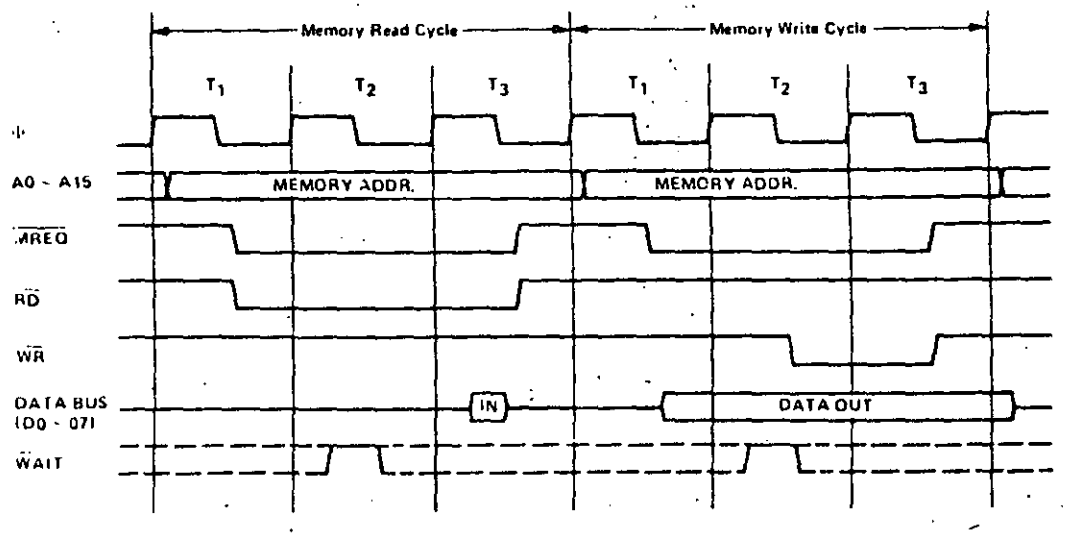
Figure 4.0-1A illustrates how the fetch cycle is delayed if the memory activates the $\overline{WAIT}$ line. During T2 and every subsequent Tw, the CPU samples the $\overline{WAIT}$ line with the falling edge of Φ. If the $\overline{WAIT}$ line is active at this time, another wait state will be entered during the following cycle. Using this technique the read cycle can be lengthened to match the access time of any type of memory device.



INSTRUCTION OP CODE FETCH WITH WAIT STATES
FIGURE 4.0-1A

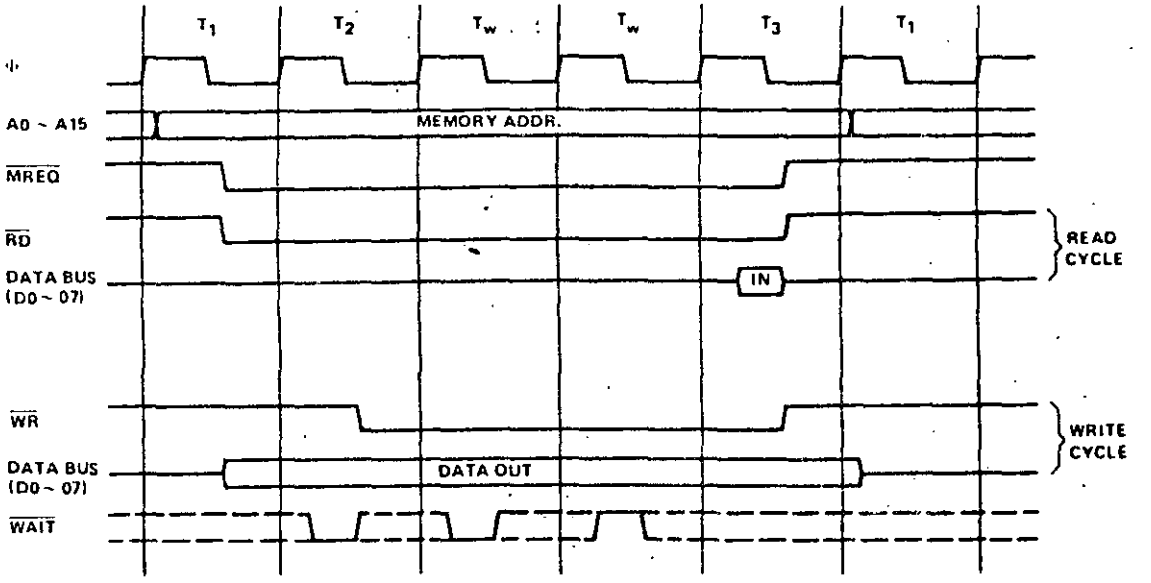
MEMORY READ OR WRITE

Figure 4.0-2 illustrates the timing of memory read or write cycles other than an OP code fetch (M1 cycle). These cycles are generally three clock periods long unless wait states are requested by the memory via the WAIT signal. The MREQ signal and the RD signal are used the same as in the fetch cycle. In the case of a memory write cycle, the MREQ also becomes active when the address bus is stable so that it can be used directly as a chip enable for dynamic memories. The WR line is active when data on the data bus is stable so that it can be used directly as a R/W pulse to virtually any type of semiconductor memory. Furthermore the WR signal goes inactive one half T state before the address and data bus contents are changed so that the overlap requirements for virtually any type of semiconductor memory type will be met.



MEMORY READ OR WRITE CYCLES
FIGURE 4.0-2

Figure 4.0-2A illustrates how a $\overline{WAIT}$ request signal will lengthen any memory read or write operation. This operation is identical to that previously described for a fetch cycle. Notice in this figure that a separate read and a separate write cycle are shown in the same figure although read and write cycles can never occur simultaneously.



MEMORY READ OR WRITE CYCLES WITH WAIT STATES
FIGURE 4.0-2A

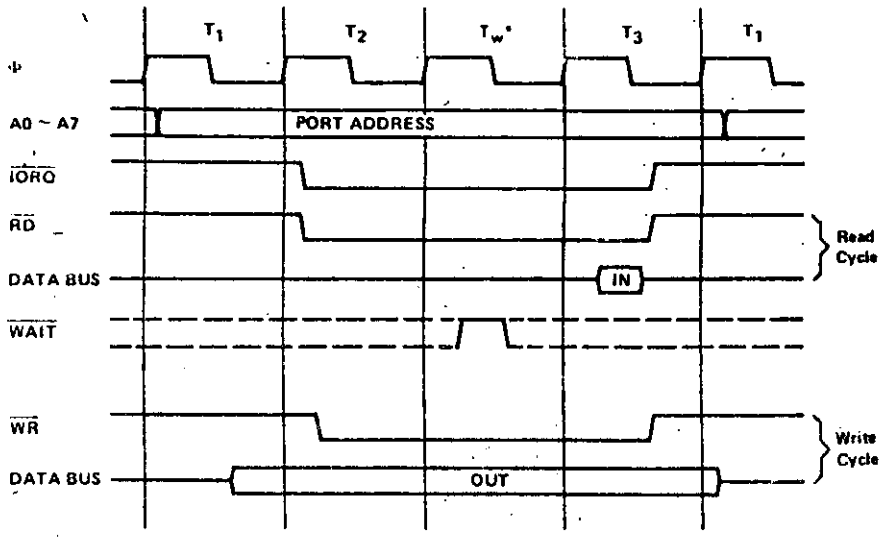
INPUT OR OUTPUT CYCLES

Figure 4.0-3 illustrates an I/O read or I/O write operation. Notice that during I/O operations a single wait state is automatically inserted. The reason for this is that during I/O operations, the time from when the $\overline{IORQ}$ signal goes active until the CPU must sample the $\overline{WAIT}$ line is very short and without this extra state sufficient time does not exist for an I/O port to decode its address and activate the $\overline{WAIT}$ line if a wait is required. Also, without this wait state it is difficult to design MOS I/O devices that can operate at full CPU speed. During this wait state time the $\overline{WAIT}$ request signal is sampled. During a read I/O operation, the $\overline{RD}$ line is used to enable the addressed port onto the data bus just as in the case of a memory read. For I/O write operations, the $\overline{WR}$ line is used as a clock to the I/O port, again with sufficient overlap timing automatically provided so that the rising edge may be used as a data clock.

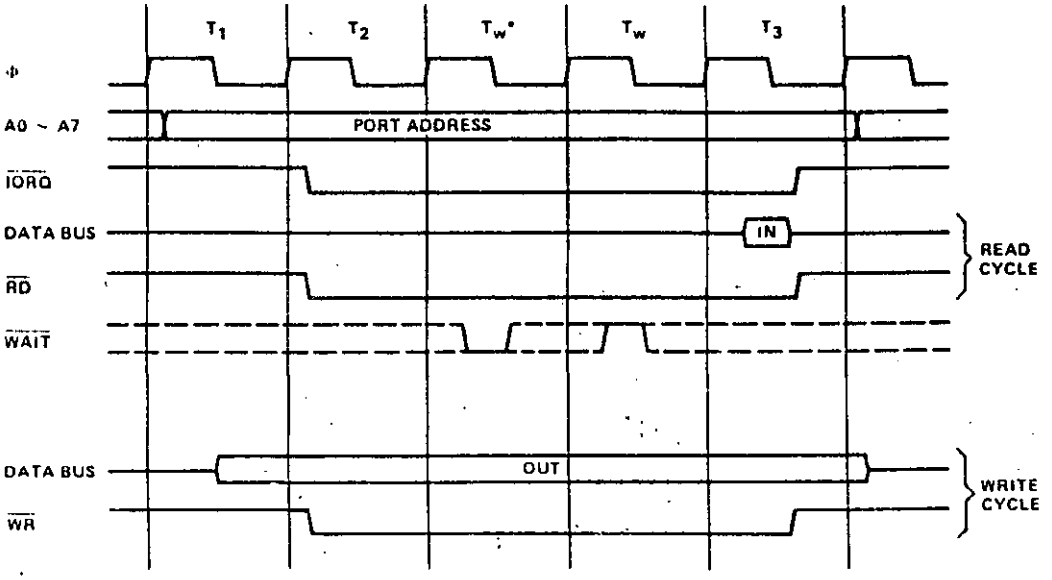
Figure 4.0-3A illustrates how additional wait states may be added with the $\overline{WAIT}$ line. The operation is identical to that previously described.

BUS REQUEST/ACKNOWLEDGE CYCLE

Figure 4.0-4 illustrates the timing for a Bus Request/Acknowledge cycle. The $\overline{BUSRQ}$ signal is sampled by the CPU with the rising edge of the last clock period of any machine cycle. If the $\overline{BUSRQ}$ signal is active, the CPU will set its address, data and tri-state control signals to the high impedance state with the rising edge of the next clock pulse. At that time any external device can control the buses to transfer data between memory and I/O devices. (This is generally known as Direct Memory Access [DMA] using cycle stealing). The maximum time for the CPU to respond to a bus request is the length of a machine cycle and the external controller can maintain control of the bus for as many clock cycles as is desired. Note, however, that if very long DMA cycles are used, and dynamic memories are being used, the external controller must also perform the refresh function. This situation only occurs if very large blocks of data are transferred under DMA control. Also note that during a bus request cycle, the CPU cannot be interrupted by either a $\overline{NMI}$ or an $\overline{INT}$ signal.

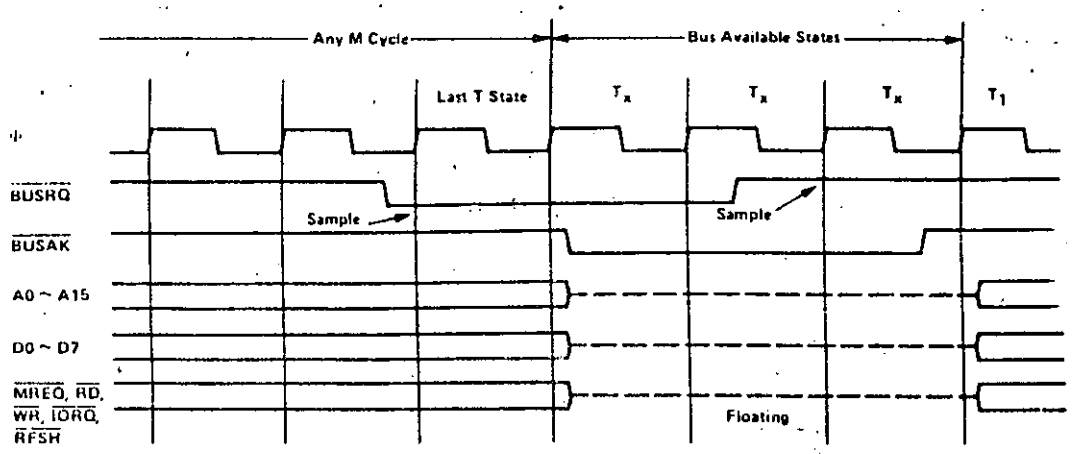


INPUT OR OUTPUT CYCLES
FIGURE 4.0-3



INPUT OR OUTPUT CYCLES WITH WAIT STATES
FIGURE 4.0-3A

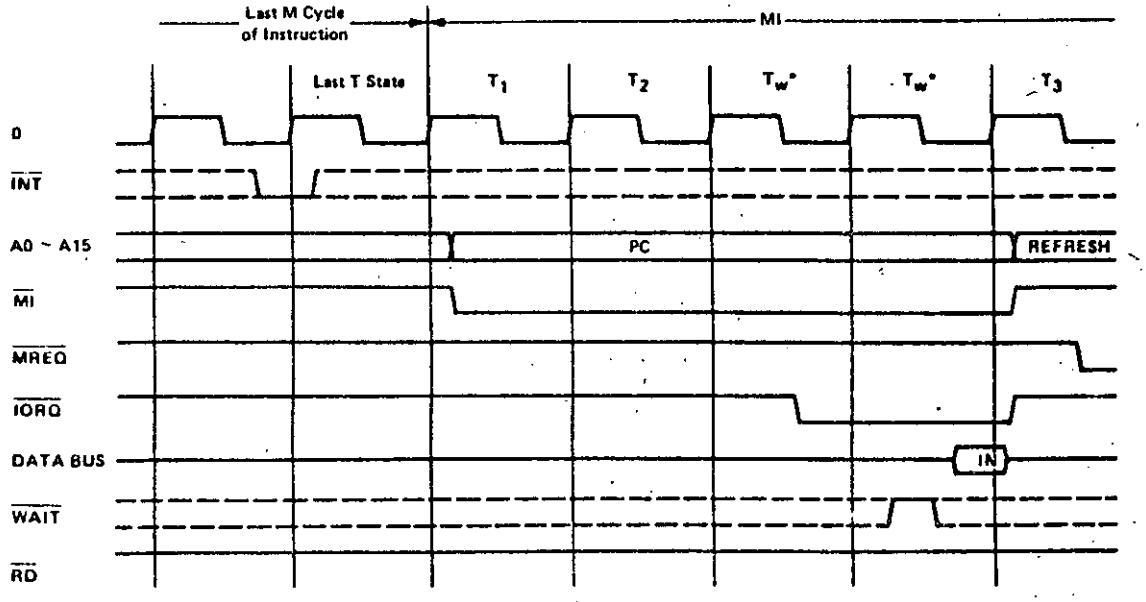
* Automatically inserted WAIT state



BUS REQUEST/ACKNOWLEDGE CYCLE
FIGURE 4.0-4

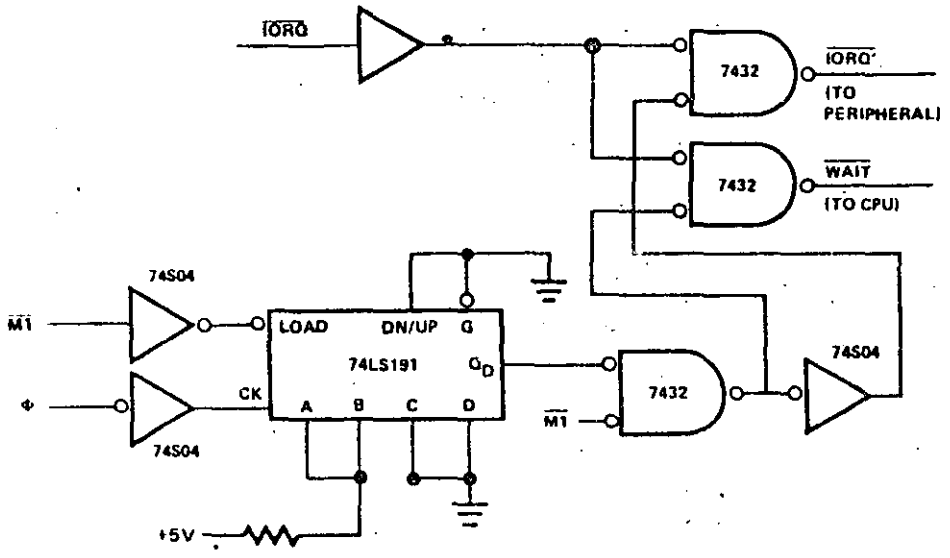
INTERRUPT REQUEST/ACKNOWLEDGE CYCLE

Figure 4.0-5 illustrates the timing associated with an interrupt cycle. The interrupt signal ($\overline{INT}$) is sampled by the CPU with the rising edge of the last clock at the end of any instruction. The signal will not be accepted if the internal CPU software controlled interrupt enable flip-flop is not set or if the $\overline{BUSRQ}$ signal is active. When the signal is accepted a special M1 cycle is generated. During this special M1 cycle the $\overline{IORQ}$ signal becomes active (instead of the normal $\overline{MREQ}$) to indicate that the interrupting device can place an 8-bit vector on the data bus. Notice that two wait states are automatically added to this cycle. These states are added so that a ripple priority interrupt scheme can be easily implemented. The two wait states allow sufficient time for the ripple signals to stabilize and identify which I/O device must insert the response vector. Refer to section 8.0 for details on how the interrupt response vector is utilized by the CPU.

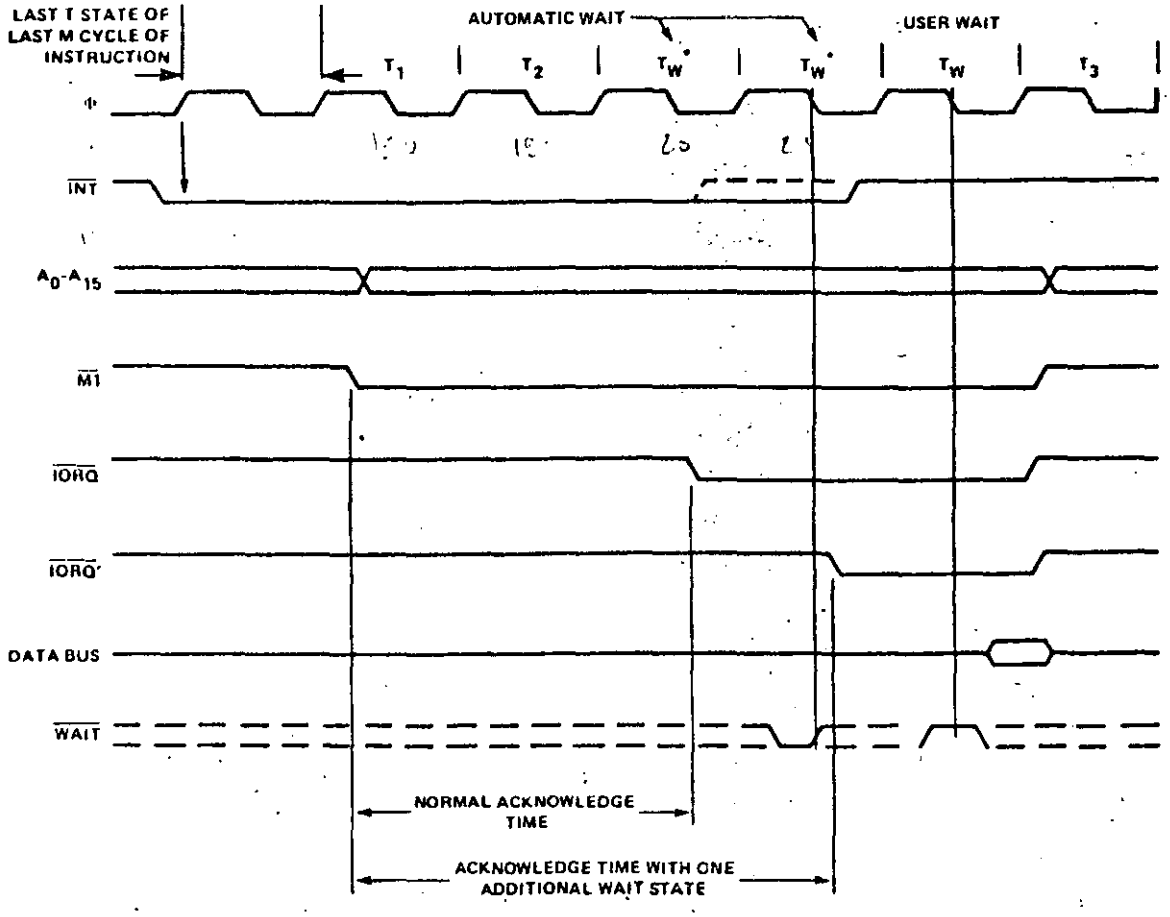


INTERRUPT REQUEST/ACKNOWLEDGE CYCLE
FIGURE 4.0-5

Figures 4.0-5A and 4.0-5B illustrate how a programmable counter can be used to extend interrupt acknowledge time. (Configured as shown to add one wait state)



EXTENDING INTERRUPT ACKNOWLEDGE TIME WITH WAIT STATE
FIGURE 4.0-5A



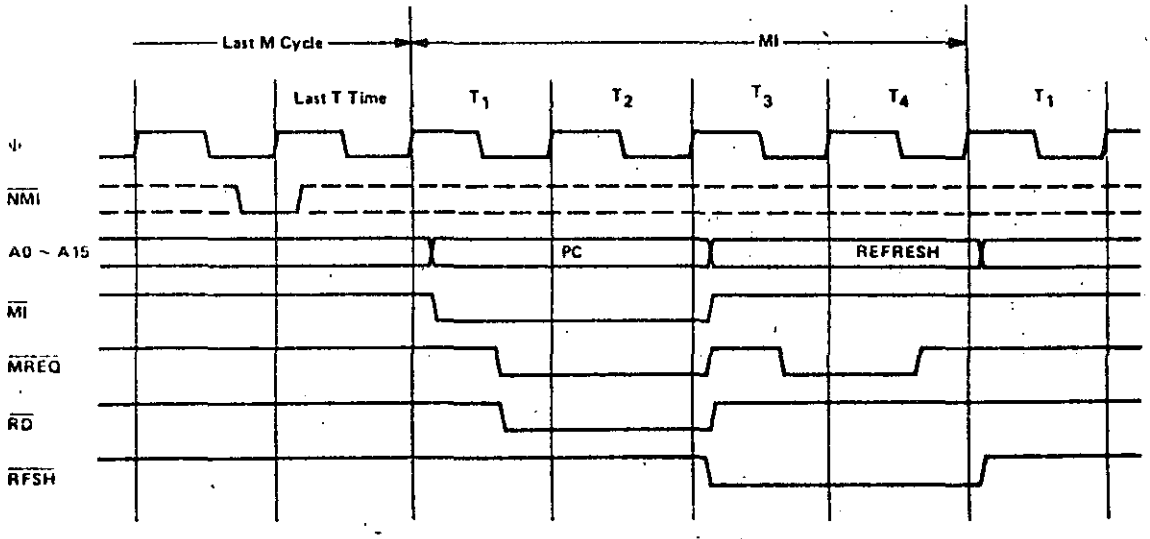
REQUEST/ACKNOWLEDGE CYCLE WITH ONE ADDITIONAL WAIT STATE
FIGURE 4.0-5B

NON MASKABLE INTERRUPT RESPONSE

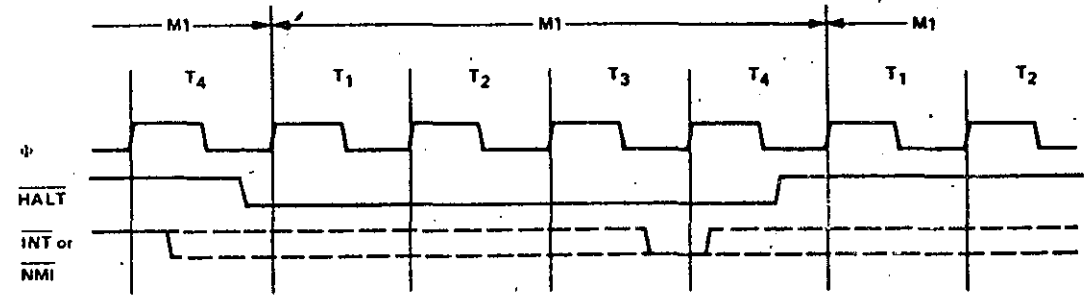
Figure 4.0-6 illustrates the request/acknowledge cycle for the non maskable interrupt. This signal is sampled at the same time as the interrupt line, but this line has priority over the normal interrupt and it can not be disabled under software control. Its usual function is to provide immediate response to important signals such as an impending power failure. The CPU response to a non maskable interrupt is similar to a normal memory read operation. The only difference being that the content of the data bus is ignored while the processor automatically stores the PC in the external stack and jumps to location 0066_H. The service routine for the non maskable interrupt must begin at this location if this interrupt is used.

HALT EXIT

Whenever a software halt instruction is executed the CPU begins executing NOP's until an interrupt is received (either a non maskable or a maskable interrupt while the interrupt flip flop is enabled). The two interrupt lines are sampled with the rising clock edge during each T4 state as shown in figure 4.0-7. If a non maskable interrupt has been received or a maskable interrupt has been received and the interrupt enable flip-flop is set, then the halt state will be exited on the next rising clock edge. The following cycle will then be an interrupt acknowledge cycle corresponding to the type of interrupt that was received. If both are received at this time, then the non maskable one will be acknowledged since it has highest priority. The purpose of executing NOP instructions while in the halt state is to keep the memory refresh signals active. Each cycle in the halt state is a normal M1 (fetch) cycle except that the data received from the memory is ignored and a NOP instruction is forced internally to the CPU. The halt acknowledge signal is active during this time to indicate that the processor is in the halt state.



NON MASKABLE INTERRUPT REQUEST OPERATION
FIGURE 4.0-6



HALT INSTRUCTION IS RECEIVED DURING THIS MEMORY CYCLE

HALT EXIT
FIGURE 4.0-7

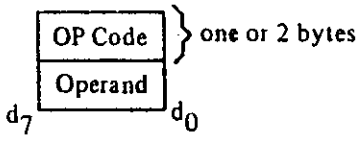
The input/output group of instructions in the Z-80 allow for a wide range of transfers between external memory locations or the general purpose CPU registers, and the external I/O devices. In each case, the port number is provided on the lower 8 bits of the address bus during any I/O transaction. One instruction allows this port number to be specified by the second byte of the instruction while other Z-80 instructions allow it to be specified as the content of the C register. One major advantage of using the C register as a pointer to the I/O device is that it allows different I/O ports to share common software driver routines. This is not possible when the address is part of the OP code if the routines are stored in ROM. Another feature of these input instructions is that they set the flag register automatically so that additional operations are not required to determine the state of the input data (for example its parity). The Z-80 CPU includes single instructions that can move blocks of data (up to 256 bytes) automatically to or from any I/O port directly to any memory location. In conjunction with the dual set of general purpose registers, these instructions provide for fast I/O block transfer rates. The value of this I/O instruction set is demonstrated by the fact that the Z-80 CPU can provide all required floppy disk formatting (i.e., the CPU provides the preamble, address, data and enables the CRC codes) on double density floppy disk drives on an interrupt driven basis.

Finally, the basic CPU control instructions allow various options and modes. This group includes instructions such as setting or resetting the interrupt enable flip flop or setting the mode of interrupt response.

5.2 ADDRESSING MODES

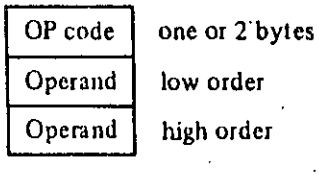
Most of the Z-80 instructions operate on data stored in internal CPU registers, external memory or in the I/O ports. Addressing refers to how the address of this data is generated in each instruction. This section gives a brief summary of the types of addressing used in the Z-80 while subsequent sections detail the type of addressing available for each instruction group.

Immediate. In this mode of addressing the byte following the OP code in memory contains the actual operand.



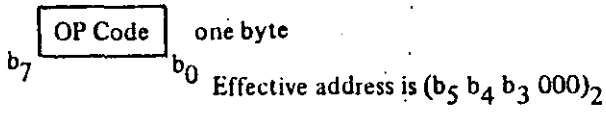
Examples of this type of instruction would be to load the accumulator with a constant, where the constant is the byte immediately following the OP code.

Immediate Extended. This mode is merely an extension of immediate addressing in that the two bytes following the OP codes are the operand.

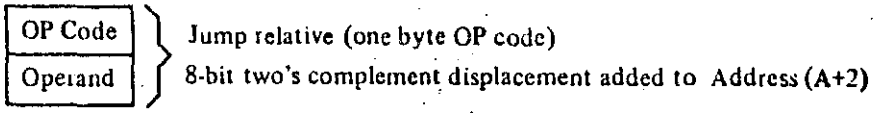


Examples of this type of instruction would be to load the HL register pair (16-bit register) with 16 bits (2 bytes) of data.

Modified Page Zero Addressing. The Z-80 has a special single byte CALL instruction to any of 8 locations in page zero of memory. This instruction (which is referred to as a restart) sets the PC to an effective address in page zero. The value of this instruction is that it allows a single byte to specify a complete 16-bit address where commonly called subroutines are located, thus saving memory space.

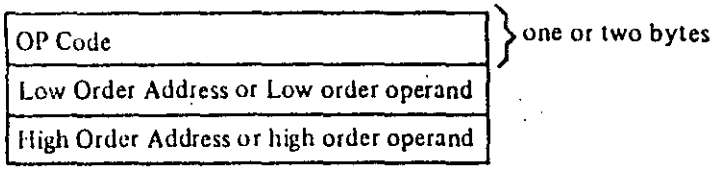


Relative Addressing. Relative addressing uses one byte of data following the OP code to specify a displacement from the existing program to which a program jump can occur. This displacement is a signed two's complement number that is added to the address of the OP code of the following instruction.



The value of relative addressing is that it allows jumps to nearby locations while only requiring two bytes of memory space. For most programs, relative jumps are by far the most prevalent type of jump due to the proximity of related program segments. Thus, these instructions can significantly reduce memory space requirements. The signed displacement can range between +127 and -128 from A + 2. This allows for a total displacement of +129 to -126 from the jump relative OP code address. Another major advantage is that it allows for relocatable code.

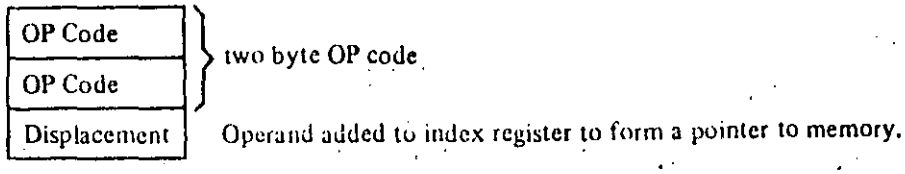
Extended Addressing. Extended Addressing provides for two bytes (16 bits) of address to be included in the instruction. This data can be an address to which a program can jump or it can be an address where an operand is located.



Extended addressing is required for a program to jump from any location in memory to any other location, or load and store data in any memory location.

When extended addressing is used to specify the source or destination address of an operand, the notation (nn) will be used to indicate the content of memory at nn, where nn is the 16-bit address specified in the instruction. This means that the two bytes of address nn are used as a pointer to a memory location. The use of the parentheses always means that the value enclosed within them is used as a pointer to a memory location. For example, (1200) refers to the contents of memory at location 1200.

Indexed Addressing. In this type of addressing, the byte of data following the OP code contains a displacement which is added to one of the two index registers (the OP code specifies which index register is used) to form a pointer to memory. The contents of the index register are not altered by this operation.



An example of an indexed instruction would be to load the contents of the memory location (Index Register + Displacement) into the accumulator. The displacement is a signed two's complement number. Indexed addressing greatly simplifies programs using tables of data since the index register can point to the start of any table. Two index registers are provided since very often operations require two or more tables. Indexed addressing also allows for relocatable code.

The two index registers in the Z-80 are referred to as IX and IY. To indicate indexed addressing the notation:

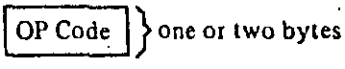
(IX+d) or (IY+d)

is used. Here d is the displacement specified after the OP code. The parentheses indicate that this value is used as a pointer to external memory.

Register Addressing. Many of the Z-80 OP codes contain bits of information that specify which CPU register is to be used for an operation. An example of register addressing would be to load the data in register B into register C.

Implied Addressing. Implied addressing refers to operations where the OP code automatically implies one or more CPU registers as containing the operands. An example is the set of arithmetic operations where the accumulator is always implied to be the destination of the results.

Register Indirect Addressing. This type of addressing specifies a 16-bit CPU register pair (such as HL) to be used as a pointer to any location in memory. This type of instruction is very powerful and it is used in a wide range of applications.



An example of this type of instruction would be to load the accumulator with the data in the memory location pointed to by the HL register contents. Indexed addressing is actually a form of register indirect addressing except that a displacement is added with indexed addressing. Register indirect addressing allows for very powerful but simple to implement memory accesses. The block move and search commands in the Z-80 are extensions of this type of addressing where automatic register incrementing, decrementing and comparing has been added. The notation for indicating register indirect addressing is to put parentheses around the name of the register that is to be used as the pointer. For example, the symbol

(HL)

specifies that the contents of the HL register are to be used as a pointer to a memory location. Often register indirect addressing is used to specify 16-bit operands. In this case, the register contents point to the lower order portion of the operand while the register contents are automatically incremented to obtain the upper portion of the operand.

Bit Addressing. The Z-80 contains a large number of bit set, reset and test instructions. These instructions allow any memory location or CPU register to be specified for a bit operation through one of three previous addressing modes (register, register indirect and indexed) while three bits in the OP code specify which of the eight bits is to be manipulated.

ADDRESSING MODE COMBINATIONS

Many instructions include more than one operand (such as arithmetic instructions or loads). In these cases, two types of addressing may be employed. For example, load can use immediate addressing to specify the source and register indirect or indexed addressing to specify the destination.

6.0 FLAGS

Each of the two Z-80 CPU Flag registers contains six bits of information which are set or reset by various CPU operations. Four of these bits are testable; that is, they are used as conditions for jump, call or return instructions. For example a jump may be desired only if a specific bit in the flag register is set. The four testable flag bits are:

1) Carry Flag (C) – This flag is the carry from the highest order bit of the accumulator. For example, the carry flag will be set during an add instruction where a carry from the highest bit of the accumulator is generated. This flag is also set if a borrow is generated during a subtraction instruction. The shift and rotate instructions also affect this bit.

2) Zero Flag (Z) – This flag is set if the result of the operation loaded a zero into the accumulator. Otherwise it is reset.

3) Sign Flag (S) – This flag is intended to be used with signed numbers and it is set if the result of the operation was negative. Since bit 7 (MSB) represents the sign of the number (A negative number has a 1 in bit 7), this flag stores the state of bit 7 in the accumulator.

4) Parity/Overflow Flag (P/V) – This dual purpose flag indicates the parity of the result in the accumulator when logical operations are performed (such as AND A, B) and it represents overflow when signed two's complement arithmetic operations are performed. The Z-80 overflow flag indicates that the two's complement number in the accumulator is in error since it has exceeded the maximum possible (+127) or is less than the minimum possible (-128) number than can be represented in two's complement notation. For example consider adding:

```
+120 =    0111 1000
+105 =    0110 1001
-------------------
  C = 0  1110 0001  = -95 (wrong) Overflow has occured
```

Here the result is incorrect. Overflow has occurred and yet there is no carry to indicate an error. For this case the overflow flag would be set. Also consider the addition of two negative numbers:

```
 -5 =    1111 1011
-16 =    1111 0000
-------------------
  C = 1  1110 1011  = -21 correct
```

Notice that the answer is correct but the carry is set so that this flag can not be used as an overflow indicator. In this case the overflow would not be set.

For logical operations (AND, OR, XOR) this flag is set if the parity of the result is even and it is reset if it is odd.

There are also two non-testable bits in the flag register. Both of these are used for BCD arithmetic. They are:

1) Half carry (H) – This is the BCD carry or borrow result from the least significant four bits of operation. When using the DAA (Decimal Adjust Instruction) this flag is used to correct the result of a previous packed decimal add or subtract.

2) Subtract Flag (N) – Since the algorithm for correcting BCD operations is different for addition or subtraction, this flag is used to specify what type of instruction was executed last so that the DAA operation will be correct for either addition or subtraction.

The Flag register can be accessed by the programmer and its format is as follows:

S	Z	X	H	X	P/V	N	C

X means flag is indeterminate.

Table 6.0-1 lists how each flag bit is affected by various CPU instructions. In this table a '•' indicates that the instruction does not change the flag, an 'X' means that the flag goes to an indeterminate state, a '0' means that it is reset, a '1' means that it is set and the symbol '‡' indicates that it is set or reset according to the previous discussion. Note that any instruction not appearing in this table does not affect any of the flags.

Table 6.0-1 includes a few special cases that must be described for clarity. Notice that the block search instruction sets the Z flag if the last compare operation indicated a match between the source and the accumulator data. Also, the parity flag is set if the byte counter (register pair BC) is not equal to zero. This same use of the parity flag is made with the block move instructions. Another special case is during block input or output instructions, here the Z flag is used to indicate the state of register B which is used as a byte counter. Notice that when the I/O block transfer is complete, the zero flag will be reset to a zero (i.e. B=0) while in the case of a block move command the parity flag is reset when the operation is complete. A final case is when the refresh or I register is loaded into the accumulator, the interrupt enable flip flop is loaded into the parity flag so that the complete state of the CPU can be saved at any time.

Instruction	C	Z	P/V	S	N	H	Comments
ADD A, s; ADC A,s	‡	‡	V	‡	0	‡	8-bit add or add with carry
SUB s; SBC A, s, CP s, NEG	‡	‡	V	‡	1	‡	8-bit subtract, subtract with carry, compare and negate accumulator
AND s	0	‡	P	‡	0	1	Logical operations
OR s; XOR s	0	‡	P	‡	0	0	And set's different flags
INC s	•	‡	V	‡	0	‡	8-bit increment
DEC m	•	‡	V	‡	1	‡	8-bit decrement
ADD DD, ss	‡	•	•	•	0	X	16-bit add
ADC HL, ss	‡	‡	V	‡	0	X	16-bit add with carry
SBC HL, ss	‡	‡	V	‡	1	X	16-bit subtract with carry
RLA; RLCA, RRA, RRCA	‡	•	•	•	0	0	Rotate accumulator
RL m; RLC m; RR m; RRC m SLA m; SRA m; SRL m	‡	‡	P	‡	0	0	Rotate and shift location m
RLD, RRD	•	‡	P	‡	0	0	Rotate digit left and right
DAA	‡	‡	P	‡	•	‡	Decimal adjust accumulator
CPL	•	•	•	•	1	1	Complement accumulator
SCF	1	•	•	•	0	0	Set carry
CCF	‡	•	•	•	0	X	Complement carry
IN r, (C)	•	‡	P	‡	0	0	Input register indirect
INI; IND; OUTI; OUTD	•	‡	X	X	1	X	Block input and output
INIR; INDR; OTIR; OTDR	•	1	X	X	1	X	Z = 0 if B ≠ 0 otherwise Z = 1
LDI, LDD	•	X	‡	X	0	0	Block transfer instructions
LDIR, LDDR	•	X	0	X	0	0	P/V = 1 if BC ≠ 0, otherwise P/V = 0
CPI, CPIR, CPD, CPDR	•	‡	‡	‡	1	X	Block search instructions Z = 1 if A = (HL), otherwise Z = 0 P/V = 1 if BC ≠ 0, otherwise P/V = 0
LD A, I; LD A, R	•	‡	IFF	‡	0	0	The content of the interrupt enable flip-flop (IFF) is copied into the P/V flag
BIT b, s	•	‡	X	X	0	1	The state of bit b of location s is copied into the Z flag
NEG	‡	‡	V	‡	1	‡	Negate accumulator

The following notation is used in this table:

Symbol	Operation
C	Carry/link flag. C=1 if the operation produced a carry from the MSB of the operand or result.
Z	Zero flag. Z=1 if the result of the operation is zero.
S	Sign flag. S=1 if the MSB of the result is one.
P/V	Parity or overflow flag. Parity (P) and overflow (V) share the same flag. Logical operations affect this flag with the parity of the result while arithmetic operations affect this flag with the overflow of the result. If P/V holds parity, P/V=1 if the result of the operation is even, P/V=0 if result is odd. If P/V holds overflow, P/V=1 if the result of the operation produced an overflow.
H	Half-carry flag. H=1 if the add or subtract operation produced a carry into or borrow from into bit 4 of the accumulator.
N	Add/Subtract flag. N=1 if the previous operation was a subtract.
	H and N flags are used in conjunction with the decimal adjust instruction (DAA) to properly correct the result into packed BCD format following addition or subtraction using operands with packed BCD format.
‡	The flag is affected according to the result of the operation.
•	The flag is unchanged by the operation.
0	The flag is reset by the operation.
1	The flag is set by the operation.
X	The flag is a "don't care."
V	P/V flag affected according to the overflow result of the operation.
P	P/V flag affected according to the parity result of the operation.
r	Any one of the CPU registers A, B, C, D, E, H, L.
s	Any 8-bit location for all the addressing modes allowed for the particular instruction.
ss	Any 16-bit location for all the addressing modes allowed for that instruction.
ii	Any one of the two index registers IX or IY.
R	Refresh counter.
n	8-bit value in range <0, 255>
nn	16-bit value in range <0, 65535>
m	Any 8-bit location for all the addressing modes allowed for the particular instruction.

SUMMARY OF FLAG OPERATION
TABLE 6.0-1

7.0 SUMMARY OF OP CODES AND EXECUTION TIMES

The following section gives a summary of the Z-80 instructions set. The instructions are logically arranged into groups as shown on tables 7.0-1 through 7.0-11. Each table shows the assembly language mnemonic OP code, the actual OP code, the symbolic operation, the content of the flag register following the execution of each instruction, the number of bytes required for each instruction as well as the number of memory cycles and the total number of T states (external clock periods) required for the fetching and execution of each instruction. Care has been taken to make each table self-explanatory without requiring any cross reference with the text or other tables.

Mnemonic	Symbolic Operation	Flags C	Z	P/V	S	N	H	OP-Code 76 543 210	No. of Bytes	No. of M Cycles	No. of T Cycles	Comments r, r'	Reg.
LD r, r'	r ← r'	•	•	•	•	•	•	01 r r'	1	1	4	r, r'	Reg.
LD r, n	r ← n	•	•	•	•	•	•	00 r 110	2	2	7	000	B
								← n →				001	C
LD r, (HL)	r ← (HL)	•	•	•	•	•	•	01 r 110	1	2	7	010	D
LD r, (IX+d)	r ← (IX+d)	•	•	•	•	•	•	11 011 101	3	5	19	011	E
								01 r 110				100	H
								← d →				101	L
LD r, (IY+d)	r ← (IY+d)	•	•	•	•	•	•	11 111 101	3	5	19	111	A
								01 r 110					
								← d →					
LD (HL), r	(HL) ← r	•	•	•	•	•	•	01 110 r	1	2	7		
LD (IX+d), r	(IX+d) ← r	•	•	•	•	•	•	11 011 101	3	5	19		
								01 110 r					
								← d →					
LD (IY+d), r	(IY+d) ← r	•	•	•	•	•	•	11 111 101	3	5	19		
								01 110 r					
								← d →					
LD (HL), n	(HL) ← n	•	•	•	•	•	•	00 110 110	2	3	10		
								← n →					
LD (IX+d), n	(IX+d) ← n	•	•	•	•	•	•	11 011 101	4	5	19		
								00 110 110					
								← d →					
								← n →					
LD (IY+d), n	(IY+d) ← n	•	•	•	•	•	•	11 111 101	4	5	19		
								00 110 110					
								← d →					
								← n →					
LD A, (BC)	A ← (BC)	•	•	•	•	•	•	00 001 010	1	2	7		
LD A, (DE)	A ← (DE)	•	•	•	•	•	•	00 011 010	1	2	7		
LD A, (nn)	A ← (nn)	•	•	•	•	•	•	00 111 010	3	4	13		
								← n →					
								← n →					
LD (BC), A	(BC) ← A	•	•	•	•	•	•	00 000 010	1	2	7		
LD (DE), A	(DE) ← A	•	•	•	•	•	•	00 010 010	1	2	7		
LD (nn), A	(nn) ← A	•	•	•	•	•	•	00 110 010	3	4	13		
								← n →					
								← n →					
LD A, I	A ← I	•	↕	IFF	↕	0	0	11 101 101	2	2	9		
								01 010 111					
LD A, R	A ← R	•	↕	IFF	↕	0	0	11 101 101	2	2	9		
								01 011 111					
LD I, A	I ← A	•	•	•	•	•	•	11 101 101	2	2	9		
								01 000 111					
LD R, A	R ← A	•	•	•	•	•	•	11 101 101	2	2	9		
								01 001 111					

Notes: r, r' means any of the registers A, B, C, D, E, H, L,

IFF the content of the interrupt enable flip-flop (IFF) is copied into the P/V flag

Flag Notation: • = flag not affected, 0 = flag reset, 1 = flag set, X = flag is unknown,

↕ = flag is affected according to the result of the operation.

Mnemonic	Symbolic Operation	Flags C	Z	P/V	S	N	H	Op-Code 76 543 210	No. of Bytes	No. of M Cycles	No. of T States	Comments
LD dd, nn	dd ← nn	•	•	•	•	•	•	00 dd0 001 ← n → ← n →	3	3	10	dd Pair 00 BC 01 DE
LD IX, nn	IX ← nn	•	•	•	•	•	•	11 011 101 00 100 001 ← n → ← n →	4	4	14	10 HL 11 SP
LD IY, nn	IY ← nn	•	•	•	•	•	•	11 111 101 00 100 001 ← n → ← n →	4	4	14	
LD HL, (nn)	H ← (nn+1) L ← (nn)	•	•	•	•	•	•	00 101 010 ← n → ← n →	3	5	16	
LD dd, (nn)	dd_H ← (nn+1) dd_L ← (nn)	•	•	•	•	•	•	11 101 101 01 dd1 011 ← n → ← n →	4	6	20	
LD IX, (nn)	IX_H ← (nn+1) IX_L ← (nn)	•	•	•	•	•	•	11 011 101 00 101 010 ← n → ← n →	4	6	20	
LD IY, (nn)	IY_H ← (nn+1) IY_L ← (nn)	•	•	•	•	•	•	11 111 101 00 101 010 ← n → ← n →	4	6	20	
LD (nn), HL	(nn+1) ← H (nn) ← L	•	•	•	•	•	•	00 100 010 ← n → ← n →	3	5	16	
LD (nn), dd	(nn+1) ← dd_H (nn) ← dd_L	•	•	•	•	•	•	11 101 101 01 dd0 011 ← n → ← n →	4	6	20	
LD (nn), IX	(nn+1) ← IX_H (nn) ← IX_L	•	•	•	•	•	•	11 011 101 00 100 010 ← n → ← n →	4	6	20	
LD (nn), IY	(nn+1) ← IY_H (nn) ← IY_L	•	•	•	•	•	•	11 111 101 00 100 010 ← n → ← n →	4	6	20	
LD SP, HL	SP ← HL	•	•	•	•	•	•	11 111 001	1	1	6	
LD SP, IX	SP ← IX	•	•	•	•	•	•	11 011 101 11 111 001	2	2	10	
LD SP, IY	SP ← IY	•	•	•	•	•	•	11 111 101 11 111 001	2	2	10	qq Pair
PUSH qq	(SP-2) ← qq_L (SP-1) ← qq_H	•	•	•	•	•	•	11 qq0 101	1	3	11	00 BC 01 DE
PUSH IX	(SP-2) ← IX_L (SP-1) ← IX_H	•	•	•	•	•	•	11 011 101 11 100 101	2	4	15	10 HL 11 AF
PUSH IY	(SP-2) ← IY_L (SP-1) ← IY_H	•	•	•	•	•	•	11 111 101 11 100 101	2	4	15	
POP qq	qq_H ← (SP+1) qq_L ← (SP)	•	•	•	•	•	•	11 qq0 001	1	3	10	
POP IX	IX_H ← (SP+1) IX_L ← (SP)	•	•	•	•	•	•	11 011 101 11 100 001	2	4	14	
POP IY	IY_H ← (SP+1) IY_L ← (SP)	•	•	•	•	•	•	11 111 101 11 100 001	2	4	14	

Notes: dd is any of the register pairs BC, DE, HL, SP
qq is any of the register pairs AF, BC, DE, HL
$(PAIR)_H$, $(PAIR)_L$ refer to high order and low order eight bits of the register pair respectively.
E.g. BC_L = C, AF_H = A

Flag Notation: • = flag not affected, 0 = flag reset, 1 = flag set, X = flag is unknown,
↕ flag is affected according to the result of the operation.

16-BIT LOAD GROUP
TABLE 7.0-2

Mnemonic	Symbolic Operation	Flags C	Z	P/V	S	N	H	Op-Code 76 543 210	No. of Bytes	No. of M Cycles	No. of T States	Comments
EX DE, HL	DE ↔ HL	•	•	•	•	•	•	11 101 011	1	1	4	
EX AF, AF'	AF ↔ AF'	•	•	•	•	•	•	00 001 000	1	1	4	
EXX	(BC DE HL) ↔ (BC' DE' HL')	•	•	•	•	•	•	11 011 001	1	1	4	Register bank and auxiliary register bank exchange
EX (SP), HL	H ↔ (SP+1) L ↔ (SP)	•	•	•	•	•	•	11 100 011	1	5	19	
EX (SP), IX	IX_H ↔ (SP+1) IX_L ↔ (SP)	•	•	•	•	•	•	11 011 101 11 100 011	2	6	23	
EX (SP), IY	IY_H ↔ (SP+1) IY_L ↔ (SP)	•	•	•	•	•	•	11 111 101 11 100 011	2	6	23	
LDI	(DE) ← (HL) DE ← DE+1 HL ← HL+1 BC ← BC-1	•	•	① 1	•	0	0	11 101 101 10 100 000	2	4	16	Load (HL) into (DE), increment the pointers and decrement the byte counter (BC)
LDIR	(DE) ← (HL) DE ← DE+1 HL ← HL+1 BC ← BC-1 Repeat until BC = 0	•	•	0	•	0	0	11 101 101 10 110 000	2 2	5 4	21 16	If BC ≠ 0 If BC = 0
LDD	(DE) ← (HL) DE ← DE-1 HL ← HL-1 BC ← BC-1	•	•	① 1	•	0	0	11 101 101 10 101 000	2	4	16	
LDDR	(DE) ← (HL) DE ← DE-1 HL ← HL-1 BC ← BC-1 Repeat until BC = 0	•	•	0	•	0	0	11 101 101 10 111 000	2 2	5 4	21 16	If BC ≠ 0 If BC = 0
CPI	A - (HL) HL ← HL+1 BC ← BC-1	•	② ↕	① ↕	↕	1	↕	11 101 101 10 100 001	2	4	16	
CPIR	A - (HL) HL ← HL+1 BC ← BC-1 Repeat until A = (HL) or BC = 0	•	② ↕	① ↕	↕	1	↕	11 101 101 10 110 001	2 2	5 4	21 16	If BC ≠ 0 and A ≠ (HL) If BC = 0 or A = (HL)
CPD	A - (HL) HL ← HL-1 BC ← BC-1	•	② ↕	① ↕	↕	1	↕	11 101 101 10 101 001	2	4	16	
CPDR	A - (HL) HL ← HL-1 BC ← BC-1 Repeat until A = (HL) or BC = 0	•	② ↕	① ↕	↕	1	↕	11 101 101 10 111 001	2 2	5 4	21 16	If BC ≠ 0 and A ≠ (HL) If BC = 0 or A = (HL)

Notes: ① P/V flag is 0 if the result of BC-1 = 0, otherwise P/V = 1
② Z flag is 1 if A = (HL), otherwise Z = 0.

Flag Notation: • = flag not affected, 0 = flag reset, 1 = flag set, X = flag is unknown.
↕ = flag is affected according to the result of the operation.

EXCHANGE GROUP AND BLOCK TRANSFER AND SEARCH GROUP

TABLE 7.0-3

Mnemonic	Symbolic Operation	Flags C	Z	P/V	S	N	H	Op-Code 76 543 210	No. of Bytes	No. of M Cycles	No. of T States	Comments
ADD A, r	A ← A + r	‡	‡	V	‡	0	‡	10 000 r	1	1	4	r Reg. 000 B 001 C 010 D 011 E 100 H 101 L 111 A
ADD A, n	A ← A + n	‡	‡	V	‡	0	‡	11 000 110 ← n →	2	2	7	
ADD A, (HL)	A ← A + (HL)	‡	‡	V	‡	0	‡	10 000 110	1	2	7	
ADD A, (IX+d)	A←A + (IX+d)	‡	‡	V	‡	0	‡	11 011 101 10 000 110 ← d →	3	5	19	
ADD A, (IY+d)	A←A+(IY+d)	‡	‡	V	‡	0	‡	11 111 101 10 000 110 ← d →	3	5	19	
ADC A, s	A ← A + s + CY	‡	‡	V	‡	0	‡	001				s is any of r, n, (HL), (IX+d), (IY+d) as shown for ADD instruction
SUB s	A ← A − s	‡	‡	V	‡	1	‡	010				
SBC A, s	A ← A − s − CY	‡	‡	V	‡	1	‡	011				
AND s	A ← A ∧ s	0	‡	P	‡	0	1	100				
OR s	A ← A ∨ s	0	‡	P	‡	0	0	110				The indicated bits replace the 000 in the ADD set above.
XOR s	A ← A ∘ s	0	‡	P	‡	0	0	101				
CP s	A − s	‡	‡	V	‡	1	‡	111				
INC r	r ← r + 1	•	‡	V	‡	0	‡	00 r 100	1	1	4	
INC (HL)	(HL) ← (HL)+1	•	‡	V	‡	0	‡	00 110 100	1	3	11	
INC (IX+d)	(IX+d) ← (IX+d)+1	•	‡	V	‡	0	‡	11 011 101 00 110 100 ← d →	3	6	23	
INC (IY+d)	(IY+d) ← (IY+d) + 1	•	‡	V	‡	0	‡	11 111 101 00 110 100 ← d →	3	6	23	
DEC m	m←m−1	•	‡	V	‡	1	‡	101				m is any of r, (HL), (IX+d), (IY+d) as shown for INC. Same format and states as INC. Replace 100 with 101 in OP code.

Notes: The V symbol in the P/V flag column indicates that the P/V flag contains the overflow of the result of the operation. Similarly the P symbol indicates parity. V = 1 means overflow, V = 0 means not overflow. P = 1 means parity of the result is even, P = 0 means parity of the result is odd.

Flag Notation: • = flag not affected, 0 = flag reset, 1 = flag set, X = flag is unknown.
‡ = flag is affected according to the result of the operation.

8-BIT ARITHMETIC AND LOGICAL GROUP
TABLE 7.0-4

Mnemonic	Symbolic Operation	Flags C	Z	P/V	S	N	H	Op-Code 76 543 210	No. of Bytes	No. of M Cycles	No. of T States	Comments
DAA	Converts acc. content into packed BCD following add or subtract with packed BCD operands	‡	‡	P	‡	•	‡	00 100 111	1	1	4	Decimal adjust accumulator
CPL	$A \leftarrow \bar{A}$	•	•	•	•	1	1	00 101 111	1	1	4	Complement accumulator (one's complement)
NEG	$A \leftarrow 0 - A$	‡	‡	V	‡	1	‡	11 101 101 01 000 100	2	2	8	Negate acc. (two's complement)
CCF	$CY \leftarrow \overline{CY}$	‡	•	•	•	0	X	00 111 111	1	1	4	Complement carry flag
SCF	CY ← 1	1	•	•	•	0	0	00 110 111	1	1	4	Set carry flag
NOP	No operation	•	•	•	•	•	•	00 000 000	1	1	4	
HALT	CPU halted	•	•	•	•	•	•	01 110 110	1	1	4	
DI	IFF ← 0	•	•	•	•	•	•	11 110 011	1	1	4	
EI	IFF ← 1	•	•	•	•	•	•	11 111 011	1	1	4	
IM 0	Set interrupt mode 0	•	•	•	•	•	•	11 101 101 01 000 110	2	2	8	
IM 1	Set interrupt mode 1	•	•	•	•	•	•	11 101 101 01 010 110	2	2	8	
IM2	Set interrupt mode 2	•	•	•	•	•	•	11 101 101 01 011 110	2	2	8	

Notes: IFF indicates the interrupt enable flip-flop
CY indicates the carry flip-flop.

Flag Notation: • = flag not affected, 0 = flag reset, 1 = flag set, X = flag is unknown.
‡ = flag is affected according to the result of the operation.

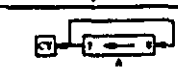
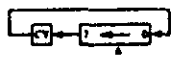
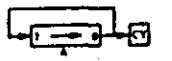
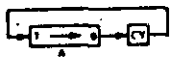
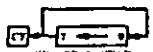
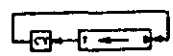
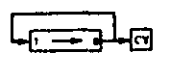
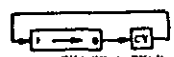
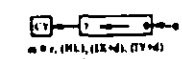
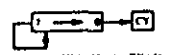
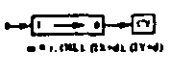
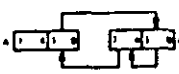
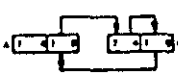
GENERAL PURPOSE ARITHMETIC AND CPU CONTROL GROUPS
TABLE 7.0-5

Mnemonic	Symbolic Operation	Flags						Op-Code	No. of Bytes	No. of M Cycles	No. of T States	Comments	
		C	Z	P/V	S	N	H	76 543 210					
ADD HL, ss	HL ← HL+ss	‡	•	•	•	0	X	00 ss1 001	1	3	11	ss	Reg.
												00	BC
												01	DE
ADC HL, ss	HL←HL+ss+CY	‡	‡	V	‡	0	X	11 101 101	2	4	15	10	HL
								01 ss1 010				11	SP
SBC HL, ss	HL←HL-ss-CY	‡	‡	V	‡	1	X	11 101 101	2	4	15		
								01 ss0 010					
ADD IX, pp	IX ← IX + pp	‡	•	•	•	0	X	11 011 101	2	4	15	pp	Reg.
								00 pp1 001				00	BC
												01	DE
												10	IX
												11	SP
ADD IY, rr	IY←IY+rr	‡	•	•	•	0	X	11 111 101	2	4	15	rr	Reg.
								00 rr1 001				00	BC
												01	DE
												10	IY
												11	SP
INC ss	ss ← ss + 1	•	•	•	•	•	•	00 ss0 011	1	1	6		
INC IX	IX ← IX + 1	•	•	•	•	•	•	11 011 101	2	2	10		
								00 100 011					
INC IY	IY ← IY + 1	•	•	•	•	•	•	11 111 101	2	2	10		
								00 100 011					
DEC ss	ss ← ss - 1	•	•	•	•	•	•	00 ss1 011	1	1	6		
DEC IX	IX ← IX - 1	•	•	•	•	•	•	11 011 101	2	2	10		
								00 101 011					
DEC IY	IY ← IY - 1	•	•	•	•	•	•	11 111 101	2	2	10		
								00 101 011					

Notes: ss is any of the register pairs BC, DE, HL, SP
pp is any of the register pairs BC, DE, IX, SP
rr is any of the register pairs BC, DE, IY, SP.

Flag Notation: • = flag not affected, 0 = flag reset, 1 = flag set, X = flag is unknown.
‡ = flag is affected according to the result of the operation.

16-BIT ARITHMETIC GROUP
TABLE 7.0-6

Mnemonic	Symbolic Operation	Flags C	Z	P/V	S	N	H	Op-Code 76 543 210	No. of Bytes	No. of M Cycles	No. of T States	Comments
RLCA		‡	•	•	•	0	0	00 000 111	1	1	4	Rotate left circular accumulator
RLA		‡	•	•	•	0	0	00 010 111	1	1	4	Rotate left accumulator
RRCA		‡	•	•	•	0	0	00 001 111	1	1	4	Rotate right circular accumulator
RRA		‡	•	•	•	0	0	00 011 111	1	1	4	Rotate right accumulator
RLC r		‡	‡	P	‡	0	0	11 001 011 00 [000] r	2	2	8	Rotate left circular register r
RLC (HL)		‡	‡	P	‡	0	0	11 001 011 00 [000] 110	2	4	15	r Reg. 000 B 001 C 010 D 011 E 100 H 101 L 111 A
RLC (IX+d)		‡	‡	P	‡	0	0	11 011 101 11 001 011 ← d → 00 [000] 110	4	6	23	
RLC (IY+d)		‡	‡	P	‡	0	0	11 111 101 11 001 011 ← d → 00 [000] 110	4	6	23	
RL m		‡	‡	P	‡	0	0	[010]				Instruction format and states are as shown for RLC,m. To form new OP-code replace [000] of RLC,m with shown code
RRC m		‡	‡	P	‡	0	0	[001]				
RR m		‡	‡	P	‡	0	0	[011]				
SLA m		‡	‡	P	‡	0	0	[100]				
SRA m		‡	‡	P	‡	0	0	[101]				
SRL m		‡	‡	P	‡	0	0	[111]				
RLD		•	‡	P	‡	0	0	11 101 101 01 101 111	2	5	18	Rotate digit left and right between the accumulator and location (HL). The content of the upper half of the accumulator is unaffected
RRD		•	‡	P	‡	0	0	11 101 101 01 100 111	2	5	18	

Flag Notation: • = flag not affected, 0 = flag reset, 1 = flag set, X = flag is unknown,
‡ = flag is affected according to the result of the operation.

ROTATE AND SHIFT GROUP
TABLE 7.0-7

Mnemonic	Symbolic Operation	Flags C	Z	P/V	S	N	H	Op-Code 76 543 210	No. of Bytes	No. of M Cycles	No. of T States	Comments
BIT b, r	$Z \leftarrow \overline{r_b}$	•	↕	X	X	0	1	11 001 011 01 b r	2	2	8	
BIT b, (HL)	$Z \leftarrow \overline{(HL)_b}$	•	↕	X	X	0	1	11 001 011 01 b 110	2	3	12	
BIT b, (IX+d)	$Z \leftarrow \overline{(IX+d)_b}$	•	↕	X	X	0	1	11 011 101 11 001 011 ← d → 01 b 110	4	5	20	
BIT b, (IY+d)	$Z \leftarrow \overline{(IY+d)_b}$	•	↕	X	X	0	1	11 111 101 11 001 011 ← d → 01 b 110	4	5	20	
SET b, r	$r_b \leftarrow 1$	•	•	•	•	•	•	11 001 011 [11] b r	2	2	8	
SET b, (HL)	$(HL)_b \leftarrow 1$	•	•	•	•	•	•	11 001 011 [11] b 110	2	4	15	
SET b, (IX+d)	$(IX+d)_b \leftarrow 1$	•	•	•	•	•	•	11 011 101 11 001 011 ← d → [11] b 110	4	6	23	
SET b, (IY+d)	$(IY+d)_b \leftarrow 1$	•	•	•	•	•	•	11 111 101 11 001 011 ← d → [11] b 110	4	6	23	
RES b, m	$s_b \leftarrow 0$ m ≡ r, (HL), (IX+d), (IY+d)							[10]				To form new OP-code replace [11] of SET b,m with [10]. Flags and time states for SET instruction

r	Reg.
000	B
001	C
010	D
011	E
100	H
101	L
111	A

b	Bit Tested
000	0
001	1
010	2
011	3
100	4
101	5
110	6
111	7

Notes: The notation s_b indicates bit b (0 to 7) of location s.

Flag Notation: • = flag not affected, 0 = flag reset, 1 = flag set, X = flag is unknown,
↕ = flag is affected according to the result of the operation.

BIT SET, RESET AND TEST GROUP
TABLE 7.0-8

Mnemonic	Symbolic Operation	Flags C	Z	P/V	S	N	H	Op-Code 76 543 210	No. of Bytes	No. of M Cycles	No. of T States	Comments
JP nn	PC ← nn	•	•	•	•	•	•	11 000 011 ← n → ← n →	3	3	10	
JP cc, nn	If condition cc is true PC ←nn, otherwise continue	•	•	•	•	•	•	11 cc 010 ← n → ← n →	3	3	10	
JR e	PC ← PC + e	•	•	•	•	•	•	00 011 000 ← e-2 →	2	3	12	
JR C, e	If C = 0, continue	•	•	•	•	•	•	00 111 000 ← e-2 →	2	2	7	If condition not met
	If C = 1, PC ← PC+e								2	3	12	If condition is met
JR NC, e	If C = 1, continue	•	•	•	•	•	•	00 110 000 ← e-2 →	2	2	7	If condition not met
	If C = 0, PC ← PC + e								2	3	12	If condition is met
JR Z, e	If Z = 0 continue	•	•	•	•	•	•	00 101 000 ← e-2 →	2	2	7	If condition not met
	If Z = 1, PC ← PC + e								2	3	12	If condition is met
JR NZ, e	If Z = 1, continue	•	•	•	•	•	•	00 100 000 ← e-2 →	2	2	7	If condition not met
	If Z = 0, PC ← PC + e								2	3	12	If condition met
JP (HL)	PC ← HL	•	•	•	•	•	•	11 101 001	1	1	4	
JP (IX)	PC ← IX	•	•	•	•	•	•	11 011 101 11 101 001	2	2	8	
JP (IY)	PC ← IY	•	•	•	•	•	•	11 111 101 11 101 001	2	2	8	
DJNZ, e	B ← B-1 If B = 0, continue	•	•	•	•	•	•	00 010 000 ← e-2 →	2	2	8	If B ≠ 0
	If B ≠ 0, PC ← PC + e								2	3	13	If B ≠ 0

cc	Condition
000	NZ non zero
001	Z zero
010	NC non carry
011	C carry
100	PO panty odd
101	PE panty even
110	P sign positive
111	M sign negative

Notes: e represents the extension in the relative addressing mode.

e is a signed two's complement number in the range <-126, 129>

e-2 in the op-code provides an effective address of pc +e as PC is incremented by 2 prior to the addition of e.

Flag Notation: • = flag not affected, 0 = flag reset, 1 = flag set, X = flag is unknown, ↕ = flag is affected according to the result of the operation.

JUMP GROUP
TABLE 7.0-9

Mnemonic	Symbolic Operation	Flags C	Z	P/V	S	N	H	Op-Code 76	543	210	No. of Bytes	No. of M Cycles	No. of T States	Comments
CALL nn	$(SP-1)\leftarrow PC_H$ $(SP-2)\leftarrow PC_L$ $PC\leftarrow nn$	•	•	○	•	•	•	11	001	101 ← n ← n	3	5	17	
CALL cc, nn	If condition cc is false continue, otherwise same as CALL nn	•	•	•	•	•	•	11	cc	100 ← n ← n	3 3	3 5	10 17	If cc is false If cc is true
RET	$PC_L\leftarrow(SP)$ $PC_H\leftarrow(SP+1)$	•	•	•	•	•	•	11	001	001	1	3	10	
RET cc	If condition cc is false continue, otherwise same as RET	•	•	•	•	•	•	11	cc	000	1 1	1 3	5 11	If cc is false If cc is true
RETI	Return from interrupt	•	•	•	•	•	•	11 01	101 001	101 101	2	4	14	
RETN	Return from non maskable interrupt	•	•	•	•	•	•	11 01	101 000	101 101	2	4	14	
RST p	$(SP-1)\leftarrow PC_H$ $(SP-2)\leftarrow PC_L$ $PC_H\leftarrow 0$ $PC_L\leftarrow P$	•	•	•	•	•	•	11	t	111	1	3	11	

cc	Condition	
000	NZ	non zero
001	Z	zero
010	NC	non carry
011	C	carry
100	PO	parity odd
101	PE	parity even
110	P	sign positive
111	M	sign negative

t	P
000	00H
001	08H
010	10H
011	18H
100	20H
101	28H
110	30H
111	38H

Flag Notation: • = flag not affected, 0 = flag reset, 1 = flag set, X = flag is unknown
‡ = flag is affected according to the result of the operation.

CALL AND RETURN GROUP
TABLE 7.0-10

Figure 8.0-1 is a summary of the effect of different instructions on the two enable flip flops.

Action	IFF_1	IFF_2	
CPU Reset	0	0	
DI	0	0	
EI	1	1	
LD A, I	•	•	$IFF_2 \rightarrow$ Parity flag
LD A, R	•	•	$IFF_2 \rightarrow$ Parity flag
Accept NMI	0	•	
RETN	IFF_2	•	$IFF_2 \rightarrow IFF_1$

"•" indicates no change

FIGURE 8.0-1
INTERRUPT ENABLE/DISABLE FLIP FLOPS

CPU RESPONSE

Non Maskable

A nonmaskable interrupt will be accepted at all times by the CPU. When this occurs, the CPU ignores the next instruction that it fetches and instead does a restart to location 0066H. Thus, it behaves exactly as if it had received a restart instruction but, it is to a location that is not one of the 8 software restart locations. A restart is merely a call to a specific address in page 0 of memory.

Maskable

The CPU can be programmed to respond to the maskable interrupt in any one of three possible modes.

Mode 0

This mode is identical to the 8080A interrupt response mode. With this mode, the interrupting device can place any instruction on the data bus and the CPU will execute it. Thus, the interrupting device provides the next instruction to be executed instead of the memory. Often this will be a restart instruction since the interrupting device only need supply a single byte instruction. Alternatively, any other instruction such as a 3 byte call to any location in memory could be executed.

The number of clock cycles necessary to execute this instruction is 2 more than the normal number for the instruction. This occurs since the CPU automatically adds 2 wait states to an interrupt response cycle to allow sufficient time to implement an external daisy chain for priority control. Section 5.0 illustrates the detailed timing for an interrupt response. After the application of RESET the CPU will automatically enter interrupt Mode 0.

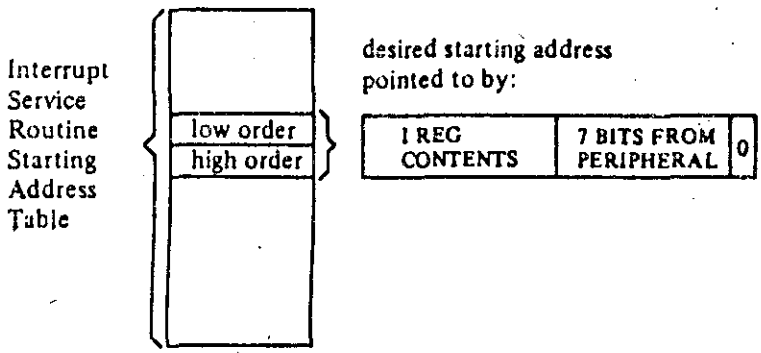
Mode 1

When this mode has been selected by the programmer, the CPU will respond to an interrupt by executing a restart to location 0038H. Thus the response is identical to that for a non maskable interrupt except that the call location is 0038H instead of 0066H. Another difference is that the number of cycles required to complete the restart instruction is 2 more than normal due to the two added wait states.

Mode 2

This mode is the most powerful interrupt response mode. With a single 8 bit byte from the user an indirect call can be made to any memory location.

With this mode the programmer maintains a table of 16 bit starting addresses for every interrupt service routine. This table may be located anywhere in memory. When an interrupt is accepted, a 16 bit pointer must be formed to obtain the desired interrupt service routine starting address from the table. The upper 8 bits of this pointer is formed from the contents of the I register. The I register must have been previously loaded with the desired value by the programmer, i.e. LD I, A. Note that a CPU reset clears the I register so that it is initialized to zero. The lower eight bits of the pointer must be supplied by the interrupting device. Actually, only 7 bits are required from the interrupting device as the least significant bit must be a zero. This is required since the pointer is used to get two adjacent bytes to form a complete 16 bit service routine starting address and the addresses must always start in even locations.



The first byte in the table is the least significant (low order) portion of the address. The programmer must obviously fill this table in with the desired addresses before any interrupts are to be accepted.

Note that this table can be changed at any time by the programmer (if it is stored in Read/Write Memory) to allow different peripherals to be serviced by different service routines.

Once the interrupting devices supplies the lower portion of the pointer, the CPU automatically pushes the program counter onto the stack, obtains the starting address from the table and does a jump to this address. This mode of response requires 19 clock periods to complete (7 to fetch the lower 8 bits from the interrupting device, 6 to save the program counter, and 6 to obtain the jump address.)

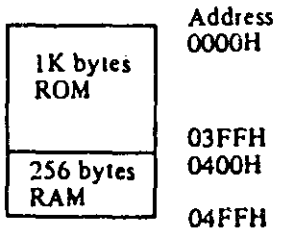
Note that the Z80 peripheral devices all include a daisy chain priority interrupt structure that automatically supplies the programmed vector to the CPU during interrupt acknowledge. Refer to the Z80-PIO, Z80-SIO and Z80-CTC manuals for details.

-BLANK- (58)

Every computer system requires I/O circuits to allow it to interface to the "real world." In this simple example it is assumed that the output is an 8 bit control vector and the input is an 8 bit status word. The input data could be gated onto the data bus using any standard tri-state driver while the output data could be latched with any type of standard TTL latch. For this example we have used a Z80-PIO for the I/O circuit. This single circuit attaches to the data bus as shown and provides the required 16 bits of TTL compatible I/O. (Refer to the Z80-PIO manual for details on the operation of this circuit.) Notice in this example that with only three LSI circuits, a simple oscillator and a single 5 volt power supply, a powerful computer has been implemented.

ADDING RAM

Most computer systems require some amount of external Read/Write memory for data storage and to implement a "stack." Figure 9.0-2 illustrates how 256 bytes of static memory can be added to the previous example. In this example the memory space is assumed to be organized as follows:



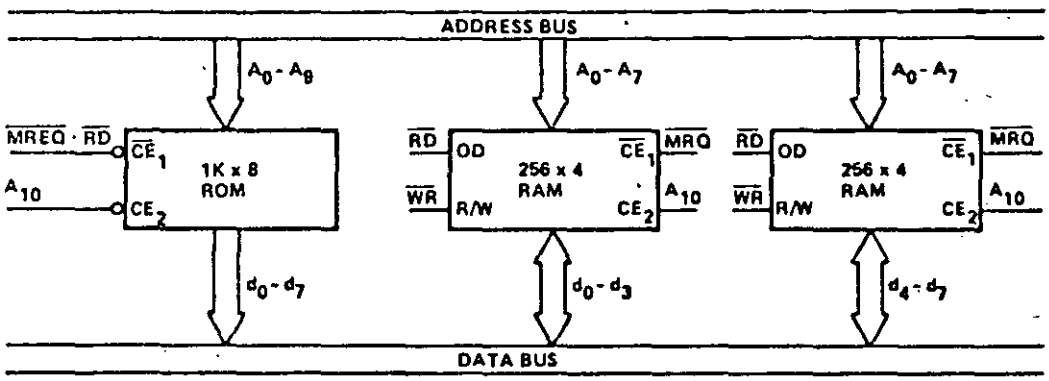


FIGURE 9.0-2
ROM & RAM IMPLEMENTATION EXAMPLE

In this diagram the address space is described in hexidecimal notation. For this example, address bit A_{10} separates the ROM space from the RAM space so that it can be used for the chip select function. For larger amounts of external ROM or RAM, a simple TTL decoder will be required to form the chip selects.

MEMORY SPEED CONTROL

For many applications, it may be desirable to use slow memories to reduce costs. The $\overline{\text{WAIT}}$ line on the CPU allows the Z-80 to operate with any speed memory. By referring back to section 4 you will notice that the memory access time requirements are most severe during the M1 cycle instruction fetch. All other memory accesses have an additional one half of a clock cycle to be completed. For this reason it may be desirable in some applications to add one wait state to the M1 cycle so that slower memories can be used. Figure 9.0-3 is an example of a simple circuit that will accomplish this task. This circuit can be changed to add a single wait state to any memory access as shown in Figure 9.0-4.

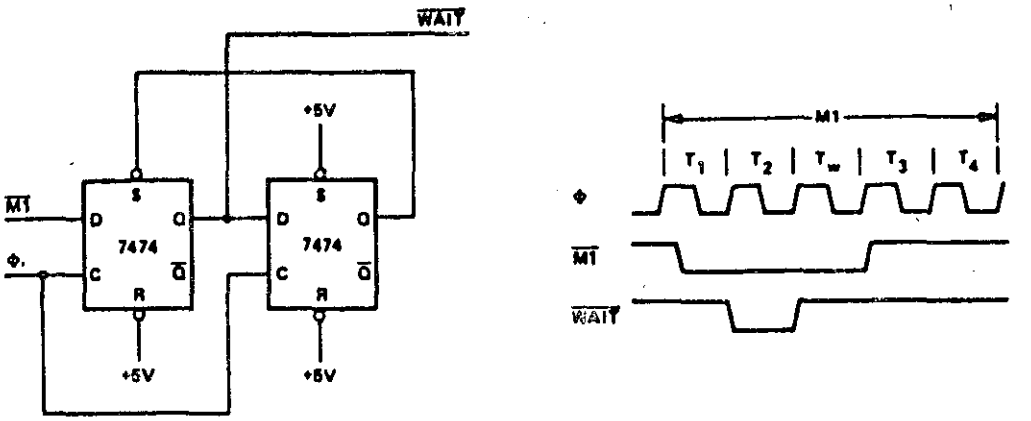


FIGURE 9.0-3
ADDING ONE WAIT STATE TO AN M1 CYCLE

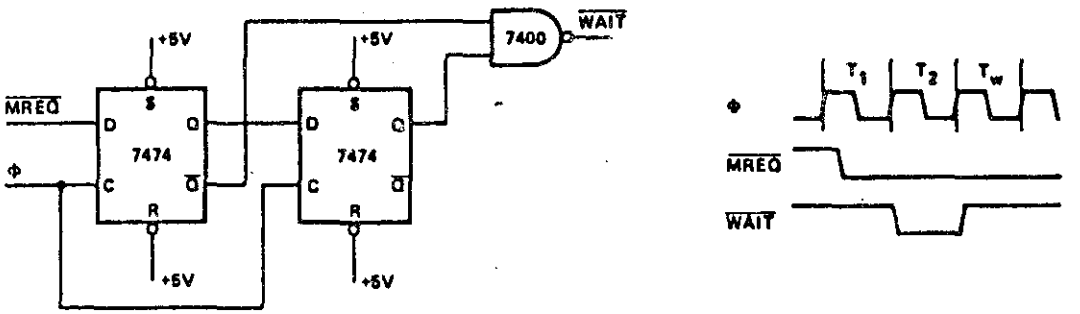


FIGURE 9.0-4
ADDING ONE WAIT STATE TO ANY MEMORY CYCLE

INTERFACING DYNAMIC MEMORIES

This section is intended only to serve as a brief introduction to interfacing dynamic memories. Each individual dynamic RAM has varying specifications that will require minor modifications to the description given here and no attempt will be made in this document to give details for any particular RAM. Separate application notes showing how the Z80-CPU can be interfaced to most popular dynamic RAM's are available from Zilog.

Figure 9.0-5 illustrates the logic necessary to interface 8K bytes of dynamic RAM using 18 pin 4K dynamic memories. This figure assumes that the RAM's are the only memory in the system so that A_{12} is used to select between the two pages of memory. During refresh time, all memories in the system must be read. The CPU provides the proper refresh address on lines A_0 through A_6. To add additional memory to the system it is necessary to only replace the two gates that operate on A_{12} with a decoder that operates on all required address bits. For larger systems, buffering for the address and data bus is also generally required.

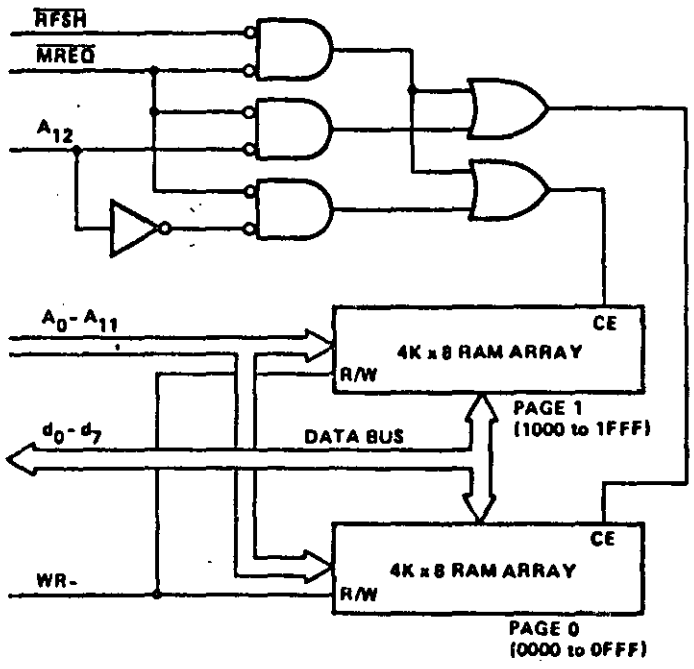


FIGURE 8.0-5
INTERFACING DYNAMIC RAMS

10.0 SOFTWARE IMPLEMENTATION EXAMPLES

10.1 METHODS OF SOFTWARE IMPLEMENTATION

Several different approaches are possible in developing software for the Z-80 (Figure 10.1). First of all, Assembly Language or PL/Z may be used as the source language. These languages may then be translated into machine language on a commercial time sharing facility using a cross-assembler or cross-compiler or, in the case of assembly language, the translation can be accomplished on a Z-80 Development System using a resident assembler. Finally, the resulting machine code can be debugged either on a time-sharing facility using a Z-80 simulator or on a Z-80 Development System which uses a Z80-CPU directly.

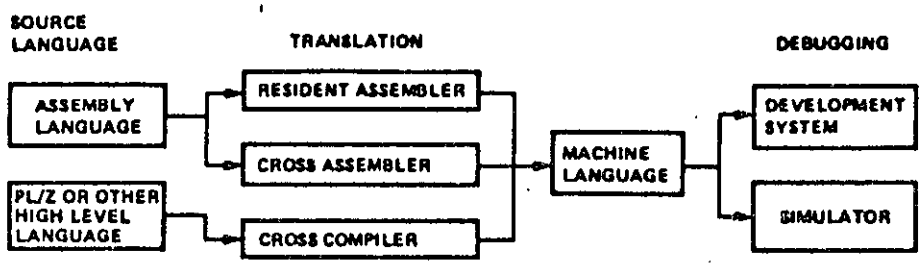


FIGURE 10.1

In selecting a source language, the primary factors to be considered are clarity and ease of programming vs. code efficiency. A high level language such as PL/Z with its machine independent constructs is typically better for formulating and maintaining algorithms, but the resulting machine code is usually somewhat less efficient than what can be written directly in assembly language. These tradeoffs can often be balanced by combining PL/Z and assembly language routines, identifying those portions of a task which must be optimized and writing them as assembly language subroutines.

Deciding whether to use a resident or cross assembler is a matter of availability and short-term vs. long-term expense. While the initial expenditure for a development system is higher than that for a time-sharing terminal, the cost of an individual assembly using a resident assembler is negligible while the same operation on a time-sharing system is relatively expensive and in a short time this cost can equal the total cost of a development system.

Debugging on a development system vs. a simulator is also a matter of availability and expense combined with operational fidelity and flexibility. As with the assembly process, debugging is less expensive on a development system than on a simulator available through time-sharing. In addition, the fidelity of the operating environment is preserved through real-time execution on a Z80-CPU and by connecting the I/O and memory components which will actually be used in the production system. The only advantage to the use of a simulator is the range of criteria which may be selected for such debugging procedures as tracing and setting breakpoints. This flexibility exists because a software simulation can achieve any degree of complexity in its interpretation of machine instructions while development system procedures have hardware limitations such as the capacity of the real-time storage module, the number of breakpoint registers and the pin configuration of the CPU. Despite such hardware limitations, debugging on a development system is typically more productive than on a simulator because of the direct interaction that is possible between the programmer and the authentic execution of his program.

10.2 SOFTWARE FEATURES OFFERED BY THE Z80-CPU

The Z-80 instruction set provides the user with a large and flexible repetoire of operations with which to formulate control of the Z80-CPU.

The primary, auxiliary and index registers can be used to hold the arguments of arithmetic and logical operations, or to form memory addresses, or as fast-access storage for frequently used data.

Information can be moved directly from register to register; from memory to memory; from memory to registers; or from registers to memory. In addition, register contents and register/memory contents can be exchanged without using temporary storage. In particular, the contents of primary and auxilary registers can be completely exchanged by executing only two instructions, EX and EXX. This register exchange procedure can be used to separate the set of working registers between different logical procedures or to expand the set of available registers in a single procedure.

Storage and retrieval of data between pairs of registers and memory can be controlled on a last-in first-out basis through PUSH and POP instructions which utilize a special stack pointer register, SP. This stack register is available both to manipulate data and to automatically store and retrieve addresses for subroutine linkage. When a subroutine is called, for example, the address following the CALL instruction is placed on the top of the push-down stack pointed to by SP. When a subroutine returns to the calling routine, the address on the top of the stack is used to set the program counter for the address of the next instruction. The stack pointer is adjusted automatically to reflect the current "top" stack position during PUSH, POP, CALL and RET instructions. This stack mechanism allows pushdown data stacks and subroutine calls to be nested to any practical depth because the stack area can potentially be as large as memory space.

The sequence of instruction execution can be controlled by six different flags (carry, zero, sign, parity/overflow, add-subtract, half-carry) which reflect the results of arithmetic, logical, shift and compare instructions. After the execution of an instruction which sets a flag, that flag can be used to control a conditional jump or return instruction. These instructions provide logical control following the manipulation of single bit, eight-bit byte (or) sixteen-bit data quantities.

A full set of logical operations, including AND, OR, XOR (exclusive - OR), CPL (NOR) and NEG (two's complement) are available for Boolean operations between the accumulator and 1) all other eight-bit registers, 2) memory locations or 3) immediate operands.

In addition, a full set of arithmetic and logical shifts in both directions are available which operate on the contents of all eight-bit primary registers or directly on any memory location. The carry flag can be included or simply set by these shift instructions to provide both the testing of shift results and to link register/register or register/memory shift operations.

10.3 EXAMPLES OF USE OF SPECIAL Z80 INSTRUCTIONS

A. Let us assume that a string of data in memory starting at location "DATA" is to be moved into another area of memory starting at location "BUFFER" and that the string length is 737 bytes. This operation can be accomplished as follows:

```
LD      HL , DATA       ;START ADDRESS OF DATA STRING
LD      DE , BUFFER     ;START ADDRESS OF TARGET BUFFER
LD      BC , 737        ;LENGTH OF DATA STRING
LDIR                    ;MOVE STRING – TRANSFER MEMORY POINTED TO
                        ;BY HL INTO MEMORY LOCATION POINTED TO BY DE
                        ;INCREMENT HL AND DE, DECREMENT BC
                        ;PROCESS UNTIL BC = 0.
```

11 bytes are required for this operation and each byte of data is moved in 21 clock cycles.

B. Let's assume that a string in memory starting at location "DATA" is to be moved into another area of memory starting at location "BUFFER" until an ASCII $ character (used as string delimiter) is found. Let's also assume that the maximum string length is 132 characters. The operation can be performed as follows:

```
        LD      HL , DATA       ;STARTING ADDRESS OF DATA STRING
        LD      DE , BUFFER     ;STARTING ADDRESS OF TARGET BUFFER
        LD      BC , 132        ;MAXIMUM STRING LENGTH
        LD      A ,'$'          ;STRING DELIMITER CODE
LOOP:CP         (HL)            ;COMPARE MEMORY CONTENTS WITH DELIMITER
        JR      Z , END – $     ;GO TO END IF CHARACTERS EQUAL
        LDI                     ;MOVE CHARACTER (HL) to (DE)
                                ;INCREMENT HL AND DE, DECREMENT BC
        JP      PE , LOOP       ;GO TO "LOOP" IF MORE CHARACTERS
END:                            ;OTHERWISE, FALL THROUGH
                                ;NOTE: P/V FLAG IS USED
                                ;TO INDICATE THAT REGISTER BC WAS
                                ;DECREMENTED TO ZERO.
```

19 bytes are required for this operation.

C. Let us assume that a 16-digit decimal number represented in packed BCD format (two BCD digits/ byte) has to be shifted as shown in the Figure 10.2 in order to mechanize BCD multiplication or division. The operation can be accomplished as follows:

```
        LD      HL , DATA       ; ADDRESS OF FIRST BYTE
        LD      B , COUNT       ; SHIFT COUNT
        XOR     A               ; CLEAR ACCUMULATOR
ROTAT: RLD                      ; ROTATE LEFT LOW ORDER DIGIT IN ACC
                                ; WITH DIGITS IN (HL)
        INC     HL              ; ADVANCE MEMORY POINTER
        DJNZ    ROTAT – $       ; DECREMENT B AND GO TO ROTAT IF
                                ; B IS NOT ZERO, OTHERWISE FALL THROUGH
```

11 bytes are required for this operation.

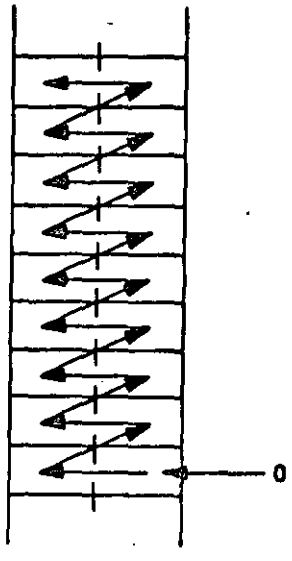


FIGURE 10.2

D. Let us assume that one number is to be subtracted from another and a) that they are both in packed BCD format, b) that they are of equal but varying length, and c) that the result is to be stored in the location of the minuend. The operation can be accomplished as follows:

```
        LD      HL , ARG1       ; ADDRESS OF MINUEND
        LD      DE , ARG2       ; ADDRESS OF SUBTRAHEND
        LD      B , LENGTH      ; LENGTH OF TWO ARGUMENTS
        AND     A               ; CLEAR CARRY FLAG
SUBDEC: LD      A , (DE)        ; SUBTRAHEND TO ACC
        SBC     A , (HL)        ; SUBTRACT (HL) FROM ACC
        DAA                     ; ADJUST RESULT TO DECIMAL CODED VALUE
        LD      (HL) , A        ; STORE RESULT
        INC     HL              ; ADVANCE MEMORY POINTERS
        INC     DE
        DJNZ    SUBDEC - $      ; DECREMENT B AND GO TO "SUBDEC" IF B
                                ; NOT ZERO, OTHERWISE FALL THROUGH
```

17 bytes are required for this operation.

10.4 EXAMPLES OF PROGRAMMING TASKS

A. The following program sorts an array of numbers each in the range ⟨0,255⟩ into ascending order using a standard exchange sorting algorithm.

```
LOC   OBJ CODE   STMT  SOURCE STATEMENT

                    1  ;  *** STANDARD EXCHANGE (BUBBLE) SORT ROUTINE ***
                    2  ;
                    3  ;  AT ENTRY:  HL CONTAINS ADDRESS OF DATA
                    4  ;              C CONTAINS NUMBER OF ELEMENTS TO BE SORTED
                    5  ;              (1<C<256)
                    6  ;
                    7  ;  AT EXIT: DATA SORTED IN ASCENDING ORDER
                    8  ;
                    9  ;  USE OF REGISTERS
                   10  ;
                   11  ;  REGISTER   CONTENTS
                   12  ;
                   13  ;  A          TEMPORARY STORAGE FOR CALCULATIONS
                   14  ;  B          COUNTER FOR DATA ARRAY
                   15  ;  C          LENGTH OF DATA ARRAY
                   16  ;  D          FIRST ELEMENT IN COMPARISON
                   17  ;  E          SECOND ELEMENT IN COMPARISON
                   18  ;  H          FLAG TO INDICATE EXCHANGE
                   19  ;  L          UNUSED
                   20  ;  IX         POINTER INTO DATA ARRAY
                   21  ;  IY         UNUSED
                   22  ;
0000  222600       23  SORT:  LD    (DATA), HL     ;SAVE DATA ADDRESS
0003  CB84         24  LOOP:  RES   FLAG, H        ;INITIALIZE EXCHANGE FLAG
0005  41           25         LD    B, C           ;INITIALIZE LENGTH COUNTER
0006  05           26         DEC   B              ;ADJUST FOR TESTING
0007  DD2A2600     27         LD    IX, (DATA)     ;INITIALIZE ARRAY POINTER
000B  DD7E00       28  NEXT:  LD    A, (IX)        ;FIRST ELEMENT IN COMPARISON
000E  57           29         LD    D, A           ;TEMPORARY STORAGE FOR ELEMENT
000F  DD5E01       30         LD    E, (IX+1)      ;SECOND ELEMENT IN COMPARISON
0012  93           31         SUB   E              ;COMPARISON FIRST TO SECOND
0013  3008         32         JR    NC, NOEX-$     ;IF FIRST > SECOND, NO JUMP
0015  DD7300       33         LD    (IX), E        ;EXCHANGE ARRAY ELEMENTS
0018  DD7201       34         LD    (IX+1), D
001B  CBC4         35         SET   FLAG, H        ;RECORD EXCHANGE OCCURRED
001D  DD23         36  NOEX:  INC   IX           ;POINT TO NEXT DATA ELEMENT
001F  10EA         37         DJNZ  NEXT-$         ;COUNT NUMBER OF COMPARISONS
                   38                              ;REPEAT IF MORE DATA PAIRS
0021  CB44         39         BIT   FLAG, H        ;DETERMINE IF EXCHANGE OCCURRED
0023  20DE         40         JR    NZ, LOOP-$     ;CONTINUE IF DATA UNSORTED
0025  C9           41         RET                  ;OTHERWISE, EXIT
                   42  ;
0026               43  FLAG:  EQU   0              ;DESIGNATION OF FLAG BIT
0026               44  DATA:  DEFS  2              ;STORAGE FOR DATA ADDRESS
                   45         END
```

B. The following program multiplies two unsigned 16 bit integers and leaves the result in the HL register pair.

```
01/22/76     11:32:36              MULTIPLY LISTING                    PAGE 1
LOC    OBJ CODE    STMT  SOURCE STATEMENT

0000                 1   MULT::;  UNSIGNED SIXTEEN BIT INTEGER MULTIPLY.
                     2   ;        ON ENTRANCE:  MULTIPLIER IN DE.
                     3   ;                      MULTIPLICAND IN HL.
                     4   ;
                     5   ;        ON EXIT:  RESULT IN HL.
                     6   ;
                     7   ;        REGISTER USES:
                     8   ;
                     9   ;
                    10   ;        H     HIGH ORDER PARTIAL RESULT
                    11   ;        L     LOW ORDER PARTIAL RESULT
                    12   ;        D     HIGH ORDER MULTIPLICAND
                    13   ;        E     LOW ORDER MULTIPLICAND
                    14   ;        B     COUNTER FOR NUMBER OF SHIFTS
                    15   ;        C     HIGH ORDER BITS OF MULTIPLIER
                    16   ;        A     LOW ORDER BITS OF MULTIPLIER
                    17   ;
0000   0610         18             LD    B, 16;       NUMBER OF BITS- INITIALIZE
0002   4A           19             LD    C, D;        MOVE MULTIPLIER
0003   7B           20             LD    A, E;
0004   EB           21             EX    DE, HL;      MOVE MULTIPLICAND
0005   210000       22             LD    HL, 0;       CLEAR PARTIAL RESULT
0008   CB39         23   MLOOP:    SRL   C;           SHIFT MULTIPLIER RIGHT
000A   1F           24             RRA                LEAST SIGNIFICANT BIT IS
                    25   ;                            IN CARRY.
000B   3001         26             JR    NC, NOADD-$; IF NO CARRY, SKIP THE ADD.
000D   19           27             ADD   HL, DE;      ELSE ADD MULTIPLICAND TO
                    28   ;                            PARTIAL RESULT.
000E   EB           29   NOADD:    EX    DE, HL;      SHIFT MULTIPLICAND LEFT
000F   29           30             ADD   HL, HL;      BY MULTIPLYING IT BY TWO.
0010   EB           31             EX    DE, HL;
0011   10F5         32             DJNZ  MLOOP-$;     REPEAT UNTIL NO MORE BITS.
0013   C9           33             RET;
                    34             END;
```

Absolute Maximum Ratings

Temperature Under Bias	Specified operating range
Storage Temperature	-65°C to +150°C
Voltage On Any Pin with Respect to Ground	-0.3V to +7V
Power Dissipation	1.5W

*Comment

Stresses above those listed under "Absolute Maximum Rating" may cause permanent damage to the device. This is a stress rating only and functional operation of the device at these or any other condition above those indicated in the operational sections of this specification is not implied. Exposure to absolute maximum rating conditions for extended periods may affect device reliability

Note For Z-80A PS at AC and DC characteristics remain the same for the military grade parts except I_{cc}

I_{cc} = 300 mA

Z80-CPU D.C. Characteristics

T_A = 0°C to 70°C, V_{cc} = 5V ± 5% unless otherwise specified

Symbol	Parameter	Min.	Typ.	Max.	Unit	Test Condition
V_{ILC}	Clock Input Low Voltage	-0.3		0.45	V	
V_{IHC}	Clock Input High Voltage	V_{cc} -.6		V_{cc}+.3	V	
V_{IL}	Input Low Voltage	-0.3		0.8	V	
V_{IH}	Input High Voltage	2.0		V_{cc}	V	
V_{OL}	Output Low Voltage			0.4	V	I_{OL}=1.8mA
V_{OH}	Output High Voltage	2.4			V	I_{OH} = -250μA
I_{CC}	Power Supply Current			150	mA	
I_{LI}	Input Leakage Current			10	μA	V_{IN}=0 to V_{cc}
I_{LOH}	Tri-State Output Leakage Current in Float			10	μA	V_{OUT}=2.4 to V_{cc}
I_{LOL}	Tri-State Output Leakage Current in Float			-10	μA	V_{OUT}=0.4V
I_{LD}	Data Bus Leakage Current in Input Mode			±10	μA	$0 \leq V_{IN} \leq V_{cc}$

Capacitance

T_A = 25°C, f = 1 MHz,
unmeasured pins returned to ground

Symbol	Parameter	Max.	Unit
C_ϕ	Clock Capacitance	35	pF
C_{IN}	Input Capacitance	5	pF
C_{OUT}	Output Capacitance	10	pF

Z80-CPU Ordering Information

C = Ceramic
P = Plastic
S = Standard 5V ±5% 0° to 70°C
E = Extended 5V ±5% -40° to 85°C
M = Military 5V ±10% -55° to 125°C

Z80A-CPU D.C. Characteristics

T_A = 0°C to 70°C, V_{cc} = 5V ± 5% unless otherwise specified

Symbol	Parameter	Min.	Typ.	Max.	Unit	Test Condition
V_{ILC}	Clock Input Low Voltage	-0.3		0.45	V	
V_{IHC}	Clock Input High Voltage	V_{cc} -.6		V_{cc}+.3	V	
V_{IL}	Input Low Voltage	-0.3		0.8	V	
V_{IH}	Input High Voltage	2.0		V_{cc}	V	
V_{OL}	Output Low Voltage			0.4	V	I_{OL}=1.8mA
V_{OH}	Output High Voltage	2.4			V	I_{OH} = -250μA
I_{CC}	Power Supply Current		90	200	mA	
I_{LI}	Input Leakage Current			10	μA	V_{IN}=0 to V_{cc}
I_{LOH}	Tri-State Output Leakage Current in Float			10	μA	V_{OUT}=2.4 to V_{cc}
I_{LOL}	Tri-State Output Leakage Current in Float			-10	μA	V_{OUT}=0.4V
I_{LD}	Data Bus Leakage Current in Input Mode			±10	μA	$0 \leq V_{IN} \leq V_{cc}$

Capacitance

T_A = 25°C, f = 1 MHz,
unmeasured pins returned to ground

Symbol	Parameter	Max	Unit
C_ϕ	Clock Capacitance	35	pF
C_{IN}	Input Capacitance	5	pF
C_{OUT}	Output Capacitance	10	pF

Z80A-CPU Ordering Information

C = Ceramic
P = Plastic
S = Standard 5V ±5% 0° to 70°C

T_A = 0°C to 70°C, Vcc = +5V ± 5%, Unless Otherwise Noted.

Signal	Symbol	Parameter	Min	Max	Unit	Test Condition	
Φ	t_c	Clock Period	4	[12]	μsec		[12] $t_c = t_{w(\Phi H)} + t_{w(\Phi L)} + t_f + t_r$
	$t_{w(\Phi H)}$	Clock Pulse Width, Clock High	180	[E]	nsec		
	$t_{w(\Phi L)}$	Clock Pulse Width, Clock Low	180	2000	nsec		
	t_r, t_f	Clock Rise and Fall Time		30	nsec		
A_{0-15}	$t_{D(AD)}$	Address Output Delay		145	nsec	C_L = 50pF	
	$t_{F(AD)}$	Delay to Float		110	nsec		
	t_{acm}	Address Stable Prior to MREQ (Memory Cycle)	[1]		nsec		[1] $t_{acm} = t_{w(\Phi H)} + t_f - 75$
	t_{aci}	Address Stable Prior to IORQ, RD or WR (I/O Cycle)	[2]		nsec		
	t_{ca}	Address Stable from RD, WR, IORQ or MREQ	[3]		nsec		[2] $t_{ca} = t_c - 80$
	t_{cal}	Address Stable From RD or WR During Float	[4]		nsec		
D_{0-7}	$t_{D(D)}$	Data Output Delay		230	nsec	C_L = 50pF	[3] $t_{ca} = t_{w(\Phi L)} + t_f - 40$
	$t_{F(D)}$	Delay to Float During Write Cycle		90	nsec		
	$t_{S\Phi(D)}$	Data Setup Time to Rising Edge of Clock During M1 Cycle	50		nsec		[4] $t_{caf} = t_{w(\Phi L)} + t_f - 60$
	$t_{S\bar{\Phi}(D)}$	Data Setup Time to Falling Edge of Clock During M2 to M5	60		nsec		
	t_{dcm}	Data Stable Prior to WR (Memory Cycle)	[5]		nsec		[5] $t_{dcm} = t_c - 210$
	t_{dci}	Data Stable Prior to WR (I/O Cycle)	[6]		nsec		
	t_{cdf}	Data Stable From WR	[7]				[6] $t_{dci} = t_{w(\Phi L)} + t_f - 210$
	t_H	Any Hold Time for Setup Time	0		nsec		[7] $t_{cdf} = t_{w(\Phi L)} + t_f - 80$
MREQ	$t_{DL\Phi(MR)}$	MREQ Delay From Falling Edge of Clock, MREQ Low		100	nsec	C_L = 50pF	
	$t_{DH\Phi(MR)}$	MREQ Delay From Rising Edge of Clock, MREQ High		100	nsec		
	$t_{DH\bar{\Phi}(MR)}$	MREQ Delay From Falling Edge of Clock, MREQ High		100	nsec		
	$t_{w(MRL)}$	Pulse Width, MREQ Low	[8]		nsec		[8] $t_{w(MRL)} = t_c - 40$
	$t_{w(MRH)}$	Pulse Width, MREQ High	[9]		nsec		[9] $t_{w(MRH)} = t_{w(\Phi H)} + t_f - 30$
IORQ	$t_{DL\Phi(IR)}$	IORQ Delay From Rising Edge of Clock, IORQ Low		90	nsec	C_L = 50pF	
	$t_{DL\bar{\Phi}(IR)}$	IORQ Delay From Falling Edge of Clock, IORQ Low		110	nsec		
	$t_{DH\Phi(IR)}$	IORQ Delay From Rising Edge of Clock, IORQ High		100	nsec		
	$t_{DH\bar{\Phi}(IR)}$	IORQ Delay From Falling Edge of Clock, IORQ High		110	nsec		
RD	$t_{DL\Phi(RD)}$	RD Delay From Rising Edge of Clock, RD Low		100	nsec	C_L = 50pF	
	$t_{DL\bar{\Phi}(RD)}$	RD Delay From Falling Edge of Clock, RD Low		130	nsec		
	$t_{DH\Phi(RD)}$	RD Delay From Rising Edge of Clock, RD High		100	nsec		
	$t_{DH\bar{\Phi}(RD)}$	RD Delay From Falling Edge of Clock, RD High		110	nsec		
WR	$t_{DL\Phi(WR)}$	WR Delay From Rising Edge of Clock, WR Low		80	nsec	C_L = 50pF	
	$t_{DL\bar{\Phi}(WR)}$	WR Delay From Falling Edge of Clock, WR Low		90	nsec		
	$t_{DH\Phi(WR)}$	WR Delay From Falling Edge of Clock, WR High		100	nsec		
	$t_{w(WRL)}$	Pulse Width, WR Low	[10]		nsec		[10] $t_{w(WRL)} = t_c - 40$
M1	$t_{DL(M1)}$	M1 Delay From Rising Edge of Clock, M1 Low		130	nsec	C_L = 50pF	
	$t_{DH(M1)}$	M1 Delay From Rising Edge of Clock, M1 High		130	nsec		
RFSH	$t_{DL(RF)}$	RFSH Delay From Rising Edge of Clock, RFSH Low		180	nsec	C_L = 50pF	
	$t_{DH(RF)}$	RFSH Delay From Rising Edge of Clock, RFSH High		150	nsec		
WAIT	$t_{s(WT)}$	WAIT Setup Time to Falling Edge of Clock	70		nsec		
HALT	$t_{D(HT)}$	HALT Delay Time From Falling Edge of Clock		300	nsec	C_L = 50pF	
INT	$t_{s(IT)}$	INT Setup Time to Rising Edge of Clock	80		nsec		
NMI	$t_{w(NML)}$	Pulse Width, NMI Low	80		nsec		
BUSRQ	$t_{s(BQ)}$	BUSRQ Setup Time to Rising Edge of Clock	80		nsec		
BUSAK	$t_{DL(BA)}$	BUSAK Delay From Rising Edge of Clock, BUSAK Low		120	nsec	C_L = 50pF	
	$t_{DH(BA)}$	BUSAK Delay From Falling Edge of Clock, BUSAK High		110	nsec		
RESET	$t_{s(RS)}$	RESET Setup Time to Rising Edge of Clock	90		nsec		
	$t_{F(C)}$	Delay to Float (MREQ, IORQ, RD and WR)		100	nsec		
	t_{ml}	M1 Stable Prior to IORQ (Interrupt Ack)	[11]		nsec		[11] $t_{ml} = 3t_c + t_{w(\Phi H)} + t_f - 80$

NOTES

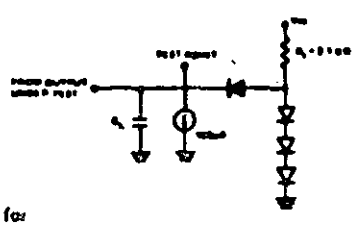
A Data should be enabled onto the CPU data bus when RD is active. During interrupt acknowledge data should be enabled when M1 and IORQ are both active.

B All control signals are internally synchronized, so they may be totally asynchronous with respect to the clock.

C The RESET signal must be active for a minimum of 3 clock cycles.

D Output Delay vs. Loaded Capacitance
TA = 70°C, Vcc = +5V ±5%.
Add 10nsec delay for each 50pf increase in load up to a maximum of 200pf for the data bus & 100pf for address & control lines.

E Although static by design, testing guarantees $t_{w(\Phi H)}$ of 200 μsec maximum.

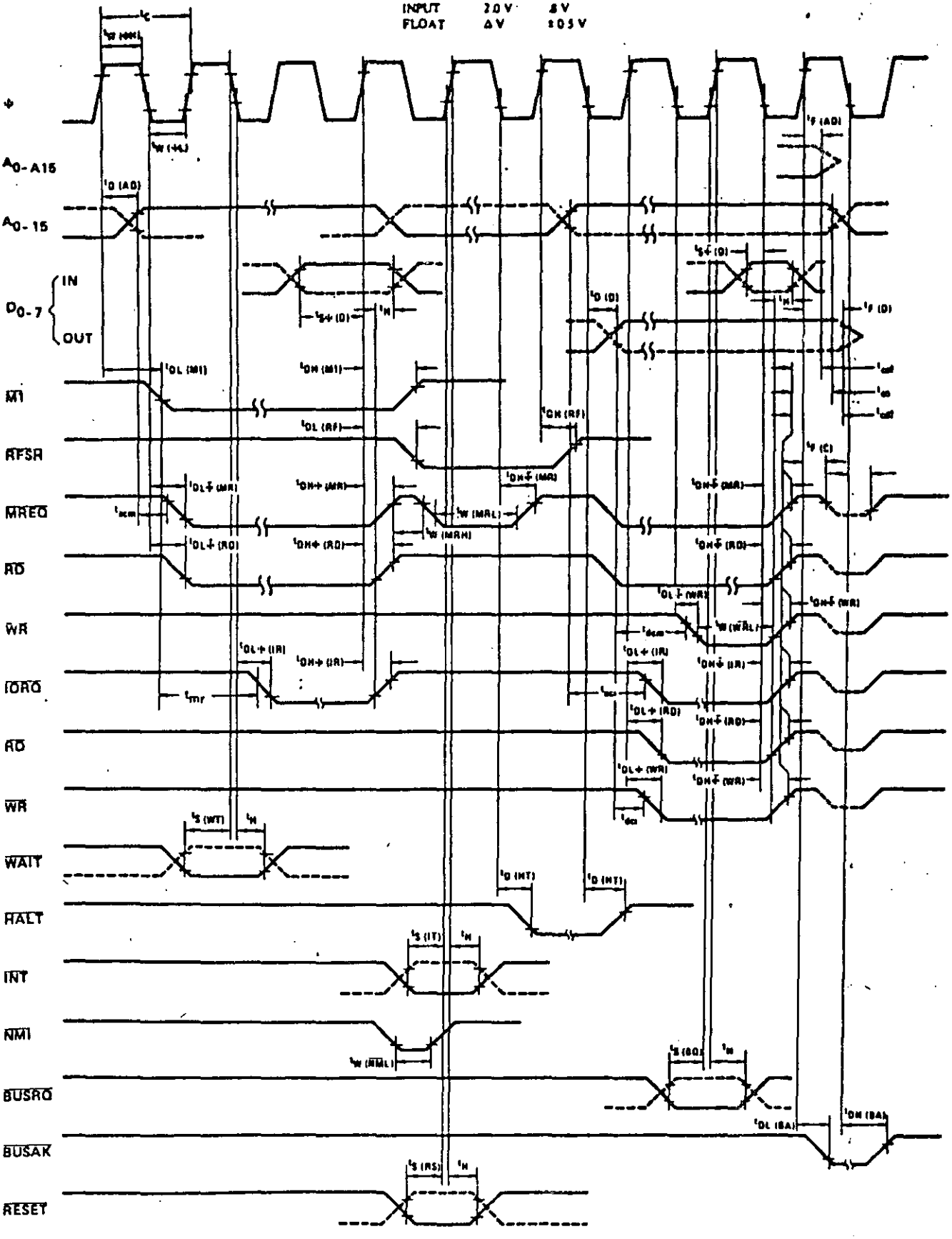


Load circuit for Output

A.C. Timing Diagram

Timing measurements are made at the following voltages, unless otherwise specified:

	"1"	"0"
CLOCK	V_{cc} -.6V	.45V
OUTPUT	2.0V	.8V
INPUT	2.0V	.8V
FLOAT	.6V	±0.5V



T_A = 0°C to 70°C, Vcc = +5V ± 5%, Unless Otherwise Noted.

Signal	Symbol	Parameter	Min	Max	Unit	Test Condition
Φ	t_c	Clock Period	25	[12]	μsec	
	$t_{w(\Phi H)}$	Clock Pulse Width, Clock High	110	[11]	nsec	
	$t_{w(\Phi L)}$	Clock Pulse Width, Clock Low	110	2000	nsec	
	t_r, t_f	Clock Rise and Fall Time		30	nsec	
A_{0-15}	$t_{D(AD)}$	Address Output Delay		110	nsec	C_L = 50pf
	$t_{F(AD)}$	Delay to Float		90	nsec	
	t_{acm}	Address Stable Prior to MREQ (Memory Cycle)	[1]		nsec	
	t_{aci}	Address Stable Prior to IORQ, RD or WR (I/O Cycle)	[2]		nsec	
	t_{ca}	Address Stable from RD, WR, IORQ or MREQ	[3]		nsec	
	t_{caf}	Address Stable From RD or WR During Float	[4]		nsec	
D_{0-7}	$t_{D(D)}$	Data Output Delay		150	nsec	C_L = 50pf
	$t_{F(D)}$	Delay to Float During Write Cycle		90	nsec	
	$t_{S\Phi(D)}$	Data Setup Time to Rising Edge of Clock During M1 Cycle	35		nsec	
	$t_{S\bar{\Phi}(D)}$	Data Setup Time to Falling Edge of Clock During M2 to M5	50		nsec	
	t_{dcm}	Data Stable Prior to WR (Memory Cycle)	[5]		nsec	
	t_{dci}	Data Stable Prior to WR (I/O Cycle)	[6]		nsec	
	t_{cdf}	Data Stable From WR	[7]			
	t_H	Any Hold Time for Setup Time		0	nsec	
MREQ	$t_{DL\bar{\Phi}(MR)}$	MREQ Delay From Falling Edge of Clock, MREQ Low		85	nsec	C_L = 50pf
	$t_{DH\Phi(MR)}$	MREQ Delay From Rising Edge of Clock, MREQ High		85	nsec	
	$t_{DH\bar{\Phi}(MR)}$	MREQ Delay From Falling Edge of Clock, MREQ High		85	nsec	
	$t_{w(\overline{MRL})}$	Pulse Width, MREQ Low	[8]		nsec	
	$t_{w(\overline{MRH})}$	Pulse Width, MREQ High	[9]		nsec	
IORQ	$t_{DL\Phi(IR)}$	IORQ Delay From Rising Edge of Clock, IORQ Low		75	nsec	C_L = 50pf
	$t_{DL\bar{\Phi}(IR)}$	IORQ Delay From Falling Edge of Clock, IORQ Low		85	nsec	
	$t_{DH\Phi(IR)}$	IORQ Delay From Rising Edge of Clock, IORQ High		85	nsec	
	$t_{DH\bar{\Phi}(IR)}$	IORQ Delay From Falling Edge of Clock, IORQ High		85	nsec	
RD	$t_{DL\Phi(RD)}$	RD Delay From Rising Edge of Clock, RD Low		85	nsec	C_L = 50pF
	$t_{DL\bar{\Phi}(RD)}$	RD Delay From Falling Edge of Clock, RD Low		95	nsec	
	$t_{DH\Phi(RD)}$	RD Delay From Rising Edge of Clock, RD High		85	nsec	
	$t_{DH\bar{\Phi}(RD)}$	RD Delay From Falling Edge of Clock, RD High		85	nsec	
WR	$t_{DL\Phi(WR)}$	WR Delay From Rising Edge of Clock, WR Low		65	nsec	C_L = 50pF
	$t_{DL\bar{\Phi}(WR)}$	WR Delay From Falling Edge of Clock, WR Low		80	nsec	
	$t_{DH\Phi(WR)}$	WR Delay From Falling Edge of Clock, WR High		80	nsec	
	$t_{w(\overline{WRL})}$	Pulse Width, WR Low	[10]		nsec	
M1	$t_{DL(M1)}$	M1 Delay From Rising Edge of Clock, M1 Low		100	nsec	C_L = 50pF
	$t_{DH(M1)}$	M1 Delay From Rising Edge of Clock, M1 High		100	nsec	
RFSH	$t_{DL(RF)}$	RFSH Delay From Rising Edge of Clock, RFSH Low		130	nsec	C_L = 50pF
	$t_{DH(RF)}$	RFSH Delay From Rising Edge of Clock, RFSH High		120	nsec	
WAIT	$t_{s(WT)}$	WAIT Setup Time to Falling Edge of Clock	70		nsec	
HALT	$t_{D(HT)}$	HALT Delay Time From Falling Edge of Clock		300	nsec	C_L = 50pF
INT	$t_{s(IT)}$	INT Setup Time to Rising Edge of Clock	80		nsec	
NMI	$t_{w(\overline{NML})}$	Pulse Width, NMI Low	80		nsec	
BUSRQ	$t_{s(BQ)}$	BUSRQ Setup Time to Rising Edge of Clock	50		nsec	
BUSAK	$t_{DL(BA)}$	BUSAK Delay From Rising Edge of Clock, BUSAK Low		100	nsec	C_L = 50pf
	$t_{DH(BA)}$	BUSAK Delay From Falling Edge of Clock, BUSAK High		100	nsec	
RESET	$t_{s(RS)}$	RESET Setup Time to Rising Edge of Clock	60		nsec	
	$t_{F(C)}$	Delay to Float (MREQ, IORQ, RD and WR)		80	nsec	
	t_{mr}	M1 Stable Prior to IORQ (Interrupt Ack.)	[11]		nsec	

[1] $t_{acm} = t_{w(\Phi H)} + t_f - 65$

[2] $t_{aci} = t_c - 70$

[3] $t_{ca} = t_{w(\Phi L)} + t_f - 50$

[4] $t_{caf} = t_{w(\Phi L)} + t_f - 65$

[5] $t_{dcm} = t_c - 170$

[6] $t_{dci} = t_{w(\Phi L)} + t_f - 170$

[7] $t_{cdf} = t_{w(\Phi L)} + t_f - 70$

[8] $t_{w(\overline{MRL})} = t_c - 30$

[9] $t_{w(\overline{MRH})} = t_{w(\Phi H)} + t_f - 30$

[10] $t_{w(\overline{WRL})} = t_c - 30$

[11] $t_{mr} = 2t_c + t_{w(\Phi H)} + t_f - 65$

[12] $t_c = t_{w(\Phi H)} + t_{w(\Phi L)} + t_r + t_f$

NOTES

A. Data should be enabled onto the CPU data bus when RD is active. During interrupt acknowledge data should be enabled when M1 and IORQ are both active.

B. All control signals are internally synchronized, so they may be totally asynchronous with respect to the clock.

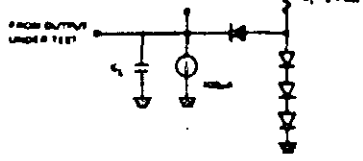
C. The RESET signal must be active for a minimum of 3 clock cycles.

D. Output Delay vs. Loaded Capacitance

TA = 70°C Vcc = +5V ±5%

Add 10nsec delay for each 50pf increase in load up to maximum of 200pf for data bus and 100pf for address & control lines.

E. Although static by design, testing guarantees $t_{w(\Phi H)}$ of 200 μsec maximum.



Load circuit for Output

12.0

Z80-CPU INSTRUCTION SET

ADC HL, ss	Add with Carry Reg. pair ss to HL
ADC A, s	Add with carry operand s to Acc.
ADD A, n	Add value n to Acc.
ADD A, r	Add Reg. r to Acc.
ADD A, (HL)	Add location (HL) to Acc.
ADD A, (IX+d)	Add location (IX+d) to Acc.
ADD A, (IY+d)	Add location (IY+d) to Acc.
ADD HL, ss	Add Reg. pair ss to HL
ADD IX, pp	Add Reg. pair pp to IX
ADD IY, rr	Add Reg. pair rr to IY
AND s	Logical 'AND' of operand s and Acc.
BIT b, (HL)	Test BIT b of location (HL)
BIT b, (IX+d)	Test BIT b of location (IX+d)
BIT b, (IY+d)	Test BIT b of location (IY+d)
BIT b, r	Test BIT b of Reg. r
CALL cc, nn	Call subroutine at location nn if condition cc if true
CALL nn	Unconditional call subroutine at location nn
CCF	Complement carry flag
CP s	Compare operand s with Acc.
CPD	Compare location (HL) and Acc. decrement HL and BC
CPDR	Compare location (HL) and Acc. decrement HL and BC, repeat until BC=0
CPI	Compare location (HL) and Acc. increment HL and decrement BC
CPIR	Compare location (HL) and Acc. increment HL, decrement BC repeat until BC=0
CPL	Complement Acc. (1's comp)
DAA	Decimal adjust Acc.
DEC m	Decrement operand m
DEC IX	Decrement IX
DEC IY	Decrement IY
DEC ss	Decrement Reg. pair ss
DI	Disable interrupts
DJNZ e	Decrement B and Jump relative if B≠0
EI	Enable interrupts
EX (SP), HL	Exchange the location (SP) and HL
EX (SP), IX	Exchange the location (SP) and IX
EX (SP), IY	Exchange the location (SP) and IY
EX AF, AF'	Exchange the contents of AF and AF'
EX DE, HL	Exchange the contents of DE and HL
EXX	Exchange the contents of BC, DE, HL with contents of BC', DE', HL' respectively
HALT	HALT (wait for interrupt or reset)
IM 0	Set interrupt mode 0
IM 1	Set interrupt mode 1
IM 2	Set interrupt mode 2
IN A, (n)	Load the Acc. with input from device n
IN r, (C)	Load the Reg. r with input from device (C)
INC (HL)	Increment location (HL)
INC IX	Increment IX
INC (IX+d)	Increment location (IX+d)
INC IY	Increment IY
INC (IY+d)	Increment location (IY+d)
INC r	Increment Reg. r
INC ss	Increment Reg. pair ss
IND	Load location (HL) with input from port (C), decrement HL and B
INDR	Load location (HL) with input from port (C), decrement HL and decrement B, repeat until B=0
INI	Load location (HL) with input from port (C); and increment HL and decrement B

INIR	Load location (HL) with input from port (C), increment HL and decrement B, repeat until B=0
JP (HL)	Unconditional Jump to (HL)
JP (IX)	Unconditional Jump to (IX)
JP (IY)	Unconditonal Jump to (IY)
JP cc, nn	Jump to location nn if condition cc is true
JP nn	Unconditional jump to location nn
JP C, e	Jump relative to PC+e if carry=1
JR e	Unconditional Jump relative to PC+e
JP NC, e	Jump relative to PC+e if carry=0
JR NZ, e	Jump relative to PC+e if non zero (Z=0)
JR Z, e	Jump relative to PC+e if zero (Z=1)
LD A, (BC)	Load Acc. with location (BC)
LD A, (DE)	Load Acc. with location (DE)
LD A, I	Load Acc. with I
LD A, (nn)	Load Acc. with location nn
LD A, R	Load Acc. with Reg. R
LD (BC), A	Load location (BC) with Acc.
LD (DE), A	Load location (DE) with Acc.
LD (HL), n	Load location (HL) with value n
LD dd, nn	Load Reg. pair dd with value nn
LD HL, (nn)	Load HL with location (nn)
LD (HL), r	Load location (HL) with Reg. r
LD I, A	Load I with Acc.
LF IX, nn	Load IX with value nn
LD IX, (nn)	Load IX with location (nn)
LD (IX+d), n	Load location (IX+d) with value n
LD (IX+d), r	Load location (IX+d) with Reg. r
LD IY, nn	Load IY with value nn
LD IY, (nn)	Load IY with location (nn)
LD (IY+d), n	Load location (IY+d) with value n
LD (IY+d), r	Load location (IY+d) with Reg. r
LD (nn), A	Load location (nn) with Acc.
LD (nn), dd	Load location (nn) with Reg. pair dd
LD (nn), HL	Load location (nn) with HL
LD (nn), IX	Load location (nn) with IX
LD (nn), IY	Load location (nn) with IY
LD R, A	Load R with Acc.
LD r, (HL)	Load Reg. r with location (HL)
LD r, (IX+d)	Load Reg. r with location (IX+d)
LD r, (IY+d)	Load Reg. r with location (IY+d)
LD r, n	Load Reg. r with value n
LD r, r'	Load Reg. r with Reg. r'
LD SP, HL	Load SP with HL
LD SP, IX	Load SP with IX
LD SP, IY	Load SP with IY
LDD	Load location (DE) with location (HL), decrement DE, HL and BC
LDDR	Load location (DE) with location (HL), decrement DE, HL and BC; repeat until BC=0
LDI	Load location (DE) with location (HL), increment DE, HL, decrement BC
LDIR	Load location (DE) with location (HL), increment DE, HL, decrement BC and repeat until BC=0
NEG	Negate Acc. (2's complement)
NOP	No operation
OR s	Logical 'OR' or operand s and Acc.
OTDR	Load output port (C) with location (HL) decrement HL and B, repeat until B=0
OTIR	Load output port (C) with location (HL), increment HL, decrement B, repeat until B=0
OUT (C), r	Load output port (C) with Reg. r
OUT (n), A	Load output port (n) with Acc.
OUTD	Load output port (C) with location (HL), decrement HL and B
OUTI	Load output port (C) with location (HL), increment HL and decrement B

POP IX	Load IX with top of stack
POP IY	Load IY with top of stack
POP qq	Load Reg. pair qq with top of stack
PUSH IX	Load IX onto stack
PUSH IY	Load IY onto stack
PUSH qq	Load Reg. pair qq onto stack
RES b, m	Reset Bit b of operand m
RET	Return from subroutine
RET cc	Return from subroutine if condition cc is true
RETI	Return from interrupt
RETN	Return from non maskable interrupt
RL m	Rotate left through carry operand m
RLA	Rotate left Acc. through carry
RLC (HL)	Rotate location (HL) left circular
RLC (IX+d)	Rotate location (IX+d) left circular
RLC (IY+d)	Rotate location (IY+d) left circular
RLC r	Rotate Reg. r left circular
RLCA	Rotate left circular Acc.
RLD	Rotate digit left and right between Acc. and location (HL)
RR m	Rotate right through carry operand m
RRA	Rotate right Acc. through carry
RRC m	Rotate operand m right circular
RRCA	Rotate right circular Acc.
RRD	Rotate digit right and left between Acc. and location (HL)
RST p	Restart to location p
SBC A, s	Subtract operand s from Acc. with carry
SBC HL, ss	Subtract Reg. pair ss from HL with carry
SCF	Set carry flag (C=1)
SET b, (HL)	Set Bit b of location (HL)
SET b, (IX+d)	Set Bit b of location (IX+d)
SET b, (IY+d)	Set Bit b of location (IY+d)
SET b, r	Set Bit b of Reg. r
SLA m	Shift operand m left arithmetic
SRA m	Shift operand m right arithmetic
SRL m	Shift operand m right logical
SUB s	Subtract operand s from Acc.
XOR s	Exclusive 'OR' operand s and Acc.

DIVISION DE EDUCACION CONTINUA
FACULTAD DE INGENIERIA U.N.A.M.

MICROPROCESADORES Y MICROCOMPUTADORAS

ANEXO 1 ARTICULOS VARIOS

NOVIEMBRE, 1984.

Palacio de Mineria Calle de Tacuba 5 primer piso Deleg. Cuauhtémoc 06000 México, D.F. Tel.: 521-40-20 Apdo. Postal M-2285

PROBLEMAS Y EJERCICIOS DE MEMORIAS ROM Y RAM

1. PROGRAMACION DEL EPROM 2708

El EPROM 2708 fue el primer EPROM del tipo NMOS. Para programarlo se le deben proporcionar tanto las direcciones como los datos en niveles TTL, una fuente de 12 volts constante y el pulso de programación de 25 volts que puede variar, según el número de iteraciones que se realice en todas las posiciones, entre 0,1 mseg y 1 mseg. El tiempo de programación por posición debe ser mayor o igual a 100 mseg. Calcule:

De acuerdo con los diagramas de tiempos del 2708 cual es el tiempo de programación, considerando todos los elementos que intervienen en el cálculo y un tiempo de programación por posición de 100 mseg.

2. PROGRAMADOR DE EPROMs 2716 y 2732

Diseñar un programador de EPROMs para los chips 2716 y 2732. Considere que se dispone de un microprocesador Z80 al cual se le puede conectar cualquier dispositivo para tal efecto, es decir que se puede usar sin restricciones los buses de dirección, datos y control. Usar la menor cantidad de hardware posible, todas las actividades y secuencias de la programación deben realizarse por software. Mostrar el diagrama del hardware que se requiere y el diagrama de flujo del software para la programación y verificación. Se deben poder programar todas las posiciones en secuencia o bien posiciones aisladas. Use los puertos que se requieran entre 80H y 90H. Por medio de switches seleccione manualmente si se trata del 2716 o 2732.

3. REDUCCION DEL CONSUMO DE POTENCIA DEL ROM

Los circuitos ROM y sus derivados son dispositivos que por su naturaleza nunca van a perder la información que tienen almacenada, sin embargo, consumen gran cantidad de energía, por lo que se presenta como una buena alternativa quitarles la alimentación de la energía eléctrica mientras no se usan. Esto reduce el consumo de potencia de los circuitos donde intervienen sin afectar prácticamente la velocidad de operación.

Diseñar un circuito que alimente de energía eléctrica a los EPROM 2732 de un sistema en el cual los primeros 16 Kbytes, de un espacio de direccionamiento de 64 Kbytes, son del tipo memoria de lectura solamente. La alimentación de energía

eléctrica a los EPROMs deberá realizarse solamente cuando sea necesario.

4. GENERADOR DE FUNCIONES PROGRAMABLE

Diseñar un generador de funciones programable que produsca 4 tipos de ondas, cuadrada, senoidal, triangular y rampa. El generador de funciones está conectado a un microprocesador Z80 el cual programa la forma de onda y la frecuencia a la cual se debe generar la señal analógica.

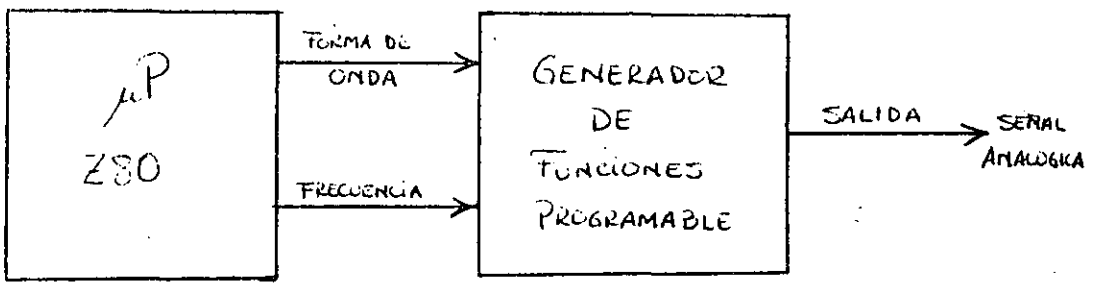


Figure 4-1: Generador de funciones programable

El ciclo de las señales que genera deben tener una resolución de 256 puntos y una definición en el rango dinámico de 8 bits.

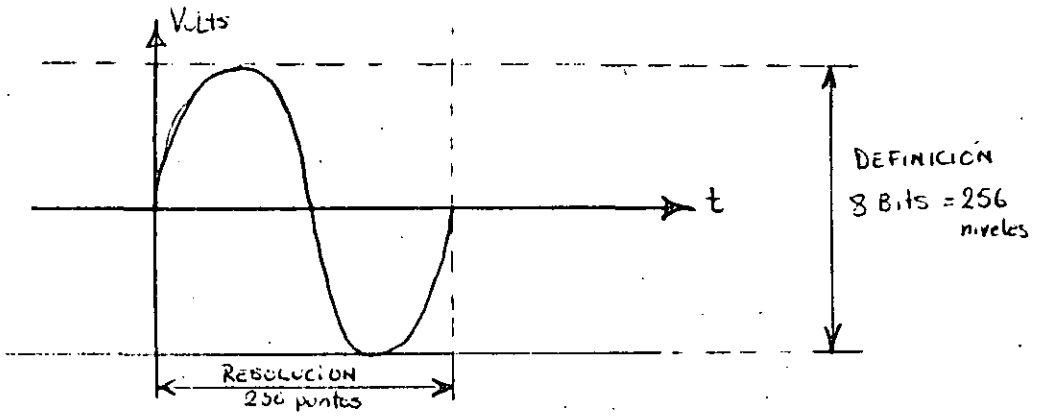


Figure 4-2: Características de la señal de salida

1. Obtener el diagrama de bloques del circuito destinando los números de puertos entre 40 y 50 (hexadecimal).
2. Cual es la frecuencia máxima a la cual pueden oscilar las señales de salida considerando que se usa un EPROM 2716 con 250 nseg de tiempo de acceso.
3. Cuantas y a que frecuencias se pueden generar las señales de salida.

5. DECODIFICADOR DE MEMORIA

Diseñar un decodificador de memoria para Z80 con las siguientes características:

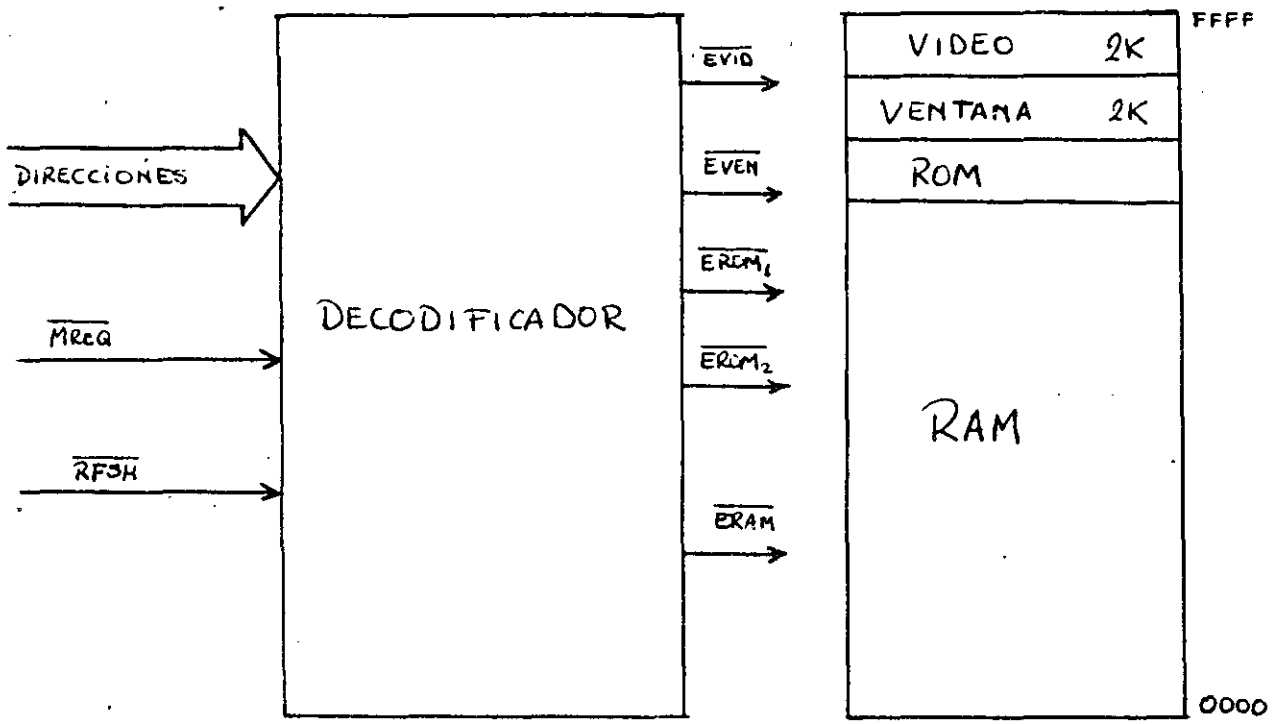


Figure 5-1: a. Decodificador de memoria b. Mapa de memoria

La parte de video se usa para desplegar en un CRT el contenido de memoria traducido en caracteres ASCII, la parte de ventana permite espiar una memoria mucho más grande, en la parte de ROM se usan 2 chips 2716 y para la parte de RAM se usan chips de 64 Kbits dinámicos. Toda salida se habilita solo cuando MREQ está habilitado. Siempre que se presente RFSH no importa cual sea la dirección debe habilitarse ERAM. De acuerdo con la operación del Z80 se asegura que cuando se presenta RFSH siempre se presenta MREQ.

1. Obtener el diseño del decodificador con la menor cantidad de chips posible.

2. Ajustar este decodificador para usarse con el sistema operativo CP/M. La restricción más grande radica en que la parte inferior de la memoria debe permanecer libre es decir debe ser RAM, no se puede tener un ROM permanente en las partes más inferiores de la memoria.

Sin embargo, el Z80 después de RESET, arranca ejecutando las instrucciones que se encuentran a partir de la dirección 0. Esto crea un gran conflicto ya que no se puede ejecutar la función de autoreset que requiere un ROM en la parte inferior de memoria.

3. Los chips de memoria RAM pueden por si solos llenar todo el espacio de direccionamiento del Z80, por lo que en esta aplicación se desperdician 8 Kbytes que son las partes de video, ventana y ROM. Diseñar un mecanismo que permita aprovechar esos 8 Kbytes de memoria RAM que en este momento no se usan.

4. La zona de ventana permite espiar una gran memoria adicional que algunos fabricantes han dado por llamarla RAM-disk por la magnitud de la memoria y por su principio de operación. Diseñar una memoria masiva de semiconductores (RAM-disk) de 512 Kbytes de la cual se pueden espiar cualquier conjunto contiguo de 2 Kbytes a través de la ventana. Es importante notar que esta memoria masiva no interfiere con el resto de la memoria principal de 64 Kbytes.

6. UNIDAD MANEJADORA DE MEMORIA

Una unidad manejadora de memoria tiene varias funciones, entre las que destacan:

- Transformar o mapear direcciones virtuales a físicas, es decir aislar el espacio de direccionamiento del procesador.
- Dividir la memoria en páginas (bloques de tamaño fijo) o segmentos (bloques de tamaño variable y traslapables).
- Checar que las direcciones que se envian pertenecen a la página o segmento de que se trate.
- Checar zonas donde no existe memoria fisicamente.
- Checar los atributos de las páginas o segmentos.
- Toda anormalidad que se trate de realizar durante el acceso la reporta al CPU mediante un trap.

Diseñar una unidad manejadora de memoria en forma discreta que cumpla con todas estas funciones, la cual maneja 512

segmentos de hasta 12 Kbytes cada uno. Se debe poder especificar el tamaño de los segmentos, los atributos de cada segmento y el inicio de cada segmento que puede ser en cualquier posición de memoria. Los atributos de los segmentos son:

- R/RW lectura solamente o lectura y escritura.
- S/U sistema o usuario.
- C/D código o datos.
- I/M intacto o modificado.
- E/NE existe o no existe fisicamente.
- D/T definitivo o temporal.
- P/NP protegido o no protegido.
- L/O libre u ocupado.

7. MEMORIA ESTATICA USANDO CHIPS DINAMICOS

Diseñar una tarjeta de memoria de 128 K x 32, capáz de operar en modo byte o modo palabra, usando chips de 64 K x 1 (4164, 8264, etc.). Aunque está diseñada con chips de memoria dinámicos desde afuera se ve como estática, es decir no necesita refrescamiento externo. Los conflictos entre el procesador externo y el refrescador interno se resuelven por orden de solicitud, es decir, el primero que solicita la memoria la usa y el otro espera a que termine, si solicitan al mismo tiempo la usa el procesador externo. El refresco a los renglones debe efectuarse cada 2 mseg y la forma de realizarlo es dividir 2 mseg entre el número de renglones y ese es el intervalo entre renglones consecutivos, ver fig 7-1.

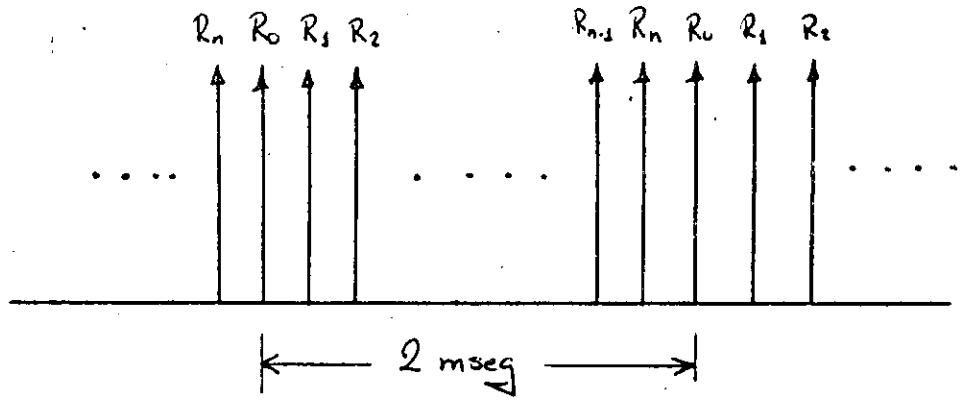


Figure 7-1: Distribución en el tiempo del refresco

1. Cual es la frecuencia de refrescamiento entre renglones?.
2. Diseñar la memoria a nivel de bloques.
3. Especificar las características internas de cada bloque por medio de tablas de verdad, diagramas de estados, diagramas de tiempos, etc.

8. MANEJADOR DE ACCESO DIRECTO A MEMORIA

EL manejador de acceso directo a memoria (DMA) es un dispositivo que controla las transferencias de datos de memoria a periféricos o de periféricos a memoria sin intervención del procesador central. Este dispositivo solicita previamente utilizar el bus al procesador central (CPU), el cual se lo concede una vez que ha terminado de ejecutar sus actividades pendientes. El CPU cuando concede el bus suelta todas las líneas

de dirección, datos y control para que el manejador de DMA tome el mando del bus a partir de ese momento. Una vez terminada la transferencia de datos el manejador de DMA debe soltar el bus y avisar al CPU para que este continue con sus actividades.

Diseñar un manejador de DMA que sea programable por el CPU y pueda transferir datos de memoria a periféricos o viceversa. Se debe poder programar en el manejador de DMA la cantidad de datos a transferir (máximo 64 K), la dirección inicial del bloque de datos en memoria (de 0 a 16 M) y el sentido de la transferencia, asimismo el manejador de DMA debe manejar un protocolo de comunicación con los periféricos que controla.

EJERCICIOS DE ACOPLAMIENTO ENTRE DISPOSITIVOS

1. FLUJO DE CORRIENTES ENTRE ETAPAS TTL

1. ?Como se acoplan las compuertas TTL?
2. ?porque existen corrientes positivas y negativas en las especificaciones de todo circuito digital?

2. ACOPLAMIENTO ENTRE COMPUERTAS TTL

1. ?Que significa fan-out y a que se denomina factor de carga en la tabla 2-1?
2. ?Cual es el consumo de potencia promedio y máximo en los diferentes chips TTL?

3. CALCULO DE FAN-OUT

1. Determinar el fan-out entre compuertas del tipo 74 ---> 74LS. En otras palabras: ?cuantas entradas 74LS puede manejar una salida 74?
2. Determinar el fan-out entre 74LS y 74L en ambos casos.

4. DISPOSITIVOS DE 3 ESTADOS

Los dispositivos de tres estados además de manejar los dos estados lógicos 0 y 1 tienen un tercer estado de alta impedancia que no es ninguno de los dos anteriores. Estas dispositivos sirven para formar buses, es decir conectar las salidas de varias compuertas a la misma línea con el fin de que la línea pueda ser manejada por cualquiera de esas compuertas.

1. ?cuales son las características de las compuertas de tres estados?
2. ?que significa corriente de alta impedancia?

5. MANEJO DE BUFFERS DRIVERS

1. ?Que significa buffer y driver?

2. ?Cuantas compuertas de tres estados del tipo 74LS125 pueden conectar sus salidas a una sola entrada 74LS?

3. ?cuantas compuertas 74LS244 pueden conectar sus salidas a una sola línea que tiene conectada 10 entradas del tipo 74?

6. CARACTERISTICAS DE DC DE MEMORIAS

Los chips de memoria del tipo MOS consumen muy poca cantidad de corriente en las señales de entrada y tienen capacidad de manejar cierta cantidad de corriente en las señales de salida (bus de datos). Las líneas de entrada mas que cargas resistivas representan cargas capacitivas, por lo que en lugar de afectar la salida que maneja la línea se retrasan los tiempos de propagación. ?a que se denomina corriente de fuga de entrada?

7. MANEJO DE CARGAS DE LOS CHIPS DE MEMORIA

Desde el punto de vista resistivo y capacitivo (según condiciones de prueba de las especificaciones) calcular:

1. ?Cuantos chips de memoria del tipo 2114 se pueden conectar juntos en un solo bus a una carga TTL 74?

2. Una tarjeta de memoria tiene las características de la fig 7-1, usa chips de memoria del tipo 2102 y maneja las siguientes corrientes I(OL) = 2.1 mA, I(OH) = -100 uA. Que se puede hacer para solucionar el problema del manejo de las cargas en el bus de datos?

3.

8. CARACTERISTICAS DE DC DE LOS MICROS

Los microprocesadores generalmente son del tipo MOS, por lo que su comportamiento, desde el punto de vista del manejo de cargas, es semejante al de las memorias. ?cuantas cargas del tipo 74LS puede manejar el micro Z80?

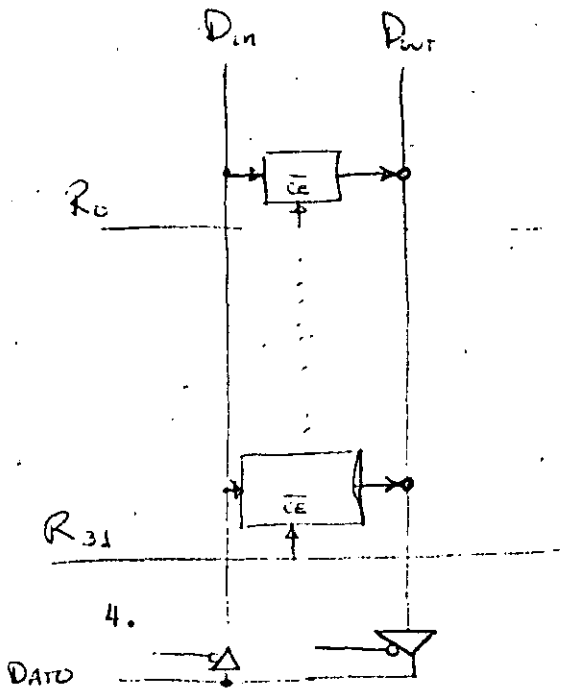


Figure 7-1: ~~Terminaciones de las líneas del bus~~
ORGANIZACION INTERNA DE MEMORIA.

5.

9. MANEJO DE CARGAS DEL Z80

?De acuerdo con el siguiente diagrama, determinar si un Z80 puede operar sin necesidad de "drivear" las señales?

10. MANEJO DE BUSES Y TERMINACIONES

Los buses son canales o vías de comunicación que usan los diferentes componentes de una computadora para interactuar entre sí. Los buses físicamente adquieren forma caprichosas y pasan por diversos elementos (conectores, peines, cables, etc.), en su trayectoria tienen diversos contactos mecánicos que pueden causar ciertas pérdidas en la línea y formar capacitancias. Los buses son manejados generalmente por compuertas bipolares que tienen la capacidad de manejar gran cantidad de corriente y son afectados menos por las cargas capacitivas.

Pese a que las compuertas manejadoras del bus pueden soportar impedancias características en la línea de hasta 50 ohms, en la transferencia de los datos se tropieza con el grave problema de las reflexiones, es decir, si la línea no está terminada con la impedancia característica la señal puede rebotar de un lado para otro ocacionando graves problemas sobre todo si se trata de líneas largas. Los fabricantes generalmente sugieren terminar las líneas de los buses de la siguiente manera:

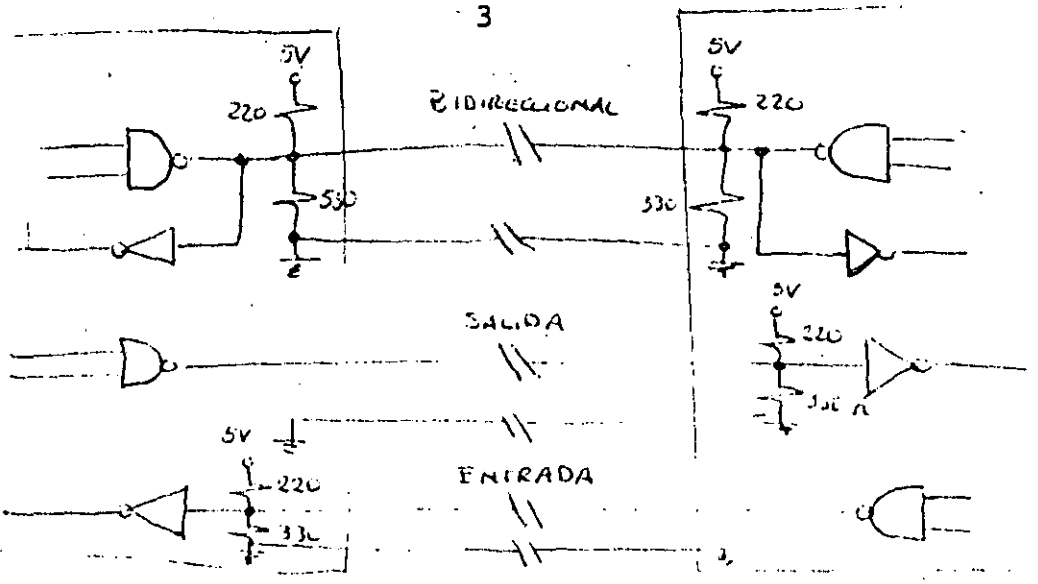
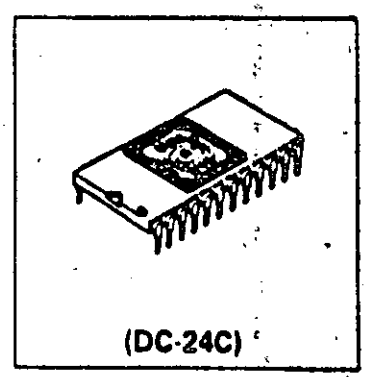


Figure 10-1: Terminaciones de las líneas del bus

HN462732

4096-word × 8-bit UV Erasable and Programmable Read Only Memory

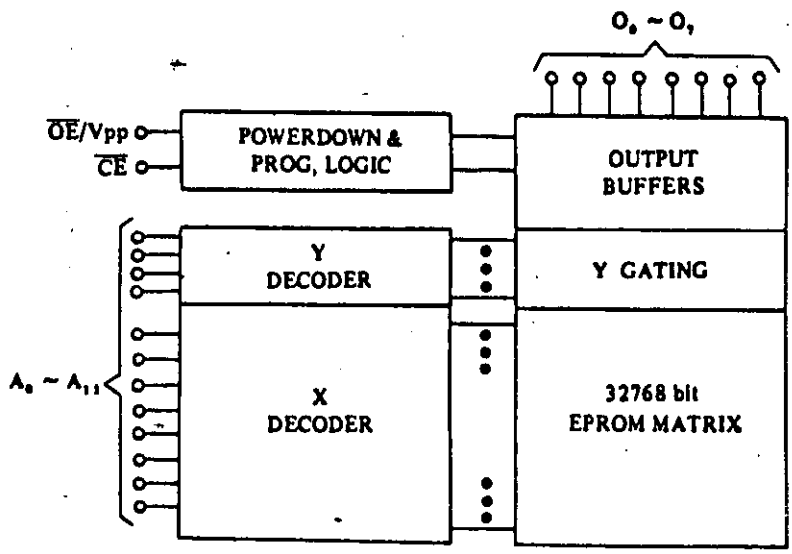
The HN462732 is a 4096 word by 8 bit erasable and electrically programmable ROM. This device is packaged in a 24-pin, dual-in-line package with transparent lid. The transparent lid allows the user to expose the chip to ultraviolet light to erase the bit pattern, whereby a new pattern can then be written into the device.



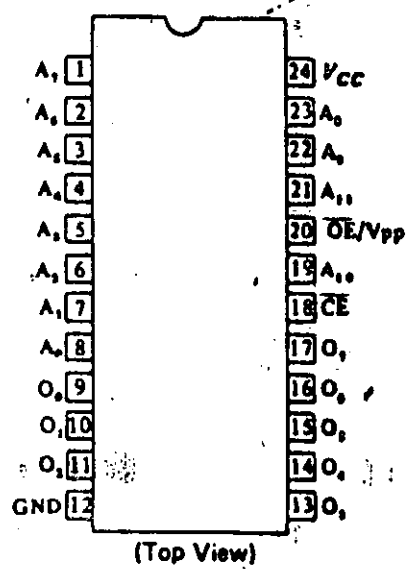
FEATURES

- Single Power Supply . . . . . +5V ±5%
- Simple Programming . . . . . Program Voltage: +25V D.C.
  Program with One 50ms Pulse
- Static . . . . . No Clocks Required
- Inputs and Outputs TTL Compatible During Both Read and Program Modes
- Fully Decoded On-Chip Address Decode
- Access Time . . . . . 450ns Max.
- Low Power Dissipation . . . . . 150mA Max. Active Current
  30mA Max. Standby Current
- Three State Output . . . . . OR-Tie-Capability
- Compatible with INTEL 2732

BLOCK DIAGRAM



PIN ARRANGEMENT



■ MODE SELECTION

Mode \ Pins	$\overline{CE}$ (18)	$\overline{OE}/V_{PP}$ (20)	V_{CC} (24)	Outputs (9 ~ 11, 13 ~ 17)
Read	V_{IL}	V_{IL}	+5	Dout
Stand by	V_{IH}	Don't Care	+5	High Z
Program	V_{IL}	V_{PP}	+5	Din
Program Verify	V_{IL}	V_{IL}	+5	Dout
Program Inhibit	V_{IH}	V_{PP}	+5	High Z

■ ABSOLUTE MAXIMUM RATINGS

Item	Symbol	Value	Unit
Operating Temperature Range	T_{opr}	0 to + 70	°C
Storage Temperature Range	T_{stg}	−65 to + 125	°C
All Input and Output Voltages*	V_{IN}, V_{out}	−0.3 to + 7	V
V_{PP} Voltage*	$\overline{OE}/V_{PP}$	−0.3 to +28	V

*with respect to GND

■ READ OPERATION

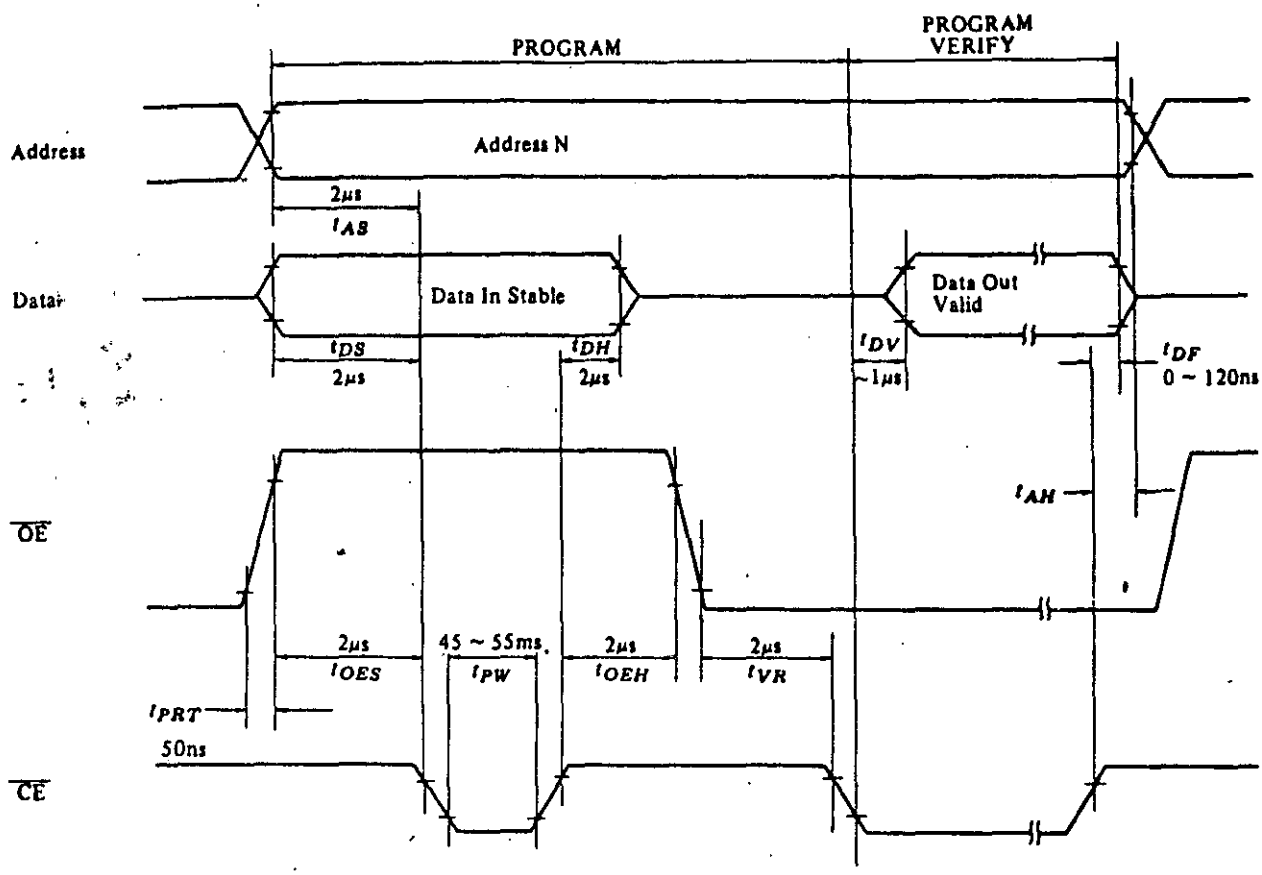
• D. C. AND OPERATING CHARACTERISTICS (T_a=0 to +70°C, V_{CC}=5V±5%)

Parameter	Symbol	Test Conditions	min.	typ.	max.	Unit
Input Leakage Current (Except $\overline{OE}/V_{PP}$)	I_{LI1}	V_{IN} = 5.25 V	–	–	10	μA
$\overline{OE}/V_{PP}$ Input Leakage Current	I_{LI2}	V_{IN} = 5.25 V	–	–	300	μA
Output Leakage Current	I_{LO}	V_{out} = 5.25 V	–	–	10	μA
V_{CC} Current (Standby)	I_{CC1}	$\overline{CE}$ = V_{IH}, $\overline{OE}$ = V_{IL}	–	–	30	mA
V_{CC} Current (Active)	I_{CC2}	$\overline{OE}$ = $\overline{CE}$ = V_{IL}	–	–	150	mA
Input Low Voltage	V_{IL}		−0.1	–	0.8	V
Input High Voltage	V_{IH}		2.0	–	V_{CC} + 1	V
Output Low Voltage	V_{OL}	I_{OL} = 2.1 mA	–	–	0.45	V
Output High Voltage	V_{OH}	I_{OH} = −400 μA	2.4	–	–	V

• A. C. CHARACTERISTICS (T_a=0 to +70°C, V_{CC}=5V±5%)

Parameter	Symbol	Test Conditions	min.	typ.	max.	Unit
Address to Output Delay	t_{ACC}	$\overline{CE}$ = $\overline{OE}$ = V_{IL}	–	–	450	ns
$\overline{CE}$ to Output Delay	t_{CE}	$\overline{OE}$ = V_{IL}	–	–	450	ns
Output Enable to Output Delay	t_{OE}	$\overline{CE}$ = V_{IL}	–	–	120	ns
Output Enable High to Output Float	t_{DF}	$\overline{CE}$ = V_{IL}	0	–	100	ns
Address to Output Hold	t_{OH}	$\overline{CE}$ = $\overline{OE}$ = V_{IL}	0	–	–	ns

• PROGRAMMING WAVE FORMS



• ERASE

Erasure of HN462732 is performed by exposure to Ultraviolet light of 2537Å, and all the output data are changed to "1" after this prosedure.

The minimum integrated close (i.e., UV intensity x exposure time) for erasure is 15W · sec/cm^2.

Description (Continued)

to various portions of the memory, and from the need to structure large, complex programs and systems.

Multiple tasks (or users) of a system that can reside anywhere in memory are called *relocatable*. Generally, systems in which all tasks are relocatable offer far greater flexibility in responding to changing system environments. Another aspect of multiple-task environments is sharing: separate tasks can execute the same program on different data, or several tasks may execute different programs using the same data.

Unfortunately, a problem that arises in multiple-task systems is that of system integrity. Tasks must be protected from unwanted interactions with other tasks; user tasks must be prohibited from performing operating system functions; and user tasks must also be protected from themselves so they cannot overflow the areas allotted to them.

In addition to these considerations, support for the design and implementation of large, complex programs and systems is itself an important consideration. Modern trends are toward the partitioning of a complex task into small, simple, self-contained subtasks that have well-defined interfaces. Because these subtasks interact with each other, communication between them must be carefully controlled. Memory-management systems can offer effective solutions for implementing large systems modularly designed.

The Z8010 Memory Management Unit supports multiple-process and large modular software systems with dynamic segment relocation. Futhermore, it enhances system integrity with a powerful set of memory protection features.

Relocation. Dynamic segment relocation makes user software addresses independent of the physical memory addresses, thereby freeing the user from specifying where information is actually located in the physical memory and providing a flexible, efficient method for supporting multi-programming systems.

The Z-MMU uses a translation table to transform the 23-bit logical addresses from the Z8001 CPU into 24-bit addresses for the physical memory. Memory segments are variable in size from 256 bytes to 64K, in increments of 256 bytes. Pairs of Z-MMUs support the 128 segment numbers available for the various Z8001 CPU address spaces. Within an address space, any number of Z-MMUs can be used to accommodate multiple translation tables for system and normal operating modes, or to support more sophisticated memory-management systems.

System Integrity. Z-MMU memory-protection features safeguard memory areas from unauthorized or unintended access by associating special access restrictions with each segment. A segment is assigned a "personality" consisting of several attributes when it is initially entered into the Z-MMU. When a memory reference is made, these attributes are checked against the status information supplied by the Z8001 CPU. If a mismatch occurs, a trap is generated and the CPU is interrupted. The CPU can then check the status registers of the MMU to determine the cause and take appropriate action to correct the problem.

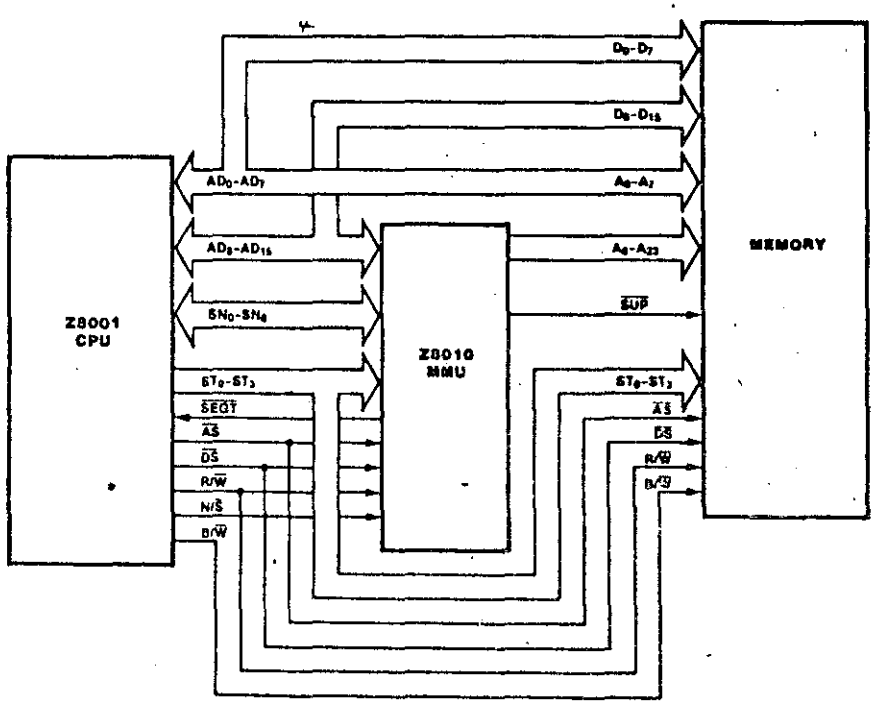


Figure 3. The MMU in a Z8000 System

The Fundamentals of Memory Management (continued)

Generally, segments can be of variable size, within limits, and a user can specify the size of each segment to be used. Thus one user may have two segments of two thousand and ten thousand words for his FORTRAN program and data, respectively, while another user might have three segments of three thousand, six thousand and two thousand words for her PASCAL program, data, and run-time stack. If the first user called his data segment number 5, then the first word in his data set would be accessed by the logical address (5,0) indicating segment 5, offset 0. The memory management system translates this symbolic name into the correct physical memory address.

Figure 1 gives a conceptual realization of these two users' logical program spaces. The first user, User A, has his program segment called "Segment 6" and his data segment called "Segment 5." The second user, User B, has her program segment called "Segment 5," her data segment called "Segment 12" and her stack segment called "Segment 2." Notice that both users have named one of their segments "Segment 5," but they refer to different entities. This causes no problem since the system keeps the two memory areas separate. The situation is analogous to both users having an integer variable called "I" in their programs: The system realizes that these are two separate variables stored in different memory locations.

User A's data segment, "Segment 5," is ten thousand words. If he references word 10,050 of Segment 5 he gets an error message from the system indicating that he has exceeded the allocation limit for Segment 5. Note that he does not access word 50 of Segment 6. That is, segments are logically distinct and unordered. A reference to one segment cannot inadvertently result in access to another segment. Thus, in this example, User A is prevented from accidentally (or deliberately) accessing his program as though it were part of his data segment.

Figure 2 illustrates one way that these segments could be arranged in the physical memory. The dotted lines indicate the memory-mapping function from the logical address space of the user to the physical memory locations allocated to him. The figure also indicates the access attributes associated with each user's segments. For example, program segments are "execute only" and data segments are "read/write." Thus a user is prevented from executing a data segment or writing into a code segment.

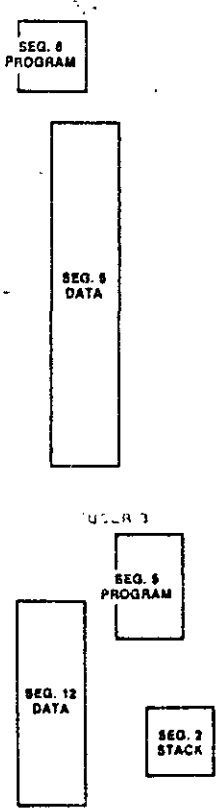


Figure 1. Two User's Logical Address Space

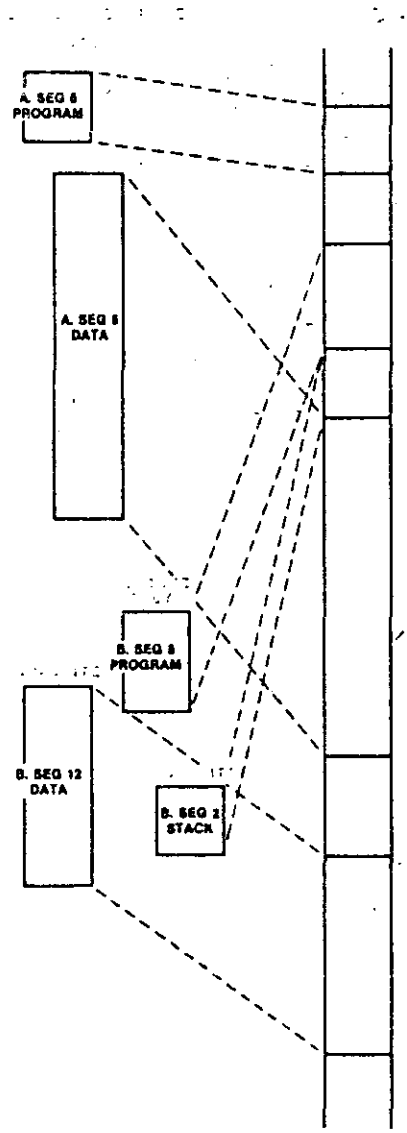


Figure 2. Mapping Logical Segments to Physical Memory

The Fundamentals of Memory Management (Continued)

Figure 3 illustrates what happens when both users have access to the same data set in primary memory, say the results of a questionnaire that both intend to analyze. Each user has a logical name associated with that data set to specify the segment in which the data set is to reside. Note that the two users have chosen to put the data set in different segments of their personal address spaces. The system-mapping function translates these different segment names to the same physical memory locations. Thus User A's access to address (2, 17) references the same physical memory location as User B's access to address (7, 17). In the figure, note that two of B's segments have been moved in physical memory to create a space large enough to hold the questionnaire data.

Another topic in memory management that is supported by Z8001-Z8010 architecture but requires additional support hardware is demand swapping, or segmented virtual memory, which means that the logical memory area may not actually reside in physical memory until a task actually tries to access it. At the time an access is made to a segment missing from physical memory, the instruction execution is held in abeyance until the logical memory can be brought into the physical memory and then the instruction is allowed to proceed with the memory access. The address translation is performed, access protection is checked and the instruction proceeds as if the logical memory area had been in the physical memory at the beginning of the instruction. The instructions in the Z8001 must run to completion before the CPU can perform any action, such as responding to a missing segment trap. But with the conjunction of hardware and software to simulate the above functions, a segmented virtual memory scheme can be implemented.

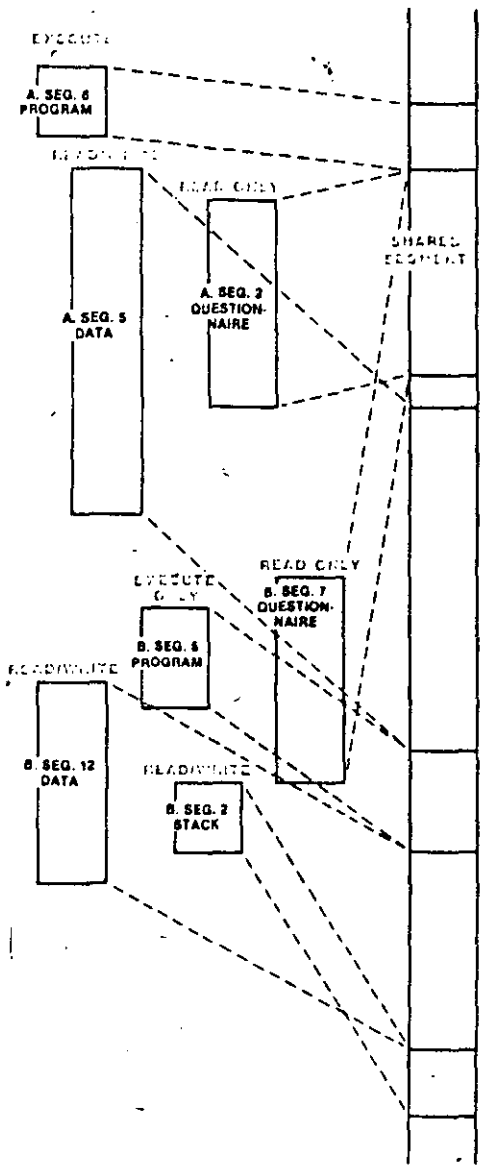


Figure 3. Two Users Sharing a Common Segment

A final topic in memory management is paging, which is another method for partitioning a user address space and mapping it onto the physical memory. Paging is most effective when demand swapping can be supported. Essentially, paging divides the logical memory into fixed-size blocks, called pages. Like segments, the individual pages can be located anywhere in the physical memory and a translation mechanism maps logical addresses to physical memory locations. There are two differences between paging and segmenting a logical memory. First, pages are of fixed size whereas segments are of various sizes. Second, under paging, the logical memory is still linear, that is, a task accesses memory using a single number, rather than a pair as in segmentation. The major advantage of paging is in treating memory as blocks of fixed sizes, which simplifies allocating memory to users and deciding where to place the logical pages in physical memory. The major disadvantage of paging is in assigning different protection attributes to different areas in a user address space because a paged memory appears homogeneous to the user and the operating system. Paging can be combined with segmentation to produce a memory management system with the advantages of both paging and segmentation. The implementation of paging for the Z8001 requires additional support hardware and may be implemented independent of the Z8010.

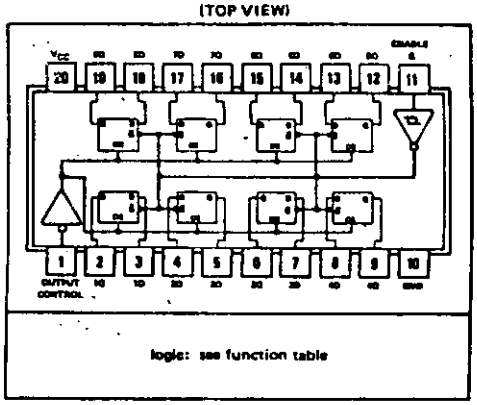
Before proceeding to the mechanism of memory management, it is instructive to review how a segmented address translation mechanism with protection attributes achieves the five major goals of memory management outlined in the previous section. The first goal permits dynamic allocation of memory during the execution of tasks; that is, a task could be located anywhere in memory and even moved about when its execution is suspended. The address translation mechanism provides this flexibility because the task deals exclusively

TYPES SN54LS373, SN54LS374, SN54S373, SN54S374, SN74LS373, SN74LS374, SN74S373, SN74S374 OCTAL D-TYPE TRANSPARENT LATCHES AND EDGE-TRIGGERED FLIP-FLOPS

BULLETIN NO. DL-S 7712350, OCTOBER 1976—REVISED AUGUST 1977

- Choice of 8 Latches or 8 D-Type Flip-Flops In a Single Package
- 3-State Bus-Driving Outputs
- Full Parallel-Access for Loading
- Buffered Control Inputs
- Clock/Enable Input Has Hysteresis to Improve Noise Rejection
- P-N-P Inputs Reduce D-C Loading on Data Lines ('S373 and 'S374)
- SN54LS363 and SN74LS364 Are Similar But Have Higher V_{OH} For MOS Interface

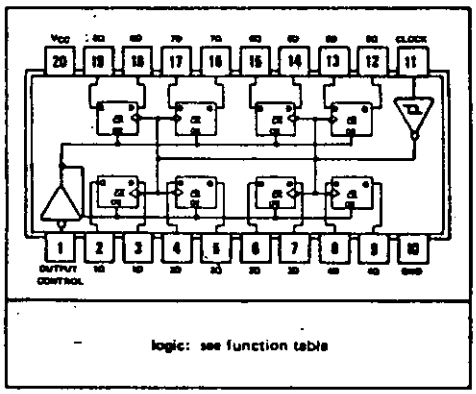
SN54LS373, SN54S373 . . . J PACKAGE
SN74LS373, SN74S373 . . . J OR N PACKAGE
(TOP VIEW)



'LS373, 'S373
FUNCTION TABLE

OUTPUT CONTROL	ENABLE G	D	OUTPUT
L	H	H	H
L	H	L	L
L	L	X	Q_0
H	X	X	Z

SN54LS374, SN54S374 . . . J PACKAGE
SN74LS374, SN74S374 . . . J OR N PACKAGE
(TOP VIEW)



'LS374, 'S374
FUNCTION TABLE

OUTPUT CONTROL	CLOCK	D	OUTPUT
L	↑	H	H
L	↑	L	L
L	L	X	Q_0
H	X	X	Z

See explanation of function tables on page 3-8.

description

These 8-bit registers feature totem-pole three-state outputs designed specifically for driving highly-capacitive or relatively low-impedance loads. The high-impedance third state and increased high-logic-level drive provide these registers with the capability of being connected directly to and driving the bus lines in a bus-organized system without need for interface or pull-up components. They are particularly attractive for implementing buffer registers, I/O ports, bidirectional bus drivers, and working registers.

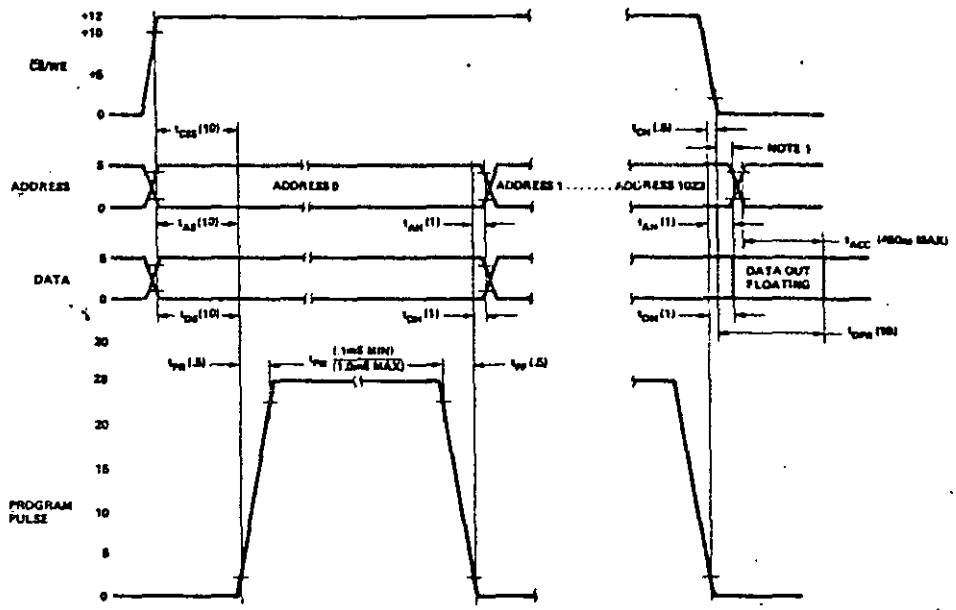
The eight latches of the 'LS373 and 'S373 are transparent D-type latches meaning that while the enable (G) is high the Q outputs will follow the data (D) inputs. When the enable is taken low the output will be latched at the level of the data that was setup.

ERASING THE 2704/2708

The 2704/2708 is erased by exposure to ultra violet light at a wavelength of 2537Å. The recommended integrated dosage (i.e. UV intensity x exposure time) is 10W-sec/CM^2. Listed in Table XIX are several suitable sources and respective erase times for the 2704/2708. The model numbers referred to are manufactured by Ultra-Violet Products, Inc. (5114 Walnut Grove Avenue, San Gabriel, CA). The lamps should be used without short wave filters and placed about one inch from the parts to be erased.

Table XIX: UV Sources for Erasing the 2704/2708.

Model	Power Rating	Typical Time to Erase a 2708 Device
S-68	12000 uW/CM2	10 minutes
S-52	12000 uW/CM2	10 minutes
UVS-54	5700 uW/CM2	30 minutes
R-52	13000 uW/CM2	10 minutes
UVS-11	5500 uW/CM2	30 minutes



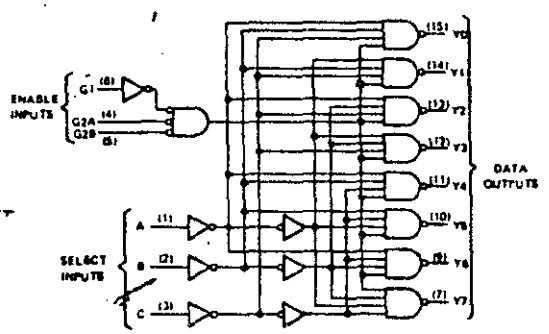
NOTE 1. THE $\overline{CS}/WE$ TRANSITION MUST OCCUR AFTER THE PROGRAM PULSE TRANSITION AND BEFORE THE ADDRESS TRANSITION.

2. NUMBERS IN ( ) INDICATE MINIMUM TIMING IN μS UNLESS OTHERWISE SPECIFIED.

Figure 39. 2704/2708 Programming Waveforms.

functional block diagrams and logic

'LS138, 'S138



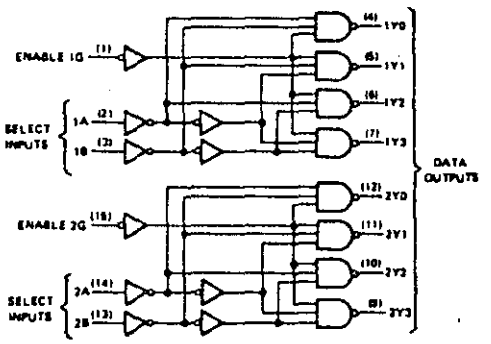
'LS138, 'S138
FUNCTION TABLE

INPUTS					OUTPUTS							
ENABLE		SELECT										
G1	G2*	C	B	A	Y0	Y1	Y2	Y3	Y4	Y5	Y6	Y7
X	H	X	X	X	H	H	H	H	H	H	H	H
L	X	X	X	X	H	H	H	H	H	H	H	H
H	L	L	L	L	L	H	H	H	H	H	H	H
H	L	L	L	H	H	L	H	H	H	H	H	H
H	L	L	H	L	H	H	L	H	H	H	H	H
H	L	L	H	H	H	H	H	L	H	H	H	H
H	L	H	L	L	H	H	H	H	L	H	H	H
H	L	H	L	H	H	H	H	H	H	L	H	H
H	L	H	H	L	H	H	H	H	H	H	L	H
H	L	H	H	H	H	H	H	H	H	H	H	L

*G2 = G2A + G2B

H = high level, L = low level, X = irrelevant

'LS139, 'S139



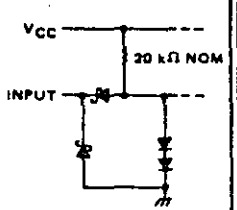
'LS139, 'S139
(EACH DECODER/DEMULTIPLEXER)
FUNCTION TABLE

INPUTS			OUTPUTS			
ENABLE	SELECT					
G	B	A	Y0	Y1	Y2	Y3
H	X	X	H	H	H	H
L	L	L	L	H	H	H
L	L	H	H	L	H	H
L	H	L	H	H	L	H
L	H	H	H	H	H	L

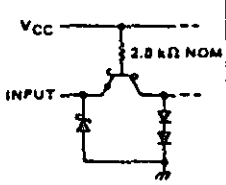
H = high level, L = low level, X = irrelevant

schematics of inputs and outputs

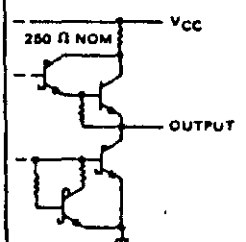
EQUIVALENT OF EACH INPUT OF 'LS138, 'LS139



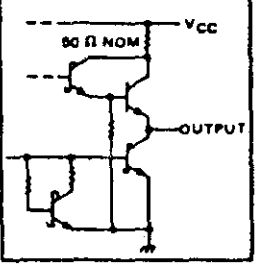
EQUIVALENT OF EACH INPUT OF 'S138, 'S139



TYPICAL OF OUTPUTS OF 'LS138, 'LS139



TYPICAL OF OUTPUTS OF 'S138, 'S139



TYPES SN54157, SN54L157, SN54LS157, SN54LS158, SN54S157, SN54S158, SN74157, SN74L157, SN74LS157, SN74LS158, SN74S157, SN74S158
QUADRUPLE 2-LINE-TO-1-LINE DATA SELECTORS/MULTIPLEXERS

BULLETIN NO. DL-S 7711847, MARCH 1974—REVISED AUGUST 1977

features

- Buffered Inputs and Outputs
- Three Speed/Power Ranges Available

TYPES	TYPICAL AVERAGE PROPAGATION TIME	TYPICAL POWER DISSIPATION
'157	9 ns	150 mW
'L157	18 ns	75 mW
'LS157	9 ns	49 mW
'S157	5 ns	250 mW
'LS158	7 ns	24 mW
'S158	4 ns	195 mW

applications

- Expand Any Data Input Point
- Multiplex Dual Data Buses
- Generate Four Functions of Two Variables (One Variable Is Common)
- Source Programmable Counters

description

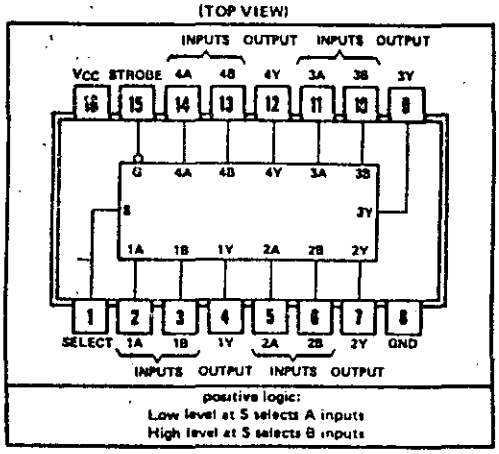
These monolithic data selectors/multiplexers contain inverters and drivers to supply full on-chip data selection to the four output gates. A separate strobe input is provided. A 4-bit word is selected from one of two sources and is routed to the four outputs. The '157, 'L157, 'LS157, and 'S157 present true data whereas the 'LS158 and 'S158 present inverted data to minimize propagation delay time.

FUNCTION TABLE

INPUTS				OUTPUT Y	
STROBE	SELECT	A	B	'157, 'L157, 'LS157, 'S157	'LS158 'S158
H	X	X	X	L	H
L	L	L	X	L	H
L	L	H	X	H	L
L	H	X	L	L	H
L	H	X	H	H	L

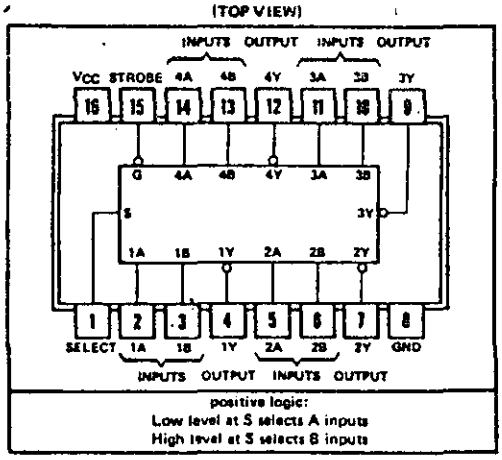
H = high level, L = low level, X = irrelevant

SN54157, SN54LS157, SN54S157 . . . J OR W PACKAGE
SN54L157 . . . J PACKAGE
SN74157, SN74L157, SN74LS157, SN74S157 . . . J OR N PACKAGE
(TOP VIEW)



positive logic:
Low level at S selects A inputs
High level at S selects B inputs

SN54LS158, SN54S158 . . . J OR W PACKAGE
SN74LS158, SN74S158 . . . J OR N PACKAGE
(TOP VIEW)



positive logic:
Low level at S selects A inputs
High level at S selects B inputs

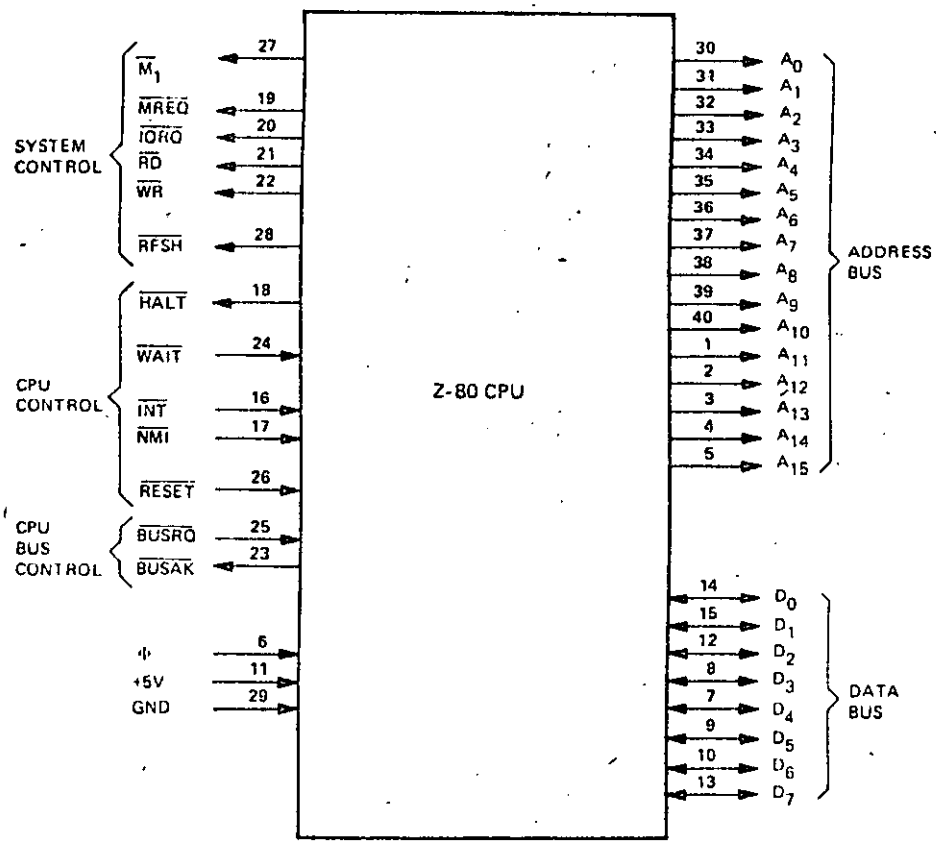
absolute maximum ratings over operating free-air temperature range (unless otherwise noted)

Supply voltage, V_{CC} (see Note 1) . . . . . 7 V
Input voltage: '157, 'L157, 'S158 . . . . . 5.5 V
'LS157, 'LS158 . . . . . 7 V
Operating free-air temperature range: SN54', SN54L', SN54LS', SN54S' Circuits . . . . . −55°C to 125°C
SN74', SN74L', SN74LS', SN74S' Circuits . . . . . 0°C to 70°C
Storage temperature range . . . . . −65°C to 150°C

NOTE 1: Voltage values are with respect to network ground terminal.

3.0 Z-80 CPU PIN DESCRIPTION

The Z-80 CPU is packaged in an industry standard 40 pin Dual In-Line Package. The I/O pins are shown in figure 3.0-1 and the function of each is described below.



Z-80 PIN CONFIGURATION
FIGURE 3.0-1

A_0-A_{15} (Address Bus)

Tri-state output, active high. A_0-A_{15} constitute a 16-bit address bus. The address bus provides the address for memory (up to 64K bytes) data exchanges and for I/O device data exchanges. I/O addressing uses the 8 lower address bits to allow the user to directly select up to 256 input or 256 output ports. A_0 is the least significant address bit. During refresh time, the lower 7 bits contain a valid refresh address.

D_0-D_7 (Data Bus)

Tri-state input/output, active high. D_0-D_7 constitute an 8-bit bidirectional data bus. The data bus is used for data exchanges with memory and I/O devices.

$\overline{M_1}$ (Machine Cycle one)

Output, active low. $\overline{M_1}$ indicates that the current machine cycle is the OP code fetch cycle of an instruction execution. Note that during execution of 2-byte op-codes, $\overline{M1}$ is generated as each op code byte is fetched. These two byte op-codes always begin with CBH, DDH, EDH or FDH. $\overline{M1}$ also occurs with $\overline{IORQ}$ to indicate an interrupt acknowledge cycle.

$\overline{MREQ}$ (Memory Request)

Tri-state output, active low. The memory request signal indicates that the address bus holds a valid address for a memory read or memory write operation.

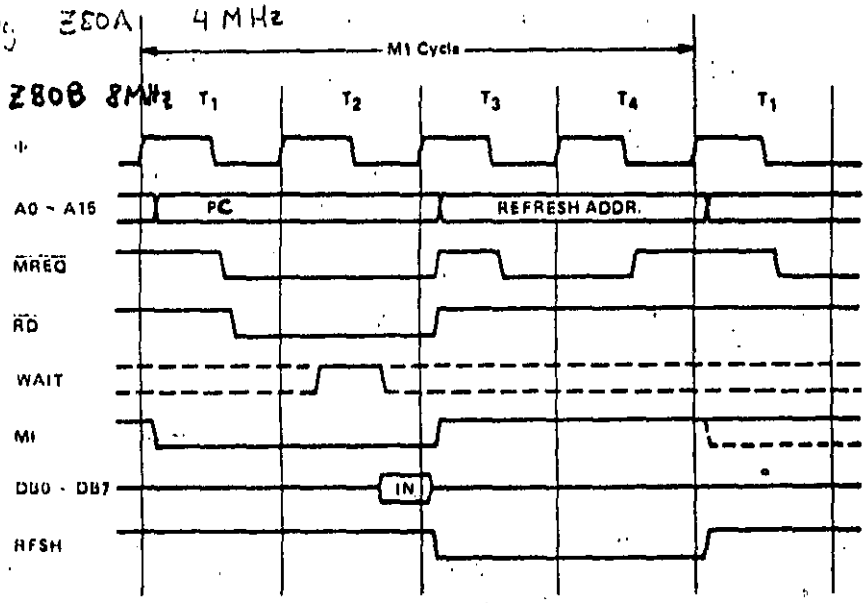
INSTRUCTION FETCH

Figure 4.0-1 shows the timing during an M1 cycle (OP code fetch). Notice that the PC is placed on the address bus at the beginning of the M1 cycle. One half clock time later the $\overline{MREQ}$ signal goes active. At this time the address to the memory has had time to stabilize so that the falling edge of $\overline{MREQ}$ can be used directly as a chip enable clock to dynamic memories. The $\overline{RD}$ line also goes active to indicate that the memory read data should be enabled onto the CPU data bus. The CPU samples the data from the memory on the data bus with the rising edge of the clock of state T3 and this same edge is used by the CPU to turn off the $\overline{RD}$ and $\overline{MRQ}$ signals. Thus the data has already been sampled by the CPU before the $\overline{RD}$ signal becomes inactive. Clock state T3 and T4 of a fetch cycle are used to refresh dynamic memories. (The CPU uses this time to decode and execute the fetched instruction so that no other operation could be performed at this time). During T3 and T4 the lower 7 bits of the address bus contain a memory refresh address and the $\overline{RFSH}$ signal becomes active to indicate that a refresh read of all dynamic memories should be accomplished. Notice that a $\overline{RD}$ signal is not generated during refresh time to prevent data from different memory segments from being gated onto the data bus. The $\overline{MREQ}$ signal during refresh time should be used to perform a refresh read of all memory elements. The refresh signal can not be used by itself since the refresh address is only guaranteed to be stable during $\overline{MREQ}$ time.

T_{MIN} = 400 nsec Z80 2.5 MHz

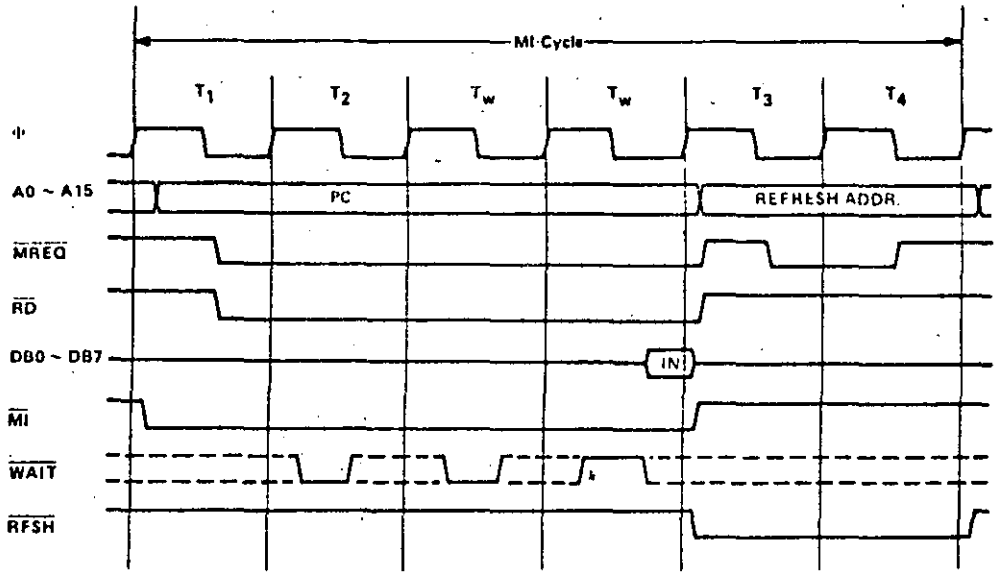
T_{MIN} = 250 nsec Z80A 4 MHz

T_{MIN} = 125 nsec Z80B 8MHz



INSTRUCTION OP CODE FETCH
FIGURE 4.0-1

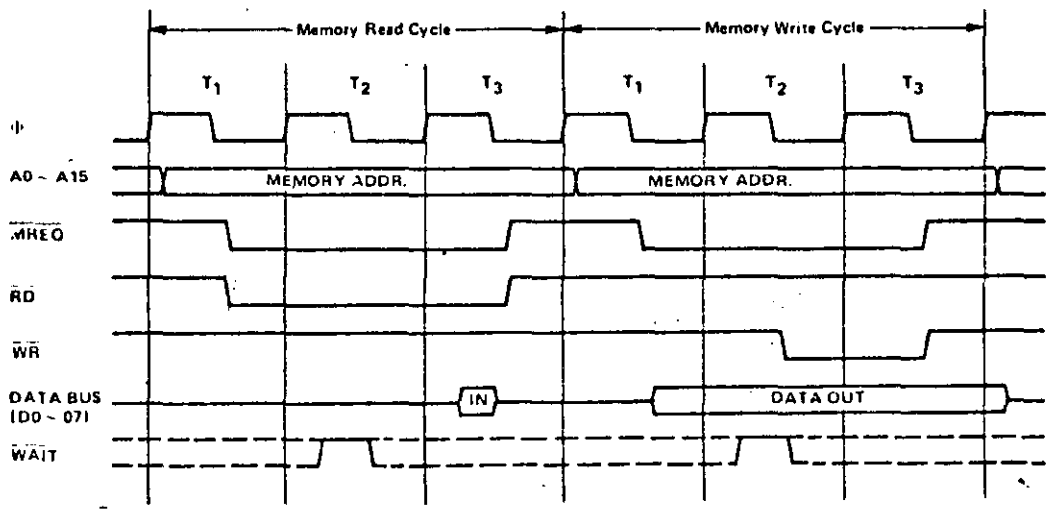
Figure 4.0-1A illustrates how the fetch cycle is delayed if the memory activates the $\overline{WAIT}$ line. During T2 and every subsequent Tw, the CPU samples the $\overline{WAIT}$ line with the falling edge of Φ. If the $\overline{WAIT}$ line is active at this time, another wait state will be entered during the following cycle. Using this technique the read cycle can be lengthened to match the access time of any type of memory device.



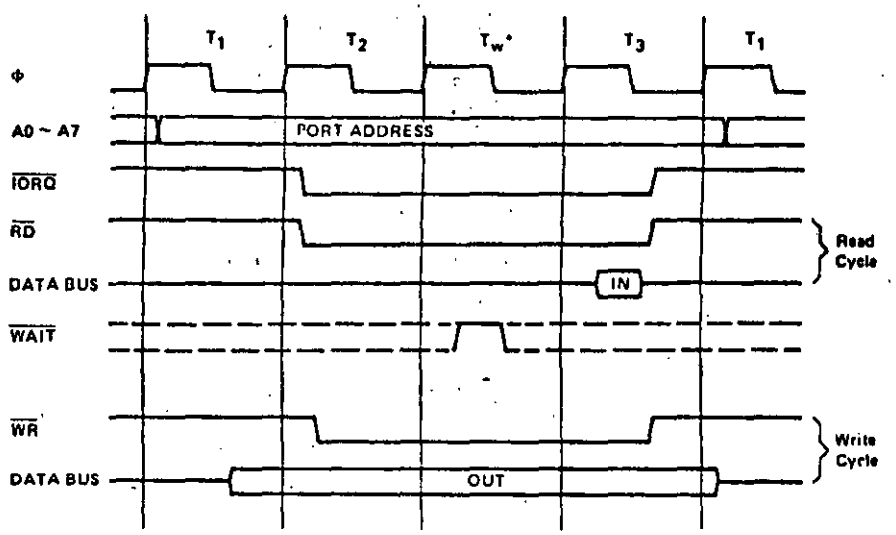
INSTRUCTION OP CODE FETCH WITH WAIT STATES
FIGURE 4.0-1A

MEMORY READ OR WRITE

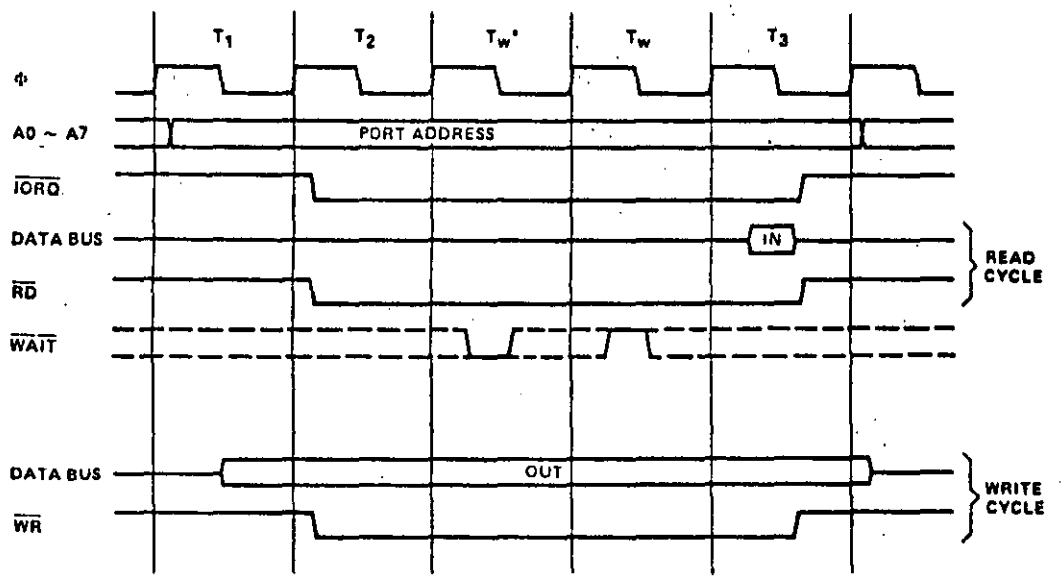
Figure 4.0-2 illustrates the timing of memory read or write cycles other than an OP code fetch (M1 cycle). These cycles are generally three clock periods long unless wait states are requested by the memory via the $\overline{WAIT}$ signal. The $\overline{MREQ}$ signal and the $\overline{RD}$ signal are used the same as in the fetch cycle. In the case of a memory write cycle, the $\overline{MREQ}$ also becomes active when the address bus is stable so that it can be used directly as a chip enable for dynamic memories. The $\overline{WR}$ line is active when data on the data bus is stable so that it can be used directly as a R/$\overline{W}$ pulse to virtually any type of semiconductor memory. Furthermore the $\overline{WR}$ signal goes inactive one half T state before the address and data bus contents are changed so that the overlap requirements for virtually any type of semiconductor memory type will be met.



MEMORY READ OR WRITE CYCLES
FIGURE 4.0-2

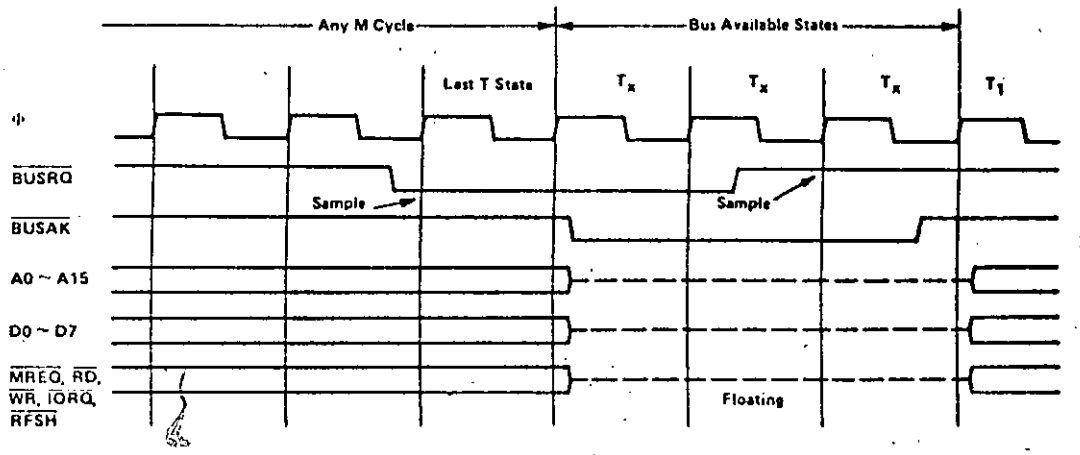


INPUT OR OUTPUT CYCLES
FIGURE 4.0-3



INPUT OR OUTPUT CYCLES WITH WAIT STATES
FIGURE 4.0-3A

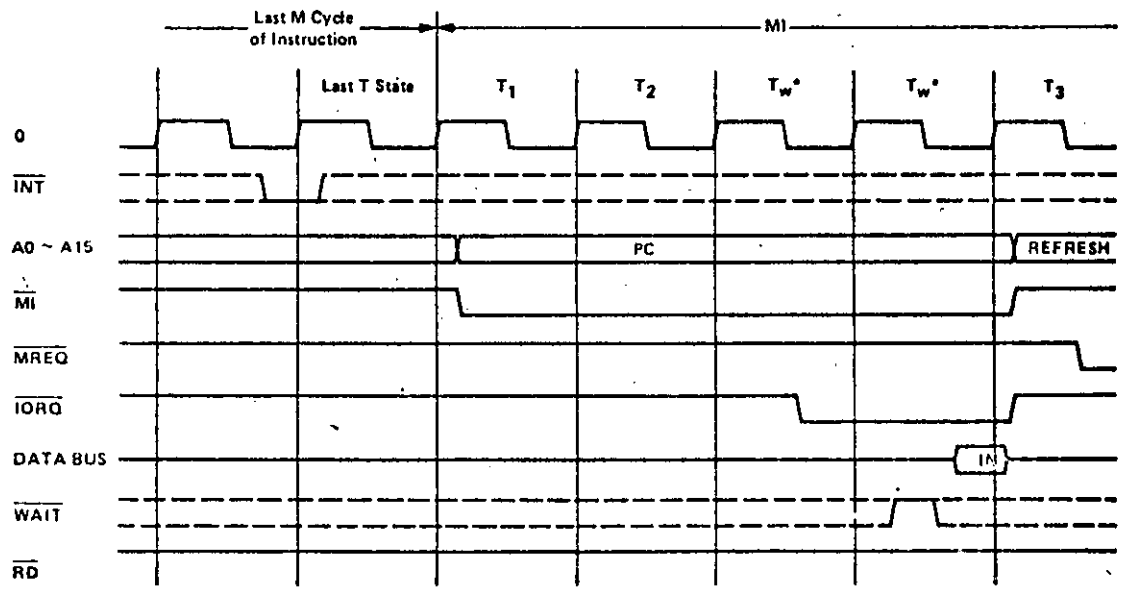
* Automatically inserted WAIT state



BUS REQUEST/ACKNOWLEDGE CYCLE
FIGURE 4.0-4

INTERRUPT REQUEST/ACKNOWLEDGE CYCLE

Figure 4.0-5 illustrates the timing associated with an interrupt cycle. The interrupt signal ($\overline{INT}$) is sampled by the CPU with the rising edge of the last clock at the end of any instruction. The signal will not be accepted if the internal CPU software controlled interrupt enable flip-flop is not set or if the $\overline{BUSRQ}$ signal is active. When the signal is accepted a special M1 cycle is generated. During this special M1 cycle the $\overline{IORQ}$ signal becomes active (instead of the normal $\overline{MREQ}$) to indicate that the interrupting device can place an 8-bit vector on the data bus. Notice that two wait states are automatically added to this cycle. These states are added so that a ripple priority interrupt scheme can be easily implemented. The two wait states allow sufficient time for the ripple signals to stabilize and identify which I/O device must insert the response vector. Refer to section 8.0 for details on how the interrupt response vector is utilized by the CPU.



INTERRUPT REQUEST/ACKNOWLEDGE CYCLE
FIGURE 4.0-5

DIVISION DE EDUCACION CONTINUA
FACULTAD DE INGENIERIA U.N.A.M.

MICROPROCESADORES Y MICROCOMPUTADORAS

ANEXO 2 ARTICULOS VARIOS

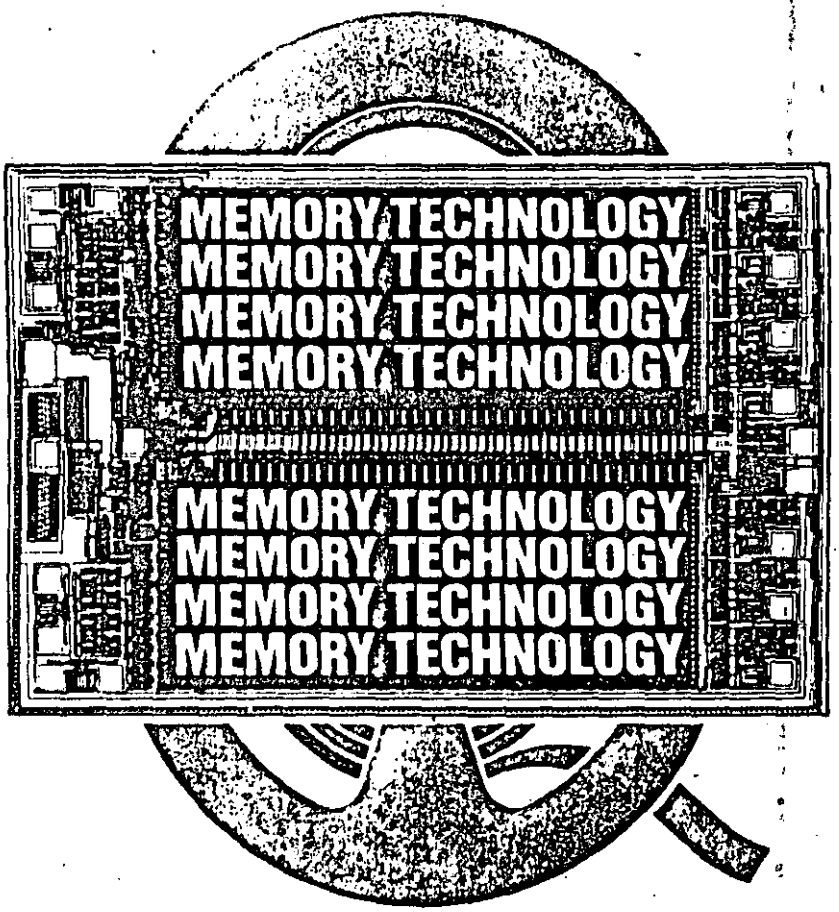
NOVIEMBRE,1984.

Palacio de Minería Calle de Tacuba 5 primer piso Deleg. Cuauhtémoc 06000 México, D.F. Tel.: 521-40-20 Apdo. Postal M-2285

Innovative chip designs lead to dense, superfast RAMs

Advanced MOS cell structures, finer lines, and improved processing bring new levels of speed and density to dynamic and static RAM chips.

Dave Bursky



The design innovations and leading-edge processing that have created today's sub-100-ns 64-kbit dynamic and static RAMs are teaming up once again, carving out even faster 256-kbit devices and holding the promise of more than 1 million bits on a single chip as early as 1985. Typically, next-generation devices will sport a fourfold increase in capacity.

The substantial research now being done in materials, including metal silicides and gallium-arsenide-based technology, point to even faster, smaller, and lower-power static RAMs than are possible today. Advanced processes and lithography also are shrinking the cells, chips, and price tags of older designs.

Advanced wafer lithography systems, such as step-and-repeat optical projectors, are contributing to several manufacturing steps of 64-kbit dynamic RAMs, making sure that the critical mask layers are accurately positioned. With 256k dynamic and 64k static RAMs, the stepping technique ensures close tolerances between mask layers and permits chip designers to minimize tolerance allowances that are usually designed into masks at the expense of chip area.

As lines inside memories get finer, designers are examining materials other than polysilicon for the second connection layer. Many different types of metal silicides have been developed, some of which are deposited atop the polysilicon connection layers to form what designers call a "polycide." Almost every 256k dynamic RAM uses either a polycide or a straight metal silicide connection layer based on metals like molybdenum, tantalum, titanium, or tungsten.

New capacitor structures in dynamic RAMs are being explored to pack more capacitance into a smaller space. Many designers feel that capacitance values cannot drop much below 35 fF without yielding an unacceptable amount of soft errors caused by alpha particles, which could nearly eliminate the total stored charge. Designers are considering alternatives, such as making a memory array with p-channel devices sitting in an n well, incorpo-

rating new dielectric materials, or integrating a more vertical capacitor structure into the substrate to gain high capacitance in a small lateral area.

For density, look to dynamic RAMs

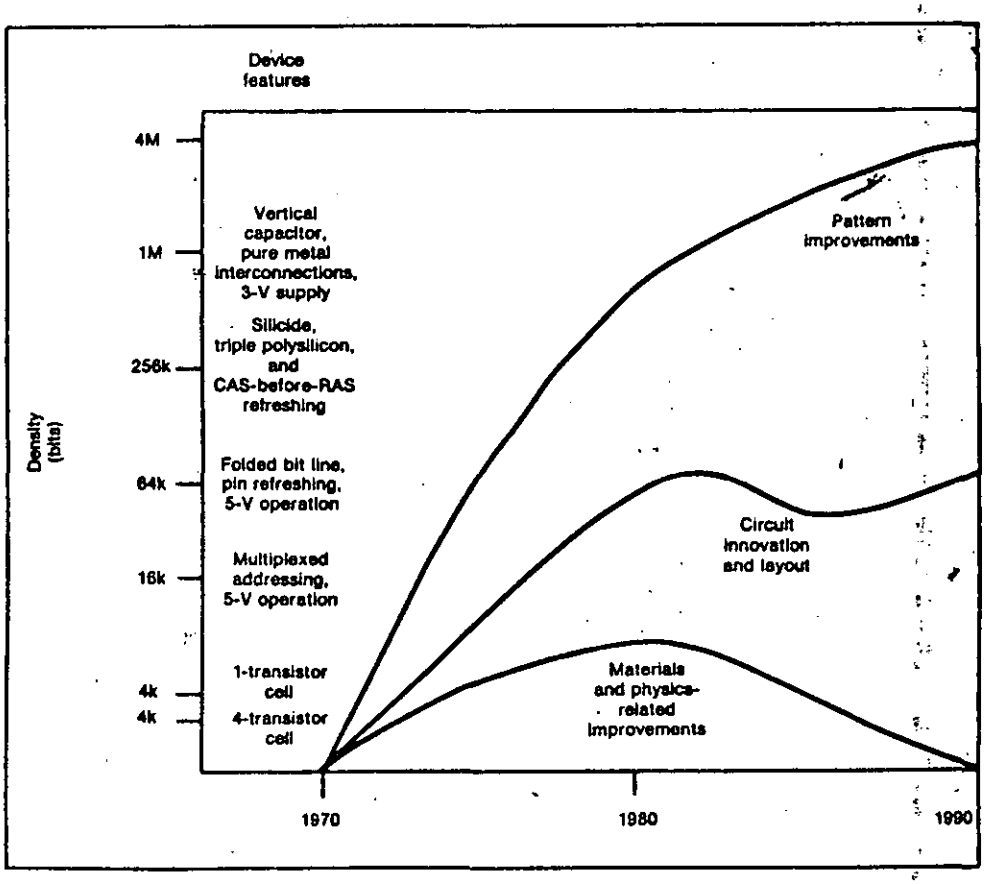
The dynamic RAM, with its capacitive storage element and single control transistor in each cell, is the densest of any volatile memory. It has come a long way in a relatively short period. Already, more than a dozen firms are producing 64-kbit devices, with 256-kbit chips now in or reaching the sampling stage. Around the design corner sits the 1-Mbit dynamic RAM, which will probably require both radical changes in circuit and capacitor structures and a drop in supply voltages from the current level of 5 V to about 3 V (Fig. 1).

Designers are pushing hard to develop practical 256k dynamic RAMs. In the United States, Advanced Micro Devices, Inmos, Intel, Micron Technology, Mostek, Motorola, National Semiconductor, and Texas Instruments all have extensive programs for commercial devices. Not to be left out, IBM and Bell Laboratories have developed 256k devices for in-house applications. IBM's chip is actually 288 kbits, organized as 32k by 9 for applications that require parity. The 256k RAM from Bell probably will be sold by Western Electric later this year or early next.

Outside the U.S. the 256-kbit activity is centered in Japan, with all the major semiconductor makers—Fujitsu, Hitachi, Mitsubishi Electric, NEC, Oki Electric, Toshiba, and Nippon Telegraph and Telephone's Musashino Electrical Communications Laboratory—designing or sampling 256-kbit RAMs.

Although most U.S. companies are reluctant to reveal the technology behind their products, Motorola Inc. (Austin, Texas) has divulged some information about its 256k chip. Designed using 2-μm rules, the component can be fabricated with either one or two polysilicon layers. The interconnections rely on a combination of polysilicon and refractory metal silicide to keep resistance as low as possible. As a result, the chip has a typical access time of 90 ns; in the nibble-mode, however, the access time is significantly faster—10 ns after the first bit.

The chip's serial nibble mode gains the high speed through a 64k-by-4 internal organization; four bits can be presented simultaneously to the output section. A 4-bit serial shift register activates some decoding logic that permits a 33-MHz data rate for reading or writing. Preliminary devices are



1. Circuit innovations and lithography are two of the most important reasons that the density of dynamic RAMs has risen dramatically over the past decade and the rest of this decade. Already 256k devices are being released as samples.

relatively large, coming in at slightly over 70,000 mil², but future versions will be scaled down to reduce costs.

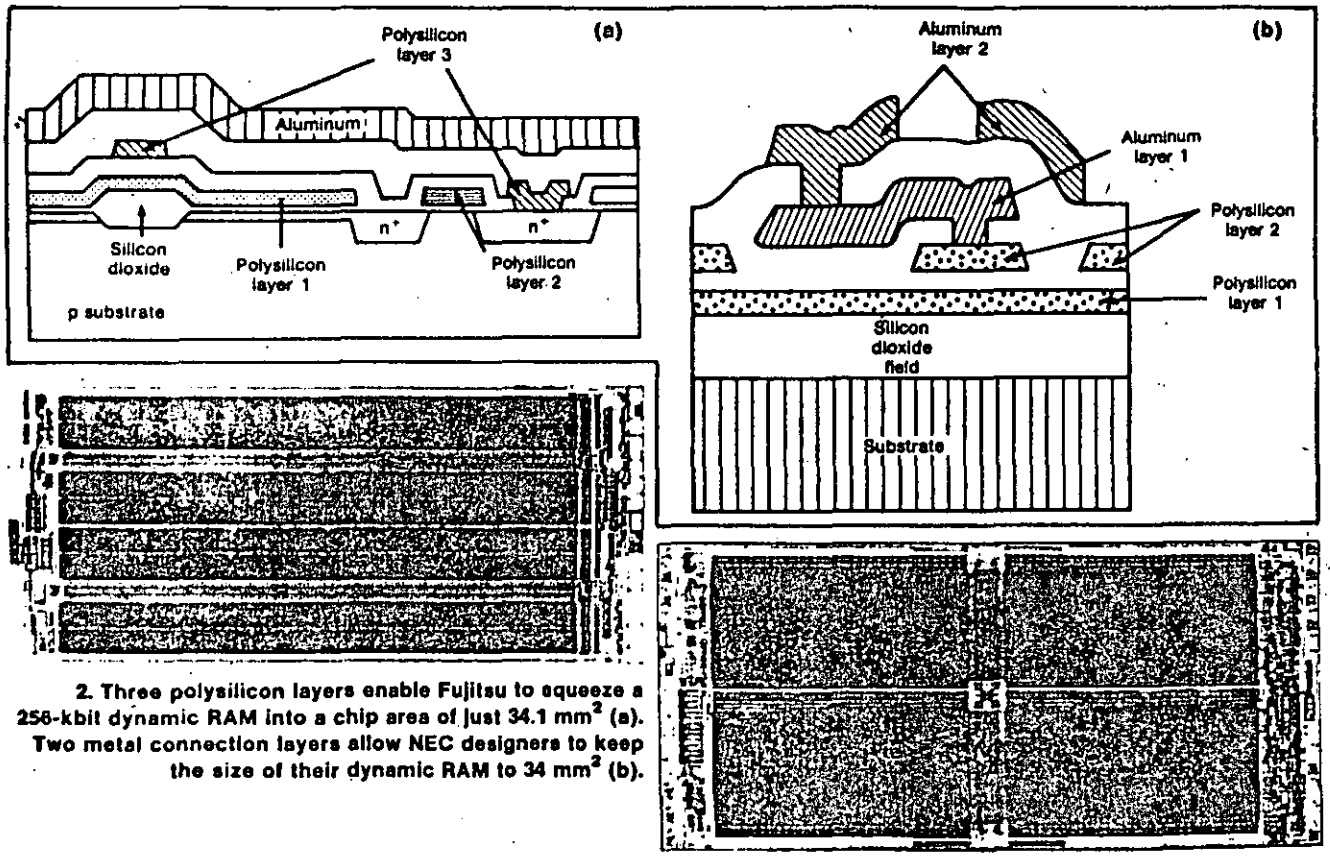
The nibble mode of a 256k dynamic RAM from Micron Technology Inc. (Boise, Idaho) has a different twist: an 8-bit nibble instead of the customary 4-bit one. Like most other 256k devices, the Micron part refreshes itself automatically using a scheme known as CAS-before-RAS (column-address strobe before row-address strobe). Samples of the chip, expected late this year, will incorporate 2.4-μm design rules and yield access times between 100 and 120 ns.

Some companies are examining CMOS technology as a way of cutting power dissipation in the periphery and possibly boosting alpha-particle immunity. Intel Corp.'s Dynamic Memory Division (Hillsboro, Ore.) has 1984 plans for a CMOS 256k dynamic chip that will perform static-column addressing but not CAS-before-RAS refreshing.

Trying to hedge its bets on which way the market will turn, Texas Instruments Inc.'s MOS Memory Group (Houston) has laid out its 256k contender so that the final interconnection mask can be adapted to a nibble or page mode, self-refreshing or CAS-before RAS refreshing, and so on. Despite that, there will definitely be a 64k-by-4 version, possibly even before a 256k-by-1. For on-chip connections, TI plans to use a polycide to keep resistance low. As in the 64k dynamic RAM, an epitaxial layer will probably be incorporated to minimize substrate noise levels. The chip will have dimensions of about 60,000 mil², although a smaller version—45,000 mil²—is in the works.

Byte-wide dynamic RAMs

Aiming at a different target—small microprocessor systems—Mostek Corp. (Carrollton, Texas) has developed a 32k-by-8 part using an advanced short-channel NMOS process, which is characterized by light doping, triple diffusions, and two layers of polysilicon and metal interconnections. Internal refresh circuitry and a standard static nonmultiplexed addressing scheme afford savings in external logic and board space. The 256k RAM,



2. Three polysilicon layers enable Fujitsu to squeeze a 256-kbit dynamic RAM into a chip area of just 34.1 mm² (a). Two metal connection layers allow NEC designers to keep the size of their dynamic RAM to 34 mm² (b).

boasting access times as short as 100 ns, operates with common data input and output lines that easily can be demultiplexed without extending the access times (see the Design article, p. 176). The company has not forsaken the bulk of the memory market and plans to introduce a 256k-by-1 chip later this year.

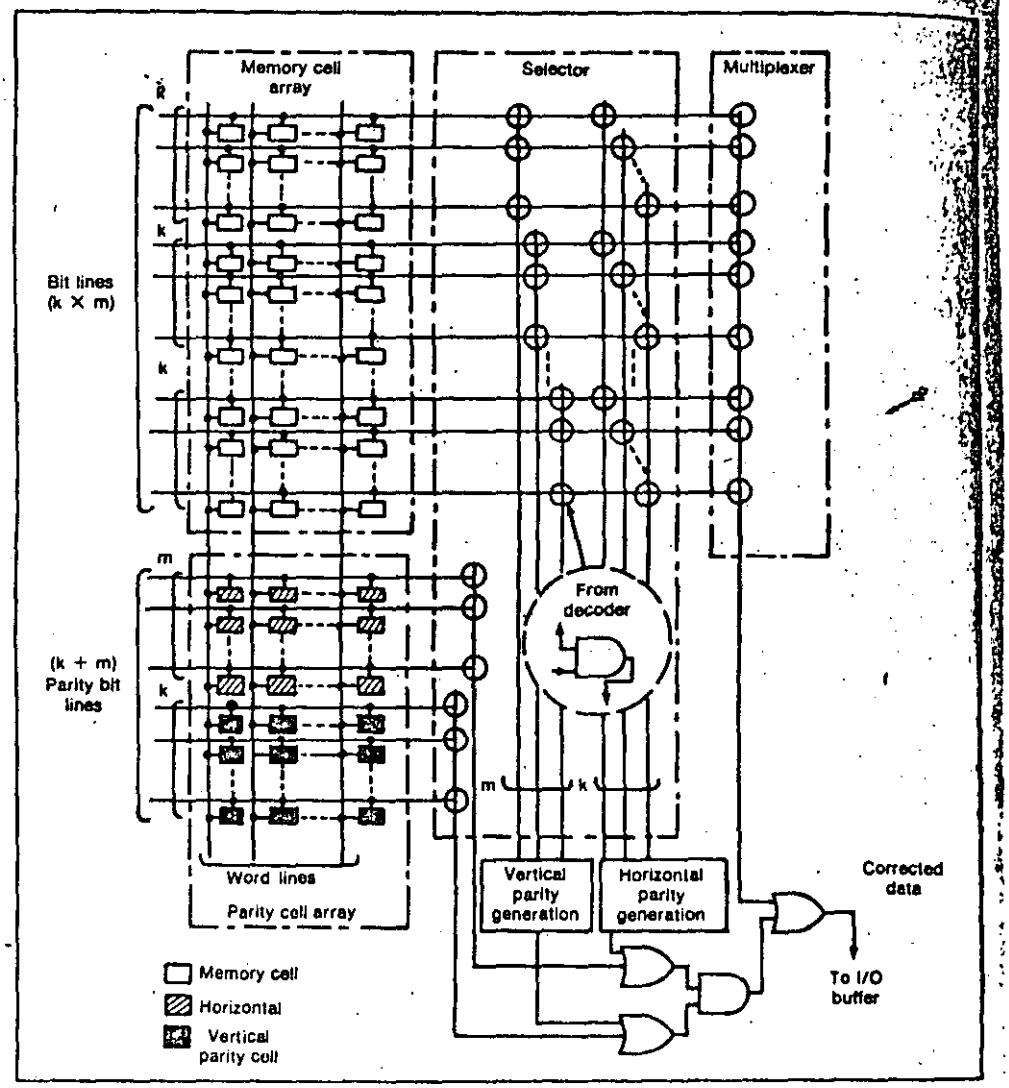
Both Advanced Micro Devices Inc. (Sunnyvale, Calif.) and National Semiconductor Corp. (Santa Clara, Calif.) have committed themselves to 100-ns, 256k dynamic RAMs. AMD is concentrating its efforts on multilevel interconnections and on a low-power technology that will be useful for this memory, as well as for a 1-Mbit chip that should surface in the second half of the decade.

Designers at National are in the process of finalizing a new triple polysilicon technology that should eliminate the potential step-coverage problems and possible layer-to-layer shorts in the firm's previous triple-polysilicon processes. The resulting 256k chip will offer CAS-before-RAS refreshing and a 4-bit nibble mode and will be laid out so that a 64k-by-4 version can be easily spun off.

A look at Japanese RAMs

When it comes to RAMs, the Japanese have made noticeable gains. Already a limited number of 256k dynamic samples have been released by Hitachi, Mitsubishi, and Toshiba, and others are expected from Fujitsu, NEC, and Oki Electric by the end of the year. Although it does not manufacture the RAM itself, NTT's Musashino lab has designed a 256k chip, which it licenses to other Japanese companies.

At last year's International Solid-State Circuits Conference, Hitachi Ltd. (Kokobunji) slipped out the first details of its 256k-RAM—a 46.8 mm² chip sporting a polycide word line and a folded-



3. Incorporating for the first time error checking and correction circuitry on a 256k memory, Nippon Telegraph and Telephone's dynamic RAM is the first that can correct a single bit error per word line.

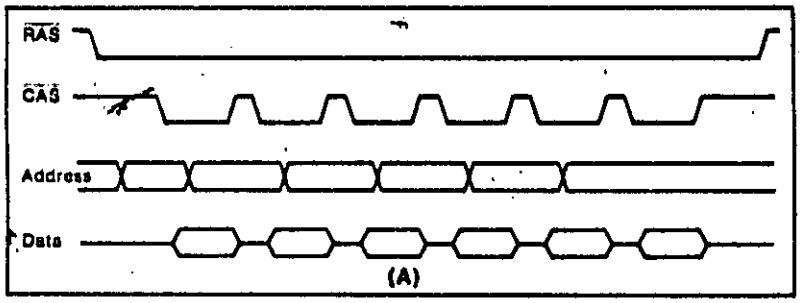
Addressing modes in brief

When choosing dynamic RAM, system designers must make a wealth of decisions, not the least of which is the addressing mode. Most designers opt, however, for either the paging or the nibbling mode. These techniques allow the chip to access stored data more rapidly, since neither one needs both a row- and a column-address strobe (RAS and CAS) to reach the data. Instead, the row address is locked in first, after which strobes enter the column addresses.

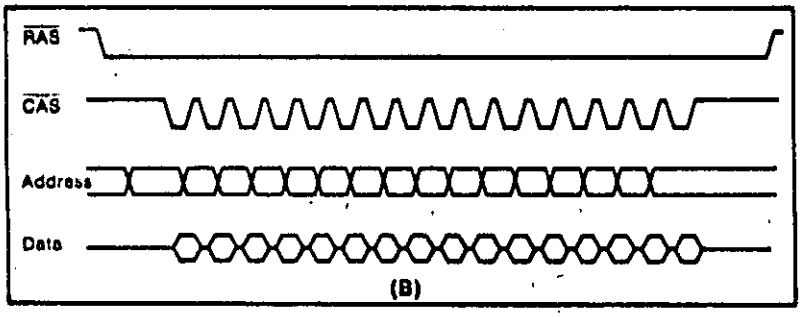
In the paging mode, new column addresses must be supplied for each bit (Fig. A). The RAM can read only part of a page before it must be refreshed. To overcome that limitation, some memories implement an extended paging mode through which they can access 256 bits by strobing in new column addresses. Intel has developed an even faster extended version, called Ripplemode, that should cut access time threefold over standard read/write times (Fig. B)

In a nibbling-mode dynamic RAM, the three bits that follow the first accessed bit can be read out by keeping the RAS line low and pulsing just the CAS signal low three times (Fig. C). Extended nibbling—in which eight bits can be obtained by pulsing the CAS line seven times—has been developed to improve memory bandwidth.

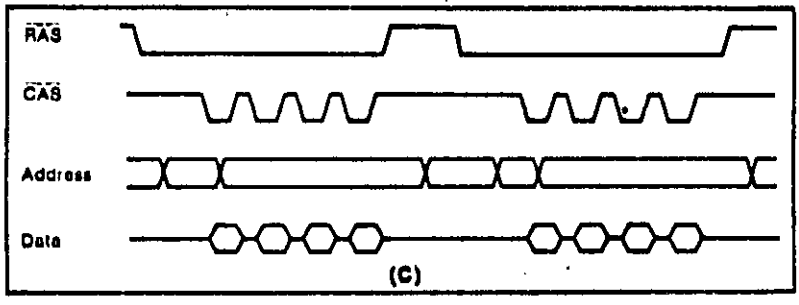
Newer addressing techniques described by both Intel and Fujitsu take advantage of static-column decoding. Prototype configurations have already demonstrated access times of about 35 ns. In the static-column technique, the RAS locks in the row address, and column addresses are read directly from the address bus (Fig. D). As addresses change, the new locations are accessed, with the data appearing on the data output pin. The unused CAS pin does not go to waste: it has been converted into a chip-enable control line.



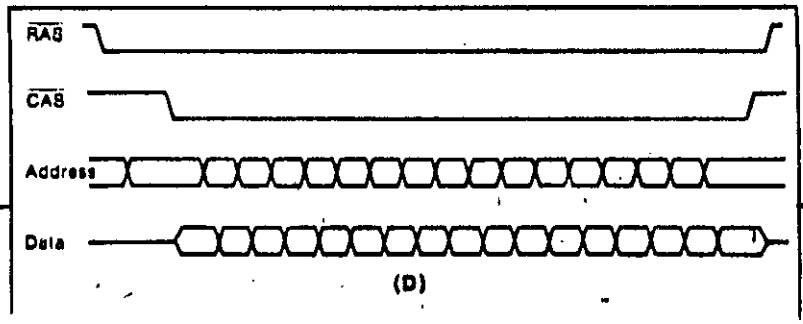
Normal cycle time = 260 ns
Page-mode cycle time = 125 ns
Average cycle time = 125.5 ns



Normal cycle time = 160 ns
Ripple-mode cycle time = 55 ns
Average cycle time = 55.4 ns



Normal cycle time = 160 ns
Nibble-mode cycle time = 55 ns
Average cycle time = 61.2 ns



Normal cycle time = 160 ns
Static-column cycle time = 55 ns
Average cycle time = 55.4 ns

aluminum bit line. Using 2-μm design rules, the chip accesses in about 120 ns; the version slated for production will be scaled down to under 40 mm^2.

Trying to hit two market segments, Toshiba Ltd. (Kawasaki) has a 256k dynamic RAM with either page or nibble capabilities. The page model, soon to be available in sample quantities, accesses in 150 or 120 ns, consumes about 330 mW maximum, and offers RAS-only and hidden refreshing. Toshiba's laboratory is already hinting at things to come, specifically a 256k chip having a CAS access time of 34 ns. (Compared with the usually specified industry-standard RAS access times, however, the chip accesses in just 94 ns.) The newer memory, which was detailed at this past February's ISSCC, incorporates molybdenum silicide gate structures and interconnections, which reduce RC delays on the word lines. A die size of 5 by 9.2 mm makes the chip one of the larger RAMs that was described at the conference.

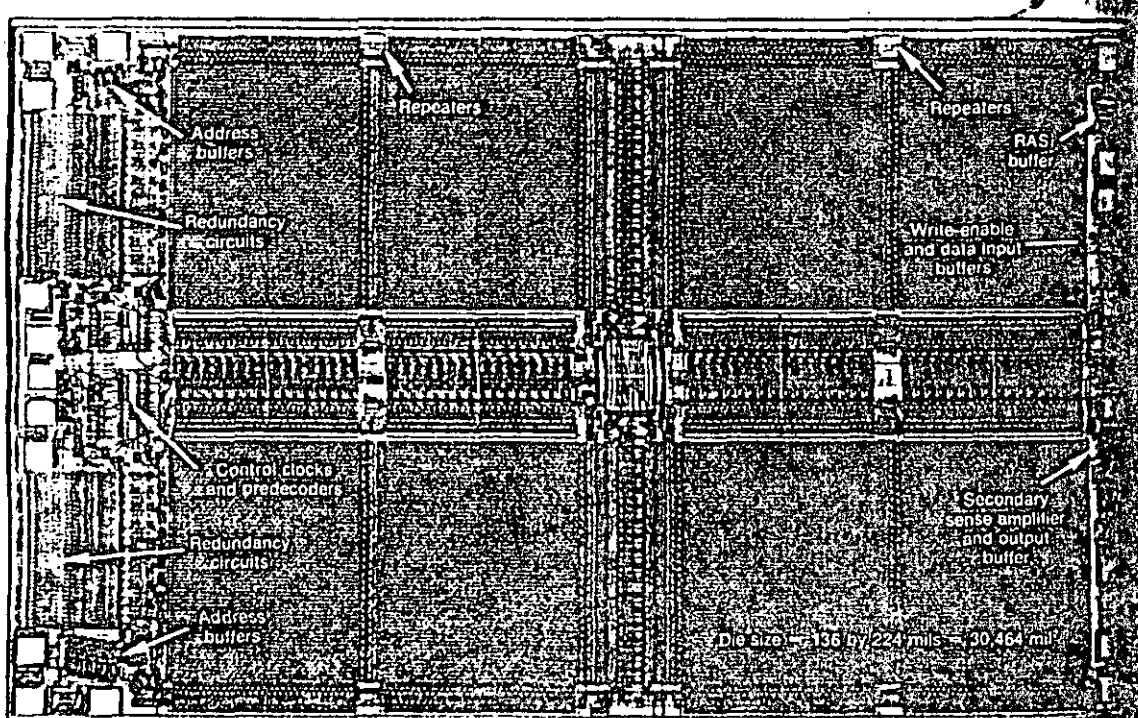
Rivaling the size of the Toshiba chip is a dynamic RAM from Mitsubishi Electric Co. Ltd. (Itami City) which checks in at 4.85 by 9.8 mm. The chip achieves a 100-ns access time and operates in both the page and nibble modes (many claim the two modes are so incompatible that they cannot be put on the same chip). Built using 2-μm design rules, the RAM's word lines, composed of molybdenum-silicide and polysilicon, and the folded-aluminum bit lines hold down RC delays. In addition, the chip employs RAS-only, CAS-before-RAS, and hidden refreshing. A proprietary high-capacitance memory cell and extremely sensitive sense amplifiers afford wide operating margins.

In early 1984 Mitsubishi will release a scaled-down version of the memory with access times ranging from 100 to 150 ns and a size of 3.78 by 8.7 mm. The company is also keeping an eye on the 64k-by-4 arena with plans to introduce a part later next year.

Moving in with the smallest 256k dynamic RAMs are Fujitsu Ltd. (Kawasaki) and NEC Corp. (Kanagawa), which claim sub-100-ns access times for their chips. The similarities end there, however. The Fujitsu chip's triple-polysilicon process and 2.5-μm gate lengths squeeze everything into 34.1 mm^2 (Fig. 2a). Specifying a nibble mode with a 15-ns/bit output, the dynamic RAM also goes with CAS-before-RAS refreshing. In contrast, NEC designers went with a two-level aluminum design to reach a 34-mm^2 chip area and with 1.3-μm design rules and oxide thicknesses of 160 Å (Fig. 2b). The chip, which accesses in 90 ns, draws about 250 mW and does not have extra features, like CAS-before-RAS refreshing.

1-Mbit dynamic memories in the works

To stretch to 1 Mbit and beyond, both Hitachi and NTT's Musashino laboratory have come up with an almost identical capacitance concept, called the corrugated capacitor cell (CCC) by Hitachi and the trench capacitor by NTT. Instead of forming a bit



4. Combining CMOS technology and static-column addressing, Intel's 64k dynamic RAM occupies about 30,000 mils^2—smaller than most production NMOS parts.

eral capacitor in each structure using polysilicon and metal layers above the transistors, both companies etch tiny pits, or trenches, into the silicon. NTT fills the pits with polysilicon; Hitachi, with sandwiches of polysilicon, silicon nitride, and polysilicon.

NTT's trench capacitor makes the memory's overall cell size small—it requires a surface area of only 1 by 2 μm. (The downwared dimension is 1.5 μm, compared with about 3 μm in the Hitachi design.) A 256k dynamic RAM described by NTT at February's ISSCC not only takes advantage of the trench structure, but also incorporates on-chip error checking and correction (ECC), CMOS support logic, and a supply voltage converter that changes 5 V into 3 V. The chip is the first to contain ECC circuitry. NTT's new ECC technique, called bidirectional parity checking, checks the parity in the X and Y directions of the memory matrix and can correct single-bit-per-word-line soft errors (Fig. 3). Built with molybdenum word lines, the dynamic RAM features CMOS circuits on the periphery that cut power drain to 200 mW when active and 3 mW on standby.

Hitachi's approach, described at last December's International Electron Devices meeting in Washington, D.C., places corrugated capacitor cells in a straightforward 64-kbit memory array. The company is now at work on a 256k memory, details of which will probably be available later this year.

Both Hitachi and NTT believe that their capacitor approaches could make the 1-Mbit level viable, but much more research and process development

When monolithic density is not enough

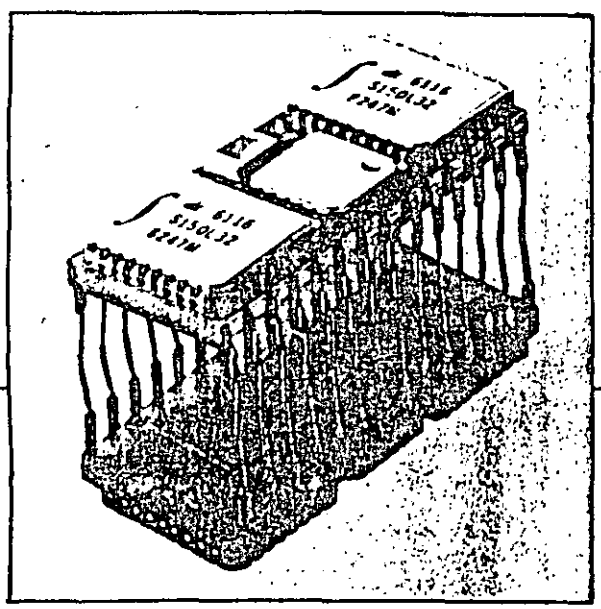
Even with today's extremely high IC densities, many system designers want more. As a result, hybrid arrangements—leadless carriers on mother DIPs or single in-line package (SIP) substrates—are gaining more popularity. Harris Corp.'s Semiconductor Group has for several years offered a plug-in CMOS static RAM that comprises 16 4-bit static chips. Recently the company applied the same modular concept to a 256-kbit hybrid circuit using 16k static RAMs and plans to upgrade that even further to make a 1-Mbit module when its 64k static RAM is available.

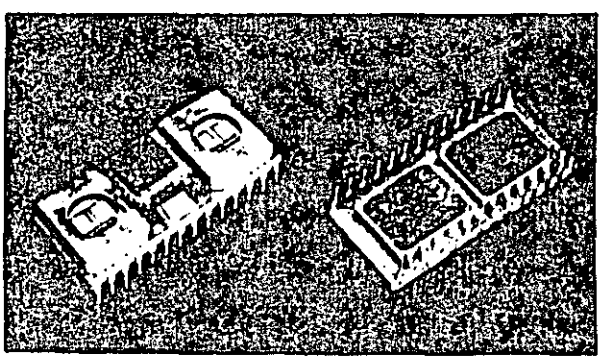
A similar but smaller arrangement comes from Integrated Device Technology, which places four 16k RAM chips on the substrate of a tiny mother DIP. The result is a 64k superfast memory for designers who need greater density and are willing to pay the price (Fig. A).

Dynamic memories are getting their turn, too. About five years ago Mostek began offering a 32k memory that consisted of two 64k chips in leadless carriers, which were mounted on an 18-pin DIP. Today the same arrangement can be used to make 128k modules with 64k RAMs; and with SIP technology used instead, 156k and larger packages can be readily produced.

Semiconductor manufacturers are not the only companies getting into the value-added business. For example, Electronic Designs Inc. (Hopkinton, Mass.) built a combination RAM and EPROM on a single mother DIP (Fig. B). The configuration includes 8 kbytes of EPROM and either 4 or 6 kbytes of static.

Hybrid packaging is already well accepted by the military, whose systems place weight and space at a premium. In the commercial world, low-cost plastic leaded carriers and automatic-handling equipment will make hybrids increasingly viable for space-limited systems.





will be needed. Additionally, Hitachi designers think that 1-Mbit memories will require pure refractory-metal connections—tungsten or molybdenum, for instance—to keep RC delays minimal when 1- to 1.5-μm design rules are used. They consider a 5-V external supply to be sufficient for the 1-Mbit part; however, on-chip converters will lower the levels to about 3 V to power the scaled-down circuits.

At the 1-Mbit level, some designers foresee three basic organizations: 1M by 1, 256k by 4, or possibly 128k by 8. As in today's market, the designers predict that the 1-bit-wide chip will dominate the orders, but for displays and desktop computers, the other organizations will probably offer a much better fit.

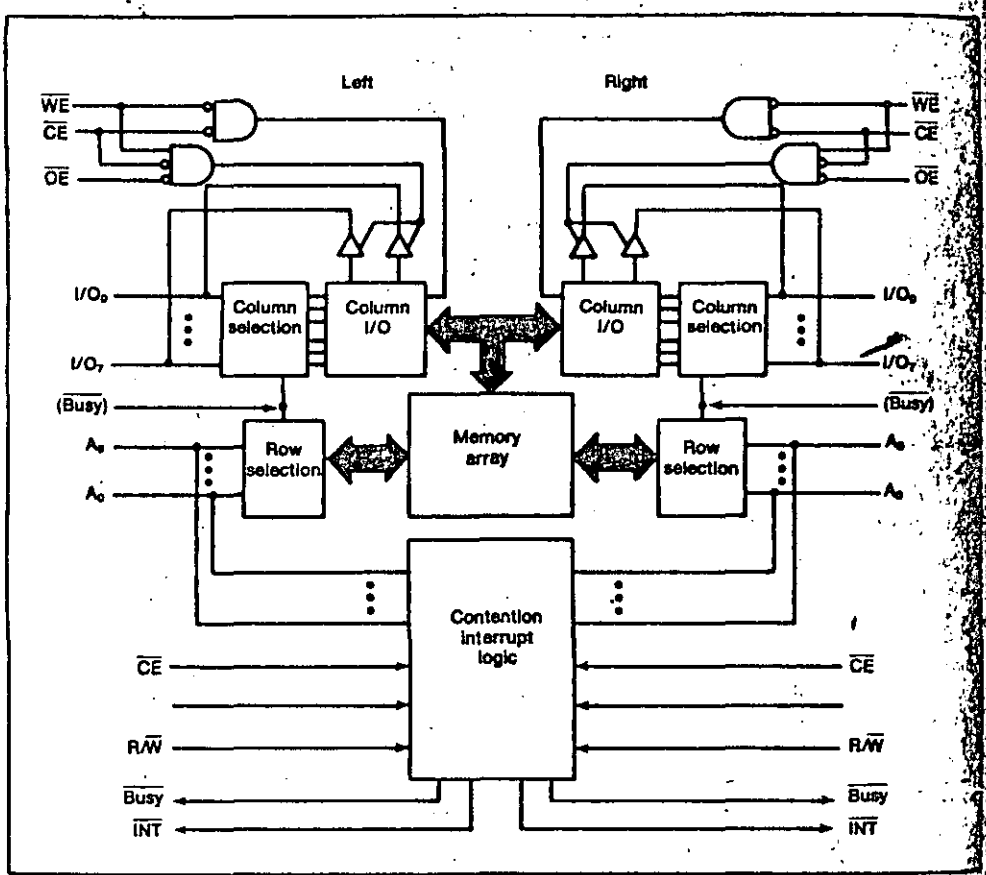
The 64k chips also reap benefits

Today's 64-kbit chips use design rules ranging between 2.5 and 3 μm and gate-oxide thicknesses of about 250 Å. Capacitances are about 55 to 80 fF for the storage cells, owing to the use of even thinner oxides (200 Å). Many advances at the 256k level are being applied to 64k memories to make the devices less expensive and better performers.

At February's ISSCC, Intel described two experimental dynamic RAMs. One, a 64k NMOS chip, uses a 150-Å capacitor dielectric and a special double-field oxidation process to reduce oxide encroachment around the gate region. The other, a CMOS static-column device, is fabricated with an n-well process, two polysilicon interconnection layers, and 150-Å capacitor dielectrics.

Special implants in the NMOS memory optimize device thresholds, isolation, and storage capacitance to obtain a speedy access time of 80 ns and an extremely high resistance to alpha particles. Occupying a chip area of less than 24,000 mil², the memory is one of the smallest 64k dynamic RAMs.

Intel's CMOS memory embeds p-channel devices in a large n-type well that is biased at the supply voltage level (Fig. 4). At 30,000 mil², the dynamic

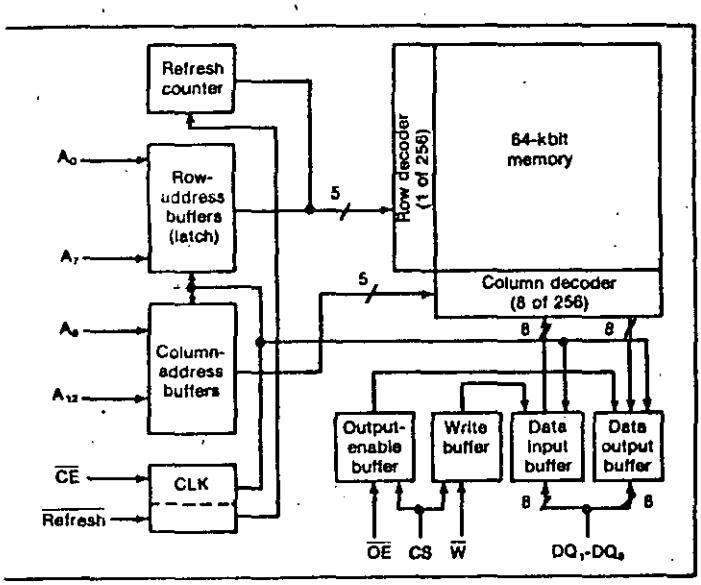


5. A static RAM from Synertek holds 1024 8-bit words in its especially designed dual-ported memory cells. The access time is a relatively swift 100 ns.

RAM accesses in 70 ns, thanks to 1.2-μm electrical channel lengths and oxide thicknesses of 250 A. Static-column addressing allows data to be accessed at 40 ns/bit. The part consumes just 5 μA on standby with a 3-V supply voltage. Samples of the commercial version are expected late this year.

As if reading Intel's mind, Fujitsu has also developed a commercial static-column dynamic RAM, but using NMOS technology. Organized as 64k by 1, the RAM accesses a column address in 55 ns, but has a high power drain of 440 mW when active. Micron Technology's 64k chip is smaller than Intel's experimental NMOS device, coming in at 22,000 mil^2. The RAM is not as fast as the Intel chip, but its 120-ns access time is nevertheless the fastest data-sheet specification around. The part consumes about 75 mW on the average, compared with 150 to 250 mW for most 64k dynamic RAMs.

Designers at Mitsubishi Electric have opted for two interconnecting layers of aluminum in their 64k entry instead of polysilicon, thereby avoiding the increasing word- and bit-line resistances normally found with polysilicon. Dual-layer aluminum, however, is more difficult to pattern, causing company designers to question whether the part can be feasibly produced.



6. A belief that byte-wide dynamic RAMs are a good solution for small systems has guided Inmos to develop an 8k-by-8 chip that contains a refresh counter but needs an external signal for each refresh cycle.

While IC designers are arguing about advances in internal circuits and processes, system designers are starting to incorporate the new features and architectures made possible by those advances. At the 64k level, Texas Instruments offers the first alternative to the standard 1-bit-wide memory—its 16k-by-4 device, the TMS 44160. The chip uses the same process flow as the company's 64k-by-1 RAM and thus follows exactly the same improvement path. TI is already selling a 64k-by-1 model that accesses in 120 ns; a 120-ns version of the 4-bit-wide RAM should be available by the end of the year. Although the design of a 4-bit-wide dynamic RAM was initially considered a maverick in the industry, other companies have sensed the potential and are rushing to become alternative sources for the TI chip. Both Fujitsu and Inmos have pin-compatible versions available, and Mitsubishi is in the process of developing one.

Alternative architectures

TI has gathered the experience it gained in the video display market and is applying it to another dynamic memory with a unique architecture. Called a multiple port memory, the chip (the TMS 4161) offers two independent ports, as well as an on-board 256-bit shift register into which an entire row of data can be loaded and then independently shifted out (see the cover article, p. 160).

Although not dynamic and thus not as dense, a dual-ported RAM from Synterek Inc. (Santa Clara, Calif.) holds 1024 8-bit words and has an access time of 100 ns (Fig. 5). It does not, however, have an on-chip shift register. Instead, the 5Y2130 offers completely static operation through the use of a true dual-ported cell and operates fully asynchronously with either port. Nearly twice as dense as a 1k-by-8 static RAM, the memory requires double the number of drivers, buffers, and address decoders, as well as extra circuitry, for each cell.

Byte-wide pseudostatic and pseudodynamic RAMs, which were introduced about four years ago as alternatives to 8-bit-wide static RAMs, are starting to pique design interest again as densities rise to 64 kbits. Now that the timing contention problems and testing difficulties inherent in a self-refreshing dynamic RAM have been largely overcome and performance has risen, several vendors are again testing the waters.

Intel, for one, has what it calls an intelligent RAM—iRAM. The 8k-by-8 dynamic part has on-chip refreshing and arbitration control circuitry. NEC has developed a pseudostatic device, and Inmos offers an 8k-by-8 dynamic memory that contains a refresh counter but no timing circuitry, unlike the other two. The Inmos part appears more

like a dynamic memory, since it requires an input pulse to initiate a refresh cycle (Fig. 6). On the other hand, when an external system does not supply any timing information, the Intel and NEC devices generate their own refresh signals, thus making them resemble static memories.

The density of static memories is benefiting from the same processing advances applied to dynamic memories. Already several CMOS 8k-by-8 static RAMs are available as samples, and designers will be able to choose from a half dozen models from close to a dozen companies by next year.

Static memories gain speed

The feasibility of a 64k-by-1 NMOS static RAM has garnered attention at Fujitsu. Described at this year's ISSCC, the chip accesses in just 40 ns; but because of its NMOS design, it dissipates about 425 mW when active and 100 mW on standby. Fabricated with 1.5-μm design rules and two polysilicon interconnection layers, the RAM occupies an area of just over 50,000 mil^2.

Although most 64k CMOS static RAMs outwardly appear the same, some major internal distinctions could make all the difference in a particular application. One of the most notable internal differences is the use of a four-transistor cell with polysilicon load resistors or a six-transistor cell with depletion loads. The former structure builds a smaller chip at a lower cost; but standby current—typically tens of microamperes—is penalized because of the polysilicon loads. By comparison, the six-transistor cell occupies a larger area and thus creates a larger chip. However, standby current is typically just 1 or 2 μA, almost eliminating any battery drain.

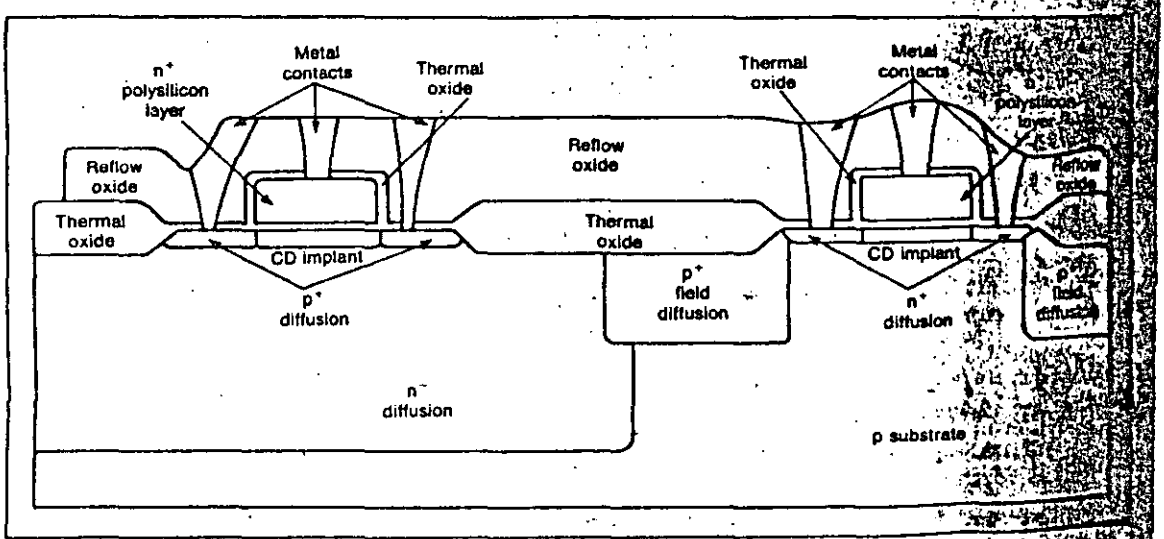
Many companies are lining up both types of cell structures for different application needs. Toshiba's four-transistor series consumes 100 μA maximum on standby, whereas its six-transistor series has a maximum standby current of 1 μA at 60°C (0.2 μA at 25°C). Mitsubishi, Fujitsu, Hitachi, and Oki Electric all are developing or supplying samples of four-transistor cell memories and expect to have six-transistor models available by late this year or sometime early next year.

Overcoming the generation gap

On many first-generation CMOS RAMs, activity on the address and data lines, even though the memory is deselected, cause standby power to rise almost to NMOS power levels. To avoid that problem, some new designs incorporate special lock-out circuitry on the periphery to make sure that general bus activity does not increase power dissipation.

NEC, for one, has built a CMOS RAM that locks out activity on the address and data lines. Designed specifically for battery-backup applications, the 6.2-by-7.3-mm chip employs two layers of aluminum to reach an access time of 80 ns. In addition, whenever the supply voltage drops to 3.5 V the memory automatically enters a low-power data-retention mode.

Designers at Mostek have conceived an interesting alternative: a specially designed chip package that holds two tiny lithium batteries. Called the Zeropower RAM, the 16k chip uses special analog circuitry to monitor the normal external supply voltage: when the latter drops below 4.5 V, internal batteries automatically take over and the write-protection feature switches on, preventing inadvertent data loss caused by power-up or power-down transients. The memory is available with a 2k-by-8



7. By scaling down its older CMOS technology, Harris Semiconductor has been able to use the new SAJI process to build 55-ns 2k-by-8 static RAMs and next year, an 8k-by-k chip. The SAJI process employs self-aligned junction isolation.

organization that can easily be upgraded when 64k chips come out next year.

Most commercial activity in byte-wide CMOS and NMOS RAMs focuses on the 16-kbit level. Many of the chips now in volume production are selling at extremely low prices; however, an exception to the low prices are the superfast static memories. Thus far, only a few companies—Inmos, Integrated Device Technology, Hitachi, Matra-Harris, and Motorola—have thrown their hats in the fast static ring (4- and 1-bit-wide memories). (Intel entered the 16k-by-1 arena early but withdrew its product because of design problems; a comeback is planned for 1984.) Inmos has already built 4k-by-4 and 16k-by-1 CMOS chips, and other companies are competing with high-speed 4k-by-1 and 1k-by-4 devices.

Harris Semiconductor has just begun: Its first entry, a 2k-by-8-bit asynchronous CMOS RAM, has a 55-ns access time. Built with a self-aligned junction-isolation process called SAJI V (Fig. 7), the chip accesses in just 70 ns over the full military temperature range (−55° to +125°C). The SAJI process will be enhanced for the company's forthcoming 64k RAM, which will consume 100 μA (worst case) over the same temperature range.

Bipolar leads the race

As density and speed rise, bipolar technology moves into the RAM picture. Already some 16k-by-1 emitter-coupled logic RAMs having access times of about 15 ns are being offered as samples by Fujitsu, Hitachi, and Fairchild Camera and Instrument Corp. (Mountain View, Calif.).

Making the 16k bipolar chips possible are advanced cell designs that use pnp loads. Along with a vertical isolation structure, these loads slash the size of the memory cell by as much as 50% compared with the cells used in earlier 4k memories. Of course, achievements at the 16k level can be passed to the 4-kbit level for even greater speed. To prove that, NEC has developed a 1k-by-4 ECL RAM that accesses data in just 4.5 ns, and Fujitsu designers have created a variable-architecture 4k RAM that accesses in 3.5 ns. The NEC memory does not use pnp loads; instead, designers joined a Schottky diode with a resistor.

Going one notch beyond bipolar speeds, gallium arsenide has been able to yield some 1024-bit static RAMs. Experimental chips from the Fujitsu and NTT's Musashino laboratory have yielded access times of a mere 2 ns. In addition to speed, a major advantage of GaAs materials is low power dissipation: The RAMs draw an active current of less than 100 mW, only about one-tenth the power of bipolar RAMs.□

A 256-kbit dynamic RAM with internal refresh logic and nonmultiplexed addressing dramatically reduces external circuitry and saves board space.

32k-by-8 DRAM cuts size, cost of μP systems

Small microprocessor systems generally need about 2 Mbits of RAM, which designers often obtain by using 256k chips arranged as 1 or 4 bits wide. However, to organize the data into the required 8-bit format necessitates extra circuitry.

A 256-kword dynamic RAM organized as 32k by 8 bits may well be the answer to a small system designer's prayers. Besides storing the greatest amount of memory on one chip, the RAM eliminates the hassles of configuring standard 8-bit words from memories 1 or 4 bits wide. Boasting internal refresh logic and nonmultiplexed addressing, the memory not only minimizes the amount of external circuitry, but also conserves board space in and cuts the cost of small microprocessor-based systems. In short, the MK4856 dynamic RAM is a providential alternative for systems that have no need for the superfast cycle times of static memories or that are sensitive to the cost per bit of smaller volatile memories.

The 4856 dynamic RAM reaches access times as short as 100 ns thanks to a new lightly doped NMOS process, called LD^3, that uses triple diffusion and double polysilicon layers (see "The Making of a New Memory"). In many smaller systems, one chip suffices for all memory requirements, especially when a limited amount of user-supplied data must be stored temporarily, as in industrial control or electronic measurement systems. Jobs requiring more memory, however, can gain other benefits from the chip's organization: Just as the RAM's 32k-by-8 structure results in a smaller size memory than that dictated by byte-wide groups using 256k-by-1 components, so, too, can the memory be expanded in smaller increments. With the 4856, a system designer can add memory in 32-kword increments to match a design application more closely and thereby reduce system cost.

The dynamic RAM is housed in a standard 600-mil, 28-pin package that has a pinout compatible with a variety of byte-oriented devices, including the industry-standard 32k-by-8 ROM and other static ROMs, RAMs, and EPROMs. For instance, the pinouts of the 4856 and 32k-by-8 ROM differ only in the Write Enable signal, which the ROM obviously does not include. However, a designer could build a memory board for the 4856 that accommodates any combination of nonvolatile and volatile memories.

Despite the fact that the 4856 chip is designed for small systems, it sacrifices nothing in a high-performance system. It operates with chip-enable access times of 100, 120, or 150 ns to handle system clock rates beyond 12.5 MHz. These fast access times eliminate memory wait states and ensure more efficient performance. Furthermore, the chip's nonmultiplexed addressing scheme eliminates the problems inherent in the multiplexing of addresses

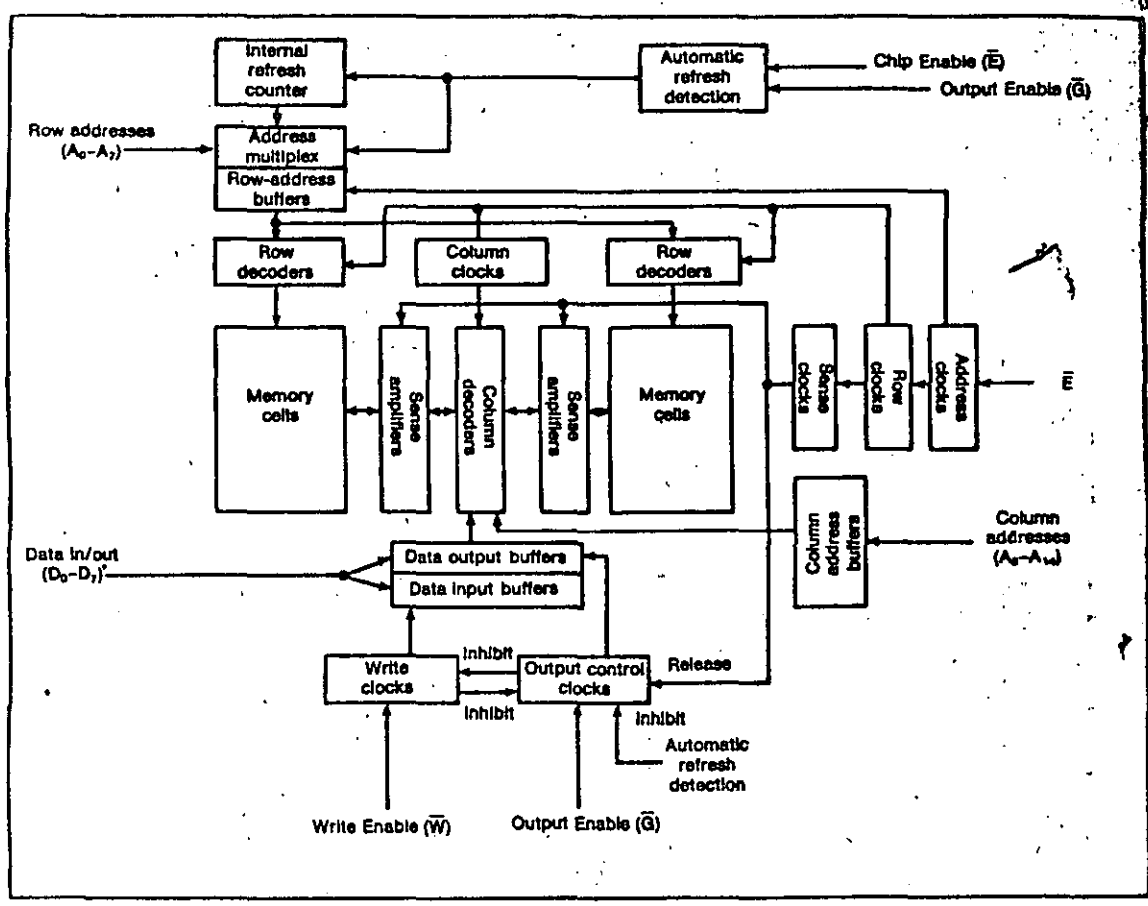
David A. Longfellow, Project Manager
Howard H. Sussman, Applications Engineering Manager
Mostek Corp.
1215 W. Crosby Rd., Carrollton, Texas 75006

in a fast RAM. In other words, the designer no longer has to worry about guaranteeing the hold time of the row address, switching between row and column addresses, or meeting the row-to-column setup times. However, he still must generate an output-enable strobe that is consistent with timing-edge skews and variations in the system's logic devices, but the task becomes easier as a result of the nonmultiplexed technique. The output-enable access time is a smaller percentage of the total access time than a column-address strobe (CAS) would be in a multiplexed design.

Internally, the 4856's nonmultiplexed architecture employs separate row and column address buffers (Fig. 1). The falling edge of the Chip Enable signal latches all addresses, letting the designer dispense with the synchronizing logic circuits normally used for multiplexed addressing. Because an address—represented by a row and a column—changes only once each cycle, setup and hold times are not as critical as with a multiplexed design. Addresses can be set up before and held long after the chip-enable strobe. That timing sequence, in turn, reduces the demands made on finely tuned delay lines, minimizes noise caused by address crosstalk, and imposes fewer constraints on the control of terminations and other transmission-line parameters of the address lines.

Since the dynamic RAM uses eight pins for both input and output, the user must multiplex some data. But data is easier to multiplex than are addresses, because there is more time allowed to switch between the data lines' input and output than there would be to switch addresses.

For instance, whereas the 4856's Output Enable command initiates only one chip function, the equivalent function in a multiplexed system, the command CAS must first select a column and then enable the output line. Thus the 4856 can wait longer for I/O data to be multiplexed than can an equally fast memory that must execute two or more data-calling commands. The 4856's larger data window translates into fewer glitches or other timing difficulties, as well as into a less costly system. At a given switching speed, the designer need not devote resources—in other words, time and money—to such critical parts of the system as normally matched I/O terminating lines.



1. The MK4856 dynamic RAM, organized as 32k by 8 bits, employs separate row- and column-address buffers to eliminate the timing constraints normally associated with multiplexing addresses. The chip's automatic refresh scheme, which includes a counter and lock-out circuitry for disabling row-address buffers and freeing the data output buffers, eliminates as many as 10 external circuits.

The need of dynamic RAMs to be refreshed discourages many system engineers from designing the chips into a system. Each row must have its data restored every 2 to 4 ms, depending on the particular device. Refreshing generally requires external logic, which obviously makes a designer's job more difficult. When external logic is added to a system containing relatively little memory, the real cost of each memory device increases substantially. On the other hand, in a system containing a large amount of memory, the cost of the additional circuitry can be spread over a large number of components and then recovered by packing the chips more tightly on a board.

Internal counter refreshes memory

With its integrated refresh counter, the 4856 manages to reduce system cost and complexity. Furthermore, when the chip is in its automatic refresh mode, it can also simplify system operation. Auto-

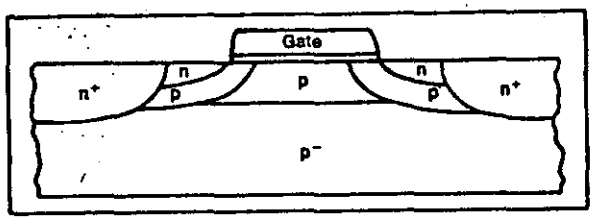
The making of a new memory

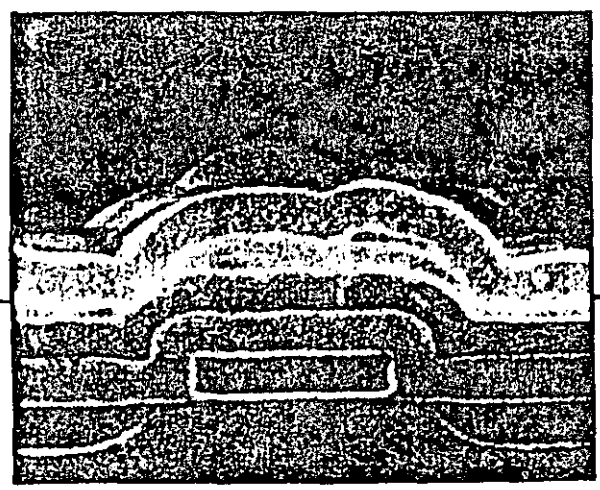
The MK4856 32k-by-8-bit dynamic RAM is built with an advanced NMOS technology that permits the device to be scaled down without encountering the short-channel effects that usually degrade operating margins and reduce long-term reliability. The manufacturing process uses ion implants to form extremely shallow source-to-drain junctions, in order to overcome the lower source-to-drain punch-through voltages caused by traditional scaling.

The RAM employs two layers of metallization, a structure that not only lends a distinct advantage to the memory designer who wants to develop a high-speed device, but also minimizes noise by strapping the signal and power lines better. A quieter environment leads to better device and system margins.

The NMOS process employs two slightly different arsenic implants and one boron implant. One arsenic implant, the n region, forms 0.1-μm junctions near each MOS gate; the other implant, the n^+ region, forms deeper 0.6-μm junctions further away (see the figure and photograph). In addition, the boron (shaded p region) is implanted around the shallow junctions to create a halo effect, which helps to stabilize the threshold voltages. Together, the arsenic and boron implants allow scaling to halve the channel length, and improve speed. They also reduce the electric fields around the transistor gate, thus limiting the number of hot electrons generated in that region and separating the generation site and the gate region to minimize the chance of lasting damage. Neither radically new equipment nor tradeoffs in device layout are needed for the process.

In addition to the advantages it affords the 32k-by-8 dynamic RAM, the process has produced an experimental 64-kbit dynamic RAM having channel lengths as small as 0.8 μm and access times approaching 50 ns. As a production part, the MK45H64 memory, organized as 64k by 1, has a guaranteed access time of 80 ns.

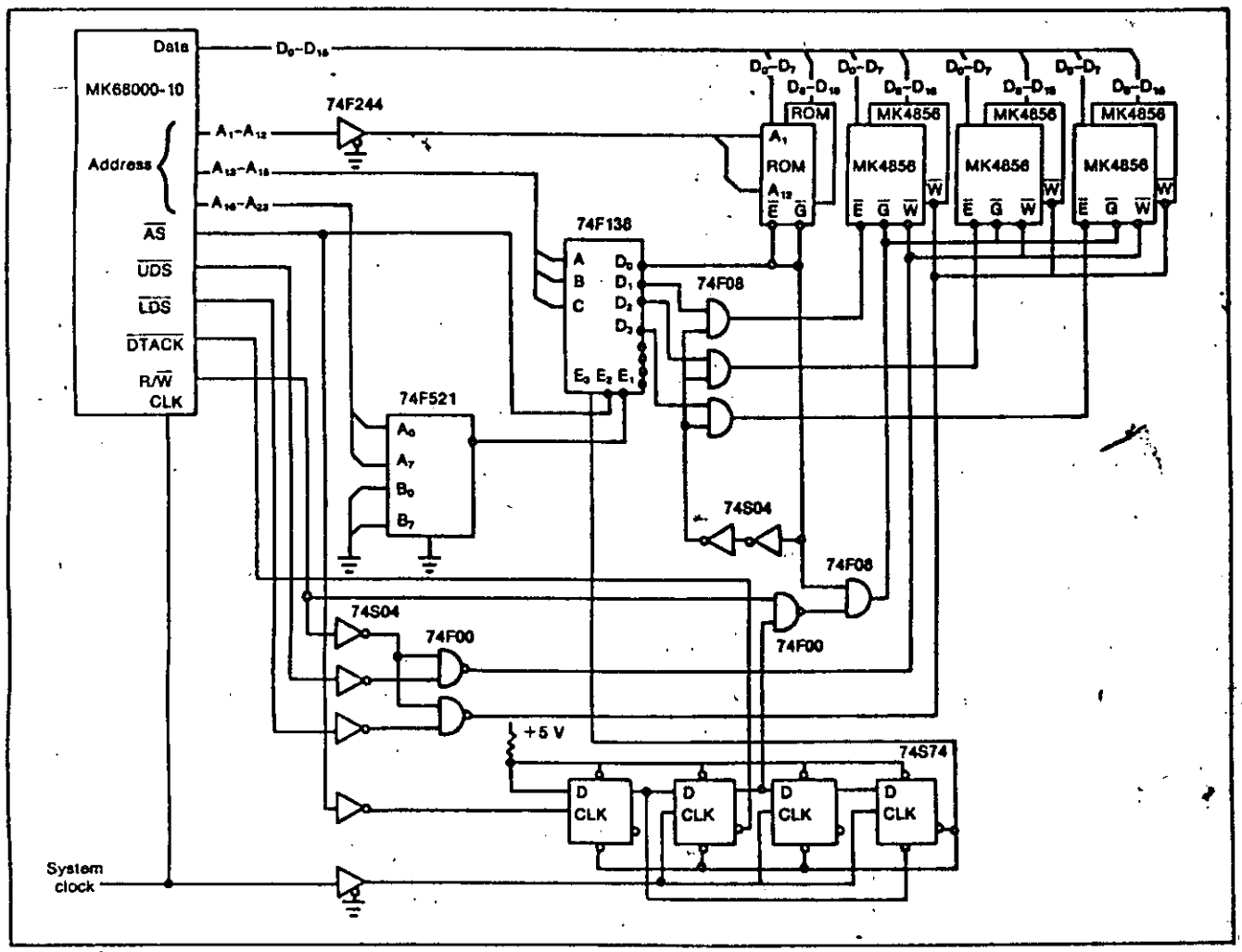




matic refreshing is initiated by bringing the Chip Enable line low, while the Output Enable line is low and the Read/Write line is high. On-chip circuitry monitors the status of each enabling line and brings the Automatic Refresh Detection line high when appropriate (see Fig. 1, again). The row-address buffers are first disconnected from external inputs A_0 through A_7 and then connected to the internal refresh counter. Later in the cycle, the counter is incremented, setting the address of the next refresh cycle. The data output buffers, which drive the external data pins, remain in the state they were in at the beginning of the cycle.

During normal operation, the 4856 can frequently execute this so-called hidden-refresh cycle, in which case the Output Enable line is kept low after a normal data access, thus maintaining valid data at the output drivers. Then the Chip Enable line is brought low, too. At this point, the chip begins refreshing automatically while holding the data captured during the previous access. If desired, several refresh cycles can be initiated while holding the same data, with speed limited only by the maximum time specified for enabling the output control line. In either case, the refresh circuitry is transparent to those designers who do not wish to use it.

To test the refresh counter, a designer writes a complement of logical 0s into the entire memory. During 256 consecutive refresh cycles, the Write Enable pin is brought low, as in a normal write cycle. The refresh counter determines the row address, and a column of logical 1s is written into memory without changing any external addresses. (Note that the Automatic Refresh Detection signal



2. A rudimentary 10-MHz system built around a 68000 microprocessor accesses data from a bank of dynamic RAMs offering 2 Mbits of memory. Only a few chips must be added to buffer signals and to implement the addressing scheme and clock-timing chain.

does not prevent the write clocks from functioning.) Finally, the designer puts the chip through a series of standard read cycles. If the refresh counter has correctly stepped through every row address, all rows will contain a column of logical 1s.

Applying the benefits

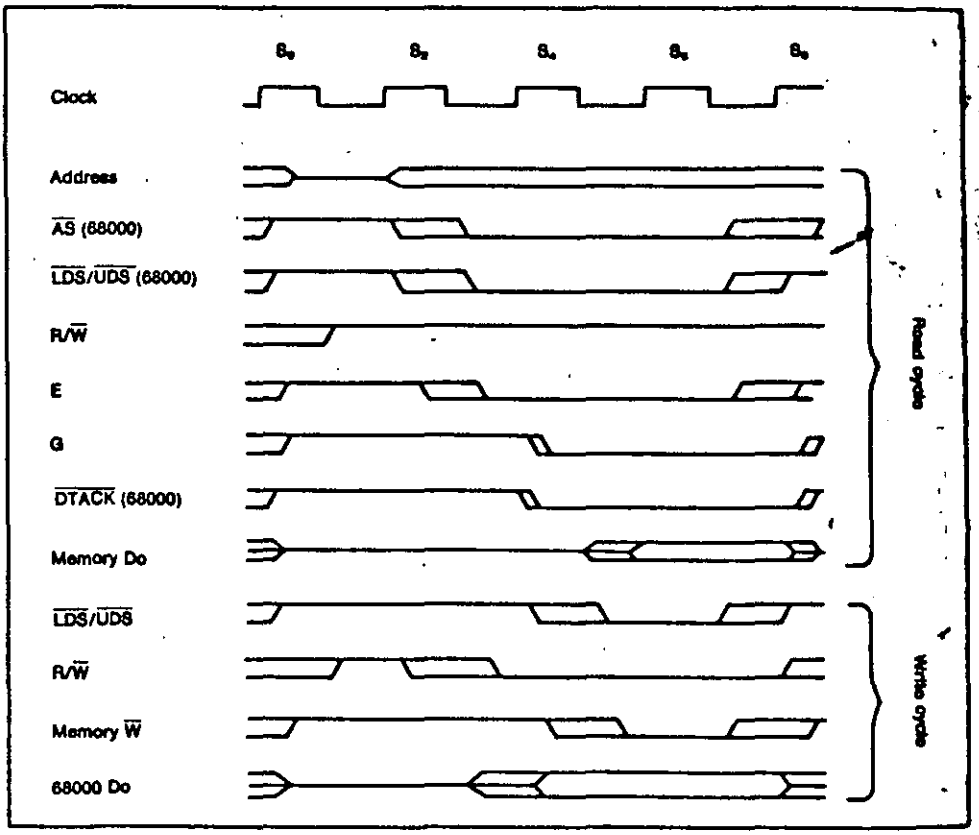
Among the advantages of using a fast byte-wide memory like the 4856 are a savings in board space brought about by the dense memory and a reduction in peripheral logic thanks to the nonmultiplexed addressing scheme and internal refresh circuitry—for a possible savings of 10 packages or more. The chip makes a significant impact on specific aspects of the design, such as a power supply: One 4856 typically draws 54 mA, whereas four equivalent devices organized as 16k by 4 bits consume 130 mA, in addition to the current drain of the peripheral logic.

Perhaps the best example of 4856's versatility is a rudimentary 68000 microprocessor-based system that runs at 10 MHz and accesses a bank of memories having chip access times of 120 ns each (Fig. 2). Although the 256-kbyte system contains both ROM and RAM, any type of 28-pin, 32k-by-8 device can be installed on the board using jumper options.

The system's parallel addressing scheme avoids address multiplexing and thereby removes a significant propagation delay customarily found in the access path of 1-bit-wide memories. The microprocessor easily interfaces with asynchronous memory peripherals and transfers data during eight half-cycles of the system clock (S_1 through S_8 in Fig. 3).

To avoid wait states and thus achieve maximum efficiency, the Data Transfer Acknowledge signal (DTACK) of the 68000 must be driven low during the fourth half-cycle of the system clock. DTACK is recognized during the fifth half-cycle (S_5) of the system clock, and the data line is sampled during S_6. If the memory cannot meet these timing constraints, a designer can insert wait states—prior to DTACK going low—simply by extending the minimum width of a half-cycle or by adding full cycles of the system clock.

To eliminate wait states entirely, the memory must be selected as soon as a valid address is decoded. The minimum setup time for determining the address prior to the Address Strobe signal (AS) is



3. The timing sequences during the dynamic RAM's read and write cycles are relatively similar. Only the upper and lower data strobes (UDS/LDS) and the R/W signal vary appreciably during the write cycle.

CIRCLE 107

32k-by-8 dynamic RAM

20 ns. An 8-bit identity comparator (the 75F521) decodes the memory address within 11 ns (maximum), which is less than the time required by the more traditional cascaded gating. The decoded addresses A_{13}, A_{14}, and A_{15} are stable before the AS line goes low; thus no glitches appear at the memories' chip-enable inputs.

In this system, the first 64 kbytes of memory are defined as the operating system, which is contained in ROM. Whenever that memory space is accessed, the Output Enable line of each 4856 device is driven low. Two gate delays later, the chip-enable lines are driven low, initiating a refresh cycle. Because an operating system is continuously accessed, the memory is refreshed many more times than the 256 cycles that are required during a 4-ms interval.

Even when a fast ROM is unavailable or when the operating system must be downloaded into RAM, this concept of refreshing remains valid. Of course, additional logic must be added to determine whether a memory cycle has been requested outside the operating system's address space. If one has been requested, the logic generates a refresh cycle for the operating-system RAMs.

The time that elapses between the system clock going high and AS going low is 55 ns, 10 ns longer than a clock phase in a 10-MHz system. Consequently, the status of AS cannot be determined at the falling edge of S_2. The leading edge of AS does, however, set a 74S74 flip-flop, so that the next rising edge of the clock defines S_4 and starts the memory-control timing chain.

Read and write cycles

Because the Read/Write line and the upper and lower data strobe lines (UDS and LDS, respectively) are invalid when the chip is enabled, the 4856 always initiates a read cycle. When a read cycle is requested, these lines become valid prior to S_4. Each Output Enable line is driven low during S_4 and remains valid until after the trailing edge of DTACK.

The 4856's output buffer, which has a delay of 35 ns, supplies valid data at the trailing edge of S_6. During a write cycle, the Read/Write line inhibits the Output Enable signal, but the Write signal will not be initiated until the data strobes become valid either during S_4 or at the beginning of S_6. Data from the microprocessor becomes valid before the data strobe lines go to logical 0. The memory executes the write function at the leading edge of $\overline{W}$.□

How useful?	Circle
Immediate design application	544
Within the next year	545
Not applicable	546

A 64-kbit EEPROM erases and programs simply and quickly, much like a RAM. Its on-chip charge pump supplies high programming voltages.

EEPROM adapts easily to in-system changes

A new generation of EEPROMs supplies almost ideal solutions to the once sticky problems of in-circuit erasing and programming of nonvolatile storage. Thanks to these chips, it is no longer necessary to limit programmable nonvolatile memory to read-only applications; today's EEPROMs are simple to program, eliminate the need for high external programming voltages, and link easily with other hardware. One such chip is the IMS3630, a 64-kbit EEPROM (organized as 8k by 8 bits) that is built with Nitrox storage cell technology (see "Nitrox Technology Shrinks Storage Cell Size").

When they were first introduced, EEPROMs solved one problem for designers by eliminating the ultraviolet light source needed to erase EPROMs. However, these early EEPROMs were difficult to use, since they required that address, data-input, and control signals remain stable over the lengthy programming and erasing intervals. Moreover, many of the first-generation chips also required an external elevated supply voltage (greater than 5 V) for programming and erasing. As a result of these difficulties, EEPROMs seldom served as anything other than read-only memories.

These drawbacks are overcome by the IMS3630. No external high voltages are required, thanks to the on-chip high-voltage generator. In addition, latched-program and erase-control features eliminate the need to hold any input signals valid during the extended intervals required for programming and erasing. That frees the system to perform other tasks.

The 3630's four primary commands—Read, Load Byte, Program/Erase Initiate, and Program/Erase Terminate—provide a new standard of simple and flexible EEPROM control. Read and Load Byte are similar to the read and write commands of a RAM: Read is used to read out data from the EEPROM array; Load Byte is used to obtain volatile write of one byte to a 64-byte buffer register, which holds the data that will be programmed in parallel into one full row of the EEPROM array.

All program and erase operations are begun with a Program/Erase Initiate command, which, during a short cycle time (300 ns) that follows, requests the desired operation. After the time required for the programming (10 ms) or erasure (100 ms) has elapsed, a Program/Erase Terminate command during another 300-ns cycle time that follows, stops the operation.

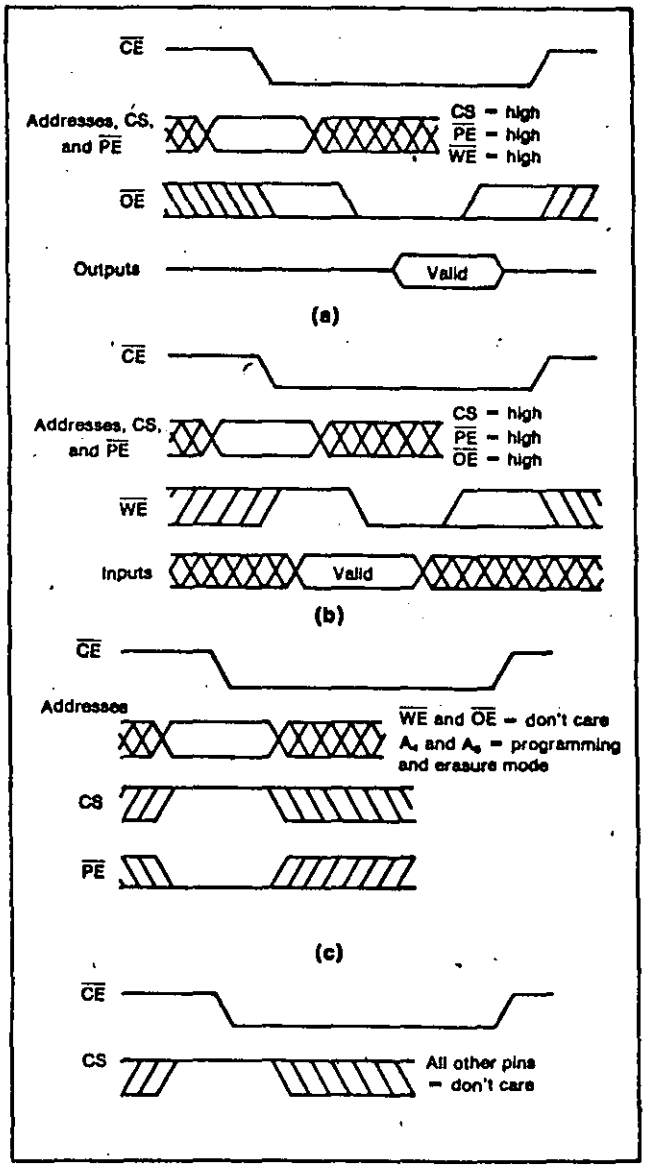
The buffer register and the parallel programming from it effectively speed the programming time 64 times that of single-byte programming. Conversely, parallel transfers can be made from any row of the EEPROM array to the buffer register. Use of this feature prior to erasure of a row allows single-bit or multiple-byte modification, which is not usually available with nitride-based EEPROMs.

Fred Jones, Manager Memory Applications and Strategic Marketing
Art Lancaster, Senior Design Engineer
Inmos Corp.
PO Box 16000, Colorado Springs, Colo. 80935

An improved pump decoder (Fig. 3b) decodes the high voltage. As in the multipliers, the decoder also eliminates MOS threshold drops with bootstrapping and thus provides improved output-voltage accuracy when compared with older pump-decoder designs.

Page-mode programming

As noted, the 3630's page mode allows up to 64 bytes of data to be modified in parallel, which makes possible very high programming data rates. Compared with conventional devices, page-mode programming reduces the time needed to program the device's entire 8 kbytes of memory from 82 to 1.38 s. This technique not only allows for high-speed in-system programming, but also decreases significantly the time needed to test the EEPROM.



2. The 3630 has four primary cycles and four program/erase modes: read cycle (a), load-byte cycle (b), program/erase initiate (c), and program/erase terminate (d).

Each of the chip's eight outputs serves 64 columns. Page-mode programming is implemented with one static latch connected to each of these columns. The data is written into these latches from the data-in buffers one byte at a time using load byte cycles, as previously described. During programming, if the latched data on a given column is a logical 1, the selected memory cell on that column will be programmed. If the latched data is a logical 0, the data in the selected memory cell will remain unchanged (that is, a logical 0 if previously erased).

During read operations, balanced differential sensing is performed to achieve the best possible noise immunity. To accomplish that, one sense amplifier, with a reference column, is assigned to each of the eight outputs. A precharge equilibration network, not previously found in EEPROMs, quickly shorts all 64 columns and the reference together to minimize sensing offsets. This is crucial for obtaining maximum retention and endurance.

Retention and endurance

High-performance, easy-to-use EEPROMs are of little value unless they can be erased and reprogrammed a large number of times. This ability, called endurance, must be accompanied by long-term data retention. Tests show that endurance and data retention are extremely good for the silicon Nitrox cell used in this device. Figure 4a is a curve of the memory cell thresholds for both erased and programmed states as a function of storage time. The measurements indicate that data will be retained more than 10 years, with over 10,000 program/erase cycles.

Users must often rely on the word of the manufacturer that an EEPROM will meet its endurance and retention specifications. The 3630, however, has a built-in margin test with which the threshold voltage (V_{TH}) of every memory cell on the chip can be measured, allowing extrapolation of these characteristics. To enter the testing mode, an illegal voltage—about 10 V—is applied to the data input D_7. Once enabled, the device remains in the test mode until an illegal voltage is applied to address pin A_{10}. While in the margin test mode, multiple $\overline{CE}$ cycles are used to address the individual cells to be measured. The outputs are automatically enabled in this mode, allowing the $\overline{OE}$ pad to become the analog input for the margin voltage. The analog voltage on $\overline{OE}$ is routed directly to the second-level polysilicon gate of the storage cell. Whenever the analog input voltage exceeds the threshold of the selected cell, the Data Out line associated with the cell goes low. Thus this analog input voltage is a direct measure of the cell's sensing threshold.

One interesting feature of the Nitrox cell is its

rapid initial threshold shift during programming and erasure (Fig. 4b). Even with programming times as short as 1 ms, the threshold voltage—about 2 V—is easily detected as a 1 by the chip's internal balanced sense-amplifier.

Although a 1-ms programming time is one tenth that required for data to be retained a full 10 years, it is sufficient when only brief retention is required, such as when a system power failure occurs over a weekend. A similar tenfold improvement is possible for erasure. So-called self-timed EEPROMs cannot make this claim, but latched programming and erasure permits a user to control program and erase times and provides far more flexibility than is found in other devices.

Even if a long retention time is needed, it can be achieved with short programming intervals. To do so, the data is first programmed on a short cycle, with a correspondingly short retention time. Later, the information can be replicated using a full-duration programming cycle. Replication is accomplished with a program/erase sequence that first uses the instruction A_4 equals 0, A_5 equals 1 (load latches from a row and erase the same row) followed by a second program/erase sequence using the instruction A_4 equals 1, A_5 equals 1 (program a row from the column latches). Thus the data can be replicated without transferring it out of the device or using external resistors.

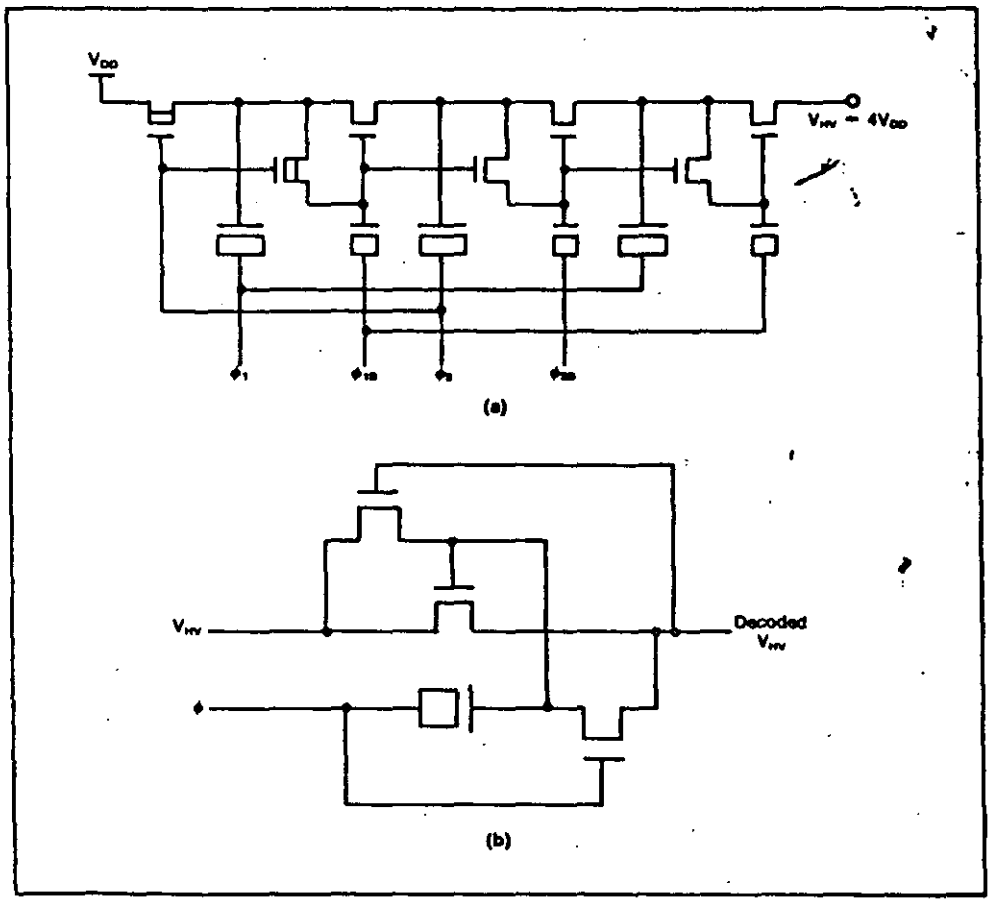
The process must be repeated for each row that is to be restored to its full data-retention level. If the entire memory is to be replicated, the 3630 requires that all 128 row locations be reprogrammed using the above sequence.

Another advantage of exercising control over the program/erase interval is that it allows the system to abort modification of nonvolatile memory. Even if sufficient time has not elapsed to guarantee that the programmed or erased cells have adequate voltage levels for replication, there may be more important data that must be written immediately into the EEPROM.

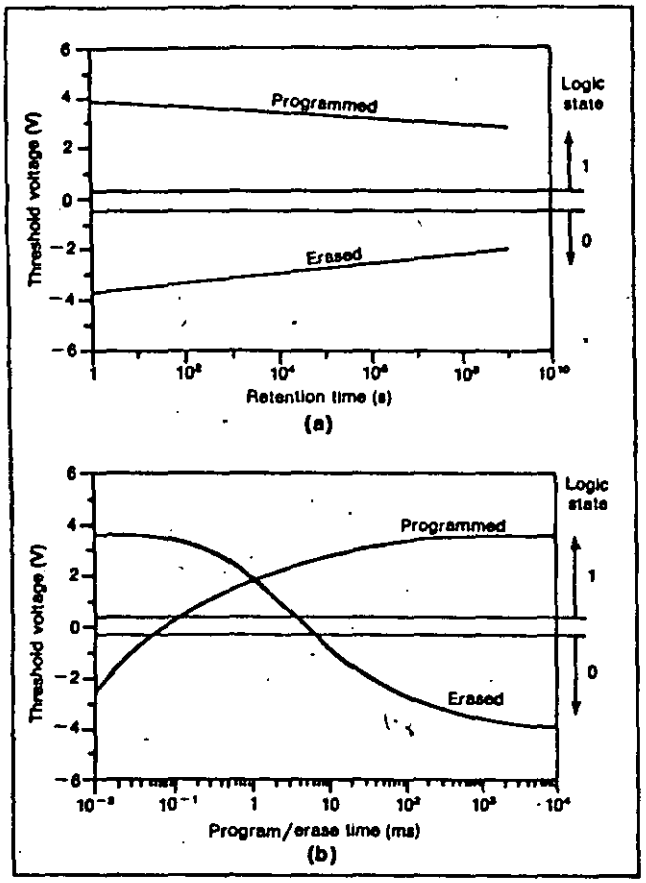
Application is easy

Very little, if any, additional hardware is required to connect the 3630 to most popular microprocessors. In many applications, the device can be plugged directly into the byte-wide memory sockets and will execute all operations under software control.

For example, the 3630 is easily interfaced with a Z80 microprocessor and 8 kbytes of dynamic RAM



3. New high-voltage circuits—a high-voltage generator (a) and a high-voltage decoder (b)—eliminate complicated regulators by maintaining accurate output voltages.



4. Long-term data retention is an essential characteristic of an EEPROM. Memory cell thresholds are functions of storage time (a). Data retention—over 10 years, or 10,000 program/erase cycles, is expected—depends on programming or erasure times (b).

(Fig. 5). It creates a system that can be expanded easily with more RAM or EEPROM. A RAM cycle is selected when address line A_{15} is high and $\overline{MREQ}$ is low. An EEPROM cycle is selected when A_{15} goes low and $\overline{MREQ}$ is also low. The address mapping for the EEPROM requires that A_{14} be low for read, load-byte, and terminate cycles, but high for all program/erase operations.

In the system shown, program/erase operations are timed with software loops executed from RAM. Alternatively, this timing could have been done using programmable interrupt timers. The easy in-system provision for data modification of nonvolatile memories includes any operating-system software stored in an EEPROM. Thus powerful self-adaptive systems are now possible with the 3630.□

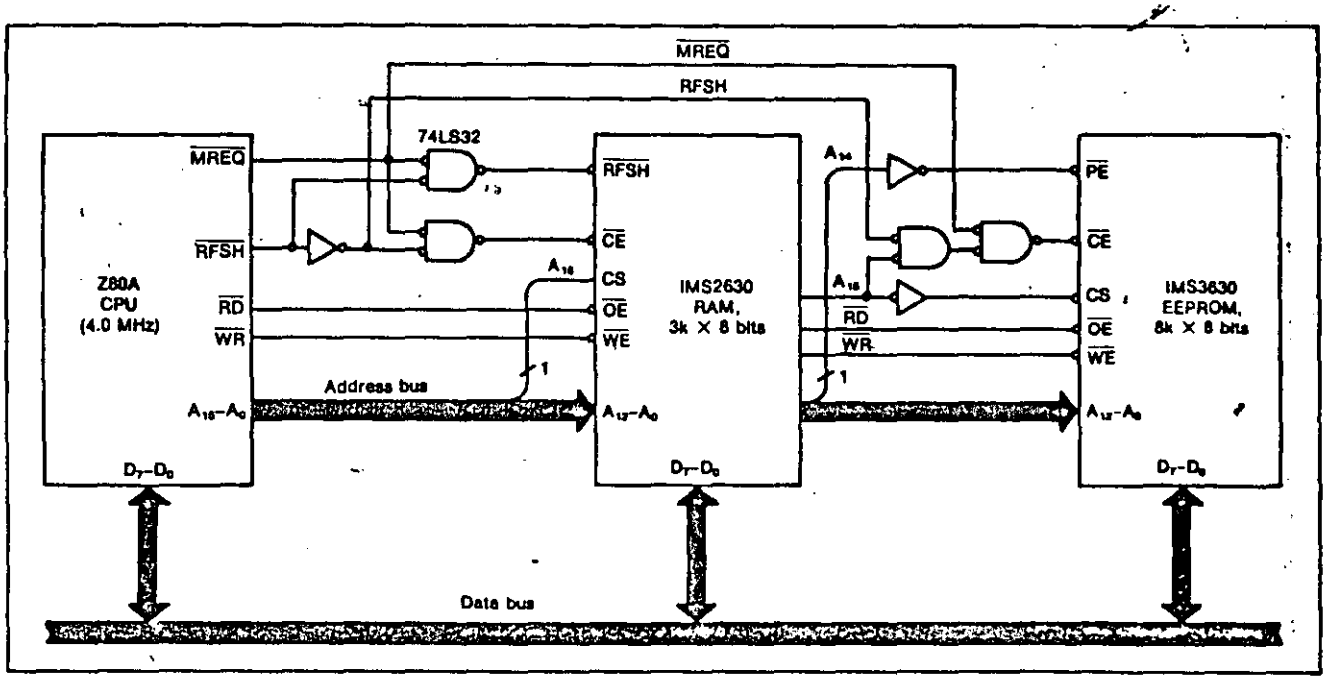
Bibliography

1. Dham, V. K., et al. "A 5-V Only E2PROM Using 1.5 μm Lithography," *ISSCC Digest of Technical Papers*, pp. 166-167, February 1983.

2. Gupta, A., et al. "A 5-V Only 16 k EEPROM utilizing Oxynitride Dielectrics and EPROM Redundancy," *ISSCC Digest of Technical Papers*, pp. 184-185, February 1982.

3. Lancaster, A., et. al. "A 5-V Only EEPROM with Internal Program/Erase Control," *ISSCC Digest of Technical Papers*, pp. 164-165, February 1983.

How useful?	Circle
Immediate design application	547
Within the next year	548
Not applicable	549



5. Connecting the 3630 to a Z80 microprocessor requires little additional hardware.

General-purpose microprocessors: Performance and features

Word size, data/instruction (bits)	Original source	Processor	Process technology	Direct addressing range (words)	No. of basic instructions	Maximum clock frequency (MHz)/phases	Instruction time, shortest/longest[a] (μs)	TTL-compatible	BCD arithmetic	On-chip interrupts/levels	No. of internal general-purpose registers	No. of stack registers
1/4	Motorola	MC14500	CMOS	0	16	1/1	1/1	Yes	No	Yes/1	1	0
4/8	Fujitsu	MB8849	NMOS	2 k	70	2	3/6	Yes	Yes	Yes/2	128	4
4/8		MB8850	CMOS	2 k	70	2	3/6	Yes	Yes	Yes	128×4	RAM
4/8		MB8859	CMOS	2 k	70	2	3/6	Yes	Yes	Yes	128×4	RAM
4/8		MB88408	NMOS	4 k	75	2	3/6	Yes	Yes	Yes	192×4	RAM
4/8		MB88409	NMOS	4 k	75	2	3/6	Yes	Yes	Yes	192×4	RAM
4/8		MB88418	NMOS	4 k	75	2	3/6	Yes	Yes	Yes	192×4	RAM
4/8		MB88419	NMOS	8 k	75	2	3/6	Yes	Yes	Yes	192×4	RAM
4/8	National Semiconductor	COP402	NMOS	1 k	49	4/1	4	Yes	Yes	Yes/3	64×4	RAM
4/8		COP402M	NMOS	1 k	49	4/1	4	Yes	Yes	No	64×4	RAM
4/8		COP404L	NMOS	2 k	49	2/1	15	Yes	Yes	Yes/3	128×4	RAM
4/8	NEC Electronics	μPD556B	PMOS	2 k	80	0.44/1	9.1/25	Yes	Yes	Yes/2	96×4	3
4/8		μPD7500G	CMOS	8 k	110	0.2/1	10	Yes	Yes	Yes/5	256×4	RAM
4/8		μPD75CG08	CMOS	4 k	92	0.2/1	10	Yes	Yes	Yes/3	224×4	RAM
4/8	Oki	MSM5840H	CMOS	4 k	98	4/1	7.6/15.2	Yes	Yes	Yes/2	128×4	4
4/8	Panasonic (Matsushita)	MN1498	NMOS	1 k	68	0.3/1	10/20	Yes	Yes	Yes/1	64×4	RAM
4/8		MN1499	NMOS	2 k	75	0.3/1	10/20	Yes	Yes	Yes/1	64×4	RAM
4/8		MN1499A	NMOS	2 k	75	0.3/1	10/20	Yes	Yes	Yes/1	128×4	RAM
4/8		MN1599	NMOS	4 k	125	1/1	2/4	Yes	Yes	Yes/4	256×4	RAM
4/8		LM6499	NMOS	2 k	82	1.2	3	Yes	Yes	Yes	128×4	4
4/8		LM64PG99	NMOS	2 k	82	1.2	3	Yes	Yes	Yes	128×4	4
4/8		LM6497	NMOS	1 k	65	1.0	4	Yes	Yes	No	64×4	2
4/8		LM64PG97	NMOS	1 k	65	1.0	4	Yes	Yes	No	64×4	2
4/8		LC65PG99	CMOS	4 k	81	0.4	10	Yes	Yes	Yes	64×4	4
4/8	Texas Instruments	SE1000JLP	PMOS	1 k	43	0.40/1	15/60	No	Yes	No	66×4	1
4/8		SE1100JLP	PMOS	2 k	40	0.40/1	15/60	No	Yes	No	130×4	1
4/8		SE1400JLL	PMOS	4 k	41	0.55/1	11/60	No	Yes	No	130×4	3
4/8		SE2100JLL	PMOS	2 k	55	0.55/1	11/60	No	Yes	Yes/1	130×4	4
4/8		SE1000JLC	CMOS	1 k	43	1/1	6/120	Yes	Yes	No	66×4	3
4/8		SE1100JLC	CMOS	2 k	40	1/1	6/120	Yes	Yes	No	130×4	3
4/8		SE2130JLL	PMOS	2 k	53	0.5/1	12/60	No	Yes	No	130×4	3
4/8	Toshiba	TMP4300C	NMOS	2 k	35	0.5/1	4/8	Yes	No	Yes/1	128×4	4×11
4/8		TMP4399C	NMOS	2 k	25	0.5/1	4/8	Yes	No	Yes/1	128×4	4×11
4/8		TCP4600AC	CMOS	4 k	52	0.2/1	10.20	Yes[c]	No	Yes/1	160×4	1×12
4/8		TMP4700C	NMOS	4 k	90	5/1	2/4	Yes	No	Yes 6	256×4	15×12
8/8	Commodore Semiconductor	MCS-650X	NMOS	64 k	56	4/1	0.5/3.5	Yes	Yes	Yes 1	0	RAM
8/8		MCS-651X	NMOS	64 k	56	4/2	0.5/3.5	Yes	Yes	Yes/1	0	RAM
8/8		MCS-6508	NMOS	64 k	56	4/2	0.5 3.5	Yes	Yes	Yes 1	256	RAM
8/8		MCS-65C0X	CMOS	64 k	56	4/1	0.5 3.5	Yes	Yes	Yes/1	0	RAM
8/8		MCS-65C1X	CMOS	64 k	56	4/2	0.5 3.5	Yes	Yes	Yes 1	0	RAM
8/8	Fairchild	F8 (3850)	NMOS	64 k	69	2/1	2/13	Yes	Yes	Yes/1	64	RAM
8/8	Fujitsu	MBL6800	NMOS	64 k	72	2/2	1/25	Yes	Yes	Yes/1	0	RAM

Q = quad-in-line package.
N.a. = not applicable.

[a] With the maximum clock.
[b] Standard TTL or MOS circuits will suffice.
[c] TTL output and CMOS input.
[d] Except clock lines
[e] String search.
[f] At 5 V, as the supply voltage increases, clock frequency may increase.

This table summarizes both existing general-purpose microprocessors and those that have been introduced since last year's update (ELECTRONIC DESIGN, Nov. 26, 1981, p. 114). Special features of the processors are flagged under "Comments." The data pages, which begin on p. 147 of this issue, offer more information about the new entries. The microprocessors are arranged first by word size and then alphabetically by manufacturer. For more information from the chip makers, circle the appropriate reader service numbers given in the table starting on p. 157.

On-chip clock	DMA capability	Specialized memory & I/O circuits available	Prototyping system available	Package size (no. of pins)	Voltages required (V)	Assembly-language development system	High-level languages	Time-sharing cross software	Comments
Yes	No	No[a]	No	16	3 to 18	No	No	No	Needs external program counter
Yes	No	No	Yes	64	5	Yes	No	Yes	ROM-less emulator chip for M88840 family of all-in-one processors
Yes	No	No	Yes	42	5	Yes	No	No	Evaluation chip for MB8850 series with EPROM
Yes	No	No	Yes	82	5	Yes	No	No	Same as above, but with address line break-out
Yes	No	No	Yes	42	5	Yes	No	No	Evaluation chip for MB88400 series with EPROM
Yes	No	No	Yes	82	5	Yes	No	No	Same as above, but with address line break-out
Yes	No	No	Yes	42	5	Yes	No	No	Evaluation chip with 8-bit, 8-channel a-d converter
Yes	No	No	Yes	82	5	Yes	No	No	Evaluation chip with 8-bit, 8-channel a-d converter and 8-kbit ROM
Yes	No	Yes	Yes	40	4.5 to 6.3	Yes	No	Yes	ROM-less version of COP420 and 440 single-chip microcomputer with serial I/O, 20 I/O lines, event counting
Yes	No	Yes	Yes	40	4.5 to 6.3	Yes	No	Yes	
Yes	No	Yes	Yes	40	4.5 to 9.5	Yes	No	Yes	
Yes	No	No	Yes	64Q	−10	Yes	No	Yes	ROM-less version of μCOM-4 series devices
Yes	No	Yes	Yes	64Q	5	Yes	No	No	ROM-less version
Yes	No	Yes	Yes	40	5	Yes	No	No	Package contains piggyback ROM
Yes	No	Yes	Yes	42	3 to 6	Yes	No	Yes	ROM-less evaluation chip; includes 30 I/O lines and 8-bit timer-counter; draws 0.8 mA
Yes	No	Yes	Yes	40	5	Yes	No	Yes	ROM-less version of MN1402, but with 68 instructions and comes in a 40-pin package
Yes	No	Yes	Yes	64	5	Yes	No	Yes	ROM-less version of MN1400
Yes	No	Yes	Yes	64	5	Yes	No	Yes	ROM-less version of MN1405, but 128 nibbles of on-chip RAM
Yes	Yes	Yes	Yes	64	5	Yes	No	Yes	ROM-less version of MN1564; has 12 4-bit I/O ports
Yes	No	Yes	Yes	80Q	4.5 to 5.5	Yes	No	No	ROM-less chip of LM6402/05 (A, H, L)
Yes	No	Yes	Yes	42	4.5 to 5.5	Yes	No	No	ROM-less piggyback package version of LM6499
Yes	No	Yes	Yes	80Q	4.5 to 6.5	Yes	No	No	ROM-less version of LM6416/13
Yes	No	Yes	Yes	42	4.5 to 6.5	Yes	No	No	ROM-less piggyback package version of LM6497
Yes	No	Yes	Yes	42	4.5 to 5.5	Yes	No	No	ROM-less piggyback package version of LC6599
Yes	No	Yes	Yes	64	−15	Yes	No	Yes	ROM-less version of TMS1000/1200 microcomputer
Yes	No	Yes	Yes	64	−15	Yes	No	Yes	ROM-less version of TMS1100/1300 microcomputer
Yes	No	Yes	Yes	64	−9	Yes	No	Yes	ROM-less version of TMS1400 microcomputer
Yes	No	Yes	Yes	64	−9	Yes	No	Yes	ROM-less version of TMS2100/2300 microcomputer
Yes	No	Yes	Yes	64	5	Yes	No	Yes	ROM-less version of 1-kbyte CMOS microcomputer
Yes	No	Yes	Yes	64	5	Yes	No	Yes	ROM-less version of 2-kbyte CMOS microcomputer
Yes	No	Yes	Yes	64	−9	Yes	No	Yes	ROM-less microcomputer with sound generator
Yes	No	No	Yes	64	5	Yes	No	Yes	ROM-less evaluator chip for TLCS-43 family
Yes	No	No	Yes	42	5	Yes	No	Yes	Piggyback ROM-less package; emulates TMP4315BP and TMP4320AP
Yes	No	No	Yes	42	5	Yes	No	Yes	ROM-less evaluator chip for TLCS-40A family
Yes	No	No	Yes	40	5	Yes	No	Yes	ROM-less evaluator chip for TLCS-47 family
Yes	No	Yes	Yes	40	5	Yes	Yes	Yes	Provides 13 addressing modes
No	No	Yes	Yes	40	5	Yes	Yes	Yes	Similar to 650X but needs two-phase clock
No	Yes	Yes	Yes	40	5	Yes	Yes	Yes	Similar to 651X with 8-bit I/O port and RAM
Yes	No	Yes	Yes	40	5	Yes	Yes	Yes	Totally compatible with NMOS version
No	No	Yes	Yes	40	5	Yes	Yes	Yes	Two-chip set; totally compatible with NMOS version
Yes	No	Yes	Yes	40	5, 12	Yes	Yes	Yes	Usually used with program storage unit
No	Yes	Yes	Yes	40	5	Yes	Yes	Yes	Compatible with the MC6800

[a] Frame pointer, too.
[b] Has 8-bit external buses and 16-bit internal buses.
[c] The clock is internally divided by 4 or 6, depending on the instruction.
[d] Double-precision 16-bit operations are available.
[e] Range in bytes.
[f] 9980 only.

Microprocessor Special: General-purpose chips

Word size, data/instruction (bits)	Original source	Processor	Process technology	Direct addressing range (words)	No. of basic instructions	Maximum clock frequency (MHz)/phases	Instruction time, shortest/longest[a] (μs)	TTL-compatible	BCD arithmetic	On-chip interrupts/levels	No. of internal general-purpose registers	No. of stack registers
8/8	Toshiba	TMP8035P/8039P-6	NMOS	64 k	96	6/1	2.5/5	Yes	Yes	Yes/1	64	RAM
8/8		TMP80C35P	CMOS	64 k	96	6/1	2.5/5	Yes	Yes	Yes/1	64	RAM
8/8		TMP80C39P-6	CMOS	4 k	96	6/1	2.5/5	Yes	Yes	Yes/1	64×8	RAM
8/8		TMP80C39AP	CMOS	64 k	97	11/1	1.36/2.7	Yes	Yes	Yes/1	64	RAM
8/8		TMP8080A	NMOS	64 k	78	2.6/2	1.5/3.75	Yes[c]	Yes	Yes/1	8	RAM
8/8		TMP8085AP	NMOS	64 k	80	5.5/1	1.3/5.2	Yes	Yes	Yes/4	8	RAM
8/8	Zilog	Z80H	NMOS	64 k	150+	8/1	0.7/3.5	Yes	Yes	Yes/1	14	RAM
8/8		Z8612	NMOS	126 k	47	12/1	1.5/3.75	Yes	Yes	Yes/6	144×8	RAM
8/8		Z8603/8613	NMOS	126 k	47	12/1	1/3.33	Yes	Yes	Yes/6	144×8	RAM
8/8		Z8671	NMOS	126 k	47 + Tiny Basic	8/1	1.5/3.75	Yes	Yes	Yes/6	144×8	RAM
8/8		Z8681	NMOS	126 k	47	12/1	1/3.33	Yes	Yes	Yes/6	144×8	RAM
8/16	Fujitsu	MBL8088	NMOS	64 k	97	10/1	0.2/19	Yes	Yes	Yes/1	8	RAM
8/16	Motorola	MC68008	HMOS	1 M	56	8/1	0.5/N.a.	Yes	Yes	Yes/3	17	15: 32
8/16	Signetics	8X300	Bipolar	8 k	N.a.	5/1	0.2	Yes	No	No	8	0
8/16		8X305	Bipolar	8 k	N.a.	8/1	0.2	Yes	No	No	15	0
8/16	Zilog	Z8108	NMOS	512 k	[m]	25/1	0.4[m]	Yes	Yes	Yes/7	16	RAM
8/32	National Semiconductor	NS16008	NMOS	16 M[k]	100 +	N.a.	N.a.	Yes	Yes	Yes	8	RAM
12/12	Harris Semiconductor	6120	CMOS	64 k	85	8/1	1/3.75	No	No	Yes/2	1	2
12/12	Intersil	IM6100	CMOS	4 k	81	3.3/1[f]	2.5/5.5	Yes	No	Yes/1	0	RAM
12/12	Toshiba	T3535	PMOS NMOS	4 k	108	2.5/1	10/30	Yes	No	Yes/8	8	RAM
12/16	LSI Computer Systems	LS7270	NMOS	4 k	10	2/1	48/72	Yes	No	Yes/4	15×1	3×12
16/16	American Microsystems	S99C91	CMOS	64 k	68	5/1	2.4/54.4	Yes[c]	Yes	Yes/4	3	RAM
16/16		AMU/PR	CMOS	64 k	68	3/1	4/90.7	Yes[c]	Yes	Yes/5	128×8	RAM
16/16	Data General	mN602	NMOS	64 k	82	8.3/2	2.4/53	Yes	No	Yes/16	4[g]	RAM
16/16	Digital Equipment	DCT11-AA	CMOS	32 k	55	7.5/1	1.6/14.5	Yes	No	Yes/4	8	1
16/16	Fairchild	F9445	I²L	64 k	100	20/1[f]	0.3/5.7	Yes	No	Yes/16	8	RAM
16/16	Ferranti	F100L	Bipolar	32 k	153[j]	14/1	1.19/14	Yes	No	Yes/1	RAM	RAM
16/16	Fujitsu	MBL8086	NMOS	1 M	97	10/1	0.2/19	Yes	Yes	Yes/1	8	RAM
16/16	Harris Semiconductor	80C86	CMOS	1 M	97	8/1	0.25/24	Yes	Yes	Yes/1	8	RAM
16/16	Intel	iAPX 86/10	NMOS	1 M[k]	97	10/1	0.2/19	Yes	Yes	Yes/1	8	RAM
16/16[h]		iAPX 88/10	NMOS	64 k[k]	97	10/1	0.2/19	Yes	Yes	Yes/1	8	RAM
16/16		iAPX 186/10	NMOS	1 M[k]	150	10/1	0.2/6.7	Yes	Yes	Yes/5	8	RAM
16/16		iAPX 286/10	NMOS	16 M	163	10/1	0.2/17.6	Yes	Yes	Yes/1	8	RAM
16/16		8089	NMOS	500 k	46	5/1	0.4/4	Yes	No	Yes	No	N.a.
16/16	Matra-Harris	HM-8086H	NMOS	1 M	97	10/1	0.2/19	Yes	Yes	Yes/1	8	RAM
16/16		HM-8088H	NMOS	1 M	97	10/1	0.2/19	Yes	Yes	Yes/1	8	RAM
16/16	Mikros Systems	MKS16	SOS/CMOS	32 k	See Comments	5.5/1	N.a.	Yes	No	Yes/2	16	RAM
16/16	Motorola	MC68000	HMOS	16 M[k]	56	12/1	0.375/N.a.	Yes	Yes	Yes/7	17	15×32
16/16		MC68010	HMOS	16 M	58	8/1	0.5/N.a.	Yes	Yes	Yes/7	17	15×32
16/16	Matsushita	MN1610	NMOS	64 k	33	2/2	2/6	Yes[d]	No	Yes/3	5	RAM
16/16		MN1613	NMOS	256 k	97	3.3/1	0.3/58	Yes	Yes	Yes/3	5×16	RAM
16/32	National Semiconductor	NS16016	NMOS	16 M	78/100 +	N.a.	N.a.	Yes	Yes	Yes	8	RAM
16/32		NS16032	NMOS	16 M[k]	100 +	N.a.	N.a.	Yes	Yes	Yes	8	RAM
16/16	Siemens	SAB8086-1	NMOS	1 M	97	10/1	0.2/19	Yes	Yes	Yes/1	8	RAM
16/16		SAB80286	NMOS	2[24]	163	10/1	0.2/17.6	Yes	Yes	Yes/1	8	RAM
16/16[h]	Texas Instruments	TMS9980/9981	NMOS	8 k	69	4/4	3.2/49.6	Yes[d]	No	Yes/4	16	RAM
16/16[h]		TMS9995	NMOS	32 k	73	12/1	1/15	Yes	No	Yes/7	128×16	RAM

Q = quad-in-line package.
N.a. = not applicable.

[a]With the maximum clock.
[b]Standard TTL or MOS circuits will suffice.
[c]TTL output and CMOS input.
[d]Except clock lines.
[e]String search.
[f]At 5 V; as the supply voltage increases, clock frequency may increase.

On-chip clock	DMA capability	Specialized memory & I/O circuits available	Prototyping system available	Package size (no. of pins)	Voltages required (V)	Assembly-language development system	High-level languages	Time-sharing cross software	Comments
Yes	Yes	Yes	Yes	40	5	Yes	Yes	Yes	ROM-less versions of 8048/8049
Yes	Yes	Yes	Yes	40	5	Yes	Yes	Yes	Pin-compatible with the 80C3J
Yes	No	Yes	Yes	40	5	Yes	No	Yes	CMOS version of 8039
Yes	Yes	Yes	Yes	40	5	Yes	Yes	Yes	Includes halt instruction
No	Yes	Yes	Yes	40	5, 12, −5	Yes	Yes	Yes	Pin-compatible with the 8080A
Yes	Yes	Yes	Yes	40	5	Yes	Yes	Yes	8080-code−compatible
No	Yes	Yes	Yes	40	5	Yes	Yes	Yes	Low-power version (the Z80L) is available
Yes	No	Yes	Yes	64	5	Yes	Yes	Yes	ROM-less version of Z8 microcomputer; all ROM address and data lines are available on the 64 pins of the DIP package
Yes	No	Yes	Yes	40	5	Yes	Yes	Yes	2-kbit and 4-kbit versions respectively, in piggyback ROM-less packages
Yes	No	Yes	Yes	40	5	Yes	Yes	Yes	Z8 microcomputer with Tiny Basic in ROM; another version, the Z8675, has power-down
Yes	No	Yes	Yes	40	5	Yes	Yes	Yes	40-pin ROM-less version of Z8 microcomputer; still has two counter-timers, serial I/O port, 24 programmable I/O lines
Yes	Yes	Yes	Yes	40	5	Yes	Yes	Yes	Pin-compatible with the 8088
No	Yes	Yes	Yes	64	5	Yes	Yes	Yes	Reduced-bus version of MC68000 for cost-sensitive systems
Yes	No	Yes	Yes	50	5	Yes	Yes	Yes	Intended for high-speed controllers
Yes	No	Yes	Yes	50	5	Yes	Yes	Yes	New chip. Enhanced version of the 8X300
Yes	Yes	Yes	Yes	40	5	Yes	Yes	Yes	New chip. Enhanced version of the Z80, binary-code−compatible, with on-chip memory management, programmable dynamic memory refreshing, programmable bus timing
No	Yes	Yes	Yes	48	5	Yes	Yes	No	Has 32-bit internal bus, 16-bit data bus
Yes	Yes	Yes	No	40	5	Yes	Yes	Yes	PDP-8E instruction set; up to four times the throughput of the 6100
Yes	Yes	Yes	Yes	40	4 to 11	Yes	Yes	Yes	Emulates PDP-8 instruction set
Yes	Yes	Yes	Yes	36	5, −5	Yes	Yes	Yes	Has multiplication and division instructions
Yes	No	No	No	40	5 to 9	No	No	No	Serial address/instruction
Yes	Yes	Yes	Yes	40	5	Yes	Yes	Yes	S9900-code−compatible
Yes	Yes	Yes	Yes	48	5	Yes	Yes	Yes	Alterable microcomputing prototyping chip
No	Yes	Yes	Yes	40	12, ±5	Yes	Yes	Yes	Executes the Nova instruction set; has on-chip multiply and divide instructions
Yes	Yes	Yes	Yes	40	5	Yes	Yes	Yes	New chip. Microprocessor implementation of the PDP-11 minicomputer; uses the same basic instruction set. User selects 8- or 16-bit data bus
No	Yes	Yes	Yes	40	5	Yes	Yes	Yes	Executes Nova IV instruction set plus high-speed multiply and divide
No	Yes	Yes	Yes	40	5, 1.2	Yes	Yes	Yes	Can do double-word operations, hardware multiplication and division with 101-L
Yes	Yes	Yes	Yes	40	5	Yes	Yes	Yes	8-bit-bus version of 8086
Yes	Yes	Yes	Yes	40	5	Yes	Yes	Yes	In development
No	Yes	Yes	Yes	40	5	Yes	Yes	Yes	Has 24 addressing modes
No	Yes	Yes	Yes	40	5	Yes	Yes	Yes	8-bit-bus version of 8086 microprocessor
Yes	Yes	Yes	Yes	68	5	Yes	Yes	Yes	New chip. Has on-chip clock generator, DMA controller, interrupt controller timers, chip-selection and wait-state logic, local bus controller
No	Yes	Yes	Yes	68	5	Yes	Yes	Yes	New chip. Has on-chip memory management and protection with hardware support for task switching and virtual memory
No	Yes	Yes	Yes	40	5	Yes	Yes	Yes	Dedicated processor for I/O; has two 8-bit or one 16-bit channel; maximum throughput is 1.25 Mbytes/s
No	Yes	Yes	Yes	40	5	Yes	Yes	Yes	
No	Yes	Yes	Yes	40	5	Yes	Yes	Yes	Version of 8086 with 8-bit internal bus, 16-bit data bus
No	Yes	Yes	Yes	48	5 to 10	Yes	Yes	No	New chips. Six-chip Multibus CMOS-on-sapphire set; number of basic instructions is microprogrammable
No	Yes	Yes	Yes	64	5	Yes	Yes	Yes	Has 32-bit-wide internal structure
No	Yes	Yes	Yes	64	5	Yes	Yes	Yes	New chip. Virtual-machine version of the MC68000; allows instructions to be continued after a fault
No	Yes	Yes	No	40	5, 12, −3	Yes	No	No	
Yes	Yes	Yes	Yes	40	5	Yes	Yes	Yes	Has serial port and 16-bit timer
No	Yes	Yes	Yes	48	5	Yes	Yes	No	Has 32-bit internal bus, 16-bit data bus; 8080-code−compatible
No	Yes	Yes	Yes	48	5	Yes	Yes	No	Has 32-bit internal bus, 16-bit data bus
No	Yes	Yes	Yes	40	5	Yes	Yes	Yes	Pin-compatible with the 8086
No	Yes	Yes	No	68	5	Yes	Yes	Yes	Has on-chip memory management and protection
Yes	Yes	Yes	Yes	40	5, 12, −5[l]	Yes	Yes	Yes	The 9980 requires external clock
Yes	Yes	Yes	Yes	40	5	Yes	Yes	Yes	Has 256 bytes of RAM on board to speed context switches

[g]Frame pointer, too.
[h]Has 8-bit external buses and 16-bit internal buses.
[i]The clock is internally divided by 4 or 6, depending on the instruction.
[j]Double-precision 16-bit operations are available.
[k]Range in bytes.
[l]9980 only.
[m]Not available.

Microprocessor Special: General-purpose chips

Word size, data/instruction (bits)	Original source	Processor	Process technology	Direct addressing range (words)	No. of basic instructions	Maximum clock frequency (MHz)/phases	Instruction time, shortest/longest[a] (μs)	TTL-compatible	BCD arithmetic	On-chip interrupts/levels	No. of internal general-purpose registers	No. of stack registers
8/8	Fujitsu	MBL6802	NMOS	64 k	72	2/1	2/5	Yes	Yes	Yes/1	128/0	RAM
8/8		MBL6809	NMOS	64 k	82	3.58/1	2/12	Yes	Yes	Yes/1	0	RAM
8/8	GTE Microcircuits	G65SC0X	CMOS	64 k	64	4/1	0.5/3.5	Yes	Yes	Yes/1	0	RAM
8/8		G65SC1X	CMOS	64 k	64	4/1	0.5/3.5	Yes	Yes	Yes/1	0	RAM
8/8		G65SC10X	CMOS	64 k	64	4/1	0.5/3.5	Yes	Yes	Yes/1	0	RAM
8/8		G65SC1XX	CMOS	64 k	64	4/1	0.5/3.5	Yes	Yes	Yes/1	0	RAM
8/8	Intel	8008	PMOS	16 k	48	0.8/2	12.5/37.5	No	Yes	Yes/1	6	7×14
8/8		8031	NMOS	128 k	111	12	1/4	Yes	Yes	Yes/2	128	RAM
8/8		8035/8039	NMOS	64 k	96	6/1	2.5/5	Yes	Yes	Yes/1	64	RAM
8/8		8080A	NMOS	64 k	78	2.6/2	1.5/3.75	Yes[c]	Yes	Yes/1	8	RAM
8/8		8085	NMOS	64 k	80	5.5/1	0.8/5.2	Yes	Yes	Yes/4	8	RAM
8/8	Matra-Harris	HM-8035H	NMOS	4 k	96	8	1.9/3.8	Yes	Yes	Yes/1	16	8
8/8		HM-80C35	CMOS	4 k	97	11	1.4/2.8	Yes	Yes	Yes/1	16	8
8/8		HM-8031	NMOS	64 k	111	12	1/4	Yes	Yes	Yes/2	32	RAM
8/8	Mitel	68SC02	CMOS	64 k	72	1/1	2/5	Yes	Yes	Yes/1	128	RAM
8/8	Mostek	MK38P70/02	NMOS	64 k	70+	4/1	2/13	Yes	Yes	Yes/2	64	RAM
8/8		MK38P73/02	NMOS	64 k	70+	4/1	2/13	Yes	Yes	Yes/4	64	RAM
8/8	Motorola	M6800	NMOS	64 k	72	2/2	1/2.5	Yes	Yes	Yes/1	0	RAM
8/8		MC6802/6808	NMOS	64 k	72	2/1	2/5	Yes	Yes	Yes/1	128/0	RAM
8/8		MC6803	NMOS	64 k	82	2/1	2/12	Yes	Yes	Yes/1	128	RAM
8/8		MC6803E	NMOS	64 k	82	2/1	2/5	Yes	Yes	Yes/1	0	RAM
8/8		M6809	NMOS	64 k	59	2/1	2/5	Yes	Yes	Yes/1	0	RAM
8/8		MC6809E	NMOS	64 k	59	2/1	2/12	Yes	Yes	Yes/1	128	RAM
8/8		MC680344	NMOS	64 k	72	1.25/1	2/5	Yes	Yes	Yes/1	128	RAM
8/8		MC146805E2	CMOS	8 k	61	5/1	2/4	Yes	Yes	Yes/1	0	RAM
8/8	National Semiconductor	INS8060	NMOS	64 k	46	4/1	5/22	Yes	Yes	Yes/1	8	RAM
8/8		INS8040	NMOS	64 k	96	11/1	1.4/2.8	Yes	Yes	Yes/1	256	8
8/8[f]		INS8070	NMOS	64 k	74	4/1	3/1000[e]	Yes	No	Yes/2	9	RAM
8/8		INS8073	NMOS	64 k	See Comments	4/1	3/1000	Yes	Yes	Yes/2	See Comments	RAM
8/8		NSC800	CMOS	64 k	150+	4/1	1/5.75	Yes	Yes	Yes/5	14	RAM
8/8	NEC Electronics	μPD7800	NMOS	64 k	140	2/1	2/4	Yes	Yes	Yes/5	128+16	RAM
8/8		μPD78C05	CMOS	64 k	101	4/1	4	Yes	Yes	Yes/3	128+16	RAM
8/8		μPD7810	NMOS	64 k	158	12/1	1/4	Yes	Yes	Yes/3	20	RAM
8/8		μPD80C35/80C39	CMOS	64 k	96	6/1	2.5/5	Yes	Yes	Yes/1	64	RAM
8/8	Oki	MSM80C35/39	CMOS	64 k	111	11/1	1.4/2.8	Yes	Yes	Yes/1	64	RAM
8/8		MSM80C85A	CMOS	64 k	113	6/1	1.3/6	Yes	Yes	Yes/4	8	RAM
8/8	RCA	CDP1802	CMOS	64 k	91	2.5/1	6.4/9.6	Yes	No	Yes/1	16x16	RAM
8/8		CDP1802A	CMOS	64 k	91	3.2/1	5/7.5	Yes	No	Yes/1	16x16	RAM
8/8		CDP1802B	CMOS	64 k	91	5/1	3.2/4.8	Yes	No	Yes/1	16x16	RAM
8/8		CDP1802M	CMOS	64 k	91	3.2/1	4/6	Yes	No	Yes/1	16x16	RAM
8/8		CDP1805	CMOS	64 k	113	4/1	4/6	Yes	No	Yes/1	16x16 +RAM	RAM
8/8		CDP1805A	CMOS	64 k	123	5/1	3.2/4.8	Yes	Yes	Yes/1	16x16 +RAM	RAM
8/8		CDP1806	CMOS	64 k	113	4/1	4/6	Yes	No	Yes/1	16x16	RAM
8/8		CDP1806A	CMOS	64 k	123	5/1	3.2/4.8	Yes	Yes	Yes/1	16x16	RAM
8/8		CDP6805E2	CMOS	8 k	61	5/1	2/4	Yes	Yes	Yes/1	RAM	RAM
8/8	Rockwell International	R650X	NMOS	64 k	56	4/1	0.5/3.5	Yes	Yes	Yes/1	0	RAM
8/8		R651X	NMOS	64 k	56	4/2	0.5/3.5	Yes	Yes	Yes/1	0	RAM
8/8		R65CXX	CMOS	64 k	68	4/2	0.5/3.5	Yes	Yes	Yes/1	0	RAM
8/8	Siemens	SAB 8031	NMOS	128 k	111	12/1	1/4	Yes	Yes	Yes/2	128	RAM
8/8		SAB 8035	NMOS	64 k	96	6/1	2.5/5	Yes	Yes	Yes/1	64	RAM
8/8		SAB8080A-1	NMOS	64 k	78	3 1/2	1.26/3.15	Yes[c]	Yes	Yes/1	8	RAM
8/8		SAB8085A-1	NMOS	64 k	80	5 5/1	0.8/5.2	Yes	Yes	Yes/4	8	RAM
8/8	Signetics	2650A	NMOS	32 k	75	2/1	1.5/6	Yes	Yes	Yes/1	7	8×15
8/8	Texas Instruments	TMS7000	NMOS	64 k	81	8/1	1.1/10.8	Yes	Yes	Yes/2	128×8	RAM

Q = quad-in-line package.
N.a. = not applicable.

[a] With the maximum clock.
[b] Standard TTL or MOS circuits will suffice.
[c] TTL output and CMOS input.
[d] Except clock lines.
[e] String search.
[f] At 5 V, as the supply voltage increases, clock frequency may increase.

On-chip clock	DMA capability	Specialized memory & I/O circuits available	Prototyping system available	Package size (no. of pins)	Voltages required (V)	Assembly-language development system	High-level languages	Time-sharing cross software	Comments
Yes	Yes	Yes	Yes	40	5	Yes	Yes	Yes	Compatible with the MC6802
Yes	Yes	Yes	Yes	40	5	Yes	Yes	Yes	Compatible with the MC6809
Yes	No	Yes	Yes	40	5	Yes	Yes	Yes	New chip, 15 addressing modes
Yes	No	Yes	Yes	40	5	Yes	Yes	Yes	New chip. Similar to 65SC0X
Yes	Yes	Yes	Yes	40	5	Yes	Yes	Yes	New chip. Memory lock and bus enable
Yes	Yes	Yes	Yes	40	5	Yes	Yes	Yes	New chip. Similar to G65SC10X
No	No	Yes	Yes	18	5, −9	Yes	Yes	Yes	Predecessor of 8080, still in wide use
Yes	No	Yes	Yes	40	5	Yes	No	Yes	ROM-less version of 8051 microcomputer
Yes	Yes	Yes	Yes	40	5	Yes	Yes	Yes	ROM-less versions of 8048/8049
No	Yes	Yes	Yes	40	5, 12, −5	Yes	Yes	Yes	By and large, still the most popular
Yes	Yes	Yes	yes	40	5	Yes	Yes	Yes	8080-code-compatible; has built-in clock
Yes	Yes	Yes	Yes	40	5	Yes	No	Yes	ROM-less version of 8048H
Yes	Yes	Yes	Yes	40	5	Yes	No	Yes	ROM-less version of 80C48
Yes	Yes	Yes	Yes	40	5	Yes	Yes	Yes	ROM-less version of 8051
Yes	Yes	Yes	Yes	40	3 to 6.5	Yes	Yes	Yes	CMOS version of 6802, has 1.5-V standby capability
Yes	No	Yes	Yes	40+	5	Yes	Yes	Yes	ROM-less piggyback package version of MK3870 microcomputer family
Yes	No	Yes	Yes	40+	5	Yes	Yes	Yes	ROM-less piggyback package version of MK3873 microcomputer family
No	Yes	Yes	Yes	40	5	Yes	Yes	Yes	Available in depletion-load version
Yes	Yes	Yes	Yes	40	5	Yes	Yes	Yes	6802 has 128×8 on-chip RAM; 6808 has no RAM
Yes	Yes	Yes	Yes	40	5	Yes	Yes	Yes	ROM-less version of 6801 microcomputer
Yes	Yes	Yes	Yes	40	5	Yes	Yes	Yes	Enhanced 6800 command set
Yes	Yes	Yes	Yes	40	5	Yes	Yes	Yes	External clock
Yes	Yes	Yes	Yes	40	5	Yes	Yes	Yes	External clock
Yes	Yes	Yes	Yes	40	5	Yes	Yes	Yes	External clock
RAM	Yes	No	Yes	40	4 to 6.5	Yes	Yes	—	ROM-less and CMOS version of the 6805 microcomputer
Yes	Yes	Yes	Yes	40	5	Yes	Yes	Yes	On-chip multiprocessing and DMA logic
Yes	No	Yes	Yes	40	5	Yes	Yes	Yes	ROM-less version of INS8050 microcomputer
Yes	Yes	Yes[b]	Yes	40	5	Yes	Yes	No	ROM-less version of INS8072; 64 bytes of on-chip cache RAM
Yes	Yes	Yes	No	40	5	Yes	Yes	No	On-chip ROM contains Tiny Basic interpreter; 74 assembly-level commands also available
Yes	Yes	Yes	Yes	40	5	Yes	Yes	Yes	Executes Z80 instructions and has 8085 bus structure
Yes	Yes	Yes	Yes	64Q	5	Yes	No	Yes	ROM-less version of μPD7801 microcomputer
Yes	No	Yes	Yes	64Q	5	Yes	No	Yes	ROM-less version of μPD78C06
Yes	No	Yes	Yes	64Q	5	Yes	No	Yes	ROM-less version is also available
Yes	Yes	Yes	Yes	40	5	Yes	Yes	Yes	ROM-less and CMOS version of μPD8048/8049
Yes	No	Yes	No	40	3 to 6	Yes	No	Yes	CMOS version of 8035/8039
Yes	Yes	Yes	Yes	40	4 to 6	No	Yes	No	New chip, CMOS version of 8085A
Yes	Yes	Yes	Yes	40	4 to 10.5	Yes	Yes	Yes	Register-based
Yes	Yes	Yes	Yes	40	4 to 10.5	Yes	Yes	Yes	Register-based
Yes	Yes	Yes	Yes	40	4 to 10.5	Yes	Yes	Yes	High-speed version of CDP1802A
Yes	Yes	Yes	Yes	40	3 to 6.5	Yes	Yes	Yes	Low-voltage version of CDP1802A
Yes	Yes	Yes	Yes	40	4 to 6.5	Yes	Yes	Yes	Has 64 bytes of RAM, 8-bit timer-counter, enhanced instruction set
Yes	Yes	Yes	Yes	40	4 to 6.5	Yes	Yes	Yes	ROM-less 1804A; has 64 bytes of RAM on chip, 8-bit timer-counter
Yes	Yes	Yes	Yes	40	4 to 6.5	Yes	Yes	Yes	Has 8-bit timer-counter, enhanced instruction set
Yes	Yes	Yes	Yes	40	4 to 6.5	Yes	Yes	Yes	ROM-less 1804A; has 8-bit timer-counter
Yes	No	Yes	Yes	40	4 to 6.5	Yes	Yes	—	
Yes	No	Yes	Yes	40	5	Yes	Yes	Yes	Provides 13 addressing modes
No	No	Yes	Yes	40	5	Yes	Yes	Yes	Similar to 650x but needs two-phase clock
Yes	No	Yes	Yes	40	5	Yes	Yes	Yes	Three clock options
Yes	No	Yes	Yes	40	5	Yes	No	Yes	ROM-less version of 8051 all-in-one processor
Yes	No	Yes	No	40	5	Yes	No	Yes	ROM-less version of 8048 all-in-one processor
No	Yes	Yes	Yes	40	5, 12, −5	Yes	Yes	Yes	Pin-compatible with the 8080A
Yes	Yes	Yes	Yes	40	5	Yes	Yes	Yes	Pin-compatible with the 8080A
No	Yes	Yes	Yes	40	5	Yes	Yes	Yes	1.25- and 2-MHz versions
Yes	No	Yes	Yes	40	5	Yes	Yes	Yes	ROM-less microcomputer. Also available are the 7020 and 7040, with 2048 and 4096 bytes, respectively, of on-chip ROM. All have timer and two 8-bit ports

[g]Frame pointer, too.
[h]Has 8-bit external buses and 16-bit internal buses.
[i]The clock is internally divided by 4 or 6, depending on the instruction.
[j]Double-precision 16-bit operations are available.
[k]Range in bytes.
[l]9980 only.

Microprocessor Special: General-purpose chips

Word size, data/instruction (bits)	Original source	Processor	Process technology	Direct addressing range (words)	No. of basic instructions	Maximum clock frequency (MHz)/phases	Instruction time, shortest/longest[a] (μs)	TTL-compatible	BCD arithmetic	On-chip interrupts/levels	No. of internal general-purpose registers	No. of stack registers
16/16	Texas Instruments	TMS/SBP9900	NMOS/I²L	32 k	69	4/4	2/31	Yes[b]	No	Yes/16	16	RAM
16/16		SBP9900	I²L	128 k	73	4.4/1	N.a.	Yes	No	Yes/16	16	RAM
16/16		SBP9989	I²L	32 k	73	4.4/1	1.37/13.6	Yes	Yes	Yes/16	0	RAM
16/16		TMS99105	NMOS	128 k	85	24/1	0.5/7.2	Yes	No	Yes/16	16	RAM
16/16		TMS99110	NMOS	128 k	98	24/1	0.5/7.2	Yes	No	Yes/16	16	RAM
16/16		TMS99120	NMOS	128 k	87	24/1	0.5/7.2	Yes	No	Yes/16	16	RAM
16/16	Western Digital	WD-16	NMOS	64 k	116	3.3/4	2.1/760	Yes	Yes	Yes/16	6	RAM
16/16		Pascal Microengine	NMOS	64 k	150+	3/4	2.4/300[e]	Yes	Yes	Yes/4	RAM	RAM
16/16	Zilog	Z8001	NMOS	48 M[k]	110+	10/1	0.30/N.a.	Yes	Yes	Yes/1	17	RAM
16/16		Z8002	NMOS	354 k	110+	10/1	0.30/N.a.	Yes	Yes	Yes/1	17	RAM
16/16		Z8003	NMOS	48 M	110+	10/1	0.30/N.a.	Yes	Yes	Yes/1	17	RAM
16/16		Z8004	NMOS	354 k	110+	10/1	0.30/N.a.	Yes	Yes	Yes/1	17	RAM
16/32	Digital Equipment	DCJ11	NMOS	2 M	96	20/1	0.2/N.a.	Yes	No	Yes/4	12	3
32/32	Intel	iAPX 432	NMOS	16 M	221	8/2	1.25/200	Yes	No	No	0	0
32/32	National Semiconductor	NS32032	NMOS	16 M[k]	100+	N.a.	N.a.	Yes	Yes	Yes	8	RAM
80/16	Intel	i8087	NMOS	1 M	68	5	9/100	Yes	Yes	Yes/1	8×80	8×80

Q = quad-in-line package.
N.a. = not available.

[a]With the maximum clock.
[b]Standard TTL or MOS circuits will suffice.
[c]TTL output and CMOS input.
[d]Except clock lines.
[e]String search.
[f]At 5 V; as the supply voltage increases, clock frequency may increase.

On-chip clock	DMA capability	Specialized memory & I/O circuits available	Prototyping system available	Package size (no. of pins)	Voltages required (V)	Assembly-language development system	High-level languages	Time-sharing cross software	Comments
No	Yes	Yes	Yes	64	5, 12, −5	Yes	Yes	Yes	Emulates 990 minicomputer's instructions
No	Yes	Yes	No	64	5, 12, −5	Yes	Yes	Yes	Enhanced military version of TMS9900
No	Yes	Yes	Yes	64 or 68	1.25	Yes	Yes	Yes	Executes enhanced set of SBP9900's instructions; 30% faster
Yes	Yes	Yes	No	40	5	Yes	Yes	Yes	Has attached processor interface and macrostore address space. Instruction set is compatible with TMS9900
Yes	Yes	Yes	No	40	5	Yes	Yes	Yes	Same as 99105 except has single-precision floating-point arithmetic capability
Yes	Yes	Yes	No	40	5	Yes	Yes	Yes	Same as 99105, plus Pascal support functions
No	Yes	Yes	Yes	40	5, 12, −5	Yes	Yes	No	Very similar to DEC LSI-11
No	Yes	Yes	Yes	40	±12, ±5	Yes	Yes	Yes	Five-chip set directly executes Pascal p-code
No	Yes	Yes	Yes	48	5	Yes	Yes	Yes	Address space is divided into segments
No	Yes	Yes	Yes	40	5	Yes	Yes	Yes	Nonsegmented version
No	Yes	Yes	Yes	40	5	Yes	Yes	Yes	8001 with abort control for virtual memory
No	Yes	Yes	Yes	40	5	Yes	Yes	Yes	8002 with abort control for virtual memory
Yes	Yes	Yes	No	60	5	Yes	Yes	Yes	New chips. Two-chip hybrid microprocessor implementation of the PDP-11/70 minicomputer; executes entire PDP-11/70 superset of instructions; compatible with all PDP-11 software.
No	Yes	Yes	Yes	2×64	5	No	Yes	Yes	A 32-bit microprocessor consisting of two VLSI chips: the 43201 and 43202
No	Yes	Yes	Yes	48	5	Yes	Yes	No	Has 32-bit internal and external data buses
No	Yes	Yes	Yes	40	5	Yes	Yes	Yes	Numeric processor, does IEEE floating-point calculations, as well as trig, log, and exponential operations; 100 times faster than software floating-point calculations

[g]Frame pointer, too.
[h]Has 8-bit external buses and 16-bit internal buses.
[i]The clock is internally divided by 4 or 6, depending on the instruction.
[j]Double-precision 16-bit operations are available.
[k]Range in bytes.
[l]9980 only.
[m]Not available.

Single-chip microcomputers: Performance and features

Word size, data/instruction (bits)	Manufacturer	Device	Process technology	On-chip RAM size (bits)	On-chip ROM/PROM size (bits)	Off-chip memory expansion	No. of basic instructions	Maximum clock frequency (kHz)	On-chip clock	Instruction time, shortest/longest (μs)	TTL-compatible	BCD arithmetic	On-chip interrupts/levels	No. of [illegible]
4/8	Fujitsu	MB8841	NMOS	128×4	2048×8	No	70	2000	Yes	8/6	Yes	Yes	Yes 2	4
4/8		MB8844	NMOS	128×4	2048×8	No	70	2000	Yes	8/6	Yes	Yes	Yes/2	4
4/8		MB8851	CMOS	128×4	2048×8	No	70	2000	Yes	2/4	Yes	Yes	Yes/2	4
4/8		MB8853	CMOS	64×4	1024×8	No	70	2000	Yes	2/4	Yes	Yes	Yes/2	4
4/8		MB88401	NMOS	192×4	4096×8	No	75	2000	Yes	8/6	Yes	Yes	Yes/4	8
4/8		MB88411	NMOS	192×4	4096×8	No	75	2000	Yes	8/6	Yes	Yes	Yes/4	8
4/8		MB88413	NMOS	192×4	2048×8	No	75	2000	Yes	8/6	Yes	Yes	Yes/4	8
4/8		MB88500	CMOS	192×4	4096×8	No	75	2000	Yes	8/6	Yes	Yes	Yes/4	8
4/8		MB88504	CMOS	192×4	4096×8	No	75	2000	Yes	8 6	Yes	Yes	Yes/4	8
4/8	ITT Semiconductors	SAA6000	CMOS	96×4	2268×8	Yes	54	32	Yes	61/122	No	No	No	1
4/8	Motorola	MC141000	CMOS	64×4	1024×8	No	43	600	Yes	10/10	Yes	Yes	No	1
4/8		COP320C	CMOS	64×4	1024×8	Yes	49	500	Yes	15/245	Yes	Yes	No	3
4/8		COP321C	CMOS	64×4	1024×8	Yes	49	500	Yes	15/245	Yes	Yes	No	3
4/8	National Semiconductor	COP410L	NMOS	32×4	512×8	Yes	40	260	Yes	15/40	Yes	Yes	No	2
4/8		COP411L	NMOS	32×4	512×8	Yes	40	260	Yes	15/40	Yes	Yes	No	2
4/8		COP420	NMOS	64×4	1024×8	Yes	49	2000	Yes	4/8	Yes	Yes	Yes/1	3
4/8		COP420C	CMOS	64×4	1024×8	Yes	49	500	Yes	15/245	Yes	Yes	Yes/1	3
4/8		COP420L	NMOS	64×4	1024×8	Yes	49	260	Yes	15/40	Yes	Yes	Yes/1	3
4/8		COP421	NMOS	64×4	1024×8	Yes	49	2000	Yes	4/10	Yes	Yes	No	3
4/8		COP421C	CMOS	64×4	1024×8	Yes	49	500	Yes	15/245	Yes	Yes	No	3
4/8		COP421L	NMOS	64×4	1024×8	Yes	49	260	Yes	15/40	Yes	Yes	No	3
4/8		COP440	NMOS	160×4	2048×8	Yes	49	1000	Yes	4/10	Yes	Yes	Yes/4	3
4/8		COP444L	NMOS	128×4	2048×8	Yes	49	260	Yes	15/40	Yes	Yes	Yes/1	3
4/8		COP445L	NMOS	128×4	2048×8	Yes	49	260	Yes	15/40	Yes	Yes	Yes/1	3
4/8		COP2440	NMOS	160×4	2048×8	Yes	60	1000	Yes	4/10	Yes	Yes	Yes/4	4
4/8		COP2441	NMOS	160×4	2048×8	Yes	60	1000	Yes	4/10	Yes	Yes	Yes/4	4
4/8		COP2442	NMOS	160×4	2048×8	Yes	60	1000	Yes	4/10	Yes	Yes	Yes/4	4
4/8	NEC Electronics	μPD546	PMOS	96×4	2000×8	No	80	440	Yes	4.5/9	Yes	Yes	Yes/1	3
4/8		μPD547	PMOS	64×4	1000×8	Yes	58	440	Yes	4.5/9	Yes	Yes	Yes/1	3
4/8		μPD547L	PMOS	64×4	1000×8	Yes	58	180	Yes	11/22	Yes	Yes	Yes/1	3
4/8		μPD550	PMOS	32×4	640×8	No	58	440	Yes	4 5/9	Yes	Yes	Yes/1	3
4/8		μPD550L	PMOS	32×4	640×8	No	58	180	Yes	11/22	Yes	Yes	Yes/1	3
4/8		μPD552	PMOS	64×4	1000×8	No	58	440	Yes	4 5/9	Yes	Yes	Yes/1	3
4/8		μPD553	PMOS	96×4	2000×8	No	80	440	Yes	4 5/9	Yes	Yes	Yes/1	3
4/8		μPD554	PMOS	32×4	1000×8	No	58	440	Yes	4.5/9	Yes	Yes	Yes/1	3
4/8		μPD554L	PMOS	32×4	1000×8	No	58	180	Yes	11/22	Yes	Yes	Yes/1	3
4/8		μPD557L	PMOS	96×4	2000×8	No	80	180	Yes	11/22	Yes	Yes	Yes/1	3
4/8		μPD650	CMOS	96×4	2000×8	No	80	440	Yes	4 5/9	Yes	Yes	Yes/1	3
4/8		μPD651	CMOS	64×4	1000×8	No	58	440	Yes	4 5 9	Yes	Yes	Yes/1	3
4/8		μPD652	CMOS	32×4	1000×8	No	58	440	Yes	4 5 9	Yes	Yes	Yes/1	3
4/8		μPD7501	CMOS	96×4	1024×8	No	92	200	Yes	6 7/20	Yes	Yes	Yes 2	3
4/8		μPD7502	CMOS	128×4	2048×8	No	92	200	Yes	6.7/20	Yes	Yes	Yes 2	3
4/8		μPD7503	CMOS	224×4	4096×8	No	92	200	Yes	6 7/20	Yes	Yes	Yes 2	3
4/8		μPD7506	CMOS	64×4	1024×8	No	92	200	Yes	6 7/20	Yes	Yes	Yes 2	3
4/8		μPD7507	CMOS	128×4	2048×8	No	92	200	Yes	6 7/20	Yes	Yes	Yes 2	3
4/8		μPD7508	CMOS	224×4	4096×8	No	92	200	Yes	6 7/20	Yes	Yes	Yes 2	3
4/8		μPD7519	CMOS	256×4	4096×8	No	106	4019	Yes	7 6 N a	Yes	Yes	Yes 4	8
4/8		μPD7520	PMOS	48×4	768×8	No	47	300	Yes	20 40	Yes	No	No	1
4/8		μPD7528	CMOS	160×4	4096×8	No	86	500	Yes	4 N a	Yes	Yes	Yes/3	8
4/8	Oki	MSM5840RS	CMOS	128×4	2048×8	Yes	98	4200	Yes	7 6/15 2	Yes	Yes	Yes 2	4
4/8		MSM58421GS	CMOS	40×4	1536×8	No	52	4200	Yes	7 6/15 2	Yes	Yes	No	1
4/8		MSM58422GS	CMOS	40×4	1536×8	No	52	4200	Yes	7.6/15.2	Yes	Yes	No	1

N a = not available FP = flat pack

This table summarizes both existing single-chip microcomputers and those that have been introduced since last year's update (ELECTRONIC DESIGN, Nov. 26, 1981, p. 122). Special features are flagged under "Comments." The data pages, which begin on p. 147 of this issue, offer more information about the new entries. The microcomputers are arranged first by word size and then alphabetically by manufacturer. For more information from the makers, circle the appropriate reader service numbers given in the table starting on p. 157.

General-purpose internal registers	No. of I/O lines	Additional special support circuits	Package size (no. of DIP pins)	Voltages required (V)	Prototyping system available	Assembly-language programming system	High-level-language programming system	Time-sharing cross software	Comments
RAM	32	No	42	5	Yes	Yes	No	No	Has 2 kbytes of ROM
RAM	23	No	28	5	Yes	Yes	No	No	Has 2 kbytes of ROM
RAM	37	No	42	5	Yes	Yes	Yes	Yes	Has 8-bit counter-timer and serial I/O port
RAM	37	No	42	5	Yes	Yes	Yes	Yes	Version of 8851 with smaller RAM and ROM
RAM	36	No	42	5	Yes	Yes	No	No	Has 4 kbytes of ROM, upwardly compatible with 8841 family
RAM	33	No	42	5	Yes	Yes	No	No	Has a-d converter, 4 kbytes of ROM
RAM	33	No	42	5	Yes	Yes	No	No	Has a-d converter, 2 kbytes of ROM
RAM	36	No	42	5	Yes	Yes	No	No	Upwardly compatible with 8851 family; has 4 kbytes of ROM
RAM	36	No	42	5	Yes	Yes	No	No	Has 4 kbits of ROM, LCD driver
RAM	51	Yes	60	3	Yes	Yes	No	Yes	New chip, LCD driver dissipates 45 μW on standby 135 μW when active
2+RAM	23	Yes	28	3 to 6	Yes	Yes	No	Yes	Exact CMOS equivalent of TMS1000 not the enhanced TMS1000C, MC141200 also available
RAM	23	Yes	28	2.4 to 6	Yes	Yes	No	Yes	Extended temperature version (−40° to +85°C) of 420C
RAM	19	Yes	24	2.4 to 6	Yes	Yes	No	Yes	Extended temperature version (−40° to +88°C) of 421C
RAM	19	Yes	24	4.5 to 9.5	Yes	Yes	No	Yes	All COP processors include serial I/O and event-counting capability. Major differences between models include the I/O arrangements: input only, bidirectional, output only, etc. I/O options include LED direct segment drive, LED direct digit drive, three-state push-pull, push-pull, open drain, and standard (active device to ground and a pull-up to V_{CC})
RAM	16	Yes	20	4.5 to 9.5	Yes	Yes	No	Yes	
RAM	23	Yes	28	4.5 to 6.3	Yes	Yes	No	Yes	
RAM	23	Yes	28	2.4 to 6.3	Yes	Yes	No	Yes	
RAM	23	Yes	28	4.5 to 9.5	Yes	Yes	No	Yes	
RAM	19	Yes	24	4.5 to 6.3	Yes	Yes	No	Yes	
RAM	19	Yes	24	2.4 to 6.3	Yes	Yes	No	Yes	
RAM	19	Yes	24	4.5 to 9.5	Yes	Yes	No	Yes	
RAM	35	Yes	40	4.5 to 6.3	Yes	Yes	No	Yes	
RAM	23	Yes	28	4.5 to 9.5	Yes	Yes	No	Yes	
RAM	19	Yes	24	4.5 to 9	Yes	Yes	No	Yes	Reduced I/O version of COP444L; narrower voltage range (4.5 to 6.3 V) also available
RAM	35	Yes	40	4.5 to 6.3	Yes	Yes	No	Yes	Each chip has two CPUs that share RAM and ROM. On the chips are a programmable counter, zero-crossing detector, and serial I/O port
RAM	23	Yes	28	4.5 to 6.3	Yes	Yes	No	Yes	
RAM	19	Yes	24	4.5 to 6.3	Yes	Yes	No	Yes	
RAM	35	No	42	−10	Yes	Yes	No	Yes	Has TTL-compatible I/O lines
RAM	35	No	42	−10	Yes	Yes	No	Yes	Has TTL-compatible I/O lines
RAM	35	No	42	−8	Yes	Yes	No	Yes	Low-power version of 547 (half the current)
RAM	21	No	28	−10	Yes	Yes	No	Yes	I/O handles −35-V vacuum fluorescent drive
RAM	21	No	28	−8	Yes	Yes	No	Yes	Low-power version of 550 (half the current)
RAM	35	No	42	−10	Yes	Yes	No	Yes	I/O handles −35-V vacuum fluorescent drive
RAM	35	No	42	−10	Yes	Yes	No	Yes	I/O handles −35-V vacuum fluorescent drive
RAM	21	No	28	−10	Yes	Yes	No	Yes	I/O handles −35-V vacuum fluorescent drive
RAM	21	No	28	−8	Yes	Yes	No	Yes	Low-power version of 554 (half the current consumption)
RAM	21	No	28	−8	Yes	Yes	No	Yes	Reduced I/O and low-power version of 553
RAM	35	No	42	5	Yes	Yes	No	Yes	CMOS version of 546 (4% of the power dissipation)
RAM	35	No	42	+5	Yes	Yes	No	Yes	CMOS version of 547 (4% of the power dissipation)
RAM	21	No	42	+5	Yes	Yes	No	Yes	CMOS version of 550 (5% of the power dissipation)
4+RAM	24	Yes	64FP	5	Yes	Yes	No	Yes	All have on-chip controller-drivers for LCD
4+RAM	23	Yes	64FP	5	Yes	Yes	No	Yes	
4+RAM	23	Yes	64FP	5	Yes	Yes	No	Yes	
4+RAM	22	Yes	28	5	Yes	Yes	No	Yes	
4+RAM	32	Yes	40	5	Yes	Yes	No	Yes	Do not include LCD controller-drivers
4+RAM	32	Yes	40	5	Yes	Yes	No	Yes	
6+RAM	58	Yes	64Q	5, −35	Yes	Yes	No	No	Has programmable vacuum fluorescent driver
RAM	24	Yes	28	−6 to −10	Yes	Yes	No	Yes	Output ports designed for LED driving; built-in programmable display controller
+RAM	35	Yes	42	5	Yes	Yes	No	No	Outputs can drive −35-V vacuum fluorescent displays
+RAM	30	Yes	42	3 to 6	Yes	Yes	No	Yes	Includes 8-bit timer-counter; ROM-less evaluation chip available
+RAM	53	Yes	60FP	3 to 6	Yes	Yes	No	Yes	Includes 12-bit timer-counter and LCD direct-drive outputs
+RAM	53	Yes	60FP	3 to 6	Yes	Yes	No	Yes	Includes 12-bit timer-counter and LCD direct-drive outputs

Microprocessor Special: Single-chip μCs

Word size, data/instruction (bits)	Manufacturer	Device	Process technology	On-chip RAM size (bits)	On-chip ROM/PROM size (bits)	Off-chip memory expansion	No. of basic instructions	Maximum clock frequency (kHz)	On-chip clock	Instruction time, shortest/longest (μs)	TTL-compatible	BCD arithmetic	On-chip interrupts/levels	No. of subroutine
4/8	Oki	MSM5842RS	CMOS	32×4	768×8	No	52	4200	Yes	7.6/15.2	Yes	Yes	No	1
4/8		MSM5845RS	CMOS	64×4	1280×8	No	49	4200	Yes	7.6/15.2	Yes	Yes	Yes/1	2
4/8		MSM6404RS	CMOS	256×4	4096×8	No	91	2000	Yes	2/N.a	Yes	Yes	Yes/5	1
4/8	Panasonic (Matsushita)	MN1400	NMOS	64×4	1024×8	No	75	300	Yes	10/20	Yes	Yes	Yes/1	2
4/8		MN1402	NMOS	32×4	768×8	No	57	300	Yes	10/20	Yes	Yes	Yes/1	2
4/8		MN1403	NMOS	16×4	512×8	No	50	300	Yes	10/20	Yes	No	Yes/1	2
4/8		MN1404	NMOS	16×4	512×8	No	48	300	Yes	10/20	Yes	Yes	Yes/1	2
4/8		MN1405	NMOS	128×4	2048×8	No	75	300	Yes	10/20	Yes	Yes	Yes/2	2
4/8		MN1430	PMOS	64×4	1024×8	No	75	200	Yes	15/30	No	Yes	Yes/1	2
4/8		MN1432	PMOS	32×4	768×8	No	57	200	Yes	15/30	No	Yes	Yes/1	2
4/8		MN1435	PMOS	128×4	2048×8	No	75	200	Yes	15/30	No	Yes	Yes/2	2
4/8		MN1450	CMOS	64×4	1024×8	No	75	500	Yes	10/20	Yes	Yes	Yes/1	2
4/8		MN1453	CMOS	16×4	512×8	No	50	500	Yes	10/20	Yes	Yes	Yes/1	2
4/8		MN1454	CMOS	16×4	512×8	No	48	500	Yes	10/20	Yes	Yes	Yes/1	2
4/8		MN1455	CMOS	128×4	2048×8	No	75	500	Yes	10/20	Yes	Yes	Yes/2	2
4/8		MN1541	NMOS	152×4	1536×8	Yes	124	1000	Yes	2/4	Yes	Yes	Yes/4	2
4/8		MN1542	NMOS	152×4	2048×8	Yes	124	1000	Yes	2/4	Yes	Yes	Yes/4	16
4/8		MN1544	NMOS	256×4	4096×8	Yes	124	1000	Yes	2/4	Yes	Yes	Yes/4	16
4/8		MN1562	NMOS	152×4	2048×8	Yes	124	1000	Yes	2/4	Yes	Yes	Yes/4	16
4/8		MN1564	NMOS	256×4	4096×8	Yes	124	1000	Yes	2/4	Yes	Yes	Yes/4	16
4/8		MM76EL	PMOS	48×4	640×8	RAM only	50	100/4	Yes	10/30	Yes	Yes	Yes/1	1
4/8		MM75	PMOS	48×4	670×8	RAM only	50	100/4	Yes	10/40	Yes	Yes	Yes/1	1
4/8	Rockwell	MM78/MM78L	PMOS	128×4	2048×8	RAM only	50	100/4	Yes	10/40	Yes	Yes	Yes/1	2
4/8		MM78LA	PMOS	128×4	2048×8	RAM only	50	100/4	Yes	10/40	Yes	Yes	Yes/1	2
4/8	Texas Instruments	TMS1000	PMOS	64×4	1024×8	No	43	400	Yes	15/60	No	Yes	No	1
4/8	Texas Instruments	TMS1000C	CMOS	64×4	1024×8	No	43	1000	Yes	6/120	Yes	Yes	No	3
4/8		TMS1004C	CMOS	256×4	1024×8	No	40	1000	Yes	6/120	Yes	Yes	No	3
4/8		TMS1018	PMOS	64×4	1024×8	No	43	400	Yes	15/60	Yes	Yes	No	N.a
4/8		TMS1022	PMOS	64×4	1024×8	No	43	400	Yes	15/60	Yes	Yes	No	N.a
4/8		TMS1070	PMOS	64×4	1024×8	No	43	400	Yes	15/60	No	Yes	No	1
4/8		TMS1070C	CMOS	64×4	1024×8	No	43	1000	Yes	6/120	Yes	Yes	No	3
4/8		TMS1100	PMOS	128×4	2048×8	No	40	400	Yes	15/60	No	Yes	No	1
4/8		TMS1100C	CMOS	128×4	2048×8	No	40	1000	Yes	6/120	Yes	Yes	No	3
4/8		TMS1117	PMOS	128×4	2048×8	No	43	400	Yes	15/60	Yes	Yes	No	N.a
4/8		TMS1121	PMOS	128×4	2048×8	No	42	400	Yes	15/60	Yes	Yes	No	N.a
4/8		TMS1170	PMOS	128×4	2048×8	No	40	400	Yes	15/60	No	Yes	No	1
4/8		TMS1200	PMOS	64×4	1024×8	No	43	400	Yes	15/60	No	Yes	No	1
4/8		TMS1200C	CMOS	64×4	1024×8	No	43	1000	Yes	6/120	Yes	Yes	No	3
4/8		TMS1204C	CMOS	256×4	1024×8	No	40	1000	Yes	6/120	Yes	Yes	No	3
4/8		TMS1270	PMOS	64×4	1024×8	No	43	400	Yes	15/60	No	Yes	No	1
4/8		TMS1270C	CMOS	64×4	1024×8	No	43	1000	Yes	6/120	Yes	Yes	No	3
4/8		TMS1300	PMOS	128×4	2048×8	No	40	400	Yes	15/60	No	Yes	No	1
4/8		TMS1300C	CMOS	128×4	2048×8	No	40	1000	Yes	6/120	Yes	Yes	No	3
4/8		TMS1370	PMOS	128×4	2048×8	No	40	400	Yes	15/60	No	Yes	No	1
4/8		TMS1400	PMOS	128×4	4096×8	No	41	550	Yes	11/60	No	Yes	No	3
4/8		TMS1470	PMOS	128×4	4096×8	No	41	550	Yes	11/60	No	Yes	No	3
4/8		TMS1600	PMOS	64×4	4096×8	No	41	550	Yes	11/60	No	Yes	No	3
4/8		TMS1670	PMOS	64×4	4096×8	No	41	550	Yes	11/60	No	Yes	No	3
4/8		TMS1700	PMOS	64×4	4096×8	No	43	400	Yes	15/60	No	Yes	No	3
4/8		TMS2100	PMOS	128×4	2048×8	No	55	550	Yes	11/60	No	Yes	Yes/1	4
4/8		TMS2170	PMOS	128×4	2048×8	No	55	550	Yes	11/60	No	Yes	Yes/1	4
4/8		TMS2300	PMOS	128×4	2048×8	No	55	550	Yes	11/60	No	Yes	Yes/1	4
4/8		TMS2370	PMOS	128×4	2048×8	No	55	550	Yes	11/60	No	Yes	Yes/1	4
4/8		TMS2400	PMOS	256×4	4096×8	No	55	550	Yes	11/60	No	Yes	Yes/1	4
4/8		TMS2470	PMOS	256×4	4096×8	No	55	550	Yes	11/60	No	Yes	Yes/1	4
4/8		TMS2600	PMOS	256×4	4096×8	No	55	550	Yes	11/60	No	Yes	Yes/1	4
4/8		TMS2670	PMOS	256×4	4096×8	No	55	550	Yes	11/60	No	Yes	Yes/1	4
4/8		TMS3132	PMOS	128×4	2048×8	No	53	500	Yes	12/60	No	Yes	Yes/1	3
4/8		TMS3240	NMOS	128×4	2048×8	No	73	5000	Yes	1.2/6	Yes	Yes	No	5
4/8	Toshiba	TMP4310AP	NMOS	48×4	1024×8	No	35	500	Yes	4/8	Yes	No	Yes/1	4

N.a = not applicable
[a]TTL output but CMOS input. [b]Program. [c]Patterns

General-purpose internal registers	No. of I/O lines	Additional special support circuits	Package size (no. of DIP pins)	Voltages required (V)	Prototyping system available	Assembly-language programming system	High-level-language programming system	Time-sharing cross software	Comments
1+RAM	21	Yes	28	3 to 6	Yes	Yes	No	Yes	Includes 8-bit timer-counter
1+RAM	30	Yes	42	3 to 6	Yes	Yes	No	Yes	Includes 8-bit timer-counter
RAM	32	Yes	42	4.5 to 5.5	Yes	Yes	No	Yes	Includes an 8-bit and a 12-bit timer-counter
2+RAM	34	No	40	5	No	Yes	Yes	Yes	Complete all-in-one controller
RAM	19	No	28	5	No	Yes	Yes	Yes	Smaller I/O version of 1400
RAM	13	Yes	18	5	Yes	Yes	No	Yes	
RAM	10	Yes	16	5	Yes	Yes	No	Yes	
2+RAM	34	No	40	5	Yes	Yes	No	Yes	
2+RAM	34	No	40	−15	Yes	Yes	No	Yes	
RAM	19	No	28	−15	Yes	Yes	No	Yes	
2+RAM	34	No	40	−15	Yes	Yes	No	Yes	
2+RAM	34	Yes	40	4.25 to 6	Yes	Yes	No	Yes	
RAM	13	Yes	18	4.25 to 6	Yes	Yes	No	Yes	
RAM	10	Yes	16	4.25 to 6	Yes	Yes	No	Yes	
2+RAM	34	Yes	40	4.25 to 6	Yes	Yes	No	Yes	CMOS version of 1405/1435
4+RAM	24	Yes	40	5	Yes	Yes	No	Yes	
4+RAM	24	Yes	40	5	Yes	Yes	No	Yes	
4+RAM	24	Yes	40	5	Yes	Yes	No	Yes	
4+RAM	48	Yes	64	5	Yes	Yes	No	Yes	
4+RAM	48	Yes	64	5	Yes	Yes	No	Yes	
1+RAM	31	Yes	42	−15/+5, −10	Yes	Yes	No	Yes	Expanded ROM version
1+RAM	22	Yes	28	−15/+5, −10	Yes	Yes	No	Yes	Reduced I/O version of MM76EL
2+RAM	31	Yes	42	−15/+5, −10	Yes	Yes	No	Yes	
2+RAM	31	Yes	42	−15/+5, −10	Yes	Yes	No	Yes	Similar to MM78L but can drive LED or 14-segment vacuum fluorescent displays; offers tone generator and push-pull speaker drive outputs
2+RAM	23	Yes	28	−9 or −15	Yes	Yes	No	Yes	Two versions available, one for −9-V operation the other for −15-V operation
2+RAM	22	Yes	28	3 to 6	Yes	Yes	No	Yes	Enhanced CMOS version of TMS1000
2+RAM	22	Yes	28	3 to 6	Yes	Yes	No	Yes	Expanded RAM version of TMS1000C
N.a.	N.a.	N.a.	28	−15	N.a.	N.a.	N.a.	N.a.	Dedicated number cruncher
N.a.	N.a.	N.a.	28	−15	N.a.	N.a.	N.a.	N.a.	Dedicated CB PLL controller
2+RAM	23	Yes	28	−9 or −15	Yes	Yes	No	Yes	Vacuum fluorescent drive (−35 V) capability on outputs, otherwise same as TMS1000
2+RAM	22	Yes	28	3 to 6	Yes	Yes	No	Yes	Vacuum fluorescent drive (−35 V) capability on outputs, otherwise same as TMS1000
2+RAM	23	Yes	28	−9 or −15	Yes	Yes	No	Yes	Double memory version of TMS1000
2+RAM	22	Yes	28	3 to 6	Yes	Yes	No	Yes	CMOS version of TMS1100
N.a.	N.a.	N.a.	28	−15	N.a.	N.a.	N.a.	N.a.	Dedicated microwave oven controller
N.a.	N.a.	N.a.	40	−9 or 15	N.a.	N.a.	N.a.	N.a.	Dedicated appliance timer-controller
2+RAM	23	Yes	28	−9 or −15	Yes	Yes	No	Yes	Vacuum fluorescent drive version of TMS1100
2+RAM	25	Yes	40	−9 or −15	Yes	Yes	No	Yes	Expanded I/O version of TMS1000
2+RAM	32	Yes	40	3 to 6	Yes	Yes	No	Yes	Expanded I/O version of TMS1000C
2+RAM	32	Yes	40	3 to 6	Yes	Yes	No	Yes	Expanded I/O version of TMS1004C
2+RAM	27	Yes	40	−9 or −15	Yes	Yes	No	Yes	Expanded I/O version of TMS1070 with vacuum fluorescent drive
2+RAM	32	Yes	40	3 to 6	Yes	Yes	No	Yes	Expanded I/O version of TMS1070C with vacuum fluorescent drive
2+RAM	25	Yes	40	−9 or −15	Yes	Yes	No	Yes	Expanded I/O version of TMS1100
2+RAM	28	Yes	40	3 to 6	Yes	Yes	No	Yes	Expanded I/O version of TMS1100C
2+RAM	27	Yes	40	−9 or −15	Yes	Yes	No	Yes	Expanded I/O version of TMS1170 with vacuum fluorescent drive
2+RAM	23	Yes	28	−9 or −15	Yes	Yes	No	Yes	Expanded memory version of TMS1100
2+RAM	22	Yes	28	−9 or −15	Yes	Yes	No	Yes	Vacuum fluorescent drive version of TMS1400
2+RAM	32	Yes	40	−9 or −15	Yes	Yes	No	Yes	Enhanced I/O version of TMS1400
2+RAM	32	Yes	40	−9 or −15	Yes	Yes	No	Yes	Vacuum fluorescent drive version of TMS1600
2+RAM	21	Yes	28	−9 or −15	Yes	Yes	No	Yes	Reduced ROM and I/O version of TMS1000
2+RAM	20	Yes	28	−9	Yes	Yes	No	Yes	Includes 8-bit a-d converter, interval timer, zero-crossing detector; handles up to −15 V on outputs
2+RAM	19	Yes	28	−9	Yes	Yes	No	Yes	Same as 2100 but one less I/O line and handles up to −35 V on outputs
2+RAM	33	Yes	40	−9	Yes	Yes	No	Yes	Expanded 2100 with timer-event counter and two a-d input channels
2+RAM	32	Yes	40	−9	Yes	Yes	No	Yes	Same as 2300 but one less I/O line and handles up to −35 V on outputs
2+RAM	20	Yes	28	−9	Yes	Yes	No	Yes	Expanded RAM/ROM version of 2100
2+RAM	19	Yes	28	−9	Yes	Yes	No	Yes	Same as 2400 but one less I/O line and handles up to −35 V on outputs
2+RAM	33	Yes	40	−9	Yes	Yes	No	Yes	Version of TMS2400 with expanded I/O and 4 a-d channels
2+RAM	32	Yes	40	−9	Yes	Yes	No	Yes	Same as 2600 but one less I/O line and handles up to −35 V on outputs
2+RAM	23	Yes	28	−9	Yes	Yes	No	Yes	Has on-chip complex-sound generator
10+RAM	30	Yes	40	5	Yes	Yes	No	Yes	Available in 28- or 40-pin package
RAM	22	No	28	5	Yes	Yes	No	Yes	

All the processors in the MN1400 family are available in at least one other technology. The 18- and 16-pin versions are about the smallest microcomputers available, although most versions still retain at least 2/3 of the instructions available on the 1405. The CMOS versions can also be ordered with operating voltages of up to 10 V, and all models include an 8-bit counter-timer with 7-bit prescaler.

All MN1500-family processors include an 8-bit counter-timer and an 8-bit serial shift register; all I/O lines are bidirectional and the chips also have a power-down mode to minimize power dissipation.

General-purpose internal registers	No. of I/O lines	Additional special support circuits	Package size (no. of DIP pins)	Voltages required (V)	Prototyping system available	Assembly-language programming system	High-level-language programming system	Time-sharing cross software	Comments
32	32	Yes	40	5	Yes	Yes	Yes	Yes	Has on-chip UART and two 16-bit counters
RAM	27	Yes	40	5	Yes	Yes	Yes	—	CMOS version of 8049
RAM	27	Yes	40	5	Yes	Yes	Yes	—	Enlarged version of 8049
RAM	20	Yes	28	5	Yes	Yes	No	Yes	28-pin version of the 3870, otherwise identical
RAM	32	Yes	40	5	Yes	Yes	No	Yes	
RAM	32	Yes	40	5	Yes	Yes	No	Yes	All devices in the MK3870 family use an improved architecture that divides internal RAM into 64 bytes of scratchpad registers and either 0 or 64 additional bytes of executable RAM; all units contain an 8-bit counter-timer with 7-bit prescaler
RAM	32	Yes	40	5	Yes	Yes	No	Yes	
RAM	32	Yes	40	5	Yes	Yes	No	Yes	
RAM	29	Yes	40	5	Yes	Yes	No	Yes	The 3873 family includes a serial port on chip with baud-rate generator and gives up three parallel I/O lines for the port. The 3875 series includes provision for battery backup for the on-chip RAM, thus permitting very-low-power standby operation
RAM	29	Yes	40	5	Yes	Yes	No	Yes	
RAM	30	Yes	40	5	Yes	Yes	No	Yes	
RAM	30	Yes	40	5	Yes	Yes	No	Yes	
RAM	31	Yes	40	5	Yes	Yes	Yes	Yes	6801 has masked ROM, 701 has EPROM
RAM	31	Yes	40	5	Yes	Yes	Yes	Yes	680144 is mask-programmable ROM version; 70144 is EPROM version
RAM	20	Yes	28	5	Yes	Yes	Yes	Yes	Includes 8-bit counter-timer, zero-crossing detector, self-check mode
RAM	32	Yes	40	5	Yes	Yes	Yes	Yes	Similar to 6805U2 but includes four-channel a-d converter
RAM	20	Yes	40	5	Yes	Yes	Yes	Yes	Has 8-bit, 4-channel a-d converter
RAM	20	Yes	28	5	Yes	Yes	Yes	Yes	Phase-locked loop on chip for synthesizer applications
RAM	32	Yes	40	5	Yes	Yes	Yes	Yes	Expanded I/O and ROM version of 6805P2
RAM	20	Yes	28	5	Yes	Yes	Yes	Yes	EPROM version of 6805P2
RAM	31	Yes	40	4.75 to 5.75	Yes	Yes	Yes	Yes	EPROM version of 6805R3; bootstrap loader is available
RAM	20	Yes	40	5	Yes	Yes	Yes	Yes	Has EPROM (not ROM), 24 I/O lines, and counter-timer
RAM	32	Yes	40	3 to 6	Yes	Yes	Yes	Yes	ROM version of 146805E2; dissipates <1 mW on standby, 20 mW when active
RAM	32	Yes	40	3 to 6	Yes	Yes	Yes	Yes	New chip. Version of 146805G2 with EPROM; requires −13-V programming voltage
RAM	27	Yes	40	5	Yes	Yes	Yes	Yes	Enlarged proprietary version of Intel 8049 processor with transparent improvements
RAM	5	Yes	40	5	Yes	Yes	Yes	No	Fast math CPU plus ROM, RAM, and DMA logic
RAM	27	Yes	40	5	Yes	Yes	Yes	Yes	CMOS version of 8048
RAM	27	Yes	40	5	Yes	Yes	Yes	Yes	CMOS 8048 with extra instruction to allow expanded counter-timer and power-down features
RAM	27	Yes	40	5	Yes	Yes	Yes	Yes	
RAM	27	Yes	40	5	Yes	Yes	Yes	Yes	
RAM	27	Yes	40	5	Yes	Yes	Yes	Yes	Piggyback prototyping version of 8048 family; operates at all speeds; has selectable on-chip RAM; accepts standard EPROMS
16+RAM	48	Yes	64Q	5	Yes	Yes	Yes	Yes	Includes 12-bit timer-counter and serial I/O port in addition to 48 I/O lines
16+RAM	48	Yes	64Q	5	Yes	Yes	No	Yes	7801 with 6-kbits of ROM
16+RAM	48	Yes	64Q	5	Yes	Yes	No	Yes	7801 with multiplication and division instructions, 8-channel a-d converter, 16-bit counter-timer, serial I/O
16+RAM	48	Yes	64FP	5	Yes	Yes	No	Yes	CMOS 7801 with reduced instruction set, halt, and stop modes
RAM	27	Yes	40	5	Yes	Yes	Yes	Yes	CMOS version of 8048 (1/30 the power)
RAM	27	Yes	40	5	Yes	Yes	Yes	Yes	CMOS version of 8049
16	23	Yes	28	5	Yes	Yes	No	Yes	New chip. Has on-chip 2-wire multimaster serial I/O
16	23	Yes	28	5	Yes	Yes	No	Yes	New chip. Has on-chip 2-wire multimaster serial I/O
16	23	Yes	28	5	Yes	Yes	No	Yes	New chip. Has on-chip 2-wire multimaster serial I/O
16	23	Yes	28	5	Yes	Yes	No	Yes	New chip. Has on-chip 2-wire multimaster serial I/O
RAM	8	Yes	40	3 to 7	Yes	Yes	No	Yes	Software-compatible with the MC6802
RAM	27	Yes	40	3 to 6	Yes	Yes	No	Yes	CMOS version of 8048
RAM	27	Yes	40	3 to 6	Yes	Yes	No	Yes	CMOS version of 8049
16×16	7	Yes	40	4 to 6.5	Yes	Yes	Yes	—	ROM version of CDP1805AC
RAM	16	Yes	28	3 to 6	Yes	Yes	Yes	Yes	ROM version of CDP6805E2
RAM	32	Yes	40	3 to 6	Yes	Yes	Yes	Yes	ROM version of CDP6805E2
RAM	32	Yes	40	5	Yes	Yes	Yes	Yes	Single-chip version of 6502
16	21	No	28	5	Yes	Yes	No	Yes	Subset of 8048H; compatible with 8048
16	18	Yes	40	5	Yes	Yes	No	Yes	Slave processor when used with MAB8048/9
16	27	Yes	40	5	Yes	Yes	No	Yes	Compatible wih 8048
16	27	Yes	40	5	Yes	Yes	No	Yes	Expanded memory version of 8048H
16	21	No	28	5	Yes	Yes	No	Yes	Subset of SAB8048
16	27	Yes	40	5	Yes	Yes	No	Yes	Compatible with the 8048
128	32	Yes	40	5	Yes	Yes	No	Yes	Has two 16-bit timer-counters, serial channel
16	30	No	40	5	Yes	Yes	No	No	SAB80310 with external EPROM, has serial input for PCM carrier signal
16	21	No	28	5	Yes	Yes	No	No	Has serial input for PCM carrier signal. SAB80312 with external EPROM
16	30	No	40	5	Yes	Yes	No	No	Watch chip with US/Euroformat, stop, alarm, reset. SAB80315 with external EPROM

Word size, data/instruction (bits)	Manufacturer	Device	Process technology	On-chip RAM size (bits)	On-chip ROM/ PROM size (bits)	Off-chip memory expansion	No. of basic instructions	Maximum clock frequency (kHz)	On-chip clock	Instruction time, shortest/longest (μs)	TTL-compatible	BCD arithmetic	On-chip interrupts/levels	No. of subroutine
8/8	Synertek	Z8S	NMOS	64×8	1024×8	Yes	47	8000	Yes	1.5/4.25	Yes	Yes	Yes/8	8
8/8		Z86C01	CMOS	128×8	2040×8	Yes	47	8000	Yes	1.5/4.25	Yes	Yes	Yes/8	8
8/8	Texas Instruments	TMS7020	NMOS	128×8	2048×8	Yes	61	8000	Yes	1.6/19.2	Yes	Yes	Yes/4	RA
8/8		TMS7040	NMOS	128×8	4096×8	Yes	61	8000	Yes	1.6/19.2	Yes	Yes	Yes/4	RA
8/8	Toshiba	TMP80C48P	CMOS	64×8	1024×8	Yes	96	6000	Yes	2.5/5	Yes	Yes	Yes/1	8
		TMP80C49P-6	CMOS	128×8	2048×8	Yes	96	6000	Yes	2.5/5	Yes	Yes	Yes/1	8
8/8		TMP8022	NMOS	64×8	1024×8	No	70	3600	Yes	8.38/16.76	Yes	Yes	No	8
8/8		TMP8048P	NMOS	64×8	1024×8	Yes	96	6000	Yes	2.5/5	Yes	Yes	Yes/1	8
8/8		TMP80C49AP	CMOS	128×8	2048×8	Yes	97	11,000	Yes	1.36/2.72	Yes	Yes	Yes/1	8
8/8		TMP8049P-6	NMOS	128×8	2048×8	Yes	96	6000	Yes	2.5/5	Yes	Yes	Yes/1	8
8/8		TMP8051P	NMOS	128×8	4096×8	Yes	111	12,000	Yes	1/4	Yes	Yes	Yes/2	RA
8/8	Zilog	Z8601	NMOS	144×8	2048×8	Yes	47	12,000	Yes	1.3.33	Yes	Yes	Yes 6	RA
8/8		Z8611	NMOS	144×8	4096×8	Yes	47	12,000	Yes	1/3.33	Yes	Yes	Yes/6	RA
8/12	General Instrument	PIC1650	NMOS	32×8	512×12	No	30+	1000	Yes	4/8	Yes	Yes	No	2
8/12		PIC1650XT	NMOS	32×8	512×12	No	30+	4000	Yes	4/8	Yes	Yes	No	2
8/12		PIC1654	NMOS	32×8	512×12	No	30+	4000	Yes	2/4	Yes	Yes	No	2
8/12		PIC1655	NMOS	32×8	512×12	No	30+	1000	Yes	4/8	Yes	Yes	No	2
8/12		PIC1655XT	NMOS	32×8	512×12	No	30+	4000	Yes	4/8	Yes	Yes	No	2
8/12		PIC16C55	CMOS	32×8	512×12	No	30+	1100	Yes	4.5/9	Yes	Yes	No	2
8/12		PIC1656	NMOS	32×8	512×12	No	30+	4000	Yes	4/8	Yes	Yes	Yes/1	3
8/13		PIC1670	NMOS	64×8	1024×13	No	40+	8000	Yes	1/2	Yes	Yes	Yes/1	6
16/16	Intel	8096	NMOS	116×16	4096×16	Yes	95	12,000	Yes	1.6.5	Yes	No	Yes 8	RA
16/16	Mostek	MK68200	NMOS	128×16	2048×16	Yes	87	6000	Yes	0.5/9.2	Yes	Yes	Yes/2	RA
16[1]	Texas Instruments	TMS9940E/ 9940M	NMOS	128×8	2048×8	No	68	4000	Yes	2/452	Yes	Yes	Yes/4	64
16/16	Texas Instruments	TMS320	NMOS	144×16	1536×16	Yes	60	20,000	Yes	0.2/0.6	Yes	No	Yes/1	RA
16/17	American Microsystems	528211	NMOS	256×16	512×18/ 128×16	Yes	731	16,667	Yes	0.3	Yes	No	Yes/1	1
16/23	NEC Electronics	μPD7720	NMOS	128×16	512×23	No	48	8000	Yes	0.25	Yes	No	Yes/1	4
25/24	Intel	2920	NMOS	40×25	192×24	No	21	6670	Yes	0.6/0.6	Yes	N.a.	N.a.	0
25/24		2921	NMOS	40×25	192×4	No	21	10,000	Yes	0.4/0.4	Yes	N.a.	N.a.	0

[1]8 bits externally, 16 bits internally.

General-purpose internal registers	No. of I/O lines	Additional special support circuits	Package size (no. of DIP pins)	Voltages required (V)	Prototyping system available	Assembly-language programming system	High-level-language programming system	Time-sharing cross software	Comments
RAM	30	Yes	40	5	Yes	Yes	Yes	Yes	Differs from the Z8601 only in two pins, interrupt vectors and some on-chip resources
RAM	32	Yes	40	3 to 6	Yes	Yes	Yes	Yes	
3+RAM	32	No	40	5	Yes	Yes	Yes	No	TMS7000 series is first microprogrammable 8-bit microcomputer. Unique strip architecture reduces chip size and eases customization
3+RAM	32	No	40	5	Yes	Yes	Yes	No	
RAM	27	Yes	40	5	Yes	Yes	Yes	Yes	CMOS version of 8048
RAM	27	Yes	40	5	Yes	Yes	Yes	Yes	CMOS version of 8049
16	13	No	20	5	Yes	Yes	No	Yes	Pin-compatible with the 8022
16	27	Yes	40	5	Yes	Yes	No	Yes	Pin-compatible with the 8048
16	27	Yes	40	5	Yes	Yes	No	Yes	Has halt instruction for software-controlled power-down
16	27	Yes	40	5	Yes	Yes	No	Yes	Pin-compatible with the 8049
32	32	Yes	40	5	Yes	Yes	Yes	Yes	Pin-compatible with 8051
RAM	32	Yes	40	5	Yes	Yes	Yes	Yes	Has two counter-timers and UART
RAM	32	Yes	40	5	Yes	Yes	Yes	Yes	Double ROM version of Z8601
RAM	32	No	40	5	Yes	Yes	No	Yes	12-bit-wide ROM, all instructions are single-word
RAM	32	No	40	5	Yes	Yes	No	Yes	Crystal-controlled oscillator
RAM	12	No	18	5	Yes	Yes	No	Yes	Reduced I/O version of PIC1650
RAM	20	No	28	5	Yes	Yes	No	Yes	Reduced I/O version of PIC1650
RAM	20	No	28	5	Yes	Yes	No	Yes	Mask-programmable $\overline{RTCC}$ prescaler
RAM	20	No	28	5	Yes	Yes	No	Yes	CMOS version of PIC1655, three state outputs
RAM	20	No	28	5	Yes	Yes	No	Yes	PIC1655 with internal and external interrupts
RAM	32	No	40	5	Yes	Yes	No	Yes	Enlarged memory version of PIC1650
116	48	No	48	5	3Q,'83	Yes	No	No	New chip. Has 8-channel 10-bit a-d converter and eight 16-bit timers
15	40	Yes	48	5	See Comments	See Comments	No	No	New chip. Modeled after 68000, prototyping, assembly language systems available mid-1983
RAM	32	No	40	5	Yes	Yes	Yes	Yes	Two versions available, one with a 2-kbyte EPROM, the other with a 2-kbyte ROM
2+RAM	27	No	40	5	No	Yes	No	Yes	New chip.
8×16	8	No	28	5	Yes	Yes	No	No	Programmable digital signal processor includes 12 × 12 parallel multiplier with 16-bit rounded product
RAM	12	No	28	5	Yes	Yes	No	Yes	Includes a second data-coefficient ROM (512 × 13 bits), has serial port, 16 × 16-bit parallel multiplier, and two accumulators
RAM	12	No	28	5, −5	Yes	Yes	Yes	Yes	Analog processor, accepts four analog inputs and delivers up to eight analog outputs digitally processed according to a program stored in EPROM
RAM	12	No	28	5, −5	Yes	Yes	Yes	Yes	Analog processor, accepts four analog inputs and delivers up to eight analog outputs digitally processed according to a program stored on chip in a mask-programmed ROM

Semiconductor houses sharpen C-MOS skills, plan high-density memory, set sights on 32-bit microprocessors; special-purpose CPUs share limelight with new configurations of general-purpose units

While continuing the race for smaller and speedier chips, the electronics industry has nevertheless begun a subtle shift in emphasis: exploiting device density to create more highly integrated components, instead of just smaller ones. The once-clear division between semiconductor and component technology has become lost as logical functions begin sneaking into memory and even linear parts and as microprocessor chips add cache memory and linear capability. Driven by this trend to more complex circuitry on single chips, the machinery for semiconductor manufacturing has already begun to slip out of the optical and into the X-ray realm.

As usual, monolithic memories lead the way to vertical advance in component technology. Even as production lines are spinning out 64-K dynamic random-access memory chips, worldwide attention has veered to 256-K dynamic RAMs and preliminary designs for megabit chips. And even while semiconductor manufacturers continue to optimize such high-speed devices as these for the computer industry, they are further refining technology like that for electrically-erasable programmable read-only memory for the merchant market. Designs like point-of-sale terminals will enjoy faster, denser EE-PROM chips that nonetheless retain standard pinouts.

Similarly, with the software marketplace demanding compatibility in the face of advances in microprocessor design, semiconductor houses are plotting cautious courses toward full 32-bit processor chips. As these 32-bit powerhouses begin to trickle into the market in the next two years, system designers will begin to experiment with alternative system architectures. Multiprocessor systems will become much more common, challenging the software industry to utilize the hardware to its full speed and capability.

Along with these microprocessor-based "mainframes," a growing class of portable computers will gain a major boost next year, when complementary-MOS designs for CPU and memory chips join new flat-panel displays. As the semiconductor houses squeeze these complex circuits into dense C-MOS chips, C-MOS technology will further refine techniques like trench isolation. On the other hand, display makers will highlight thin-film transistors built right on the display substrate.

This same turn to C-MOS and integrated function will be applied across other analog devices like converters. Along with high-speed, high-precision hybrids, converter firms are keying in more closely on monolithic C-MOS converters that will soon rival the resolution of board-level subsystems.

Even while C-MOS becomes the standard process for highly integrated components, power electronics is handing out new twists to power sources. Exotic magnetic configurations will bring new heights of efficiency to smaller switching power supplies. Much of this higher efficiency can be traced back to new high-power MOS field-effect transistors capable of handling hundreds of volts at gigahertz frequency with low losses.

Thus, the stage is set for a new generation of industrial and consumer systems. Backed by new lithography equipment exploiting X-ray and even plasma sources, device sizes will continue to plummet to a point where even the 5-volt industry power-supply standard will be in jeopardy of bowing to lower system voltages.

Contributors to this section were Roderic Beresford, former Solid State Editor; Stephen Evanczuk, Software Editor; Steve Zollo, Assistant New Products Editor; and Jerry Lyman, Packaging & Production Editor.

Semiconductor houses fine-tune MOS and bipolar technologies; designers exploit gallium arsenide for fast, high-density parts

Although the major semiconductor companies are still tweaking 2-micrometer MOS and bipolar processes for volume production, the state of the art will be dipping down towards 1-μm geometries in the coming year. As scaled-down complementary-MOS processes show their colors, the designers of very large-scale integrated circuits are gearing up for the challenge posed by chips using multilevel interconnections to link a half million devices, including subnanosecond transistors. At the same time, development engineers are gaining familiarity and proficiency with exotic technologies built on silicon-on-insulator and gallium arsenide substrates.

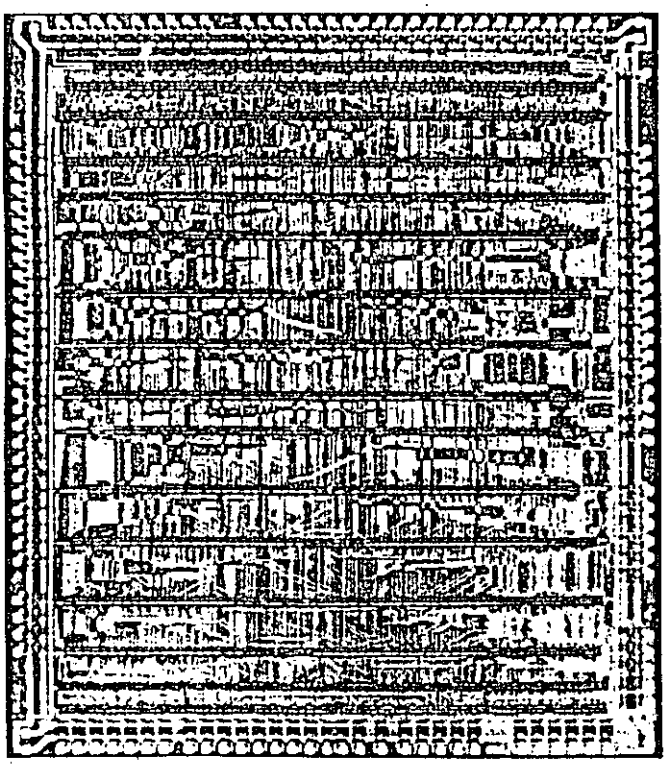
The dollar value of IC production in 1983—some $12 billion—looks to split about evenly between MOS and bipolar parts, with memory accounting for almost half of the former and glue logic making up a comparable fraction of the latter. The forces of integration will be swinging the balance to the MOS side in the coming year as bipolar logic loses out to semicustom C-MOS chips and fast memory continues to fall prey to power-efficient MOS designs. Still, the bipolar legacy will live on, as it does in the Hitachi Ltd. C-MOS arrays that borrow the high-current devices for output stages and in MOS VLSI systems that use epitaxial layers to control substrate effects.

With the new 1-to-1.5-μm C-MOS processes, typical loaded gate delays well under 2 nanoseconds beckon. C-MOS leaders Toshiba Corp. and Hitachi Ltd. of Japan will not be the only forces at the beachhead as such U. S. companies as National Semiconductor Corp., Intel Corp., and General Electric Co. rally behind the flexibility of C-MOS designs.

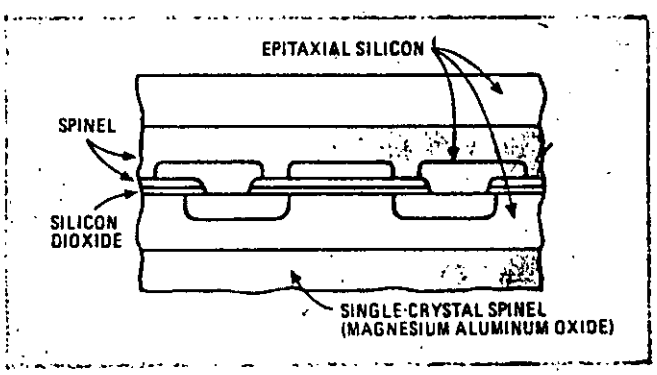
Next year Toshiba should push into production a 1.2-μm p-well process with shallow low-resistance junctions and 200-to-250-angstrom-thick gate oxides. Likely targets include 20,000-gate-arrays and 256-K static random-access memories. A modified local-oxidation scheme will suffice for isolation spacings around 2 μm. As scaling continues, deluxe C-MOS processes will acquire high-performance options like trench isolation for dense low-capacitance gates, twin wells for optimized n- and p-channel characteristics, and even epitaxial layers. Epitaxial C-MOS is capable of circumventing latchup problems when the layer thickness is comparable to the device feature sizes.

Having tested the C-MOS waters with some easy microcomputer chips, Intel is putting the finishing touches on CH-MOS III, as the Santa Clara, Calif., company calls its next generation of high-performance n-well C-MOS, which optimizes the speed of its n-channel devices. The technology features wafer-stepper lithography of 1.5-μm lines and 250-Å gate oxides. As applied to a 64-K dynamic RAM shown this year, it also holds oxide encroachment (the bird's beak) to just 0.2 μm.

For an impending foray into merchant waters, GE is tuning a twin-well 1.2-μm C-MOS process that will migrate to the company's Intersil subsidiary, in Cupertino, Calif., during the coming year. Commercial targets include gate arrays and microcomputers. The process, which yielded 0.5-ns gates in test chips, uses a retrograde p well whose peak dopant concentration occurs 1 μm below the surface of the wafer. The retrograde well kills the gain of the parasitic npn transistor, helping with latchup immunity.



Automated design. RCA Corp.'s computer-aided design system automatically laid out this C-MOS-on-sapphire standard-cell chip for the U. S. Army. With 3-μm design rules, the 13,000-plus transistors and their manifold interconnections occupy some 110,000 mil².



Toward a 3-d chip. An all-epitaxial approach using single-crystal insulators marks out Sanyo Co.'s 3-d circuitry. The SiO_2 layer (color) needed for a quality interface is formed by diffusing oxygen through the thin spinel layer before growing source, drain, and gate contacts.

Another retrograde-well approach surfaced at IBM Corp.'s Thomas J. Watson Research Center, Yorktown Heights, N. Y. Researchers there favor a retrograde n-well for 1-μm C-MOS circuitry. Counterdoping an n well forms a buried p-channel device with sharper turn-off characteristics than can otherwise be obtained. Although the computer giants continue to push bipolar technology, the industry seems to be warming up to C-MOS gate arrays for mainframe applications. Still, for all their efficient use of power, C-MOS devices as yet cannot command a speed advantage over bipolar ones.

On the n-channel MOS side, 1-μm processes strike new highs in IC speed and density. At Bell Laboratories, Murray Hill, N. J., 1-to-1.5-μm X-ray lithography is turning out such wonders as a 20-ns 16-bit multiplier and a 5-ns 4-K static RAM. In the commercial arena, Intel is patterning 1-μm minimum features in its H-MOS III-E process, which this year led to the first 256-K erasable programmable read-only memory.

Bipolar is a moving target

The bipolar faction is hardly holding still under fire from the MOS camp; rather, scaling-down, isolation, and interconnection technology are sharpening up the competitive edge of the high-speed parts. International Business Machines Corp.'s General Technology division, East Fishkill, N. Y., has demonstrated one of the first four-level-metal processes on a 10,000-gate Schottky TTL array that marks a healthy step beyond 3081-era technology. One version of the array also includes a 32-by-18-bit RAM with a worst-case access time of 10 ns.

At Nippon Telegraph & Telephone Public Corp., a 1.5-μm oxide-isolated bipolar process scored a 5,000-gate array with loaded delays of 0.5 ns. Basewidths of 1,400 Å contribute to the high transistor cut-off frequency of 7 gigahertz. NTT chose nonthreshold logic (similar to current-mode logic) for its high speed and density, as well as for its low power of 1 milliwatt per gate.

Commercial bipolar circuitry forged ahead this year as Fairchild Camera & Instrument Corp., Mountain View, Calif., took its Isoplanar process down to 1.25-μm minimum features. The firm is applying the technology to emitter-coupled-logic gate arrays with typical 0.35-ns internal delays. Elsewhere, the family tree of bipolar ICs is sprouting a new branch as Texas Instruments Inc., Dallas, prepares to debut its next advance in low-power Schottky logic. Featuring fully oxide-isolated subnanosecond 2-μm devices, the process will launch large-scale ICs like 8-bit-slice processors and high-speed controllers.

Future bipolar devices may benefit from an innovation conceived for MOS parts—silicon-on-insulator substrates. Researchers at the Massachusetts Institute of Technology, in Cambridge, Mass., succeeded in fabricating bipolar transistors on the substrates, a feat impossible without nearly defect-free crystals. The SOI community continues to expand as more and more ingenious methods for stacking up silicon layers spur development of three-dimensional circuitry.

The Sanyo Co. revealed its unique approach to SOI, which is being buoyed up by the Japanese government's project on future electron devices. Rather than recrystalize deposited silicon layers, Sanyo says it can grow single-crystal films atop single-crystal insulators of spinel, an oxide of magnesium and aluminum. At Dortmund University, West Germany, engineers are implanting nitrogen about a quarter micrometer deep in silicon wafers to form a buried insulating layer. Although a promising technique for doubling C-MOS density and trimming down delays, it opens no new avenues to three-dimensional circuits.

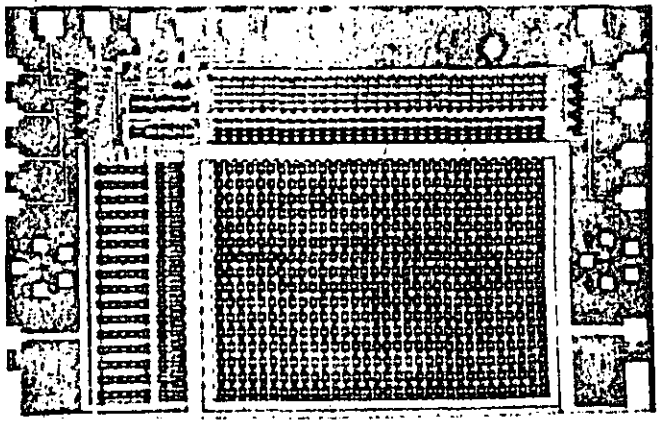
Working with what promises to become the "traditional" SOI technology—recrystalization of deposited polysilicon—researchers at France's Centre d'Etudes des Télécommunications claim to have achieved 90% functional yields for a variety of test circuits. The technique, refined by the Grenoble laboratory, uses a silicon nitride mask during recrystalization. SOI's deleterious grain boundaries tend to form below the nitride, where the heating is greater; because they are located precisely, the boundaries can be removed by oxidation, leaving high-quality silicon islands.

Finally, although silicon-on-sapphire technology remains too costly for high-volume applications, it nonetheless has strategic value. The first chip out of Hughes Aircraft Co.'s efforts for the government's Very High-Speed Integrated Circuits program is a C-MOS-on-sapphire correlator that clocks along at 80 megahertz, according to the Culver City, Calif., firm.

The military-industrial complex

Even as VHSIC drives the semiconductor producers to 1.25-μm silicon chip sets, the Department of Defense is preparing to shell out what it takes to get gallium arsenide ICs out of the laboratory and into production. In the absence of a substantial effort at military contractors, not to mention commercial semiconductor companies, U. S. participation in GaAs markets hinges on an aggressive Government program. DOD is lining up just that, with its sights set on 10-ns 16-K static RAMs and 6,000- to 10,000-gate arrays. Even if the program succeeds in establishing pilot production within the 2½ years allotted, the U. S. may be playing second fiddle to Japan again, as aggressive research and development in the East has yielded big gains in processes for manufacturing GaAs.

Four companies—Fujitsu Ltd., NEC Corp., Hitachi



GaAs RAM. Fujitsu Ltd.'s self-aligned tungsten-silicide-gate technology produced this 1-K static random-access memory. The 9,300-square-mil chip achieves parity with n-MOS RAM area, accesses typically in 4 nanoseconds, and consumes 68 milliwatts.

Ltd., and Mitsubishi Electric Corp.—are developing GaAs ICs under the auspices of the supercomputer project of Japan's Ministry of International Trade and Industry. Furthermore, government funding is only a small part of the total expended on GaAs research. Unlike the other three companies, Fujitsu is not using government money to research "conventional" GaAs ICs—it has already mastered them on its own, apparently.

Achieving manufacturable GaAs chips is now primarily a matter of substrate purity and surface stability. Crystal suppliers are at work on the first problem. Their growth processes leave an uneven concentration of chromium impurities along the growth axis, resulting in variations in the background doping of the wafers sliced from the crystal. Because of the nonuniformity, wafer-to-wafer variation of FET threshold voltages is impractically large. From chip to chip, however, threshold variations are only 5% to 10%, thanks to clever device structures that place the active layers below the unstable surface. Fujitsu's 1-K static RAM has a cell almost the same size as an n-MOS silicon version. A study predicts that scaling down to 1-μm gate lengths on the 1-K chip will achieve power dissipation less than 500 MW and access times faster than 1 ns.

With this self-aligned metal-semiconductor-FET technology already being transferred to the factory, Fujitsu's laboratories are focusing on high-electron-mobility transistors. Watch for the company to fabricate a several-hundred-gate array using HEMTs in the coming year. Operated at 77 K, with a 0.4-V logic swing, HEMT gates should switch in 30 picoseconds and consume only about 150 microwatts.

Memory makers focus on 256-K dynamic RAMs, n-MOS static RAMs with bipolar speeds, and high-speed EE-PROMs that challenge bipolar PROMs

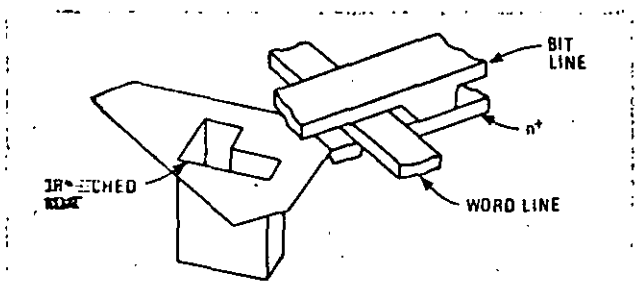
The MOS memory markets remain a paragon of elasticity: the further prices fall, the more bits they buy. As the recovery picked up this year, home and personal computers, terminals, word processors, and the like gobbled up 64-K dynamic random-access memories as fast as the suppliers could ship them. The battle lines are mostly drawn for next year's 256-K shoot-out.

Some of the volleys will pick off traditional static-RAM markets as byte-wide and C-MOS dynamic parts make their debuts. Static-RAM efforts continue to bear fast fruit as skilled circuit crafters push n-MOS delays down to 35 nanoseconds in 16-K parts. Fast 64-K parts fairly beckon as well in C-MOS, a 256-K design will be unveiled even as the 64-K chips are just getting off the ground.

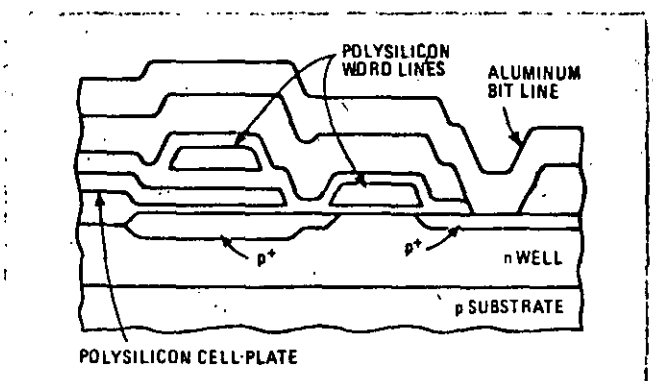
Among nonvolatile advances is to be counted the jump to 64-K densities in electrically erasable programmable read-only memory. High-speed EE-PROM will rear its head as well, perhaps spelling doom for bipolar fuse PROMs. Meantime, the ultraviolet E-PROM is giving up erasability, to become a true packed-in-plastic jelly bean, and mask ROMs are staggering ahead to the megabit mark.

Although 2-μm processes and novel interconnection technology put the 256-K dynamic RAMs on their way to market, process and layout are still being sharpened up to shoehorn the bulky designs into plastic packages. With 64-K assembly lines churning around the clock to fill unleashed demand, new development dollars are being siphoned into further 256-K work and even into preliminary studies of the megabit part. The drama now lies less in looking for the singularly successful process than in the verdict of the users on the various organizations and special features being adopted willy-nilly by a still-expanding field of contestants.

As had been predicted by some observers, many of the Japanese came around to wafer-stepper lithography and laser redundancy for this round of competition, while most U. S. producers already had those tricks mastered. Still, no sign as yet says that the new manufacturing technologies are bogging down Japanese efforts the way they did the ambitious U. S. 64-K development programs last time around. As before, the Japanese strengths are an early presence and adequate parts.



Megabit cell? Studies at Hitachi Ltd. show this moat structure can accommodate the storage capacitance of a conventional dynamic RAM cell. Filling the trench with a sandwich of polysilicon and silicon oxide and nitride achieves a 45-fF figure for a cell just 32 μm square.



C-MOS dynamic RAM. The cell structure of Intel's complementary-MOS dynamic RAM shows the p-channel select transistor in the n-type well that protects the array from the stray charge of an alpha-particle strike. Soft-error rates drop by orders of magnitude.

In the U. S., Western Electric Co. in Allentown, Pa., shines in technology, even if others have clearer sights on new market angles. The first American 64-K chip packaged in plastic will come from Western Electric, whose initially oversized RAM—already well appointed with an epitaxial laver and tantalum silicide wiring—is reportedly headed for the 50,000-mil-square mark. Among the Japanese chips, those of Fujitsu, NEC, and Oki Electric Industry Co. are in plastic already.

The approaches of Fujitsu and NEC diverge about as much as any. Fujitsu's triple-polysilicon 2.5-μm process contrasts sharply with NEC's double-metal 1.3-μm scheme. Yet the chips are virtually the same size—around 52,700 mils square. Thanks to the less aggressive design rules, Fujitsu's 64-K lines can spew out 256-K chips on demand. Further scaling-down may leave Fujitsu with the most cost-effective part, although its storage capacitance is already a perilously low 35 femtofarads. To handle the scaling problem in an experimental 512-bit dynamic RAM, IBM's Essex Junction, Vt., facility adopted Mitsubishi's push-plate pulsing technique, which, in effect, doubles signal strength.

Oki's first chip—conservative and large—was never a serious contender (the company has had prototypes for almost two years). Redesigned on a 2-μm double-metal process, Oki's new part is in plastic, having achieved a respectable 56,500 mils square. Although the company farmed out its 64-K design to National Semiconductor Corp., Santa Clara, Calif., it has made no announcement of such a joint venture at the 256-K level. (Nor has National commented on its own 256-K program, save to say that it has one.) The remaining Japanese competitors, Hitachi, Toshiba Corp., and Mitsubishi Electric Corp., are all building with similar processes that sport 2-μm linewidths and molybdenum polycide interconnects.

Grinning over profits from a 64-K rebound that has the company headed for 4 million pieces a month by year end, Mostek Corp., Carrollton, Texas, is mounting two 256-K designs. Said to be loaded with circuit-design innovations, the first is an unorthodox 32-K-by-8-bit chip aiming squarely at the small memory systems that are this year's saving grace for the dynamic-RAM producers. The byte-wide behemoth will be joined by a plain-vanilla cousin in several months. Both chips use the lightly-doped-drain process that Mostek recently enlisted for a fast 80-ns 64-K design. With the higher-performance niche expanding, watch for Mostek or others to put sub-100-ns nonmultiplexed dynamic RAMs on the market.

Specialization trumps

Texas Instruments Inc.'s trump card in this round is said to be an all-in-one chip that can be customized at the metal level for half a dozen specialty markets. The Dallas-based giant might thereby efficiently serve the users of nibble mode, page mode, the various refreshing options, by-4-bit or by-8-bit parts, and so on. Retaining the grounded epitaxial layer and introducing an as-yet-unnamed polycide, TI's prototypes reportedly check in at around 60,000 mils square. Formal announcements are expected shortly.

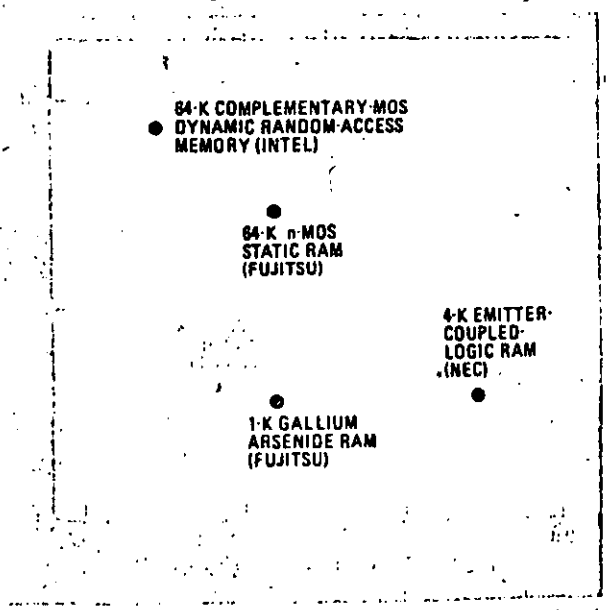
Despite a large and highly partitioned array that adds up to a supremely conservative design, Motorola's 256-K chip has been stalled at the starting gate nearly a year. Once over the hurdle posed by the new interconnection technology, the Schaumburg, Ill.-based company still faces some scaling-down work to supply a plastic-packaged part. Finally, Micron Technology Inc., Boise, Idaho, and Inmos Corp. in Colorado Springs, Colo., are expected to announce 256-K parts in the coming year.

As advanced-development teams look ahead to the megabit array, their required reading list most likely starts with Intel Corp.'s complementary-MOS dynamic RAM technology. The 64-K C-MOS part Intel showed this year casts an array of p-channel devices in an n-type well that protects the bits from the stray charges of alpha particle hits. C-MOS has scored an orders-of-magnitude improvement in soft-error rates, down to 10 FITs: mere parts per million per thousand hours, compared with 1,000 FITs or more for conventional n-MOS parts.

Though the C-MOS logic gates use more transistors than n-MOS versions, the peripheral circuitry on Intel's chip ends up needing far fewer clock generators and far less random logic. C-MOS has thus also scored a boost in area efficiency and design flexibility. A 256-K C-MOS dynamic RAM, says an Intel study, would cost about the same as an n-MOS chip, considering the number of masks and process steps, as well as the chip size.

While Intel in particular is eyeing short-term opportunities for special-purpose (and special-price) parts, other dynamic-RAM makers confirm C-MOS as the long-term favorite. The process should clear away many of the problems that will face a 1-megabit chip—die size, circuit complexity, low-voltage operation, and soft-error rate, not to mention power consumption.

With a 32-K-by-8-bit dynamic RAM already announced and C-MOS clearly in the cards, the microprocessor-oriented static RAMs are going to be sharing their turf with their denser dynamic cousins. Meantime, Toshiba will soon be pushing the state of the art in C-MOS static RAMs



New era. The fundamental RAM parameters—speed, power, density—attain unprecedented values in Bell Labs' n-MOS part, made with X-ray lithography. The power-delay-area product of these new parts is 10 times better than in other state-of-the-art RAMs.

choosing the Inmos fast static RAM for its next machine. Yet the ECL partisans are doing all they can to stay a step ahead. They have put pnp-load technology to work in making the jump to 16-K ECL parts, with 25-ns maximum access times the happy result. Hitachi, Fujitsu, and Fairchild Camera & Instrument Corp. are neck and neck on the new generation of chips. With lateral pnp transistors as active loads on the cross-coupled npn devices of the ECL memory cell, standby currents can be held to a few microamperes or less per bit, holding promise for denser arrays as well.

Despite those successes, the handwriting is on the wall for bipolar static RAMs because the truly speed-crazed applications will soon receive the ministrations of the gallium arsenide developers. In contrast to the nearly complete lack of interest at major U. S. semiconductor companies, the Japanese are dead serious about commercial parts. In the case of Fujitsu, self-aligned MES-FET technology is already being transferred to factories. Production of 1-K or denser RAMs with access times below 5 ns is on the horizon.

MOS PROMs promised

Towards the other bastion of bipolar memory technology—programmable ROMs—the merciless masters of MOS are readying a fatal blow. Two start-up companies—Lattice Semiconductor Corp., Portland, Ore., and Exel Microelectronics Inc., Milpitas, Calif.—promise EE-PROMs—at 32-K and 64-K densities—with bipolar-like speeds of 45 ns. With standard pinouts and a fraction of the bipolar versions' power consumption, the new parts will be shoe-ins around the industry. The added benefits of in-circuit alterability would seem finally to seal the fate of fuse PROMs.

Meanwhile, the mainstream EE-PROM market—such as it is—may finally be taking off with the debut of 64-K parts. Although the present 16-K chips lag too far behind the density of other kinds of nonvolatile storage, 64-K versions will find plenty of sockets that today house ultraviolet E-PROMs. As the full-featured configuration of the device from Xicor Inc., Milpitas, Calif., gets adopted by others, a standard 5-V EE-PROM emerges: one with enough support circuitry on chip to make the part look like a static RAM to the system designer.

With the recently introduced 32-K part from NCR Corp.'s Microelectronics division, Miamisburg, Ohio, and Xicor's imminent 64-K chip, a page-write mode further eases the application of EE-PROMs. Borrowing a shadow-RAM idea, the page-write mode puts a small static-RAM buffer on chip with the EE-PROM, so that up to 16 bytes can be written in at microprocessor speeds and then transferred off line to the slow EE-PROM array.

Shadow RAMs themselves hit the 4-K mark this year. Xicor, Intel, and NCR lead the way and also, apparently, are cooperating on a pinout standard. The "dream RAM"—a dense EE-PROM cell mated with a dynamic-RAM buffer—remains a dream.

While EE-PROMs try to displace E-PROMs, E-PROMs will close further in on ROMs. Though it has been tried before, Intel has at last succeeded in putting E-PROMs in plastic packages, saving the costs of Cerdip for only that minority of users that actually intends to erase the chips. The problem with these one-time-only-programmable E-PROMs has been in testing for quality and reliability—in this case, programming margins and retention time—without programming the devices. Intel manages that feat with some proprietary test programs and on-chip monitors. Other vendors indicate that they have comparable efforts under way.

Intel added further munitions to its arsenal this year, in the form of H-MOS II-E, a next-generation E-PROM process sporting minimum features of 1 μm. The technology was applied to the first 256-K part, which is about half the size of Fujitsu's C-MOS attempt and a third smaller than AMD's recently announced n-MOS chip. The technology is bound to have an impact on the lower-density high-volume 64-K and 128-K parts.

Ballooning ROMs

With E-PROMs snapping at their heels, ROMs can only get bigger and bigger. Megabit chips surfaced this year from NEC and Hitachi, with the latter showing a novel approach to redundancy. Additional cells store parity bits, which are automatically checked during a read operation. Fabrication flaws get fixed each time bad data is retrieved from the array.

Hitachi's 1-Mb ROM accesses at typical microprocessor speeds. NEC's, on the other hand, is one of the "mass ROMs": slow chips adequate in applications such as character tables and speech synthesis. Mass ROMs at the 4-Mb level are almost within reach of several vendors.

At lower densities, fast-turnaround manufacturing methods are seen as a help in holding off the onslaught of plastic-packaged E-PROMs. Several companies are developing a process to program ROMs after the metalization step by implanting through finished devices.

Chip houses ready major 32-bit microprocessors, while compatibility issues spur drive to virtual machines and special-purpose processors aid move to multiprocessing

Without any doubt, VLSI has brought with it brand new concepts of system design. But associated with these developments in VLSI has been the influence of software technology on the hardware design process. Fewer and fewer chips now enter production without previous scrutiny to ensure their compatibility with the overall objectives of system-software designers.

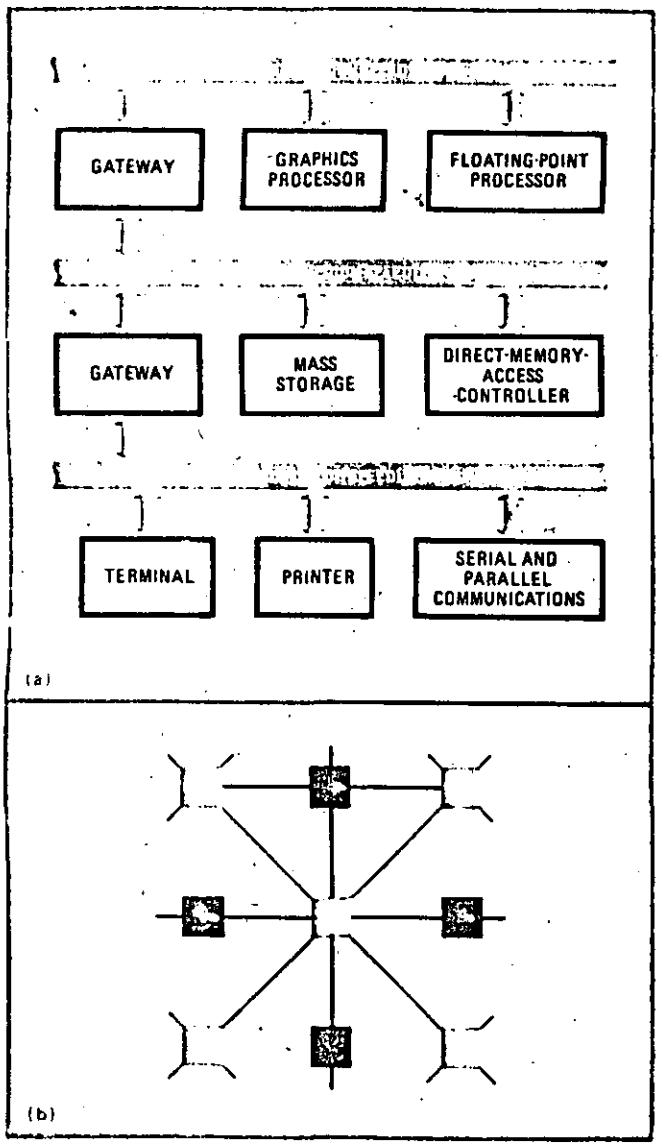
Thus, there has been a marked tendency toward inclusion of direct hardware support for the needs of high-level languages, such as sophisticated instructions for bit manipulation and handling character strings. Chips are even reflecting software engineers' movement to struc-

tured programming techniques, in which isolated software routines handle specific tasks of the overall system: hardware designers are partitioning the overall system function onto separate chips for tasks like mathematics, graphics, communications, and text processing. Even more significantly, hardware design is anticipating the next generation of distributed software systems.

Distributed systems will be characterized by several individual processors or groups of processing elements communicating over a number of different buses. Besides the system bus, which has traditionally served as the sole communications medium between processor, memory, and peripherals, separate buses will be available for communications between central processing units, as well as local communications between CPU, memory, and certain dedicated peripherals.

Proprietary systems, like those from Masscomp, Littleton, Mass., or Lisp Machines Inc., Culver City, Calif., already use multiple-bus architectures. Similarly, bus products like Intel's LBX local-bus extension of the Multibus should find increasing popularity as designers venture more deeply into multiprocessor systems. Using Intel's LBX memory boards, for example, system designers can create microprocessor subsystems—with local memory, accessed through the local bus—that communicate across the (Multibus) system bus.

Advances in bus technology and alternative system configurations will come from industry designers and academic researchers pushing forward on parallel-processing systems, the next frontier for computer science. Architectures like those created by the Blue CHiP project, at Purdue University, West Lafayette, Ind., and the systolic array work at Carnegie-Mellon University, Pittsburgh, emphasize homogeneous arrays of processing elements (CPUs). The transputer—a single-chip processing element to be announced by Inmos Corp., Colorado Springs, Colo.—will be the first industry contribution designed for parallel systems. Its parallel-processing language, Occam, puts Inmos in the unique position of providing both hardware and software building blocks for future parallel systems.



Alternate architectures. Single-bus designs are yielding to architectures using two or more buses of various speeds (a). Future systems (b) may use an array of similar processors (black) variously configured with intelligent switches (colored boxes).

End of the race

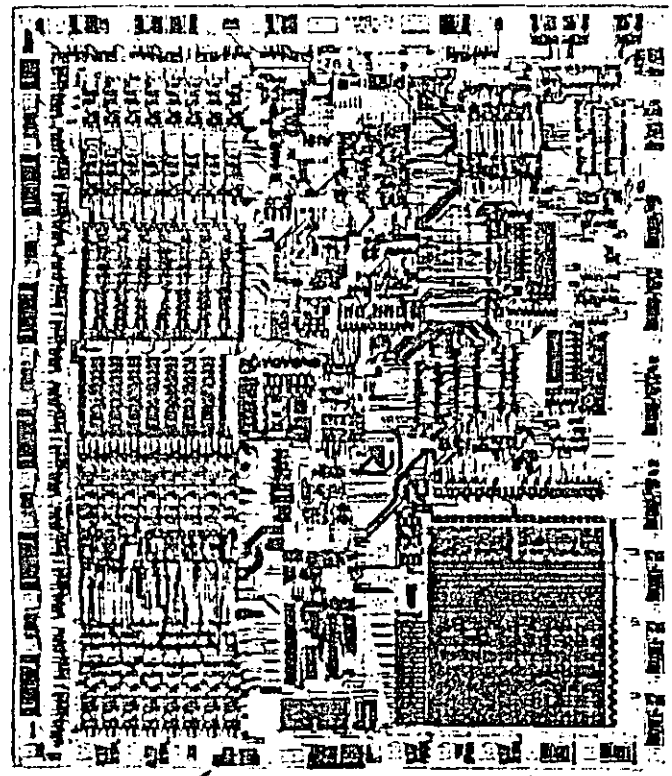
On the one hand, the introduction over the next year or two of nonproprietary 32-bit microprocessor chips, like Intel's iAPX-386, Motorola's 68020, National's 32132, and Zilog's 80000, is the culmination of experience gained through 8- and 16-bit chips, marking the end of the race for more "bit-ness." On the other hand, the race for horizontal expansion of basic functional capability will just be beginning.

With the use of 16- and 32-bit microprocessors in an increasing variety of portable equipment, low-power C-MOS versions of the popular microprocessor chips will become more common. Already Harris Corp., Melbourne, Fla., has introduced a C-MOS version of Intel's 16-bit 8086 microprocessor and should soon be joined by C-MOS versions of 16-/32-bit microprocessors from the other semiconductor houses. Furthermore, while Western Electric leads the way with its C-MOS Bellmac-32, the major semiconductor houses will be racing to produce C-MOS versions of their full 32-bit chips.

Responding to designers' needs in work stations and robotics, microprocessor chip designers are including features for memory management, as well as overall system security. Work stations—intended to manipulate the relatively large software structures of computer-aided design (CAD) or of data bases in office automation—will need access to large virtual storage within networks of shared resources and interprocessor communications.

So along with the need for efficient memory management of a large virtual address space, the CPU will have to ensure the system's security—both from the users of proprietary software within local work stations and from remote users accessing the work station through the network. Even more vital is the role of protection in robotics systems, where inadvertent corruption of system memory could result in damage and even deaths.

Still, besides high-powered 32-bit microprocessors, the next few years should see an explosion in special-purpose processors optimized for special tasks, like speech pro-



C-MOS processing power. Presaging the next generation of processors, this C-MOS version of Intel's 16-bit 8086 from Harris Semiconductor needs an operating current of only 10 mA/MHz. The 69,300-mil² chip runs at 5 MHz, with an 8-MHz version soon to follow.

cessing, communications, graphics, and data-base access. Similarly, designers can expect a host of new microprocessor chips designed around familiar CPUs but with extra peripheral features, like analog-to-digital converters added to fit target environments.

Alternative architectures

Special- and general-purpose CPUs with enhanced features set the stage for the next major showdown in microprocessor design, though one likely to see a shift in champion as technology develops. The more familiar approach backs general-purpose CPUs, whose instruction sets contain enough instructions for any logic or arithmetic task. That philosophy was used throughout the evolution of 8-, 16-, and 32-bit microprocessors.

Reduced-instruction-set (RISC) architectures—as espoused by David Patterson, among others—are extreme examples of this general-purpose approach. RISC machines concentrate on fundamental logical instructions within a high-performance architecture, so that complex operations—often represented by single machine instructions—are performed by several low-level but high-speed instructions. If it is unclear how deeply this design might influence future microprocessor architectures, the elegance of design—reflected in the relative ease in building compilers for these machines—stands to make a mark in the microprocessor industry.

The alternative approach elects to build special-purpose CPUs, with instruction sets and architectures optimized for certain kinds of tasks. For TI, a special-pur-

Seccombe: pilot of a work station's parallel development efforts

When a group is attempting to advance on three fronts simultaneously—process technology, chip integration, and system design—it is hard to anticipate the next stumbling block. But, suggests S. Dana Seccombe, keeping the design teams small and using graphs to track potential causes of failure can make the course a little less parlous.

Seccombe should know: he helped manage the N-MOS III parallel development effort that created new semiconductor processing technology and very large-scale integrated-circuit designs for Hewlett-Packard Co.'s 9000 series of 32-bit engineering work stations.

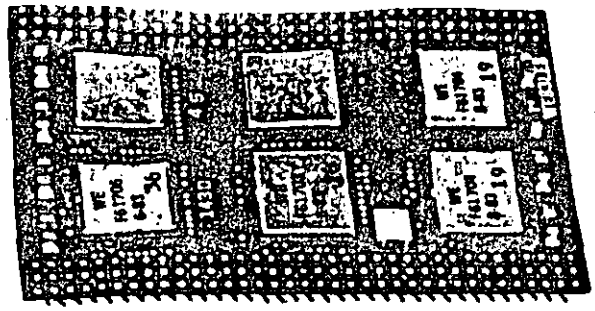
The result was six chips, some with nearly 500,000 transistors. They are a 32-bit central processing unit, an input/output processor, a memory controller, a clock device, a 128-K random-access memory, and a 640-K read-only memory. All but the ROM were eventually used in the 9000 computer.

"We have a history here of going through technology development while working on the product," comments the research and development manager of the systems technology operation in Hewlett-Packard's computer integrated-circuit division at Fort Collins, Colo. The company's N-MOS I process, for instance, was in devices for HP calculators back in 1970, and N-MOS II chips followed for the 9825 and 9845 calculators.

When the project began, in 1975, Seccombe was heading the process thrust as one of four project managers. Holder of a master of science and a BSEE from Massachusetts Institute of Technology, he had joined HP in 1972, after completing the MIT engineer's graduate degree (between the master's and a Ph.D.). Then in 1976, the N-MOS III parallel development effort defined the fabrication technology, but few of the automated engineering tools needed to complete the job were as yet available, he says. In the following year, he assumed responsibility for half the circuit design effort, and two years later, he was named to his present position.

In addition to developing the process, chips, and basic system architecture, the team of nearly 30 engineers had to weigh related technologies not generally available in the mid-1970s—including advanced testing concepts and production techniques.

Then, to speed production, the N-MOS III team used a "state variable" charting system to separate the independent factors impacting chip yields. "It's an attempt to decouple this morass of data," whose different elements normally interact with one another, Seccombe explains. **—J. Robert Lineback**



C machine. Expecting big things of Bell Labs' Unix operating system, Western Electric optimized its Bellmac-32 microprocessor for the language C. Six C-MOS VLSI chips on a multilayer board fulfill CPU, bus interface, bus control, and memory management functions.

pose signal-processing chip like the 320 is perhaps a natural outgrowth of its Macrostore chips, which allow users to define custom instruction sets for special applications. But the 320 takes this the required further step in optimizing the chip's architecture for the application.

On a system level, Western Electric's Bellmac-32 is an example of a microprocessor optimized for a particular language—C. From Western Electric's point of view, a C machine is the best bet, particularly with the explosive growth enjoyed by Bell Laboratories' Unix operating system. Similarly, Intel's iAPX-432 was designed to optimize object-oriented programming, where software structures are treated as objects. Although they recognize that this approach will result in optimal execution of applications for the intended environment, strategic marketing managers are cautious about commitments to any environment, even one with a heady potential, like the Unix software market.

For companies hoping to cash in on specialized operating environments, a third approach has possibilities: a hybrid technique that employs a user-microprogrammable architecture combining a general-purpose CPU design with the ability to offer special-purpose instruction sets. Only NCR has as yet committed itself to this approach, in a user-microprogrammable 32-bit chip set. Most designers have not embraced microprogrammable CPUs because they have a decided reputation as hard to microprogram.

Nevertheless, software technology is gradually catching up and should make microprogramming less of a problem. Other semiconductor houses, aware of the problems of microprogrammable architectures, express skepticism about the acceptance of such systems. Still, the same chip makers, aware of the advantages of special-purpose instruction sets afforded by user-microprogrammable architectures, are not closing the door on future offerings.

Special-purpose chips

Meanwhile, semiconductor manufacturers concentrate on broadening the base of support chips. Besides hardware support for such aspects of system design as timing control and interrupt handling, chip makers are beginning to offer hardware support for various aspects of application handling. For example, Intel's coprocessor chips for text and graphics are among the first examples of melding application-level software with system hardware. National continues the trend toward more integrated controllers with its new CRT controller chip, replacing a boardful of support chips and offloading graphics screen maintenance from the CPU.

Besides these, Weitek Corp.'s introduction last year of a set of chips for high-speed math offers designers an even greater range of alternatives for systems. The Santa Clara, Calif., company used a pipelined architecture to squeeze five million operations a second out of 32-bit multiplier and addition chips. Furthermore, the chips support IEEE floating-point format and exception handling. Alternatively, for designers who do not require such high-speed performance, semiconductor manufacturers are pairing their CPU chips with their own line of math chips, such as National's 16081 floating-point chip or Intel's 80287 floating-point coprocessor.

Just as floating-point chips remove the burden of number crunching from the CPU, system data-transfer transactions will enjoy advances in special chips for peripheral control. Much as most mainframe designs include special input/output processors to relieve the CPU of continual low-level interrupt activity, chip designers are building a high degree of independence into this new generation of peripheral controller chips.

Integration expands abilities of converters, displays; flat-panel displays, C-MOS join in portable products; magnetics, power MOS FETs back new power supplies

The buzz words by which new electronic components are measured—faster, smaller, cheaper—are humming more this year than in any other. Products like smart power devices are coming of age, and such aging components as operational amplifiers are refusing to die. Power MOS FETs are getting cheaper and more powerful, while op amps are getting faster.

The accuracy of analog-to-digital converters is keeping just a couple of bits behind the accuracy of their digital-to-analog counterparts. A 16-bit a-d converter is no longer a rarity. And it seems that the achievements gained in display technology are progressing geometrically rather than arithmetically. Power supply gains are being quietly made thanks to achievements in the areas of magnetics, power transistors, and voltage regulators.

In display technology, it was the best of times for the Japanese and the worst of times for U. S. companies. Virtually all the significant achievements in flat-panel display technology over the past year were made by the Japanese, while American companies suffered setbacks. First the bad news. Fairchild, Texas Instruments, Beckman, and NCR no longer supply commercial displays.

MICROPROCESADORES Y MICROCOMPUTADORAS

MATERIAL ADICIONAL

NOV. 1984 PROF. LUIS PEÑARRIETA.

Chart 1. Company and Configuration Data

COMPANY NAME AND ADDRESS	PRODUCT NAME	PRICE	HARDWARE REQUIREMENTS	OPERATING SYSTEM	WRITTEN IN	HARD DISK COMPATIBLE	MULTI-USER COMPATIBLE
For the Apple							
Apple Computer, Inc. 20525 Mariani Ave., Cupertino, CA 95014	Quick File III	$100	128K Apple III, external drives	Apple III SOS	Pascal	y	n
Applied Software Tech. 14125 Capri Dr., Ste. 4, Los Gatos, CA 95030	Versa Form	$389/$495	Apple II (64K), 2 drives	UCSD Pascal	Pascal	y	y
Avant-Garde Creations Box 30160, Eugene, OR 97402	Complete Mailing and Billing System	$75	Apple II	Apple II	Basic	n	n
High Technology Software Products Box 14665, Oklahoma City, OK 73113	Information & Data Master	$100/$150	48K Apple II, 1,2 disk drives	Apple DOS	Applesoft Basic	n	n
Information Unlimited Software 2401 Marinship Way, Sausalito, CA 94965	Datadex	$150	Apple II (48K) 1 drive	DOS 3.3	Applesoft	y	y
LJK Enterprises, Inc. Box 10827, St. Louis, MO 63129	Data Perfect	$100	48K Apple	DOS 3.3	Machine	n	y
Pascal Systems, Inc. 830 Menlo Ave., #109, Menlo Park, CA 94025	The Data Machine	$625	Apple II/III, IBM PC (128K)	UCSD 2.1 and 2.4	UCSD Pascal	y	y
Readers Digest (Microcomputer Division) Pleasantville, NY 10570	List Maker	$98	48K Apple II/II+, TRS-80 I/III	depends on version	Basic/Assembly	n	n
Sierra On-Line, Inc. 36575 Mudge Ranch Rd., Coarsegold, CA 93614	The General Manager	$230	48K Apple II	Apple DOS	Assembly/Basic	y	n
Silicon Valley Systems 1625 El Camino Real, #4, Belmont, CA 94002	List Handler	$90	48K Apple, 1 disk drive	DOS	Assembly	n	n
Software Publishing Corp. 1901 Landings Dr., Mountain View, CA 94043	PFS:File	$125/$175	Apple II (48K), Apple III (128K)	Apple's UCSD-p	Pascal	y	n
Software Technology for Computers 430 A Main St., Watertown, MA 02172	IFO-Version II	$200	Apple II/II+ (48K) 2 drives	DOS 3.3	Assembly	y	n
Stoneware, Inc. 50 Belvedere St., San Rafael, CA 94901	DB Master	$229	Apple II, 1-4 drives	n/a	Basic	y	n
Westware, Inc. 2455 SW 4th, Ontario, CA 97914	Systems II	$425	Apple II (48K), 2 drives	DOS 3.3	Applesoft	y	n
For CP/M computers							
Ashton-Tate 9929 West Jefferson, Culver City, CA 90230	dBASE II	n/a	Z80, 8080, 8085/6/8	CP/M, MSDOS (16-bit)	Assembly	y	n
Chang Laboratories, Inc. 10228 N. Stelling Rd., Cupertino, CA 95014	Data Plan	n/a	CP/M	CP/M	n/a	y	n
Compumax Box 7239, Menlo Park, CA 94025	Microbase	$149	Z80	CP/M	Basic	y	n
DJR Associates Inc. 303 S. Broadway, Tarrytown, NY 10591	FMS-80	$500-$990	48K RAM	CP/M, MP/M, Turbo Dos	Assembly, C	y	y
Friends Software Corp. Box 527, Berkeley, CA 94701	Friends Report Writer	$295-$495	56K bytes, 2 drives	CP/M	Assembly	y	n
International Software Enterprise, Inc. 85 W. Algonguin, Arlington Hts, IL 60005	MDBS	$1500-$5000	Z80, 8080/5/6/8 PDP-11	CP/M, MP/M, PCDOS, MSDOS	C	y	y
Micro Ap, Inc. 7033 Village Parkway, Dublin, CA 94568	Selector	$900	52K working space	CP/M, MP/M, (80-86)	CB80 Machine	y	y

y = yes; feature is included
n = no; feature is not included
n/a = information not available

Chart 1. Company and Configuration Data (continued)

COMPANY NAME AND ADDRESS	PRODUCT NAME	PRICE	HARDWARE REQUIREMENTS	OPERATING SYSTEM	WRITTEN IN	HARD DISK COMPATIBLE	MULTI-USER COMPATIBLE
For CP/M computers (continued)							
Micro Applications Group 20201 Sherman Way #205, Canoga Park, CA 91306	MAG/Base	$295-$795	48K, Z80, 8080, 8086	CP/M, MP/M	CBasic/CB80	y	y
MicroPro 33 San Pablo, San Rafael, CA 94903	Infostar	n/a	48K, dual floppies	CP/M-80	8080 Assembly	y	n
Micro Data Base Systems Box 248, Lafayette, IN 47902	The Knowledge Manager	$250	Z80, 8080/5/6, PDP-11	CP/M, PCDOS, TRSDOS	Assembly, C	y	n
MicroRIM, Inc. Box 585, Bellevue, WA 98009	MicroRIM	$795	Z80, CP/M, 8080/5/6	CP/M	Fortran	y	n
Pearlsoft Box 13850, Salem, OR 97309	Personal Pearl	$295	Z80, 8080, 8086	CP/M, SOS, MSDOS	Pascal	y	y
Quanteckna Research Corp. 6902 220th S.W., Mountlake Terrace, WA 98043	The Quad	$495/$595	Z80, 8080, 8085	CP/M, MP/M	CB-80	y	y
Structured Systems Group 5204 Claremont, Oakland, CA 94618	Analyst	$250	48K RAM	CP/M	CBasic 2	y	y
For other machines							
American Planning Corp. 4600 Duke St., Alexandria, VA 22304	MIS Builder	$495-$5000	North Star Horizon/ Advantage	North Star DOS	APC Basic	y	y
B&B Software Box 2090, Ann Arbor, MI 48106	Corp	$195	TRS-80 I/II/III	TRSDOS	Basic	y	n
Cado Systems Corp. 2771 Toledo St., Torrance, CA 90510	Wordbase	varies	Cado Computer System	Cado OS	Cado4	y	y
CRS Associates Box 40283, Philadelphia, PA 19106	Waloss II	$250	64K, 1 drive TRS-80 II/III	TRSDOS	Basic	y	n
Dravac Ltd. 16 Muller Rd., Oakland, NJ 07436	ANDI	$2500	AM-100, 32K, CRT	AMOS	Assembly	y	y
Insoft, Inc. 10175 SW Barbur Blvd #202B, Portland, OR 97219	Data Design	$225	IBM (8088), 2 drives	MSDOS	Basic	y	n
Micro Architect, Inc. 96 Dothan St., Arlington, MA 02174	IDM-X	$399-$598	TRS-80 II, IBM PC	TRSDOS, PCDOS, CP/M 2.2	Basic	y	n
Phase One Systems, Inc. 7700 Edgewater St. #830, Oakland, CA 94621	Control	$695	48K	Oasis	Basic/Assembly	y	y
Programming Technology 1417 Bridgeway, Sausalito, CA 94965	The Organization Analyst	$287	IBM PC (64K+) 2 drives	IBM DOS	Pascal/Macro	y	n
Savvy Marketing, Int'l. 100 S. Ellsworth 9th floor, San Mateo, CA 94401	Savvy	$950	n/a	Savvy	Fortran	y	n
Super Soft, Inc. Box 1628, Champaign, IL 61820	Personal Data Base	$125	IBM PC (48K)	IBM PC, DOS	Basic	y	n
Tamarack Software Water Street, Darby, MT 59829	The Wiz	$130	Commodore 8032, 8050	Manufacturer	Basic/Assembly	y	n
Threshold Software, Inc. 1832 Tribute Rd., Ste. E, Sacramento, CA 95815	File/idea	$250	HP-85, 1 drive	HP-85 (proprietary)	Basic	y	n
Vector Graphic 500 S. Ventura Park Rd., Thousand Oaks, CA 91320	Data Manager	$495	Vector (64K)	CP/M (Vector only)	Basic	y	n

y = yes; feature is included
n = no; feature is not included
n/a = information not available

Chart 2. An Overview of Database Structure

PRODUCT NAME	MAXIMUM NUMBER OF RECORDS PER FILE	FILES HAVE FIXED OR VARIABLE RECORD SIZES	MAXIMUM NUMBER OF FIELDS PER RECORD	MAXIMUM FIELD SIZE IN CHARACTERS	FORMAT OF STORED FILES (i.e. HOW THEY ARE STORED ON DISK)	CAN NON-DATA-BASE PROGRAMS ACCESS FILES CREATED BY THE DATA-BASE PROGRAM?	HOW ARE DATABASE FILES INDEXED?	MAXIMUM NUMBER OF INDEXES	SPECIAL FEATURES AND PROGRAMS
For the Apple									
Quick File III	30K-110K	variable	15	76	packed binary, ASCII	y*	other	15	
Versa Form	300	variable	50-75	78	non-std. Pascal files	y*	sep. indexes	1	
Complete Mailing Label and Billing System	1,200	variable	17	28	Apple random text	y	sep. indexes	17	name, phone, zip lists, 2-level search, alphabetic directory
Information & Data Master	100	variable	20	99	ASCII	y	sep. indexes	5	
Datadex	n/a	either	50	38	ASCII	y	other	1	user-written programs can be used on data and called from program
Data Perfect	variable	either	32	127	Text files	y*	relation tags	variable	can design specialized screen masks
The Data Machine	32,767	either	40/100	60	packed binary, ASCII	y	sep. indexes	1	DIF-file interface, Pascal Utilities
List Maker	250	either	12	30	ASCII	n	sep. indexes	3	
The General Manager	16M bytes	variable	99	39	packed binary	y	relation tags	1	Interface to *most* databases
List Handler	3,000	variable	250	250	Nibble format	y*	other	unlimited	label, list and letter printer
PFS:File	1,000-3,200	variable	1,600-3,200	839-1,679	packed binary	y	relation tags	n/a	
IFO-Version II	1,600	either	40	1,440	packed binary	y	relation tags	2	Phonetic search capability
DB Master	<5M bytes	either	200	220	packed binary	y*	sep. indexes	10	utility and stat programs
Systems II	700	variable	40	25	ASCII	y	sep. indexes	1	
For CP/M computers									
dBASE II	65,535	variable	32	254	ASCII	y*	other	7	
Data Plan	64K	variable	32	99	ASCII	y	sep. indexes	n/a	
Microbase	4,000	variable	50-600	29	packed binary, ASCII	y	sep. indexes	1	backup, reload, mailing label
FMS-80	65,535	either	255	255	packed binary, ASCII, other	y	sep ind.; rel. tag	unlimited	report generator, menu creator, "Transform" utility
Friends Report Writer	65,000	either	ltd. by memory	256	packed binary, ASCII	y	other	n/a	reads many other database files, has multi-dimensional cross tabulation
MDBS	unlimited	variable	65,535	65,535	special internal form	y	other	unlimited	manipulation language, retrieval system generates output
Selector	99,999	variable	89	screen width	ASCII	y	sep. indexes	99	auto-pilot

The "raw power" of a database program can be measured by the capacity of the system. The differences in the systems can be seen in the maximum field size, and the format of stored files. The number of indexes available is a measure of how many ways of organizing the data can exist simultaneously. (In other words, a sales file could be indexed or "ordered" by invoice number, date, customer and salesman, if four indexes were simultaneously available.) The ability to access database files with non-database programs allows a user to create custom applications and possibly interface with other packaged software.

Key

y = yes; feature is included n = no; feature is not included n/a = information not available *non-database programs can access these files, but only with the aid of a "file translation" program.

Chart 2. An Overview of Database Structure (continued)

PRODUCT NAME	MAXIMUM NUMBER OF RECORDS PER FILE	FILES HAVE FIXED, OR VARIABLE RECORD SIZES	MAXIMUM NUMBER OF FIELDS PER RECORD	MAXIMUM FIELD SIZE IN CHARACTERS	FORMAT OF STORED FILES (i.e. HOW THEY ARE STORED ON DISK)	CAN NON-DATA-BASE PROGRAMS ACCESS FILES CREATED BY THE DATA-BASE PROGRAM?	HOW ARE DATABASE FILES INDEXED?	MAXIMUM NUMBER OF INDEXES	SPECIAL FEATURES AND PROGRAMS
For CP/M computers (continued)									
MAG/Base	2,500	either	999	35	ASCII	yes	rel. tags, sep. indexes	99	screen formating, file management
Infostar	ltd. by disk size	variable	255	120	ASCII	y	sep. indexes	1	
The Knowledge Manager	65,535	either	255	65,535	packed binary, ASCII, other	y*	sep. index, other	unlimited	integrated spreadsheet, table processing & programming language
MicroRIM	999,999,999	either	127	254	MicroRIM binary	y	other	127	direct access to database
Personal Pearl	no limit	variable	250	screen width	variable length	y*	sep. indexes	unlimited	sort, file maintenance, compress data, rebuild index.
The Quad	65,535	variable	30	30	ASCII	y	relation tags	2	payroll, linking, data conversion
Analyst	ltd. by storage	variable	50	1-255	ASCII	y	sep. indexes	n/a	create, modify, print, select
For other machines									
MIS Builder	disk limited	fixed	99	40	ASCII	y	sep. indexes	15	access data files, full system utilities
Corp	n/a	variable	n/a	30	packed binary	no	relation tags	n/a	
Wordbase	4,194,303	fixed	240	255	packed binary	y	sep. indexes	1	any sub-program may be encoded in Basic language and invoked by DBMS
Waloss II	variable	either	25	250	packed binary	y	sep. indexes	1	
ANDI	65,535	fixed	63	9,011	float, binary, ASCII	y	sep. indexes	unlimited	maintenance utilities, program generation sub-systems
Data Design	65,000	variable	40	79	packed binary	y	sep. indexes	26	backup, restore, form letter programs
IDM-X	30,000	variable	60	255	packed binary, ASCII	y	other	1	text formatter, full utilities
Control	65,535	either	170	60	packed binary	y	sep. indexes	1	forms option
The Organization Analyst	unlimited	variable	ltd. by memory	256	Pascal formatted	y	other	1	
Savvy	20,239	n/a	255	255	ASCII	y	sep. indexes	255	Robot programmer
Personal Data Base	unlimited	variable	20	50	packed binary	no	sep. indexes	unlimited	
The Wiz	300,000	variable	64	78	ASCII	y*	other	n/a	
File/Idea	2,000	variable	25	95	ASCII	y*	sep. indexes	5	
Data manager	disk limited	variable	99	70	ASCII	y*	sep. indexes	3	

The "raw power" of a database program can be measured by the capacity of the system. The differences in the systems can be seen in the maximum field size, and the format of stored files. The number of indexes available is a measure of how many ways of organizing the data can exist simultaneously. (In other words, a sales file could be indexed or "ordered" by invoice number, date, customer and salesman, if four indexes were simultaneously available.) The ability to access database files with non-database programs allows a user to create custom applications and possibly interface with other packaged software.

Key

y = yes; feature is included n = no; feature is not included n/a = information not available *=non-database programs can access these files, but only with the aid of a "file translation" program.

Chart 3. Update on Updates

PRODUCT NAME	ABILITY TO UPDATE MORE THAN ONE FILE SIMULTANEOUSLY	ABILITY TO UPDATE FILES SELECTIVELY	ABILITY TO PERFORM MATH CALCULATION WITH UPDATE	CAPABILITY TO PERFORM LOGICAL COMPARISON IN UPDATE FUNCTION	CAN ONE DATABASE OPERATE (OR UPDATE) ANOTHER?	
					MERGE	APPEND
For the Apple:						
Quick File III	n	y	n	n	n	n
Versa Form	n	y	y	y	n	n
Complete Mailing Label and Billing System	n/a	n/a	n/a	n/a	n	y
Information & Data Master	n/a	n/a	n/a	n/a	n	y
Datadex	n/a	n/a	n/a	n/a	n/a	n/a
Data Perfect	n	y	y	y	y	n
The Data Machine	n	y	y	y	y	y
List Maker	n	y	n	n	y	y
The General Manager	n	y	y	y	y	n
List Handler	n	y	n	n	y	y
PFS:File	n	n	n	y	n	y
IFO-Version II	n	y	y	y	n	n
DB Master	y	n	y	n	y	n
Systems II	n	n	y	n	n	n
For CP/M computers:						
dBASE II	n	y	y	y	y	y
Data Plan	n/a	n/a	n/a	n/a	y	y
Microbase	n	y	n	n	n	n
FMS-80	y	y	y	y	y	y
Friends *Report* Writer	y	y	y	y	y	n
MDBS	y	y	y	y	n	n
Selector	y	y	y	y	y	y
MAG/Base	y	y	y	n	y	y
Infostar	y	y	y	y	y	y
The Knowledge Manager	y	y	y	y	y	y
MicroRIM	y	y	n	n	y	y
Personal Pearl	n	y	y	y	y	y
The Quad	y	y	y	y	y	y
Analyst	y	y	y	n	n	n
For other machines:						
MIS Builder	y	y	y	y	y	y
Corp	n	y	n	n	n	n
Wordbase	n	y	y	y	y	y
Waloss II	y	y	y	n	y	n
ANDI	y	y	y	n	y	y
Data Design	n	y	y	n	n	n
IDM-X	n	y	y	y	y	y
Control	n	y	n	n	n	n
The Organization Analyst	y	y	n	n	y	y
Savvy	y	y	y	y	y	y
Personal Data Base	n	y	n	n	n	n
The Wiz	n	y	n	n	y	n
File/idea	n	y	y	n	n	n
Data Manager	n	y	y	y	n	y

The ability of a database program to update or modify a database file from another database file is important in many business applications. For example, an accounts receivable transaction file may be updated by sales transactions and cash receipts transactions (two separate files). When creating an application with a database management program, the systems designer should have some flexibility. The several different kinds of "update" capabilities are summarized above. Manufacturers of programs with "n/a" notations had no response to these survey questions about updates.

Key:
y = yes; feature is included
n = no; feature is not included
n/a = information not available

Continued on page 148

Continued from page 66

Chart 4. Program Documentation Survey

PRODUCT NAME	COMPUTER INSTALLATION HELP	SAMPLE PROGRAMS	MANUAL LENGTH (PAGES)
For the Apple:			
Quick File III	n	y	180
Versa Form	y	y	220
Complete Mailing Label and Billing System	n	y	n/a
Information & Data Master	n	n	200
Datadex	n	n	96
Data Perfect	n	n	110
The Data Machine	y	y	185
List Maker	n	y	80
The General Manager	n	y	150
List Handler	n	y	73
PFS:File	n	n	100
IFO-Version II	n	n	200
DB Master	n	n	n/a
Systems II	n	y	25
For CP/M computers:			
dBASE II	n	y	300
Data Plan	y	y	n/a
Microbase	n	y	n/a
FMS-80	y	y	n/a
Friends Report Writer	y	y	120
MDBS	y	y	450
Selector	y	y	352
MAG/Base	y	y	n/a
Infostar	n	y	600
The Knowledge Manager	y	y	300
MicroRIM	y	y	150
Personal Pearl	y	y	76-184
The Quad	y	y	400
Analyst	n	n	n/a
For other machines:			
MIS Builder	n	y	300
Corp	n	y	27
Wordbase	n	y	n/a
Waloss II	y	n	75
ANDI	n	y	400+
Data Design	n	y	n/a
IDM-X	y	n	60
Control	n	y	54
The Organization Analyst	n	y	150
Savvy	y	y	n/a
Personal Data Base	n	y	50
The Wiz	n	y	120
File/idea	n	y	60
Data Manager	n	n	100

Program documentation can make the difference between a successful installation and a frustrated user with a system that delivers only a fraction of its potential. While manual length is not an all-inclusive indicator, it can provide a clue as to a combination of program complexity and level of documentation. (In defense of short manuals, some programs—a minority—are so user-friendly that they don't require extensive documentation.)

Key:
y = yes; feature is included
n = no; feature is not included
n/a = information not available

Chart 5. Report Generator Features

"PRINT MASKS"				IS REPORT GENERATOR SUITABLE FOR THE PREPARATION OF		
DOLLARS	WHOLE NUMBERS	NEGATIVE SIGNS	EXPONENTIAL	CHECKS	INVOICES & BILLINGS	LISTS
y	y	n	n	y	y	y
n	n	n	n	y	y	y
n	n	n	n	n	n	y
y	y	n	n	n	n	y
y	y	n	n	y	y	y
y	y	y	y	y	y	y
y	y	y	n	y	y	y
n	n	n	n	y	y	y
y	y	y	y	y	y	y
n	n	n	n	y	y	y
n	n	n	n	n	n	y
y	y	n	n	y	y	y
y	y	n	n	y	y	y
n	n	n	n	n	n	y
y	y	y	n	n	n	y
y	n	y	n	y	y	y
n	y	n	n	n	n	y
y	y	y	n	y	y	y
n	n	n	n	n	n	y
y	y	y	y	y	y	y
y	y	y	n	y	y	y
y	y	y	y	y	y	y
y	y	y	n	y	y	y
y	y	y	n	y	y	y
y	y	y	y	y	y	y
y	y	n	n	n	n	y
y	y	y	n	y	y	y
n	n	n	n	n	n	y
y	y	n	n	y	y	y
y	y	n	n	y	y	y
y	y	y	y	y	y	y
y	y	y	n	y	y	y
y	n	y	n	y	y	y
n	n	n	n	n	n	y
y	y	n	n	y	y	y
y	y	y	y	y	y	y
n	y	y	n	n	n	n

Key:
y = yes; feature is included
n = no; feature is not included

Chart 6. Inquiry Functions

	A) Read and search						Miscellaneous			
PRODUCT NAME	LAST RECORD	PREVIOUS RECORD	NEXT RECORD	SEARCH FOR RECORD NUMBER	SEARCH WITH FIELD	COMPARE	B) USES WILD CARDS	C) READ AND PROCESS DATA	D) BUILT-IN HELP SCREEN	D) HELP SCREEN USER-DEFINED
For the Apple:										
Quick File III	y	y	y	n	y	y	y	y	y	n
Versa Form	y	y	y	y	y	y	y	n	y	n
Complete Mailing Label and Billing System	n	n	n	y	n	y	n	n	y	n
Information & Data Master	y	y	y	y	y	y	y	n	n	n
Datadex	y	y	y	y	n	n	n	y	n	n
Data Perfect	y	y	y	y	y	n	y	y	n	y
The Data Machine	n	n	y	y	y	n	y	y	y	n
List Maker	y	y	y	y	y	n	n	n	y	n
The General Manager	n	n	y	n	y	y	y	y	y	y
List Handler	n	y	y	y	n	n	y	n	y	n
PFS:File	y	y	y	n	y	y	y	y	n	y
IFO-Version II	y	y	y	y	y	y	y	n	y	n
DB Master	y	n	y	y	y	n	y	n	y	n
Systems II	n	n	n	y	y	y	n	y	n	n
For CP/M computers:										
dBASE II	y	y	y	y	y	y	y	y	n	n
Data Plan	y	y	y	n	y	n	y	y	y	n
Microbase	n	n	y	n	y	y	n	n	y	n
FMS-80	y	y	y	y	y	n	y	y	y	y
Friends Report Writer	n	n	n	n	y	y	n	y	y	y
MDBS	y	y	y	y	y	y	y	y	n	y
Selector	y	y	y	n	y	y	n	y	y	n
MAG/Base	y	y	y	n	y	y	y	n	y	n
Infostar	n	y	y	n	y	y	n	y	y	n
The Knowledge Manager	y	y	y	y	y	y	y	y	n	y
MicroRIM	n	n	n	n	y	y	n	y	y	n
Personal Pearl	y	y	y	y	y	y	y	y	y	y
The Quad	y	y	y	y	y	y	n	y	y	n
Analyst	y	y	y	y	y	y	y	y	n	n
For other machines:										
MIS Builder	n	n	y	y	y	y	n	y	y	n
Corp	n	n	n	n	y	n	y	y	n	n
Wordbase	y	y	y	y	y	y	y	n	y	y
Waloss II	y	y	y	y	y	y	y	n	n	n
ANDI	n	n	y	y	y	n	y	y	y	y
Data Design	y	y	y	y	y	y	y	y	y	n
IDM-X	y	y	y	y	y	y	n	y	y	n
Control	n	n	n	y	y	y	y	y	n	y
The Organization Analyst	n	n	n	y	y	y	y	y	y	n
Savvy	y	y	y	y	y	y	y	y	y	y
Personal Data Base	n	y	y	y	y	n	y	n	y	n
The Wiz	y	n	y	y	y	y	y	y	y	n
File/idea	y	y	y	y	y	n	y	n	y	n
Data Manager	n	n	y	n	y	y	n	y	y	n

(A) The ways a user can inquire of the database vary between systems. Ideally, a user should have as many ways as possible to "move around" in the database, while searching for information. If a system has only a limited search capability, the user may well become frustrated in an attempt to locate information within the database.

(B) A wild card, like CP/M's "*" allows a user to search for records when only incomplete information exists. For example, if all persons in a customer file with last names beginning with "S" were desired, the wild card could be used to specify them as follows: "S*".

(C) The ability of the inquiry function to read and simultaneously process data can be extremely valuable in creating systems that minimize storage space requirements. A code or abbreviated key for a field may be used in place of a full data item and the correct data item interpreted or calculated at inquiry time. A practical use of this type of "read and process" data function occurs when "multilayer" prices are used in an inventory file. Rather than store each price, the system may store only one price, with several discount percentages. Upon inquiry, the system will calculate dollar prices for user viewing.

(D) Built-in help screens allow the user to use and access the database system without constant reference to the instruction manual. The ability of a user to create his own help screens is an important part of a fully "user-friendly" customized application package.

Key:
y = yes; feature is included
n = no; feature is not included

Chart 7. Input Edit Functions

	UNUSUAL INPUT EDIT ATTRIBUTES (TESTS FOR FORM)						CONDITIONAL LOGIC			
							NUMERIC		ALPHA	
PRODUCT NAME	TIME	JULIAN DATE	SOCIAL SECURITY #	TELE-PHONE #	WHOLE #	DOLLARS & CENTS	IF/AND/OR	RANGE	IF/AND/OR	RANGE
For the Apple:										
Quick File III	y	n	n	n	y	y	y	y	y	y
Versa Form	n	n	y	y	y	y	n	y	n	y
Complete Mailing Label and Billing System	y	n	y	y	y	y	n	n	n	n
Information & Data Master	y	y	y	y	y	y	n	n	n	y
Datadex	n	n	n	n	n	n	n	n	n	n
Data Perfect	y	y	y	y	y	y	n	n	n	n
The Data Machine	n	n	n	n	n	n	y	y	y	y
List Maker	y	n	n	n	n	n	n	n	n	n
The General Manager	n	n	n	n	n	y	n	n	n	n
List Handler	n	n	n	n	y	y	y	y	y	y
PFS:File	y	n	y	y	y	y	y	y	y	y
IFO-Version II	y	n	y	y	y	y	y	y	y	y
DB Master	n	n	n	y	y	y	n	y	n	n
Systems II	y	n	y	y	y	y	n	y	n	y
For CP/M computers:										
dBASE II	y	y	y	y	y	y	y	y	y	y
Data Plan	n	n	y	y	y	n	n	n	n	n
Microbase	y	y	y	y	y	y	y	y	y	y
FMS-80	y	y	y	y	y	y	y	y	y	y
Friends Report Writer	y	n	y	y	y	y	y	y	y	y
MDBS	n	y	n	n	n	n	n	y	n	y
Selector	y	n	y	y	y	y	y	y	y	y
MAG/Base	y	y	y	y	y	y	y	y	y	y
Infostar	y	y	y	y	y	y	y	y	y	y
The Knowledge Manager	y	n	n	n	y	y	y	y	y	y
MicroRIM	y	y	y	y	y	y	y	y	y	y
Personal Pearl	y	y	n	y	y	y	y	y	y	y
The Quad	n	n	n	n	y	y	n	y	n	n
Analyst	y	y	y	y	y	y	y	y	y	y
For other machines:										
MIS Builder	n	n	n	n	n	n	y	y	y	y
Corp	y	y	y	y	y	y	y	y	y	y
Wordbase	n	n	n	n	n	n	y	y	y	y
Waloss II	n	y	n	n	y	n	n	y	n	y
ANDI	n	n	n	n	n	n	n	y	n	n
Data Design	n	y	y	y	y	y	n	y	n	y
IDM-X	n	y	n	n	y	y	n	y	n	y
Control	n	n	n	n	y	y	y	y	y	y
The Organization Analyst	y	y	y	y	y	y	y	y	y	y
Savvy	y	y	n	n	y	y	n	n	n	n
Personal Data Base	y	n	y	y	y	y	n	n	n	n
The Wiz	y	y	y	y	y	y	y	y	y	y
File/idea	y	n	n	y	y	y	n	n	n	n
Data manager	y	n	y	y	y	y	n	y	n	y

Many programs offer special "tests" that check for data format, data range, or data content. A few programs even allow for "conditional logic" of the order "if/and/or" for user-specified parameters. These extra "tests" are an integral part of an error-free database program application, and become even more critical in a multi-user complex environment.

Key:
y = yes; feature is included
n = no; feature is not included

SELECCION DE UNA MICROCOMPUTADORA

-¿Se justifica comprarla?

BENEFICIOS QUE PUEDE TRAER.

* Manejo eficiente de información
  - reporte semanal o diario
  - información actualizada
  - facilita toma de desiciones

* Reducción de costos
  - Reduce inversiones en inventarios
  - Aprovisionamiento óptimo
  - Ordenes de compra automáticos
  - Evita servicios periódicos de contador

* Aumenta rendimiento
  - Genera facturas rapido
  - Estatus de pago de clientes
  - Clasificación o historial de clientes
  - Análisis detallado de ventas

* Competir mejor en el mercado

- Facilita:

  incrementar productividad
  mejorar eficiencia
  mejorar atención a clientes
  aumentar confiabilidad
  mejorar precios de venta
  ampliar rangos de credito

Aspectos importantes en Selección

- Costo total hardware
- Costo total software de sistema
- Costo total software de aplicación
- Costos adicionales o imprevistos
- Costo de instalación
- Mantenimiento
- Detallada clarificación de como el sistema propuesto cumple los requerimientos
- Tiempos de entrega
- Plan de Implementación, debe cubrir
    - Firma del contrato
    - Desarrollo o adaptaciones de los programas de aplicación
    - Instalación
    - Entrenamiento personal
    - Pruebas de calidad
    - Documentación amplia y clara.

CRITERIOS DE SELECCION

Guia para la Evaluación de Propuestas

- Demostración de operación
- Analizar comentarios y experiencia de usuarios de equipo idéntico
- Crecimiento futuro del sistema
- Relación mantenimiento vs. horas de trabajo.
- Procedimientos de recuperación en casos de falla.
- Si las fallas son muy frecuentes saber de antemano si es preferible duplicar el equipo o cubrir un nivel de mantenimiento más amplio.
- Eventos de emergencia
- ¿historial de confiabilidad del equipo?
- ¿Instalaciones especiales?
- Mantenimiento del software
- Adaptación de los paquetes de aplicación si son estandar

- Experiencia del vendedor en:
  - . area de computadoras
  - . industria particular del cliente
- Claridad de la garantia
- Servicio en los lugares de operación
- Opciones de financiamiento del sistema
- ¿Usa lenguage estandar en software de aplicación?
- Gastos aparte del equipo:
  - * Honorarios consultor
  - * Entrenamiento personal
  - * Nuevos procedimientos
  - * Captura de datos
  - * Conversión y gastos paralelos
  - * Refacciones
  - * Seguro
  - * Gastos legales
  - * Adptación instalaciones

CRITERIO DE SELECCION

- Responde a las especificaciones
- Operación a niveles adecuados
- Expansión futura
- Confiabilidad del fabricante y vendedor
- Evaluación según la guia de selección

Aplicaciones

- Instrumentación y control
- Telecomunicaciones y transmisión de datos
- Medicina
- Sistemas de Información, procesamiento de datos
- Educacionales
- Personales y caseras
- Juegos y entretenimientos

Instrumentación y Control

* Aplicaciones Industriales
* Control de procesos
* Control de calidad
* Control numérico en Maquinas herramientas
* Control directo de máquinas
* Control y manejo de líneas de producción
* Robots
* Pruebas automáticas e inspección
* Adquisición de datos remotos
* Control y análisis de experimentos de laboratorio
* Control de tráfico
* Control automático de barcos, trenes, aviones
* Automóviles
* Instrumentos de medición
* Terminales inteligentes
* Periféricos

Aplicaciones Industriales

- Generación, transporte y distribución de energía eléctrica.
- Industria textil
- Industria del cemento
- Industria del vidrio
- Industria del acero, metal, peletizadoras
- Minas
- Industria petroquímica
- Oleoductos y gasoductos
- Pulpa y papel
- Industria Llantera
- Transporte y distribución de agua
- Control automático de trenes rápidos y subterraneos
- Producción de alimentos

Telecomunicaciones y transmisión de datos

* Telemetria
* Switcheo de mensajes (telefonos)
* Controladores de comunicaciones para grandes computadoras
* Concentradores de líneas de comunicación
* Redes de computadoras
* Control de comunicaciones via Satelite y microndas.
* Control de tráfico de comunicaciones
* Comunicaciones via fibras ópticas

Aplicaciones Médicas

* Monitoreo de presión y flujo sanguíneo
* Electrocardiografía
* Aparatos de soporte para sordo-mudos y ciegos
* Facilidades a menos-validos, manejo de articulaciones, protesis
* Tomografía computarizada
* Control y disminución de radiaciones
* Sistemas expertos, diagnosticos automáticos
* Procesamiento de señales biomédicas
* Procesamiento de imágenes, radiografias
* Modelado de sistemas fisiológicos, psicológicos, etc.
* Instrumentación biomédica

Sistemas de Información

* Manejadores de bases de datos
* Inventarios
* Manejo de clientes y/o provedores
* Nominas y pagos
* Adquisiciones
* Sistemas gerenciales
* Procesadores de palabra
* Editores inteligentes
* Buzones electrónicos
* Indices, catálogos y directorios
* Graficadores
* Formateadores de texto
* Capturadoras de datos
* Acceso a bancos de información

- Educacionales

* Instructores de personal
* Simuladores
* Educación Infantil
* Enseñanza ayudada por computadora
* Facilidades para la generación de audiovisuales
* Facilidades para la generación de cursos
* Evaluación de alumnos
  - En línea
  - Fuera de línea

Juegos y Entretenimientos

* Juegos de video
  - ping-pong
  - futbol, etc.

* Juegos de azar
  - generador de números aleatorios

Aplicaciones personales y caseras

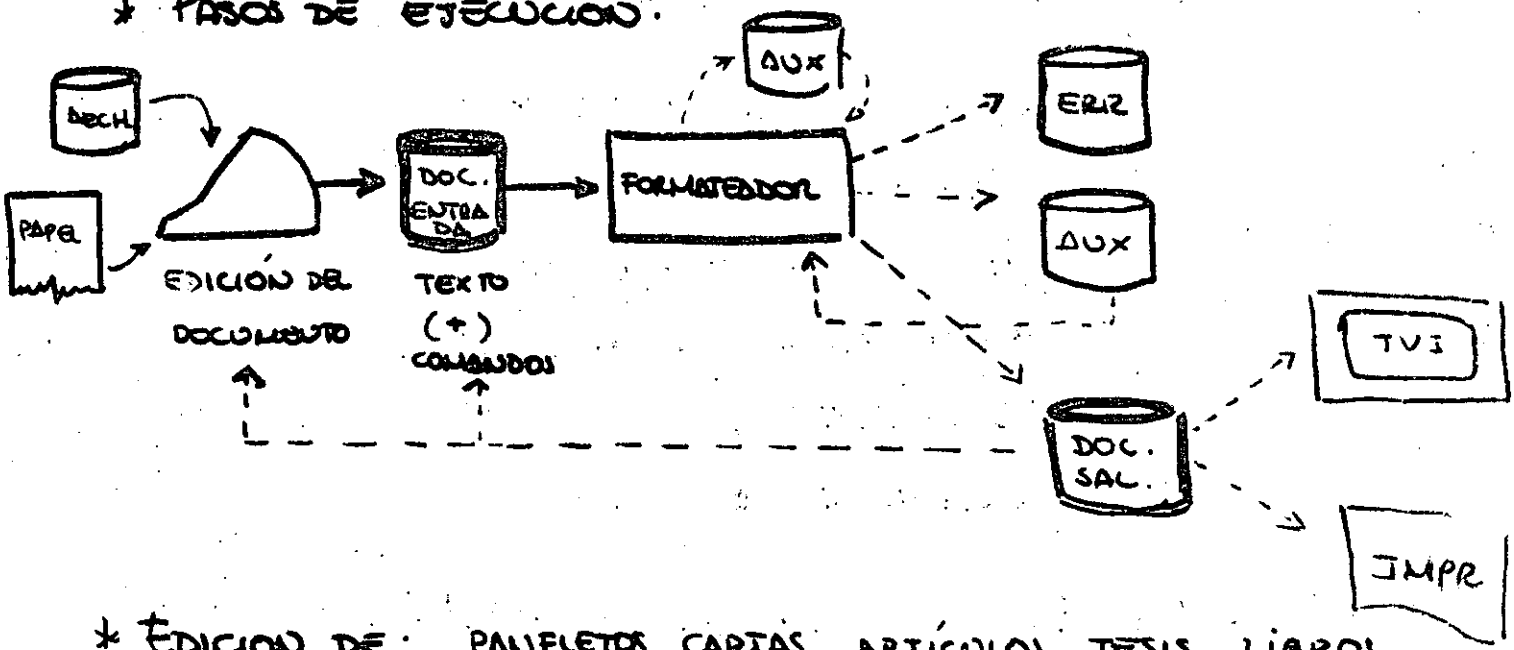
* Discador de telefonos automático
* Recordador de citas y compromisos
* Control de equipo casero
    - lavadoras de ropa y platos
    - Estufas
    - Hornos de micro-ondas
    - Aspiradoras automáticas
    - Televisores
    - Secadores de ropa
* Recibidor de llamadas telefónicas
* Manejar dispositivos a distancia (por telefono)
* Control de puertas y ventanas contra ladrones
* llamador automático a bomberos, dueño, policia, etc.
* Agenda computarizada

APLICACIÓN: WORD PROCESSING

* PRODUCCIÓN DE DOCUMENTOS DE MEDIANA A ALTA CALIDAD.

* REQUIERE DE: EDITOR (PANTALLA) Y FORMATEADOR DE TEXTOS (FORMATTER II, SCRIBE etc.).

* PASOS DE EJECUCIÓN.



* EDICIÓN DE: PANFLETOS, CARTAS, ARTÍCULOS, TESIS, LIBROS, REPORTES, MACHOTES, ETC.

* CARACTERÍSTICAS PRINCIPALES.

- SELECCIÓN DE TIPO DE DOCUMENTO (DEFAULT).
- CREACIÓN DE UN NUEVO TIPO DE DOCUMENTO
- MODIFICACIÓN A UN TIPO DE DOCUMENTO ESTANDARD.
- PARRAFEO Y SILABEO AUTOMÁTICOS
- CONTROL (AUTOMÁTICO O NÓ) DEL ESTILO DE LA HOJA.
  - SANGRÍAS
  - MÁRGENES SUP., INF., IZQ Y DER.
- MANEJO DE ENCABEZADOS Y PIES DE HOJA (PÁGINA), ASÍ COMO DE LA NUMERACIÓN

-- CONTINUAN LAS CARACTERISTICAS --

- MANEJO AUTOMÁTICO DE REGIONES DE TEXTO
  - NUMERACIÓN DE PÁRRAFOS
  - MARCAS EN LOS
  - RESPETO DE ZONAS DE TEXTO
  - FORMATEO ESPECIAL DE Z. DE TEXTO.

- MANEJO DE NUMERACIONES (CONTADORES)
  - DE CAPÍTULO, DE SECCIONES, SUBSECCIONES, PÁRRAFOS, ETC.
  - FIGURAS, TABLAS
  - PALABRAS, LETRAS Ó SIGNOS.

- MANEJO DE ROMPIMIENTOS:
  - DE PALABRAS
  - DE LÍNEAS
  - DE PÁGINAS
- MANEJO DE ESPACIOS:
  - LIBRES
  - FIGURAS Y TABLAS (CREACIÓN).
  - PIES DE PÁGINAS
- "FACILIDADES" PARA CREACIÓN DE TABLAS, FIGURAS Y DIBUJOS.

- MANEJO DE REFERENCIAS:
  - CRUZADAS, PÁGINAS, CAPS., SECS, MARCAS EN LOS TEXTOS.
  - BIBLIOGRÁFICAS.

- SELECCIÓN DE TIPOS ("FONTS"):
  - NEGRITAS, CURSIVAS, ITALICAS, SUBRAYADAS, GÓTICAS, MAYÚSCULAS ETC
  - SÍMBOLOS MATEMÁTICOS Y DE ESCRITURA (CORCHETES).
- CREACIÓN DE NUEVOS "FONTS".

(TEX y METAFONT - D. E. KNUTH)

SOFTWARE DE APLICACION

- Multiplan — tablas (spreadsheet) 63 col. x 255 rengl.
- Paquete administrativo — Administración del tiempo
  " " de proyectos
  " " personal
- Visicalc — tablas
- Visifile — manejador de archivos
- Visidex — Organiza información del personal, alerta y avisa eventos
- Visiplot — Analisis Financiero
- Visitrend/plot — Captura, transforma y proyecta series de tiempo (Ruta Critica) Grafica resultados.
- Visischedule — Planea desarrollo de proyectos, citas, calendarización
- Visiterm — Comunica Apple II con otras comput. transferencia de archivos

- DesKtop/PLAN — Modelo de planeación financiera
- Wordstar — Procesador de palabra
- spellstar — Corrector de ortografía
- Mail/merge — Mezclador de archivos de propósito múltiple
- Datastar — Captura, recuperación y actualización de datos, facilita el diseño del formato y pantalla.
- supersort — Sortea y mezcla datos a alta velocidad
  Maneja 32 archivos simultaneamente
  velocidad 560 records/minuto
- WordMaster — Editor de texto
- Calcstar — Tablas y modelado financiero
- Infostar — Generador de reportes
- Contabilidad — Retail Science Inc.
  - Gerencia
  - Cantidades recibidas
  - Cuentas por pagar
  - n por cobrar
- Power text — procesador de palabra
- dBASE II — Manejador de base de datos

Paquetes de Aplicacion en México

- Contabilidad
- Nominas
- Inventarios
- Cuentas por cobrar
- Sistema de producción
- Presupuesto y control de obra
- Fraccionamientos y venta departamentos
- Seguros y Aseguradoras
- Farmacias
- Estimaciones y aprovisionamiento de insumos.

MICROPROCESADORES Y MICROCOMPUTADORAS

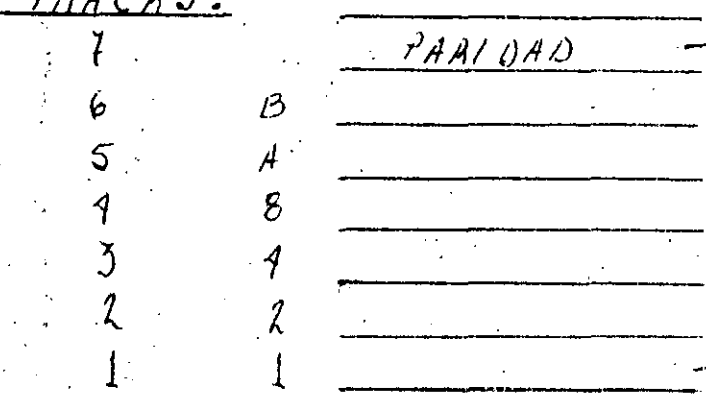
A N E X O S

M.EN I. LUIS H.PEÑARRIETA E.

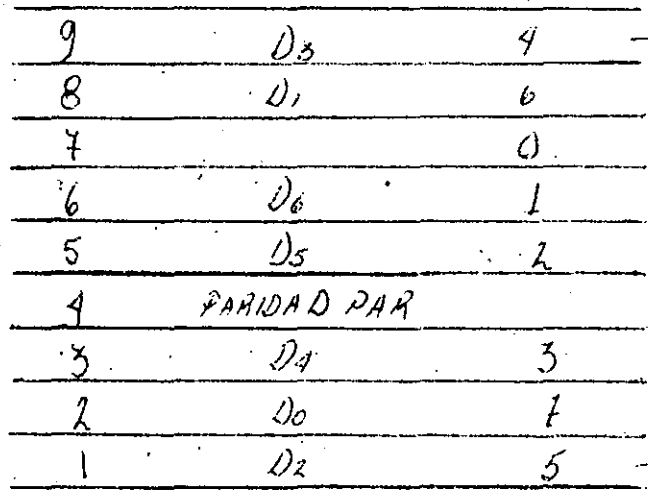
NOV. 1984

ORGANIZACION DE LOS BITS EN LAS CINTAS

7 TRACKS:



9 TRACKS:



ASCII

$D_7\ D_6\ D_5\ D_4 \quad D_3\ D_2\ D_1\ D_0$

$1\ 0\ 0 \quad 0\ 0\ 0\ 1 \quad = 41_H = A$

MSb LSb

EBCDIC

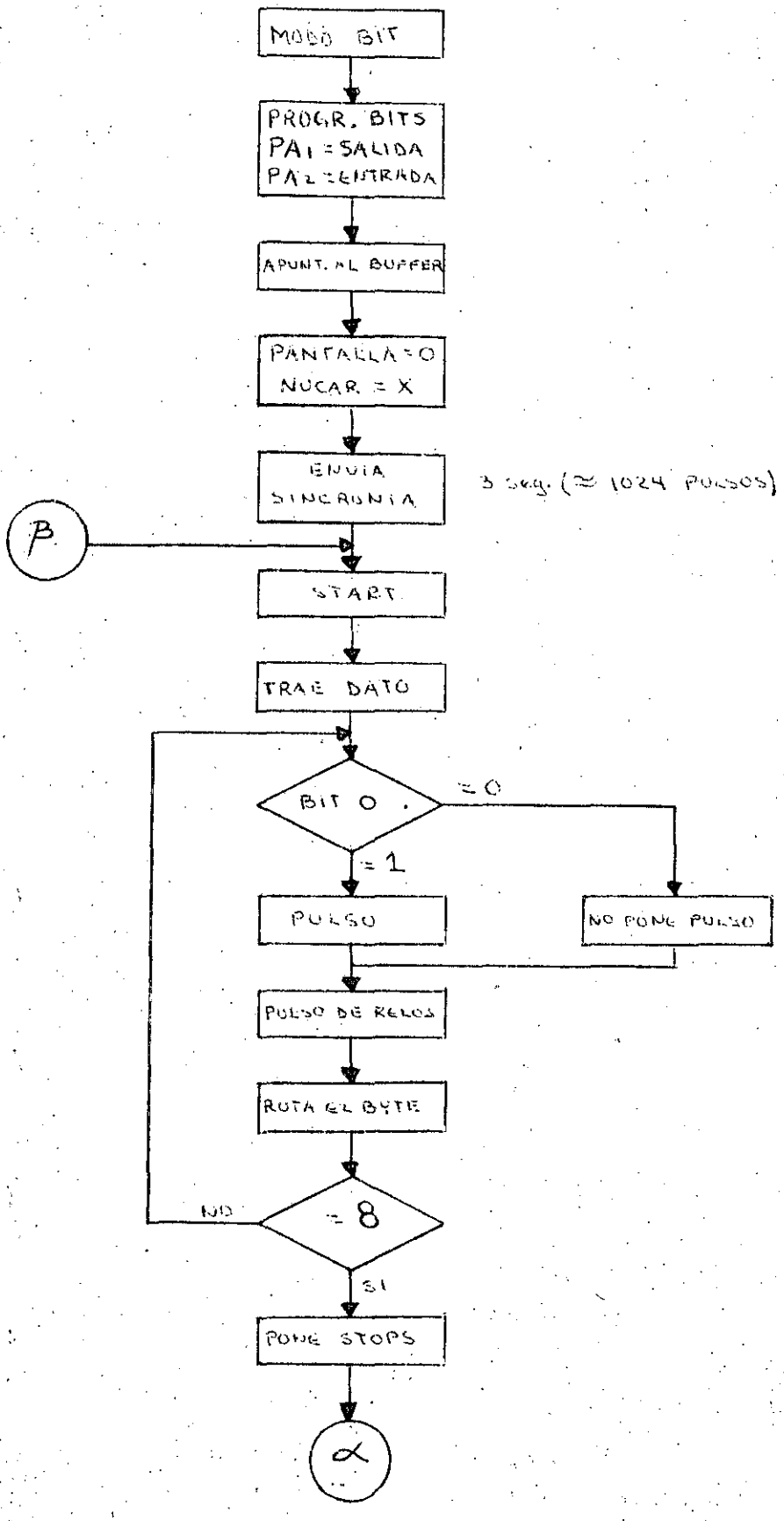
$0\ 1\ 2\ 3\ 4\ 5\ 6\ 7$

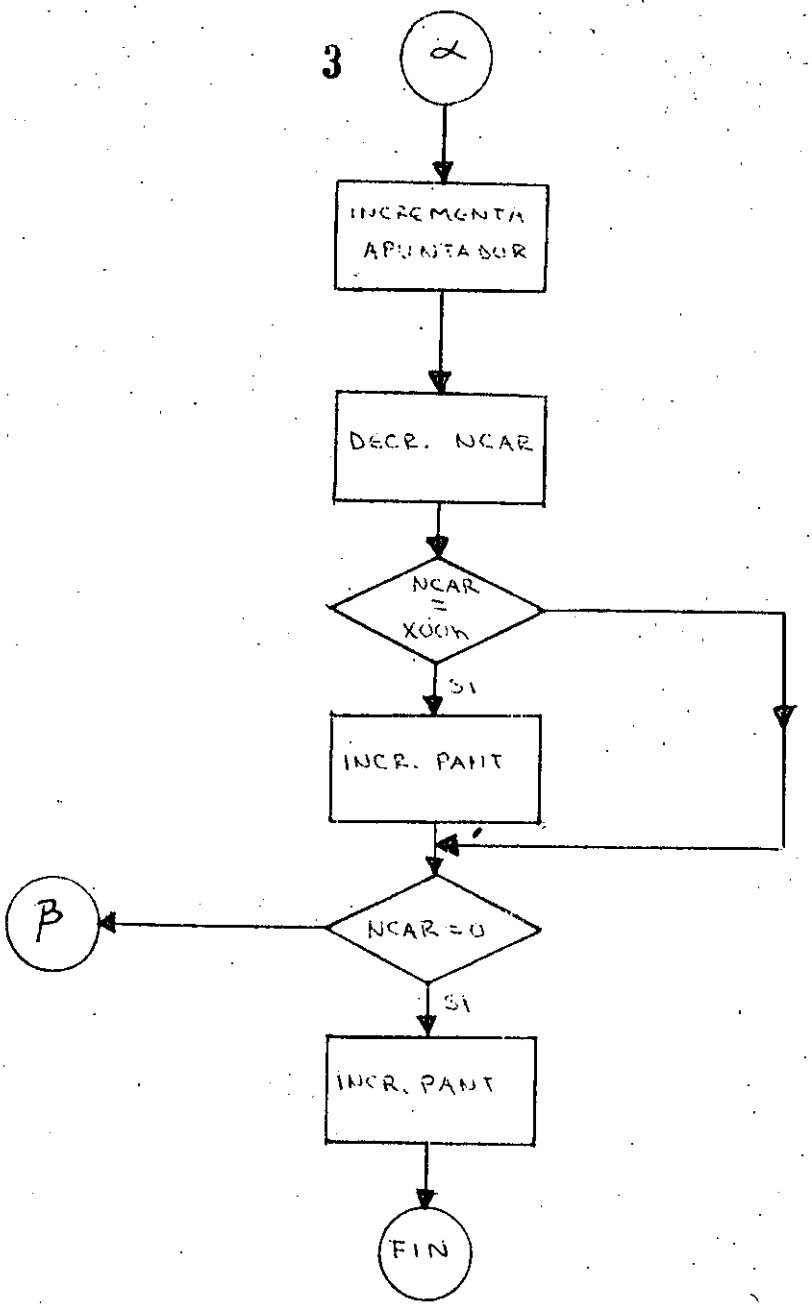
$1\ 1\ 0\ 0 \quad 0\ 0\ 0\ 1 = C1_H = A$

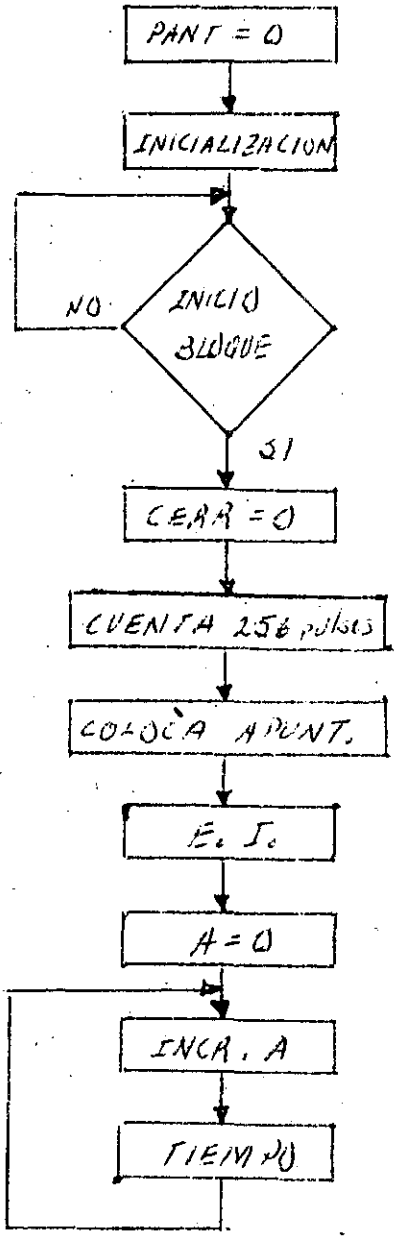
MSb LSb

GRABADORA DE CASSETTE

GRABACION

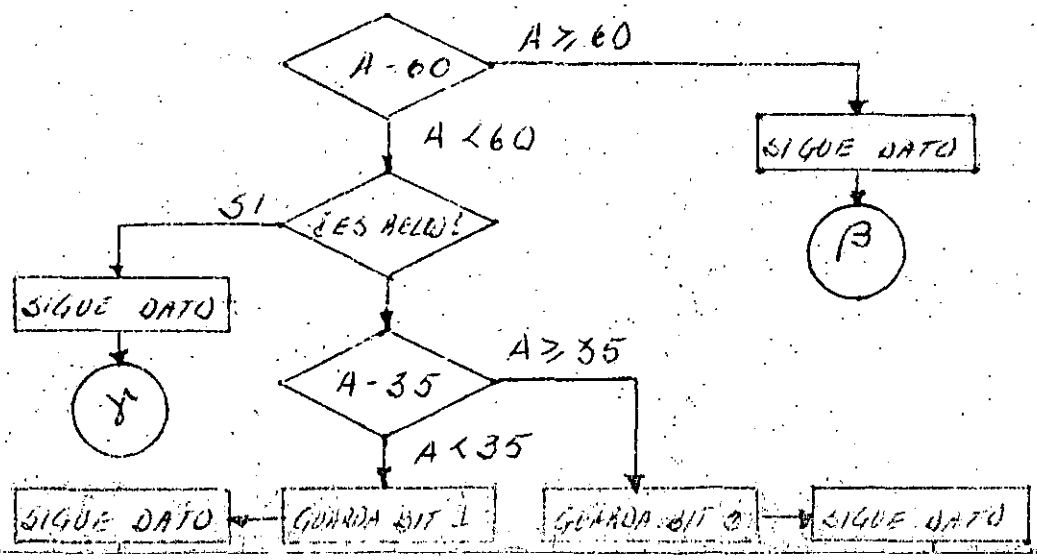


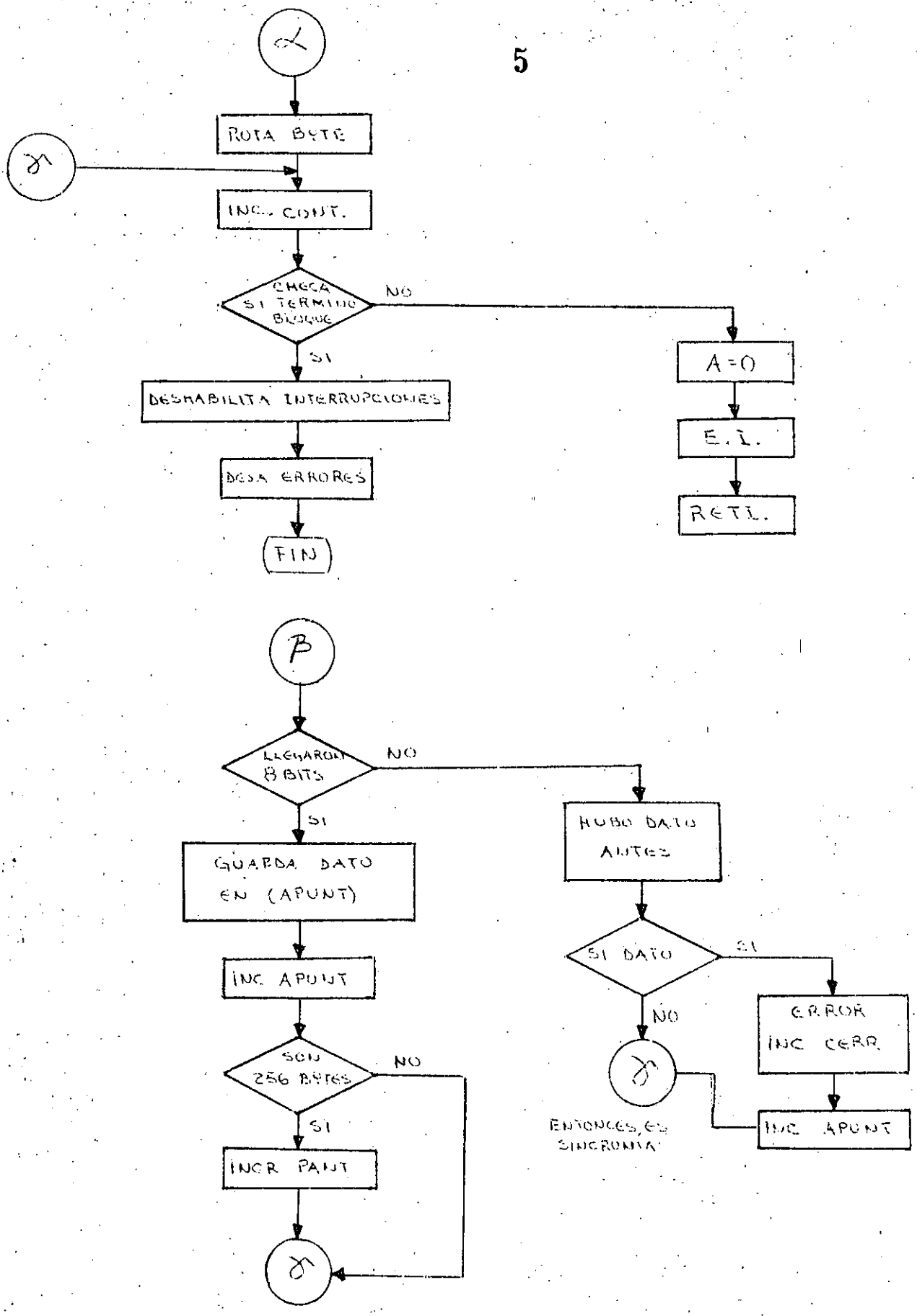

α
INCREMENTA APUNTADOR
DECR. NCAR
NCAR = XOOH
SI
INCR. PANT
β
NCAR = 0
SI
INCR. PANT
FIN



EN ESTE CASO SOLO CUENTA 256 PULSOS, (ASUMIENDO QUE SON PULSOS DE BASURA Y DE SINCRONIA), NO DETECTA PRECISAMENTE LA SINCRONIA

RUTINA DE ATENCION DE INTERRUPCIONES




α
ROTA BYTE
γ
INC. CONT.
CHECA SI TERMINO BLOQUE
NO
SI
A=0
E.I.
RETI.
DESHABILITA INTERRUPCIONES
BORRA ERRORES
FIN
β
LLEGARON 8 BITS
NO
SI
HUBO DATO ANTES
GUARDA DATO EN (APUNT)
SI DATO
SI
NO
ERROR
INC CERR
INC APUNT
SON 256 BYTES
NO
SI
ENTONCES, ES SINCRONIA
INCR PANT
γ

CODIFICACION EN GRUPO

Expande a 5 bits cada 4 bits secuenciales que le llegan.
Dos reglas importantes:

1. No dejar más de 2 ceros internos juntos.
2. No colocar más de 1 cero a los extremos

00000		10000	
00001		10001	
00010		10010	cumple
00011		10011	cumple
00100		10100	
00101		10101	cumple
00110		10110	cumple
00111		10111	cumple
01000		11000	
01001	cumple	11001	cumple
01010	cumple	11010	cumple
01011	cumple	11011	cumple
01100		11100	
01101	cumple	11101	cumple
01110	cumple	11110	cumple
01111	cumple	11111	cumple

De las 32 opciones 17 cumplen con las 2 reglas, se requieren 16, por lo tanto, sobra 1.

PERIFERICOS DE ENTRADA Y SALIDA

1. MANEJO DEL TECLADO

Los teclados son mallas de conductores que normalmente no tienen contacto entre si, sin embargo, al oprimir una tecla los conductores que cruzan por la tecla hacen contacto entre si. Esto produce un código diferente cada vez que se oprime una tecla distinta. Los teclados pueden entregar la información en diferentes formas, bien sea en paralelo o en serie. Si es en paralelo puede ser el código de la malla, o un código estandar como ASCII, EBCDIC, etc., y si es en serie es un código estandar. Características más comúnes de los teclados:

- Envian los datos en código ASCII.
- Pueden enviar la información en serie (RS232-C) o en paralelo manejando una señal de control (strobe) que indica el momento en que se oprime una tecla, esta señal de control puede ser un pulso o un nivel.
- Manejan un bit de paridad en la transferencia de los datos.
- Manejan la tecla de control que sirve para manejar las teclas alfabeticas en la zona de control del código ASCII.
- Manejan mayusculas y minusculas por medio de la tecla "shift".
- Manejan una o dos teclas de cambio de estado, la tecla de "shift lock" que sirve para cambiar a la segunda alternativa de todas las teclas y la tecla de "alfa lock" que cambia a la segunda alternativa solamente las teclas alfabéticas.
- Manejan teclas especificas para el movimiento del cursor en la pantalla.
- Manejan una tecla de repetición, que sirve para enviar continuamente un código cualquiera mientras se opriman, en forma simultanea, la tecla correspondiente y la tecla de repetición.

Existen dos maneras para que el procesador pueda atender al teclado. Por consulta repetitiva (polling) o por interrupciones.

- CONSULTA REPETITIVA. En este caso el procesador está continuamente preguntando al(los) teclado(s) en que momento se oprime una tecla, el gran inconveniente de este enfoque es que el procesador se mantiene ocupado durante el tiempo en que espera que alguien oprima una tecla, sin embargo, la programción y operación es muy simple.

- INTERRUPCIONES. En este caso el teclado le indica al procesador el momento en que alguien oprime una tecla, de esta manera el procesador puede estár dedicado a otras actividades sin preocuparse directamente del teclado, sabiendo que en el momento en que alguien oprima una tecla será interrumpido por el teclado, el cual le indicará que tecla se oprimío. Es preciso reconocer que este enfoque es más complejo de programar que el anterior y requiere más componentes de hardware.

2. GENERADOR DE VIDEO

El generador de video es un dispositivo que controla el despliegue de caracteres en una pantalla de video. Para esto requiere generar la sincronía de video del CRT (tubo de rayos catódicos), manejar una memoria de video o refresco la cual se accesa a muy alta frecuencia ya que todos los puntos de la pantalla deben refrescarse uno por uno 60 veces por segundo y finalmente, mostrar los caracteres que se deseen formateados en una matriz de puntos.

3. MANEJO DE TERMINAL DE VIDEO

- Acoplamiento RS232C.

- Manejo de terminal, polleo o interrupciones.

4. MANEJO DE IMPRESORA

- Conección serie.

- Conección paralela, interfase centronix.

9

5. CONVERSOR ANALOGICO-DIGITAL A/D

- Operación.
- Adquisición de datos analógicos.

6. CONVERSOR DIGITAL-ANALOGO DAC

- Operación.
- Despliegue de imágenes almacenadas en memoria.
- Conversión de otro tipo de información analógica.

signal data. It is not the case, however, that one state of saturation signifies a 1 and the other a 0. More complex schemes are used, in part to introduce a periodic variation in the readout current. Such periodicities are useful as synchronizing signals in computer systems. In part the schemes are used because, as I have noted, the readout of the data depends on placing a coil in the presence of a changing magnetic field and not a field that is constant. In a code called double frequency modulation a 1 is represented by a reversal of magnetization and a 0 by the absence of a reversal. An additional reversal is inserted between each bit to provide a timing signal. The encoding requires a maximum of two reversals per bit. Other codes do better: they require fewer reversals. Such codes, however, are more susceptible to error, so that part of the extra capacity must be used for extra bits that serve for error correction.

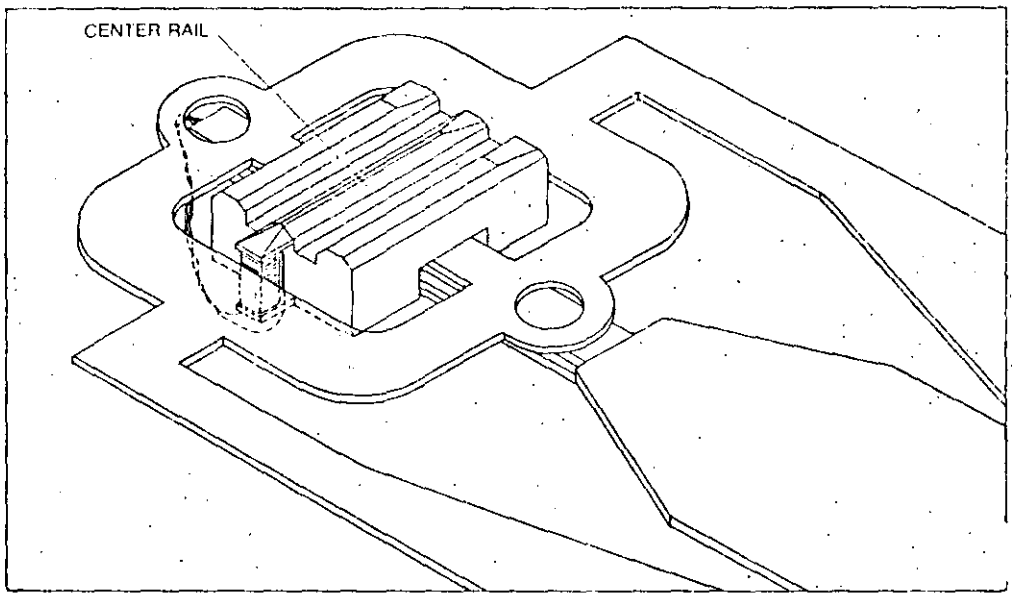
The storage of a document will provide an example of the simplest such scheme. In the American Standard Code for Information Interchange each character in a language is represented by seven bits. The letter *A*, for instance, is 1000001. An eighth bit is often added to each character in storage as a "parity" bit, or check bit, to aid in determining whether the preceding seven bits are correct. The value of the parity bit is 0 if the preceding seven bits add up to an odd number and 1 if they add up to an even number. Thus the letter *A* in storage is 10000011.

This use of parity bits can aid in locating an error only to within the preceding seven bits. A more complex code developed by R. W. Hamming in 1950 employs a greater number of bits and yields the precise address of a single bit that is in error. The correction of the error then requires simply the conversion of a 1 into a 0, or the reverse. In a still more complex scheme the data bits are treated as the coefficients of a polynomial. The polynomial is manipulated algebraically to yield a smaller set of bits. These bits are put in storage. In the event of an error they can be called up to reconstruct the data. The last scheme is particularly well suited to the correction of bursts of errors, which is generally the way errors develop on a magnetic disk.

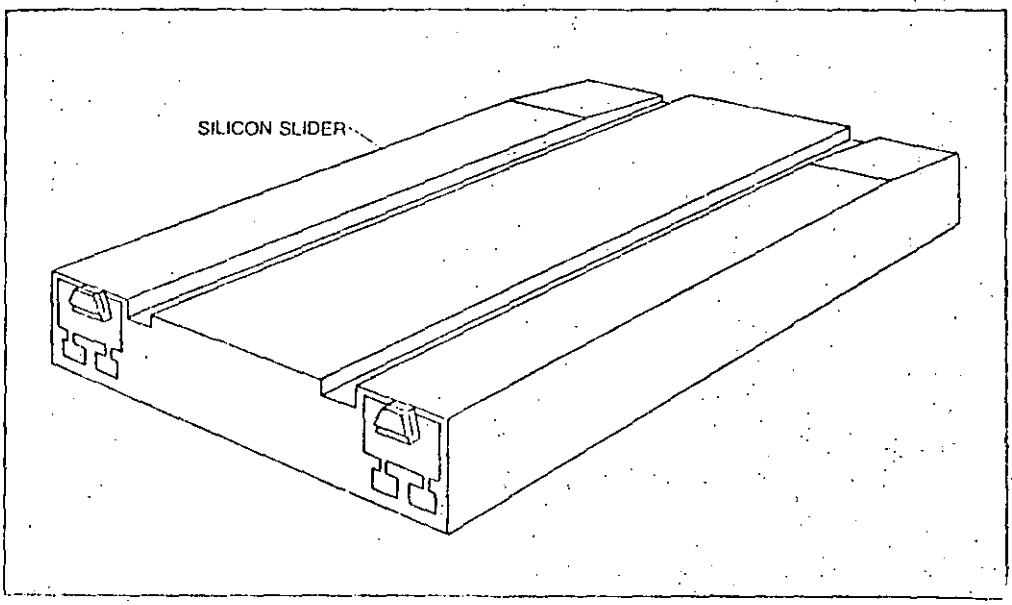
For the magnetic material that is the core of the electromagnet in the head the requirements are distinctive. Here a small flow of current through the coil around the core should yield a large magnetization, and when the flow of current stops, the magnetization should return as nearly as possible to zero. Moreover, a reversal of the direction of the flow of current to only a modest value should yield a reversal of the magnetization. In many heads the core is a ceramic consisting of spherical ferrite particles. A ferrite is an oxide of iron together with another metal or a mixture of metals. In the case of a magnetic head the metals are usually nickel and zinc; sometimes manganese is added.

The design of the head must conform to the design of the disk. In one technology the disk is "floppy": it is a thin sheet of Mylar plastic on which the gamma form of iron oxide is coated. The standard diameter of the disk is eight inches, except for minifloppies, in which the diameter of each disk is 5¼ inches. In floppies and minifloppies the head actually makes contact with the surface of the disk. On occasion the head is bounced from the surface by a particle of dirt. Therefore the error rate of the device is relatively high, as is the wear on the medium. Of necessity the disk in such a device spins slowly.

In high-performance memories the magnetic medium is the coating on a rigid aluminum disk eight or 14 inches in diameter, and the head is kept from touching the medium by what is called the air-bearing effect. Consider a head that is nearly in contact with the surface



WINCHESTER HEAD was introduced by the International Business Machines Corporation in 1973. It is shown upside down: actually the rail-like surfaces of the head confront the disk at a distance of half a micrometer. The flow of air under the outside rails generates an aerodynamic force that supports the head; the trailing end of the center rail holds the electromag-

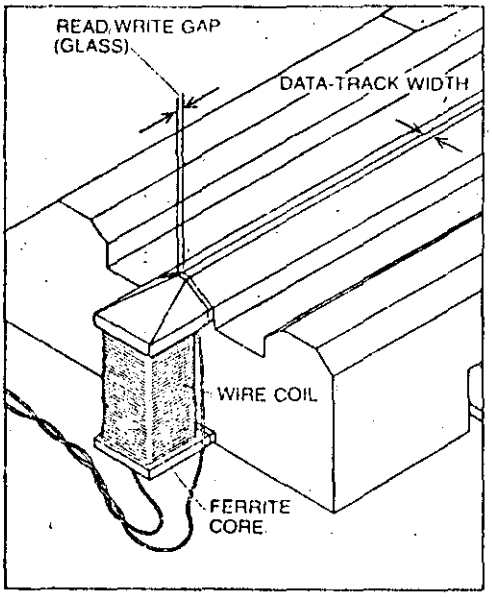


THIN-FILM HEAD (also shown upside down but not at the same scale) has no coil of wire in its electromagnet. Instead it employs a spiral film of electrical conductor. The core of the electromagnet is Permalloy, a mixture of nickel and iron. (The core in a Winchester head is ferrite: an oxide of iron in combination with other metals.) The electromagnet again is at the

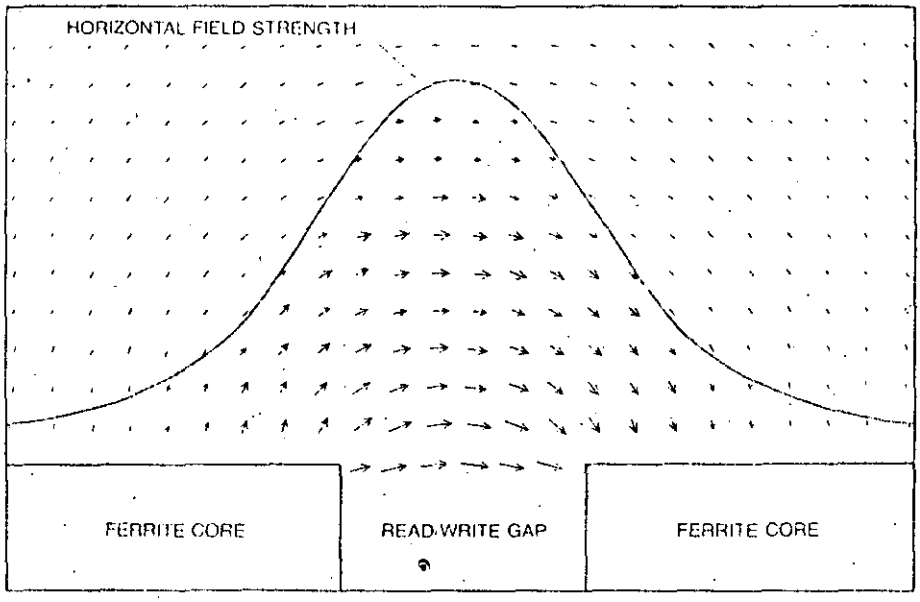
of a 14-inch disk spinning at 3,000 revolutions per minute. The velocity of the head with respect to a data track in the medium on the disk is approximately 100 miles per hour. If the length of the head along the direction of relative motion is two orders of magnitude longer than the separation between the head and the medium, the flow of air between the head and the medium provides support for a head weighing up to several grams.

In 1973 the International Business Machines Corporation introduced the Model 3340 disk memory. The technology of the system has since been adopted by many manufacturers. It is now known generically as Winchester technology, that being the code name under which the device was developed at IBM. A Winchester disk memory has one or more rigid disks, either eight or 14 inches in diameter. Each disk is coated on both sides with a magnetic medium, so that two surfaces per disk are available for the storage of data. Each Winchester head has three rails, or raised surfaces. The trailing end of the middle rail holds a magnetic core with wire coiled around for writing and reading the data. The two outer rails govern the flow of air. The force that results is sufficient to support a weight of 10 grams at a height of half a micrometer above the disk. The disks and the head assemblies in such a memory are sealed in a small "clean room": a chamber approximately the size of a hatbox, in which the air is continuously recirculated and filtered to exclude any dust particles larger than .3 micrometer in diameter.

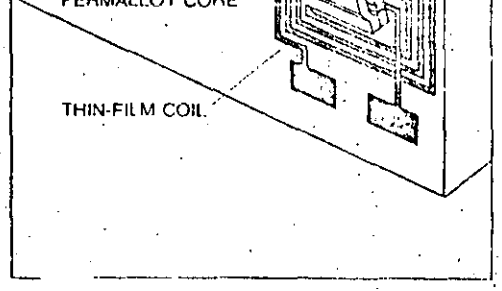
The quantity of data that can be



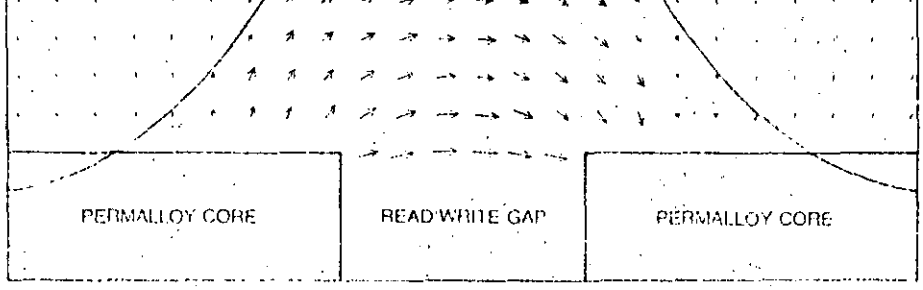
net that writes and reads the data. At the right the electromagnet is seen in closer view. The width of the beveled part of the center rail corresponds to the width of a track of data.



FRINGE FIELD OF WINCHESTER HEAD is the magnetic field that lies outside a gap in the core of the electromagnet. It is the field that writes the data. The arrows show the orientation of the field; their lengths show the intensity. The curve shows the intensity too: it plots the horizontal field strength at a distance from the head equal to half the width of the gap.



ding end of a structure designed to generate a supporting aerodynamic force when a disk is spinning under it. A memory with thin-film heads was introduced this year by IBM.



FRINGE FIELD OF THIN-FILM HEAD decreases steeply at each side of the electromagnet's gap. The steepness of the decrease allows the writing (and subsequent reading) of data at a greater packing density on a disk. Specifically, a Winchester head can record about 10,000 reversals of magnetization per inch along a data track. A thin-film head can record 15,000.

```
LOC   OBJ CODE M STMT SOURCE STATEMENT
                    1
                    2           GLOBAL  PRIN
                    3
                    4
                    5   ;       S U B R U T I N A       I N I C I O
                    6
                    7   ;       Inicializa la operacion del SIO en modo asincrono
                    8
0000  3E04          9   INICIO  LD      A,4
0002  D30D         10           OUT     (0DH),A         ;selecciona REG. 4
0004  3E4C         11           LD      A,4CH
0006  D30D         12           OUT     (0DH),A         ;2 stops, clock x 16, no paridad
0008  C9           13           RET
                   14
                   15
                   16   ;       S U B R U T I N A       T T Y R X D
                   17
                   18   ;       Recibe un caracter de la terminal
                   19
0009  3E03         20   TTYRXD  LD      A,3
000B  D30D         21           OUT     (0DH),A         ;selecciona REG. 3 --> RECEPCION
000D  3EC1         22           LD      A,0C1H
000F  D30D         23           OUT     (0DH),A         ;Rx habilitada, 8 bits
0011  3E00         24   VUEL1   LD      A,0
0013  D30D         25           OUT     (0DH),A         ;selecciona REG. 0
0015  DB0D         26           IN      A,(0DH)         ;lee REG. de lect. 0
0017  CB47         27           BIT     0,A             ;Prueba bit 0 (recepcion de caracter)
                   28                                   ;1 = Z=0, si lleso caracter
                   29                                   ;0 = Z=1, no lleso caracter
0019  28F6         30           JR      Z,VUEL1
001B  DB0C         31           IN      A,(0CH)         ;lee el caracter recibido
001D  325B00   R   32           LD      (DATO),A        ;salva caracter recibido
                   33   ;       LD      A,1
                   34   ;       OUT     (0DH),A         ;selecciona REG. 1 --> ERRORES
                   35   ;       IN      A,(0DH)         ;lee el resistro de errores
                   36   ;       AND     30H
                   37   ;       JR      Z,SIGUE         ;0 si no hay errores
0020  3E18         38           LD      A,18H
0022  D30D         39           OUT     (0DH),A         ;resetea el canal
0024  3E03         40   SIGUE   LD      A,3
0026  D30D         41           OUT     (0DH),A         ;selecciona REG. 3 --> RECEPCION y
                   42                                   ;resetea error
0028  3E00         43           LD      A,0
002A     0D        44           OUT     (0DH),A         ;deshabilita recepcion
002C  C9           45           RET
                   46
```

```
                        47
                        48  ;       S U B R U T I N A       T T Y T X D
                        49
                        50  ;       Transmite un caracter a la terminal
                        51
002D   3E05             52  TTYTXD  LD      A,5
002F   D30D             53          OUT     (0DH),A         ;selecciona REG.5 --> TRANSMISION
0031   3E6A             54          LD      A,6AH
0033   D30D             55          OUT     (0DH),A         ;RTS, Tx habilitada, 8 bits
0035   3E00             56  VUEL2   LD      A,0
0037   D30D             57          OUT     (0DH),A
0039   DB0D             58          IN      A,(0DH)         ;lee REG. de lect. 0
```



```
 LOC   OBJ CODE M STMT SOURCE STATEMENT                      ASM 5.8

003B   CB57             59          BIT     2,A             ;prueba el bit 2 (listo para transmitir)
                        60                                  ;1 = Z=0, si buffer vacio
                        61                                  ;0 = Z=1, si buff. ocupado (en transm.)
003D   28F6             62          JR      Z,VUEL2
003F   3A5B00   R       63          LD      A,(DATO)
0042   D30C             64          OUT     (0CH),A         ;transmite el caracter
0044   3E05             65          LD      A,5
0046   D30D             66          OUT     (0DH),A         ;selecciona REG. 5 --> TRANSMISION
0048   3E00             67          LD      A,0
004A   D30D             68          OUT     (0DH),A         ;deshabilita transmision
004C   C9               69          RET
                        70
                        71
                        72  ;       R U T I N A     P R I N C I P A L
                        73
                        74  ;       Realiza las funciones de un eco con la terminal
                        75
004D   CD0000   R       76  PRIN    CALL    INICIO
0050   CD0900   R       77  OTRO    CALL    TTYRXD
0053   CD0000   R       78          CALL    INICIO
0056   CD2D00   R       79          CALL    TTYTXD
0059   18F5             80          JR      OTRO
                        81
005B   00               82  DATO    DEFB    0
%
```

```
%PRINT SIO.INT.L
```

```
                                SIO.INT                                 PAGE   1
 LOC   OBJ CODE  M STMT SOURCE STATEMENT                                 ASM 5.8

                    1
                    2  DATOA   EQU     0CH             ;puerto A de datos del SIO
                    3  CTRLA   EQU     0DH             ;puerto A de control del SIO
                    4  DATOB   EQU     0EH             ;puerto B de datos del SIO
                    5  CTRLB   EQU     0FH             ;puerto B de control del SIO
                    6
                    7  PARA    EQU     1BH             ;ESCAPE
                    8  CABORT  EQU     21H             ;es admiracion !
                    9  CRET    EQU     0DH             ;retorno de carro
                   10  LINEF   EQU     0AH             ;line feed
                   11
                   12
                   13          GLOBAL   PRIN
                   14  ;       EXTERNAL TECLA
                   15
                   16
0000  2A00      R    17  VECTI   DEFW    INTREC          ;apunta a la rutina de interrupcion
                   18
0002  02            19  INTERR  DEFB    2               ;REG. 2 --> VECTOR DE INTERR.
0003  00            20          DEFB    0               ;vector de interrupcion
                   21
0004  04            22  INIC    DEFB    4               ;REG. 4 --> INICIAL
0005  4C            23          DEFB    4CH             ;2 stops, clock x 16
0006  03            24          DEFB    3               ;REG. 3 --> RECEPCION
0007  C1            25          DEFB    0C1H            ;Rx habilitada, 8 bits
0008  05            26          DEFB    5               ;REG. 5 --> TRANSMISION
0009  6A            27          DEFB    6AH             ;RTS, Tx habilitada, 8 bits
000A  01            28          DEFB    1               ;REG. 1
000B  18            29          DEFB    18H             ;habilita interrup. en Rx
                   30
                   31
                   32  ;       S U B R U T I N A       c o n i n i t
                   33
                   34  ;       Inicializa la operacion del SIO en modo asincrono
                   35
000C  ED5E          36  coninit IM      2
000E  211400    R   37          LD      HL,SIGUE
0011  E5            38          PUSH    HL
0012  ED4D          39          RETI
0014  210000    R   40  SIGUE   LD      HL,VECTI
0017  7C            41          LD      A,H
0018  ED47          42          LD      I,A             ;carga reg. I de interrup.
001A  0E0F          43          LD      C,CTRLB         ;salida, puerto CTRLB
001C     0200    R   44          LD      HL,INTERR       ;a partir de INTERR
001F  0602          45          LD      B,2             ;2 outs
0021  EDB3          46          OTIR                    ;programa interrupcion
```

```
0023   0E0D          47            LD      C,CTRLA
0025   0608          48            LD      B,8
0027      3          49            OTIR                            ;programa el SIO
0029   C9            50            RET
                     51
                     52
                     53  ;         S U B R U T I N A       I N T R E C
                     54
                     55  ;         Recibe un caracter de la terminal
                     56
002A   32AB00    R   57  INTREC    LD      (SALVAA),A
002D   DB0C          58            IN      A,(DATOA)               ;lee el caracter recibido

                                SIO.INT                                 PAGE   2
 LOC   OBJ CODE  M STMT SOURCE STATEMENT                                 ASM 5.8

002F   FE1B          59            CP      PARA                    ;compara con ESCAPE
0031   2808          60            JR      Z,ALTO
0033   32A800    R   61            LD      (TECLA),A               ;salva el caracter recibido
0036   3AAB00    R   62            LD      A,(SALVAA)
0039   ED4D          63            RETI
003B   E1            64  ALTO      POP     HL
003C   22A900    R   65            LD      (GUARDA),HL
003F   00            66            NOP                             ;CALL   stres
0040   214E00    R   67            LD      HL,INTESP
0043   220000    R   68            LD      (VECTI),HL              ;cambia vector de interrupcion
0046   214C00    R   69            LD      HL,ESPERA
0049   E5            70            PUSH    HL
004A   ED4D          71            RETI
                     72
                     73
                     74  ;         S U B R U T I N A       E S P E R A
                     75
                     76  ;         Despues de una ? (detiene acciones) espera otro teclaso
                     77
004C   FB            78  ESPERA    EI
004D   76            79            HALT
                     80
                     81
                     82  ;         S U B R U T I N A       I N T E S P
                     83
                     84  ;         Rut. de interr. que atiende todo teclaso despues de un ESCAPE
                     85
004E   E1            86  INTESP    POP     HL
004F   212A00    R   87            LD      HL,INTREC
0052   220000    R   88            LD      (VECTI),HL              ;repone el vector de interrupcion
0055   DB0C          89            IN      A,(DATOA)               ;lee caracter recibido
0057   FE21          90            CP      CABORT                  ;compara con caracter de aborta
0059   2807          91            JR      Z,ABORTA                ;si son iguales aborta
005B   2AA900    R   92            LD      HL,(GUARDA)
005E   E5            93            PUSH    HL                      ;repone dir. de retorno de la primer interr.
```

```
005F   00          94            NOP                 ;CALL  ldres
0060   ED4D        95            RETI
0062   219400   R  96  ABORTA    LD      HL,PRIN     ;aborta es (PRIN en este caso)
0065   E5          97            PUSH    HL          ;regresa a aborta
0066   ED4D        98            RETI
                   99
                  100
                  101  ;         S U B R U T I N A       c o n i n
                  102
                  103  ;         Recibe el ultimo caracter que se haya apretado en el teclado
                  104
0068   21A800   R 105  conin     LD      HL,TECLA
006B   7E          106           LD      A,(HL)      ;dejo el ultimo dato oprimido en A
006C   3600        107           LD      (HL),0      ;borro, TECLA = 0
006E   C9          108           RET
                  109
                  110
                  111  ;         S U B R U T I N A       c o n o u t
                  112
                  113  ;         Transmite un caracter a la terminal
                  114
006F   32AB00   R 115  conout    LD      (SALVAA),A
0072   3E00        116  VUEL2     LD      A,0

                                SIO.INT                                 PAGE   3
 LOC   OBJ CODE  M STMT SOURCE STATEMENT                                 ASM 5.8

0074   D30D        117           OUT     (CTRLA),A
0076   DB0D        118           IN      A,(CTRLA)           ;lee REG. de lect. 0
0078   CB57        119           BIT     2,A         ;Prueba el bit 2 (listo para transmitir)
                   120                               ;1 = Z=0, si buffer vacio
                   121                               ;0 = Z=1, si buff. ocupado (en transm.)
007A   28F6        122           JR      Z,VUEL2
007C   3AAB00   R 123           LD      A,(SALVAA)
007F   FE0D        124           CP      CRET        ;compara con retorno de carro
0081   200E        125           JR      NZ,SALTA
0083   D30C        126           OUT     (DATOA),A           ;transmite el caracter
0085   3E00        127  VUEL3     LD      A,0
0087   D30D        128           OUT     (CTRLA),A
0089   DB0D        129           IN      A,(CTRLA)   ;lee REG. de lectura 0
008B   CB57        130           BIT     2,A         ;Prueba bit de buffer vacio
008D   28F6        131           JR      Z,VUEL3
008F   3E0A        132           LD      A,LINEF
0091   D30C        133  SALTA     OUT     (DATOA),A   ;envia caracter o line-feed
0093   C9          134           RET
                   135
                   136
                   137  ;         R U T I N A     P R I N C I P A L
                   138
                   139  ;         Realiza las funciones de un eco con la terminal
                   140
```

```
0094  CD0C00    R  141  PRIN    CALL    coninit
0097  FB           142  OTRO    EI
0098  76           143          HALT
0099  3E18         144          LD      A,18H
009B  D30D         145          OUT     (CTRLA),A       ;resetea el canal A
009D  CD0C00    R  146          CALL    coninit
00A0  CD6800    R  147          CALL    conin
00A3  CD6F00    R  148          CALL    conout
00A6  18EC         149          JR      PRIN
                   150
00A8  00           151  TECLA   DEFB    0
00A9  0000         152  GUARDA  DEFW    0
00AB  00           153  SALVAA  DEFB    0
                   154
%x
```