DIRECTORIO DE PROFESORES DEL CURSO MICROPROCESADORES Y MICROCOMPUTADORAS SEPT.-OCT. 1983

1. M. EN I. LUIS HUGO PEÑARRIETA ECHENIQUE (Coordinador)
Investigador Asociado B
IIMAS
UNAM
México, D.F.
550 55 85

2. ING. HECTOR CALVARIO MARTINEZ
Técnico Académico Asociado "C"
Profesor
Facultad de Ingeniería
U N A M
México, D.F.
543 06 89 (casa)

3. ING. ROLANDO SAMUEL CARRERA SANCHEZ
Técnico Asociado C
I I M A S
Profesor de Estructura de Datos
Facultad de Ingeniería
U N A M
México, D.F.
550 55 85

4. ING. OCTAVIO OROZCO Y OROZCO
Técnico Académico
Instituto de Investigaciones en
Matemáticas Aplicadas y Sistemas
U N A M
México, D.F.

5. ING. DAVID BETANCOURT
Técnico Académico
Auxiliar "C"
I I M A S
U N A M
México, D.F.
550 52 15 ext. 4584

MICROPROCESADORES Y MICROCOMPUTADORAS 19 de Sept.al 1° de Oct. 1983.

Fecha	Hora	Tema	Profesor
19 Sept.	17 a 21 h	DESARROLLO DE LA MICROCOMPUTACION LOGICAS DE SEMICONDUCTORES	Ing. Luis H. Peñarrieta Echenique
20 Sept.	9 a 14 h	MICROPROCESADORES	Ing. Héctor Calvario Martínez
21 Sept.	17 a 21 h	MICROCOMPUTADORAS Unidad Central de Procesamiento Memoria Principal Chips de soporte para periféricos	Ing. Luis H. Peñarrieta Echenique
22 Sept.	17 a 21 h	MICROCOMPUTADORAS Memoria Masiva	" " " " "
23 Sept.	17 a 21 h	MICROCOMPUTADORAS Terminales de video Impresoras Ejemplos de microcomputadoras	Ing. Héctor Calvario Martínez
24 Sept.	9 a 14 h	PRACTICAS DE HARDWARE	Ing. " " " Ing. Luis H. Peñarrieta Echenique
26 Sept.	17 a 21 h	LOS LENGUAJES ENSAMBLADORES	Ing. Rolando Samuel Carrera Sánchez
27 Sept.	17 a 21 h	EDITORES	Ing. Octavio Orozco y Orozco
28 Sept.	17 a 21 h	INTERPRETES	Ing. David Betancourt
29 Sept.	17 a 21 h	SISTEMAS OPERATIVOS	Ing. Rolando Samuel Carrera Sánchez
30 Sept.	17 a 21 h	MESA REDONDA	Ing.David Betancourt Ing. Héctor Calvario Martínez Ing. Rolando Samuel Carrera Sánchez Ing. Samuel Carrera Ing. Octavio Orozco y Orozco Ing. Luis H. Peñarrieta E.
1° Oct.	9 a 14 h	APLICACIONES	Ing.David Betancourt Ing. Rolando S.Carrera Ing. Octavio Orozco y Orozco

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

SEPTIEMBRE, 1983

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 Jonn 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 chips 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 alcance; 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 Shima 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 hasta 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 hacia 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 $us. 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 Intél 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 chip de 8 bits de 212 x 224 mils (milesimas de pulgada).

El chip fue demostrado a CTC en marzo de 1971, sin embargo, la terminal no utilizó este chip. Aunque el contrato con CTC habia 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 hacia 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 mucnos 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 habian 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 considearar 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 hubo 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 chips 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 periféricos 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 Michael 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 chip 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 ha 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 hace 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 componarán 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:

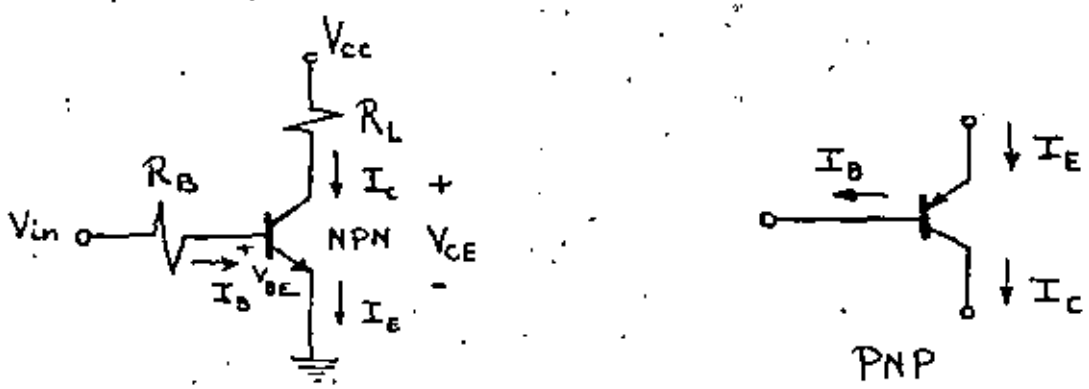


Figura 2-1: Transistores NPN y PNP

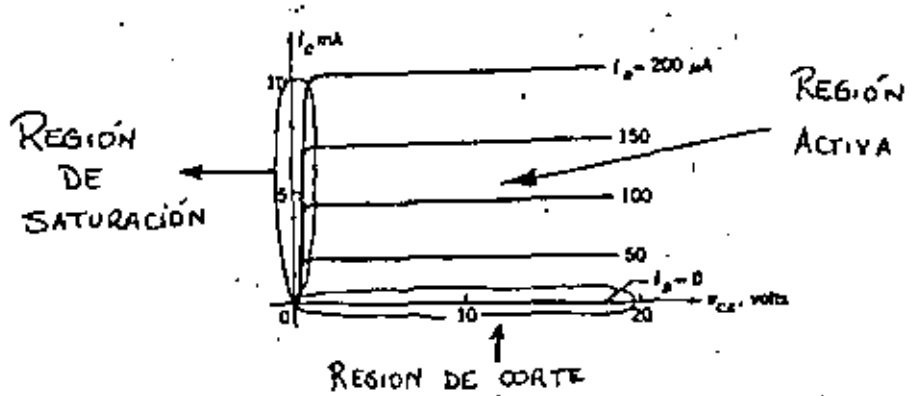


Figura 2-2: Regiones de operación del Transistor Bipolar

- 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.

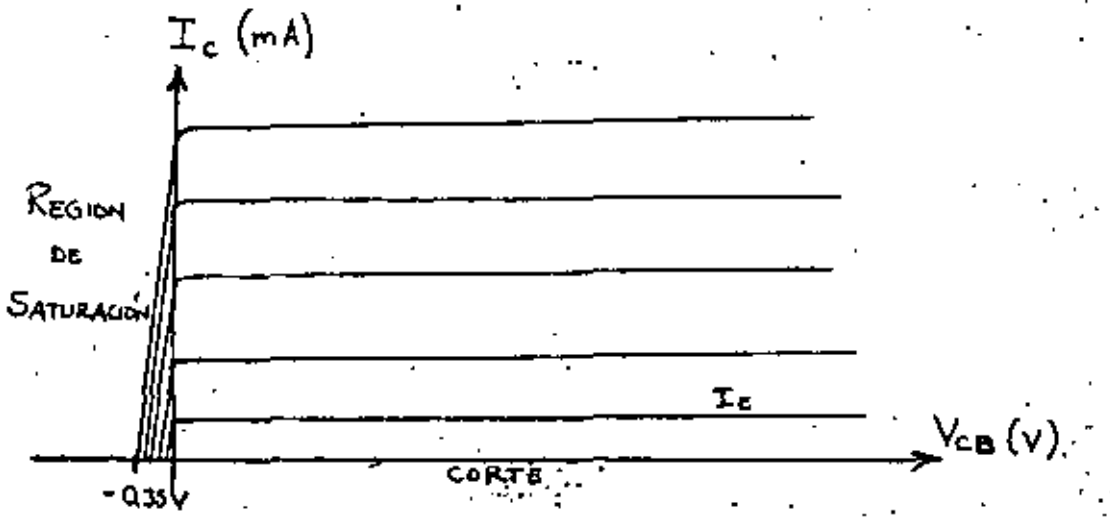


Figura 2-3: Región de saturación muy bien definida

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

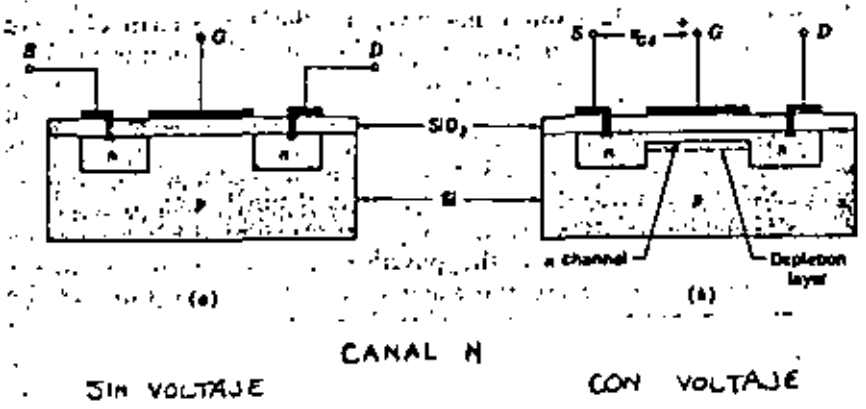


Figura 2-4: Estructura Básica del MOSFET

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.

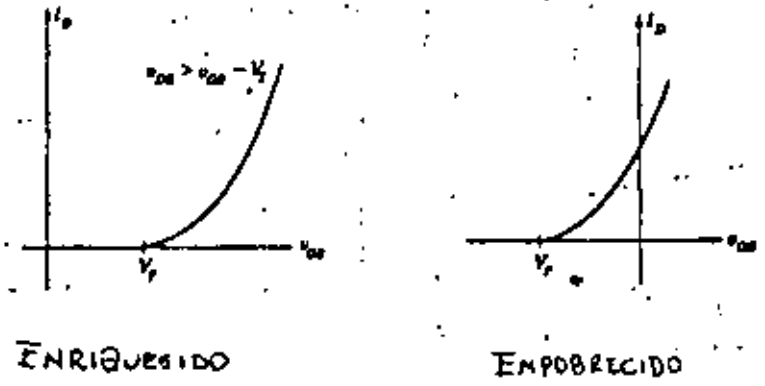


Figura 2-7: Voltajes de ruptura en MOSFETs enriquesidos y empobrecidos

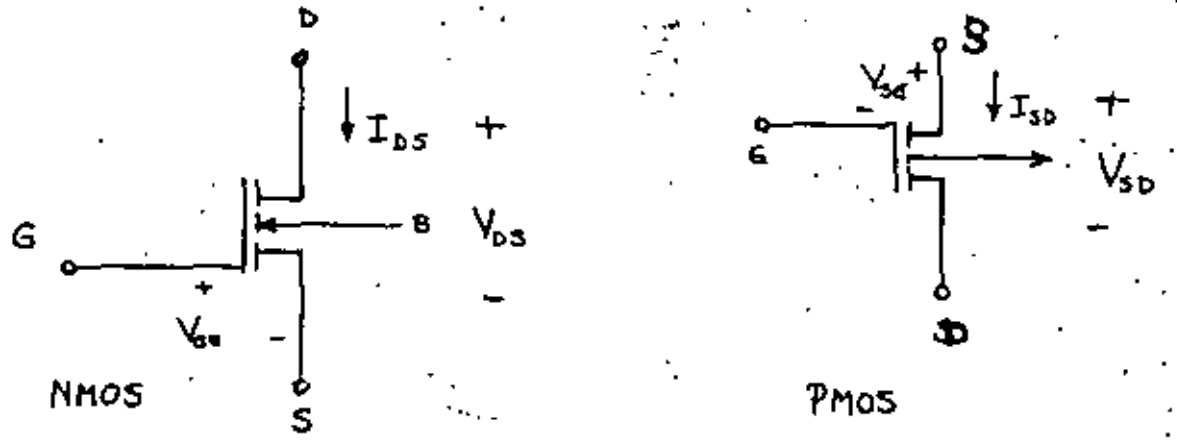


Figura 2-5: Transistores NMOS y PMOS

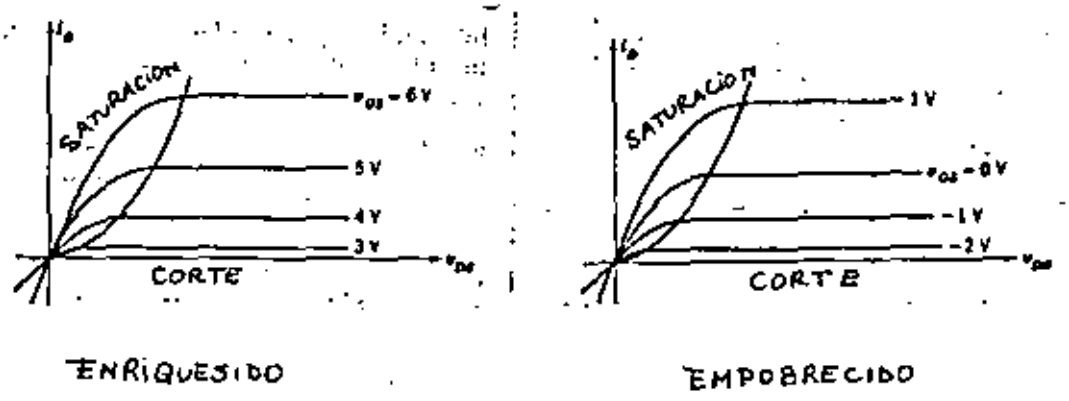


Figura 2-6: Regiones de operación de los transistores MOSFET

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 longitud del canal

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

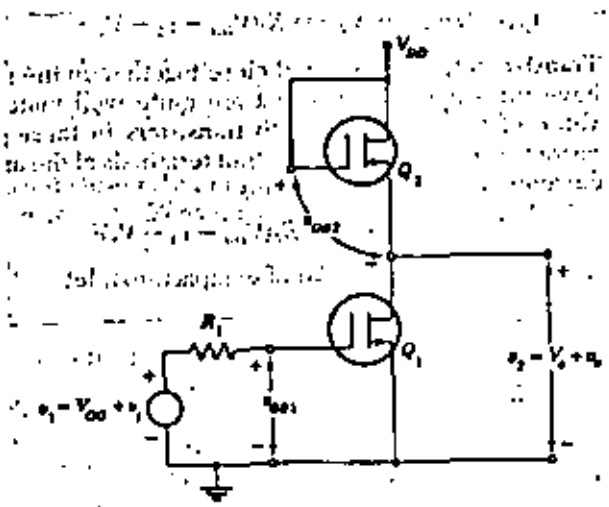


Figura 2-8: El Inversor MOSFET

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 difícil 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 longitud. En la figura 2-9 se observa que mientras mayor

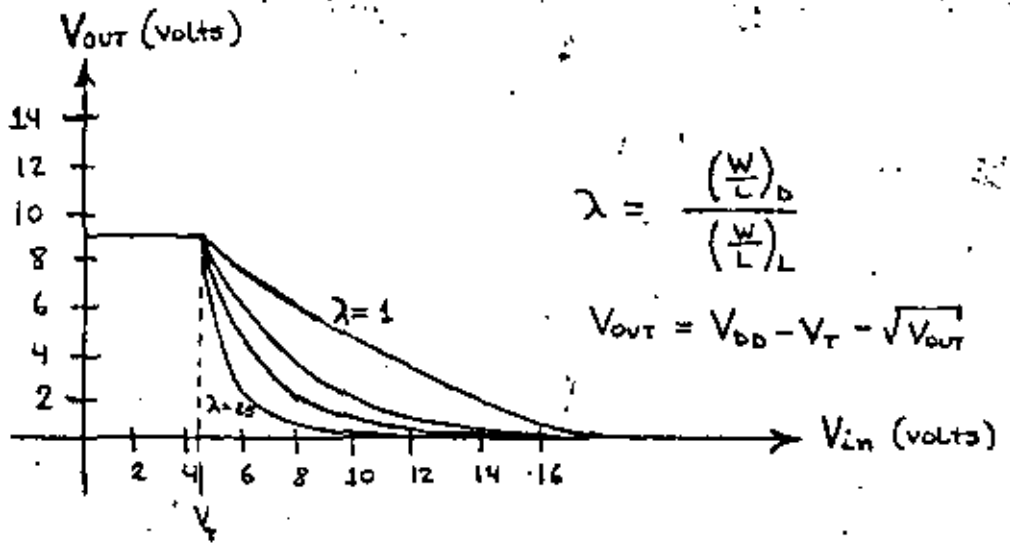


Figura 2-9: Curva de respuesta del Transistor MOSFET

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.

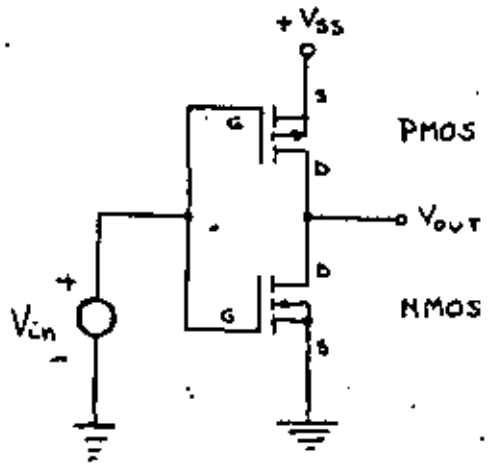


Figura 2-10: El Inversor CMOS

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.

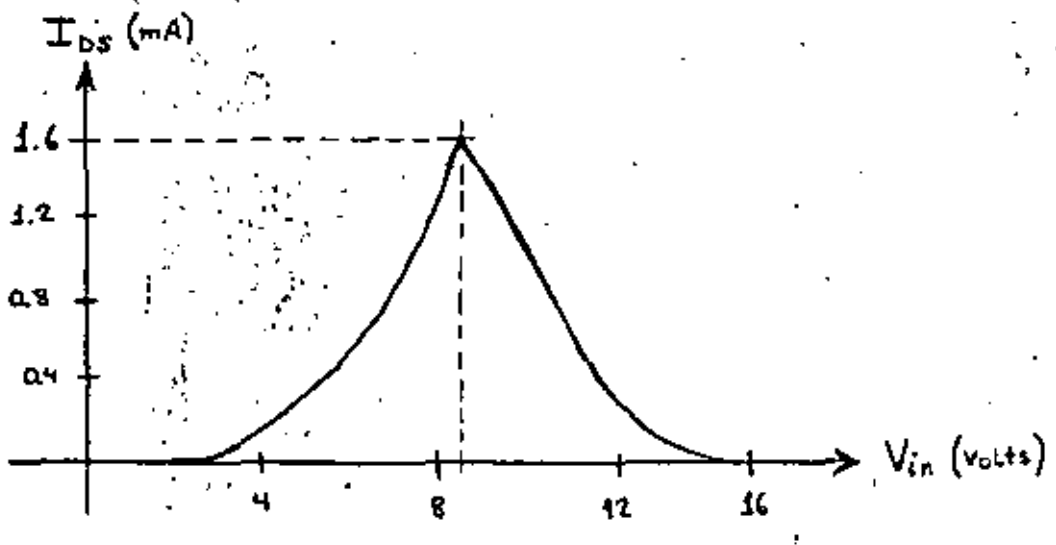


Figura 2-11: Consumo de corriente del CMOS

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 poco, 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.

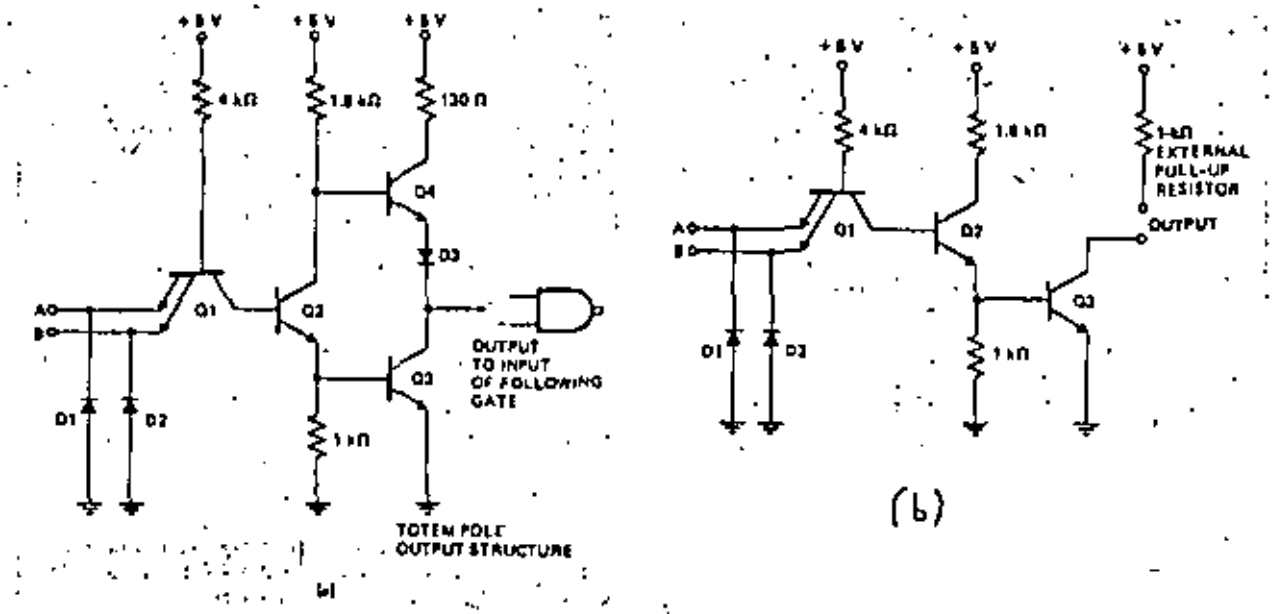


Figura 2-12: Compuertas NAND TTL estandar:
(a) Salida "totem pole"
(b) Salida de colector abierto

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.

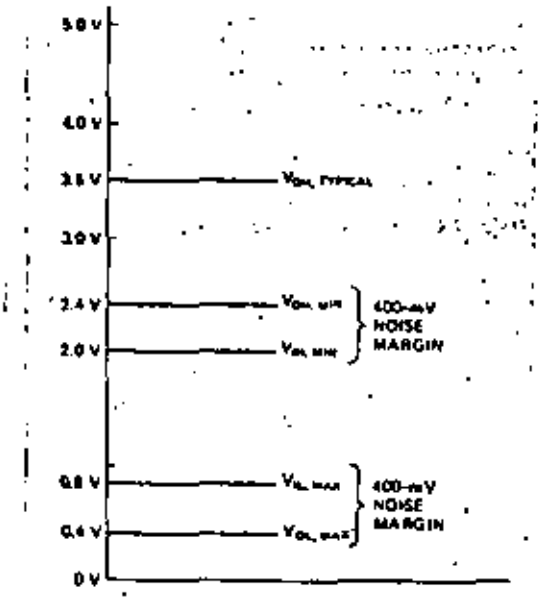


Figura 2-13: Niveles de voltaje del TTL estandar

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 microfarath o 0.01 microfarath 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

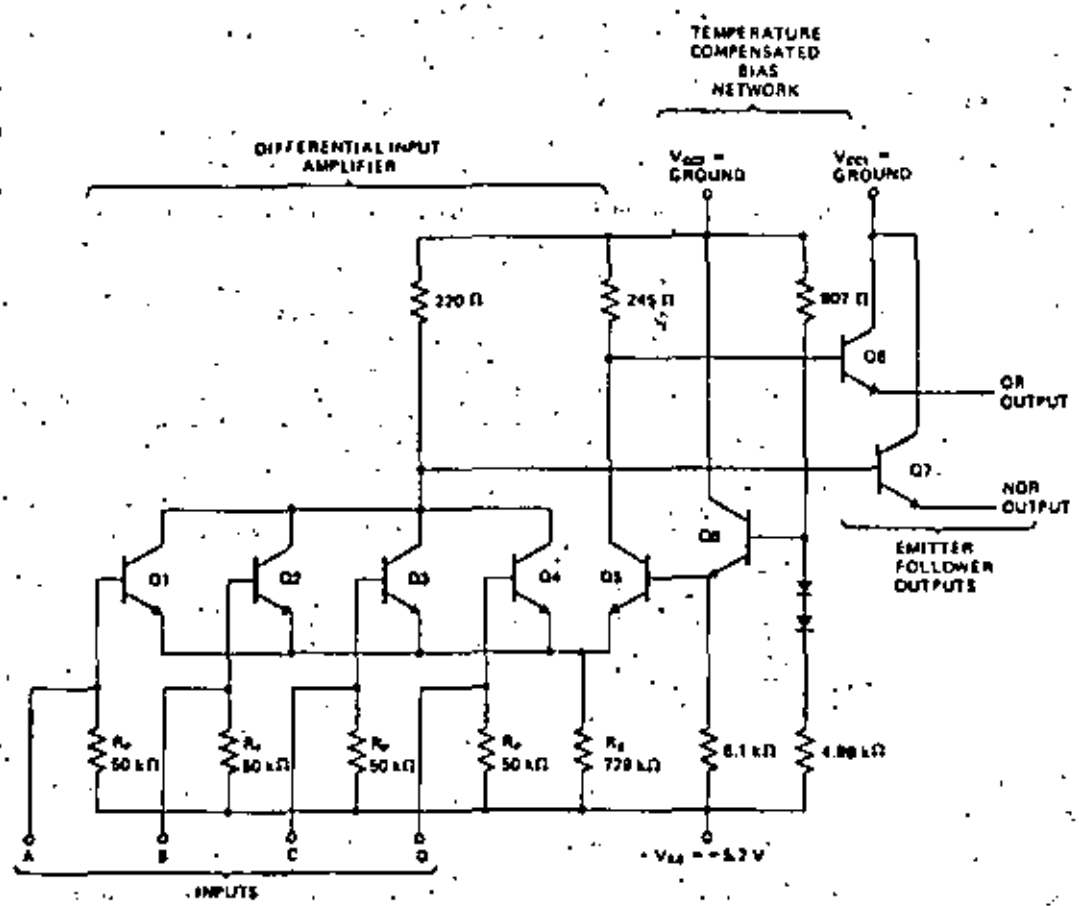


Figura 2-14: Compuerta elemental ECL

nseg. a 3.5 nseg., con esto es posible manejar lineas no terminadas hasta 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 (I²L). 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.

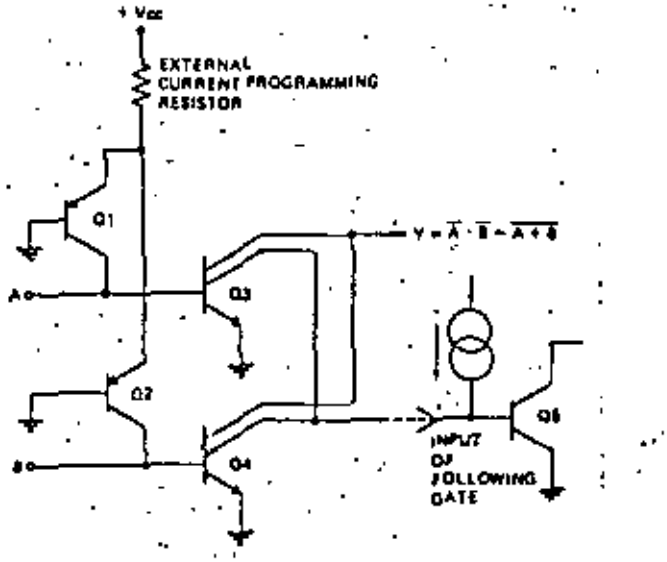


Figura 2-15: Compuerta IIL básica

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 amplificadores operacionales.

Una desventaja de esta lógica es que requiere un paso más en el proceso de manufactura 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 entoque 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.

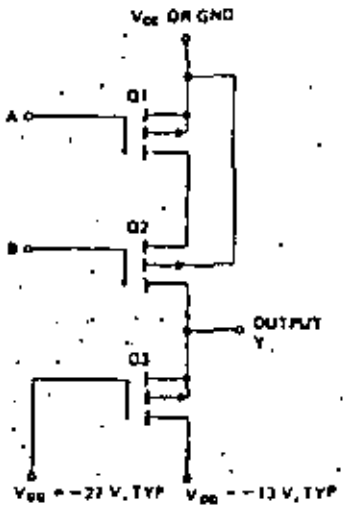


Figura 2-16: Compuerta PMOS básica

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 "pusn-pull" en las salidas

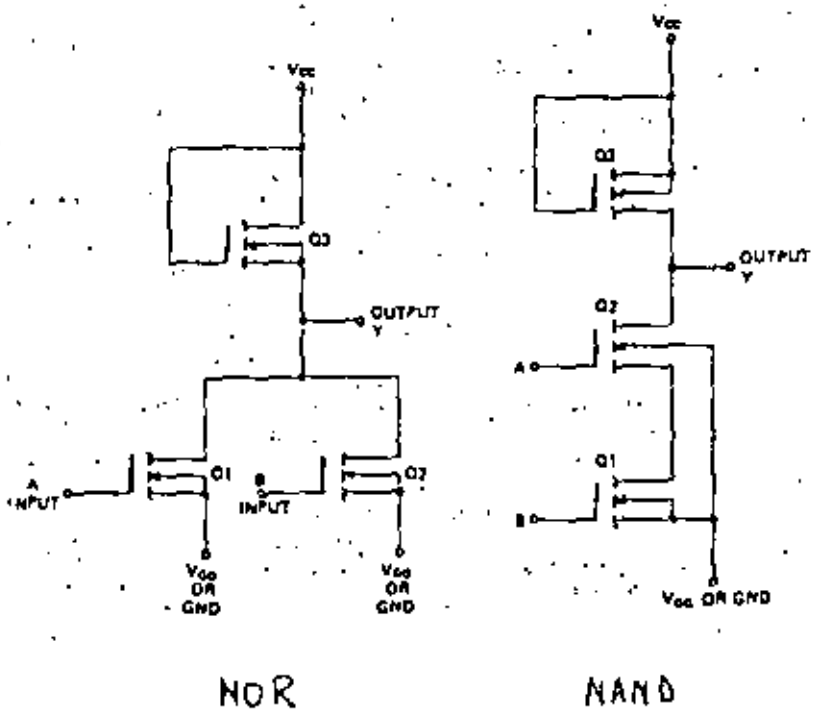


Figura 2-17: Compuertas NOR y NAND del tipo NMOS

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, asi como para circuitos de lógica LSI.

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

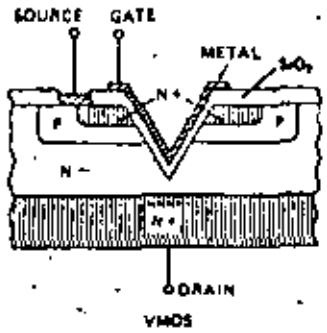


Figura 2-18: Transistor VMOS

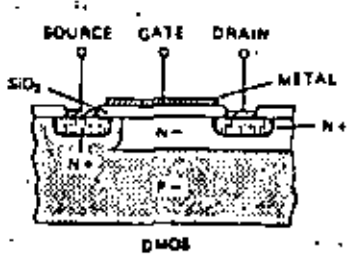


Figura 2-19: Transistor DMOS

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 había 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.

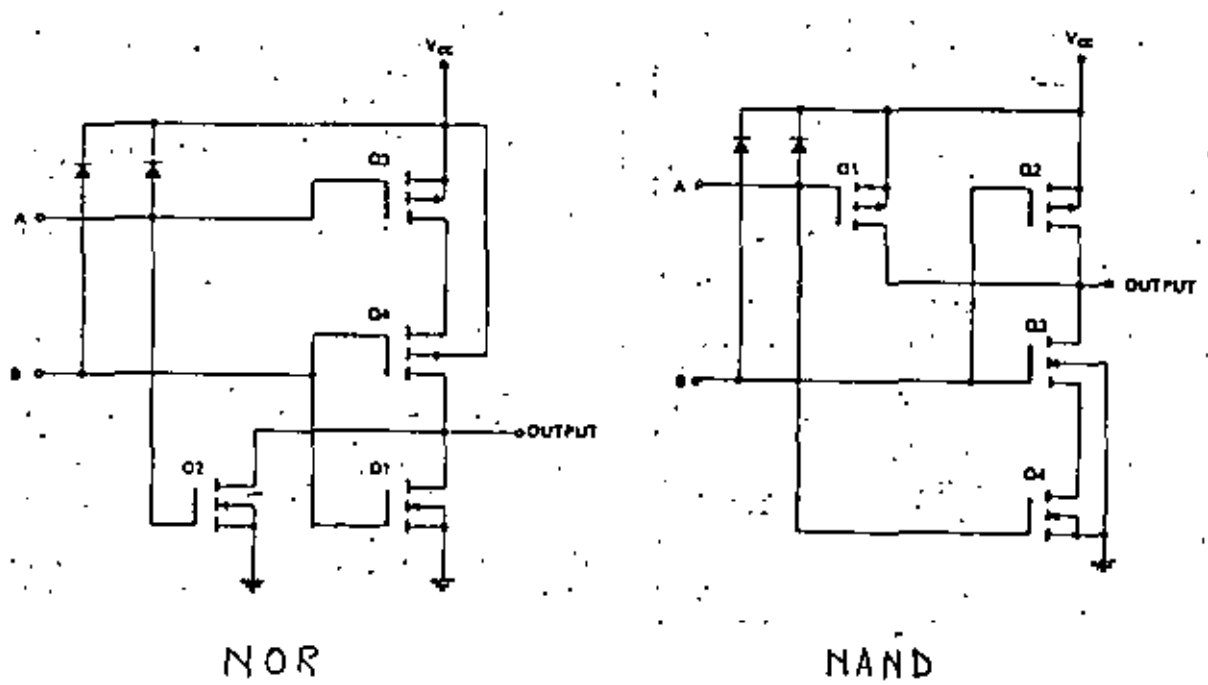


Figura 2-28: Compuertas NOR y NAND del tipo CMOS

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.

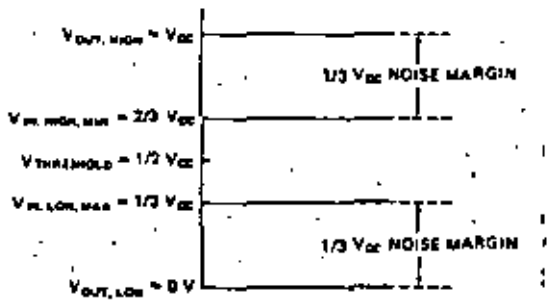


Figura 2-21: Niveles de voltaje y margenes de ruido en CMOS

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 alto y 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 (picotaradios) 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 tipica 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.

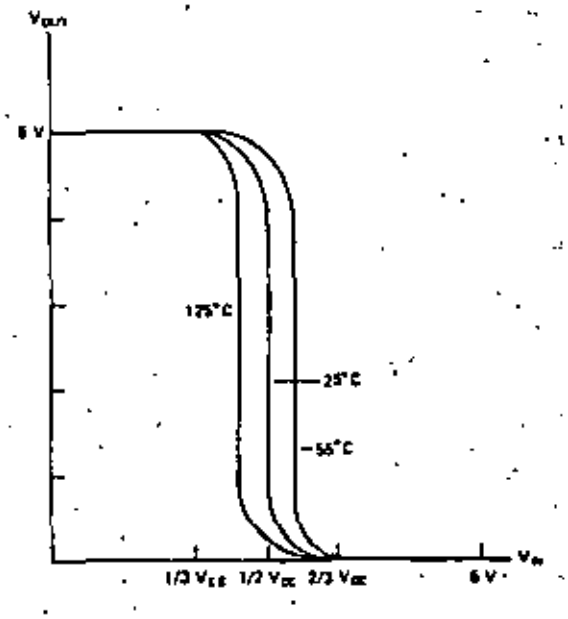


Figura 2-22: Curva de transferencia del CMOS. Inmunidad a la temperatura

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.

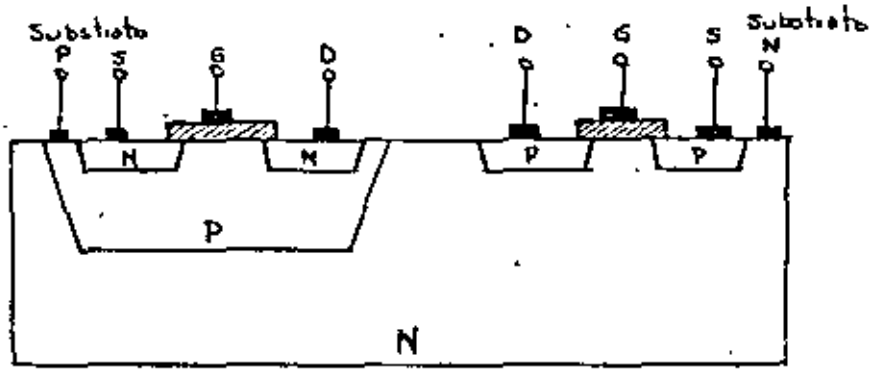


Figura 2-23: Estructura básica del CMOS

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 difícil, 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

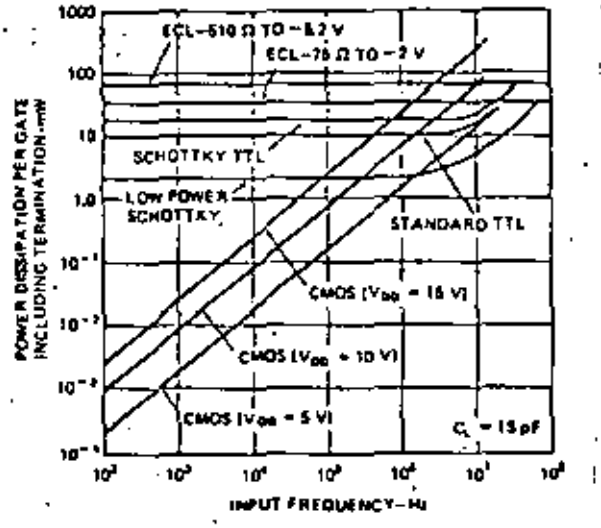


Figura 2-24: Típica disipación de potencia vs. frecuencia de la señal de entrada

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.

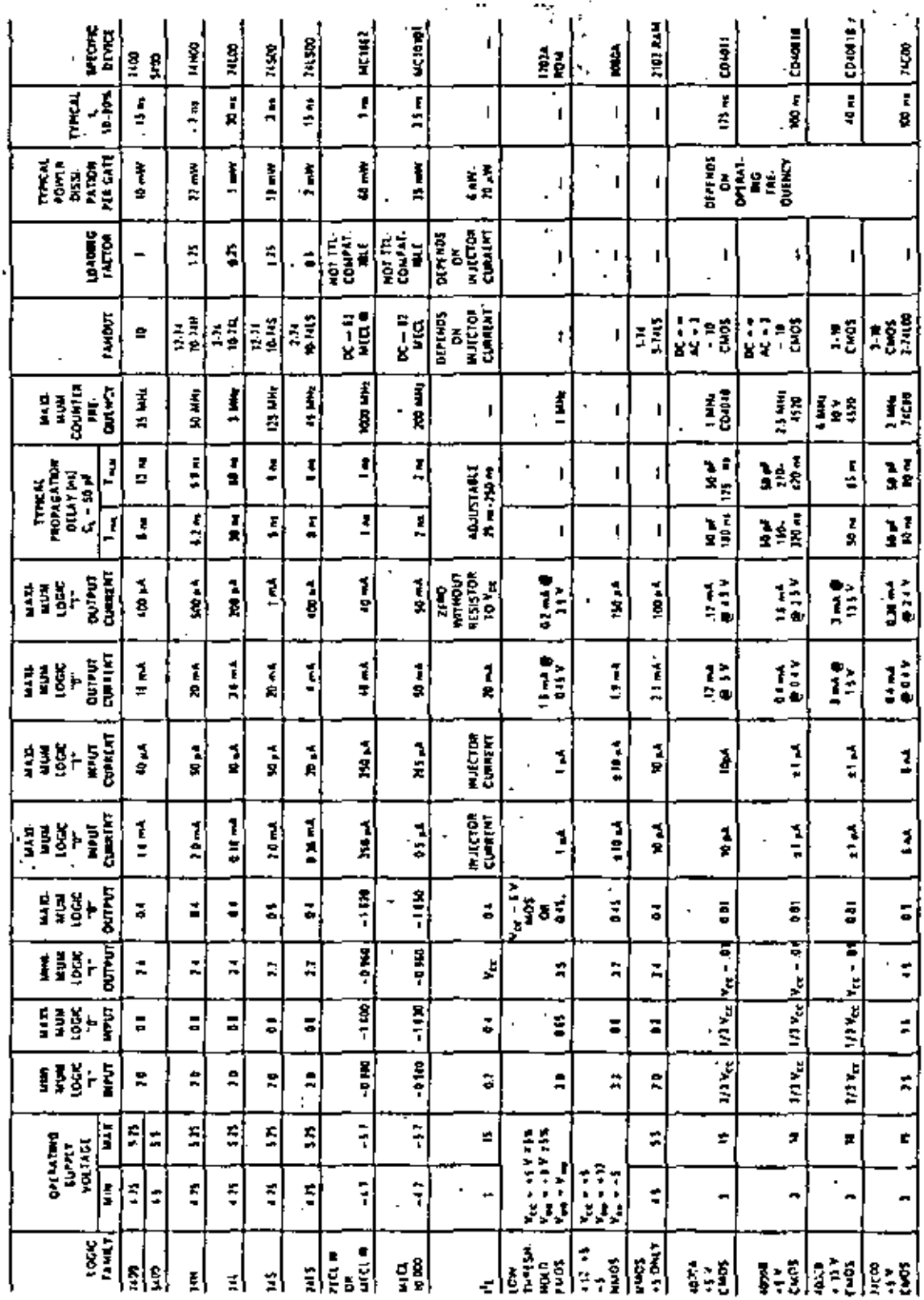


Tabla 2-1: Comparación de características comunes en varias familias lógicas

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 digital 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 especificas 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 nablar 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 categorías antes mencionadas, sin embargo, actualmente es mas difícil 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 variaciones 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.

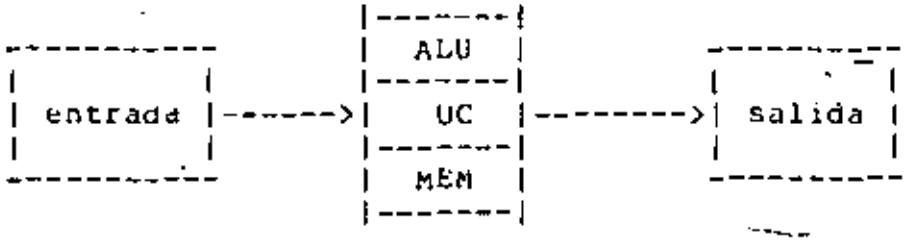


Figura 3-1: Componentes de una computadora.

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

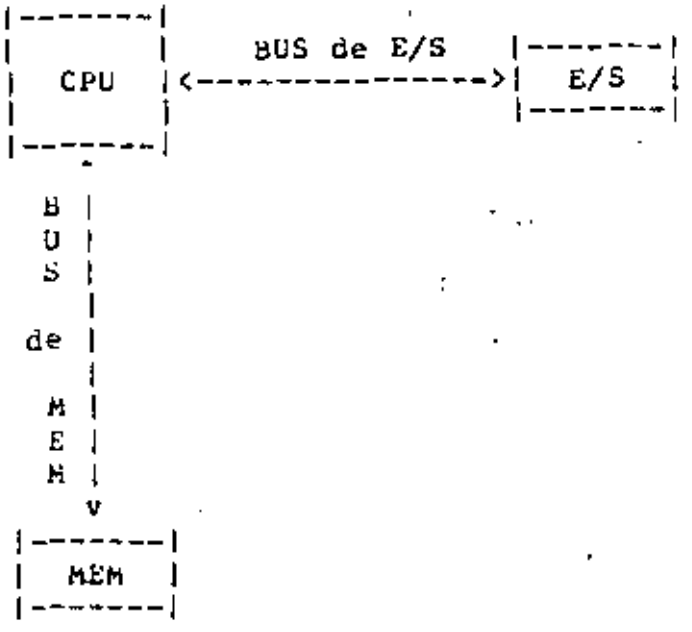


Figura 3-2: Representación de una computadora.

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 cojuntos 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.

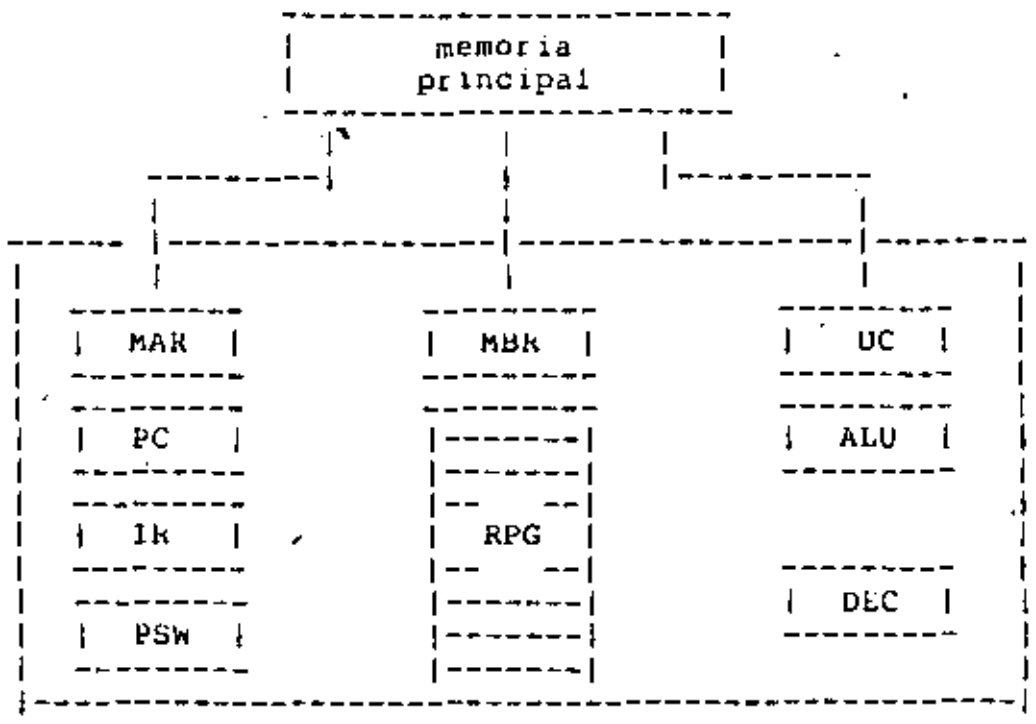


Figura 3-3: Conecciones entre el CPU y la memoria principal.

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 naciendo 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 hay que mencionar que los elementos que conforman

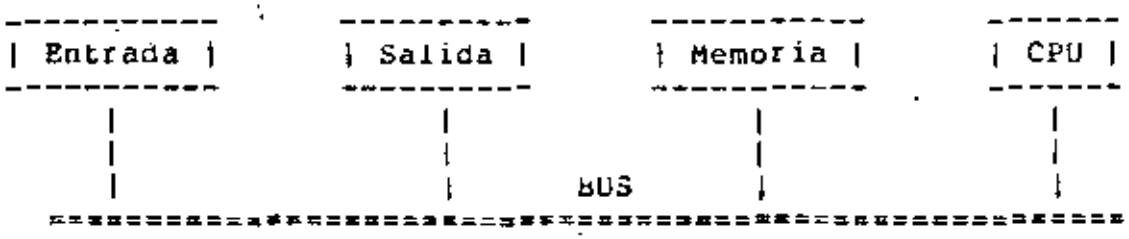


Figura 3-4: Estructura de un solo bus.

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 Charles 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 capacidad 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 líneamiento 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 contendra 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 simplemente 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 simplemente 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 específico 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 catastróticas 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 "habilitadas" o "deshabilitadas" bajo el control de programa (esto es, por el usuario), otras no pueden serlo. El procesador ignora aquellas peticiones de interrupción deshabilitadas; 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 nabilitadas y desnabilitadas 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 dispósitivos 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 hablando, 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"

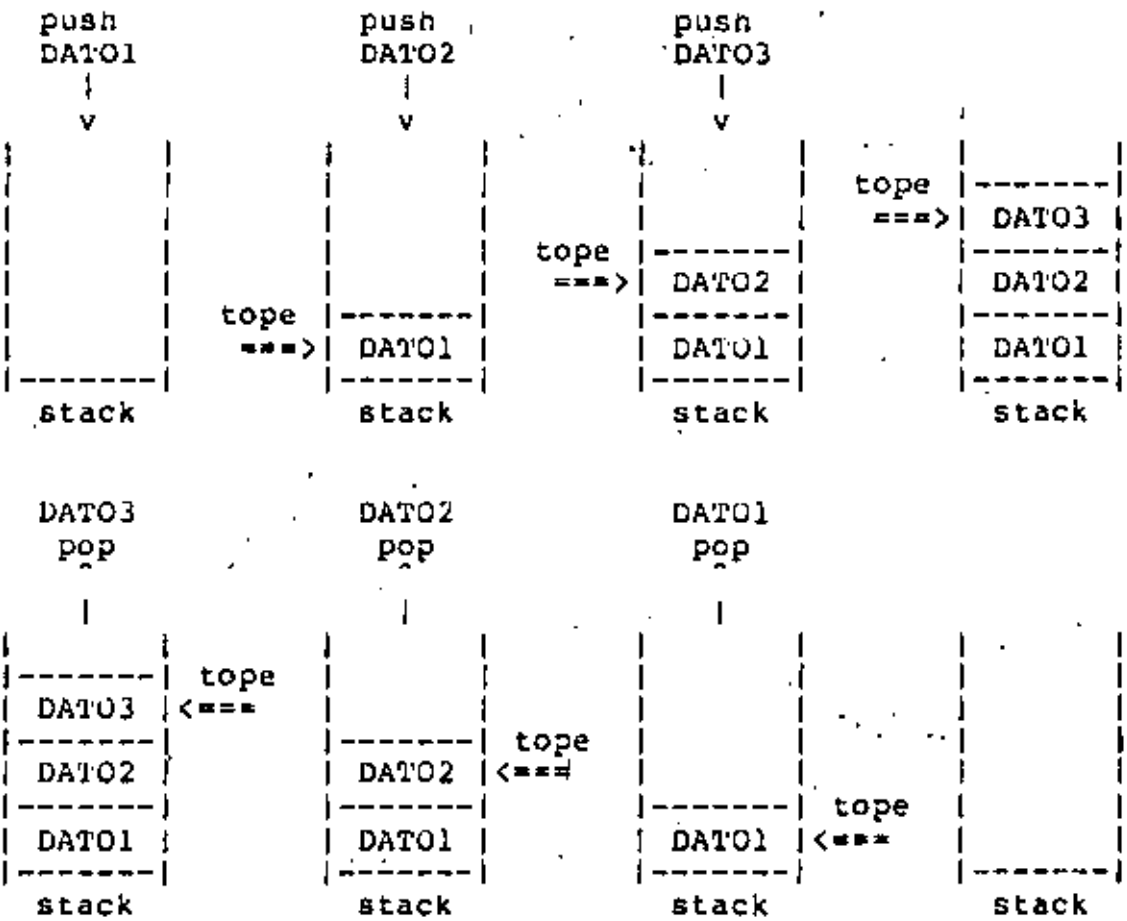
Los microprocesadores generalmente manejan un tamaño fijo de palabra (la cuál puede ser 8, 12, 16 y en ocasiones de 32 bits); también existen aquellos en los cuales sus unidades funcionales, están divididos modularmente en varios chips idénticos que pueden ser ligados en paralelo, el número total de 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 característica 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 característica de manejo de palabras de 8 bits como por lo común que son en el mercado, su marcada preferencia y su trascendencia histórica (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 específicas, 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 decoditicación 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.

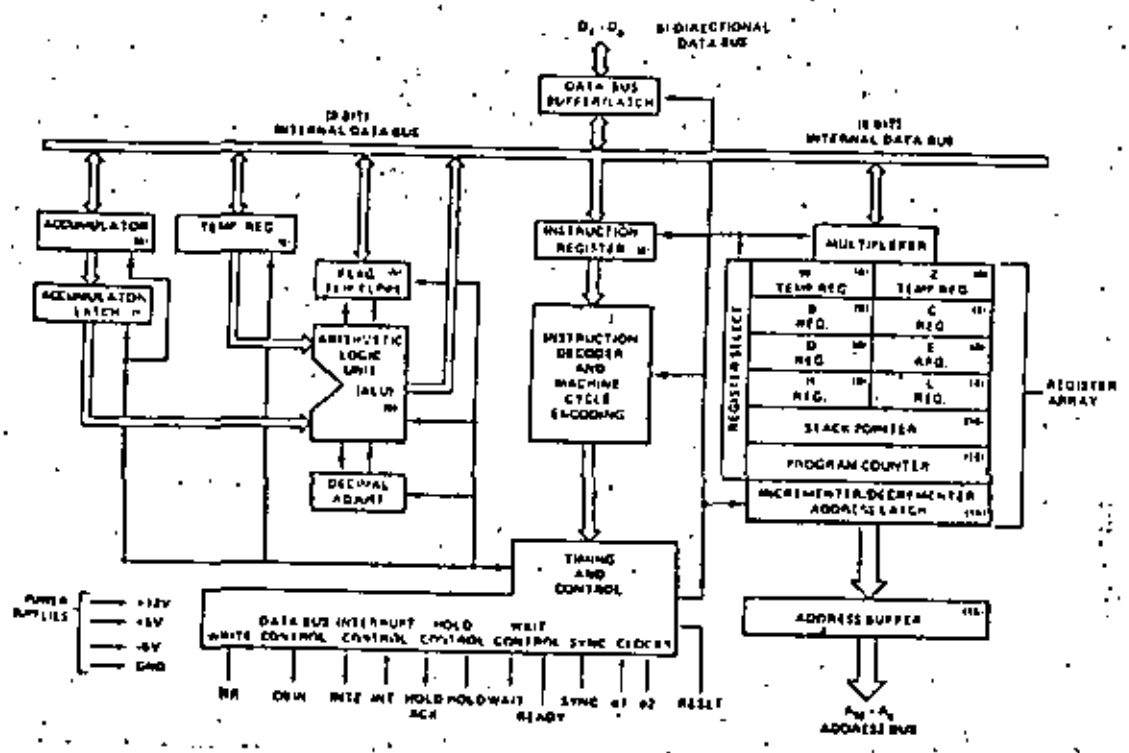


Figura 3-6: Diagrama de Bloques Funcionales del CPU 8080.

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.

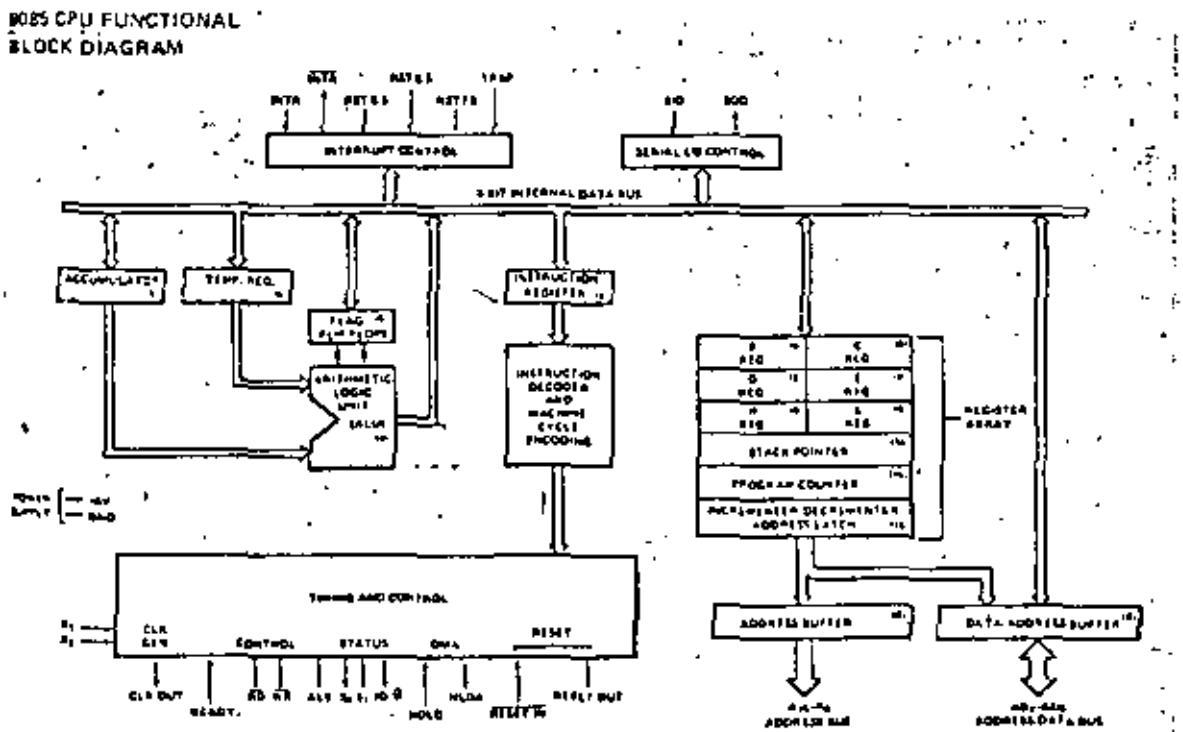


Figura 3-7: Diagrama de los bloques funcionales del CPU 8085

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

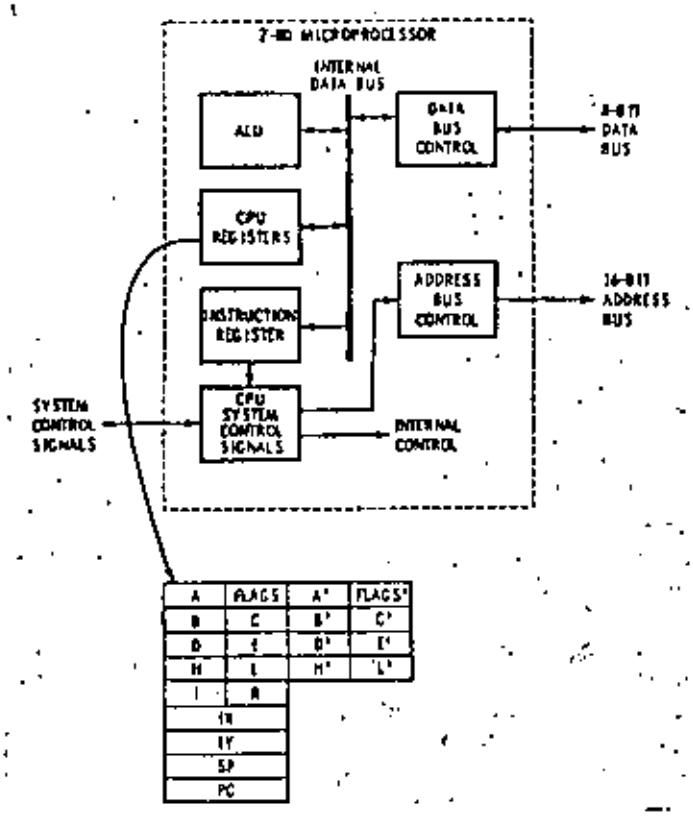


Figura 3-8: Diagrama de los bloques funcionales del CPU Z80.

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',L',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 Buffer, 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 FCR llamado Condition Code Register. A continuación se muestran las caracteristicas importantes de este microprocesador:

TECNOLOGIA: NMOS, en chip 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.

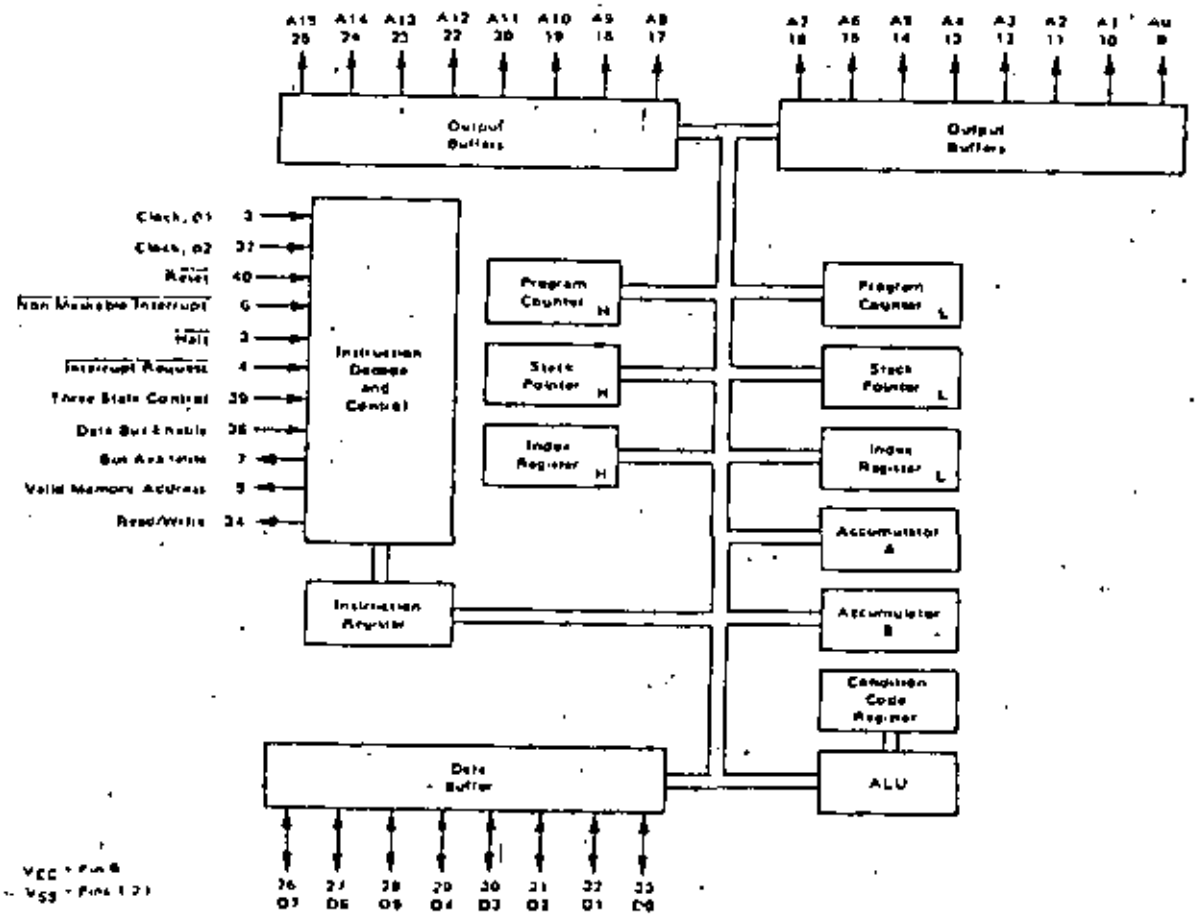


Figura 3-9: Diagrama de los bloques funcionales del CPU 6800.

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).

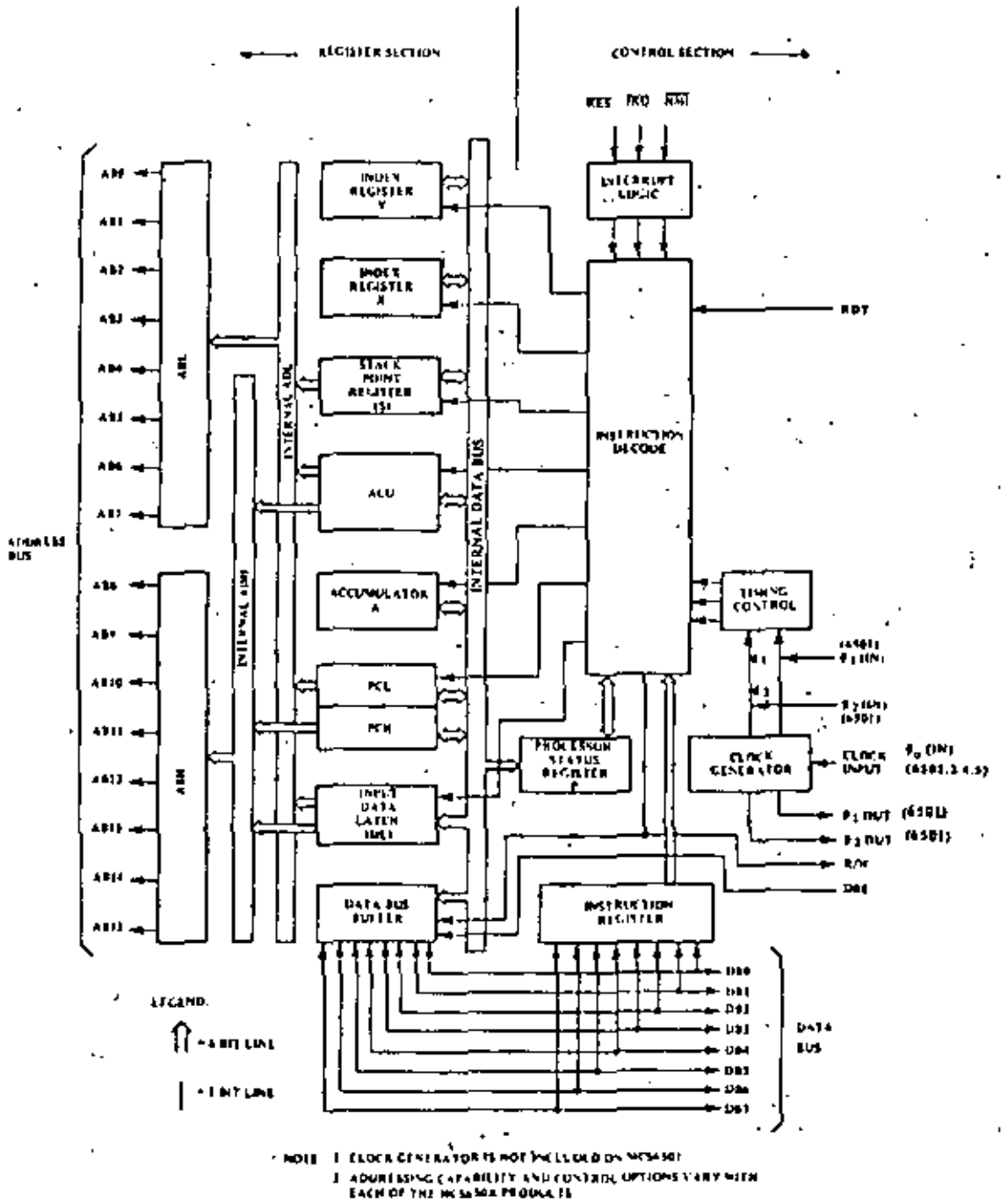


Figura 3-10: Diagrama de los bloques funcionales del CPU 6502.

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 deshabilitar 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 efectúar 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

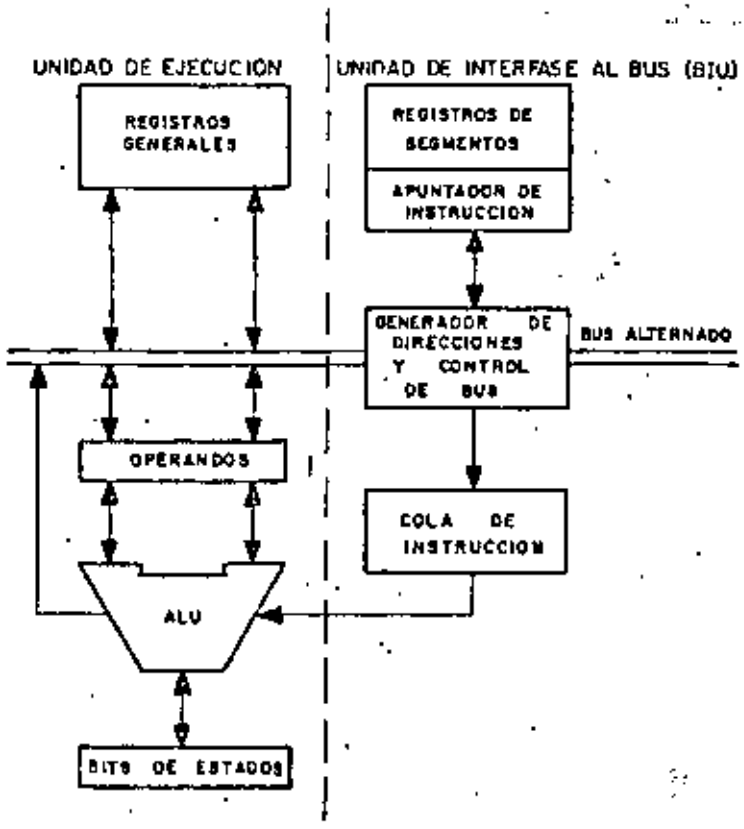


Figura 3-11: Estructura interna del microprocesador Intel 8086 (iAPX 86).

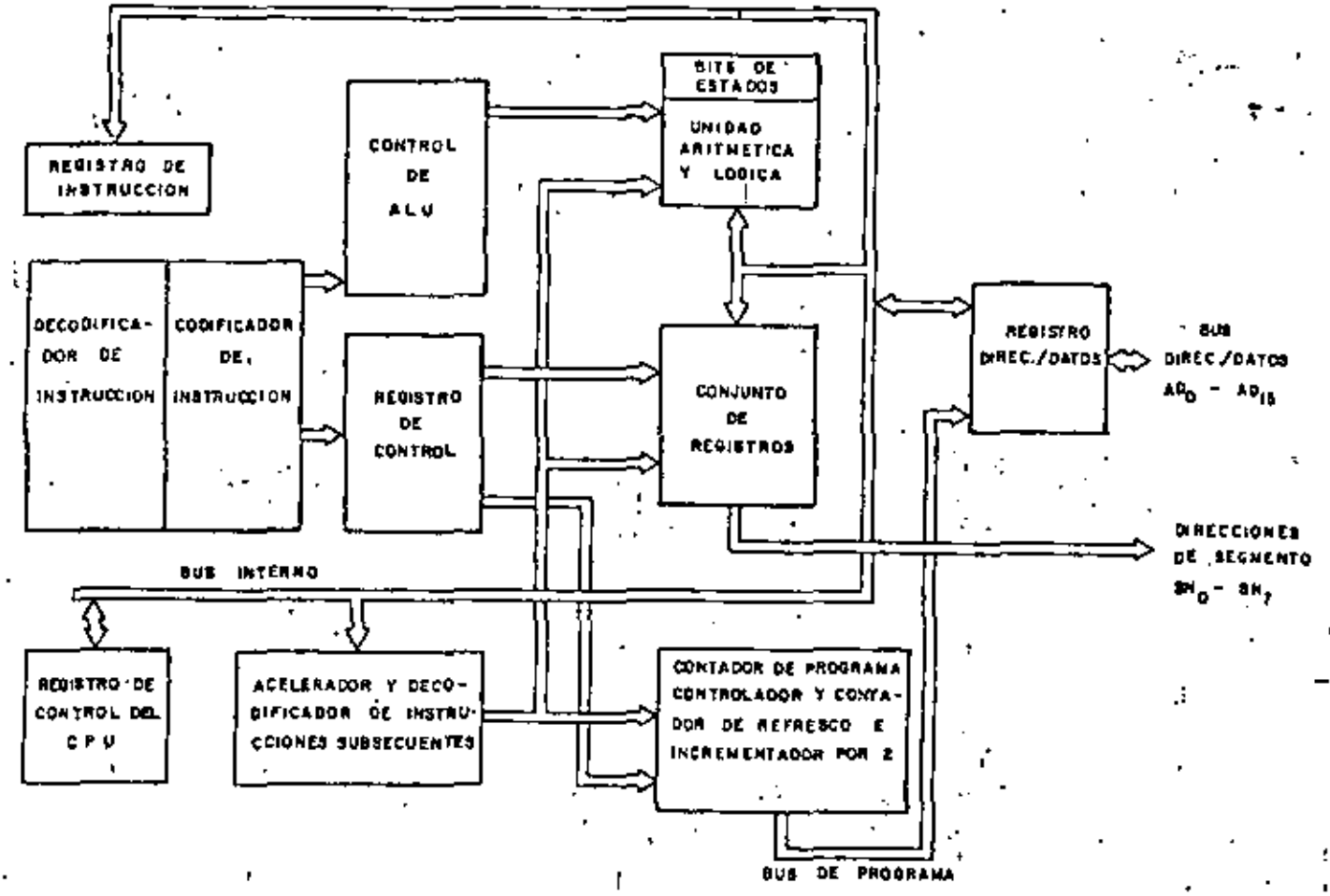


Figura 3-12: Estructura interna del microprocesador Zilog Z8000.

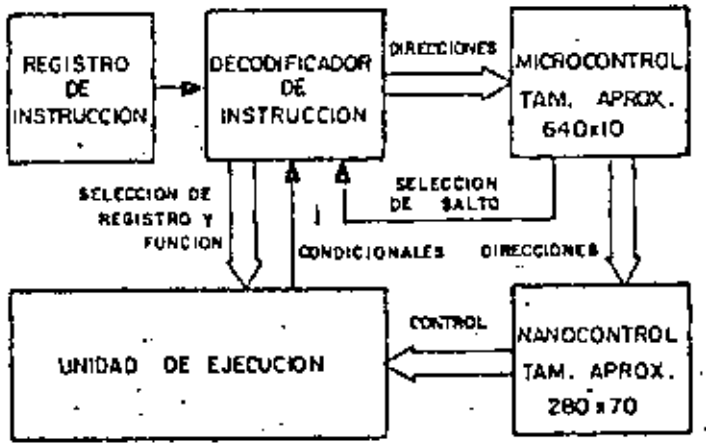


Figura 3-13: Estructura interna del microprocesador Motorola MC68000.

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 dos 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

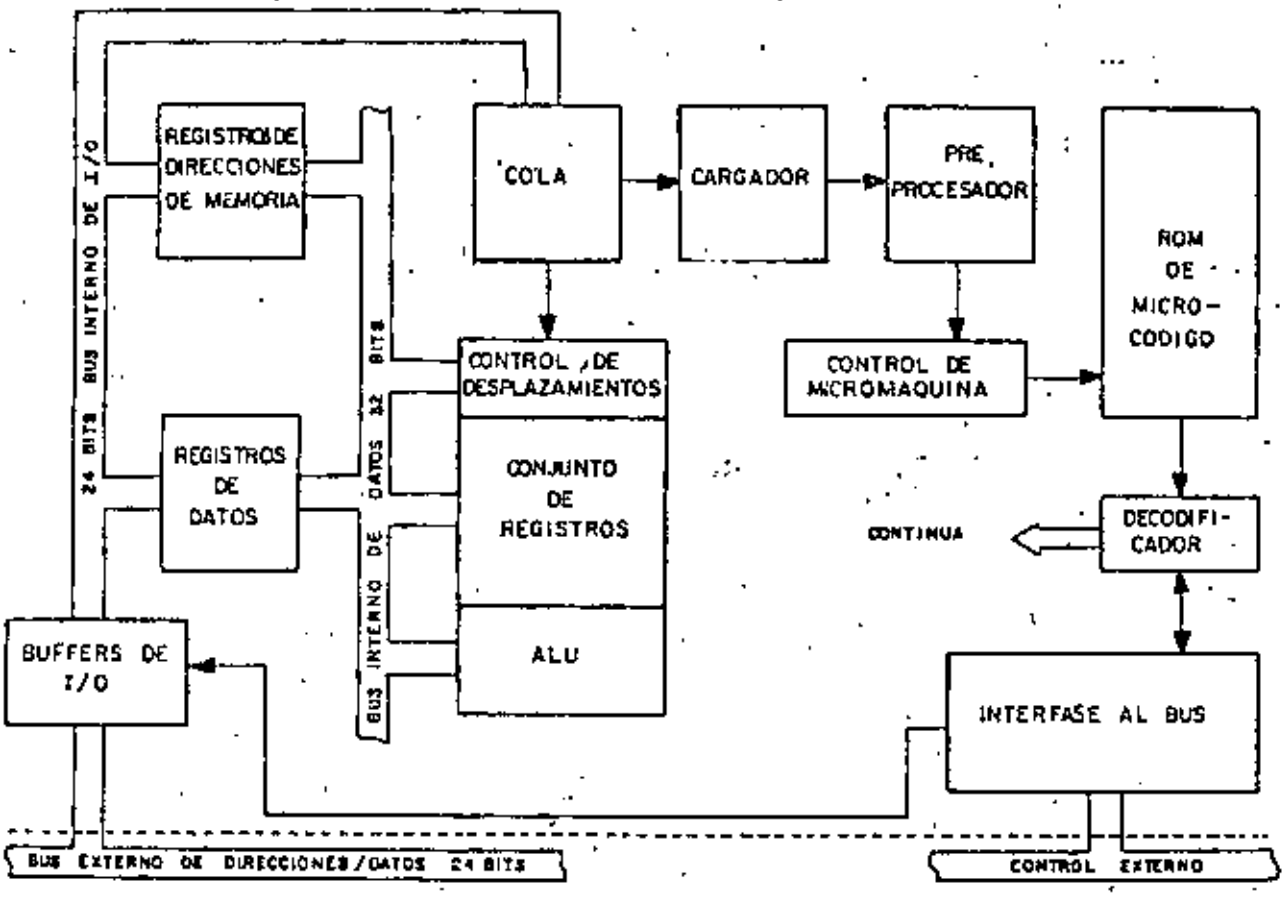


Figura 3-14: Estructura internal del microprocesador National Sémiconductor NS16000.

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 entoque 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 Z8000 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 haróware 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 28000 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 28000 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 sottware de alto nivel. Los 4 modos comunes son:

- 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.

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 280. El 8086 tiene dos instrucciones para manejar dispositivos externos ESC y LOCK.

El 28000 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 28000 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.

- 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 software 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 índice de un arreglo está dentro de sus tronteras, 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 inhibe. 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.

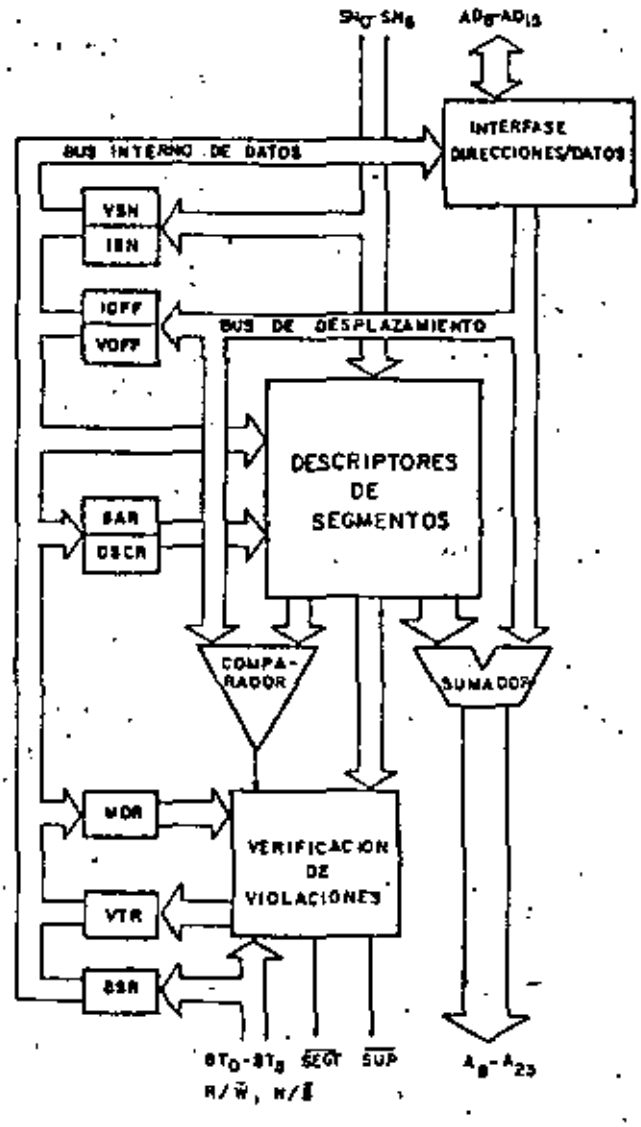


Figura 3-16: Diagrama de bloques de la Unidad de Manejo de Memoria (MMU) para el Z8000.

- 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.

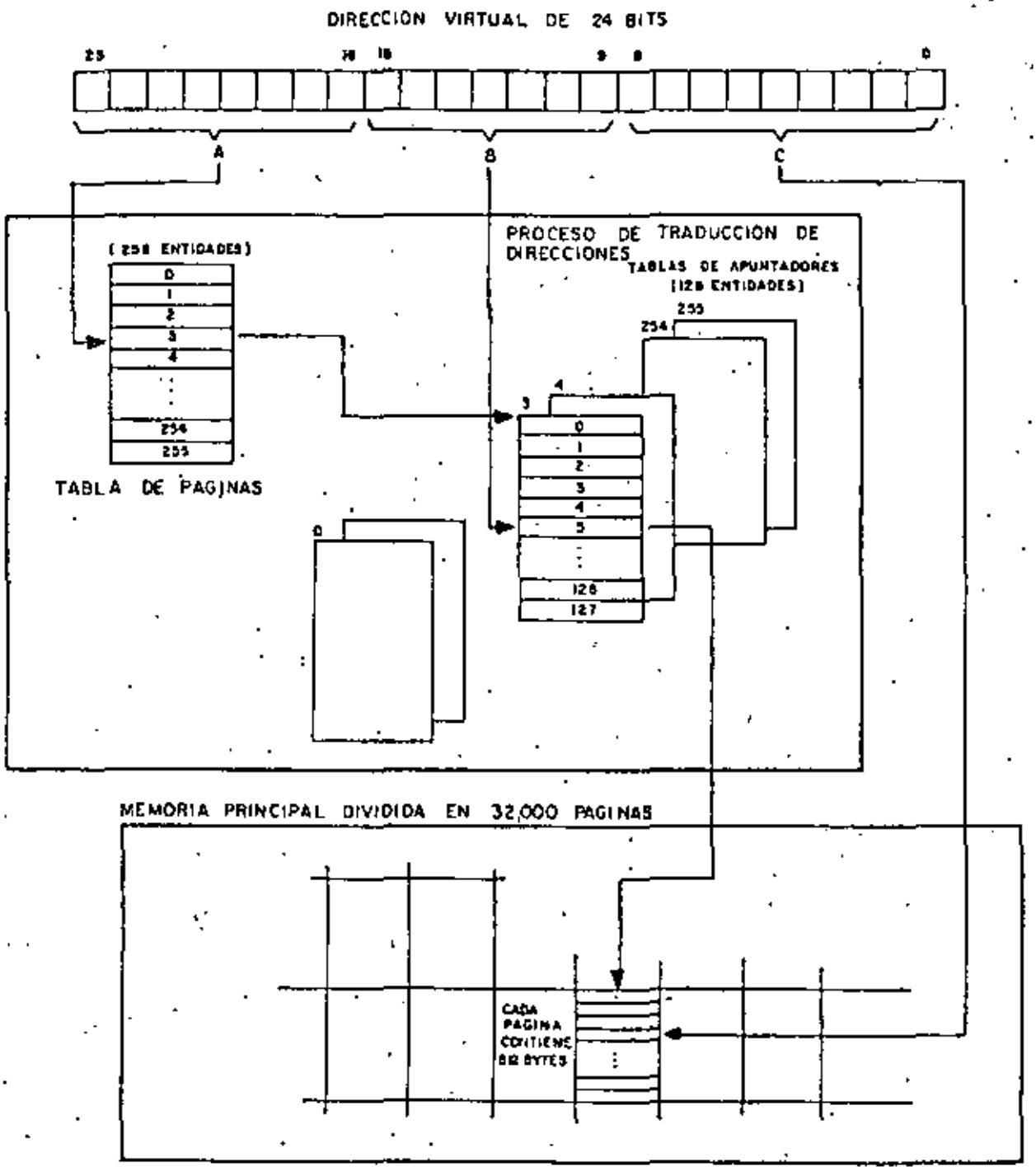


Figura 3-17: Estructura del manejo de paginas (MMU) para el NS16032

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	Byte/Palab	0.40	0.75	0.50	0.30
reg -> reg	Doble pal.	0.80	1.25	0.50	0.30
MOVIMIENTO	Byte/Palab	7.00	7.00	2.50	1.60
mem -> mem	Doble pal.	14.00	8.50	3.75	2.40
SUMA	Byte/palab	3.60	3.75	1.50	1.10
mem -> mem	Doble pal.	7.20	5.25	2.25	1.50
MULT	Byte	13.00	20.25	8.75	2.80
mem -> mem	Palabra	23.00	16.00	8.75	4.60
	Doble pal.	115.20	85.75	43.00	7.60
Salto	Efectuado	1.60	1.50	1.25	1.30
Condicional	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 "butters" 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 dígitos 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 leídas, 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 leídas 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

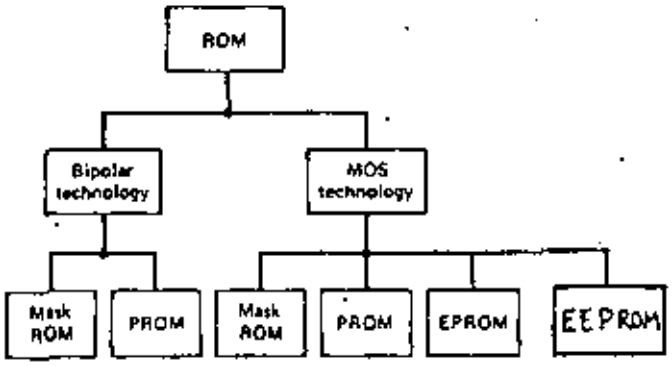


Figura 4-1: Tipos de memorias ROM

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 hasta 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 tecnológico, 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,

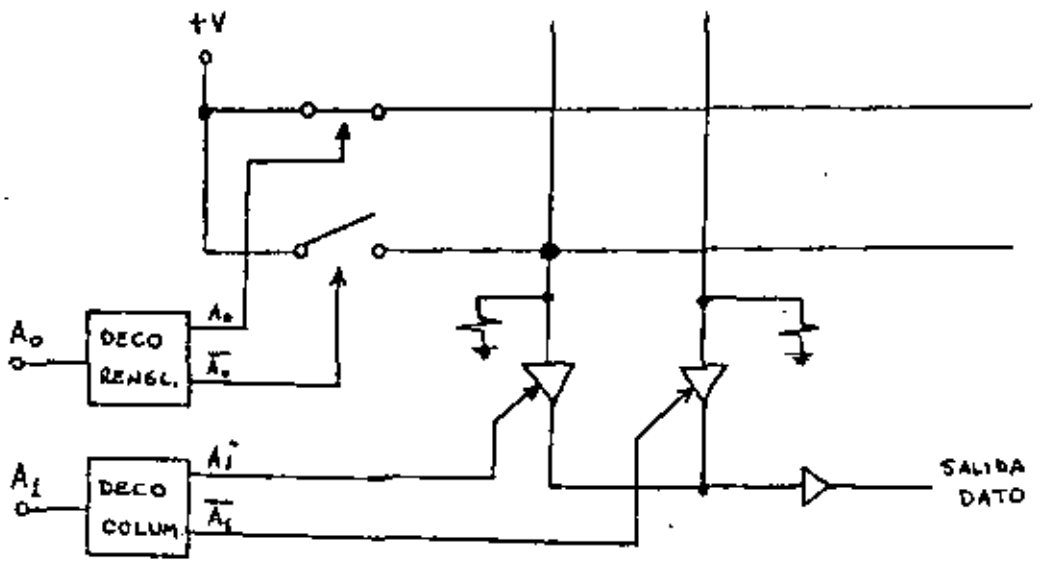


Figura 4-2: Metalizado en las celdas del ROM de mascara

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 dicha interseccion. 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

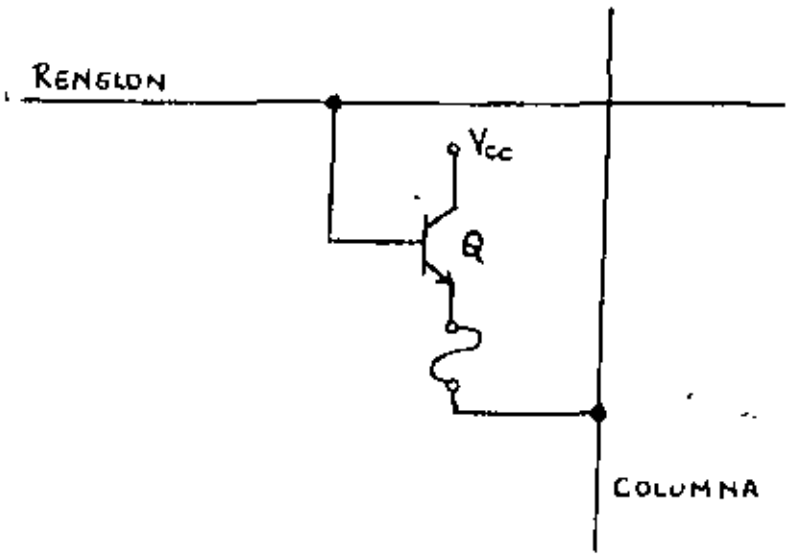


Figura 4-3: Celda básica de un PROM

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 hasta lograr la versión definitiva del programa. También, son usados en situaciones en las cuales la información almacenada eventualmente puede cambiar.

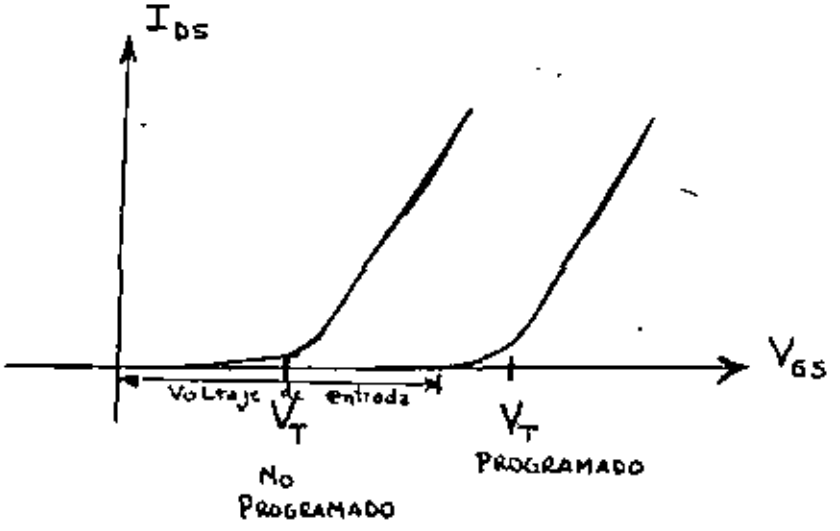


Figura 4-4: Efecto en el voltaje de ruptura al programar el EPROM

En 1971 Intel lanzó al mercado el primer EPROM, fue el 1702 el cual tenía como celda de memoria un transistor llamado FAMOS (floating gate avalanche 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 territas 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					

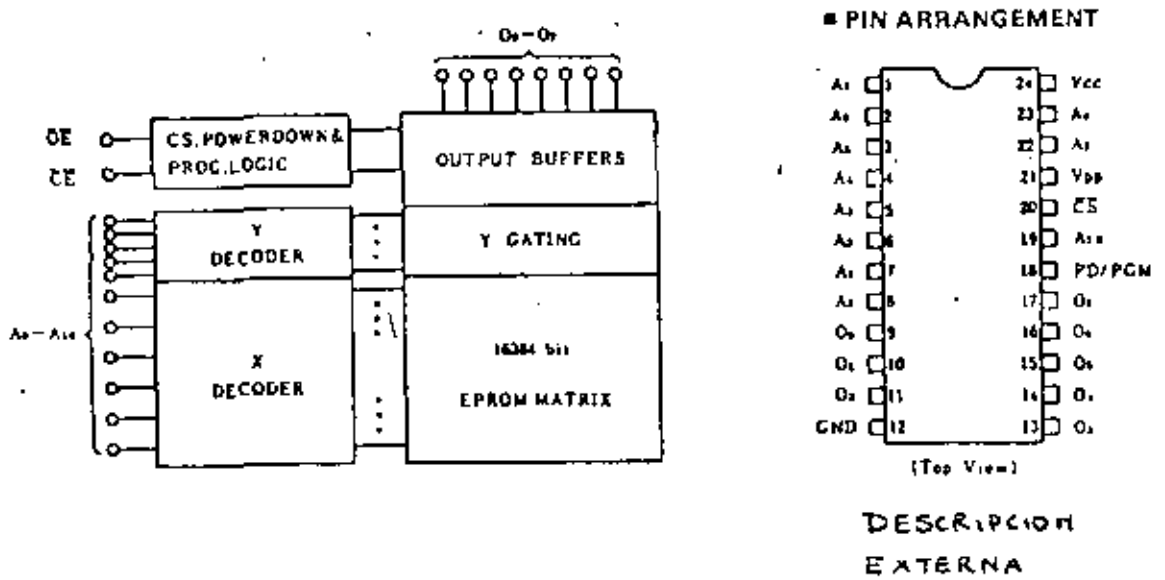


Figura 4-5: Organización interna del EPROM 2716

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})

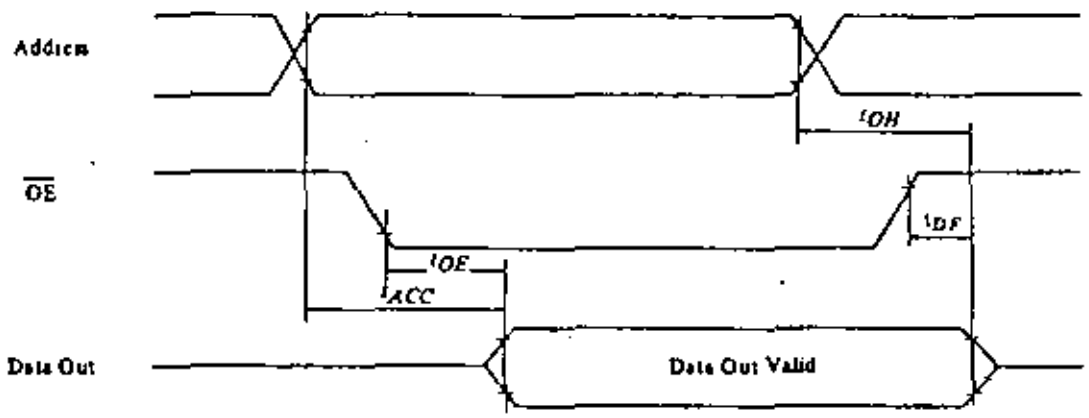


Figura 4-6: Modo de lectura del EPROM 2716

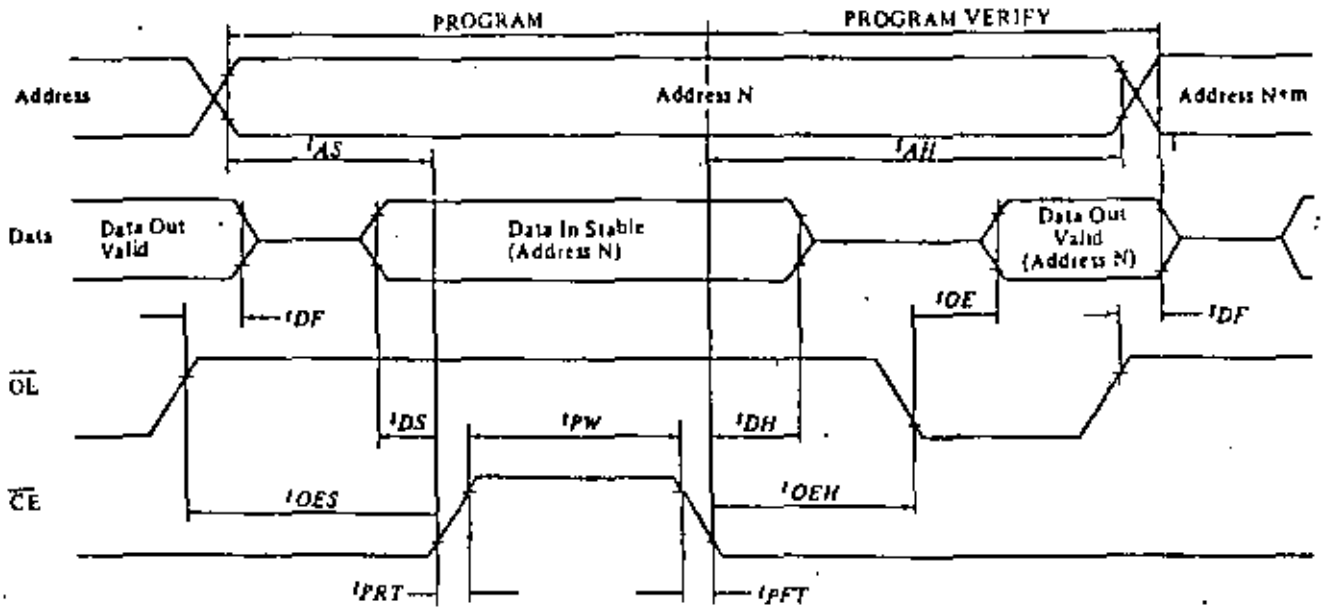


Figura 4-7: Modo de programación y verificación del EPROM 2716

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 aleatoria. 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.

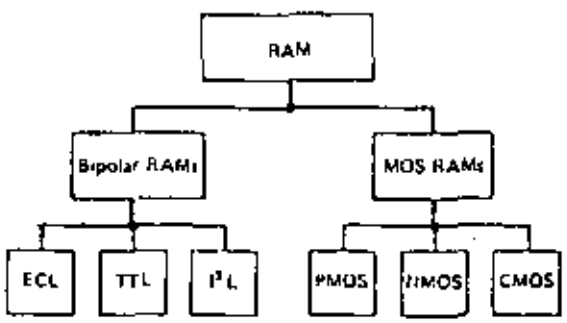


Figura 4-8: Tecnologías en memorias de semiconductores

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 diferencia 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

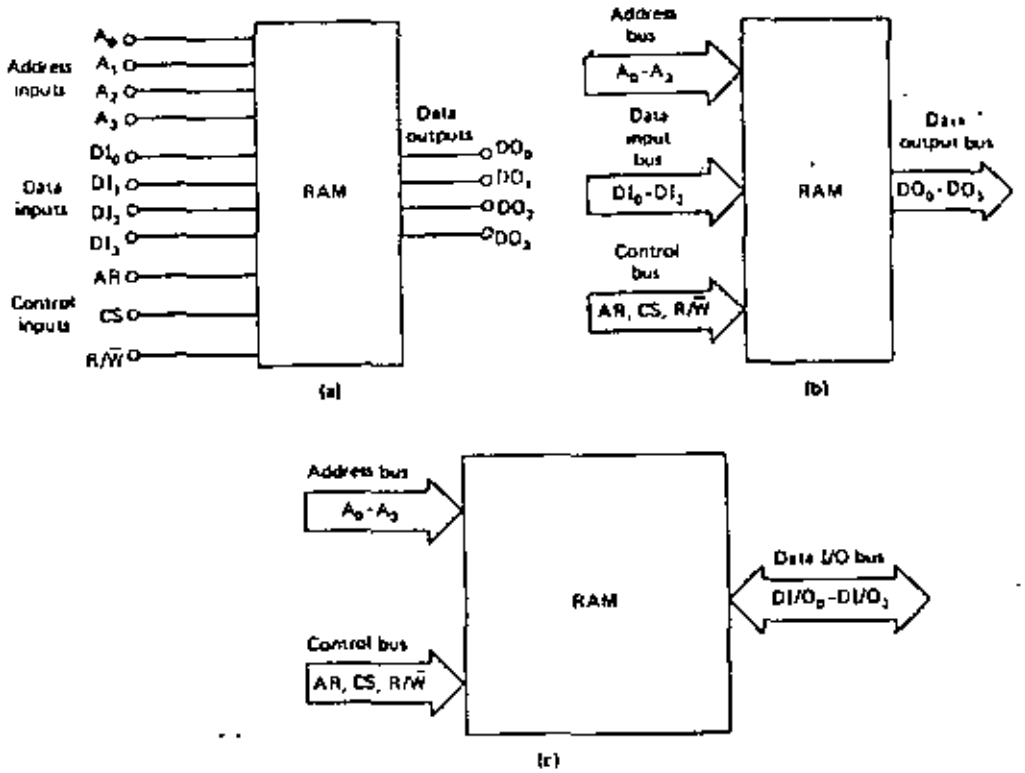


Figura 4-9: Memoria RAM vista desde afuera

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.

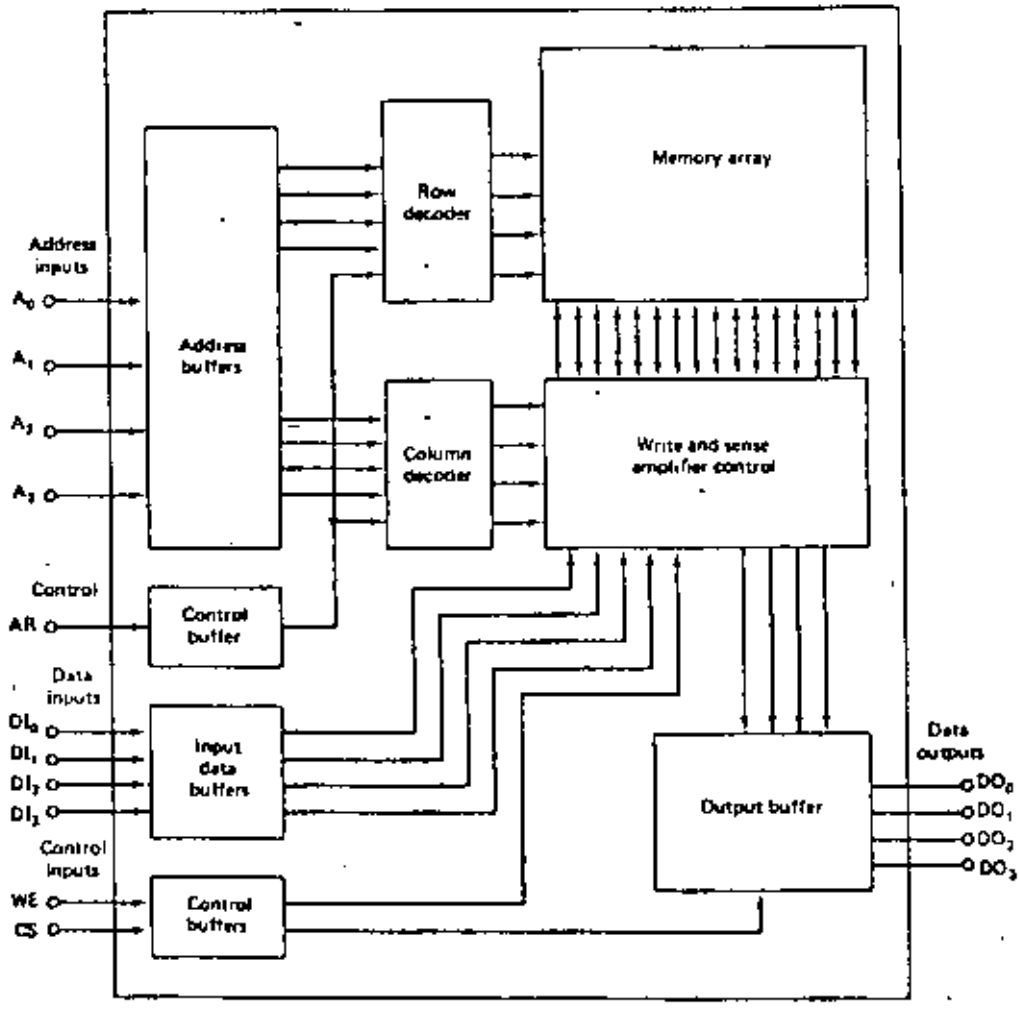


Figura 4-18: Arquitectura típica de una RAM

La figura 4-18 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.

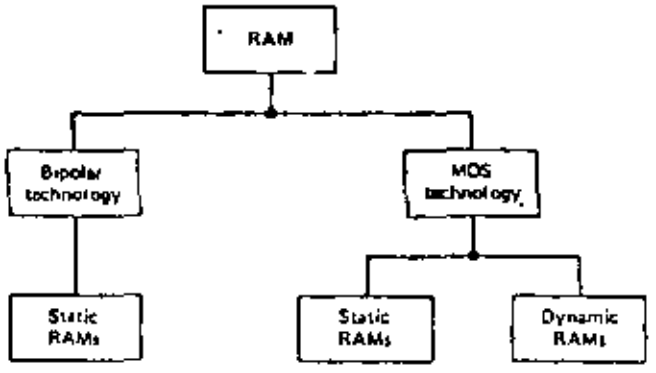


Figura 4-11: Categorías de memorias RAM

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,

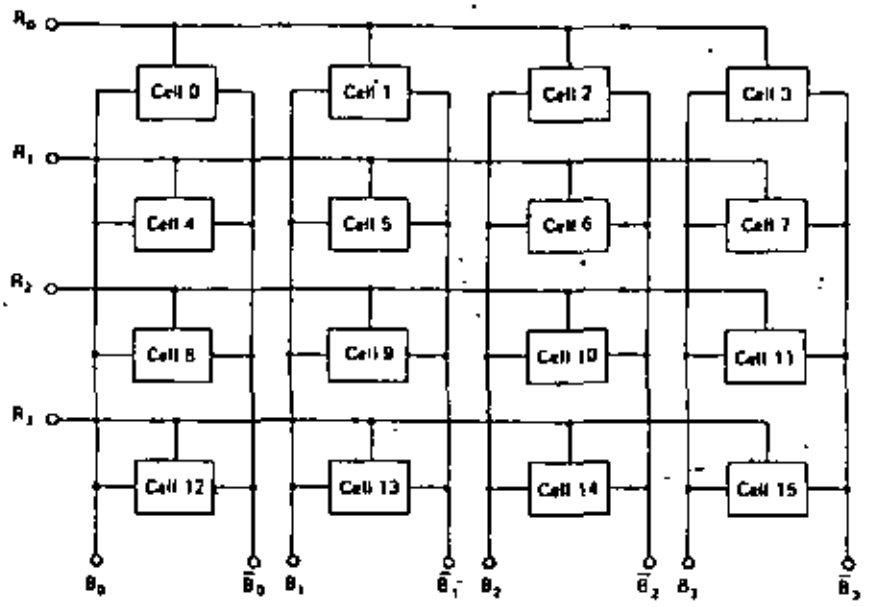


Figura 4-12: Organización interna de las celdas

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.

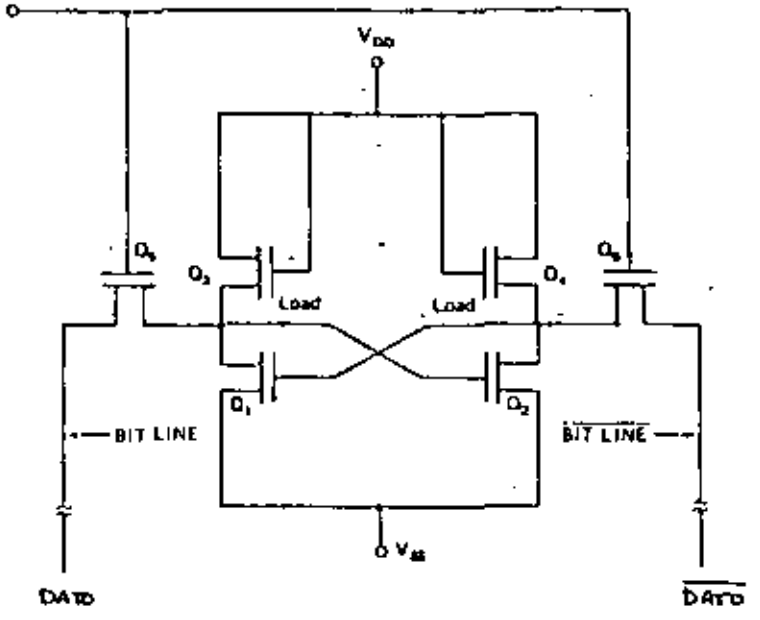


Figura 4-13: Celda de memoria de 6 transistores (estática).

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.

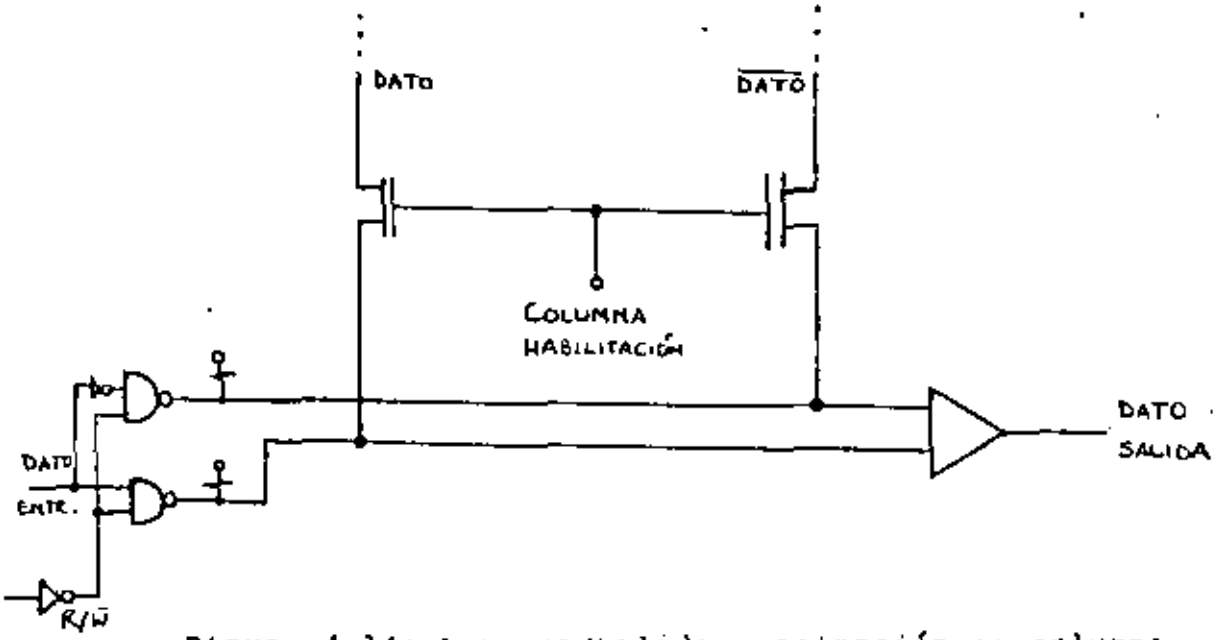


Figura 4-14: Entrada/salida y selección de columna

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 information 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.

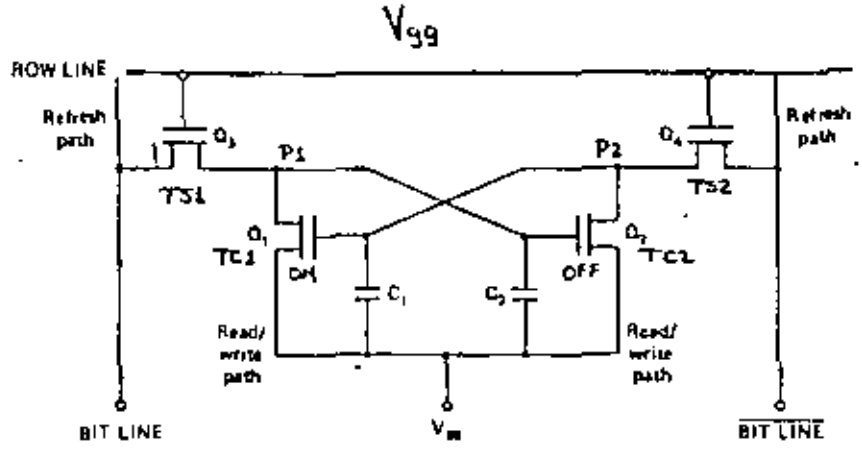


Figura 4-15: Celda de memoria de 4 transistores

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 = 0, 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

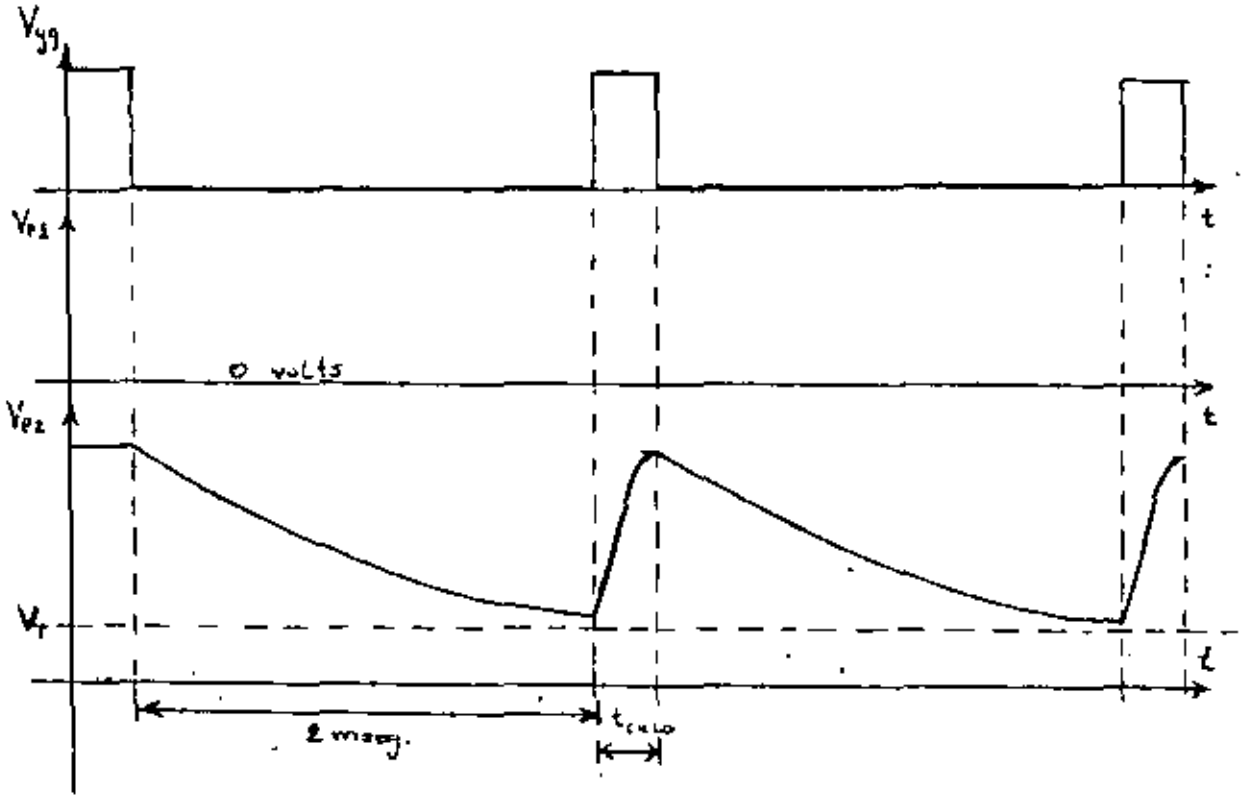


Figura 4-16: Carga y descarga de los capacitores C1 y C2

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 estaban 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

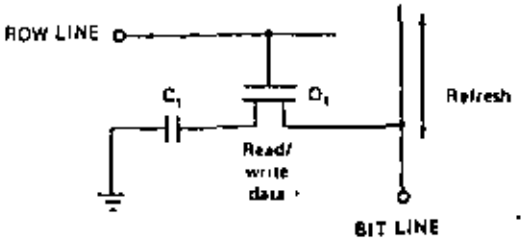


Figura 4-17: Celda dinámica de un solo transistor

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 4 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 organzación cuadrada de 256 x 256 bits.

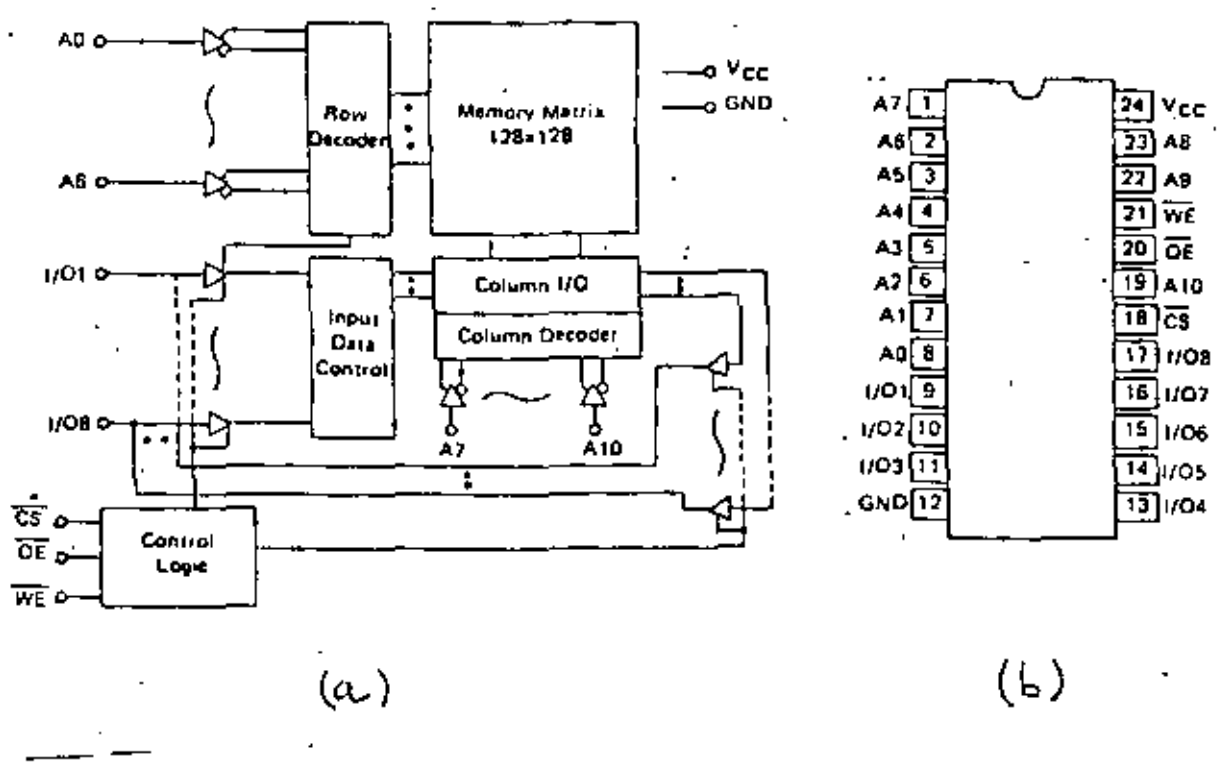


Figura 4-18: RAM CMOS estática de 2K bytes
(a) Diagrama de bloques
(b) Distribución de patas del chip

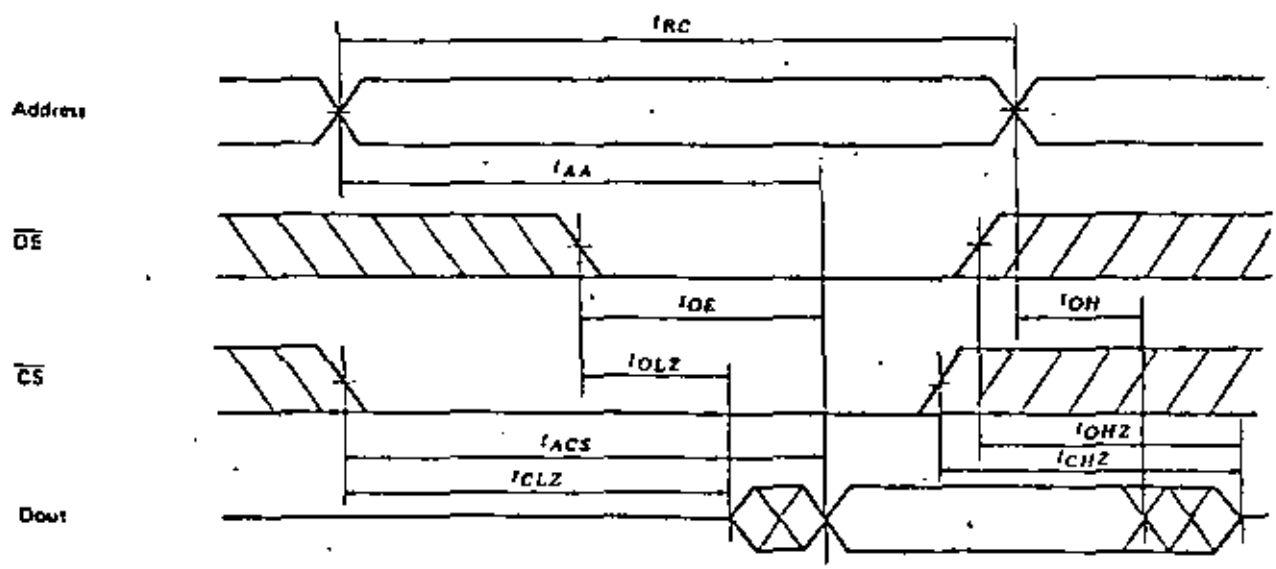


Figura 4-19: Ciclo de lectura de la memoria 6116

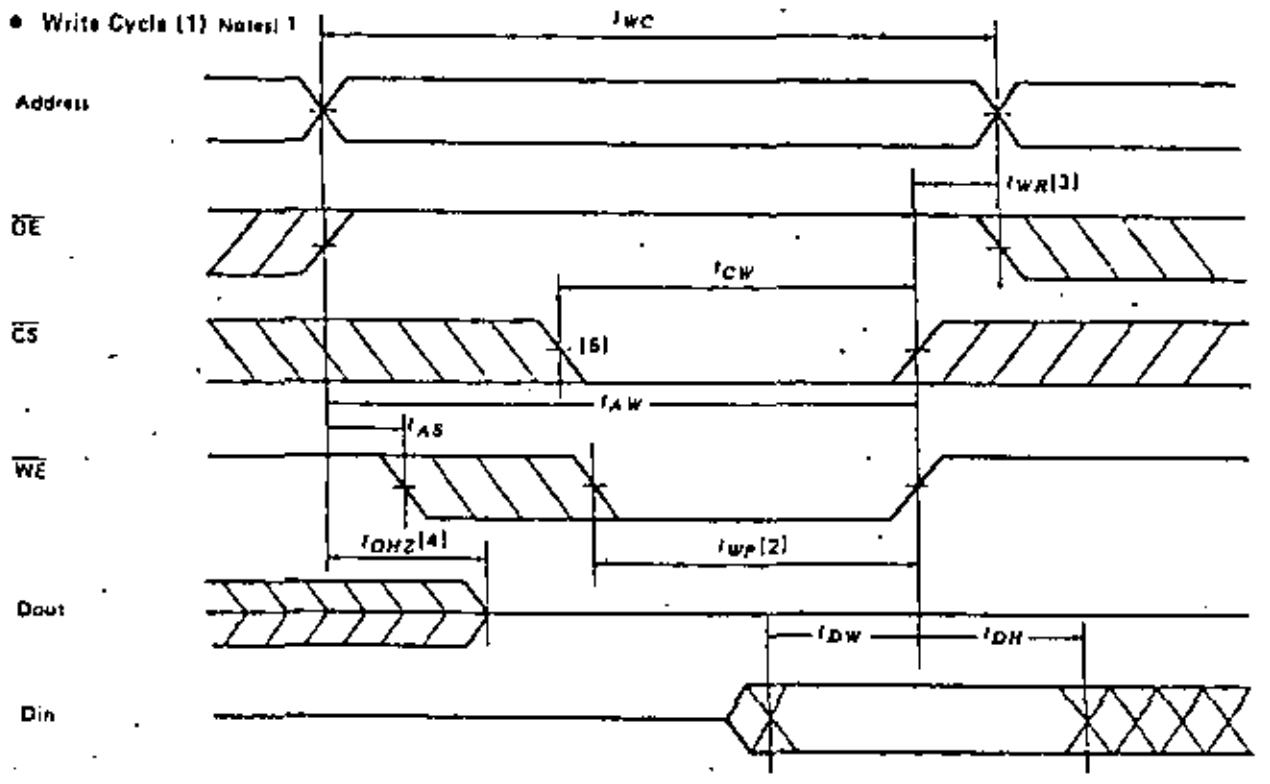


Figura 4-20: Ciclo de escritura de la memoria 6116

• **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**

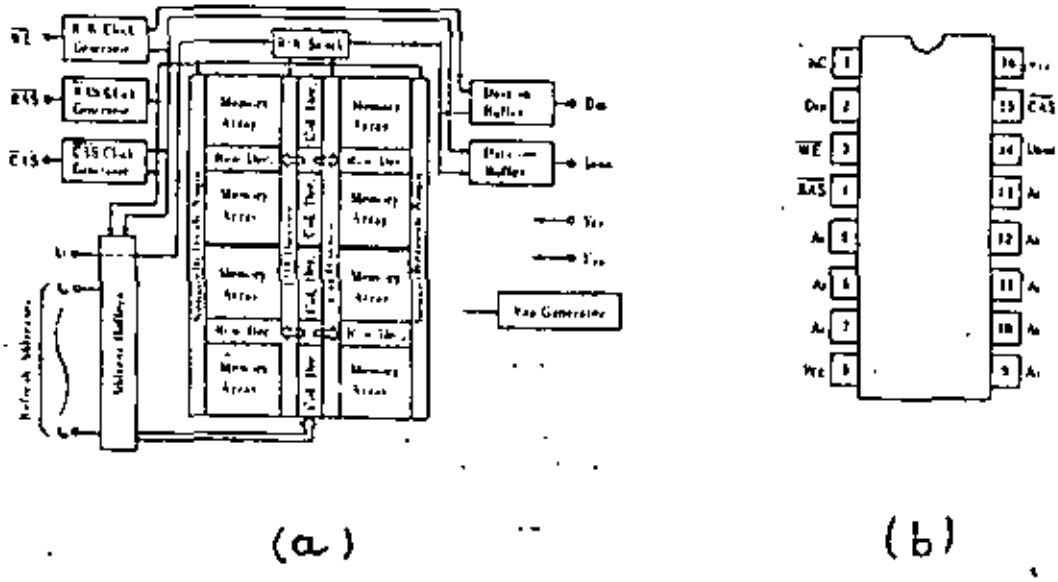
Item	Symbol	HM6116P-2		HM6116P-3		HM6116P-4		Unit
		min.	typ.	min.	max.	min.	max.	
Write Cycle Time	t_{WC}	120	–	150	–	200	–	ns
Chip Selection to End of Write	t_{CW}	70	–	90	–	120	–	ns
Address Valid to End of Write	t_{AW}	105	–	120	–	140	–	ns
Address Set Up Time	t_{AS}	20	–	20	–	20	–	ns
Write Pulse Width	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

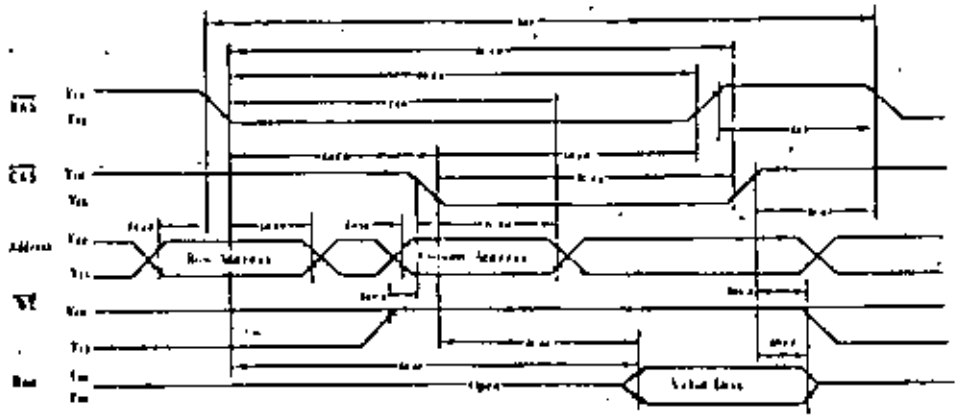


Figura 4-22: Ciclo de lectura de la memoria 4864

• WRITE CYCLE

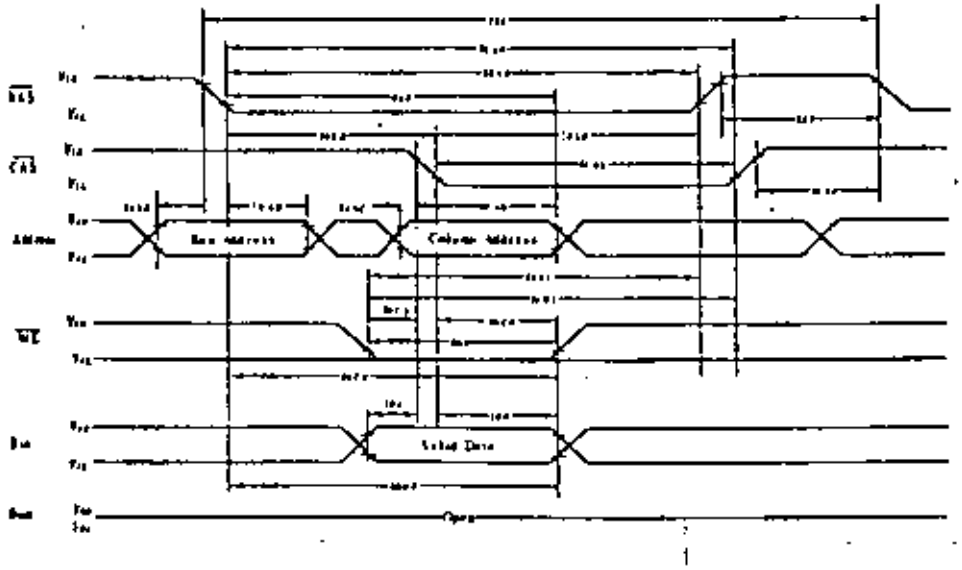


Figura 4-23: Ciclo de escritura de la memoria 4864

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

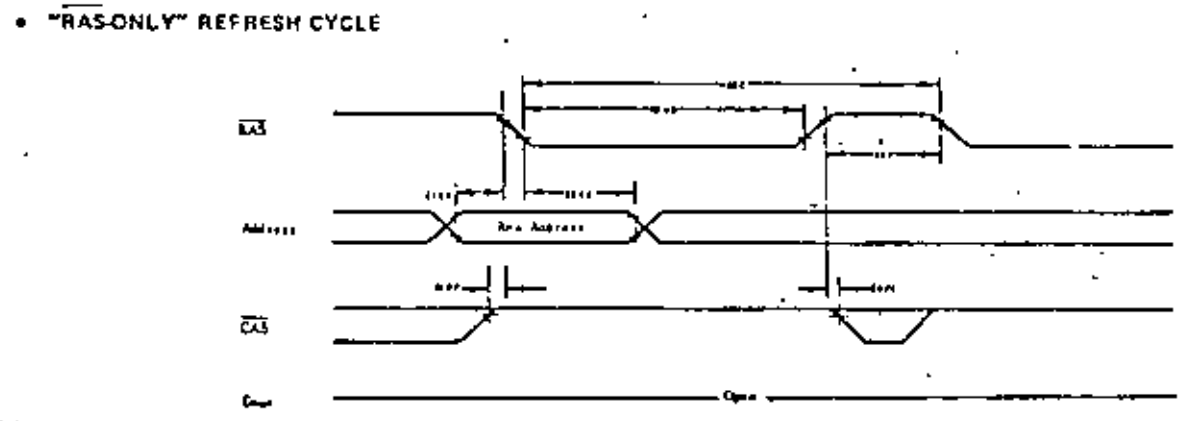


Figura 4-24: Ciclo de refresco para la memoria 4864

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 conectan 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:

- 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.

Requiere una sola fuente de alimentación 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 especificadas 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 16 de las cuales son para conectarse con el CPU, 2 de la fuente de alimentación y 16 de cada puerto.

A continuación se describen 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).

- 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 empaquetado 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.

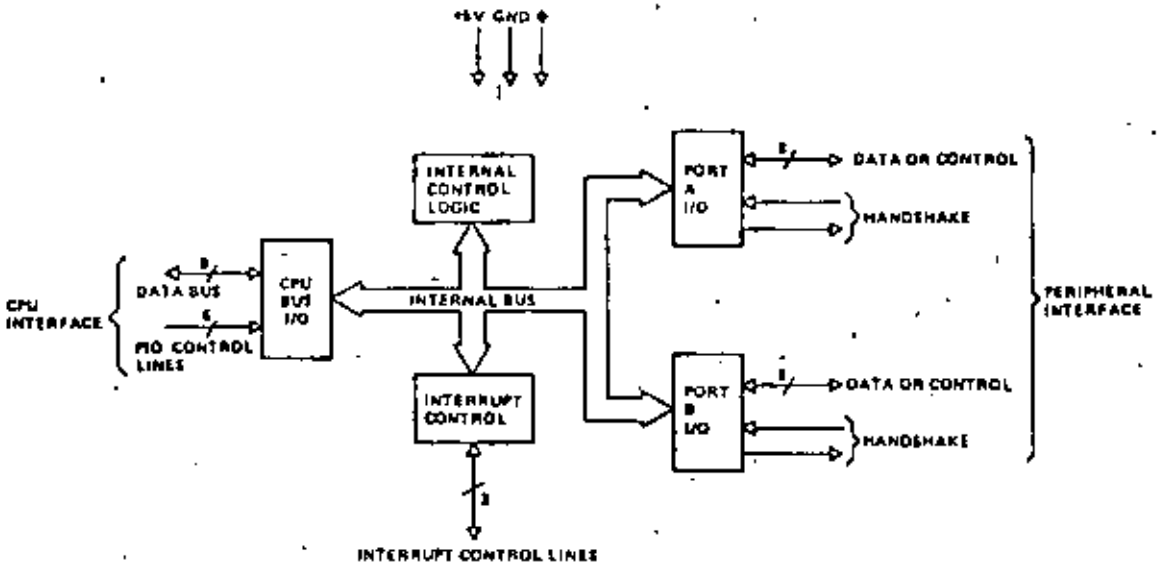


Figura 4-25: Diagrama de bloques interno del PIO

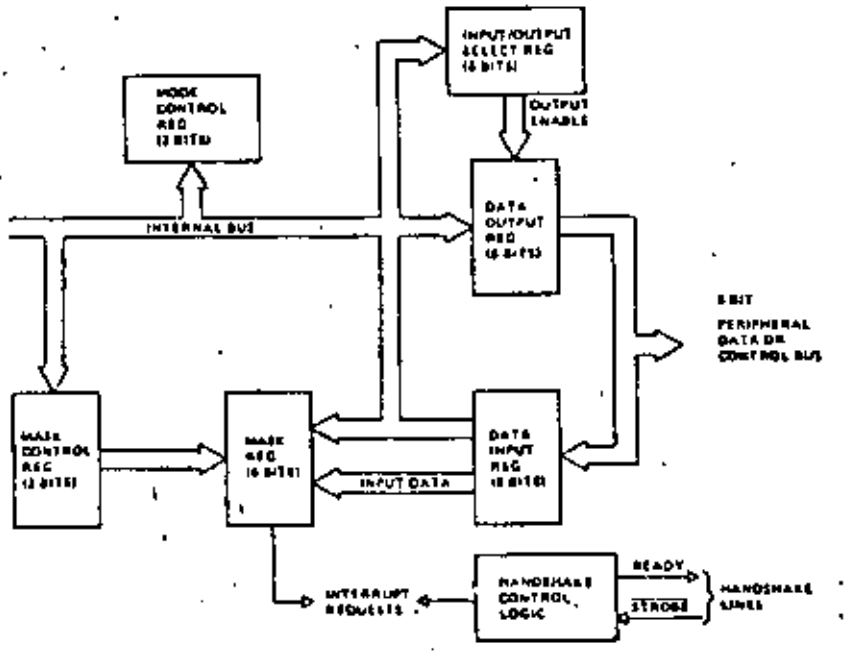


Figura 4-26: Diagrama de bloques del puerto

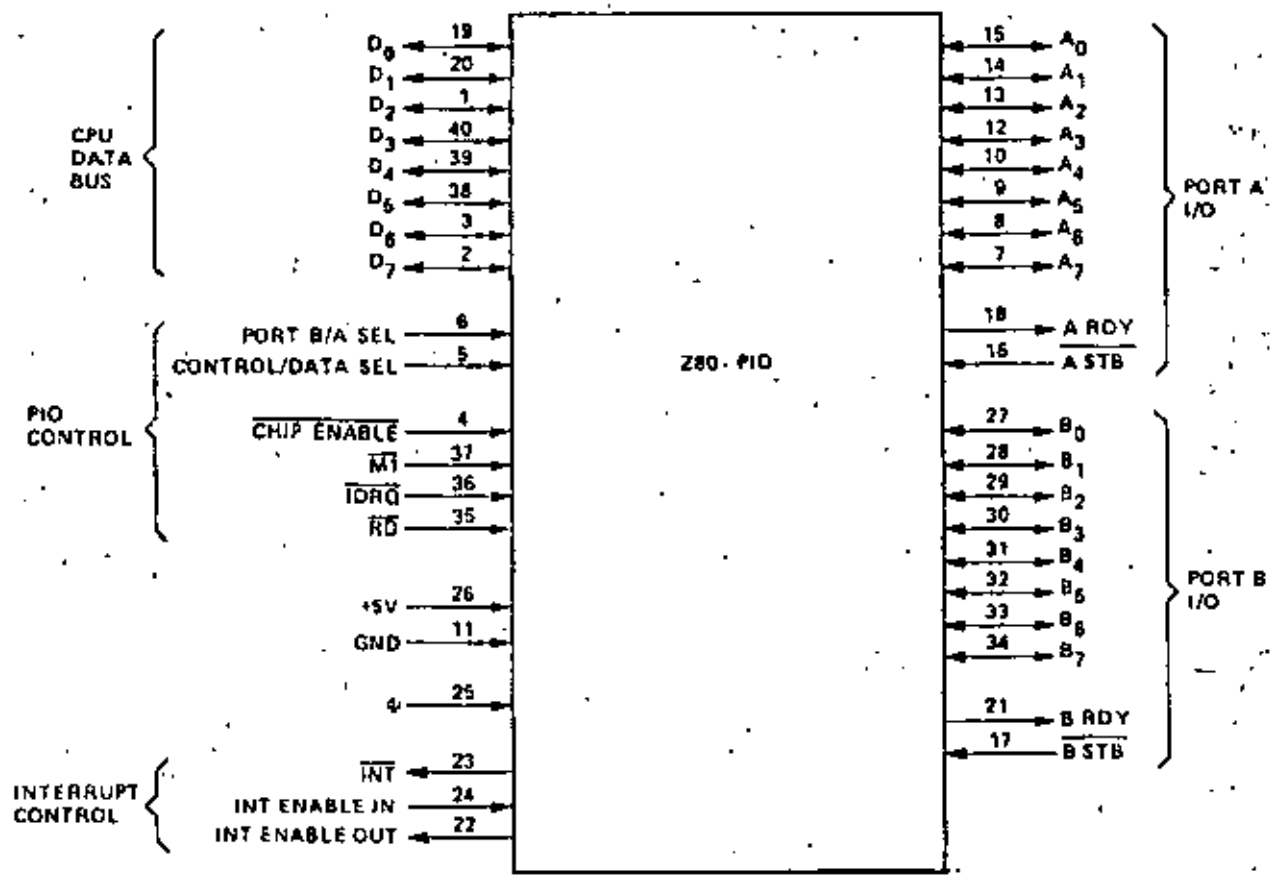


Figura 4-27: Descripción externa del PIO

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 0 en el bit menos significativo (d0).

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	D0	MODO	
0	0	x	x	1	1	1	1	0	salida
0	1	x	x	1	1	1	1	1	entrada
1	0	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 0, 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 0 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 0 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 0 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 0 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 0 el estaoo 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 = 0 y D2 = D1 = D0 = 1.

La mascara debe tener 0 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.

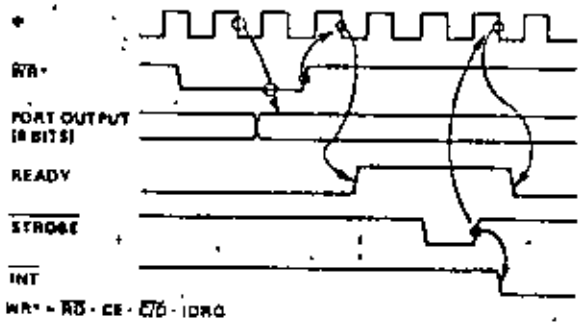


Figura 4-28: Modo 0 salida del puerto

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

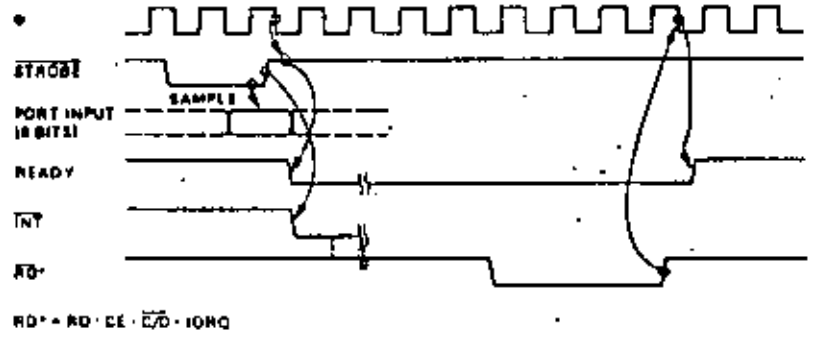


Figura 4-29: Modo 1 entrada al puerto

El modo bidireccional sirve solamente para el puerto A ya que usa las 4 señales de protocolo (handshake), 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 (handshake).

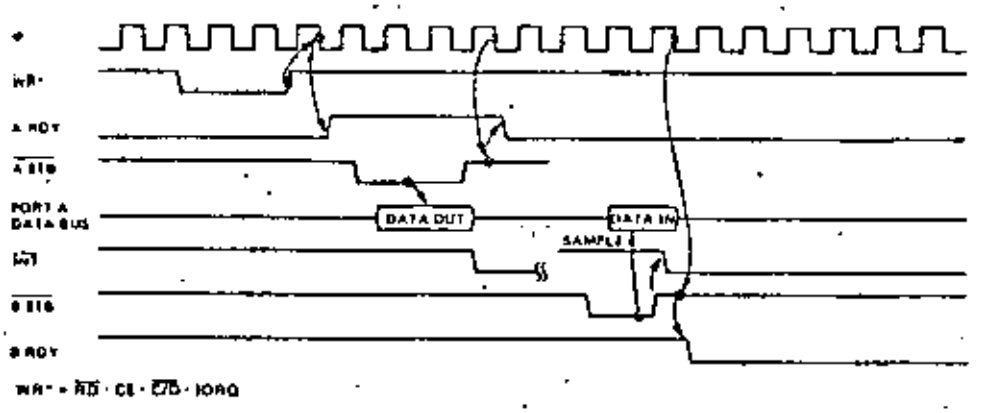


Figura 4-30: Puerto A en modo 2, bidireccional

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.

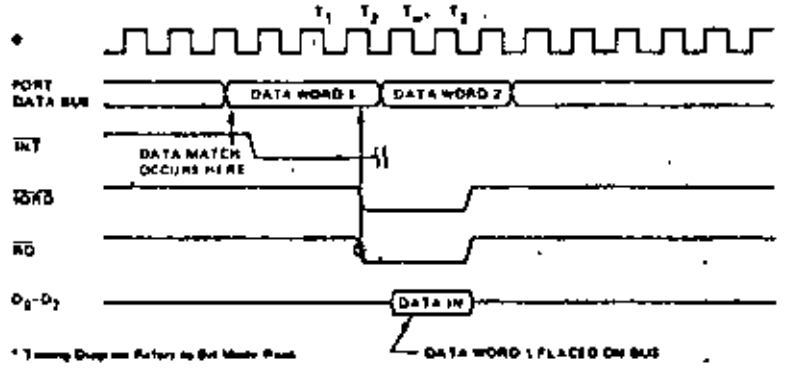


Figura 4-31: Modo 3, bits independientes

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.

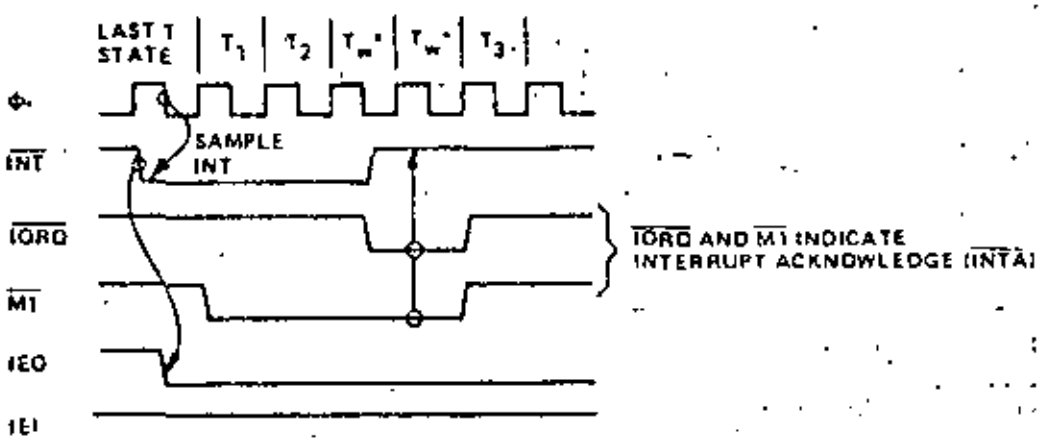


Figura 4-32: Ciclo de reconocimiento de interrupción

La señal de IEO del PIO cambia a 0 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

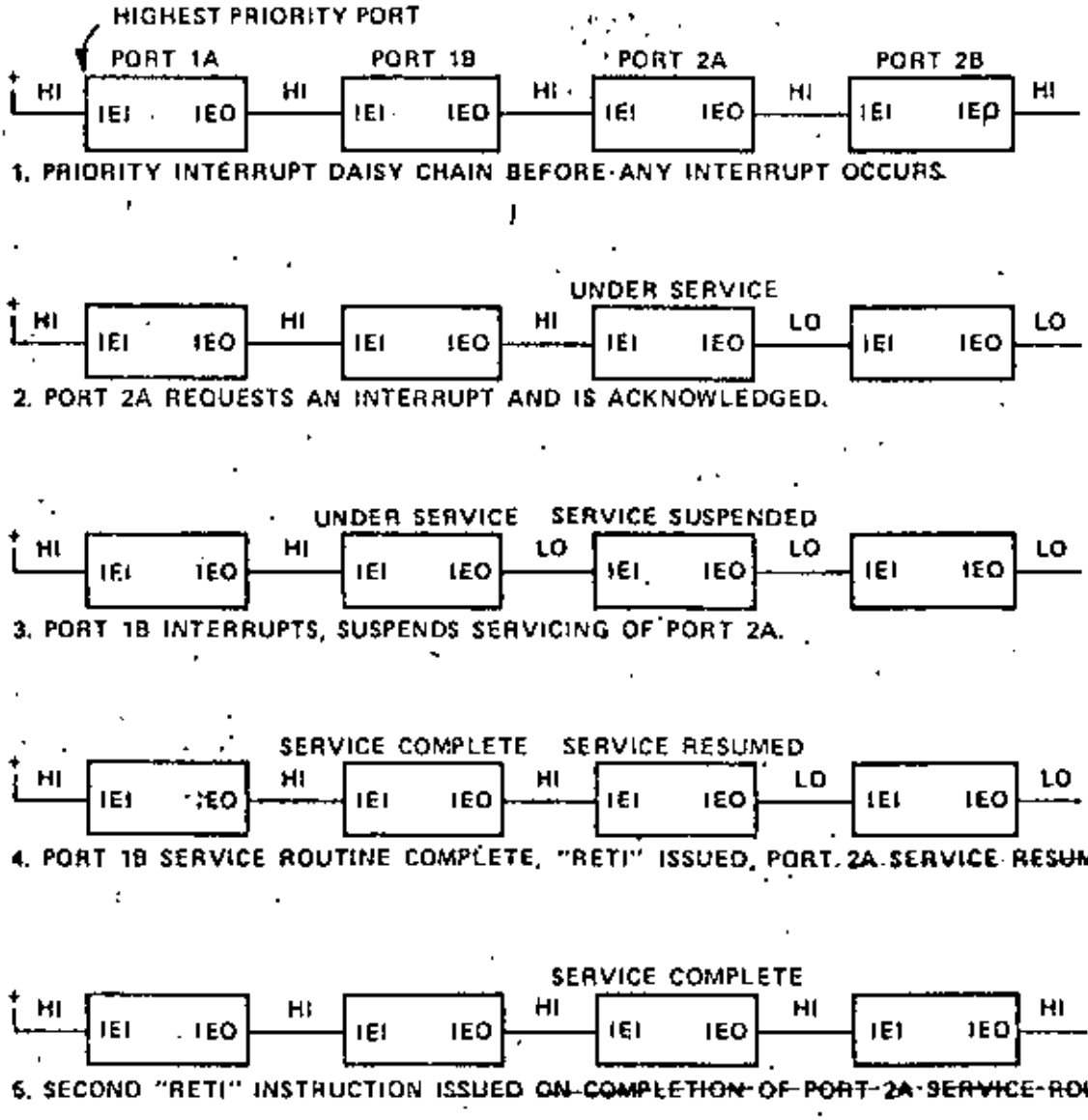


Figura 4-33. Atención de interrupciones en una estructura de prioridades del tipo cadena

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.

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.

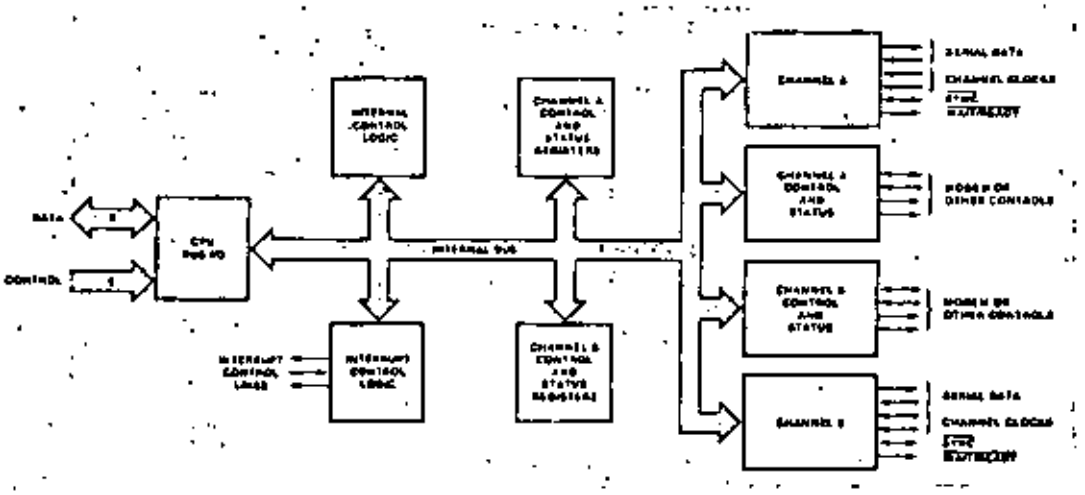


Figura 4-34: Diagrama de bloques del SIO

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.

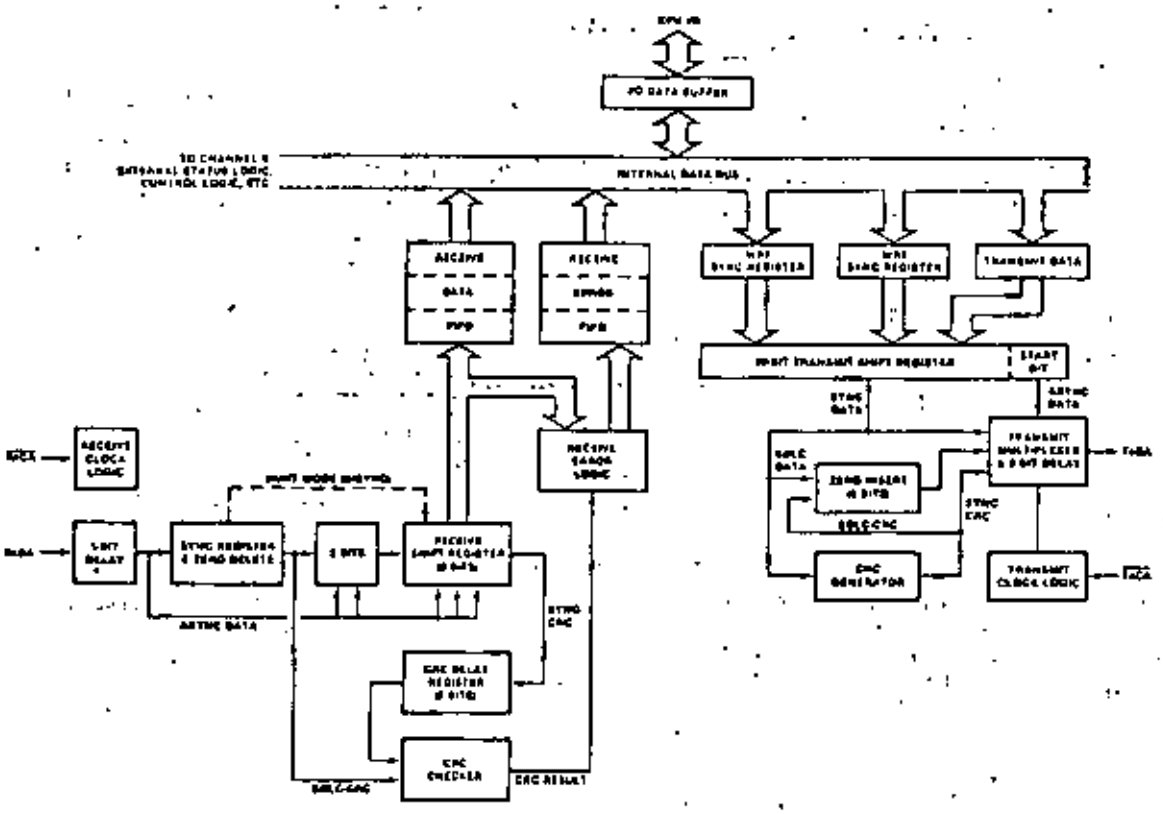


Figura 4-35: Diagrama de bloques interno del canal

- 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/chequeador 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.
Φ	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 butter 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 hhubo 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.

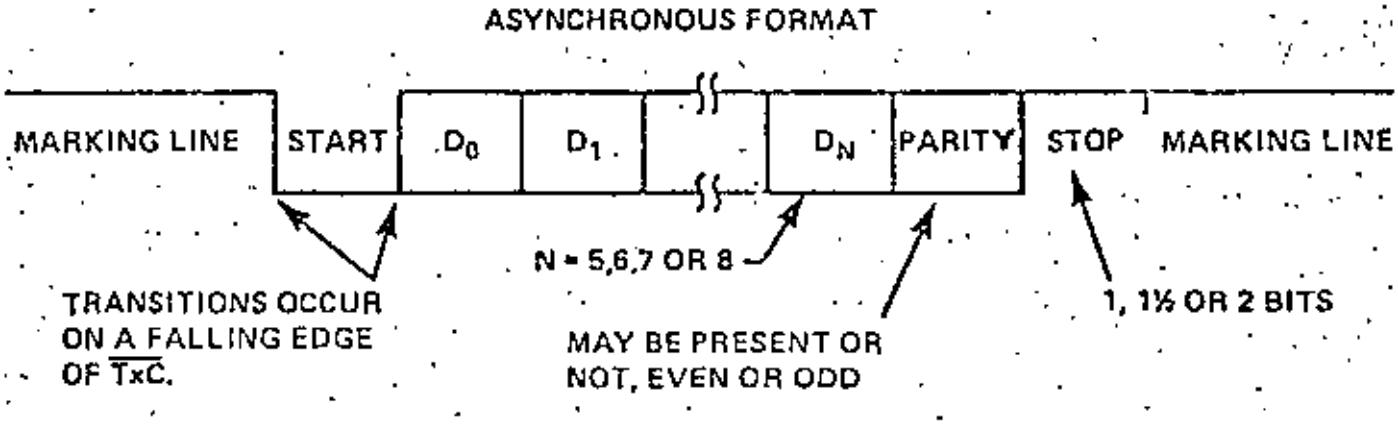


Figura 4-36: Formato de operación en modo asíncrono

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

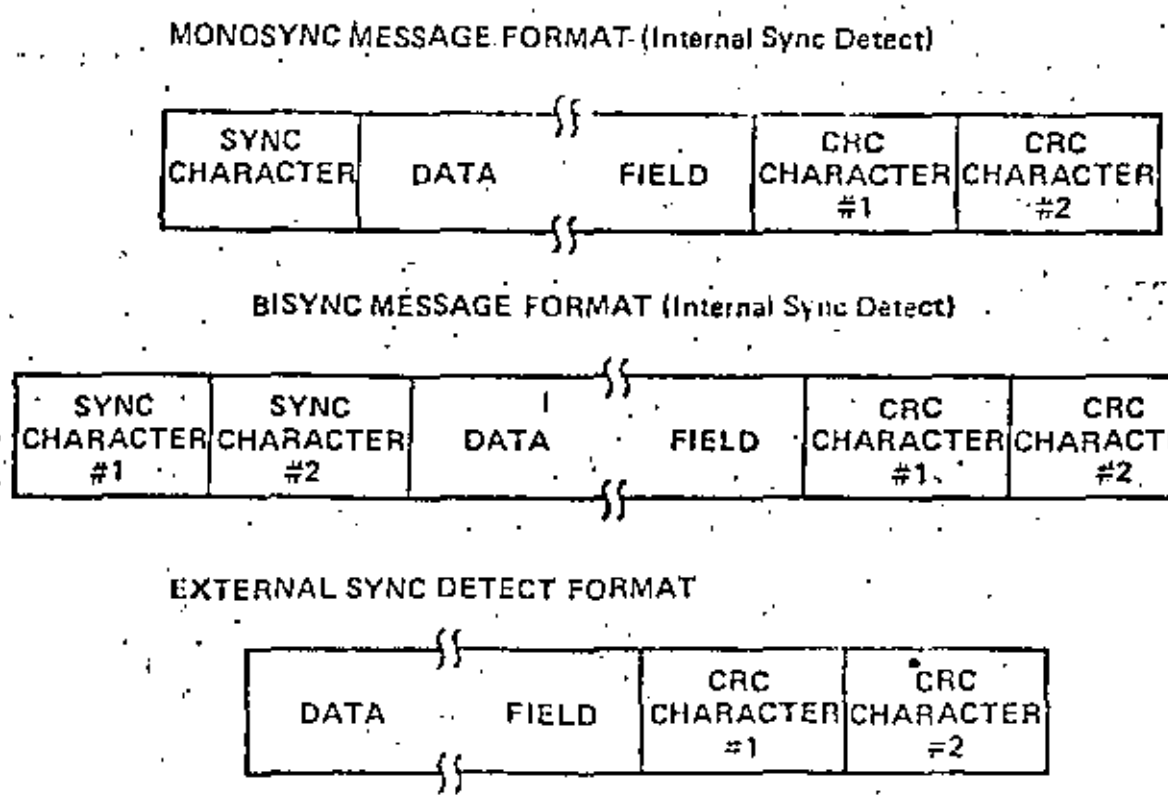


Figura 4-37: Formato de operación en modo síncrono

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

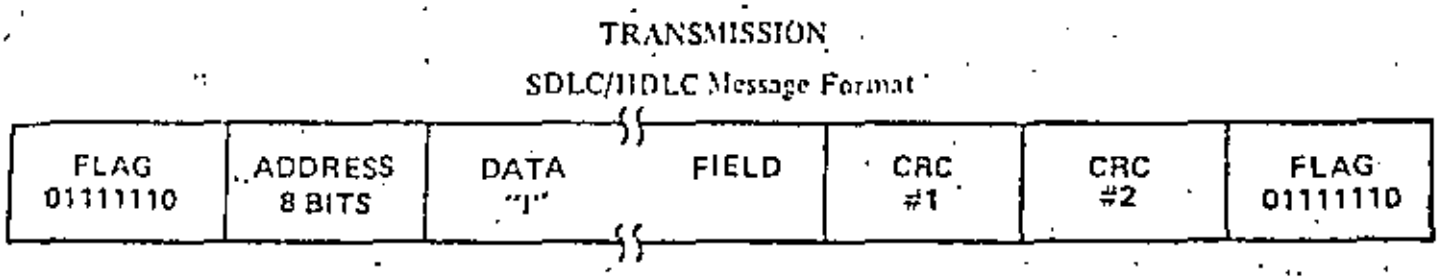


Figura 4-38: Formato para el modo síncrono SDLC/HDLC

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).

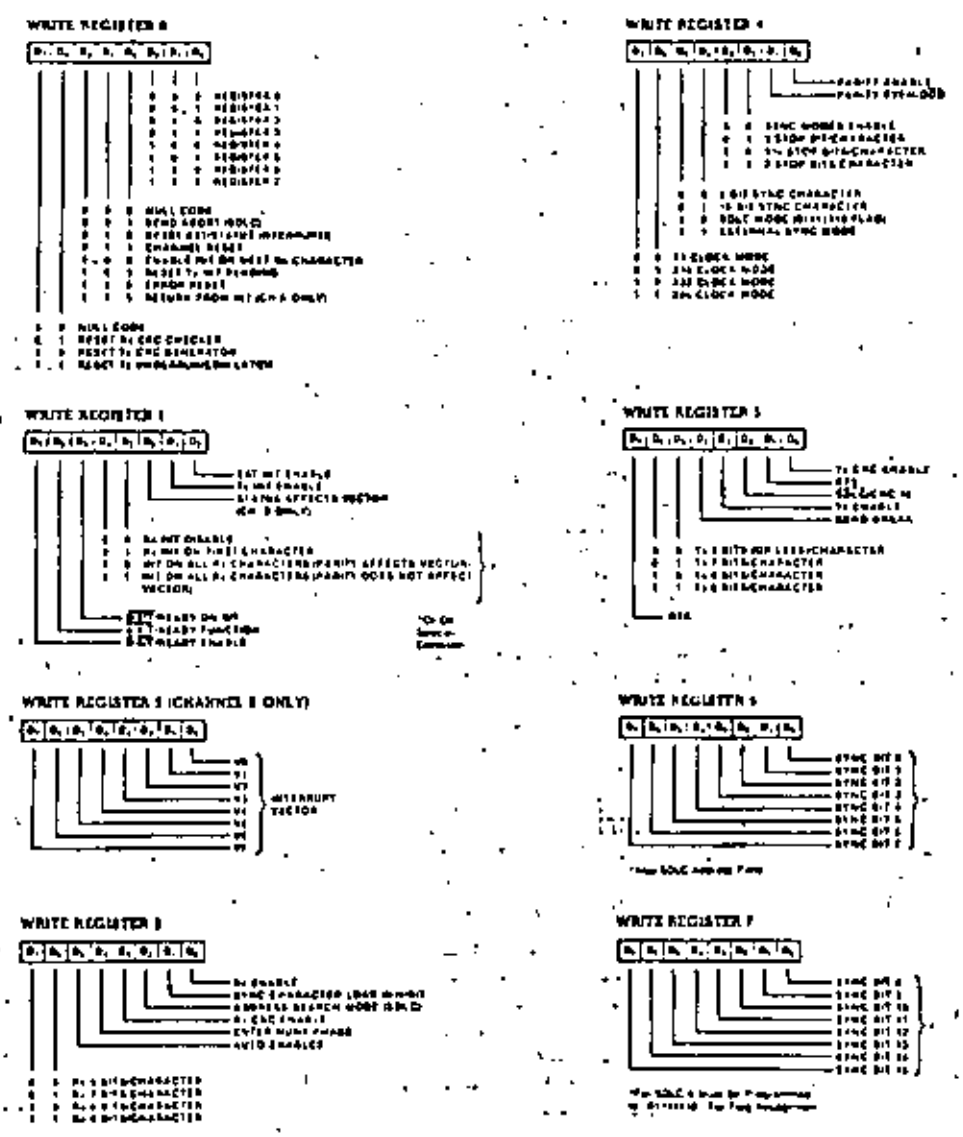


Figura 4-39: Funciones de los registros de comandos

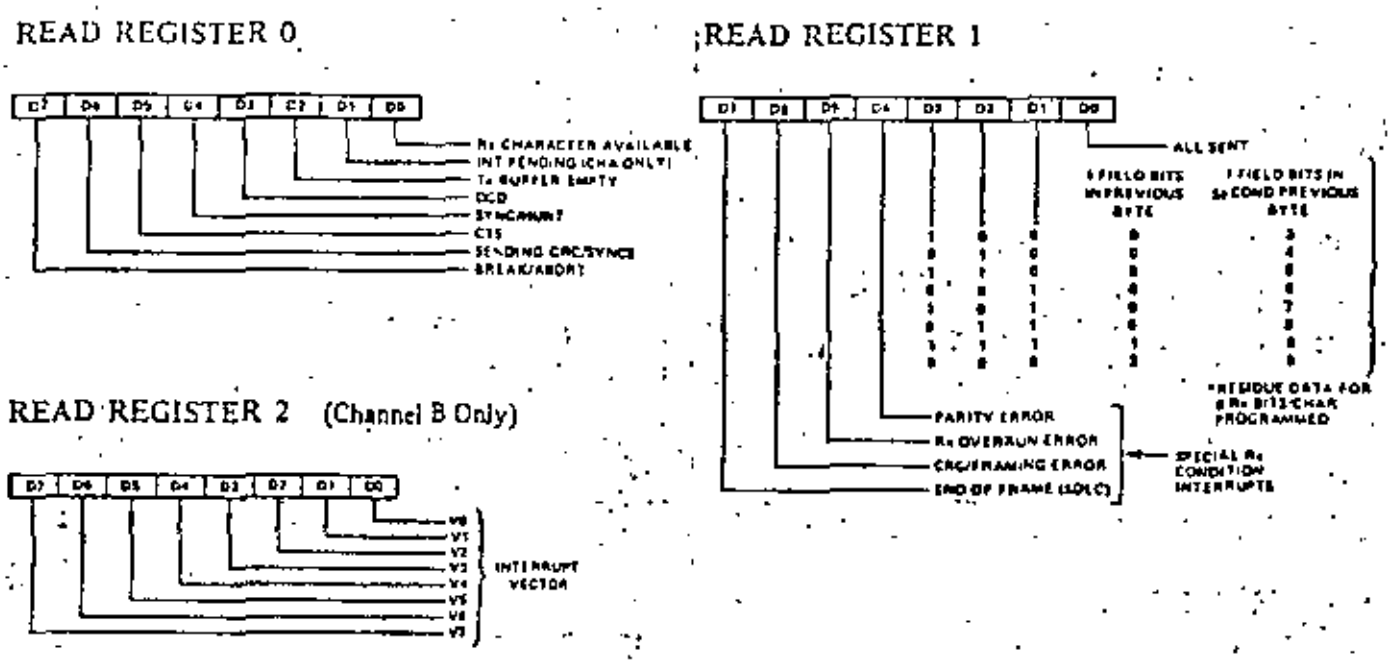


Figura 4-40: Funciones de los bits de los registros de lectura

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 "I" 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 unicamente 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

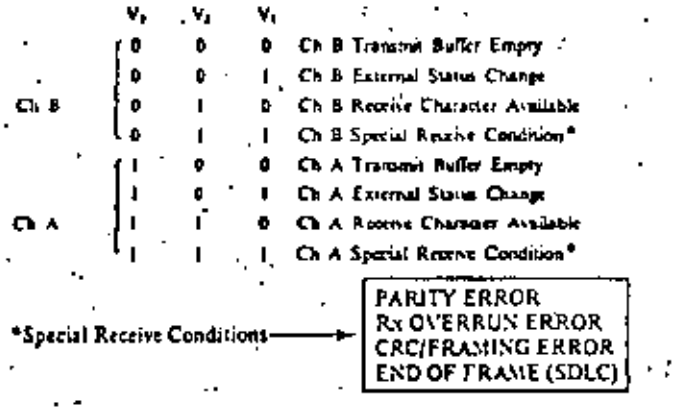


Figura 4-41: Forma en como se afecta el vector de interrupción

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 "Winchester", 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.

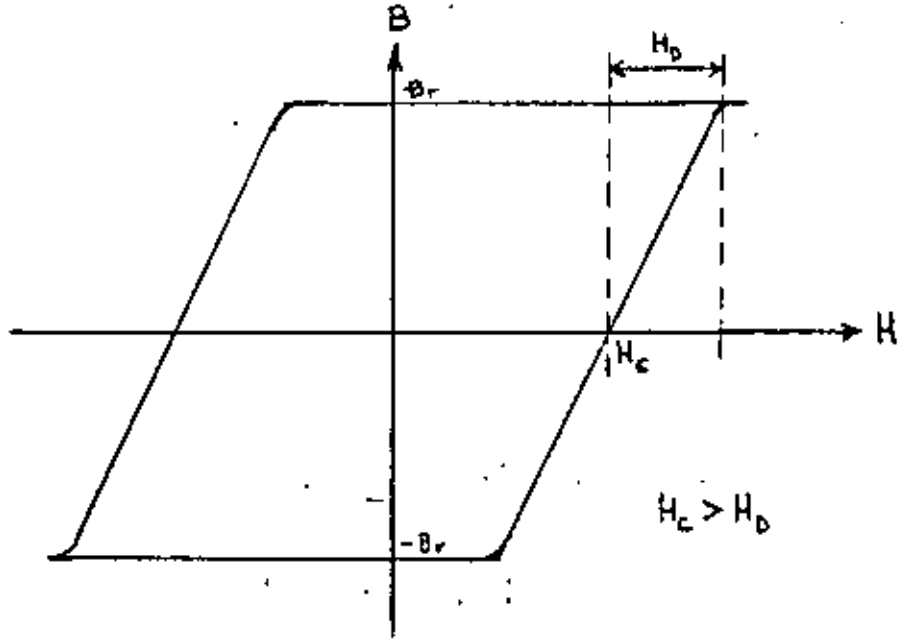


Figura 4-42: Comportamiento de un material ferromagnético, ciclo BH

Las propiedades del medio magnético:

1. Tamaño de la partícula en el oxido y el tamaño del grano del medio no magnético.
2. Fuerza coercitiva 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).

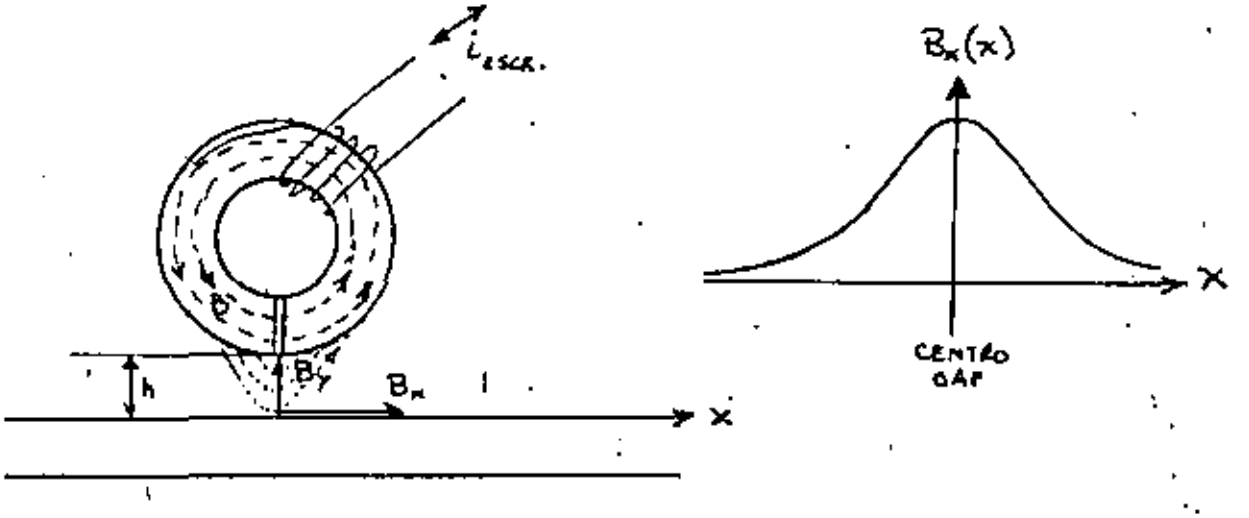


Figura 4-43: Flujo magnético en la cabeza

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.

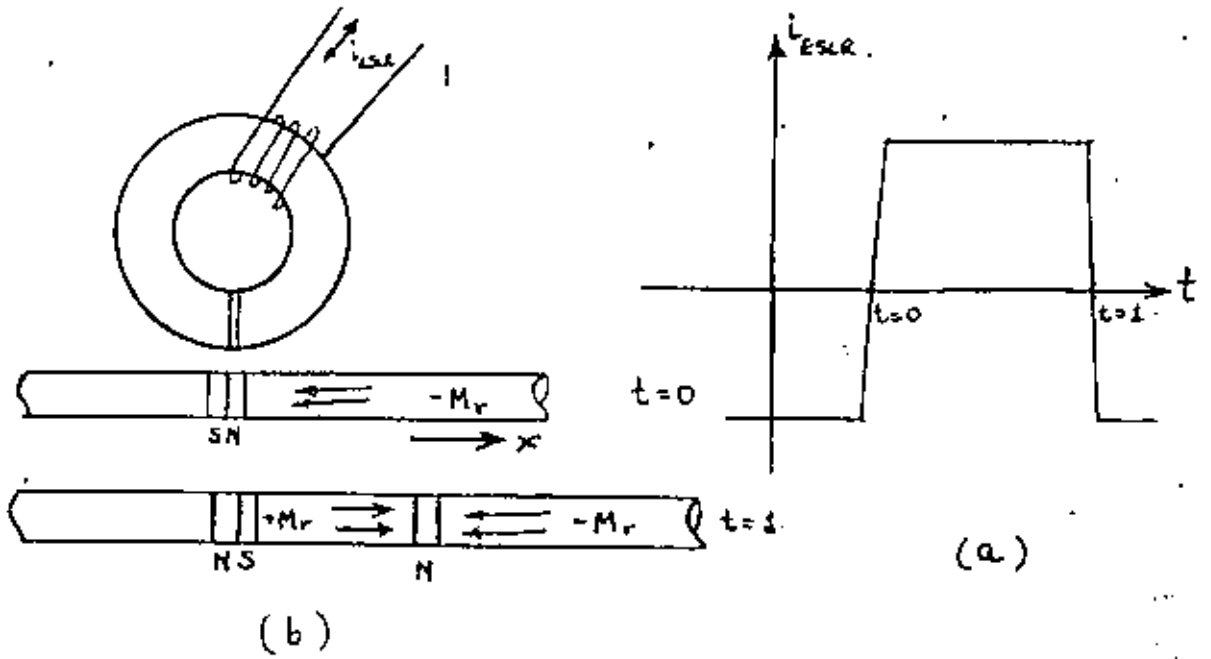


Figura 4-44: Efectos de la grabación magnética en el medio
(a) Corriente de escritura
(b) Dipolos magnéticos en el medio

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.

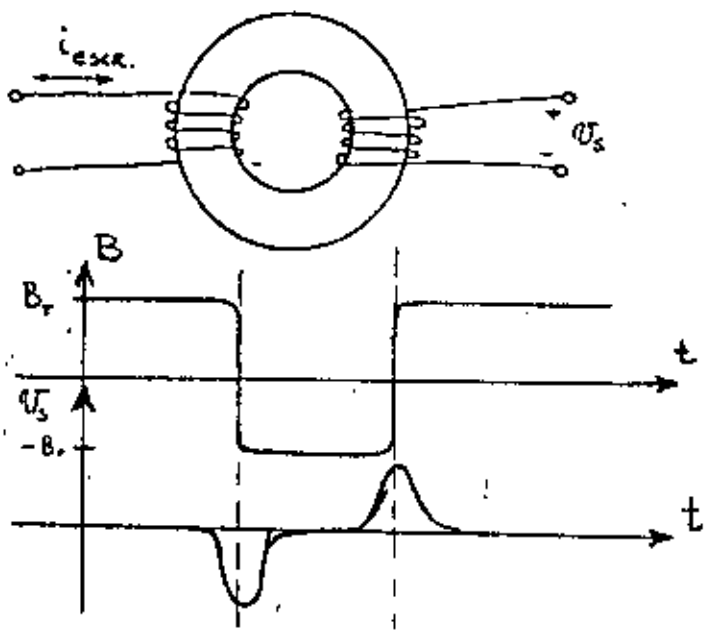
4.4.1.3. MECANISMO DE SENSADO O LECTURA

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

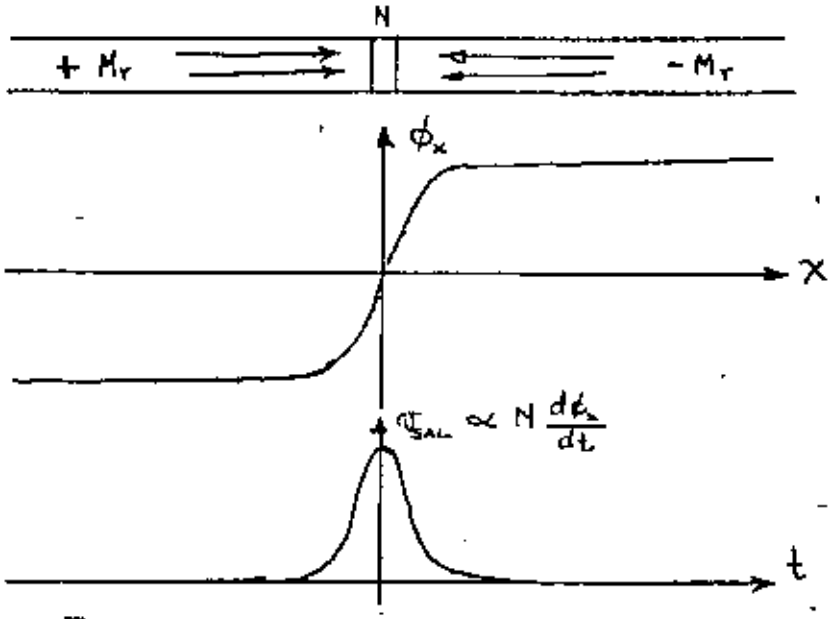


Figura 4-46: Señal de sensado

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.

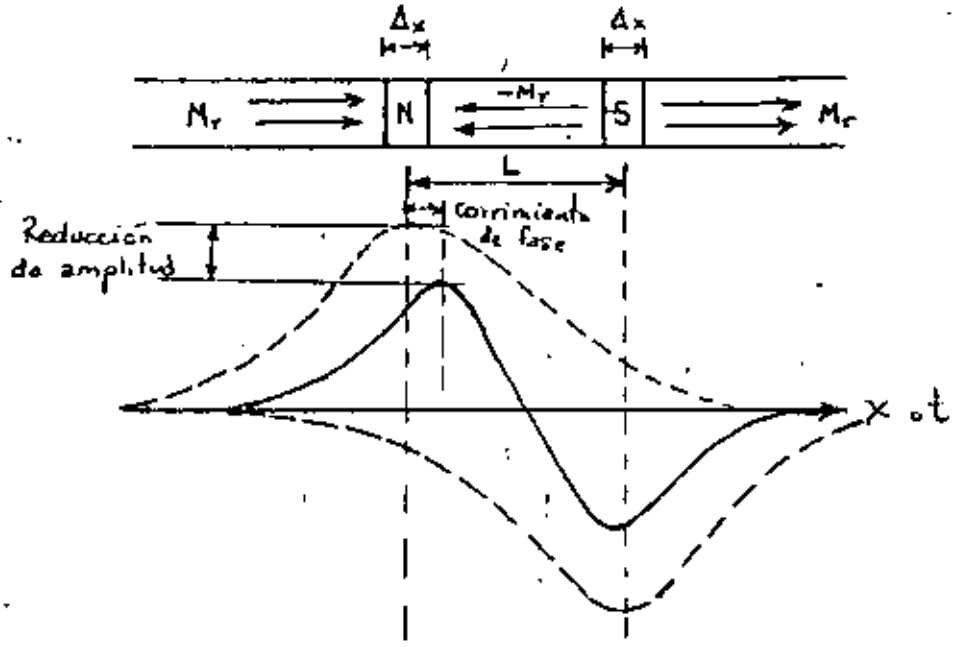


Figura 4-47: Efecto de juntar mucho las transiciones

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,

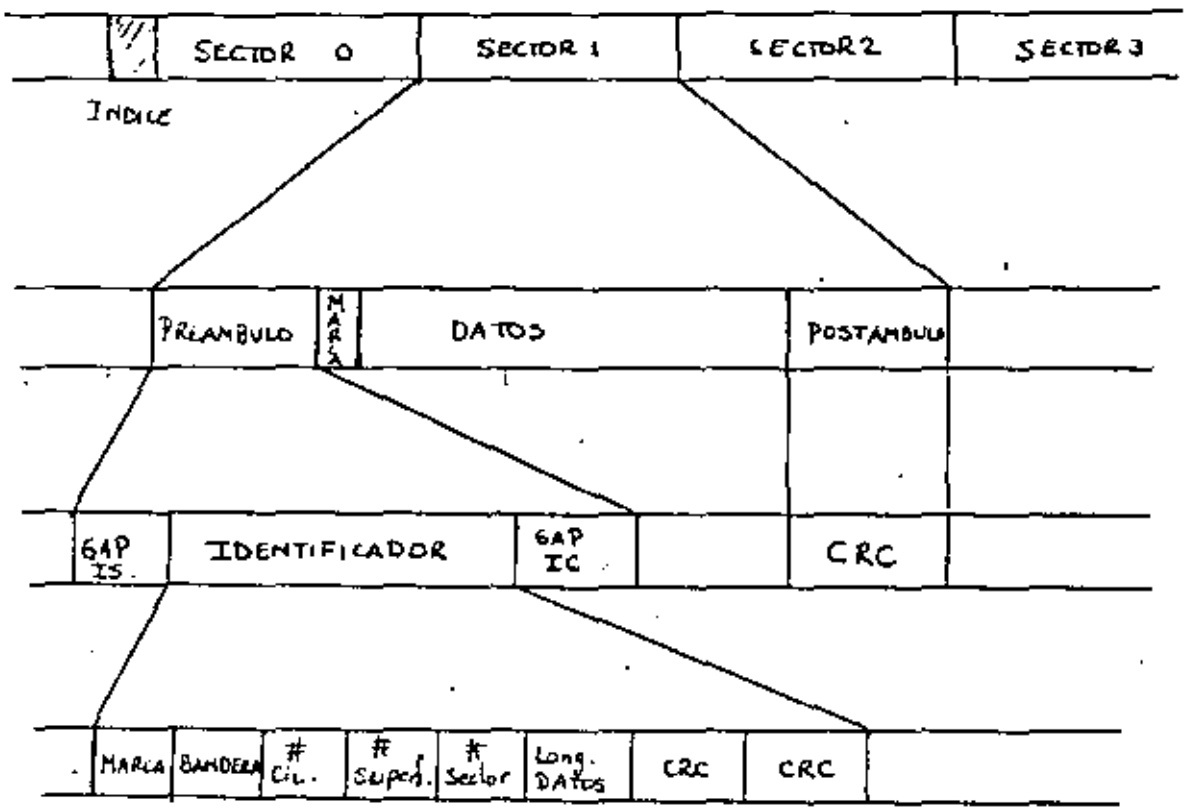


Figura 4-48: Formateo de una pista de disco

un reloj para identificar los datos, el cual puede ser externo al código o bien estar incluido dentro del mismo, este último caso se conoce como código con reloj implicito o inherente.

2. Sentido de las polaridades generadas por el mismo código.

4.4.2.1. RELOJ DE SINCRONIZACION

Existen dos tipos de reloj para sincronizar la recuperación de los datos, un reloj externo o bien uno implicito dentro del mismo código.

El reloj externo son pulsos a determinada frecuencia que se usan para determinar los momentos del sensado de la señal. A medida que la densidad se incrementa las variaciones de velocidad del disco o de la cinta son más significativas y aunque se tenga un reloj muy preciso los intervalos de sensado no son constantes por lo que se dificulta cada vez más recuperar la información en forma precisa.

El reloj implicito o innerente dentro del código reduce los problemas de sincronización, porque en este caso las variaciones de velocidad de la cinta o el disco no afectan, ya que en la misma proporción varia el reloj. Códigos con reloj implicito facilitan la grabación a mayores densidades, sin embargo, la densidad física se reduce ya que hay que gastar espacio para grabar el código.

4.4.2.2.. CODIGO NRZI

Este código es llamado no retorno a cero, ya que la señal de escritura no regresa al nivel cero nunca, siempre está variando entre Ir y -Ir.

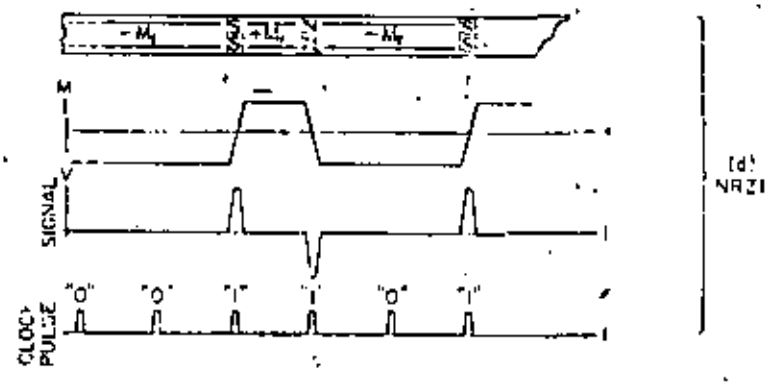


Figura 4-49: Código NRZI

En este código el 1 representa una transición (no importa la polaridad) y el cero no tiene transición. Este código es muy usado en cintas magnéticas a 800 bpi (bits por pulgada). Este código no tiene reloj implicito por lo que hay que usar un reloj externo para sensar la información, lo cual implica que no se puede usar en alta densidad. La detección de cadenas de ceros es problematica.

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.

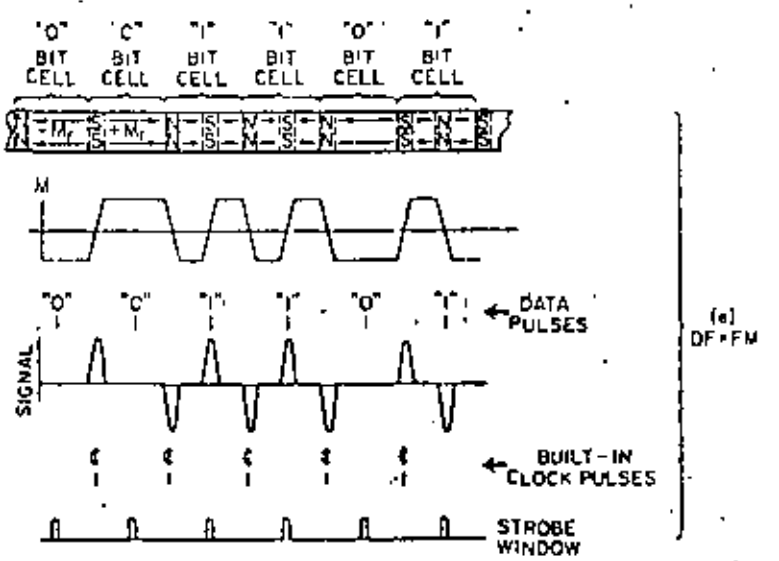


Figura 4-50: Código Fm, Frecuencia Modulada

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

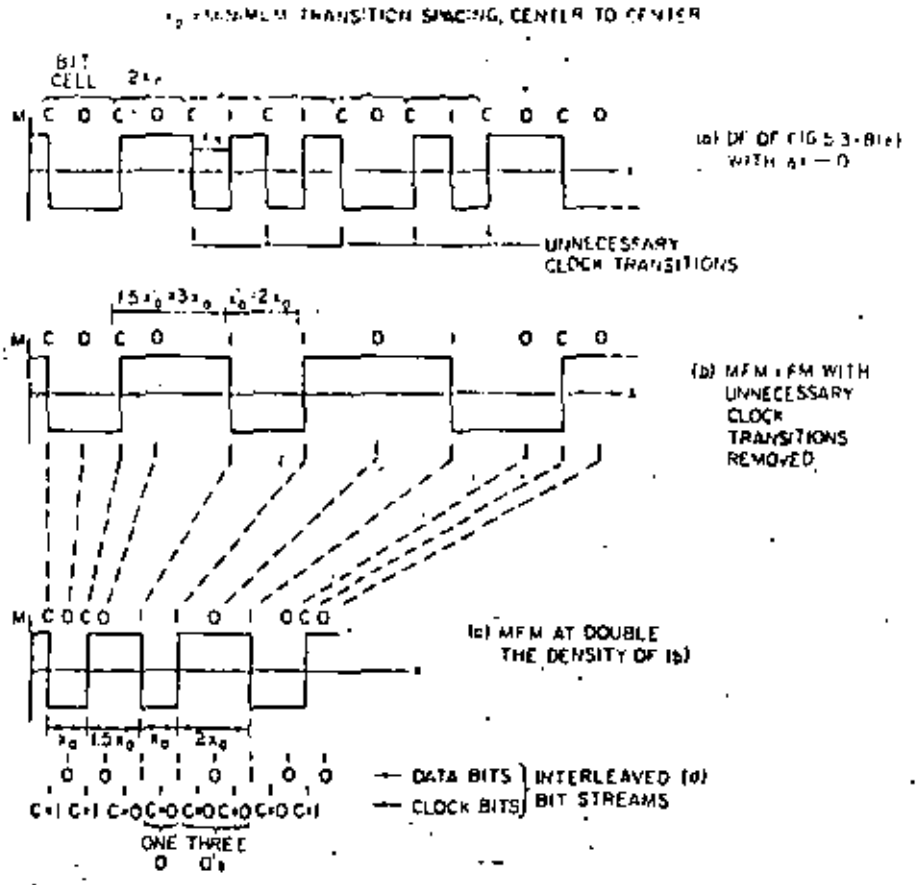


Figura 4-51: Código MFM o código Miller

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, pudiéndose 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	3	1200	FSK, NRZI
Digital	7	9600	NRZI, PE
Limite mecanico		32000	

Tabla 4-5: Comparación entre cassettes

La clasificación de estos cassettes se basa en la calidad de la cinta, o sea la cantidad de oxido de hierro que tengan, uniformidad y pulido de la misma. Mientras más barato es el cassette tiene menos oxiuo y no es uniforme, pueden existir zonas donde no tenga oxido, lo cual resultaría catastrótico para la grabación digital y no tanto para la grabación analógica (audio).

4.4.3.1. CASSETTE DE AUDIO

En los cassettes de audio siempre se debe borrar antes de grabar nuevamente, la baja distorsión es el requerimiento principal.

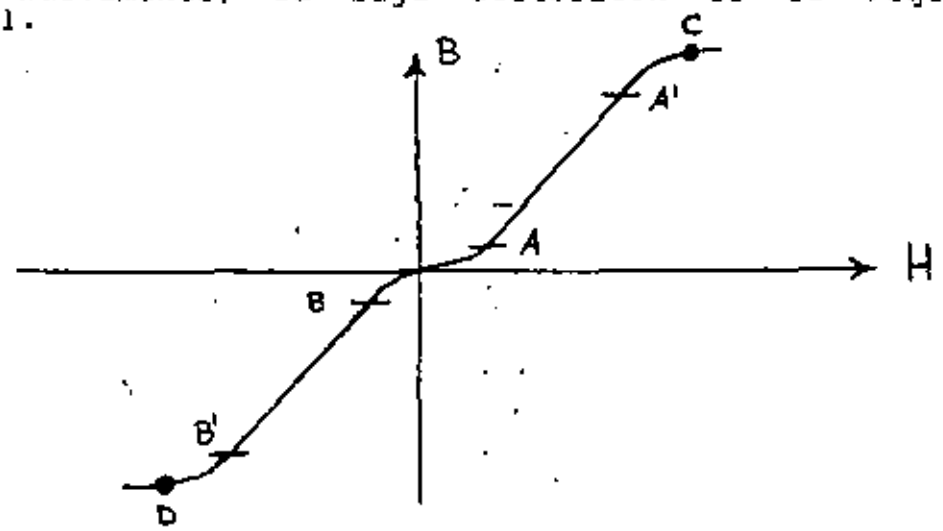


Figura 4-52: Zonas de grabación digital y analógica

Se deben usar solo las porciones líneales de la curva Bh para obtener la menor distorción posible. Para evitar la nolinealidad del cruce por cero la señal se graba sobre una portadora a alta frecuencia para asegurar que la señal (contenida en la envolvente, grabación en amplitud modulada) estarán dentro de las zonas A-A´y b-b´.

4.4.3.2. CASSETTE DIGITAL

El cassette digital no necesita preborrado, se interesa solo en dos estados 0 y 1 por lo que se usan solamente las regiones de saturación, o sea los puntos C y D, ver fig 4-52. Algunas de las ventajas de operar en modo digital son las siguientes:

- En lectura, la señal que se obtiene de la cabeza es proporcional a los cambios de flujo.

- Se obtiene el máximo margen de señal a ruido, debido que la magnetización se mueve de un extremo de saturación al otro.

- No hay niveles críticos de portadora ya que no existe.

- Las densidades que se pueden lograr son muy superiores que en la grabación analógica.

4.4.4. DISCOS FLEXIBLES

La proliferación de discos flexibles en el mercado es enorme, esto hace que se puedan clasificar por diferentes características:

- Por el tamaño existen tres tipos de discos flexibles, los de 8 pulgadas (floppy), 5 1/4 pulgadas (minifloppy) y los de 3 pulgadas (microfloppy), en estos últimos la portadora del medio de grabación no es tan flexible como en los dos anteriores casos, por lo que resultan del tipo cartucho.

- Por el número de cabezas magnéticas, existen los de una sola cara, en los que se graba por un solo lado y los de dos caras que tienen dos cabezas y se puede grabar por ambos lados.

- Por densidad, existen los de densidad sencilla y los de doble densidad, sin embargo, esta característica, generalmente no es una limitante del disco, ya que el disco puede grabar en cualquiera de las dos densidades, la diferencia entre una y otra radica en el código de grabación ya que densidad sencilla usa FM, mientras que doble densidad usa MFM, el que determina en que tipo de densidad (tipo de código) se va ha grabar es el controlador del disco. De hecho existen sistemas que pueden grabar en cualquiera de las dos densidades y simplemente por un comando del sistema operativo se puede escoger cual usar, esto es posible gracias a que el controlador del disco tiene la opción de grabar en uno u otro código.

- Por la identificación del sector, los discos pueden ser "hard-sector" o "soft-sector". Los discos hard-sector identifican a los sectores por medio de marcas físicas (ejemplo, huecos en el disco), por lo que siempre usan un número fijo de sectores por pista. Esta estructura es rígida por lo que no se puede cambiar. En cambio los discos del tipo "soft-sector" identifican a los

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 longitud 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	Longitud 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

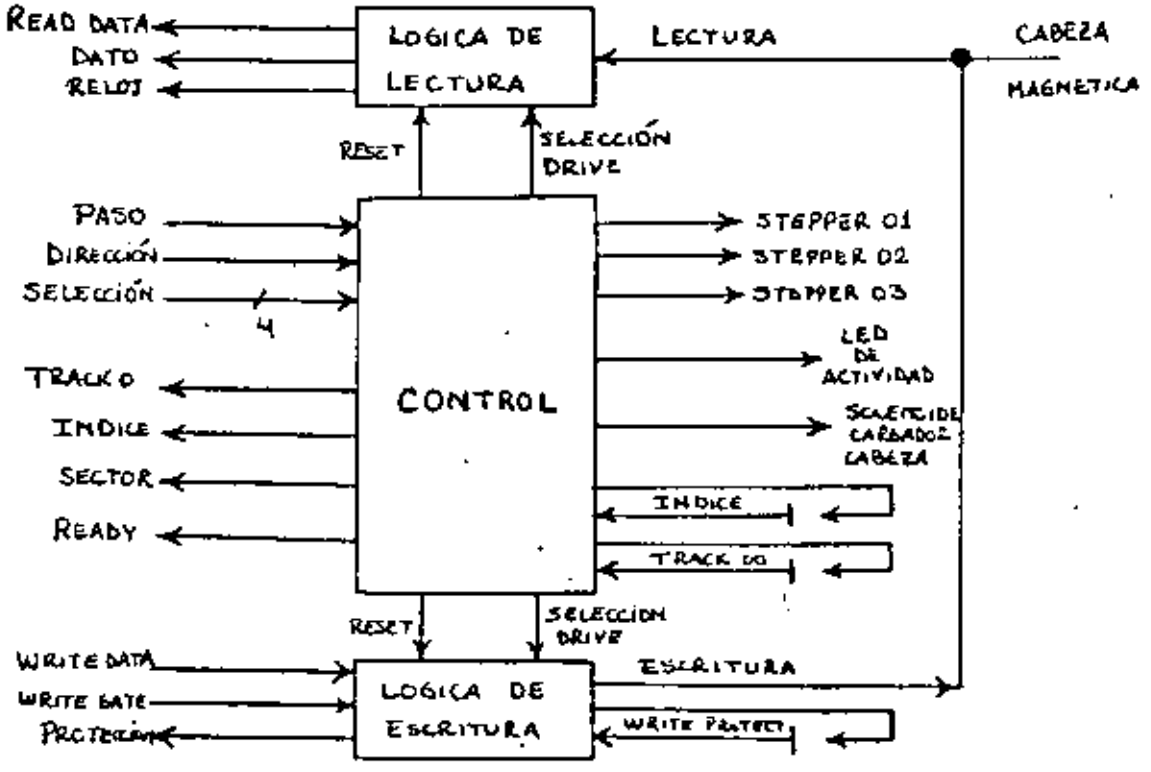


Figura 4-53: Diagrama de bloques del disco flexible

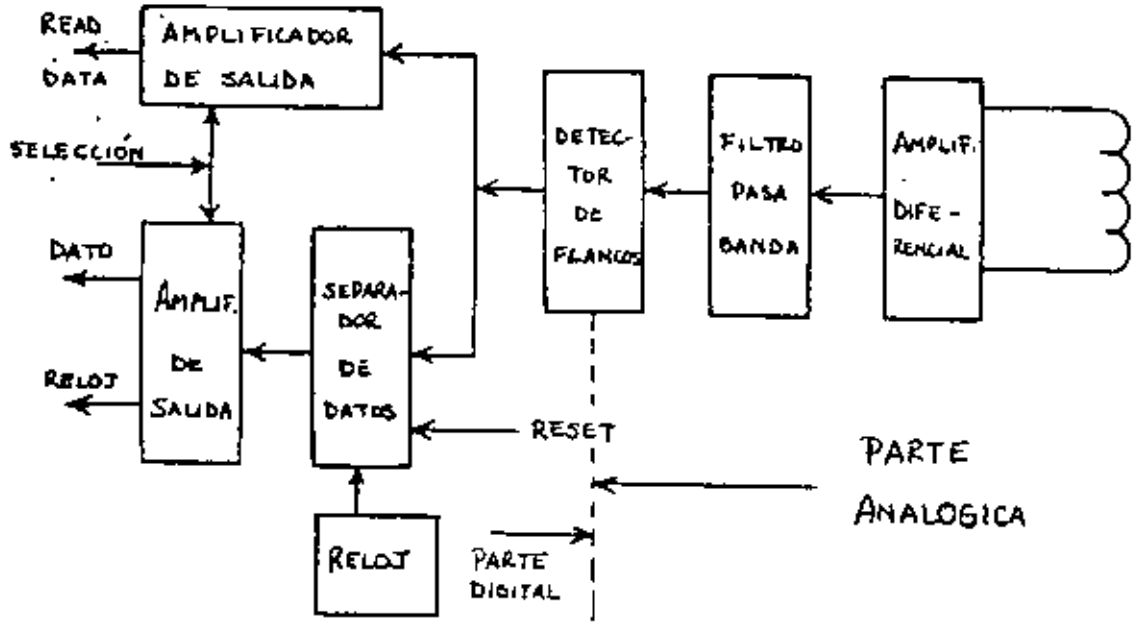


Figura 4-54: Lógica de lectura

está conectada directamente con la cabeza magnética y la parte

digital es la forma en como entrega los datos al controlador. En algunos casos la unidad de disco tiene incluida la parte de separación de datos, aunque esta, generalmente, es función del controlador.

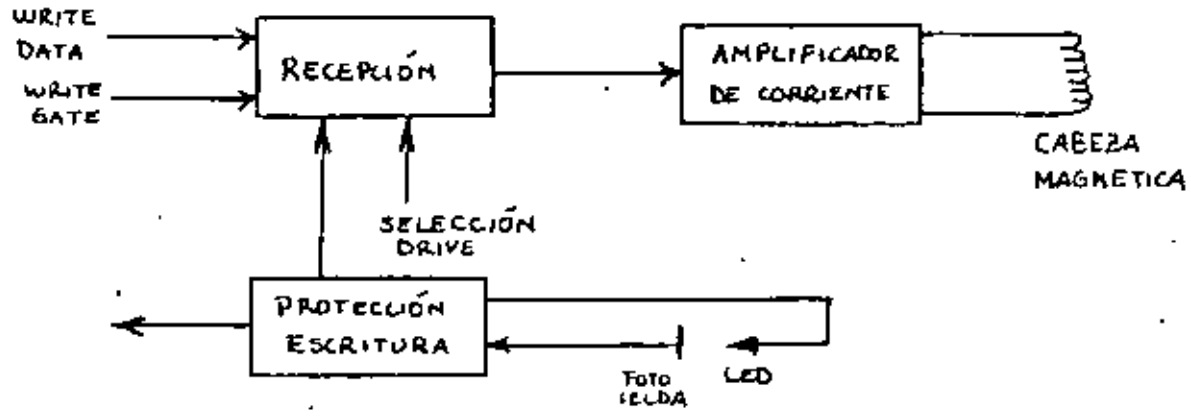


Figura 4-55: Lógica de escritura

La circuitería de escritura es menos compleja porque simplemente recibe los datos a escribir ya codificados y los amplifica en corriente para alimentarlos a la cabeza. Esta circuitería también controla la protección de escritura ya que los discos se pueden proteger físicamente para que no se escriba sobre ellos.

4.4.4.4. EJEMPLOS DE DISCOS FLEXIBLES

Existen muchos fabricantes que producen unidades de discos flexibles tanto de 8 como de 5 1/4 pulgadas. Los minifloppies están ganando popularidad sobre los floppies, ya que las capacidades de ambos se están igualando a un precio menor. Los minifloppies típicos de la actualidad son los de media altura, dos ejemplos típicos de estos son el SA455 y el SA465 de Shugart. El costo unitario de estos discos está en el orden de 175 y 210 dolares respectivamente.

PERFORMANCE SPECIFICATIONS	SA455 SINGLE/DOUBLE DENSITY	SA465 SINGLE/DOUBLE DENSITY
CAPACITY		
Unformatted		
Per Disk	250/500 kbytes	0.5/1 Mbyte
Per Surface	125/250 kbytes	250/500 kbytes
Per Track	3.1/6.2 kbytes	3.1/6.2 kbytes
Formatted		
Per Disk	204.8/409.6 kbytes	409.6/819.2 kbytes
Per Surface	102.4/204.8 kbytes	204.8/409.6 kbytes
Per Track	2.56/5.12 kbytes	2.56/5.12 kbytes
Sectors/track	10	10
TRANSFER RATE	125/250 kbits/sec	125/250 kbits/sec
LATENCY (avg)	100 msec	100 msec
ACCESS TIME		
Track to Track	6 msec	3 msec
Average (incl setting)	93 msec	94 msec
Setting Time	15 msec	15 msec
Motor Start Time	500 msec	500 msec
FUNCTIONAL SPECIFICATIONS		
ROTATIONAL SPEED	300 rpm	300 rpm
RECORDING DENSITY (inside track)	2938/5876 bpi	2961/5922 bpi
FLUX DENSITY	5876 fci	5922 fci
TRACK DENSITY	48 tpi	96 tpi
TRACKS (per surface)	40	80
INDEX	1	1
ENCODING METHOD	FM/MFM	FM/MFM
MEDIA REQUIREMENTS		
Soft sectored	SA154	SA164
16 Sector hard sectored	SA155	SA165
10 Sector hard sectored	SA157	SA167

Tabla 4-6: Especificaciones de los minifloppies SA455 y SA465

4.4.5. DISCOS MAGNETICOS DUROS

Existen dos tipos de discos magnéticos duros, los removibles y los fijos. Las microcomputadoras generalmente usan los discos fijos, en cambio, los discos removibles han sido usados, tradicionalmente, por las máquinas grandes.

4.4.5.1. DISCOS REMOVIBLES

Estos discos han sido privilegio de las computadoras grandes debido, sobre todo, al costo. Existen discos removibles de un solo plato o de un conjuntos de varios platos unidos por un solo eje, llamados paquetes de discos. La ventaja de estos discos es que usando una sola unidad se puede tener gran cantidad de información almacenada en varios discos, los cuales se pueden intercambiar a voluntad. El problema grave de estos discos es la alineación ya que las cabezas deben estar alineadas dentro de

ciertas tolerancias para que cualquier disco pueda ser usado. Estas tolerancias reducen la capacidad de almacenamiento, por lo que son de menor densidad que los discos fijos. Estos discos generalmente son de 14 pulgadas de diametro.

4.4.5.2. DISCOS DE TECNOLOGIA WINCHESTER

La tecnología de los discos fijos ha sido denominada tecnología winchester, sus principales características son:

- Los discos y las cabezas de grabación están encapsuladas herméticamente, lo cual impide la contaminación con basura o cuerpos extraños en los discos.

- En estos encapsulados existe circulación de aire a través de filtros especiales.

- Las densidades que se logran son del orden de 1,000 tracks por pulgada y 10,000 bits por pulgada.

- Las cabezas de grabación son totalmente distintas de las cabezas de los discos removibles. La cabeza está formada por tres rieles separadas entre si por canales, que facilitan la circulación del aire, la bobina de la cabeza está colocada en uno de los extremos del riel del centro. Las cabezas más modernas usan bobinas de pelicula delgada en lugar de los embobinados tradicionales.

- La cabeza descanza siempre sobre la superficie, es decir, al girar el disco, la cabeza flota sobre un colchon de aire del orden de 20 micropulgadas, en cambio, al detenerse el disco, la cabeza se acienta sobre el disco y permanece pegada al disco mientras este está quieto. Algunos discos tienen una pista especial de aterrizaje en donde descanza la cabeza mientras el disco está detenido.

- Las superficies del disco son lubricadas para facilitar el aterrizaje y despegue de las cabezas y asimismo no dañar ni la cabeza ni la superficie del disco. El hecho de que las superficies sean lubricadas permiten ahora que la cabeza tenga contacto con la superficie y puedan aterrizar en cualquier pista sin alterar los datos, sin embargo, es recomendable que los discos tengan una pista especial de aterrizaje para evitar cualquie daño.

- El peso o la fuerza que la cabeza ejerce sobre la

superficie es de 10 gramos. Esta es una de las más importantes diferencias con los discos de cabeza removible porque el poco peso y poca masa de la cabeza reduce al mínimo la posibilidad de dañar tanto la cabeza como la superficie cuando hay un aterrizaje o despegue de la cabeza. En algunos discos removibles la cabeza ejerce una fuerza sobre la superficie del orden de 350 gramos, lo cual es muy peligroso cuando hacen contacto.

- Estos discos no requieren alineación, se calibran una sola vez al momento de fabricarlos, si se desalinean las cabezas no importa porque los tracks en el disco, también, se desalinean en la misma proporción. Por lo tanto, se reduce al mínimo el mantenimiento preventivo, que es uno de los más engorrosos problemas de los discos removibles.

Dos problemas de los discos fijos son: la higiene del disco es muy importante para mantener su confiabilidad, la circulación del aire filtrado es importante para eliminar cualquier contaminante. Y el respaldo de la información, como estos discos no son removibles, la capacidad disponible en disco es la que tiene la unidad unicamente, si se consume esta capacidad hay que guardar información en discos flexibles o en cinta. Un tipo de cinta que se ha popularizado bastante para el respaldo de información de discos Winchester son las cintas de un cuarto de pulgada, conocidas como cintas de cartucho (streaming tape).

Existen unidades de discos Winchester de 14, 8 y 5 1/4 pulgadas, los más usados en microcomputadoras son los dos últimos. Como en los discos flexibles existen discos Winchester de media altura, tanto para 8 como para 5 1/4 pulgadas, las dimenciones de las unidades de discos Winchester son idénticas que las unidades de discos flexible, por lo que en el mismo espacio se pueden sustituir uno por otro. Las capacidades de estos discos varian desde 5 hasta 150 Mbytes y los costos se han abatido de tal manera que ahora resultan muy baratos.

4.4.5.3. EJEMPLOS DE DISCOS WINCHESTER

Un disco insólito es el Maxtor, el cual tiene 140 Mbytes de capacidad bruta en 8 platos de 5 1/4 pulgadas, respetando las dimenciones estandar. Lo más increible es que su costo es de 2,600 dolares (precio de distribuidor).

Haciendo a un lado estas excepciones, los discos más comunes, actualmente, tienen capacidades entre 10 y 50 Mbytes, para el caso de discos de 5 1/4. Estos minidiscos están ganando

mucha popularidad y desplazando a los discos más grandes. Un ejemplo típico de estos se pueden considerar a los discos Shugart SA706 y SA712, los cuales son discos de 5 1/4 y de media altura. El costo unitario de estos discos está en el orden de 500 y 600 dolares respectivamente. Asi como estos hay muchos fabricantes que tienen discos semejantes en capacidad y costo.

PERFORMANCE SPECIFICATIONS		
CAPACITY	SA706	SA712
Unformatted		
Per Drive	6.67 Mbytes	13.33 Mbytes
Per Surface	3.3 Mbytes	3.3 Mbytes
Per Track	10,416 bytes	10,416 bytes
Formatted		
Per Drive	5.24 Mbytes	10.48 Mbytes
Per Surface	2.62 Mbytes	2.62 Mbytes
Per Track	8,192 bytes	8,192 bytes
Per Sector	256 bytes	256 bytes
Sectors/Track	32	32
TRANSFER RATE	5.0 Mbits/sec	5.0 Mbits/sec
LATENCY (avg)	8.40 msec	8.40 msec
ACCESS TIME (includes settling)		
Track to Track	16.2 msec	16.2 msec
Average	99 msec	99 msec
Maximum	199 msec	199 msec
FUNCTIONAL SPECIFICATIONS		
ROTATIONAL SPEED	3600 rpm	3600 rpm
RECORDING DENSITY	9036 bpi	9036 bpi
FLUX DENSITY	9036 fci	9036 fci
TRACK DENSITY	360 tpi	360 tpi
CYLINDERS	320	320
TRACKS	640	1280
R/W HEADS	2	4
DISKS	1	2
INDEX	1	1

Tabla 4-7: Especificaciones de los discos SA706 y SA712

4.4.6. CONTROLADORES DE DISCOS MAGNETICOS

Los controladores de discos son los que le dan la inteligencia al disco para que se puedan usar, sin el controlador no es posible que el procesador pueda usarlos adecuadamente. Existen dos tipos de discos, los no inteligentes que son los que usan todas las microcomputadoras y son como los descritos anteriormente que necesitan un controlador para ser usados. En cambio los discos inteligentes tienen el controlador integrado en el disco y pueden operar en forma autónoma descargando gran parte de las funciones del procesador, estos discos los usan, generalmente, las super computadoras.

Existen dos enfoques para que el procesador se conecte con el disco magnético.

1. Usar controladores universales, los cuales se pueden conectar a una gran cantidad de buses de procesador, requiriendo para cada caso simplemente un pequeño adaptador particular para el bus de que se trate.

2. Usar controladores particulares, los cuales se conectan directamente al bus del procesador. Tienen la desventaja que son muy difíciles de conectar a otro bus.

Las funciones de un controlador de discos flexibles son las mismas funciones básicas de los controladores de discos Winchester. O sea que un controlador de discos flexibles es un subconjunto del controlador de discos Winchester. Los controladores de discos flexibles los constituyen uno o dos chips, en cambio los controladores de discos Winchester son tarjetas.

4.4.6.1. DIAGRAMA DE BLOQUES

Existen algunos componentes básicos en todo controlador de discos magnéticos, como son la lógica de lectura que implica decodificar, y convertir los datos de serie a paralelo, la lógica de escritura que efectua, exactamente, las funciones contrarias a la lógica de lectura, la interfase al procesador o hacia un bus generalizado, la interfase a las unidades de disco para manejar las señales de control y finalmente, el manejador central del procesador que se encarga de manejar las secuencias de operación y controlar la sincronización de todos los componentes internos.

La figura 4-56 muestra a la izquierda la interfase de conección con el bus del procesador y a la derecha la interfase de conección con las unidades de disco.

Las figuras 4-57 y 4-58 muestran los componentes básicos que todo controlador de discos debe tener, tanto en la etapa de escritura al disco como en la lectura de los datos.

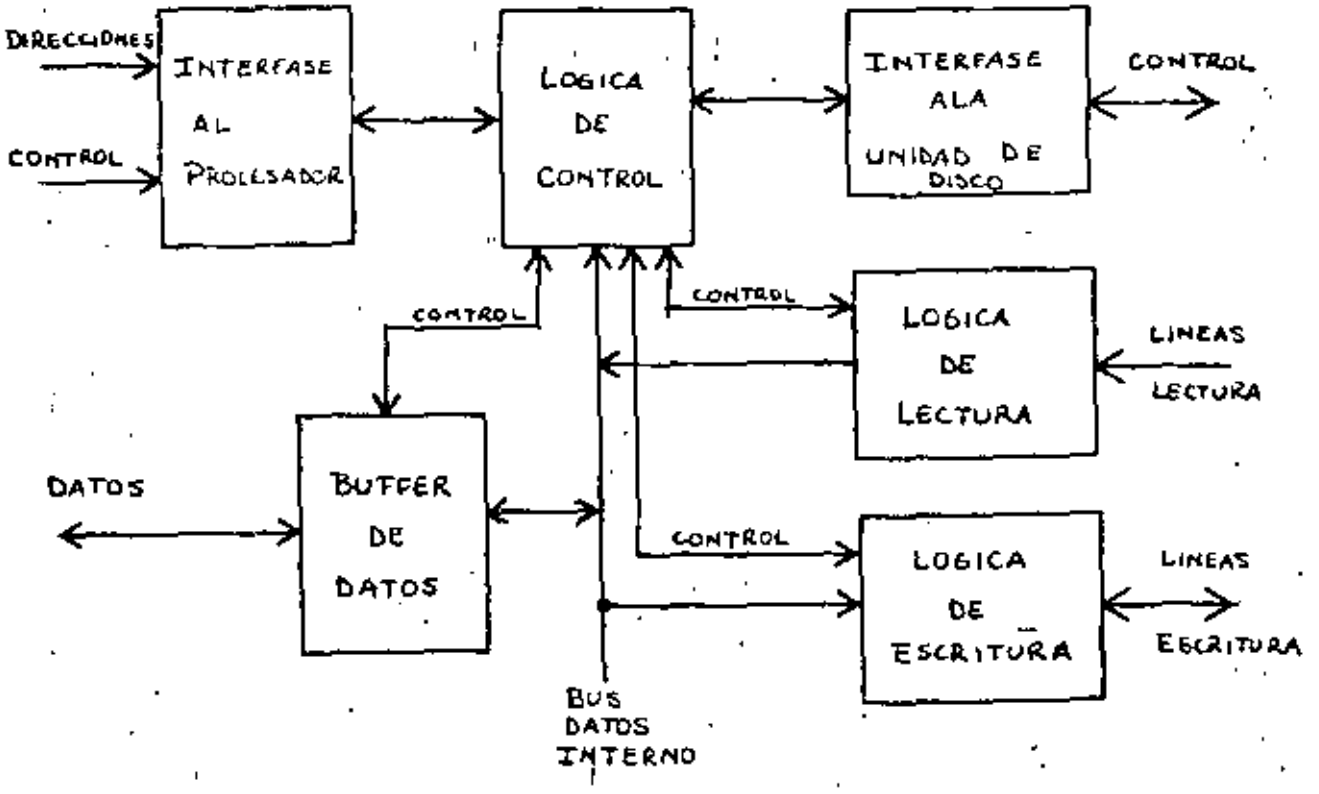


Figura 4-56: Bloques básicos del controlador de discos

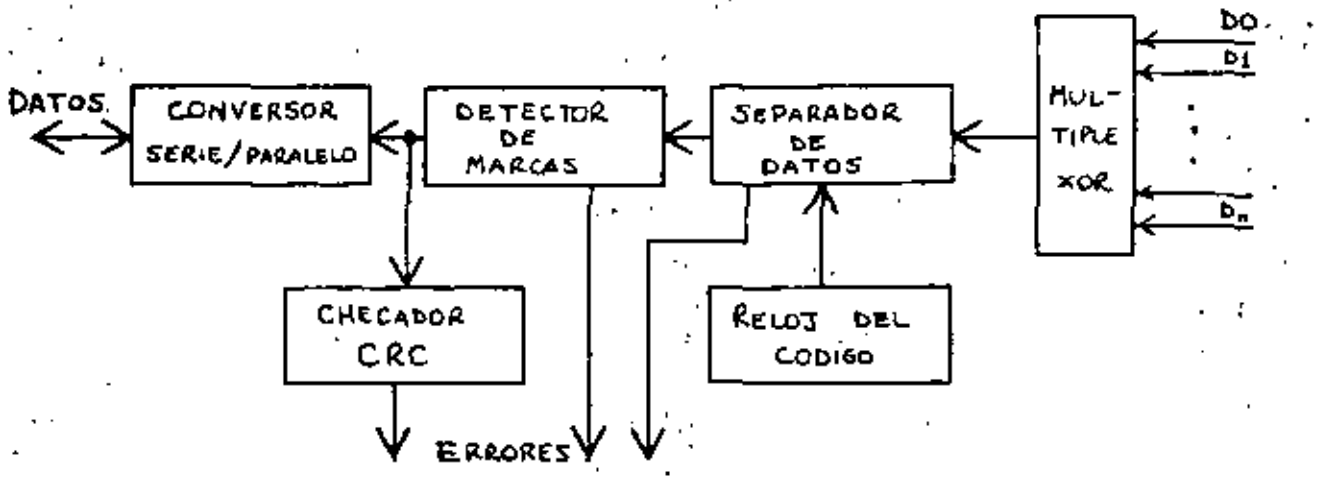


Figura 4-57: Lógica de lectura

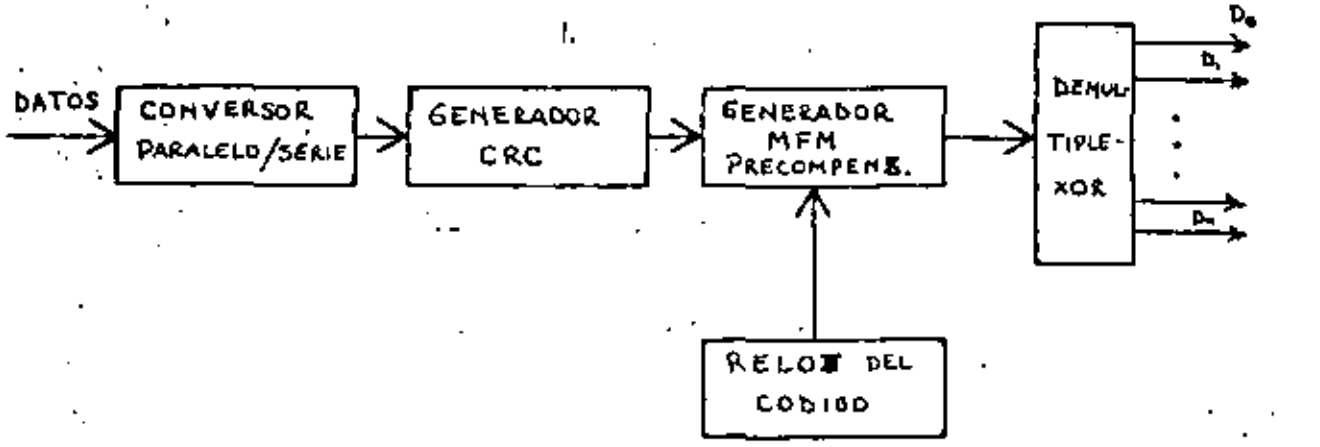


Figura 4-58: Lógica de escritura

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 localizacón de un sector de un track.
- Tamaño del sector programable.
- Generación y cnequeo 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

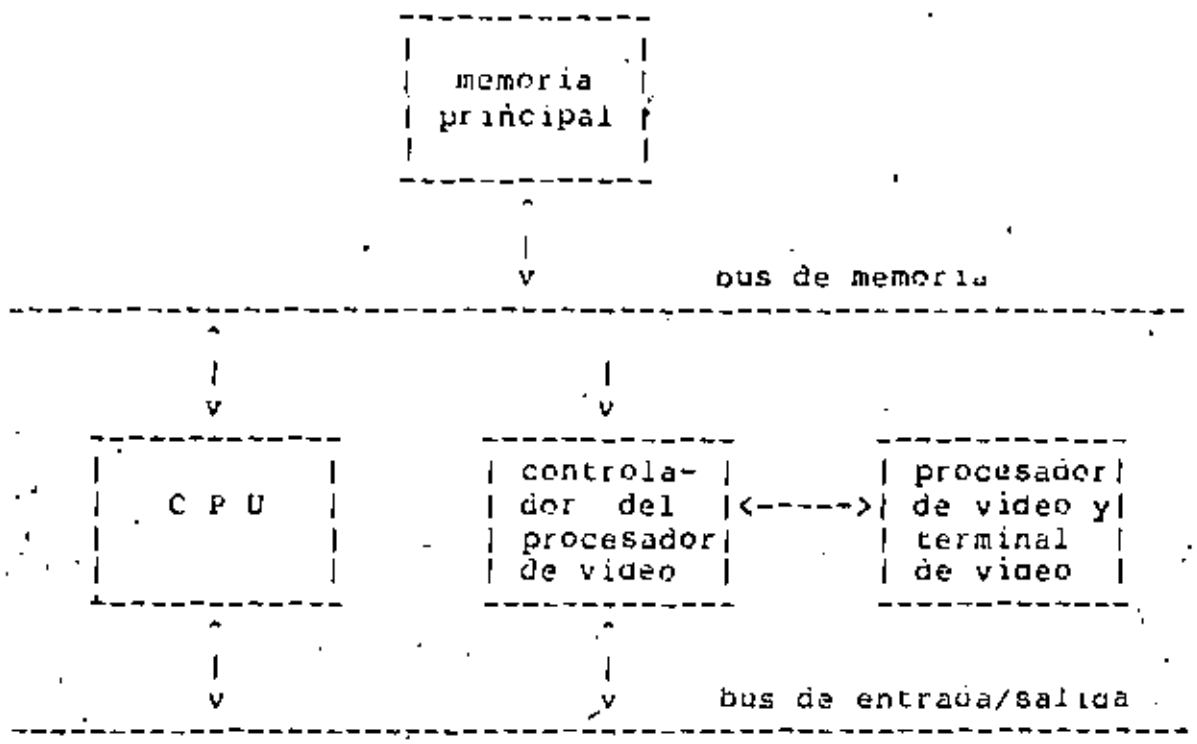


Figura 4-59: Terminal de video en una computadora.

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 computadoras 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 Wincnester; 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 pais), 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, wincnesters, 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 afrontarse, ya que es mayor el tiempo y esfuerzo que el programador ahorra 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 nacer 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 específicos.

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 debé 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 la etiqueta	código de operación o mnemónico	operando o campo de la dirección	campo de comentarios
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 ensamolador 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 suorutinas. 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 l 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 retencia 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 nacer 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 hará 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 había sido definida aún, lo que hacía 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 regresar a esos lugares que habían sido dejados en blanco y poner así 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 hace 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 cargador 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. Chaca 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 ha convertirse en tema de investigación, conforme se empiezan ha 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 quehacer 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. Intertaz 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

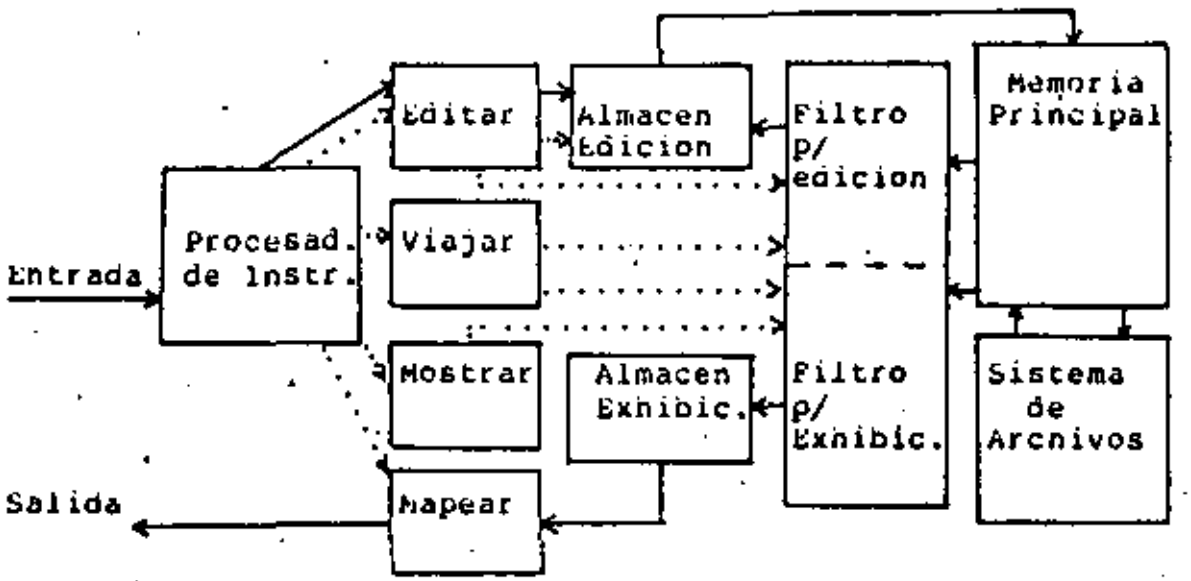


Figura 6-1: Arquitectura general de un editor

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 categorias 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 antitrión.

   Estas funciones pueden ser obtenidas en parte de un pequño 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 inoependiente 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 hacía correcciones 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 corrían 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, había 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:

- Permitían crear y modificar archivos desde una terminal.
- Las líneas de estos archivos eran de longitud fija, inicialmente de 80 caracteres.
- Muchos de estos editores permitían hacer correcciones, usando mucho 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 parratos y bloques de programas, usualmente con la imagen de el formato de las tarjetas; no habia 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 limitada 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 benético 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 :

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 ortografía 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.** Facilidades 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 limitaciones 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 : durante la primera se traduce el código fuente a un lenguaje intermedio ó forma interna; durante la segunda se ejecutan 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 GOTOś, lo que impide hacer programas verdaderamente legibles. Sus ventajas incluyen: simplicidad de instrucciónes 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 número.

- 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 requiére un dato a entrar por el teclado.

- DATA

Esta proposición creá 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 interior 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 dígitos 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 ahorro 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 aplicaciones, 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 parámetros 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 posfija 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 cúal 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

Cómo 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 algúna 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 palaora 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 diccionario 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 hacia las direcciones bajas.

- PILA DE RETORNO

Para no interferir con los parámetros, el contador de programa se almacena en otra pila que también crece hacia 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 poderoso. 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 h 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
```

- CSET

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.

```
1234n
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.

```
1111h 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 poder 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 nativo, 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.

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 cuál 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 pásale 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

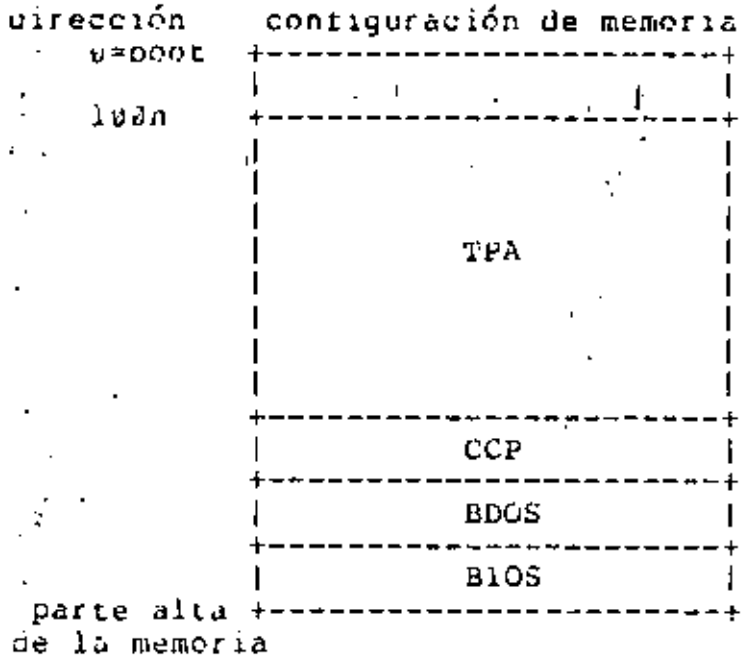
Antes de comenzar este tema deseo hacer notar que un S.O. para una microcomputadora que maneja un sólo usuario y no maneja multiprogramación, es bastante más sencillo en cuanto al control, asignación y recuperación de los recursos de la micro que el S.O. de una supercomputadora que maneja múltiples usuarios 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 8088, 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 leído, 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 perroradora. 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 seleccionar 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. LISTST 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 lógica 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 pude 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 está 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 buffer 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 leído 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 butter 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 butter, 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 0-1 si encontro algúno o FF si no, y lo que hace es buscar el primer entry del directorio que case 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 contenorá un código de error de 0 a 6. La información 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 dicienoo 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 tenorá 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 están 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 había 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 dentro 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 que 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 bB.

   1. 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 280 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, User's Manual", MOTOROLA INC, (enero 1980).

3.18 - Zilog, "Z8000 Family, Technical 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., Menachem, 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, Technical Report No. 47, 17 pag. (dic. 1979).

3.28 Fairclough, D.A., "A Unique Microprocessor Instruction Set", IEEE MICRO, vol. 2, No. 2, pp. 8-16 (mayo 1982).

3.29 Best, D.W., Kress, C.E., Nykris, 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.M., "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. 125-129 (abril 24, 1980).

CAPITULO 4

LIBROS

4.1 Triebel, Walter A., y Chu, Altred 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

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, Goblingen, 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 (1976), 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

4.19 Knoll, A.L., "Spectrum Analysis of Digital Magnetic Recording Waveforms", IEEE TRANS. ON ELECTRON. COMPUT., vol EC-16, No.6, pp. 732-743 (diciembre 1967).

4.20 Franchini, R.C. y Hartner, 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 Sidhu, P.S., "Group-Coded Recording Reliably Doubles Diskette Capacity", COMPUTER DESIGN, pp. 84-88 (diciembre 1976).

CINTA MAGNETICA

4.23 Sallet, D.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 Houghton, R.E., "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 1978).

4.29 Hoages, D.A., "A Review and Projection of Semiconductor Components for Digital Storage", PROCEEDINGS OF THE

IEEE, vol. 63, No. 6, pp. 1136-1147 (agosto 1975).

4.30 Bursky, D., "Special Report: Memories pace systems growth", ELECTRONIC DESIGN, pp. 65-76 (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 K.G. Loeliger, Threaded Interpretive Languages, Byte Books, Peterborough Nn, 1981.

CAPITULO 8

8.1 Lister, A.M.A "Fundamentals of Opertaing Systems" Springer-Verlag. 2nd edition.

8.2 Haunick, 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 Gagle, Michael. Konemer, Gary J. Winston, 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

SEPTIEMBRE, 1983

Palacio de Minería Calle de Tacuba 5 primer piso Deleg. Cuauhtemoc 06000 México, D.F. Tel.: 521-40-20 Apdo. Postal M-2285

Desarrollo/Tecnología

Presentado en IEEE COMPUTACION '87
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.

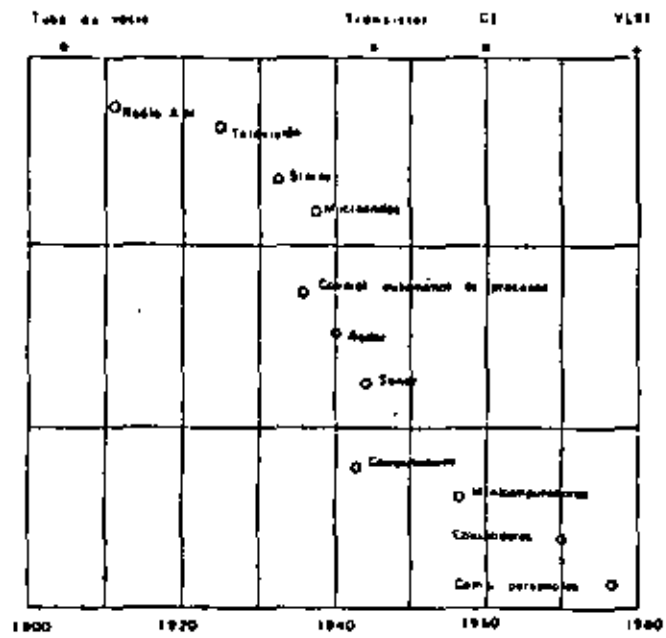
Según algunos economistas (sobre todo Mandel) después de la revolución industrial el mundo moderno ha atravesado por otras dos "revoluciones tecnológicas": aquella inaugurada por el cambio de la fuerza motriz de los motores de vapor a los motores de combustión interna y aquella que se verifica con el descubrimiento de la fusión nuclear controlada. Esta última iría acompañada por el gran desarrollo de la electrónica que hallaría aplicaciones en todo los ámbitos 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 revolucion 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 rapida 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 producían 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

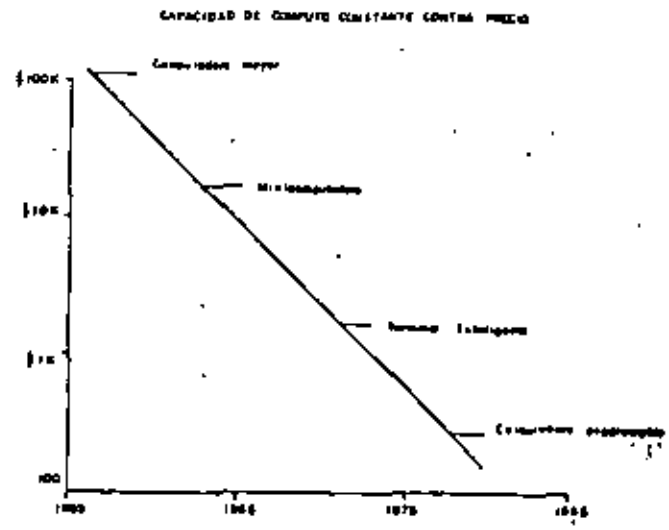


Fig. 2

Raúl Rojas González
Universidad Nacional Autónoma de México
Instituto Nacional de Investigaciones Nucleares

X 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 caída 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.

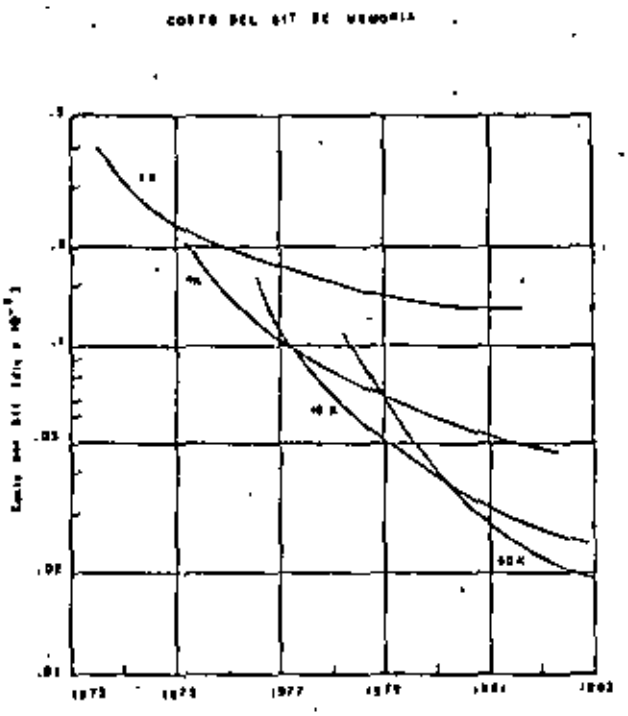


Fig. 13.

(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.

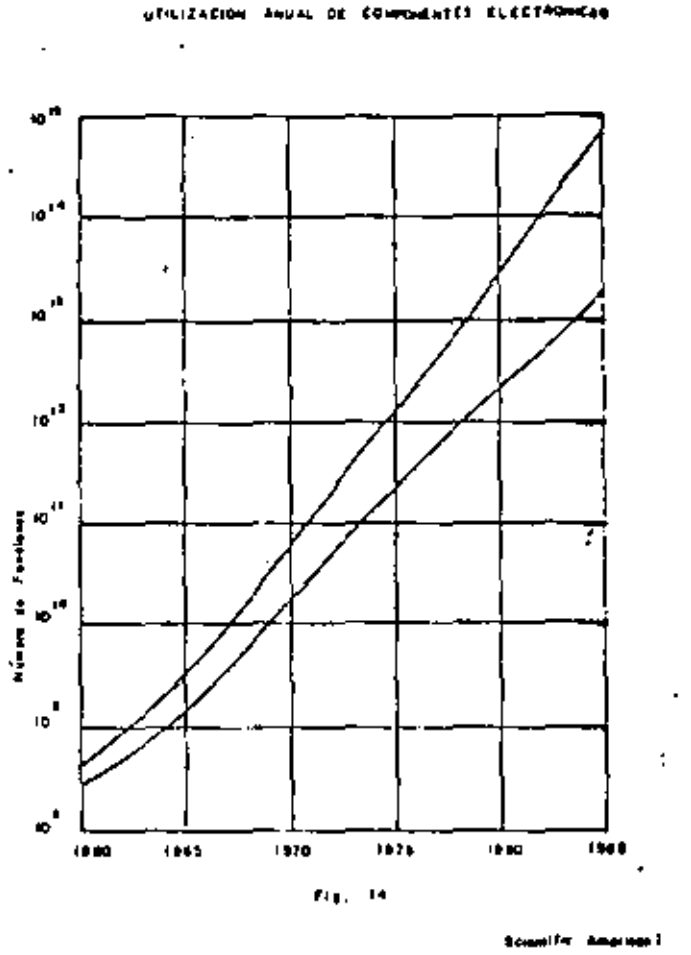


Fig. 14

(Scientific American)

III. Dos obstáculos a la revolución de las computadoras

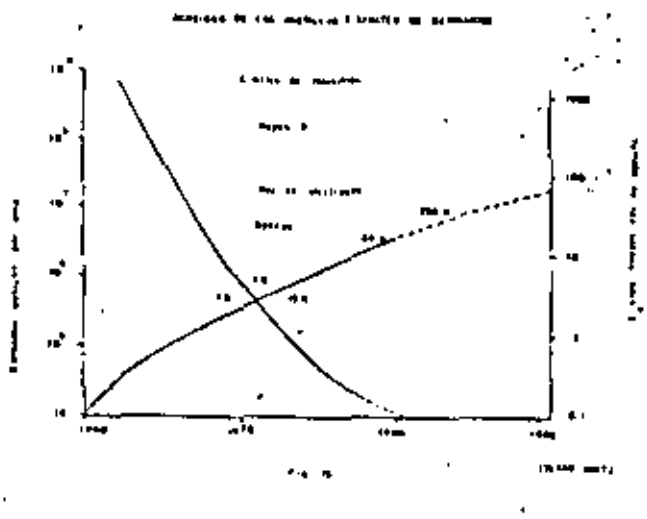
Frente a esta situación de crecimiento exponencial de la complejidad de los circuitos y de extensión de la utilización de las computadoras hay dos obstáculos principales. El primero es un obstáculo "débil": la tecnología de impresión de los circuitos integrados. El segundo es un obstáculo "fuerte": el software 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 electrones o de rayos X.

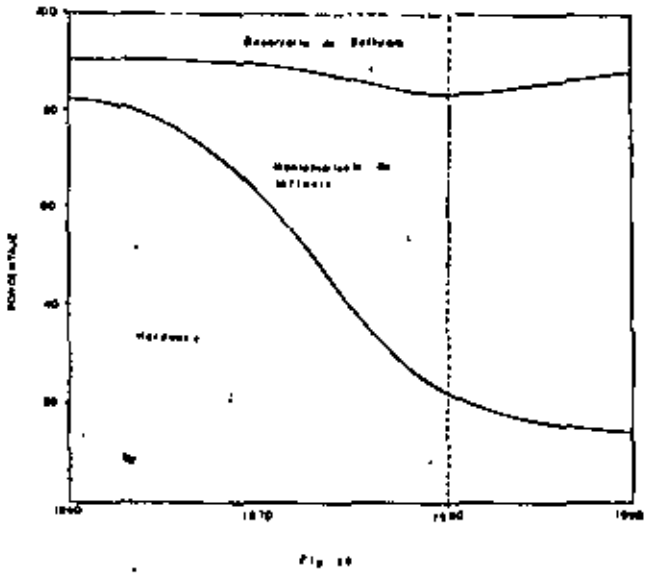
En la figura 15 se puede ver esta cuestión. El problema sin embargo, no es insuperable ya que las nuevas técnicas se están ajustando y es posible que en los próximos años comience su utilización en gran escala.

El segundo obstáculo es lo que se ha llamado el "cuello de botella" de la computación: el software. Resulta que mientras 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 tenderá a agudizarse en el futuro (23). De los costos de software, además, la mayor parte está constituida por los gastos de

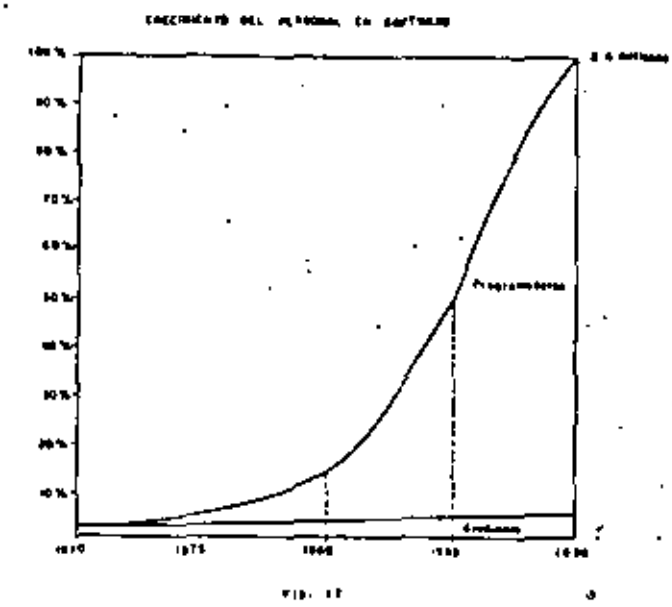


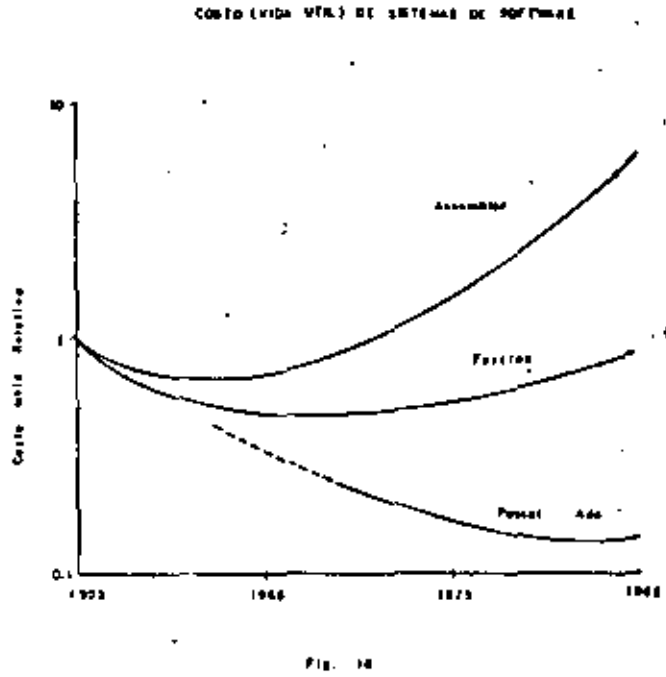
mantenimiento, es decir, por la constante revisión y actualización de los sistemas.

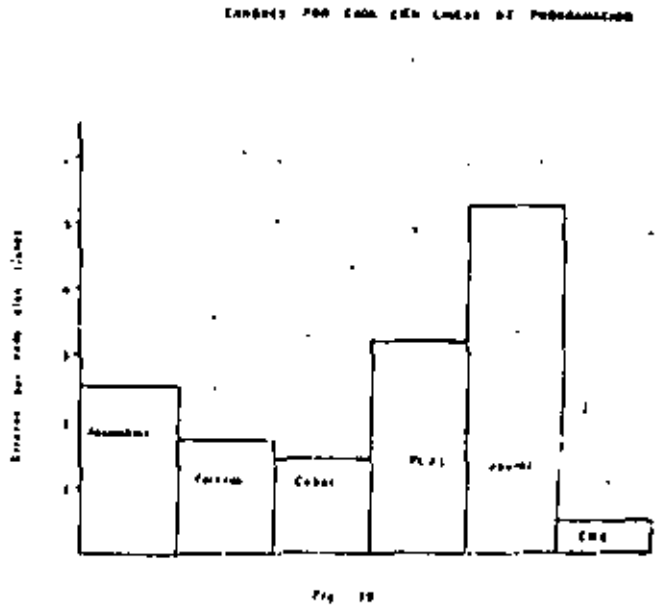


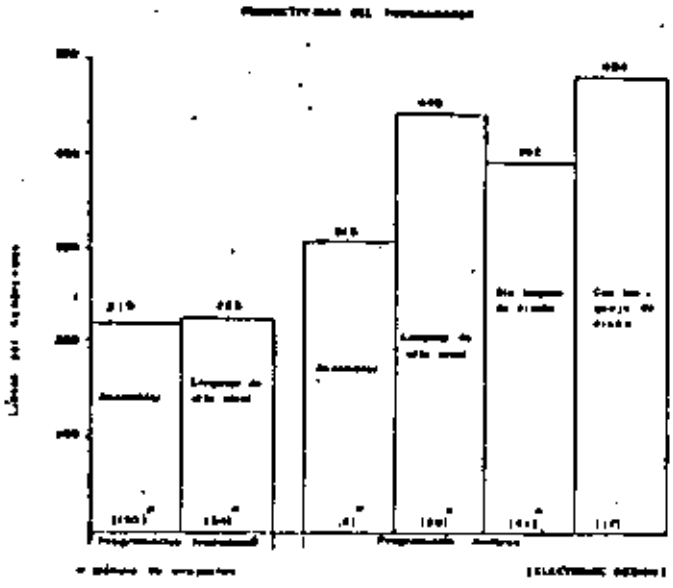
El problema del software tiene un doble origen. En primer lugar los lenguajes de programación no han avanzado hasta el punto en que se cuente con un lenguaje módular, de alto nivel y que minimice los errores de programación. Pascal es un avance en ese sentido, pero aún falta diseñar, crear nuevas técnicas 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.









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 actuales. 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 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).

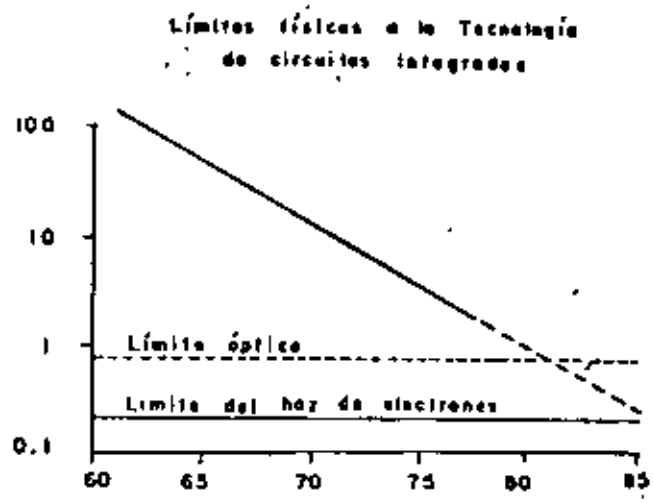


Fig. 21

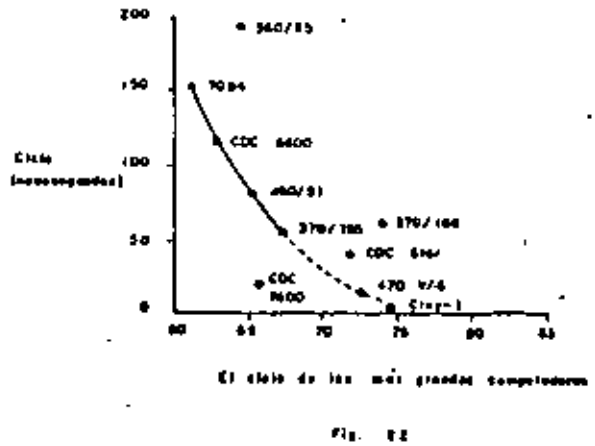


Fig. 22

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 picosegundo. 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

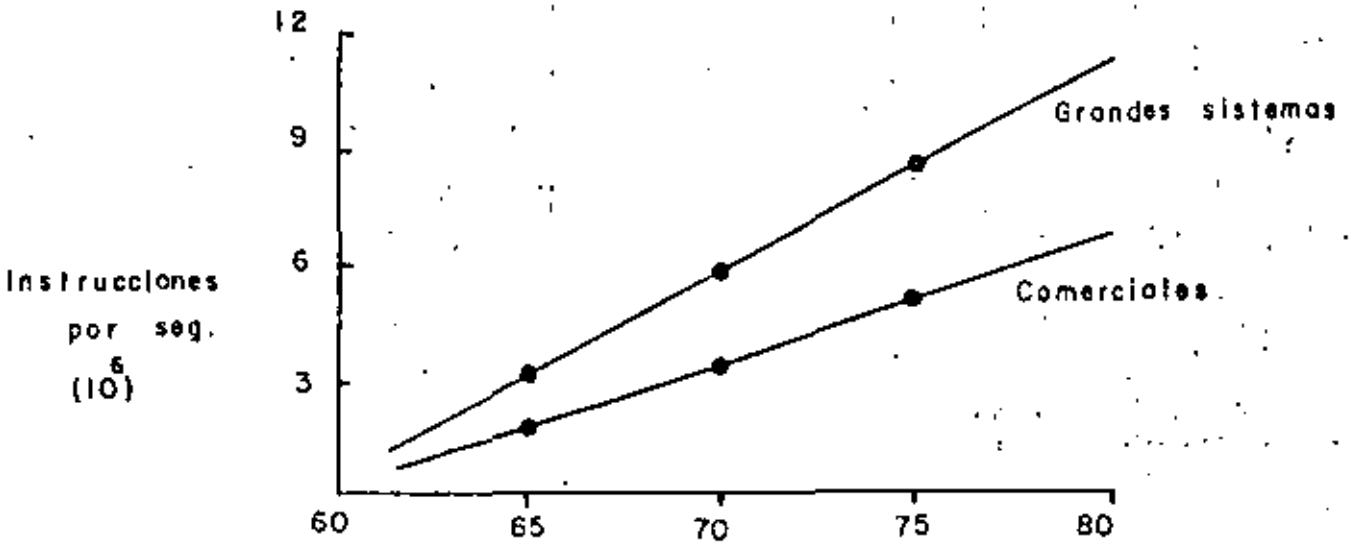


Fig. 23

Velocidad de las computadoras de mono y multiprocesador

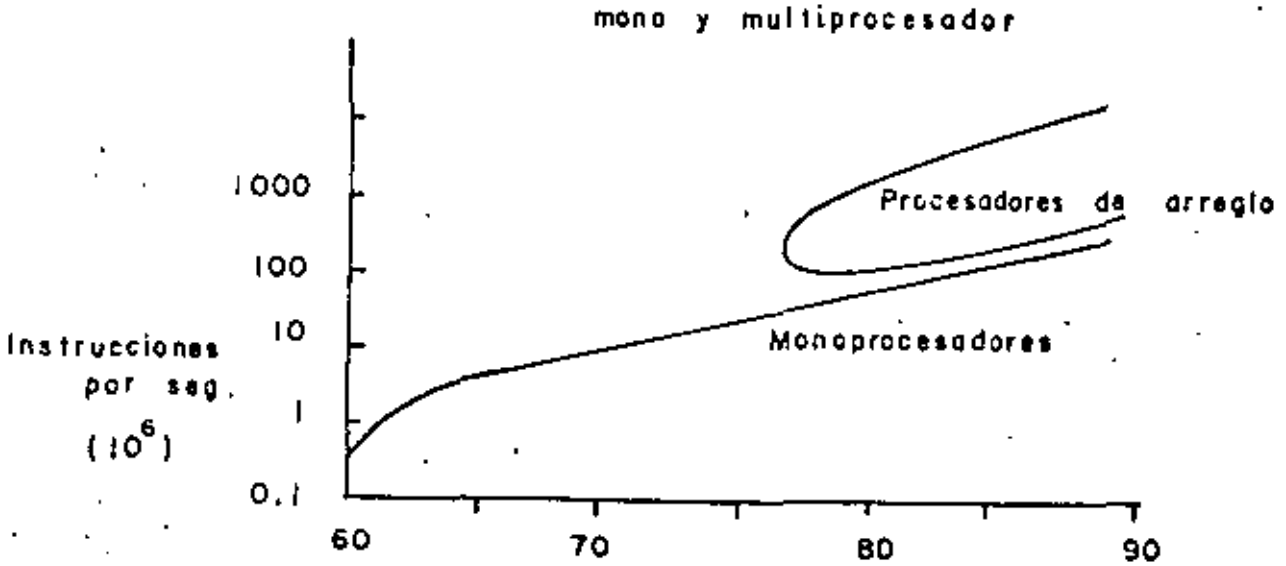


Fig. 24

(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 anterior: 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 work force) 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

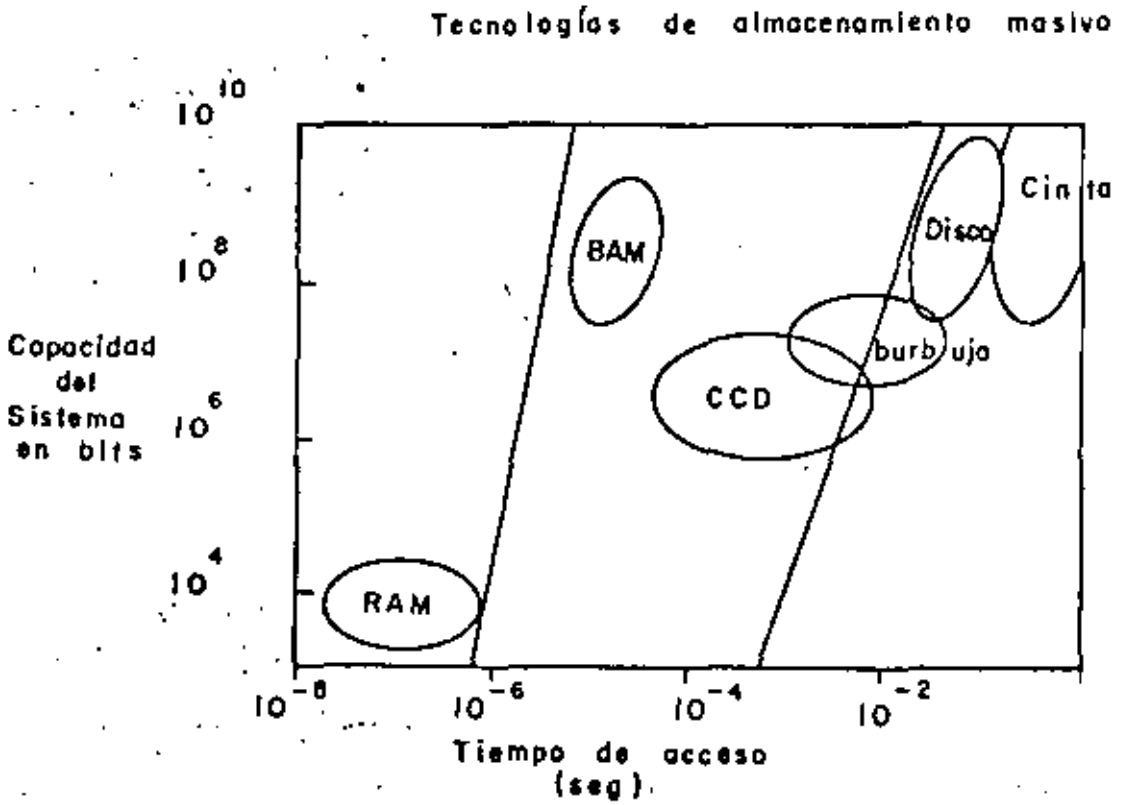


Fig. 25

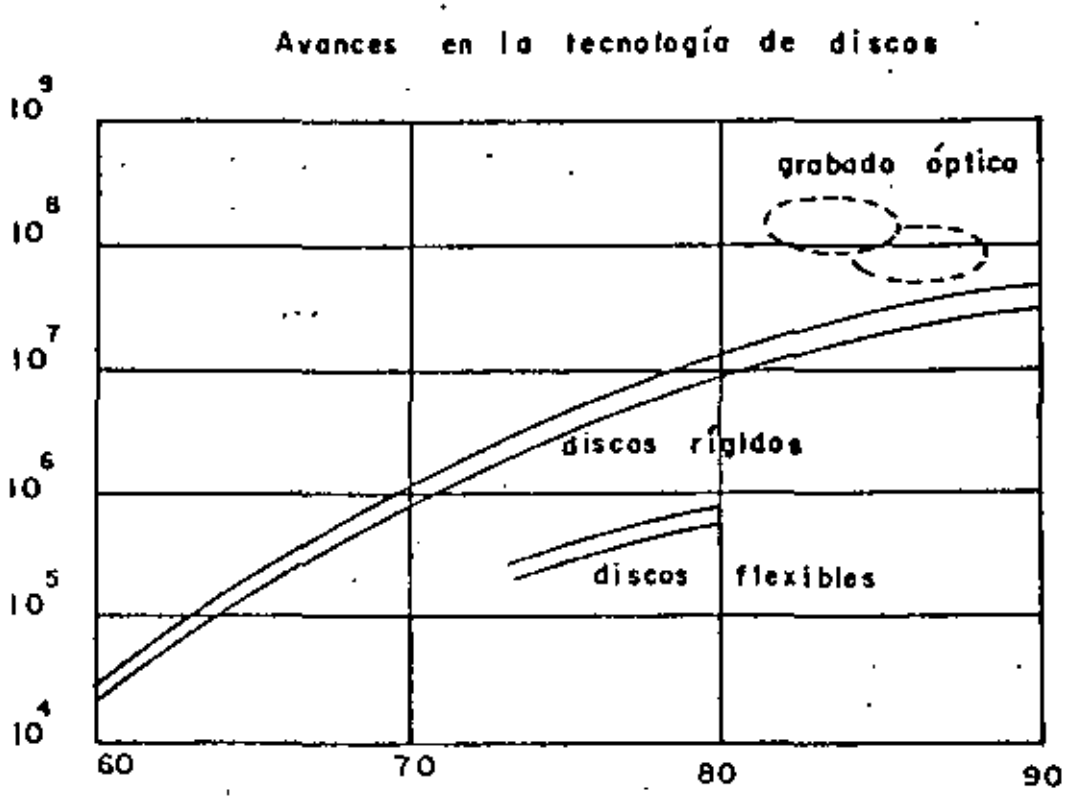


Fig. 26

(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 solo 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. ★

MICROPROCESADORES Y MICROCOMPUTADORAS

Z8400
Z80 CPU Central Processing Unit

SEPTIEMBRE, 1983

Z8400
Z80® CPU Central Processing Unit

(5)

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.

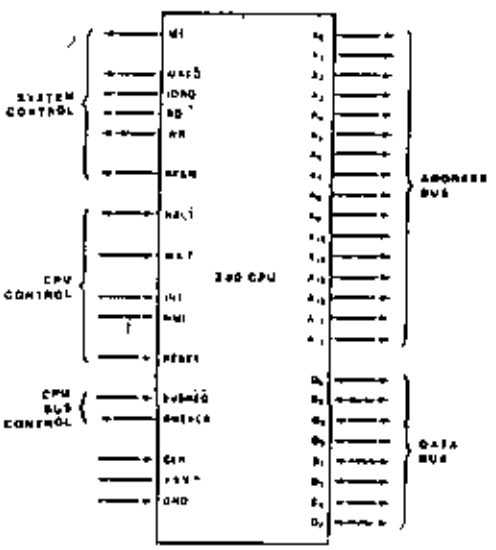


Figure 1. Pin Functions

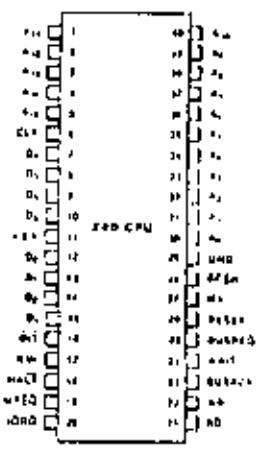


Figure 2. Pin Assignments

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 interface with and requires a system clock and 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.

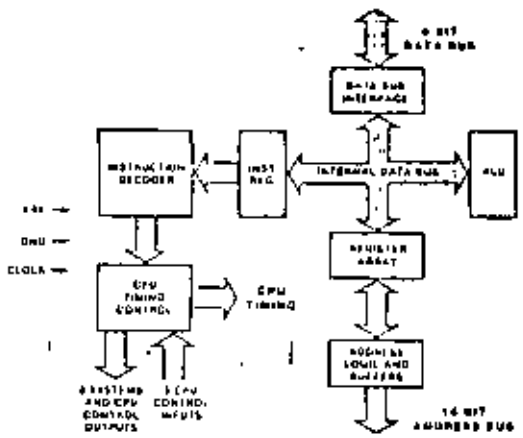


Figure 3. Z80 CPU Block Diagram

Z80 Microprocessor Family

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 I provides further information on these registers.

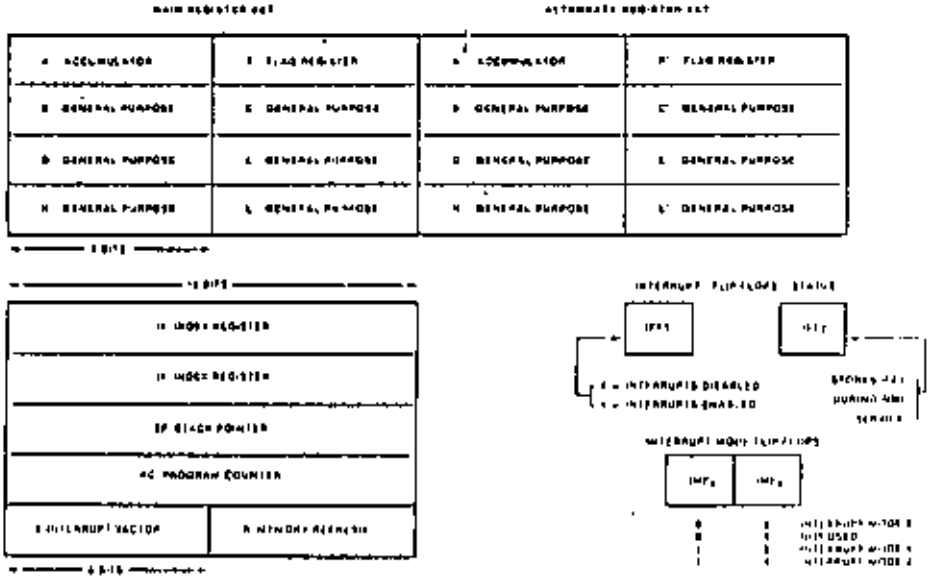


Figure 4. CPU Registers

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).
IM0-IM2	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.

Interrupts: General Operation (Continued)

Non-Maskable Interrupt (NMI). The non maskable interrupt cannot be disabled by program control and therefore will be accepted 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 hardwired to a High level. This 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

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	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 ← n →	2	2	7	000 B 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 01 r 110 ← d →	3	5	19	011 E 100 H 101 L
LD r, (IY + d)	r ← (IY + d)	·	·	X	·	X	·	·	·	11 111 101 01 r 110 ← d →	3	5	19	111 A
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 01 110 r ← d →	3	5	19	
LD (IY + d), r	(IY + d) ← r	·	·	X	·	X	·	·	·	11 111 101 01 110 r ← d →	3	5	19	
LD (HL), n	(HL) ← n	·	·	X	·	X	·	·	·	00 110 110 ← n →	2	3	10	
LD (IX + d), n	(IX + d) ← n	·	·	X	·	X	·	·	·	11 011 101 00 110 110 ← d → ← n →	4	5	15	
LD (IY + d), n	(IY + d) ← n	·	·	X	·	X	·	·	·	11 111 101 00 110 110 ← d → ← n →	4	5	19	
LD A, (BC)	A ← (BC)	·	·	X	·	X	·	·	·	00 001 010	1	2	7	
LD A, (DE)	A ← (DE)	·	·	X	·	X	·	·	·	00 011 010	1	2	7	
LD A, (nn)	A ← (nn)	·	·	X	·	X	·	·	·	00 111 010 ← n → ← n →	3	4	13	
LD (BC), A	(BC) ← A	·	·	X	·	X	·	·	·	00 000 010	1	2	7	
LD (DE), A	(DE) ← A	·	·	X	·	X	·	·	·	00 010 010	1	2	7	
LD (nn), A	(nn) ← A	·	·	X	·	X	·	·	·	00 110 010 ← n → ← n →	3	4	13	
LD A, I	A ← I	↕	↕	X	0	X	IFF	0	·	11 101 101 01 010 111	2	2	9	
LD A, R	A ← R	↕	↕	X	0	X	IFF	0	·	11 101 101 01 011 111	2	2	9	
LD I, A	I ← A	·	·	X	·	X	·	·	·	11 101 101 01 000 111	2	2	9	
LD R, A	R ← A	·	·	X	·	X	·	·	·	11 101 101 01 001 111	2	2	9	

NOTES: r, r' means any of the registers A, B, C, D, E, H, L.
IFF the contents of the interrupt enable flip-flop (IFF) is copied into the P/V flag.
For explanation of the notation used to denote the addressing modes and symbolic operation, see Symbolic Notation section, following in text.

16-Bit Load Group

Mnemonic	Symbolic Operation	S	Z	Flags H	P/V	N	C	Opcode	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 Pair 00 BC 01 DE 10 HL 11 SP
LD IX, nn	IX ← nn	·	·	X	·	X	·	11 011 101 00 100 001 — n — — n —	DD 21	4	4	14	
LD IY, nn	IY ← nn	·	·	X	·	X	·	11 111 101 00 100 001 — n — — n —	FD 21	4	4	14	
LD HL, (nn)	H ← (nn + 1) L ← (nn)	·	·	X	·	X	·	00 101 010 — n — — n —	2A	3	5	16	
LD dd, (nn)	ddH ← (nn + 1) ddL ← (nn)	·	·	X	·	X	·	11 101 101 01 dd1 011 — n — — n —	ED	4	6	20	
LD IX, (nn)	IXH ← (nn + 1) IXL ← (nn)	·	·	X	·	X	·	11 011 101 00 101 010 — n — — n —	DD 2A	4	6	20	
LD IY, (nn)	IYH ← (nn + 1) IYL ← (nn)	·	·	X	·	X	·	11 111 101 00 101 010 — n — — n —	FD 2A	4	6	20	
LD (nn), HL	(nn + 1) ← H (nn) ← L	·	·	X	·	X	·	00 100 010 — n — — n —	22	3	5	16	
LD (nn), dd	(nn + 1) ← ddH (nn) ← ddL	·	·	X	·	X	·	11 101 101 01 dd0 011 — n — — n —	ED	4	6	20	
LD (nn), IX	(nn + 1) ← IXH (nn) ← IXL	·	·	X	·	X	·	11 011 101 00 100 010 — n — — n —	DD 22	4	6	20	
LD (nn), IY	(nn + 1) ← IYH (nn) ← IYL	·	·	X	·	X	·	11 111 101 00 100 010 — n — — n —	FD 22	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 11 111 001	DD F9	2	2	10	
LD SP, IY	SP ← IY	·	·	X	·	X	·	11 111 101 11 111 001	FD F9	2	2	10	
PUSH qq	(SP − 1) ← qqH (SP − 2) ← qqL SP ← SP − 2	·	·	X	·	X	·	11 qq0 101		1	3	11	qq Pair 00 BC 01 DE 10 HL 11 AF
PUSH IX	(SP − 1) ← IXH (SP − 2) ← IXL SP ← SP − 2	·	·	X	·	X	·	11 011 101 11 100 101	DD E5	2	4	15	
PUSH IY	(SP − 1) ← IYH (SP − 2) ← IYL SP ← SP − 2	·	·	X	·	X	·	11 111 101 11 100 101	FD E5	2	4	15	
POP qq	qqH ← (SP + 1) qqL ← (SP) SP ← SP + 2	·	·	X	·	X	·	11 qq0 001		1	3	12	
POP IX	IXH ← (SP + 1) IXL ← (SP) SP ← SP + 2	·	·	X	·	X	·	11 011 101 11 100 001	DD E1	2	4	14	
POP IY	IYH ← (SP + 1) IYL ← (SP) SP ← SP + 2	·	·	X	·	X	·	11 111 101 11 100 001	FD E1	2	4	14	

NOTES: qq is any of the register pairs BC, DE, HL, SP.
qq is one of the register pairs AF, BC, DE, HL.
Subtraction of binary numbers is done by adding the two's complement of the subtrahend.
e.g., $BC_L - C$, $AF_H = A$.

Exchange, Block Transfer, Block Search Groups

Mnemonic	Symbolic Operation	S	Z	H	P/V	N	C	Opcode	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	IXH ↔ (SP + 1) IXL ↔ (SP)	·	·	X	·	X	·	11 011 101 11 100 011	DD E3	2	6	23	
EX (SP), IY	IYH ↔ (SP + 1) IYL ↔ (SP)	·	·	X	·	X	·	11 111 101 11 100 011	FD E3	2	6	23	
LDI	(DE) ← (HL) DE ← DE + 1 HL ← HL + 1 BC ← BC − 1	·	·	X	①	X	0	11 101 101 10 100 000	ED 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	11 101 101 10 110 000	ED B0	2 2	5 4	21 16	If BC ≠ 0 If BC = 0

NOTES: ① P/V is reset 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	0	X	①	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	0	X	0	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	↕	② ↕	X	↕	X	① ↕	1	•	11 101 101 ED 10 100 001 A1				
CPIR	A − (HL) HL ← HL + 1 BC ← BC − 1 Repeat until A = (HL) or BC = 0	↕	② ↕	X	↕	X	① ↕	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	↕	② ↕	X	↕	X	① ↕	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	↕	② ↕	X	↕	X	① ↕	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	No. of Bytes	No. of M Cycles	No. of T States	Comments
DAA	Converts acc. contents into packed BCD following add or subtract with packed BCD operands	↕	↕	↕	P	•	↕	27	1	1	4	Decimal adjust accumulator
CPL	A = $\bar{A}$	•	•	1	•	1	•	2F	1	1	4	Complement accumulator (one's complement)
NEG	A = 0 − A	↕	↕	↕	V	1	↕	ED 44	2	2	8	Negate acc. (two's complement)
CCF	CY = $\bar{CY}$	•	•	↕	X	0	↕	3F	1	1	4	Complement carry flag
SCF	CY = 1	•	•	0	0	1	•	37	1	1	4	Set carry flag
NOP	No operation	•	•	•	•	•	•	00	1	1	4	
HALT	CPU halted	•	•	•	•	•	•	76	1	1	4	
DI	IFF = 0	•	•	•	0	•	•	F3	1	1	4	
EI	IFF = 1	•	•	•	1	•	•	FB	1	1	4	
IM 0	Set interrupt mode 0	•	•	•	•	•	•	ED 46	2	2	8	
IM 1	Set interrupt mode 1	•	•	•	•	•	•	ED 56	2	2	8	
IM 2	Set interrupt mode 2	•	•	•	•	•	•	ED 5E	2	2	8	

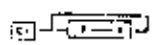
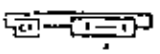
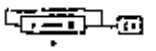
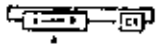
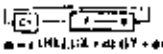
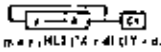
NOTES: ↕ indicates the flag changes according to the result of the operation.
0 indicates the flag is reset.
• indicates the flag is not affected by the operation.

16-Bit Arithmetic Group

Mnemonic	Symbolic Operation	S	Z	H	P/V	N	C	Opcode	No. of Bytes	No. of M Cycles	No. of T States	Comments
ADC HL, ss	HL = HL + ss + CY	↕	↕	X	V	0	↕	ED 4A	2	4	15	ss: 00 BC, 01 DE, 10 HL, 11 SP
SBC HL, ss	HL = HL − ss − CY	↕	↕	X	V	1	↕	ED 42	2	4	15	
ADD IX, pp	IX = IX + pp	•	•	X	X	0	↕	DD 09	2	4	15	pp: 00 BC, 01 DE, 10 IX, 11 SP
ADD IY, rr	IY = IY + rr	•	•	X	X	0	↕	FD 19	2	4	15	rr: 00 BC, 01 DE, 10 IY, 11 SP
INC ss	ss = ss + 1	•	•	X	•	X	•	03	1	1	6	
INC IX	IX = IX + 1	•	•	X	•	X	•	DD 23	2	2	10	
INC IY	IY = IY + 1	•	•	X	•	X	•	FD 23	2	2	10	
DEC ss	ss = ss − 1	•	•	X	•	X	•	0B	1	1	6	
DEC IX	IX = IX − 1	•	•	X	•	X	•	DD 2B	2	2	10	
DEC IY	IY = IY − 1	•	•	X	•	X	•	FD 2B	2	2	10	

NOTES: ss can be any of the register pairs BC, DE, HL, SP.
pp can be any of the register pairs BC, DE, IX, SP.
rr can be any of the register pairs BC, DE, IY, SP.

Rotate and Shift Group

Mnemonic	Symbolic Operation	S	Z	H	P/V	N	C	Opcode	No. of Bytes	No. of M Cycles	No. of T States	Comments
RLCA		•	•	0	0	X	↕	07	1	1	4	Rotate left circular accumulator
RLA		•	•	0	0	X	↕	17	1	1	4	Rotate left accumulator
RRCA		•	•	0	0	X	↕	0F	1	1	4	Rotate right circular accumulator
RRA		•	•	0	0	X	↕	1F	1	1	4	Rotate right accumulator
RLC r		↕	↕	0	P	X	↕	CB 00	2	2	8	Rotate left circular register r
RLC (HL)		↕	↕	0	P	X	0	CB 06	2	4	15	r: 000 B, 001 C, 010 D, 011 E, 100 H, 101 L, 111 A
RLC (IX + d)	 (IX + d)	↕	↕	0	P	X	↕	DD CB d 06	4	6	23	
RLC (IY + d)		↕	↕	0	P	X	↕	FD CB d 06	4	6	23	
RL m		↕	↕	0	P	X	↕	CB 10				The operation and flag status are as shown for RLC s. To form new opcode replace 06 of RLC (IX + d) with shown value
RR m		↕	↕	0	P	X	↕	CB 18				

Rotate and Shift Group (Continued)

Mnemonic	Symbolic Operation	S	Z	H	P/V	N	C	Op-Code 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)	↕	↕	0	P	0	↕	011					
SLA m	m ≡ r, (HL), (IX+d), (IY+d)	↕	↕	0	P	0	↕	100					
SRA m	m ≡ r, (HL), (IX+d), (IY+d)	↕	↕	0	P	0	↕	101					
SRL m	m ≡ r, (HL), (IX+d), (IY+d)	↕	↕	0	P	0	↕	111					
RLD	A, (HL)	↕	↕	0	P	0	•	11 101 101 01 101 111	ED 6F	2	5	18	Rotate digit left and right between the accumulator and location (HL).
RRD	A, (HL)	↕	↕	0	P	0	•	11 101 101 01 100 111	ED 67	2	5	18	The content of the upper half of the accumulator is unaffected.

Bit Set, Reset and Test Group

Mnemonic	Symbolic Operation	S	Z	H	P/V	N	C	Op-Code 76 543 210	Hex	No. of Bytes	No. of M Cycles	No. of T States	Comments
BIT b, r	$Z \leftarrow \overline{r_b}$	X	↕	1	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
BIT b, (HL)	$Z \leftarrow \overline{(HL)_b}$	X	↕	1	X	0	•	11 001 011 01 b 110	CB	2	3	12	
BIT b, (IX+d)$_b$	$Z \leftarrow \overline{(IX+d)_b}$	X	↕	1	X	0	•	11 011 101 11 001 011 ← d → 01 b 110	DD CB	4	5	20	b Bit Tested 000 0 001 1 010 2 011 3 100 4 101 5 110 6 111 7
BIT b, (IY+d)$_b$	$Z \leftarrow \overline{(IY+d)_b}$	X	↕	1	X	0	•	11 111 101 11 001 011 ← d → 01 b 110	FD CB	4	5	20	
SET b, r	$r_b \leftarrow 1$	•	•	•	•	•	•	11 001 011 [11] b r	CB	2	2	8	
SET b, (HL)	$(HL)_b \leftarrow 1$	•	•	•	•	•	•	11 001 011 [11] b 110	CB	2	4	15	
SET b, (IX+d)	$(IX+d)_b \leftarrow 1$	•	•	•	•	•	•	11 011 101 11 001 011 ← d → [11] b 110	DD CB	4	6	23	
SET b, (IY+d)	$(IY+d)_b \leftarrow 1$	•	•	•	•	•	•	11 111 101 11 001 011 ← d → [11] b 110	FD CB	4	6	23	
RES b, m	$m_b \leftarrow 0$ m ≡ r, (HL), (IX+d), (IY+d)	•	•	•	•	•	•	[10]					To form new opcode replace [11] of SET b, s with [10]. Flags and time states for SET instruction.

NOTES: The notation m_b indicates location m bit b (0 to 7).

Jump Group

Mnemonic	Symbolic Operation	S	Z	H	P/V	N	C	Op-Code 76 543 210	Hex	No. of Bytes	No. of M Cycles	No. of T States	Comments
JP nn	PC ← nn	•	•	•	•	•	•	11 000 011 ← n → ← n →	C3	3	3	10	
JP cc, nn	If condition cc is true PC ← nn, otherwise continue	•	•	•	•	•	•	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	•	•	•	•	•	•	00 011 000 ← e−2 →	18	2	3	12	
JR C, e	If C = 0, continue If C = 1, PC ← PC + e	•	•	•	•	•	•	00 111 000 ← e−2 →	38	2 2	2 3	7 12	If condition not met If condition is met
JR NC, e	If C = 1, continue If C = 0, PC ← PC + e	•	•	•	•	•	•	00 110 000 ← e−2 →	30	2 2	2 3	7 12	If condition not met If condition is met
JP Z, e	If Z = 0 continue If Z = 1, PC ← PC + e	•	•	•	•	•	•	00 101 000 ← e−2 →	28	2 2	2 3	7 12	If condition not met If condition is met
JR NZ, e	If Z = 1, continue If Z = 0, PC ← PC + e	•	•	•	•	•	•	00 100 000 ← e−2 →	20	2 2	2 3	7 12	If condition not met If condition is met
JP (HL)	PC ← HL	•	•	•	•	•	•	11 101 001	E9	1	1	4	
JP (IX)	PC ← IX	•	•	•	•	•	•	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 76 543 210	Hex	No. of Bytes	No. of M Cycles	No. of T States	Comments
JP (IY)	PC ← IY	•	•	X	•	X	•	•	•	11 111 101 11 101 001	FD E9	2	2	8	
DJNZ, e	B ← B−1 If B = 0, continue If B ≠ 0, PC ← PC + e	•	•	X	•	X	•	•	•	00 010 000 ← e−2 →	10	2 2	2 3	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.

Call and Return 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
CALL nn	(SP−1) ← PC_H (SP−2) ← PC_L PC ← nn	•	•	X	•	X	•	•	•	11 001 101 ← n → ← n →	CD	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 01 001 101	ED 4D	2	4	14	
RETN[1]	Return from non maskable interrupt	•	•	X	•	X	•	•	•	11 101 101 01 000 101	ED 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

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
IN A, (n)	A ← (n)	•	•	X	•	X	•	•	•	11 011 011 ← n →	DB	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 01 r 000	ED	2	3	12	C to A_0 ~ A_7 B to A_8 ~ A_{15}
INI	(HL) ← (C) B ← B−1 HL ← HL + 1	X	‡ ①	X	X	X	X	1	X	11 101 101 10 100 010	ED 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	X	1	X	X	X	X	1	X	11 101 101 10 110 010	ED 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	X	‡ ①	X	X	X	X	1	X	11 101 101 10 101 010	ED 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	X	1	X	X	X	X	1	X	11 101 101 10 111 010	ED 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 ← n →	D3	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 01 r 001	ED	2	3	12	C to A_0 ~ A_7 B to A_8 ~ A_{15}
OUTI	(C) ← (HL) B ← B−1 HL ← HL + 1	X	‡ ①	X	X	X	X	1	X	11 101 101 10 100 011	ED 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	X	1	X	X	X	X	1	X	11 101 101 10 110 011	ED 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	X	‡ ①	X	X	X	X	1	X	11 101 101 10 101 011	ED 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	?	?	?	?	1	?	11 101 101 Bin ED 10 111 011 BB	2	5 If B ≠ 0 4 If B = 0	21 16	C on A_{23} ← A_{7} B on A_{15} ← 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	↕	↕	V	V	0	↕	8-bit add or add with carry
SUB s; SBC A, s; CP s; NEG	↕	↕	V	V	1	↕	8-bit subtract, subtract with carry, compare and negate accumulator
AND s	↕	↕	1	P	0	0	Logical operations
OR s; XOR s	↕	↕	0	P	0	0	
INC s	↕	↕	V	V	0	•	8-bit increment
DEC s	↕	↕	V	V	1	•	8-bit decrement
ADD HL, ss	•	•	X	•	0	↕	16-bit add
ADC HL, ss	↕	↕	V	V	0	↕	16-bit add with carry
SBC HL, ss	↕	↕	V	V	1	↕	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	↕	↕	0	P	0	↕	Rotate and shift locations
RLD, RRD	↕	↕	0	P	0	•	Rotate digit left and right
DAA	↕	↕	↕	P	•	↕	Decimal adjust accumulator
CPL	•	•	1	•	1	•	Complement accumulator
SCF	•	•	0	0	0	1	Set carry
CCF	•	•	X	•	0	↕	Complement carry
IN r, (C)	↕	↕	0	P	0	•	Input register indirect
INI, INIR, IND, INDR, OUTI, OTIR, OUTD, OTDR	?	?	?	?	1	•	Block input and output; Z = 0 if B ≠ 0, otherwise Z = 1
LDI, LDIR	X	X	0	↕	0	•	Block transfer instructions; P/V = 1 if BC ≠ 0, otherwise P/V = 0
LDD, LDDR	X	X	0	↕	0	•	
CPI, CPIR, CPD, CPDR	↕	↕	↕	↕	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	↕	↕	0	IFF	0	•	The content of the interrupt enable flip-flop (IFF) is copied into the P/V flag
BIT b, s	?	↕	1	?	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	8-bit value in range < 0, 255 >.
nn	16-bit value in range < 0, 65535 >.

Pin Descriptions

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 nonmaskable 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

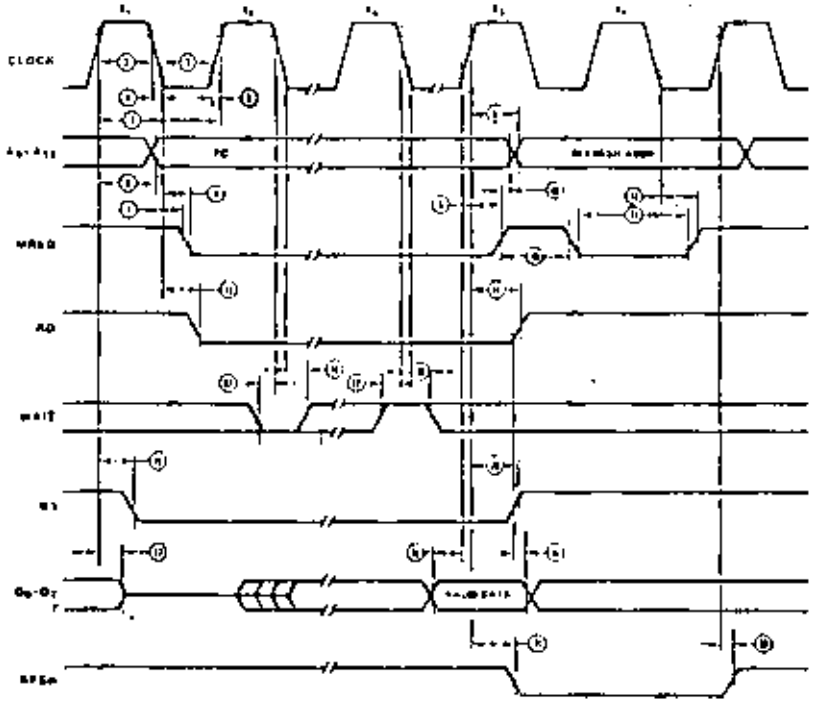
The Z80 CPU executes instructions by proceeding through a specific sequence of operations:

- 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 of 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.

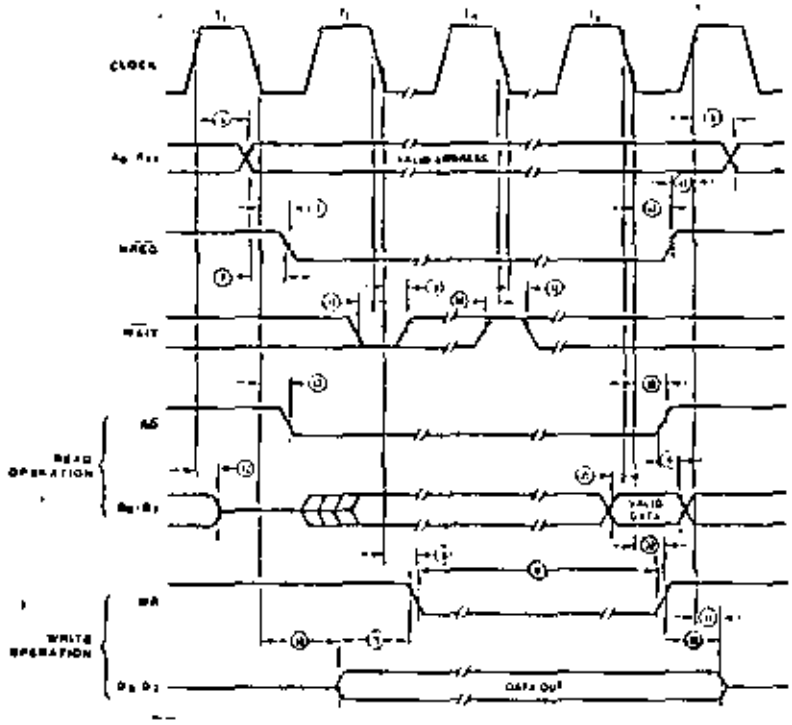
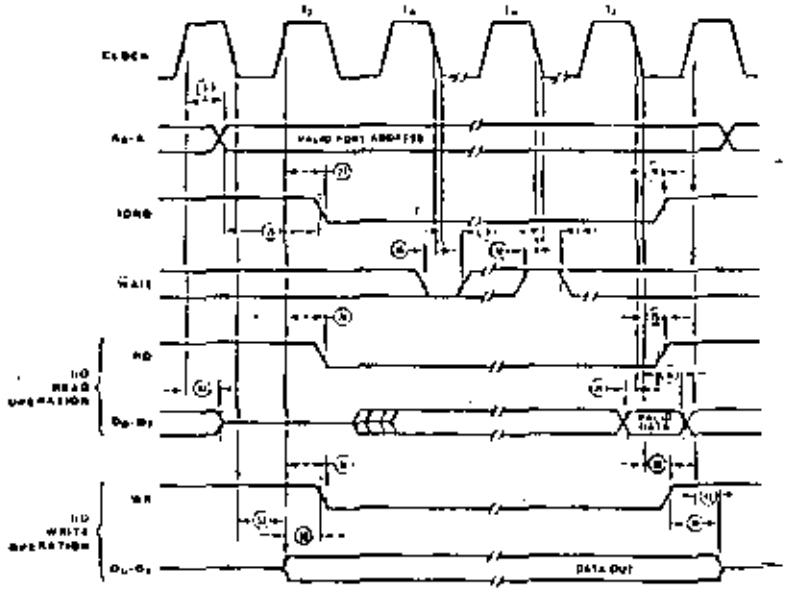


Figure 6. Memory Read or Write Cycles

CPU Timing (Continued)

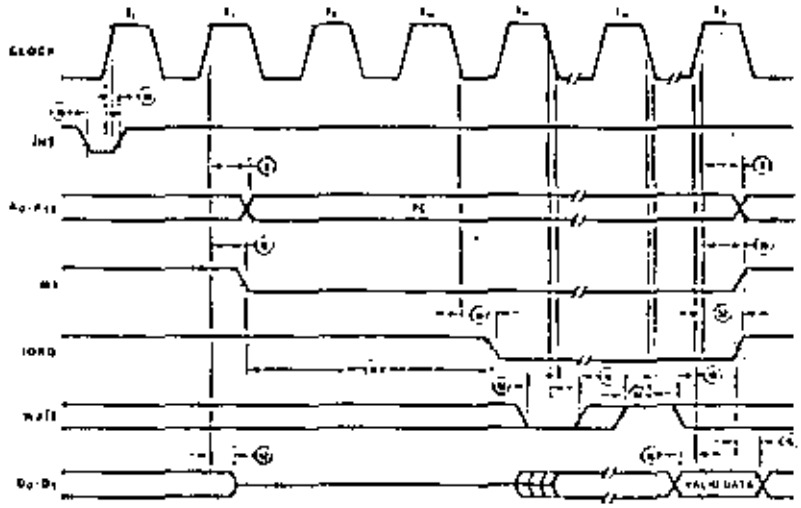
Input or Output Cycles. Figure 7 shows the timing for an I/O read or I/O write operation. During I/O operations, the CPU automatically inserts a single Wait state (T_w). This extra Wait state allows sufficient time for an I/O port to decode the address and the port address lines.



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.

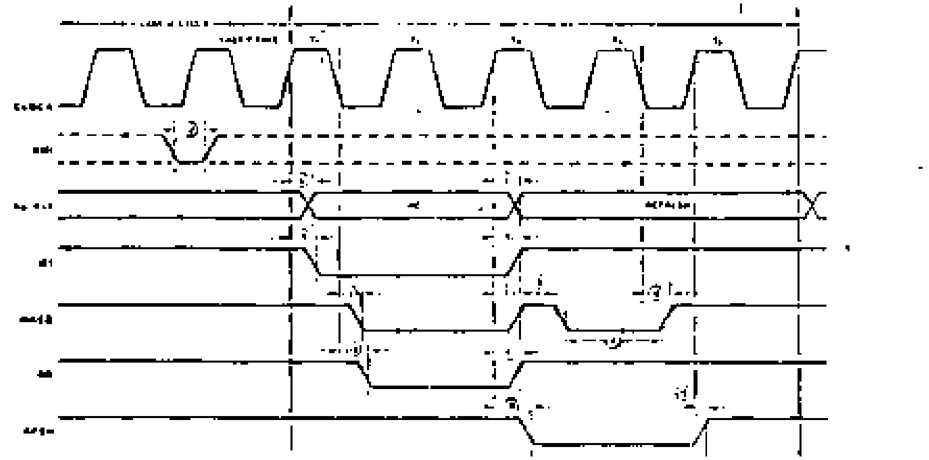


NOTE: 1) T_L = Last state of previous instruction. 2) Two Wait cycles automatically inserted by CPU.

Figure 8. Interrupt Request/Acknowledge Cycle

CPU Timing (Continued)

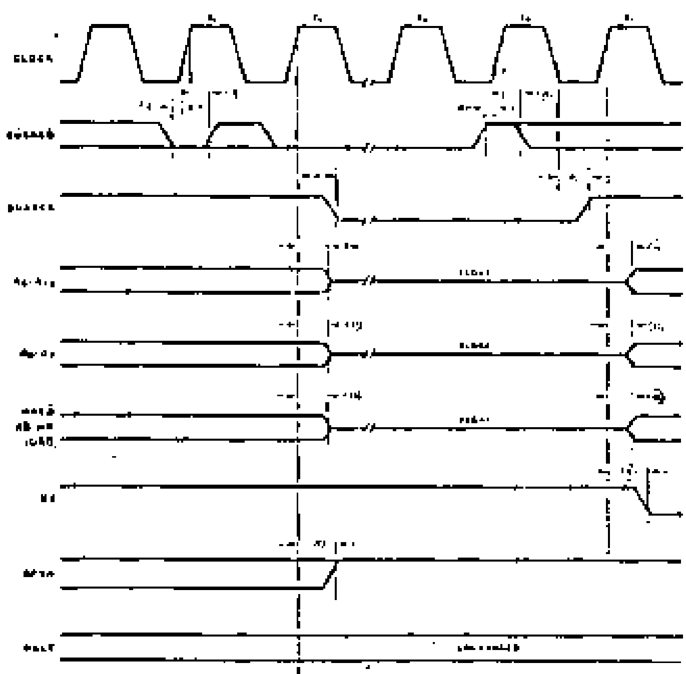
Non-Maskable Interrupt Request Cycle. $\overline{NMI}$ is sampled at the same time as the maskable interrupt input $\overline{INT}$ but has higher priority and cannot be disabled under software control. The subsequent timing is similar to that of a normal memory read operation except that data put on 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.



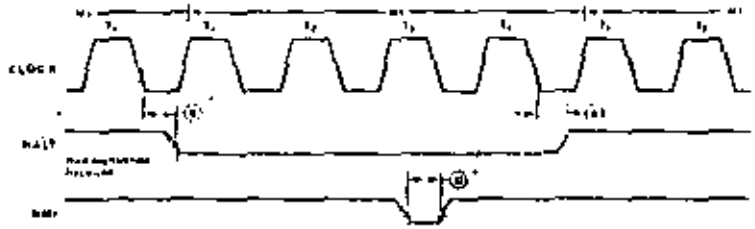
NOTE: T_L = Last state of any M cycle. T_X = An arbitrary clock cycle used by requesting device.

Figure 10. Bus Request/Acknowledge Cycle

Z80 CPU

CPU Timing (Continued)

Halt Acknowledge Cycle. When the CPU receives a HALT instruction, it executes NOP states until either an $\overline{INT}$ or $\overline{NMI}$ input is received. While in the Halt state, the 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).

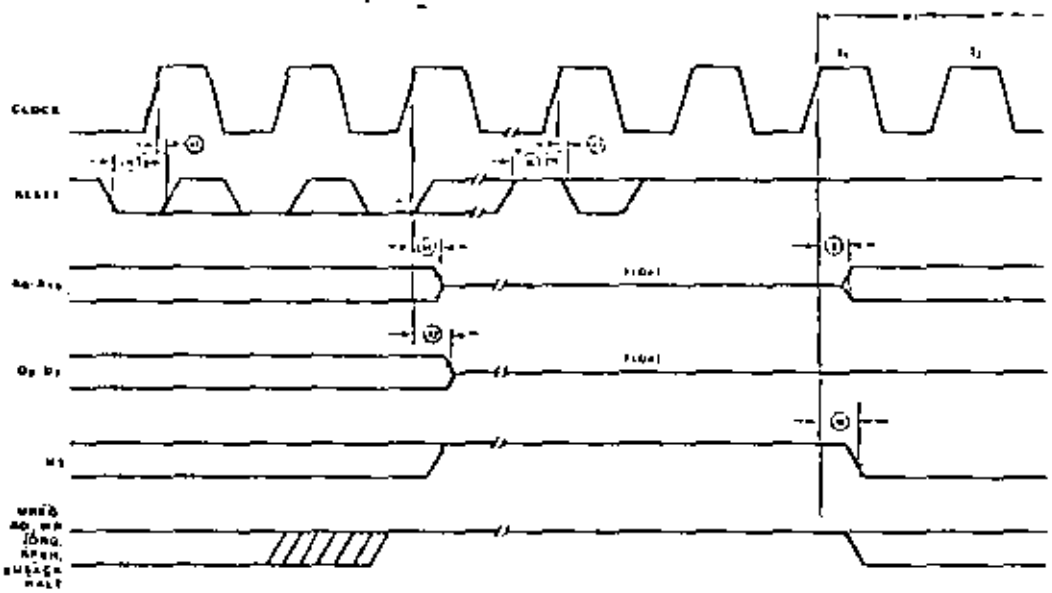


Figure 12. Reset Cycle

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	TdCh(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	TdD(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	TdCr(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	TsINTr(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	930*		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 given in the table recalculate 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, TwCl 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
14	TdA(IORQf)	TcC − 60	TcC − 70	TcC − 55
29	TdD(WRf)	TcC − 210	TcC − 170	TcC − 140
31	TwWRl	TcC − 40	TcC − 30	TcC − 30
33	TdD(WRf)	TwCl + TrC − 180	TwCl + TrC − 140	TwCl + TrC − 140
35	TdWRf(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 + TwCl + TfC − 80	2TcC + TwCl + TfC − 65	2TcC + TwCl + TfC − 50

AC Test Conditions
V_{IH} = 2.0 V
V_{IL} = 0.8 V
V_{thC} = V_{CC} −0.5 V
V_{thC} = 0.45 V

V_{OH} = 2.0 V
V_{OL} = 0.8 V
FLOAT = ±0.5 V

Absolute Maximum Ratings

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

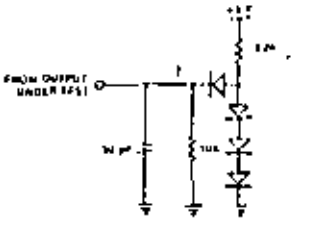
Stresses greater than those listed under Absolute Maximum Ratings may cause permanent damage to the device. This is a stress rating only; operation of the device at any condition above those indicated in the operational sections of these specifications is not implied. Exposure to absolute maximum rating 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, $+4.75\ V \le V_{CC} \le +5.25\ V$
- −40°C to +85°C, $+4.75\ V \le V_{CC} \le +5.25\ V$
- −55°C to +125°C, $+4.5\ V \le V_{CC} \le +5.5\ V$

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[1]	mA	
	Z80B		200	mA	
I_{LI}	Input Leakage Current		10	μA	V_{IN} = 0 to V_{CC}
I_{LOAK}	3 State Output Leakage Current in Float	−10	10[2]	μA	V_{OUT} = 0.4 to V_{CC}

1. For int/ext clock pins, I_{CC} is 2μA mA
2. Typical rate for Z80A is 40 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
$C_{\phi Clock}$	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

Ordering Information

Product Number	Package/ Temp	Speed	Description	Product Number	Package/ Temp	Speed	Description
Z8400	CE	2.5 MHz	Z80 CPU (40-pin)	Z8400A	DE	4.0 MHz	Z80A CPU (40-pin)
Z8400	CM	2.5 MHz	Same as above	Z8400A	DS	4.0 MHz	Same as above
Z8400	CMB	2.5 MHz	Same as above	Z8400A	PE	4.0 MHz	Same as above
Z8400	CS	2.5 MHz	Same as above	Z8400A	PS	4.0 MHz	Same as above
Z8400	DE	2.5 MHz	Same as above	Z8400B	CE	6.0 MHz	Z80B CPU (40-pin)
Z8400	DS	2.5 MHz	Same as above	Z8400B	CM	6.0 MHz	Same as above
Z8400	PE	2.5 MHz	Same as above	Z8400B	CMB	6.0 MHz	Same as above
Z8400	PS	2.5 MHz	Same as above	Z8400B	CS	6.0 MHz	Same as above
Z8400A	CE	4.0 MHz	Z80A CPU (40-pin)	Z8400B	DE	6.0 MHz	Same as above
Z8400A	CM	4.0 MHz	Same as above	Z8400B	DS	6.0 MHz	Same as above
Z8400A	CMB	4.0 MHz	Same as above	Z8400B	PE	6.0 MHz	Same as above
Z8400A	CS	4.0 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 +85°C with MIL-STD 883 Class B processing, S = 0°C to +70°C

MICROPROCESADORES Y MICROCOMPUTADORAS

Z8420
Z80 PIO Parallel
Input/Output Controller

Septiembre, 1983

Z8420
Z80' PIO Parallel Input/Output Controller

Zilog

Product Specification

March 1981

Z80 PIO

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.

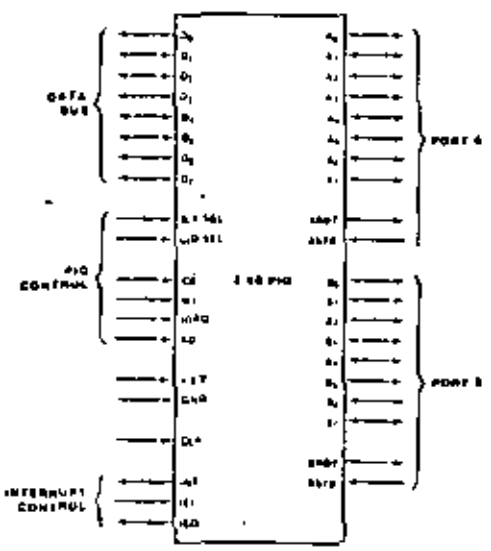


Figure 1. Pin Functions

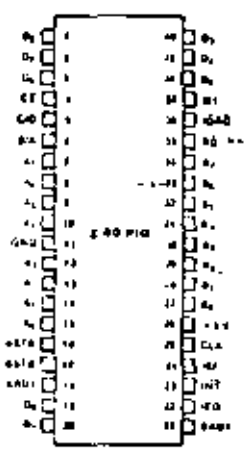


Figure 2. Pin Assignments

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.

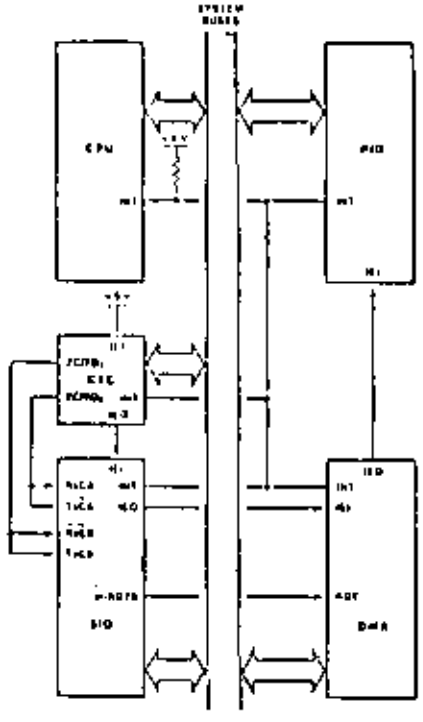


Figure 3. PIO in a Typical Z80 Family Environment

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.

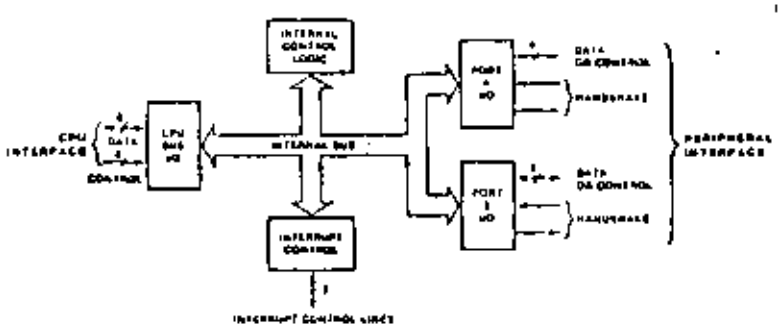


Figure 4. Block Diagram

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.

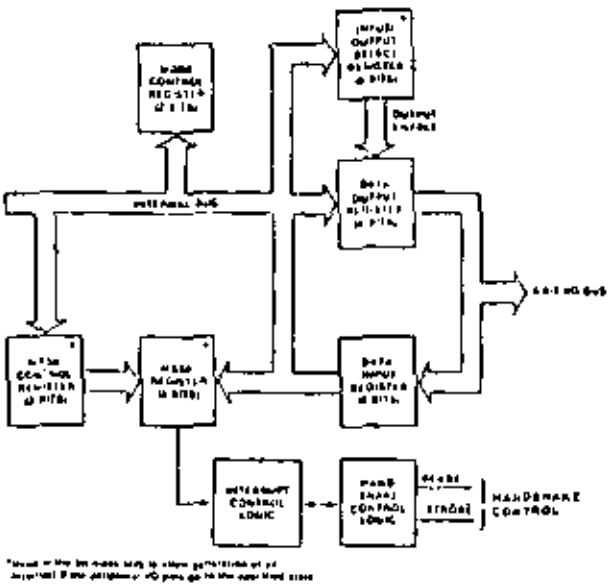


Figure 5. Typical Port I/O Block Diagram

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 level 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).

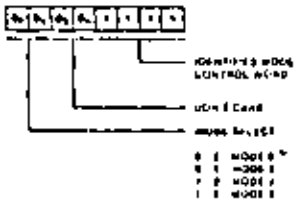


Figure 6. Mode Control Word

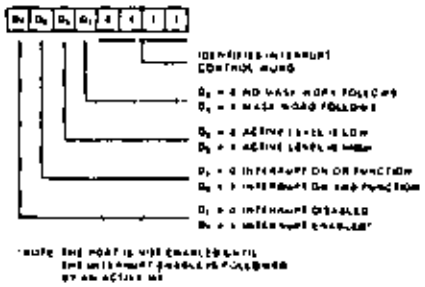


Figure 9. Interrupt Control Word

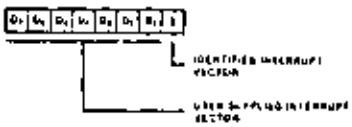


Figure 7. Interrupt Vector Word

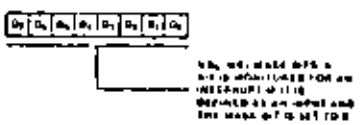


Figure 10. Mask Control Word

Figure 8. I/O Register Control Word

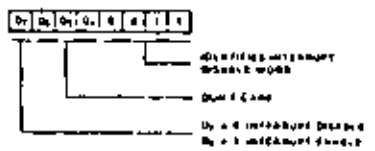


Figure 11. Interrupt Disable Word

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.

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 placed 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. Other address bit A_0 from the CPU is used for that 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. Other 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)

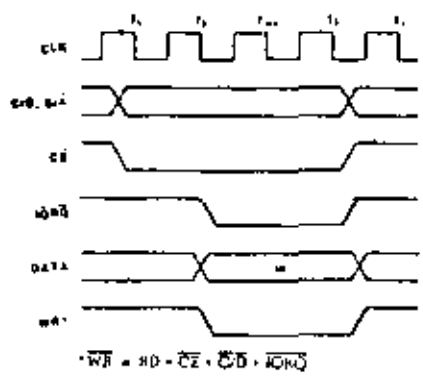
$\overline{M1}$, *Machine Cycle* (input from CPU, active Low). This signal is used as a sync pulse to control several internal PIO operations. When both the $\overline{M1}$ and $\overline{RD}$ signals are active, the Z-80 CPU is fetching an instruction from memory. Conversely, when both $\overline{M1}$ 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.

$\overline{RD}$, *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.

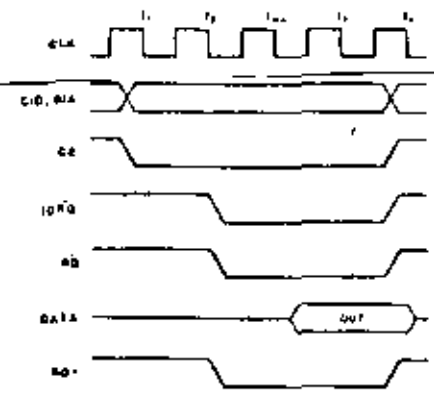


$^*\overline{WR} = RD \cdot \overline{CE} \cdot \overline{C/D} \cdot \overline{IORQ}$

Figure 12. Write Cycle Timing

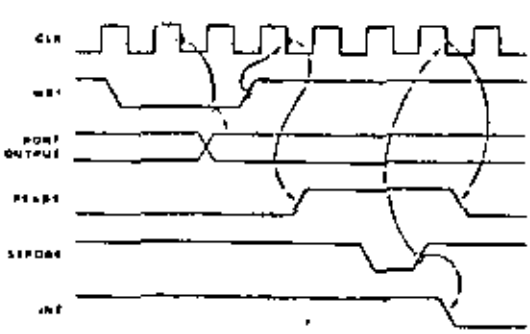
Read Cycle. Figure 13 illustrates the timing for reading the data input from an external device to one of the Z-80 PIO ports. No Wait states are allowed for reading the PIO other than the automatically inserted T_{WA}.

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.



$^*\overline{RD} = \overline{RD} \cdot \overline{CE} \cdot \overline{C/D} \cdot \overline{IORQ}$

Figure 13. Read Cycle Timing

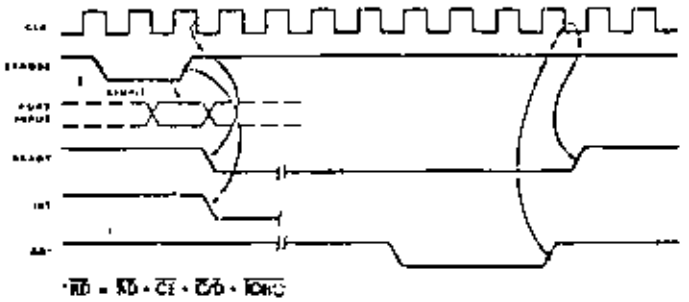


$^*\overline{WR} = RD \cdot \overline{CE} \cdot \overline{C/D} \cdot \overline{IORQ}$

Figure 14. Mode 0 Output Timing

Timing (Continued)

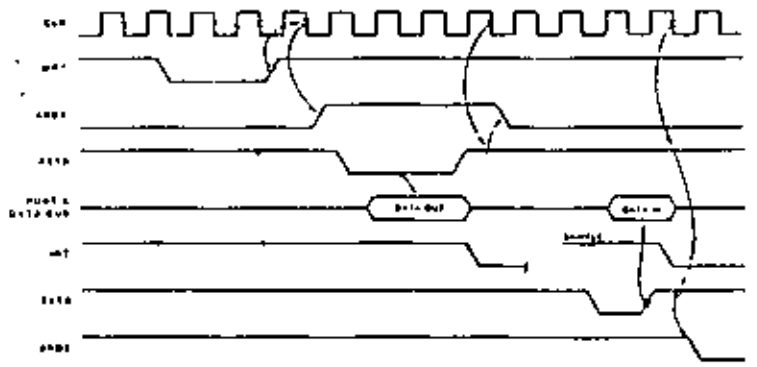
Input Mode (Mode 1). When $\overline{STROBE}$ goes Low, data is loaded into the selected port input register (Figure 15). The next rising edge of strobe activates $\overline{INT}$, if Interrupt Enable is set and this is the highest-priority requesting device. The following falling edge of CLK resets 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.



$\cdot\overline{RD} = \overline{RD} \cdot \overline{CE} \cdot \overline{C/D} \cdot \overline{IORQ}$

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 10). 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.



$\cdot\overline{WR} = \overline{RD} \cdot \overline{CE} \cdot \overline{C/D} \cdot \overline{IORQ}$

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.

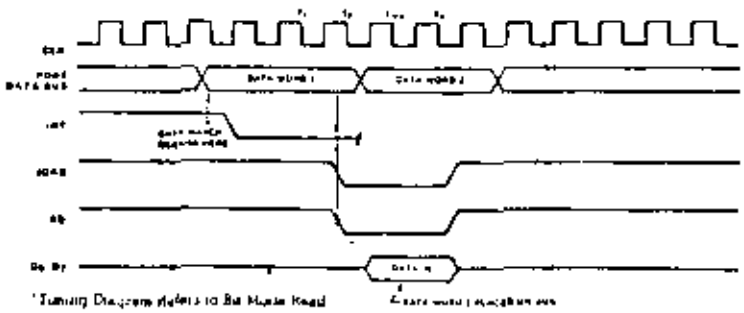


Figure 17. Mode 3 Bit Mode Timing

Interrupt Acknowledge Timing. During 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.

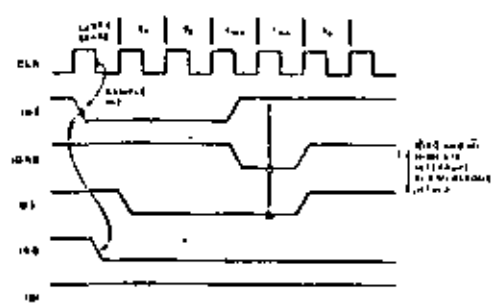


Figure 18. Interrupt Acknowledge Timing

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.

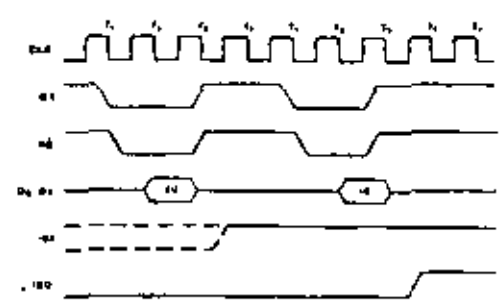
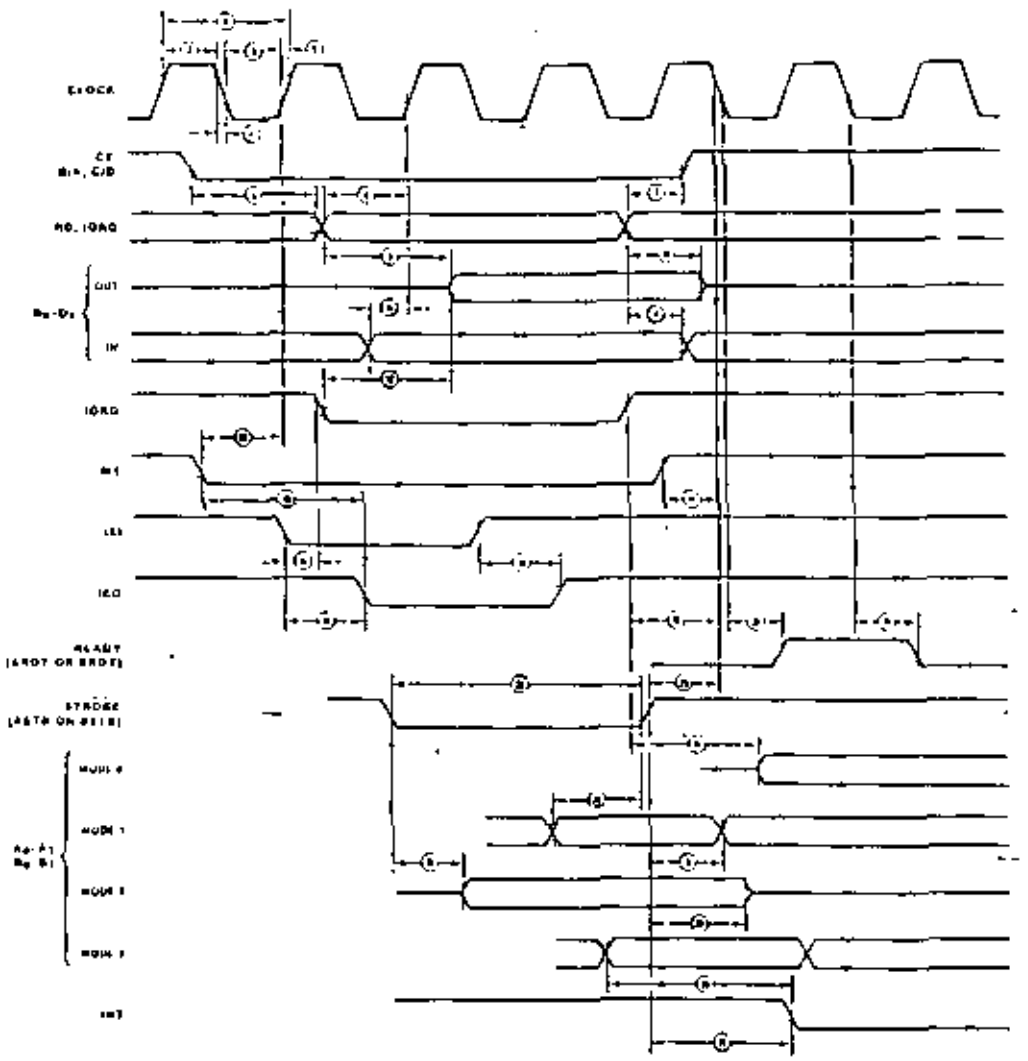


Figure 19. Return From Interrupt

AC Characteristics



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(DOz)	$\overline{RD}$, $\overline{IORQ}$ ↑ 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(Cl)	$\overline{M1}$ ↓ 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	TdIO(C)	$\overline{IORQ}$ ↓ to Clock ↑ Setup Time (To Activate READY on Next Clock Cycle)	220		200		170		
20	TdC(RDYf)	Clock ↑ to READY ↓ Delay	200		190		170		[5] CL = 50 pF
21	TdC(RDYr)	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(PDz)	$\overline{STROBE}$ ↑ 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 ns for each 50 pF increase in loading up to 200 pF max.

[4] For Mode 2 TwSTB > TsASTB.

[5] Increase these values by 2 ns for each 10 pF increase in Loading up to 250 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)TwIEA(IEOf) + TdM1(IEO) + TsIEI(IO) + TTL Buffer Delay, if any.

[8] M1 must be active for a minimum of two clock cycles to reset the PIO.

[9] Z8-B PIO numbers are preliminary and subject to change.

Z80 PIO

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.

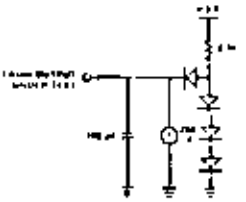
Test Conditions

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_{LO}	3 State Output/Data Bus Input Leakage Current	-10.0	+10.0	μA	$0.4 \le V_{IN} \le V_{CC}$
I_{CC}	Power Supply Current		100.0	mA	V_{CC} = 1.5V
I_{ONDP}	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 = 1 MHz)

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	Same as above
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.

57

MICROPROCESADORES Y MICROCOMPUTADORAS

Z8440
Z80 SIO Serial
Input/Output Controller

Septiembre, 1983

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 function 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

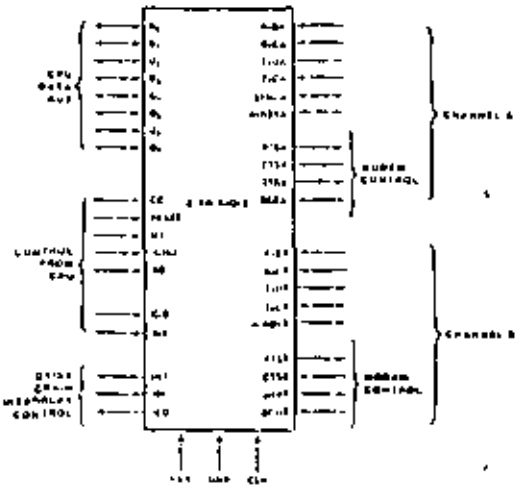


Figure 1. Z-80 SIO/2 Pin Functions

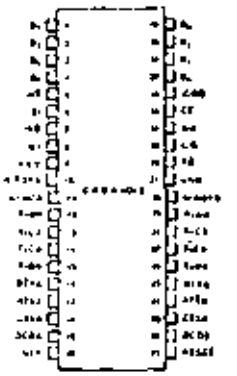


Figure 2. Z-80 SIO/2 Pin Assignments

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-rise-time 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-rise-time signals. The SIO detects pulses on these pins and interrupts the CPU on both logic level transitions. Schmitt trigger buffer-

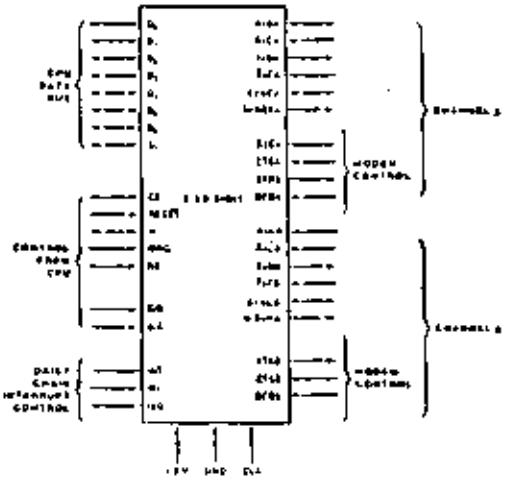


Figure 3. Z-80 SIO/1 Pin Functions

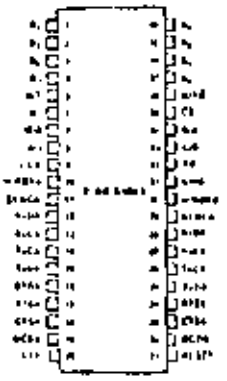


Figure 4. Z-80 SIO/1 Pin Assignments

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.

INT. *Interrupt Request* (output, open drain, active Low). When the SIO is requesting an interrupt, it pulls 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.

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.

R$\overline{x}$C$\overline{A}$, R$\overline{x}$C$\overline{B}$. *Receiver Clocks* (inputs). Receive data is sampled on the rising edge of R$\overline{x}$C. 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, R$\overline{x}$C$\overline{B}$ is bonded together with T$\overline{x}$C$\overline{B}$.

$\overline{RD}$. *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

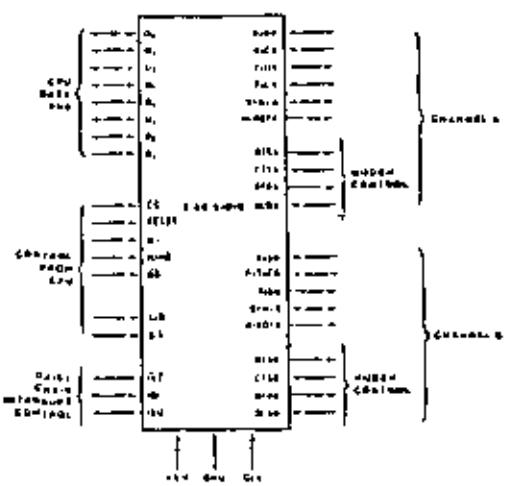


Figure 5. Z-80 SIO/0 Pin Functions

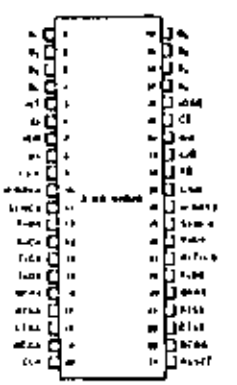


Figure 6. Z-80 SIO/0 Pin Assignments

Pin Description (Continued)

rewritten after the SIO is reset and before data is transmitted or received.

RTSA, 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.

SYNCA, 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.

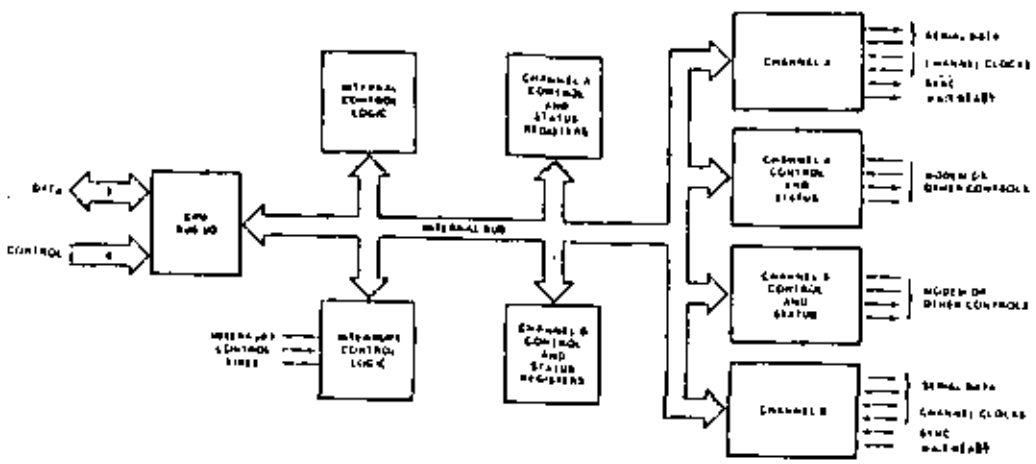


Figure 7. Block Diagram

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.

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.

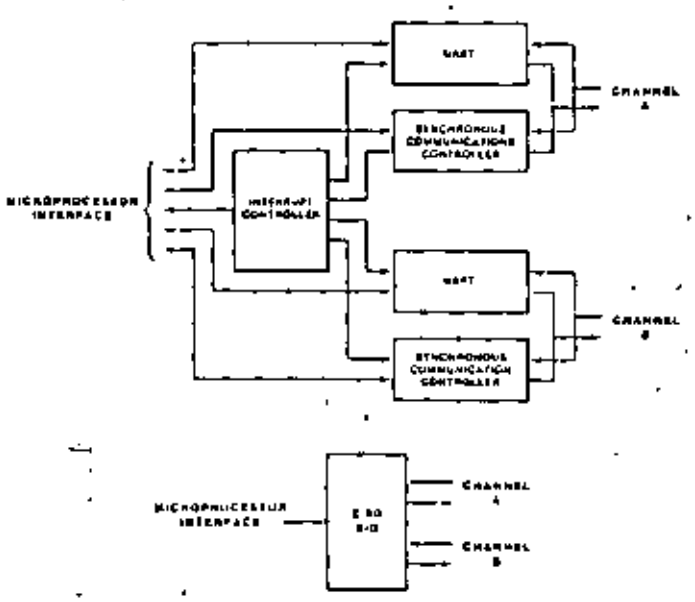


Figure 8. Conventional Devices Replaced by the Z-80 SIO

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 $\overline{SYNC}$ 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 incoming 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 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.

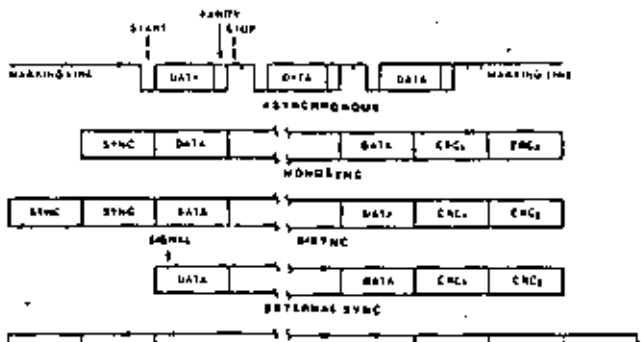


Figure 9. Some Z-80 SIO Protocols

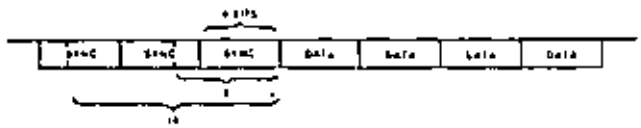


Figure 10.

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 $\overline{WAIT}$ line in the CPU block-transfer mode or as a $\overline{READY}$ line in the DMA block-transfer mode.

To a DMA controller, the SIO $\overline{READY}$ output indicates that the SIO is ready to transfer data to or from memory. To the CPU, the $\overline{WAIT}$ output indicates that the SIO is not ready to transfer data, thereby requesting the CPU to extend the I/O cycle.

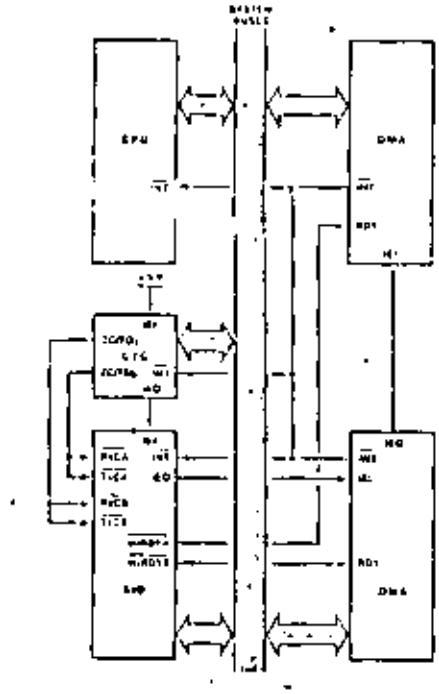


Figure 11. Typical Z-80 Environment

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).

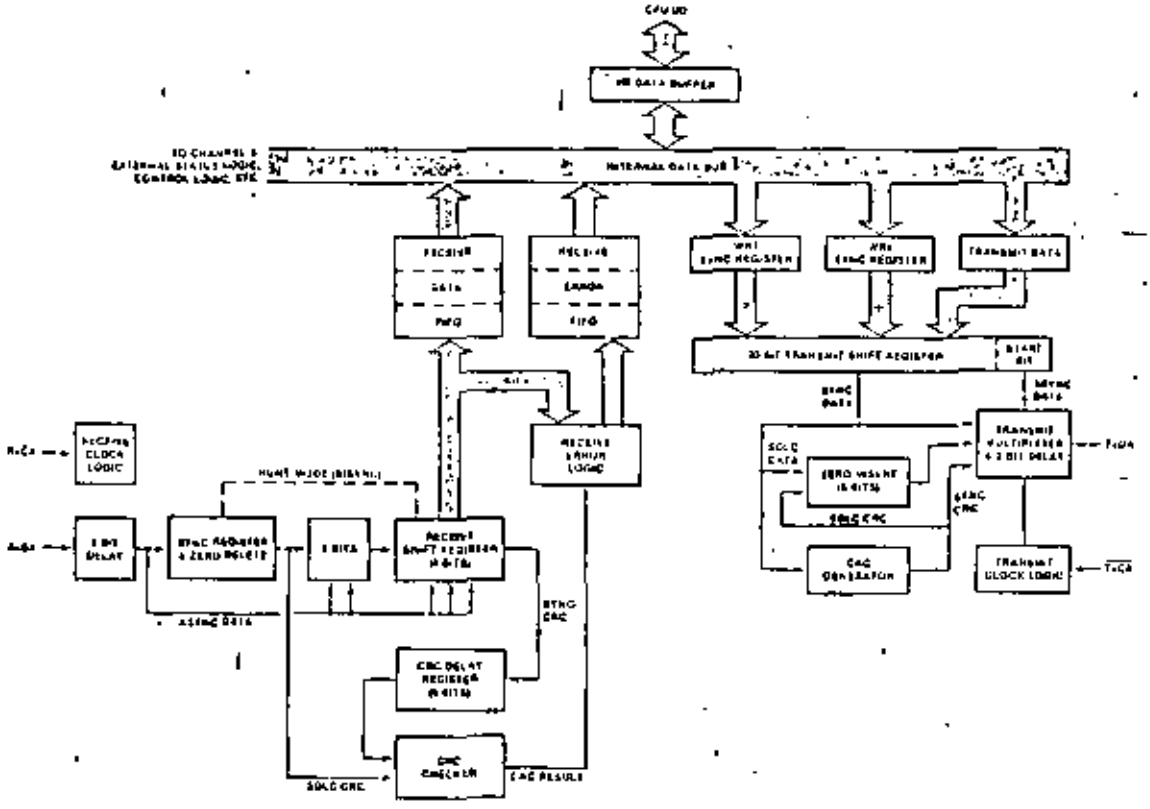


Figure 12. Transmit and Receive Data Path (Channel A)

Programming (Continued)

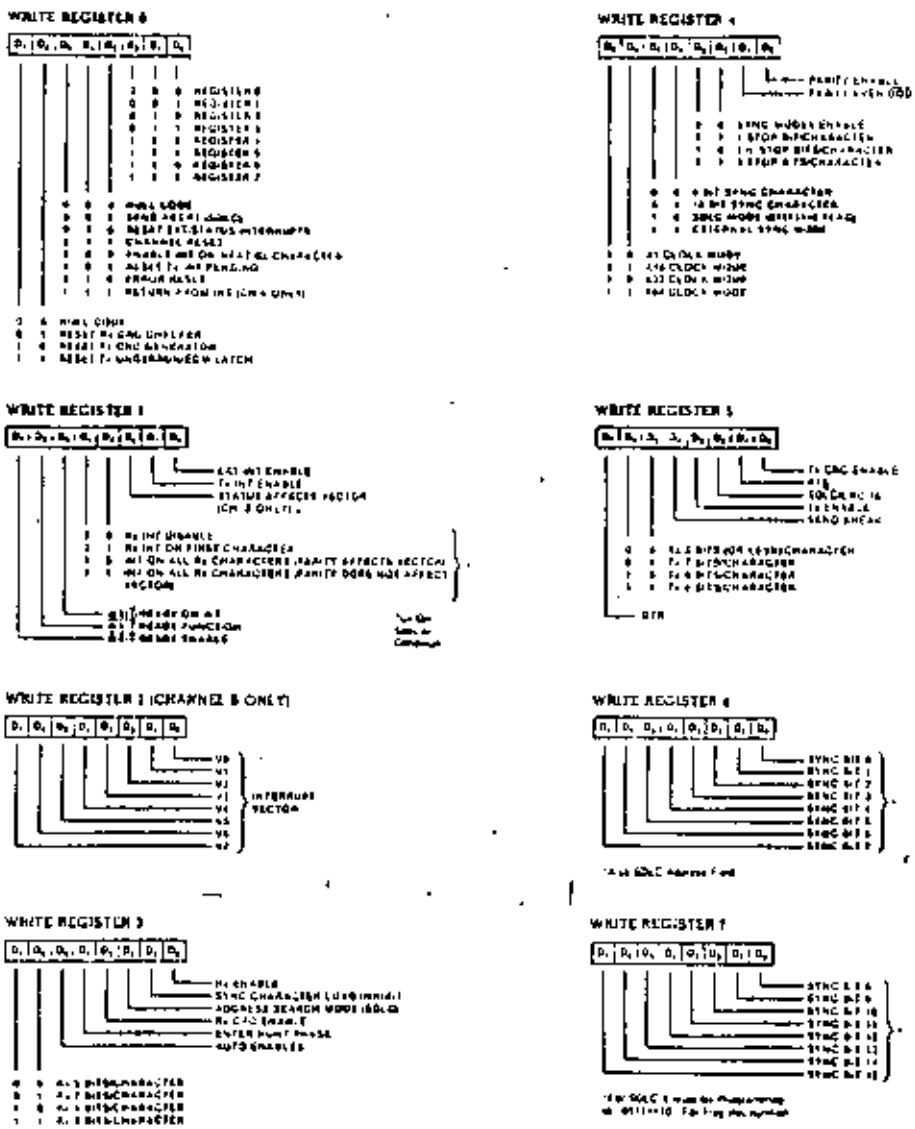


Figure 14. Write Register Bit Functions

Timing

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*.

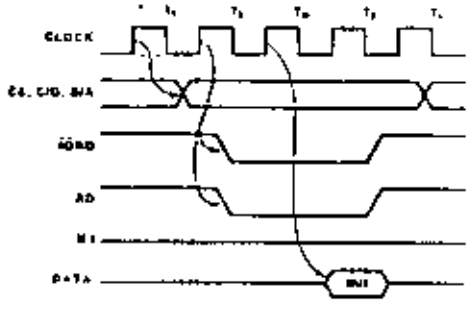


Figure 15. Read Cycle

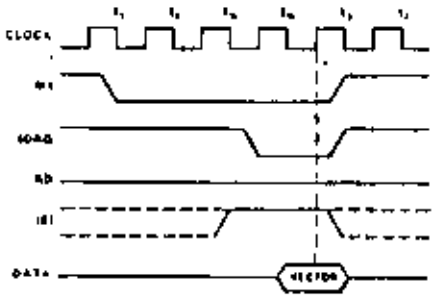


Figure 17. Interrupt Acknowledge Cycle

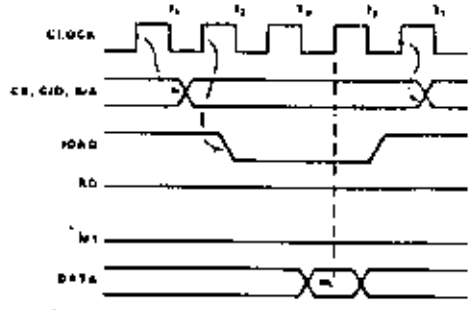


Figure 16. Write Cycle

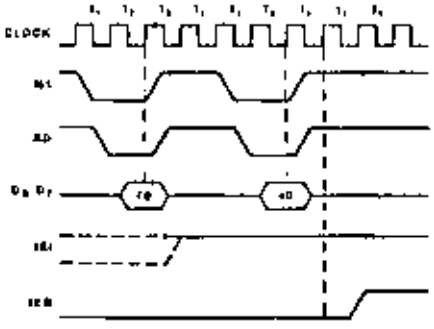


Figure 18. Return from Interrupt Cycle

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

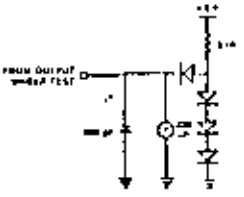
Stresses greater than those listed under Absolute Maximum Ratings may cause permanent damage to the device. This is a stress rating only; operation of the device at any condition above those indicated in the operational sections of these specifications is not implied. Exposure to absolute maximum rating 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 ≤ 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

*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	$0 \le V_{IN} \le V_{CC}$
I_Z	3-State Output/Data Bus Input Leakage Current	-10	+10	µA	$0 \le V_{IN} \le V_{CC}$
I_{LSYI}	SYNC Pin Leakage Current	-40	+10	µA	$0 \le V_{IN} \le V_{CC}$
I_{CC}	Power Supply Current		100	mA	

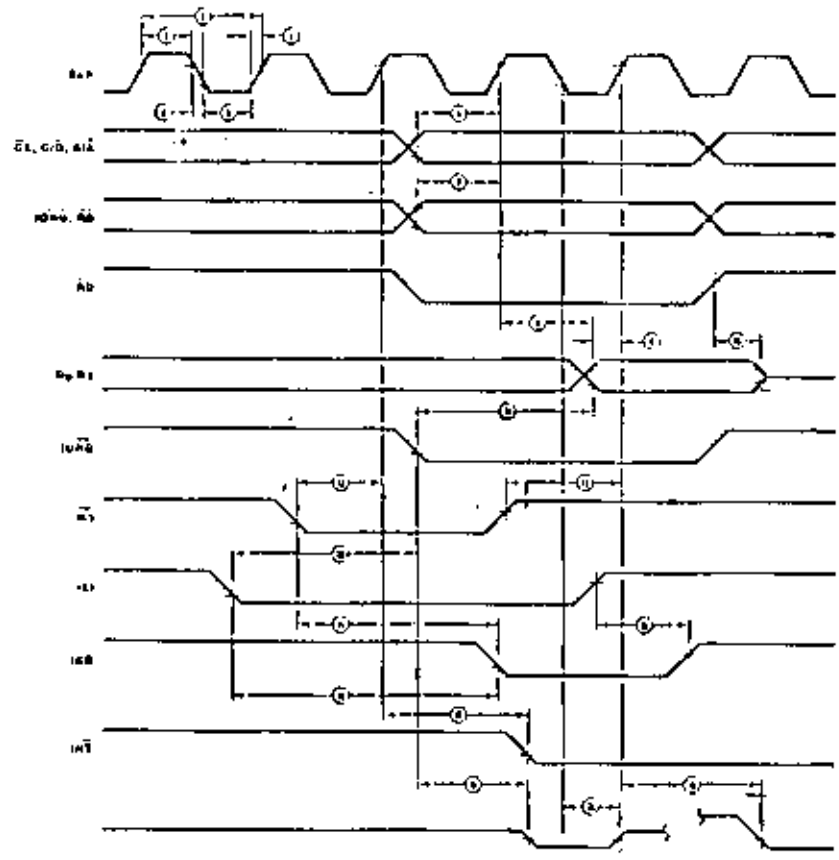
Over specified temperature and voltage range

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

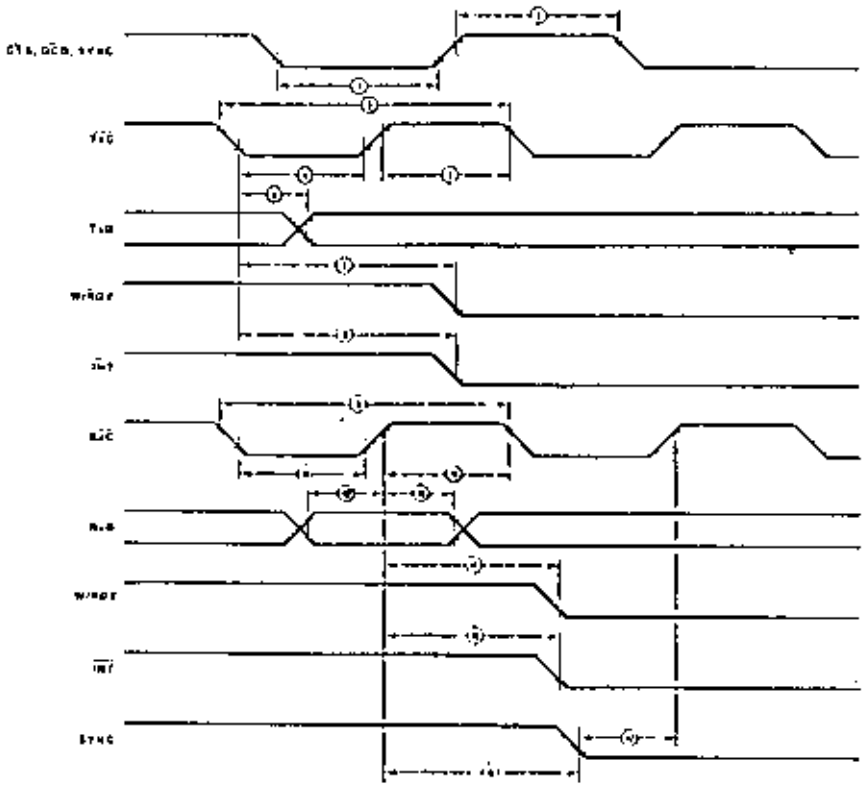
Over specified temperature range, f = 1MHz

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	Td(EI)(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	Td(IO)(W/RWf)	$\overline{IORQ}$ ↓ or $\overline{CE}$ ↓ to $\overline{W}$/$\overline{RDY}$ ↓ Delay (Wait Mode)		300		210		175	ns
19	TdC(W/RR)	Clock ↓ to $\overline{W}$/$\overline{RDY}$ ↓ Delay (Ready Mode)		120		120		100	ns
20	TdC(W/RWz)	Clock ↓ to $\overline{W}$/$\overline{RDY}$ 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/RDY)	$\overline{TxC}$ ↓ to W/$\overline{RDY}$ ↑ 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/RDY)	$\overline{RxC}$ ↑ to W/$\overline{RDY}$ ↑ 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
A RESET must be active a minimum of three consecutive Clock Cycles
*System Clocks

Ordering Information

Product Number	Package/ Temp	Speed	Description
Z8448	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

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 of Channel B must be programmed to 0.

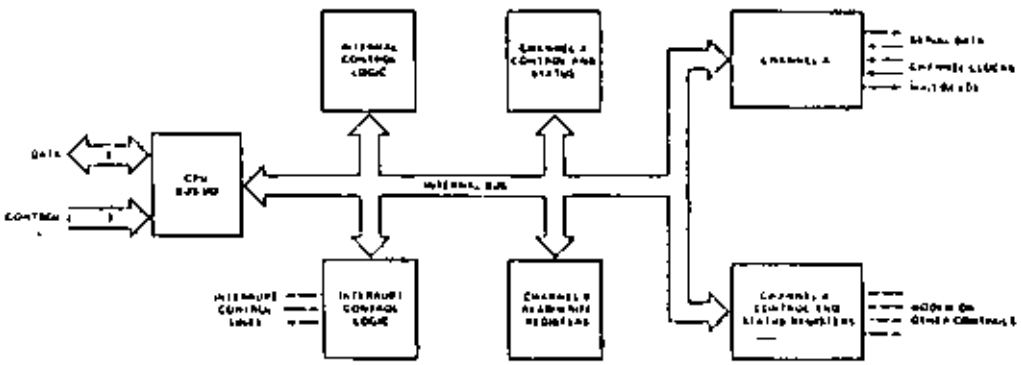


Figure 1. Block Diagram

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				
Z8449	PE	2.5 MHz	Same as above	Z8449B	CM	6.0 MHz	Same as above
Z8449	PS	2.5 MHz	Same as above	Z8449B	CMB	6.0 MHz	Same as above
Z8449A	CE	4.0 MHz	Z80A SIO/9 (40-pin)	Z8449B	CS	6.0 MHz	Same as above
				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

MICROPROCESADORES Y MICROCOMPUTADORAS

ANEXO 4

TERMINALES DE VIDEO E IMPRESORAS

ING. HÉCTOR CALVARIO MARTÍNEZ

SEPTIEMBRE, 1983

TERMINALES DE VIDEO

- TUBOS DE RAYOS CATODICOS + TECLADO
  (1840 POR WILLIAM CROOKOFT THOMPSONT FARADAY).

- TRABAJAN A MAYOR VELOCIDAD QUE LOS TTY

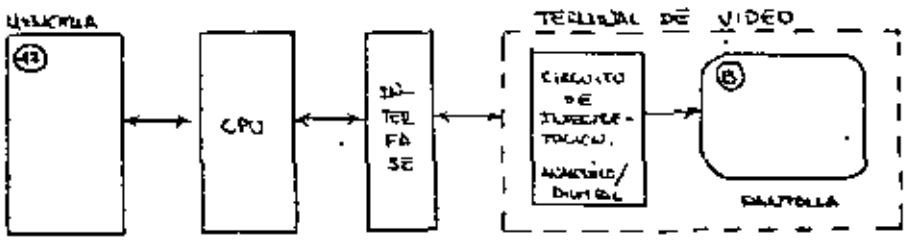
- APLICACIONES MAYORES: - DESPLIEGUE DE IMAGENES
  - GRAFICACION

- 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 RETRACAR 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 N 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 S8G)

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

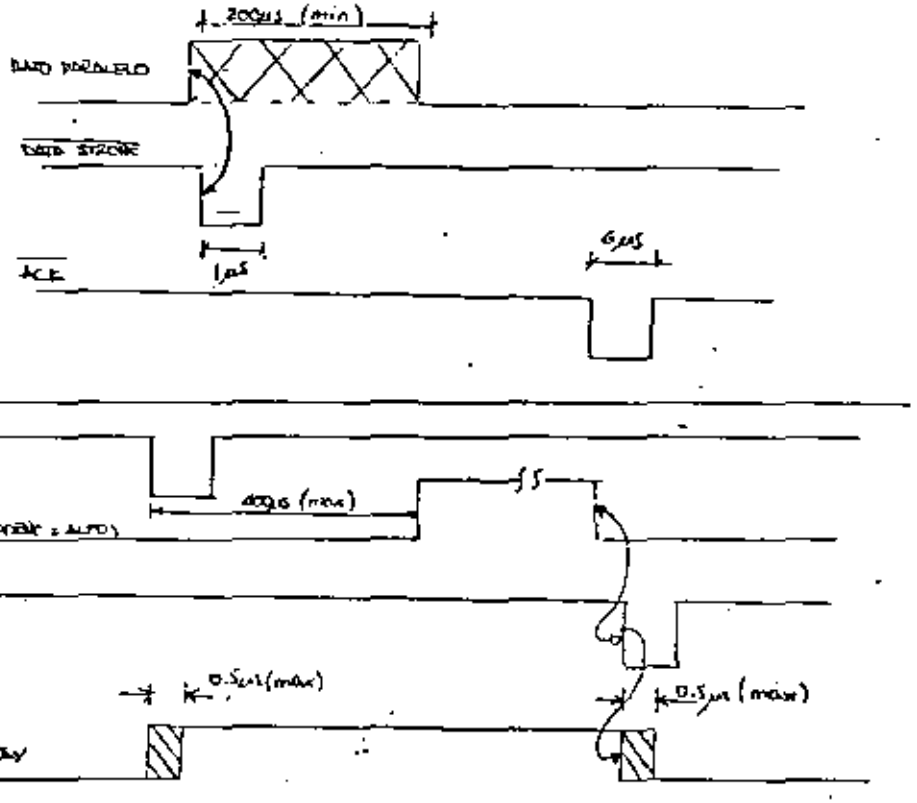
OPERACION POR LOOP DE CORRIENTE (SERIAL)

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 $\Rightarrow$ "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.



MICROPROCESADORES Y MICROCOMPUTADORAS

EDITOR DE PANTALLA

SEPTIEMBRE, 1983.

Editor de Pantalla

Introduccion

1. Modelo Conceptual

- Descripcion -

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 presione 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 !

Busque 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 LunaÁLa 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&&
          ↑       ↑
        lore      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 presiona 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&&
↑
lore                                                      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 linea en blanco )
fin
```

! Note como cada separador marca el inicio de una linea, 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 espíritu

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 espíritu&&
 ↑↑                                                      !
tope ay                                                 fin
```

! Y aquí aparece el último apuntador que usamos en la representación interna y cuyo nombre es ay !

Este apuntador es importante ya que a la linea a la que apunta se copia en el renglón, y si se modifica el contenido de el renglón al teclear algo nuevo, ay nos sirve como referencia para actualizar la representación interna.

Regresando a el ejemplo, si usted ahora baja con la flechita una linea y no modificó la anterior la representación interna se verá :

```
 ¿La Luna¿La luz, a la que, lentamente, enamora mi espiritú¿¿
 ↑       ↑                                                 ↑
lope     au                                               fin
```

Y en el renglón tendremos :

```
 "La luz, a la que, lentamente, enamora mi espiritú
  ↑
renglon
  ↑
  ax
```

! Note como au siempre apunta a el primer caracter de la linea !

Si en las condiciones actuales usted presiona la tecla de return entonces tendremos :

La representacibn interna se verá :

```
 ¿La Luna¿La luz, a la que, lentamente, enamora mi espiritú¿¿¿
 ↑                                                        ↑↑
lope                                                     au fin
```

Y en el renglón tendremos :

```
 ( linea en blanco )
  ↑
renglon
  ↑
  ax
```

! Note como en la representacibn interna dos separadores, uno seguido de otro, representan una linea en blanco !

Y lo que usted estará viendo en la pantalla será lo siguiente :

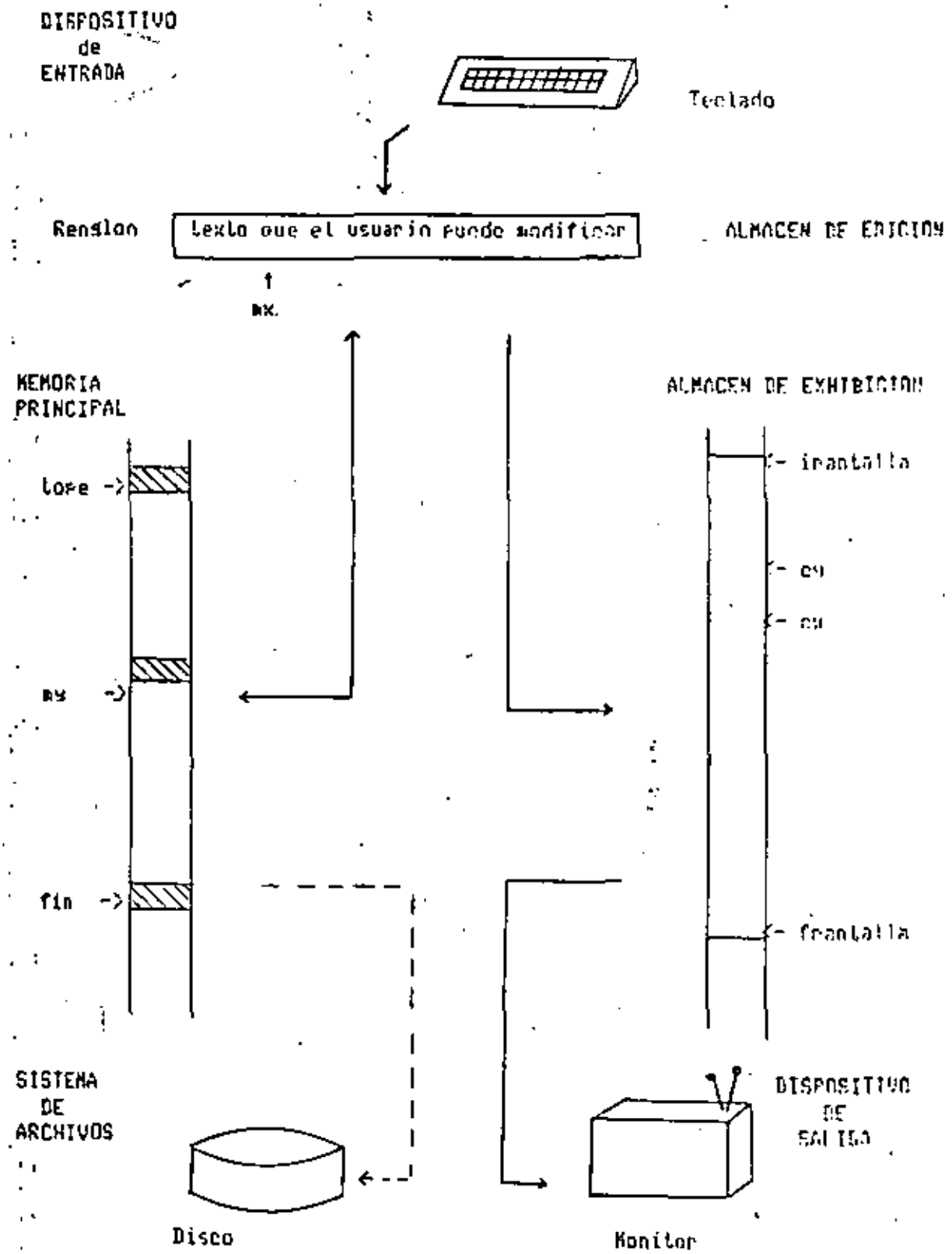
```
La Luna
la luz, a la que, lentamente, enamora mi espiritú
( linea en blanco )   en esta linea deberá estar el cursor
( linea en blanco )
fin
```

MICROPROCESADORES Y MICROCOMPUTADORAS

ARQUITECTURA DE EL EDITOR

SEPTIEMBRE, 1983.

ARQUITECTURA DE EL EDITOR



EDITOR DE PANTALLA

IMPLEMENTACION EN FORTH

```
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
             ilope
             iaw
             ifin
         almacen de edicion
             iren
             iax
        almacen de exhibicion
             ilfin
    MUESTRA la primera pantalla del documento en la pantalla
    LLAMA al procesador de instrucciones
FIN dela rutina
```

```
: edita
  editando.
  trae/crea
  iaen
  lope 14 refrescar
  leditor
;
```

```
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
```



```
:: leditor ** car ldp & ( 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 @ .ldp
      2 c,
      c# f c,
      car         % lee el caracter de  control, hay que prenderle un bit
      40 or       % , el 6 , para convertirlo a caracter imprimible
      c,
      ldp dp !
      context @ @  % vocabulario
      search
      not
      if
        execute
      fi
     else
       car texto
       sicambio
     fi
   fi
  od>
```

EDITAR

(6)

```
RUTINA documento ( caracter )
SI Posicion del cursor = # de columnas
    ENTONCES
        LLAMA asigna caracter % cursor > limite *> no eco
    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

(7)

```
:: texto >> caracter . &
   cx @columnas 1- =
   if
     caracter asasig
     else
       modo
       if
        caracter insercion
      else
        caracter echo
        caracter asasig
      fi
   fi
;
```

ASIGNA CARACTER

```
RUTINA asigna caracter (caracter)
ASIGNA a donde apunta ax el caracter
SI posicion = # de columnas
    ENTONCES
        LLAMA retorno de carro automatico
    SINO
        AVANZA apuntadores ax y cx
FIN de la pregunta
FIN de la rutina
```

IMPLEMENTACION EN FORTH

```
:: ax axasig >> val dir &          % asigna un caracter a el renglon en la
   val dir @ c!
    distancia #columnas >=
   if
       rcautomatico
     else
       ax 1+ ax
   then
```

INSERTA CARACTER


```
RUTINA inserta caracter ( caracter )
    RECORRE los caracteres hacia la 'derecha' de ax
    LLAMA asigna caracter
    MUESTRA el contenido del almacen de edicion
FIN de la rutina
```

IMPLEMENTACION EN FORTH

```
: renglon insercion >> caracter posicion &
   1 agrandarbuffer                % agrandar el renglon
   caracter axasig                 % insertar caracter en  ax avanza ax y cx
   sicambio
   )pantalla                       % enviar renglon a la pantalla
```

MICROPROCESADORES Y MICROCOMPUTADORAS

EJEMPLO DE PROGRAMA EN ENSAMBLADORES

ING. ROLANDO CARRERA SANCHEZ

SEPTIEMBRE, 1983

viernes, septiembre 23, 1983 ARCHIVO FUENTE
SUMA.S


```
; 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               ; BAJO = 0
        LD      C,A             ; ALTO = 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 ALTO
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 BAJO
        INC     HL
        LD      (HL),C          ; GUARDA EL ALTO
        HALT
        END
```

viernes, septiembre 23, 1983
SUM16,L

Archivo Ensamblado


```
OBJ CODE M STMT SOURCE STATEMENT                    ASM 5.5

                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          ;
0000           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          ;
0000           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          ; iniciaeos 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               ; BAJO = 0
0006   4F               53          LD      C,A             ; ALTO = 0
0007   86               54  SUM:    ADD     A,(HL)          ; SUM = SUM + ARGL(HL)
0008   23               55          INC     HL
0009   3001             56          JR      NC,SIGUY        ; VE SI HUBO CARRY DE LA SUMA DE 8 BITS
000B   0C               57          INC     C               ; SUMA EL ACARREO AL ALTO
000C   10F9             58  SIGUY   DJNZ    SUM             ; VE SI TODAVIA HAY DATOS PARA SUMAR.
```

4

SUM16 PAGE 2
ASM 5.8

```
LOC   OBJ CODE  M STMT SOURCE STATEMENT

                        59          ; SI YA TERMINO LA SUMA GUARDALA EN SUMA
000A   210A00   R       60          LD      HL,SUMA
000D   77               61          LD      (HL),A          ; GUARDA EL BAJO
000E   23               62          INC     HL
000F   71               63          LD      (HL),C          ; GUARDA EL ALTO
0010   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 -- 0020 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 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 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
```

```
;
MAXARG: LD      HL,ARGL         ; DIRECCION DEL ARREGLO
        LD      B,LNG           ; TAMANO DEL ARREGLO
        SUB     A               ; MAXIMO = 0
ESMAX:  CP      (HL)            ; ES EL ELEMENTO ARG(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
```

7

Archivo Ensamblado

8

```
                        MAXI                                    PAGE    1
OBJ CODE M STMT SOURCE STATEMENT                                ASM 5.0

                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 GUARDARA EL VALOR MAXIMO
               13       ;
               14       ; OJO estoy usando la directiva o pseudooperacion EQU
               15       ; con la que estoy definiendo el tamano del arreglo o 10 tal
               16       ; ya que OAH = 10 decimal.
               17  ;
0000           18  LNG:    EQU     OAH
               19  ;
               20       ; ahora define 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  ;
0008  210000      R     51  MAXARC: LD      HL,ARCL         ; DIRECCION DEL ARREGLO
000E  0604              52          LD      B,LNG           ; TAMANO DEL ARREGLO
0010  97                53          SUB     A               ; MAXIMO = 0
0011  BE                54  ESMAX:  CP      (HL)            ; ES EL ELEMENTO ARCL(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 = ARCL(HL)
0015  23                57  SIGMAX  INC     HL              ; SIGUIENTE ELEMENTO DEL ARREGLO
0016  10F9              58          DJNZ    ESMAX           ; VE SI TODAVIA HAY DATOS
```

9

MAXI PAGE 2
LOC OBJ CODE M STMT SOURCE STATEMENT ASM 5.8

```
                        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
```

10

```
OCH
viernes, septiembre 23, 1983
MAXI,MAP
LINK 1.6

LOAD MAP
MODULE      ORIGIN LENGTH

.MAXI         5000   001D

GLOBAL    ADDRESS   MODULE

MAXARC        500B   MAXI

PROGRAM MAXI -- 0020 BYTES
ENTRY: 500B
```

MAPA DE MEMORIA
DONDE SE CARGARÁ
EL PROGRAMA OBJETO

FUENTE


```
        ; SE DEFINE UN MACRO PARA HACER LA COMPARACION DE
        ; DOS BLOQUES DE MEMORIA
;
COMPARA MACRO   #BL01,#BL02,#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,#BL01             ;CARGA LOS REGISTROS CON LOS PARAMETROS
        LD      DE,#BL02
        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 0CH
        HALT
        END
```

```
                    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 PARAMETRO
                   14          LD      DE,&BLQ2
                   15          LD      C,&BYTES            ; SE OMITIÓ LA N
*** MACRO DEFINITION ERROR ***                             DE LA ETIQUETA DEL
                   16          CALL    COMPR               TERCER PARAMETRO
                   17          POP     AF                  ≠ NBYTES
                   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&SYM
                   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&SYM 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 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 PARAMETRO
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
```

```
                                COMPARA                                 PAGE  2
 LOC  OBJ CODE M STMT SOURCE STATEMENT                                   ASM 5.8

                                        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    ; 'OAH'=''
                               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  210050                           LD      HL,5000H        ;CARGA LOS REGISTROS CON LOS PARAMETROS
 0022  110060                           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    ; 'OAH'=''
                               FCP0001
*** MULTIPLE DECLARATION ***
 002C  76                       45      HALT
                                47      END
```

* Como el ensamblador sólo toma los primeros 6 caracteres, cree que {FCP000|0} son iguales {FCP000|1}

** y protesta.

Viernes, septiembre 23, 1983 FUENTE SIN ERRORES 14
COMPARA.S

```
        ; SE DEFINE UN MACRO PARA HACER LA COMPARACION DE
        ; DOS BLOQUES DE MEMORIA
;
COMPARA MACRO   #BLO1,#BLO2,#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,#BLO1            ;CARGA LOS REGISTROS CON LOS PARAMETROS
        LD      DE,#BLO2
        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 0FH
        HALT
        END
```

Viernes, septiembre 23, 1983
COMPARA.L

```
                        COMPARA                                 PAGE  1
LOC   OBJ CODE M STMT SOURCE STATEMENT                          ASM 5.8

                        1
                        2  ;
                        3          ; SE DEFINE UN MACRO PARA HACER LA COMPARACION DE
                        4          ; DOS BLOQUES DE MEMORIA
                        5  ;
                        6  COMPARA MACRO  &BLO1,&BLO2,&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,&BLO1            ;CARGA LOS REGISTROS CON LOS PARAMETROS
                       14          LD     DE,&BLO2
                       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  110060                      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
```

```
                        COMPARA                                 PAGE  2
LOC  OBJ CODE M STMT SOURCE STATEMENT                           ASM 5.8

                                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  215050                     LD      HL,5000H         ;CARGA LOS REGISTROS CON LOS PARAMETROS
0024  116060                     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 (#).

```
                        COMPARA                                 PAGE   1
 LOC  OBJ CODE  M STMT SOURCE STATEMENT                          ASM 5.8

                1
                2 ;
                3         ; SE DEFINE UN MACRO PARA HACER LA COMPARACION DE
                4       ; DOS BLOQUES DE MEMORIA
                5 ;
                6 COMPARA MACRO   #BLO1,#BLO2,#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,#BLO1        ;CARGA LOS REGISTROS CON LOS PARAMETRO
               14         LD      DE,#BLO2
               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
                          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 PARAMET
                          LD      DE,6000H
                          LD      C,
                          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      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 PARAME
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    1809                    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              45      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

APENDICE CAPITULO 8-INTERPRETES

ING:DAVID BETANCOURT

SEPTIEMBRE, 1983

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

CARH

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

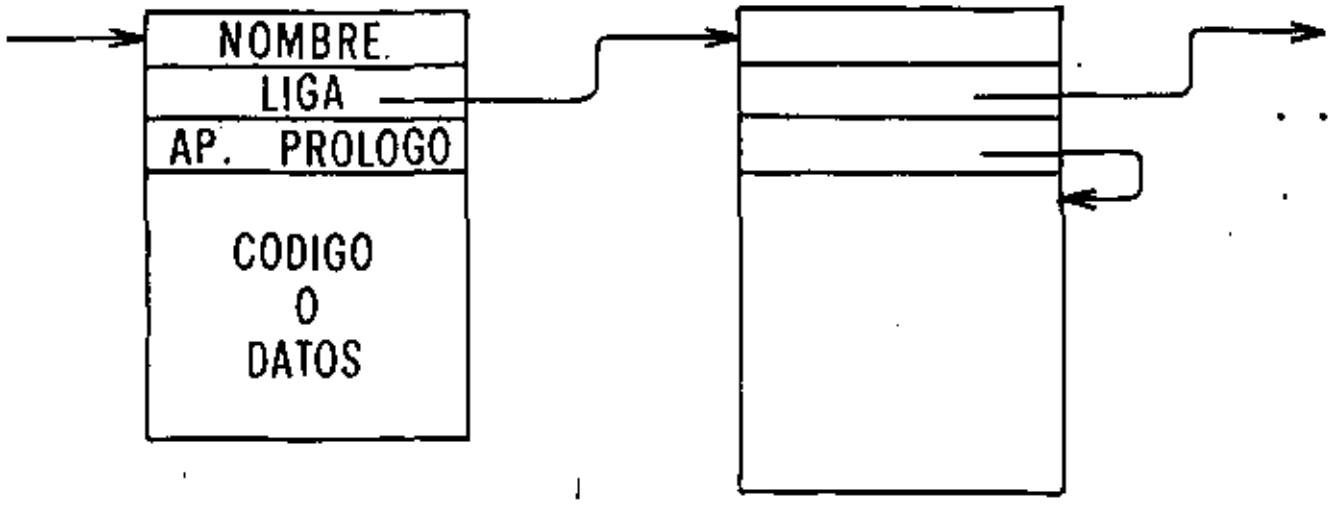
COLECCION DE FUNCIONES QUE PUEDE SEPARARSE Y RECIBIR UN NOMBRE

MEMORIA VIRTUAL

SCREENS

LENGUAJE INTERACTIVO:

DICCIONARIO



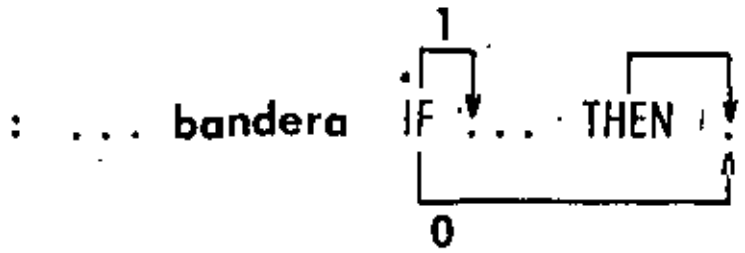
INTERPRETE EXTERNO (INTERFAZ CON EL USUARIO)

- ACEPTA CARACTERES DE LA TERMINAL
- SI RECIBE UN NUMERO LO METE AL STACK
- SI RECIBE UNA FUNCION, LA BUSCA Y LA EJECUTA

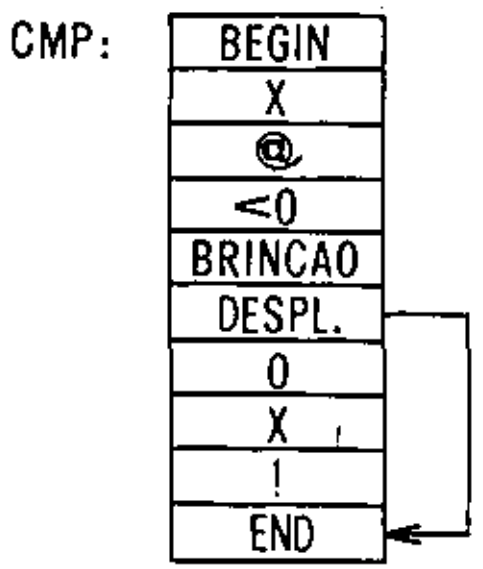
10 20 30 * + PRINT

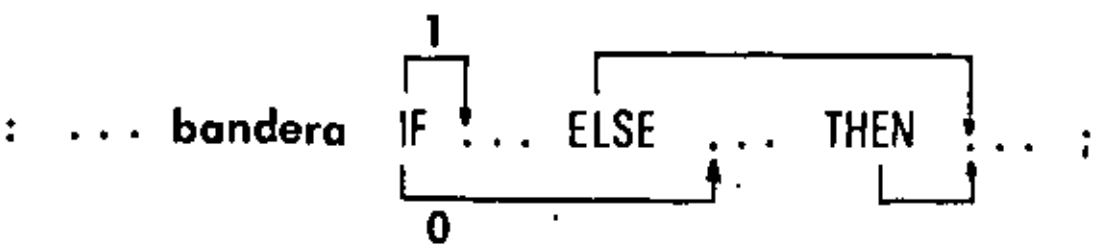
- UN SOLO DELIMITADOR: ESPACIO

- CODIGO ESTRUCTURADO



: CMP X @ <0 IF 0 ! THEN ;





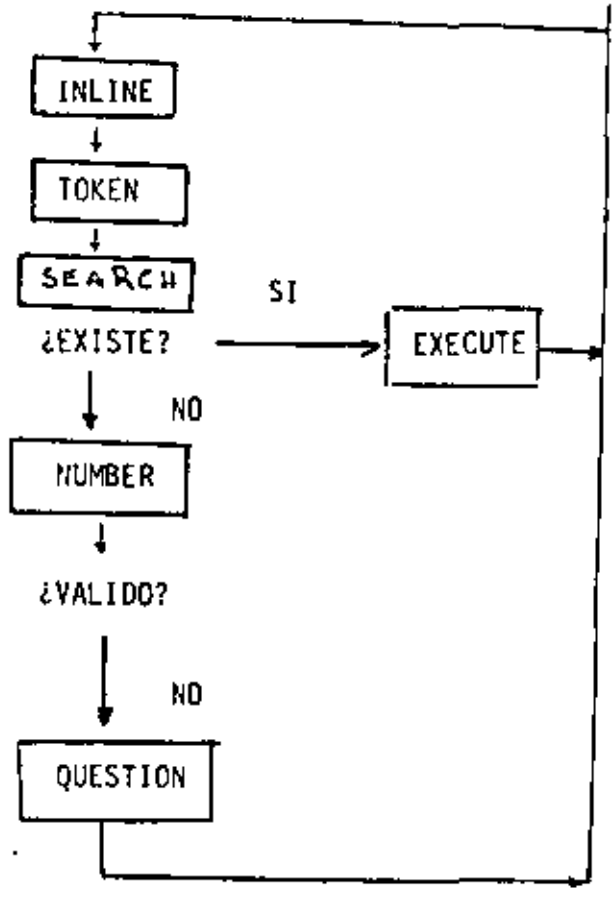
POCO LEGIBLE

PARAMETROS SIN NOMBRE

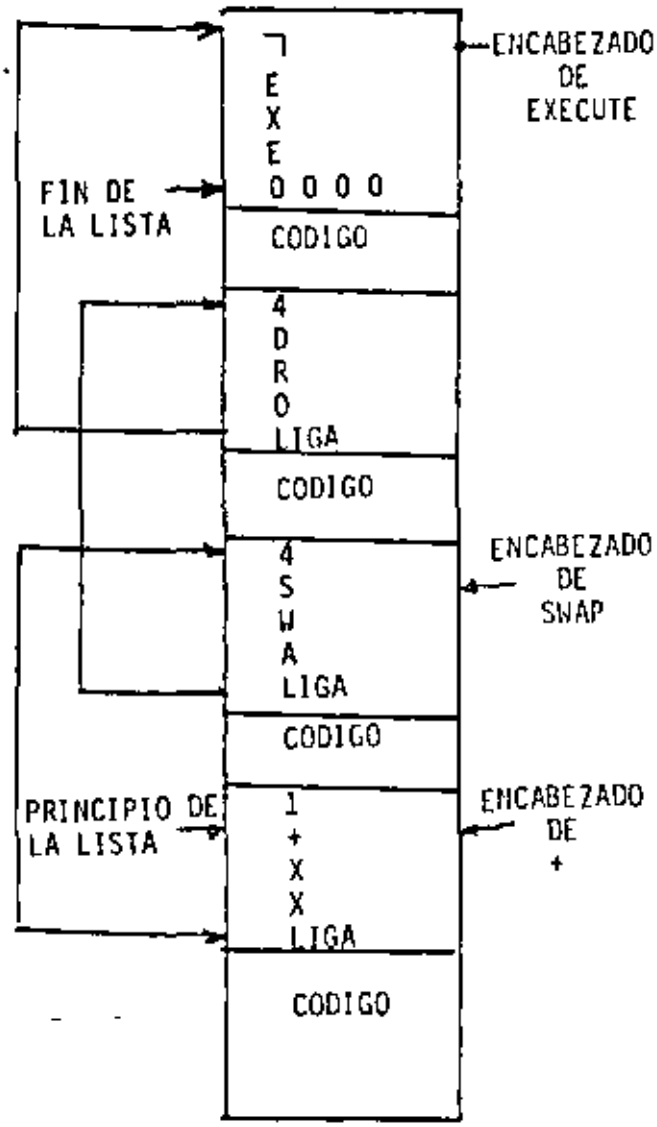
FALTA DE ANALISIS DE TIPOS

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
```

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$ (1)  <= 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
```

DIRECTORIO DE ALUMNOS DEL CURSO DE "MICROPROCESADORES Y MICROCOMPUTADORAS" IMPARTIDO DEL 19 DE SEPTIEMBRE AL 1o. DE OCTUBRE DE 1983.

1.- ALDAMA JAMAICA GUADALUPE ROSALIA
COORDINACION DE LA INVEST. CIENT.
U. N. A. M.
TECNICO ACADEMICO
550-52-15 ext. 4813

AV. MORELOS OTE. No. 172
SAN CRISTOBAL ECATEPEC
55000 EDO. DE MEXICO
787-34-98

2.- ALVARADO SERRANO CARLOS
PHILIPS MEXICANA
DISEÑADOR ELECTRONICO
NORTE 45 No. 669
COL. INDUSTRIAL VALLEJO
DELEGACION AZCAPOTZALCO
569-21-11 ext. 257

AV. 535 No. 260
UNIDAD ARAGON
DELEGACION GUSTAVO A. MADERO
551-00-86

3.- ALVAREZ TINOCO JESUS GABRIEL
INSTITUTO MEXICANO DEL SEGURO SOCIAL
SUPERVISOR DE INSTALACIONES TELECOMUN.
REFORMA No. 496
DELEGACION CUAUHTEMOC
525-68-02

CALLE 1910 No. 403
COL. TICOMAN
DELEGACION GUSTAVO A. MADERO
07330 MEXICO, D.F.
586-90-94

4.- ARRIAGA TINAJERO ALFONSO
INSTITUTO MEXICANO DEL SEGURO SOCIAL
JEFE SECCION CONMUTACION
REFORMA No. 476
COL.JUAREZ
DELEGACION CUAUHTEMOC
511-82-41

CIRCUITO RIO PTE No. 69
PASEOS DE CHURUBUSCO
657-99-44

5.- AVILA MUÑOZ JOSE DE JESUS
SIDERURGICA LAZARO CARDENAS
ING. HARD WARE
DOMICILIO CONOCIDO
APARTADO POSTAL 46-A
SUPERINTENDENCIA DE INST. Y CONTROL

FCO. I. MADERO No. 69
APARTADO POSTAL A-32
LAZARO CARDENAS, MICHOACAN

6.- ARCOS CORIA RODOLFO
DIRECCION GENERAL DE INTERCAMBIO ACAD.

7.- AWAD ABED FOVAD AMINE
UNIVERSIDAD ANAHUAC
CATEDRATICO
LOMAS ANAHUAC
5950 MEXICO, D.F.
589-22-00

FUENTE DE NAYADES No. 22
COL. TECAMACHALCO
EDO. DE MEXICO
53950
589-97-96

8.- BOLAÑOS ESPEJO J. JAVIER
DATA CARD DE MEXICO
INGENIERO SERVICIO DE COMPUTADORAS
AV. CUAUHTEMOC No. 1486-601 A
COL. STA. CRUZ ATOYAC
DELEGACION BENITO JUAREZ
03310 MEXICO, D.F.
534-73-80

U. LEGARIO EDIF. 14
COL. PENSIL
DELEGACION MIGUEL HIDALGO

9.- BUTRON GANDARILLAS FERNANDO
UNIVERSIDAD MAYOR DE SAN ANDRES
JEFE DE TRABAJOS PRACTICOS
LA PAZ, BOLIVIA

GHIPUZCOA No. 37-9
COL. NIÑOS HEROES DE CHAPULTEPEC

10.- CALDERON HERNANDEZ SAUL
SICARTSA
INSTRUMENTISTA Y PROYECTISTA
DOMICILIO CONOCIDO
LAZARO CARDENAS, MICHOACAN
203-33

EDIF. 9 DEPTO. 5
COL. CAMPAMENTO LA MIRA
MICHOACAN

11.- CARREÑO COLORADO ROBERTO
DATA CARD DE MEXICO
SOPORTE TECNICO SERV. COMPUTADORAS
AV. CUAUHTEMOC No. 1486-601 A
COL. STA. CRUZ ATOYAC
DELEGACION BENITO JUAREZ
03310 MEXICO, D.F.
534-73-80

SUR 73 A No. 231-401
COL. SINATEL
DELEGACION IXTAPALAPA
09470 MEXICO, D.F.

12.- CASTELLANOS VAZQUEZ MA. GUADALUPE
PROGRAMA UNIVERSITARIO DE COMPUTO
SUPERVISOR
CIUDAD UNIVERSITARIA
550-52-15 ext. 4135

SALINA CRUZ No. 33
COL. ROMA SUR
DELEGACION CUAUHTEMOC
564-43-41

13.- CRUZ CEBALLOS VICTOR ARMANDO
CENTRO DE CALCULO DE LA F.I.
JEFE DE OPERADORES
FACULTAD DE INGENIERIA
550-52-15 ext. 3765 y 3729

AV. 10 No. 290
COL. PUEBLA
DELEGACION VENUSTIANO CARRANZA
15020 MEXICO, D.F.
763-46-13

14.- DE LA CRUZ GONZALEZ RAFAEL
FACULTAD DE INGENIERIA
PROFESOR DE ASIGNATURA
AV. COPILCO Y UNIVERSIDAD
550-52-15 ext. 3640

AV. 16 DE SEPTIEMBRE No. 45
XOCHIMILCO
16090 XOCHIMILCO

15.- DE LOS REYES IONES MONICA

BOGOTA No. 668
COL. LINDAVISTA
DELEGACION GUSTAVO A. MADERO
07300 MEXICO, D.F.
577-62-80

16.- DIEZ GUERRERO EDUARDO
UNIVERSIDAD POPULAR DEL EDO. PUEBLA
DOCENCIA EN COMPUTACION
PUEBLA, PUE.

48 PONIENTE No. 701
COL. STA. MARIA
72080 MEXICO, D.F.
41-05-29 y 46-46-55

17,- DOMINGUEZ GARCIA CONSUELO

AV. SAN JERONIMO No. 111
COL. SAN ANGEL
DELEGACION ALVARO OBREGON
01000 MEXICO, D.F.

18.- DOWLING PELLEGRIN JUAN ALFONSO

AV. UNIVERSIDAD No. 1953 EDIF. 17-902
COL. COPILCO UNVIERSIDAD
550-66-57

19.- ESPARZA GONZALEZ SERGIO LUIS
INSTITUTO MEXICANO DEL SEGURO SOCIAL
SUPERVISOR DE INSTALACIONES
REFORMA No. 476
COL. JUAREZ
DELEGACION CUAUHTEMOC
533-53-60

3a. CERRADA DE RETOÑO No. 107-20B
COL. EL RETOÑO
DELEGACION IZTACALCO
09440 MEXICO, D.F.
539-02-95

20.- FORCADA GRANADOS JULIO
U. A. M.
JEFE AREA DE INSTRUMENTACION
SAN PABLO No. 480
COL. REYNOSA
DELEGACION AZCAPOTZALCO
02200 MEXICO, D.F.

AV. DEL RIO 6-A DEPTO. 307
COL. VILLA COAPA
DELEGACION TLALPAN
14390 MEXICO, D.F.
594-28-42

21.- GARCIA ROSALES DANIEL JULIAN
PEMEX
TECNICO ESP. ELECTRONICA AERONAUTICA
HANGAR PEMEX PTA. 9 AEROPUERTO INTERNAC.
COL. AVIACION CIVIL
DELEGACION VENUESTIANO CARRANZA
558-29-99

ATZAYACATL No. 16-3
COL. ANAHUAC
DELEGACION MIGUEL HIDALGO
11370 MEXICO, D.F.
535-47-80

22.- GARCIA SOLACO MANUEL ALBERTO
U. N. A. M.
PROFESOR
CIUDAD UNIVERSITARIA
550-52-15

EJE CENTRAL LAZARO CARDENAS No. 169-1
COL. GUERRERO
DELEGACION CUAUHTEMOC
06300 MEXICO, D.F.
526-65-93

23.- GASTELUM OROZCO RAFAEL BALTAZAR
INSTITUTO MEXICANO DEL PETROLEO
INVESTIGADOR
EJE CENTRAL LAZARO CARDENAS No. 152
COL. SAN BARTOLO ATEPEHUACAN
DELEGACION GUSTAVO A. MADERO
567-66-00

PLAN DE GUADALUPE No. 56-1
COL. TICOMAN
DELEGACION GUSTAVO A. MADERO
07330 MEXICO, D.F.
586-59-59

24.- GONZALEZ CANCIÑO JOSE LEON
SIDERURGICA LAZAR CARDENAS LAS TRUCHAS
INGENIERO SOFTWARD
DOMICILIO CONOCIDO
CD. LAZAR CARDENAS, MICHOACAN
APARTADO POSTAL 46-A
203-33 ext. 1367

PRIV. LERDO DE TEJADA No. 59
CD. LAZARO CARDENAS, MICHOACAN

25.- GONZALEZ PULIDO VICTOR JAVIER
INST. TEC. DE EST. SUP. DE MONTERREY
PROFESOR DE PLANTA
CARR. A LAGO DE GUADALUPE KM. 4
ATIZAPAN DE ZARAGOZA, EDO. DE MEXICO
336-00

1er. RETORNO DEL NEVADO No. 20
JARDINES DEL ALBA
CUAUTITLAN IZCALLI, EDO. DE MEXICO
386-13

26.- GONZALEZ ROBLES RAUL
INSTITUTO MEXICANO DEL SEGURO SOCIAL
PASEO DE LA REFORMA No. 476 P.B.
COL. JUAREZ
DELEGACION MIGUEL HIDALGO
511-91-38

CALZ. VIGA No. 1827
COL. UNIDAD MODELO
DELEGACION IXTAPALAPA
09090 MEXICO, D.F.
511-91-38

27.- GUERRERO ZARCO MA. DE LOURDES
PROGRAMA UNIVERSITARIO DE COMPUTO
SUPERVISOR DE PROYECTOS
CIUDAD UNIVERSITARIA
550-52-15 ext. 4135

CALLE 16 No. 109
COL. PANTITLAN
57460 EDO. DE MEXICO
558-83-59

28.- GUTIERREZ ROBLES JORGE
BANCO DEL ATLANTICO
INGENIERO DE TELEPROCESO
BOLIVAR No. 38
COL. CENTRO
DELEGACION CUAUHTEMOC
06000 MEXICO, D.F.
518-75-00 ext. 2396

CANTERA No. 43
COL. VILLA DE GUADALUPE
GUSTAVO A. MADERO
07050 MEXICO, D.F.

29.- HERNANDEZ SONI ARTURO
DATA CARD DE MEXICO
INGENIERO SERVICIO DE COMPUTADORAS
AV. CUAUHTEMOC No. 1486-601 A
COL. STA. CRUZ ATOYAC
DELEGACION BENITO JUAREZ
03310 MEXICO, D.F.
534-73-80

BLVD. AEROPUERTO No. 465-213
COL. MOCTEZUMA
DELEGACION VENUSTIANO CARRANZA

30.- JIMENEZ GARCIA ALEJANDRO
FACULTAD DE INGENIERIA
JEFE DEL CENTRO DE CALCULO
CD. UNIVERSITARIA
550-57-34

IXCATEOPAN No. 62-201
COL. LETRAN VALLE
DELEGACION BENITO JUAREZ
03650 MEXICO, D.F.
575-70-31

31.- KENDI IZHIZU ENRIQUE
DIRECCION GRAL. DE INTERCAMBIO ACAD.

32.- LOPEZ ARCE SIU OSCAR
TELECOPIA, S.A. DE C.V.
INGENIERO DE DESARROLLO
CONSTITUYENTES No. 1054-3
COL. LOMAS ALTAS
DELEGACION MIGUEL HIDALGO
11950 MEXICO, DF.
570-25-44

BOULEVARD CAPRI No. 108-15 B
COL. LOMAS ESTRELLA
DELEGACION IZTAPALAPA
09890 MEXICO, D.F.
515-67-17

33.- LOPEZ SANCHEZ JUAN CARLOS
PROGRAMA UNIVERSITARIO DE COMPUTO
BECARIO
CIRCUITO EXTERIOR C. U.

AV. UNIVERSIDAD No. 1643
COL. AGRICOLA
DELEGACION ALVARO OBREGON
01030 MEXICO, D.F.

34.- LOPEZ TORRES GUSTAVO
CENTRO DE INVESTIG. EST. AVANZADOS
DEL I. P. N.,
AUXILIAR DE INVESTIGACION
AV. I. P. N. Y CALZ. TICOMAN
COL. TICOMAN
754-02-00

BEGONIA No. 28
ECATEPEC DE MORELOS
JARDINES DEL TEPEYAC

35.- LLAMAS DE LA FUENTE FRANCISCO
S. P. P.
COORDINADOR
IZAZAGA No. 29-3er. PISO
COL. CENTRO
761-50-53

COLIMA No. 45 G
COL. ROMA
DELEGACION CUAUHTEMOC
06700 MEXICO, D.F.
514-30-91

36.- MEJIA ALONSO FELIPE
PHILIPS MEXICANA
TECNICO

ZEMPOALA No. 88-3
COL. NARVARTE
03020 MEXICO, D.F.
538-50-19

37.- MELO BAUTISTA JOSE
CENTRO ESCOALR DEL LAGO A.C.
MAESTRO DE EDUCACION FISICA
LAGO DE GUADALUPE
CUAUTITLAN IZCALLI

AV. PATRIOTISMO No. 112-1
COL. ESCANDON
DELEGACION MIGUEL HIDALGO
11800 MEXICO, D.F.
515-81-60

38.- MOSSO SALVADOR DE JESUS

39.- MUÑOZ CARETA ADALBERTO
S. C. T.
PROGRAMADOR
CENTRO S.C.O.P. CUERPO "H"
6o. PISO XOLA Y UNIVERSIDAD
COL.NARVARTE
DELEGACION BENITO JUAREZ
538-66-03

SANTA CRUZ 210-casa 20
COL. CANDELARIA
DELEGACION COYOACAN
04380 MEXICO, D.F.

40.- MUÑOZ VARGAS FRANCISCO
ELECTROTAX, S.A.
INSTRUMENTOS TECNICOS
GERENTE CONTROL DE PROYECTOS
NORTE 92 No. 5213
COL. GERTUDRIS SANCHEZ

MARMOLERIA No. 127
COL. 20 DE NOVIEMBRE
789-85-81

41.- MURILLO MANZANO ENRIQUE

42.- NOHPAL JUAREZ EVODIO
EUROBAL
JEFE DE LABORATORIO
HUERTAS No. 107-1001
COL. DEL VALLE

CAMINANTE No. 94
COL. AURORA OTE
DELEGACION NETZAHUALCOYOTL
EDO. DE MEXICO

43.- ORTEGA GUERRERO MARCOS ADRIAN
FACULTAD DE INGENIERIA
PROFESRO ASIGNATURA A.

GABINO ORTIZ No. 16
COL. CONSTITUCION
DELEGACION GUSTAVO A. MADERO
07460 MEXICO, D.F.
781-20-11

44.- PONCE FLORES MIGUEL ANGEL
MULTIBANCO COMERMEX
EJECUTIVO DE NORMAS Y PROCEDIMIENTO
LORENZO BOTURINI No. 203
COL. TRANSITO
DELEGACION CUAUHTEMOC
06820 MEXICO, D.F.
761-20-11

AV. ANILLO DE CIRCUNVALACION No. 314-13
COL. CENTRO
DELEGACION VENUSTIANO CARRANZA
15100 MEXICO, D.F.
542-14-84

45.- PRIDA MENENDEZ JOSE ANTONIO
CITIBANK, N. A.
SUBGERENTE DE REPORTES DE CONTABILIDAD
PASEO DE LA REFORMA No. 390
COL. JUAREZ
DELEGACIÒN CUAUHTEMOC
06600 MEXICO, D.F.
553-07-00

GRAL. SALVADOR ALVARADO No. 186-1
COL. CONDESA
DELEGACION CUAUHTEMOC
06170 MEXICO, D.F.
547-42-86

46.- RODRIGUEZ ORTEGON SALVADOR

COSCOMATE No. 47
COL. TORIELLO GUERRA
DELEGACION TLALPAN
14050 MEXICO, D.F.
573-23-13

47.- RODRIGUEZ SANCHEZ BERNABE
TELMEX
INGENIERO DE PROYECTOS
SULLIVAN No. 199 of. 301
COL. SAN RAFAEL
DELEGACION CUAUHTEMOC

NORTE 83-A No. 525-1
COL. ELECTRICISTAS
DELEGACION ATZCAPOTZALCO
561-70-80

48.- RUIZ BETARANO JAIME
INSTITUTO MEXICANO DEL PETROLEO
ING. ESPECIALIZADO

CALLE PONIENTE No. 21
COL. NUEVA VALLEJO

49.- TIRADO RAMIREX JOSE DE JESUS
INSTITUTO MEXICANO DEL PETROLEO
PROFESIONAL "A"
EJE LAZARO CARDENAS No. 152
COL. SAN BARTOLO ATEPEHUACAN
DELEGACION GUSTAVO A. MADERO
567-66-00

AV. 525 No. 25
COL. UNIDAD SAN JUAN ARAGON
DELEGACION GUSTAVO A. MADERO
07920 MEXICO, D.F.
551-79-85

50.- TLAHUIZO GARCIA RICARDO
UNIVERSIDAD POPULAR AUTONOMA EDO. PUEBLA
DIRECCION DE CIENCIAS ECONOMICAS
PUEBLA, PUE.
46-57-22

13 NORTE No. 3
PUEBLA, PUE.

51.- TORRES MUÑOZ ALBERTO
COMISION DE VIALIDAD Y TRANSPORTE URBANO
INGENIERO DE PROYECTO Y ESTUDIOS
AV. UNIVERSIDAD No. 800
COL. STA. CRUZ ATOYAC
DELEGACION BENITO JUAREZ

BOLONCHEN MANZ 278 LOTE 5
COL. EJIDOS DE PADIERNA
DELEGACION TLALPAN
14260 MEXICO, D.F.
652-12-62

52.- SILVA MARVAL JUAN RAMON
FUERZA Y CLIMA, S.A.
SUPERVISOR
CALZ. MEXICO TACUBA No. 392
COL. POPOTLA
396-19-61 y 396-39-81

AV. SAN MARGARITA No. 218
COL. UNIDAD TUZANIA
GUADALAJARA, JAL. 45130

53.- VARGAS SOTO SALOMON
SISTEMA NACIONAL PARA EL DESARROLLO DE LA C.
MAESTRO DE ACTIVIDADES ESPECIALES
EMILIANO ZAPATA No. 340
COL. PORTALES
DELEGACION BENITO JUAREZ

C. ORIENTE 217 A No. 14
COL. AGRICOLA ORIENTAL
DELEGACION IZTACALCO

54.- ZANOTTO DAVINI HECTOR
UNIVERSIDAD AUTONOMA DE PUEBLA
PROFESOR TIEMPO COMPLETO
CIUDAD UNIVERSITARIA
PUEBLA, PUE.

18 SUR No. 5718
COL. SAN MANUEL
PUEBLA, PUE.

55.- WONG ORTEGA MARIO ALBERTO
TELEVISA, S.A.
INGENIERO EN DISEÑO
AV. CHAPULTEPEC No. 18
COL. DOCTORES
DELEGACION CUAUHTEMOC
06724 MEXICO, D.F.
585-33-33 ext. 1117

HAMBURGO No. 316-103
COL. JUAREZ
DELEGACION CUAUHTEMOC
06600 MEXICO, D.F.
286-17-73