DIRECTORIO DE PROFESORES DEL CURSO: INTRODUCCION A LOS MICROPROCESADORES del 3 al 31 de agosto de 1984.

1. ING. LUIS CORDERO BORBOA (COORDINADOR)
Jefe del Departamento de Computación
División de Ingeniería Mecánica y Eléctrica
Facultad de Ingeniería
UNAM
México, D.F.
550 52 15 Ext. 3750 ó 3746

2. SR. CARLOS LOPEZ SANCHEZ
Ayudante de Profesor
División de Ingeniería Mecánica y Eléctrica
Departamento de Computación
Facultad de Ingeniería
UNAM
México, D.F.
550 52 15 Ext. 3750

3. ING. DANIEL RIOS ZERTUCHE
Director de Informática
Subsecretaría de Planeación
del Desarrollo S.P.P.
Izazága No. 38-11° Piso
México, D.F.
521 98 98

4. ING. JOSE L. RODRIGUEZ ARAUZ
Coordinador de la Evaluación de
Sistemas de Cómputo
Departamento de Soporte Técnico
PEMEX
Marina Nal. 329 Edificio "C"
México, D.F.
2502623 y 250 26 11 Ext. 3175 ó 2512

5. SR. FRANCISCO VERDUZCO MARTINEZ
Ayudante de Profesor
División de Ingeniería Mecánica y Eléctrica
Departamento de Computación
Facultad de Ingeniería
UNAM
México, D.F.
550 52 15 Ext. 3750

6. ING. JUAN B. MARTINEZ GARCIA
Técnico Académico
Instituto de Ingeniería
Sección de Automatización
UNAM
México, D.F.
550 52 15 Ext. 3632

INTRODUCCION A LOS MICROPROCESADORES

FECHA	HORARIO	T E M A S	PROFESORES
3 Agosto	17:-21:00	Conceptos Básicos	Luis G. Cordero B.
4 Agosto	9:-14:00	Introducción a los microprocesadores	Luis G. Cordero B.
		Arquitectura y operación de un microprocesador	Luis G. Cordero B.
10 Agosto	17:-21:00	Programación del microprocesador	Luis G. Cordero B.
11 Agosto	9:-14:00	Práctica	Luis G. Cordero B. Francisco Verduzco M.
17 Agosto	17:-21:00	Interfaces de entrada y salida	Daniel Ríos Zertuche
18 Agosto	9:-14:00	Práctica	José L. Rodríguez A. Francisco Verduzco M. Juan C. López S.
24 Agosto	17:-21:00	Aplicaciones	Juan B. Martínez G.
25 Agosto	9:-14:00	Práctica	José L. Rodriguez Juan C. López S.
31 Agosto	17:-21:00	Estado actual de los microprocesadores	Juan B. Martínez G.

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

INTRODUCCION A LOS MICROPROCESADORES

PRACTICAS DE MICROCOMPUTADORAS

JUAN CARLOS LOPEZ SANCHEZ
FRANCISCO VERDUZCO MARTINEZ

AGOSTO, 1984.

UNIVERSIDAD NACIONAL AUTONOMA DE MEXICO

FACULTAD DE INGENIERIA

DIVISION DE EDUCACION CONTINUA

CURSO: INTRODUCCION A LOS MICROPROCESADORES

PRACTICAS DE MICROCOMPUTADORAS

ELABORARON:

JUAN CARLOS LOPEZ SANCHEZ.

FRANCISCO VERDUZCO MARTINEZ.

AGOSTO 1984.

PRACTICA # 1

Conocimiento y manejo del Z-80 starter kit.

OBJETIVO.

El objetivo de esta practica es que el alumno conozca cuales son los elementos de software y hardware que componen al Z-80 Starter Kit.

DESCRIPCION DEL SOFTWARE.

1.1 INTRODUCCION.

EL Z-80 Starter Kit fue diseñado para que se utilice en 5 campos principalmente. Estos son experimentos electronicos, operadores de radio amateurs, hobbies de computacion, control y evaluacion en la industria, ademas de servir para la enseñanza de los primeros conocimientos sobre microcomputadoras.

Tanto un instructor como un alumno que este tomando un curso de microcomputadoras, puede usar el Z-80 starter kit como un medio de un costo relativamente bajo, para adquirir experiencia en el manejo de microcomputadoras.

Los componentes del Z-80 forman parte de la tercera generacion cuyo diseño se baso en el CI de Intel 8080. Las instrucciones del Z-80 son las mismas que las del 8080A mas un conjunto de 80 instrucciones, las cuales nos dan un total de 158 instrucciones. Maneja direccionamiento relativo, dos registros indexados de 16 bits, direccionamientos de bit, 22 registros del CPU, capacidad para operaciones aritmeticas de 16 bit, operaciones en bloque, refresco de memoria dinamico, y codigos de operaciones de 16 bits los cuales lo hacen mas flexible y mas poderoso que cualquier otra maquina de 8 bits como: 8080A/8085, 6502, o el 6800/6802.

1.2 BOTON DE INICIO (RESET).

El boton de Inicio nos sirve para inicializar al CPU y hacer que se empieze a ejecutar el programa que esta en la direccion 0000H. Puesto que el monitor ZBUG se encuentra almacenado en los dos primeros Kbytes de memoria, al apretar este boton se transfiere el control del programa al ZBUG. Con esto, el ZBUG inicializa el Stack Pointer del usuario con la direccion 23C0H, inicializa las variables a cero, y si el switch S3 esta colocado en la posicion (MONITOR RESTART), el ZBUG coloca el prompt "-" en el despliege de mas a la izquierda . Si el interruptor S3 esta en la posicion (PROM1 RESTART) el monitor ZBUG automaticamente inicializa la ejecucion del programa en la localidad 0800h de la PROM1. De esta manera, es posible que el Z80 inicie el programa almacenado en la PROM1.

1.3 MONITOR (MON).

El proposito de esta tecla es suspender la ejecucion del programa que se esta llevando a cabo en ese momento, y regresar el control a la porcion del ZBUG que se encarga de esperar una nueva entrada o comando. Esta tecla preserva el contenido de los registros del CPU y los valores de las variables de la memoria RAM del ZBUG. Los dos principales usos en los que se emplea esta tecla son los siguientes: Es usado para terminar un dato o comando, de tal manera que si nos equivocamos al proporcionar un dato o comando, esta tecla nos permite volver a introducirlo correctamente. Tambien es usado para "abortar" un programa que se esta ejecutando en un momento determinado y que no se alteren los registros, de tal manera que podamos examinarlos para detectar posibles fallas en el programa. Cuando se utiliza esta tecla, se despliega el prompt "-" en el despliege de mas a la izquierda del Kit.

1.4 EXAMINADOR DE MEMORIA (MEM EXAM).

Esta tecla nos sirve para examinar y cambiar el contenido de las localidades de memoria. Para usar esta tecla, primero se teclea la direccion (4 digitos hexadecimales) cuyo contenido queremos observar; una vez proporcionada la direccion, la cual aparecera en los cuatro despliegues de la izquierda, se presiona la tecla MEM EXAM, luego de tal aparecera el contenido de la direccion en los despliegues de mas a la derecha. Si solo hemos tecleado tres, el monitor ZBUG esperara a que se tecleen los digitos que falten para poder desplegar el contenido de tal localidad. Presionando la tecla de NEXT causaremos que se incremente en uno la direccion que aparece en el despliegue, pudiendo de esta manera examinar el contenido de la siguiente localidad sin tener que proporcionar su direccion. Para cambiar el contenido de una localidad, basta con teclear los numeros, en hexadecimal, que queremos que tenga la localidad que aparece en el despliegue, para que el monitor ZBUG cambie el contenido de tal localidad.

1.5 EXAMINADOR DE PUERTOS (PORT EXAM).

La tecla PORT EXAM nos sirve para observar y cambiar el contenido de las localidades de un puerto, habiendo 256 puertos los que puede soportar el Z-80. Para usar esta tecla, primero se proporciona la direccion del puerto que se quiere examinar o modificar, mediante dos digitos hexadecimales. Una vez hecho esto se presiona la tecla PORT EXAM, apareciendo el contenido de tal puerto en los despliegues de mas a la derecha. Si solo se proporciona un digito de la direccion y luego se presiona la tecla PORT EXAM, el monitor ZBUG esperara a que se proporcione el segundo numero para desplegar el contenido de tal puerto. Presionando la tecla NEXT podemos observar el contenido del siguiente pueto, de manera similar a la de MEM EXAM. Para modificar el contenido de tal puerto basta con teclear dos digitos (en hexadecimal) para que el monitor ZBUG modifique el contenido de tal puerto en forma automatica.

1.6 EXAMINADOR DE REGISTROS (REG EXAM).

Esta tecla tiene un uso similar a la de las dos anteriores, pero esta sirve para examinar y modificar el contenido de los registros: A, B, C, D, E, F, H, L, I, PC, IX, IY. El apuntador de la pila (stack pointer) puede ser examinado con esta tecla, pero no se puede modificar su valor. Para examinar el contenido del registro deseado, se teclea el registro a ver y luego se presiona la tecla REG EXAM. Los valores de los registros se toman de una area de la memoria RAM denominada "Mapa de los Registros del Usuario". Para modificar el valor de los registros, basta con teclear el nuevo contenido que se quiere. Cuando se manda a ejecutar el programa, los registros son inicializados con los valores del "Mapa de Registros del Usuario", con lo que con esta tecla podemos modificar tales contenidos antes de la ejecucion del programa.

1.7 EXAMEN DE LOS REGISTROS ALTERNOS (REG EXAM').

Su uso es el mismo que el de la tecla REG EXAM, pero en este caso se utiliza la tecla REG EXAM' y se pueden examinar y modificar los registros: A' B', C', D', E', F', H', L'.

1.8 PUNTOS DE RUPTURA (BREAKPOINTS).

El monitor ZBUG tiene la capacidad de usar hasta 5 puntos de ruptura. El metodo que se utiliza para poder emplear los puntos de ruptura es cambiar el codigo de operacion del usuario con una instruccion RST8 (CFH) y almacenar el codigo de operacion del usuario en una tabla de direcciones de puntos de ruptura y codigos de operacion (BPTAB). La forma en que operan los puntos de ruptura es la siguiente: cuando se esta ejecutando el programa y se encuentra un punto de ruptura, el control es transferido al ZBUG a travez de la instruccion RST8, siendo salvados todos los registros del CPU en el "Mapa de Registros del Usuario", para que puedan ser examinados y modificados posteriormente. Despues de esto, todas las instrucciones RST8 son reemplazadas por el codigo de de la instruccion del usuario, que estaba almacenado en la BPTAB. Para introducir los puntos de ruptura en un programa se procede de la siguiente manera: se proporciona la direccion en la cual queremos que ocurra el punto de ruptura y luego presionamos la tecla de BREAKPOINT. El despliegue mostrara la direccion suministrada despues de que se tecleo un punto de ruptura para indicar que este ha sido aceptado. Para suministrar otro punto de ruptura se presiona la tecla MON para obtener el prompt "-", y se procede de la forma anteriormente descrita. Se pueden poner hasta cinco puntos de ruptura. Cuando no aparece la direccion despues despues de presionar la tecla de BREAKPOINT, el ZBUG indica que ya no se pueden poner mas puntos de ruptura. Para eliminar los breakpionts lo podemos hacer de dos maneeras: Presionar la tecla de BRAKPOINT sin suministrar ninguna direccion cuando se tiene el prompt "-"; o usar la tecla SINGLE STEP que cancelara y removera todos los puntos de ruptura existentes; o presionar el boton de RESET.

1.9 PASO A PASO (SINGLE STEP).

La tecla SINGLE STEP nos permite ejecutar el programa instruccion por instruccion. Asi, podemos examinar el contenido de los registros, los puertos, etc. Esta tecla puede ser usada en programas almacenados en la memoria RAM, ROM o EPROM dado que no se requiere una modificacion del codigo de operacion del usuario. Despues de que se detuvo la ejecucion del programa, si presionamos de nuevo la tecla de SINGLE STEP se hace que se vuelvan a cargar los registros del CPU y se ejecuta la siguiente instruccion. Presionando repetidamente la tecla de SINGLE STEP, el usuario puede ejecutar el programa paso por paso, pudiendo ver y modificar el contenido de los registros, etc.

1.10 EJECUTAR (EXEC).

La tecla de EXEC permite al usuario ejecutar programas almacenados ya sea en la RAM, ROM o en la EPROM. Se puede ejecutar un programa de dos maneras: Presionando la tecla de EXEC, se ejecuta el programa a partir de la direccion que tenga el contador del programa (esta direccion esta salvada en el mapa de registros del usuario). Otra forma es proporcionar la direccion a partir de la cual deseamos que se ejecute el programa, y luego presionar la tecla de EXEC. Una vez que el programa esta corriendo, solo se puede detener mediante un punto de ruptura o mediante la tecla de MON.

1.11 TRANSFERENCIA A CASSETE (CASS DUMP).

Por medio de esta tecla salvamos los programas volatiles que se encuentran almacenados en RAM, en cassetes normales usando la tecnica de grabacion de la ciudad de Kansas. Este modo puede ser usado con la mayoria de las grabadoras existentes en el mercado. Para proceder a grabar programas hay que conectar la entrada AUX del Kit a la entrada "AUX" o "MIC" de la grabadora; despues se procede de la siguiente manera:

1) Colocar el cassete en la grabador y regrezar la cinta.

2) Suministrar la direccion a partir de la cual se va a grabar en las localidades 23C0H y 23C1H (el byte mas significativo en la 23C0H, y el byte menos significativo en la localidad 23C1H).

3) Asegurarse que el prompt "-" aparece. Presionar la tecla de CASS DUMP y poner la grabadora en modo de grabacion. El prompt desaparecera.

4) No se necesita hacer ningun ajuste de volumen en la grabadora. Una vez que se llevo a cabo la grabacion reaparecera el prompt. Al menos 30 segundos se requieren para llevar a cabo la grabacion.

1.12 LECTURA DE CASSETE (CASS LOAD).

Para cargar programas almacenados en cassete se debe proceder asi:

1) Conectar la grabadora de la salida marcada con "EAR" en el Kit a la salida "MONITOR OUT" o "EARPHONE" de la grabadora.

2) Colocar el control de la grabadora al maximo de agudos y minimo de graves.

3) Colocar el control de volumen al minimo.

4) Asegurarse de que aparece el prompt; presionar la tecla CASS LOAD, y el prompt desaparecera.

5) Incrementar el volumen hasta que se encienda el LED de carga , e incrementar un 20% mas el volumen. El LED permanecera encendido durante la lectura.

6) Si se llevo a cabo la lectura en forma correcta aparecera el prompt "-" una vez que se haya llevado a cabo la lectura.

7) Si ocurre un error, aparecera en el despliegue el proximo bloque de datos, habiendose leido correctamente los bloques anteriores. Si ocurre la falla verifique el volumen y el control de tono.

1.13 PROGRAMADOR DE EPROMS.

El programador mueve datos de memoria RAM de la localidad 2000H al EPROM 2716/2758 que se coloca en el base PROM2 (1000H). Se requiere una fuente de 25+- 1 volt capaz de suministrar 30ma. Se inserta el EPROM con el sistema apagado; se carga el programa en RAM. Ahora desde el monitor se teclea el numero de bytes (4 digitos) que se desean subir al EPROM; luego ponemos el interruptor S2 en la posicion PGM y presionamos la tecla PROM PROG. El despliegue se apagara y una vez terminada la grabacion el ZBUG puede responder de dos maneras: La primera es la aparicion del prompt "-" lo cual indica que la programacion se realizo satisfactoriamente. La segunda indicacion es la aparicion de una direccion de 4 digitos, que es la direccion a partir de la cual los datos no se grabaron correctamente. Si presionamos la tecla de NEXT, el ZBUG continua checando la EPROM con la RAM y desplegara la proxima direccion en la EPROM donde el dato no se grabo. Este error nos indica que estamos grabando sobre una EPROM que no ha sido borrada, o una EPROM que tiene fallas. Regrese el interruptor S2 a la posicion de READ despues de que termine la programacion.

1.14 TECLA NEXT.

El efecto de la tecla de Next en el modo de examen de memoria ocasiona que pasemos a la siguiente localidad. Al examinar un puerto pasara al siguiente puerto, etc.

1.15 CALCULOS DE SALTO RELATIVO.

Hay una rutina para el calculo de saltos relativos. Para utilizarla hay que poner en el registro HL la direccion destino, y en DE la direccion de la instruccion relativa. Luego hay que correr la rutina que se encuentra en la direccion C0H. Esta rutina pone el despalzamiento en la localidad que le corresponde. Si las direcciones dadas no son correctas, en el display apareceran numeros fuera del rango 00 y FF.

1.16 INSTRUCCIONES RST.

El conjunto de instrucciones del Z-80 tiene 8 instrucciones de reinicio direccionadas y una direccion para el vector NMI. La direccion del boton de inicio por hardware (0000H) es usada como punto de entrada al ZBUG. Cuando encendemos el Kit, automaticamente se empieza a ejecutar el ZBUG. La direccion del vector NMI (0066H) es utilizada por la tecla de MON para abortar ciertas funciones, y para tener un control del Z-80 por medio del teclado. La direccion de reinicio (0000H) se usa para poder manejar los puntos de ruptura y por lo tanto no puede ser usado. Las otras seis instrucciones de reinicio (16-56) son para uso del programador. Estas instrucciones de un Byte, cuando se ejecutan, salvan el contador del programa en la pila (Stack) y direccionan las localidades 0010H, 0018H, 0020H, 0028H, 0030H y 0038H dependiendo de la instruccion utilizada. Desde que estas localidades estan en el EPROM donde esta el ZBUG, se han puesto brincos a localidades especificas en RAM donde el usuario puede poner otro brinco a alguna localidad que el desee. La siguiente tabla es un resumen del mapeo en RAM de las instrucciones de reinicio libres:

Instruccion	Codigo de operacion	Direccion enla EPROM del ZBUG	Direccion en RAM
RST 00	C7H	0000H	-----
RST 08	CFH	0008H	-----
RST 16	D7H	0010H	23C4H
RST 24	DFH	0018H	23C7H
RST 32	E7H	0020H	23CAH
RST 40	EFH	0028H	23CDH
RST 48	F7H	0030H	2300H
RST 56	FFH	0038H	23D3H

Por ejemplo , si un RST 32 es usado en un programa, el Z-80 CPU salva el contador del programa y va a la direccin 0020H. En esta direccion se puso una instruccion de brinco a la localidad 23CAH.

1.17 INTERRUPCIONES AL CANAL CERO DEL CTC.

Dentro de un programa el usuario puede poner los vectores de interrupciones del CTC y PIO como el quiera. Sin embargo si desea utilizar el canal cero del CTC mientras el ZBUG monitor esta usando los otros tres (Canal uno es usado para cuando grabamos un programa en un casseto, el canal dos se utiliza para dar el tiempo cuando utilizamos el programador, y el canal 3 se utiliza para dar los tiempos cuando leemos un programa grabado en casseto) se tiene que tener en cuenta varia cosas. Ya que los cuatro canales del CTC estan relacionados entre si, en el ZBUG se previno esto y se mapeo la interrupcion del canal cero en RAM y donde un brinco a la actual rutina de servicio se puede poner. Cuando el ZBUG lo pone disponible, el canal cero del CTC primeramente ve la tabla que esta en la direccion 07F8H y donde se puso la direccion 23D6H. 23D6H es la direccion de la primera instruccion que sera ejecutada despues de que el canal cero se interrumpa.

1.18 MAPA DE MEMORIA- USO DE LA RAM.

El mapa de memoria del Starterd Kit es el siguiente:

0000H-07FFH	ZBUG MONITOR
0800H-0FFFH	PROM1
1000H-17FFH	PROM2 PROGRAMADOR
1800H-1FFFH	NO USADA
2000H-238FH	RAM DEL USUARIO
2390H-23A8H	PILA DEL ZBUG
23A9H-23C0H	MAPA DE REGISTROS
23C1H-23FFH	TABLAS
2400H-27FFH	1K DE RAM OPCIONAL
2800H-FFFFH	NO USADA

1.19 RUTINAS DEL ZBUG.

Las rutinas del ZBUG disponibles para el usuario son 5:

1) UIX3 direccion de llamado 0634H. Esta rutina incrementa IX en tres y decrementa el registro B. Los registros afectados son IX, B, F.

2) UFOR1- Direccion de llamado 063CH. IX apunta a dos localidades en memoria y A contiene dos digitos hexadecimales que seran escritos en la memoria (el digito representado en los bits 4-8 se escribe en (IX), y el representado en los bits 0-3 en (IX+1)). Los registros afectados son A, B, F.

3) D20MS- Direccion de llamado 064FH. Esta rutina es un retardo de 20 mseg antes de encontrar el retorno de la rutina. Los registros afectados son H, L, y F.

4) UABIN - Direccion de llamado 0683H. Convierte un caracter ASCII a su equivalente en binario. Los registros usados son A y F. El caracter ASCII esta en A antes de la llamada, y el resultado queda en A despues de la llamada.

5) UBASC - Direccion de llamado 06B8H. Convierte un caracter binario en el acumulador a su caracter ASCII correspondiente. El resultado queda en el acumulador. Los registros usados son A y F.

DESCRIPCION DEL HARDWARE.

2.1 GENERAL.

La figura 1 es el diagrama de bloques del Z-80 STARTER KIT. Nos referiremos a este diagrama y los diagramas de las figuras 2 y 3 durante la siguiente discusion.

2.2 CIRCUITERIA DE RELOJ.

El reloj del kit trabaja a una frecuencia de 2 MHz. teniendo un periodo de 500 nseg. Esto se hizo asi para asegurar una maxima compatibilidad con el 8080. La fuente del reloj es un cristal oscilador basado en un 74LS04 y que corre a 3.9936 MHz. Se eligio esta frecuencia ya que es multiplo de 1200/2400 Hz y las frecuencias de 300 bauds requeridas para poder grabar informacion con el formato de la ciudad de Kansas. Esta frecuencia se divide en dos para obtener un reloj de 1.9968 MHz. Una compuerta 74LS04 con una resistencia de PULL-UP de 330 ohm es usada para manejar las entradas del reloj de los tres dipositivos que tiene este kit.

2.3 Z80-CPU.

El Z-80 CPU es el cerebro del Z-80 Starter kit y provee la mayoria de las senales de control para hacer un barrido tanto sobre los displays como en el teclado, o bien cuando se esta escribiendo o leyendo en la memoria. El CPU genera 16 bits para el bus de direcciones, 8 bits para el bus bidireccional de los datos, y 8 senales de control.

2.4 Z-80 PIO.

El Z-80 PIO es una interfase en paralelo disenada para la familia Z-80. Este soporta una gran cantidad de software para el manejo de interrupiones con un dos conjuntos de lineas de control y un controlador integral de interrupciones. Dos puertos de 8 bits mas las lineas de control, estan disponibles en el PIO para poder hacer la interface en paralelo con los dispositivos.

2.5 Z-80 CTC.

El Z-80-CTC es un dispositivo que contiene cuatro contadores independientes de 16 bits que pueden ser usados para dividir la frecuencia del reloj, o para contar eventos. El ZBUG monitor usa el CTC para implementar varias de las funciones del Starter Kit. El canal cero del CTC no es usado por el ZBUG y esta disponible para el usuario. Los demas canales del CTC son utilizados por el ZBUG asi:

CTC	Canal 1	Audio Cassette during Dump.
CTC	Canal 2	Single Step and EPROM Programmer.
CTC	Canal 3	Audio Cassette during Load.

Cuando alguno de los contadores llega a cero, un pulso se produce en la linea de salida que cuenta ceros (ZC0). Una entrada a cada canal es la de CLOCK/TRIGGER (C/T0) que puede ser utilizada para iniciar el conteo de tiempo o como un reloj externo.

2.6 TECLADO Y DESPLEGADO (DISPLAY).

Los desplegados hexadecimales son barridos por el Z-80 CPU bajo el control del ZBUG. Los datos se mandan a los desplegados a travez del puerto que tiene la direccion 88H, mientras que el desplegado activo se selecciona por el puerto de direccion 8CH. La informacion en cada desplegado permanece aproximadamente durante 1 mseg. Mientras el puerto de direccion 8CH barre los desplegados, tambiern barre el teclado. El puerto que tiene la direccion 90H es la entrada del teclado al bus de datos. Como el puerto de direccion 8CH barre el teclado, entonces si pulsamos una tecla (puerto 90H) podemos determinar que tecla fue, viendo el contenido de cada uno de los puertos. El interruptor MONITOR/PROM1 RESTART es leido por el puerto 90H cuando se da un inicializacion para determinar si el programa que se ejecutara es el ZBUG o el que se encuentre en la PROM1.

2.7 INTERFACE A CASSETTE.

La interface para grabar informacion en un cassette usa el formato de grabacion de la ciudad de Kansas. El CTC genera las frecuencias requeridas para la grabacion, y el Z80 simula el UART con software del ZBUG.

2.8 PROGRAMADOR DE EPROM.

Por medio del Kit podemos grabar memorias 2716/2758 que trabajen a 5 volts, las direcciones y los datos correctos se aplican a la EPROM, la pata de Vpp se pone a +25 VDC, CS (negado)=1, y PD/PGM pulsa en un estado alto durante 50-55 mseg. Esto se repite para cada direccion que sera programada. El software usa la instruccion LDI para mover los datos que se encuntran en RAM a la EPROM (direccion 1000-17FFH). El CTC mantiene la linea de WAIT (negada) activa para mantener los datos y las direcciones validas durante la grabacion.

2.9 DECODIFICACON DE MEMORIA.

La decodificacion de la memoria se lleva a cabo por medio del CI 74LS138. Este CI es un decodificador de 1 a 8. Cada una de las salidas de este CI se usa para habilitar 2k de memoria, de esta manera podemos selecionar los 16 Kbytes mas bajos de la memoria. La siguiente figura nos muestra esto de una manera mas clara.

Patas del bus de direccion						No. de salida decodificador		
15	14	13	12	11	10- 0		Direcciones	Conecxion
0	0	0	0	0	* *	CS0	0000H-07FFH	ZBUG
0	0	0	0	1	* *	CS1	0800H-0FFFH	PROM1
0	0	0	1	0	* *	CS2	1000H-0FFFH	PPG-PROM2
0	0	0	1	1	* *	CS3	1800H-0FFFH	UNUSED
0	0	1	0	0	* *	CS4	2000H-0FFFH	RAM
0	0	1	0	1	* *	CS5	2800H-0FFFH	UNUSED
0	0	1	1	0	* *	CS6	3000H-0FFFH	UNUSED
0	0	1	1	1	* *	CS7	3800H-0FFFH	UNUSED

2.10 DECODIFICACION DE PUERTOS.

Los puertos de entrada/salida estan decodificados en bloques de cuatro por un decodificador de 1 a 8. El Z-80 PIO y el Z-80 CTC tienen decodificadas sus cuatro direcciones en una sola direccion. La siguiente tabla nos muestra como estan decodificados los puertos.

Puerto Selecionado	Direcciones	Conectado a
PS0	00H-03H	Z-80 PIO
PS1	04H-07H	Z-80 CTC
PS2	08H-08H	Latch segmentos
PS3	0CH-0FH	Latch displays
PS4	90H-93H	KB-SEL
PS5	94H-97H	UNUSED
PS6	98H-98H	UNUSED
PS7	9CH-9FH	UNUSED

PS5, PS6, y PS7 estan marcados en el area de alambrado para decodificar la entrada/salida de algun circuito propio. La siguientes tablas nos muestra como estan asigandas las direcciones de los puertos del PIO y del CTC.

Direcion del puerto	Usado por el PIO
80H	Registro de datos puerto A
81H	Registro de datos puerto B
82H	Registro de control de A
83H	Registro de control de B

Direcion del puerto	Usado por el CTC
80H	Canal 0
81H	Canal 1
82H	Canal 2
83H	Canal 3

2.11 SISTEMA DE RAM.

El usuario tiene 1K de RAM proporcionada por los circuitos 21L02-1. Se puede agregar 1K mas de memoria si ponemos 8 circuitos de los anteriores. Los datos son tomados de la memoria por medio del circuito 74LS244 que es un Buffer.

2.12 LOGICA DE LA EJECUCION PASO A PASO (SINGLE STEP).

El Z-80 Starter Kit tiene un "HARDWARE SINGLE STEP" que permite que el comando de SINGLE STEP pueda operar en programas localizados ya sea en RAM o en EPROM/ROM. El canal 2 del CTC se usa para producir un pulso al inicio de la primera instruccion del usuario despues de que el comando de Single Step fue utilizado. Este pulso pasa por varias compuertas y luego llega a la entrada de interrupciones no mascarables. La entrada NMI negada se reconoce despues de que una instrucion se termina de ejecutar, y el CPU direcciona 66H, esta direccion esta en el ZBUG Monitor. El monitor salva los valores de los registros del CPU y despliega el contenido del contador del programa y del acumulador.

2.13 INICIO DE PROM1.

Cuando inicializamos el CPU, el monitor checa la posicion del interruptor S3. Si S3 esta puesto en la posicion MONITOR RESTART, el ZBUG pondra el prompt "-" en el despliegue y empezara a barrer el teclado. Pero si S3 esta puesto en PROM1 RESTART, entonces automaticamente se empieza a ejecutar el programa que se encuentre en la direccion 0800H. Esta caracteristica nos permite iniciar un programa del usuario sin necesidad de ocupar el tclado.

2-14 ENCADENAMIENTO DE INTERRUPCIONES.

Toda la familia de dispositivos perifericos para el Z-80 tienen una circuiteria para el control de interrupciones, de esta manera no se requiere de un controlador muy complejo. La prioridad de las interrupciones esta determinada por una cadena de margarita que corre a travez de los dispositivos del Z-80 (IEI-entrada, IEO-salida). En este kit la mas alta prioridad la tiene asignada el CTC, y le sigue el PIO. La salida de la cadena de margarita se encuentra en el area de alambrado marcada con IEO, puede ser usada para continuar la cadena con algunos otros dispositivos que se agregen. Vea el manual del Z-80 para mayor informacion de la estructura de la cadena de margarita.

2.15 INTERFASE CON EL BUS S-100.

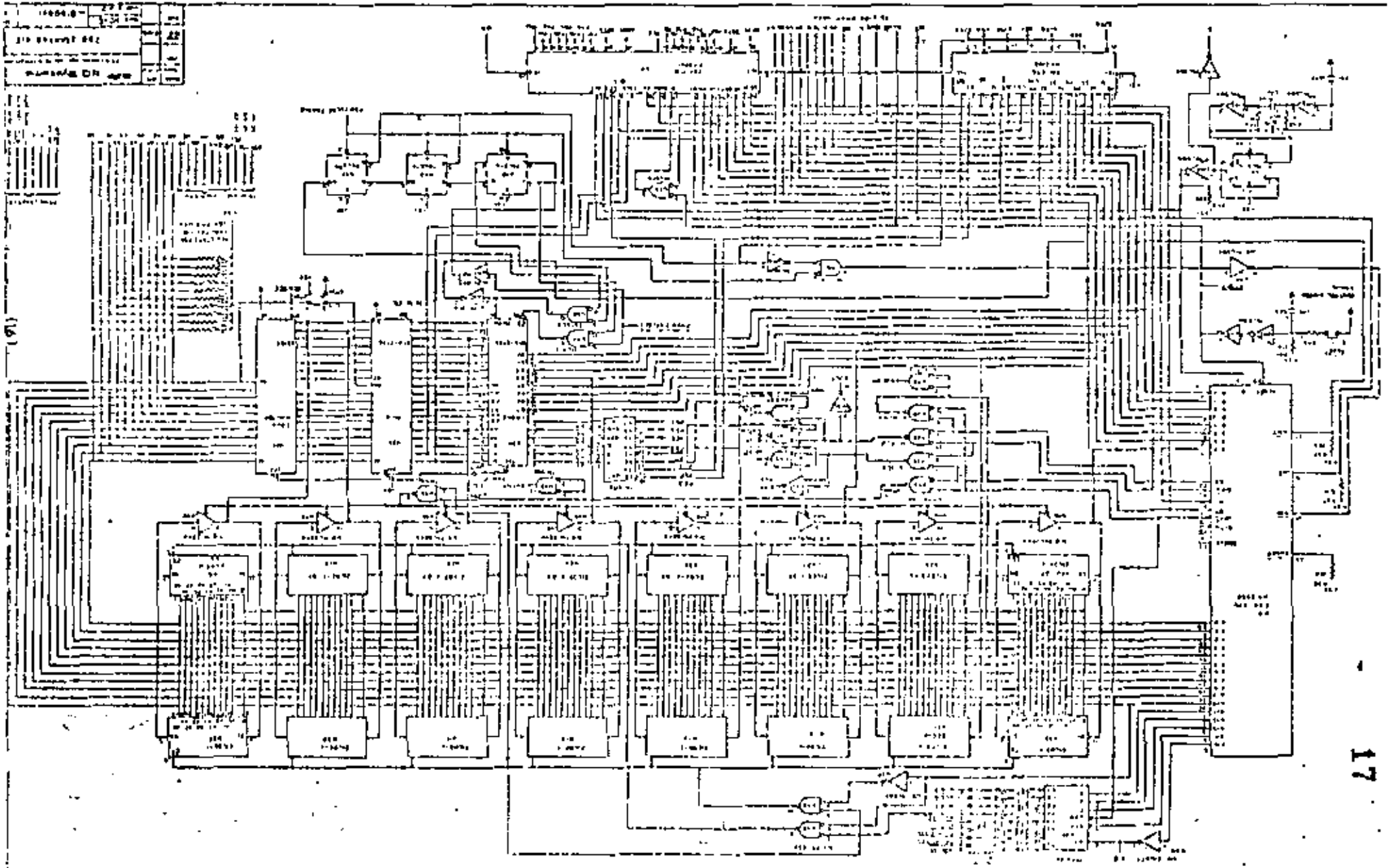
El Starter Kit tiene dos hileras para colocar conectores de 100 patas, que tengan la configuracion S-100. Esta interface es compatible con la mayoria de las memoria estaticas o con las terjetas de expansion de entrada/salida. Especificamente es compatibble con "S. D. Systems' 4K byte Static RAM card", pero no con modulos de EXPANDORAM. La interfase con modulos que requieren senales especificas del 8080A como SYNC, INTA, DBIN, POC, PWR, y PRD requieren de una logica adicional y de la conexion de algunas senales al conector S-100. Vea el esquema del KIT para que las conexiones sean las correctas. Una fuente de poder de +8, +-18 Volts tiene que ser conectada al S-100 por medio de las patas correspondientes en el kit.

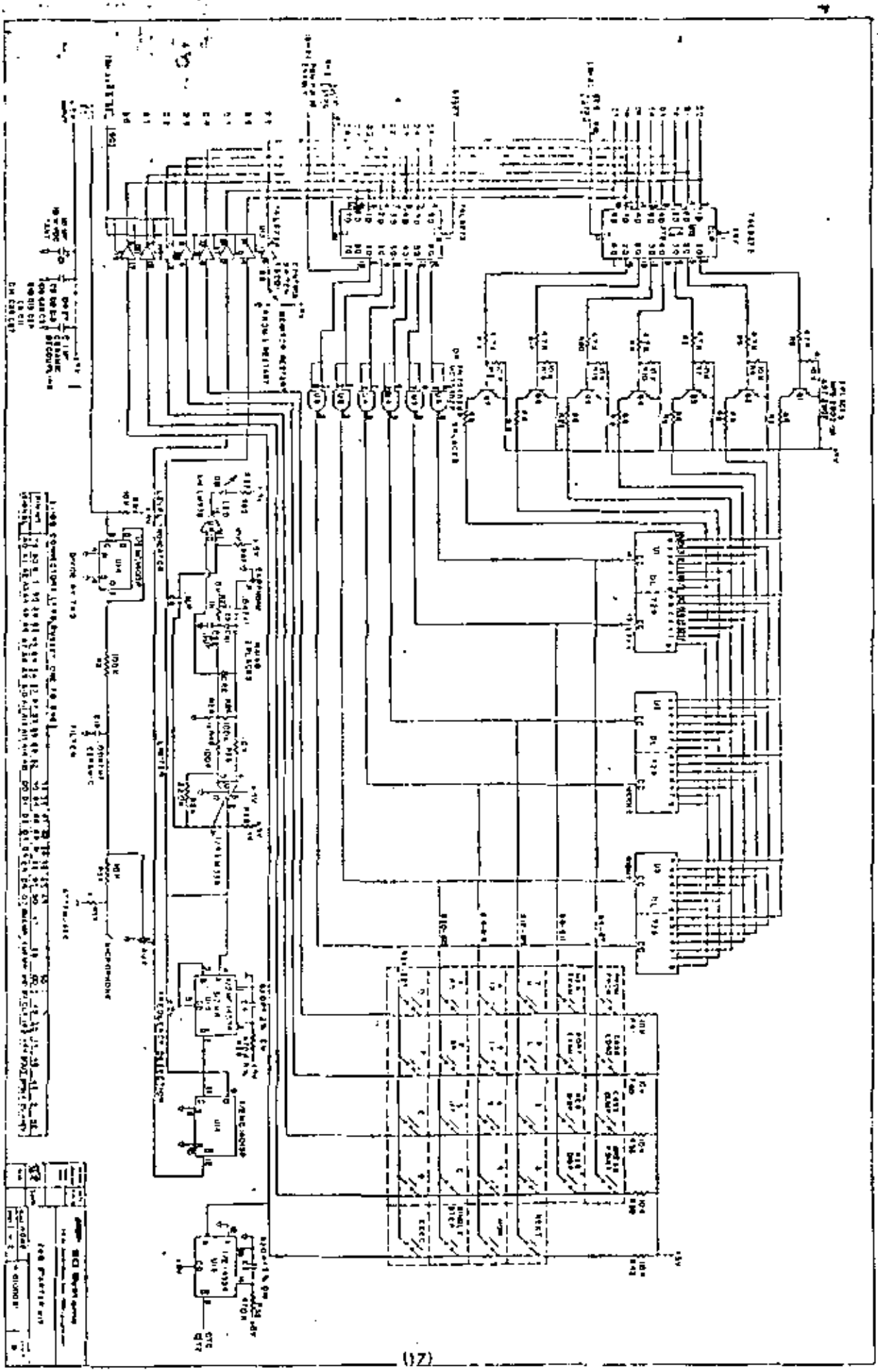
2-16 AREA DE ALAMBRADO.

El area de alambrado sirve para colocar aproximadamente 25 chips de algun experimento, de expansion de memoria, de alguna interfase de video, etc. La alimentacion y la tierra estan alternados en el lado de abajo para que cada IC tenga una baja impedancia tanto a la fuente de poder como a la tierra, reduciendo considerablemente el problema del ruido en la circuiteria del usuario. Z-80 CPU, PIO, y el sistema para decodificacion de senales tienen una extencion en esta area para asi poder hacer las conexiones de una manera mas facil.

```
 CROMEMCO Z80 Macro Assembler version 03.09
*** MOVOCHO ***

                        0001  ;
                        0002  ;
                        0003  ;
                        0004  ;
                        0005  ;       Este programa sirve para mover el caracter "8" de derecha a
                        0006  ;       izquierda a travez de los displays
                        0007  ;
                        0008  ;
      (064F)            0009  D20MS   EQU 064FH
      (2000)            0010          ORG 2000H       ; Se indica que el programa se ensamblara a partir de
                        0011                          ; la direccion 2000H
2000  3E00              0012          LD A,00H
2002  D368              0013          OUT (68H),A     ; Se activan todos los segmentos
2004  3E01              0014          LD A,01H
2006  D38C              0015  LOOP:   OUT (8CH),A     ; Se activa el digito a encender
2008  CD4F06            0016          CALL D20MS
200B  CD4F06            0017          CALL D20MS
200E  CD4F06            0018          CALL D20MS
2011  CD4F06            0019          CALL D20MS      ; Retardo de aproximadamente 100 mseg.
2014  07                0020          RLCA            ; Rotacion para seleccionar el digito siguiente
2015  18EF              0021          JR LOOP
                        0022


Errors                   0
Range Count              0

 CROMEMCO Z80 Macro Assembler version 03.09
*** MOVOCHO ***
Symbol      Value  Defn  References

D20MS       064F   0009  0016 0017 0018 0019
LOOP        2006   0015  0021
```

```
 CROMEMCO Z80 Macro Assembler version 03.00
*** RAIZCUAD ***

                        0001  ;
                        0002  ;
                        0003  ;
                        0004  ;
                        0005  ;
                        0006  ;    Este programa calcula la raiz cuadrada
                        0007  ;de un numero entero.
                        0008  ;    En la direccion 2100 hay que cargar el numero
                        0009  ;al cual hay que sacarle raiz cuadrada
                        0010  ;    El cuadrado del numero aparece en el registro 'B'; en
                        0011  ;caso de que el numero no tenga raiz cuadrada exacta (por
                        0012  ;ejemplo 6), el resultado que se obtiene es el numero mas
                        0013  ;cercano a la raiz.
                        0014
      (2000)            0015          ORG 2000H
      (2100)            0016  DATO:   EQU 2100H
2000  3A0021            0017          LD A,(DATO)     ;Cargamos 'A' con el contenido de 'DATO'
2003  0600              0018          LD B,00H        ;Inicializamos 'B' con 0
2005  04                0019  RAIZ:   INC B           ;Incrementamos 'B' en 1
                        0020                          ;Llamamos a una rutina que calcula el
                        0021                          ;cuadrado de 'B'.
2006  CD1420            0022          CALL CUAD
2009  BA                0023          CP A,D          ;Si 'A'='D' then finalizamos else if 'D'>'N'
                        0024                          ;then decrementa 'B' y final else goto raiz
200A  CA1120      R     0025          JP Z,FIN
200D  F20520            0026          JP P,RAIZ
2010  05                0027          DEC B
2011  CD0000            0028  FIN:    CALL 00H
                        0029  ;
                        0030  ;Subrutina que calcula el cuadrado del numero 'B'
                        0031  ;
2014  F5                0032  CUAD:   PUSH AF         ;Ponemos en el stack el contenido de 'AF'
2015  48                0033          LD C,B          ;Hacemos 'C'='B'
2016  3E00              0034          LD A,00H        ;Inicializamos 'A' con 0
2018  80                0035  SUMA:   ADD A,B         ;Sumamos 'A' con 'B'
2019  0D                0036          DEC C           ;Hacemos 'C'='C'-1
201A  C21820      R     0037          JP NZ,SUMA      ;Entramos en un Loop hasta que 'C'=0
                        0038                          ;Si 'C'=0, entonces 'D'='B'+'B' y sacamos
                        0039                          ;el contenido del stack
                        0040                          ;else regresamos a suma
201D  57                0041          LD D,A
201E  F1                0042          POP AF
201F  C9                0043          RET             ;Fin de la subrutina


Errors              0
Range Count         2

 CROMEMCO Z80 Macro Assembler version 03.00
*** RAIZCUAD ***
Symbol      Value  Defn  References

CUAD        2014   0032  0022
DATO        2100   0016  0017
FIN         2011   0028  0025
RAIZ        2005   0019  0026
SUMA        2018   0035  0037
```





PRACTICA # 2

Ensamblado de programas en Cromemco.

OBJETIVO.

Al final de la practica el alumno sabra como utilizar la microcomputadora Cromemco CS-2 para que ensamble sus programas elaborados en Z-80.

INTRODUCCION.

El sistema Cromemco es una microcomputadora basada en el microprocesador Z-80. Nos permite la utilizacion de una serie de programas con los que el programar en ensamblador resulta relativamente facil. Estos programas son el editor SCREEN, el ASMB, el LINK y el DEBUG.

El SCREEN es un editor de pantalla que nos permite teclear los programas en ensamblador, corregirlos y almacenarlos en el Diskette.

Una vez que ya tenemos preparado el programa fuente utilizamos el ASMB para ensamblarlo. El ensamblador lee el programa fuente, ensambla las instrucciones en Z-80, y produce un codigo objeto y una lista del programa ensamblado.

Como paso siguiente utlizaremos el programa LINK. Este programa carga los archivos objeto relocalizables en la memoria y cambia todas las direcciones relativas en las direcciones de memoria actuales. Link puede cargar uno o mas archivos durante su ejecucion y buscar una libreria especifica del usuario de alguna rutina relocalizable para ponerla en un modulo de codigo comun.

Por ultimo utilizaremos el programa DEBUG. DEBUG es un programa que hace posible evaluar, depurar, y trabajar en los programas del usuario.

Los parrafos anteriores nos muestran de manera general los pasos que se siguen para trabajar con Cromomco. La figura 1 es la grafica de estos. Cabe aclarar que esto se detallara mas adelante.

DESARROLLO.

Supongamos que tenemos un programa elaborado en Z-80, entonces los pasos para meterlo son cuatro :

1) Teclear:

A.screen binombre.z80

Se crea dos programas llamados:

nombre.z80 y nombre.bak

El denominado nombre.z80 es el programa fuente, mientras que el nombre.bak es la version anterior de este.

2) Teclear:

B.asmb nombre

Tambien se crean dos programas que son:

nombre.prn y
nombre.rel

El archivo nombre.prn tiene el listado del programa ensamblado; y el denominado nombre.rel es el archivo con el programa en codigo objeto.

3) Teclear:

B.link nombre,nombre/n/e

Se crea el programa ejecutable nombre.com.

4) Teclear:

B.nombre

Esto origna que el programa se ejecute.

5) Teclear:

B.debug nombre.com

Si deseAmos utilizar el depurador.

A continuacion se mencionaran algunos de los comandos fundamentales que se ocupan al invocar cada programa. Para esto suponemos que el texto que a continuacion se muestra es la pantalla y que los comentarios (que no aparecen en la pantalla) estaran entre <...> picoparentesis. OK??.

< Primero llamamos al editor SCREEN >

```
A.Screen b:programa1.z80                    <   llamamos al screen>
Editor Screen 910: D I M E                  < se borra la pantalla>

>Edit: @ Copy Delete Exit Find Insert Jump Move Other Substitute..
```

<El letrero aparece en la parte superior de la pantalla, lo que> <viene despues de los dos puntos son los comandos. Para invocar> <algun comando simplemente hay que teclear la letra mayuscula> <correspondiente. Por ejemplo si queremos insertar tecleamos la> <'I' y la maquina respondera con : >

```
>Insert <text.....> <esc>
```

<Esto quiere decir que vamos a insertar y que terminaremos de> <insertar tecleando la tecla de ESC que esta en la parte superior> <derecha.>

<Ahora si queremos borrar teclearemos la D, la respuesta es: >

>Delete : <esc>

<Esto quiere decir que borraremos texto tecleando espacios o la>
<tecla de RETURN, y que terminaremos tecleando la tecla ESC. >
<La misma filosofia se sigue para los demas comandos. >
<Por ultimo explicaremos el que se utiliza para salvar el >
<programa en disco. Teclearemos la letra 'E', la respuesta sera: >

>Exit: Quit Update <esc>
Quit -exit without updating nombre.z80
Update -update nombre.z80 and exit
<esc> -return to editor

<Ahora tenemos que selecionar entre la letra 'Q' o la 'U'. Si >
<tecleamos la letra 'Q' todos los cambios realizados no seran >
<tomados en cuenta. Si tecleamos la 'U' el programa se salva y se>
<crean los programas llamados nombre.z80 y el nombre.bak. La >
<maquina responde con: >

A.

<Ahora si como siguiente punto ensamblaremos el programa, tenemos>
< que cambiar el control al Drive B. Teclearemos:>

A.B:

< La maquina pone: >

B.

< Para ensamblar ponemos: >

B.asmb nombre

<La maquina ensambla el programa y nos indica si existen errores >

CROMEMCO Z80 Macro Assembler version nn.nn

Errors 0
Range Count 0

Program Length 005F (95)

END of assembly
B.

<Si existen errores, se tiene que corregir el programa por medio>
<del screen y reensamblar el programa. En caso contrario hay que>
<ligar el programa. Hay que hacer notar que si nuestro programa>
<no se va a correr en la Cromemco no se necesita ligar. >

B.link nombre=nombre/n/e

```
< La opcion /n causa que el link cree un nuevo archivo ejecutable>
<con el nombre nombre.com.  Mientras que /e sirve para terminar y>
<regresar  el control al sistema operativo.  Si no hay errores la>
<respuesta es:                                                    >

Link version nn.nn

Data      0103     0162

[0103  0162 1]

B.

<Para ejecutar el programa simplemente teclee:                    >

B.nombre

< Ahora si queremos utilizar el debug tendremos que teclear:      >

B.debug nombre.com
DEBUG version 00.17
NEXT  =  4F00
NEXTM =  4F00
-

< Entre los comandos mas utilizados en este modo tenemos:         >

-D direc1 direc2 <Despliegue de localidades de memoria

< Que nos despliega el contenido de las localidades de memoria en>
< hexadecimal de la dereccion 1 a la direccion 2                  >

0100   40 41 42 43  44 45 46 47  40 .......        @ABCDEFGH(JKLMNO
0110   50 51 52 53  54 55 56 57  58 .......        PQRSTUVWXYZ01234
0120   35 36 37 38  39 00 00 00  00 .......         56789

(Si queremos ver el valor de los registros y las banderas ponemos>

-DR

Su respuesta sera:                                                >

SZ1VNCE     A =00 BC =0000 DE =0000 HL =0000  ......     LD E,A
SZ1VNC      A'=00 BC'=0000 DE'=0000 HL'=0000

< Para poner los puntos de ruptura, Teclee:                       >

-B 0106

  Esta instruccion pone un punto de ruptura en la direccion 0106.>
<Se   pueden  poner  hasta 12 Break Points.    Para quitarlos hay>
<que poner: >

BX direc1
```

```
< Si queremos ver el ultimo punto de ruptura simplemente ponemos >

B

< Para correr un programa simplemente hay que teclear!          >

G direc

< El programa empieza a correr a partir de la direccion marcada. >

A                          < Ensamblado en la memoria           >
A beginning-ador


<Este comando permite al usuario teclear nemonicos de lenguaje>
<ensamblador desde la consola y tener su ensamblado en la memoria>
<El primer foramato ensmbla en la memoria desde la direccion de>
<default (100H). El segundo formato comienza a partir de la>
<direccion de inicio indicada por el usuario.>
<El DEBUG responde con la direccion absoluta, la direccion>
<relativa, y la instruccion que se encuentra en esa localidad.  >
<Este comando lee de la consola y ensambla en la memoria las>
<instrucciones. Si no existe error en la instruccion esta se>
<ensambla y el usuario recibe como respuesta la siguiente>
<instruccion. Las reglas para direccionar una expresion tambien >
<se aplican a las direcciones en mnemonicos. En el siguiente>
<ejemplo el registro B contiene la direccion 1234H              >

                1274    0040'   NOP     ADD B
                1275    0041'   NOP     CALL 893
                1278    0044'   NOP     JP 1032+95
                127B    0047'   NOP     .

<Si solo tecleamos un RETURN, el Debug no altera la instruccion >
<que existe en esa localidad y va a la siguiente instruccion en >
<la memoria.                                                    >
<Este comando se termina se termina inicando la linea con un '.'>
<Si existe un error en la linea que se tecleo, esta no es>
<aceptada y una marca es mandada a la consola.                  >

S direc          <          >
S

< Este comando sirve para cambiar las localidades de memoria. >
< Tiene las siguientes opciones:                               >
<                                                              >
<       - Si tecleamos un dato seguido por un RETURN, el dato  >
<         es almacenado en la direccion del prompt. La         >
<         direccion se incrementa en uno y se despliega la     >
<         siguiente linea.                                     >
<       -  Si ponemos un string colocado entre apostrofes (')  >
<         seguido por un RETURN, el string se almacenara       >
<         desde el inicio de la direccion del promp. La        >
<         direccion se incrementa tantos caracteres como       >
<         hayamos tecleados y se despliega la siguiente        >
<         linea.                                               >
```

< - Si tecleamos un signo de menos (-), la direccion se >
< decrementa y se despliega la localidad a la que ta >
< regreso. - Si solamente ponemos un RETURN, la >
< direccion se incrementa y esa localidad no es >
< afectada. >
< - Un punto (.) nos sirve para salir de este modo. >
< >
<En el siguiente ejemplo, asumimos que el registro @ tiene como >
<valor 2800H . >

```
-S 0100
2900    0100'   32      0
2901    0101'   17      00
2902    0102'   31      'this is an ASCII string'
2919    0119'   7A      'AAAA' 0 0 1 2 3 4 5 6 7 8 9
2928    0128'   22
2929    0129'   29
292A    012A'   87-
2929    0129'   55
292A    012A'   87.
```

< Tambien tenemos un comando para listar el contenido de las >
< localidades de memoria, es: >

```
L
L starting-addr
L starting-addr ending-addr
```

< El primer formato lista 16 lineas a partir de la direccion >
< actual. El segundo formato los lista a partir de la dir ecto >
< que le indicamos. Y el tercero lo hace dentro del rango >
< seleccionado. >
< La primera direccion del desensamblado es la direccion >
< absoluta. La segunda direccion es la direccion relativa. Si >
< la instruccion desensamblada contiene una direccion, la >
< direccion absoluta se despliega a la derecha de esa linea. En >
< el ejemplo que sigue, el registro @ contiene un 2800H. >

```
-L@800  0811
3000    0800'   ADD     A,B
3001    0801'   CALL    3200    (0A00')
3004    0804'   CALL    3243    (0A43')
3007    0807    CALL    3333    (0833')
300A    080A    LD      A,B
300B    080B'   OR      A,C
300C    080C'   JR      Z,3000  (0800')
300E    080E'   INC     HL
300F    080F'   INC     DE
3010    0810'   INC     BC
3011    0811'   LD      A,H
```

< Para terminar de utilizar el Debug teclee: >
< ^C < el caracter de control CTRL-C > >
B.

Con esto terminamos la sesion. Para mayor informacion sobre todos los comandos que se utilicaron, favor de consultar los manuales de referencia.

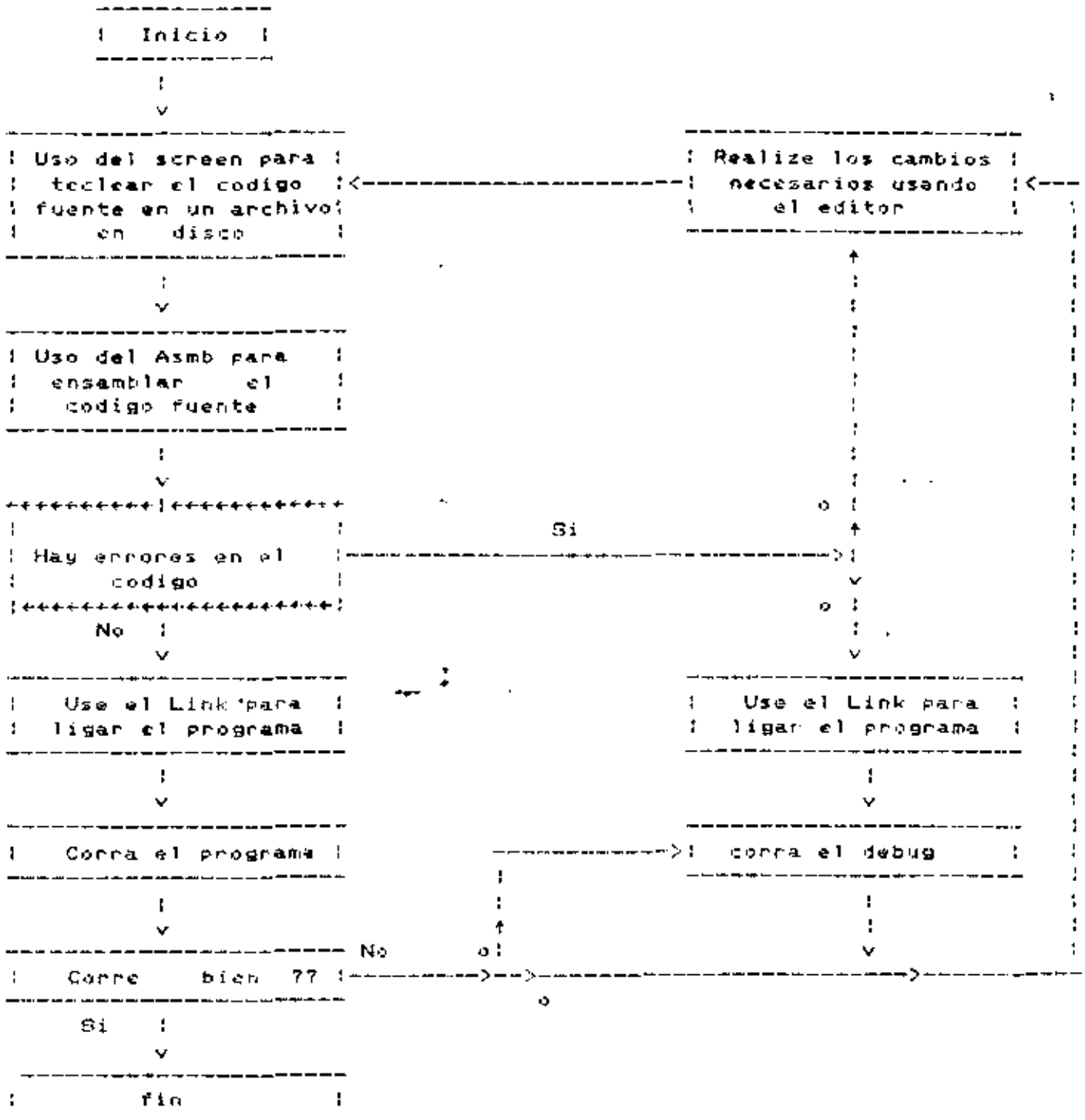

Inicio
Uso del screen para teclear el codigo fuente en un archivo en disco
Uso del Asmb para ensamblar el codigo fuente
Hay errores en el codigo
Si
No
Use el Link para ligar el programa
Corra el programa
Corre bien ??
No
Si
fin
Realize los cambios necesarios usando el editor
Use el Link para ligar el programa
corra el debug

FIGURA (1)

PRACTICA No. 3

Z-80 PIO CONTROLADOR ENTRADA/SALIDA EN PARALELO

OBJETIVO.

El objetivo de esta practica es que el alumno conozca de manera general cual es la arquitectura del PIO, y los diferentes modos en los que se puede programar.

INTRODUCCION.

El Z-80 PIO es en circuito programable que tiene dos puertos independientes compatibles con TTL, ademas es una interface entre el procesador y los perifericos. Estos perifericos pueden ser teclados, impresoras, convertidores D/A, etc.

Una de las principales caracteristicas del Z-80 PIO es que todas las transferencias de datos entre el procesador y los perifericos se realiza usando la logica de control de interrupciones. Esta logica es usada por los circuitos de soporte del procesador Z-80. Otra caracteristica del PIO es su habilidad para interrumpir al CPU al ocurrir ciertas condiciones programadas por el usuario. Sin esta capacidad de interrupciones el procesador tendria que estar checando el estatus de cada dispositivo, lo que haria mas lenta determinadas tareas.

Cada uno de los puertos tiene un canal de datos de 8 bits y dos senales de enlace (READY y STROBE). Estas senales controlan la transferencia de datos .

Los puertos pueden ser programados en 4 modos diferentes de operacion:

MODO 0: Este modo se usa para enviar informacion del CPU a los perifericos a traves de las 8 lineas del puerto seleccionado.

MODO 1: Se usa para leer datos de uno de los perifericos hacia el CPU.

MODO 2: En este caso solo se puede programar al puerto A debido a que utiliza todas las lineas de enlace, tanto de el mismo, como del puerto B. En este modo se tiene, en un instante dado, al puerto A trabajando como salida y en otro instante actuando como entrada.

MODO 3: Se usa para aplicaciones de control y evaluacion. Puede ser usado por ambos puertos. Las senales de enlace no se ocupan. El usuario puede programar que lineas de un puerto son de entrada y cuales son de salida.

ARQUITECTURA.

El diagrama de bloques del Z-80 PIO se muestra en la figura 1. La estructura interna del Z-80 PIO consiste de un canal de interface; una logica de control interna; y la logica de entrada / salida para los puertos. Una aplicacion tipica puede ser la de usar el puerto A para transferencia de datos y el puerto B para la evaluacion y el monitoreo.

La figura 2 nos muestra la arquitecuta de un puerto de entrada/salida. Los registros que tiene son: Un registro de entrada de 8 bits; uno de salida de 8 bits; un registro de 2 bits para controlar el modo de operacion; un registro de seleccion de entrada/salida de 8 bits y dos registros de 2 bits de control de mascara. Los ultimos tres registros se usan solo cuando un puerto ha sido programado para operar en el modo bit (modo 3).

DESCRIPCION DE LOS REGISTROS.

El registro de modo de control (2 bits); es cargado por el CPU para seleccionar el modo de operacion :'BYTE OUTPUT' (modo 0); 'BYTE INPUT' (modo 1); 'BYTE BIDIRECTIONAL' (modo 2); o el modo bit.

El registro de salida de datos (8 bits); permite que un dato sea transferido desde el CPU hacia un periferico.

El registro de entrada de datos (8 bits); recibe los datos de un periferico para transferirlos al CPU.

El registro de control de mascara (2 bits); es cargado por el CPU para especificar un estado activo (alto o bajo) de alguno de los 'PINS' de un periferico que se este monitoreando; ademas indica si una interrupcion puede ser generada cuando todos los 'PINS' esten activos (condicion AND) o bien cuando alguno de ellos este activo (condicion de OR).

El registro de mascara (8 bits); se carga por el CPU para determinar que 'PINS' de algun periferico se estan monitoreando para una condicion de estatus especifica.

El registro de seleccion de entrada/salida (8 bits); es cargado por el CPU para permitir que algun 'PIN' sea utilizado como de entrada o de salida durante el modo de operacion de bit.

FIG. 2

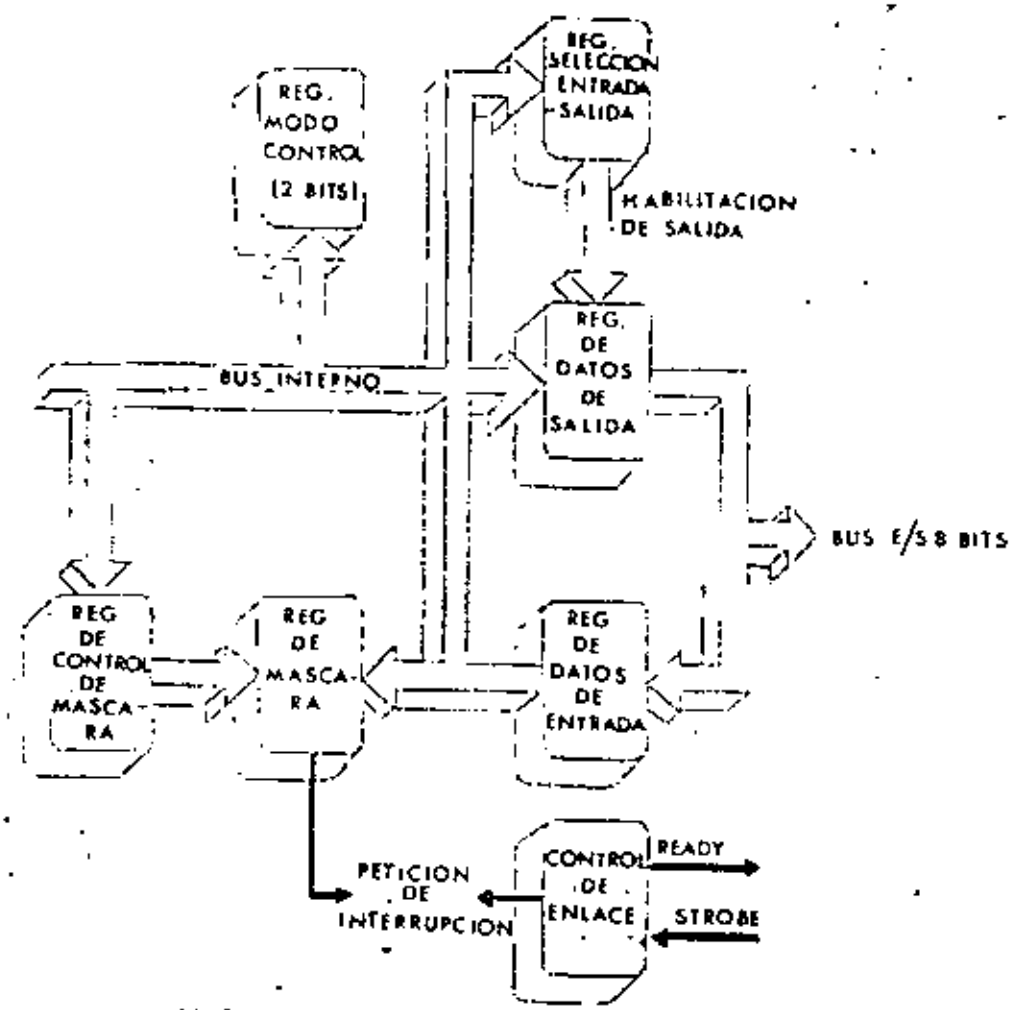


DIAGRAMA DE BLOQUES DE UN PUERTO DE E/S

FIG. 1

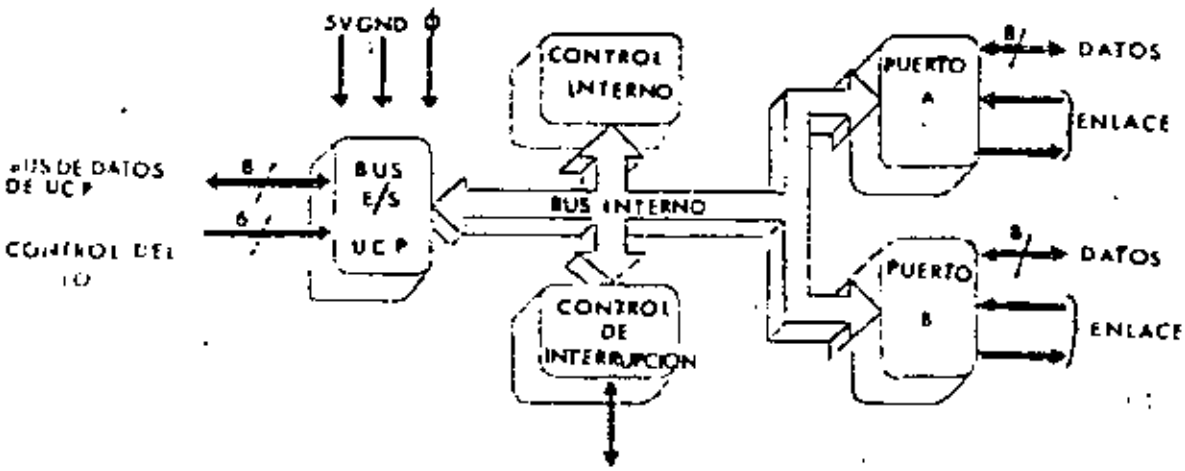
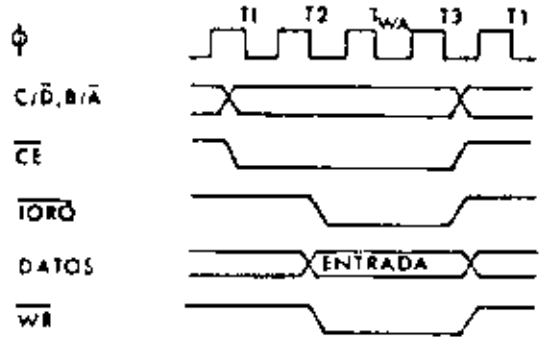


DIAGRAMA DE BLOQUES DE PIO

CICLO DE ESCRITURA.

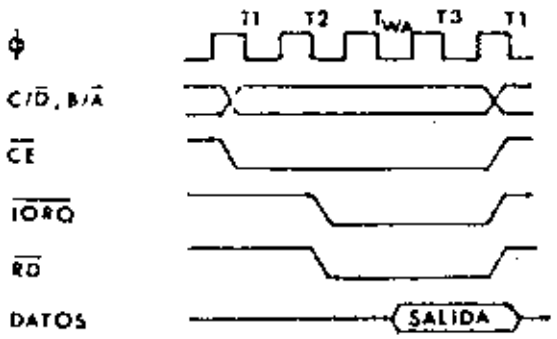
Esta figura nos muestra los tiempos que se necesitan para programar el Z-80 PIO o para escribir datos en uno de sus puertos. No se necesitan asignar estados de espera (WAIT) para escribir en el PIO ya que estos se asigan en el pulso TWA. El PIO no recibe señales especificas de escritura; internamente genera sus señales debido a la falta de una señal activa de RD (negada)..



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

CICLO DE LECTURA.

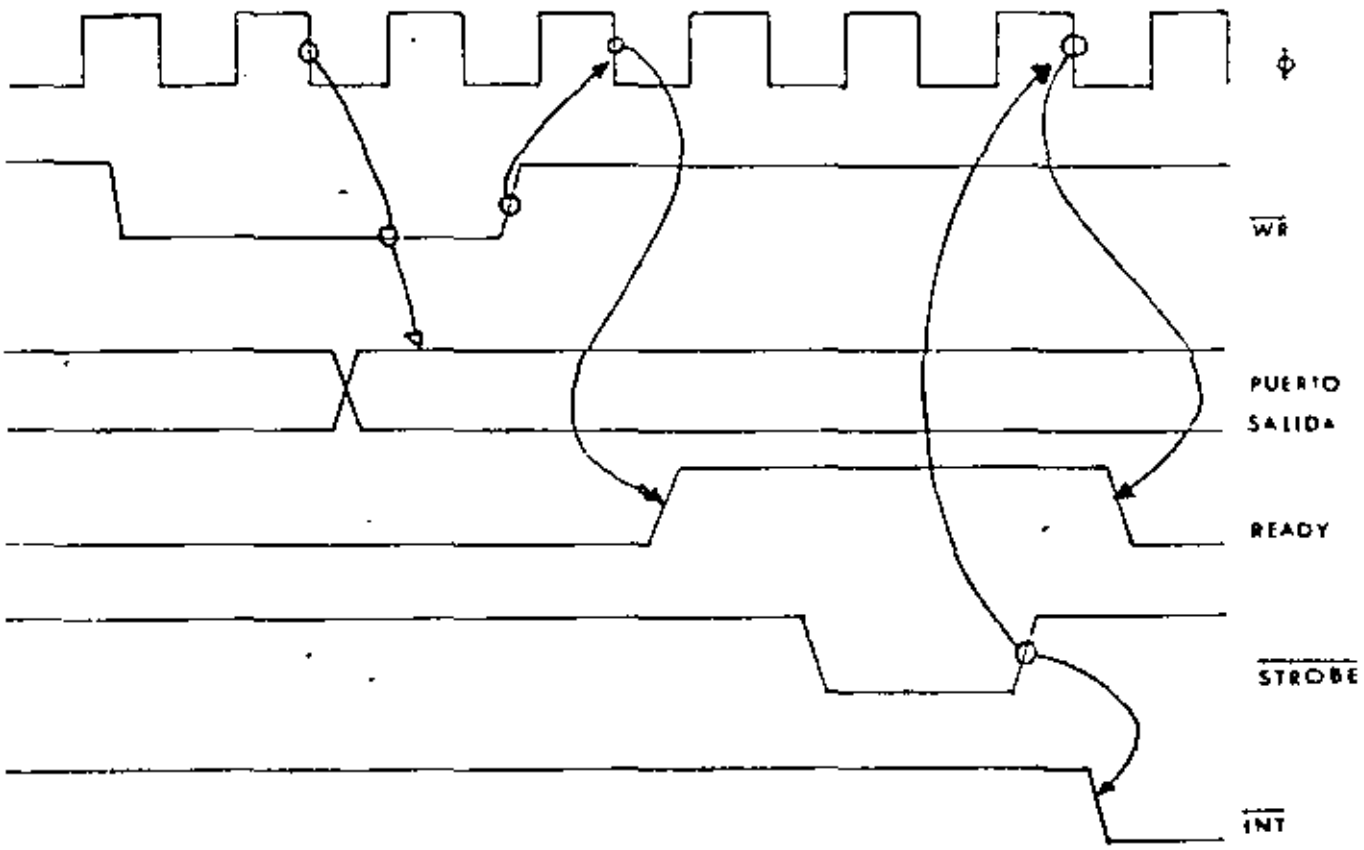
Esta figura nos muestra los tiempos de cuando se lee un dato en alguno de los puertos del Z-80 PIO. Tampoco requiere de la asignacion de tiempos de espera WAIT dado que automaticamente se inserta TWA.



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

MODO SALIDA.

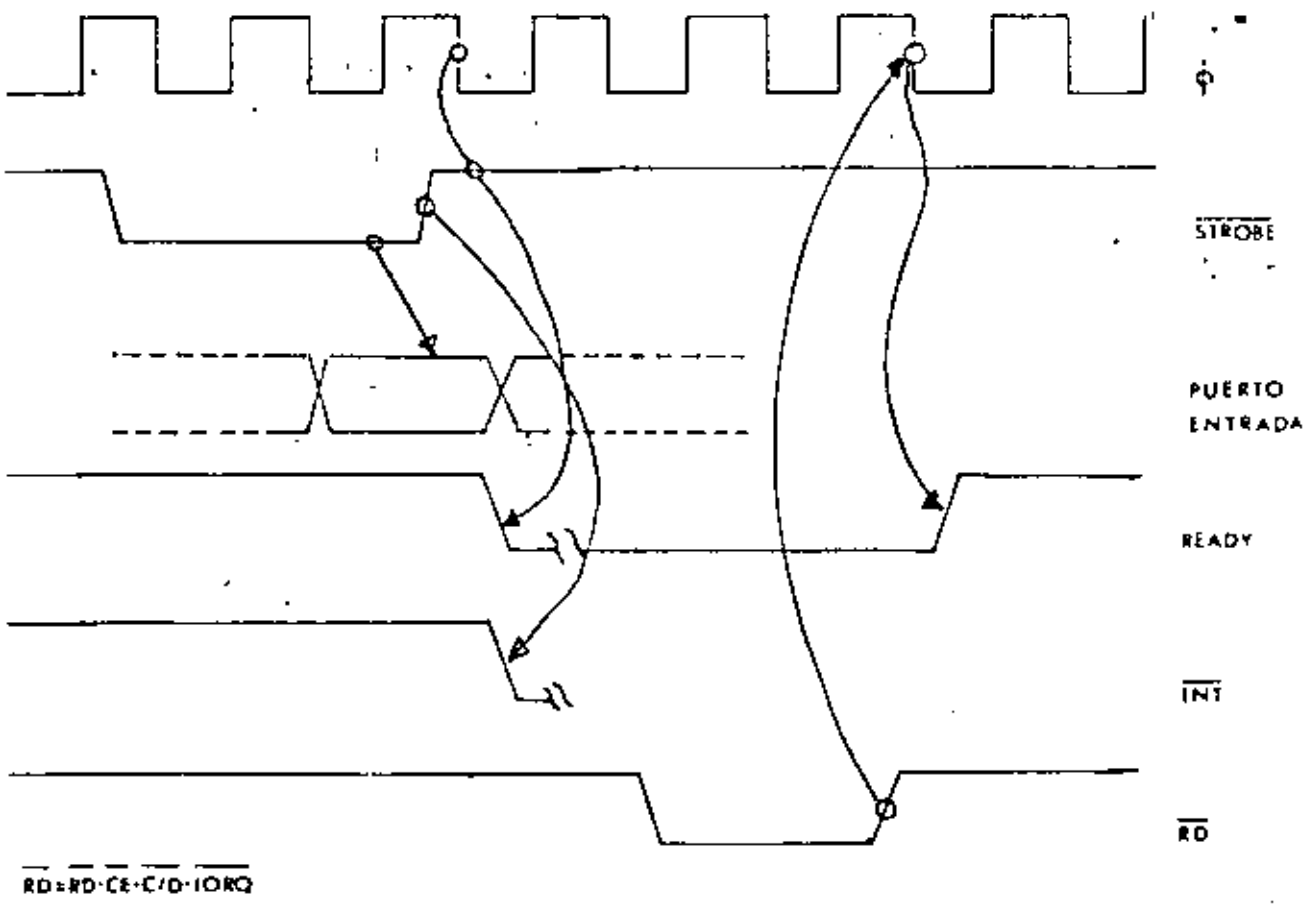
Un ciclo de salida se iniciara con la ejecucion de una instruccion de salida por el CPU. La senal WR (negada) del CPU, guarda los datos del bus del CPU en el registro de salida del puerto seleccionado. El pulsos de escritura, prende la bandera de READY en el flanco de bajada del reloj , indicando que un dato esta disponible. La senal de READY estara activa hasta que el flanco de subida de la linea de STROBE sea recibido, indicando que un dato fue tomado por el periferico . El flanco de subida del pulso de STROBE genera la senal INT (negada), si el FLIP FLOP que habilita las interrupciones ha sido prendido, y si este dispositivo tiene la mas alta prioridad.



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

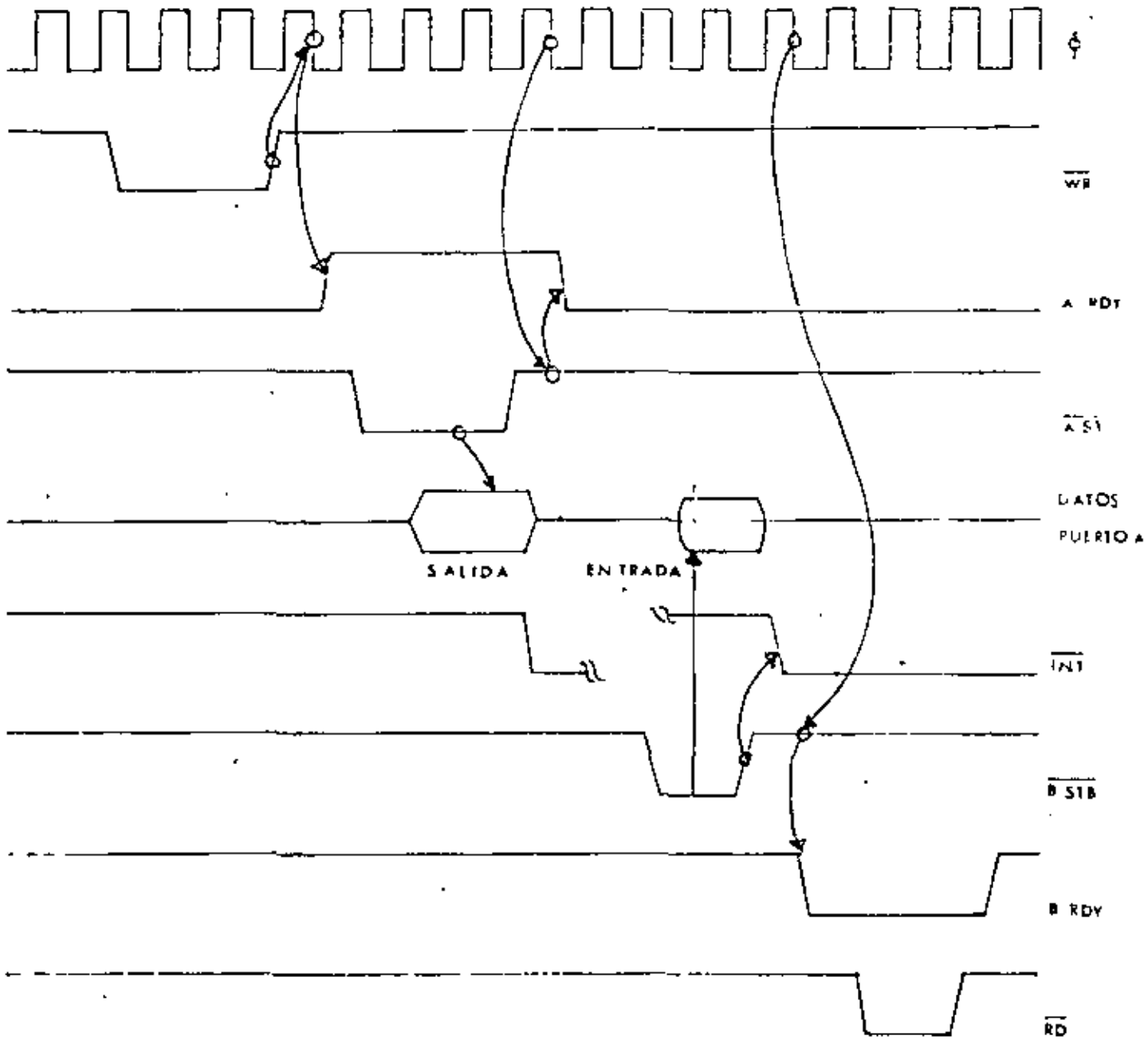
MODO DE ENTRADA.

Cuando la senal de STROBE (negada) es baja, los datos son cargados en el registro de entrada del puerto seleccionado. Cuando sube el flanco de la senal de STROBE (negada), la senal de INT (negada) es activada si la habilitacion de interrupciones esta prendida y ademas el dispositivo que la requiere tiene la mas alta prioridad. En la siguiente caida del reloj, la senal de READY pasa a un estado inactivo, indicando que el registro de entrada esta lleno y que no puede aceptar mas datos hasta que el CPU complete la lectura. Cuando la lectura es terminada el flanco de subida de la senal RD (negada) prende la senal de READY en la proxima transicionn de bajada del reloj. Cuando esto ocurre un nuevo dato puede ser cargado en el PIO.



MODO BIDIRECCIONAL.

Esta es una combinación de los modos 0 y 1 que usa las cuatro líneas de enlace y las 8 líneas de entrada/salida del puerto A. El puerto B se puede utilizar en el modo bit. Las líneas de enlace del puerto A se usan para controlar la salida y las líneas del puerto B se usan para el control de la entrada. Los datos son puestos en el bus del puerto A cuando la señal de ASTB (negada) es baja. El flanco de subida de esta señal se puede usar para almacenar los datos en el periférico.

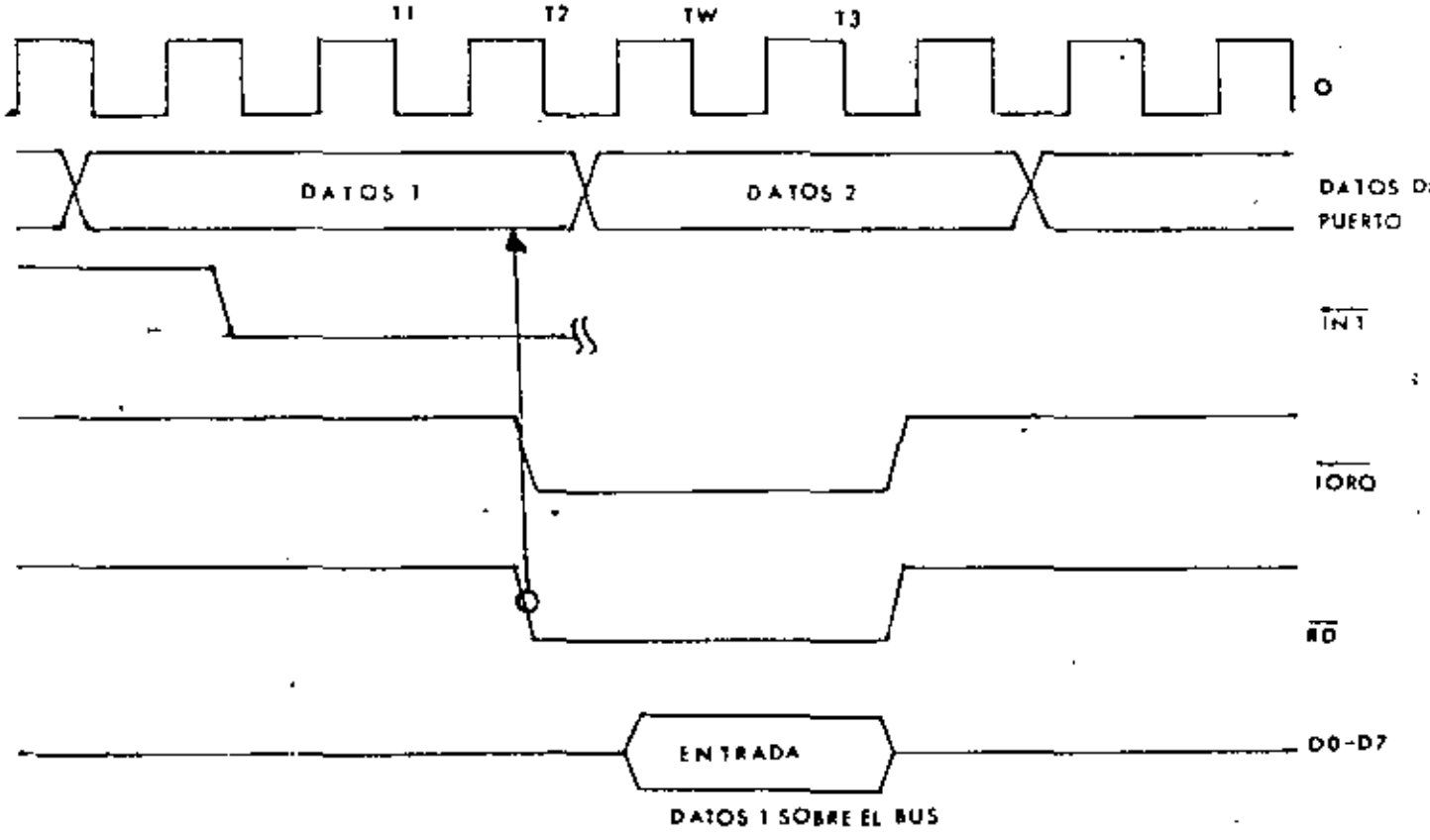


$\overline{WR} = RD \cdot \overline{CE} \cdot C/\overline{D} \cdot \overline{IORQ}$

MODO BIT.

El modo bit no utiliza las señales de enlace. La escritura o lectura en un puerto se pueden realizar en cualquier momento. Cuando realizamos una escritura, los datos serán almacenados en los registros de salida tardando el mismo tiempo que el modo de salida.

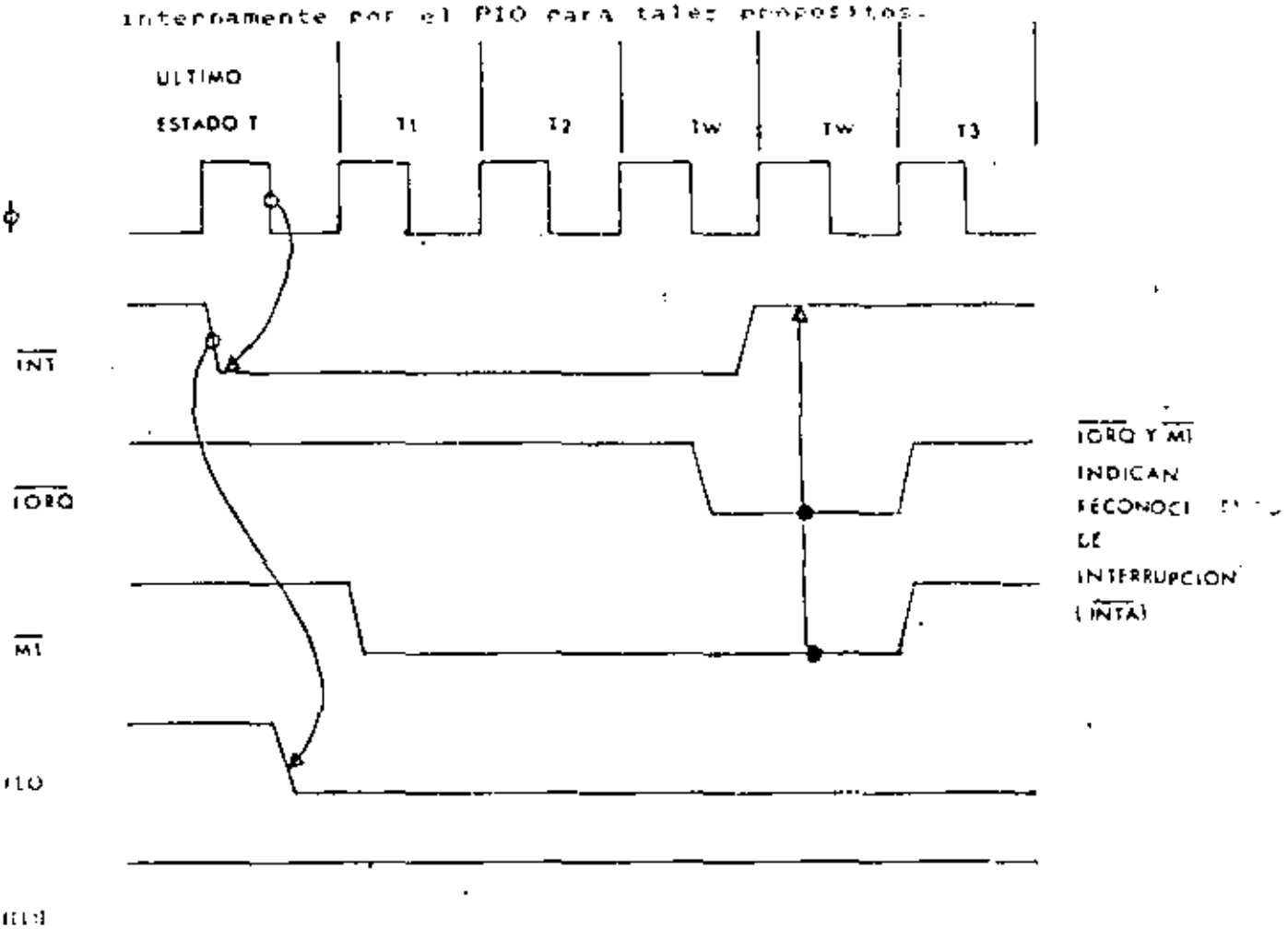
Cuando el PIO lee, los datos regresarán al CPU compuestos de los registros de datos de salida cuyas linea del puerto sean designadas como lineas de salidas y de registros de entrada de datos por las lineas del puerto que han sido designadas como lineas de entrada. El registro de entrada puede contener datos que fueron presentados inmediatamente despues de que el flanco de bajada de la senal RD (negada) ocurre. Una interrupcion sera generada si las interrupciones de un puerto estan habilitadas y los datos en las lineas de los puertos satisfagan la ecuacion logica definida por los registros de control de mascara de 8 y 2 bits respectivamente.



DATOS 1 SOBRE EL BUS

RECONOCIMIENTO DE INTERRUPCIONES.

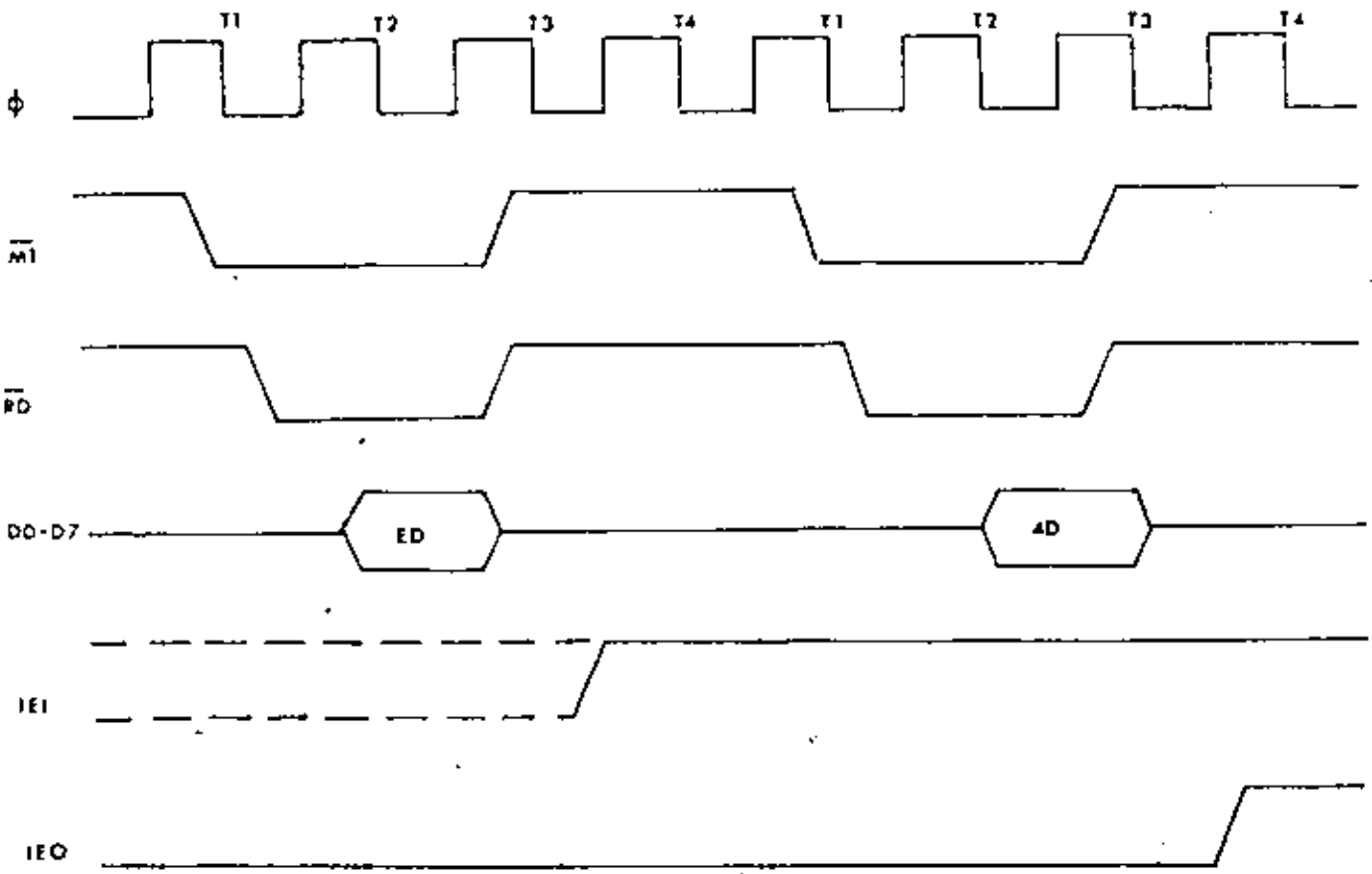
Durante el tiempo que dura M1 (negada), los controladores de los periféricos se inhiben cambiando el estatus que habilita las interrupciones, permitiendo que la señal habilitada de INT pase a través de la cadena de margarita. El periférico que tiene IEI alto e IEO bajo, durante el tiempo que dura INTA (negada) pone el vector de interrupciones preprogramado (8 bits) en el bus de datos. IEO es tomada baja hasta que la instrucción de retorno de interrupción (RETI) sea ejecutada por el CPU mientras que IEI es alta. La instrucción de dos bytes RETI es decodificada internamente por el PIO para tales propósitos.



RETORNO DEL CICLO DE INTERRUPCIONES

Si un dispositivo periferico del z-80 no tiene una interrupcion pendiente y no esta en servicio, entonces su IEO=IEI. Si este tiene una interrupcion bajo servicio (por ejemplo, si ya fue interrumpido y recibe un reconocimiento de interrupciones) entonces su IEO siempre es bajo, inhabilitando a los chips de prioridad mas baja que puedan interrumpir. Si existe una interrupcion pendiente que no haya sido reconocida, IEO sera baja hasta que un 'ED' sea decodificado como el primer byte del codigo de operacion de una instruccion de dos bytes. En este caso IEO subira hasta que el proximo byte del codigo de operacion sea decodificado, luego de esto bajara. Si el segundo byte del codigo de operacion fue un '4D' entonces la instruccion es un RETI.

Despues de que el codigo 'ED' es decodificado, solo el periferico que esta interrumpido y que se esta atendiendo tendra la senal IEI alta e IEO baja. Este dispositivo es el de mayor prioridad en la cadena margarita que ha recibido un reconocimiento de interrupcion. Los demas perifericos tienen IEI=IEO. Si el proximo byte de el codigo de operacion a decodificar es '4D' este dispositivo iniciara su condicion de interrupcion bajo servicio.



PROGRAMACION.

1) Carga del vector de interrupciones.

El CPU-Z80 requiere de un vector de interrupciones de 8 bits que es suministrado por el dispositivo que interrumpe. El CPU forma la direccion de la rutina de atencion de interrupciones del puerto usando este vector. Durante el ciclo de reconocimiento de una interrupcion, el vector es puesto en el bus de datos del CPU por el dispositivo que tiene la mas alta prioridad para ser atendido. El vector de interrupciones se carga en el PIO escribiendo una palabra de control al puerto que lo direcciona. El formato utilizado es:

D7	D6	D5	D4	D3	D2	D1	D0
V7	V6	V5	V4	V3	V2	V1	0

El cero que aparece en el bit D0 nos sirve para indicar que esta palabra de control es el vector de interrupciones. Los bits del D7 al D1, se ocupan para poner una direccion.

Al generarse una interrupcion, el PIO envía este dato hacia el CPU y lo asigna como la parte menos significativa de una direccion de memoria, la parte mas significativa la da el registro I (programado por el usuario) :

direccion = Registro I + Vector de interrupciones

En la localidad obtenida y en la siguiente se encuentra la direccion de una rutina de servicio a la interrupcion generada.

2) Seleccion del modo de operacion.

Cuando seleccionamos un modo de operacion, el registro de control de modo (2 bits) puede tomar uno de cuatro valores. Estos bits son los mas significativos del registro, bit 7 y 6; los bits 5 y 4 no se usan, mientras que del bit 3 al bit 0 se ponen en 1 para indicar que estamos en el modo de seleccion de operacion.

D7	D6	D5	D4	D3	D2	D1	D0
M1	M2	x	x	1	1	1	1

MODO	M1	M0
Salida	0	0
Entrada	0	1
Bidireccional	1	0
Bit	1	1

Modo 3 solamente.

Cuando usamos el modo tres, tenemos que indicar que lineas se usaran como entrada y que lineas se usaran como salida. Para tal efecto, la siguiente palabra de control debera tener en los bits de salida un 0 y en los bits de entrada un 1.

3) Palabra de control de interrupciones.

Bits 3,2,1,0 — Los valores de estos bits nos indican que este byte es la palabra de control.

Bits 6,5,4 — Se usan para interrupciones en el modo bit; de otra manera no se toman en cuenta.

Bit 7 = 0 — Interrupcion deshabilitada.

Bit 7 = 1 — Interrupcion habilitada.

D7	D6	D5	D4	D3	D2	D1	D0
Habilita Interrupcion	And / Or	Alto / Bajo	Indicador Mascara	0	1	1	1

Solo se usan en modo 3

Modo 3 Solamente.

Bit D4 = 0 — La palabra siguiente no es la de mascara.

Bit D4 = 1 — La palabra siguiente es la de mascara.

Bit D5 = 0 — El nivel activo es el bajo.

Bit D5 = 1 — El nivel activo es el alto.

Bit D6 = 0 Interrumpe con una funcion OR.

Bit D7 = 1 Interrumpe con una funcion AND.

Para el modo 3 el bit D4 indica si la siguiente palabra de control es una mascarilla para las lineas de entrada seleccionadas. El bit D5 indica cual es el nivel activo para las entradas, ya sea nivel alto o bajo, y D6 indica que funcion logica se debe realizar con las lineas mascarilladas de entrada. Al tomar el valor verdadero esta funcion, se genera una interrupcion si el puerto esta habilitado para interrumpir.

El 'flip-flop' que habilita las interrupciones de un puerto, puede ser prendido o apagado modificando el resto de la palabra de interrupcion (no importa el modo de operacion) de la siguiente manera:

	D7	D6	D5	D4	D3	D2	D1	D0
Habilitacion Interrupciones	x	x	x	0	0	1	1	1

IMPORTANTE.

- Cuando se trabaje un puerto con capacidad de interrupcion el procesador debe ser programado en el modo de interrupciones 2.

- En el Starter-Kit las direcciones de los puertos del PIO son:

80H	Datos de puerto A.
81H	Datos de puerto B.
82H	Control del puerto A.
83H	Control del puerto B.

DESARROLLO.

El instructor explicara el funcionamiento del Z-80 PIO y dara un ejemplo de su programacion. Los alumnos ensamblaran un programa en la Microcomputadora Cromemco y lo ejecutaran en el Z-80 Starter Kit.

```
                        ; PROGRAMA QUE CONVIERTE CODIGO
                        ;BCD A CODIGO GRAY.
                        ;USANDO EL PIO

        ;***** PROGRAMA PRINCIPAL *****

TABLA:  EQU 2050H
GRAY:   EQU 2060H
RUTSER: EQU 2130H
        ORG 21C0H
        DB 24H
        DB 20H
        ORG 2000H

        LD A,21H        ;Se programa el registro de in-
        LD I,A          ;terrupcion del UCP.
        IM2             ;Se programa al UCP en modo de
        LD A,4FH        ;interrupcion 2.
        OUT (82H),A     ;Programacion del PIO:
        LD A,80H        ;Se programa al puerto A como
        OUT (82H),A     ;entrada con un vector de in--
        LD A,83H        ;terrupcion de 00 y con capa--
        OUT (82H),A     ;cidad de interrumpir al UCP.
        LD A,0FH        ;El puerto B sera de salida y
        OUT (83H),A     ;sin poder interrumpir al UCP.
        LD A,03H
        OUT (83H),A
        EI              ;Se habilitan interrupciones
        JR LOOP         ;y mientras no ocurra alguna
        HALT            ;interrupcion el UCP se en-
LOOP:   NOP             ;cuentra en un loop de NO ope-
                        ;raciones.

        ;*** RUTINA DE SERVICIO DE INTERRUPCIONES ***

        ORG 2024H
        DI              ;Se deshabilitan las interrup-
        LD IX,TABLA     ;ciones y se inician apuntador-
        LD IY,GRAY      ;res.Tabla apunta a un bloque
        IN A,(80H)      ;de memoria en donde se hallan
        AND A,0FH       ;los codigos en BCD y GRAY al
COMP:   CP A,(IX+0)     ;codigo correspondiente en --
        JR Z,REQCAR     ;GRAY.Se introducen los datos
        INC IX          ;por el puerto A ,se mascarilla
        INC IY          ;la parte alta del acumulador
        JR COMP         ;y se compara con la tabla de
REQCAR: LD A,(IX+0)     ;BCD.Para saber que numero se
        AND A,0FH       ;introdujo,se compara con el
        OUT (81H),A     ;contenido de la memoria apun-
        EI              ;tada por IX,si la comparacion
        RETI            ;es verdadera,se va a REQCAR y
```

```
                ;si no se incrementan los apun-
                ;tadores y se compara nuevamen-
                ;te hasta hallar el dato.IY a-
                ;puntara al codigo en GRAY del
                ;acumulador se carga con este
                ;dato y se envia al puerto de
                ;salida,se habilitan interrup-
                ;ciones .

;*** Fin de la rutina de servicio de interrup-
;    ciones ***

END
```

PRACTICA # 4

CIRCUITO INTEGRADO CONTADOR Y TEMPORIZADOR

(COUNTER TIMER CIRCUIT) CTC-Z80

OBJETIVO.

El alumno aprendera la arquitectura y el funcionamiento de este circuito de la familia Zilog .

INTRODUCCION.

Este circuito es uno de los mas usados en esta familia y erta presente en el Starter-Kit . El C.I. circuito contador y de temporizacion Z80-CTC es un circuito programable de 4 canales que provee funciones de conteo y temporizacion para el CPU-Z80 . El CPU configura los cuatro canales independientes del CTC para operar bajo varios modos y condiciones requeridas .

CARACTERISTICAS.

- Cada canal puede operar en modo contador o modo temporizador.

- Interrupciones programables sobre estados de conteo o de temporizacion.

- Un registro de constante de tiempo que recarga el contador a cero (contador hacia abajo) y el ciclo se repite.

- El contador hacia abajo se puede leer e indica el numero de la cuenta que va a cero.

- Posee una preescala que puede ser de 16 o de 256 , que divide el reloj del sistema para cada canal que opere en modo de temporizacion.

- Se puede seleccionar el disparo (positivo o negativo) del reloj que pone a funcionar el temporizador.

- Tres de los canales tienen salida Cuenta a cero/ Tiempo final capaces de manejar transistores Darlington.

- Se puede usar en una cadena margarita (daisy chain) de interrupciones con prioridades , con interrupcion vectorizada sin logica externa .

- Todas las entradas y salidas son totalmente compatibles con niveles T.T.L.

ARQUITECTURA.

Un diagrama de bloques se muestra en la pagina siguiente. La estructura interna del Z80-CTC consiste de un canal ? comunica con el canal de datos del Z80-CPU. Posee logica de control interna, cuatro canales contadores y logica de control de interrupcion. Cada canal tiene un vector de interrupcion para interrupcion vectorizada automatica, y la prioridad de la interrupcion es determinada por el numero del canal; el canal con la mas alta prioridad es el canal 0.

La estructura de canal tambien se muestra en la siguiente pagina. Posee 2 registros, 2 contadores, y logica de control. Los registros son: un registro de constante de tiempo (8 bits), un registro de control del canal (8 bits). Los contadores, son: un contador hacia abajo (contador a cero) que puede ser leido y, un registro de preescala o preescalador, este puede ser programado para dividir el reloj del sistema entre 16 o 256 .

Registro de la constante de tiempo (8 bits). Cargado por el CPU para iniciar y recargar el contador hacia abajo al llegar a cero.

Registro de control de canal (8 bits). Cargado por el CPU para seleccionar el modo y las condiciones de operacion del canal.

Contador hacia abajo o contador a cero (8 bits). Cargado por el registro de la constante de tiempo bajo control del programa y automaticamente al llegar a cero. En cualquier tiempo el CPU puede leer el numero de la cuenta que falta para ce . E contador es decrementado por la preescala en modo de temporizacion y por el CLK/TRIG en modo contador.

Preescalador (8 bits). Divide el reloj del sistema por 16 o 256 para decrementar el contador hacia abajo. Es usado solamente en modo de temporizacion.

CLK/TRIG = Disparador externo del reloj / temporizador.

= Clock external or timer trigger.

ZC / TO = Cuenta a cero o tiempo final.

= Zero count or timeout.

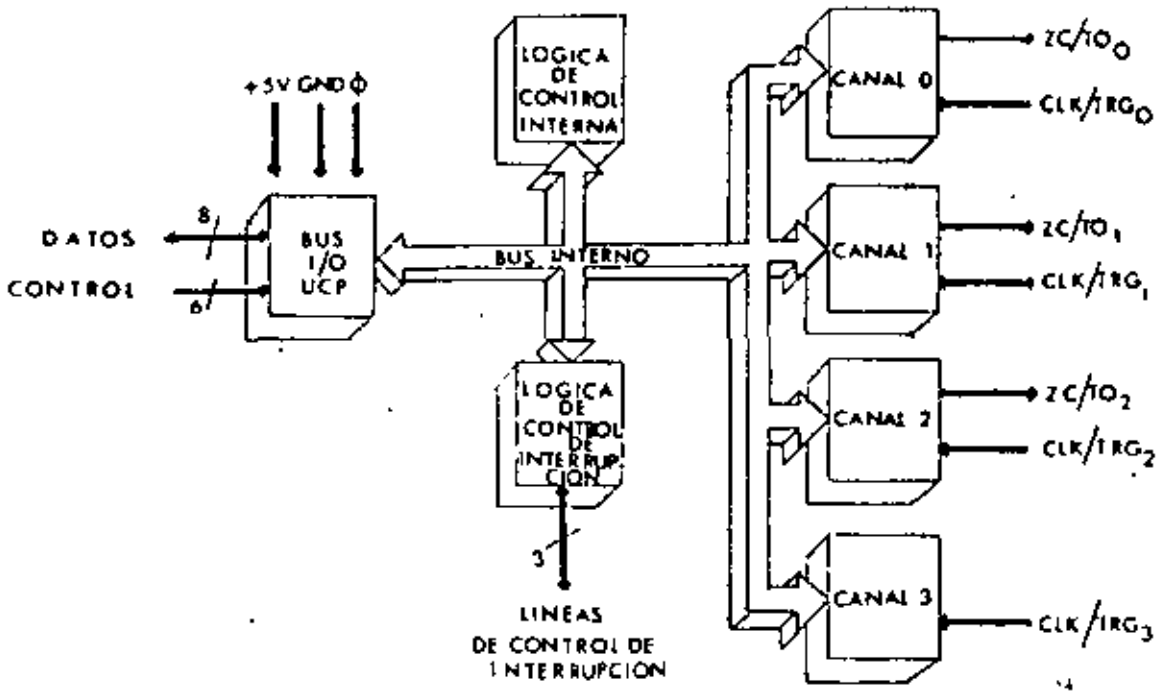


DIAGRAMA DE BLOQUES DEL C1C

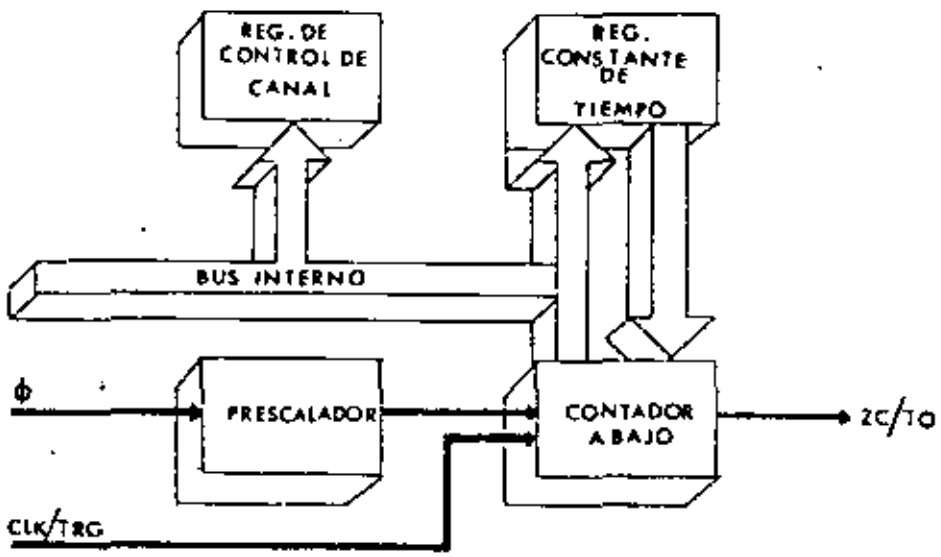
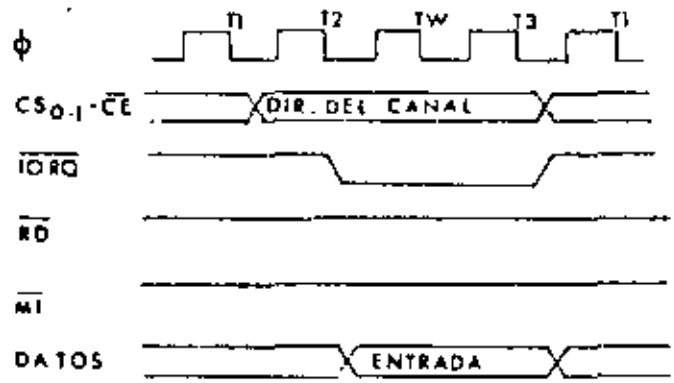


DIAGRAMA DE BLOQUES DE UN CANAL

CICLOS DE TIEMPO DEL CTC-Z80.

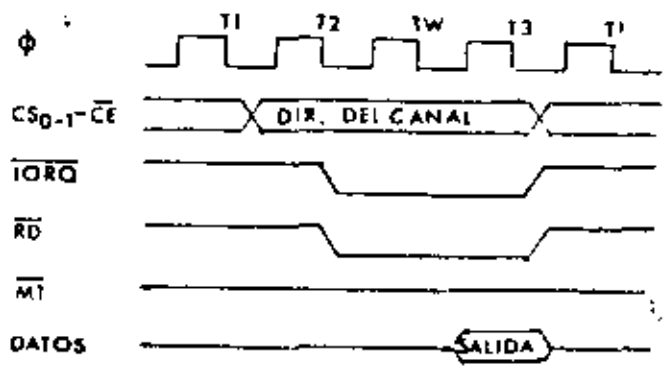
CICLO DE ESCRITURA.

Nos muestra el tiempo para cargar una palabra de control, la constante de tiempo y el vector de interrupcion. No se involucran estados de espera (WAIT STATES) para la escritura al CTC mas que el que se inserta automaticamente (Tw). Ya que el CTC no recibe una senal especifica de escritura, esta se genera internamente.



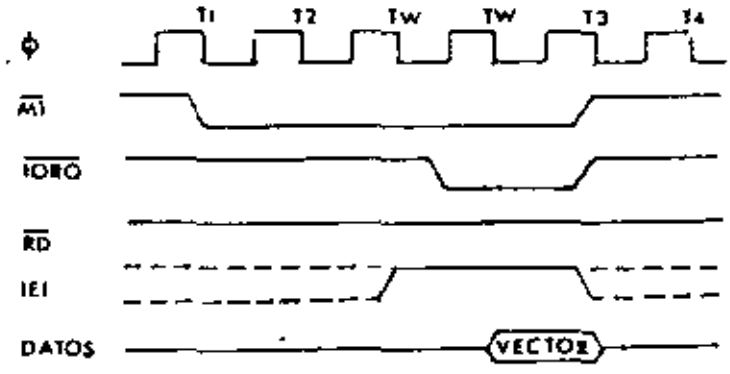
CICLO DE LECTURA.

Se muestran los tiempos que se requieren para leer el contador abajo cuando se esta en modo contador. El valor leido sobre el canal de datos refleja el numero de frentes de subida del reloj externo antes del frente de subida del ciclo T2. Unicamente se utiliza el ciclo de espera (Tw) mostrado.



CICLO DE RECONOCIMIENTO A UNA INTERRUPCION.

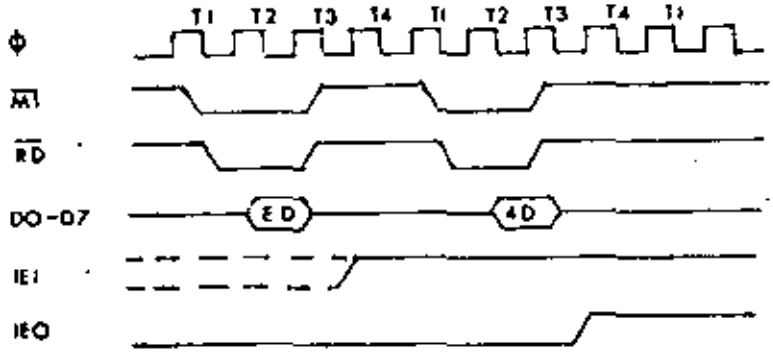
Algun tiempo despues que una interrupcion es solicitada por el CTC, el procesador enviara un reconocimiento de interrupcion (M1 y IORQ). Durante este tiempo la logica de interrupcion del CTC determinara cual es el canal de mas alta prioridad que este solicitando una interrupcion. Para asegurarse que las lineas de habilitacion de la cadena margarita estan estables, los canales son inhibidos a cambiar su estado de requerimiento de interrupcion cuando M1 esta activa. Si la entrada habilitadora de interrupcion (IEI) del CTC esta activa, entonces el canal de mas alta prioridad que esta interrumpiendo coloca su vector de interrupcion dentro del canal de datos cuando IORQ es activada. Se permiten ciclos adicionales de espera.



CICLO DE RETORNO DE UNA INTERRUPCION.

Si un periferico no tiene interrupcion pendiente y no esta bajo servicio de interrupcion entonces IEO=IEI. Si esta bajo servicio de interrupcion (ya interrumpio y recibio un reconocimiento de interrupcion) entonces IEO esta en nivel bajo, inhibiendo los C.I. de menor prioridad de interrupcion. Si esta pendiente de interrupcion, la cual todavia no ha sido reconocida, IEO estara baja a menos que un codigo de operacion ED sea decodificado como el primer byte de un codigo de operacion de 2 bytes. En este caso IEO ira a alto hasta que el byte del siguiente codigo de operacion sea decodificado, con lo cual nuevamente sera bajo. Si el segundo byte del codigo de operacion fue 4D entonces el codigo de operacion fue el de la instruccion RETI (RETorno de Interrupcion). Despues de que un codigo ED es decodificado, unicamente el periferico que interrumpio y que esta bajo servicio de interruipcion tendra su IEI alto y su IEO bajo. Este periferico posee la mas alta prioridad de la cadena margarita que haya recibido un reconocimiento de interrupcion. Los demas perifericos tendran IEI=IEO. Si el siguiente byte del codigo de operacion decodificado es 4D, este periferico presentara un estado de condicion bajo servicio.

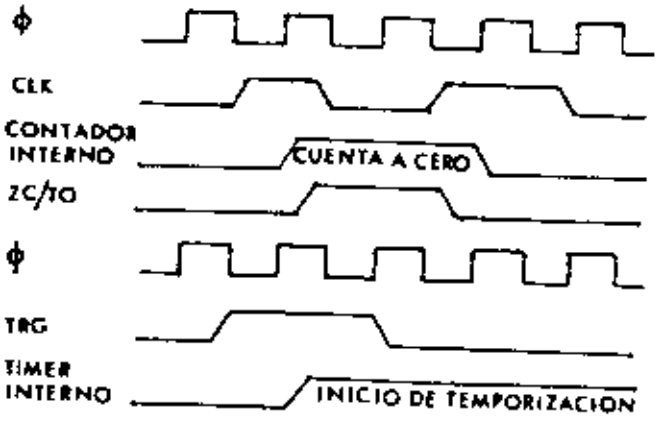
Ciclos de espera son permitidos en el ciclo M1 .



CICLOS DE CONTEO Y TEMPORIZACION .

En el modo contador el flanco de subida o de bajada de la entrada de reloj (CLK) origina que el contador sea decrementado. El frente es detectado asincronicamente y debe tener un ancho del pulso de reloj minimo. Sin embargo, el contador esta en sincronia con el reloj del sistema, por lo tanto debe ser encontrado un frente de subida del reloj del sistema para obtener el contador decrementado.

En el modo de temporizador el preescalador puede ser habilitado por un frente de subida o de bajada en la entrada de disparo (TRG INPUT). Como en el modo contador, el frente es detectado asincronamente y debe tener un ancho del pulso TRG minimo. Sin embargo, la temporizacion iniciara con el siguiente frente de subida del reloj del sistema. El preescalador cuenta los frentes de onda del reloj.



PROGRAMACION.

SELECCIONANDO UN MODO DE OPERACION.

Cuando se selecciona un modo de operacion de un canal, el bit 0 es ajustado a 1 para indicar que esta palabra debe ser almacenada en el registro de control del canal.

D7	D6	D5	D4	D3	D2	D1	D0
Interrupcion habilitada	modo	rango	frente	disparo	carga de constante de tiempo	RESET	1

Bit D7 = 0 Interrupcion del canal deshabilitada.

D7 = 1 Interrupcion del canal habilitada. Ocurrira cuando el contador hacia abajo (contador a cero) sea cero

Bit D6 = 0 Modo temporizador (TIMER) solamente . El contador hacia abajo es manejado por el reloj de la preescala . El periodo del contador es:

$$(t_c)(P)(CT)$$

t_c = Periodo del reloj del sistema

P= Preescala : 16 o 256

CT = Constante de tiempo de 8 bits programado (256 maximo)

D6 = 1 Modo contador . El contador hacia abajo es manejado por el reloj externo . La preescala no es usada.

Bit D5 = 0 Modo temporizador solamente . El reloj del sistema es dividido por 16 en la preescala.

D5 = 1 Modo temporizador solamente . El reloj del sistema es dividido por 256 en la preescala.

Bit D4 = 0 Modo temporizador . El frente negativo dispara (inicia) la operacion de temporizacion.

Modo contador. Frente negativo decrementa el contador hacia abajo .

D4 = 1 Modo temporizador . El frente de onda positivo inicia la operacion de tomporizacion.

Modo contador . El frente positivo decrementa el contador hacia abajo.

Bit D3 = 0 Modo temporizador solamente . Se inicia la temporizacion sobre el frente de subida del ciclo T2 del ciclo de maquina siguiente al que carga la constante de tiempo.

D3 = 1 Modo temporizador unicamente . Un disparo externo es valido para iniciar la operacion de temporizacion despues del frente de subida del ciclo T2 del ciclo de maquina siguiente al que carga la constante de tiempo . El preescalador es decrementado 2 ciclos de reloj mas tarde si el tiempo del sistema esta presente , de otra manera son 3 ciclos de reloj.

Bit D2 = 0 Indica que no se envia la constante de tiempo despues de la palabra de control . Una constante de tiempo debe ser escrita al canal para iniciar la operacion.

D2 = 1 La constante de tiempo para el contador a cero sera la siguiente palabra escrita al canal seleccionado . Si una constante de tiempo es cargada durante el conteo , la cuenta presente se completara antes de que la nueva constante de tiempo sea cargada al contador a cero.

Bit D1 = 0 El canal continua contando.

D1 = 1 Se detiene la operacion . Si el bit D2 es 1 el canal reasumira el conteo despues de haberse cargado una constante de tiempo de otra manera una nueva palabra de control debe ser cargada.

CARGANDO UNA CONSTANTE DE TIEMPO.

Una constante de tiempo de 8 bits es cargada al registro de la constante de tiempo , a continuacion de la palabra de control cuando el bit 2 de esta palabra es 1 . Cuando todos los bits son cero indican una constante de tiempo de 256.

D7	D6	D5	D4	D3	D2	D1	D0
bit7	bit6	bit5	bit4	bit3	bit2	bit1	bit0

CARGANDO UN VECTOR DE INTERRUPCION.

El procesador Z-80 requiere que un vector de interrupcion (8 bits) sea alimentado por el canal interruptor (cuando la interrupcion es modo 2 , como en nuestro caso) . El procesador forma la direccion de la rutina de servicio de interrupcion del canal usando este vector . Durante un ciclo de reconocimiento de interrupcion , el vector es colocado sobre el canal de datos del procesador por medio del canal de mas alta prioridad que esta requiriendo servicio en ese instante . El vector de interrupcion deseado es cargado al CTC , escribiendo al canal 0 , un 0 en D0 . D7 - D3 contienen el vector de interrupcion almacenado y , D2 y D1 no son usados al cargar el vector . Cuando el CTC responde a un reconocimiento d interrupcion , estos dos bits contienen el codigo binario del canal de mas alta prioridad que esta solicitando una interrupcion , y D0 contiene un cero puesto que la direccion de la rutina de servicio a la interrupcion inicia en un'byte par . El canal 0 es el canal de mas alta prioridad.

D7	D6	D5	D4	D3	D2	D1	D0
bit7	bit6	bit5	bit4	bit3	X	X	0

```
                ;***** PROGRAMA PRINCIPAL *****

                        ; Por medio de este programa , obte -
                        ;nemos un reloj , que nos muestra por
                        ;los displays , la hora , minutos y
                        ;segundos .

RESTAR: EQU 039FH
        ORG 2200H       ;El ensamblado del programa se lleva
        IM2             ;a cabo partir de la localidad 2000H
        LD A,20H        ;Se programa al CPU en modo de inte--
        LD I,A          ;rrupcion 2 ,se asigna el valor de 20H
        LD A,1AH        ; al registro de interrupcion del CPU
        OUT (84H),A     ; y  al vector de interrupciones del
        LD A,845H       ;con el valor de 1AH .Se habilita al
        OUT (85H),A     ;canal 1 para poder interrumpir
        LD A,0C8H       ;trabajando en modo timer ,con una pre-
        OUT (85H),A     ;escala de 256.La constante de tiempo
        EXX             ;es de 200 (C8H). CT X tc X P =
        LD B,39D        ;(200)(500 ns)(256)= 2.56 E -02 .Para
        EXX             ;que interrumpa aprox. cada segundo.Se
        EI              ;utiliza el registro B' como contador,
        JP RESTAR       ;por lo tanto 1 s / 2.56 E-2 = 39 aprox.
                        ;Se carga entonces este valor al registro
                        ;B' .
                        ;Finalmente se habilitan las interrupciones.

        ORG 201AH       ;En estas direccion se almacena
        DB 20H          ;la direccion de inicio de la rutina de
        DB 20H          ;servicio a la interrupcion.

                        ;En 201C se guarda el valor de segundos
                        ;en 201D el de los minutos
                        ;y en 201E el de las horas
                        ; * Fin al programa principal *

        ;* RUTINA DE SERVICIO A INTERRUPCIONES AL CANAL 1 *

        ORG 2020H
        EX AF,AF'       ; En esta rutina se decrementa el
        EXX             ;valor del registro B' ,si no es
                        ;cero ,entonces regresa a esperar
                        ;otra interrupcion.Si es cero en
        DJNZ,REGRES     ;tonces se incrementa el segundero
        LD B,27H        ;y se recarga el valor de B' nueva-
        LD A,(201CH)    ;mente ,se compara con 60 si es --
        INC A           ;menor entonces solo actualiza los
        DAA             ;segundos,si es 60 el segundero se
        CP 60H          ;hace 00 y se incrementa el minu--
        JR C,NO         ;tero.A su vez este se compara con
        XOR A           ;60 ,si es menor actualiza solo --
        LD (201CH),A    ;los minutos,si es mayor se incre-
        LD A,(201DH)    ;menta el contador de las horas.
        INC A           ;Este contador llega hasta  13 en
        DAA             ;donde se hace igual a 01 .
        CP 60H          ;incrementa el minutero hasta que
        JR C,MIN        ; se hace igual a 60, se incrementa
        XOR A           ;el contador de las horas y el minu--
        LD (201DH),A    ;tero se hace 00 y se incrementa el
        LD A,(201EH)    ;contador de las horas hasta que se
        INC A           ;hace 13,en donde el valor que se
        DAA             ;despliega es 010000 y se continua
        CP 13H          ;con la operacion.
        JR C,HORA
        LD A,01
        JR HORA
MIN:    LD (201DH),A
        LD A,(201CH)
        JR SEGU
HORA:   LD (201EH),A
        LD A,(201CH)
        JR SEGU
```

```
NO:     LD (201CH),A    ;Para el despliegue de la hora se
SEGU:   LD A,(201CH)    ;hace uso del monitor.Posee un --
        LD IX,23F8H     ;area de memoria disponible para
        CALL FMI        ;el despliegue por los displays.
        LD A,(201DH)    ;Los valores a desplegar se almace-
        LD IX,23F9H     ;nan en decimal,uno en cada locali-
        CALL FMI        ;dad de memoria. Esta area inica en
        LD A,(201EH)    ;la localidad 23F7H,ahi se almacena
        LD IX,23F7H     ;el digito a mostrar por el display
        CALL FMT        ;de la izquierda,el siguiente digito
REGRES: EX AF,AF'       ;se guarda en la localidad 23F8H --
        EXX             ;y asi sucesivamente hasta la di---
        EI              ;reccion 23FCH en donde se almacena
                        ;el digito a mostrar en el display
                        ;de la derecha.

        RETI            ;* Fin de la rutina de servicio a
                        ; la interrupcion *
FMT:    LD C,A
        LD (IX+1),A
        SRL C
        SRL C
        SRL C
        SRL C
        LD (IX+0),C
        RET
        END
```

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

INTRODUCCION A LOS MICROPROCESADORES

MODOS DE DIRECCIONAMIENTO

ING. LUIS CORDERO BORBOA

AGOSTO, 1984

IV.- MODOS DE DIRECCIONAMIENTO

1.- ESQUEMAS DE DIRECCIONAMIENTO.

La unidad central de proceso (CPU) en las computadoras debe realizar las siguientes funciones:

- Obtener y traer de memoria primaria al CPU la siguiente instrucción a ejecutar.
- Entender los operandos, esto es, definir la localización de los operandos necesarios para ejecutar la instrucción y traerlos al CPU.
- Ejecutar la instrucción.

Para llevar a cabo las funciones anteriores el CPU debe contar con la siguiente información:

- El código de operación de la instrucción a ejecutar.
- Las direcciones de los operandos y la del resultado.
- La dirección de la siguiente instrucción a ejecutar.

Existen diferentes soluciones que satisfacen los requerimientos anteriores, los cuales determinan la arquitectura de los procesadores que las utilizan.

Se supondrán operaciones aritméticas en las que se tienen dos operandos y un resultado ya que son las que proporcionan el caso más general.

a) Máquinas de "3+1" direcciones

El formato de instrucción en este esquema de direcciona-- miento contiene todos los elementos necesitados por el CPU

para realizar sus funciones.

Un posible formato de instrucción se muestra en la figura IV.1

CODIGO DE OPERAC.	DIRECCION PRIMER OPERANDO	DIRECCION SEGUNDO OPERANDO	DIRECCION RESULTADO	DIRECCION DE LA SIGUIENTE INSTRUCCION	Palabra de memoria

FIG. IV.1

En este caso se tienen cinco campos en el formato de instrucción: Uno para el código de operación que sirve para indicar el tipo de operación a realizar (suma, resta, multiplicación, etc.), tres campos para las direcciones de los operandos y resultado de las operaciones, un campo para indicar la dirección de la siguiente instrucción a ejecutar.

Las instrucciones para ésta máquina podrían ser escritas en forma simbólica en la siguiente forma: ADD A, B, C, D donde ADD representa el código de operación suma y A, B, C y D son nombres simbólicos asignados a localidades de memoria.

Suponiendo que existen las instrucciones suma (ADD), substracción (SUB) y multiplicación (MUL), entonces una posible traducción de la expresión A=(B*C)-(D*E) en FORTRAN a lenguaje simbólico en la máquina de 3+1 direcciones sería:

L1: MUL B, C, T1, L3

L3: MUL D, E, T2, L7

L7: SUB T2, T1, A, L8

L8: Siguiente instrucción

donde T1 y T2 representan localidades temporales usadas para guardar resultados aritméticos intermedios.

Las conclusiones más importantes en este esquema son:
Los programas no necesitan estar almacenados en memoria en forma secuencial ya que el campo de dirección de la siguiente instrucción permite conocer donde fueron almacenados.

Debido a que cada instrucción contiene en forma explícita tres direcciones, no es necesario tener en el CPU hardware para guardar los resultados de las operaciones.

b) Máquinas de "3" direcciones

Considerando que los programas se escriben secuencialmente y que por consiguiente es muy lógico almacenarlos en este mismo orden, se llega a un nuevo esquema de direccionamiento en el cual se sustituyen todos los campos de dirección de la siguiente instrucción por un solo registro dentro del procesador que lleva en forma secuencial y automáticamente la dirección de la siguiente instrucción a ejecutar. Un posible formato de instrucción se muestra en la fig. IV.2 .

Dirección de la sig. inst.	Registro en el procesador	Código de operac.	Dirección primer operando	Dirección segundo operando	Dirección resultado	Palabra n de memoria

FIG. IV.2

Utilizando este esquema de direccionamiento la expresión A=(B*C)-(D*E) en FORTRAN, quedaría expresada como:

```
MUL   B, C, T1
MUL   D, E, T2
SUB   T2, T1, A
Siguiente instrucción
```

Donde se ha suprimido la dirección de la siguiente instrucción ya que ésta es llevada en forma secuencial y automática por un registro del procesador conocido como contador del programa (PC).

Con el esquema de 3 direcciones se logra aprovechar la memoria en forma más eficiente y reducir la longitud de palabra lo que redunda directamente en los costos de la misma.

c) Máquinas de "2" direcciones.

En las operaciones aritméticas no siempre es necesario guardar el resultado en una localidad de memoria y preservar los operandos, por lo que se puede pensar en utilizar uno de ellos para guardar el resultado una vez que la operación se ha efectuado. Las consideraciones anteriores llevan a presentar un posible formato de instrucción en esta máquina, mostrado en la figura IV.3

DIR. DE LA SIG. INST. A EJECUTAR	REG. EN EL PROC.	COD. OP.	DIR. P. OP.	DIR. SEG. OP.	Palabra n de memoria

FIG. IV.3

En este esquema se usará la dirección del segundo operando como la dirección del resultado una vez que la operación se haya efectuado, por lo que el segundo operando será destruído. Así pues la expresión A=(B*C)-(D*E) en FORTRAN, quedaría:

```
MUL   B, C
MUL   D, E
SUB   E, C
ADD   C, A
```

La eliminación del campo de dirección del resultado permite reducir la longitud de la palabra de memoria y los costos de la misma, lo que permite usar este esquema en máquinas medianas y chicas.

d) Máquinas de "1" dirección

Este esquema de direccionamiento permite eliminar de todas las instrucciones el campo de dirección de uno de los operando y sustituirlo por un registro dentro del procesador, el cual contendrá a uno de los operandos. A este registro se le conoce como acumulador. - El formato de instrucción para la máquina de 1 dirección se muestra en la figura IV 4

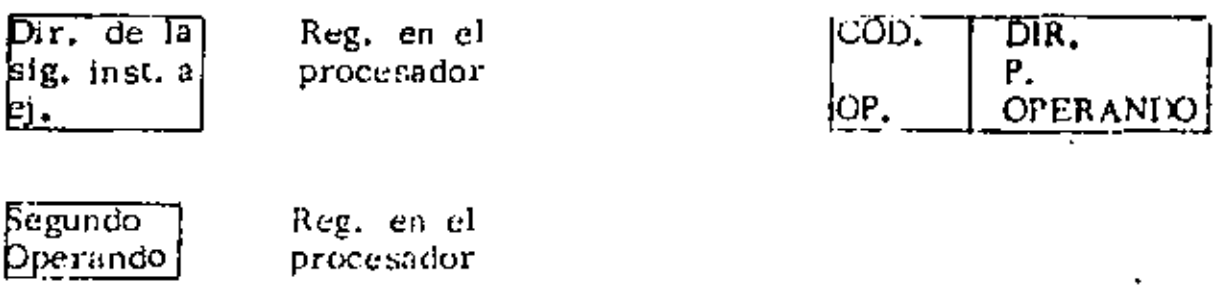


FIG. IV.4

Lo anterior implica la creación de instrucciones que permitan cargar el acumulador con el segundo operando (LAC) y depositar el contenido del acumulador en memoria (DAC).

Es importante hacer notar que todas las operaciones se llevan a cabo implícitamente contra el acumulador y que éste contendrá el resultado de la operación efectuada. La expresión A=(B*C)-(D*E) en FORTRAN, podría traducirse a:

```
LAC   D
MUL   E
DAC   T1
LAC   B
MUL   C
SUB   T1
DAC   A
```

Este esquema de direccionamiento ha sido ampliamente implementado en una gran mayoría de las minicomputadoras, como por ejemplo: PDP-8, -- PDP-15, IBM-1130, IBM-7090 y CDC 3600.

e) Máquinas de "0" direcciones

Este esquema de direccionamiento solo utiliza el campo de código de operación, por lo que es necesario contar con algún mecanismo que implícitamente permita conocer los operandos.

El mecanismo anterior se implementa usando una pila ó stack, el cual se puede pensar como un conjunto de localidades contiguas de

memoria accesadas usando una disciplina UEPS (últimas entradas, primeras salidas). De lo anterior se concluye que en cada momento se tendrá disponible el elemento que se encuentre en el tope del stack.

El formato de instrucción para este esquema de direccionamiento se encuentra en la figura IV.5

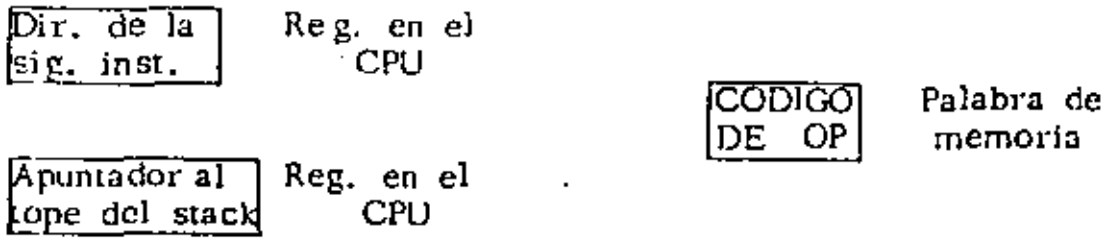
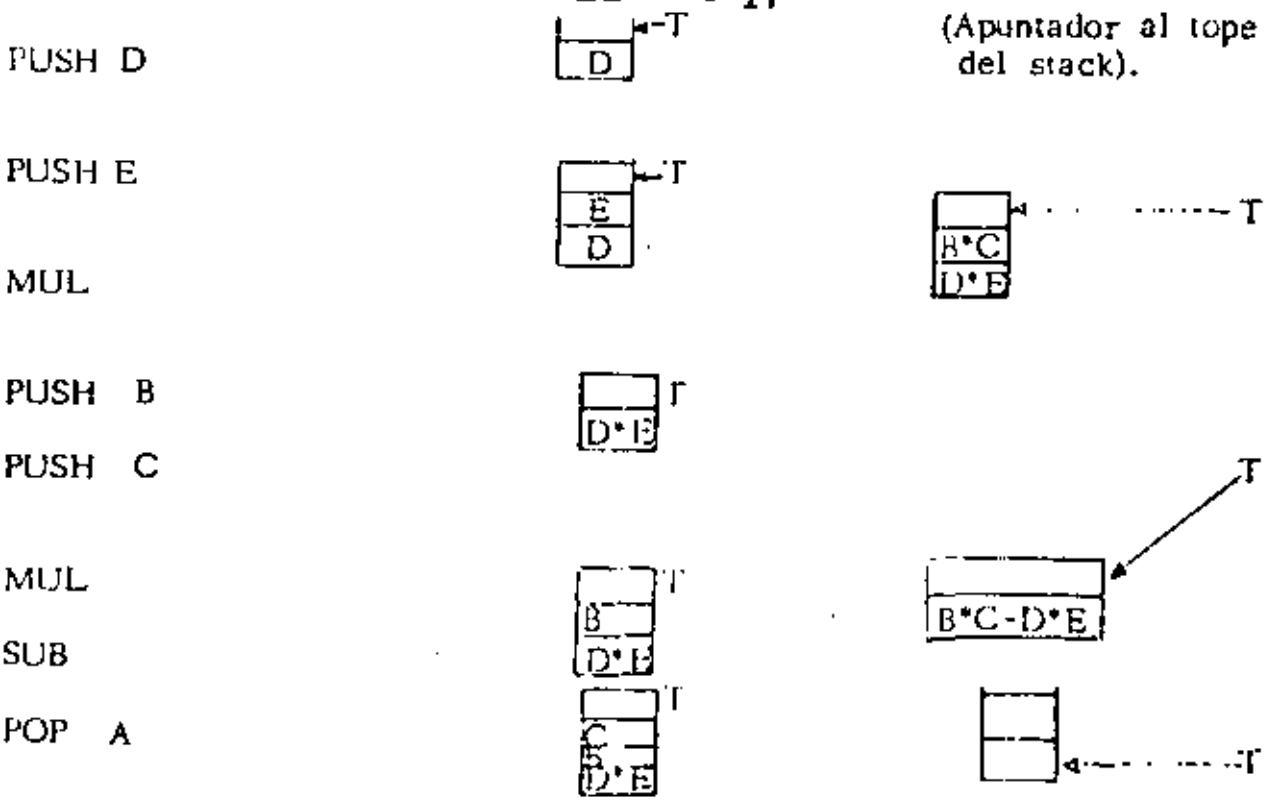


FIG. IV.5

Es necesario contar con instrucciones que permitan meter elementos de memoria al stack (PUSH) y sacar elementos del stack a memoria (POP).

La expresión A=(B*C)-(D*E) en FORTRAN, podría expresarse como:

FALTA IV.6



En la fig. IV.6 se ilustra el estado del stack después de cada una de las inst. anteriores.

Se puede concluir que el conjunto de instrucciones de la máquina no está formado solamente por instrucciones de cero direcciones ya que también se requieren instrucciones de una dirección para meter y sacar elementos al stack.

Se requiere un registro en el procesador que apunte al tope del stack y se elimine el acumulador ya que el resultado de las operaciones también quedará en el stack.

2.- METODOS DE DIRECCIONAMIENTO

En las máquinas de una sola dirección el formato de las instrucciones que hace referencia a memoria consta de dos campos: el campo de código de operación y el campo de dirección del operando. Si suponemos que el campo de dirección consta de n bits, entonces la máxima capacidad de memoria direccionable será 2^n localidades. Lo anterior puede resultar bastante drástico en el caso de las minicomputadoras ya que, por lo general tienen palabras de 12 ó 16 bits y si se asignan cuatro de ellos al campo de código de operación solo se pueden direccionar $2^8 = 256$ localidades de memoria en el caso de palabras de 12 bits ó $2^{12} = 4096$ localidades de memoria en el caso de palabras de 16 bits, lo cual resulta insuficiente para la gran mayoría de las aplicaciones.

Lo anterior ha ocasionado diferentes modos de direccionamiento, en los cuales el campo de dirección sirve para calcular la dirección efectiva del operando, logrando una mayor capacidad de memoria direccionable.

a) Inmediato

En este caso el operando puede estar contenido directamente en el campo de dirección ó en la localidad de memoria siguiente a la instrucción.

Será necesario dedicar un bit de la palabra para saber como se debe interpretar la instrucción.

b) Directo

Existe direccionamiento directo cuando el campo de dirección de la instrucción contiene la dirección del operando ó cuando éste campo combinado con algún registro ó palabra de memoria generan la dirección del operando.

b.1) Usando página cero

Uno de los esquemas más comunes de organización de memoria, divide ésta en n páginas de longitud fija, donde n dependerá del tamaño de la memoria y del tamaño de las páginas.

Las máquinas que usan estos esquemas generalmente usan la página cero con propósitos especiales, como son: manejo de interrupciones, traps, localidades autoincrementables, etc.

La forma de indicar si el contenido del campo de dirección se refiere a la página cero, es usando un bit para este propósito, p. ej. si este bit es cero el campo de dirección apunta a una localidad en la página cero.

b.2) Usando página actual

Si el bit de página está en uno, se asume que el campo de dirección apunta a una localidad en la página en la que se encuentra la instrucción. A esta página se le conóce como

página actual.

La dirección del operando se determina sumando los bits de orden superior del PC al campo de dirección de la instrucción.

b.3) Relativo al PC

En este modo de direccionamiento el contenido del campo de dirección de la instrucción, interpretado como un entero con signo, se suma al PC para obtener la dirección del operando.

b.4) Relativo a un registro índice

El contenido del campo de dirección de la instrucción, interpretado como un entero con signo, se suma al contenido de un registro índice para obtener la dirección del operando. En caso de existir más de un registro índice es preciso asignar los bits necesarios para su identificación.

c) Indirecto

En el direccionamiento indirecto el campo de dirección de la instrucción contiene un apuntador a la dirección del operando ó este campo combinado con algún registro ó palabra de memoria genera un apuntador a la dirección del operando.

Mediante un bit en la instrucción se puede saber si el direccionamiento usado es directo ó indirecto.

c.1) Usando página cero

El campo de dirección de la instrucción apunta a una localidad en la página cero. A su vez ésta localidad contiene la dirección del operando.

c.2) Usando página actual

El campo de dirección de la instrucción apunta a una localidad en la página actual. Esta localidad contiene la dirección del operando.

c.3) Relativo al PC

El contenido del campo de dirección de la instrucción, interpretado como un entero con signo, se suma al PC para obtener la dirección del apuntador al operando.

c.4) Relativo a un registro índice

El contenido del campo de dirección de la instrucción, interpretado como un entero con signo, se suma al contenido de un registro índice para obtener la dirección del apuntador al operando.

La combinación de todos los métodos de direccionamiento anteriores con registros de propósito general, permiten lograr modos de direccionamiento bastante poderosos. Cuando se usan los registros de propósito general, el campo de dirección de la instrucción especifica que registro se usa y como se interpreta la información que contiene.

3.- D I R E C C I O N A M I E N T O E N Z - 80

El microprocesador Z-80 es una máquina de una dirección en la que los diferentes modos de direccionamiento son usados por grupos de instrucciones y no se aplican de una forma general a todo el conjunto de instrucciones.

a) Implícito

En este modo de direccionamiento el operando no se define en forma explícita ya que el formato de instrucción es fijo y en los códigos de operación se especifica implícitamente sobre que registros del procesador actúan las instrucciones, por lo que el usuario no puede alterarlo de ninguna manera.

Los grupos de instrucciones, que utilizan este modo de direccionamiento son: carga de 8 bits; carga de 16 bits; intercambio, transferencia de bloques y búsqueda; aritméticas de propósito general y control del CPU.

Ejemplos 1.

b) Inmediato

El operando se encuentra en la localidad de memoria siguiente a la instrucción y se considera que forma parte de la misma. Los valores de los operandos inmediatos en ningun caso podrán exceder la capacidad de representación de un byte. Este modo de direccionamiento se utiliza cuando se desean realizar operaciones con valores constantes.

Los grupos de instrucciones que utilizan este modo de direccionamiento son: carga de 8 bits; aritméticas y lógicas de 8 bits y entrada/salida.

Ejemplos 2.

c) Inmediato extendido

El operando se encuentra en los dos bytes (16 bits) siguientes al código de operación de la instrucción. El primer byte contiguo al código de operación es el menos significativo y el siguiente es el más significativo.

Este modo de direccionamiento es usado por algunas instrucciones de carga de 16 bits.

Ejemplos 3.

d) Registro

El formato de instrucción contiene un campo de dirección de operando donde se especifica cual de los registros del CPU será utilizado como operando.

Los grupos de instrucciones que utilizan este modo de direccionamiento son: carga de 8 bits; carga de 16 bits; aritméticas y lógicas de 8 bits; aritméticas y lógicas de 16 bits; rotaciones y desplazamientos; encendido y apagado de bits; entrada/salida.

Ejemplos 4.

e) Registro indirecto

En este modo de direccionamiento un par de registros (16 bits) contiene la dirección de memoria en la que se encuentra el operando.

Es utilizado por los grupos de instrucciones de carga de 8 bits; intercambio, transferencia de bloques y búsqueda; rotaciones y desplazamientos; prendido y apagado de bits; saltos, llamadas y regreso de subrutinas; entrada/salida.

Ejemplos 5.

f) Extendido

La dirección del operando está contenida dentro del campo de operando de la instrucción. El campo de dirección tiene una longitud de 16 bits por lo que la máxima capacidad de memoria direccionable es de 64 K bytes.

Este modo de direccionamiento es utilizado por los grupos de instrucciones de carga de 8 bits; carga de 16 bits; saltos, llamadas y regreso de subrutinas.

Ejemplos 6.

g) Modificado de página cero

En este modo de direccionamiento el campo de dirección del operando se refiere a una localidad de memoria dentro de la página cero. Este campo de dirección consta de 3 bits y para su correcta interpretación se multiplica por 08H, obteniéndose de esta forma la referencia a las localidades deseadas.

Este modo de direccionamiento se utiliza exclusivamente por la instrucción RST.

Ejemplos 7.

h) Relativo

La dirección del operando se determina sumando al contador del programa el contenido del byte siguiente al código de operación de la instrucción.

El desplazamiento anterior se interpretará como un número en complemento a dos, con lo que se logra un rango de direccionamiento de -126 a +129 localidades relativas al contador del programa.

Este modo de direccionamiento es usado por el grupo de instrucciones de salto, llamada y regreso de subrutinas.

Ejemplos 8.

i) Indexado

La dirección del operando se determina sumando al registro de índice especificado el contenido del byte de desplazamiento.

El desplazamiento se interpreta como una cantidad en complemento a dos, con lo que se logra un rango de direccionamiento de -128 a +127 localidades relativas al registro de índice.

Los grupos de instrucciones que utilizan este modo de direccionamiento son: carga de 8 bits; aritméticas y lógicas de 8 bits; rotaciones y desplazamientos; encendido y apagado de bits; saltos, llamada y regreso de subrutinas.

Ejemplos 9.

j) Bit

Este modo de direccionamiento permite prender o apagar un bit dentro de un operando seleccionado, usando los modos antes descritos.

Ejemplos 10.

E J E M P L O S

Se asumirá que todos los ejemplos siguientes utilizan el sistema de numeración hexadecimal.

Ejemplos 1.

```
                    ; MODOS DE DIRECCIONAMIENTO DEL MICROPROCESADOR Z-80
                    ; PROGRAMA CARGADO EN CASSETE CON EL NOMBRE DE -
                    ; "EJE"
                    ;
                    ;
                    ; DIRECCIONAMIENTO IMPLICITO.
                    ;
0000 ED5F                   LD      A,R
                    ;
                    ; CARGA EN EL REGISTRO A EL CONTENIDO DEL REGISTRO
                    ; DE REFRESCAMIENTO R.
                    ;
                    ;
0002 2F                     CPL
                    ;
                    ; REALIZA EL COMPLEMENTO LOGICO DEL CONTENIDO DEL
                    ; ACUMULADOR Y LO DEJA EN EL MISMO REGISTRO
                    ;
                    ;
0003 DD23                   INC     IX
                    ;
                    ; EL CONTENIDO DEL REGISTRO DE INDICE IX SE IN--
                    ; CREMENTA EN UNO.
                    ;
                    ;*
```

Ejemplos 2.

```
                    ;
                    ; DIRECCIONAMIENTO INMEDIATO.
                    ;
0005 C634                   ADD     A,34H
                    ;
                    ; SUMA AL CONTENIDO DEL REGISTRO ACUMULADOR A, EL
                    ; DATO 34H Y DEJA EL RESULTADO EN EL MISMO RE--
                    ; GISTRO.
                    ;
                    ;
0007 E610                   AND     10H
                    ;
                    ; REALIZA LA OPERACION LOGICA AND ENTRE EL CONTE--
                    ; NIDO DEL REGISTRO A Y EL DATO 10H. DEJANDO EL -
                    ; RESULTADO EN EL MISMO REGISTRO
                    ;
                    ;
```

Ejemplos 3.

```
                ; DIRECCIONAMIENTO  INMEDIATO  EXTENDIDO
                ;
0009 FD212030           LD      IY,2030H
                ;
                ; CARGA  EN  EL  REGISTRO  DE  INDICE  IY EL DATO 2030H
                ;
                ;
000D 213F12             LD      HL,123FH
                ;
                ; CARGA  EL  REGISTRO  PAR  HL  CON  EL  DATO  123FH.
                ;
                ;
```

Ejemplos 4.

```
                ;
                ; DIRECCIONAMIENTO  DE  REGISTRO
                ;
0010 4F                 LD      C,A
                ;
                ; CARGA  EL  REGISTRO  C  CON  EL  CONTENIDO  DEL  REGIS-
                ; TRO  A.
                ;
                ;
0011 80                 ADD     A,B
                ;
                ; SUMA  AL  CONTENIDO  DEL  REGISTRO  A  EL  CONTENIDO  -
                ; DEL  REGISTRO  B  Y  DEJA  EL  RESULTADO  EN  EL REGIS-
                ; TRO  A.
                ;
                ;
0012 ED52               SBC     HL,DE
                ;
                ; SUBSTRAE  DEL  CONTENIDO  DEL  REGISTRO  HL, EL  CONTE-
                ; NIDO  DE  LOS  REGISTROS  DE  Y  ACARREO  CY.  DEJANDO
                ; EL  RESULTADO  EN  EL  REGISTRO  HL.
                ;
                ;
```

Ejemplos 5.

```
                        ;
                        ; DIRECCIONAMIENTO DE REGISTRO INDIRECTO.
                        ;
0014 0A                         LD      A,(BC)
                        ;
                        ; CARGA EL REGISTRO A CON EL CONTENIDO DE LA LO-
                        ; CALIDAD DE MEMORIA APUNTADA POR EL REGISTRO PAR
                        ; BC.
                        ;
                        ;
0015 34                         INC     (HL)
                        ;
                        ; INCREMENTA EN UNO EL CONTENIDO DE LA LOCALIDAD
                        ; DE MEMORIA APUNTADA POR EL REGISTRO PAR HL.
                        ;
                        ;
0016 12                         LD      (DE),A
                        ;
                        ; DEPOSITA EL CONTENIDO DEL ACUMULADOR EN LA LOCA-
                        ; LIDAD DE MEMORIA APUNTADA POR EL REGISTRO PAR DE.
                        ;
                        ;
```

Ejemplos 6.

```
                        ;
                        ; DIRECCIONAMIENTO EXTENDIDO
                        ;
0017 3A2010                     LD      A,(1020H)
                        ;
                        ; CARGA EL ACUMULADOR CON EL CONTENIDO DE LA LOCA-
                        ; LIDAD DE MEMORIA 1020H.
                        ;
                        ;
001A FD220400                   LD      (0004H),IY
                        ;
                        ; DEPOSITA EL CONTENIDO DEL REGISTRO DE INDICE EN
                        ; LAS LOCALIDADES DE MEMORIA 0004H (BYTE BAJO) Y
                        ; 0005H (BYTE ALTO).
                        ;
                        ;
```

Ejemplos 7.

```
                ;
                ; DIRECCIONAMIENTO MODIFICADO DE PAGINA CERO
                ;
001E CF                 RST     08H
                ;
                ; EFECTUA UN SALTO INCONDICIONAL A LA LOCALIDAD DE
                ; MEMORIA 08H DESPUES DE HABER GUARDADO EN EL --
                ; STACK EL CONTENIDO DEL CONTADOR DEL PROGRAMA.
                ;
                ;
```

Ejemplos 8.

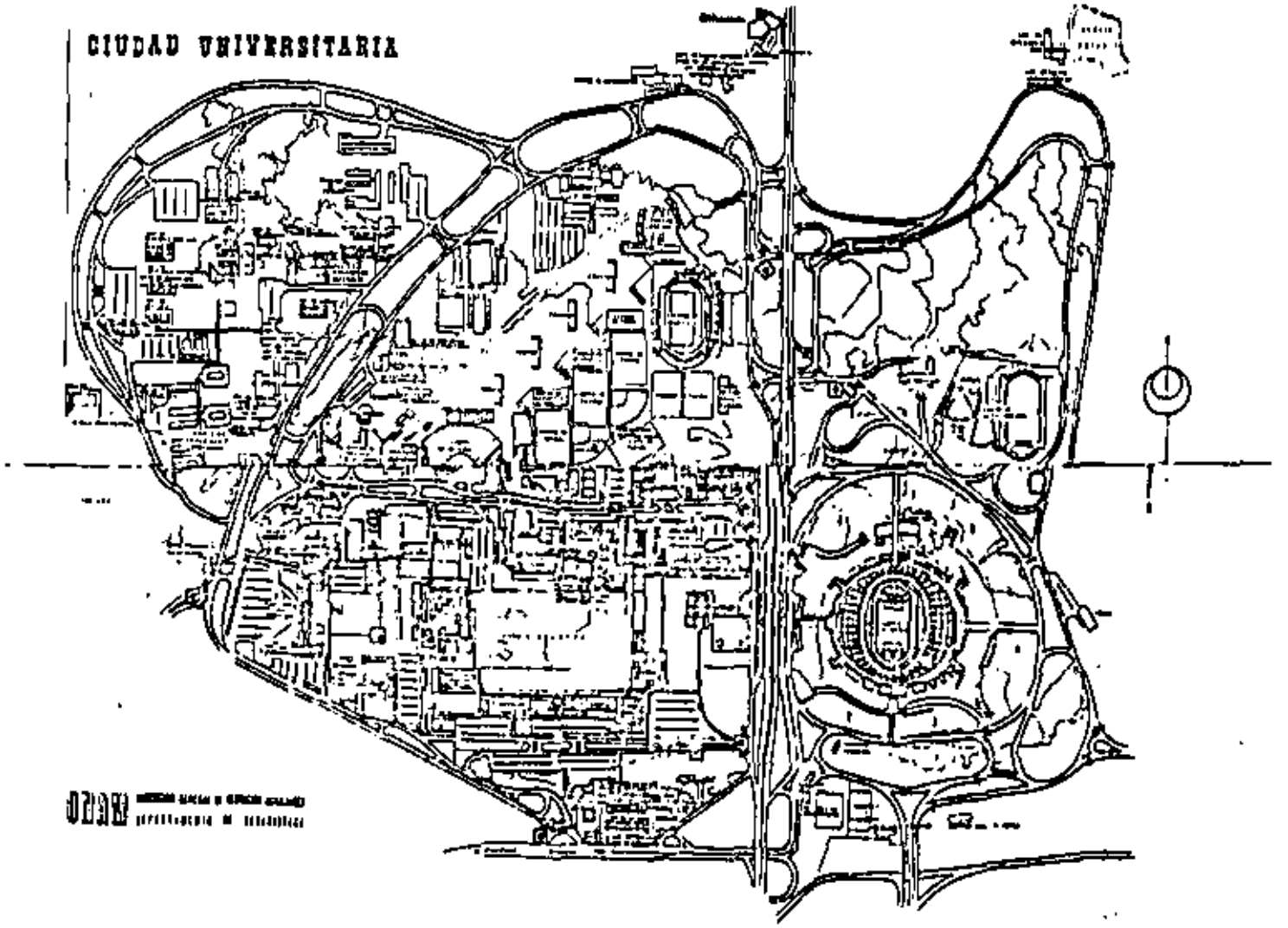
```
                ;
                ; DIRECCIONAMIENTO RELATIVO
                ;
001F 2804               JR      Z,25H
                ;
                ; SI LA BANDERA Z=1, AL CONTADOR DEL PROGRAMA SE LE
                ; SUMA EL VALOR 04H CON LO QUE SE EFECTUARA UN SAL-
                ; TO A LA LOCALIDAD DE MEMORIA 25H.
                ; SI LA BANDERA Z=0 SE CONTINUARA EJECUTANDO LA SI-
                ; GUIENTE INSTRUCCION DEL PROGRAMA.
                ;
                ;
0021 30F4               JR      NC,17H
                ;
                ;SI LA BANDERA C=0, AL CONTADOR DEL PROGRAMA SE LE
                ; SUMA EL VALOR F4H CON LO QUE SE EFECTUARA UN SAL-
                ; TO A LA LOCALIDAD DE MEMORIA 17H
                ; SI LA BANDERA C=1 SE CONTINUARA EJECUTANDO LA -
                ; SIGUIENTE INSTRUCCION DEL PROGRAMA
                ;
                ;L
```

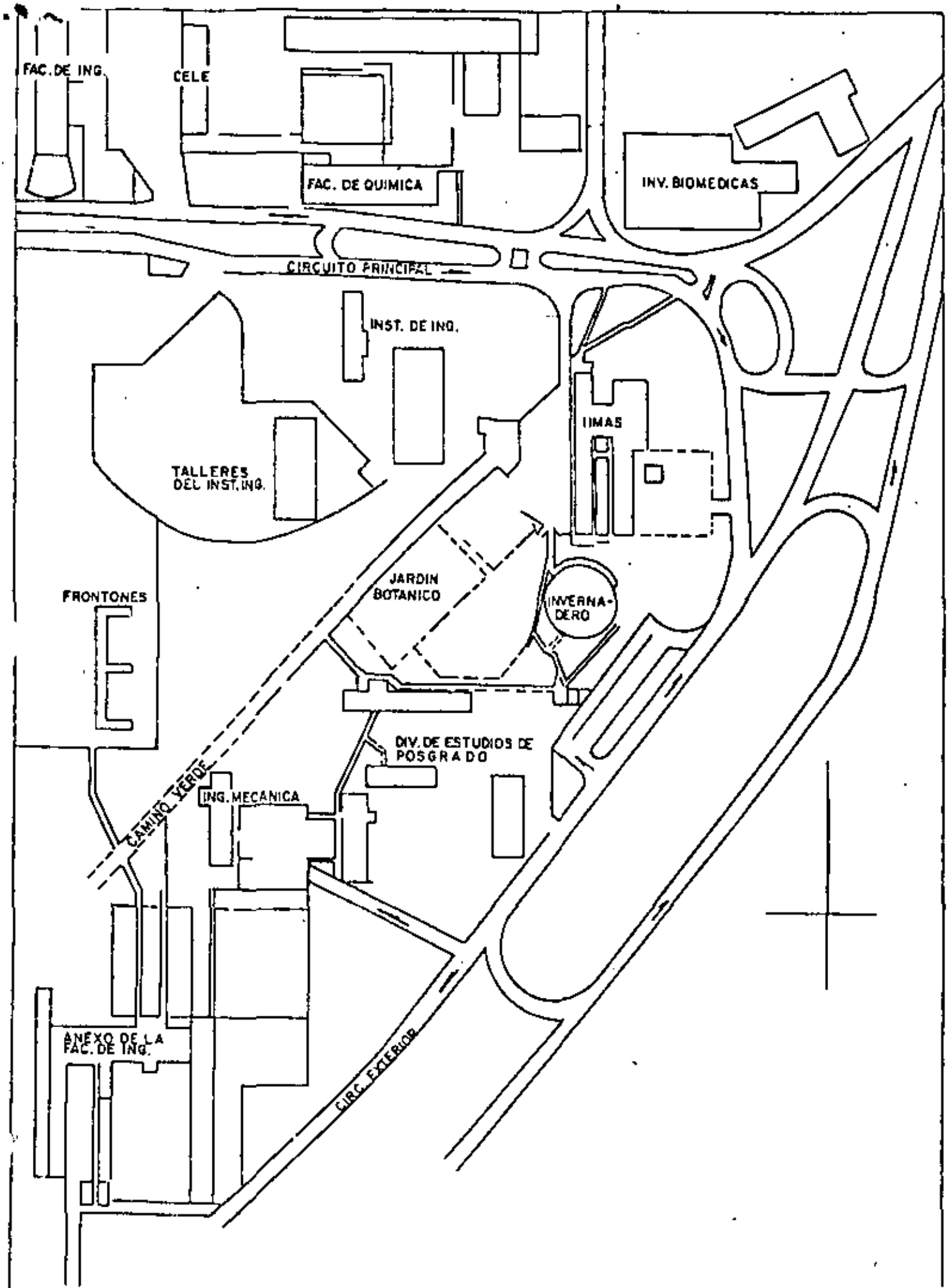
Ejemplos 9.

```
                        ;
                        ; DIRECCIONAMIENTO  INDEXADO
                        ;
0023 FD364313                   LD      (IY+43H),13H
                        ;
                        ; EL  DESPLAZAMIENTO  43H  SE  SUMA  AL CONTENIDO DEL PE-
                        ; GISTRO  IY  PARA DETERMINAR  LA  DIRECCION  EFECTIVA A
                        ; DONDE  SE  DEPOSITARA  EL  DATO  13H
                        ;
                        ;
0027 DD8621                     ADD     A,(IX+21H)
                        ;
                        ; EL  DESPLAZAMIENTO  21H  SE  SUMA  AL  CONTENIDO  DEL
                        ; REGISTRO  IX  PARA  DETERMINAR  LA  DIRECCION  DEL  O-
                        ; PERANDO  QUE SERA  SUMADO  AL  REGISTRO  A.  EL  RESUL-
                        ; TADO  QUEDA  EN  EL  REGISTRO  A.
                        ;
                        ;
002A DD3407                     INC     (IX+07H)
                        ;
                        ; EL  DESPLAZAMIENTO  07H  SE  SUMA  AL  CONTENIDO  DEL
                        ; REGISTRO  IX  PARA  DETERMINAR  LA  DIRECCION  DE  LA
                        ; LOCALIDAD  DE  MEMORIA  CUYO  CONTENIDO  SE  INCREMEN-
                        ; TA  EN  UNO.
                        ;
                        ;
```

Ejemplos 10.

```
                        ;
                        ; DIRECCIONAMIENTO  DE  BIT.
                        ;
0020 CBC7                       SET     0000H,A
                        ;
                        ; ENCIENDE  EL  BIT  0  DEL  REGISTRO  A.
                        ;
                        ;
002F CBAE                       RES     05H,(HL)
                        ;
                        ; APAGA  EL  BIT  5  DE  LA  LOCALIDAD  DE  MEMORIA  DI-
                        ; RECCIONADA  POR  EL  REGISTRO  HL.
```


CIUDAD UNIVERSITARIA


FAC. DE ING.
CELE
FAC. DE QUIMICA
INV. BIOMEDICAS
CIRCUITO PRINCIPAL
INST. DE ING.
IIMAS
TALLERES DEL INST. ING.
FRONTONES
JARDIN BOTANICO
INVERNA-DERO
DIV. DE ESTUDIOS DE POSGRADO
CAMINO VERDE
ING. MECANICA
ANEXO DE LA FAC. DE ING.
CIRC. EXTERIOR

DIRECTORIO DE ALUMNOS DEL CURSO "INTRODUCCION A LOS MICROPROCESADORES" IMPARTIDO EN ESTA DIVISION DEL 3 DE AGOSTO AL 1o. DE SEPTIEMBRE 1984.

1.- GOMEZ OCHOA BERNARDO RAFAEL
PRODUCTORA MEXICANA DE TUBERIA S.A.
COORDINADOR
AV. INSURGENTES SUR No. 664-11o. P
COL. DEL VALLE
543-61-42

CUARTEL No. 20
CONTADERO CUAJIMALPA
05500 MEXICO, D.F.
812-15-81

2.- PALOMERAS MURILLO JULIAN

DR. ANDRADE No. 448-1
COL. SALAS NARVARTE
519-49-31

3.- ACUÑA AZNAR ALFONSO
CONSTRUCTORA METRO, S.A.
SUPERINTENDENTE GRAL.
ALTADENA No. 23
COL. NAPOLES
DELEGACION BENITO JUAREZ
03810 MEXICO, D.F.
687-61-99

GASPAR DE ZUÑUGA No. 120
COL. LOMAS DE CHAPULTEPEC
DELEGACION MUGUGEL HIDALGO
520-05-94

4.- ALACALA CAYTAN UBALDO
NACIONAL FINANCIERA, S.A.
ESPECIALISTA
BOLIVAR No. 38-8o. PISO
COL. CENTRO

PASEO DE SAN JACINTO No. 37
COL. ALTEÑA III
53120 NAUCALPAN DE JUAREZ
585-59-68 y 562-93-85

5.- ALVARADO GAMAS HECTOR ARTURO
DIREC. GRAL. ING. DE SISTEMAS
ANALISTA PROGRAMADOR
AV. TEPALCATES S/N
691-70-50

SAYONA No. 60
COL. RESIDENCIAL ACOXPA
DELEGACION TLALPAN
684-55-44

6.- ASCENCION SOLORIO JOSE ARTURO
INVESTIG. CIENTIFICA EN COMPUTACION
GERENTE DE SISTEMAS
TENANCO No. 17
COL. ROMA
DELEGACION CUAUHTEMOC
06760 MEXICO, D.F.
564-51-14

AV. UNIVERSIDAD No. 2016
COL. COPILCO UNIVERSIDAD
06660 MEXICO, D.F.

7.- BARAJAS VELAZQUEZ JOSE
SEMBAM
LABORATORISTA
BOULEVARD PUERTO AEREO No. 485
AVIACION CIVIL
571-35-00 ext. 281

SUR 103 No. 405
COL. HEROES DE CHURUBUSCO
582-82-30

8.- CASTRO MEDINA GABRIEL
INSTITUTO MEXICANO DEL PETROLEO
JEFE OFNA. SISTEMATIZACION Y DESARROLLO
AV. DE LOS CIEN METROS No. 152
COL. SAN BARTOLO ATEPEHUACAN
DELEGACION VENUSTIANO CARRANZA
567-66-00 ext. 2398

BOSQUES DE LA INDIA No. 113
COL. BOSQUES DE ARAGON
NETZAHUALCOYOTL

10.- CARRO DE LA FUENTE EMIGDIO FCO.
PEMEX

11.- DIP TOBIAS ANUAR
PEMEX
SUPERVISOR TELECOMUNICACIONES
MERINA NACIONAL No. 329
COL. VERONICA ANZURES
545-23-38

NICOLAS SAN JUAN No. 1629
COL. DEL VALLE
DELEGACION BENITO JUAREZ
524-54-92

12.- FLORES HUITRON MARCOS
INSTITUTO MEXICANO DEL PETROLEO
ANALISIS Y DISEÑO ESTRUCTURAS MARINAS
EJE CENTRAL LAZARO CARDENAS No. 152
COL. SAN BARTOLO ATEPEHUACAN
DELEGACION VENUSTIANO CARRANZA
07730 MEXICO, D.F.
567-66-00

FRESNOS No. 17
COL. VALLE DEL SUR
09810 IZTAPALAPA
581-95-85

13.- FLORES VAZQUEZ NESTOR
BANCA CONFIA, S.N.C.
TECNICO EN TELECOMUNICACIONES
BALDERAS No. 36-5o. PISO
COL. CENTRO
DELEGACION CUAUHTEMOC
06050 MEXICO, D.F.
510-41-80

ANDADOR HACIENDA DE BUSTILLOS No. 1
COL. UNIDAD HABITACIONAL FCO. VILLA
DELEGACION AZCAPOTZALCO
02420 MEXICO, D.F.

14.- FLORES MARCOS
I. M. P.

15.- GONZALEZ MARQUES LUIS ARTURO
S. C. T. DIR. GRAL. CARRETERAS
PROYECTISTA DE TERRACERIAS
UNIVERSIDAD Y XOLA
COL. NARVARTE
DELEGACION BENITO JUAREZ

UNION POSTAL No. 80
COL. POSTAL
DELEGACION BENITO JUAREZ
034010 MEXICO, D.F.
696-78-30

16.- GONZALEZ ROMANILLOS JUAN JOSE
MEXACO, S.A.
INGENIRO INSTRUMENTACION
VILLA DE MADRID No. 1
COL. CONDESA
533-50-20

CIRUELO No. 28
COL. XOTEPINGO
DELEGACION COYOACAN
544-00-44

17.- GONZALEZ TOVANY LUIS
INSTIT. NAC. INVEST. NUCLEARES.
PROFESIONISTA "D" INVESTIGADOR
CENTRO NUCLEAR KM. 34.5
CARRETERA FEDERAL MEXICO TOLUCA
518-23-60 ext. 181-187

SUR 119 No. 833
COL. ESCUADRON 201
DELEGACION IZTAPALAPA
09060 MEXICO, D.F.
582-87-64

18.- GOUDINOFF HERREPA MARIO
CENTRO EVALUACION PROYECTOS SEMIP
DIRECTOR EVALUACION TECNOLOGICA
RIO RHIN No. 22-3er. PISO
COL. CUAUHTEMOC
546-01-06

CALLE DEL RELOJ No. 32-A
COL. RINCONADA HERRADURA
HUIXQUILUCAN EDO. DE MEXICO
294-46-79

19.- GUTIERREZ CERDA ALFREDO RAMON
CONSTRUCTORA METRO, S.A.
COORDINADOR DE MAQUINARIA
ALTADENA No. 23
COL.NAPOLES
DELEGACION BENITO JUAREZ
03810 MEXICO, D.F.
687-61-99

LAS HUERTAS No. 119
COL. DEL VALLE
DELEGACION BENITO JUAREZ
03100 MEXICO, D.F.
524-15-13

20.- GUEVARA RASCADO RENE
CONSULTORES ING. PLANEAC. URBANISMO
JEFE DE AREA
PETEN No. 543
COL. LETRAN VALLE
DELEGACION BENITO JUAREZ
03690 MEXICO, D.F.
575-25-11

PETEN No. 543
COL. LETRAN VALLE
DELEGACION BENITO JUAREZ

21.- HAWING ABDALA CAMERINO
CONSULTORES ING. PLANEAC. URBANISCO
JEFE DE AREA
PETEN No. 543
COL. LETRAN VALLE
DELEGACION BENITO JUAREZ
03650 MEXICO, D.F.

CONCEPCION MENDEZ No. 81-3
COL. NARVARTE
DELEGACION BENITO JUAREZ
02070 MEXICO, D.F.

22.- HERNANDEZ ESCUTIA REYNA
PETROLEOS MEXICANOS
INGENIERO ESPECIALISTA
MARINA NACIONAL No. 329
COL. ANAHUAC

PROL. LAGO AULLOGOS No. 90
COL.HUICHAPAN
DELEGACION MIGUEL HIDALGO
11290 MEXICO, D.F.
527-93-70

23.- HERNANDEZ RODRIGUEZ BEATRIZ EUGENIA
COMISION DEL PLAN NACIONAL HIDRAUL.
JEFE DE PROYECTOS
TEPIC No. 39
COL. ROMA
DELEGACION CUAUHTEMOC
06760 MEXICO, D.F.
574-66-92

AV. JALISCO No. 278-2
COL. TACUBAYA
DELEGACION MIGUEL HIDALGO
11870 MEXICO, D.F.
516-19-53

24.- HERNANDEZ YAÑEZ LUIS ENRIQUE
BANCRESER, S. N. C.
SUBGERENTE DE LABORATORIO
REFORMAN No. 243-8o. PISO
COL. CUAUHTEMOC
DELEGACION CUAUHTEMOC
06500 MEXICO, D.F.
525-62-24

EDIFICIO ALDAMA ENT. E DEPTO. 320
TLATELOLCO
DELEGACION CUAUHTEMOC
06500 MEXICO, D.F.
597-23-09

25.- KACHADOURIAN LEBLANC GREGORIO JOSE
THOMSON CSF MEXICO
APOYO DIVISION DE CONTROL RADARES
RIO NILO No. 80-2o. PISO
COL. CUAUHTEMOC
533-10-52

PIE DE LA CUESTA No. 7
COL. LOMAS DE BEZARES
DELEGACION MIGUEL HIDALGO
11910 MEXICO, D.F.
570-38-59

26.- LOPEZ ARCE SIU OSCAR
TELECOPIA, S.A.
INGENIERIA DE DESARROLLO
AV. CONSTITUYENTES No. 1054-2
COL. LOMAS ALTAS
DELEGACION MIGUEL HIDALGO
01150 MEXICO, D.F.
570-40-00

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

27.- LOPEZ GOMEZ MIGUEL ANGEL
S. A. R. H.
JEFE DE PROYECTO
TEPIC No. 39
COL. ROMA
DELEGACION CUAUHTEMOC
06760 MEXICO, D.F.
574-66-92

ORIENTE No. 2513
COL. G. RAMOS MILLAN
DELEGACION IZTACALCO
08720 MEXICO, D.F.
657-46-36

28.- MENDOZA BERNAL NOE
BANCA CONFIA
TECNICO DE TELECOMUNICACIONES
AV. JUAREZ No. 95-110
COL. CENTRO
510-41-80

C. VICENTE SUAREZ No. 28
COL. EL MORAL
DELEGACION IZTAPALAPA
09300 MEXICO, D.F.
686-55-09

29.- MORALES NOBLE ROBERTO SERGIO
CONSULTORES MEXICANOS, S..A
PROYECTISTA
MONTE ELBRUZ No. 134-1er. PISO
COL. POLANCO
DELEGACION MIGUEL HIDALGO
540-10-00

PROLONGACION UXMAL No. 1069 No. 1
COL. GENERAL ANAYA
DELEGACION BENITO JUAREZ
03340 MEXICO, D.F.
688-29-16

30.- OLAZARAN MORON GILDARDO
PEMEX

31.- PADILLA CHACON JOSE ANTONIO
PEMEX
INGNEIERO
MARINA NACIONAL
COL. ANAHUAC
545-23-38

METANO No. 37-8
COL. VALLEJO
759-13-06

32.- PEREZ ZAMORA CARLOS FABIAN
COMETRO GERENCIA "A"
JEFE DE FRENTE "B"
ALTADENA No. 23-8o. PISO
COL. NAPOLES
DELEGACION BENITO JUAREZ
03810 MEXICO, D.F.
523-10-67

HEGEL No. 346-12
COL. POLANCO
DELEGACION BENITO JUAREZ
11520 MEXICO, D.F.
531-84-83

33.- PRO TORRES ADRIAN
JUNTA LOCAL DE CAMINOS DEL EDO. MEX.
JEFE OFNA. INFORMAC. COMUN. SOCIAL

34.- SANCHEZ PADILLA JAVIER
TELECOPIA, S.A.
GERENTE DE INGENIERIA
AV. CONSTITUYENTES No. 1054
COL. LOMAS ALTAS
DELEGACION MIGUEL HIDALGO
01195 MEXICO, D.F.
570-40-00

VALLE DEL SENA No. 53-1
COL. VALLE DE ARAGON
ECATEPEC DE MORELOS EDO. DE MEXICO

35.- SANCHEZ SORIA RIZULO
PEMEX
ING. ESPECIALISTA
MARINA NACIONAL No. 329
COL. VERONICA ANZURES
254-46-02

U. TORRES DE MIXCOAC E15-24
COL. PLATEROS
DELEGACION ALVARO OBREGON
01460 MEXICO, D.F.
651-16-03

36.- URIBE LEYVA RENE
TELEFONOS DE MEXICO
AUXILIAR MANTENIMIENTO
PARQUE VIA No. 90
COL. SAN RAFAEL

AND. MAYAPAN No. 222 U. CULHUACAN
V. C T M
DELEGACION COYOACAN
04480 MEXICO, D.F.

37.- VADILLO DOMINGUEZ ARMANDO A.
MELCO DE MEXICO, S.A.
JEFE DE PRUEBAS
KM. 4 CARR. SAN JUAN DEL RIO
76800 QUERETARO, QRO.

CUAUHTEMOC No. 30
SAN JUAN DEL RIO QRO.

38.- WONG VARGA ROBERTO
CONSULTORIA EMPRESARIAL
CONSULTOR
AV. INSURGENTES SUR No. 1799-201
COL. SAN JOSE INSURGENTES
DELEGACION ALVARO OBREGON
524-48-75

AV. SAN BERNABE No. 872-19
COL. SAN JERONIMO
DELEGACION CONTRERAS