UNIVERSIDAD NACIONAL AUTONOMA DE MEXICO
FACULTAD DE INGENIERIA

PRACTICAS DEL

LABORATORIO DE MICROCOMPUTADORAS

FRANCISCO VERDUZCO MARTINEZ
JUAN CARLOS LOPEZ SANCHEZ

DIVISION DE INGENIERIA MECANICA Y ELECTRICA
DEPARTAMENTO DE COMPUTACION

FI/DIME/87-008

PRESENTACION.

En éste laboratorio conocerás y usarás ampliamente el microprocesador Z-80, además de algunos circuitos de proposito especial que permiten a este microprocesador comunicarse con un equipo exterior o bien con el usuario.

Existen en el laboratorio dos tipos de kits educacionales, que precisamente se basan en el microprocesador ya mencionado. Uno de ellos es el Starter kit y el otro es el MKE-Z80. Ambos necesitan una fuente de alimentación de + 5 volts. En ellos aprenderás a hacer uso del grupo de instrucciones de este microprocesador y realizarás circuitos y programas con los que notarás lo poderoso que es este CPU.

Sin embargo te recomendamos que sigas las siguientes reglas para el mejor aprovechamiento de tu equipo y del laboratorio en general.

1.- NO FUMAR EN EL LABORATORIO.

2.- NO INTRODUCIR ALIMENTOS.

3.- REALIZAR LAS PRACTICAS CON EL MENOR RUIDO POSIBLE, A ALGUNOS DE TUS COMPAÑEROS LES PUEDE MOLESTAR EL RUIDO.

Antes de trabajar con el equipo es necesario que conozcas como hacer uso de el.

En esta introduccion se describe el funcionamiento de las fuentes de voltaje.

En el laboratorio hallarás fuentes de voltaje variable.

Estas son :

FUENTE DE PODER FP-LAB 1 (I.M.A.C.) (de 0 a 40 volts)

FUENTE DE PODER FP-LAB 2 (I.M.A.C.) (de 0 a 15 volts)

FUENTE DE PODER TRIPLE 6236B (H.P.) (de 0 a +6 ,

0 a +20 y 0 a - 20)

INDICE

PRACTICA # 1

Conocimiento y manejo del Z-80 Starter Kit.

OBJETIVO.

El objetivo de esta práctica es que el alumno conozca cuáles 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 electrónicos, operadores de radio amateurs, hobbies de computación, control y evaluación en la industria, además de servir para la enseñanza de los primeros conocimientos sobre microcomputadoras.

Tanto un instructor como un alumno que esté 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 generación cuyo diseño se basó en el CI de Intel 8080. Las instrucciones del Z-80 son las mismas que las del 8080A más 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 aritméticas de 16 bit, operaciones en bloque, refresco de memoria dinámico, y códigos de operaciones de 16 bits los cuales lo hacen más flexible y más poderoso que cualquier otra máquina de 8 bits como: 8080A/8085, 6502, o el 6800/6802.

1.2 BOTON DE INICIO (RESET).

El botón de inicio nos sirve para inicializar al CPU y hacer que se empiece a ejecutar el programa que está en la dirección 0000H. Puesto que el monitor ZBUG se encuentra almacenado en los dos primeros Kbytes de memoria, al apretar este botón se transfiere el control del programa al ZBUG. Con esto, el ZBUG inicializa el Stack Pointer del usuario con la dirección 23C0H, inicializa las variables a cero, y si el interruptor S3 está colocado en la posición (MONITOR RESTART), el ZBUG coloca el prompt "-" en el despliegue de más a la izquierda. Si el

Todas estas fuentes tienen una perilla de control de voltaje y algunas de ellas también tienen perilla de control de corriente (las que no, la corriente se ajusta internamente). Para trabajar se colocan primero a sus valores mínimos (sin conectar los kits), a continuación debes de poner la perilla de corriente a su valor máximo y, por último ajustar el valor del voltaje a +5 Volts. En este instante puedes ya conectar tú kit.

FUENTES DE VOLTAJE FIJO (5 VOLTS)

FUENTE DE PODER N15005A (NORTH ELECTRIC)

Estas fuentes no tienen ni control de voltaje ni de corriente, es decir sus salidas están ajustadas para dar los + 5 volts que requieren tus kits.

ES IMPORTANTE QUE SIEMPRE AJUSTES LA SALIDA DE TUS FUENTES A ANTES DE CONECTAR LOS KITS, YA QUE UNA ENTRADA DE VOLTAJE MAYOR OCASIONARA QUE DANES LOS KITS.

puerto en los despliegues de más a la derecha. Si solo se proporciona un dígito de la dirección y luego se presiona la tecla PORT EXAM, el monitor ZBUG esperará a que se proporcione el segundo número para desplegar el contenido del puerto. Presionando la tecla NEXT podemos observar el contenido del siguiente puerto, de manera similar a la de MEM EXAM. Para modificar el contenido de ese puerto basta con teclear dos dígitos (en hexadecimal) para que el monitor ZBUG modifique el contenido del puerto en forma automática.

1.6 EXAMINADOR DE REGISTROS (REG EXAM).

Esta tecla tiene un uso similar a la de las dos anteriores, pero 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 área en 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 ejecución del programa.

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

Su uso es similar al 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 método que se utiliza para poder emplear los puntos de ruptura es cambiar el código de operación del usuario con una instrucción RST8 (CFH) y almacenarlo en una tabla de direcciones de puntos de ruptura y códigos de operación (BPTAB). La forma en que operan los puntos de ruptura es la siguiente: cuando se está ejecutando el programa y se encuentra un punto de ruptura, el control es transferido al ZBUG a través de la instrucción RST8, siendo salvados todos los registros del CPU en el "Mapa de Registros del Usuario", para que puedan ser examinados y modificados posteriormente. Después de esto, todas las instrucciones RST8 son reemplazadas por el codigo de la instrucción 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 dirección en la cual queremos

que ocurra el punto de ruptura y luego presionamos la tecla de BREAKPOINT. El despliegue mostrará la dirección suministrada después de que se tecleó un punto de ruptura para indicar que este ha sido aceptado. Para poner 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 dirección después de presionar la tecla de BREAKPOINT, el ZBUG indica que ya no se pueden poner más puntos de ruptura. Para eliminar los puntos de ruptura lo podemos hacer de dos maneras: Presionar la tecla de BRAKPOINT sin suministrar ninguna dirección cuando se tiene el prompt "-"; o usar la tecla SINGLE STEP que cancelará y removerá todos los puntos de ruptura existentes; o presionar el botón de RESET.

1.9 EJECUCION PASO A PASO (SINGLE STEP).

La tecla SINGLE STEP nos permite ejecutar el programa instrucción por instrucción. Así, 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 modificación del código de operación del usuario. Después de que se detuvo la ejecución 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 instrucción. 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 dirección que tenga el contador del programa (esta dirección esta salvada en el mapa de registros del usuario). Otra forma es proporcionar la dirección 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, sólo se puede detener mediante un punto de ruptura o mediante la tecla de MON.

1.11 TRANSFERENCIA A CASSETTE (CASS DUMP).

Por medio de esta tecla salvamos los programas volátiles que se encuentran almacenados en RAM, en cassettes normales usando la técnica de grabación 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; después se

procede de la siguiente manera:

1) Colocar el cassette en la grabadora y regresar la cinta.
2) Suministrar la dirección a partir de la cual se va a grabar en las localidades 23C0H y 23C1H (el byte más significativo en la 23C0H, y el byte menos significativo en la localidad 23C1H).
3) Dar la direccion de la ultima localidad a grabar, en la 23C2H y 23C3H.
4) Asegurarse que el prompt "-" aparece. Presionar la tecla de CASS DUMP y poner la grabadora en modo de grabación. El prompt desaparecerá.
5) No se necesita hacer ningún ajuste de volumen en la grabadora. Una vez que se llevo a cabo la grabación reaparecerá el prompt. Al menos 30 segundos se requieren para llevar a cabo la grabación.

1.12 LECTURA DE CASSETTE (CASS LOAD).

Para cargar programas almacenados en cassette se debe proceder así:

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 máximo de agudos y mínimo de graves.

3) Colocar el control de volumen al mínimo.

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

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

6) Si se llevó a cabo la lectura en forma correcta aparecerá el prompt "-" una vez que ésta se termine.

7) Si ocurre un error, aparecerá en el despliegue el próximo bloque de datos, habiéndose leído correctamente los bloques anteriores. Si existe 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 número de bytes (4 dígitos) que se desean subir al EPROM, luego ponemos el interruptor S2 en la posición PGM y presionamos la tecla PROM PROG. El despliegue se apagará y una vez terminada la grabación el ZBUG puede responder de dos maneras: La primera es la aparición del prompt "-" lo cual indica que la programación se realizó satisfactoriamente. La segunda indicación es la aparición de una dirección de 4 dígitos, que es la dirección a partir de la cual los datos no se grabaron correctamente. Si presionamos la tecla de NEXT, el ZBUG continúa checando la EPROM con la RAM y desplegará la próxima dirección en la EPROM donde el dato no se grabó. 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 posición de READ después de que termine la programación.

1.14 TECLA (NEXT).

El efecto de la tecla de NEXT en el modo de exámen de memoria ocasiona que pasemos a la siguiente localidad. Al examinar un puerto pasará al siguiente puerto, etc.

1.15 CALCULOS DE SALTO RELATIVO.

Hay una rutina para el cálculo de saltos relativos. Para utilizarla hay que poner en el registro HL la dirección destino, y en DE la dirección de la instrucción relativa. Luego hay que correr la rutina que se encuentra en la dirección C0H. Esta rutina pone el desplazamiento en la localidad que le corresponde. Si las direcciones dadas no son correctas, en el despliegue aparecerán números 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 dirección para el vector NMI. La dirección del botón de inicio por Hardware (0000H) es usada como punto de entrada al ZBUG. Cuando encendemos el Kit, automáticamente se empieza a ejecutar el ZBUG. La dirección 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 dirección de reinicio (0008H) se usa para poder manejar los puntos de ruptura y por lo tanto no puede ser usada. 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 instrucción utilizada. Desde que estas localidades están en el EPROM donde esta el ZBUG, se han puesto brincos a localidades específicas en RAM donde el usuario puede

poner otro brinco a alguna localidad que él desee. La siguiente tabla es un resumen del mapeo en RAM de las instrucciones de reinicio libres:

Instrucción	Código de operación	Dirección en la EPROM del ZBUG	Dirección 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 dirección 0020H. En esta dirección se puso una instrucción 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 él quiera. Sin embargo si desea utilizar el canal cero del CTC mientras el ZBUG monitor está usando los otros tres (Canal uno es usado para cuando grabamos un programa en un cassette, 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 cassette) se tienen que tener en cuenta varias cosas. Ya que los cuatro canales del CTC están relacionados entre sí, en el ZBUG se previno esto y se mapeo la interrupción del canal cero en RAM en 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 está en la dirección 07F8H y donde se puso la dirección 23D6H. 23D6H es la dirección de la primera instrucción que será ejecutada después de que el canal cero se interrumpa.

1.18 MAPA DE MEMORIA- USO DE LA RAM.

El mapa de memoria del Starter 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 cinco:

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

2) UFOR1- Dirección de llamado 063CH. IX apunta a dos localidades en memoria y A contiene dos dígitos hexadecimales que serán escritos en la memoria (el dígito 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- Dirección 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 - Dirección de llamado 06B3H. Convierte un caracter ASCII a su equivalente en binario. Los registros usados son A y F. El caracter ASCII está en A antes de la llamada, y el resultado queda en A después de la llamada.

5) UBASC - Dirección de llamado 06BBH. 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. Consulte este diagrama y los diagramas de las figuras 2 y 3 para entender mejor la siguiente discusión.

2.2 CIRCUITERIA DE RELOJ.

El reloj del Kit trabaja a una frecuencia de 2 MHz. teniendo un período de 500 nseg. Esto se hizo así para asegurar una máxima compatibilidad con el 8080. La fuente del reloj es un cristal oscilador basado en un 74LS04 y que corre a 3.9936 MHz. Se eligió esta frecuencia ya que es múltiplo de 1200/2400 Hz y las frecuencias de 300 bauds requeridas para poder grabar información 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 dispositivos que tiene este kit.

2.3 Z80-CPU.

El Z-80 CPU es el cerebro del Z-80 Starter kit y provee la mayoría de las señales de control para hacer un barrido tanto sobre los despliegues como en el teclado, o bien cuando se está 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 señales de control.

2.4 Z-80 PIO.

El Z-80 PIO es una interface en paralelo diseñada para la familia Z-80. Este soporta una gran cantidad de software para el manejo de interrupciones con dos conjuntos de líneas de control y un controlador integral de interrupciones. Dos puertos de 8 bits mas las líneas de control, están 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 está disponible para el usuario. Los demás canales del CTC son utilizados por el ZBUG así:

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 cinco:

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

2) UFOR1- Dirección de llamado 063CH. IX apunta a dos localidades en memoria y A contiene dos dígitos hexadecimales que serán escritos en la memoria (el dígito 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- Dirección 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 - Dirección de llamado 06B3H. Convierte un caracter ASCII a su equivalente en binario. Los registros usados son A y F. El caracter ASCII está en A antes de la llamada, y el resultado queda en A después de la llamada.

5) UBASC - Dirección de llamado 06BBH. 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. Consulte este diagrama y los diagramas de las figuras 2 y 3 para entender mejor la siguiente discusión.

2.2 CIRCUITERIA DE RELOJ.

El reloj del Kit trabaja a una frecuencia de 2 MHz. teniendo un período de 500 nseg. Esto se hizo así para asegurar una máxima compatibilidad con el 8080. La fuente del reloj es un cristal oscilador basado en un 74LS04 y que corre a 3.9936 MHz. Se eligió esta frecuencia ya que es múltiplo de 1200/2400 Hz y las frecuencias de 300 bauds requeridas para poder grabar información 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 dispositivos que tiene este kit.

2.3 Z80-CPU.

El Z-80 CPU es el cerebro del Z-80 Starter kit y provee la mayoría de las señales de control para hacer un barrido tanto sobre los despliegues como en el teclado, o bien cuando se está 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 señales de control.

2.4 Z-80 PIO.

El Z-80 PIO es una interface en paralelo diseñada para la familia Z-80. Este soporta una gran cantidad de software para el manejo de interrupciones con dos conjuntos de líneas de control y un controlador integral de interrupciones. Dos puertos de 8 bits mas las líneas de control, están 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 está disponible para el usuario. Los demás canales del CTC son utilizados por el ZBUG así:

CTC	Canal 1	Audio Cassette durins Dump.
CTC	Canal 2	Sinsle Step and EPROM Prosrammer.
CTC	Canal 3	Audio Cassette durins Load.

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

2.6 TECLADO Y DESPLIEGUES (DISPLAY).

Los despleaados hexadecimales son barridos por el Z-80 CPU bajo el control del ZBUG. Los datos se mandan a los despliegues a través del puerto que tiene la dirección 88H, mientras que el despliegue activo se selecciona por el puerto de dirección 8CH. La información en cada despliegue permanece aproximadamente durante 1 mseg. Mientras el puerto de dirección 8CH barre los despliegues, también barre el teclado. El puerto que tiene la dirección 90H es la entrada del teclado al bus de datos. Como el puerto de dirección 8CH barre el teclado, entonces si pulsamos una tecla (puerto 90H) podemos determinar qué tecla fue, viendo el contenido de cada uno de los puertos. El interruptor MONITOR/PROM1 RESTART es leído por el puerto 90H cuando se da una inicialización para determinar si el programa que se ejecutará es el ZBUG o el que se encuentra en la PROM1.

2.7 INTERFACE A CASSETTE.

La interface para grabar información en un cassette usa el formato de grabación de la ciudad de Kansas. El CTC genera las frecuencias requeridas para la grabación, 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 dirección que será programada. El software usa la instrucción LDI para mover los datos que se encuentran en RAM a la EPROM (dirección 1000-17FFH). El CTC mantiene la línea de WAIT (negada) activa para mantener los datos y las direcciones válidas durante la grabación.

2.9 DECODIFICACON DE MEMORIA.

La decodificación de la memoria se lleva a cabo por medio del C.I. 74LS138. Este C.I. 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 fisura 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	Conexion
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	80H-83H	Z-80 PIO
PS1	84H-87H	Z-80 CTC
PS2	88H-8BH	Latch segmentos
PS3	8CH-8FH	Latch displays
PS4	90H-93H	KB-SEL
PS5	94H-97H	UNUSED
PS6	98H-9BH	UNUSED
PS7	9CH-9FH	UNUSED

PS5, PS6, y PS7 están marcados en el área de alambrado para decodificar la entrada/salida de algún circuito propio. Las siguientes tablas nos muestran como están asignadas las direcciones de los puertos del PIO y del CTC.

Dirección 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

Dirección del puerto	Usado por el CTC
84H	Canal 0
85H	Canal 1
86H	Canal 2
87H	Canal 3

2.11 SISTEMA DE RAM.

El usuario tiene 1K de RAM proporcionada por los circuitos 21L02-1. Se puede agregar 1K más 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 instrucción del usuario después 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 después de que una instrución se termina de ejecutar, y el CPU direcciona 66H, esta dirección está 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 posición del interruptor S3. Si S3 está puesto en la posición MONITOR RESTART, el ZBUG pondrá el prompt "-" en el despliegue y empezará a barrer el teclado. Pero si S3 está puesto en PROM1 RESTART, entonces automáticamente se empieza a ejecutar el programa que se encuentre en la dirección 0800H. Esta característica nos permite iniciar un programa del usuario sin necesidad de ocupar el teclado.

2-14 ENCADENAMIENTO DE INTERRUPCIONES.

Toda la familia de dispositivos periféricos para el Z-80 tienen una circuitería para el control de interrupciones, de esta manera no se requiere de un controlador muy complejo. La prioridad de las interrupciones está determinada por una cadena de margarita que corre a través de los dispositivos del Z-80 (IEI-entrada, IEO-salida). En este kit la más alta prioridad la tiene asignada el CTC, y le sigue el PIO. La salida de la cadena de margarita se encuentra en el área 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 información de la estructura de la cadena de margarita.

2.15 INTERFACE CON EL BUS S-100.

El Starter Kit tiene dos hileras para colocar conectores de 100 pins, que tengan la configuración S-100. Esta interface es compatible con la mayoría de las memorias estáticas o con las tarjetas de expansión de entrada/salida. Específicamente es compatible con "S. D. System's 4K byte Static RAM card", pero no con módulos de EXPANDORAM. La interface con módulos que requieren señales específicas del 8080A como SYNC, INTA, DBIN, POC, PWR, y PRD requieren de una lógica adicional y de la conexión de algunas señales 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 entradas correspondientes en el kit.

2-16 AREA DE ALAMBRADO.

El área de alambrado sirve para colocar aproximadamente 25 chips de algún experimento, de expansión de memoria, de alguna interface de video, etc. La alimentación y la tierra estan alternados en el lado de abajo para que cada C.I. tenga una baja impedancia tanto a la fuente de poder como a la tierra, reduciendo considerablemente el problema del ruido en la circuitería del usuario. Z-80 CPU, PIO, y el sistema para decodificación de señales tienen una extensión en esta área para así poder hacer las conexiones de una manera más fácil.

DESARROLLO.

1) Resuelva el cuestionario previo.

2) El profesor explicará de manera general el hardware y el software con los que cuenta el Z-80 Starter Kit.

3) Teclee el programa que aparece a continuación y ejecutelo.

Este programa sirve para mover el caracter "8" de derecha a izquierda a través de los despliegues.

```
2000  3E00          LD A,00H
2002  D388          OUT (88H),A    ; Se activan todos los segmentos
2004  3E01          LD A,01H
2006  D38C    LOOP: OUT (8CH),A    ; Se activa el dígito a encender
2008  CD4F06        CALL D20MS
200B  CD4F06        CALL D20MS
200E  CD4F06        CALL D20MS
2011  CD4F06        CALL D20MS     ; Retardo de aproximadamente 80 mseg.
2014  07            RLCA           ; Rotación para seleccionar el dígito
                                   ; siguiente.
2015  18EF          JR LOOP
```

4) Una vez que el programa funciona, el profesor explicará cómo realizar los cálculos de saltos relativos sin necesidad de contar.

CUESTIONARIO PREVIO

1.- Qué impacto han tenido los microprocesadores en nuestra vida?

2.- Mencione tres diferencias entre las primeras microcomputadoras y las microcomputadoras actuales.

3.- Diga cuáles son los elementos principales que componen a un microprocesador?

4.- Qué diferencia existe entre una microcomputadora y una minicomputadora?

5.- Qué capacidad tiene la memoria 2102 y la 2716?

6.- Estudie el diagrama del reloj del Kit. Rediseñe el circuito para que la señal de reloj sea la misma si cambiamos el cristal por uno de 8 Mhz.

7.- Qué pasa cuando pulsamos el botón de Reset?

8.- Analice el diagrama de decodificación de memoria del Kit. Diga qué modificaciones tiene que hacer para que el área de RAM disponible para el usuario se encuentre a partir de la dirección 4000H.

9.- Qué dispositivos utiliza el Kit para comunicarse con el exterior?

10.- Diga cuáles dispositivos son de salida, cuáles son de entrada y cuáles sirven para ambas cosas.

FIGURA 1

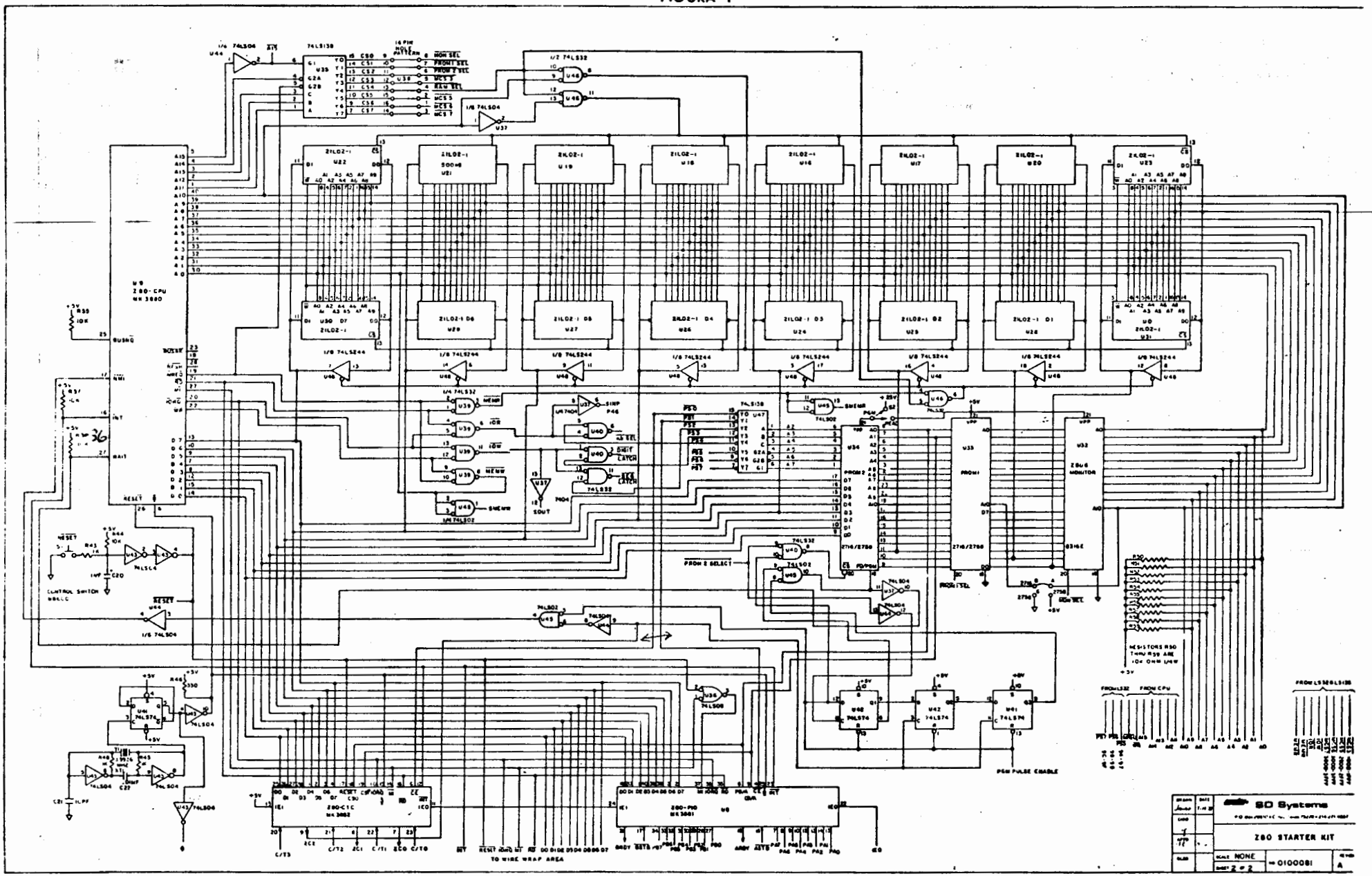
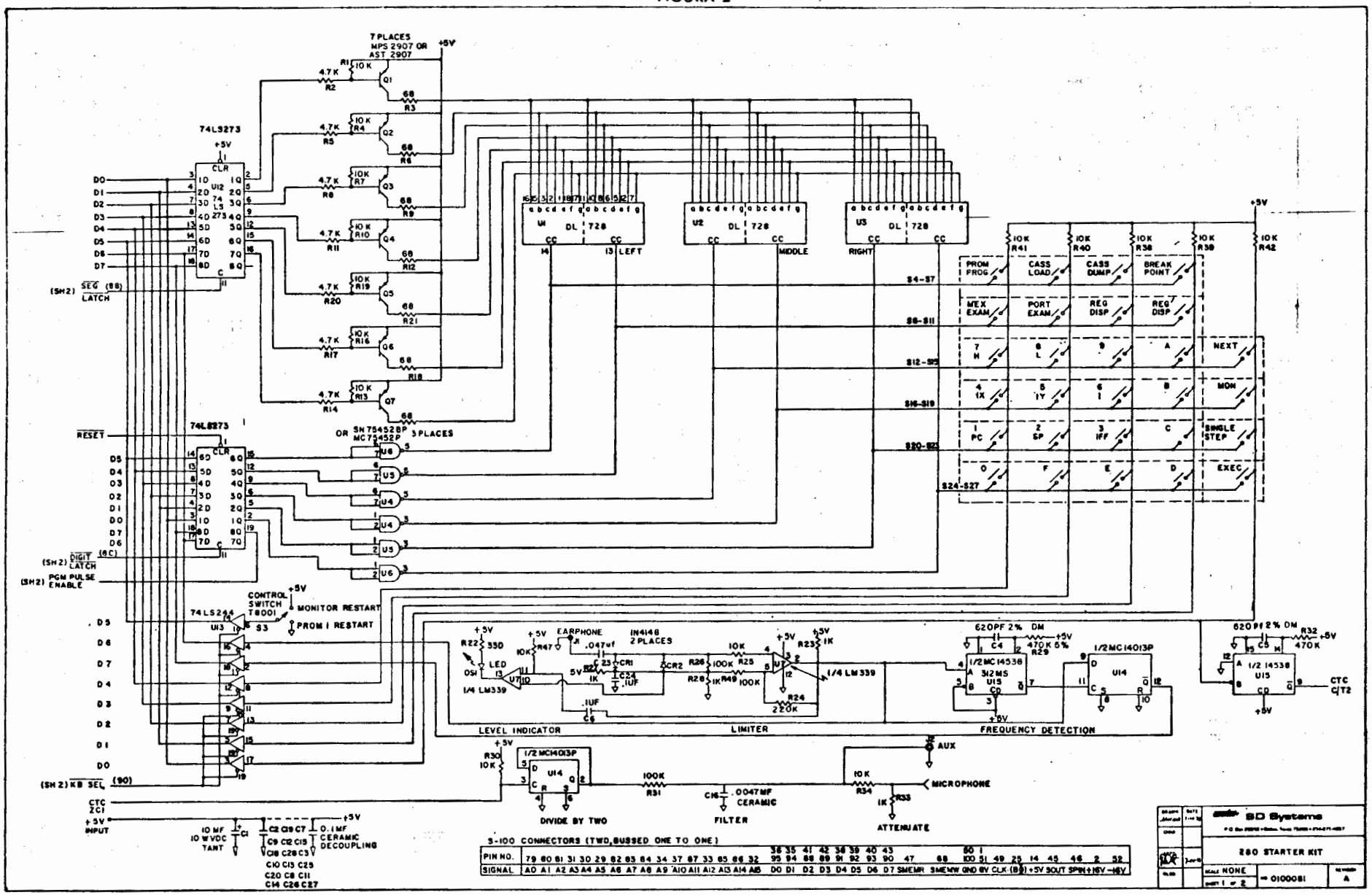

Z80 STARTER KIT
SD Systems
SCALE NONE
SHEET 2 OF 2
TO WIRE WRAP AREA
CONTROL SWITCH
PROM 2 SELECT
PROM 1 SEL
PSM PULSE ENABLE
RESISTORS R30 THRU R39 ARE 10K OHM 1/4W

FIGURA 2

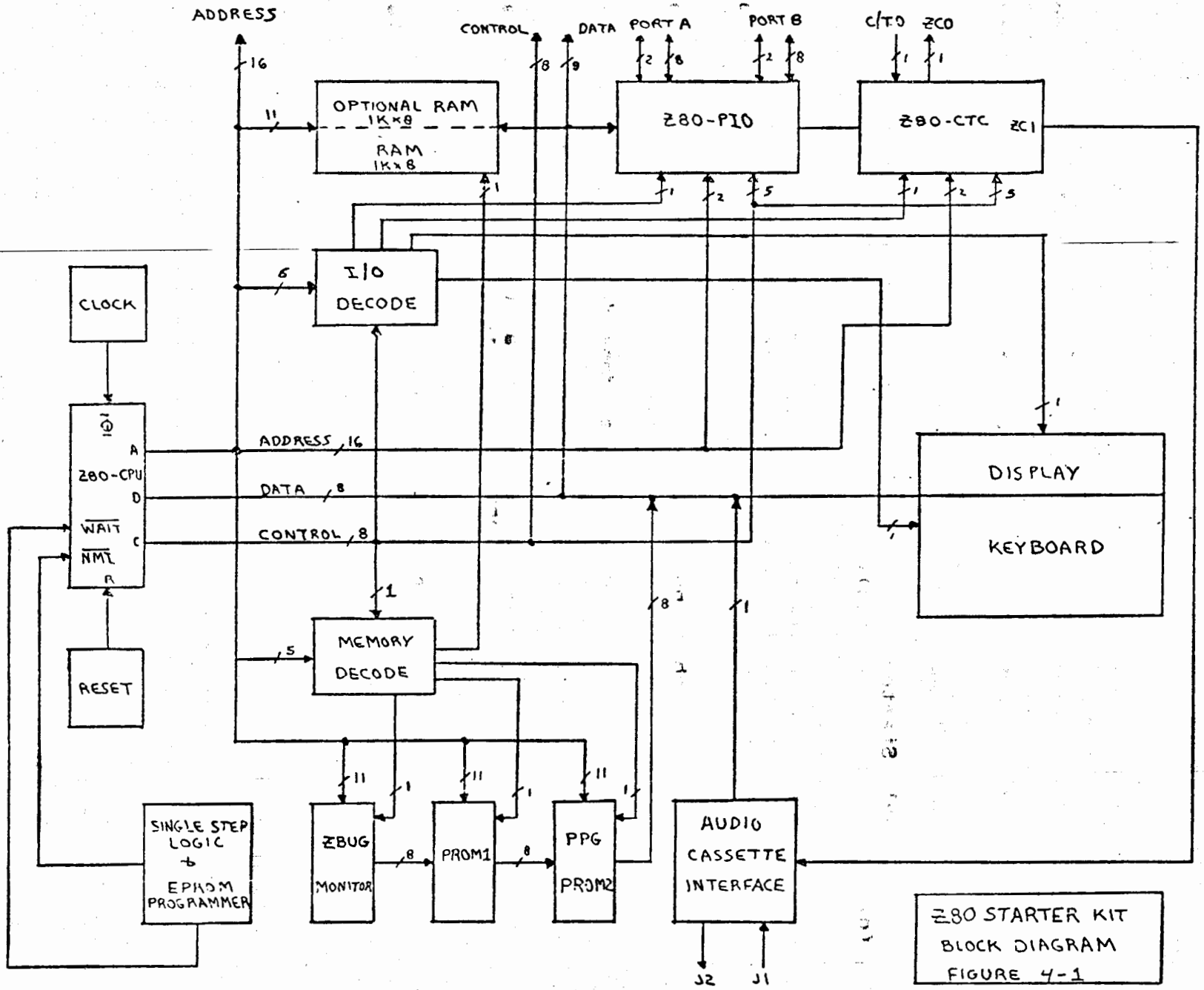


5-100 CONNECTORS (TWO, BUSSED ONE TO ONE)

PIN NO.	79 80 81 31 30 29 82 83 84 34 37 87 33 85 86 32	36 35 41 42 38 39 40 43	95 94 88 89 91 92 93 90 47	48	50 1	100 51 49 25 14 45 46 2 52
SIGNAL	A0 A1 A2 A3 A4 A5 A6 A7 A8 A9 A10 A11 A12 A13 A14 A15	D0 D1 D2 D3 D4 D5 D6 D7	SMEMR	SMEMW	GND	8V CLK (8X) +5V SOUT SPIN +16V −16V

	SD Systems
	280 STARTER KIT
	SCALE NONE
	SHEET 1 OF 2
	010008
	REVISION A

FIGURA 3



Z80 STARTER KIT
BLOCK DIAGRAM
FIGURE 4-1

4-1A

PRACTICA # 2

Manejo de las unidades de despliegue.

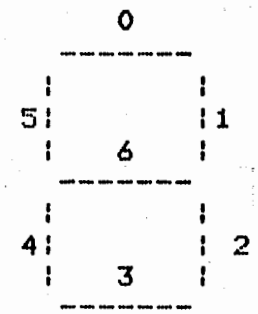
OBJETIVO.

El alumno aprenderá a usar las unidades de despliegue (U. de D.) de información luminosas. También aprenderá como generar las debidas rutinas de retraso para un correcto despliegue de la información.

INTRODUCCION.

En el Starter Kit contamos con 6 unidades de despliegue (display's de 7 segmentos), son usados por el monitor Z-BUG para desplegar direcciones y datos. A través de ellos podemos conocer cual es la información almacenada en cierta localidad de memoria, en los puertos o en alguno de los registros del procesador . Las 4 unidades de despliegue colocadas más a la izquierda, indicaran la dirección, si nos estamos refiriendo a localidades de memoria o a puertos, o bien, el nombre del registro interno del procesador, cuando queremos ver su valor. En tanto, las 2 últimas (de izquierda a derecha) nos muestran los datos contenidos en esa localidad, puerto o registro. Ver figura # 1.

La información que se despliega depende de dos tipos de datos. El primer dato se usa para indicar cual de las 6 unidades de despliegue se quiere usar, y el segundo, es la información a desplegar. Por lo tanto debemos habilitar la U. de D. deseada. Para habilitarla enviamos un 1 lógico a través de la línea correspondiente del puerto 8CH. Ahora, la información a mostrar depende del código que se envie por el puerto 88H, este puerto es el que maneja los siete segmentos de cada una de las unidades de despliegue. Para encender uno de los siete segmentos, se envia un cero por la línea correspondiente. Las U. de D. tienen la siguiente estructura :



Si queremos habilitar el despliegue N, se envía un 1 por el bit N.

```
Bit  7  ------ (No importa su valor)
Bit  6  ------ (No importa su valor)
Bit  5  ------ Habilitamos despliegue  # 5
Bit  4  ------ Habilitamos despliegue  # 4
Bit  3  ------ Habilitamos despliegue  # 3
Bit  2  ------ Habilitamos despliegue  # 2
Bit  1  ------ Habilitamos despliegue  # 1
Bit  0  ------ Habilitamos despliegue  # 0
```

Para habilitar el segmento N se envía un 0 por el bit N.

```
Bit  7  ------ (No importa su valor)
Bit  6  ------ Habilita segmento  # 6
Bit  5  ------ Habilita segmento  # 5
Bit  4  ------ Habilita segmento  # 4
Bit  3  ------ Habilita segmento  # 3
Bit  2  ------ Habilita segmento  # 2
Bit  1  ------ Habilita segmento  # 1
Bit  0  ------ Habilita segmento  # 0
```

DESARROLLO.

1) Resuelva el cuestionario previo.

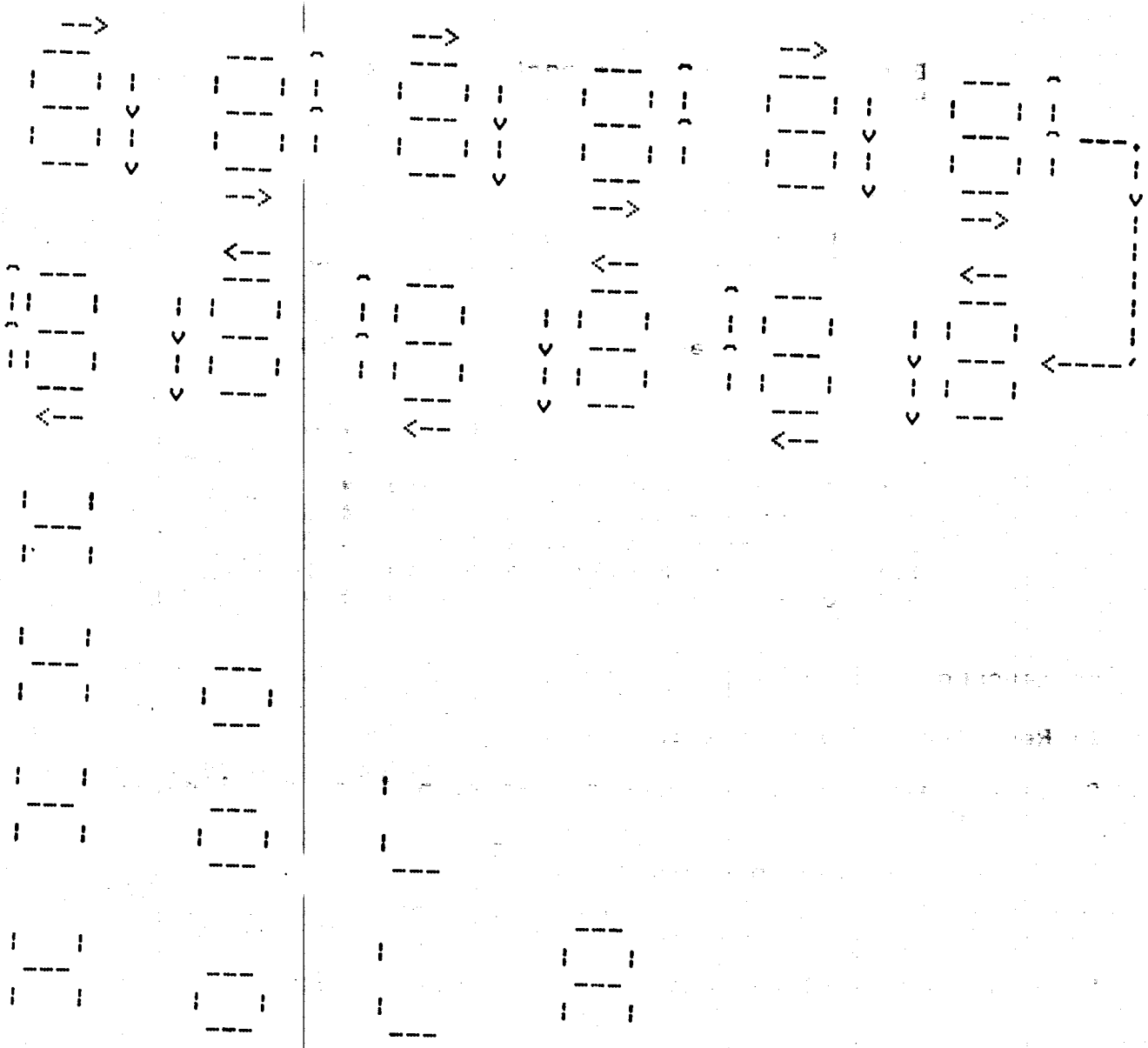
2) El profesor explicará como se manejan los despliegues en el Starter Kit.

3) Ejecute el siguiente programa.

Programa que despliega una "L" en el desplegado #3 del Starter Kit.

```
INICIO: LD A,08H        ;Se carga el acumulador con el dato que
        OUT (8CH),A     ;se enviará al puerto de los despliegues.
                        ;Note que el bit 3 es uno y los demás son
                        ;ceros    (0000 1000).
        LD A,47H        ;Se envía al puerto de los segmentos el
        OUT (88H),A     ;código (0100 0111), el cual encenderá
                        ;los segmentos 3, 4, y 5. El valor del
                        ;bit 7 no importa.
        Halt
```

4) Realice un programa que encienda los segmentos con la siguiente secuencia.



Nota:

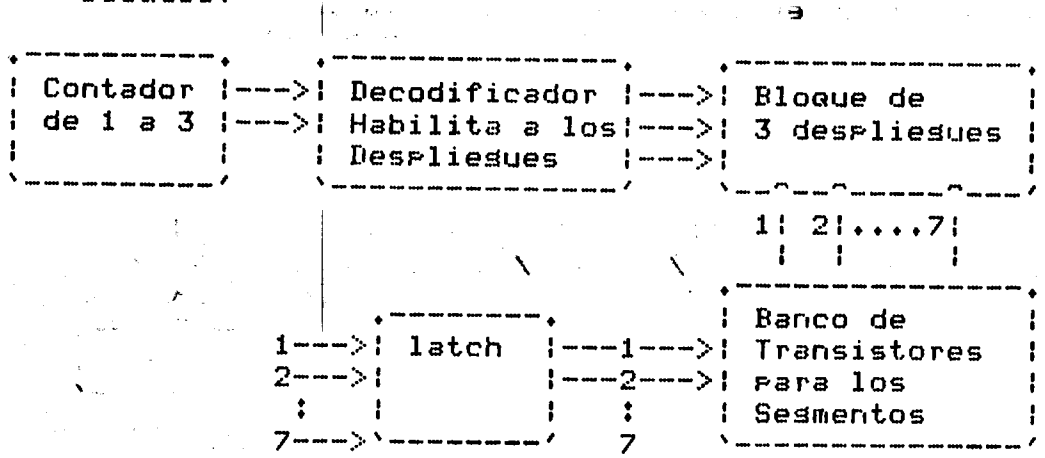
- El letrero de HoLA deberá permanecer encendido sin parpadear.
- Repita la secuencia tres veces.

CUESTIONARIO PREVIO

1.- Cuántos tipos de desplegados existen?

2.- Explique el diagrama que enciende los desplegados en el Starter Kit. Ver página 17 práctica 1.

3.- Realice un circuito que cumpla con el siguiente diagrama de bloques:



El circuito empezará a funcionar cuando los datos sean cargados. Después moverá la información por los tres despliegues 4 veces.

Este circuito deberá funcionar para el día en que se realice la práctica.

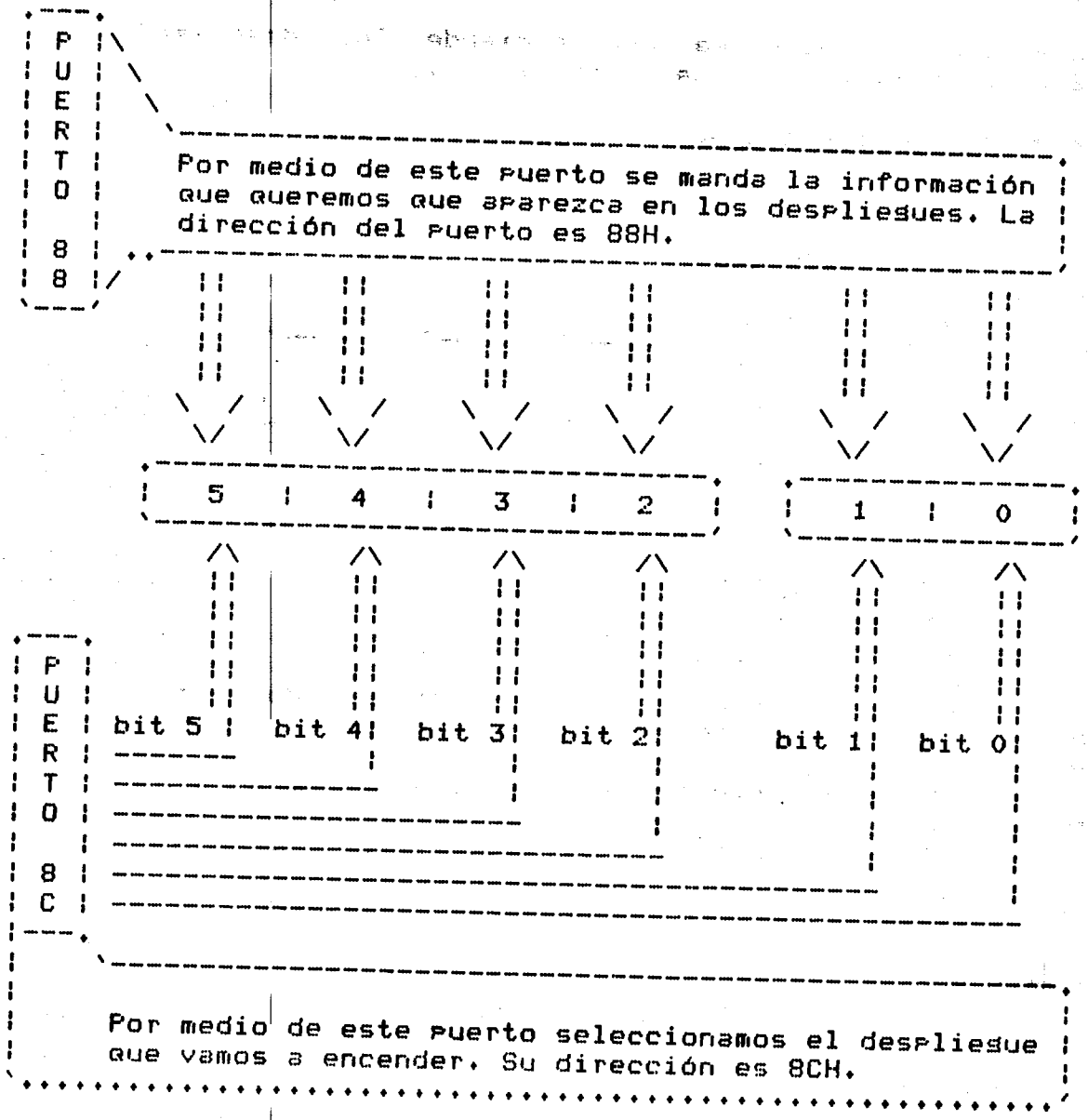

PUERTO 88
Por medio de este puerto se manda la información que queremos que aparezca en los despliegues. La dirección del puerto es 88H.
5
4
3
2
1
0
PUERTO 8C
bit 5
bit 4
bit 3
bit 2
bit 1
bit 0
Por medio de este puerto seleccionamos el despliegue que vamos a encender. Su dirección es 8CH.

FIGURA (1)

PRACTICA # 3

Conjunto de Instrucciones.

OBJETIVO.

El objetivo de esta práctica es que el alumno profundice su conocimiento del grupo de instrucciones del microprocesador Z-80. Además de aprender a manejar el registro de condición o de banderas (F).

INTRODUCCION.

El CPU Z-80 puede ejecutar 158 tipos diferentes de instrucciones que incluyen las 78 del 8080. Las instrucciones pueden dividirse de la siguiente manera:

- De carga e intercambio.
- De transferencia y búsqueda de bloques.
- Aritméticas y lógicas.
- De rotación y corrimiento.
- De manejo de bits.
- De saltos, llamadas y retornos.
- De entrada/salida.
- De control para el CPU.

Las instrucciones de carga pueden mover datos entre los registros del CPU o entre los registros y la memoria externa. Todas estas instrucciones deben especificar una localidad fuente desde la cual un dato será movido a una localidad destino. La localidad fuente no es alterada por la instrucción de carga. En este grupo estan incluídas las instrucciones de carga inmediata a cualquier registro del CPU y a las localidades de memoria. Las instrucciones de intercambio, como su nombre lo indica, sirven para intercambiar el contenido de dos registros.

Un conjunto único de instrucciones para la transferencia de bloques esta provisto en el Z-80. Con una sola instrucción un bloque de memoria de cualquier tamaño, puede moverse a cualquier otra parte de la memoria. El conjunto de movimiento de bloques, es extremadamente valioso cuando tenemos que procesar grandes cadenas de caracteres. Las instrucciones de búsqueda también son muy valiosas para este tipo de operaciones. Con una simple instrucción, en un bloque de memoria de una determinada longitud podemos buscar un caracter de 8 bits. Tanto las instrucciones de búsqueda como las de movimiento de bloques, pueden ser interrumpidas durante su ejecución para que no ocupen al procesador por largos periodos de tiempo.

Las instrucciones aritméticas y lógicas de 8 bits, operan sobre un dato que está en el acumulador, y sobre otro que se

encuentra en cualquiera de los registros del CPU o en alguna localidad de memoria externa. En este grupo se encuentran también las operaciones de 16 bits.

Las operaciones de rotación y corrimiento permiten que cualquier registro o localidad de memoria sea rotada tanto a la derecha como a la izquierda, con o sin su acarreo, ya sea aritmético o lógico. También un dígito en el acumulador puede ser rotado a la izquierda o a la derecha con dos dígitos en alguna localidad de memoria.

Las instrucciones para el manejo de bits permiten que cualquier bit en el acumulador, de cualquier registro de propósito general, o de una localidad de memoria externa sea prendido, apagado o evaluado con una simple operación. Por ejemplo, el bit mas significativo del registro H puede ser prendido. Este grupo es especialmente usado en aplicaciones de control y para evaluar las banderas de software en programas de propósito general.

Las instrucciones de salto, llamadas y retornos se usan para la transferencia de control entre las diferentes localidades de un programa. Este grupo usa varias técnicas para obtener la nueva dirección del contador del programa. Un tipo único de llamada, es la instrucción de RESTART. Esta instrucción actua conteniendo la nueva dirección como parte de de un código de operación de 8 bits. Esto es posible desde que solo 8 direcciones localizadas en la página cero de la memoria externa pueden ser especificadas. Los saltos en los programas pueden ser llevados a cabo por medio de una carga de los registros HL, IX o IY directamente en el contador del programa.

El grupo de instrucciones de entrada/salida nos permiten la transferencia de información entre las localidades de memoria y entre los registros del procesador, con los dispositivos de entrada/salida. En cada caso, el número del puerto es provisto de los 8 bits mas bajos del canal de direcciones durante la operación de entrada/salida. Una instrucción permite que el número de puerto sea especificado por el segundo byte de la instrucción, mientras otras instrucciones del Z-80 nos permiten esto, especificandolo como el contenido del registro C. Una mayor ventaja de usar el registro C como un apuntador a los dispositivos de entrada/salida es que permite que diferentes puertos compartan las mismas rutinas de software. Esto no es posible cuando la dirección es parte de el código de operación que esta grabado en ROM. Otras características de las instrucciones de entrada es que automáticamente prenden el registro de banderas y con esto, no requerimos operaciones extras para determinar el estado de los datos de entrada. El CPU incluye instrucciones simples que pueden mover bloques de datos automáticamente a o desde algun puerto, directamente de cualquier localidad de memoria. Agregadas con el conjunto de registros de propósito general, estas instrucciones nos permiten una transferencia de datos bastante rápida. El valor de estas instrucciones se puede demostrar, cuando el CPU provee

todos los requisitos para el formateo de un disco flexible.

Finalmente, las instrucciones básicas de control permiten varios modos de operación. Este grupo incluye instrucciones para encender o apagar el flip flop de interrupciones, o para encender el modo de respuesta a las interrupciones.

DESAROLLO.

1) Resuelva el cuestionario previo.

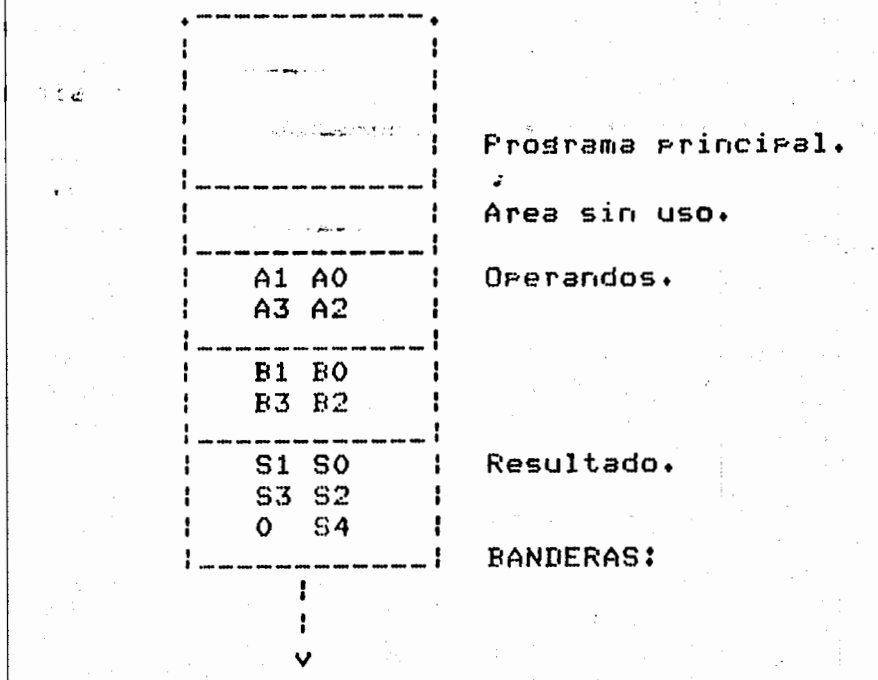
2) El profesor explicará los diferentes tipos de instrucciones que componen al Z-80.

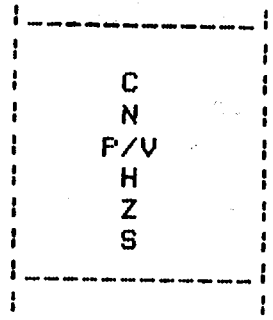
3) Realice los siguientes programas.

3.1 Programa que sume dos cantidades en BCD de doble precisión, es decir números en BCD de 4 cifras (2 bytes), estas cantidades estan almacenadas en localidades subsiguientes de memoria . Después de realizar la suma, el resultado debe almacenarse en las localidades siguientes a la de los operandos.

Una vez hecha la suma, debe guardarse el estado del registro de banderas. Cada condición debe ocupar una localidad de memoria. Si un bit de condición esta encendido, entonces se almacena un FF, si esta apagado se guarda un 00 .

El mapa de memoria del programa ès :





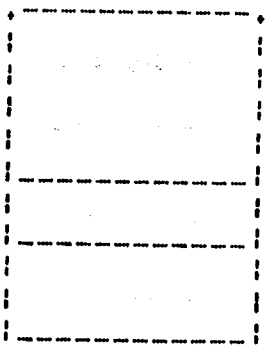
Acarreo.
Resta o comparación.
Paridad o sobreflujo.
Acarreo medio.
Cero.
Signo.

OPERANDOS

```
      A3 A2 A1 A0
    + B3 B2 B1 B0
    -------------
S4 S3 S2 S1 S0
```

En la práctica No. 5 se encuentra una explicación del resistro de banderas.

3.2 Este programa debe realizar la búsqueda de una cadena de caracteres en un parte de la memoria. En la memoria se hallan N caracteres, la cadena a buscar en el área es AA, BB, CC, si esta cadena se encuentra, se deberá mostrar por los desplegados el letrero de SI, en caso contrario deberá mostrarse el de NO.



Programa principal.

Area sin uso.

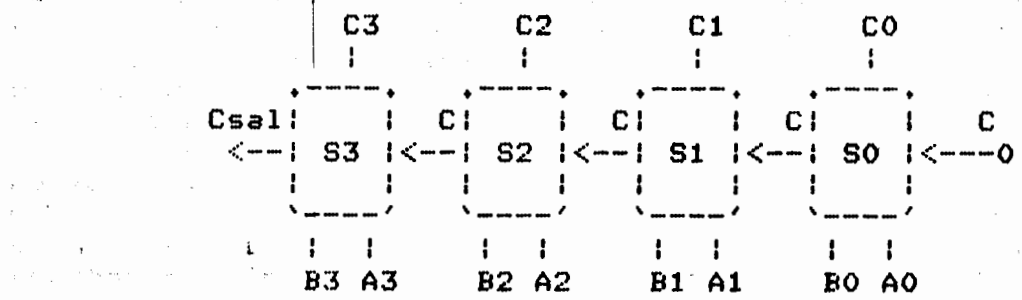
Area de memoria fuente.

CUESTIONARIO PREVIO

1.- Explique cuál es la diferencia entre el significado del bit de acarreo y del bit de sobreflujo.

2.- Obtenga los tiempos de ejecución del programa # 1, si usara:
a) Operaciones de 8 bits,
b) Operaciones de 16 bits.

3.- Diseñe la lógica para detectar las banderas de acarreo, sobreflujo, acarreo medio, cero y signo, si usted tuviera la siguiente unidad aritmética .



Donde:

C_0 = Bit de acarreo de entrada.

C = Acarreo de salida de cada sumador completo.

Csal= Acarreo de salida del sumador 3, que a su vez es el bit de acarreo de salida de la unidad aritmética de 4 bits.

A0-A3 = Operando 1.

B0-B3 = Operando 2.

C0-C7 = Resultado.

4.- Arme el circuito que diseñó. Se tiene que llevar armado para el día en que se realice la práctica.

5.- Cuál es el rango de valores (en decimal) que pueden expresarse en:
a) 8 bits,
b) 16 bits,
c) 24 bits.

PRACTICA No. 4

Manejo del teclado.

OBJETIVO.

Al final de la práctica, el alumno aprenderá como llevar a cabo el barrido del teclado del Starter Kit para distinguir que tecla fue pulsada. Aprenderá cual es la disposición del hardware del Starter Kit para introducir la información por el teclado.

INTRODUCCION.

La entrada de los datos al sistema se realiza con un teclado de forma matricial. Por medio de éste, el usuario puede editar y ejecutar un programa en código hexadecimal. En este tipo de teclados, tenemos una matriz formada por M renglones y por N columnas. En el Starter Kit los renglones se habilitan a través del puerto (8CH) y el dato leído se toma del puerto (90H). Cada columna esta conectada a un 1 lógico con una resistencia. Si en este teclado no se oprime ninguna tecla, a la salida de las N columnas siempre habrá un 1 lógico en todas las líneas. Para sensar los renglones mandamos un 0 lógico por el renglón a sensar, y por los otros, enviamos un 1 lógico. Si se oprime alguna tecla de ese renglón aparecerá un 0 por su columna correspondiente. De esta manera intersectando el renglón y la columna se sabe que tecla fue oprimida. Por ejemplo: en el teclado de la figura 1 se esta sensando el renglón número 2, por ese puerto se mando un (111011), ahora bien si pulsamos alguna de las teclas de ese renglón, en la columna correspondiente aparece un 0, en este caso es la columna 3 (10111). El diagrama del teclado del Starter Kit esta en la página 17 de la práctica número 1.

DESARROLLO.

1) Resuelva el cuestionario previo.

2) El profesor explicará el funcionamiento del teclado en el Starter Kit.

3) Elabore un programa que detecte que tecla se oprime en el Starter Kit. El programa debe de tener las siguientes características:

   - Cuando demos la dirección de inicio y pulsemos la tecla de EXEC, el prompt se deberá apagar.

   - Luego, se debe realizar un barrido del teclado.

- Cuando se detecte una tecla, se deberá mandar a algún desplegado el número que se le asigne a ésta, este número deberá permanecer encendido durante cinco segundos, al cabo de los cuales se podrá pulsar otra tecla.

- Se debe entregar el listado del programa muy bien documentado.

4) Cuál es el tiempo de ejecución de su programa?

CUESTIONARIO PREVIO

1.- Cuáles son los principales problemas que se presentan en los teclados?

2.- Mencione los métodos que se utilizan para resolverlos.

3.- Explique cómo haría un barrido del teclado por hardware?

4.- Explique cómo haría un barrido por software?

5.- Cómo se decodifican las señales generadas de la identificación del teclado en el código de caracteres ASCII?

6.- Qué sucede cuando se oprimen varias teclas.

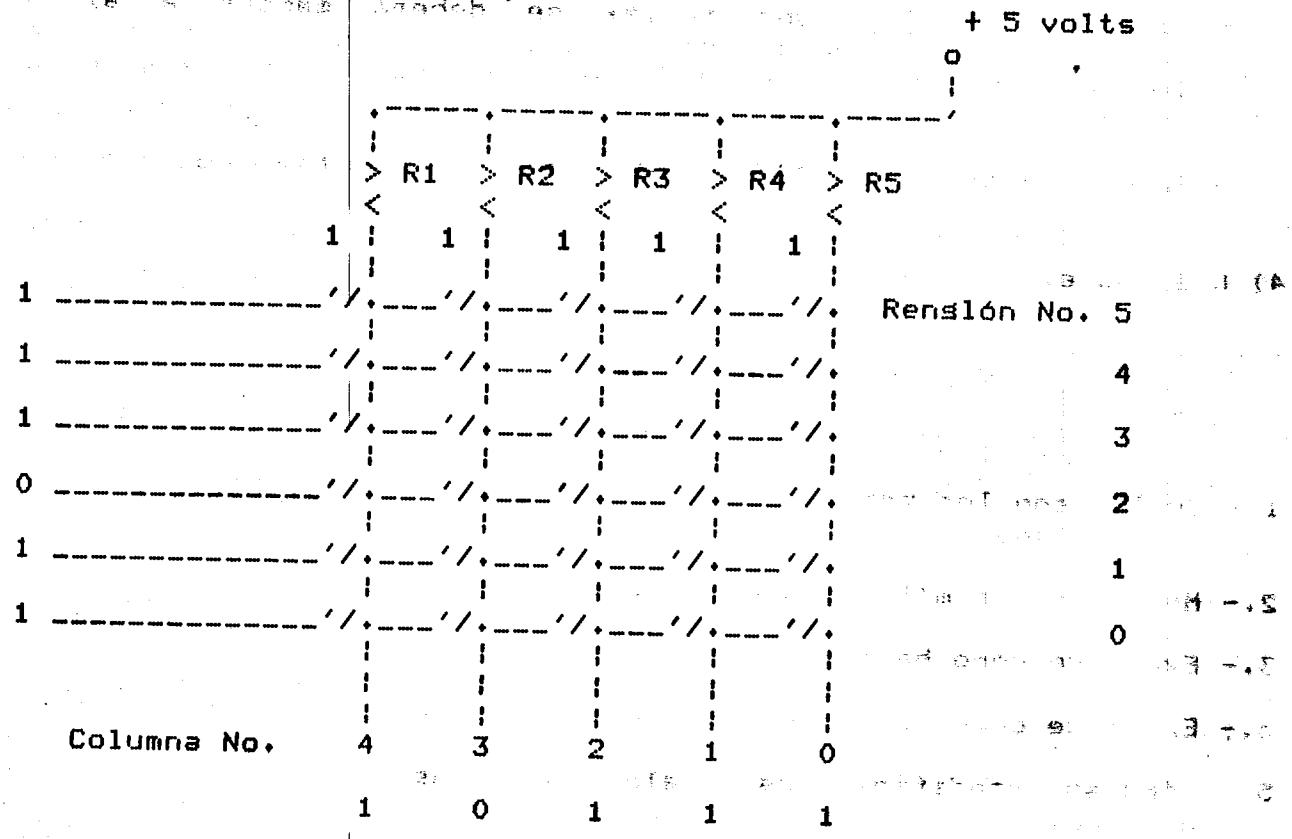


FIGURA (1)

De acuerdo a la figura, la tecla oprimida esta en el renglón 2 y la columna 3. Note que si alguna otra tecla de otro renglón se pulso, ésta no se podrá detectar.

PRACTICA # 5

Ensamblado de programas en Cromemco.

OBJETIVO.

Al final de la práctica el alumno sabrá 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 utilización de una serie de programas con los que el programar en ensamblador resulta relativamente fácil. 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 código objeto y un archivo con el 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 más archivos durante su ejecución y buscar una librería especifica del usuario de alguna rutina relocalizable para ponerla en un módulo de código común.

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

Los párrafos anteriores nos muestran de manera general los pasos que se siguen para trabajar con Cromemco. La figura 1 es la gráfica de éstos. Cabe aclarar que ésto se detallará mas adelante.

DESARROLLO.

Suponsamos que tenemos un programa elaborado en Z-80, entonces los pasos para trabajar en la computadora son cinco:

1.- EDICION DEL PROGRAMA.

Para editar un programa tenemos que teclear:

```
A.screen b:nombre.z80
```

Se crean dos programas llamados: nombre.z80 y nombre.bak. El

denominado nombre.z80 es el programa fuente, mientras que el nombre.bak es la versión anterior de éste. La máquina responde con:

```
Editor Screen 910, D I M E
 >Edit: @ Copy Delete Exit Find Insert Jump Move Other Substitute..
```

Este letrero aparece en la parte superior de la pantalla; lo que viene despues de "Edit:" son los comandos. Para invocar algún comando simplemente hay que teclear la letra mayuscula correspondiente. Es importante hacer notar que todos los comandos siguen la misma filosofía por lo que solo se explicaran unos cuantos.

a) INSERT.

Si queremos insertar un programa tecleamos la "I" , la máquina responde con:

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

Lo que quiere decir que vamos a insertar un texto, y que terminaremos este comando por medio de la tecla de ESC. Esta tecla se encuentra en la parte superior derecha del teclado.

b) DELETE.

En este caso la letra que se teclea es la "D". El programa pone:

```
>Delete: <esc>
```

Esto quiere decir que borraremos texto tecleando espacios o la tecla de RETURN, y que terminaremos el comando por medio de la tecla ESC.

c) EXIT.

Por último explicaremos el que se utiliza para salvar el programa en disco. Teclearemos la letra "E", la respuesta será:

```
 >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" , la "U" o la tecla de ESC. Si tecleamos la letra "Q" todos los cambios realizados no seran tomados en cuenta. Si tecleamos ESC regresamos al editor. Si tecleamos la "U" el programa se salva y la respuesta es:

```
A.
```

2.- ENSAMBLADO DEL PROGRAMA.

Si vamos a ensamblar el programa, tenemos que cambiar el

control al Drive B. Para esto tecleamos:

```
A.B:
```

La respuesta es:

```
B.
```

Para ensamblar ponemos:

```
B.asmb nombre
```

La máquina ensambla el programa y nos indica cual fue el resultado del ensamblado por medio de los siguientes letreros:

```
CROMEMCO Z80 Macro Assembler versión 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 editor y reensamblar el programa. En caso contrario hay que ligar el programa. Hay que hacer notar que si nuestro programa no se va a ejecutar en la Cromemco no se necesita ligar.

El Range Count nos indica el número de saltos absolutos que se pueden poner como saltos relativos, y el Program Lenght es la longitud del programa.

También se crean dos programas que son: el 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 código objeto.

3.- LIGADO DEL PROGRAMA.

Para ligar el prorama hay que poner:

```
B.link nombre,nombre/n/e
```

La opción /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 versión nn.nn

Data      0103    0162
[0103  0162 1]

B.
```

El Data nos indica desde donde inicia nuestro programa y donde termina. En este caso el programa ejecutable cominenza en la dirección 0103H y termina en la dirección 0162H.

Si ahora deseamos correr el programa simplemente hay que poner:

```
B.nombre
```

y el programa será ejecutado.

4.- DEPURACION DEL PROGRAMA.

Teclear:

```
B.debug   nombre.com
```

Si deseamos utilizar el depurador. La respuesta es:

```
DEBUG versión 00.17
NEXT   =   4F80
NEXTM  =   4F80
-
```

a) INFORMACION GENERAL.

El indicador de que ya estamos en el depurador es el "-". El depurador trae el programa denominado nombre.com y lo carga en la memoria a partir de la localidad 100H, indicando por medio del letrero NEXT, cual es la ultima localidad que ocupa nuestro programa.

El depurador utiliza comandos de uno o dos caracteres dados desde la terminal. Es un formato libre con respecto a los espacios. Las comas pueden ser usados en lugar de los espacios. Los comandos se deben terminar con un caracter de retorno (CR).

La dirección actual del contador del programa es representada por el signo de dolares. El siguiente ejemplo ejecuta el programa a partir de la dirección del contador del programa mas tres, y es terminado por medio de un punto de ruptura.

```
G"3
```

Los comandos para ensamblar en la memoria (A), despliegue de la memoria (D), listado del programa en mnemonicos de ensamblador (L), y el comando para cambiar una localidad de memoria (S) mantienen la dirección de inicio (100H) si no se da ninguna cuando se utiliza el comando. Esta dirección se cambia cada vez que un comando específico se ejecuta. Entonces la próxima dirección de inicio estará donde se terminó de ejecutar el programa.

b) REGISTRO @.

El registro @ sirve para indicar al depurador donde se localiza el módulo que deseamos arreglar. Para cambiar el valor de este registro teclee:

@

La máquina responde con:

@ -xxxx

Donde xxxx es el valor actual del registro. La computadora esperará a que se le de una dirección nueva. Si solo tecleamos un caracter de retorno, el registro no se altera. Si una dirección y un caracter de retorno es tecleado, entonces el registro tendrá la nueva dirección. El registro @ ahora ya puede ser usado como parte de una dirección, así:

G/@ @A3 1000

Este es un ejemplo con el comando (G). Se tienen que poner puntos de ruptura en el inicio del módulo actual, en la localidad A3H en el módulo actual y en la localidad 1000H. Esta característica nos permite probar un módulo sin necesidad de calcular direcciones absolutas.

La dirección relativa se pone sumada a la dirección absoluta, de otra manera las direcciones se presentan con el registro @ igual a cero.

c) DESPLEGADO DE LA MEMORIA (D o DM).

El formato para este comando es:

```
-D
-D direc1
-D direc1 direc2
-D direc1 S numero de localidades en hexadecimal
-D ,direc2
-D S numero de localidades en hexadecimal
```

El primer formato despliega 80H localidades de memoria a partir de la dirección actual. Del lado derecho de la terminal aparece el contenido de cada localidad de memoria en código ASCII. El segundo, despliega 80H localidades a partir de la dirección indicada. El tercero, las despliega entre la dirección 1 y la dirección 2. El cuarto, despliega tantas localidades como se le indiquen a partir de la dirección 1. El quinto pone las localidades que se encuentren entre la dirección actual y la dirección 2. Y el último pone el número de localidades indicadas a partir de la dirección actual. Por ejemplo:

```
-D 100,S30

0100   40 41 42 43  44 45 46 47  48........4F     @ABCDEFGHIJKLMNO
0110   50 51 52 53  54 55 56 57  58 .......34     PQRSTUVWXYZ01234
0120   35 36 37 38  39 00 00 00  00 .......00     56789.....
```

d) DR -DESPLEGADO DE REGISTROS.

Cuando el programa con el que estamos trabajando se para por medio de un punto de ruptura, podemos ver el contenido de los registros del procesador gracias al comando DR. El formato es:

-DR

La máquina responde con:

```
SZHVNCE    A =00 BC =0000 DE =0000 HL =0000  ......    LD E,A
SZHVNC     A'=00 BC'=0000 DE'=0000 HL'=0000
```

Las letras SZHVNCE del primer renglón representan las banderas, mientras que las del segundo renglón son las banderas primas. Si la bandera esta prendida, ésta es desplegada, y si esta apagada aparece en su lugar un blanco. Por ejemplo, si solo se prendieron las banderas de "Carry" y "Cero" lo que se desplegara es " Z C ". A continuación esta la descripción de cada bandera.

-S Bandera de signo. S=1 si el bit más significativo del resultado es uno. Es decir, el resultado es un número negativo.

-Z Bandera de Cero. Z=1 si el resultado de una operación es uno.

-H Bandera de "Half Carry". H=1 si el resultado de una suma produce un acarreo en el cuarto bit del acumulador, o si en una resta se produce un "Borrow" en el cuarto bit del acumulador.

-V Bandera de paridad y "overflow". Esta bandera es afectada por operaciones lógicas y aritméticas. Si un "overflow" ocurre durante una operación aritmética, la bandera toma el valor de uno. Después de una operación lógica, la bandera toma el valor uno si el resultado de la operación tuvo una paridad par.

-N Bandera de suma y resta. N será igual a uno si la última operación que se realizó fue una resta.

-C Bandera de acarreo. C=1 si la operación produce un acarreo.

La bandera "E" que aparece en la primera línea es el estado del flip-flop de interrupciones (IFF). Si las interrupciones están habilitadas, una "E" será desplegada, de otra manera aparecerá un espacio. El registro A se despliega a continuación seguido por los registros pares BC, DE y HL y luego el apuntador a la pila. El contador del programa aparece con su valor absoluto y su valor relativo. Y el código de operación al que apunta el

contador del programa es desplegado como una instrucción. En la segunda línea, aparecen los registros primos.

e) B- PUNTOS DE RUPTURA.

El comando B es usado para poner puntos de ruptura a nuestro programa. Usando esta característica podemos detener nuestro programa en algún punto específico cada vez que usamos el comando G. El punto de ruptura permanecerá en esa dirección hasta que sea removido por medio de la instrucción BX.

Se pueden poner hasta doce puntos de ruptura incluyendo los puntos de ruptura permanentes (comando B), los puntos de ruptura temporales (comando G), y los puntos de ruptura de reccorrido (comandos T y C).

Para desplegar los puntos de ruptura permanentes que hemos puesto hay que poner:

B

Para poner un punto de ruptura permanente hay que poner:

B punto de ruptura1 punto de ruptura-2....punto de ruptura-n

Los puntos de ruptura se especifican en cuatro campos:

Campo	Uso
R	Reporte de los registros bandera (opcional).
addr	Dirección de ruptura para puntos de ruptura.
{cond}	Condición para un punto de ruptura (opcional).
;count	Cuenta de repetición para un punto de ruptura.

El registro para reportes de bandera es usado para desplegar los registros cada vez que un punto de ruptura se encuentra durante la ejecución. Este es usado en conjunto con el campo de cuenta repetitiva.

El campo de la condición se usa para especificar un punto de ruptura condicional. Si una condición es especificada el punto de ruptura ocurrira hasta que esta condición se cumpla. La condición puede ser una expresión que involucre registros o localidades de memoria.

El campo para contar repeticiones se usa para detener el programa la enésima vez que el programa cruce por ese punto de ruptura. El valor de este campo es decrementado cada vez que el programa pasa por el punto de ruptura. Si la cuenta llega a cero el programa se para. El valor de este campo nunca vuelve a tomar su valor original. Note que si una condición y una repetición se usan, el campo de cuenta sólo se decrementa si la condición es verdadera.

Hay 128 bytes dedicados a expresiones condicionales de tal manera que todas las condiciones juntas pueden contener un máximo de 128 caracteres.

Si queremos poner un punto de ruptura en la localidad 106H, hay que poner:

```
B 106
```

De esta manera el contador del programa toma el valor de 106H, la ejecución del programa se detiene, y los registros se despliegan en la forma estándar.

Si queremos que el programa se pare cuando pase 5 veces por ese punto de ruptura tendremos que poner:

```
B 106:5
```

Si deseamos que el registro de reporte de bandera sea desplegado ponemos:

```
B R106:5
```

En este caso el registro de reporte aparecerá cada vez que se pase por el punto de ruptura. La máquina se detendrá en la dirección 106H cuando se pase por el punto de ruptura durante cinco veces.

Si queremos poner alguna condicion pondremos:

```
B 106 {^A=0}
```

De esta manera ponemos un punto de ruptura condicional. La ejecución se terminará cuando el contador del programa tome el valor de 106H y el registro A sea igual a cero.

Otro ejemplo es:

```
B 5{[^C=9][^C=A]}:25
```

En este ejemplo el programa detiene su ejecución cuando, después de 25 veces el registro C sea igual a 9H o igual a AH y el contador del programa tome el valor de 5. Los nombres de los registros estan precedidos por el simbolo "^" mientras que los números nó. Cuando estamos depurando un programa que está corriendo bajo CDOS, este punto de ruptura puede tener el efecto de parar el programa en la llamada al sistema número 25H, para imprimir o tomar una línea del buffer.

f) BX - BORRADO DE PUNTOS DE RUPTURA PERMANENTE

Este comando es usado para borrar puntos de ruptura permanentes. Tiene dos formatos:

```
BX
BX dirección-1 dirección-2.....
```

El primer formato borra todos los puntos de ruptura permanentes. El segundo formato borrara los puntos de ruptura que se especifiquen en las direcciones.

g) G- EJECUCION DEL PROGRAMA.

El comando G tiene el siguiente formato:

G dirección de inicio/Punto de Ruptura-1 ... punto de ruptura-n

Cada una de las direcciones es opcional. Si la dirección de inicio es omitida, entonces el contenido del contador del programa es usado. Los registros son cargados con los registros del usuario (Estos son desplegados por el comando DR). La ejecución se empieza en la dirección de inicio o en la dirección que tiene el contador del programa. Si un punto de ruptura se especifica, un RST 30H se inserta en esa dirección y un brinco en esa localidad es puesto. Cuando un punto de ruptura se ejecuta, el control se regresa al depurador, y todos los registros del programa del usuario son salvados y desplegados. Todos los puntos de ruptura temporales son removidos del programa del unsuario. Cuando se utilice este comando hay que tener en cuenta:

g.1 Un punto de ruptura temporal (como su usa en el comando G) puede ser especificado de la misma manera y con los mismos campos con los que se utilizan los puntos de ruptura permanentes.

g.2 Los puntos de ruptura sólo pueden ser puestos en programas que residen en memoria RAM. Esto se debe a que un RST 30H es insertado en la localidad del punto de ruptura. El contenido de esa localidad es salvado y restablecido una vez que el punto de ruptura fue ejecutado.

g.3 Se pueden poner hasta 12 puntos de ruptura. Si queremos poner mas de 12, el comando no es aceptado. Los 12 puntos de ruptura pueden estar compuestos de puntos de ruptura permanentes y puntos de ruptura temporales.

g.4 Cuando se usa un punto de ruptura, una instrucción de brinco se almacena en la dirección 30H. De esta manera, las localidades 30H, 31H, y 32H no estan disponibles para el usuario.

h) A- ENSAMBLADO EN LA MEMORIA.

El formato de esta instruccion es:

A

A beginning-adrr

Este comando permite al usuario teclear nemónicos de lenguaje ensamblador desde la consola y tener su ensamblado en la memoria. El primer formato ensambla en la memoria desde la dirección de default (100H). El segundo formato comienza a partir de la dirección de inicio indicada por el usuario.

El DEBUG responde con la dirección absoluta, la dirección relativa, y la instrucción 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 instrucción ésta se ensambla y el usuario recibe como respuesta la siguiente instrucción. Las reglas para direccionar una expresión también se aplican a las direcciones en nemónicos. En el siguiente ejemplo el registro @ contiene la dirección 1234H

```
A@ 40
1274    0040'   NOP     ADD B
1275    0041'   NOP     CALL @93
1278    0044'   NOP     JP 1032+95
127B    0047'   NOP     .
```

Si sólo tecleamos un RETURN, el Debug no altera la instrucción que existe en esa localidad y va a la siguiente instrucción en la memoria.

Este comando se termina iniciando la línea con un '.' Si existe un error en la línea que se tecleo, ésta no es aceptada y una marca es mandada a la consola.

i) S- SUBSTITUCION DE MEMORIA.

Este comando sirve para cambiar las localidades de memoria. Su formato es:

```
S
S dirección
```

Si utilizamos el primer formato entonces la última localidad que se sustituya (inicialmemte 100H) será la dirección de inicio.

El depurador despliega la dirección absoluta, seguida por la direccion relativa, y luego por el contenido de ese byte. Tiene las siguientes opciones:

i.1 Si tecleamos un dato seguido por un RETURN, el dato es almacenado en la dirección del prompt. La dirección se incrementa en uno y se despliega la siguiente línea.

i.2 Si ponemos un string colocado entre apóstrofos (') seguido por un RETURN, el string se almacenará desde el inicio de la dirección del promp. La dirección se incrementa tantos caracteres como hayamos tecleado y se despliega la siguiente línea.

i.3 Si tecleamos un signo de menos (-), la dirección se decrementa y se despliega la localidad a la que se regresó.

i.4 Si sólamente ponemos un RETURN, la dirección se incrementa y esa localidad no es afectada.

i.5 Un punto (.) nos sirve para salir de este modo.

En el siguiente ejemplo, asumimos que el registro @ tiene como valor 2800H .

```
-S @100
2900    0100'   32      0
```

```
2901    0101'   17      00
2902    0102'   31      'this is an ASCCI 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.
```

J) L- LISTADO EN MNEMONICOS DE ENSAMBLADOR.

También tenemos un comando para listar el contenido de las localidades de memoria, su formato es:

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

El primer formato lista 16 líneas a partir de la dirección actual. El segundo formato los lista a partir de la dirección que le indicamos. Y el tercero lo hace dentro del rango seleccionado.

La primera dirección del desensamblado es la dirección absoluta. La segunda dirección es la dirección relativa. Si la instrucción desensamblada contiene una dirección, la dirección absoluta se despliega a la derecha de esa línea. En el ejemplo que sigue, el registro @ contiene un 2800H.

```
-L@800  @811
3000    0800'   ADD     A,B
3001    0801'   CALL    3200    (0A00')
3004    0804'   CALL    3243    (0A43')
3007    0807    CALL    3333    (0B33')
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
```

k) T- EJECUCION PASO A PASO.

El formato de este comando es:

```
T
T número de instrucciones
T (expr)
```

El primer formato ejecuta una instrucción del programa. El segundo formato ejecuta el programa según el número de instrucciones que especifiquemos. El tercer formato lo realizará hasta que la expresión sea verdadera (no igual a cero). Después de que todas las instrucciones son ejecutadas los valores de los registros son desplegados.

Sólo a un programa almacenado en RAM se le puede aplicar este comando. Este comando pone un punto de ruptura después de la instrucción, carga los registros y ejecuta la instrucción. Luego se ejecuta el punto de ruptura y los registros son resalvados. Los registros son desplegados y la siguiente instrucción es ejecutada hasta que el número de instrucciones indicadas sean ejecutadas, o la condición sea verdadera, en este caso el indicador para el siguiente comando es desplegado.

Para abortar el comando, presione cualquier tecla de la terminal.

Si encontramos un punto de ruptura durante la ejecución del comando, el comando se terminará.

T {^B>10}

En este ejemplo la ejecución del programa se realiza instrucción por instrucción, desplegando los registros después de cada una, hasta que el registro B toma un valor mayor que 10H.

l) TN- EJECUCION SIN DESPLIEGUE.

El formato de este comando es:

TN
TN número de instrucciones
TN {expr}

Este comando es similar al T excepto que la información no es desplegada conforme el comando se ejecuta. Cuando se termina éste, la información de los registros aparece en la forma estandar.

m) TJ- EJECUCION CON BRINCOS.

Su formato es:

TJ
TJ número de instrucciones
TJ {expr}

El comando TJ es similar al T excepto que los puntos de ruptura son sólo puestos en el programa después de las instrucciones que alteran el contador del programa (JP, JR, CALL, y RET). Los registros serán desplegados, y el número de instrucciones contadas serán decrementadas o la expresión evaluada sólo cuando el contador del programa sea alterado. Por ejemplo:

TJ

esta instrucción ocasiona que el programa se ejecute hasta que el contador del programa sea alterado por alguna de las instrucciones mencionadas. Luego de ésto los registros son desplegados y aparece el prompt para dar algún otro comando.

n) TNJ- EJECUCION CON BRINCOS Y SIN DESPLIEGUE.

```
TNJ
TNJ número de instrucciones
TNJ {expr}
```

Este comando es similar al comando TJ excepto que la información no es desplegada conforme se ejecuta el comando. Cuando el comando termina, la información de los registros es desplegada en la forma estandar.

o) Sr- SUBSTITUCION DE REGISTROS.

El comando de Sr permite cambiar el valor de los registros. La letra r indica el registro que será cambiado. Cuando deseamos cambiar el registro F o F', teclearemos SF o SF'. El depurador desplegará las banderas que están prendidas, y esperará a que el programador le indique qué banderas serán prendidas y cuales apagadas. En el siguiente ejemplo, al aplicar el comando vemos que las banderas que están prendidas son SZHC. Nosotros queremos prender solamente Z y C. Para lograrlo ponemos:

```
-sf
SZH C zc
```

Las letras minúsculas son las que el programador teclea.

Cuando substituimos un registro de un byte, sólo el valor de un byte es aceptado. Lo mismo sucede para los registros de dos bytes. Por ejemplo, SD permite que el valor del registro D sea cambiado, con SE cambiamos el valor del registro E, y con SDE cambiamos el valor de D y de E.

Si no se da ningún valor, o si un error ocurre, el valor de los registros no se altera. En el siguiente ejemplo, el valor del registro A se cambia a 41H.

```
-sa
A=98 41
```

P) TERMINACION DEL DEPURADOR.

Para terminar de utilizar el Debug teclee:

```
-^C
```

el caracter de control CTRL-C y la máquina responde con:

```
B.
```

Con esto terminamos la sesión. Para mayor información sobre todos los comandos que se utilizaron, favor de consultar los manuales de referencia.

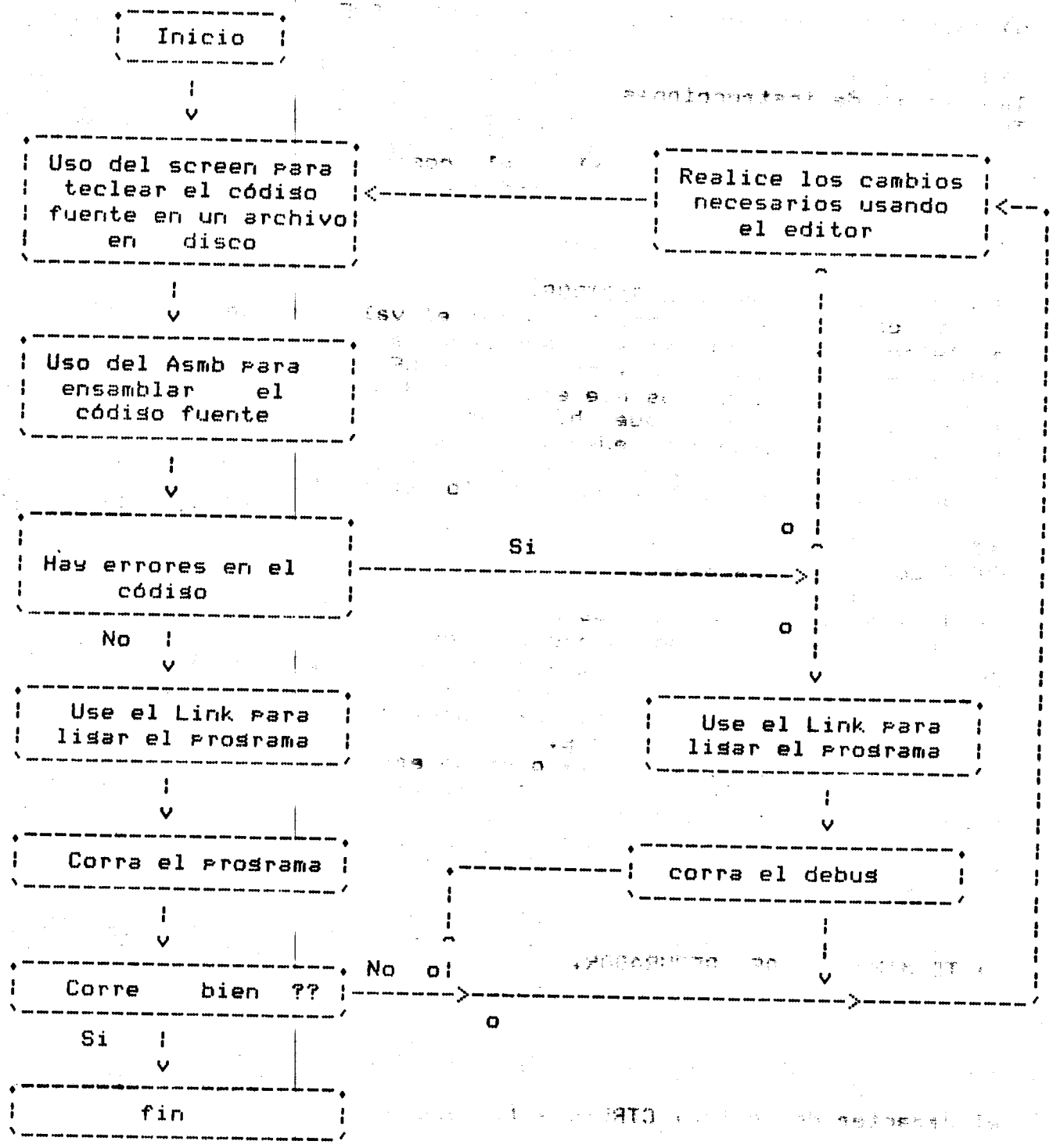

Inicio
Uso del screen para teclear el código fuente en un archivo en disco
Realice los cambios necesarios usando el editor
Uso del Asmb para ensamblar el código fuente
Hay errores en el código
Si
No
Use el Link para ligar el programa
Use el Link para ligar el programa
Corra el programa
corra el debus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

FIGURA (1)

5.- Realice un programa que convierta un numero de 8 bits en binario a un numero en codigo gray, realizando los pasos anteriores (del 1 al 4).

CUESTIONARIO PREVIO

1.- Qué ventajas tiene el utilizar la microcomputadora Cromemco, en lugar del Starter Kit?

2.- Explique tres comandos del editor Screen que no hayan sido explicados en la práctica.

3.- Para qué sirve el ligador?

4.- Qué es el Debug? y para qué sirve?

5.- Qué extensión tienen los programas que al ligarse forman el programa "nombre.COM"?

6.- Qué diferencias existen entre los puntos de ruptura del Starter Kit y los de la Cromemco?

7.- En que casos no es posible correr un programa en ensamblador en la Cromemco, que pueda ser corrido en el Starter Kit?

8.- Para qué sirve el flip-flop de interrupciones?

9.- Comó le asignamos el valor a una localidad de memoria por medio del Depurador?

10.- Qué instrucciones se utilizan para poder hacer uso de los registros primos?

PRACTICA # 6

Z-80 PIO Controlador entrada/salida en paralelo

OBJETIVO.

El objetivo de esta práctica 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, además es una interface entre el procesador y los periféricos. Estos periféricos 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 realizan usando la lógica de control de interrupciones. Esta lógica es usada por los circuitos de soporte del procesador Z-80. Otra característica 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 estado 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 señales de enlace (READY y STROBE). Estas señales controlan la transferencia de datos .

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

MODO 0: Este modo se usa para enviar información del CPU a los periféricos a través de las 8 líneas del puerto seleccionado.

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

MODO 2: En este caso sólo se puede programar al puerto A debido a que utiliza todas las líneas 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 evaluación. Puede ser usado por ambos puertos. Las señales de enlace no se ocupan. El usuario puede programar que líneas 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 lógica de control interna, y la lógica de entrada / salida para los puertos. Una aplicación típica puede ser la de usar el puerto A para transferencia de datos y el puerto B para la evaluación 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 selección de entrada/salida de 8 bits, un registro de dos bits de control de máscara y un registro de máscara. Los últimos tres registros se usan sólo 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 operación :'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 periférico.

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

El registro de control de máscara (2 bits), es cargado por el CPU para especificar el nivel activo (alto o bajo) de alguno de los 'PINS' de un periférico que se esten monitoreando, además indica si una interrupción puede ser generada cuando todos los 'PINS' esten activos (condición AND) o bien cuando alguno de ellos este activo (condición de OR).

El registro de máscara (8 bits), se carga por el CPU para determinar que 'PINS' de algún periférico se están monitoreando para una condición de estado específico.

El registro de selección de entrada/salida (8 bits), es cargado por el CPU para permitir que algún 'PIN' sea utilizado como de entrada o de salida durante el modo de operación de bit.

FIG. 2

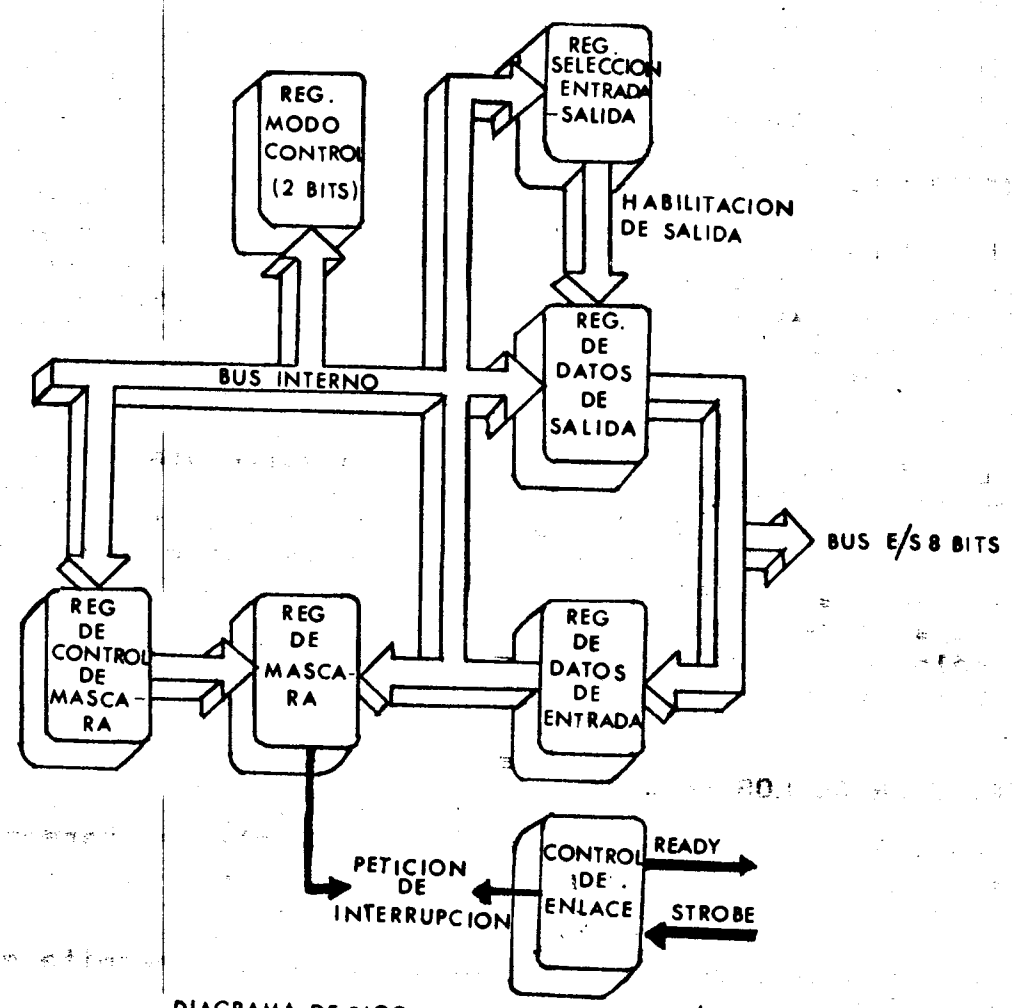


DIAGRAMA DE BLOQUES DE UN PUERTO DE E/S

FIG. 1

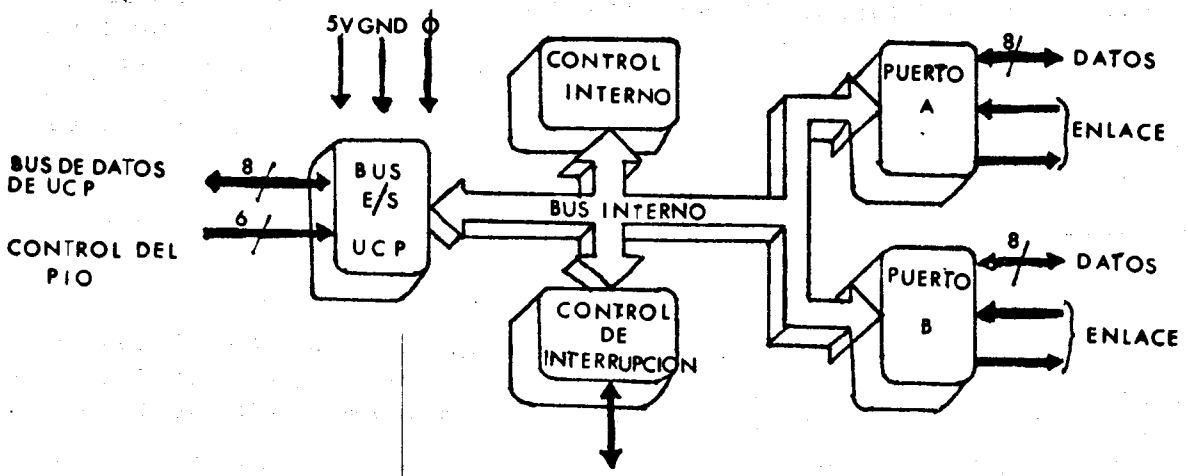
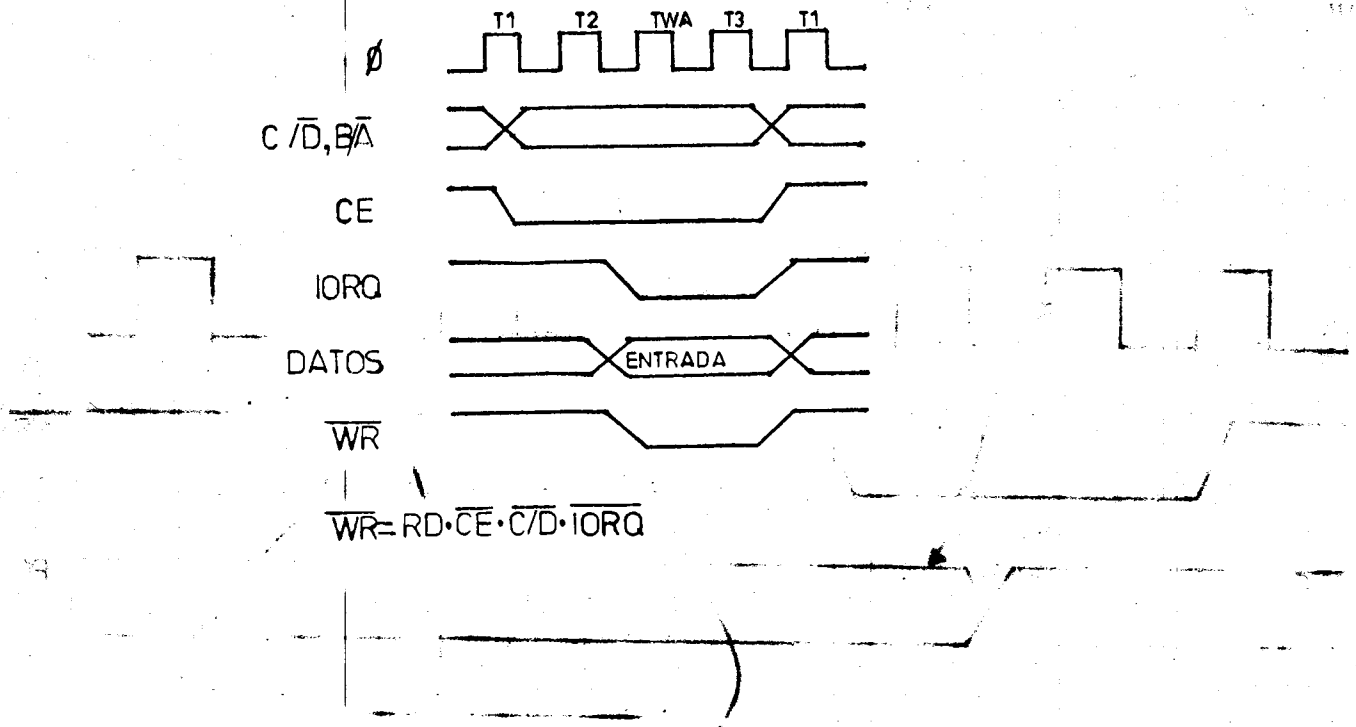


DIAGRAMA DE BLOQUES DE PIO

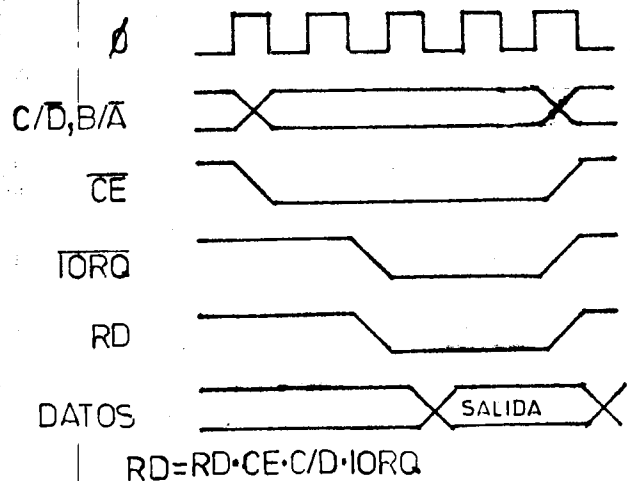
CICLO DE ESCRITURA.

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 éstos se asignan 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).



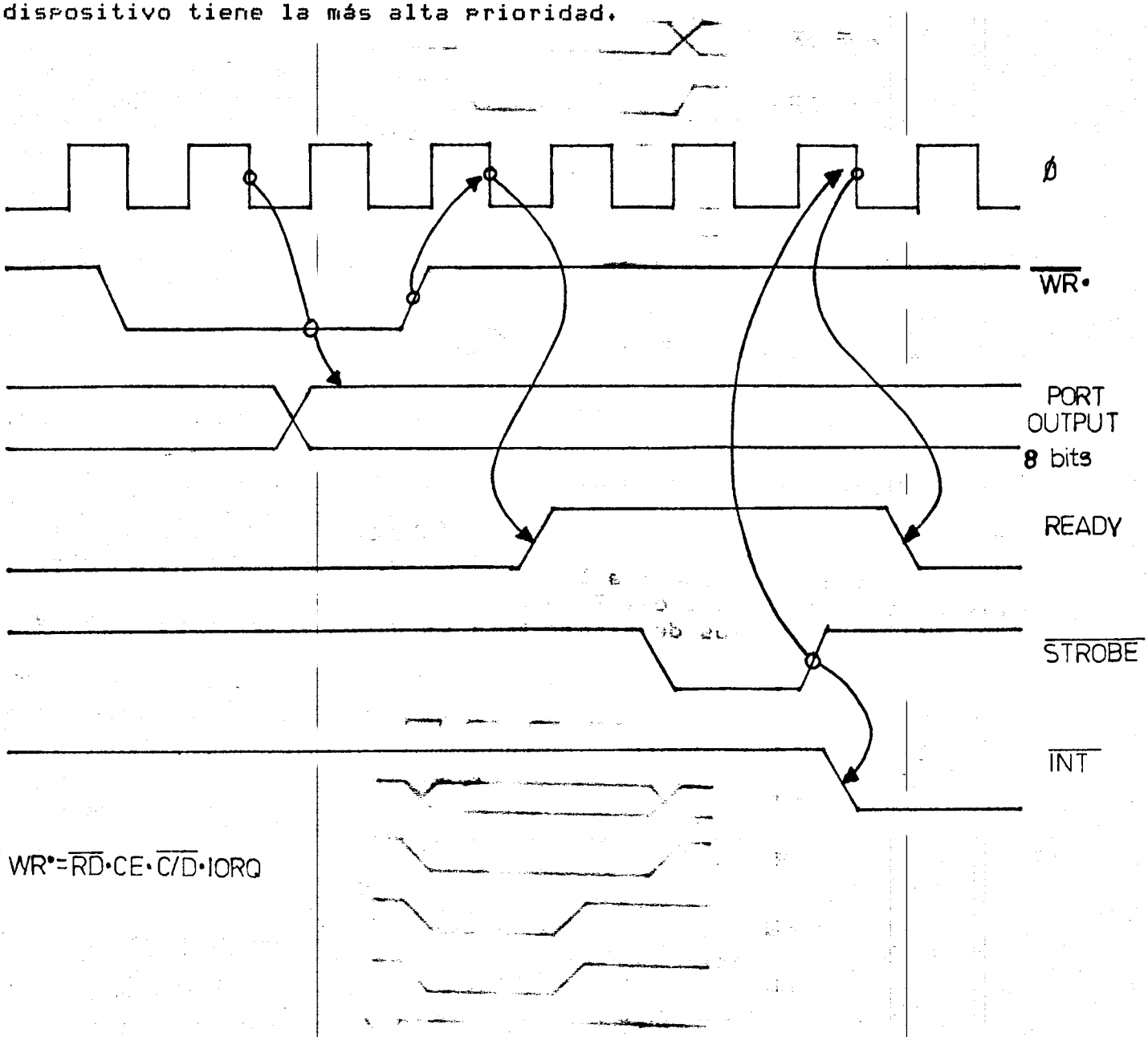
CICLO DE LECTURA.

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 asignación de tiempos de espera WAIT dado que automáticamente se inserta TWA.



MODO SALIDA.

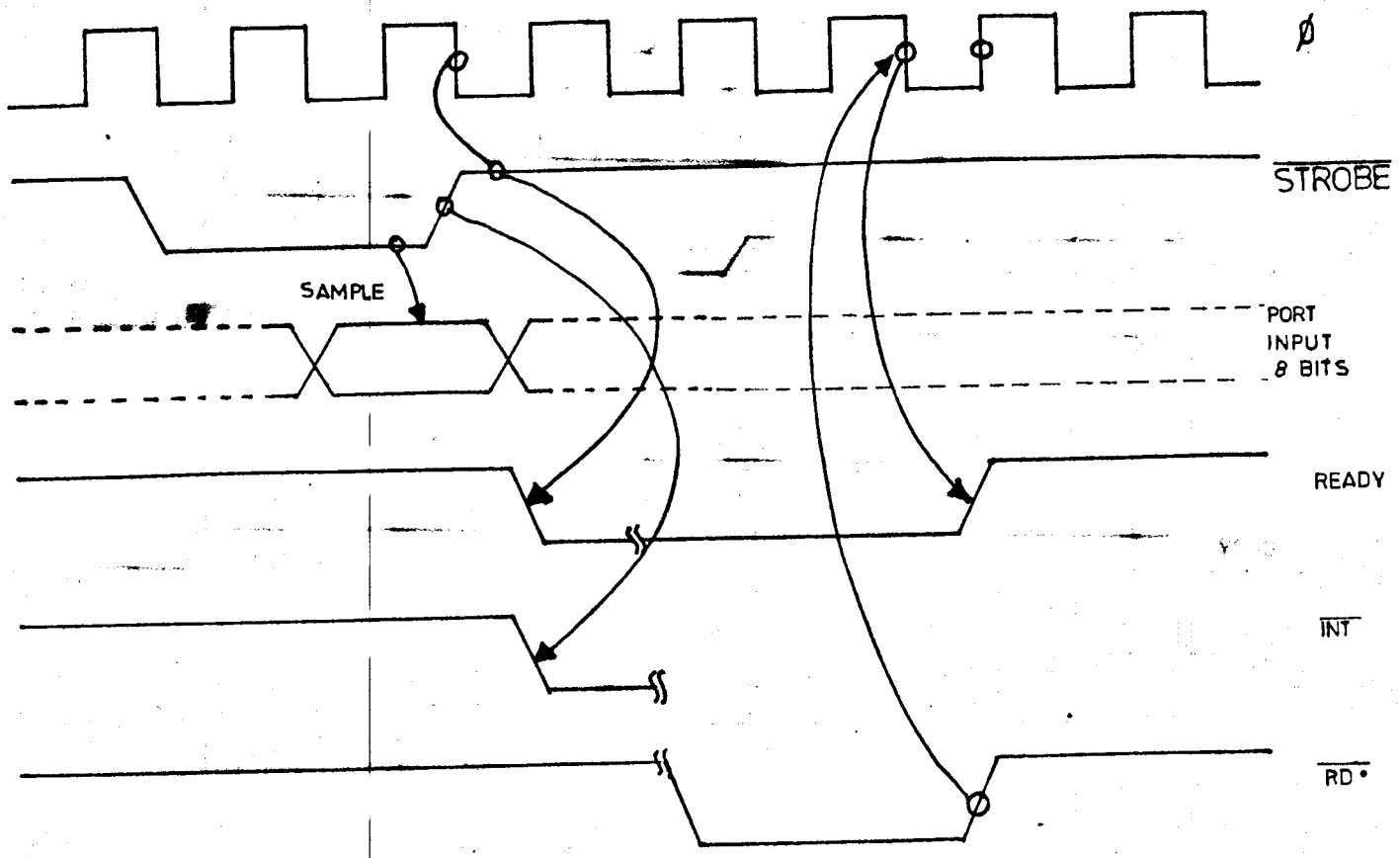
Un ciclo de salida se iniciará con la ejecución de una instrucción de salida por el CPU. La señal WR (negada) del CPU, guarda los datos del canal del CPU en el registro de salida del puerto seleccionado. El pulso de escritura, prende la bandera de READY en el flanco de bajada del reloj, indicando que un dato esta disponible. La señal de READY estará activa hasta que el flanco de subida de la línea de STROBE sea recibido, indicando que un dato fué tomado por el periférico. El flanco de subida del pulso de STROBE genera la señal INT (negada), si el FLIP FLOP que habilita las interrupciones ha sido prendido, y si éste dispositivo tiene la más alta prioridad.



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

MODO DE ENTRADA.

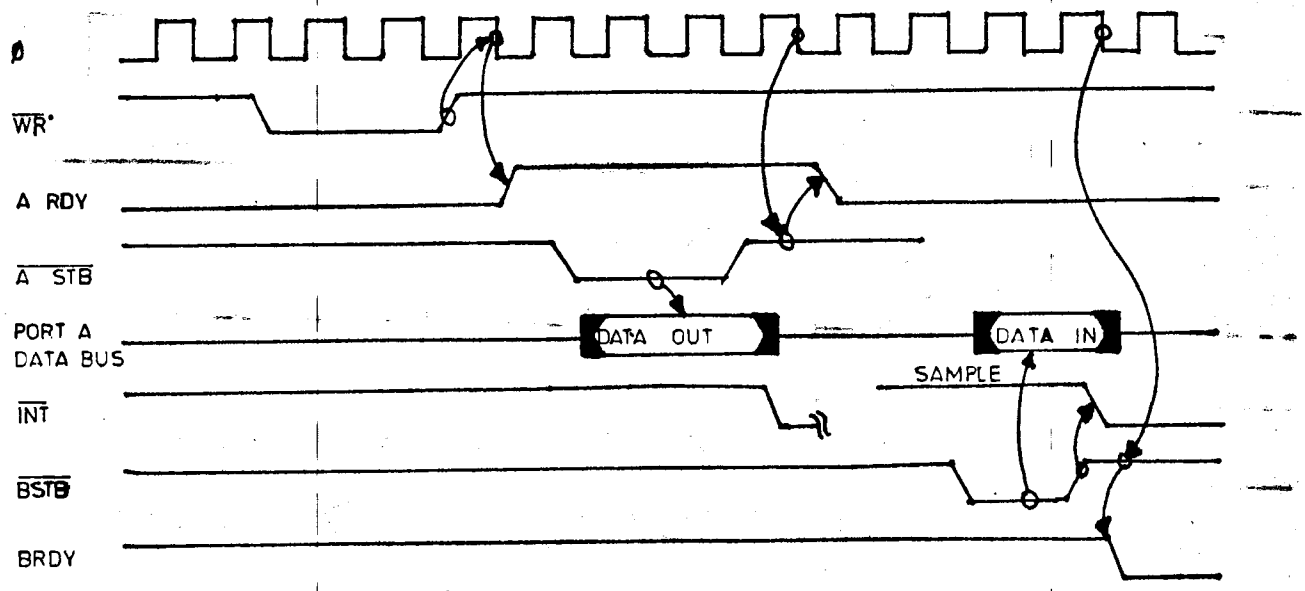
Cuando la señal de STROBE (negada) es baja, los datos son cargados en el registro de entrada del puerto selecionado. Cuando sube el flanco de la señal de STROBE (negada), la señal de INT (negada) es activada si la habilitación de interrupciones esta prendida y además el dispositivo que la requiere tiene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



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

MODO BIDIRECCIONAL.

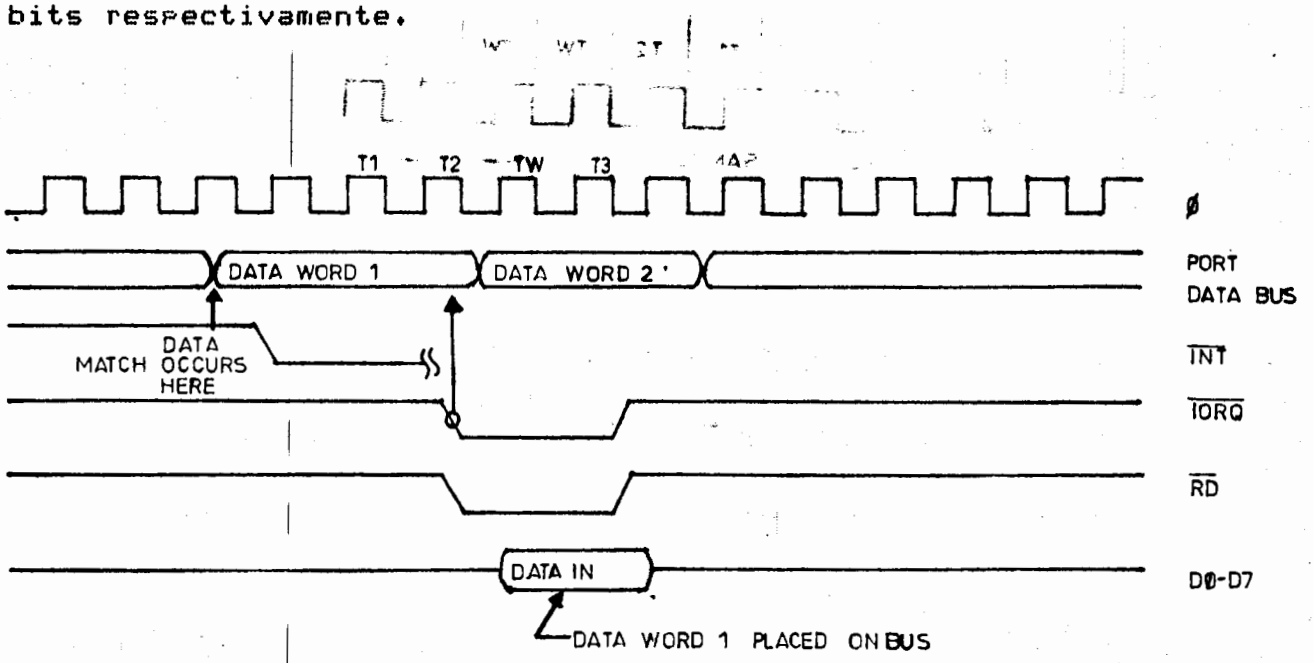
Esta es una combinación de los modos 0 y 1 que usa las cuatro líneas de enlace y las 8 líneas de entrada/salida del puerto A. El puerto B se puede utilizar en el modo bit. Las líneas de enlace del puerto A se usan para controlar la salida y las líneas del puerto B se usan para el control de la entrada. Los datos son puestos en el canal 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.



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

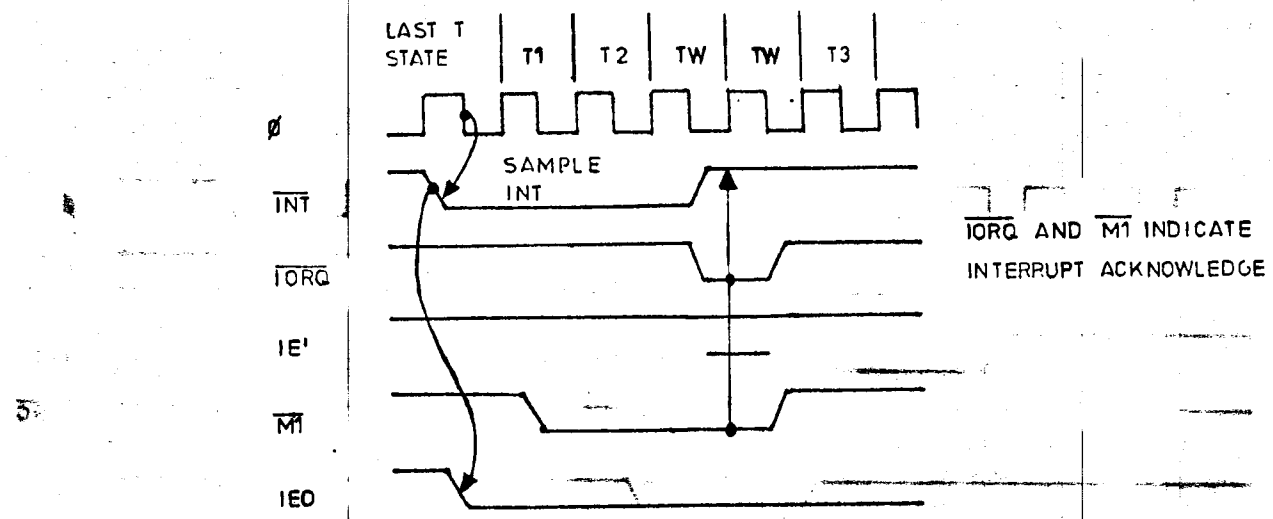
MODO BIT.

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 lea, los datos regrezarán al CPU compuestos de los registros de datos de salida cuyas líneas del puerto sean designadas como líneas de salida, y de registros de entrada de datos por las líneas del puerto que han sido designadas como lineas de entrada. El registro de entrada puede contener datos que fueron presentados inmediatamente después de que el flanco de bajada de la señal RD (negada) ocurre. Una interrupción será generada, si las interrupciones de un puerto están habilitadas, y los datos en las líneas de los puertos satisfacen la ecuación lógica definida por los registros de control de máscara de 2 y de 8 bits respectivamente.



RECONOCIMIENTO DE INTERRUPCIONES.

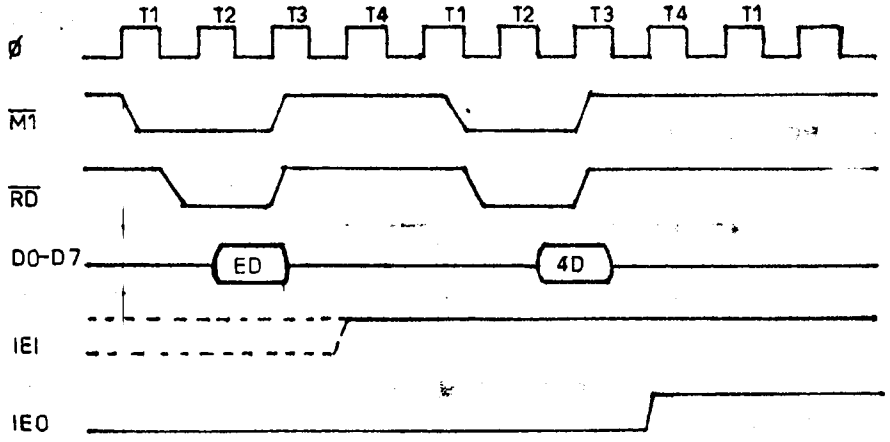
Durante el tiempo que dura M1 (negada), los controladores de los periféricos se inhiben cambiando el estatus que habilita las interrupciones, permitiendo que la señal habilitada de INT pase a traves de la cadena de margarita. El periférico que tiene IEI alto e IEO bajo, durante el tiempo que dura INTA (negada) pone el vector de interrupciones preprogramado (8 bits) en el canal 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 periférico del Z-80 no tiene una interrupción pendiente y no está en servicio, entonces su IEO=IEI. Si este tiene una interrupción bajo servicio (por ejemplo, si ya interrumpio y recibe un reconocimiento de interrupciones) entonces su IEO siempre es bajo, inhabilitando a los periféricos de prioridad más baja que puedan interrumpir. Si existe una interrupción pendiente que no haya sido reconocida, IEO será baja hasta que un 'ED' sea decodificado como el primer byte del código de operación de una instrucción de dos bytes. En este caso IEO subirá hasta que el próximo byte del código de operación sea decodificado, luego de esto bajará. Si el segundo byte del código de operación fue un '4D' entonces la instrucción es un RETI.

Después de que el código 'ED' es decodificado, sólo el periférico que está interrumpiendo y que se está atendiendo tendrá la señal IEI alta e IEO baja. Este dispositivo es el de mayor prioridad en la cadena margarita que ha recibido un reconocimiento de interrupción. Los demas periféricos tienen IEI=IEO. Si el proximo byte de el código de operación a decodificar es '4D' este dispositivo iniciará su condición 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 dirección de la rutina de atención de interrupciones del puerto usando este vector. Durante el ciclo de reconocimiento de una interrupción, el vector es puesto en el canal de datos del CPU por el dispositivo que tiene la más 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 dirección.

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

dirección =	Registro I	Vector de interrupciones

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

2) Selección del modo de operación.

Cuando seleccionamos un modo de operación, el registro de control de modo (2 bits) puede tomar uno de cuatro valores. Estos bits son los más 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 selección de operación.

D7	D6	D5	D4	D3	D2	D1	D0
M1	M0	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 qué líneas se usarán como entrada y que líneas se usarán como salida. Para tal efecto, la siguiente palabra de control deberá tener en los bits de salida un 0 y en los bits de entrada un 1.

D7	D6	D5	D4	D3	D2	D1	D0

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 — Interrupción deshabilitada.

Bit 7 = 1 — Interrupción habilitada.

D7	D6	D5	D4	D3	D2	D1	D0
Habilita Interrupción	And Or	Alto Bajo	Indicador Máscara	0	1	1	1

Solo se usan en modo 3

Modo 3 Solamente.

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

Bit D4 = 1 La palabra siguiente es la de máscara.

Bit D5 = 0 El nivel activo es el bajo.

Bit D5 = 1 El nivel activo es el alto.

Bit D6 = 0 Interrumpe con una función OR.

Bit D7 = 1 Interrumpe con una función AND.

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

Para el modo bit.

Si el bit 4 de la palabra de control de interrupciones en el modo bit es uno, debe enviarse a continuación la palabra de máscara, en ésta se debe de poner un cero en el bit de entrada que desee monitorearse para poder generar una interrupción.

M7	M6	M5	M4	M3	M2	M1	M0

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

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

IMPORTANTE.

- Cuando se trabaje un puerto con capacidad de interrupción, 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 entrada/salida del puerto A.
81H	Datos de entrada/salida del puerto B.
82H	Datos de control del puerto A.
83H	Datos de control del puerto B.

DESARROLLO.

1.- El instructor explicará el funcionamiento del Z-80 PIO y dara un ejemplo de su programación.

2.- Realice los siguientes programas.

2.1. Control de un motor de pasos tipo HS25 SLO-SYN.

Un motor de pasos es un motor de magneto permanente que convierte señales electrónicas a movimiento mecánico. Cada vez que la dirección de la corriente en las bobinas del motor es cambiada, la flecha del motor gira una distancia angular especifica. La flecha puede operar en cualquier dirección.

Estos motores ofrecen muchas ventajas como actuadores en sistemas de posicionamiento controlados digitalmente y son facilmente interconectados con microcomputadoras o microprocesadores .

En particular el motor HS25 posee un rotor de magneto permanente y un estator de 8 polos. Puede ser manejado en pasos de ángulo completo (1.8 grados) o bien a medio ángulo (0.9 grados). La flecha del motor avanza 200 pasos por revolución cuando trabaja a ángulo completo (secuencia de 4 pulsos de entrada), y 400 pasos por revolución en medio ángulo (secuencia de 8 pulsos de entrada). Las bobinas del motor se conectan a la lógica de control a través de transistores de potencia para dar la corriente suficiente. Mientras que sea mantenida una secuencia dada en las fases del motor, la flecha poseerá una alta inercia. Motores de 6 hilos pueden ser usados como motores de 5 hilos conectando los hilos blanco y negro juntos, use una resistencia de la mitad del valor y del doble de la potencia de la requerida en la conexion de 6 hilos.

SECUENCIA DE 4 PASOS PARA ANGULO COMPLETO .

PASO	SW1	SW2	SW3	SW4
1	1	0	1	0

2	1	0	0	1
3	0	1	0	1
4	0	1	1	0
1	1	0	1	0

SW1, SW2, SW3 y SW4 representan las bobinas del motor.

Si se desea que el motor gire en sentido contrario, la secuencia debe ser ..,1,4,3,2,1,4,....

SECUENCIA DE 8 PASOS PARA MEDIO ANGULO

PASO	SW1	SW2	SW3	SW4
1	1	0	1	0
2	1	0	0	0
3	1	0	0	1
4	0	0	0	1
5	0	1	0	1
6	0	1	0	0
7	0	1	1	0
8	0	0	1	0
1	1	0	1	0

Si se desea que gire en sentido contrario, la secuencia debe ser entonces ..,1,8,7,6,5,4,3,2,1,.....

Se pueden generar peligrosos voltajes transitorios en las fases al realizarse la conmutación, la manera usual de protección es usando diodos (IN4002, IN4003 y similares), como se muestra en la figura (1), la resistencia puede variar desde 0 hasta 50 ohms.

El programa debe hacer que la flecha del motor de pasos gire en sentido horario una revolución, una pausa de aproximadamente un segundo, y otra revolución pero ahora en sentido contrario, otro segundo y se repite la misma secuencia de movimiento.

2.2 Convertidores Digital/Analogico, Analogico/digital.

En esta practica se hará uso de un convertidor digital/analogico, un comparador y el PIO para construir un convertidor analogico/digital del tipo contador-rampa.

La figura (2) muestra el diagrama de este tipo de convertidores, cuando se usa un contador, en esta ocasion se utilizará el microprocesador, que enviará la cuenta hacia el PIO. Cuando el voltaje a la salida del convertidor D/A iguala el voltaje analógico desconocido, el comparador cambia de estado, en ese instante, la cuenta en el PIO representa el equivalente digital de la entrada analógica.

El puerto A del PIO se conecta al convertidor D/A, se realiza una realimentación de la salida del comparador, la cual estará siendo sensada por el microprocesador, esta señal se lee a traves de una línea del puerto B (programado en modo bit), cuando el CPU detecte el cambio en esta señal se habrá obtenido el valor digital equivalente al analógico (ver figura 3).

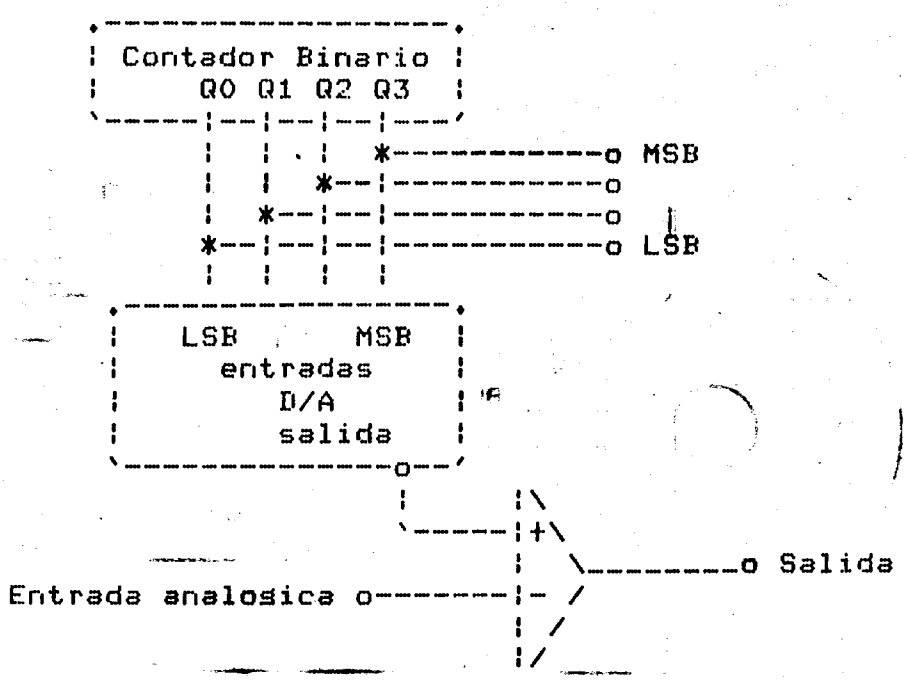


Figura (2)

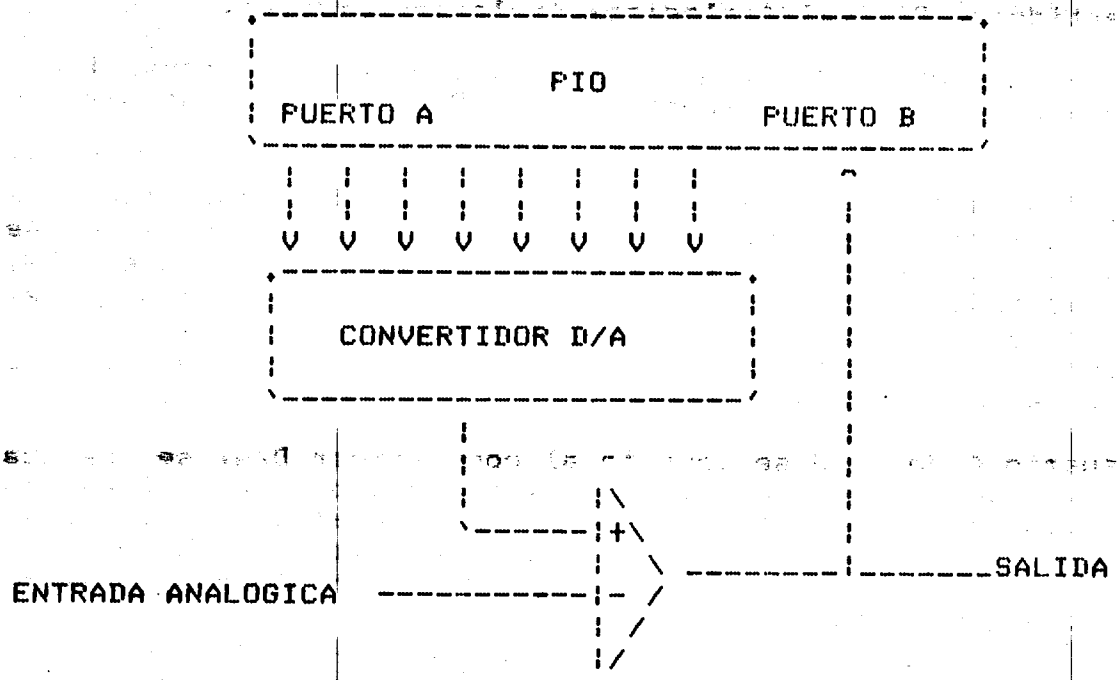


Figura (3)

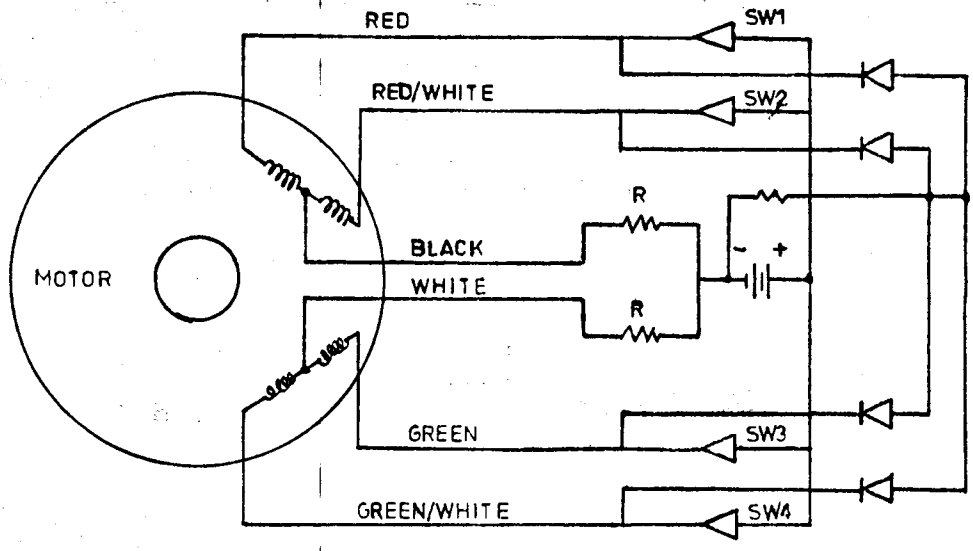


FIGURA (1)

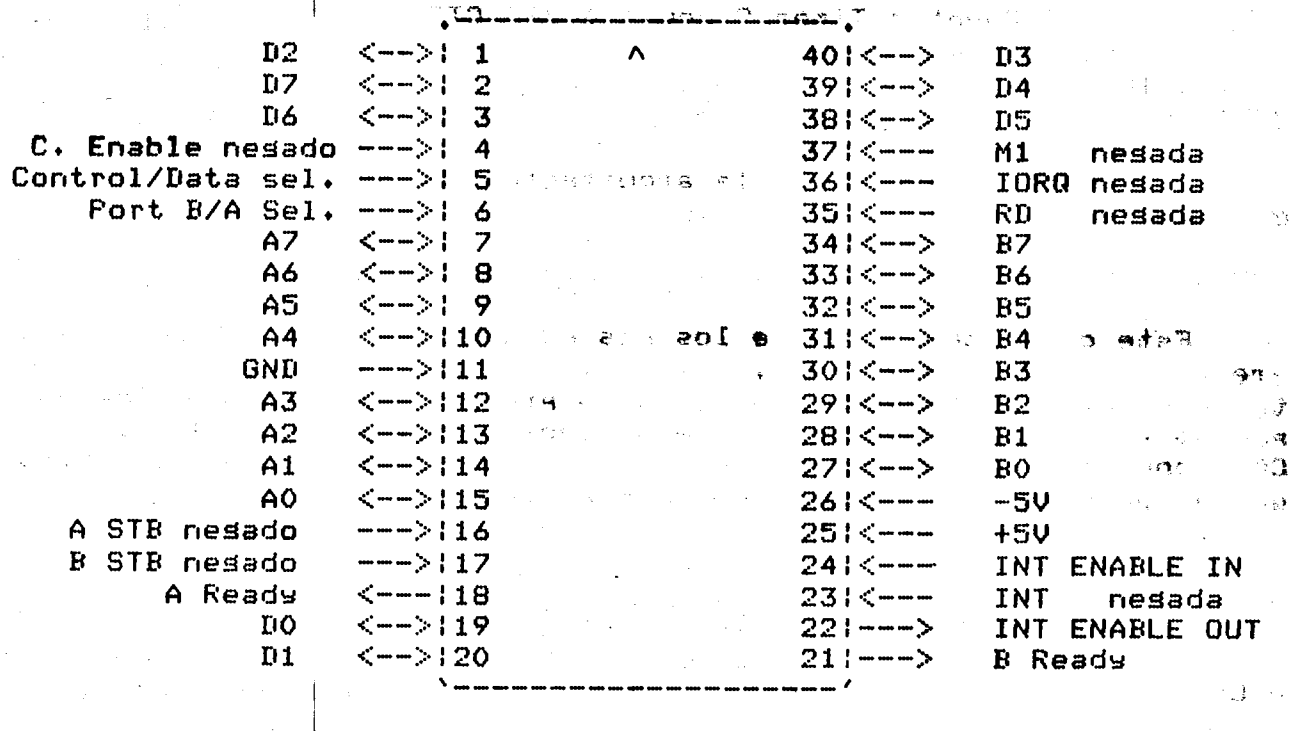


DIAGRAMA DE LAS PATAS DEL Z80-PIO

PRACTICA # 7

Circuito integrado contador y temporizador

(Counter Timer Circuit) Z80-CTC

OBJETIVO.

El alumno aprenderá la arquitectura y el funcionamiento de éste circuito de la familia Zilog .

INTRODUCCION.

Este circuito es uno de los más usados en esta familia y está presente en el Starter-Kit . El C.I. circuito contador y de temporización Z80-CTC es un circuito programable de 4 canales que proveé funciones de conteo y temporización 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 temporización.

- 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 número de la cuenta que va a cero.

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

- 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 interrupción vectorizada sin lógica externa .

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

ARQUITECTURA.

Un diagrama de bloques se muestra en la página siguiente. La estructura interna del Z80-CTC consiste de un canal (bus) que lo comunica con el canal de datos del Z80-CPU. Poseé lógica de control interna, cuatro canales contadores y lógica de control de interrupción. Cada canal tiene un vector de interrupción para interrupción vectorizada automática, y la prioridad de la interrupción es determinada por el número del canal; el canal con la más alta prioridad es el canal 0.

La estructura de canal también se muestra en la siguiente página. Poseé 2 registros, 2 contadores, y lógica 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, éste 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 operación 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 se recarga automáticamente al llegar a cero. En cualquier tiempo el CPU puede leer el número de la cuenta que falta para cero. Este contador es decrementado por la preescala en modo de temporización 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 temporización.

CLK/TRIG = Disparador externo del reloj / temporizador.

= Clock external or timer trigger.

ZC / TO = Cuenta a cero o tiempo final.

= Zero count or timeout.

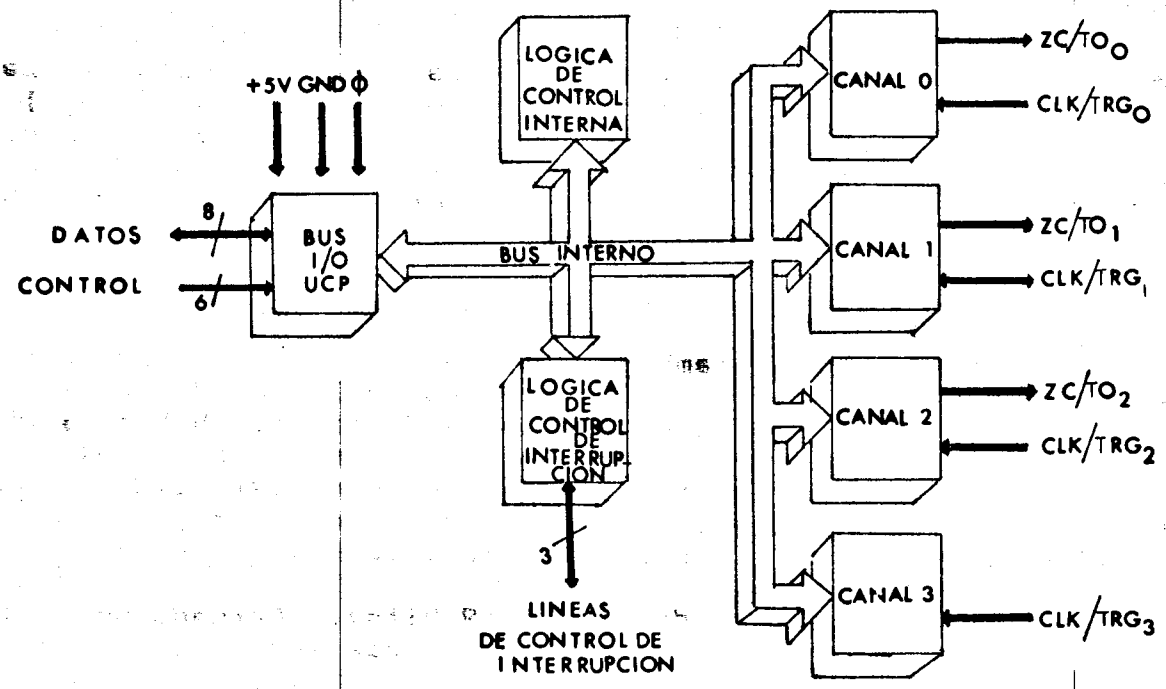


DIAGRAMA DE BLOQUES DEL CTC

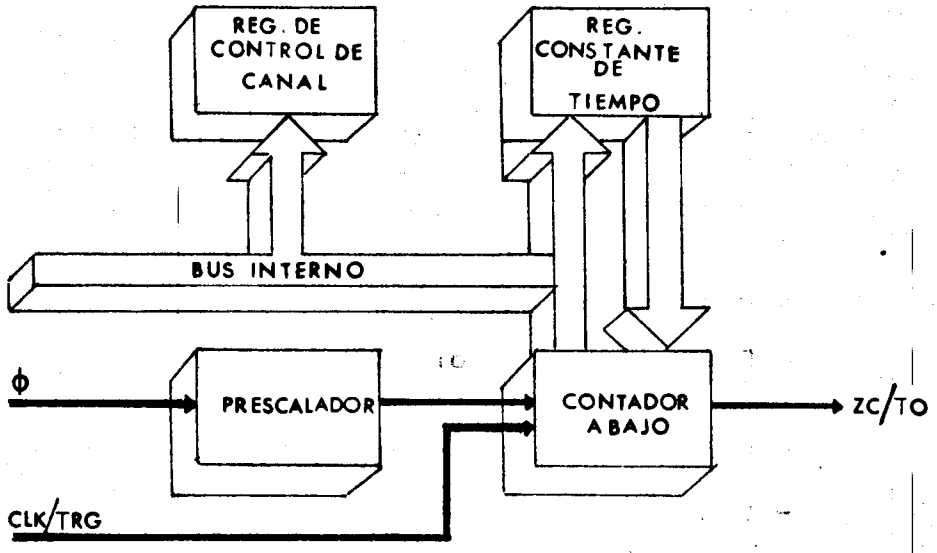
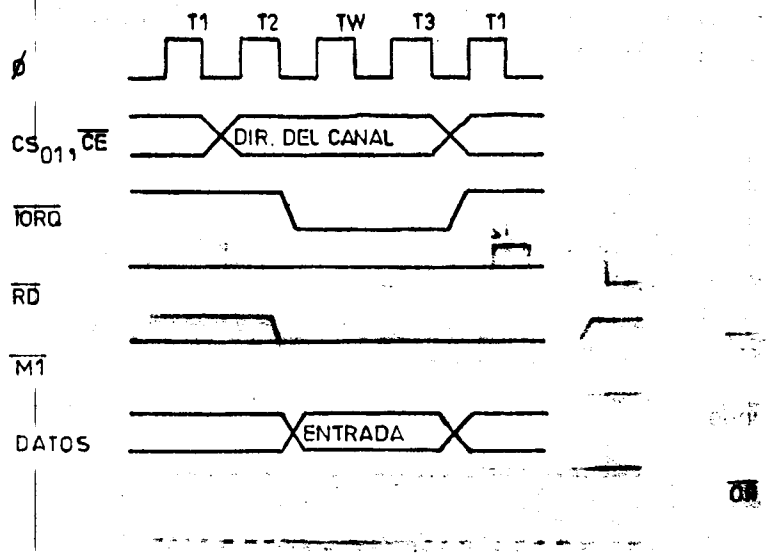


DIAGRAMA DE BLOQUES DE UN CANAL

CICLOS DE TIEMPO DEL CTC-Z80.

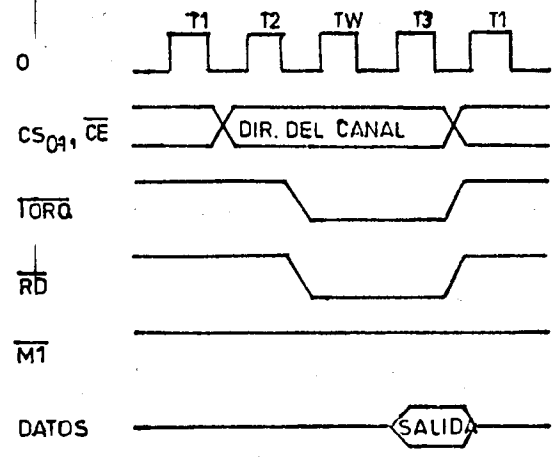
CICLO DE ESCRITURA.

Nos muestra el tiempo para cargar una palabra de control, la constante de tiempo y el vector de interrupción. No se involucran estados de espera (WAIT STATES) para la escritura al CTC más que el que se inserta automáticamente (Tw). Ya que el CTC no recibe una señal específica de escritura, está se genera internamente.



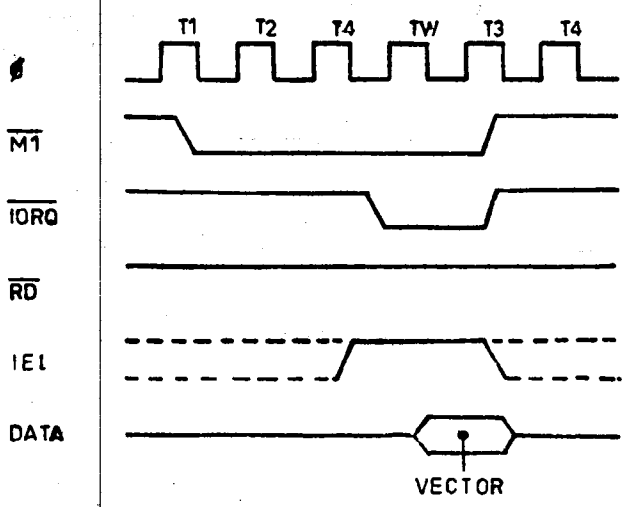
CICLO DE LECTURA.

Se muestran los tiempos que se requieren para leer el contador abajo cuando se está en modo contador . El valor leido sobre el canal de datos refleja el número 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.

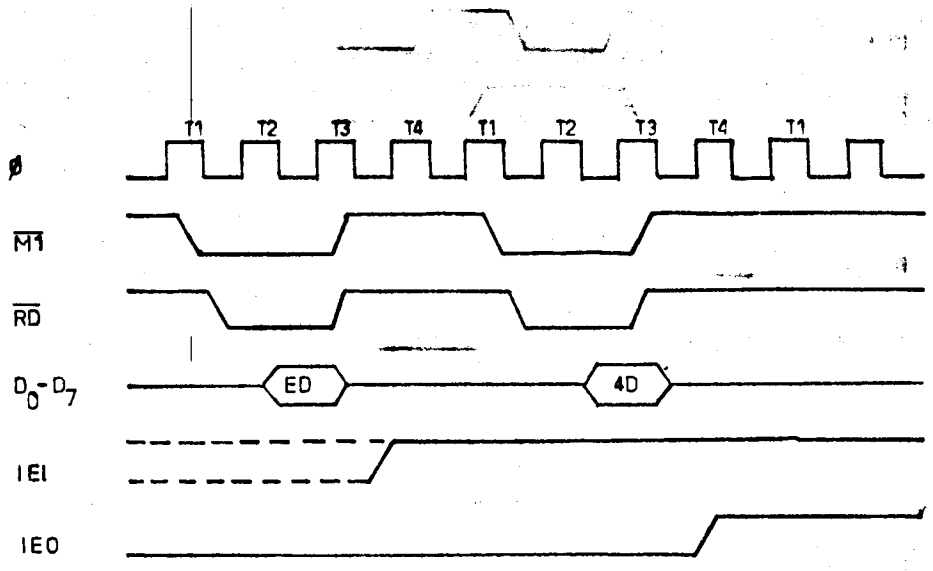
Algún tiempo después que una interrupción es solicitada por el CTC, el microprocesador enviará un reconocimiento de interrupción (M1 y IORQ). Durante éste tiempo la lógica de interrupción del CTC determinará cual es el canal de más alta prioridad que esté solicitando una interrupción. Para asegurarse que las líneas de habilitación de la cadena margarita están estables, los canales son inhibidos a cambiar su estado de requerimiento de interrupción cuando M1 está activa. si la entrada habilitadora de interrupción (IEI) del CTC está activa, entonces el canal de más alta prioridad que está interrumpiendo coloca su vector de interrupción dentro del canal de datos cuando IORQ es activada. Se permiten ciclos adicionales de espera.



CICLO DE RETORNO DE UNA INTERRUPCION.

Si un periférico no tiene interrupción pendiente y no está bajo servicio de interrupción, entonces IEO=IEI. Si está bajo servicio de interupción (ya interrumpió y recibió un reconocimiento de interrupción) entonces IEO está en nivel bajo, inhibiendo los C.I. de menor prioridad de interrupción. Si está pendiente de interrupción, la cual todavia no ha sido reconocida, IEO estará baja a menos que un código de operación ED sea decodificado como el primer byte de un código de operación de 2 bytes. En este caso IEO irá a alto hasta que el byte del siguiente código de operación sea decodificado, con lo cual nuevamente será bajo. Si el segundo byte del código de operación fue 4D entonces el código de operación fue el de la instrucción RETI (RETorno de Interrupción). Después de que un código ED es decodificado, unicamente el periférico que interrumpió y que está bajo servicio de interrupción tendrá su IEI alto y su IEO bajo. Este periférico poseé la más alta prioridad de la cadena margarita que haya recibido un reconocimiento de interrupción. Los demás periféricos tendran IEI=IEO. Si el siguiente byte del código de operación decodificado es 4D, éste periférico presentará un estado de condición bajo servicio.

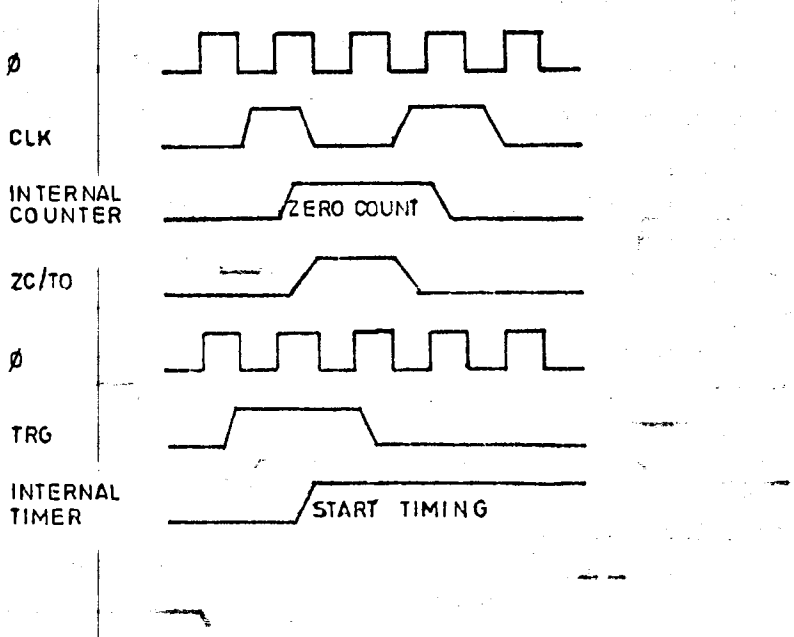
Ciclos de espera son permitidos en el ciclo M1 .



CICLOS DE CONTEO Y TEMPORIZACION .

En el modo contador el flanco de subida o de bajada de la entrada de reloj (CLK) orisina que el contador sea decrementado. El frente es detectado asincrónicamente y debe tener un ancho del pulso de reloj mínimo. 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 (CLK/TRG). Como en el modo contador, el frente es detectado asincrónicamente y debe tener un ancho del pulso TRG mínimo. Sin embargo, la temporización iniciará con el siguiente frente de subida del reloj del sistema. El preescalador cuenta los frentes de onda del reloj.



PROGRAMACION.

1) Seleccionando un modo de operación.

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

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

Bit D7 = 0 Interrupción del canal deshabilitada.

D7 = 1 Interrupción del canal habilitada. Esta ocurrirá 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 operación de temporización.
	Modo contador. Frente negativo decrementa el contador hacia abajo .
D4 = 1	Modo temporizador . El frente de onda positivo inicia la operación de temporización.
	Modo contador . El frente positivo decrementa el contador hacia abajo.
Bit D3 = 0	Modo temporizador solamente . Se inicia la temporización sobre el frente de subida del ciclo T2 del ciclo de máquina siguiente al que carga la constante de tiempo.
D3 = 1	Modo temporizador unicamente . Un disparo externo es válido para iniciar la operación de temporización después del frente de subida del ciclo T2 del ciclo de máquina siguiente al que carga la constante de tiempo . El preescalador.es decrementado 2 ciclos de reloj más tarde si el reloj del sistema está presente , de otra manera son 3 ciclos de reloj.
Bit D2 = 0	Indica que no se envia la constante de tiempo después de la palabra de control. Una constante de tiempo debe ser escrita al canal para iniciar la operación.
D2 = 1	La constante de tiempo para el contador a cero será la siguiente palabra escrita al canal seleccionado . Si una constante de tiempo es cargada durante el conteo , la cuenta presente se completará 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 operación . Si el bit D2 es 1 el canal reasumirá el conteo después de haberse cargado una constante de tiempo de otra manera una nueva palabra de control debe ser cargada.

2) Cargando una constante de tiempo.

Una constante de tiempo de 8 bits es cargada al registro de la constante de tiempo, a continuación 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

3) Cargando un vector de interrupción.

El microprocesador Z-80 requiere que un vector de interrupción (8 bits) sea alimentado por el canal interruptor (cuando la interrupción es modo 2 , como en nuestro caso). El microprocesador forma la dirección de la rutina de servicio de interrupción del canal usando éste vector. Durante un ciclo de reconocimiento de interrupción, el vector es colocado sobre el canal de datos del microprocesador por medio del canal de más alta prioridad que está requiriendo servicio en ese instante. El vector de interrupción deseado es cargado al CTC, escribiendo al canal 0 , un 0 en D0 . D7 - D3 contienen el vector de interrupción deseado y , D2 y D1 no son usados al cargar el vector. Cuando el CTC responde a un reconocimiento de interrupción, estos dos bits contienen el código binario del canal de más alta prioridad.

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

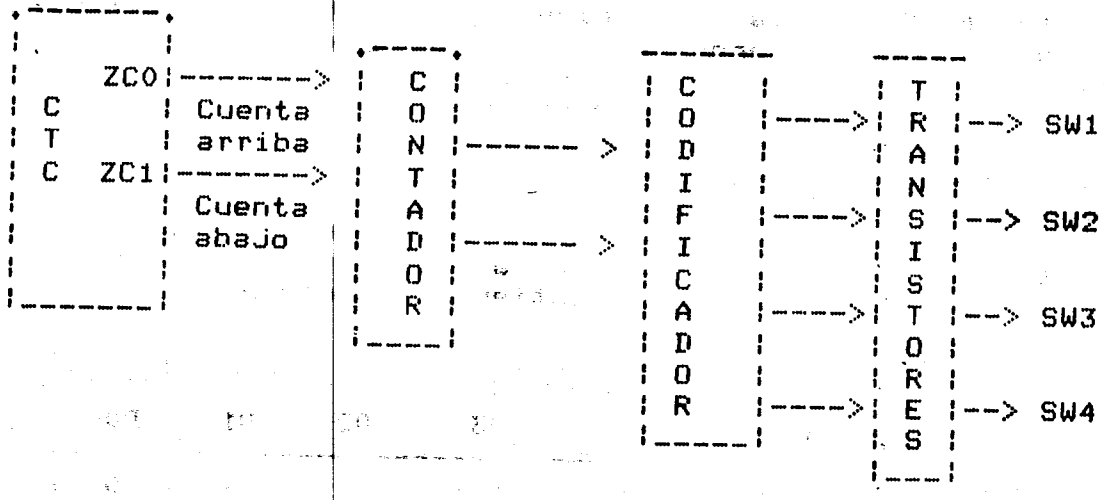
DESARROLLO.

1.-El profesor explicará la arquitectura y el funcionamiento del Z80-CTC y dará un ejemplo de su programación.

2.-Realizar las dos prácticas que a continuación se explican .

a).-CONTROL DE UN MOTOR DE PASOS EN TIEMPO REAL .

Ya vimos que se requiere enviar una secuencia de valores a las fases del motor de pasos para hacerlo girar cierto ángulo, sin embargo hacerlo con el PIO puede llevarse mucho tiempo del microprocesador, por el contrario, si se usa el CTC, puede programarse éste para que genere una señal que habilite periodicamente el movimiento del motor, la figura nos muestra un diagrama de bloques que controla el movimiento del motor.



Se puede usar como en la figura un canal del CTC para habilitar la cuenta arriba en el contador y otro de los canales para decrementar el contador. Dependiendo si el contador se está incrementando o decrementando, la entrada al codificador cambia, a su vez éste debe dar el código para mover hacia adelante o hacia atrás la flecha del motor .

b).-PROGRAMA DE RELOJ DIGITAL .

Se debe generar un reloj en los despliegues del Starter - Kit, para ello se programa al CTC de tal manera que cada X tiempo interrumpa al microprocesador, éste debe llevar la cuenta de las interrupciones y, al tenerse el equivalente a un segundo debe actualizarse la hora mostrada por los despliegues. En los 6 despliegues debe mostrarse la hora, minutos y segundos. El reloj debe tener la capacidad de contar desde las 00:00:00 hasta las 23:59:59 .

G- 613183

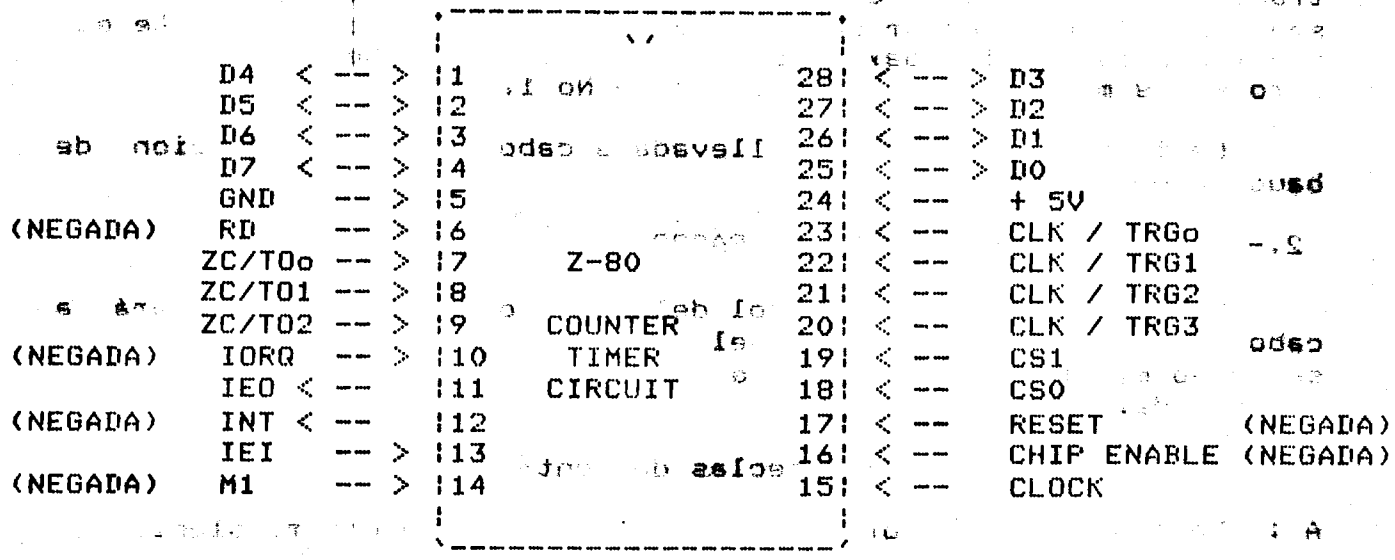


DIAGRAMA DE LAS PATAS DEL CTC-Z80

Donde IEO = INT. ENABLE OUTPUT

IEI = INT. ENABLE INPUT

CLK/TRG = DISPARO O RELOJ EXTERNO

ZC/TO = CUENTA A CERO O TIEMPO FINAL

PROYECTOS FINALES.

INTRODUCCION:

El alumno hará uso de los conocimientos adquiridos durante el semestre, los aplicará en proyectos donde deberá manejar la capacidad de los periféricos para poder comunicar al microprocesador con el exterior.

DESARROLLO.

1.- TRANSMISION SERIAL DE UN STARTER-KIT A OTRO .

En la familia ZILOG Z-80 existe un circuito llamado Z-80 SIO, el cual permite la comunicación serial con el microprocesador Z-80; ya que el Starter Kit no lo tiene se hará un programa que transfiera información de un kit a otro, usando el PIO. El programa deberá estar sensando el teclado, cuando se detecte que una tecla es presionada, su código hexadecimal debe ser enviado al otro kit y mostrado por el despliegue No 1.

La transmision debe ser llevada a cabo con una relacion de baudaje de 1200 .

2.- CONTROL DE UN MOTOR DE PASOS .

En ésta ocasión el control del motor de pasos se llevará a cabo a través del teclado y del PIO, el programa principal estará sensando el teclado, hasta que alguna tecla de control sea presionada.

Se definen las siguientes teclas de control :

A : La flecha del motor gira en sentido horario una revolución y se detiene.

B : La flecha del motor gira en sentido antihorario una revolución y se detiene.

C : La flecha del motor girará en sentido horario.

D : La flecha del motor girará en sentido antihorario.

E : Detiene el funcionamiento del motor, además en éste instante se puede modificar la velocidad de rotación de la flecha del motor a través de las teclas numéricas.

F : Termina la rutina o comando de la tecla E.

3.- ACCESO DIRECTO A MEMORIA.

Diseñar e implementar el circuito que permita el acceso directo a la memoria RAM en el Starter - Kit. De ésta manera se podrá introducir información a cualquier localidad de memoria, ésto debe ser hecho en forma independiente del teclado hexadecimal. El microprocesador podra estar corriendo el monitor Z-BUG o bien corriendo algún programa del usuario.

4.- SINTESIS MUSICAL USANDO EL CTC.

4.1.- Diseñar el circuito amplificador que permita la conexión de un canal del CTC a una bocina, de tal manera que se pueda escuchar una señal generada por dicho canal.

4.2.- Programar al CTC de tal manera que genere señales periódicas, éstas señales deben corresponder en frecuencia a notas musicales.

4.3.- Acoplar el canal programado al amplificador y correr el programa, deberá escucharse una melodia por la bocina.

5.- SEMAFOROS INTELIGENTES CONTROLADOS POR UN MICROPROCESADOR.

Diseñar un sistema digital con el Starter - Kit, que controle el cruce de dos avenidas, el sistema deberá tener las siguientes características.

5.1.- Deben existir sensores de tráfico sobre las dos avenidas.

5.2.- La avenida principal siempre tiene prioridad, pero la avenida lateral debe ser servida cuando exista tráfico continuo.

5.3.- El semáforo de la avenida principal deberá estar por lo general en luz verde.

5.4- Si un vehículo es detectado sobre la avenida lateral, el microprocesador deberá checar primero, si ya han pasado 60 segundos desde la última vez que se le dió la señal de siga a la avenida lateral.

5.5.- Si ya pasaron 60 segundos de que la avenida lateral tuvo el siga, la señal deberá cambiar a la luz verde para la Av. lateral, después de 3 segundos en que la avenida principal estuvo con la luz ambar y la lateral en rojo.

5.6.- La avenida lateral no deberá estar con luz verde por un lapso mayor a los 60 segundos.

5.7- Si se detecta la presencia de 10 vehículos o más sobre la avenida principal, deberá ser servida inmediatamente, a pesar de que la avenida lateral esté en verde y no hayan pasado los 60 segundos .

BIBLIOGRAFIA

- INTRODUCTION TO MICROPROCESSOR SYSTEM DESIGN
  HARRY GARLAND
  MC. GRAW-HILL

- Z80 USERS MANUAL
  JOSEPH J. CARR
  RESTON

- MICROPROCESADORES Y MICROCOMPUTADORES
  SERIE MUNDO ELECTRONICO
  MARCOMBO

- TEMAS DE MICROCOMPUTACION
  JUAN B. MARTINEZ
  FACULTAD DE INGENIERIA U.N.A.M.

- Z80 STARTER SYSTEM

- Z80 PIO PRODUCT SPECIFICATION
  ZILOG

- Z80 CPU PRODUCT SPECIFICATION
  ZILOG

- Z80 CTC PRODUCT SPECIFICATION
  ZILOG

- MK-3080 TECHNICAL MANUAL
  MOSTEK

- MK-3881 TECHNICAL MANUAL
  MOSTEK

- MK-3882 TECHNICAL MANUAL
  MOSTEK

- CROMEMCO Z80 MACRO ASSEMBLER
  CROMEMCO INC.