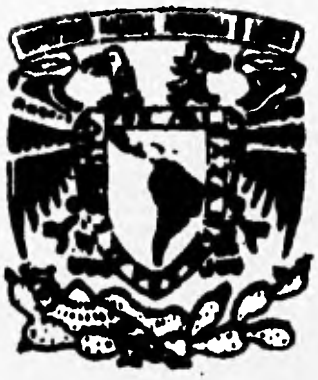
UNIVERSIDAD NACIONAL AUTONOMA DE MEXICO

CAMPUS "ARAGON"



DISEÑO Y CONSTRUCCION DE UN SISTEMA DE DESARROLLO PARA EL LABORATORIO DE MICROPROCESADORES

TESIS PROFESIONAL

QUE PARA OBTENER EL TITULO DE

INGENIERO MECANICO ELECTRICISTA

P R E S E N T A

JORGE TORRES ZAVALA

SAN JUAN DE ARAGON, EDO. DE MEX. 1996

UNIVERSIDAD NACIONAL AUTONOMA DE MEXICO CAMPUS "ARAGON"

UNIVERSIDAD NACIONAL AVENOMA DE MEXICO

JEFATURA DE CARRERA DE INGENIERIA MECANICA ELECTRICA

OF. No. JCIME/ 314/95 .

LIC. ALBERTO IBARRA ROSAS
JEFE DE LA UNIDAD ACADEMICA
P R E S E N T E

Por este medio me permito relacionar los nombres de los Profesores que sugiero integren el Sinodo del Examen Profesional del alumno (a): **JORGE TORRES ZAVALA**, con el tema de tesis : **"DISEÑO Y CONSTRUCCION DE UN SISTEMA DE DESARROLLO PARA EL LABORATORIO DE MICROPROCESADORES"**.

PRESIDENTE:	**ING. RAUL BARRON VERA**	**OCTUBRE, 78**
VOCAL:	**ING. BENITO ZUÑIGA VILLEGAS**	**OCTUBRE, 81**
SECRETARIO:	**DAVID GONZALEZ MAXINES**	**FEBRERO, 85**
SUPLENTE:	**ING. FORTUNATO CERECEDO HERNANDEZ**	**JULIO, 85**
SUPLENTE:	**ING. NARCISO ACEVEDO HERNANDEZ**	**ENERO, 86**

Quiero subrayar que el Director de la Tesis es el **DAVID GONZALEZ MAXINES**, el cual es incluido en base a lo que reza el Reglamento de Exámenes Profesionales de esta Escuela.

A T E N T A M E N T E
"POR MI RAZA HABLARA EL ESPIRITU"
San Juan de Aragón, Edo. de México, 18 de octubre de 1995

EL JEFE DE LA CARRERA

ING. RAUL BARRON VERA

c c.p. Ing. Manuel Martínez Ortiz.- Jefe del Depto. de Servicios Escolares
Ing. Miguel Angel Maldonado Muñoz.- Secretario Técnico de IME
Ing. DAVID GONZALEZ MAXINES.- Asesor de Tesis
ALUMNO

A MIS PADRES

QUE CON SU AMOR Y DEDICACION
NOS DIERON A SUS HIJOS LAS OPORTUNIDADES
QUE EN SU MOMENTO HUBIERON DESEADO.

A MI ESPOSA

CON QUIEN COMPARTO LOS MOMENTOS
MAS DIFICILES Y PLACENTEROS DE MI VIDA

A MIS HIJOS

A QUIENES DEDICO TODO MI ESFUERZO
Y LOGROS OBTENIDOS, MOTIVANDOLOS A SU
SUPERACION CONTINUA Y PERMANENTE.

Quiero agradecer a todos aquellos compañeros que con sus aportaciones, comentarios y sugerencias hicieron posible este trabajo.
Mension especial merecen mis amigos y compañeros JUAN MORA, GERARDO OROZCO, ALBERTO MENDOZA Y GUILLERMO DOMINGUEZ; de quienes recibí su apoyo total e incondicional en los momentos mas criticos y oportunos durante la elaboracion del presente proyecto.

Guadalajara Jal. 24 de noviembre de 1995

UNIVERSIDAD NACIONAL AUTONOMA DE MEXICO
ESCUELA NACIONAL DE ESTUDIOS PROFESIONALES "ARAGON"

DISEÑO Y CONSTRUCCION DE UN SISTEMA DE DESARROLLO PARA EL LABORATORIO DE MICROPROCESADORES

AUTOR: JORGE TORRES ZAVALA No. CTA. 8059981-2

CAPITULADO

CAPITULO V

APENDICES

V.1 Apéndice A. Listado completo del programa MONITOR.

V.2 Apéndice B. Listado completo del CODIGO DE INSTRUCCIONES.

V.3 OTROS APENDICES.(Manual del Usuario,etc)

CONCLUSIONES.

BIBLIOGRAFIA.

CAPITULO I

DESCRIPCIÓN DEL SISTEMA BÁSICO DE DESARROLLO

I.1 Conceptos fundamentales

Debido a su tamaño pequeño y bajo costo, los microprocesadores han revolucionado la tecnología de diseño de los sistemas de computadores digitales, dándole al diseñador la capacidad para crear diseños que anteriormente eran antieconómicos. La aplicaciones de los microprocesadores se pueden ver en diferentes campos, tales como una CPU (Central Processor Unit, Unidad Central de Proceso) en un computador personal, controladores de tráfico, instrumentación, centrales telefónicas,etc.

Una CPU puede diseñarse y construirse combinando varias funciones digitales, tales como registros, ALU's (Arithmetic Logic Unit, Unidad aritmética y Lógica, multiplexores, decodificadores,etc. Un sistema construido para fines específicos tiene implícito el uso de los elementos anteriores. Sin embargo, debido a que la CPU es ampliamente utilizada como componente de un computador digital, también se encuentra disponible (al igual que multiplexores, decodificadores, etc.) en pastillas de circuitos integrados estándar.

A manera de definición muy simple podemos decir que un microprocesador es un sistema que explora secuencialmente una información almacenada, llamada programa, la interpreta y la ejecuta.

La palabra micro se utiliza para indicar el tamaño pequeño físico de los elementos que están involucrados. Un microprocesador combinado con memoria y capacidades de entrada-salida se vuelve un microcomputador. La segunda parte de la palabra microcomputador es lo que marca la diferencia de concepto. La palabra procesador es una abreviación para la unidad procesadora central (CPU). El término microcomputador es utilizado para indicar un sistema de desarrollo completo. Consta de tres unidades básicas: CPU, MEMORIA y ENTRADA-SALIDA.

La ALU (Arithmetic Logic Unit, Unidad Aritmética y Lógica) es un circuito combinacional que realiza diferentes opciones con uno o dos operandos (suma, resta, operaciones lógicas OR, AND, OR-EX, corrimientos). Asociados a la ALU se encuentra un Registo Codificador de Instrucciones, que corresponde a varios flip-flops cuyas condiciones de salida las determina el resultado de la última operación realizada por la ALU, e indica si hay o no ACARREO, SOBREFLUJO, MEDIO ACARREO, si el resultado es CERO o NEGATIVO. También está interrelacionada con uno o varios ACUMULADORES que básicamente son registros que pueden recibir y mantener el resultado de una operación Aritmética y Lógica de un contenido previo y algún otro número.

El PC (Program Counter, Contador de Programa). Es un registro que contiene la dirección de la siguiente instrucción a ejecutar, para ello se incrementa adecuadamente al realizar cada instrucción. En una instrucción de SALTO INCONDICIONAL, el contenido del Contador de Programa se reemplaza por una dirección especificada en la propia instrucción. En un SALTO CONDICIONAL dicho reemplazo ocurre si una condición predeterminada tiene lugar.

Registro de Instrucciones. Recibe cada instrucción y la almacena durante su ejecución.

El Decodificador de Instrucciones es un circuito combinacional cuyas salidas determinan el curso de ejecución de la instrucción leída.

La Unidad de Control. Consiste en un grupo de flip-flops y registros que regulan la operación de todo el sistema, haciendo que los difenentes eventos ocurran en la secuencia adecuada al ejecutar la instrucción correspondiente. Las señales que recibe son:

a) Señal de reloj.
b) Salidas del decodificador de isntrucciones.
c) Señales de restablecimiento, libre y solicitudes de interrupción o mantenimiento de dispositivos externos.
d) Señales internas de control que disparan a otras unidades internas y establecen trayectorias de datos y transferencias de datos dentro del microprocesador.
e) Señales de reloj para sincronización.
f) Señales de habilitación para solicitudes de interrupción y otras señales al estado del microprocesador y los buses.

La Memoria. Esta unidad almacena información (programas y datos) que después suministra a los otros subsistemas, al serle demandada una información especifica por parte de la CPU. Usualmente se organiza en 2n diferentes grupos de celdas individuales, donde cada grupo corresponde an bits (n palabras de m bits), existiendo un dispositivo de acceso que nos permite leer la información en una palabra especifica o bién modificarla. Para ello cada palabra de memoria lleva asociada una dirección que la distingue de las demàs.

Bus. Es un camino por medio del cual se transfiere la información digital, desde una o varias fuentes a cualquiera de varios destinos. En un tiempo determinado solamente puede tener lugar una de estas transferencias de información. mientras que se está produciendo una de estas transferencias de información, todas las demás fuentes que estén unidas a este bus deben quedar bloqueadas.

Bus de Datos Bidireccionales. Es un bus de datos en el cual la información digital puede ser transferida en cualquier dirección. En el presente sistema de desarrollo, es el camino de datos bidireccional mediante el cual se transfieren los datos entre la CPU, MEMORIA y otros dispositivos externos.

Bus de Direcciones. Es un bus inidireccional mediante el cual la información digital sirve para identificar o bien una determinada posición de memoria o un dispositivo determinado de I/O. El bus de direcciones del Z-80 comprende 16 lineas paralelas.

Dirección. Es un grupo de bits que identifican una posición de memoria determinada o un dispositivo concreto de I/O.

Control. Son aquellas partes del computador que tratan a las instrucciones en su secuencia propia, interpretan las instrucciones, y generan las señales adecuadas de sincronización.

Bus de Control. Es un conjunto de lineas que proporcionan las señales que regulan el funcionamiento de un sitema de microcomputador, incluyendo la memoria y los dispositivos externos. Estas señales pueden provenir de la CPU o de un dispositivo externo. El bus de control del microprocesador Z-80 está formado por 13 bits y comprende unas señales que sincronizan las operaciones de I/O entre la CPU y la memoria y otros dispositivos externos; otras señales tales como las interrupciones, espera y restablecimiento y otras más que controlan el acceso a los buses de datos y de direcciones.

I/O. Son las siglas de entrada/salida (Input/Output). Una lectora de tarjetas, unidad de cinta mgnética, impresora, o un dispositivo similar que transmite datos o los recibe desde un computador o un dispositivo secundario de almacenamiento. En un sentido más amplio, cualquier dispositivo digital incluyendo un solo circuito integrado digital, que transmite datos o los recibe, o dirige los impulsos desde un computador.

A los dispositivos que se conectan a los puertos se le conocen generalmente como periféricos. Estos periféricos incluyen una gran variedad de elementos electrónicos y electromecánicos, cuya complejidad va desde unos simples interruptores y lámparas indicadoras, hasta complicados sistemas de adquisición de datos, pantallas de video, etc.

La función de cualquier sistema basado en un microprocesador se implanta por medio de transferencias de datos entre los registros del mismo, la memoria y los dispositivos de entrada/salida, y de transformaciones de datos que ocurren principalmente en los registros internos del microprocesador.

I.2 Estructura del sistema.

Existen dos criterios principales para la clasificación de los microprocesadores, uno se basa en la longitud de palabra y el otro en la tecnología de fabricación.

La longitud de palabra se refiere al número de bits que puede procesar simultáneamente un microprocesador y está determinada por su arquitectura, es decir, por el tamaño de los registros de la ALU y de los buses internos.

La longitud de palabra de los microprocesadores ha ido creciendo a través de los años, desde los 4 bits del primer microprocesador hasta los 32 bits de los microprocesadores más recientes.

Actualmente los microprocesadores de 4 bits se consideran obsoletos y los de 32 se reservan para aplicaciones muy complejas. En la generalidad de los casos se utilizan microprocesadores de 8 bits y de 16 bits, los primeros son más usuales por haber sido de más temprana su aparición y porque tienen disponible un amplio soporte de programación y circuitería, factores que determinaron la elección del microprocesador Z-80 para el desarrollo del presente proyecto.

Con respecto a las tecnologías de fabricación, los primeros microprocesadores se implantaron con tecnología PMOS; sin embargo, actualmente la tecnología de fabricación de microprocesadores más difundida es la NMOS. Ultimamente se ha desarrollado bastante la tecnología CMOS para dispositivos de bajo consumo de energía.

I.2.1 Los Registros internos.

La CPU Z-80 contiene 208 bits de memoria de lectura/escritura dividida en 18 registros de 8 bits y 4 registros de 16 bits, como se muestra en la Fig. I.2.1

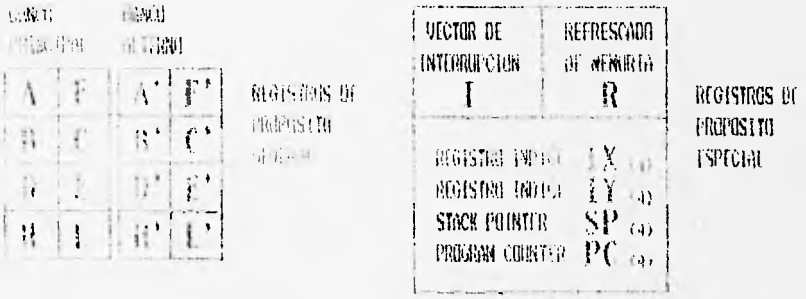


Fig. I.2.1 REGISTROS INTERNOS DEL MICROPROCESADOR Z-80.

Los registros incluyen dos conjuntos o "bancos" de 6 registros de propósito general que pueden usarse individualemente como registros de 8 bits o en pares como registros de 16 bits. El banco principal consiste en los registros BC, DE, y HL, en tanto que el banco alterno comprende los registros B'C', D'E' y H'L' (Fig.I.2.1). También hay 2 Acumuladores (A y A') y dos registros de banderas (F y F'). Sin embargo, solo es posible trabajar a un tiempo con uno de los bancos de registros.

En los registros de uso especifico se encuentran el Contador del Programa (PC, Program Counter), el Apuntador de Pila (SP, Stack Pointer), los Registros Indices (IX, IY, Index Registers), el Registro de Refrescado (R, Refresh Register) y el Registro de Vector de Interrupciones (I, Interrupt Register) Cada uno de ellos tiene funciones especificas que se describirán en el desarrollo del presente proyecto, conforme se avance en el mismo.

El Contador del Programa es un registro que guarda los 16 bits de la dirección de la instrucción que en ese momento ésta obtiene de la memoria. Siendo el PC un registro de 16 bits, es capáz de direccionar hasta 2 = 65,536 localidades de memoria. El PC se incrementa automáticamente después que su contenido ha sido transferido a las líneas de dirección. Cuando ocurre un salto en el progreama, el nuevo valor se coloca en el PC sustituyendo al valor resultante del incremento normal.

Después que se ha obtenido una instrucción de la memoria, la CPU la almacena en un registro conocido como REGISTRO DE INSTRUCCIONES (IR,Instruction Register). La instrucción almacenada en el IR es decodificada y usada para activar una de varias líneas. Cada línea representa un conjunto de actividades asociadas con la ejecución de una instrucción particular. El dispositivo que traduce la instrucción en acciones concretas es el DECODIFICADOR DE INSTRUCCIONES (ID, Instruction Decoder).

1.2.2 La Unidad Aritmético/Lógica (ALU, Arithmetic Logic Unit)

La Unidad Aritmético/Lógica es de 8 bits y ejecuta las instrucciones Aritméticas y Lógicas de la CPU. Internamente, la ALU se comunica con los registros y el bus externo de datos a través del bus interno de datos. El tipo de funciones ejecutadas por la ALU son:

Suma	Rotaciones o desplazamientos hacia la derecha o izquierda (Aritméticos o Lógicos).
Resta	Incrementar.
AND	Decrementar.
OR	Poner un bit en uno (set).
X-OR	Poner un bit en cero (reset).
Comparación	Probar el estado de un bit (test).

Además la ALU contiene un conjunto de flip-flops llamados BANDERAS (Flags), las cuales guardan información relacionada con el resultado de una operación aritmética o lógica. Por ejemplo una de las banderas sirve para indicar si el resultado de la última operación fué cero.

Existe un registro intimamente relacionado con la ALU, denominado ACUMULADOR Generalmente el Acululador contiene uno de los operandos que serán manipulados por la ALU y, también muy frecuentemente, el resultado de la operación se deposita en ese registro, reemplazando a uno de los operandos originales.

1.2.3 Los Buses de Interconexión

Un microprocesador se comunica con otros circuitos del sistema a través de grupos de lineas o buses de interconexión. Con base en el tipo de información que conducen, las lineas de interconexión se pueden agrupar en tres buses: Datos, Direciones y Control.

EL BUS DE DATOS es un conjunto de lineas bidireccionales, que transportan información del microprocesador hacia la memoria o puertos y, de éstos hacia el microprocesador.

EL BUS DE DIRECCIONES es unidireccional, es decir, por él solamente circula información proveniente del microprocesador. Comprende a las lineas que transmiten una dirección generada por el CPU, la cual selecciona a un puerto o a una localidad de memoria. Esta dirección especifica el origen o destino de la información que transmitirá por el bus de datos.

La sincronización y el sentido de la transferencia de información en el bus de datos (hacia adentro o hacia afuera del microprocesador) y el tipo de transferencia se indican por medio de señales de control originadas en el CPU. Todas ellas conforman el llamado BUS DE CONTROL. cada una de las señales en el bus de control es unidireccional. Algunas de ellas son salidas del microprocesador, mientras que otras son entradas a él.

Además de los buses de interconexión externos existen otros dentro del microprocesador que sirven para comunicar entre si a la ALU, los registros internos y la unidad de control. A éstos se le denomina BUSES INTERNOS.

A continuación se describen las unidades funcionales que integran éste proyecto.

I.2.4 Memorias de Datos y de Programas.

La MEMORIA DE DATOS es una memoria de lectura/escritura (RAM), cuyo objetivo es permitir el almacenamiento temporal de datos o programas de aplicación. Por sus características de lectura/escritura, la información que reside en ella se puede alterar fácilmente; sin embargo, ésta se pierde cuando se apaga la fuente de alimentación.

La memoria de datos se puede considerar como una extensión de los registros internos de propósito general, ya que cada una de sus localidades es precisamente un registro. La diferencia está en que los registros de la memoria son externos al microprocesador, y por lo tanto la velocidad de la transferencia de información es lenta.

La MEMORIA DEL PROGRAMA es una memoria de lectura solamente, Como por ejemplo una ROM o una EPROM que contiene el programa, es decir, la secuencia de instrucciones que va a determinar las tareas que realice el microprocesador. En algunos casos la memoria del programa también almacena parámetros o tablas de datos que no sufren modificaciones. En este proyecto utilizaremos las EEPROM durante la etapa de desarrollo del prototipo del sistema, para asi poder depurar y alterar el programa. Las ROM se prefieren cuando el sistema ya se encuentra en producción y se ha llegado ya a la versión final del programa.

I.2.5 Los Puertos de Entrada/Salida

Los Puertos de Entrada/Salida son circuitos que se encargan de establecer el contacto del microprocesador (CPU) con el mundo exterior.

Los puertos se conectan a dispositivos periféricos que generan información para ser procesada por la CPU (dispositivos de entrada) o que aceptan datos provenientes del microprocesador y los transforman en información significativa para el mundo exterior (dispositivos de salida).

Estos periféricos incluyen una gran variedad de elemetos electrónicos y electromecánicos, cuya complejidad va desde unos simples interruptores y lámparas indicadoras, hasta complicados sistemas de adquisición de datos, pantallas de video, etc. En este proyecto, por su naturaleza didáctica, utilizaremos un teclado hexadecimal como dispositivo de entrada y de visualizadores de siete segmentos (displays) asi como diodos emisores de luz (leds) como dispositivos de salida.

I.3 Conceptos de Interrupción y DMA.

I.3.1 Conceptos básicos de Interrupciones.

Una INTERRUPCION (interrupt) se puede definir como una señal proveniente de un dispositivo externo, que llega a una entrada del microprocedasor dedicada a este propósito y que le indica al microprocesador que el dispositivo que la originó está solicitando servicio.

Cuando ucurre una interupción, el microprocesador suspende temporalmente la ejecución del programa principal y transfiere el control a una subrutina especialemte diseñada para atender al dispositivo que provocó la interrupción, a esta subrutina se le denomina SUBRUTINA DE SERVICIO DE LA INTERRUPCION (Interupt Service Routine) o MANEJADOR DE INTERRUPCION (Interrupt Handler). Al terminar el servicio, el microprocesador regresa al programa principal, continuando con sus actividades normales. Desde este punto de vista, una interrupción es esencialmente una llamada a subrutina iniciada por un circuito externo (hardware) y con un retorno controlado por programa (software). Ver la fig. I.3.1

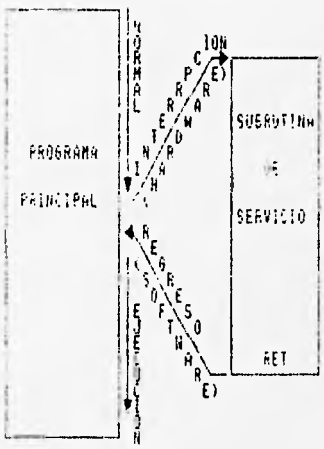


Fig. I.3.1 TRANFERENCIA DE CONTROL EN RESPUESTA A UNA INTERRUPCION

Además, las interrupciones tienen la caracteristica de que son eventos asincronos, es decir, pueden ocurrir en cualquier momento durante la ejecución del programa principal sin posibilidad de hacer una predicción exacta. Por ello hay que hacer consideraciones especiales y tener cuidados adicionales al manejarlas.

I.3.2 Uso de la Interrupciones.

Las interrupciones se utilizan:

a) En la detección de eventos catastróficos para el sistema, tales como fallas súbitas en las fuentes de alimentación. Por medio de una interrupción el microprocesador puede ser alertado a tiempo de preservar la información vital y desconectar los dispositivos que pueden dañarse.

b) En dispositivos donde el microprocesador no necesita obtener información con mucha frecuencia. Por ejemplo, en sistemas de alarma, ya que el microprocesador no sabe cuando va a aparecer una situación que requiera sus servicios. El dispositivo genera una interrupción cuando llega el momento, y mientras tanto el microprocesador puede ejecutar otras tareas.

c) En la atención a dispositivos periféricos que son mucho más lentos que el microprocesador: teclados pantallas, impresoras. En vez de que el microprocesador los espere, es más eficiente que los dispositivos lo interrumpan cuando tengan algo nuevo que enviar o estén listos para recibir.

d) En el servicio de periféricos de alta velocidad (discos) o dispositivos para los cuales sea muy difícil mantener la información hasta el momento en que el microprocesador lo solicite.

I.3.3 Ventajas y desventajas de las Interrupciones.

Ventajas:

1. Liberan al microprocesador de la pérdida de tiempo causada por esperar a que ocurra cierto evento.

2. Permiten la ejecución de un programa principal junto con el control simultáneo de varios dispositivos externos.

3. Proporcionan servicio prioritario a dispositivos criticos dentro del sistema.

4. Facilitan la deteción de eventos de "tiempo real".

Desventajas:

1. Pueden ser necesarios circuitos externos adicionales para la generación de la señal de interrupción y la identificación del dispositivo.

2. Dada su naturaleza asincrona, es muy difícil probar su funcionamiento, y todavia más, encontrar posibles fallas.

3. Pueden requerir instrucciones adicionales además de las estrictamente necesarias para dar servicio al dispositivo. Por ejemplo, para preservar registros o identificar al dispositivo.

I.3.4 Tipos de interrupciones.

Un microprocesador puede tener entradas para responder a dos tipos de interrupciones: inhibilbes (maskable) y no inhibibles (non maskable).

Cuando se activa una entrada de INTERRUPCION NO INHIBIBLE, el microprocesador siempre es interumpido, es decir, la señal de interrupción es aceptada bajo cualquier condición. Esto hace que las interrupciones no inhibibles sean las más apropladas para manejar eventos catastróficos tales como la pérdida de energia.

Por otro lado, cuando se activa una entrada de INTERRUPCION INHIBIBLE, el microprocesador reconoce la interrupción solamente si esa entrada se encuentra habilitada. Las entradas de interrupción se habilitan o inhabilitan bajo el control del programa. Si la entrada está inhabilitada, el microprocesador ignora la interrupción. En la fig. I.3.4.1 muestra la configuración interna de las entradas de interrupción inhibibles y no inhibible en un microprocesador.

Para muchos dispositivos que obtienen servicios basándose en las interrupciones, las interrupciones no enmascarables son utilizadas de forma que el software de servicio de las mismas puede controlar la sensibilidad de la CPU a las interrupciones. La primera tarea de una rutina de servicio de interrupciones es la de guardar el estado de la CPU, de forma que la tarea interrumpida pueda continuarse en donde se quedó, cuando se ha completado el servicio de la tarea interrumpida.

Una entrada de interrupción no inhibible se puede inhabilitar externamente por medio de una señal proveniente de un puerto de salida. En la figura I.3.4.2 se muestra como el bit 0 de un puerto de salida controla la señal de interrupción. Si una instrucción de salida envia un uno lógico por ese bit, la entrada de interrupción se habilita; si envia un cero lógico, se inhabilita.

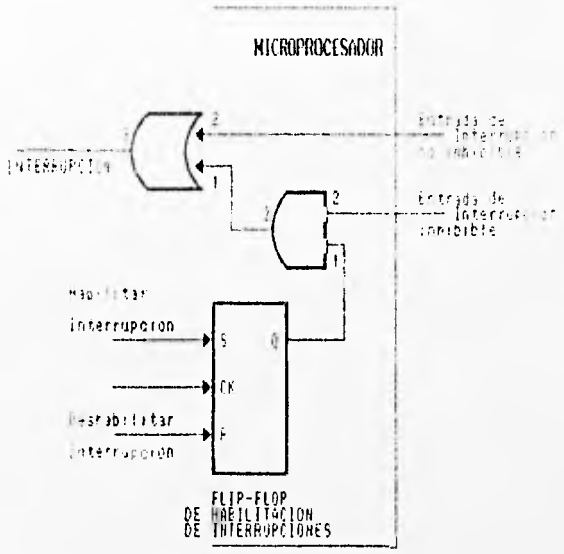


Fig. I.3.4.1 Configuracion de las entradas de interrupcion inhibible y no inhibible en un microprocesador.

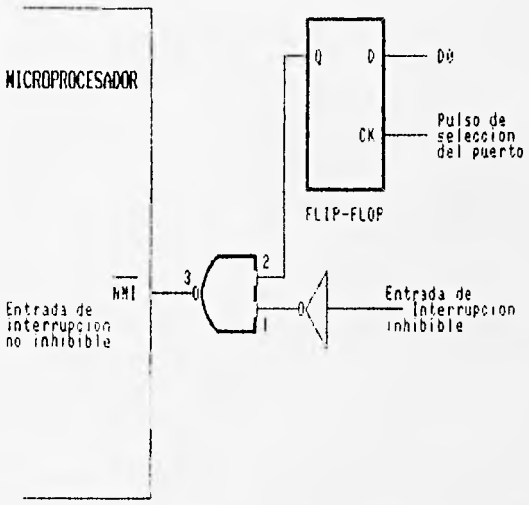


Fig. I.3.4.2 Convirtiendo una entrada de interrupcion no inhibible en una entrada de interrupcion inhibible.

I.3.3 Repuesta a una interrupción.

En respuesta a una interrupción el microprocesador realiza las siguientes operaciones:

1. Se completa la instrucción que esta en proceso.

2. Se ejecuta un ciclo de máquina especial, durante el cual se almacena el contenido del contador del programa y se transfiere el control a una dirección apropiada.

3. Se inicia la ejecución de la subrutina de servicio a la interrupción, preservando el estado de la CPU si esto es necesario.

4. Si son varios los dispositivos que pudieron haber causado la interrupción, se identifica aquel que haya solicitado la interrupción primero y tenga mayor prioridad.

5. Se ejecuta la parte de la subrutina de servicio que compete directamente a la atención del dispositivo que interrumpió.

6. Se restaura el estado original del microprocesador.

7. Se devuelve el control a la instrucción siguiente, a aquélla donde ocurrió la interrupción.

La fig.I.3.4.3 muestra la secuencia descrita arriba. Cada etapa requiere de una cierta cantidad de tiempo. La combinación de los tiempos para un microprocesador dado, junto con la circuiteria externa de interrupción , determinan la velocidad de repuesta del microprocesador a una solicitud de servicio generada por un dispositivo de E/S.

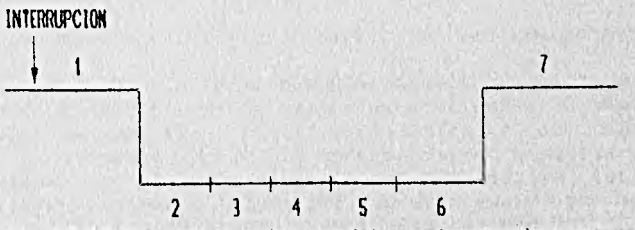


Fig. I.3.4.3 Secuencia de accione asociadas con la respuesta a una interrupcion.

El tiempo que trancurre entre la ocurrencia de la interrupción y el inicio de la ejecución de la subrutina de servicios se conoce como el TIEMPO DE RESPUESTA y es la suma de los tiempos de las etapas 1 a 4. A diferencia entre el tiempo total que el microprocesador dura interrumpido y al tiempo real de ejecución de la subrutina de servicio se les llama sobrecarga (overload).

El número de circuitos adicionales requeridos para producir la respuesta a una interrupción varía mucho. Algunos microprocesadores construyen internamente la instrucción que transfiere el control a la subrutina de servicio; otros necesitan circuitos externos para generar la instrucción de transferencia de control o para formar la dirección de la subrutina de servicio.

El número de entradas de interrupción que tenga el circuito integrado de un microprocesador determina el número de respuestas que puede producir sin hardware o sofware adicionales. La CPU solamente puede generar una instrucción o direccion diferente por cada entrada de interrupción.

Si el número de dispositivos que pueden provocar una interrupción excede al número de entradas de interrupción, el microprocesdador necesitará hardware o software extra para identificar el causante de la interrupción. En el caso más simple, el software puede ser una subrutina de encuesta o muestreo (polling) que revise es estado de los dispositivos que pudieran haber interrumpido. La solución alternativa es hardware adicional que proporcione una clave especifica o "vector" para cada dispositivo. Las dos alternativas se pueden mezclar; los vectores pueden identificar grupos de dispositivos y en ellos se puede detectar un dispositivo particular por medio de muestreo (polling).

Todo lo anteriormente expuesto fué a manera de introducción a lo que son las interrupciones en cualquier microprocesador. Sin embargo, el sistema de interrupciones correspondiente al Z-80 se analizarán en detalle en los capitulos siguientes.

I.4 Accseso Directo a Memoria (Direct Memory Access,DMA)

El ACCESO DIRECTO A MEMORIA (Direct Memory Access) o DMA consiste en establecer un camino directo entre la memoria y los dispositivos de E/S, sustituyendo software por hardware. Es decir, a diferencia de la E/S controlada por programa y de la E/S controlada por interrupciones, en las cuales los datos se transfieren pasando a través del microprocesador por medio de una secuencia de instrucciones, en la E/S por DMA la transferencia entre un puerto de E/S y la memoria se realiza directamente, sin pasar por el microprocesador. Para esto, la subrutina de servicio se reemplaza por un circuito especializado denominado cntrolador de DMA.

El CONTROLADOR DE DMA es un dispositivo que puede enviar comandos a la memoria, que se comportan de la misma manera que los comandos generados por el microprocesador. En este sentido el controlador es un segundo procesador en el sistema, pero que está dedicado exclusivamente a funciones de E/S. Como lo muestra la fig.I.4.1, el controlador conecta uno o más puertos de E/S directamente a la memoria.

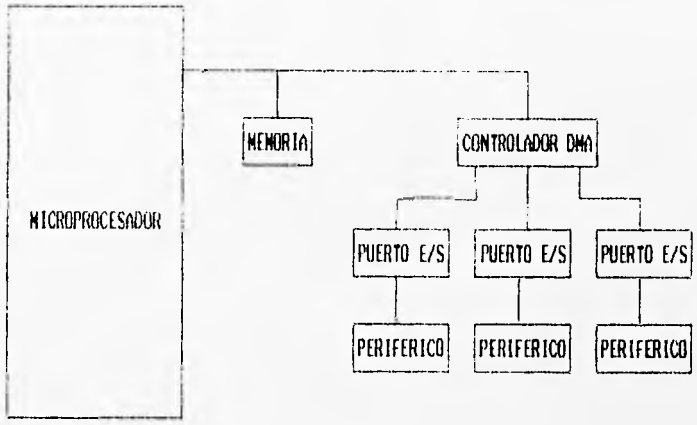


Fig. I.4.1 Sistema con DMA

Para poder llevar a cabo sus funciones, un controlador DMA debe ser capaz de proporcionar las direcciones y señales de control a la memoria y de transferir los datos; contiene su propio registro de direcciones, su registro contador de palabras y la circuiteria de control necesaria para leer o escribir en la memoria. La fig. I.4.2 muestra los componentes básicos de un controlador de DMA. En realidad, un controlador de DMA es un dispositivo mucho más complejo que eso; anteriormente para implantarlo se requerian mas de 50 circuitos, que en la actualidad han sido reducidos a un solo circuito LSI.

Además, el microprocesador debe tener ciertas caracteristicas para permitirle al controlador de DMA el uso y control de los buses de direcciones, datos y control del sistema. Esta es la razón por la cual las lineas de estos buses son de tercer estado en un gran número de microprocesadores, entre ellos el Z-80.

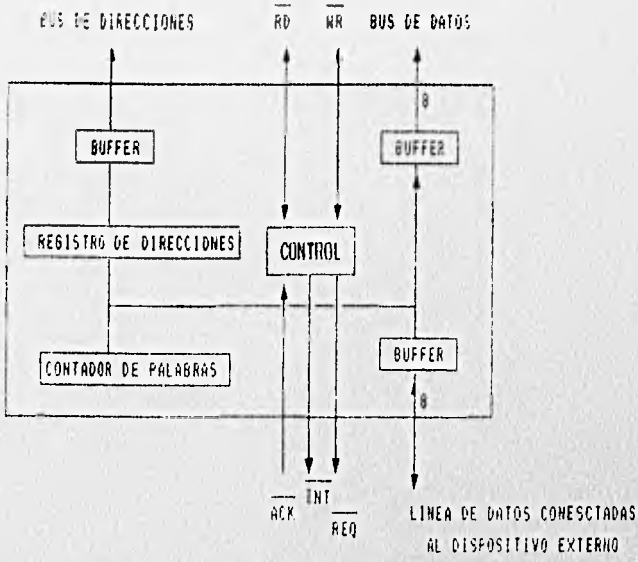


Fig. I.4.2 Estructura basica de un controlador de DMA.

El acceso directo a memoria se usa comúnmente en tres tipos de transferencia de datos: bloque continuo, robo de ciclos y transparente.

El método de DMA EN BLOQUE CONTINUO transfiere un bloque de datos a la mayor velocidad posible. Los pasos en la ejecución de una transferencia de datos usando DMA en el bloque continuo, son los siguientes:

1. El microprocesador transmite al controlador de DMA la dirección inicial de memoria para la transferencia y el número de palabras a ser transferidas.

2. El microprocesador regresa a efectuar otras actividades.

3. Cuando el puerto asociado al periférico de entrada tiene un dato listo para ser transmitido a la memoria o cuando el puerto asociado al periférico de salida está listo para recibir un dato de la memoria, entonces se lo indican al controlador de DMA, el cual a su vez genera una solicitud de DMA para el microprocesador.

4. El microprocesador responde a la solicitud de DMA, pone a "flotar" sus lineas de direcciones, datos y control y suspende cualquier procesamiento que requiera el uso de los buses.

5. El controlador de DMA proporciona una dirección y las señales de control para leer o escribir en la localidad de memoria apuntada por esa dirección. El puerto de E/S envia o acepta el dato a través del bus de datos. Después de que se ha transferido un dato, el controlador de DMA incrementa su registro apuntador de direcciones y decrementa su registro contador de palabras. Si no se ha terminado de transferir el número de datos requerido, el contador de DMA repite este paso cuando el puerto de E/S esté listo de nuevo.

6. Cunado se ha transferido el número de palabras especificado, el controlador retira su solicitud de DMA e interrumpe al microprocesador para indicarle que el proceso de DMA se ha completado.

La máxima velocidad de transferencia para DMA en bloque continuo está limitada únicamente por la duración de un ciclo de lectura o escritura de la memoria y por la velocidad del controlador de DMA. Abajo de este valor máximo, la velocidad de transferencia está limitada por la velocidad a la que el dispositivo periférico puede transmitir o recibir información. Por otro lado, los pasos anteriores muestran que el controlador de DMA se comporta como un puerto normal antes y después de la transferencia de un bloque.

En el método de DMA POR ROBO DE CICLOS, los datos se transfieren concurrentemente con otras tareas de procesamiento llevadas a cabo por el microprocesador. La ejecución de éste método es similar a la de DMA en el bloque continuo, la diferencia está en que los pasos 2, 3, 4 y 5 se repiten por cada palabra hasta que todas han sido transferidas. Así, lo que hace el controlador de DMA es robarle ciclos de reloj al microprocesador, durante los cuales reliza la operación anterior Evidentemente, la velocidad del proceso que se está realizando simultáneamente al DMA disminuye proporcionalmente al tiempo robado para la transferencia.

En algunos microprocesadores se pueden añadir circuitos externos, con el propósito de que el robo de ciclos ocurra durante el procesamiento interno, cuando no se están utilizando los buses del sistema. A este método se le denomina DMA TRANSPARENTE en el sentido de que no interfiere o disminuye la velocidad de operación y ejecución normal de instrucciones y, por lo tanto, las transferencias son "transparentes" al microprocesador. Aunque este método es el más eficiente, es también el más complicado de implantar, ya que requiere dispositivos para detectar los estados del microprocesador que implican únicamente actividades internas.

En los tre tipos de DMA, la única programación que se requiere es la secuencia de instrucciones necesarias para cargar los valores iniciales en el registro apuntador de direcciones y en el registro contador de palabras del controlador.

El microprocesador Z-80 posee dos terminales para la sincronización de un controlador de DMA: BUSRQ Y BUSAK. BUSRQ es una entrada activa en 0 lógico, a través de la cual el controlador le solicita al microprocesador la utilización de los buses. BUSAK es una salida activa en 0 lógico con la que el microprocesador le responde al controlador que su solicitud de DMA ha sido recibida y que los buses se encuentran en estado de alta impedancia, pudiendo ser manejados a partir de ese momento por el controlador de DMA. En la fig. I.4.3 se muestra la interconexión común entre un controlador de DMA y el microprocesador.

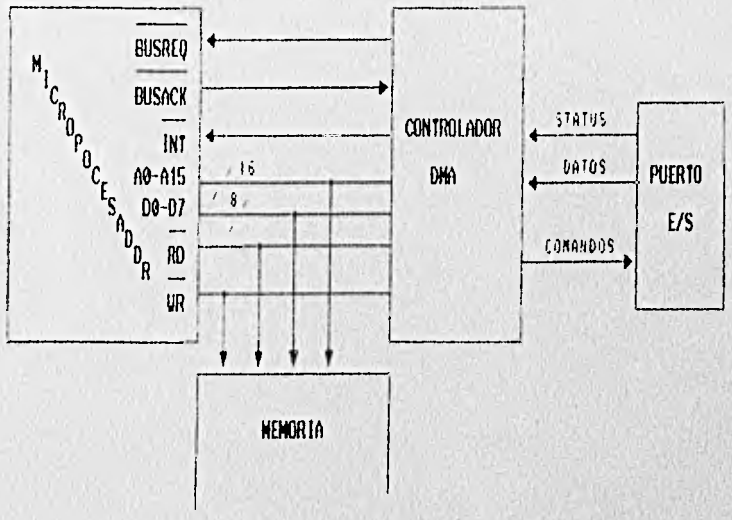


Fig. I.4.3 Interconexion comun entre un controlador de DMA y un microprocesador.

I.5 Memorias.

Una consideración muy importante en cualquier sistema de desarrollo que tenga un microprocesador es la memoria. Tanto las instrucciones del programa como los datos deben estar almacenados en algún tipo de memoria, de donde puedan ser obtenidos en el momento apropiado para que el microprocesador lleve a cabo su función. Aun cuando el Z-80 posee un conjunto de registros internos, éstos no pueden usarse para guardar instrucciones, sino únicamente para el almacenamiento temporal de datos; además, su número es muy reducido. or lo tanto, surge la necesidad de interconectar al microprocesador con circuitos externos de memoria.

La fig.I.5.2.1 muestra un circuito de memoria común y la fig. I.5.2.2 muestra las líneas de interconexión más comunes entre el microprocesador y la memoria.

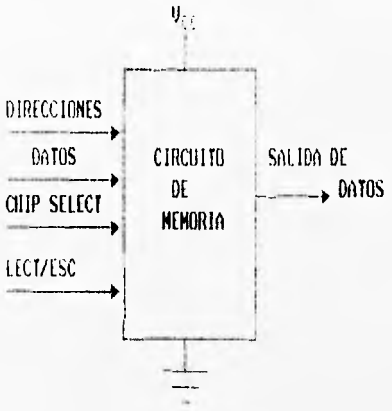


Fig. I.5.2.1 Circuito comun de memoria.

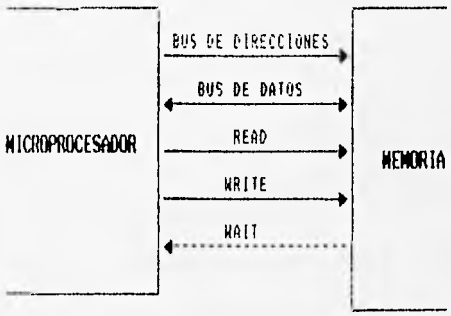


Fig. I.5.2.2 Lineas de interconexion entre un microprocesador y la memoria.

I.5.1 Tipos de memorias.

La memoria externa se puede dividir en dos grandes categorias: memoria de sólo lectura o ROM (Read Only Memory) y memoria de lectura y escritura o RWM (Rear/Write memory), mejor conocida como RAM. La ROM se utiliza para guardar pasos de un programa o datos que no requieran modificación. El contenido de estas localidades de memoria se considera permanente y sólo en casos y con dispositivos especiales puede ser alterado. Por otra parte, la RAM se usa para almacenar información que cambia constantemente mientras el microprocesador está en operación como en el caso de cálculos efectuados por el programa. A su vez, la memoria de lectura y escritura se divide en dos clases: RAM estática, que guarda información mientras no se desconecte la fuente de alimentación, y RAM dinámica, que necesita un refrescado periódico para mantener la información.

Independientemente de sus caracteristicas particulares, la función de cualquier memoria es la misma: proporcionar, cuando el microprocesador lo requiera, ya sea una instrucción para ser ejecutada o una localidad donde se pueda almacenar un dato.

I.5.2 Caracteristicas de la interconexión con Memoria.

Los microprocesadores se comunican con la memoria externa a través de sus buses de direcciones, datos y control. El acceso a una localidad de memoria se hace por medio de una dirección. Para cada operación con memoria, el microprocesador especifica en el bus de direcciones la ubicación de la localidad de memoria con la que se va a efectuar la transferencia de información. Además, por medio de las señales del bus de control el microprocesador indica el sentido del flujo de la información en el bus de datos, es decir, si se trata de una operación de lectura (transferencia de un dato de la memoria al microprocesador) o de una opreración de escritura (transferencia de un dato del microprocesador a la memoria). Finalmente, el microprocesador cuenta con medios para distinguir entre una transferencia con la memoria y una transferencia con los puertos de E/S.

Por otra parte, para conectarse al microprocesador, un circuito de memoria tiene:

- Entradas de direcciones.

- Entradas/Salidas de datos.

- Entradas de control para seleccionar el circuito (chip select).

- Entradas de control para inhabilitar (output disable) o habilitar (output enable) los buffers de tercer estado de las salidas.

- Además, en las memorias RAM existe una entrada de control para especificar la lectura o escritura (R/W) de la memoria.

I.6 Algoritmos y Diagramas de Flujo.

Un ALGORITMO es la especificación, paso por paso, de la solución de un problema dado. Este algoritmo puede ser expresado en cualquier lenguaje o simbolismo y debe terminar con un número finito de pasos. En programación, un algoritmo es un método lógico o computacional para producir el resultado deseado. Casi siempre es necesario un paso intermedio entre el diseño del algoritmo y la codificación del programa, es este paso se construye el diagrama de flujo.

Un DIAGRAMA DE FLUJO es la representación gráfica de un algoritmo, indicando el orden en que las operaciones son llevadas a cabo y las decisiones que determinan esta secuencia.

El diagrama de flujo es el equivalente en software de lo que es el diagrama de bloques en hardware. Sus finalidades son:

a) Simplificar la codificación del algoritmo en el lenguaje particular de la computadora o microprocesador.

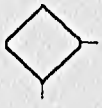
b) Facilitar la comprensión del algoritmo por otras personas.

I.6.1 Reglas de los diagramas de flujo.

Los diagramas de flujo consisten en una secuencia de símbolos geométricos que contienen los pasos del algoritmo. Los símbolos básicos se muestran en la fig. I.5.1.1.

Línea de flujo. Indica la dirección de la secuencia

Entrada/Salida. Señala la recepción y la transmisión de información.



Decisión. Corresponde a una bifurcación condicional en referencia a una relación de magnitudes (igual, diferente, mayor, mayor o igual, menor o igual); o bien la existencia o no de una condición especifica (acarreo, sobreflujo, cero, etc.)

Proceso. Indica las operaciones a realizar.

Terminal. Marca el principio, el fin o un punto de interrupción de la secuencia.

Los simbolos están conectados por lineas sólidas con puntas de flecha que indica el flujo del proceso. Mientras sea posible, el flujo debe ser de arriba hacia abajo y de izquierda a derecha.

Como es muy importante mantener la claridad y el orden en los diagramas de flujo, frecuentemente se utilizan conectores que indican los puntos de rompimiento y reanudación de la trayectoria de flujo, con el fin de evitar el cruce de lineas o permitir la continuación del diagrama en una hoja distinta. Los conectores se representan con circulos con un número o caracter, que sirve para identificar los conectores que enlazan una misma trayectoria de flujo.

Los enunciados que indican las operaciones o decisiones asociadas con cada simbolo aparecen dentro de èl. Estas frases no precisan ninguna formalidad; sin embargo, se recomienda que sean cortas, expresadas con conceptos generales aplicables a cualquier lenguaje de programación y completamente claras con respecto a lo que se quiere que haga la computadora o microprocesador. A fin de lograr lo anterior es preciso que, previamente al diseño del digrama de flujo, se tenga la compresión total del problema a resolver.

Normalemente existirá más de un camino o algoritmo para la solución del problema, cada uno con sus pros y sus contras. Es mucho más sencillo analizar todas las alternativas, con las ventajas y desventajas que presentan, en términos de diagrama de flujo, que con programas ya codificados.

En el proceso de creación de un programa es más conveniente pasar de lo general a lo particular. Primero se desarrolla un diagrama de flujo que represente al algoritmo de una manera global, plasmando a grandes rasgos las etapas o tareas que se tienen que realizar para llegar a una solución final del problema. Este diagrama de flujo se divide progresivamente en pasos cada vez más detallados, hasta que cada simbolo pueda ser convertido en una instrucción o un grupo de ellas.

I.7 Lenguajes de Programación

Cada tipo de microprocesador entiende únicamente su propio lenguaje binario (lenguaje de máquina), expresándose sus instrucciones como grupos de ceros y unos (aunque nosotros los manejemos en hexadecimal por claridad).

Un programa codificado en lenguaje de máquina es muy difícil de entender para el programador y por lo general hace referencia a direcciones absolutas de memoria, hecho que lo hace inflexible ante cambios o modificaciones.

Por esta razón empleamos un seudocódigo simbólico basado en mnemonicos (grupos de letras que facilitan recordar la operación de que se trate), el cual simplifica el trabajo del programador. Cada instrucción está formada por un mnemónico y un operando (que puede ser una direccioón o una constante) y debe ser traducida a una sola instrucción en lenguaje de máaquina.

La traducción la puede hacer manualmente el programador usando la hoja de codificación del microprocesador (se anexará como apéndice al final de este trabajo), o bien de manera autómatica, usando una computadora con un programa denominado ENSAMBLADOR.

Los programas ensambladores ofrecen una serie de facilidades al programador, estando entre las más usuales las siguiente:

1. Traducción autómatica de mnemónicos.

2. Referencia simbólica a las direcciones de memoria, con ello se introduce el concepto de variable, con la gran ventaja de que es más fácil recordar el nombre de una variable que la dirección de memoria en que se encuentra. En este caso, los operandos se proporcionan simplemente suministrando los nombres de las variables. Además, un programa en un ambiente simbólico es mucho más flexible que el mismo programa en un ambiente absoluto, ya que el reubicarlo en otra área de memoria puede hacerlo la computadora de manera automática en el momento en que lo indique el programador.

También se introduce el concepto de etiqueta, que es una referencia simbólica a renglones dentro del programa donde se emplean bifurcaciones condicionales o incondicionales.

3. Representación flexible de los datos de entrada, por ejemplo, pueden proporcionarse en decimal y hacer que la propia computadora los traduzca a binario.

4. Detección autómatica de errores de sintaxis.

5. Listado de programas a petición del programador, permitiendo el libre uso de comentarios y anotaciones marginales.

Los desensambladores toman un programa en binario y lo convierten a seudocódigo, anotando los mnemónicos correspondientes a los códigos de operación. Parte del proceso es arbitraria, por ejemplo la elección de los nombres de las variables y las etiquetas, permitiéndosele al programador reasignar dichos nombres a su elección.

Para la codificación manual de las instrucciones de in programa en lenguaje de máquina, el fabricante del microprocesador proporciona una tarjeta del programador en que aparecen unas tablas con los códigos correspondientes a cada instrucción

En las distintas columnas de ella se indican las posibles operaciones que realiza el microprocesador, los mnemónicos correspondientes, los modos de direccionamiento (proporcionando para cada uno de ellos el código de operación, la cantidad de pulsos de reloj que tarda en efectuarse y el número de bytes que emplea), la representación simbólica de la operación y una Sindicación de cuáles bits del código de condición se ven afectados por cada instrucción.

El programador, una vez que ha establecido el algoritmo y dibujado el diagrama de flujo correspondiente al programa deseado, procede a anotarlo de manera ordenada, isntrucción por instrucción, usando un seudocódigo de máquina basado en mnemónicos. Después de ello emplea la hoja de codificación para traducirlo a lenguaje máquina.

Para cada mnemónico hay que leer de la tarjeta del programador el código de operación para la forma de direccionamiento deseada y anotarlo en el mismo renglón, seguido de un operando (que puede ser una dirección o un valor constante según sea el caso) si se quiere. El número total de bytes de cada instrucción debe ser exactamente igual al que para ella indique la tarjeta del programador.

Las instrucciones en lenguaje de máquina posteriormente se introducen en la memoria RAM del sistema de evaluación, se corre el programa y se verifica su correcto desempeño (caso contrario procede revisarlo y corregirlo).
El programa resultante puede ser grabado en cinta magnética si su uso va a ser ocacional o puede almacenarse permanentemente en un circuito integrado por el programador de EPROMs.

Como conclusión para este capitulo se describe a grandes rasgos el sitema de desarrollo.

A fin de facilitar la labor de llevar a la realidad una idea de aplicación de microprocesadores a una tarea especifica, se han implementado diversos sistemas de desarrollo con diverso grado de complejidad, construidos alrededor de un microprocesador, éste en particular tiene las caracteristicas básicas, con opción a mejorarse o ampliarse. Estas son las siguientes:

1. Teclado para introducir información.

2. Indicadores hexadecimales para observar el contenido de registros y de las diferentes direcciones de memoria.

3. Interruptor de restablecimiento.

4. Un programa monitor almacenado en ROM, que se inicializa después de la señal de restablecimiento y que permite:

a) Leer una secuencia de bytes introducidos mediante el teclado y almacenarlos en direcciones consecutivas de RAM.

b) Leer el contenido de cualquier dirección de memoria o de cualquier registro y mostrarlo a voluntad en los indicadores hexadecimales. También permite cambiar información contenida en registros o memorias RAM.

c) Correr un programa a partir de cualquier dirección inicial.

d) Correr un programa paso a paso, es decir, ejecutando una instrucción en cada ocación para poder observar sus efectos.

e) Introducir pausas o sea puntos de ruptura en la secuencia de instrucciones, deteniendo la ejecución del programa para examinar el contenido de los registros, memorias o puntos de entrada y salida.

f) Utilizar diversas subrutinas de propósito general, por ejemplo, Temporización, Cálculo de desplazamiento relativo, Manejo de indicadores hexadecimales, etc.

5. RAM del sistema que puede utilizarse para memorizar el programa y los datos.

6. Bases para insertar un PROM de usuario, para dar al sistema una aplicación especifica.

7. Interfases para dispositivos de E/S tales como sensores, LEDs, teclado hexadecimal,etc.

8. Un programador de memorias integrado al sistema de desarrollo, opcional.

Cada una de las teclas del sistema se analizarán cuando se haga la etapa de ensamblado.

CAPITULO II

ANALISIS Y DISEÑO DEL SISTEMA PROPUESTO

II.1 Arquitectura del Microprocesador Z80.

Se han escrito muchos libros sobre las propiedades del software y hardware del Z80, no es la intención de este proyecto duplicar la información y el esfuerzo de otros autores; sin embargo la realización del mismo sería incompleto sin una sección que describa el procesador en algún detalle.

La figura II.1 es un diagrama de bloques de la arquitectura interna del Z80. El Z80 posee más registros internos y formas de direccionamiento que la mayoría de los microprocesadores de 8 bits; estas características le permiten procesar una mayor cantidad de información en un tiempo dado.

A continuación se da una descripción breve de la función y de la estructura de los componentes principales del microprocesador.

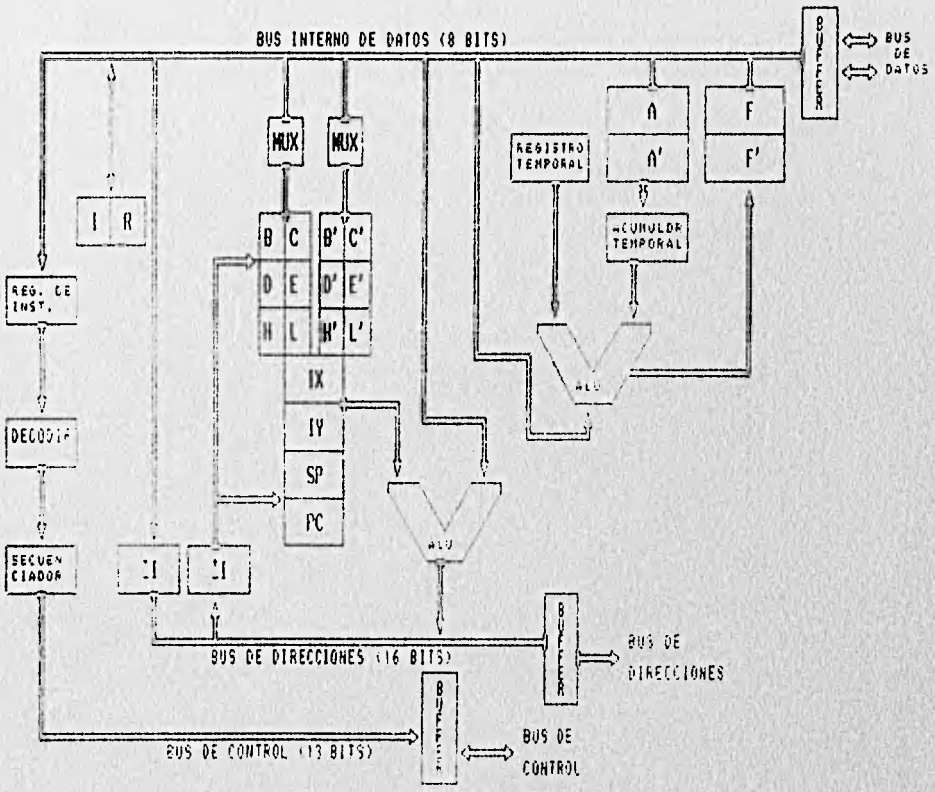


Fig. II.1 DIAGRAMA A BLOQUES DEL MICROPROCESADOR Z-80

Registros.

- Acumuladores y registros de estado.

El procesador central contiene dos pares independientes de acumulador y registro de estado, uno en el conjunto de registros principales y el otro en el conjunto de registros alternativos. El acumulador recibe los resultados de todas las operaciones lógicas y aritméticas de 8 bits, mientras que el registro de estado (F') indica la ocurrencia de condiciones especificas lógicas o aritméticas en el procesador tales como paridad, cero, signo, acarreo y desbordamiento. Una instrucción de intercambio simple permite al programador seleccionar uno u otro par de acumuladores o registro de estado.

- Registros de uso general.

Hay dos conjuntos similares de registros de uso general. El conjunto de registros principales contiene seis registros de 8 bits llamados B, C, D, E, H y L; el conjunto de registros alternativos contiene también seis registros de 8 bits que se denominan B', C', D', E', y L'. Para operaciones de 8 bits, estos registros pueden agruparse en pares de 16 bits (BC, DE, HL o BC', DE', HL'). Una instrucción de intercambio único permite al programador elegir alternativamente entre los conjuntos pares de registros.

- Registros de uso especial.

PC (Program Counter, Contador de Programa). El contador de programa contiene una dirección de 16 bits en memoria a partir de la cual se buscará la instrucción en curso. Después de la ejecución de la instrucción, el contador PC se incrementa, si el programa ha de proseguir al siguiente byte en memoria,o si ha de ejecutarse un salto o una instrucción de llamada el contenido actual del PC se sustituye por un nuevo valor.

- SP (Stack Pointer, Puntero de Pila). El Z80 permite varios niveles de "anidación" (jerarquización) de subrutinas mediante el uso de una "pila" y de un "puntero de pila": cuando se ejecutan algunas instrucciones o cuando se hacen llamadas a subrutinas, el contador PC y otros datos pertinentes pueden almacenarse temporalmente en una pila. Una pila es una zona reservada de varias posiciones de memoria, en cuya parte superior se indica el contenido del puntero de pila. Es decir, el puntero de pila indica la dirección de la entrada hecha recientemente, porque las posiciones de memoria están organizadas como un fichero de "último en entrar, primero en salir". Al examinar las entradas particulares en la pila, el procesador central retorna a un programa principal haciendo caso omiso de la profundidad de las subrutinas "anidadas" teóricamente, la pila podria tener una longitud de 64 Kbyts.

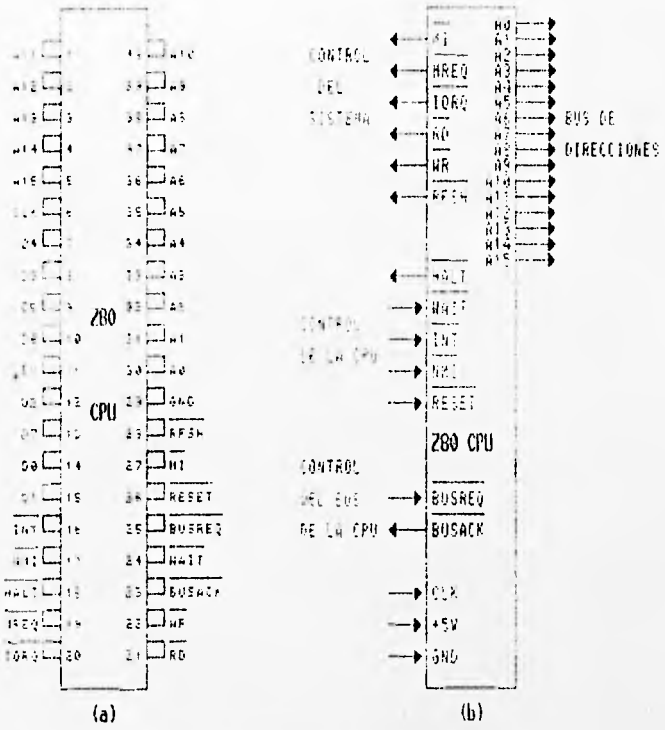


Fig. II.2.1 El microprocesador Z80. (a) Disposicion de las terminales del circuito integrado.
(b) Funciones de las terminales del Z80.

La función de cada patilla es la siguiente:

II.2.1 Control del sistema

M1 Salida, activa a nivel bajo. M1 indica que el ciclo de máquina actual es el ciclo de búsqueda del código de operación en la ejecución de una instrucción. Obsérvese que durante la ejecución de códigos de operación de dos bytes, M1 se genera en el momento en que se busca cada código de operación. Estos códigos de operación de dos bytes siempre empiezan con CB, DD, o FD Hex.
M1 también se produce con IORQ para indicar el reconocimiento de un ciclo de interrupción. (Ciclo máquina uno).

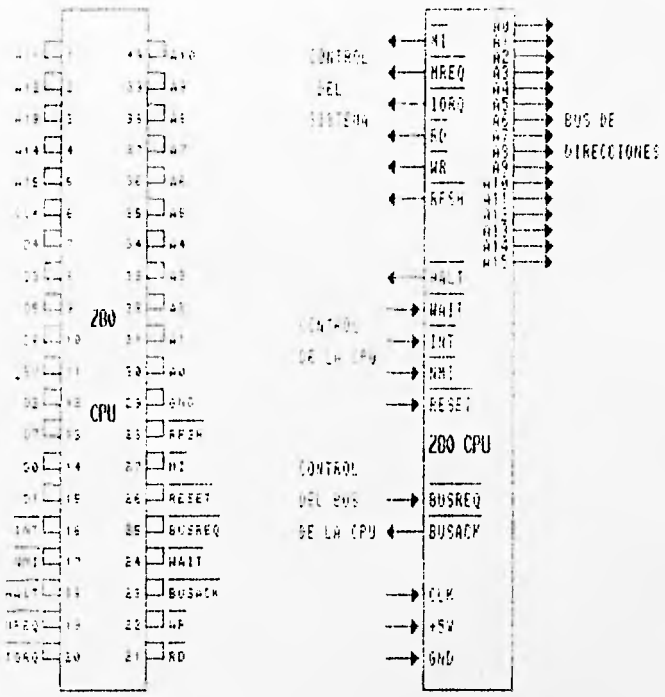


Fig. II.2.1 El microprocesador Z80. (a) Disposicion de las terminales del circuito integrado.
(b) Funciones de las terminales del Z80.

La función de cada patilla es la siguiente:

II.2.1 Control del sistema

M1 Salida, activa a nivel bajo. M1 indica que el ciclo de máquina actual es el ciclo de búsqueda del código de operación en la ejecución de una instrucción. Obsérvese que durante la ejecución de códigos de operación de dos bytes, M1 se genera en el momento en que se busca cada código de operación. Estos códigos de operación de dos bytes siempre empiezan con CB, DD, o FD Hex.
M1 también se produce con IORQ para indicar el reconocimiento de un ciclo de interrupción. (Ciclo máquina uno).

- Registros índice IX y IY.

Estos registros facilitan la manipulación de datos de tablas. Son dos registros independientes de 16 bits, que retienen las direcciones de base utilizadas en modos de direccionamiento indexado y apuntan a posiciones de memoria, cuando han de almacenarse o recuperarse datos particulares.

- Vector de interrupción I (registro vector).

Este es un registro de 8 bits que puede cargarse con una dirección de memoria de una rutina de atención, durante el proceso el control vectorizará esta dirección de memoria.

- Puntero de refrescamiento (contador de memoria (R).

Un registro de refrescamiento de memoria de 7 bits para validar señales dinámicas para el Z80, se incrementa automáticamente después de cada búsqueda de instrucción.

- Unidad aritmética y lógica, ALU (arithmetic logic unit).

Las manipulaciones lógicas y las operaciones aritméticas se realizan como partes separadas en ALU del Z80. La ALU se comunica internamente con los registros del procesador central y no es directamente accesible por el programador.

- Registro de instrucciones y control del microprocesador.

El registro de instrucciones retiene el contenido de la posición de memoria direccionada por el PC y se carga durante el ciclo de búsqueda de cada instrucción. La unidad de control del procesador central ejecuta las funciones definidas por la instrucción en el registro de instrucciones y genera todas las señales de control necesarias para transmitir los resultados a los registros adecuados.

II.2 Lineas de control del sistema.

Para programar a un microprocesador, suponiendo que se cuenta con una computadora y con las herranientas auxiliares de programación que permiten comunicarse con él, únicamente es necesario conocer a un nivel aceptable el código de instrucciones y la estructura interna del microprocesador. Sin embargo, cuando se trata de utilizar al microprocesador como elemento central en la implantación de un sistema de desarrollo, se vuelve indispendsable el conocimiento de las características de las señales eléctricas y de la función de cada una de las terminales del circuito integrado, a fin de poder interconectar al microprocesador con los otros componentes del sistema.

En esta sección se presentará una descripción del circuito integrado que constituye el Z80 y la función que desempeña cada de sus terminales. El circuito integrado del microprocesador Z80 está colocado en un encapsulado DIP de 40 terminales. La figura II.2.1 muestra su configuración.

MREQ Salida de tres estados, activa a nivel bajo. La señal de petición de meoria indica que el bus de direcciones mantiene una dirección válida para poder efectuar una operación de lectura o escritura de memoria. (Petición de memoria).

IORQ Salida de tres estados, activa a nivel bajo. La señal de IORQ indica que la mitad baja del bus de direcciones mantiene una dirección válida de E/S para efectuar una operación de lectura ó escritura de E/S. También se genera una señal IORQ con una señal M1 cuando se está reconociendo una interrupción para indicar que el vector de respuesta de intwrrupción puede ser colocado en el bus datos. Las operaciones de reconocimiento de las interrupciones se producen durante el tiempo M1, mientras que las operaciones de E/S nunca se producen durante un tiempo M1. (Petición de entrada/salida).

RD Salida de tres estados, activa a nivel bajo. RD indica que la CPU desea leer datos desde la memoria o de un dispiositivo de E/S. El dispositivo de E/S direccionado o la memoria debe utilizar esta señal para dirigir los datos al bus de datos de la CPU.(Lectura de memoria).

WR Salida de tres estados, activa a nivel bajo. WR indica que el bus de datos de la CPU mantiene un dato válido para ser almacenado en la memoria direccionada o en el dispositivo de E/S.(Escritura de memor

RFSH Salida, activa a nivel bajo RFSH indica que los siete bits menos significativos del bus de direcciones contienen una dirección de refresco para las memorias dinámicas y que la señal actual de MREQ debe ser utilizada para efectuar una lectura de refresco para todas las memorias dinámicas.(Refresco).

II.2.2 Control de la CPU.

HALT Salida, activa a nivel bajo HALT indica que la CPU ha ejecutado una instrucción de software HALT y que está esperando o bien a una interrupción no enmascarable o una interrupción enmascarable (con la máscara validada) antes de que pueda continuar la operación. Mientras está es este estado, la CPU ejecuta instrucciones NOP para mantener la memoria en estado de refresco.(Estado de Paro).

WAIT Entrada, activa a nivel bajo. WAIT indica a la CPU que la memoria direccionada o los dispositivos de E/S no están preparados. La CPU continua generando estados de espera durante todo el tiempo en que esta señal está activa. Esta señal permite que se puedan sicronizar con la CPU, la memoria o dispositivos de E/S de culaquier velocidad.(Espera).

INT Entrada, activa a nivel bajo. La señal de petición de interrupción se genera por los dispositivos de E/S. Se atenderá una petición al final de la instrucción que se está ejecutando si el flip-flop de control de las interrupciones (IFF) está habilitado y si la señal BUSRQ no está activ. Cuando la CPU acepta la interrupción, se envia una señal de reconocimiento (IORQ durante el tiempo M1) al principio del próximo ciclo de instrucciones.(Petición de interrupción)

NMI Entrada, activa mediante el flanco negativo. La 116lnea de interrupción no enmascarable tiene una prioridad más alta que INT y siempre es reconocida al final de la instrucción que se está ejecutando, independientemente del estado del flip-flop de interrupción. NMI fuera automáticamente a la CPU a un reinicio a la posición 0066 Hex. El contador de pprograma se guarda automáticamente en el stack externo de forma que el usuario puede retornar al programa que fué interrumpido.Nótese que al ejecutarse ciclos continuos de espera WAIT evitan que se termine la instrucción que se está ejecutando, y que un BUSRQ pueda eliminar a un NMI.

RESET Entrada, activa a nivel bajo. RESET fija el contador de programa a cero e inicializa la CPU. La inicialización de la CPU incluye:
- Deshabilita el flip-flop de interrupción.
- Coloca el registro I =00 hex.
- Coloca el registro R =00 hex.
- Coloca el modo 0 de interrupción.

Durante el tiempo de reset, el bus de direcciones y el bus de datos se quedan en un estado de alta impedancia y todas las señales de control de salidas pasan a estado inactivo.(Reinicio).

II.2.3 El bus de control de la CPU.

BUSRQ Entrada, activa a nivel bajo. La señal de petición de bus se utiliza para pedir que el bus de direcciones de la CPU, el bus de datos, y las señales de tres estados de control de salidas vayan a un estado de alta impedancia de tal forma que otros dispositivos puedan controlar estos buses. Cuando la señal BUSRQ es activada la CPU colocará a estos en un estado de alta impedancia en el momento en que el ciclo de máquina actual de la CPU termine.(Petición de bus).

BUSAK Salida, activa a nivel bajo. El reconocimiento de bus se utiliza para indicar al dispositivo que lo pide, que los buses de dirección de la CPU, el bus de datos, y las señales de control de bus de tres estados han sido colocadas a su estado de alta impedancia y que el dispositivo externo puede controlar ahora estas señales (Reconocimiento de bus).

CLK Entrada, una sola fase de reloj a nivel TTL.

+5V Fuente de alimentación 5V.

GND Conexión a tierra.

II.2.4 Bus de direcciones.

A(0)-A(15) Salida de tres estados, activa a nivel bajo. A0-A15 constituyen un bus de direcciones de 16 bits. El bus de direcciones proporciona la dirección para la memoria (hasta 64 Kbytes), intercambio de datos con los dispositivos de E/S. El direccionamiento de los dispositivos de E/S utiliza los 8 bits menos significativos y permite al usuario seleccionar directamente hasta 256 puertos de salida. A(0) es el bit menos significativo. (Bus de direcciones).

II.2.5 Bus de datos.

D(0)-D(7) Entrada/Salida de tres estados, activo alto. D0-D7 constituyen un bus de datos bidireccional de 8 bits. El bus de datos se utiliza para el intercambio de datos con la memoria y los dispositivos de E/S. (Bus de datos).

II.3 Software del sistema.

La función de un SISTEMA OPERATIVO es proporcionar al programador un conjunto de herramientas para ayudarle a desarrollar, depurar y ejecutar un programa. Por lo general, el sistema operativo ayuda al programador gestionando los recursos de la computadora y eliminando su implicación con manipulaciones repetitivas de código de máquina. Los sistemas operativos en las computadoras de cualquier tamaño permiten explorar absolutamente todos los recursos de la máquina con sus periféricos y manejar algunos lenguajes de alto nivel, en cambio el software de este proyecto llamado MONITOR le permite al programador introducir y leer datos de 8 bits hacia y desde la memoria. Este monitor HOLA (programado para que se anuncie como ENEP) es un programa grabado en ena EPROM, que permite explotar los recursos del sistema de desarrollo.

Estos recursos son:

-Microprocesador Z80A (Microprocesador de 8 bits).
-Espacio de memoria EPROM (compatible a 4 Kbytes).
-4 Kbytes de RAM.
-Teclado Hexadecimal.
-Display de dos caracteres para datos.
-Display de cuatro caracteres para direcciones.
-Programador de memorias.
-Programación de puertos.

Mediante el teclado permite al usuario:

-Examinar localidades de memoria.
-Modificar localidades de memoria.
-Insertar programas.
-Examinar culaquiera de los 22 registros internos del Z80.
-Modificar cualquiera de los 22 registros internos del Z80.
-Examinar el contenido de los puertos.
-Modificar el contenido de los puertos de salida.
-Correr programas.
-Inicializar el sistema.
-Simular solicitudes de WAIT a la CPU.
-Simular solicitudes de INTERRUPCION a la CPU.
-Calcular desplazamientos para saltos relativos.
-Mover bloques de memoria.
-Comparar dos áreas de memoria.
-Solicitar desplegado automático de memoria.
-Programar EPROM's.

La Figura II.3.1 es un diagrama de flujo de alto nivel del sistema operativo del sistema de desarrollo. Existen cuatro puntos de entrada distintos para el sistema operativo.

RESET/AL APLICAR TENSION. Este camino se toma cuando se pulsa la tecla RESET, o cuando se aplica tensión al sistema. La primera función de este camino es realizar una prueba simple de la memoria en el área de memoria/escritura utilizada por el sistema operativo para alamacén y datos. Si se completa con éxito esta prueba de la RAM, el sistema operativo coloca en el display "HOLA" y luego pasa el control a la secuencia de tareas que inicia con GUARDAR EL ESTADO DE LA CPU.

RST 38H. Un reinicio a la posición 0038H es es segundo método para regresar el control al sistema operativo del MONITOR. Este método se utiliza para volver al sistema operativo (MONITOR) desde un programa definido por el programador.

NMI. Una tercera forma de entrar al sistema operativo es mediante una interrupción no enmascarable. Se puede provocar una interrupción no enmascarable en la CPU o bien pulsando la tecla BREAK o colocando a tierra momentáneamente la entrada NMI de la CPU.

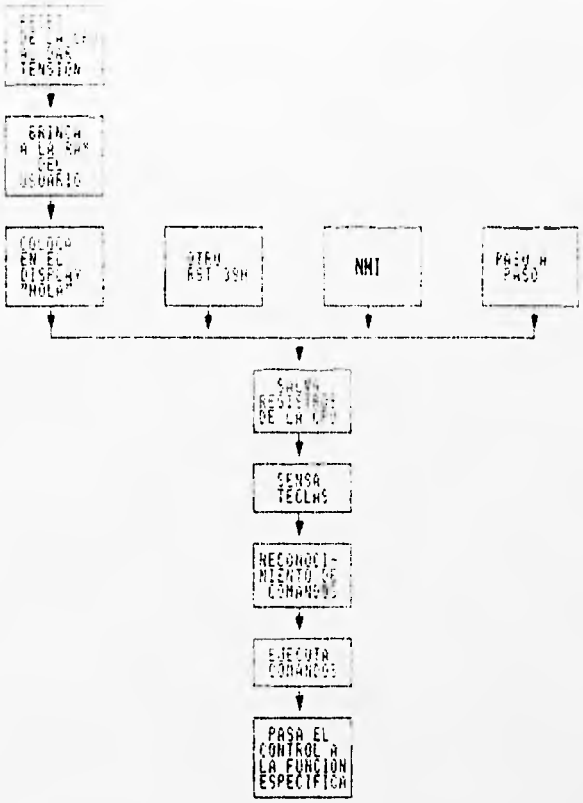


Fig. II.3.1 Diagrama de flujo del software del sistema de desarrollo.

En la figura anterior se puede ver que existe una diferencia entre pulsar la tecla BREAK y RESET. La diferencia estriba en la realización de dos funciones adicionales cuando se pulsa RESET, es decir, la prueba de la memoria y el desplegado "HOLA".
Al realizar estas tareas, el sistema operativo inicializa los registros de la CPU a cero. Este es un método que se utiliza varias veces para poder pasar el control desde un programa del usuario al sistema operativo del procesador central sin destruir el contenido de los registros de la CPU.

PASO A PASO. La última forma de entrar al sistema operativo del sistema de desarrollo es para avanzar paso a paso por un programa del usuario. Cada vez que se avanza un paso, instrucción por instrución, a través de un programa que se ha introducido en la RAM, la CPU Z80 está ejecutando alternativamente sus instrucciones y también varias instrucciones del sistema operativo. Con cada transferencia de control al sistema operativo después de la ejecución de su isntrucción, el control pasa a través de este último camino.

Una vez que la CPU está ejecutando el sistema operativo, habiendo entrado en cualquiera de los cuatro puntos precedentes, se realizan una serie de seis funciones básicas. En primer lugar, el estado de todos los registros de la CPU es guardado introduciendo sus contenidos en el stack del sistema. En el apéndice se darán, la magnitud, posición y utilización exacta de las áreas de stack y de datos del sistema. En segundo lugar, la próxima tarea realizada por el sistema operativo es la de componer y mostrar los datos apropiados en el periférico teclado/display. Cabe mencionar que la naturaleza de este display está determinada por la posición de los segmentos iluminados y los contenidos de los registros que se van a mostrar. La tercera tarea consiste en esperar a que el usuario pulse una tecla. Como se verá en la información más detallada de las rutinas de entrada al display y al teclado presentadas más adelante en este capítulo, el sistema operativo lee alternativamente el puerto de entrada al display y escribe en el puerto de salida del display. Esta alternación en la lectura y escritura es necesaria para mantener renovados los LEDs en el display, de una forma muy parecida a la de la CPU que debe refrescar las memorias RAM dinámicas del sistema de desarrollo. Una vez que ha detectado una entrada desde el teclado, el sistema de software analiza esta entrada, la interpreta, y vuelve a leer más caracteres de entrada, si es necesario. Una vez que se ha ejecutado un comando completo, se pasa el control a la función que lo generó. El resto del sistema operativo del sistema de desarrollo consiste en un conjunto de módulos de software, uno para cada posible comando del teclado. Así, existen módulos que implementan las teclas GO, DUMP, LD, ST e INC así como un módulo que maneja todas las entradas numéricas hex (0,1,...F) haciendo desplazar los cuatro dígitos que están en el display de datos un espacio hacia la izquierda y mostrando el número asociado con la última tecla pulsada. Con cada tecla del teclado, existe un módulo de software asociado al cual se le pasa el control. Cuando se completa la ejecución del módulo, se devuelve el control al sistema operativo mediante el segundo punto de entrada (RST 38H) que se muestra en la figura anterior.

Existen dos caminos mediante los cuales el usuario puede tomar el control de la CPU. El primero es mediante el comando GO. Al detectar que se ha pulsado la tecla GO, el sistema operativo pasa el control al "módulo GO", que realiza un solo paso, y ejecuta un paso a la dirección contenida en el registro PC. El segundo método por el cual un programa del usuario es ejecutado es mediante el sistema paso a paso. En este caso, como se mencionó anteriormente, el control se alterna entre el programa del usuario y el sistema operativo.

El listado completo del programa MONITOR se anexará como apéndice A de este trabajo.

II.4 Rutinas de trabajo del sistema.

Frecuentemente se encuentra la necesidad de que en diferentes puntos de un programa se lleve a cabo un proceso que requiere la ejecución del mismo grupo de instrucciones. En otras ocaciones es común que la misma tares forme parte de distintos programas.

Una alternativa de solución consiste en reproducir dentro de un programa el conjunto de instrucciones que realizan el proceso, tantas veces como sea requerido (Figura II.4.1). Sin embargo, aunque ésta parezca la manera más directa de atacar el problema, es también, la mayoría de las veces, la más ineficiente, al menos en lo que al uso de memoria se refiere. En cuanto al flujo del programa, éste es meramente secuencial.

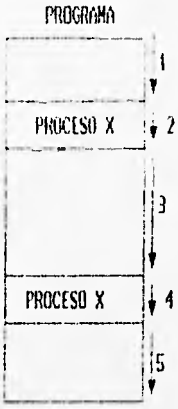


Fig. II.4.1 Programa en el que existe un proceso que se repite dos veces.

En la figura II.4.1 el flujo de ejecución se indica con las flechas. Las flechas 2 y 4 representan la ejecución del proceso X en dos ocsciones.

La solución más efectiva, en términos del ahorro de memoria, se obtiene si el grupo de instrucciones que se repite aparece una sola vez en el programa, pero con capacidad para ser ejecutado desde todos los puntos en que aquél lo pide. La estructura de programación que implementa esta solución es la subrutina.

Una SUBRUTINA es simplemente un conjunto de instrucciones al que se tiene acceso desde cualquier punto del programa principal. Como una subrutina conceptualmente queda fuera del flujo secuencial de ejecución del programa principal, son necesarios ciertos mecanismos para poder llegar a ella, y una vez que se han ejecutado las instrucciones que la componen, debe ser posible regresar al punto donde se quedó la ejecución del programa principal.

La acción de pasar del programa principal a la subrutina se denomia "llamada" a la subrutina y la acción de volver al programa principal después de llevar a cabo las tareas determinadas por la subrutina se conoce como "retorno" de la subrutina.

La figura II.4.2 muestra el mismo programa de la figura II.4.1, con la diferencia de que el proceso X se ha codificado como una subrutina. Este cambio ha afectado la secuencia de ejecución del programa de la siguiente manera: las lineas del programa principal son ejecutadas sucesivamente hasta que se encuentra la primera instrucción CALL. Su ejecución resulta en la transferencia de la secuencia de ejecución al proceso X (flecha 2a), después de lo cual éste se ejecuta como cualquier otra sección del programa (flecha 2). La última instrucción de la subrutina es un RET que causa del regreso de la secuencia de ejcución del programa principal (flecha 2b). La instrucción ejecutada después del RETURN es la que sigue al CALL en el programa principal. La ejecución del programa continúa normalmente hasta que aparece una segunda llamada al proceso X. De nuevo se transfiere el control a la subrutina (flecha 4a) y el proceso X es ejecutado (flecha 4). Al llegar a la instrucción RET ocurre un retorno al programa principal, esta vez a la instrucción siguiente al segundo CALL (flecha 4b).

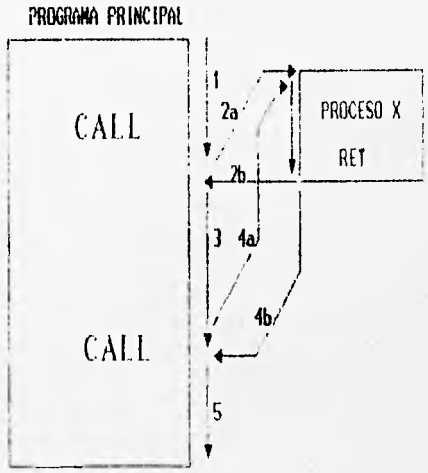


Fig. II.4.2 Utilizacion de una subrutina para evitar la repeticion del proceso X.

En el APENDICE A correspondiente a la sección II.3 Software del sistema, se presenta además una documentación adicional de las rutinas de utileria que contiene el MONITOR de ester sistema de desarrollo. Contiene siete puntos de interrupción por software (break points) para ser utilizados por el usuario (desde el restart 08Hex hasta el 38Hex). Contiene 29 subrutinas de utilerias que pueden ser llamadas por el usuario para facilitar su trabajo de programación con el consiguiente ahorro de memoria y tiempo.

II.5 Código de instrucciones

La función de cualquier sistema basado en un microprocesador se implanta por medio de transferencia de datos entre los registros del mismo, la memoria y los dispositivos de E/S, y de transformaciones de datos que ocurresn principalmente en los registros internos del microprocesador.

Las clases de transferencias y de transformaciones que son posibles se sepecifican en el CODIGO DE INSTRUCCIONES.Cada microprocesador está diseñado para ejecutar un código particular de instrucciones.

II.5.1 Formato de las instrucciones.

Una instrucción especifica la operación que debe ser realizada por el microprocesador. Cada instrucción está dividida en secciones o CAMPOS, donde cada campo es un conjunto de bits que proporciona parte de la información requerida para saber el tipo de operación a ejecutar y la localización de los operandos en los que ésta se realizará.

Desde un punto de vista simplificado, todas las instrucciones pueden ser representadas como un código de operación, seguido de campos adicionales que contienen los operandos de la instrucción, generalmente constantes numéricas o direcciones.

El CODIGO DE OPERACION determina la actividad que se llevará a cabo. Además indica cuál es el contenido de las siguientes localidades de memoria en el programa, es decir, si contiene un dato, una dirección, un desplazamiento u otro código de operación. or razones de eficiencia es conveniente que el código de operación resida totalmente en la primera palabra de la instrucción; éste es un factor limitante en el número de instrucciones disponibles en un microprocesador. Cunado su longitud de palabra es de 8 bits, se pueden tener hasta 256 códigos de operación diferentes en un byte.

En el caso del Z80, los códigos de operación son, en general, de un byte, a excepción de instrucciones especiales que requieren un código de operación de dos bytes. Tomando en cuenta la información que necesita cada instrucción además del código de operación, en el Z80 existen instrucciones de uno, dos, tres o cuatro bytes. (Fig.II.5.1).

Por cada instrucción, la unidad de control tiene que ralizar un ciclo máquina durante el cual se lee este byte de memoria. Por lo tanto, mientras más corta sea una instrucción más rápido será ejecutada.

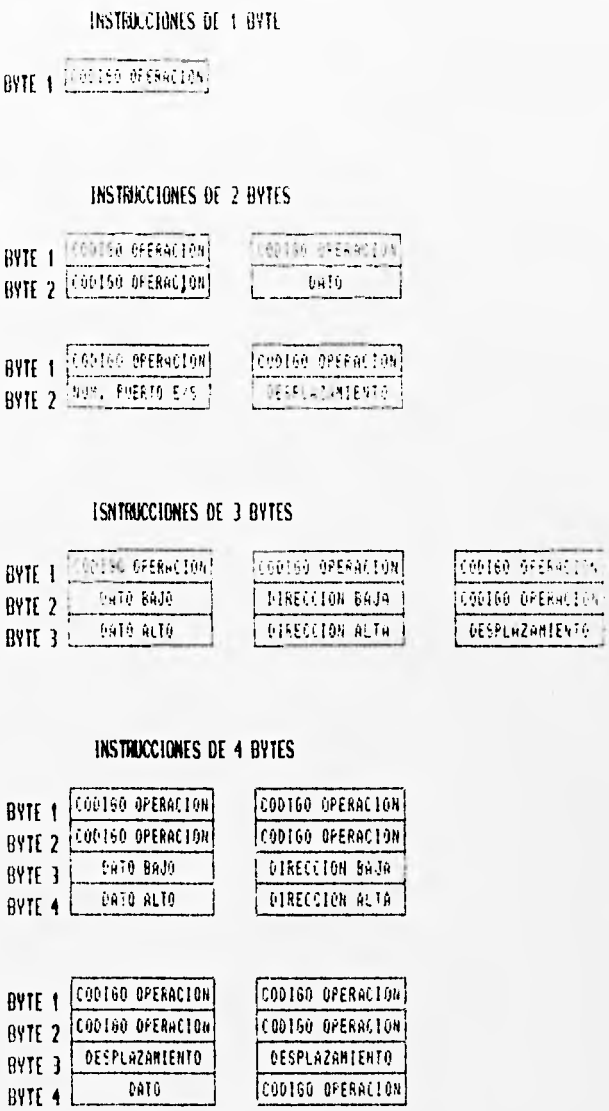


Fig. II.5.1 Formatos de las instrucciones del Z-80.

II.5.1.1 Instrucciones de un byte

Las instrucciones de un byte son, en principio, las más rápidas. Un ejemplo común de estas instrucciones en el Z80 el LD r,r'. Instrucción que significa: "carga el contenido del registro r' en el registro r". Esta es una operación de registro a registro, la cual permite la transferencia de información entre dos de los registros internos del microprocesador..

Cualquier instrucción debe ser representada internamente en sistema binario. La expresión "LD r,r' " es meramente simbólica y se denomina el mnemonico de la instrucción; sin embargo, ésta es la forma más conveniente de mostrarla desde el punto de vista del programador. El código binario que cosntituye la instrucción dentro de la memoria es: 0 1 D D D S S S.

Esta representación todavia es parcialmente simbólica. Cada una de las tres letras S y D ocupa el lugar de un bit. Las letras "D D D" se refieren a los tres bits que identifican al registro que actúa como destino de la información. Estos tres bits permiten seleccionar uno de los ocho registros posible. Por ejemplo la clave para el registro C es 001.

De modo similar, "S S S" representa los tres bits que apuntan al registro donde se origina la transferencia ' de información. La convención que se sigue es que el registro r' es el origen y el registro r es el destino. Aun cuando aparentemente el mnemónico de la instrucción está al revés (primero el destino y después el origen), se ha escogido esta convención para mantener la compatibilidad con la notación binaria, la cual ha sido diseñada para facilitarle la tarea a la unidad de control.

II.5.1.2 Instrucciones de dos bytes.

Algunas instrucciones requieren que un dato esté inmediatamente después del código de operación; entonces son necesarios dos bytes.

Por ejemplo, la instrucción ADD A,n es una instrucción de dos bytes, que suma el contenido del segundo byte de la instrucción con el acumulador. Este segundo byte es un dato o constante numérica que es tratado como un grupo de ocho bits sin ningún significado particular. El código binario de esta instrucción es 1 1 0 0 0 1 1 0, seguido de un dato de ocho bits.

II.5.1.3 Instrucciones de tres byts.

En ocaciones, la instrucción puede necesitar la especificación de alguna dirección. Una dirección requiere de 16 bits, osea dos bytes. De lo anterior se desprende que, incluyendo el código de operación, la instrucción es de tres bytes.

La instrucción JP pq tiene como función producir un salto en el programa a la dirección señalada por "pq". Como las direcciones tienen una longitud de 16 bits, la instrucción ocupa tres bytes.

En binario, esta instrucción está representada por el código 1 1 0 0 0 0 1 1, seguido de 8 bits para el byte bajo de dirección y 8bits más para el byte alto de la misma.

II.5.2 Clases de Instrucciones

Con base en la función que desempeñan, las instrucciones que es capaz de ejecutar el microprocesador Z80 se pueden agrupar en varias clases, cada una de las cuales se analizará en las secciones siguientes.

II.5.2.1 Instrucciones de transferencia de datos.

Las instrucciones de transferencia de datos mueven información entre los registros internos de la CPU, o entre los registros y la memoria externa. Todas estas instrucciones deben especificar el lugar de donde la información será obtenida (origen) y el lugar donde será trasladada (destino).

Las instrucciones de transferencia de datos en el Z80 se pueden agrupar en cinco categorías: transferencias de 8 bits, transferencias de 16 bits, intercambios, operaciones de pila y transferencias de bloques.

II.5.2.2. Instrucciones de procesamiento de datos.

Las instrucciones de procesamiento de datos operan sobre la información almacenada en acumulador y los registrosc de propósito general o en localidades de memoria externa. Los resultados de lasoperaciones se colocan en el acumulador, y el estado de las banderas se determina de acuerdo con aquéllos.

Estas instrucciones incluyem: operaciones aritméticas (suma y resta), operaciones lógicas (AND,OR,XOR,complemento), operaciones de incremento y decremento, operaciones de rotación y desplazamiento, y operaciones de manipulación de bits (set, reset, test).

II.5.2.3 Instrucciones de transferencia de control.

Las instrucciones de transferencia de control son aquéllas que trasladan el control del programa a una dirección determinada. Estas instrucciones cambian el flujo normal de ejecución a otro en el que una sección diferente del programa es procesada.

Existen dos tipos importantes de instrucciones de transferencia de control: las que trasladan el control dentro del programa principal, denominadas "saltos", y las que temporalmente lo transfienren a una sección o subrutina fuera del programa principal. La transferencia de control puede ser condicional (dependiendo del estado de ciertas banderas en el microprocesador) o incondicional.

II.5.2.4 Instrucciones de E/S.

El grupo de instrucciones de E/S en el Z80 permite una amplia gama de transferencias de datos entre las localidades de memoria externa, los registros internos de propósito general y los dispositivos externos de E/S.

programa. Estas instrucciones afectan de modo de operación de la CPU o manipulan la información de su estado (status) interno.

El CODIGO DE INSTRUCCIONES completo se anexará como apéndice B al final del presente trabajo.

II.6.1 Operación del microprocesador.

En esta sección se establecen las bases de operación del microprocesador Z80. Para entender mejor los conceptos se dan algunas definiciones y explicaciones basadas en diagramas de tiempos.

II.6.1.1 Definiciones.

Durante la operación normal del microprocesador todas las instrucciones son procesadas secuencialmente en tres fases: OBTENCION (Fetch), DECODIFICACION (Decode), Y EJECUCION (Execute). Cada una de estas fases requiere del transcurso de varios ciclos de reloj.

A un ciclo de reloj se le llama ESTADO (State). En el Z80 un estado se define como el periodo entre dos transiciones positivas consecutivas de la señal de reloj (Figura II.6.1.1). Los estados se identifican con la letra "T" : T1,T2,T3,etc.

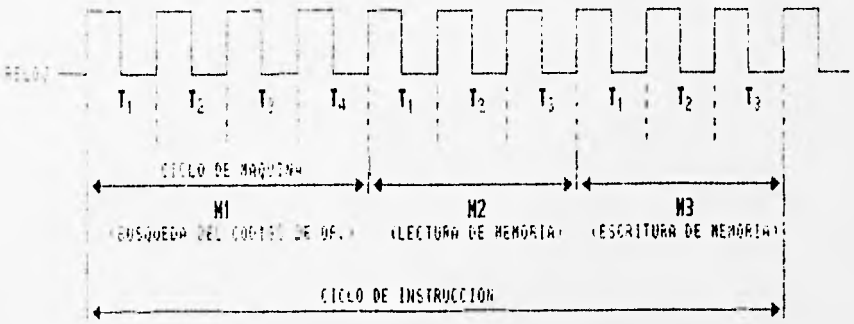


Fig. II.6.1.1 Nomenclatura de la operacion del microprocesador.

La obtención y ejecución de una instrucción constituye un CICLO INSTRUCCION (Istruction Cycle), el cual consiste en una o más operaciones elementales en las que el microprocesador lleva a cabo alguna transferencia de información con la memoria o con un dispositivo de E/S. Cada una de estas operaciones recibe el nombre de CICLO DE MAQUINA (Machine Cycle) y se desarrolla a través de una sucesión de estados (Figura II.6.1.1). Los ciclos máquina se identifican con la letra "M": M1,M2,etc.

Un ciclo de máquina del Z80 puede durar de tres a seis estados y un ciclo de instrucción puede tener de uno a seis ciclos máquina. La duración de cada ciclo máquina, y, por tanto, de cada ciclo de instrucción es dependiente de la instrucción en particular. Invariablemente, el primer ciclo de máquina de una instrucción (M1) se emplea para la obtención y decodificación del código de operación, y puede durar cuatro, cinco o seis estados. Algunas instrucciones no requieren de más ciclos de máquina fuera del utilizado en la obtención de la instrucción. Otras necesitan ciclos adicionales para transferir datos entre la CPU y la memoria y los puertos de E/S. Estos cilcos de máquina pueden tener de tres a cinco estados.

Con base en lo anterior, las instrucciones más rápidas del Z80 duran 4 estados; por ejemplo, INC r requiere de un ciclo de máquina de cuatro estads, lo que equivale a 1.6 microsegundos para un reloj de 2.5MHz. Las instrucciones más lentas son de 23 estados; por ejemplo, INC(IX+d) necesita 6 ciclos de máquina, el primero de seis estados, el segundo de cinco estados y los cuatro restantes de tres estados cada uno, dando un total de 23 estados que representan 9.2 microsegundos para un reloj de 2.5MHz.

II.6.1.2 Operaciones básicas y sus diagramas de tiempos.

Como se mencionó en la sección anterior, todas las instrucciones del Z80 son meramente una serie ciclos de máquina, en cada uno de los cuales el microprocesador realiza una operación especifica. Dentro de estas operaciones básicas están las siguientes:

- Obtención del código de operación de una instrucción (OP code fetch).
- Lectura de un dato de la memoria (Memory data read).
- Escritura de un dato en la memoria (Memory data write).

Para entender mejor la relación que existe entre las señales del bus de datos, del bus de direcciones y de las lineas de control descritas anteriormente, es conveniente examinar los diagramas de tiempos de las diferentes operaciones que efectúa el microprocesador. Estos diagramas muestran el comportamiento de las señales en el lapso que comprende la ejecución de un ciclo de máquina. A continuación se analizarán los diagramas de tiempos de las tres operaciones básicas mencionadas arriba.

La obtención del código de operación de una instrucción se refiere al ciclo de máquina en el cual el microprocesador lee de la memoria un byte, que será interpretado como el código de operación de esa instrucción. La Figura II.6.1.2.1 muestra el diagrama de timepos para esta operación, también llamada ciclo M1. Un ciclo de máquina comienza con la transición de 0 a 1 de la señal de reloj.

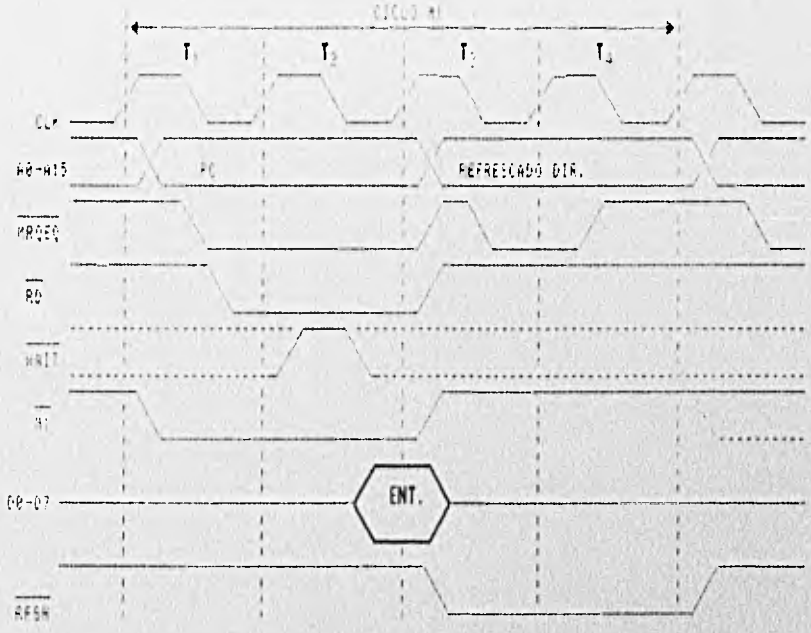


Fig. II.6.1.2.1 Diagrama de tiempos del ciclo de obtencion de una instruccion.

En la figura anterior se observa que al inicio del primer ciclo de reloj (T1), la señal M1 se activa (baja a 0 lógico), indicando que se trata de una operación para la obtención de una instrucción. Al mismo tiempo, el contenido del contador del programa (PC) aparece en el bus de direcciones. Medio estado después, la señal de lectura RD también se activa, para indicarle a la memoria que coloque la información en el bus de datos. Durante el segundo estado (T2) se incrementa automáticamente el PC y la memoria tiene tiempo de poner en la linea de datos la instrucción almacenada en la localidad especificada por la dirección. El microprocesador utiliza la subida del reloj en el estado T3 para leer la instrucción y para desactivar las señales M1 y RD (regresan a 1 lógico). Durante los siguientes dos estados (T3 y T4) se decodifica el código de operación y si es posible el microprocesador ejecuta la instrucción. Es necesario aclarar que en los diagramas aparecen señales que por el momento no se explican, pero se aclararán posteriormente.

En la operación de lectura de memoria se transfiere un dato (información que no sea el código de operación de una instrucción) de una localidad de memoria al microprocesador, mientras que en la operación de escritura en memoria ocurre una transferencia de información del microprocesador a una localidad de memoria. En la Figura II.6.1.2.2 están los digramas de tiempos correspondientes a cada una de estas operaciones.

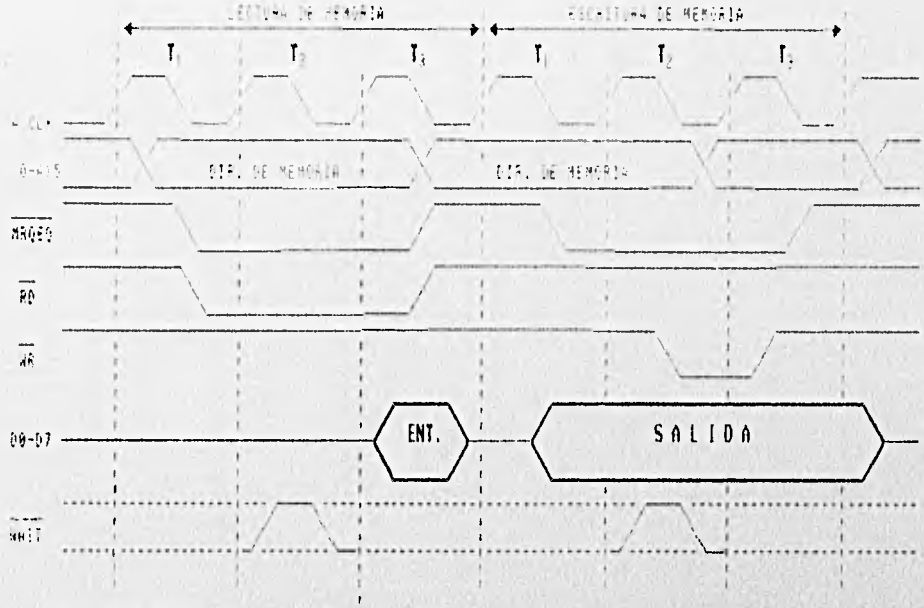


Fig. II.6.1.2.2.1 Diagrama de tiempos de los ciclos de lectura o escritura en memoria.

Al inicio del primer estado (T1), el microprocesador coloca en el bus de direcciones la ubicación de la localidad de memoria asociada a la transferencia. Esta dirección se obtiene del PC cuando el dato es parte de la instrucción, y cuando no es así, entonces se obtiene de alguno de los registros apuntadores de 16 bits (BC, DE, HL, IX, IY, o SP).

En el caso de una operación de lectura, medio estado después se activa la señalc de lectura RD, como sucedió en la obtención de una instrucción. En la mitad del tercer estado (T3), el microprocesador lee el dato colocado por la memoria y RD vuelve a 1 lógico, con lo que finaliza el ciclo de máquina.

Cuando se trata de una operación de escritura, en el microprocesador coloca el bus de datos la información que va a escribir a partir de la segunda mitad del primer estado (T1). Entre la segunda mitad del segundo estado (T2) y la primera mitad del tercer estado (T3), se genera un pulso a 0 lógico en 1 señal de escritura WR, con el cual el microprocesador ordena a la memoria que almacene el dato que se encuentra en ese momento en el bus de datos. Las líneas de datos mantienes la información válida hasta el término del tercer y último estado del ciclo de máquina.

Por ejemplo, la instrucción LD (pq),A es una instrucción de tres bytes que almacena el contenido del acumulador con la localidad de memoria, cuya dirección (pq) está indicada por el segundo y tercer bytes de la instrucción. Su ciclo instrucción consta de cuatro ciclos de máquina, el primero de cuatro estados y los otros de tres; en total 13 estados (5.2 microsegundos para un reloj de 2.5 MHz). Se requiere de tres ciclos de máquina para obtener la instrucción de la memoria. En em primero (M1) se lee el código de operación (OP code fetch) y se coloca en el registro de instrucciones, doonde es decodificado. Lueo se leen los dos bytes de la dirección por medio de dos ciclos de lectura de memoria (memory read), y se almacenan temporalmente dentro del microprocesador. En este punto, la instrucción completa está en el microprocesador y puede ser ejecutada. La ejecución se lleva a cabo durante el cuarto ciclo de máquina (M4) y consiste en colocar en el bus de direcciones los dos bytes obtenidos durante M2 y M3, luego poner el contenido del acumulador en el bus de datos y, en seguida, efectuar un ciclo de máquina de escritura en memoria (memory write), con lo cual la localidad apuntada por la dirección queda grabada en el contenido del acumulador. Con M4 termina el procesamiento de esta instrucción y el microprocesador puede pasar a obtener el primer byte de la siguiente instrucción en un nuevo ciclo de máquina M1.

En síntesis, el ciclo de instrucción de LD (pq),A se puede expresar como:

Mnemónico	Byte de instrucción	Ciclo de máquina	Estados
LD (pq),A	Código de operación	Obtención de inst.	4
	Dirección baja	Lectura de memoria	3
	Dirección alta	Lectura de memoria	3
		Escritura en memoria	3

La Figura II.6.1.2.2.2 presenta un diagrama simplificado de la instrucción LD (pq),A.

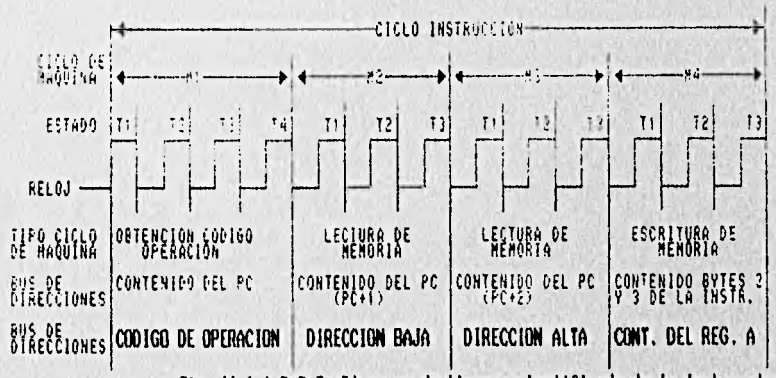


Fig. II.6.1.2.2.2 Diagrama de tiempos simplificado de la instruccion LD (pq),A.

CAPITULO III
INTERFACES.

CIRCUITO DE ENTRADA/SALIDA PARALELO DEL Z-80.

III.1 Puertos de Entrada/Salida.

Dada la gran variedad de dispositivos periféricos que pueden ser conectados a un sistema de desarrollo y las características especiales, tanto eléctricas como funcionales, de cada uno de ellos, la transferencia de información entre el microprocesador y el periférico no se efectúa de manera directa sino a través de ciertos elementos externos, los cuales reciben información proveniente del microprocesador y la envían a los dispositivos de salida o recaban la información originada en los dispositivos de entrada y la transmiten al microprocesador. A estos elementos se les da el nombre de PUERTOS DE ENTRADA/SALIDA.

Los puertos de Entrada/Salida son básicamente registros externos. Algunos microprocesadores proporcionan señales de control que permiten que los registros externos que forman los puertos de E/S ocupen un espacio de direcciones separado, es decir, distinto del espacio de direcciones de los registros externos que componen la memoria. Cuando los puertos tienen asignado un espacio de direcciones separado, se dice que está en modo de ENTRADA/SALIDA AISLADA o E/S ESTANDAR. Por el contrario, cuando se ubican dentro del mismo espacio que la memoria, se dice que están en modo de ENTRADA/SALIDA MAPEADA o PROYECTADA EN MEMORIA.

Además, un puerto de E/S contiene circuitos para el control de la transferencia de datos, así como para el acoplamiento con el dispositivo externo. Es importante señalar que el microprocesador únicamente tiene relación con el puerto, no tiene comunicación directa con el periférico; es el puerto el que tiene relación directa con el periférico.

III.1.1 Entrada/Salida Aislada.

Para que el microprocesador pueda implementar el modo de E/S aislada (isolated I/O) son indispensables las siguientes condiciones:

- El microprocesador debe proporcionar señales de control que permitan distinguir entre una operación con un puerto y una referencia a memoria.

- El código de instrucciones debe tener instrucciones especiales con las que se pueda leer (entrada) o escribir (salida) en los puertos.

El Z-80 cumple con los requisitos anteriores y, por lo tanto, permite la interconexión de puertos en el modo de E/S aislada.

Como normalmente un sistema de desarrollo no necesita un número muy grande de puertos, es común que no se utilicen todas las líneas de dirección del microprocesador en el acceso a puertos en el modo de E/S aislada. En el caso del Z-80, éste usa únicamente las ocho líneas menos significativas del bus de direcciones (A0-A7) para direccionar un puerto. Con esto el sistema puede accesar un máximo de 256 direcciones asociadas con puertos de entrada o salida.

III.1.2 Entrada/Salida Mapeada a Memoria.

El modo de E/S mapeada a memoria (memory mapped I/O) se basa en que tanto las localidades de memoria como los puertos de E/S se consideran como registros externos desde el punto de vista del microprocesador. Entonces, las instrucciones que hacen referencia a la memoria también pueden transferir datos entre un dispositivo periférico y el microprocesador, siempre y cuando el puerto de E/S que los interconecta se encuentre dentro del espacio de direcciones de memoria, es decir, controlado por las señales de control para memoria, que en el caso del Z-80 son MREQ, RD, y WR. De esta forma, el registro asociado con el puerto de E/S es tratado simplemente como una localidad de memoria más.

III.1.3 Comparación entre E/S aislada y E/S mapeada a memoria.

El modo de E/S aislada presenta las siguientes ventajas:

- Como se usan instrucciones especiales para E/S, en un programa éstas pueden distinguirse fácilmente de las instrucciones que hagan referencia a memoria.
- Como sólo se utilizan ocho líneas en el direccionamiento de un puerto, se necesitan menos circuitos para su decodificación.
- Como el número de puerto se puede representar en un byte, las instrucciones son más cortas.
- Como los puertos están asigandos a un espacio separado de la memoria, se tiene disponible la capacidad total de direccionamiento del microprocesador para circuitos de memoria.

Las desventajas de estre método son:

- La capacidad de procesamiento y flexibilidad de las instrucciones de E/S es en general muy restringida.
- Se debe dedicar al menos una terminal del circuito integrado del microprocesador para la señal de control que distingue las operaciones con puertos de las operaciones con memoria.

Por otra parte, el modo de E/S mapeada a memoria tiene como ventajas:

- Permite la utilización de la gran variedad d instrucciones que hacen referencia a la memoria, para la transferencia de información y la ejecución de operaciones aritméticas o lógicas directamente en los puertos, sin necesidad de transferir los datos a los registros internos del microprocesador.
- Reduce el número de lineas de control que debe tener el microprocesador.

Sus desventajas son las siguientes:

- Cada puerto implantado de este modo disminuye en uno las direcciones disponibles para memoria.
- Es necesario decodificar las 16 lineas de direcciones para seleccionar el puerto.
- Las instrucciones que hacen referencia a la memoria requieren de dos bytes para representar la dirección, por lo tanto son más largas y también pueden ser más lentas.

De lo anterior se concluye que ninguno de los dos modos de E/S es claramente mejor que el otro. Por lo tanto, la decisión de cuál utilizar depende de las caracteristicas particulares que tenga el sistema en cuestión. OJO,,, AQUI DEBO DEFINIR QUE MODO DE ENTRADA/SALIDA SE UTILIZO EN EL SISTEMA MINIMO.

III.2 Puertos de Entrada/Salida.

Los puertos (de entrada/salida) proporcionan un metodo para trasferir información binaria entre el almacenamiento interno, tal como la memoria y los registros de la CPU, y el mundo exterior.

El propósito de los puertos es resolver las diferencias de comunicación existentes entre el usuario y la CPU.

En su forma más elemental, un puerto de entrada está compuesto sólo por un buffer de tercer estado y con más frecuencia por un buffer de tercer estado junto con un registro de almacenamiento (latch). El buffer de tercer estado tiene la función de controlar, es decir, aislar o permitir el flujo de información del puerto al bus de datos del microprocesador. El registro tiene la función de almacenar temporalmente la información generada por el dispositivo periférico de entrada hasta que pueda ser leida por el microprocesador (Fig. III.1). El dispositivo de entrada controla el almacenamiento de un dato en el registro, mientras que el microprocesador determina el estado del buffer (activo o en estado de alta impedancia) por medio de un pulso de selección.

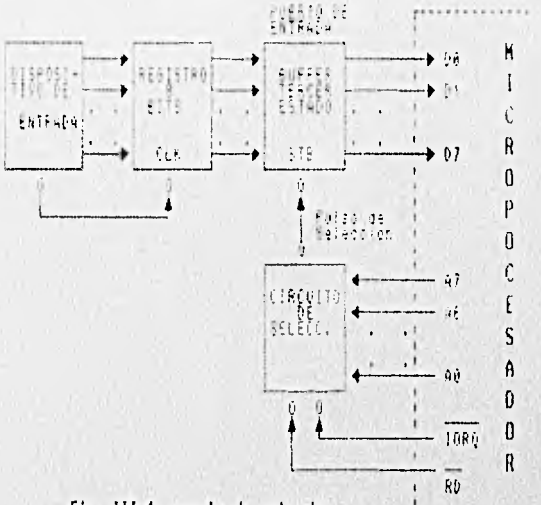


Fig. III.1 puerto de entrada

La concepción básica de un puerto de salida es un simple registro de almacenamiento (latch), que conecta al bus de datos con el dispositivo de salida. El microprocesador coloca en el registro el dato que va a ser enviado al dispositivo periférico por medio de un pulso de selección aplicado a la entrada de reloj del registro. (Fig. III.2).

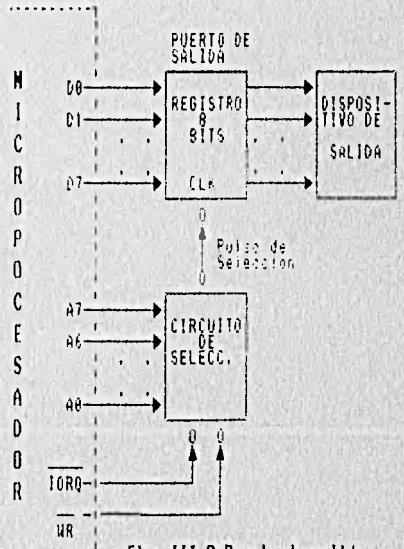


Fig. III.2 Puerto de salida

III.3 Circuitos Programables de Entrada/Salida

Los puertos de E/S pueden implantarse con circuitos SSI, MSI, LSI Sin embargo, para minimizar el número de componentes en general se usan circuitos MSI o LSI.

Existen circuitos LSI que contienen combinaciones de registros, buffers y banderas formando puertos de E/S. un solo circuito puede contener varios puertos. Estos circuitos tienen la gran ventaja de que son programables; es decir, el modo de operación de cada puerto se establece por medio de una secuencia de instrucciones en el programa. Dentro de estos circuitos se encuentra el PIO (Parallel Input Output) implantado con el versátil M8255A de Intel.

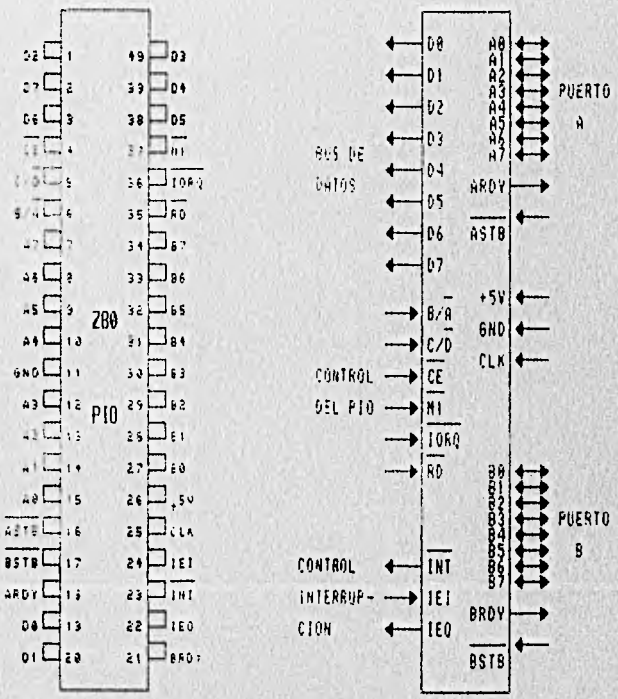


Fig. II.3.3 (a) Asignacion de terminales en el encapsulado del PIO.
(b) Funciones de las terminales del PIO.

III.4 Descripción General del PIO.

El PIO es un circuito programable con tres puertos de E/S que proporciona un medio de acoplamiento, compatible con la lógica TTL, entre dispositivos periféricos y el microprocesador Z-80. La CPU puede configurar al PIO para la interconexión con una amplia gama de periféricos, sin necesidad de compuertas adicionales. Dentro de los dispositivos que son totalmente compatibles con el PIO se incluyen la mayoría de los teclados, impresoras, programadores de EPROM, etc.

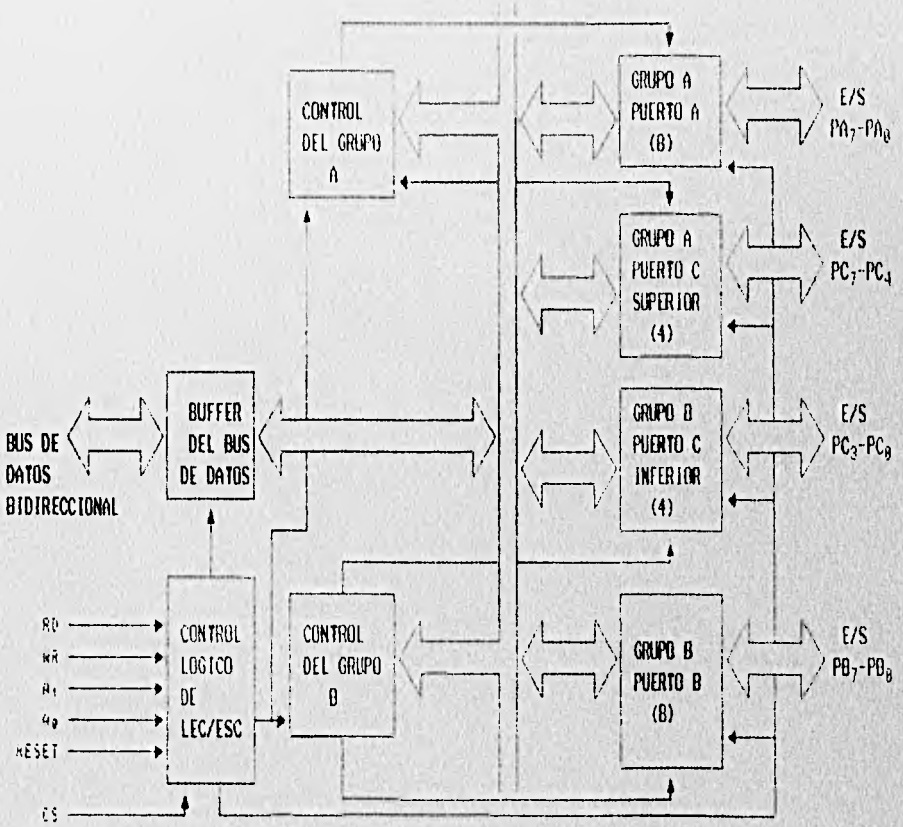


Fig. III.4.1 Diagrama a bloques del PIO.

El PIO se fabrica en un encapsulado DIP de 40 terminales. Para su operación se requiere de una fuente de 5V y una señal de reloj, que normalmente son las mismas que utiliza el microprocesador Z-80. Todas sus entradas y salidas son compatibles con la lógica TTL. La Figura III.3.1a muestra la configuración de las terminales en el circuito y la Figura III.3.1b las muestra agrupadas en cuanto a sus funciones.

III.4 Estructura interna del PIO.

Un diagrama a bloques del PIO se muestra en la Figura III.4.1.

La estructura interna del PIO consiste en:

- Buffer del Bus de Datos. Este buffer es triestado, de 8 bits y bidireccional, se usa para interconectar el M8255A al sistema del bus de datos. Los datos son transmitidos o recibidos por el buffer durante la ejecución de instrucciones de entrada o salida por el microprocesador. Las palabras de control y la información de estado, también se transfieren a través de éste buffer.
- Lectura/Escritura y Lógica de Control. La función de este bloque es manejar todas las transferencias internas y externas tanto de datos y control como de palabras de estado. Acepta entradas de direcciones de la CPU y de los buses de Control y en su momento, permite comandos a ambos Grupos de Control (A y B).

III.5 Direcionamiento y control del PIO.

Controles del Grupo A y del Grupo B.

La configuración funcional de cada puerto se progrma por el software del sistema. De hecho, las salidas de la CPU envían una palabra de control hacia el M8255A. La palabra de control contiene información tal como "mode", "bit set", "bit reset",etc. que inicializa la configuración funcional del M8255A.

Cada uno de los bloques de control (Grupo A y Grupo B) aceptan "comandos" de la lógica de control de Lectura/Escritura, recibe "palabras de control" del bus interno de datos y canaliza al comando apropiado a su puerto asociado.

El Grupo de Control A comprende al puerto A y la parte superior del puerto C (C7-C4).

El grupo de Control B comprende al puerto B y a la parte baja del puerto C (C3-C0).

III.5.1 Puertos A,B y C.

El M8255A tiene tres puertos de 8 bits (A,B y C). Todos ellos pueden configurarse en una gran variedad de caracteristicas funcionales por medio del software del sistema, pero cada una de ellas tiene su propia caracteristica o "personalidad" que hace poderoso y flexible al M8255A.

Puerto A. Una salida de datos de 8 bits del tipo cerrojo/buffer y una entrada de datos de 8 bits del tipo cerrojo.

Puerto B. Una entrada/salida de datos de 8 bits del tipo cerrojo/buffer y una entrada de datos de 8 bits del tipo buffer.

Puerto C. Una salida de datos de 8 bits del tipo cerrojo/buffer y una entrada de datos de 8 bits del tipo buffer (sin cerrojo para entrada). Este puerto puede ser dividido en puertos de 4 bits bajo el control de "mode". Cada puerto de 4 bits tiene un cerrojo de 4 bits y puede ser usado para las salidas de las señales de control y las señales de entrada de estado en conjunción con los puertos A y B.

III.6 Modos de operación del PIO.

Selección de modos de operación.

Existen tres modos básicos de operación que pueden ser seleccionados por el software del sistema:

Modo 0 - Entrada/Salida básica.
Modo 1 - Entrada/Salida "Strobed".
Modo 2 - Bus bidireccional.

Modo 0. Esta configuración funcional tiene una operación de entrada y salida simple para cada uno de los tres puertos. No se requiere "handshaking", los datos simplemente se leen o se escriben desde un puerto especifico. Sus definiciones funcionales son:

- Dos puertos de 8 bits y dos puertos de 4 bits.
- Cualquier puerto puede ser entrada o salida.
- Sus salidas son del tipo cerrojo.
- Sus entradas no son del tipo cerrojo.
- 16 combinaciones diferentes son posibles en este modo.

Modo 1. Esta configuración funcional provee un medio para transferir datos de Entrada/Salida hacia o desde un puerto especifico en conjunción con señales de "handshaking" o strobes. En el modo 1 el puerto A y el puerto B usan las lineas del puerto C para generar o aceptar esas señales de "handshaking". Sus definiciones funcionales son:

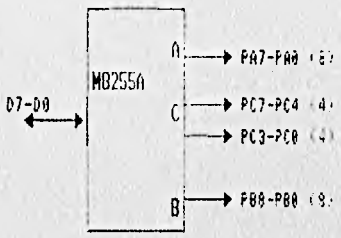
- Dos grupos (Grupo A y Grupo B).
- Cada grupo contiene un puerto de datos de 8 bits y un puerto de 4 bits de control/datos.
- Los puertos de datos de 8 bits pueden ser entradas o salidas. Ambas posibilidades son del tipo cerrojo.
- El puerto de 4 bits se usa para controlar y ver el estado del puerto de datos de 8 bits.

Modo 2. Esta configuración operacional ofrece un medio para la comunicación con dispositivos periféricos o dispositivos configurados en un bus simple de 8 bits tanto para tranamitir o recibir datos (bus bidireccional). Las señales de "Handshaking" se usan para el propio mantenimiento del bus, de una manera muy similar al modo 1. Se dispone también de señales de interrupción y funciones de habilitación/deshabilitación.

Las definiciones funcionales del modo 2 son:

- Se usan en el grupo A solamente.
- Un puerto con bus birireccional de 8 bits (puerto A) y un puerto de control de 5 bits (puerto C)
- Tanto las entradas como las salidas son del tipo cerrojo.
- El puerto de control de 5 bits (purto C) para control y para ver el estado del puerto del bus bidireccional de 8 bits (puerto A).

En la siguiente figura se ilustra un resumen de las posibles definiciones del M8255A.



PALABRA DE CONTROL # 80

D7	D6	D5	D4	D3	D2	D1	D0
1	0	0	0	0	0	0	0

PALABRA DE CONTROL # 81

D7	D6	D5	D4	D3	D2	D1	D0
1	0	0	0	0	0	0	1

PALABRA DE CONTROL # 82

D7	D6	D5	D4	D3	D2	D1	D0
1	0	0	0	0	0	1	0

PALABRA DE CONTROL # 83

D7	D6	D5	D4	D3	D2	D1	D0
1	0	0	0	0	0	1	1

PALABRA DE CONTROL # 88

D7	D6	D5	D4	D3	D2	D1	D0
1	0	0	0	1	0	0	0

PALABRA DE CONTROL # 89

D7	D6	D5	D4	D3	D2	D1	D0
1	0	0	0	1	0	0	1

PALABRA DE CONTROL # 8A

D7	D6	D5	D4	D3	D2	D1	D0
1	0	0	0	1	0	1	0

PALABRA DE CONTROL # 8B

D7	D6	D5	D4	D3	D2	D1	D0
1	0	0	0	1	0	1	1

PALABRA DE CONTROL # 90

D7	D6	D5	D4	D3	D2	D1	D0
1	0	0	1	0	0	0	0

PALABRA DE CONTROL # 91

D7	D6	D5	D4	D3	D2	D1	D0
1	0	0	1	0	0	0	1

PALABRA DE CONTROL # 92

D7	D6	D5	D4	D3	D2	D1	D0
1	0	0	1	0	0	1	0

PALABRA DE CONTROL # 93

D7	D6	D5	D4	D3	D2	D1	D0
1	0	0	1	0	0	1	1

Fig. III.6.1 Programacion de los puertos E/S (8255).

CAPITULO IV

PRUEBA Y CARACTERIZACION DEL SISTEMA DE DESARROLLO.

IV.1 Circuito generador de pulsos de reloj.

Cualquier oscilador que cumpla con las especificaciones de la entrada *h* del Z80 puede servir para generar la señal de reloj. En este caso, la frecuencia de oscilacion del cristal determina la frecuencia de la señal de reloj, y gracias a la estabilidad del cristal se puede mantener un tiempo de ejecución constante. La figura IV.1 muestra el diagrama para este oscilador.

IV.2 Circuito de reinicialización (Rutina Reset).

Cuando la fuente de alimentación del sistema se enciende, el capacitor se encuentra descargado. El nivel de 0 lógico en la entrada RESET debe mantenerse durante el tiempo suficiente para que el nivel de voltaje de la fuente de alimentación se estabilice dentro del intervalo valido para Vcc. Este tiempo se obtiene de la contante de carga del capacitor RC, determinada por los valores de R y C.
Una vez que el programa está en operación se puede detener la ejecución del programa y volverla a empezar a partir de la dirección 0, oprimiendo momentáneamente el interruptor del circuito de restablecimiento. Al presionar el interruptor, el capacitor queda en cortocircuito descargándose rápidamente y provocando que la entrada RESET baje a 0 lógico. El microprocesador permanece en estado de "reset" mientras se mantiene oprimido el interruptor. Al soltarlo, el capacitor se carga a través de la resistencia y la entrada RESET vuelve a 1 lógico con lo cual, el microprocesador reinicia la ejecución.
Los inversores con gatillo Schmitt del circuito de la figura IV.2 aumentan la confiabilidad del diseño al introducir histéresis y acelerar la transición entre niveles lógicos; además permiten la obtención de una señal de restablecimiento (reset) activa a 1 lógico y otra en 0 lógico. El uso del diodo tiene por objeto descargar rápidamente el capacitor, para asegurar la generación del pulso de restablecimiento (reset) aún en los casos en que se apague y se vuelva a encender el sistema en un lapso muy corto.

IV.3 Interconexión de los elementos que integran el sistema de desarrollo.

En las figuras IV.3 (a) y IV.3 (b) se muestran los diagramas que complementan el sistema de desarrollo, para su ensamblado total.

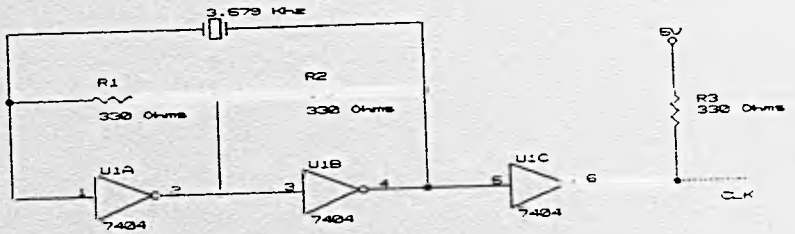


Fig. IV.1 Circuito generador de pulsos de reloj

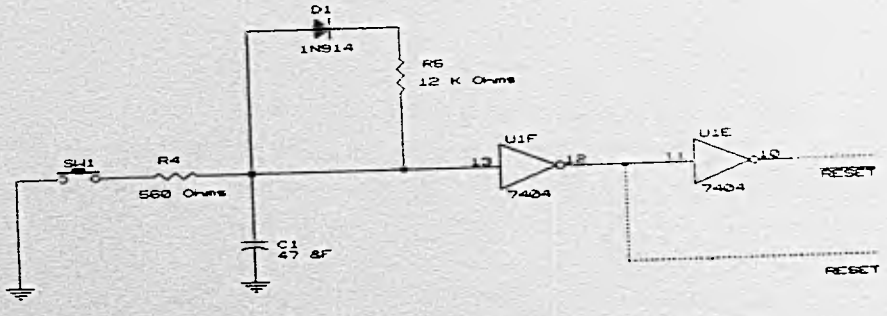


Fig. IV.2 Circuito de reinicialización (Rutina Reset)

Fig. IV.3 (a) Interconexion de los elemtos que integran el sistema de desarrollo

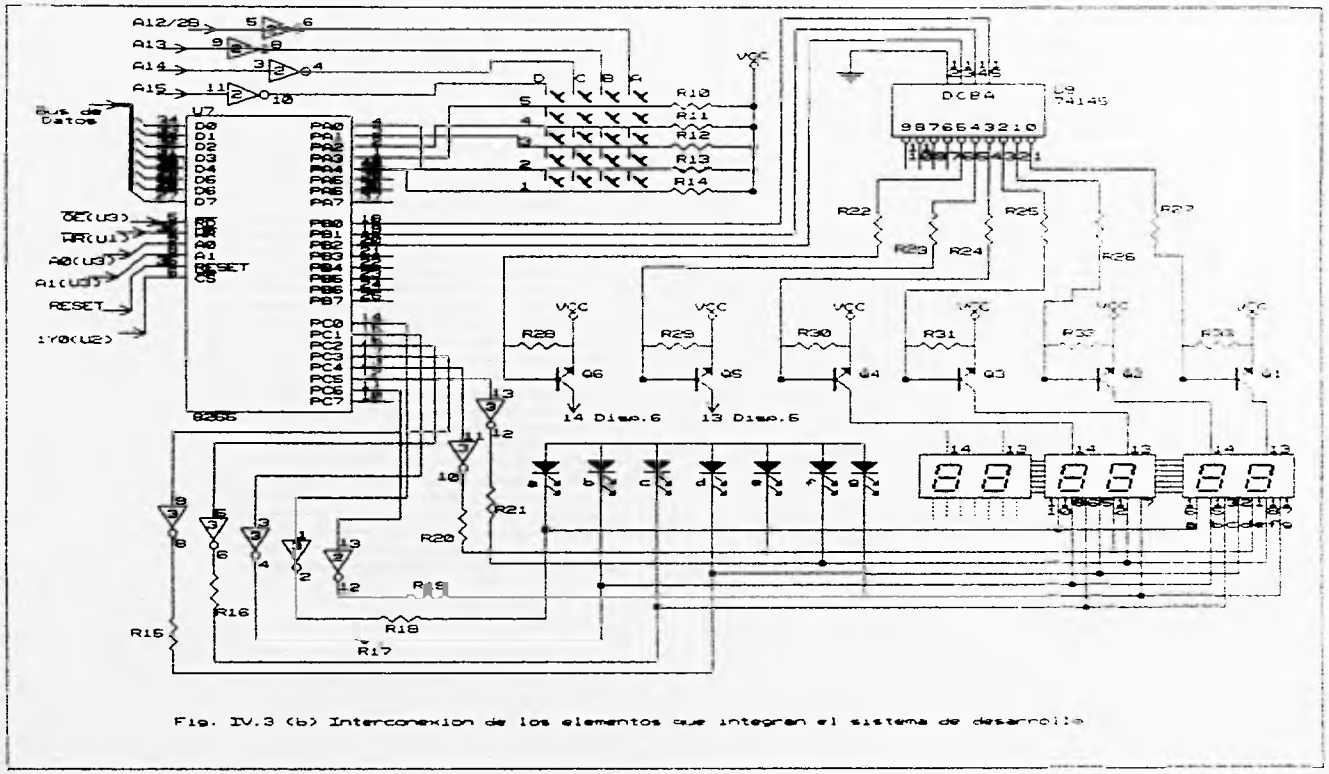
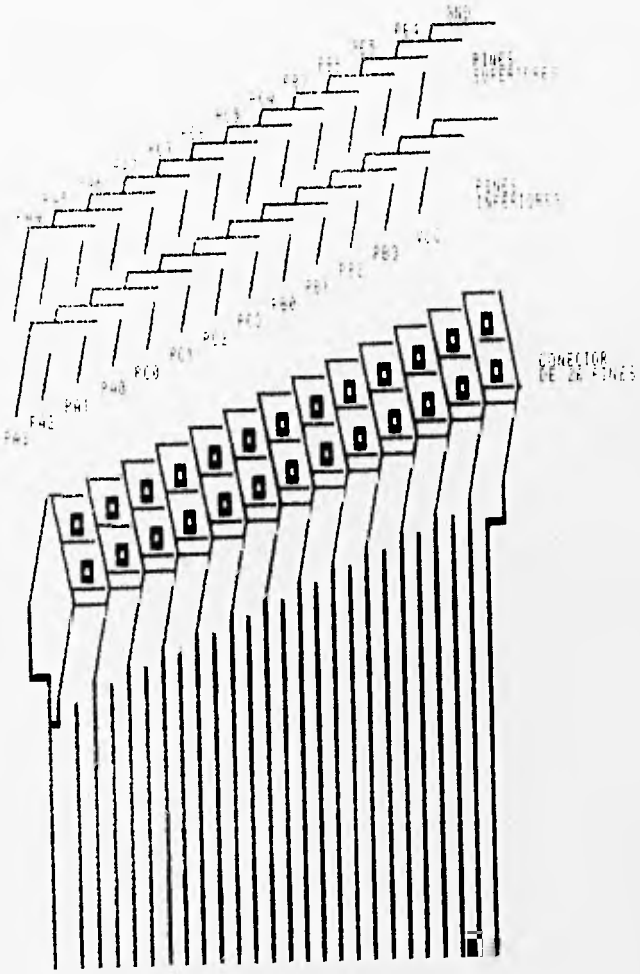


Fig. IV.3 (b) Interconexion de los elementos que integran el sistema de desarrollo

DISTRIBUCION DE LOS PINES EN EL PUERTO DEL USUARIO.



CABLE PLANO HACIA EL PROGRAMADOR DE MEMORIAS.

APENDICE A

DOCUMENTACION DEL SOFTWARE UTILIZADO EN ESTE PROYECTO

```
FAM:    EQU 17FDH
DATDES: EQU 17FBH
DIRDES: EQU 17F7H
AA:     EQU 17F6H
LLL:    EQU 17DDH
IXL:    EQU 17EBH
IYL:    EQU 17E9H
SPL:    EQU 17E7H
PCL:    EQU 17E5H
BUFF1:  EQU 17DCH
BUFF2:  EQU 17DBH
BUFF3:  EQU 17DAH
BUFF4:  EQU 17D9H
BUFF5:  EQU 17D8H
BUFF6:  EQU 17D7H
BUFF7:  EQU 17D6H
BUFF8:  EQU 17D5H
BUFF9:  EQU 17D4H
BUF10:  EQU 17D3H
SAI:    EQU 17D0H
REST10: EQU 17CDH
REST18: EQU 17CAH
REST20: EQU 17C7H
REST28: EQU 17C4H
REST30: EQU 17C1H
REST38: EQU 17BEH
        ORG 0000H
INICIO: JP  INICIO
INICI2: LD  SP,17BEH
        JR  HOLA
RST08:  LD  (BUFF1),A
        JR  ENTRA
        NOP
        NOP
        NOP
PRST10: JP  REST10
        NOP
        NOP
        NOP
        NOP
        NOP
PRST18: JP  REST18
        NOP
        NOP
        NOP
        NOP
        NOP
PRST20: JP  REST20
```

```
        NOP
        NOP
        NOP
        NOP
        NOP
PRST28: JP  REST28
        NOP
        NOP
        NOP
        NOP
        NOP
         PRST30: JP  REST30
        NOP
        NOP
        NOP
        NOP
        NOP
PRST38: JP  REST38
ENTRA:  LD  (BUFF3),BC
        POP BC
        LD  (PCL),BC
        LD  BC,(BUFF3)
        LD  A,0FFH
        LD  (BUFF3),A
        LD  A,(BUFF1)
ENTRA1: LD  (SPL),SP
        LD  SP,DIRDES
        PUSH AF
        PUSH BC
        PUSH DE
        PUSH HL
        LD  A,I
        LD  B,A
        LD  A,R
        LD  C,A
        PUSH BC
        PUSH IX
        JR  ENTRA2
NMI:    JP  ASI
ENTRA2: PUSH IY
        DEC SP
        DEC SP
        DEC SP
        DEC SP
        EX  AF,AF
        EXX
        PUSH AF
        PUSH BC
        PUSH DE
        PUSH HL
        LD  SP,(SPL)
HOLA:   CALL LIMPIA     ;DESPLEGA
        LD  HL,TAHOLA   ;HOLA
        CALL DESP1
COMO:   CALL LLAC1      ;RUTINA DE
```

```
        CALL Z,ERROR     ;RECONOCIMIENTO
        JR  Z,COM0       ;DE COMANDOS
        LD  HL,TCOM      ;CHECAR SI ES 1 O 2
        LD  A,(BUFF2)
        ADD A,L
        LD  L,A
        JP  (HL)
INICIO: LD BC,0020H      ;INICIALIZACION
INICI1: DJNZ INICI1
        DEC C
        JR  NZ,INICI1
        DI
        LD  A,90H
        OUT (03),A
        JP  INICI2
COM1:   CALL LIMPIA      ;COMANDOS 1
        LD  HL,TACOM1
        CALL DESP1
        CALL LLAC
        LD  A,(BUFF2)
        CP  03
        JR  C,COM11
CERR:   CALL ERROR
        LD  SP,17BEH     ;INICIALIZA SP
        JR  COM0
COM11:  CP  00
        JR  Z,MEM
        CP  Z,REG
ES:     LD  HL,TAPE      ;COMANDO
        CALL DESP1       ;E/S
        CALL DAT
        CALL LLAC
ALD:    LD  A,(BUFF2)
        LD  B,C
        LD  C,A
        CALL CORRE2
        CALL CONVIA
        LD  (DIRDES+3),A
        CALL LLAC1
        JR  Z,ALD
        CALL LLAC2
AIN:    CALL ACTPUE
        IN  A,(C)
        CALL MBFF4
DESPDP: CALL DESP
        CALL DATUS
        LD  A,(BUFF6)
        CP  00
        JR  NZ,PS
        INC C
        JR AIN
PS:     LD  AL,TAPS       ;PUERTO DE SALIDA
        CALL DESP1
        CALL ACTPUE
        LD  A,(BUFF7)
```

```
        OUT (C),A
        CALL LLAC1
        JR  NZ,PS1
PS01:   JR  CERR
PS1:    LD  A,(BUFF2)
        CP  10H
        JR  Z,CINC
        CP  08H
        JR  NZ,PS01
        JP  HOLA
CINC:   INC C
        LD  HL,TAPE
        CALL DESP1
        JR  AIN
TCON:   JP  GO              ;TABLA DE
        NOP                 ;COMANDOS
        JP  COM1
        NOP
        JP  COM2
        NOP
        JR  PS01
MEM:    LD  HL,TADIR        ;COMANDO
        CALL DESP1          ;MEM
        CALL LLAC1
        CALL BFF90
AQUI:   CALL ARLD
        CALL BFF98
        CALL LLAC1
        JR AQUI
GATO:   LD  A,E             ;RETORNO DE ARLD
        CP  4AH             ;(TABLA)
        JR  Z,GATO2
        CP  0B2H
        JR  Z,GO1
        CP  8AH
        JP  Z, CDR2
        CP  0F7H
        JP  Z,R4CALL        ;RETORNA
        CP  00
        JP  Z,HOLA
        CP  00
        JP  Z,HOLA
        CP  00
        JP  Z,HOLA
        CP  00
        JP  Z,HOLA
        CP  00
        JP  Z,HOLA
        JP  PAM
GATO2:  LD  HL,(BUFF9)
GATO1:  CALL ACTDIR
        LD  A,(HL)
        CALL MBFF4
        CALL DATOS
        LD  A,(BUFF6)
```

```
        CP  00
        JR  NZ,MODMEN
        INC HL
        JR  GATO1
MODMEN: LD A,(BUFF7)
        LD  (HL),A
        INC HL
        JR GATO1
GO:     LD  HL,TAGO         ;COMANDO
        CALL DESP1          ;GO
        CALL LLAC1
        CP  10H
        JR  Z,GO2
        CALL BFF90
ASI:    CALL ARLD
        CALL BFF98
        CALL LLAC1
        JR  ASI
GO1:    LD  HL,(BUFF9)
        LD  (PCL),HL
GO2:    LD  HL,(PCL)

        LD  (BUFF9),HL
        LD  A,0C3H
        LD  (BUF10),A
        LD  (SPL),SP
        LD  SP,LLL          ;BUFFER DE L
        POP HL
        POP DE
        POP BC
        POP AF
        EXX
        EX  AF,AF'
        LD  SP,IYL
        POP IY
        POP IX
        POP BC
        LD  A,C
        LD  R,A
        LD  A,B
        LD  I,A
        POP HL
        POP DE
        POP BC
        LD  A,05H
        OUT (01),A
        LD  A,79H
        OUT (02),A
        POP AF
        LD  SP,(SPL)
        JP  BUF10
REG:    LD  HL,TAREG        ;COMANDO
        CALL DESP1          ;REG
        CALL LLAC
```

```
MODR:   LD  A,(BUFF2)
        CP  03
        JP  C,CERR
        LD  DE,TAMREG
        LD  HL,AA
        AND 0FH
        SUB 03
        SLA A
        LD  C,A
        LD  B,A
        LD  A,L
        SUB B
        LD  L,A

     SLA C
        LD  A,E
        SUB C
        LD  E,A
        CALL ACTREG
        CALL LLAC1
        CP  10H
        JR  NZ,MODR
VALOR:  LD  A,(HL)
        CALL MBFF4
        CALL DESP
        PUSH DE
        POP BC
        CALL DATOS
        PUSH BC
        POP DE
        LD  A,(BUFF6)
        CP  00
        JR  NZ,MODCON
VALOR1: DEC HL
        DEC DE
        LD  A,L
        CP  0DCH
        JR  NZ,DEDEC
        LD  HL,AA
        LD,DE,TAMREG
        CALL ACTREG
        JR  VALOR
DEDEC:  CALL ACTREG
        JR  VALOR
MODCOM: LD  A,(BUFF7)
        LD  (HL),A
        JR  VALOR1
COM2:   CALL LIMPIA          ;COMANDOS 2
        LD  HL,TACOM2
        CALL DESP1
        CALL LLAC
        LD  HL,TACOM2
        LD  A,(BUFF2)
        SLA A
        ADD A,L
```

```
        LD  L,A
        LD  B,(HL)
        INC HL
        LD  C,(HL)
        LD  (BUFF9),BC
        LD  A,0C3H
        LD  (BUF10),A
        JP  BUF10
CDR:    LD  HL,TACDR        ;CALCULO DE
        CALL C3CAL1         ;DESPLAZAMIENTO
CDR1:   CALL ALRDL          ;RELATIVO
        CALL BFF98
        LD  A,40H
        LD  (DIRDES),A
        CALL LLAC1
        JR  CDR1
CDR2:   LD  HL,BUFF9
        LD  A,(HL)
        DEC AL
        LD  (HL),A
        RRCA
        RRCA
        RRCA
        RRCA
        INC HL
        LD  (HL),A
        LD  D,03
        LD  C,00
        INC HL
CDR02:  LD  A,(HL)
        AND 0FH
        CP  0AH
        JP  NC,CERR
        DEC HL
        DEC D
        JR  NZ,CDR02
        LD  D,03
        LD  HL,BUFF8
        LD  A,(HL)
        AND 0FH
        LD  B,A
        LD  A,C
        RLCA
        RLCA
        ADD A,C
        RLCA
        ADD A,B
        LD  C,A
        DEC HL
        DEC D
        JR  NZ,CDR01
        CPL
        INC A
        LD  B,A
        LD  A,C
```

```
        LD  C,B
        CALL LIMPIA
        CALL MBFF4
        CALL ACTPUE
        LD  A,40H
        LD  (DIRDES+1),A
        JP  COMO
MBM:    CALL LIMPIA         ;MUEVE BLOQUES
        LD  HL,TAMBM        ;DE MEMORIA
        CALL CMCALL
        LDIR
        JP  HOLA
CM:     CALL LIMPIA         ;COMPARA
        LD  HL,TACM         ;MEMORIA
        CALL CMCALL
VERIFM: LD  A,(DE)
        CPI
        DEC HL
        CALL NZ,PINTA
        EX  DE,HL
        CALL NZ,PINTA
        EX  DE,HL
        INC HL
        INC DE
        JP  PO,HOLA
        JR  VERIFM
DAM:    CALL LIMPIA         ;DESPLEGADO
        LD  HL,TADAM        ;AUTOMATICO
        CALL DESP1          ;DE MEMORIA
        CALL CMCAL1
        INC C
        INC B
        EX  DE,HL
        DEC HL
DAM1:   INC HL
        CALL PINTA
        DEC C
        JR  NZ,DAM1
        DEC B
        JR  NZ,DAM1
        JP  HOLA
DAR:    CALL LIMPIA         ;DESPLEGADO
        LD  HL,TAREG        ;AUTOMATICO
        CALL DESP1          ;DE REGISTROS
DAR2:   LD  DE,TAMREG
        LD  HL,AA
DAR1:   CALL ACTREG
        LD  A,(HL)
        CALL MBFF4
        LD  BC,00F01H       ;TIEMPO
CDEST:  JR  DESPT
CDEST1: DJNZ CDEST
        DEC C
        JR  NZ,CDEST
        JR  SAS
```

```
        DESPT:  CALL DESP
        LD   A,(BUFF1)
        CP   00H            ;TECLA ?
        JR   Z,CDEST1
        CALL LLAC1
SAS:    DEC HL
        DEC DE
        LD   A,L
        CP   0DCH           ;TERMINO ?
        JR   NZ,DAR1
        JR   DAR2

          ;SUBRUTINAS DE UTILERIA

        ORG 497H
        ;PINTA: DESPLEGA EL VALOR DE H Y L ASI COMO SU
        ;CONTENIDO DURANTE POCO MENOS DE UN SEGUNDO, SI
        ;DURANTE ESTE TIEMPO ES PULSADA CUALQUIER TECLA
        ;(COMANDOS O DATOS) EL DESPLEGADO SE MANTIENE
        ;MOSTRANDO EL ACTUAL CONTENIDO HASTA QUE OTRA TECLA
        ;SE OPRIMA.
        ;ENTRADA: CONTENIDO DE HL
        ;SALIDA: DESPLEGADO
        ;MODIFICA:REGISTROS ALTERNOS
        ;LLAMA: ACTDIR,MBFF4,DESP,LLAC1

PINTA:  PUSH AF
        PUSH BC
        PUSH DE
        PUSH HL
        CALL ACTDIR
        LD   A,(HL)
        CALL MBFF4
        LD   BC,0FF01H
CDESP:  JR   DESPE
CDESP1: DJNZ CDESP
        DEC  C
        JR   NZ,CDESP
PPPOP:  POP  HL
        POP  DE
        POP  BC
        POP  AF
        RET
DESPE:  CALL DESP
        LD   A,(BUFF1)
        CP   00
        JR   Z,CDESP1
        CALL LLAC1
        JR   PPPOP

       ;CMCALL:  SOLICITA 3 DATOS DE 16 BITS (FUENTE,
       ;CANTIDAD Y DESTINO), COLOCANDOLOS EN HL,BC Y DE
       ;RESPECTIVAMENTE.
       ;ENTRADA: TECALDO
       ;SALIDA: HL, BC Y DE
```

```
        ;MODIFICA: A,F,BC,DE,HL Y REGISTROS ALTERNOS
        ;LLAMA: DESP1,C4CALL,LIMPIA

CMCALL: LD   A,0F1H
        LD   (DATDES+1),A
        CALL DESP1
        CALL C4CALL       ;SOLICITA
        LD   HL,(BUFF9)   ;FUENTE
CMCAL1: PUSH HL
        CALL LIMPIA
        LD   A,39H
        LD   (DATDES+1),A
        CALL C4CALL       ;SOLICITA
        LD   BC,(BUFF9)   ;CANTIDAD
        PUSH BC
        CALL LIMPIA
        LD   A,39H
        LD   (DATDES+1),A
        CALL C4CALL       ;SOLICITA
        LD   DE,(BUFF9)   ;DESTINO
        POP  BC
        POP  HL
        RET

        ;C4CALL: DESPLEGA, SI LA TECLA PULSADA ES COMANDO DA
        ;ERROR, SI ES DATO LO COLOCA COMO EL NIBLE MENOS
        ;SIGNIFICATIVO DEL CAMPO DE DIRECCIONES; SI SE
        ;TECLEAN MAS DATOS, LOS VA RECORRIENDO A LA IZQUIERDA
        ;DEJANDO AL NUEVO A LA DERECHA, ;PERO SI LA TECLA ES
        ;"OTRO" REGRESA CON LOS DATOS QUE ESTAN DESPLEGADOS
        ;EN LOS BUFFERS 9 Y 8.
        ;ENTRADA: TECLADO
        ;SALIDA: DESPLEGADO DE BUFFERS 9 Y 8
        ;MODIFICA: A,F,C,DE,HL
        ;LLAMA: C3CAL1,ARLD,BFF98,LLAC1

C4CALL: CALL C3CAL1
C4CAL1: CALL ARLD
        CALL BFF98
        CALL LLAC1
        JR   C4CAL1
R4CALL: RET

        ;DESP: DESPLEGA LO QUE SE ENCUENTRA EN EL AREA DE
        ;MEMORIA DEL DESPLEGADO PSEUDO-ALFANUMERICO (CAMPOS
        ;DE DATOS Y DIRECCIONES).
        ;ENTRADA: DATOS EN EL AREA DE MEMORIA PARA EL
        ;DESPLEGADO.
        ;SALIDA: DESPLEGADO Y VALOR DE LA TECLA PULSADA EN EL
        ;BUFFER2. BUFFER 1=00, SI NO HUBO TECLA. BIT 4 DEL
        ;BUFFER 1=1; SI LA TECLA FUE UN COMANDO Y BUFFER 1
        ;DIFERENTE DE 00 SI FUE UNA TECLA DE DATOS.
        ;MODIFICA: A,F
        ;LLAMA: TECLAS
```

```
DESP:   PUSH HL
        LD   HL,DIRDES
DESPO:  PUSH BC
        PUSH DE
        LD   BC,0701H
        LD   DE,2005H
SIGUE:  LD   A,(HL)
        OUT  (02),A
        OUT  (C),E
        CALL TECLAS
        LD   A,(BUFF1)
        CP   00
        JR   NZ,EPOP
        INC  HL
        DEC  E
        LD   A,D
        RRCA
        LD   D,A
        OUT  (C),B
        JR   NC,SIGUE
EPOP:   POP  DE
        POP  BC
        POP  HL
        RET

        ;TECLAS: SENSA EL TECLADO Y EN CASO DE REGISTRAR UN
        ;DATO, LO CONVIERTE EN SU CORRESPONDIENTE VALOR
        ;BINARIO, GUARDA EL DATO EN EL BUFF2, COLOCA EN EL
        ;BUFF1 UN VALOR DIFERENTE DE 00 Y REGRESA A QUIEN LE
        ;LLAMO, SI NO SE PULSA NINGUNA TECLA EN ESTE TIEMPO
        ;EL BUFF1 REGRESA CON 00.
        ;ENTRADA: TECLADO
        ;SALIDA: BUFF1 Y BUUF2
        ;MODIFICA: A,F
        ;LLAMA: NINGUNA

TECLAS: PUSH HL
        PUSH BC
        PUSH DE
        LD   BC,1000H
        LD   HL,BUFF1
        XOR  A
        LD   (HL),A
        DEC  HL
METE:   CALL METE0
        JR   Z,ROTA
        LD   D,A
        LD   A,0FFH
ADEC:   DEC  A
        JR   NZ,ADEC
        CALL METE0
        JR   Z,ROTA
METE2:  CALL METE0
        JR   NZ,METE2N
        LD   A,D
```

```
SALE:   LD   (BUFF1),A
SACA:   LD   D,00
SACAS:  RRCA
        JR   C,VALE
        INC  D
        JR SACAS
VALE:   LD   A,B
        RRCA
        RRCA
        RRCA
        RRCA
DALE:   RRCA
        LD   (HL),D
        JR   C,VUELVE
        LD   E,A
        LD   A,0H
        ADD  A,D
        LD   D,A
        LD   A,E
        JR   DALE
ROTA:   RLC  B
        JR   NC,METE
VUELVE: POP  DE
        POP  BC
        POP  HL
        RET

        ;METE0:  LEE EL CONTENIDO DEL PUERTO DIRECCIONADO POR
        ;EL REGISTRO C, LO COMPLEMENTA Y DEJA EN A SOLO LOS 5
        ;BYTES MENOS SIGNIFICATIVOS.
        ;ENTRADA: C CON LA DIRECCION DEL PUERTO
        ;SALIDA: A CON LOS BYTES MENOS SIGNIFICATIVOS
        ;MODIFICA: A,F
        ;LLAMA: NINGUNA

METE0:  IN   A,(C)
        CPL
        AND  1FH
        RET

        ;DESP1:  RECIBE  EN  HL  LA DIRECCION DEL  MENSAJE  A
        ;DESPLEGAR  SOLAMENTE EN EL CAMPO DE DIRECCIONES  DEL
        ;DESPLEGADO Y LO DESPLEGA.
        ;ENTRADA: HL APUNTANDO A LA TABLA DEL MENSAJE
        ;SALIDA:  AREA  DE MEMORIA DEL CAMPO DE DIRECCIONES Y
        ;DESPLEGADO PSEUDO-ALFANUMERICO
        ;MODIFICA: A,F,HL
        ;LLAMA:DESP

DESP1:  PUSH DE
        PUSH BC
        LD   DE,DIRDES
        LD   BC,0004H
        LD DIR
        CALL DESP
```

```
        POP   BC
        POP   DE
        RET

        ;ERROR: DESPLEGA MENSAJE DE ERROE Y REGRESA CON A=00.
        ;ENTRADA: NINGUNA
        ;SALIDA: DESPLEGADO Y A=00
        ;MODIFICA: A,AL Y REGISTROS ALTERNOS
        ;LLAMA: LIMPIA

ERROR:  CALL  LIMPIA
        LD    HL,TAERR
        CALL  DESP1
        XOR   A
        RET

        ;LIMPIA: LIMPIA EL AREA DE MEMORIA DEL DESPLEGADO
        ;ENTRADA: NINGUNA
        ;SALIDA: AREA DE MEMORIA DEL DESPLEGADO
        ;MODIFICA: REGISTROS ALTERNOS (HL',DE')
        ;LLAMA: NINGUNA

LIMPIA: EXX
        LD    DE,0600H
        LD    HL,DIRDES
MAS:    LD    (HL),E
        INC   HL
        DEC   D
        JR    NZ,MAS
        EXX
        RET

        ;CORRE3: RECORRE A LA IZQUIERDA UNA VEZ LOS DATOS DEL
        ;CAMPO DE DIRECCIONES EN EL AREA DE MEMORIA DEL
        ;DESPLEGADO.
        ;ENTRADA: NINGUNA
        ;SALIDA: AREA DE MEMORIA DEL DESPLEGADO
        ;MODIFICA: REGISTROS ALTERNOS (BC',DE',HL')
        ;LLAMA: NINGUNA

CORRE3: EXX
        LD    HL, DATDES+1
        LD    B,(HL)
        DEC   HL
        LD    (HL),B
        EXX
        RET

        ;CONVI:  CONVIERTE EL DATO DEL ACUMULADOR DE  BINARIO
        ;AL  CODIGO  DE  7  SEGMENTOS  (C7S)  PARA  PODER  SER
        ;DESPLEGADO. COLOCA EL NIBLE MENOS SIGNIFICATIVO Y EL
        ;NIBBLE MAS SIGNIFICATIVO EN EL BUFFER 4.
        ;ENTRADA: DATO EN A
        ;SALIDA: BUFFERS 3 Y 4
        ;MODIFICA: A,F Y REGISTROS ALTERNOS
```

```
        POP   BC
        CP    08H
        JP    Z,HOLA
        JP    CERR
DATO2:  XOR   A
        LD    (BUFF6),A
        POP   BC
        RET
        DB    00H,00H,00H,00H
DATO1:  LD    A,40H          ;GUION
        LD    (DATDES),A     ;LA PRIMERA TECLA
        CALL  ALDB2          ;ES UN DATO
        LD    (DATDES+1),A
        XOR   A
        LD    (BUFF5),A
        CALL  LAAC
LADA:   LD    A,(BUFF2)      ;LA SEGUNDA TECLA
        LD    D,C
        LD    C,A
        CALL  CORRE1
        CALL  CONVIA
        LD    (DATDES+1),A
        CALL  LLAC1
        JR    Z,LADA
        CALL  LLAC2
        LD    (BUFF7),A
        POP   BC
        RET

        ;ALDB2: TOMA EL ULTIMO DATO TECLEADO, LO COLOCA EN C,
        ;LO   CONVIERTE  AL  C7S  Y  DEJA  EL  NIBBLE   MENOS
        ;SIGNIFICATIVO (C7S) EN A.
        ;ENTRADA:BUFFER 2
        ;SALIDA: A,C
        ;MODIFICA: A,C Y REGISTROS ALTERNOS
        ;LLAMA: CONVIA

ALDB2:  LD    A,(BUFF2)
        LD    C,A
        CALL  CONVIA
        RET

        ;LLAC:  DESPLEGA  LO  QUE  ENCUENTRE EN  EL  AREA  DE
        ;MEMORIA DEL DESPLEGADO,ESPERA UNA TECLA,  EN CASO DE
        ;SER UN COMANDO DESPLEGA ERROR Y BRINCA A LA TABLA DE
        ;RECONOCIMIENTO DE COMANDOS, DE OTRA FORMA REGRESA.

LLAC:   CALL  DESP
        CALL  ABIT4
        CP    00H
        JR    Z,LLAC
        RET
        DB    00H,00H,00H,00H

        ;LLAC1:  DESPLEGA  LO  QUE  ENCUENTRE EN EL  AREA  DE
```

```
        EXX
        RET

;CONVIA: RECIBE EN EL BUFFER 2 EL DATO A SER
;CONVERTIDO A CODIGO DE 7 SEGMENTOS (C7S) Y DESPUES
;DE CONVERTIRLO DEJA EL NIBBLE MENOS SIGNIFICATIVO EN A.
;ENTRADA: DATO EN A
;SALIDA: NIBBLE MENOS SIGNIFICATIVO EN A.
;MODIFICA: A,F Y REGISTROS ALTERNOS.
  ;LLAMA: CONVI

CONVIA: LD   A,(BUFF2)
        CALL CONVI
        LD   A,(BUFF3)
        RET

        ;ACTPUE: CONVIERTE EL CONTENIDO DE A AL C7S Y LO
        ;COLOCA EN EL BYTE MENOS SIGNIFICATIVO DEL CAMPO DE
        ;DIRECCIONES EN EL AREA DE MEMORIA DEL DESPLEGADO.
        ;ENTRADA: DATO EN A
        ;SALIDA:  BYTE MENOS SIGNIFICATIVO DEL CAMPO DE
        ;DIRECCIONES EN EL AREA DE MEMORIA DEL DESPLEGADO.
        ;MODIFICA: A,F,C Y REGISTROS ALTERNOS
        ;LLAMA: CONVI

ACTPUE: PUSH DE
        LD   A,C
        CALL CONVI
        LD   DE,(BUFF4)
        LD   (DIRDES+2),DE
        CALL DESP
        POP  DE
        RET

        ;DATOS: ESPERA HASTA QUE 3 TECLAS SEAN PULSADAS
        ;(MINIMO), LAS DOS PRIMERAS DEBEN SER DATOS Y LA
        ;TERCERA "OTRO". SI "SALE" O "COM2" SON OPRIMIDAS
        ;AQUI, SE DESPLEGA ERROR, PERO SI ES "COM1"
        ;DESPLEGARA HOLA Y EN AMBOS CASOS (SALE,COM2 O COM1)
        ;BRINCA A LA RUTINA DE RECONOCIMIENTO DE COMANDOS.
        ;ENTRADA: TECLADO
        ;SALIDA:  DESPLEGADO (MODIFICANDO CAMPO DE DATOS) Y
        ;BUFFER 7
        ;MODIFICA. A,F,BC,D Y REGISTROS ALTERNOS
        ;LLAMA: LLAC1,ALDB2,LLAC,CORRE1,CONVIA,LLAC2

DATOS:  PUSH BC
        LD   A,0FFH
        LD   (/BUFF6),A
DATO:   CALL LLAC1
        JR   Z,DATO1
        LD   A,(BUFF2)
        CP   10H
        JR   Z,DATO2
        POP  BC
```

```
DAT:    CALL LLAC
DAT1:   CALL ALDB2
        LD   (DIRDES+3),A
        RET

        ;ACTDIR: CONVIERTE EL VALOR DE HL A C7S Y LO COLOCA
        ;EN EL CAMPO DE DIRECCIONES DEL AREA DE MEMORIA PARA
        ;EL DESPLEGADO.
        ;ENTRADA: VALOR DE HL
        ;SALIDA: CAMPO DE DIRECCIONES EN EL AREA DE MEMORIA
         DEL DESPLEGADO.
        ;MODIFICA: A,C,DE
        ;LLAMA: actpue

ACTDIR: EX   DE,HL
        LD   C,E
        CALL ACTPUE
        EX   DE,HL
        LD   A,H
        CALL CONVI
        LD   DE,(BUF4)
        LD   (DIRDES),DE
        CALL DESP
        RET

        ;ALRLD: TOMA LA ULTIMA TECLA PULSADA Y:
        ;- SI ES DATO, ROTA A LA IZQUIERDA LOS 4 NIBBLES DE
        ;LAS DOS LOCALIDADES DE MEMORIA CONTIGUAS, DE LAS
        ;CUALES LA MAS ALTA ES DIRECCIONADA POR HL, COLOCA
        ;ESTA ULTIMA TECLA COMO EL NIBBLE MENOS SIGNIFICATIVO
        ;DE (HL-1) Y REGRESA A QUIEN LO LLAMO.
        ;- SI ES LA TECLA "OTRO" BRINCA A LA TABLA DE RETORNO
        ;DE ARLD CON LA DIRECCION DE REGRESO EN DE PARA
        ;CONTINUAR CON EL COMANDO QUE LA LLAMO.
        ;- SI ES CUALQUIER OTRO COMANDO DESPLEGA ERROR Y
        ;BRINCA A LA RUTINA DE RECONOCIMIENTO DE COMANDOS.
        ;ENTRADA: HL APUNTANDO A LA LOCALIDAD MAS ALTA DE LAS
        ;DOS POR ROTAR EN FORMA DECIMAL A LA IZQUIERDA Y EL
        ;BUFFER 2 CON LA ULTIMA TECLA PULSADA.
        ;SALIDA: (HL) Y (HL-1)
        ;MODIFICA: A,F,C,DE
        ;LLAMA: ALDB2

ALRLD:  CALL ALDB2
        LD   A,(BUFF1)
        BIT  4,A
        JR   Z,AARLD
        LD   A,(BUFF2)
        CP   10H
        JP   NZ,CERR
        POP  DE
        JP   GATO
AARLD:  LD   A,C
        DEC  HL
        RLD
```

```
        ;MEMORIA DEL DESPLEGADO, SENSA EL TECLADO, Y NO
        ;REGRESA HASTA QUE UNA TECLA SEA PULSADA
        ;ENTRADA: AREA DE MEMORIA DEL DESPLEGADO Y TECLADO
        ;SALIDA: DESPLEGADO Y BUFFERS 1 Y 2
        ;MODIFICA: A,F
        ;LLAMA: NINGUNA

LLAC1   CALL DESP
        LD   A,(BUFF1)
        CP   00H
        JP   Z,LLAC1
        BIT  4,A
        RET

        ;LLAC2: RECIBE EN EL BUFFER 2 EL NIBBLE MENOS
        ;SIGNIFICATIVO QUE JUNTO CON EL NIBBLE MENOR DE C
        ;FORMARAN UN BYTE QUEDANDO EN C. SI EL VALOR DEL
        ;BUFFER 2 REPRESENTA A "SALE" (04H) O "COM2" (0CH)
        ;DESPLEGA ERROR, PERO SI ES "COM1" (08H) DESPLEGA
        ;HOLA Y BRINCA A LA RUTINA DE RECONOCIMIENTO
        ;DEW COMANDOS.
        ;ENTRADA: BUFFER 2
        ;SALIDA: C
        ;MODIFICA: A,F,BC,D
        ;LLAMA: NINGUNA

LLAC2:  LD   A,(BUFF2)
        CP   10H
        JR   Z,LLAC3
        POP  BC
        CP   08H
        JP   Z,HOLA
        JP   CERR
LLAC3:  LD   A,D
        RLCA
        RLCA
        RLCA
        RLCA
        AND 0F0H
        OR   D
        LD   C,A
        RET

        ;DAT:  ESPERA A QUE UNA TECLA SEA PULSADA, SI ES DATO
        ;LO  CONVIERTE  A C7S DEJANDOLO EN  EL  NIBBLE  MENOS
        ;SIGNIFICATIVO DEL CAMPO DE DIRECCIONES EN EL AREA DE
        ;MEMORIA DEL DESPLEGADO Y REGRESA, PERO SI ES COMANDO
        ;DESPLEGA ERROR Y BRINCA A LA TABLA DE RECONOCIMIENTO
        ;DE COMANDOS.
        ;ENTRADA: TECLADO
        ;SALIDA:  CAMPO  DE DIRECCIONES EN EL AREA DE MEMORIA
        ;DEL DESPLEGADO
        ;MODIFICA. A,F,C
        ;LLAMA: LLAC,ALDB2
```

```
        RET
        DB   00H,00H,00H

        ;MBFF4:  CONVIERTE EL DATO DE A A C7S Y LO COLOCA  EN
        ;EL  CAMPO  DE  DATOS  DEL AREA DE  MEMORIA  PARA  EL
        ;DESPLEGADO.
        ;ENTRADA:A
        ;SALIDA:  CAMPO  DE DATOS EN EL AREA DE  MEMORIA  DEL
        ;DESPLEGADO.
        ;MODIFICA: A,F Y REGISTROS ALTERNOS
        ;LLAMA: CONVI

MBFF4:  PUSH DE
        CALL CONVI
        LD   DE,(BUFF4)
        LD   (DATDES),DE
        POP  DE
        RET

        ;ACTREG:  RECIBE AL REGISTRO DEL APUNTADOR A LA TABLA
        ;DE  REGISTROS,  COLOCA EL NOMBRE DEL REGISTRO EN  EL
        ;BYTE MENOS SIGNIFICATIVO DEL CAMPO DE DIRECCIONES EN
        ;EL  AREA DE MEMORIA DEL DESPLEGADO,  DECREMENTA DE Y
        ;REGRESA
        ;ENTRADA:  DE  APUNTANDO  AL REGISTRO DESEADO (EN  LA
        ;TABLA DE REGISTROS)
        ;SALIDA:   BYTE  MENOS  SIGNIFICATIVO  DEL  CAMPO  DE
        ;DIRECCIONES EN EL AREA DE MEMORIA DEL DESPLEGADO.
        ;MODIFICA: A,DE
        ;LLAMA: NINGUNA

ACTREG: LD   A,(DE)
        LD   (DIRDES+2),A
        DEC  DE
        LD   A,(DE)
        LD   (DI9RDES+3),A
        RET

        ;BFF90: COLOCA LOS BUFFERS 9 6 8 EN CEROS.
        ;ENTRADA: NINGUNA
        ;SALIDA: BUFFERS 9 & 8
        ;MODIFICA: A,F,HL
        ;LLAMA: NINGUNA

BFF90:  LD   HL,BUFF9
        XOR  A
        LD   (HL),A
        INC  HL
        LD   (HL),A
        RET

        ;BFF98: EL CONTENIDO DE LOS BUFFERS 9 Y 8 ES COLOCADO
        ;EN  EL CAMPO DE DIRECCIONES DEL AREA DE MEMORIA  DEL
        ;DESPLEGADO.
        ;ENTRADA:DATOS (16 BITS) O DIRECCIONES EN LOS BUFFERS 9 Y
```

```
        ;SALIDA:  CAMPO DE DIRECCIONES EN EL AREA DE  MEMORIA
        ;DEL DESPLEGADO Y HL APUNTANDO AL BUFER 8
        ;MODIFICA: A,C,DE,HL
        ;LLAMA: ACTDIR

BFF98:  LD    HL,(BUFF9)
        CALL  ACTDIR
        LD    HL,BUFF8
        RET

        ;C3CAL1:  DESPLEGA  EN  EL CAMPO DE  DIRECCIONES  DEL
        ;DESPLEGADO  ;EL MENSAJE APUNTADO POR HL.  SENSA  LAS
        ;TECLAS, EN CASO DE SER COMANDO DA ERROR Y SI ES DATO
        ;COLOCA LOS BUFFERS 9 Y 8 EN CEROS

C3CAL1: CALL  DESP
        CALL  CLAC
        CALL  BFF90
        RET

        ORG 0738H

        ;TABLAS;

TA2COM: DB    02H,81H       ;DCDR
DMBM:   DB    02H,0DEH
DCM:    DB    02H,0ECH
DDAM:   DB    03H,08H
DDAR:   DB    03H,25H
DPM:    DB    00H,0B6H
D6:     DB    00H,0B6H      ;00B6H=CERR
D7:     DB    00H,0B6H
D8:     DB    00H,0B6H
D9:     DB    00H,0B6H
DA:     DB    00H,0B6H
DDB:    DB    00H,0B6H
DC:     DB    00H,0B6H
DD:     DB    00H,0B6H
DDE:    DB    00H,0B6H
DDEN:   DB    00H,0B6H
TADEM:  DB    0,0,0,0
TAE:    DB    0,0,0,0
TAD:    DB    0,0,0,0
TAC:    DB    0,0,0,0
TAB:    DB    0,0,0,0
TAA:    DB    0,0,0,0
TA9:    DB    0,0,0,0
TA8:    DB    0,0,0,0
TA7:    DB    0,0,0,0
TA6:    DB    0,0,0,0
TAPM:   DB    73H,50H,5CH,00H
TADAR:  DB    5EH,77H,50H,00H
TADAM:  DB    5EH,77H,00H,00H
TACM:   DB    58H,5CH,00H,00H
TAMBM:  DB    7CH,18H,5CH,00H
```

```
TACDR:  DB    58H,5EH,50H,00H

        ;TABLA DE REGISTROS (CODIGO DE 7 SEGMENTOS, C7S)

        DB    20H,38H,20H
        DB    76H,20H,79H,20H
        DB    5EH,20H,39H,20H
        DB    7CH,20H,71H,20H
        DB    77H,18H,73H,76H
        DB    73H,18H,6DH,76H
        DB    6DH,18H,66H,76H
        DB    66H,18H,76H,76H
        DB    76H,00H,50H,00H
        DB    06H,00H,38H,00H
        DB    76H,00H,79H,00H
        DB    5EH,00H,39H,00H
        DB    7CH,00H,71H,00H
TAMREG: DB    77H
TAGO:   DB    6DH,77H,38H,79H
TACOM2: DB    39H,3FH,00H,5BH
TAREG:  DB    50H,00H,00H,00H
TADIR;  DB    5EH,04H,50H,00H
TAPS:   DB    73H,6DH,00H,00H
TAPE:   DB    76H,79H,40H,00H
TACOM1: DB    39H,3FH,00H,06H
TAHOLA: DB    76H,3FH,38H,77H
TAERR:  DB    79H,50H,50H,00H
TABLA7: DB    3FH,06H,5BH,4FH
        DB    66H,6DH,7DH,07H
        DB    7FH,67H,77H,7CH
        DB    39H,5EH,79H,71H
        END;
```

APENDICE B

DESCRIPCION DE LAS INSTRUCCIONES.

ADC A,(HL) (ADC M) Sumar memoria con acarreo.

Función: A <- A + (HL) + C

Descripción: El contenido de la localidad de memoria direccionada por el registro HL y la bandera de acarreo se suman con el acumulador. El resultado se queda en el acumulador.

Formato: 1 0 0 0 1 1 1 0 8E Duración: 2 Ciclos; 7 Estados

Direccionamiento: indirecto Banderas: S,Z,H,P/V,C;N=0

ADC A,(X+D) Sumar memoria indexada con acarreo.

Función: A <- A + (X+D) + C

Descripción: El contenido de la localidad de memoria direccionada por x+d, donde x es el registro indice y d es el valor del desplazamiento, y la bandera de acarreo se suman con el acumulador. El resultado queda en el acumulador.

Formato: 1 1 X 1 1 1 0 1 Duración: 5 ciclos; 19 estados.
1 0 0 0 1 1 1 0 8 E
desplazamiento

Direccionamiento: indexado Banderas: S,Z,H,P/V,C;N=0

ADC A,n (ACI n) Sumar dato inmediato con acarreo.

Función: A <D A + n + C

Descripción: El contenido del segundo byte de la instrucción y la bandera de acarreo se suman con el acumulador. El resultado queda en el acumulador.

Formato: 1 1 0 0 1 1 1 0 CE Duración: 2 ciclos; 7 estados.
dato

Direccionamiento: inmediato Banderas: S,Z,H,P/V,C;N=0

ADC A,r (ADC r) Sumar registro con acarreo.

Función: A <- A + r + C

Descripción: El contenido del registro r y la bandera de acarreo se suman con el registrodor. El

resultado queda en el acumulador.

Formato: 1 0 0 0 1 R R R Duración: 1 ciclo; 4 estados.

Direccionamiento: implícito Banderas: S,Z,H,P/V,C;N=0

ADC HL,ss Sumar registro par con acarreo al registro HL.

Función: HL <- HL + ss + C

Descripción: El contenido del registro par ss y la bandera de acarreo suman con el registro par HL.El resultado queda en el registro par HL.

Formato: 1 1 1 0 1 1 0 1 ED Duración: 4 ciclos; 15 estados.
0 1 S S 1 0 1 0

Direccionamiento: implícito Banderas: S,Z,H,P/V,C;N=0

ADD A,(HL) (ADD M) Sumar memoria.

Función: A <- A + HL

Descripción: El contenido de la localidad de memoria direccionada por el registro par HL se suman con el acumulador. El resultado queda el el acumulador.

Formato: 1 0 0 0 0 1 1 0 86 Duración: 2 ciclos; 7 estados.

Direccionamiento: indirecto Banderas: S,Z,H,P/V,C;N=0

ADD A,(x+d) Sumar memoria indexada.

Función: A <- A + (x+d)

Descripción: El contenido de la localidad de memoria direccionada por x+d, donde x es el registro indice y d es el desplazamiento, se suman con el acumulador. El resultado queda en el acumulador.

Formato: 1 1 X 1 1 1 0 1 Duración: 1 ciclo; 4 estados.
1 0 0 0 0 1 1 0 8 E
desplazamiento

Direccionamiento: indexado Banderas: S,Z,H,P/V,C;N=0

ADD A,n (ADI n) Sumar dato inmediato.

Función: A <- A + n

Descripción: El contenido del segundo byte de la instrucción se suma con el acumulador. El resultado queda en el acumulador.

Formato: 1 1 0 0 0 1 1 0 CE Duración: 2 ciclos; 7 estados.
dato

Direccionamiento: inmediato Banderas: S,Z,H,P/V,C;N=0

ADD A,r (ADD r) Sumar registro.

Función: A <- A + r

Descripción: El contenido del registro r se suma con el acumulador. El resultado queda en el acumulador.

Formato: 1 0 0 0 0 R R R Duración: 4 ciclo; 4 estados.

Direccionamiento: implicito Banderas: S,Z,H,P/V,C;N=0

ADD HL,ss (ADD ss) Sumar registro par al registro par HL.

Función: HL <- HL + ss

Descripción: El contenido del registro par ss se suma con el registro par HL. El resultado queda en el registro par HL.

Formato: 0 0 S S 1 0 0 1 6 Duración: 3 ciclos; 11 estados.

Direccionamiento: implicito Banderas: H,C:N=0

ADD x,rr Sumar registro par al registro indice

Función: x <- x + rr

Descripción: El contenido del registro par rr se suma con el registro indice x. El resultado queda en el registro indice x.

Formato: 1 1 X 1 1 1 0 1 Duración: 4 ciclos; 15 estados.
0 0 R R 1 0 0 1 DD

Direccionamiento: implicito Banderas: H=1 si hay acarreo del bit 11,
C=1 si hay acarreo del bit 15.

AND (HL) (ANA M) AND memoria.

Función: A <- A + ^(HL)

Descripción: Cada bit en el acumulador es reemplazado por el resultado de la operación lógica AND efectuada entre el bit que existia originalmente en el acumulador y el correspondiente en el byte almacenado en la localidad de memoria apuntada por HL.

Formato: 1 0 1 0 0 1 1 0 A6 Duración: 2 ciclos; 7 estados.

Direccionamiento: indirecto Banderas: S,Z,H,P/V,H=1,N=0,C=0

AND (x + d) AND memoria indexada.

Función: A <- A ^ (x + d)

Descripción: Se efectúa bit a bit una operación AND entre el acumulador y el contenido de la localidad de memoria direccionada por x+d, donde x es el registro índice y d es el valor del desplazamiento. El resultado queda en al acumulador

Formato: 1 1 X 1 1 1 1 1 A6 Duración: 5 ciclos; 19 estados.
1 0 1 0 0 1 1 0

Direccionamiento: indexado Banderas: S,Z,P/V,H=1;N=0;C=0

AND n (ANI n) dato inmediato.

Función: A <- A ^ n

Descripción: Se efectúa bit a bit una operación AND entre el acumulador y el contenido del segundo byte de la instrución. El resultado queda en el acumulador

Formato: 1 1 1 0 0 1 1 0 E6 Duración: 2 ciclos; 7 estados.
d a t o

Direccionamiento: inmediato Banderas: S,Z,P/V;H=1;N=0;C=0

AND r (ANA r) AND registro.

Función: A <- A ^ r

Descripción: Se efectúa bit a bit una operación AND entre el acumulador y el contenido del registro r. El resultado queda en el acumulador.

Formato: 1 0 1 0 0 R R R Duración: 1 ciclo; 4 estados.

Direccionamiento: implícito Banderas: S,Z,P/V,H=1;N=0;C=0

BIT b,(HL) Probar bit de memoria.

Función: Z <- $\overline{(HL)b}$

Descripción: Se prueba el bit especificado de la localidad de memoria direccionada por el registro par HL. La bandera zero indica el estado lógico del bit.

Formato: 1 1 0 0 1 0 1 1 CB Duración: 3 ciclos; 12 estados.
0 1 B B B 1 1 0

Direccionamiento: indirecto Banderas: S,Z,P/V,H=1; N=0

BIT b,(x+d) Probar bit de memoria indexada.

Función: $Z \leftarrow \overline{(x+d)b}$

Descripción: Se prueba el bit especificado de la localidad de memoria direccionada por x+d, donde x es el registro índice y d es el valor del desplazamiento. La bandera cero indica el estado lógico del bit.

Formato: 1 1 X 1 1 1 0 1 Duración: 5 ciclos; 20 estados.
1 1 0 0 1 0 1 1 CB
desplazamiento
0 1 B B B 1 1 0

Direccionamiento: indexado Banderas: Z,S,P/V;H=1;N=0

BIT b,r Probar bit de registro.

Función: Z <- rb

Descripción: El bit especificado del registro dado se prueba. La bandera cero indica el estado lógico del bit.

Formato: 1 1 0 0 1 0 1 1 Duración: 2 ciclos; 8 estados.
0 1 B B B R R R CB

Direccionamiento: implicito Banderas: Z,SP/V;H=1;N=0

CALL cc,pq (Ccc pq) Llamada condicional a subrutina.

Función: Si cc es verdadera: (SP-1)<-PCH;(SP-2)<-PCL
SP<-SP-2;PC<-pq
Si cc es falsa: PC<-PC+3

Descripción: Si la condición especificada es verdadera, el contenido del contador del programa es almacenado en la pila. En seguida, el contenido del segundo byte de la instrucción es cargaso en la parte baja del PC y el contenido del tercer byte es cargado en la parte alta del PC. En caso contrario la ejecución continùa secuencialmente.

Formato: 1 1 0 0 1 1 0 1 CD Duración: condición verdadera:
dirección baja 5 ciclos; 17 estados
dirección alta condición falsa:
3 ciclos; 10 estados

Direccionamiento: inmediato Banderas: ninguna

CALL pq (Call pq) Llamada a subrutina.

Función: (SP-1)<-PCH;(SP-2)<-PCL; SP<-SP-2;PC<-pq

Descripción: El contenido del PC es almacenado en la pila. Luego, el contenido del segundo byte de la instrucción es cargado en la parte baja del PC y el contenido del tercer byte de la instrucción es cargado en la parte alta del PC. El control se transfiere a esta nueva dirección.

Formato: 1 1 0 0 1 1 0 1 CD Duración: 5 ciclos; 17 estados
dirección baja
dirección alta

Direccionamiento: inmediato Banderas: ninguna

CCF (CMC) Complementar la bandera de acarreo.

Función: $C \leftarrow \overline{C}$

Descripción: Se invierte el estado lógico de la bandera de acarreo.

Formato: 0 0 1 1 1 1 1 1 3F Duración: 1 ciclo; 4 estados.

Direccionamiento: implicito Banderas: C;N=0;H=estaado previo de C

CP(HL) (CMP M) Comparar memoria.

Función: A - (HL)

Descripción: El contenido de la localidad de memoria direccionada por el registro par HL es restado al acumulador. El estado de las banderas Z y C se establece de acuerdo al resultado de la resta.

Formato: 1 0 1 1 1 1 1 0 BE Duración: 2 ciclos; 7 estados.

Direccionamiento: indirecto Banderas: Z=1 si A=(HL);C=1 si A<(HL); S,H,P/V;N=1

CP(x+d) Comparar con memoria indexada.

Función: A - (x+d)

Descripción: El contenido de la localidad de memoria direccionada por x+d. donde x es el registro indice y d es desplazamiento, es restado al acumulador. El contenido del acumulador no se altera. El estado de las banderas Z y C se

establece de acuerdo al resultado de la resta.

Formato: 1 1 X 1 1 1 0 1 Duración: 5 ciclos; 19 estados.
1 0 1 1 1 1 1 0 BE
desplazamiento

Direccionamiento: indexado Banderas: Z=1 si A=(x+d);
C=1 si A<(x+d);
S,H,P/V;N=1

CP n (CPI n) Comparar con dato inmediato.

Función: A - n

Descripción: El contenido del segundo byte de la instrucción es restado al acumulador. El contenido del acumulador no se altera. El estado de las banderas Z y C se establece de acuerdo al resultado de la resta.

Formato: 1 1 1 1 1 1 1 0 FE Duración: 2 ciclos; 7 estados.
d a t o

Direccionamiento: inmediato Banderas: Z=1 si A=n; C=1 si A<n;
S,H,P/V;N=1

CP r (CMP r) Comparar registro

Función: A - r

Descripción: El contenido del registro r es restado al acumulador. El contenido de éste no se altera. El estado de las banderas Z y C se establece de acuerdo al resultado de la resta.

Formato: 1 0 1 1 1 R R R FE Duración: 1 ciclo; 4 estados.

Direccionamiento: implicito Banderas: Z=1 si A=r; C=1 si
si A<n; S,H,P/V;N=1

CPD Comparar con decremento.

Función: A-(H);HL<-HL-1;BC<-BC-1; repetir hasta que BC=0
o A=(HL)

Descripción: El contenido de la localidad de memoria direccionada por el registro par HL es restado al acumulador. El contenido del acumulador no se altera. En seguida tanto el registro par HL como el registro par BC son decrementados.

Formato: 1 1 1 0 1 1 0 1 ED Duración: 4 ciclos; 16 estados.
1 0 1 0 1 0 0 1 A9

Direccionamiento: indirecto Banderas: Z=1 si A=(HL);P/V=0
si BC=/0; P/V=1 si BC=0;S,H;N=1

CPDR Comparar bloque con decremento.

Función: A-(HL);HL<-HL-1;BC<-BC-1; repetir hasta que BC=0 o A=(HL)

Descripción: El contenido de la localidad de memoria direccionada por el registro par HL es restado al acumulador. El contenido del acumulador no se altera. En seguida los registros pares BC y HL son decrementados. Si BC=0 y A=(HL), el PC se decrementa en 2 y la instrucción se vuelve a ejecutar.

Formato: 1 1 1 0 1 1 0 1 ED Duración para BC=0 o A=(HL):
1 0 1 1 1 0 0 1 B9 4 ciclos; 16 estados
Duración para BC=/0 o A=/(HL):
5 ciclos; 21 estados

Direccionamiento: indirecto Banderas: Z=1 si A=(HL);P/V=0 si BC=/0; P/V=1 si BC=/0;S,H;N=1

CPI Comparar con incremento.

Función: A-(HL);HL<-HL+1;BC<-BC-1

Descripción: El contenido de la localidad de memoria direccionada por el registro par HL es restado al acumulador. El contenido del acumulador no se altera. El registro par HL se incrementa y el registro par BC se decrementa.

Formato: 1 1 1 0 1 1 0 1 ED Duración: 4 ciclos;16 estados
1 0 1 0 0 0 0 1 A1

Direccionamiento: indirecto Banderas: Z=1 si A=(HL);P/V=0 si BC=/0; P/V=1 si BC=/0;S,H;N=1

CPIR Comparar bloque con incremento.

Función: A-(HL);HL<-HL+1;BC<-BC-1; repetir hasta que BC=0 o A=(HL)

Descripción: El contenido de la localidad de memoria direccionada por el registro par HL es restado al acumulador. El contenido del acumulador no se altera. En seguida el registro par HL se incrementa y el registro par BC se decrementa. Si BC=0 y A=HL, el PC se decrementa en 2 y la instrucción se vuelve a ejecutar.

Formato: 1 1 1 0 1 1 0 1 ED Duración para BC=0 o A=(HL):
1 0 1 1 0 0 0 1 B1 4 ciclos; 16 estados
Duración para BC=/0 o A=/(HL):

5 ciclos; 21 estados

Direccionamiento: indirecto Banderas: Z=1 si A=(HL);P/V=0 si BC=/0; P/V=1 si BC=/0;S,H;N=1

CPL (CMA) Complemento al acumulador.

Función: A <- $\overline{A}$

Descripción: El contenido del acumulador es complementado, es decir, invertido (complemento a 1) y el resultado queda en el acumulador.

Formato: 0 0 1 0 1 1 1 1 2F Duración: 1 ciclo; 4 estados.

Direccionamiento: implicito Banderas: H=1; N=1

DAA (DAA) Ajuste decimal al acumulador.

Descripción: El número de 8 bits en el acumulador es ajustado para que forme 2 dígitos de 4 bits en BCD por medio del siguiente proceso:

A) Después de una operación de adición (ADD, ADC, INC);N=0;

1. El valor de los 4 bits menos significativos del acumulador es mayor que 9 o si la bandera H=1, se suma 6 al acumulador.
2. Si el valor de los 4 bits más significativos del acumulador es ahora mayor que 9 o si la bandera C=1, se suma 6 a los 4 bits más significativos del acumulador.

B) Después de una operación de resta (SUB, SBC, DEC, NEG);N=1

1. Si la bandera H=1, se resta 6 de los 4 bits menos significativos del acumulador.

2. Si la bandera C=1, se resta 6 de los 4 bits más significativos del acumulador.

Formato: 0 0 1 0 0 1 1 1 27 Duración: 1 ciclo; 4 estados.

Direccionamiento: implicito Banderas: S,Z,H,P/V,C

DEC (HL) (DCR M) Decrementar memoria.

Función: (HL)<-(HL)-1

Descripción: El contenido del acumulador es complementado, es decir, invertido (complemento a 1) y el resultado queda en el acumulador.

Formato: 0 0 1 0 1 1 1 1 2F Duración: 1 ciclo; 4 estados

Direccionamiento: implicito Banderas: H=1; N=1

DEC (x+d) Decrementar memoria indexada.

Función: (x+d)<-(x+d)-1

Descripción: El contenido de la localidad de memoria direccionada por x+d, donde x es el registro indice y d es el valor del desplazamiento,se decrementa en uno.

Formato: 1 1 X 1 1 1 0 1 Duración: 6 ciclos;23 estados.
0 0 1 1 0 1 0 1 35
desplazamineto

Direccionamiento: indexado Banderas: S,Z,H,P/V;N=1

DEC r (DCR r) Decrementar registro.

Función: r <- r-1

Descripción: El contenido del registro r se decrementa en uno.

Formato: 0 0 R R R 1 0 1 Duración: 1 ciclo;4 estados.

Direccionamiento: implicto Banderas: S,Z,H,P/V;N=1

DEC ss (DCX ss) Decrementar registro par.

Función: ss <- ss-1

Descripción: El contenido del registro par ss se decrementa en uno.

Formato: 0 0 S S 1 0 1 1 Duración: 1 ciclo;6 estados.

Direccionamiento: implicto Banderas: ninguna.

DEC x Decrementar registro indice.

Función: x <- x-1

Descripción: El contenido del registro indice se decrementa en uno.

Formato: 1 1 X 1 1 1 0 1 Duración: 2 ciclos;10 estados.
0 0 1 0 1 0 1 1 2B

Direccionamiento: implicto Banderas: ninguna.

DI (DI) Desactivar interrupciones.

Función: IFF <- 0

Descripción: Se ponen en cero los flip-flops de interrupciones, desactivando por lo tanto todas las interrupciones inhibibles. Una interrupción inhibible puede ser desactivada durante su ejecución por DI. Es reactivada por una instrucción EI.

Formato: 1 1 1 1 0 0 1 1 F3 Duración: 1 ciclo;4 estados.

Direccionamiento: implicto Banderas: ninguna.

DJNZ e Decrementar el registro B y saltar relativamente si no es cero.

Función: B <- B-1; si B=/0: PC <- PC + e

Descripción: El registro B es decrementado. Si el resultado no es cero, el valor del desplazamiento inmediato se suma al PC usando aritmética complemento a dos para permitir saltos hacia adelante y hacia atrás. El valor del desplazamiento se añade al valor del PC+2 (después de haber obtenido los dos bytes de la instrucción). Por tanto, el desplazamiento efectivo es -126 a +129 bytes.

Formato: 0 0 0 1 0 0 0 0 10 e-2 Duración: B=/0:3 ciclos; 13 estados. B=0:2 ciclos; 8 esatados

Direccionamiento: inmediato Banderas: ninguna.

EI (EI) Activar interrupciones.

Función: IFF <- 1

Descripción: Se ponen en uno los flip-flops de interrupciones, activando tanto todas las interrupciones inhibibles después de la ejecución de la instrucción siguiente a EI. Mientras tanto, las interrupciones inhibibles estan desactivadas.

Formato: 1 1 1 1 1 0 1 1 FB Duración: 1 ciclo;4 estados.

Direccionamiento: implicto Banderas: ninguna.

EX AF,AF' Intercambiar acumulador y banderas por sus registros internos.

Función: AF <-> AF'

Descripción: El contenido del acumulador y del registro de banderas se intercambian con el contenido del

acumulador alterno y el registro de banderas alterno.

Formato: 0 0 0 0 1 0 0 0 08 Duración: 1 ciclo;4 estados.

Direccionamiento: implicto Banderas: S,Z,H,P/V,NC

EX DE,HL' (XCHG) Intercambiar registros HL y DE.

Función: DE <-> HL'

Descripción: Los contenidos de los registros pares DE y HL se intercambian.

Formato: 1 1 1 0 1 0 1 1 E8 Duración: 1 ciclo;4 estados.

Direccionamiento: implicto Banderas: ninguna

EX (SP),HL (XTHL) Intercambiar HL con su parte superior de la pila.

Función: (SP) <-> L; (SP+1) H

Descripción: El contenido del registro L se intercambia con el contenido de la localidad de memoria direccionada por el registro SP. El contenido del registro H se intercambia con el contenido de la localidad de memoria direccionada por el registro SP+1.

Formato: 1 1 1 0 0 0 1 1 E3 Duración: 5 ciclos;19 estados.

Direccionamiento: indirecto Banderas: ninguna

EX (SP),x Intercambiar registro indice con parte superior de la pila.

Función: (SP) <-> x bajo; (SP+1) <-> x alto

Descripción: Los ocho bits menos significativos del registro índice se intercambian con el contenido de la localidad de memoria direccionada por el registro SP. Los ocho bits más significativos del registro índice se intercambian con el contenido de la localidad de memoria direccionada por el registro SP+1.

Formato: 1 1 X 1 1 1 0 1 Duración: 6 ciclos;23 estados.
1 1 1 0 0 0 1 1 E3

Direccionamiento: indirecto Banderas: ninguna

EXX Intercambiar registros por sus registros alternos.

Función: BC <-> BC'; DE <-> DE'; HL <-> HL'

Descripción: El contenido de los registros de propósito general se intercambia con el contenido de los registros alternos correspondientes.

Formato: 1 1 0 1 1 0 0 1 D9 Duración: 1 ciclo;4 estados.

Direccionamiento: implicito Banderas: ninguna

HALT (HLT) Detener el funcionamiento de la CPU.

Descripción: La CPU suspende su operación, pero permanece ejecutando instrucciones NOP para continuar con los ciclos de refrescado de las memorias dinámicas hasta que se recibe una interrupción o un reset.

Formato: 0 1 1 1 0 1 0 1 76 Duración: 1 ciclo;4 estados.

Direccionamiento: implicito Banderas: ninguna

IM 0 Establecer modo de interrupción 0.

Función: Control interno de interrupciones.

Descripción: Establecer el modo de interrupción 0. En esta condición el dispositivo que causa la interrupción puede insertar en el bus de datos una instrucción a ejecutar, el primer byte de la cual debe aparecer durante el ciclo de espuesta de la interrupción (interrupt acknowledge cycle).

Formato: 1 1 1 0 1 1 0 1 ED Duración: 2 ciclos;8 estados.
0 1 0 0 0 1 1 0 46

Direccionamiento: implicito Banderas: ninguna

IM 1 Establecer modo de interrupción 1.

Función: Control interno de interrupciones.

Descripción: Establecer el modo de interrupción 1. En esta condición una instrucción RST 38H será ejecutada cuando ocurra una interrupción.

Formato: 1 1 1 0 1 1 0 1 ED Duración: 2 ciclos;8 estados.
0 1 0 1 0 1 1 0 56

Direccionamiento: implicito Banderas: ninguna

IM 2 Establecer modo de interrupción 2.

Función: Control interno de interrupciones.

Descripción: Establecer el modo de interrupción 2. Cuando ocurre una interrupción, el dispositivo que la causa debe proporcionar un byte de información que es usado como la parte baja de una dirección. La parte alta de esta dirección se toma del contenido del registro I. La dirección así obtenida apunta a una segunda dirección almacenada en memoria, la cual se carga en el PC para continuar su ejecucución.

Formato: 1 1 1 0 1 1 0 1 ED Duración: 2 ciclos;8 estados.
0 1 0 1 1 1 1 0 5E

Direccionamiento: implicito Banderas: ninguna

IN r,(C) Carga registro ccon puerto.

Función: rC)

Descripción: El dispositivo periférico direccionado por el contenido del registro C es leido y el resultado se coloca en el registro r.El registro C proporciona los bits A0 a A7 del bus de direcciones, el registro B proporciona los bits A8 a A15.

Formato: 1 1 1 0 1 1 0 1 ED Duración: 2 ciclos;8 estados.
1 1 R R R 0 0 0

Direccionamiento: externo Banderas: S,Z,H,P/V;N=0

IN A,(N) (IN N) Cargar acumulador con puerto.

Función: A<-(C)

Descripción: El dispositivo periférico direccionado por el segundo byte de la instrucción es leido y el resultado se coloca en el acumulador. El byte N proporciona los bits A0 a A7 del bus de direcciones. El registro A proporciona los bits A8 a A15.

Formato: 1 1 0 1 1 0 1 1 DB Duración: 3 ciclos;11 estados.
dirección puerto
Direccionamiento: externo Banderas: ninguna.

INC r (INR r) Incrementar registro.

Función: r<-r+1

Descripción: El contenido del registro r se incrementa en

uno.

Formato: 0 0 R R R 1 0 0 Duración: 1 ciclos;4 estados.

Direccionamiento: implicito Banderas: S,Z,H,P/V;N=0

INC (HL) (INR M) Incrementar memoria.

Función: (HL)<-(HL)+1

Descripción: El contenido de la localidad de memoria direccionada por el registro par HL se incrementa en uno.

Formato: 0 0 1 1 0 1 0 0 34 Duración: 3 ciclos;11 estados.

Direccionamiento: indirecto Banderas: S,Z,H,P/V;N=0

INC (x+d) Incrementar memoria indexada.

Función: (x+d)<-(x+d)+1

Descripción: El contenido de memoria direccionada por x+d, donde x es el registro indice y d es el valor del desplazamiento, se incrementa en uno.

Formato: 1 1 X 1 1 1 0 1 Duración: 6 ciclos;23 estados.
0 0 1 1 0 1 0 0 34
desplazamiento

Direccionamiento: indexado Banderas: S,Z,H,P/V;N=0

INC ss (INX ss) Incrementar registro de 16 bits.

Función: ss<-ss+1

Descripción: El contenido del registro de 16 bits se incrementa en uno.

Formato: 0 0 S S 0 0 1 1 Duración: 3 ciclos;6 estados.

Direccionamiento: implicito Banderas: ninguna

INC x Incrementar registro indice.

Función: x<-x+1

Descripción: El contenido del registro indice x se incrementa en uno.

Formato: 1 1 X 1 1 1 0 1 Duración: 2 ciclos;10 estados.
0 0 1 0 0 0 1 1 23

Direccionamiento: implicito Banderas: ninguna

IND Entrada desde puerto con decremento

Función: (HL)<-(C); B<-B-1; HL<-HL-1

Descripción: El dispositivo periférico direccionado por el registro C es leído y el resultado se guarda en la localidad de memoria direccionada por el registro par HL. El registro B y el registro par HL se decrementan.

Formato: 1 1 1 0 1 1 0 1 ED Duración: 4 ciclos;16 estados.
1 0 1 0 1 0 1 0 AA

Direccionamiento: externo Banderas: S,H,P/V;N=1;
Z=1 si B=0;Z=0 si B=/0

INDR Entrada de bloque desde puerto con decremento.

Función: (HL)<-(C); B<-B-1; HL<-HL-1; repetir hasta que B=0

Descripción: El dispositivo periférico direccionado por el registro C es leído y el resultado se guarda en la localidad de memoria direccionada por el registro par HL. Luego el registro B y el registro par HL se decrementan. Si B no es cero, el PC se decrementa en 2 y la instrucción se ejecuta de nuevo.

Formato: 1 1 1 0 1 1 0 1 ED Duración: 4 ciclos;16 estados.
1 0 1 1 1 0 1 0 BA

Direccionamiento: externo Banderas: S,H,P/V;Z=1;N=1

INI Entrada desde puerto con incremento.

Función: (HL)<-(C); B-1; HL<-HL+1

Descripción: El dispositivo periférico direccionado por el registro C es leído y el resultado se guarda en la localidad de memoria direccionada por el registro de 16 bits HL. El registro B se decrementa y el registo HL se incrementa. El contenido del regsitro B se coloca en la parte baja del bus de direcciones. El contenido del registro C se coloca en la parte alta. La selección del puerto se hace generalmente por C, esto es, por A0 a A7. B es un contador de bytes.

Formato: 1 1 1 0 1 1 0 1 ED Duración: 4 ciclos;16 estados.
1 0 1 0 0 0 1 0 A2

Direccionamiento: externo Banderas: S,H,P/V;N=1;Z=1 si B=0;
Z=0 si B=/0

INIR Entrada de bloque desde puerto con incremento.

Función: (HL)<-(C);B<- B-1; HL<-HL+1; repetir hasta que B=0

Descripción: El dispositivo periférico direccionado por el registro C es leído y el resultado se guarda en la localidad de memoria direccionada por el registro de 16 bits HL. El registro B se decrementa y el registo HL se incrementa. Si B no es cero, el PC se decrementa en 2 y la instrucción se ejecuta de nuevo.

Formato: 1 1 1 0 1 1 0 1 ED Duración: B=0:4 ciclos;
1 0 1 1 0 0 1 0 B2 16 estados
B=/0:5 ciclos;21 estados
Direccionamiento: externo Banderas: S,H,P/V;N=1;Z=1

JP cc,pq (Jcc pq) Saltar condicionalmente.

Función: Si cc:PC<-pq

Descripción: Si la condición especificada es verdadera, se transfiere el control a la instrucción cuya dirección está especificada en el segundo y tercer bytes de la instrucción de salto. El segundo byte se carga en la parte baja del PC y el tercer byte en la parte alta del PC. Si la condición es verdadera, se continúa la ejecución secuencialmente.

Formato: 1 1 C C C 1 1 0 Duración: 3 ciclos;10 estados.
dirección baja
dirección alta

Direccionamiento: inmediato Banderas: ninguna

JP pp (JMP pq) Saltar incondicionalmente.

Función: PC <- pq

Descripción: Se transfiere el control a la instrucción cuya dirección está especificada en el segundo y tercer bytes de la instrucción de salto. El segundo byte se carga en la parte baja del PC y el tercer byte en la parte alta del PC.

Formato: 1 1 0 0 0 0 1 1 C3 Duración: 3 ciclos;10 estados.
dirección baja
dirección alta

Direccionamiento: inmediato Banderas: ninguna

JP (HL) (PCHL) Saltar a donde indica HL.

Función: PCH <- H; PCL<-L

Descripción: El contenido del registro H se carga en los ocho bits más significativos del PC.El contenido del registro L se carga en los ocho bits menos significativos del PC. La siguiente instrucción se obtiene de esta nueva dirección.

Formato: 1 1 1 0 1 0 1 1 E9 Duración: 1 ciclo ;4 estados.

Direccionamiento: implicito Banderas: ninguna

JP (x) Saltar a donde se indica el registro índice.

Función: PC <- x

Descripción: El contenido del registro índice se carga en el PC. La siguiente instrucción se obtiene de esta nueva dirección.

Formato: 1 1 X 1 1 1 0 1 Duración: 1 ciclo ;4 estados.
1 1 1 0 1 0 0 1 E9
Direccionamiento: implicito Banderas: ninguna

JR cc,e Saltar relativamente con condición.

Función: Si cc:PC<-PC+e

Descripción: Si la condición especificada es verdadera, el valor del desplazamiento inmediato se suma al PC usando aritmética complemento a dos para permitir saltos hacia adelante y hacia atrás. El valor del desplazamiento se añade al valor del PC+2 (después de haber obtenido los dos bytes de la instrucción). Como resultado, el desplazamiento es de -126 a + 129 bytes. Si la condición no es verdadera, la ejecución del programa continúa secuencialmente, cc puede ser:NZ(00),Z(01),NC(10) y C(11).

Formato: 0 0 1 C C 0 0 0 Duración: condición verdadera:
e - 2 3 ciclos; 12 estados
condición falsa:
2 ciclos; 7 estados
Direccionamiento: inmediato Banderas: ninguna

JR e Saltar relativamente.

Función: PC<-PC+e
Descripción: El valor del desplazamiento inmediato se suma al PC usando aritmética complemento a dos para permitir saltos hacia atrás.El valor del desplazamiento se añade al valor del PC+2 (después de haber obtenido los dos bytes de la instrucción).Como resultado, el desplazamiento

efectivo es de -126 a +129 bytes.La instrucción continúa en esta nueva dirección.

Formato: 0 0 0 1 1 0 0 0 Duración:3 ciclos; 12 estados
e - 2

Direccionamiento: inmediato Banderas: ninguna

LD ss,(pq) Cargar registro de 16 bits con memoria

Función: ss bajo <- (pq); ss alto <- (pq+1)

Descripción: El contenido de la localidad de memoria direccionada por el tercer y cuarto byte de la instrucción se carga en el registro bajo del registro de 16 bits. El contenido de la localidad de memoria siguiente a la direccionada por la instrucción se carga en el registro alto del registro de 16 bits.

Formato: 1 1 1 0 1 1 0 1 ED Duración:6 ciclos; 20 estados
0 1 S S 1 0 1 1
dirección baja
dirección alta

Direccionamiento: directoto Banderas: ninguna

LD ss,nn (LXI ss,nn) Cargar registro de 16 bits con dato inmediato.

Función: ss <- nn

Descripción: El contenido del segundo byte de la instrucción se carga en el registro bajo del registro de 16 bits. El contenido del tercer byte de la instrucción se carga en el registro alto de 16 bits.

Formato: 0 0 S S 0 0 0 1 Duración:3 ciclos; 10 estados
dato bajo
dato alto

Direccionamiento: inmediato Banderas: ninguna

LD r,n (MVI r,n) Cargar registro con dato inmediato.

Función: r <- n

Descripción: El contenido del segundo byte de la instrucción se carga en el registro r.

Formato: 0 0 R R R 1 1 0 Duración:2 ciclos; 7 estados
dato

Direccionamiento: inmediato Banderas: ninguna

LD r,r' (MOV r,r') Cargar registro con registro.
Función: r <- r'

Descripción: El contenido del registro r'(origen) se carga en el registro r registro r(destino).

Formato: 0 1 R R R R'R'R' Duración:1 ciclo; 4 estados

Direccionamiento: implicito Banderas: ninguna

LD (ss),A (STAX ss) Cargar memoria con acumulador indirectamente.

Función: (ss) <- A

Descripción. El contenido del acumulador se carga en la localidad de memoria direccionada por el registro par ss. Sólo se pueden usarse los registros pares BC (00) y DE (01).

Formato: 0 0 S S 0 0 1 0 Duración:2 ciclos; 7 estados

Direccionamiento: indirecto Banderas: ninguna

LD (HL),n (MVI M,n) Cargar memoria con dato inmediato.

Función: (HL) <- n

Descripción: La localidad de memoria direccionada por el registro de 16 bits HLse carga con el contenido del segundo byte de la instrucción.

Formato: 0 0 1 1 0 1 1 0 36 Duración:3 ciclos;10 estados
dato

Direccionamiento: inmediato/indirecto Banderas: ninguna

LD (HL),r (MOV M,r) Cargar memoria con registro.

Función: (HL) <- r

Descripción: La localidad de memoria direccionada por el registro de 16 bits HL se carga con el contenido del registro r.

Formato: 0 1 1 1 0 R R R Duración:2 ciclos; 7 estados

Direccionamiento: indirecto Banderas: ninguna

LD r,(x+d) Cargar registro con memoria indexada.

Función: r <- (x+d)
Descripción: El contenido de la localidad de memoria direccionada por x+d, donde x es el registro

LD (x+d),n Cargar memoria indexada con dato inmediato.

Función: (x+d) <- n

Descripción: El contenido del cuarto byte de la instrucción se carga en la localidad de memoria direccionada por x+d, donde x es el registro índice y d es el desplazamiento.

Formato: 1 1 X 1 1 1 0 1 Duración:5 ciclos;19 estados
0 0 1 1 0 1 1 0 36
desplazamiento
dato

Direccionamiento: indexado/inmediato Banderas: ninguna

LD (x+d),r Cargar memoria indexada con registro.

Función: (x+d) <- r

Descripción: El contenido del registro r se carga en la localidad de memoria direccionada por x+d, donde x es el registro índice y d es el valor del desplazamiento.

Formato: 1 1 X 1 1 1 0 1 Duración:5 ciclos;19 estados
0 1 1 1 0 R R R
desplazamiento

Direccionamiento: indexado Banderas: ninguna

LD A,(pq) (LDA pq) Cargar acumulador con memoria directamente.

Función: A <- (pq)

Descripción: El contenido de la localidad de memoria direccionada por elsegundo byte y el tercer bytes de la instrucción se carga en el acumulador.

Formato: 0 0 1 1 1 0 1 0 3A Duración:4 ciclos;13 estados
dirección baja
dirección alta

Direccionamiento: directo Banderas: ninguna

LD (pq),A (STA pq) Cargar memoria con acumulador directamente

Función: (pq) <- A

Descripción: El contenido del acumulador se carga en la localidad de memoria direccionada por el segundo y tercer bytres de la instrucción.

Formato: 0 0 1 1 0 0 1 0 32 Duración:4 ciclos;13 estados
dirección baja
dirección alta

Direccionamiento: directo Banderas: ninguna

LD (pq),ss Cargar memoria con rgistro de 16 bits.

Función: (pq) <- ss bajo; (pq+1) <- ss alto

Descripción: El contenido del registro bajo del registro de 16 bits ss se carga en la localidad de memoria direccionada por el tercer y cuarto bytes de la instrucción. El contenido del registro alto del registro de 16 bits ss secarga en la localidad de memoria sigueiente a la direccionada por la instrucción.

Formato: 1 1 1 0 1 1 0 1 ED Duración:6 ciclos;20 estados
0 1 S S 0 0 1 1
dirección baja
dirección alta

Direccionamiento: directo Banderas: ninguna

LD (pq),HL (SHLD pq) Cargar memoria con HL.

Función: (pq) <- L; (pq+1) <- H

Descripción: El contenido del registro L se carga en la localidad de meoria direccionada por el segundo y tercer byte de la instrucción.El contenido del registro H se carga en la localidad de memoria siguiente a la direccionada por la instrucción.

Formato: 0 0 1 0 0 0 1 0 22 Duración:5 ciclos;16 estados
dirección baja
dirección alta

Direccionamiento: directo Banderas: ninguna

LD (pq),x Cargar memoria con registro índice.

Función: (pq) <- x bajo; (pq+1) <- x alto

Descripción: El contenido de la parte baja del registro índice se carga en la localidad de meoria direccionada por el tercer y cuarto bytes de la instrucción. El contenido de la parte alta del registro índice x se carga en la localidad de memoria siguiente a la direccionada por la instrucción.

Formato: 1 1 X 1 1 1 0 1 Duración:6 ciclos;20 estados
0 1 1 0 0 0 1 0
dirección baja
dirección alta

Direccionamiento: directo Banderas: ninguna

LD A,(ss) (LDAX ss) Cargar acumulador con memoria indirectamente.

Función: A <- (ss)

Descripción: El contenido de la localidad de meoria direccionada por el registro de 16 bits ss se carga en el acumulador. Nota: sólo pueden usarse los registro pares BC(00) y DE(01).

Formato: 0 0 S S 1 0 1 0 Duración:2 ciclos;7 estados

Direccionamiento: indirecto Banderas: ninguna

LD A,I Cargar acumulador con registro I.

Función: A <- I

Descripción: El contenido del registro del vector de interrupciones (I) se carga al acumulador.

Formato: 1 1 1 0 1 1 0 1 ED Duración:2 ciclos;9 estados
0 1 0 1 0 1 1 1 57

Direccionamiento: implícito Banderas: S,Z;H=0;N=0;P/V=IFF

LD I,A Cargar registro I con acumulador.

Función: I <- A

Descripción: El registro del vector de interrupciones (I) se carga con el contenido del acumulador.

Formato: 1 1 1 0 1 1 0 1 ED Duración:2 ciclos;9 estados
0 1 0 0 0 1 1 1 47

Direccionamiento: implícito Banderas: ninguna.

LD A,R Cargar acumulador con registro R.

Función: A <- R

Descripción: El contenido del registro para el refrescado de memorias dinámicas (R) se carga en el acumulador.

Formato: 1 1 1 0 1 1 0 1 ED Duración:2 ciclos;9 estados
0 1 0 1 1 1 1 1 57

Direccionamiento: implícito Banderas: S,Z;H=0;N=0;P/V=IFF

LD HL,(pq) (LHLD pq) Cargar HL con memoria.

Función: L <- pq; H <- (pq+1)

Descripción: El contenido de la localidad de meoria direccionada por el segundo y tercer bytes de la instrucción se carga en el registro L.El contenido de la localidad de memoria siguiente a la direccionada por la instrucción se carga en el registro H.

Formato: 0 0 1 0 1 0 1 0 2A Duración:5 ciclos;16 estados
dirección baja
dirección alta

Direccionamiento: directo Banderas: ninguna

LD x,nn Cargar registro indice con dato inmediato.

Función: x <- nn

Descripción: Los contenidos del tercer y cuarto bytes de la instrucción se cargan en el registro índice x. El tercer byte corresponde a los 8 bits menos significativos y el cuarto byte a los 8 bits más significativos.

Formato: 1 1 X 1 1 1 0 1 Duración:4 ciclos;14 estados
0 0 1 0 0 0 0 1 21
dirección baja
dirección alta

Direccionamiento: inmediato Banderas: ninguna

LD x,(pq) Cargar registro índice con memoria.

Función: x bajo <- (pq); x alto <- (pq+1)

Descripción: El contenido de la localidad de memoria direccionada por el tercer y cuarto bytes de la

instrucción se carga en la parte baja delregistro índice x. El contenido de la localidad de meoria direccionada por la instrucción, se carga en la parte alta del registro índice x.

Formato: 1 1 X 1 1 1 0 1 Duración:6 ciclos;20 estados
0 0 1 0 1 0 1 0 2A
dirección baja
dirección alta

Direccionamiento: directo Banderas: ninguna

LD R,A Cargar registro R con acumulador.

Función: R <- A

Descripción: El contenido del acumulador se carga en el registro para el refrescado de memorias dinámicas (R).

Formato: 1 1 1 0 1 1 0 1 ED Duración:2 ciclos;9 estados
0 1 0 0 1 1 1 1 47

Direccionamiento: implícito Banderas: ninguna.

LD SP,HL (SPHL) Cargar registro SP con registro HL.

Función: SP <- HL

Descripción: El contenido del registro HL se carga en el acumulador de pila (SP).

Formato: 1 1 1 1 1 0 0 1 F9 Duración:1 ciclo;6 estados

Direccionamiento: implícito Banderas: ninguna

LD SP,x Cargar registro SP con registro índice.

Función: SP <- x

Descripción: El contenido del registro índice x se carga en el apuntador de pila (SP).

Formato: 1 1 X 1 1 1 0 1 Duración:2 ciclos;10 estados
1 1 1 1 1 0 0 1 F9

Direccionamiento: implícito Banderas: ninguna.

LDD Transferir bloque con decremento.

Función: (DE) <- (HL);DE <- DE-1;HL <- HL-1;BC <- BC-1

Descripción: El contenido de la localidad de memoria

direccionada por el registro de 16 bits HL se carga en la localidad de memoria direccionada por el registro de 16 bits DE. Luego los registros pares BC, DE y HL se decrementan. El registro BC actúa como contador.

Formato: 1 1 1 0 1 1 0 1 ED Duración:4 ciclos;16 estados
1 0 1 0 1 0 0 0 A8

Direccionamiento: indirecto Banderas: P/V=0 si BC=0;P/V=1; si B=/0;H=0;N=0

LDDR Transferir bloque repetidamente con decremento

Función: (DE) <- (HL);DE <- DE-1;HL <- HL-1;BC <- BC-1; repetir hasta que BC=0

Descripció n: El contenido de la localidad de memoria direccionada por el registro de 16 bits HL se carga en la localidad de memoria direccionada por el registro de 16 bits DE. Luego los registros pares BC, DE y HL se decrementan. Si BC=/0, entonces el PC se decrementa en 2 y la instrucción se vuelve a ejecutar.

Formato: 1 1 1 0 1 0 0 0 E8 Duración:BC=/0; 5 ciclos;
1 0 1 0 1 0 0 0 A8 21 estados
BC=0; A ciclos; 16 estados
Direccionamiento: indirecto Banderas: H=0;P/V=0,N=0

LDI Transferir bloque con incremento.

Función: (DE) <- (HL);DE <- DE+1;HL <- HL+1;BC <- BC+1

Descripción: El contenido de la localidad de memoria direccionada por el registro de 16 bits HL se carga en la localidad de memoria direccionada por el registro de 16 bits DE. Luego los registros pares DE y HL se incrementan y el registro de 16 bits BC se decrementa.

Formato: 1 1 1 0 1 1 0 1 ED Duración: 4 ciclos;16 estados
1 0 1 0 0 0 0 0 A0

Direccionamiento: indirecto Banderas: P/V=0 sI BC=0;P/V=1 si BC=1; H=0; N=0

LDIR Transferir bloque repetitivamente con incremento.

Función: (DE) <- (HL);DE <- DE+1;HL <- HL+1;BC <- BC-1; repetir hasta que BC=0

Descripción: El contenido de la localidad de memoria direccionada por el registro de 16 bits HL se

carga en la localidad de memoria direccionada direccionada por el registro de 16 bits DE. Luego los registros pares DE y HL se incrementan y el registro de 16 bits BC se decrementa Si BC=/0, entonces el PC se decrementa en 2 y la instrucción se vuelve a ejecutar

Formato: 1 1 1 0 1 1 0 1 ED Duración:BC=/0:5 ciclos;
1 0 1 1 0 0 0 0 B0 21 estados
BC=0: 4 ciclos; 16 estados
Direccionamiento: indirecto Banderas: H=0;P/V=0;N=0;

LD r,(HL) (MOV r,M) Cargar registro con memoria

Función: r <- (HL)

Descripción: El contenido de la localidad de memoria direccionada por el registro de 16 bits HL se carga en el registro r.

Formato: 0 1 R R R 1 1 0 F9 Duración:2 ciclos;7 estados

Direccionamiento: indirecto Banderas: ninguna

NEG Negar acumulador.

Función: A <- 0-A

Descripción: El contenido del acumulador se resta de cero (complemento a dos) y el resultado queda en el acumulador.

Formato: 1 1 1 0 1 1 0 1 ED Duración: 2 ciclos; 8 estados
0 1 0 0 0 1 0 0 44

Direccionamiento: implicito Banderas: S,Z,H;N=1;C=1 si A=0 antes de la instrucción;
P/V=1 si A=80H antes de la instrucción.

NOP (NOP) No operación.

Función: Gastar tiempo.

Descripción: El procesador no hace nada durante un ciclo.

Formato: 0 0 0 0 0 0 0 0 00 Duración:1 ciclos;4 estados

Direccionamiento: implicito Banderas: ninguna

OR(HL) (ORA M) OR memoria.

Función: A <- A v (HL)

Descripción: Se efectúa bit a bit una operación OR entre el acumulador y el contenido de la localidad direccionada por el registro de 16 bits HL. El resultado queda en el acumulador

Formato: 1 0 1 1 0 1 1 0 B6 Duración:2 ciclos;7 estados

Direccionamiento: indirecto Banderas: S,Z,P/V;H=0,N=0;C=0

OR(x+d) OR memoria indexad

Función: A <- A v (x+d)

Descripción: Se efectúa bit a bit una operación OR entre el acumulador y el contenido de la localidad direccionada por x+d, donde x es el registro índice y d es el valor del esplazamiento. El resulttado queda en el acumulador.

Formato: 1 1 X 1 1 1 0 1 Duración:5 ciclos;19 estados
1 0 1 1 0 1 1 0 B6
desplazamiento

Direccionamiento: indexado Banderas: S,Z,P/V;H=0,N=0;C=0

OR n (OR) n) OR dato inmediato

Función: A <- A v n

Descripción: Se efectúa bit a bit una operación OR entre el acumulador y el contenido del segundo byte de instrucción.El resultado queda en el acumulador.

Formato: 1 1 1 1 0 1 1 0 F6 Duración: 2 ciclos; 7 estados
dato

Direccionamiento: inmediato Banderas:S,Z,P/V=0;H=0;N=0;C=0

OR r (ORA r) OR registro

Función: A <- A v r

Descripción: Se efectúa una operación OR entre el acumulador y el contenido del registro r. El resultado queda en el acumulador.

Formato: 1 0 1 1 0 R R R Duración: 1 ciclos; 4 estados

Direccionamiento: implicito Banderas: S,Z,P/V=0;H=0;N=0;C=0

OTDR Salida de bloque hacia puerto con decremento

Función: (C) <- (HL);B <- B-1;HL <- HL-1; repetir hasta

que B=0

Descripción: El contenido de la localidad de memoria direccionada por el registro HL es enviado al dispositivoo periférico direccionado por el contenido del registro C.Luego el registro B y el registro par HL se decrementan. Si B=/0, el PC se decrementa en 2 y la instrucción se ejecuta de nuevo. C proprciona los bits A0 a A7 del bus de direcciones. B

Formato: 1 1 1 0 1 1 0 1 ED Duración:B=0:4 ciclos;
1 0 1 1 1 0 1 1 B3 16 estados
B=/0:5 ciclos;21 estados

Direccionamiento: externo Banderas: S,Z,P/V;Z=1;N=1

OTIR Salida de bloque hacia puerto con incremento

Función: (C) <- (HL);B <- B-1;HL <- HL+1; repetir hasta que B=0

Descripción: El contenido de la localidad de memoria direccionada por el registro HL es enviado al dispositivo periférico direccionado por el contenido del registro C. Luego, el registro B se decrementa y el registro par HL se incrementa.Si B=/0, el PC se decrementa en 2 y la instrucción se ejecuta de nuevo.

Formato: 1 1 1 0 1 1 0 1 ED Duración:B=0:4 ciclos;
1 0 1 1 0 0 1 1 BB 16 estados
B=/0:5 ciclos;21 estados

Direccionamiento: externo Banderas: S,Z,P/V;Z=1;N=1

OUT(C),r Sacar registro a puerto.

Función: (C) <- r

Descripción: El contenido del registro r es sacado al dispositivo periférico direccionado por el contenido del registro C. El registro C proporciona los bits A0 a A7 del bus de direcciones. El registro B proporciona los bits A8 a A15.

Formato: 1 1 1 0 1 1 0 1 ED Duración:3 ciclos;12 estados
0 1 R R R 0 0 1

Direccionamiento: externo Banderas:ninguna

OUT(N),A (OUT N) Sacar acumulador a puerto.

Función: (N) <- A

Descripción: El contenido del acumulador es sacada al

dispositivo periférico direccionado por el segundo byte de la instrucción.

Formato: 1 1 0 1 0 0 1 1 D3 Duración:3 ciclos;11 estados
direcc. puerto

Direccionamiento: externo Banderas:ninguna

OUTD Salida a puerto con decremento.

Función: (C) <- (HL);B <- B-1; HL <- HL-1

Descripción: El contenido de la localidad de memoria direccionada por el registro par HL es sacado al dispositivo periférico direccionado por el registro C. Luego el registro B y el registro par HL se decrementan.

Formato: 1 1 1 0 1 1 0 1 ED Duración:4 ciclos;16 estados
1 0 1 0 1 0 1 1 AB

Direccionamiento: externo Banderas:S,H,P/V;N=1;Z=1 si B=0
Z=0 si B=/0

OUTI Salida a puerto con incremento.

Función: (C) <- (HL);B <- B-1; HL <- HL+1

Descripción: El contenido de la localidad de memoria direccionada por el registro par HL es sacado all dispositivo periférico direccionado por el registro C. Luego el regista y
el registro par HL se incrementa.

Formato: 1 1 1 0 1 1 0 1 ED Duración:4 ciclos;16 estados
1 0 1 0 0 0 1 1 A3

Direccionamiento: externo Banderas:S,H,P/V;N=1;Z=1 si B=0
Z=0 si B=/0

POP pq (POP pq) Sacar registro par de la pila.

Función: qq bajo <- (SP);qq alto <- (SP+1);SP <- SP+2

Descripción: El contenido de la localidad de memoria direccionada por el apuntador de la pila (SP) se carga en el registro par qq y el SP se incrementa. El contenido de la localidad de memoria ahora direccionada por el nuevo valor del SP se carga en el registro alto del registro par qq y SP se vuelve a incrementar.

Formato: 1 1 Q Q 0 0 0 1 Duración:3 ciclos;10 estados

Direccionamiento: indirecto Banderas: ninguna

POP x Sacar registro índice de la pila.

Función: x bajo <- (SP);x alto <- (SP+1);SP <- SP+2

Descripción: El contenido de la localidad de memoria direccionada por SP se carga en la parte baja del registro índice x, el SP se incrementa El contenido de la localidad de memoria ahora direccionada por el nuevo valor del SP se carga en la parte alta del registro índice x, el SP se vuelve a incrementar.

Formato: 1 1 X 1 1 1 0 1 Duración:4 ciclos;14 estados
1 1 1 0 0 0 0 1 E1

Direccionamiento: indirecto Banderas: ninguna

PUSH qq (PUSH qq) Guardar registro par en la pila

Función: (SP-1) <- qq alto;(SP-2) <- qq bajo;SP <- SP-2

Descripción: El SP se decrementa y el contenido del registro alto del registro par qq se carga en la localidad de memoria direccionada por SP. El SP se decrementa de nuevo y el contenido del registro bajo del registro par qq se carga en la localidad de memoria direccionada ahora por SP.

Formato: 1 1 X 1 1 1 0 1 Duración:4 ciclos;15 estados
1 1 1 0 0 1 0 1 E5

Direccionamiento: indirecto Banderas: ninguna

RES b(HL) Poner bit de memoria en 0 lógico.

Función: (HL) <- 0

Descripción: El bit especificado de la localidad de memoria direccionada por el registro par HL se pone en 0 lógico.

Formato: 1 1 0 0 1 0 1 1 CB Duración:4 ciclos;15 estados
1 0 B B B 1 1 0

Direccionamiento: indirecto Banderas: ninguna

RES b(x+d) Poner bit de memoria indexada en 0 lógico.

Función: (x+d) <- 0

Descripción: El bit especificado de la localidad de memoria direccionada por x+d, donde x es el registro índice y d el valor del desplazamiento, se pone en

0 lógico.

Formato: 1 1 X 1 1 1 0 1 Duración:6 ciclos;23 estados
1 1 0 0 1 0 1 1 CB
desplazamiento
1 0 B B B 1 1 0

Direccionamiento: indexado Banderas: ninguna

RES b,r Poner bit de registro en 0 lógico.

Función: r <- 0

Descripción: El bit especificado del registro r se pone en 0 lógico.

Formato: 1 1 0 0 1 0 1 1 CB Duración:2 ciclos; 8 estados
1 0 B B B R R R

Direccionamiento: implícito Banderas: ninguna

RET Retorno de subrutina.

Función: PCL <- (SP); PCH <- (SP+1);SP <- SP+2

Descripción: El contenido de la localidad de memoria direccionada por SP se carga en la parte baja de PC y el SP se incrementa. El contenido de la localidad de memoria ahora direccionada por SP se carga en la parte alta del PC y el SP se incrementa de nuevo. La siguiente instrucción se obtiene de la localidad direccionada por el PC.

Formato: 1 1 0 0 1 0 0 1 C9 Duración:3 ciclos;10 estados

Direccionamiento: indirecto Banderas: ninguna

RET cc (Rcc) Retorno condicional de subrutina.

Función: si cc es verdadera:PCL<-(SP);PCH<-(SP+1);SP<-SP+2

Descripción: Si la condición especificada es verdadera, el contenido de la localidad de memoria se carga en la parte baja del PC y el SP se incrementa direccionada por SP. El contenido de la localidad de memoria ahora direccionada por SP se carga en la parte alta del PC y el SP se incrementa de nuevo. En caso contrario la ejecución continúa secuencialmente.

Formato: 1 1 C C C 0 0 0 Duración:condición verdadera
3 ciclos; 11 estados
condición falsa:
1 ciclo; 5 estados

Direccionamiento: indirecto Banderas: ninguna

RETI Retorno de interrupción.

Función: PCL <- (SP); PCH <- (SP+1);SP <- SP+2

Descripción: Ejecuta la misma función que RET. Sin embargo, esta instrucción es reconocida por los dispositivos periféricos de Zilog como el fin de una rutina de servicio iniciada por una interrupción. Se utiliza para permitir el control apropiada de interrupciones con prioridad anidada (nestes priority interrupts). Es necesario que se ejecute una instrucción EI antes de RETI para reactivar el mecanismo de interrupciones

Formato: 1 1 1 0 1 1 0 1 ED Duración:4 ciclos, 14 estados
0 1 0 0 1 1 0 1 4D

Direccionamiento: indirecto Banderas: ninguna

RETN Retorno de interrupción no inhibible

Función: PCL <- (SP); PCH <- (SP+1);SP <- SP+2

Descripción: Ejecuta la misma función que RET pero además restaura el estado de la bandera de interrupciones (IFF) a su posición anterior a la interrupción no inhibible (non-maskable interrupt).interrupciones.

Formato: 1 1 1 0 1 1 0 1 ED Duración:4 ciclos; 14 estados
0 1 0 0 0 1 0 1 45

Direccionamiento: indirecto Banderas: ninguna

RL(HL) Rotar memoria a la izquierda a través de la bandera de acarreo (cr).

Función: (HL) <- (HL); C <- (HL);(HL) <- C

Descripción: El contenido de la localidad de memoria direccionada por el registro par HL se rota una posición a la izquierda utilizando la bandera de acarreo (carry) como noveno bit. El bit más significativo se pasa a la bandera de acarreo (carry) y el contenido de la bandera de acarreo (carry) pasa a ocupar el lugar del bit menos significativo.

Formato: 1 1 0 0 1 0 1 1 CB Duración:4 ciclos; 15 estados
0 0 0 1 0 1 1 0 16

Direccionamiento: indirecto Banderas: S,Z,P/V,C;H=0;N=0

RL(x+d) Rotar memoria indexada a la izquierda a través del bit de acarreo (cr).

Función: (x+d) <- (x+d); C <- (x+d);(x+d) <- C

Descripción: El contenido de la localidad de memoria direccionada por x+d, donde x es el registro índice y d es el valor del desplazamiento, se rota una posición a la izquierda utilizando la bandera de acarreo (carry) como noveno bit. El bit más significativo se pasa a la bandera de acarreo (carry) y el contenido de la bandera de acarreo (carry) pasa a ocupar el lugar del bit menos significativo

Formato: ! ! X 1 1 1 0 1 Duración:6 ciclos; 23 estados
! ! 0 0 1 0 1 1 CB

Direccionamiento: indexado Banderas: S,Z,P/V,C;H=0;N=0

RL r Rotar registro a la izquierda a través de la bandera de acarreo (CY).

Función: r <- r; C <- r ;r <- C

Descripción: El contenido del registro r se rota una posición a la izquierda utilizando a la bandera de acarreo como noveno bit. El más significativo se pasa a la bandera de acarreo y su contenido pasa a ocupar el lugar del bit menos significativo.

Formato: 1 1 0 0 1 0 1 1 CB Duración:2 ciclos;8 estados
0 0 0 1 0 R R R

Direccionamiento: implícito Banderas: S,Z,P/V,C;H=0;N=0

RLA (RAL) Rotar acumulador a la izquierda a través de la bandera de acarreo (CY)

Función: A <- A; C <- A ;A <- C

Descripción: El contenido del acumulador se rota una posición a la izquierda utilizando a la bandera de acarreo como noveno bit. El más significativo se pasa a la bandera de acarreo y el contenido del carry pasa a ocupar el lugar del bit menos significativo.

Formato: 0 0 0 1 0 1 1 1 17 Duración:1 ciclo ;4 estados

Direccionamiento: implícito Banderas: C;H=0;N=0

RLCA (RLCA) Rotar acumulador a la izquierda.

Función: A <- A; A <- A ;C <- A

Descripción: El contenido del acumulador se rota una posición a la izquierda. El contenido del bit más significativo pasa al bit menos significativo y a la bandera de acarreo.

Formato: 0 0 0 0 0 1 1 1 07 Duración:1 ciclo ;4 estados

Direccionamiento: implicito Banderas: C;H=0;N=0

RLC r Rotar registro a la izquierda.

Función: r <- r; r <- r ;C <- r

Descripción: El contenido del registro r se rota una posición a la izquierda. El contenido del bit más significativo pasa al bit menos significativo y a la bandera de acarreo.

Formato: 1 1 0 0 1 0 1 1 CB Duración:2 ciclo ;8 estados
0 0 0 0 0 R R R

Direccionamiento: implicito Banderas: S,Z,P/V,C;H=0;N=0

RLC (HL) Rotar memoria a la izquierda.

Función: (HL) <- (HL);(HL) <- (HL);C <- (HL)

Descripción: El contenido de la localidad de memoria direccionada por el registro par HL se rota una posición a la izquierda. El contenido del bit más significativo pasa al bit menos significativo y a la bandera de acarreo.

Formato: 1 1 0 0 1 0 1 1 CB Duración:2 ciclo ;8 estados
0 0 0 0 0 1 1 0 06

Direccionamiento: indirecto Banderas: S,Z,P/V,C;H=0;N=0

RLC (x+d) Rotar memoria indexada a la izquierda.

Función: (x+d) <- (x+d);(x+d) <- (x+d);C <- (x+d)

Descripción: El contenido de la localidad de memoria direccionada por x+d, donde x es el registro indice y d es el valor del desplazamiento se rota una posición a la izquierda. El contenido del bit más significativo pasa al bit menos significativo y a la bandera de acarreo.

Formato: 1 1 X 1 1 1 0 1 Duración:6 ciclo;23 estados
1 1 0 0 1 0 1 1 CB
desplazamiento
0 0 0 0 0 1 1 0 06

Direccionamiento: indexado Banderas: S,Z,P/V,C;H=0;N=0

RLD Rotar digito decimal a la izquierda.

Función: (HL) <- (HL);A <- (HL);(HL) <- A

Descripción: Los 4 bits menos significatios de la localidad de memoria direccionada por el registro par HL se mueven a los 4 bits más significativos de la misma localidad. Los 4 bits más significativos se mueven a los 4 bits menos significativos del acumulador. Los 4 bits menos significativos del acumulador se mueven a los 4 bits menos significativos de la localidad de memoria especificada. Todas estas operaciones ocurren simultáneamente.

Formato: 1 1 1 0 1 1 0 1 ED Duración:5 ciclo;18 estados
0 1 1 0 1 1 1 1 6F

Direccionamiento: indirecto Banderas: S,Z,P/V,C;H=0;N=0

RR(HL) Rotar memoria a la derecha a través de la bandera de acarreo (CY).

Función: (HL) <- (HL);C <- (HL);(HL) <- C

Descripción: El contenido de la localidad de memoria direccionada por el registro par HL se rota una posición a la derecha utilizando a la bandera de acarreo como noveno bit. El bit menos significativo se pasa y su contenido a la bandera de acarreo pasa a ocupar el lugar del bit más significativo.

Formato: 1 1 0 0 1 0 1 1 CB Duración:5 ciclo;18 estados
0 0 0 1 1 1 1 0 1E

Direccionamiento: indirecto Banderas: S,Z,P/V,C;H=0;N=0

RR(x+d) Rotar memoria indexada a la derecha a través de la bandera de acarreo (CY).

Función: (x+d) <- (x+d);C <- (x+d);(x+d) <- C

Descripción: El contenido de la localidad de memoria direccionada por x+d, donde x es el registro índice y d es el valor del desplazamiento, se rota una posición a la derecha utilizando a la bandera de acarreo como noveno bit. El bit menos significativo se pasa y su contenido a la bandera de acarreo pasa a ocupar el lugar del bit más significativo.

Formato: 1 1 X 1 1 1 0 1 Duración:6 ciclo;23 estados

1 1 0 0 1 0 1 1 CB
desplazamiento
0 0 0 1 1 1 1 0 1E

Direccionamiento: indexado Banderas: S,Z,P/V,C;H=0;N=0

RR r Rotar registro a la derecha a través de la bandera de acarreo (CY).

Función: r <- r;C <- r;r <- C

Descripción: El contenido del registro r se rota una posición a la derecha utilizando a la bandera de acareo como noveno bit. El bit menos significativo se pasa a la bandera de acarreo y su contenido pasa a ocupar el lugar del bit más significativo.

Formato: 1 1 0 0 1 0 1 1 CB Duración:2 ciclo; 8 estados
0 0 0 1 1 R R R

Direccionamiento: implicito Banderas: S,Z,P/V,C;H=0;N=0

RRA (RAR) Rotar acumulador a la derecha a través de la bandera de acarreo (CY).

Función: A <- A;C <- A;A <- C

Descripción: El contenido del acumulador se rota una posición a la derecha utilizando a la bandera de acareo como noveno bit. El bit menos significativo se pasa a la bandera de acarreo y su contenido pasa a ocupar el lugar del bit más significativo.

Formato: 0 0 0 1 1 1 1 1 1F Duración:1 ciclo; 4 estados

Direccionamiento: implicito Banderas: C;H=0;N=0

RRCA (RCC) Rotar acumulador a la derecha.

Función: A <- A;C <- A;C <- A

Descripción: El contenido del acumulador se rota una posición a la derecha. El contenido del bit menos significativo pasa al bit más significativo y a la bandera de acarreo.

Formato: 0 0 0 0 1 1 1 1 0F Duración:1 ciclo; 4 estados

Direccionamiento: implicito Banderas: C;H=0;N=0

RRC (HL) Rotar memoria a la derecha.

Función: (HL) <- (HL);(HL) <- (HL);C <- (HL)

Descripción: El contenido de la localidad de memoria

direccionada por el registro par HL se rota una posición a la derecha. El contenido del bit menos significativo pasa al bit más significativo y a la bandera de acarreo

Formato: 1 1 0 0 1 0 1 1 CB Duración:4 ciclo; 5 estados
0 0 0 0 1 1 1 0 0E

Direccionamiento: implicito Banderas: C;H=0;N=0

RRC(x+d) Rotar memoria indexada a la derecha.

Función: (x+d) <- (x+d);(x+d) <- (x+d);C <-(x+d)

Descripción: El contenido de la localidad de memoria direccionada por x+d, donde x es el registro indice y d es el valor del desplazamiento, se rota una posición a la derecha. El contenido del bit menos significativo pasa al bit más significativo y a la bandera de acarreo.

Formato: 1 1 X 1 1 1 0 1 Duración:6 ciclo;23 estados
1 1 0 0 1 0 1 1 CB
desplazamiento
0 0 0 0 1 1 1 0 0E

Direccionamiento: indexado Banderas: S,Z,P/V,C;H=0;N=0

RRC r Rotar registro la derecha.

Función: r <- r;r <- r;C <- r

Descripción: El contenido del registro r se rota una posicón a la derecha. El contenido del bit menos significativo pasa al bit más significativo y a la bandera de acarreo (CY).

Formato: 1 1 0 0 1 0 1 1 CB Duración:2 ciclo; 8 estados
0 0 0 0 1 R R R

Direccionamiento: implicito Banderas: S,Z,P/V,C;H=0;N=0

RRD Rotar digito decimal a la dercha.

Función: A <- (HL);(HL) <- A;(HL) <- (HL)

Descripción: Los 4 bits más significativos de la localidad de memoria direccionada por el registro par HL se mueven a los 4 bits menos significativos de la misma localidad. Los 4 bits menos significativos se mueven a los 4 bits menos significativos del acumulador. Los 4 bits menos significativos del acumuladorse mueven a los 4 bits más significativos de la localidad de memoria especificada.

Todas estas operaciones ocurren simultáneamente.

Formato: 1 1 1 0 1 1 0 1 ED Duración:5 ciclo,18 estados
0 1 1 0 0 1 1 1 67

Direccionamiento: indirecto Banderas: S,Z,P/V,C;H=0;N=0

RST p (RST p) Restablecimiento.

Función: (SP-1)<-PCH;(SP-2)<-PCL;SP<-SP-2;PCH<-0;PCL<-p

Descripción: El contenido del PC se almacena en la pila de la manera descrita en la instrucción PUSH. Luego, el valor del p se carga en los 8 bits menos significativos del PC. Los 8 bits más significativos del PC se ponen en ceros. La siguiente instrucción se obtiene de la localidad direccionada por el nuevo valor del PC. Esta instrucción efectúa un salto a cualquiera de los 8 direcciones en la parte baja de la memoria y requiere de un byte únicamente. Se puede usar para responder rápidamente a una interrupción.

Formato: 1 1 P P P 1 1 1 Duración:3 ciclo;11 estados

Direccionamiento: indirecto Banderas: ninguna

SBC A,(HL) (SBB M) Restar memoria con préstamo.

Función: A <- A -(HL)-C

Descripción: El contenido de la localidad de memoria direccionada por el registro par HL y la bandera de acarreo se restan del acumulador. El resultado queda en el acumulador.

Formato: 1 0 0 1 1 1 1 0 9E Duración:2 ciclos; 7 estados

Direccionamiento: indirecto Banderas: S,Z,H,P/V,C;N=1

SBC A,(x+d) Restar memoria indexada con préstamo

Función: A <- A -(x+d)-C

Descripción: El contenido de la localidad de memoria direccionada por x+d, donde x es el registro indice y d es el valor del desplazamiento, y la bandera acarreo se restan del acumulador. El resultado queda en el acumulador.

Formato: 1 1 X 1 1 1 0 1 Duración:5 ciclos;19 estados
1 0 0 1 1 1 1 0 9E
desplazamiento

Direccionamiento: indexado Banderas: S,Z,H,P/V,C;N=1

SBC A,n (SBI n) Restar dato inmediato con préstamo.

Función: A <- A -n-C

Descripción: El contenido del segundo byte de la instrucción y la bandera de acarreo se restan del acumulador. El resultado queda en el acumulador.

Formato: 1 1 0 1 1 1 1 0 DE Duración:2 ciclos; 7 estados
dato

Direccionamiento: indexado Banderas: S,Z,H,P/V,C;N=1

SBC A,r (SBR r) Restar registro con préstamo.

Función: A <- A -r-C

Descripción: El contenido del registro r y la bandera de acarreo se restan del acumulador. El resultado queda en el acumulador.

Formato: 1 0 0 1 1 R R R DE Duración:1 ciclo; 4 estados

Direccionamiento: implicito Banderas: S,Z,H,P/V,C;N=1

SBC HL,ss Restar registro con préstamo del registro par HL.

Función: HL <- HL -ss-C

Descripción: El contenido del registro de 16 bits ss y la bandera de acarreo se restan del registro par HL. El resultado queda en el registro par HL.

Formato: 1 1 1 0 1 1 0 1 ED Duración:4 ciclos; 15 estados
0 1 S S 0 0 1 0

Direccionamiento: implicito Banderas: S,Z,H,P/V,C;N=1
H=1 si hay préstamo del bit 12

SCF (STC) Poner bandera de acarreo en 1 lógico.

Función: C <- 1

Descripción: Se pone la bandera de acarreo en 1 lógico.

Formato: 0 0 1 1 0 1 1 1 37 Duración:1 ciclo; 4 estados

Direccionamiento: implicito Banderas: H=0;N=0;C=1

SET b,(HL) Poner bit de memoria en 1 lógico.

Función: (HL) <- 1

Descripción: El bit especificado de la localidad de memoria direccionada por el registro par HL se pone en 1 lógico.

Formato: 1 1 0 0 1 0 1 1 CB Duración:4 ciclos;15 estados
1 1 B B B 1 1 0

Direccionamiento: indirecto Banderas: ninguna

SET b,(x+d) Poner bit de memoria indexada en 1 lógico.

Función: (x+d) <- 1

Descripción: El bit especificado de la localidad de memoria direccionada por x+d, donde x es el registro indice y d es el valor del desplazamiento, se pone en 1 lógico.

Formato: 1 1 X 1 1 1 0 1 Duración:6 ciclos;23 estados
1 1 0 0 1 0 1 1 CB
desplazamiento
1 1 B B B 1 1 0

Direccionamiento: indexado Banderas: ninguna

SET b,r Poner bit de registro en 1 lógico.

Función: r <- 1

Descripción: El bit especificado del registro r se pone en 1 lógico.

Formato: 1 0 1 1 0 0 1 1 C3 Duración:2 ciclos; 8 estados
1 1 B B B R R R

Direccionamiento: implicito Banderas: ninguna

SLA (HL) Desplazar memoria a la izquierda aritméticamente.

Función: (HL) <- (HL);(HL) <- 0;C <- (HL)

Descripción: El contenido de la localidad de memoria direccionada por el registro par HL se desplaza una posición a la izquierda. El bit más significativo pasa al de acarreo y en el bit menos significativo se carga un cero.

Formato: 1 0 1 1 1 0 1 1 CB Duración:4 ciclos;15 estados
0 0 1 0 0 1 1 0 26

Direccionamiento: indirecto Banderas: S,Z,P/V,C;H=0;N=0

SLA (x+d) Desplazar memoria indexada a la izquierda aritméticamente.

Función: (x+d) <- (x+d);(x+d) <- 0;C <- (x+d)

Descripción: El contenido de la localidad de memoria direccionada por x+d, donde x es el registro indice y d es el valor del desplazamiento, se desplaza una posición a la izquierda. El bit más significativo pasa al de acarreo y en el bit menos significativo se carga un cero.

Formato: 1 1 X 1 1 1 0 1 Duración:6 ciclos;23 estados
1 1 0 0 1 0 1 1 CB
desplazamiento
0 0 1 0 0 1 1 0 26

Direccionamiento: indirecto Banderas: S,Z,P/V,C;H=0;N=0

SLA r Desplazar registro a la izquierda aritméticamente.

Función: r <- r;r <- 0;C <- r

Descripción: El contenido del registro r se desplaza una posición a la izquierda. El bit más significativo pasa al de acarreo y en el bit menos significativo se carga un cero.

Formato: 1 1 0 0 1 0 1 1 CB Duración:2 ciclos; 8 estados
0 0 1 0 0 R R R

Direccionamiento: implicito Banderas: S,Z,P/V,C;H=0;N=0

SRA (HL) Desplazar memoria a la derecha aritméticamente.

Función: (HL) <- (HL);(HL) <- (HL);C <- (HL)

Descripción: El contenido de la localidad de memoria direccionada por el registro par HL se desplaza una posición a la derecha. El bit menos significativo pasa a la bandera de acarreo (CY). El bit más significativo no cambia.

Formato: 1 1 0 0 1 0 1 1 CB Duración:4 ciclos;15 estados
0 0 1 0 1 1 1 0 2E

Direccionamiento: indirecto Banderas: S,Z,P/V,C;H=0;N=0

SRA (x+d) Desplazar memoria indexada a la derecha aritméticamente.

Función: (x+d) <- (x+d);(x+d) <- (x+d);C <- (x+d)

Descripción: El contenido de la localidad de memoria direccionada por x+d, donde x es el registro indice y d es el valor del desplazamiento, se desplaza una posición a la derecha. El bit menos significativo pasa a la bandera de acarreo (CY).

El bit más significativo no cambia.

Formato: 1 1 X 1 1 1 0 1 Duración:6 ciclos;23 estados
1 1 0 0 1 0 1 1 CB
desplazamiento
0 0 1 0 1 1 1 0

Direccionamiento: indexado Banderas: S,Z,P/V,C;H=0;N=0

SRA r Desplazar registro a la derecha aritméticamente

Función: r <- r;r <- r;C <- r

Descripción: El contenido del registro r se desplaza una posición a la derecha. El bit menos significativo pasa a la bandera de acarreo (CY). El mit más significativo no cambia.

Formato: 1 1 0 0 1 0 1 1 CB Duración:2 ciclos; 8 estados
0 0 1 0 1 R R R

Direccionamiento: implícito Banderas: S,Z,P/V,C;H=0;N=0

SRL (HL) Desplazar memoria a la derecha lógicamente.

Función: (HL) <- (HL);(HL) <- 0;C <- (HL)

Descripción: El contenido de la localidad de memoria direccionada por el registro par HL se desplaza una posición a la derecha. El bit menos significativo pasa al de acarreo y en el bit más significativo se carga un cero.

Formato: 1 1 0 0 1 0 1 1 CB Duración:2 ciclos; 8 estados
0 0 1 1 1 1 1 0 3E

Direccionamiento: indirecto Banderas: S,Z,P/V,C;H=0;N=0

SRL (x+d) Desplazar memoria indexada a la dercha lógicamente.

Función: (x+d) <- (x+d);(x+d) <- 0;C <- (x+d)

Descripción: El contenido de la localidad de memoria direccionada por x+d, donde x es el registro indice y d es el valor del desplazamiento, se desplaza una posición a la derecha. El bit menos significativo pasa al de acarreo y en el bit más significativo se carga un cero.

Formato: 1 1 X 1 1 1 0 1 Duración:6 ciclos;23 estados
1 1 0 0 1 0 1 1 CB
desplazamiento
0 0 1 1 1 1 1 0 3E

Direccionamiento: indexado Banderas: S,Z,P/V,C;H=0;N=0

SRL r Desplazar registro a la dercha lògicamente

Función: r <- r;r <- 0;C <- r

Descripción: El contenido del registro r se desplaza una posición a la derecha. El bit menos significativo pasa al de acarreo y en el bit más significativo se carga un cero.

Formato: 1 1 0 0 1 0 1 1 CB Duración:2 ciclos; 8 estados
0 0 1 1 1 R R R

Direccionamiento: implicito Banderas: S,Z,P/V,C;H=0;N=0

SUB (HL) (SUB M) Restar memoria.

Función: A <- A-(HL)

Descripción: El contenido de la localidad de memoria direccionada por el registro par HL se resta del acumulador. El resultado queda en el acumulador.

Formato: 1 0 0 1 0 1 1 0 96 Duración:2 ciclos; 7 estados

Direccionamiento: indirecto Banderas: S,Z,P/V,C;H=0;N=0

SUB (x+d) Restar memoria indexada.

Función: A <- A-(x+d)

Descripción: El contenido de la localidad de memoria direccionada por x+d, donde x es el registro índice y d es el valor del desplazamiento, se resta del acumulador. El resultado queda en el acumulador.

Formato: 1 1 X 1 1 1 0 1 Duración:5 ciclos;19 estados
1 0 0 1 0 1 1 0
desplazamiento

Direccionamiento: indexado Banderas: S,Z,P/V,C;N=1

SUB n (SUI n) Restar dato inmediato.

Función: A <- A-n

Descripción: El contenido del segundo byte de la instrucción se resta del acumulador. El resultado queda en el acumulador.

Formato: 1 1 0 1 0 1 1 0 D6 Duración:2 ciclos; 7 estados
dato

Direccionamiento: inmediato Banderas: S,Z,P/V,C;N=1

XOR (HL) (XRA M) XOR memoria.

Función: A <- A V(HL)

Descripción: El contenido de la localidad de memoria direccionadaa por el registro par HL se pasa a través de OR exclosuvas junto con el acumulador. El resultado queda en el acumulador.

Formato: 1 0 1 0 1 1 1 0 AE Duración:2 ciclos; 7 estados

Direccionamiento: indirecto Banderas: S,Z,P/V,H=0;N=0;C=0

XOR (x+d) XOR memoria indexada.

Función: A <- A V(x+d)

Descripción: El contenido de la localidad de memoria direccionada por x+d, donde x es el registro indice y d es el valor del desplazamiento, se pasa a través de OR exclusivas junto con el acumulador. El resultado queda en el acumulador.

Formato: 1 1 X 1 1 1 0 1 Duración:5 ciclos;19 estados
1 0 1 0 1 1 1 0 AE
desplazamiento

Direccionamiento: indexado Banderas: S,Z,P/V,H=0;N=0;C=0

XOR n (XRI n) XOR dato inmediato.

Función: A <- A Vn

Descripción: El contenido del segundo byte de la instrucción se pasa a través de OR exclusivas junto con el acumulador. El resultado queda en el acumulador.

Formato: 1 1 1 0 1 1 1 0 EE Duración:2 ciclos; 7 estados
dato

Direccionamiento: inmediato Banderas: S,Z,P/V,H=0;N=0;C=0

XOR r (XRA r) XOR registro.

Función: A <- A Vr

Descripción: El contenido del registro r se pasa a través de OR exclusivas junto con el acumulador. El resultado queda en el acumulador.

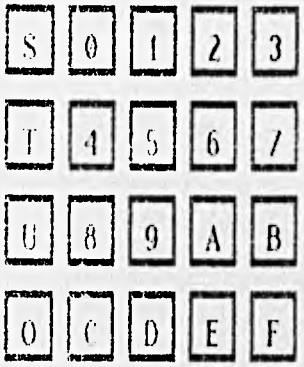
Formato: 1 0 1 0 1 R R R Duración:1 ciclo; 4 estados

Direccionamiento: implicito Banderas: S,Z,P/V,H=0;N=0;C=0

APENDICE C

Manual del usuario

TECLADO



S [SALE] ——→ Para correr programas
T [COM1] ——→ Comando 1
U [COM2] ——→ Comando 2
O [OTRO] ——→ Ejecutar (Enter)

0,1,2,3
4,5,6,7
8,9,A,B
C,D,E,F
——→ Sistema numerico Hexadecimal

Una vez que se ha desplegado el mensaje HOLA en el sistema, estamos listos para probar el funcionamiento del teclado.

Oprimir las teclas como se indica a continuacion:

TECLAS	DESPLEGADO
COM1 (T)	C O 1
MEM (0)	d i r
0	0 0 0 0
1	0 0 0 1
2	0 0 1 2
3	0 1 2 3
4	1 2 3 4
5	2 3 4 5
6	3 4 5 6
7	4 5 6 7
8	5 6 7 8
9	6 7 8 9
A	7 8 9 A
B	8 9 A B
C	9 A B C
D	A B C D
E	B C D E
F	C D E F
OTRO (O)	C D E F X X

El dato X X representa cualquier valor impredecible.

Continuamos con las teclas:

TECLAS	DESPLEGADO
SALE (S)	E r r
COM2 (U)	C O 2
DAR (4)	7 A X X

Y quedara desplegando los registros internos del procesador con sus contenidos hasta que la tecla INICIO sea pulsada.

INICIO (R)	H O L A

Con esto quedan probadas todas las teclas del sistema. Como observacion, los valores o letras que estan entre parentesis, indica la tecla a oprimir en el teclado. Se puede dar el caso de que una misma tecla se utilice como comando o como valor, esto quedara claro en los ejemplos siguientes.

Instrucciones de operacion

Los comandos en el monitor estan divididos en dos clases:

Comandos 1 Con 1

Comandos 2 Con 2

Los comandos 1 son quellos que permiten al usuario el manejo del sistema como una microcomputadora, es decir, mediante estos comandos es posible tener acceso a la memoria, a los registros y a los puertos, para examinar y/o modificar su contenido.

Los comandos 2 nos permiten hacer uso de programas utiles que facilitan nuestro trabajo, y estos son; calcular el valor de un desplazamiento para saltos relativos, obteniendo el resultado tanto hacia atras como hacia adelante, mover bloques de memoria de una area a otra, comparar dos areas de memoria, solicitar desplegado automatico de memoria o de registros, etc.

Que de aqui en adelante los veremos como se ilustra:

CLASE DE COMANDO	TIPO DE COMANDO
CO 1	MEMORIA
	REGISTROS
	ENT/SAL
CO 2	DESPLAMTO. RELATIVO
	MOVER MEMORIA
	COMPARAR MEMORIA
	DESPLEGADO DE MEMORIA
	DESPLEGADO DE REGISTROS

OPERACION

1. Indicar que clase de comando deseamos; CO 1 o CO 2.
2. Indicar que tipo de comando deseamos; MEM, REG, E/S para CO 1, o CDR, MBM, CM, DAM, DAR para CO 2.
3. Introducir los datos requeridos.
4. Ordenar que se ejecute el comando; OTRO (O).

EJEMPLOS

EXAMINAR LOCALIDADES DE MEMORIA

TECLA	DESPLEGADO
INICIO (R)	H O L A
CO 1 (T)	C O 1
MEM	d i r
1	0 0 0 1
4	0 0 1 4
0	0 1 4 0
0	1 4 0 0 x x

El contenido de la localidad 1400 (RAM) que aparece a la derecha es impredecible (aleatorio) cuando la energia es aplicada al sistema; pero una vez que esta alimentado no cambia a menos que nosotros lo modifiquemos.

OTRO (O)	1 4 0 1 Y Y
OTRO (O)	1 4 0 2 Z Z
CO 1 (T)	H O L A

MODIFICAR LOCALIDADES DE MEMORIA

TECLA	DESPLEGADO	
COM 1 (T)	C O '	
MEM (0)	d i r	
1	0 0 0 1	
6	0 0 1 6	
C	0 1 6 C	
8	1 6 C 8	
OTRO (0)	1 6 C 6	X X
5	1 6 C 8	X 5
A	1 6 C 8	5 A
OTRO (0)	1 6 C 9	X X
COM 1 (T)	H O L A	

Para verificar que la localidad fue modificada, examinarla nuevamente.

INSERTAR PROGRAMAS

TECLA	DESPLEGADO	
COM 1 (T)	C O '	
MEM (0)	d i r	
1	0 0 0 1	
4	0 0 1 4	
0	0 1 4 0	
0	1 4 0 0	
OTRO (0)	1 4 0 0	X X
A	1 4 0 0	X A
F	1 4 0 0	A F
OTRO (0)	1 4 0 1	Y Y
C	1 4 0 1	Y C
F	1 4 0 1	C F
OTRO (0)	1 4 0 2	Z Z
COM 1 (T)	H O L A	

El programa que se inserto fue:

LOC	CODIGO	ETIQUETA	MNEMONICO	COMENTARIOS
1400	3E	CERO	MVI A,	;A <- 00
1401	CF		RST 08	;Brinca al monitor

EXAMINAR REGISTROS INTERNOS

TECLA	DESPLEGADO	
COM 1 (T)	C O ' 1	
REG (1)	r	
AF	r A	
OTRO (0)	r A	X X
OTRO (0)	r F	Y Y
OTRO (0)	r b	Z Z
COM 1 (T)	H O L A	

OTRO EJEMPLO:

TECLA	DESPLEGADO	
COM 1 (T)	C O ' 1	
REG (1)	r	
COM 1 (T)	r P H	
OTRO (0)	r P H	U U
OTRO (0)	r P P	X X
OTRO (0)	r A'	Y Y
OTRO (0)	r F'	Z Z
COM 1 (T)	H O L A	

MODIFICAR REGISTROS INTERNOS

TECLA	DESPLEGADO	
COM 1 (T)	C O ' 1	
REG (1)	r	
SP (A)	r S H	
OTRO (0)	r S H	X X
1	r S H	- 1
6	r S H	1 6
OTRO (0)	r S L	Y Y
F	r S L	- F
D	r S L	F D
OTRO (0)	r S H	U U
COM 1 (T)	H O L A	

EXAMINAR PUERTOS

TECLA	DESPLEGADO	
COM 1 (T)	C O 1	
E/S (2)	P E	
0	P E - 0	
4	P E 0 4	
OTRO (O)	P E 0 4	X X
OTRO (O)	P E 0 5	X X
COM 1 (T)	H O L A	

MODIFICAR PUERTOS DE SALIDA

TECLA	DESPLEGADO	
COM 1 (T)	C O 1	
E/S (2)	P E	
0	P E - 0	
7	P E 0 7	
OTRO (O)	P E 0 7	X X
8	P E 0 7	- 8
0	P E 0 7	8 0
OTRO (O)	P E 0 7	8 0

Aqui fue enviada una palabra de control (80H) al control de los puertos 4,5 y 6 para programar todos los puertos como de salida. Vease " Direccinamiento y control del PIO ", en el capitulo III.

COM 1 (T)	H O L A	

Ahora sacaremos un 55H por el puerto 05

COM 1 (T)	C O 1	
E/S (2)	P E	
0	P E - 0	
5	P E 0 5	
OTRO (O)	P E 0 7	X X
5	P E 0 5	- 5
5	P E 0 5	5 5
OTRO (O)	P E 0 7	8 0
COM 1 (T)	H O L A	

CORRER PROGRAMAS

En la localidad 1400 insertamos ya un programa (ver INSERTAR PROGRAMAS) que coloca un 00 en el Acumulador (A).

TECLA	DESPLEGADO
SALE (S)	S A L E
1	0 0 0 1
4	0 0 1 4
0	0 1 4 0
0	1 4 0 0
OTRO (O)	E
	H O L A

La E que aparece en el despiegado queda prendida durante el tiempo de ejecucion del programa, pero con este es muy rapido no alcanzamos a ver este mensaje de Ejecucion; solamente podemos ver el mensaje HOLA.

Si se desea confirmar que el programa corrio, bastara con examinar el registro A y encontraremos un 00.

Algo muy importante es que si se desean analizar los resultados de un programa es necesario que dicho programa finalice con un RST 08 que nos envia al monitor, entrando por un punto de interrupcion por programa que recupera y guarda todos los registros de la CPU para que nosotros podamos analizarlos posteriormente.

Si se desea correr un programa PASO A PASO basta con intercalr un RST 08 en cada lugar que deseamos que el programa se interrumpa para poder analizar los registros, la memoria o los puertos.

EJEMPLO DE PROGRAMA PASO A PASO

TECLA	DESPLEGADO	
COM 1 (T)	C O 1	
MEM (0)	d i r	
1	0 0 0 1	
4	0 0 1 4	
0	0 1 4 0	
0	1 4 0 0	
OTRO (0)	1 4 0 0	X X
3	1 4 0 0	X 3
E	1 4 0 0	3 E
OTRO (0)	1 4 0 1	Y Y
1	1 4 0 1	X 1
2	1 4 0 1	1 2
OTRO (0)	1 4 0 2	X X
C	1 4 0 2	X C
F	1 4 0 2	C F
OTRO (0)	1 4 0 3	X X
0	1 4 0 3	X 0
6	1 4 0 3	0 6
OTRO (0)	1 4 0 4	X X
2	1 4 0 4	X 2
3	1 4 0 4	2 3
OTRO (0)	1 4 0 5	X X
C	1 4 0 5	X C
F	1 4 0 5	C F
OTRO (0)	1 4 0 6	X X
8	1 4 0 6	X 8
0	1 4 0 6	8 0
OTRO (0)	1 4 0 7	X X
C	1 4 0 7	X C
F	1 4 0 7	C F
OTRO (0)	1 4 0 8	X X
COM 1 (T)	H O L A	

Hasta aqui esta insertado el programa que coloca un 12H en A, un 23H en B y los suma, dejando el resultado en A.

Para correrlo:

TECLA	DESPLEGADO
SALE (S)	S A L E
1	0 0 0 1
1	0 0 1 4
0	0 1 4 0
0	1 4 0 0
OTRO (0)	E
	H O L A

En este momento solo se ha corrido la primer primer instruccion y vale la pena examinar los registros para ver si el programa va haciendo lo que debe:

Para continuar

TECLA	DESPLEGADO
SALE (S)	S A L E
OTRO (0)	H O L A

Examinar B (registro B=23)

Continuar

TECLA	DESPLEGADO
SALE (S)	S A L E
OTRO (0)	H O L A

Ya debio hacer la suma, y se puede examinar todos los registros involucrados en el programa.

INICIALIZAR EL SISTEMA

TECLA	DESPLEGADO
R	H O L A

Esta tecla no aparece en el teclado, se aisló para evitar su accionamiento accidental. Al pulsar la tecla R (RESET) se da un RESET general al microprocesador:

- Los puertos E/S se inicializan, queda sin definir el opcional (entradas y salidas en tercer estado).
- El contador de programa (PC) se coloca en ceros y se inicia la ejecución del programa monitor HOLA.
- Ninguno de los registros internos del procesador que utiliza el usuario son alterados (22 registros).

CALCULO DE DESPLAZAMIENTOS RELATIVOS

TECLA	DESPLEGADO
COM 2 (U)	C O 2
COR (0)	c d r
1	- 0 0 1
0	- 0 1 0
OTRO (0)	- F 6 0 A

Donde el guion junto al primer resultado indica que se trata del calculo de desplazamiento relativo hacia atras y el segundo resultado es hacia hacia adelante.

El COR acepta solo numeros decimales, los convierte a hexadecimales (segundo resultado, 0A) y les saca su complemento a dos (primer resultado, F6).

Una vez desplegado el resultado del COR, el comando esta terminado y el programa se encuentra en la RUTINA DE RECONOCIMIENTO DE COMANDOS, por lo que el sistema esta listo para aceptar otro comando.

MOVER BLOQUES DE MEMORIA

TECLA	DESPLEGADO	
COM 2 (U)	C O 2	
MBM (1)	b l o	F
Hay que indicar la fuente		
0	0 0 0 0	f
OTRO (0)		f
Hay que indicar la cantidad de bytes a mover		
2	0 0 0 2	C
0	0 0 2 0	C
OTRO (0)		d
Indicar el destino del bloque a mover		
1	0 0 0 1	d
4	0 0 1 4	d
0	0 1 4 0	d
0	1 4 0 0	d
OTRO (0)	H O L A	

El bloque fue ya trasladado de la localidad 0000H a la localidad 1400H y consta de 20H bytes (esto fue solo un ejemplo).

COMPARAR DOS AREAS DE MEMORIA

TECLA	DESPLEGADO	
COM 2 (U)	C O 2	
CM	C O	F
0	0 0 0 0	F
OTRO (0)		C
2	0 0 0 2	C
0	0 0 2 0	C
OTRO (0)		d
1	0 0 0 1	d
4	0 0 1 4	d
0	0 1 4 0	d
0	1 4 0 0	d
OTRO (0)	X X X X	X X

Si las dos areas de memoria tienen el mismo contenido, se desplegara el mensaje HOLA, pero si hay diferencias, se desplegara la localidad de la fuente con su contenido primero, seguida por la localidadde destino con su contenido.

En el caso de ser varias las localidades distintas, apareceran en el desplegado de una en una, siempre primero la fuente seguida por el destino.

Si desea detenerse el desplegado para ver con calma tanto localidad como contenido, basta con oprimir cualquier tecla, y para continuar, repetir la operacion.

Si se desea ver localidad por localidad oprimir dos veces seguidas cualquier tecla cada vez.

DESPLEGADO AUTOMATICO DE MEMORIA

TECLA	DESPLEGADO	
COM 2 (U)	C 0 2	
DAM (3)		
1	0 0 0 1	C
0	0 0 1 0	C
OTRO (O)		d
0	0 0 0 0	d
OTRO (O)	0 0 0 0	C 3

Al oprimir la tecla OTRO (O) se inicia el desplegado automatico de las localidades de memoria a partir de la que se indico como destino.

Para detener el DAM, pulsar cualquier tecla y para continuar, hacer lo mismo.

Aqui son iguales los comandos CM, DAM y DAR.

DESPLEGADO AUTOMATICO DE REGISTROS

TECLA	DESPLEGADO
COM 2 (U)	C 0 2
DAR (4)	r A X X

A partir de aqui desplegara los registros uno a uno con su contenido en forma ciclica.

Para detener y continuar con el DAR actuar como en caso de CM y DAM.

Para abandonar este comando pulsar la tecla R (RESET).

PRACTICAS

PRACTICA 1.

Los objetivos de esta practica consisten en lograr que el alumno sepa como realizar lo siguiente:

- Restablecer el microprocesador.
- Examinar el contenido de una memoria.
- Cambiar el contenido de una memoria.
- Transferir al acumulador el contenido de una memoria y viceversa.
- Complementar el contenido del acumulador.
- Correr y detener un programa.

1.1 Complemento a uno de un dato con longitud de un byte.

LOCALIDAD	CODIGO DE MAQUINA	ETIQUETA	MNEMONICO	COMENTARIOS
1410	3E 70		LD A,N	A <- N (70)
1412	2F		CPL	Complemento a uno de A
1413	32		LD (1416),A	Carga A en la localidad de memoria (1416)
1415	CF		RST 08H'	Regreso al monitor

La forma de cargar este programa es tecleando: R,COM1(T),MEM(0),1410,3E70; MEM(0),1412,2F;MEM(0),1413,32;MEM(0),1415,CF;SALE(S),1410

1.2 Complemento a dos de un dato con longitud de un byte.

LOCALIDAD	CODIGO DE MAQUINA	ETIQUETA	MNEMONICO	COMENTARIOS
1410	3E 70		LD A,N	A <- N (70)
1412	ED44		NEG	Complemento a dos de A
1414	32		LD (1416),A	Carga A en la localidad de memoria (1416)
1416	CF		RST 08H'	Regreso al monitor

La forma de cargar este programa es tecleando: R,COM1(T),MEM(0),1410,3E70; MEM(0),1412,ED44;MEM(0),1414,32;MEM(0),1416,CF;SALE(S),1410

1.3(a) Transferir un dato de una localidad de memoria a otra.

LOCALIDAD	CODIGO DE MAQUINA	ETIQUETA	MNEMONICO	COMENTARIOS
1410	3E 70		LD A,N	A <- N (70)
1412	32		LD (1416),A	Carga A en la localidad de memoria (1416)
1414	CF		RST 08H'	Regreso al monitor
1416				

La forma de cargar este programa es tecleando: R,COM1(T),MEM(0),1410,3E70; MEM(0),1412,32;MEM(0),1414,CF;SALE(S),1410

1.2(b) Transferir un dato de una localidad de memoria a otra.
Enmascarando los primeros cuatro bits.

LOCALIDAD	CODIGO DE MAQUINA	ETIQUETA	MNEMONICO	COMENTARIOS
1410	3E 70		LD A,N	A <- N (70)
1412	F6 0F		AND 0F	Enmascara los primeros 4 bits
1414	CF		RST 08H'	Regreso al monitor
1415				

La forma de cargar este programa es tecleando: R,COM1(T),MEM(0),1410,3E70;
MEM(0),1412,2F;MEM(0),1414,CF;SALE(S),1410

1.3(c) Transferir un dato de una localidad de memoria a otra.
Colocando un uno en el bit de la extrema derecha y manteniendo sin cambio los bits restantes.

LOCALIDAD	CODIGO DE MAQUINA	ETIQUETA	MNEMONICO	COMENTARIOS
1410	3E 70		LD A,N	A <- N (70)
1412	F6 01		OR A,01	Enmascara el bit de la extrema izquierda
1414	CF		RST 08H'	Regreso al monitor
1415				

La forma de cargar este programa es tecleando: R,COM1(T),MEM(0),1410,3E70;
MEM(0),1412,F601;MEM(0),1414,CF;SALE(S),1410

1.3(e) Transferir un dato de una localidad de memoria a otra.
Cambiando el bit de la extrema izquierda. (Si vale 0 hacerlo 1 y viceversa)

LOCALIDAD	CODIGO DE MAQUINA	ETIQUETA	MNEMONICO	COMENTARIOS
1410	3E 70		LD A,N	A <- N (70)
1412	EE 80		XOR 80	Coloca un cero en el bit de la extrema izquierda
1414	CF		RST 08H'	Regreso al monitor
1415				

La forma de cargar este programa es tecleando: R,COM1(T),MEM(0),1410,3E70;
MEM(0),1412,EE80;MEM(0),1414,CF;SALE(S),1410

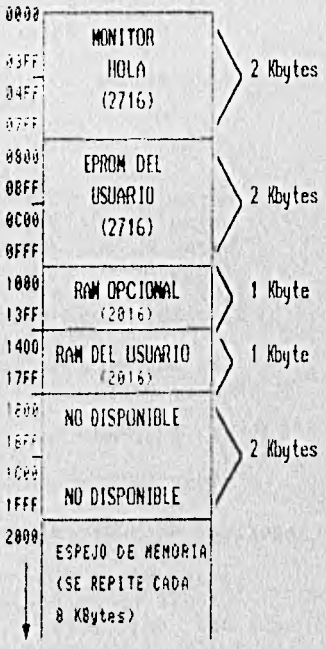
APENDICE D

MAPA DE MEMORIA DEL SISTEMA DE DESARROLLO.

Se le llama MAPA DE MEMORIA a la representación de los bloques en que se ha dividido el espacio de memoria direccionable por el microprocesador. Cada bloque o partición corresponde al rango de direcciones ocupado por un circuito de memoria, de acuerdo a la signación que se hace de las lineas del bus de direcciones que no van conectadas a las entradas de direcciones del circuito de memoria.

En este proyecto, la memoria EPROM (la que almacena el programa MONITOR) utiliza las direcciones mas bajas del espacio de memoria; es decir, se le han asigando los 2 primeros K de memoria (direcciones 0000H a 07FFH), conectando las lineas de dirección A11-A15 en forma tal que el circuito se active únicamente cuando todas ellas están en 0 lógico.

En la siguiente figura se muestra el direccionamiento completo utilizado en este proyecto.



APENDICE E

ESTUDIO ECONOMICO DEL PROYECTO PROPUESTO.

El siguiente es un listado completo del material utilizado en la elaboración del presente trabajo. Considérese que los precios en dispositivos electrónicos tienden a bajar de precio y por lo tanto podrían no estar vigentes.

Además de lo anterior, si se da el caso de la elaboración masiva (por grupos), los costos se reducen tanto en material como en la tarjeta de circuitos impresos, haciendo más factible su adquisición por la mayoría de los alumnos.

CANTIDAD	DESCRIPCION	PRECIO
1	Cristal de cuarzo 3579.54 Khz	N$ 05.00
10	Resistencias 330 *j*	N$ 00.10
1	Resistencia 560 *j*	N$ 00.10
15	Resistencias 12K *j*	N$ 00.10
6	Resistencias 1K *j*	N$ 00.10
1	CPU Z80A	N$ 10.00
2	Memorias EEPROM 2716	N$ 31.00
1	Memoria PROM 2016	N$ 35.00
2	Puertos 8255	N$ 21.00
6	Displays 7 segmentos	N$ 08.00
2	CI 7404	N$ 02.00
1	CI 7406	N$ 02.50
1	CI 74145	N$ 03.00
1	CI 74139	N$ 03.00
2	Bases p/CI de 16 patas	N$ 00.50
3	Bases p/CI de 14 patas	N$ 00.50
4	Bases p/CI de 24 patas	N$ 01.40
3	Bases p/CI de 40 patas	N$ 02.10
1	Tarj. p/ctos. imp. 20X15 cms.	N$ 09.00
21	Interruptores miniatura	N$ 01.40
1	Diodo 1N914	N$ 00.40
2	Capacitores de 250 *fF*	N$ 00.40
1	Capacitor electrol. 1000 *fF* a 10 V.	N$ 00.90
1	Capacitor electrol. 47 *fF* a 16 V.	N$ 00.40
6	Transistores NPN 2907A	N$ 02.00

Para el programador de memorias (opcional):

2	Bases p/CI de 24 patas	N$ 01.40
3	Resistencias de 220 K *j*	N$ 00.10
1	Tarj. p/ctos. imp. 5X5 cms.	N$ 00.50
1	Cable plano de 26 hilos y 20 cms de longitud.	N$ 09.00
1	Conector de 26 patas p/cable plano.	N$ 02.50
	TOTAL	N$ 151.70

CONCLUSIONES

En el presente trabajo de tesis, se presenta un computador digital pequeño (llamado en la industria SISTEMA DE DESARROLLO) para propósitos generales a partir de especificaciones funcionales, para concluir con su ensamblado.

Aunque el computador es pequeño, está muy lejos de ser útil comparado con sistemas comerciales electrónicos de procesamiento de datos, sin embargo, incluye especificaciones funcionales suficientes para construirse en el laboratorio y el producto ya terminado puede ser un sistema útil capaz de procesar datos digitales.

El computador puede dividirse en tres fases interrelacionadas:

a) Diseño del sistema. El diseño del sistema versa sobre las especificaciones y propiedades del sistema (teoria basicamente), la formulación de las instrucciones del computador y la investigación de su factibilidad económica.

b) Diseño lógico. El diseño lógico es enormemente influenciado por la programación en lenguaje máquina (MONITOR), por los diferentes circuitos lógicos y circuitos de memoria.

c) Diseño del circuito. El diseño del circuito especifica los componentes de los diferentes circuitos integrados utilizados para su ensamblado, asi como su disposición dentro del circuito impreso.

El diseño y construcción del presente proyecto es una tarea complicada, no se debe esperar que se cubran todos los aspectos en detalle del mismo. Se asume que el lector tiene el nivel de conocimientos suficientes para su total comprensión.

El SISTEMA DE DESARROLLO será ensamblado por estudiantes con un nivel elemental de conocimientos en circuitos lógicos y de computación/programación.

Aquí la importacia estriba en la aplicación didáctica que se le puede dar; ya que los mismos principios utilizados aquí para el minicomputador, son los mismos empleados en sistemas comerciales.

Se optó por utilizar el microprocesador Z80 por su gran soporte técnico y literatura comparado con microprocesadores similares; se hizo la siguiente evaluación:

- El microprocesador Z80 tiene 256 distintas instrucciones.
- El microprocesador 8080 tiene 244 diferentes instrucciones.
- El microprocesador 6800 tiene 197 diferentes instrucciones.

Del análisis anterior se observa que el microprocesador Z80 tiene 12 instrucciones más que el 8080 y que soporta las instrucciones de este microprocesador, sin considerar instrucciones especiales extras que no contiene ningún otro microprocesador. En el APENDICE B (LISTADO COMPLETO DEL CODIGO DE INSTRUCCIONES) se muestra entre parétesis los mnemónicos del microprocesador 8080. Esto con la intención de darle mayor aplicación al microcomputador utilizando tentativamente instrucciones del microprocesador 8080 y que corran en ambos microprocesadores.

En el APENDICE V.3 (APENDICE.C) se incluyen algunas prácticas muy sencillas para su implementción inmediata, sin olvidar que las mismas solamente son ilustrativas y puede aumentarse el grado de dificultad de acuerdo al avance del curso o interés del usuario.

Por último, en el APENDICE.D, se incluye un listado completo de los elementos que integran el presente proyecto, asi como su costo unitario y el costo total.

BIBLIOGRAFIA

1. Uruñuela M. José María. MICROPROCESADORES, PROGRAMACION E INTERCONEXION.
   McGRAW-HILL INTERAMERICANA DE MEXICO, S.A DE C.V. 1989.

2. Ciarcia Steve. BUILD YOUR OWN Z-80 COMPUTER.
   Byte books Mc Graw Hill. 1991.

3. Hugo G. García Guerra. MICROPROCESADORES, TEORIA Y PRACTICA.
   Limusa, 1988.

4. Joseph C. Nichols, Elizabeth A. Nichols, Peter R. Rony
   MICROPROCESADOR Z-80, PROGRAMACION E INTERFACES.

5. Morris Mano M. ARQUITECTURA DE COMPUTADORES.
   Prentce Hall, 1983

6. Morris Mano M. LOGICA DIGITAL Y DISEÑO DE COMPUTADORES.
   Prentice Hall, 1982.
   Publicaciones Marcombo, S.A. 1994

7. E. Mandado, E. Tassis. DISEÑO DE SISTEMAS DIGITALES CON MICROPROCESADORES.
   Publicaciones Marcombo, S.A. 1993