UNIVERSIDAD LA SALLE

ESCUELA DE INGENIERIA

INCORPORADA A LA UNAM

LA TERMINAL INTELIGENTE CON UN MICROCOMPUTADOR EDUCATIVO

TESIS PROFESIONAL

QUE PARA OBTENER EL TITULO DE

INGENIERO MECANICO ELECTRICISTA

AREA: ELECTRONICA

P R E S E N T A:

PABLO TOMASINI CAMPOCOSIO

MEXICO, D.F. 1986

UNAM – Dirección General de Bibliotecas Tesis Digitales Restricciones de uso

I N D I C E

CAPITULO III INTERFACE DE UN MICROCOMPUTADOR

CAPITULO IV DISPOSITIVOS DE ALMACENAMIENTO

CAPITULO V IMPLEMENTACION DEL SISTEMA

I N T R O D U C C I O N

I N T R O D U C C I O N

El avance tecnológico es cada vez más rápido y notable en los campos de más alta tecnología, como la computación electrónica.

Desde las primeras computadoras electrónicas construídas a raíz de la Segunda Guerra Mundial hasta las computadoras actuales, media una diferencia tal, que más parece un abismo que el producto de una evolución. Las etapas por las que ha pasado esta evolución fueron claras en el principio: del bulbo al transistor y luego al circuito integrado; del funcionamiento casi esotérico reservado solo a unos cuantos científicos iniciados, hasta los cursos por correspondencia. Sin embargo, esta clara evolución perdió su transparencia hace unos 25 años, en que hicieron su aparición los llamados microprocesadores.

La apariencia inicial de los microprocesadores era la de una buena solución a un problema estrictamente electrónico. Se trataba de incluir en un solo circuito integrado (chip) buena parte de la computadora: el Procesador.

Esto traería como consecuencia inmediata un incremento im--

portante en la confiabilidad del equipo, simplificaría su mantenimiento al convertir una parte de él en "desechable", ya que cualquier falla dentro del chip microprocesador solo se puede corregir mediante el reemplazo del mismo; permitiría equipos más pequeños y con menos problemas de ventilación, y finalmente, se haría posible una importante reducción de su costo.

Los primeros intentos, como podía esperarse, no lograron -- los objetivos planteados; los microprocesadores de esa época eran tan lentos y hacían tan pocas operaciones diferentes, que más -- que computadoras, los equipos construídos con ellos parecían simples calculadoras.

Ese aparente fracaso fué solo pasajero; pronto aparecieron dos microprocesadores que habrían de rescatar los objetivos originales: el 6800 de Motorola y el 8080 de Intel.

La comunidad de especialistas en computación recibió con -- frialdad estos dos microprocesadores, considerándolos más como juguetes que como instrumento serio.

Esto parecía confirmarse en la publicidad de las primeras -

microcomputadoras basadas en el 8080 y en el 6800 que ofrecían - programas para jugar al gato y al backgamon contra la máqui--- na.

Sin embargo, se trataba del lobo con piel de oveja. Unos cuantos entusiastas empezaron a trabajar en una serie de programas básicos y lenguajes de programación que hacían que las mi-- crocomputadoras se comportaran en forma muy parecida a sus hermanas mayores. Entre estos entusiastas destaca la compañía Digital Research de Estados Unidos, que con su CP/M, sentó las ba-- ses para utilizar las microcomputadoras en los negocios.

La evolución de las microcomputadoras se aceleró aún más - con la aparición de la siguientes generación de microprocesa----- dores: el Z80 de Zilog y el 6502 de MOS Technology.

El Z80 descendiente del 8080, ya indicaba un cambio de concepto importante en el desarrollo de los microprocesadores: sus - mejoras estaban más del lado del juego de instrucciones, que ahora facilitaban el trabajo del programador, que no del lado de la electrónica pura.

Este sutil cambio de concepto marca una de las vertientes - actuales de desarrollo de los microprocesadores. Se trata de hacerlos tan poderosos y fáciles de usar como una computadora --- grande.

La evolución está muy lejos de terminar con el Z80 y el 6502.

Está ya en el mercado la siguiente generación. En ella - destacan el Z8000, el 68000 y el 8088, siguiendo en todos ellos la línea evolutiva del Z80; mejorar el juego de instrucciones y su - capacidad. Actualmente el Z8000 y el 68000 pueden manejar tanta memoria como cualquiera de los grandes equipos modernos: ocho millones de caracteres.

El presente trabajo es el desarrollo de un microcomputador basado en algún microprocesador de 8 bits, (comercial y de pro-- pósito general), que cumpla con una serie de características es-- cenciales para que pueda utilizarse como un Sistema de Microprocesador Educativo. Dirigido no solamente a instituciones educativas, sino también a cualquier tipo de industria que tenga necesidades en el área de microcomputadores.

Las características de nuestro sistema son las siguientes, y las dividimos en dos aspectos importantes que son: el físico -- (hardware) y la programación (software).

El aspecto físico considerará:

- *En una sola tableta.*
- *Mínimo de componentes.*
- *Fácil de expander (la cantidad de memoria y el número de dispositivos de Entrada/Salida).*
- *Conector compatible con cualquier terminal (impresora, - video, teletipo, etc.) comercial, usando la interface RS-232.*
- *Funcional por su tamaño.*
- *Diferentes velocidades de operación.*

El aspecto programción:

- *Fácil operación.*
- *Versátil.*
- *Acceso sencillo al programa monitor.*
- *Cantidad de comandos y funciones suficientes para que -*

sea educacional.

- *Puerto paralelo programado como puerto serie.*
- *Posibilidad de sustitución del programa monitor, por o-- tro de interés para el usuario.*

Con las características anteriores pretendemos que el Microcomputador eduacional pueda, en un momento dado, cambiar su objetivo principal (Educativo) y convertirse en un sistema para a-- plicarse en la solución de algún problema específico.

Esto es factible, cambiando el programa monitor original por el programa específico y expandiendo la circuitería del sistema si se requiere.

El puerto en paralelo, programado como puerto serie nos facilita enormemente cualquier aplicación, ya que con un solo cir-- cuito integrado tenemos dos formas de transmisión-recepción (serie y/o paralelo).

Esta capacidad de modificación de nuestro sistema puede ser utilizada por el educando para comprobar que los Sistemas de Mi

croprocesador practicamente no tienen límites en sus aplicaciones y que programandolos de una forma adecuada resuelven casi cualquier problema; dejando la circuitería muy elemental.

El programa que permitirá la operación de nuestro sistema estará estructurado de forma tal que, sea fácil el acceso al sistema y que cualquier programa pueda ser probado, depurado y almacenado en un cassette de audio convencional; para todo esto -- tendremos una serie de comandos que abarquen el mayor número - de funciones y den simpleza al momento de programar.

Un aspecto más de suma importancia es el costo que pueda tener el sistema; una forma de reducirlo será teniendo un diseño eléctrico muy compacto, con componentes baratos y de fácil adquisición en el mercado Nacional, apoyando el diseño con un poderoso y sofisticado programa monitor.

C A P I T U L O I

DESCRIPCION GENERAL DE UN MICROCOMPUTADOR

1.- Los Microprocesadores y los Microcomputadores:

Debido a que el presente trabajo es diseñar un microcomputador basado en un microprocesador, empezaremos por aclarar y definir ambos términos para evitar confusiones en referencias posteriores.

Uno de los resultados en los avances tecnológicos en la fabricación de componentes de estado sólido, es la construcción de componentes con un gran número de transistores, dentro de una sola oblea de silicón. Si el número de circuitos que componen un sistema es grande, digamos de 50 a 100 ó más, como sucede en el caso de los computadores y sistemas similares, la tecnología se conoce como integración en gran escala (LSI). Entonces la integración en gran escala implica la incorporación de la mayor parte posible de un sistema a la oblea de mayor tamaño que pueda usarse. Una consecuencia directa de la LSI es el microprocesador. En general, el microprocesador es un dispositivo lógico programable fabricado conforme el concepto de integración en gran escala y con alto grado de flexibilidad. Por sí mismo no puede

ejecutar tareas, pero puede ser programado y conectado a un conjunto adicional de dispositivos lógicos que incluyen elementos de memoria y de entrada/salida, para llevar a cabo algunas funciones bien definidas. Este conjunto de elementos interconectados incluyendo al microprocesador usándose en propósitos establecidos, conforman un microcomputador o sistema de microprocesador.

Algunas características principales del microcomputador son un bajo costo y tamaño reducido. A diferencia de los computadores y minicomputadores, un sistema de microprocesador hace posible - el uso de un sistema lógico programable, de costo reducido, de - buena velocidad y una gran potencia computacional en aplicaciones específicas, tales como terminales inteligentes, calculadoras, instrumentos, control de máquinas, procesos químicos, sistemas de tráfico, etc., en los cuales un computador o minicomputador serían excesivamente grandes en todos aspectos.

2.- *Organización Básica de un Microcomputador:*

Después de haber aclarado los conceptos de microprocesador

y microcomputador, haremos una breve descripción de cómo funcio na y cómo está constituído un microcomputador; además definire-- mos algunos términos importantes que citaremos continuamente en el transcurso de este trabajo.

Cualquier sistema de computador, minicomputador o micropro cesador, está compuesto escencialmente de dos partes que son: -- hardware y software. Los componentes y circuitos físicos que -- constituyen un sistema de computador se conoce como hardware. - Estos circuitos son capaces de ejecutar solamente un número pe-- queño de operaciones y cualquier operación adicional debe ser - realizada por medio de un programa, que es un conjunto organi-- zado de operaciones elementales del computador, llamadas instruc- ciones que manejan información o datos.

El proceso de elaborar un programa desde que se concibe la idea hasta su depuración, le llamaremos software.

Un sistema de computador consta básicamente de cinco unida des funcionales que son (Figura 1):

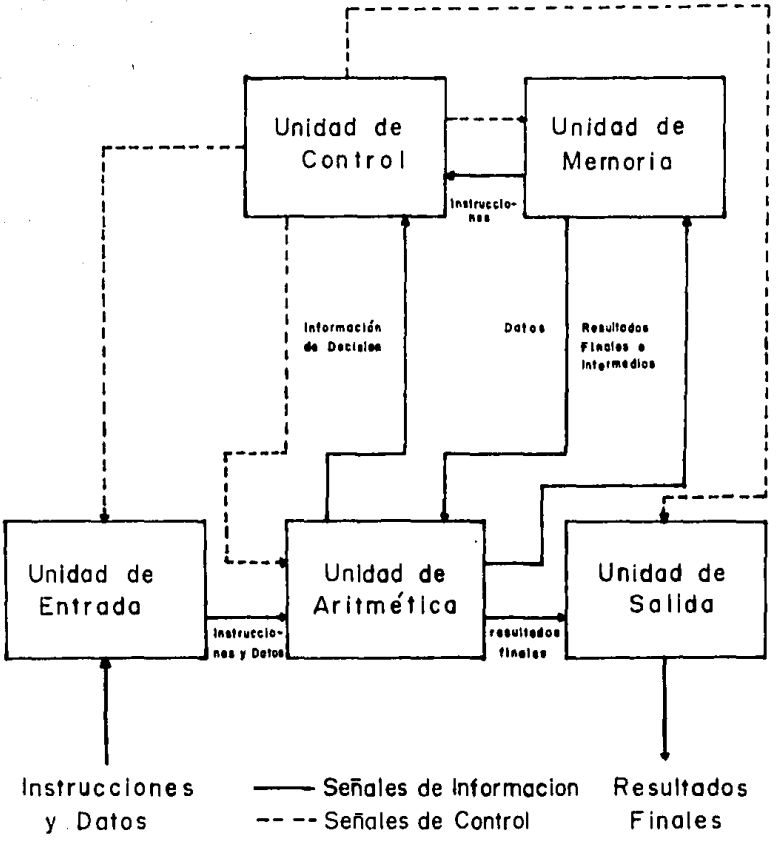


Fig. 1: Organización Básica de un Computador

- *LA UNIDAD DE MEMORIA.*
- *LA UNIDAD ARITMETICA.*
- *LA UNIDAD DE CONTROL.*
- *LA UNIDAD DE ENTRADA.*
- *LA UNIDAD DE SALIDA.*

El programa y los datos se almacenan en la unidad de memoria a través de la unidad de entrada. Cada instrucción del programa llega automáticamente a la unidad de control, donde son interpretadas y ejecutadas. La ejecución necesariamente requiere de datos, que son llevados a la unidad aritmética. Durante el proceso de ejecución de un programa o al término de él los resultados derivados son enviados a la unidad de salida. La unidad aritmética junto con la de control generalmente son llamadas Unidad Central de Proceso (CPU). La CPU de un microcomputador es el microprocesador.

Además de la unidad de memoria, hay otras unidades capaces de almacenar información en grupos de dígitos binarios (bits), llamadas registros. Un grupo de dígitos binarios manejados to-

dos al mismo tiempo, por el computador se conocen como palabra, y el largo de palabra es la cantidad de bits. Entonces una palabra es la unidad lógica básica de un computador. Las instrucciones y los datos, pueden estar formados por una o más palabras. Los microprocesadores se han fabricado con palabras de diferente largo cuatro, ocho, doce, diez y seis bits, etc., siendo los últimos mucho más poderosos. En los microprocesadores de ocho bits, se acostumbra llamar a la palabra "byte".

2.1.- LA UNIDAD DE MEMORIA

Es el lugar físico donde se almacena toda la información, ya sea datos y/o instrucciones. Esta unidad se encuentra dividida en unidades de subalmacenamiento que contienen hasta una palabra de computador. El lugar dentro de memoria donde se halla cada unidad de subalmacenamiento le llamaremos localidad de memoria y se identifica por una dirección, que es simplemente un número entero y se asigna a una y solamente una localidad. La palabra de computador que se encuentra en una localidad es re--

ferida como el contenido de la localidad.

Una característica importante de los computadores digitales es que los datos y los programas son almacenados en la unidad - de memoria. Esto condujo a dos filosofías básicas del diseño.

En la primera, el computador tiene dos memorias separadas y distintas; el programa estará siempre dentro de una y los da-- tos en otra.

La segunda, no hace diferencia entre los datos y el progra ma, que pueden estar en cualquier área dentro de memoria, sien- do responsabilidad del programador hacer la distinción del tipo - de información guardada.

La ventaja que presenta sobre la anterior es la facilidad de manejar datos e instrucciones de igual manera.

2.2.- LA UNIDAD ARITMETICA.

Es donde ocurren la mayoría de las manipulaciones de datos, que involucran a operaciones lógicas y aritméticas. Las operacio

nes aritméticas muy complejas, tienen que ser ejecutadas por medio de un programa y utilizando solamente las operaciones lógicas y aritméticas disponibles en el microprocesador.

Típicamente, el registro (localidad de memoria dentro de la unidad aritmética) más importante es el acumulador; normalmente contiene el operando prioritario antes de realizar la operación y después de ella, el resultado.

La unidad aritmética tiene también, un registro bandera, -- donde se almacena la palabra bandera. Cada bit de ésta genera información que indica el estado del acumulador después de realizada una operación. Esta información es usada por el programador para saber qué tipo de resultado se obtiene, y cómo proceder si se cumplen algunas condiciones, como: resultado negativo, tipo de paridad, etc.

2.3.- LA UNIDAD DE CONTROL.

La función de esta unidad, es vigilar y dirigir las operaciones del computador. Automáticamente recibe las instrucciones,

una a la vez, desde la unidad de memoria, las decodifica y ge-- nera las señales necesarias para llevar a cabo su ejecución. Las instrucciones están en orden secuencial dentro de memoria, y las localidades proporcionan su dirección a la unidad de control por el registro contador de programa, que se encuentra dentro de -- ésta. La instrucción se mantiene en otro registro, únicamente -- mientras se está ejecutando; el registro se llama registro de instrucción. En este lugar la instrucción es interpretada, siempre y cuando cumpla con el formato predefinido por el fabricante.

La sincronización de todas las operaciones y la coordina--- ción entre todas las unidades, se hace por medio de una señal -- periódica (señal de reloj del sistema), que recibe directamente la unidad de control.

2.4.- LA UNIDAD DE ENTRADA Y DE SALIDA.

Las encargadas de mantener el contacto con el mundo ex--- terior, son éstas dos unidades.

Se encargan de transferir la información en diferentes velo-

cidades y con los diferentes lenguajes manejados por los programadores; la unidad de entrada recibe los datos y las instrucciones del mundo exterior, y los coloca eventualmente en la memoria. La unidad de salida recibe los resultados y los comunica al operador a través de los dispositivos llamados Puertos. Los resultados pueden ser comunicados a otros sistemas de computador según sea el caso. Los microprocesadores pueden manejar varios Puertos, y de diferentes tipos; direccionandolos siempre como localidades de memoria.

Un computador maneja información digital y frecuentemente la debe comunicar a dispositivos o sistemas no digitales, para tales casos, existen componentes capaces de transformar esta información en analógica, o viceversa. Estos componentes son los convertidores digitales/analógicos y analógicos/digitales, respectivamente.

Los Puertos de E/S tienen la capacidad de conectarse a los convertidores directamente para lograr el intercambio de información.

2.5.- BUSES.

Las diferentes unidades de los microcomputadores se conec--

tan por un conjunto de líneas, sobre las cuales se transfiere la información desde alguna fuente, hacia algún destino. En la figura 2, mostramos la estructura de un microcomputador a nivel de bloques, especificando cada unidad y las líneas de interconexión - entre ellas. Las líneas de conexión, tomadas como un conjunto, - se les llama bus. Los buses se clasifican en:

- **Bus de Direcciones**, que es unidireccional y por el cual fluyen solamente direcciones para memoria y puertos de E/S desde el microprocesador.

- **Bus de Datos**, bidireccional, es para transferir datos e instrucciones a las unidades que lo requieran estableciendo el intercambio de información a través de este bus.

- **Bus de Control**, por el cual viajan las señales que mantienen los tiempos de sincronía entre las diferentes unidades y la información de estado de los dispositivos. Algunas líneas son bi direccionales, mientras que otras son unidireccionales (por esta razón no indicamos el sentido del Bus de Control en la figura).

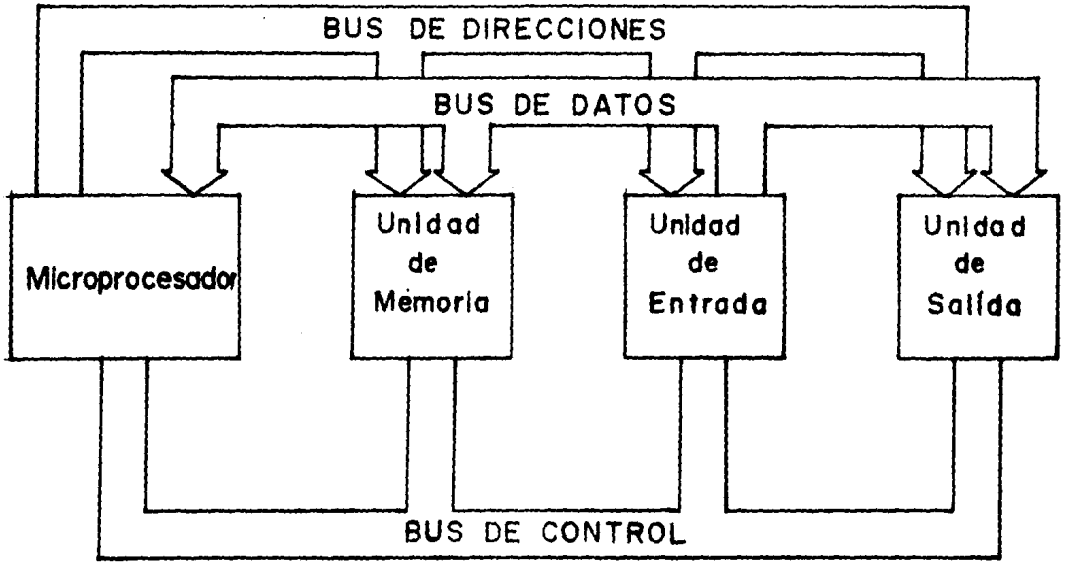


Fig.2: Estructura Típica de los BUSES

3.- Operación del Microcomputador:

Una explicación de cómo interactúan, cuál es el flujo de información y cuál es el mecanismo de funcionamiento de las diferentes unidades del microcomputador es el tópico a tratar.

La unidad de control trabaja progresivamente sobre tres -- fases: Traer, decodificar y ejecutar. Después que el programa y los datos han sido cargados en memoria, la dirección de la primera instrucción por ejecutarse se pone en el contador de programa, y la unidad de control realiza la primera fase: traer de memoria la instrucción. Esta instrucción almacenada en la localidad de memoria especificada por la dirección que tiene el contador de programa es enviada al registro de instrucciones de la unidad de control por el bus de datos. Ya que el programa es almacenado en orden secuencial, el contador de programa se incrementa en uno para que contenga la dirección de la siguiente instrucción o palabra del programa. Ahora, la unidad de control - decodifica la palabra recibida (fase de decodificación). Si determina que la instrucción recibida es de más de una palabra, la - fase de traer se repite el número de veces suficiente para que la

instrucción se complete. Después de que las fases de traer y de codificar son realizada, la unidad de control procede a la fase de ejecución. En este momento, genera las diferentes señales para - activar a los circuitos involucrados en cumplir la operación específica. Cuando la instrucción proporciona la dirección de un o-- perando, la unidad de control hace entonces los arreglos para -- transferir la información entre la localidad especificada en la -- instrucción y la unidad correspondiente (aritmética o de salida). Finalmente supervisa la ejecución de la operación. Una vez com-- pletada la operación, regresa a la fase de traer y obtiene la siguiente instrucción desde memoria, basándose en la dirección que tiene el contador de programa. El proceso se repite hasta que - aparezca una instrucción de Alto.

En general, las instrucciones son ejecutadas en orden secuencial; sin embargo, hay ocasiones en que es necesario ignorar este orden. En tales casos la localidad de la siguiente instrucción a ejecutarse puede ser otra a la que está inmediatamente de la - que prevalece. Las instrucciones quepermiten esto, se conocen como

de salto o llamada y proporcionan la nueva dirección a partir de la cual se realizará la fase de traer.

4.- Análisis Comparativo de Algunos Microprocesadores:

Los microprocesadores son unos dispositivos lógicos nuevos y fascinantes que han tenido un marcado efecto sobre el desarrollo tecnológico actual, lo mismo pueden ser encontrados en calculadoras, equipos de oficina, instrumentos científicos, equipos médicos, terminales inteligentes bancarias o de almacenes de servicio, que en juegos de video o en aplicaciones para el hogar, etc.

Debido a que la oferta y la demanda está creciendo continuamente y a que cada vez son más variados y diversos (los microprocesadores), decidimos hacer un análisis comparativo sobre ellos y seleccionar el más apropiado a nuestras necesidades.

En la tabla I mostramos las características de algunos microprocesadores de propósito general y de ocho bits sobre la cual realizamos la selección del microprocesador que usaremos.

	CLASIFICACION	TECNOLOGIA	CARACTERISTICAS DE LA FAMILIA. CARACTERISTICAS DEL MICROPROCESADOR													TAMAÑO DE PALABRA (datos/instrucción)	DIRECCIONAMIENTO	RELOJ (kHz./FASES)	TIEMPO DE ACCESO A LOS REGISTROS (µseg. por dato)	[illegible]	[illegible]	[illegible]	TAMAÑO DEL STACK (en 8 bits)	VOLTAJES REQUERIDOS	POTENCIA DISIPADA	ENCAPSULADO (número de patitas)	NOTAS
ELECTRONICS ARRAYS	8 - bit CPU	NMOS		✓		✓	✓		✓		✓	✓	✓		✓	8/8	64K	1000/2		1	1	15	7 X 16			40	OPERACIONES EXPANDIBLES A 16 B.
FAIRCHILD F-8	8 - bit CPU	NMOS	✓	✓	✓	✓	✓	✓	✓		✓	✓	✓		✓	8/8	64K	2000/0	2	1			(64B)	+ 5, + 12	.6	40	RELOJ EN EL CHIP.
GENERAL INSTRUMENTS CP1600	8 - bit CPU	NMOS				✓	✓		✓				✓			8/8	2K	800/2	5	1		14	3 X 11	+ 12, + 5	.5	40	ESPACIO PARA PROGRAMAS 16K EN 8 BANCOS.
INTEL 4040	4 - bit CPU	PMOS	✓	✓		✓	✓	✓	✓				✓		✓	4/8	4K	740/2	10.8	1		24	7 X 12	+15 ó (-10,+5)	1.0	16,24	ESPACIO PARA PROGRAMAS 8K EN 2 BANCOS.
INTEL 8008-1	8 - bit CPU	PMOS	✓	✓	✓	✓	✓	✓	✓					✓	✓	8/8	16K	800/2	12.5	1		8	7 X 8	+ 5, - 9	1.0	18	
INTEL 8080	8 - bit CPU	NMOS	✓	✓	✓	✓	✓	✓	✓		✓	✓	✓	✓	✓	8/8	64K	2000/2	2	1		8	(64K)	-5, +5, +12	1.0	40	OTROS FABRICANTES: AMD, TI
MONOLITHIC MEMORIES 6701	4 - bit SLICE	TTL								✓				✓		4/12		6000/1	.1	1		16	(RAM)	5	1.0	40	SIMILAR AL AMD 2901
MOS TECHNOLOGY 6502	8 - bit CPU	NMOS		✓		✓	✓		✓		✓		✓		✓	8/8	64K	1000/2	2	1	1		(64B)	5	.25	40	SIMILAR AL MOTOROLA 6800
MOSTEK 5065	8 - bit CPU	PMOS							✓		✓	✓			✓	8/8	32K	1600/2	10	1			(64B)	-12, +5, +5	.7	40	CPU 3-ESTADOS.
MOTOROLA 6800	8 - bit CPU	NMOS		✓	✓	✓	✓		✓		✓	✓	✓		✓	8/8	64K	1000/2	2	1	1		(64K)	5	.75	24,40	OTRO FABRICANTE: AMI
NATIONAL CPU - 8	8 - bit CPU	NMOS		✓	✓	✓	✓	✓	✓		✓	✓			✓	8/8	64K	/2	1.6	2	2		(64K)			40	
NATIONAL IMP - 4	4 - bit CPU	PMOS						✓		✓			✓		✓	4/4	4096	500/4	12	4			7 X 12	- 12, + 5	1.0	24,40	DATOS EN STACK 16 X 4
NATIONAL IMP - 8	8 - bit CPU	PMOS								✓	✓	✓			✓	8/8	64K	715/4	4.8	1	1		16 X 8	- 12, + 5	1.0	22,24	
RCA COSMAC	8 - bit CPU	CMOS							✓		✓	✓		✓	✓	8/8	64K	2020/1	6	1		9	7 X 16	de 5 a 12	.01	28,40	[illegible]
ROCKWELL PPS-4	4 - bit CPU	PMOS	✓	✓	✓	✓	✓		✓		✓		✓			4/8	4K	200/2	5	1		1	2 X 12	17	.275	42	ESPACIO PARA PROGRAMAS 8K EN 2 BANCOS.
ROCKWELL PPS-8	8 - bit CPU	PMOS	✓	✓	✓	✓	✓		✓		✓	✓	✓		✓	8/8	64K	256/2	4	1	1	2	(64B)	17	.3	42	
SIGNETICS 2650	8 - bit CPU	NMOS							✓				✓		✓	8/8	32K	1200/1	4.8	2			8 X 15	5	.5	40	
TEXAS INSTRUMENTS SBP0400	4 - bit SLICE	TTL								✓			✓			4N/9+		1000/1	1	16				.05	.13	40	
TRANSITRON 1601	4 - bit SLICE	TTL								✓	✓	✓			✓	16/16	32K		.4	4	4		(64B)			40	[illegible]
ZILOG Z - 80	8 - bit CPU	NMOS	✓	✓	✓	✓	✓	✓	✓		✓	✓	✓	✓	✓	8/8	64K	4000/2	2	1	8	16	(64K)	5	1.5	40	OTRO FABRICANTE: MOSTEK

TABLA I: CARACTERISTICAS DE MICROPROCESADORES DE PROPOSITO GENERAL

4.1.- CUADRO DE CARACTERISTICAS.

Los puntos más importantes en los que basamos el análisis para la selección del microprocesador son:

- Posibilidad de diseñar el sistema con un mínimo de componentes y circuitos externos.
- Alto grado de compatibilidad con otros microprocesadores y con sus componentes o familias.
- Versatilidad en el Grupo de Instrucciones.
- Factibilidad de adquisición en el mercado Nacional.
- Diferentes velocidades de trabajo.
- Compatibilidad con diferentes tipos de memoria.
- Facilidad en el manejo de memoria (direccionamiento sencillo).
- Soporte con otros componentes de la misma familia.
- Versatilidad en el manejo de dispositivos de Entrada/Salida.
- Facilidad de adquisición de información sobre la familia.

Después de haber estudiado los puntos anteriores llegamos a la conclusión de que el microprocesador más apropiado es el Z-80 y su familia de componentes (el PIO Z-80 y el CTC Z-80).

C A P I T U L O I I

UNIDAD CENTRAL DE PROCESO

La Unidad Central de Proceso (CPU) es la cabeza de cualquier sistema de computador, minicomputador o microcomputador. La función de ésta es obtener la información desde la memoria y llevar a cabo operaciones lógicas, aritméticas, transferencias de datos y manejo de interrupciones.

1.- PROGRAMACION:

1.1- Principios Básicos y Características del Lenguaje de Programación.

El microcomputador digital es un dispositivo electrónico capaz de aceptar, guardar y manipular información en forma de códigos digitales y proporcionar resultados. La información que maneja son una serie de sentencias estructuradas de forma tal, que tomadas como un conjunto, permitan realizar una serie de operaciones y así lograr una tarea específica. Esta serie de sentencias determinan un programa de computador. Cuando se escribe un programa para un microcomputador se utiliza el código mnemónico o lenguaje ensamblador, que son una serie de símbolos

representativos de una operación que realizará la CPU. Un programa escrito en este código o lenguaje se convierte en lenguaje de máquina utilizando un programa traductor de lenguaje llamado ensamblador; la secuencia de números que han sido así traducidos, para que los entienda la máquina, forman el programa objeto; el programa del cual se hace la traducción (lenguaje ensamblador escrito por una persona), es el programa fuente.

El programa fuente contiene diferentes tipos de información:

- *Códigos de Operación*
- *Byte de Datos*
- *Códigos de Dispositivo*
- *Bytes de Dirección*
- *Bytes de Desplazamiento*

- ***Códigos de Operación.-*** *Definen la acción específica que realizará el microprocesador; y puede ser una transferencia de datos, una operación lógica, una instrucción de bifurcación, una operación de stack, una operación de E/S (entrada/salida) o una*

operación de control de máquina.

- ***Byte de Datos.-*** *Es un número binario de ocho bits que el microprocesador utilizará en alguna instrucción aritmética o lógica, o para guardarlo en memoria. El código de este número puede ser cualquiera (código binario, ASCII, hexadecimal, etc.)*

- ***Códigos de Dispositivo.-*** *Es una palabra de ocho bits que se utiliza para identificar al dispositivo de E/S con el cual la CPU desea intercambiar información, sincronizada además con un impulso de selección o de habilitación.*

- ***Bytes de Dirección.-*** *El direccionamiento de memoria se maneja con dos bytes, el byte alto (HI) que está formado por los ocho bits de mayor peso de los diez y seis disponibles para direccionamiento; y el byte bajo (LO) que son los ocho restantes. Este artificio de dividir las líneas de dirección en dos, se debe a que el microprocesador Z80 es de ocho bits (el bus de datos con ocho líneas).*

- ***Bytes de Desplazamiento.-*** *Estos bytes aparecen en las instrucciones que utilizan direccionamiento indexado, que es una*

técnica para definir una dirección de memoria de dos bytes añadiendo un desplazamiento al número de diez y seis bits que reside en una área de memoria dentro de la CPU. Entenderemos por desplazamiento de un número, la obtención del segundo complemento de éste.

1.2- Pasos comunes para Programar un Microcomputador:

a) Planificación del Programa

b) Escritura del Programa (Programa Fuente)

c) Ensamblar el Programa (Crear el Programa Objeto)

d) Carga del Programa en Memoria

e) Depurar el Programa.

La figura 1 detalla un diagrama de bloques de los pasos anteriores.

La planificación del programa incluye el desarrollo del algoritmo y la estructuración de este por medio de un diagrama de flujo. Una vez hecho lo anterior, se escribe el programa en lenguaje ensamblador (Programa Fuente); usando después un programa

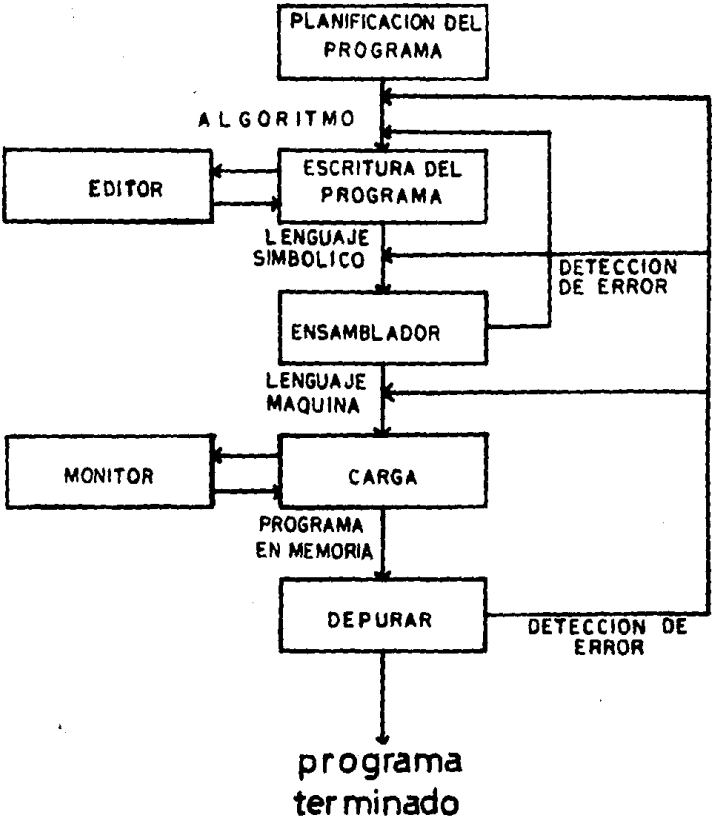


Fig.1 Pasos comunes para programar un microcomputador

llamado editor se prepara y modifica el programa inicial; el lenguaje ensamblador tiene cuatro campos:

1. Etiquetas
2. Mnemónico de la Instrucción
3. Operando
4. Comentarios

Para especificar direcciones se usan mnemónicos en lugar de números y para referencia a instrucciones, etiquetas en lugar de direcciones absolutas. Ejemplo de un Programa Fuente:

ETIQUETA	MNEMONICO DE LA INSTRUCCION	OPERANDO	COMENTARIO
	LD	A,X	;Cargar el acumulador con el byte X a probar
	LD	C,08H	;Cargar C como contador de bit
CHECA:	DEC	C	;Actualiza el contador de bit
	JP	M,FIN	;Todos los bytes comprobados?
	RL	A	;Desplaza el próximo bit más significativo al acarreo
	JP	NC,CHECA	;Si es cero prueba el próximo bit
FIN:	RST	38H	;Devuelve el control al sistema operativo

El programa anterior encuentra el bit de mayor orden - que no sea cero, en el acumulador, utilizando la rotación iz--- quierda a través del acarreo.

Se traduce el Programa Fuente a Objeto (lenguaje máquina) con direccionamiento absoluto o relocalizable; luego con ayuda - de un programa monitor se prueba el buen funcionamiento (se almacena en algún área de memoria para ejecutarse) y si es posible se reduce y mejora.

2.- ***ARQUITECTURA:***

La Unidad Central de Proceso ejecuta un programa que reside en alguna área de memoria; cada instrucción secuencial es leída desde memoria poniendo la dirección contenida en el registro "Contador de Programa" en el bus de direcciones, generando las señales apropiadas de control sobre el bus de control para activar la memoria, y después leer el dato del bus de datos -- dentro del registro indicado de la CPU. La señal de tiempo debe ser precisa para asegurar que la operación se realiza ade--

cuadamente. La función de control de la CPU coordina esta tarea y asegura que el Código de Operación (OP) de la instrucción sea puesto en el registro de intrucciones y codificado correctamente. También, ésta función controla la Unidad Lógica-Aritmética (ALU) en la ejecución de todas las operaciones lógicas y aritméticas contenidas en el grupo de instrucciones del Microprocesador Z80. Estas instrucciones incluyen:

- **Operaciones Lógicas:** Lógica AND
  Lógica OR
  Lógica XOR
  Comparación
  Borra un Bit
  Prueba un Bit
  Corrimiento y rotación a la izquierda y derecha.

- **Operaciones Aritméticas:** Adición
  Sustracción.

2.1.- **La Unidad Lógico-Aritmética,** cuando está llevando a cabo alguna operación, se comunica mediante el bus de datos interno con los veintidos registros, el registro de instrucciones y el controlador del bus de datos. El controlador de los buses y dirección vigila todas las actividades relacionadas con el in-

tercambio de datos entre la CPU y el exterior, mediante los buses respectivos. Hay que hacer notar que el bus de datos es bi direccional y el bus de dirección es unidireccional y hacia afuera de la CPU. La figura 2 es un diagrama de bloques de la CPU.

Un registro es el área de almacenamiento a corto plazo -- que tiene la CPU y cuya capacidad es de una o dos palabras. Existen varios tipos de registros en el Z80 y están divididos en dos clases: los que están accesibles al programador y los que son direccionados por el Microprocesador cuando está ejecutando algún programa.

Los veintidos registros están configurados de la siguiente manera:

- ***18 Registros de 8bits cada uno;** de los cuales 12 pueden ser utilizados por pares y formar 6 de 16 bits.*
- ***4 Registros de 16 bits.***

La figura 3 muestra la estructura e indica la función de cada bloque de registros. Los doce registros que pueden ser u

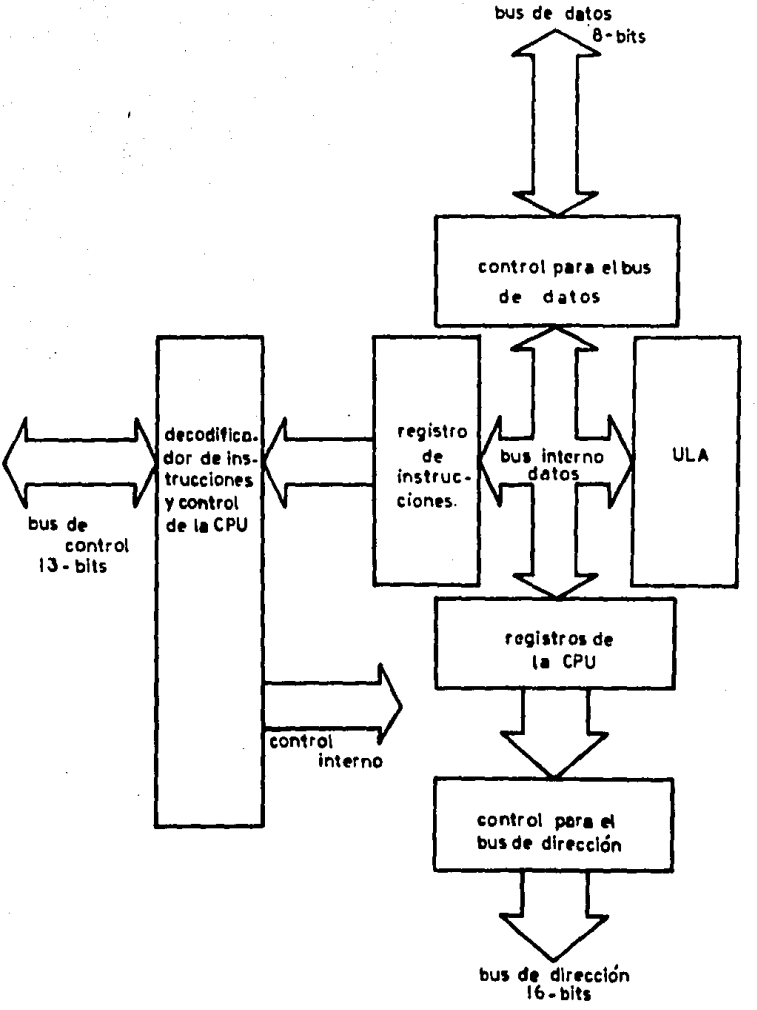


Fig. 2: Diagrama de Bloques de la Unidad Central de Proceso.

Registros de Propósito General y Acumulador

Principales		Alternos	
A	F	A'	F'
B	C	B'	C'
D	E	D'	E'
H	L	H'	L'

Registros de Propósito Específico	
I	R
IX	IY
PC	SP

Fig. 3: Configuración de los Registros de la CPU.

tilizados individualmente o por pares (registros de propósito general) están divididos en dos grupos:

- *El grupo de registros principales BC, DE y HL.*
- *El grupo de registros alternos BC', DE' y HL'.*

En cualquier momento el programador puede seleccionar cada conjunto de registros con un solo comando para intercambiar de grupo completo. Cuando una respuesta rápida al requerimiento de interrupción es necesaria, un conjunto de estos registros de propósito general con los registros acumulador/bandera pueden ser reservados y dar el servicio de una forma eficaz; evitando salvar o recuperar los contenidos de una área de memoria externa a la CPU; ésta área se llama almacén (stack).

Los registros restantes son de propósito especial 4 de 16 - bits y 6 de 8 bits:

- *Acumulador A y A'*
- *Bandera F y F'*
- *Vector de Interrupción I*
- *Renovar Memoria R*
- *Registros Indice IX e IY*
- *Contador de Programa PC*
- *Apuntador de Stack SP*

2.2.- Acumulador y Banderas.

El acumulador es un registro que mantiene los resultados de operaciones aritméticas y lógicas de ocho bits, mientras que las banderas indican condiciones especificas para las operaciones de ocho o diez y seis bits, tales como indicar si el acumulador, como resultado de una operación es cero, negativo, etc. Cuando se selecciona el registro A, implica la selección del registro F y asociado a A' está F'. El registro de banderas también llamado palabra de estado del programa (Program Status Word), es un conjunto de ocho bits agrupados de la siguiente manera:

BIT

7	6	5	4	3	2	1	0
S	Z	X	H	X	P/V	N	C

S - Signo
Z - Cero
X - Indeterminado
H - Medio Acarreo en BCD
P/V - Paridad/Sobreflujo
N - Sustracción BCD
C - Acarreo.

2.3.- Registro para el Direccionamiento del Vector de Interrupción (I).

La CPU Z80 puede ser usada en un modo, donde un llamado indirecto a cualquier localidad de memoria puede ser ejecutado, en respuesta a una interrupción. En este registro se almacena la parte alta (del bit 8 al 15) de la dirección indirecta, mientras que el dispositivo que interrumpe proporciona la parte baja (del bit 0 al 7). Esto permite a las rutinas de interrupción ser localizadas dinámicamente en cualquier área de memoria, con un tiempo de acceso absoluto mínimo.

*2.4.- **Registro para Renovar Memoria (R).***

Este registro es de ocho bits, siete de los cuales son un contador ascendente que se incrementa en uno cada vez que la CPU realiza el ciclo de ir a buscar y traer la información. El octavo bit permanece programado por el resultado de la instrucción LD R,A.

El dato que existe en el contador es enviado hacia afuera

de la CPU, por el bus de dirección junto con una señal de control llamada RFSH (renovación o refresco de memoria) enviada también por la CPU. Este registro simplifica el uso de la memoria dinámica, haciendo casi tan fácil su uso, como el de la memoria estática.

2.5.- Registros de Indice (IX E IY).

Estos registros mantienen una dirección base de diez y seis bits para usarse en el modo de direccionamiento indexado. En este modo de direccionamiento indexado, un registro de índice mantiene una base para indicar una región de memoria donde será almacenado o retirado un dato. Un byte en la instrucción o instrucciones que maneja este direccionamiento, indica el desplazamiento que tiene el número de diez y seis bits. Una aplicación para este modo de direccionar es el manejo de tablas de datos.

2.6.- Contador de Programa (PC).

Es un registro que mantiene direcciones de diez y seis bits,

correspondientes a la instrucción que está siendo traída desde - memoria para ser codificada por la CPU. Este registro se incre menta automáticamente después de que su contenido se vacía en el bus de direcciones. Cuando ocurre un salto en el programa el nuevo valor de dirección a ejecutar es cargado en este registro.

2.7.- Apuntador de Stack (SP).

Es un registro de diez y seis bits para almacenar una dirección de memoria (área de memoria externa a la CPU) en cuya localidad se mantiene un dato o una serie de datos que serán u tilizados posteriormente. Cada vez que se requiere de este regis tro, la dirección anterior al requerimiento es decrementada en u no; entonces, la organización de memoria es: el primer dato sal vado será el último en salir (se acostumbra utilizar la última - localidad de memoria o la parte más alta, como la primera de - almacenamiento), y el primer dato en salir es el último que se almacenó. Gracias a este registro es factible la implementación de múltiples niveles de interrupción, ilimitado número de encade

namiento de subrutinas y simplificación en algunos tipos de manipulación de datos.

3.- CICLO DE INSTRUCCION:

Cada instrucción del Z80 consiste en una serie de operaciones básicas llamadas ciclos de máquina; hay solamente siete ciclos de máquina que puede llevar a cabo la CPU Z80 y son:

1.- Búsqueda del código de operación de la instrucción.

2.- Ciclo de lectura o escritura de un dato en memoria.

3.- Ciclo de lectura o escritura en un puerto de E/S.

4.- Ciclo de Reconocimiento/Requerimiento de interrupción.

5.- Ciclo de Reconocimiento/Requerimiento del bus (disponibilidad).

6.- Ciclo de Reconocimiento/Requerimiento de interrupción no -- mascarillable.

7.- Instrucción para salir del estado de ALTO (HALT).

Cada instrucción puede tomar desde tres hasta seis períodos de reloj para completarse o pueden ser alargados para bus

car sincronía de velocidad con dispositivos externos. Estos perío dos de reloj son llamados ciclos T, así las operaciones básicas son un conjunto de ciclos T y cada operación constituye un ci-- clo de máquina, representándose con M1.

Cuando se ha completado la ejecución de una instrucción, se tiene un ciclo de instrucción terminado; así, cada ciclo de - instrucción está hecho por ciclos de máquina, los cuales compren den a la vez un número determinado de ciclos T. El primer -- byte de cada instrucción es un código de operación, por lo tan- to el 280 siempre lleva a cabo por lo menos un ciclo M1 en la ejecución de cada instrucción. Mientras hay diferentes tipos de ciclos de máquina, solamente hay un tipo de ciclos T, es decir, una transición de tiempo (señal de reloj) de 250 nanosegun-- dos de duración en 4MHZ; para un cambio en la señal de lógica cero a lógica uno y a lógica cero.

La figura 4 muestra cómo una típica instrucción es sola-- mente una serie de ciclos específicos M1 y una cantidad determi nada de ciclos T.

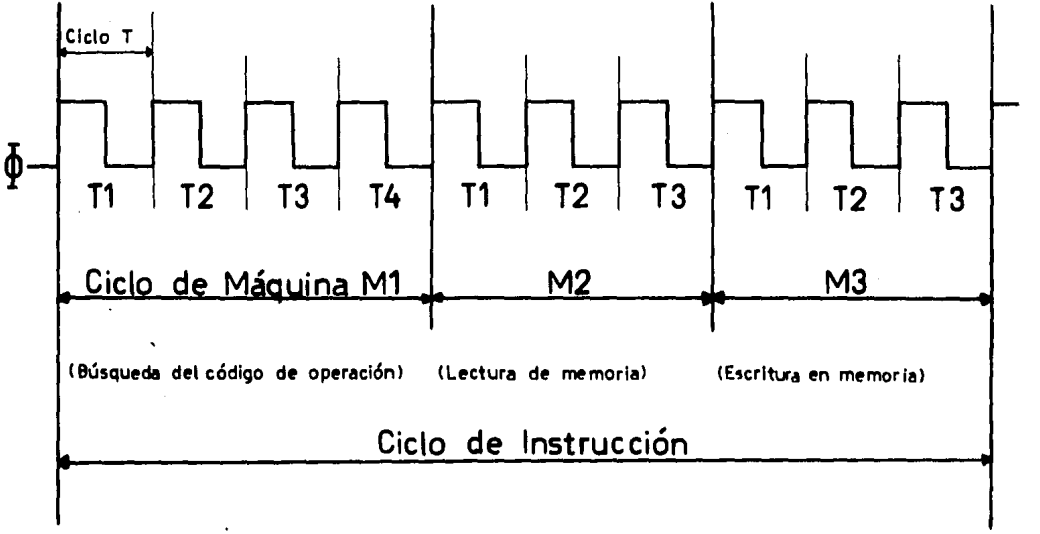


Fig.4 Ciclo de instrucción básico

La forma para calcular el tiempo en segundos de la ejecución de una instrucción y por lo tanto de un programa, varía - de acuerdo a la velocidad de la CPU: sólo hay dos tipos:

CPU Z80 a 2.5 MHZ
CPU Z80A a 4 MHZ

Nuestro sistema trabaja con la primera, por lo tanto cada ciclo T dura $1/2.5X10^{6}$ seg.= 0.0000004 seg.

En las tablas del apéndice, para cada instrucción se indica el número de ciclos T. Con la ayuda de éstas podemos hacer los siguientes cálculos a manera de ejemplos:

Supongamos:

INSTRUCCION	NUM. CICLOS T	NUM. VECES EJECUTADA	TIEMPO
LD A,36H	7	1	2.8
LD B,49H	7	1	2.8
OR B	4	1	1.6
AND 99H	7	1	2.8
RL A	4	1	1.6

El tiempo de ejecución para la secuencia anterior de instrucciones de 11.6 microsegundos. Si existiera alguna instrucción de salto condicionada, se deben considerar las instrucciones tantas veces como sean ejecutadas.

INSTRUCCION	NUM. CICLOS T	NUM. VECES EJECUTADA	TIEMPO
LD A,06H	7	1	2.8
LD B,08H	7	1	2.8
INC A	4	9	14.4
DEC B	4	9	14.4
JP NZ L	12(Condición - Alcanzada)	1	4.8
	7(Condición - No Alcanzada)	8	22.4

Consumiendo un total de 61.6 microsegundos.

El circuito (Fig. 5) que utilizamos para generar la señal de reloj está constituída por:

Y1 Cristal de Cuarzo

IC1 74LS74

IC2 74LS04

R1,R2 Resistencias de 1K Ohm y 1/4 de watt

R3 Resistencia de 330 Ohm y 1/4 de watt

C1 Capacitor de 0.01 microfaradios

C2 Capacitor de 10 picofaradios.

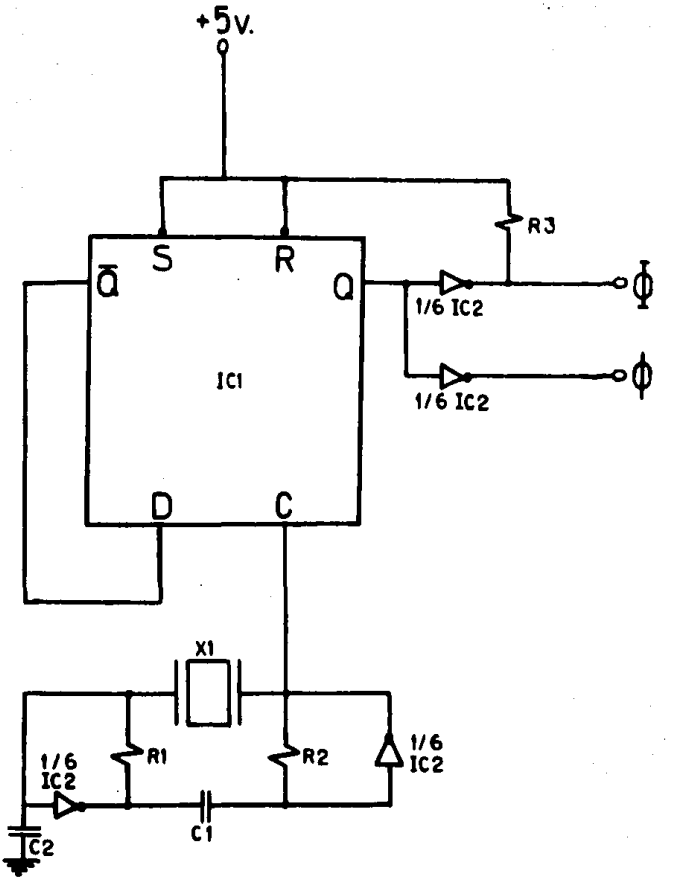


Fig. 5 Cto. generador de ciclos T.

4.- *INTERRUPCIONES:*

Una interrupción es la suspensión de la ejecución normal de un programa que está realizando la CPU, para atender a algún dispositivo interno o externo al sistema, o realizar una tarea específica, continuando con la ejecución del programa que estaba realizandose una vez que se ha atendido a dicho dispositivo o una vez realizada dicha tarea específica. Las interrupciones en el Z80 sirven entonces, para el mismo propósito que una interrupción en otros microcomputadores o computadores y así la suspensión asociada a un evento externo puede ser una transferencia de datos de E/S, o una función de tiempo para el microprocesador, o por condicones externas anormales.

Cuando las interrupciones están asociadas con la transferencia de datos de E/S, el mecanismo de interrupción sobrepone el tiempo de procesamiento de la CPU a las actividades de E/S. Si se requieren actividades múltiples de E/S, el mecanismo de interrupción es similar, indicando cuál dispositivo es el que requiere atención mediante un vector de interrupción. El Z80 maneja 128

vectores de interrupción separados.

Para generar funciones de tiempo, es conveniente suministrar mediciones de intervalos de tiempo a la CPU. Típicamente estos intervalos de tiempo los genera una interface programable contadora de tiempo interrumpiendo a la CPU cada décimo de segundo. La CPU reconocerá esta interrupción como una interrupción de tiempo al ir a procesar la rutina de servicio, para ajustar un sistema de reloj y/o llevar a cabo funciones de tiempo; la salida de esta rutina se hace directamente al punto donde se suspendió el proceso del programa.

Cuando se presenta una condición anormal para el sistema, la CPU puede ser informada mediante una interrupción. Una -- condición típica es durante la pérdida de energía o la falla de una porción del sistema, dentro del sistema de tiempo real. Frecuentemente estas funciones se implementan usando interrupción no-mascarillable, puesto que debe ser reconocida inmediatamente y no demorarse hasta que el procesamiento corriente este completo.

El Z80 tiene este tipo de interrupción especial, y es llama

da NMI (Nonmaskable Interrupt).

Cualquier sistema que maneja interrupciones tiene, por lo tanto dos tipos de interrupciones:

4.1.- No Mascarillable.

Este tipo de interrupción no puede ser ignorada bajo ninguna circunstancia. La primera tarea de una rutina de servicio de interrupción es salvar el estado de la CPU, así que la tarea interrumpida puede ser continuada en el punto donde se paró, cuando la rutina de servicio esté realizada.

4.2.- Mascarillable.

Cuando la CPU bajo control de programa puede decidir si la ignora o no. La CPU Z80 tiene una línea de interrupción mascarillable llamada INT.

Existe una intrucción DI (Deshabilita Interrupción) que provoca la desatención a todas las interrupciones mascarillables; la instrucción que reactiva el reconocimiento nuevamente es EI (Ha

bilita Interrupción).

Dentro de la extensa clasificación de las interrupciones -- mascarillables, están las categorías que relacionan el mecanismo de interrupción al manejo de éstas.

Por ejemplo: como la CPU y el dispositivo que está in-- terrumpiendo, comunican información importante:

Desde el dispositivo a la CPU.

A. Requerimiento de Servicio

B. Identificación, para que la CPU sepa qué servicio realizar.

Desde la CPU al dispositivo.

A. Reconocer la Interrupción

B. Otro dato necesario.

Existen tres tipos básicos para el manejo de las interrupciones:

- *Interrupción de una sola línea o interrupción de rastreo*
- *Interrupción de múltiples niveles*
- *Interrupción Vectorial.*

Interrupción de una sola línea. *En un sistema donde el -*

microprocesador tiene solamente una línea de interrupción, pueden ser conectados dispositivos múltiples de E/S a un arreglo de compuertas OR y conectar la salida de este arreglo a la entrada de interrupción del microprocesador (Fig. 6). Cada entrada del arreglo de compuertas OR, corresponde a un dispositivo de E/S, de tal forma que si un dispositivo solicita servicio, genera una señal -- para interrumpir la CPU.

Interrupción de Múltiples Niveles. *Es un sistema en el cual el procesador tiene varias líneas de entrada de interrupción, siendo una técnica muy efectiva, pero poco usual, pues son raros los microprocesadores que tienen más de cuatro entradas. En este sistema la CPU no necesita rastrear al dispositivo porque cada línea tiene ya una prioridad establecida (Fig. 7).*

Interrupción Vectorial. *Consiste en una línea para requerimiento de interrupción y un identificador que permite a la CPU pasar directamente a la rutina de servicio de la interrupción. -- Para el Z80, después del pulso de interrupción que se envía a la CPU, sigue un número codificado de ocho bits identificando al dispositivo que interrumpe. Generalmente el código de ocho bits in-*

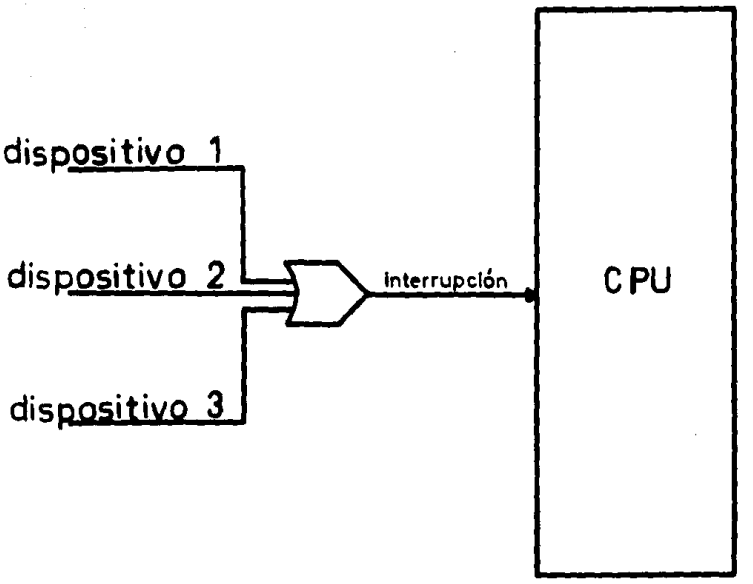


Fig.6 Interrupción de una sola línea.

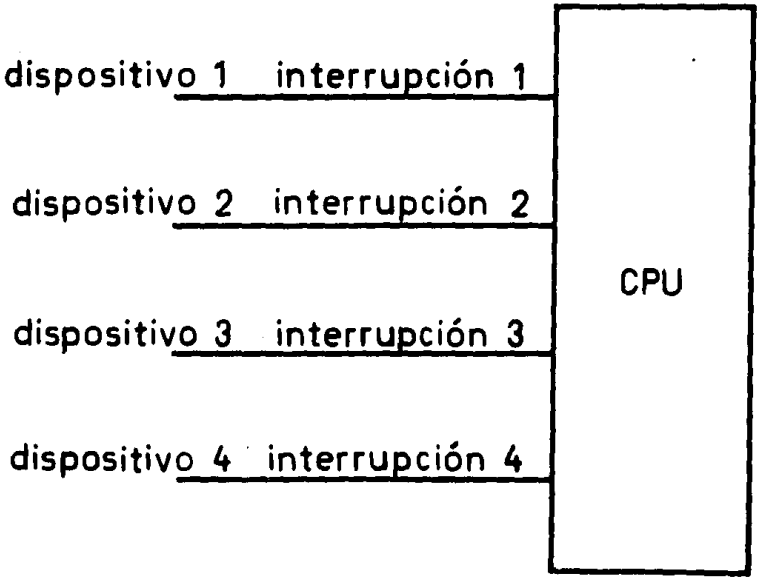


Fig.7 Interrupción de múltiples niveles.

dica una dirección, un punto para una dirección o el primer byte - de una instrucción (Fig. 8).

La CPU Z80 maneja interrupciones de una sola línea o vectoriales, mediante tres modos: Modo Cero, Modo Uno y Modo Dos.

Modo Cero. Utiliza el vector de interrupción y el byte de identificación; este último es interpretado como un simple código de operación, representando a la primera instrucción para ser ejecutada después del reconocimiento de la interrupción. Esta instrucción es un llamado a una subrutina o a una instrucción de restablecimiento a una sección del programa.

Modo Uno. Este modo maneja interrupciones de una sola línea, teniendo que establecer prioridades, si múltiples dispositivos de E/S son necesarios. Este modo provoca un salto a la dirección 0038H ya que la primera instrucción después de la respuesta a la interrupción así lo indica. El esquema de prioridades se puede controlar por software volviendose muy sofisticado, o por hardware realizandose muy rápido (Fig. 9).

Modo Dos. Este modo es el más eficaz porque permite hasta - 128 dispositivos externos para interrumpir a la CPU, cada uno total

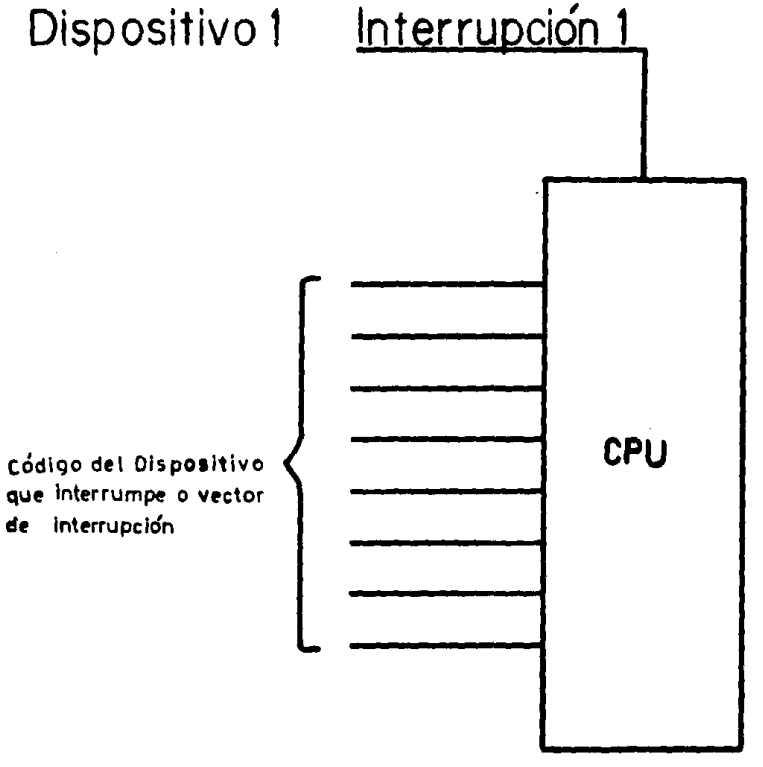


Fig. 8: Interrupción Vectorial

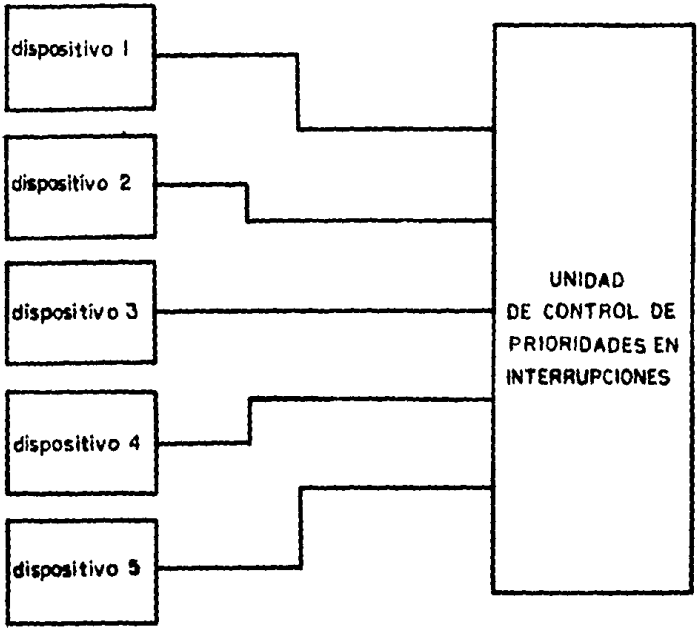


Fig. 9: Codificacion de Prioridades para Interrupción en Modo 0 ó Modo 1

mente independiente con una localización en memoria. La escencia de este modo de interrupción es una tabla de interrupción vectorial en cualquier lugar en la memoria. En general la tabla tiene una longitud de 2XN bytes, donde N es el número de interrupciones en el sistema y el principio de la tabla está apuntado por IIIIIIII-00000000 (Sistema Binario), donde I es el contenido del registro del vector de interrupciones (I). Para cualquier interrupción, el registro I proporciona los ocho bits más significativos de la dirección de la tabla, mientras que el dispositivo interrumpiendo proporciona los ocho bits menos significativos de esta. - Los 128 vectores estan mostrados en la figura 10.

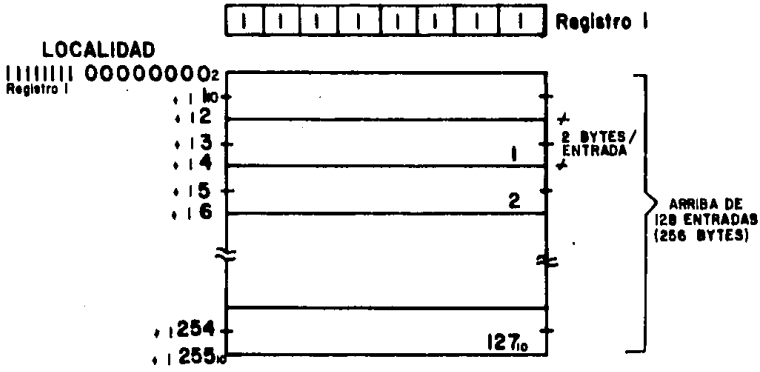
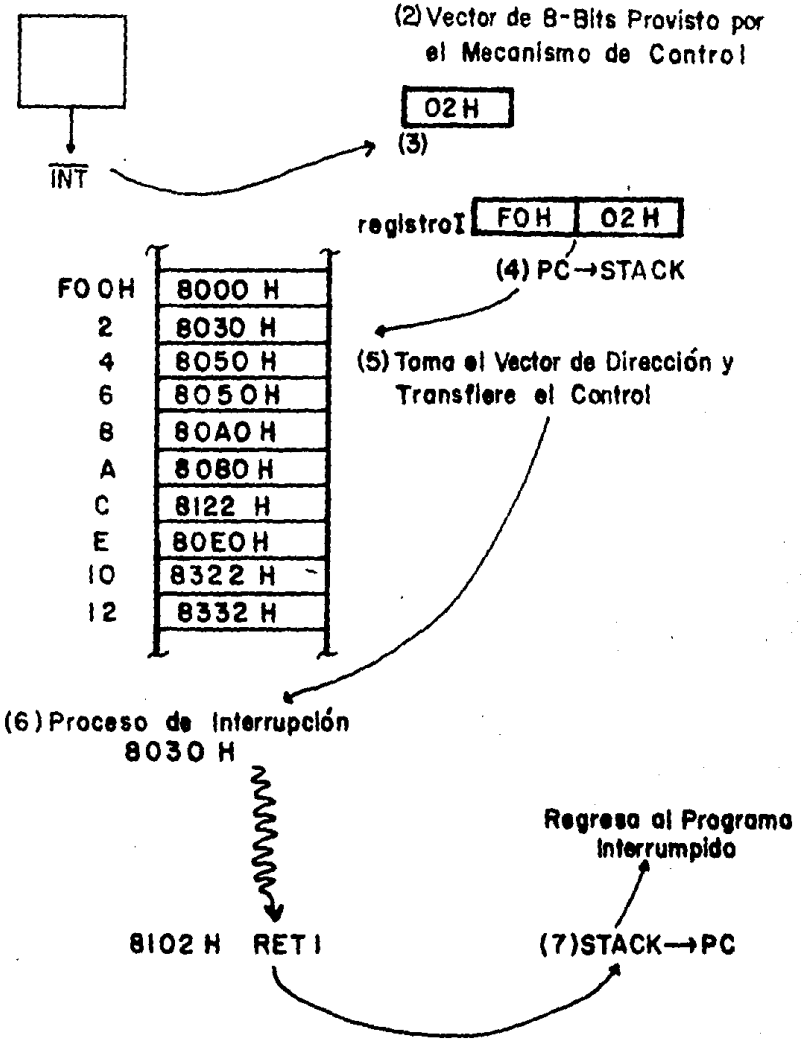


fig. 10: Los 128 Vectores de Interrupcion.


(2) Vector de 8-Bits Provisto por el Mecanismo de Control
02 H
(3)
INT
registroI
F0 H
02 H
(4) PC→STACK
F0 0H
2
4
6
8
A
C
E
10
12
8000 H
8030 H
8050 H
8050 H
80A0 H
8080 H
8122 H
80E0 H
8322 H
8332 H
(5) Toma el Vector de Dirección y Transfiere el Control
(6) Proceso de Interrupción
8030 H
Regresa al Programa Interrumpido
8102 H RETI
(7) STACK→PC

CAPITULO III

P I O

El circuito PIO (puerto de entrada/salida en paralelo), consta de dos puertos totalmente compatibles con la tecnología TTL -- para interface entre dispositivos periféricos y la CPU.

Una de las características del PIO que lo separa de los demás controladores de este tipo, es que la transferencia de datos se sujeta a un control total de interrupciones; implementando toda la lógica necesaria para ello.

1.- DESCRIPCION FUNCIONAL:

1.1.- Arquitectura.

La estructura interna del PIO consiste en un bus para interface directa con la CPU mediante las ocho líneas de dato, más las líneas de control. El bus de datos acarrea todos los comandos y datos entre la CPU y el PIO, mientras las líneas de control habilitan o activan el circuito ($\overline{CE}$), definen la naturaleza del byte - que se va a transmitir ($C/\overline{D}$), seleccionan alguno de los dos puertos ($B/\overline{A}$) y especifican la dirección del flujo de datos ($\overline{M1}$, $\overline{IORQ}$, $\overline{RB}$). El bus interno del PIO comunica a los dos puertos (A y B)

con el controlador de interrupciones y con la lógica de control interno además de conectar al bus para interface con la CPU. La figura 1 muestra el diagrama de bloques interno del circuito.

1.2.- Configuración de los Puertos A y B.

Los dos puertos de E/S son virtualmente idénticos y son usados para la interface directa con los dispositivos periféricos. Cada uno de ellos está compuesto por seis registros con lógica para control de "handshake" como se muestra en la figura 2. El registro para control de modo de operación es de dos bits que especifican la forma de trabajar del PIO (salida de un byte, entrada de un byte, bus bidireccional o bit para para control de modo) y es cargado en el programa por un comando de salida al PIO.

Todos los datos transmitidos al PIO de dispositivos periféricos deben pasar a través del registro de entrada de ocho bits y la información enviada por el PIO a un periférico debe pasar por el registro de salida de datos de ocho bits.

Las líneas para el control del "handshake" transfieren da--

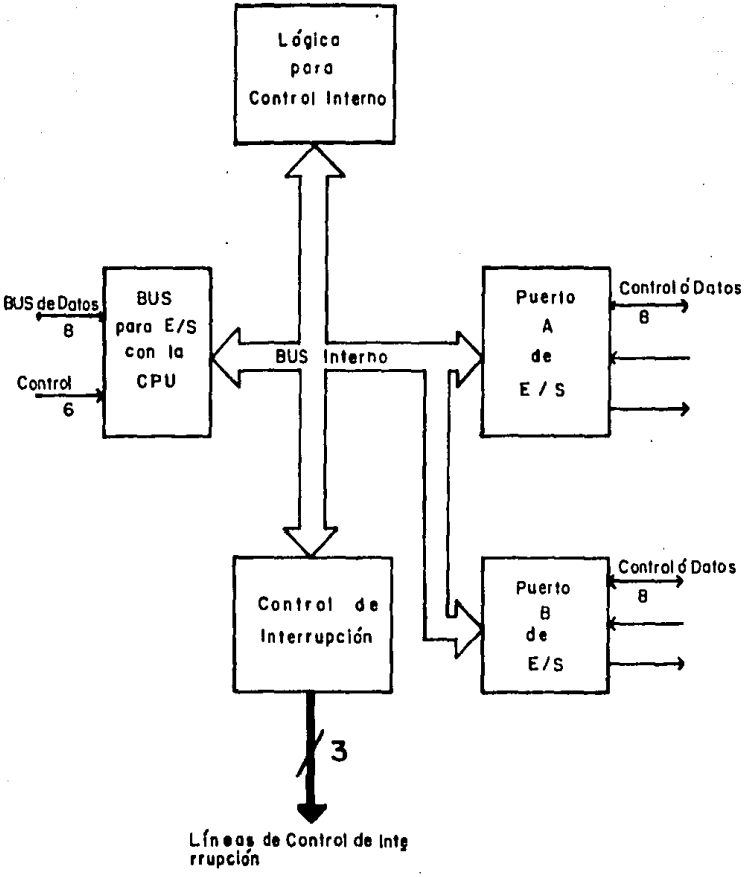


fig. 1 Diagrama de Bloques del Puerto de E/S en Paralelo.

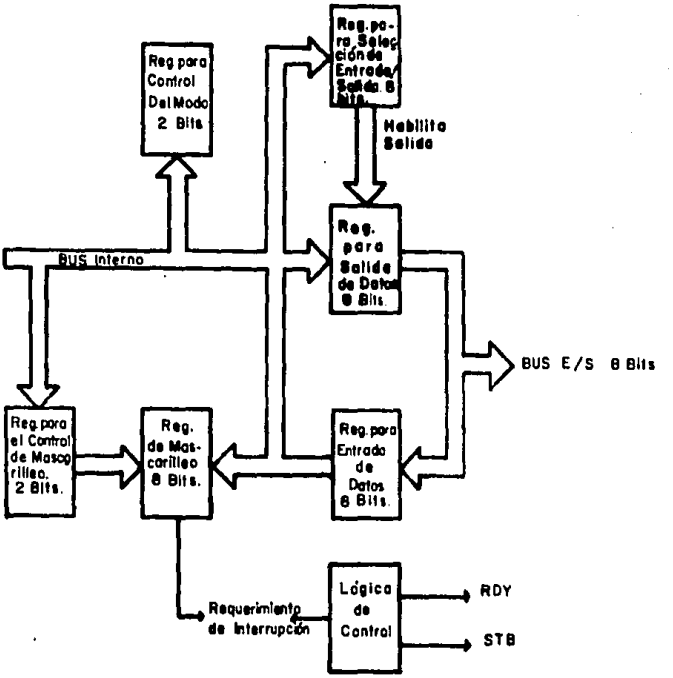


fig. 2 Diagrama de Bloques Típico de un Puerto del PIO.

tos indicando cuando el dato está listo para ser mandado o recibido.

El selector de registro de entrada/salida, el registro para - control de mascarilleo y el registro de mascarilleo son usados únicamente en el modo tres de operación del PIO que se caracteriza por:

1.- Cada bit, de los ocho del puerto, pueden ser especificados como entrada o salida de bits, cargando el registro de entrada-salida por programa, dando a cada bit el valor lógico de - uno si es línea de entrada o nivel lógico cero si es línea de salida.

2.- En el modo tres de operación, el PIO monitorea sus líneas de entrada por la presencia de un bit patrón. Cuando el - bit patrón está activado, el PIO genera automáticamente una interrupción a la CPU. La especificación del bit patrón es acompañada del registro de mascarilleo y del registro de control de mascarilleo. El primero especifica cuál subgrupo de las líneas de - entrada serán monitoreadas y cuáles se mascarillean (se ignoran). El registro de control de mascarilleo consiste en sólo dos bits, el primero de los cuales indica si una línea será considerada en --

estado activo con, cero o con uno lógico; el segundo bit especifica cuándo debe ser aplicada una función OR ó AND para una entrada no mascarillable como un criterio para generar una interrupción. En particular, un registro de control de mascarilleo cargado con los bits 01 especifica que una interrupción debería ser generada solo si todas las líneas de entrada no mascarilleables están en - cero lógico. Por otro lado, un 00 especifica que una interrupción sería generada sólo si existe una línea de entrada no mascarillable en cero lógico.

2.- *PROGRAMACION DEL PIO:*

2.1.- Estado de RESET.

En el estado de RESET (borrar), el circuito deja de recibir y transmitir información, y se mantiene en un estado inactivo hasta que recibe una palabra de control de la CPU que le indicará - la forma de funcionar.

El PIO entra automáticamente al estado de RESET cuando se le aplica energía; este estado lleva a cabo las siguientes funciones:

1.- En ambos puertos el registro de mascarilleo entra al estado de Reset para inhibir todos los bits de datos en los puertos.

2.- Las líneas del bus de datos de los puertos son puestas en estado de alta impedancia y las señales de "handshake" son desactivadas. El modo 1 es automáticamente seleccionado.

3.- El vector de dirección en los registros no entra al estado de RESET.

4.- En ambos puertos el flip-flop que habilita interrupciones entra al estado de RESET.

5.- Los registros de salida de los puertos también entran al estado de RESET.

Si es necesario que el PIO realice un borrado interno evitando la secuencia apagado-encendido, se activa la señal $\overline{MI}$ sin activar $\overline{RD}$ ni $\overline{IORQ}$.

2.2.- Vector de interrupción.

El PIO ha sido diseñado para operar con la CPU usando el modo 2 en respuesta a las interrupciones. Este modo requiere que

un vector de interrupciones sea proporcionado por el dispositivo que genera la interrupción. Este vector es usado por la CPU para formar la dirección de la rutina de servicio de este puerto. Este vector es puesto sobre el bus de datos durante un ciclo de reconocimiento de interrupción por el dispositivo de mayor prioridad que está requiriendo servicio en ese tiempo. El vector es cargado dentro del PIO por una palabra de control escrita con el siguiente formato y sobre el puerto deseado:

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

El cero en D0 (bit bandera) significa que la palabra de control es un vector de interrupción, y provoca que los valores V7 a V1 sean cargados con el contenido del registro de vector de interrupción.

2.3.- Selección del Modo de Operación.

El puerto A del PIO puede ser operado en cualquiera de cuatro formas distintas: Modo 0 (modo de salidad), Modo 1 (modo de entrada), Modo 2 (modo bidireccional), Modo 3 (modo de control). El -

puerto B opera solamente en Modo 0, Modo 1, Modo 3. El modo de operación debe ser establecido escribiendo la palabra de control en el PIO con el siguiente formato:

D7	D6	D5	D4	D3	D2	D1	D0
M1	M0	X	X	1	1	1	1

Donde M1 y M0 seleccionan el modo de acuerdo a la siguiente -- tabla:

D7	D6	Modo
0	0	0 (salida)
0	1	1 (entrada)
1	0	2 (bidireccional)
1	1	3 (control)

X - estos dos bits son ignorados.

Los bits D3 a D0 deben estar en 1 para indicar que se está seleccionando el modo.

Seleccionando el Modo 0 se habilita el registro de salida para que la CPU pueda escribir cualquier dato sobre el bus del puerto. El contenido del registro de salida puede ser cambiado en cualquier tiempo por la CPU escribiendo simplemente la nueva palabra de dato para

el puerto, o puede ser leído por la CPU si se ejecuta una instrucción de entrada. Un dato escrito por la CPU provoca que la línea de READY HANDSHAKE del puerto notifique al dispositivo periférico que está disponible.

Seleccionando el Modo 1, el puerto queda en modo de entrada; para empezar la operación de "handshake", la CPU hace una lectura de entrada desde el puerto, lo que activa la línea de READY para indicar al periférico que el dato puede ser cargado en el registro de entrada.

El Modo 2 es una transferencia bidireccional que usa las cuatro líneas de "handshake". Por lo tanto, el puerto A solamente puede ser usado en Modo 2, usando las líneas de "handshake" del puerto A como señales de control de salida y las líneas de "handshake" del puerto B como señales de control de entrada. La única diferencia entre el Modo 0 y la parte de salida del Modo 2, es que el dato del registro de salida del puerto A, es puesto en el Bus de datos del puerto, cuando la señal $\overline{ASTB}$ es activada.

El Modo 3 de operación está propuesto para aplicaciones de control y estado del puerto, sin utilizar las señales de "handshake"; cuando es seleccionado, la siguiente palabra de control enviada -

al PIO debe definir cuáles líneas del bus de datos del puerto serán - entradas y cuáles serán salidas. El formato de la palabra de control está mostrado a continuación:

D7	D6	D5	D4	D3	D2	D1	D0
I/O7	I/O6	I/O5	I/O4	I/O3	I/O2	I/O1	I/O0

Cada bit de la palabra de control puede ser entrada, si la línea correspondiente vale uno lógico, o puede ser salida si la - línea tiene el valor de cero; la señal READY se mantiene en cero lógico y el dato puede ser escrito en el puerto o leído desde él por la CPU en cualquier momento durante este modo cuando el -- puerto está leyendo; el dato que fué regresado a la CPU está formado por un dato de entrada en las líneas del bus de datos del puerto asignado como entrada más un dato en el registro de salida del puerto en líneas asignadas como salidas.

2.4.- Palabra para Control de Interrupción.

La palabra control de interrupción de cada pueto tiene el siguiente formato:

D7	D6	D5	D4	D3	D2	D1	D0
habilita interrup.	AND/ OR	ALTO/ BAJO	MASC	0	1	1	1

Si el bit D7=1, el flip-flop que habilita interrupción en el puerto está siendo activado para que el puerto pueda generar interrupción a la CPU, si es cero, la interrupción no puede ser generada. Si hay una interrupción pendiente cuando ésta bandera se habilita, se reflejará en la línea de interrupción de la CPU. Los bits D6, D5 y D4 son usados solamente en el Modo 3; sin embargo, activado el bit D4 de la palabra de control de interrupción durante cualquier modo, las interrupciones pendientes serán ignoradas. Estos tres bits en el Modo 3 son usados para controlar las interrupciones de un grupo de dispositivos de E/S cuando van a ciertos estados definidos; el bit D6 (AND/OR) define la acción lógica que ejecutará el puerto que está monitoreando, la función AND cuando el bit es uno lógico y la función OR cuando es cero; el bit D5 define la polaridad que debe considerar el puerto para que sean activas o no las líneas del bus de datos que se está monitoreando, considerando que si el bit es uno lógico - serán monitoreados con el estado alto, y si es cero lógico con estado bajo; si el bit D4=1 la siguiente palabra de control enviada al puerto debe definir el mascarilleo como sigue:

D7	D6	D5	D4	D3	D2	D1	D0
MB7	MB6	MB5	MB4	MB3	MB2	MB1	MB0

Solamente aquéllas líneas del puerto cuyo bit de mascari-- lleo es cero, deberán ser monitoreadas para generar una interrupción.

3.- APLICACION DEL PIO:

Este puerto está utilizado como un puerto serie, esto es: - valiéndonos de la versatilidad de este dispositivo, decidimos u- sarlo como convertidor paralelo-serie (salida) y serie-paralelo - (entrada), conectado a una interface de comunicación RS-232 y - lograr el uso de cualquier terminal. Para esta conversión se - diseñó una subrutina que, primero inicializa al puerto; después le envía la palabra de control para especificar el modo de operación (Modo 3); a continuación se define qué bit será entrada y cuál será salida. La subrutina procede a leer la información - que pueda contener el bit de entrada y organizarla para formar un byte. Cuando la información es hacia la terminal, el byte - por transmitir se recorre varias veces para que salga en serie.

4.- *DESCRIPCION FUNCIONAL DEL CTC.*

El circuito contador, temporizador (CTC-Z80), es un dispositivo programable de cuatro canales que puede funcionar como --- contador o como generador de frecuencias. La CPU configura a los cuatro canales del CTC independientemente para que operen - bajo las condiciones requeridas.

Algunas características:

- *Cada canal puede ser seleccionado en uno de do modos de operación; en el modo de contador, o en el modo de generador de frecuencia (temporizador).*
- *Interrupciones programables para cada modo.*
- *Registro de tiempo constante recargable automáticamente, para cuenta descendente hasta cero y en forma cíclica.*
- *Posibilidad de leer el valor del contador descendente en cualquier momento.*
- *Reloj divisor por diez y seis o por doscientos cincuenta y -- seis en cada canal para el modo de temporizador.*
- *Inicialización del temporizador con disparo positivo o negativo.*
- *Tres canales con las salidas ZC/TO (cuenta cero/temporizador)*

capaces de manejar Transistores Darlington.

- Salidas y entradas compatibles con la lógica TTL.
- Lógica de interrupciones vectoriales, con las prioridades preestablecidas y de selección automática.

4.1.- Arquitectura:

La estructura interna del CTC la conforman, un bus de E/S como interface para la CPU, lógica de control interno, cuatro canales contadores y lógica de control para las interrupciones vectoriales, siendo el canal cero el de más alta prioridad. Todas las partes están interconectadas por un bus interno bidireccional, fig. 1.

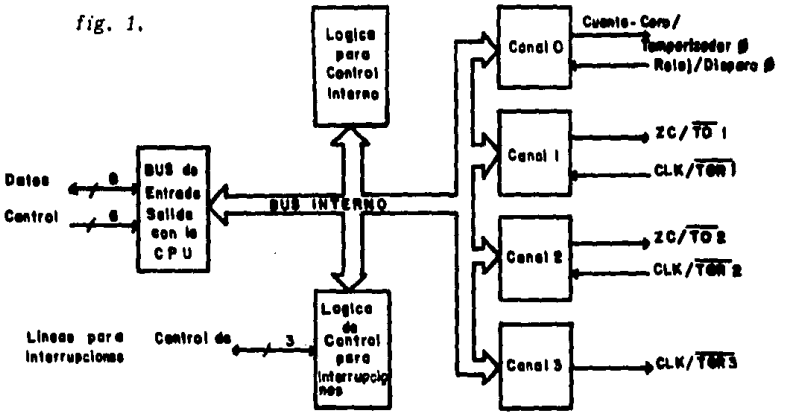


fig. 1 Diagrama de Bloques.

4.2.- Canales del CTC:

Cada canal está formado por un registro de tiempo constante, un contador descendente, un reloj divisor o temporizador y un registro para control del canal. En la figura 2 mostramos la interconexión de cada una de las partes que constituyen al canal.

Función de cada componente del canal:

- *El contador descendente; es de ocho bits, que recibe del registro de tiempo constante bajo control de programa y marca el punto a partir del cual empezará la cuenta descentende hasta cero, manteniendo un ciclo. En cualquier momento la CPU puede leer el valor del contador descendente que está dividiendo la señal que existe en la línea de entrada CLK/$\overline{TRIG}$ (reloj/disparo). Cuando el canal no está trabajando como contador el reloj divisor es el encargado de proporcionar la señal para que cuente descendentemente.*
- *El registro de tiempo proporciona ocho bits al contador para que empiece su cuenta. Esta palabra la genera la CPU.*

- *El reloj divisor o temporizador divide la señal de reloj del sistema de microcomputador por diez y seis o por doscientos cincuenta y seis según sea el caso.*

- *Registro para control del canal; la CPU genera una palabra para indicar las condiciones de operación al o a los canales y para seleccionar el modo de trabajo.*

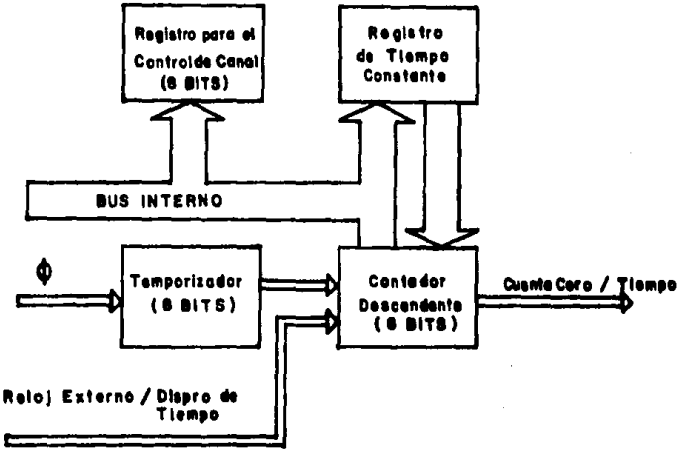


fig. 2: Diagrama de Bloques del Canal.

5.- *PROGRAMACION Y APLICACION DEL CTC.*

Para programar a este dispositivo, la CPU le envía tres -- palabras que son, para cargar: el control del canal, el tiempo -- constante y el vector de interrupción y con esto, definir el modo de operación. En estos momentos la CPU no manda una señal específica de escritura, por lo que la CTC la genera internamente a partir del valor de la señal de entrada $\overline{RD}$. Cuando está en modo de operación cuenta descendente, entonces la señal $\overline{RD}$ indicará al CTC el instante deseado para leer el valor del contador, si así se requiere.

A continuación presentaremos cada palabra de programación especificando sus características y funciones.

5.1.- *Modo de operación:*

Cuando se selecciona el modo de operación de algún canal, la palabra que envía la CPU tiene el siguiente formato:

D7	D6	D5	D4	D3	D2	D1	D0
B7	B6	B5	B4	B3	B2	B1	B0

Donde:

D7 Habilita interrupciones.
D6 Modo.
D5 Rango.
D4 Disparo positivo o negativo.
D3 Disparo externo.
D2 Tiempo constante.
D1 Borra o para operación.
D0 Uno lógico.

Cada uno de los bits puede tener dos estados lógicos (uno o cero) y de acuerdo al valor tendremos:

B7 = 0, se deshabilitan las interrupciones.

B7 = 1, habilita las interrupciones en el canal cada vez que llega a cero el contador descendente. La interrupción no será reconocida cuando el contador este en algún número diferente de cero.

B6 = 0, modo temporizador; el contador descendente es manejado por el reloj divisor y la frecuencia del contador puede ser:

tc = período de reloj del sistema.
P = reloj divisor (16 ó 256).
TC = Tiempo constante programable en un número binario de ocho bits (máximo 256).

B6 = 1, modo contador; el contador descendente es manejado por un reloj externo.

B5 = Ø, modo temprizador; el reloj del sistema es dividido por diez y seis por el reloj divisor.

B5 = 1, modo temporizador; el reloj del sistema es dividido por doscientos cincuenta y seis por el reloj divisor.

B4 = Ø, modo temprizador; indica el inicio de la operación con disparo negativo.

B4 = 1, modo temprizador; indica el inicio de la operación con disparo positivo.

B3 = Ø, modo temporizador; comienza la operación temporizadora en la transición ascendente del segundo período del ciclo de máquina, continuando con el ciclo que carga el tiempo constante.

B3 = 1, modo temporizador; valída un disparo externo para inicializar la operación, después del segundo período del ciclo de máquina, continuando con el ciclo que carga el tiempo constante. El reloj divisor es decrementado posteriormente, dos ciclos de reloj.

B2 = Ø, un tiempo constante debe ser cargado o escrito en el canal para empezar la operación. Después de cargar la palabra de control, no seguirá un tiempo constante.

B2 = 1, el tiempo constante para el contador descendente se escribe en el canal, después de la palabra de control; si éste es cargado mientras un canal está contando, tendrá que terminar la cuenta para que el tiempo nuevo afecte al contador.

B1 = 0, el canal continúa contando.

B1 = 1, para la operación. Si el bit B2 = 1, el canal estará disponible para una nueva palabra de control y cambiar el modo de operación.

B0 = 0, la palabra es un vector de interrupción.

B1 = 1, indica que es una palabra de control.

5.2.- Programar un tiempo constante:

Antes de que un canal pueda empezar a contar, debe recibir una palabra de tiempo constante de la CPU. Durante la programación o reprogramación, una palabra de control en la cual el B2 = 1, deberá preceder a la palabra de tiempo constante. La palabra de tiempo constante puede tener cualquier valor entre uno y doscientos cincuenta y seis; el valor binario 0000 0000 equivale a doscientos cincuenta y seis.

El formato de ésta es:

D7	D6	D5	D4	D3	D 2	D1	DØ
TC_7	TC_6	TC_5	TC_4	TC_3	TC_2	TC_1	TC_0

En el modo temporizador, el intervalo de tiempo es contro-- lado por tres factores:

- *El período de reloj del sistema.*
- *El factor del reloj divisor.*
- *El tiempo constante, el cual es programado en el registro de tiempo constante.*

5.3.- Programar el vector de interrupción:

El CTC debe ser pre-programado con los cinco bits más significativos de la palabra vector de interrupción; ya que la CPU generará la dirección de la rutina de servicio para el canal, usando esta información. Entonces, durante un ciclo de reconocimiento de interrupción, la CTC entrega a la CPU un vector de -- interrupción, por medio del canal con más alta prioridad. El - formato de la palabra vector de interrupción es:

D7	D6	D5	D4	D3	D2	D1	DØ
V7	V6	V5	V4	V3	X2	X1	Ø

Donde:

V7 a V3 son propiamente el vector de interrupción y es información proporcionada por el usuario.

X3 - X2, identificador del canal; la proporciona automáticamente la CTC, con el siguiente código:

X3	X2	-	número de canal
Ø	Ø		canal cero
Ø	1		canal uno
1	Ø		canal dos
1	1		canal tres.

DØ, será siempre cero, porque de otra manera, la palabra sería para control del canal.

CAPITULO IV

DISPOSITIVOS DE ALMACENAMIENTO

Dentro de los dispositivos de almacenamiento podemos mencionar: las cintas magnéticas, cintas perforadas, tarjetas perforadas, discos magnéticos duros y flexibles y las memorias propiamente.

El objetivo de este capítulo, es hacer un breve estudio de los tipos de memoria que manejan los Sistemas de Microprocesador; realizando una clasificación de ellas, según su forma de acceso, el tipo de operaciones que pueden realizar, etc.; comparando sus tiempos de acceso y su tiempo de ciclo lectura-escritura, principalmente de memorias RAM, de diferentes marcas y con diferentes fabricantes.

Definiremos la organización de la memoria de trabajo del programa monitor de nuestro Sistema de Microprocesador y propondremos la estructura del Hardware de las memorias RAM y EPROM que pretendemos utilizar en el Microcomputador Educativo.

Trataremos a las memorias como subsistemas destinados a -- almacenar los programas y datos para suministrarlos a otros dispositivos (normalmente a la CPU).

Una memoria se caracteriza fundamentalmente por:

1.- *Un conjunto de celdas capaces de almacenar en cada una -- de ellas un bit de información y que se organiza en conjunto de palabras normalmente de uno, cuatro, ocho o dieci--- séis celdas.*

2.- *Un dispositivo de acceso que nos permite la lectura de la - información contenida en una palabra y/o su modificación.*

Según su forma de acceso las memorias se clasifican como:

a) De acceso directo o aleatorio

b) De acceso secuencial

c) Asociativas.

a) *De acceso directo o aleatorio.- En estas memorias se asocia una dirección a cada palabra, y al suministrar a la memoria una dirección, determina que se suministre o modifique la información de la palabra asociada a dicha dirección en un tiempo que no depende del valor de la dirección.*

b) *De acceso secuencial.- En estas memorias el tiempo de acceso a una palabra determinada depende de su posición con respecto a una posición de referencia. El dato es accesible mediante una secuencia temporal.*

c) *Asociativas.- En estas memorias el acceso a una palabra determinada se consigna mediante la información contenida en una parte de la palabra.*

Según las operaciones que se puedan efectuar con la información contenida en sus palabras, las palabras se clasifican en:

1.- *Memorias Vivas.*

2.- *Memorias Muertas*

1.- *Vivas.- Se puede leer y modificar el valor de las palabras.*

2.- *Muertas.- Se puede leer el valor de una palabra, pero no se puede modificar.*

Cuando al leer una palabra se destruye la información que contienen sus celdas, le llamaremos Memoria Destructiva y en caso contrario No Destructiva. Las memorias destructivas serán --- siempre vivas; a todo ciclo de lectura debe seguir un ciclo de -- escritura.

Una memoria es volátil si se necesita suministro de energía para mantener la información. En caso contrario no es volátil.

Memorias dinámicas son aquéllas en que la información almacenada se degenera en el tiempo, aunque esten alimentadas y es preciso un refresco para que no se pierda la información. En caso contrario, las llamaremos Estáticas.

1.- MEMORIAS DE ACCESO DIRECTO. RAM, ROM, PROM, EPROM y EEMPROM.

Dentro de las memorias de acceso directo tenemos dos tipos:

- *Vivas*
- *Muertas.*

Las memorias vivas, como ya dijimos, son aquéllas que pueden modificar su contenido. Esto significa que en el momento que se desee, podremos variar su información o cuando suceda una interrupción o falta de energía eléctrica el valor anterior a la falla se perderá. Normalmente se conoce a este tipo de memoria como RAM (Random Access Memory).

Las memorias muertas son aquéllas en las que el contenido se graba de alguna manera y ya no puede ser modificado, más - que por medios externos. Tenemos los siguientes tipos:

ROM (Read Only Memory)

PROM (Programable Read Only Memory)

EPROM (Erasable Programable Read Only Memory)

EEPROM (Electrically Erasable PROM)

Además de la clasificación anterior, es muy importante mencionar que existen diferentes métodos o técnicas de fabricación y que influyen directamente en el comportamiento eléctrico de las -- memorias.

Brevemente podemos mencionar dos tecnologías muy comunes:

- *Bipolar*
- *MOS*

La tecnología de fabricación Bipolar se basa en los transistores (TBJ) tradicionales multiemisor conectados como circuitos biestables.

La tecnología MOS basa su funcionamiento en transistores de efecto de campo (FET).

1.1 Memorias Vivas RAM (Random Access Memory). Describiremos a las Memorias RAM bipolares que además de ser de acceso directo son vivas, no destructiva, no volátiles y estáticas. La célula de memoria RAM multiemisor que más se utiliza es la representada en la figura 1.

Si se mantiene la línea de palabra en cero lógico, no se lee ni escribe. Para leer se cambia esta línea a estado lógico alto o uno lógico.

Las memorias RAM-MOS también se basan en un elemento biestable, pero solamente se utilizan transistores de efecto de campo (FET), los cuales sustituyen los elementos de carga tradicionales y a los transistores bipolares.

La figura 2 es un elemento biestable con FET's y describe a una memoria RAM-MOS estática.

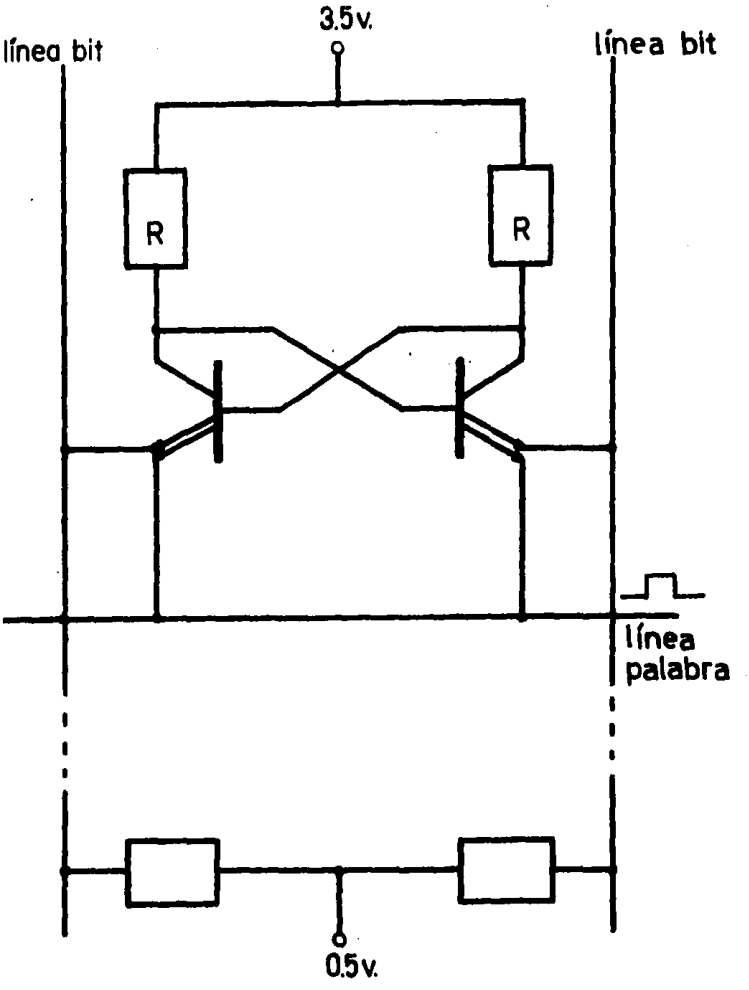


fig. 1 Célula RAM

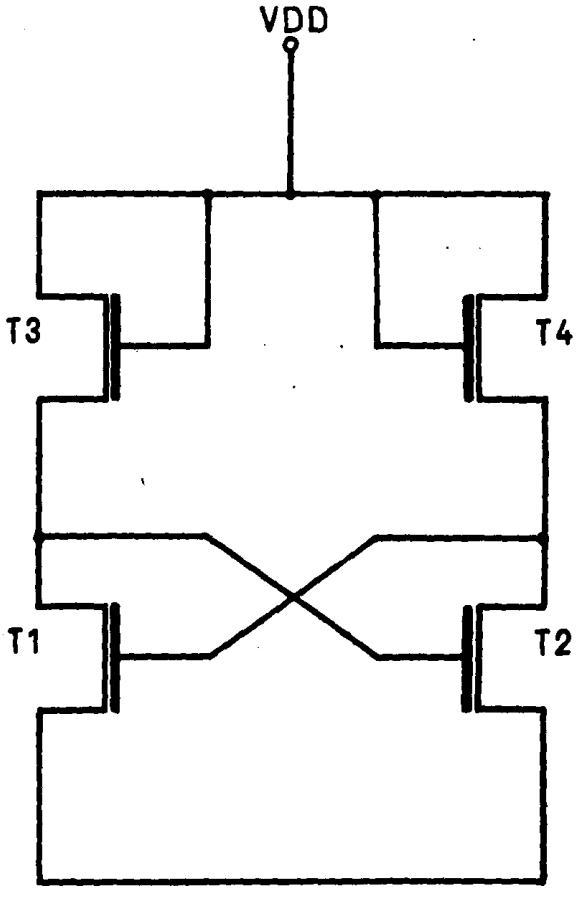
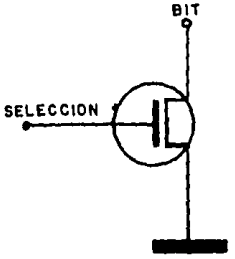


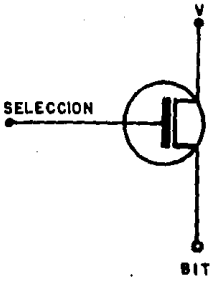
fig.2 Célula RAM-MOS estática

1.2 Memorias Muertas ROM (Read Only Memory). Con este nombre se conoce a las memorias de acceso directo, muertas, de lectura no destructiva. De ellas se conocen los tipos que se tratan a continuación.

ROM Programable por Máscara.- Estas memorias son programadas por medio de máscaras durante el proceso de fabricación.

La programación consiste en controlar el espesor de la capa de óxido de la puerta. Este espesor es el que determina la tensión de umbral del transistor. Por ejemplo, podemos tener una tensión de umbral de -5V mediante una capa fina de óxido y una tensión de -50V con una capa gruesa. Si aplicamos una tensión de -14V a la puerta del transistor, este conducirá o no, según su capa sea fina o gruesa. Existen tres formas de utilización según se muestra en la figura 3.





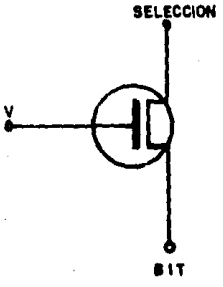


fig. 3: Memoria ROM en 3 Posibilidades.

1.3 PROM (ROM Programable).- Estas memorias pueden ser programadas por el usuario. En las memorias bipolares, la programación consiste en cortocircuitar algunas uniones de aislamiento según se aprecia en la figura 4.

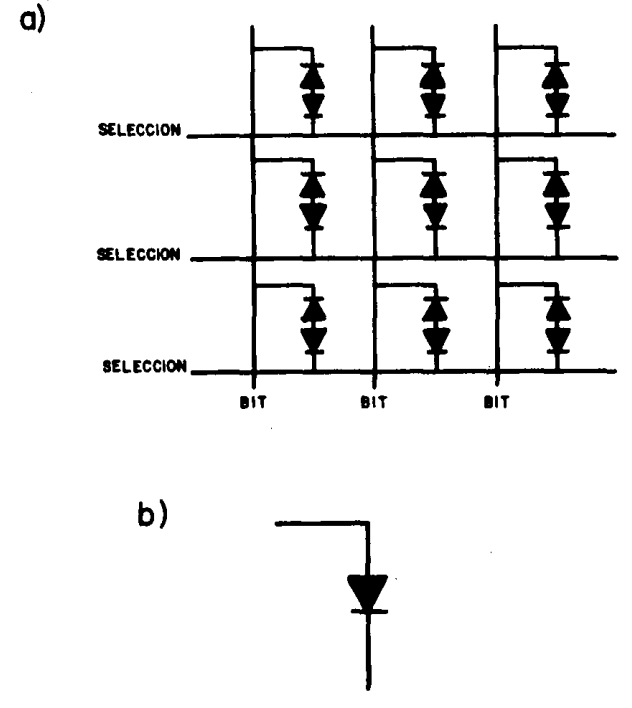


fig. 4: Memoria PROM (Bipolar).

Inicialmente la memoria contiene solo ceros ya que los dos diodos en oposición no permiten la circulación de corriente en ninguna célula.

Si queremos cortocircuitar un bit determinado se coloca a -- tierra la línea de selección correspondiente y por la línea de --- bits se aplican una serie de impulsos de corriente, llegando a -- cortocircuitar por fusión de fusibles conectados a los transistores, figura 5.

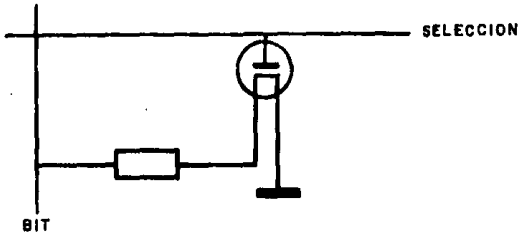


fig. 5: Memoria PROM (MOS).

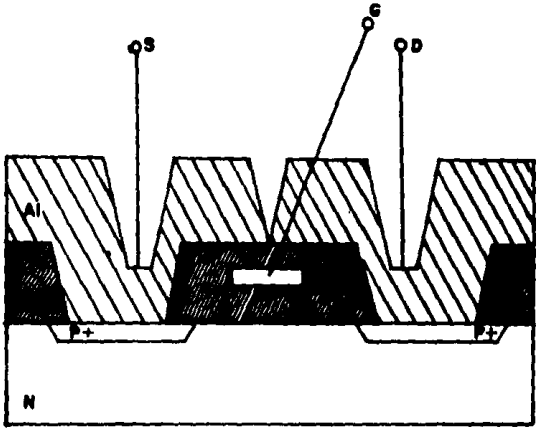
Si el fusible no se destruye queda almacenado un cero, y - si se destruye queda almacenado un uno. Estas memorias no son reprogramables.

1.4 EPROM.- Son memorias PROM que pueden ser borradas o reprogramadas.

Para su fabricación se utilizan transistores MOS con la --- pueta flotante, figura 6.

Para la programación aplicamos una carga sobre la puerta G, mediante una inyección por avalancha.

Mediante exposición a rayos ultravioletas se puede establecer una corriente entre la puerta flotante y el substrato de forma que disminuímos la carga en la mencionada pueta, con lo que obtendremos la desaparición de la información almacenada.



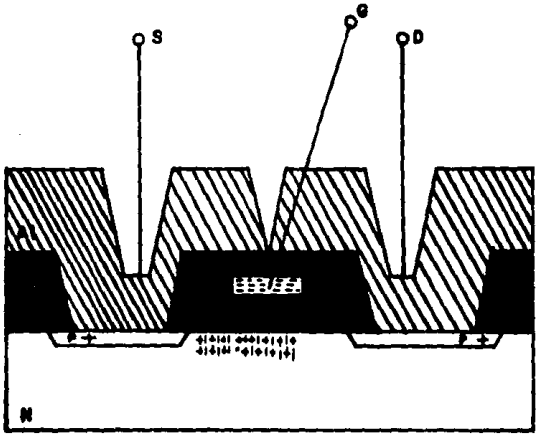


fig. 6 MOS de Puerta Flotante.

1.5 EEPROM (Electrically Erasable PROM).- Son memorias de -- funcionamiento análogo a las EPROM, excepto en que se pueden - borrar eléctricamente en lugar de utilizar rayor ultravioleta.

2.- *MEMORIAS DE ACCESO SECUENCIAL*

Se caraterizan por su tiempo de acceso, cuyo valor depende de la posición que se quiere acceder respecto a un punto de referencia. Son ejemplos típicos los registros de desplazamiento (shift register) y los dispositivos acoplados por carga (CCD).

2.1 Registros de Desplazamiento.- Podemos considerarlos como una cadena lineal de circuitos biestables que por medio de una señal de reloj, cada biestable transfiere su contenido de información al siguiente. El diagrama general es el de la figura 7.

A cada señal de reloj, el contenido de la señal de entrada actúa sobre el biestable A, A sobre B, y así sucesivamente, el contenido de L se pierde.

Si no queremos modificar la información, conectamos a la entrada la propia salida produciendose una recirculación de la in-

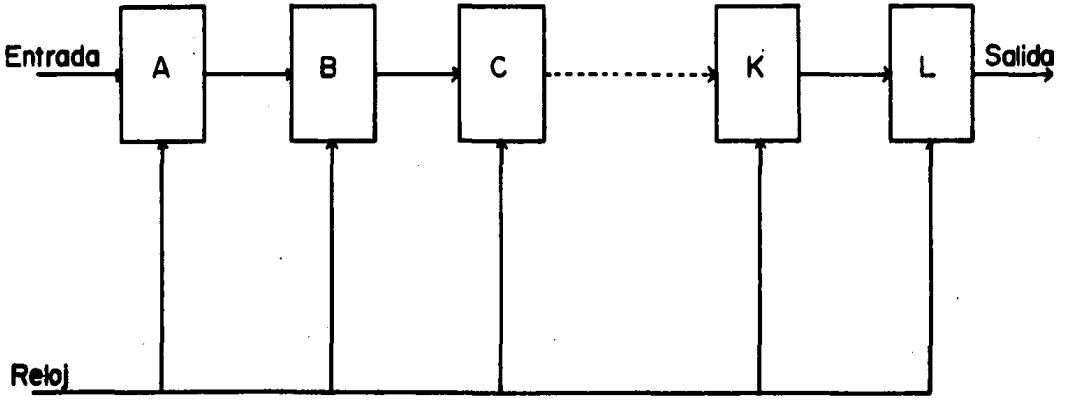


fig. 7 : Registros de Desplazamiento o Corrimiento.

formación y evitándose de esta forma la pérdida de la misma, — contenida en L. En cuanto a su tecnología de fabricación, puede ser al igual que las memorias RAM bipolares, MOS dinámicas, o - CMOS. Los registros de desplazamiento con tecnología MOS re---- quieren a menudo señales de reloj complejas. Las hay, en principio con dos fases de reloj y con cuatro fases, es decir, con — una secuencia de dos a cuatro impulsos aplicados a las corres--- pondientes entradas.

2.2 Dispositivos Acoplados por Carga (CCD).- La tecnología de esta familia está todavía en sus principios, previendo que en un futuro próximo puedan competir con las actuales memorias de masa (discos, cintas magnetofónicas, etc.)

Su funcionamiento básico es idéntico al del registro de desplazamiento, pero realizado tecnológicamente de forma distinta. - En la figura 8 se representa una capa MOS, es decir una capa - con material dopado, una capa de óxido y una metalización selectiva sobre el óxido.

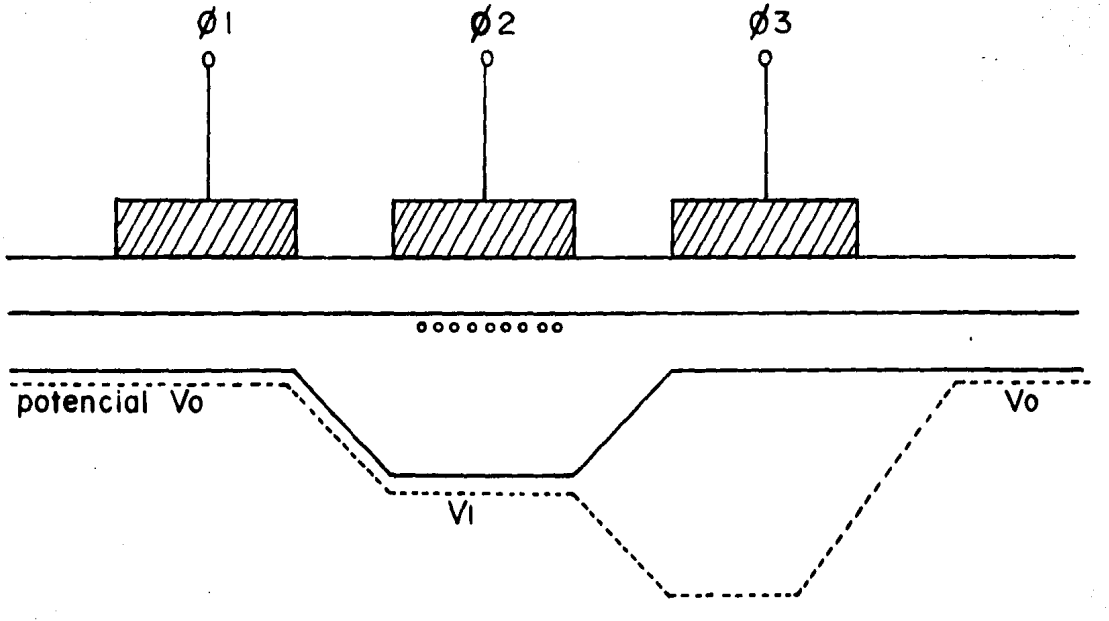


fig. 8: Dispositivo Acoplado por Carga MOS.

La aplicación de un potencial más positivo en ϕ_2, respecto a las otras terminales, crea un pozo de potencial en la superficie del semiconductor frente a ϕ_2, capaz de mantener fija una -- cuarta carga introducida. Si se aplica en este instante un im-- pulso positivo a ϕ_3 de tensión mayor que la de ϕ_2, el potencial aparece ahora como una línea de puntos en la superficie del se-- miconductor. Esto significa que las cargas almacenadas bajo ϕ_2 se desplazan hacia la región situada bajo ϕ_3. De esta forma se origina un desplazamiento de carga análogo al desplazamiento de su información en un registro convencional. La carga se inyecta normalmente en el dispositivo a través de una unión PN de un - transistor MOS convencional; y se recoge con un amplificador de funcionamiento análogo a los amplificadores usados en las memori as dinámicas.

Una de las grandes ventajas de esta reciente tecnología es que no se necesita ningún tipo de difusión selectiva, con lo que se evitan problemas de almacenamiento de máscara, permitiendo - mayor contracción de la información por unidad de superficie.

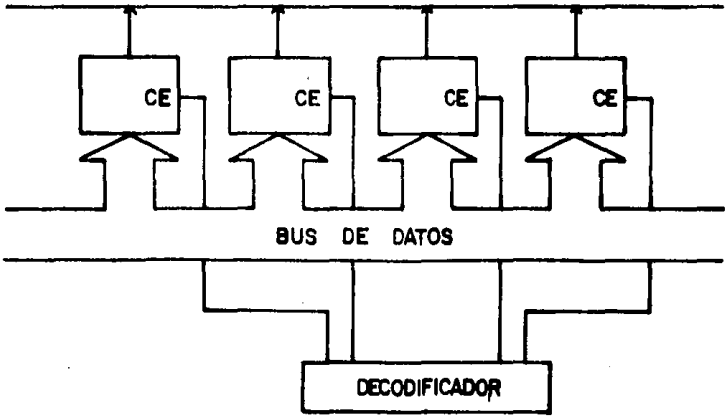
Como ejemplo de las posibilidades de integración se compara la cantidad de información almacenada en memorias actuales estandar.

MOS estática	*1024 bits*	*tipo 2102*
MOS dinámica	*4096 bits*	*tipo 2107*
CCD	*16384 bits*	*tipo 2416*

3.- MEMORIAS ASOCIATIVAS

Estas memorias son poco utilizadas actualmente en los microprocesadores. Realmente son usadas en algunos sistemas de gran volumen, son memorias de acceso aleatorio que responden en la dirección o direcciones en las cuales está contenido el dato que entra. La búsqueda del dato se hace normalmente en paralelo, comparando simultáneamente el dato de entrada con el contenido de la palabra devolviendo un nivel lógico si ambos son iguales.

Permiten también su escritura y lectura por el método normal de una memoria RAM. Existen integradas pero en volumen muy reducido (16 bits).



4.- RELACION DE MEMORIAS Y CUADRO COMPARATIVO

La mayoría de las más recientes RAM, ROM, PROM, ya sea - tecnología bipolar o MOS, han surgido como consecuencia de la aparición de diferentes tipos de microprocesadores LSI. A pesar - de esta circunstancia, muchas de estas memorias tienen gran utilización en circuitos electrónicos diseñados con CI convencionales. Pero es el avance en lo que se refiere a la tecnología LSI en microprocesadores lo que motiva la necesidad de lograr por parte - de los fabricantes de semiconductores, memorias en todas sus gamas con mayores prestaciones, ya sea en cuanto a su capacidad de almacenamiento, al tiempo de acceso más reducido, con un menor costo, etc.

A continuación mostramos un cuadro comaprativo en el cual diferentes fabricantes ofrecen gran variedad de características, - en sus memorias de 4K bytes.

También se incluye, al final del capítulo una tabla en la que se relacionan memorias ROM, PROM y RAM con diferentes técnicas de fabricación.

CUADRO COMPARATIVO DE MEMORIAS RAM DE 4K BYTES.

FABRICANTE	*TIPO*	*TIEMPO DE ACCESO (ns)*	*CICLO DE LEC./ ESC.*
AMS	*7270*	*200ns*	*400ns*
	7280	*200*	*400*
MMI	*2170*	*200*	*400*
	2180	*200*	*400*
INTEL	*2107A*		
	2107B	*200*	*400*
	2107B-4	*270*	*470*
	2107B-6	*350*	*800*
NS	*5270*	*200*	*400*
	5280	*200*	*400*
	5271	*250*	*430*
	5280-5	*300*	*470*
AMD	*9050 CDC*	*300*	*470*
	9050 DDC	*250*	*430*
	9050 EDC	*200*	*400*
	9060 CDC	*300*	*470*
	9060 DDC	*250*	*430*
	9060 EDC	*200*	*400*
TI	*4030*	*300*	*470*
	4050	*300*	*470*
	4051	*300*	*470*
	4060	*300*	*470*
	4060-1	*250*	*430*
SIGNETICS	*2604*	*400*	*470*
	2670	*200*	*200*

5.- ORGANIZACION DE LA MEMORIA DE TRABAJO DEL PROGRAMA MONITOR DE NUESTRO SISTEMA

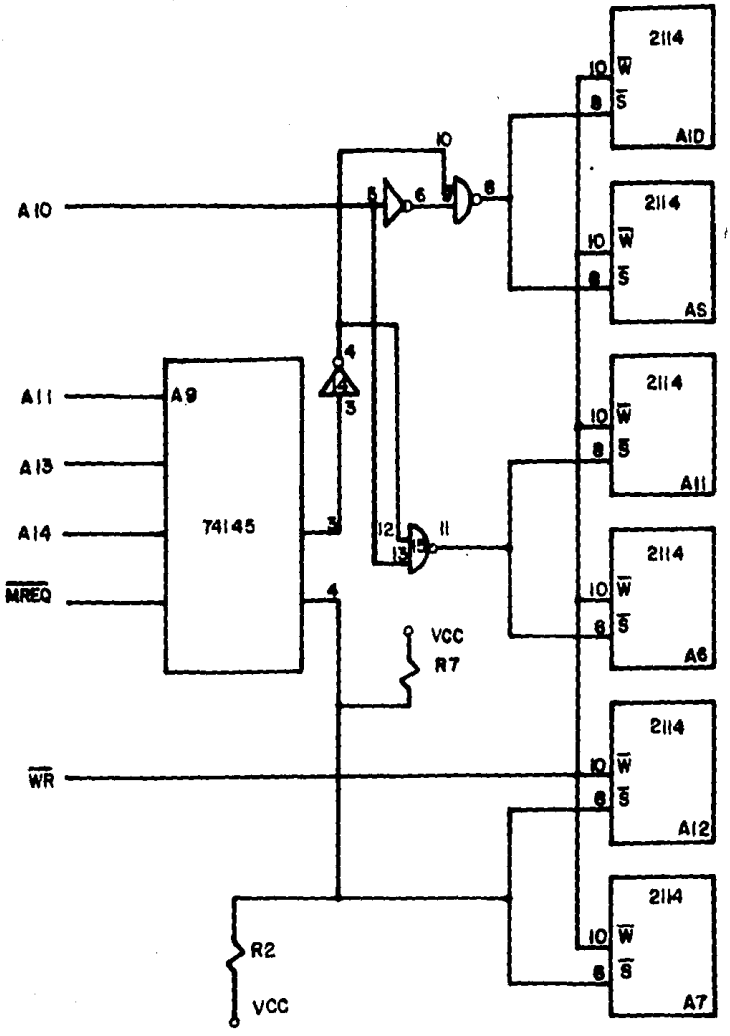
El programa monitor cuando es ejecutado necesita de un área de memoria RAM que está definida dentro del mismo programa monitor y que se organiza de la manera siguiente:

DIRECCION	FUNCION
2BFF 2BD3	Area de trabajo del monitor
2BDO	Brinco del usuario para la SAI
2BCD	Brinco del usuario para RST 10
2BCA	Brinco del usuario para RST 18
2BC7	Brinco del usuario para RST 20
2BC4	Brinco del usuario para RST 28
2BC1	Punto de interrupción para correr programas parcialmente (3 bytes) RST 30.
2BBE	Punto de interrupción para regresar al programa monitor (3 bytes) RST 38.

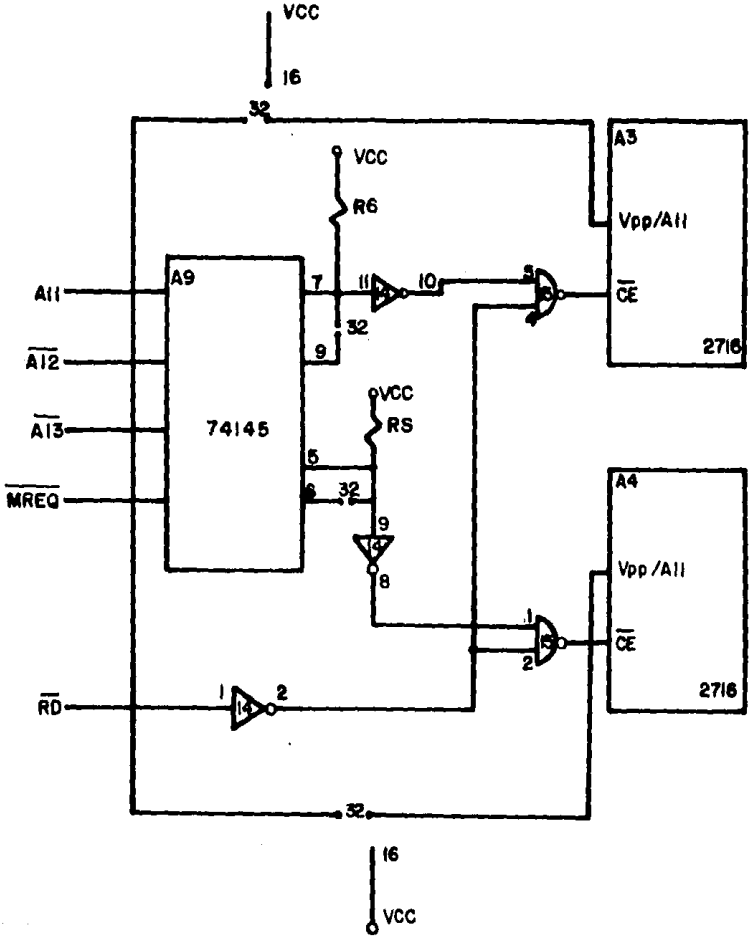
DIRECCION	FUNCION
2BBD	Reservado para manejar los puntos de interrupción para correr programas parcialmente. 5 PI'S máximo.
2BA4	
2BA4 2B60	Area de la PILA o STACK del sistema.
2BF5	Area para los comandos encadenados 16H bytes cada uno, en promedio.

Nuestro sistema cuenta con 3K bytes de memoria RAM de la dirección 2000H en adelante, pero las áreas especificadas arriba tienen que ser respetadas por el programador. El programa Monitor está en un circuito EPROM de 2K bytes que se localiza a - partir de la dirección 0000H.

La configuración de la memoria RAM y EPROM se muestra a continuación, indicando los circuitos integrados que constituyen - cada una de ellas y las características de la interfase que las direcciona.



MEMORIA RAM 3K Bytes.



MEMORIA EPROM 4K Bytes.

Nº BITS	BIPOLAR ROM	BIPOLAR PROM	BIPOLAR RAM	MOS ROM	MOS PROM	MOS RAM Estáticas Dinámicas
64			74189 (TI NS) 74 L 89 74289 (TI NS AMD) 93403 (FSC),3101,(I) [93S5] 9462 (MMI) 85121 (NS) 74S189 (NS, TI, AMD) 5561 (MMI) 82S25 (S)			74C89 (NS, MM)
256	7488 (NS, TI) 93434 (FSC) 5230 (MMI) 8591 (NS) MCM 4002	74188(TI,NS) 5330 (MMI) 8573 (NS)	74S200 (NS), 74S201(TI) 74S301 (TI), 74S206(NS) 2700(AMD) 93410 (FSC) 93411 (FSC) 3101 (I) 3107 (I) IM 5503 IM 5523 5530 5531 (MMI) N 82S10, N 82S11 (S)			CD 4061 (RCA) 1101 (NS) 4256 (NS) 74 C 200 (NS 256.1) 74 C 910 (NS 64.4) MCM 14537 MCM 14552
1024	74187 (TI,NS) 93406 (FSC) 3301 (I) 5200 (MMI) N 82S226 (S) DM 8597 (NS)	MCM 10149(256.4) 8573 (NS 256.4) 93416 (FSC) 74S287 (NS, TI) 74387 (TI,NS)27S10(AMD) 93426 (FSC) HM7610 3601 (I), N82S126 (S) 5300 (MMI)	74S200 (TI, 1024.1) 93415 (FSC, NS) 3106 (I) N 82S11 (S) MCM 10146	3501 (FSC, NS) MCM 14524 5210, 5220 (NS) 8206 (SSS) JM 7621 N 24101 (S) 2802 (TI)		2102 (I, SY, NS, AMD, S) 2101 (I, SY, NS, AMD, S) 2111, 2112 (I, SY, NS,AMD,S) 5200 (NS) 5201 (NS) 1103 (I, MK, SY, NS, S)
2048	74S270 (TI,NS) 74S370 (TI,NS) 93436 93446 (FSC) HM 7620, HM 7621 IM 5604 IM 5624 5205, 5201 (MMI) 82S230, 82S231 (S)	74S470 (TI) 74S471 (TI) HM 7620 HM 7621 3602 3622 (I)		IM 7603 2800 (TI) 5230 (NS) MCM 1117 N 2430 (S)	1702 [(EPROM)](I,NS,AMD) 5203 [(EPROM)](NS) 5202 [(PROM)](NS)	74C200 (NS) 5101 (I) MCM 7001 8130 (SSS) 1801(EMM) HM 7001 (RCA) HM 7040 (RCA) IM 6508 5202 (NS)
4096	5245 (MMI) 5241 (MMI) 8705 (NS) 8701 (NS)	6240 (MMI) 6241 (MMI) 3604 (I) 3624 (I) 6250, 6251 (MMI)		MM 2500, MM 2600 5232 (NS) 5233 N 240 (SSS)	2704 [(EPROM)](I) 5204 [(EPROM)](NS)	2270 7200 (AMS) 9050 9060 (AMS) 2107 (I) 2170 2180 (MMI) 2604 2670 (S) 4030 4050 4057 4060 (TI) 5270, 5271, 5280 (NS) MCM 6604 MCM 6605
8192	85S28 (NS) 85S29 (NS) 5280 (MMI)	5380 (MMI)		MCM 6570 (RCA) 4700 (TI) MCM 6511 MCM 6562 MCM 6531 MCM 6560 8308 (I) EA 8301 (AMS)	2708 (EPROM) (I)	
12288				5217 (NS) 5215 (NS)		
16384	8531 (NS 2 4.8) 8501 (NS 1 4.16)			MCM 6592 (RCA) EA 4800, EA 4900 4800 (TI) MM 2602 8316 (I) MCM 6580 MCM 6592		

SIGLAS	FABRICANTE
IM:	INTERSIL
MCM:	MOTOROLA
I:	INTEL
NS:	NATIONAL SEMICONDUCTOR
TI:	TEXAS INSTRUMENT
MMI:	MONOLITIC MEMORIES
AMD:	ADVANCED MICRO DEVICES
S:	SIGNETICS
NCM:	NITRON

SIGLAS	FABRICANTE
SSS:	SSS - ATEK
FSC:	FAIRCHILD
RCA:	RCA
AMS:	ADVANCED MEMORY SYSTEMS
HM - HP:	HARRIS SEMICONDUCTOR
SY:	SYNERTEK
EMM:	ELECTRONIC MEMORIES
EA:	ELECTRONIC ARRAY
MK:	MOSTEK

Tabla 1. Cuadro General de las Diferentes Memorias M O S.

C A P I T U L O V

IMPLEMENTACION DEL SISTEMA

En este capítulo describiremos el manejo operativo del Microcomputador Educacional, especificando los comandos disponibles y el método para su utilización, la cantidad de Subrutinas de Utilería, su localización en memoria y los registros que modifican. También incluimos un diagrama de bloques del Sistema de Microprocesador y el circuito a detalle.

1.- ORGANIZACION DEL PROGRAMA MONITOR.

El monitor es un programa grabado en memoria ***EPROM*** *que permite explotar los recursos del Microprocesador Z-80. Es gracias a este programa que podemos comunicarnos con la máquina (Microcomputador Educacional).*

El monitor es el software más cercano al hardware esto es, la programación dependiente en su totalidad de los circuitos.

Para desplegar mensajes, el programa monitor contiene información que direcciona al PIO como puerto de salida y para sensar una tecla, el monitor sabe cómo está estructurado el hardware, es decir, sabe qué señales debe enviar como estímulo, qué

esperar como respuesta y por supuesto cómo interpretar los resultados.

Si pensaramos en un sistema de programación que nos permitiera operar la computadora totalmente para aprovecharla al máximo estaríamos hablando de un Sistema Operativo. Los Sistemas Operativos en los computadores de cualquier tamaño permiten manejar absolutamente todos los recursos de la máquina e inclusive lenguajes de alto nivel.

El programa monitor está dividido en tres partes que son:

1.1.- PROGRAMA PRINCIPAL.

1.2.- SUBRUTINAS DE UTILERIA.

1.3.- TABLAS.

1.1.- Programa Principal:

El flujo del programa principal empieza con un ciclo de espera para dar tiempo a que la señal de inicio habilite al puerto PIO, enseguida lo programa como un puerto serie y después fija el apuntador de pila (stack). A continuación se despliega

el mensaje HOLA y el monitor se queda en lazo cerrado esperando que una tecla sea pulsada y mientras esto sucede se está desplegando el mensaje HOLA, que puede ser cambiado si se modifica el contenido del AREA DE MEMORIA DE DESPLEGADO.

Cuando se oprime una tecla, el monitor sale del lazo cerrado y procede a la identificación del comando, comparando los -- caracteres pulsados con una tabla en memoria, si fue comando - solicita los datos requeridos para ejecutarlo y darlo por teminado, quedando en espera de otro comando según sea el caso; si - no fué se envía mensaje de ERROR y pregunta por un nuevo co-- mando.

El puerto PIO programado como puerto serie, puede traba-- jar a diferentes velocidades que son programadas por hardware mediante un puente (ver S1 en diagrama a detalle); estas pue--- den ser 4800, 2400, 1200 o 600 baudios. Cuando se inicializa el Sistema de Microprocesador y en la terminal de video aparece el mensaje C.T.S. significa que ésta no está conectada correctamente o que la velocidad de la terminal es diferente a la del Siste-

ma de Microprocesador Educacional.

A continuación explicaremos los comandos disponibles en -- nuestro Sistema y la forma de utilización. Cuando algún comando requiere 4 dígitos y se introducen más de 4, sólo hará caso a los últimos 4; si se requieren 2, sólo hará caso a los dos últimos; y para ambos casos si sólo se introduce 1, el monitor asume que los datos faltantes son cero. Cuando hay que definir algún rango, como en el caso de DM (despliega memoria), hay dos formas de hacerlo:

a) *Dar la dirección inicial y la dirección final* **DM 2800 280F CR.**

b) *Dar dirección inicial y la cantidad de datos a desplegar en sistema hexadecimal, separados por una letra - "S" que es el indicador para el monitor de que sigue el número de datos.* **DM 2800S1F CR.**

Los comandos de nuestro sistema son:

- *Despliega memoria o registros:*

  DM *dir. inicio dir. final* ***CR*** *(regreso del carro)*

  DM *dir.* ***S*** *cantidad de datos* ***CR***

  DR ***CR***

- *Envía datos al pueto:*

  E *dato #de puerto* ***CR***

- *Localiza un par de datos en memoria:*

  F *dir. inicio dir. final datos* ***CR***

  F *dir. inicio* ***S*** *cantidad de dir. datos* ***CR***

- *Corre un programa:*

  G ***CR***

  G *dir.* ***CR***

- *Corre un programa parcialmente:*

  G*/dir. parcial* ***CR***

  G *dir./dir. parcial* ***CR***

- *Conversión de un número en un código a otros:*

  HA*número* ***CR*** ***HH****número* ***CR***

  HD*número* ***CR*** ***HM****dir.1 dir.2* ***CR***

- *Información de los comandos disponibles:*

  I CR

- *Lee el contenido de un puerto:*

  L*número de puerto* ***CR***

- *Mueve el contenido de un bloque de memoria a otro lugar en memoria:*

  M*dir. inicio dir.final dir.destino* ***CR***

  M*dir. S cantidad de localidades dir.destino* ***CR***

- *Envía nulos por el PIO:*

  N*cantidad de nulos* ***CR***

- *Recibe datos en binario por el PIO y colocalos en memoria:*

  R*dir. destino dir.final* ***CR***

  R*dir. destino* ***S*** *cantidad de localidades* ***CR***

- *Substituye el contenido de memoria o de registros:*

  SM *dir.* ***CR***

  S *registro* ***CR***

- *Transmite por el PIO el contenido de memoria en binario:*

  T*dir. inico dir. final* ***CR***

  T*dir. inicio* ***S*** *cantidad de datos* ***CR***

- *Verifica o compara dos áreas de memoria:*

  Vdir. inicio dir.final dir. destino **CR**

  Vdir. inicio **S** *cantidad de localidades dir. destino* **CR**

La lista anterior muestra los comandos tal y como los despliega el programa monitor y qué información debe tener cada uno de ellos. Para los casos de desplegado de información, éste se puede detener pulsando simultáneamente las teclas de CONTROL y S en la terminal; para continuar con el despliegue basta oprimir cualquier tecla, siempre y cuando no sean ESCAPE o ALT MODE ya que estas abortan cualquier comando que esté realizandose. El monitor tiene una opción para manejar los comandos de una forma encadenada, esto consiste en separar con un símbolo "menor que" (<) los comandos deseados, escribiendo en la terminal el último en ejecutarse en primer lugar, seguido por su antecesor en la ejecución pero separados por el símbolo "menor que" y así sucesivamente hasta introducir el último comando y se termina con un regreso del carro (CR).

1.2.- Subrutinas de utilería:

El programa monitor está formado por 75 subrutinas que -- pueden ser llamadas por el usuario en cualquier momento que se desee; en este inciso nos dedicaremos a las subrutinas del sistema mostrando cada una de ellas, en qué dirección se localizan, qué hacen, qué entradas necesitan, qué salidas, qué registros - modifican y a qué otras subrutinas llaman.

LAS SUBRUTINAS DE UTILERIA:

DIRECCION: inicio de la subrutina.

NOMBRE: nombre con una breve descripción de ella.

ENTRADA: datos requeridos para su funcionamiento.

SALIDA: lugar y datos entregados por ella. Como resultados.

MODIFICA: registros y localidades importantes afectadas o des-- truídas durante su ejecución.

LLAMA: lista de subrutinas que utiliza durante su ejecución.

Subrutinas de Utileria:

```
03AAH
INICIO: ESPERA PARA ESTABILIZACION DEL
MAE Z80 Y PROGRAMA LOS PUERTOS:
- 00 = ENTRADA
- 01 = SALIDA
- 02 = SALIDA
ENTRADA: NINGUNA
SALIDA: PUERTOS PARALELOS (NO OPCIONALES)
MODIFICA: A, HL
LLAMA: ESPER2
```

```
03B2H
DESP21 DESPLEGA EL MENSAJE DE 6 DIGITOS
APUNTADO POR HL
ENTRADA: HL
SALIDA: DESPLEGADO
MODIFICA: A, F
LLAMA: DESP
```

```
03CAH
LOCA: LOCALIZA UN PAR DE DATOS (8 BITS C/U)
CONTIGUOS, EN MEMORIA
HL APUNTA AL PRIMER DATO AL ENCONTRAR EL
PAR Y A=00. SI NO LOS ENCUENTRA A=0FFH
ENTRADA: HL APUNTA AL INICIO DE LA BUSQUEDA
         BC CANTIDAD DE LOC. POR REVISAR
         DE CON EL PAR DE DATOS (16BITS)
SALIDA: A, HL
MODIFICA: A, F, BC, DE, HL
LLAMA: NINGUNA
```

```
03DAH
PEPROM: PROGRAMA EPROM CON EL MODULO
MAE PEPROM, 2716 o 2732

*** REQUIERE DEL CRISTAL DE 2 MHZ ***
*** (Y1), INSTALADO EN EL MAE Z80 ***

ESTA SUBRUTINA ES TOTALMENTE DEPENDIENTE
DE LA CIRCUITERIA, POR LO QUE HAY QUE
CONSULTAR TODO LO REFERENTE AL PROGRAMADOR
DE EPROMS DEL MAE Z80
EL PULSO DE PROGRAMACION ES DE 50 mseg
ENTRADA: HL, DE, BC
SALIDA: EPROM EN EL MAE PEPROM
MODIFICA: A, F
LLAMA: NOPGM,PGM,CPLDE,DIREPR,CE,MSEG50,
       NOCE
```

```
040EH
PGM: QUITA LA SELECCION AL EPROM, ENCIEN-
DE LA FUENTE DE PROGRAMACION Y ESPERA UN
SEGUNDO
ENTRADA: NINGUNA
SALIDA: EPROM DEL MAE PEPROM
MODIFICA: A, F
LLAMA: MSEG50
```

```
0420H
NOPGM: QUITA LA SELECCION AL EPROM, APAGA
LA FUENTE DE PROGRAMACION Y ESPERA UN SEC
ENTRADA: NINGUNA
SALIDA: EPROM DEL MAE PEPROM
MODIFICA: A
LLAMA: MSEG50
```

```
0420H
MSEG50: ESPERA 50 mseg
*** REQUIERE DEL CRISTAL DE 2 MHZ***
*** (Y1), INSTALADO EN EL MAE Z80 ***
ENTRADA: NINGUNA
SALIDA: NINGUNA
MODIFICA: NADA
LLAMA: NINGUNA
```

```
0436H
DIREPR: ENVIA DIRECCIONES AL EPROM
ENTRADA: DE CON DIR. INVERTIDAS
SALIDA: EPROM DEL MAE PEPROM
MODIFICA: A, F
LLAMA: NINGUNA
```

```
0447H
VERDOL: VERIFICA O COMPARA LOS CONTENIDOS
DEL EPROM Y DE LA MEMORIA DEL MAE Z80
SI SON DIFERENTES LOS ENVIA AL DESPLEGADO
PSEUDO ALFANUMERICO, SI SON IGUALES NO HACE
NADA
ESPERA EN HL LA FUENTE (EPROM), EN DE EL
DESTINO (MEMORIA DEL MAE Z80) Y EN BC LA
CANTIDAD DE BYTES POR COMPARAR
ENTRADA: HL, DE, BC
SALIDA: DESPLEGADO PSEUDO ALFANUMERICO
MODIFICA: A, F, REGISTROS ALTERNOS
LLAMA: LEEEPR,DIREPR,NOFF4,CPLDE,ACTDIR
       DESP,LLAC1,PINTA,[illegible]
```

```
0497H
PINTA: DESPLEGA EL VALOR DE H Y L ASI
COMO SU CONTENIDO DURANTE POCO MENOS DE UN
SEGUNDO. SI DURANTE ESTE TIEMPO ES PULSADA
CUALQUIERA TECLA (COMANDOS O DATOS) EL DES-
PLEGADO SE MANTIENE MOSTRANDO EL ACTUAL -
CONTENIDO HASTA QUE OTRA TECLA SE OPRIMA.
ENTRADA: CONTENIDO DE HL
SALIDA: DESPLEGADO
MODIFICA: REGISTROS ALTERNOS
LLAMA: ACTDIR,MDFF4,PINTA1,DESP,LLACI
```

```
04AAH
PINTA1: DESPLEGA LO QUE SE ENCUENTRE EN
EL AREA DE MEMORIA PARA EL DESPLEGAD,
DURANTE APROXIMADAMENTE UN SEGUNDO
ENTRADA: AREA DE MEMORIA DEL DESPLEGADO
SALIDA: DESPLEGADO
MODIFICA: A, F, BC
LLAMA: DESP,LLACI
```

```
04C0H
C4CALL: SOLICITA 3 DATOS DE 16 BITS.
FUENTE, CANTIDAD Y DESTINO,COLOCANDOLOS
EN HL, BC Y DE RESPECTIVAMENTE.
ENTRADA: TECLADO
SALIDA: HL, BC Y DE
MODIFICA: A,F,BC,DE,HL Y REGISTROS ALTERNOS
LLAMA: DESP1,C4CALL,LIMPIA
```

```
04F1H
C4CALL: DESPLEGA, SI LA TECLA PULSADA ES
COMANDO DA ERROR; SI ES DATO LO COLOCA COMO
EL NIBBLE MENOS SIGNIFICATIVO DEL CAMPO DE
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 DES-
PLEGADOS EN LOS BUFFERS 9 Y 8.
ENTRADA: TECLADO
SALIDA: DESPLEGADO Y BUFFERS 9 Y 8
MODIFICA: A,F,C,DE,HL
LLAMA: C3CAL1, ARLD, BFF98, LLACI
```

```
0500H
DESP: DESPLEGA LO QUE SE ENCUENTRA EN EL
AREA DE MEMORIA DEL DESPLEGADO PSEUDO-
ALFANUMERICO (CAMPOS DE DATOS Y DIRECCIO-
NES).
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
```

```
0528H
TECLAS:
SENSA AL TECLADO Y EN CASO DE REGISTRAR
UN DATO, LO CONVIERTE EN SU CORRESPON-
DIENTE 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 BUFF2
MODIFICA: A,F
LLAMA: METEO,ESPER2
```

```
056EH
METEO: LEE EL CONTENIDO DEL PUERTO DIREC-
CIONADO POR EL REGISTRO C, LO COMPLEMENTA
Y DEJA EN A SOLO LOS 5 BYTES MENOS SIGNIFI-
CATIVOS.
ENTRADA: C CON LA DIRECCION DEL PUERTO
SALIDA: A CON LOS 5 BYTES MENOS SIGNIFI-
CATIVOS
MODIFICA: A,F
LLAMA: NINGUNA
```

```
0574H
DESP1: RECIBE EN HL LA DIRECCION DEL MEN-
SAJE A DESPLEGAR SOLAMENTE EN EL CAMPO DE
DE DIRECCIONES DEL DESPLEGADO Y LO DESPLEGA
ENTRADA: HL APUNTANDO A LA TABLA DEL MENSAJE
SALIDA: AREA DE MEMORIA DEL CAMPO DE DIREC-
CIONES Y DESPLEGADO PSEUDO ALFANUMERICO
MODIFICA: A,F,HL
LLAMA: DESP
```

```
;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
;       0584H                                        ;
;       ERROR: DESPLEGA MENSAJE DE ERROR Y REGRESA   ;
;       CON A=00.                                    ;
;       ENTRADA: NINGUNA                             ;
;       SALIDA: DESPLEGADO Y A=00                    ;
;       MODIFICA: A,HL Y REGISTROS ALTERNOS          ;
;       LLAMA: LIMPIA, DESPI                         ;
;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;

;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
;       058FH                                        ;
;       LIMPIA: LIMPIA EL AREA DE MEMORIA DEL DES-   ;
;       PLEGADO                                      ;
;       ENTRADA: NINGUNA                             ;
;       SALIDA: AREA DE MEMORIA PARA DESPLEGADO      ;
;       MODIFICA: REGISTROS ALTERNOS (HL' , DE')     ;
;       LLAMA: NINGUNA                               ;
;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;

;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
;       0591H                                        ;
;       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                               ;
;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;

;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
;       05A6H                                        ;
;       CORRE1: RECORRE A LA IZQUIERDA UNA VEZ LOS   ;
;       DATOS DEL CAMPO DE DATOS DEL DESPLEGADO EN   ;
;       SU AREA DE MEMORIA                           ;
;       ENTRADA: NINGUNA                             ;
;       SALIDA: CAMPO DE DATOS DEL AREA DE MEMORIA   ;
;       DEL DESPLEGADO                               ;
;       MODIFICA: REGISTROS ALTERNOS (B' , HL')      ;
;       LLAMA: NINGUNA                               ;
;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;

;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
;       05B4H                                        ;
;       CONV1: CONVIERTE EL DATO DEL ACUMULADOR DE   ;
;       BINARIO AL CODIGO DE 7 SEGMENTOS (C7S) PARA  ;
;       PODER SER DESPLEGADO. COLOCA EL NIBBLE       ;
;       (MEDIO BYTE) MENOS SIGNIFICATIVO EN EL       ;
;       BUFFER 3 Y EL NIBBLE MAS SIGNIFICATIVO EN    ;
;       EL BUFFER 4.                                 ;
;       ENTRADA: DATO EN A                           ;
;       SALIDA: BUFFERS 3,Y 4                        ;
;       MODIFICA: A,F Y REGISTROS ALTERNOS           ;
;       LLAMA: NINGUNA                               ;
;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;

;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
;       05D4H                                        ;
;       ABIT4: VERIFICA EL VALOR DEL BIT 4 DEL REG.  ;
;       A. SI ES UN UNO DESPLEGA ERROR. SI ES UN     ;
;       CERO REGRESA A QUIEN LE LLAMO.               ;
;       ENTRADA: DATO EN A                           ;
;       SALIDA: NINGUNA                              ;
;       MODIFICA: A,F                                ;
;       LLAMA: NINGUNA                               ;
;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;

;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
;       05DFH                                        ;
;       ESPERA2: RETARDO                             ;
;       ENTRADA: NINGUNA                             ;
;       SALIDA: NINGUNA                              ;
;       MODIFICA: HL                                 ;
;       LLAMA: NINGUNA                               ;
;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;

;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
;       05E2H                                        ;
;       ESPERA: ES UN RETARDO DEFINIDO POR HL        ;
;       ENTRADA: HL                                  ;
;       SALIDA: NINGUNA                              ;
;       MODIFICA: NADA                               ;
;       LLAMA: NINGUNA                               ;
;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;

;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
;       05F2H                                        ;
;       CORRE2: RECORRE A LA IZQUIERDA UNA VEZ EL    ;
;       NIBBLE MENOS SIGNIFICATIVO DEL CAMPO DE      ;
;       DIRECCIONES EN EL AREA DE MEMORIA DEL        ;
;       DESPLEGADO.                                  ;
;       ENTRADA: NINGUNA                             ;
;       SALIDA: BYTE MENOS SIGNIFICATIVO EN EL       ;
;       AREA DE MEMORIA DEL DESPLEGADO.              ;
;       MODIFICA: B',HL'                             ;
;       LLAMA: NINGUNA                               ;
;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;

;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
;       05FBH                                        ;
;       CONV1A: RECIBE EN EL BUFFER 2 EL DATO A SER  ;
;       CONVERTIDO A CODIGO 7 SEGMENTOS (C7S) Y DES- ;
;       PUES 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: CONV1                                 ;
;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
```

```
;*****************************************************
;       0489H                                        ;
;       ACTPUE: CONVIERTE EL CONTENIDO DE C AL C7S   ;
;       Y LO COLOCA EN EL BYTE MENOS SIGNIFICATIVO   ;
;       DEL CAMPO DE DIRECCIONES EN EL AREA DE ME-   ;
;       MORIA DEL DESPLEGADO.                        ;
;       ENTRADA: DATO EN C                           ;
;       SALIDA: DMS DEL CAMPO DE DIRECCIONES EN EL   ;
;       AREA DE MEMORIA DEL DESPLEGADO               ;
;       MODIFICA: A,F Y REGISTROS ALTERNOS           ;
;       LLAMA: CONVI, DESP                           ;
;*****************************************************


;*****************************************************
;       0497H                                        ;
;       DATOS: ESPERA HASTA QUE 3 TECLAS SON PULSA-  ;
;       DAS (MINIMO). LAS DOS PRIMERAS DEBEN SER     ;
;       DATOS Y LA TERCERA "OTRO". SI "CO" O "CON2"  ;
;       SON OPRIMIDAS AQUI, SE DESPLEGA ERROR, PERO  ;
;       SI ES "CON1" DESPLEGARA HOLA Y EN AMBOS      ;
;       CASOS (CO, CON2 O CON1) BRINCA A LA RUTINA   ;
;       DE RECONOCIMIENTO DE COMANDOS.               ;
;       ENTRADA: TECLADO                             ;
;       SALIDA: DESPLEGADO (MODIFICANDO CAMPO DE     ;
;       DATOS) Y BUFFER 2                            ;
;       MODIFICA: A,F,BC,D Y REGISTROS ALTERNOS      ;
;       LLAMA: LLAC1, AL002, LLAC, CORRE1, CONVIA,   ;
;       LLAC2                                        ;
;*****************************************************


;*****************************************************
;       04A4H                                        ;
;       AL002: 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                                ;
;*****************************************************


;*****************************************************
;       04F3H                                        ;
;       LLAC1: 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.             ;
;       ENTRADA: AREA DE MEMORIA PARA DESPLEGADO Y   ;
;       EL TECLADO                                   ;
;       SALIDA: DESPLEGADO Y BUFFERS 1 Y 2           ;
;       MODIFICA: A,F                                ;
;       LLAMA: DESP, ADIT4                           ;
;*****************************************************


;*****************************************************
;       04B1H                                        ;
;       LLAC1: DESPLEGA LO QUE ENCUENTRE EN EL AREA  ;
;       DE MEMORIA DEL DESPLEGADO, ESPERA 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: DESP                                  ;
;*****************************************************


;*****************************************************
;       04BEH                                        ;
;       LLAC2: RECIBE EN EL BUFFER 2 EL NIBBLE MENOS;
;       SIGNIFICATIVO QUE JUNTO CON EL MENOR NIBBLE  ;
;       DE C FORMABAN UN BYTE QUEDANDO EN C.         ;
;       SI EL VALOR DEL BUFFER 2 REPRESENTA A CO     ;
;       (04H) O CON2 (0CH) DESPLEGA ERROR, PERO SI   ;
;       ES CON1 (08H) DESPLEGA HOLA Y BRINCA A LA    ;
;       RUTINA DE RECONOCIMIENTO DE COMANDOS.        ;
;       ENTRADA: BUFFER 2                            ;
;       SALIDA: C                                    ;
;       MODIFICA: A,F,BC,D                           ;
;       LLAMA: NINGUNA                               ;
;*****************************************************


;*****************************************************
;       04ACH                                        ;
;       DAT: ESPERA A QUE UNA TECLA SEA PULSADA, SI  ;
;       ES DATO LO CONVIERTE A C7S REJUNTOLO EN EL    ;
;       NIBBLE MENOS SIGNIFICATIVO DEL CAMPO DE DI-  ;
;       RECCIONES EN EL AREA DE MEMORIA DEL DESPLE-  ;
;       GADO Y REGRESA, PERO SI ES COMANDO DESPLIEGA ;
;       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, AL002                           ;
;*****************************************************


;*****************************************************
;       04B6H                                        ;
;       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   ;
;       DE MEMORIA DEL DESPLEGADO                    ;
;       MODIFICA: A,C,DE                             ;
;       LLAMA: ACTPUE, CONVI, DESP                   ;
;*****************************************************
```

```
06CCH
ARLD: TOMA LA ULTIMA TECLA PULSADA Y
-SI ES DATO, ROTA A LA IZQUIERDA LOS 4
NIBBLES EN LAS 2 LOCALIDADES DE MEMORIA CON-
TIGUAS, DE LAS CUALES LA MAS ALTA ES DIREC-
CIONADA POR HL, COLOCA ESA ULTIMA TECLA
COMO EL NIBBLE MENOS SIGNIFICATIVO DE (HL-1)
Y REGRESA A QUIEN LE 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 RECONOCIMIEN-
TO 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: ALDD2
```

```
06E0H
HBFF4: CONVIERTE EL DATO DE A AL C7S Y LO
COLOCA EN EL CAMPO DE DATOS DEL AREA DE ME-
MORIA PARA EL DESPLEGADO.
ENTRADA: A
SALIDA: CAMPO DE DATOS EN EL AREA DE MEMO-
RIA DEL DESPLEGADO
MODIFICA: A,F Y REGISTROS ALTERNOS
LLAMA: CONVI
```

```
06F0H
ACTREG: RECIBE AL REGISTRO DE APUNTANDO 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
```

```
0705H
BFF90: COLOCA LOS BUFFERS 9 Y 8 EN CEROS.
ENTRADA: NINGUNA
SALIDA: BUFFERS 9 Y 8
MODIFICA: A,F,HL
LLAMA: NINGUNA
```

```
070DH
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 8
SALIDA: CAMPO DE DIRECCIONES EN EL AREA DE
MEMORIA DEL DESPLEGADO Y HL APUNTANDO AL
BUFFER 8
MODIFICA: A,C,DE,HL
LLAMA: ACTDIR
```

```
0717H
C3CAL1: DESPLEGA LO QUE ESTA EN LA MEM,
SENSA LAS TECLAS, EN CASO DE SER COMANDO DA
ERROR Y SI ES DATO COLOCA LOS BUFFERS 9 Y 8
EN CEROS.
ENTRADA: HL APUNTANDO AL MENSAJE
SALIDA: DESPLEGADO Y BUFFERS 9 Y 8
MODIFICA: A,F,HL
LLAMA: DESP,LLAC,BFF90
```

```
0723H
CE: SELECCIONA AL EPROM
ENTRADA: NINGUNA
SALIDA: EPROM DEL NKE PEPROM
MODIFICA: A
LLAMA: NINGUNA
```

```
072AH
NOCE: QUITA LA SELECCION AL EPROM
ENTRADA: NINGUNA
SALIDA: EPROM DEL NKE PEPROM
MODIFICA: A
LLAMA: NINGUNA
```

```
;       07H00H                                                  ;
;       CPLDE: COMPLEMENTA EL VALOR DEL REG. DE                 ;
;       ENTRADA: DE                                             ;
;       SALIDA: DE                                              ;
;       MODIFICA: A, F, DE                                      ;
;       LLAMA: NINGUNA                                          ;
```

```
;       07J1H                                                   ;
;       LEPROM: TRAE LOS DATOS DEL EPROM A LA                   ;
;       MEMORIA DEL VAX Z80                                     ;
;       HL TIENE LA FUENTE (EPROM), DE TIENE EL                 ;
;       DESTINO Y BC LA CANTIDAD                                ;
;       ENTRADA: HL, DE, BC                                     ;
;       SALIDA: MEMORIA DEL VAX Z80                             ;
;       MODIFICA: A, F                                          ;
;       LLAMA: LEEEPR,DIREPR,NOPCH                              ;
```

```
;       07K0H                                                   ;
;       LEEEPR: PROGRAMA LOS PUERTOS OPCIONALES                 ;
;       PARA LEER EL EPROM, LO SELECCIONA, COM-                 ;
;       PLEMENTA AL REG. DE Y ESPERA UN SEGUNDO                 ;
;       ENTRADA: NINGUNA                                        ;
;       SALIDA: EPROM DEL VAX PEPROM                            ;
;       MODIFICA: A, DE, HL                                     ;
;       LLAMA: NOPCH,CE,CPLDE                                   ;
```

```
;       07L7H                                                   ;
;       LLAC4: DESPLIEGA LO QUE SE ENCUENTRE EN EL              ;
;       AREA DE MEMORIA DEL DESPLEGADO Y COMPARA                ;
;       CONTRA 00 EL VALOR DEL BUFFER; PARA SABER               ;
;       SI SE TECLEO ALGO                                       ;
;       ENTRADA: AREA DE MEM. DEL DESPL. Y TECLADO              ;
;       SALIDA: A, F, DESPLEGADO                                ;
;       MODIFICA: A, F                                          ;
;       LLAMA: DESP                                             ;
```

```
;       08B0H                                                   ;
;       MRI: Coloca el bronco del RSI 3BH para el usuario y     ;
;       trae el dato de la terminal                             ;
;       ENTRADA: USART                                          ;
;       SALIDA: A                                               ;
;       MODIFICA: A,F                                           ;
;       LLAMA: Ninguna                                          ;
```

```
;       08D0H                                                   ;
;       MRI: Pone dato de la terminal si esta listo             ;
;       ENTRADA: USART                                          ;
;       SALIDA: A                                               ;
;       MODIFICA: A,F                                           ;
;       LLAMA: Ninguna                                          ;
```

```
;       08I5H                                                   ;
;       TRTE: Toma el caracter de la entrada, pero si es NULO   ;
;       regresa por otro. Quita el bit mas significativo        ;
;       ENTRADA: USART                                          ;
;       SALIDA: A                                               ;
;       MODIFICA: A,F                                           ;
```

```
;       08I8H                                                   ;
;       PRTE: Pinta byte o despliega caracter en la terminal    ;
;       ENTRADA: A                                              ;
;       SALIDA: Puerto de datos del USART                       ;
;       MODIFICA: Nada                                          ;
;       LLAMA: Ninguna                                          ;
```

```
;       08ZEH                                                   ;
;       USART: Inicializa el USART (825181) en el modo de los   ;
;       cosas: 1200 bandios, 2 bits de parada, 8 bits de datos  ;
;       ENTRADA: Ninguna                                        ;
;       SALIDA: Puerto de control del USART (puerto 09)         ;
;       MODIFICA: A,F                                           ;
;       LLAMA: Ninguna                                          ;
```

```
;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
;       0842H                                                     ;
;       D2BHL: Despliega los 2 bytes en (HL) y (HL-1), decremen-  ;
;       ta en 2 el HL, verifica si se teclea "ESCAPE" o "^S"      ;
;       e inserta un "ESPACIO" al final                           ;
;       ENTRADA: HL, USART                                        ;
;       SALIDA: USART                                             ;
;       MODIFICA: A, HL                                           ;
;       LLAMA: PHM, MRX1, PBYTE, SACHEN, CMND1                    ;
;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;

;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
;       084AH                                                     ;
;       ESPACIO: Envia un espacio y verifica si es tecleado       ;
;       "ESCAPE" o "^S"                                           ;
;       ENTRADA: USART                                            ;
;       SALIDA: USART                                             ;
;       MODIFICA: A                                               ;
;       LLAMA: MRX1, PBYTE, SACHEN, CMND1                         ;
;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;

;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
;       084CH                                                     ;
;       PCAR: Despliega el caracter del reg. A y si es:           ;
;               "ESCAPE" .- deja de imprimir y regresa            ;
;                       al monitor                                ;
;               "^S" .- deja de imprimir y espera hasta           ;
;                       que el RX del USART reciba cual-          ;
;                       quier caracter para continuar             ;
;               (" .- deja de imprimir, envia un "8" a            ;
;                       la terminal y entra al monitor            ;
;                       por otro comando, para encadenarlo        ;
;       ENTRADA: A, USART                                         ;
;       SALIDA: USART                                             ;
;       MODIFICA: Nada                                            ;
;       LLAMA: MRX1, PBYTE, SACHEN, CMND1                         ;
;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;

;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
;       [illegible]H                                              ;
;       TCAR: Trae caracter de la terminal y despliegalo          ;
;       como un "eco", verifica si es "ESCAPE", "^S" o "("        ;
;       ENTRADA: USART                                            ;
;       SALIDA: A, USART                                          ;
;       MODIFICA: A,F                                             ;
;       LLAMA: TBYTE, PCAR                                        ;
;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
```

```
;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
;       0092H                                                 ;
;       BINDEC: Convierte el valor del reg. A de binario a    ;
;       decimal, dejando centenas, decenas y unidades en los  ;
;       registros D,E y B respectivamente                     ;
;       ENTRADA: A                                            ;
;       SALIDA: D, E, B                                       ;
;       MODIFICA: A, F, B, DE                                 ;
;       LLAMA: Ninguna                                        ;
;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;

;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
;       00AAH                                                 ;
;       CORREI: Recorre a la izquierda el contenido de A,     ;
;       cuatro veces                                          ;
;       ENTRADA: A                                            ;
;       SALIDA: A                                             ;
;       MODIFICA: A, F                                        ;
;       LLAMA: Ninguna                                        ;
;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;

;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
;       00B3H                                                 ;
;       CORRED: Recorre a la derecha el contenido del A,      ;
;       cuatro veces                                          ;
;       ENTRADA: A                                            ;
;       SALIDA: A                                             ;
;       MODIFICA: A, F                                        ;
;       LLAMA: Ninguna                                        ;
;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;

;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
;       00BCH                                                 ;
;       CRLF: Envia un "CR" a la terminal y verifica si es    ;
;       tecleado "ESCAPE" o "^S"                              ;
;       ENTRADA: USART                                        ;
;       SALIDA: USART                                         ;
;       MODIFICA: A                                           ;
;       LLAMA: MRXL, PBYTE, SACMEN, CAMD1                     ;
;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
```

```
;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
;       0BC0H                                                     ;
;       L2NCR0: Carga HL con la dir. fuente, BC y DE con el       ;
;       incremento, si el separador no es "CR" senala error       ;
;       y regresa al monitor, de otra forma envia un "CR"         ;
;       ENTRADA: USART                                            ;
;       SALIDA: A, HL, USART                                      ;
;       MODIFICA: A, F, DE, HL                                    ;
;       LLAMA: LD2N, NOESP1                                       ;
;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;

;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
;       0BC4H                                                     ;
;       ESPCR: Brinca los espacios iniciales y si el separador    ;
;       no es "CR" da error y brinca al monitor                   ;
;       ENTRADA: USART                                            ;
;       SALIDA: A                                                 ;
;       MODIFICA: A, F, DE, HL                                    ;
;       LLAMA: NOESP1                                             ;
;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;

;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
;       0BCCH                                                     ;
;       DIRPC: Envia un "CR" a la terminal y despliega el numero  ;
;       en HL, seguido por dos puntos y verifica si teclean       ;
;       "ESCAPE" o "^S"                                           ;
;       ENTRADA: HL                                               ;
;       SALIDA: USART                                             ;
;       MODIFICA: A, F                                            ;
;       LLAMA: CRLF, PNHL, NRX1, PBYTE, SACMEN, CANDI             ;
;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;

;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
;       0BD6H                                                     ;
;       LD2N: Carga 2 numeros. Carga DE con la dir. inicial,      ;
;       N1, BC y HL con el incremento N2-N1+1 , o con N2 si       ;
;       se uso el operador "S", A regresa con el ultimo           ;
;       separador                                                 ;
;       ENTRADA: USART                                            ;
;       SALIDA: A, BC, DE, HL                                     ;
;       MODIFICA: A, F, BC, DE, HL                                ;
;       LLAMA: GNHL, NOESP1, GNHL0                                ;
;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;

;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
;       0BF0H                                                     ;
;       L3NCR: Carga 3 operandos. En HL la fuente, en HL el       ;
;       incremento y en DE el tercer operando                     ;
;       ENTRADA: USART                                            ;
;       SALIDA: A, BC, DE, HL                                     ;
;       MODIFICA: A, BC, DE, HL                                   ;
;       LLAMA: LD2N, GNHL, NOESP1                                 ;
;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
```

```
;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
;       08F3H                                                  ;
;       LINCR: Recibe de la terminal un "ESPACIO" o un numero. ;
;       Carga en HL un numero nuevo e intercambia DE y HL.     ;
;       Si el separador no es "CR" va al monitor con ERROR     ;
;       Finaliza poniendo el cursor en la nueva linea          ;
;       ENTRADA: USART                                         ;
;       SALIDA: A, HL, USART                                   ;
;       MODIFICA: A, F, HL                                     ;
;       LLAMA: GNHL, NOESP1                                    ;
;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;

;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
;       08FBH                                                  ;
;       GNHLO: Limpia HL. Si se recibe un caracter hexadecimal ;
;       lo inserta en HL. De otra forma ignora los primeros    ;
;       ESPACIOS. El primer caracter debe ser hexadecimal.     ;
;       Continua insertando hasta que se recibe un caracter    ;
;       no hexadecimal y regresa con el ultimo caracter en A   ;
;       ENTRADA: USART                                         ;
;       SALIDA: A, HL                                          ;
;       MODIFICA: AF, HL                                       ;
;       LLAMA: NOESP1, HEXSH, TCAR                             ;
;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;

;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
;       0911H                                                  ;
;       HEXSH: Si A contiene un caracter hexadecimal, pone en  ;
;       HL el equivalente binario. Si no es hexadecimal regresa ;
;       con la bandera CY=1. Guarda el caracter original en B  ;
;       ENTRADA: A                                             ;
;       SALIDA: B, HL, CY                                      ;
;       MODIFICA: A, F, B, HL                                  ;
;       LLAMA: Ninguna                                         ;
;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;

;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
;       0928H                                                  ;
;       NOESP: Regresa con un caracter diferente a "ESPACIO"   ;
;       en A. Si se recibe un "NULO" o un "ESPACIO" espera     ;
;       hasta que entre otro caracter que no sea uno de los 2  ;
;       ENTRADA: USART                                         ;
;       SALIDA: A                                              ;
;       MODIFICA: A,F                                          ;
;       LLAMA: TCAR                                            ;
;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;

;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
;       0934H                                                  ;
;       PGNM: Envia "ESPACIO" seguido por el contenido de la   ;
;       localidad apuntada por HL. Verifica si se teclea "^S"  ;
;       o "ESCAPE"                                             ;
;       ENTRADA: HL, USART                                     ;
;       SALIDA: USART                                          ;
;       MODIFICA: A, F                                         ;
;       LLAMA: ESPACIO, P1HEX, PCAR                            ;
;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
```

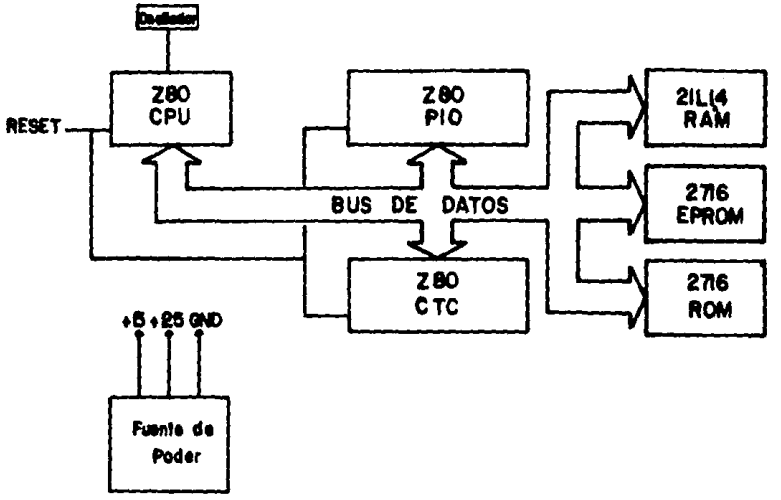
```
;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
;       093AH                                                ;
;       PSNHL: Despliega el numero en HL                     ;
;       Verifica si se teclea "ESCAPE", "^S"                 ;
;       ENTRADA: HL, USART                                   ;
;       SALIDA: USART                                        ;
;       MODIFICA: A, F                                       ;
;       LLAMA: ESPACIO, P2HEX, P1HEX, PCAR                   ;
;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;

;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
;       0942H                                                ;
;       P2HEX: Convierte al A a 2 caracteres ASCII y verifica ;
;       si se teclea "ESCAPE", "^S"                          ;
;       ENTRADA: A, USART                                    ;
;       SALIDA: USART                                        ;
;       MODIFICA: A, F                                       ;
;       LLAMA: P1HEX, PCAR                                   ;
;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;

;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
;       095AH                                                ;
;       SACMEN: Saca mensaje. HL apunta al inicio del mensaje. ;
;       El mensaje finaliza despues de desplegar un caracter ;
;       cuyo bit de paridad sea 1. HL apunta a la siguiente  ;
;       localidad del fin de mensaje.                        ;
;       Verifica si se teclea "ESCAPE", "^S" o "(", o si forman ;
;       parte del mensaje, en cuyo caso se comportara el des- ;
;       plegado como se indica en la subrutina PCAR          ;
;       ENTRADA: HL, USART                                   ;
;       SALIDA: USART                                        ;
;       MODIFICA: HL                                         ;
;       LLAMA: PCAR                                          ;
;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;

;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
;       0965H                                                ;
;       MVE: Mueve de una localidad a otra. HL con la fuente. ;
;       DE con el destino y BC con la cantidad por mover     ;
;       ENTRADA: BC, DE, HL, MEMORIA                         ;
;       SALIDA: MEMORIA                                      ;
;       MODIFICA: Nada                                       ;
;       LLAMA: Ninguna                                       ;
;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;

;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
;       096EH                                                ;
;       CSUBV: Entrada con HL apuntando a memoria y B contenien- ;
;       do la bandera de 1 byte o la de 2 bytes. Pinta: ESPACIO, ;
;       contenido de (HL) y (HL-1) para registros de 2 bytes. ;
;       trae el valor de substitucion y lo carga. Regresa con ;
;       la bandera de CERO (Z) en 1 si el separador es un "CR". ;
;       Verifica si se teclea "ESCAPE", "^S"                 ;
;       ENTRADA: B, HL, USART                                ;
;       SALIDA: USART                                        ;
;       MODIFICA: A, F, HL                                   ;
;       LLAMA: PNHL, PCAR, TCAR, CNHL, ESPACIO               ;
;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
```

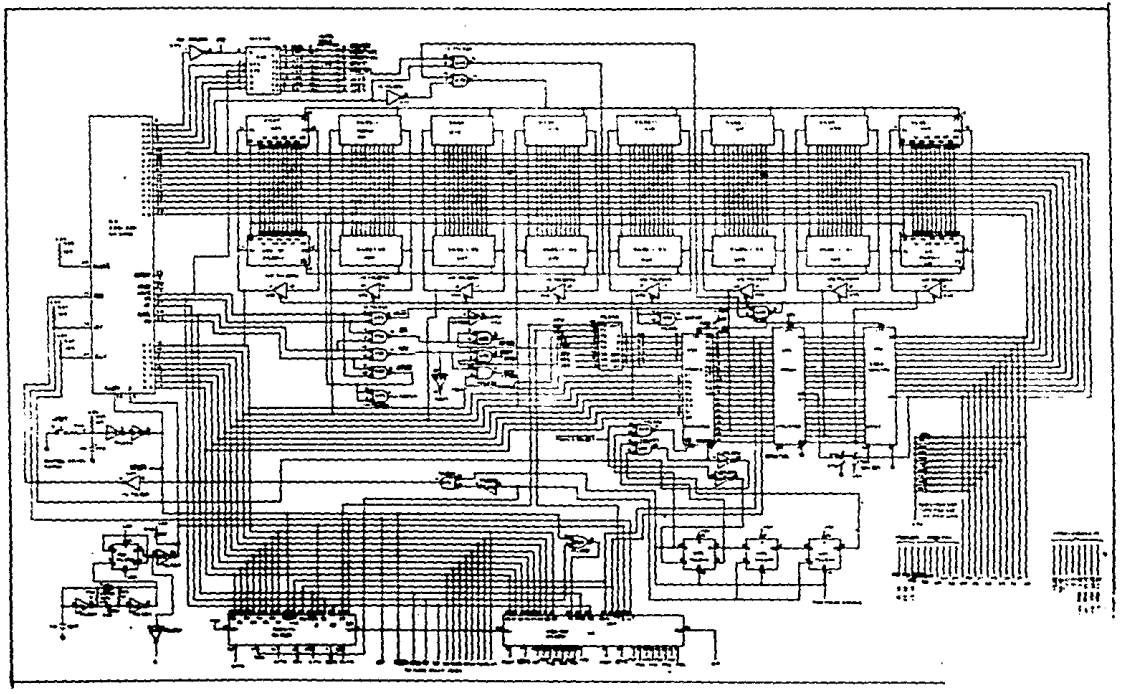
1.3.- Tablas:

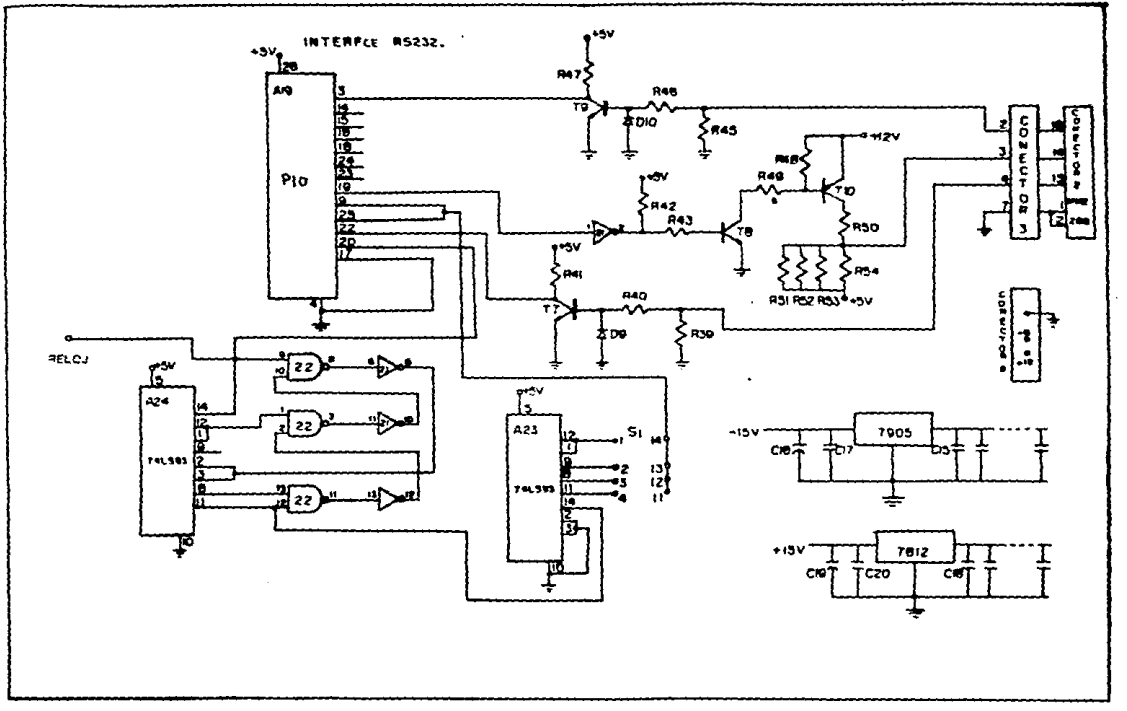
Las tablas en el monitor son utilizadas para obtener mensajes, conversiones de códigos y direcciones de referencias.

2.- DIAGRAMACION DE BLOQUES DEL MICROCOMPUTADOR EDUCACIONAL.

El siguiente diagrama es la representación del hardware -- principal de nuestro sistema, indicando los componentes principales, el flujo de información, el tipo de memoria y su direccionamiento; todo a nivel de bloques.






INTERFCE RS232.
+5V
A19
P10
R47
T9
R46
D10
R45
+12V
R48
R49
T10
R50
R42
R43
T8
R54
R51 R52 R53
R41
T7
R40
D9
R39
RELOJ
A24
74LS393
22
A23
S1
7905
C16
C17
C15
7812
C19
C20
C18
CONECTOR

C O N C L U S I O N E S

C O N C L U S I O N E S

Debido al gran avance tecnológico en el mundo y a que éste se basa en la electrónica digital; sentimos la necesidad cada vez mayor de estar al día en este campo; para esto, proponemos un Microcomputador Educativo.

Este sistema ayudará a aquéllas personas con necesidades en el área digital a capacitarse y perfeccionarse en el uso de - Microprocesadores y además a presentar soluciones viables a problemas industriales en difernetes áreas.

El Microprocesador Educativo permite al usuario profundizar en el conocimiento y manejo de sistemas digitales, a un bajo costo y en un tiempo relativamente corto. Este sistema de Microprocesador, puede crecer en la medida en que el usuario lo requiera, teniendo como limitante la cretividad del mismo.

Después de concebir la idea de desarrollar un sistema de Microprocesador, nos enfrentamos algunas veces a problemas técnicos y otras a problemas con la adquisición de información bibliográfica, siendo estos últimos los menos importantes. Dentro de los problemas técnicos podemos mencionar: el ajuste y adap-

tación del diseño original (teórico), a componentes del mercado - Nacional; la definición de funciones específicas del sistema (número de comandos, formato de los mismos, dispositivos de almacenamiento, etc.); las limitaciones y hacer el microcomputador verdaderamente didáctico.

De las características planteadas al principio del trabajo, algunas (en su mayoría) pudieron cumplirse tales como:

Aspecto físico: *El microcomputador es en una sola tableta; de - fácil expansión; con difernetes velocidades de transmisión - recepción; con interface de comunicación estandarizada RS - 232.*

Aspecto programación: *La estructura del sistema operativo está diseñada de tal forma que permite al educando o usuario implementar o modificar, si lo requiere, cualquier comando simplemente cambiando o desarrollando el software necesario.*

Una característica planteada al inicio, que preferimos cambiar fué la implementación de la circuitería para la interface -- con una grabadora ya que esto nos acarreaba un incremento considerable en el tamaño de la tableta; pero sin embargo pudimos

añadir un pequeño programador de memoria EPROM (2K bytes), ya que esto significó tener un sistema versátil en su aplicación.

Consideramos que el punto más importante de este trabajo, es haber logrado un sistema que además de ser educativo puede cambiar su finalidad y convertirse en una herramienta que soluciona problemas industriales, simplemente cambiando un circuito integrado (EPROM con el Monitor) por otro con el Monitor nuevo; y es esta la justificación del diseño del sistema con un puerto - paralelo convertido a serie.

BIBLIOGRAFIA

BIBLIOGRAFIA

AN INTRODUCTION TO MICROCOMPUTERS. VOLUMEN I BASIC CON-- CEPTS.
Adam Osborne and Associates, Incorporated. 1976.

AN INTRODUCTION TO MICROCOMPUTERS. VOLUMEN II SOME REAL - PRODUCTS.
Adam Osborne and Associates, Incorporated. 1976.

THE MICROPROCESSOR HANDBOOK.
The Texas Instruments Learning Center. 1975,

INTRODUCTION TO MICROPROCESSOR.
Charles M. Gilmore. 1981.

MICROPROCESSORS/MICROCOMPUTERS: AN INTRODUCTION.
Donal D. Givone; Robert P. Roesser. 1980.

PROGRAMACION DEL MICROPROCESADOR Z-80.
Elizabeth A. Nichols; Joseph C. Nichols; Peter R. Rony. 1981.

HOW TO PROGRAM MICROCOMPUTERS.
William Barden, Jr. 1977.

Z80 PROGRAMING FOR LOGIC DESIGN.
Adam Osborne; Jerry Kane; Russell Rector; Susana Jacobson. 1978.

MICROCOMPUTERS/MICROPROCESSORS: HARDWARE, SOFTWARE AND A-- PLICATIONS.
John L. Hilburn; Paul M. Julich. 1976.

METODOS DE SERVICIO PARA EL MANEJO DE INFORMACION EN SERIE EN MICROCOMPUTADORES.
Tesis Profesional. ULSA.
J.G. González; R. C. Ruiz Mestas. 1980.

Z80-CPU, Z80A-CPU. TECHNICAL MANUAL.
Ziolog Incorporated. 1977.

Z80-PIO. TECHNICAL MANUAL.
Zilog Incorporated. 1977.

PRODUCT SPECIFICATION CTC.
Zilog Incorporated. 1979.

Z80-CPU, Z80A-CPU. TECHNICAL MANUAL.
Ziolog Incorporated. 1977.

Z80-PIO. TECHNICAL MANUAL.
Zilog Incorporated. 1977.

PRODUC SPECIFICATION CTC.
Zilog Incorporated. 1979.

A P E N D I C E

Mnemonic	Symbolic Operation	Flags C	Z	P/V	S	N	H	OP-Code 76	543	210	No. of Bytes	No. of M Cycles	No. of T Cycles	Comments	
LD r, r'	r ← r'	•	•	•	•	•	•	01	r	r'	1	1	4	r, r'	Reg.
LD r, n	r ← n	•	•	•	•	•	•	00	r	110	2	2	7	000	B
								←	n	→				001	C
LD r, (HL)	r ← (HL)	•	•	•	•	•	•	01	r	110	1	2	7	010	D
LD r, (IX+d)	r ← (IX+d)	•	•	•	•	•	•	11	011	101	3	5	19	011	E
								01	r	110				100	H
								←	d	→				101	L
LD r, (IY+d)	r ← (IY+d)	•	•	•	•	•	•	11	111	101	3	5	19	111	A
								01	r	110					
								←	d	→					
LD (HL), r	(HL) ← r	•	•	•	•	•	•	01	110	r	1	2	7		
LD (IX+d), r	(IX+d) ← r	•	•	•	•	•	•	11	011	101	3	5	19		
								01	110	r					
								←	d	→					
LD (IY+d), r	(IY+d) ← r	•	•	•	•	•	•	11	111	101	3	5	19		
								01	110	r					
								←	d	→					
LD (HL), n	(HL) ← n	•	•	•	•	•	•	00	110	110	2	3	10		
								←	n	→					
LD (IX+d), n	(IX+d) ← n	•	•	•	•	•	•	11	011	101	4	5	19		
								00	110	110					
								←	d	→					
								←	n	→					
LD (IY+d), n	(IY+d) ← n	•	•	•	•	•	•	11	111	101	4	5	19		
								00	110	110					
								←	d	→					
								←	n	→					
LD A, (BC)	A ← (BC)	•	•	•	•	•	•	00	001	010	1	2	7		
LD A, (DE)	A ← (DE)	•	•	•	•	•	•	00	011	010	1	2	7		
LD A, (nn)	A ← (nn)	•	•	•	•	•	•	00	111	010	3	4	13		
								←	n	→					
								←	n	→					
LD (BC), A	(BC) ← A	•	•	•	•	•	•	00	000	010	1	2	7		
LD (DE), A	(DE) ← A	•	•	•	•	•	•	00	010	010	1	2	7		
LD (nn), A	(nn) ← A	•	•	•	•	•	•	00	110	010	3	4	13		
								←	n	→					
								←	n	→					
LD A, I	A ← I	•	‡	IFF	‡	0	0	11	101	101	2	2	9		
								01	010	111					
LD A, R	A ← R	•	‡	IFF	‡	0	0	11	101	101	2	2	9		
								01	011	111					
LD I, A	I ← A	•	•	•	•	•	•	11	101	101	2	2	9		
								01	000	111					
LD R, A	R ← A	•	•	•	•	•	•	11	101	101	2	2	9		
								01	001	111					

Notes: r, r' means any of the registers A, B, C, D, E, H, L

IFF the content of the interrupt enable flip-flop (IFF) is copied into the P/V flag

Flag Notation: • = flag not affected, 0 = flag reset, 1 = flag set, X = flag is unknown.

‡ = flag is affected according to the result of the operation

8-BIT LOAD GROUP

Mnemonic	Symbolic Operation	Flags C	Z	V/P	S	H	N	Op Code	No. of Bytes	No. of M Cycles	No. of T States	Comments
LD dd,nn	dd ← nn	•	•	•	•	•	•	00 dd0 001 ← n → ← n →	3	3	10	dd Pair 00 BC 01 DE 10 HL 11 SP
LD IX,nn	IX ← nn	•	•	•	•	•	•	11 011 101 00 100 001 ← n → ← n →	4	4	14	
LD IY,nn	IY ← nn	•	•	•	•	•	•	11 111 101 00 100 001 ← n → ← n →	4	4	14	
LD HL,(nn)	H ← (nn+1) L ← (nn)	•	•	•	•	•	•	00 101 010 ← n → ← n →	3	5	16	
LD dd,(nn)	dd_H ← (nn+1) dd_L ← (nn)	•	•	•	•	•	•	11 101 101 01 dd1 011 ← n → ← n →	4	6	20	
LD IX,(nn)	IX_H ← (nn+1) IX_L ← (nn)	•	•	•	•	•	•	11 011 101 00 101 010 ← n → ← n →	4	6	20	
LD IY,(nn)	IY_H ← (nn+1) IY_L ← (nn)	•	•	•	•	•	•	11 111 101 00 101 010 ← n → ← n →	4	6	20	
LD (nn),HL	(nn+1) ← H (nn) ← L	•	•	•	•	•	•	00 100 010 ← n → ← n →	3	5	16	
LD (nn),dd	(nn+1) ← dd_H (nn) ← dd_L	•	•	•	•	•	•	11 101 101 01 dd0 011 ← n → ← n →	4	6	20	
LD (nn),IX	(nn+1) ← IX_H (nn) ← IX_L	•	•	•	•	•	•	11 011 101 00 100 010 ← n → ← n →	4	6	20	
LD (nn),IY	(nn+1) ← IY_H (nn) ← IY_L	•	•	•	•	•	•	11 111 101 00 100 010 ← n → ← n →	4	6	20	
LD SP,HL	SP ← HL	•	•	•	•	•	•	11 111 001	1	1	6	
LD SP,IX	SP ← IX	•	•	•	•	•	•	11 011 101 11 111 001	2	2	10	
LD SP,IY	SP ← IY	•	•	•	•	•	•	11 111 101 11 111 001	2	2	10	qq Pair 00 BC 01 DE 10 HL 11 AF
PUSH qq	(SP-2) ← qq_L (SP-1) ← qq_H	•	•	•	•	•	•	11 qq0 101	1	3	11	
PUSH IX	(SP-2) ← IX_L (SP-1) ← IX_H	•	•	•	•	•	•	11 011 101 11 100 101	2	4	15	
PUSH IY	(SP-2) ← IY_L (SP-1) ← IY_H	•	•	•	•	•	•	11 111 101 11 100 101	2	4	15	
POP qq	qq_H ← (SP+1) qq_L ← (SP)	•	•	•	•	•	•	11 qq0 001	1	3	10	
POP IX	IX_H ← (SP+1) IX_L ← (SP)	•	•	•	•	•	•	11 011 101 11 100 001	2	4	14	
POP IY	IY_H ← (SP+1) IY_L ← (SP)	•	•	•	•	•	•	11 111 101 11 100 001	2	4	14	

Notes: dd is any of the register pairs BC, DE, HL, SP
qq is any of the register pairs AF, BC, DE, HL
$(PAIR)_H$, $(PAIR)_L$ refer to high order and low order eight bits of the register pair respectively
E.g. BC_L = C, AF_H = A

Flag Notation: • = flag not affected, 0 = flag reset, 1 = flag set, X = flag is unknown,
↕ flag is affected according to the result of the operation

Mnemonic	Symbolic Operation	Flags C	Z	P/V	S	N	H	Op Code 76 543 210	No. of Bytes	No. of M Cycles	No. of T States	Comments
EX DE, HL	DE ↔ HL	•	•	•	•	•	•	11 101 011	1	1	4	
EX AF, AF'	AF ↔ AF'	•	•	•	•	•	•	00 001 000	1	1	4	
EXX	(BC) (DE) (HL) ↔ (BC') (DE') (HL')	•	•	•	•	•	•	11 011 001	1	1	4	Register bank and auxiliary register bank exchange
EX (SP), HL	H ↔ (SP+1) L ↔ (SP)	•	•	•	•	•	•	11 100 011	1	5	19	
EX (SP), IX	IX$_H$ ↔ (SP+1) IX$_L$ ↔ (SP)	•	•	•	•	•	•	11 011 101 11 100 011	2	6	23	
EX (SP), IY	IY$_H$ ↔ (SP+1) IY$_L$ ↔ (SP)	•	•	•	•	•	•	11 111 101 11 100 011	2	6	23	
LDI	(DE) ← (HL) DE ← DE+1 HL ← HL+1 BC ← BC−1	•	•	①	1	•	0	11 101 101 10 100 000	2	4	16	Load (HL) into (DE), increment the pointers and decrement the byte counter (BC)
LDIR	(DE) ← (HL) DE ← DE+1 HL ← HL+1 BC ← BC−1 Repeat until BC = 0	•	•	①	0	•	0	11 101 101 10 110 000	2	5 4	21 16	If BC ≠ 0 If BC = 0
LDD	(DE) ← (HL) DE ← DE−1 HL ← HL−1 BC ← BC−1	•	•	①	1	•	0	11 101 101 10 101 000	2	4	16	
LDDR	(DE) ← (HL) DE ← DE−1 HL ← HL−1 BC ← BC−1 Repeat until BC = 0	•	•	①	0	•	0	11 101 101 10 111 000	2 2	5 4	21 16	If BC ≠ 0 If BC = 0
CPI	A − (HL) HL ← HL+1 BC ← BC−1	•	②	①	②	1	②	11 101 101 10 100 001	2	4	16	
CPIR	A − (HL) HL ← HL+1 BC ← BC−1 Repeat until A = (HL) or BC = 0	•	②	①	②	1	②	11 101 101 10 110 001	2 2	5 4	21 16	If BC ≠ 0 and A ≠ (HL) If BC = 0 or A = (HL)
CPD	A − (HL) HL ← HL−1 BC ← BC−1	•	②	①	②	1	②	11 101 101 10 101 001	2	4	16	
CPDR	A − (HL) HL ← HL−1 BC ← BC−1 Repeat until A = (HL) or BC = 0	•	②	①	②	1	②	11 101 101 10 111 001	2 2	5 4	21 16	If BC ≠ 0 and A ≠ (HL) If BC = 0 or A = (HL)

Notes: ① P/V flag is 0 if the result of BC−1 = 0, otherwise P/V = 1
② Z flag is 1 if A = (HL), otherwise Z = 0.

Flag Notation: • flag not affected, 0 = flag reset, 1 = flag set, X = flag is unknown
↕ = flag is affected according to the result of the operation.

EXCHANGE GROUP AND BLOCK TRANSFER AND SEARCH GROUP

Mnemonic	Symbolic Operation	Flags C	Z	P/V	S	N	H	Op Code 76 543 210	No. of Bytes	No. of M Cycles	No. of T States	Comments
ADD A,r	A ← A + r	↕	↕	V	↕	0	↕	10 000 r	1	1	4	r Reg.
ADD A,n	A ← A + n	↕	↕	V	↕	0	↕	11 000 110 n	2	2	7	000 B 001 C 010 D 011 E
ADD A,(HL)	A ← A + (HL)	↕	↕	V	↕	0	↕	10 000 110	1	2	7	100 H 101 L 111 A
ADD A,(IX+d)	A ← A + (IX+d)	↕	↕	V	↕	0	↕	11 011 101 10 000 110 d	3	5	19	
ADD A,(IY+d)	A ← A+(IY+d)	↕	↕	V	↕	0	↕	11 111 101 10 000 110 d	3	5	19	
ADC A,s	A ← A + s + CY	↕	↕	V	↕	0	↕	001				s is any of r, n, (HL), (IX+d), (IY+d) as shown for ADD instruction
SUB s	A ← A - s	↕	↕	V	↕	1	↕	010				
SBC A,s	A ← A - s - CY	↕	↕	V	↕	1	↕	011				
AND s	A ← A ∧ s	0	↕	P	↕	0	1	100				The indicated bits replace the 000 in the ADD set above
OR s	A ← A ∨ s	0	↕	P	↕	0	0	110				
XOR s	A ← A ⊕ s	0	↕	P	↕	0	0	101				
CP s	A - s	↕	↕	V	↕	1	↕	111				
INC r	r ← r + 1	•	↕	V	↕	0	↕	00 r 100	1	1	4	
INC (HL)	(HL) ← (HL)+1	•	↕	V	↕	0	↕	00 110 100	1	3	11	
INC (IX+d)	(IX+d) ← (IX+d)+1	•	↕	V	↕	0	↕	11 011 101 00 110 100 d	3	6	23	
INC (IY+d)	(IY+d) ← (IY+d) + 1	•	↕	V	↕	0	↕	11 111 101 00 110 100 d	3	6	23	
DEC m	m ← m-1	•	↕	V	1	1	↕	101				m is any of r, (HL), (IX+d), (IY+d) as shown for INC. Same format and states as INC. Replace 100 with 101 in OP code

Notes: The V symbol in the P/V flag column indicates that the P/V flag contains the overflow of the result of the operation. Similarly, the P symbol indicates parity, V = 1 means overflow, V = 0 means no overflow. P = 1 means parity of the result is even, P = 0 means parity of the result is odd.

Flag Notation: • = flag not affected, 0 = flag reset, 1 = flag set, X = flag is unknown
↕ = flag is affected according to the result of the operation

8-BIT ARITHMETIC AND LOGICAL GROUP

Mnemonic	Symbolic Operation	Flags							Op-Code	No. of Bytes	No. of M Cycles	No. of T States	Comments
		C	Z	P/V	S	N	H		76 543 210				
DAA	Converts acc. content into packed BCD following add or subtract with packed BCD operands	↕	↕	P	↕	•	↕		00 100 111	1	1	4	Decimal adjust accumulator
CPL	$A \leftarrow \overline{A}$	•	•	•	•	1	1		00 101 111	1	1	4	Complement accumulator (one's complement)
NEG	$A \leftarrow 0 - A$	↕	↕	V	↕	1	↕		11 101 101 01 000 100	2	2	8	Negate acc. (two's complement)
CCF	$CY \leftarrow \overline{CY}$	↕	•	•	•	0	X		00 111 111	1	1	4	Complement carry flag
SCF	CY ← 1	1	•	•	•	0	0		00 110 111	1	1	4	Set carry flag
NOP	No operation	•	•	•	•	•	•		00 000 000	1	1	4	
HALT	CPU halted	•	•	•	•	•	•		01 110 110	1	1	4	
DI	IFF ← 0	•	•	•	•	•	•		11 110 011	1	1	4	
EI	IFF ← 1	•	•	•	•	•	•		11 111 011	1	1	4	
IM 0	Set interrupt mode 0	•	•	•	•	•	•		11 101 101 01 000 110	2	2	8	
IM 1	Set interrupt mode 1	•	•	•	•	•	•		11 101 101 01 010 110	2	2	8	
IM2	Set interrupt mode 2	•	•	•	•	•	•		11 101 101 01 011 110	2	2	8	

Notes: IFF indicates the interrupt enable flip-flop
CY indicates the carry flip-flop.

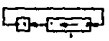
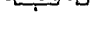
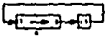
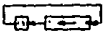
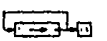
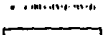
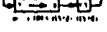
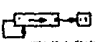
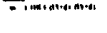
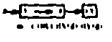
Flag Notation: • = flag not affected, 0 = flag reset, 1 = flag set, X = flag is unknown,
↕ = flag is affected according to the result of the operation.

Mnemonic	Symbolic Operation	Flags C	Z	$P_{/V}$	S	N	H	Op-Code 76 543 210	No. of Bytes	No. of M Cycles	No. of T States	Comments	
ADD HL, ss	HL ← HL + ss	↕	•	•	•	0	X	00 ss1 001	1	3	11	ss	Reg.
												00	BC
												01	DE
ADC HL, ss	HL ← HL + ss + CY	↕	↕	V	↕	0	X	11 101 101	2	4	15	10	HL
								01 ss1 010				11	SP
SBC HL, ss	HL ← HL − ss − CY	↕	↕	V	↕	1	X	11 101 101	2	4	15		
								01 ss0 010					
ADD IX, pp	IX ← IX + pp	↕	•	•	•	0	X	11 011 101	2	4	15	pp	Reg.
								00 pp1 001				00	BC
												01	DE
												10	IX
												11	SP
ADD IY, rr	IY ← IY + rr	↕	•	•	•	0	X	11 111 101	2	4	15	rr	Reg.
								00 rr1 001				00	BC
												01	DE
												10	IY
												11	SP
INC ss	ss ← ss + 1	•	•	•	•	•	•	00 ss0 011	1	1	6		
INC IX	IX ← IX + 1	•	•	•	•	•	•	11 011 101	2	2	10		
								00 100 011					
INC IY	IY ← IY + 1	•	•	•	•	•	•	11 111 101	2	2	10		
								00 100 011					
DEC ss	ss ← ss − 1	•	•	•	•	•	•	00 ss1 011	1	1	6		
DEC IX	IX ← IX − 1	•	•	•	•	•	•	11 011 101	2	2	10		
								00 101 011					
DEC IY	IY ← IY − 1	•	•	•	•	•	•	11 111 101	2	2	10		
								00 101 011					

Notes: ss is any of the register pairs BC, DE, HL, SP
pp is any of the register pairs BC, DE, IX, SP
rr is any of the register pairs BC, DE, IY, SP.

Flag Notation: • = flag not affected, 0 = flag reset, 1 = flag set, X = flag is unknown,
↕ = flag is affected according to the result of the operation.

16-BIT ARITHMETIC GROUP

Mnemonic	Symbolic Operation	Flags C	Z	P/V	S	N	H	Op Code 76 543 210	No. of Bytes	No. of M Cycles	No. of T States	Comments
RLCA		1	•	•	•	0	0	00 000 111	1	1	4	Rotate left circular accumulator
RLA		1	•	•	•	0	0	00 010 111	1	1	4	Rotate left accumulator
RRCA		1	•	•	•	0	0	00 001 111	1	1	4	Rotate right circular accumulator
RRA		1	•	•	•	0	0	00 011 111	1	1	4	Rotate right accumulator
RLC r		1	1	P	1	0	0	11 001 011 00 000 r	2	2	8	Rotate left circular register r
RLC (HL)		1	1	P	1	0	0	11 001 011 00 000 110	2	4	15	r Reg. 000 B 001 C 010 D 011 E 100 H 101 L 111 A
RLC (IX+d)		1	1	P	1	0	0	11 011 101 11 001 011 ← d → 00 000 110	4	6	23	
RLC (IY+d)		1	1	P	1	0	0	11 111 101 11 001 011 ← d → 00 000 110	4	6	23	
RL m		1	1	P	1	0	0	010				Instruction format and states are as shown for RLC m. To form new OP-code replace 000 of RLC m with shown code
RRC m		1	1	P	1	0	0	001				
RR m		1	1	P	1	0	0	011				
SLA m		1	1	P	1	0	0	100				
SRA m		1	1	P	1	0	0	101				
SRL m		1	1	P	1	0	0	111				
RLD		•	1	P	1	0	0	11 101 101 01 101 111	2	5	18	Rotate digit left and right between the accumulator and location (HL). The content of the upper half of the accumulator is unaffected
RRD		•	1	P	1	0	0	11 101 101 01 100 111	2	5	18	

Flag Notation: • = flag not affected, 0 = flag reset, 1 = flag set, X = flag is unknown, 1 = flag is affected according to the result of the operation.

ROTATE AND SHIFT GROUP

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

Notes: The notation s_b indicates bit b (0 to 7) of location s.

Flag Notation: • = flag not affected, 0 = flag reset, 1 = flag set, X = flag is unknown, ‡ = flag is affected according to the result of the operation

BIT SET, RESET AND TEST GROUP

Mnemonic	Symbolic Operation	Flags C	Z	P/V	S	N	H	Op Code 76 543 210	No. of Bytes	No. of M Cycles	No. of T States	Comments
JP nn	PC ← nn	•	•	•	•	•	•	11 000 011 ← n → ← n →	3	3	10	
JP cc, nn	If condition cc is true PC ← nn, otherwise continue	•	•	•	•	•	•	11 cc 010 ← n → ← n →	3	3	10	cc Condition 000 NZ non zero 001 Z zero 010 NC non carry 011 C carry 100 PO parity odd 101 PE parity even 110 P sign positive 111 M sign negative
JR e	PC ← PC + e	•	•	•	•	•	•	00 011 000 ← e-2 →	2	3	12	
JR C, e	If C = 0, continue	•	•	•	•	•	•	00 111 000 ← e-2 →	2	2	7	If condition not met
	If C = 1, PC ← PC+e								2	3	12	If condition is met
JR NC, e	If C = 1, continue	•	•	•	•	•	•	00 110 000 ← e-2 →	2	2	7	If condition not met
	If C = 0, PC ← PC + e								2	3	12	If condition is met
JR Z, e	If Z = 0 continue	•	•	•	•	•	•	00 101 000 ← e-2 →	2	2	7	If condition not met
	If Z = 1, PC ← PC + e								2	3	12	If condition is met
JR NZ, e	If Z = 1, continue	•	•	•	•	•	•	00 100 000 ← e-2 →	2	2	7	If condition not met
	If Z = 0, PC ← PC + e								2	3	12	If condition met
JP (HL)	PC ← HL	•	•	•	•	•	•	11 101 001	1	1	4	
JP (IX)	PC ← IX	•	•	•	•	•	•	11 011 101 11 101 001	2	2	8	
JP (IY)	PC ← IY	•	•	•	•	•	•	11 111 101 11 101 001	2	2	8	
DJNZ, e	B ← B-1 If B = 0, continue	•	•	•	•	•	•	00 010 000 ← e-2 →	2	2	8	If B = 0
	If B ≠ 0, PC ← PC + e								2	3	13	If B ≠ 0

Notes: e represents the extension in the relative addressing mode

e is a signed two's complement number in the range <-126, 129>

e-2 in the op-code provides an effective address of pc + e as PC is incremented by 2 prior to the addition of e.

Flag Notation: • = flag not affected, 0 = flag reset, 1 = flag set, X = flag is unknown, ↕ = flag is affected according to the result of the operation.

JUMP GROUP

Mnemonic	Symbolic Operation	Flags C	Z	P/V	S	N	H	Op-Code 76 543 210	No. of Bytes	No. of M Cycles	No. of T States	Comments
CALL nn	$(SP-1)\leftarrow PC_H$ $(SP-2)\leftarrow PC_L$ $PC\leftarrow nn$	•	•	•	•	•	•	11 001 101 ← n → ← n →	3	5	17	
CALL cc, nn	If condition cc is false continue, otherwise same as CALL nn	•	•	•	•	•	•	11 cc 100 ← n → ← n →	3	3	10	If cc is false
									3	5	17	If cc is true
RET	$PC_L\leftarrow(SP)$ $PC_H\leftarrow(SP+1)$	•	•	•	•	•	•	11 001 001	1	3	10	
RET cc	If condition cc is false continue, otherwise same as RET	•	•	•	•	•	•	11 cc 000	1	1	5	If cc is false
									1	3	11	If cc is true
RETI	Return from interrupt	•	•	•	•	•	•	11 101 101 01 001 101	2*	4	14	
RETN	Return from non maskable interrupt	•	•	•	•	•	•	11 101 101 01 000 101	2	4	14	
RST p	$(SP-1)\leftarrow PC_H$ $(SP-2)\leftarrow PC_L$ $PC_H\leftarrow 0$ $PC_L\leftarrow P$	•	•	•	•	•	•	11 t 111	1	3	11	

cc	Condition	
000	NZ	non zero
001	Z	zero
010	NC	non carry
011	C	carry
100	PO	parity odd
101	PE	parity even
110	P	sign positive
111	M	sign negative

t	P
000	00H
001	08H
010	10H
011	18H
100	20H
101	28H
110	30H
111	38H

Flag Notation: • = flag not affected, 0 = flag reset, 1 = flag set, X = flag is unknown
↕ = flag is affected according to the result of the operation.

CALL AND RETURN GROUP

Mnemonic	Symbolic Operation	Flags C	Z	P/V	S	N	H	Op-Code 76 543 210	No. of Bytes	No. of M Cycles	No. of T States	Comments
IN A, (n)	A ← (n)	•	•	•	•	•	•	11 011 011 ← n →	2	3	11	n to A_0 ~ A_7 Acc to A_8 ~ A_{15}
IN r, (C)	r ← (C) if r = 110 only the flags will be affected	•	↕	P	↕	0	↕	11 101 101 01 r 000	2	3	12	C to A_0 ~ A_7 B to A_8 ~ A_{15}
INI	(HL) ← (C) B ← B - 1 HL ← HL + 1	•	① ↕	X	X	1	X	11 101 101 10 100 010	2	4	16	C to A_0 ~ A_7 B to A_8 ~ A_{15}
INIR	(HL) ← (C) B ← B - 1 HL ← HL + 1 Repeat until B = 0	•	① ↕	X	X	1	X	11 101 101 10 110 010	2 2	5 (If B ≠ 0) 4 (If B = 0)	21 16	C to A_0 ~ A_7 B to A_8 ~ A_{15}
IND	(HL) ← (C) B ← B - 1 HL ← HL - 1	•	① ↕	X	X	1	X	11 101 101 10 101 010	2	4	16	C to A_0 ~ A_7 B to A_8 ~ A_{15}
INDR	(HL) ← (C) B ← B - 1 HL ← HL -1 Repeat until B = 0	•	↕	X	X	1	X	11 101 101 10 111 010	2 2	5 (If B ≠ 0) 4 (If B = 0)	21 16	C to A_0 ~ A_7 B to A_8 ~ A_{15}
OUT (n), A	(n) ← A	•	•	•	•	•	•	11 010 011 ← n →	2	3	11	n to A_0 ~ A_7 Acc to A_8 ~ A_{15}
OUT (C), r	(C) ← r	•	•	•	•	•	•	11 101 101 01 r 001	2	3	12	C to A_0 ~ A_7 B to A_8 ~ A_{15}
OUTI	(C) ← (HL) B ← B - 1 HL ← HL + 1	•	① ↕	X	X	1	X	11 101 101 10 100 011	2	4	16	C to A_0 ~ A_7 B to A_8 ~ A_{15}
OTIR	(C) ← (HL) B ← B - 1 HL ← HL + 1 Repeat until B = 0	•	① ↕	X	X	1	X	11 101 101 10 110 011	2 2	5 (If B ≠ 0) 4 (If B = 0)	21 16	C to A_0 ~ A_7 B to A_8 ~ A_{15}
OUTD	(C) ← (HL) B ← B - 1 HL ← HL - 1	•	① ↕	X	X	1	X	11 101 101 10 101 011	2	4	16	C to A_0 ~ A_7 B to A_8 ~ A_{15}
OTDR	(C) ← (HL) B ← B - 1 HL ← HL -1 Repeat until B = 0	•	↕	X	X	1	X	11 101 101 10 111 011	2 2	5 (If B ≠ 0) 4 (If B = 0)	21 16	C to A_0 ~ A_7 B to A_8 ~ A_{15}

Notes: ① If the result of B - 1 is zero the Z flag is set, otherwise it is reset.

Flag Notation: • = flag not affected, 0 = flag reset, 1 = flag set, X = flag is unknown, ↕ = flag is affected according to the result of the operation

INPUT AND OUTPUT GROUP