Curso de educacion continua sobre micropro_ cesadores

Jara, Freddy

UNIVERSIDAD DE CARABOBO

FACULTAD DE INGENIERIA

CURSO DE EDUCACION CONTINUA SOBRE

MICROPROCESADORES

Trabajo presentado ante el Ilustre Consejo de Facultad de Ingeniería por los Profesores Hyxia Villegas y Fredy Jara, para ascender a la categoría de Profesor Agregado.

ING. HYXIA VILLEGAS
ING. FREDY JARA

DICIEMBRE 1984

I N D I C E

CAPITULO I INTRODUCCION A LOS MICROPROCESADORES

1.1. INTRODUCCION, 1
1.2. MICROPROCESADORES Y MICROCOMPUTADORES, 2
1.2.1. GENERALIDADES, 2
1.2.2. EL MICROPROCESADOR, 4
1.2.3. CARACTERISTICAS BASICAS DE UN MICROPROCESADOR, 5
REGISTROS, 5
UNIDAD ARITMETICA Y LOGICA, 6
UNIDAD DE CONTROL, 7
PUERTOS DE ENTRADA Y SALIDA, 7
BUSES, 7
BUS DE DATOS, 8
BUS DE CONTROL, 10
BUS DE DIRECCIONES, 11

1.2.4. OPERACION DE UN MICROPROCESADOR, 13
FASE DE CAPTACION, 14
FASE DE EJECUCION, 14

1.2.5. MICROCOMPUTADORES, 20

1.3. INTERFASES CON EL MUNDO REAL, 20

CAPITULO II ARQUITECTURA Y ORGANIZACION DEL 8035

2.1. ARQUITECTURA DE UN COMPUTADOR, 23
2.1.1. REGISTROS, 24
2.1.2. MEMORIA, 27
2.1.3. UNIDAD DE CONTROL, 31
2.1.4. UNIDAD DE PROCESO, 33
2.1.5. UNIDADES DE ENTRADA Y SALIDA, 35
E/S MANEJADA POR PROGRAMA, 37
E/S MANEJADA POR INTERRUPCION, 38
E/S MANEJADA POR DMA, 39

2.2. ARQUITECTURA DEL 8035, 40
2.2.1. UNIDAD DE PROCESO, 43
2.2.1.1. UNIDAD ARITMETICA Y LOGICA DEL 8035, 43
2.2.1.2. ACUMULADOR Y REGISTRO AUXILIAR, 44

3.3.3. INSTRUCCIONES PARA MANIPULACION DE PROGRAMA, 108
3.3.4. INSTRUCCIONES PARA MANIPULACION DE ESTATUS DE PROGRAMA, 118
3.3.5. INSTRUCCIONES PARA MANEJO DE ENTRADA/SALIDA, 119
3.3.6. INSTRUCCIONES PARA MANEJO DEL TIMER, 122

CAPITULO IV SISTEMA IMSAI

4.1. SISTEMA IMSAI, 123
4.1.1. DESCRIPCION FUNCIONAL, 123
4.1.2. TEORIA DE OPERACION, 128
4.1.3. CONEXIONES DISPONIBLES EN EL IMSAI, 136

4.2. INTERFASES DE E/S, 139
4.2.1. EXPANSOR DE PUERTOS 8243, 139
4.2.1.1. DESCRIPCION DEL 8243, 139
4.2.2. MANEJADOR DE TECLADO Y DISPLAY 8279, 146
4.2.2.1. TEORIA BASICA DE TECLADO Y DISPLAY, 146
4.2.2.2. DESCRIPCION FUNCIONAL DEL 8279, 159
4.2.2.3. PROGRAMACION DEL 8279, 168
4.2.2.4. DESCRIPCIONDE PINES DEL 8279, 177

4.3. OTRAS CARACTERISTICAS DEL SISTEMA IMSAI, 179

4.4. MANUAL DEL USUARIO, 182

CAPITULO V CONEXION DE MICROPROCESADOR CON PERIFERICOS Y MEMORIA

5.1. INTRODUCCION, 185

5.2. CONEXION CON MEMORIA, 185
5.2.1. DECODIFICACION TOTAL, 186
5.2.2. DECODIFICACION PARCIAL, 188
5.2.3. BANCOS DE MEMORIA, 191
5.2.4. TIEMPO PARA LEER Y ESCRIBIR DE MEMORIA, 192

5.3. CONEXION CON PERIFERICOS LENTOS, 194

PRACTICA 8, 296

APENDICES, 298

APENDICE A, 299

APENDICE B, 300

APENDICE C, 301

APENDICE D, 302

BIBLIOGRAFIA, 303

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

Los microprocesadores han venido a constituirse en la revolución mas grande de la electrónica de los últimos años; su desarrollo ha producido un cambio drástico en el diseño de los sistemas electrónicos, abriendo nuevos campos de aplicación, que incluyen todas las areas de investigación pura y aplicada, en practicamente todos los campos de la ingeniería.

Los microprocesadores han permitido tambien masificar el uso de los computadores, pues en virtud de su relativamente bajo costo y a su extraordinario poder y versatilidad, se han desarrollado los microcomputadores, los cuales tienen un precio bastante accesible.

El presente trabajo está dirigido principalmente a los ingenieros ya graduados que deseen introducirse en el mundo de los microprocesadores a fin de conocerlos mas de cerca, lo que les permitirá explotar al máximo sus posibilidades y les abrirá la capacidad de poder introducir este nuevo desarrollo de la electrónica moderna, en sus respectivas areas de trabajo. Este trabajo se ha concebido como un curso básico integral sobre la operación y utilización de los microprocesadores; en el mismo, se ha tratado en lo posible de presentar los conceptos y explicaciones, de una manera sencilla que pueda ser comprendida en forma relativamente facil, por un lector con conocimientos básicos de lógica digital y que esté familiarizado con algún lenguaje de programación. En algunos casos, se ha hecho un breve recordatorio de tópicos que se supone el lector deba conocer, con la finalidad de facilitarle al mismo, la comprensión de las nuevas ideas que se introducen.

El desarrollo de un trabajo para enseñar microprocesadores, involucra siempre la necesidad de escoger o seleccionar uno o varios de estos elemento, a fin de permitir realizar experimentos y ejercicios sobre la base de dispositivos reales, que permitan llevar a la práctica las soluciones y desarrollos teóricos, lo cual será lo que en definitiva permitirá que el lector asimile y comprenda cabalmente los conceptos que se pretenden introducir; en este trabajo se ha seleccionado el microprocesador 8048/8035 de INTEL, para la implementación de todos los ejercicios y experimentos, en primer lugar por que consideramos que para los

efectos de la enseñanza del principio básico de operación de los microprocesadores, se puede utilizar cualquier microprocesador y en segundo lugar, por que el sistema de desarrollo de que dispone actualmente el laboratorio de microprocesadores, está basado en ese microprocesador(8048/8035), y en el se implementaron todos los experimentos del presente trabajo. El microprocesador 8048/8035 es muy usado actualmente, debido a que el contiene todos los elementos necesarios para el desarrollo de un sistema sencillo, como son: memoria ROM, memoria RAM, puertos, reloj, contador de eventos, etc., integrados en un solo cómponente.

El trabajo se ha organizado en dos partes: una primera parte, constituida por 6 capítulos, en donde se dan las bases teóricas de la operación de los microprocesadores, y la otra parte, constituida por una serie de experimentos, agrupados en 8 prácticas, que tienen como ya se dijo antes, la finalidad de afianzar los conocimientos expuestos en la parte teórica.

La parte teórica, como ya se dijo, la hemos dividido en 6 capítulos; se comienza con un capítulo introductorio al mundo de los microprocesadores, en el cual, se introduce el concepto de microprocesador y microcomputador y se describe el principio básico general de operación de los mismos.

En el segundo capítulo, se tratan en forma mas específica algunos de los conceptos presentados en el CAPITULO I y se introducen conceptos básicos de arquitectura de microprocesadores, estudiando de manera especial y en detalle, lo referente a la arquitectura del microprocesador 8035 de INTEL, con el cual como ya se dijo antes, se realizan todos los experimentos y ejercicios del presente trabajo.

En el CAPITULO III, hacemos un parentesis en lo referente al hardware y operación de los microprocesadores, para introducir los conceptos básicos de programación de computadores; en la primera parte de este capítulo se tratan brevemente los distintos lenguajes de programación (de bajo y alto nivel), y algunas técnicas y herramientas de uso general en programación (diseño Top-Down, subrutinas, diagramas de flujo, etc.), haciendo enfasis especial en el lenguaje assembler, por ser el lenguaje con que se trabajará en el curso; al respecto, se dan algunas técnicas y recomendaciones útiles para el desarrollo de programas con este tipo de lenguaje, que consideramos seran de gran utilidad a la hora de la implemantación de programas. En la segunda parte de este CAPITULO III, se trata en forma específica el conjunto de instrucciones básicas de que dispone el 8035, complementándose con algunos ejemplos de programación, que permiten ilustrar el

uso de las técnicas descritas en la primera parte del capítulo y familiarizar al lector con el conjunto de instrucciones del 8035.

El CAPITULO IV, constituye una descripción detallada del sistema IMSAI, el cual fue utilizado para el desarrollo de todos los ejercicios y experimentos de este trabajo; en este capítulo se describen en forma detallada las interfases con que cuenta el sistema IMSAI como son el manejador de teclado y display 8279 y el expansor de puertos 8243, los programas y facilidades que trae el monitor del sistema y en general, todo los elementos de software y hardware de que dispone este sistema de desarrollo.

El CAPITULO V, trata de la expansión y conexión de elementos adicionales al sistema básico de desarrollo IMSAI; en este capítulo se enseña las consideraciones y forma de conectar expansiones de memoria, técnicas de conexión de equipos periféricos (rápidos y lentos) y la conexión con el mundo analógico mediante los conversores A/D y D/A.

El CAPITULO VI, trata acerca de la comunicación serial y de interfases estandares. En la primera parte de este capítulo se da una breve introducción de los conceptos básicos de la comunicación serial, describiendose brevemente los diferentes tipos de comunicación serial y los patrones adoptados generalmente para esos tipos de comunicación; adicionalmente se describen en esta primera parte componentes electrónicos de alta escala de ingración, diseñados especialmente para facilitar la comunicación serial de los microprocesadores con otros dispositivos que usen esta modalidad de comunicación; especificamente en este trabajo se describen en detalle el USART 8251A y el UART AY-5-1013. En la segunda parte de este capítulo se trata lo referente a diferentes estandares que se han desarrollado para normar ciertas conexiones como son: la interfase estandard RS-232C para comunicación serial; el bus S-100 para la expansión de los sistemas a microprocesadores y el bus IEEE 488, para la interconexión de instrumentos controlados por microprocesadores o computadores.

Las prácticas o experimentos se han organizado de la siguiente forma:

La PRACTICA 1 tiene por objetivo primordial familiarizar al estudiante con el sistema IMSAI, y con su programa monitor.

La PRACTICA 2 tiene por objetivo continuar el proceso de familiarización del estudiante con los comandos y rutinas

del monitor del sistema IMSAI para el manejo del teclado y el display; tambien en esta práctica se trata de examinar y hacer notar en el estudiante, algunas fallas comunes en las que incurren los que se inician en la programación de microprocesadores.

La PRACTICA 3 tiene por objetivo enseñar al estudiante la forma como el sistema IMSAI maneja el teclado, asi como tambien el uso de los bancos de registros (0 y 1) y los bancos de memoria (0 y 1) de que dispone el 8035 y los distintos modos de direccionamiento disponibles en el mismo.

La PRACTICA 4 versa sobre la familiarización del estudiante con el manejo de los distintos puertos del 8035 y con el expansor de puertos 8243. Se trata tambien de familiarizar al estudiante con el cálculo de la interfase (consideraciones de tiempo y carga), para la conexión de equipos a los distintos puertos.

La PRACTICA 5 tiene por objetivo familiarizar al estudiante con el manejador de teclado y display 8279; enseñar al estudiante el uso como entrada serial de la línea T1 del 8035 y el uso y aplicación del TIMER/COUNTER del mismo.

La practica 6 trata sobre la familiarización del estudiante con la conexión de memoria adicional a un sistema a microprocesador; la conexión de un transductor de audio para generación de sonidos por medio del microprocesador y con las facilidades de almacenamiento de información en cassette de que dispone el sistema IMSAI.

La PRACTICA 7 tiene por finalidad familiarizar al estudiante con la conexión de conversores A/D y D/A a los microprocesadores y con la utilización de los analizadores lógicos.

La PRACTICA 8 tiene por finalidad familiarizar al estudiante con todos los aspectos de la comunicación serial y con los elementos o dispositivos electrónicos que generalmente se usan para su realización.

C A P I T U L O I

1.1. INTRODUCCION

El microprocesador ha sido un desarrollo de la electrónica que ha convulsionado el mundo tecnológico, siendo imprescindible para el ingeniero actual, conocerlo y saberlo usar en virtud de que dicho dispositivo ha invadido todos los campos de la ingeniería.

Los microprocesadores han abierto nuevos caminos facilitando el diseño de sistemas complejos y posibilitando la introducción de nuevas opciones en sistemas que desarrollados con la vieja tecnología, hace pocos años hubiesen parecido de ficción, bien sea por su costo, complejidad o tamaño. Entre las muchas ventajas que proporcionan los microprocesadores se cuenta la reducción de costos en aplicaciones donde decenas de circuitos integrados pueden reemplazarze por un microprocesador, con una mayor confiabilidad del sistema al reducirse el número de interconexiones y los componentes electrónicos discretos.

En la actualidad, los microprocesadores están siendo utilizados en numerosas y muy variadas aplicaciones cubriendo un espectro que ya desde las comunicaciones, instrumentos, microcomputadores hasta productos de consumo general. Se ha hecho familiar conseguir microprocesadores en sistemas de control de procesos, control numérico, terminales inteligentes, sistemas de ventas, juegos electrónicos, instrumentos avanzados, equipos biomédicos, control de vehículos, artefactos electrodomésticos, equipos militares etc.

Existen areas en las cuales el microprocesador ha penetrado con gran fuerza y en las que no solo ha reemplazado vieja tecnología sino que ha abierto nuevos campos en los que se trabaja intensamente en la actualidad.

En el campo de control de procesos, la implementación del control local en base a microprocesadores tiende a reemplazar el uso de grandes computadores centrales controlando en forma remota el proceso. El control local tiene grandes ventajas desde el punto de vista de operación, ya que el operador local puede controlar, supervisar y practicamente ver el efecto de sus acciones, con una probabilidad reducida de existencia de errores introducidos por la comunicación remota, como sería el caso del

control centralizado.

En el campo de la instrumentación, el microprocesador ha hecho posible el desarrollo de instrumentos "inteligentes" y paneles de instrumentos sofisticados. En una forma relativamente fácil y con un costo comparativamente pequeño, el microprocesador puede muestrear infinidades de switches en un panel, supervisar señales y manipularlas para generar otras señales, digitales o analógicas, y desplegar las mismas en medios complejos de visualización o en simples indicadores luminosos. Los modernos instrumentos que se están desarrollando, en algunos casos no solo permiten medir una determinada señal sino tambien procesar y almacenar la misma, con la finalidad, por ejemplo, de conocer su contenido armónico, su espectro de frecuencia, su porcentaje de distorsión, etc o para tomar una acción específica como activar una alarma, encender un indicador piloto, determinar en base a una cierta función otras medidas relacionadas con la señal muestreada.

1.2. MICROPROCESADORES Y MICROCOMPUTADORES

1.2.1. GENERALIDADES

Un computador digital se puede entender como un sistema que explora secuencialmente un medio de almacenamiento para extraer información codificada, la cual interpreta o decodifica, a fin de generar una serie de señales eléctricas que son usadas para iniciar acciones específicas asociadas con la información codificada.

El computador digital cuenta con una serie de componentes electrónicos agrupados en elementos funcionales con características definidas como son: LA UNIDAD DE PROCESO, LA UNIDAD DE CONTROL, CANALES DE ENTRADA SALIDA, y LA MEMORIA. Este cojunto de elementos y su interconexión son característicos de cada computador y constituyen lo que se denomina HARDWARE del equipo.

El hardware de un computador tiene la potencialidad de realizar una serie de actividades y operaciones, sin embargo, es la información codificada y almacenada en su memoria la que define y comanda las acciones del hardware; cada acción específica realizada por el hardware tiene asociado un cierto código que constituye lo que se denomina una INSTRUCCION; cada computador tiene un conjunto básico de estas instrucciones que lo definen programáticamente, este comjunto de instrucciones es lo que se denomina SOFTWARE básico del computador.

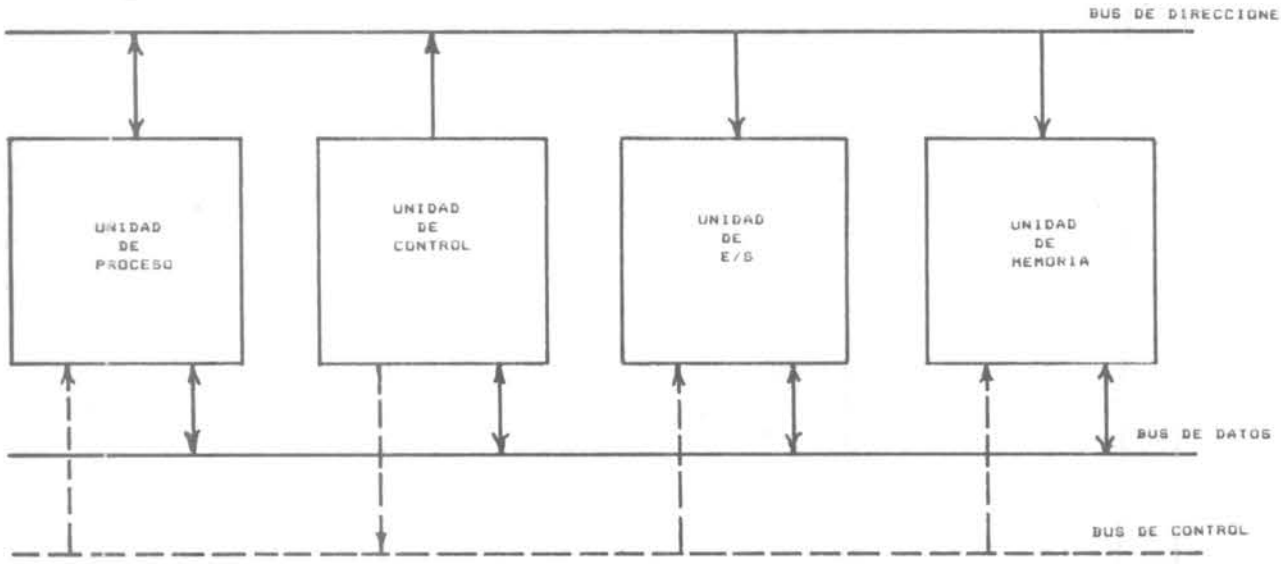


Figura 1-1. Diagrama funcional de un computador

Para conseguir resultados intelegibles y útiles con un computador, normalmente es necesario ejecutar mas de una de estas instrucciones básicas; al conjunto de instrucciones logicamente relacionadas y almacenadas secuencialmente en memoria se le conoce con el nombre de PROGRAMA.

Como el computador es un dispositivo digital, solo puede trabajar con dos niveles de voltaje; por ello, toda la información que el manipula (instrucciones, datos y direcciones) deben estar representados en base a un sistema binario. En el sistema binario solo existen dos posibles estados, cada uno de estos estados se representan con los dos únicos digitos del sistema numérico binario: el cero(0) y el uno(1). A los digitos binarios se les conoce con el nombre de BITS. En los computadores digitales los dos estados binarios normalmente representan dos niveles de voltaje distintos y bien definidos conocidos con los nombres de CERO LOGICO y UNO LOGICO.

Las diferentes instrucciones o datos dentro del computador se representan en el mismo como una combinación de bits (digitos binarios) de manera que una instrucción, consiste en una secuencia de ceros y unos. Este tipo de instrucción consistente en combinaciones de ceros y unos es el único lenguaje que el computador puede entender, por lo que se conoce con el nombre de LENGUAJE DE MAQUINA.

La manipulación por parte de los seres humanos del lenguaje de máquina es muy complicada, por lo cual se han desarrollado otros lenguajes de programación de computadores, que permiten escribir los programas en una forma fácil de entender por los humanos. Los programas escritos en un lenguaje distinto del de máquina no pueden ser ejecutados directamente por el

computador, es necesario convertir previamente las instrucciones escritas en un determinado lenguaje de programación en lenguaje de máquina. Este proceso de conversión es llevado a cabo por el mismo computador bajo control de programas especializados conocidos con el nombre de COMPILADORES. Un compilador no es mas que un programa cuya finalidad es convertir una serie de instrucciones, escritas en cualquier lenguaje de programación, en instrucciones en lenguaje de máquina, o sea como combinaciones de ceros y unos.

1.2.2. EL MICROPROCESADOR

Anteriormente se dijo que un computador digital basicamente constaba de cuatro elementos funcionales a saber : La unidad de proceso, la unidad de control, los canales de entrada/salida y la memoria.

La unidad de proceso es la parte del computador responsable de la manipulación de los datos que lleguen a ella provenientes de la memoria o de los canales de entrada/salida. Cuenta para ello con una Unidad Aritmética y Lógica (ALU), que le permite realizar operaciones aritméticas y lógicas con los datos. Y con unos registros que usa como medios intermedios o temporales de almacenamiento de información.

La unidad de control, cuya finalidad es la de mantener la secuencia de eventos requeridos para ejecutar una determinada instrucción, logrando que cada unidad funcional del computador haga la actividad que le corresponda en el momento que se requiera.

Los canales de entrada/salida son el medio usado por el computador para conectarse a dispositivos periféricos del mundo exterior.

La memoria es el sitio usado por el computador para almacenar tanto las instrucciones y datos introducidos a través de los dispositivos periféricos como los resultados obtenidos del procesamiento de esos datos.

En los computadores digitales típicos, las funciones de la unidad de proceso, la unidad de control, y los canales de entrada/salida, son realizadas por una serie de componentes electrónicos o chips interconectados externamente y diseminados en una o varias tarjetas de circuito impreso. Los avances en las técnicas de integración a gran escala(LSI), han permitido integrar en un solo componente o chip á la unidad de proceso, la unidad de control y los canales de entrada/salida; este chip, producto de avanzadas técnicas de integración de circuitos, es lo que comunmente se conoce con el nombre de MICROPROCESADOR. (ver figura 1-2).

1.2.3. CARACTERISTICAS BASICAS DE UN MICROPROCESADOR

Un microprocesador o CPU consiste de los siguientes elementos funcionales interconectados de alguna manera como puede verse en la figura 1-2

a) Registros
b) Unidad Aritmética y Lógica
c) Unidad de control
d) Puertos de entrada/salida
e) Buses

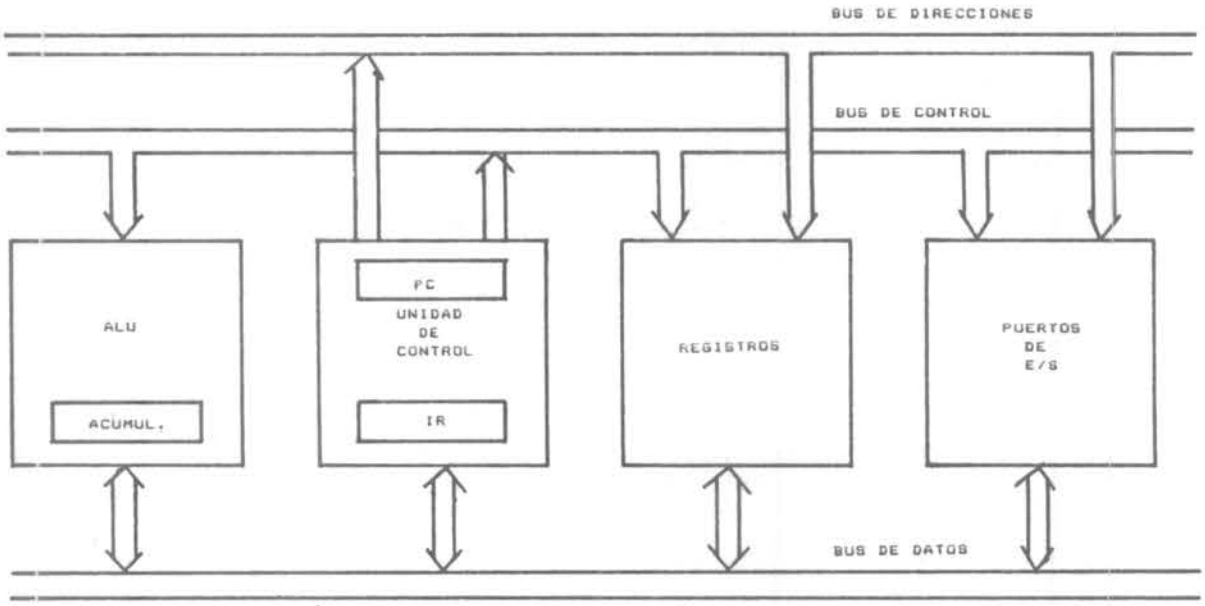


Figura 1-2 Diagrama de bloque elemental de un microprocesador

A continuación una descripción general de cada uno de estos bloques funcionales.

a) Registros

Los registros son usados en los microprocesadores como elementos de almacenamiento temporal. El tamaño de los registros se mide en bits y tipicamente encontramos registros de 4, 8, 16 y 32 bits.

En el argot de computadores y microprocesadores, al conjunto de 8 bits se le conoce con el nombre de BYTE y al

conjunto de 4 bits se le conoce con el nombre de NIBBLE.

Existen en el microprocesador algunos registros con propósitos específicos, entre estos registros se encuentra el ACUMULADOR, el CONTADOR DE PROGRAMA(PC) y el REGISTRO DE INSTRUCCION(IR).

El acumulador, es un registro usado generalmente para almacenar uno de los operandos a ser manipulados por la unidad aritmética y lógica. Algunos microprocesadores tienen mas de un acumulador lo cual les permite mejorar la eficiencia del procesamiento.

El contador de programa(PC), es un registro del microprocesador usado para direccionar la memoria. Las instrucciones y datos son almacenadas en la memoria en localizaciones identificadas con un número único que las diferencia una de las otras; este número, el cual identifica una posición de memoria es lo que constituye la dirección de esa posición. El contador de programa, siempre contiene la dirección de la próxima instrucción a ejecutar; la lógica del microprocesador es hecha de manera tal que el PC es incrementado en uno(1) cada vez que una instrucción es buscada en la memoria, de forma que siempre se mantendrá apuntando a la próxima instrucción o localización. Se desprende de esta lógica que las diferentes instrucciones que conforman un programa, deben ser almacenadas en posiciones adyacentes de memoria, de manera que la dirección mas baja contenga la primera instrucción del programa y así sucesivamente.

El registro de instrucción(IR), es otro de los registros especializados con los que cuenta un microprocesador, es usado para almacenar las instrucciones provenientes de la posición de memoria apuntada por el PC(una a una). El decodificador de instrucción de la unidad de control usa como entrada el contenido del IR para generar las señales que correspondan para dirigir todas las actividades relacionadas con la instrucción particular.

b) Unidad Aritmética y Lógica

El segundo gran bloque funcional del microprocesador lo constituye la ALU. Consta esta unidad de todos los circuitos necesarios para realizar operaciones aritméticas elementales tales como sumar y restar y en algunos microprocesadores multiplicar y dividir. Tambien está en capacidad de realizar operaciones lógicas elementales como AND, OR, XOR, rotaciones a la derecha y a la izquierda con el contenido del acumulador etc. Adicionalmente cuenta la ALU con: un registro temporal en el cual guarda, cuando así lo requiera la operación, un operando(el otro operando generalmente se guarda en el acumulador); la lógica de acarreo, overflow (un overflow ocurre cuando el resultado de una

operación aritmética excede la capacidad de la unidad o de los registros de la misma) y underflow (un underflow ocurre cuando el resultado de una operación aritmética es mas pequeño que el menor número que se puede representar en el microprocesador)

c) Unidad de Control

La unidad de control es para muchos el corazón del microprocesador. Como parte de la misma encontramos el contador de programa(PC), el registro de instrucción(IR), el decodificador de instrucciones(ID) y el secuenciador.

Es responsabilidad de la unidad de control hacer que se ejecuten las acciones necesarias en el momento adecuado y por el elemento que le corresponda a fin de conseguir que se cumplan las actividades que requiera la instrucción particular. Para lograr esto, la unidad de control toma como entrada la instrucción contenida en el registro de instrucción(IR), la cual fue leída de memoria en una primera fase, la decodifica o identifica por medio del decodificador de instrucción(ID), para que el secuenciador, usando esta información y una señal de reloj active las líneas de control requeridas según la instrucción particular.

d) Puertos de entrada/salida

Los puertos de entrada/salida tienen por finalidad permitir el intercambio de información entre el mundo exterior y el microprocesador. Existe en el micro un conjunto de terminales o pines para este propósito asi como tambien instrucciones especiales (llamadas de entrada/salida) que permiten que el micro lea y escriba datos en el puerto.

En general los puertos son controlados por la unidad de control y el bus de direcciones; los dispositivos periféricos se conectan a estos puertos a través de alguna interfase o algunaa veces directamente, dependiendo del dispositivo y del tipo de señal que el mismo genere.

e) Buses

Los distintos elementos funcionales que conforman el microprocesador estan interconectados entre si por un conjunto de líneas conocidos con el nombre de BUSES. En los microprocesadores se usan generalmente tres tipos de buses: el bus de datos, el bus de control y el bus de direcciones.

BUS DE DATOS

El bus de datos es usado por el microprocesador para transferir la información a ser leída o escrita en los puertos de entrada/salida y/o en la memoria. Esta información a ser transferida está codificada en base a un esquema binario y el número de bits usado para su codificación dependerá del número de líneas disponibles en el bus de datos. Este número de líneas o bits del bus de datos es lo que se llama ANCHO DE BUS, parámetro muy usado en la taxonomía de los microprocesadores puesto que en general tambien define EL ANCHO O TAMAñO DE LA PALABRA; tamaños típicos de ancho de bus en micros son 4 bits, 8 bits, 16 bits y ultimamente 32 bits.

En todo momento, los niveles lógicos de las líneas del bus de datos definen una palabra de datos específica; para el caso de un microprocesador con un bus de datos de 8 bits de ancho, como el mostrado en la figura 1-3, la palabra de datos está compuesta de 8 digitos binarios, D0 hasta D7. D0 es llamado el bit menos significativo(LSB) y D7 es llamado el bit mas significativo(MSB).

La palabra de datos en el microprocesador puede ser representada en distintas formas, ya se dijo que la mas usada es el sistema numérico binario. En este sistema los 8 bits de la palabra de datos (ver figura 1-3) se escriben como: 11010111_2; se le agrega el subindice 2 para indicar que esta en binario

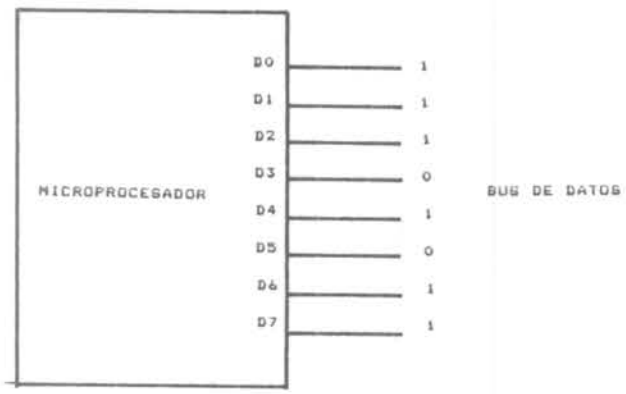


Figura 1-3. Microprocesador con bus de datos de 8 bits

UNIVERSIDAD DE CARABOBO

FACULTAD DE INGENIERIA

CURSO DE EDUCACION CONTINUA SOBRE

MICROPROCESADORES

Trabajo presentado ante el Ilustre Consejo de Facultad de Ingeniería por los Profesores Hyxia Villegas y Fredy Jara, para ascender a la categoría de Profesor Agregado.

ING. HYXIA VILLEGAS
ING. FREDY JARA

DICIEMBRE 1984

Como el sistema binario es un código de los llamados "pesados" cada bit según su posición tiene un peso, que en este caso es una potencia de 2; así el número anterior (11010111_2) equivale en decimal al número 343_{10} tal como se muestra a continuación

```
D7   D6   D5   D4   D3   D2   D1   D0

1    1    0    1    0    1    1    1   <--binario

 7    6    5    4    3    2    1    0
2    2    2    2    2    2    2    2   <--peso de los bits

256+ 64+  0+   16+  0+   4+   2+   1  =343
                                      10
```

El mismo patrón de datos mostrados en la figura 1-3 puede ser tambien representado como un número octal. En el sistema octal existen 8 digitos (0 al 7) y la representación octal de un patrón binario se consigue facilmente separando el paquete de digitos binarios que conforman la palabra, en grupos de tres, comenzando por el bit menos significativo, en este caso D0, como se muestra a continuación

```
  D7   D6  | D5   D4   D3  | D2   D1   D0

  1    1   | 0    1    0   | 1    1    1   <--binario

   1    0  |  2    1    0  |  2    1    0
  2    2   | 2    2    2   | 2    2    2   <--peso de los bits

  2+   1   | 0+   2+   0   | 4+   2+   1

    3      |      2        |      7       <--representación octal
```

327_8 es el equivalente octal del número binario 11010111_2, se coloca el subindice 8 para indicar que el número está representado en octal.

Una tercera forma de representar una palabra de datos es usando la representación hexadecimal; en ésta existen 16 digitos (0, 1, 2, 3, 4, 5, 6, 7, 8, 9, A, B, C, D, E, F). Las letras A, B, C, D, E, F representan los números 10, 11, 12, 13, 14 y 15 respectivamente.

El equivalente hexadecimal del patrón de datos de la figura 1-3 se puede conseguir separando el conjunto de bits de la palabra en paquetes de cuatro bits y asignandole a cada uno de estos paquetes el número hexadecimal que le corresponda tal como se muestra a continuación:

```
D7   D6   D5   D4  |  D3   D2   D1   D0
                   |
1    1    0    1   |  0    1    1    1  --- binario
                   |
 3    2    1    0  |   3    2    1    0
2    2    2    2   |  2    2    2    2  ---peso de los bits
                   |
8+   4+   0+   1   |  0+   4+   2+   1
                   |
        13         |          7
                   |
              D7H      <---  hexadecimal
```

El subindice H se le agrega para indicar que el número está en hexadecimal. Los diferentes tipos de códigos y sus usos pueden ser estudiados en detalle en los textos de lógica digital.

BUS DE CONTROL

Las distintas señales de control requeridas para la operación apropiada de los elementos que conforman el microprocesador y los elementos externos que se conecten al mismo son enviados y/o recibidas a través del bus de control; señales como "lea de memoria", "escriba en memoria", operaciones de la ALU(suma, resta, AND, OR, XOR, etc) y señales externas tales como reset, interrupción, y halt son líneas del bus de control y algunas de ellas permiten la sincronización de un dispositivo externo con el microprocesador. Tanto las líneas de control

internas como las líneas de control externas dependen en número de la arquitectura del microprocesador como se verá en capítulos posteriores.

BUS DE DIRECCIONES

El bus de direcciones es usado para transferir la dirección de los puertos de entrada/salida y/o de las diferentes localides de memoria.

En el caso de la memoria, se ha mencionado que la misma está constituida por localizaciones de igual tamaño identificadas cada una con un número que hemos llamado dirección. El tamaño de estas localizaciones de memoria viene dado en general por el ancho del bus de datos del microprocesador; Un microprocesador cuyo bus de datos sea de 8 bits requerirá que cada localización de memoria pueda albergar como máximo una palabra de 8 bits (1 byte). En el caso de microprocesadores con buses de datos de 4, 16 o 32 bits la organización de la memoria debe ser tal que cada localización tenga una capacidad máxima de 4, 16 y 32 bits respectivamente; El conjunto de localidades que el microprocesador puede direccionar se llama ESPACIO DE MEMORIA.

Al igual que en el caso del bus de datos una dirección la constituye una serie de ceros y unos en las líneas del bus de direcciones; en virtud de que cada línea del bus de direcciones solo podrá tener dos estados (cero o uno), el número máximo de combinaciones distintas para el caso de un bus de direcciones de N bits de ancho será de 2 elevado a la potencia N (2^N), constituyendo este el número máximo de localizaciones que el microprocesador puede direccionar.

En general se acostumbra a representar las direcciones en forma hexadecimal por ser esta una representación mas compacta y fácil de manipular, sin embargo, en las líneas de dirección solo existirán unos y ceros.

Un ejemplo que puede clarificar lo antes dicho es el que se muestra en la figura 1-4 en donde se ha representado un microprocesador cuyo bus de direcciones tiene un ancho de 16 bits identificadas desde A0 hasta A15. En este ejemplo la dirección binaria sobre este bus es 1110001111111111_2.

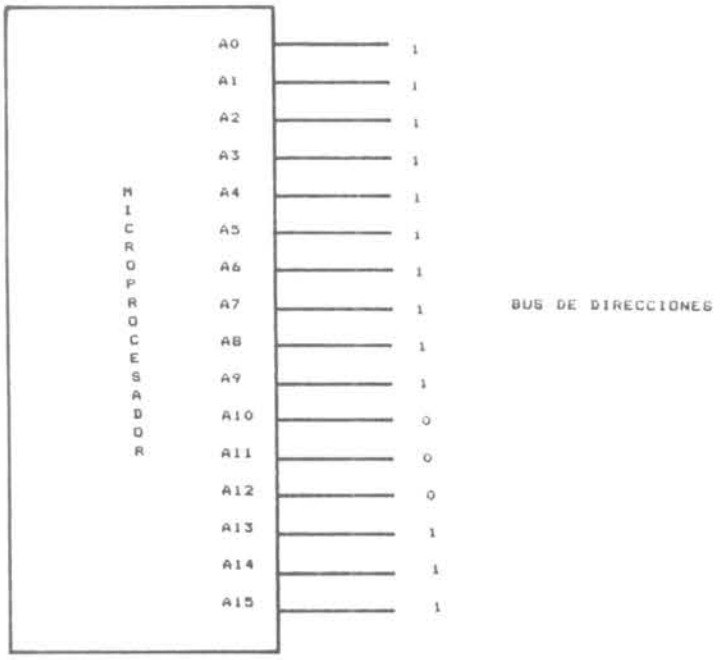


Figura 1-4. Microprocesador con un bus de direcciones de 16 bits

Usando el método descrito anteriormente se puede encontrar el equivalente hexadecimal de esa dirección tal como se muestra a continuación:

A15	A14	A13	A12	A11	A10	A9	A8	A7	A6	A5	A4	A3	A2	A1	A0
1	1	1	0	0	0	1	1	1	1	1	1	1	1	1	1
14				3				15				15			
E				3				F				F			

El número hexadecimal E3FF es mucho mas compacto que su equivalente binario y es por esta razon que la notación hexadecimal se usa muy frecuentemente para especificar las direcciones. Para un microprocesador con un bus de direcciones de 16 bits la localidad mas baja del espacio de memoria es la 0000_H y la mas alta la $FFFF_H$.

En algunos microprocesadores, el bus de direcciones y el bus de datos comparten las mismas líneas físicas, en estos

casos, el microprocesador suministrará alguna señal de control adicional a fin de definir en un momento dado que tipo de "dato" está presente en las líneas del bus: una dirección o una instrucción o dato.

1.2.4. OPERACION DE UN MICROPROCESADOR

El esquema de la figura 1-5 es un diagrama de bloque funcional simplificado de un microprocesador, el mismo será usado en esta sección para ilustrar la forma en la que un microprocesador opera para ejecutar un programa.

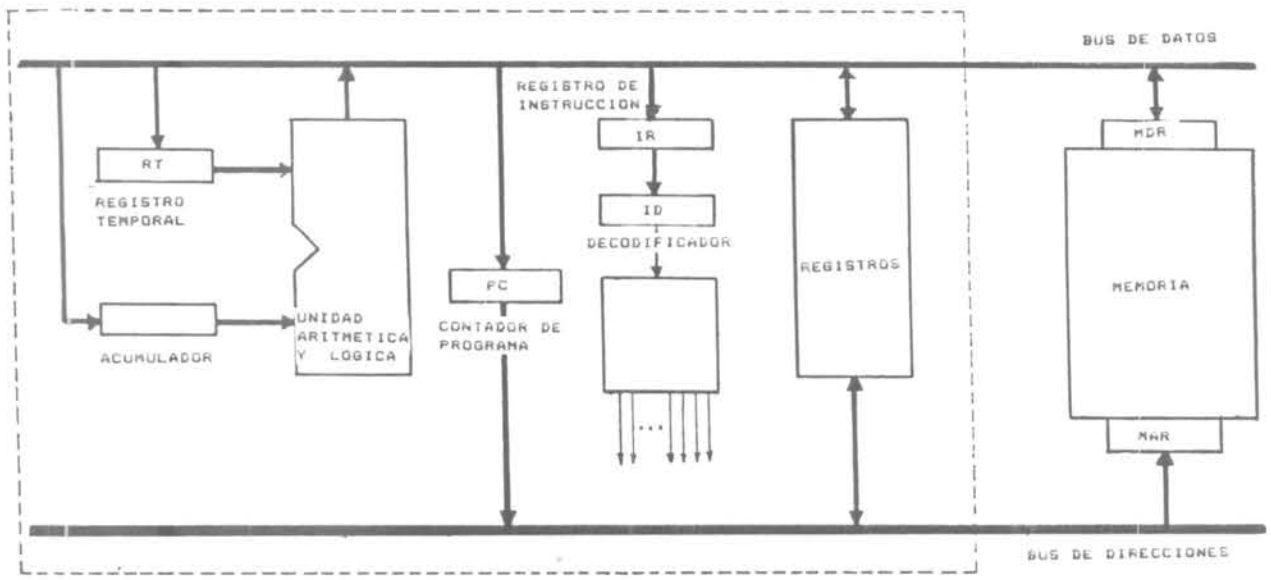


Figura 1-5. Diagrama de bloque simplificado de un micro

Como se dijo anteriormente un programa consiste de una secuencia ordenada de instrucciones almacenadas en memoria; las actividades del procesador en la ejecución de un programa son cíclicas repitiendo siempre la siguiente secuencia: una primera etapa, llamada FASE DE CAPTACION, en la cual el microprocesador busca en la memoria la instrucción a ejecutar y otra etapa llamada FASE DE EJECUCION, en la cual el microprocesador ejecuta la actividad o actividades que especifique dicha instrucción. La secuencia descrita se repite indefinidamente mientras el microprocesador esté operando.

FASE DE CAPTACION

En la fase de captación el contenido del contador de programa PC, el cual siempre apunta a la próxima instrucción a ejecutar, se vacía en el bus de direcciones y la lógica de control envía las señales de control a la memoria para indicarle que se va a realizar una operación de lectura. Esta dirección es cargada en el registro de dirección de memoria (MAR).

El contenido de la posición de memoria direccionada aparece en el registro de datos de memoria (MDR) y de allí pasa al bus de datos para ser luego cargada en el registro de instrucción IR. En este punto el contador de programa PC se incrementa en uno apuntando así a la próxima dirección que contendrá la siguiente instrucción del programa.

De esta forma queda completa la fase captación de la instrucción. Es de hacer notar que esta fase de captación o busqueda siempre tarda el mismo tiempo y ejecuta los mismos pasos los cuales son dirigidos por el secuenciador.

FASE DE EJECUCION

En la fase de ejecución el contenido del IR es decodificado por el decodificador y su resultado es pasado al secuenciador, el cual generará las señales que producirán una serie de eventos correspondientes a la ejecución de la instrucción. La fase de ejecución no tiene una duración determinada sino que depende del tipo de instrucción y del microprocesador, pudiendo existir instrucciones en un determinado microprocesador que requieran de uno, dos y hasta tres ciclos de máquina o sea accesos a memoria.

Con la finalidad de ilustrar la operación del microprocesador vamos a ejecutar un programa sencillo tal como lo es la suma de dos números, por ejemplo 5 y 4. Para ello supongamos lo siguiente:

a) Dentro del conjunto de instrucciones del microprocesador de la figura 1-5 , existen dos instrucciones MOV A, #dato y ADD A, #dato; la primera carga el acumulador con el dato especificado y la segunda permite sumar al acumulador el dato que se especifíca. Estas instrucciones son completadas en dos ciclos de máquina o sea se necesitan dos ciclos de busqueda a memoria para disponer de la instrucción completa. La tabla siguiente resume todo lo relacionado con estas dos instrucciones

MNEMONICO	CODIGO binario	hexa.	DESCRIPCION
MOV A, #dato	00100011 ?	23 ?	carga el acu. con el dato
ADD A, #dato	00000011 ?	03 ?	suma el dato al acu.

b)en la memoria, comenzando en la dirección (00) se cargó de alguna manera el programa para sumar los dos números (en este ejemplo 5 y 4) usando las instrucciones descritas anteriormente. Es de observar que el mnemónico es solo con fines ilustrativos ya que en memoria solo se cargan los códigos binarios, tal como se muestra en la siguiente esquematización de la memoria.

DIRECCION	CONTENIDO	MNEMONICO
00	00100011	MOV A,
01	00000101	5
02	00000011	ADD A,
03	00000100	4

A continuación la descripción detallada de como el microprocesador de la figura 1-5, ejecuta el programa que nos hemos propuesto:

1) El contenido del contador de programa (00H), que apunta la primera instrucción, es colocado en el bus de direcciones y eventualmente pasa al registro de direcciones de memoria (MAR) ver figura 1-6.

2) La lógica de control envía una señal a memoria para indicarle que se desea hacer una operación de lectura.

3) El contenido de la dirección de memoria apuntada por el MAR, que en este caso específico es la instrucción MOV A (23H), se vacía en el registro de datos de memoria MDR y de allí va al bus de datos.

4) El registro de instrucción IR, se carga con el contenido del bus de datos (23H).

5) El decodificador interpreta el código contenido en el IR, determina que la instrucción en el mismo es un MOV A y le pasa esta información al controlador de secuencia. El controlador de secuencia incrementa el PC en 1, terminando así la fase de busqueda de la primera instrucción, encontrando ahora el microprocesador internamente como se muestra en la figura 1-7.

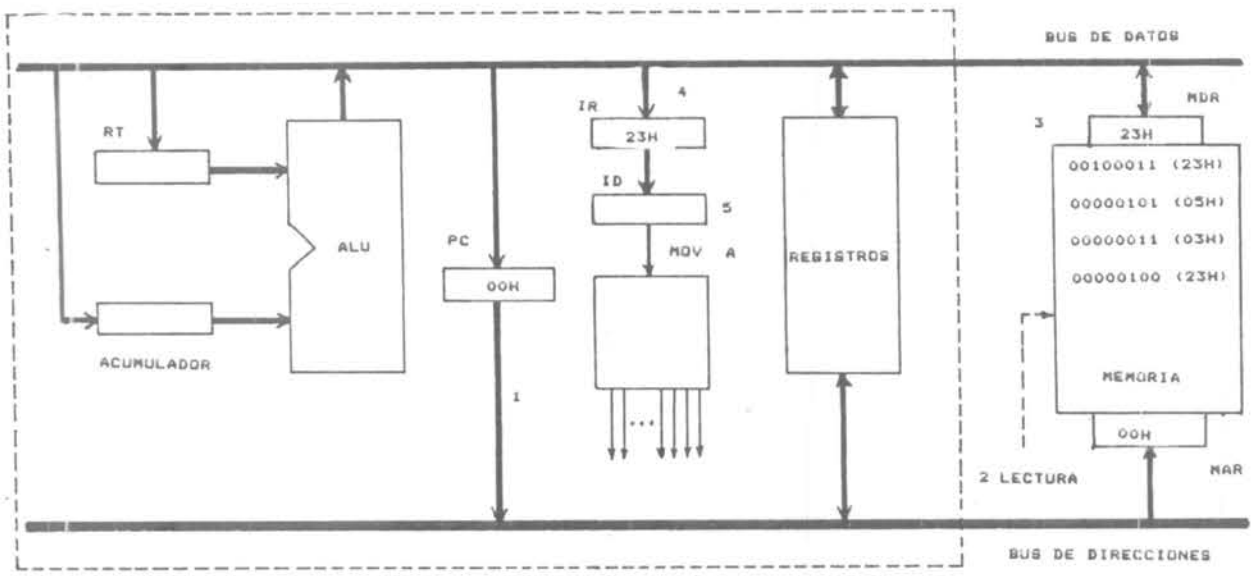


Figura 1-6. Fase de captación o busqueda de la primera instrucción

Como parte ya de la fase de ejecución de la instrucción MOV A, el controlador de secuencia hace que se ejecuten los siguientes pasos (ver figura 1-7)

6) Hace que se vacie el contenido del PC(01H) en el bus de direcciones y de allí pasa al MAR.

7) Envía a memoria una señal de control para indicarle que se desea realizar una operación de lectura, con lo cual el contenido la posición de memoria apuntada por el MAR(01H), que en este ejemplo corresponde al 05H, se vacía en el MDR.

8) A través del bus de datos el contenido del MDR(05H) se carga en el acumulador.

9) El PC se incrementa en 1, apuntando a la próxima instrucción y terminando asi la fase de ejecución de la primera instrucción de el programa.

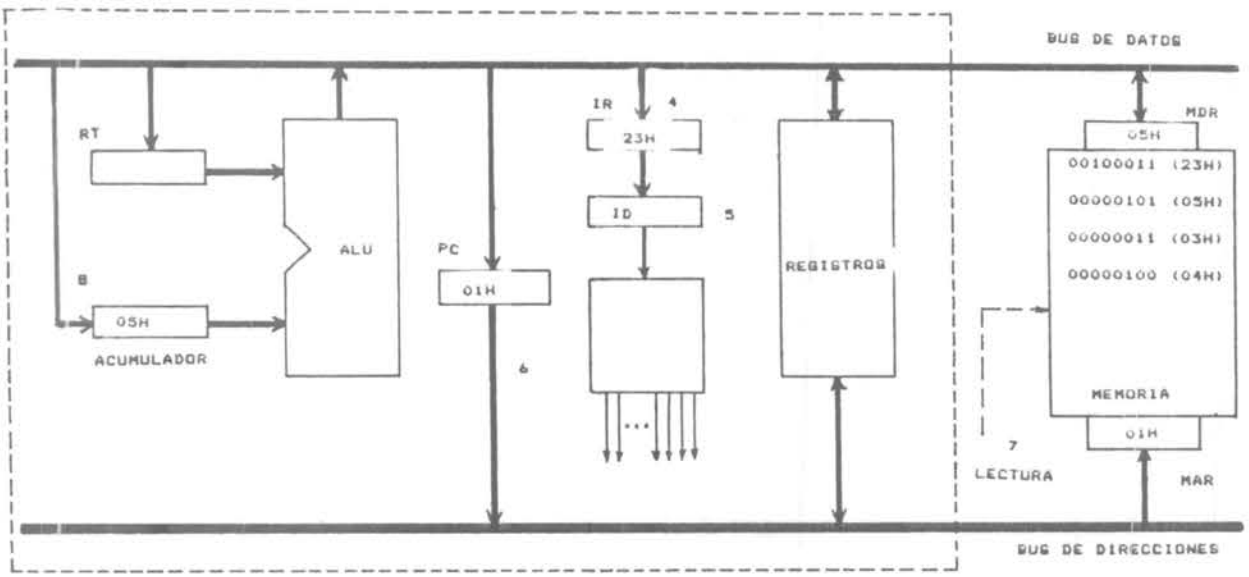


Figura 1-7. Fase de ejecución de la instrucción MOV A, dato

Entramos ahora en la fase de busqueda de la segunda instrucción del programa, para lo cual se ejecutan los siguientes pasos (ver figura 1-8)

10) El contenido del PC(02H), se vacía en el bus de direcciones y de allí pasa al MAR.

11) La lógica de control envía a memoria la señal de control para indicar que se va a hacer una operación de lectura.

12) El contenido de la dirección de memoria apuntada por el MAR, que en este caso corresponde a la instrucción ADD A (03H), se vacía en el MDR y de allí al bus de datos.

13) El IR, se carga con el contenido del bus de datos (03H).

14) El decodificador interpreta el código contenido en el IR, determina que la instrucción es un ADD A y le pasa esta información al controlador de secuencia. El controlador de secuencia incrementa PC en 1, terminando asi la fase de busqueda de la segunda instrucción que como se ve es idéntica a la de la primera instrucción.

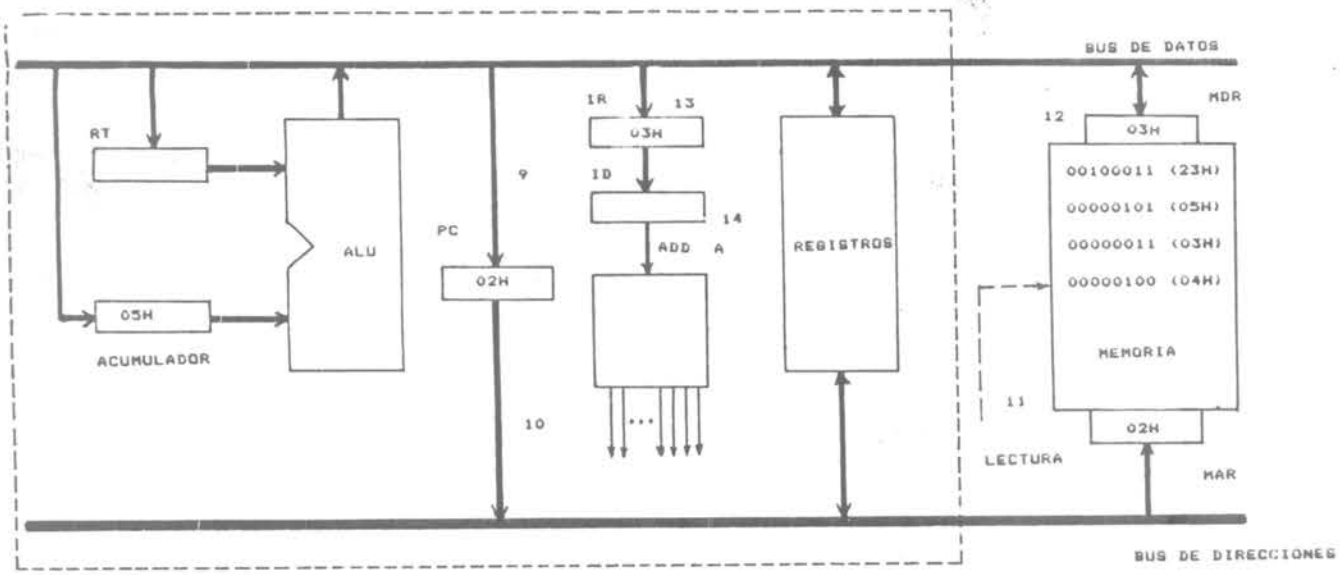


Figura 1-8. Fase de captación de la segunda instrucción

Entramos ahora a la fase de ejecución de la segunda instrucción de nuestro programa ejemplo. En la fase de ejecución de esta segunda instrucción, el controlador de secuencia hace que se ejecuten los siguientes pasos (ver figura 1-9)

15) Hace que se vacie el contenido del PC(03H), en el bus de direcciones y de allí pasa al MAR.

16) Envía a memoria la señal de control para indicarle que se va a realizar una operación de lectura, con lo cual el contehido de la posición direccionada por MAR, se vacía en el MDR(04H).

17) A través del bus de datos, el contenido del MDR(04H) se carga en el registro temporal de la ALU.

18) Envía a la ALU la señal correspondiente a la operación de suma (add).

19) La ALU, realiza la operación y coloca el resultado de la operación en el bus de datos.

20) Envía al acumulador la señal correspondiente para que se cargue con el contenido del bus de datos (09H) (ver figura 1-10).

21) Incrementa el PC en 1 con lo cual el microprocesador queda listo para ejecutar la siguiente instrucción.

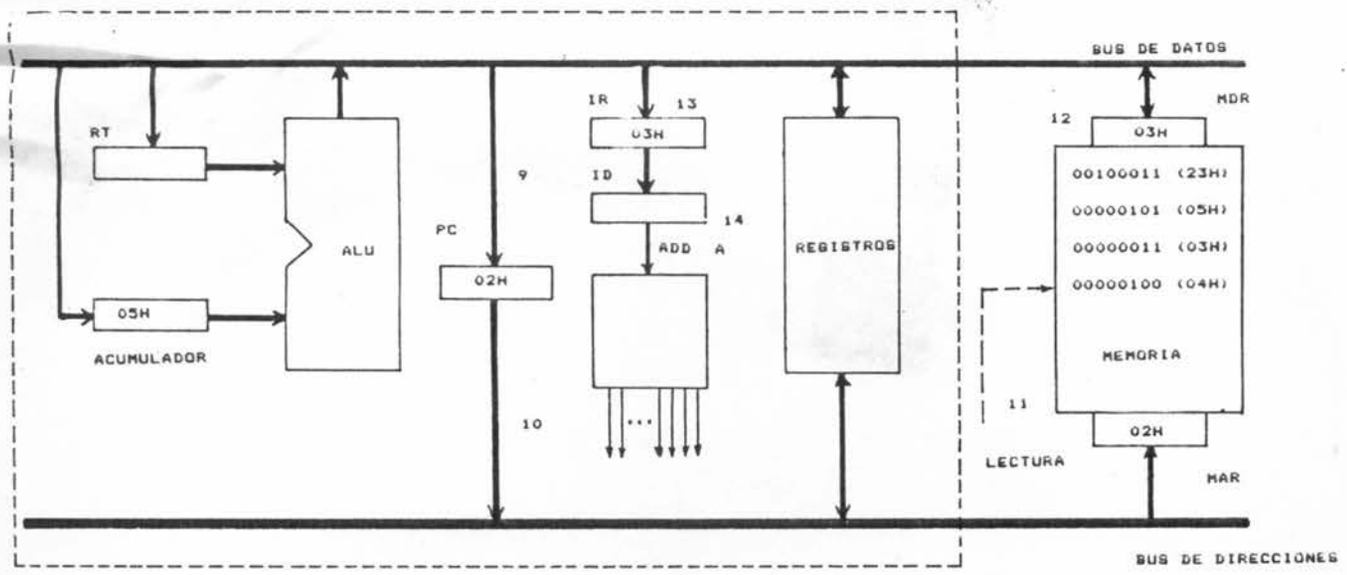


Figura 1-8. Fase de captación de la segunda instrucción

Entramos ahora a la fase de ejecución de la segunda instrucción de nuestro programa ejemplo. En la fase de ejecución de esta segunda instrucción, el controlador de secuencia hace que se ejecuten los siguientes pasos (ver figura 1-9)

15) Hace que se vacie el contenido del PC(03H), en el bus de direcciones y de allí pasa al MAR.

16) Envía a memoria la señal de control para indicarle que se va a realizar una operación de lectura, con lo cual el contehido de la posición direccionada por MAR, se vacía en el MDR(04H).

17) A través del bus de datos, el contenido del MDR(04H) se carga en el registro temporal de la ALU.

18) Envía a la ALU la señal correspondiente a la operación de suma (add).

19) La ALU, realiza la operación y coloca el resultado de la operación en el bus de datos.

20) Envía al acumulador la señal correspondiente para que se cargue con el contenido del bus de datos (09H) (ver figura 1-10).

21) Incrementa el PC en 1 con lo cual el microprocesador queda listo para ejecutar la siguiente instrucción.

1.2.5. MICROCOMPUTADORES

Un microcomputador, es un computador cuya unidad central de procesamiento (CPU) está constituida por un microprocesador. La figura 1-11 muestra un diagrama de bloques funcional de un microcomputador.

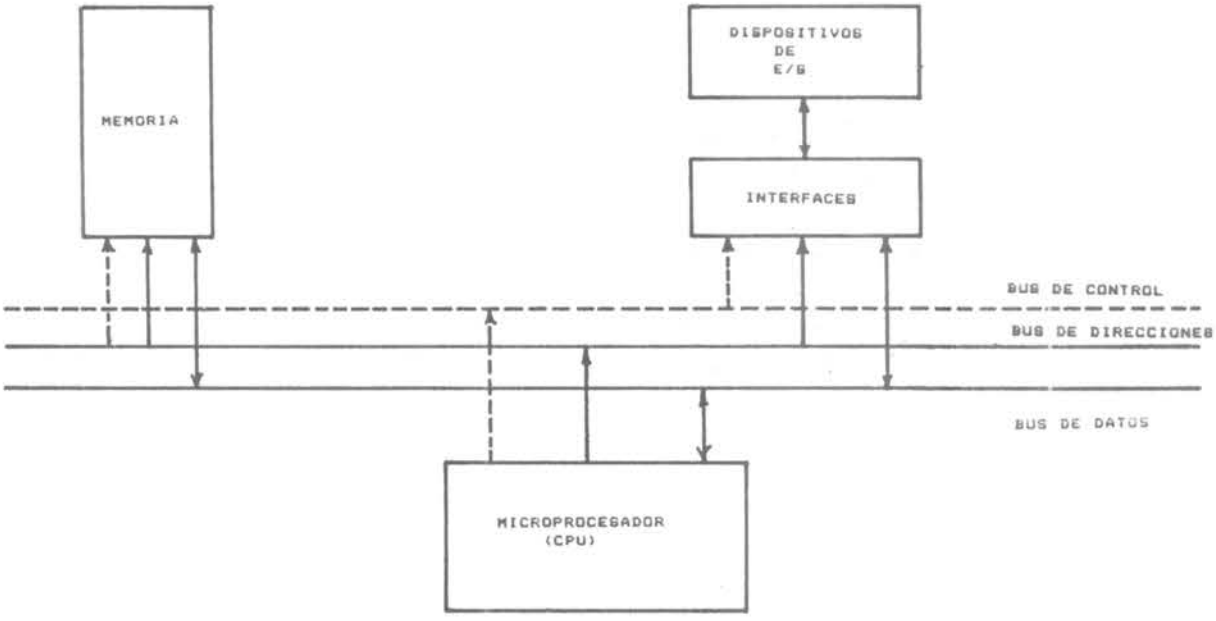


Figura 1-11. Diagrama de bloque de un microcomputador

El comportamiento y potencialidad de un microcomputador depende en gran parte de las potencialidades del microprocesador, o sea, de su Unidad Central de Procesamiento; sin embargo la visión que un usuario particular se forme del microcomputador depende tambien en gran medida del software que acompañe el equipo tales como: Sistema operativo, compiladores, editores, etc., que muestran al sistema en conjunto, con unas potencialidades mayores que las que la simple interconexión de la figura 1-11 sugiere.

1.3. INTERFASES CON EL MUNDO REAL

El microprocesador es capaz de realizar manipulaciones aritméticas y lógicas de datos representados en forma binaria; si se quiere aprovechar la potencialidad del mismo, se deberá dotar al sistema a micro de los elementos necesários que permitan que el microprocesador intercambie, en forma directa o indirecta, información con el mundo exterior.

Los elementos que permiten conectar el microprocesador con dispositivos externos para el intercambio de información entre el y el mundo interior se conocen con el nombre de interfases.

Una interfase debe cumplir funciones tales como: Convertir las señales analógicas en señales digitales y viceversa, convertir señales seriales en paralelas y viceversa, adaptar los niveles de voltaje y corriente entre el dispositivo externo y el micro a los niveles requeridos, resolver el problema de la diferencia de velocidad de operación entre los elementos a conectarse, proporcionar señales de tiempo y control para facilitar y permitir la comunicación entre el microprocesador y el dispositivo externo y en general, realizar o producir todas las acciones necesarias que hagan posible que un dispositivo externo y un microprocesador tengan puntos de convergencia que les permitan intercambiar información y aprovechar la misma en la forma en que la aplicación lo requiera.

Las interfases destinadas a la conversión de señales analógicas en digitales y viceversa se les ha denominado interfases analógicas y son elementos "obligatorios" en sistemas de control de procesos y adquisición de datos, siendo una aplicación típica la mostrada en la figura 1-12a y 1-12b.

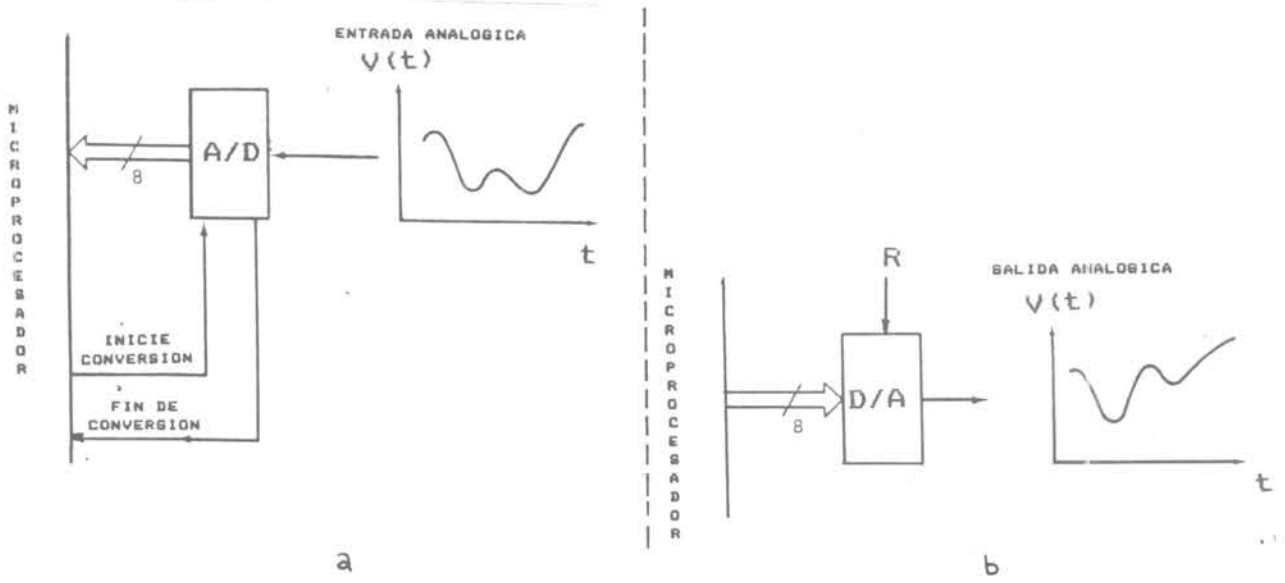


Figura 1-12. Conversión digital a analógica y analógica digital

En la categoría de interfases seriales se ubican aquellas interfases que permiten convertir información digital representada en forma paralela en serial y viceversa. Se han desarrollado elementos electrónicos de alta escala de integración que pueden cumplir esta función; estos dispositivos conocidos como UARTS (Universal Asynchronous Receiver Transmitters) y

USARTS (Universal Synchronous Asynchronous Receiver Transmitters) se describirán en forma mas detallada en capítulos posteriores. La figura 1-13 muestra una conexión típica de un UART con un microprocesador.

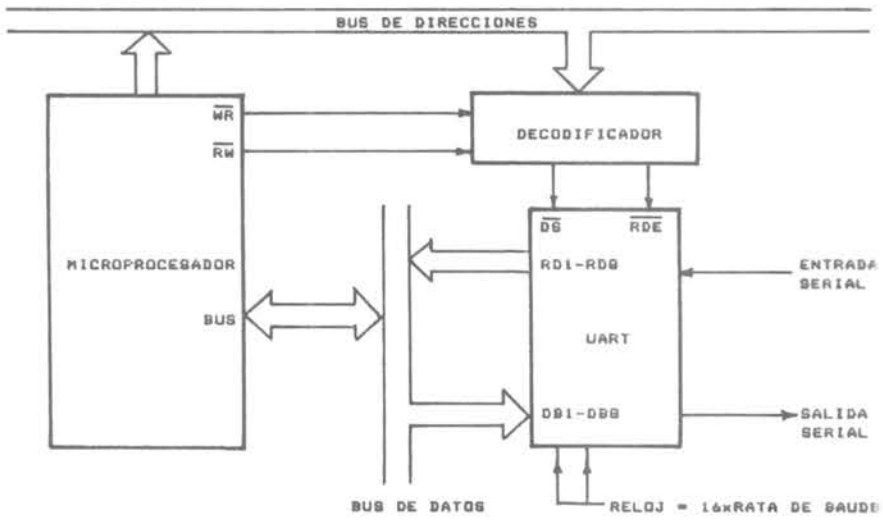


Figura 1-13. Conexión de un UART a un microprocesador

Existen adicionalmente unidades de interfase de propósito general, especialmente desarrolladas para el uso con microprocesadores, llamadas interfases programables, muy versátiles y entre las que destacan los PIA (Programmable Interface Adapter), los PPI (Peripheral Programmable Interface), Expansores de puertos etc., que serán descritos en forma detallada en capítulos posteriores.

Con la finalidad de facilitar la conexión de equipos o elementos al microprocesador se han adoptado una serie de reglas y normas las cuales se han agrupado en paquetes en base a su aplicación, y se han establecidos como estandares que la mayoría de los fabricantes de los sistemas a microprocesadores cumplen a fin de conseguir la compatibilidad entre equipos de diferentes compañias. Entre estas interfases se encuentran muy popularizadas los estandares para configuraciones de buses como son el S-100 y el IEEE-488 usado el primero en la conexión de los diferentes elementos que conforman un sistema a microprocesador y el segundo usado en sistemas a microprocesadores para el control y supervisión de instrumentos.

Tambien se han desarrollado estandares para normalizar las transmisiones seriales entre diferentes equipos bien sean micros, terminales o cualquier otro elemento capaz de recibir y transmitir información serialmente, siendo la mas popular y ampliamente usada el estandard adoptado por la asociación de industrias eléctricas y electrónicas conocida como RS-232 y la cual en conjunto con la S-100 y la IEEE 488 se describirán en forma detallada en capítulos posteriores.

C A P I T U L O I I

2.1. ARQUITECTURA DE UN COMPUTADOR

Este capítulo tiene por finalidad explicar algunos conceptos básicos de arquitectura de computadores y en especial el estudio de la arquitectura del microprocesador 8035 de INTEL, con el cual se realizaran todos los experimentos y ejercicios del presente trabajo.

Se ha mencionado en el capítulo I que en todos los computadores encontraremos basicamente los mismos elementos funcionales tales como: Registros, Unidad Aritmética y Lógica, Unidad de control, Puertos de Entrada/Salida y Memoria; sin embargo, la forman en que estos elementos se organizan e interconectan entre si son diferentes en los distintos equipos. Esta forma de organizar e interconectar los diferentes elementos que constituyen el computador se conoce como ARQUITECTURA DE UN COMPUTADOR.

El microprocesador, como parte integrante de los modernos computadores y como unidad básica de procesamiento en los microcomputadores, tiene tambien una organización característica cuyo estudio será de gran utilidad tanto para diseñadores como usuarios, pues la organización o arquitectura del microprocesador será determinante en la organización e interconexión de los diferentes elementos constituyentes de los sistemas de aplicación y en el comportamiento global de los mismos.

El aprovechamiento eficiente de las potencialidades de un microprocesador, la determinación de sus limitaciones y las ventajas o desventajas de su uso en una aplicación determinada es posible a la luz del conocimiento de su organización o arquitectura.

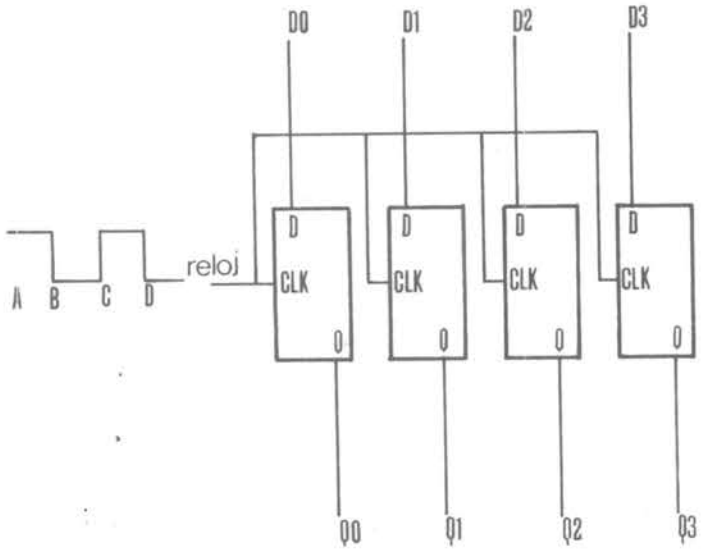
Antes de entrar a describir la arquitectura propiamente dicha del microprocesador 8035, se va a profundizar en algunos de los tópicos referentes a arquitectura que fueron mencionados en el capítulo I que se considera contribuiran a formar una mejor base en el lector que le permitirá aprovechar y entender mejor las explicaciones referentes a la organización del 8035 y en

general de la arquitectura de cualquier computador o microcomputador.

2.1.1. REGISTROS

Los registros, como se ha mencionado anteriormente, son parte fundamental de los computadores y microprocesadores. Las siguientes líneas tienen por finalidad dar al lector una idea básica de la implementación de un registro y de la forma en que dicho registro pudiera conectarse a un bus.

Un registro es basicamente un conjunto de flips-flops tipo D, sincronizados por medio de un reloj comun tal como se muestra en la figura 2-1. La idea básica de ese circuito consiste en que la información presente en las entradas (D) del flip-flop solo pasaran a sus salidas (Q) cuando ocurra un pulso de reloj. Las salidas conservaran sus valores independientemente de los cambios en las entradas mientras no ocurra un pulso de reloj.



INSTANTE	D0	D1	D2	D3	Q0	Q1	Q2	Q3
A	0	0	0	0	0	0	0	0
B	1	0	1	0	0	0	0	0
C	1	0	1	0	1	0	1	0
D	1	1	1	0	1	0	1	0

Figura 2-1. Implementación de un registro de 4 bits con flips flops tipo D

La operación del circuito de la figura 2-1 se resume en la tabla al lado derecho de esa misma figura. En el instante A, las entradas (D_0 D_1 D_2 D_3) y las salidas (Q_0 Q_1 Q_2 Q_3) tienen el mismo patron de datos (0000); en el instante B en las entradas se presenta un nuevo patron de valores (1010) sin embargo, las salidas todavía conservan su patron original; en el

instante C, se produce un pulso de reloj con lo cual, los flips flops ponen en sus salidas el patron de datos presente en sus entradas.

En la figura 2-2 se muestra una representación simbólica de un registro como el de la figura 2-1 y la forma en que dicho registro se esquematizaría en el instante C del ejemplo anterior.

Figura 2-2. Representación de un registro de 4 bits

En el capítulo I cuando se presentó un diagrama de bloque general de un microprocesador (figura 1-5, capítulo I) se observa que los distintos registros que conforman al microprocesador estan conectados al bus (tanto sus entradas como sus salidas). Parece contradictorio pensar en un bus que tan pronto maneja las entradas D, como tan pronto es manejado por las salidas Q de dichos registros, sumandose a esto el hecho de que las varias salidas o entradas de los distintos registros esten conectadas a unas mismas líneas.

En la practica lo que sucede es que se usa una forma especial de conexión de todos los elementos que comparten un bus determinado, realizandose esta conexión a traves de compuertas lógicas del tipo tres estados (tristate) cuya operación describiremos brevemente a continuación.

La figura 2-3 representa una compuerta "tristate" y en la parte derecha de la misma figura se encuentra la tabla de verdad que determina su forma de operación.

HABILITACION	ENTRADA	SALIDA
0	X	ALTA IMPEDANCIA
1	0	0
1	1	1

Figura 2-3. Compuerta de tres estados y su tabla de verdad

La operación de la compuerta es como sigue: Cuando en la entrada denominada de habilitación se encuentra un cero lógico, el valor presente en las entradas de la compuerta no se refleja en la salida sino que la misma permanece en un estado o condición especial llamada tercer estado o "tristate", en el cual presenta una alta impedancia apareciendo para los efectos prácticos como un circuito abierto; en el caso de que la señal de habilitación sea un uno lógico la compuerta funciona como cualquier compuerta normal de ese tipo, colocando en su salida el valor o dato presente en su entrada.

En la figura 2-4 se muestra un posible esquema de conexión de dos registros al bus de datos de un microprocesador; su operación es como sigue:

El encargado de enviar la señal de habilitación (E1,E2) a uno u otro registro es la unidad de control a traves del bus de control. Por otra parte, las entradas D de los registros seran habilitadas tambien por la unidad de control a traves del mismo bus, enviando una señal de "selección de chip" (CS1,CS2). Cuando una de las señales CS1 o CS2 se activa (en este caso se activa cuando tienen el valor uno lógico), el pulso de reloj pasará a traves de la compuerta AND correspondiente y llegará hasta la entrada de reloj de los flips flops que constituyen el registro respectivo, haciendo que las entradas a ese registro aparezcan en sus salidas. Cuando las señales de selección CS1 o CS2 tienen el valor cero lógico, el registro tiene en su entrada de reloj siempre un cero por lo que sus salidas permaneceran inalterables, independientemente de los cambios en sus entradas.

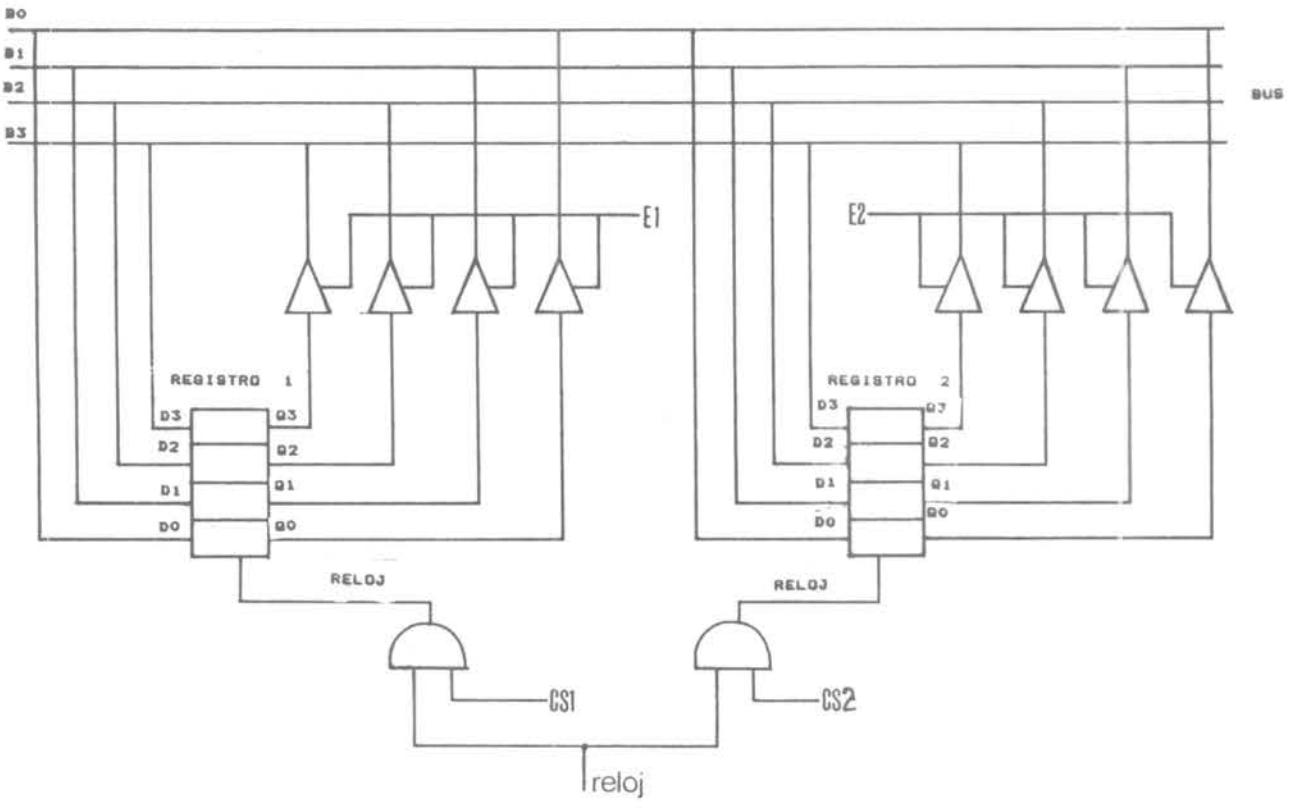


Figura 2-4. Conexión de dos registros de 4 bits a un bus

Como se ve, es la unidad de control quien decide mediante las señales de control CS1, CS2, E1 Y E2 que registro va a tener sus entradas D activas y que registro tendrá sus salidas Q conectadas al bus. Tambien es necesario recalcar que ningun registro del microprocesador tiene conectadas sus salidas o entradas al bus de control; este bus esta conformado por una serie de líneas de control que envía la unidad de control a todas las unidades que conforman el microprocesador.

2.1.2. MEMORIA

En el capítulo I se mencionó que sobre la memoria, el microprocesador, podía realizar basicamente dos operaciones: "leer de memoria" (memory read), "escribir en memoria" (memory

write). Sin embargo, en esta parte veremos que existen varios tipos de memorias con características particulares, algunas de las cuales no permiten que sobre ellas se realicen las dos operaciones básicas anteriormente descritas.

Existen basicamente dos tipos de memoria que se usan con microprocesadores: MEMORIAS VOLATILES y MEMORIAS NO VOLATILES.

La memoria VOLATIL tiene como característica fundamental el hecho de que la información en ella almacenada se pierde si la fuente de energía eléctrica que la alimenta se remueve. La memoria NO VOLATIL tiene como característica fundamental que la información en ella almacenada no se pierde si se le remueve la fuente que la alimenta.

Dentro de la categoría de memorias no volatiles encontramos las memorias conocidas como ROM (Read Only Memory) las cuales tienen la particularidad de que la información almacenada en ellas solo puede ser leída, en otras palabras, no se puede hacer sobre ellas operaciones de escritura.

Existen varios tipos de memorias ROM; algunas de estas memorias son programadas en la fábrica en el momento en que se fabrican y su contenido no puede ser cambiado por ningun procedimiento.

Otro tipo de memoria ROM es aquella que puede ser programada por el usuario. Dentro de esta categoría encontramos: las llamadas PROM, las cuales pueden ser programadas (escritas) por el usuario, pero una vez que en ellas se ha escrito algo no es posible bajo ninguna forma modificarlo; y las llamadas EPROM las cuales pueden ser programadas por el usuario mediante un procedimiento y equipo especial y de la misma manera mediante un procedimiento especial el contenido de dichas memorias puede ser "borrado" para su posterior reprogramación.

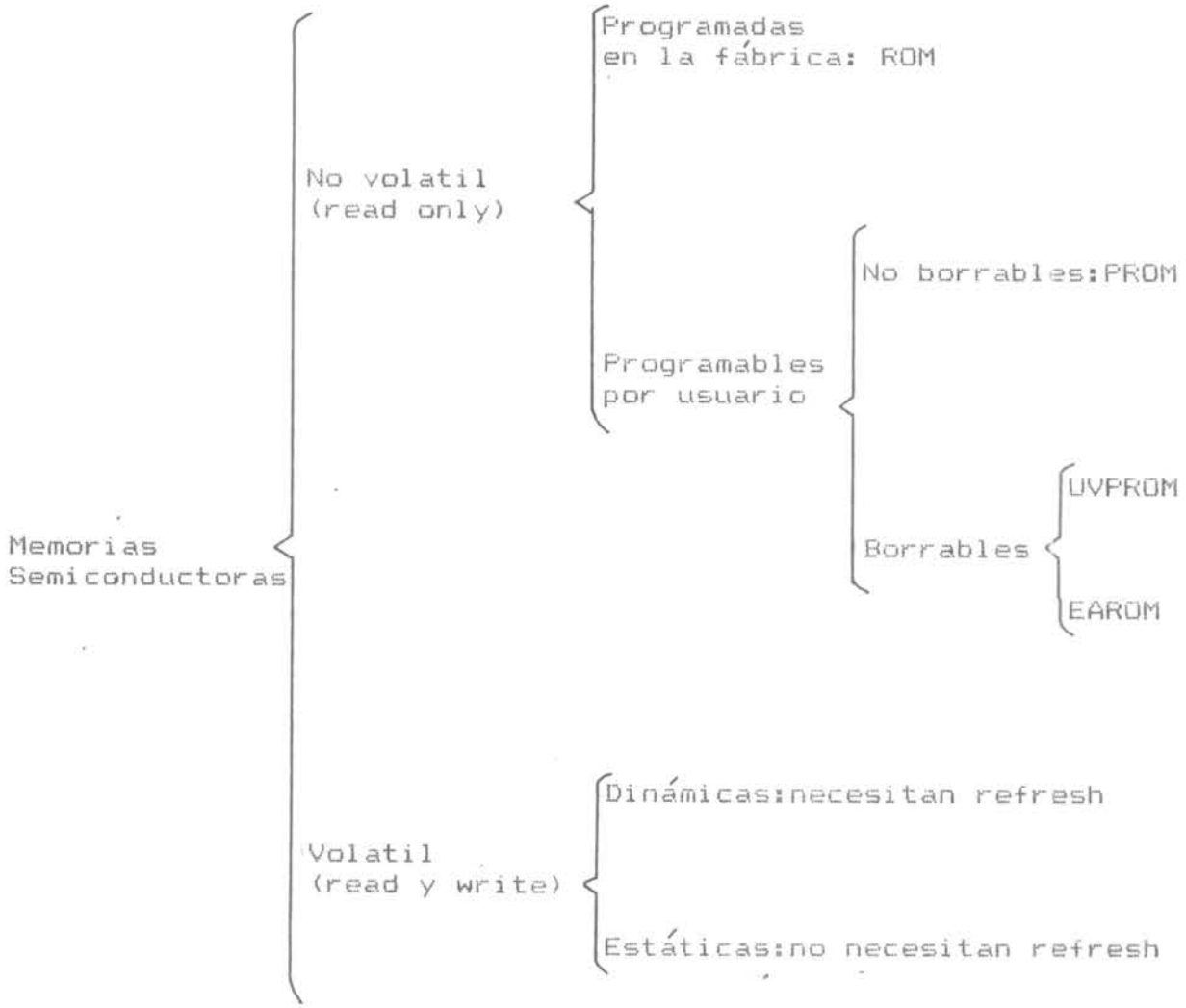
Es de hacer notar que normalmente sobre estas memorias el computador solo puede hacer operaciones de lectura y que es solo con un dispositivo especial y siguiendo el procedimiento adecuado para cada tipo de memoria que se podrá escribir sobre estas últimas memorias.

En la categoria de memorias volátiles encontramos las llamadas memorias RAM (Ramdon Access Memory) sobre las cuales es posible realizar las dos operaciones básicas.

Existen dos tipos de memoria RAM: Las llamadas estáticas y las dinámicas. Las memorias RAM dinámicas tienen la particularidad de que la información almacenada en ellas se pierde despues de transcurrido un cierto tiempo, aun cuando no se interrumpa su fuente de alimentación, lo cual obliga a que mediante un circuito externo periodicamente se este "repasando" su contenido, a fin de no perder la información almacenada, esta tecnica es conocida como REFRESH MEMORY.

La memoria RAM estática no necesita de ningun circuito externo para refrescar su contenido ya que su construción esta hecha teniendo como unidad básica de almacenamiento elementos semejantes a los de los registros antes descritos.

La tabla siguiente resume todo lo referente a los diferentes tipos de memorias semiconductoras que acabamos de describir brevemente.



La otra consideración que debe ser tomada en cuenta en

cuanto a la operación de la memoria es lo referente a su tiempo de respuesta. En el capítulo I, cuando se corrió un pequeño programa con el fin de ilustrar la operación de un microcomputador y se realizaban las operaciones de lectura de memoria, se colocaba en el registro MAR de la memoria la dirección, y el contenido de la misma aparecía en forma "instantanea" en el MDR, una vez que la memoria recibía el comando de lectura.

En la practica esto no es realmente cierto ya que en primer lugar es necesario dejar que transcurra un cierto tiempo t_1 (ver figura 2-5), necesario para que se estabilizen las direcciones en el bus antes de enviar la señal de "lectura" y en segundo lugar, a partir del momento en que se envía dicha señal, transcurrirá un cierto tiempo, que se ha llamado t_2 en la figura 2-5, antes de que el contenido de la posición direccionada por el MAR aparezca en el MDR.

El tiempo t_2, es el tiempo que transcurre entre el momento en que la memoria recibe la orden de lectura y el momento en que el contenido de la posición direccionada aparece en forma estable en el MDR y se conoce con el nombre de TIEMPO DE ACCESO A MEMORIA.

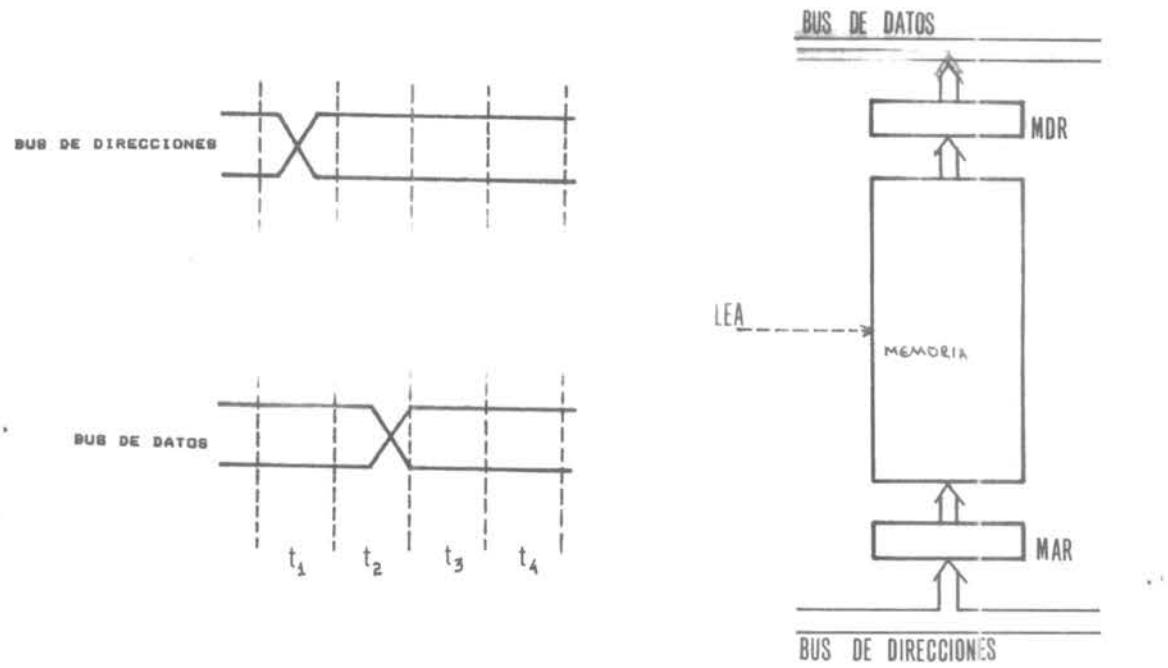


Figura 2-5. Operación de lectura de memoria con su diagrama de tiempo

En la figura 2-5 se muestra en forma detallada como ocurre una operación de lectura en una memoria y las consideraciones de tiempo a ser tomadas en cuenta cuando se realize dicha operación. En forma resumida la secuencia de eventos que ocurren en una operación de lectura es la siguiente:

En el bus de direcciones se coloca la dirección de la posición de memoria que se quiere leer y se espera por un tiempo t_1 a que esta dirección esté estabilizada antes de enviar la señal que ordena la operación de lectura; despues que se ha dado el comando de lectura, el microproprocesador debe esperar como mínimo un tiempo t_2 (tiempo de acceso a memoria) para leer el contenido del MDR, que será el contenido de la posición de memoria direccionada; una vez que los datos han pasado al bus de datos y de allí a cualquier otro registro del microprocesador, la señal de lectura es desactivada, con lo cual el MDR pasa al estado de alta impedancia "desconectandose" del bus de datos (t_4).

2.1.3. UNIDAD DE CONTROL

La unidad de control, como se mencionó anteriormente, es la encargada de generar las señales de control requeridas para la operación apropiada del resto de las unidades y para controlar la secuencia, tiempo y dirección de todos los datos que fluyen dentro del microprocesador y/o computador.

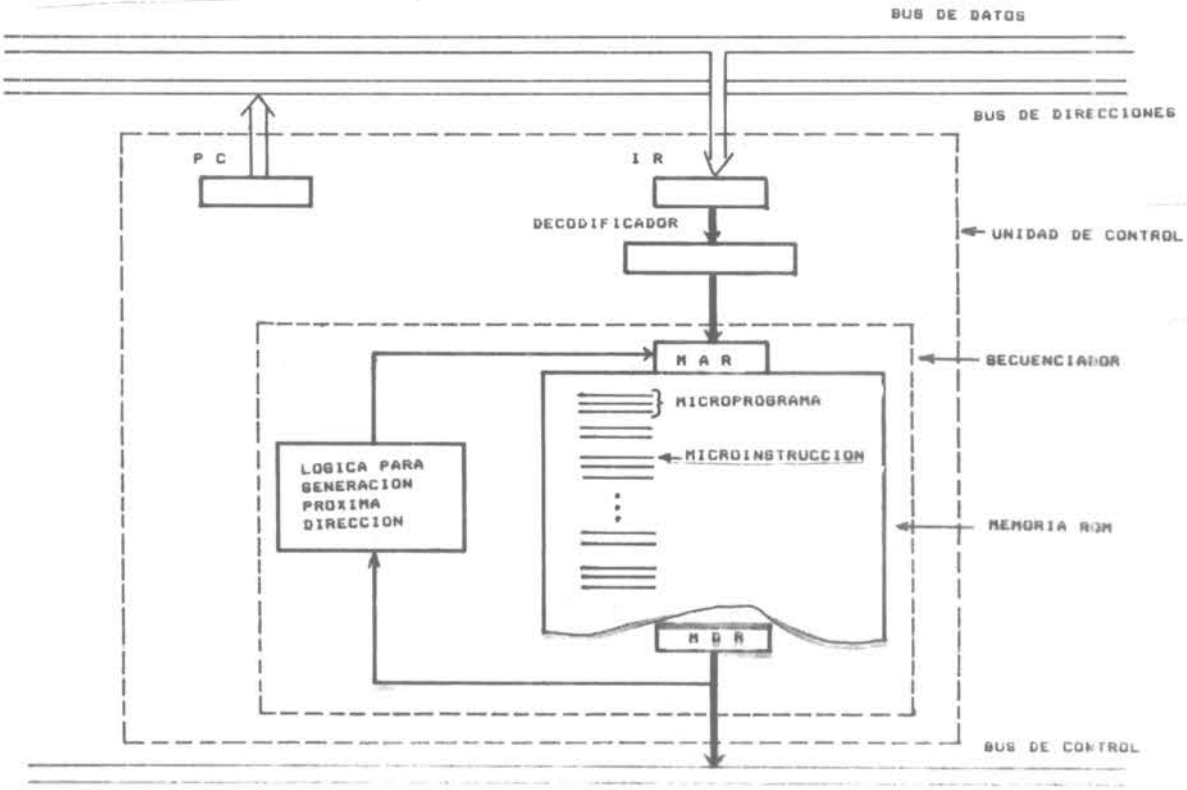
Posee la unidad de control el registro de instrucción IR, el contador de programa PC, el decodificador de instrucciones ID y el secuenciador.

El registro de instrucción, como se dijo anteriormente, es la entrada al decodificador, el cual, se encarga de decodificar o interpretar el código de operación de la instrucción presente en el IR y pasar esta información al secuenciador

El secuenciador está constituido por una memoria ROM, interna al microprocesador, en la cual, se almacenan las secuencias de señales de control que deberan enviarse para que se ejecute una instrucción.

En cada una de las direcciones de la ROM se almacenan patrones de bits, cada uno de los cuales corresponde a las operaciones mas elementales que el microprocesador puede realizar. El código almacenado en una localidad dentro de esta ROM interna se llama MICROINSTRUCCIÓN. Un conjunto de estas microinstrucciones vienen a constituir lo que se conoce con el nombre de MICROPROGRAMA; un microprograma tiene por finalidad realizar una instrucción del microprocesador. Desde este punto de vista, la unidad de control opera en forma similar a el

microprocesador como un todo, constituyendo una especie de pequeño computador dentro de otro computador mayor. La figura 2-6 muestra un diagrama funcional simplificado de una unidad de. control de un microprocesador



ama de la unidad de control
procesador

La salida del decodificador viene a constituir la dirección de la memoria ROM donde esta almacenada la primera microinstrucción del microprograma que corresponda a la instrucción que está en el IR. El registro de salida de la ROM osea el MDR, se carga una a una con cada microinstrucción hasta que el microprograma es terminado; todos los microprogramas terminan con una microinstrucción que hace que el secuenciador inicie la ejecución, en forma automática, de un microprograma especial, cuya finalidad es hacer que se lleve a cabo la fase de captación de la proxima instrucción del programa que se está ejecutando.

Cada microprocesador tendra almacenado en la memoria ROM de su unidad de control, un conjunto de microprogramas los cuales corresponden al conjunto de instrucciones de la máquina.

Estos microprogramas son grabados en el momento de su fabricación y generalmente no pueden ser cambiados por ningún procedimiento.

Es de hacer notar que el grado mas bajo de programación esta representado por este nivel, en donde las microinstrucciones son señales eléctricas directas que habilitan y deshabilitan compuertas, flip flops, registros, etc.. Este nivel de programación por estar a mitad de camino entre el software, que es programación pura, y el hardware, que lo constituyen los circuitos electrónicos y las señales eléctricas, se le conoce con el nombre de FIRMWARE.

2.1.4. UNIDAD DE PROCESO

La unidad de proceso es la encargada de realizar las operaciones aritméticas y lógicas asi como las de transferencia entre acumulador, registro auxiliar, registros de proposito general, flags. etc. Cuenta para ello la unidad de proceso con los siguientes elementos funcionales:

Unidad Aritmética y Lógica(ALU)
Acumulador (A)
Registro Auxiliar (AUX)
Registros de proposito general
Flags o Banderas

Los resultados de las operaciones realizadas por la ALU se almacenan generalmente en el acumulador destruyendo lo que previamente estuviese allí. La unidad de control es la que comanda a la ALU a ejecutar una u otra operación; ella ademas es la encargada de controlar que los resultados producidos por la ALU pasen al bus de datos, al acumulador o a cualquier otro registro.

El resultado de las operaciones de la ALU ademas de afectar al acumulador o a otro registro de almacenamiento, afecta tambien unos registros de un bit que sirven de chequeo de la operación realizada; Estos registros, presentes en la mayoría de los microprocesadores son:

El registro de OVERFLOW el cual se activa si el resultado de la operación realizada por la ALU sobrepasa la capacidad del acumulador; asi, si el acumulador es de 8 bits y el resultado es mayor que 255_{10} (FF_H) el se activara indicando que hubo overflow.

Otro de estos registros es el registro Z el cual, cuando se activa, indica que el resultado de la operación ejecutada por la ALU fue cero.

El registro N, el cual se activa si el resultado de la operación ejecutada por la ALU es negativo.

El registro C de acarreo, el cual se activa en caso de suma y el resultado de mayor que el tamaño del registro acumulador o en el caso de una resta se quite prestado a un digito mas alla de la capacidad del acumulador. En el caso de un acumulador de 8 bits se activará el registro de acarreo C si se le tiene que quitar prestado una unidad al noveno digito.

Un último registro que generalmente encontramos en la mayoría de los microprocesadores es el llamado de Acarreo Auxiliar. Este registro tiene un uso específico en la detección de acarreo cuando se realizan operaciones con números codificados en BCD (números decimales codificados en binario). En esta forma de codificación binaria, cada digito decimal se sustituye por su equivalente binario tal como se aprecia en los siguientes ejemplos:

Como se puede observar, cada digito decimal se sustituye por 4 bits que constituyen su equivalente binario; puesto que solo existen 10 digitos decimales (0 al 9), solo existiran 10 combinaciones validas en BCD de las 16 posibles que se pueden representar con 4 bits y que corresponderan a los digitos del 0 al 9.

Cuando se realizan operaciones aritméticas con números codificados en BCD, existe la posibilidad de que el resultado sea mayor de nueve en un detrminado digito o de que haya acarreo, en cuyo caso, el resultado no quedaría codificado en BCD (por definición); para estos casos especiales, en los cuales se opere con números codificados en BCD, se usa el registro de acarreo auxiliar, el cual se activará si ocurre una de las condiciones antes mencionadas.

Estos registros de un bit estan formados por un sólo flip flop; sin embargo, ellos suelen ser agrupados para formar una palabra que en general refleja el estatus del programa por lo cual se acostumbra llamarla Palabra de Estatus de Programa (PSW)

2.1.5. UNIDADES DE ENTRADA/SALIDA

La necesidad de conectar el microprocesador con el mundo exterior exige la conexión al mismo de dispositivos externos conocidos como periféricos, con los cuales el microprocesador intercambia información.

Se ha mencionado anteriormente que la memoria de los computadores esta dividida en localidades identificadas cada una por una dirección, las cuales pueden ser accesadas en forma independiente por el microprocesador. De la misma manera se hace necesario que el microprocesador identifique o se refiera a los dispositivos periféricos de alguna forma que los diferencie unos de otros y de los demas elementos del computador.

Existen basicamente dos tecnicas por medio de las cuales los microprocesadores pueden identificar o direccionar los diferentes dispositivos de entrada/salida o periféricos:

Identificación del periférico mediante el uso de las mismas direcciones del espacio de memoria del microprocesador.

Identificación del periférico mediante direcciones especiales distintas a las del espacio de memoria del microprocesador.

La primera consiste en asignar direcciones a los periféricos como si fuesen localidades de memoria; en este caso el dispositivo periférico es tratado como una dirección mas de memoria y todas las instrucciones que el microprocesador posea para referenciar la memoria pueden ser usadas para transferir información entre el periférico y el microprocesador o viceversa.

Como quiera que las direcciones usadas para referenciar el periférico forman parte del espacio disponible para memoria, el hecho de conectar n periféricos, reducirá en n el espacio disponible para memoria propiamente dicha, que como se dijo anteriormente será determinado por el número de bits del bus de direcciones (2^N, en donde N es el número de bits del bus de direcciones). La tecnica de conexión de periféricos utilizando direcciones que corresponden al espacio de la memoria se conoce como MEMORY MAPPED I/O y aparece esquematizada en la figura 2-7.

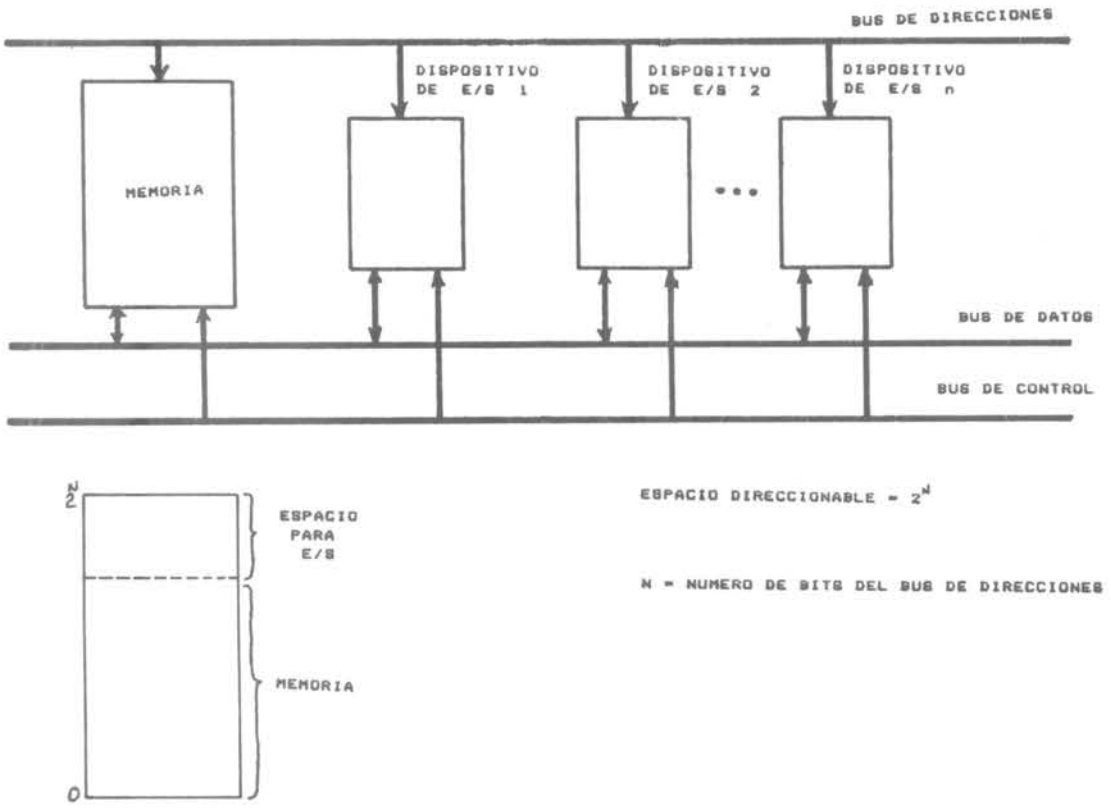


Figura 2-7. Conexión de un periférico usando la tecnica de entrada/salida con asignación de localidades de memoria

La otra tecnica usada consiste en asignar direcciones a los periféricos distintas a las utilizadas para el direccionamiento de la memoria. Esta tecnica requiere que el microprocesador posea instrucciones especiales para comunicarse con los periféricos, llamadas instrucciones de entrada salida y líneas especiales destinadas a la conexión de estos periféricos. Las distintas localizaciones de entrada/salida son llamadas puertos de entrada/salida; los puertos de entrada salida tienen líneas específicas para la introducción de los datos constituyendo verdaderas interfases internas al microprocesador. La figura 2-8 muestra este tipo de conexión el cual es comunmente llamado I/O MAPPED I/O.

Algunos microprocesadores solo permiten el uso de la primera tecnica para la conexión de sus periféricos careciendo de instrucciones y líneas especiales para la conexión de los mismos.

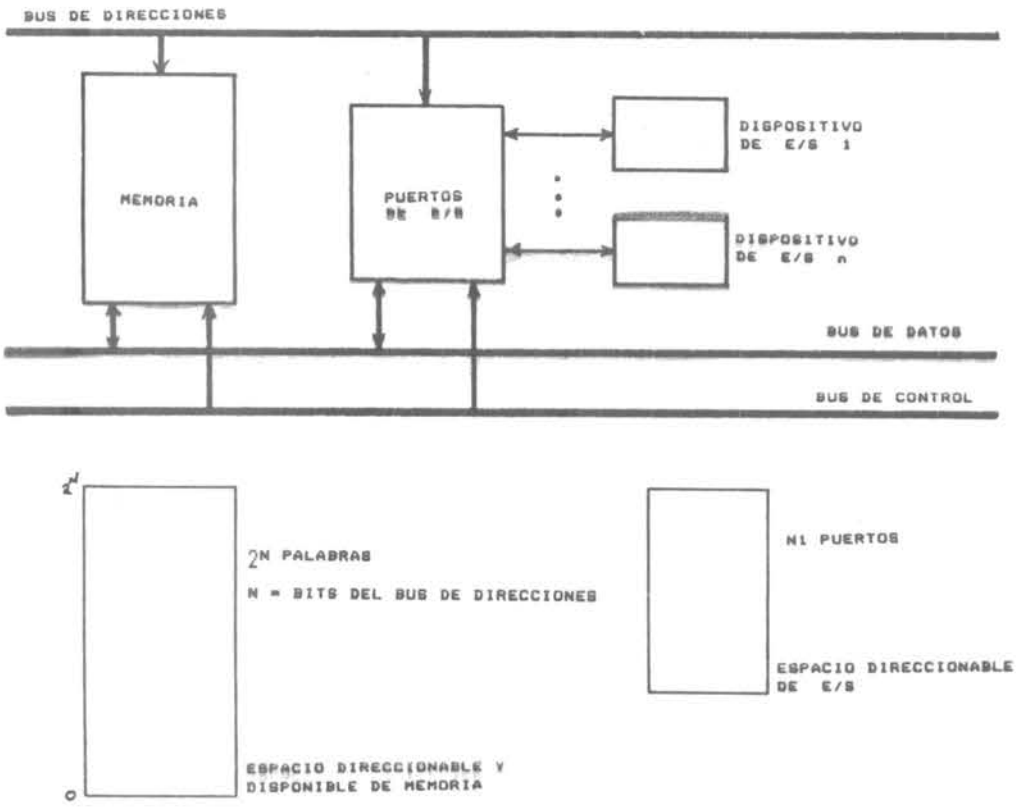


Figura 2-8. Tecnica de conexión con direcciones especiales para entrada/salida (I/O MAPPED I/O)

Adicional a la identificación del periférico, que acabamos de ver se puede hacer por los dos métodos descritos, debe proveerse de alguna forma que garantize que la información enviada por el periférico sea tomada por el microprocesador y viceversa.

Tres son las tecnicas mas ampliamente usadas para lograr estos propositos:

E/S manejada por programa
E/S manejada por interrupción
E/S manejada con DMA

a) E/S manejada por programa

Una forma comun es la realizada por control de programa en la cual en el programa se toman las previsiones para periodicamente chequear para ver si un determinado periférico requiere hacer una transacción. Este chequeo se hace generalmente sobre una señal de control que es activada por el periférico; el periférico por su parte espera a que el

computador "autorize" la transacción, lo cual tambien se hace por una señal de control activada a través del programa en ejecución; Al recibir esta señal, el periférico procede a enviar la información. Este protocolo de comunicación, consistente en pedir atención y esperar por una respuesta antes de proceder, se conoce comunmente como "HANDSHAKING".

b) E/S manejada por interrupción

Otra forma de manejar la entrada/salida es aprovechando un hardware especial que generalmente traen incorporado todos los microprocesadores y que permite detener la ejecución de un programa, en forma totalmente asincrónica, para iniciar la ejecución automática de otro, cuya finalidad es atender un requerimiento de atención de un periférico cualquiera que esté conectado al microprocesador; el programa que se interrumpe se reinicia en forma automática, en el sitio donde fue interrumpido, una vez que se haya atendido la solicitud del periférico.

El periférico llama la atención del microprocesador colocando en el nivel lógico que corresponda, según el microprocesador que se esté usando, una entrada de control que para tal efecto traen los microprocesadores llamada de INTERRUPCION (INT), por lo que a este método en general se le conoce como el método por interrupciones.

Existen variaciones en la forma en que se implementa el sistema de interrupción en los microprocesadores. En algunos casos, la señal de interrupción puede ser activada y/o desactivada por el microprocesador, bajo control del programa del usuario, bloqueando cuando lo desee cualquier señal de interrupción que sea generada por un periférico; este sistema de interrupción es conocido como SISTEMA DE INTERRUPCION ENMASCARABLE (MASKABLE INTERRUPT).

En otros casos el sistema de interrupción se denomina NO-ENMASCARABLE (NO MASKABLE INTERRUPT) pues la señal de interrupción no puede ser bloqueada por software y el microprocesador puede ser interrumpido por cualquier periférico que requiera su atención; algunos microprocesadores poseen los dos sistemas y tienen líneas separadas de interrupción, una bloqueable bajo control de programa (MASKABLE INTERRUPT) y otra no bloqueable bajo control de programa (NO MASKABLE INTERRUPT).

Los sistemas de interrupción pueden tambien ser clasificados en FIJOS y VECTORIZADOS; en algunos microprocesadores el programa que atiende las llamadas de atención o interrupciones de los periféricos debe ser colocado en una dirección específica y fija de memoria; en estos casos, cada vez que se genera una interrupción por parte de un periférico, el programa en ejecución es interrumpido y el microprocesador inicia la ejecución en forma automática del

programa que se encuentra en la posición de memoria definida por los diseñadores del microprocesador para tal efecto; este tipo de sistema de interrupción se le conoce como INTERRUPCION EN LOCALIZACIONES FIJAS (FIXED INTERRUPT).

Otros microprocesadores permiten que el programa de atención de interrupción sea colocado en cualquier posición de la memoria. En estos casos el microprocesador al recibir una señal de interrupción inicia la ejecución de una instrucción que permite leer del bus de datos, la dirección y/o instrucción que le permitirá saber a que posición de memoria deberá "saltar", para ejecutar a partir de la misma, el programa de atención de la interrupción del periférico en cuestión; Es responsabilidad del usuario en estos casos, proveer el hardware necesario para colocar en el bus la dirección o vector que corresponda al periférico que interrumpe. Este sistema es conocido como INTERRUPCION VECTORIZADA (VECTORED INTERRUPT).

Sea cual sea el sistema de interrupción del microprocesador, en esencia cumple los mismos propósitos cual es el de permitir que un programa sea interrumpido en un momento determinado por un periférico cualquiera, suspendiendose su ejecución mientras se ejecuta el programa de atención de interrupción, y retornar posteriormente al programa principal en el mismo punto y en las mismas condiciones en que se encontraba para el momento en que se generó la interrupción.

Es de hacer notar la diferencia existente entre los dos métodos que se han descrito o sea atención bajo control de programa y atención por interrupción; en el primer caso en el programa se deben incluir instrucciones que periodicamente interroquen al periférico para ver si requiere atención; en cambio en el segundo caso, dentro del programa principal no se incluye ninguna instrucción para estos propósitos si no que se hace un programa para atención de las interrupciones, que normalmente no se ejecuta, a menos que un periférico determinado requiera de atención; este programa de atención de interrupción es automáticamente "arrancado" por el hardware del sistema de interrupción del microprocesador.

c) E/S manejada por DMA

Una tercera técnica es la llamada de ACCESO DIRECTO A MEMORIA (DMA). Esta técnica se usa generalmente para la conexión de periféricos de alta velocidad; existen diferentes implementaciones de DMA, las cuales se explicaran en un capítulo posterior, sin embargo, todas tienen como característica fundamental el hecho de poder transferir información entre un dispositivo periférico y la memoria del computador, sin tener que pasar por ningún registro del procesador y sin que este intervenga en forma directa en el control y realización de ninguna actividad relacionada con la transferencia de información.

2.2. ARQUITECTURA DEL 8035

El microprocesador 8035, pertenece a la familia MCS-48 desarrollada por INTEL; es fabricado con tecnología MOS de canal N (NMOS) y se presenta en un encapsulado de 40 pines, como se muestra en la figura 2-9.

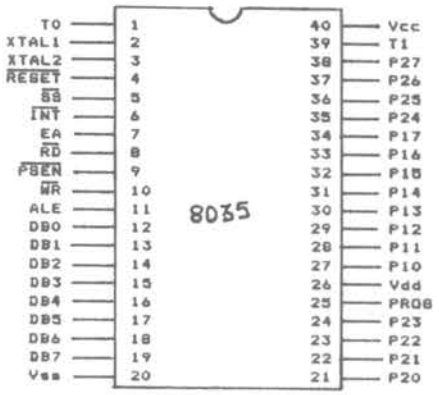


Figura 2-9. Microprocesador 8035

El miembro principal de la familia MCS-48, el 8048, constituye un verdadero microcomputador encapsulado en un solo chip pues posee todos los elementos requeridos en un microcomputador incluyendo 2Kbytes de memoria de programa. El 8035 es una versión del 8048 que no posee memoria de programa interna, pero que por lo demas es totalmente compatible con el 8048.

El 8035 posee un bus de datos de 8 bits, el cual comparte las mismas líneas físicas con el bus de direcciones; todos sus registros son de 8 bits y es capaz de direccionar hasta 4096 bytes de memoria (4 Kbytes); posee tambien dos puertos de entrada/salida, cuasi bidireccionales y requiere de solo una fuente de alimentación, +5volts, para su operación; el circuito de reloj viene tambien integrado en el mismo chip requiriendo externamente solo de un cristal o un inductor para generar el reloj del sistema.

El 8035 posee un sistema de interrupción del tipo que se ha definido como enmascarable (activable o desactivable desde un programa) y de localización fija (fixed interrup); dicho sistema acepta interrupciones desde tres fuentes distintas: desde la línea de interrupción externa del microprocesador (INT), lo cual genara un salto automático a la posición 3 de memoria de programa; desde la línea de RESET del microprocesador, lo cual genera un salto automático a la posición 0 de la memoria de programa y por último desde el Timer/Counter interno del microprocesador, lo cual genera un salto automático a la dirección 7 de la memoria de programa. La interrupción generada por la entrada de RESET no es enmascarable.

Los programas pueden ser ejecutados en forma normal, secuencialmente recorriendo en forma automática todas las instrucciones del programa, o en la modalidad "paso a paso" en la cual a traves de una entrada de control que trae para ese propósito el microprocesador, se puede ejecutar el programa avanzando de una instrucción a otra en forma manual.

En resumen las características mas resaltantes del 8035 se dan a continuación:

-CPU de 8 bits

-Capacidad de entrada/salida de 27 líneas

-Dos bancos de registros (cada banco con 8 registros)

-Capacidad de direccionamiento de 4Kbytes (4096 bytes)

-Ocho niveles de subrutina

-Sistema de interrupción enmascarable y de vector fijo

-Posibilidad de ejecución de programas "paso a paso"

-Fuente única de alimentación de +5V

-Circuito de reloj interno en un rango de 1 a 6 MHz

-Aproximadamente 70% de las instrucciones requieren un solo ciclo

-Contador/Temporizador de 8 bits

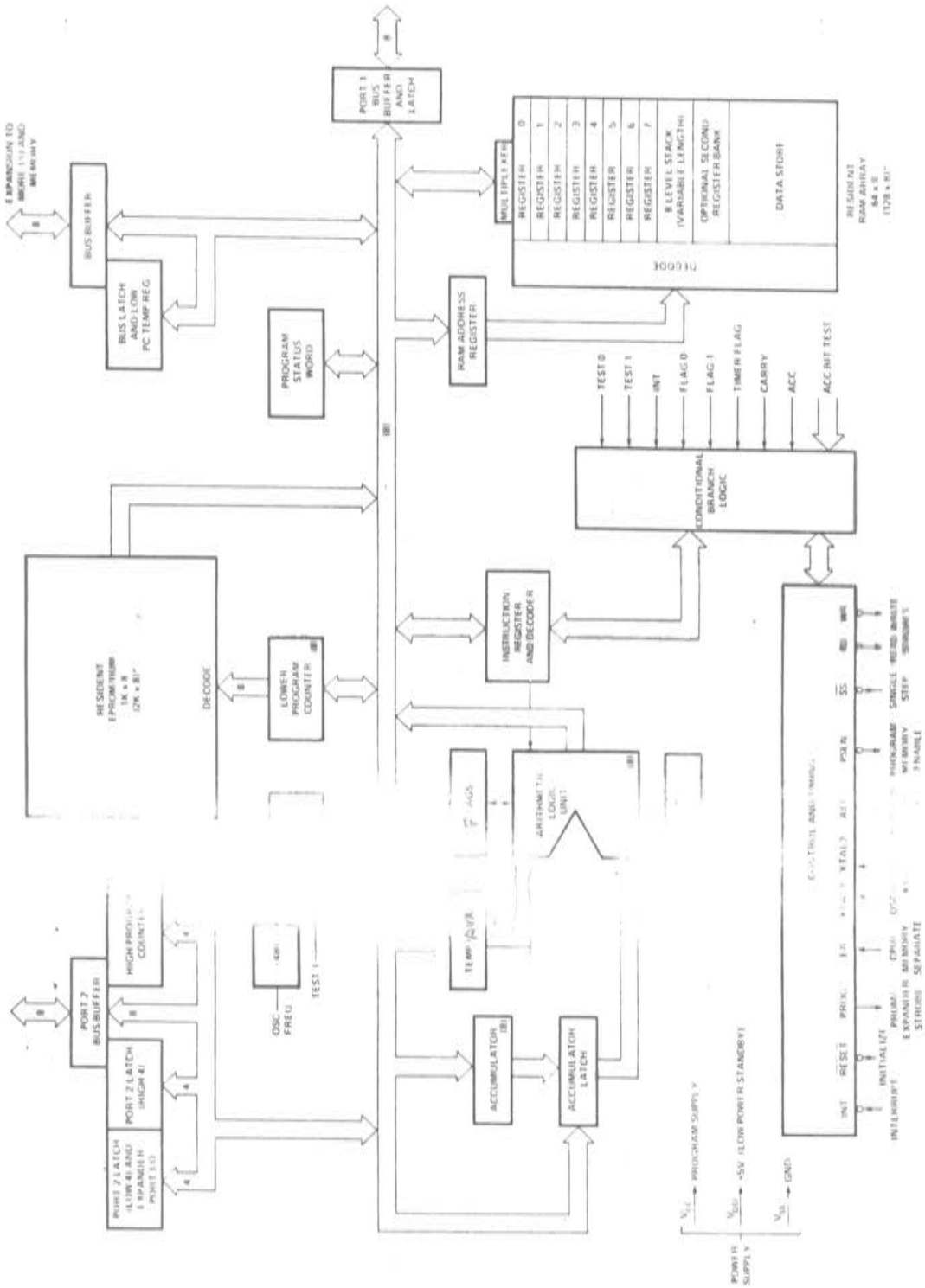


Figura 2-10. Diagrama de bloque del 8085

2.2.1. UNIDAD DE PROCESO

La unidad de proceso del 8035 esta compuesta por (ver figura 2-11):

Unidad Aritmética y Lógica (ALU).
Acumulador.
Registro auxiliar.
Flag de acarreo (carry).
Flag de acarreo auxiliar.
Logica de ajuste decimal.

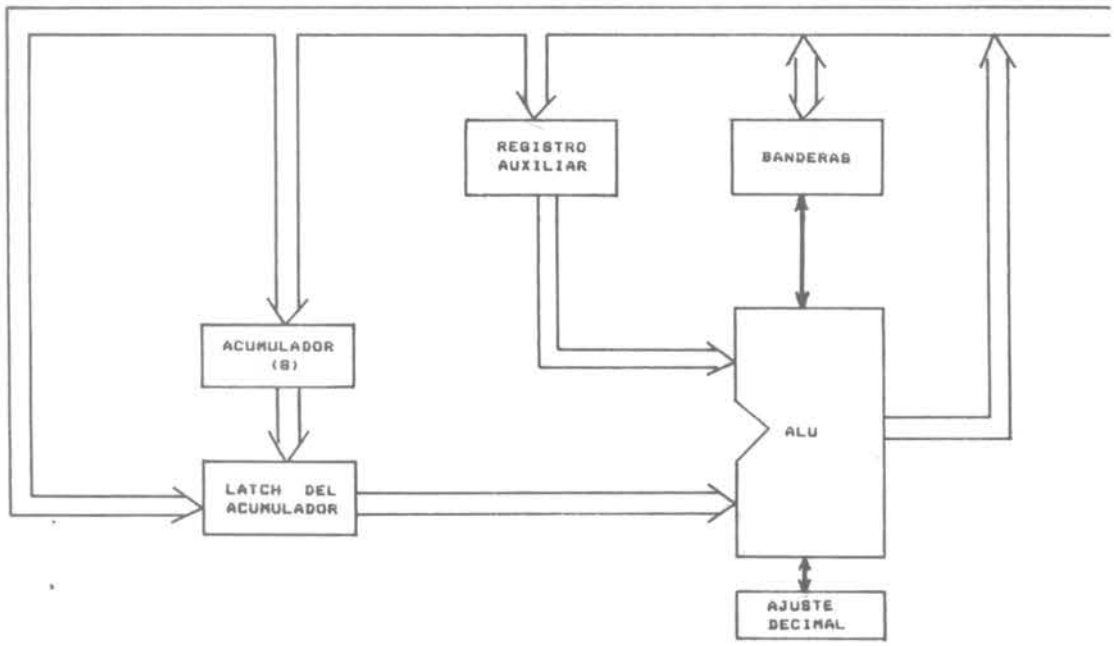


Figura 2-11. Unidad de proceso del 8035

2.2.1.1. Unidad Aritmética y Lógica del 8035

La ALU del 8035 es una unidad aritmética de 8 bits que

puede aceptar operandos desde una de dos fuentes, el acumulador y/o el registro auxiliar. Es capaz de realizar las siguientes operaciones:

Suma con o sin acarreo.
Operaciones lógicas de AND, OR, XOR
Complementar bits del acumulador.
Rotaciones a la derecha.
Rotaciones a la izquierda.
Intercambio de nibbles.
Incremento del acumulador.
Decremento del acumulador.
Ajuste decimal para suma en BCD.

2.2.1.2. Acumulador y Registro Auxiliar

El acumulador es el registro de datos mas importante del 8035. Es fuente de entrada de datos a la ALU y el destino mas frecuente de las operaciones realizadas por ella; ademas los datos provenientes de los puertos de Entrada/Salida normalmente tambien pasan por el acumulador.

El registro auxiliar tambien sirve de fuente de datos a la ALU, el mismo se usa cuando se ejecutan las siguientes instrucciones del 8035:

ADD A, #dato
ADD A, R_n
ADD A, @R_o

2.2.1.3. Registros Banderas (flags)

Los registros de acarreo, Acarreo C y Acarreo auxiliar AC, los usa el 8035 para indicar si hubo acarreo o si la suma en BCD requiere de ajuste o corrección de resultados segun se explicó en líneas anteriores. Si el bit de acarreo se hace uno (C=1), significa que hubo acarreo en la operación que realizó la ALU; si el bit de acarreo auxiliar se encuentra en uno (AC=1), entonces hubo acarreo en una suma en BCD y se requiere de ajuste decimal.

2.2.2. UNIDAD DE CONTROL

La Unidad de Control del 8035 esta compuesta por (ver figura 2-12):

Registro de Instrucción (IR).
Decodificador de Instrucciones (ID)
Lógica de Salto Condicional.
Lógica de control de tiempo.
Contador de Programa (PC).
Registro o palabra de estatus (PSW).

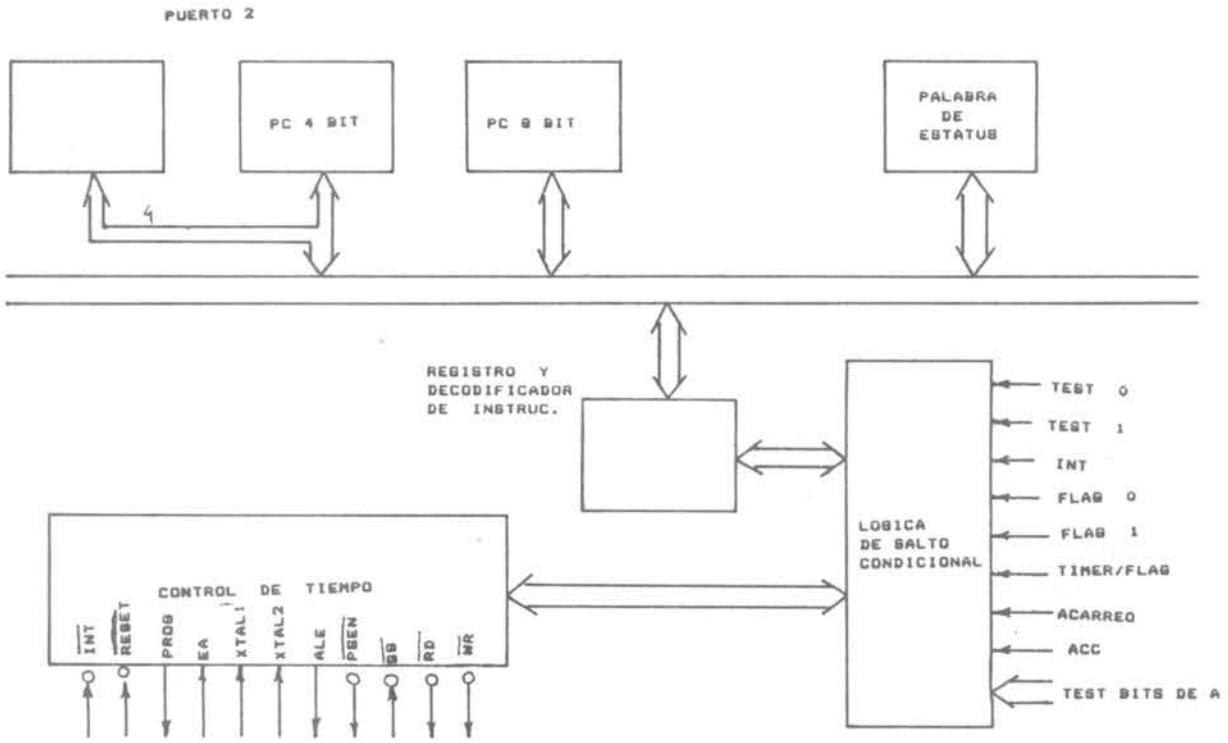


Figura 2-12. Diagrama de bloque simplificado de la unidad de control del 8035

2.2.2.1. Registro de instrucción y Decodificador

El registro de Instrucción del 8035 es un registro de 8 bits en el cual se almacena la instrucción buscada de memoria para que el circuito decodificador la interprete y genere las señales a que haya lugar según la instrucción. La forma en la que el decodificador genera estas señales de control fue explicado en este mismo capítulo en la sección 2.1.3.

2.2.2.2. Lógica de Control de Salto Condicional

La lógica de salto condicional permite al usuario examinar ciertas condiciones internas y/o externas al microprocesador y en base al estado lógico en que se encuentren las mismas decidir si se prosigue con la ejecución normal del programa (secuencialmente) o si se rompe esa secuencia normal de ejecución.

Existen en el 8035 una serie de instrucciones que permiten hacer estos chequeos sobre un determinado número de señales o condiciones y en base a estas "saltar" a una porción de programa distinta a la que correspondería si se siguiese la ejecución secuencial o normal del programa.

La siguiente tabla resume todas las condiciones que permite chequear el 8035 y las instrucciones asociadas a estos chequeos; ejemplos de uso de estas instrucciones se verán en el CAPITULO III

SEÑAL CHEQUEADA	INSTRUCCION DE SALTO	CONDICION QUE GENERA SALTO
Acumulador A	JZ dir	A=0
Acumulador A	JNZ dir	A≠0
Bit de A	JB_n dir	B_n=1
Flag Acarreo	JNC dir	C=0
Flag acarreo	JC dir	C=1
Señal F0	JF0 dir	F0=1
Señal F1	JF1 dir	F1=1
Flag overflow	JTF dir	F=1
Señal T0	JNT0 dir	T0=0
Señal T0	JT0 dir	T0=1
Señal T1	JNT1 dir	T1=0
Señal T1	JT1 dir	T1=1
Interrupcion I	JNI dir	I=0

Si un programa en ejecución tiene entre sus instrucciones la instrucción de salto condicional JNZ dir (salte a la dirección especificada si el contenido del acumulador es distinto de cero), en el momento en que esta instrucción sea cargada en el IR y decodificada por el decodificador de instrucciones, se generarán las señales de control apropiadas para hacer que la lógica de salto condicional chequee la condición del registro Z, indicador de si el contenido del acumulador es cero (Z=1) o si es distinto de cero (Z=0); si Z=0 significa entonces que el contenido del acumulador es distinto de cero (se cumple la condición de salto) y por lo tanto se generan las señales de control a que haya lugar, de manera que el contenido del Contador de Programa PC, sea sustituido por la dirección especificada en la instrucción de salto, con lo cual se interrumpe el avance secuencial en la ejecución del programa.

2.2.2.3. Circuito de Control y Tiempo

Este circuito es el encargado de producir los distintos tiempos de la máquina, recibir y enviar señales de control precisas en el tiempo las cuales sincronizan el sistema. A continuación se estudiará el circuito de reloj y tiempo del 8035 y luego las distintas señales de control que entran y salen de este bloque funcional.

Reloj y Circuitos de Tiempo

La generación de los tiempos para sincronizar todas las unidades funcionales del 8035 son internas a el, con la excepción de la referencia en frecuencia, para lo cual se usa generalmente un cristal de cuarzo conectado externamente al microprocesador. El reloj y el circuito de tiempo se pueden dividir en los siguientes bloques funcionales:

a. Oscilador

b. Contador de Estados

c. Contador de Ciclos

a. Oscilador

El oscilador interno del 8035 esta constituido por un circuito resonante de alta ganancia y rango de frecuencia de 1 a 6 MHz. El pin externo XLT1 es la entrada a la etapa amplificadora y XLT2 es la salida. Un cristal o un inductor conectado entre XLT1 y XLT2 proveerá la realimentación y el desplazamiento de fase requerido para la oscilación del

circuito.

b. Contador de Estados

La salida del oscilador es dividida por 3 en el contador de estados de manera de crear asi un reloj que defina los tiempos de duración de los estados del microprocesador. Este reloj denominado CLK puede estar disponible externamente al 8035 a traves del pin denominado T0 al ejecutar una instrucción que para tal efecto trae el 8035 (ENT0 CLK). La salida de CLK por T0 solo puede ser deshabilitada dandole un reset al microprocesador.

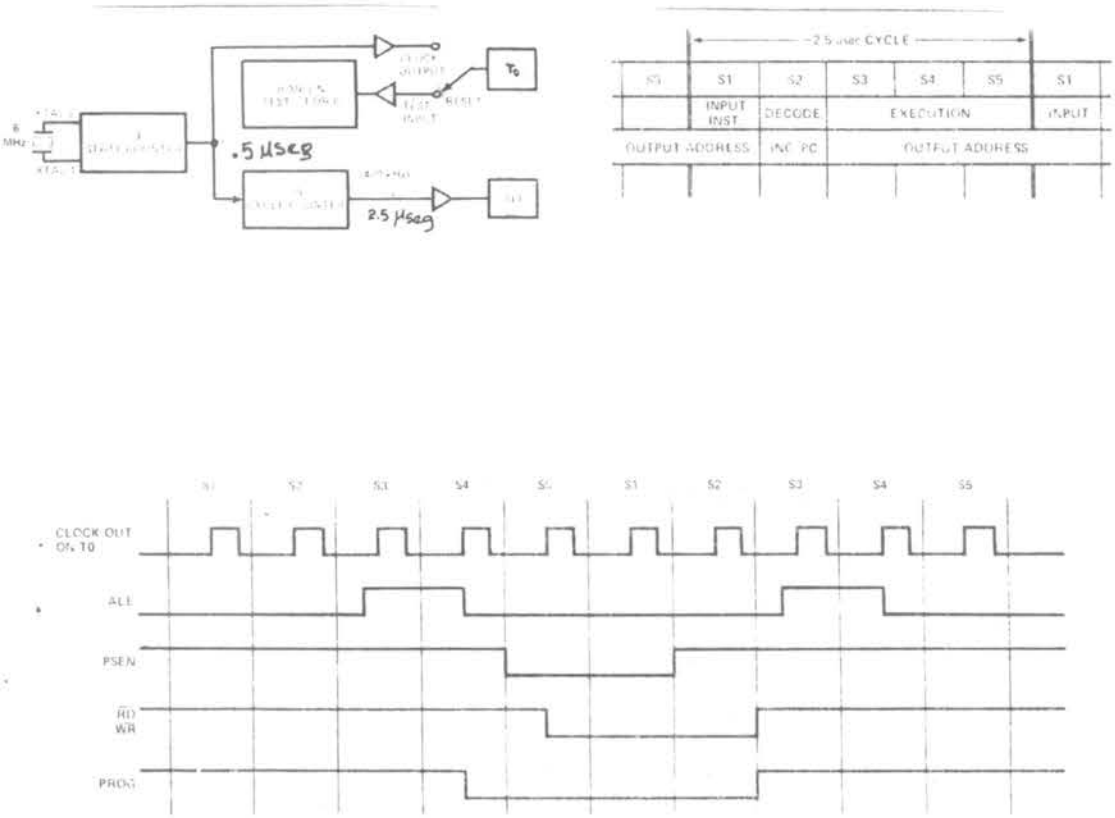


Figura 2-13. Contador de estados del 8035

c. Contador de Ciclos

La señal CLK proveniente del contador de estados es dividida por 5 en el contador de ciclos, produciendo así un reloj el cual define un ciclo de máquina que viene a estar constituido por 5 estados del procesador. Este reloj se llama habilatador de los registros de dirección (Address Latch Enable= ALE) debido a su función en el 8035 ya que indica cuando en el bus de datos estan estables las direcciones y pueden ser almacenadas en los registros de dirección (el 8035, como ya se dijo antes comparte las líneas del bus de datos con el bus de direcciones). La señal ALE esta disponible externamente al 8035 en el pin denominado con ese mismo nombre (ALE pin 11).

Señales de Control de Entrada/Salida

Estas señales seran usadas ampliamente durante el curso en las conexiones del microprocesador a los distintos periféricos; a continuación una lista las señales de control del 8035 y una descripción de la función que cumple cada una de ellas en el 8035:

Señales de control del 8035 { $\overline{INT}$, $\overline{RESET}$, PROG, EA, ALE, $\overline{PSEN}$, $\overline{SS}$, $\overline{RD}$, $\overline{WR}$

a) Señal $\overline{INT}$ (pin 6)

$\overline{INT}$ es la señal prevista en el 8035 para permitir la interrupción de programas en forma asincrónica y como consecuencia de la ocurrencia de un evento externo que requiera la atención del microprocesador. Por medio de la señal $\overline{INT}$ es posible interrumpir la ejecución de un programa, para proceder a la ejecución del programa de atención de la interrupción y retornar luego al programa interrumpido al finalizar la ejecución del programa de atención de la interrupción, tal como se vió en líneas anteriores. En el microprocesador 8035 la interrupción se inicia al colocar la entrada $\overline{INT}$ en el nivel cero lógico. Puesto que el sistema de interrupción del 8035 es del tipo que hemos llamado enmascarable, para que la interrupción sea atendida el sistema de interrupción debe estar habilitado; el 8035 posee dos instrucciones que permiten desde

un programa habilitar o deshabilitar el sistema de interrupción por la línea $\overline{INT}$, estas instrucciones son: ENI (Enable Interrup), para habilitar y permitir la interrupción y la instrucción DISI (Disable Interrup) para deshabilitar e impedir que un determinado programa sea interrumpido.

Cuando el programa en ejecución es interrumpido y antes de que se empiece a ejecutar el programa de atención de la interrupción, se hace necesario guardar el estatus del primer programa para que cuando se termine de ejecutar el de atención de la interrupción se pueda regresar el control al primer programa. El estatus del programa viene dado por: El valor del PC que está apuntando a la proxima instrucción a ejecutar y el estado de los flag o banderas de acarreo, acarreo auxiliar, etc.; estas últimas agrupadas, según se verá mas adelante, en los 4 bits más significativos del registro de estatus de programa (PSW)

El contador de programa del 8035 tiene un ancho de 12 bits y en conjunto con los 4 bits de la PSW$^{(MSB)}$ ocupan 16 bits. Los 16 bits correspondientes al PC (12 bits) y a las banderas de estatus (4 bits), son almacenados automáticamente por el microprocesador en una memoria interna que posee el mismo y que se denomina STACK. Una vez que el PC y la PSW estan almacenados en el stack, el contenido del PC se cambia por el de 03_H, dirección esta donde se deberá colocar la primera instrucción del programa de atención de la interrupción, ya que como se dijo anteriormente, el sistema de interrupción del 8035 es del tipo llamado de interrupción fija, en el cual cuando ocurre una interrupción, el microprocesador salta siempre a ejecutar la instrucción presente en una dirección fija que en el caso del 8035 y para interrupciones externas es la 03_H.

Se describirá a continuación el funcionamiento de la memoria STACK y sus operacions PUSH (almacene en el stack) y POP (saque del stack) y luego se procederá a explicar como funciona el stack del 8035.

Un STACK o pila es una memoria que se caracteriza primordialmente por el hecho de que los datos que en ella se almacenan salen en orden inverso a como entran, esto es, el último dato que entra es el primero en salir. Todo stack tiene un apuntador que indica cual fue el último dato que se almacenó.

En la figura 2-14 se muestra un stack en donde se han almacenado los datos A, B y C en ese orden de entrada.

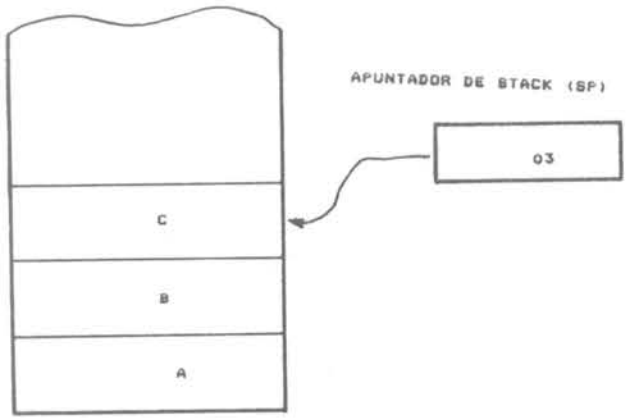


Figura 2-14. STACK y su apuntador

Supongase ahora que se va a almacenar un nuevo dato D en el stack, esta es una operación de PUSH (empuje). Primero el apuntador de stack se incrementa en uno y luego se almacena el dato D en la dirección apuntada por el apuntador de stack, tal como se muestra en la figura 2-15.

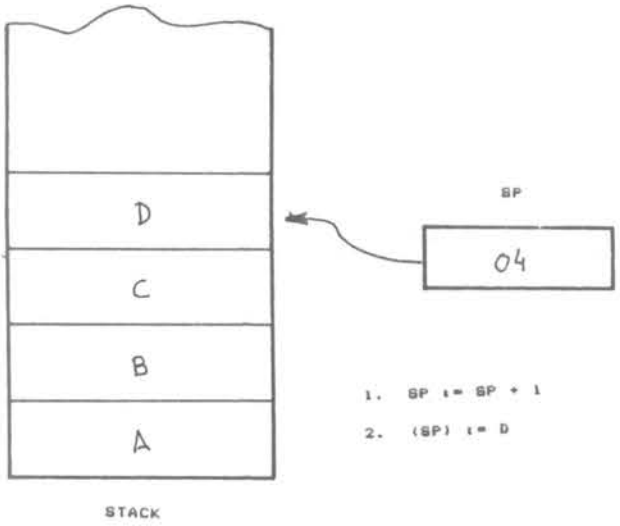


Figura 2-15. Operación de meter información en un stack (PUSH)

Si la operación que se desea realizar es la extracción de un dato almacenado en el stack para guardarlo en una dirección de memoria M se procede como sigue: se guarda primero en la dirección de memoria M el dato apuntado por el apuntador de stack y luego el apuntador de stack se decrementa en uno tal

como se muestra en la figura 2-16. Esto constituye una operación POP (POP M).

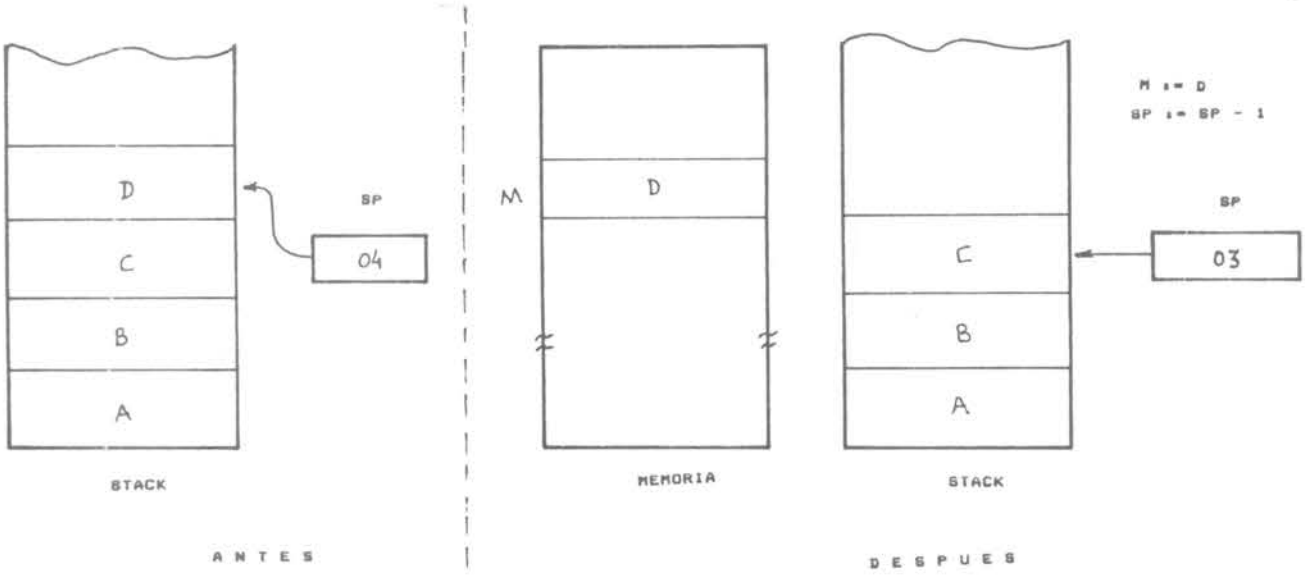


Figura 2-16. Operación de extracción de un dato del stack (pop)

Conociendo ya como funcionan las operaciones POP y PUSH de un stack supongamos ahora que se está ejecutando un programa P1 en el 8035 y que el mismo es interrumpido cuando el contenido del contador de programa era 020_H; automáticamente se almacenarán el PC y la PSW en el stack, mediante una instrucción push y luego se procede a atender la interrupción tal como se muestra en la figura 2-17.

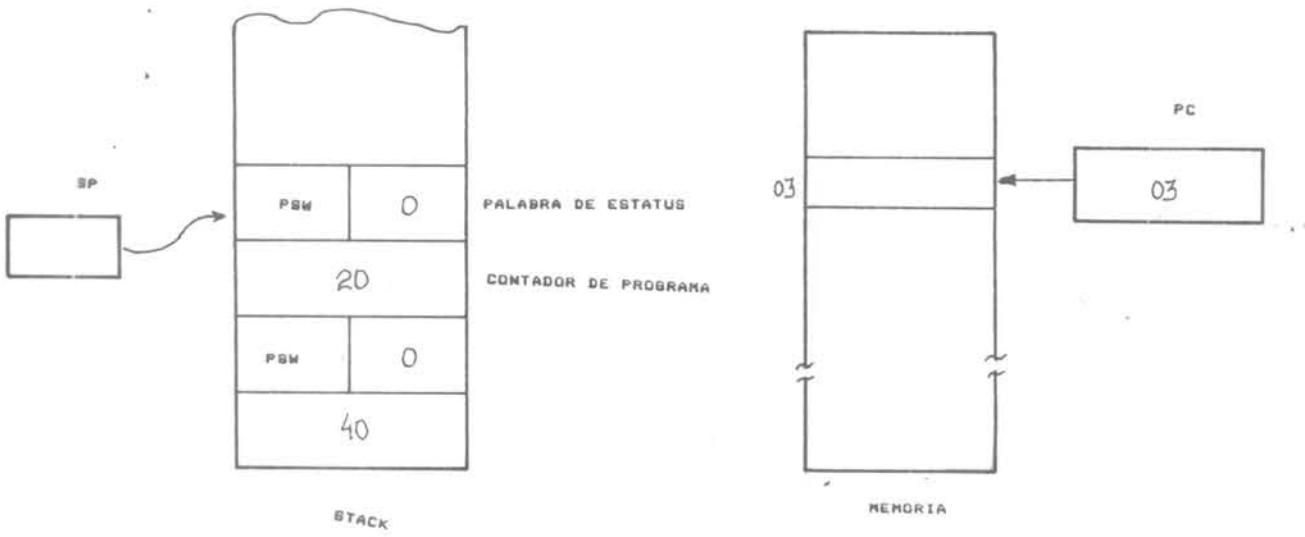


Figura 2-17. Secuecia de atención de una interrupción

Cuando el programa P2, el cual es el que atiende la interrupción termina, se procede a ejecutar automáticamente un pop del stack, con lo cual se restaura el viejo PC y la PSW, retornandose al programa P1 tal como se muestra en la figura 2-18, en la que tambien se muestra un esquema del flujo de trabajo del microprocesador.

En referencia al programa que atiende la interrupción vale la pena destacar que puede contener como parte de el cualquier instrucción del set del microprocesador, sin embargo, deberá ser terminado con una instrucción especial, RETR (retorne y restaure registros) con la finalidad de que ocurra la operación descrita en el parrafo anterior y retornar así al programa interrumpido en las mismas condiciones en las que estaba al momento de la interrupción.

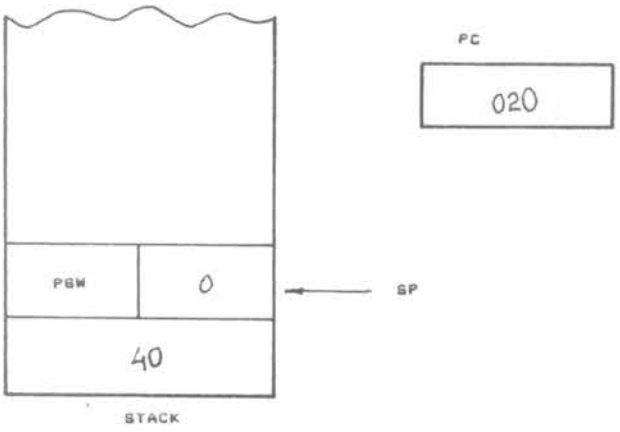


Figura 2-18. Operación de retorno de una interrupción

b) Señal $\overline{\text{RESET}}$ (pin 4)

La entrada de $\overline{\text{RESET}}$ permite inicializar el 8035 y empezar a ejecutar el programa que comienza en la dirección 000_H de la memoria; durante la ejecución de un programa la entrada de $\overline{\text{RESET}}$ se debe mantener en el nivel lógico uno y cuando se desee realizar un reset, esta señal deberá mantenerse en el nivel cero lógico durante por lo menos cinco (5) ciclos de máquina. El siguiente circuito se usa generalmente para garantizar la aplicación de la señal de reset por el tiempo apropiado al pin $\overline{\text{RESET}}$.

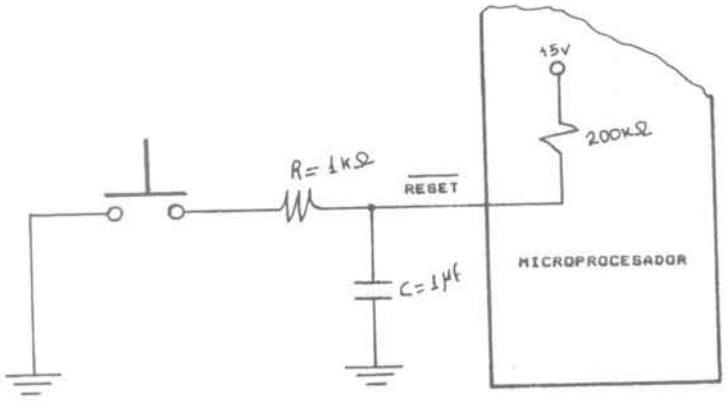


Figura 2-19. Circuito para producir el reset en el 8035

A continuación un resumen de las funciones que cumple la señal de $\overline{\text{RESET}}$ en el 8035:

Coloca el contador de programa PC en cero

Coloca el apuntador de stack en cero

Selecciona el banco de registro cero

Selecciona el banco de memoria cero

Coloca el bus de datos en el estado de alta impedancia

Inicializa los puertos 1 y 2 como puertos de entrada

Deshabilita el sistema de interrupciones

Detiene el contador de tiempo (TIMER)

Desactiva el flag del timer

Desactiva F0 y F1

Deshabilita salida de reloj por T0

c) Señal PROG (pin 25)

La señal de control PROG es usada para controlar el expansor de puertos 8243 que será usado en este curso y cuya descripción se hará en capítulos posteriores. En otros miembros de la familia MCS 48 específicamente el 8748, este pin sirve para aplicar los impulsos de programación de la memoria EPROM que el mismo trae internamente.

d) Señal EA (pin 7)

Esta señal permite indicar al microprocesador que la memoria de programa es externa. En el caso del 8035 esta señal debe estar siempre activada en virtud de que el mismo no posee memoria de programa interna; otros miembros de la familia MCS 48 si poseen memoria de programa interna y esta señal les define en cual memoria buscarán los datos y las instrucciones: en la externa (EA=1) o en la interna (EA=0).

e) Señal ALE (pin 11)

La señal ALE (Address Latch Enable) permite saber en que momento la información presente en el bus de datos constituyen una dirección ya que como se mencionó anteriormente, en el 8035 el bus de datos y el de direcciones comparten las mismas líneas físicas.

En el sistema IMSAI la señal ALE se usa para cargar unos flip flops que mantienen la dirección despues que la misma ha desaparecido del bus permitiendo así disponer de un bus de direcciones separado tal como se muestra en la siguiente figura.

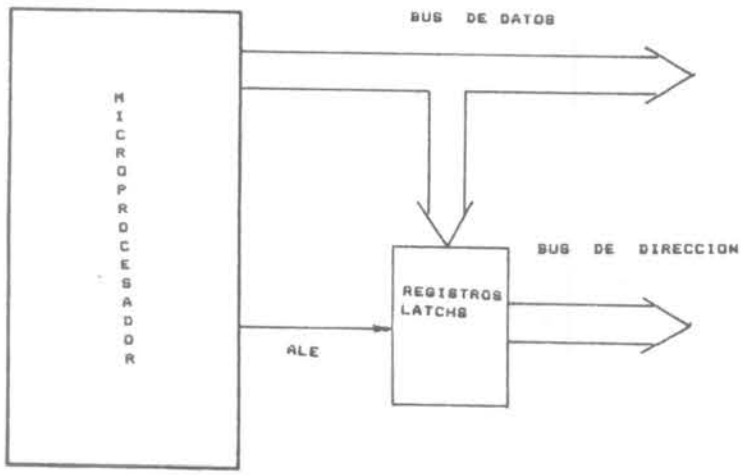


Figura 2-20. Uso de la señal ALE para la construcción de un bus de direcciones separado

f) Señal $\overline{\text{PSEN}}$ (pin 9)

La señal $\overline{\text{PSEN}}$ (Program Store Enable) es usada para indicar que se esta realizando la busqueda de una instrucción desde memoria externa. Esta señal solo se activa en la fase de busqueda de una instrucción.

g) Señal $\overline{RD}$ (pin 8)

Esta señal es usada para indicar que se está realizando una operación de lectura sobre el BUS. Se usa generalmente para habilitar la entrada de datos en el bus desde un dispositivo externo. Esta señal se activa siempre que se ejecute una operación de lectura sobre el bus a diferencia de la señal $\overline{PSEN}$ que solo se activa, cuando se está en la fase de busqueda de una instrucción.

h) Señal $\overline{SS}$ (pin 5)

La entrada de control $\overline{SS}$ (Single Step) permite al usuario ejecutar los programas paso a paso ejecutando una instrucción a la vez, lo cual es útil cuando se desean eliminar posibles errores de un programa. La operación del microprocesador en esta modalidad permite que en el lapso que el está parado muestre sobre el bus de datos y sobre la parte baja del puerto 2, la dirección de la próxima instrucción a ejecutar, permitiendo al usuario ir paso a paso en su programa y verificar la secuencia de las instrucciones que estan siendo ejecutadas.

En la figura 2-21 se muestra el diagrama de tiempo de la interacción entre la salida ALE y la entrada $\overline{SS}$. El 8035 opera en el modo single step(pasa a paso) como sigue:

1) Al procesador se le pide que pare aplicando un nivel bajo sobre $\overline{SS}$.

2) El procesador responde parandose durante la porción de captación de la siguiente instrucción. Si una instrucción de doble cíclo se esta ejecutando cuando se recibe el comando de single step, los dos ciclos seran completados antes de parar.

3) El procesador avisa que está parado, colocando la señal de ALE en un nivel alto. En este estado (el cual puede ser mantenido indefinidamente), la dirección de la próxima instrucción a ser captada está presente sobre el bus y la mitad baja del puerto 2.

4) Colocando $\overline{SS}$ en un nivel alto el procesador saldrá del modo parado, lo cual le permite captar la siguiente instrucción. La salida de este modo es indicada por el procesador colocando ALE en un nivel bajo.

5) Para parar el procesador en la siguiente instrucción $\overline{SS}$ se debe de poner en un nivel bajo de nuevo tan pronto como ALE caiga a un nivel bajo. Si $\overline{SS}$ se deja a un nivel alto el procesador continuará ejecutando el programa normalmente.

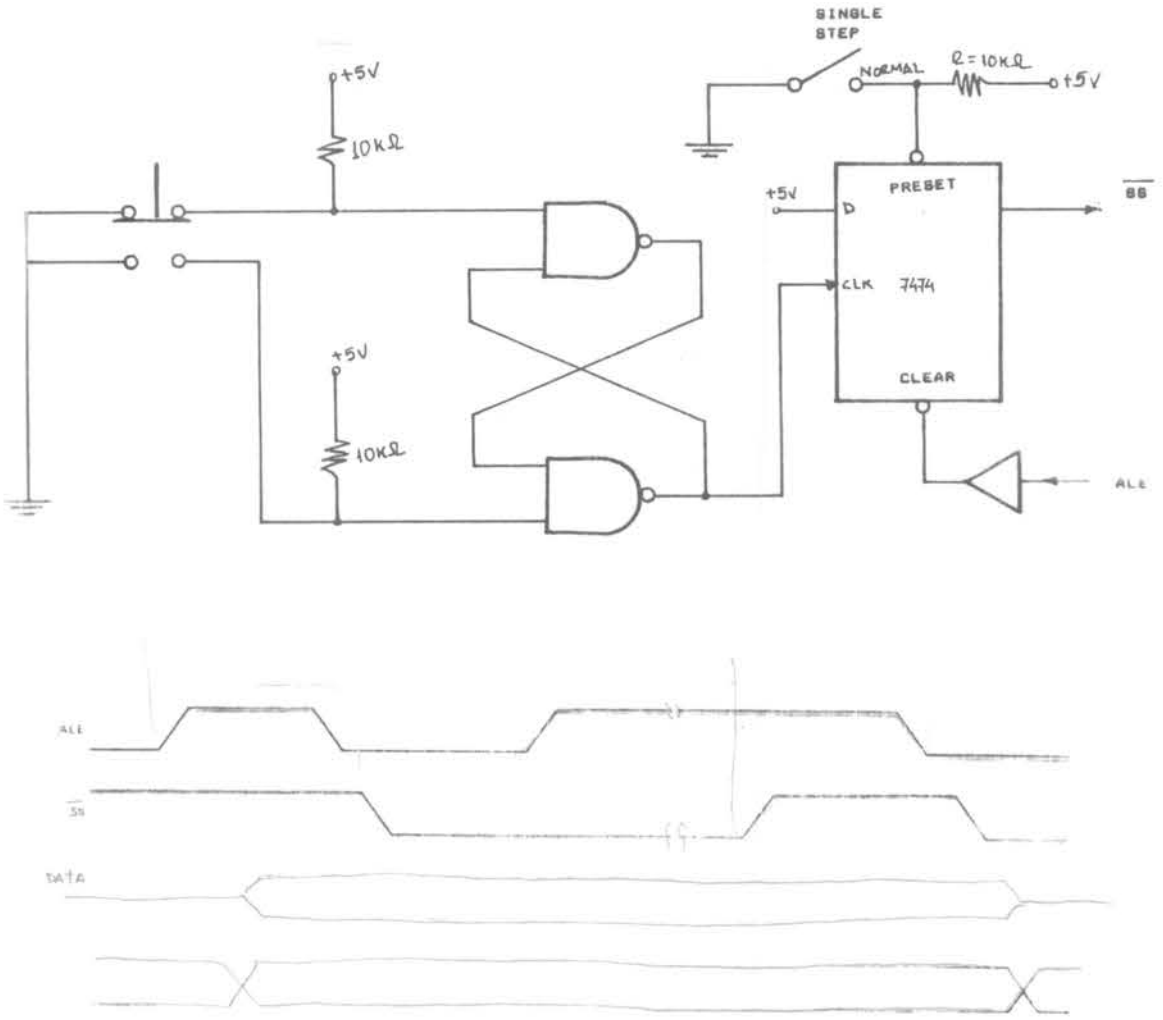


Figura 2-21. Circuito y diagrama de tiempo de la operación paso a paso

Un circuito para implementar la operación del 8035 en single step (paso a paso) se muestra tambien en la figura 2-21; su operación es como sigue: un flip flop tipo D con preset y clear se usa para generar $\overline{SS}$. En el modo run (operación normal), $\overline{SS}$ se mantiene en el estado alto por medio de la entrada preset del flip flop (en este flip flop el preset tiene prioridad sobre el clear). Para entrar en el modo single step (paso a paso), la señal de preset es removida permitiendo a ALE llevar a un nivel bajo $\overline{SS}$ a traves de la entrada de clear. La señal de ALE debe ser buferizada por que la entrada clear del flip flop 7474 es equivalente a tres cargas TTL. El protesador está ahora en el estado parado. La proxima instrucción es iniciada metiendo un uno en el flip flop a traves de un pulso de reloj. Este uno no aparecerá sobre $\overline{SS}$ a menos que la señal ALE en estado alto remueva la orden de clear del flip flop. En respuesta al flanco

de subida de $\overline{SS}$, el procesador, comienza la captación de una instrucción lo cual trae a ALE al nivel bajo, poniendo a $\overline{SS}$ en cero por medio de la entrada clear y causando que el procesador entre de nuevo en el estado parado.

i) Señal $\overline{WR}$ (pin 10)

La señal $\overline{WR}$ es usada para indicar una operación de escritura sobre el bus de datos; se usa generalmente para cargar datos desde el bus en dispositivos externos conectados al mismo.

2.2.2.4. Registro Contador de Programa y su Stack

El registro contador de programa PC, como se mencionó anteriormente, siempre apunta a la proxima instrucción a ejecutar; tiene un ancho de doce bits (12 bits) y es usado para direccionar hasta 2^{12} (4096) localizaciones de memoria de programa. En el caso del 8035 esta memoria es externa al microprocesador y en ella se almacenan los programas que se van a ejecutar; la organización de la misma se describirá en secciones posteriores.

Los 12 bits del contador de programa PC, se dividen en la siguiente forma: los 8 bits menos significativos, que salen por el bus de datos compartiendo las mismas líneas y los 4 bits mas significativos que salen por los cuatro bits menos significativos del puerto 2 del 8035 (P_{2-0} a P_{2-3}) tal como se muestra en la siguiente figura

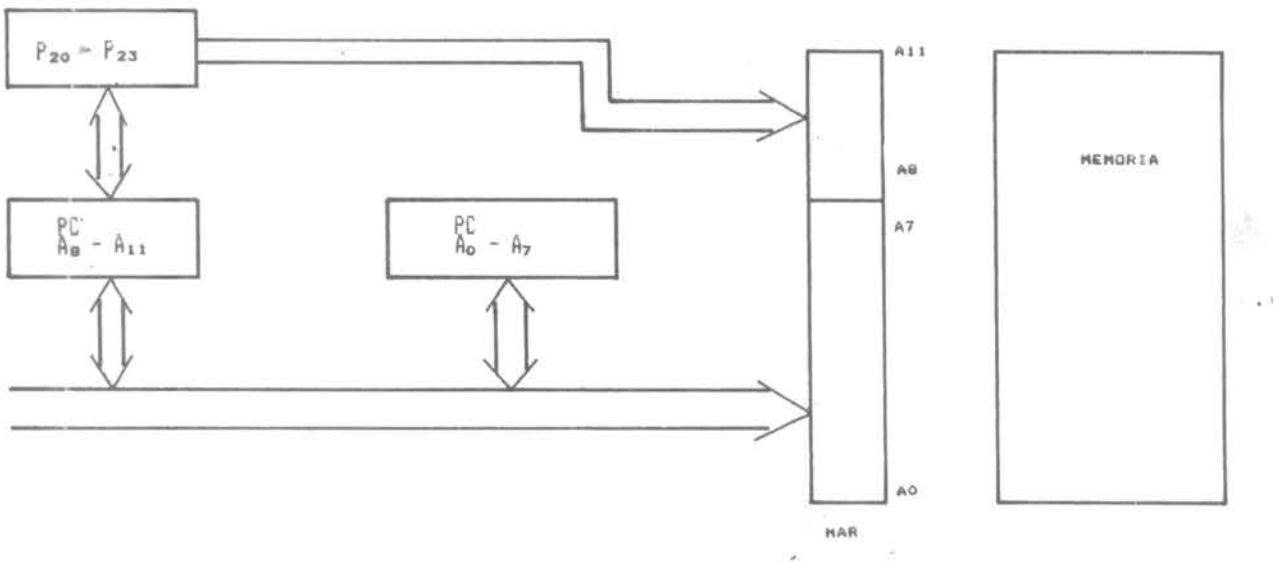


Figura 2-22. Registro contador de programa y organización del bus de direcciones

El contador de programa es un registro independiente, pero la memoria de datos donde el se guarda en el momento de una interrupción o salto a una subrutina, está hecha de registros que pertenecen a la memoria de datos del microprocesador; esta memoria de datos está constituida por 64 registros de 8 bits cada uno y es interna al microprocesador. Dentro del set de instrucciones del 8035 existen instrucciones específicas para leer y escribir datos en esta memoria. Adicionalmente la memoria de datos esta organizada de tal forma de disponer de dos bancos de registros lo cual será tratado en detalle cuando se describa la memoria de datos en secciones posteriores; por lo pronto en la figura 2-23 se muestra la organización de la memoria de datos del 8035 lo cual nos permitirá explicar algunos puntos referentes a el PC y al stack.

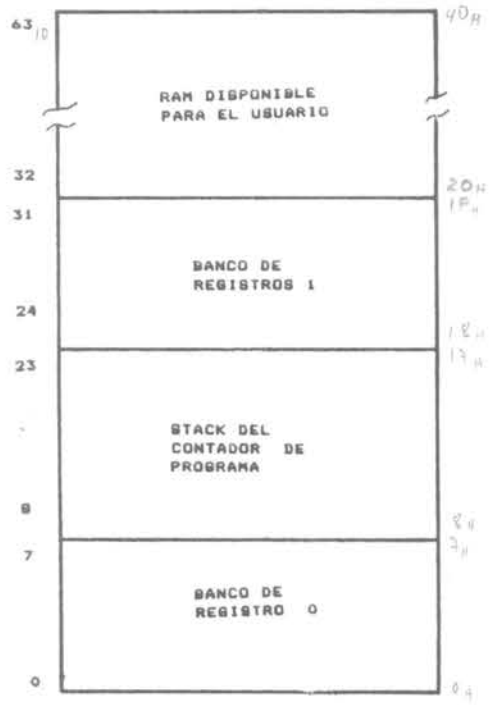


Figura 2-23. Organización de la memoria de datos del 8035

La memoria de datos tiene disponibles desde la localidad 8 hasta la 23 para el stack.

Estas localidades son usadas para almacenar los 12 bits del PC y los 4 bits mas significativos de la PSW tal como se muestra en la figura 2-24. Cuando el apuntador del stack vale 000_2 apunta a las localidades 8 y 9; el primer salto a subrutina o interrupción hace que el contenido del contador de programa se transfiera a las localidades 8 y 9 del arreglo de RAM. El apuntador de stack sera incrementado en uno y apuntará a las localidades 10 y 11 anticipandose así a la siguiente llamada o interrupción.

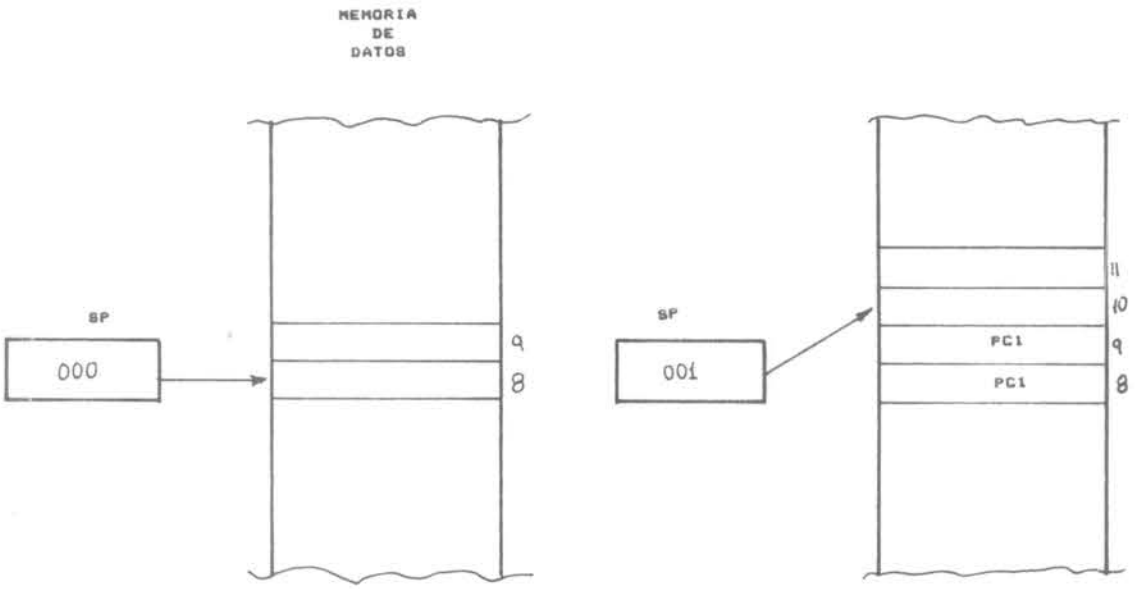


Figura 2-24. Organización y operación del stack

El anidamiento de subrutinas es permitido en el 8035, es decir una llamada a subrutina dentro de una subrutina, aceptandose un máximo de 8 niveles; en caso de sobrepasar la capacidad de llamadas a subrutinas dentro de una subrutina, ocurrirá un overflow del stack, perdiendose la información almacenada en las direcciones mas profundas del mismo, en este caso las direcciones 8 y 9 de la memoria de datos del microprocesador. En la figura 2-25 se muestra la situación en la cual se excede la capacidad del stack, en cuyo caso se destruye el contenido de las localidades 8 y 9, que contenía el valor del PC de la primera llamada, perdiendose en consecuencia la dirección apropiada de retorno.

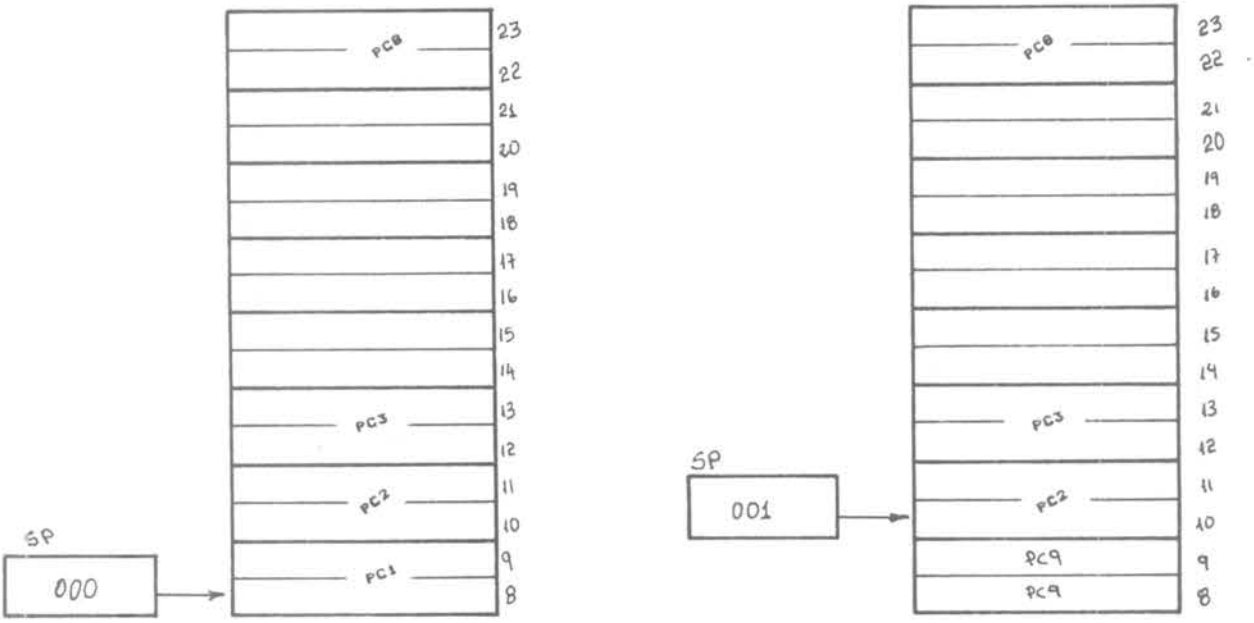


Figura 2-25. Overflow del stack por exceder capacidad de llamadas a subrutinas

El final de una subrutina, indicado por la instrucción RET o RETR, causa que el contenido del par de registros apuntados por el apuntador de stack sean transferidos al PC y a la PSW y se decremente el apuntador en uno.

En resumen una interrupción y/o una llamada a subrutina causan que el contenido del PC y parte de la PSW sean salvados en uno de los ocho pares de registros del stack. El par de registro que será usado es determinado por el apuntador del stack el cual como se mencionó anteriormente siempre apunta a la proxima dirección libre del stack. El apuntador de stack está compuesto por tres bits, los cuales forman parte de la PSW (palabra de estatus de programa).

2.2.2.5. Registro de Estatus de Programa (PSW)

La palabra de estatus de programa (PSW) que describe el estado del programa está constituida de 8 bits tal como se muestra en la figura 2-26; el uso de cada uno de los bits de la PSW se especifican a continuación.

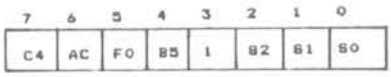


Figura 2-26. Organización de la palabra de estatus de programa (PSW)

bits 0,1,2

Estos tres bits indican cual de las ocho localizaciones del stack del PC es la próxima disponible

bit 3

Este bit no tiene ningún uso siempre permanece en el estado uno lógico.

bit 4

Indica el banco de registro que se está usando en cualquier momento dentro de un programa.

bit 4 = 0 selecionado el banco de registro 0

bit 4 = 1 seleccionado el banco de registro 1

bit 5

Este bit denominado F0 es activado o desactivado por el usuario a voluntad con las instrucciones CLR F0 y CPL F0. Se pueden hacer saltos condicionales en base al estado de este bit.

bit 6 (AC)

Este es el bit de acarreo auxiliar AC, es usado para ajuste decimal en operaciones con números codificados en BCD.

bit 7 (CY)

Este es el bit de acarreo el cual como se mencionó anteriormente indica la ocurrencia o no de un overflow en la operación realizada por la ALU.

2.2.3. ENTRADA/SALIDA DEL 8035

El microprocesador 8035 tiene 27 líneas de ENTRADA/SALIDA. Estas líneas están agrupadas en: 2 puertos (puerto 1 y 2) cuasibidireccionales de 8 líneas cada uno, los cuales, pueden ser usados como entradas, salidas o como puertos bidireccionales; un bus de datos de 8 bits bidireccional y que viene a constituir un tercer puerto del micro. Las otras tres líneas restantes (T0, T1 e $\overline{INT}$) son usadas como entradas y se pueden examinar desde programa con las instrucciones de salto condicional, pudiendo alterar de este modo la secuencia del programa. En la figura 2-27 se muestra un esquema de los puertos de E/S del 8035 y su conexión al bus interno.

2.2.3.1. Puertos de E/S 1 y 2

Los puertos 1 y 2 son de 8 bits y sus características son idénticas. Cuando operan como puertos de salida los datos que se sacan por el puerto permanecen almacenados y sin cambiar hasta que se saca otro dato por el mismo puerto; si se usan como puertos de entrada no tienen capacidad de almacenamiento, por lo tanto, las señales en las entradas deben mantenerse hasta que sean leídas por el microprocesador por medio de una instrucción de lectura de puerto. Las entradas de estos puertos son totalmente compatibles con los de la familia TTL y cuando se usan como salidas pueden manejar una sola carga TTL estandard.

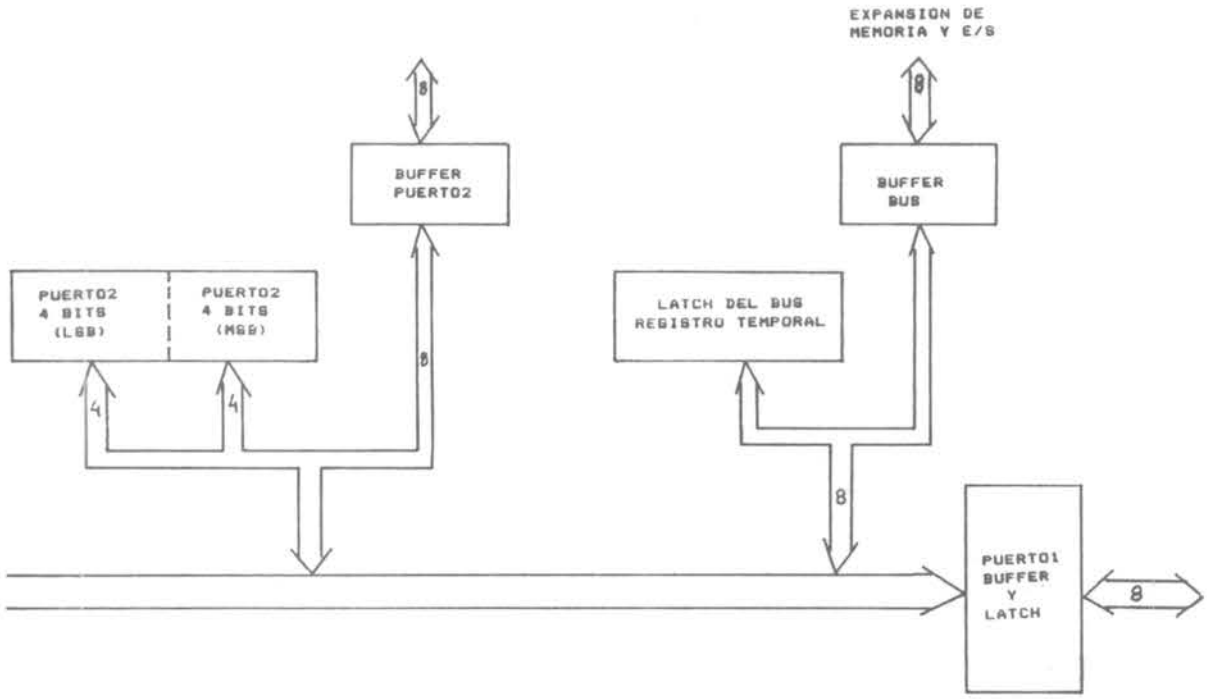


Figura 2-27. Puertos del 8035

El puerto 2 del 8035 tiene un uso muy especial; dicho puerto puede ser usado de 3 formas diferentes: como un puerto estático cuasibidireccional operando en este caso en forma idéntica al puerto 1; como un puerto para conexión del EXPANSOR DE PUERTOS 8243, el cual, es un miembro de la familia MCS-48 que permite dotar al 8035 de 4 puertos de 4 bits cada uno y cuya operación será descrita en detalle en el capítulo cuatro; y por último el puerto 2 se usa tambien para direccionamiento de memoria externa ya que por el sacan los cuatro bits mas significativos del contador de programa (PC_{11}-PC_8), cuando se realiza la fase de busqueda de la instrucción.

Cuando el puerto 2 se usa como puerto de salida, los datos sacados por el mismo no son afectados por la utilización como parte del bus de direcciones; por ejemplo si un dato es sacado por el puerto 2 con una instrucción de escritura en puerto, al momento de entrar en la fase de busqueda de la próxima instrucción, "automáticamente" por el puerto dos (por sus bits menos signicativos P_{20-23}) se saca parte de la dirección de memoria donde se encuentra dicha instrucción (los bits mas significativos de la dirección PC_{8-11}), sin embargo, el dato que se encontraba previamente en el puerto no se pierde ya que el mismo vuelve a tomar sus valores anteriores, al terminar la fase de captación de la instrucción

En el caso de estar usando el puerto dos para conexión

del expansor de puertos 8243, la información previamente almacenada en el puerto es removida al momento de ejecutar la fase de busqueda de la instrucción y no se restaura. Despues de una operación de entrada desde el 8243, los bits P_{20-23} quedarán en el modo de entrada (flotante). Despues de una operación de salida por el 8243 los bits P_{20-23} contendrán el valor escrito o bien el resultado de la operación AND u OR del puerto 2 y el puerto indicado por la instrucción.

2.2.3.2. BUS del 8035

El bus, a diferencia de los puertos 1 y 2, es realmente un puerto bidireccional con señales de control que controlan la entrada y la salida. Si la característica bidireccional del bus no se requiere, el mismo se puede usar como un puerto estático de salida con latch, o como un puerto de entrada sin latch igual que los puertos 1 y 2.

Como puerto estático el bus funciona de la siguiente manera: la instrucción OUT L hace que se escriba y se almacene en los latch del bus el dato; la instrucción INS hace que se lea un dato del bus. Las instrucciones INS y OUT L generan las señales de control $\overline{RD}$ y $\overline{WR}$.

Como puerto bidireccional la instrucción MOVX (move external) se usa para leer o escribir al puerto. Cuando no se está leyendo ni escribiendo en el puerto(BUS), las líneas del mismo están en estado de alta impedancia.

Las instrucciones OUTL y MOVX pueden mezclarse si es necesario, sin embargo, un dato previamente almacenado en el bus como consecuencia de la instrucción OUTL se destruirá al ejecutarse la instrucción MOVX y el bus quedará despues de esta instrucción, en el estado de alta impedancia

Entradas T0, T1, $\overline{INT}$

Estas señales fueron descritas ya cuando se vió la sección de control del 8035 en donde se mencionó los usos de las mismas muy especialmente en la parte de instrucciones de salto condicional.

2.2.4. MEMORIA DE PROGRAMA Y MEMORIA DE DATOS

En el microprocesador 8035 se distinguen dos tipos de memoria; una interna compuesta de 64 palabras llamada memoria de datos y una externa que puede crecer hasta 4096 palabras llamada memoria de programa.

2.2.4.1. Memoria de Datos

La memoria de datos consta de 64 palabras de 8 bits cada una; viene integrada en el mismo chip que constituye el microprocesador. La figura 2-28 es una representación esquemática de la organización de la memoria de datos del 8035.

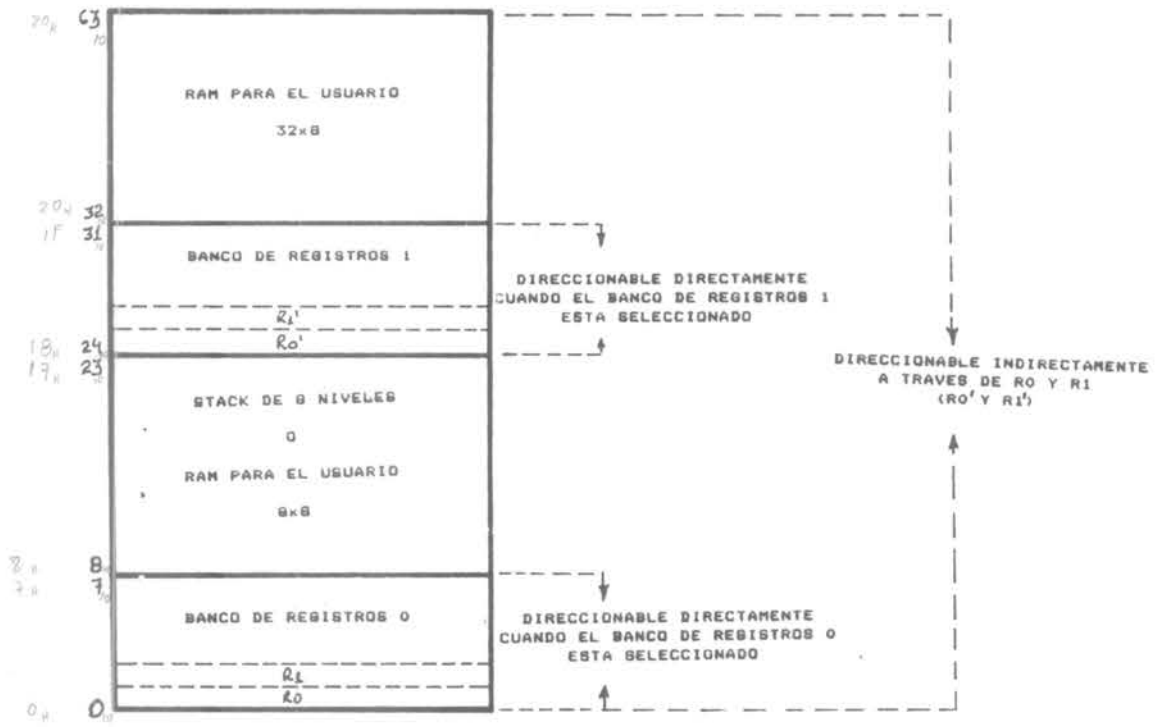
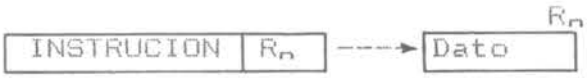


Figura 2-28. Organización de la memoria de datos del 8035

El 8035 dispone de dos posibles modos de accesar esta memoria de datos: el MODO DIRECTO y el MODO INDIRECTO.

Modo Directo

En este modo de direccionamiento la instrucción que se va a ejecutar contiene el identificador del registro en el que se encuentra el dato sobre el cual se va operar, tal como se muestra a continuación:



Modo de Direccionamiento Indirecto

En este modo la instrucción contiene el identificador del registro donde está la dirección del dato, tal como se muestra a continuación:

A continuación dos ejemplos para ilustrar ambos modos de direccionamiento:

Ejemplo 1 MODO DIRECTO

Mov A,R_0 mueva el dato del registro 0 al acumulador

Antes de ejecutar la instrucción

Despues de ejecutar la instrucción

UNIVERSIDAD DE CARABOBO

FACULTAD DE INGENIERIA

CURSO DE EDUCACION CONTINUA SOBRE

MICROPROCESADORES

Trabajo presentado ante el Ilustre Consejo de Facultad de Ingeniería por los Profesores Hyxia Villegas y Fredy Jara, para ascender a la categoría de Profesor Agregado.

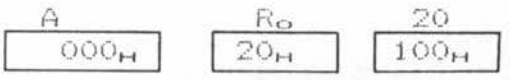
ING. HYXIA VILLEGAS
ING. FREDY JARA

DICIEMBRE 1984

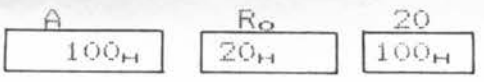
Ejemplo 2 MODO INDIRECTO

MOV A, @R_0 Mueva el dato apuntado por el registro 0 al Acumulador

Antes de ejecutar la instrucción



Despues de ejecutar la instrucción



La memoria de datos del 8035 está dividida en:

a. Banco de registros 0: localidades del 0 al 7.

b. Banco de registros 1: localidades del 24 al 31.

c. RAM para uso general: localidades del 32 al 63.

d. Stack: localidades del 8 al 23.

a. Banco de registros 0

Las localidades de 0 a 7 conforman el banco de registro 0; este banco, compuesto por los registros R_0 hasta R_7, son directamente direccionables a través de varias instrucciones que operan con ellos. Como estos registros son mas accesibles que el resto de la memoria de datos, ellos son usados mas frecuentemente para guardar resultados intermedios.

La instrucción DJNZ R_n (decremente y salte si no es cero el registro n) permite usar cualquiera de los registros del banco 0 como contador en un lazo de control, con la ventaja de decrementar y chequear el registro en una sola instrucción

b. Banco de registros 1

Este segundo banco de registros puede ser usado como una extensión del banco de registros 0.

Los subprogramas, los cuales se explicaran mas adelante, son la solución ideal para implementar la solución de problemas.

El programa principal como ya se dijo, se dividirá en pasos o etapas, que pueden ser implementadas como subprogramas, con un grado de complejidad menor del que se hubiese logrado con un solo programa.

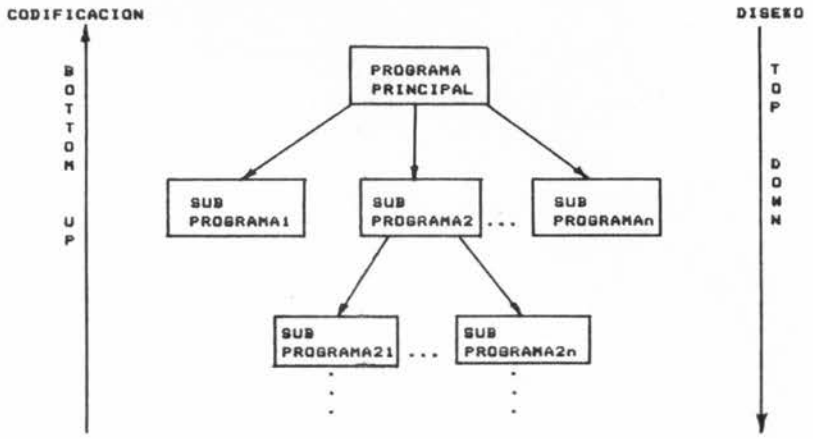


Figura 3-5. Ilustración de la programación estructurada

Cada subprograma se dividirá a su vez en otros subprogramas menos complejos y con mas detalle y asi se continuará hasta llegar al mas bajo nivel, que nos permita codificar directamente. Se empezará a codificar entonces de abajo hacia arriba "botton-up", de lo mas sencillo a lo mas complejo, lo cual implica introducir los detalles de la codificación al final.

3.2.2. DIAGRAMAS DE FLUJO

Algoritmo, es el procedimiento desarrollado paso a paso para realizar una determinada tarea; siempre esta relacionado con hacer tareas complejas que se dividen en una secuencia de tareas sencillas.

El 8035 posee instrucciones especiales que permiten seleccionar uno u otro banco como los registros de trabajo del programa en ejecución; asi al ejecutar la instrucción SEL RB1 (seleccione el banco de registros 1) se estarán designando las localidades de memoria de datos desde la 24 a la 31 como los registros de trabajo en lugar de las localidades de 0 al 7. En el caso de estar seleccionado el banco 1 las localidades que ahora son directamente direccionables son las ubicadas entre la 24 y la 31, ambas inclusive. Las dos primeras localizaciones de este banco de registros se identifican como R_0' y R_1'.

Todas las localidades de la memoria de datos son indirectamente direccionables a través de los registros R_0 y R_1, cuando está seleccionado el banco 0, y con los registros R_0' y R_1', cuando está seleccionado el banco de registros 1; estos mismos registros pueden ser usados para direccionar indirectamente 256 palabras de memoria externa, pudiendo tener hasta cuatro apuntadores indirectos a memoria simultaneamente.

El banco de registros 1 es muy usado en el caso de rutinas de servicio para la atención de interrupciones; en estos casos se acostumbra guardar los resultados del programa principal en el banco de registros 0, y la subrutina de servicio o atención de la interrupción seleccionará como banco de trabajo el 1, ejecutando como primera instrucción de la misma SEL RB1. Cuando el banco de registros 1 no se usa como tal, las localidades del 24 al 31 pueden usarse como direcciones de RAM de propósito general.

c. RAM de proposito general

Las localidades de la memoria de datos desde la posición 32 a la 63 pueden usarse como localidades de memoria RAM para propósito general; estas localidades son indirectamente direccionables por medio de los dos primeros registros de cualquiera de los dos bancos como ya se mencionó anteriormente.

d. Stack

Las localidades de la 8 a la 23 de la memoria de datos tienen una doble función: primero como localidades de memoria RAM para propósitos generales y segundo como stack para salvar la dirección de retorno al programa principal cada vez que se llame a una subrutina tal como se explicó en la sección correspondiente a la unidad de control. Estas localidades son direccionadas por el apuntador de stack durante las llamadas a subrutinas y por R1 y R0 indirectamente (tambien por sus homólogos del banco 1).

Si hay menos de 8 llamadas a subrutinas, una dentro de la otra sin retornar, es decir, si el nido de subrutinas es menor de 8, el resto de estas localidades puede ser usado como RAM de propósito general. Cada nivel de stack no usado provee al usuario de dos localidades de RAM, ya que como se vió anteriormente se requieren de 16 bits para salvar el estatus del programa (PC y PSW).

2.2.4.2. Memoria de Programa

El espacio de memoria que el 8035 puede direccionar o accesar es de 4096 palabras, expresado este espacio en kilopalabras (1 Kilopalabra = 1024 palabras) se diría que el 8035 es capaz de direccionar 4 Kilopalabras.

La memoria de programa está dividida en dos bancos de memoria de 2 Kilopalabras cada uno tal como se muestra en la figura 2-29. El banco de memoria 0 que va desde la dirección 0 hasta la 2047 (000 a 7FF) y el banco de memoria 1 el cual va desde la dirección 2048 a la 4095 (800 a FFF). El usuario podra seleccionar desde el programa uno u otro banco usando la instrucción SEL MB0 o SEL MB1, dependiendo del banco escogido; esta instrucción realmente lo que hace es apagar o prender el bit 11 del contador de programa, al momento de ejecutar una instrucción de salto o una llamada a subrutina. No seleccionar apropiadamente el banco de registro es una fuente comun de error en los programas.

La memoria externa puede ser accesada de dos formas: primero cuando el microprocesador necesita captar una instrucción que está apuntada por el PC, en cuyo caso se colocarán en el bus de los 8 bits menos significativos del PC y los 4 bits mas significativos, que corresponderian a la página saldrán por los bits menos significativos del puerto 2. En este caso el término página está siendo usado para indicar 256 posibles localidades de memoria; la página mas baja corresponde a la 0 y la mas alta a F_H o sea un total de 16 páginas de 256 palabras cada una.

La figura 2-30 ilustra esta forma de acceso a la memoria externa.

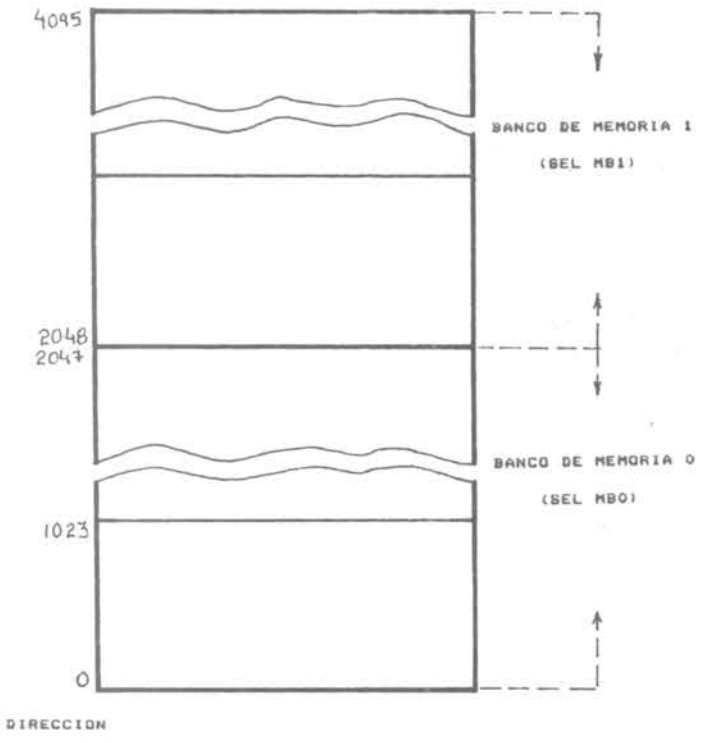


Figura 2-29. Organización de la memoria de programa del 8035

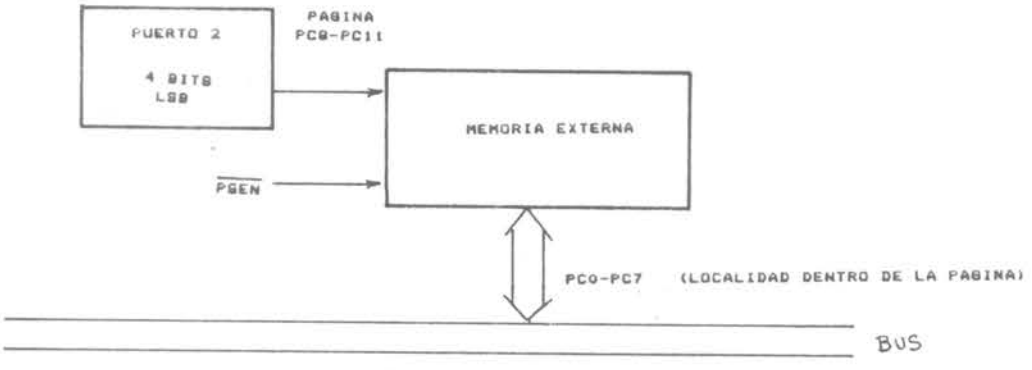


Figura 2-30. Acceso a memoria externa en la fase de captación

La segunda forma ocurre cuando el usuario desde programa desea direccionar un dato almacenado en una posición de memoria externa; en estos casos el usuario deberá sacar la página correspondiente por los bits menos significativos del puerto 2 y la posición o dezplazamiento dentro de la página deberá sacarse

por el bus proveniente de alguno de los registros R_0 o R_1. Es de hacer notar que en el primer caso la dirección es colocada automáticamente en el bus y el puerto 2 sin intervención del programador, en cambio en el segundo caso, es responsabilidad del programador que esto ocurra; a continuación un ejemplo que ilustra la manera de hacer el acceso a memoria en la segunda modalidad.

Ejemplo: Cargue el acumulador con el dato contenido en la localidad 8 página 7 de la memoria de programa externa

```
MOV R0, 08    carga en R0 el dezplazamiento
                  dentro de la página

MOV A, 07         carga en  A la página

OUTL P2,A         saca la página por P2

MOVX A,@R0    carga A con el dato que está en
                  la página 7  desplazamiento 8
```

Hay tres localidades de memoria de especial significación ellas son (ver figura 2-31):

Localidad 0

A esta posición de memoria apunta el contador de programa al activarse la línea de RESET del 8035; en otras palabras la próxima instrucción a ejecutar despues de un Reset será la que se encuentre en la dirección 0 de memoria de programa.

Localidad 3

A esta posición apuntará el contador de programa despues de haberse activado la línea de interrupción externa ($\overline{INT}$) con el sistema de interrupción habilitado. En otras palabras, la próxima instrucción a ejecutar despues de una señal de interrupción válida será la que se encuentre en la localidad 3 de memoria de programa; independientemente del dispositivo periférico que interrumpa (en el caso de haber varios), el salto por interrupción externa siempre será a la posición 3 de memoria.

Localidad 7

Una interrupción del Timer/Counter del 8035, como consecuencia de overflow de su contador, tambien causa un salto a una posición específica de memoria que para estos casos es la localidad 7 de memoria de programa. El sistema de interrupción debe estar habilitado y la primera instrucción despues de esta tipo de interrupción será buscada en la posición 7.

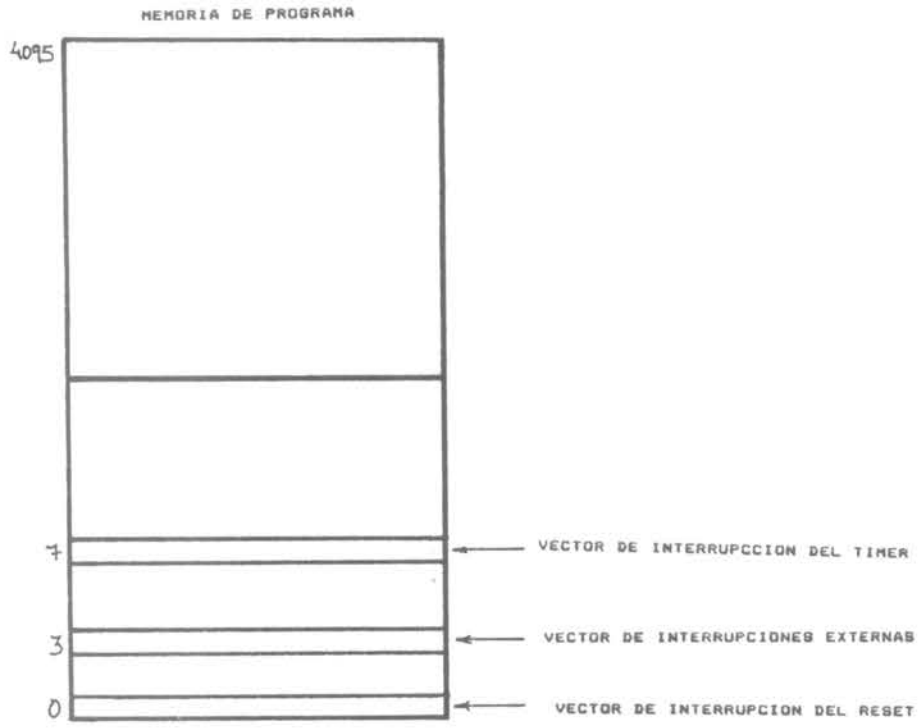


Figura 2-31. Ubicación de los vectores de interrupción

El orden de prioridades en que se atienden estas tres interrupciones que se acaban de mencionar es: en primer lugar el RESET, en segundo lugar la interrupción externa ($\overline{INT}$) y en tercer lugar la interrupción por overflow del timer/counter.

2.2.5. OTRAS CARACTERISTICAS

Como característica adicional del 8035 se puede mencionar el hecho de que el mismo posee un timer interno y las instrucciones necesarias para inicíalizarlo, arrancarlo y detectar overflow tal como lo muestra la figura 2-31

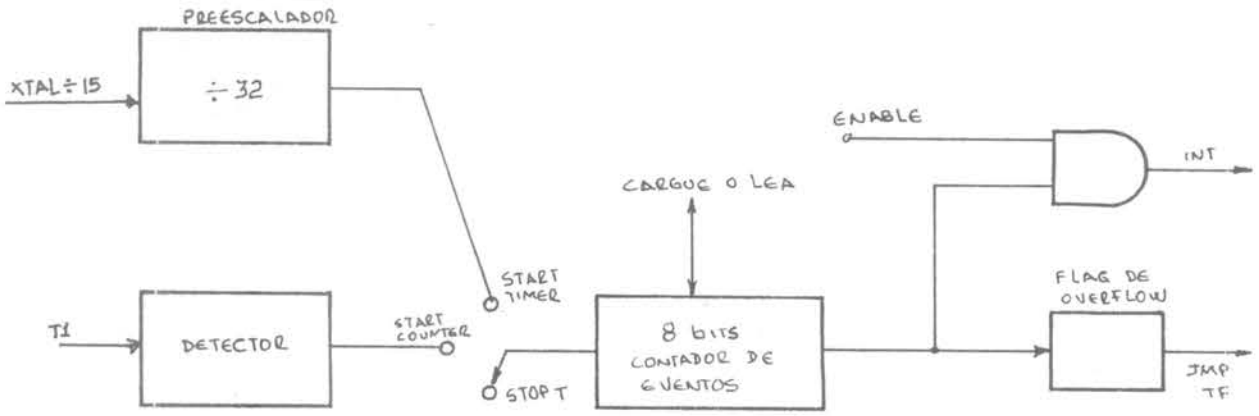


Figura 2-32. Timer del 8035

Este contador puede ser usado por el usuario para contar eventos externos, evitando así el tener que usar el procesador en esta tarea, o como timer si usa el reloj del microprocesador.

Uso como Contador

El contador ascendente de 8 bits se puede inicializar o leer con dos instrucciones, que permiten transferir el contenido del açumulador a el contador y viceversa; estas instrucciones son: MOV A,T para cargar el contador en el acumulador y MOV T,A para inicializar el contador con el valor contenido en el acumulador.

El contenido del contador no es afectado por el RESET y solo puede ser inicializado con la instrucción MOV T,A. El contador detiene su cuenta si se hace un RESET o si se ejecuta la instrucción STOP TCNT permaneciendo detenido indefinidamente hasta que se arranque de nuevo con la instrucción START CNT, en cuyo caso, comienza a funcionar como un contador de eventos. Una vez que el contador es arrancado empieza a incrementarse con cada evento hasta que llega a su máximo valor FF_H; el próximo evento ocasionará que se genere un overflow, activandose en este caso el flag de overflow generando una señal de interrupción. El estado del flag de overflow del timer/counter se puede examinar con la instrucción JTF; el flag de overflow se pone automáticamente en el estado cero despues de ejecutarse la

instrucción JTF.

Para que la interrupción que genera el timer/counter sea atendida, es necesario que esté habilitado el sistema de interrupción interno; el sistema de interrupción interno puede ser habilitado o deshabilitado con dos instrucciones que trae el microprocesador 8035 ellas son EN TCNTI para habilitarlo y DIS TCNTI para deshabilitarlo. En el caso de estar habilitado, la señal de interrupción por overflow causará una llamada a subrutina, debiendo estar la primera instrucción de esta subrutina en la dirección 7 de memoria de programa.

Uso como Timer

La ejecución de la instrucción START T, conecta la entrada del contador a un reloj interno y activa el contador. El reloj interno se deriva de pasar la señal de reloj ALE por un preescalador que la divide entre 32. Con este timer se pueden obtener retardos desde 80 microsegundos hasta 20 milisegundos, el máximo y el mínimo retardo depende de la frecuencia del cristal que se esté usando en las entradas XTAL1 y XTAL2.

C A P I T U L O I I I

3.1. LENGUAJES DE PROGRAMACION

Se ha mencionado en los capítulos anteriores que los microprocesadores (o cualquier computador digital), operan bajo control de un programa; dicho programa esta compuesto por un conjunto de intrucciones. Se ha mencionado tambien que las únicas instrucciones que en definitiva entiende e interpreta el microprocesador son las que consisten de combinaciones de digitos binarios (1 y 0); cada microprocesador posee un conjunto de combinaciones de unos y ceros (instrucciones), que el es capaz de interpretar, para producir o ejecutar una acción especifica.

Existen varias alternativas en la forma en la que se pueden escribir los programas o programar los microprocesadores, una de ellas sería escribir los programas directamente como combinaciones de unos y ceros, lo cual equivaldría a programar en lenguaje de máquina; otra alternativa sería escribir los programas en un lenguaje mas facil de entender y manipular por los humanos y traducir o convertir posteriormente, con la ayuda del mismo microprocesador, el programa escrito en dicho lenguaje a lenguaje de máquina. Al respecto se han desarrollado una serie de lenguajes que van desde el ASSEMBLER, el cual es un lenguaje de programación en mnemonicos, intimamente relacionado con la máquina, hasta los lenguajes llamados de alto nivel, que son independientes de la máquina. En esta primera parte de este capítulo se estudiaran brevemente los lenguajes mas populares en los sistemas a microprocesadores, comenzando por el ASSEMBLER.

3.1.1. LENGUAJE ASSEMBLER

La escritura de programas en lenguaje de máquina suele ser muy complicada, en virtud de la dificultad en recordar la serie de combinaciones que componen una instrucción y a la probabilidad elevada de cometer errores, en razon de la cantidad de combinaciones que hay que manejar (cada instrucción la constituyen un conjunto de unos y ceros), las cuales tienden a confundirse a la hora de escribir un programa.

Los diferentes lenguajes de programación que se han

implementado se basan en el desarrollo de programas que permiten trasladar o convertir de un determinado lenguaje a lenguaje de máquina. El primer avance que se hizo en los lenguajes de programación, fue el realizado con el ASSEMBLER. El mismo consiste en una serie de mnemónicos que tienen una correspondencia uno a uno con cada instrucción de máquina (combinaciones de unos y ceros), en otras palabras, el conjunto de instrucciones usados para escribir un programa en lenguaje ASSEMBLER es idéntico al conjunto de instrucciones usados, para escribir dicho programa en lenguaje de máquina, la diferencia esencial estriba en el hecho de usar mnemónicos en vez de combinaciones de ceros y unos. Por esta razon el ASSEMBLER no se considera un lenguaje de alto nivel, aun cuando el tiene varias características, aparte del uso de mnemónicos, que facilitan la escritura de los programas. Las características mas resaltantes del ASSEMBLER son:

a) Uso de mnemónicos en lugar de combinaciones de unos y ceros para representar las instrucciones.

b) Uso de etiquetas o labels en vez de direcciones absolutas, lo cual facilita y simplifica la modificación de los programas, sobre todo cuando se usan instrucciones de salto.

c) Posibilidad de usar macroinstrucciones en la elaboración de los programas, que permite "definir" un conjunto de instruciones como una nueva, que puede ser usada en cualquier parte del programa; en donde se coloque la macroinstrucción, el ensamblador automáticamente coloca el conjunto de instrucciones que se definieron como tal.

Los mnemónicos en general son abreviaturas de las instrucciones o de la acción o función que ellos realizan, lo cual permite recordarlas facilmente.

La escritura de programas en assembler y su posterior introducción y ejecución en un determinado microprocesador requieren en primer lugar de un teclado normal, en virtud de que los mnemónicos estan formados por las letras del alfabeto y por números; y en segundo lugar de la existencia de un programa que al ejecutarlo, convierta a lenguaje de máquina, los simbolos del lenguaje ASSEMBLER. El conjunto de instrucciones o mnemónicos constituyen lo que se denomina PROGRAMA FUENTE. El programa encargado de hacer la conversión de mnemónicos a lenguaje de máquina se llama ensamblador; el mismo posee una tabla interna donde estan los mnemónicos y su equivalente en binario. El programa ensamblador va instrucción por instrucción "barriendo" el programa fuente y traduciendo las instrucciones a lenguaje de máquina, generando un programa secundario en lenguaje de máquina, equivalente al escrito en lenguaje ASSEMBLER, este programa es llamado PROGRAMA OBJETO.

Para producir un programa objeto es necesario alimentar

al computador con dos programas: El programa fuente, escrito con los mnemonicos del ASSEMBLER y el programa ensamblador, que usará como datos de entrada los mnemonicos del programa fuente y producirá como salida, los codigos o instrucciones equivalentes en lenguaje de máquina, que constituiran el programa objeto, tal como se resume en la figura 3-1.

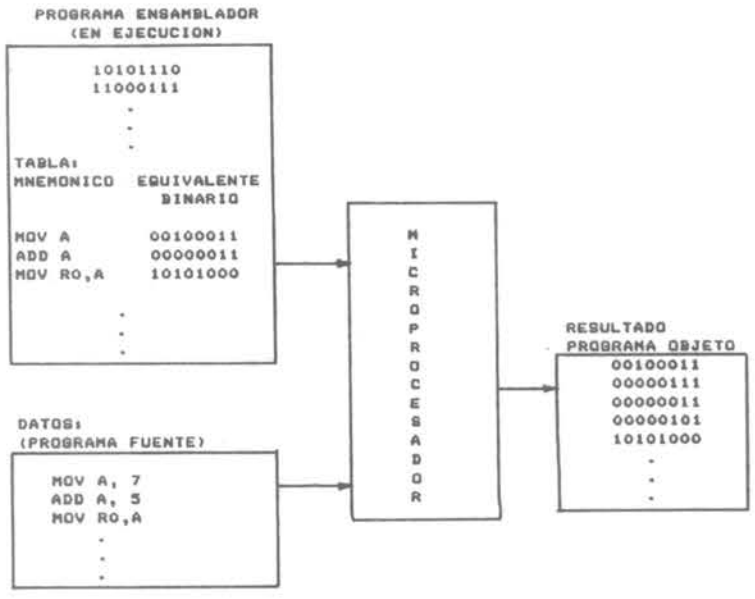


Figura 3-1. Generación de un programa objeto a partir del fuente escrito en ASSEMBLER

El lenguaje ASSEMBLER depende del computador para el cual fue escrito, esto se debe a que el tiene una relacion 1 a 1 con el lenguaje de máquina, el cual maneja los recursos de hardware del computador; ejemplo de ello es la instrucción MOV Rd,A, la cual, implicitamente indica la existencia en el microprocesador o computador, de un registro llamado Rd y otro llamado A.

3.1.2. LENGUAJES DE ALTO NIVEL

Los lenguajes de alto nivel hacen mas facil aun la programación de los microprocesadores y computadores. Estos lenguajes de alto nivel son independientes de la máquina y un programa escrito en uno de estos lenguajes es basicamente igual para cualquier computador; cada máquina tendrá su propio programa para trasladar los programas escritos en el lenguaje de alto nivel, a su respectivo lenguaje de máquina.

Los lenguajes de programación de alto nivel mas ampliamente usados en los microprocesadores son: BASIC (Beginners

Algebraic Symbol Interpreter Compiler), FORTRAN (Formula Translation), COBOL (Common Business Oriented Language), y mas recientemente el PASCAL.

El BASIC es sumamente utilizado por hobistas de la computación y principiantes. En la siguiente tabla se muestran los simbolos aritméticos y la operación que representan en el lenguaje BASIC, con ejemplos que ilustran su utilización:

SIMBOLO	OPERACION	EJEMPLO
+	sume	X+Y sumele X a Y
-	reste	X-Y reste Y de X
*	multiplique	X*Y multiplique X veces Y
/	divida	X/Y divida X entre Y
	exponenciación	X 2 X al cuadrado (X^2)
SQR	raiz cuadrada	SQR(X) toma la raiz de X

Una expresión comun en Basic es:
LET C = (A * B) + 3 sea C igual al producto de A por B mas 3.

El FORTRAN tiene su maxima aplicación en el campo cientifico; en la siguiente tabla se veran los simbolos aritméticos, la operación que realizan y un ejemplo de su utilización:

SIMBOLO	OPERACION	EJEMPLO
+ -	suma	X + Y suma X a Y
-	resta	X - Y resta Y de X
*	multiplicación	X * Y multiplica X veces Y
/	división	X/Y divide X entre Y
**	exponenciación	X ** 2 eleva al cuadrado X
SQRT	raiz cuadrada	SQRT(X), saca la raiz de X

Una expresión comun en FORTRAN es
C = (A * B) + 3; Asignele a C el producto de A por B mas 3.

El COBOL es un lenguaje comercial hecho a proposito para programar reportes, inventarios, nominas, etc. A continuacion una tabla que ilustra algunas de las instrucciones comunes en COBOL:

OPERACION	INSTRUCCIONES TIPICAS
ADD	ADD Factura TO Balance
GIVING	ADD Factura TO Balance GIVING Total
SUBSTRACT	SUBSTRACT Costo FROM Ventas GIVING Gan
MULTIPLY	MULTIPLY Total BY 0.5 GIVING Impuesto
DIVIDE	DIVIDE Total BY Kilos GIVING Costpkilo

Ejemplos comparativos con los tres lenguajes:

FORTRAN F = M+ N

BASIC LET F = M + N

COBOL ADD Manzana TO Naranjas GIVING Frutas

Problema: encontrar el area de un circulo de radio 2.

FORTRAN	BASIC	COBOL
R = 2	LET R = 2	RADIO IS 2
A = 3.14*(R**2)	PRINT R,3.14*R 2	MULTIPLY Radio TIMES Radio GIVING Rac
PRINT 40,R,A		MULTIPLY Rac times 3.14 GIVING AREA
40 FORMAT I2,I4		WRITE area

Con todos estos lenguajes de alto nivel se producen

programas fuentes, los cuales les sirven de entrada a su compilador respectivo y este genera el programa en lenguaje de máquina de manera similar a como se vio en el assembler; Sin embargo, existe una diferencia con el programa en assembler y el programa escrito en lenguaje de alto nivel: a cada instrucción en Assembler le corresponde una instrucción en lenguaje de máquina, mientras que cada instrucción en lenguaje de alto nivel, corresponde a varias instrucciones en lenguaje de máquina.

Para la ejecución de un programa escrito en lenguaje de alto nivel se requieren basicamente dos pasos: en un primer paso se ejecuta el programa compilador, que usa como dato el programa fuente escrito en el lenguaje de alto nivel que corresponda al compilador, para generar el programa objeto en lenguaje de máquina, el cual si es ejecutable en el microprocesador. En el segundo paso se ejecuta el programa objeto generado por el compilador, con los datos que corresponda a dicho programa, generandose los resultados del procesamiento de los datos tal como se resume en la figura 3-2.

Figura 3-2. Etapas en la ejecución de un programa escrito en lenguaje de alto nivel

3.2. TECNICAS DE PROGRAMACION

3.2.1 PROGRAMACION POR APROXIMACIONES SUCESIVAS O ESTRUCTURADA.

Primero se define el problema y empleando el método "TOP-DOWN", del todo hacia las partes, se produce la estructura.

El problema (el todo), se descompone en subproblemas (las partes), los cuales, pueden ser subdivididos a su vez en problemas mas pequeños, tal como se muestra en la siguiente figura.

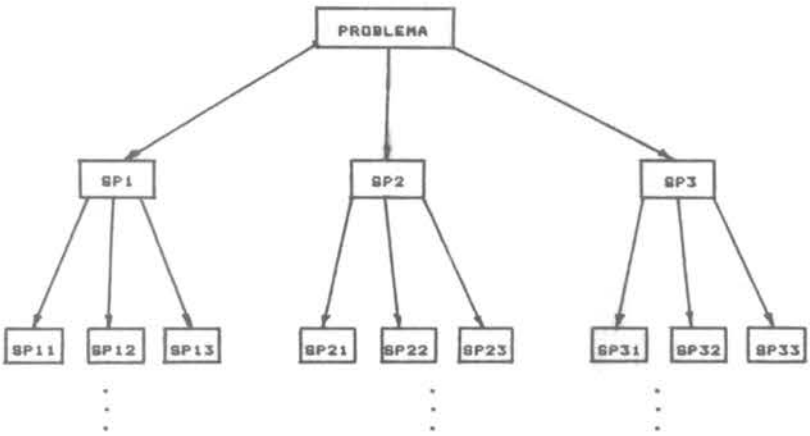


Figura 3-3. Descomposición de un problema en subproblemas.

Al final, estos subproblemas mas pequeños seran faciles de codificar.

En el desarrollo de la estructura se debe pensar primero ¿que hacer para resolver el problema?, lo cual produce los algoritmos y segundo ¿como hacerlo?, lo cual indica el detalle de la solución; este detalle debe ser introducido al final y no al comienzo, pues de lo contrario el programador se dejará envolver por los detalles y perderá la visión general del todo.

El principal reto es limitar la complejidad de el problema en cada paso dado. Cada paso adicional, es entonces la elaboración de un paso previo, con mayor detalle y complejidad. Un proceso de estructuración genera muchos beneficios tanto en terminos de esfuerzo de la solución del problema, como en

terminos de la programación en si. A continuación un resumen de los principales beneficios de la estructuración de los problemas:

1. Cada paso es independiente de otros pasos, lo cual permite probar aisladamente cada paso.

2. Cada paso puede ser probado etapa por etapa a traves del proceso de estructuración, desde la definición de la tarea hasta la codificación del programa.

3. Los errores se detectan en forma sistemática.

4. En cualquier punto, solo una pequeña cantidad de información debe ser recordada y manipulada por el programador.

5. Las estructuras ayudan a probar rigurosamente si el algoritmo es correcto.

El diseño TOP-DOWN es facil de aplicar cuando se desarrollan programas en grupos. El jefe del proyecto puede definir la tarea y definir las distintas fases, dirigiendo los nombres del grupo sobre algoritmos separados y ellos a su vez podran llegar hasta la codificación de las tareas . Esta forma de atacar los problemas trae las ventajas antes mencionadas.

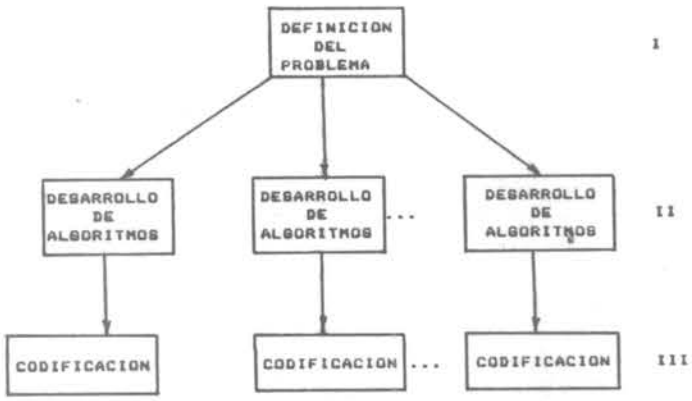


Figura 3-4. Diseño TOP-down.

Los subprogramas, los cuales se explicaran mas adelante, son la solución ideal para implementar la solución de problemas.

El programa principal como ya se dijo, se dividirá en pasos o etapas, que pueden ser implementadas como subprogramas, con un grado de complejidad menor del que se hubiese logrado con un solo programa.

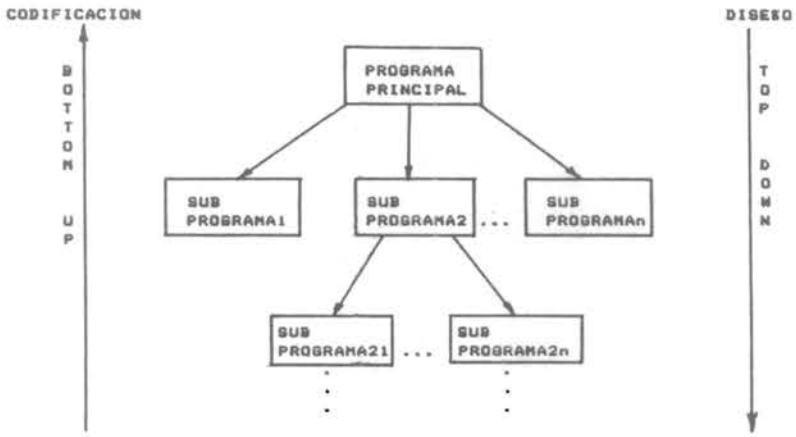


Figura 3-5. Ilustración de la programación estructurada

Cada subprograma se dividirá a su vez en otros subprogramas menos complejos y con mas detalle y asi se continuará hasta llegar al mas bajo nivel, que nos permita codificar directamente. Se empezará a codificar entonces de abajo hacia arriba "botton-up", de lo mas sencillo a lo mas complejo, lo cual implica introducir los detalles de la codificación al final.

3.2.2. DIAGRAMAS DE FLUJO

Algoritmo, es el procedimiento desarrollado paso a paso para realizar una determinada tarea; siempre esta relacionado con hacer tareas complejas que se dividen en una secuencia de tareas sencillas.

En el caso de algoritmos de computación, se tiene que el computador por si solo no es capaz de resolver ningun problema, el programador deberá por lo tanto, producir sus algoritmos en donde le especifique, paso a paso, como deberá resolver el problema.

Por ejemplo, en la implementación de un programa para sumar dos números, se le deberá indicar al computador:

1. La lectura del primer número
2. La lectura del segundo número
3. Operación a realizar con los números (suma)
4. Escritura del resultado

La forma usada en programación para representar los algoritmos es a traves de diagramas de flujo. El diagrama de flujo representa el procedimiento que debera seguir el computador para resolver un determinado problema. Un diagrama de flujo que representa la solución al problema de sumar dos números se muestra a continuación:

Figura 3-6. Diagrama de flujo de un programa

En este momento se hace importante recordar lo siguientes puntos tratados:

1. El computador no resuelve problemas. el programador de computadoras es el que lo hace.

2. El programador producirá la estructura de programación para resolver el problema, la cual generará los algoritmos que seran representados en diagramas de flujo, los cuales a su vez mostraran paso a paso el procedimiento que el computador seguirá.

3. El proceso de desarrollar el procedimiento que el computador seguirá es llamado programacion.

4. La secuencia de instrucciones que se producen cuando el diagrama de flujo es traducido a instrucciones del computador (codificado) se llama PROGRAMA.

Para la realización de los diagramas de flujo deben seguirse una serie de convenciones en los simbolos utilizados, adicionalmente, el diagrama de flujo debe ser ordenado. La figura 3-7 muestra los simbolos convencionalmente usados en la elaboración de los diagramas de flujo.

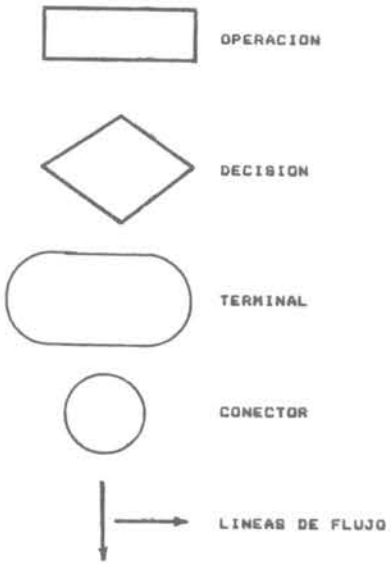


Figura 3-7. Simbolos empleados en la elaboración de diagramas de flujo

A continuación se dará una breve descripción de los usos que se le da a cada simbolo en los diagramas de flujo:

Simbolo terminal: se usa para indicar el inicio y el fin del diagrama de flujo.

Caja de operación: se usa para indicar un proceso, una operación, una asignación, etc..

Ejemplos:

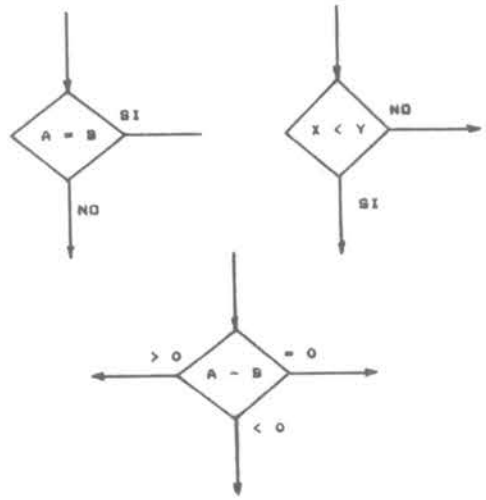
S = A + B

M = A x B

A = 15

Diamante de decisión: El diamante de decisión se usa para representar una condición que se desea examinar o chequear, para tomar una decisión en base al estado de esa condición. No existe una regla específica para la colocación de los posibles valores de la condición chequeda, pudiendo colocarse la misma, en cualquiera de los lados del diamante.

Ejemplos:

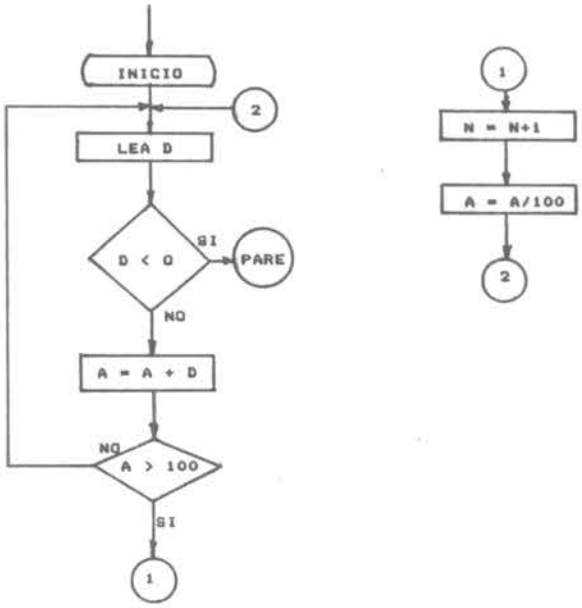


Líneas de flujo: se usan para indicar la secuencia a seguir y el orden en que se realizaran las operaciones.

Conectores: se usan para empatar porciones de diagramas

de flujo; si un diagrama de flujo no cabe en una página, para continuarlo en otra página, se emplean los conectores para enlazar las dos secciones; de la misma manera, si se require saltar a una porción determinada del diagrama de flujo, con la finalidad de no complicar el trazado de las líneas de flujo, se pueden emplear conectores para estos propositos.

Ejemplo



A continuación se desarrollará un ejemplo de un programa que permitirá ilustrar el uso de las tecnicas y herramientas descritas en los parrafos anteriores. Cuando se lluegue a la etapa de codificación, se usaran los mnemónicos del 8035, y como sistema de procesamiento se usará el sistema IMSAI.

El sistema IMSAI, es un sistema de desarrollo basado en el 8035, que será descrito ampliamente en el capítulo IV; por lo pronto, solo interesa saber que dicho sistema posee un teclado hexadecimal, un display alfanumérico y un monitor con una serie de rutinas que permiten entre otras cosas, el manejo del teclado y el display. No se dispone en el sistema IMSAI de un programa ensamblador, por lo cual, aun cuando en la codificación de los programas se esten usando mnemónicos, a la hora de introducir esta codificación en el sistema, se hará usando los codigos hexadecimales equivalentes a estos mnemónicos.

PROBLEMA

Sacar el promedio de N números. El número de datos (N) y los números mismos, seran introducidos por el teclado; el resultado deberá ser mostrado por el display.

Se desarrollará la estructura empleando el método de diseño Top-Down; se empezará con una definición de la tarea, a partir de lo cual se desarrollará el algoritmo, y se terminará con la codificación del programa.

Este proceso de descomposición de arriba hacia abajo, ayuda de una forma natural a generar codigo estructurado.

Primera aproximación: definición de las tareas

Figura 3-8. Primera aproximación para resolver el problema de la suma de N números

Ahora se hará la segunda aproximación, desarrollando cada uno de estos bloques.

Segunda aproximación de "LEA LOS N NUMEROS": se introduce el detalle de preguntar si el valor de N es cero, para evitar una división por cero; la figura 3-9, es una representación de esta segunda aproximación.

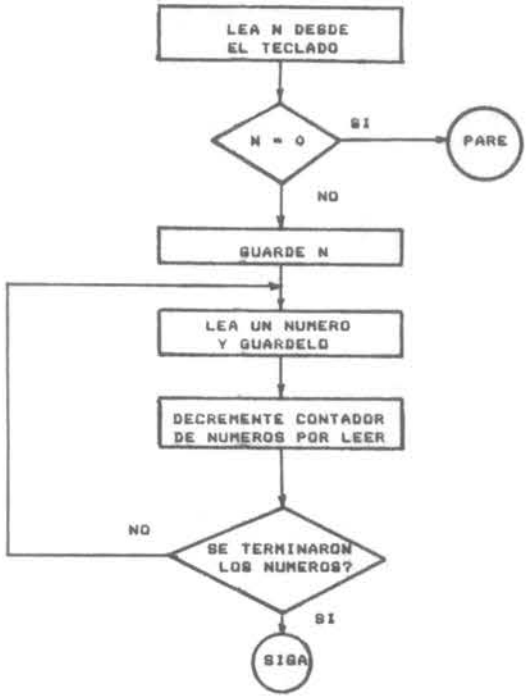


Figura 3-9. Segunda aproximación para resolver la lectura de N números

Ahora se hará una tercera aproximación para resolver cada uno de los bloques de la segunda aproximación.

La parte que corresponde a la lectura de N y su almacenamiento esta practicamente en la etapa de codificación; el valor de N se guarda en los registros R0 y R1 (se pudo guardar en cualquier otro), para resguardar su valor para su utilización posteriormente; la figura 3-10 constituye el diagrama de flujo de esta tercera aproximación.

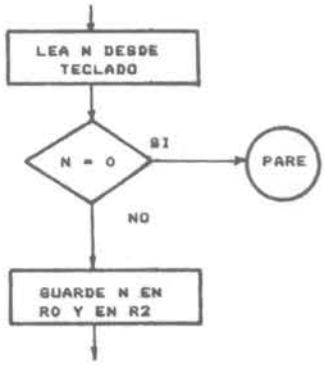


Figura 3-10. Tercera aproximación, lectura y almacenamiento de N

El segundo bloque que se desarrollará en esta tercera aproximación es la lectura de un número desde el teclado y su almacenamiento. La figura 3-11 muestra el diagrama de flujo para este bloque en una tercera aproximación.

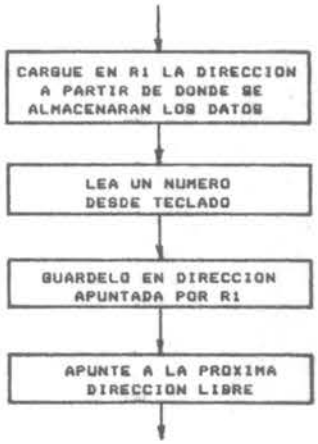


Figura 3-11. Tercera aproximación para leer un número desde teclado y guardarlo

El algoritmo final para el desarrollo del primer bloque del diagrama de la figura 3-8, se muestra en la figura 3-12; a partir de este diagrama de flujo referente a la lectura de N números desde el teclado, la codificación es directa, como se verá mas adelante. La figura 3-12 es el diagrama de flujo definitivo del primer bloque: "LEA N NUMEROS DESDE TECLADO".

Ahora se procederá al desarrollo de la segunda aproximación para el bloque que corresponde a la suma de los N números. En la figura 3-13 se muestra el diagrama de flujo del algoritmo para sumar los N números leidos.

FIGURA 3-12. Algoritmo final para lectura de los N números

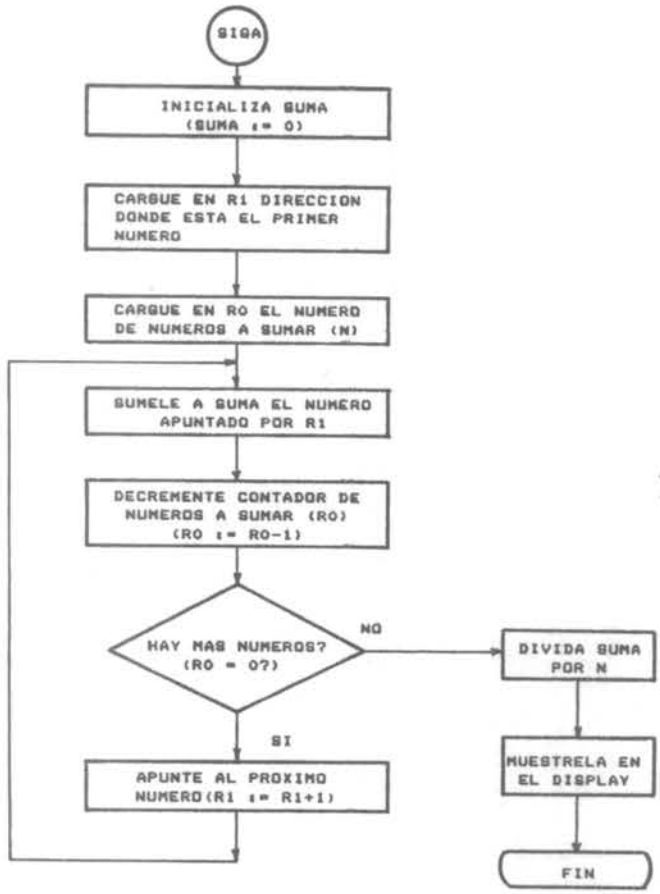


Figura 3-13. Segunda aproximación para resolver la suma de los N números

A continuación se procedera a desarrollar el bloque de división por N; en virtud de que el microprocesador que se usará no tiene en su conjunto de instrucciones una que permita la realización de la división, será necesario implementar un algoritmo para dividir. Se usará el algoritmo de restas sucesivas para determinar el cociente.

El algoritmo que se implementará solo dará el resultado correspondiente a la parte entera; la parte fraccional será siempre cero; por tanto, en el caso de que la suma de los N números sea menor que el número de datos, el promedio calculado por este algoritmo será cero. La figura 3-14 constituye el diagrama de flujo de la segunda aproximación del algoritmo de división por N.

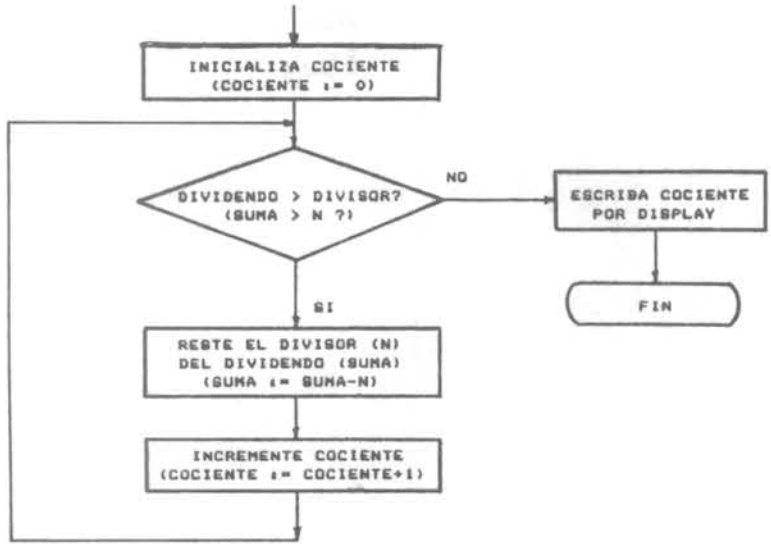
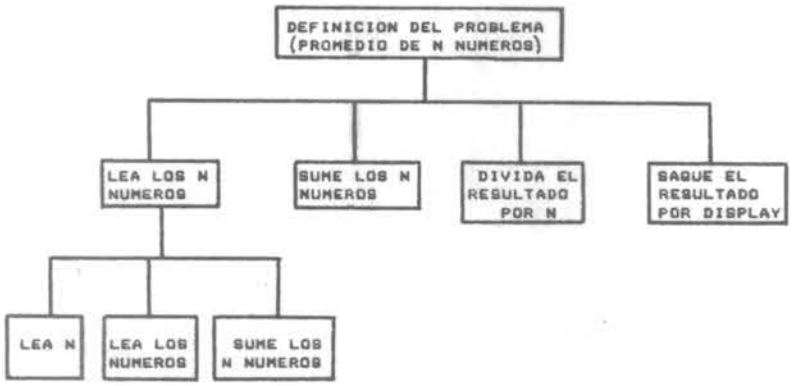


Figura 3-14. División por N mediante restas sucesivas

El bloque que corresponde a sacar por el display el valor del promedio de los N números (cociente), no requiere ningun desarrollo adicional, ya que el sistema que se está usando trae ya implementadas rutinas especiales para mostrar por el display al igual que para leer desde el teclado. La codificación del diagrama de flujo desarrollado para la solución del problema de sacar el promedio de N números, se dejará para despues que se haya tratado el set de instrucciones del 8035, sin embargo, se hace notar el hecho de que a partir de este diagrama de flujo la codificación es directa. A continuación un diagrama general de la estructura de programación generada por el problema y las distintas aproximaciones que se hicieron para resolverlo.



3.2.3. TECNICAS DE PROGRAMACION EN ASSEMBLER

En esta sección, se explicaran las estructuras que normalmente conforman los programas, y de las que se dispone a nivel de instrucción, en los lenguajes de alto nivel. Es el caso de las instrucciones WHILE, DO UNTIL, FOR, CASE; cuando se programa en lenguaje assembler, no se dispone de instrucciones de este tipo en forma directa, ya que como se mencionó en líneas anteriores, el assembler tiene una correspondencia uno a uno con las instrucciones de la máquina y estas estructuras requieren de varias instrucciones para su realización; sin embargo, es una excelente tecnica de programación en assembler, desarrollar estas estructuras, bien sea como una subrutina o una macroinstrucción, y usarlas en la elaboración de programas en assembler,

Un programa se reduce a una combinación de las siguientes estructuras: SECUENCIA, RUPTURA DE SECUENCIA e ITERACION.

En la estructura de SECUENCIA, las instrucciones se ejecutan una detras de la otra, en el mismo orden en fueron escritas. Un programa que solo posea esta estructura, es limitado, y solo usa una porción del poder del microprocesador o computador.

La estructura de RUPTURA DE SECUENCIA, tiene su basamento en las instruccciones de saltos condicionales; estas instrucciones permiten cambiar la secuencia de ejecución del programa, en base a una determinada condición chequeada. Las estructuras de ruptura de secuencia mas conocidas se muestran en la figura 3-15.

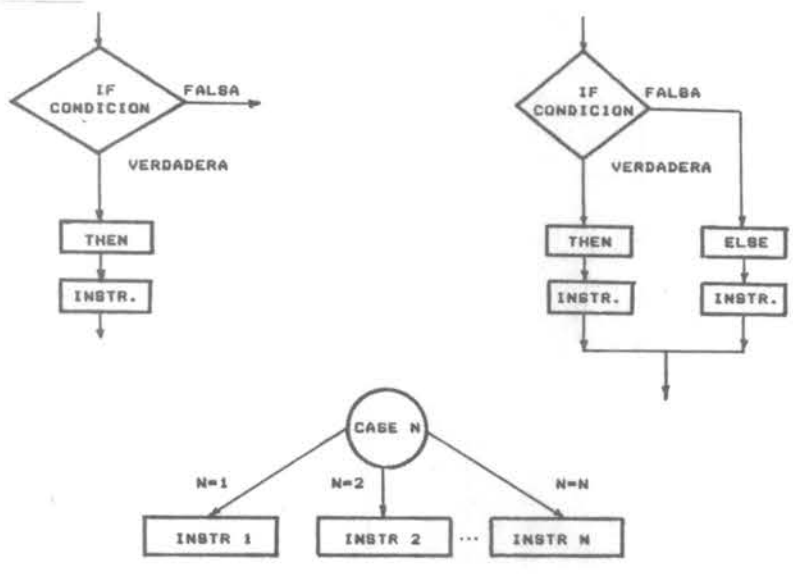


Figura 3-15. Estructuras de ruptura de secuencia

En la estructura de ITERACION, se hace uso de la habilidad de los computadores y microprocesadores de ejecutar una sección de programa tantas veces como sea requerido. El lazo iterativo puede estar controlado por un indice o por una condición cualquiera; la figura 3-16 muestra los lazos iterativos mas comunes.

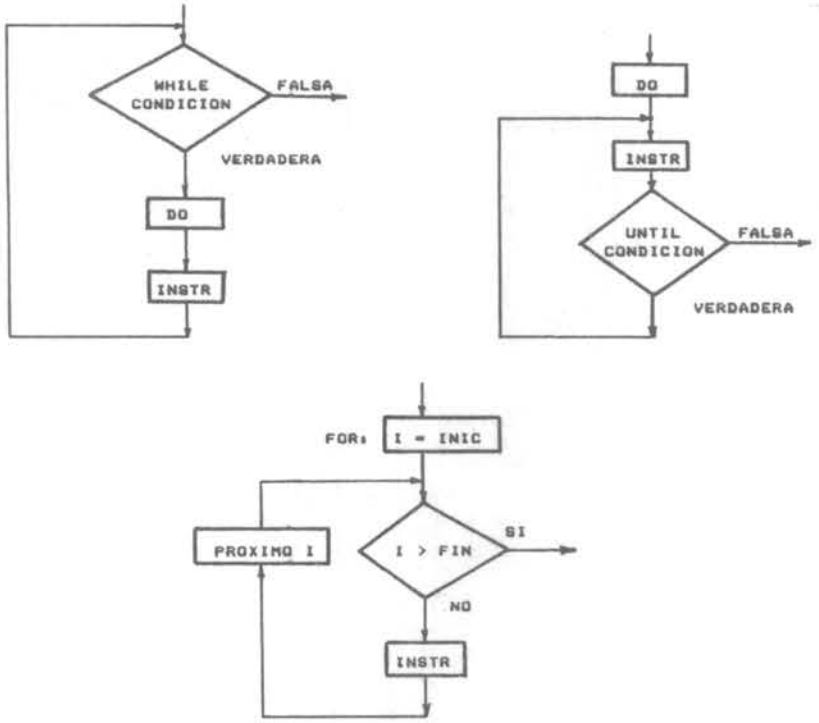


Figura 3-16. Estructuras de iteración

En las estructuras de iteración, es de hacer notar la diferencia entre el lazo de WHILE y el de DO-UNTIL en el primero, se pregunta por la condición antes de ejecutar la instrucción o bmoque de instruccines del lazo; en el segundo, se ejecuta primero la instrucción o bloque de instrucciones dentro del lazo iterativo, y se chequea posteriormente el estado de la condición, de manera que, independientemente del estado inicial de la condición, la instrucción o instrucciones dentro del lazo, se ejecutan por lo menos una vez. En ambos casos, la condición chequeada debe ser susceptible de cambio dentro del lazo, pues de lo contrario se estaría en un lazo infinito.

El lazo iterativo FOR esta controlado por un indice, el cual se le da un valor inicial, que se incrementa en un valor constante establecido, en cada paso por la instrucción o onstrucciones dentro del lazo, hasta alcanzar un valor final preestablecido; en este caso, si el valor inicial del indice supera el valor final previsto, la instrucción o instrucciones

dentro del lazo, no se ejecutan ninguna vez.

3.2.4. SUBRUTINAS

Cuando una misma tarea es realizada en distintos lugares en un programa, es usualmente indeseable repetir el grupo de instrucciones en cada sitio del programa principal. Una tecnica que se ha desarrollado para evitar esta situación, es agrupar este conjunto de instrucciones que realizan una tarea definida, en una unidad separada del programa principal, llamada subrutina o subprograma; aun cuando esta subrutina realiza una tarea específica, normalmente no puede ser ejecutada por si sola sino que debe ser invocada o llamada por un programa (de allí el nombre de subprograma). Un ejemplo típico de subrutina lo constituye la que permite calcular la raíz cuadrada de un número; al disponer de una subrutina que permita realizar este cálculo, en cualquier sección del programa principal puede ser invocada o llamada, con solo usar la instrucción correspondiente que permite la llamada a subrutina, segun el lenguaje que se esté usando para programar.

Cuando se invoca o llama una subrutina, se transfiere el control del programa principal a la subrutina, y el microprocesador o CPU, comienza a ejecutar el conjunto de instrucciones que constituyen la subrutina, en tanto el programa principal, o sea, el programa que invocó a la subrutina, se suspende hasta terminarse de ejecutar la misma. La transferencia de control del programa principal a la subrutina, implica que se debe salvar el valor del contador de programa, el cual apunta a la proxima instrucción a ejecutar, o sea la instrucción que sigue a la de llamada a la subrutina y que constituirá la dirección de retorno desde la subrutina. Al terminarse de ejecutar la subrutina, el control debe transferirse de nuevo al programa principal, el cual proseguirá su ejecución normal a partir de la instrucción que sigue a la que invoca a la subrutina. Cuando se ejecuta la llamada a la subrutina, el contador de programa se carga con la dirección donde comienza la subrutina.

La transferencia de control del programa a la subrutina y viceversa se le denomina ENLACE DE SUBRUTINA. El enlace se realiza: del programa a la subrutina, cuando se ejecuta la llamada a la subrutina, lo cual hace, en primer lugar, que se guarde la dirección de retorno al programa principal, constituida por el valor del contador de programa en el momento de la llamada, y en segundo lugar que se cargue en el contador de programa la dirección de inicio de la subrutina. Y el enlace subrutina programa principal, el cual hace que se restaure el valor del contador de programa, es decir que se cargue en el mismo, la dirección de retorno al programa principal; el retorno desde una subrutina requiere de la ejecución de una instrucción especial, en FORTRAN por ejemplo, se usa la instrucción RETURN, en assembler se usa la instrucción RET o RETR.

Cada máquina tiene su forma peculiar de hacer los enlaces a subrutinas, usualmente tienen instrucciones especiales relacionadas con esta función. En el 8035, segun se estudió en el capítulo II, se resuelve el enlace de subrutinas, mediante el uso del STACK; cada vez que en un programa aparece una llamada a subrutina (JSB), el 8035 salva el valor del contador de programa (el cual apunta a la proxima instrucción a ejecutar en el programa) y el estatus de los flags, en el stack; La ejecución en la subrutina de una instrucción de retorno de subrutina (RET o RETR), causan que se haga un POP al stack, cargandose el contador de programa con la dirección de retorno que se había almacenado en el mismo al momento de la llamada a la subrutina.

Ventajas del uso de subrutinas

Las subrutinas son usadas para ahorrar espacio en memoria; las subrutinas no permiten ahorrar tiempo de ejecución del programa, sin embargo, si permiten ahorrar tiempo por el hecho de no tener que cargar varias veces un mismo conjunto de instrucciones y ahorrar tiempo de compilación, pues el compilador o programa emsamblador, tendrá que analizar y traducir menos instrucciones. Es de hacer notar el hecho de que las subrutinas introducen un tiempo extra en la ejecución del programa por el hecho de tener que hacer el enlace de subrutina, sin embargo, este tiempo extra es en la mayoria de las veces compensado por las otras ventajas originadas del uso de las subrutinas y que veremos a continuación.

-Las subrutinas son la base de la programación estructurada, ya que permiten dividir un problema complejo en subproblemas mas simples.

-Facilita la corrección de programas, pues se hace posible probar cada subprograma por separado, y despues que se está seguro que funciona se usa en cualquier parte de un programa.

-Elimina la duplicidad de esfuerzos de programación, pues una subrutina hecha para un determinado propósito, puede ser usada por cualquier programa que la invoque.

-Permite ahorrar memoria al evitar la duplicidad de código.

-Facilita la comprensión y modificación de los programas.

-Permite enrriquecer el repertorio básico de instrucciones, al permitir implementar funciones que pueden ser usadas por cualquier programa, como si fuese una instrucción del CPU (ejemplo: extracción de raiz cuadrada, cálculo del seno de un ángulo, división, etc.).

-Facilita la realización de proyectos en equipos.

-Permiten la utilización de un algoritmo sin tener que conocer exactamente como funciona. Es el caso por ejemplo de las subrutinas KEYINP y DISPRG que trae el programa monitor del IMSAI, las cuales se usan para leer desde el teclado y mostrar por el display respectivamente; en estos casos, no se necesita conocer "el como lo hacen", sino "el que hacen".

por último con el fin de resaltar la gran utilidad e importancia del uso de la subrutinas se hace la siguiente cita.

"En el diseño de software, especialmente si se trata de un sistema grande y complejo, es crucial reducir la cantidad de detalle que va a ser considerado en cada momento. Una gran ayuda en este sentido es la idea de abstracción implicada por los subprogramas. En el nivel al cual uno "llama" al subprograma, uno separa el detalle importante del "que hacer?", de los detalles irrelevantes, del "como hacerlo?". De manera semejante, al nivel cuando se implementa el subprograma, es a menudo innecesario complicar el "como?", con consideraciones de "por que?"; es decir, las razones exactas para invocar un subprograma, no tienen por que importar siempre a quien lo implemente. Como los subprogramas pueden llamar a otros subprogramas, uno puede desarrollar toda una gerarquia de abstracciones"

GUTTAG (CACM 20(6) : 397)

Como una buena técnica de programación cuando se usan subrutinas en assembler, se recomienda que al hacer un salto a una subrutina, las primeras instrucciones de la subrutina, tengan por finalidad salvar los valores de los registros que usará la subrutina y restaurarlos posteriormente, antes de retornar al programa principal; de esta manera, el programa principal no se verá afectado.

En el microprocesador 8035, se pueden salvar los registros que el programa principal este usando y que vayan a ser utilizados o afectados por la subrutina que se llame, haciendo uso de los bancos de registros; generalmente, el programa principal usa los registros del banco de registros 0 y si hay una llamada a subrutina, la misma operará con el banco de registros 1; esta forma de operar cuando se usan subrutinas, es muy util y simplifica grandemente el problema de resguardar los registros; sin embargo, algunas veces se hace necesario guardar adicionalmente el acumulador, el cual no es salvado por el simple hecho de cambiar de banco de registros. En cualquier caso, independientemente de que se use el 8035 y sus facilidades para el manejo de este tipo de problema, el hecho importante que es necesario tener presente, es que si hay algun registro que se use en el programa principal y que la subrutina tambien requiera usarlo, el contenido del mismo debera ser salvado, bien por el programa principal antes de llamar a la subrutina, o implementar

la subrutina de manera tal, que la misma salve los registros que ella usará, y los restaure antes de retornar el control al programa principal; con esta última forma, el usuario que llama la subrutina no tiene que preocuparse mucho por los registros que usa la subrutina en su ejecución, pues los mismos son salvados y restaurados por la propia subrutina.

Otra aspecto importante de tratar, es la forma en que se le pasan los parámetros o datos a la subrutina y la forma en que la misma devuelve los resultados de la manipulación de estos datos. Cuando se programa en lenguaje assembler o en lenguaje de máquina, el programador tiene que decidir la forma en que le pasará los datos a la subrutina; es muy comun pasar datos o parámetros a una subrutina vía los registros, es decir, si por ejemplo se esta desarrollando una subrutina para determinar el menor de dos numeros, el programador puede implementar la subrutina de manera que al llamarla, la misma busque los dos números en unos registros específicos del microprocesador; será responsabilidad del programa que llame a la subrutina, colocar en dichos registros los dos números requeridos por la subrutina; por otra parte, la respuesta de la subrutina, en el caso del ejemplo, el menor de los dos números, deberá ser colacada en algun sitio en el cual el programa principal la pueda accesar y usar; este sitio puede ser de nuevo un registro específico del microprocesador, o una localidad específica de memoria. Existe tambien la posibilidad de usar el direccionamiento indirecto para pasar parámetros a las subrutinas y en estos casos, en vez de colocar en los registros los datos propiamente dichos, se colocan en los mismos las direcciones de memoria donde estan los datos; en este caso el registro se esta usando como apuntador de el dato o datos.

3.3. SET DE INSTRUCCIONES DEL 8035

A continuación se estudiará el set de instrucciones del microprocesador 8035, para lo cual se han agrupado las distintas instrucciones en 5 tipos, segun el efecto y uso que tienen; estos son:

Instrucciones para manipulación de datos

Instrucciones para movimiento de datos

Instrucciones para manipulación de programa

Instrucciones para manipulación de estatus de programa

Instrucciones para manejo de la Entrada/Salida

Dentro de la categoría de instrucciones de manipulación

de datos se encuentran las instrucciones para operaciones aritméticas y lógicas, instrucciones para rotaciones e instrucciones para transferencias de paquetes de bits.

En la categoría de instrucciones para manipulación de programa se encuentran las instrucciones para llamadas a subrutinas, instrucciones de saltos condicionales e incondicionales etc..

En la categoría de instrucciones para movimiento de datos se encuentran las instrucciones para lectura y escritura de datos en memoria, con sus distintos modos de direccionamiento, las instrucciones para manipulación del stack, etc..

Dentro de la categoría de instrucciones de manipulación de estatus de programa se encuentran las instrucciones para modificar y leer los registros de acarreo, acarreo auxiliar, registro cero y registro de condición negativa; se encuentran tambien en esta categoría las instrucciones para la manipulación de la PSW (Prgram Status Word) e instrucciones de saltos condicionales en base al estado de los de estatus de programa.

A continuación se dará una descripción detallada de las distintas instrucciones del 8035.

3.3.1. INSTRUCCIONES PARA MANIPULACION DE DATOS

Este tipo de instrucciones tienen un formato específico, el cual describe su efecto y el modo de direccionar los operandos. El formato de estas instrucciones consta de un código de operación y de uno o dos operandos; en el caso de dos operandos, uno de los operandos se suele llamar operando fuente y el otro operando destino; el operando fuente no será afectado con la operación especificada, en cambio el operando destino si será afectado por dicha operación en virtud de que el resultado de la operación será almacenado en dicho operando. Por ejemplo en la instrucción ADD A,R6, el operando fuente es el registro R6 y el operando destino es el registro acumulador; despues de la ejecucion de la instruccion, el contenido del registro R6 permanece inalterado, en cambio el contenido del registro acumulador contiene ahora el valor resultante de la suma del dato que está en R6 con el dato que se encontraba en el acumulador, antes de ejecutar esta última instrucción (ADD A,R6).

La tabla siguiente resume todas las instrucciones del 8035 pertenecientes a este grupo de instrucciones. Observese que el primer grupo de instrucciones tienen como operando destino el acumulador, por lo que afectaran su contenido; en cambio en segundo grupo de instrucciones tiene como operando destino alguno de los registros, por lo que el contenido de estos registros se verá afectado por la ejecución de las mismas.

TABLA 3-1

INSTRUCCIONES DEL 8035 PARA MANIPULACION DE DATOS

MNEMONICO	CODIGO DE OPERACION	SEMANTICA	DESCRIPCION
ADD A,Rr	68-6F 68==>R0, 69==>R1... 6F==>R7	A:=A+Rr (r=0 a 7)	Contenido del registro Rr (r=0 a 7) es sumado al acumulador. El carry es afectado.
ADD A,@R0 @R1	60-61 60==>R0 61==>R1	A:=A+(Rr) (r=0 a 1)	El contenido de la memoria de datos apuntada por Rr se suma al acumulador.
ADD A,#dato	03 dato	A:=A+dato	Esta es una instrucción de dos bytes. Dato especificado en el segundo byte es sumado al acumulador.
ADDC A,Rr	78-7F 78==>R0 79==>R1... 7F==>R7	A:=A+Rr+C luego C:=0 (r=0 a 7)	El contenido del carry se agrega al bit cero del acumulador y luego se le suma el registro seleccionado (R0 a R7).
ADDC A,@Rr R0 R1	70-71 70==>R0 71==>R1	A:=A+(Rr)+C luego C:=0 (r=0 a 1)	El contenido del carry se agrega al bit cero del acumulador y luego el contenido de la memoria de datos apuntada por Rr se suma al acumulador.
ADDC A,#dato	13 dato	A:=A+dato+C luego C:=0	Esta es una instrucción de dos bytes. Carry se suma al bit cero del acumulador y luego el dato especificado en el segundo byte se suma al acumulador.
ANL A,Rr	58-5F 58==>R0, 59==>R1... 5F==>R7	A:=A AND Rr (r=0 a 7)	Hace un AND entre el dato del acumulador y el dato almacenado en el registro especificado.
ANL A,@Rr R0 R1	50-51 50==>R0 51==>R1	A:=A AND (Rr) (r=0 a 1)	Hace un AND entre el dato del acumulador y el dato almacenado en la

			memoria de datos apunta da por Rr.
ANL A,#dato	53 dato	A:=A AND dato	Esta es una instrucción de dos bytes. Hace un AND entre el acumulador y el dato especificado en el segundo byte.
ORL A,Rr	48-4F 48==>R0 49==>R1... 4F==>R7	A:=A OR Rr (r=0 a 7)	Hace un OR entre el acu mulador y el registro seleccionado(R0 a R7).
ORL A,@Rr R0 R1	40-41 40==>R0 41==>R1	A:=A OR (Rr) (r=0 a 1)	Hace un OR entre el acu mulador y el dato conte nido en la memoria de datos apuntada por Rr.
ORL A,#dato	43 dato	A:= A OR dato	Esta es una instrucción de dos bytes. Hace un OR entre el acumulador y el dato especificado en el segundo byte.
XRL A,Rr	D8-D7 D8==>R0, D9==>R1...	A:=A XOR Rr (r=0 a 7)	Hace un OR EXCLUSIVO en entre el dato contenido en el acumulador y el dato contenido en el re gistro Rr.
XRL A,@Rr R0 R1	D0-D1 D0==>R0 D1==>R1	A:=A XOR (Rr) (R=0 a 1)	Hace un OR EXCLUSIVO en tre el acumulador y el dato contenido en la me moria de datos apuntada por Rr.
XRL A,#dato	D3 dato	A:=A XOR dato	Esta es una instrucción de dos bytes. Hace un OR EXCLUSIVO entre el a cumulador y el dato es pecificado en el segun do byte.
INC A	17	A:=A+1	Incrementa acumulador en uno.
DEC A	07	A:=A-1	Decrementa acumulador en uno.
CLR A	27	A:=0	Pone todos los bits del acumulador en cero.
CPL A	37	A:=NOT A	Complementa todos los bits del acumulador.

DA A	57	si (A1-A3) >9 o AC=1 ==> A:=A+6 luego si (A7-A4) >9 o C=1 ==> (A7-A4):= (A7-A4)+6	Hace ajuste decimal del dato contenido en el acumulador. Si AC=1 o el nibble menos significativo del acumulador es mayor de nueve se le suma 6 al acumulador; se chequea luego el nibble mas significativo y si excede el valor nueve o C=1, se incrementa este nibble en 6; C se activa si hay overflow.
RLC A	F7	An+1:=An (n=0 a 6) A0:=C; C:=A7	Rota el acumulador a la izquierda a traves del carry.
RL A	E7	An+1:=An A0:=A7	Rota el acumulador a la izquierda.
RR A	77	An:=An+1 (n=0 a 6) A7:=A0	Rota a la derecha el acumulador.
RRC A	67	An:=An+1 (n=0 a 6) A7:=C; C:=A0	Rota a la derecha el acumulador a traves del carry.
SWAP A	47	A4-A7 A0-A3	Intercambia los nibbles del acumulador
INC Rr	18-1F	Rr:=Rr+1 (r=0 a 7)	Incrementa en uno el registro especificado.
DEC Rr	C8-CF	Rr:=Rr-1 (r=0 a 7)	Decrementa en uno el registro especificado.
INC @Rr R0 R1	10-11 10==>R0 11==>R1	(Rr):=(Rr)+1 (r=0 a 1)	Incrementa en uno el dato contenido en la memoria de datos apuntada por Rr(R0 o R1).

3.3.2. INSTRUCCIONES PARA MOVIMIENTOS DE DATOS

El microprocesador 8035 posee un conjunto de instrucciones que permiten mover los contenidos de los registros hacia el acumulador o viceversa, mover datos a o desde memoria

(de datos o de programa). El conjunto completo de estas instrucciones se muestra en la tabla 3-2.

TABLA 3-2

INSTRUCCIONES DEL 8035 PARA MOVIMIENTOS DE DATOS

MNEMONICO	CODIGO DE OPERACION	SEMANTICA	DESCRIPCION
MOV A,Rr	F8-FF F8==>R0, F9==>R1... FF==>R7	A:=Rr (r=0 a 7)	Mueve el contenido del registro especificado al acumulador.
MOV A,@Rr R0 R1	F0-F1 F0==>R0 F1==>R1	A:=(Rr) (r=0 a 1)	Mueve al acumulador el contenido de la memoria de datos direccionada por Rr(R0 o R1).
MOV A,#dato	23 dato	A:=dato	Esta es una instrucción de dos bytes. Mueve al acumulador el dato especificado en el segundo byte.
MOV Rr,A	A8-AF A8==>R0, A9==>R1... AF==>R7	Rr:=A (r=0 a 7)	Mueve al registro especificado el contenido acumulador.
MOV @Rr,A R0 R1	A0-A1 A0==>R0 A1==>R1	(Rr):=A (r=0 a 1)	Mueve el contenido del acumulador a la localidad de memoria apuntada por Rr(R0 o R1).
MOV Rr,#dato	B8-BF	Rr:=dato (r=0 a 7)	Esta es una instrucción de dos bytes. El registro especificado se carga con el dato del segundo byte.
MOV @Rr,#dato R0 R1	B0-B1 B0==>R0 B1==>R1	(Rr):=dato R=0 a 1	Esta es una instrucción de dos bytes. El dato en el segundo byte es carga en la localidad de memoria de datos a puntada por Rr(R0 o R1)
XCH A,Rr	28-2F 28==>R0, 29==>R1... 2F==>R7	A Rr r=0 a 7	El contenido del acumulador y el del registro especificado se intercambian.

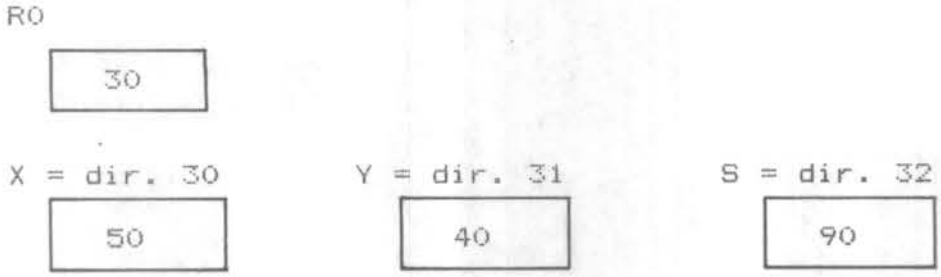
XCH A,@Rr R0 R1	20-21 20==>R0 21==>R1	A (Rr) r=0 a 1	El contenido del acumulador y el de la localidad de memoria de datos apuntada por Rr se intercambian.
XCHD A,@Rr R0 R1	30-31 30==>R0 31==>R1	A0 (Rr)0 A1 (Rr)1 A2 (Rr)2 A3 (Rr)3 (r=0 a 1)	Hace un intercambio de los bits 0 a 3 del acumulador con sus correspondientes bits de la localidad de memoria apuntada por Rr. El resto de los bits permanece inalterado.
MOV A,PSW	C7	A:=PSW	Mueve al acumulador el contenido de la palabra de estatus de programa.
MOV PSW,A	D7	PSW:=A	Mueve el contenido del acumulador a la palabra de estatus de programa.
MOVX A,@Rr R0 R1	80-81 80==>R0 81==>R1	A:=(Rr) (r=0 a 1)	Esta es una instrucción de dos ciclos. Hace que el contenido de la localidad de memoria exter apuntada por Rr se mueva al acumulador.
MOVX A,@Rr R0 R1	90-91 90==>R0 91==>R1	(Rr):=A (r=0 a 1)	Esta es una instrucción de dos ciclos. Mueve el contenido del acumulador a la localidad de memoria externa apuntada por Rr(R0 o R1).
MOVP A,@A	A3	PC0-PC7:=A A:=(PC)	Esta es una instrucción de dos ciclos. Hace que el contenido de la localidad de memoria de programa apuntada por el acumulador sea movida hasta el acumulador. El PC se restablece al terminar su ejecución.
MOVP3 A,@A	E3	PC0-7:=A PC8-11:=0011 A:=(PC)	Esta es una instrucción de dos ciclos. Hace que el contenido de la localidad de memoria de programa apuntada por el acumulador (dentro de la pagina 3), sea cargada en el acumulador.

A manera de ilustrar el uso de las instrucciones de movimiento de datos se daran los siguientes ejemplos:

Ejemplo 1

Suponga que las direcciones X, Y, S corresponden a las direcciones 30, 31 y 32 respectivamente, de la memoria de datos interna del 8035; un programa para hacer la suma de X con Y y dejar el resultado en S se puede codificar de la forma siguiente:

```
MOV R0,#30   Coloca en R0 dirección de primer dato.
MOV A,@R0    Carga en acumulador dato apuntado por
             R0.
INC R0       R0 apunta proximo dato (dirección 31).
ADD A,@R0    Suma al acumulador el dato apuntado
             por R0.
INC R0       R0 apunta a la dirección 32 (donde se
             guardará resultado de la suma).
MOV @R0,A    Se almacena el resultado de la suma
             en la dirección 32.
```



Ejemplo 2

Supongase ahora que las direcciones X, Y, S corresponden a las direcciones C00, C01 y C02 de la memoria externa de programa; un programa para sumar X con Y dejando el resultado en S, se puede codificar de la siguiente manera:

```
MOV A,#0C    Carga A con la página de memoria ex
             terna donde estan los datos.
OUTL P2,A    Saca la página por el puerto 2 del
```

C A P I T U L O I V

4.1. SISTEMA IMSAI

El sistema IMSAI es un microcomputador con todas sus funciones implementadas en unos pocos circuitos integrados que ocupan una sola tarjeta de circuito impreso. El sistema IMSAI, está basado en el microprocesador 8035 y está constituido por:

Microprocesador 8035

Memoria RAM externa de programa

Memoria ROM externa de programa, en la que se encuentra grabado el monitor

Expansor de puertos 8243, el cual provee cuatro puertos adicionales de 4 bits c/u

Un display alfanumérico de 9 digitos

Un teclado hexadecimal

Un manejador de teclado y display 8279

4.1.1. DESCRIPCION FUNCIONAL

La tarjeta del IMSAI como se dijo anteriormente trae el 8035 como microprocesador con un cristal de cuarzo de 3 MHz, el cual, permite que el microprocesador tenga un ciclo de máquina de 4.1905 microsegundos y que la velocidad del timer interno sea de 134.095 microsegundos por cuenta. Las facilidades y características mas resaltantes del sistema son:

a) Interfase para cassette con todas las rutinas para el manejo de dicha interfase incluidas

```
                  8035 (bits 0 a 3).
MOV R0,#00        Coloca en R0 el desplazamiento den
                  tro de la página (localidad dentro
                  de la página donde se encuentra el
                  primer dato).
MOVX A,@R0        Carga en A dato apuntado por R0(00)
                  en la pagina que esta en P20-P23(0C)
MOV R3,A          mueve a R3 el dato que se encontraba
                  en A (primer dato).
INC R0            Apunta a proximo dato(C01).
MOVX A,@R0        Carga en A el segundo dato desde me
                  moria externa.
ADD A,R3          Suma el segundo dato con el primero
INC R0            Apunta a la proxima(C02) direccion.
MOVX @R0,A        Almacena resultado de la suma en la
                  localidad de memoria apuntada por R0
```

Notese que la página se sacó una sola vez en el programa que se acaba de ver; la razon de esto es que mientras no se realice una instrucción de E/S que use el puerto 2, el dato que se haya sacado por dicho puerto (en este caso la página), permanece inalterado.

3.3.3. INSTRUCCIONES DE MANIPULACION DE PROGRAMA

Este grupo de instrucciones permiten controlar la secuencia o flujo de ejecución los programas. En este grupo de instrucciones se encuentran las de saltos condicionales e incondicionales, llamadas a subrutinas etc.. Todas estas instrucciones se caracterizan por que tienen la capacidad de alterar el contenido del contador de programa, lo que les permite alterar la secuencia de ejecución de un programa.

Las instrucciones de salto incondicionales permiten saltar a una localidad de memoria dentro de la página corriente o en otra página cualquiera. El 8035 posee dos instrucciones para estos propositos, ellas son:

JMPP @A

JMP dirección

La instrucción JMPP @A, permite saltar a cualquier localidad de memoria de programa, dentro de la página corriente. Esta es una instrucción de dos ciclos y un byte; cuando se ejecuta hace que el contenido del acumulador se cargue en los bytes menos significativos del PC (PC0-PC7); los bits mas significativos del PC (PC8-PC11), permanecen inalterados, razon por la cual los saltos son dentro de la página corriente.

La instrucción JMP dirección, es una instrucción de dos ciclos y dos bytes; esta instrucción permite ejecutar saltos incondicionales a cualquier localidad de memoria de programa, dentro de un bloque de 2kbytes de memoria. Como se mencionó en la descripción del 8035, el espacio de memoria del mismo, esta dividido en dos bancos de memoria, los cuales pueden ser seleccionados mediante dos instrucciones especiales (SEL MB 0 o SEL MB 1); estas instrucciones hacen que en el momento de hacer un salto (JMP o JSB), el bit 11 del contador de programa se haga 0 o 1. El primer byte de la instrucción JMP, contiene el código de operación, en los 5 bits menos significativos y en los tres bits mas significativos(correspondientes a los bits 10, 9, 8 del PC) la página de memoria a la que se hará el salto, dentro del bloque de 2Kbytes; el segundo byte contiene la dirección, dentro de la página especificada, a la que se hará el salto. La secuencia de eventos cuando se ejecuta la instrucción JMP dirección, es como sigue: en el primer ciclo el contenido de los bits mas significativos del PC (PC8-PC10) se sustituye por los de los bits mas significativos del primer byte de la instrucción; en el segundo ciclo, los bits menos significativos del PC, se sustituyen por el segundo byte de la instrucción; el bit 11 del PC (PC11), permanece inalterado y su valor lo define la última instrucción SEL MB (0 o 1) que se haya ejecutado, de manera que, solo es posible hacer los saltos dentro del bloque de 2Kbytes que se hayan seleccionado con la instrucción SEL MB.

Las instrucciones de saltos condicionales, permiten saltar a cualquier localidad de memoria, dentro de la página corriente, en base al cumplimiento o no de una determinada condición. Todas las instrucciones de saltos condicionales tienen como caracteristica básica que son de dos bytes; en el primer byte, está el código de la instrucción y en el segundo byte, está la dirección a donde se saltará si se cumple la condición especificada en el código de operación. Todos los saltos que se pueden hacer con estas instrucciones de saltos condicionales, son dentro de la página corriente, en virtud de que dichas instrucciones, no alteran los bits mas significativos del PC (PC8-PC11). Si la condición especificada en el código de operación de una instrucción de salto condicional no se cumple, el contador de programa se incrementará en dos (normalmente el PC

se incrementa en uno), de manera de proceder a ejecutar la instrucción que sigue en secuencia, a la de salto condicional. La tabla siguiente resume todas las instrucciones de manipulación de programa que trae el 8035.

TABLA 3-3

INSTRUCCIONES DEL 8035 PARA MANIPULACION DE PROGRAMA

MNEMONICO	CODIGO DE OPERACION	SEMANTICA	DESCRIPCION
JMPP @A	B3	PC0-7:=(A)	El contador de programa se carga con el contenido de la localidad de memoria de programa apuntada por el acumulador.
JMP dirección	04,24,44, 64,84,A4, C4,E4. 04==>pag.0 24==>pag.1 44==>pag.2 64==>pag.3 84==>pag.4 A4==>pag.5 C4==>pag.6 E4==>pag.7	PC8-10:=A8-10 PC0-7:=A0-7 PC11 igual.	Los bits mas significativos del primer byte (A8-10), se cargan en los bits mas significativos del PC; el segundo byte de la instrucción (A0-7), se carga en los bits menos significativos del PC; el bit 11 del PC se deja igual y su valor depende de la última instruccion SEL MB que se haya ejecutado.
CALL direc	14,34,54, 74,94,B4, D4,F4 14==>pag.0 34==>pag.1 54==>pag.2 74==>pag.3 94==>pag.4 B4==>pag.5 D4==>pag.6 F4==>pag.7	(SP):=PC y PSW SP := SP+1 PC0-7:=direc Pc8-10:=A8-A10 PC8-10:=A8-10 PC11 igual	Instrucción para llamar a subrutina. El contenido del PC en conjunto con los bits 4 a 7 de la PSW se salvan en el stack. los bits mas significativos del primer byte(A8-10), se cargan en los bits mas significativos del PC. El segundo byte de la instrucción (A0-7), se carga en los bits menos significativos del PC. El bit 11 del PC depende de de la última instrucción SEL MB.
RET	83	SP:=SP-1	Retorna de subrutina,

		PC:=(SP)	sin restaurar estatus.
RETR	93	SP:=SP-1 PC:=(SP) PSW:=(SP)	Retorna de subrutina, restaurando estatus del programa(C,AC,F0,BS).
JC direc	F6	si C=1 ==> PC0-7:=direc si C=0 ==> PC:=PC+2	Salta a la dirección es pecificada en el segun do byte si el flag de carry está en uno.
JNC direc	E6	si C=0 ==> PC0-7:=direc si C=1 ==> PC:=PC+2	Salta a la dirección es pecificada en el segun do byte, si el flag de es cero.
JZ direc	C6	si A=0 ==> PC0-7:=direc si A=0 ==> PC:=PC+2	Salta a la dirección es pecificada en el segun do byte, si el conteni do del acumulador es ce ro.
JNZ direc	96	si A=0 ==> PC0-7:=direc si A=0 ==> PC:=PC+2	Salta a la dirección es pecificada en el segun do byte, si el conteni do del acumulador es di ferente de cero.
JT0 direc	36	si T0=1 ==> PC0-7:=direc si T0=0 ==> PC:=PC+2	Salta a la dirección es pecificada en el segun do byte, si la línea T0 está en uno.
JNT0 direc	26	si T0=0 ==> PC0-7:=direc Si T0=1 ==> C:=PC+2	Salta a la dirección es pecificada en el segun do byte, si la línea T0 está en cero.
JT1	56	si T1=1 ==> PC0-7:=direc si T1=0 ==> PC:=PC+2	Salta a la dirección es pecificada en el segun do byte, si la línea T1 está en uno.
JNT1	46	si T1=0 ==> PC0-7:=direc si T1=1 ==> PC:=PC+2	Salta a la dirección es pecificada en el segun do byte, si la línea T1 vale cero.
JF0 direc	B6	si F0=1 ==> PC0-7:=direc si F0=0 ==> PC:=PC+2	Salta a la dirección es pecificada en el segun do byte, si el flag 0 está en uno.
JF1 direc	76	si F1=1 ==>	Salta a la dirección es

		PC0-7:=direc si F1=0 ==> PC:=PC+2	pecificada en el segun do byte, si el flag 1 está en uno.
JTF direc	16	si TF=1 ==> PC0-7:=direc si TF=0 ==> PC:=PC+2	Salta a la dirección es pecificada en el segun do byte, si el flag del timer está en uno.
JNI direc	86	si INT=0 ==> PC0-7:=direc si INT=1 ==> PC:=PC+2	Salta a la dirección es pecificada en el segun do byte, si la línea de interrupción está acti va (INT=0).
JBb direc	12-F2 12==>bit0 32==>bit1 52==>bit2 72==>bit3 92==>bit4 b2==>bit5 D2==>bit6 F2==>bit7	si Bb=1 ==> PC0-7:=direc (b=0 a 7) si Bb=0 ==> PC:=PC+2	Salta a la dirección es pecificada en el segun do byte, si el bit b del acumulador es uno.
DJNZ Rr, direc	E8-EF E8==>R0 E9==>R1 EA==>R2 EB==>R3 EC==>R4...	Rr:=Rr-1, si Rr=0 ==> PC0-7:=direc si Rr=0 ==> PC:=PC+2 (r=0 a 7)	Decrementa el registro Rr(R0-R7), y salta a la dirección especificada en el segundo byte, si el contenido del regis tro Rr se hace cero.

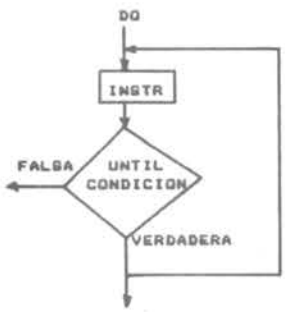
A manera de ejemplo del uso de estas instrucciones de manipulación de programa se implementaran algunas de las estructuras de ruptura de secuencia y de iteración que se vieron en la sección 3.2.3 y que aparecen esquematizadas en las figuras 3-15 y 3-16. Es interesante recordar que el uso de estas estructuras nos permite simplificar la codificación de los programas a partir de los diagramas de flujo.

Las implementaciones se haran con el doble propósito de ilustrar por una parte el uso de las instrucciones vistas y por la otra el de tener codificadas en assembler, las estructuras presentes en los lenguajes de alto nivel, de forma de poderlas usar en cualquier programa que se esté desarrollando.

Estructura IF THEN ELSE (figura 3-15)

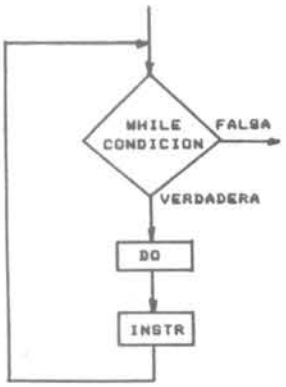
ETIQUETA	MNEMONICO	COMENTARIO
IF	JTO THEN	SALTE A THEN SI TO=1.
	JMP ELSE	SI NO VAYA A ELSE.
THEN	INSTR. 1	
	INSTR. 2	
	.	BLOQUE DE INSTRUCCIONES
	.	DENTRO DEL LAZO DE THEN.
	.	
	Jmp FIN	
ELSE	INSTR. 1	
	INSTR. 2	
	.	BLOQUE DE INSTRUCCIONES
	.	DENTRO DEL LAZO ELSE.
	.	
FIN	NOP	LUGAR DONDE COINCIDEN AM
		BAS BIFURCACIONES.

Estructura DO-UNTIL (figura 3-16)



ETIQUETA	MNEMONICO	COMENTARIO
DO	INSTR. 1	
	INSTR. 2	
	.	BLOQUE DE INSTRUCCIONES
	.	DENTRO DEL LAZO DO.
	.	
	JNZ DO	LA CONDICION USADA PARA
		EL CHEQUEO ES A $\neq$ 0; EN
		TANTO A $\neq$ 0 SE REPETIRA
		EL CONJUNTO DE INSTRUCCIO
		NES DENTRO DEL DO.
FIN	NOP	SALE DEL LAZO AL CUMPLIR
		SE LA CONDICION QUE SE
		CHEQUEA (A = 0).

Estructura WHILE-DO (figura 3-16)

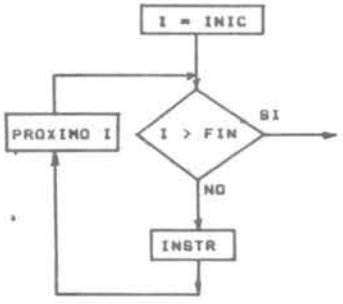



```
ETIQUETA   MNEMONICO      COMENTARIO

 WHILE     JZ DO          LA CONDICION USADA PARA
                          EL CHEQUEO ES A = 0; EN
                          TANTO SE CUMPLA SE EJECU
                          TA EL BLOQUE DE INSTRUC
                          CIONES ENGLOBADOS EN DO.
           JMP FALSA      SALE DEL LAZO AL NO CUM
                          PLIRSE LA CONDICION(A≠0)
 DO        INSTR. 1   ┐
           INSTR. 2   │
              .       │   BLOQUE DE INSTRUCCIONES
              .       │   DENTRO DE DO.
              .       │
           JMP WHILE  ┘
 FALSA     NOP            SALE DEL LAZO
```

Estructura FOR (figura 3-16)

La estructura FOR de ALGOL o el DO del FORTRAN, se pueden implementar en forma muy fácil con elassembler del 8035, mediante el uso de la instrucción "decremente registro y salte si se hace cero (DJNZ Rr)"; a continuación una posible forma de codificación del lazo iterativo DO




```
ETIQUETA   MNEMONICO      COMENTARIO
           MOV RR,#I      SE CARGA CUALQUIER REGIS
                          TRO CON EL NUMERO DE ITE
                          RACIONES REQUERIDOS
 LAZO      INSTR. 1   ┐
           INSTR. 2   │
              .       │   BLOQUE DE INSTRUCCIONES
              .       │   DENTRO DEL LAZO
              .       │
           DJNZ RR,LAZO   SI NO SE HAN HECHO TODAS
                          LAS ITERACIONES REQUERI
                          DAS (RR ≠ 0), REPITE EL
                          BLOQUE DE INSTRUCCIONES
                          DENTRO DEL LAZO.
 FIN       NOP            SE REALIZARON YA TODAS
                          LAS ITERACIONES.
```

La codificación de las estructuras de iteración y de ruptura de secuencia que se realizó en las líneas anteriores, puede ser utilizada para la codificación de los diagramas de flujo correspondientes a los distintos bloques del problema de sacar el promedio a N números (figuras 3-12, 3-13 y 3-14); a

continuación se ilustrará como pasar del diagrama de flujo a la codificación, aprovechando adicionalmente la codificación de los lazos iterativos y de ruptura de secuencia, que ya se codificaron.

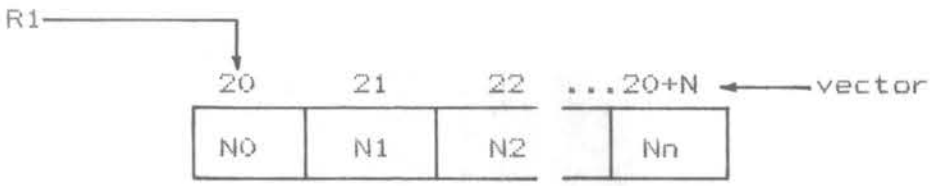
Se comenzará por el diagrama de flujo de la figura 3-12, el cual corresponde a la lectura de N números desde el teclado; si se observa este diagrama se podrá notar que está constituido por una secuencia de instrucciones y por un lazo iterativo que encaja perfectamente con el de DO-UNTIL; a continuación la codificación:

```
        CALL KEYINP     Lea N desde teclado.
        JZ PARE         Si N=0 pare.
        MOV R0,A        Inicializa contador de nú
                        meros a leer.
        MOV R2,A        Guarda en R2 a N.
        MOV R1,#20      Pone en R1 direccion a par
                        tir de la que se almacena
                        ran los números leidos.
DO      CALL KEYINP     Lee un número.
        MOV @R1,A       Almacena número en direc
                        ción apuntada por R1.
        INC R1          Apunta a proxima dirección
        DJNZ R0,DO      Decrementa contador de nú
                        meros a leer, repite el la
                        zo si todavía faltan por
                        leer números.
SIGA    NOP             Leyo los N números y sale.
```

Si se observa el diagrama de la figura 3-13, que corresponde al diagrama de flujo del algoritmo para sumar los N números, se notará que se encuentra claramente definida una secuencia de instrucciones y un lazo compuesto por una secuencia y la estructura IF-THEN-ELSE; a continuación la codificación:

```
        MOV A,R2        Carga en A el valor de N.
        MOV R0,A        Inicializa el contador de
                        números.
        MOV A,#00       Inicializa suma en cero.
        MOV R1,#20      Dirección de primer número
LOOP    ADD A,@R1       Suma número apuntado por
                        R1, a la suma parcial.
  IF    DJNZ R0,THEN    Decrementa y chequea a ver
                        si sumó todos los números.
ELSE    JMP DIVIDA      Si sumó todos los números,
                        va a dividir por N para sa
                        car promedio
THEN    INC R1          Apunta al proximo número.
        JMP LOOP        Repite el lazo.
DIVIDA ...
```

Cuando sale del lazo, el acumulador contiene el resultado de la suma de los N números. Es interesante hacer notar que este programa es un buen ejemplo de como se lee, almacena y se opera sobre un vector de datos. El uso del direccionamiento indirecto, usando al registro R1 como apuntador, ha facilitado enormemente la tarea, lo cual se puede visualizar en el siguiente esquema.



Solo falta por codificar el algoritmo representado en el diagrama de flujo de la figura 3-14, que corresponde a la división por N de la suma de los N números, para obtener el promedio; si se observa el diagrama de flujo se puede notar que encaja perfectamente con una estructura de WHILE-DO.

Antes de entrar en la codificación del algoritmo para dividir por N, se hace necesario resolver un detalle que no se ha tomado en cuenta, como es el hecho de que el 8035 no tiene instrucciones que permitan realizar en forma directa la operación aritmética resta. Para la realización de la resta es necesario utilizar el método de complemantar a dos el sustraendo y sumarselo al minuendo. El procedimiento para sacarle el complemento a dos a un número binario, consiste en sacarle el complemento a uno a ese número (cambiando los ceros por unos y los unos por ceros) y a este resultado se le suma 1. Por ejemplo para sacar el complemento a dos del número binario 0100 se procede como sigue:

```
0100    número al que se sacará complemento a dos

1011    complemento a uno de ese número
0001+   se le suma uno al complemento a uno
-----
1100    complemento a dos del número
```

Adicional al inconveniente de la carencia de una instrucción para restar, se presenta el problema de la comparación por mayor menor o igual entre dos números cualesquiera; el 8035 no posee ninguna instrucción que permita realizar directamenta este tipo de comparaciones, por lo cual es necesario desarrollar un método par realizar estas comparaciones, requeridas en el algoritmo de división.

A continuación se desarrollaran algunos ejemplos que

permitiran ilustrar el método a seguir para realizar las comparaciones por mayor, menor o igual; adicionalmente se ilustrará la resta por el método del complemento a dos.

Resta con complemento a dos

a) restar A-B (siendo A>B)

```
                1 <---------flag de acarreo

A=0010           0010+
B=0001  ===>     1111 <-----CII(B)
                 ------
                 0001 <-----resultado
```

b) Restar A-B (siendo A=B)

```
                1 <---------flag de acarreo

A=0010           0010+
B=0010  ===>     1110 <-----CII(B)
                 ------
                    0000 <----resusltado
```

c) Restar A-B (siendo A<B)

```
               0 <---------no hubo acarreo

A=0001           0001+
B=0010  ===>     1110
                 -------
                 1111 <-----este  no es el
                                  resultado.  el
                 0000            resultado  ver
                 0001            dadero se  con
                --------         sigue  sacando
resultado-----> 0001             complemento  a
                                 dos.
```

Tomando en cuenta los resultados de las operaciones de resta de dos números mediante el método del complemento a dos, es posible saber si un número es mayor, menor o igual a otro número, mediante la siguiente regla:

si C=1 cuando se hace A-B ===> A > B

si C=0 cuando se hace A-B ===> A < B

Cuando los números son iguales, adicional a la activación del bit de acarreo el resultado de la suma con el complemento a dos es cero; las instrucciones JC, JNC y JZ, pueden ser utilizadas para realizar las comparaciones por mayor, menor e igual, en conjunto con la resta mediante el metodo del complemento a dos.

A continuación la codificación del algoritmo para división por N con la finalidad de obtener el promedio de los N números.

```
        MOV R0,#00      Inicializa cociente.
        MOV R3,A        Guarda la suma en R3.
        MOV A,R2        A := N
        CPL A           A := NOT A  (NOT N)
        INC A           A := A + 1  (NOT N + 1); sa
                        ca complemento a dos de N.
        MOV R4,A        Guarda en R4 complemento a
                        dos de N.
        MOV A,R3        A := SUMA
        ADD A,R4        A := SUMA - N
WHILE   JNC ESC         Si no hubo acarreo, el divi
                        sor es mayor que el dividen
                        do (SUMA>N) y sale del lazo
        INC R0          Si hubo acarreo, incrementa
                        cociente.
        ADD A,R4        A := SUMA - N
        JMP WHILE
ESC     MOV A,R0        Coloca el cociente en A.
        CALL DISPRG     Lo saca por display.
```

3.3.4. INSTRUCCIONES DE MANIPULACION DE ESTATUS DE PROGRAMA

Este grupo de instrucciones tienen por finalidad chequear o cambiar las condiciones de estatus del programa, las cuales se encuentran almacenadas en la palabra de estatus de programa PSW, la cual se describió detalladamente en el capítulo II. En este grupo de instrucciones se incluyen tambien las que permiten deshabilitar y habilitar el sistema de interrupción del 8035. La tabla 3-4 resume todas las instrucciones de este grupo que posee el 8035.

TABLA 3-4

INSTRUCCIONES PARA MANIPULACION DE ESTATUS DE PROGRAMA

MNEMONICO	CODIGO DE OPERACION	SEMANTICA	DESCRIPCION
CLR C	97	C:=0	Coloca en cero el flag de acarreo C.
CPL C	A7	C:=NOT C	Complementa el flag de acarreo.
CLR F0	85	F0:=0	Coloca en cero el flag F0.
CPL F0	95	F0:=NOT F0	Complementa el flag F0.
CLR F1	A5	F1:=0	Coloca en cero el flag F1.
CPL F1	B5	F1:=NOT F1	Complementa el flag F1.
EN I	05		Habilita el sistema de interrupción externa.
DIS I	15		Deshabilita el sistema de interrupción externa
SEL RB0	C5	PSW4:=0	selecciona banco de registros 0.
SEL RB1	D5	PSW4:=1	Selecciona banco de registros 1.
SEL MB0	E5	PC11:=0	Selecciona banco de memoria cero.
SEL MB1	F5	PC11:=1	Selecciona banco de memoria 1.
ENTO CLK	75	T0:=reloj	Saca por la línea T0 el reloj interno del micro procesador.

3.3.5. INSTRUCCIONES DE ENTRADA/SALIDA (E/S)

Las instrucciones de E/S, comprenden todas las instrucciones que permiten leer o escribir en cualquiera de los puertos del microprocesador propiamente dicho o en sus puertos de extensión. Estas instrucciones permiten sacar el contenido del acumulador por los puertos e introducir datos al acumulador desde

un puerto cualquiera del microprocesador. Tambien permiten la realización de las operaciones lógicas AND y OR, de los puertos con el acumulador y almacenar el resultado en el acumulador. Como el 8035 posee instrucciones específicas para el manejo de la entrada/salida, es posible conectarle periféricos mediante la tecnica descrita en el capítulo II, denominada I/O MAPPED I/O. La tabla 3-5 resume las instrucciones de E/S disponibles en el 8035.

TABLA 3-5

INSTRUCCIONES DEL 8035 PARA ENTRADA/SALIDA

MNEMONICO	CODIGO DE OPERACION	SEMANTICA	DESCRIPCION
IN A,P1	09	A:=P1	Mete el dato presente en el puerto 1, en el a cumulador.
OUTL P1,A	39	P1:=A	Vacía el contenido del acumulador en el puerto 1 (P1).
ANL P1,#dato	99	P1:= P1 AND dato	Hace un AND entre el da to presente en el puer to 1 y el dato en el se gundo byte de la ins trucción.
ORL P1,#dato	89	P1:= P1 OR dato	Hace un OR entre el da to presente en el puer to 1 y el dato en el se gundo byte de la ins trucción.
IN A,P2	0A	A:=P2	Mete el dato presente en el puerto 2, en el a cumulador.
OUTL P2,A	3A	P2:=A	Vacía el contenido del acumulador en el puerto 2(P2).
ANL P2,#dato	9A	P2:= P2 AND dato	Hace un AND entre el da to presente en el puer to 2 y el dato en el se gundo byte de la ins trucción; el resultado lo vacía en el puerto.
ORL P2,#dato	8A	P2:=	Hace un OR entre el da

		P2 or dato	to presente en el puerto y el dato en el segundo byte; el resultado se vacía en P2.
INS A,BUS	08	A:=BUS	Pone en el acumulador el dato presente en el bus.
OUTL BUS,A	02	BUS:=A	Vacía en el bus el contenido del acumulador.
ANL BUS,#dato	98	BUS:= BUS AND dato	Coloca en el bus el resultado del AND entre el dato presente en el bus y el dato en el segundo byte de la instrucción.
ORL BUS,#dato	88	BUS:= BUS OR dato	Pone en el bus el resultado del OR entre el dato presente en el bus y el dato en el segundo byte de la instrucción.
MOVD Pp,A	3C-3F 3c==>P4 3D==>P5 3E==>P6 3F==>P7	Pp:=A0-3 (p=4 a 7)	Los 4 bits menos significativos del acumulador son movidos al puerto especificado del expansor de puertos 8243.
MOVD A,Pp	0C-0F 0C==>P4 0D==>P5 0E==>P6 0F==>P7	A0-3:=Pp A4-7:=0 (p=4 a 7)	El dato presente en el puerto especificado (p) del expansor de puertos 8243, se carga en los bits menos significativos del acumulador; el resto de los bits del acumulador se ponen en cero.
ANLD Pp,A	9C-9F 9C==>P4 9D==>P5 9E==>P6 9F==>P7	Pp:= Pp AND A0-3	Coloca en el puerto especificado del expansor el resultado del AND de los 4 bits menos significativos del acumulador y el dato presente en dicho puerto.
ORLD Pp,a	8C-8F 8C==>P4 8D==>P5 8E==>P6 8F==>P7	Pp:= Pp OR A0-3	Coloca en el puerto especificado del 8243, el resultado del OR entre el dato presente en dicho puerto y los 4 bits menos significativos

del acumulador.

3.3.6. INSTRUCCIONES PARA MANEJO DEL TIMER

EL 8035 posee un timer interno, el cual puede ser controlado por software; operaciones como arrancar el timer, pararlo, inicializarlo a un valor determinado a partir del cual comience a operar asi como tambien chequear su estatus pueden ser hechas con el set de instrucciones que para esos propósitos trae el 8035. A continuación la tabla que resume todas las instrucciones con que cuenta el 8035 para manejar el timer.

TABLA 3-6

INSTRUCCIONES DEL 8035 PARA MANEJO DEL TIMER

MNEMONICO	CODIGO DE OPERACION	SEMANTICA	DESCRIPCION
MOV A,T	42	A:=T	Mueve al acumulador el contenido del timer.
MOV T,A	62	T:=A	Mueve al timer el contenido del acumulador.
STRT T	55		Arranca el timer a contar. El timer se incrementa cada 32 ciclos de instrucción.
STRT CNT	45		Selecciona la entrada T1 del 8035, como la entrada al timer/counter e inicia el conteo. El contador es incrementado a cada transición alta a bajo de la señal en la línea T1.
STOP TCNT	65		Para el timer/counter.
EN TCNTI	25		habilita la interrupción producida por el timer.
DIS TCNTI	35		Deshabilita la interrupción producida por el timer.

C A P I T U L O I V

4.1. SISTEMA IMSAI

El sistema IMSAI es un microcomputador con todas sus funciones implementadas en unos pocos circuitos integrados que ocupan una sola tarjeta de circuito impreso. El sistema IMSAI, está basado en el microprocesador 8035 y está constituido por:

Microprocesador 8035

Memoria RAM externa de programa

Memoria ROM externa de programa, en la que se encuentra grabado el monitor

Expansor de puertos 8243, el cual provee cuatro puertos adicionales de 4 bits c/u

Un display alfanumérico de 9 digitos

Un teclado hexadecimal

Un manejador de teclado y display 8279

4.1.1. DESCRIPCION FUNCIONAL

La tarjeta del IMSAI como se dijo anteriormente trae el 8035 como microprocesador con un cristal de cuarzo de 3 MHz, el cual, permite que el microprocesador tenga un ciclo de máquina de 4.1905 microsegundos y que la velocidad del timer interno sea de 134.095 microsegundos por cuenta. Las facilidades y características mas resaltantes del sistema son:

a) Interfase para cassette con todas las rutinas para el manejo de dicha interfase incluidas

en el programa monitor.

b) Entrada y salida serial tanto para el tipo RS-232, como para lazo de corriente con todas las rutinas para su manejo incluidas en el monitor

c) 5 reles capaces de manejar hasta 2amp a 220V, o 5 amp a $24V_{DC}$

d) 1 Kbyte de memoria RAM de programa

e) Una sola fuente de alimentación de +5V

El monitor del sistema IMSAI, llamado IMP-48 es el programa supervisor que permite la comunicación entre el usuario y la tarjeta IMSAI. El programa monitor ocupa 1 kbyte de memoria ROM y le da las siguientes facilidades al usuario:

- Permite cargar programas en la memoria de programa

- Permite cargar datos tanto en memoria externa de programa, como en memoria interna de datos

- Permite examinar el contenido de cualquier posición de memoria, bien sea interna o externa

- Permite ejecutar los programas del usuario en dos modalidades: en una modalidad, llamada "standalone", los programas se ejecutan en forma continua a partir de una direccion especificada; en la segunda modalidad, llamada "break point", el programa se ejecutará desde una cierta dirección inicial hasta una cierta dirección final, las cuales, son suministradas por el usuario

- Permite almacenar y leer información o programas desde cassette

- Permite cargar datos por el puerto serial o vaciar a este puerto serial programas o datos desde memoria

- Permite ajustar el volumen y el tono del reproductor de cassette, para lo cual trae una rutina especial, que permite ir verificando la señal de la interfase hasta conseguir una operación libre de error

- Permite verificar la operación del sistema completo, con la corrida de un programa de autodiagnóstico que trae incorporado

El programa monitor debe manipular los recursos del microprocesador 8035 tales como: stack, registros, memoria y el timer, de forma tal de compartirlos al máximo con los programas del usuario, imponiendo al usuario el menor número de restricciones posibles; sin embargo, en algunos casos, para implementar determinadas funciones en el monitor se ha hecho necesario imponer ciertas restricciones, como en el caso de la función "break point"; dicha función es de gran ayuda para el proceso de depuración y puesta a punto de los programas, ya que, mediante la misma es posible ejecutar trozos de programa y obtener los contenidos de los registros y de la memoria de datos interna y externa, despues de la ejecución de esos trozos de programa, permitiendo un seguimiento preciso del programa que se está desarrollando, lo que facilita la labor de detección de lazos y errores de lógica en el programa. Cuando se ejecuta un programa en la modalidad "breakpoint", existe una interacción continua entre el programa del usuario y el programa monitor, lo que hace necesario imponer al programa de usuario una serie de restricciones a fin de garantizar que el mismo, no se verá afectado por las acciones y operaciones realizadas por el monitor, cuando dicho programa se ejecute en esta modalidad de operación (breakpoint), lo cual originaría resultados inconsistentes y erroneos.

RESTRICCIONES IMPUESTAS A LOS PROGRAMAS DEL USUARIO

a) El registro 7 del banco de registros 1 no debe ser usado por el programa del usuario en virtud, de que dicho registro lo usa el monitor para salvar el contenido del acumulador, al momento de ocurrir el "breakpoint", o sea, cuando se alcanza la dirección que el usuario suministró como fin de ejecución.

b) Las localidades de memoria externa comprendidas entre FE1 y FFD (ambas inclusive) no deben ser usadas por el programa del usuario, en virtud de que el monitor al momento de alcanzar el punto de "breakpoint", en el programa del usuario, guarda en esas localidades el contenido de la memoria interna del 8035 comprendidas entre la 0 y la 19 ambas inclusive(esto corresponde a el banco de registros 0, el stack y los dos primeros registros del banco de registros 1, R0 y R1).

c) El stack del 8035, como se vió en el capítulo II, tiene capacidad hasta para 8 llamadas a subrutinas dentro de subrutina; debido a que el uso del "breakpoint" requiere un nivel de los 8 disponibles, el usuario quedará limitado a un máximo de 7 niveles; si excede este límite perderá el nivel mas bajo del stack lo que generará resultados impredecibles.

d) Debido a que en el IMSAI todos los programas del usuario deben residir a partir de la localidad 800_H en adelante, o sea en el segundo banco de memoria de 2 Kbytes ya que el primero está constituido por la memoria PROM (2 Kbytes), que contiene el monitor, todas las instrucciones de salto (JUMPS) y llamadas a subrutinas dentro del programa del usuario deben tener seleccionado el banco de memoria 1, por lo cual es recomendable que la primera instrucción de los programas del usuario sea SEL MB1 ($F5_H$), recomendación esta que es válida tanto para la ejecución de programas en la modalidad "breakpoint" como en la modalidad "standalone"; cuando el usuario desee llamar alguna de las subrutinas que trae incorporado el monitor, deberá previamente seleccionar el banco de memoria 0 con la instrucción SEL MB0.

e) El monitor está hecho de manera tal que el requiere de la existencia de las localidades de memoria comprendidas entre FE1 y FFD, por lo que, cuando solo se disponga de 1 Kbyte de memoria externa, este estará mapeado de manera de que constituya el segundo kbyte de memoria o sea el comprendido entre las localidades C00 y FFF; dicho en otras palabras, el monitor requiere siempre de la existencia de las localidades de memoria comprendida entre los límites antes mencionados. Esta consideración es válida en cualquiera de las dos modalidades de operación que se han descrito

.f) Nunca se debe colocar el punto de "breakpoint" en localidades que contengan las instrucciones RET, RETR, JMPP @A o cualquier otra instrucción sencilla que esté precedida por una instrucción que sea el destino de una instrucción de CALL o de BRANCH. No hay restricción de colocar el "breakpoint" en instrucciones de dos bytes, siempre y cuando se coloque al comienzo de la instrucción, o sea, en el primer byte de la misma

La figura 4-1 muestra un esquema del mapeo de memoria en el sistema IMSAI y en la tabla siguiente se muestra los usos particulares que el programa monitor les da a ciertas localidades de memoria.

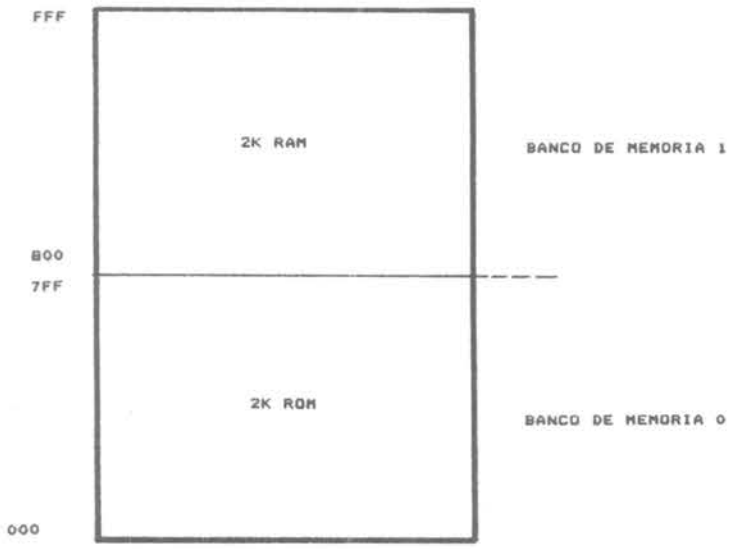


Figura 4-1. Mapa de memoria del sistema IMSAI

TABLA 4-1

USOS PARTICULARES DE LA MEMORIA POR EL MONITOR

LOCALIDAD	DESCRIPCION
FFF	Puerto de control del 8279
FFE	Puerto de datos del 8279
FFD	Reservada para salvar R1 de RB1
FFC	Reservada para salvar R0 de RB1
FEC a FFB	Reservadas para salvar el stack
FEB	Reservada para salvar R7 de RB0
FEA	Reservada para salvar R6 de RB0
FE9	Reservada para salvar R5 de RB0
FE8	Reservada para salvar R4 de RB0
FE7	Reservada para salvar R3 de RB0
FE6	Reservada para salvar R2 de RB0
FE5	Reservada para salvar R1 de RB0
FE4	Reservada para salvar R0 de RB0
FE3	Reservada para el apuntador de stack
FE2	Reservada para el byte de break en la modalidad "breakpoint"
FE1	Reservada para el byte que sigue al de break en modalidad "breakpoint"

4.1.2. TEORIA DE OPERACION

La figura 4-2 es un diagrama de bloques del sistema IMSAI, el cual como se mencionó anteriormente está basado en el microprocesador 8035, operando con un cristal de cuarzo de 3 MHz que le confieren un ciclo de instrucción de 4.1905 microsegundos.

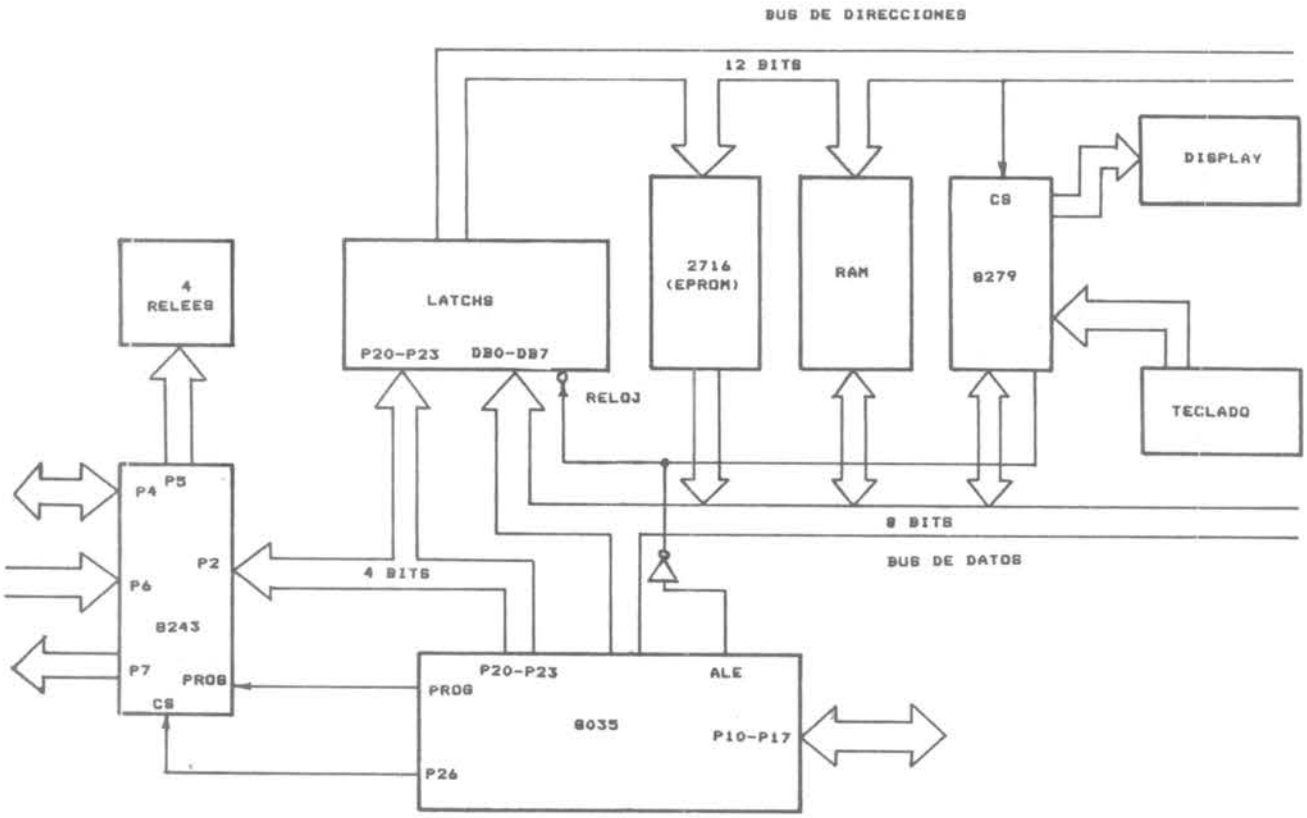


Figura 4-2. Diagrama de bloques del sistema IMSAI

En el capítulo II se vió que el 8035 multiplexa sobre el bus las direcciones y los datos; en virtud de lo cual, se hace necesario el uso de elementos externos para guardar las direcciones y poder disponer así de un verdadero bus de direcciones. En el IMSAI se implementa el bus de direcciones con unos flip flop tipo D, específicamente el circuito integrado 74LS174, que contiene 6 flip flop tipo D por chip; a las entradas D de estos flip flops, se conectan los 8 bits del bus del 8035 (las cuales constituirán los bits menos significativos de la dirección) y los 4 bits menos significativos del puerto 2 del 8035 (los cuales constituirán los 4 bits mas significativos de la

dirección) y se usa la señal ALE (provista por el 8035 para indicar la presencia de una dirección en el bus) para cargar estos datos en los flip flops y obtener así lo que se denomina en el IMSAI el Address Latched Bus (bus de direcciones almacenadas), tal como se muestra en la figura 4-3. A este bus de direcciones se conectan las memorias (RAM y ROM), el manejador de teclado y display 8279 y cualquier otro dispositivo periférico que se desee conectar al sistema.

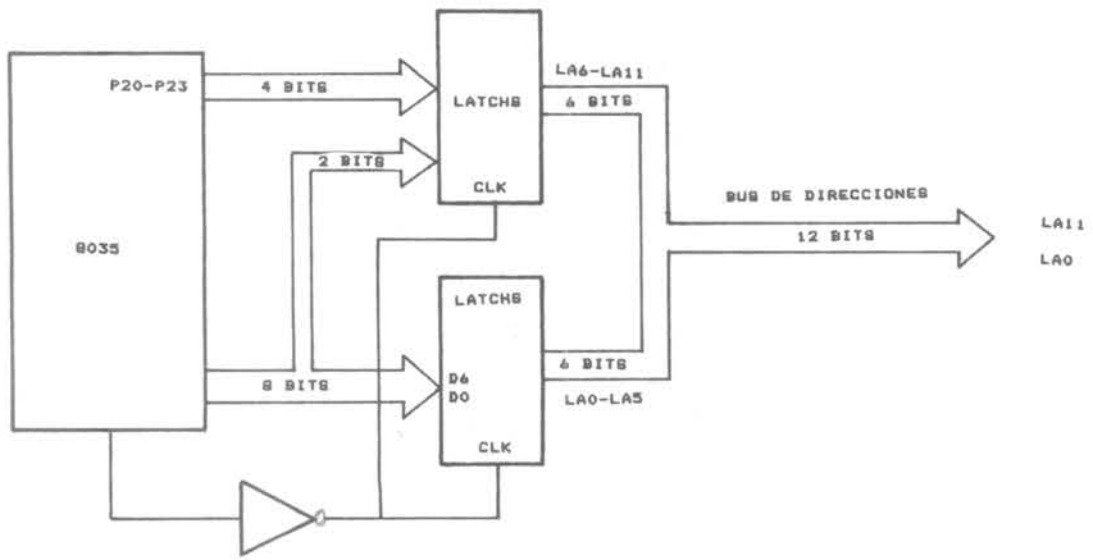


Figura 4-3. Implementación del bus de direcciones en el IMSAI

Para cargar los flip flops se usa el negado de la señal ALE, en virtud de que los flip flops 74LS174 responden al flanco de bajada de la señal de reloj y las direcciones aparecen en el bus del 8035 en el flanco de subida de la señal ALE.

Así como la señal ALE es provista por el 8035 para indicar la presencia de una dirección válida en el bus, las señales "READ" y "WRITE" (salidas $\overline{RD}$ y $\overline{WR}$ del 8035), indican la presencia en el bus del 8035 de un dato válido. La figura 4-4 muestra un diagrama de tiempo de las señales $\overline{RD}$, $\overline{WR}$, ALE y $\overline{PSEN}$ cuando se realizan operaciones de lectura y escritura en memoria externa.

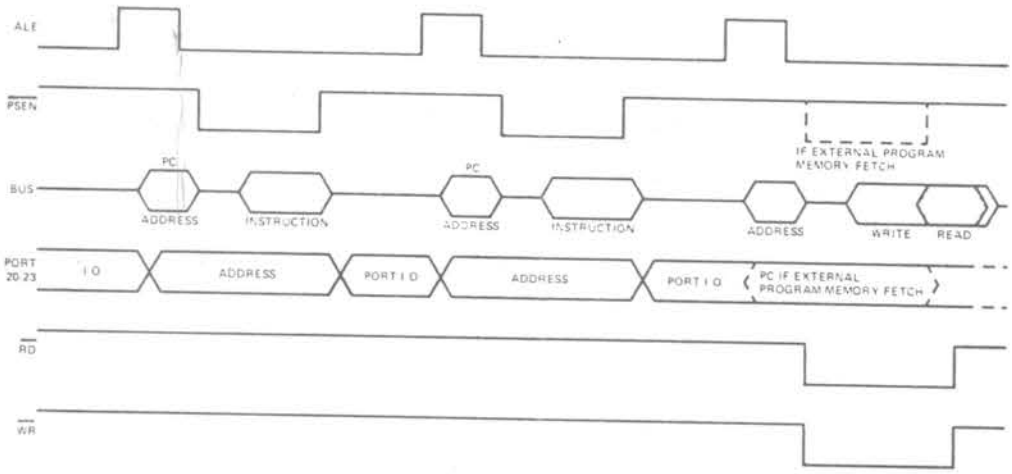


Figura 4-4. Diagrama de tiempo de las señales $\overline{RD}$, $\overline{WR}$, ALE, $\overline{PSEN}$ y el contenido del bus

Cuando el 8035 saca datos por el bus, produce el pulso $\overline{WR}$, el cual en su flanco de subida indica la presencia de un dato válido en el bus; esta señal puede ser aprovechada para sincronizar los periféricos que deseen leer del bus de datos así como tambien, puede ser usada en conjunto con el bus de direcciones para escribir datos en la memoria RAM externa. Las operaciones de lectura del bus, son indicadas por el 8035 con la señal $\overline{RD}$ o $\overline{PSEN}$; La señal $\overline{RD}$ se produce cuando el 8035 va a leer un dato a través del bus; el dato es introducido en el 8035 en el flanco de subida de la señal RD; la señal $\overline{PSEN}$ se produce cuando el 8035 va a leer una instrucción de memoria externa; la instrucción es introducida en el flanco de subida de la señal $\overline{PSEN}$. Las señales $\overline{RD}$ y $\overline{PSEN}$ se usan en el IMSAI de una manera muy particular con la finalidad de permitir al usuario la introducción de instrucciones en memoria y que las mismas sean leídas como tales por el 8035; se mencionó en el capítulo II que las instrucciones que constituyen los programas solo pueden ser leídas desde memoria de programa; ahora bien, el 8035 no posee ninguna instrucción que permita escribir o leer la memoria de programa y la misma solo es leída por el propio microprocesador

en la fase de busqueda y ejecución de las instrucciones (proceso este realizado en forma automática y sin control del usuario). Es necesario por tanto implementar algun mecanismo que permita que los usuarios puedan introducir programas, ejecutarlos y corregirlos; en el IMSAI se solventó este inconveniente colocando la memoria RAM externa, en el mismo espacio de memoria que correspondería a la memoria de programa, utilizando las señales $\overline{RD}$ y $\overline{PSEN}$ para leer indistintamente un dato o una instrucción, en virtud de usar como señal de lectura de memoria, un OR de estas dos señales (READ = $\overline{RD}$ + $\overline{PSEN}$) de manera que el usuario introduce las instrucciones de su programa como datos en la memoria externa y el 8035 ejecuta estos datos (instrucciones), como si fuesen instrucciones en memoria de programa.

El IMSAI, tiene una capacidad de direccionamiento de 4096 bytes, que puede ser extendida, valiendose de la técnica de páginas de memoria, hasta 16 Kbytes como se verá mas adelante. La memoria del IMSAI esta configurada como se muestra en la figura 4-5.

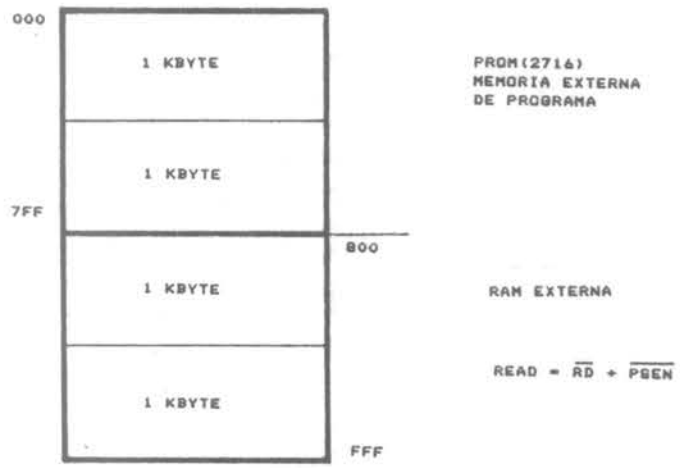


Figura 4-5. Configuración de la memoria del IMSAI

El IMSAI tiene un display de 9 digitos, del tipo de siete segmentos, los cuales son manejados por el controlador 8279, que se describirá en detalle en la sección 4.2.2.2 de este capítulo.

Los digitos son multiplexados aprovechando las salidas S0 a S3 del 8279 y un driver decodificador (7445), el cual permite seleccionar uno a uno cada digito del display. Los bits de cada caracter (cada bit corresponde a un segmento del digito seleccionado) salen por las líneas A0 a A3 y B0 a B3, que

constituyen las salidas de la RAM de display del 8279, tal como se muestra en la figura 4-6; la velocidad a la cual son multiplexados los digitos del display, viene dado por el contador de tiempo interno del 8279, el cual usa como entrada de reloj la señal ALE del 8035, que en el caso específico del IMSAI con un cristal de 3 MHz, produce una rata de multiplexado de 10.3 milisegundos.

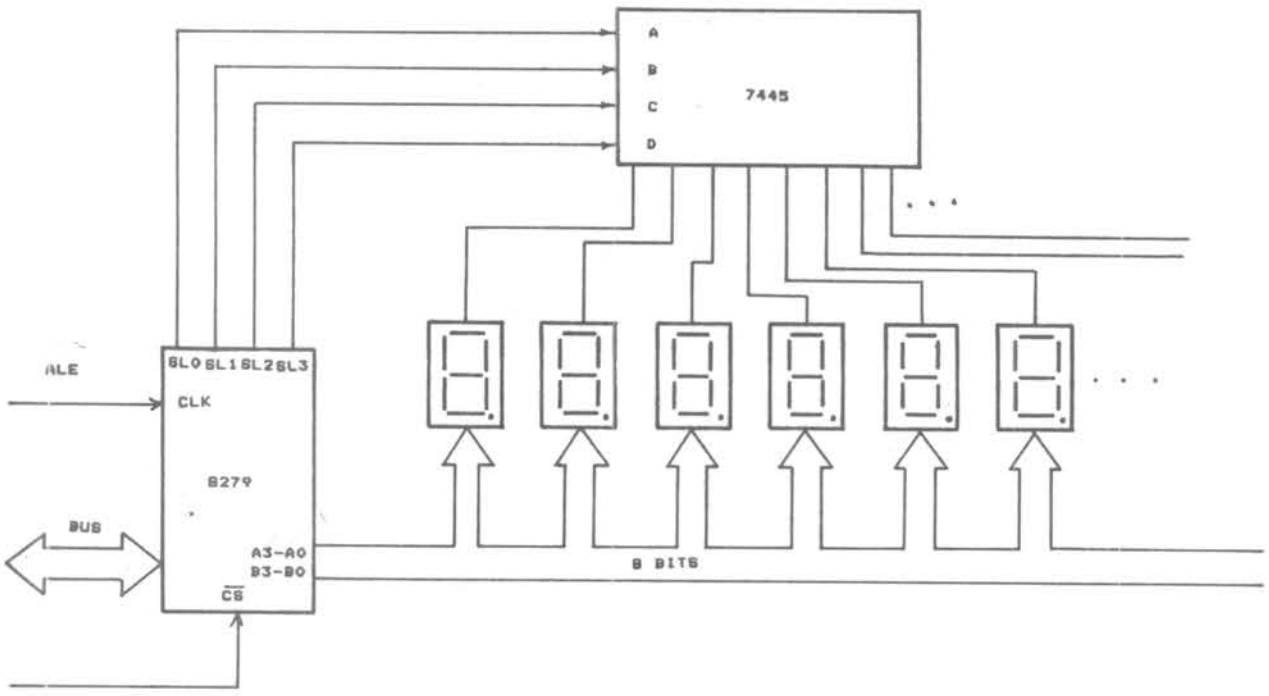


Figura 4-6. Manejo del display en el IMSAI

El IMSAI trae tambien un teclado hexadecimal con teclas adicionales para funciones específicas de uso en el monitor. Para el manejo del teclado se usan las salidas S0 a S2 del 8279 y las entradas R0 a R7 del mismo, en conjunto con un decodificador 7442. Las tres primeras salidas del 7442 (0,1,2), se usan para muestrear las tres filas del teclado; al pulsar una tecla en una cierta fila, una de las entradas R0 a R7 detectará el cierre cuando esa fila sea muestreada; despues de un cierto tiempo para la eliminación del rebote, si la tecla está todavía presionada, el número que representa el código de la tecla (en binario) entra en el registro FIFO del 8279; estas últimas operaciones (eliminación de rebote, muestreo, etc.), son realizas por el 8279 sin la intervención del procesador, como se verá mas adelante en la sección que corresponda a la descripción de esta interfase.

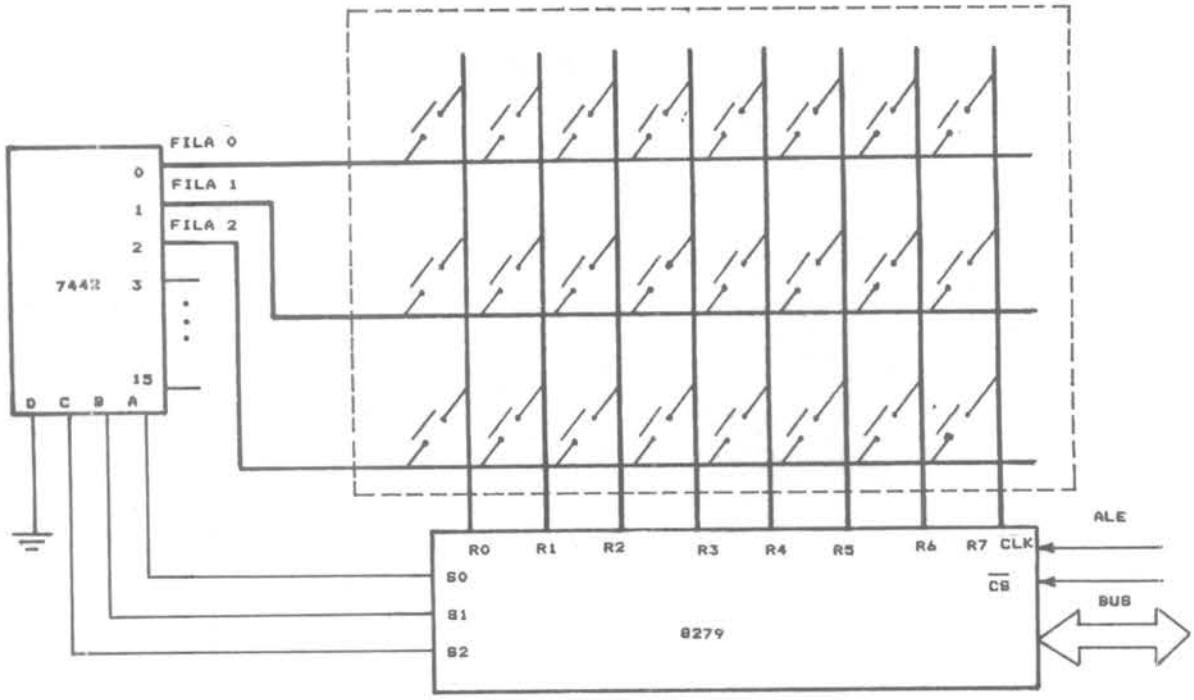


Figura 4-7. Manejo del teclado en el IMSAI

Para la entrada/salida en el IMSAI se usan las 27 líneas de entrada/salida del 8035; de esas 27 líneas 8 corresponden al bus, 16 corresponden a los puertos 1 y 2(8 líneas por cada puerto) y las entradas o puertos de 1 bit constituidos por T0, T1, $\overline{INT}$; adicionalmente trae el expansor de puertos 8243, que permite disponer de 16 líneas adicionales de entrada/salida.

El puerto 1 del 8035, no tiene uso específico en el IMSAI y esta disponible al usuario para entrada/salida en general.

El puerto 2 del 8035 es ampliamente usado en el sistema IMSAI; los dos bits mas significativos del puerto 2 del 8035, P27 y P26 se usan para la conexión de los expansores de puertos; si solo existe un expansor de puertos P26 se conecta a la entrada $\overline{CS}$ del expansor de puertos; estos dos bits del puerto 2(P27 y P26) pueden tambien ser usados por el usuario para salida en general. Las líneas P25 y P24 del puerto 2, están previstas en el IMSAI para usarlas como líneas de "handshaking" cuando se conecte algún periférico en el puerto 1. Las líneas P20 a P23, se usan en el IMSAI para direccionamiento, constituyendo los 4 bits mas significativos de la dirección; estas líneas del puerto 2 se usan tambien en conjunto con la señal PROG para controlar la operación del expansor de puertos 8243.

Las entradas T0 y T1 estan disponibles al usuario para ser chequeadas por software y pueden ser usadas por el usuario para introducir información en forma serial. La entrada $\overline{INT}$ del

8035, es usada en el sistema IMSAI por el monitor del mismo, para la operación en la modalidad "breakpoint", para lo cual dicha entrada está continuamente conectada a tierra a través de un resistor de 10 Kohm; si no se usa el monitor, se puede liberar y usar esta entrada con el sistema de interrupción del 8035, para lo cual el IMSAI trae conexión opcional.

El resto del sistema de entrada salida del IMSAI es implementado con el expansor de puertos 8243. El 8243 provee al IMSAI de cuatro puertos adicionales, de cuatro bits cada uno. El 8035 posee instrucciones para manejar estos puertos que incluyen tanto instrucciones de lectura y escritura sobre dichos puertos como instrucciones para hacer las operaciones lógicas OR y AND de dichos puertos con el acumulador, permitiendo estas últimas el manejo de los bits del puerto individualmente, en una forma relativamente fácil.

El puerto 4 del 8243, no tiene un uso específico en el IMSAI y esta disponible para el usuario para entrada/salida en general.

El puerto 5 del 8243, se usa en el IMSAI para el manejo de 4 de los 5 relees que trae el sistema.

El puerto 6 del 8243 esta especificado en el IMSAI como un puerto de entrada; los bits 0 y 1 de dicho puerto (P60 y P61) son usados para handshaking del puerto 1 del 8035; el bit 2 de este puerto (P62), es usado en el IMSAI como entrada serial para teletipo o terminal, bien sea con interfase de lazo de corriente o RS-232 (no es exactamente una entrada RS-232, ya que los niveles de tensión que maneja dicha entrada, no satisfacen las exigencias de el estandard, por lo que es necesario utilizar elementos adicionales para realizar la adaptación); el bit 3 de este puerto se usa para entrada serial desde cassette.

El puerto 7 del 8243, se usa en el IMSAI como puerto de salida. Los bits 0 y 1 de dicho puerto (P70 y P71), los puede usar el usuario como salida en general o darle un uso específico en la extensión de la capacidad de almacenamiento en el IMSAI; mediante unas conexiones que han sido previstas, estos dos bits del puerto 7 se pueden convertir en los bits A12 y A13 del bus de direcciones y extender la capacidad de direccionamiento del IMSAI hasta 16 Kbytes, mediante la técnica de páginas de memoria. El bit 2 del puerto 7 (P72) del 8243 se usa como salida serial para teletipo o terminal, bien sea con interfase de lazo de corriente o RS-232 (en este caso, al igual que en la entrada, la interfase no satisface los requerimientos de tensión de la RS-232 y es necesario usar elementos adicionales para su adaptación). El bit 3 de este puerto (P73) es la salida serial para cassette.

La tabla 4-2 resume todo lo referente al uso de los diferentes puertos en el sistema IMSAI.

TABLA 4-2

USO DE LOS PUERTOS Y LINEAS DE I/O EN EL SISTEMA IMSAI

Puerto 1 del 8035

P10 a P17	Todos sus bits disponibles para uso del usuario en el conector J1

Puerto 2 del 8035

P20 a P23	Se usan como los bits de mas alto orden del bus de direcciones (A8-A11) se usan tambien para controlar el expansor de puertos 8243
P24 y P25	Se usan para handshaking de los periféricos que se conecten al puerto 1
P26 y P27	Se usan para la conexion del expansor de puertos. Con estos dos bits es posible conectar hasta 4 expansores

Puerto 4 del 8243 sin uso específico

Puerto 5 del 8243 manejo de los relee

Puerto 6 del 8243 puerto de entrada

Puerto 7 del 8243 puerto de salida

P70 y P71	Usables en el IMSAI para la extensión de la capacidad de almacenamiento del mismo, constituyendo los bits A12 y A13 del bus de direcciones; pueden ser usadas por el usuario para otros propósitos
P72	Se usa como salida serial para teletipo con interfase lazo de corriente o RS-232
P73	Se usa como salida serial para cassette. Normalmente es conectada a la entrada "MICROPHONE" del reproductor

4.1.3. CONEXIONES DISPONIBLES EN EL IMSAI

Para usar el IMSAI se requiere conectar el pin 2 del conector 2 a una fuente regulada de +5V y el pin 3 de ese mismo conector a tierra. Alternativamente se puede alimentar el IMSAI con una batería de 6V, colocando el terminal positivo en el pin 1 del conector 2 y el terminal negativo de dicha batería al pin 3 del mismo conector.

Ademas de este conector, el IMSAI posee cinco conectores adicionales, identificados como J1, J3, J4, J5 y J6. A continuación se explicaran de que líneas se dispone en cada conector.

En el conector J1 se encuentran las señales para las expansiones del bus. Es un conector de 50 pines y a el llegan: parte baja del bus de direcciones A0-A7 (pines 11-18, señales LAD0-LAD7); la parte alta del bus de direcciones A8-A11 (pines 19-22, señales LAD8-LAD11); y la extensión del bus de direcciones A12 y A13 (pines 24 y 25). En el plano del IMSAI las señales que van a este conector se identifican con un circulo (○).

El conector J3 tiene 23 pines y en el plano del IMSAI las señales que van a este conector se identifican con un cuadrado (□). Este conector se usa para conectar un puerto de entrada/salida con señales para handshaking. El puerto 1 del 8035 (P10 a P17), está conectado entre los pines 3 y 18 de este conector; cada bit del puerto ocupa dos pines del conector (P10 pines 3 y 4, P11 pines 5 y 6 ...etc.). Los bits 4 y 5 del puerto 2 del 8035, P24 y P25, están conectados a los pines 19 y 20 de este conector, y se usarán como señales de salida para el handshaking. Los bits 0 y 1 del puerto 6 del 8243, P60 y P61, están conectados a los pines 21 y 22, y se usaran como líneas de entrada para el handshaking.

El conector J4 tiene 26 pines y las señales conectadas a el son identificadas en el plano con el simbolo ⊏> . Los pines 20 y 22 corresponden a la entrada y salida para teletipo con interfase RS-232; los pines 23-24 y 25-26 corresponden a la salida y entrada respectivamente para teletipo con interfase lazo de corriente. Los pines 16 y 18 de este conector corresponden a la entrada y la salida respectivamente de cassette. Los pines 8 y 9 corresponden a las señales T0 y T1 respectivamente.

El conector J5 tiene los contactos para 4 de los 5 relees del IMSAI. En el conector J6 se encuentran los contactos y la bobina del quinto relee del IMSAI. La tabla 4-3 resume todas las conexiones de los conectores del IMSAI.

TABLA 4-3

ALAMBRADO DE LOS DIFERENTES CONECTORES DEL IMSAI

CONECTOR	PIN	USO
J1	11 a 18	Parte baja del bus de dereccio-nes(LAD0 a LAD7)
J1	19 a 22	Parte alta del bus de direccio-nes(LAD8 a LAD11)
J1	24 y 25	Extension del bus de direcciones (LAD12 y LAD13)
J1	39 y 40	Bits 6 y 7 del puerto 2 del 8035 (P26-P27)
J2	1	Entrada de alimentación del IM-SAI con bateria de +6V
J2	2	Entrada de alimentación del IM-SAI para fuente regulada de +5V
J2	3	Tierra o negativo de la batería
J3	4 y 3	Bit 0 del puerto 1 (P10)
J3	5 y 6	Bit 1 del puerto 1 (P11)
J3	7 y 8	Bit 2 del puerto 1 (P12)
J3	9 y 10	Bit 3 del puerto 1 (P13)
J3	11 y 12	Bit 4 del puerto 1 (P14)
J3	13 y 14	Bit 5 del puerto 1 (P15)
J3	15 y 16	Bit 6 del puerto 1 (P16)
J3	17 y 18	Bit 7 del puerto 1 (P17)
J3	19	Bit 4 del puerto 2 (P24)
J3	20	Bit 5 del puerto 2 (P25)
J3	21	Bit 0 del puerto 6 del 8243(P60)
J3	22	Bit 1 del puerto 6 del 8243(P61)

J4	1	Tierra
J4	2 a 5	Puerto 4 del 8243 (P40-P43)
J4	8	Entrada T0 del 8035
J4	9	Entrada T1 del 8035
J4	10	Bit 6 del puerto 2 (P26)
J4	11	Bit 7 del puerto 2 (P27)
J4	12	Bit 2 del puerto 5 del 8243(P52)
J4	13	Bit 3 del puerto 5 del 8243(P53)
J4	14	Bit 0 del puerto 5 del 8243(P50)
J4	15	Bit 1 del puerto 5 del 8243(P51)
J4	16	Bit 3 del puerto 6 del 8243(P63) entrada de cassette
J4	17	tierra
J4	18	Bit 3 del puerto 7 del 8243(P73) salida para cassette
J4	19	Tierra
J4	20	Bit 2 del puerto 6 del 8243(P62) entrada de teletipo con RS-232
J4	21	Tierra
J4	22	Bit 2 del puerto 7 del 8243(P72) Salida para teletipo con RS-232
J4	23 y 24	Salida para teletipo con lazo de corriente(23=+ y 24=-); va a P72
J4	25 y 26	Entrada de teletipo con lazo de corriente(25=+ y 26=-); va a P62
J5	1,2,3	Contactos del relee K3; 1 y 2 NC , 3 y 2 NA
J5	7,8,9	Contactos del relee K4; 7 y 8 NC , 9 y 8 NA
J5	13,14,15	Contactos relee K2; 13 y 14 NC , 15 y 14 NA

J5	19,20,21	Contactos relee K1; 19 y 20 NC , 21 y 20 NA
J6	2,3,5,6	Bobina y contactos del relee K5; 2 bobina; 5 y 3 NC; 6 y 3 NA

4.2. INTERFASES DE ENTRADA Y SALIDA PARALELA

Como se explicó anteriormente, el IMSAI posee el expansor de puertos 8243 y el manejador de teclado y display 8279. Estas dos interfases proveen al IMSAI de entrada y salida paralela, son programables y se usarán extensamente en este curso.

4.2.1. EXPANSOR DE PUERTOS 8243

EL 8243 es un expansor de puertos, diseñado para proveer de puertos extras a la familia de microprocesadores MCS-48. Se fabrica con tecnología NMOS y requiere de solo una fuente de alimentación (+5V); constituye una buena alternativa para la expansión de los sistemas basados en los microprocesadores de la familia MCS-48, por su costo relativamente bajo, su configuración y modo de operación que practicamente lo hacen aparecer como un puerto externo de los microprocesadores de dicha familia, los cuales están dotados de instrucciones (que forman parte del set de instrucciones de los microprocesadores) para operar directamente con el 8243, sumandose a ello el hecho de que sus salidas tienen una capacidad de corriente relativamente alta (hasta 20 miliamperios en algunos puertos).

El 8243 cuenta con 4 puertos estáticos bidireccionales de cuatro bits cada uno y un puerto adicional de cuatro bits que sirve de interfase con los microprocesadores de la familia MCS-48.

4.2.1.1. Descripción del 8243

En la figura 4-8 se muestra un diagrama de bloques funcional del 8243; se observa en este diagrama el puerto 2 bidireccional de cuatro bits, el cual a través de un multiplexer llega a un bus interno del 8243. A este bus están conectados el decodificador de instrucciones y el decodificador de direcciones, tanto las direcciones como las instrucciones para su operación entran al 8243 por el puerto 2; al bus tambien está conectado un registro temporal, el cual es una de las entradas al circuito lógico donde se realizan las operaciones de AND y OR, bajo control del decodificador de instrucciones. La otra entrada a este circuito de AND/OR proviene del registro latch del puerto direccionado. La salida del circuito de AND/OR se pasa al puerto direccionado despues que se haya ejecutado la operación seleccionada.

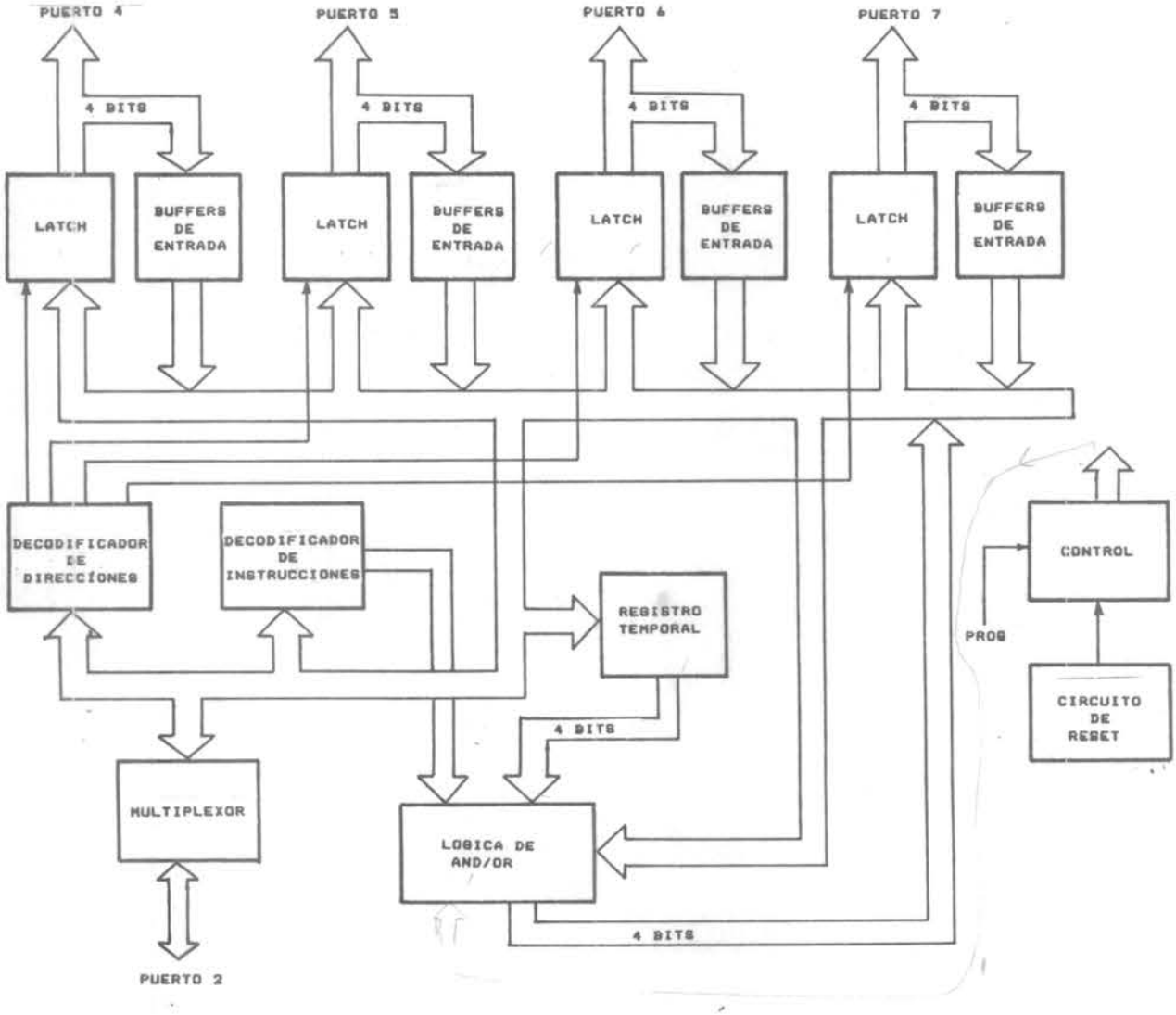


Figura 4-8. Diagrama de bloques del 8243

Es interesante hacer notar que cada puerto del 8243 tiene un registro latch para sacar y mantener datos por dicho puerto y ademas posee un buffer de entrada, para la lectura de datos a través de ese puerto. Cuando se hacen operaciones de lectura a través del puerto, los latch de salida permanecen en el estado de alta impedancia (deshabilitados).

El circuito de control tiene como entradas la señal PROG, la señal de RESET y la señal $\overline{CS}$; La señal de PROG se usa para indicar al 8243 el tipo de información presente en el puerto 2, una instrucción-dirección o un dato. La señal de RESET cuando se activa hace que la unidad de control inicialize el 8243, lo cual hace que el puerto 2 se coloque en el modo de entrada y los puertos 4,5,6 y 7 se coloquen en el modo de alta impedancia. A continuación una descripción funcional del expansor de puertos (ver figura 4-8):

El 8243 como ya se dijo antes, tiene cuatro puertos de E/S, los cuales sirven como extensión a la E/S de los microprocesadores de la familia MCS-48; estos puertos están identificados con los números 4,5,6 y 7 pudiendo realizarce sobre ellos las siguientes operaciones:

- Transferencia del acumulador a los puertos
- Transferencia de los puertos al acumulador
- Operación AND entre el acumulador y puertos
- Operación OR entre el acumulador y puertos

Toda la comunicación entre el microprocesador y el expañsor de puertos 8243 ocurre a través del puerto 2 (P20-P23), sincronizandose las misma por medio de la señal PROG. Cada transferencia de información entre un puerto y el microprocesador consta de dos nibbles (paquetes de 4 bits); el primer nibble, se escribe sobre el puerto 2 y contiene el código de la operación que se desea realizar y la dirección del puerto con la que se desea realizar dicha operación; el segundo nibble contendrá los cuatro bits de datos los cuales, dependiendo de la operación, serán escritos sobre el puerto 2 para posteriomente ser transferidos al puerto direccionado, o serán leídos desde el puerto direccionado a través del puerto 2; la figura 4-9 muestra el formato del primer nibble de la transferencia y el significado de cada bit.

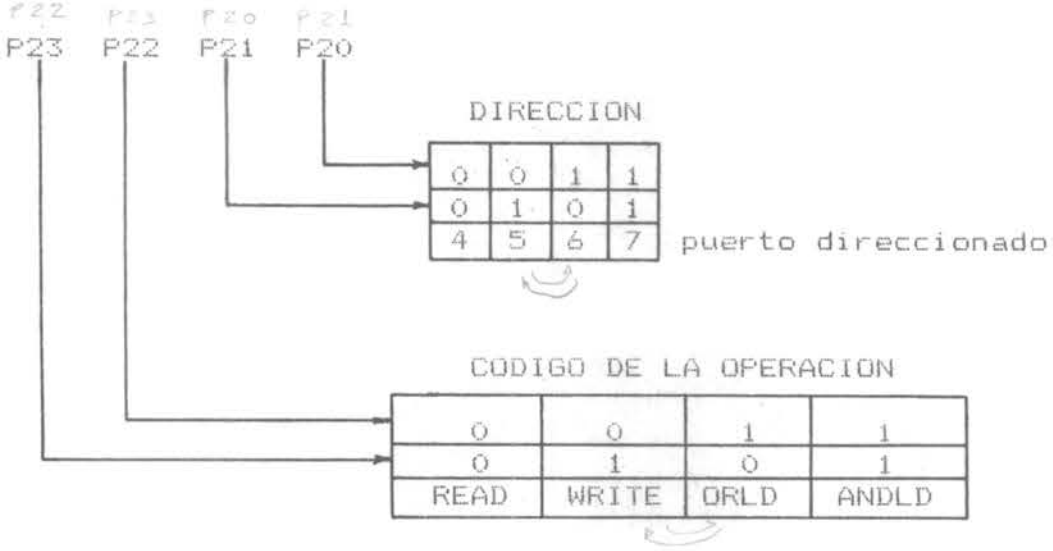


Figura 4-9. Formato del nibble de programación del 8243

El 8243 puede operar en dos modos diferentes: en el modo de escritura y en el modo de lectura.

a) Modo de escritura del 8243

El 8243 permite tres formas diferentes de escribir en sus puertos:

Escritura directa sobre el puerto direccionado mediante la instrucción del microprocesador MOVD Pi, A; en esta forma de escritura sobre los puertos, el dato colocado en el puerto 2 (interfase con el microprocesador), se vacía sobre el puerto seleccionado (Pi), reemplazando al dato anterior que se encontraba en dicho puerto.

Esritura sobre el puerto mediante la operación OR; en esta modalidad se usa la instrucción del microprocesador ORLD Pi, A; el dato que se escribe en el puerto 2 es tomado por el 8243 y la lógica de AND/OR del mismo le hace un OR con el dato presente en el puerto direccionado (Pi) y el resultado de esta operación es vaciado en dicho puerto.

Escritura sobre el puerto mediante la operación AND; en esta modalidad se usa la instrucción del microprocesador ANDLD Pi, A; el dato que se escribe en el puerto 2 es tomado por el 8243 y mediante la lógica de AND/OR del mismo, se le hace un AND con el dato presente en el puerto direccionado (Pi) y el resultado es vaciado en dicho puerto.

El código de operación (OUT, OR, AND) y la dirección

del puerto (P4, P5, P6, P7), son almacenados en el decodificador de instrucción y decodificador de direcciones respectivamente, desde el puerto de interfase (puerto 2), en el flanco de bajada de la señal PROG; los datos se transfieren desde el puerto de interfase al puerto direccionado, directamente o a través de la lógica de AND/OR, en el flanco de subida de la señal PROG.

Si la instrucción de escritura es un AND u OR, mientras el 8243 realiza la operación, el dato viejo que estaba en el puerto direccionado, permanece allí hasta tanto se ejecute la operación especificada, momento en el cual, se transfiere el resultado de dicha operación al puerto direccionado. La figura 4-10 muestra un diagrama de tiempo de las señales PROG, CS y los puertos, en una operación de escritura sobre los mismos.

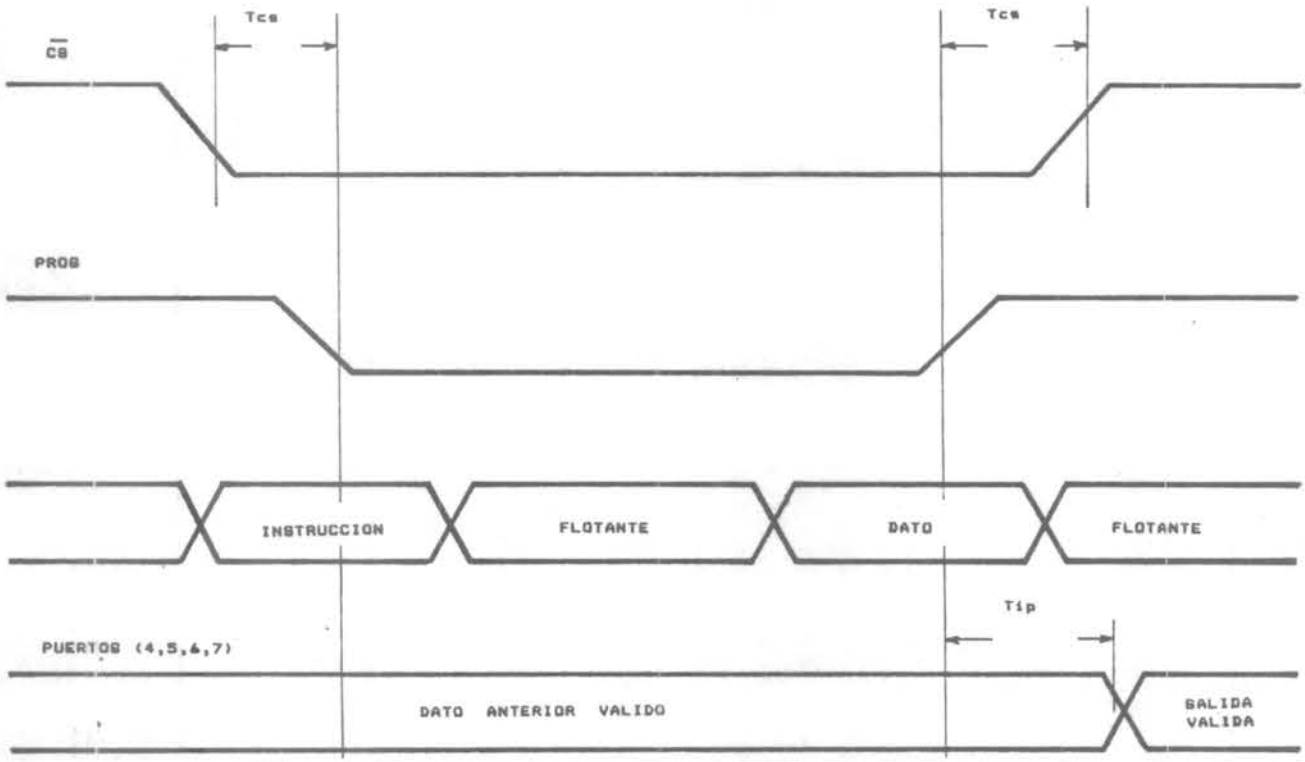


Figura 4-10. Diagrama de tiempo en una operación de escritura en un puerto del 8243

b) Modo de lectura del 8243

El 8243 tiene un solo modo de lectura. El código de operación y la dirección del puerto son tomadas del puerto 2 de entrada y almacenadas en el flanco de bajada del pulso aplicado en PROG. Tan pronto como el tipo de operación (lectura) y la dirección del puerto son decodificadas, las salidas del puerto especificado son puestas en estado de alta impedancia y el buffer de entrada del puerto direccionado es habilitado; La operación de

lectura termína con el flanco de subida del pulso aplicado en PROG. El puerto que se seleccione (4,5,6 o 7), una vez terminada la operación de lectura, retorna al estado de alta impedancia y el puerto 2 (interfase con el microprocesador), retorna a su modo de entrada. La figura 4-11 muestra un diagrama de tiempo de las señales PROG, $\overline{CS}$, y los datos en los puertos, en una operación de lectura de un puerto.

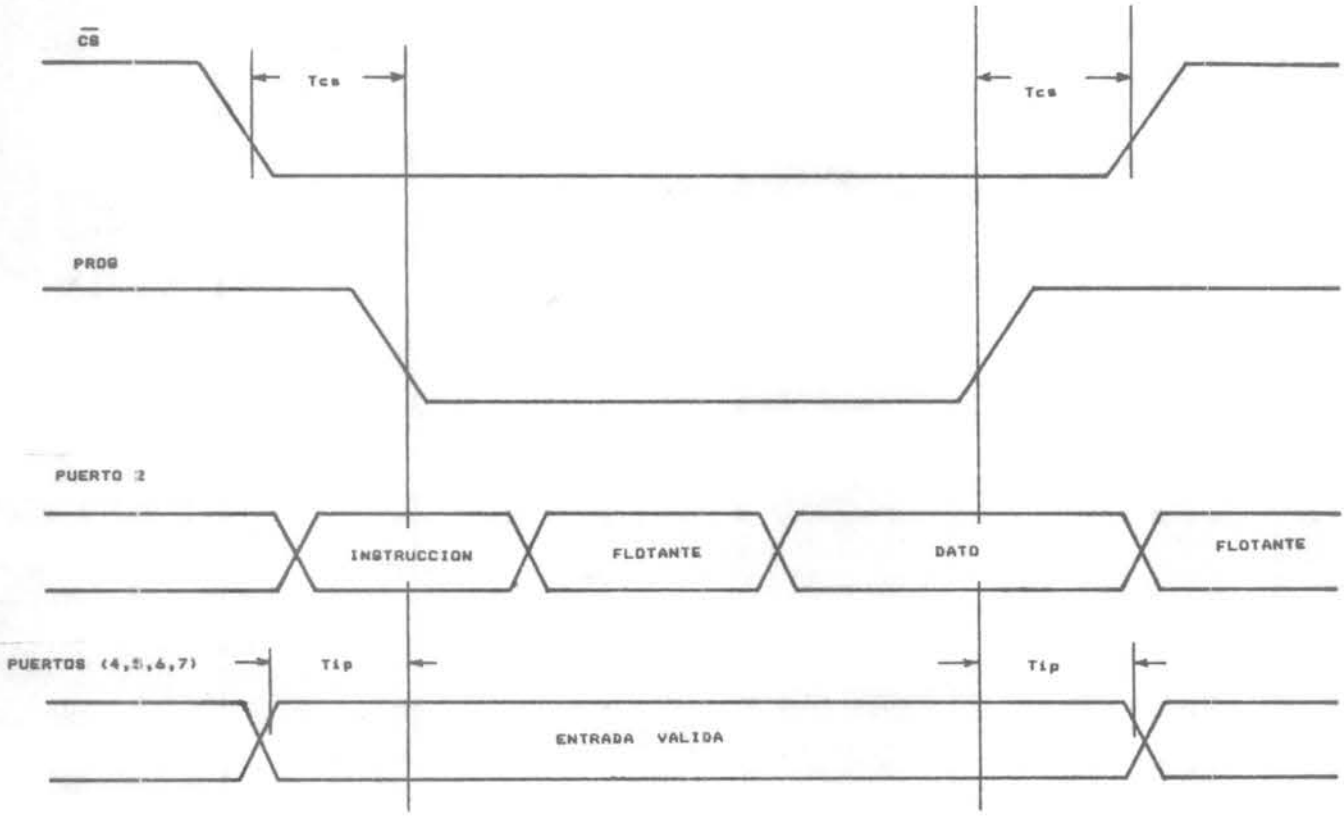


Figura 4-11. Diagrama de tiempo en una operación de lectura de un puerto del 8243

Los puertos del 8243 permanecen en el modo de operación que se especificó en la última operación que se realizó sobre los mismos (en el modo de escritura, los latch de salida permanecen habilitados y el buffer de entrada permanece en el estado de alta impedancia; en el modo de lectura los latch de salida permanecen deshabilitados y el buffer de entrada permanece en el estado de alta impedancia); si se cambia de modo de operación, entre dos instrucciones sucesivas de E/S sobre un determinado puerto, la primera lectura que sigue a una operación de escritura debe ser ignorada, a fin de permitir que el latch de salida del puerto involucrado se deshabilite y evitar posibles lecturas erroneas, ocasionadas por el dato que se encuentre almacenado en los latch de salida.

La figura 4-12 es un diagrama de pines del 8243 y a continuación se resume la funcion de cada uno de ellos.

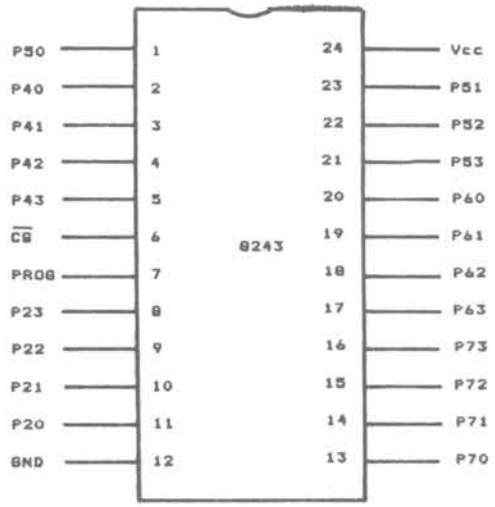


Figura 4-12. Diagrama de pines del 8243

TABLA 4-4

DESCRIPCION DE LOS PINES DEL 8243

PIN	DESCRIPCION
6	$\overline{CS}$: entrada para selección del 8243; $\overline{CS}$=0 seleccionado el 8243; $\overline{CS}$=1 deshabilitado el 8243
7	PROG: entrada de reloj; permite conocer la disponibilidad de datos o información de control en el puerto de interfase con el microprocesador (P20-P23). Flanco de bajada de PROG ==> dirección y control en el puerto 2; flanco de subida de PROG ==> datos en el puerto 2 del 8243
12	GND: tierra del 8243; se conecta a 0 volts
24	V_{cc}: alimentación del 8243; se conecta a +5 volts
2-5	P40-P43: puerto 4, bidireccional (4 bits)
20-17	P60-P63: puerto 6, bidireccional (4 bits)

13-16 P70-P73: puerto 7, bidireccional (4 bits)

1,23-21 P50-P53: puerto 5, bidireccional (4 bits)

11-8 P20-P23: puerto bidireccional, sirve de interfase con el microprocesador (4 bits). La naturaleza de la información presente en este puerto (control o datos), lo determina la entrada PROG

4.2.2. MANEJADOR DE TECLADO Y DISPLAY 8279

EL 8279 es una interfase programable de E/S para manejar teclado y display. La parte de teclado puede ser usada para muestrear una matriz de 64 teclas pudiendo expandirse para muestrear un máximo de 128 teclas; a la entrada de la sección de teclado se le elimina rebote y los caracteres recibidos por estas entradas se almacenan en una memoria FIFO interna del 8279, en donde caben un máximo de 8 caracteres; si se escriben mas de ocho caracteres, se producirá un error de overrun, indicación esta que pudiera ser usada para generar una interrupción al microprocesador.

La porción de display provee una interfase para muestrear un display de LED. El 8279 tiene una RAM de display de 16x8; esta RAM puede ser cargada o interrogada por el microprocesador. Los dos formatos tipicamente usados en el display de caracteres, entrada por la derecha o entrada por la izquierda, pueden ser seleccionados por software en el 8279. Tanto la lectura como la escritura del RAM de display se puede hacer con autoincremento de la dirección de display (a cada lectura o escritura sobre la RAM, se incrementa automáticamente el registro de direcciones de la RAM)

Antes de entrar en la descripción propiamente dicha del 8279 se dará una pequeña introducción a la teoría básica de teclado y display.

4.2.2.1. Teoría básica de teclado y display

a) Teoría básica sobre teclado

Un teclado es un conjunto de switches, en donde cada switche representa para el usuario una letra, una función, un caracter de control, etc. mientras que para el microprocesador cada tecla representa un conjunto de unos y ceros, los cuales pueden constituir un caracter en determinado código (generalmente ASCII) o bien representar la posición relatíva de dicha tecla; en ambos casos, el microprocesador manipúla la información proveniente del teclado, a través de programas y tablas, a fín de convertirla al código o acción que el usuario desee que represente.

En la figura 4-13 se muestra una conexión de una serie de switches a un puerto de un microprocesador constituyendose en un teclado rudimentario. Por medio de lecturas del puerto 1, bajo control de un programa, el microprocesador determina en primer lugar si alguno de los switches está cerrado, lo cual ocasionará que alguno de los bits del caracter leído por el puerto, sea cero (el que corresponda al switche cerrado) y en segundo lugar, despues de haber detectado que un switche fue cerrado, identifíca al switche que fue cerrado, mediante la determinación de cual de los bits del caracter leído a través del puerto es cero.

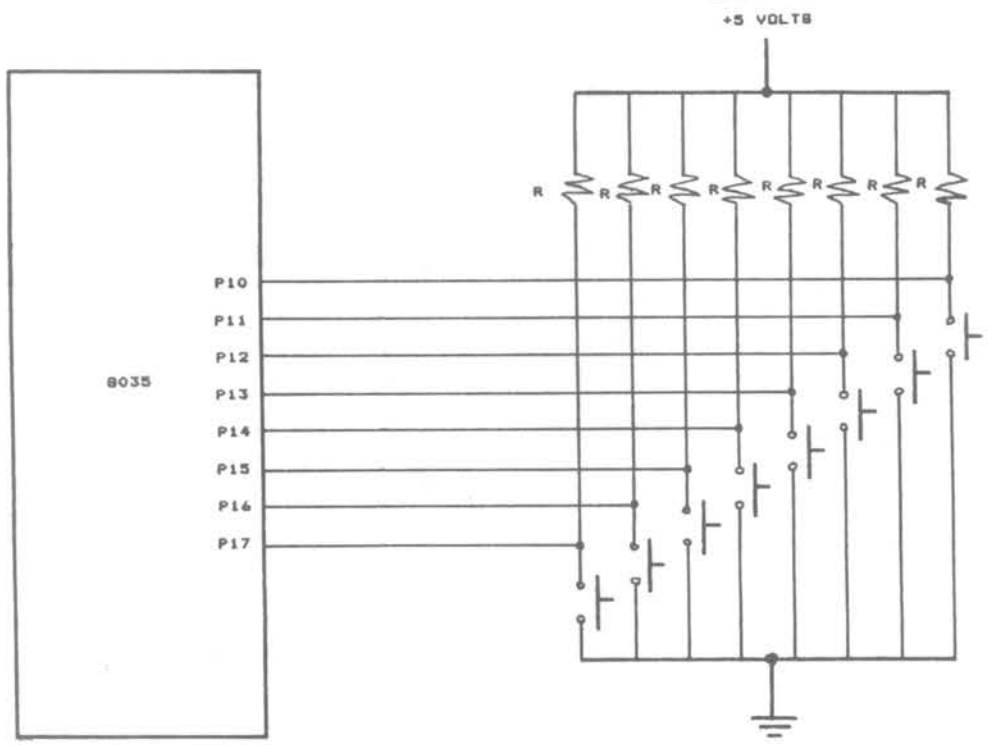


Figura 4-13. Conexión de un arreglo de switches a un puerto del 8035

A continuación se dara un programa, hecho con los mnemónicos del 8035, que permitiría en el arreglo de switches de la figura 4-13, realizar las acciones que se acaban de mencionar: detección de switche cerrado e identificación del mismo.

PROGRAMA PARA MANEJO DEL ARREGLO DE SWITCHES

```
ETIQUE.  MNEMONICO        COMENTARIO

% detección de switche cerrado
ESPERE   INS A,P1         lee puerto 1(salva en acumulador)

         MOV R1,A         salva acumulador en el registro R1

         XOR A,FF         OR exclusivo del acumulador con el
                          caracter FF(H). si el caracter
                          en el acumulador no tiene ningún
                          bit en cero (switches abiertos) el
                          resultado de esta operación será
                          A=00(H); si el caracter tiene
                          algun bit en cero(switche cerrado)
                          A será distinto de cero.

         JZ ESPERE        si A=00 regresa a ESPERE
% fin de la detección de switche cerrado

% identificación del switche cerrado

         MOV A,R1         carga de nuevo el dato leído en A
                          (salvado anteriormente en R1)

         MOV R1,#07       inicializa R1 para usarlo como con
                          tador de bit del caracter

LOOP     RLC A            rota A a la izquierda a través del
                          carry

         JNC SWITCHE      C=0 ==> bit en prueba es cero (en
                          R1 esta el número del bit)

         DJNZ R1,LOOP     decrementa el contador de bit, R1
                          y va al lazo de identificación

SWITCHE  ...
```

Las acciones que se tomen en el programa, posterior a la identificación del switche cerrado, dependerán de lo que el programador desee que represente ese determinado switche.

Los teclados pueden ser organizados en dos configuraciones básicas:

Switches conectados independientemente

Switches organizados en forma matricial

Switches conectados independientemente

El esquema de la figura 4-13, conocido como el de teclado con switches independientes, tiene varias desventajas: en primer lugar el número de líneas requeridas será igual al número de switches; en segundo lugar, se requiere del uso del microprocesador para la detección de switche cerrado; en tercer lugar, existe el problema del rebote en los switches. En el caso de requerirse pocas teclas, el bajo costo de este esquema constituye su principal ventaja que compensaría las desventajas que se acaban de mencionar.

La figura 4-14 muestra el fenómeno que se produce cuando se cierra un switche mecánico, fenómeno este que se ha denominado rebote. El rebote, se debe a la naturaleza mecánica del switche, el cual, no cierra completamente en un primer momento, si no que "rebota", abriendose y cerrandose con un movimiento amortiguado, que se mantiene por un periodo de varios milisegundos; este tiempo es lo suficientemente grande como para que el microprocesador lo interprete como multiples cerradas y aperturas del switche.

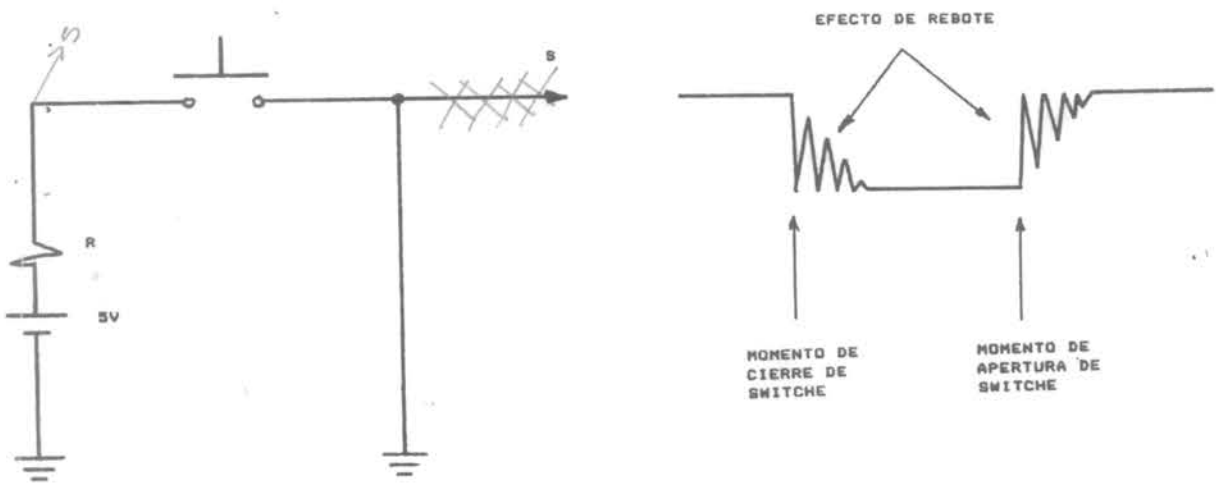


Figura 4-14. Fenómeno del rebote en los contactos de un switche

El problema que se presenta por el fenomeno de rebote de los switches, se puede resolver bien sea por hardware o por software. En el caso de adoptar una solución por hardware, existen varias alternativas, una de las cuales se muestra en la figura 4-15.

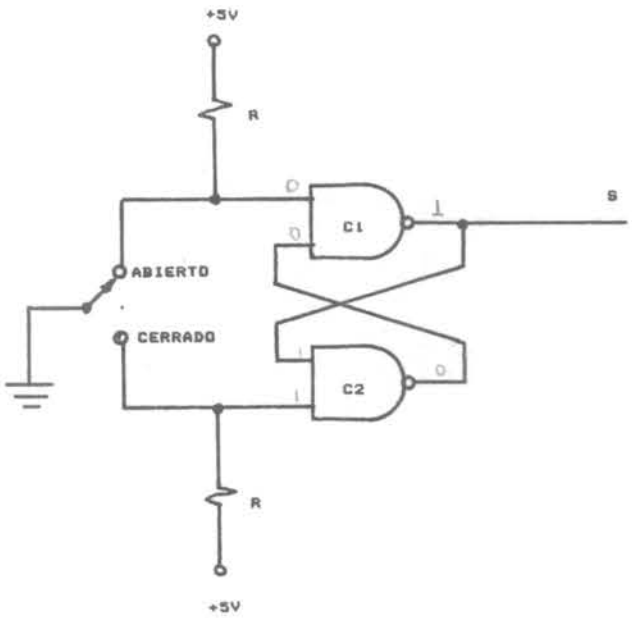


Figura 4-15. Circuito para eliminar el rebote producido por un switche

En el circuito de la figura 4-15, cuando el switche está en la posición "abierto", la salida S permanece en el estado 1; al momento de pasar el switche a la posición "cerrado", al primer rebote, la salida S cambiará a cero y los subsiguientes rebotes no afectarán dicha salida, ya que el cero de la misma, realimentado a la compuerta 2, los bloqueará; Cuando el switche se retorna a la posición "abierto", al primer rebote la salida S va a 1, lo que genera un cero en la salida de la compuerta 2, que al realimentarse a la compuerta 1, bloqueara los subsiguientes rebotes.

Si se adopta la solución por software, el rebote puede ser eliminado mediante la incorporación en el programa de lectura del teclado, de un lazo de retardo, suficiente para dejar que el switche se estabilice (en general 20 milisegundos son suficientes), antes de leer efectivamente el dato desde el teclado. Un diagrama de flujo de un programa completo para el manejo del juego de switches de la figura 4-13 se muestra en la figura 4-16. En una primera parte, el programa entra en el lazo

de detección de cierre de algún switche; al momento de detectar el cierre de un switche, antes de pasar a la determinación de cual switche fue cerrado, se introduce un lazo de retardo de aproximadamente 20 milisegundos, despues de lo cual se procede a determinar si el switche todavía está cerrado, en cuyo caso se da por válida la detección de switche cerrado y se procede a la identificación del switche que se cerró.

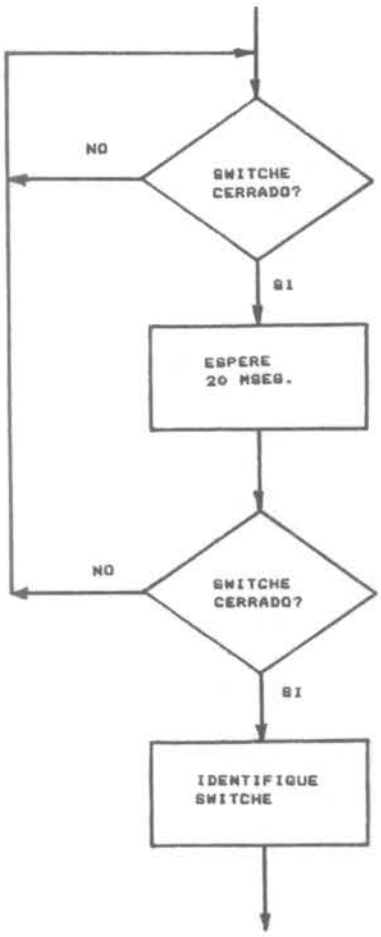


Figura 4-16. Diagrama de flujo de un programa para lectura de switches con eliminación de rebote por software

Switches organizados en forma matricial

Este tipo de organización tiene varias ventajas respecto de la organización como switches independientes, siendo quizas la mas importante la considerable reducción en las líneas requeridas para la conexión de un determinado número de switches. En este tipo de organización, cada switche se identifica por una

combinación única de número de columna y número de fila. La figura 4-17 representa un arreglo matricial de 32 switches que permitiría disponer de hasta 32 teclas diferentes.

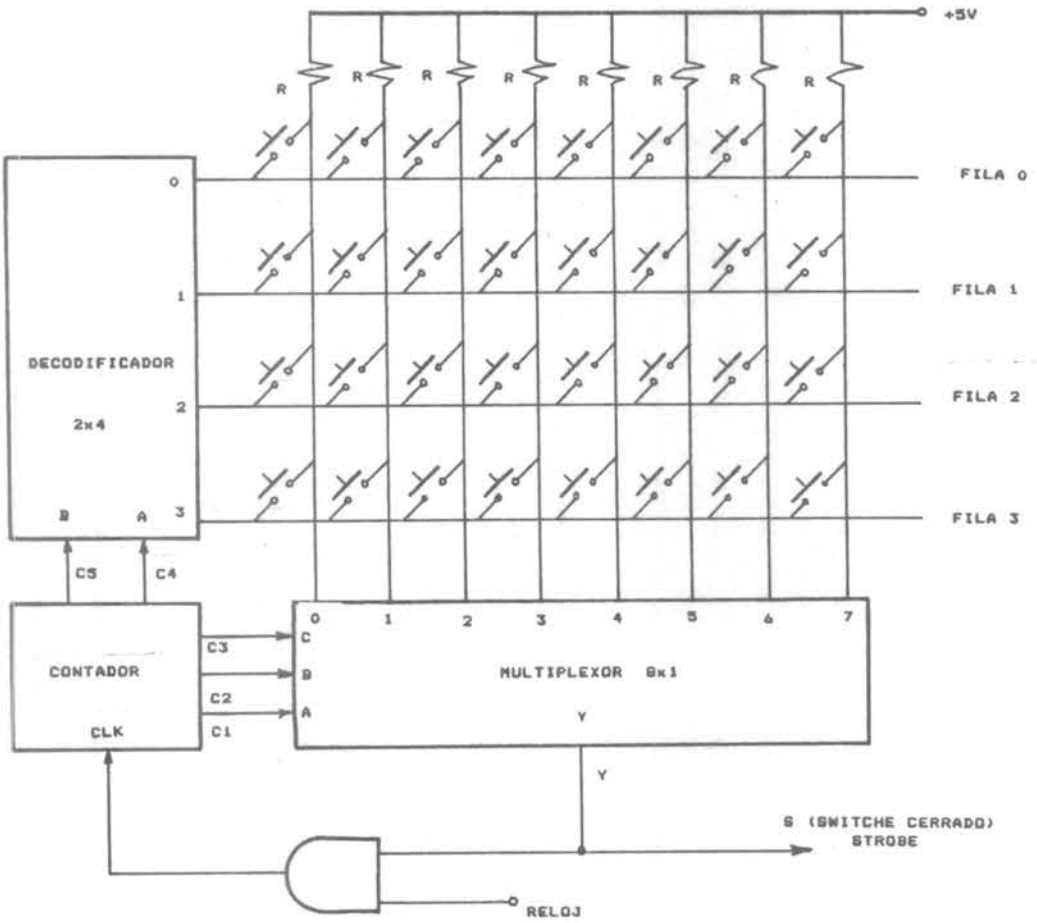


Figura 4-17. Switches conectados en forma matricial

El contador permanentemente está contando desde 0 a 31 cíclicamente, mientras no exista ningún switche cerrado. Las salidas mas significativas del contador (C4,C5) se aplican al decodificador; cada línea de salida del decodificador constituye una fila y su operación es tal que la salida seleccionada por las entradas A,B (C4,C5), se pone en cero; de manera que, mientras el contador esté operando, se irá colocando en forma secuencial un cero en cada fila.

Al multiplexor se le aplican las salidas menos significativas del contador (C1,C2,C3); este multiplexor se caracteriza por sacar por la salida "Y", el dato de la entrada seleccionada (cero o uno) por las líneas A,B,C del multiplexor. La operación del circuito es como sigue:

Al iniciar la operación, el contador selecciona una fila a través de las líneas A,B (C4,C5) del decodificador, con lo cual aparece un cero en dicha fila; el estado de los switches conectados entre cualquier columna y la fila seleccionada es determinado muestreando cada una de ellas, con las salidas menos significativas del contador aplicadas al multiplexor. Si algún switche conectado a la fila seleccionada se cierra, en la entrada respectiva del multiplexor aparecerá un cero y cuando esta línea sea seleccionada por las entradas A,B,C del multiplexor, aparecerá dicha entrada (cero lógico) reflejada en la salida "Y" del mismo, lo cual inhibirá la entrada de reloj del contador, deteniendose en consecuencia su operación, permaneciendo el contador detenido en la fila y columna del switche que se presionó, de manera que, cuando la cuenta se detiene por el cierre de un switche cualquiera, la salida de los contadores representa la fila y la columna del switche que se cerró. La salida "Y" del multiplexor puede ser usada para "avisar" al microprocesador que un switche fue cerrado.

Este esquema de organización, mostrado en la figura 4-17 tiene la ventaja respecto de los switches independientes que la detección de switche cerrado es automáticamente determinado por el mismo circuito y el microprocesador solo tendrá que leer el valor de los contadores (fila, columna) y con ayuda de una tabla, determinar la acción o efecto a tomar como respuesta al cierre del switche.

Es de hacer notar que este circuito presenta el mismo inconveniente del rebote de los switches y en este caso tambien se puede obviar este inconveniente usando alguna técnica de software o por hardware, tal como se explicó en líneas anteriores.

Existe una variante de la organización matricial de switches, que permite "entregar" al microprocesador, la información proveniente del arreglo de switches directamente en un código determinado, tal como ASCII o BAUDOT; este tipo de arreglo de switches, que entregan la información directamente en un código, se conoce como TECLADO CODIFICADO y el tipo anterior que se acaba de describir, en el cual la información no se entrega codificada, sino que el microprocesador se encarga de hacerlo, se conoce con el nombre de TECLADO NO CODIFICADO.

Un arreglo de switches que permite entregar la información directamente en código ASCII, se muestra en la figura 4-18; este código usa 7 bits, que permiten representar hasta 128 caracteres distintos.

En el esquema de la figura 4-18, lo que se hace es hacer coincidir la ubicación del switche (fila, columna), con su representación en ASCII; por ejemplo, los números del 0 al 9 se representan en ASCII por los códigos 30_H al 39_H (0110000 al 0111001), tal como se muestra en la tabla 4-5.

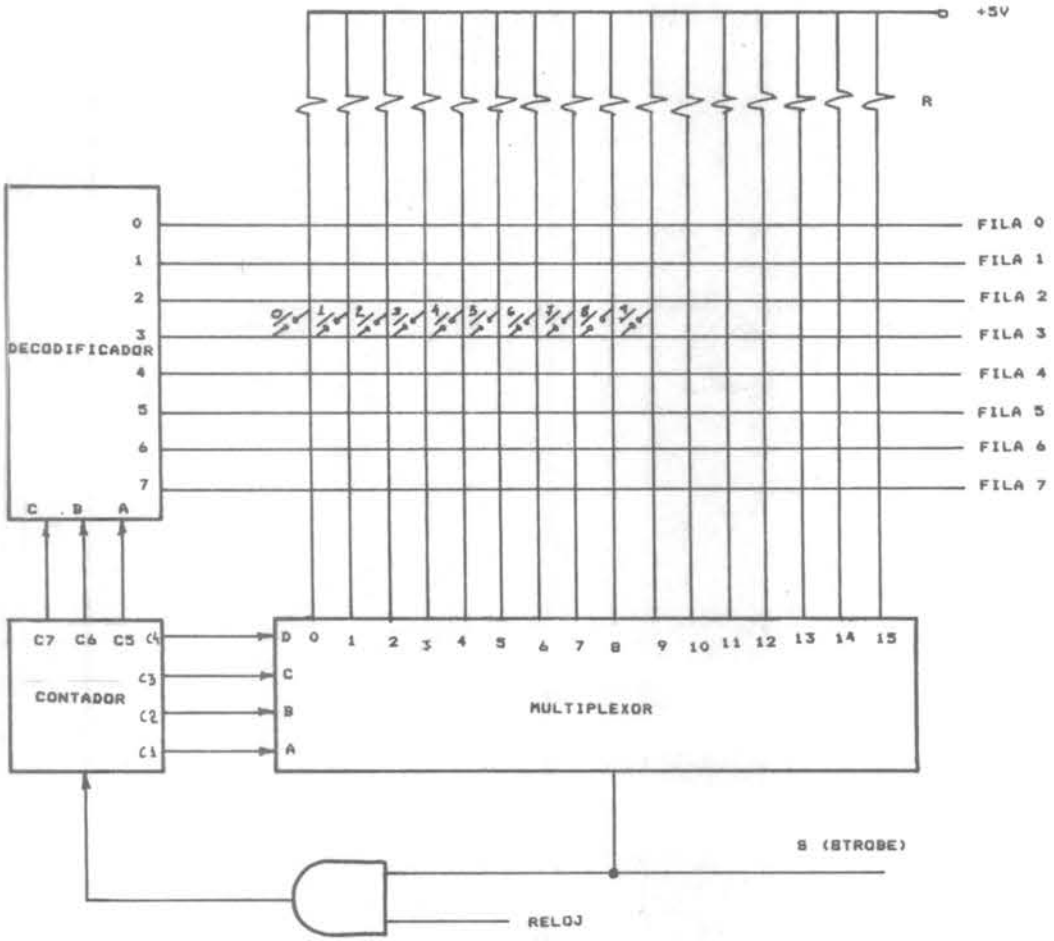


Figura 4-18. Teclado codificado para código ASCII

TABLA 4-5

NUMERO	CODIGO
0	0110000
1	0110001
2	0110010
3	0110011
4	0110100
5	0110101
6	0110110
7	0110111
8	0111000
9	0111001

fila columna

b) Teoría de display

Uno de los dispositivos mas comunmente usados con los microprocesadores es el display de LEDS (diodos emisores de luz) de siete segmentos.

El display de siete segmentos, consiste de 7 diodos emisores de luz, arreglados en la forma como se muestra en la figura 4-19 y generalmente traen incluido un octavo diodo que se usa para mostrar el punto decimal.

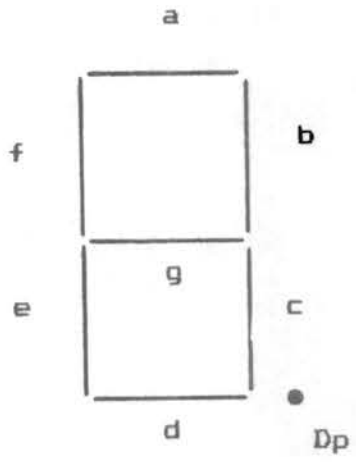


Figura 4-19. Display de siete segmentos

Cuando se activan los segmentos apropiados, se puede obtener casi cualquier letra o número; como son ocho segmentos (8 diodos emisores de luz) se pueden obtener hasta 256 combinaciones que van desde todos los segmentos apagados hasta todos los segmentos prendidos, mostrandose en este último caso el número 8.

Hay dos tipos de display de siete segmentos: el de ánodo comun y el de cátodo común. En el de ánodo común, los ánodos de los 8 diodos que constituyen el display están conectados a un punto común, que se conectará en operación normal a una tensión mayor que la del cátodo, tal como se muestra en la figura 4-20. Un diodo estará polarizado en directo (y emitirá luz en consecuencia), cuando se coloque en el cátodo respectivo una tensión mas baja que la aplicada en el ánodo; generalmente en los circuitos digitales el ánodo se conecta a +5 volt y el cátodo del segmento que se quiere encender se coloca a 0 volt; se requiere de una resistencia externa para limitar la corriente a los niveles tolerables por el diodo.

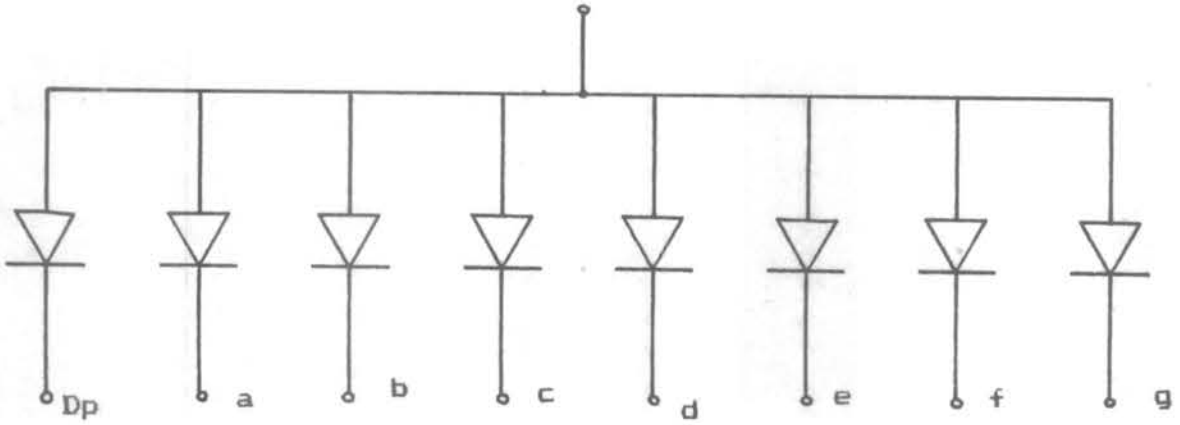


Figura 4-20 Display de 7 segmentos de ánodo común

El tipo de cátodo común, el cual se muestra en la figura 4-21, tiene todos los cátodos de los distintos diodos conectados a un punto común que generalmente se conecta a tierra. En este tipo de LED para encender un diodo se aplicará un nivel alto (+5 volt) en el ánodo del segmento que se desee activar; en este caso tambien se requiere de una resistencia externa para limitar la corriente a través de los diodos.

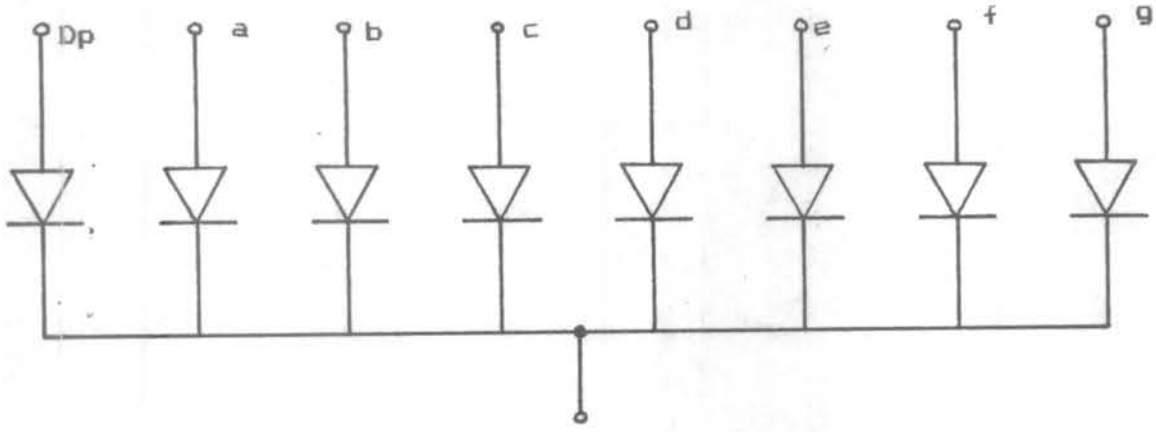


Figura 4-21. Display de siete segmentos de cátodo común

Para manejar el display de siete segmentos, bastará entonces colocar el patrón de unos y ceros que permitan iluminar los LEDS que correspondan al caracter que se desee formar; este patron depende del tipo de LED (cátodo o ánodo común). Por ejemplo si se quiere mostrar las letras A, C, F en un display de

cátodo común, se colocará para la A el patrón 01110111, para la C el patrón 01001110 y para la F el patrón 01000111, tal como se muestra en la figura 4-22.

	Dp	a	b	c	d	e	f	g
A	0	1	1	1	0	1	1	1

	Dp	a	b	c	d	e	f	g
C	0	1	0	0	1	1	1	0

	Dp	a	b	c	d	e	f	g
F	0	1	0	0	0	1	1	1

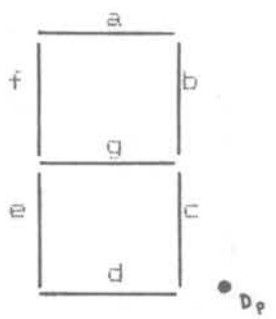


Figura 4-22. Ejemplos de patrones de bits aplicados a LEDS de cátodo común

En general los patrones de bits se aplican a las entradas de cada segmento y se mantienen en las entradas de los mismo mediante un latch o algún registro tal como se muestra en la figura 4-23.

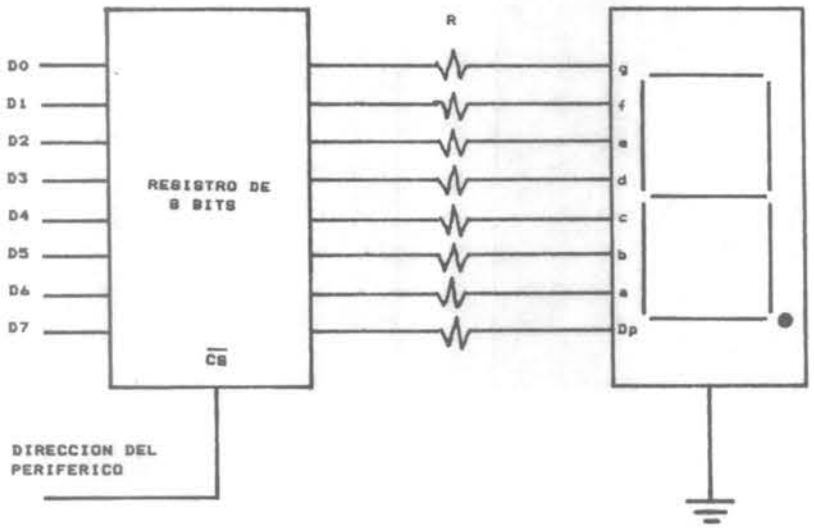


Figura 4-23. Conexión de un display de siete segmentos a un bus

Lo mas común es la presencia de varios de estos display en los sistemas a microprocesadores. En principio cada display requerirá de un registro para almacenar el dato que se está mostrando, si se usa el mismo esquema de la figura 4-23, ademas de una línea para direccionar y habilitar cada display. Así, si son 8 display de siete segmentos, se requerirán 8 señales para direccionamiento y 8 registros; sin embargo, existe otra vía alterna más económica y la cual es la que generalmente se usa, que permite conectar los 8 display de siete segmentos usando solamente un registro o latch para los datos, esto se logra mediante una técnica de multiplexado, tal como se muestra en la figura 4-24.

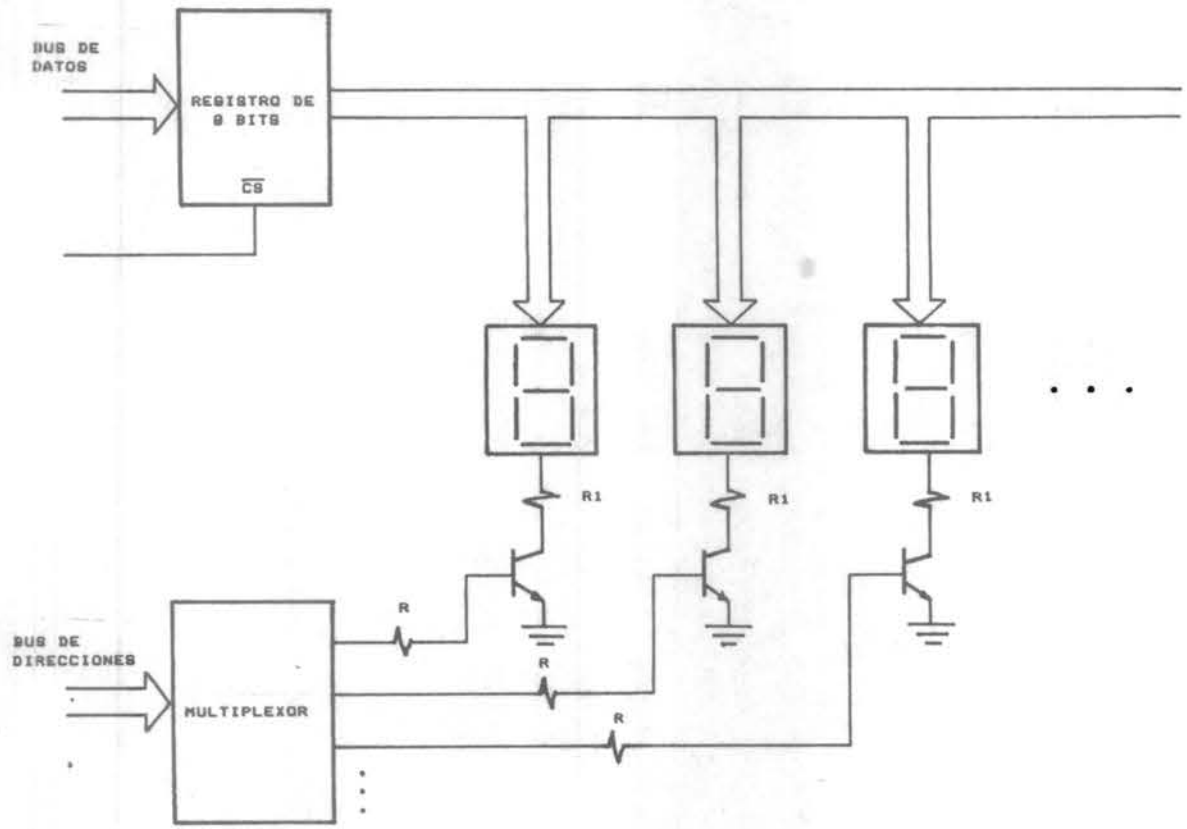


Figura 4-24. Display multiplexado

La idea en el esquema de la figura 4-24 es que en vez de activar todos los 8 display al mismo tiempo, se pueden mostrar en orden rapidamente, uno despues del otro. La rata de muestreo debe ser lo suficientemente rápida para que los segmentos se vean como que si estuviesen permanentemente alimentados; en general, una rata de muestreo mayor de 50 Hz (20 milisegundos) garantiza la eliminación del efecto de "parpadeo", generado por el multiplexado de los display.

4.2.2.2. Descripción funcional del 8279

El 8279, es una interfase programable diseñada para encargarse de la captura de datos desde un teclado o arreglo de sensores y para mostrar datos en un banco de display alfanuméricos o luces indicadoras, manejando todas las actividades relacionadas con estas dos operaciones, en forma autónoma e independiente, liberando al CPU de gran parte de la carga originada por estas funciones, lo que permite aprovechar mas eficientemente el potencial del mismo.

El 8279 fue diseñado para conectarse directamente al bus del microprocesadory puede ser programado por este, a través del bus, para operar en cualquiera de las modalidades disponibles en el mismo. El 8279 puede operar basicamente como dispositivo de entrada de datos desde un teclado o elemento similar o como dispositivo de salida de datos para un display; las diferentes variantes de operación en cada una de estas modalidades se verán a continuación.

MODOS DE ENTRADA

Existen basicamente 3 variantes de operación para la entrada de datos al 8279, ellas son:

Operación para muestreo de teclado no codificado

Operación para muestreo de una matriz de sensores

Operación para entrada controlada desde el teclado

a) Operación para muestreo de teclado no codificado

En esta forma de operación se usan las 8 líneas de entrada RL0 a RL7, en conjunto con las 4 líneas de scan, SL0 a SL3, para determinar el estado de las teclas o switches. Las líneas de scan se pueden programar para que salgan directamente decodificadas, en cuyo caso, de las cuatro líneas, solo una se activará a la vez (activa cuando está en cero), lo que permite muestrear un máximo de 32 switches o teclas (4 filas x 8 columnas); o programarse para que salgan codificadas, en cuyo caso, se podrán muestrear un máximo de 64 teclas o switches (8 filas x 8 columnas). En el primer caso no se requiere de un decodificador externo pues ya las salidas estan decodificadas, en cambio en el segundo caso si se requiere de un decodificador externo.

Cada tecla o switche que se cierre, generará un código

de 6 bits; 3 corresponden a las líneas de scan (SL0 a SL2) y 3 corresponden a la codificación de la columna donde se detectó el cierre del switche; este código de 6 bits, en conjunto con las líneas de entrada SHIFT y CNTL, generan los 8 bits que se almacenarán en el FIFO interno del 8279. El rebote es automáticamente eliminado y el método de "2 KEY LOCKOUT" o "N KEY ROLLOVER", para el manejo del teclado, pueden ser escogidos por software, bajo control de los programas del usuario.

b) Operación para muestreo de una matríz de sensores

En esta forma de operación se usan tambien las 8 líneas de entrada RL0 a RL7, en conjunto con las cuatro líneas de scan para muestrear 32 sensores (4 filas x 8 columnas), si se usa salida decodificada en las líneas de scan o 64 sensores (8 filas x 8 columnas), si se usa salida codificada de las líneas de scan. En este caso las 8 líneas de entrada son directamente almacenadas en la posición de la RAM (FIFO) que corresponda a la fila, y en consecuencia el estado de cada uno de los sensores (abierto o cerrado) de cada fila, queda almacenado en la RAM.

c) Operación para entrada controlada

En esta forma de operación, las líneas de entrada son cargadas en el FIFO, cada vez que se activa la señal CNTL/STB. Esto permite la conexión de un teclado codificado directamente al 8279; la señal de "tecla pisada" (strobe), proveniente del teclado, puede conectarse a la entrada CNTL/STB del 8279. En este caso es posible disponer de un teclado de hasta 256 teclas o caracteres.

MODOS DE SALIDA

El 8279 se puede usar para mostrar datos en display alfanúmérico o en un banco de luces indicadoras. Las líneas de salida para display (8 en total), en conjunto con las líneas de scan, pueden usarse para mostrar de 8 a 16 caracteres. Las salidas de display del 8279, 8 bits en total, se pueden considerar en conjunto como un solo caracter de 8 bits, en cuyo caso el bit menos significativo del bus de datos, DB0, le corresponderá la salida B0 de display y el bit mas significativo del bus de datos, DB7, le corresponderá la salida A3 de display; o considerarlas como dos caracteres de 4 bits cada uno, en cuyo caso, los primeros cuatro bits del bus, DB0-DB3, que constituiyen el primer caracter, le corresponderá en la salida de display los bits B0-B3 respectivamente y el segundo caracter, formado en el bus por los bits DB4-DB7, le corresponderá en la salida de display los bits A0-A3 respectivamente. En el primer caso, la salida de display puede venir directamente en código para manejo de 7 segmentos y en el segundo caso, cada conjunto de 4 bits

puede constituir un caracter BCD que deberá ser convertido a código de 7 segmentos, si se está usando un display de ese tipo. En ambos casos, el formato de alimentación de los caracteres al display se puede seleccionar por software para que sea por la izquierda o por la derecha.

El 8279 se puede considerar dividido funcionalmente en 4 grandes bloques: sección de control, sección bus de datos, sección de teclado y la sección de display. Cada una de estas secciones está compuesta de elementos funcionales mas pequeños, tal como se puede ver en la figura 4-25, que constituye un diagrama de bloque funcional simplificado del 8279. A continuación se presenta un esquema que resume la constitución funcional del 8279 y una descripción breve de cada uno de los elementos que lo constituyen:

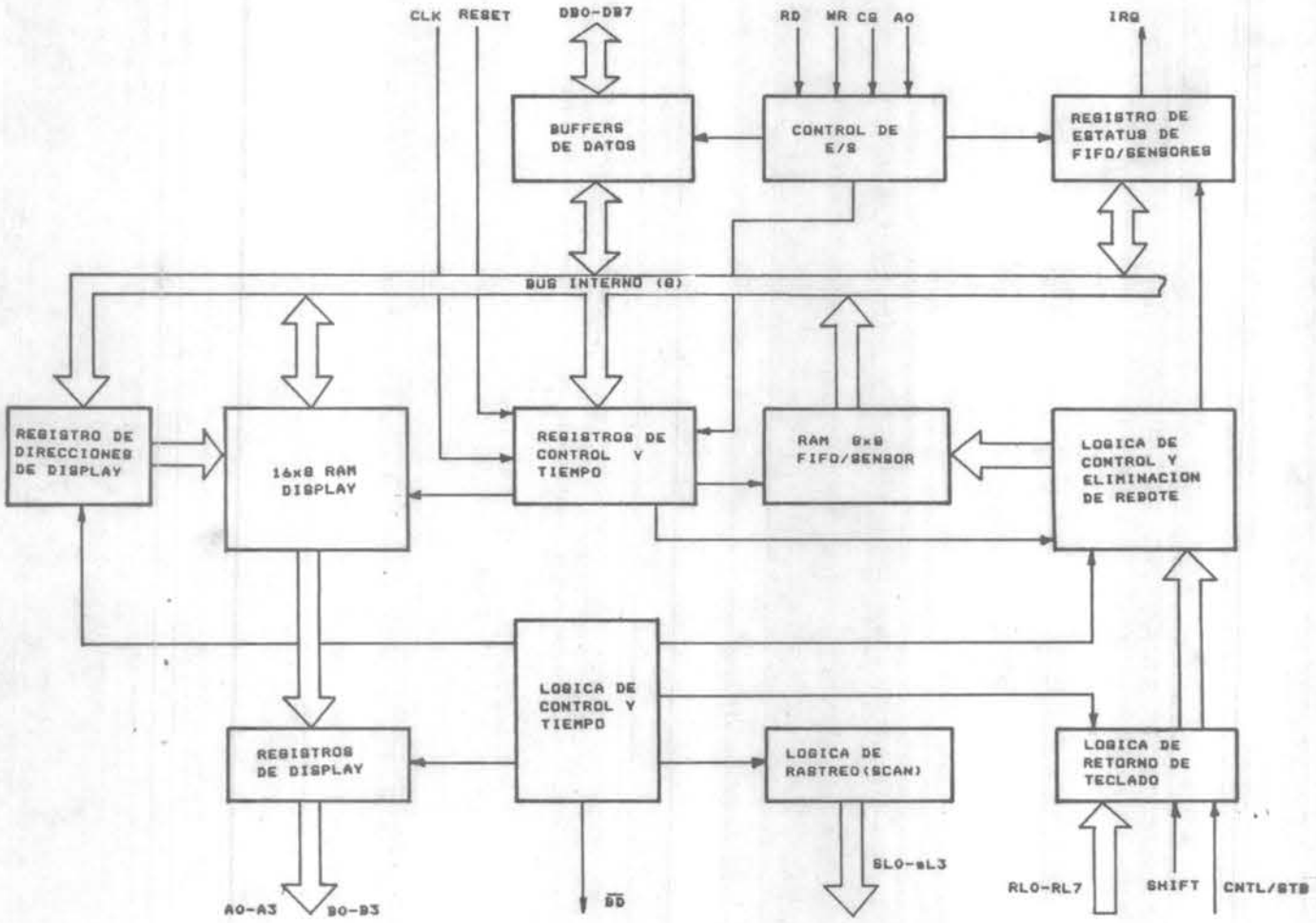


Figura 4-25. diagrama de bloques funcional del 8279

a) Bus de datos

b) Sección de control

Control de E/S
Registros de control y tiempo
Lógica de control y tiempo
Registro de estatus de RAM FIFO/SENSOR
Lógica de scan

c) Seccion de teclado

RAM FIFO/SENSORES
Lógica de control y eliminación de rebote
Registro de retorno de teclado y lógica codificadora

d) Seccion de display

Registro de direcciones de display
RAM de display (16x8)
Registro de display

a) Sección bus de datos

La sección bus de datos está constituida por 8 líneas bidireccionales, DB0-DB7, y sus respectivos buffers de datos, los cuáles permiten al 8279 conectar el bus interno del mismo con su bus externo, el cuál, es el medio por el que el 8279 se conecta a un microprocesador. La dirección del flujo de datos, el tipo de dato presente en el bus y la habilitación o deshabilitación (tri state) del bus, pueden ser controladas por el microprocesador a través de la sección de control del 8279 que se verá a continuación.

b) Sección de control

La sección de control del 8279 se puede considerar dividida en 5 bloques funcionales:

Control de entrada/salida

Registros de control y tiempo

Lógica de control y tiempo

Registro de estatus de RAM (FIFO/SENSOR)

Lógica de scan

El primer bloque funcional de esta sección de control lo constituye la sección de control de entrada/salida; a esta sección entran las líneas $\overline{CS}$, A0, $\overline{RD}$ y $\overline{WR}$, que permiten controlar el estado del bus de datos externo (habilitado o deshabilitado), el tipo de información en el bus (dato o comando de control) y el tipo de operación sobre el 8279 (escritura o lectura de datos o comandos). Ver figura 4-25.

La entrada $\overline{CS}$ permite habilitar o deshabilitar el bus del 8279; cualquier operación que el microprocesador requiera hacer con el 8279, necesitará que esta entrada esté activa ($\overline{CS}$=0). La entrada $\overline{RD}$, en conjunto con la entrada $\overline{CS}$, permiten al microprocesador leer datos o información de control desde el 8279 a través del bus. La entrada $\overline{WR}$, en conjunto con $\overline{CS}$ permiten al microprocesador escribir datos o información de control en el 8279. La naturaleza de la información que el microprocesador va a leer o escribir sobre el 8279, o sea si es un dato o un comando de control, la controla el microprocesador por medio de la señal aplicada en la entrada A0 de esta sección de control de E/S; si A0=0 la información en el bus de datos es un dato y si A0=1, la información en el bus de datos es un comando de control. La tabla siguiente resume las diferentes operaciones de entrada/salida que el microprocesador puede realizar con el 8279 a través del bus.

TABLA 4-6

OPERACIONES DE ENTRADA/SALIDA SOBRE EL 8279

A0	$\overline{RD}$	$\overline{WR}$	$\overline{CS}$	TIPO DE OPERACION
0	0	1	0	Bus de datos del 8279 como salida lectura de un dato desde el 8279
0	1	0	0	Bus de datos del 8279 como entrada escritura de un dato en el 8279
1	0	1	0	Bus de datos del 8279 como salida lectura de información de control
1	1	0	0	Bus de datos del 8279 como entrada escritura de un comando de control
X	X	X	1	Bus de datos del 8279 en tri-state ninguna operación a través del bus

Otro bloque de esta sección de control lo constituyen los registros de control y tiempo y la lógica de control y tiempo. Los registros de control y tiempo se usan en el 8279 para almacenar la información de control proveniente del microprocesador, con la finalidad de programar el modo de operación del 8279 en el manejo del teclado y el display, y en general estos registros se usan para almacenar toda la información que definirá en definitiva el comportamiento funcional del 8279. La información de control se almacena en estos registros, mediante una operación de escritura sobre el bus del 8279, con la entrada A0 activa (A0=1), lo que como se vió anteriormente, le indica al 8279 que la información presente en el bus es un comando de control, que debe ser almacenado en los registros de control y tiempo. Como parte de esta sección de registros de control y tiempo se encuentran una cadena de contadores que permiten, al ser utilizados como divisores de frecuencia programables, controlar la rata de muestreo del teclado y el display, y el tiempo de espera que se usará para eliminar el rebote de los switches del teclado; como primer elemento de esta serie de contadores se tiene un preescalador, el cuál no es mas que un divisor por N, en donde N lo programa el usuario mediante un comando de control, como se verá en la sección programación del 8279. El valor de N puede ser escogido entre 2 y 31 y tipicamente se ajusta de manera tal de obtener una frecuencia interna de 100KHz, lo cuál dará un tiempo de muestreo de teclado de 5.1 milisegundos y un tiempo de eliminación de rebote de 10.3 milisegundos.

Otro elemento de esta sección de control y tiempo lo constituye la lógica de control y tiempo, la cuál se encarga de enviar todas las señales requeridas por los distintos elementos internos del 8279 para que estos funcionen en concordancia con la información almacenada en los registros de control y tiempo; adicionalmente este elemento provee una salida externa denominada BD, que se usa para realizar blanqueos del display, programados por el usuario mediante un comando de control o automáticamente durante el switcheo en un display de un caracter a otro.

Otro bloque de esta sección de control lo constituye la lógica de scan. Como parte de esta lógica se encuentra un contador que puede operar en dos modos: en el modo codificado, en el cuál el contador cuenta cíclicamente en binario entre 0 y 15, apareciendo sus salidas reflejadas en las salidas SL0-SL3 del 8279; estas 4 líneas de salida (SL0-SL3) pueden ser decodificadas externamente para obtener hasta 16 líneas que podrán ser usadas para muestrear 16 filas de un teclado o 16 caracteres de un display, tal como se muestra en la figura 4-26.

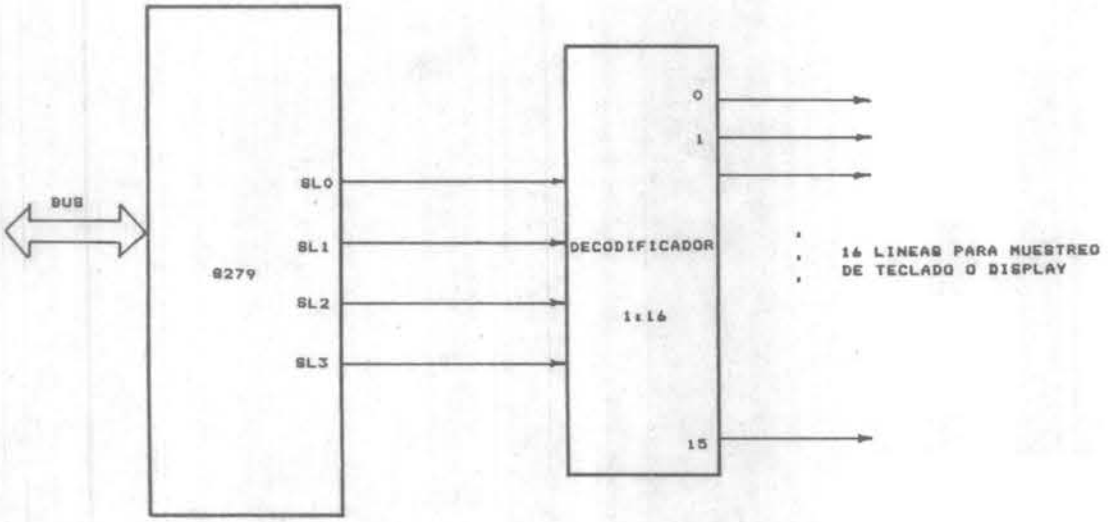


Figura 4-26. Decodificación externa de las salidas de scan del 8279

En el modo decodificado, el contador provee sobre sus líneas de salida (SL0-SL3), un código 1 de 4 que permite muestrear un máximo de 4 filas de un teclado o 4 caracteres en un display; en este caso no se requerirá decodificación externa tal como se muestra en la figura 4-26, en la cuál se puede apreciar que solo una de las líneas está activa (activa cuando es cero) a la vez.

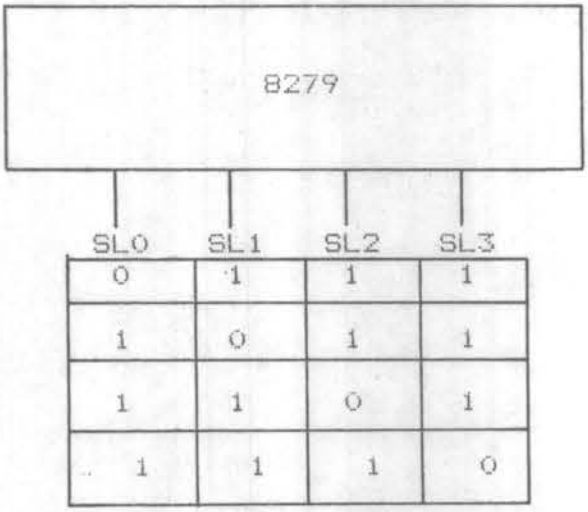


Figura 4-27. Operación con salida decodificada de las líneas de scan

La operación codificada o decodificada de las líneas de scan es programada por el usuario mediante un comando de control, sin embargo, es necesario llamar la atención sobre el hecho de que una vez que el usuario ha decidido usar un determinado modo de operación del contador de scan (codificado o decodificado), el mismo será usado tanto para la sección de teclado como para la sección de display, o sea, no es posible programar por ejemplo, una operación decodificada de las líneas de scan para controlar el teclado y programar al mismo tiempo una operación codificada para la sección de display.

El último bloque de la sección de control lo constituye el registro de estatus de la RAM de teclado. En este registro se almacenan: condiciones de errores ocurridas (overrun y underrun), estatus del FIFO (completamente lleno o con un cierto número de caracteres), número de caracteres almacenados en el FIFO y la disponibilidad del display (disponible o no disponible por estar realizandose una operación de blanqueo). Este bloque es tambien el encargado de proveer la señal de interrupción del 8279 al microprocesador, IRQ, que se activará (IRQ=1), cuando se detecte un cambio en un sensor, si se está operando en el modo muestreo de sensores, o cuando se carga algún caracter en el FIFO, si se está operando en el modo matríz de switches o entrada controlada; en este último caso la señal IRQ se mantiene activa mientras exista algún caracter en el FIFO.

c) Sección teclado

La sección teclado está constituida por una memoria RAM de 8x8, la lógica de eliminación de rebote y el registro de retorno de teclado con su lógica codificadora.

La memoria RAM puede cumplir una doble función, dependiendo del modo de operación de la sección teclado; si se usa el 8279 en el modo teclado o en el modo de entrada controlada, esta memoria se comporta como una memoria FIFO (First In-First Out), de manera que los datos son escritos en la RAM en posiciones sucesivas y luego cada dato es leído en el mismo orden en que fueron escritos. Si el 8279 se usa para muestrear una matríz de sensores, la memoria RAM se usa como RAM normal; cada localización de la RAM (fila) le corresponde una única fila de la matríz de sensores y en dicha localización se carga el estado de cada uno de los sensores conectados en esa fila.

El registro de retorno de teclado y su lógica codificadora tienen por finalidad almacenar la información proveniente de las entradas de teclado, RL0-RL7; este registro cumple varias funciones adicionales a la de simple almacenador del caracter proveniente del teclado, dependiendo del modo que se esté usando para el manejo del teclado.

En el modo de teclado, las líneas de entrada RL0-RL7 son muestreadas para detectar si alguna tecla ha sido pulsada en una determinada fila; al detectarse una condición de tecla pulsada, el circuito antirebote hace que se espere aproximadamente 10 milisegundos (si f=100KHz) y vuelve a chequear si la tecla esta todavía pulsada, en cuyo caso las líneas de scan SL0-SL3 (que constituyen la fila) y la codificación de la columna donde se detectó el cierre del switche (tres líneas internas del 8279 que permiten codificar las 8 líneas RL0-RL7 de entrada de teclado), se almacenarán en el FIFO en conjunto con las entradas SHIFT y CNTL, constituyendo estas últimas los bits mas significativos del dato almacenado en el FIFO.

En el modo de muestreo de una matríz de sensores, el contenido de las líneas de retorno RL0-RL7 se transfiere directamente a la fila correspondiente de la RAM, cada vez que la fila es muestreada, de manera que la RAM es continuamente refrescada con los valores leídos de la matríz de sensores, y cada vez que las líneas de scan seleccionan una fila, el estado de los sensores en esa fila se almacenará en la RAM.

En el modo de lectura controlada, por la entrada CNTL/STB, el contenido de las líneas de retorno, RL0-RL7, se transfiere directamente al FIFO, cada vez que se produzca un pulso en la entrada CNTL/STB; la carga del dato en el FIFO ocurre en forma efectiva en el flanco de subida del pulso en CNTL/STB.

4.2.2.3. Programación del 8279

A continuación se verán las instrucciones que sirven para programar el 8279, para su operación en los distintos modos que se describieron en la sección anterior. Los comandos son enviados por el microprocesador a través del bus de datos, con las entradas $\overline{CS}$=0 y A0=1, y el 8279 los almacena en su registro de control y tiempo en el flanco de subida de la señal $\overline{WR}$, tal como se aprecia en la figura 4-28 que constituye un diagrama de tiempo de una operación de escritura de comando del 8279.

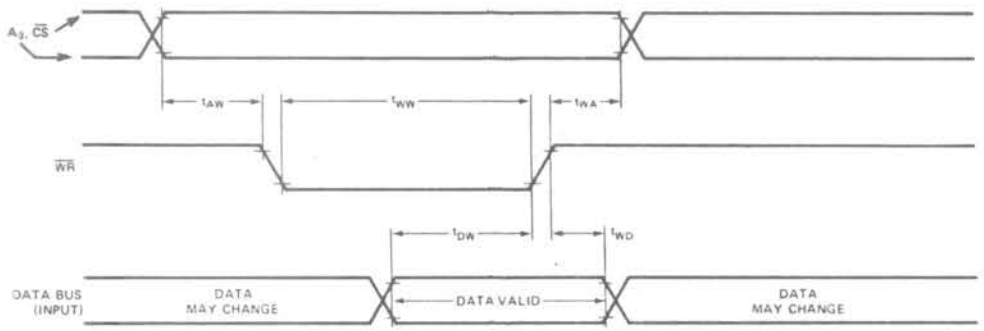


Figura 4-28. Diagrama de tiempo de una operación de programación del 8279

Existen una serie de instrucciones para la programación del 8279 las cuáles se han agrupado según la función que cumplen como se muestra a continuación:

Instrucción para inicialización de teclado y display

Instrucción para programación del reloj del 8279

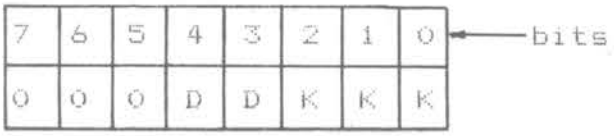
Instrucción para programar el uso de la memoria RAM

Instrucciones para manipulación de la RAM (Read/Write)

Instrucciones para manipulación del display

Instrucción para inicialización del teclado y el display.

El formato de la instrucción de comando para la inicialización del teclado y el display se muestra a continuación:



DD especifíca el modo en que operará el display según, el siguiente esquema:

DD = 00 especifíca un display de 8 caracteres de 8 bits cada uno y formato de entrada de datos por la izquierda del display

DD = 01 especifíca que se manejará un display de 16 caracteres de 8 bits cada uno con entrada de datos por la izquierda

DD = 10 especifíca un display de 8 caracteres de 8 bits cada uno con entrada de datos por la derecha del display

DD = 11 especifíca un display de 16 caracteres de 8 bits cada uno con entrada de datos por la derecha

KKK especifíca el modo en que operará el teclado según el siguiente esquema:

KKK = 000 especifíca operación para matríz de switches (teclado no codificado); con salida codificada de las líneas de scan y método antirrebote "2 Key Lock Out"

KKK = 001 especifíca operación para matríz de switches (teclado no codificado); con salida decodificada de las líneas de scan y método de eliminación de rebote "2 Key Lock Out"

KKK =010 especifíca operación para matríz de switches; con salida codificada de las líneas de scan y método de eliminación de rebote "N Key Rollover"

KKK = 011 especifíca operación para matríz de switches; con salida decodificada de las líneas de scan y método de elimina ción de rebote "N Key Rollover"

KKK = 100 especifíca operación para matríz de sensores con salida codificada de las líneas de scan

KKK = 101 especifíca operación para matríz de sensores con salida decodificada de las líneas de scan

KKK = 110 especifíca operación para teclado codificado (no requiere de líneas de scan) las líneas de scan, aplicables en este caso al display, están codificadas

KKK = 111 especifíca operación para teclado codificado, con líneas de scan para el display en forma decodificada

Instrucción para lectura de la RAM de teclado.

El formato de la instrucción para programar el uso de la memoria RAM del 8279 se muestra a continuación:

7	6	5	4	3	2	1	0	← bits
0	1	0	A1	X	A	A	A	

X = Don't care

A1 es un flag usado para especificar el uso de autoincremento de la dirección al hacer una operación de lectura de la RAM; solo es aplicable en operación para matríz de sensores

AAA especifíca la dirección de la fila que será leída; solo es aplicable en operación para matríz de sensores

Este comando se usa para especificar que la fuente de donde leerá los datos el microprocesador es del RAM del 8279. No se requiere ningún comándo adicional mientras solo se desee leer de este RAM; sin embargo, si se desea leer una fila distinta a la que está seleccionada se requiere de otro comando en donde se especifíque en AAA la fila que se desea leer. Si se especifíca A1=1, el contador selector de fila se incrementará en uno automáticaente despues de cada lectura, de manera que la próxima

lectura será de la próxima fila de la RAM.

Instrucciones para lectura y escritura en la RAM de display.

La RAM de display puede ser accesada para la lectura o para la escritura mediante los comandos de control apropiados; a continuación se verá el formato de los comandos de control que permiten leer y escribir la RAM de display del 8279:

Instrucción para lectura de la RAM de display.

7	6	5	4	3	2	1	0	←bits
0	1	1	A1	A	A	A	A	

A1 Es un flag que se usa para especificar el auto incremento; A1=1 autoincremento A1=0 sin auto incremento

AAAA especifíca la dirección desde donde se leerá el caracter. En vista de que el registro de direcciones de la RAM de display es el mismo tanto para lectura y escritura, al usar el autoincremento en la lectura tambien afectará la escrira

Instrucción para escritura en la RAM de display.

7	6	5	4	3	2	1	0	←bits
1	0	0	A1	A	A	A	A	

A1 es el flag para especificar el autoincremento

AAAA especifíca la dirección de la RAM de display donde se escribirá el dato

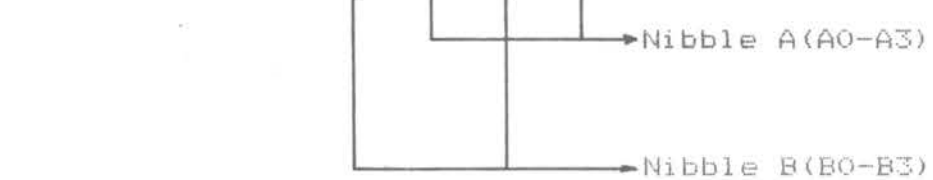
El direccionamiento y el autoincremento en esta instrucción son idénticos que en la instrucción de lectura, la única diferencia estriba en que el comando de lectura no afecta

la fuente desde donde se extraera el dato y el de [illegible]

Instrucción para la escritura Parcial en la RAM de display.

Esta instrucción es útil cuando el manejo del display se haya programado para operar como dos caracteres de 4 bits cada uno (un caracter lo constituye las salidas A0-A3 y el otro caracter las salidas B0-B3 de display); en algunos casos se hace necesario enmascarar una de las dos mitades de 4 bits, de manera que los datos que entren al 8279 provenientes del microprocesador, no afecten la otra mitad (el microprocesador siempre escribirá 8 bits a través del bus) del display. El formato de la instrucción es el siguiente:

7	6	5	4	3	2	1	0
1	0	1	X	IW	IW	BL	BL



Los bits IW (de los nibbles A y B), permiten al programador inhibir la escritura en el RAM de display del nibble que se desee; IW=1 enmascara el nibble o caracter, de manera que se podrá escribir en la RAM de display sin afectar el nibble con IW=1; si IW=0 la escritura de un caracter en la RAM de display afecta el nibble con IW=0.

Los bits BL de ambos nibbles permiten limpiar el display, en forma separada si se desea, de cada uno de los caracteres (nibble A y nibble B). La limpieza del display mediante este comando coloca todos los bits de display en cero. Cuando se haya programado operación con caracteres de 8 bits (nibbles A y B en conjunto), para borrar el display se requerirá que los bits BL de ambos nibbles sean colocados en 1, para que el display sea borrado.

Instrucción para limpieza del display y el registro de estatus del FIFO.

Existe otro comando que permite limpiar el display, limpiar el registro de estatus del FIFO, limpiar todas las posiciones de la RAM de display con un código determinado; el formato de esta instrucción se da a continuación:

7	6	5	4	3	2	1	0	←bits
1	1	0	CD	CD	CD	CF	CA	

Los bits CD se usan para la limpieza del display según el siguiente esquema:

4	3	2 ←bits	
CD	CD	CD	
0	X		llena la RAM de display con ceros A0-A3 = 0000 y B0-B3 = 0000
1	0		llena la RAM de display con 20_H A0-A3 = 0010 y B0-B3 = 0000
1	1		llena la RAM de display con unos A0-A3 = 1111 y B0-B3 = 1111

1 permite habilitar la limpieza del display (esto tambien se puede hacer con el bit CA)

CF permite inicializar el registro de estatus del FIFO, colocándolo en el estado vacio y da reset a la línea de interrupción del 8279. Despues de la ejecución de esta instrucción si CF=1 el apuntador de direcciones de la RAM para matríz de sensores apuntará a la fila cero

CA tiene el efecto combinado de CD y CF. Cuando CA se activa (CA=1), se usa el código definido en CD para borrar el display, da reset a la línea de interrupción del 8279, inicializa el flag de estatus del FIFO y resincroniza la cadena de divisores interna o base de tiempo del 8279

Cuando se realiza una operación de limpieza de display se requiere hacer un rastreo completo de todo el display; durante el tiempo en que el 8279 realize la limpieza del display, el microprocesador no puede escribir en el RAM de display; para indicar esta condición al microprocesador, el 8279 coloca en 1 el bit mas significativo de la palabra de estatus del FIFO (indicación de RAM de display no disponible), mientras esté realizando la limpieza del display.

Instrucción para dar reset a la línea de interrupción y para activar el modo ERROR.

Existe un comando específico para cuando se programa la operación del teclado en el modo matríz de sensores, se pueda dar un reset a la línea de interrupción, IRQ, con la finalidad de permitir futuros pedidos de interrupción; cada vez que se detecta un cambio en un sensor el 8279 genera una interrupción (IRQ=1) permaneciendo la señal IRQ activa hasta tanto sea borrada; mientras la señal IRQ esta activa, se inhibe la generación de otras interrupciones cuando se detecte algún cambio en uno de los sensores y se impide la escritura en la RAM de teclado. A continuación el formato de esta instrucción de control:

7	6	5	4	3	2	1	0	←bits
1	1	1	E	X	X	X	X	

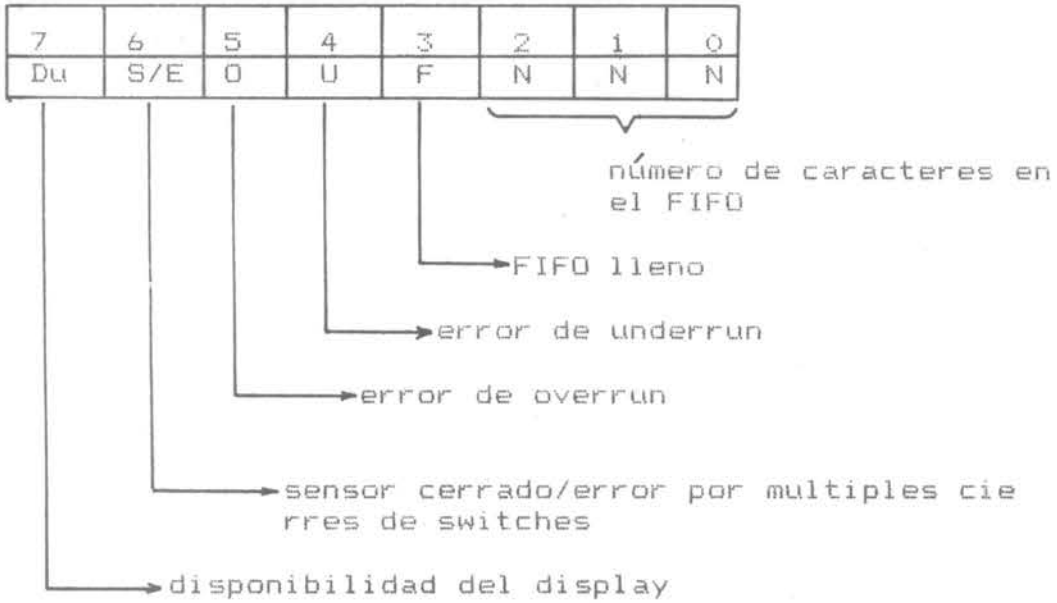
X= Don't care

Adicionalmente este comando permite programar al 8279 para que opere en el modo de detección de error, cuando se haya seleccionado el método de antirrebote "N Key Rollover"; en este modo especial de operación del 8279, si se presionan dos o mas teclas simultaneamente, se encenderá el bit de error en el registro de estatus del FIFO lo que genera una interrupción que deshabilita futuras escrituras en el FIFO. La activación del modo de detección de error con "N Key Rollover", se logra, si en esta instrucción de comando se coloca E=1.

Instrucción para lectura de la palabra de estatus del FIFO.

La palabra de estatus del 8279 puede ser leída a través del bus mediante una instrucción especial. La palabra de estatus

contiene campos para indicar el estado del FIFO, la ocurrencia de errores (overrun y underrun), disponibilidad del display y el número de caracteres almacenados en un momento determinado en el FIFO. El formato de la palabra de estatus se muestra a continuación:



Para la lectura de la palabra de estatus se colocan A0=1 y $\overline{CS}$=$\overline{RD}$=0.

Instrucción para programación del reloj

Anteriormente se mencionó que el 8279 tiene un circuito de reloj interno que puede ser programado para hacer coincidir los tiempos de muestreo y antirrebote con los tiempos de un ciclo del microprocesadpor.

El formato de la instrucción para programar el reloj se muestra a continuación.

7	6	5	4	3	2	1	0	← bits
0	0	1	P	P	P	P	P	

En esta instrucción PPPPP, es el valor del pre-escalador, el cuál puede ir de 2 a 31. El preescalador programable, divide el reloj externo por PPPPP, para obtener la frecuencia básica interna del 8279; al escoger un valor de PPPPP que produzca una frecuencia básica de 100KHz, se asegura que el 8279 trabaje con los tiempos de muestreo y de antirrebote

especificados. Por defecto despues de un RESET el valor de PPPPP es de 31(11111).

4.2.2.4. Descripción de los pines del 8279

El 8279 es encapsulado en un chip de 40 pines, tal como se muestra en la figura4-29; a continuación una breve descripción de la función de cada pin en el 8279.

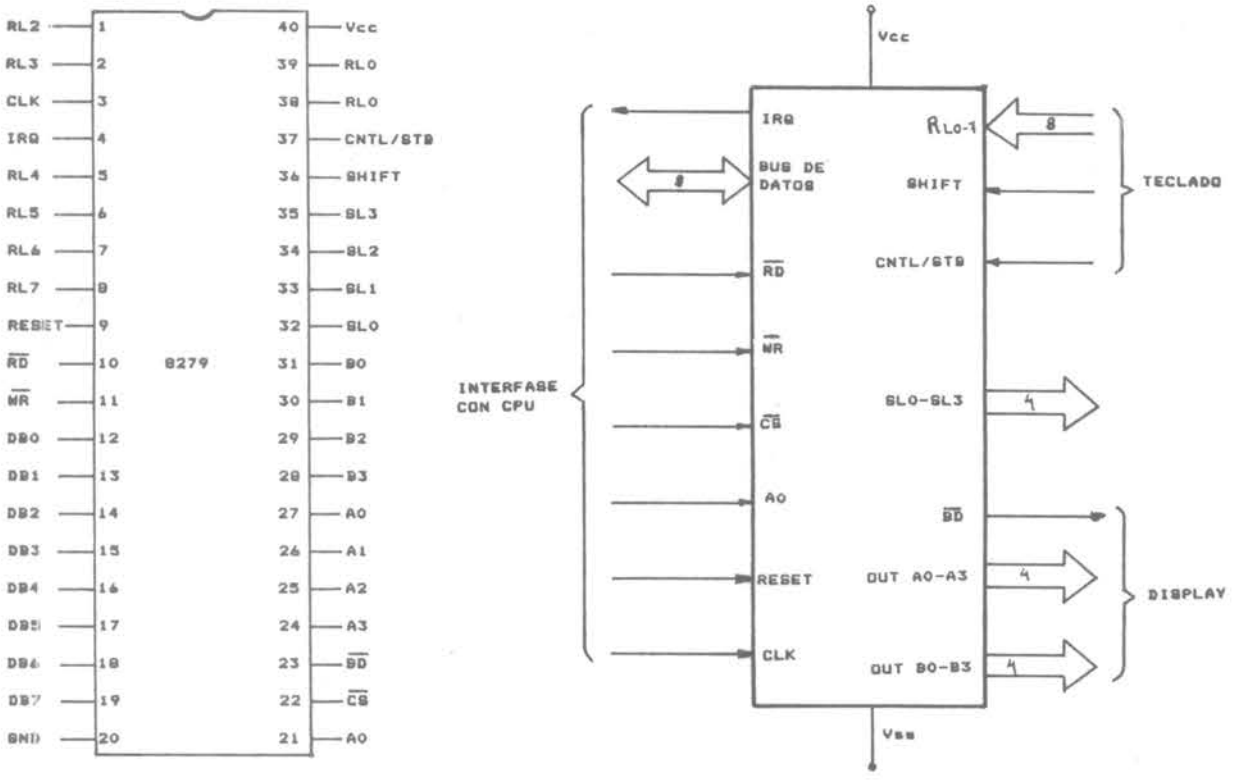


Figura 4-29. Configuración de los pines del 8279

PIN	DESCRIPCION
12-19	DB0-DB8: bus de datos bidireccional. todos los datos y comandos desde el microprocesador son transferidos a través de estas 8 líneas
31-28	OUT B0-OUT B3: puerto de salida de 4 bits para manejo de el display. Puede usarse en conjunto con el otro puerto de 4 bits para el display, para constituir un puerto de 8 bits
27-24	OUT A0-OUT A3: puerto de salida de 4 bits para manejo del display. Puede usarse en conjunto con el otro puerto del display para formar un puerto de 8 bits

38,39, 1,2, 5-8 — RL0-RL7: líneas de retorno de entrada. Estas líneas están conectadas a las líneas de rastreo a través de los sensores o switches. Internamente poseen unos resistores de "pullup", los cuales las mantienen en un nivel alto hasta que al cerrar un switche o sensor, la línea de muestreo lo lleve a cero

32-35 — SL0-SL3: estas son las líneas de rastreo o scan son usadas para rastrear los switches de un teclado o una matríz de sensores y los digitos de el display. Estas 4 líneas pueden salir directamente decodificadas (1 de 4) o codificadas (1 de 16)

3 — CLK: entrada de reloj al 8279; se usa para generar las señales de tiempo internas del 8279

9 — RESET: es la entrada de reset del 8279; se usa para inicializar el 8279. activa con el nivel 1

22 — $\overline{CS}$: selección de chip; un cero sobre este pin habilita las funciones de la interfase ya sea para recibir o transmitir información

21 — A0: dirección del buffer; A0=1 indica que las señales que están saliendo o entrando al 8279 son comandos o información de estatus; A0=0 indica que la información que está entrando o saliendo del 8279, son datos

10,11 — $\overline{RD}$, $\overline{WR}$: señales de control usadas para indicar si la operación sobre el 8279 es de lectura o escritura

4 — IRQ: requerimiento de interrupión; en el modo teclado esta línea va a 1 cuando hay algún dato en el FIFO; en el modo matríz de sensores, esta línea se hace 1 cada vez que se detecta un cambio en un sensor

36 — shift: el estado de esta línea se almacena junto con la posición de la tecla pisada en el modo teclado. Ella tiene internamente una resistencia "pullup" que la mantiene en estado alto hasta que se cierre el switche de shift, lo cuál la pone en el estado bajo

37 — CNTL/STB (control/strobe): en el modo teclado esta línea es usada como una entrada de control y se almacena como estatus de la tecla pisada, constituyendo el bit 8 de la palabra correspondiente en la RAM. En el modo de entrada controlada (teclado codificado), esta línea se usa para habilitar la carga de datos en el FIFO del

teclado. Esta línea tiene una resistencia interna de "pullup" que la mantiene alta hasta que se cierre el switche que la pone en bajo.

4.4. OTRAS CARACTERISTICAS DEL SISTEMA IMSAI

SUBRUTINAS UTILES DEL MONITOR

Varias subrutinas del monitor pueden ser útiles al usuario. Algunas de ellas se describen mas adelante. Se describirá el procedimiento para llamarlas, ya que se encuentran en el banco cero de memoria. Este banco de memoria debe ser seleccionado antes de ejecutar una llamada a estas subrutinas. For ejemplo si un programa de usuario residente en la dirección 800H desea llamar a la rutina de entrada de teletipo (TELEIN), la cual reside en la localidad 366H, la secuencia de llamada será la que sigue:

Loc	Instr	Mnemónico
800	E5	SEL MB0
801	55	STRT T
802	74	CALL TELEIN
803	66	
804	F5	SEL MB1

Todas las subrutinas requieren que el banco de registros cero sea seleccionado y regresan con este mismo banco seleccionado. Muchas de las subrutinas usan y destruyen uno o mas registros de los bancos de registros 0 y 1, durante su operación. Estos son indicados en la descripción que sigue. Para mayor información se recomienda leer el listado del monitor el cual está en el apendice B.

SUBRUTINA SYNCST: (200H) Genera continuamente bytes de sincronización (E6H) al puerto del cassette. no retorna al programa. Se debe hacer un reset para regresar al monitor.

SUBRUTINA CTR8279: (208H) Saca el caracter que está en el registro A al puerto de control del chip 8279. Usa los registros R1 de RB1, P2 y A.

SUBRUTINA CLEAR:(213H) Limpia el display y empieza en

la posición cero. Registros usados: P2 y R1 de RB1.

SUBRUTINA OUTDSP: (21FH) Saca el registro A en la próxima posición del display. Registros usados: P2, R1 de RB1.

SUBRUTINA KEYINP: (234H) Lee del teclado los valores de 0 - 17H. Regresa el valor en el registro A. Registros usados: R1 de RB1.

SUBRUTINA DISPRG: (268H) Muestra el registro A en hexadecimal en los próximos dos caracteres del display. Registros usados: R7, R1 de RB1, A y Carry.

SUBRUTINA DISPAD: (26FH) Muestra los 12 bits de dirección que están en R0 y R5 (R0 =bits 0-7, R5=bits 8-11, los bits 4-7 de R5 no se usan) sobre el display en las posiciones 2, 3 y 4. Registros usados: A, R1 de RB1 y R7.

SUBRUTINA BANKER: (2AEH) Usa los bits 0-3 de R5 para seleccionar el banco de memoria externa colocando estos bits correspondientes a la página en el puerto 2 en los bits 0-3.

SUBRUTINA DISPKA: (2C9H) Muestra 12 bits de direccionamiento que están en R2 y R3 en las posiciones 5, 7 y 8 de el display. R2 = bits 0-7. Registros usados: R3 bits 0-3 corresponden a los bits 8 - 11 de la dirección, A, R1 de RB1 y R7.

SUBRUTINA CASIN:(300H) Lee un byte desde cassette y lo almacena en el registro A. Requiere que el timer sea arrancado antes de llamarla (START T). Registros usados: R2, R3 y R6.

SUBRUTINA CASOUT: (31FH) Saca un byte desde el registro A hacia cassette. Registros usados: R2, R6, Timer, A y F1.

SUBRUTINA TELEOUT: (34BH) Saca el registro A por el puerto serial. Registros usados: R2, A, Acarreo, R6 y Timer.

SUBRUTINA TELEIN: (369H) Lee un byte desde el puerto serial y lo almacena en el registro A. Registros usados: R2, acarreo y R6.

PROGRAMA DE AUTODIAGNOSTICO

En la localidad 1D3H esta grabada una rutina de chequeo la cual examina toda la memoria presente, el teclado y el display. Se puede ejecutar esta rutina con el comando

<EXEC> <1D3> <,> <.>

Esta rutina tiene dos modos: test de la memoria y test del teclado y el display. En el modo de test del teclado y display se entra al comienzo de la rutina. En este modo cualquier tecla pisada aparecerá en todo el display. Esto indicará algún segmento malo del display y/o fallas en el contacto del teclado.

Para entrar en el segundo modo basta presionar la tecla del cero. Se examinará a partir de la dirección FFDH hacia abajo a través de toda la memoria, se escribirán todos los patrones de bits en cada celda, si hay alguna diferencia entre el patrón escrito y el patrón leído, el programa se para y muestra el siguiente display.

A A A ↘ W W I I ↘

Donde, AAA es la dirección del byte incorrecto, WW es el patrón que se escribió y II es el patrón que se leyó.

El algorítmo para recorrer la memoria es tal que si solo 1K está presente en el tope, y está bueno, aparecerá en el display

B 0 0 ↙ 00 ff ↙

Si los 2K están instalados, aparecerá en el display lo siguiente para indicar que está en buen estado.

7 0 0 ↙ 0 0 F F 0 0 ↙

Cualquier otro display indicará que la memoria está mala o que falta memoria en la posición de 1K en el tope.

Tan pronto como termina el display de memoria, empieza el test de los relees. Este test cierra y abre los cuatro relees basado en lo que se introduzca por el teclado. Los cuatro bits menos significativos de la tecla pisada son sacados por los

relees. El ruido de los relees indicarán que están funcionando. Una salida cero cierra el relee y una salida 1 lo abre. Para cerrar cada relee las siguientes teclas deben ser pisadas

<E> <C> <8> <0>

Cuatro ruidos "click", separados, se deberán oir a medida que cada relee se cierra. Ahora al presionar las siguientes teclas se abrirá un relee en cada momento produciendo de nuevo cuatro "click" audibles.

<1> <3> <7> <F>

4.5. MANUAL DEL USUARIO

EJECUCION DE PROGRAMAS

Una vez que el programa ha sido cargado en la memoria de programa a través del teclado o usando otro medio externo, el usuario puede ejecutar el programa de dos modos distintos: "debug" o "standalone". En el modo standalone el programa del usuario puede manejar todos los recursos del sistema tales como memoria externa e interna, teclado, display, interrupciones, etc.

El modo debug permite al usuario localizar errores de programación. En este modo hay varias restricciones en el uso de los recursos del sistema, ya que el monitor necesita algunos de estos recursos para funcionar apropiadamente. Vea la sección sobre restricciones (125) para mayor información. La función EXEC/BREAK puede ser usada de cuatro formas:

I. Modo standalone

a. <EXEC> <,> <.> - Ejecuta el programa apartir de la localidad 800 16

b. <EXEC> <direccion> <,> <.> - Ejecuta el programa a partir de la dirección especificada.

II. Modo debug usando "breakpoints"

a. <BREAK> <dirección1> <,> <dirección2 > <.>

Empieza la ejecución en la dirección1 y continua hasta

alcanzar la dirección2.

b. <BREAK> <,> <dirección2> <.>

Continua a partir del último breakpoint (o a partir de la localidad 800 si no había ningún breakpoint) hasta alcanzar la dirección2.

La función de brakpoint la lleva a cabo el monitor sustituyendo las direcciones especificadas como dirección2 y dirección2 + 1 por la instrucción EN I (05H habilita el sistema de interrupciones externas), el monitor salva el contenido de estas direcciones en las localidades FE1 y FE2 ubicadas en el RAM externo. Cuando se llega al breakpoint en la ejecución, ocurre una interrupción y la rutina manejadora de interrupciones del monitor realiza todos los arreglos necesarios para salvar el estatus del programa y el valor de los registros internos. Despues de cada breakpoint el display mostrará:

El breakpoint no debe ser colocado en donde esten las siguientes instrucciones: RET (83), RETR (93), JMPP @A (B3), o en cualquier instrucción simple que preceda una instrucción CALL o BRANCH. No hay restricciones en colocar el breakpoint en instrucciones de dos bytes, siempre y cuando se coloque en el primer byte de la instrucción.

FUNCION RESET

Al presionar la tecla de reset se ejecuta la secuencia de encendido con la cual resulta que el display es limpiado, el banco de registros cero es seleccionado, las interrupciones son deshabilitadas y se eliminan los breakpoints. El usuario deberá proceder a almacenarlos manualmente antes de usarlos.

FUNCION CLEAR ENTRY

Cualquier función que se haya escogido erroneamente puede ser eliminada antes de presionar la tecla final de ENTER basta presionar la tecla CLEAR ENTRY. Esta función limpiará el display y ademas mostrará el caracter de listo (" = ").

USO DE LOS RELEES

El IMSAI contiene 5 relees, cuatro estan bajo control del CPU a través de los programas del usuario y el quinto relee no está conectado a ninguna línea de entrada o salida, pero puede ser facilmente conectado a cualquiera de las líneas de entrada y salida sin uso, disponibles en los conectores J3 o J4.

Los relees RELY0 hasta RELY3 están conectados al puerto 5 del 8243, bits 0, 1, 2, 3 respectivamente. Para cerrar un relee en particular se debe enviar un cero al bit respectivo del puerto 5. Para abrir un relee se debe escribir un uno en el bit respectivo del puerto 5.

C A P I T U L O V

5.1. INTRODUCCION

Hay dos cosas que se pueden hacer con los microprocesadores; programarlos y conectarlos a otros circuitos. Hasta el momento se ha enseñado como programar el 8035 y como el está conectado en el IMSAI al 8279 y al 8243. En este capítulo se enseñará como conectarlo con: memoria, periféricos lentos, periféricos rápidos, y finalmente como conectarlo al mundo analógico que lo rodea a través del uso de conversores A/D y D/A.

5.2. CONEXION CON MEMORIA

Es bastante importante la conexión del microprocesador con memoria ya que en ella guardará programas, direcciones, resultados intermedios y datos, de allí que uno de los puntos de evaluación de un microprocesador es cuanta memoria puede direccionar. Esto viene dado por el número de bits que el microprocesador dispone para dirección. Estos bits son los que constituyen el bus de direcciones. Asi si un microprocesador posee un bus de direcciones de 12 bits puede direccionar hasta 2^{12} posiciones de memoria o lo que es lo mismo 4K bytes (4096 bytes). Si tiene un bus de direcciones de 16 bits podrá direccionar 64K bytes (2^{16} = 65536 bytes). Como el microprocesador viene con esta capacidad incluida en su construcción, el dispone de mecanismos directos para accesar cualquier información que esté en esta memoria. Estos mecanismos son los que se llaman modos de direccionamiento y la capacidad de memoria que puede direccionar directamente a través de los modos de direccionamiento se ha venido llamando espacio de memoria. Asi el primer microprocesador tiene un espacio de memoria de 4k palabras y el segundo tiene un espacio de memoria de 64k. La forma en que se distribuye el espacio de memoria, es lo que se conoce como "mapa de la memoria".

Unos de los problemas que se presentan cuando se desea conectar un chip de memoria a un microprocesador son: la

diferencia entre el número de bits del bus de direcciones y el número de bits para dirección que posee el chip de memoria; otro problema es cuando se desea direccionar una cantidad de memoria mayor que el espacio de memoria que posee el microprocesador. Tenemos así las siguientes combinaciones:

espacio de memoria > capacidad de memoria del chip

espacio de memoria = capacidad de memoria del chip

espacio de memoria < capacidad de memoria deseada

El caso mas sencillo es cuando el espacio de memoria es igual a la capacidad de memoria del chip. La conexión será limpia sin ningún circuito intermedio; cada línea del bus de direcciones del microprocesador irá a un pin de dirección del chip de memoria y las líneas de control para lectura o escritura irán conectadas a las correspondientes en la memoria. Ahora bien las otras dos positilidades, se resuelven decodificando las direcciones del microprocesador. La decodificación puede ser total o parcial.

Asi, si un microprocesador posee 12 bits para direccionar y se desean conectar chips de memoria de 8 bits de direcciones, o sea con una capacidad de almacenamiento máxima de 256 bytes o bits, dependiendo del chip, para llenar completamente el espacio de memoria, se deberán conectar 16 de estos chips de memoria, ya que con 12 bits se pueden direccionar 4096 bytes, y puesto que cada chip solo posee 256 bytes, se requerirán 16 de estos chips. A continuación se verá una posible forma de conectar estos 16 chip de memoria usando decodificación total, con decodificadores y con lógica discreta.

5.2.1. DECODIFICACION TOTAL

Una conexión con decodificación total para este arreglo se puede lograr con un decodificador o con lógica discreta:

a) Decodificación total usando un decodificador

La figura 5-1, muestra un circuito que permite decodificación total con un circuito decodificador. Los 4 bits mas significativos de la dirección van al decodificador 1 de 16, el cual habilitará una de las 16 RAM al colocar un cero en su entrada CS y un uno en las entradas CS de las otras 15 memorias restantes, como se muestra en la siguiente figura.

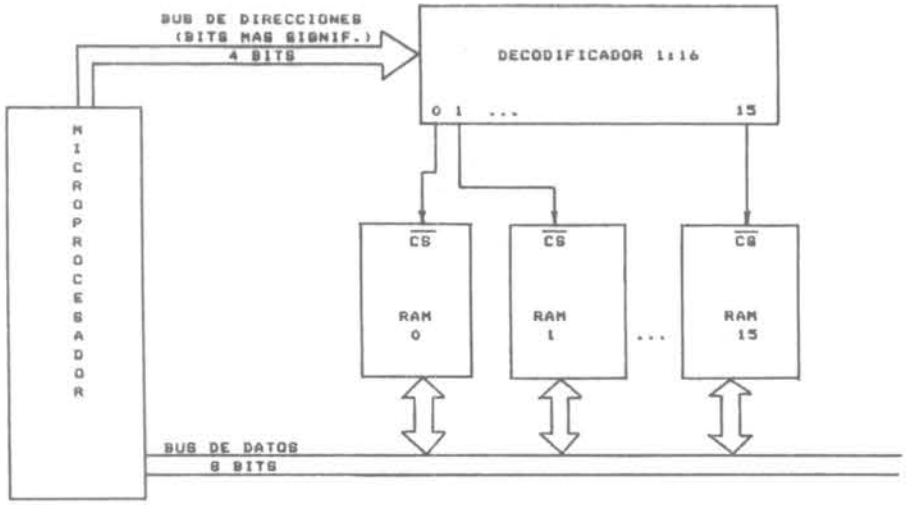


Figura 5-1. Decodificador total de la dirección usando decodificadores.

b) Decodificación total con lógica discreta.

Para seleccionar por ejemplo el chip 0, se conectará en su entrada $\overline{CS}$ el circuito que se muestra en la figura 5-2, el cual activará la entrada $\overline{CS}$ de la memoria 0, cuando se cumpla la condición A11=A10=A9=A8=0000

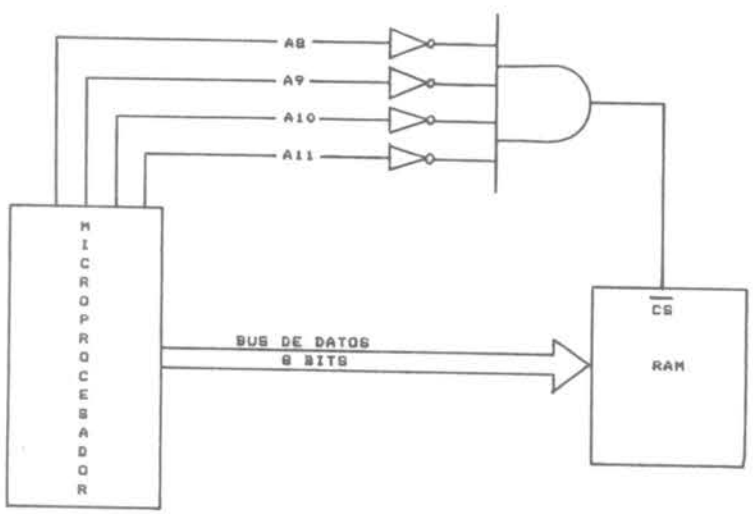


Figura 5-2. Decodificación total de la dirección usando lógica discreta.

Como se puede observar en ambas conexiones se usaron los 4 bits más significativos para seleccionar el chip de memoria y los 8 bits menos significativos para direccionar 1 de las 256 localidades del chip de memoria. Notese que los 12 bits ayudaron a determinar una dirección específica, lo cual nos indica que la memoria esta totalmente decodificada.

5.2.2. DECODIFICACION PARCIAL

Ahora se estudiarán las conexiones del IMSAI para ilustrar la decodificación parcial. La figura 5-3 muestra la distribución del espacio de memoria en el sistema IMSAI.

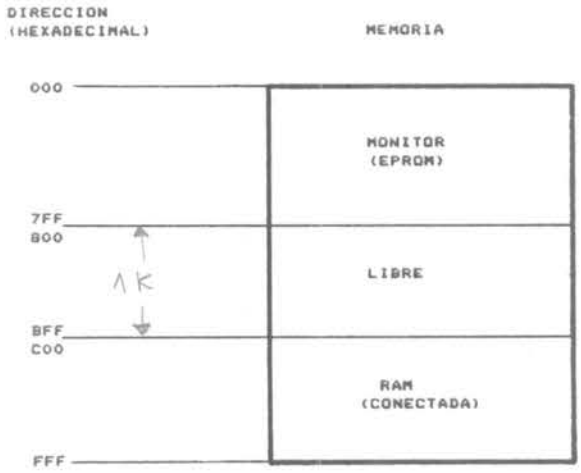


Figura 5-3. Espacio de memoria del IMSAI

Como se puede observar se tiene desde la dirección 800_H hasta BFF_H disponible ya que en ella no se ha conectado memoria, supongamos que se desea conectar una RAM de 16 palabras, 4 bits de direcciones a partir de la dirección 800h. Si se usara codificación total se tendría que detectar el siguiente patrón de bits.

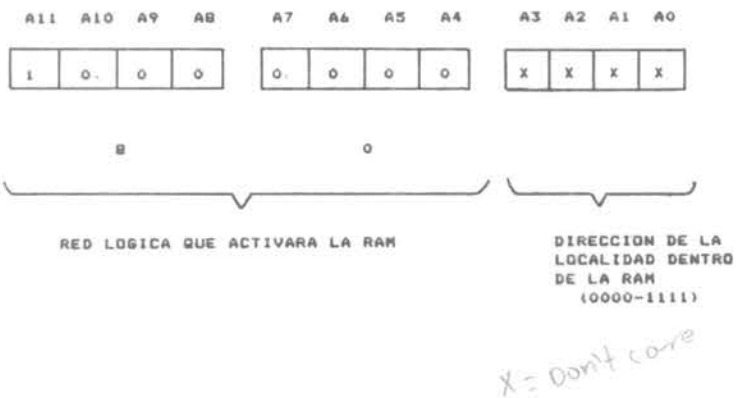


Figura 5-4. Patrón de bits para decodificación total

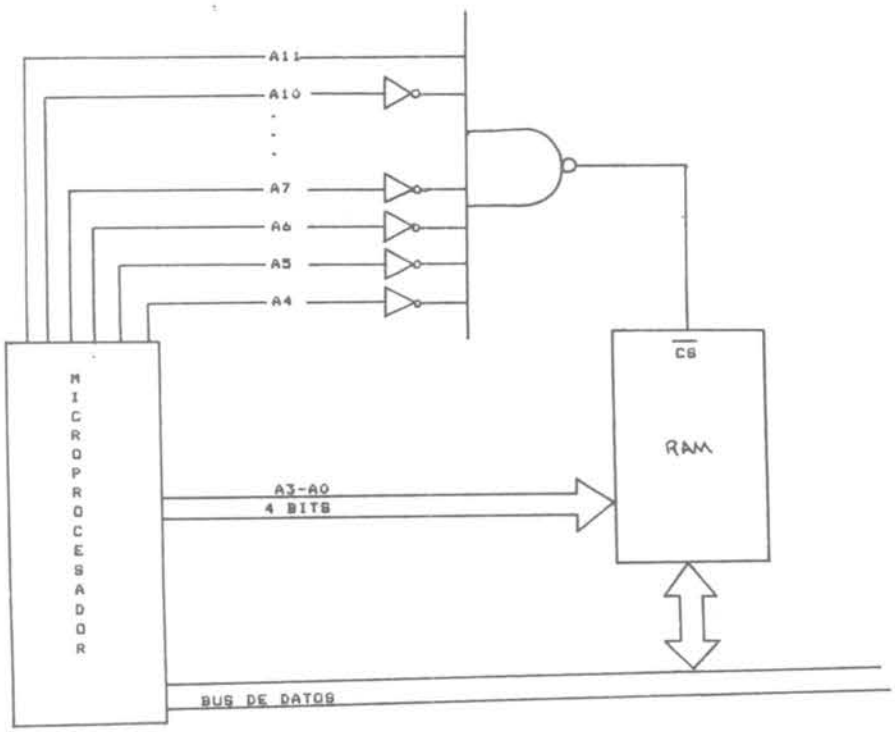


Figura 5-5. Red de decodificación total

Si se decodifica parcialmente esta dirección el patrón de bits que hay que chequear se reduce a A11=1 y A10=0.

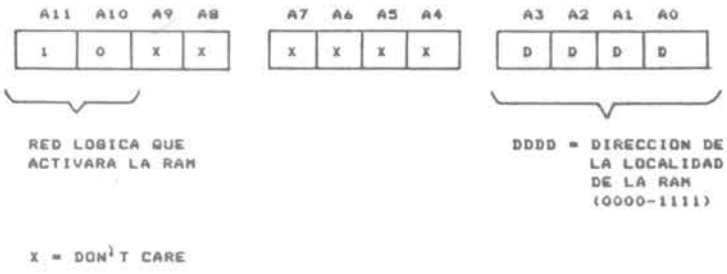


Figura 5-6. Patrones de bits a chequear y direcciones decodificadas.

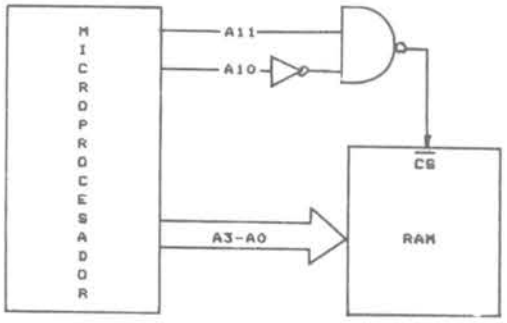


Figura 5-7. Red de decodificación parcial

Como se puede observar del mapa de memoria, desde la posición 800 hasta la BFF no hay nada conectado y como:

8=1000
9=1001
A=1010
B=1011

A11, A10=10 direcciona esta area. Con esta decodificación parcial se estan accesando una gran cantidad de direcciones como son 80H - BFH en los 8 bits mas significativos, sin embargo, esto no debe preocupar, pues en esa zona del espacio de memoria no hay nada conectado. En el patron de bits se indican los bits A9 - A4 con "X", para indicar que son condición don't care, ya que no han sido conectados. La decodificación parcial trae como ventaja un ahorro en el número de compuertas.

5.2.3. BANCO DE MEMORIA

El último caso que faltaría por tratar es el correspondiente a cuando se desea accesar una memoria mayor que el espacio de memoria disponible en el microprocesador. Esto se resuelve con el simple hecho de usar una o mas líneas del microprocesador adicionales al bus de direcciones. Asi si se tiene un microprocesador con 4k de capacidad de direccionamiento (bus de direcciones de 12 bits) se puede incrementar a tantos k como se deseen, sacando por ejemplo, los otros bits de direccionamiento por un puerto.

En el caso del microprocesador 8035, el cual tiene un bus de direcciones de 12 bits, se puede sacar por uno de los puertos, 4 bits extras de dirección, con lo cual se incrementará su capacidad de 4k a 64k.

La tecnica de usar líneas adicionales a las del bus de direcciones, a través de un puerto, se conoce como "banco de memoria". En el caso del ejemplo que se acaba de mencionar, se tendran 16 bancos de memoria de 4k cada uno. Para cambiarnos de banco solo será necesario sacar por el puerto escogido el número del banco en que se desea trabajar (0H - FH). En este ejemplo, se implementó el banco de memoria con decodificación total, pero se podrian usar los criterios mencionados para decodificación parcial, si no se está usando todo el espacio de memoria, que ahora sería de 64k.

Finalmente cabe recordar que en el espacio correspondiente a memoria tambien se pueden conectar periféricos de entrada/salida, constituyendo lo que se llama "memory mapped I/O" (entrada/salida conectada en el espacio de memoria).

Obviamente dichos periféricos pueden ser direccionados con las mismas técnicas de decodificación que se han mencionado para memoria.

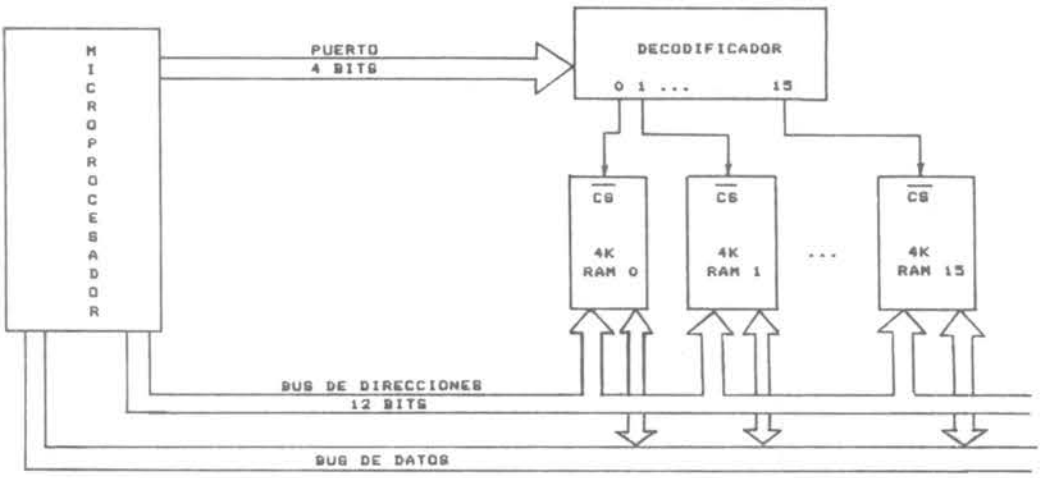


Figura 5-8. Banqueo de memoria con decodificación total.

5.2.4. TIEMPO PARA LEER Y ESCRIBIR DE MEMORIA

En la sección 2.12 se estudió la operación de lectura de memoria y su diagrama de tiempo, tambien se definió tiempo de acceso a memoria como el tiempo transcurrido entre el momento en que se le envía a la memoria el comando de "lea", previamente establecida la dirección de la localidad desde donde se va a leer, y el momento en que la memoria coloca el contenido de dicha localidad en su salida.

Escribir en memoria requiere un cuidadoso estudio del tiempo. Normalmente la dirección y las líneas de datos deben ser estables mientras se envíe el comando de "escriba" como se muestra en la Figura 5-9. Normalmente las memorias tienen un pin para recibir este comando, dicho pin viene, por ejemplo, con la indicación R/W, por lo cual, si se desea leer (R/W=1) se aplica un nivel alto en este pin y de igual forma si se desea escribir se colocará en este pin un nivel bajo (R/W=0).

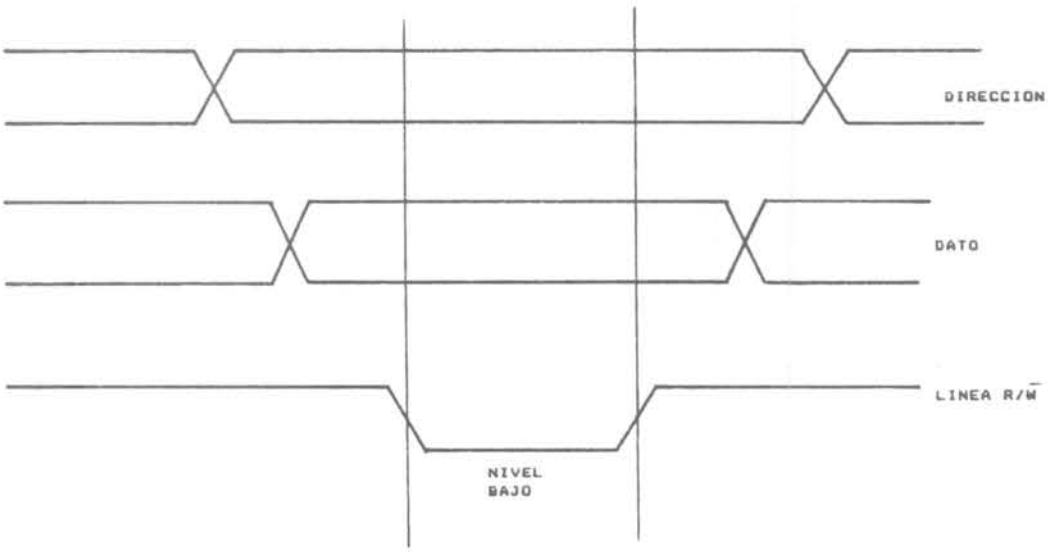


Figura 5-9. Diagrama de tiempo correcto de una operación de escritura en memoria.

Realmente el dato es almacenado apenas la línea R/W alcanza el nivel cero. Si los datos cambian durante el tiempo en que la línea R/W esta en cero, la memoria almacenará los nuevos datos. De la misma forma, si la dirección cambia cuando la línea R/W=0, el mismo dato será almacenado en la nueva dirección.

La Figura 5-10. muestra un ejemplo donde se exagera una mala distribución del tiempo.

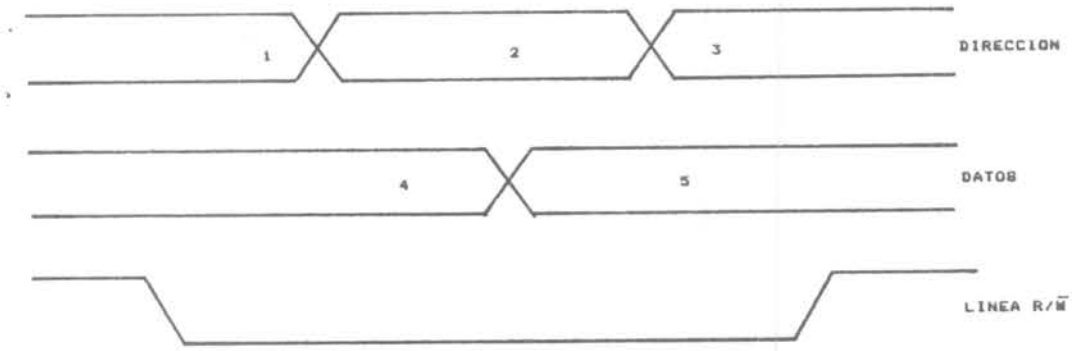


Figura 5-10. Diagrama de distribución de tiempo erronea en operación de escritura en memoria.

En las condiciones de la dirección (1) el dato (4) será almacenado. Despues el dato 4 será almacenado en la dirección 2. Sin embargo mientras está la dirección 2, los datos cambian a la condición 5. Por lo tanto el dato 5 es ahora almacenado en la dirección 2. Finalmente el dato 5 es almacenado en la dirección 3. De aquí se puede entender la importancia que tiene una distribución de tiempo correcta de la información presente en los buses de dirección y datos y su sincronización con la señal de control R/W.

5.3. CONEXION CON PERIFERICOS LENTOS.

La conexión del microprocesador con periféricos lentos trae el compromiso de usar eficientemente el microprocesador a la velocidad que el es capaz de trabajar y sin embargo, comunicarse con periféricos con una velocidad de 2 y 3 ordenes de magnitud menor que la velocidad del microprocesador. Hasta el momento se ha estudiado todas las señales de control y el hardware que poseen los microprocesadores para comunicarse con el mundo exterior a través de los perifericos. Ahora se verá desde el punto de vista de programación como se lleva a cabo ese cometido y como se resuelve el problema de la diferencia de velocidades.

Desde el punto de vista de la programación de la E/S, la misma se puede realizar de tres modos diferentes, estos son:

entrada/salida incondicional.

entrada/salida condicional.

entrada/salida con interrupcion de programa.

5.3.1. TRANSFERENCIA INCONDICIONAL

La transferencia incondicional es la tecnica mas simple y directa. Se hacen varias suposiciones bastantes críticas, la primera suposición es que el dispositivo está listo para transmitir o recibir datos. La segunda suposición es que la información de estatus y control no es necesaria. y la tercera suposición es que los datos por si mismos están estables y son válidos en algún registro o localidad de memoria; la secuencia típica para la transferencia de datos se muestra en la figura 5-11. Solo se emplea una instrucción, puede ser OUT, IN.

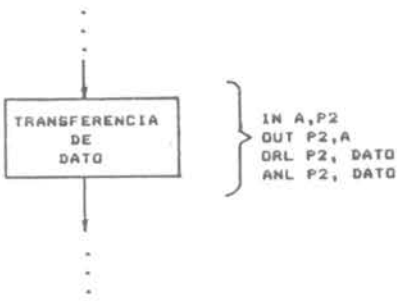


Figura 5-11. Transferencia incondicional.

5.3.2. TRANFERENCIA CONDICIONAL.

Esta tranferencia es la que se explicó en la sección 2.15 como transferencia de datos programada. En esta transferencia se hace necesario averiguar el estatus del periférico antes de ejecutar una transferencia. Aquí la tranferencia consta de dos pasos, ver figura 5-12, primero se deberá chequear el estatus del periférico para determinar si está listo o no; si lo está, se realiza la transferencia, si no, se debe volver a chequear, manteniendose en lo que se llama lazo de espera, hasta que el periférico esté listo.

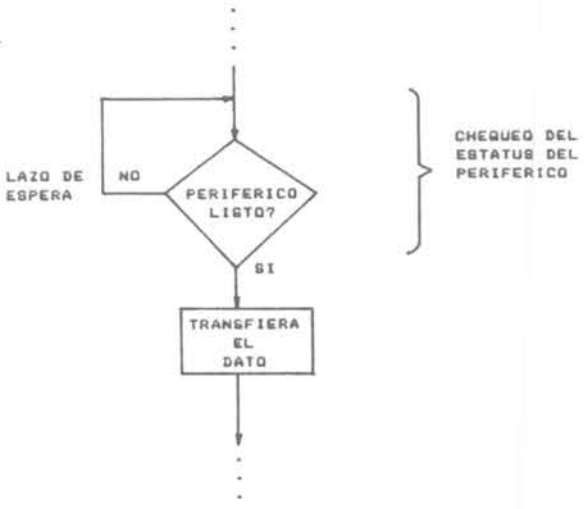


Figura 5-12. Transferencia condicional.

El mismo procedimiento se requiere para leer datos desde el teclado a través del 8279. En efecto muchos periféricos suponen que se hará una transferencia de datos condicional. En el ejemplo siguiente se muestra como el 8035 realiza una transferencia condicional, leyendo la palabra de estatus del FIFO del 8279, para detectar si se ha leído alguna tecla.

EJEMPLO

```
        MOV A, 0F     Saca la página (0F) por el puerto 2
        OUT P2,A

        MOV R1, FE    Pone en R1 la dirección del registro de
                      datos del 8279

        MOV R0, FF    Pone en R0 la dirección del registro de
                      control del 8279

LEA     MOVX A, R0    Lee registro de estatus del FIFO

        AND A,@07     Chequea el stack pointer del FIFO; si
        JZ LEA        esta vacío vuelve a LEA; si no sigue

        MOVX A,@R1    Lee el dato y lo almacena en A
                .
                .
                .
```

5.3.3. TRANSFERENCIA MEDIANTE INTERRUPCION DE PROGRAMA.

Las dos técnicas vistas anteriormente poseen algunos inconvenientes; la transferencia incondicional supone que el periférico está listo para el momento de la transferencia de datos. La transferencia condicional requiere que el microprocesador espere por el dispositivo, permaneciendo en el lazo de espera hasta que el periférico esté listo, inhibiendo así cualquier otra tarea del microprocesador; mientras mas lento sea el periférico mas tiempo pasará el microprocesador en el lazo de espera llegando a ser este retardo inaceptable. En este caso se puede emplear la técnica de entrada/salida con interrupciones. Ella permite la transferencia de datos con periféricos, sin tener que estar ocioso el microprocesador en un lazo de espera. La secuencia aparece en la figura 5-13; el programa principal se ejecuta sin ninguna huella de que hubo interrupción, excepto por un pequeño retardo de tiempo. Para cumplir esta actividad de forma invisible o transparente, el programa de interrupción debe salvar el contenido de cualquier registro del microprocesador que vaya a usar; al final del programa de interrupción, el contenido

viejo de los registros debe ser restaurado. Esta preservación y restauración es hecha automáticamente por hardware en algunos microprocesadores. En el 8035 se vio en la sección 2.2.3 y 2.2.4 que se almacenan automáticamente en el stack, el contador de programa y los 4 bits mas significativos de la PSW.

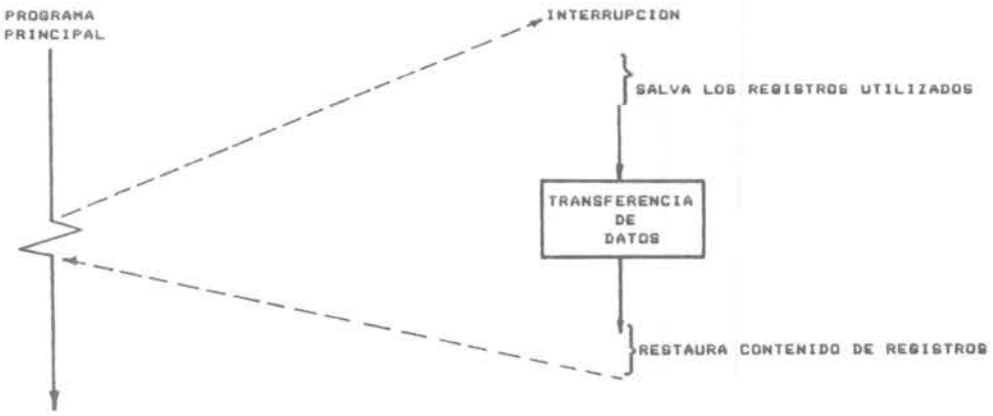


Figura 5-13. Transferencia con interrupción.

Para reforzar el comportamiento trasparente del programa de interrupciones, el programa principal y las subrutinas, deben ser escritas en código reentrante; esto es, cualquier módulo o subrutina deberá permitir ser interrumpido (saltar a una rutina de servicio de interrupción y regresar), sin destruir el contenido de ningún registro en uso del microprocesador, ni de algún dato que esté en el stack o en alguna localidad de memoria; El código reentrante permite hacer esto. El stack es vital para guardar los registros importantes, y por ende para la programación reentrante, pues en el se salvaran el contenido del PC y los registros de condición del programa interrumpido los cuales serán restaurados al finalizar la rutina de servicio de la interrupción, quedando así inalterado la función básica del programa principal, a pesar de la interrupción.

Otras técnicas de programación recomendados para codificar programas con transparencia de interrupción son:

Uso de código reentrante

Salvar solo los registros del microprocesador que se van a usar en la subrutina de servicio de la interrupción.

Usar el stack en forma limitada para el almacenamiento de datos, de una forma consistente y solo cuando sea necesario.

En el microprocesador 8035 se estudió que se inicia una interrupción cuando la entrada INT se hace igual a cero (INT=0). La entrada de interrupción es chequeada por el microprocesador en cada ciclo de máquina, durante la señal ALE y cuando se detecta que dicha entrada está activada, se produce un salto incondicional a la localidad 3 de memoria de programa, tan pronto se complete la instrucción que se esté ejecutando. La localidad 3 tiene usualmente una instrucción de salto incondicional a una subrutina de servicio de interrupción almacenada en cualquier parte de la memoria. El final de la subrutina de servicio esta señalado por la instrucción RETR. El sistema de interrupción del 8035 es de un solo nivel, es decir, una vez que una interrupción se detecta, todas las interrupciones posteriores se ignoran hasta que se ejecute la instrucción RETR (retorne y restaure estatus), la cual habilita de nuevo el sistema de interrupción. Esto último que se acaba de mencionar tambien es válido para las interrupciones internas producidas por el overflow del timer: si ocurren "simultaneamente" una interrupción del timer y una interrupción externa, se reconocerá primero la interrupción externa.

La entrada de interrupción puede ser habilitada o deshabilitada bajo control de programa, usando las instrucciones EN I y DIS I respectivamente. Las interrupciones son tambien deshabilitadas por el RESET y permanecen así hasta que son habilitadas por el programa del usuario. La solicitud de interrupción se debe remover antes de que en la subrutina de servicio de la interrupción se ejecute la instrucción RETR, de lo contrario el procesador volverá a atender la interrupción pues considerará que ha ocurrido una nueva interrupción. Muchos periféricos preveen esto limpiando su línea de pedido de interrupción cada vez que el procesador accesa (lee o escribe) en el registro de datos del periférico; en el caso de que dicho periférico no requiera ser accesado por el microprocesador a raíz de una interrupción, se puede usar una de las líneas de salida del 8035 para borrar el pedido de interrupción del periférico; esta línea de borrar el pedido de interrupción será activada por la subrutina de servicio de la interrupción. La entrada $\overline{\text{INT}}$ puede tambien ser interrogada mediante la instrucción JNI; esta instrucción puede ser usada para detectar la presencia de interrupciones pendientes, antes de que las interrupciones sean habilitadas; si las interrupciones se dejan deshabilitadas, $\overline{\text{INT}}$ se puede usar como otra entrada de manera similar a T0 y T1.

A continuación se presentará un ejemplo que permitirá ilustrar la implementación de una subrutina de servicio de un periférico conectado en el puerto 1 del 8035, desde donde el microprocesador leera datos cuando el periférico lo interrumpa, mediante la activación de la línea INT. El periférico a su vez espera que el microprocesador le envíe una señal de borre la interrupción, para desactivar el pedido de interrupción, tal como se muestra en la figura 5-14.

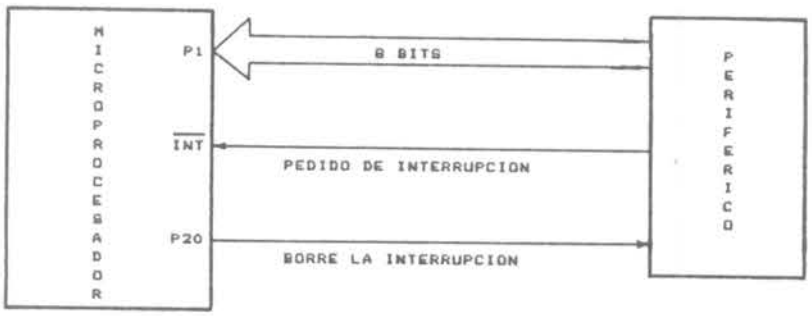


Figura 5-14. Conexión de un periférico mediante el sistema de interrupción

La subrutina de servicio deberá encargarse de salvar los registros en uso en el programa principal que se interrumpe. En vista de que el microprocesador 8035 posee dos bancos de registros (0 y 1), se puede hacer la programación de manera tal que el programa principal trabaje con el banco de registros 0 y la subrutina de servicio de la interrupción trabaje con el banco de registros 1, con la finalidad de minimizar la cantidad de registros que es necesario resguardar para que no se alteren al ejecutarse la subrutina de servicio de la interrupción; con estos propósitos la primera instrucción de la subrutina de servicio será la selección del banco de registros 1 como su banco de trabajo; es de hacer notar que antes de ejecutar la instrucción de retorno de subrutina se deberá seleccionar nuevamente el banco de registros 0 como banco de trabajo de manera que al retornar al programa principal se tenga de nuevo el banco de registros 0 como el banco de trabajo. Tambien es importante tener presente el hecho de que si bien el estatus del programa, representado por el valor del PC y los flag de condición son automáticamente salvados por el hardware del microprocesador, cualquier otro registro o localidad de RAM que sea usada por el programa principal y que eventualmente pueda ser necesaria usarla en la ejecución de la subrutina de servicio de la interrupción, no será salvada automáticamente por el hardware del microprocesador y por lo tanto deberá ser resguardada, bien sea en memoria o en un registro que no se vaya a usar, antes de que se comienze la ejecución propiamente dicha de la subrutina de servicio; el resguardo de los registros e información necesaria para garantizar la continuidad normal del programa interrumpido, es tambien función de la subrutina de servicio y de la misma manera la reposición de estos registros y datos en las localidades donde se encontraban para el momento de la interrupción debe ser llevada a cabo en la subrutina de servicio. En el caso de una rutina de servicio para la atención de la interrupcción del periférico conectado en el puerto 1 (ver figura 5-14), la misma se puede diseñar de manera de que lea el dato proveniente del

periférico, y lo guarde en el registro R2 del banco de registros 1; pero, como el dato leído pasa a través del acumulador, el contenido del registro acumulador, se debe salvar previamente en algún registro no usado, por ejemplo el registro R3 del banco de registros 1, y reponerlo antes de terminar la rutina de servicio de la interrupción. La rutina de servicio se puede poner a trabajar con el banco de registros 1, y el programa interrumpido con el banco de regisros 0. A continuación la codificación de la subrutina de servicio para la atención de la interrupción del esquema de la figura 5-14.

```
SEL RB1     selecciona el banco de registros 1
MOV R3,A    salva el acumulador en R3
IN A,P1     lee dato del periferico
MOV R2,A    almacena el dato leido en R2
ORL P2, 01  se borra el pedido de interrupción
MOV A,R3    restaura el valor del acumulador
SEL RB0     retorna al banco de registros 0
RETR        retorna al programa principal reponien
            do el estatus y habilitando de nuevo
            el sistema de interrupción
```

5.4. CONEXION CON PERIFERICOS RAPIDOS

Periféricos rápidos como discos, floppy disk, cintas magnéticas etc., requieren de atención inmediata y continua. En estos casos es deseable transferir la información entre el periférico y la memoria del microprocesador a una velocidad suficientemente alta y en forma sincronizada, ademas no se desea que el programa o programas que el microprocesador esté ejecutando, interfieran o sean interferidos por la atención al periférico rápido. Es dificil conseguir una atención adecuada del periférico por parte del microprocesador, mediante un programa de atención al periférico, sin que se degraden las otras actividades que el microprocesador deba realizar, en virtud de que por la alta velocidad del periférico, el microprocesador deberá pasar gran parte de su tiempo atendiendo dicho periférico.

La técnica de acceso directo a memoria, DMA, es una de las formas mas usadas para proveer una respuesta inmediata y continua a los periféricos rápidos, sin afectar grandemente la actividad del procesador. Mediante la técnica de acceso directo a memoria, DMA, es posible acoplar directamente la memoria RAM

del microcomputador con el dispositivo periférico, sin tener que usar ninguno de los registros del microprocesador y sin que este intervenga activamente en la transferencia de los datos. El acceso directo a memoria requiere de un hardware extra, el controlador de DMA, el cual puede venir en un circuito integrado, como es el caso del DMA controller de Motorola (6844), o construido en lógica discreta en una tarjeta de interfase; en cualquier caso, el controlador de DMA se conectará al bus de dirección y a los buses de datos y control del microprocesador, para constituir el canal de DMA.

Controlador de DMA

La figura 5-15 es un diagrama de bloque de un microcomputador con un canal de DMA; en la misma se puede apreciar que el controlador de DMA tiene un registro de direcciones (semejante al PC del microprocesador), en donde guarda la dirección de memoria a partir de la cual se va a comenzar a leer o escribir en memoria; tiene tambien un registro contador de palabras, en el cual se almacenan inicialmente el número de palabras que se van a transferir entre memoria y el periférico, y el que se decrementa en uno, cada vez que se transfiere una palabra, hasta llegar a cero, de manera que dicho registro contiene siempre el número de palabras que faltan por transferir; posee el controlador de DMA un registro de datos, en el cual se coloca temporalmente el dato a transferir.

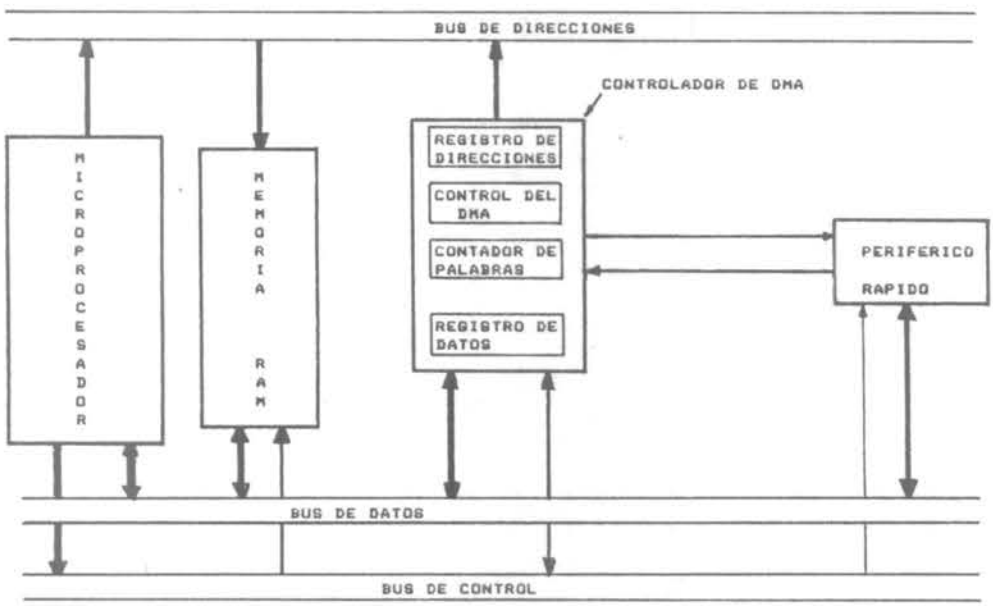


Figura 5-15. Diagrama de bloque de un microcomputador con un canal de DMA

Forma de operación del DMA

La operación del DMA comprende tres etapas:

Etapa de inicialización o de arranque

Etapa de transferencia de la información

Etapa completadora o de finalización de la transferencia

Etapa inicializadora

En el momento que el programador decide realizar una transferencia de datos vía el canal de DMA del microcomputador, mediante instrucciones que generalmente forman parte de una subrutina, se inicializa el canal de DMA. La inicialización del canal de DMA consiste en programar al controlador de DMA, para que transfiera el número de palabras requeridos, la dirección desde la cual leerá o escribirá los datos en la memoria y el sentido de la transferencia (desde memoria al periférico o del periférico a la memoria); mediante los comandos de control apropiados al controlador de DMA que se esté usando, se escribirá en el registro de direcciones del controlador, la dirección de la memoria a partir de la cual se transferirán los datos y en el registro de datos, se escribirá el número de palabras involucrados en la transferencia y por último, la subrutina o el programa inicializador de la transferencia de DMA, envía el comando de control que corresponda, para que el controlador de DMA inicie la operación.

Etapa de transferencia

La etapa de transferencia en la operación del DMA es llevada a cabo en forma automática por el hardware del controlador, sin intervención activa ni del procesador ni de ninguna subrutina desarrollada por el programador (ver figura 5-16); existen varios métodos, los cuales se describirán posteriormente, que permiten al hardware hacer la transferencia, para los efectos de esta explicación se considerará uno de ellos, llamado el del robo de ciclo. En general el controlador de DMA usado con esta técnica, es capaz de generar dos tipos de interrupciones: una interrupción para avisar que requiere usar el canal de DMA y la otra para producir una interrupción normal, que la usa el controlador para interrumpir al microprocesador cuando ha terminado la transferencia del bloque de información.

Una vez inicializado el controlador de DMA, el mismo permanece en espera de que el periférico le avise que está listo

para transferir una palabra; en este momento el controlador de DMA activa la señal de interrupción por DMA, para avisar al microprocesador que requiere realizar la transferencia de una palabra vía el canal de DMA; el microprocesador al reconocer que hay una petición de transferencia vía DMA, reacciona desconectandose del bus de datos y del de direcciones (colocando sus salidas en alta impedancia), dejando los mismos bajo control del controlador del DMA, el cual, vaciará el contenido de su registro de direcciones en el bus de direcciones y ordenará a través del bus de control, la realización de una operación de lectura o escritura en memoria (dependiendo de si va a realizar una transferencia de memoria al periférico o del periférico a memoria); el controlador de DMA, despues de haber realizado la transferencia de la palabra, chequea a ver si su registro contador de palabras a transferir todavía no ha llegado a cero, en cuyo caso, procede a decrementar en uno dicho registro e incrementa en uno el registro de direcciones, de manera de apuntar a la siguiente dirección de la RAM y repite de nuevo la secuencia que se acaba de describir, hasta que el registro contador de palabras llega a cero, en cuyo caso, el controlador de DMA genera una interrupción normal, que hace que el microprocesador arranque la subrutina que realiza la etapa completadora de la transferencia.

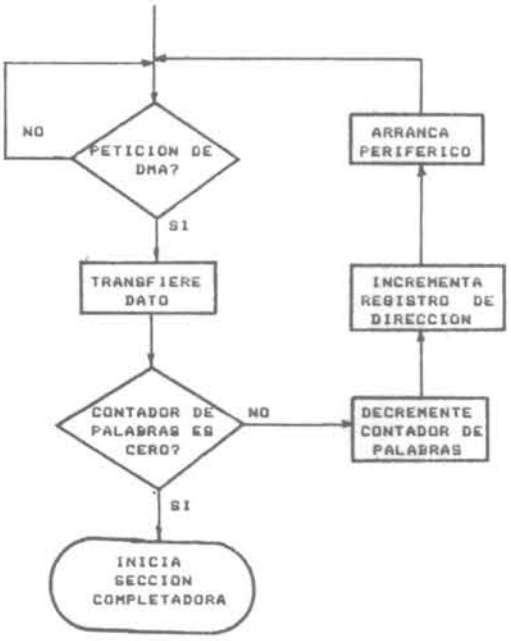


Figura 5-16. Etapa de transferencia del DMA mediante la técnica de robo de ciclo

Etapa completadora

Cuando el microprocesador es interrumpido por el controlador de DMA en forma normal (sin solicitar el uso del canal de DMA), este atiende la interrupción mediante una subrutina de servicio del periférico, cuya finalidad es llevar a cabo la etapa completadora de la transferencia; esta etapa puede ser usada para identificar el area de memoria que se ha transferido o para avisar al programa del usuario, la completación de la transferencia de la información que el había solicitado. Notese que las etapas inicializadora y completadora son responsabilidad del programador y en ellas si interviene activamente el microprocesador, mientras que la etapa de transferencia, es responsabilidad del controlador de DMA y en la misma la intervención del microprocesador se limita a desconectarse del resto de los elementos del sistema, por un ciclo de máquina, mientras el controlador de DMA realiza la transferencia de la palabra.

Técnicas de implementación de DMA

Hay muchas formas de implementar un canal de DMA; en los sistemas a microprocesadores generalmente se emplean tres técnicas:

DMA por HALT (parada) del microprocesador

DMA por compartimiento del tiempo con el microprocesador

DMA por robo de un ciclo de máquina del microprocesador

DMA por HALT del microprocesador

Esta técnica de implementación de DMA aprovecha una señal de control que traen algunos microprocesadores, mediante la cual es posible colocar al microprocesador en un estado de HOLD (parado) en forma indefinida; en el estado de HOLD, el microprocesador se aisla del bus de datos y del de direcciones, mientras dure la señal que lo lleva a esta condición. Mediante esta técnica es posible transferir un número indefinido de palabras como un bloque, pero mientras dure la transferencia del bloque, el programa del usuario permanece detenido.

El microprocesador 8080 de INTEL, tiene la capacidad de trabajar con un controlador de DMA de este tipo, disponiendo de una entrada de control llamada HOLD, y de una salida de control

llamada HOLDA, que le permiten hacer el handshaking con el controlador de DMA. Un uno lógico sobre la línea de HOLD le indica al microprocesador que un periférico está solicitando controlar el bus de datos y el de direcciones; el microprocesador atiende esta solicitud colocando sus salidas a los buses de dirección y datos, en el estado de alta impedancia, avisando de esta condición al periférico que está haciendo la solicitud, mediante la señal HOLDA (reconocimiento del HOLD), tal como se muestra en la figura 5-17. El controlador de DMA está diseñado de tal manera que no inicia ninguna transferencia hasta no recibir la señal HOLDA del microprocesador.

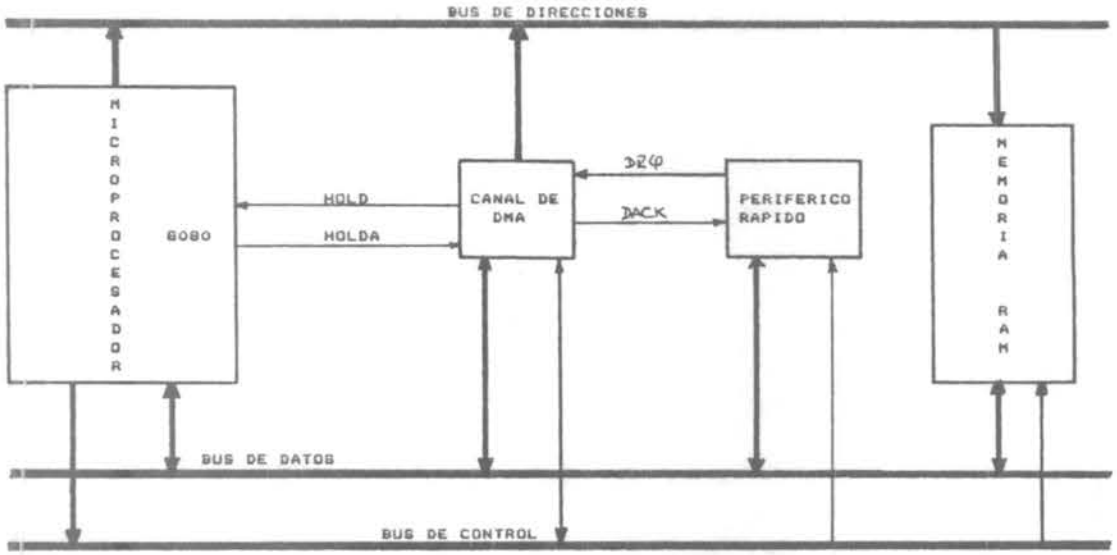


Figura 5-17. Implementación de DMA usando el estado HOLD en el 8080

DMA por compartimiento de tiempo

Esta técnica es usada por algunos microprocesadores, los cuales permiten compartir el tiempo y el uso de los buses de dirección y datos; en este método, el microprocesador al tener una solicitud de interrupción por parte de un periférico para hacer una transferencia directa a memoria, libera por un determinado número de ciclos el bus de dirección y el de datos, los cuales son aprovechados por el controlador de DMA para realizar la transferencia. El microprocesador mediante una señal de control, es obligado por el controlador de DMA a operar en una forma especial, en la cual, el uso de los buses de dirección, control y datos, son compartidos entre el microprocesador y el controlador de DMA. En general, por cada ciclo de máquina que el

microprocesador use los buses, dejara disponible al controlador de DMA varios ciclos de máquina; este número de ciclos viene predeterminado en algunos microprocesadores. Cuando no se tenga ninguna petición de uso del canal de DMA, el microprocesador operará normalmente (sin detenerse por el número de ciclos predeterminado).

DMA por robo de ciclo

Esta tercera técnica empleada en la implementación de DMA, permite transferir datos entre periféricos y memoria directamente, en forma perfectamente sincronizada con el microprocesador; esta técnica es conocida como el método de robo de ciclo. El controlador de DMA se apodera de los buses, solo por el tiempo necesario para transferir una palabra, normalmente un ciclo de máquina; la operación de este tipo de controlador fué la que se describió anteriormente cuando se explicaba la etapa de transferencia en la operación del DMA. Este último método de implementación de DMA es casi transparente al usuario, pero es el mas lento, pues solo transfiere una palabra en cada ciclo robado. El primer método, que transfiere datos durante el estado HALT, retarda considerablemente la ejecución del programa del usuario, pero provee atención continua al DMA y por tiempo indefinido. El método de compartir tiempo, constituye un compromiso entre estos dos extremos.

Implementación de DMA para el 8035

El microprocesador 8035, no tiene previsión para hacer transferencia con DMA; sin embargo, la implementación de un canal de DMA se puede lograr aprovechando la entrada de $\overline{SS}$ (Single Step) del 8035 adicionandole un hardware externo para aislar el bus de datos y el de direcciones, cuando se desee hacer transferencia vía DMA. En el capítulo II se vió que la señal $\overline{SS}$ permite ir ejecutando un programa instrucción por instrucción; el microprocesador, a cada pulso aplicado a la entrada $\overline{SS}$, ejecuta una sola instrucción y se detiene, pero no aisla el bus de datos, sino que mantiene en el bus la dirección de la próxima instrucción a ejecutar. Por otro lado, si se usa, como es el caso del IMSAI, un latch externo para la implementación de un bus de direcciones, éste deberá ser aislado de manera de permitir al controlador de DMA tomar control de dicho bus.

Para los efectos de esta implementación se va a usar el controlador de DMA desarrollado por INTEL para la familia 8080, el DMA controler 8237, el cual usa la técnica de transferencia que aprovecha el estado de HALT del microprocesador. Usando este controlador y el hardware adicional para aislar los buses, es posible implementar el DMA con un circuito como el que se muestra

en la figura 5-18.

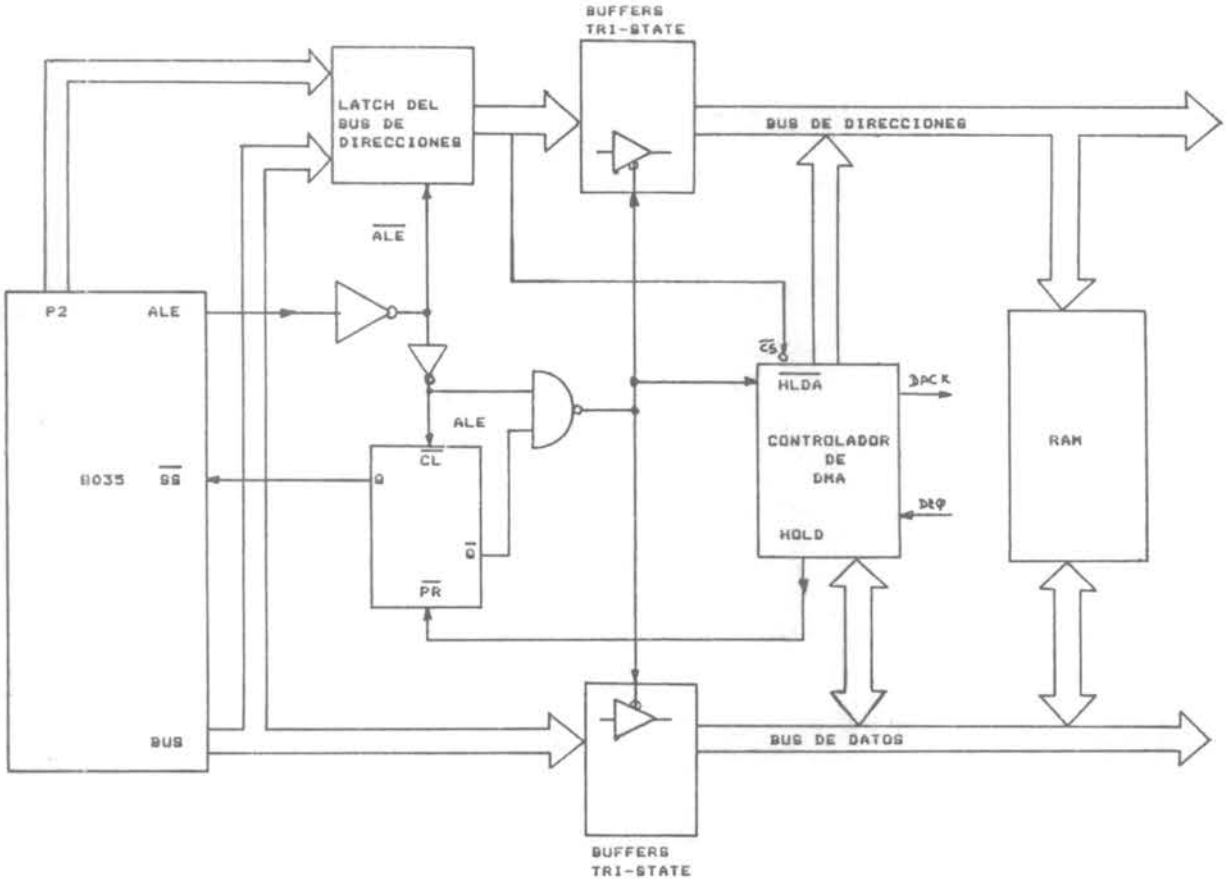


Figura 5-18. Implementación de un canal de DMA en el sistema IMSAI

La operación del circuito de la figura 5-18 es como sigue: la señal HOLD, con la cual el controlador de DMA avisa que requiere hacer una transferencia de datos directa a memoria, se aplica directamente al PRESET del flip-flop (en este flip-flop el PRESET tiene precedencia sobre el CLEAR); normalmente, cuando el controlador de DMA no requiere hacer ninguna transferencia, la señal HOLD se mantiene en cero (ver figura 5-18), con lo cual la salida Q del flip-flop, que es tambien la entrada $\overline{SS}$ del microprocesador, se mantiene en alto (desactivada), por lo que el microprocesador opera normalmente. Al momento de requerir hacer una transferencia, el controlador de DMA activa la señal HOLD, colocandola en uno; como la señal ALE, que está continuamente cambiando entre uno y cero, se aplica directamente al CLEAR del flip-flop, la primera vez que vaya al estado cero despues que la señal HOLD se active, forzará en el flip-flop un CLEAR, con lo cual, la salida Q del mismo, que está conectada a la entrada $\overline{SS}$

del microprocesador, se va a cero, indicandole al microprocesador que va a operar en la modalidad single step. En respuesta a la activación de la señal $\overline{SS}$, el microprocesador, despues de terminar de ejecutar la instrucción en curso, coloca la señal ALE en uno y detiene su operación por un tiempo indefinido, que dependerá de la duración de la señal aplicada en $\overline{SS}$. La señal ALE (en uno), en conjunto con el negado de la salida del flip-flop, se pasan por una compuerta NAND para producir la señal HLDA; esta señal se usa para avisar al controlador de DMA que la petición de acceso directo a memoria ha sido reconocida y para deshabilitar los buffers tri-state del bus de direcciones y de datos, de manera de que el controlador de DMA tome control de los mismos. Al momento de terminar la transferencia directa hacia o desde memoria, el controlador de DMA coloca la señal de HOLD en cero, con lo cual la salida del flip-flop se va a uno, desactivandose en consecuencia la señal $\overline{SS}$, lo que hace que el microprocesador salga de la modalidad de operación single step y comience a operar normalmente. Es de observar que mientras la señal HOLD esté en cero (desactivada), los cambios en la señal ALE no afectan la salida del flip-flop en virtud de que la señal de PRESET (HOLD), tiene precedencia sobre la señal CLEAR.

La figura 5-19 muestra un diagrama de tiempo de una transferencia de datos vía DMA con el circuito de la figura 5-18.

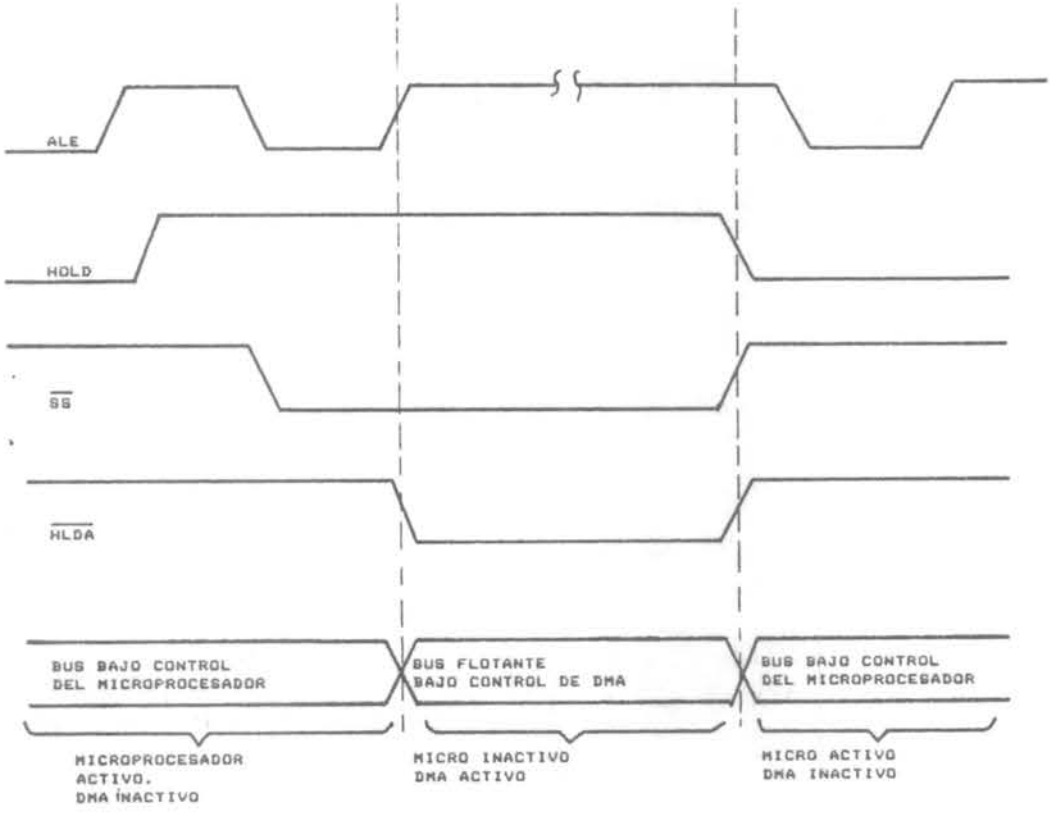


Figura 5-19. Diagrama de tiempo de una transferencia vía DMA con el circuito de la figura 5-18

5.5. CONEXION DEL MICROPROCESADOR CON EL MUNDO ANALOGICO

Hasta el momento se ha conectado el microprocesador con periféricos lentos y periféricos rapidos; esta conexión ha sido entre dispositivos digitales, sin embargo, el mundo exterior es basicamente analógico, de manera que si se desea que un microprocesador controle un proceso, se requerirá que el mismo controle señales analógicas tales como presión, temperatura, concentración, flujo, etc. para lo cual ejecutará los programas que implementan los algorítmos de control, que permitirán muestrear y procesar las variables de entrada al proceso (analógicas) y generar posteriormente las señales que corresponda, segun los resultados obtenidos del procesamiento de las variables de entrada, de manera de mantener el proceso dentro de los límites y parámetros requeridos.

Puesto que el microprocesador solo trabaja con señales digitales, la manipulación de señales analógicas requiere de la conversión previa de la señal analógica a digital, de forma de hacerlas manipulables por el microprocesador; de la misma manera, muchos de los dispositivos de control tales como válvulas, actuadores, etc., requieren de una señal analógica para su operación, por lo cual las señales de control generadas por el microprocesador para controlar el proceso, deberán ser convertidas previamente a señales analógicas, antes de ser aplicadas al dispositivo final de control. Esta sección tiene por finalidad estudiar brevemente los elementos de hardware utilizados para llevar a cabo las conversiones analógicas a digitales y las digitales a analógicas y los principios en los que se basa el proceso de conversión de ambos dispositivos.

5.5.1. CONVERSION ANALOGICA/DIGITAL

Una señal analógica se caracteriza por que, dentro de su rango y dominio puede tomar cualquier valor, en otras palabras, puede tomar infinitos valores.

El proceso de conversión de una señal analógica en digital involucra basicamente dos aspectos: el proceso de muestreo de la señal analógica y el proceso de conversión propiamente dicho.

Las señales digitales, a diferencia de las analógicas se caracterizan por quetienen un rango y un dominio discreto o sea, solo pueden tomar ciertos valores claramente definídos, de manera que, el convertir una señal analógica a digital, implica pasar de una señal que puede tomar cualquier valor , a una que solo puede tomar ciertos y determinados valores. La conversión analógica a digital involucra siempre un muestreo de la señal a convertir; este muestreo se debe hacer a una rata tal que garantize que la señal discretizada contiene basicamente la misma información que la señal analógica. La rata a la cual una señal analógica debe ser muestreada con el fín de garantizar sea

representativa de la señal analógica original, depende del contenido armónico de la misma; cualquier señal analógica se puede considerar constituida por la suma ponderada de una señal senoidal de una determinada frecuencia fundamental, con otras señales senoidales de frecuencias multiplos de la fundamental; estas otras señales con frecuencias multiplos de la fundamental, constituyen lo que se conoce con el nombre de armónicos. Se ha demostrado matemáticamente que la rata a la que se debe muestrear una señal analógica para garantizar que la señal muestreada contenga basicamente la misma información que la señal analógica original, tiene que ser por lo menos igual a dos veces la frecuencia del armónico significativo mas alto; por ejemplo, si se tiene una señal analógica con una frecuencia fundamental de 300Hz y un contenido armónico significativo hasta 1500Hz, se deberá muestrear a una rata de por lo menos 3000 veces por segundo (2x1500).

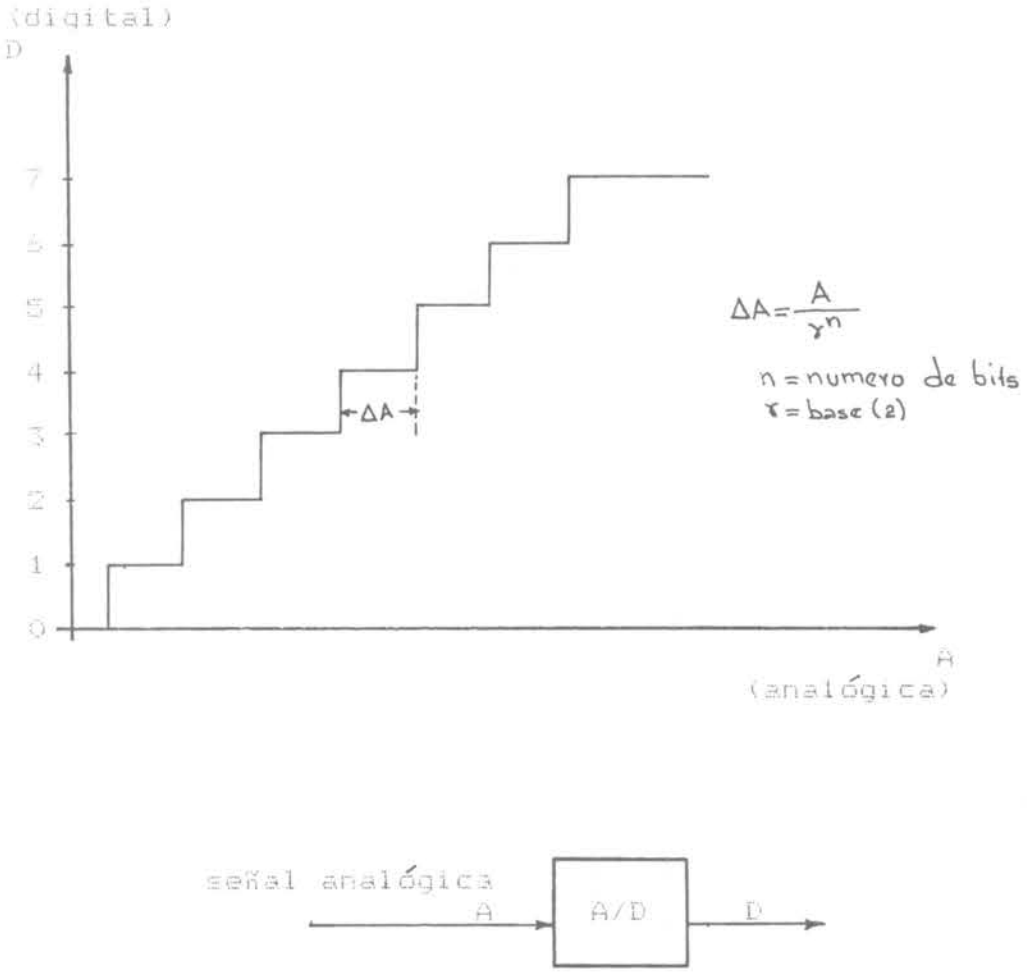


Figura 3-20. Relación entre la entrada y la salida en un conversor A/D

El otro aspecto en el proceso de conversión A/D, lo constituye la conversión propiamente dicha. El conversor A/D es un dispositivo que tiene por finalidad convertir una señal analógica, en una señal discreta, que solo puede tomar ciertos y determinados valores; la relación entre la señal de entrada y la de salida en un conversor A/D se muestra en la figura 5-20.

Cuando se usa un conversor A/D es necesario tener presente una serie de consideraciones a fín de obtener el resultado más apropiado a la aplicación específica. Es razonable pensar que mientras más muestras se tomen de la señal analógica y mas conversiones se realicen, mas representativa será la señal digital de la original analógica ; es deseable por tanto, tomar el mayor número de muestras de la señal y hacer las respectivas conversiones, sin embargo, es necesario tener presente que los conversores A/D, tienen un cierto tiempo de conversión, es decir, requieren de un tiempo finito para convertir una muestra de la señal analógica en un valor discreto y por lo tanto, el número de conversiones por unidad de tiempo es limitado; por otro lado tambien es importante tener presente el factor costo, ya que mientras mas rápido es el conversor mayor es su costo, de manera que no siempre el conversor mas rápido es la solución mas apropiada y conveniente en una determinada aplicación; en todo caso los requerimientos mínimos de muestreo y conversión los determinará como ya se mencionó anteriormente, el contenido armónico significativo de la señal analógica.

Cuando se usan conversores A/D es importante tambien tener presente los errores y limitaciones que el proceso de conversión genera. El error de cuantización es inherente al proceso de conversión A/D y tiene que ver con el hecho de pasar de una señal que puede tomar infinitos valores (analógica), a una que solo puede tomar valores discretos (digital), por lo que necesariamente valores distintos de la señal analógica tendrán el mismo valor digital; existe un diferencial mínimo de señal analógica que el conversor es capaz de diferenciar y reflejar a la salida como un valor diferente; el valor de este diferencial, depende del número de bits del conversor; mientras más bits tenga el conversor, más pequeño será el valor diferencial que es capaz de detectar el conversor. Una medida de la capacidad del conversor de diferenciar entre dos valores distintos de señal analógica es la especificación de su resolución; la resolución de un conversor puede ser expresada por el número de bits del conversor o como el inverso de 2 elevado al número de bits del conversor (resolución = $1/2^N$).

Varias técnicas se emplean en la construcción de conversores A/D, que va desde el de un simple integrador hasta el conversor A/D por aproximaciones sucesivas, que usa un conversor digital a analógico (D/A), como parte del dispositivo que realiza la conversión A/D. En la sección siguiente se dará un algorítmo que permite implementar un conversor A/D en base a un microprocesador y a un conversor D/A; el detalle de las diferentes técnicas y fundamentos de los diferentes tipos de

conversores puede ser estudiado en los libros que sobre el tema aparecen en la bibliografía al final del presente trabajo.

Usando una de las técnicas descritas de I/O MAPPED I/O y MEMORY MAPPED I/O, se puede conectar en forma relativamente sencilla y sin mucho hardware adicional, un conversor A/D a un microprocesador; si se emplea la técnica de MEMORY MAPPED I/O, solo se requerirá una lógica decodificadora y unos buffers de tres estados. En todo conversor A/D se encuentran dos señales de control que se usan para indicar el inicio y fin de la conversión. La señal START OF CONVERSION (SOC), es una entrada al conversor A/D y se usa para indicarle al mismo que inicie la conversión de la señal presente en su entrada analógica; la señal EOC, es una salida del conversor A/D que se usa para indicar que el conversor ya realizó la conversión de la señal analógica en digital y que por tanto en las salidas digitales del mismo se encuentra el equivalente digital de esta señal analógica.

La figura 5-21 muestra un esquema de conexión de un conversor A/D al microprocesador 8035, mediante la técnica de Memory Mapped I/O; se usa la entrada T0 del 8035 para el muestreo de la señal EOC (End of Conversion). Para la generación de la señal de SOC (Start of Conversion), se decodifica la dirección en donde se conectará el conversor y en combinación con la señal $\overline{WR}$ del 8035 se genera la señal SOC. Los buffers de tres estados se habilitan cuando se hace una operación de lectura en la dirección que se ha seleccionado para la conexión del conversor A/D (en este caso específico se ha seleccionado la dirección 800H).

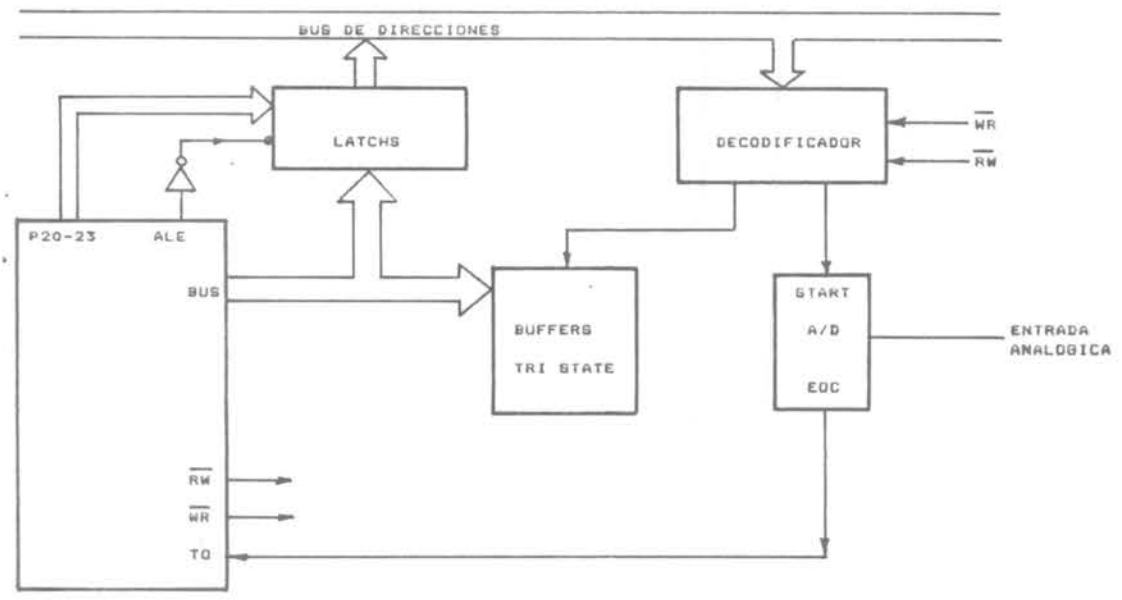


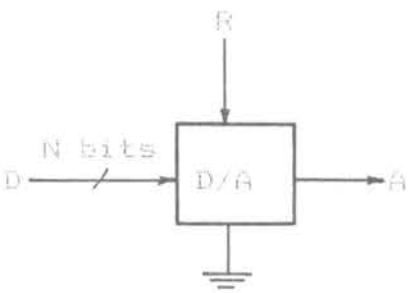
Figura 5-21. Conexión de un conversor A/D al 8035

5.5.2. CONVERSION DIGITAL/ANALOGICA

El conversor D/A es un dispositivo que tiene por finalidad convertir una señal discreta, representada por un conjunto de bits, en una señal analógica equivalente.

Basicamente el conversor D/A usa la señal digital, representada con N bits, y una señal de referencia, para generar la señal analógica equivalente; la siguiente ecuación sintetiza la operación del conversor D/A.

$A = R \times D$ en donde: R = referencia
D = señal digital
A = señal analógica



La relación entre la señal de entrada a un conversor D/A y la señal de salida (función de transferencia), se muestra en la figura 5-22.

Para la realización del proceso de conversión D/A se han implementado diferentes técnicas; en este trabajo no se estudiarán las diferentes técnicas de implementación de los conversores D/A, el lector podrá consultar cualquiera de los libros que aparecen al final del trabajo como referencias bibliográficas. Todas estas técnicas involucran el uso de elementos de precisión tales como resistores, condensadores, etc., y elementos como fuentes de voltaje de referencia, amplificadores operacionales, comparadores etc., que influyen en el comportamiento global del conversor; muchos de los errores y limitaciones de los conversores D/A las determinan las tolerancias y desviaciones de los diferentes elementos que se usan en su implementación.

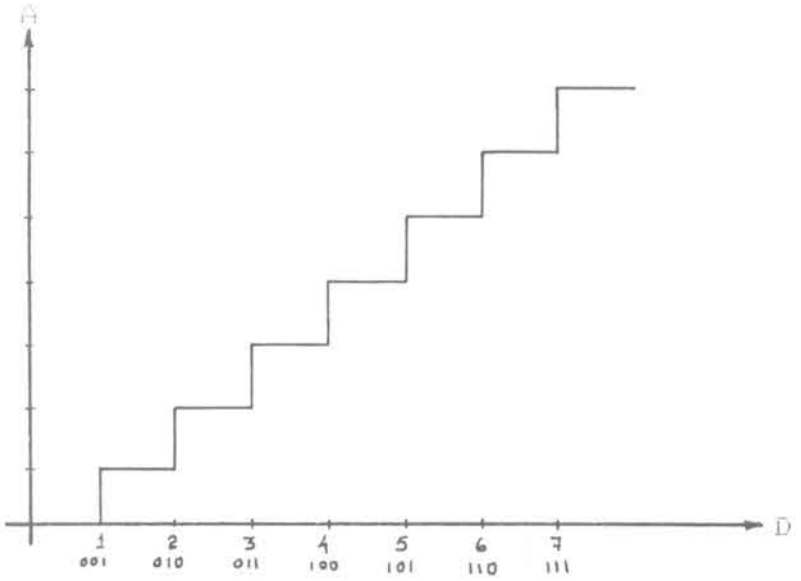


Figura 5-22. Relación entre la entrada y la salida en un conversor D/A

La conexión de un conversor D/A a un microprocesador es relativamente más sencilla que la conexión de un conversor A/D, pues en el mismo no se requieren las señales de control SOC y EOC presentes en los conversores A/D. La figura 5-23 muestra la conexión de un conversor D/A de 8 bits al microprocesador 8035, mediante la técnica de Memory Mapped I/O; obsérvese que solo se requiere la señal de carga de los latchs que mantienen el dato estable en las entradas del conversor; esta señal puede ser generada mediante la decodificación de la dirección en conjunto con la señal $\overline{WR}$ del microprocesador. En el caso de requerir usar un conversor que tenga un número de bits mayor que el del bus de datos, se puede adoptar la solución de conectar el conversor mediante una interfase programable (PPI) o conectar el conversor en los puertos del microprocesador; esta última solución se muestra en la figura 5-24, en la cual se ha conectado un conversor D/A de 10 bits en el puerto 1 y en el puerto 2 del 8035; se usa el bit mas significativo del puerto 2 para cargar los latch que mantienen el dato en las entradas del conversor D/A. A continuación se presenta un programa que permitirá manejar el conversor D/A de la figura 5-24. Se ha supuesto que el dato a convertir se encuentra en dos direcciones adyacentes de la memoria interna; los 8 bits menos significativos se encuentran apuntados por R1 y los bits mas significativos del dato, se encuentran en los dos bits menos significativos del dato contenido en la dirección que sigue a la apuntada por R1.

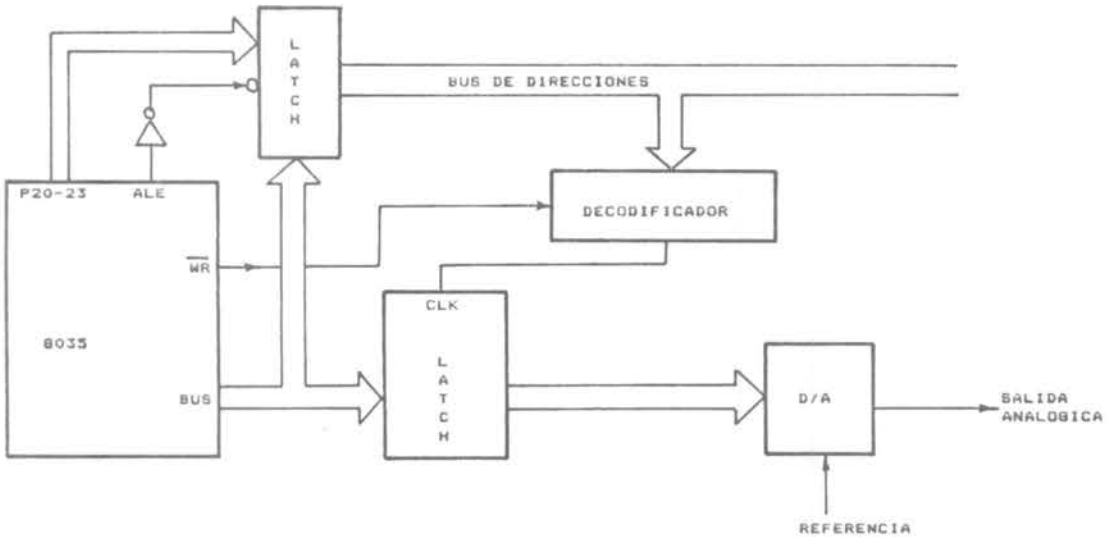


Figura 5-23. Conexión de un conversor D/A al microprocesador 8035

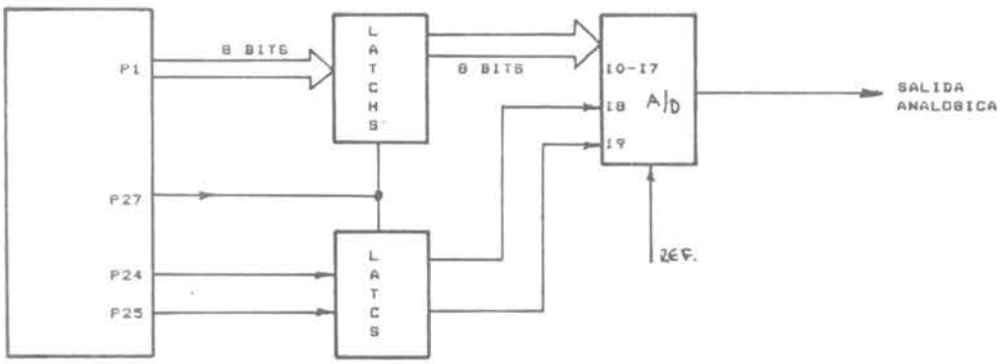


Figura 5-24. Conexión de un conversor D/A de 10 bits al microprocesador 8035

```
MOV A,@r1      Mueve  al  acumulador los bits menos
               significativos  del dato a convertir

OUTL R1,A      Saca  los  bits menos significativos
               del  dato por el puerto 1
INC R1         Apunta  al segundo byte que contiene
```

```
MOV A, 0B
ANL A,@R1
SWAP A
OUTL P2,A

MOV A, 00
OUTL P2,A
```

Los bits mas significativos del dato (R1)0-1, se colocan en los bits A4 y A5 del acumulador. Se coloca tambien el bit 7 del acumulador en uno para producir el pulso de carga de los latchs.

C A P I T U L O V I

6.1. INTERFASES PROGRAMABLES

Casi todos los microprocesadores poseen una serie de componentes o chips de soporte que facilitan la comunicación del mismo con el mundo exterior; estos chips, productos tambien de tecnicas de integración a gran escala, constituyen interfases muy versátiles ya que en general la operación y/o función que realizan puede ser establecida desde el microprocesador mediante los comandos apropiados, de allí el nombre de interfases programables.

Existen varios tipos de interfases desarrolladas para los diferentes microprocesadores, en general es posible agruparlas en dos categorías: las interfases seriales, cuya finalidad es permitir la conexión del microprocesador con dispositivos periféricos que envian y reciben información en forma serial; este tipo de interfase tiene la doble función de convertir la información proveniente del microprocesador (en forma paralela) en información serializada para ser transmitida a un equipo periférico y convertir la información proveniente del equipo periférico (en forma serial) en información en paralelo, para que sea tomada por el microprocesador. Y las interfases paralelas de propósito general, que permiten la conexión al microprocesador de una gran variedad de periféricos en forma muy simple y con muy poco hardware adicional. A continuación se describirán algunos elementos de interfases, pertenecientes a los tipos antes mencionados.

6.1.1. INTERFASES SERIALES

Las interfases seriales como se mencionó anteriormente, tienen la doble función de convertir información paralela en serial para su transmisión y recibir información serial y convertirla en paralela. Adicional a la simple tarea de serializar o deserealizar, estas interfases realizan una serie de chequeos y ediciones en la información que transmiten o reciben, a fin de facilitar la verificación y el chequeo de la información recibida y/o transmitida.

6.1.1.1. Tópicos de comunicación de datos

Antes de entrar en la descripción de algún chip específico perteneciente a esta categoría de interfase, se van a dar una serie de conceptos relacionados con la comunicación de datos que facilitarán la comprensión de las explicaciones referentes a estas interfases.

Se ha mencionado que los datos dentro de un computador o cualquier dispositivo digital se representan en forma de patrones de digitos binarios llamados bits; la transferencia de información entre dos dispositivos electrónicos se puede hacer sobre una sola línea de comunicación, enviando uno a uno los diferentes bits que componen el dato, o sobre n líneas de comunicación, enviando simultaneamente todos los bits que componen el dato; la primera forma de comunicación se le llama serial y la segunda, comunicación paralela.

La transferencia de información entre dos dispositivos en forma serial puede ser hecha en forma ASINCRONICA o SINCRONICA. En la comunicación SINCRONICA se conoce en forma precisa el tiempo de partida y llegada de cada bit de información, lo cual se logra haciendo llegar al receptor, bien sea por una línea aparte o en conjunto con los datos valiendose de alguna tecnica de modulación, el reloj del transmisor; en cambio en la comunicación asincrónica, no existe sincronismo continuo entre el receptor y el transmisor.

Transmisión Asincrónica

En la comunicación ASINCRONICA no se conoce el tiempo de partida de los bits del caracter y el reloj del transmisor no se hace llegar al receptor; en este tipo de transmisión, en virtud de la ausencia de un reloj que acompañe al dato propiamente dicho y a la carencia de un continuo acuerdo entre el receptor y el transmisor, es necesario implementar algún mecanismo que permita al momento de una transmisión, el "enganche" entre ambos, de manera que el receptor pueda identificar en forma precisa, cada bit que llegue por la línea, hasta ensamblar un caracter completo.

Se han adoptado una serie de convenciones para la comunicación asincrónica y un formato o patron general bajo el cual se deben enviar los caracteres; la figura 6-1 es una representación del formato de envío de un caracter para transmisión serial asincrónica. A continuación una breve explicación del mismo y de las convenciones establecidas para esos propósitos.

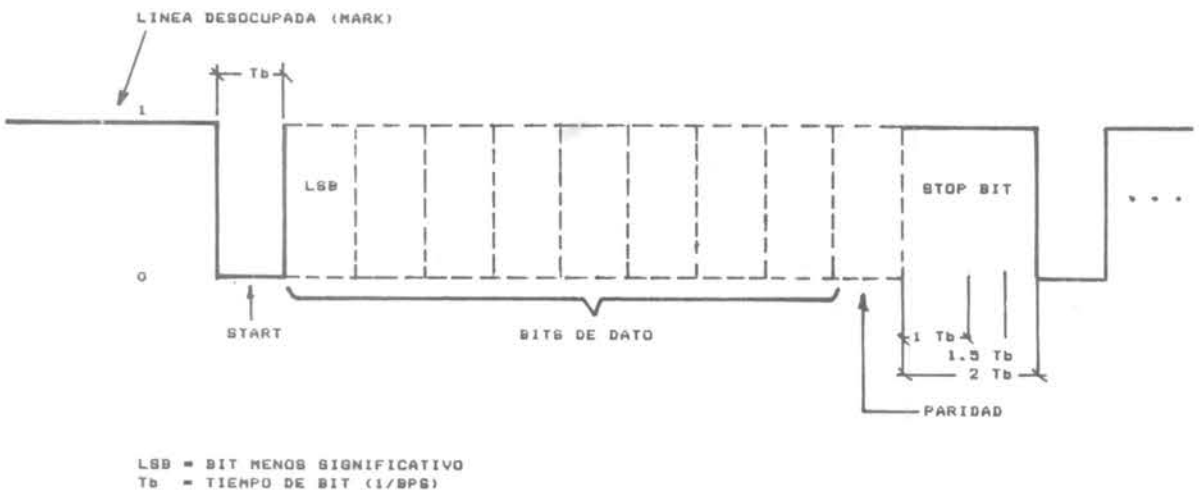


Figura 6-1. Formato para transmisión serial asincrónica

Se ha establecido por convención, que la salida de un transmisor cuando no este transmitiendo ninguna información, debe permanecer en un estado o condición tal, que la corriente fluya hacia la línea de transmisión; esta condición es llamada condición de MARK o ESTADO 1; la condición opuesta, o sea cuando la corriente fluye de la línea al transmisor, es llamada condición de SPACE. El transmisor mantendra siempre una condición de MARK en la línea, cuando no esté transmitiendo.

Para iniciar una transmisión, el transmisor cambia la condición de la línea desde el estado MARK (desocupada) a el estado SPACE (ocupada) y mantiene esta señal por un tiempo de bit(Tb); este primer bit del formato, en virtud de la función que cumple, se le llama START bit. En el lado del receptor, para ser consistente con la convención adoptada, se diseña el mecanísmo de recepción de manera que se active cuando se detecte en la línea de transmisión una TRANSICION de la condición de MARK a la condición de SPACE.

Despues que se ha enviado el START bit, para activar el mecanísmo de recepción del receptor, se procede a enviar uno a uno los bits qué constituyen el caracter, comenzando por el menos significativo(LSB); el número de bits por caracter depende del código que se esté usando para representar los datos; cada bit del caracter será mantenido en la línea por un tiempo que lo determinará la velocidad a la que se esté transmitiendo; ese tiempo, de igual duración para todos los bits, se conoce con el nombre de TIEMPO DE BIT(Tb) y en general es igual al inverso de la velocidad de transmisión expresada en bits por segundo.

Despues de los bits de datos propiamente dichos, se agrega un bit adicional (este bit es opcional) para permitir el chequeo de paridad por parte del receptor, proveyendo así un elemento de verificación de la información recibida, lo que reduce la probabilidad de recepción de información errada.

El final de la transmisión de un caracter debe ser avisado al receptor de manera que el mecanísmo de recepción permanezca en condición o se prepare para detectar un eventual inicio de transmisión de otro caracter. La forma que en una transmisión asincrónica se le avisa al receptor el final de una transmisión, es manteniendo la línea de transmisión por un periodo determinado en la condición de MARK, despues que se haya transmitido el bit de paridad (si lo hubiere) o el último bit del caracter. Este periodo de tiempo que marca el final de la transmisión de un caracter, se conoce como INTERVALO DE STOP o STOP BIT. El INTERVALO DE STOP debe tener como mínimo una duración de un tiempo de bit y en general puede variar entre 1, 1.5 y 2 tiempos de bits.

En resumen, en la comunicación serial asincrónica, conocida tambien como comunicación START-STOP, el transmisor y el receptor no se mantienen en sincronismo continuamente sino, que por cada caracter a transmitir se debe establecer el sincronismo del receptor con el transmisor. La transmisión de un caracter se inicia con la transmisión del bit de START y finaliza con el intervalo de STOP, siendo a través de estos dos bits que se logra sincronizar el receptor con el transmisor; los bits que constituyen el caracter se transmiten entre estos dos bits que marcan el inicio y el fin de la transmisión.

Transmisión Sincrónica

En la transmisión sincrónica el transmisor y el receptor se mantienen continuamente en sincronismo y los bits de los diferentes caracteres son enviados uno tras otro sin elementos (bits) adicionales que marquen el inicio y fin de cada caracter. La ausencia de los bits START y STOP en la transmisión sincrónica permite aprovechar un 20% adicional del tiempo de la línea para la transmisión de datos (en tranasmisión asincrónica, con una longitud de caracter de 8 bits, se requieren 2 bits adicionales por cada caracter o sea un total de 10 bits por caracter, de los cuales 8 corresponden al dato y 2 se usan exclusivamente para propósitos de sincronismo), sin embargo, se hace necesario proveer al receptor de un mecanismo que permita delimitar en la cadena de bits que recibe por la línea, cuales corresponden a un caracter determinado.

La figura 6-2 muestra el formato transmisión sincrónica, en el cual se puede apreciar que la delimitación de los bits de cada caracter se consigue, enviando un caracter

especial, llamado caracter de sincronismo, referido comunmente como SYNC; este caracter se transmite al receptor en forma previa a los caracteres de información o datos. Existen algunas variaciones en los sistemas de transmision sincrónica respecto al número de caracteres SYNC que se deben incluir en la transmisión, antes de cada paquete o bloque de datos, a fin de garantizar el ensamblaje perfecto de los caracteres en el lado del receptor, sin embargo, para los efectos de la siguiente explicación se considerará que el sistema requiere de un solo caracter SYNC.

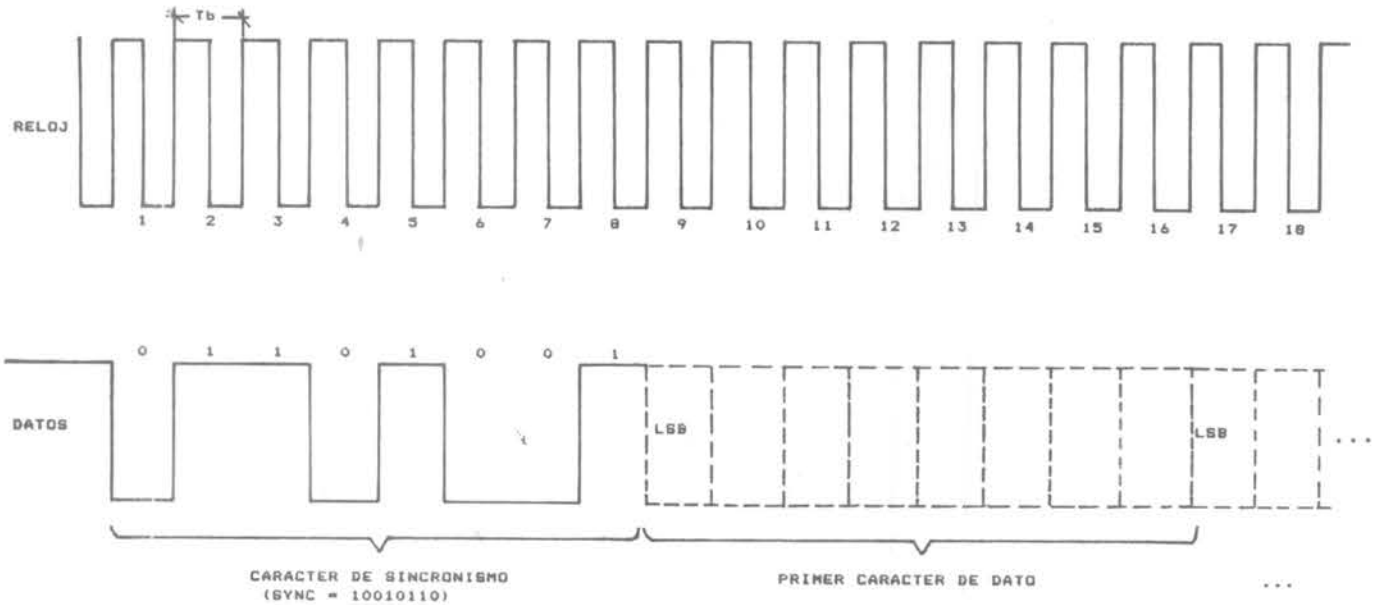


Figura 6-2. Formato de transmisión Sincrónica

Los receptores son dotados de un mecanismo tal, (dicho mecánismo puede ser implementado por hardware o software) que al momento de comenzar una transmisión, se colocan en un modo especial de operación llamado de "busqueda de SYNC"; en este modo, el receptor comienza a ensamblar los bits que llegan por la línea hasta obtener un posible caracter, compara este posible caracter con el caracter definido como SYNC, y si coinciden, el receptor está perfectamente acoplado con el transmisor y se procede al ensamblaje de los caracteres a partir de los bits que llegan por la línea; si el caracter que se "armó" inicialmente no coincide con el de SYNC que se haya definido, el receptor se mantiene en el modo de "busqueda de SYNC" hasta conseguir armar un caracter coincidente con el definido como SYNC.

El caracter SYNC se escoge de manera tal que su arreglo o patron de bits sea significativamente diferente del patron de

bits que se formaría con los caracteres de datos transmitidos en forma "empaquetada", unos a continuación de los otros. El caracter ASCII 226_8 se usa frecuentemente en muchos sistemas de transmisión sincrónica como caracter SYNC. El paso en un momento dado del receptor al modo de "busqueda de SYNC", puede ser decidido en cualquier momento, bien sea por mecanismos de hardware o por software, para resincronizar el receptor y el transmisor

En este punto es necesario resaltar que si bien en la transmisión sincrónica es necesario enviar uno o dos caracteres completos para sincronizar el receptor con el transmisor, contra los dos bits requeridos en la transmisión asincrónica, en ésta última se requiere hacer esta sincronización por cada caracter transmitido, en cambio en la sincrónica, solo se requiere hacer la sincronización al comienzo de un bloque de caracteres, y entre caracteres, no se requiere de elementos adicionales de sincronismo.

Un parametro de gran interes en los sistemas de transmisión, sincrónicos o asincrónicos, es la velocidad de transmisión. La velocidad de transmisión se acostumbra a expresarla en BAUDS y en BITS POR SEGUNDO (bps); los términos BAUD y BIT POR SEGUNDO no tienen necesariamente que ser iguales en un sistema de transmisión, aun cuando generalmente si lo son; es común que las personas tiendan a usarlos indistintamente, pudiendo algunas veces incurrir en errores, por la confusión de estos términos. La rata de BAUD es una medida del número de transiciones por segundo que ocurren en la señal transmitida o sea, es una medida de la rata de modulación de la señal transmitida; en cambio los bps, son como su nombre lo indica, el número de bits transmitidos en un segundo.

En un sistema de transmision binario, en el cual las señales solo puede tomar dos posibles estados (pudiendo estar representados estos estados por un par de voltajes distintos, un par de frecuencias distintas o un par de ángulos o fases distintas de la señal), la rata de modulación y la rata de bps serán SIEMPRE iguales y en estos casos el número de bauds y el de bps son coincidencialmente iguales; sin embargo, en sistemas de transmisión NO binarios, en los cuales la señal puede tomar mas de dos estados, la rata de baud y la rata de bits por segundo no tienen necesariamente que coincidir. El siguiente ejemplo permitirá aclarar lo antes dicho:

Supongase que se tiene un transmisor capaz de enviar señales con cuatro niveles de voltaje distintos, bien diferenciados (cuatro estados); si estos cuatro niveles de voltaje se usan para definir cuatro posibles combinaciones distintas de dos bits consecutivos, tal como se muestra a continuación, se podrán enviar entonces 2 bits a un tiempo en vez de uno.

Nivel de voltaje	Combinación de bits
V1	00
V2	01
V3	10
V4	11

Estas combinaciones constituyen todas las posibles que se pueden obtener con dos bits. Si se quiere enviar un caracter cuyo código o patron de bits sea 10001101, se enviarán los voltajes V2, V4, V1 y V3; el tiempo de duración en la línea de cada nivel de voltaje lo definirá la rata de modulación o sea la rata de baudios; como se puede observar en la figura 6-3, el número de transiciones de la señal por unidad de tiempo (baudios) es de 4, para la transmisión de un caracter completo, y el número de bits transmitidos en esa misma unidad de tiempo (bps) corresponde a 8.

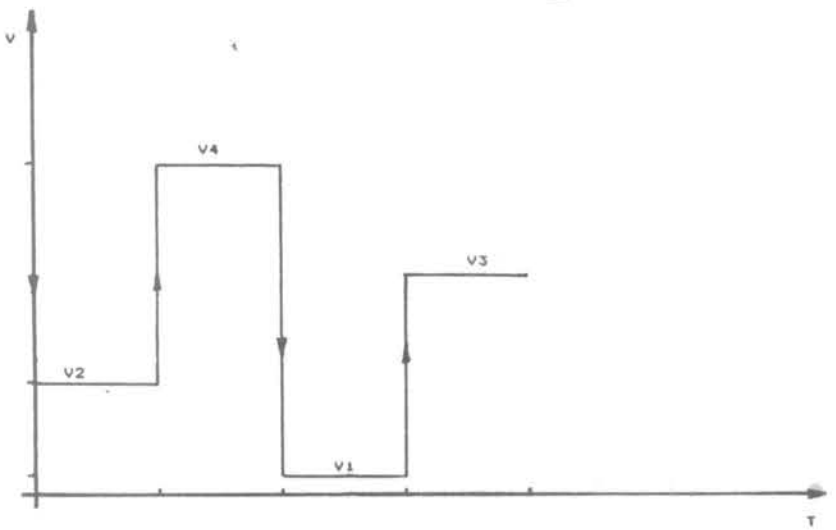


Figura 6-3. Transmisión de un caracter en un sistema no binario

Con un sistema como el del ejemplo anterior, al transmitir a una rata de modulacion de 1200 baudios, se estaran enviando 2400 bits por cada segundo; en el caso de un sistema binario de transmisión a una rata de modulación de 1200 baudios correspoderan 1200 bps.

La transmisión sincrónica o asincrónica, se puede realizar a su vez en dos modalidades llamadas FULL DUPLEX y HALF DUPLEX. En la modalidad FULL DUPLEX, se puede transmitir y recibir información simultaneamente; en la modalidad HALF DUPLEX, no se puede transmitir y recibir simultaneamente, sino en un solo sentido a la vez. Generalmente la transmisión FULL duplex se asocia con cuatro líneas de conexión, un par para el receptor y otro para el transmisor; sin embargo, mediante tecnicas de modulación, es posible tener comunicación full duplex, con dos líneas solamente, usando por ejemplo, 4 frecuencias distintas, dos para el receptor (una para el cero y una para el uno) y dos para el transmisor (una para el cero y otra para el uno).

6.1.1.2. Transmisor-Receptor Universal Sincrónico Asincrónico (USART 8251A)

El 8251A, es una interfase universal para transmision sincronica y asincronica, usado en una gran variedad de sistemas a microprocesadores como dispositivo periférico debido a su versatilidad que le permite operar usando practicamente cualquiera de las tecnicas empleadas en transmisión serial.

En la figura 6-4a se muestra un diagrama de la asignación de los diferentes pines del USART 8251A y en la 6-4b se muestra un diagrama funcional del mismo, A continuación una breve descripción funcional del 8251A.

El 8251A se puede considerar constituido funcionalmente por 5 bloques:

a) Buffer del bus de datos

b) Lógica de control de lectura/escritura

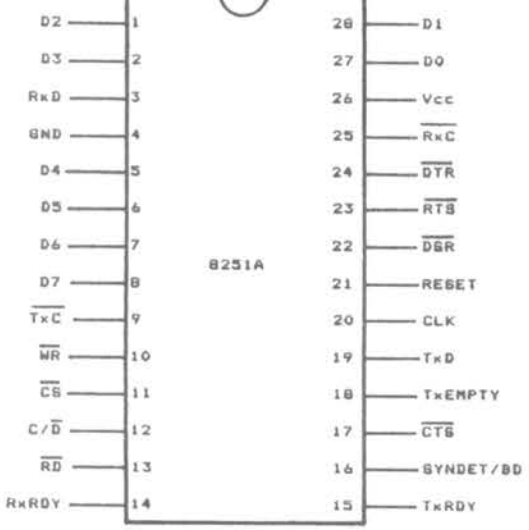
c) Lógica de control para operación con MODEMS

d) Sección receptora

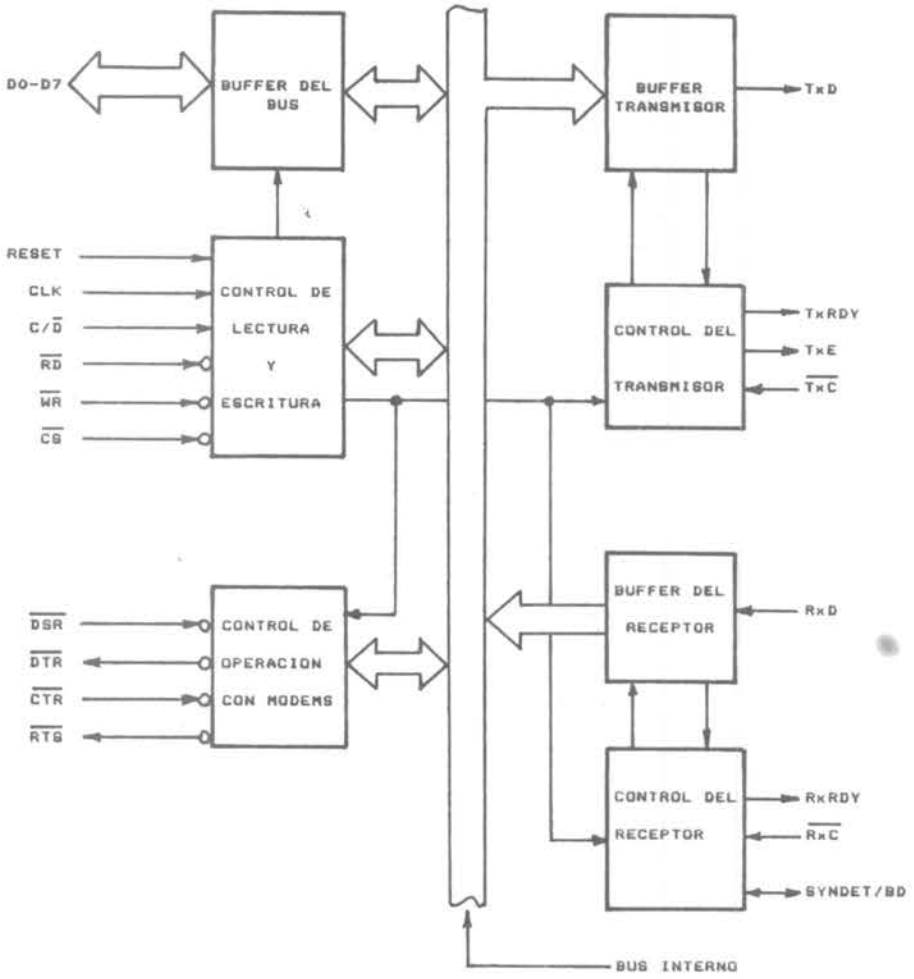
   Buffer del receptor
   Lógica de control

e) Sección transmisora

   Buffer del transmisor
   Lógica de control



A



B

Figura 6-4. USART 8251A. A) Diagrama de pines. B) diagrama de bloques funcional

a) Bus de datos y su Buffer

El bus de datos del 8251A es bidireccional, bufferizado y con salida tri-state. Todas las transferencias entre el microprocesador y el USART se hacen a través de este bus, bien sea que la información sea un dato, un comando de control o información de estatus. Realmente el USART 8251A tiene la posibilidad de trabajar bajo control de programa o por medio del sistema de interrupción del microprocesador que se esté usando, ya que las señales de estatus y control pueden ser "leídas" o "alteradas" a través del bus, por medio de instrucciones enviadas por el programa al USART, o por medio de pines específicos que para esos propósitos trae el USART, utilizando algún hardware adicional, que en general es sumamente simple, que permita conectar el USART al sistema de interrupción del microprocesador.

b) Lógica de control de lectura/escritura

Este bloque funcional es el encargado de recibir las señales de control desde el microprocesador y generar las señales apropiadas para todos los demas bloques funcionales. Contiene este bloque funcional registros para almacenar las palabras que definen la operación funcional del USART.

La forma definitiva como operara el USART 8251A en una aplicación cualquiera, es definida por software; antes de que el USART comience a transmitir o recibir datos, el mismo, debe ser programado mediante unas palabras o instrucciones de control que se le envian a traves del bus de datos. Existen dos tipos de instrucciones de control:

- Instrucciones de control de modo de operación.
- Instrucciones o comandos de operación.

Las instrucciones de control de modo de operación, permiten definirle al USART la modalidad en la que va a operar (sincrónica o asincrónica), ademas permite especificar una serie de parámetros asociados con la modalidad seleccionada tales como: factor de rata de baud, longitud del caracter, tipo de paridad, número de stops bits (en el caso de transmisión asincrónica) y número de caracteres de sincronismo (en el caso de transmisión sincrónica). El formato y la forma de "armar" este caracter para la programación del modo de operación se verá en secciones posteriores.

Las instrucciones de comandos permiten controlar la operación real del USART en el modo programado; funciones tales

como: habilitar o deshabilitar el transmisor o el receptor, el reset de las condiciones de error y de las señales de control en operación con modems, son llevadas a cabo mediante las instrucciones de comando.

A la lógica de control de lectura/escritura entran las siguientes señales:

RESET

CLK (Clock)

$\overline{WR}$ (Write)

$\overline{RD}$ (Read)

C/$\overline{D}$ (Control/Dato

$\overline{CS}$ (Chip select)

La señal RESET, forza al 8251A a la condición "libre" en la cual está en espera de la instrucción de modo que defina la forma como va a operar; esta señal está activa cuando se encuentra en el nivel uno lógico. Al 8251A tambien se le puede dar un RESET por software, a través del bus de datos, con una instrucción de comando, como se verá mas adelante.

La señal $\overline{CS}$, habilita el USART; cuando esta señal está desactivada el bus de datos del USART permanece en el estado de alta impedancia. La recepción de caracteres que lleguen por la línea de recepción y la transmisión de los caracteres previamente cargados en el buffer interno, continua normalmente aun cuando la señal $\overline{CS}$ esté desactivada. La señal $\overline{CS}$ esta activa cuando se encuentra en el nivel cero lógico.

La señal CLK es una entrada de reloj al USART, que se usa para generar una base de tiempo interna del mismo. A esta entrada debe aplicarse un reloj que tenga una frecuencia mayor que 30 veces la rata de transmisión o recepción. Es de hacer notar que este reloj no se usa para referenciar ninguna señal externa (bien sean entradas o salidas al USART), para las cuales existen otras entradas de reloj que se verán mas adelante.

A continuación un cuadro que resume la actividad del USART, segun el estado de las señales de control que se acaban de describir:

C/$\overline{D}$	$\overline{RD}$	$\overline{WR}$	$\overline{CS}$	OPERACION
0	0	1	0	Se carga un caracter desde el bus para su transmisión.
0	1	0	0	Coloca en el bus de datos el caracter recibido.
1	0	1	0	El USART coloca su estatus en el bus de datos.
1	1	0	0	Coloca en el bus de datos la información referente al modo de operación para el que fue programado
X	X	X	1	El bus de datos se coloca en el estado de alta impedancia.

c) Lógica de control para operación con MODEMS

El USART posee una serie de entradas y salidas, que se pueden usar para facilitar la comunicación cuando se esten usando equipos moduladores/demoduladores (MODEMS) para la transmisión de la información a distancia. Estas entradas y/o salidas, accesibles y manejables tanto por hardware como por software, pueden ser usadas para propósitos distintos a los especificados para los MODEMS; cuando se usen para los propósitos específicos de la comunicación con MODEMS, su significado y uso se ajusta a los de la señal equivalente en el MODEM. Estas señales son:

$\overline{DSR}$ (Data Set Ready, en los MODEMS)

$\overline{DTR}$ (Data Terminal Ready, en los MODEMS)

$\overline{RTS}$ (Request To Send, en los MODEMS)

$\overline{CTS}$ (Clear To Send, en los MODEMS)

d) Sección Receptora

La sección receptora es la encargada de manejar todas las actividades relacionadas con la recepción de los caracteres, para lo cual, cuenta con dos buffer y una lógica para control de la recepción. El buffer receptor está constituido por un shift

register en el cual se ensamblan los bits recibidos por la entrada RxD del receptor, hasta "armar" un caracter completo; una vez que se tiene un caracter completo, el mismo, se coloca en un segundo buffer con la finalidad de dejar disponible el buffer receptor para recibir eventualmente otro caracter.

Las entradas o salidas asociadas con este bloque funcional son:

RxD (entrada serial)

RxRDY (caracter completo)

$\overline{\text{RxC}}$ (reloj del receptor)

SYNDET/BD (detección de sincronismo o break)

La entrada RxD, constituye la entrada serial del USART; la señal serializada proveniente de la línea de transmisión deberá conectarse a esta entrada de la sección receptora.

El pin llamado RxRDY, es una salida de la sección receptora que tiene por finalidad, avisar al dispositivo al que esté conectado el USART (un microprocesador por ejemplo), que el USART ha ensamblado un caracter completo y que el mismo puede ser leído. Si se desea aprovechar el sistema de interrupción del microprocesador al que esté conectado el USART, la entrada para interrupción del microprocesador, puede ser conectada a esta salida del USART, de manera de generar una interrupción al momento de tener un caracter completo ensamblado.

En el caso de no querer usar el sistema de interrupción del microprocesador, la salida RxRDY puede ser leída, en una operación de lectura de estatus, bajo control de programa a través del bus de datos, con el fin de determinar en un momento dado la existencia de un caracter recibido. En ambos casos, la señal RxRDY es automáticamente desactivada cuando se realiza una operación de lectura del caracter, por parte del microprocesador.

La entrada $\overline{\text{RxC}}$ (Receiver Clock), es provista para la aplicación de la señal de reloj al receptor, la cual será la que determina la rata a la cual los caracteres serán recibidos. En el caso de operación sincrónica, la frecuencia del reloj que se aplique a esta entrada debe ser igual a la rata de recepción requerida (300 bps, requeriran un reloj de 300Hz); en el caso de operación asincrónica, la frecuencia del reloj aplicado al receptor, puede ser un multiplo de la rata de recepción deseada; este multiplo, es determinado por un par de campos en la instrucción de modo, llamado factor de rata de baud, que se verá mas adelante.

El pin denominado SYNDET/BD puede ser usado como entrada o salida y su función depende del modo en que esté operando el USART.

En el modo sincrónico, SYNDET/BD, se puede usar como entrada o salida; si la detección del o los caracteres de sincronismo (SYNC), va a ser realizada por el USART, SYNDET/BD es una salida, usada para indicar que el receptor ha ensamblado el o los caracteres de sincronismo especificados, encontrandose el receptor en sincronismo con el transmisor. La señal SYNDET/BD, es automáticamente desactivada cuando se hace una lectura de estatus del USART.

Si la detección del o los caracteres de sincronismo va a ser realizada por un elemento externo al USART (algunos MODEMS realizan ellos esta labor), SYNDET/BD, es una entrada al USART y le indica al mismo que ya el receptor está en sincronismo con el transmisor y que puede comenzar a ensamblar los bits que lleguen por la línea receptora.

En el modo asincrónico, SYNDET/BD, es una salida y es usada para indicar que el USART ha detectado una condición de BREAK(línea receptora en la condición de espacio por un tiempo equivalente al de recepción de dos o mas caracteres consecutivos) en la línea receptora.

e) Sección Transmisora

La sección transmisora del USART es la encargada de aceptar los datos en paralelo desde el microprocesador, para serializarlos e insertar los bits y caracteres requeridos segun la tecnica de comunicación empleada, manejando todas las actividades asociadas con la transmisión serial del caracter. El transmisor posee dos buffers en los cuales se cargan los caracteres a transmitir; uno de estos buffers, llamado buffer transmisor, es el encargado de serializar el caracter, sacando cada bit a transmitir por la línea TxD. Las entradas y/o salidas asociadas con este bloque funcional son:

T_xD (salida serial)

T_xRDY (buffer del transmisor libre)

T_xE (transmisor desocupado)

$\overline{T_xC}$ (reloj del transmisor)

TxD, es la salida del transmisor del USART, por esta línea, el envía uno a uno los bits que componen el caracter que

se encuentra en el buffer transmisor. El tiempo que se mantendra cada bit en la línea lo determina la frecuencia del reloj del transmisor, como se verá mas adelante.

TxRDY, es una salida de la sección transmisora; se usa para indicar que el USART está en capacidad de recibir un caracter desde el bus de datos, para su transmisión. Esta salida puede ser chequeda por el microprocesador, mediante una operación de lectura de estatus, antes de cargar un caracter en el buffer del transmisor, o puede ser usada en conjunto con la lógica de interrupción del microprocesador, para controlar y aprovechar al máximo el tiempo del USART en la transmisión de caracteres.

TxE, es una salida del la sección transmisora usada para indicar que el transmisor no tiene ningun caracter por transmitir. Esta salida difiere de la anterior en el hecho de que cuando está activa significa que el transmisor no tiene ningún caracter por transmitir (ambos buffers disponibles) a diferencia de la otra (TxRDY), que solo se refiere a uno de los buffer, sin importar el estatus del buffer transmisor.

$\overline{\text{TxC}}$, es la entrada de reloj del transmisor, su frecuencia determina la rata de transmisión de los caracteres. En el modo de operación sincrónico, la frecuencia del reloj debe ser igual a la rata de transmisión; en el caso de operación asincrónica, la frecuencia del reloj es un multiplo de la rata de transmisión requerida; dicho multiplo se define en la instrucción de modo, cuando se esté programando la forma como va a operar el USART, la cual se verá a continuación.

PROGRAMACION DEL USART 8251A

Una vez descritos brevemente los diferentes bloques funcionales que constituyen el USART 8251A, se procederá a la descripción de la programación del mismo, para definir la forma como va a operar.

Se ha mencionado anteriormente que el USART requiere de dos tipos de instrucciones para definir la forma como va a operar: la instrucción de modo y la instrucción de comando; la figura 6-5 muestra la secuencia general en la que deben enviarse estas instrucciones para controlar la operación del USART.

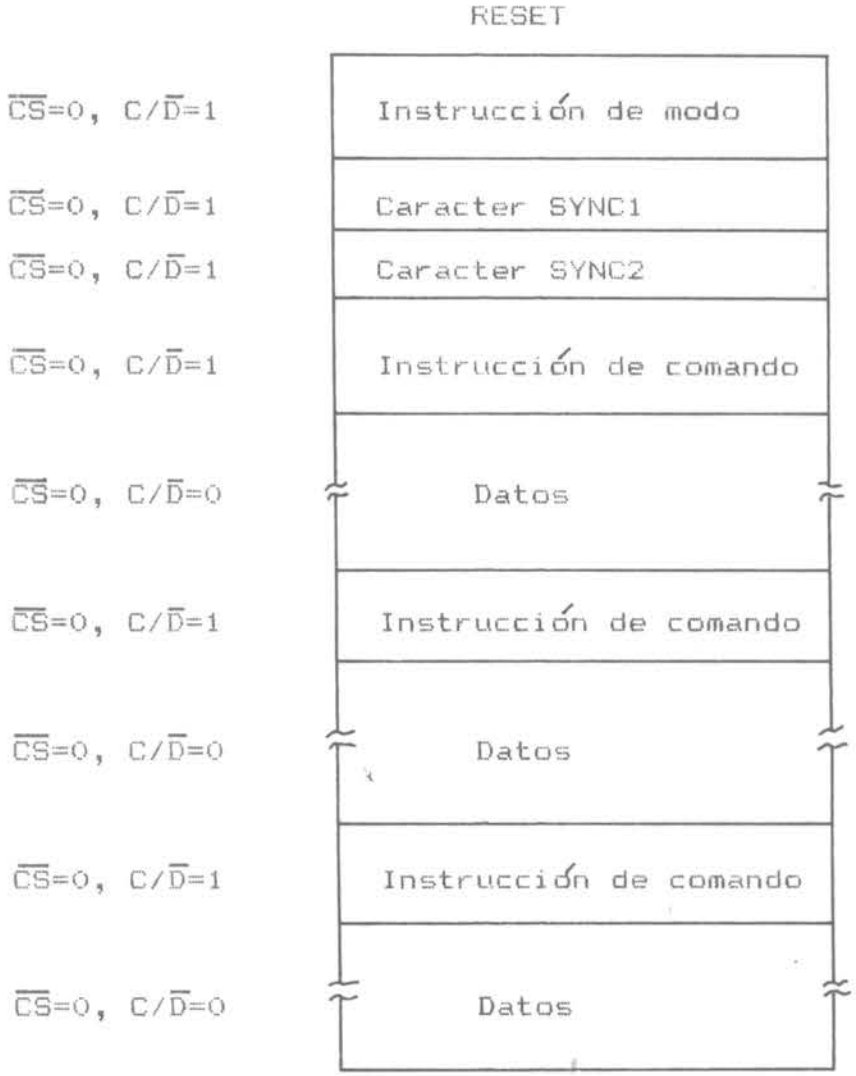


Figura 6-5. Secuencia de programación del 8251A

A continuación se describirá la instrucción de modo para operación asincrónica y para operación sincrónica; despues se describirá la instrución de comando.

a) Instrucción de modo para operación Asincrónica

El formato de la instrucción de modo para programar el USART a operar en la modalidad asincrónica se muestra a continuación:

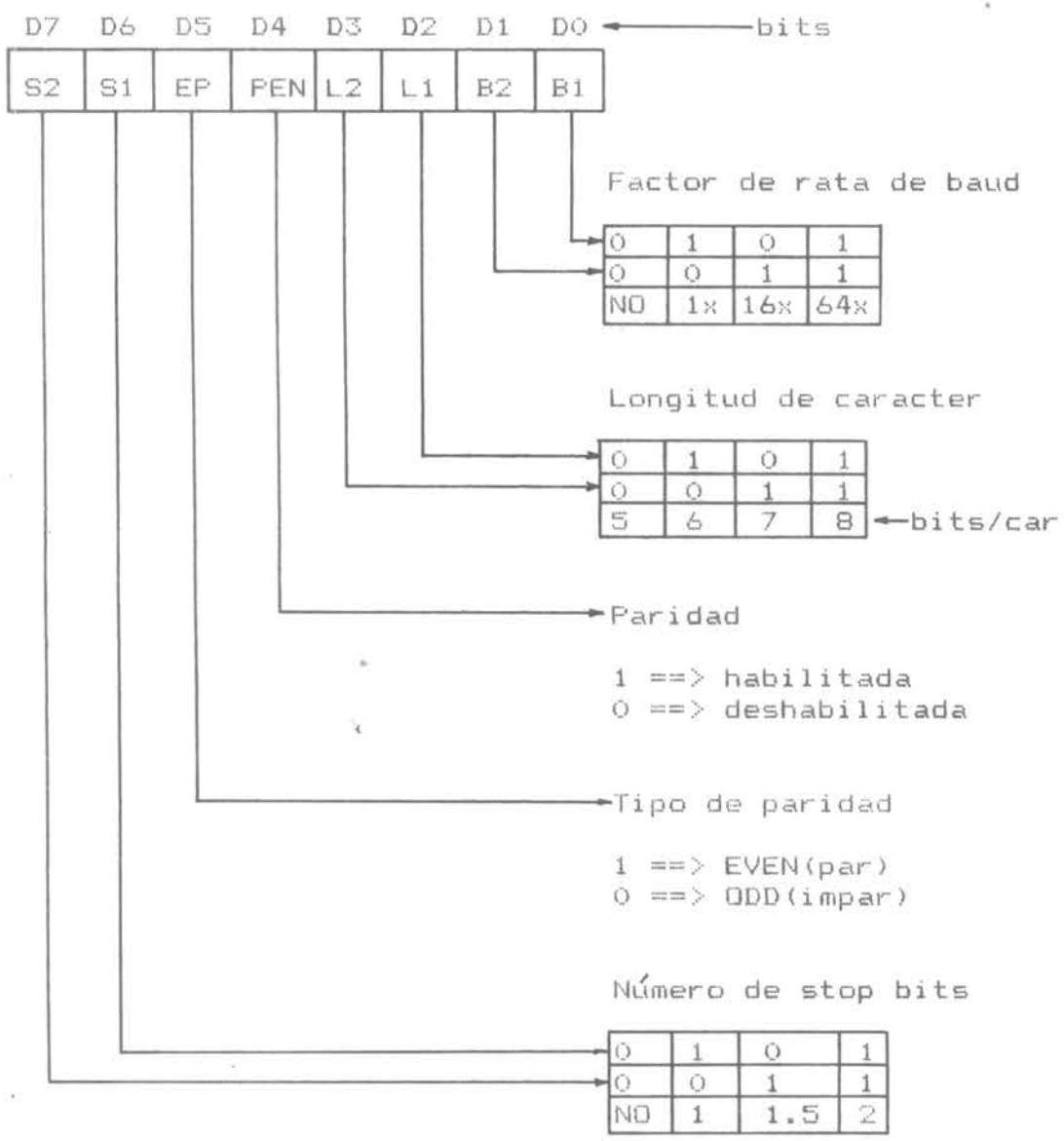


Figura 6-6. Instrucción de modo en operación asincrónica

Los dos bits menos significativos de la palabra de modo, para operación asincrónica (D0, D1) llamados B1 y B2 respectivamente, definen el factor de rata de baud, el cual como ya se dijo anteriormente, viene a constituir el elemento que en definitiva definirá la frecuencia del reloj que debe ser aplicado a las entradas de reloj $\overline{RxC}$ y $\overline{TxC}$, para transmitir y/o recibir a una rata de baud determinada. Si por ejemplo se desea transmitir y recibir caracteres a una rata de 2400 baudios y los bits que definen en la palabra de modo el factor de rata de baud son B1=0 y B2=1, lo cual nos dá un factor de 16, la frecuencia del reloj que debe ser aplicado a las entradas de reloj $\overline{RxC}$ y $\overline{TxC}$ del USART

deberá ser de 16x2400 hertz = 38400 hertz. Es de hacer notar que si bien es cierto que la rata de baud del receptor y del transmisor pueden ser distintas, el factor de rata de baud es único, consideración esta que debe ser tomada en cuenta cuando se calcule la frecuencia del reloj que debe ser aplicado al transmisor y la frecuencia del reloj que se deberá aplicar al receptor. Por ejemplo: se desea transmitir a una rata de 1200 baud y recibir a una rata de 2400 baud, con un factor de rata de baud de 64 (B1=1, B2=1); la frecuencia de cada reloj se determina como sigue:

$$\overline{RxC} = 64x2400 \text{ Hertz} = 153600 \text{ Hertz}$$

$$\overline{TxC} = 64x1200 \text{ Hertz} = 76830 \text{ Hertz}$$

Observese tambien que una misma frecuencia de reloj aplicada en $\overline{RxC}$ o en $\overline{TxC}$, puede significar distintas ratas de baudios, dependiendo del factor de rata de baud; Una señal de 38400 Hertz puede significar una rata de baud de 2400, si el factor de rata de baud es 16; esa misma frecuencia de reloj puede tambien significar una rata de baud 38400, si el factor de rata de baud es 1 y puede tambien significar una rata de baud de 600, si el factor de rata de baud es 64

Los bits 3 y 4 de la instrucción de modo (D2,D3), llamados L1 y L2 respectivamente, definen la longitud del caracter o sea, el número de bits por caracter; es posible seleccionar cuatro posibles longitudes de caracter: 5 bits, 6 bits, 7 bits y 8 bits. Cuando se escogen las longitudes de caracter 5,6 y 7 los bits no usados aparecen en el bus como ceros.

El bit 5 de la instrucción de comando (D4), llamado PEN, se usa para indicar al USART si se desea que se le agregue o chequee la paridad en el caracter transmitido o recibido. Un valor de uno en este bit, habilita la generación/chequeo de la paridad y un cero lo deshabilita.

El bit 6 de la instrucción de modo (D5), llamado EP, define el tipo de paridad que se va a generar o chequear; si la generación/chequeo de paridad está habilitada (PEN=1) y EP=1, el USART genera/chequea paridad par, en el caracter transmitido/recibido; si EP=0, la paridad generada/chequeada por el USART es impar.

Los bits 7 y 8 de la instrucción de modo (D6, D7), llamados S1 y S2 respectivamente, en el caso de transmisión, determinan el número de stops bits que se agregarán al final de cada caracter transmitido. Hay tres combinaciones validas que definen intervalos de stops de: 1 tiempo de bit (S1=1, S2=0), 1.5 tiempos de bit (S1=0, S2=1) y 2 tiempos de bit (S1=1, S2=1); la combinación S1=0 y S2=0 no es valida.

b) Instrucción de modo para operación sincrónica

El formato de la instrucción de modo para la operación sincrónica del USART, se muestra a continuación:

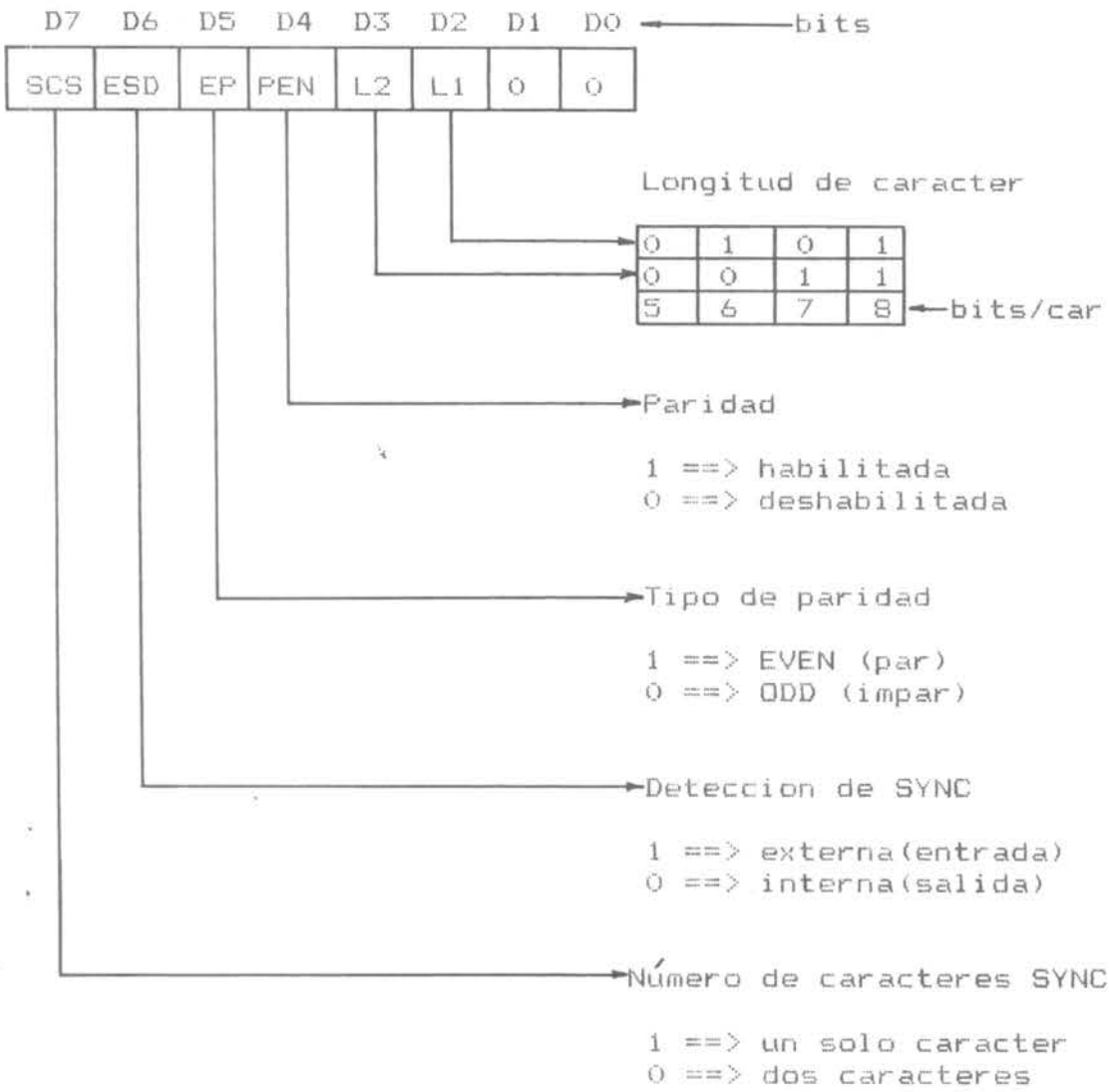


Figura 6-7. Instrucción de modo en operación sincrónica

Los bits 1 y 2 de la instrucción de modo para operación sincrónica, no tienen ningún uso en operación sincrónica y deben ser colocados en cero; realmente la colocación de estos dos bits en cero, son los que determinan o seleccionan el modo de operación sincrónica.

Los bits 3, 4, 5 y 6 de la instrucción de modo (D2, D3, D4, D5), cumplen la misma función que en la operación asincrónica, y se describieron ampliamente en líneas anteriores.

El bit 7 de la instrucción de modo para operación sincrónica (D6), llamado ESD (External Sync Detect), se usa para indicarle al USART si la detección del o los caracteres de sincronismo, SYNC, va a ser llevada a cabo por un dispositivo externo (ESD=1) o va a ser realizada por el USART (ESD=0); en el primer caso el pin SYNCDET/BD queda programado como una entrada y en el segundo caso como una salida.

El bit 8 de la instrucción de modo para operación sincrónica (D7), llamado SCS (Single Character Sync), se usa para definir el número de caracteres SYNC que se requerirán para establecer el sincronismo del receptor con el transmisor; hay dos posibles alternativas: usar un solo caracter SYNC (SCS=1) o usar dos caracteres SYNC (SCS=0). Cuando la detección del caracter SYNC es programada para ser llevada a cabo por un dispositivo externo, el valor de SCS no afecta la recepción pues el USART no interviene en el proceso de detección de los caracteres SYNC; la transmisión si se verá afectada ya que el USART deberá insertar, en forma automática, uno o dos caracteres SYNC, dependiendo del valor de SCS. Cabe destacar que cuando se usen dos caracteres SYNC, los mismos no tienen por que ser iguales; el usuario podrá escoger cualquier par de caracteres como sus caracteres SYNC, con las consideraciones que se explicaron en líneas anteriores.

c) Instrucción de comando

Despues de programar el USART con la instrucción de modo, que le define la forma como va a operar, el mismo se encuentra ya en capacidad de ser usado para la comunicación de datos. Las instrucciones de comando son comunes a cualquiera de los modos de operación del USART; se usan para controlar, en el modo de operación seleccionado, el comportamiento real del mismo.

La instrucción de comando es aceptada por el USART en cualquier momento despues de programado el modo de operación, a diferencia de la instrucción de modo, que solo es aceptada como tal por el USART, despues de un RESET.

A continuación el formato de la instrucción de comando:

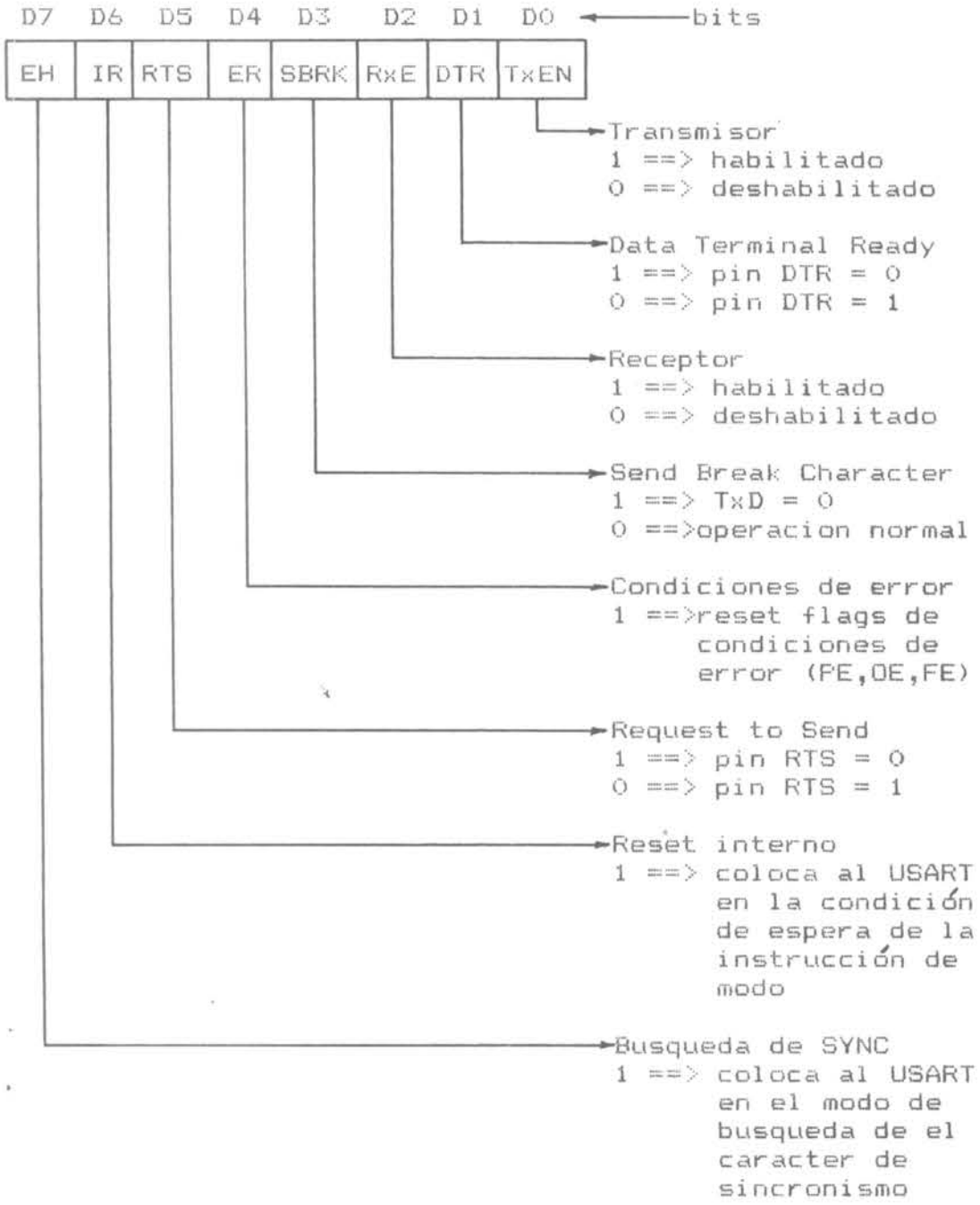


Figura 6-8. Formato de la instrucción de comando

Los bits 1 y 3 de la instrucción de comando (D0,D2), llamados TxEN y RxE respectivamente, permiten habilitar y/o deshabilitar el transmisor y el receptor. El USART, no iniciará la transmisión o recepción de ningún caracter si el transmisor y/o el receptor se encuentran deshabilitados.

El bit 2 de la instrucción de comando (D1), llamado

DTR, es usado para controlar por software, la salida $\overline{DTR}$ del USART; un nivel alto en este bit forza la salida $\overline{DTR}$ a cero y viceversa. En la operación con MODEMS, esta señal debe estar activa (cero lógico), para que el MODEM inicie la comunicación.

El bit 6 de la instrucción de comando (D5), llamado RTS, permite controlar la salida $\overline{RTS}$ del USART; un nivel alto en este bit forza la salida $\overline{RTS}$ a cero. Esta señal es usada en la operación con MODEMS, para indicar al MODEM que el transmisor tiene para transmitir uno o varios caracteres; la transmisión de un caracter por parte del USART no se inicia, hasta tanto el MODEM responda a esta señal RTS, lo cual se hace con la señal CTS.

El bit 5 de la palabra de comando (D4), llamado ER, permite borrar los flags de errores del USART. Existen tres tipos de errores que el USART es capaz de detectar: errores de paridad (PE), errores de overrun (OR) y errores de framing (FE). El flag de error de paridad se activa cuando el USART detecta en el caracter recibido una paridad que no se corresponde con la que se le ha programado en la instrucción de modo; el flag de error de overrun (OE), se activa cuando el caracter previamente recibido por el USART, no es leído por el microprocesador o dispositivo que lo maneje, y se recibe otro caracter por la línea; el flag de overrun, cuando está activo, es un indicativo de que se perdió un caracter; el flag indicador de error de framing se activa cuando el USART no detecta un stop bit válido, lo cual generalmente ocurre, cuando el corrimiento en frecuencia entre el reloj del receptor y el del transmisor, es mayor que la tolerancia permitida para una longitud de caracter determinada. La ocurrencia de uno cualquiera de estos errores NO inhiben la operación del USART.

El bit 8 de la instrucción de comando (D7), llamado EH, se usa para forzar al USART a entrar en el modo de busqueda del caracter o caracteres de sincronismo; el microprocesador al que esté conectado el USART, puede decidir en cualquier momento resincronizar el receptor con el transmisor, lo cual se hace generalmente, cuando se detectan errores en los caracteres recibidos.

d) Palabra de estatus del USART

El estatus de las líneas de control del USART es posible leerlo a través del bus de datos del mismo, con un comando de lectura de estatus; casi todos los campos o bits de la palabra de estatus tienen un pin del USART asociado y el estado lógico en la palabra de estatus, refleja el estado lógico de la señal en el pin correspondiente, teniendo por supuesto el mismo significado. Adicional a estas señales de control, la palabra de estatus trae campos que reflejan el estado de los flag de errores (PE, FE, y OE), los cuales no tienen ningún pin externo asociado. A continuación el formato de la palabra de estatus:

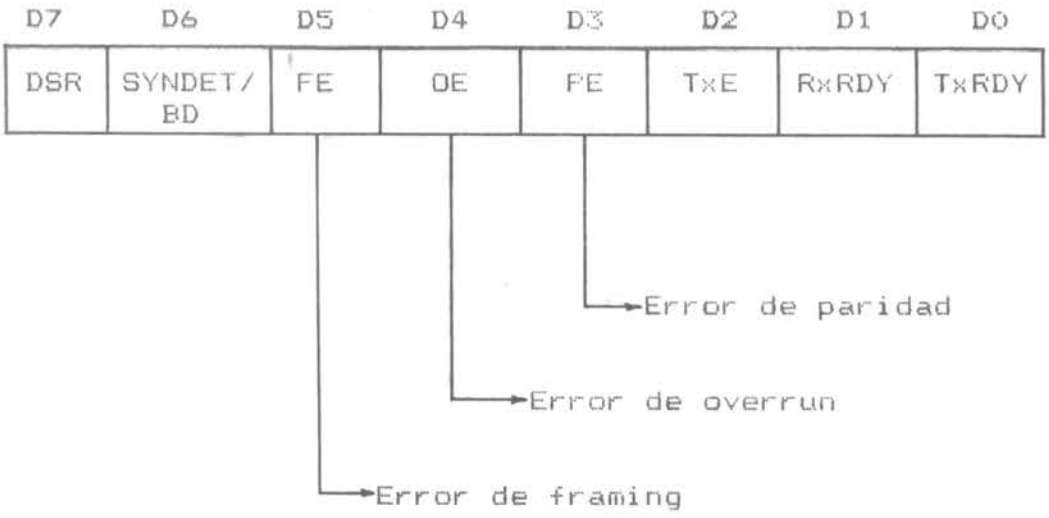


Figura 6-9. Formato de la palabra de estatus

6.1.1.3. Transmisor Receptor Universal Asincronico UART

El UART es un circuito integrado de alta escala de integración, que transmite y recibe datos asincronicamente. El receptor y el transmisor pueden operar en forma totalmente independiente, inclusive a ratas de baud distintas. El UART solo puede operar ASINCRONICAMENTE.

El UART, se asemeja al USART, cuando este último se programa para operación asincrónica; la operación del UART es contrplable y programable por una serie de entradas de control que permiten seleccionar la longitud de caracter (5,6,7 y 8 bits/caracter), chequeo/generación de paridad (par o impar), y el número de bits de stops; el factor de rata de baud, que se definió en la descripción del USART y en el cual podía tener valores de 1, 16 y 64, es constante en el caso del UART, y tiene un valor de 16; o sea, el transmisor o el receptor para operar a una rata de baud determinada, requieren de un reloj, cuya frecuencia sea de 16 veces la rata de baud deseada.

El formato de transmisión/recepción usado por el UART es el mismo que se describió en la sección 6.1.1.1, el cual corresponde al formato de transmisión serial asincrónica.

La programación y operación del UART es mucho mas sencilla que la del USART y en la generalidad de los casos se reduce a conectar las entradas de control que trae el UART, a el nivel lógico que corresponda, para adaptarlo a las necesidades

específicas de una transmisión/recepción particular; en este aspecto se diferencia apreciablemente del USART, en el cual la programación se hace a través del bus, y se puede cambiar dinámicamente en forma relativamente sencilla.

La figura 6-10 muestra un diagrama de bloque simplificado de el UART (especificamente el UART AY-5-1013), en el cual se pueden apreciar las dos secciones principales que corresponden al transmisor y al receptor y cuya operación se describirá a continuación:

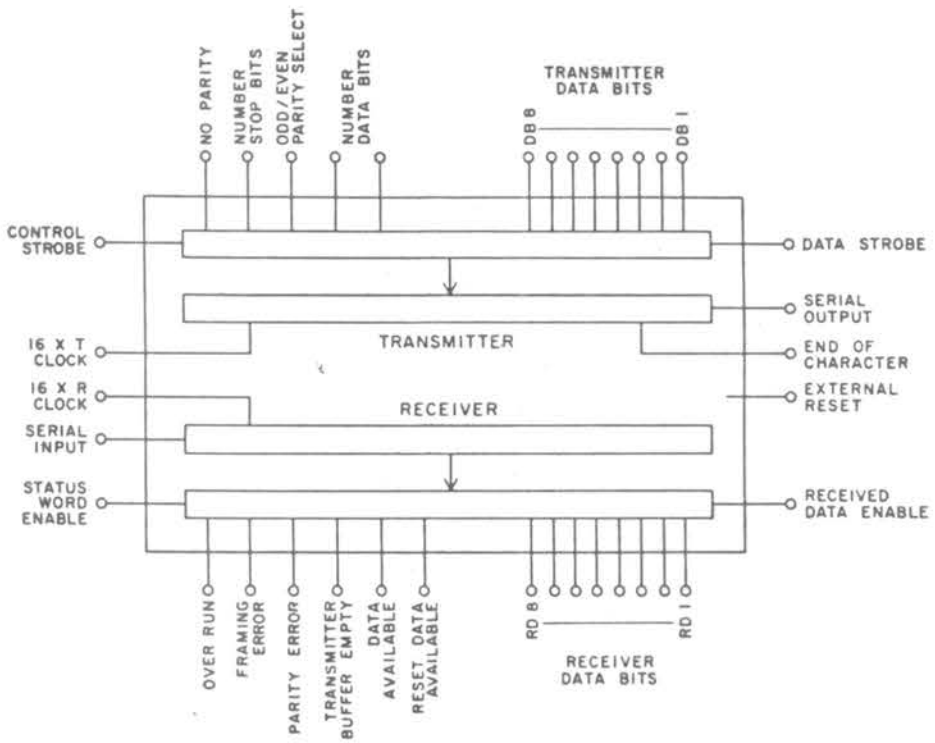


Figura 6-10. Diagrama de bloque simplificado del UART AY-5-1013

a) Sección transmisora

El transmisor posee dos registros; el registro holding, en el cual se carga inicialmente el caracter a transmitir y el registro serializador o shift register transmisor, al cual se traslada el caracter almacenado en el registro holding para serializarlo. El registro holding puede ser cargado con un caracter cuando la señal TBMT (Transmitter Buffer Empty), esté en un nivel alto (uno lógico), lo cual significa que el registro está libre; la carga de un caracter en el registro holding se hace aplicando un pulso de corta duración en la entrada $\overline{DS}$; esta

entrada $\overline{DS}$, normalmente debe permanecer en uno lógico y ser llevada a cero al momento de carga del caracter en el registro holding; la carga del caracter en registro holding se hace efectiva en el flanco de subida de la señal $\overline{DS}$. Si el registro transmisor está disponible (no está serializando ningún caracter), el dato cargado en el registro holding es automáticamente transferido a el, para su serialización; en caso de estar serializando un caracter, el caracter cargado en el registro holding permanece en éste hasta que el registro serializador esté disponible; en ambos casos, la transferencia del caracter del registro holding al serializador es automática y en ella no interviene ninguna señal de control externa; sin embargo, el UART trae una señal externa para indicar si el registro serializador está desocupado (salida EOC).

b) Sección receptora:

El receptor del UART es activado por una transición de la condición de MARK a la condición de SPACE en la línea de recepción SI. Cuando una transición de este tipo es detectada, el receptor asume que está recibiendo un START bit; este start bit es considerado válido si para el momento en que la lógica de recepción hace el muestreo de la línea, lo cual ocurre aproximadamente en el punto medio del start bit, la condición de SPACE se mantiene en la línea de recepción, con lo cual, se asume la recepción de un start bit válido, procediendo el receptor a muestrear la línea a intervalos de 16 pulsos del reloj del receptor, comenzando a partir del momento en que se dió como válido el start bit; esto se hace con el fin de tomar el valor o estado de cada bit del caracter, aproximadamente en el punto medio del tiempo de cada bit, lo que permite mayor tolerancia en las frecuencias del receptor y el transmisor y garantiza un valor estable del bit correspondiente. El caracter se ensambla en el registro receptor (shift register receptor), de acuerdo con las señales de control que se programaron (número de bits/caracter, tipo de paridad y número de stops bits). En el UART, y específicamente en la sección receptora, se encuentran presentes los flags para indicar la detección de errores de paridad, framing y overrun; estos flags estan disponibles externamente en las líneas PE, FE y OE del UART. Despues que el STOP bit es recibido, el contenido del registro receptor es transferido, en paralelo, al registro holding del receptor, indicando a través de la salida de control DAV (Data AVailable), que se ha ensamblado un caracter completo que esta esperando para ser tomado por el microprocesador. Cuando el microprocesador lea el caracter que se encuentra en el registro holding de la sección receptora, deberá "borrar" la señal DAV para evitar la ocurrencia de un error de overrun, en virtud de que en el UART esta señal (DAV) no se borra automáticamente cuando se lee el caracter, como ocurre en el USART 8251A; en el UART el "borrado" de la señal DAV se hace a través de la señal de control $\overline{RDAV}$ (Reset Data AVailable); si la línea DAV no se borra antes de que otro caracter completo se haya recibido, el flag de overrun (OE) se activará, indicando la posible perdida de un caracter.

La figura 6-11 es un diagrama de pines del UART AY-5-1013 y la tabla siguiente es una descripción funcional de cada uno de estos pines.

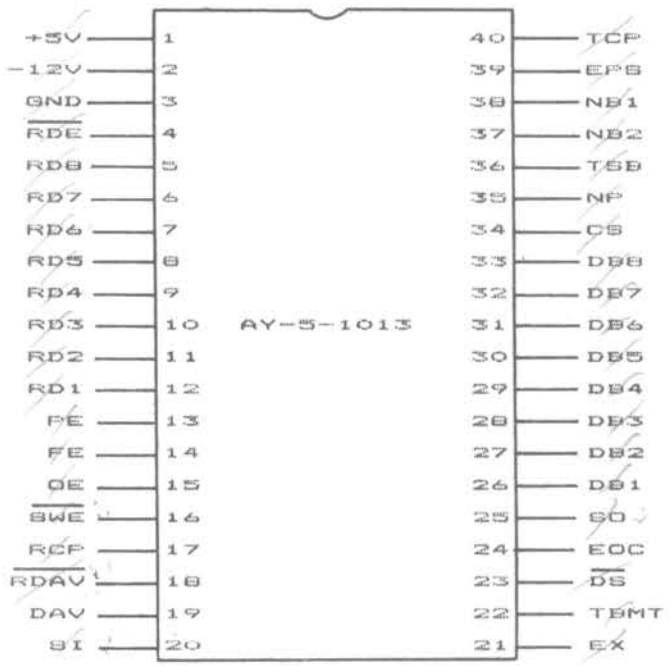


Figura 6-11. Diagrama de pines del UART AY-5-1013

PIN	NOMBRE	FUNCION
1	+5V	
2	-12V	
3	GND	
4	$\overline{\text{RDE}}$	$\overline{\text{RDE}}$=0 ==> Bus habilitado $\overline{\text{RDE}}$=1 ==> Bus receptor en tri state
5-12	RD8-RD1	Bus del receptor RD1 = bit menos significativo
13	PE	PE=1 ==> error de paridad(tri-state)
14	FE	FE=1 ==> error de framing(tri-state)
15	OE	OE=1 ==> error de overrun(tri-state)
16	$\overline{\text{SWE}}$	$\overline{\text{SWE}}$=1 ==>tri state(PE,FE,OE,DAV,TBMT)

		$\overline{SWE}$=0 ==>normal (PE,FE,OE,DAV,TBMT)
17	RCP	reloj receptor (F_{reloj}=16xBauds)
18	$\overline{RDAV}$	$\overline{RDAV}$=0 forza linea DAV=0
19	DAV	DAV=1 ==>un caracter completo recibido
20	SI	entrada serial del receptor
21	EX	EX=1 ==> RESET (SO=TBMT=EOC=1, DAV=PE=FE=OE=0) EX=0 ==> operacion normal
23	$\overline{DS}$	$\overline{DS}$=0 ==> Carga registro holding del transmisor
24	EOC	EOC=1 ==> un caracter completo ha sido transmitido
25	SO	salida serial del transmisor
26 a 33	DB1-DB8	bus de datos del transmisor
34	CS	CS=1 ==> carga los bits de control(NP, EPS,NB1,NB2 y TSB)
35	NP	NP=1 ==> eliminar paridad NP=0 ==> agregar paridad programada
36	TSB	TSB=1 ==> agregar 2 stops bits TSB=0 ==> agregar 1 stop bit
37-38	NB2,NB1	0 0 1 1 NB2 0 1 0 1 NB1 5 6 7 8 bits/caracter
39	EPS	EPS=0 ==> paridad impar (odd) EPS=1 ==> paridad par (even)
40	TCP	reloj transmisor(F_{reloj}=16xBauds)

6.1.2. INTERFASES PARALELAS

Hasta el momento se han descrito dos interfases que permiten realizar la conexión de un computador a dispositivos que envian y reciben información en forma serial; sin embargo, en muchas oportunidades los dispositivos no manejan la información en forma serial sino en forma paralela. Se han diseñado para los microprocesadores una serie de componentes de alta escala de

integración que permiten satisfacer los requerimientos de conexión de dispositivos en forma paralela; ejemplos de estas interfases son los PIA (Programmable Interface Adapter) desarrollados por MOTOROLA y los PPI (Programmable Peripheral Interfase) desarrollados por INTEL. Existen otra serie de interfases paralelas programables, sin embargo, en este trabajo solo se tratará con el PPI desarrollado por INTEL, identificado como 8255A.

6.1.2.1. Interfase programable para periféricos PPI 8255A

El PPI 8255A, es una interfase programable, que permite la conexión al bus de un microprocesador de dispositivos periféricos tales como: impresoras, floppy disk, conversores A/D y D/A, teclados, displays, etc.; en una forma relativamente sencilla y practicamente sin hardware externo adicional, que puede adaptarse a muchas aplicaciones particulares por medio de comandos de software, enviados por el microprocesador a través del bus de datos.

El 8255A cuenta con 24 líneas de entrada/salida, cuya organización es determinada por el usuario, mediante los comandos apropiados que se describirán posteriormente.

La figura 6-12 muestra el diagrama de bloques funcional del 8255A y el diagrama de pines del mismo.

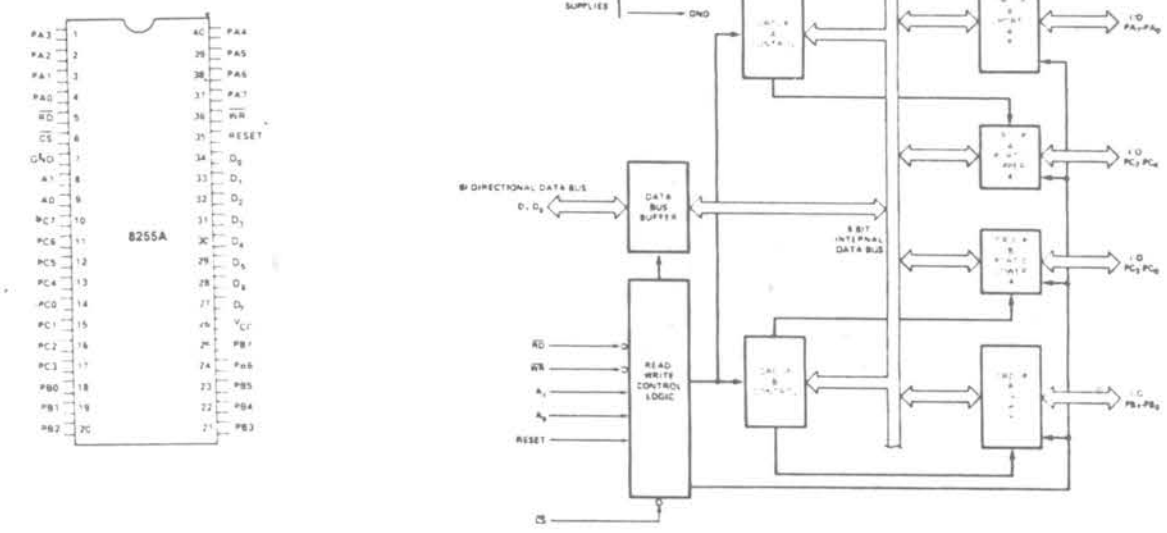


Figura 6-12. PPI 8255A. A) Diagrama de pines. B) diagrama de bloques

El 8255A se puede considerar constituido por 6 bloques funcionales, ellos son:

a) Buffer del bus de datos

b) Lógica de control de lectura/escritura

c) Lógica de control de líneas de E/S del grupo A

d) Lógica de control de líneas de E/S del grupo B

e) Líneas de E/S del grupo A
   Puerto A, 8 bits (PA7-PA0)
   Puerto C, 4 bits (PC7-PC4)

f) Líneas de E/S del grupo B
   Puerto B, 8 bits (PB7-PB0)
   Puerto C, 4 bits (PC3-PC0)

Según sea programado, el 8255A puede proveer: 3 puertos de E/S independientes, de 8 bits cada uno; 2 puertos de E/S, con sus correspondientes líneas para "handshaking" (8 líneas para dato y 4 líneas handshaking); o un puerto de E/S tipo bus, de 8 bits (bidireccional), con 5 líneas para handshaking y otro puerto de 8 bits con tres líneas para handshaking. Existen una serie de combinaciones de programación del 8255A que permiten configurarlo en una gran variedad de formas. A continuación, una breve descripción de los diferentes bloques que constituyen el PPI 8255A y de las líneas que entran y salen de estos bloque funcionales.

a) Buffer y bus de datos

Toda transferencia de información (datos o comandos de control) entre el microprocesador y el 8255A se hace a traves de las líneas del bus de datos que trae el 8255A. El bus del 8255A es de tres estados, bidireccional y buferizado; el sentido y el estado del bus de datos y sus buffers, es controlado por la lógica de control de lectura/escritura que se verá a continuación.

b) Lógica de control de lectura/escritura

La función de esta sección del 8255A es manejar y controlar todas las transferencias, ya sean datos, comandos de control o información de estatus. A esta sección llegan señales de control provenientes del bus de control del microprocesador

($\overline{RW}$ y $\overline{WR}$) y señales provenientes del bus de direcciones ($\overline{CS}$, A0, A1), a partir de las cuales se generan las señales de control apropiadas para dirigir la operación de los demas bloques funcionales. Las señales externas que entran a esta sección son:

$\overline{RD}$ (READ)

$\overline{WR}$ (WRITE)

A0, A1

$\overline{CS}$ (Chip Select)

RESET

Las líneas $\overline{RD}$ Y $\overline{WR}$ se usan para indicarle al 8255A si el microprocesador va a leer o escribir sobre el alguna información, que puede ser un dato, un comando de control o información de estatus.

Las líneas A0, y A1 se usan para seleccionar el puerto desde donde se leeran o escribirán los datos; generalmente, estas dos líneas se conectàn a los dos bits menos significativos del bus de direcciones del microprocesador.

La línea $\overline{CS}$ permite habilitar o deshabilitar la comunicación entre el 8255A y el microprocesador ($\overline{CS}$=1 ==> bus en tri state). La tabla siguiente resume todas las operaciones realizables entre el 8255A y el microprocesador.

A1	A1	$\overline{RD}$	$\overline{WR}$	$\overline{CS}$	OPERACION
0	0	0	1	0	Lectura del puerto A(BUS←—puerto A)
0	1	0	1	0	Lectura del puerto B(BUS←—puerto B)
1	0	0	1	0	Lectura del puerto C(BUS←—puerto C)
0	0	1	0	0	Escritura sobre el puerto A (puerto A←—BUS)
0	1	1	0	0	Escritura sobre el puerto B (puerto B←—BUS)
1	0	1	0	0	Escritura sobre el puerto C (puerto C←—BUS)
1	1	1	0	0	Información de control para el 8255A (BUS = control para el 8255A)

c) y d) Control de lineas de E/S de los grupos A y B

Las líneas de entrada/salida del 8255A, están agrupadas desde el punto de vista de su control, en dos grupos, llamados grupo A y grupo B. El grupo A está formado por los 8 bits del puerto A (PA7-PA0) y los 4 bits mas significativos del puerto C PC7-PC4); el grupo B, formado por los 8 bits del puerto B y los 4 bits menos significativos del puerto C (PC3-PC0). Cada grupo de líneas (A y B), tiene sus respectivas lógicas de control, independientes una de la otra.

Cada sección de control (control del grupo A y control del grupo B), cumplen la misma función para sus correspondientes líneas de E/S. Cada uno de estos controles de grupo, acepta los comandos de control generados por la lógica de control de lectura/escritura y en conjunto con las palabras de control enviadas a través del bus interno del 8255A, definen la operación y la configuración de las distintas líneas de E/S de los respectivos grupos. A partir de las palabras de control almacenadas en la respectiva sección de control, se generan los comandos apropiados a los grupos, de manera que operen en la forma que el usuario lo desee.

e) y f) Líneas de entrada/salida de los grupos A y B

Las líneas de entrada/salida del 8255A, como ya se mencionó anteriormente, han sido agrupadas para su control, en dos grupos llamados grupo A y grupo B. Cada uno de estos grupos tiene 12 líneas de entrada/salida, cuya utilización se define mediante los comandos de control apropiados. Sin embargo, desde el punto de vista operacional, es posible considerar que el 8255A dispone de tres puertos de 8 bits cada uno (puertos A, B y C), que pueden ser configurados en una gran variedad de formas, bajo control de software, para adaptarlos a los requerimientos del usuario. Todos los puertos se pueden programar para que operen bien sea como entradas o como salidas, y en especial el puerto A puede ser programado para operar en forma bidireccional (tipo bus); todas las líneas de entrada/salida del 8255 tienen latchs tanto para la entrada como para la salida. El puerto C tiene la particularidad de poder ser tratado como dos puertos de 4 bits cada uno (un puerto constituido por los 4 bits mas significativos y el otro por los cuatro menos significativos).

Descripción operacional y programación del 8255A

El 8255A puede operar basicamente en tres modos,

seleccionables por software, ellos son:

Modo 0 (Entrada/salida basica, sin líneas de control)

Modo 1 (Entrada/salida con líneas para handshaking)

Modo 2 (Entrada/salida tipo bus, bidireccional)

Los puertos A y B del 8255A, pueden ser programados en forma independiente para que operen en un determinado modo, esto es, el puerto A se puede programar para operar por ejemplo en el modo 2 y el puerto B para operar en el modo 0. El puerto C, no puede ser programado en forma independiente, ya que su modo de operación lo determina los requerimientos del modo en que fueron programados los puertos A y B. La figura 6-13 muestra un esquema de los diferentes modos de operación del 8255A y su interfase con el bus del microprocesador.

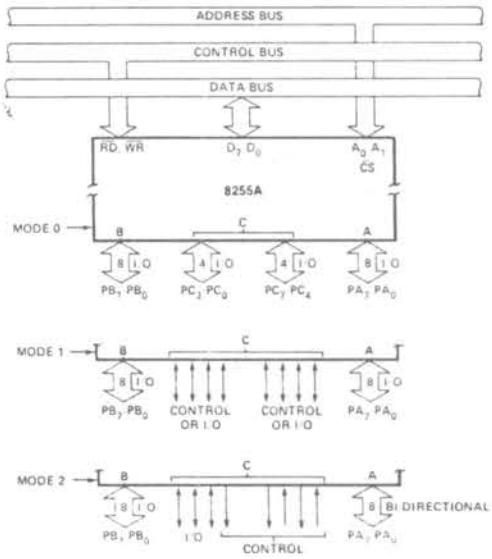


Figura 6-13. Modos de operación del 8255A y su interconexión con un microprocesador

El formato de la instrucción para la programación del 8255A, se muestra en la figura 6-14. Existen, como se puede apreciar en dicha figura, la posibilidad de combinar y mezclar modos de operación de los distintos puertos. En la palabra de control se han reservado campos (bits) que permiten programar en forma independiente ambos grupos de líneas de E/S (grupo A y grupo B),

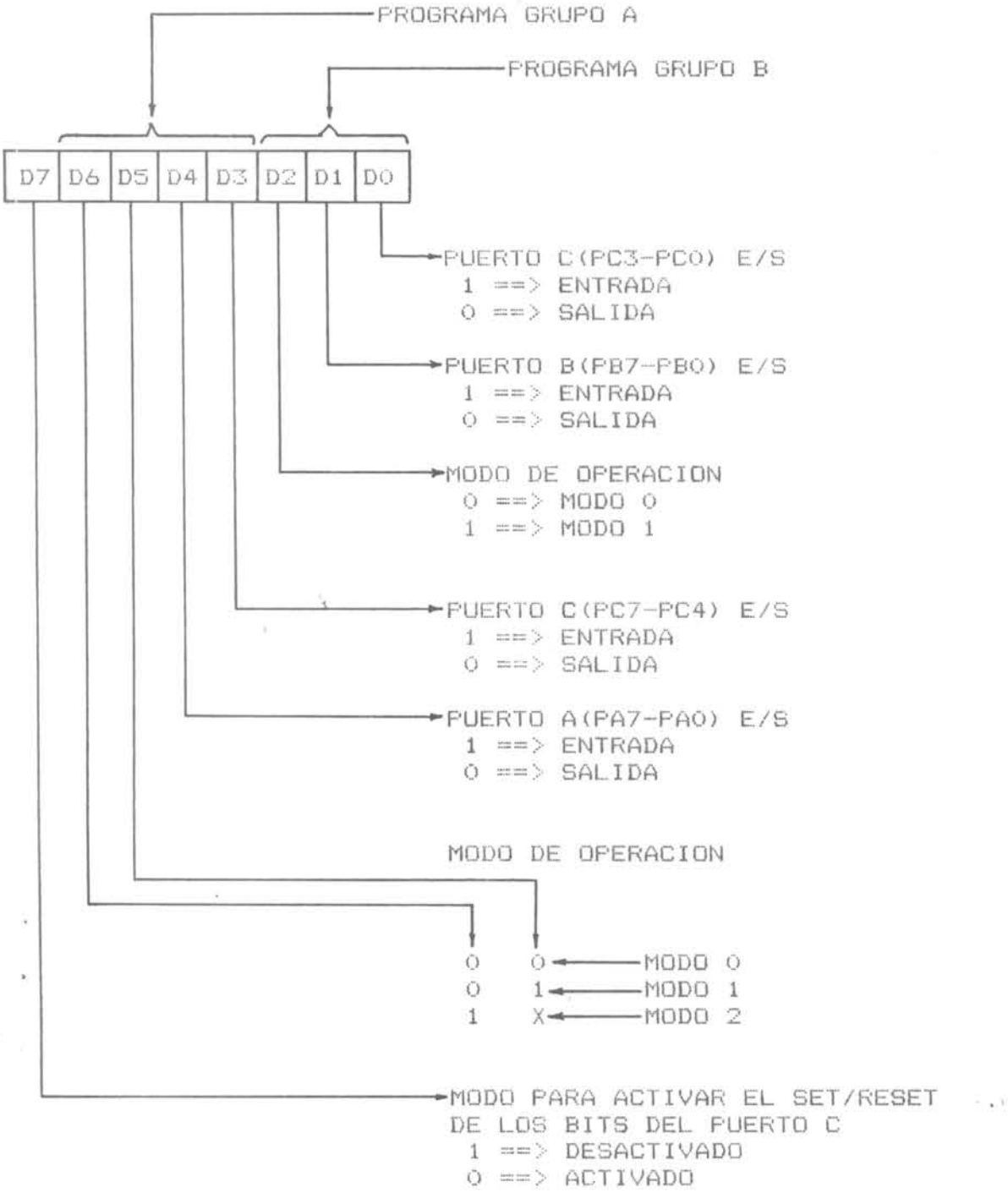


Figura 6-14. Formato de la instrucción de programación del 8255A

MODO 0: Esta modalidad de operación permite proveer al usuario de dos puertos de 8 bits cada uno y dos puertos de 4 bits

cada uno; no se preveen líneas específicas para handshaking. Los puertos pueden ser programados bien sea como entradas o como salidas. La tabla siguiente resume todas las combinaciones para operación de ambos grupos de líneas; observese que el modo de operación del puerto C, lo determina el modo de operación del grupo al que pertenece la mitad correspondiente de dicho puerto; sin embargo, el uso de cada una de las líneas del puerto (entrada o salida) si puede ser programado en forma específica para el puerto C en forma independiente de como se programen los puertos A y B.

COMANDO PARA MODO 0				GRUPO A		GRUPO B	
D6=D5=D2=0				PUERTO A	PUERTO C	PUERTO B	PUERTO C
D4	D3	D1	D0		PC7-PC4		PC3-PC0
0	0	0	0	SALIDA	SALIDA	SALIDA	SALIDA
0	0	0	1	SALIDA	SALIDA	SALIDA	ENTRADA
0	0	1	0	SALIDA	SALIDA	ENTRADA	SALIDA
0	0	1	1	SALIDA	SALIDA	ENTRADA	ENTRADA
0	1	0	0	SALIDA	ENTRADA	SALIDA	SALIDA
0	1	0	1	SALIDA	ENTRADA	SALIDA	ENTRADA
0	1	1	0	SALIDA	ENTRADA	ENTRADA	SALIDA
0	1	1	1	SALIDA	ENTRADA	ENTRADA	ENTRADA
1	0	0	0	ENTRADA	SALIDA	SALIDA	SALIDA
1	0	0	1	ENTRADA	SALIDA	SALIDA	ENTRADA
1	0	1	0	ENTRADA	SALIDA	ENTRADA	SALIDA
1	0	1	1	ENTRADA	SALIDA	ENTRADA	ENTRADA
1	1	0	0	ENTRADA	ENTRADA	SALIDA	SALIDA
1	1	0	1	ENTRADA	ENTRADA	SALIDA	ENTRADA
1	1	1	0	ENTRADA	ENTRADA	ENTRADA	SALIDA
1	1	1	1	ENTRADA	ENTRADA	ENTRADA	ENTRADA

MODO 1: Esta configuración de los puertos del 8255A, permite al usuario disponer de dos grupos de líneas de entrada/salida; cada grupo de líneas cuenta con un puerto para datos de 8 bits y un puerto de control/datos de 4 bits, con líneas específicas para handshaking. Los puertos de 8 bits pueden ser programados para operar bien sea como entradas o como salidas, en ambos casos con latchs. Los puertos de 4 bits, se usan como puertos de control y handshaking del respectivo puerto de 8 bits.

Cuando se programan los grupos de líneas de entrada/salida para operar en el modo 1, el puerto C, constituido por dos grupos de 4 bits (PC7-PC4 y PC3-PC0), se comporta como un puerto de control, con líneas que permiten realizar funciones específicas como: carga de datos en los latchs del puerto correspondiente de 8 bits, producir una señal de reconocimiento de solicitud de atención, generar una interrupción al microprocesador, etc.. La figura 6-15 muestra las diferentes líneas de control, cuando se programam los puertos para que operen en el modo 1; en la 6-15a, se muestra cuando se programan

como entradas y en la 6-15b cuando se programan como salidas.

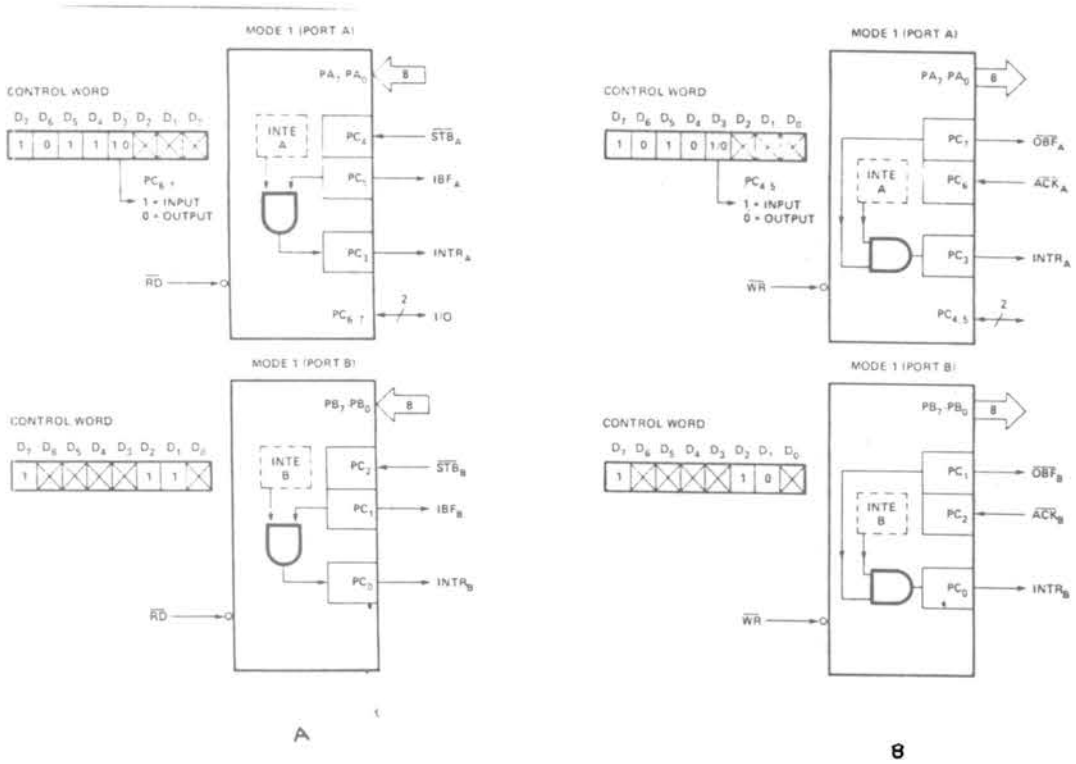


figura 6-15. Operación del 8255A en modo 1. A) Operación como entrada. B) Operación como salida

En el caso de programación para operación como puertos de entrada, las diferentes líneas de control se identifican y usan con los siguientes propósitos:

$\overline{STB}$ (A o B): Es una entrada al 8255A; un cero en esta línea, carga en los latch del puerto correspondiente, el dato presente en dicho puerto, proveniente del dispositivo que esté conectado en el puerto.

IBF(A o B): Es una salida del 8255A, que se usa para indicar que se ha cargado un dato en los latch del puerto respectivo; en general se usa para darle un reconocimiento a una petición de atención por parte del dispositivo que esté conectado en el puerto. Esta salida se activa (1 lógico), cuando STB va a cero; se desactiva, cuando se realiza una operación de lectura del puerto, por parte del microprocesador.

INTR(A o B): Esta es una salida del 8255A que puede ser usada para generar una interrupción al microprocesador. Se activa (INTR=1), cuando STB e IBF están en

uno; se desactiva cuando el microprocesador realiza una operación de lectura del puerto que interrumpe

En el caso de programación para operación como puertos de salida, las diferentes líneas de control se identifican y usan con los siguientes propósitos:

$\overline{OBF}$(A o B): Salida del 8255A, usada para indicar que el microprocesador escribió un dato en el puerto. La línea OBF se activa (OBF=0) cuando el microprocesador realiza una operación de escritura sobre el puerto y se desactiva cuando la línea $\overline{ACK}$ va a cero

$\overline{ACK}$(A o B): Entrada al 8255A, usada para indicarle al mismo, que el dato almacenado en los latch del puerto fueron tomados por el dispositivo que está conectado en ese puerto

INTR(A o B): Salida del 8255A que puede ser usada para generar una interrupción al microprocesador. Esta línea se activa (INTR=0), cuando OBF está en 1 y ACK sube a 1; se desactiva cuando el microprocesador hace una operación de escritura sobre el 8255A (WR=0)

Modo 2: El modo 2 de operación del 8255A, solo es aplicable al grupo A de líneas de entrada/salida; este modo permite que el puerto A funcione en una manera similar a la del modo 1, pero las líneas de dicho puerto operan en forma bidireccional. Este modo de operación del puerto A permite al usuario disponer de un conjunto de 8 líneas bidireccionales para datos, y 5 líneas de control, para el puerto bidireccional. La operación en el modo 2 solo es posible con el grupo A.

6.2. INTERFASES ESTANDARES

La estandarización de muchas interfases y su adopción por un número creciente de compañías fabricantes de computadores y microcomputadores, ha permitido entre otras cosas, simplificar y facilitar la expansión de los sistemas a microprocesadores.

Las interfases estandares son cada vez de uso mas generalizado por parte de los diseñadores y fabricantes de equipos, con el fin de lograr una mayor compatibilidad con otros equipos y ampliar sus mercados.

Se han adoptado interfases estandares para una serie de aplicaciones destacandose las de conexión de equipos para comunicación serial (RS232C, RS422, RS423, etc.), para buses

(S-100, IEEE488, J11, etc.) y muchas otras para distintos propósitos.

La interfase mas ampliamente usada en la actualidad para comunicación serial es la RS232C; recientemente se han implementado la RS-422 y la RS-423, con los mismos propósitos, pero para distancias mayores, las cuales en un futuro seguramente reemplazaran a la RS-232C. En este trabajo se tratará solamente la interfase RS-232C por considerarla de gran utilidad en vista de su uso tan generalizado tanto en equipos terminales como en microcomputadores.

En cuanto a las interfases para buses se describirá brevemente la llamada S-100 y la IEEE-488; la primera para conexión de expansiones a un sistema a microprocesador y la segunda para la conexión de paneles de instrumentos asociados con microprocesadores o computadores en general.

6.2.1. INTERFASE EIA RS-232C

La interfase RS-232, fue desarrollada con la finalidad de establecer una interfase que permitiese normalizar la conexión de equipos terminales de datos (microprocesadores, computadores, CRT, teletipos, etc.) con equipos de comunicación de datos (MODEMS o Data Set), cuando estos utilizen como método de transferencia de datos, LA TRANSMISION SERIAL BINARIA. Esta interfase, se aplica tanto a la transmisión serial sincrónica como a la asincrónica, bien sea en la modalidad half duplex o full duplex.

El estandard RS-232, contiene las normas que debe satisfacer la interfase en lo referente a: características eléctricas de las señales (nivel de tensión, niveles permitidos de corriente, máxima rata de cambio, etc.), características mecánicas de la interfase (tipo de conector, número de pines del conector, asignación de los pines a los circuitos de intercambio, etc.) y la definición funcional de los distintos circuitos de intercambio.

A continuación un resumen de las especificaciones eléctricas mas importantes establecidas en el estandard RS-232C.

a) El voltaje en circuito abierto en cualquier circuito de intercambio, no debe ser mayor, en magnitud, de 25 volts.

b) El cortocircuito entre dos circuitos de intercambio cualesquiera, debera ser soportado por los drivers de los circuitos de intercambio, sin sufrir daños; la corriente de

cortocircuito entre dos circuitos de intercambio, debera estar limitada a un máximo de 0.5 amperes.

c) Una señal se considerará que se encuentra en la condición MARK ("1"), cuando su voltaje sea mas negativo que -3 volts, respecto del circuito de tierra de señal. La señal se considerará se encuentra en la condición SPACE ("0"), cuando su voltaje sea mas positivo que +3 volts, respecto del circuito de tierra de señal. La región entre -3 volts y +3 volts se ha definido como región de transición y dentro de la misma el estado de la señal no está definido en RS-232 (ver figura 6-16).

d) La rata de cambio de voltaje de las señales en cualquier circuito (slew rate), no deberá ser mayor de 30 volts/microsegundo, ni menor de: 0.006 volts/microsegundo o 4% de un tiempo de bit, lo que sea menor.

e) El nivel de voltaje de salida de un circuito de intercambio, con una carga de entre 3 a 7 kiloohm será: para cero lógico (SPACE) = +5 a +15 volts; para uno lógico (MARK) = -5 a -15 volts.

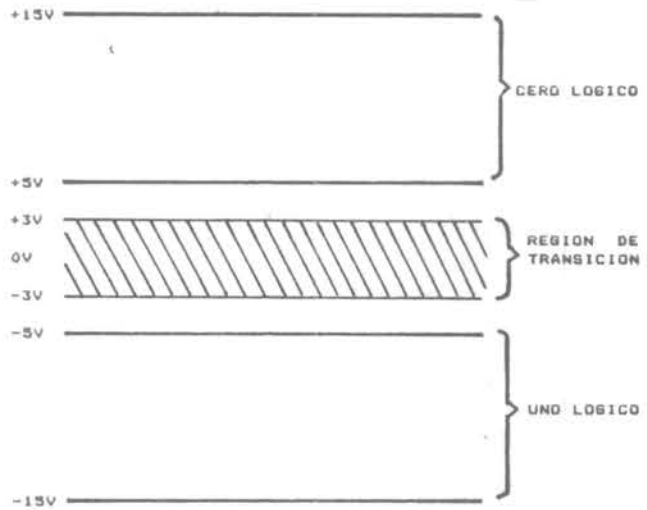


Figura 6-16. Niveles de voltaje de las señales en el estandard RS-232C

Existen 25 circuitos de intercambio definidos en este estandard; se ha adoptado el uso de un conector de 25 pines, correspondiendole a cada pin un circuito de intercambio. Es de hacer notar el hecho de que cuando se dice que un determinado equipo cumple con el estandard RS-232, eso no necesariamente significa que en el conector de salida se encuentren todos los circuitos definidos en el estandard, ya que, muchos de estos circuitos son de uso específico en una determinada modalidad de comunicación (sincrónica, asincrónica, half duplex, full duplex, etc.), y no necesitan estar presentes si dicha modalidad no se

está usando; lo mas importante sin embargo, cuando se dice que un determinado equipo cumple con el estandard RS-232, es el hecho de que eso signifique que los circuitos que se esten usando, no violan ninguna de las disposiciones establecidas en el estandard, como por ejemplo, el nivel de voltaje de las señales o el pin de salida en el conector etc.. La tabla siguiente resume todo lo referente a los nombres de los diferentes circuitos de intercambio del estandard, el simbolo con que se representa y su correspondiente pin en el conector de salida.

PIN	NOMBRE DEL CIRCUITO	SIMBOLO
1	Tierra de Protección	AA
2	Transmisor de Datos	BA
3	Receptor de Datos	BB
4	Request to Send	CA
5	Clear to Send	CB
6	Data Set Ready	CC
7	Tierra de Señal	AB
8	Receive Line Signal Detector	CF
9	Reservado para pruebas del MODEM	--
10	Reservado para pruebas del MODEM	--
11	No asignado (sin uso definido)	--
12	Secundary Receive Line Signal detec	SCF
13	Secundary Clear to Send	SCB
14	Secundary Transmit Data	SBA
15	Transmission Signal Element	DB
16	Secundary Receive Data	SBB
17	Receiver Signal Element Timing	DD
18	No asignado(sin uso definido)	--
19	Secundary Request to Send	SCA
20	Data Terminal Ready	CD
21	Signal Quality Detector	CG

22	Ring Indicator	CE
23	Data Signal Rate Selector	CH/CI
24	Transmit Signal Element Timing	DA
25	No asignado (sin uso definido)	--

A continuación una breve descripción de los circuitos mas comunmente usados en la interfase RS-232:

Circuito AB : Signal Ground (tierra de señal, pin 7)

Este circuito es usado para fijar el potencial de referencia para todos los demas circuitos, con la excepción del de tierra de protección (Protective Ground); en otras palabras, todos los niveles de voltaje en los diferentes circuitos, corresponden al voltaje diferencial entre el circuito en cuestión y este circuito AB. Este circuito se requiere en todas las modalidades de comunicación.

Circuito BA: Transmitted Data (Transmisor de Datos, pin 2)

Este circuito se usa para transferir la información o dato desde el equipo terminal de datos (DTE) al equipo de comunicación de datos (DCE), en forma serial y siguiendo el patrón de comunicación que corresponda, según el tipo de transmisión (sincrónica o asincrónica). En todo equipo que cumpla con el estandard RS-232, para que el equipo terminal de datos transmita un dato a través de este circuito, se deberán cumplir las siguientes condiciones en los circuitos que se especifican a continuación:

- Circuito CA(Request to Send) en ON(entre +5 y +15 volts)
- Circuito CB(Clear to Send) en ON(entre +5 y +15 volts)
- Circuito CC(Data Set Ready) en ON(entre +5 y +15 volts)
- Circuito CD(Data Terminal Ready) en ON(+5 a +15 volts)

Cualquier señal que se transfiera a través de este circuito, cuando se cumplan las 4 condiciones que se acaban de mencionar, será transmitida por el equipo de comunicación de datos al canal de comunicación.

Circuito BB: Received Data (Receptor de datos, pin 3)

Este circuito se usa para recibir datos desde el equipo de comunicación de datos (provenientes del canal de comunicación), en forma serial, siguiendo el patron de comunicación que corresponda al tipo de transmisión (sincrónica, asincrónica). El equipo de comunicación de datos deberá mantener esta línea en la condición MARK (-5 a -15 volts), mientras la señal en el circuito CA (Request to Send) esté en la condición ON(+5 a +15 volts); esta última consideración es válida solo si se está usando la modalidad half duplex (transmisión y recepción no simultanea).

Circuito CA: Request to Send (Requerimiento de Envio, pin 4)

Este circuito se usa para controlar el flujo de datos en el equipo de comunicación de datos. El equipo de comunicación de datos puede transmitir datos desde el equipo terminal al canal de comunicación y desde el canal de comunicación al equipo terminal; en el primer caso, se dice que el equipo de comunicación de datos se encuentra en el modo transmisión (recibe datos desde el equipo terminal y los transmite al canal de comunicación) y en el segundo caso se dice que el equipo de comunicación de datos se encuentra en el modo recepción (recibe datos del canal de comunicación y los transmite al equipo terminal). En la modalidad de comunicación half duplex, el equipo de comunicación de datos solo podrá estar en uno de estos dos modos (transmisión o recepción) y en estos casos, el nivel lógico del circuito CA (Request to Send), define el modo en que operará el equipo de comunicación de datos; si CA está en ON (+5 a +15 volts), el equipo de comunicación de datos permanece en el modo transmisión, recibiendo datos desde el terminal y enviandolos al canal de comunicación, inhibiendo adicionalmente la recepción de datos desde el canal de comunicación; si CA (Request to Send) se encuentra en la condición OFF(-5 a -15 volts), el equipo de comunicación de datos permanece en el modo recepción, recibiendo datos desde el canal de comunicación y enviandolos al equipo terminal. El circuito CA(Request to Send, en conjunto con el circuito CB(Clear to Send), son usados para el "handshaking" entre el equipo terminal y el equipo de comunicación de datos.

En la modalidad full duplex, en la cual el equipo de comunicación de datos puede operar simultaneamente en ambos modos, la condición ON en el circuito CA (Request to Send), mantiene el equipo de comunicación de datos en el modo transmisión, sin inhibir la recepción desde el canal de comunicación; la condición OFF en CA, mantiene al equipo de comunicación de datos en el modo de no transmisión.

Circuito CB: Clear to Send (pin 5)

El estado de la señal en este circuito se usa para indicar al equipo terminal que el equipo de comunicación de datos

está listo para transmitir datos al canal de comunicación. La condición ON (+5 a +15 volts) de la señal en el circuito CB, en conjunto con la condición ON en los circuitos CA (Request to Send), CC (Data Set Ready) y CD (Data Terminal Ready), es una indicación para el equipo terminal de datos que las señales colocadas por el en el circuito BA (Transmitt Data), serán transmitidas al canal de comunicación de datos(ver figura 6-17).

Circuitos CC (Data Set Ready) y CD (Data Terminal Ready) (pines 6 y 20 respectivamente)

Estos circuitos son provistos para que el equipo de comunicación de datos y el equipo terminal de datos indiquen su estatus. La condición ON en cualquiera de ellos , indicará que el equipo correspondiente (equipo terminal o equipo de comunicación) se encuentran listos para operar.

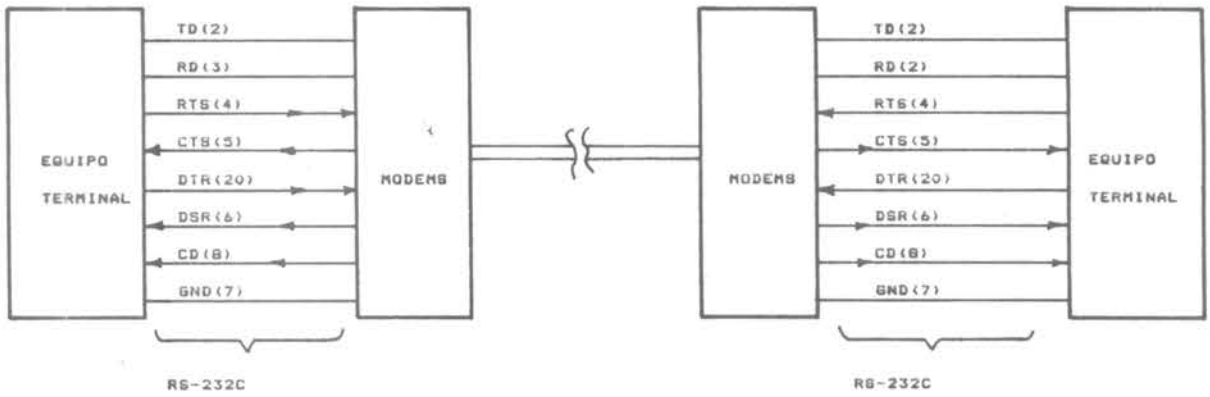


Figura 6-17. Conexión de dos equipos que cumplen el estandard RS-232 usando MODEMS

La interfase RS-232, como se mencionó anteriormente, fue desarrollada para normar la conexión entre equipos terminales de datos (computadores, microprocesadores, CRT, teletipos, etc.) y equipos de comunicación de datos (MODEMS o Data Set); sin embargo, muchas veces no se requiere del uso de un equipo de comunicación de datos, generalmente en distancias cortas y con líneas dedicadas o líneas muertas, es posible la conexión directa de dos equipos terminales, sin intervención de equipos de comunicación de datos. La conexión de dos equipos terminales (que cumplan con el estandard RS-232) en forma directa, requiere generalmente de algunas conexiones adicionales en ambos equipos terminales, con la finalidad de suplir la ausencia del equipo de comunicación de datos.

En la figura 6-17 se muestra una conexión típica de dos equipos terminales con interfase RS-232, usando equipos de comunicación de datos. Adicional a las conexiones para transmisión, recepción y tierra de señal, existen otras líneas que permiten establecer un cierto protocolo, cuya finalidad es lograr el acoplamiento entre el equipo de comunicación de datos y el equipo terminal. El estandard RS-232 exige que para que el equipo terminal de datos pueda iniciar el envío de un dato o cadena de datos a través del circuito Transmit Data (BA), las señales en los circuitos CA, CB y CC, deben estar en ON; cuando se usa el equipo de comunicación de datos, el mismo, se encarga de mantener en ON la señal sobre el circuito CC (Data Set Ready) y de poner en ON la señal sobre el circuito CB((Clear to Send) en respuesta a la activación por parte del equipo terminal de la señal sobre el circuito CA (Request to Send); en otras palabras, el equipo terminal cuando requiere hacer una transmision, le "avisa" al equipo de comunicación de datos de esta situación colocando en ON la señal sobre el circuito CA (Request to Send); el equipo de comunicación de datos, en respuesta a esta indicación le avisa al terminal que esta listo para iniciar la transmisión al canal de comunicación, de los datos recibidos por la entrada BA (Transmitt Data), activando la señal Clear to Send; el equipo terminal no iniciará la transmisión hasta tanto no reciba la autorización del equipo de comunicación de datos. Si no se usa equipo de comunicación de datos, es necesario proveer algun medio que permita cubrir las actividades del equipo de comunicación de datos en el protocolo de acoplamiento; la figura 6-18 muestra la forma en la que se simula el equipo de comunicación de datos, a fin de lograr la conexión directa de dos equipos terminales con interfase RS-232.

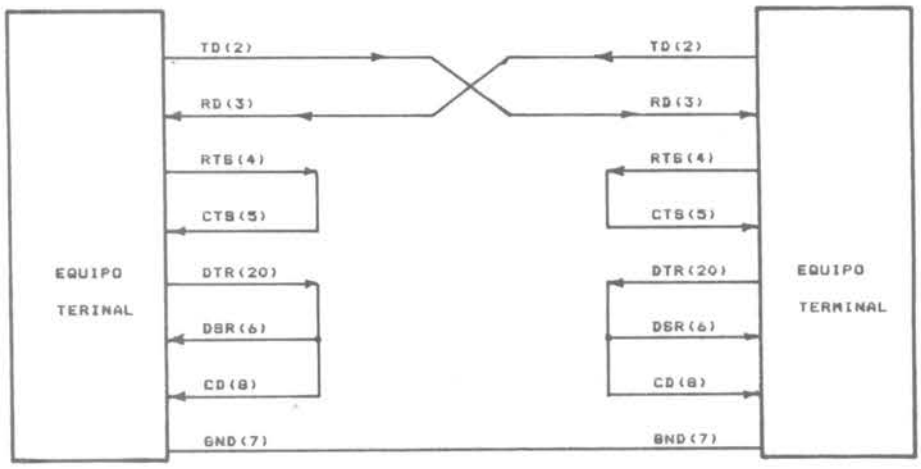


Figura 6-18. Conexión de dos equipos con RS-232, en forma directa

Otro inconveniente que generalmente se presenta, es el de los niveles de voltaje; en algunos casos, la interfase del equipo terminal no es capaz de suministrar las señales en los distintos circuitos de la interfase, a los niveles exigidos por el estandard RS-232; es común encontrar elementos receptores o transmisores con salida TTL, como es el caso del USART y el UART anteriormente descritos, manejando directamente la entrada y la salida serial de datos; en estos casos no es posible conectarse directamente a un equipo con interfase RS-232 (niveles entre -15 y +15 volts); sin embargo, existen diversos circuitos integrados que permiten realizar la conversión desde niveles de tensión TTL (0 a +5 volts) a niveles de tensión RS-232 (+15 a -15 volts) y viceversa. Los circuitos integrados MC1488 (convierte de nivel TTL a nivel RS-232) y MC1489 (convierte de nivel RS-232 a nivel TTL), son comunmente usados para los propósitos de adaptación de voltajes en los distintos circuitos de interfase.

6.2.2. BUS S-100 PARA MICROCOMPUTADORES

El bus S-100, es una interfase diseñada para la conexión de los distintos elementos componentes de un sistema microcomputador tales como: memoria, interfases para entrada/salida o cualquier otro módulo de interfase. Como su nombre lo sugiere, el bus S-100 consta de 100 líneas o contactos, en los cuales se conectan diferentes señales. Las señales en el bus S-100, según su función se pueden clasificar en 4 categorías:

Señales para fuente de poder.

Señales para direccionamiento.

Señales para datos.

Señales para reloj y sincronización.

a) Señales para fuente de poder

Tres fuentes de voltaje DC, no regulados, estan disponibles en el bus S-100: una de +8 volts, una de +18 volts y una de -18 volts. Las fuentes de alimentación en el bus S-100, como ya se dijo antes, son todas NO reguladas; la regulación de voltaje debe ser llevada a cabo por cada expansión o módulo de interfase que se vaya a conectar al bus. La existencia de tantos elementos de regulación como módulos existan en el sistema, tiene ciertas ventajas respecto de la conexión de todos los módulos del sistema a una fuente común de voltaje regulado, las principales ventajas de esta distribución de los elementos de regulación son:

Las caidas de voltaje en el bus propiamente dicho, no afectan significativamente el voltaje regulado independientemente en cada uno de los módulos.

El calor producido por los elementos reguladores, es distribuido en un volumen relativamente grande, lo que facilita su disipación.

La falla en un elemento regulador, no necesariamente implica la no operación del sistema total.

Por último está el costo inicial del sistema, el cual será menor, en virtud de que que el sistema básico no tiene que traer previsión de capacidad de voltaje regulado, mas alla del requerido por el componente básico, ya que cuando se realicen las expansiones, las mismas traeran su propia fuente de regulación.

Los pines o contactos del bus S-100, en los cuales se encuentran los voltajes no regulados que se mencionaron anteriormente (+8v, +18V y -18V) se dan en la tabla 6-1.

TABLA 6-1

CONTACTOS DE ALIMENTACION EN EL STANDARD S-100

pin	señal
1,51	+8 volts no regulados
2	+18 volts no regulados
52	-18 volts no regulados
50,100	tierra (GND)

b) Señales para direccionamiento

El estandard S-100 tiene previsión para 16 líneas de direccionamiento. Las señales de direccionamiento en el bus, son manejadas por drivers TTL de tres estádos. Se ha tomado previsión para la conexión de canales de DMA en el bus, para lo cual se dispone de una señal que permite habilitar o deshabilitar los drivers de direcciones del bus S-100, con la finalidad de permitir la operación del DMA.

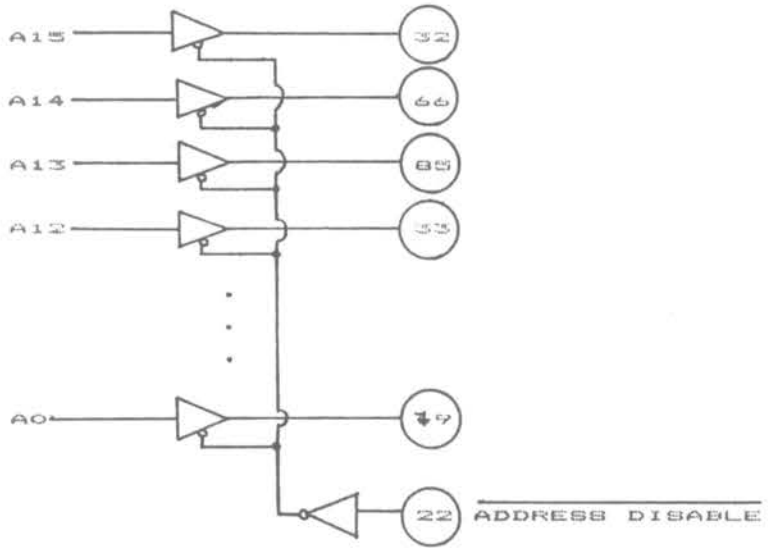


Figura 6-19. Bus de direcciones del estandard S-100

El esquema de conexión del bus de direcciones de los diferentes módulos componentes del sistema al bus de direcciones del bus S-100, se muestra en la figura 6-19, en la cual se puede apreciar la forma en que la señal de deshabilitación del bus (Address Disable), puede ser usada para deshabilitar los drivers del bus de direcciones del estandard S-100. La tabla siguiente resume las distintas señales de direccionamiento en el bus S-100, con sus correspondientes pines o contactos que le han sido asignados.

TABLA 6-2

SEÑALES DE DIRECCIONAMIENTO EN EL BUS S-100

Pin	Señal	Pin	Señal
79	A0	84	A8
80	A1	34	A9
81	A2	37	A10
31	A3	87	A11
30	A4	33	A12
29	A5	85	A13
82	A6	86	A14
83	A7	32	A15

c) Señales para datos

El bus estandard S-100, fue concebido inicialmente para un microprocesador de 8 bits de bus de datso (especificamente el 8080 de INTEL), sin embargo, se tomaron previsiones en el bus hasta para 16 líneas de bus de datos, en virtud de que en el S-100, el bus de datos, a diferencia del de el microprocesador, es unidireccional; en otras palabras el bus bidireccional del microprocesador (8 bits), se convierte en el estandard S-100 en dos buses unidireccionales de 8 bits cada uno, tal como se muestra en la figura 6-20. De manera que en el estandard S-100 existen dos buses de datos: un bus de datos de salida (DO0-DO7 y un bus de datos de entrada (DI0-DI7).

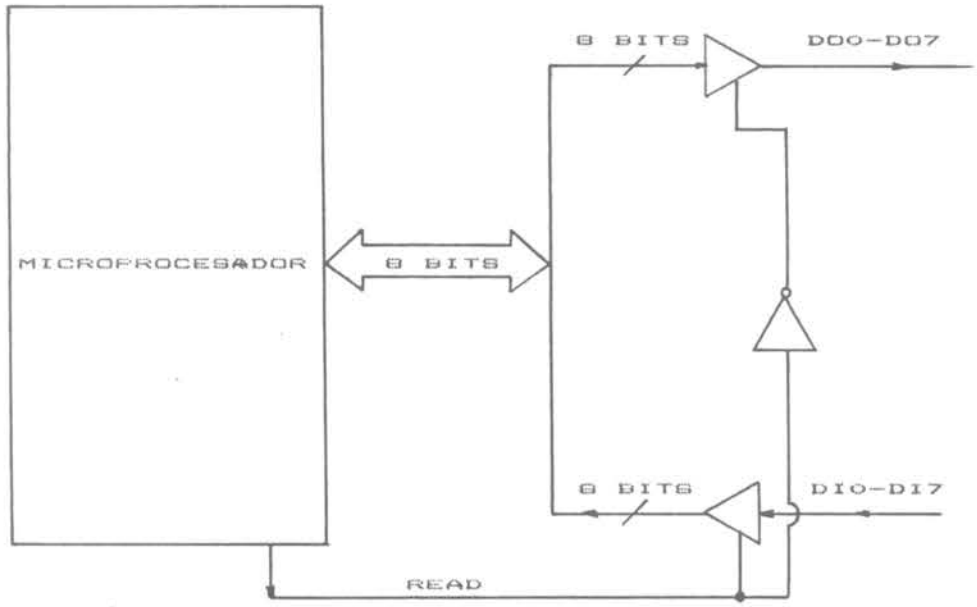


Figura 6-20. Bus de datos en el estandard S-100

Como se puede apreciar en la figura 6-20, se usa la señal READ (en el 8080 se usa la señal DBIN), para habilitar los drivers del bus de entrada y esta misma señal negada, se usa para habilitar o deshabilitar los drivers de salida del bus de datos del estandard S-100.

Semejante a como está implementado en el bus de direcciones, el bus de salida de datos del estandard S-100, se puede habilitar o deshabilitar con una señal de control especial, llamada DO DISABLE, lo que se usa generalmente en la implementación de los canales de DMA. La figura 6-21 muestra el esquema de conexión que permite la deshabilitación del bus de datos de salida.

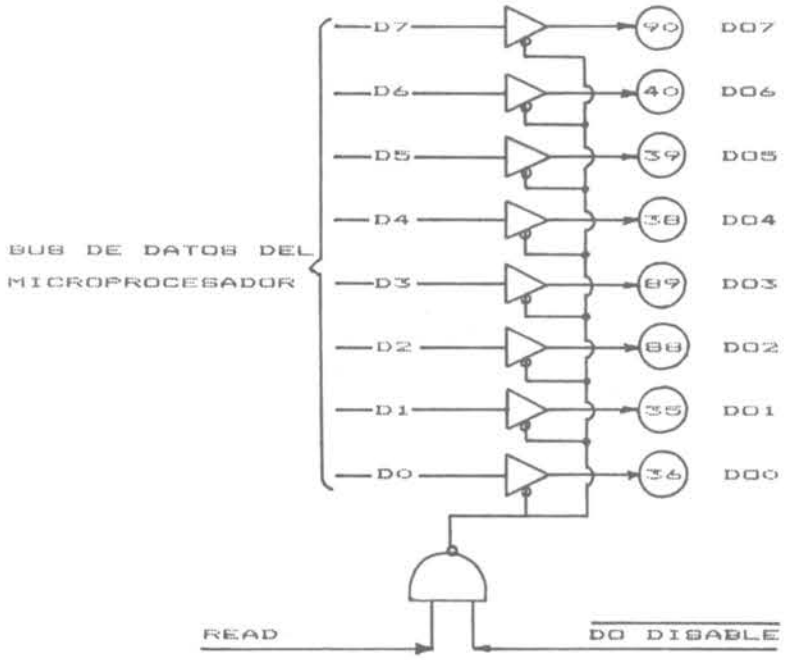


Figura 6-21. Bus de datos de salida en el estandard S-100

La tabla siguiente muestra las diferentes señales de datos previstas en el estandard S-100, con su correspondiente contacto o pin.

TABLA 6-3

BUS DE DATOS EN EL ESTANDARD S-100

Pin	Señal	PIN	Señal
95	DI0	36	DO0
94	DI1	35	DO1
41	DI2	88	DO2
42	DI3	89	DO3
91	DI4	38	DO4
92	DI5	39	DO5
93	DI6	40	DO6
43	DI7	90	DO7

d) Señales de control y sincronización

En el estandard S-100 estan previstas una serie de líneas destinadas a las señales de control, que permiten la conexión y operación adecuada de las diferentes extensiones o módulos que compongan al sistema. Como ya se dijo anteriormente, el S-100 fue concebido inicialmente para un sistema basado en el microprocesador 8080, por esa razon la mayoría de las señales de control en el bus, tienen un equivalente en el microprocesador 8080.

Se han previsto hasta 3 líneas para conexión de señales de reloj (ϕ_1, ϕ_2, ϕ_{clock}). La frecuencia del reloj aplicado en clock es fija, e igual a 2MHz, independientemente del microprocesador empleado; Las frecuencias de los relojes aplicados en ϕ_1 y ϕ_2, dependen del microprocesador que se esté usando. Todas las señales de reloj en el estandard S-100 son compatibles TTL. La tabla 6-4 resume todas las señales de control disponibles en el estandard S-100. Todas las señales que tienen antepuestas una "p" o una "s", son funcionalmente equivalentes a las respectivas señales en el microprocesador 8080.

Las líneas de control ADDR DSBL, STAT DSBL, DO DSBL, STAT DSBL, C/C DSBL, son usadas para controlar los drivers de tres estados del bus de direcciones, bus de datos y bus de control. ADDR DSBL, cuando se activa (0 lógico), deshabilita el bus de direcciones; DO DSBL, cuando se activa (0 lógico), deshabilita el bus de datos de salida; C/C DSBL, cuando se activa (0 lógico), deshabilita las líneas de control y las señales de reloj en el bus S-100; la línea STAT DSBL, se usa para desactivar las líneas de estatus en el bus S-100. Algunas de las señales en el bus S-100 se aplican especificamente a un determinado microprocesador, es el caso específico de las líneas NMI, MREQ y RFSH, las cuales son aplicables unicamente cuando el sistema esté basado en el microprocesaor Z80; sus funciones en el bus S-100, son idénticas a las cumplidas por sus correspondientes líneas en el Z80.

Adicionalmente se han previsto en el bus S-100 líneas para permitir operar libremente el procesador o para operar en la modalidad "single step"; la señal RUN, cuando se activa, permite que el procesador opere libremente; cuando se desactiva (0 lógico), permite operar al microprocesador (el que disponga de esta facilidad), en la modalidad de "single step".

Las líneas de control PROTECT y UNPROTECT, son provistas para permitir la escritura o no en la memoria RAM del sistema. La línea PROTECT, permite proteger la RAM de escrituras no deseadas y la línea UNPROTECT, permite habilitar la escritura en la RAM. La línea PROTECT STATUS, ha sido provista para averiguar el estatus de las líneas PROTECT y UNPROTECT.

TABLA 6-4

BUS DE CONTROL EN EL ESTANDARD S-100

Pin	Señal	Pin	Señal
3	XRDY	44	sM1
12	$\overline{\text{NMI}}$	45	sOUT
18	$\overline{\text{STAT DSBL}}$	47	sMEMR
19	$\overline{\text{C/C-DSBL}}$	48	sHLTA
20	UNPROTECT	49	$\phi_{\text{clock(2 MHz)}}$
21	SINGLE STEP	53	$\overline{\text{SSW DSBL}}$
22	$\overline{\text{ADDR DSBL}}$	54	$\overline{\text{EXT CLR}}$
23	$\overline{\text{DO DSBL}}$	65	$\overline{\text{MREQ}}$
24	ϕ_2	66	$\overline{\text{RFSH}}$
25	ϕ_1	68	MWRITE
26	pHLDA	69	$\overline{\text{PS}}$
27	pWAIT	70	PROTECT
28	pINTE	71	RUN

6.2.3. BUS IEEE 488

El bus estandard IEEE 488, es un estandard diseñado para aplicar a la interfase que conecta instrumentos electrónicos de medida, programables o no, con otros aparatos y accesorios requeridos en un sistema de instrumentación. El bus IEEE 488 es aplicable solo a instrumentos que manejan señales digitales y en el mismo, no pueden conectarse mas de 15 dispositivos de medida.

Consta el bus IEEE 488 de 16 líneas, de las cuales, 8 son usadas para la transmisión paralela de un byte, y el resto de las líneas son usadas para control y operaciones de "handshaking" entre los diferentes elementos del sistema de instrumentación.

Existen tres tipos de elementos que pueden ser conectados en el bus:

Elementos o dispositivos capaces de identificar y reconecer una referencia a ellos (mediante una dirección

específica asociada con el) y recibir un mensaje o información proveniente de otro dispositivo conectado al bus; este tipo de dispositivos se definen en el estandard como LISTENERS.

Elementos o dispositivos capaces de identificar y reconocer una referencia a ellos, y responder enviando información a través del bus a otro dispositivo conectado en el mismo; este tipo de dispositivo es llamado en el estandard TALKERS.

Elementos o dispositivos capaces de controlar y direccionar a otros dispositivos conectados al bus; este tipo de dispositivo es llamado en el estandard CONTROLERS

Un mismo dispositivo puede ser LISTENER, TALKER, CONTROLER o solamente uno de estos tipos. La figura 6-22 muestra una estructura típica de conexión de elementos en un bus IEEE 488.

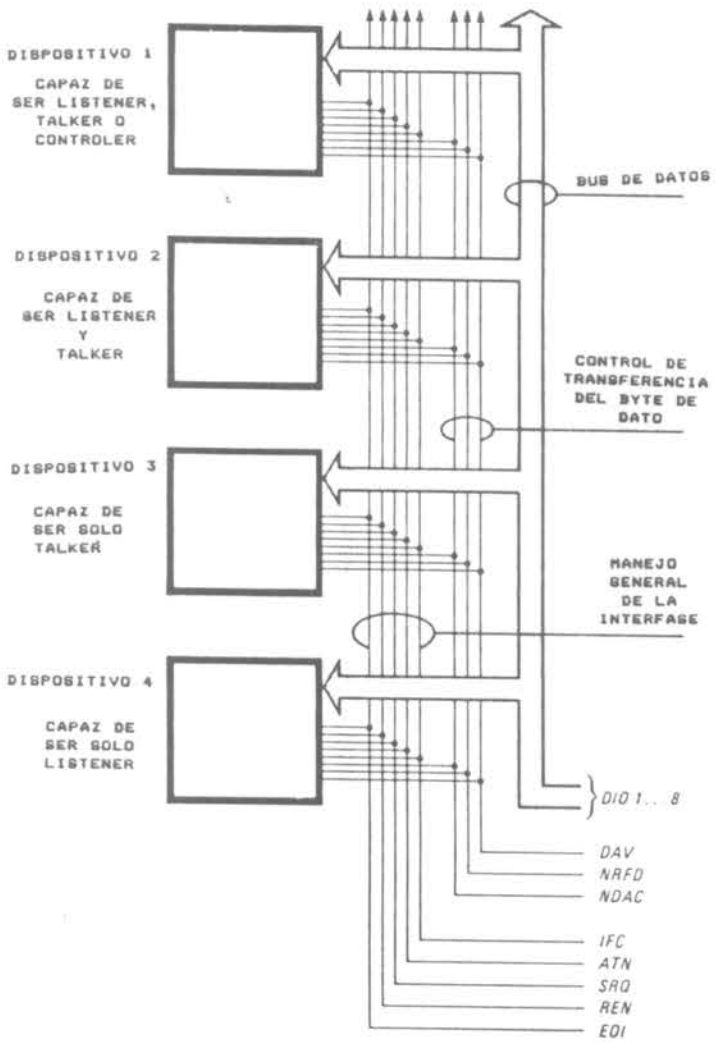


Figura 6-22. Conexión de elementos en un bus IEEE 488

Como se puede apreciar en la figura 6-22, existen 8 líneas de bus de datos (DIO1-DIO8) y 8 líneas de control; de estas 8 líneas de control, 3 se usan para el "handshaking", con el fin de coordinar el flujo de datos entre los TALKERS y los LISTENERS; estas líneas de control son DAV, NRFD y NDAC.

La línea DAV es usada para indicar la disponibilidad o no de datos en las líneas que coresponden al bus de datos (DI1-DI8); esta línea es colocada en el nivel uno lógigo por el TALKER direccionado, cuando coloca la información o dato en las líneas del bus.

La línea NRFD (Not Ready For Data), es usada para indicar una condición de "no listo para recibir datos"; cuando todos los LISTENERS en el sistema estan listos para recibir datos, la señal NRFD es colocada en el nivel cero lógico; si alguno de los LISTENERS conectados al bus no está listo para recibir datos, la línea NRFD se mantiene en el nivel uno lógico (not ready).

La línea NDAV es usada para indicar la aceptación o no, de los datos colocados en el bus por los TALKERS, por parte de los LISTENERS direccionados. La línea NDAV permanece en el nivel uno lógico, hasta que todos los LISTENERS direccionados hayan aceptado el dato en el bus.

Las cinco líneas de control restantes, lo constituyen ATN(Attention), IFC(Interface Clear), SRQ(Service Request), REN(Remote Enable) y EOI(End or Identify).

La línea ATN es usada en el bus para controlar el uso que se le dará al bus de datos. En operación normal (TALKERS enviando información a LISTENERS), esta línea se mantiene en el nivel cero lógico; cuando el controlador quiere llamar la atención de un determinado elemento (TALKER o LISTENER) conectado en el bus, cambia el uso del bus de datos, desde su modo normal (bus de datos) a su modo de comando, cambiando el nivel lógico de la señal ATN; cuando la señal ATN está activa (uno lógico), la información presente en el bus es interpretada por los LISTENERS y TALKERS como comandos o direcciones.

La línea IFC es usada por el controlador para darle un reset a todos los dispositivos conectados al bus.

La línea SRQ es usada por un dispositivo cualquiera (LISTENER o TALKER), para requerir atención especial por parte del controlador, o para solicitar una interrupción de la secuencia de eventos que esten ocurriendo en un momento determinado en el bus.

La línea REN es usada generalmente con alguna otra línea, para seleccionar entre dos fuentes alternas de programación de los dispositivos o instrumentos del sistema, como por ejemplo, entre un panel de control remoto y un panel de control local.

La línea EOI es usada para indicar el fin de una transferencia de un bloque de datos (varios bytes). En algunos casos, se usa en conjunto con la línea ATN para ejecutar una secuencia de "poling". Cuando el bus de datos es colocado por el controlador en el modo comando(ATN en el nivel uno lógico), una activación de EOI(nivel uno lógico), indica que cada dispositivo deberá colocar en la línea que le corresponda en el bus, un bit que aportara información referente a su estatus, lo cual permitirá al controlador, identificar cual de los dispositivos conectados al bus ha hecho un requerimiento de atención (SRQ activa).

Todas las señales en el bus IEEE 488 estan definidas en voltajes de lógica negativa; al nivel cero lógico le corresponden valores de voltaje entre 2 V y 5 V y al nivel uno lógico le corresponden valores de voltaje comprendidos entre 0.8 V y 0 V. Los voltajes comprendidos entre 0.8 V y 2 V no estan definidos en el estandard. La tabla 6-5 resume todas las señales en el bus IEEE 488 y la asignación de contactos o pines en el conector usado por esta interfase (conector de 25 pines).

TABLA 6-5

ASIGNACION DE PINES EN LA INTERFASE IEEE 488

Pin	Señal	Pin	Señal
1	DIO1	13	DIO5
2	DIO2	14	DIO6
3	DIO3	15	DIO7
4	DIO4	16	DIO8
5	EOI	17	REN
6	DAV	18	GND(6)
7	NRFD	19	GND(7)
8	NDAC	20	GND(8)
9	IFC	21	GND(9)
10	SRQ	22	GND(10)
11	ATN	23	GND(11)
12	SHIELD	24	GND

P R A C T I C A S

PRACTICA 1

OBJETIVO:

Familiarizar al estudiante con los comandos del monitor, el teclado y el display de la tarjeta IMSAI 48.

PREPARACION PARA LA PRACTICA

Leer la función EXAMINE/MODIFIQUE en el manual de usuarios del sistema IMSAI, la cual le permitirá cargar y revisar programas o datos en memoria de programa o en memoria de datos.

Leer los modos "Standalone" y "Debug" de la sección xxx. Estos modos corresponden a las distintas formas de ejecutar un programa.

Leer la función de las teclas Reset y Clear en la seccion xxx.

Lea cuidadosamente las restricciones y comportamiento del monitor cuando se ejecutan programas en el modo "DEBUG".

PROCEDIMIENTO:

1. Conecte la tarjeta IMSAI 48 a 5 Volts y a tierra.

2. Cargue a partir de la dirección C00 el siguiente programa de prueba.

DIR	CONTEN	ETIQUET	MNEMONICO	COMENTARIOS
C00	F5	INICIO	SEL MB1	Selecciona banco de memoria 1

C01	23	MOV A, 07	A := 07
C02	07		
C03	03	ADD A, F0	A := A + F0
C04	F0		
C05	A8	MOV R0,A	R0 := A
C06	17	INC A	A := A + 1
C07	47	SWP A	A0-A3 A4-A7 cambia nibbles del acumulador A
C08	A8	MOV R0,A	R0 := A
C09	00		

Pasos a seguir para cargar el programa de prueba:

Presione las teclas:

EXAM/MODIFY
PROGRAM MEM

Ahora escriba la dirección de inicio del programa
C00

Presione la tecla NEXT

Ahora comience a introducir el programa separando cada código de instrucción por el comando NEXT:

F5
NEXT
23
NEXT

.
.
.

A8
NEXT
00
NEXT
ENTER

Ahora examine o revise el programa cargado; para ello presione:

```
EXAM/MODIFY
PROG MEM
C00
NEXT
```

vaya comparando lo que tiene en memoria con lo que UD. tiene en el papel.

3. Ejecución del programa

a) Corra el programa de prueba en el modo "Standalone", para lo cual presione las teclas:

```
EXEC
C00 (dirección de inicio del programa)
NEXT
ENTER
```

Ahora examine el contenido de los registros A y R0, presione para ello:

```
EXAM/MODIFY
REG MEM (numeros del 0 al 7 )
```

b) Corra ahora el programa usando "breakpoint" y en cada parada examine e interprete el contenido de: PC, A, R0 y cualquier otro registro de interes. Este modo se usa para depurar de errores un programa. Para la ejecución del programa con "breakpoint" presione las siguientes teclas:

```
EXEC/BREAKPOINT
C00             dirección de inicio del programa
,
C09             dirección hasta donde se ejecuta
                        ra el programa
```

Con el comando que se acaba de dar se ha mandado a ejecutar el programa en el modo "DEBUG", con el breakpoint colocado en la última instrucción. Proceda a revisar los registros y luego ejecute de nuevo el programa pero con el breakpoint en la dirección C07. Examine de nuevo los registros

y compare los resultados con los de la ejecución anterior.

Tienen los registros A, R0, PC los mismos contenidos?. Por que?

Revise el programa que se cargo en memoria y observe si despues de la última ejecución coincide con el que se introdujo. Coincide?. Por que?.

4. Cargue ahora el programa saludo, siguiendo el procedimiento usado en la parte 2. Este programa saludo tiene por finalidad mostrar en el display el mensaje: "HOLA SOY IMSAY48" (en vista de que el display del IMSAY es del tipo 7 segmentos, no se pueden mostrar los caracteres M e Y para lo cual se usa la m y el 4 respectivamente).

Ejecute el programa saludo tanto en el modo "Standalone" como en el modo "DEBUG", usando breakpoint al final de cada lazo; revise los registros y el programa despues de cada corrida; recuerde las restricciones que impone la corrida de programas en el modo "DEBUG".

PROGRAMA SALUDO

DIR	CONTE.	ETIQUE	MNEMONICO	COMENTARIO
C00	E5		SEL MB0	selecciona banco de memoria cero
C01 C02	54 13		CALL CLEAR	llamado a la rutina que limpia el display
C03	F5		SEL MB1	selecciona banco de moria cero
C04 C05	B8 20	RET1	MOV R0, DIR	R0 := 20 dirección de dirección de los da tos(el mas bajo)
C06 C07	BB 02		MOV R3, 02	inicializa contador de anuncios
C08 C09	B9 09	RET2	MOV R1, 09	inicializa contador de caracteres
C0A C0B	23 0C	RET3	MOV A, 0C	A := 0C página del da to en RAM externo
C0C	3A		OUT P2,A	saca la página por P2

C0D	80	MOVX A, @R0	carga A con el dato almacenado en la loca lidad apuntada por R0 memoria externa
C0E	E5	SEL MB0	selecciona el banco 0
C0F C10	54 1F	CALL OUTDSP	llamado a la rutina para mostrar en datos en el display (perte nece al monitor)
C11	F5	SEL MB1	selecciona banco de memoria cero
C12	C5	SEL RB0	selecciona banco de registros 0
C13	18	INC R0	incrementa R0 para apuntar al siguiente caracter a mostrar
C14 C15	E9 0A	DJNZ R1,RET3	regresa a RET3 si no se han escrito todos los caracteres
C16 C17	94 40	CALL DELAY	llamado a la rutina DELAY, del usuario, para evitar parpadeos
C18	E5	SEL MB0	
C19 C1A	54 13	CALL CLEAR	
C1B	F5	SEL MB1	
C1C C1D	EB 08	DJNZ R3,RET2	decrementa contador de anuncios y salta a RET2 si no es cero
C1E C1F	84 04	JMP RET1	salta de nuevo al ini cio (RET1) repitiendo se indefinidamente

fin de saludo

Los datos del programa, constituidos por los códigos de los caracteres a mostrar en el display, se cargan a partir

de la dirección C20 (apuntada inicialmente por R0) y fueron sacados del programa monitor líneas 39 a 71. A continuación se muestran las direcciones y los datos a cargar:

DIR	CONTE.	ETIQUE	MNEMONICO	COMENTARIO
C20	76		H	código para mostrar H
C21	3F		O	código para mostrar O
C22	38		L	código para mostrar L
C23	77		A	código para mostrar A
C24	00			código de display apagado
C25	6D		S	código para mostrar S
C26	3F		O	código para mostrar O
C27	66		Y	código para mostrar Y
C28	00			display apagado
C29	o6		I	código para mostrar I
C2A	54		m	código para mostrar m
C2B	44			segunda parte de la m
C2C	6D		S	código para mostrar S
C2D	77		A	código para mostrar A
C2E	06		I	código para mostrar I
C2F	00			display apagado
C30	66		4	código para mostrar 4
C31	7F		8	código para mostrar 8

La subrutina DELAY, es una rutina hecha por el usuario que introduce un retardo de aproximadamente 1 segundo, entre el display de uno y otro anuncio, suficiente para ver el display estable; un listado de esta subrutina se da a continuación y la misma será cargada a partir de la dirección C40, tal como se muestra a continuación:

DIR	CONTE.	ETIQUE	MNEMONICO	COMENTARIO
C40	C5	DELAY	SEL RB0	selecciona banco 0
C41	35		DISTCNTI	deshabilita interrup ción por tiempo
C42 C43	BA 20		MOVE R2, 20	inicializa contador de flags
C44	55	INIC	START T	inicializa el timer
C45 C46	16 49	LOOP	JTF COUNT	salta a COUNT si el flag del timer se ac tivo
C47 C48	84 45		JMP LOOP	va a lazo de espera por activación del flag del timer
C49 C4A	EA 44	COUNT	DJNZ R2,INC	decrementa R2 y salta a INIC si no es cero
C4B	FA		MOV A, R2	A := R2; esto es para usar el break point
C4C	83		RET	retorna al programa que la llamó
C4D	00		NOP	no hace nada

PRACTICA 2

OBJETICOS:

a) Continuar familiarizando al estudiante con el IMSAI 48 en lo referente a los comandos del monitor, manejo del teclado y el display.

b) Examinar algunas fallas comunes en la programación de microprocesadores.

PREPARACION PARA LA PRACTICA

Leer en el listado del programa monitor del IMSAI la sección "DISPLAY SEGMENT CODE" (código para los segmentos del display), líneas 39 a 71 del programa monitor. Esta lectura lo ayudará en la construcción de caracteres en el display.

Leer los comandos de control del sistema IMSAI.

Leer las rutinas OUTDISPLAY y OUTDISPR6 del monitor.

PROCEDIMIENTO

1) Modifique el segundo programa de la practica 1 (SALUDO), para que escriba el nombre de cada uno de los integrantes del equipo.

2) Ahora usted examinará algunas fallas comunes en programación de microprocesadores.

2.1. Cargue el siguiente programa, el cual, con un lazo indefinido, almacena en memoria externa, a partir de una dirección determinada, números consecutivos.

PROGRAMA 1

DIR	CONTEN.	ETIQUETA	MNEMONICO	COMENTARIO
C00	88		MOV R3, 00	R3 := 00; dato ini
C01	00			cial a guardar
C02	B9		MOV R1, 00	R1 := 00; comienzo
C03	00			de la página
C04	23	LOOP	MOV A, 0C	A := 0C; página
C05	0C			en donde se guarda
				ra el dato
C06	3A		OUT P2,A	P2 := A; saca la
				página por P2
C07	FB		MOV A,R3	A := R3 pone en A
				el dato a escribir
C08	91		MOVX @R1,A	escribe A(dato) en
				memoria externa
C09	19		INC R1	R1 := R1+1; apunta
				próxima dirección
C0A	1B		INC R3	R3 := R3+1; próxi
				mo dato
C0B	84		JMP LOOP	salta de nuevo a
C0C	23			LOOP

Corra el programa 1 en el modo STANDALONE (EXEC). Como en el programa hay un lazo infinito, no debe parar por si solo. como haría Ud. para pararlo?.

Suspenda la ejecución del programa y revise mediante los comandos del monitor, el contenido de las localidades de memoria donde cargo su programa; está su programa completo?; por que?.

2.2. Cargue el siguiente programa a partir de la dirección C00.

PROGRAMA 2

DIR	CONTEN.	ETIQUETA	MNEMONICO	COMENTARIO
C00	BA		MOV R2, 05	R2 := 5
C01	05			
C02	23		MOV A, 0A	A := 0A
C03	0A			
C04	47		SWAP A	A0-A3 A4-A7 intercambia los nibbles de A
C05	03		ADD A, 08	A := A + 8
C06	08			
C07	79		ADD A,R2	A := A + R2

Corra el programa 2 en el modo BREAKPOINT desde la C00 hasta la C07; chequee el contenido del registro A; tiene el valor que Ud. esperaba?. Que sucedio?. Corrija el programa 2 y corralo de nuevo.

2.3) Cargue el siguiente programa y corralo en el modo STANDALONE.

DIR	CONTEN.	ETIQUETA	MNEMONICO	COMENTARIO
C00	F5		SEL MB1	selecciona banco de memoria 1
C01	23	LOOP	MOV A, 08	A := 8
C02	08			
C03	E5		SEL MB0	selecciona banco de memoria 0
C04	54		CALL DISPRG	muestra A por el
C05	68			display
C06	84		JMP LOOP	salta para repe
	01			tir lazo que sa
				ca por display A

Muestra el display lo que Ud. esperaba?. por que?. Modifique el programa y vuelvalo a correr.

3) Escriba un programa que permita multiplicar 4x4, usando el método de las sumas sucesivas (4x4 = 4 + 4 + 4 + 4).

-Muestre el resultado por display.
-Deje el resultado en R2 de RB1.

-Haga el diagrama de flujo, codifiquelo y corralo.

Cuando esté desarrollando programas y para evitar incurrir en errores innecesarios, se recomienda seguir los siguientes pasos:

a) Haga un diagrama de flujo del programa, lo cual le permitirá ahorrar tiempo.

b) Codifique el diagrama en assembler (mnemónicos) y vaya colocando los comentarios para tener una guia rápida de lo que se pretende hacer con cada instrucción.

c) Coloque ahora el código en lenguaje de máquina, que representa a cada uno de los mnemónicos utilizados; revise cuidadosamente los códigos que escribe para asegurar que corresponden efectivamente al mnemónico anotado.

d) Escriba por último las direcciones, teniendo especial cuidado con las instrucciones de saltos (condicionales e incondicionales), las cuales usan estas direcciones.

PRACTICA 3

OBJETIVOS

1) Familiarizar al estudiante con el manejo del teclado.

2) Familiarizar al estudiante con el uso de los dos bancos de registros de la memoria de datos del 8035.

3) Familiarizar al estudiante con el uso de los dos bancos de memoria de programa y con los distintos modos de direccionamiento de que dispone el 8035.

PREPARACION

Leer en el manual del IMSAY el uso y forma de utilización de las subrutinas KEYINP, OUTDISPLAY y CLEAR

DISCUCION

Tipos de direccionamiento en el 8035

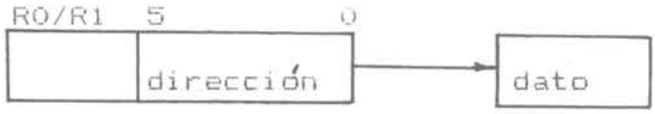
-INMEDIATO: dato como parte de la instrucción.

Ejemplo MOV A, dato (A := dato).

-REGISTRO: El dato a manipular está en el registro que se especifique.

Ejemplo MOV A,R1 (A := R1).

-INDIRECTO en memoria interna de datos: El dato a manipular está en la dirección de memoria de datos interna, apuntada por R0 o R1, tal como se esquematiza en la siguiente figura. Solo se usan los últimos 5 bits del registro (R0 o R1), en vista de que se requiere direccionar nada mas que 64 localidades (son las que trae el 8035 para memoria de datos).



ejemplos

MOVE A,@R0 (A := (R0))

MOVE A,@R1 (A := (R1))

Cuando se use este último modo de direccionamiento, se debe tener especial cuidado en el banco de registros que se esté utilizando, pues esto puede dar origen a errores fatales.

-INDIRECTO en memoria externa de programa: El dato a manipular se encuentra en memoria externa de programa; su ubicación dentro de la página corriente, la da el registro que se esté usando como apuntador indirecto (R0 o R1); este tipo de direccionamiento requiere que se saque por el puerto 2 la página de memoria en la que se colocará o de la que se extraerá el dato, tal como se esquematiza en la siguiente figura.

Puerto 2

Ejemplos

MOVX A, @R0 (A := (R0))

MOVX @R0,A ((R0) := A)

-DIRECCIONAMIENTO USANDO EL ACUMULADOR: El acumulador puede ser usado como registro para direccionamiento indirecto, de una forma similar a como se explicó con los registros R0 y R1 en el modo de direccionamiento indirecto a memoria externa de programa; sin embargo en este caso solo se usa la página corriente.

Ejemplo MOVP A,@A (A := (A))

PROCEDIMIENTO

1. Modifique el programa de multiplicación de 4x4, que se pidió en la práctica 2, de manera que reciva el multiplicando y el multiplicador por el teclado. Se recomienda usar la subrutina KEYINP, para introducir los datos desde el teclado. Haga el diagrama de flujo del programa, codifíquelo y corralo.

2. Usando los comandos del monitor, coloque los caracteres "R", "B", "A", "N","K" y "1", en el banco de registros 1 y los caracteres "R", "B", "A", "N", "K" y "0", en el banco de registros 0. Haga un programa que muestre alternativamente en el display, estos dos conjuntos de datos. Haga un diagrama de flujo del programa, codifíquelo y corralo tanto en el modo BREAKPOINT como en el modo STANDALONE.

3. Haga un programa que permita leer desde el teclado, dos vectores de datos, A y B, de ocho elementos cada unoy los almacene en la memoria de datos interna, a partir de la dirección 20 a partir de estos dos vectores, mediante una suma 1 a 1 de los elementos de ambos vectores, generar un vector C; dicho vector se deberá almacenar a partir de la dirección C80 de la memoria externa.

Ejemplo:

dir = 20	dir = 28	dir = C80
A(1) = 2	B(1) = 1	C(1) = 3
dir = 21	dir = 29	dir = C81
A(2) = 5	B(2) = 4	C(2) = 9

Haga el diagrama de flujo del programa, codifíquelo y corralo tanto en la modo BREAKPOINT como en el modo STANDLONE; use los comandos del monitor para mostrar al instructor los contenidos de las direcciones 20 a 2F de la memoria interna de datos, y C80 a C87, de la memoria externa.

PRACTICA 4

OBJETIVOS

1) Familiarizar al estudiante con el manejo de los puertos del 8035

2) Familiarizar al estudiante con el uso y manejo del expansor de puertos

3) Familiarizar al estudiante con el cálculo de la interfase

PREPARACION

Leer en el CAPITULO II la descripción del 8035 y en el apendice A, las especificaciones y características eléctricas del 8035.

Leer en el CAPITULO IV, sección 4.2.1.1., la descripción y operación del 8243 y en el apendice A, las características y especificaciones eléctricas del mismo.

Leer en el CAPITULO IV, secciones 4.1.1 y 4.1.2., la descripción funcional y la teoría de operación del sistema IMSAI.

Estudiar en un manual de circuitos TTL, las características eléctricas mas importantes de las compuertas TTL.

Ver en el apendice C, el diagrama de emsamblaje del sistema IMSAI y el diagrama de conexión de los diferentes componentes del mismo.

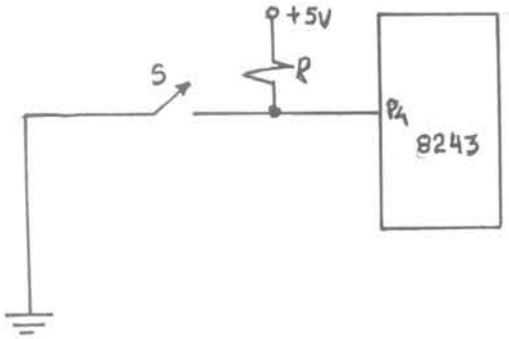
PROCEDIMIENTO

I.

1) Realice un programa que lea del puerto 4 y muestre el valor en el display (diagrama de flujo y codificación).

2) Calcule la interfase para la conexión en el puerto 4 del 8243 de 4 switches tal como se muestra a continuación (tome en consideración las especificaciones eléctricas del 8243).

3) Conecte los switches al puerto 4 del 8243 y corra el programa del punto 1, que lee y muestra por el display la lectura del puerto 4; haga las lecturas de los switches con varias combinaciones para demostrar la correcta operación del programa.



II.

1) Escriba un programa que permita leer desde el teclado y saque el dato leído por el puerto 1 del 8035 (diagrama de flujo y codificacion).

2) Cargue el programa anterior y corralo. Chequee con la punta de prueba lógica que las salidas se corresponden con las esperadas según la tecla que se pulsó.

3) Calcule la interfase necesaria para conectar LEDS en las líneas de salida del puerto 1 del 8035 (tome en consideración las especificaciones eléctricas para el puerto 1 del 8035 y el consumo del LED).

4) Conecte los LEDS a través de la interfase calculada, a las líneas del puerto 1 del 8035 y corra de nuevo el programa del punto 1.

III.

1) Escriba un programa que lea por el puerto 4 del 8243 y muestre la lectura por el puerto 1 del 8035. Use los LEDS y los switches que se conectaron en los puntos I y II. (haga el diagrama de flujo y la codificación del programa).

2) Cambie los switches a distintas posiciones y demuestre que el programa del punto 1 está operando correctamente.

IV.

1. Haga un programa que permita leer información por el puerto 1 del 8035 y la saque por el puerto 4(P4) del 8243, los bits menos significativos y por el puerto 5 (P5), los 4 bits mas significativos.

2) Haga los cálculos de las interfases de los switches que conectará en P1 y de los LEDS que conectará en P4 y P5.

3) Corra el programa del punto 1 y cambiando los switches, demuestre la operación correcta del mismo.

PRACTICA 5

OBJETIVOS

1) Familiarizar al estudiante con el manejador de teclado y display 8279.

2) Enseñar al estudiante el uso de T1 como una entrada serial del sistema IMSAI.

3) Familiarizar al estudiante con la operación y uso del TIMER/COUNTER del 8035.

PREPARACION

Estudiar en el CAPITULO IV:

a) Sección 4.2.2.1., referente a la teoría básica de teclado y display.

b) Sección 4.2.2.2., referente a la descripción funcional y programación de la operación del 8279.

Estudiar en el CAPITULO II, sección 2.2.5, lo referente a la operación y uso del TIMER/COUNTER del 8035.

PROCEDIMIENTO

I.

1) Realice un programa que lea números desde el teclado y los escriba en el display, sin usar las rutinas que el monitor trae para esos propósitos (haga el diagrama de flujo y la codificación).

2) Ejecute el programa y demuestre que funciona.

II.

1) Implemente un contador de eventos con el sistema IMSAI. Este contador debe cumplir con los siguientes requisitos:

a) La entrada de eventos será a través de T1, por medio de un pulsador.

b) La salida será por el display; el programa que realice para la captura de datos a través de T1, debe llevar la cuenta de las veces que se cerró el pulsador. El contador de eventos se incrementará en uno cada vez que se cierre el pulsador, por lo que el rebote del switche deberá ser eliminado para evitar conteos erroneos. Para la eliminación del rebote se deben emplear en primer lugar técnicas de software y en segundo lugar técnicas de hardware.

2) Realice un programa que saque por el puerto 7 del 8243 (P72) una onda cuadrada de frecuencia 100Hz. Ejecute el programa y observe la onda en el osciloscopio para demostrar la operación correcta del programa.

En todos los casos, el estudiante deberá calcular la interfase para conectar cualquier elemento requerido a los puertos o entradas del 8243 o del 8035; para ello tenga siempre presente las especificaciones eléctricas de los distintos componentes. Los programas que requiera realizar deben venir acompañados del diagrama de flujo correspondiente y de los comentarios necesarios para entender facilmente lo que se está haciendo.

PRACTICA 6

OBJETIVOS

1) Familiarizar al estudiante con la interfase para cassette del sistema IMSAI.

2) Mostrar la conexión de un transductor, como una corneta de audio, puede ser conectado a un microprocesador y manejado por el mismo.

3) Familiarizar al estudiante con el uso de tablas en el diseño de programas en microprocesadores.

4) Familiarizar al estudiante con los parámetros y consideraciones que se deben tener en cuenta para la conexión de memoria a un microprocesador.

5) Familiarizar al estudiante con el uso de los relees del sistema IMSAI.

PREPARACION

Leer en el CAPITULO IV, lo referente al uso de los puertos del 8243, para Entrada/Salida por cassette y las rutinas que trae el monitor para esos propósitos.

Leer en el CAPITULO V, sección 5.2., lo referente a la conexión de memoria a los microprocesadores.

Leer en un manual de circuitos TTL las especificaciones y características de la memoria estática 8279(16x4).
7489

Leer en el manual MOS las especificaciones y características de la memoria dinámica 4116 (16K x 1).

PROCEDIMIENTO

I.

Conecte el sistema IMSAI a un reproductor de cassette a través de las salidas y entradas que para esos propósitos trae el sistema IMSAI. Haga un programa cualquiera y grabelo en cassette, usando la rutina CASOUT, del monitor. Proceda ahora a leer el programa almacenado en el cassette, para lo cual usará la rutina CASIN del monitor.

II.

1) Construcción de un organo electrónico sencillo, usando el IMSAI y una corneta pequeña. La operación del organo debe ser tal que al presionar una tecla, deberá salir un pulso por P72 con la frecuencia correspondiente a esa tecla, tal como se especifica a continuación:

TECLA	NOTA	FRECUENCIA	TECLA	NOTA	FRECUENCIA
1	DO	261.6Hz	9	RE	587.3Hz
2	RE	293.7Hz	A	MI	659.3Hz
3	MI	329.6Hz	B	FA	698.5Hz
4	FA	349. Hz	C	SOL	784. Hz
5	SOL	382. Hz	D	LA	880. Hz
6	LA	440. Hz	E	SI	987.8Hz
7	SI	493.9Hz	F	DO	1046.5Hz
8	DO	523.3Hz			

a) Haga el programa para la implementación del organo electrónico que se acaba de describir. Dicho programa debe contemplar el hecho de que mientras una tecla esté pulsada la señal o nota correspondiente a esa tecla debe salir por el puerto 7 (P72).

b) Calcule la interfase para conectar la corneta al puerto 7 del 8243 (P72)

c) Cree Ud. que se puede modificar el programa para permitir que se pulsen simultaneamente mas de una tecla y salgan las notas correspondientes a las teclas pulsadas?. Explique. Es necesario hacer modificaciones de hardware para conseguir estos propósitos?.

III.

Conexión de memoria RAM adicional al sistema IMSAI.

a) Conecte la memoria RAM estática 8279 (16x4) al IMSAI, de manera que pueda grabar y leer en ella con los comandos del monitor "EXAMI/MODIFY" y "PROGRAM/MEMORY". Calcule la interfase tomando en consideración los tiempos y características eléctricas de la memoria 8279 y del sistema IMSAI.

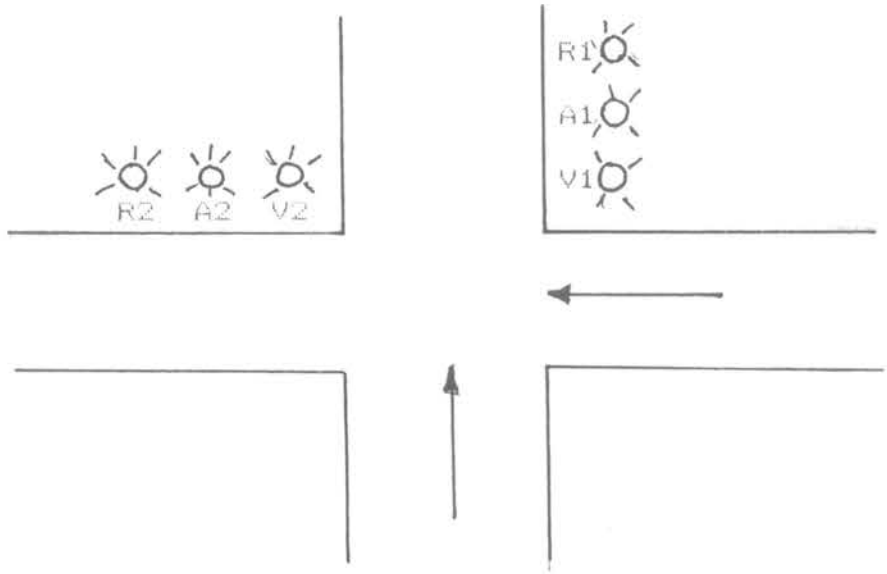
b) Que consideraciones especiales habría que hacer para conectar al sistema IMSAI una memoria dinámica como la 4116 (16K x 1). Tome en consideración que esta memoria, por ser dinámica, requiere de un refrescamiento periódico y adicionalmente debe considerar el hecho de que las direcciones se le colocan a la memoria en dos etapas (multiplexadas), primero la columna y despues la fila. Compare con la conexión de la memoria estática. Explique detalladamente todo lo referente a la conexión de dicha memoria al sistema IMSAI.

IV.

Implementación de un semáforo en un ruce de calles, usando los relees del sistema IMSAI. Las siguientes consideraciones deben ser tomadas en cuenta en la implementación del semáforo (ver el esquema):

a) La luz roja en la dirección 1 (R1) durará 60 seg. La amarilla (A1) para esa misma dirección durará, 10 seg y la verde (V1) durará 40 seg.

b) La luz roja en la dirección 2 (R2) tendrá una duración de 50 seg; la amarilla en esa dirección (A2) durará 10 seg y la verde (V2) 50 seg.



Haga el programa para implementar el semáforo descrito. Use leds con los colores apropiados, conectadolos al IMSAI a través de los relees. Calcule la interfase.

Todos los programas deben incluir diagrama de flujo y comentarios explicativos de lo que se está realizando

PRACTICA 7

OBJETIVOS

1) Mostrar y familiarizar al estudiante con la conexión de conversores D/A a los microprocesadores.

2) Ilustrar como es posible convertir información digital en analógica.

3) Generar distintas señales analógicas con ayuda del conversor D/A y el sistema IMSAI.

5) Mostrar y familiarizar al estudiante con los conversores A/D.

6) Mostrar al estudiante la implementación de un conversor A/D a partir de un conversor D/A y el sistema IMSAI.

7) Familiarizar al estudiante con la utilización de los analizadores lógicos.

PREPARACION

Leer en el CAPITULO V, sección 5.5., lo referente a conversión D/A y A/D.

Leer en el CAPITULO II, lo referente a la entrada Single Step($\overline{SS}$), del 8035.

Leer los manuales de los analizadores lógicos de que dispone el laboratorio (analizador lógico HP, analizador lógico TEKTRONIC y analizador lógico HP2100).

Leer en el manual correspondiente las especificaciones y características del conversor D/A que se le suministre.

PROCEDIMIENTO

I.

Conexión de un conversor D/A al sistema IMSAI

1) Diseñe un generador de señales usando el sistema IMSAI y un conversor D/A. El generador de señales debe cumplir las siguientes especificaciones:

a) Debe poder produciruna señal triangular, una señal diente de sierra y una señal sinusoidal, de acuerdo a un comando que se le dará por el teclado(A=triangular, B=diente de sierra y C=senoide).

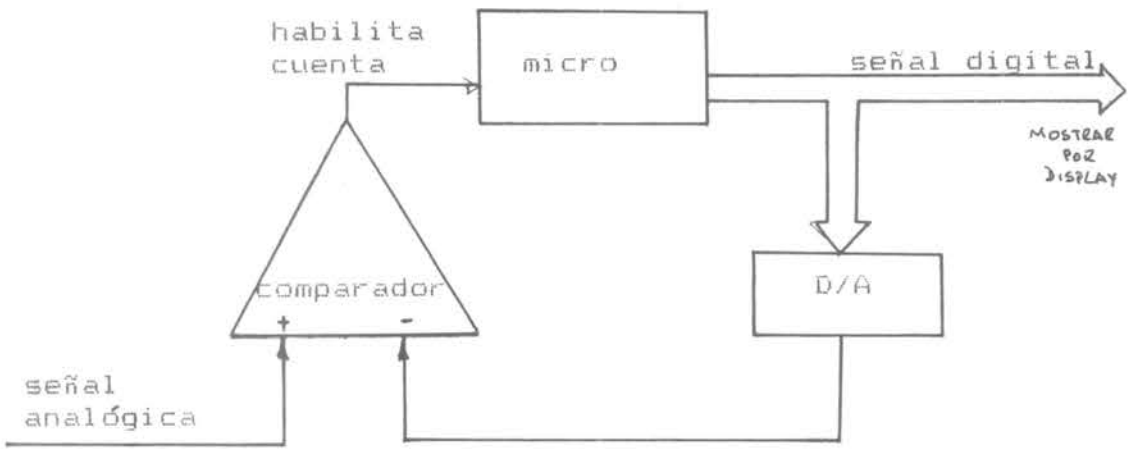
b) El conversor D/A debe ser conectado en el bus de datos (no en un puerto).

Haga el programa que implemente el generador de señales antes mencionado. Calcule la interfase y observe en el osciloscopio las señales generadas para demostrar que su programa funciona.

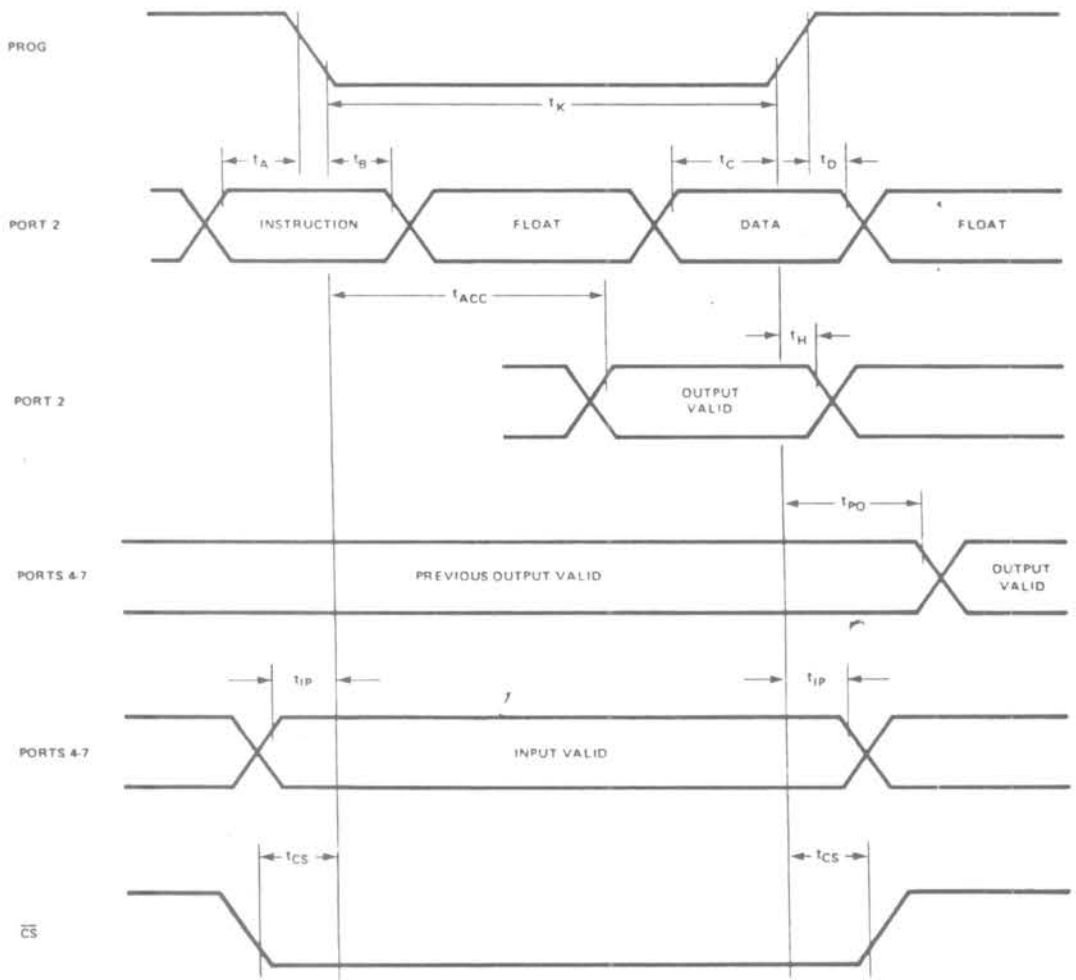
II.

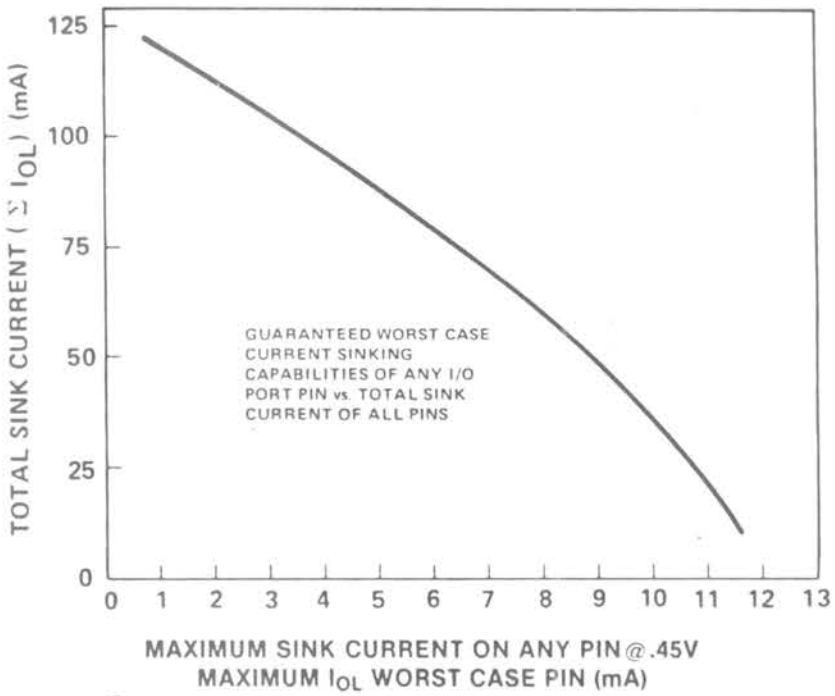
Conversión A/D

1) Construya un conversor A/D del tipo rampa, usando un conversor D/A, un comparador y el IMSAI, siguiendo el esquema de bloques que se muestra a continuación:



WAVEFORMS


PROG
t_K
t_A
t_B
t_C
t_D
PORT 2
INSTRUCTION
FLOAT
DATA
FLOAT
t_{ACC}
t_H
PORT 2
OUTPUT VALID
t_{PO}
PORTS 4-7
PREVIOUS OUTPUT VALID
OUTPUT VALID
t_{IP}
t_{IP}
PORTS 4-7
INPUT VALID
t_{CS}
t_{CS}
$\overline{CS}$



Sink Capability

The 8243 can sink 5 mA@ .45V on each of its 16 I/O lines simultaneously. If, however, all lines are not sinking simultaneously or all lines are not fully loaded, the drive capability of any individual line increases as is shown by the accompanying curve.

For example, if only 5 of the 16 lines are to sink current at one time, the curve shows that each of those 5 lines is capable of sinking 9 mA@ .45V (if any lines are to sink 9 mA the total I_{OL} must not exceed 45 mA or five 9 mA loads).

Example: How many pins can drive 5 TTL loads (1.6 mA) assuming remaining pins are unloaded?

$I_{OL} = 5 \times 1.6\ mA = 8\ mA$
$\varepsilon I_{OL} = 60$ mA from curve
pins = 60 mA ÷ 8 mA/pin = 7.5 = 7

In this case, 7 lines can sink 8 mA for a total of 56 mA. This leaves 4 mA sink current capability which can be divided in any way among the remaining 8 I/O lines of the 8243

Example: This example shows how the use of the 20 mA sink capability of Port 7 affects the sinking capability of the other I/O lines.

An 8243 will drive the following loads simultaneously

2 loads — 20 mA@ 1V (port 7 only)
8 loads — 4 mA@ .45V
6 loads — 3.2 mA@ .45V
Is this within the specified limits?

$\varepsilon I_{OL} = (2 \times 20) + (8 \times 4) + (6 \times 3.2) = 91.2$ mA.
From the curve: for $I_{OL} = 4$ mA, $\varepsilon I_{OL} \approx 93$ mA
since 91.2 mA < 93 mA the loads are within specified limits.

Although the 20 mA@ 1V loads are used in calculating εI_{OL}, it is the largest current required@ .45V which determines the maximum allowable εI_{OL}.

Note: A 10 to 50KΩ pullup resistor to +5V should be added to 8243 outputs when driving to 5V CMOS directly.

ABSOLUTE MAXIMUM RATINGS*

Ambient Temperature . . . . . 0°C to 70°C
Storage Temperature . . . . . -65°C to 125°C
Voltage on any Pin with
Respect to Ground . . . . . -0.5V to +7V
Power Dissipation . . . . . 1 Watt

*COMMENT: Stresses above those listed under "Absolute Maximum Ratings" may cause permanent damage to the device. This is a stress rating only and functional operation of the device at these or any other conditions above those indicated in the operational sections of this specification is not implied. Exposure to absolute maximum rating conditions for extended periods may affect device reliability.

PRELIMINARY

D.C. CHARACTERISTICS

$T_A = 0°C$ to $70°C$, $V_{SS} = 0V$, $V_{CC} = +5V \pm 5\%$, $V_{CC} = +5V \pm 10\%$ (8279-5)

Symbol	Parameter	Min.	Max.	Unit	Test Conditions
V_{IL1}	Input Low Voltage for Return Lines	-0.5	1.4	V	
V_{IL2}	Input Low Voltage for All Others	-0.5	0.8	V	
V_{IH1}	Input High Voltage for Return Lines	2.2		V	
V_{IH2}	Input High Voltage for All Others	2.0		V	
V_{OL}	Output Low Voltage		0.45	V	Note 1
V_{OH1}	Output High Voltage on Interrupt Line	3.5		V	Note 2
V_{OH2}	Other Outputs	2.4			
I_{IL1}	Input Current on Shift, Control and Return Lines		+10 -100	μA μA	$V_{IN} = V_{CC}$ $V_{IN} = 0V$
I_{IL2}	Input Leakage Current on All Others		±10	μA	$V_{IN} = V_{CC}$ to 0V
I_{OFL}	Output Float Leakage		±10	μA	$V_{OUT} = V_{CC}$ to 0V
I_{CC}	Power Supply Current		120	mA	

Notes:
8279, $I_{OL} = 1.6mA$; 8279-5, $I_{OL} = 2.2mA$.
8279, $I_{OH} = -100\mu A$; 8279-5, $I_{OH} = -400\mu A$.

CAPACITANCE

SYMBOL	TEST	TYP.	MAX.	UNIT	TEST CONDITIONS
C_{in}	Input Capacitance	5	10	pF	$V_{in} = V_{CC}$
C_{out}	Output Capacitance	10	20	pF	$V_{out} = V_{CC}$

A.C. CHARACTERISTICS

T_A = 0°C to 70°C, V_{SS} = 0V, (Note 1)

Bus Parameters

Read Cycle:

Symbol	Parameter	8279 Min.	8279 Max.	8279-5 Min.	8279-5 Max.	Unit
t_{AR}	Address Stable Before $\overline{READ}$	50		0		ns
t_{RA}	Address Hold Time for $\overline{READ}$	5		0		ns
t_{RR}	$\overline{READ}$ Pulse Width	420		250		ns
t_{RD}[2]	Data Delay from $\overline{READ}$		300		150	ns
t_{AD}[2]	Address to Data Valid		450		250	ns
t_{DF}	$\overline{READ}$ to Data Floating	10	100	10	100	ns
t_{RCY}	Read Cycle Time	1		1		μs

Write Cycle:

Symbol	Parameter	8279 Min.	8279 Max.	8279-5 Min.	8279-5 Max.	Unit
t_{AW}	Address Stable Before $\overline{WRITE}$	50		0		ns
t_{WA}	Address Hold Time for $\overline{WRITE}$	20		0		ns
t_{WW}	$\overline{WRITE}$ Pulse Width	400		250		ns
t_{DW}	Data Set Up Time for $\overline{WRITE}$	300		150		ns
t_{WD}	Data Hold Time for $\overline{WRITE}$	40		0		ns
t_{WCY}	Write Cycle Time	1		1		μs

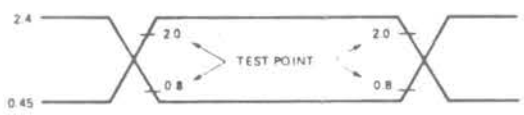
Notes:
1. 8279, V_{CC} = +5V ±5%; 8279-5, V_{CC} = +5V ±10%.
2. 8279, C_L = 100pF; 8279-5, C_L = 150pF.

Other Timings:

Symbol	Parameter	8279 Min.	8279 Max.	8279-5 Min.	8279-5 Max.	Unit
$t_{\phi W}$	Clock Pulse Width	230		120		nsec
t_{CY}	Clock Period	500		320		nsec

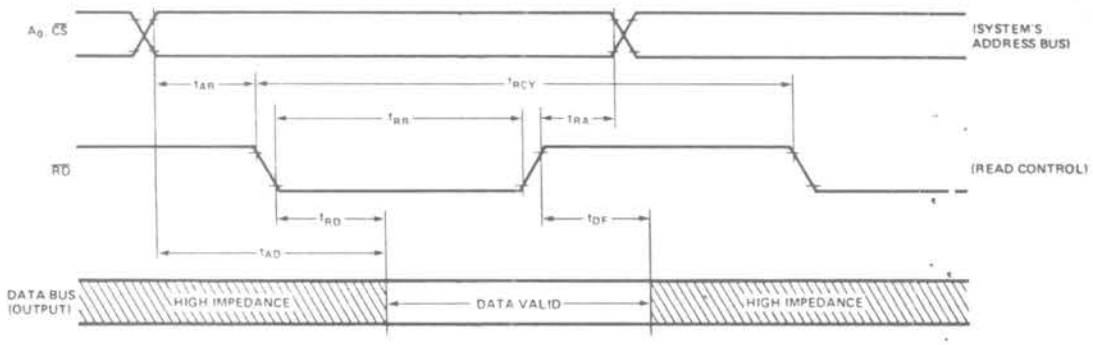
Keyboard Scan Time:	5.1 msec	Digit-on Time:	480 μsec
Keyboard Debounce Time:	10.3 msec	Blanking Time:	160 μsec
Key Scan Time:	80 μsec	Internal Clock Cycle:	10 μsec
Display Scan Time:	10.3 msec		

Input Waveforms For A.C. Tests

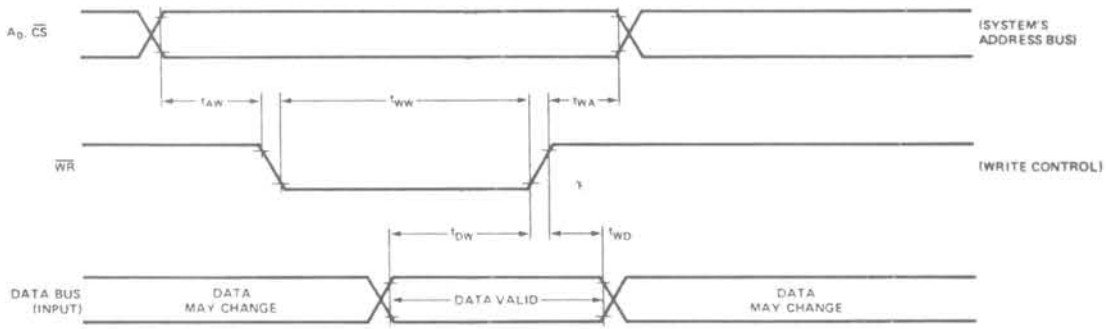


WAVEFORMS

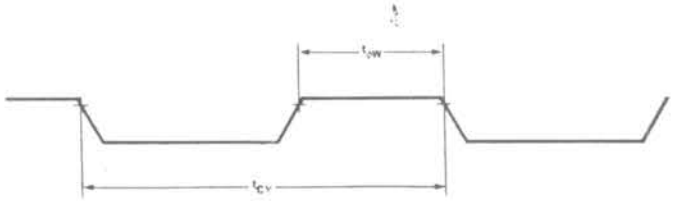
Read Operation



Write Operation

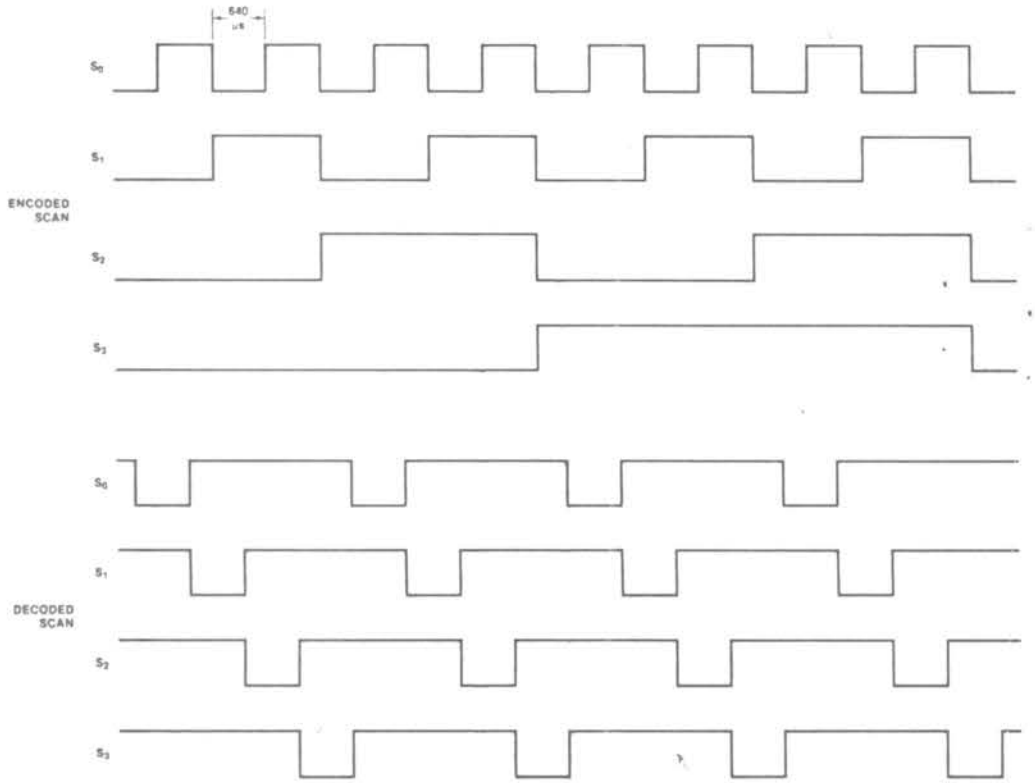


Clock Input

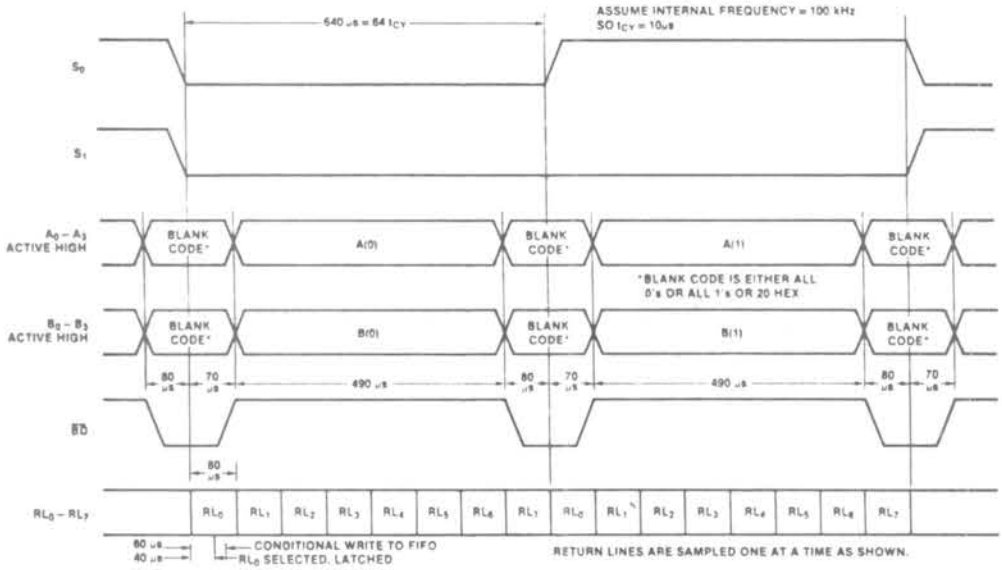


8279 SCAN TIMING

SCAN WAVEFORMS



DISPLAY WAVEFORMS



NOTE: SHOWN IS ENCODED SCAN LEFT ENTRY
S_2-S_3 ARE NOT SHOWN BUT THEY ARE SIMPLY S_1 DIVIDED BY 2 AND 4

00742A

8251A

ABSOLUTE MAXIMUM RATINGS*

Ambient Temperature Under Bias	0°C to 70°C
Storage Temperature	−65°C to +150°C
Voltage On Any Pin With Respect to Ground	−0.5V to +7V
Power Dissipation	1 Watt

*COMMENT: Stresses above those listed under "Absolute Maximum Ratings" may cause permanent damage to the device. This is a stress rating only and functional operation of the device at these or any other conditions above those indicated in the operational sections of this specification is not implied. Exposure to absolute maximum rating conditions for extended periods may affect device reliability.

D.C. CHARACTERISTICS

$T_A = 0°C$ to $70°C$; $V_{CC} = 5.0V \pm 5\%$; GND = 0V

Symbol	Parameter	Min.	Max.	Unit	Test Conditions
V_{IL}	Input Low Voltage	−0.5	0.8	V	
V_{IH}	Input High Voltage	2.2	V_{CC}	V	
V_{OL}	Output Low Voltage		0.45	V	$I_{OL} = 2.2$ mA
V_{OH}	Output High Voltage	2.4		V	$I_{OH} = -400\ \mu A$
I_{OFL}	Output Float Leakage		±10	μA	$V_{OUT} = V_{CC}$ TO 0.45V
I_{IL}	Input Leakage		±10	μA	$V_{IN} = V_{CC}$ TO 0.45V
I_{CC}	Power Supply Current		100	mA	All Outputs = High

CAPACITANCE

$T_A = 25°C$; V_{CC} = GND = 0V

Symbol	Parameter	Min.	Max.	Unit	Test Conditions
C_{IN}	Input Capacitance		10	pF	fc = 1MHz
$C_{I/O}$	I/O Capacitance		20	pF	Unmeasured pins returned to GND

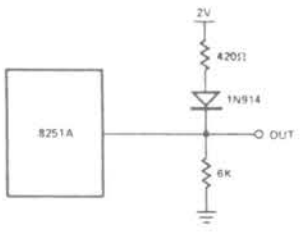


Figure 16. Test Load Circuit

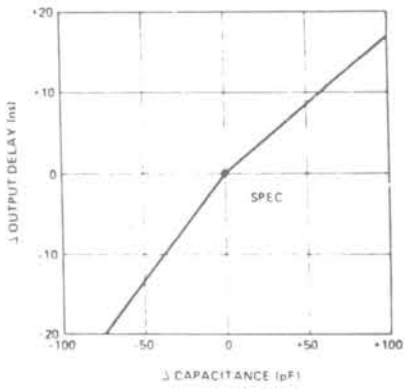


Figure 17. Typical Δ Output Delay vs. Δ Capacitance (pF)

8251A

A.C. CHARACTERISTICS

T_A = 0°C to 70°C; V_{CC} = 5.0V ±5%; GND = 0V

Bus Parameters (Note 1)

Read Cycle:

SYMBOL	PARAMETER	MIN.	MAX.	UNIT	TEST CONDITIONS
t_{AR}	Address Stable Before $\overline{READ}$ ($\overline{CS}$, C/$\overline{D}$)	50		ns	Note 2
t_{RA}	Address Hold Time for $\overline{READ}$ ($\overline{CS}$, C/$\overline{D}$)	50		ns	Note 2
t_{RR}	$\overline{READ}$ Pulse Width	250		ns	
t_{RD}	Data Delay from $\overline{READ}$		250	ns	3, C_L = 150 pF
t_{DF}	$\overline{READ}$ to Data Floating	10	100	ns	

Write Cycle:

SYMBOL	PARAMETER	MIN.	MAX.	UNIT	TEST CONDITIONS
t_{AW}	Address Stable Before $\overline{WRITE}$	50		ns	
t_{WA}	Address Hold Time for $\overline{WRITE}$	50		ns	
t_{WW}	$\overline{WRITE}$ Pulse Width	250		ns	
t_{DW}	Data Set Up Time for $\overline{WRITE}$	150		ns	
t_{WD}	Data Hold Time for $\overline{WRITE}$	30		ns	
t_{RV}	Recovery Time Between WRITES	6		t_{CY}	Note 4

NOTES: 1. AC timings measured V_{OH} = 2.0, V_{OL} = 0.8, and with load circuit of Figure 1.
2. Chip Select ($\overline{CS}$) and Command/Data (C/$\overline{D}$) are considered as Addresses.
3. Assumes that Address is valid before $\overline{RD}$↓.
4. This recovery time is for Mode Initialization only. Write Data is allowed only when TxRDY = 1.
Recovery Time between Writes for Asynchronous Mode is 8 t_{CY} and for Synchronous Mode is 16 t_{CY}.

Input Waveforms for AC Tests

8251A

Other Timings:

SYMBOL	PARAMETER	MIN.	MAX.	UNIT	TEST CONDITIONS
t_{CY}	Clock Period	320	1350	ns	Notes 5, 6
t_{ϕ}	Clock High Pulse Width	140	t_{CY-90}	ns	
$t_{\overline{\phi}}$	Clock Low Pulse Width	90		ns	
t_R, t_F	Clock Rise and Fall Time	5	20	ns	
t_{DTx}	TxD Delay from Falling Edge of $\overline{TxC}$		1	μs	
t_{SRx}	Rx Data Set-Up Time to Sampling Pulse	2		μs	
t_{HRx}	Rx Data Hold Time to Sampling Pulse	2		μs	
f_{Tx}	Transmitter Input Clock Frequency				
	1x Baud Rate	DC	64	kHz	
	16x Baud Rate	DC	310	kHz	
	64x Baud Rate	DC	615	kHz	
t_{TPW}	Transmitter Input Clock Pulse Width				
	1x Baud Rate	12		t_{CY}	
	16x and 64x Baud Rate	1		t_{CY}	
t_{TPD}	Transmitter Input Clock Pulse Delay				
	1x Baud Rate	15		t_{CY}	
	16x and 64x Baud Rate	3		t_{CY}	
f_{Rx}	Receiver Input Clock Frequency				
	1x Baud Rate	DC	64	kHz	
	16x Baud Rate	DC	310	kHz	
	64x Baud Rate	DC	615	kHz	
t_{RPW}	Receiver Input Clock Pulse Width				
	1x Baud Rate	12		t_{CY}	
	16x and 64x Baud Rate	1		t_{CY}	
t_{RPD}	Receiver Input Clock Pulse Delay				
	1x Baud Rate	15		t_{CY}	
	16x and 64x Baud Rate	3		t_{CY}	
t_{TxRDY}	TxRDY Pin Delay from Center of last Bit		8	t_{CY}	Note 7
$t_{TxRDY\ CLEAR}$	TxRDY ↓ from Leading Edge of $\overline{WR}$		180	ns	Note 7
t_{RxRDY}	RxRDY Pin Delay from Center of last Bit		24	t_{CY}	Note 7
$t_{RxRDY\ CLEAR}$	RxRDY ↓ from Leading Edge of $\overline{RD}$		150	ns	Note 7
t_{IS}	Internal SYNDET Delay from Rising Edge of $\overline{RxC}$		24	t_{CY}	Note 7
t_{ES}	External SYNDET Set-Up Time Before Falling Edge of $\overline{RxC}$		16	t_{CY}	Note 7
$t_{TxEMPTY}$	TxEMPTY Delay from Center of Last Bit		20	t_{CY}	Note 7
t_{WC}	Control Delay from Rising Edge of WRITE (TxEn, $\overline{DTR}$, $\overline{RTS}$)		8	t_{CY}	Note 7
t_{CR}	Control to READ Set-Up Time ($\overline{DSR}$, $\overline{CTS}$)		20	t_{CY}	Note 7

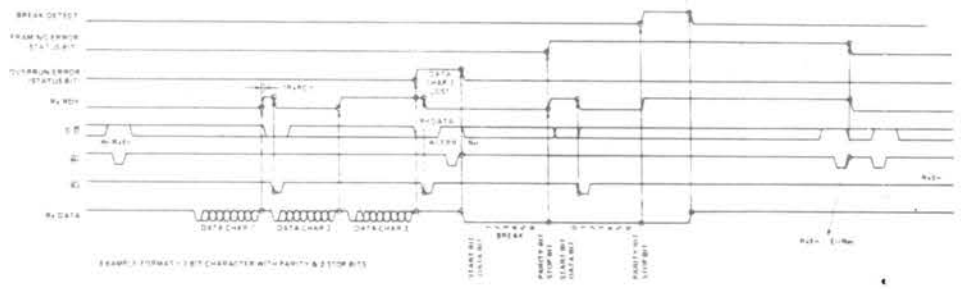
5. The TxC and RxC frequencies have the following limitations with respect to CLK
   For 1x Baud Rate, f_{Tx} or $f_{Rx} \leq 1/(30\ t_{CY})$
   For 16x and 64x Baud Rate, f_{Tx} or $f_{Rx} \leq 1/(4.5\ t_{CY})$

6. Reset Pulse Width = 6 t_{CY} minimum; System Clock must be running during Reset.

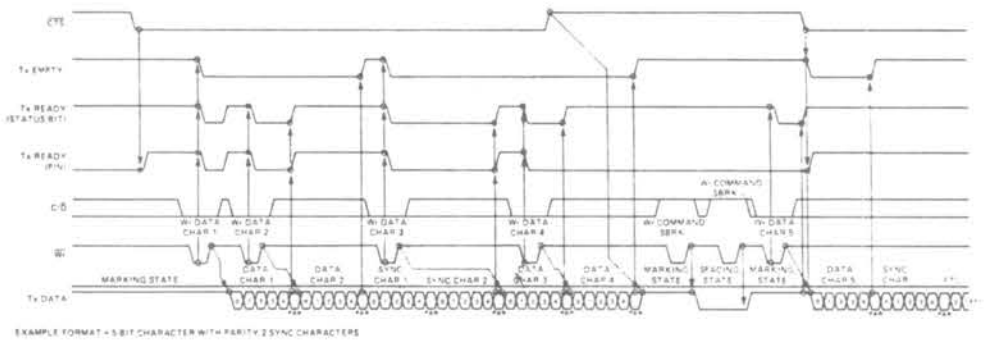
7. Status update can have a maximum delay of 28 clock periods from the event affecting the status.

8251A

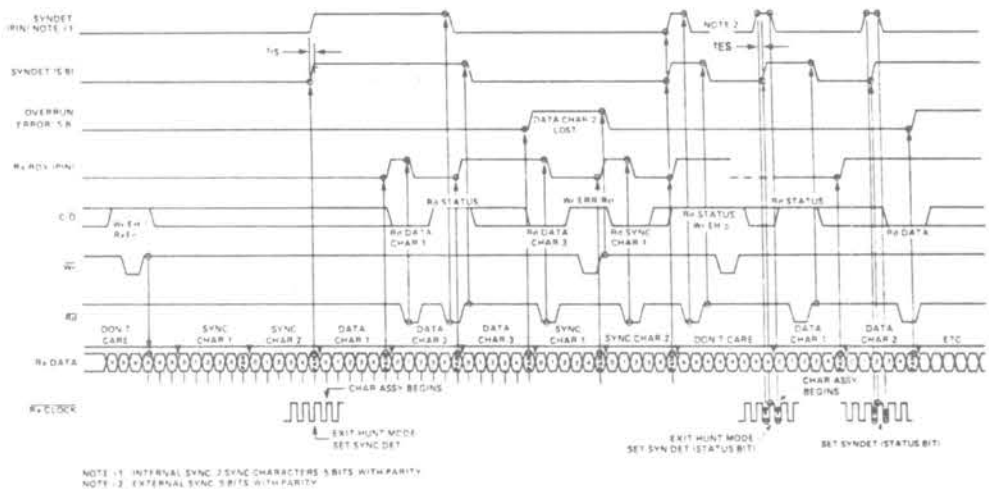
Receiver Control & Flag Timing (ASYNC Mode)



Transmitter Control & Flag Timing (SYNC Mode)



Receiver Control & Flag Timing (SYNC Mode)



ABSOLUTE MAXIMUM RATINGS*

Ambient Temperature Under Bias . . . . . 0°C to 70°C
Storage Temperature . . . . . −65°C to +150°C
Voltage on Any Pin
With Respect to Ground . . . . . −0.5V to +7V
Power Dissipation . . . . . 1 Watt

**COMMENT: Stresses above those listed under "Absolute Maximum Ratings" may cause permanent damage to the device. This is a stress rating only and functional operation of the device at these or any other conditions above those indicated in the operational sections of this specification is not implied. Exposure to absolute maximum rating conditions for extended periods may affect device reliability.*

D.C. CHARACTERISTICS

T_A = 0°C to 70°C, V_{CC} = +5V ±5%; GND = 0V

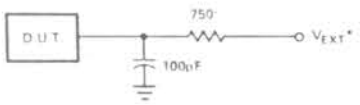
SYMBOL	PARAMETER	MIN.	MAX.	UNIT	TEST CONDITIONS
V_{IL}	Input Low Voltage	-0.5	0.8	V	
V_{IH}	Input High Voltage	2.0	V_{CC}	V	
V_{OL} (DB)	Output Low Voltage (Data Bus)		0.45	V	I_{OL} = 2.5mA
V_{OL}(PER)	Output Low Voltage (Peripheral Port)		0.45	V	I_{OL} = 1.7mA
V_{OH}(DB)	Output High Voltage (Data Bus)	2.4		V	I_{OH} = -400μA
V_{OH}(PER)	Output High Voltage (Peripheral Port)	2.4		V	I_{OH} = -200μA
I_{DAR}[1]	Darlington Drive Current	-1.0	-4.0	mA	R_{EXT} = 750Ω; V_{EXT} = 1.5V
I_{CC}	Power Supply Current		120	mA	
I_{IL}	Input Load Current		±10	μA	V_{IN} = V_{CC} to 0V
I_{OFL}	Output Float Leakage		±10	μA	V_{OUT} = V_{CC} to 0V

Note 1: Available on any 8 pins from Port B and C.

CAPACITANCE

T_A = 25°C; V_{CC} = GND = 0V

SYMBOL	PARAMETER	MIN.	TYP.	MAX.	UNIT	TEST CONDITIONS
C_{IN}	Input Capacitance			10	pF	fc = 1MHz
$C_{I/O}$	I/O Capacitance			20	pF	Unmeasured pins returned to GND



*V_{EXT} is set at various voltages during testing to guarantee the specification.

Figure 24. Test Load Circuit (for dB)

00744A

A.C. CHARACTERISTICS

T_A = 0°C to 70°C; V_{CC} = +5V ±5%; GND = 0V

NOTE:
The 8255A-5 specifications are not final. Some parametric limits are subject to change.

Bus Parameters

Read:

SYMBOL	PARAMETER	8255A		8255A-5		UNIT
		MIN.	MAX.	MIN.	MAX.	
t_{AR}	Address Stable Before READ	0		0		ns
t_{RA}	Address Stable After READ	0		0		ns
t_{RR}	READ Pulse Width	300		300		ns
t_{RD}	Data Valid From READ[1]		250		200	ns
t_{DF}	Data Float After READ	10	150	10	100	ns
t_{RV}	Time Between READs and/or WRITEs	850		850		ns

Write:

SYMBOL	PARAMETER	8255A		8255A-5		UNIT
		MIN.	MAX.	MIN.	MAX.	
t_{AW}	Address Stable Before WRITE	0		0		ns
t_{WA}	Address Stable After WRITE	20		20		ns
t_{WW}	WRITE Pulse Width	400		300		ns
t_{DW}	Data Valid to WRITE (T.E.)	100		100		ns
t_{WD}	Data Valid After WRITE	30		30		ns

Other Timings:

SYMBOL	PARAMETER	8255A		8255A-5		UNIT
		MIN.	MAX.	MIN.	MAX.	
t_{WB}	WR = 1 to Output[1]		350		350	ns
t_{IR}	Peripheral Data Before RD	0		0		ns
t_{HR}	Peripheral Data After RD	0		0		ns
t_{AK}	ACK Pulse Width	300		300		ns
t_{ST}	STB Pulse Width	500		500		ns
t_{PS}	Per. Data Before T.E. of STB	0		0		ns
t_{PH}	Per. Data After T.E. of STB	180		180		ns
t_{AD}	ACK = 0 to Output[1]		300		300	ns
t_{KD}	ACK = 1 to Output Float	20	250	20	250	ns
t_{WOB}	WR = 1 to OBF = 0[1]		650		650	ns
t_{AOB}	ACK = 0 to OBF = 1[1]		350		350	ns
t_{SIB}	STB = 0 to IBF = 1[1]		300		300	ns
t_{RIB}	RD = 1 to IBF = 0[1]		300		300	ns
t_{RIT}	RD = 0 to INTR = 0[1]		400		400	ns
t_{SIT}	STB = 1 to INTR = 1[1]		300		300	ns
t_{AIT}	ACK = 1 to INTR = 1[1]		350		350	ns
t_{WIT}	WR = 0 to INTR = 0[1]		850		850	ns

Notes: 1. Test Conditions: 8255A: C_L = 100pF; 8255A-5: C_L = 150pF.
2. Period of Reset pulse must be at least 50μs during or after power on. Subsequent Reset pulse can be 500 ns min.

00744A

Figure 25. Input Waveforms for A.C. Tests

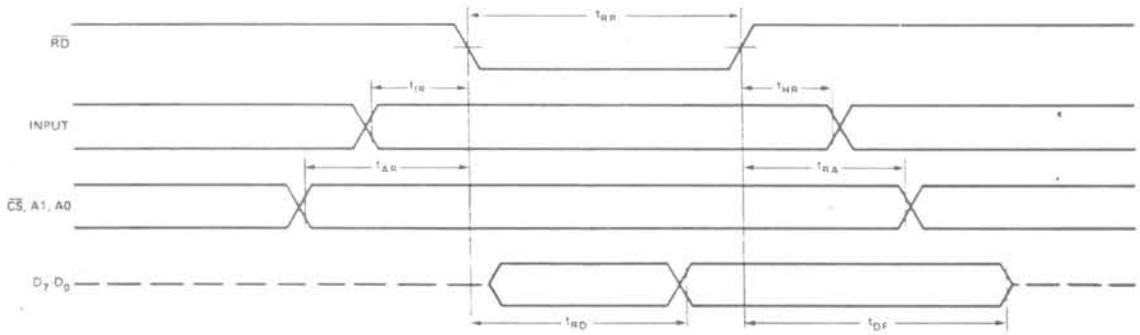


Figure 26. MODE 0 (Basic Input)

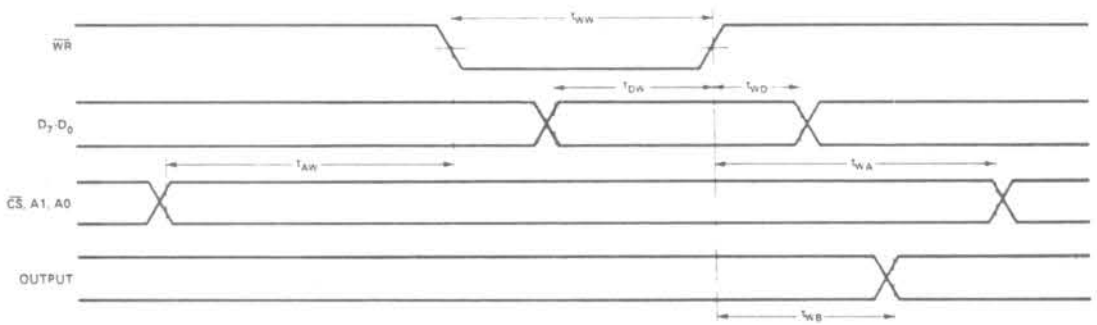


Figure 27. MODE 0 (Basic Output)

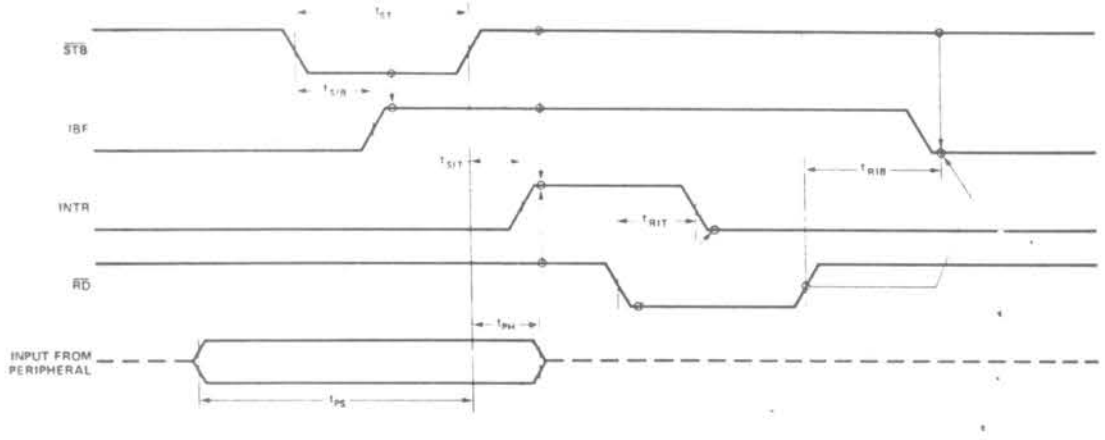


Figure 28. MODE 1 (Strobed Inut)

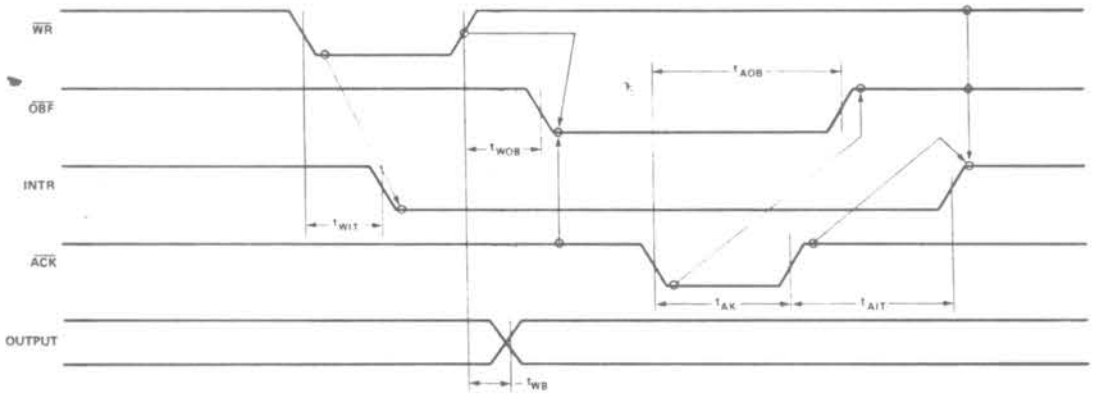


Figure 29. MODE 1 (Strobed Output)

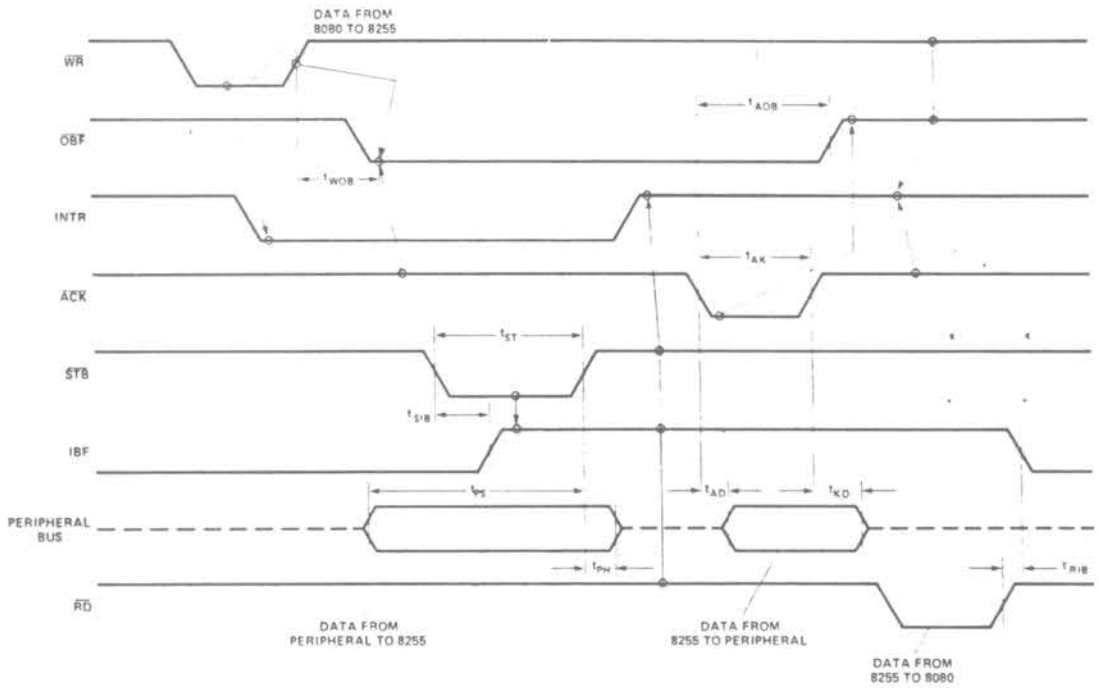


Figure 30. MODE 2 (Bidirectional)

NOTE: Any sequence where $\overline{WR}$ occurs before $\overline{ACK}$ and $\overline{STB}$ occurs before $\overline{RD}$ is permissible.
(INTR = IBF • $\overline{MASK}$ • $\overline{STB}$ • $\overline{RD}$ + $\overline{OBF}$ • $\overline{MASK}$ • $\overline{ACK}$ • $\overline{WR}$)

ELECTRICAL CHARACTERISTICS

ABSOLUTE MAXIMUM RATINGS

V_{GG} (with respect to V_{CC})	−20 to +0.3V
Clock and logic input voltages (with respect to V_{CC})	−20 to +0.3V
Storage Temperature	−65°C to 150°C
Operation Temperature	0°C to 70°C
Lead Temperature (Soldering, 10 sec)	330°

STANDARD TEST CONDITIONS

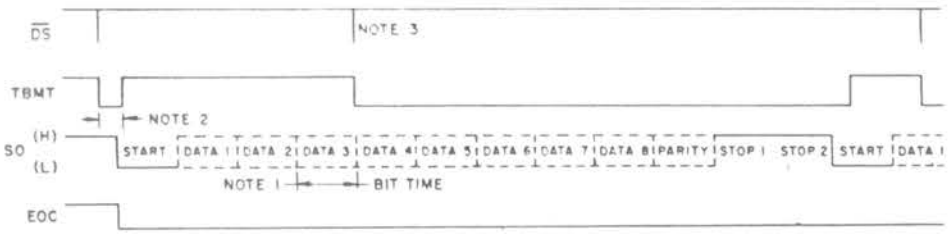
The following characteristics apply for any combination of the following test conditions, unless otherwise noted. All voltages are measured with respect to ground. Positive current is defined as flowing into the referenced pin.

$V_{GG} = -12V \pm 5\%$ $V_{CC} = 5V \pm 5\%$ $0°C < T_A < 70°C$

ELECTRICAL CHARACTERISTICS (see standard conditions)

PARAMETER	CONDITIONS AND COMMENTS	MIN.	TYP.	MAX.	UNITS
Input Logic Levels					
Logic 0	V_{IL} (I_{IL} = −1.6mA max.)	0	–	0.8	volts
Logic 1	V_{IH} Unit has internal pullup resistors	V_{CC}−1.5	–	V_{CC}+0.3	volts
Input Capacitance					
All Inputs	0 volts bias, f = 1MHz	–	–	20	pF
Leakage Currents					
Tri-State Outputs	0 volts	–	–	1.0	μA
Data Output Levels					
Logic 0	I_{OL} = 1.6mA (sink)	–	–	+0.4	volts
Logic 1	I_{OH} = −.3mA (source)	V_{CC}−1.0			volts
Output Capacitance		–	10	15	pF
Short Ckt. Current	See Fig. 24	–	–	–	–
Power Supply Current					
I_{GG} 25°C, all inputs +5V	See Fig. 26a	–	14	16	mA
I_{CC} 25°C, all inputs +5V	See Fig. 26b	–	18	20	mA
A.C. CHARACTERISTICS	T_A = 25°C, Output load capacitance 50pF max.				
Clock Frequency	AY-5-1013	DC	–	480	kHz
	AY-5-1013A	DC	–	640	kHz
Baud Rate	AY-5-1013	0	–	30	k baud
	AY-5-1013A	0	–	40	k baud
Pulse Width					
Clock Pulse	AY-5-1013 See Fig. 10	1.0	–	–	μs
	AY-5-1013A See Fig. 10	750	–	–	ns
Control Strobe	See Fig. 16	300	–	–	ns
Data Strobe	See Fig. 15	190	–	–	ns
External Reset	See Fig. 14	500	–	–	ns
Status Word Enable	See Fig. 22	500	–	–	ns
Reset Data Available	See Fig. 23	250	–	–	ns
Received Data Enable	See Fig. 22	500	–	–	ns
Set Up & Hold Time					
Input Data Bits	See Fig. 15	>0	–	–	ns
Input Control Bits	See Fig. 16	>0	–	–	ns
Output Propagation Delay					
TPD0	See Fig. 22 & 25		–	500	ns
TPD1	See Fig. 22 & 25		–	500	ns

FIGURE 6 UAR/T-TRANSMITTER TIMING



NOTE: SEE FIGURES 7, 8, 9 FOR DETAILS.

TRANSMITTER INITIALLY ASSUMED INACTIVE AT START OF DIAGRAM. SHOWN FOR 8 LEVEL CODE AND PARITY AND TWO STOPS.

1: BIT TIME = 16 CLOCK CYCLES

2: IF TRANSMITTER IS INACTIVE THE START PULSE WILL APPEAR ON LINE WITHIN 1 CLOCK CYCLE OF TIME DATA STROBE OCCURS. SEE DETAIL.

3: SINCE TRANSMITTER IS DOUBLE BUFFERED ANOTHER DATA STROBE CAN OCCUR ANYWHERE DURING TRANSMISSION OF CHARACTER 1 AFTER TBMT GOES HIGH.

DETAIL:

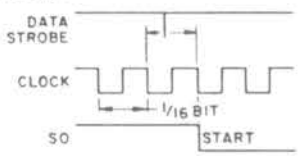


FIGURE 7 TRANSMITTER AT START BIT

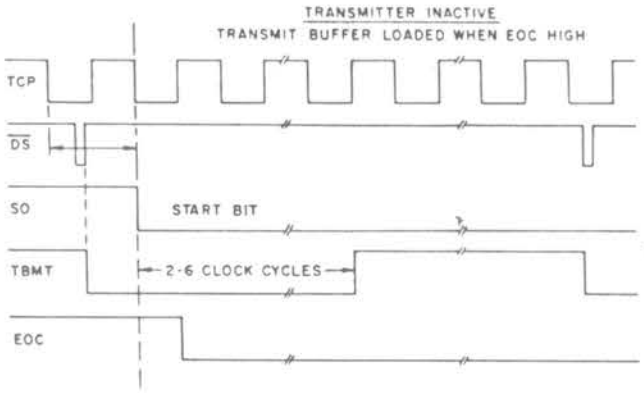


FIGURE 8 TRANSMITTER AT START BIT

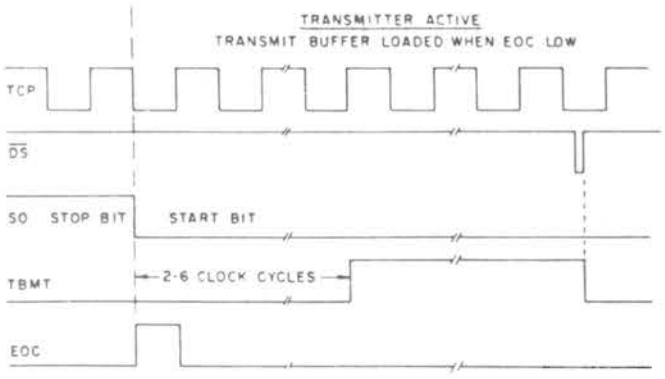


FIGURE 9 ALLOWABLE POINTS TO USE CONTROL STROBE

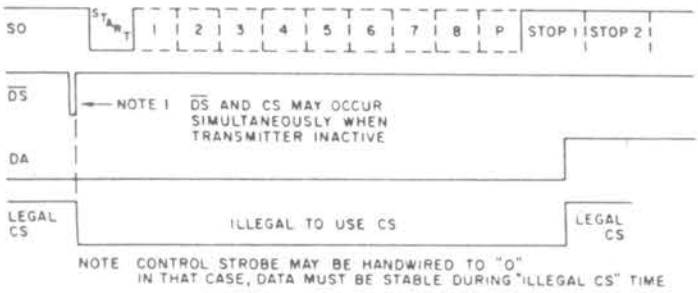


FIGURE 10 ALLOWABLE TCP, RCP

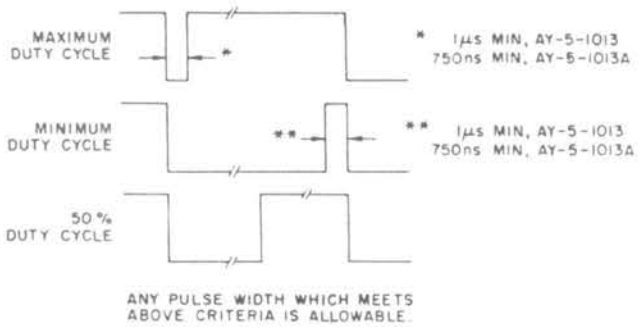


FIGURE 11 UAR/T–RECEIVER TIMING

NOTES:

1. THIS IS THE TIME WHEN THE ERROR CONDITIONS ARE OUTPARTED, IF ERROR OCCURS.
2. DATA AVAILABLE IS SET ONLY WHEN THE RECEIVED DATA, PE, FE, OR HAS BEEN TRANSFERRED TO THE HOLDING REGISTERS (SEE RECEIVER BLOCK DIAGRAM)
3. ALL INFORMATION IS GOOD IN HOLDING REGISTER UNTIL DATA AVAILABLE TRIES TO SET FOR NEXT CHARACTER.
4. ABOVE SHOWN FOR 8 LEVEL CODE PARITY AND TWO STOP. FOR NO PARITY, STOP BITS FOLLOW DATA.
5. FOR ALL LEVEL CODE THE DATA IN THE HOLDING REGISTER IS *RIGHT JUSTIFIED*, THAT IS, LSB ALWAYS APPEARS IN RD1 (PIN 12).

FIGURE 12

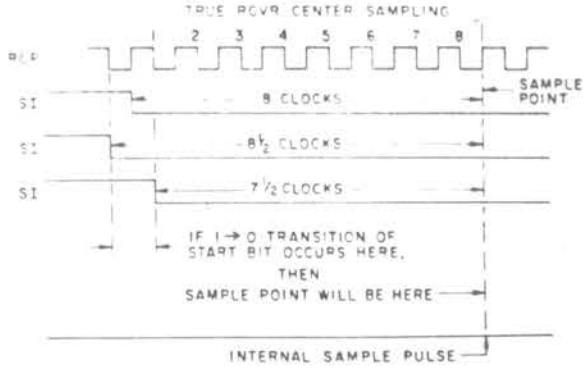


FIGURE 13 RECEIVER DURING 1st STOP BIT

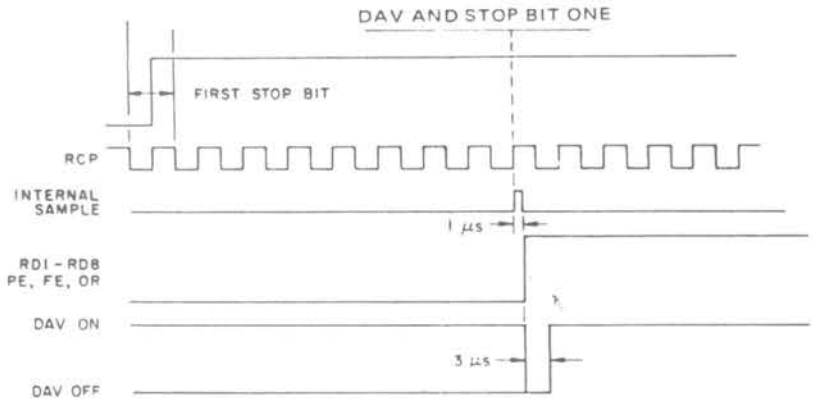
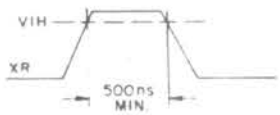


FIGURE 14 XR PULSE



WHEN NOT IN USE, XR MUST BE HELD AT GND.

XR RESETS EVERY REGISTER EXCEPT CONTROL REGISTER RECEIVED DATA SO, TBMT, EOC ARE RESET TO 5V ALL OTHER OUTPUTS RESET TO 0V.

FIGURE 15 $\overline{DS}$

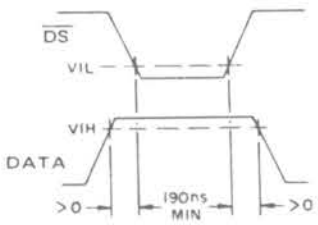
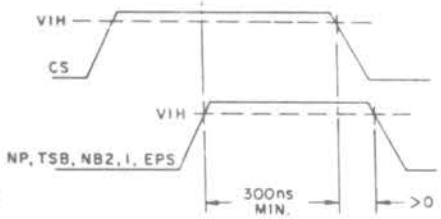
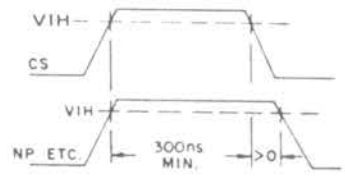


FIGURE 16a CS



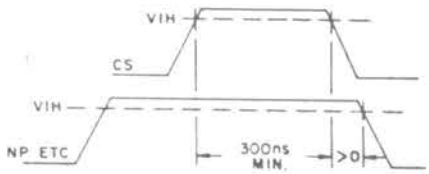
CONTROL BITS MUST BE STABLE FOR LAST 300ns OF CS.

FIGURE 16b



CONTROL STROBE AND CONTROL BITS MUST BE 300ns MINIMUM.

FIGURE 16c



LEADING EDGE OF DATA IS NOT CRITICAL AS LONG AS TRAILING EDGE AND PULSE WIDTH SPECS ARE OBSERVED.

FIGURE 17 SEROUT

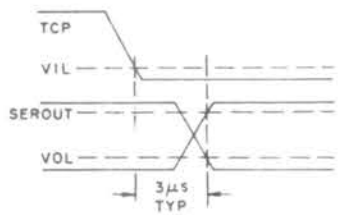


FIGURE 18 EOC TURN-ON

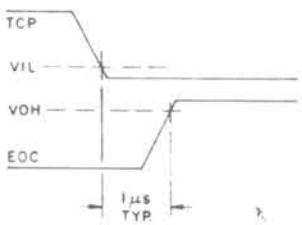


FIGURE 19 TBMT TURN-OFF

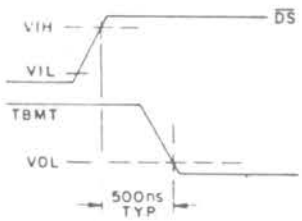


FIGURE 20 EOC TURN-OFF

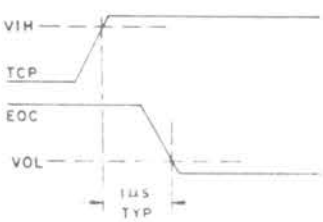
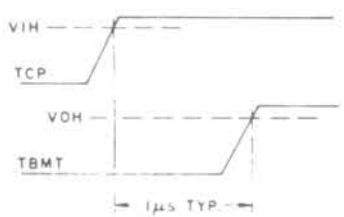


FIGURE 21 TBMT TURN-ON



Haga el programa que implemente el conversor; la señal convertida debe ser mostrada por el display.

Use los analizadores logicos TEKTRONIC y HP para ver la operacion de los conversores D/A y A/D.

Calcule la resolucion del conversor implementado y el tiempo maximo de conversion.

III)

Haga un programa y ejecutelo en la modalidad Single Step de que dispone el 8035. Que utilidad tiene el uso del single step?. Explique.

PRACTICA 8

OBJETIVOS

1) Familiarizar al estudiante con la comunicacion serial de datos.

2) Familiarizar al estudiante con el UART (Transmisor Receptor Universal Asincronico)

3) Familiarizar al estudiante con un sistema de desarrollo.

PREPARACION

Leer el en el CAPITULO VI:

a) Todo lo referente a la comunicacion serial de datos y la operacion del UART y el USART.

b) Todo lo referente a la interfase estandard RS-232C

c) Todo lo referente a la operacion y programacion del PPI 8255 de INTEL

Leer en el apendice D el codigo ASCII y el EBCDIC.

PROCEDIMIENTO

I.

UART en software

1) Conecte un teletipo al sistema IMSAI y transmita y reciba informacion de el, usando las rutinas que para tal efecto trae el monitor del IMSAI.

2) Haga la misma conexion del teletipo, pero en este caso no use las rutinas del monitor; en vez de eso, implemente un programa que implemente un UART en software.

II.

UART en hardware

1) Conecte en el bus de datos el UART AY-5-1013 y luego conecte a traves de el un teletipo al IMSAI. Haga las rutinas para el manejo del UART y calcule la interfase tomando en consideracion las especificaciones del AY-5-1013, que se dan en el apendice A.

2) Haga un programa que permita recibir y transmitir informacion al teletipo a traves del UART.

3) Compare ambos metodos (UART software, UART hardware).

III.

USART en hardware

1) Conecte el USART (Transmisor Receptor Universal Sincronico y Asincronico) 8251 de INTEL, al sistema IMSAI a traves del bus.

2) Haga las rutinas que permitan manejar el USART para transmitir informacion en forma asincronica a un teletipo. Cambie por software algunas de los parametros de la comunicacion (paridad, stop bits, factor de rata de baud) y demuestre la operacion correcta de su programa. Haga su programa versatil de manera de poder dar por el teclado los parametros de la comunicacion.

IV

Uso de ayudas

1) Ensamble los programas anteriores en la HP2100, y corra sus subrutinas en el simulador del 8035 que esta implementado en la misma.

2) Interconecte el IMSAI A la HP2100 y cargue el programa ensamblado en la HP al sistema IMSAI; corralo y verifique uue funciona manejando el teletipo con el programa que cargo desde la HP.

A P E N D I C E S

A P E N D I C E A

ABSOLUTE MAXIMUM RATINGS*

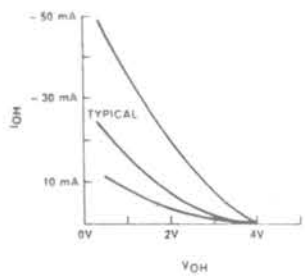
Ambient Temperature Under Bias	0°C to 70°C
Storage Temperature	-65°C to +150°C
Voltage On Any Pin With Respect to Ground	-0.5V to +7V
Power Dissipation	1.5 Watt

*COMMENT
Stresses above those listed under "Absolute Maximum Ratings" may cause permanent damage to the device. This is a stress rating only and functional operation of the device at these or any other conditions above those indicated in the operational sections of this specification is not implied.

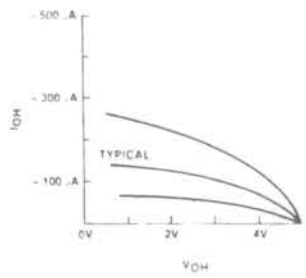
D.C. AND OPERATING CHARACTERISTICS $T_A = 0°C$ to $70°C$, $V_{CC} = V_{DD} = +5V \pm 10\%$, $V_{SS} = 0V$

Symbol	Parameter	Limits			Unit	Test Conditions
		Min.	Typ.	Max.		
V_{IL}	Input Low Voltage (All Except $\overline{RESET}$, X1, X2)	-.5		.8	V	
V_{IL1}	Input Low Voltage ($\overline{RESET}$, X1, X2)	-.5		.6	V	
V_{IH}	Input High Voltage (All Except XTAL1, XTAL 2, $\overline{RESET}$)	2.0		V_{CC}	V	
V_{IH1}	Input High Voltage (X1, X2, $\overline{RESET}$)	3.8		V_{CC}	V	
V_{OL}	Output Low Voltage (BUS)			.45	V	V_{OL} = 2.0 mA
V_{OL1}	Output Low Voltage ($\overline{RD}$, $\overline{WR}$, $\overline{PSEN}$, ALE)			.45	V	I_{OL} = 1.8 mA
V_{OL2}	Output Low Voltage (PROG)			.45	V	I_{OL} = 1.0 mA
V_{OL3}	Output Low Voltage (All Other Outputs)			.45	V	I_{OL} = 1.6 mA
V_{OH}	Output High Voltage (BUS)	2.4			V	$I_{OH} = -400\ \mu A$
V_{OH1}	Output High Voltage ($\overline{RD}$, $\overline{WR}$, $\overline{PSEN}$, ALE)	2.4			V	$I_{OH} = -100\ \mu A$
V_{OH2}	Output High Voltage (All Other Outputs)	2.4			V	$I_{OH} = -40\ \mu A$
I_{LI}	Input Leakage Current (T1, $\overline{INT}$)			±10	μA	$V_{SS} \leq V_{IN} \leq V_{CC}$
I_{LI1}	Input Leakage Current (P10-P17, P20-P27, EA, $\overline{SS}$)			-500	μA	$V_{SS} + .45 \leq V_{IN} \leq V_{CC}$
I_{LO}	Output Leakage Current (BUS, TO) (High Impedance State)			±10	μA	$V_{SS} + .45 \leq V_{IN} \leq V_{CC}$
I_{DD}	V_{DD} Supply Current		5	15	mA	
$I_{DD} + I_{CC}$	Total Supply Current		60	135	mA	

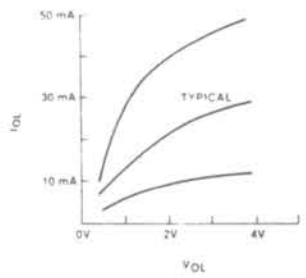
BUS



P1, P2

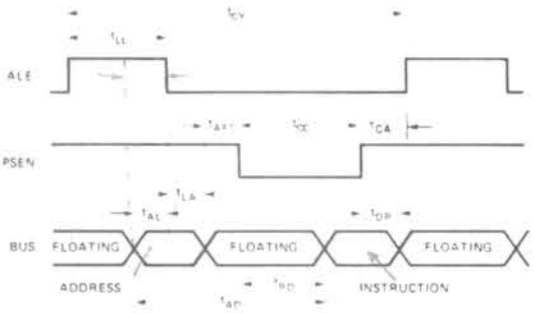


BUS, P1, P2

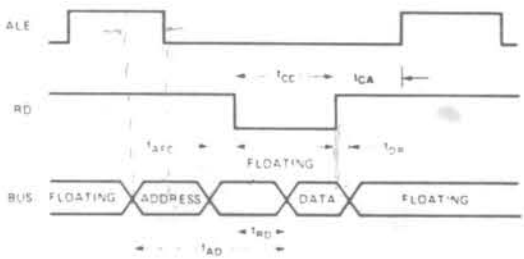


WAVEFORMS

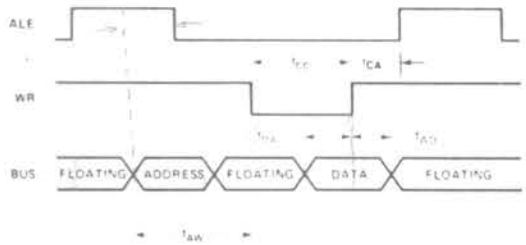
Instruction Fetch From External Program Memory



Read From External Data Memory



Write to External Data Memory



Input and Output Waveforms for A.C. Tests

A.C. CHARACTERISTICS T_A = 0°C to 70°C, V_{CC} = V_{DD} = +5V ±10% , V_{SS} = 0V

Symbol	Parameter	8048 8648 8748/8035/8035L		8748-8* 8035-8		Unit	Conditions (Note 1)
		Min.	Max.	Min.	Max.		
t_{LL}	ALE Pulse Width	400		600		ns	
t_{AL}	Address Setup to ALE	120		150		ns	
t_{LA}	Address Hold from ALE	80		80		ns	
t_{CC}	Control Pulse Width ($\overline{PSEN}$, $\overline{RD}$, $\overline{WR}$)	700		1500		ns	
t_{DW}	Data Setup before $\overline{WR}$	500		640		ns	
t_{WD}	Data Hold After $\overline{WR}$	120		120		ns	C_L = 20pF
t_{CY}	Cycle Time	2.5	15.0	4.17	15.0	μs	6 MHz XTAL = 2.5 (3.6 MHz XTAL for −8)
t_{DR}	Data Hold	0	200	0	200	ns	
t_{RD}	$\overline{PSEN}$, $\overline{RD}$ to Data In		500		750	ns	
t_{AW}	Address Setup to $\overline{WR}$	230		260		ns	
t_{AD}	Address Setup to Data In		950		1450	ns	
t_{AFC}	Address Float to $\overline{RD}$, $\overline{PSEN}$	0		0		ns	
t_{CA}	Control Pulse to ALE	10		20		ns	

Note 1 Control outputs C_L = 80 pF t_{CY} = 2.5 μs for standard parts
BUS Outputs C_L = 150 pF = 4.17 μs for -8 parts

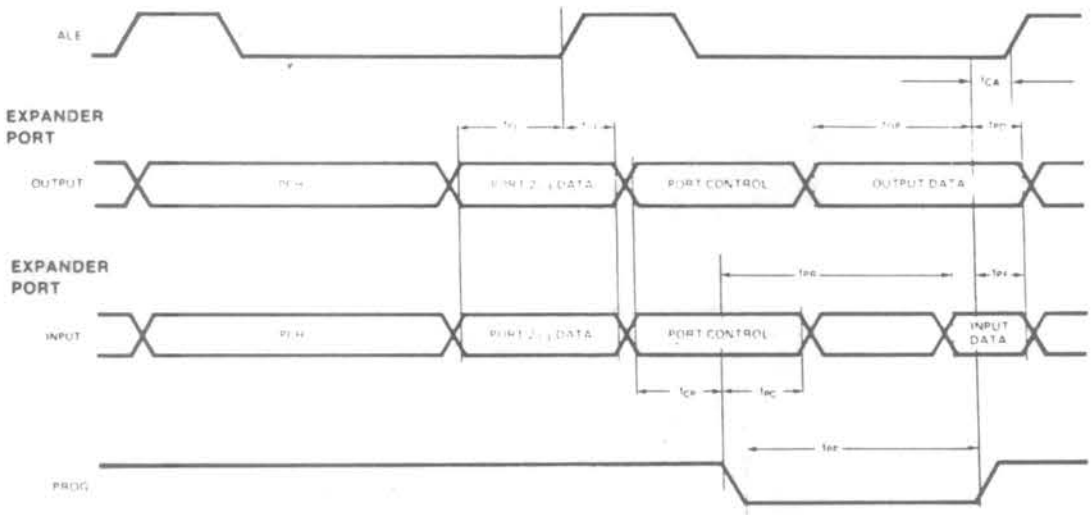
*V_{CC} and V_{DD} for 8748-8 and 8035-8 are ±5%

A.C. CHARACTERISTICS (PORT 2 TIMING)

T_A = 0°C to 70°C, V_{CC} = 5V±10%, V_{SS} = 0V

Symbol	Parameter	Min.	Max.	Unit	Test Conditions
t_{CP}	Port Control Setup Before Falling Edge of PROG	110		ns	
t_{PC}	Port Control Hold After Falling Edge of PROG	100		ns	
t_{PR}	PROG to Time P2 Input Must Be Valid		810	ns	
t_{PF}	Input Data Hold Time	0	150	ns	
t_{DP}	Output Data Setup Time	250		ns	
t_{PD}	Output Data Hold Time	65		ns	
t_{PP}	PROG Pulse Width	1200		ns	
t_{PL}	Port 2 I/O Data Setup	350		ns	
t_{LP}	Port 2 I/O Data Hold	150		ns	

PORT 2 TIMING



ABSOLUTE MAXIMUM RATINGS*

Ambient Temperature Under Bias	0°C to 70°C
Storage Temperature	-65°C to +150°C
Voltage on Any Pin	
With Respect to Ground	-0.5V to +7V
Power Dissipation	1 Watt

*COMMENT: Stresses above those listed under "Absolute Maximum Ratings" may cause permanent damage to the device. This is a stress rating only and functional operation of the device at these or any other conditions above those indicated in the operational sections of this specification is not implied. Exposure to absolute maximum rating conditions for extended periods may affect device reliability.

D.C. AND OPERATING CHARACTERISTICS

T_A = 0°C to 70°C, V_{CC} = 5V ±10%

SYMBOL	PARAMETER	MIN.	TYP.	MAX.	UNITS	TEST CONDITIONS
V_{IL}	Input Low Voltage	-0.5		0.8	V	
V_{IH}	Input High Voltage	2.0		V_{CC}+0.5	V	
V_{OL1}	Output Low Voltage Ports 4-7			0.45	V	I_{OL} = 5 mA*
V_{OL2}	Output Low Voltage Port 7			1	V	I_{OL} = 20 mA
V_{OH1}	Output High Voltage Ports 4-7	2.4			V	I_{OH} = 240µA
I_{IL1}	Input Leakage Ports 4-7	-10		20	µA	V_{in} = V_{CC} to 0V
I_{IL2}	Input Leakage Port 2, CS, PROG	-10		10	µA	V_{in} = V_{CC} to 0V
V_{OL3}	Output Low Voltage Port 2			45	V	I_{OL} = 0.6 mA
I_{CC}	V_{CC} Supply Current		10	20	mA	
V_{OH2}	Output Voltage Port 2	2.4				I_{OH} = 100µA
I_{OL}	Sum of all IO_L from 16 Outputs			80	mA	5 mA Each Pin

*See following graph for additional sink current capability

A.C. CHARACTERISTICS

T_A = 0°C to 70°C, V_{CC} = 5V ±10%

SYMBOL	PARAMETER	MIN.	MAX.	UNITS	TEST CONDITIONS
t_A	Code Valid Before PROG	100		ns	80 pF Load
t_B	Code Valid After PROG	60		ns	20 pF Load
t_C	Data Valid Before PROG	200		ns	80 pF Load
t_D	Data Valid After PROG	20		ns	20 pF Load
t_H	Floating After PROG	0	150	ns	20 pF Load
t_K	PROG Negative Pulse Width	700		ns	
t_{CS}	CS Valid Before/After PROG	50		ns	
t_{PO}	Ports 4-7 Valid After PROG		700	ns	100 pF Load
t_{LP1}	Ports 4-7 Valid Before/After PROG	100		ns	
t_{ACC}	Port 2 Valid After PROG		650	ns	80 pF Load

FIGURE 22 $\overline{RDE}$, $\overline{SWE}$

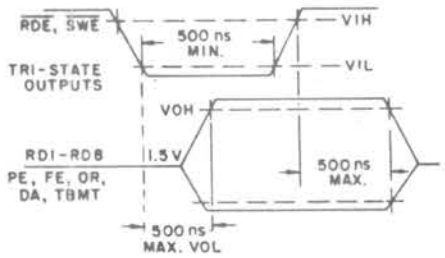


FIGURE 23 $\overline{RDAV}$

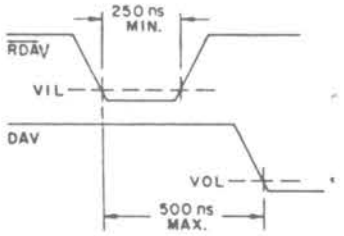


FIGURE 24 SHORT CIRCUIT OUTPUT CURRENT

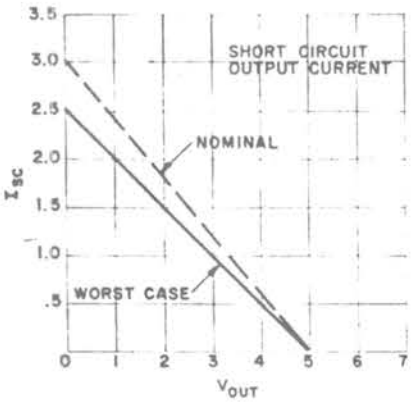


FIGURE 25 RD1–RD8, PE, FE, OR, TBMT, DAV

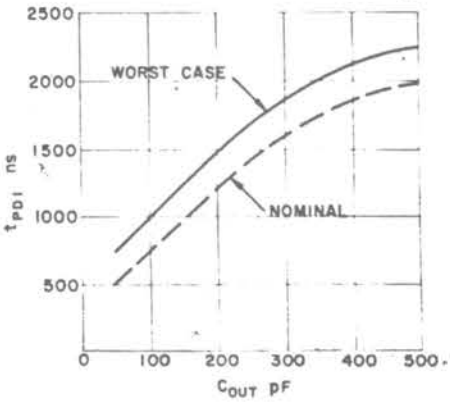


FIGURE 26a –12 VOLT SUPPLY CURRENT

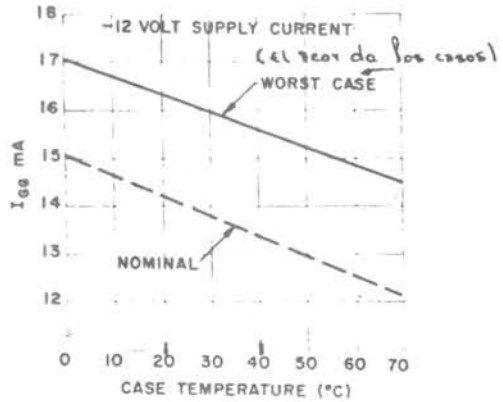
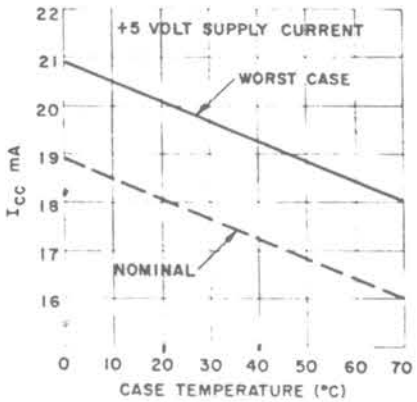


FIGURE 26b +5 VOLT SUPPLY CURRENT



A P E N D I C E B

PROGRAMA MONITOR DEL INSAI

```
PASS 1

PASS 2.

LINE ADDR INST SOURCE
 0001            ;*****************************************************
 0002            ;       IMP-48  IMSAI MONITOR PROGRAM
 0003            ;       TIMING SET FOR 8035-8
 0004            ;       COPYRIGHT IMSAI MANUFACTURING CORP
 0005            ;                 SAN LEANDRO, CALIFORNIA
 0006            ;                 AUGUST 24,1977
 0007            ;       ALL RIGHTS RESERVED
 0008            ;*****************************************************
 0009            ;       SYSTEM EQUATES
 0010            ;
 0011            ;       KEYPAD  NPUTS
 0012    = 0000 K0       =       0
 0013    = 0001 K1       =       1
 0014    = 0002 K2       =       2
 0015    = 0003 K3       =       3
 0016    = 0004 K4       =       4
 0017    = 0005 K5       =       5
 0018    = 0006 K6       =       6
 0019    = 0007 K7       =       7
 0020    = 0008 K8       =       8
 0021    = 0009 K9       =       9
 0022    = 000A KA       =       10
 0023    = 000B KB       =       11
 0024    = 000C KC       =       12
 0025    = 000D KD       =       13
 0026    = 000E KE       =       14
 0027    = 000F KF       =       15
 0028    = 0011 EXEC     =       11H     ;EXECUTE/BREAKPOINT FUNCTION
 0029    = 0012 EXAM     =       12H     ;EXAMINE/MODIFY FUNCTION
 0030    = 0013 PROGM    =       13H     ;PROGRAM MEMORY
 0031    = 0014 REGM     =       14H     ;REGISTER MEMORY
 0032    = 0015 NEXT     =       15H     ;NEXT MEMORY LOCATION
 0033    = 0016 ENTER    =       16H     ;ENTER FUNCTION
 0034    = 0017 CLEER    =       17H     ;CLEAR ENTRY
 0035    = 0000 TAPEO    =       0       ;CASETTE OUT
 0036    = 0001 TAPE     =       1       ;CASETTE IN
 0037    = 000F TTY      =       15      ;TELETYPE COMMANDS
 0038            ;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
 0039            ;       DISPLAY SEGMENT CODES
 0040            ;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
 0041            ;               A
 0042            ;            .......
 0043            ;       F   .       . B
 0044            ;           .   G   .
 0045            ;            .......
 0046            ;           .       .
 0047            ;         E .       . C
 0048            ;            .......    .
 0049            ;               D       P
 0050            ;
 0051            ; A '1' IN A SEGMENT POSITION CORRESPONDS TO LIGHT ON
 0052            ;
```

```
0054     = 003F D0      =                  00111111B       ;0
0055     = 0006 D1      =                  00000110B       ;1
0056     = 005B D2      =                  01011011B       ;2
0057     = 004F D3      =                  01001111B       ;3
0058     = 0066 D4      =                  01100110B       ;4
0059     = 006D D5      =                  01101101B       ;5
0060     = 007D D6      =                  01111101B       ;6
0061     = 0007 D7      =                  00000111B       ;7
0062     = 007F D8      =                  01111111B       ;8
0063     = 0067 D9      =                  01100111B       ;9
0064     = 0077 DAA     =                  01110111B       ;A
0065     = 007C DBB     =                  01111100B       ;B
0066     = 0039 DC      =                  00111001B       ;C
0067     = 005E DD      =                  01011110B       ;D
0068     = 0079 DE      =                  01111001B       ;E
0069     = 0071 DF      =                  01110001B       ;F
0070     = 0000 BLNK    =                  00000000B       ;BLANK
0071     = 0080 PERIOD  =                  10000000B       ;PERIOD
0072     = 0050 DR      =                  01010000B       ;R
0073     = 0041 EQSIGN  =                  01000001B       ;=
0074               ;
0075               ;*****************************************
0076               ;       8279 COMMANDS
0077               ;*****************************************
0078     = 0024 CLOCK   =       24H       ;PRESET VALUE=4 FOR 100KHZ SCAN
0079     = 0040 FIFORD  =       40H       ;READ FIFO NEXT INPUT
0080     = 00D0 CLEARD  =       0D0H      ;CLEAR DISPLAY TO BLANKS
0081     = 0090 DISP0   =       90H       ;DISPLAY AUTO INC FROM POS 0
0082     = 0091 DISP1   =       91H       ;DISPLAY POSITION 1
0083     = 0096 DISP6   =       96H       ;DISPLAY AUTO INC FROM POS 2
0084     = 0092 DISP2   =       92H       ;POSITION 2
0085     = 0095 DISP5   =       95H
0086     = 0007 FIFOCNT =       7         ;MASK FOR FIFO STATUS COUNT
0087               ;*****************************************************
0088               ;       SAVE AREA MEMORY MAP
0089               ;        FILLED UPON EXECUTION OF BREAKPOINT
0090               ;*************************************************
0091               ;LOCATION        CONTENTS
0092               ;FFF             8279 CONTROL PORT (MEMORY MAPPED)
0093               ;FFE             8279 DATA PORT (MEMORY MAPPED)
0094               ;FFD             R1 OF REGISTER BANK 1
0095               ;FFC             R0 OF REGISTER BANK 1
0096               ;FFB-FEC         STACK
0097               ;FEB             R7 OF RB0
0098               ;FEA             R6 OF RB0
0099               ;FE9             R5 OF RB0
0100               ;FE8             R4 OF RB0
0101               ;FE7             R3 OF RB0
0102               ;FE6             R2 OF RB0
0103               ;FE5             R1 IF RB0
0104               ;FE4             R0 OF RB0
0105               ;FE3             STACK POSITION PTR
0106               ;FE2             REMOVED BREAKPOINT BYTE
0107               ;FE1             BYTE FOLLOWING BREAKPOINT
0108               ;*************************************************
0109               ;
0110 0000                 ORG 0              ;RESET ENTRY POINT
0111 0000 15              DIS I              ;TURN OFF INTERRRUPT SYSTEM
0112 0001 0453            JMP INIT
```

```
0113                ;
0114 0003                   ORG 3               ;EXTERNAL INTERRUPT ENTRY POINT
0115 0003 15                DIS I
0116 0004 D5                SEL RB1
0117 0005 0409              JMP SAVE
0118 0007                   ORG 7               ;TIMER INTERRUPT ENTRY POINT
0119 0007 35                DIS TCNTI
0120 0008 93                RETR
0121 0009                   ORG 9
0122                ; SAVE INTERNAL REGISTERS IN EXTERNAL MEMORY
0123 0009 8A0F SAVE:        ORL P2,#15          ;BANKING
0124 000B AF                MOV R7,A            ;SAVE USER A REG
0125 000C F9                MOV A,R1
0126 000D B9FD              MOV R1,#0FDH        ;TOP OF REGISTER SAVE AREA
0127 000F 91                MOVX @R1,A          ;SAVE R0 AT TOP
0128 0010 2319              MOV A,#19H          ;# BYTES TO SAE
0129 0012 28                XCH A,R0
0130 0013 C9                DEC R1              ;NEXT SAVE LOCATION
0131 0014 91                MOVX @R1,A
0132 0015 C8   SAVE1:       DEC R0
0133 0016 F0                MOV A,@R0           ;GET INTERNAL RAM BYTE
0134 0017 91                MOVX @R1,A          ;SAVE IN EXTERNAL MEMORY
0135 0018 F8                MOV A,R0            ;GET COUNT
0136 0019 C9                DEC R1
0137 001A 9615              JNZ SAVE1           ;NOT DONE YET
0138 001C C7                MOV A,PSW           ;STACK PTR
0139 001D 91                MOVX @R1,A          ;SAVE
0140 001E 5442              CALL INITIO         ;INIT ALL I/O DEVICES AND CLOCK
0141 0020 23FC              MOV A,#DBB+PERIOD           ;"B."
0142 0022 541F              CALL OUTDSP         ;DISPLAY
0143                ; GET USERS RETURN ADDRESS FROM STACK
0144 0024 54D3              CALL FNDST          ;PT AT USER STACK IN EXT MEM
0145 0026 07                DEC A               ;BACK UP RETURN ADDRESS
0146 0027 07                DEC A               ;BACK UP 2 BYTES
0147 0028 A8                MOV R0,A            ;PC0-7
0148 0029 91                MOVX @R1,A          ;BACK IN STACK
0149 002A 0302              ADD A,#2            ;MUST WE BORROW
0150 002C 19                INC R1              ;PT AT PC8-11
0151 002D 81                MOVX A,@R1          ;GET IT
0152 002E AB                MOV R3,A            ;SAVE STATUS
0153 002F 530F              ANL A,#0FH          ;ONLY PC8-11
0154 0031 AD                MOV R5,A
0155 0032 07                DEC A
0156 0033 E637              JNC BR1
0157 0035 AD                MOV R5,A
0158 0036 91                MOVX @R1,A          ;BACK IN STACK
0159 0037 546F BR1:         CALL DISPAD         ;ADDRESS OF BREAKPOINT
0160 0039 8A0F              ORL P2,#0FH         ;BANK TO TOP
0161 003B B9E2              MOV R1,#0E2H        ;BREAK BYTE
0162 003D 81                MOVX A,@R1          ;GET BYTE
0163 003E AE                MOV R6,A            ;READY FOR OUTPUT
0164 003F C9                DEC R1              ;BYTE FOLLOWING BREAKPOINT PTR
0165 0040 81                MOVX A,@R1
0166 0041 AF                MOV R7,A            ;SAVE IT
0167 0042 5483              CALL PUTM           ;BREAK BYTE REINSERTED INLINE,BUM
0168 0044 FF                MOV A,R7
0169 0045 AE                MOV R6,A
0170 0046 5483              CALL PUTM           ;FOLLOWING BYTE REINSERTED
0171 0048 D5                SEL RB1
0172 0049 FF                MOV A,R7            ;USER A REGISTER
```

```
0174 004B 5468          CALL DISPRG     ;DISPLAY REGISTER
0175 004D FB            MOV A,R3        ;STATUS
0176 004E 47            SWAP A
0177 004F 5462          CALL DSG
0178 0051 0465          JMP KEY
0179                ;
0180 0053 54D3 INIT:    CALL FNDST      ;PT AT USER STACK
0181 0055 27            CLR A
0182 0056 91            MOVX @R1,A      ;DEFAULT RETURN ADDR=800H
0183 0057 2308          MOV A,#8
0184 0059 19            INC R1
0185 005A 91            MOVX @R1,A
0186 005B 3F            MOVD P7,A       ;INIT P7 TO OUTPUT 0
0187 005C 5442          CALL INITIO     ;INIT ALL I/O DEVICES AND CLOCK
0188 005E 0C            MOVD A,P4
0189 005F 0D            MOVD A,P5
0190 0060 0E            MOVD A,P6
0191                ;*************************************
0192                ;       COMMAND ENTRY POINT
0193                ;*************************************
0194 0061 5413 CMD:     CALL CLEAR
0195 0063 2341          MOV A,#EQSIGN
0196 0065 5427 KEY:     CALL KO         ;DISPLAY A THEN READ KEY
0197 0067 BC00          MOV R4,#0       ;EXT MEM FLAG
0198 0069 AE            MOV R6,A        ;SAVE
0199 006A 5413          CALL CLEAR      ;CLEAR DISPLAY
0200 006C FE            MOV A,R6        ;RESTORE INPUTTED KEY
0201 006D C682          JZ TAPEOUT      ;WRITE CASETTE/TELE
0202 006F 07            DEC A
0203 0070 C684          JZ TAPEIN       ;READ TAPE
0204 0072 03F1          ADD A,#-15      ;SKIP REST OF BOTTOM 2 ROWS
0205 0074 F288          JB7 CMDUD       ;COMMAND UNDEFINED
0206 0076 07            DEC A
0207 0077 C686          JZ EX           ;EXEC/BREAKPOINT FUNCTION
0208 0079 07            DEC A
0209 007A C694          JZ EXAMOD
0210 007C 03FD          ADD A,#-3
0211 007E F288          JB7 CMDUD       ;PROG/REG INVALID COMMANDS
0212 0080 0461          JMP CMD         ;IGNORE , . CLEAR
0213                ;
0214                ;
0215 0082 648A TAPEOUT: JMP TPOUT
0216 0084 246B TAPEIN: JMP TPIN
0217 0086 04F2 EX:      JMP EXBR
0218    = 0088 ERROR = $
0219 0088 5413 CMDUD: CALL CLEAR        ;CLEAR DISPLAY
0220 008A 2379          MOV A,#DE                       {E
0221 008C 541F          CALL OUTDSP
0222 008E 2350          MOV A,#DR       ;"R"            {r
0223 0090 541F          CALL OUTDSP
0224 0092 0465          JMP KEY
0225                ;
0226                ;
0227                ;
0228    = 0094 EXAMOD = $
0229 0094 AA            MOV R2,A        ;KA0-7=0
0230 0095 2379          MOV A,#DE
0231 0097 5427          CALL KO
0232 0099 03ED          ADD A,#-13H     ;PROG KEY?
```

```
 234 009D 07           DEC A             ;REG KEY
 235 009E 9688         JNZ ERROR
0236            ; REGISTER MEMORY KEY
0237 00A0 BC80         MOV R4,#80H       ;REG MEM FLAG
0238 00A2 23D0         MOV A,#DR+PERIOD  ;"R."
0239 00A4 BB00         MOV R3,#0         ;DEFAULT =0
0240 00A6 04AA         JMP EX3
0241            ;
0242 00A8 23F3 PMEM:   MOV A,#0F3H       ;"P."
0243 00AA 541F EX3:    CALL OUTDSP       ;POS 0 DISP
0244 00AC BB08         MOV R3,#08H       ;DEFAULT=800H
0245 00AE 54B4         CALL INPADR       ;INPUT ADDRESS
0246 00B0 FC   EX4:    MOV A,R4          ;MEM FLAG
0247 00B1 37           CPL A
0248 00B2 F2BA         JB7 EX1           ;VALID EXTERNAL RANGE
0249 00B4 FA           MOV A,R2          ;GET KA0-7
0250 00B5 53C0         ANL A,#0C0H
0251 00B7 4B           ORL A,R3          ;KA8-11
0252 00B8 9688         JNZ ERROR         ;INVALID RANGE>63
0253 00BA FA   EX1:    MOV A,R2
0254 00BB A8           MOV R0,A          ;PC0-7
0255 00BC FB           MOV A,R3
0256 00BD AD           MOV R5,A          ;PC8-11
0257 00BE 546F         CALL DISPAD
0258 00C0 5499         CALL GETM         ;GET MEMORY BYTE AT PC,USE FLAG
0259 00C2 AE           MOV R6,A
 260 00C3 5468 EX0:    CALL DISPRG
 261 00C5 14D9         CALL INBYTE       ;READ MODIFIER IF ANY
0262 00C7 F7           RLC A             ;GET FLAG FROM INBYTE
0263 00C8 AF           MOV R7,A          ;SAVE IT
0264 00C9 5483         CALL PUTM         ;STORE MODIFIED OR ORIG VALUE BAC
0265 00CB FF           MOV A,R7          ;RETRIEVE FLAG
0266 00CC 1261         JB0 CMD
0267            ; "NEXT " KEY HIT BUMP PC AND CONTINUE
0268 00CE 1A           INC R2
0269 00CF FA           MOV A,R2
0270 00D0 96B0         JNZ EX4           ;NO OVERFLOW IN PC8-11
0271 00D2 1B           INC R3            ;BUMP PC8-11
0272 00D3 FB           MOV A,R3
0273 00D4 530F         ANL A,#0FH        ;ONLY 4 BITS VALID
0274 00D6 AB           MOV R3,A
0275 00D7 04B0         JMP EX4
0276            ;*************************************************
0277            ;      INBYTE- GET DATA BYTE FROM KEYPAD AND DISPLAY
0278            ;*************************************************
0279 00D9 54E6 INBYTE: CALL GTVALD       ;GET VALID HEX DIGIT
0280 00DB F6F0         JC INBOUT
0281 00DD BE00         MOV R6,#0
0282 00DF 2E   INB1:   XCH A,R6
0283 00E0 47           SWAP A
0284 00E1 53F0         ANL A,#0F0H
0285 00E3 4E           ORL A,R6
 286 00E4 AE           MOV R6,A
 287 00E5 2396         MOV A,#DISP6
0288 00E7 5408         CALL CTR8279
0289 00E9 FE           MOV A,R6
0290 00EA 5468         CALL DISPRG
0291 00EC 54E6         CALL GTVALD
0292 00EE E6DF         JNC INB1          ;STILL VALID DIGIT
```

```
0294 00F1 83            RET
0295            ;
0296            ;*********************************************
0297            ;       EXECUTE AND BREAK FUNCTIONS
0298            ;*********************************************
0299 00F2 23D0 EXBR:    MOV A,#DR+PERIOD        ;"R,"
0300 00F4 541F          CALL OUTDSP
0301 00F6 27            CLR A
0302 00F7 AA            MOV R2,A
0303 00F8 AB            MOV R3,A        ;KA=0
0304 00F9 541F          CALL OUTDSP
0305 00FB 54D3          CALL FNDST      ;PT AT USER STACK RETUREN
0306 00FD A8            MOV R0,A        ;PC0-7 FOR RETURN
0307 00FE AA            MOV R2,A
0308 00FF 19            INC R1
0309 0100 81            MOVX A,@R1      ;PC8-11
0310 0101 530F          ANL A,#0FH
0311 0103 AD            MOV R5,A        ;PC8-11
0312 0104 AB            MOV R3,A        ;SET KA
0313 0105 54C9          CALL DISPKA     ;ADDRESS IN POS 2
0314 0107 54B4          CALL INPADR     ;GET RETURN ADDR FROM USER
0315 0109 F7            RLC A
0316 010A AE            MOV R6,A        ;SAVE FLAG
0317 010B FA            MOV A,R2        ;KA0-7
0318 010C A8            MOV R0,A        ;PC0-7
0319 010D FB            MOV A,R3
0320 010E AD            MOV R5,A        ;SET PC FROM KA
0321 010F 546F EXBR1:   CALL DISPAD     ;NEW RETURN ADDR IN POS 2
0322 0111 FE            MOV A,R6        ;GET INPADR FLAG
0323 0112 1251          JB0 RESTORE     ;NO BREAKPOINT
0324 0114 27            CLR A
0325 0115 AA            MOV R2,A
0326 0116 AB            MOV R3,A        ;KA=0 DEFAULT
0327 0117 54C9          CALL DISPKA
0328 0119 54D3          CALL FNDST      ;SET UP STACK FOR RETURN
0329 011B F8            MOV A,R0        ;PC0-7 FOR RETURN
0330 011C 91            MOVX @R1,A
0331 011D 19            INC R1
0332 011E 81            MOVX A,@R1      ;GET PC8-11 RETURN
0333 011F 53F0          ANL A,#0F0H     ;STATUS BITS
0334 0121 4D            ORL A,R5        ;RETURN ADDR 8-11
0335 0122 91            MOVX @R1,A      ;BACK IN STACK
0336 0123 FE            MOV A,R6        ;FLAGS FROM INPADR
0337 0124 1251          JB0 RESTORE     ;ENTER KEY
0338            ;
0339            ; MUST GET BREAKPOINT ADDR AND STUFF EN I THERE
0340            ;
0341 0126 2391          MOV A,#DISP1
0342 0128 5408          CALL CTR8279
0343 012A 23FC          MOV A,#DBB+PERIOD       ;"B."
0344 012C 541F          CALL OUTDSP
0345 012E 27            CLR A
0346 012F AA            MOV R2,A
0347 0130 AB            MOV R3,A
0348 0131 54B4          CALL INPADR     ;GET BREAK ADDR
0349 0133 FA            MOV A,R2
0350 0134 4B            ORL A,R3
0351 0135 C651          JZ RESTORE      ;NO ENTRY
0352 0137 FA            MOV A,R2
```

```
                        MOV R0,A        ;PC0-7
 0354 0139 F8           MOV A,R3
 0355 013A AD           MOV R5,A        ;PC8-11 FOR BREAKPOINT
0356 013B 5499          CALL GETM       ;GET IN CODE BYTE
0357 013D AF            MOV R7,A        ;SAVE
0358 013E 2305          MOV A,#05H      ;EN I INSTRUCTION CODE
0359 0140 90            MOVX @R0,A      ;STUFF
0360 0141 548A          CALL BUMPPC     ;PC=PC+1
0361 0143 5499          CALL GETM       ;GET SECOND BYTE FOR STORAGE
0362 0145 8A0F          ORL P2,#0FH
0363 0147 B9E1          MOV R1,#0E1H    ;PT AT SECOND BYTE
0364 0149 91            MOVX @R1,A      ;SAVE UP HIGH
0365 014A FF            MOV A,R7        ;RETRIEVE BREAK BYTE
0366 014B 19            INC R1          ;R1=E2H
0367 014C 91            MOVX @R1,A      ;SAVE IT
0368 014D BE05          MOV R6,#05      ;NOP INST
0369 014F 5483          CALL PUTM       ;STUFF IN LINE
0370 0151 8A0F RESTORE: ORL P2,#15      ;MEM BANK
0371 0153 D5            SEL RB1
0372 0154 B8FB          MOV R0,#0FBH    ;PT AT STACK TOP
0373 0156 B918          MOV R1,#18H     ;COUNTER AND INTERNAL RAM PTR
0374 0158 80    RES1:   MOVX A,@R0      ;GET EXT BYTE
0375 0159 C9            DEC R1
0376 015A A1            MOV @R1,A       ;STORE INTERNAL
0377 015B C8            DEC R0
0378 015C 19            INC R1
0379 015D E958          DJNZ R1,RES1
 380 015F 80            MOVX A,@R0      ;GET STACK PTR
 381 0160 D7            MOV PSW,A       ;RESTORE PSW
0382 0161 D5            SEL RB1
0383 0162 B8FC          MOV R0,#0FCH    ;R0 OF RB1
0384 0164 80            MOVX A,@R0      ;GET R1 OF RB0
0385 0165 A9            MOV R1,A        ;RESTORE IT
0386 0166 19            INC R1          ;PT AT R0 OF RB1
0387 0167 80            MOVX A,@R0
0388 0168 A8            MOV R0,A
0389 0169 FF            MOV A,R7        ;USERS A REG
0390 016A 93            RETR            ;RETURN TO USERS PROG
0391                ;
0392 016B 2330 TPIN:    MOV A,#30H      ;"I"
0393 016D 5427          CALL KO
0394 016F 03F1          ADD A,#-0FH
0395 0171 C697          JZ TTYIN        ;INPUT TAPE FROM TELETYPE
0396 0173 23B9          MOV A,#DC+PERIOD        ;"C."
0397 0175 34B6          CALL OC
0398 0177 BE00 SYNC:    MOV R6,#0       ;SYNC ON TAPE WITH 0E6H CHR
0399 0179 85            CLR F0
0400 017A 95    SYNC1:  CPL F0          ;SET NON-INVERT FLAG
0401 017B 7406          CALL BITIN      ;GET NEXT BIT
0402 017D D3E6          XRL A,#0E6H     ;EQU SYNC CHR?
0403 017F C685          JZ TPIN1        ;BRIF NON-INVERT SYNC
0404 0181 85            CLR F0          ;SET INVERT FLAG
0405 0182 37            CPL A           ;EQU INVERT SYNC CHR?
 406 0183 967A          JNZ SYNC1       ;BRIF NOT EQU
 407 0185 C5    TPIN1:  SEL RB0
0408 0186 7400          CALL CASIN      ;READ BYTE
0409 0188 34AF          CALL TAPINC     ;COMMON STUFF
0410 018A E985          DJNZ R1,TPIN1
0411 018C D5            SEL RB1
0412 018D E985          DJNZ R0,TPIN1
```

```
0414 0190 7400          CALL CASIN      ;CHECKSUM FROM TAPE
0415 0192 6F   CHECK:   ADD A,R7        ;IS CHECKSUM THE SAME?
0416 0193 96D1          JNZ ERR1                ;CHECKSUM ERROR
0417 0195 0461          JMP CMD
0418 0197 23F3 TTYIN:   MOV A,#0F3H     ;"P."
0419 0199 34B6          CALL OC
0420 019B 7469 TALE1:   CALL TELEIN     ;READ BYTE
0421 019D 17            INC A           ;CHECK FOR RECORD MARK FF
0422 019E 969B          JNZ TALE1       ;NOT YET
0423 01A0 C5   TALE2:   SEL RB0
0424 01A1 7469          CALL TELEIN     ;READ BYTE FROM TAPE
0425 01A3 34AF          CALL TAPINC     ;COMMON
0426 01A5 E9A0          DJNZ R1,TALE2
0427 01A7 D5            SEL RB1
0428 01A8 E8A0          DJNZ R0,TALE2
0429 01AA C5            SEL RB0
0430 01AB 7469          CALL TELEIN     ;READ CHECKSUM
0431 01AD 2492          JMP CHECK
0432                    ;
0433 01AF 2F   TAPINC:  XCH A,R7
0434 01B0 6F            ADD A,R7        ;ADD NEW BYTE TO OLD CHECKSUM
0435 01B1 2F            XCH A,R7        ;NEW CHECKSUM
0436 01B2 541F          CALL OUTDSP     ;OUTPUT TO DISPLAY
0437 01B4 4483          JMP PUTM        ;STORE IN MEMORY
0438                    ;
0439 01B6 541F OC:      CALL OUTDSP
0440                    ;
0441 01B8 27   CASCOM:  CLR A
0442 01B9 A8            MOV R0,A
0443 01BA AA            MOV R2,A
0444 01BB BD08          MOV R5,#8
0445 01BD 546F          CALL DISPAD
0446 01BF 54B4          CALL INPADR     ;GET END ADDRESS
0447 01C1 FA            MOV A,R2
0448 01C2 17            INC A
0449 01C3 A9            MOV R1,A
0450 01C4 FB            MOV A,R3
0451 01C5 03F8          ADD A,#-8
0452 01C7 F2D1          JB7 ERR1
0453 01C9 D5            SEL RB1
0454 01CA 17            INC A
0455 01CB A8            MOV R0,A
0456 01CC C5            SEL RB0
0457 01CD 27            CLR A
0458 01CE AF            MOV R7,A        ;CLEAR CHECKSUM
0459 01CF 6419          JMP TR2         ;INITIALIZE TIMER
0460                    ;
0461 01D1 0488 ERR1:    JMP ERROR
0462                    ;*****************************************************
0463                    ;       PRODUCTION TEST ROUTINES FOR TESTING MEMORY
0464                    ;       KEYPAD AND DISPLAYS FOR PROPER FUNCTIONING
0465                    ;*****************************************************
0466 01D3 5413 PRODTST: CALL CLEAR
0467 01D5 2308          MOV A,#8
0468 01D7 BA10 PROD1:   MOV R2,#16
0469 01D9 AB            MOV R3,A        ;SAVE
0470 01DA 5448          CALL DISPDG
0471 01DC FB            MOV A,R3
0472 01DD EAD9          DJNZ R2,$-4     ;FILL DISPLAY WITH SAME CHAR
```

```
0473 01DF 5434 PROD2:   CALL KEYINP      ;READ KEYPAD
 74 01E1 96D7           JNZ PROD1
 75                ;
0476 01E3 5413 MEMTEST: CALL CLEAR
0477 01E5 B8FD          MOV R0,#0FDH
0478 01E7 BD0F          MOV R5,#0FH      ;TOP OF VALID MEMORY
0479 01E9 5413 NEWAD:   CALL CLEAR
0480 01EB 5473          CALL DISPAD+4
0481 01ED 54AE          CALL BANKER      ;BANK P2 TO ADDRESS IN R0,R5
0482 01EF FA    LOOP:   MOV A,R2
0483 01F0 90            MOVX @R0,A       ;OUTPUT CHAR
0484 01F1 80            MOVX A,@R0       ;INPUT CHAR
0485 01F2 AB            MOV R3,A         ;SAVE
0486 01F3 DA            XRL A,R2         ;IS IT SAME?
0487 01F4 C6F8          JZ $+4
0488 01F6 64E8          JMP MEMERR
0489 01F8 EAEF          DJNZ R2,LOOP     ;TRY NEXT PATTERN
0490 01FA E8E9          DJNZ R0,NEWAD    ;BACK UP 1 AND CONTINUE
0491 01FC CD            DEC R5           ;PREVIOUS PAGE
0492 01FD 24E9          JMP NEWAD
0493                ;
0494 0200               PAGE
0495                ;
0496                ;*****************************************************
0497                ;       SYNCST:- GENERATES CONTINOUS CASETTE SYNC BYTES
0498                ;*****************************************************
0499 0200 55    SYNCST: STRT T
 500 0201 A5            CLR F1
 501 0202 BEE6          MOV R6,#0E6H
0502 0204 741F          CALL CASOUT
0503 0206 4402          JMP $-4
0504 0208 8A0F CTR8279: ORL P2,#15       ;BANK FOR 8279
0505 020A D5            SEL RB1
0506 020B B9FF          MOV R1,#0FFH     ;CONTROL PORT
0507 020D 91            MOVX @R1,A       ;OUTPUT COMMAND
0508 020E 81    CTR1:   MOVX A,@R1       ;READ STATUS
0509 020F F20E          JB7 CTR1         ;WAIT FOR DISPLAY AVAILABLE(FOR C
0510 0211 C5            SEL RB0
0511 0212 83            RET
0512                ;
0513 0213 23D0 CLEAR:   MOV A,#CLEARD    ;CLEAR DISPLAY
0514 0215 5408          CALL CTR8279
0515 0217 2308          MOV A,#8
0516 0219 5408          CALL CTR8279     ;16 DIG DISPLAY
0517 021B 2390          MOV A,#DISP0     ;DISPLAY POS 0
0518 021D 4408          JMP CTR8279
0519                ;
0520 021F D5    OUTDSP: SEL RB1
0521 0220 8A0F          ORL P2,#0FH      ;MEMORY MAP TP DISPLAY
0522 0222 B9FE          MOV R1,#0FEH     ;DATA PORT
0523 0224 91            MOVX @R1,A       ;OUTPUT TO DISPLAY
0524 0225 C5            SEL RB0
0525 0226 83            RET
 526                ;                                          0010
 527 0227 541F KO:      CALL OUTDSP
0528                ;
0529 0229 5434 KEYIN:   CALL KEYINP      ;READ KEY
0530 022B 03E9          ADD A,#-17H      ;CLEAR?
0531 022D 9631          JNZ KEY2         ;NO
0532 022F 0461          JMP CMD          ;YES
```

```
0534 0233 83            RET
0535 0234 2340 KEYINP:  MOV A,#FIFORD   ;READ FROM FIFO COMMAND
0536 0236 5408          CALL CTR8279
0537 0238 D5            SEL RB1
0538 0239 81    KEY1:   MOVX A,@R1      ;STATUS
0539 023A 5307          ANL A,#FIFOCNT  ;ANY KEY PRESSED YET?
0540 023C C639          JZ KEY1         ;NO
0541 023E C9            DEC R1          ;DATA PORT
0542 023F 81            MOVX A,@R1      ;READ KEY
0543 0240 C5            SEL RB0
0544 0241 83            RET
0545                    ;
0546 0242 2324 INITIO:  MOV A,#24H      ;CLOCK PRESET
0547 0244 5408          CALL CTR8279
0548 0246 4413          JMP CLEAR
0549                    ;
0550 0248 5462 DISPDG:  CALL DSG        ;GET DISPLAY CODE
0551 024A 441F DISPDG1: JMP OUTDSP      ;DISPLAY IT
0552                    ;
0553 024C 5462 DISPDGP: CALL DSG        ;GET DISPLAY CODE
0554 024E 4380          ORL A,#PERIOD   ;ADD PERIOD BIT
0555 0250 444A          JMP DISPDG1     ;GO DISPLAY IT
0556                    ;
0557                    ;
0558 0252 3F    DISPTB: D0
0559 0253 06            D1
0560 0254 5B            D2
0561 0255 4F            D3
0562 0256 66            D4
0563 0257 6D            D5
0564 0258 7D            D6
0565 0259 07            D7
0566 025A 7F            D8
0567 025B 67            D9
0568 025C 77            DAA
0569 025D 7C            DBB
0570 025E 39            DC
0571 025F 5E            DD
0572 0260 79            DE
0573 0261 71            DF
0574 0262 530F DSG:     ANL A,#0FH
0575 0264 0352          ADD A,#DISPTB
0576 0266 A3            MOVP A,@A
0577 0267 83            RET
0578                    ;
0579 0268 AF    DISPRG: MOV R7,A        ;DISPLAY A REGISTER
0580 0269 47            SWAP A          ;GET HIGH BYTE FIRST
0581 026A 5448          CALL DISPDG     ;DISPLAY DIGIT
0582 026C FF            MOV A,R7        ;RETRIEVE LOW NIBBLE
0583 026D 444C          JMP DISPDGP     ;DISPLAY WITH PERIOD
0584                    ;
0585    = 026F DISPAD   =       $       ;DISPLAY ADDRESS WITHIN BLANKS
0586 026F 2392          MOV A,#DISP2
0587 0271 5408          CALL CTR8279
0588 0273 FD            MOV A,R5        ;PC8-11
0589 0274 5448          CALL DISPDG     ;DISPLAY DIGIT
0590 0276 F8            MOV A,R0
0591 0277 5468          CALL DISPRG     ;DISPLAY REGISTER
0592 0279 27            CLR A
```

```
 0594 027B 91           MOVX @R1,A
 0595 027C 91           MOVX @R1,A
0596 027D 91           MOVX @R1,A
0597 027E 91           MOVX @R1,A          ;BLANK TO END OF DISPLAY
0598 027F 2396         MOV A,#DISP6
0599 0281 4408         JMP CTR8279
0600                ;
0601    = 0283 PUTM    =       $
0602 0283 FC           MOV A,R4            ;MEM FLAG
0603 0284 F290         JB7 PUTM1           ;REGISTER MEMORY
0604 0286 54AE PUTM3:  CALL BANKER         ;EXTERNAL MEM BANKING
0605 0288 FE           MOV A,R6            ;GET BYTE TO STORE
0606 0289 90           MOVX @R0,A          ;STORE IT
0607 028A 18   BUMPPC: INC R0
0608 028B F8           MOV A,R0
0609 028C 968F         JNZ $+3
0610 028E 1D           INC R5              ;IF CARRY NEEDED
0611 028F 83           RET
0612 0290 F8   PUTM1:  MOV A,R0            ;GET ADDR0-7
0613 0291 03E6         ADD A,#-1AH         ;WHERE IS IT
0614 0293 F286         JB7 PUTM3           ;IN EXTERNAL MEMORY
0615 0295 FE           MOV A,R6            ;GET BYTE
0616 0296 A0           MOV @R0,A           ;STORE INTERNAL
0617 0297 448A         JMP BUMPPC          ;BUMP PC AND RETURN
0618                ;
0619    = 0299 GETM    =       $
 0620 0299 FC           MOV A,R4            ;MEM FLAG
 0621 029A F2A0         JB7 GETM1           ;REG MEMORY
0622 029C 54AE GETM3:  CALL BANKER         ;BANK FOR EXTERNAL MEM
0623 029E 80           MOVX A,@R0
0624 029F 83           RET
0625 02A0 F8   GETM1:  MOV A,R0            ;ADDR0-7
0626 02A1 03E6         ADD A,#-1AH         ;WHERE IS IT
0627 02A3 F2A7         JB7 GETM2
0628 02A5 F0           MOV A,@R0           ;GET IT FROM INTERNAL MEMORY
0629 02A6 83           RET
0630 02A7 03FE GETM2:  ADD A,#-2           ;PT AT EXTERNAL ADDR
0631 02A9 A8           MOV R0,A
0632 02AA BD0F         MOV R5,#0FH
0633 02AC 449C         JMP GETM3
0634                ;
0635    = 02AE BANKER  = $                 ;BANK TO PC8-11,PC0-7 IN R0
0636 02AE 0A           IN A,P2             ;READ EXISTING P2
0637 02AF 53F0         ANL A,#0F0H         ;REMOVE BANK BITS ONLY
0638 02B1 4D           ORL A,R5            ;MERGE BANK BITS WITH P2 HIGH NIBB
0639 02B2 3A           OUTL P2,A           ;PERFORM BANKING
0640 02B3 83           RET
0641    = 02B4 INPADR = $
0642 02B4 54E6         CALL GTVALD         ;GET VALID HEX DATA
0643 02B6 F6C7         JC INPEND           ;TERM . OR ,
0644 02B8 2A           XCH A,R2            ;KA 0-7
0645 02B9 47           SWAP A
 646 02BA AB           MOV R3,A
 647 02BB 53F0         ANL A,#0F0H
0648 02BD 4A           ORL A,R2
0649 02BE AA           MOV R2,A            ;LEFT SHIFT KA0-11 AND MERGE NEW
0650 02BF 230F         MOV A,#0FH
0651 02C1 5B           ANL A,R3
0652 02C2 AB           MOV R3,A            ;KA8-11
```

```
                        CALL DISPKA     ;DISPLAY KA IN POS 5-7
0654 02C5 44B4          JMP INPADR      ;GET ANOTHER
0655 02C7 67    INPEND: RRC A           ;C=0="NEXT",C=1="ENTER"
0656 02C8 83            RET
0657 02C9 2396  DISPKA: MOV A,#DISP6
0658 02CB 5408          CALL CTR8279    ;POS 5 SETUP
0659 02CD FB            MOV A,R3        ;KA8-11
0660 02CE 5448          CALL DISPDG
0661 02D0 FA            MOV A,R2
0662 02D1 4468          JMP DISPRG
0663 02D3 8A0F  FNDST:  ORL P2,#15
0664 02D5 B9E3          MOV R1,#0E3H    ;STACK PTR PTR
0665 02D7 81            MOVX A,@R1      ;STACK PTR
0666 02D8 5307          ANL A,#7
0667 02DA 97            CLR C           ;FOR RLC A
0668 02DB 96DF          JNZ FNDST1      ;NO ROTATE STACK
0669 02DD 2308          MOV A,#8        ;TOP OF STACK
0670 02DF 07    FNDST1: DEC A
0671 02E0 F7            RLC A           ;*2BYTES/ENTRY
0672 02E1 03EC          ADD A,#0ECH     ;BASE PTR OF EXT STORAGE
0673 02E3 A9            MOV R1,A
0674 02E4 81            MOVX A,@R1      ;GET PC0-7
0675 02E5 83            RET
0676                ;
0677 02E6 5429  GTVALD: CALL KEYIN      ;READ KEYPAD
0678 02E8 03F0          ADD A,#-16
0679 02EA F2F0          JB7 GT1         ;VALID HEX DIGIT
0680 02EC 03F9          ADD A,#-7       ;SET CARRY AND A=FF IF ENTER,A=
0681 02EE A7            CPL C           ;SET CARRY FOR TERMINATE
0682 02EF 83            RET
0683 02F0 0310  GT1:    ADD A,#16       ;RETSTORE VALUE
0684 02F2 97            CLR C
0685 02F3 83            RET
0686                ;
0687 02F4 62    TEND1:  MOV T,A
0688 02F5 00            NOP
0689 02F6 00            NOP
0690 02F7 00            NOP
0691 02F8 00            NOP
0692 02F9 55            STRT T
0693 02FA 2308          MOV A,#8
0694 02FC 83            RET
0695                ;
0696 0300               PAGE
0697                ;************************************
0698                ;       CASETTE INPUT BYTE (TARBEL)
0699                ;****************************************
0700                ;
0701 0300 BA07  CASIN:  MOV R2,#7       ;BIT COUNT-1
0702 0302 7406          CALL BITIN
0703 0304 EA02          DJNZ R2,CASIN+2
0704 0306 160A  BITIN:  JTF BIT1        ;WAIT FOR TIMEOUT
0705 0308 6406          JMP BITIN
0706 030A 0E    BIT1:   MOVD A,P6       ;GET STATE OF LINE
0707 030B AB            MOV R3,A        ;SAVE IT
0708 030C 47            SWAP A          ;PUT IN BIT 7
0709 030D F7            RLC A           ;INTO CARRY
0710 030E B611          JF0 NOCOMP      ;BRIF NON-INVERT
0711 0310 A7            CPL C           ;INVERT
0712 0311 FE    NOCOMP: MOV A,R6        ;GET ACCUMULATED BYTE
```

```
0714 0313 AE            MOV R6,A        ;SAVE BYTE
 15 0314 0E    TR1:     MOVD A,P6       ;WAIT FOR TRANSITION
0716 0315 37            CPL A           ;INVERT BITS
0717 0316 DB            XRL A,R3        ;TRANSITION YET?
0718 0317 7214          JB3 TR1         ;NO
0719 0319 23FD TR2:     MOV A,#-3       ;INITIALIZE TIMER
0720 031B 7449          CALL TIMEND
0721 031D FE            MOV A,R6        ;GET BYTE BACK
0722 031E 83            RET
0723                ;
0724                ;***********************************************
0725                ;       CASETTE OUTPUT BYTE (TARBEL)
0726                ;***********************************************
0727 031F BA08 CASOUT:  MOV R2,#8       ;#BITS
0728 0321 FE            MOV A,R6
0729 0322 67            RRC A
0730 0323 AE            MOV R6,A        ;SAVE BYTE
0731 0324 7443          CALL TIMEOUT
0732 0326 F632          JC CASOUT1
0733 0328 07            DEC A
0734 0329 9F            ANLD P7,A
0735 032A 7439          CALL PTIMEOUT
0736 032C 8F            ORLD P7,A       ;FORCE HIGH
0737 032D FE    CASOUT2: MOV A,R6
0738 032E EA22          DJNZ R2,CASOUT+3 ;GET NEXT BIT
0739 0330 67            RRC A           ;RESTORE A
 740 0331 83            RET
 741 0332 8F    CASOUT1: ORLD P7,A      ;FORCE HIGH
0742 0333 7439          CALL PTIMEOUT
0743 0335 07            DEC A           ;A=7
0744 0336 9F            ANLD P7,A       ;FORCE LOW
0745 0337 642D          JMP CASOUT2
0746                ;
0747 0339 FA    PTIMEOUT: MOV A,R2
0748 033A 07            DEC A
0749 033B 9643          JNZ TIMEOUT
0750 033D 7641          JF1 PTIME1
0751 033F 6443          JMP TIMEOUT
0752 0341 74E1 PTIME1:  CALL IO1
0753                ;
0754 0343 23FE TIMEOUT: MOV A,#-2       ;
0755 0345 1649          JTF TIMEND
0756 0347 6445          JMP TIMEOUT+2
0757 0349 44F4 TIMEND:  JMP TEND1
0758                ;
0759                ;***********************************************
0760                ;       TELETYPE OUTPUT DRIVER 110 BAUD
0761                ;***********************************************
0762 034B AE    TELEOUT: MOV R6,A       ;SAVE
0763 034C BA0B          MOV R2,#11      ;BIT COUNT
0764 034E 97            CLR C           ;FOR START BIT
0765 034F 1653 TELE1:   JTF TELE2
 766 0351 644F          JMP $-2 ;WAIT FOR TIMEOUT
 767 0353 E65A TELE2:   JNC BITOUT1     ;OUTPUT 0 BIT
0768 0355 230B          MOV A,#0BH      ;OUTPUT INVERTED 1 BIT
0769 0357 9F            ANLD P7,A       ;PORT 7 BIT 2
0770 0358 645D          JMP BITOUT3
0771 035A 2304 BITOUT1: MOV A,#4
0772 035C 8F            ORLD P7,A
```

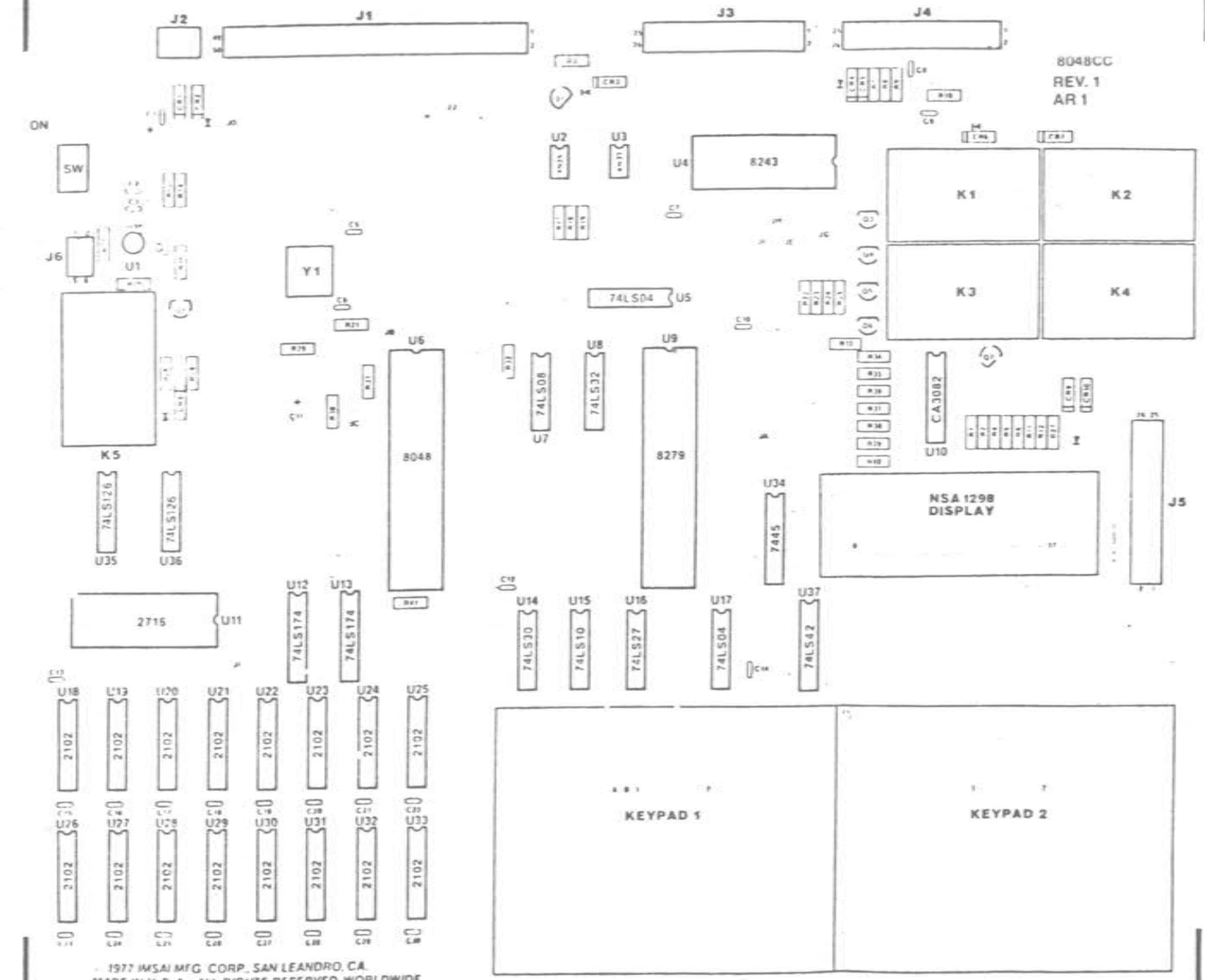
```
0774 035E A7            CPL C           ;FOR STOP BITS
0775 035F FE            MOV A,R6
0776 0360 67            RRC A
0777 0361 AE            MOV R6,A
0778 0362 23BD          MOV A,#-67      ;FOR TIMER FOR 9040 USEC DELAY
0779 0364 7449          CALL TIMEND     ;SET TIMER AND START
0780 0366 EA4F          DJNZ R2,TELE1   ;GET NEXT BIT IF ANY
0781 0368 83            RET
0782                ;*********************************************
0783                ;       TELETYPE INPUT ROUTINE 110 BAUD
0784                ;*********************************************
0785 0369 BA08 TELEIN: MOV R2,#8        ;8 BITS/BYTE
0786 036B 0E    T1:     MOVD A,P6       ;READ LINE
0787 036C 526B          JB2 T1
0788 036E 23DE          MOV A,#-34      ;HALF BIT TIME
0789 0370 7445          CALL TIMEOUT+2  ;START CLOCK
0790 0372 23BD          MOV A,#-67      ;
0791 0374 7445          CALL TIMEOUT+2
0792 0376 0E            MOVD A,P6       ;READ LINE
0793 0377 526B          JB2 T1          ;FALSE START BITE
0794 0379 23BD T3:      MOV A,#-67      ;
0795 037B 7445          CALL TIMEOUT+2  ;1 BIT IMTE
0796 037D 0E            MOVD A,P6
0797 037E 67            RRC A
0798 037F 67            RRC A
0799 0380 67            RRC A    ;BIT IN CARRY
0800 0381 FE            MOV A,R6
0801 0382 67            RRC A           ;MERGE NEW BIT
0802 0383 AE            MOV R6,A        ;SAVE
0803 0384 EA79          DJNZ R2,T3      ;GET NEXT BIT
0804 0386 7443          CALL TIMEOUT    ;WAIT FOR STOP BIT
0805 0388 FE            MOV A,R6        ;RETRIVE BYTE
0806 0389 83            RET
0807                ;
0808                ;*****************************************************
0809                ;       TAPEOUT- OUTPUT BINARY TAPE TO CASETTE OR PTAPE
0810                ;*****************************************************
0811 038A 233F TPOUT:   MOV A,#03FH     ;"O"
0812 038C 5427          CALL KO
0813 038E 03F1          ADD A,#-0FH     ;IS IT TTY COMMAND?
0814 0390 C6C8          JZ TTYOUT       ;YES
0815 0392 1C            INC R4          ;R4=1=CASETTE,R4=0=TELETYPE
0816                ; CASETTE OUTPUT HERE
0817 0393 23B9          MOV A,#DC+PERIOD        ;"C."
0818 0395 34B6          CALL OC
0819 0397 BB05          MOV R3,#5
0820 0399 27    CT2:    CLR A
0821 039A AE            MOV R6,A
0822 039B 741F          CALL CASOUT
0823 039D EF99          DJNZ R7,CT2     ;256 ZEROS OUT
0824 039F EB99          DJNZ R3,CT2     ;5 TIMES 256 ZEROS ABOUT 7 SECS
0825 03A1 BEC3          MOV R6,#0C3H
0826 03A3 741F          CALL CASOUT     ;START BYTE FOR BIT SYNC
0827 03A5 BEE6          MOV R6,#0E6H
0828 03A7 B5            CPL F1
0829 03A8 741F          CALL CASOUT     ;SYNC BYTE
0830 03AA C5    CT1:    SEL RB0
0831 03AB 548A          CALL BUMPPC
0832 03AD 74DB          CALL ICOUT      ;OUTPUT TO  DEVICE
```

```
0833 03AF E9AA          DJNZ R1,CT1
  34 03B1 D5            SEL RB1
  35 03B2 E8AA          DJNZ R0,CT1     ;DONE YET
0836 03B4 C5            SEL RB0
0837 03B5 FF            MOV A,R7        ;YES,GET CHECKSUM
0838 03B6 37            CPL A
0839 03B7 17            INC A           ;NEGATE CHECKSUM
0840 03B8 6E            ADD A,R6        ;CORRECT CHECKSUM
0841 03B9 AE            MOV R6,A
0842 03BA 74DB          CALL IOOUT
0843 03BC BB28          MOV R3,#40      ;TRAILER FOR PAPER TAPE
0844 03BE 27            CLR A
0845 03BF AE            MOV R6,A
0846 03C0 74DB          CALL IOOUT
0847 03C2 EBBE          DJNZ R3,$-4
0848 03C4 65            STOP T
0849 03C5 A5            CLR F1
0850 03C6 0461          JMP CMD         ;RETURN TO NEW COMMAND
0851                ;
0852 03C8 23F3 TTYOUT:  MOV A,#0F3H     ;"P." FOR PAPER TAPE
0853 03CA 34B6          CALL OC
0854 03CC BF28          MOV R7,#40      ;LEADER
0855 03CE 27            CLR A
0856 03CF 744B          CALL TELEOUT
0857 03D1 EFCE          DJNZ R7,$-3
0858 03D3 23FF          MOV A,#0FFH     ;RECORD MARK
0859 03D5 744B          CALL TELEOUT
  60 03D7 74E1          CALL IO1
  61 03D9 64AA          JMP CT1         ;OUTPUT RECORD
0862                ;
0863 03DB FC    IOOUT:  MOV A,R4        ;GET FLAG
0864 03DC 121F          JB0 CASOUT
0865 03DE FE            MOV A,R6
0866 03DF 744B          CALL TELEOUT
0867 03E1 5499 IO1:     CALL GETM
0868 03E3 2F            XCH A,R7
0869 03E4 6F            ADD A,R7
0870 03E5 2F            XCH A,R7
0871 03E6 AE            MOV R6,A
0872 03E7 83            RET
0873 03E8 2394 MEMERR:  MOV A,#94H      ;POS 4
0874 03EA 5408          CALL CTR8279
0875 03EC FA            MOV A,R2        ;SHOULD BE DATA
0876 03ED 5468          CALL DISPRG
0877 03EF FB            MOV A,R3
0878 03F0 5468          CALL DISPRG
0879 03F2 5429 RELAY:   CALL KEYIN
0880 03F4 3D            MOVD P5,A
0881 03F5 64F2          JMP RELAY
0882 03FF               ORG 3FFH
0883 03FF 3F            MOVD P7,A       ;PROVIDES FOR PAGE SWITCHING
0884
```

D ASSEMBLY

A P E N D I C E C

DIAGRAMA ELECTRICO Y DE EMSAMBLAJE DEL IMSAI



A P E N D I C E D

TABLA DE CODIGOS ASCII Y EBCDIC

MNEMONICO	USASCII	EBCDIC	MNEMONICO	USASCII	EBCDIC
NUL	00	00	@	40	7C
SOH	01	01	A	41	C1
STX	02	02	B	42	C2
ETX	03	03	C	43	C3
EOT	04	37	D	44	C4
ENQ	05	2D	E	45	C5
ACK	06	2E	F	46	C6
BEL	07	2F	G	47	C7
BS	08	16	H	48	C8
HT	09	05	I	49	C9
LF	0A	25	J	4A	D1
VT	0B	0B	K	4B	D2
FF	0C	0C	L	4C	D3
CR	0D	0D	M	4D	D4
SO	0E	0E	N	4E	D5
SI	0F	0F	O	4F	D6
DLE	10	10	P	50	D7
DC1	11	11	Q	51	D8
DC2	12	12	R	52	D9
DC3	13	13	S	53	E2
DC4	14	3C	T	54	E3
NAK	15	3D	U	55	E4
SYN	16	32	V	56	E5
ETB	17	26	W	57	E6
CAN	18	18	X	58	E7
EM	19	19	Y	59	E8
SUB	1A	3F	Z	5A	E9
ESC	1B	27	[	5B	4A
FS	1C	1C	\	5C	E0
GS	1D	1D	]	5D	5A
RS	1E	1E	^	5E	5F
US	1F	1F	_	5F	6D
SP	20	40	`	60	79
!	21	4F	a	61	81
"	22	7F	b	62	82
#	23	7B	c	63	83
$	24	5B	d	64	84
%	25	6C	e	65	85
&	26	50	f	66	86
'	27	7D	g	67	87
(	28	4D	h	68	88
)	29	5D	i	69	89
*	2A	5C	j	6A	91
+	2B	4E	k	6B	92
,	2C	6B	l	6C	93
-	2D	60	m	6D	94
.	2E	4B	n	6E	95
/	2F	61	o	6F	96
0	30	F0	p	70	97
1	31	F1	q	71	98
2	32	F2	r	72	99
3	33	F3	s	73	A2
4	34	F4	t	74	A3
5	35	F5	u	75	A4
6	36	F6	v	76	A5
7	37	F7	w	77	A6
8	38	F8	x	78	A7
9	39	F9	y	79	A8
:	3A	7A	z	7A	A9

B I B L I O G R A F I A

s, Michael, Programing Microprocesor Interfaces for l and Instrumentation, Prentice Hill inc, Englewood Cliffs 1982,Pp.361

, Joseph J., Microcomputer Interfacing Handbook: A/D & D/A, Books Inc, Pensislvania,1980, Pp.350

arr, Joseph J., Microprocesor Interfasing, Tab Books Inc., ensilvania, 1982, Pp.246

reeman, Roger L., Telecomunication System Engineering Analog and Digital Network Desing, John Wiley & Soncs inc., N.J., 1980, p.480

arland, Harry, Introduction To Microprocesor System Desing, RR onnelley Sons Company, 1979, Pp.192

nuth, Donald E., FUNDAMENTAL ALGORITHMS/ volume 1 THE ART OF OMPUTER PROGRAMING

cNamara, John E., Technical Aspects of Data Communication, igital Equipment Corporation, 1977, Pp.387

eatman, John B., Microcomputer-Based Desing, McGraw Hill Book ompany, New York, 1977, Pp.590

chmid, Hermann, Electronic Anmalog Digital Conversions, Van ostrand Reinhold ltd, New York, 1970, Pp.525

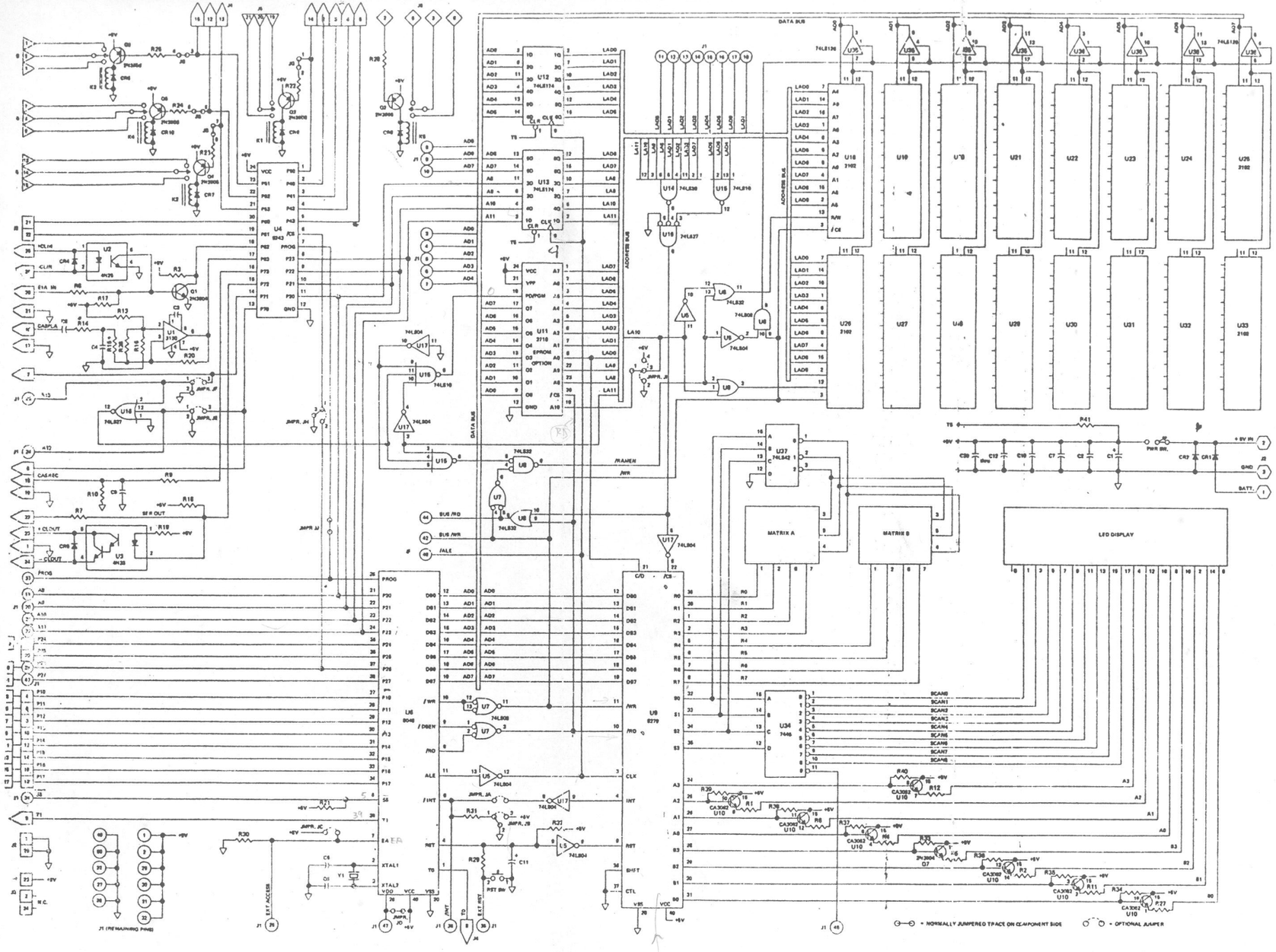
----------,MCS-48 Family of Single Chip Microcomputers, INTEL Corporation, california, April 1979

----------,The 8086 Family User s Manual, INTEL Corporation, California October 1979

----------, SC/MP Microprocessor Applications Handbook, National Semiconductor, california, 1976

----------,MICROCOMPUTADORES Y MICROCOMPUTADORES, serie Mundo Electronico, Marcombo Boixareu Editores, España, 1977,Pp.138

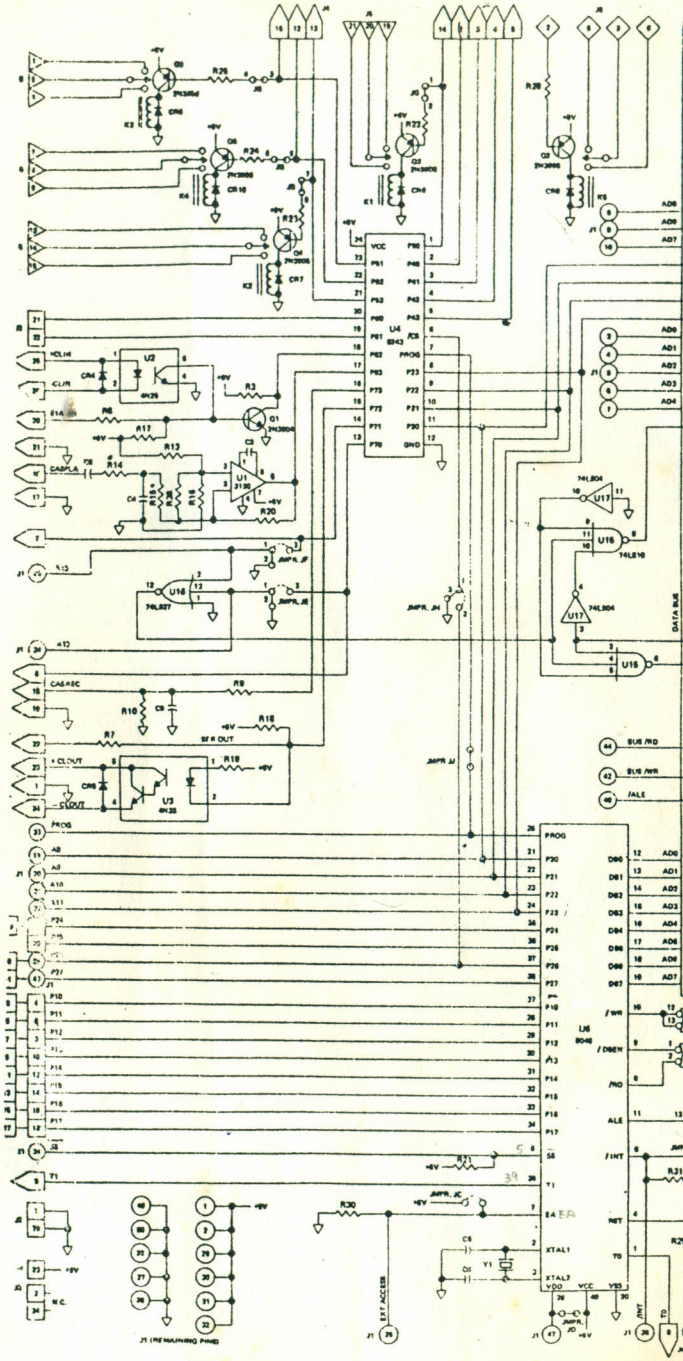
----------,Individual Learning Program Microprocessor, HEATH COMPANY, Michigan,1977

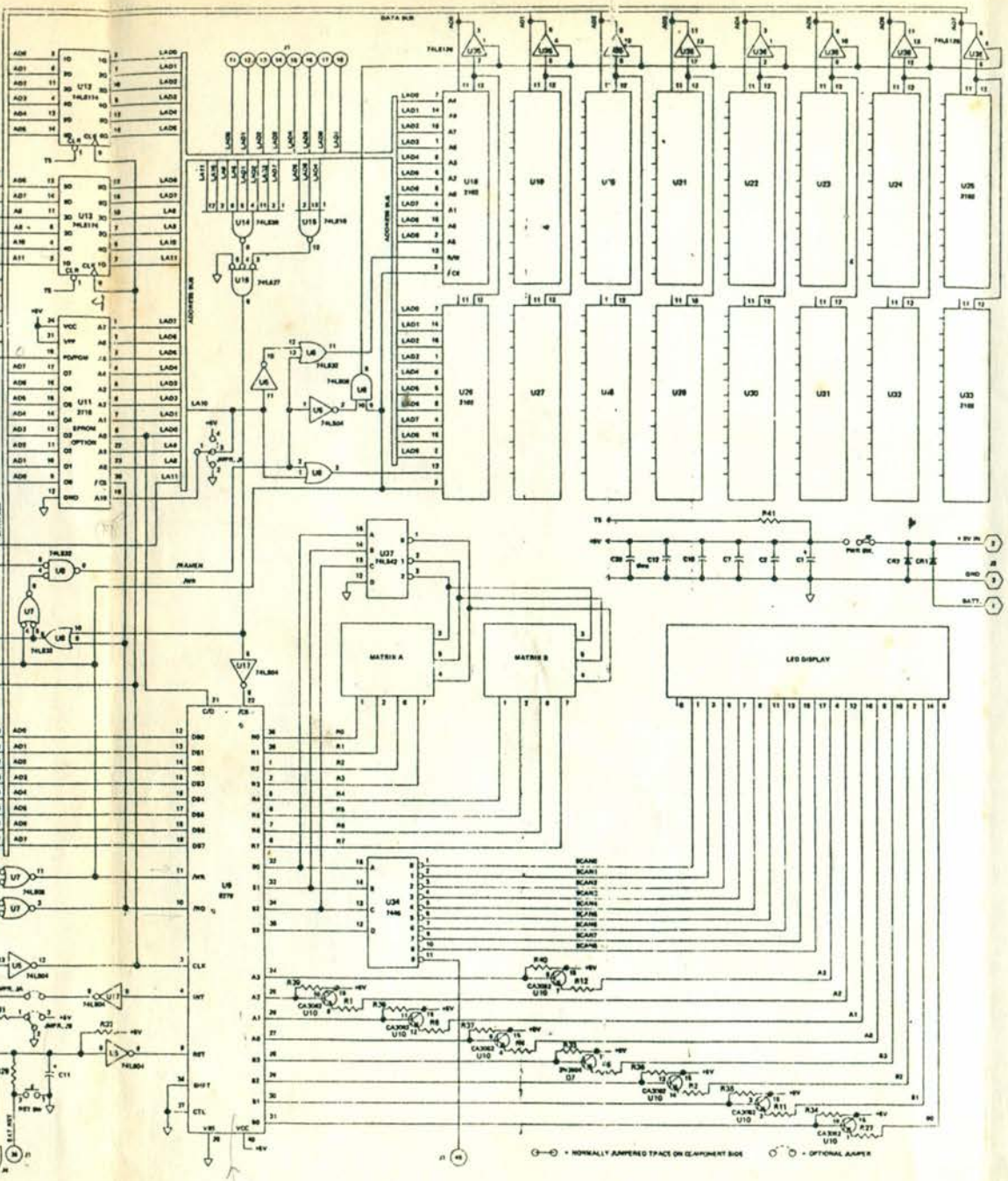


U1	3130
U2	4N25
U3	4N33
U4	8243 w/SKT
U5	74LS04
U6	8048 w/SKT
U7	74LS08
U8	74LS32
U9	8279 w/SKT
U10	CA3082
U11	2716 w/SKT
U12, U13	74LS174
U14	74LS30
U15	74LS10
U16	74LS27
U17	74LS04
U18 THRU *U33	2102
U34	7445
U35, U36	74LS126
U37	74LS42
C1	33uF
C3	39pF
C4, C8	.01uF
C5, C6	20pF
C9	.47uF
C11	2.2uF
C2, C7, C10, C12 THRU *C30	.1uF
CR1, CR2	IN4002
CR3 THRU CR10	IN4148

J1	50 PIN CONN
J2	3 PIN CONN
J3 THRU J5	26 PIN CONN
J6	6 PIN CONN
K1 THRU K5	SPDT RELAY
Q1, Q7	2N3904
Q2 THRU Q6	2N3906
R1, R2, R4 THRU R6, R11, R12, R27	47 ¼w
R3	2.2K ¼w
R7, R19, R21, R29, R30, R33 THRU R41	1K ¼w
R8	3.3K ¼w
R9, R13, R16, R10, R22 THRU R25, R28	4.7K ¼w
R14, R15, R26	22 K ¼w

TOLERANCES UNLESS OTHERWISE SPECIFIED FRACTIONS DEC. ANGLES ± ± ±	IMSAI MFG. CORP. SAN LEANDRO, CA.
APPROVALS DATE DRAWN	IMSAI 8048 CONTROL COMPUT REV. 1 4/77 SCHEMATIC DIAGRAM





U1	3130	J1	50 PIN CONN	R17
U2	4N25	J2	3 PIN CONN	R18
U3	4N33	J3 THRU J5	26 PIN CONN	R20
U4	8243 w/SKT			R31
U5	74LS04			R32
U6	8048 w/SKT	J6	6 PIN CONN	
U7	74LS08			Y1
U8	74LS32	K1 THRU K5	SPDT RELAY	
U9	8279 w/SKT			
U10	CA3082	Q1 Q7	2N3904	
U11	2716 w/SKT	Q2 THRU Q6	2N3906	
U12 U13	74LS174			
U14	74LS30			
U15	74LS10			
U16	74LS27			
U17	74LS04			
U18 THRU *U33	2102	R1 R2 R4 THRU R6 R11 R12 R27	47 ¼w	
U34	7445			
U35 U36	74LS126			
U37	74LS42			
C1	33uF	R3	2.2K ¼w	
C3	39pF	R7 R19 R21 R29 R30 R33 THRU R41	1K ¼w	
C4 C8	.01uF			
C5 C6	20pF			
C9	.47uF			
C11	2.2uF			
C2 C7 C10 C12 THRU *C30	.1uF	R8	3.3K ¼w	
		R9 R13 R16 R10 R22 THRU R25 R28	4.7K ¼w	
			470 ¼w	
CR1 CR2	IN4002			
CR3 THRU CR10	IN4148	R14 R15 R26	22 K ¼w	

TOLERANCES UNLESS OTHERWISE SPECIFIED FRACTIONS DEC. ANGLES ± ± ±	IMSAI MFG. CORP. SAN LEANDRO, CA.
APPROVALS DATE DRAWN	IMSAI 8048 CONTROL COMPUT... REV. 1 4/77