54

UNIVERSIDAD NACIONAL AUTONOMA DE MEXICO

FACULTAD DE INGENIERIA

SISTEMA DE DESARROLLO PARA EL MICROCONTROLADOR MC68HC705J2

T E S I S

QUE PARA OBTENER EL TITULO DE:

INGENIERO EN COMPUTACION

P R E S E N T A N:

ISABEL FABIOLA GUZMAN PEÑA

SERGIO ERNESTO MARTINEZ ARBEZU

DIRECTOR: ING. ANTONIO SALVA CALLEJA

FALLA DE ORIGEN

MEXICO, D. F. 1995

Agradezco a Dios

infinitamente por haberme dado la oportunidad de estar entre personas que me brindaron su apoyo, comprensión y cariño, durante todo este tiempo para cristalizar esta meta.

ÍNDICE

CAPÍTULO I

INTRODUCCIÓN

El avance que se ha obtenido desde la creación de los circuitos integrados es muy significativo, debido a que en cualquier diseño de electrónica se puede observar la gran variedad de estos dispositivos.

Los circuitos integrados se encuentran en forma rectángular o cuadrada, formados en su interior por cristales muy pequeños semiconductores de silicio, en donde cada uno de éstos circuitos contienen desde algunas decenas hasta varios millones de componentes electrónicos, como son: transistores, resistencias, diodos y capacitores, los cuales se encuentran conectados entre sí para formar un circuito electrónico. La parte externa que cubre a cada uno de estos componentes es de plástico o cerámica.

En todos los circuitos integrados existe una denominación impresa, dada por el fabricante, donde se especifica el nombre de este integrado, la fecha en que fue creado, nombre del fabricante y código que lo identifica.

Dentro de la gran variedad de estos dispositivos se encuentran los microcontroladores, los cuales ofrecen varias ventajas, como son:

- Reducción del costo.
- Disminución del consumo de potencia, dando como resultado una operación más económica.
- Confiabilidad contra fallas.
- Velocidad de operación rápida
- Reduce el número de conexiones de alambrado extemo, debido a que la mayoría de las conexiones se encuentran en el interior del microcontrolador.

En el mercado de los microcontroladores existe una gran variedad de éstos, debido a que cada uno presenta características diferentes, por tal motivo el usuario puede elegir el que mejor se adecué a sus necesidades. Entre los proveedores de estos dispositivos se encuentra la empresa Motorola.

Los microcontroladores de esta empresa se encuentran clasificados por familias, en donde cada una de éstas maneja un mismo código de instrucciones para programar a cada uno de los elementos que la componen, pero cada uno de los microcontroladores que pertenecen a la familia presenta características diferentes. Dentro de la familias que se producen se encuentra el microcontrolador MC68HC05 y dentro de ésta existe la versión MC68HC705J2, que fue la empleada para crear al sistema de desarrollo.

Las razones para elegir a este microcontrolador se expresan a continuación:

1. Presentación del Circuito. El tamaño de este elemento es muy pequeño. por consecuencia. el implementarlo en una aplicación nos reduce considerablemente el espacio.

   Este dispositivo se puede encontrar en dos presentaciones:

   a) Programable solo una vez. Como su nombre lo indica, éstos solo pueden grabarse una sola vez, por tal motivo, si el programa que se almacenó en la memoria EPROM no funciona bien o se quieren agregar algunas instrucciones, se debe de ocupar otro de estos dispositivos hasta que funcione de acuerdo a las necesidades requeridas.

      Dentro de esta presentación se encuentran los empaquetados DIP y SOIC.

   b) Programable varias veces. A este tipo de dispositivo se le conoce como microcontrolador con ventana (windowed, en inglés). A este circuito se le puede almacenar información en la memoria EPROM y actualizarla tan constante como sea necesario, con solo exponer al dispositivo a la luz ultravioleta. Es importante recordar que al exhibir al microcontrolador a este tipo de luz se borrará toda la información de esta área, por lo que es recomendable tener un archivo de respaldo, al cual se le realizarán las modificaciones pertinentes, para que posteriormente se graben nuevamente los datos.

2. Capacidad de Almacenamiento. Por el reducido tamaño que presenta este circuito, posee una gran cantidad de memoria EPROM, en la que se puede almacenar un programa de aproximadamente 2 KB de extensión.

3. Costo. El precio del microcontrolador es bajo, si se compara con todos los circuitos integrados que se tendrían que comprar para realizar las funciones que éste nos ofrece; además el costo se reduce si se adquiere un dispositivo que pueda ser grabado una sola vez, siempre que el sistema a almacenarse este probado y no se le vaya a realizar algún cambio.

El principal motivo para usar microcontroladores es la comodidad de usar sus dispositivos internos, para realizar una función o procedimiento específico, por ejemplo: para activar el sistema de suministro de agua en una población. El uso de estos circuitos es muy común, pero las herramientas para programarlos son muy complicadas, por tal motivo se creó un sistema que fuera lo más sencillo posible para trabajar con el dispositivo.

En el trabajo realizado se elaboró un sistema que facilita el uso del microcontrolador, para efectuar esta función nos apoyamos en una computadora, la cual envía y recibe información del microcontrolador, por medio del puerto serial. Debido a que el MC68HC705J2 carece de transmisión serial, se tuvo que crear un sistema que pueda recibir o enviar información por uno de sus puertos. Cuando el sistema de transmisión quedó completamente probado se procedió a generar un sistema que cumpliera con los objetivos.

Entre las principales características que presenta el sistema desarrollado se encuentran las siguientes:

1. Sistema completamente en español.

2. Ensamblador integrado.

3. Graba a otro microcontrolador.

4. Ejecuta programas en una memoria RAM, sin importar el tamaño de éstos.

La función de cada una de estas características están definidas en cada uno de los capítulos de este trabajo, así como algunas utilerías extras que se agregaron.

En el Capítulo II inicialmente se describe que es un microcontrolador, posteriormente se explican las características generales, que es una unidad central de procesos y que dispositivos la conforman, los modos de direccionamiento que maneja, que es una interrupción y cuantas formas existen para generarla, la inicialización del sistema, el mapa de memoria, los registros que se ocupan para programar la memoria EPROM interna, como funciona el temporizador y por último se da una breve explicación de que es el modelo de retención de datos.

En el Capítulo III se describen algunos conceptos básicos, como son: la transmisión serial, asíncrona y el código empleado para generar el programa que se encontrará residente en la memoria EPROM del microcontrolador y que va a ser utilizado para la comunicación entre la computadora y el MC68HC705J2.

Dentro del Capítulo IV se especifican las características de un lenguaje ensamblador, que son las instrucciones y pseudoinstrucciones, el programa analizador así como los diferentes tipos de ensambladores que se pueden encontrar, finalmente se explica que el ensamblador generado es sencillo y como se desarrrollo éste.

El Capítulo V nos describe en forma detallada como se generaron todas las opciones que conforman al sistema, además se muestran los algoritmos que nos ayudaron a realizar la función de cada una de éstas. Las opciones que tiene el sistema es el manejo del ensamblador, de disco, de memoria y utilerías, en donde cada una de estas nos presenta varias alternativas.

Para demostrar el buen funcionamiento del sistema en el Capítulo VI se describe una aplicación, la cual tiene como función suministrar agua o cualquier líquido a diferentes altitudes, controlando varias bombas, las cuales van a subir el líquido a diferentes tanques.

Finalmente se creó un Apéndice, que es un manual para el usuario, en donde se le explica cuales son las características de la computadora, así como los componentes que debe de tener para manejar al sistema. Además se describe en forma detallada como manejar el programa y que función realiza cada una de las opciones que se despliegan en la pantalla de la computadora.

CAPÍTULO II

GENERALIDADES DEL MICROCONTROLADOR MC68HC705J2

INTRODUCCIÓN.

Un microcontrolador es un circuito integrado que contiene varias funciones de un sistema de computación, formado por una Unidad Central de Procesos (CPU, por sus siglas en inglés. Central Processing Unit), memoria, reloj interno y puertos de entrada-salida (I/O, Input/Output). Cuando alguna de estas partes no forma parte del circuito integrado, por ejemplo la memoria o algún puerto de entrada o salida, se obtiene la configuración de un microprocesador. En la figura 1.1 se muestra en forma general los componentes de un microcontrolador.

FIGURA 1.1 SISTEMA DE COMPUTACIÓN PARA UN MICROCONTROLADOR

En la figura 1.2 se muestra un diagrama de bloques, detallado de la estructura interna de un microcontrolador. El cristal que se observa no es parte interna de éste, pero es un componente necesario para el funcionamiento del reloj interno. Este cristal puede sustituirse por otro componente como un resonador cerámico o un circuito con capacitores y resistencias.

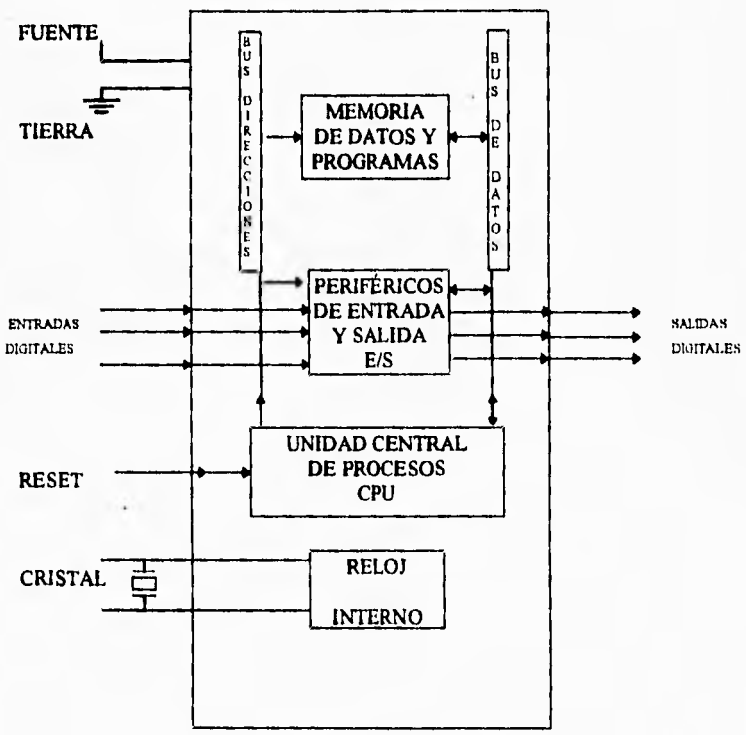


FIGURA 1.2 DIAGRAMA GENERAL DE UN MICROCONTROLADOR

Los microcontroladores son circuitos integrados muy pequeños, los cuales son usados en diferentes aplicaciones. Una ventaja de un sistema que emplee estos dispositivos es evitar el uso excesivo de componentes electrónicos, reduciendo asi el costo y el espacio para elaborar un circuito impreso.

Para el desarrollo de una aplicación, el usuario debe realizar un programa empleando el conjunto de instrucciones propias de la familia del microcontrolador. Dentro de la familia de microcontroladores de Motorola se encuentra la versión MC68HC05 basada en la Arquitectura de Von Neumann. lo cual permite emplear las localidades de memoria para almacenar tanto instrucciones de un programa como datos. Las funciones que realiza esta arquitectura son:

1. Analizar la instrucción.

2. Procesar los datos.

3. Analizar la siguiente instrucción.

4. Realizar los pasos anteriores secuencialmente.

En esta arquitectura, la unidad central de procesos y un arreglo de memoria se encuentran interconectados por medio de un Bus de Direcciones y un Bus de Datos. El bus de direcciones se utiliza para identificar las localidades de memoria que se van a manejar y la función del bus de datos es transportar la información desde la unidad central de procesos a las localidades de memoria o viceversa.

La familia de los microcontroladores MC68HC05 se encuentran aproximadamente formada por más de 100 miembros, dentro de ésta, se encuentra el MC68HC705J2 que es de bajo costo y de alto funcionamiento. Este microcontrolador usa una unidad central de procesos de ocho bits, que le permite ser compatible con una gran variedad de subsistemas, con diferentes tipos y tamaños de memorias.

CARACTERÍSTICAS DEL MICROCONTROLADOR MC68HC705J2

Las características principales del microcontrolador se describen en el párrafo siguiente:

- Unidad Central de Procesos (CPU)

- Mapa de Memoria de Registros de Entrada/Salida (I/O).

- 2064 bytes de memoria EPROM incluyendo 16 localidades de vectores.

- 112 bytes de memoria RAM Estática (SRAM).

- 14 pines bidireccionales de Entrada-Salida (I/O).

- Operación completamente estática, sin tope inferior para la frecuencia de reloj.

- Oscilador al que se conecta un resonador de cristal.

- Temporizador multifuncional de 15 bits.

- Circuito de interrupción en tiempo real.

- Cargador de Inicio en Memoria de solo Lectura (Bootloader ROM).

- Capacidad para bajar consumo de energía, al ejecutar las instrucciones especiales Alto (Stop) y Espera (Wait), y modos de funcionamiento que permiten retener datos.

- Capacidad de emular al MC68HC05J1.

- Capacidad para habilitar interrupciones externas mediante flanco o nivel.

- Capacidad para modificar el tiempo de reseteo del watch dog.

- Instrucciones que permiten efectuar la multiplicación de dos operandos validados en 2 bytes cada uno.

- Se puede encontrar en los siguientes tipos de encapsulados:

  * Empacado encapsulado de 20 pines sin ventana para grabarse una sola vez.
  * Encapsulado SOIC (por sus siglas en inglés Small Outline Integrated Circuit).
  * Encapsulado de 20 pines con ventana para programar y borrar.

En la figura 1.3 aparece el **Diagrama de Bloques del MC68HC705J2.** Este diagrama muestra los subsistemas principales y como se relacionan estos con cada uno de los pines del microcontrolador. Posteriormente se describen las partes principales que lo conforman.

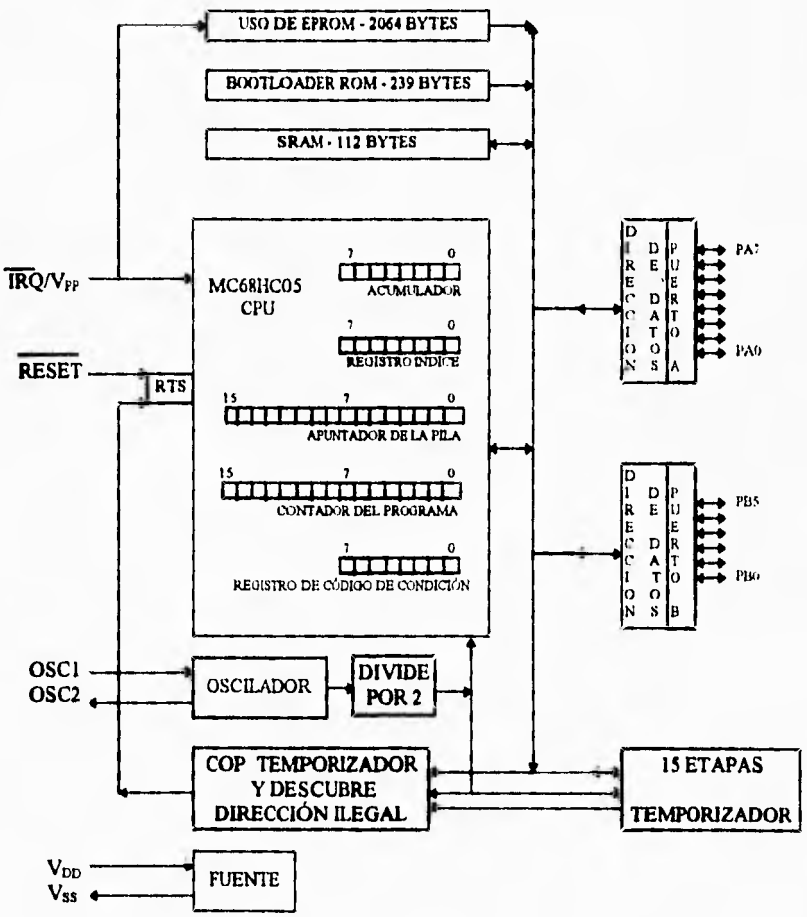


FIGURA 1.3 DIAGRAMA DE BLOQUES PARA EL MICROCONTROLADOR

UNIDAD CENTRAL DE PROCESOS.

El análisis de la Unidad Central de Procesos (CPU) interno del MC68HC705J2 se basa en los siguientes elementos:

- Capacidad de 8 bits de datos
- Modelo de Programación
- Conjunto de Instrucciones

La arquitectura interna de la Unidad Central de Procesos esta constituida por los elementos siguientes:

- Unidad de Control.
- Unidad Aritmética y Lógica (ALU, por sus siglas en inglés Aritmetic Logic Unit).
- Registros Internos: Acumulador, Apuntadores, Banderas de Estado.

MODELO DE PROGRAMACIÓN.

El modelo de programación del MC68HC705J2 se muestra en la figura 1.4.

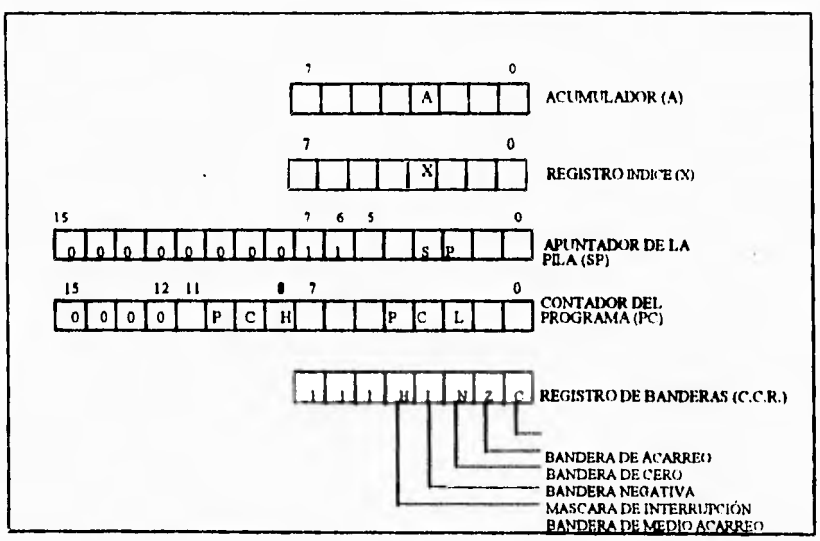


FIGURA 1.4 MODELO DE PROGRAMACIÓN

En las siguientes líneas se describe cada uno de los registros que conforman al modelo de programación.

REGISTRO ACUMULADOR.

El acumulador es un registro de 8 bits, el cual es usado por la unidad central de procesos para guardar operandos, resultados de cálculos aritméticos y operaciones no aritméticas, así como la manipulación de datos.

REGISTRO ÍNDICE X.

El registro índice X es de 8 bits, este puede ser empleado en cualquiera de sus dos funciones:

a) Direccionamiento Indexado. El CPU usa el byte del registro índice para determinar la dirección condicional del operando.
b) Almacenamiento Temporal. Es utilizado como acumulador auxiliar para almacenar información temporalmente.

REGISTRO APUNTADOR DE LA PILA.

El apuntador de la pila (SP, Stack Pointer) es un registro de 16 bits que contiene la dirección de la siguiente localidad libre en la pila. Durante un Reset o después de un Reset el contenido del apuntador de la pila esta direccionando a la localidad de memoria $00FF. La dirección en el apuntador de la pila efectúa un decremento cuando se coloca un dato dentro de ésta y realiza un incremento cuando se extrae información.

Los 10 bits más significativos del apuntador de la pila son colocados permanentemente en 0000000011, por lo que el apuntador de la pila se direcciona a partir de la localidad de memoria $00C0 hasta $00FF. Si las subrutinas e interrupciones usan más de 64 localidades de la pila, el apuntador de la pila retorna a la dirección de memoria $00C0 y sobreescribe en los datos previamente almacenados. Una subrutina usa dos localidades dentro de la pila; una interrupción usa cinco localidades de la pila.

REGISTRO CONTADOR DEL PROGRAMA.

El contador del programa (PC, por sus siglas en inglés Program Counter) es un registro de 16 bits que contiene la dirección de la siguiente instrucción u operando para ser ejecutado. Los 4 bits más significativos del contador del programa se colocan en 0000. Normalmente la dirección en el contador del programa incrementa automáticamente y secuencialmente a la siguiente localidad de memoria que contenga el código de operación de la siguiente instrucción, cada vez que una instrucción ha sido ejecutada. En una operación de salto a una subrutina o una interrupción se carga al contador del programa con la dirección de memoria de inicio de dicha subrutina o interrupción.

REGISTRO DE BANDERAS.

El registro de código de condición (CCR, por sus siglas en inglés Condition Code Register) es de 8 bits en donde los tres bits más significativos siempre permanecen en 111. El CCR contiene la mascara de interrupción y cuatro banderas que muestran el resultado de la instrucción ejecutada. En el párrafo siguiente se describe la función del registro CCR.

a) Bandera de Medio Acarreo (H). La CPU enciende esta bandera al ocurrir un acarreo entre los bits tres y cuatro del acumulador.

b) Mascara de Interrupción (I). Al seleccionar la mascara de interrupción se deshabilita el llamado a las interrupciones. Si se genera una interrupción cuando la mascara es cero, el CPU almacena los registros en la pila, selecciona la mascara, deshabilita las interrupciones y carga el vector de interrupciones.

c) Bandera Negativa (N). La CPU enciende esta bandera cuando el resultado de una operación aritmética, operación lógica o una manipulación de datos es negativo.

d) Bandera de Cero (Z). La bandera de cero enciende cuando una operación aritmética, operación lógica, o una manipulación de datos produce un $00.

e) Bandera de Acarreo (C). La CPU usa esta bandera para indicar que el resultado de la operación suma produce un acarreo fuera de los ocho bits del acumulador. Algunas manipulaciones de instrucciones y operaciones lógicas pueden apagar o encender esta bandera.

UNIDAD DE CONTROL.

Es la unidad encargada de decodificar y ejecutar las instrucciones que le son enviadas al microcontrolador por la unidad de memoria, de tal forma que esta unidad es la que interpreta el programa que deberá ejecutar el procesamiento central.

UNIDAD ARITMÉTICA Y LÓGICA .

La Unidad Aritmética y Lógica (ALU, por sus siglas en inglés Arithmetic Logic Unit) ejecuta las operaciones aritméticas y lógicas definidas por el conjunto de instrucciones. Los circuitos aritméticos decodifican las instrucciones binarias y lo sitúan arriba de la ALU para seleccionar la operación.

MODOS DE DIRECCIONAMIENTO

La CPU usa ocho modos de direccionamiento. Estos modos definen la manera en la cual la unidad central de procesos encuentra los datos que se requieren para ejecutar una instrucción. En los párrafos siguientes se describen los diferentes modos de direccionamiento, así como los símbolos especiales que son usados en estos modos.

- El signo $, indica que el número con el que se está trabajando se encuentra en código hexadecimal.

- El signo #, indica una operación inmediata y el número se encuentra en la localidad siguiente al código de la instrucción.

Direccionamiento Inherente. Esta instrucción no tiene operando y actúan sobre un dato en los registros de la CPU. Las instrucciones inherentes no requieren de direcciones de memoria y son de un byte de longitud. Ejemplo:

```
0200    4C    INCA
```

Direccionamiento Inmediato. Es la que contiene los valores para ser usados en una operación con el valor en el acumulador o en el registro índice. Las instrucciones inmediatas no requieren de direcciones de memoria y son de dos bytes de longitud. El primer byte contiene el código de la instrucción y el segundo byte contiene el valor del dato inmediato. Ejemplo:

```
0200    A6 02    LDA #$02
```

Direccionamiento Directo. Las instrucciones directas accesan las primeras 256 localidades de la memoria, por tal motivo estas instrucciones son de dos bytes. El primer byte es el código de la instrucción y el segundo byte es la parte baja de la dirección del operando. Ejemplo:

```
0200    B6 E0    LDA $E0
```

Direccionamiento Extendido. Las instrucciones que operan en modo extendido utilizan tres bytes. El primer byte es el código de la instrucción, el segundo y tercer byte representan la parte alta y baja respectivamente de la localidad de memoria a direccionar. Ejemplo:

```
0200    C6 03 65    LDA #$0365
```

Direccionamiento Indexado sin Desplazamiento. Estas instrucciones son de un byte y pueden accesar datos con direcciones variables dentro de las primeras 256 localidades de memoria. El registro índice contiene el byte menos significativo de la dirección del operando condicional. La CPU asigna automáticamente $00 al byte alto de la dirección del operando condicional. Estas instrucciones pueden ser localidades de memoria que se encuentren a partir de la dirección $0000 hasta $00FF.

```
0200    F6    LDA $00,X
```

Direccionamiento Indexado con 8 Bits sin Desplazamiento. Las instrucciones son de 2 bytes y pueden accesar datos con direcciones variables dentro de las primeras 511 localidades de memoria. La CPU suma el byte sin signo al registro índice para obtener el siguiente byte del registro índice del operando. La suma es la dirección condicional, pueden ser localidades de memoria que se encuentren dentro del rango de direcciones $0000 a $01FE.

```
0200    E6 05    LDA $05,X
```

Direccionamiento Indexado con 16 Bits de Desplazamiento. Estas instrucciones son de tres bytes y pueden accesar datos con direcciones variables en una localidad de memoria. La CPU suma el byte sin signo al registro índice con los dos bytes sin signo siguientes del código de instrucción. El primer byte después del código de instrucción indica la parte alta de los 16 bits; el segundo especifica la parte baja del byte de desplazamiento. Estas instrucciones pueden hacer referencia a direcciones de memoria.

```
0200    D6 0377    LDA $0377,X
```

Direccionamiento Relativo. Este modo se utiliza en las instrucciones de salto. Para permitir brincar hacia delante o hacia atrás, el desplazamiento a efecuarse debe encontrarse dentro del rango de -127 a + 128 bytes, para verificar que el salto se encuentra dentro de este rango, se cuentan las localidades de memoria que se van a saltar a partir de la siguiente localidad después de la instrucción de brinco. Ejemplo:

```
0200    27 02        BEQ ETI
0202    A6 05        LDA #$05
0204    B7 00  E1:   STA $00
```

RESET E INTERRUPCIONES

RESET.

El reset es una señal de control bidireccional que se activa en bajo. Esta señal es utilizada para inicializar al sistema, además se emplea como salida para indicar que se detectó una falla en el microcontrolador. Al efectuarse un reset, inmediatamente se detiene la operación que se efectúe en ese momento en la unidad central de procesos y le asigna al contador del programa la dirección del vector de reset. Un reset se produce de diferentes formas:

- Al encender la fuente.
- Aplicando un cero lógico al pin de RESET.
- Si existe un tiempo fuera dentro del temporizador COP.

En cualquier forma que se aplique un reset el microcontrolador se reinicializa, es decir, se limpian todos los registros de dirección de datos, los bits de control del timer y los registros de estado, la bandera de alto, habilita el reloj de la unidad central de procesos y los puertos son configurados como entradas, se carga al apuntador de la pila con $FF y se asigna la mascara de interrupciones.

INTERRUPCIONES.

Al generarse una interrupción se detiene el proceso normal para realizar una función particular. La diferencia que existe entre un llamado de una interrupción y un reset, consiste en que la interrupción no detiene la operación que empieza a ejecutarse. Una interrupción principia cuando se ha terminado de ejecutar la instrucción, posteriormente se almacenan los registros de la unidad central de procesos en la pila y se le asigna al contador del programa la dirección del vector de interrupción. La llamada a una interrupción se ocasiona por:

- Un sobreflujo en el timer.
- Aplicar un cero lógico al pin de IRQ, el cual produce una interrupción externa.
- Activar las interrupciones de software.

Cuando se atiende el llamado de una interrupción la unidad central de procesos almacena todos los registros en la pila, coloca la mascara de interrupciones para prevenir una nueva invocación a una interrupción y le asigna al contador del programa el valor que se encuentra almacenado en la localidad de memoria del vector de interrupción que corresponde al tipo de llamado. Los vectores de interrupciones se encuentran almacenados en las siguientes localidades de memoria:

a) $0FF8 y $0FF9 vector de interrupción para el temporizador.
b) $0FFA y $0FFB vector para interrupciones externas.
c) $0FFC y $0FFD vector para interrupciones de software.

Al invocar la instrucción de retorno de interrupción RTI se regresan los valores a los registros de la CPU que se encontraban almacenados en la pila, retira la mascara de interrupciones. En la figura 1.5 se muestran los registros que se utilizan cuando se invoca una interrupción, así como la forma en que son almacenados y extraídos de la pila.

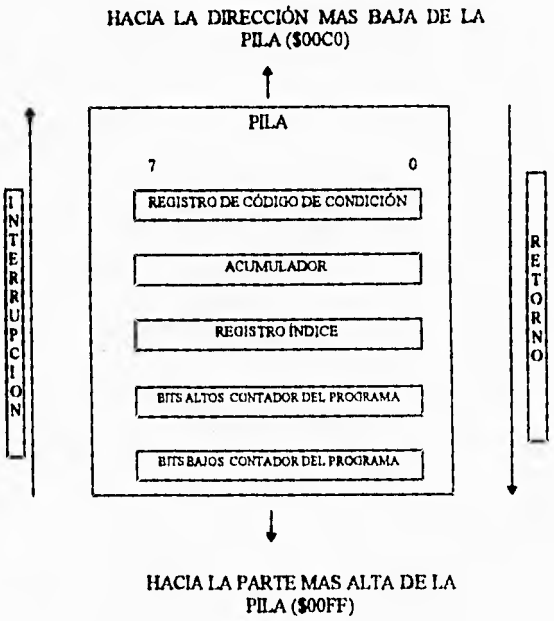


FIGURA 1.5 ALMACENAMIENTO DE LOS REGISTROS EN LA PILA DURANTE UNA INTERRUPCIÓN

El temporizador genera dos tipos de interrupciones, en ambos casos se coloca la mascara de interrupciones para deshabilitar las interrupciones del timer.

a) Interrupción de sobreflujo en el temporizador. Esta interrupción se activa si la bandera sobreflujo (TOF) enciende mientras que la interrupción habilita al bit TOIE. Los bits TOF y TOIE se encuentran en el registro de estado y control (TCSR), que se encuentra en la dirección $0008. El bit TOF es la bandera de sobreflujo del temporizador. Esta bandera configura a los bits dejándolos de lectura, los coloca en las primeras ocho etapas de la lista del contador que están sobre de $FF a $00. TOF genera una interrupción de sobreflujo del temporizador solo si TOIE se ha seleccionado. Cuando escribimos un cero en el bit TOF, éste se limpia, pero no se produce ningún efecto al escribir un uno.

El bit TOIE es de lectura-escritura y al seleccionarlo se habilita la Bandera de Interrupción de Sobreflujo del Temporizador.

b) Interrupción en Tiempo Real.Esta interrupción ocurre si se enciende la bandera RTIF, mientras se habilita el bit para la interrupción en tiempo real RTIE. El bit RTIE habilita la Interrupción en Tiempo Real. El bit RTIF es la bandera que se ocupa para activar la interrupción en tiempo real.

Las interrupciones externas ocurren en el pin IRQ, al terminar el llamado a una interrupción. el pin testigo retorna a su estado inicial. Cuando existe una llamada a una interrupción y se da un reset a la mascara de interrupción, la unidad central de procesos inicia la secuencia de interrupción, posteriormente la CPU limpia el llamado a la interrupción mientras que el vector de interrupciones se carga, de tal forma que si se requiere de otra interrupción externa, ésta puede ejecutarse durante la rutina de servicio de interrupciones.

MAPA DE MEMORIA.

La CPU puede direccionar 4 Kbytes de memoria. El contador del programa avanza una dirección en un tiempo por la memoria, lee las instrucciones del programa y los datos. En los primeros 32 bytes de espacios de la memoria, se definen las direcciones de control, los registros de estado y los registros de datos de I/O. En el mapa de memoria se puede observar que el microcontrolador tiene 112 bytes de memoria estática de lectura-escritura que se ocupa para almacenar variables y datos temporales durante la ejecución del programa , 2 Kbytes de memoria EPROM, en esta memoria se almacenan las instrucciones de un programa y se fijan los datos, 239 bytes de bootloader que sirve para copiar y verificar el contenido de una memoria EPROM externa al chip y 16 bytes para el uso de vectores. En la figura 1.6 se observa el mapa de memoria, así como las direcciones para los registro de I/O y el área de trabajo de los vectores .

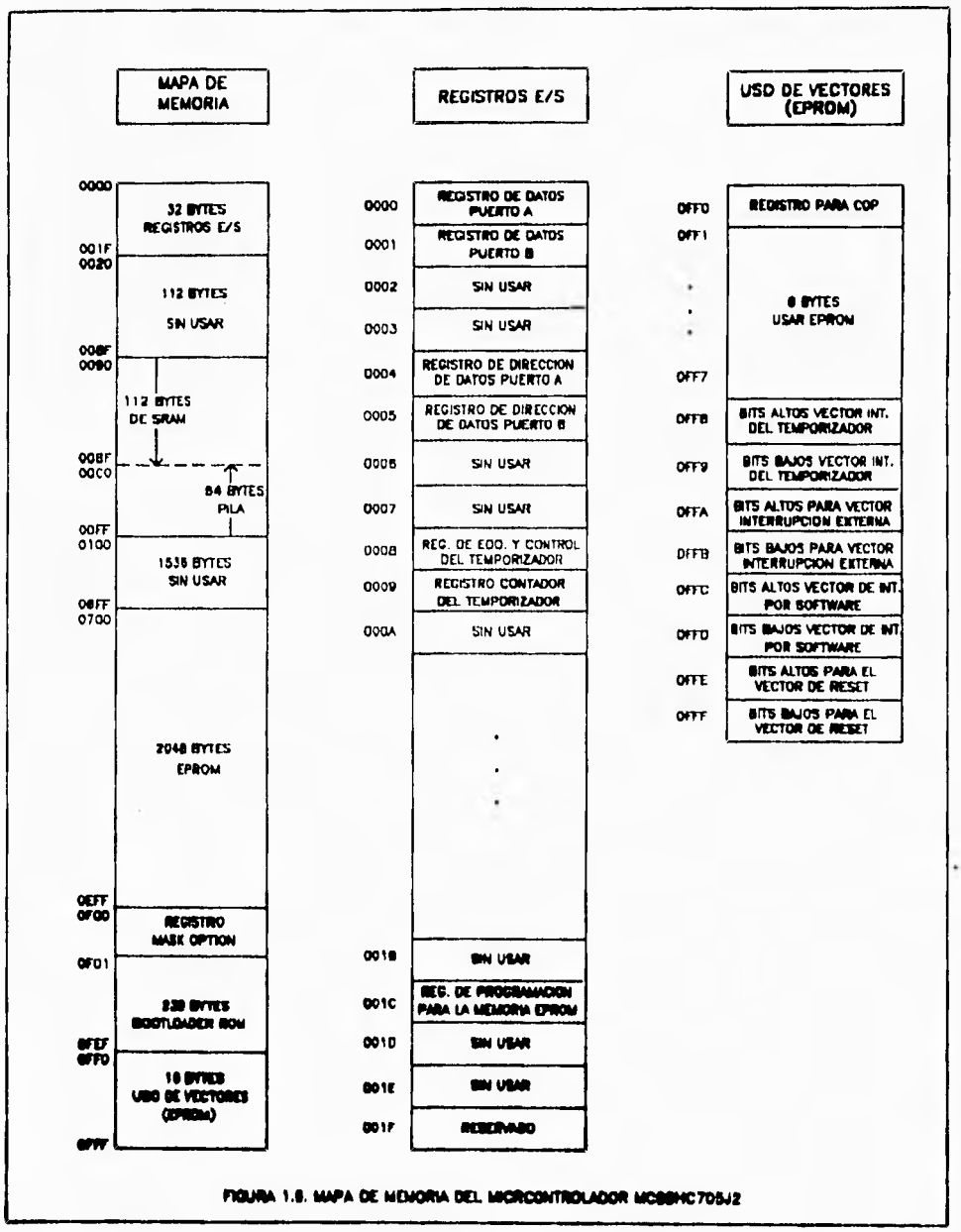


FIGURA 1.8. MAPA DE MEMORIA DEL MICROCONTROLADOR MC68HC705J2

EPROM.

Las direcciones de la memoria EPROM se encuentra a partir de la localidad $0700 hasta $0EFF, es decir, tiene una equivalencia aproximada de 2 kbytes. Existen ocho direcciones de memoria que se encuentran en el rango de $0FF8 a $0FFF, las cuales son localidades de memoria reservadas para los vectores de interrupción y de reset. Existen dos formas para escribir datos en la memoria EPROM:

1. Los registros de control de la memoria EPROM contienen los bits para que se programe esta memoria byte por byte.

2. El bootloader ROM contiene las rutinas para bajar información desde una memoria EPROM externa del chip.

PROGRAMACIÓN DE LA MEMORIA EPROM.

Los registros de programación y los bits de control para programar la memoria EPROM se muestran en la figura 1.7. Este registro se encuentra en la dirección de memoria $001C.

BIT	7	6	5	4	3	2	1	0
	0	0	0	0	0	LATCH	0	EPGM
RESET	0	0	0	0	0	0	0	0

FIGURA 1.7 REGISTRO PROG DE PROGRAMACIÓN PARA LA MEMORIA EPROM.

Si se limpia el bit de Latch automáticamente se limpia el bit de EPGM.

Bit EPGM. Este bit de lectura-escritura aplica el voltaje para programar la memoria EPROM. Para escribir en el bit EPGM, el bit de Latch debe seleccionarse.

Para programar un byte de la memoria EPROM se deben seguir los siguientes pasos:

1. Aplica 16.5 V al pin IRQ-V_{pp}.
2. Seleccionar el bit Latch.
3. Escribir en una dirección de la memoria EPROM.
4. Seleccionar el bit EPGM durante cuatro milisegundos, para aplicar el voltaje de programación.
5. Limpiar el bit Latch.

Latch-EPROM. Este bit de lectura-escritura activa al bus de direcciones y al de datos para ser empleado en la programación de la memoria EPROM.

Para borrar la memoria EPROM interna del microcontrolador se debe exponer en luz ultravioleta de 15 Ws/cm^2 con una longitud de onda de 2537 angstroms a una distancia de 3 cm aproximadamente.

TEMPORIZADOR

El temporizador es un periférico usado para medir o generar tiempo en un evento que se relaciona con el sistema del microcontrolador. Este periférico es capaz de funcionar con medidas de frecuencia o con un tren de pulsos.

El temporizador que se encuentra dentro del MC68HC05 se activa cuando el reloj interno de éste, inicia su ciclo, la señal que se genera es llamada Reloj E (E- clock, en inglés). Esta señal se origina al dividir la frecuencia del cristal entre dos; el Reloj E se usa para manejar a un preescalador, que se encuentra dividido entre cuatro y éste a su vez habilita un contador de 8 bits. El valor del contador se encuentra almacenado en el Registro Contador del Temporizador (TCR, por sus siglas en inglés, Timer Counter Register) que se encuentra en la localidad de memoria $09. El valor que se encuentra dentro de esta dirección no puede ser modificado por el usuario, solo puede leer el dato.

Al encender la fuente de alimentación los bits del TCR se inicializan en cero. Después de 4064 ciclos de reloj interno, se libera el reset, se inicializa nuevamente el contador para permitir que el microcontrolador deje el valor de cero.

Si el contador tiene el valor de $FF y se incrementa éste, el dato siguiente es un $00, se produce un sobreflujo, es decir, se enciende la bandera de sobreflujo (TOF, Timer Overflow Flag), que es un bit que se encuentra localizado en el Registro de Estados y Control del Temporizador (TCSR, Timer Control and Status Register). El estado de esta bandera puede verificarse por diversas instrucciones que realice la CPU.

Opcionalmente, el programador puede seleccionar el bit que habilita la interrupción de sobreflujo en el temporizador (TOIE, Timer Overflow Interrupt Enable), para que cuando el contador encienda la bandera de sobreflujo, la CPU de inicio a una interrupción. Para que se efectúe un sobreflujo deben de haber transcurrido 1024 ciclos del reloj E.

Además de que el contador ocasiona una interrupción periódica, la salida de éste maneja la entrada de un contador de siete bits. La salida de los cuatro bits menos significativos de este contador, puede ser usado para generar una interrupción periódica real adicional. Para generar una interrupción en tiempo real, se puede seleccionar uno de cuatro tipos de periodos de operación, con solo configurar los bits de RT1 y RT0 que se encuentran dentro del Registro de Estado y Control del Timer. En la tabla siguiente se muestran los cuatro tipos de configuraciones que se pueden emplear para generar las interrupciones en tiempo real.

RTI	RT0	RTI	PERIODO DE RESET PARA EL COP MÍNIMO
0	0	8.2 ms	57.3 ms
0	1	16.4 ms	114.7 ms
1	0	32.8 ms	229.4 ms
1	1	65.5 ms	458.8 ms

TABLA 1.2. CONFIGURACIÓN PARA EL TEMPORIZADOR EN TIEMPO REAL

Finalmente el temporizador tiene un contador de tres bits que forman al Sistema Guardián Propio de Operación de Computación. (COP, Computer Operating Properly). La función del Sistema COP es proteger a un sistema para que no se pierda cuando se ejecuta. Cuando habilitamos a este sistema, una secuencia de reset en el COP debe de ser ejecutada antes de que el periodo de tiempo termine, de tal forma que el COP no ha terminado su periodo de tiempo y el microcontrolador ya se ha inicializado. Para prevenir que el sistema COP se active antes de que finalice su periodo de tiempo, se debe escribir un cero en el bit cero de la localidad de memoria $03F0, antes de que el COP vaya a ejecutar el reset, porque la entrada del temporizador guardián es activada por la salida del circuito de interrupción del temporizador en tiempo real.

MODO DE RETENCIÓN DE DATOS.

En el modo de retención de datos el microcontrolador retiene el contenido de la memoria RAM y el contenido de los registros de la CPU con un voltaje V_{DD} tan bajo como 2 V de C.D. La retención de datos permite al microcontrolador permanecer en un estado de consumo de bajo poder, pero la CPU no puede ejecutar instrucciones.

Para colocar al microcontrolador en modo de retención de datos:

1) Conducir al pin de RESET a cero.

2) Bajar el voltaje de V_{DD}. La línea de RESET se mantiene continuamente en bajo durante el modo de retención de datos.

Para extraer al microcontrolador del modo de retención de datos:

1) Retornar el voltaje V_{DD} a su operación normal.

2) Retornar al pin de RESET a uno lógico.

CAPÍTULO III

KERNEL BÁSICO

En este capítulo se describe el sistema realizado para la simulación del puerto serie en el microcontrolador MC68HC705J2 para la comunicación entre este circuito y una computadora personal que trabaja a una velocidad de 1200 bauds, así como algunos conceptos básicos.

El programa fue desarrollado en lenguaje ensamblador propio del 68HC05, posteriormente el software se cargo en la memoria EPROM del microcontrolador a través de una memoria auxiliar.

DESARROLLO DEL NÚCLEO DEL SISTEMA.

Para crear la parte fundamental de un sistema que va a residir en la memoria del microcontrolador es necesario tomar en cuenta varios aspectos para comunicar ambos dispositivos por medio de un puerto, por tal motivo se da una breve explicación de las dispositivos que deben de tomarse en cuenta, así como el funcionamiento básico de éstos mismos.

Para realizar la comunicación entre dos sistemas es necesario que ambos estén comunicados por medio de un canal o una línea, es decir, una vía sobre la cual se transfieran datos entre los dispositivos. Estas líneas de comunicación se encuentran clasificadas en:

1. Circuitos paralelos. Transmiten datos en múltiples líneas, es decir, se pueden enviar uno o más bytes a la vez.

2. Circuitos seriales. Estos realizan la transmisión de datos por una sola línea bit por bit.

Una vez que los dispositivos se encuentran conectados entre sí, por medio de las líneas de comunicación, éstas pueden enviar información de un dispositivo a otro por medio de las técnicas de transmisión síncrona o asíncrona.

En una transmisión síncrona los datos que van a enviarse se almacenan en un bloque, para que posteriormente se emitan individualmente conforme fueron generados. Dentro de este tipo de transmisión se requiere de un carácter especial, llamado carácter sincrónico, el cual es instalado al inicio el bloque de datos. Usualmente se colocan dos o más caracteres sincrónicos para asegurar que la información que se envía no se pierda en la transmisión. Cuando la transmisión se ha iniciado , los datos se envían continuamente hasta que el bloque se ha terminado. El carácter de sincronía es similar a un bit de inicio pero este no necesita de un bit final como en la transmisión asíncrona. En la figura 2.1. se ilustra la transmisión síncrona, en donde los caracteres forman un grupo que requieren ser almacenadas en una memoria intermedia. Para generar este tipo de transmisión se envían uno o más caracteres sincrónicos para establecer una sincronía con el reloj del otro dispositivo. Las letras C1 y C2 representan a los bits de sincronía y las letras D1 hasta Dn son los datos a transmitirse.

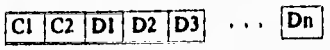


FIGURA 2.1 TRANSMISIÓN SINCRÓNICA

En la transmisión asíncrona, el carácter que va a enviarse esta codificado dentro de una serie de pulsos. La transmisión de cada carácter esta inicializada por un pulso de inicio de igual longitud que los pulsos que representan a los datos. Al terminar la serie de estos pulsos o caracteres codificados, estos se encuentran seguidos de un pulso final. El bit de inicio representa el origen de una cadena de caracteres para transmitirse y sirve para indicar que después de éste siguen los bits con la información, posteriormente se encuentra el bit final y este es utilizado para señalar que la información se ha enviado completamente.

En la figura 2.2 se muestra una transmisión de un carácter de ocho bits que puede expandirse a diez u once, dependiendo de la longitud del bit final. Los ocho bits de datos pueden usar un bit de paridad con el cual se detectan y corrigen errores.

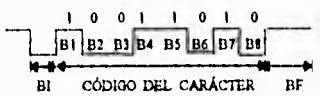
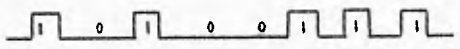


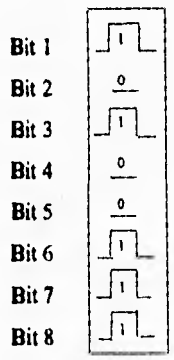
FIGURA 2.2 TRANSMISIÓN ASINCRÓNICA DE UN CARÁCTER DE OCHO BITS

Ahora bien, una vez que se conocen las dos técnicas de transmisión, se elige el tipo de transmisión que se va a emplear para enviar los datos. Dentro de los tipos de transmisión se encuentran las seriales y paralelas, en la transferencia serial, los bits que componen a un carácter son enviados en secuencia por una línea, mientras que en la paralela, los caracteres se transmiten serialmente, pero con la diferencia que los bits que representan a la información, se envían en forma simultánea, por ejemplo, si un carácter esta formado por ocho bits, en una transmisión paralela se requiere mínimo de ocho líneas para enviarlos, mientras que en la transmisión serial se requiere solo de una línea.

En la figura 2.3 se observan los dos tipos de transmisiones empleadas. En la serial los bits que conforman a un carácter son enviados secuencialmente por una sola línea, mientras que en la paralela son emitidos serialmente, pero los bits que representan los caracteres se envían por varias líneas.



a) Transmisión Serial



b) Transmisión Paralela

FIGURA 2.3 TIPOS DE TRANSMISIÓN

Para realizar la transmisión de información de un dispositivo a otro, además de contar con las líneas de comunicación y las técnicas de transmisión, se debe contar con una interface, por tal motivo se empleó la interface RS-232-C creada por la Asociación de Industrias Electrónicas (EIA) en Washington, D.C. USA.

Dentro de esta interface se encuentra el estándar RS-232-C la cual transmite información entre el equipo terminal de datos y el equipo de comunicación, en el rango de 0 a 19,200 bits por segundo. Esta interface opera con datos sincrónicos o asincrónicos, pero se encuentra limitada por una longitud de cable, es decir, para realizar una interface entre dispositivos es necesario no exceder de los 15 metros de longitud entre ambas terminales.

Existen dos tipos de conectores para la interface RS-232-C, el DB25 y el DB9, en donde el DB25 se fabricó para adaptar a la tarjeta serial-paralela con IBM PC AT y una computadora personal compatible. Posteriormente se creó un conector de nueve líneas para el puerto serial RS-232-C. Las características principales de esta interface se describen en el párrafo siguiente:

- Define las señales de control a través de la interface.

- Determina el movimiento de los datos en la transmisión.

- Transmite las señales para sincronizar los dispositivos conectados.

- Por medio de una función se definen las caracteristicas de los dispositivos conectados.

La señal de cada uno de los circuitos que intervienen trabajan con un voltaje de transmisión predefinido. Una señal se considera como "encendida" (ON), en un circuito de control o un cero lógico: cuando el voltaje en los circuitos se encuentran en el rango de +3 a +15 volts. Si el voltaje se encuentra entre -3 a -15 V. Se considera un uno lógico y apagado (OFF) para el circuito de control. Cuando el voltaje se encuentra en el rango de +3 a -3 V., es una región de transición que no tiene ningún efecto sobre la condición en los circuitos.

Tomando en cuenta que para unir al microcontrolador con la computadora personal, se debe de definir el tipo y la técnica de transmisión, asi como el puerto por donde se van a comunicar ambos dispositivos. Debido a que el microcontrolador solo cuenta con dos puertos, uno de ocho bits y otro de 6 bits, se eligió el puerto serie, que trabaja con una transmisión asíncrona, empleando el formato estándar de No Retorno a Cero (NRZ, por sus siglas en inglés Not Return to Zero) , en donde la transmisión de un byte es precedida por un bit de inicio y seguida por un bit de alto (8N1).

La interface de comunicación empleada al elegir el puerto serial de la computadora es semidúplex, es decir, la transmisión de datos puede ser en ambas direcciones, pero en una sola dirección a la vez.

Para iniciar el envío de la información se creó un programa en lenguaje C, en el cual se habilita la interrupción que maneja el puerto serie de las computadoras. La interrupción catorce es la que maneja a este puerto, posteriormente se emplearon las funciones uno y dos de la misma, para recibir o transmitir datos. Dentro de la función cero se configuró la velocidad de transmisión de datos de 1200 bauds, es decir, la computadora transmitirá 1200 bits por segundo. No se seleccionó el bit de paridad y cada carácter estará formado por ocho bits. El algoritmo empleado para cumplir con esta función se describe en el párrafo siguiente:

Una vez habilitado el puerto serie de la computadora personal se procedió a crear el programa que simula el puerto serie en el microcontrolador. Para realizar el programa se utilizaron dos líneas del puerto B del microcontrolador, el pin PB_0 se utilizó para transmitir y el PB_1 para recibir la información enviada de la computadora; para indicar que el microcontrolador esta listo para iniciar alguna función se colocó un led en el pin que corresponde al PB_2 del puerto.

El programa creado en el mierocontrolador se desarrollo en el lenguaje ensamblador que maneja éste dispositivo. Al inicializar el sistema todos los pines del Puerto B son configurados como salidas, excepto el PB_1, que se conforma como entrada, posteriormente se almacena un dato en los pines de salida en el microcontrolador para que el transmisor se encuentre en un nivel alto hasta que éste vaya a transmitir y el led encienda para indicar que el usuario puede ejecutar alguna función desde la computadora, en caso de que éste no encienda, se debe de verificar el voltaje de la fuente, si se encuentra en el voltaje de 5 volts, se debe de oprimir del reset para inicializar el programa. El sistema se encuentra divido en tres subrutinas principales:

- Recepción. En este procedimiento se reciben los bits enviados por la computadora y que conforman a un carácter. Debido a que el pin PB_1 se configuró como entrada, el microcontrolador se encuentra verificando que dentro de este pin no haya recibido información proveniente de la computadora, cuando detecta la presencia de los bits enviados, estos los recibe uno por uno y los va almacenando en una localidad de memoria temporal, en donde se va a guardar el programa que va a ejecutarse.

- Transmisión. Esta subrutina envía información a la computadora. Generalmente esta función se utiliza cuando el microcontrolador manda los datos contenidos dentro de las localidad de memoria o da un resultado de algún proceso que se le ha asignado. Para efectuar la transmisión, el programa envía el valor de cero al PB_0, para indicarle a la computadora que éste va emitirle información. Para efectuar el traslado de los datos, éstos se van colocando en una localidad de memoria temporal y se van transmitiendo cada uno de los bits que conforman a los caracteres.

- Tiempo. Este método se emplea para hacer que el microcontrolador corra a la misma velocidad que la computadora y de esta forma no perder la información que se envía o transmite de una terminal a otra.

El algoritmo que genera la recepción y la transmisión, así como el tiempo que debe de esperar para efectuar alguna de las opciones antes mencionadas se muestra a continuación.

```
Configura al Puerto B

HAZ MIENTRAS que no cambie ningún bit del puerto

   SI PB1 cambia su estado.

      HAZ MIENTRAS no termine la recepción

         HAZ de y=1 hasta 231
            no operes
         FIN HAZ

         HAZ de c= 0 hasta 8
               recibe el bit
               verifica el valor del bit
               almacena en una localidad
               aplica corrimiento a la izquierda a la localidad
         FIN HAZ

         envía el dato uno al Puerto B
      FIN HAZ MIENTRAS
   FIN SI

   SI PB0 se habilita con un cero
      el dato del acumulador guardalo en una localidad

      HAZ MIENTRAS transmisión no haya terminado

         HAZ de y=1 hasta 144
            no operes
         FIN HAZ

         HAZ de c= 1 hasta 9
            obtén valor del bit a enviar
            coloca dato del bit en el PB0
            aplica corrimiento a la izquierda a la localidad
         FIN HAZ
      FIN HAZ MIENTRAS
   FIN SI

   regresa a verificar el estado del puerto
```

ALGORITMO 2.2. RUTINA DE RECEPCIÓN Y TRANSMISIÓN

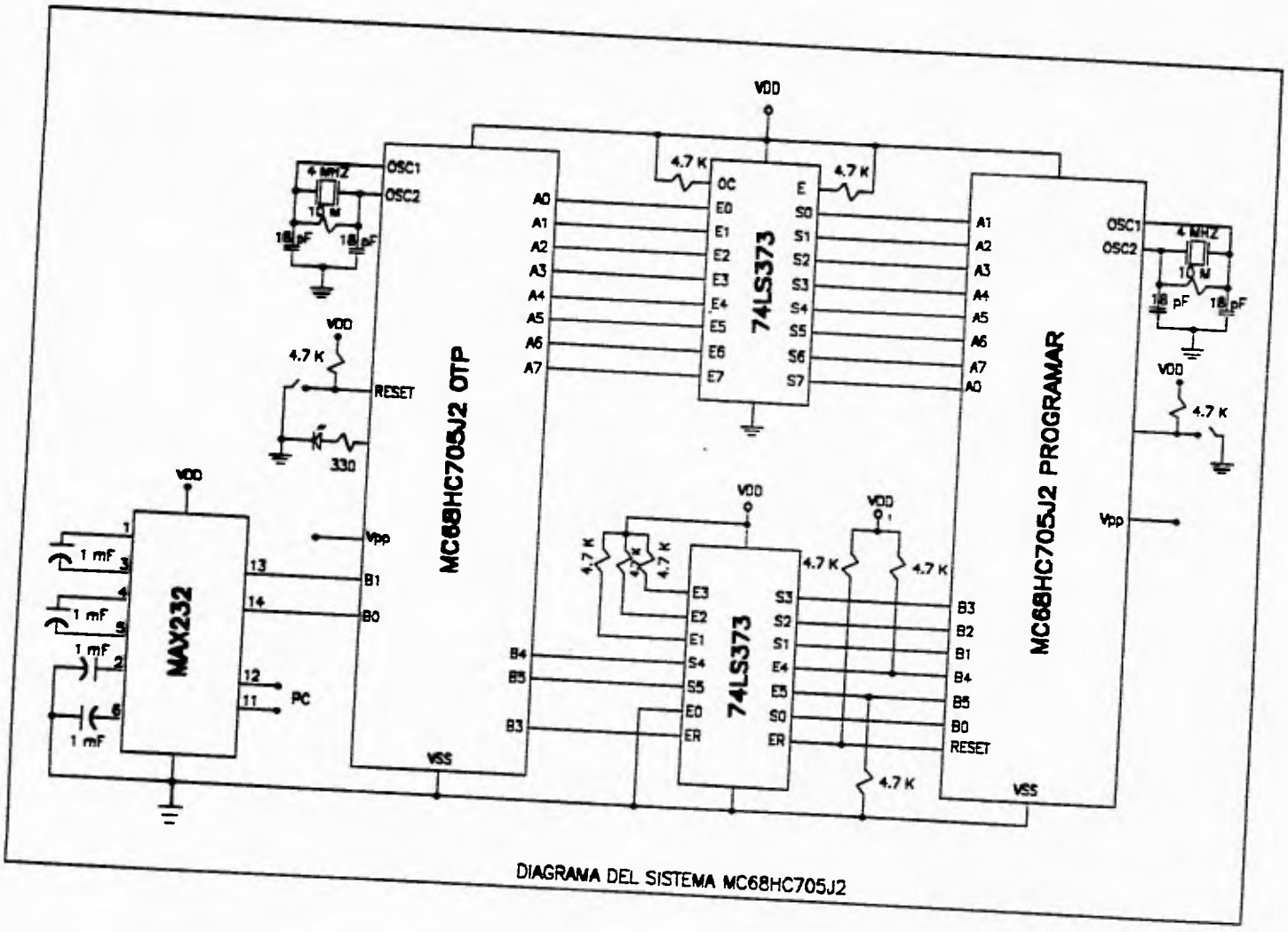


DIAGRAMA DEL SISTEMA MC68HC705J2

CAPÍTULO IV

ENSAMBLADOR

El ser humano trabaja con el sistema de numeración decimal, mientras que los microcontroladores manejan el sistema de numeración binario, por lo que permite que toda la información pueda ser representada por un conjunto de dígitos, los cuales pueden ser unos o ceros; en donde el uso de éstos dígitos puede representar un voltaje lógico, una señal o el estado de encendido o apagado.

A pesar de que los microcontroladores manejan el sistema de numeración binario, trabajan con un código especial que les ayuda a representar información alfabética o instrucciones.

En la base binaria los dígitos a la izquierda son dos veces más grandes que los que se encuentran a la derecha; el primer número a la derecha del entero binario es el dígito uno, el siguiente número a la izquierda del primero es el segundo dígito, el siguiente es el dígito cuatro, por lo tanto el que continua es el dígito ocho y así sucesivamente. Con la siguiente ecuación se determinan los números sucesivos:

$$\text{No. Dígito} = 2^n \qquad n = 0, 1, 2, 3, \ldots$$

Los microcontroladores trabajan con números binarios de 8,16,32 dígitos. El inconveniente de trabajar con esta base es la gran cantidad de dígitos con que el ser humano tiene que trabajar, para dar solución a éste problema se creó el sistema de numeración hexadecimal, que representa a cuatro dígitos binarios, por lo tanto un número binario de ocho dígitos puede ser representado por dos dígitos hexadecimales.

En la siguiente tabla se muestra la relación que existe entre las bases decimal, binaria y hexadecimal. Se puede observar que los tres sistemas de numeración utilizan diferentes formas para representar el mismo número. En la base hexadecimal las letras A, B, ..., F se utilizan para representar a los valores del diez al quince, porque cada dígito hexadecimal puede representar 16 cantidades diferentes.

DECIMAL	BINARIA	HEXADECIMAL
0	0000 0000	0
1	0000 0001	1
2	0000 0010	2
3	0000 0011	3
4	0000 0100	4
5	0000 0101	5
6	0000 0110	6
7	0000 0111	7
8	0000 1000	8
9	0000 1001	9
10	0000 1010	A
11	0000 1011	B
12	0000 1100	C
13	0000 1101	D
14	0000 1110	E
15	0000 1111	F
16	0001 0000	10
17	0001 0001	11
100	0110 0100	64
255	1111 1111	FF

TABLA 3.1 RELACIÓN ENTRE LAS BASES DECIMAL, BINARIA Y HEXADECIMAL

Para identificar al sistema de numeración decimal del hexadecimal, al sistema hexadecimal se le asignará el símbolo de $ antes de la cantidad, por ejemplo:

$ 0400

Este número en sistema decimal representa el valor de 1024.

Al emplear el sistema de numeración hexadecimal se crearon los códigos de operación y posteriormente el conjunto de instrucciones, que le indican al microcontrolador los pasos que tiene que seguir para realizar una función determinada. Los códigos de operación se encuentran en código de máquina, es decir, un valor hexadecimal representa a una función determinada que va ejecutarse. El trabajar con el código de máquina implica aprenderse todas las funciones asignadas a cada código de operación, por tal motivo se crearon los mnemónicos para facilitar el trabajo. Los mnemónicos son instrucciones que le indican al microcontrolador la función que debe realizarse al emplear esta instrucción. Al usar los mnemónicos se tuvo que crear un programa que realizara la traducción de éstos mnemónicos a código máquina. El programa que realiza la conversión se llamó ensamblador y el programa que se escribe para que el ensamblador traduzca se llama lenguaje ensamblador.

Un ensamblador es un programa, en el cual las instrucciones reflejan la estructura interna de la computadora y permiten al programador comunicarse directamente con los acumuladores, direcciones códigos de funciones.

El lenguaje ensamblador surge como consecuencia del desarrollo de la programación, al aumentar el tamaño de los programas y el número de subrutinas de bibliotecas disponibles. Cuando un programa muy largo se tiene que codificar generalmente lo divide en subprogamas o rutinas, las cuales pueden ser dependientes o independientes; los escribe, traduce y los prueba por separado. Posteriormente el traductor debe seguir la pista de todas las referencias entre los diversos subprogramas y subrutinas con la finalidad de generar un programa que pueda ser procesado, es decir, el programa debe estar en condiciones de unir todas las partes para dar un único resultado, por tal motivo de aquí surge el nombre de ensamblador.

A las subrutinas o rutinas se les conoce como ensambladores modulados, los cuales generalmente trabajan en dos pasos. En la primera pasada el ensamblador examina el programa en lenguaje ensamblador construyendo una tabla de símbolos y acumulan todas las definiciones de símbolos que se encuentran en la rutina para efectuar la traducción en el segundo paso.

La principal ventaja de dividir un programa en subprogramas, es el tiempo de proceso de los parámetros de ensamblaje porque es proporcional al cuadrado del número de instrucciones del programa. Los ensambladores de rutinas o de subprogramas son los que pueden realizar la incorporación automática de subrutinas de bibliotecas y el enlazamiento de las diversas partes de un programa.

Un ensamblador debe de trabajar con tres tipos de entidades en el programa fuente:

1. Entidad Absoluta. No depende del lugar en que se almacene el subprograma en memoria, como pueden ser los códigos de operación, numéricos y cadenas de valores constantes. Esta entidad coloca la primera dirección. La traducción es independiente del encadenamiento con otros subprogramas.

2. Entidad Definida Externamente. Es usada dentro de un módulo, pero no se define dentro de éste. El valor de los símbolos no puede ser conocido hasta que todo el programa sea enlazado.

3. Entidad Relativa. Es la información que esta definida o se hace referencia en ese subprograma y su dirección absoluta de almacenamiento en memoria dependerá de la posición del subprograma que es independiente de la combinación de otros subprogramas.

Para obtener un programa objeto, es necesario encadenar todos los subprogramas entre sí y con las subrutinas de biblioteca que sean necesarias. Este procedimiento es realizado por un programa de servicio denominado "cargador".

El cargador funciona de la siguiente forma. Por cada rutina que entra en forma semicompilada anota la dirección de memoria en que almacena la primera instrucción, el llamado origen del subprograma. Según se vaya almacenando la rutina va transformando la información localizable, es decir, toda aquella información referida a direcciones, en información absoluta, sumándole el origen y sigue la pista de los símbolos globales que se definen en ella.

Cuando termina con un subprograma, si existe otro, determina el nuevo origen para éste. Si al terminar de cargar todas las subrutinas existe todavía algún símbolo no definido, explora la biblioteca de subrutinas para incorporar las que necesite el programa.

El encadenamiento es realizado por un consolidador, cuya salida es un programa en binario completamente compacto, el cual es capaz de ser almacenado y convertido en direcciones absolutas por un introductor localizador en el momento de su carga en memoria. Por este motivo el programador no tiene control sobre la localización en memoria de sus datos y programas.

CARACTERÍSTICAS DEL LENGUAJE ENSAMBLADOR.

1. Las direcciones son simbólicas.
2. Traducción de las instrucciones en lenguaje ensamblador a lenguaje máquina de una en una.
3. Existen instrucciones no traducibles a código máquina llamadas pseudoinstrucciones que sirven para control del programa.

INSTRUCCIONES Y PSEUDOINSTRUCCIONES

Instrucciones Simbólicas. Una instrucción en ensamblador consta de los campos: etiqueta, código de operación, operandos y comentarios.

La etiqueta es un símbolo que añadido a una instrucción permite que se pueda referenciar simbólicamente en el programa. El valor que toma es el de la dirección a la cual la instrucción va a ser almacenada. No es necesario que todas las instrucciones tengan etiqueta. Ejemplo:

DIRECCIÓN	ETIQUETA	CÓDIGO DE OPERACIÓN	OPERANDOS	COMENTARIOS
		ORG	$0090	
0096 86 80		LDA	#$80	; carga el acumulado con 80
0098 B7 05		STA	$0005	; almacena en el Puerto B
009B 96 00	ET:	LDA	$00	; carga el acumulador
009E B7 96		STA	$96	; almacena en la localidad $96

EJEMPLO 3.1 IDENTIFICACIÓN DE ETIQUETAS

Para nuestro ejemplo el campo etiqueta es ET y el valor que toma es la dirección $009B.

El código de operación es generalmente un símbolo, aunque en algunos sistemas es un entero octal, hexadecimal o decimal que indica la operación que debe realizar. Este código es imprescindible. Para el ejemplo 3.1. el código de operación es 86, que equivale a un LDA.

El campo operando es un campo de dirección de datos que puede estar dividido en varios subcampos. Cada subcampo puede ser un símbolo o una expresión algebraica de símbolos y números enteros con las operaciones: suma, resta, multiplicación, división y lógicas.

El comentario no es obligatorio, además de no ser procesado por el ensamblador generalmente se encuentra después de un punto y coma. Las instrucciones pueden ser de formato fijo, formato libre o formato mixto.

En los sistemas que emplean instrucciones de formato fijo, los campos son reconocidos por el ensamblador de acuerdo a la posición que ocupan. Un ejemplo de este tipo de formato se muestra a continuación:

COLUMNAS	1 a 5	7 a 10	12 a 40	42 a 72
CAMPO	Etiqueta	Código de Operación	Operandos	Comentarios

Los sistemas generalmente son de formato libre. Para este formato es necesario utilizar delimitadores para separar los diferentes campos y subcampos. Estos delimitadores pueden ser comas, espacios en blanco, símbolos especiales tales como #, $.

Las instrucciones de formato mixto tienen la etiqueta en una posición fija; el código de operación al igual que el campo etiqueta esta en una posición fija y los campos restantes se encuentran separados por delimitadores.

PSEUDOINSTRUCCIONES EXPLICATIVAS.

Las pseudoinstrucciones no producen ningún efecto directo en el programa objeto, pero ayudan al control de la operación de ensamblaje. Estas instrucciones no se traducen a lenguaje máquina. Un ejemplo de este tipo de instrucciones son el principio y fin de un programa, los cuales le indican al ensamblador el inicio y final de su trabajo.

PROGRAMA ANALIZADOR

Las instrucciones son recogidas por la computadora de una en una y generalmente, se almacenan provisionalmente en un registro, hasta que el ensamblador termina su trabajo. El proceso de ensamblaje se divide en tres etapas:

a) Análisis de los símbolos entrados. Identificar los campos de los símbolos y almacenar su nombre simbólico.

b) Transformación de los nombres simbólicos en códigos y direcciones binarias.

c) Ensamblar o insertar los campos binarios obtenidos de una instrucción en un solo registro.

El análisis es efectuado por un programa llamado **Analizador o explorador** (scaner en inglés), este programa busca en la cadena de símbolos los elementos de separación de los diversos campos, tales como: etiquetas, códigos de operación, operandos y comentarios, para separarlos y reconoce los diversos nombres simbólicos.

La operación más importante en un ensamblador es el tratamiento de tablas. Existen dos tipos de tablas: Las tablas fijas y las tablas variables.

Tablas Fijas. Estas tablas se encuentran divididas en dos tipos:

a) Tabla de Código de Operación. A cada nombre simbólico le corresponde un código de operación fijo, que se encuentra en la tabla de código de operación llamada E5_BAS5. El ensamblador busca la correspondencia en la tabla y da el resultado.

b) Tabla de Pseudoinstrucciones. Está formada por todos las pseudoinstrucciones pertenecientes al lenguaje ensamblador.

Tablas Variables. Son las tablas de símbolos que se crean en el primer paso y se utilizan en el segundo paso. En el caso del código de dirección no puede encontrarse prefijados los códigos, debido que el símbolo es escogido libremente por el programador. El ensamblador debe atribuirle los correspondientes códigos de acuerdo con el esquema que utilice para direccionar los datos y las instrucciones. Las direcciones son relativas al origen del programa y según vaya reconociendo las etiquetas va creando la tabla de símbolos de direcciones de instrucciones.

Crea otra tabla de símbolos indefinidos en la que va colocando las direcciones, cuando encuentra una instrucción de definición del símbolo o cuando ha terminado la traducción de todo el programa.

El primer paso de un ensamblador tiene por misión el análisis de las sentencias o instrucciones. Las etapas del primer paso son las siguientes:

1. Leer sentencia o instrucción.

2. Analiza sentencia o instrucción

a) Tratamiento de etiquetas.

- Buscar en tabla de símbolos
- Insertar en tabla de símbolos

b) Tratamiento de código de operación

- Buscar en tabla de código de operación y actualizar campo de dirección
- Escribir código de operación.
- Buscar en tabla de pseudoinstrucciones y hacer tratamiento de la pseudoinstrucción.
- Análisis del operando.

* Almacenar en tabla de símbolos.
* Buscar en tabla de símbolos.
* Sustituir por dirección en tabla de símbolos.

En algunos analizadores la función de reconocer los diversos campos lo efectúa un programa especial llamado reconocedor.

Si la dirección de un operando contiene expresiones aritméticas es necesaria la intervención de una nueva subrutina, llamada reconocedor y evaluador de expresiones aritméticas, que efectúa las operaciones necesarias para determinar la dirección absoluta o relativa. El resultado del analizador depende del tipo de campo analizado:

a) Si es un campo de operación, el resultado es el código de operación binario.

b) Si es símbolo de dirección, el resultado es un código binario de dirección.

c) Si es símbolo de localización de campo (etiqueta), el resultado es una definición de símbolo.

TABLAS DE SÍMBOLOS.

Una tabla de símbolos es un conjunto de pares ordenados (S_i, T_i) en los que el primer elemento S_i es el símbolo y el segundo T_i el valor asociado, es decir, la traducción correspondiente dentro de la tabla de símbolos se consideran dos códigos distintos: el código de operación y el código de pseudointrucciones. En el código de operación se conocen los elementos que la forman desde el principio y no se pueden modificar, sin embargo en el código de pseudoinstrucciones se va definiendo sobre la marcha.

Existen varios tipos de ensambladores, entre los cuales se encuentran los siguientes:

1. *Ensambladores de un Paso.* Los cuales pueden ser de dos tipos:

   a) Ensambladores que generan una salida en binario.

   b) Ensambladores que producen una salida simbólica binaria.

2. *Ensambladores de Dos Pasos.* La principal tarea durante el primer paso es extraer del programa fuente todas las definiciones de símbolos y crear las correspondientes tablas. El objetivo del segundo paso es obtener una visión semicompilada, es decir, una visión simbólica-binaria del programa o rutina que se está ensamblado; además de las tablas de uso para el cargador y la información necesaria para la localización de las variables.

3. *Ensambladores de Dos Pasos con Estructura de Superposiciones.* Recibe este nombre porque sus segmentos se superponen durante la ejecución.

DESARROLLO DEL ENSAMBLADOR

Dentro del sistema desarrollado se generó un **Ensamblador Sencillo**. La descripción general de las funciones de los dos pasos para el sistema creado es la siguiente:

Primer Paso. Se definen los símbolos. Inicialmente el ensamblador procede a analizar las cadenas de entrada para convertirlas en un conjunto de campos y buscarlos o añadirlos en la tabla correspondiente. En esta fase se suele efectuar un análisis sintáctico de las sentencias del programa fuente, con la finalidad de detectar los posibles errores. La definición de los símbolos depende del tipo de instrucción. Una definición es una etiqueta, por lo tanto automáticamente el contador de direcciones refleja la dirección relativa del símbolo. En las tablas se almacena el símbolo, un identificador de tipo símbolo y el valor, si se puede determinar en ese momento.

Al finalizar el paso uno inicia la subrutina postanalizadora, que de acuerdo con el indicador asociado a cada símbolo en las tablas, llena los valores.

Segundo Paso. En este paso el programa fuente nuevamente se lee. Al leer una instrucción , ignora aquellas que fueron procesadas completamente en el primer paso. El código de operación se traduce de acuerdo con la tabla que le corresponde. Determina si las cantidades y direcciones se encuentran en modo absoluto, localizable, efectuando cálculos o evaluaciones necesarios y genera los valores binarios correspondientes.

TABLAS Y LÓGICA DEL ENSAMBLADOR.

El ensamblador de dos pasos maneja dos tablas principales internas: la tabla de código de operación, E5_BAS5 y la tabla de símbolos, E5_ETIQ. La tabla de operación se utiliza para examinar los códigos de operación mnemónicos y traducirlos a su equivalente en lenguaje máquina. La tabla de símbolos almacena las direcciones asignadas a las etiquetas.

En el ensamblador es necesario un contador de localidades, CON, que es una variable empleada como ayuda en la asignación de direcciones. El valor inicial de CON se asigna a la dirección de inicio que se especifica en la proposición ORG. Después de procesar cada proposición fuente, se suma a CON la longitud de la instrucción ensamblada. De esta forma cada vez que se alcanza una etiqueta en el programa fuente, el valor de CON proporciona la dirección asociada a la etiqueta.

La tabla E5_BAS5 contiene el código de operación mnemónico, el equivalente de éste código en lenguaje máquina, información sobre el formato y el número de bytes para cada instrucción. Durante el paso 1, la tabla E5_BAS5 se utiliza para examinar, confirmar los códigos de operación del programa fuente y para traducir los códigos de operación a lenguaje máquina. En el paso 2, debe tener información de E5_BAS5 para conocer el formato de instrucción que se debe manejar para ensamblar la instrucción y cualesquiera peculiaridades del código objeto de instrucción.

La información de la tabla E5_BAS5 está predefinida cuando se escribe el ensamblador, en lugar de cargarla en la tabla durante la ejecución. E5_BAS5 es una tabla estática, es decir, no se le pueden añadir o eliminar instrucciones.

La tabla de símbolos E5_ETIQ incluye nombre y dirección de cada etiqueta del programa fuente, junto con banderas de indicación de condiciones de error, por ejemplo, un símbolo definido en dos lugares. Esta tabla contiene información con respecto al tipo, dirección y código de máquina para cada instrucción, así como todas las instrucciones. Durante el primer paso del ensamblador, se introducen las etiquetas en la tabla E5_ETIQ a medida que se encuentran en el programa fuente, junto con sus direcciones asignadas tomando encuenta a CON; durante el segundo paso, se buscan en esta tabla los símbolos empleados como operandos para obtener las direcciones que se van a insertar en las instrucciones ensambladas.

Las figuras 3.1 y 3.2 muestran el flujo lógico de los dos pasos definidos para el ensamblador.

PASO 1.

```
Inicia
Lee la Primera Línea

Si  x = ORG entonces
    Inicia
    Guarda #operando como dirección inicial
    Asigna a CON la dirección inicial
    Escribe la línea en un archivo intermedio
    Termina
De otro modo
    Asigna a CON el valor de cero

Mientras la longitud del archivo intermedio > 0 haz
    Inicializa variables auxiliares
    Busca símbolos en instrucciones
    Verifica error

    Si encuentra comentario entonces
        Obtener instrucción sin comentario
        Verifica error
        Asignar modo de dirección en instrucción
        Verifica error
        Encontrar etiqueta
        Coloca banderas
        Busca saltos

        Si encuentra saltos entonces
            Coloca banderas para identificar
            Busca instrucción generada con el modo de direccionamiento

            Si se encuentra entonces
                Verifica longitud de instrucción
                Verifica error

                Si las banderas de etiquetas están encendidas entonces
                    Inserta etiqueta en tabla E5_ETIQ
                    Inserta la dirección de etiqueta
                    Inserta el tipo de etiqueta
                    Incrementa contador

                Si las banderas de salto están encendidas entonces
                    Inserta nombre de etiqueta a saltar en E5_ETIQ
                    Inserta la dirección del salto
                    Inserta tipo de salto

    Si encontró error entonces
        Despliega mensaje
```

FIGURA 3.1 ALGORITMO PARA EL PASO 1 DEL ENSAMBLADOR

PASO 2.

```
Seleccionar tabla E5_ETIQ
Ordenar tabla
Obtener primer nombre de etiqueta

Si tipo etiqueta = E entonces
   d= Dirección etiqueta
De otro modo

   Si d < > 0

      Si tipo de etiqueta = 1
         Restar la dirección de etiqueta a d
         Verificar si el valor obtenido esta en rango [-129 a 128

         Si esta dentro del rango entonces
            Obtener la dirección a saltar
         En otro caso
            error
      En otro caso
         Asigna el valor a una variable
   En otro caso
      error

   Si se duplica etiqueta entonces
      error

   Si no hay error entonces
      Reemplaza identificador de saltos con valor obtenido

Si no hay error entonces
   Coloca mensaje Ensamblado Correctamente
En otro caso
   Coloca mensaje Error al Ensamblar
```

FIGURA 3.2 ALGORITMO PARA EL PASO 2 DEL ENSAMBLADOR

Después de que el programa se ha escrito, éste debe convertirse a un código de máquina para que la CPU pueda ejecutar posteriormente. Para realizar la conversión a lenguaje máquina el ensamblador lee al mnemónico del programa fuente y produce una versión de código máquina del programa en una forma en la que pueda ser almacenado dentro de la memoria del microcontrolador.

El ensamblador también produce un archivo compuesto, es decir, genera un archivo donde se muestra al programa fuente original y la translación a código objeto, a este archivo se le conoce con el nombre de **Listado del Programa** y se genera con una extensión LST. Este listado es utilizado como parte de la documentación para el programa y además es empleado cuando se desee ejecutar un archivo a pasos. La figura 3.3 muestra el listado resultante de un programa cuando es ensamblado bajo este formato. Los comentarios se suman después de que el programa se ha ensamblado y no se toman en cuenta cuando se realiza la ejecución a pasos.

```
                        ORG $0700
0700  A6FF              LDA #$FF    ; Carga el Acumulador
0702  B704              STA  $04    ; Almacena FF en DDR del Pto. A
0704  B700              STA  $00    ; Almacena FF en el Puerto A.
0706  AD06        E:    BSR RET     ; Salta a la Subrutina Ret
0708  4A                DECA        ; Decrementa el Acumulador
0709  26F9              BNE E       ; Brinca Etiqueta E si no es igual a cero
070B  CC0DBD            JMP $0DBD   ; Salta a la dirección $0DBD
070E  AEFF              LDX #$FF    ; Carga Registro Índice con FF
0710  CD0E1E      RET: JSR $0E1E     ; Salta a la Dirección $0E1E en donde
0713  CD0E1E      V:   JSR $0E1E     ; se encuentra instrucción NOP y retorna
0716  5A                DECX        ; Decrementa el Registro Índice
0717  26F7              BNE V       ; Brinca Etiqueta V si no es igual a cero
0719  81                RTS         ; Retorna a donde se llamó a  Subrutina
```

FIGURA 3.3 LISTADO DE UN PROGRAMA CONTADOR

La figura 3.4 nos muestra algunas líneas del listado anterior, las cuales se encuentran asociadas a un nombre.

```
                              ORG     $0700
0700        A6FF              LDA     #$FF    ; Carga al Acumulador con FF
0702        B704        ET:   STA     $04

Dirección   Código Objeto  Etiqueta  Campo   Operando        Comentarios
```

FIGURA 3.4 EXPLICACIÓN DEL LISTADO DE UN PROGRAMA

Después de que el programa se ha escrito, éste debe convertirse a un código de máquina para que la CPU pueda ejecutar posteriormente. Para realizar la conversión a lenguaje máquina el ensamblador lee al mnemónico del programa fuente y produce una versión de código máquina del programa en una forma en la que pueda ser almacenado dentro de la memoria del microcontrolador.

El ensamblador también produce un archivo compuesto, es decir, genera un archivo donde se muestra al programa fuente original y la translación a código objeto, a este archivo se le conoce con el nombre de **Listado del Programa** y se genera con una extensión LST. Este listado es utilizado como parte de la documentación para el programa y además es empleado cuando se desee ejecutar un archivo a pasos. La figura 3.3 muestra el listado resultante de un programa cuando es ensamblado bajo este formato. Los comentarios se suman después de que el programa se ha ensamblado y no se toman en cuenta cuando se realiza la ejecución a pasos.

```
                        ORG $0700
0700   A6FF             LDA #$FF     ; Carga el Acumulador
0702   B704             STA  $04     ; Almacena FF en DDR del Pto. A
0704   B700             STA  $00     ; Almacena FF en el Puerto A.
0706   AD06      E:     BSR RET      ; Salta a la Subrutina Ret
0708   4A               DECA         ; Decrementa el Acumulador
0709   26F9             BNE E        ; Brinca Etiqueta E si no es igual a cero
070B   CC0DBD           JMP $0DBD    ; Salta a la dirección $0DBD
070E   AEFF             LDX #$FF     ; Carga Registro Índice con FF
0710   CD0E1E    RET:   JSR $0E1E    ; Salta a la Dirección $0E1E en donde
0713   CD0E1E    V:     JSR $0E1E    ; se encuentra instrucción NOP y retorna
0716   5A               DECX         ; Decrementa el Registro Índice
0717   26F7             BNE V        ; Brinca Etiqueta V si no es igual a cero
0719   81               RTS          ; Retorna a donde se llamó a  Subrutina
```

FIGURA 3.3 LISTADO DE UN PROGRAMA CONTADOR

La figura 3.4 nos muestra algunas líneas del listado anterior, las cuales se encuentran asociadas a un nombre.

```
                                ORG      $0700
0700        A6FF                LDA      #$FF     ; Carga al Acumulador con FF
0702        B704        ET:     STA      $04
```

Dirección	Código Objeto	Etiqueta	Campo	Operando	Comentarios

FIGURA 3.4 EXPLICACIÓN DEL LISTADO DE UN PROGRAMA

La primera línea muestra una directiva del ensamblador. El mnemónico ORG, le dice al ensamblador, donde inicia el programa, es decir, la dirección de inicio de la primera instrucción siguiente. después de la directiva ORG. La directiva ORG es usada en un programa para indicarle a un ensamblador donde debe colocar las diferentes partes del programa en lugares específicos de la memoria. Para seleccionar una localidad de memoria apropiada es necesario dar un inicio al programa en el rango de la RAM o de la memoria EPROM.

En el listado del programa los dos primeros campos: Dirección y Código Objeto son generados por el ensamblador y los siguientes cuatro campos: Etiqueta, Mnemónico, Operando y Comentarios son escritos por el programador. El campo etiqueta ET es referida en otra instrucción . Cuando se utiliza una instrucción de salto donde se hace referencia a una etiqueta, la CPU realiza una diferencia entre el valor del PC, el cual contiene la dirección de la localidad de memoria donde se efectúa el llamado a la etiqueta; y la dirección de la localidad de memoria donde se encuentra la etiqueta. La instrucción de campo son los mnemónicos que hacen un llamado para almacenar o extraer información de los registros. El mnemónico LDA almacena datos al acumulador. De este mnemónico existen seis diferentes instrucciones para almacenar datos en el acumulador, esta información se requiere conocer antes de que el ensamblador elija el código correcto, para que la CPU use durante la ejecución del programa.

El campo operando provee información acerca de la localidad de memoria especifica o valor para ser ejecutado por una instrucción. El ensamblador usa ambas instrucciones: las de Campo y las de Operando, para determinar el código especifico para la instrucción. Las diferentes especificaciones para los operandos son llamados *modos de direccionamiento.* La sintaxis del campo es diferente para cada modo de direccionamiento, así el ensamblador puede determinar la sintaxis del operando correcto de acuerdo al modo empleado. En el ejemplo el operando del campo es #$FF, por lo que el ensamblador interpreta al #$FF como un modo de direccionamiento inmediato y le corresponde el código objeto $A6. Si el operando fuera únicamente $00 la interpretación del ensamblador sería un modo de direccionamiento directo, porque se encuentra localizado en la primera página de la memoria, es decir, se localiza dentro del rango $0000 a $00FF y el código que le corresponde es $B6.

El campo Comentario, éste campo no lo usa el ensamblador cuando efectúa el traslado del programa a código máquina. Éste campo es usado por el programador para documentar su programa. El punto y coma indica que el resto de la línea es un comentario.

El ensamblador genera un ***Archivo de Código Objeto*** que recibe el nombre de **S-record**. Un archivo S-record es un archivo de texto ASCII, el cual puede ser visto por otro editor de texto o por un procesador de palabras. Cada línea del archivo S-record es un registro. Cada registro inicia con la letra **S** seguida por un código numérico que puede ser un uno o un nueve. El registro **S1** es el registro principal de datos. Un registro ***S9*** es usado para indicar el final del archivo S-record. La figura 3.5 muestra la sintaxis generada por el ensamblador al seleccionar el formato del registro S1.

S1	13	0090	6FFB704A6FEB705B601A4012616A601	5D
Tipo	Longitud	Dirección	Datos en Código Objeto	Suma de Verificación

FIGURA 3.5 SINTAXIS DE UN REGISTRO S1

Todos los números generados en el registro S-record se encuentran en código hexadecimal. Dentro del formato S19, los dos primeros caracteres de cada línea representan el tipo de campo, los cuales pueden ser S1 que representa la información del programa y el S9 que nos indica el final del formato, dentro de esta línea no hay código del programa. La longitud del campo esta representado por el siguiente byte, éste campo se obtiene sumando los pares de dígitos hexadecimales del registro, excluyendo el tipo y la longitud del campo. La dirección del campo esta formada de 16 bits y es donde se almacena el primer byte en la memoria. Cada par de dígitos hexadecimales en código máquina representa un dato de 8 bits, el cual es almacenado sucesivamente en las localidades de memoria. El campo de suma de verificación es un valor de 8 bits que representa el complemento a uno de la suma de todos los bytes en el registro S-record con excepción del tipo y el campo de suma de verificación. Este campo es utilizado para garantizar que los datos se transmiten sin error. Al generar el valor después de realizar la suma se envía con los demás datos. En el extremo de recepción se calcula una nueva suma de verificación y se compara con la transmitida, si los datos no son iguales indica que existe un error en la transmisión y es necesario volver a mandar la información.

En la figura 3.6 se muestra el archivo S-record resultante que se obtiene al ensamblar el programa ejemplo del contador de la figura 3.3. En el archivo S-record observamos que al registro S1 se le asigna la dirección $0700. Al desplazarnos a la derecha se encuentra el dato $13 para la dirección $0700, $0D para la dirección $0710 y finalmente encontramos el registro S9 que indica el final del archivo S-record.

```
S1130700A6FFB704B700AD064A26F9CC0DBDAEFF6F
S10D0710CD0E1ECD0E1E5A26F781F1
S9030000FC
```

FIGURA 3.6 SINTAXIS DE UN REGISTRO S19

CAPÍTULO V

DESARROLLO DEL SISTEMA COMPATIBLE CON EL KERNEL BÁSICO

El objetivo de la creación de este sistema es hacer más fácil la comunicación del usuario con el microcontrolador, para cumplir con esta finalidad fue necesario apoyarnos en una computadora personal, la cual se conecta directamente al microcontrolador por medio del puerto serie.

En esta sección se describirá detalladamente todo el software compatible con el kernel básico, explicaremos como está estructurado y el flujo del programa.

La mayoría de los programas fueron diseñados en el compilador Clipper, excepto el programa que inicializa al puerto de la computadora personal, que se realizó en lenguaje ensamblador. El sistema esta formado por un menú principal, que se encuentra dividido en cuatro partes principales:

- Manejo del Ensamblador.
- Manejo de Disco.
- Manejo de Memoria.
- Utilerías.
- Salida al Dos.

MANEJO DEL ENSAMBLADOR

Esta opción nos permite utilizar el ensamblador desarrollado para que el usuario pueda crear y verificar sus propios programas, sin la necesidad de apoyarse en un ensamblador externo. Dentro del manejo del ensamblador existen las opciones de editar y crear un archivo en lenguaje ensamblador, que nos permiten hacer uso directamente con el ensamblador. Las otras opciones solo nos permiten verificar el buen funcionamiento del programa generado así como realizar una copia de los archivos que el usuario elija.

Para manejar el ensamblador es importante que el usuario conozca el área de memoria RAM y EPROM, en donde puede generar su programa, así como las instrucciones del microcontrolador. Esta sección está constituida por varias opciones:

- *Editar un Archivo con Extensión ASM.* Esta subrutina nos permite abrir un programa que ha sido generado anteriormente en el ensamblador del sistema o en otro que emplee las mismas instrucciones y área de memoria que el microcontrolador MC68HC05J2.

  Cuando el usuario elige la edición de un archivo, se muestra una ventana con los diferentes programas que tienen esta extensión. Para moverse dentro de la ventana se emplean las teclas de desplazamiento hacia arriba o abajo según corresponda el caso; si el usuario encuentra el archivo de interés debe colocarse en él y oprimir la tecla de aceptar (enter) para editarlo.

El algoritmo generado para la creación de esta subrutina se muestra a continuación:

```
Asignación de  Variables
Declaración del Arreglo para la extensión ASM
Asigna Variable "n" al Arreglo

Si n = 0
   Archivo no encontrado
   Retorna al Menú
Fin

Si n > 0

  Muestre Mensaje de Selección
   Coloque Archivos Extensión ASM
   Asignación de Variable  "n1"

   Si n1=0
      Regresa al menú
   Fin

   fic = Archivo Seleccionado
   Haz Edición con el Archivo fic
Fin
```

ALGORITMO 1. EDITA UN ARCHIVO CON EXTENSIÓN ASM

- *Generar un Archivo con Extensión ASM.* Al elegir está alternativa se presentará un editor de textos en donde el usuario puede escribir un programa en lenguaje ensamblador. El programador puede generar su archivo utilizando letras mayúsculas, minúsculas o ambas. Cuando el programa se ha terminado de escribir y se desea verificar si no existe ningún error, se debe de seleccionar la tecla F3 que permite ensamblar el programa generado, en caso de que exista algún tipo de error, se desplegará en la parte inferior la línea en donde se encuentra, así como el tipo de error. El ensamblador puede detectar los siguientes tipos de errores:

a) Inicio. Este error es un indicativo de que la directiva de ORG no aparece por alguna de las siguientes causas:

- No se encuentra en el inicio del programa.

```
      LDA #$00
      STA $05
E:    STA $01
      INCA
      BEQ E
```

- Está asignado como un comentario

```
; ORG $0090
  LDA $00
  STA $90
```

- Se encuentra después de una serie de instrucciones que no son definidas como comentarios.

```
Prueba de error
ORG    $0090
LDA    #$80
STA     $05
```

b) Dirección. Si después de la indicación de origen ORG no se asigno el carácter de $ para identificar la dirección en la cual inicia el programa, el ensamblador marcará error de Dirección. En el ejemplo siguiente se puede observar que después del origen y antes de la dirección no se ha colocado el indicativo $ para referirse al inicio del programa.

```
ORG 0700
```

c) Extensión. Cuando al usuario se le olvida asignar la localidad de memoria en la cual va a iniciar el programa creado o al no asignar las localidades de memoria a las que se va accesar para almacenar o extraer información. Ejemplo:

```
ORG $
LDA $ 90
INCA
STA $
```

d) Instrucción. Este error aparecerá cuando una instrucción usada en el programa que se creó no se encuentre dentro del conjunto de instrucciones que se manejan para el microcontrolador. Ejemplo:

```
STAA $00
```

Si se examina el conjunto de instrucciones se puede verificar que esta instrucción no existe, la forma correcta para usar esta instrucción es STA.

e) Sintaxis. Este error se activa bajo los siguientes criterios:

- Cuando a los comentarios no los antecede el indicativo punto y coma.

```
ORG $0090
LDA #$ 00          Carga el acumulador con cero
```

- Si después del nombre de las etiquetas no se asigno dos puntos.

```
    ORG $0090
    LDA  #$FF
    STA    $04
E   STA  $01
    DECA
    BNE E
```

- Al hacer referencia a una etiqueta, pero sin dar el nombre a ésta antes de los dos puntos.

```
    ORG $0090
    LDA #$FF
    STA  $04
:   STA  $01
    DECA
    BNE E
```

f) Fuera de Rango. Al ejecutar una instrucción de salto relativo, el ensamblador debe verificar si la dirección a la cual va a dirigirse se encuentra dentro del rango de -128 a 127 localidades de memoria. Si esta instrucción no se encuentra dentro de este rango se marca el error fuera de rango.

g) No Existe Etiqueta. Si al hacer referencia a una etiqueta, ésta no se encuentra asignada en alguna parte del programa, se comete el error de no existe etiqueta.

```
ORG $0090
LDA  #$FF
STA   $04
STA   $01
DECA
BNE E
```

h) Etiqueta Duplicada. Cuando un mismo nombre se le asigna a más de una etiqueta, el ensamblador va detectar la presencia de un duplicado de etiquetas.

```
      ORG $0090
      LDA #$00
      STA  $04
   E: LDA  $01
      CMP  #$80
      BNE E
      INCA
      BEQ E
   E:  JMP $DBD0
```

i) Fuera de Memoria. Para que el usuario puede bajar un programa, éste se debe de encontrar en el área de la memoria RAM o en la memoria EPROM. En el caso de que el programa sea más grande que la memoria RAM, se genera un error de fuera de memoria, así como también si no se encuentra en el área de la memoria RAM la cual se encuentra a partir de la dirección $0090 hasta $00FF o en rango de la memoria EPROM de $0700 a $0EFF.

Algoritmo empleado para la creación de un nuevo programa.

```
Asignación de Variables
Lectura del nombre de Archivo

Si nombre > 0
   d= verifica si hay extensión

   Si d>0
     omite extensión asignada
      nombre = nombre + extensión ASM

   Entonces
      nombre = nombre + extensión ASM
   Fin

   Abre Editor con el nombre asignado
Fin
```

ALGORITMO 2. CREACIÓN DE UN ARCHIVO CON EXTENSIÓN ASM

- *Ejecución Paso a Paso.* Esta opción es muy útil cuando nuestros programas son muy largos y no pueden ser almacenados dentro de la memoria RAM porque esta solo acepta un programa aproximadamente de 64 bytes. Si se quisiera probar un programa con mayor capacidad que la memoria RAM se tendría que grabar en la memoria EPROM, pero si al probarlo, el programador se da cuenta de que el programa no corre correctamente tiene que realizar las modificaciones adecuadas, borrar el microcontrolador, y volver a grabar la nueva versión del programa; por lo que se puede observar ambas opciones son una limitante, por tales razones se creó la ejecución Paso a Paso, ya que al elegir esta opción el tamaño del programa no importa y se puede trabajar en la zona de la memoria RAM sin ningún problema, cuando el programa se ha probado y su funcionamiento es el correcto, éste se puede grabar en la memoria EPROM del microcontrolador si el programador así lo desea.

Otra función que se encuentra dentro de esta opción es que el programador puede verificar el contenido del acumulador, registro índice, apuntador de la pila, contador del programa y estado de los puertos, cada que el microcontrolador ejecuta una instrucción.

Dentro de esta opción se encuentran tres posibles formas de correr un programa:

1. Ejecuta instrucciones paso a paso. Al elegir la ejecución a pasos el microcontrolador recibirá la instrucción, la ejecutará y esperará que el usuario oprima la tecla F5 que le indica que pase a la segunda instrucción, el ciclo se repite hasta completar todas el conjunto de instrucciones. Cuando se empleen los saltos a subrutinas o etiquetas se colocan banderas que le indicarán al microcontrolador a donde debe de saltar.

2. Ejecuta las instrucciones hasta encontrar una señal. Si un programador no quiere ejecutar la opción de paso a paso y solo tiene interés por verificar una parte del programa, éste puede colocar una marca con solo oprimir la tecla F6 en el lugar donde desea que se detenga el microcontrolador para que pueda empezar a verificar la falla o el buen funcionamiento del programa.

3. Ejecuta instrucciones continuamente. Si ninguna de las dos opciones anteriores son adecuadas para comprobar el funcionamiento del programa existe la ejecución continua, no espera que el usuario oprima la tecla para indicar que puede continuar con la ejecución del programa a pasos y no busca una señal que le indique en donde debe de detenerse.

El algoritmo empleado para realizar la ejecución a pasos se muestra a continuación:

```
envía instrucción para ejecutar programa
despliega información inicial de los registros
p1 = " "
lee el tipo de ejecución
inicializa variables

HAZ MIENTRAS NO HAY FIN DE ARCHIVO
    pc = dirección inicial
    in = operando
    op = Instrucción
    identifica tipo de instrucción

    SI es salto
          in = byte de la operación
          jm = 0
    EN OTRO CASO
          manda instrucción al MC
    FIN SI

    busca in en tabla

    SI in se encuentra
       pc =  dirección de salto
       manda instrucción al MC
    FIN SI

    recibe información del MC
    incrementa pc
    establece nuevos valores para registros
    salta a la siguiente instrucción
FIN HAZ
```

ALGORITMO 3. EJECUCIÓN A PASOS

- *Copiar un Archivo con Extensión ASM.* Al seleccionar esta opción el usuario puede realizar una copia de un archivo con el mismo nombre o con otro diferente; cuando se ha oprimido la tecla de aceptar dentro de esta alternativa el programa despliega una ventana, en donde se escribirá el nombre con el cual se va a identificar el nuevo programa que va a ser igual al archivo a copiarse. Los pasos a seguir para la realización de este procedimiento se describen a continuación:

```
Muestra los archivos existentes
n= número de archivos
Si no existen archivos
   mensaje
   regresa al menú
FIN SI
s=1
nac = cadena de 14
es = cadena de 14
p=1
HAZ MIENTRAS p <> 0 .AND. n>0
    muestra los archivos existentes
    p= número del archivo seleccionado
    SI p=0
       regresa a menú
    EN OTRO CASO
        despliega nombre del archivo a copiar
        despliega mensaje para copiar
        lee el nombre
        nac = nombre del nuevo programa
        a= busca punto en el nombre
        n= longitud de nac
        SI a=0
          SI la  n >13
             nac = Nombre + extensión FYS
          EN OTRO CASO
             nac = Nombre + extensión FYS
          FIN SI
        EN OTRO CASO
          SI a<9
             n = 8-a
             nac = Nombre + espacios vacíos + extensión
          EN OTRO CASO
              nac =  nombre + extensión
          FIN SI
        FIN SI
        fic = arreglo del tamaño de p
        copia  archivo a nac
    FIN SI
FIN HAZ MIENTRAS
SI n=0
   mensaje archivo no encontrado
FIN SI
regresa al menú
```

ALGORITMO 4. COPIAR UN ARCHIVO

- *Regresar al Menú Principal.* Al seleccionar esta opción el usuario retorna al menú principal, en donde se muestran las cuatro alternativas para trabajar.

MANEJO DE DISCO

Esta alternativa se emplea para editar y ejecutar archivos previamente ensamblados con un formato LEM o S19. El formato LEM guarda un programa dentro de un archivo, éste programa se encuentra en código máquina. Al inicio del archivo se sitúa la dirección inicial y final del programa, posteriormente se escriben las instrucciones y datos del programa. Las direcciones de inicio y final, así como el código de cada instrucción del programa, se encuentran separados por medio del carácter que corresponde al oprimir la tecla de aceptar (enter).

Dentro del Manejo de Disco el usuario puede elegir una de las siguientes alternativas:

- *Leer un Archivo con Extensión LEM.* Esta opción le permite al usuario leer un archivo previamente generado con un formato LEM, el cual posteriormente se puede ejecutar o editar en el microcontrolador. Cuando se ha elegido esta opción se despliega una ventana que contiene todos los archivos con extensión LEM, dentro de ésta el usuario con las teclas de desplazamiento arriba o abajo puede colocarse en el archivo requerido, oprime la tecla de aceptar. Cuando no se encuentra ningún archivo con la extensión antes mencionada se envía un mensaje de que no se ha encontrado ningún archivo con esta extensión. El algoritmo empleado para efectuar este procedimiento se muestra a continuación:

```
Asignación de variables
Busca Archivos con extensión LEM

Si no hay Archivos
   Archivo no encontrado
   Retorna al Menú
FIN

Seleccionar Archivo
Abre el Archivo
Obtención de la Dirección inicial del Programa
Obtención de la Dirección final del Programa
Calcula el valor inicial en decimal
Calcula el Valor final en decimal

cad = texto dentro del archivo
di = valor de di1 en hexadecimal
df = valor de df1 en hexadecimal
```

ALGORITMO 5. LECTURA DE ARCHIVOS CON EXTENSIÓN LEM

- *Leer un archivo con extensión S19.* Esta opción nos permite trabajar con archivos generados con el formato S19, los cuales se obtienen al ensamblar un programa con el mismo ensamblador que maneja este sistema o por cualquier ensamblador que genere este formato. Al leer el programa, éste puede ser editado o ejecutado dentro del microcontrolador. De la misma forma que la opción anterior este despliega un mensaje en caso de que no se encuentren archivos con esta extensión. El algoritmo para la lectura de este forma se describe a continuación:

```
Asigna variables.
Busca archivos con extensión S19

Si no hay Archivos
   Archivos no encontrados
   Retorna al Menú
FIN

Seleccionar un Archivo
Lee el contenido del Archivo
Obtén Dirección Inicial

HAZ MIENTRAS se encuentre la cadena S1 en el Archivo
   cad1 = Conjunto de Instrucciones
   cad  = Archivo leído
FIN MIENTRAS

cad = cad1
Obtén Dirección Final
Retorna al Menú
```

ALGORITMO 6. LECTURA DE ARCHIVOS CON FORMATO S19

- *Edición de un Archivo.* Cuando un programa que ha sido generado con un formato LEM o S19 se desea examinar empleamos la Edición del Archivo. Esta opción muestra las instrucciones en código del ensamblador en la memoria RAM o EPROM del microcontrolador según el área en la cual se haya generado el programa. Si por algún motivo el programa no ha sido leído anteriormente, al seleccionar esta opción se desplegará un mensaje donde se le indica al usuario que no se encuentra ningún programa en el área de memoria.

El algoritmo que representa a la Edición de un Archivo se describe en las siguientes líneas.

```
Despliega Letrero
di1 = Dirección Inicial

SI dirección inicial es de un byte
  convertir en dos bytes
FIN

SI dirección final es de un byte
  convertir en dos bytes
FIN

q = dirección inicial en decimal
r = dirección final en decimal
p = numero de bytes dentro del rango
q1= 1
j= último dígito del valor inicial

Haz de y= 1 hasta p
  coloca dirección inicial
  di = dirección inicial en hexadecimal sumando 16

  SI di es de un byte
    convierte di a dos bytes
  FIN

 MIENTRAS j<16
   coloca q1

   SI q <= r
     l= dato en código de máquina
     incrementa q1

   EN OTRO CASO
     l = espacio vacío
   FIN

   incrementa q
   incrementa j

 FIN MIENTRAS
 j=0
FIN HAZ
di = dirección inicial
despliega información
retorna al menú al oprimir tecla
```

ALGORITMO 7. EDICIÓN DE UN ARCHIVO CON FORMATO S19 O LEM

- *Correr un Programa.* Al generar un archivo con el formato S19 o LEM el usuario puede correr su aplicación después de haber leído el archivo. Los programas que se prueban dentro de esta alternativa solo pueden encontrarse en el área de la memoria RAM, es decir, los programas que se ejecutan bajo esta subrutina tienen un límite de 64 bytes. Cuando un programa que se ha seleccionado previamente no se encuentra dentro del rango permitido para la memoria RAM, el sistema muestra un mensaje donde le indica al usuario el área de memoria permitida para trabajar. La notación que representa a esta subrutina es:

```
n1= Dirección final en  Decimal
n2= Dirección inicial en Decimal
n  = resta n1 de n2
Verifica inicio y fin del programa

SI no se encuentra en el rango
   Despliega mensaje de Error
   retorna al menú
FIN

Checa que el microcontrolador transmita

SI no puede transmitir
   Exhibe mensaje de error
FIN

n= numero de bytes a enviar
Obtén  datos a enviar por el puerto serie

HAZ de k =1 a n
   envía datos por el puerto serie
 FIN HAZ

Obtén instrucción para ejecutar programa

HAZ de k=1 hasta 5
   Envía orden para ejecutar el programa
  FIN HAZ
Retorna al Menú
```

ALGORITMO 8. EJECUCIÓN DE UN PROGRAMA EN EL MICROCONTROLADOR

- *Menú principal.* Esta alternativa nos envía al menú principal con solo oprimir la tecla de aceptar (enter en inglés).

MANEJO DE MEMORIA

Para tener acceso a esta opción es muy importante que el microcontrolador se encuentre conectado a una computadora personal por medio del puerto serial, debido a que esta alternativa nos permite examinar la información que se encuentra almacenada dentro del área de la memoria RAM o EPROM, así como la zona de los vectores y la configuración e información que se encuentra almacenada en la zona destinada a los puertos.

La opción de Manejo de Memoria se encuentra formada por varias opciones las cuales se listan a continuación:

- *Memoria RAM.* Al seleccionar esta alternativa el usuario puede almacenar o extraer información temporal de los 64 bytes de la memoria, es decir, al ejecutar Memoria RAM el sistema despliega un menú con diferentes procedimientos:

1. Examinar Memoria. El usuario tiene la posibilidad de verificar la información que se encuentra almacenada en los 64 bytes que ocupa esta memoria. Los valores que toma la memoria RAM al encender la fuente que alimenta al microcontrolador son aleatorios por tal motivo si se realiza una operación con los datos que se encuentran en esta zona, las instrucciones u operaciones que se le indiquen van a cambiar de acuerdo al valor que se encuentre almacenado en ese momento.

   Cuando se elige esta opción la rutina verifica que el microcontrolador no se encuentre ejecutando un programa, extraviado o con alguna falla, en caso de detectar algún problema en el sistema el programa despliega un mensaje de error en donde se le indica al usuario que el sistema se encuentra desconectado. Algunas de las fallas pueden observarse a simple vista, es decir, el hardware que forma al sistema contiene un led testigo que se encuentra destellando e indica que el microcontrolador esta operando en forma correcta, si por algún motivo éste led deja de centellar es un indicativo de que el sistema no se encuentra en la posibilidad de ejecutar ninguna función que se asigne, unas de las causas por las que este led deja de brillar son:

   a) Por una serie de instrucciones que se están ejecutando y las cuales no han sido terminadas. En este caso solo es necesario esperar a que termine de correr el programa que se ha ejecutado o oprimir el botón del reset para inicializar el sistema.

   b) Si un programa que se ha ejecutado no contiene un retorno a la dirección $ 0DBD, que es la dirección en donde se encuentra el programa para preparar al sistema y pueda recibir o enviar la información, el microcontrolador no se encontrara en la posibilidad de ejecutar ninguna instrucción hasta que se oprima el botón de reset.

   c) Cuando la fuente de alimentación del microcontrolador se encuentre en un valor inferior a los 5 volts el microcontrolador no va ha inicializar el sistema.

d) Por un cable que se encuentre conectado en forma errónea al microcontrolador y la computadora personal que se emplee para la realizar la comunicación entre los dispositivos.

El algoritmo que se empleo para el desarrollo de esta subrutina se escribe en la líneas siguientes:

```
Verifica que el MC pueda transmitir
manda llave por puerto serie
envía orden de transmisión
di = dirección inicial de la memoria RAM
df = dirección final de la memoria RAM
 q = valor de di en decimal
 r = valor de df en decimal
 p = número de bytes a recibir del microcontrolador
 asigna arreglo del número de bytes de p
 q1=1
 j= valor decimal del número menos significativo de di

 HAZ de  i=1 hasta p
       coloca datos de memoria para diagrama
       m = di + 16 en decimal
       di = número en hexadecimal de m

       SI di = 1 byte
         di = convierte a di en dos bytes
     FIN

     HAZ MIENTAS j<16

        SI q<=r
           envía llave
           k= dato transmitido del microcontrolador
            l= convierte a k a hexadecimal
           mensaje de cuantos bytes se transmitieron
           incrementa a q1
        EN OTRO CASO
            l= asigna cualquier valor
        FIN

        SI i<=7
           imprime numeración en la parte superior
        FIN
         incrementa q
         incrementa j
     FIN MIENTRAS
     asigna a j el valor de cero
FIN HAZ
 regresa al menú
```

ALGORITMO 9. EXAMINA MEMORIA RAM

2. Cargar Datos. Esta opción es empleada para almacenar información en código máquina, es decir el usuario puede escribir un programa con el código de operación que representa a las instrucciones manejadas. Si al escribir el programa que se desean depositar en la RAM se comete un error en el momento de escribirlo, se puede elegir nuevamente esta opción para cambiar las direcciones de memoria en donde ocurrió el error, para efectuar esta operación al ejecutar nuevamente esta altemativa el programador solo necesita dar la dirección inicial y final en donde se haya cometido la equivocación, no es necesario que escriba nuevamente todo el conjunto de instrucciones. La secuencia de pasos que se siguió para generar esta subrutina se describen a continuación:

```
inicialización de variables
di1 = primer byte de la dirección inicial
di2 = segundo byte de la dirección inicial
di = di1 + di2
df1 = primer byte de la dirección final
df2 = segundo byte de la dirección final
df = df1 + df2
d1= byte menos significativo de di
d2= byte menos significativo de df
transmite valor de las direcciones
q= valor de di en decimal
r = valor de df en decimal
p = r - q
q = número de bytes a llenar
s=1
j= número menor del primer byte + 1

HAZ de  i=1 hasta q
    o  = valor en decimal de di + 16
    di = valor en hexadecimal de o

    HAZ MIENTRAS  j<=16
       m= dato a almacenar
       ar[s]= valor en decimal de m
       incrementa s
       incrementa j

       SI s>p
          j=17
       FIN
    FIN MIENTRAS
    j=1
FIN HAZ

HAZ de  i=1 hasta p
    envía datos contenidos en la base temporal
FIN HAZ
regresa al menú
```

ALGORITMO 10. ALMACENA INFORMACIÓN EN LA MEMORIA RAM

3. Correr un Programa. Esta opción ejecuta un programa. Para realizar este procedimiento es necesario que el programa a probarse se encuentre dentro del área de la memoria RAM. Para que un conjunto de instrucciones quede contenida en el área de memoria, el usuario puede introducir el programa haciendo uso de la opción de Cargar Datos en la Memoria RAM. Al emplear esta forma el usuario debe de convertir el programa que se encuentra escrito en lenguaje ensamblador a código de máquina para el microcontrolador, así como también debe de obtener el número de bytes que debe desplazarse el apuntador de la pila sobre las localidades de memoria, cuando haya un salto a una subrutina o etiqueta. Para crear esta función, los pasos a seguir fueron:

inicialización de variables
Imprime Dirección Inicial
Imprime Dirección Final

di1 = primer byte de la dirección inicial
introduce di1
verifica validez de di1

di2 = segundo byte de la dirección inicial
introduce di2
verifica validez de di2

di = di1 + di2
df1 = primer byte de la dirección final
introduce df1
verifica validez de df1

df2 = segundo byte de la dirección final
introduce df2
verifica validez de df2

df = df1 + df2
d1= byte menos significativo de la dirección
verifica que el MC pueda transmitir

Si no puede transmitir
  mensaje de error
  retorna
FIN

envía la dirección inicial del programa a ejecutar

ALGORITMO 11. EJECUTA UN PROGRAMA

4. Limpiar Memoria. Cuando se enciende el microcontrolador la zona de la memoria RAM se inicializa con información aleatoria, es decir, nunca va acontecer la misma información cuando se encienda el sistema. Si esta área no sitúa en las localidades de memoria valores de cero, un programa que accese a alguna localidad de memoria de esta zona va tomar el valor que en ese momento contenga la localidad, siempre que no se haya inicializado ésta; por tal motivo es importante que a la memoria de acceso aleatorio se asignen valores de cero. Para que ésta función se realice fue necesario crear el conjunto de pasos siguientes:

```
verifica que el MC pueda transmitir
envía llave
manda dato cero a las localidades de memoria
Verifica que el MC pueda transmitir
manda llave por puerto serie
envía orden de transmisión
di = dirección inicial de la memoria RAM
df = dirección final de la memoria RAM
q = valor de di en decimal
 r = valor de df en decimal
 p = número de bytes a recibir del microcontrolador
 q1=1
  j= valor decimal del número menos significativo de di
 HAZ de  i=1 hasta p
     coloca datos de memoria para diagrama
     m = di + 16 en decimal
     di = número en hexadecimal de m
      SI di = 1 byte
        di = convierte a di en dos bytes
      FIN
      HAZ MIENTAS j<16
          SI q<=r
             k= dato transmitido del microcontrolador
              l= convierte a k a hexadecimal
              mensaje de cuantos bytes se transmitieron
              incrementa a q1
          ELSE
              l= asigna cualquier valor
          FIN

          SI i<=7
             imprime numeración en la parte superior
          FIN
          incrementa q
          incrementa j
      FIN MIENTRAS
      asigna a j el valor de cero
 FIN HAZ
 Coloca barra de desplazamiento
 regresa al menú
```

ALGORITMO 12. INICIALIZACIÓN DE LA MEMORIA RAM

5. Menú Anterior. Cuando se ha terminado de trabajar con las opciones anteriores, si el usuario elige esta alternativa se traslada al menú de Manejo de Memoria.

- *Memoria EPROM.* El área que comprende a la memoria EPROM se encuentra a partir de la dirección $0700 hasta $0EFF. Dentro de este rango el usuario puede almacenar el programa que ha de ejecutarse. Al elegir esta alternativa, el sistema despliega un menú con las siguientes opciones:

1. Examinar la Memoria EPROM. Esta opción es útil para verificar si esta zona de memoria se encuentra ocupada por algún programa o esta desocupada. Si la memoria EPROM no tiene información almacenada, en la pantalla de la computadora se desplegarán valores de ceros en los bytes que correspondan a ésta; en el caso de que se encuentre uno o más programas en esta área el usuario podrá observarlos en código de máquina. La notación empleada para la realización de este procedimiento es:

```
Inicialización de Variables
di = dirección inicial
df = dirección final
d1= byte más significativo de df
df = byte más significativo de df + FF
q= valor de di en decimal
r = valor de df en decimal
p = número de bytes de (r-q)/16
q1=1
 j= valor en decimal del último dígito de di
HAZ desde i=1 hasta  p
    coloca datos de memoria para diagrama
    m = valor decimal de di + 16
    di = valor hexadecimal de m
    HAZ MIENTRAS  j<16
       SI q<=r
          k = dato transmitido del MC
           l = valor en decimal de k
           incrementa arc
           incremente q1
       EN OTRO CASO
           l= espacio vacío
       FIN SI
       incrementa arreglo
       incrementa q
       incrementa j
    FIN MIENTRAS
    j=0
FIN HAZ
regresa al menú
```

ALGORITMO 13. EXAMINA MEMORIA EPROM

5. Menú Anterior. Cuando se ha terminado de trabajar con las opciones anteriores, si el usuario elige esta alternativa se traslada al menú de Manejo de Memoria.

- *Memoria EPROM.* El área que comprende a la memoria EPROM se encuentra a partir de la dirección $0700 hasta $0EFF. Dentro de este rango el usuario puede almacenar el programa que ha de ejecutarse. Al elegir esta alternativa, el sistema despliega un menú con las siguientes opciones:

1. Examinar la Memoria EPROM. Esta opción es útil para verificar si esta zona de memoria se encuentra ocupada por algún programa o esta desocupada. Si la memoria EPROM no tiene información almacenada, en la pantalla de la computadora se desplegarán valores de ceros en los bytes que correspondan a ésta; en el caso de que se encuentre uno o más programas en esta área el usuario podrá observarlos en código de máquina. La notación empleada para la realización de este procedimiento es:

```
Inicialización de Variables
di = dirección inicial
df = dirección final
d1= byte más significativo de df
df = byte más significativo de df + FF
q= valor de di en decimal
r = valor de df en decimal
p = número de bytes de (r-q)/16
q1=1
j= valor en decimal del último dígito de di
HAZ desde i=1 hasta p
   coloca datos de memoria para diagrama
   m = valor decimal de di + 16
   di = valor hexadecimal de m
   HAZ MIENTRAS j<16
      SI q<=r
         k = dato transmitido del MC
         l = valor en decimal de k
         incrementa arc
         incremente q1
      EN OTRO CASO
         l= espacio vacio
      FIN SI
      incrementa arreglo
      incrementa q
      incrementa j
   FIN MIENTRAS
   j=0
FIN HAZ
regresa al menú
```

ALGORITMO 13. EXAMINA MEMORIA EPROM

2. **Programar la Memoria EPROM.** Al comprobar el buen funcionamiento de un programa que ha sido generado en el área de la memoria RAM, éste puede ser almacenado en la zona de la memoria EPROM, con la finalidad de que el sistema creado no necesite de la computadora personal para ejecutarse.

El usuario puede grabar esta zona de memoria siempre que no haya información en ésta, para verificar que se encuentre vacía en su totalidad o las áreas en las cuales se puede almacenar el programa, se puede examinar la zona con la opción de Examinar Memoria, si al desplegar la información en pantalla se observa que existen valores de ceros en los bytes que corresponden al rango de la memoria RAM, el programador puede emplear las localidades de memoria que se encuentran desocupadas para almacenar su información, tomando en cuenta que si el microcontrolador funcionará independientemente de la computadora, el apuntador de la pila se desplazará a la dirección $0700, por tal motivo, si existen otros programas antes del deseado, éstos serán ejecutados antes que el programa que se desea probar.

Si el microcontrolador tiene información previa almacenada, la cual se desea borrar, se debe de exponer a la luz ultravioleta durante 15 minutos aproximadamente para que se borre todas las instrucciones que se encontraban en la memoria EPROM.

En las líneas siguientes se describe el proceso que se siguió para la pronagramación de la memoria EPROM.

```
Asigna variables.
Seleccionar un Archivo
Obtén Dirección Inicial
cad1 = Conjunto de Instrucciones
cad  = Archivo leído
Obtén Dirección Final
SI tecla = 13
    recibe respuesta
    SI l = 30
      HAZ MIENTRAS cad2 > 0
          ci = valor entero de di
          ce = valor entero de di + 64
          cad1 = los primeros 64 datos de cad2
          cad2 = valores de cad2 - los 64 datos a enviar
          transmite la dirección inicial a programar
          despliega mensaje para desconectar el Voltaje
          envía otro dato diferente de 30
      FIN HAZ MIENTRAS
    EN OTRO CASO
      transmite otro dato diferente de 30
    FIN SI
EN OTRO CASO
   envía otro dato diferente de 30
FIN SI
Regresa al menú
```

ALGORITMO 14. PROGRAMACIÓN DE LA MEMORIA EPROM

- Puertos de E/S. Esta opción fue generada con la finalidad de que el usuario verifique el estado en el que se configuraron los puertos, así como la información que se encuentra en éstos, es decir, verifica si los puertos se encuentran como entrada, salida o entrada/salida y los datos que se encuentran almacenados en estas localidades de memoria. A continuación se describen los pasos que se siguieron para realizar esta función.

```
Despliega letrero de espera.
verifica que el MC pueda transmitir
   Si no puede transmitir
      mensaje de error
      retorna
   FIN
ordena que transmita los datos de los puertos
regresa al menú
```

ALGORITMO 15. VERIFICA EL ESTADO DE LOS PUERTOS

- Zona de Vectores. Al elegir esta alternativa el programador puede observar la información o el estado de los diferentes registros que se manejan en el microcontrolador, estos registros son:

1. COP.
2. Bytes Altos del Temporizador.
3. Bytes Bajos del Temporizador.
4. Interrupciones Externas.
5. Interrupciones de Software.
6. Bytes Altos del Reset.
7. Bytes Bajos del Reset.

Al poder verificar el contenido de estos registros el usuario se da una idea del movimiento que se realiza al invocar una instrucción que afecte a éstos. La rutina empleada para la realización de este procedimientos se muestra en el párrafo siguiente.

```
Despliega Mensaje de espera
verifica que el MC pueda transmitir
envía llave
asigna un arreglo
HAZ desde i=1 hasta 16
    ar[i]= dato transmitido
FIN HAZ
despliega nombre del registro e información almacenada
regresa al menú
```

ALGORITMO 15. EXAMINA LA ZONA DE VECTORES

UTILERÍAS

Esta alternativa fue creada para que la persona que maneja el sistema E5_MENU, pueda realizar algunas funciones del sistema operativo, sin salirse del sistema; dentro de utilerías se puede verificar todos los archivos que se encuentran dentro del disco duro, disco flexible o en un subdirectorio determinado; así como también se pueden seleccionar todos los programas que se desean eliminar o a los cuales se les desea realizar una copia.

Dentro de este menú aparecen las opciones de:

1. *Directorio de Archivos.* Esta sección nos muestra todos los nombres de los archivos que se encuentran almacenados en una área de trabajo determinada. Para que el sistema nos muestre la información correcta se despliega una ventana en donde la persona puede escribir la ruta del directorio de trabajo que desee examinar. Si la información se encuentra en disco flexible solo debe de asignar el nombre de la unidad en donde se encuentran los archivos a buscar.

En esta opción el usuario puede manejar las diferentes formas que se emplean en el sistema operativo para ver los programas que se encuentran dentro de un directorio de trabajo.

Para efectuar este proceso se siguieron los siguientes pasos:

```
Despliega mensaje.
Lee archivos a buscar
Busca el archivo
abre ventana
SI no se encuentra el archivo
   mensaje de error
FIN SI
muestra los programas
regresa al menú principal
```

ALGORITMO 16. DIRECTORIO DE LOS ARCHIVOS

2. *Borrar Archivos*. Nos permite eliminar los programas que no se ocupan dentro de una área de trabajo. Al seleccionar esta opción el sistema despliega una ventana, en donde se especifica el nombre de los archivos que desean desplegar en una ventana, antes de elegir los programas que van a ser eliminados. Dentro de esta opción existen dos formas para eliminar archivos:

a) Todos los archivos. Para activar esta forma solo es necesario que cuando se despliega la ventana para leer los archivos se le asigne un asterisco, punto y un asterisco, para indicarle que elimine todos los programas que se encuentran dentro de este directorio.

b) Elegir archivos. En esta alternativa el usuario solo escribe el nombre o las extensiones de los programas que van a borrarse, para que éstos sean desplegados dentro de una ventana. Para seleccionar el archivo, solo es necesario que el usuario se mueva por la ventana con las teclas de desplazamiento. Cuando el cursor se encuentra sobre el archivo deseado solo es necesario oprimir la tecla de aceptar para anular e archivo. Después de borrar el archivo, los archivos restantes siguen desplegándose en la pantalla hasta que el usuario oprima la tecla de escape.

Los pasos que se siguieron para efectuar este procedimiento se muestran a continuación.

```
Despliega Archivos a Borrar
Lee nombre
n = numero de archivos encontrados
inicializa variables
f= .T.

HAZ MIENTRAS n > 0  .AND. f
    Despliega nombre de los archivos
    p = numero del archivo seleccionado

    SI p=0
      RETORNA
    EN OTRO CASO
      EN CASO DE
          Archivos = *.*
             Despliega mensaje de seguridad
            r = lee respuesta

            SI r = s .OR. r = S
              Borra archivos
               f = .F.
            FIN HAZ MIENTRAS
        FIN EN CASO
        EN OTRO CASO

           HAZ MIENTRAS p>0
              borra archivo seleccionado
              despliega los archivos restantes
              p= número del archivo seleccionado
              f = .F.
           FIN HAZ
        FIN EN CASO
   FIN SI
FIN HAZ

SI  n= 0 .AND. F=.T.
   archivo no encontrado
FIN SI
regresa al menú
```

ALGORITMO 17. ELIMINAR ARCHIVOS DE UNA ZONA DE TRABAJO

CAPÍTULO VI

SISTEMA AUTOMATIZADO DE BOMBEO

Este sistema tiene la finalidad de suministrar agua o cualquier otro líquido a diferentes altitudes, para lograr este objetivo se controlan una o varias bombas para subir el líquido de un tanque a otro de mayor altitud, además se lleva un registro de tiempo de cada bomba, mientras se encuentra operando, con lo cual se puede determinar la cantidad de líquido que se ha suministrado de un tanque a otro.

Frecuentemente este tipo de sistema se emplea en lugares donde se tiene que transportar algún líquido de una región a otra, en donde sus altitudes varían. Por ejemplo, el suministro de agua al Valle de México.

La aplicación generada opera de la siguiente forma:

Como se puede observar en la figura 6.1. se encuentran tres tanques a diferentes niveles, interconectados uno a otro de mayor altitud. En el tanque uno y dos se encuentran conectadas las bombas de suministro A y C, respectivamente. En los tanques dos y tres se instalaron dos sensores de nivel en cada uno, con lo que se determina la cantidad de líquido en cada tanque.

Para controlar la operación del sistema se uso el microcontrolador MC68HC705J2, utilizando los puertos de entrada y salida para el control de los sensores de nivel, el encendido de las bombas, además de la administración del panel de control.

Cada sensor de nivel esta conectado a una entrada del puerto del microcontrolador, a través de una lógica de acoplamiento. Dentro de los dos tanques, los sensores que se encuentran en cada uno, forman un rango de histéresis, en el cual opera o no funciona la bomba que suministra el líquido a dicho tanque.

A través de una salida del puerto del microcontrolador y por medio de un circuito de acoplamiento se controla el encendido y apagado de las bombas de suministro, operando éstas únicamente cuando el nivel del líquido es menor a la posición del sensor inferior y deja de operar cuando ya a almacenado la suficiente cantidad de líquido, es decir, cuando éste rebasa el nivel mayor del sensor superior.

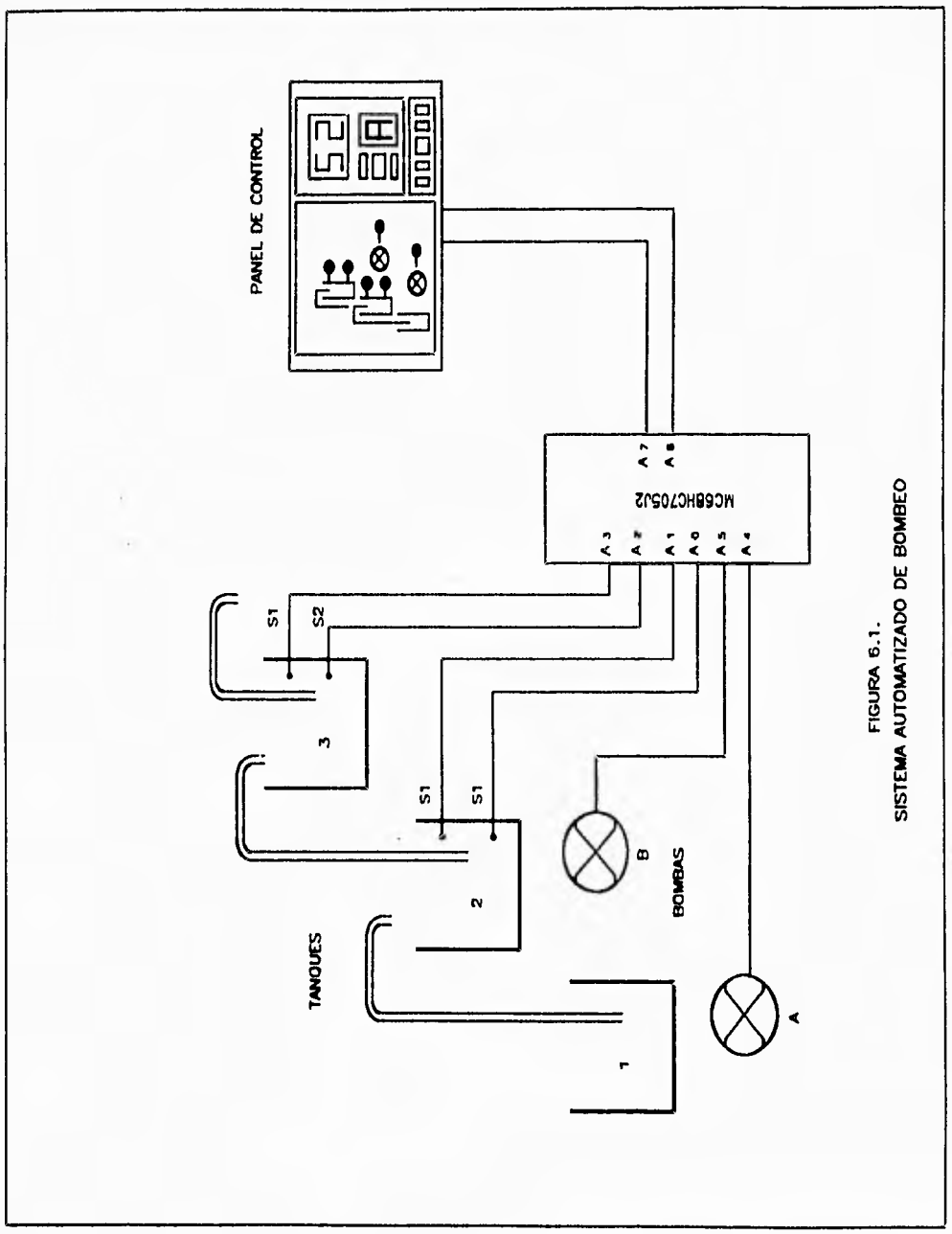


FIGURA 6.1.
SISTEMA AUTOMATIZADO DE BOMBEO

A continuación se muestra una tabla de estados para ambos tanques:

S1	S2	Bomba A
1	0	Apagado
0	0	Apagado
0	1	Encendido
0	0	Encendido

FIGURA 6.2. TANQUE 2

S1	S2	Bomba C
0	1	Encendido
0	0	Encendido
1	0	Apagado
0	0	Apagado

FIGURA 6.3 TANQUE 3

Como se puede observar, el funcionamiento de cada bomba es independiente, por tal motivo, pueden estar encendidas ambas bombas, solo una o ambas apagadas.

En el panel de control se tiene una plantilla que muestra un contador en decimal de dos dígitos, el cual nos indica el tiempo en que la bomba está funcionando, además se tiene un display en donde se despliega un carácter, que nos indica que bomba se está supervisando, y varias luces de control. Este panel es operado a través del microcontrolador por un par de salidas del mismo, que se encuentran conectadas en varios circuitos en serie, que van a ser activados por éstas salidas. El diagrama del circuito completo se muestra en la figura 6.4.

Para el control de este sistema se generó un programa en lenguaje ensamblador, el cual fue almacenado en la zona de la memoria EPROM interna del microcontrolador.

El programa consta de cinco subrutinas y un módulo principal. En el instante de dar un reset al microcontrolador, se inicia la operación del sistema, estando trabajando indefinidamente.

A continuación se describe la función de cada subrutina del programa y posteriormente se lista el código fuente del mismo, así como el código en formato S19, que fueron generados y evaluados con el sistema FASE705.

SUBRUTINAS.

EST: Esta verifica el estado de los pines de entrada del puerto A. La parte baja de este puerto, se configuró como bits de entrada, de tal forma que va a leer un uno lógico si el sensor al cual está conectado esta seleccionado, si se encuentra un cero lógico si no esta seleccionado. A este tipo de verificación se le denomina poleo, porque siempre se esta revisando el estado del dato de entrada.

Cuando se determina el estado del bit de entrada enciende una bandera de control en una localidad de memoria para que posteriormente sea empleada para encender o mantener apagadas las bombas.

CON: Lleva un contador interno para cada bomba. Si se encuentra encendida la bomba y durante el tiempo que permanezca así, se incrementará el contador, realizando una conversión interna entre el reloj E del microcontrolador, con una aproximación de segundos, con lo cual, se determina el tiempo real en que cada bomba esta funcionando.

CUE: Es un contador en decimal y es incrementado cada vez que cumple con el tiempo predeterminado, es decir, si se programa el microcontrolador para que su unidad mínima de conteo de tiempo, sea de un segundo, un minuto, u otro valor. Como en la pantalla del panel de control existen un par de display para efectuar esta cuenta, el valor máximo será hasta 99, por tal motivo, si se rebasa de este dato, se inicia en cero el contador.

PRE: Se preparan las localidades de memoria que son empleadas para almacenar la información referente al estado de cada bomba, de los contadores decimales y las luces de control. Estas localidalidades de memoria se encuentran en la zona de memoria RAM, por lo que cada vez que se inicia el sistema se borra todo el contenido de esta zona.

DES: Esta subrutina sirve para enviar la información contenida en los registros internos del microcontrolador al panel de control a través de dos líneas de salida del puerto. La información se transmite en serie y posteriormente se convierte en paralelo, en el momento de pasar al panel de control.

El módulo principal del sistema se encuentra en un ciclo infinito y éste se encuentra constantemente activando a las subrutinas antes descritas.

Todo el programa generado fue realizado en el ensamblador del sistema FASE705 y posteriormente fue grabado en la memoria EPROM del microcontrolador, utilizando uno de los procedimientos que ofrece el sistema.

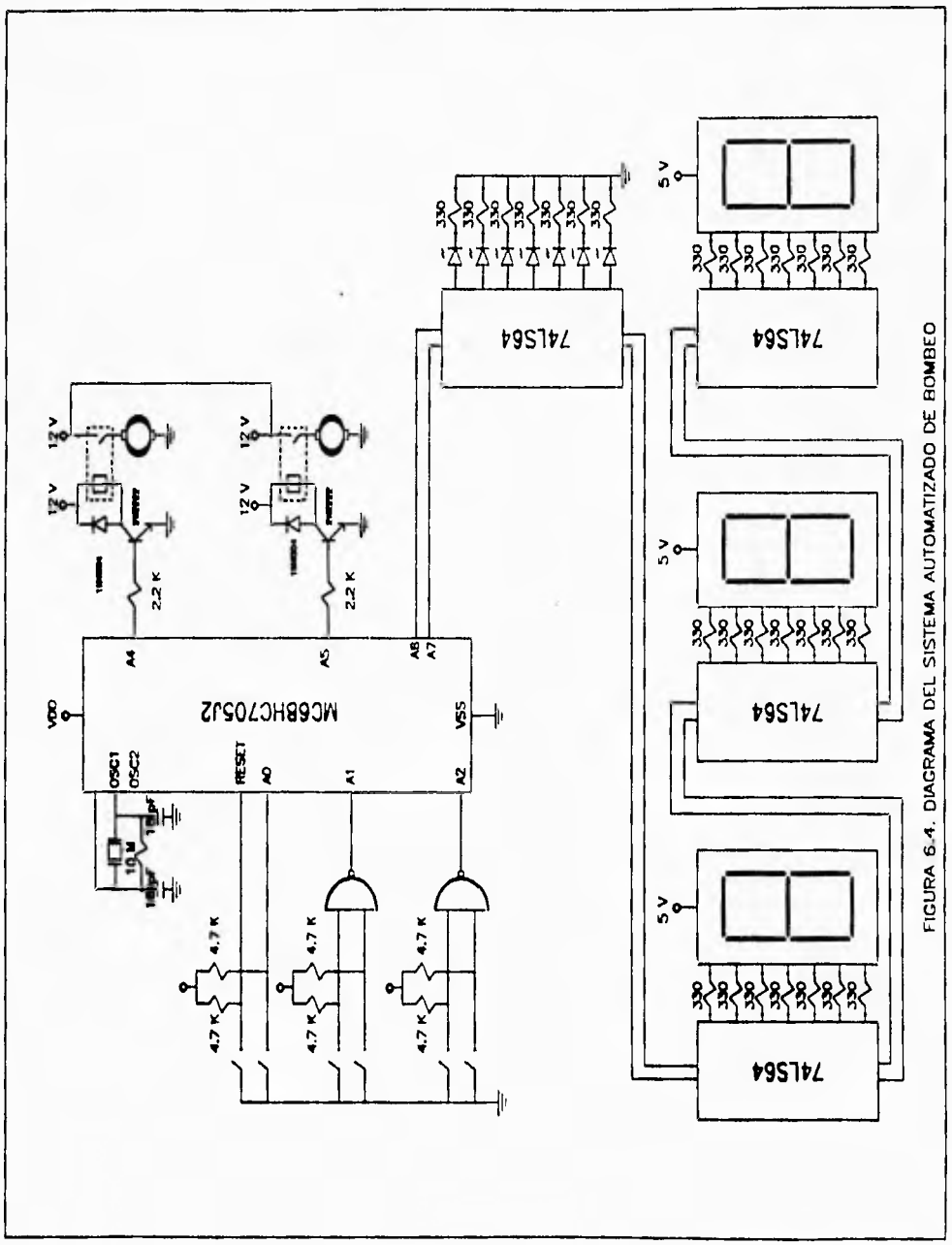


FIGURA 6.4. DIAGRAMA DEL SISTEMA AUTOMATIZADO DE BOMBEO

PROGRAMA EN ENSAMBLADOR
PARA EL SISTEMA DE AUTOMATIZACIÓN DE BOMBEO

```
        ORG  $0700
        LDX  #$90
        LDA  #$00

A1:     STA  $00,X
        INCX
        CPX  #$A0
        BNE  A1
        LDA  #$F0
        STA  $04
        CLR  $00
        LDA  #$FE
        STA  $05
        CLR  $01

A3:     INC  $9F
        LDA  #$FF
        CMP  $9F
        BNE  A2
        INC  $9E
        LDA  #$80
        CMP  $9E
        BNE  A2
        CLR  $9E
        BRSET $07,$93,A4
        BSET $07,$93
        BRA  A2

A4:     BCLR $07,$93

A2:     JSR  EST
        JSR  CON
        JSR  CUE
        JSR  PRE
        JSR  DES
        BRA  A3
```

```
EST:  NOP
      BRSET $00,$00,E1
      BRSET $01,$00,E2
      BRA  E3

E1:   BSET  $04,$00
      BSET  $00,$93
      BSET  $01,$93
      BCLR  $02,$93
      BRA  E3

E2:   BCLR  $04,$00
      BCLR  $00,$93
      BCLR  $01,$93
      BSET  $02,$93

E3:   NOP
      BRSET $02,$00,E4
      BRSET $03,$00,E5
      BRA  E6

E4:   BSET  $05,$00
      BSET  $03,$93
      BSET  $04,$93
      BCLR  $05,$93
      BRA  E6

E5:   BCLR  $05,$00
      BCLR  $03,$93
      BCLR  $04,$93
      BSET  $05,$93
E6:   RTS

CON:  NOP
      BRCLR $00,$93,C1
      INC  $99
      LDA  #$FF
      CMP  $99
      BNE  C1
      INC  $98

C1:   NOP
      BRCLR $03,$93,C2
      INC  $9D
      LDA  #$FF
      CMP  $9D
      BNE  C2
      INC  $9C
C2:   RTS
```

```
CUE:  LDA  #$80
      CMP  $98
      BNE  V1
      CLR  $98
      INC  $96
      LDA  $0A
      CMP  $96
      BNE  V1
      INC  $97
      CLR  $96
      LDA  #$0A
      CMP  $97
      BNE  V1
      CLR  $97
V1:   LDA  #$80
      CMP  $9C
      BNE  V2
      CLR  $9C
      INC  $9A
      LDA  $0A
      CMP  $9A
      BNE  V2
      INC  $9B
      CLR  $9A
      LDA  #$0A
      CMP  $9B
      BNE  V2
      CLR  $9B
V2:   RTS

PRE:  NOP
      BRSET $00,$01,P1
      BCLR  $06,$93
      LDA  #$47
      STA  $92
      LDA  $9A
      STA  $90
      LDA  $9B
      STA  $91
      BRA  P3
P1:   BSET $06,$93
      LDA  #$77
      STA  $92
      LDA  $96
      STA  $90
      LDA  $97
      STA  $91
P3:   RTS
```

```
DES:  LDX  $90
      LDA  $A0,X
      STA  $90
      LDX  $91
      LDA  $A0,X
      STA  $91
      LDX  #$90

D4:   LDA  #$07
      STA  $94
      LDA  $00,X
      STA  $95
D3:   BSET $06,$00
      RSET $00,$95,D1
      CLR  $07,$00
      BRA  D2
D1:   BSET $07,$00
D2:   BCLR $06,$00
      LSR  $95
      DEC  $94
      LDA  #$00
      CMP  $94
      BNE  D3
      INCX
      CPX  #$94
      BNE  D4
      RTS
```

PROGRAMA EN FORMATO S19
PARA EL SISTEMA DE AUTOMATIZACIÓN DE BOMBEO

S1130700AE90A600F75CA3A026FAA6F0B7043F00BB
S1130710A6FEB7053F013C9FA6FFB19F26103C9E55
S1130720A680B19E26083F9E1E9320021F93CD07EC
S11307303FCD076ACD0781CD07BACD07DA20D79D13
S1130740201218001093129315932008190011938 6
S1130750139314939D20121A00169318931B93203D
S1130760081B00179319931A93819D3C99A6FFB116
S1130770992602 3C989D3C9DA6FFB19D26023C9C77
S113078081A680B19826163F983C96B60AB1962663
S11307900C3C973F96A60AB19726023F97A680B1D4
S11307A09C26163F9C3C9AB60AB19A260C3C9B3F69
S11307B09AA60AB19B26023F9B819D1D93A647B72B
S11307C092B69AB790B69BB791200E1C93A677B7B2
S11307D092B696B790B697B79181BE90E6A0B790BF
S11307E0BE91E6A0B791AE90A607B794F6B7951C54
S11307F0001F0020021E001D0034953A94A600B18B
S10C08009426EC5CA39426E0812B
S9030000FC

CONCLUSIONES

Dentro del campo de la electrónica, los microcontroladores cada día toman mayor importancia, debido a que incorporar miles de circuitos dentro de un encapsulado, por tal motivo es muy importante contar con las herramientas que sean fáciles de manejar y que dentro de un solo sistema existan diversas utilerías.

Por esta razón fue desarrollado el sistema FASE705, el cual está enfocado para ser empleado tanto por personas con poca experiencia en el campo de los microcontroladores, así como aquellas usuarios muy experimentados.

Desde el inicio se planteó generar un sistema muy amigable, que fuera transparente para toda las personas que lo usaran y que no tuvieran que emplear equipo y sistemas sofisticados para operar a este dispositivo.

El resultado que se obtuvo, fue un sistema que interactúo entre una computadora personal y la tarjeta de evaluación del microcontrolador. Éste sistema presenta las siguientes características:

- Interface computadora personal y microcontrolador a través del puerto serie.
- Kernel básico de comunicación en el microcontrolador.
- Manejo de un microcontrolador externo en la tarjeta de evaluación.
- Verificación de programas en la tarjeta.
- Programación de un microcontrolador externo.
- Salidas y/o entradas para validar circuitos externos
- Emulación de circuitos de prueba.
- Software manejado por menús de fácil acceso.
- Ensamblador sencillo.
- Generador de archivos con formato S19.
- Rutinas de depuración de programas.
- Rutinas para el manejo de memoria del microcontrolador.
- Utilerías para la administración de archivos.

- Sistema totalmente en español.

- Tamaño reducido de tarjeta.

- Operación con configuración mínima en computadora personal.

- Editor de archivos.

Con este sistema se pretende brindar al usuario una herramienta para la generación de aplicaciones de diversas índoles, tomando como ejemplo el sistema de automatización de bombeo, reflejando las facilidades que el sistema FASE705 ofreció para su elaboración

APÉNDICE

MANUAL PARA EL MANEJO DEL SISTEMA FASE

OBJETIVO.

Con el presente manual, el usuario estará capacitado para realizar actividades de interface entre el microcontrolador y una computadora personal; utilizando las herramientas que el sistema ofrece, como son: manejo de ensamblador, de disco, de memoria y algunas utilerías de administración de archivos.

REQUISITOS PARA EL SISTEMA.

- Computadora personal con capacidad mínima

  * 640 Kbytes en RAM
  * MS-DOS Ver 3.3
  * 1 Mb en Disco Duro
  * Puerto Serie RS-232C

- Fuente Regulada a 5 V.

- Fuente Regulada a 16.5 V.

ADVERTENCIA: Tanto el voltaje de alimentación como el de programación, deben estar regulados de acuerdo a las especificaciones. De no ser así podría dañarse seriamente el circuito.

CONSIDERACIONES.

Este sistema utiliza ventanas que muestran diferentes opciones para el manejo de la información, se hace referencia al uso de ésta utilizando la siguiente terminología.

Señalar. Es la acción de colocar la barra de selección sobre una opción en particular, utilizando las teclas marcadas con flechas direccionales.

Oprimir <tecla> Implica presionar la tecla a la cual se está haciendo referencia, por ejemplo: <Enter>, <ESC>.

Oprimir <tecla> + <tecla>. Nos indica que debemos presionar simultáneamente ambas teclas para realizar una función específica; por ejemplo: <CTRL> + <N>.

INGRESANDO AL SISTEMA DEL MICROCONTROLADOR MC68HC705J2

Este sistema fue desarrollado especialmente para funcionar en el ambiente MS-DOS, por tal motivo para trabajar con éste, se deberá instalar en la computadora personal. para iniciar una sesión de trabajo, el usuario deberá ejecutar el programa FASE705; además su tarjeta de evaluación deberá estar conectada al puerto serie a través de la interface RS-232C y estar alimentada con un voltaje de 5 volts.

A continuación se detallan los pasos para instalar su sistema en la computadora e iniciar una sesión:

1. Encienda su computadora.
2. Inserte el disco marcado con FASE705 en su unidad.
3. Desde el punto indicativo del MS-DOS teclee:

C:\>A: Instala
presione <Enter>

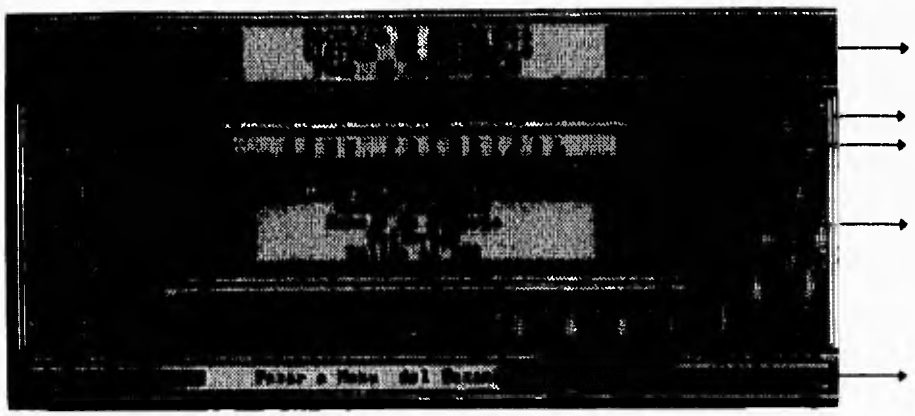
4. Espere a que el sistema termine de instalarse en la computadora.
5. Cuando termine la instalación, en el disco duro se ha generado un subdirectorio de trabajo en su disco duro; para cambiarse a éste teclee:

C:\> CD FASE705
oprima <Enter>

6. Siempre que se desee iniciar una sesión deberá escribir:

C:\> FASE705\FASE705
apretar <Enter>

Después de pulsar la tecla se mostrará en su pantalla varias ventanas que contienen diversa información, la cual se detalla a continuación:



a) Contiene la información de presentación del sistema, ésta ventana estará permanentemente.

b) Dentro de ésta se abrirán una o más ventanas, que nos darán a conocer diferentes opciones y mensajes.

c) Despliega el nombre de la opción que se encuentra habilitada.

d) Contiene las diferentes opciones que representan a un menú en especifico.

e) Muestra mensajes que le sirven al usuario para identificar lo que realiza cada opción.

MENÚ PRINCIPAL

Este menú es la raiz del sistema y a través de las cinco opciones que contiene, se puede expandir a varios niveles. Las opciones que se encuentran dentro del Menú Principal son:

Manejo de Ensamblador. Dentro de esta alternativa el usuario tendrá diferentes herramientas para trabajar con un lenguaje ensamblador propio para la Arquitectura del microcontrolador empleado.

Manejo de Disco. Podrá cargar y editar en la memoria de su computadora programas compatibles con los formatos más comunes.

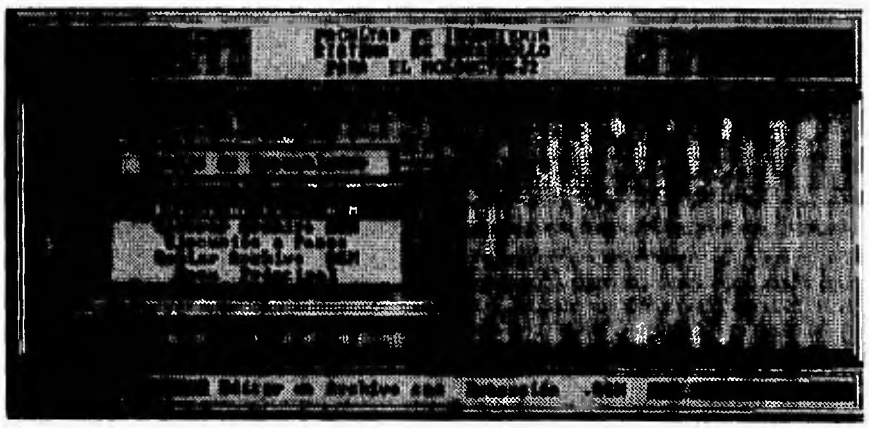
Manejo de Memoria. Realiza tareas de intercomunicación entre su computadora y el microcontrolador, con la finalidad de desplegar en la pantalla la información contenida en diferentes áreas de la memoria del microcontrolador.

Utilerías. Administra la capacidad en su disco duro borrando y copiando archivos, así como también genera la impresión de los programas seleccionados.

Salida al Dos. Termina la sesión de trabajo, regresando al punto indicativo del sistema operativo **MS-DOS**.

MANEJO DE ENSAMBLADOR

El usuario puede elegir esta opción del menú principal, oprimiendo la tecla <Enter>, posteriormente se expone una ventana para manejar el ensamblador. Esta ventana contiene las siguientes alternativas:



EDITAR ARCHIVO ASM.

Al seleccionar ésta, se mostrará una nueva ventana que contiene los archivos con extensión ASM localizados en el directorio activo de su disco; en el caso de no existir algún programa, se despliega un mensaje en la misma ventana para indicar que no se encontró algún archivo con esta extensión.

Si el número de archivos localizados es mayor de cinco, el usuario puede desplazarse en la lista utilizando la flechas direccionales hacia arriba o abajo para señalar el archivo de interés. Cuando se encuentre sobre el programa requerido, pulse la tecla <Enter>, después se abrirá la *pantalla de edición*, la cual contiene la información que se encuentra almacenada en el programa. Para cancelar la operación se debe de oprimir la tecla <ESC>.

GENERAR UN ARCHIVO ASM.

En esta alternativa se crea un nuevo archivo con la extensión ASM. En el momento de seleccionar dicha operación se desplegará una ventana en la cual se debe escribir el nombre del archivo a ser generado. El sistema automáticamente asigna la extensión ASM, inmediatamente después coloca al usuario en la ventana de edición.

EJECUCIÓN A PASOS.

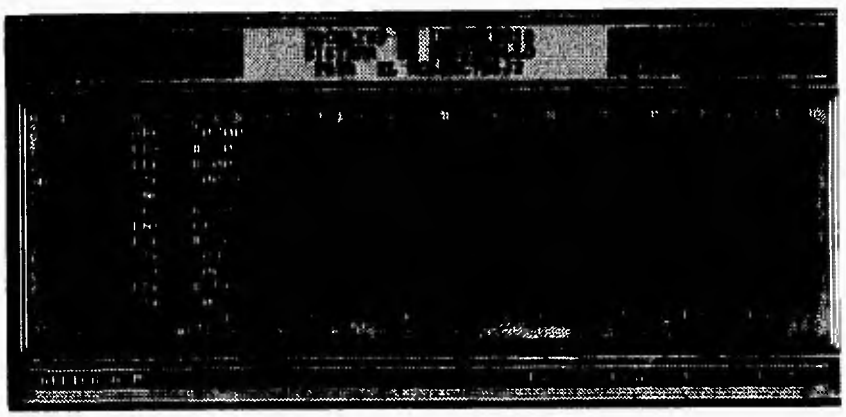
Sirve para ejecutar un programa paso a paso. La ejecución generalmente es empleada para depurar los programas antes de ser grabados en la memoria EPROM. Cuando se elige esta opción se despliegan todos los archivos con extensión LST dentro de una ventana y que pueden ser electos para ser ejecutados. El formato LST se genera a partir de un programa en lenguaje ensamblador y que es generado con un ensamblador que maneje las instrucciones del microcontrolador.

MENÚ PRINCIPAL.

Al terminar de hacer uso de las herramientas que este menú ofrece se puede regresar al menú principal con solo seleccionar esta alternativa.

EDITOR DE ARCHIVOS

El editor fue creado especialmente para trabajar con archivos de formato ASM y sirve para generar, modificar y almacenar en disco los programas que el usuario elija. Dentro de este editor se encuentran las siguientes alternativas:



F2: *SALVAR.* Al oprimir la tecla <F2> se guarda en disco la información del archivo que se encuentra desplegado en ese momento en el editor de textos. Si el usuario realiza alguna modificación al programa y desea salir de éste, se exhibirá un mensaje donde se pregunta si desea grabar las modificaciones realizadas al archivo. En el caso de que se elija que si se desean almacenar los cambios realizados, se salva la información pero no abandona el editor, por tal motivo se debe de teclear <ESC> para abandonar esta pantalla.

F3: *ENSAMBLAR.* Al seleccionar está alternativa se efectúa el ensamblado del programa que se este editando, es decir, nos informa si existen errores de sintaxis en el programa. En el momento de realizar la revisión, se presenta una ventana que contiene información referente al número de líneas ensambladas, el número de etiquetas encontradas en el programa y la asignación de líneas.

En caso de que el programa detecte un error, éste se mostrará en la ventana inferior de la pantalla. Si no detecta ningún error en esa misma sección se despliega el mensaje ensamblado correctamente.

F4: *GENERA S19.* Al elegir esta alternativa se generará un archivo con el formato estándar S19, el cual es muy útil porque se puede transportar los programas a cualquier otro sistema que utilice este mismo formato y que sean las mismas instrucciones que generen a éste.

F5: *GENERA LST.* Este formato es empleado en la ejecución a pasos y no tiene el mismo formato que manejan los ensambladores comerciales. Dentro de éste se muestra la dirección de las instrucciones y operandos, así como los bytes correspondientes a éstos y el código en lenguaje ensamblador.

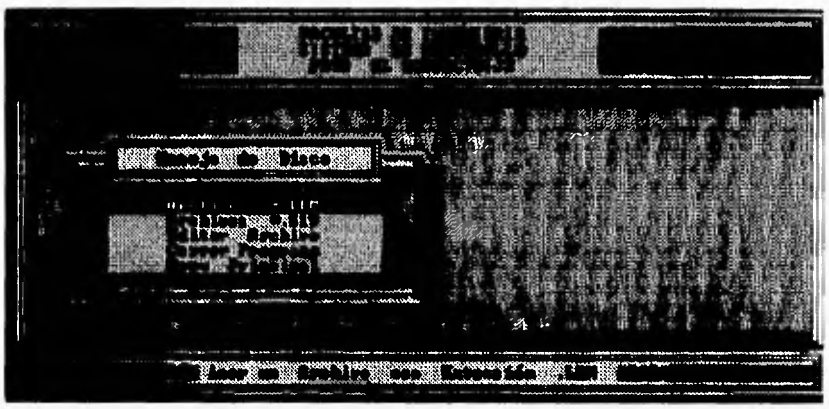
ESC: *SALIR.* Al pulsar esta tecla, regresará al Menú de Ensamblador. Si el archivo que se encontraba dentro del editor fue modificado y no se ha guardado estos cambios, al intentar salirse, desplegará un mensaje, que pregunta si se desea salvar los cambios realizados.

Además de las funciones descritas anteriormente, el editor cuenta con funciones propias para la modificación de los archivos, éstas funciones se detallan a continuación.

<Alt> + 24	Sube una línea
<Alt> + 25	Baja una línea
<Alt> + 27	Palabra a la izquierda
<Alt> + 26	Palabra a la derecha
<Home>	Principio de la línea
<End>	Final de la línea
<Ctrl> + <Home>	Principio del texto
<Ctrl> + <End>	Final del texto
<PgUp>	Página arriba
<PgDn>	Página abajo
<Ctrl> + <PgDn>	Principio del página
<Ctrl> + <PgUp>	Final de la página
<Ctrl> + <W>	Salvar y salir
<Esc>	Aborta y sale
<Ctrl> + <Y>	Borra línea
<Ctrl> + <T>	Borra palabra a la derecha

MANEJO DE DISCO

En el menú de manejo de disco se tienen las funciones relacionadas con la apertura de archivos con formato LEM y S19, los cuales fueron generados anteriormente. Dentro de este menú se encuentran las siguientes opciones:



ARCHIVOS LEM.

Al seleccionar esta alternativa, se muestra una ventana con los nombres de los programas que tienen un formato LEM y que se encuentran dentro del subdirectorio activo. Si dentro del área de trabajo no se encuentra ningún archivo con esta extensión, se desplegará un mensaje de que no existe ningún archivo. Ahora bien si existen programas, el usuario podrá desplazar la barra de selección hasta donde se encuentre el nombre del programa apropiado, posteriormente se debe presionar la tecla <Enter> para leer el contenido de éste. El programa se mantendrá en la memoria RAM de la computadora para su uso posterior.

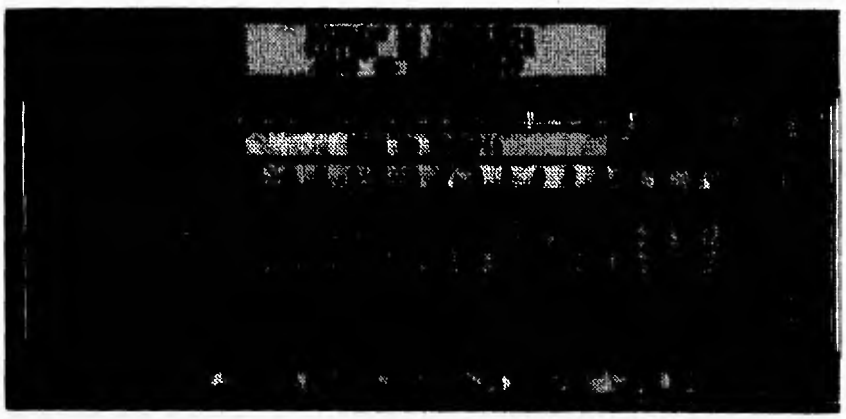
ARCHIVOS S19.

Realiza la misma función que la opción descrita anteriormente, pero ésta muestra los archivos que presentan el formato S19.

EDITAR ARCHIVO.

Después de almacenar en la memoria de la computadora algún archivo leído con formato S19 o LEM, éste puede ser editado al seleccionar está opción. Al editar un archivo se abre una ventana que contiene el mapa de memoria del programa seleccionado. En el lado izquierdo de la ventana se muestra la parte alta de las localidades de memoria. En la parte superior se encuentra el valor menos significativo de la localidad de memoria. Para observar los datos que se encuentran almacenados en una localidad determinada, se debe elegir la intersección entre la columna y el renglón necesario.

Los archivos con formato S19 y LEM contienen la información en bytes, mismos que se muestran en la pantalla y representan una instrucción o un dato. La dirección inicial del programa se puede visualizar con la intersección de la primera columna y el primer renglón, que aparece en el editor. La dirección final se obtendrá a partir de la unión del renglón y la columna, generada por el último byte que se muestra en la pantalla.



Al editar un archivo, la ventana que presenta los bytes, solo puede mostrar 80 bytes como máximo; en caso de que el programa sea mayor de este valor, el usuario podrá observar los demás valores desplazándose por medio de esta ventana por medio de las flechas de desplazamiento hacia arriba o abajo.

CORRER PROGRAMA.

Nos permite correr un programa no mayor de 64 bytes. Para hacer uso de esta alternativa es importante que el sistema este interconectado con la tarjeta de evaluación. Posteriormente de que el programa ha sido seleccionado, se almacena dentro de la memoria del microcontrolador para que éste lo ejecute a una velocidad plena.

Este procedimiento es útil para realizar pruebas de algunas rutinas o programas pequeños. Es importante considerar que las pruebas realizadas se efectúan en el microcontrolador maestro, como consecuencia no se puede utilizar el Puerto B del mismo, debido a que este es empleado para la comunicación con la computadora.

En el caso de que el programa se encuentre fuera del rango de la memoria RAM o su origen se encuentre en los limites máximos de ésta, se desplegará un mensaje de error.

MANEJO DE MEMORIA

En el menú de manejo de memoria se tienen las opciones que permiten manejar directamente el área de memoria. Dentro de esta sección se encuentran las siguientes formas para trabajar con esta zona:

MEMORIA RAM.

El rango de la memoria RAM es a partir de la dirección $0090 a $00CF, en esta zona se pueden escribir programas o subrutinas. Dentro de esta opción el usuario tiene la posibilidad de manejar la opción de examinar los datos que se encuentran almacenados en este rango de memoria, con solo dirigirse a esta opción, por medio de las teclas de desplazamiento y oprimir la tecla de <Enter> del menú, inmediatamente se mostrará una ventana con la información contenida en esta área.

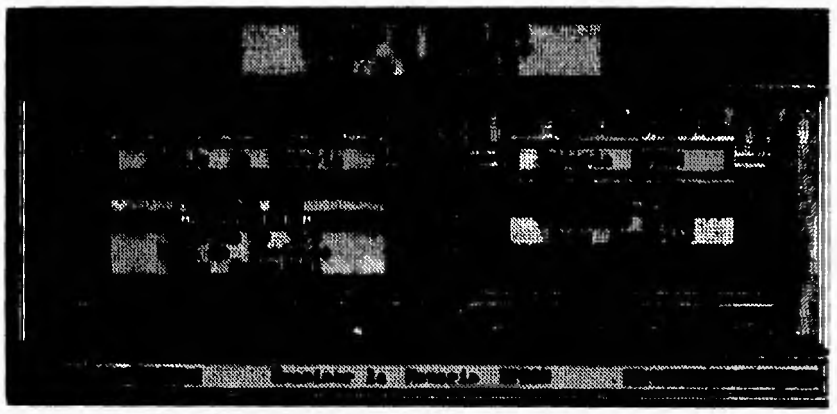
Para cargar datos en la memoria solo es preciso colocarse sobre este título y pulsar la tecla <Enter>, posteriormente se abrirá una ventana, donde el usuario debe colocar la dirección inicial y final de los datos que van a ser almacenados en el área; inmediatamente después de dar estos valores se exhibe una ventana con un mapa de memoria que nos indica la zona asignada para trabajar y el cursor se encuentra localizado en el primer byte.

La alternativa de correr un programa nos permite ejecutar un archivo previamente almacenado en el área de memoria RAM. Para iniciar la ejecución, se debe de introducir la dirección de inicio, la cual aparece en una ventana e inmediatamente el microcontrolador se encargará de iniciar el proceso.

Si se desea inicializar la memoria RAM con valores de cero, con la finalidad de borrar toda la información que se encuentra contenida en esta zona, de tal forma que no afecte la ejecución de un programa, solo es necesario elegir la opción de Limpiar Memoria.

MEMORIA EPROM.

La memoria EPROM está dividida en dos partes, una se encuentra a partir de la dirección $0700 hasta $0EFF y el otro a partir de $0FF0 hasta $FFFF. El primer rango es utilizado para almacenar código y/o datos y el segundo rango es empleado para validar los vectores de reset e interrupción. En el menú de manejo de memoria, se puede examinar esta área con solo elegirla, en la ventana de edición se muestran los bytes que corresponden a cada localidad. El procedimiento nos permite examinar todo el rango de memoria o solo ciertas localidades. Al seleccionar esta alternativa se despliega una ventana, en la cual donde se debe introducir la dirección inicial y final de la zona para ser analizada.



Con la opción de Programar EPROM se grabarán los datos que representen a un programa que se encuentre en formato S19, el cual habrá sido generado previamente. Se recomienda que esta acción se realice únicamente si es necesaria, porque al programar la memoria, esta no podrá ser borrada eléctricamente, solo se puede eliminar su información exponiédola a los rayos ultravioleta durante 15 minutos.

PUERTOS DE E/S.

Al seleccionar está opción se despliega una ventana, que nos muestra el estado de los puertos de E/S. El microcontrolador maneja dos puertos los cuales tienen sus respectivos bytes de control, con los que podemos configurar a los puertos como entradas o como salidas. Además nos muestra los bytes correspondientes de los siguientes registros:

- Control y estado del temporizador.
- Contador del Temporizador.
- Programación de la EPROM.

Para poder cambiar el estado de cualquiera de estos registros, se debe realizar un pequeño programa en la zona de la memoria RAM del microcontrolador y ejecutarlo.

ZONA DE VECTORES.

Esta nos muestra la lista de vectores localizada en la parte alta de la memoria. Los vectores que son manejados por el microcontrolador y son desplegados en pantalla son:

- Registro COP.
- Vector de Interrupción del Temporizador.
- Vector de Interrupción Externo.
- Interrupción por Software.
- Vector de Reset.

Esta zona pertenece a la zona de la Memoria EPROM, por lo que el usuario podrá programarlos con la utilería del sistema destinado a esta función. En el momento exponer el microcontrolador a la luz ultravioleta, se borra toda la información almacenada en el área que corresponde a esta memoria, como consecuencia, también se elimina la información que se encuentre dentro de estos vectores.

UTILERÍAS

En el menú de utilerías se tienen las funciones para administrar el espacio en disco. Esta opción cuenta con las opciones siguientes:

El Directorio de Archivos nos permite examinar la información que se encuentra en el subdirectorio activo. Al seleccionar esta opción aparece una ventana en la cual se debe escribir los nombres o extensiones que se desean buscar, posteriormente se debe oprimir <Enter> para que el programa busque toda la información que se requiere. Finalmente se desplegará una ventana con todos los programas encontrados. En caso de no existir ningún archivo con las características deseados se mostrará un mensaje de que no existen programas.

Dentro de Borrar Archivos el usuario tiene la posibilidad de eliminar archivos del área en donde se encuentre trabajando. Al seleccionarla se abrirá una ventana con todos los archivos posibles a ser borrados, de éstos se debe elegir el que se va a suprimir. Para elegir el programa solo es necesario colocarse sobre éste y oprimir <Enter>, después de la eliminación se seguirá desplegando la misma ventana, pero con los archivos restantes.

En el caso de seleccionar todos los archivos contenidos en el área de trabajo, el sistema realizará una pregunta para confirmar la supresión de los archivos.

Si no desea borrar los archivos o cuando no quiera eliminar más solo debe oprimir <ESC> para retornar al menú de manejo de información

SALIR AL DOS

Nos permite terminar el uso de este programa y retornar al sistema operativo.

BIBLIOGRAFÍA

- Understanding Data Comunications
  from Fundamentals to Networking.
  Gilbert Held.
  Jonh Wiley & Sais.
  1991.

- Data Comunication a Beginner's Guide
  to Concepts and Technology.
  Scott A. Helmers.
  Prentice Hall.
  1990.

- Assemblers, Compilers and Program Translation.
  Peter Calingaert.
  University of North Carolina at Chapel Hill.
  Computer Science Press.
  1989.

- System Software an Introduction to
  Systems Programming.
  Leland L. Beck.
  Addison-Wesley.
  1989.

- Computer Operating Systems
  David Barrón
  Chapman and Hall
  1984.

- Manual del Microcontrolador MC68H05
  Motorola