UNIVERSIDAD DE EL SALVADOR

FACULTAD DE CIENCIAS NATURALES Y MATEMATICA

ESCUELA DE MATEMATICA

"GUIA DE ESTUDIO BASICA PARA EL APRENDIZAJE DEL LENGUAJE ENSAMBLADOR BASADO EN EL MICROPROCESADOR 8088"

TRABAJO DE GRADUACION PRESENTADO POR:

BR. LEDA PATRICIA PALACIOS

BR. JORGE ALFONSO HERNANDEZ

PARA OPTAR AL TITULO DE:

LICENCIADO EN MATEMATICA

JULIO DE 1993

SAN SALVADOR, EL SALVADOR, CENTROAMERICA

UNIVERSIDAD DE EL SALVADOR

FACULTAD DE CIENCIAS NATURALES Y MATEMATICA

ESCUELA DE MATEMATICA

"GUIA DE ESTUDIO BASICA PARA EL APRENDIZAJE DEL LENGUAJE ENSAMBLADOR BASADO EN EL MICROPROCESADOR 8088"

TRABAJO DE GRADUACION PRESENTADO POR:

BR. LEDA PATRICIA PALACIOS
BR. JORGE ALFONSO HERNANDEZ

PARA OPTAR AL TITULO DE:

LICENCIADO EN MATEMATICA

JULIO DE 1993

SAN SALVADOR, EL SALVADOR, CENTROAMERICA

UNIVERSIDAD DE EL SALVADOR

AUTORIDADES UNIVERSITARIAS

DR. FABIO CASTILLO
RECTOR

LIC. MIRNA ANTONIETA PERLA DE ANAYA
SECRETARIA GENERAL

FACULTAD DE CIENCIAS NATURALES Y MATEMATICA

LIC. MARINA ESTELA CONTRERAS DE TOBAR
DECANO

LIC. RODOLFO FERNANDO MENJIVAR
SECRETARIO

ESCUELA DE MATEMATICA

ING. JOSE FRANCISCO MARROQUIN
DIRECTOR

LIC. JUAN AGUSTIN CUADRA
SECRETARIO

UNIVERSIDAD DE EL SALVADOR

ING. SALVADOR PALACIOS FUENTES

COORDINADOR

LIC. MANUEL ALBERTO YANEZ DOÑO

ASESOR

DEDICATORIA

A DIOS TODOPODEROSO, por haberme guiado en el transcurro de mi vida y permitirme culminar mi carrera.

A MI MADRE: SALVADORA PALACIOS, quien con su esfuerzo ha sabido alentarme en toda mi vida y en especial para la obtención de este nuevo triunfo.

A MI ESPOSO, OMAR ALBERTO BATRES, con amor, Por haberme apoyado en todos los momentos difíciles y se parte fundamental de mi vida.

A MIS HERMANAS, MATTY Y ALEJANDRINA, con agradecimiento y amor fraternal por todo su apoyo.

A MI BEBE, que sirvió de inspiración para la culminación de esta nueva etapa de mi vida.

A MIS DEMAS FAMILIARES Y AMIGOS: con agradecimiento, por haberme apoyado en todo momento para la realización de este trabajo.

LEDA PATRICIA DE BATRES

DEDICATORIA

A DIOS, por haberme permitido culminar mi carrera.

A MIS PADRES: ADELA HERNANDEZ Y HUGO VALENCIA, quienes con su esfuerzo me alentaron para la obtención de este nuevo triunfo.

A MIS DEMAS FAMILIARES, con agradecimiento.

JORGE ALFONSO HERNANDEZ

AGRADECIMIENTOS

A los señores ING. JOSE SALVADOR PALACIOS y LIC. MANUEL YANEZ DOÑO, coordinador y asesor respectivamente de nuestro trabajo, con agradecimiento especial por su valiosa colaboración y paciencia.

A todos nuestros maestros que de una u otra forma ayudaron a la realización de este material.

A todas las personas que nos colaboraron desinteresadamente y que estuvieron pendientes del desarrollo de este trabajo.

Sinceramente,

LEDA PATRICIA DE BATRES

JORGE ALFONSO HERNANDEZ

INDICE

INTRODUCCION

El área computacional de nuestro país carece de un material de apoyo sobre programación en lenguaje de bajo nivel, en particular sobre lenguaje ensamblador, es por esto que abordamos este tema para nuestro trabajo de graduación; para que este material pueda ser tomado como referencia para las personas que trabajan en diseño de software.

El lenguaje ensamblador es un lenguaje de programación de bajo nivel; en el que a cada instrucción simbólica le corresponde una instrucción de lenguaje de máquina.

En este trabajo presentamos los elementos fundamentales que pueden ser utilizados para elaborar programas en lenguaje ensamblador.

Esperamos que este documento sirva de base para que se sigan desarrollando trabajos posteriores de esta naturaleza.

CAPITULO 1

CONCEPTOS BASICOS

1.1 ¿QUE ES EL LENGUAJE ENSAMBLADOR?

Se define como microprocesador a la unidad central de proceso de datos, constituida por un solo circuito integrado, C.I., de alta escala en la integración de sus componentes. El microprocesador opera bajo control de un programa y efectúa operaciones sobre los datos:

- operaciones de tipo lógico;
- operaciones de tipo aritmético;
- control de entrada/salida de datos;
- control de funciones y de unidades externas al mismo, pero internas a la computadora.

Al microprocesador se le conoce en el ámbito de la computación como la C.P.U. (Unidad Central de Proceso de datos).

El desarrollo de los microprocesadores se ha debido, en primer lugar, al avance tecnológico en el desarrollo de C.I., cada vez con mayor grado de integración, y por consiguiente a la disminución de costos, y; en segundo lugar, al diseño de circuitos cuya función puede ser cambiada por programación.

Podemos agrupar a los microprocesadores atendiendo al tamaño de palabra, o en función del número de bits del bus de datos. Con ello tenemos microprocesadores de 4, 8, 16, y 32 bits. Estudiaremos en este texto el microprocesador 8088 aunque a veces hagamos referencia a otros microprocesadores.

El 8088 es un procesador de 16 bits, que puede reemplazar al microprocesador 8080 o al 8085 con bus de datos de 8 bits (ambos de 8 bits) en un sistema ya existente.Además, los nuevos sistemas, pequeños y baratos aunque muy potentes ,pueden diseñarse basándose en el chip 8088.

1.2 ESTRUCTURA INTERNA DEL MICROPROCESADOR 8088

El microprocesador 8088 esta dividido en dos subprocesadores separados,la UNIDAD DE INTERFAZ DE BUS (BIU: Bus Interface Unit) y la UNIDAD DE EJECUCION(EU: Execution Unit). La BIU se encarga de controlar las transferencias entre la C.P.U. y el mundo exterior , mientras que la EU es la que realiza las operaciones aritméticas y lógicas.

El 8088 contiene 14 registros de 16 bits. Algunos pertenecen a la EU y otros a la BIU. Los de la EU se suelen usar para direccionamiento.

El subprocesador EU consta de:

- A.L.U.(Unidad Aritmética Lógica): Esta unidad es el elemento calculador del sistema, capaz de realizar operaciones lógicas (AND, OR, XOR) y operaciones aritméticas (suma,resta,etc.).

- Cuatro registros generales de 16 bits(AX,BX,CX,DX), que pueden subdividirse en ocho registros de 8 bits (AH,AL,BH,BL,CH,CL,DH y DL).

- Cuatro registros punteros y de índice (SP,BP,SI,DI).

- Un registro de indicadores de 16 bits, que contiene varios bits de estado para el procesador.

El subprocesador BIU consta de :

- Unidad de control: Es el elemento que regula el flujo de la información (instrucciones y datos) en el sistema.

- Cuatro registradores de segmento (CS, DS, SS y ES). Sus códigos representan a los registradores de segmento de código, datos, pila y extra, respectivamente.

- Un puntero de instrucción (IP).

- Cola de instrucciones: Una cola es una línea de espera como las que se forman en las cajas de los supermercados. Algunos micros de 16 bit como el 8088 de intel, utiliza tales "líneas de espera" para sus instrucciones. Dicho de otra manera, las instrucciones que han de ejecutarse llegan al procesador antes de lo necesario y "esperan" en una cola de instrucciones; como muestra la figura 1.1. Este sistema posee la ventaja de que cada instrucción puede extraerse de la memoria mientras otras se están ejecutando, reduciéndose en consecuencia el tiempo de proceso.

La cola de instrucciones es normalmente corta de 4 a 6 bytes; en concreto la del 8088 es de 4 bytes.

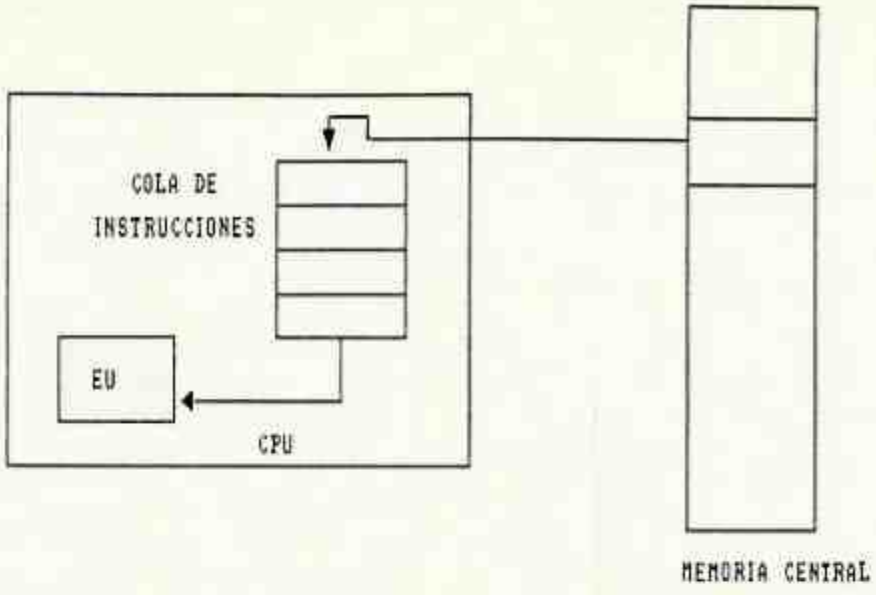


FIGURA 1.1 CADA INSTRUCCION PUEDE EXTRAERSE DE LA MEMORIA MIENTRAS OTRAS SE ESTAN EJECUTANDO.

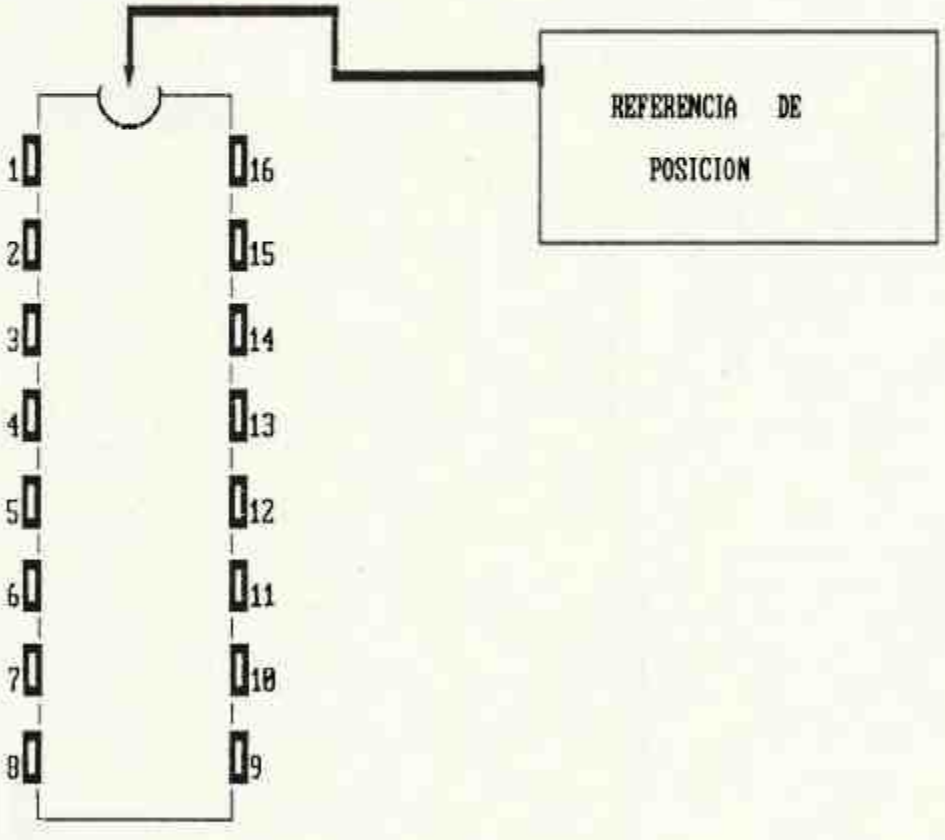


FIGURA 1.2 CIRCUITO INTEGRADO

1.3 CIRCUITO INTEGRADO O CHIP

Dispositivo electrónico compuesto por un conjunto de componentes conectados permanentemente entre sí e incluidos en una placa de silicio de menos de 1mm^2, formando un conjunto en miniatura capaz de desarrollar las mismas funciones de un circuito formado por elementos discretos.

En un circuito integrado los componentes activos: diodos, transistores, etc.; y los componentes pasivos: resistencias, condensadores,etc., están integrados dentro de un mismo bloque llamado substrato.

Normalmente se mide la densidad de un circuito integrado por el número de puertas que contiene. Un dispositivo que contenga de 1000 puertas en adelante tiene una escala de integración muy alta (VLSI: Very Large Escale Integration). El chip 8088 contiene varios miles de puertas considerándose por lo tanto dispositivo VLSI.

Los circuitos integrados son las células elementales de las computadoras. Se conectan entre sí para formar el circuito impreso de la máquina a través de patillas o pines, numerados en el orden que indica la figura 1.2, y poseen unas muescas en su parte superior que permiten conocer la posición que deben ocupar en el montaje.

1.4 BUSES

La CPU se comunica con todas las posiciones de memoria y todos los periféricos de la computadora a través de grupos de conductores llamados BUSES.(ver figura 1.3).

El bus se puede definir como el canal o camino a través del cual los componentes de una computadora, o las unidades en que ésta se divide, se comunican entre sí.

Resulta muy conveniente adoptar el modelo lógico de un computador como el de un bus, al cual se le han conectado diversos dispositivos. Como se muestra en la figura 1.4.

Los buses pueden ser de líneas homogéneas entre sí, como el bus de datos y el bus de direcciones, o de líneas completamente heterogéneas como el bus de control.

Los buses más comúnmente presentes en las computadoras son:

- BUS DE DATOS;
- BUS DE DIRECCIONES;
- BUS DE CONTROL.

BUS DE DATOS:(Data Bus); En él viajan todos los datos de una parte a otra de la computadora. En este bus, los datos pueden ser de entrada o de salida con respecto a la CPU, memoria y controladores de entrada/salida. En el 8088 este bus consta de 8 líneas.

BUS DE DIRECCIONES:(Address Bus); Tiene la función de seleccionar o direccionar las distintas partes de la computadora. La CPU puede seleccionar mediante este bus una dirección de la memoria para posteriormente leer los datos que contiene. En el 8088 este bus consta de 20 líneas. Este bus con respecto a la CPU, siempre es un bus de salida.

BUS DE CONTROL:(Control Bus); En este bus viajan las señales de control de todo el sistema. Con respecto a la CPU, este bus puede ser de entrada o de salida, ya que la CPU, además de controlar las unidades periféricas de la misma,puede recibir información del estado de una determinada unidad.

Todas las informaciones, bien sean datos, direcciones o control, viajan por los hilos de los buses en forma de ausencia o presencia de tensión sobre cada uno de ellos, por lo que la ausencia de tensión será 0 ó la presencia de tensión será 1. También es común representar 0 y 1 por dos niveles diferentes de voltaje. Toda información que viaja sobre un bus lo hace codificada en forma binaria.

En la figura 1.5, los pines etiquetados AD0 - AD7 forman el bus de datos de 8 bits. Las mismas 8 líneas de menor orden forman parte del bus de direcciones de 20 líneas.

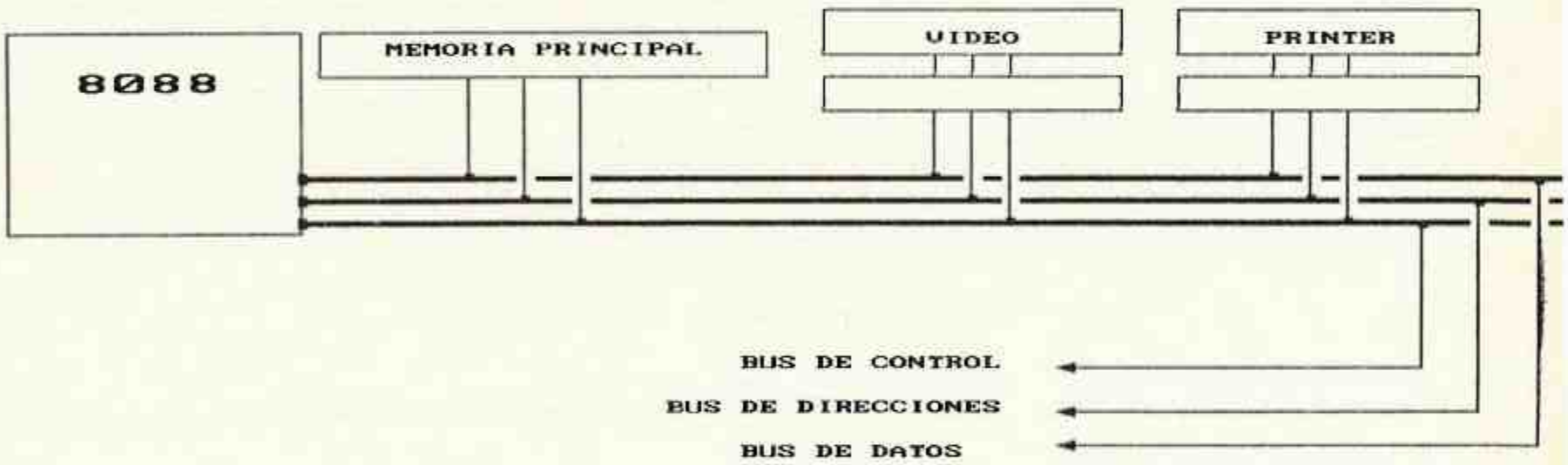


FIGURA 1.3 BUSES

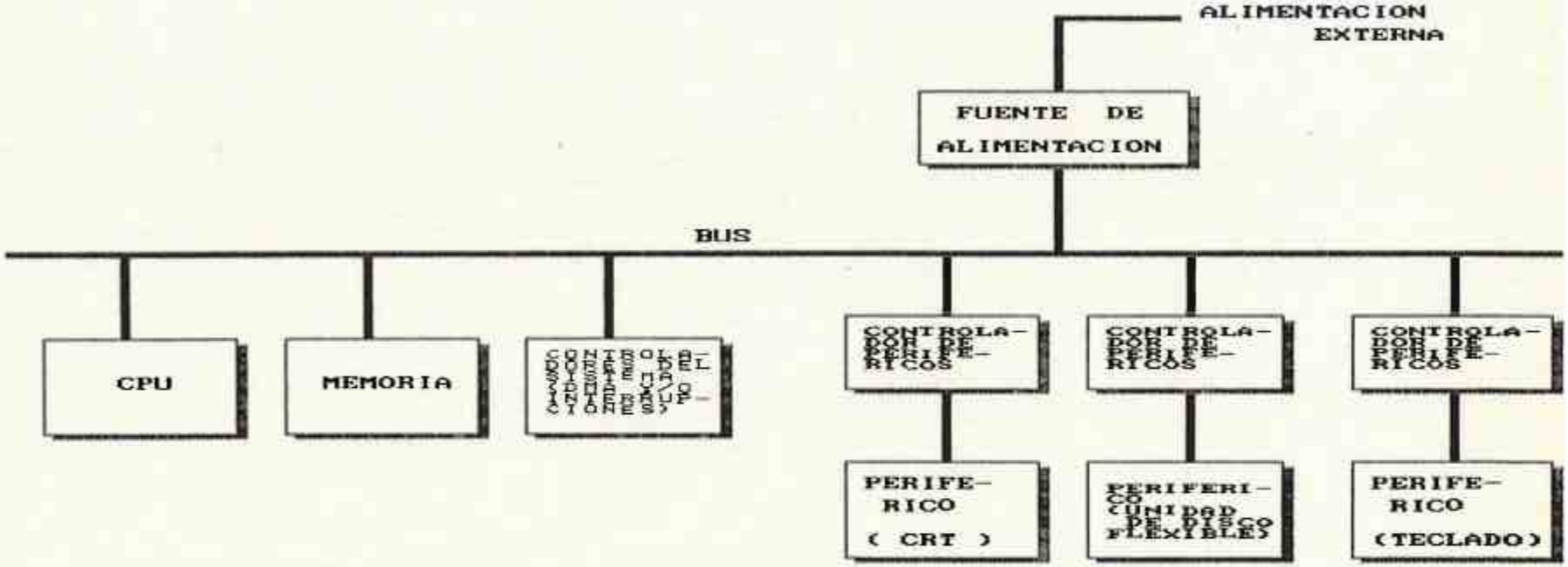


FIGURA 1.4 DIAGRAMA DE BLOQUES DE UN SISTEMA BASADO EN BUSES.

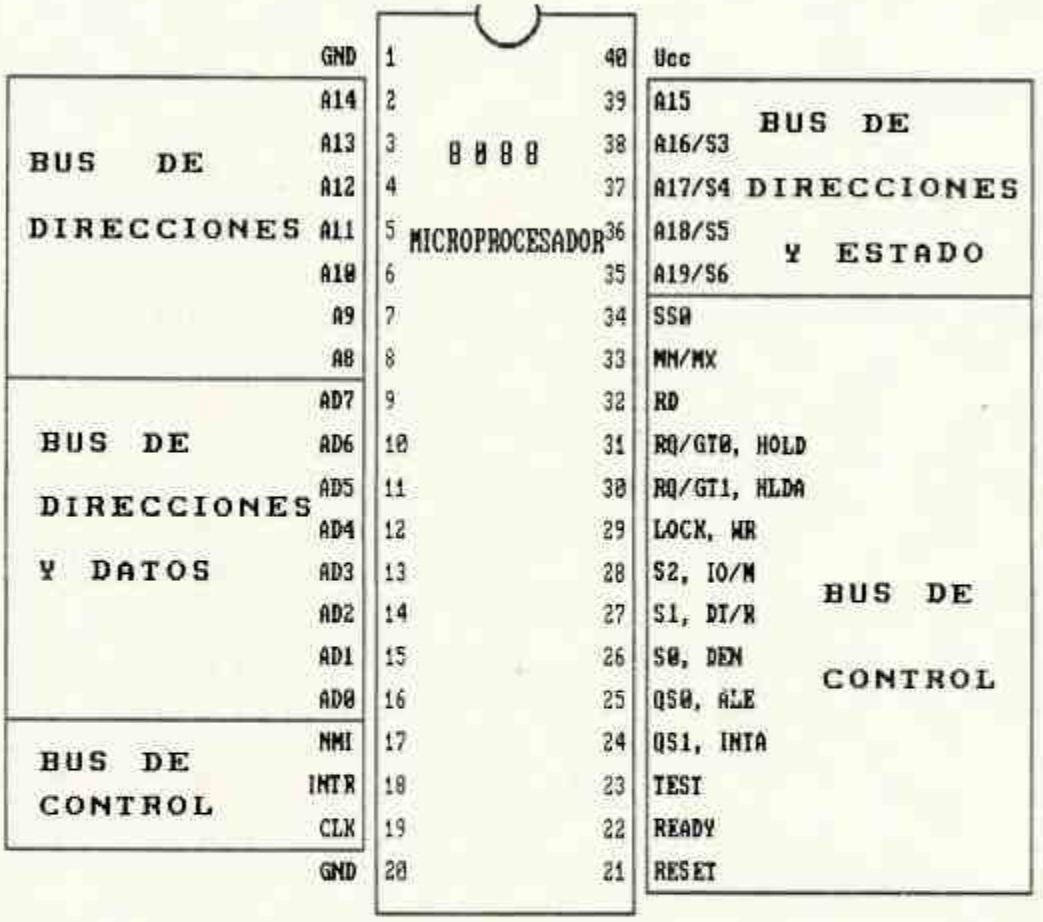


FIGURA 1.5 PINES DE SALIDA DEL MICROPROCESADOR 8088.

1.5 CODIFICACION DE LA INFORMACION

Sabemos que la computadora codifica toda la información en forma numérica, pero el sistema decimal que utilizamos diariamente es de difícil empleo en las computadoras, ya que para representar los números y trabajar con ellos son necesarios 10 símbolos (0,1,2,3,4,5,6,7,8,9), por lo que se ha optado por un sistema de numeración que simplifica mucho el diseño de los circuitos por exigir sólo dos estados o posiciones de funcionamiento.

Como hay sólo dos dígitos, diremos que el sistema de numeración es binario. A las cifras o símbolos binarios les llamaremos, por convención, bits.

La palabra BIT es el acrónimo de las palabras inglesas BInary-digiT.

El bit es la unidad fundamental de almacenamiento de información, el cual puede tomar el valor de 0 ó 1. La combinación de estos dos símbolos, un determinado número de veces permite la codificación de toda la información posible. Así:

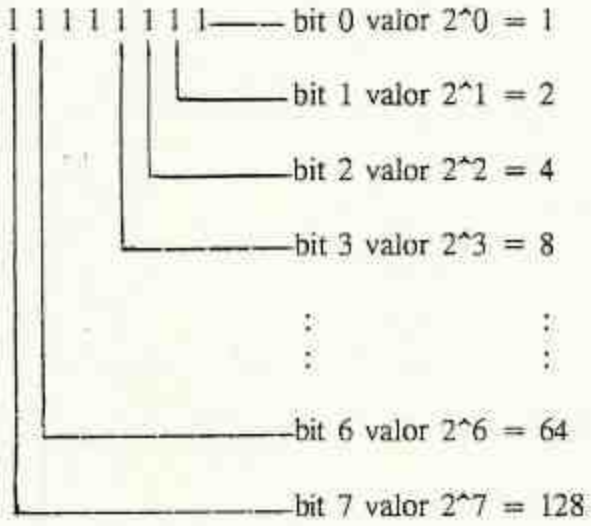
- Con un solo bit, se representan dos informaciones (2^1);
- Con dos bits (2^2), obtenemos cuatro combinaciones de información;
- Con tres bits (2^3), ocho combinaciones de información;
- Con n bits (2^n), 2 a la n combinaciones de información.

Siguiendo con estos razonamientos, podemos llegar a una numeración base 2. Así tenemos que el número base 2, ó binario 10110 equivale a:

$$1*2^4 + 0*2^3 + 1*2^2 + 1*2^1 + 0*2^0 =$$

$$16 + 0 + 4 + 2 + 0 = 22 \text{ (base 10)}$$

Cada bit, según la posición que ocupa dentro del conjunto de un número binario, tiene un valor determinado. Ejemplo:



Se puede observar que, si los números crecen el número de bits que necesitamos para expresarlos, también crece. Para evitar escribir una larga corriente de 1's y 0's, usaremos el sistema de números hexadecimales. Hexadecimal es un sistema numérico base 16, que consta de 16 dígitos; por lo que se utilizan los números decimales de 0 a 9 y las primeras seis letras del alfabeto; de esta manera, el dígito hexa A, tiene el valor de 10 y así sucesivamente hasta el dígito hexa F, que tiene el valor de 15. La posición de cada dígito en un número hexa representa una potencia de 16. Siempre denotaremos

los números hexa poniéndoles la letra H como subfijo. Si algún número hexa comienza con algunos de los dígitos de la A a la F, se le antepondrá un cero, para evitar confundir el número con una etiqueta, (es decir, 0A2H, 0FFH, etc.).

A continuación se presenta el proceso para obtener el correspondiente valor en base 10, del numero 3A7H:

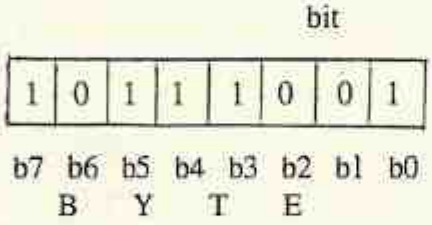
3 * 16^2 + A * 16^1 + 7 * 16^0 =

3 * 256 + 10 * 16 + 7 * 1 =

768 + 160 + 7 = 935 (base 10)

1.5.1 EL BYTE

A la combinación o agrupación de ocho bits se le da el nombre de BYTE.



Dado que el byte es la combinación de ocho bits. El byte puede asumir 256 valores distintos, ya que 256 son las combinaciones posibles con 8 bits.

La mitad de un byte recibe el nombre de NIBBLE, mientras que un grupo de dos bytes se conoce en el 8088 como PALABRA. En la figura 1.6 se presenta una ilustración de estas unidades.

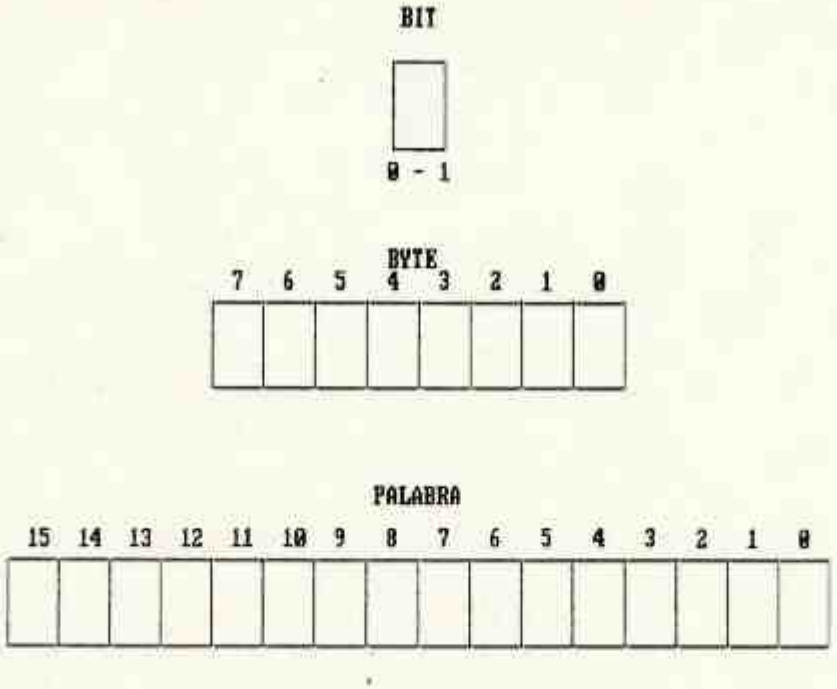


FIGURA 1.6 UNIDADES DE ALMACENAMIENTO DE INFORMACION

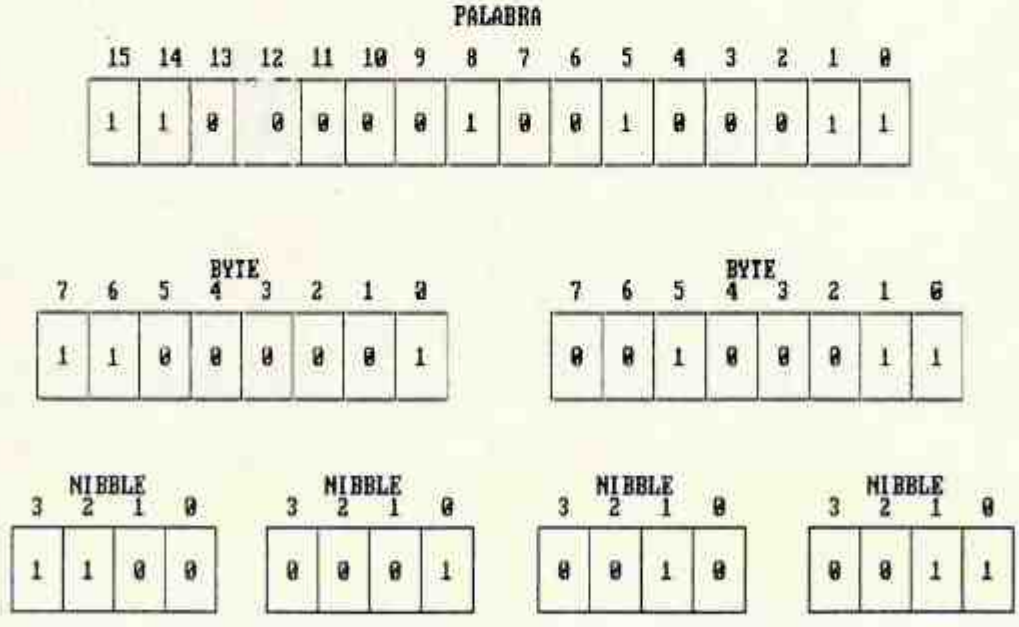


FIGURA 1.7 COMPONENTES MAS Y MENOS SIGNIFICATIVOS

Ahora podemos definir nuestro byte de 8 bits en una mitad superior de 4-bit y en una mitad inferior de 4-bit y cada mitad representada con un dígito hexa. Algunas veces los bytes serán escritos, como un par de dígitos hexa. El dígito hexa, viene a ser la mitad de un byte, y es llamado nibble. Así, cada byte consiste de un nibble más significativo y de un nibble menos significativo. Similarmente, cada palabra consiste de un byte más significativo y de un byte menos significativo. Ver figura 1.7.

Es muy frecuente que el byte se represente en forma hexadecimal.

Así: 1 0 1 1 1 0 0 1 = 0B9H

Para diferenciar los bits contenidos en un byte se enumeran de 0 a 7 y de derecha a izquierda. De este modo, se dirá que b0 es el bit menos significativo o de menor peso dentro de un byte, y b7 es el bit más significativo o de peso más alto.

1.5.2 LOS CODIGOS Y LA CODIFICACION

El conjunto de reglas con las cuales utilizamos los bit para representar las informaciones (Por ejemplo: números y letras) se llamarán códigos. Por consiguiente, la codificación es la operación que transforma una información en paquetes o grupos de bits.

Existe una multitud de códigos que han ido surgiendo a lo largo de la historia de la computación para resolver distintos problemas.

Estos son los más importantes:

- CODIGO BCD (Binary Coded Decimal). Es una codificación a 6 bits y permite la representación de 64 caracteres (26 letras del alfabeto, 10 cifras decimales y 28 caracteres diversos). Este tipo de código no permite distinguir entre letras mayúsculas y minúsculas ni tampoco introducir nuevos caracteres necesarios para la comunicación entre computadoras.

- CODIGO EBCDIC (Extended BCD Interchange Code). Es una codificación en la que se usan los 8 bits, con ellos se consigue representar 256 caracteres distintos. En este código cada letra, número o caracter especial requiere una secuencia de 8 bits.

- CODIGO ASCII (American Standard Code for Information Interchange). Codificación de 8 bits. Funciona igual al anterior.

- EBCDIC, pero su significado es distinto. Se utilizará el código ASCII, por ser un código estándar en programación. Este es presentado en el apéndice A.

- CODIGO BINARIO, Expresa, números binarios enteros, donde a cada posición de bit se le asigna una potencia de 2.

- CODIGO INTERNO. Se llama así al que cada computadora adopta para representar los caracteres en su propia memoria.

1.6 ALMACENANDO INFORMACIÓN EN LA MEMORIA.

La memoria principal esta compuesta de un gran número de celdas, cada uno de los cuales puede tener un byte de información. Una única dirección es asignada a cada celda comenzando con el número 0 y avanzando de 1 en 1 en cada celda sucesiva. El acceso a la información contenida en alguna de estas celdas se hace especificando su dirección. Esto será explicado con mayor detenimiento en el próximo capítulo.

1.7 ORGANIZACION FISICA Y LOGICA DE LA MEMORIA

Una computadora, por pequeña que sea, puede manejar gran cantidad de datos, pero a condición de que estos estén organizados de una manera determinada para poder acceder a ellos en una forma más fácil y rápida.

La información se encuentra almacenada en los dispositivos o unidades de almacenamiento externo de la memoria.

Esta información se puede organizar de dos formas: física y lógicamente. La organización física depende de los soportes sobre los que estará grabada la información,

y por lo tanto, puede variar según las características de cada uno de estos soportes. En cambio, la organización lógica varía a voluntad del usuario, aunque las herramientas que éste tiene a su disposición para definir esta organización son fijas.

1.8 FUNCION DEL PROCESADOR CENTRAL

El trabajo del procesador central es ejecutar una secuencia de instrucciones que son colocadas dentro de la memoria principal del computador. Cada uno de los bytes de datos es interpretado por el procesador central como funciones que puede realizar. Los bytes siguientes a los baytes de datos pueden contener datos específicos para ser usados en la ejecución de funciones. Estos bytes adicionales son conocidos como operandos.

Algunas veces el procesador tiene que realizar una función apropiada, luego continúa con el próximo byte en la memoria y ejecuta la función especificada. Así podemos ver, que programar la computadora consiste en preparar una secuencia de bytes que representen funciones deseadas, colocando estos byte de datos dentro de la memoria, e instruyendo al procesador para comenzar ejecutando la instrucción de la dirección apropiada.

El repertorio de funciones que el procesador puede ejecutar es conocido como su conjunto de instrucciones.

La unidad central de proceso constituye el cerebro o centro de control de la computadora.

La CPU está formada normalmente por unos pocos chips de una alta densidad de componentes (LSI y/o VLSI) constituyendo lo que se llama el microprocesador junto con hasta 40 ó más chips de menor densidad (MSI y SSI) con misiones de soporte.

1.9 PORQUE NECESITAMOS UN LENGUAJE ENSAMBLADOR?

Hoy en día, el procesador central de un microprocesador es usualmente contenido en su totalidad en un chip y es llamado un microprocesador. A pesar de su diminuto tamaño, el microprocesador es capaz de reconocer y ejecutar cientos de instrucciones diferentes.

El lenguaje máquina es el que la computadora puede ejecutar directamente; sin embargo, este lenguaje no puede denominarse como tal a causa de que no está formado por símbolos o signos, como cualquier otro lenguaje, sino que está compuesto por cantidades numéricas expresadas en base 16 ó hexadecimal. De ello se deduce que programar directamente en lenguaje máquina resulta muy complicado. Para evitar esta dificultad existe el lenguaje ensamblador, que es un lenguaje de programación muy cercano al lenguaje máquina.

El lenguaje ensamblador es de bajo nivel. En este lenguaje, cada línea de programa que se escribe con él, es una orden para la computadora; cada una de estas líneas puede ser dividida en cuatro campos separados: LABEL, OPCODE, OPERANDO y COMENTARIO.

El campo label va primero, pero es opcional dependiendo si se necesita. Es usado para asignar un nombre simbólico a la oración.

El campo opcode sigue al campo label. Si un label no es usado entonces uno o más blancos deben preceder al opcode. El opcode es el "Código de Operación". Función que queremos que realice el ensamblador.

El campo operando sigue al opcode. Este campo (operando) es usado para codificar además la función a ser realizada. A veces, más de un operando es requerido para una instrucción. En este caso, los operandos son separados uno de otro por coma. Para muchas de estas instrucciones que requieren dos operandos, esto es posible para identificar uno de ellos como el fuente y el otro como destino de la operación. Una convención estándar del lenguaje ensamblador es siempre colocar el operando destino antes del operando fuente.

El campo comentario puede seguir opcionalmente al campo operando. Este siempre comienza con el caracter punto y coma(;). En todo programa de lenguaje ensamblador, este campo es muy importante usarlo para describir mucho mejor la documentación. Si esto no se hace, podrías encontrar tu propia lectura de un programa que escribiste hace meses y preguntarte, cómo hice este trabajo?.

Como ya se dijo, en los programas escritos con lenguaje ensamblador cada instrucción o línea de programa se corresponde con una acción que debe ejecutar la computadora.

Los programas escritos en lenguaje ensamblador ocuparán menos espacio, una vez ensamblados y traducidos al lenguaje máquina, que los de alto nivel una vez compilados. Incluso normalmente se ejecutarán más rápidamente, siempre que estén bien construidos y optimizados.

Si parte del código de un programa escrito en lenguaje de alto nivel tarda mucho en ejecutarse, entonces la mejor opción para el programador es escribir una rutina en lenguaje ensamblador que haga lo mismo que el código en cuestión.

Una vez que se aprende un lenguaje ensamblador es sencillo comprender otros. La programación en lenguaje ensamblador es la base para evaluar cualquier actividad profesional de programación.

A continuación se enumeran algunas de la ventajas que se obtienen al aprender un lenguaje ensamblador:

1- Habilidad para control de hardware.

2- Habilidad para desarrollar fragmentos de programa que sean de rápida ejecución.

3- Habilidad para acceder, de manera óptima y eficiente el coprocesador.

4- Comprensión de los métodos utilizados para realizar la sintaxis asociada con lenguajes de alto nivel.

5- Conocimiento profundo de los sistemas basados en microcomputadores y de la interfaz hardware/software.

6- Disciplina para programar de manera estructurada.

7- Comprensión de la forma en que se maneja, a bajo nivel, diversas estructuras de datos.

CAPITULO 2

MEMORIA DEL COMPUTADOR

2.1 UNIDADES DE ALMACENAMIENTO

Como se mencionó anteriormente, el bit es la unidad más pequeña de información . El conjunto formado por cuatro bits, es lo que se conoce como nibble, donde dos nibbles forman un byte, es decir, un byte consta de ocho bits. Una palabra está formada por dos bytes. En la figura 1.6 se ilustra la estructura de estas unidades de almacenamiento de información.

2.2 SEGMENTACION DE DIRECCIONES DE MEMORIA

Todas las computadoras personales, tienen un dispositivo en el cual se lee y almacena información, esta se conoce con el nombre de MEMORIA PRINCIPAL. Una ilustración de la memoria principal se muestra en la figura 2.1.

Dicha memoria se encuentra dividida en posiciones que constan de 8 bits cada una; Es decir, esta dividida en bytes. Cada una de estas posiciones tiene asociada a ella una dirección, la cual esta representada en el sistema numérico hexadecimal (base 16).

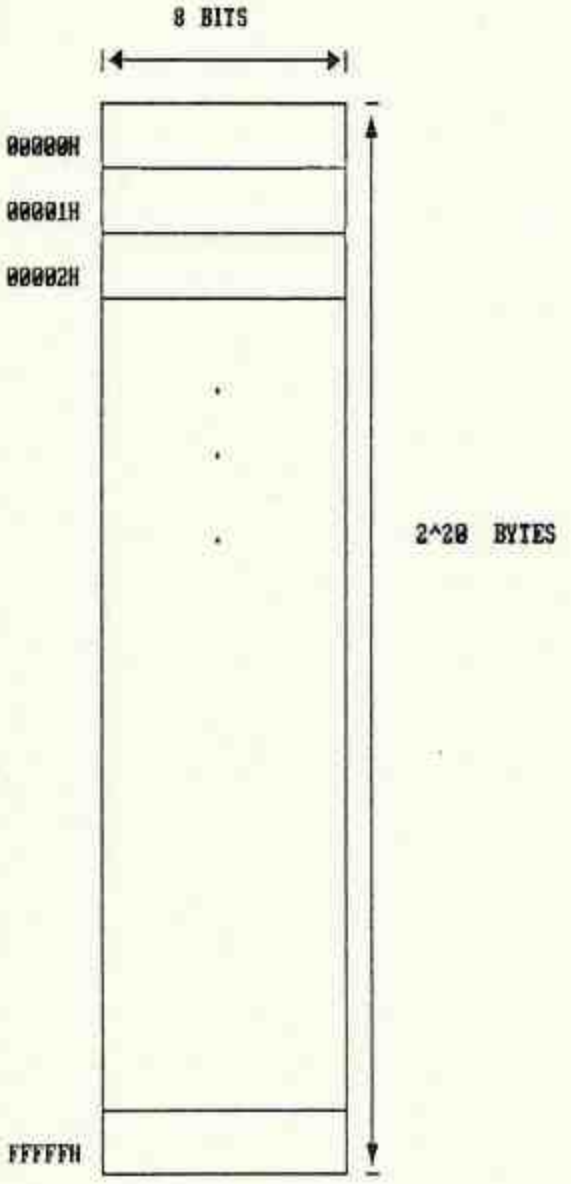


FIG. 2.1 ESPACIO DE MEMORIA DE UN MEGABYTE.

La memoria contiene un número de bytes igual a 2^20, de manera que se puede decir, que la memoria con que cuenta el microprocesador, para leer y almacenar datos es de 1,048,576 bytes. Así el rango de direcciones de memoria es de 0=00000h hasta 1,048,575=fffffh. Ademas, se tiene que 2^10=1K y que 2^20=1mega, por lo que, se puede decir, que la memoria principal tiene capacidad de almacenamiento de un megabyte. Como se puede ver en la figura 2.1, estas direcciones ocupan hasta cinco dígitos hexadecimales (20 bits) para poder representarse, pero el 8088 unicamente procesa palabras de 16 bits, con las cuales solamente puede representar cantidades de hasta cuatro dígitos hexadecimales. A continuación pasaremos a explicar como es posible direccionar cada una de las posiciones de memoria.

Para direccionar la memoria, las microcomputadoras utilizan 20 bits; sin embargo la CPU procesa palabras de 16 bits en sus registradores. Entonces, como se lleva a cabo en tiempo real? para responder a esta pregunta, empezaremos aclarando que dentro de la memoria se definen areas continuas de memoria que pueden tener hasta un máximo de 64K bytes de longitud que son llamadas SEGMENTOS. Un segmento de 64K bytes dentro de un espacio de memoria de un megabyte, es mostrado en la figura 2.2.

El segmento debe comenzar en una posición de memoria cuya dirección sea multiplo de 16, a esta dirección se le conoce con el nombre de PARRAFO.

En la figura 2.3 se muestran las posibles direcciones de inicio de un segmento.

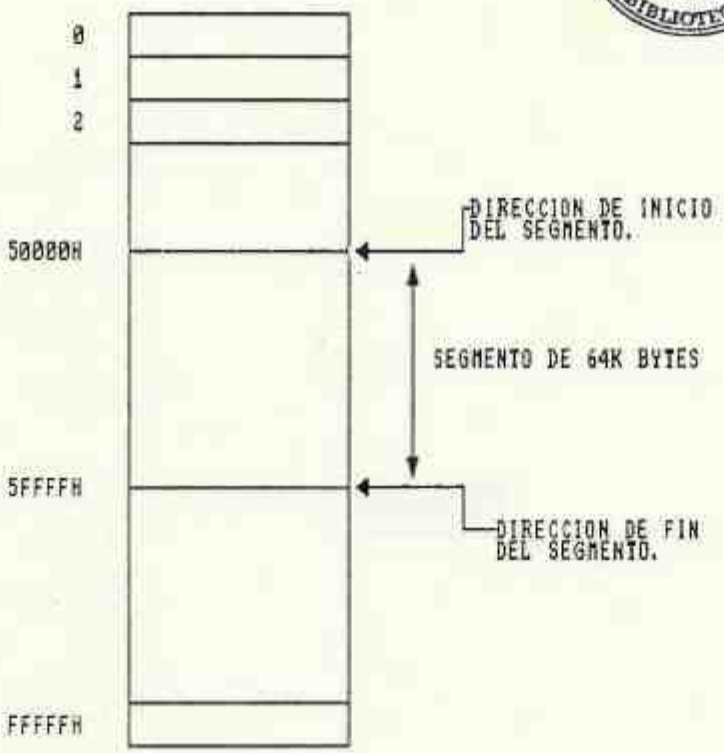


FIG. 2.2 UN SEGMENTO DE 64K DENTRO DE UN ESPACIO DE DIRECCIONES DE UN MEGABYTE.

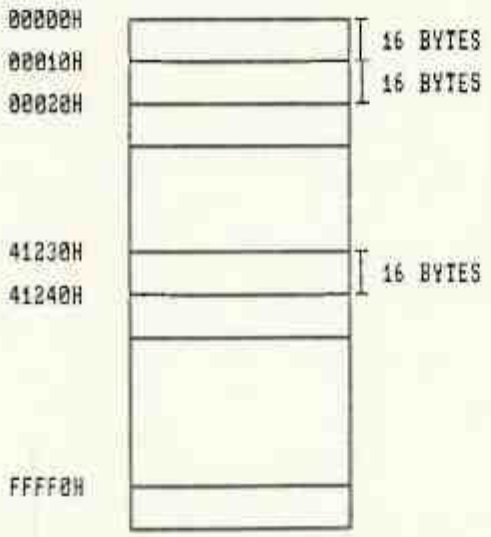


FIGURA 2.3 POSIBLES DIRECCIONES DE INICIO DE UN SEGMENTO.

Los segmentos pueden traslaparse con otros intercalando posiciones de memoria. La dirección de inicio de un segmento define su posición; es decir, cuando la CPU necesite tener acceso a un segmento, sera por medio de este que sabrá donde encontrarlo. Estas direcciones pueden estar contenidas en uno de los cuatro registradores de segmento disponibles en el 8088. Los registradores del 8088 sarán estudiados en el capítulo siguiente.

Los segmentos pueden tener como máximo 64k bytes de longitud, y se pueden definir hasta cuatro, lo que sumaria hasta 256K bytes= 262144, cantidad que no es posible almacenar en un registrador; ya que estos son de 16 bits. Es necesario especificar otro parametro para accesar las posiciones de memoria dentro de un segmento. Este parametro es el desajuste de la posición de memoria, el cual necesita de una palabra para definir todas las posiciones posibles dentro de un segmento de hasta 64K bytes de longitud máxima. La figura 2.4 muestra un diagrama de un desajuste dentro de un segmento. De esta manera podemos ver que las direcciones efectivas (de 20 bits) estan divididas en dos componentes: direcciones de segmentos y direcciones desajustadas (cada una de 16 bits).

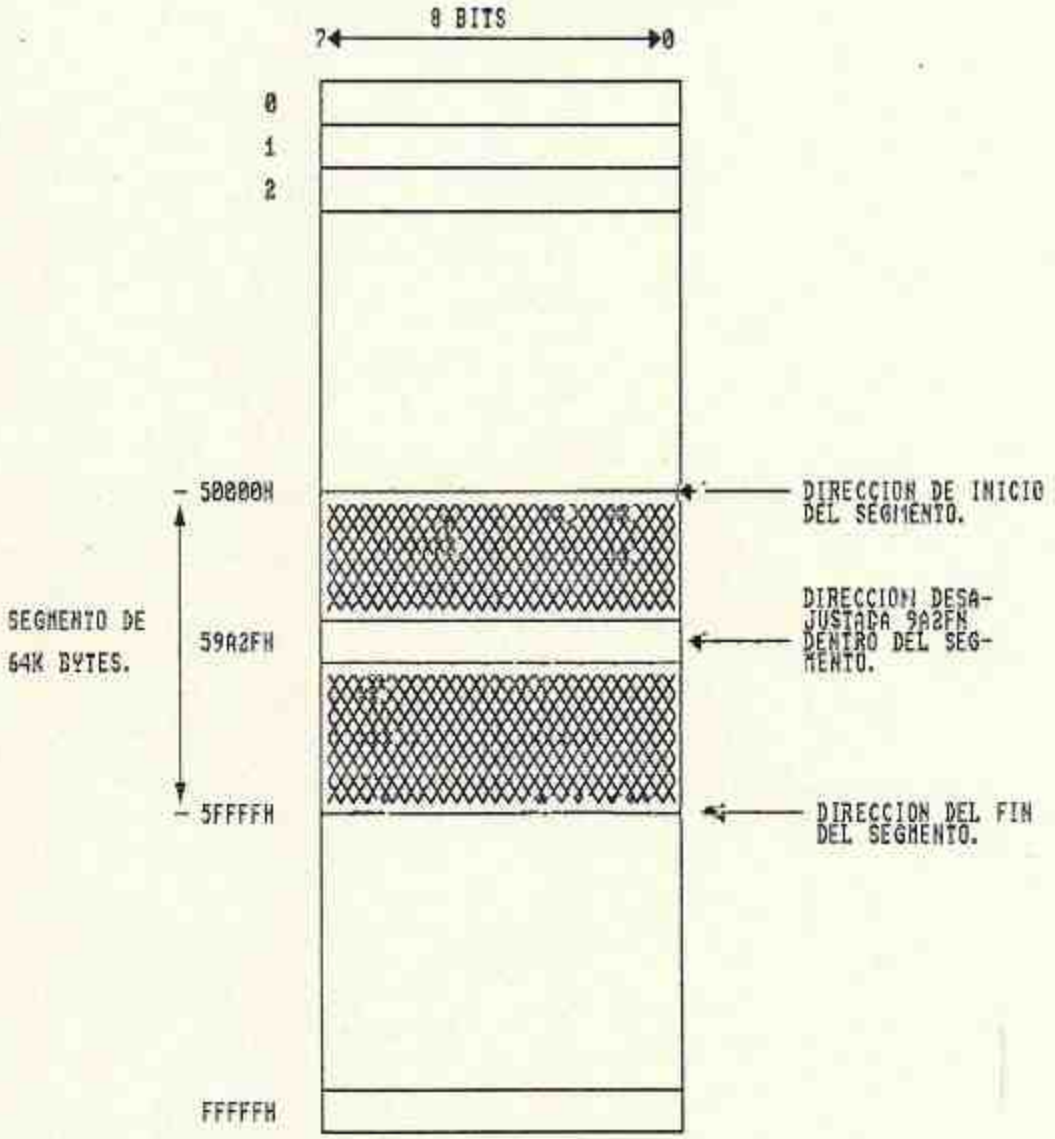


FIG. 2.4 DIRECCION DESAJUSTADA DE UN SEGMENTO DENTRO DE UN ESPACIO DE MEMORIA DE UN MEGABYTE.

La dirección efectiva en el espacio de memoria de un megabyte, se obtiene combinando las direcciones de segmento con la del desajuste. Para esto, primero se hace un corrimiento de la dirección de segmento cuatro bits a la izquierda (introduciendo ceros por la derecha en los bits menos significativos) y después se suma al resultado la dirección desajustada, con lo que se obtiene una dirección de 20 bits. Para ejemplificar el procedimiento descrito, véase la figura 2.5. Con esto se responde a la pregunta planteada anteriormente.

Después de dar una respuesta teórica, en donde solamente se mencionan direcciones de 16 y 20 bits, pasaremos a un ejemplo con numeros de dichas direcciones.

Supongamos que la dirección de inicio de un segmento es 10AFH y que la dirección desajustada es F0FFH. Calcularemos la dirección efectiva asociada a la dirección desajustada F0FFH como se muestra en la figura 2.6. Las posiciones de memoria de éste ejemplo se ilustran en la figura 2.7.

2.3 ORGANIZACION FISICA Y LOGICA DE LA MEMORIA.

La organización de la memoria de una computadora digital depende no sólo de la computadora particular sino también del punto de vista adoptado, ver la figura 2.8.

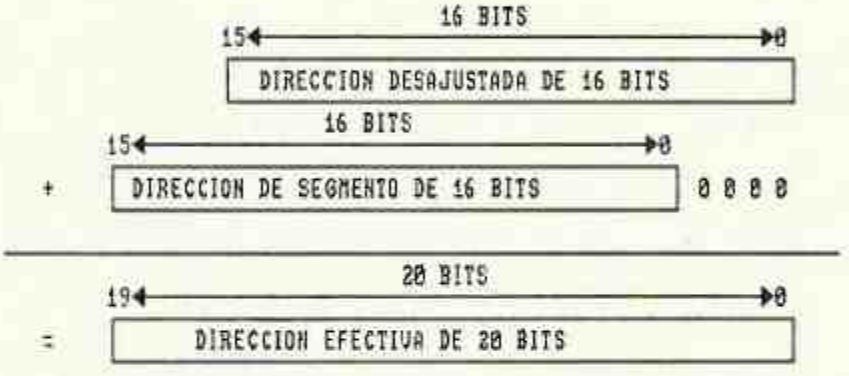


FIG. 2.5 CALCULANDO DIRECCIONES EFECTIVAS.

```
    1   0   A   F   (0)
+       F   0   F   F
-------------------------
=   1   F   B   E   F
```

FIG. 2.6 CALCULO DE LA DIRECCION EFECTIVA ASOCIADA A LA DIRECCION DESAJUSTADA F0FFH.

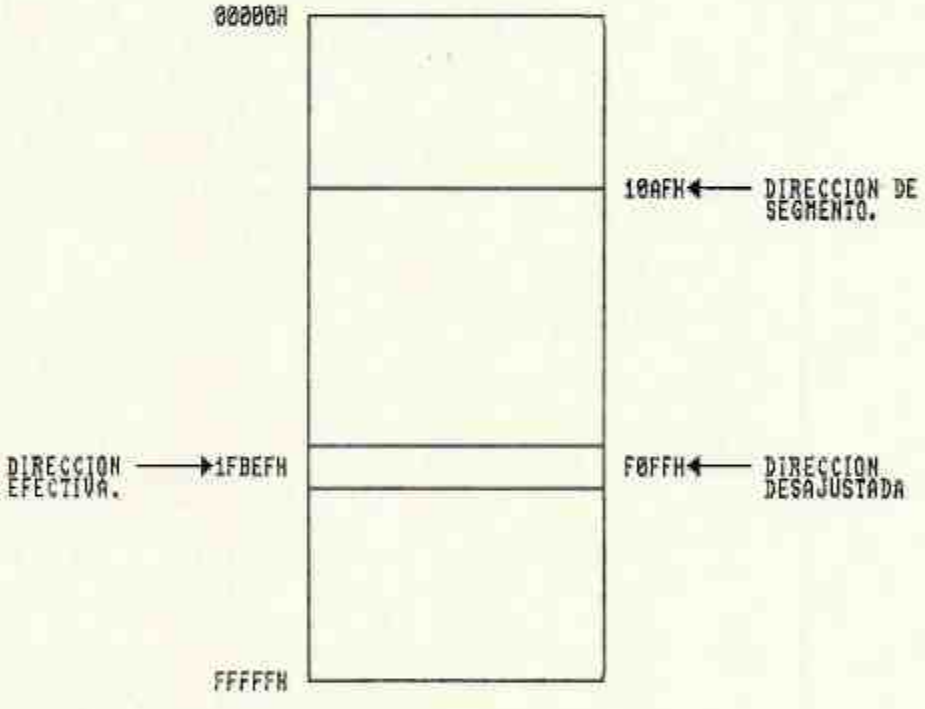


FIG. 2.7 ILUSTRACION DE LAS POSICIONES DE MEMORIA DE SEGMENTO Y DESAJUSTADA DE LA FIG. 2.6.

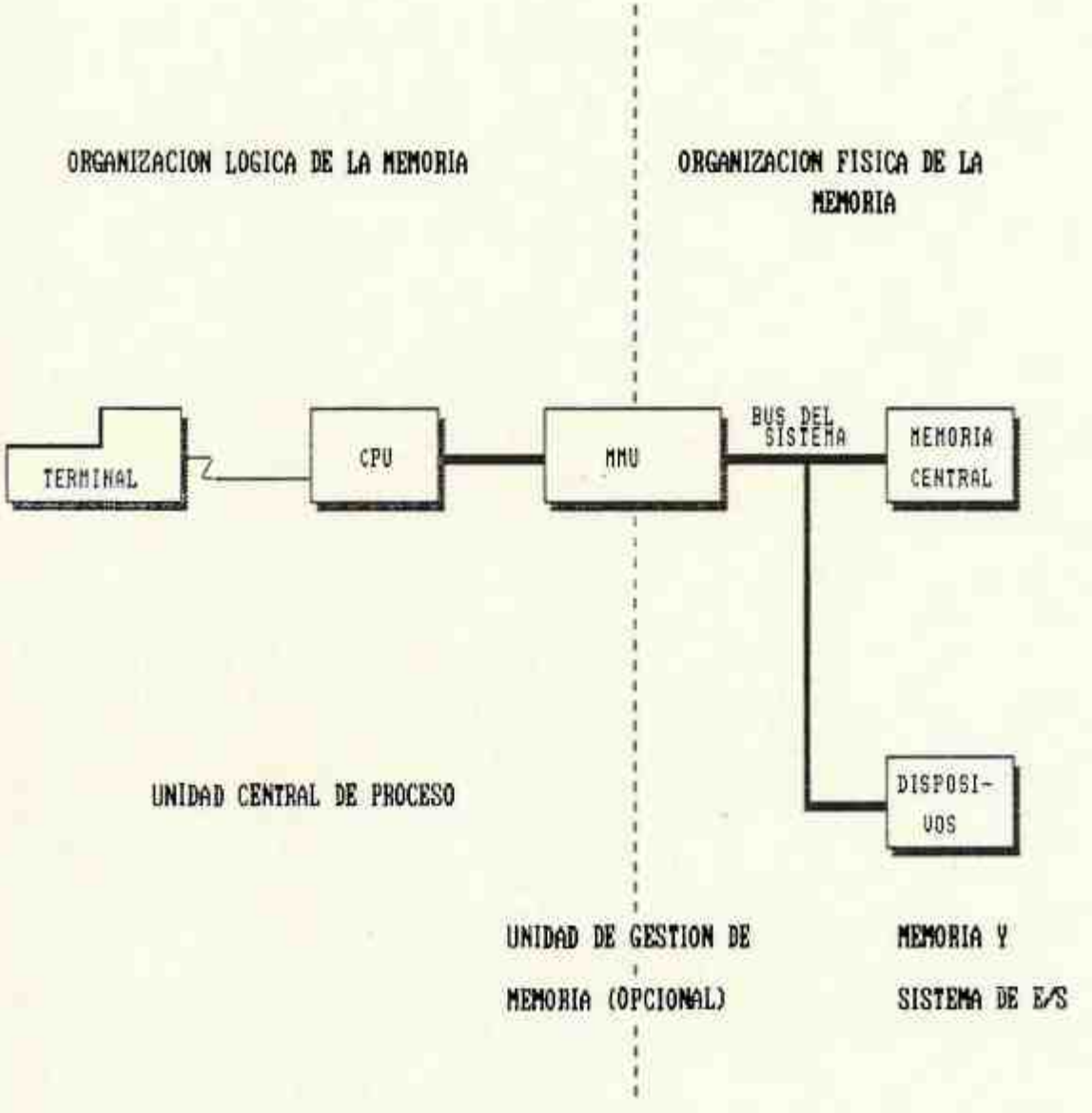


FIGURA 2.8 ORGANIZACION FISICA Y LOGICA DE LA MEMORIA.

La memoria que ocupan las computadoras personales con el 8088 son varios chips que tiene la capacidad para almacenar hasta un megabyte. Dentro de este espacio de memoria se pueden definir cuatro segmentos fundamentales. Los cuales son:

-Segmento de código; -Segmento de datos;

-Segmento de pila; -Segmento extra.

Cada posición de memoria es un byte el cual tiene asociada una dirección particular. Esta dirección está representada en el sistema hexadecimal y son direcciones efectivas que constan de cinco dígitos.

La dirección de inicio de un segmento (párrafo), consta de cinco dígitos. A partir de esta dirección siguen otras referidas a ésta, llamadas direcciones desajustadas dentro del segmento, ver figura 2.4.

2.4 ALMACENAMIENTO DE INFORMACIÓN EN LA MEMORIA PRINCIPAL

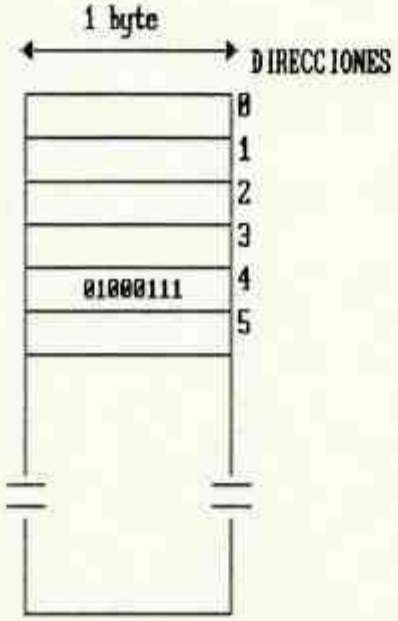
La memoria principal esta compuesta de un gran número de celdas, cada una de los cuales puede tener un byte de información. A cada una de estas celdas es asignada una única dirección comenzando con el número 0 y avanzando de 1 en 1 en cada celda sucesiva. El acceso a la información contenida en alguna de estas celdas se hace especificando su dirección.

En la fig. 2.9, vemos como un byte de datos almacenado en la memoria podrá ser referido por la dirección apropiada.

Para almacenar una palabra en memoria usaremos dos posiciones de memoria consecutivas como se muestra en la figura 2.10.

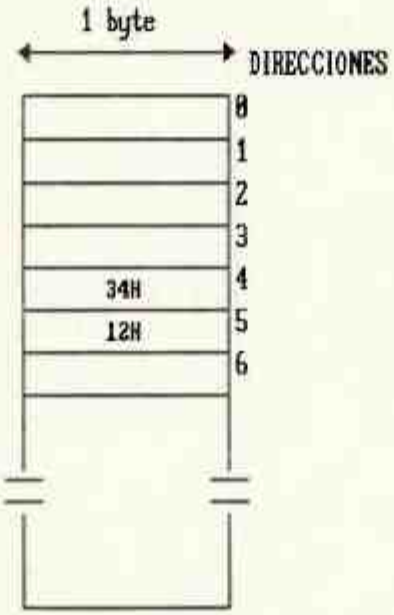
En resumen, los dos byte que componen la palabra son almacenados en orden reverso. El byte menos significativo es almacenado en la dirección menor y el byte más significativo es almacenado en la dirección mayor.

El número de celdas de memoria de un computador es referido como su tamaño de memoria.



CONTENIDO DE LA MEMORIA EN LA DIRECCION 4 : 01000111

FIGURA 2.9 MEMORIA PRINCIPAL



CONTIENE ALMACENADA EN LA POSICION 4 LA PALABRA :1234H

FIGURA 2.10 ALMACENANDO UNA PALABRA EN LA MEMORIA.

CAPITULO 3

ARQUITECTURA DEL MICROPROCESADOR 8088

En este capítulo estudiaremos primero como esta construido internamente el microprocesador 8088, para luego hacer un bosquejo de la arquitectura del 8088 con sus periféricos; es decir, veremos que los periféricos no estan conectados directamente al 8088; si no, que hay dispositivos de interface entre el 8088 y sus periféricos.

3.1 ARQUITECTURA INTERNA DEL 8088

En esta sección, veremos las componentes internas del chip 8088 y la manera en la cual se comunica con la memoria. Además, estudiaremos el conjunto de registradores del 8088.

Un diagrama de bloque de la CPU 8088 aparece en la figura 3.1 en donde puede verse que los datos viajan entre la CPU y la memoria principal a través de la interfaz de memoria. Los datos de entrada a ser interpretados como instrucciones viajan a través del BUS C y son colocados en la cola de bytes de la corriente de instrucciones.

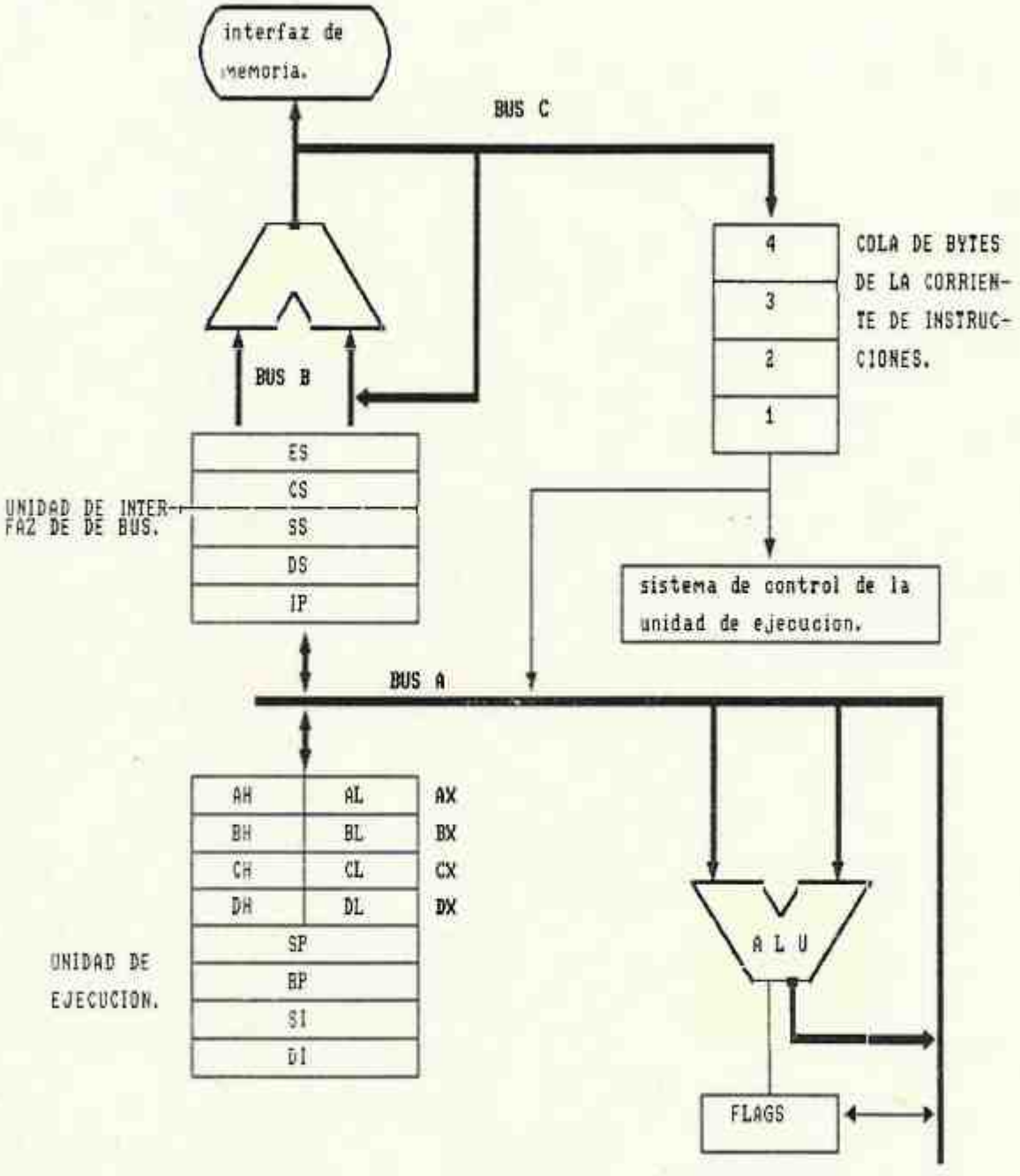


FIG. 3.1. DIAGRAMA DE BLOQUE DEL 8088

El sistema de control de la unidad de ejecución toma una instrucción (si es que hay) de esta cola y la envía a la unidad de ejecución para ser interpretada y ejecutada; mientras tanto, la unidad de interfaz de bus esta tratando de rellenar la cola de bytes de la corriente de instrucciones. Este diseño es llamado PIPELINED (entubado) y hace lo posible para que, cuando la CPU ejecute una instrucción haya otra esperando ser ejecutada. Este diseño da gran velocidad a la ejecución de varias instrucciones; ya que, al terminar la ejecución de una hay otra en la cola de bytes esperando ser ejecutada, de no ser así tendría que tomar esta siguiente instrucción desde la memoria lo cual le tomaría mucho más tiempo. Por ser PIPELINED el 8088 puede ejecutar instrucciones mas rápidamente.

Dentro de la unidad de ejecución esta la UNIDAD ARITMETICA LOGICA (ALU). Esta componente es la responsable de ejecutar todos los cálculos y comparaciones aritméticas. Además de producir un resultado aritmético la ALU siempre utiliza diferentes FLAGS para indicar el estado del resultado. Los FLAGS serán estudiados al final de esta sección.

3.1.1 CONJUNTO DE REGISTRADORES DEL 8088

El microprocesador 8088 tiene internamente 14 registradores que pueden almacenar datos binarios. Todos los registradores son de 16 bits, estos son mostrados en la figura 3.2

Desde el punto de vista del programador, los registradores son las componentes más importantes dentro del microprocesador, es por eso que estaremos usando y haciendo referencia constantemente a los diferentes registradores del 8088 al desarrollar nuestros propios programas en lenguaje ensamblador.

3.1.1.1 REGISTRADORES DE PROPOSITO GENERAL

Los registradores AX, BX, CX y DX son registradores de 16 bits de propósito general, cada uno de estos puede ser utilizado opcionalmente como un par de registradores de 8 bits (1 byte). Estos registradores son mostrados en la figura 3.3. El byte de la mitad de la izquierda (la más significativa) es referenciado como el byte "HIGH" (alto), y el byte de la mitad de la derecha (menos significativa) es conocido como el byte "LOW" (bajo). Así por ejemplo, el registrador AX puede ser dividido en los registradores AH (A High) y AL (A Low). De esta manera la mitad de la izquierda de cada uno de los registradores generales son llamados: AH, BH, CH y DH. Estos registradores pueden ser accedidos directamente.

Similarmente, la mitad de la derecha de cada uno de los registradores generales son llamados: AL, BL, CL y DL.

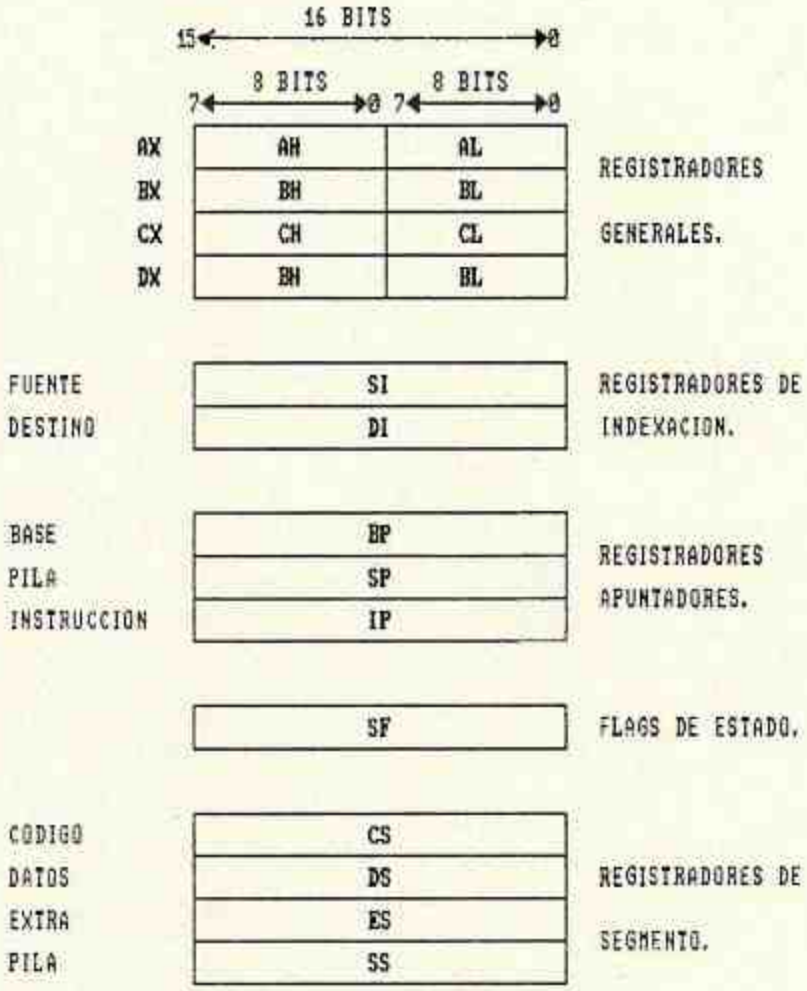


FIG. 3.2 REGISTRADORES INTERNOS DEL MICROPROCESADOR 8088.

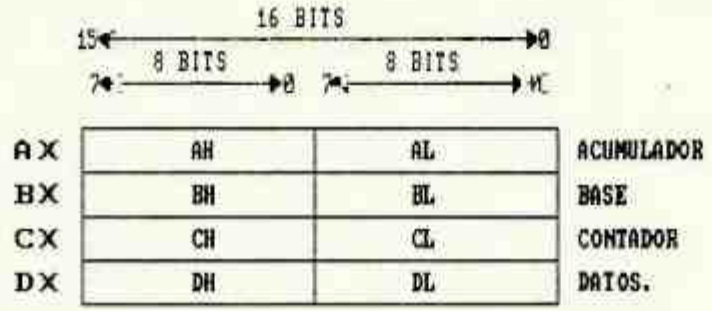


FIG.3.3 REGISTRADORES DEL 8088 DE PROPOSITO GENERAL.

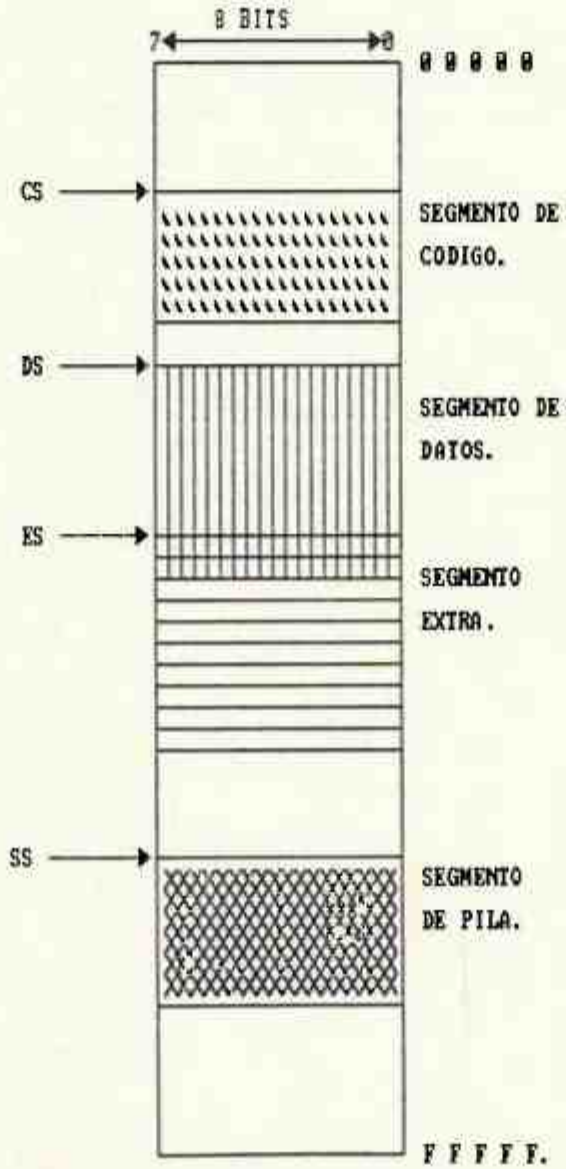


FIG. 3.4 LOS REGISTRADORES DE SEGMENTO CS, DS, ES Y SS DEFINEN 4 SEGMENTOS DE 64K BYTES.

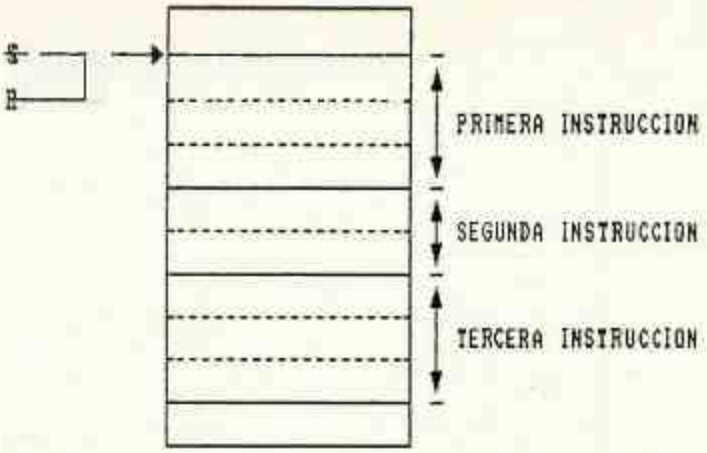
Aunque todos los registradores generales son usados para muchas operaciones, cada uno de estos son usados para cierto propósitos específicos. Esto es lo que da salida a los nombres ACUMULADOR, BASE, CONTADOR y DATOS para los registradores AX, BX, CX y DX respectivamente.

3.1.1.2 REGISTRADORES DE SEGMENTO

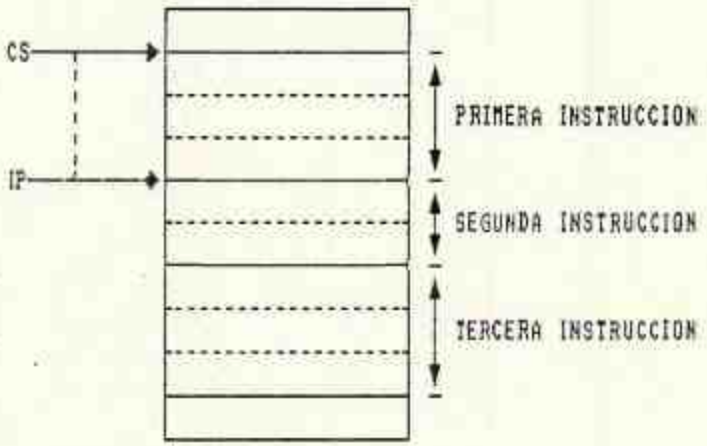
El microprocesador 8088 tiene cuatro registradores que contienen direcciones de segmento. Estos son llamados: El registrador de segmento de código, CS, el registrador de segmento de datos, DS, el registrador de segmento de pila, SS, y el registrador de segmento extra, ES. Por medio de estos es que en cualquier instante de tiempo cuatro diferentes segmentos de 64k byte son definidos. Estos son mostrados en la figura 3.4.

Si los registradores de segmento CS, DS, ES, y SS todos contienen el mismo valor, entonces un solo segmento de 64k bytes es definido.

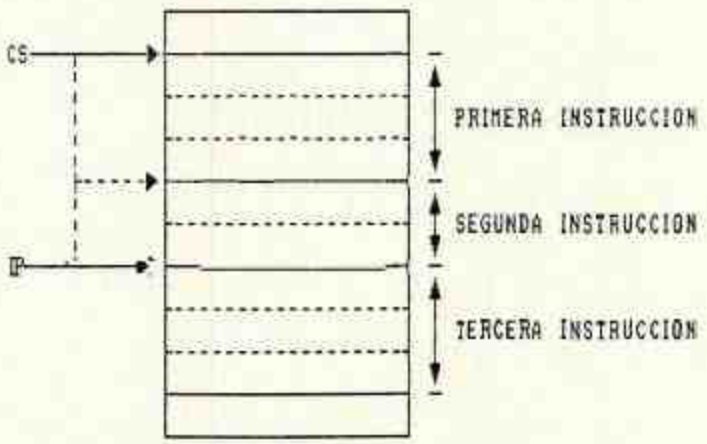
El segmento de código definido por CS, contiene el programa que esta siendo ejecutado. El pointer de instrucción, IP, que apunta a la siguiente instrucción a ser ejecutada usa el registrador de segmento de código, CS, para formar la dirección "EFECTIVA" de la siguiente instrucción. La figura 3.5 muestra como el registrador IP, esta variando en la ejecución del programa.



A) EL IP APUNTA A LA PRIMERA INSTRUCCION ANTES DE COMENZAR A EJECUTAR EL PROGRAMA.



B) EL IP APUNTA A LA SEGUNDA INSTRUCCION CUANDO LA PRIMERA ESTA SIENDO EJECUTADA.



C) EL IP APUNTA A LA TERCERA INSTRUCCION CUANDO LA SEGUNDA ESTA SIENDO EJECUTADA.

FIG. 3.5 EL IP APUNTA A LA SIGUIENTE INSTRUCCION A SER EJECUTADA.

El segmento de datos definido por DS contiene los datos usados por el programa. El segmento definido por SS contiene la pila; el segmento extra definido por ES es usado para una variedad de propósitos útiles. El registrador del segmento extra es también usado por ciertas instrucciones de manipulación de string.

3.1.1.3 REGISTRADORES DE INDEXACION

Los registradores de indexación SI y DI son registradores de 16 bits que son usados para varios propósitos diferentes. Ellos pueden ser usados de una manera similar a los registradores generales. Para almacenamiento temporal cuando se mueven 16 bits de datos a y desde la memoria (usando la instrucción MOV). El principal uso de los registradores SI y DI es en conjunción con los varios modos de direccionamiento.

3.1.1.4 REGISTRADORES DE PILA

La pila es una región de la memoria que es colocada aparte para almacenamiento temporal de datos. El apuntador de la pila SP es un registrador de 16 bits que contiene la dirección desajustada del tope de la pila. Una ilustración se puede ver en la figura 3.6. La dirección efectiva es encontrada sumándole la dirección de segmento almacenada en el registrador de segmento de pila, SS.

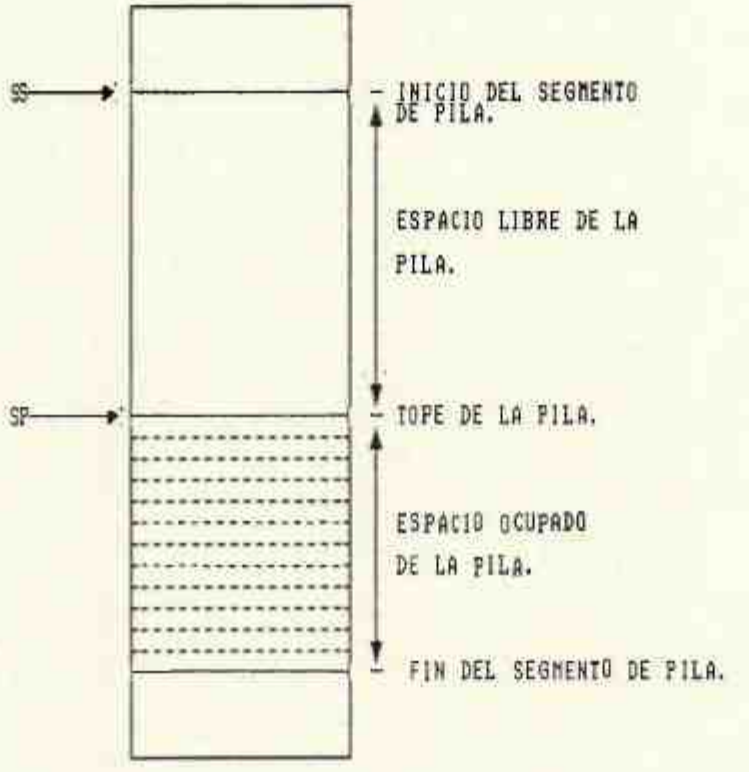


FIG. 3.6 EL REGISTRADOR SP APUNTA HACIA EL TOPE DE LA PILA.

La pila es usada por el 8088 para grabar las direcciones de retorno cuando una subrutina es llamada. Es también usada para guardar valores de los registradores cuando una interrupción es llamada. Las subrutinas serán descritas en el capítulo 6. Y las interrupciones en el capítulo 5.

El base pointer, BP, es un registrador de 16 bits que es usado para accesar datos en el segmento de pila. El registrador de segmento, SS, es usado para encontrar la dirección efectiva asociada con una dirección desajustada en BP.

3.1.1.5 REGISTRADORES DE INSTRUCCION

El pointer de instrucción, IP, es un registrador de 16 bits que contiene la dirección desajustada de la siguiente instrucción a ser ejecutada; Como se muestra en la figura 2.5. Cuando una instrucción es ejecutada el pointer de instrucción es automáticamente incrementado el número de veces necesarias para apuntar a la siguiente instrucción. Las instrucciones pueden ser desde uno a seis bytes de longitud. Por lo tanto, el conteo del programa puede ser incrementado desde uno a seis dependiendo de la instrucción que esta siendo ejecutada.

3.1.1.6 REGISTRADORES DE ESTADO

El 8088 contiene un registrador de estado el cual contiene seis flags de estado y tres flags de control. Los seis flags de estado son: El flag carry (C), el flag cero (Z), el flag de signo (S), el flag fuera de rango (O), el flag auxiliar (A) y el flag de paridad (P). Los flags de control son: El flag de interrupción (I), el flag de trap (T) y el flag de dirección (D). Cada uno de los flags es un bit del registrador de estado.

3.1.1.6.1 EL FLAG CARRY

El flag carry es el bit cero del registrador de estado. El bit carry es cambiado por tres tipos de instrucciones diferentes: Las primeras son las instrucciones aritméticas, así como la suma y resta. El segundo grupo de instrucciones que pueden cambiar el bit carry son las instrucciones SHIFTING y ROTANTING. Finalmente, el bit carry puede ser colocado a uno con la instrucción STC (set carry) y borrado a cero con la instrucción CLC (clear carry).

3.1.1.6.2 EL FLAG CERO

El flag cero es el bit seis del registrador de estado. Este flag es colocado a uno cuando el resultado de una instrucción es cero. Si el resultado de una instrucción no es cero el flag, Z, es borrado a cero.

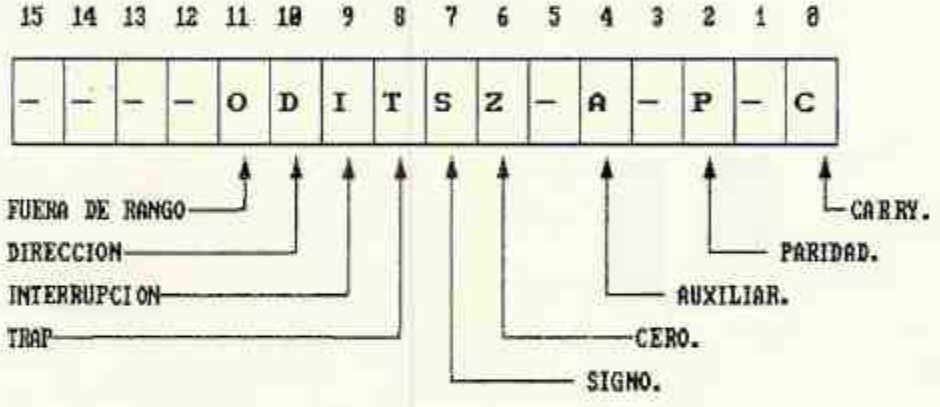


FIG. 3.7 EL REGISTRADOR DE ESTADO.

3.1.1.6.3 EL FLAG DE SIGNO

El flag de signo es el bit siete del registrador de estado. Los números negativos son almacenados en el computador por el 8088 usando la representación en complementos de dos. En esta representación un número negativo es indicado cuando el bit siete (el mas significativo) de un byte es colocado a uno. Cuando el resultado de una instrucción deja el signo negativo (bit siete de un byte o bit quince de una palabra), el flag S es colocado a uno. Si el resultado de una instrucción es positivo, (el bit de signo es cero), el flag S es borrado a cero.

3.1.1.6.4 EL FLAG FUERA DE RANGO

El flag fuera de rango es el bit once del registrador de estado. Es colocado en uno cuando el resultado de una operación señalado esta fuera de rango (overflow).

3.1.1.6.5 EL FLAG AUXILIAR

El flag de carry auxiliar es el bit cuatro del registrador de estado, contiene el carry desde el bit tres al bit cuatro; resultando de una operación ADICION o SUSTRACCION de ocho bits. El flag carry auxiliar es usado por el microprocesador cuando realiza adición binaria codificada (Binary Code Decimal).

3.1.1.6.6 EL FLAG DE PARIDAD

El flag de paridad es el bit dos del registrador de estado. Este es colocado en uno, si los ocho bits menos significativos del resultado de una instrucción contiene un número par de bits uno. El flag de paridad será cero si una instrucción produce un resultado de ocho bits con un número impar de bits uno.

3.1.1.6.7 EL FLAG DE DIRECCION

El flag de dirección es el bit diez del registrador de estado. Este bit determina cuando los registradores de indexación SI y DI son automáticamente incrementados o decrementados en cierta instrucción de manipulación de strings. Si D=0, entonces SI y DI son incrementados. Si D=1, entonces SI y DI son decrementados. El flag D es borrado a cero con la instrucción CLD (Clear Direction Flag) y es colocado a uno con la instrucción STD (Set Direction Flag).

3.1.1.6.8 EL FLAG DE TRAP

El flag trap es el bit ocho del registrador de estado. Cuando es colocado a uno, producirá una interrupción después que una sola instrucción ha sido ejecutada.

3.1.1.6.9 EL FLAG DE INTERRUPCION

El flag de interrupción es el bit nueve del registrador de estado. Cuando él es borrado a cero, las interrupciones de hardware son disfrazadas y el 8088 no responderá a una instrucción. Cuando el flag, I, es colocado a uno las interrupciones son habilitadas y el 8088 dará servicio de interrupciones de hardware.

El flag I, es colocado a uno con la instrucción STI (Set Interrupt flag) y es borrado a cero con la instrucción CLI (Clear Interrupt flag).

En el capítulo anterior, vimos como se calculaban las direcciones efectivas por medio de las direcciones de segmento y desajustadas. Estas dos últimas, el 8088 las posee en sus registradores. Por ejemplo, para encontrar la dirección efectiva de la siguiente instrucción a ser ejecutada , el 8088 calcula la suma del contenido del registrador de segmento de código y el del puntero de instrucción, CS + IP. Así, cada vez que se va a direccionar la posición de memoria, se hace a través de un registrador de segmento y otro que contiene una dirección desajustada; en la tabla 3.1 se muestran cada uno de los registradores de segmento y el grupo de registradores que por default se referencian a cada uno de ellos.

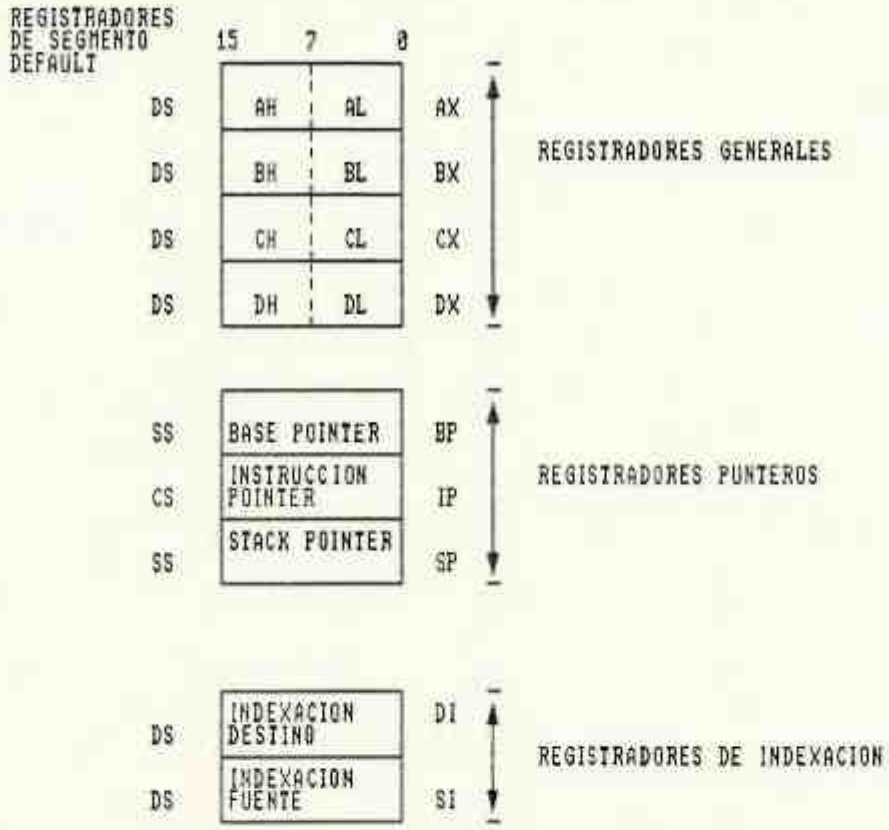


TABLA 3.1 REGISTRADORES DE SEGMENTO AGRUPADOS POR DEFAULT PARA EL CALCULO DE DIRECCIONES EFECTIVAS.

3.2 ARQUITECTURA EXTERNA DEL 8088

3.2.1 CONTROLADORES DEL SISTEMA

Ciertas funciones, tales como las transferencias de bloques de información de un lugar a otro (transferencia DMA), o el control de las interrupciones de E/S, se realizan mas eficazmente si son otros elementos independientes de la CPU los que lo llevan a cabo. Tales dispositivos reciben el nombre de "CONTROLADORES DEL SISTEMA". La existencia de los controladores no solo hace posibles ciertas funciones, si no que a la vez libera a la CPU de la supervisión de ciertas tareas distribuyendo la inteligencia de la máquina a lugares fuera de la CPU. Mas adelante veremos dos chips controladores del sistema, el controlador programable DMA 8237 y el controlador de interrupciones 8259.

3.2.2 CONTROLADORES DE DISPOSITIVOS

Hay un cierto número de dispositivos conectados a la computadora central como son: El teclado, monitor de vídeo, discos flexibles o impresora. Tales elementos reciben el nombre de: "DISPOSITIVOS DE ENTRADA SALIDA" (E/S). Un dispositivo de entrada salida nunca se conecta directamente al bus principal del computador, sino que a través de una especie de cable, se conecta a su propio (o compartido) CONTROLADOR DE DISPOSITIVO, que es el que a su vez se conecta al bus principal. Los controladores de dispositivos actúan de "INTERFAZ" entre los dispositivos y la computadora; mientras que los controladores del sistema realizan funciones internas.

A cada controlador del sistema y de dispositivo se le asocia una dirección de la misma forma que se hizo con las celdas de memoria. Existen dos filosofías diferentes en lo relativo a colocar direcciones de E/S: En un área separada del área de memoria, o en una parte del espacio de memoria.

Cuando las direcciones asignadas a los dispositivos de entrada/salida se mantienen en un área separada de la memoria. Las señales del bus de control permiten distinguir entre los accesos a memoria o a E/S, y en el procesador se utilizan distintas instrucciones de transferencia de datos para cada una. Por otra parte, cuando las direcciones de los dispositivos de E/S y las celdas de memoria estan en la misma área, son distinguidos solo por sus direcciones. Consideramos importante mencionar que hay chips controladores de dispositivos los cuales son de propósito especial, como el controlador de discos flexibles, o el CRT; y otros son de propósito general como el controlador de interfaz en paralelo, o el de serie. Estos dos últimos pueden utilizarse para buscar gran variedad de dispositivos de computadora. Además, los controladores del sistema y los dispositivos son ahora programables, de manera que pueden adaptarse y realizar varias funciones bajo control de SOFTWARE.

3.2.3 SISTEMAS BASADOS EN EL 8088

La figura 3.8 muestra un diagrama de bloque de un sistema formado por una serie de chips INTEL interconectados,formando una computadora. En particular, el diagrama

muestra como se conectan los tres procesadores siguientes:

- La unidad central de proceso (CPU) 8088.
- El procesador de datos numéricos (NDP) 8087.
- El procesador de entrada/salida (IOP) 8089.

Estos tres procesadores forman una potente y completa unidad de tratamiento de datos, localizada en la parte izquierda de la figura 3.8. La CPU 8088 se encarga de gestionar todo el sistema. Distribuyendo tareas a los otros procesadores, los cálculos aritméticos de alta precisión se realizan en el procesador de datos numéricos 8087, y el procesador de entrada/salida 8089 (IOP, Input/Output) es el encargado de los movimientos de bloques dentro de la memoria de las unidades CRT y entre la memoria central de la computadora y las memorias del sistema de visualización del video.

Este grupo tiene su propia estructura de bus local. Las líneas de control, provenientes de la CPU 8088, entran al NDP 8087 y al IOP 8089, y se encargan de decidir quien tiene control del bus local en cada momento (solo un modulo puede utilizar el bus a la vez).

Hay tres chips de interfaz de bus:

- El controlador de bus 8288.
- El latch de dirección octal 8082.
- El transceptor de datos octal 8286.

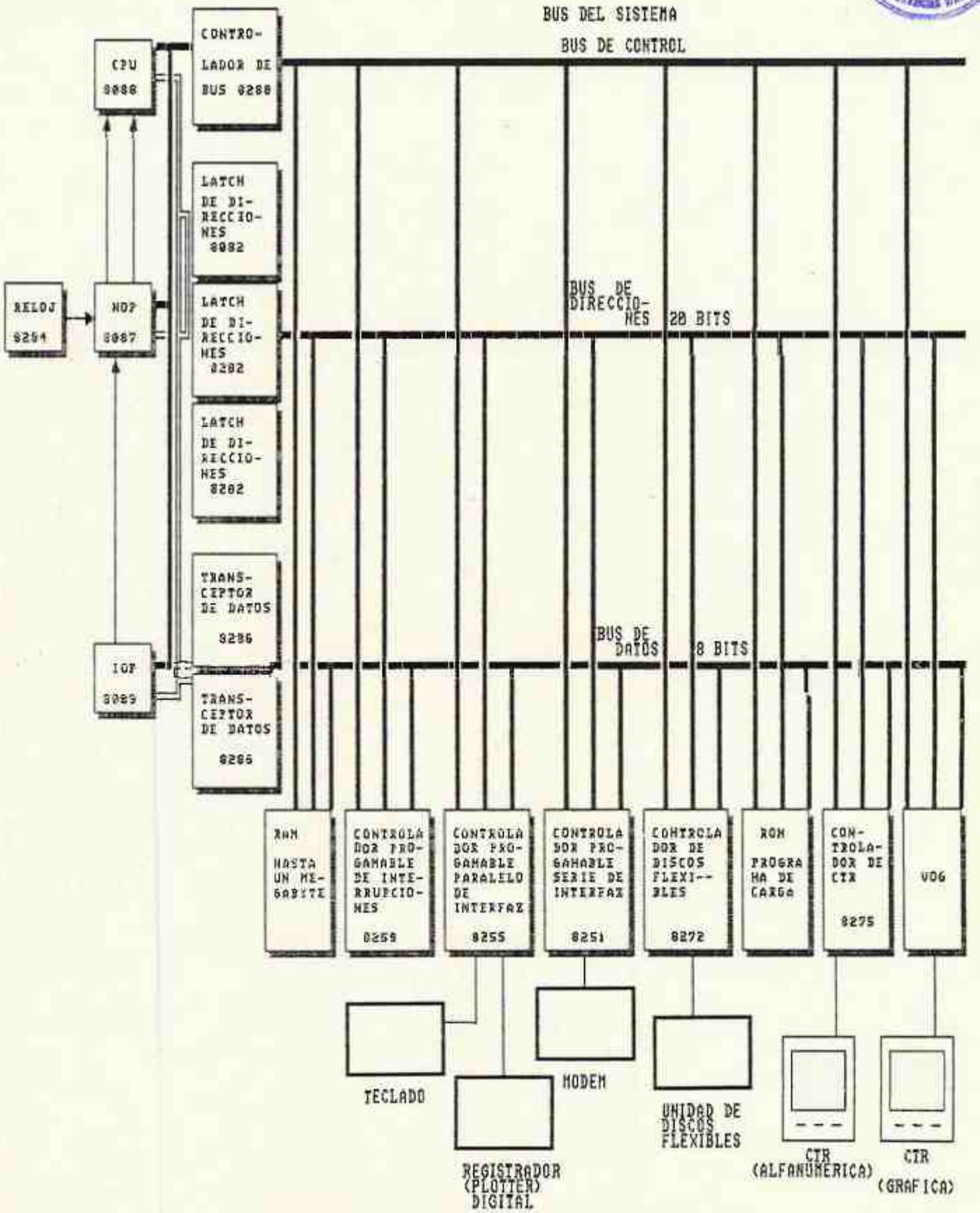


FIGURA 3.8 DIAGRAMA DE BLOQUES DE UNA COMPUTADORA DE 8 BITS BASADA EN LA CPU 8088.

Un controlador de bus 8288, tres latch de dirección octal 8082 y un transceptor de datos son necesarios para conectar el bus local compartido por los tres procesadores al bus del sistema (no local). Estos chips son necesarios por dos razones:

1. Para reorganizar las señales.
2. Para suministrar la corriente suficiente a los numerosos dispositivos conectados al bus del sistema.

Como interfaz con cada uno de los sub-buses se utilizan diferentes tipos de chips: El controlador de bus 8288, como interfaz con el sub-bus de control; el latch de dirección octal 8082, como interfaz con el sub-bus de control, y el transceptor de datos octal 8286 para el sub-bus de datos. Hay un cuarto sub-bus, el de alimentación externa y a cada dispositivo del sistema.

Hay dos controladores del sistema programable:

- El controlador programable de interrupciones 8259.
- El controlador programable de DMA 8237.

Estos dos controladores realizan las funciones internas de transferencias por DMA y control de interrupciones ya discutidas.

Hay también cuatro controladores de dispositivos programables:

- El controlador programable serie de interfaz 8251.
- El controlador programable paralelo de interfaz 8255.

- El controlador programable de CTR 8275.
- El controlador programable de discos flexibles 8273.

Cada uno de los dispositivos externos esta conectado a uno de estos controladores, que a su vez se conectan con el bus principal del sistema. Es importante fijarnos que, de hecho, la memoria no es mas que otro dispositivo de los procesadores. El sistema descrito puede soportar hasta un Megabyte de memoria RAM y ROM. La ROM se utiliza para almacenar un pequeño programa de autocarga (bootstrap) para la inicialización del controlador de discos flexibles. El programa carga el primer sector del disco flexible en memoria. Este primer sector contiene un programa que a su vez carga en memoria el resto del sistema operativo; este se encarga de inicializar los diversos controladores, activándolos y dejándolos en disposición de recibir ordenes.

El 8088 tiene un bus de datos externos de 8 bits, por lo que simplifica la conección de algunos controladores de dispositivos de E/S, debido a que en su mayor parte utilizan buses de datos de ocho bits.

La computadora diseñada podría ser muy válida como sistema de desarrollo de gráficos. El controlador de video visualizaría gráficos de funciones matemáticas mas o menos complejos, artísticos o dibujos animados de escenas complicadas. El teclado se utilizaría para entrar y modificar programas, el disco flexible sería útil a la hora de almacenar tales programas y el MODEN conectaría al computador con alguna red computadora, lo que

permitiría tener acceso a grandes bases de datos y a ciertos programas públicos, el controlador de CTR se encargaría de visualizar el texto de los programas y cualquier información alfanumérica que se utilizase.

3.3 EL CONJUNTO DE INSTRUCCIONES DEL 8088

El conjunto completo de funciones primitivas que un microprocesador puede realizar es conocido como su conjunto de instrucciones.

El conjunto de instrucciones del 8088 consiste de seis tipos de instrucciones, las cuales son resumidas en la tabla 3.2.

Para tener éxito en la programación en lenguaje ensamblador para el 8088, requiere un entendimiento de todos estos tipos de instrucciones, y este será nuestro siguiente objetivo. Para mas información en detalle sobre cualquier instrucción especifica, referirse al apéndice B.

Las instrucciones de transferencia de datos nos sirven para mover datos desde un punto a otro. En general, los datos pueden ser movidos de cualquiera, un byte o palabra a un tiempo.

1. INSTRUCCIONES DE TRANSFERENCIA DE DATOS.

MOV	Mover
PUSH,POP	Operaciones de pila
XCHG	Intercambiar
IN,OUT	Puertos de entrada/salida
XLATB	Trasladar.

2. INSTRUCCIONES ARITMETICAS.

ADD	Adicion
INC	Incrementar
SUB	Resta
DEC	Decrementar
NEG	Negacion
CMP	Comparacion
MUL	Multiplicacion
DIV	Divide

3. INSTRUCCIONES LOGICAS.

NOT	Complemento
AND,OR	Operadores logicos
XOR	O exclusivo
TEST	Test bits
SHL,SHR	Shift izquierdo/derecho
ROL,ROR	Rotate izquierdo/derecho

4. INSTRUCCIONES DE MANIPULACION DE STRING.

MOVS	Mueve string
CMPS	Compara string
SCAS	Busca string
LODS	Carga de string
STOS	Almacena dentro de string

5. INSTRUCCIONES DE TRANSFERENCIA DE CONTROL.

CALL	Llamada a subrrutina
RET	Retorno de subrrutina
JMP	Salto incondicional
JZ,JNZ,..	Salto condicional
LOOP	Iteracion
LOOPNE,...	Iteracion condicional
INT	Interrupcion
IRET	Retorno desde INT.

6. INSTRUCCIONES DE PROCESAMIENTO Y CONTROL.

CLC,STC,...	Borrar/colocar flags
HLT	Halt procesador

TABLA 3.2 CONJUNTO DE INSTRUCCIONES DEL 8088.

El movimiento de datos puede ser visto como un simple concepto, pero la situación es complicada porque muchos modos diferentes son disponibles para direccionar los datos a ser movidos. Estos diferentes modos de direccionamiento de datos son también usados por los tipos de instrucciones aritméticas y lógicas. Así nosotros ganaremos mucho terreno para estudiarlas ahora.

CAPITULO 4

ESTRUCTURA DE UN PROGRAMA

La estructura general de un programa en lenguaje ensamblador para el microprocesador 8088, puede ser visto como una colección de segmentos. Una ilustración se tiene a continuación:

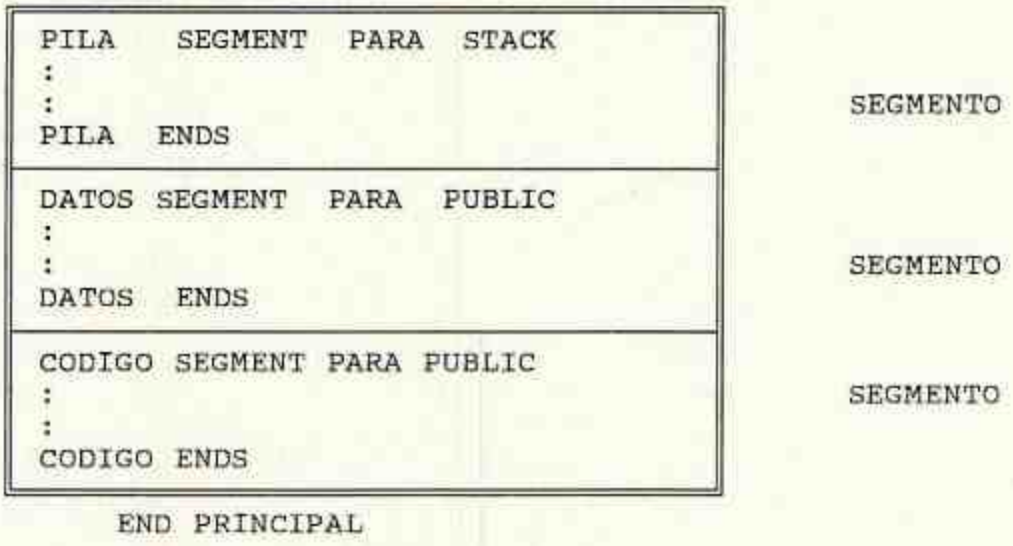


FIG. 4.1 Estructura de un Programa Fuente en Lenguaje Ensamblador.

Un programa básico en lenguaje ensamblador debe tener al menos dos segmentos:

El segmento de pila y

El segmento de código.

En esta estructura del programa se usan unas palabras como: SEGMENT, ENDS y PUBLIC; llamadas seudo-operadores.

4.1. PSEUDO-OPERADORES

Los seudo-operadores o mandatos son comandos para el lenguaje ensamblador, específicamente para el microprocesador 8088. Estos pueden ser usados para colocar o definir los segmentos y procedimientos (por ejemplo: Subrutinas), definir símbolos, reservar localizaciones de memoria para almacenamiento temporal y otras funciones.

Como vimos en el capítulo uno, las instrucciones del lenguaje ensamblador pueden tener a lo sumo 4 campos:

[etiqueta] pseudo-operador [operando] [;comentario]

Los corchetes indican que los campos son optativos y solamente el campo seudo-operador es siempre requerido.

El macroensamblador posee una variedad de mandatos, los más comúnmente usados se presentan en la tabla 4.1

MANDATOS			
ASSUME	DQ	EQU	PROC
DB	END	EXTRN	PUBLIC
DW	ENDP	INCLUDE	SEGMENT
DD	ENDS		

TABLA 4.1 Pseudo-operadores Generales del Ensamblador

Los seudo-operadores pueden ser divididos en cuatro grupos de acuerdo a su función. Estos grupos son:

1.- **Los seudo-operadores de Definición de Símbolos:**

Asignan un nombre simbólico a una expresión. La expresión puede ser una constante de 16 bits, una referencia de dirección, otro nombre simbólico, un identificador de segmento (Prefix) o una etiqueta de instrucción.

Los seudo-operadores "EQU" e "=" son de uso similar, pero se diferencian en que los símbolos definidos con "=" se pueden redefinir, pero los símbolos definidos con EQU son permanentes. Además, "=" solamente se puede usar para expresiones numéricas, mientras que EQU puede ser usada para expresiones numéricas y alfanuméricas.

2.- **Los seudo-operadores de Definición de Datos:**

Asignan espacio de memoria para las variables. Entre estos tenemos: DB, Define Byte(Define Byte); DW, Define Palabra(Define Word); DD, Define Doble Palabra(Define DoubleWord) y DQ, Define Cuatro Palabras(Define Quadword). Cuando se define una variable, se puede especificar cualquier valor para el espacio de memoria o reservar primero el espacio necesario y después dar el valor. Por ejemplo:

```
MYVAR DB 4
```

Coloca al byte de la variable MYVAR el valor de 4, mientras que

MYVAR DB?

simplemente reserva un byte como espacio para ella.

El seudo-operador, DB también puede aceptar un texto como una expresión; permite almacenar menús, mensajes, Prompts, y otros textos en memoria, como por ejemplo:

PROMPT DB 'Presione cualquier tecla para continuar....'

Con esta instrucción se guarda en la variable PROMPT el mensaje que aparece como texto.

Note que se debe encerrar el texto entre apóstrofes; éstos dicen al ensamblador donde comienza y termina el texto.

3- **Los Seudo-operadores de Especificación Segmento/ Procedimiento**

SEGMENT y ENDS dividen el programa fuente en segmentos. Un programa puede tener cuatro tipos de segmentos: de Datos, de Código, Extra y de Pila.

Los mandatos PROC y ENDP, marcan el inicio y el final de un procedimiento o subrutina. Un procedimiento siempre debe tener una de las dos distancias atribuidas: NEAR o FAR.

4- **Los Seudo-operadores de Referencia Externa:**

Permiten repartir información entre módulos, que eventualmente se encadenan para formar un programa. El seudo-operador PUBLIC hace disponibles uno o

más símbolos para otros módulos. Este dice al encadenador (LINK) "Aquí está la lista de ITEMS que tienes, si necesitas alguno de ellos, dilo y tómalo de mí". Un mandato PUBLIC puede listar nombres de variables, etiquetas (incluyendo etiqueta PROC), y definir símbolos utilizando el seudo-operador EQU o "=". El EXTRN dice al encadenador (LINK) obtener un ITEMS de otro módulo no especificado. EXTRN tiene la forma general: EXTRN nombre: tipo[,...] Aquí el nombre es un símbolo definido (y declarado PUBLIC) en algún otro módulo y tipo puede ser cualquiera de los siguientes:

* Si nombre es una variable, entonces tipo puede ser Byte, Palabra, Doble-palabra, o Cuatro palabras.

* Si nombre es una etiqueta de procedimiento o subrutina, entonces tipo puede ser NEAR o FAR.

* Si nombre es una constante definida por un mandato EQU o "=", entonces el tipo debe ser ABS.

PUBLIC y EXTRN son usados generalmente para repartir subrutinas. Por ejemplo, para correr una subrutina llamada SORT desde un programa principal, el módulo que contiene SORT debe incluir PUBLIC SORT y el programa principal debe contener:

EXTRN SORT: NEAR.

El mandato INCLUDE inserta un archivo fuente especificado en el archivo fuente corriente, para ensamblarlos juntos. Se puede usar también INCLUDE para leer macros dentro de un programa.

4.2. OPERADORES

Un operador es un modificador, usado en el campo operando de una instrucción del lenguaje ensamblador. Los operadores más comúnmente usados caen dentro de tres categorías: Aritméticos, Retorno-valor, y atributo.

4.2.1 Operadores Aritméticos

Los operadores aritméticos combinan operandos numéricos y producen un resultado numérico. El ensamblador provee los operadores: Suma(+), Resta(-), multiplicación(*), División(/) y MOD(que devuelve el residuo de una operación de división).

OPERADOR ARITMETICO	FORMATO	FUNCION
+	VALOR1 + VALOR2	Suma el valor1 y valor2, y retorna la suma
-	VALOR1 - VALOR2	Resta valor2 de valor1, y retorna el resultado
*	VALOR1 * VALOR2	Multiplica valor1 por valor2, y retorna el producto
/	VALOR1 / VALOR2	Divide valor1 entre valor2, y retorna el cociente.
MOD	VALOR1 MOD VALOR2	Divide valor1 entre valor2, y retorna el residuo

TABLA 4.2 OPERADORES ARITMETICOS.

4.2.2 Operadores Retorno de Valor

Los operadores de este grupo, proveen información acerca de las variables o etiquetas en un programa. El operador $, retorna el valor del tamaño de la variable. Este

operador es adecuado para que el ensamblador haga el cálculo del largo de un texto de string.

SEG y OFFSET retornan el número de segmento y el desajuste de la variable o etiqueta, respectivamente. Por ejemplo, las instrucciones siguientes cargan los valores de segmento y desajuste de la variable TABLA dentro de AX y BX respectivamente.

```
MOV AX, SEG TABLA
MOV BX, OFFSET TABLA
```

OPERADOR	FORMATO	FUNCION
$	$ Variable	Retorna el valor del tamaño de la variable.
SEG	SEG Variable o SEG Etiqueta	Retorna el valor del segmento de la variable o etiqueta.
OFFSET	OFFSET Variable o OFFSET Etiqueta	Retorna el valor desajuste de la variable o etiqueta.

TABLA 4.3 OPERADORES RETORNO - VALOR.

4.2.3 Operadores de Atributo:

El operador PTR (POINTER) especifica el tipo BYTE, PALABRA, DOBLE PALABRA o CUATRO PALABRAS o el atributo de distancia NEAR o FAR de un operando. PTR es a menudo usado para especificar el tamaño del dato sobre el cual tú estas operando.

Por ejemplo, la oración

MOV [BX], 1

Producirá un mensaje de error, porque el ensamblador desconoce si BX esta señalando a una posición de byte o palabra. Si éste apunta a una localización de byte, la forma correcta es:

MOV BYTE PTR [BX], 1

OPERADOR	FORMATO	FUNCION
PTR	Tipo PTR expresión	Sobrecarga el tipo Byte, word, (Dword o Qword) o distancia (NEAR o FAR) de un operando de dirección de memoria.
CS: DS: FS: SS:	seg-reg: dirección o exp. o seg-reg: etiqueta o seg-reg: variable.	Sobrecarga al segmento un atributo de etiqueta, variable o dirección-expresión.

TABLA 4.4 OPERADORES DE ATRIBUTO

4.3. PROGRAMAS FUENTE EN LENGUAJE ENSAMBLADOR

Como vimos anteriormente la memoria principal se organiza en segmentos: Segmento de código, Segmento de datos, Segmento de pila y Segmento extra. Esta organización de la memoria se refleja en el texto de un programa fuente en lenguaje ensamblador.

4.3.1 ESTRUCTURA GENERAL DE UN PROGRAMA EN LENGUAJE ENSAMBLADOR PARA EL MICROPROCESADOR 8088

Un programa en lenguaje ensamblador puede ser visto como una colección de segmentos.

```
PILA   SEGMENT  PARA  STACK
        :
PILA   ENDS
DATOS  SEGMENT  PARA  PUBLIC
        :
DATOS  ENDS
CODIGO  SEGMENT
        :
CODIGO  ENDS
```

FIG. 4.2 Estructura Básica de un Programa Fuente

El segmento de código es equivalente al cuerpo del programa.

El segmento de código es una colección de rutinas de la misma forma que el programa es una colección de segmentos.

A continuación, en la fig. 4.3 se presenta la estructura de un programa; pero ahora un poco más completo, con ciertas instrucciones necesarias para escribirlos.

```
PILA SEGMENT PARA STACK
   DB 256 DUP(0) ; Forma una pila de 256 bytes y en cada byte
                 ; se guarda el valor 0
PILA ENDS
DATOS SEGMENT PARA PUBLIC
;en esta parte se definen las variables
DATOS ENDS
CODIGO SEGMENT PARA PUBLIC
    INICIO PROC FAR ; Inicio de la rutina principal cuyo
                    ; nombre es INICIO
      ASSUME CS:CODIGO ; asuma que el registrador de segmento
                       ; de código apunta a CODIGO
        PUSH DS
        MOV AX,0
        PUSH AX
        MOV AX,DATOS ; Si hay segmentos de datos
        MOV DS, AX   ; Si hay segmentos de datos
        ASSUME DS: DATOS; Si hay segmento de datos
        RET ; como el PROC es FAR da el control al sistema
            ; operativo
    INICIO ENDP ; fin de la rutina principal
 CODIGO ENDS ; fin del segmento código
END INICIO; fin del programa
```

FIG. 4.3 Estructura Fundamental de un Programa Fuente

4.4. CREANDO ARCHIVOS FUENTE

El ensamblador puede usar archivos fuentes que contengan solamente caracteres standard ASCII. Si estás usando un procesador de palabras, ten en mente que no todos los procesadores de palabras escriben archivos usando solamente los caracteres ASCII. WORDSTAR es uno así, usa el modo no-documento, para grabar tus archivos. Primero prueba los caracteres ASCII de tu programa PRUEBA.ASM.(Ejemplo 4.1)

Del DOS, digita (negrita):

A> **TYPE PRUEBA.ASM**

```
  PILA SEGMENT PARA STACK
      DB 256 DUP(0)
  PILA ENDS
;
  CODIGO SEGMENT PARA PUBLIC
     PRINCIPAL PROC FAR
       ASSUME CS:CODIGO
          PUSH DS
          MOV AX,0
          PUSH AX
          CALL SUB
       RET
     PRINCIPAL ENDP
;
```

```
    SUB PROC NEAR
            MOV AH,2 ; posiciona el cursor
            MOV BH,0 ; en (0,0)
            MOV DX,0 ; dado por DX
             INT 10H
;
            MOV AH,14 ; despliega caracter 'A'
            MOV BH,0  ; en la pantalla
            MOV AL,'A'; en la posición dada
             INT 10H
           RET
     SUB ENDP
  CODIGO ENDS
END PRINCIPAL
```

Ej. 4.1 Programa Ejemplo:PRUEBA.ASM

Podrás ver el mismo texto, que tu grabaste como lo digitaste. Si ves caracteres extraños en tu programa puedes usar un editor diferente o procesador de palabras para guardar tus programas. También necesitarás una línea en blanco después de la declaración END en tu archivo.

Ahora comencemos a ensamblar el programa PRUEBA.ASM. Para ello usaremos el archivo MASM.EXE. Digita lo siguiente(negrita):

A>MASM

```
Microsoft (R) Macro Assembler Versión 1.25
Copyright (C) Microsoft Corp 1981,1982,1983. All rights reserved.
Source filename [ .ASM]:A:PRUEBA.ASM
Object filemane [PRUEBA.OBJ]:A:
Source listing [NUL.LST]:A:PRUEBA
Cross-reference [NUL.CRF]: A:PRUEBA
  49968 Bytes symbol space free
     0 Warning Errors
     0 Severe Errors
A>
```

Todavía no estamos listos para correr el programa. En este momento, el ensamblador (MASM.EXE) produjo un archivo llamado PRUEBA.OBJ, el cual encontrarás dentro de tu disco. Este es un archivo intermedio, llamado Archivo Objeto. Este archivo contiene tu programa en lenguaje máquina, junto a mucha información contabilizada, usada por otro programa llamado Link (enlace). Inmediatamente necesitamos el Link para tomar nuestro archivo .OBJ y crear una versión .EXE. Para ello, digita (negrita):

```
A>LINK
Microsoft (R) Object Linker Versión 2.00
Copyright (C) Microsoft Corp 1982. Ali rights reserved.
```

Object Modules [.OBJ]:**A:PRUEBA**

Run File[PRUEBA.EXE]:**A:**

List File [NUL.MAP]:

Libraries [.LIB]:

A>

Para verificar que tenemos todos los archivos necesarios, listemos todos los archivos PRUEBA creados hasta aquí, digitando lo siguiente (negrita):

A>**Dir PRUEBA.***

```
 Volume in drive A has no label
 Directory of a:\
 PRUEBA ASM    78    3-11-92    2:00p
 PRUEBA OBJ    46    3-11-92    4:00p
 PRUEBA EXE   640    3-11-92    4:10p
       3 file(s)  23552 bytes free
A>
```

CAPITULO 5.

INTERRUPCIONES

Antes de comenzar a estudiar las interrupciones, veremos dos archivos de lo que es el sistema operativo.

El sistema operativo se divide en BIOS (Sistema Básico de E/S) y DOS (Sistema Operativo de Disco). El BIOS proporciona las subrutinas de bajo nivel que el DOS usa para proporcionar las funciones de alto nivel. Sin embargo, hay solapamiento entre los dos sistemas.

El BIOS es un conjunto de programas para el 8088 que son almacenados en la computadora Personal. Cuando la computadora es encendida, estos programas reciben el control e inicializan el sistema completo de la computadora. Además proveen el mínimo soporte necesario de software para controlar los mecanismos que pueden estar ligados al computador.

El DOS es un programa de control que maneja varios recursos (memoria, espacio de disco, etc.) del computador. Este también provee el comando de lenguaje con el cual podemos controlar fácilmente las acciones del computador.

El ensamblador (MASM.EXE) es una herramienta de programación que podemos usar para convertir nuestros programas en lenguaje ensamblador a código objeto. El código objeto contiene los patrones actuales de bit que el microprocesador 8088 reconoce como instrucciones y datos.

Las interrupciones son instrucciones que se dan al microprocesador 8088, las cuales detienen la ejecución del programa y colocan el apuntador de Instrucciones (IP) en la posición de memoria específica donde se reanudará el procesamiento. El Apéndice C contiene Instrucciones de Interrupción del 8088.

INT número de Interrupción

Cuando se encuentra ésta instrucción, el control se transfiere a la dirección contenida en la localidad especificada por accionar cuatro veces el tipo de interrupción. Por ejemplo, la interrupción 10H transfiere el control a la localidad de memoria cuya dirección está contenida en 0000:0040. La tabla de direcciones de interrupción, o tabla de vectores de interrupción, está contenida en la parte baja de la memoria y, por tanto, la dirección de su segmento es 0000. La tabla 5.1 muestra la tabla de vectores de interrupción de la IBM PC. Nótese que el vector asociado con la interrupción 10H es F000:F065, con la interrupción 15H es F600:0000 y con la interrupción 21H es 0534:01B0 (subrayado). Estas direcciones corresponden a localidades de memoria contenida en el área del BIOS ROM.

```
-D 0000:0000
0000:0000  72 30 E3 00 ED 08 00 06-C3 E2 00 F0 E6 08 00 06  r0c.a...Cb.pf...
0000:0010  47 01 70 00 54 FF 00 F0-00 00 00 00 00 00 00 00  G.p.t..p........
0000:0020  A5 FE 00 F0 87 E9 00 F0-00 00 00 00 00 00 00 00  Z~ .p.i.p........
0000:0030  00 00 00 00 00 00 00 00-57 EF 00 F0 47 01 70 00  ........Wo.pG.p.
0000:0040  65 F0 00 F0 4D F8 00 F0-41 FB 00 F0 59 EC 00 F0  ep.pMx.pAx.pYl.p
0000:0050  39 E7 00 F0 59 F8 00 F0-2E EB 00 F0 D2 EF 00 F0  9g.pYx.p.h.pRo.p
0000:0060  00 00 00 F6 F2 E6 00 F0-6E FE 00 F0 40 01 70 00  ...vrf.pn ~ .p@.p.
0000:0070  33 FF 00 F0 A4 F0 00 F0-22 05 00 00 00 00 00 00  3..p$p.p".......
-D 0000:0080
0000:0080  07 08 E3 00 80 01 34 05-42 02 00 06 70 02 00 06  ..c...4.B...p...
0000:0090  F2 04 34 05 E0 13 E3 00-2E 14 E3 00 13 27 E3 00  b.4 ` .C...C..'C.
0000:00A0  13 0B E3 00 2E 01 70 00-00 00 00 00 00 00 00 00  ..C....p........
0000:00B0  00 00 00 00 00 00 00 00-6D 03 34 05 00 00 00 00  ........a.4.....
0000:00C0  EA 14 0B E3 00 00 00 00-00 00 00 00 00 00 00 00  ................
0000:00D0  00 00 00 00 00 00 00 00-00 00 00 00 00 00 00 00  ................
0000:00E0  00 00 00 00 00 00 00 00-00 00 00 00 00 00 00 00  ................
0000:00F0  00 00 00 00 00 00 00 00-00 00 00 00 00 00 00 00  ................
```

TABLA 5.1 VECTORES DE INTERRUPCION DE LA IBM PC.

En estas direcciones se encuentran instaladas las subrutinas, del sistema básico de E/S (BIOS) y a continuación de estas se encuentran instaladas las subrutinas del Sistema Operativo de Disco (DOS). De esta forma podemos hacer uso de las subrutinas BIOS y DOS.

5.1 INTERRUPCIONES 0 - 0FH

Las interrupciones 0H-4H, no son muy útiles para el programador en ensamblador, debido a que tienden a llamar funciones orientadas al sistema. La INT 5H puede utilizarse para imprimir el contenido de la pantalla del tubo de rayos catódicos (CRT) bajo el control de un programa. Si, por ejemplo, un programa genera salida de la pantalla, quizá sea deseable salvar esta salida automáticamente y es aquí donde la interrupción 5H es de gran utilidad. La INT 9H se activa cada vez que se presiona o

libera una tecla y su acción es simplemente detener la ejecución del programa.

Mas adelante será estudiada la interrupción 16H(E/S por teclado). Ambas (INT 9H e INT 16H) se refieren a la operación del teclado; sin embargo la INT 16H tiene una flexibilidad mayor que la INT 9H. La INT 10H, sin embargo, es la rutina de E/S del video y tiene muchas opciones. Esencialmente, todas las operaciones de E/S de la pantalla bajo el control del programa pueden llevarse a cabo con la ayuda de esta interrupción.

5.2. INTERRUPCION 10H:E/S DE VIDEO

La interrupción 10H tiene 16 interrupciones básicas, las cuales son controladas por el registro AH en el momento de generar la interrupción. El apéndice D describe todas estas opciones, así como los registros que deben activarse para controlar la salida.

5.3. INTERRUPCIONES 11H A 15H

La interrupción 11H proporciona un resumen de las opciones disponibles en la unidad del sistema básico. El resumen se proporciona en el registro AX. Cuando la interrupción 12H es llamada; ésta proporciona en el registro AX la cantidad (en bloques de 1024 bytes) de memoria de acceso aleatorio instalada en el sistema. La interrupción 13H trata de la administración de archivos. La interrupción 14H se emplea en conjunción con el adaptador de comunicaciones y, básicamente, se utiliza sólo cuando se llevan a cabo

operaciones de E/S. Finalmente, la interrupción 15H maneja operaciones de E/S en cassette.

5.4 INTERRUPCION 16H: TECLADO

Cuando se hace uso de ella con AH=0, detiene la ejecución del programa y espera hasta que se lee el siguiente caracter proveniente del teclado.

La mayor limitación de INT 16H es que está orientada a caracteres.

5.5 INTERRUPCIÓN 17H: IMPRESORA

Esta interrupción se utiliza para comunicar la impresora conectada al sistema. Al igual que con la mayor parte de las rutinas de interrupción, el valor contenido en AH determina cual opción de interrupción es seleccionada. Cuando AH=0, el servicio de interrupción imprime el caracter ASCII contenido en AL.

Algunas impresoras no responden de manera adecuada al conjunto de caracteres ASCII extendido de IBM y únicamente imprimen los primeros 128 caracteres que pertenecen al ASCII normal. Por lo que el lector debe consultar el manual de su impresora para ver como maneja los caracteres.

Cuando AH=1, un llamado a INT 17H inicializa la impresora; cuando AH=2 la interrupción proporciona, en AH, el byte del estado de la impresora.

5.6 INTERRUPCIONES 18H A 20H

La interrupción 18H se utiliza para cargar desde cassette
BASIC. La interrupción 19H provoca que la computadora reinicialice desde disco el sistema operativo. La interrupción 20H apunta al inicio del área del segmento prefijo del programa (PSP) y provoca la terminación de un programa.

5.7 INTERRUPCION 21H: LLAMADAS A FUNCIONES DEL DOS

La interrupción 21H tiene muchas opciones y estas se escogen dependiendo del contenido en AH. El apéndice E muestra un listado de estas funciones junto con una breve explicación de las mismas.

5.8 INTERRUPCIONES RESTANTES DEL DOS

Las demás interrupciones del DOS están destinadas, principalmente, a funciones del sistema y en general no son utilizadas por el programador.

5.9 PROGRAMAS EJEMPLOS

Ahora el lector tiene una idea de lo que son las interrupciones en el lenguaje ensamblador. Para hacer más claro el estudio, se presentan a continuación programas ejemplos.

```
 PILA SEGMENT PARA STACK
         DB 256 DUP(0)
 PILA ENDS
;
 CODIGO SEGMENT PARA PUBLIC
   PRINCIPAL PROC FAR
     ASSUME CS:CODIGO
          PUSH DS
          MOV AX,0
          PUSH AX
;
          MOV DH,10; FILA
          MOV DL,20; COLUMNA
;
        MOV AH,2 ; Instrucciones necesarias
          MOV BH,0 ; para posicionar
           INT 10H; el cursor
     RET
    PRINCIPAL ENDP
   CODIGO ENDS
 END PRINCIPAL
```

Ejemplo 5.1: Programa que posiciona el cursor en la fila(DH) 10, columna(DL) 20.

```
 PILA SEGMENT PARA STACK
         DB 256 DUP(0)
 PILA ENDS
;
 CODIGO SEGMENT PARA PUBLIC
   PRINCIPAL PROC FAR
     ASSUME CS:CODIGO
         PUSH DS
         MOV AX,0
         PUSH AX
;
         MOV AL,'A' ; Carga caracter en AL
;
         MOV AH,14  ; Instrucciones necesarias
         MOV BH,0   ; para desplegar el caracter
          INT 10H   ; que esta en AH
     RET
    PRINCIPAL ENDP
   CODIGO ENDS
 END PRINCIPAL
```

Ejemplo 5.2: Despliega un caracter en la posición en la que se encuentra el cursor.

```
PILA SEGMENT PARA STACK
        DB 256 DUP(0)
PILA ENDS
CODIGO SEGMENT PARA PUBLIC
  PRINCIPAL PROC FAR
    ASSUME CS:CODIGO
        PUSH DS
        MOV AX,0
        PUSH AX
        MOV DH,8   ; Fila
        MOV DL,12  ; Columna
        MOV AH,2   ; Posiciona
        MOV BH,0   ; el cursor
          INT 10H  ; en (8,12)
        MOV AL,'*' ; Carga caracter en AL
        MOV AH,14  ; Escribe caracter
        MOV BH,0   ; en la posición dada
         INT 10H   ; y avanza cursor
    RET
   PRINCIPAL ENDP
  CODIGO ENDS
END PRINCIPAL
```

Ejemplo 5.3: Despliega un caracter en la posición (8,12).

```
 PILA SEGMENT PARA STACK
      DB 256 DUP(0)
 PILA ENDS
;
 CODIGO SEGMENT PARA PUBLIC
   PRINCIPAL PROC FAR
     ASSUME CS:CODIGO
;
          PUSH DS
          MOV AX,0
          PUSH AX
;
          MOV AH,0 ; Detiene el programa
           INT 16H ; hasta que se oprime una tecla
      RET
;
   PRINCIPAL ENDP
;
   CODIGO ENDS
;
 END PRINCIPAL
```

Ejemplo 5.4: Lee un caracter vía teclado y lo carga en el registrador AL

```
PILA SEGMENT PARA STACK
        DB 256 DUP(0)
PILA ENDS
CODIGO SEGMENT PARA PUBLIC
  PRINCIPAL PROC FAR
    ASSUME CS:CODIGO
        PUSH DS
        MOV AX,0
        PUSH AX
        MOV DH,10 ; Fila
        MOV DL,25 ; Columna
        MOV AH,2
        MOV BH,0
         INT 10H
        MOV AH,0 ; Lee caracter
         INT 16H ; vía teclado y lo carga en AL
        MOV AH,14 ; Escribe
        MOV BH,0  ; el caracter
        INT 10H   ; digitado
    RET
  PRINCIPAL ENDP
  CODIGO ENDS
END PRINCIPAL
```

Ejemplo 5.5: Lee un caracter vía teclado y lo despliega en la posición (10,25)

CAPITULO 6.

PILAS Y SUBRUTINAS

6.1 PILAS

La pila es un conjunto de posiciones de memoria en la cual se pueden almacenar temporalmente datos. Una pila es diferente de cualquier otro grupo de posiciones de memoria, ya que en ésta los datos son colocados y tomados desde el tope. Muchas veces nos referimos a la pila como una lista LIFO (Last In, First Out = Ultimo en entrar, primero en salir).

La dirección desajustada del tope de la pila (la última posición llena) es almacenada en el apuntador, SP. La dirección efectiva del tope de la pila es encontrada combinando la dirección desajustada, contenida en el apuntador SP, con la dirección de segmento, contenida en el registrador SS. En la figura 6.1 se ilustra este procedimiento.

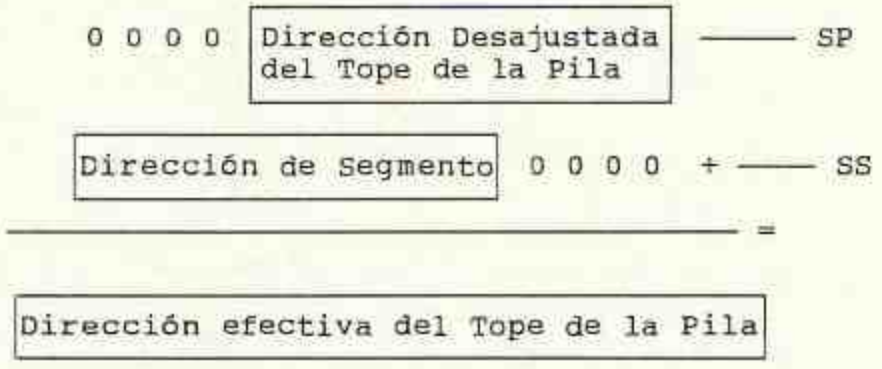


Fig 6.1 Calculo de la Dirección Efectiva del tope de la Pila.

Cuando los datos son colocados en la pila, el registrador, SP, es decrementado. Como los valores de los datos son colocados en la pila, éstos son puestos en posiciones de memoria con direcciones más bajas. Por ejemplo, si el registrador apuntador de pila, SP, contiene el valor 0FFFFH, el siguiente valor colocado en la pila, ocupará la posición con dirección 0FFFEH. Los datos pueden ser colocados y tomados de la pila usando las instrucciones PUSH y POP. (Los datos tienen el tamaño de dos Bytes).

6.2. INSTRUCCIONES PUSH Y POP

INSTRUCCION	OPERACION
PUSH reg	Coloca el contenido del registrador en la pila.
PUSH regseg	Coloca el contenido del registrador de segmento en la pila.
PUSHF	Coloca el registrador de estado en la pila.
PUSH men	Coloca 16 Bits desde la memoria.
POP reg	Toma el contenido (16 bits) del tope de la pila y lo coloca en el registrador.
POP reg seg	Toma el contenido 16(bits) del tope de la pila y lo coloca en el registrador de segmento.
POPF	Toma el contenido (16 bits) del tope de la pila y lo coloca en el registrador de estado.
POP men	Toma el contenido (16 bits) del tope de la pila y lo coloca en la posición de memoria especificada.

TABLA 6.1 INSTRUCCIONES PUSH Y POP DEL MICROPROCESADOR 8088.

Reg = { AX, BX, CX, DX, SP, BP, SI, DI }

RegSeg = { ES, CS, SS, DS }

Las instrucciones PUSH y POP siempre mueven dos Bytes (16 Bits) a y desde la pila, al mismo tiempo. Como muestra la tabla 6.1 las instrucciones POP y PUSH no trabajan únicamente con los registradores, sino también podemos hacerlo con una posición de memoria especificada; en este libro se trabajará sólo con registradores.

Por ejemplo, supongamos que deseamos colocar el valor del registrador AX en la pila (empilar AX). Las instrucciones de la Fig. 6.2 almacenan el valor de 1234H en el registrador AX, la primera instrucción coloca el contenido de AX en la pila.

```
Mov AX, 1234H
Push AX
```

Fig. 6.2 Instrucciones para empilar AX.

Note que cuando la instrucción PUSH AX es ejecutada, ocurren los pasos siguientes:

1. El registrador SP es decrementado en 1.
2. El contenido de AH es almacenado en la dirección que contiene SP.
3. Nuevamente el registrador, SP, es decrementado en 1.
4. El contenido de AL es almacenado en la dirección que contiene SP.

Una ilustración de este proceso aparece en la figura siguiente:

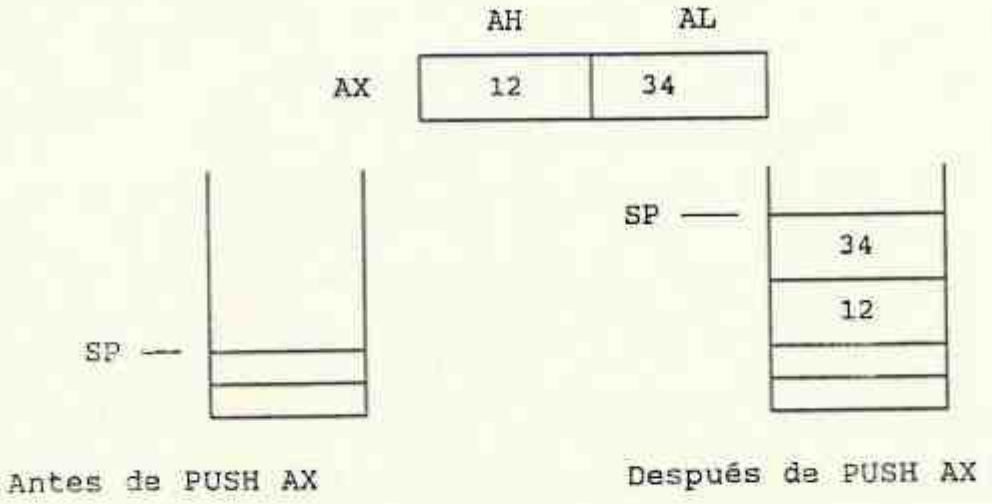


Fig. 6.3 Colocando AX en la pila.

Si la instrucción POP BX es agregada a las instrucciones de la Fig. 6.2 entonces los mismos dos valores colocados en la pila, serán sacados de ésta y almacenados en el registrador BX. Esta hace que ocurran los pasos siguientes:

1. El valor de la posición de memoria, cuya dirección está contenida en SP, es almacenado en BL.
2. El registrador, SP, es incrementado en 1.
3. El valor de la posición de memoria, de la nueva dirección contenida en SP, es almacenada en BH.
4. El registrador, SP, es incrementado en 1.

Este proceso se ilustra en la figura 6.4

Fig. 6.4 Desempilando valores en BX.

Note que:

- Los valores colocados en la pila, pueden ser sacados y almacenados en cualquier registrador.

- A las dos instrucciones de PUSH que aparecen en el inicio del programa, la máquina automáticamente ejecuta el POP respectivo al llegar a la instrucción RET.

- El número de veces que se ejecuta la instrucción PUSH debe ser igual al número de veces que se usa la instrucción POP, excepto las instrucciones explicadas en la nota anterior.

```
PILA SEGMENT PARA STACK
        DB 256 DUP(0)
PILA ENDS
CODIGO SEGMENT PARA PUBLIC
  PRINCIPAL PROC FAR
    ASSUME CS:CODIGO
        PUSH DS
        MOV AX,0
        PUSH AX
    MOV BX,25; se guarda el valor de 25 en BX
    PUSH BX  ; se guarda el contenido de BX en la pila
    POP AX   ; se saca el valor del tope de la pila y se
    guarda en AX
             ; (AX=0025, AH=00, AL=25)
    MOV AH,14; con estas instrucciones
    MOV BH,0; se despliega el contenido de AL
    INT 10H; en la pantalla
    RET
  PRINCIPAL ENDP
  CODIGO ENDS
END PRINCIPAL
```

Ejemplo 6.1: Programa que utiliza las instrucciones PUSH y POP para guardar y sacar un dato de la pila (25). El valor desplegado será un código ASCII correspondiente al N° 25.

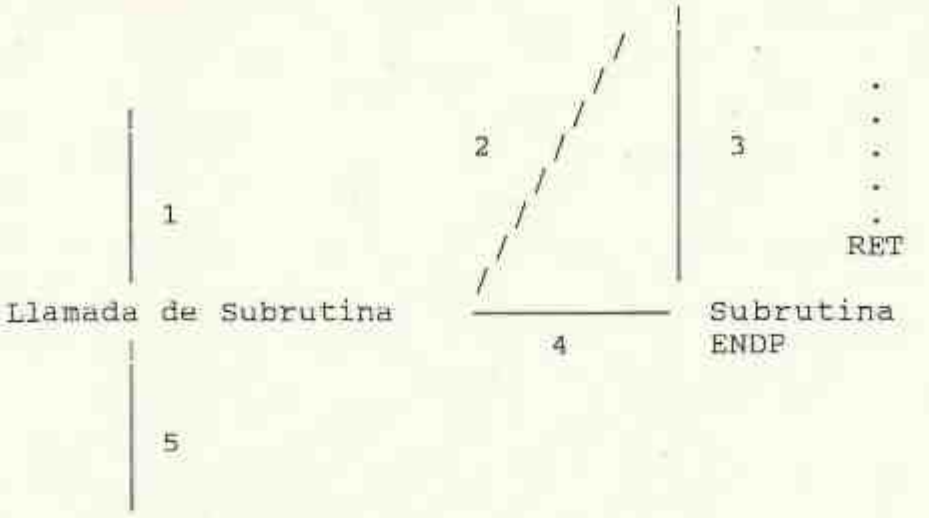
6.3 SUBRUTINAS.

Frecuentemente cuando escribimos un programa, encontramos que una secuencia particular de instrucciones es necesaria en varios puntos diferentes; lo cual sería redundante, al insertar estas instrucciones en cada punto del programa donde sean necesarias, esto se puede resumir en lo que llamamos subrutinas.

Una subrutina es un conjunto de instrucciones, normalmente escrita para realizar una función particular.

Esta subrutina es llamada desde cada punto del programa donde se necesita. El control es transferido a la subrutina cuando es llamada, y las instrucciones dentro de ésta son ejecutadas.

Cuando la subrutina es completada, retorna el control hacia la siguiente instrucción desde la cual fue llamada. Este proceso se ilustra en la figura 6.5.



1. Programa Principal.
2. Llamada de subrutina.
3. Ejecutar Subrutina.
4. Retorno desde subrutina.
5. Continuar el programa Principal.

Fig. 6.5 Llamada de Subrutina

El microprocesador 8088 realiza este mecanismo de subrutinas con las instrucciones CALL y RET. Ambas instrucciones (CALL y RET) pueden ser intrasegmento ó intersegmento. En la Fig. 6.6 se ilustra el mecanismo de una llamada Intrasegmentaria.

Cuando una subrutina es llamada (call subrutina), el registrador, IP, es almacenado en la pila, (PUSH IP) y luego toma la dirección de inicio de la subrutina que ha sido llamada.

Cuando la subrutina es ejecutada, el registrador, IP, es desempilado (POP IP); es de hacer notar que el empilamiento y desempilamiento del registrador, IP, se hace en forma automática por el microprocesador, de esta manera el microprocesador sabe en donde retornar cuando una subrutina ha sido ejecutada.

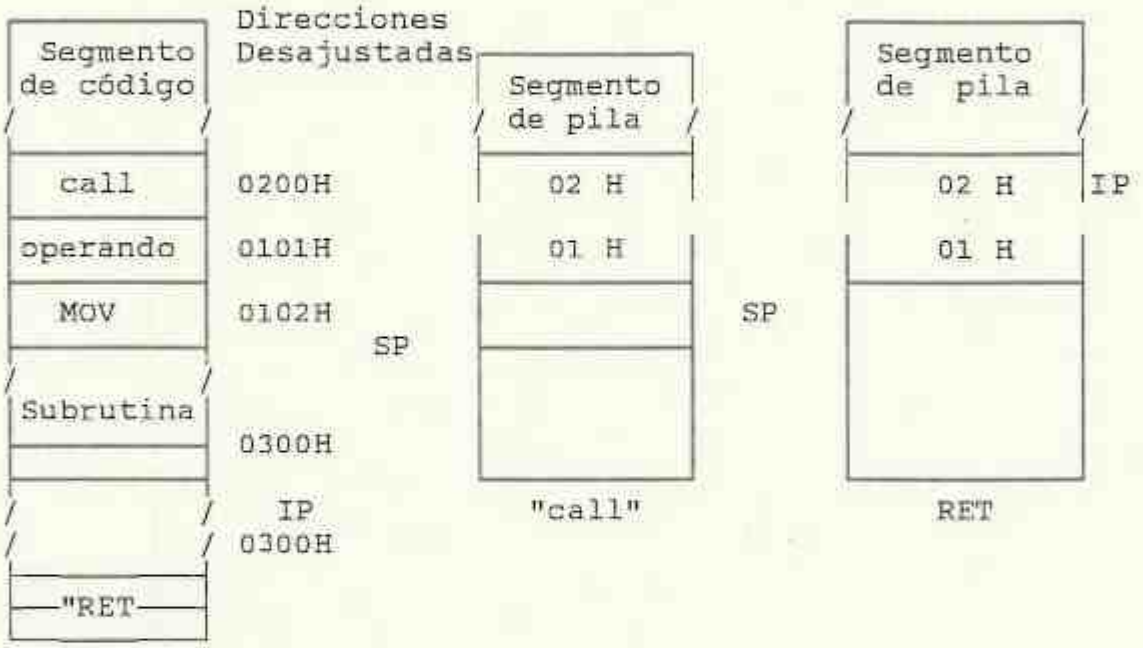


Fig. 6.6 Llamada de subrutina Intersegmentaria

```
PILA SEGMENT PARA STACK
        DB 256 DUP(0)
PILA ENDS
CODIGO SEGMENT PARA PUBLIC
  PRINCIPAL PROC FAR
    ASSUME CS:CODIGO
        PUSH DS
        MOV AX,0
        PUSH AX
        MOV DH,5 ; Fila
        MOV DL,5 ; Columna
        MOV AH,2
        MOV BH,0
         INT 10H
      MOV AL,'P'
      CALL ESCRIBE
      MOV AL,'A'
      CALL ESCRIBE
      MOV AL,'T'
      CALL ESCRIBE
      MOV AL,'Y'
      CALL ESCRIBE
      RET
    PRINCIPAL ENDP
    ESCRIBE PROC NEAR
        MOV AH,14 ; Escribe
        MOV BH,0  ; el caracter
        INT 10H   ;
      RET
 ESCRIBE ENDP
 CODIGO ENDS
END PRINCIPAL
```

Ejemplo 6.1 Este programa escribe el nombre 'PATY', a partir de la posición (5,5) usando subrutinas.

```
 PILA SEGMENT PARA STACK
        DB 256 DUP(0)
 PILA ENDS
;
CODIGO SEGMENT PARA PUBLIC
   PRINCIPAL PROC FAR
     ASSUME CS:CODIGO
         PUSH DS
         MOV AX,0
         PUSH AX
;
         MOV DL,35 ; Columna
         CALL CURSOR
       MOV AL,'P'
       CALL ESCRIBE
         MOV DL, 36
         CALL CURSOR
       MOV AL,'A'
       CALL ESCRIBE
         MOV DL,37
         CALL CURSOR
       MOV AL,'T'
       CALL ESCRIBE
         MOV DL,38
         CALL CURSOR
       MOV AL,'Y'
       CALL ESCRIBE
       RET
     PRINCIPAL ENDP
;
```

```
        CURSOR PROC NEAR
         MOV DH,16
         MOV AH,2
          MOV BH,0
          INT 10H
          RET
        CURSOR ENDP
;
      ESCRIBE PROC NEAR
          MOV AH,14 ; Escribe
          MOV BH,0  ; el caracter
          INT 10H   ;
        RET
  ESCRIBE ENDP
  CODIGO ENDS
 END PRINCIPAL
```

Ejemplo 6.2 Este programa escribe el nombre 'PATY' verticalmente a partir de la posición (16,35) usando subrutinas.

CAPITULO 7.

SALTOS Y LAZOS

Hasta esta parte los programas han sido realizados en forma continua, una por una sus instrucciones. Pero el programador puede cambiar el curso de un programa por medio de los saltos condicionales o incondicionales.

7.1 INSTRUCCIONES DE SALTOS CONDICIONALES PARA EL 8088

OPERACION	INSTRUCCION	RESULTADO
SALTA SI ES IGUAL A CERO	JE/JZ	Z = 1
SALTA SI ES DIFERENTE DE CERO	JNE/JNZ	Z = 0
SALTA SI NO ES NEGATIVO	JNS	S = 0
SALTA SI ES NEGATIVO	JS	S = 1
SALTA SI NO TIENE CARRY	JNB/JAE/JNC	C = 0
SALTA SI TIENE CARRY	JB/JNAE/JC	C = 1
SALTA SI SUCEDE OVERFLOW	JO	O = 1
SALTA SI NO SUCEDE OVERFLOW	JNO	O = 0
SALTA SI HAY PARIDAD	JP/JPE	P = 1
SALTA SI NO HAY PARIDAD	JNP/JPO	P = 0

TABLA 7.1 Algunas Instrucciones de Salto Condicional

```
PILA SEGMENT PARA STACK
        DB 256 DUP(0)
 PILA ENDS
CODIGO SEGMENT PARA PUBLIC
   PRINCIPAL PROC FAR
     ASSUME CS:CODIGO
         PUSH DS
         MOV AX,0
         PUSH AX
;
          MOV AL, 'A'; guarda el caracter 'a' en AL
          MOV DX, 10 ; DX toma el valor de 10
LAZO:     MOV AH,14
          MOV BH,0 ; despliega el caracter guardado en AL
          INT 10H
          DEC DX; decrementa el valor de DX en 1
          JNE LAZO; si no es igual a cero salta al lazo
          RET
     PRINCIPAL ENDP
     CODIGO ENDS
 END PRINCIPAL
```

Ejemplo 7.1 Este programa escribe 10 veces la letra 'a' controlado por medio de instrucciones de saltos condicionales

Una instrucción de salto condicional causará un salto si la condición es cierta. Por ejemplo, la instrucción de salto condicional JE (salta si es igual) causará un salto en el programa si el flag Z en el registrador de estado es uno. Este será el caso si el resultado de la instrucción previa produce un resultado igual a cero. Si la condición es falsa entonces la instrucción siguiente a la de salto condicional es ejecutada. Esto es ilustrado en la figura siguiente:

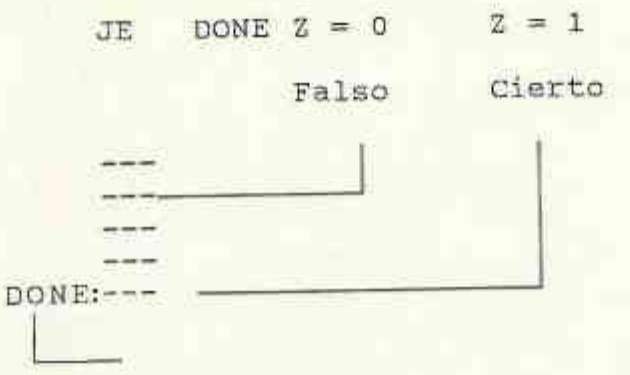


Fig. 7.1 Ilustración de Salto condicional

7.2 INSTRUCCIONES DE SALTO INCONDICIONAL PARA EL 8088

Las instrucciones de la tabla anterior son instrucciones de salto condicional que pueden o no causar un salto dependiendo del valor del flag de status. Algunas veces queremos realizar un salto sin necesidad de que cumpla una condición.

```
--------

SIGA: -----

      -----

      JMP SIGA
```

Fig. 7.2 Ilustración de Salto Incondicional

OPERACION	INSTRUCCION	OPERANDO
SALTO CORTO	JMP	8 - Bit de desplazamiento
SALTO LARGO	JMP	16-Bit de desplazamiento
SALTO ABSOLUTO	JMP	Dirección de Segmento: Dirección Desajustada

TABLA 7.2 Algunas Instrucciones de Saltos Incondicionales

```
 PILA SEGMENT PARA STACK
         DB 256 DUP(0)
 PILA ENDS
;
CODIGO SEGMENT PARA PUBLIC
   PRINCIPAL PROC FAR
     ASSUME CS:CODIGO
          PUSH DS
          MOV AX,0
          PUSH AX
           MOV AL, 'A'; guarda el caracter 'a' en AL
;
LAZO:      MOV AH,14; despliega el caracter guardado en AL
           MOV BH,0
           INT 10H
           JMP LAZO; salta a la etiqueta lazo
           RET
```

```
    PRINCIPAL ENDP
    CODIGO ENDS
END PRINCIPAL
```

Ejemplo 7.2 Este programa escribe la letra 'a' infinitas veces usando la instrucción de salto incondicional

7.3 LAZOS

La instrucción LOOP tiene con finalidad controlar el flujo de ejecución del programa hacia determinada etiqueta, cada vez que se ejecute un LOOP, el contenido del registro CX se decrementa en uno, cuando CX = 0, el LOOP deja de ejecutarse y el programa continúa con la instrucción que siga a LOOP. Como ejemplo, considérense las siguientes instrucciones:

```
        MOV CX,10
        LABEL1:----
                ---
                LOOP LABEL1
                MOV DX,10
                ----
```

Fig. 7.3 Ejemplo de un Lazo

En este caso, el contador del LOOP se establece en 10. Por tanto, las instrucciones que se encuentren entre LABEL1: y LOOP LABEL1 se ejecuta 10 veces.

```
 PILA SEGMENT PARA STACK
         DB 256 DUP(0)
 PILA ENDS
;
 CODIGO SEGMENT PARA PUBLIC
   PRINCIPAL PROC FAR
     ASSUME CS:CODIGO
          PUSH DS
          MOV AX,0
          PUSH AX
          MOV DH,15 ; Fila
          MOV DL,0  ; Columna
          MOV AH,2
          MOV BH,0
           INT 10H
        MOV CX,79
        MOV AL,'$'
  LAZO:   MOV AH,14 ; Escribe
          MOV BH,0  ; el caracter
          INT 10H   ;
        LOOP LAZO
        RET
  PRINCIPAL ENDP
   CODIGO ENDS
 END PRINCIPAL
```

Ejemplo 7.1 Este programa escribe el caracter'$' en toda la fila 15 de la pantalla utilizando un lazo.

```
 PILA SEGMENT PARA STACK
        DB 256 DUP(0)
 PILA ENDS
;
 CODIGO SEGMENT PARA PUBLIC
   PRINCIPAL PROC FAR
     ASSUME CS:CODIGO
         PUSH DS
         MOV AX,0
         PUSH AX
        MOV DH,10 ; Fila
         MOV AL,'0'
         MOV CX,10
LAZO:  CALL CURSOR
         CALL ESCRIBE
         INC AL
         INC DH
       LOOP LAZO
     RET
PRINCIPAL ENDP
     CURSOR PROC NEAR
          MOV DL,20
          MOV AH,2
          MOV BH,0
          INT 10H
          RET
     CURSOR ENDP
     ESCRIBE PROC NEAR
          MOV AH,14
          MOV BH,0
          INT 10H
          RET
     ESCRIBE ENDP
   CODIGO ENDS
 END PRINCIPAL
```

Ejemplo 7.1 Este programa escribe los números del 0 al 9 en forma vertical utilizando un lazo subrutinas.

CAPITULO 8.

PROGRAMANDO CONTROLADORES DE PERIFERICOS

En esta sección estudiaremos la progrmación de controladores de periféricos y para ello hemos seleccionado el controlador del periférico Bocina.

Las computadoras personales (IBM) pueden producir sonidos usando la bocina. Estos sonidos pueden tener diferente duración y tono.

A continuación haremos una descripción de como se generan los sonidos en la bocina.

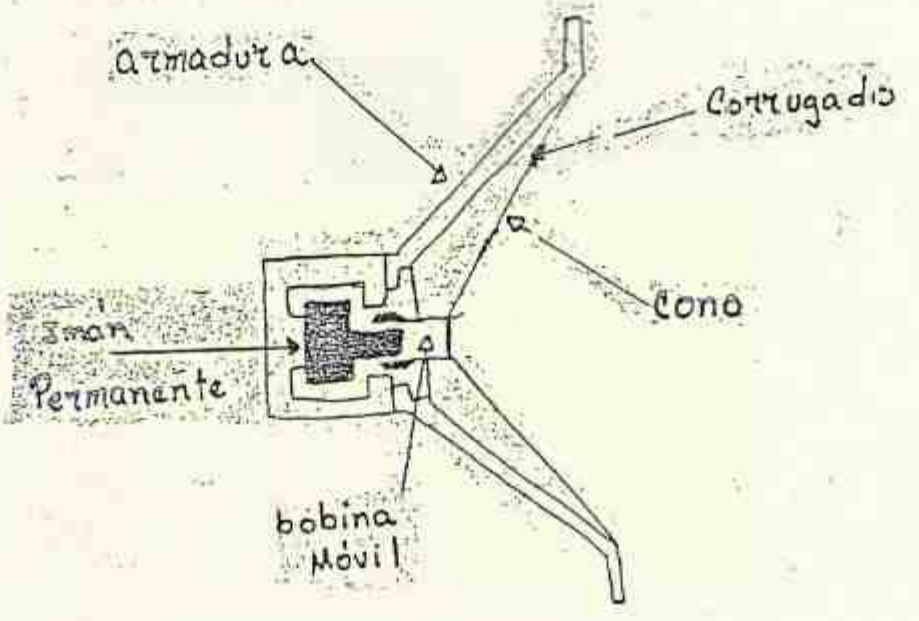
8.1 COMO SE PRODUCEN LOS SONIDOS EN LA BOCINA

La figura 8.1 ilustra la construcción típica de una bocina(el armazón es la pieza principal de soporte). El cono (o también conocido como diafragma) está hecho de un papel especial y los extremos están cimentados al armazón. Pegados al círculo interno del cono hay una forma cilíndrica que sostiene la bobina móvil.

Esta bobina penetra en un pequeño espacio donde se halla colocado permanentemente el imán. Hay una interacción entre el campo magnético del imán y el de la bobina móvil, que produce un movimiento hacia adentro y hacia afuera en el cono de la bocina, esto ocurre cuando diferentes cargas de energía llegan a la bobina. Hay que tener claro que

si éstas son constantes no producen dicho movimiento, ni el sonido, ya que el cono no vibra y por lo tanto no mueve el viento que es el encargado de llevar las ondas sonoras hasta nuestros oídos.

El cono de papel sigue movimientos de la bobina produciendo el sonido. Este está corrugado para que tenga la suficiente flexibilidad que le permita vibrar sin dificultad.



8.2 IMPORTANCIA DE LOS PULSOS DE RELOJ EN UNA PC.

En primer lugar debemos tener claro que todas las computadoras traen un reloj incorporado, dado que el sistema necesita estar sincronizado; es decir, tener un ordenamiento en el tiempo de ejecución de cada una de las actividades desarrolladas por el computador. Entonces cualquier sistema basado en el microprocesador 8088 requiere una lógica adicional encargada de generar las señales (pulsos de reloj) de sincronización para todo el sistema. Los pulsos de reloj determinan la velocidad del funcionamiento del sistema.

8.3 INTERFACE DE LA BOCINA DE LA PC IBM

Dos chips programables de interface, uno el Programable Peripheral Interface (PPI) 8255A-5 y el Programable Interval Timer (PIT) 8253-5 son involucrados en la interface de la bocina como muestra la figura 8.2. El sonido básico(señal enviada al amplificador de la bocina), es una onda cuadrada producida por el PIT. El período de la onda cuadrada(que es el tono del sonido), depende de uno de los valores de los dieciséis bits almacenados en el registrador de conteo del PIT. Este registrador de conteo (los 16 bits) tiene la dirección de E/S 0042H.

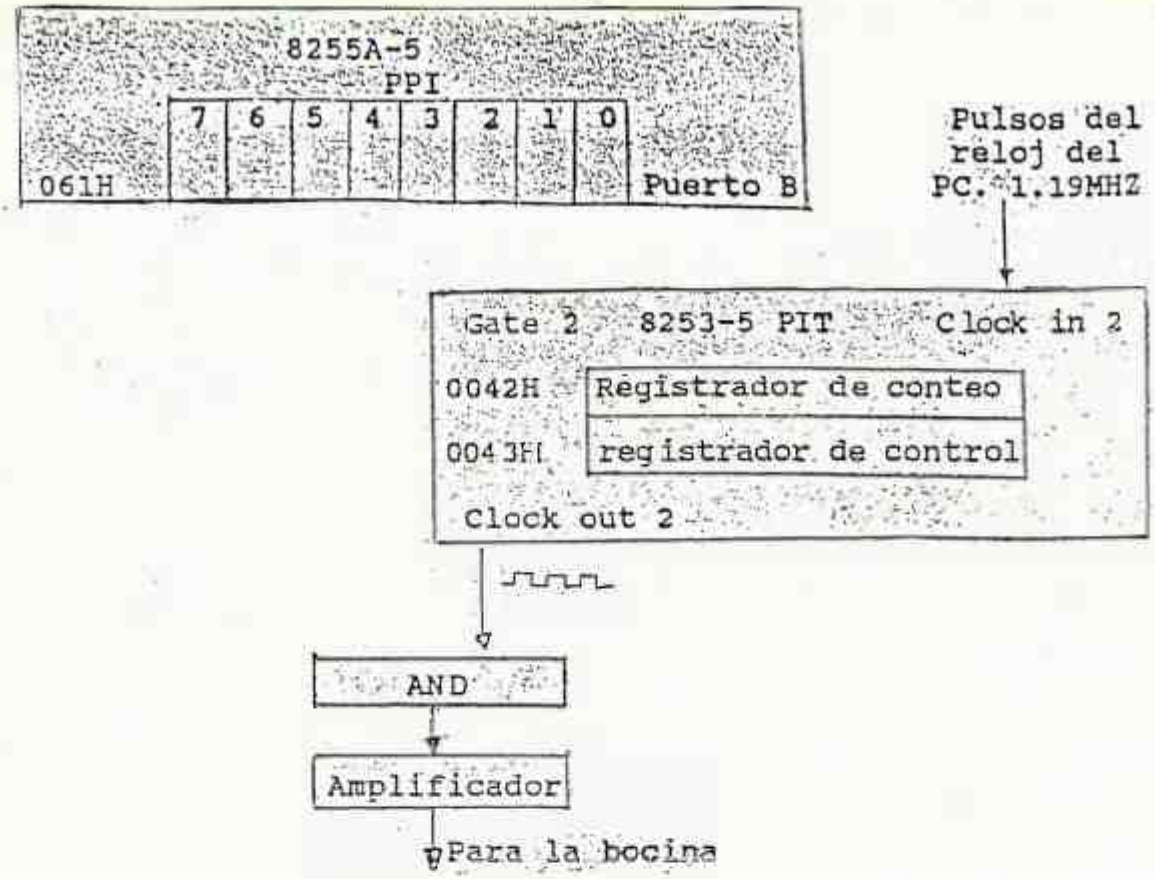


Fig. 8.2 Interface de la bocina de la PC IBM.

Para el interface de la bocina almacenaremos el valor B6H en el registrador de control. El significado de los bits en este valor están dados en la figura 8.3.

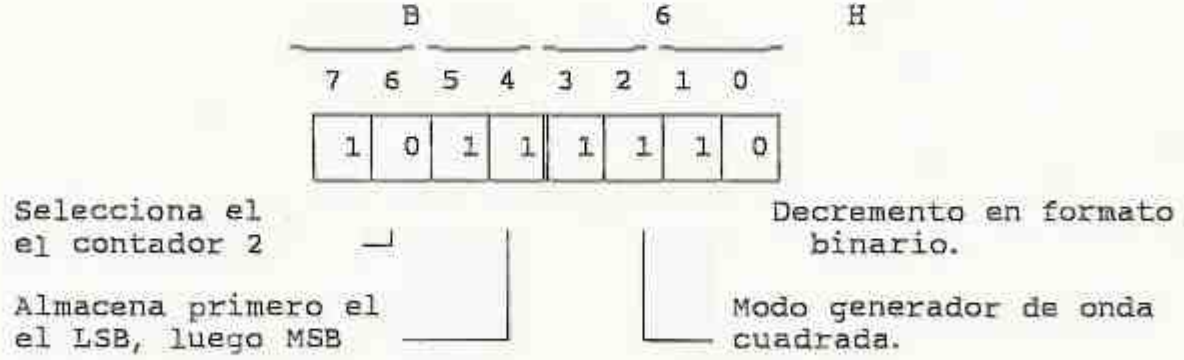


Fig. 8.3 Descripción de los valores en el registrador de control.

En general los bits 6 y 7 son usados para seleccionar cual canal está siendo programado, esto depende de los valores que en ellos se almacenen, el resto de los bits en este registrador definen como ese canal operará y como se comunicará con su registrador de conteo (que está en la dirección 0042H).

Dado que el registrador de conteo es de 16 bits de ancho y su puerto de acceso es solamente de 8 bits,se necesitan dos accesos de Entrada/Salida para leer o escribir el registrador de conteo, por eso es que en los bits 4 y 5 del registrador de control cargamos el valor de uno, que indica que primero se almacena el byte menos significativo (LSB) y luego el byte más significativo (MSB).

Esto hay que tenerlo en cuenta a la hora de escribir un programa; porque nosotros podemos leer el registrador de conteo usando dos instrucciones IN sucesivas, la primera retornará el byte menos significativo del registrador de conteo y la segunda retornará el byte más significante de dicho registrador.

Similarmente, podemos escribir el registrador de conteo (en el orden LSB-MSB) con dos instrucciones OUT sucesivas.

8.3.1 COMO ENCENDER O APAGAR LA BOCINA DEL COMPUTADOR.

Cuando nosotros queremos que el computador produzca un sonido primero debemos encender o activar la bocina, para esto los bit 0 y 1 del puerto B (con dirección de Entrada/Salida 0061H) deben tener el valor de 1, y si deseamos desactivarla, el valor de 0.

8.4 PROGRAMACIÓN EN LENGUAJE ENSAMBLADOR PARA GENERAR UN SONIDO.

A continuación se explica lo que hace y como funciona el programa dado en la ejemplo 8.1; este programa produce un tono cuya graduación (o timbre) depende del valor cargado en el registrador DX y la duración del valor dado en el registrador BX.

Para probar el sonido del programa, guardaremos el valor de graduación 0300H en el registrador DX y el valor de duración 000FH en el registrador BX.

```
PILA SEGMENT PARA STACK
DB 256 DUP(0)
PILA ENDS
    ;
CODIGO SEGMENT PARA PUBLIC
PRINCIPAL PROC FAR
     ASSUME CS:CODIGO ; Prólogo del programa.
     PUSH DS
     MOV AX,0
     PUSH AX
     ; Empilar los cuatro registradores.
     PUSH AX
     PUSH BX
     PUSH CX
     PUSH DX
     ;
     MOV DX, 0300H ;Graduación y
     MOV BX,000FH  ;duración del sonido
     ;
     MOV AL,0B6H   ;Se guarda 0B6H en AL
     OUT 43H,AL    ;Almacena 0B6H en el regiStrador de
                   ;control del PIT.
     MOV AL,DL     ;EScribe el byte menos significativo
     OUT 42H,AL    ;de la graduación en el regiStrador
                   ;de conteo del PIT.
     MOV AL,DH     ;EScribe el byte máS significativo
     OUT 42H,AL    ;de la graduación en el registrador
                   ;de conteo del PIT.
     IN AL,61H     ;Lee el puerto B del PPI, cuyo valor
                   ;Se guarda en Al.
     MOV AH,AL     ;Luego guarda en AH.
     OR AL,03H     ;Se colocan l'S en el bit 0 y 1 en el
     OUT 61H,AL    ;puerto B y con esto se activa la
                   ;bocina.
TN1: MOV CX,1118H  ;Lazo que se ejecuta en un tiempo de
TN2: LOOP TN2      ;1/16 de Segundo.
     DEC BX        ;Decrementa BX.
     JNE TN1       ;Si BX<>0 ir a TN1
     MOV AL,AH     ;Devuelve el valor original
     OUT 61H,AL    ;al puerto PB con lo cual Se
                   ;desactiva la bocina.
; Desempila los regiStradoreS
     POP DX
     POP CX
     POP BX
     POP AX
     RET
PRINCIPAL ENDP
CODIGO ENDS
END PRINCIPAL
```

Ejemplo 8.1 Programa para producir un Sonido.

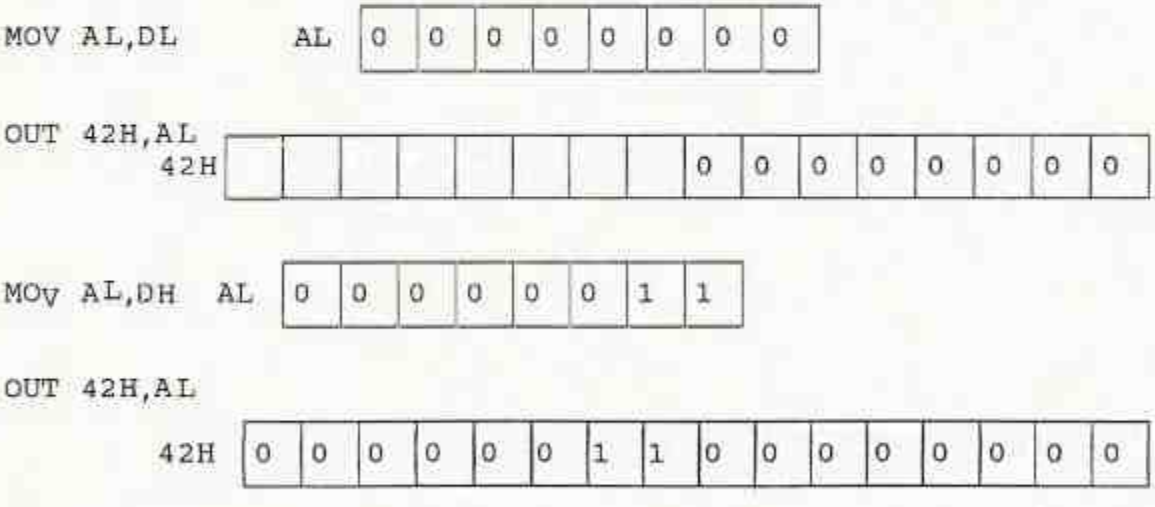
8.4.1 FUNCIONAMIENTO DEL PROGRAMA DEL EJEMPLO 8.1.

Comenzaremos a explicar como funciona dicho programa a partir de las instrucciones siguientes:

```
MOV DX,0300H
MOV BX,000FH
```

Con estas instrucciones DX toma el valor de 0300H y BX el de 000FH como se muestra a continuación:

Con la primera instrucción el valor B6H es almacenado en AL, luego en la segunda instrucción es transferido al registrador de control del PIT, que está en la dirección de memoria 43H, con esto estamos seleccionando un modo de funcionamiento particular.



En el primer par de instrucciones anteriores se escribe el byte menos significativo del registrador DX (timbre) en el registrador de conteo del PIT, el cual tiene dirección de memoria 42H. Luego con las dos instrucciones siguientes, se escribe el byte más significativo del registrador DX en el registrador de conteo.

Con la instrucción IN AL,61H se guarda en AL el contenido del puerto B del PPI cuya dirección es 0061H como se muestra a continuación:

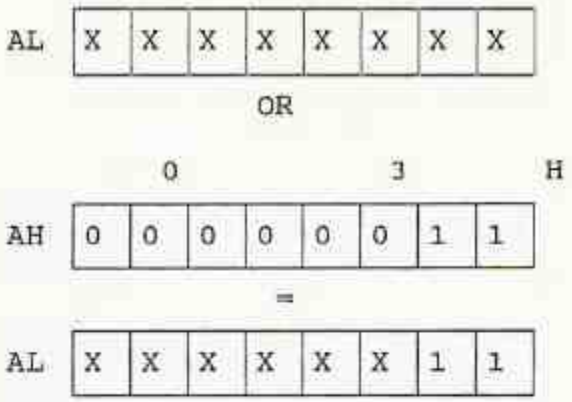
AL	X	X	X	X	X	X	X	X

Donde X representa el valor (0 ó 1) de cada uno de los bits leídos.

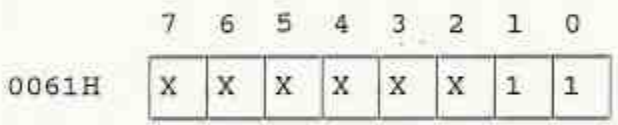
MOV AH,AL con esto el contenido de AL se copia en AH.

AL	X	X	X	X	X	X	X	X
AH	X	X	X	X	X	X	X	X

OR AL,03H esta instrucción efectúa el "ó lógico" guardando el resultado en AL, como se muestra en la figura siguiente:



OUT 61H,AL con esta instrucción escribimos el contenido de AL en el puerto B del PPI, cuya dirección de memoria es 61H.



Con esto estamos escribiendo el valor de 1 en los bits 0 y 1 del puerto B, que es lo necesario para activar la bocina.

```
TN1: MOV CX,1118H
TN2: LOOP TN2
```

Estas dos instrucciones es un lazo que se ejecuta en un tiempo de aproximadamente 1/16 de segundo, el cual es el tiempo en que la bocina estará encendida.

DEC BX esta instrucción decrementa el registrador BX en 1. JNE TN1 verifica, si BX < > 0 entonces pasa el control de ejecución a la instrucción que tiene la etiqueta TN1.

MOV AL,AH transfiere el valor de AH a AL

AL	X	X	X	X	X	X	X	X

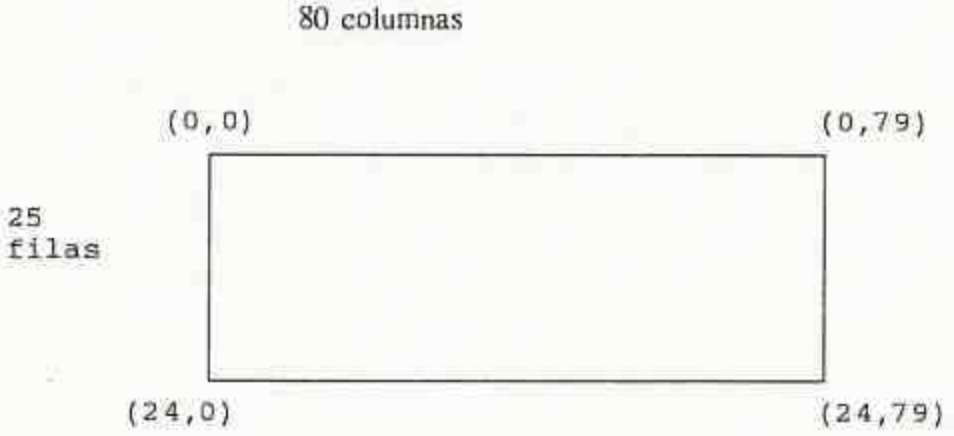
OUT 61H,AL escribe el contenido de AL en el puerto B del PPI, que es el valor que tenía antes de ejecutar el programa.

CAPITULO 9.

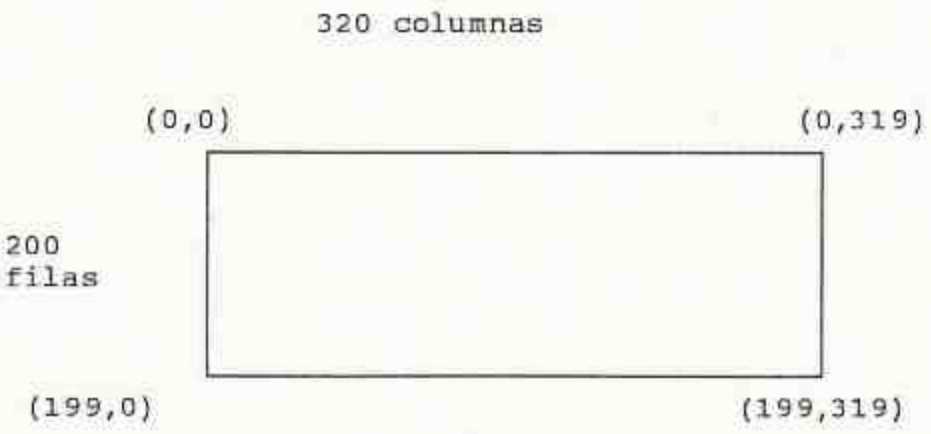
GRAFICOS

Los monitores poseen una resolución para desplegar texto, dicha resolución es de 80x25 (80 filas y 25 columnas) Pero cuando queremos graficar la resolución debe cambiar, puede ser media (320x200) o alta (640x200).

a)Resolución alfanumérica 80 x 25



b) Resolución Media 320 x 200



c) Resolución Alta 640 x 200

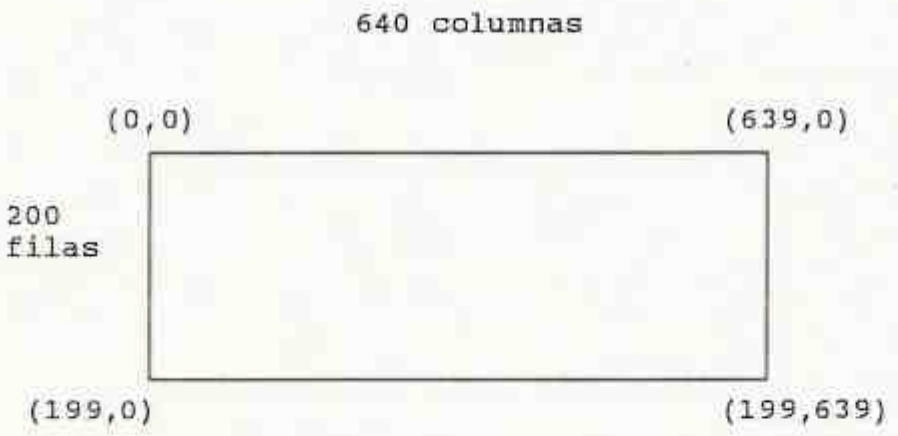


Fig. 9.1 Resoluciones que poseen los monitores

9.1 INT 10H EN EL MODO GRAFICO

Al entrar al estudio de gráficos a la intersección de una fila y columna le llamaremos pixel.

Para utilizar el modo gráfico usaremos la interrupción 10H de manera similar al modo texto. La siguiente tabla muestra algunas funciones de la INT 10H.

AH	PROPOSITO	DESCRIPCION
0	MODO	El registro AL contiene el modo de video: AL=0 - 45x25 pixels B/N AL=1 - 45x25 pixels Color AL=2 - 80x25 pixels B/N AL=3 - 80x25 pixels Color AL=4 - 320x200 pixels Color AL=5 - 320x200 pixels B/N AL=6 - 640x200 pixels B/N
12	ESCRIBE UN PUNTO	DX= número de renglón, Cx = número de columna, AL= Color (para monitores de alta resolución AL varía la intensidad)
13	LECTURA DE UN PUNTO	DX= número de renglón, CX = número de columna, AL= punto leído

TABLA 9.1 Funciones de la INT 10H para Gráficos

9.2 PROGRAMAS EJEMPLOS

A continuación se presentan programas ejemplos sobre el uso del modo gráfico.

Ejemplo 9.1 El siguiente programa dibuja un pixel en la posición (15,25)

```
PILA SEGMENT PARA STACK
     DB 256 DUP (0)
PILA ENDS
;
CODIGO SEGMENT PARA PUBLIC
PRINCIPAL PROC FAR
     ASSUME CS: CODIGO
     PUSH DS
     MOV AX,0
     PUSH AX
     ;
     MOV DX,0
     MOV BH,0  ; Posicionan el cursos en (0,0)
     MOV AH,2
     INT 10H
     ;
     MOV CX,2000
     MOV AH,10
     MOV BH,0
     MOV AL,' ';Borran la pantalla
     INT 10H
     ;
     MOV AH,0
     MOV AL,5  ; Establecen el modo gráfico 320 x 200
     INT 10H
     ;
     MOV CX,15  ; Posición inicial de la línea
     MOV DX,25
       MOV AH,12
     MOV AL,1  ; Iluminan un pixel en la fila CX
     INT 10H   ; Columna DX
     MOV AH,0  ; Espera que una tecla se presionada
     INT 16H
     ;
     MOV AH,0
     MOV AL,2  ; Establece el modo alfanumérico
     INT 10H
     ;
     RET
PRINCIPAL ENDP
CODIGO ENDS
     END PRINCIPAL
```

Ejemplo 9.2 El siguiente programa dibuja una línea inclinada desde la posición (5,5) hasta la posición (15,15)

```
PILA SEGMENT PARA STACK
     DB 256 DUP (0)
PILA ENDS
;
CODIGO SEGMENT PARA PUBLIC
PRINCIPAL PROC FAR
     ASSUME CS: CODIGO
     PUSH DS
     MOV AX,0
     PUSH AX
     ;
     MOV DX,0
     MOV BH,0  ; Posicionan el cursor en (0,0)
     MOV AH,2
     INT 10H
     ;
     MOV CX,2000
     MOV AH,10
     MOV BH,0
     MOV AL,' ';Borran la pantalla
     INT 10H
     ;
     MOV AH,0
     MOV AL,5  ; Establecen el modo gráfico 320 x 200
     INT 10H
     ;
     MOV CX,5  ; Posición inicial de la línea
     MOV DX,5
LAZO 1:MOV AH,12
     MOV AL,1  ; Iluminan un pixel en la fila CX
     INT 10H   ; Columna DX
     INC CX
     INC DX
     CMP CX,15
     JNE LAZO1
     ;
     MOV AH,0  ; Espera que una tecla se presionada
     INT 16H
     ;
     MOV AH,0
     MOV AL,2  ; Establece el modo alfanumérico
     INT 10H
     ;
     RET
PRINCIPAL ENDP
CODIGO ENDS
     END PRINCIPAL
```

Ejemplo 9.3: Este programa dibuja una línea inclinada desde la posición (0,0) y al presionar una tecla borra la mitad superior de la línea.

```
PILA SEGMENT PARA STACK
     DB 256 DUP (0)
PILA ENDS
;
CODIGO SEGMENT PARA PUBLIC
PRINCIPAL PROC FAR
     ASSUME CS: CODIGO
     PUSH DS
     MOV AX,0
     PUSH AX
     ;
     MOV DX,0
     MOV BH,0  ; Posicionan el cursos en (0,0)
     MOV AH,2
     INT 10H
     ;
     MOV CX,2000
     MOV AH,10
     MOV BH,0
     MOV AL,' ';Borran la pantalla
     INT 10H
     ;
     MOV AH,0
     MOV AL,5  ; Establecen el modo gráfico 320 x 200
     INT 10H
     ;
     MOV CX,5  ; Posición inicial de la línea
     MOV DX,5

LAZO1: MOV AH,12
     MOV AL,1  ; Ilumina un pixel en la fila CX
     INT 10H   ; Columna DX
     INC CX
     INC DX
     CMP CX,100
     JNE LAZO1
     ;
     MOV AH,0  ; Espera que una tecla se presionada
     INT 16H
     ;
     MOV CX,5  ; Posición inicial de la línea
     MOV DX,5
LAZO2: MOV AH,12
     MOV AL,0  ; Apaga un pixel en la fila CX
     INT 10H   ; Columna DX
     INC CX
```

```
        INC DX
        CMP CX,50
        JNE LAZO2
        ;
        MOV AH,0   ; Espera que una tecla sea presionada
        INT 16H
        ;
        MOV AH,0
        MOV AL,2   ; Establece el modo alfanumérico
        INT 10H
        ;
        RET
PRINCIPAL ENDP
CODIGO ENDS
        END PRINCIPAL
```

CAPITULO 10

DISCO E/S

Las disketteras usadas en la IBM PC pueden usar diskettes de:

- Un solo lado
- Doble lado
- Doble densidad

Un disco de un solo lado y doble densidad tiene 40 pistas circulares cada uno con 8 ó 9 sectores.

Como se muestra en la figura 10.1 cada sector por pista contiene 512 bytes; de la misma manera, cada pista contiene 4,096 (DOS 1.1) ó 4,608 (DOS 2.0) bytes y cada disco contiene 163,840 (DOS 1.1) ó 184,320 (DOS 2.0) bytes.

Un disco de doble lado tiene 2 cabezas de lectura separados y contiene un total de 327,680 (DOS 1.1) ó 368,640 (DOS 2.0) bytes. Las disketteras de doble lado llegan a ser standard en la IBM PC. Las disketteras de doble lado pueden leer discos formateados en diskettes de un solo lado, pero no viceversa.

También el DOS 2.0 puede leer archivos con DOS 1.1 pero no viceversa.

Puedes usar el comando FORMAT mostrado en la figura 10.2 para formatear un disco.

Hay 2 maneras diferentes de accesar datos en un disco. La primera es usar la rutina incorporada al ROM BIOS para leer y escribir en un pista y sector específico. El segundo es para usar comandos del DOS I/O para accesar archivos específicos en un disco. Consideraremos ambos métodos de accesa al disco.

Fig. 10.1 Sectores de un disco

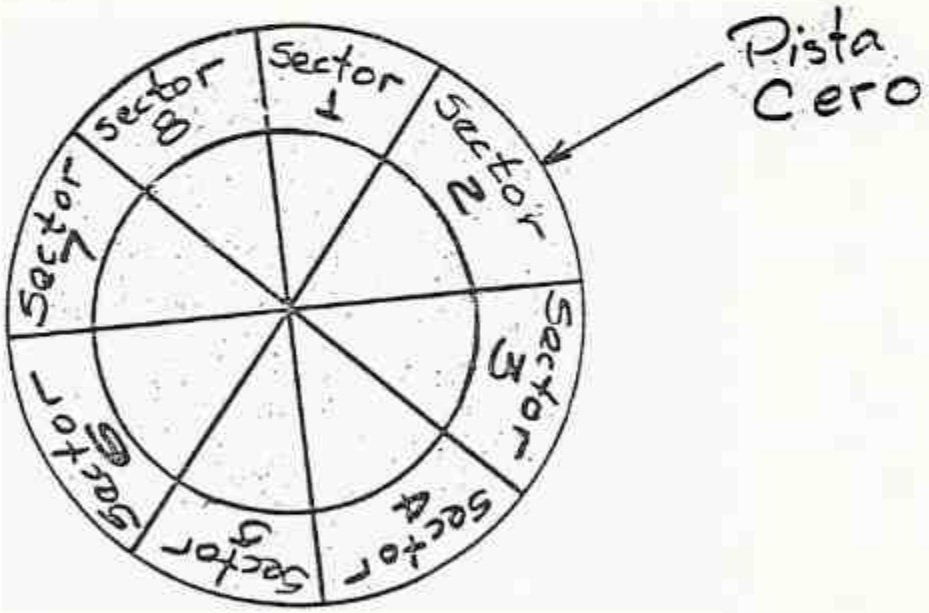


Fig. 10.2 Comando FORMAT usado para formatear discos nuevos

FORMAT [D:][/S][/1][/8][/V][/B]

D: Drive (Ej. A:)

/S Sistema

/1 Formatear solo un lado

/8 Formatear 8 sectores por pista

/V Prompt para etiqueta de volumen

/B Formatear 8 sectores por pista y deja espacio para los archivos del sistema (usando comando SYS del DOS)

Ej. FORMAT B:/S/V Formateará 9 sectores por pista (DOS 2.0), copia archivos del sistema para disco nuevo, y pregunta si deseas colocar un nombre al volumen.

Tabla 10.1 Algunas opciones de disco E/S disponibles con la INT 13H.

AH	FUNCION
0	Resetea el sistema de disco
1	Lee el byte de estado del disco SALIDA: AL = Byte de estado del disco

AH	FUNCION
2	Lee el sector dentro de la memoria ENTRADA: AL = Número de sectores a leer (máx = 9) DL = Número de drive (0-3) DH = Número de cabezal (0-1) CH = Número de Pista (0 - 39) CL = Número de sector (1-9) ES:BX = Direcciones de Buffer SALIDA: AL = Número de sectores actualmente leídos AH = Byte de estado del disco CARRY = 0 Operación de éxito CARRY = 1 Operación de fracaso
3	Escribe sectores de la memoria ENTRADA: AL = Número de sectores a escribir (máx =9) DL = Drive número (0-3) DH = Cabezal número (0-1) CH = Número de pista (0-39) CL = Número sector (1-9) ES:BX = Dirección del buffer SALIDA: AL = Número de sectores actualmente escritos AH = Byte de estado del disco CAREY = 0 Operación de éxito CARRY = 1 Operación de fracaso

10.1 RUTINAS BIOS DE DISCO

La INT 13H puede ser usada para leer y escribir sectores específicos en un disco usando las opciones mostradas en la tabla 10.1, siguiendo una lectura u operación escrita, el byte de estado del disco estará en el registrador AH. El valor de este byte indicará posibles errores como muestra la tabla 10.2.

VALOR	ESTADO
00H	No hay error
01H	Mal comando
02H	Marca de dirección no encontrada
03H	Disco protegido contra escritura
04H	Sector no encontrado
08H	DMA sobrepasado
09H	DMA error
20H	Controlador de error de disco
40H	Busca error
80H	Error fuera de tiempo

Tabla. 10.2 Posibles valores del byte de estado del disco.

10.2 PROGRAMAS EJEMPLOS DE MANIPULACION DE DISCO

A continuación se presentan algunos programas ejemplos sobre el uso de disco utilizando las subrutinas de disco. El programa del ejemplo 10.2 selecciona la unidad de disco A.

Suponiendo que dicho programa a sido grabado con el nombre de DISC102.EXE, al ejecutarlo desde la unidad de disco C, su resultado se muestra en la figura 10.1.

```
; ESTE PROGRAMA SELECCIONA EL DRIVE A
pila segment para stack
     db 256 dup(0)
pila ends
codigo segment para public
principal proc far
          assume cs:codigo
          push ds
          mov ax,0
          push ax
          ;
          mov dl,0        ; selecciona el drive A
          mov ah,0EH
          int 21h
          ret
principal endp
codigo ends
end principal
```

Ejemplo 10.1 Este programa selecciona el drive "A"

```
C>DISC101

A>_
```

FIG. 10.1 Salida del programa del ejemplo 10.1

```
; este programa escribe, lee y despliega una letra 'l'
pila segment para stack
   db 256 dup (0)
pila ends
;
datos segment para public
  letras db 'l'
  salida d b ?
datos ends
;
codigo segment para public
principal proc far
    assume cs:codigo
    push ds
    mov ax,0
    push ax
    mov ax, datos
    mov ds,ax
    assume ds:datos
    mov es,ax
    assume es:datos
    ;
    mov bx,offset letras      ; BX toma la dirección
                              ; desajustada
                              ; de la variable letras
    mov dh,0                  ; numero de cabezal
    mov dl,0                  ; numero de drive (A)
    mov ch,20                 ; numero de pista
    mov cl,4                  ; numero de sector
    mov al,1                  ; numero de sectores a escribir
    mov ah,3                  ; escribe el sector
    int 13h
    mov bx,offset salida      ; BX toma la dirección
                              ; desajustada
                              ; de la variable salida
    mov dh,0                  ; numero de cabezal
    mov dl,0                  ; numero de drive
    mov ch,20                 ; numero de pista
    mov cl,4                  ; numero de sector
    mov al,1                  ; numero de sectores a leer
    mov ah,2                  ; leer el sector
    int 13h
    mov bx,offset salida      ; BX toma la dirección
                              ; desajustada
                              ; de la variable salida
    mov al,[bx]               ; AL toma el contenido de la
                              ; dirección que contiene BX
    mov ah,14                 ; instrucciones para
    mov bh,0                  ; escribir caracteres en
    int 10h                   ; pantalla
    ret
```

```
    principal endp
 codigo ends
end principal
```

Ejemplo10.2: Programa que escribe, lee y despliega una letra "l"

El programa del ejemplo 10.2, le asigna la letra 'l' a la variable 'letras' y luego el contenido de dicha variable es escrito en el disco de la unidad A (DL=0), en lado 0 (DH=0), pista 20 (CH=20) y sector 4 (CL=4). Luego, dicha letra es leída de disco de la misma posición del disco y almacenada en la variable 'salida'. Después, es desplegada en pantalla como se muestra en la figura 10.2 suponiendo que dicho programa ha sido grabado con el nombre de DISC102.EXE.

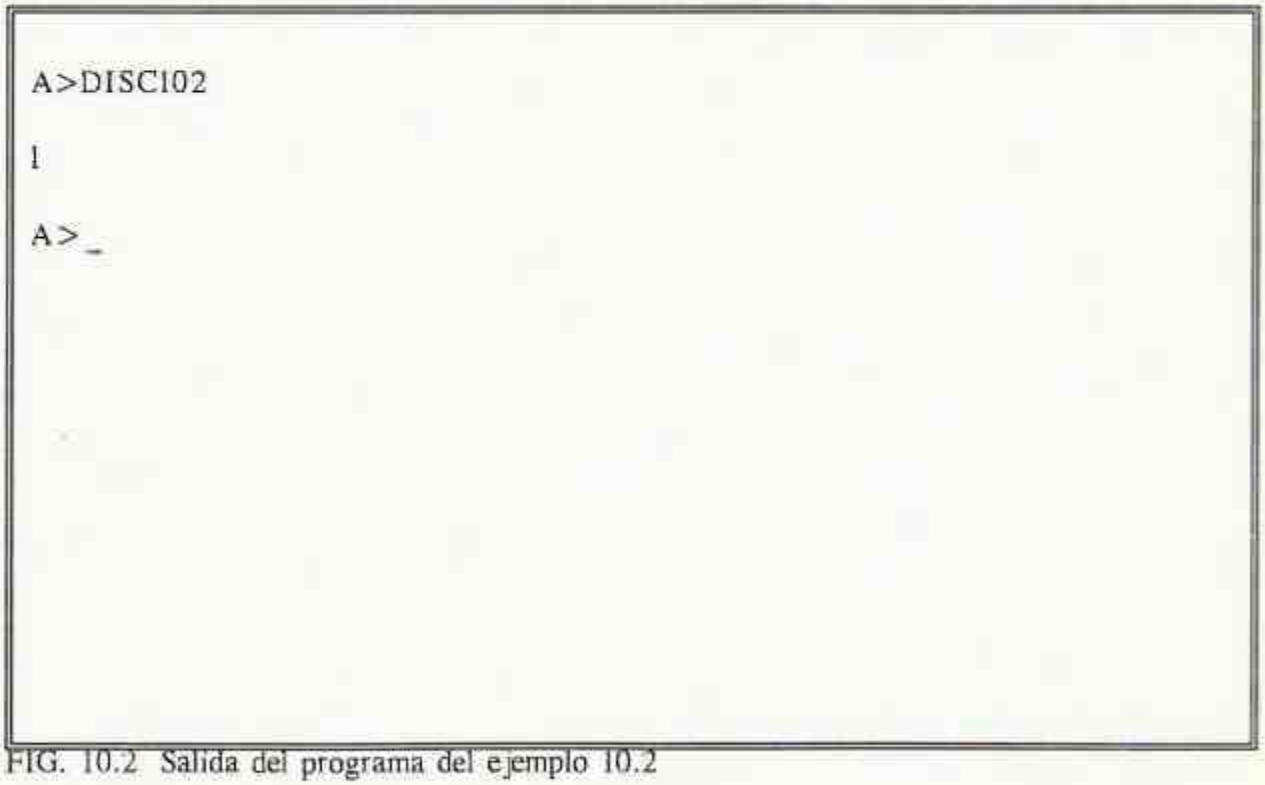


FIG. 10.2 Salida del programa del ejemplo 10.2

```
; este programa escribe, lee y despliega una letra 'j'
pila segment para stack
   db 256 dup (0)
pila ends
datos segment para public
  letras db 'j'
  salida d b ?
datos ends
codigo segment para public
principal proc far
    assume cs:codigo
    push ds
    mov ax,0
    push ax
    mov ax, datos
    mov ds,ax
    assume ds:datos
    mov es,ax
    assume es:datos
    mov bx,offset letras      ; BX toma la dirección
                              ; desajustada de
                              ; la variable letras
    mov dh,0                  ; numero de cabezal
    mov dl,0                  ; numero de drive
    mov ch,20                 ; numero de pista
    mov cl,4                  ; numero de sector
    mov al,1                  ; numero de sectores a escribir
    mov ah,3                  ; escribe el sector
    int 13h
    ;
    mov bx,offset salida      ; BX toma la dirección
                              ; desajustada de
                              ; la variable salida
    mov dh,0                  ; numero de cabezal
    mov dl,0                  ; numero de drive
    mov ch,20                 ; numero de pista
    mov cl,4                  ; numero de sector
    mov al,1                  ; numero de sectores a leer
    mov ah,2                  ; leer el sector
    int 13h
    mov bx,offset salida      ; BX toma la dirección
                              ; desajustada de
                              ; la variable salida
    mov al,[bx]               ; AL toma el contenido de la
                              ; dirección
                              ; que contiene BX
    mov ah,14
    mov bh,0
    int 10h
    ret
```

```
    principal endp
 codigo ends
end principal
```

```
Ejemplo 10.3    Este programa escribe, lee y despliega en
                pantalla el caracter "j"
```

En la figura 10.3 se muestra la salida del programa del ejemplo 103, suponiendo que ha sido grabado con el nombre DISC103.EXE, dicho programa es similar que el del ejemplo 10.2 con la variante que lo hace para letra 'j'.

```
A>DISC103

j

A>_
```

FIG. 10.3 Salida del programa del ejemplo 10.3.

```
;
; este programa escribe, lee de disco y despliega el mensaje
; 'Leda y Jorge'
;
pila segment para stack
        db 256 dup (0)
pila ends
;
datos segment para public
      letras db 'Leda y Jorge'
      salida db 5 dup(?)
datos ends
;
codigo segment para public
principal proc far
          assume cs:codigo
           push ds
           mov ax,0
           push ax
;
           mov ax, datos
           mov ds,ax
           assume ds:datos
           mov es,ax
           assume es:datos
           ;
           mov bx,offset letras      ; BX toma la dirección
                                     ; desajustada de la
                                     ; variable
                                     ; letras
           mov dh,0                  ; numero de cabezal
           mov dl,0                  ; numero de drive
           mov ch,20                 ; numero de pista
           mov cl,4                  ; numero de sector
           mov al,1                  ; numero de sectores a
                                     ; escribir
           mov ah,3                  ; escribe el sector
           int 13h
           ;
           mov bx,offset salida      ; BX toma la dirección
                                     ; desajustada de la
                                     ; variable
                                     ; salida
           mov dh,0                  ; numero de cabezal
           mov dl,0                  ; numero de drive
           mov ch,20                 ; numero de pista
           mov cl,4                  ; numero de sector
           mov al,1                  ; numero de sectores a
                                     ; leer
           mov ah,2                  ; lee el sector
           int 13h
           ;
```

```
            mov bx,offset salida      ; BX toma la dirección
                                      ; desajustada de la
                                      ; variable
                                      ; salida
            mov cx,12                 ; numero de iteraciones
;
    lazo :  mov al,[bx]               ; AL toma el contenido de
                                      ; la dirección que
                                      ; contiene BX
            push bx                   ; guarda en la pila el
                                      ; valor de BX
            mov ah,14                 ; instrucciones para
            mov bh,0                  ; escribir en pantalla
            int 10h
;
            pop bx                    ; desempila el valor de BX
            inc bx                    ; incrementa BX
            loop lazo                 ; fin de lazo
            ret
 principal endp
 codigo ends
        end principal
```

Ejemplo 4: Programa que escribe, lee de disco y despliega el mensaje "Leda y Jorge"

El programa del ejemplo 10.4 se hecho para que grabe y lea de disco el mensaje 'Leda y Jorge' y luego lo despliegue en la pantalla, esto se muestra en la figura 10.4 suponiendo que dicho programa ha sido grabado con el nombre DISC104.EXE.

CAPITULO 11.

USO DEL IMPRESOR.

La interfase del impresor es un puerto paralelo de E/S. Por este medio el dato es enviado al impresor un byte (8 bits) al mismo tiempo. El impresor debe tener una interfase en paralelo capaz de recibir esos datos.

Algunos impresor tienen interfase serial y ellos no pueden ser usados con la IBM PC con interfase de impresor en paralelo. La mayoría de impresor usados con computadoras personales, no obstante, vienen con interfase en paralelo.

Para usar un impresor debes tener un impresor con adaptador en paralelo.

Para imprimir un caracter en el impresor, la IMB PC coloca el código ASCII del caracter a ser imprimido en la interfase y cuando el impresor esta listo para aceptar datos, el caracter es enviado al impresor para ser imprimido.

Para la manipulación del impresor usaremos las rutinas del BIOS, las cuales son llamadas usando la interrupción 17H la cual se encuentra en la siguiente tabla 11.1.

INT 17H		
AH	FUNCION	REQUERIMIENTO/RESULTADO
0	Imprime caracter	Entrada: AL= Código ASCII del caracter a imprimir DX=Número de impresor(0-3) Salida: AH= Estado del impresor
1	Inicializa el puerto de la impresora	Entrada: DX= Número de Impresor (0-3) Salida: AH= Estado del impresor
2	Lee el estado del impresor	Entrada: DX= Número de impresor (0-3) Salida: AH= Estado del impresor

Tabla 11.1 Opciones de E/S del impresor usando la INT 17H

Las computadoras personales IBM pueden usar no solo un impresor; es decir, que a una misma computadora podemos tener conectado más de un impresor, por esto es que

cuando usamos la interrupción 17H debemos cargar en DX el número de impresor a utilizar, el cual puede ser cualquier número entero entre 0 y 3, de manera que podemos tener hasta cuatro impresores conectados a una computadora.

En la tabla 11.1 se muestran las funciones que podemos realizar con el impresor usando la interrupción 17H, las cuales consisten en:

a) Inicializar el puerto del impresor. Con esta función habilitamos el impresor; es decir, que el impresor queda listo para imprimir cualquier texto que sea enviado por el computador.

b) Imprime caracter. Esta función es la que nos permite a nosotros visualizar en papel la información que nosotros deseamos, como por ejemplo, la salida de un programa, el listado de un programa, un arhivo de texto,...,etc.

c) Lee el estado del impresor. Esta función es importante cuando queremos saber el estado del impresor, de manera que podamos saber si el impresor esta en disponibilidad de aceptar datos para ser imprimidos.

Como se muestra en la tabla 10.1 el estado del impresor es retornado en el registrador AH, en donde los bit's tienen su significado como se muestra en la figura 11.1

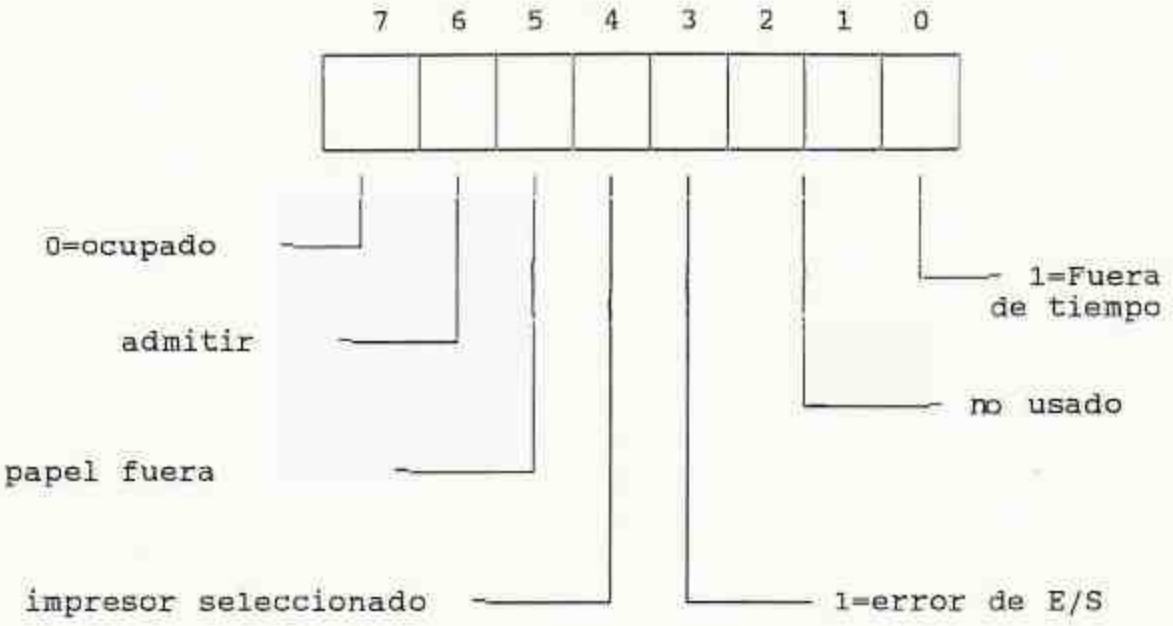


FIG. 11.1 Byte del estado del impresor (AH).

11.1 PROGRAMAS EJEMPLOS SOBRE IMPRESOR

A continuación se presentan programas ejemplos sobre la manipulación del impresor utilizando las funciones que nos proporcionan las rutinas del sistema basico de Entrada/Salida (BIOS) invocadas por medio de la interrupción 17H.

El programa del ejemplo 11.1 al ejecutarlo inicializa el impresor, dejándolo listo para que pueda realizar la función de imprimir caracter.

```
COMMENT * ESTE PROGRAMA INICIALIZA EL IMPRESOR*
;
pila segment para stack
       db 256 dup(0)
pila ends
;
codigo segment para public
inicio proc far
       assume cs:codigo
       push ds
       mov ax,0
       push ax
       ;
       mov ah,1          ; Inicializa el impresor
       mov dx,0          ; Numero del Impresor
       int 17h
       ;
       ret
inicio endp
codigo ends
end inicio
```

Ejemplo 11.1 Consiste en un programa el cual inicializa el impresor

El programa del ejemplo 11.2, la función que realiza al ejecutarlo es que envía el codigo ASCII 41H que corresponde al caracter 'A' para ser imprimido. Tambien, es de hacer notar el uso de los códigos ASCII 0aH y 0dH los cuales corresponden a fin de linea y retorno de carro respectivamente.

```
COMMENT * ESTE PROGRAMA IMPRIME 1 LETRA A  *
pila segment para stack
            db 256 dup(0)
pila ends
codigo segment para public
inicio proc far
       assume cs:codigo
       push ds
       mov ax,0
       push ax
       mov ah,1        ; Inicializa el impresor
       mov dx,0        ; numero del impresor
       int 17h
       ;
       mov ah,0        ; Modalidad para imprimir un caracter
       mov dx,0        ; numero del impresor
       mov al,41h      ; letra A
       int 17h
       ;
       mov ah,0        ; Modalidad para imprimir un caracter
       mov dx,0        ; numero del impresor
       mov al,0ah      ; Codigo de fin de linea
       int 17h
       ;
       mov ah,0        ; Modalidad para imprimir un caracter
       mov dx,0        ; numero del impresor
       mov al,0dh      ; Codigo de retorno de carro
       int 17h
       ret
inicio endp
codigo ends
end inicio
```

Ejemplo 11.2 Es un programa el cual imprime un carácter "A"

La figura 11.2 muestra la salida del programa del ejemplo 11.2 el cual imprime el caracter 'A'.

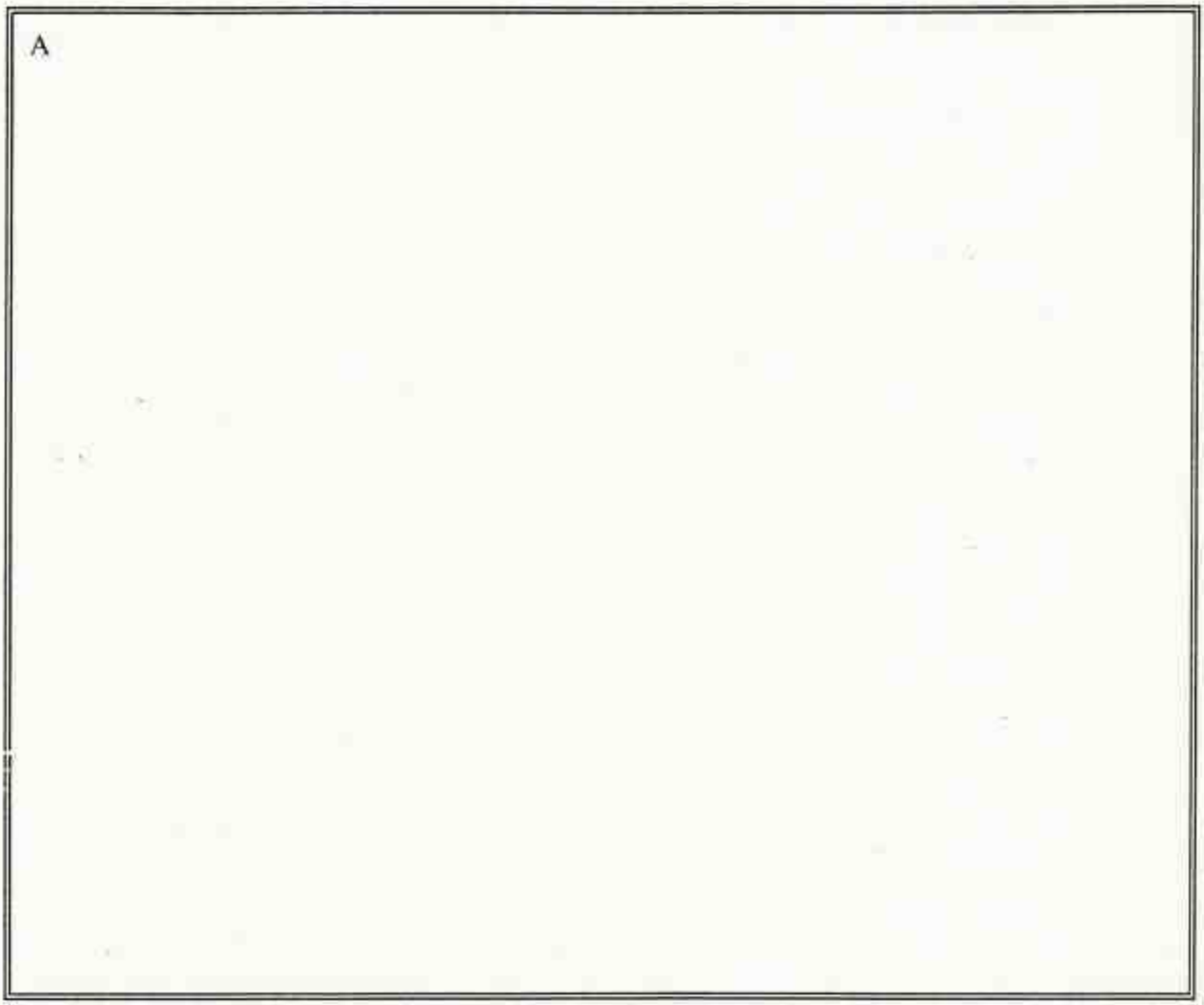


FIG. 11.2 Salida del programa del ejemplo 11.2

```
COMMENT * ESTE PROGRAMA IMPRIME 10 LETRAS A EN FORMA VERTICAL
*
;
pila segment para stack
        db 256 dup(0)
pila ends
codigo segment para public
inicio proc far
        assume cs:codigo
        push ds
        mov ax,0
        push ax
        ;
        mov ah,1            ; inicializan el impresor
        mov dx,0            ; numero de impresor
        int 17h
        ;
        mov cx,10           ; Numero de iteraciones del lazo
 lazo : mov ah,0            ; Modalidad para imprimir
         mov dx,0           ; numero del impresor
        mov al,41h          ; letra A
        int 17h
        ;
        mov ah,0            ; Modalidad para imprimir
        mov dx,0            ; numero del impresor
        mov al,0ah          ; avanza una linea
        int 17h
        ;
        mov ah,0            ; Modalidad para imprimir
        mov dx,0            ; numero del impresor
        mov al,0dh          ; retorno de carro
        int 17h
        ;
        loop lazo
        ret
inicio endp
codigo ends
end inicio
```

Ejemplo 11.3 Es un programa el cual imprime la letra "A" 10 veces en forma vertical.

El programa del ejemplo 11.3 que aparece en la página anterior la función que realiza es imprimir el caracter 'A' (similar a la función del programa del ejemplo 11.2) diez

veces y lo hace en forma vertical. Para eso en el programa se ha usado un lazo el cual realiza diez iteraciones, y en cada una de ellas se usa la función de imprimir caracter tres veces. En la primera vez se envia el código ASCII 41H, que corresponde a la letra 'A' para ser imprimido, la segunda vez se envía el codigo ASCII 0aH con lo cual se realiza un retorno de carro, y en la tercera vez se envía el código ASCII 0dH con lo cual el carro del impresor avanza una linea. De esta forma se logra que los caracteres sean impresos en forma vertical, como se muestra en la figura 11.3.

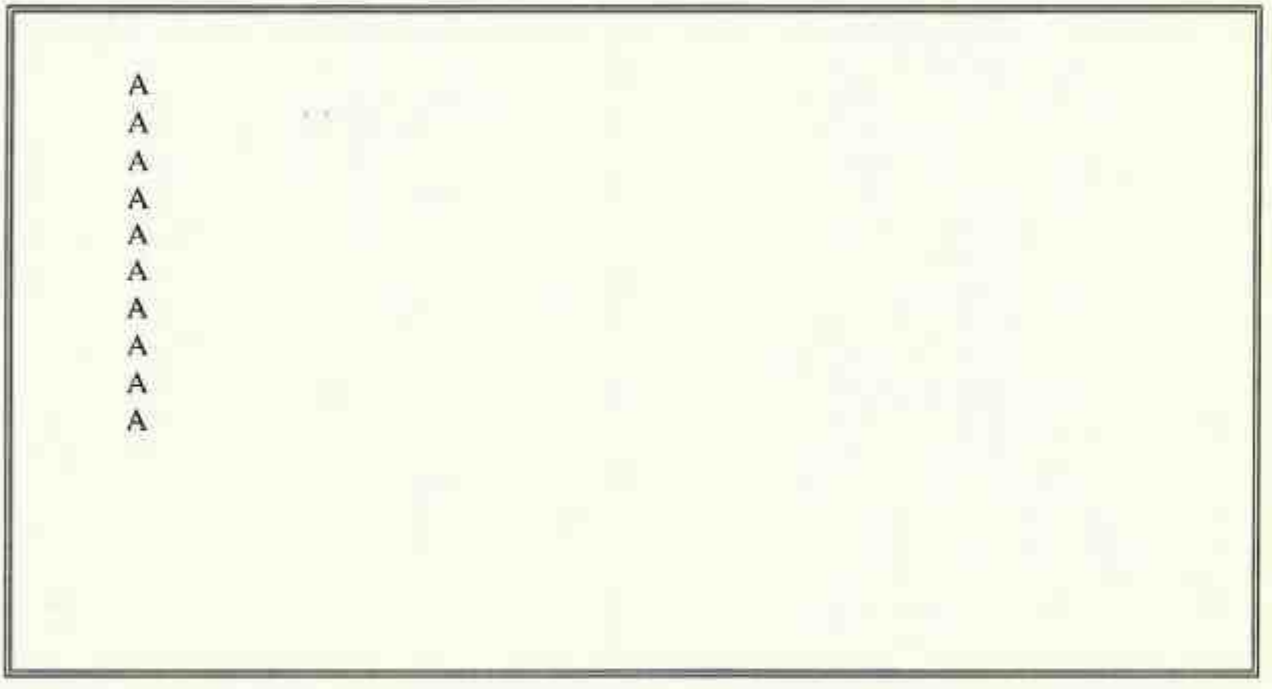


FIG. 11.3 Salida del programa del ejemplo 11.3

```
COMMENT * ESTE PROGRAMA IMPRIME 10 LETRAS A EN UNA LINEA*
;
pila segment para stack
        db 256 dup(0)
pila ends
;
codigo segment para public
inicio proc far
        assume cs:codigo
        push ds
        mov ax,0
        push ax
        ;
        mov ah,1            ; Inicializan el impresor
        mov dx,0            ; numero de impresor
        int 17h
        ;
        mov cx,10           ; Numero de iteraciones del lazo
lazo :  mov ah,0            ; Modalidad para imprimir
        mov dx,0            ; numero de impresora
        mov al,41h          ; letra A
        int 17h
        loop lazo
        ;
        mov dx,0            ; numero de impresora
        mov ah,0            ; Modalidad para imprimir
        mov al,0ah          ; codigo de fin de linea
        int 17h
        ;
        mov dx,0            ; numero de impresora
        mov ah,0            ; Modalidad para imprimir
        mov al,0dh          ; codigo de retorno de carro
        int 17h
        ;
        ret
inicio endp
codigo ends
end inicio
```

Ejemplo 11.4 Programa que imprime 10 letras "A" en una línea

El programa del ejemplo 11.4 de manera similar al programa del ejemplo 11.3 lo que realiza al ejecutarlo es que imprime diez veces el caracter 'A', con la variante que esta vez son imprimidas en una misma linea.

Como puede verse en el ejemplo 11.4 dentro del lazo solamente es enviado al impresor el codigo ASCII 41H (el caracter 'A') y al finalizar el lazo son enviados los codigos ASCII 0ah y 0dh (fin de linea y retorno de carro respectivamente), la salida de este programa puede verse en la figura 11.4.

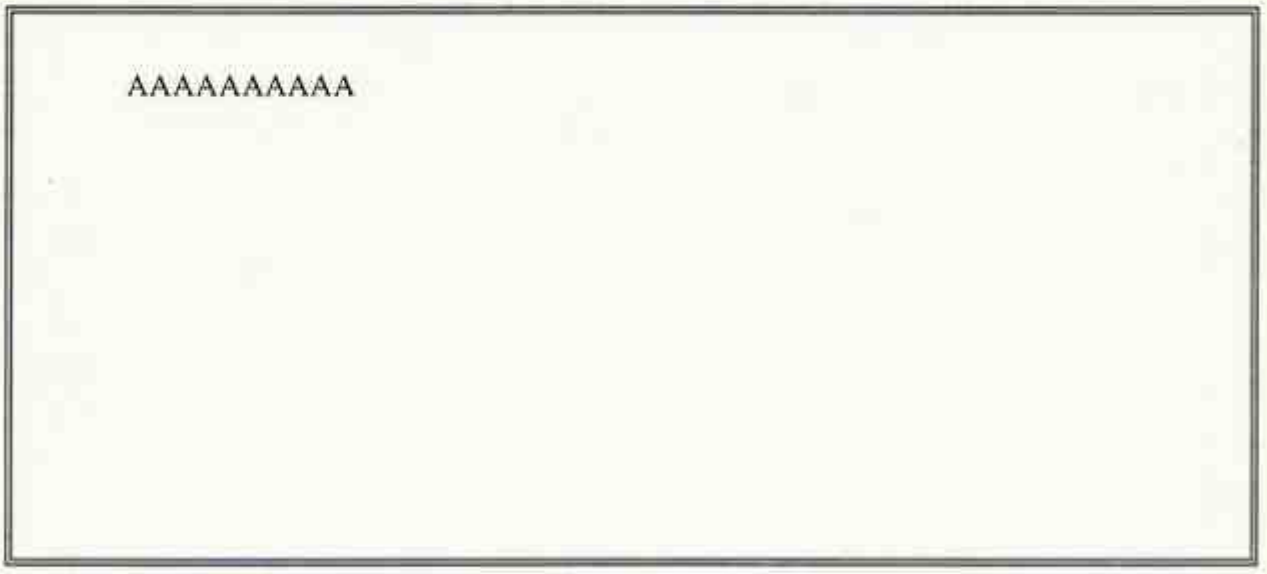


FIG. 11.4 Salida del programa del ejemplo 11.4

```
; Este programa imprime un mensaje
;
PILA SEGMENT PARA STACK
          DB 256 DUP(0)
PILA ENDS
;
DATOS SEGMENT PARA PUBLIC
          SALIDA DB 'ESTO ES USO DE IMPRESOR'
DATOS ENDS
;
CODIGO SEGMENT PARA PUBLIC
PRINCIPAL PROC FAR
          ASSUME CS:CODIGO
          PUSH DS
          MOV AX,0
          PUSH AX
          MOV AX,DATOS
          MOV DS,AX
          ASSUME DS:DATOS
          MOV ES,AX
          ASSUME ES:DATOS
          ;
          MOV BX, OFFSET SALIDA     ; BX toma la direccion
desajustada
                                    ; de la variable SALIDA
          MOV CX,23                 ; Numero de iteraciones del
LAZO
LAZO    : MOV AL,[BX]               ; AL toma el contenido de la
                                    ; direccion que contiene BX
          MOV AH,0                  ; Modalidad para imprimir
          MOV DX,0                  ; Numero de impresor
          INT 17H
          ;
          INC BX                    ; Incrementa a BX en uno
          LOOP LAZO
          ;
          MOV AH,0                  ; Modalidad para imprimir
          MOV AL,0AH                ; Codigo de fin de linea
          MOV DX,0                  ; Numero de impresor
          INT 17H
          MOV AH,0                  ; Modalidad para imprimir
          MOV AL,0DH                ; Codigo de retorno de carro
          MOV DX,0                  ; Numero de impresor
          INT 17H
          ;
          RET
  PRINCIPAL ENDP
CODIGO ENDS
END PRINCIPAL
```

Ejemplo 11.5 Programa que imprime un mensaje sobre uso de impresor

En la figura 11.5 se muestra la salida del programa del ejemplo 11.5, el cual al ejecutarlo imprime el mensaje 'ESTO ES USO DE IMPRESOR' que esta contenido en la variable SALIDA declarada en el segmento de datos del programa.

FIG. 11.5 Salida del programa del ejemplo 11.5

```
COMMENT * ESTE PROGRAMA IMPRIME 30 CARACTERES DIGITADOS*
;
pila segment para stack
      db 256 dup(0)
pila ends
;
codigo segment para public
inicio proc far
      assume cs:codigo
      push ds
      mov ax,0
      push ax
      ;
      mov ah,1          ; Inicializa el impresor
      mov dx,0          ; Numero del impresor
      int 17h
      ;
      mov cx,30         ; Numero de iteraciones del lazo
lazo : mov ah,0
      int 16h
      ;
      mov ah,0          ; Modalidad para Imprimir caracter
      mov dx,0          ; Numero del impresor
      int 17h
      ;
      loop lazo
      ;
      mov ah,0          ; Modalidad para Imprimir caracter
      mov dx,0          ; Numero del impresor
      mov al,0ah        ; Codigo de fin de linea
      int 17h
      ;
      mov ah,0          ; Modalidad para Imprimir caracter
      mov dx,0          ; Numero del impresor
      mov al,0dh        ; retorno de carro
      int 17h

      ret
inicio endp
codigo ends
end inicio
```

Ejemplo 11.6 Este programa imprime 30 caracteres leídos vía teclado

La salida del programa del ejemplo 11.6 se muestra en la figura 11.6. Al ejecutar dicho programa espera que treinta caracteres sean digitados y luego los envía para ser imprimido.

FIG. 11.6 Salida del programa del ejemplo 11.6

APENDICES

APENDICE A
TABLA DE CODIGO ASCII

DEC	HEX	CHAR	DEC	HEX	CHAR	DEC	HEX	CHAR
0	0		48	30	0	96	60	`
1	1	☺	49	31	1	97	61	a
2	2	☻	50	32	2	98	62	b
3	3	♥	51	33	3	99	63	c
4	4	♦	52	34	4	100	64	d
5	5	♣	53	35	5	101	65	e
6	6	♠	54	36	6	102	66	f
7	7	•	55	37	7	103	67	g
8	8	◘	56	38	8	104	68	h
9	9	○	57	39	9	105	69	i
10	A	◙	58	3A	:	106	6A	j
11	B	♂	59	3B	;	107	6B	k
12	C	♀	60	3C	<	108	6C	l
13	D	♪	61	3D	=	109	6D	m
14	E	♫	62	3E	>	110	6E	n
15	F	☼	63	3F	?	111	6F	o
16	0	►	64	40	@	112	70	p
17	11	◄	65	41	A	113	71	q
18	12	↕	66	42	B	114	72	r
19	13	‼	67	43	C	115	73	s
20	14	¶	68	44	D	116	74	t
21	15	§	69	45	E	117	75	u
22	16	▬	70	46	F	118	76	v
23	17	↨	71	47	G	119	77	w
24	18	↑	72	48	H	120	78	x
25	19	↓	73	49	I	121	79	y
26	1A	→	74	4A	J	122	7A	z
27	1B	←	75	4B	K	123	7B	{
28	1C	∟	76	4C	L	124	7C	|
29	1D	↔	77	4D	M	125	7D	}
30	1E	▲	78	4E	N	126	7E	~
31	1F	▼	79	4F	O	127	7F	⌂
32	20		80	50	P	128	80	Ç
33	21	!	81	51	Q	129	81	ü
34	22	"	82	52	R	130	82	é
35	23	#	83	53	S	131	83	â
36	24	$	84	54	T	132	84	ä
37	25	%	85	55	U	133	85	à
38	26	&	86	56	V	134	86	å
39	27	'	87	57	W	135	87	ç
40	28	(	88	58	X	136	88	ê
41	29	)	89	59	Y	137	89	ë
42	2A	*	90	5A	Z	138	8A	è
43	2B	+	91	5B	[	139	8B	ï
44	2C	,	92	5C	\	140	8C	î
45	2D	-	93	5D	]	141	8D	ì
46	2E	.	94	5E	^	142	8E	Ä
47	2F	/	95	5F	_	143	8F	Å

DEC	HEX	CHAR	DEC	HEX	CHAR	DEC	HEX	CHAR
144	90	É	192	C0	└	240	F0	≡
145	91	æ	193	C1	┴	241	F1	±
146	92	Æ	194	C2	┬	242	F2	≥
147	93	ô	195	C3	├	243	F3	≤
148	94	ö	196	C4	─	244	F4	⌠
149	95	ò	197	C5	┼	245	F5	⌡
150	96	û	198	C6	╞	246	F6	÷
151	97	ù	199	C7	╟	247	F7	≈
152	98	ÿ	200	C8	╚	248	F8	°
153	99	Ö	201	C9	╔	249	F9	∙
154	9A	Ü	202	CA	╩	250	FA	·
155	9B	¢	203	CB	╦	251	FB	√
156	9C	£	204	CC	╠	252	FC	ⁿ
157	9D	¥	205	CD	═	253	FD	²
158	9E	₧	206	CE	╬	254	FE	■
159	9F	ƒ	207	CF	╧	255	FF	
160	A0	á	208	D0	╨			
161	A1	í	209	D1	╤			
162	A2	ó	210	D2	╥			
163	A3	ú	211	D3	╙			
164	A4	ñ	212	D4	╘			
165	A5	Ñ	213	D5	╒			
166	A6	ª	214	D6	╓			
167	A7	º	215	D7	╫			
168	A8	¿	216	D8	╪			
169	A9	⌐	217	D9	┘			
170	AA	¬	218	DA	┌			
171	AB	½	219	DB	▉			
172	AC	¼	220	DC	▐			
173	AD	¡	221	DD	▌			
174	AE	«	222	DE	▊			
175	AF	»	223	DF	▀			
176	B0	▒	224	E0	α			
177	B1	▓	225	E1	ß			
178	B2	▓	226	E2	Γ			
179	B3	┃	227	E3	π			
180	B4	┤	228	E4	Σ			
181	B5	╡	229	E5	σ			
182	B6	╢	230	E6	µ			
183	B7	╖	231	E7	τ			
184	B8	╕	232	E8	Φ			
185	B9	╣	233	E9	Θ			
186	BA	║	234	EA	Ω			
187	BB	╗	235	EB	δ			
188	BC	╝	236	EC	∞			
189	BD	╜	237	ED	φ			
190	BE	╛	238	EE	∈			
191	BF	┐	239	EF	∩			

APENDICE B
CONJUNTO DE INSTRUCCIONES DEL 8088

INSTRUCCION	FUNCION
AAA	Ajuste para la adición ASCII
AAD	Ajuste para la división ASCII
AAMM	Ajuste para la multiplicación ASCII
AAS	Ajuste para la resta ASCII
ADC	Acarreo con la suma
ADD	Adición
AND	"y" lógico
CALL	Llama un procedimiento
CBW	Convierte un byte a palabra
CLC	Borra el flag de acarreo
CLD	Borra el flag de dirección
CLI	Borra del flag de interrupción
CMC	Complemento del flag de acarreo
CMP	Compara
CMPS	Compara string
CWD	Convierte una palabra a doble palabra
DDA	Ajuste decimal para la suma
DAS	Ajuste decimal para la resta
DEC	Decrementa en 1
DIV	División (sin signo)
ESC	Escape a dispositivo externo
HLT	Alto
IDIV	División (con signo)
IMUL	Multiplicación entera (con signo)
IN	Entrada o salida a AH ó AX
INC	Incremento en 1
INT	Interrupción
INTO	Interrupción sobre overflow
IRET	Retorno de interrupción

INSTRUCCION	FUNCION
JA/JNBE	Salta si es mayor/si no es menor o igual
JAE/JNB	Salta si es mayor o igual/ no menor
JB/JNAE	Salta si es menor/no mayor o igual
JBE/JNA	Salta si es menor o igual/no mayor
JC	Salta si hay acarreo
JCXZ	Salta si CX es cero
JE/JZ	Salta si es igual/cero
JG/JNLE	Salta si es mayor/no menor o igual
JGE/JNL	Salta si es mayor o igual/no menor
JL/JNGE	Salta si es menor/no mayor o igual
JLE/JNG	Salta si en menor o igual/no mayor
JMP	Salto incondicional
JNC	Salta si no hay acarreo
JNE/JNZ	Salta si no es igual/no es cero
JNO	Salta si hay overflow
JNP/JPO	Salta si no hay paridad/paridad par
JNS	Salta si no hay signo
JO	Salta si hay overflow
JP/JPE	Salta si hay paridad/hasta que haya paridad
JS	Salta si hay signo
LAHF	Carga en AH los flags
LDS	carga pointer dentro del registrador DS
LOCK	Bus prefijo
LODS	Carga strings
LOOP	ciclo de CX veces
LOOPE/LOOPZ	Ciclo si es igual/cero
LOOPNE/LOOPNZ	Ciclo si no es igual/no es cero
LEA	Carga la dirección efectiva a un registrador
LES	Carga puntero al registrador ES
MOV	Mover

INSTRUCCION	FUNCION
MOVS	Mueve string
MUL	Multiplicación (sin signo)
NEG	Negación (cambia signo)
NOP	No operación
NOT	"No" lógico
OR	"O" lógico inclusive
OUT	Salida desde AL o a AX
POP	Saca una palabra de la pila
POPF	Saca los flags desde la pila
PUSH	Coloca una palabra dentro de la pila
PUSHF	Coloca los flags dentro de la pila
RCL	Rota através del carry izquierdo
RCR	Rota através del carry derecho
REP	Repite operación de string
REPE/REPZ	Repite operación de strings mientras es igual/cero
REPNE/REPNZ	Repite la operación de strings
RET	Retorno desde procedimiento
ROL	Rotación izquierda
ROR	Rotación derecha
SAHF	Almacena los flags dentro AH
SAL/SHL	Corrimiento aritmetico izquierdo/lógico
SAR	Corrimiento artimetico derecho
SBB	Sustrae prestando
SCAS	Busca string
SHR	Corrimiento logico derecho
STC	Coloca el flag de acarreo

APENDICE C

MANDATOS DEL ENSAMBLADOR DIVIDIDOS EN CINCO GRUPOS FUNCIONALES
(de acuerdo a su función).

GRUPO	SEUDO-OP.	FORMATO	FUNCION
DEFINICION DE SIMBOLOS	EQU	nombre EQU texto nombre EQU exp-numérica	Asigna texto o el valor de la expresión numérica al nombre permanentemente
	=	nombre = exp-numérica	Asigna el valor de la expresión numérica al nombre, pero puede ser reasignado.
DEFINICION DE DATOS	DB	[nombre] DB exp[,...]	Define una variable o asigna inicialmente. DB asigna uno o más bytes.
	DW	[nombre] DW exp[,...]	Similar a DB, pero asigna dos bytes palabras
	DD	[nombre] DD exp[,...]	Asigna cuatro byte, doble palabra
	DQ	[nombre] DQ exp[,...]	Asigna ocho bytes, cuatro palabras
REFERENCIAS EXTERNAS	PUBLIC	PUBLIC simbolo[,...]	Hace la especificación del símbolo(s) disponible(s) para ser usado por otro módulo objeto que será encadenado a este módulo
	EXTRN	EXTRN nombre : tipo[,]	Especifica símbolos definidos en otro módulo ensamblado
	INCLUDE	INCLUDE arch-especificado	Lee el contenido del archivo fuente especificado dentro del archivo fuente corriente
ESPECIFICACION DE SEGMENTOS	SEGMENT	nomb-seg SEGMENT[tipo alineamiento] : [tipo combinado] : ['clase'] : nomb-seg ENDS	Define límite de un segmento. Cada definición de segmento debe terminar con la oración ENDS
	ASSUME	reg-seg : nomb-seg[,...]	Dice al ensamblador que registrador de segmento (CS, DS, ES o SS) corresponde a un segmento
	PROC	nombre PROC [NEAR] nombre PROC [FAR] nombre ENDP	Asigna un nombre a una secuencia de oraciones del ensamblador. Todas las definiciones PROC deben terminar con una oración ENDP
CONTROL DE ENSAMBLAJE	END	END [etiqueta del punto de entrada]	Marca el final del programa fuente

APENDICE D
INTERRUPCIONES DEL 8088

INTERRUPCIONES DEFINIDAS POR INTEL, DEL BIOS Y DE DOS

INT.	DIRECCION SEG:OFF	DIRECCION 20 BITS	TIPO	PROPOSITO	DESCRIPCION
0	00E3:3072	03EA2	INTEL	Sobreflujo (Overflow) al dividir	Esta interrupción ocurre cuando al dividir se presenta un sobreflujo. El vector de interrupción asociado con ella cambia de acuerdo con las diferentes versiones de DOS.
1	0600:08ED	068ED	INTEL	Paso a paso	Esta interrupción simula la ejecución de programas paso a paso.
2	F00:E2C3 (F000:F85F-XT)	FE2C3	BIOS	Interrupción no enmascarable	Este tipo de interrupción no se puede evitar. Utiliza el BIOS NEN2 procedimiento NMI-INT y aparecen cuando se detectan errores en la memoria sobre la tajeta del sistema (PARITY CHECK 1) o se tienen problemas con tarjetas que se añaden al sistema (PARITY CHECK 2).
3	0600:08E6	068E6	INTEL	Selecciona punto de ruptura (breakpoint)	La interrupción para el procesamiento en una dirección particular.
4	0070:0147	00847	INTEL	Interrupción si existe sobreflujo	La interrupción activa una instrucción INTO de retorno (IRET).
5	F000:FF54	FFF54	BIOS	Imprime pantalla	Esta interrupción se encarga de imprimir el contenido de la pantalla bajo el control del programa. El llamado al procedimiento tipo FAR en PRINT-SCREEN y la dirección 0050:0000 contiene el estado.
6	-	-	-	-	No utilizada
7	-	-	-	-	No utilizada
8	F000:FEA5	FFEA5	BIOS	Interrupción del temporizador	Esta rutina maneja la interrupción del temporizador proveniente del canal 0 del temporizador 8253. Ocurren, aproximadamente, 18.2 interrupciones por segundo. La rutina lleva el conteo del número de interrupciones desde que se energizó la computadora. sigue...

INT.	DIRECCION SEG:OFF	DIRECCION 20 BITS	TIPO	PROPOSITO	DESCRIPCION
8	F000:FEA5	FFEA5	BIOS	Interrupción del temporizador	El procedimiento, de tipo FAR es TIMER-INT y también llama la interrupción 1CH, correspondiente. La que puede contener una rutina escrita por el usuario.
9	F000:E987	FE987	BIOS	Interrupción vicoc del teclado	Esta rutina es un procedimiento FAR KB-INT. La rutina continúa en la dirección f000:ec32 y constituye la interrupción del teclado. La INT 16H es la rutina de I/O del teclado y es más flexible.
A	-	-	-	-	No utilizada
B	-	-	-	-	No utilizada
C	-	-	-	-	No utilizada
D	-	-	-	-	No utilizada
E	F000:EF57	FEF57	BIOS	Disco flexible	Este procedimiento, de tipo FAR, DISK-INT maneja la interrupción del diskette.
F	0070:0147	00847	DOS	INTO	Esta interrupción activa la misma llamada que TYPE 4.
10	F000:F065	FF065	BIOS	Video Interfaz	El conjunto de rutinas asociado con este procedimiento NEAR VIDEO-IO, constituye la interfaz con el TRC. Esta interrupción tiene muchas opciones, las cuales están contenidas en otra tabla de este capítulo.
11	F000:F84D	FF84D	BIOS	Equipo	El procedimiento proporciona el número de puerto para impresora, adaptadores de juego, interfaces RS-232C, número de unidades de diskette, modo de video y tamaño de la RAM.
12	F000:F841	FF841	BIOS	Tamaño de memoria	Proporciona el tamaño de la memoria.
13	F000:EC59 (C800:0256-XT)	FF841	BIOS	I/O en diskette	Este procedimiento llama varias rutinas para llevar a cabo operaciones de I/O en disco.
14	F000:E739	FE739	BIOS	Adaptador de comunicaciones	Este procedimiento permite al usuario la entrada/salida de datos desde el puerto de comunicaciones RS-232C.

INT.	DIRECCION SEG:OFF	DIRECCION 20 BITS	TIPO	PROPOSITO	DESCRIPCION
15	F000:F859	FF859	BIOS	I/O en cassete	Interrupción empleada para controlar operaciones de I/O en cassette.
16	F000:E82E	FE82E	BIOS	I/O del teclado	Esta interrupción utiliza AX para leer el teclado. Se tratará separadamente.
17	F000:EFD2	FEFD2	BIOS	I/O de la impresora	Esta rutina proporciona la comunicación con la impresora. Los parámetros necesarios son colocados en los registros AX y DX.
18	F600:000	F6000	BIOS	BASIC en cassette	Esta interrupción llama al cassette de BASIC.
19	F000:E6F2	FE6F2	BIOS	Carga(bootstrap)	La rutina asociada con esta interrupción lee el sector 1 de la pista 0 del disco en la unidad A, a la que le transfiere el control.
1A	F000:FE6E	FFE6E	BIOS	Hora	Esta rutina permite seleccionar o leer el contenido del reloj que lleva la hora. El registro CX contiene la palabra más significativa del conteo mientras que en DX se encuentra la menos significativa.
1B	0070:0140	00840	DOS	Crtl-Break	Esta interrupción se presenta cada vez que se genera una interrupción proveniente del teclado.
1C	F000:FF53	FFF53	BIOS	Retorno Ficticio (dummy)	Esta interrupción provoca la ejecución de una instrucción RET.
1D	F000:F0A4	FF0A4	BIOS	Parámetros de video	Es una tabla de bytes y rutinas necesarias para establecer varios parámetros para gráficos.
1E	0000:0522	00522	DOS	Tabla del diskette	
1F	00E3:0B07	01937	DOS	Tabla de gráficos	Utilizada con DOS 3.0.
20	PSP:0000	-	DOS	Terminación de programa	Esta interrupción fué generada por DOS para salirse de un programa. Es la primera dirección del área correspondiente al segmento prefijo del programa.
21	Relocalizable	-	DOS	Solicitud de función	Esta interrupción tiene varias opciones.

INT.	DIRECCION SEG:OFF	DIRECCION 20 BITS	TIPO	PROPOSITO	DESCRIPCION
22	PSP:000A	-	DOS	Dirección terminal	Cuando termina la ejecución de un programa esta interrupción transfiere el control a la dirección especificada por el vector de interrupción. Esta interrupción nunca debe generarse de manera directa.
23	PSP:000E	-	DOS	Dirección Ctrl-Break Exit	Esta interrupción generada como respuesta a un Ctrl-Break.
24	PSP:0012	-	DOS	Vector Manejador de un error crítico	Esta interrupción se llama cada vez que ocurre un error crítico dentro de DOS, como puede ser un error de disco.
25	Relocalizable	-	DOS	Lectura absoluta en disco	Esta interrupción transfiere el control, para lectura, al manejador del dispositivo (driver)
26	Relocalizable	-	DOS	Escritura absoluta en disco	Esta interrupción transfiere el control, para escritura, al manejador de dispositivo.
27	Relocalizable	-	DOS	Terminación de programa, dejándolo residente	Este vector es empleado para que al término de un programa ese permanezca residente en la memoria del sistema una vez que DOS toma de nuevo el control .
28-2E	RESERVADO	PARA DOS			
2F	Relocalizable	-	DOS	Interrupción Multiplex	Esta interrupción define un interfaz general entre dos procesos. El número especificado en AH indica a cada manejador. AL contiene la función del manejador.
30-3F	RESERVADO	PARA DOS			

OPCIONES DE INT 10H (I/O DE VIDEO)

AH	PROPOSITO	DESCRIPCION
0	Modo	El registro AL contiene el modo de video: Si AL = 0: 40*25 pixels blanco y negro 1: 40*25 pixels color 2: 80*25 pixels blanco y negro 3: 80*25 pixels color 4: 320*200 pixels color gráfico 5: 320*200 pixels BN gráfico 6: 640*200 pixels BN gráfico
1	Selección del tipo de cursor	Esta posición utiliza los registros CH y CL. Los bits 4 a 0 de CH indican la línea donde comienza el cursor mientras que los bits 4 a 0 de CL señalan donde termina. Los demás bits deben ponerse en 0 con el fin de evitar errores. (Bits 5-7 son 0)
2	Selección de posición del cursor	(DH,DL)=(fila,columna) donde se colocará el cursor. La esquina superior izquierda corresponde a la posición (0,0). El registro BH contiene el número de página (0 para gráficos).
3	Lectura de posición del cursor	(DH,DL)=(fila, columna) donde se encuentra el cursor. (CH,CL)=dimensiones del cursor.
4	Lectura la posición del lector de óptico	Devuelve: Si AH=0 lápiz no pulsado Si AH=1 lápiz pulsado DH,DL= fila, columna CH:línea de rastreo (0-199) BX:pixel columna (0-319 ó 0-639)
5	Selección de página desplegada activa	Cuando existen varias páginas en la memoria de video, esta opción permite seleccionar una de ellas para su exhibición en pantallas de 40*25 y 80*25. AL= 0-7 para 40*25 mientras que para 80*25 AL= 0-3.
6	Cambio de la página anterior activa	AL=número de líneas. Las líneas de la parte inferior son puestas en blanco. Si AL=0 entonces toda la pantalla se pone en blanco. (CH,CL)= coordenadas de la esquina superior izquierda (fila,columna); (DH,DL)= coordenadas de la esquina inferior derecha (fila, columna). El registro BH= atributo a utilizar para las líneas en blanco.
7	Cambio a la página siguiente activa	Idéntica a la anterior con la diferencia de que las líneas se ponen en blanco desde la parte superior hacia la parte inferior.
8	Lee atributo y carácter en el cursor	BH = página en exhibición, AL= carácter y AH= atributo. Esta opción trabaja únicamente en 80*25 y 40*25.
9	Escribe atributo y carácter en el cursor	BH = página en exhibición, CX= cuenta de caracteres, AL= Carácter a escribir, AH= atributo del carácter.
10	Escribe carácter en la posición del cursor	Igual que el anterior, pero sin atributo.
11	Selección de la paleta de color	Coloca paleta de color. El usuario debe experimentar con esta opción para seleccionar los registros.
12	Escribe un punto	DX = número de renglón, CX= número de columna, AL= color(para monitores de alta resolución AL varía la intensidad).
13	Lectura de un punto	DX= número de renglón, CX= número de columna, AL= punto leído.

AH	PROPOSITO	DESCRIPCION
14	Escritura de caracteres en forma alfanumérica y gráfico	AL= caráctar, BL = color de fondo en modo gráfico BH = despliega página en modo alfanumérico
15	Estado actual del video	AL= modo, AH= número de columnas en la pantalla y BH = despliega página activa.

OPCIONES DE INTERRUPCION 16H DE TECLADO

AH	PROPOSITO	DESCRIPCION
0	Lee código de rastreo	Devuelve: AH:Código de rastreo AL:Código de caracter
1	Obtiene estado de buffer	Devuelve: ZF:1 buffer vacío ZF:0 caracteres esperando con el siguiente caracter en AX
2	Obtiene estado del teclado	

FUNCIONES E/S BIOS IMPRESORA- INT 17H

AH	PROPOSITO	DESCRIPCION
1	Imprimir un caracter	AL:Caracter DX: Imprimir número Devuelve AH: Estado
2	Inicializar impresora	DX: Número de impresora Devuelve AH: Estado
3	Leer estado	DX: Número de impresora Devuelve AH: Estado

INT 21H: DE DOS

AH	Propósito	Tipo	Descripción
0	Terminación de programa.	Control	Termina la ejecución de un programa.
1	Entrada desde el teclado.	Teclado	Espera entrada proveniente del teclado, la exhibe y la coloca en el registro AL.
2	Exhibe salida	Display	Exhibe el carácter en DL.
3	Entrada auxiliar	Diversos	Espera un carácter proveniente del puerto COM y lo coloca en AL.
4	Salida auxiliar	Diversos	Envía al puerto COM el carácter en DL.
5	Salida Impresora	Impresora	Envía a la impresora el carácter en DL.
6	I/O directo de consola	Teclado	Espera hasta recibir un carácter proveniente del teclado (no verifica Ctrl-Break.
7	Entrada de consola directa con eco desactivado	Teclado	Espera hasta recibir un carácter proveniente del teclado. Y lo coloca en AL.
8	Entrada desde la consola sin eco.	Teclado	Espera hasta recibir un carácter proveniente del teclado, entrega en AL y se ejecuta una interrupción Ctrl-Break.
9	Impresión de cadena.	Display	Presenta una cadena de caracteres en la pantalla. La cadena debe finalizar en $.
A	Entrada desde el teclado a través de buffer	Teclado	Lee los caracteres que proviene del teclado en un buffer. DS:DX apunta al buffer. El primer byte es el número máximo de caracteres mientras que el segunda byte indica el número de caracteres leídos.
B	Verifica el estado de entrada normal.	Teclado	Verifica, si existe un carácter disponible proveniente del teclado.
C	Limpia el buffer del teclado e invoca una función del teclado.	Teclado	Limpia el buffer del teclado y ejecuta la llamada a la función en AL, (únicamente 01H, 06H, 07H, 08H o 0AH).
D	Restablece el disco	Disco	Se pierden todos los archivos que no hayan sido cerrados.
E	Selección de disco.	Disco	Selecciona la unidad de disco en DL (0 = A, 1 = B, etc.)
F	Abre archivo	Archivo.	Busca el directorio para apuntar el archivo que entra DS:DX. AL = FFH (no se encuentra) o AL = 00H (encontrado). Si se encuentra se llena el FCB.
10	Cierra archivo	Archivo	Cierra el archivo después de una operación de escritura. DS:DX apunta aFCB.
11	Búsqueda para la primera entrada	Disco	Busca en el directorio la primera ocurrencia en que igual el nombre del archivo. Si no encuentra Al = FFH.
12	Búsqueda para la siguiente entrada.	Disco	Después de haber encontrado el nombre del archivbo, esta llamada continuará la busqueda para la siguiente ocurrencia.
13	Borrar archivo.	Archivo.	Borra del directorio todas las entradas que señala el apuntador DS:DX.
14	Lectura secencial.	Disco	Carga el registro direccionado por el bloque actual y lo graba en DTA e incrementa la dirección del registro.
15	Escritura secuencial	Disco	Lo contrario a 14H.
16	Crear archivo	Archivo	Busca en el directorio la entrada deseada, si lo encuentra la utiliza nuevamente, de los contrario abre un nuevo archivo.
17	Renombrar un archivo.	Archivo	Cámbia el nombre del archivo por el nombre en DS:DX + 11.
19	Unidad de disco actual	Disco	Determina el default de la unidad de disco in AL.

AH	Propósito	Tipo	Descripción
1A	Coloca la DTA del disco.	Disco.	Coloca la dirección de transferencia de disco en DS:DX.
1B	Información de la tabla de asignación.	Disco	Entrega un apuntador contenido en DS:DX apunta descriptor del medio, DX = número de la unidad de asignación, AL = número del sector/unidad de asignación CX = tamaño del sector.
1C	Tabla de información de asignación para la unidad de disco.	Disco	DL = número de la unidad de disco; esta función proporciona el mismo parámetro que 1CH.
21	Lectura aleatoria.	Disco.	Lee la grabación direccionada por el bloque actual y registra los campos en el área de memoria, correspondientes a DTA.
22	Escritura aleatoria.	Disco.	Lo contrario a la 21H.
23	Tamaño de archivo	Archivo	Busca en el directorio una entrada a igualar según DS:DX coloca el registro de grabación aleatorio FCB igual al número de grabaciones en el archivo.
24	Campo de registro relativo.	Archivo	Coloca el campo de registro aleatorio en la misma dirección que el bloque actual y los campos de registro.
25	Colocar vector de interrupción.	Diversos	Coloca el vector de interrupción en AL en la dirección DS:DX.
26	Crea nuevo segmento de programa.	Diversos.	Esta llamada nunca debe utilizarse.
27	Lectura de bloque aleatorio.	Disco	Lee el número de registros en CX desde DS:DX, en DTA.
28	Escritura de vloque aleatorio.	Disco	Lo contrario a 27H.
29	Analizar el nombre de archivo.	Archivo	Vease el manual DOS Technical Reference.
2A	Obtener la fecha	Diversos	Regresa AL = día de la semana, CX = año, DH = mes y DL = día del mes.
2B	Coloca la fecha.	Diversos	Inverso a 2AH.
2C	Obtener la hora.	Diversos	Regresa:CH = hora, CL = minutes, DH = segundos y DL = día delo mes.
2D	Coloca la hora.	Diversos	Servico opuesto al 2CH.
2E	Activa/Desactiva switch de verificación.	Diversos	Cuando se encuentra activada, DOS realiza la verificación para cada operación de escritura en disco. AL = 0 desactivar; AL = 1 activar.
2F	Obtiene DTA.	Disco	Regresa la dirección de transferencia en ES:BX.
30	Obtener la versión del DOS.	Diversos	Regresa en AL el número superior que corresponde a la versión del DOS, AH contiene el número inferior.
31	Terminación del proceso/conserva residente	Control	Vease el manual DOS Technical Reference.
33	Verificar Ctrl-Break.	Control	Solicita/coloca el estado BREAK.AL = 0 (solicita) y Al = 1 (coloca). Si DL = 1 (activa).
35	Obtener vector.	Diversos	Para el número de interrupción en AL, regresa el apuntador en ES:BX.
36	Obtener espacio libre en disco.	Disco	Regresa para DL (unidad de disco): en BX, los espacios (en clusters) disponibles; en DX, clusters/unidad de disco; en CX, los bytes/sector; y en AX, los sectores/cluster.

AH	Propósito	Tipo	Descripción
38	Información dependiente del país.	Diversos	Véase el manual DOS technical Refrence.
39	Crear subdirectorio.	Disco	Genera la función MKDIR; con DS:DX apuntando a una cadena ASCIIZ que contiene la unidad de disco y nombre de la ruta del directorio.
3A	Eliminar subdirectorio.	Disco	Función RMDIR; DS:DX apunta a la cadena que contiene los nombres de la unidad de disco y la ruta.
3B	Cambio de directorio.	Disco	Función CHDIR:DS:DX apunta a la cadena que contiene los nombres de la unidad de disco y la ruta.
3C	Crear archivo.	Archivo	Función CREATE; si el archivo al que apunta DS:DX no existe se abre un nuevo archivo.
3D	Abre archivo.	Archivo	DS:DX apunta al archivo; AL = 0 (sólo lectura), 1 (sólo escritura) o 2 (lectura/escritura). (Vease Manual DOS Technical Reference.)
3E	Cierra manejador de archivo.	Archivo	BX = manejador de archivo; se cierra el archivo, se actualiza el directorio y se remueven los buffers internos del archivo.
3F	Lectura desde archivo/dispositivo.	Archivo.	BX = manejador de archivo, CX = número de bytes que se desea leer y DS:DX = buffer a ser cargado; después de la llamada, AX = número de bytes leídos.
40	Escritura en archivo/dispositivo.	Archivo	Operación inversa a 3F.
41	Borra archivo del directorio.	Archivo	Elimina una entrada del directorio asociada con el nombre del archivo apuntado en DS:DX.
42	Mueve el apuntador de lectura/escritura del archivo.	Archivo	Véase el manual de DOS Technical Reference.
43	Cambia modo de archivo.	Archivo	Véase el manual de DOS technical Reference.
44	Control de I/O para dispositivos.	I/O	Véase el manual de DOS Technical Reference.
45	Manejador de archivo duplicado.	Archivo	A la entrada BX = manejador de archivo al terminar, AX = duplicado.
46	Fuerza duplicación en el Manejador de archivo.	Archivo	Fuerza a que el manejador en CX se refiera al mismo archivo en la misma posición que el manejador en BX.
47	Obtiene el directorio actual.	Disco	DL = número de la unidad de disco; DS:SI = apuntador al área del usuario de 64 bytes, la que contiene el directorio; AX contiene el código de error.
48	Asigna memoria.	Memoria	BX = número de párrafos, y AX1,0000 apunta a los bloques de asignación.
49	Libera memoria asignada.	Memoria	Libera la memoria asignada en 48H.
4A	Modifica los bloques de memoria asignada.	Memoria	Modifica los bloques para contener el tamaño de un bloque nuevo. ES = bloque del segmento BX = tamaño en párrafos del nuevo bloque.
4B	Carga/ejecuta programa.	Control	Permite que un pprograma de aplicación ejecute otro. Al término de éste, el control vuelve al primer programa. DS:DX apunta al programa y ES:BX apunta a un bloque de parámetros. Para la carga.

AH	Propósito	Tipo	Descripción
4C	Termina proceso	Control	Finaliza el proceso de ejecución
4D	Obtiene el código de retorno	Diversos	Véase el manual DOS Technical Reference
4E	Encuentro del primer archivo indicado	Archivo	Encuentro del primer nombre de archivo indicado que se iguala al nombre de archivo que apunta DS:DX. CX atributo de búsqueda
4F	Encuentra el siguiente archivo indicado	Archivo	Es igual al 4EH con la excepción de que encuentra el segundo archivo. La DTA contiene, en este caso información proporcionada por 4EH o por una llamada previa a 4FH.
54	Obtiene el estado de verificación	Diversos	Regresa el valor de la verificación con 2EH en AL
56	Renombra un archivo	Archivo	Modifica el nombre de un archivo en DS:DX con ES:DI.
57	Obtiene/coloca fecha y hora de archivo	Diversos	A la entrada AL = 0 (obtener) o AL = 1 (fijar), BX = manejador de archivo, CX = hora y DX = fecha.
59	Obtiene error extendido (DOS 3.00 y 3.10)	Error	Proporciona información adicional de un error. (Véase el manual DOS Technical Reference)
5A	Crea archivo único (DOS 3.00 y 3.10).	Archivo	Genera un archivo apuntado por DS:DX (la ruta debe terminar con /). CX = atributo.
5B	Crea un nuevo archivo (DOS 3.00 y 3.10).	Archivo	Genera un nuevo archivo apuntado por DS:DX CX = atributo.
5C	Abre/cierra acceso a un archivo (DOS 3.00/3.10)	Archivo	AI = 0 (abre) o AI = 1 (cierra), BX manejador de archivo, CX = byte más significativo del desplazamiento, DX = Byte menos significativo del desplazamiento, SI = parte más significativa, DI = parte menos significativa.
5E00	Obtiene nombre de la máquina (DOS 3.10).	Diversos	DS:DX apunta a la localidad donde se regresa, el nombre de la computadora.
5E02	Inicialización de la impresora/	Red	BX = índice de la lista redireccionada: CX = longitud de la cadena; DS:SI apunta a la cadena que será colocada delante de todos los archivos a imprimirse.
5E03	Obtiene el estado de la impresora.	Red	Opuesto al 5E02.
5F02	Obtiene entrada de la lista de redireccionada (DOS 3.10)	Red	Regresa asignaciones de la red no local.
5F03	Redirecciona dispositivo (DOS 3.10).	Red	Principalmente para redes.
5F04	Cancela redirección	Red	Principalmente para redes.
62	Obtiene PSP (DOS 3.00 y 3.10)	Diversos	Proporciona en BX el segmento prefijo del programa.

APENDICE E

PROGRAMAS ADICIONALES SOBRE APLICACIONES EN LENGUAJE ENSAMBLADOR

```
; este programa es un modelo de la presentacion de pantallas.
pila segment para stack
     db 256 dup(0)
pila ends
datos segment para public
  var db '-------------------------------------------------------------',0ah,0dh
      db '|                                                           |',0ah,0dh
      db '|       M E N U   P R I N C I P A L                         |',0ah,0dh
      DB '|                                                           |',0ah,0dh
      DB '|           1. Consulta                                     |',0ah,0dh
      db '|                                                           |',0ah,0dh
      db '|           2. Modificar                                    |',0ah,0dh
      db '|                                                           |',0ah,0dh
      db '|           3. Agregar                                      |',0ah,0dh
      db '|                                                           |',0ah,0dh
      db '|      Digite su opcion...                                  |',0ah,0dh
      db '-------------------------------------------------------------','$'
datos ends
;
codigo segment para public
principal proc far
          assume cs:codigo
          push ds
          mov ax,0
          push ax
          mov ax,datos
          mov ds,ax
          assume ds:datos
          ;
          mov dx,0            ; coloca el cursor en la posicion (0,0)
          mov ah,2
          mov bh,0
          int 10h
          ;
          mov cx,2000         ; borra pantalla con atributo intenso.
          mov ah,9
          mov al,' '
          mov bh,0
          mov bl,70h
          int 10h
          ;
          mov ah,9            ; despliega el contenido de la variable 'var'
          lea dx,var
          int 21h
          ;
          mov dh,10           ; posiciona el cursor en (10,28)
          mov dl,28
          mov ah,2
          mov bh,0
          int 10h
          ;
          mov ah,0            ; espera que una tecla sea presionada
          int 16h
          ret
principal endp
codigo ends
end principal
```

```
;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
; Este programa produce diferentes sonidos al presio-;
; nar cualquier tecla.                                ;
;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;; ;;;;;;;;;
PILA SEGMENT PARA STACK
     DB 256 DUP(0)
PILA ENDS
DATOS SEGMENT PARA PUBLIC
      FREQ DW 1000,500,100,2,3 DUP(294)    ; ESTOS SON LOS VALORES DE LOS
           DW 3 DUP(294),3300,392,392      ; TONOS DEL SONIDO
           DW 330,294,262,294,4 DUP(330)
           DW 294,294,330,294,262
      DURA DW 6 DUP(25),50                 ; ESTOS SON LOS VALORES DE LA
           DW 2 DUP(25,25,50)              ; DURACION DEL SONIDO.
           DW 12 DUP(25),99
DATOS ENDS
CODIGO SEGMENT PARA PUBLIC
PRINCIPAL PROC FAR
          ASSUME CS : CODIGO
          PUSH DS
          MOV AX,0
          PUSH AX
          MOV AX , DATOS
          MOV DS,AX
          ASSUME DS:DATOS
          ;
          MOV BX, OFFSET FREQ    ; BX TOMA LA DIRTECCION DESAJUSTADA
          ;                        DE LA VARIABLE 'FREQ'
          MOV CX,26
LAZO :    IN AL,61H              ; SE ENCIENDE LA BOCINA
          OR AL,3
          OUT 61H,AL
          MOV AL,0B6H
          ;
          OUT 43H,AL             ; SE DA EL TONO DEL SONIDO
          MOV DX,[BX]
          MOV AL,DL
          OUT 42H,AL
          MOV AL,DH
          OUT 42H,AL
          ;
          INC BX                 ; INCREMENTA BX
          ;
          MOV AH,0               ; ESPERA QUE UNA TECLA SEA PRESIONADA
          INT 16H
          LOOP LAZO              ; FIN DEL LAZO.
          ;
          IN AL,61H              ; SE APAGA LA BOCINA
          AND AL,0FCH
          OUT 61H,AL
          RET
PRINCIPAL ENDP
CODIGO ENDS
END PRINCIPAL

;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
; Este programa produce diferentes sonidos con dife-  ;
; rente duracion.                                     ;
;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;; ;;;;;;;;;
PILA SEGMENT PARA STACK
     DB 256 DUP(0)
PILA ENDS
DATOS SEGMENT PARA PUBLIC
```

```
      FREQ DW 330,294,262,294,3 DUP(330)     ; ESTOS VALORES SON EL
           DW 3 DUP(294),330,392,392         ; TONO DEL SONIDO
           DW 330,294,262,294,4 DUP(330)
           DW 294,294,330,294,262
           DW 330,294,262,294,3 DUP(330)
           DW 3 DUP(294),330,392,392
           DW 330,294,262,294,4 DUP(330)
           DW 294,294,330,294,262
           ;
      DURA DW 6 DUP(20),40                   ; ESTOS VALORES SON LA
           DW 2 DUP(20),40,2 DUP(20),40      ; DURACION DEL SONIDO
           DW 12 DUP(20),40
           DW 6 DUP(20), 40
           DW 2 DUP(20), 40, 2 DUP(20),40
           DW 12 DUP(20),40
DATOS ENDS
CODIGO SEGMENT PARA PUBLIC
PRINCIPAL PROC FAR
;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
; inicio del prologo del programa;
;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
          ASSUME CS : CODIGO
          PUSH DS
          MOV AX,0
          PUSH AX
          MOV A X , DATOS
          MOV DS,AX          ;;;;;;;;;;;;;;;;;;;
          ASSUME DS:DATOS ; fin del prologo;
                             ;;;;;;;;;;;;;;;;;;;
                             ;
          MOV DI,OFFSET FREQ ; da la direccion desajustada
                             ; de la variable FREQ al regis-
                             ; trador DI.
          MOV SI,OFFSET DURA ; da la direccion desajustada
                             ; de la variable DURA al regis-
                             ; trador SI.
                             ;
          MOV AL,0B6H        ;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
          OUT 43H,AL         ; este bloque de instrucciones   ;
          IN AL,61H          ; activa la bocina para producir;
          OR AL,03H          ; sonidos.                       ;
          OUT 61H,AL         ;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
                             ;
          MOV CX,42          ; inicio de LAZO.
  LAZO :  MOV DX,[DI]        ; carga la frecuencia en DX.
          MOV BX,[SI]        ; carga la duracion en BX.
          MOV AL,DL          ;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
          OUT 42H,AL         ; con este bloque de instruc-    ;
          MOV AL,DH          ; ciones se carga en el regis-   ;
          OUT 42H,AL         ; trador de conteo el valor del  ;
                             ; del tono.                      ;
                             ;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
                             ;
   SIGA : CALL DURACION      ; llamada a la subrutina DURACION.
           DEC BX            ; decrementa el valor de BX en 1.
          JNZ SIGA           ; si BX<>0 ir a SIGA.
          INC DI             ; inrementa el DI.
          LOOP LAZO          ; fin de LAZO.
                             ;
          IN AL,61H          ;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
          AND AL,0FCH        ; este bloque de instruc-        ;
          OUT 61H,AL         ; ciones desactiva la vocina.    ;
                             ;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
```

```
          RET
PRINCIPAL ENDP
;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
; esta subrutina consiste en un lazo el cual se ejecuta  ;
; en un tiempo de aproximadamente 1/2 de segundo.        ;
;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
DURACION PROC NEAR
          PUSH CX
          MOV CX,8942H
  LASO3 : LOOP LASO3
          POP CX
          RET
DURACION ENDP
CODIGO ENDS
    END PRINCIPAL
```

```
;-------------------------------------------------
; Este programa produce diferentes sonidos tratando
; de simular el sonido emitido por una sirena de
; ambulancia.
;-------------------------------------------------
PILA SEGMENT PARA STACK
     DB 256 DUP(0)
PILA ENDS
;
MUSICA SEGMENT PARA PUBLIC
RUT_PRINCIPAL PROC FAR
          ASSUME CS : MUSICA
          PUSH DS
          MOV AX,0
          PUSH AX
;    el siguiente bloque de instrucciones activa el    ;
;              altavoz del sistema
          MOV AL,0B6H          ; instrucciones
          OUT 43H,AL           ; para activar
          IN AL,61H            ; el altavoz
          OR AL,03H            ; del sistema
          OUT 61H,AL
          MOV DX,300           ;cargamos en DX la primer frecuencia
          MOV CX,80            ;inicio del lazo
 LAZO:    mov bx,000dh         ; valor inicial de BX
                               ;
          MOV AL,DL            ; instrucciones para cargar
          OUT 42H,AL           ; en el registrador de conteo
          MOV AL,DH            ; el valor del tono a ser ge-
          OUT 42H,AL           ; nerado.
          ;
   SIGA : CALL PAUSA           ; la subrutina PAUSA estara
          DEC BX               ; ejecutandose hasta que BX
          JNZ SIGA             ; sea cero
                               ;
          ADD DX,4             ; incrementamos la frecuencia en 4
          LOOP LAZO            ; ejecutar para proxima frecuencia
                               ;
          IN AL,61H            ; bloque de instrucciones
          AND AL,0FCH          ; para apagar el altavoz
          OUT 61H,AL           ; del sistema
          RET
RUT_PRINCIPAL ENDP
PAUSA PROC NEAR                ; subrutina utilizada
          PUSH CX              ; para mantener el sonido
          MOV CX,8942H         ; de una frecuencia en 1/2
  LASO3 : LOOP LASO3           ; segundo aproximadamente
          POP CX
          RET
PAUSA ENDP
MUSICA ENDS
    END RUT_PRINCIPAL
```

GLOSARIO

ALU
Unidad Aritmética Lógica

ASCII
American Standard Code for Information Interchange

BIOS
Conjunto de programas para el 8088 que con almacenados en la computadora personal. Sistema Básico de Entrada/Salida

BIT
Unidad fundamental de almacenamiento de información. Acrónimo de las palabras inglesas **BI**nary-digiT

BUS
Canal o camino a través del cual los componentes de una computadora, o las unidades en que ésta se divide, se comunican entre sí

BYTE
Combinación o agrupación de ocho bits

CIRCUITO INTEGRADO O CHIP
Dispositivo electrónico compuesto por un conjunto de componentes conectados permanentemente entre sí e incluidos en una placa de silicio de menos de 1mm^2, formando un conjunto en miniatura capaz de desarrollar las mismas funciones de un circuito formado por elementos discretos

CODIFICACION
Operación que transforma una información en paquetes o grupos de bits

CODIGO
Conjunto de reglas con las cuales utilizamos los bit para representar las informaciones (Por ejemplo: números y letras)

CODIGO BINARIO
Expresa, números binarios enteros, donde a cada posición de bit se le asigna una potencia de 2.

CODIGO INTERNO
Se llama así al que cada computadora adopta para representar los caracteres en su propia memoria.

COLA
Es una línea de espera como las que se forman en las cajas de los supermercados

CPU
Unidad de Procesamiento Central

CTR
Controlador que se encarga de visualizar el texto de los programas y cualquier información alfanumérica que se utilizase

DIRECCION DESAJUSTADA
La dirección de inicio de un segmento (párrafo), consta de cuatro dígitos. A partir de esta dirección siguen otras referidas a ésta, llamadas direcciones desajustadas

DIRECCION DE SEGMENTO
Dirección de inicio de un segmento

DIRECCION EFECTIVA
La dirección efectiva en el espacio de memoria de un megabytes, se obtiene combinando las direcciones de segmento con la del desajuste

DISKETTE
Unidad de almacenamiento de archivos

DMA
Direct Acces Mode

DOS
Sistema Operativo de Disco. El DOS es un programa de control que maneja varios recursos (memoria, espacio de disco, etc.) del computador.

FLAG
Se utiliza como una bandera

INTERFACE
Medio para relacionar o comunicar dos unidades de un computador

INTERRUPCION
Son instrucciones que se dan al microprocesador 8088, las cuales detienen la ejecución del programa

LAZO
Se utiliza para repetir instrucciones varias veces

LENGUAJE MAQUINA
Lenguaje interno que utilizan las computadoras para leer los programas

LENGUAJE ENSAMBLADOR
Es un lenguaje de programación muy cercano al lenguaje máquina

LINK
Se utiliza para enlazar los programas en lenguaje ensamblador. Genera el archivo ejecutable.

MASM
Se utiliza para transformar el programa fuente el programa objeto. Este convierte tu archivo en lenguaje máquina

MEMORIA RAM
Memoria de Lectura y escritura de acceso aleatorio

MEMORIA ROM
Memoria de Lectura

MICROPROCESADOR
Se define como microprocesador a la unidad central de proceso de datos, constituida por un solo circuito integrado, C.I., de alta escala en la integración de sus componentes

NIBBLE
Se llama así a la mitad de un byte

OPERADOR
Un operador es un modificador, usado en el campo operando de una instrucción del lenguaje ensamblador

PALABRA
Grupo de dos bytes

PARRAFO
El segmento debe comenzar en una posición de memoria cuya dirección sea multiplo de 16, a esta dirección se le conoce con el nombre de PARRAFO

PILA
Es una región de la memoria que es colocada aparte para almacenamiento temporal de datos

PIPELINED
Es el método que hace lo posible para que, cuando la CPU ejecute una instrucción haya otra esperando ser ejecutada

PIT
Programable Interval Timer

PIXEL
En el estudio de gráficos a la intersección de una fila y columna

POP
Saca un dato de la pila

PPI
Programable Peripheral Interface

PSEUDO OPERADOR
Los seudo-operadores o mandatos son comandos para el lenguaje ensamblador, específicamente para el microprocesador 8088

PUSH
Coloca un dato en la pila

REGISTRADORES
Son los componentes más importantes dentro del microprocesador

SALTO
cambia el curso normal de un programa

SEGMENTO
Area continua de memoria que pueden tener hasta un máximo de 64K bytes de longitud

SUBRUTINAS
Conjunto de instrucciones utilizadas varias veces en un mismo programa

WORDSTAR
Procesador de palabras, que posee la forma No-Documento y sirve para digitar programas