Unidad I: Introducción al lenguaje ensamblador

1.1 Importancia de la programación en lenguaje ensamblador

Para comenzar el curso empezaremos conociendo que es el lenguaje ensamblador que utilizaremos y algunos conceptos básicos del mismo:

Definición: El lenguaje ensamblador es un tipo de lenguaje de bajo nivel utilizado para escribir programas informáticos, y constituye la representación más directa del código máquina específico para cada arquitectura de microprocesador.

La importancia del lenguaje ensamblador es principalmente que se trabaja directamente con el microprocesador; por lo cual se debe de conocer el funcionamiento interno de este, tiene la ventaja de que en el se puede realizar cualquier tipo de programas que en los lenguajes de alto nivel no lo pueden realizar. Otro punto sería que los programas en ensamblador ocupan menos espacio en memoria.

1.2 El procesador y sus registros internos

Los registros del procesador se emplean para controlar instrucciones en ejecución, manejar direccionamiento de memoria y proporcionar capacidad aritmética. Los registros son direccionable por medio de un nombre. Los bits por convención, se numeran de derecha a izquierda, como en:

... 15 14 13 12 11 10 9 8 7 6 5 4 3 2 1 0

Registros de segmento

Un registro de segmento tiene 16 bits de longitud y facilita un área de memoria para direccionamiento conocida como el segmento actual.

Registro CS. El DOS almacena la dirección inicial del segmento de código de un programa en el registro CS. Esta dirección de segmento, mas un valor de desplazamiento en el registro apuntador de instrucción (IP), indica la dirección de una instrucción que es buscada para su ejecución.

Registro DS. La dirección inicial de un segmento de datos de programa es almacenada en el registro DS. En términos sencillos, esta dirección, más un valor de desplazamiento en una instrucción, genera una referencia a la localidad de un byte específico en el segmento de datos.

Registro SS. El registro SS permite la colocación en memoria de una pila, para almacenamiento temporal de direcciones y datos. El DOS almacena la dirección de inicio del segmento de pila de un programa en el registro SS. Esta dirección de segmento, más un valor de desplazamiento en el registro del apuntador de pila (SP), indica la palabra actual en la pila que está siendo direccionada.

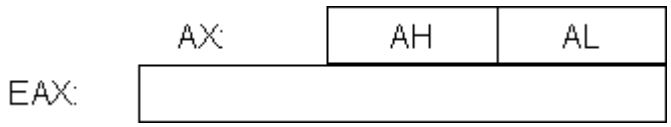
Registros ES. Algunas operaciones con cadenas de caracteres (datos de caracteres) utilizan el registro extra de segmento para manejar el direccionamiento de memoria. En este contexto, el registro ES está asociado con el registro DI (índice). Un programa que requiere el uso del registro ES puede inicializarlo con una dirección de segmento apropiada.

Registros FS y GS. Son registros extra de segmento en los procesadores 80386 y posteriores.

Registros de propósito general.

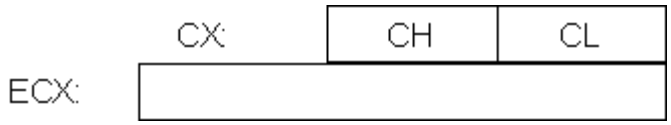
Los registros de propósito general AX, BX, CX y DX son los caballos de batalla del sistema. Son únicos en el sentido de que se puede direccionarlos como una palabra o como una parte de un byte. El último byte de la izquierda es la parte "alta", y el ultimo byte de la derecha es la parte "baja". Por ejemplo, el registro CX consta de una parte CH (alta) y una parte Cl (baja), y usted puede referirse a cualquier parte por su nombre.

Registro AX. El registro AX, el acumulador principal, es utilizado para operaciones que implican entrada/salida y la mayor parte de la aritmética. Por ejemplo, las instrucciones para multiplicar, dividir y traducir suponen el uso del AX. También, algunas operaciones generan código más eficiente si se refieren al AX en lugar de a los otros registros.



Registro BX. El BX es conocido como el registro base ya que es el único registro de propósito general que puede ser índice para direccionamiento indexado. También es común emplear el BX para cálculos.

Registro DX. El DX es conocido como l registro de datos. Alguna operaciones de entrada/salida requieren uso, y las operaciones de multiplicación y división con cifras grandes suponen al DX y al AX trabajando juntos.



Pude usar los registros de propósito general para suma y resta de cifras de 8, 16 o 32 bits.

Registro de Apuntador de Instrucciones.

El registro apuntador de instrucciones (IP) de 16 bits contiene el desplazamiento de dirección de la siguiente instrucción que se ejecuta. El IP está asociado con el registro CS en el sentido de que el IP indica la instrucción actual dentro del segmento de código que se está ejecutando actualmente. Los procesadores 80386 y posteriores tienen un IP ampliado de 32 bits, llamado EIP.

En el ejemplo siguiente, el registro CS contiene 25A4 [0]H y el IP contiene 412H. Para encontrar la siguiente instrucción que será ejecutada, el procesador combina las direcciones en el CS y el IP:

Segmento de dirección en el registro CS: 25A40H

Desplazamiento de dirección en el registro IP: + 412H

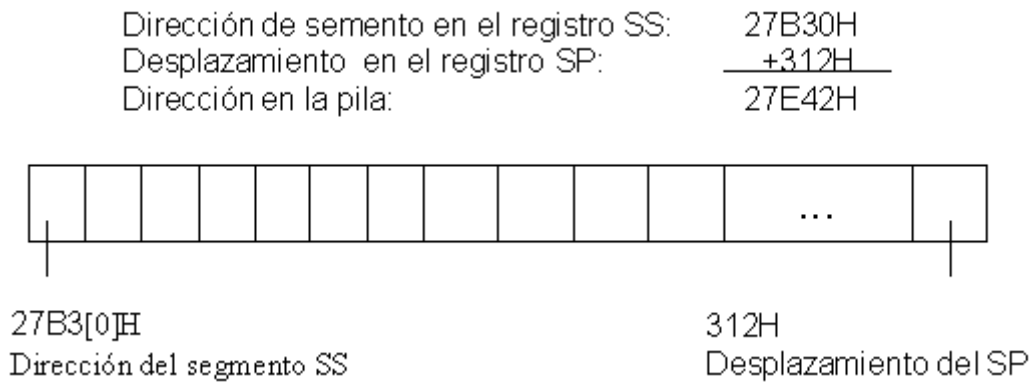
Dirección de la siguiente instrucción: 25E52H

Registros Apuntadores.

Los registros SP (apuntador de la pila) Y BP (apuntador de base) están asociados con el registro SS y permiten al sistema accesar datos en el segmento de la pila.

Registro SP. El apuntador de la pila de 16 bits está asociado con el registro SS y proporciona un valor de desplazamiento que se refiere a la palabra actual que está siendo procesada en la pila. Los procesadores 80386 y posteriores tienen un apuntador de pila de 32 bits, el registro ESP. El sistema maneja de forma automática estos registros.

En el ejemplo siguiente, el registro SS contiene la dirección de segmento 27B3 [0]H y el SP el desplazamiento 312H. Para encontrar la palabra actual que está siendo procesada en la pila, la computadora combina las direcciones en el SS y el SP:



Registro BP. El BP de 16 bits facilita la referencia de parámetros, los cuales son datos y direcciones transmitidos vía pila. Los procesadores 80386 y posteriores tienen un BP ampliado de 32 bits llamado el registro EBP.

Registros Índice.

Los registros SI y DI están disponibles para direccionamiento indexado y para sumas y restas.

Registro SI. El registro índice fuente de 16 bits es requerido por algunas operaciones con cadenas (de caracteres). En este contexto, el SI esta asociado con el registro DS. Los procesadores 80386 y posteriores permiten el uso de un registro ampliado de 32 bits, el ESI.

Registro DI. El registro índice destino también es requerido por algunas operaciones con cadenas de caracteres. En este contexto, el DI esta asociado con el registro ES. Los procesadores 80386 y posteriores permiten el uso de un registro ampliado de 32 bits, el EDI.

Registro de Banderas.

De los 16 bits del registro de banderas, nueve son comunes a toda la familia de procesadores 8086, y sirven para indicar el estado actual de la máquina y el resultado del procesamiento. Muchas instrucciones que piden comparaciones y aritmética cambian el estado de las banderas, algunas cuyas instrucciones pueden realizar pruebas para determinar la acción subsecuente. En resumen, los bits de las banderas comunes son como sigue:

OF (Overflow, desbordamiento). Indica desbordamiento de un bit de orden alto (más a la izquierda) después de una operación aritmética.

DF (dirección). Designa la dirección hacia la izquierda o hacia la derecha para mover o comparar cadenas de caracteres.

IF (interrupción). Indica que una interrupción externa, como la entrada desde el teclado, sea procesada o ignorada.

TF (trampa). Permite la operación del procesador en modo de un paso. Los programas depuradores, como el DEBUG, activan esta bandera de manera que usted pueda avanzar en la ejecución de una sola instrucción a un tiempo, para examinar el efecto de esa instrucción sobre los registros de memoria.

SF (signo). Contiene el signo resultante de una operación aritmética (0 = positivo y 1 = negativo).

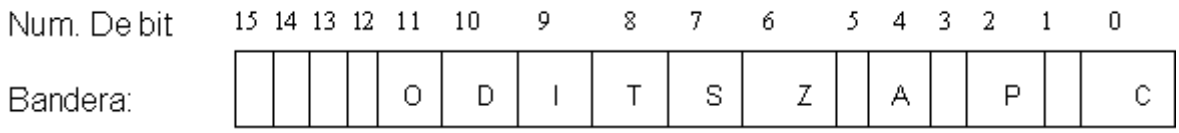
ZF (cero). Indica el resultado de una operación aritmética o de comparación (0 = resultado diferente de cero y 1 = resultado igual a cero).

AF (acarreo auxiliar). Contiene un acarreo externo del bit 3 en un dato de 8 bits para aritmética especializada.

PF (paridad). Indica paridad par o impar de una operación en datos de 8 bits de bajo orden (más a la derecha).

CF (acarreo). Contiene el acarreo de orden más alto (más a la izquierda) después de una operación aritmética; también lleva el contenido del ultimo bit en una operación de corrimiento o de rotación.

Las banderas están en el registro de banderas en las siguientes posiciones:



Las banderas más importantes para la programación en ensamblador son O, S, Z y C, para operaciones de comparación y aritméticas, y D para operaciones de cadenas de caracteres. Los procesadores 80286 y posteriores tienen algunas banderas usadas para propósitos internos, en especial las que afectan al modo protegido. Los procesadores 80286 y posteriores tienen un registro extendido de banderas conocido como Eflags.

SEGMENTO

Un segmento es un área especial en un programa que inicia en un límite de un párrafo, esto es, en una localidad de regularmente divisible entre 16, o 10 hexadecimal. Aunque un segmento puede estar ubicado casi en cualquier lugar de la memoria y, en modo real, puede ser hasta de 64K, solo necesita tanto espacio como el programa requiera para su ejecución.

Un segmento en modo real puede ser de hasta 64K. Se puede tener cualquier número de segmentos; para direccionar un segmento en particular basta cambiar la dirección en el registro del segmento apropiado. Los tres segmentos principales son los segmentos de código, de datos y de la pila.

Segmento de código.

El segmento de código (CS) contiene las instrucciones de máquina que son ejecutadas por lo común la primera instrucción ejecutable está en el inicio del segmento, y el sistema operativo enlaza a esa localidad para iniciar la ejecución del programa. Como su nombre indica, el registro del CS direcciona el segmento de código. Si su área de código requiere más de 64K, su programa puede necesitar definir más de un segmento de código.

Segmento de datos.

El segmento de datos (DS) contiene datos, constantes y áreas de trabajo definidos por el programa. El registro DS direcciona el segmento de datos. Si su área de datos requiere más de 64K, su programa puede necesitar definir más de un segmento de datos.

Segmento de pila.

En términos sencillos, la pila contiene los datos y direcciones que usted necesita guardar temporalmente o para uso de sus "llamadas" subrutinas. El registro de segmento de la pila (SS) direcciona el segmento de la pila.

LIMITES DE LOS SEGMENTOS.

Los registros de segmentos contienen la dirección inicial de cada segmento. La figura 3.1 presenta un esquema de los registros CS, DS y SS; los registros y segmentos no necesariamente están en el orden mostrado. Otros registros de segmentos son el ES (segmento extra) y, en los procesadores 80386 y posteriores, los registros FS y GS, que contienen usos especializados.

Coma ya dijimos, un segmento inicia en un límite de párrafo, que es una dirección por lo común divisible entre el 16 decimal o 10 hexadecimal. Suponga que un segmento de datos inicia en la localidad de memoria 045F0H.

Ya que en este y todos los demás casos el ultimo dígito hexadecimal de la derecha es cero, los diseñadores de computadora decidieron que sería innecesario almacenar el dígito cero en el registro del segmento. Así, 045F0H se almacena como 045F, con el cero de la extrema derecha sobrentendido. En donde sea apropiado, el texto indica al cero de la derecha con corchetes, como 045F[0].

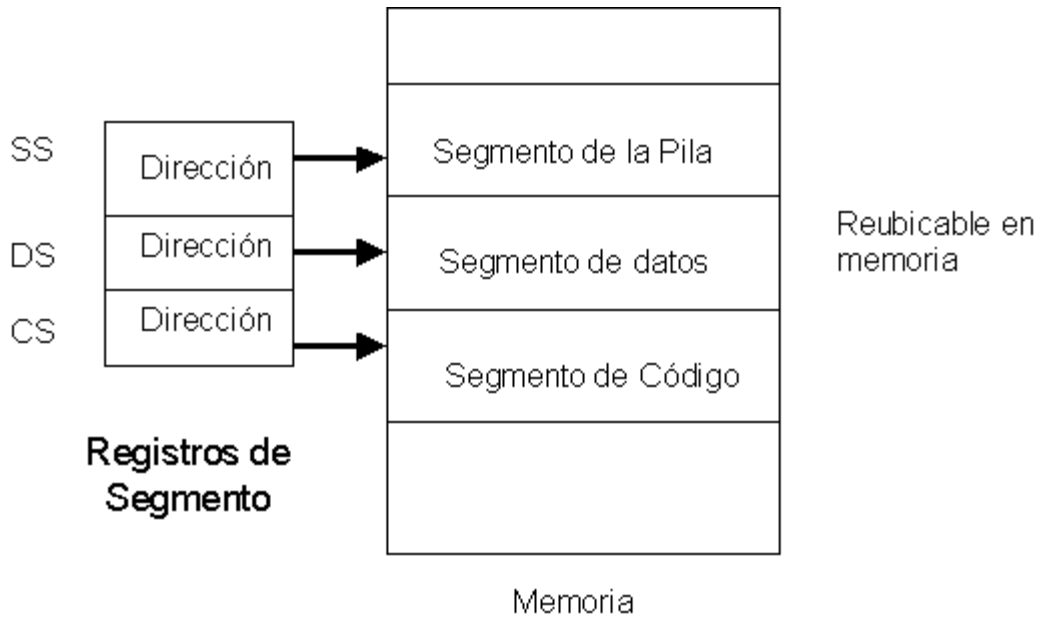


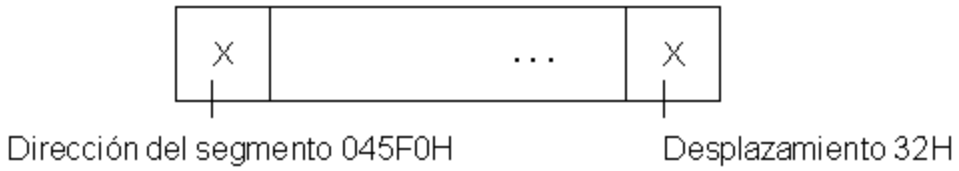
Figura 3.1. Segmentos y registros.

DESPLAZAMIENTO

En un programa, todas la localidades de memoria están referidas a una dirección inicial de segmento. La distancia en bytes desde la dirección del segmento se define como el desplazamiento (offset).

Un desplazamiento de dos bytes (16 bits) puede estar en el rango de 0000H hasta FFFFH, o bien, desde cero hasta 65, 535. Así el primer byte del segmento de código tiene un desplazamiento 00, el segundo byte tiene un desplazamiento 01, etc. hasta el desplazamiento 65, 535. Para referir cualquier dirección de memoria en un segmento, el procesador combina la dirección del segmento en un registro de segmento con un valor de desplazamiento.

En el ejemplo siguiente, el registro DS contiene la dirección de segmento del segmento de datos en 045F[0]H y una instrucción hace referencia a una localidad con un desplazamiento de 0032H bytes dentro del segmento de datos.



Por lo tanto, la localidad real de memoria del byte referido por la instrucción es 04622H;

Dirección del segmento DS: 045F0H
Desplazamiento: +0032H
Dirección real: 04622H

Note que un programa tiene uno o más segmentos, los cuales pueden iniciar casi en cualquier lugar de memoria, variar en tamaño y estar en cualquier orden.

METODOS DE DIRECCIONAMIENTO

El campo de operación de una instrucción especifica la operación que se va a ejecutar. Esta operación debe realizarse sobre algunos datos almacenados en registros de computadora o en palabras de memoria. La manera en que eligen los operandos durante la ejecución del programa depende del modo de direccionamiento de la instrucción. El modo de direccionamiento especifica una regla para interpretar o modificar el campo de dirección de la instrucción antes de que se haga la referencia real al operando. Las computadoras utilizan técnicas de modo de direccionamiento para acomodar una o las dos siguientes consideraciones:

1. Proporcionar al usuario versatilidad de programación al ofrecer facilidades como apuntadores a memoria, contadores para control de ciclo, indexación de datos y reubicación de datos.
2. Reducir la cantidad de bits en el campo de direccionamiento de la instrucción.

La disponibilidad de los modos de direccionamiento proporciona al programador con experiencia en lenguaje ensamblador la flexibilidad para escribir programas más eficientes en relación con la cantidad de instrucciones y el tiempo de ejecución.

Para comprender los diferentes modos de direccionamiento que se presentaran en esta sección, es imperativo entender el ciclo de operación básico de la

computadora. La unidad de control de una computadora está diseñada para recorrer un ciclo de instrucciones que se divide en tres fases principales:

1. Búsqueda de la instrucción de la memoria.
2. Decodificar la instrucción.
3. Ejecutar la instrucción.

Hay un registro en la computadora llamado contador de programa o PC, que lleva un registro de las instrucciones del programa almacenado en la memoria. Pc contiene la dirección de la siguiente instrucción que se va a ejecutar y se incrementa cada vez que se recupera una instrucción de la memoria. La decodificación realizada en el paso 2 determina la operación que se va a ejecutar, el modo de direccionamiento de la instrucción y la posición de los operandos.

Después la computadora ejecuta la instrucción y regresa al paso 1 para hacer la búsqueda de la siguiente instrucción en secuencia.

En algunas computadoras el modo de direccionamiento de la instrucción se especifica con un código binario distinto, como se hace con el código de operación. Otras computadoras utilizan un código binario único que representa la operación y el modo de la instrucción. Pueden definirse instrucciones con diversos modos de direccionamiento y, en ocasiones, se combinan dos o más modos de direccionamiento en una instrucción.

Aunque la mayoría de los modos de direccionamiento modifican el campo de dirección de la instrucción, hay dos modos que no necesitan el campo de dirección. Son los modos implícito e inmediato.

MODO IMPLICITO.

En este modo se especifican los operandos en forma implícita en la definición de la instrucción. Por ejemplo, la instrucción "complementar acumulador" es una instrucción de modo implícito porque el operando en el registro de acumulador está implícito en la definición de la instrucción. De hecho todas las instrucciones de referencia a registro que utilizan un acumulador son instrucciones de modo implícito.

Las instrucciones de dirección cero en una computadora organizada con pila son instrucciones de modo implícito porque está implícito que los operandos están en la parte superior de la pila.

MODO INMEDIATO.

En este modo se especifica el operando en la instrucción misma. En otras palabras, una instrucción de modo inmediato tiene un campo operando, en lugar de una campo de dirección. Un campo de operando contiene el operando real que se va a usar junto con la operación especificada en la instrucción. Las instrucciones de modo inmediato son útiles para inicializar registros en un valor constante.

Se mencionó antes que el campo de dirección de una instrucción puede especificar una palabra de memoria o un registro de procesador. Cuando el campo de dirección especifica un registro de procesador se dice que la instrucción esta en modo de registro.

MODO DE REGISTRO.

En este modo, los operandos están en registros que residen dentro de la CPU. Se selecciona el registro particular de un campo de registro en la instrucción. Un campo k bits puede especificar cualquiera de 2 a la k registros.

MODO INDIRECTO POR REGISTRO.

En este modo la instrucción especifica un registro en la CPU cuyo contenido proporciona la dirección del operando en la memoria. En otras palabras, el registro seleccionado contiene la dirección del operando en lugar del operando mismo. Antes de utilizar una instrucción de modo indirecto por registro, el programador debe asegurarse de que la dirección de memoria del operando está colocada en el registro del procesador con una instrucción previa. Entonces una referencia al registro es equivalente a especificar una dirección de memoria. La ventaja de una instrucción de modo de registro indirecto es que el campo de dirección de la instrucción utiliza menos bits para seleccionar un registro de los que necesitaría para especificar una dirección de memoria en forma directa.

MODO DE DIRECCIONAMIENTO DIRECTO.

En este modo la dirección efectiva es igual a la parte de dirección de la instrucción. El operando reside en memoria y su dirección la proporciona en forma directa el campo de dirección de la instrucción. En una instrucción de tipo brinco el campo de dirección especifica la dirección de transferencia de control del programa real.

MODO DE DIRECCIONAMIENTO INDIRECTO.

En este modo, el campo de dirección de la instrucción proporciona la dirección en que se almacena la dirección efectiva en la memoria. El control recupera la instrucción de la memoria y utiliza su parte de dirección para accesar la memoria una vez más con el fin de leer la dirección efectiva.

Unos cuantos modos de direccionamiento requieren que el campo de dirección de la instrucción se sume al contenido de un registro específico en la CPU. En estos modos la dirección efectiva se obtiene del cálculo siguiente:

Dirección efectiva = Parte de la instrucción + El contenido de registro CPU.

EL registro de CPU utilizado en el cálculo puede ser el contador de programa, un registro de índice o un registro base. En cualquier caso tenemos un modo de direccionamiento diferente que se utiliza para una aplicación distinta.

MODO DE DIRECCIONAMIENTO INDEXADO.

En este modo el contenido de un registro índice se suma a la parte de dirección de la instrucción para obtener la dirección efectiva. El registro índice es un registro CPU especial que contiene un valor índice. Un campo de dirección de la instrucción define la dirección inicial del arreglo de datos en la memoria. Cada operando del arreglo se almacena en la memoria en relación con la dirección inicial.

La distancia entre la dirección inicial y la dirección del operando es el valor del índice almacenado en el registro de índice. Cualquier operando en el arreglo puede accesarse con la misma instrucción siempre y cuando el registro índice contenga el valor de índice correcto. El registro índice puede incrementarse para facilitar el acceso a operandos consecutivos. Nótese que si una instrucción de tipo índice no incluye un campo de dirección en su formato, la instrucción se convierte al modo de operación de indirecto por registro.

Algunas computadoras dedican un registro de CPU para que funcione exclusivamente como un registro índice. De manera implícita este registro participa cuando se utiliza una instrucción de modo índice. En las computadoras con muchos registros de procesador, cualquiera de los registros de la CPU pueden contener el número de índice. En tal caso, el registro debe estar

especificado en forma explícita en un campo de registro dentro del formato de instrucción.

MODO DE DIRECCIONAMIENTO DE REGISTRO BASE.

En este modo, el contenido de un registro base se suma a la parte de dirección de la instrucción para obtener la dirección efectiva. Esto es similar al modo de direccionamiento indexado, excepto en que el registro se denomina ahora registro base, en lugar de registro índice. La diferencia entre los dos modos está en la manera en que se usan más que en la manera en que se calculan. Se considera que un registro base contiene una dirección base y que el campo de dirección de la instrucción proporciona un desplazamiento en relación con esta dirección base. El modo de direccionamiento de registro base se utiliza en las computadoras para facilitar la localización de los programas en memoria.

CONCEPTO DE INTERRUPCION

Una interrupción es una operación que suspende la ejecución de un programa de modo que el sistema pueda realizar una acción especial. La rutina de interrupción ejecuta y por lo regular regresa el control al procedimiento que fue interrumpido, el cual entonces reasume su ejecución.

TABLA DE SERVICIO DE INTERRUPCION.

Cuando la computadora se enciende, el BIOS y el DOS establecen una tabla de servicios de interrupción en las localidades de memoria 000H-3FFH. La tabla

permite el uso de 256 (100H) interrupciones, cada una con un desplazamiento:segmento relativo de cuatro bytes en la forma IP:CS.

El operando de una instrucción de interrupción como INT 05H identifica el tipo de solicitud. Como existen 256 entradas, cada una de cuatro bytes, la tabla ocupa los primeros 1, 024 bytes de memoria, desde 000H hasta 3FFH. Cada dirección en la tabla relaciona a una ruina de BIOS o del DOS para un tipo específico de interrupción. Por lo tanto los bytes 0-3 contienen la dirección para la interrupción 0, los bytes 4-7 para la interrupción 1, y así sucesivamente:

INT 00H	INT 01H	INT 02H	INT 03H	INT 04H	INT 05H	INT 06H	. . .
IP:CS	IP:CS	IP:CS	IP:CS	IP:CS	IP:CS	IP:CS	. . .
00H	04H	08H	0CH	10H	14H	18H	. . .

EVENTOS DE UNA INTERRUPCION.

Una interrupción guarda en la pila el contenido del registro de banderas, el CS, y el IP. por ejemplo, la dirección en la tabla de INT 05H (que imprime la que se encuentra en la pantalla cuando el usuario presiona Ctrl + PrtSC) es 0014H (05H x 4 = 14H). La operación extrae la dirección de cuatro bytes de la posición 0014H y almacena dos bytes en el IP y dos en el CS.

La dirección CS:IP entonces apunta al inicio de la rutina en el área del BIOS, que ahora se ejecuta. La interrupción regresa vía una instrucción IRET (regreso de interrupción), que saca de la pila el IP, CS y las banderas y regresa el control a la instrucción que sigue al INT.

TIPOS DE INTERRUPCIONES.

Las interrupciones se dividen en dos tipos las cuales son: Externas y Internas. Una interrupción externa es provocada por un dispositivo externo al procesador. Las dos líneas que pueden señalar interrupciones externas son la línea de interrupción no enmascarable (NMI) y la línea de petición de interrupción (INTR).

La línea NMI reporta la memoria y errores de paridad de E/S. El procesador siempre actúa sobre esta interrupción, aun si emite un CLI para limpiar la bandera de interrupción en un intento por deshabilitar las interrupciones externas. La línea INTR reporta las peticiones desde los dispositivos externos, en realidad, las interrupciones 05H a la 0FH, para cronometro, el teclado, los puertos seriales, el disco duro, las unidades de disco flexibles y los puertos paralelos.

Una interrupción interna ocurre como resultado de la ejecución de una instrucción INT o una operación de división que cause desbordamiento, ejecución en modo de un paso o una petición para una interrupción externa, tal como E/S de disco. Los programas por lo común utilizan interrupciones internas, que no son enmascarables, para accesar los procedimientos del BIOS y del DOS.

INTERRUPCION DE BIOS.

El BIOS contiene un extenso conjunto de rutinas de entrada/salida y tablas que indican el estado de los dispositivos del sistema. El dos y los programas usuarios pueden solicitar rutinas del BIOS para la comunicación con los dispositivos conectados al sistema. El método para realizar la interfaz con el BIOS es el de las interrupciones de software. A continuación se listan algunas interrupciones del BIOS.

INT 00H: División entre cero. Llamada por un intento de dividir entre cero. Muestra un mensaje y por lo regular se cae el sistema.

INT 01H: Un solo paso. Usado por DEBUG y otros depuradores para permitir avanzar por paso a través de la ejecución de un programa.

INT 02H: Interrupción no enmascarare. Usada para condiciones graves de hardware, tal como errores de paridad, que siempre están habilitados. Por lo tanto un programa que emite una instrucción CLI (limpiar interrupciones) no afecta estas condiciones.

INT 03H: Punto de ruptura. Usado por depuración de programas para detener la ejecución.

INT 04H: Desbordamiento. Puede ser causado por una operación aritmética, aunque por lo regular no realiza acción alguna.

INT 05H: Imprime pantalla. Hace que el contenido de la pantalla se imprima. Emita la INT 05H para activar la interrupción internamente, y presione las teclas Cltr + PrtSC para activarla externamente. La operación permite interrupciones y guarda la posición del cursor.

INT 08H: Sistema del cronometro. Una interrupción de hardware que actualiza la hora del sistema y (si es necesario) la fecha. Un chip temporizador programable genera una interrupción cada 54.9254 milisegundos, casi 18.2 veces por segundo.

INT 09H: Interrupción del teclado. Provocada por presionar o soltar una tecla en el teclado.

INT OBH, INT OCH: Control de dispositivo serial. Controla los puertos COM1 y COM2, respectivamente.

INT 0DH, INT OFH: Control de dispositivo paralelo. Controla los puertos LPT1 y LPT2, respectivamente.

INT 0EH: Control de disco flexible. Señala actividad de disco flexible, como la terminación de una operación de E/S.

INT 10H: Despliegue en vídeo. Acepta el número de funciones en el AH para el modo de pantalla, colocación del cursor, recorrido y despliegue.

INT 11H: Determinación del equipo. Determina los dispositivos opcionales en el sistema y regresa el valor en la localidad 40:10H del BIOS al AX. (A la hora de encender el equipo, el sistema ejecuta esta operación y almacena el AX en la localidad 40:10H).

INT 12H: Determinación del tamaño de la memoria. En el AX, regresa el tamaño de la memoria de la tarjeta del sistema, en términos de kilobytes contiguos.

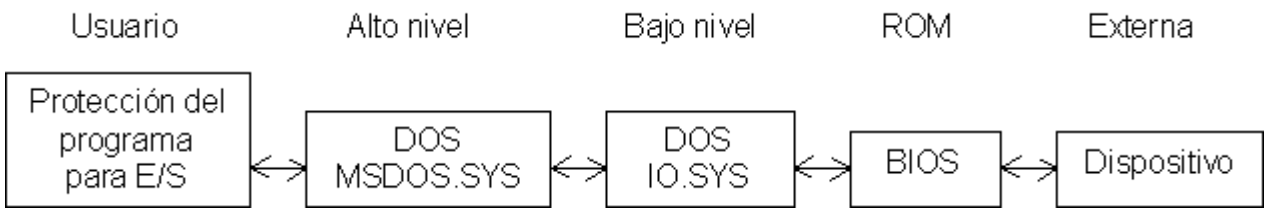
INT 13H: Entrada/salida de disco. Acepta varias funciones en el AH para el estado del disco, sectores leídos, sectores escritos, verificación, formato y obtener diagnóstico.

INTERRUPCION DEL DOS.

Los dos módulos del DOS, IO.SYS y MSDOS.SYS, facilitan el uso del BIOS. Ya que proporcionan muchas de las pruebas adicionales necesarias, las operaciones del DOS por lo general son más fáciles de usar que sus contrapartes del BIOS y por lo común son independientes de la máquina.

IO.SYS es una interfaz de nivel bajo con el BIOS que facilita la lectura de datos desde la memoria hacia dispositivos externos.

MSDOS.SYS contiene un administrador de archivos y proporciona varios servicios. Por ejemplo, cuando un programa usuario solicita la INT 21H, la operación envía información al MSDOS.SYS por medio del contenido de los registros. Para completar la petición, MSDOS.SYS puede traducir la información a una o más llamadas a IO.SYS, el cual a su vez llama al BIOS. Las siguientes son las relaciones implícitas:



INTERUPCIONES DEL DOS.

Las interrupciones desde la 20H hasta la 3FH están reservadas para operaciones del DOS. A continuación se mencionan algunas de ellas.

INT 20H: Termina programa. Finaliza la ejecución de un programa .COM, restaura las direcciones para Cltr + Break y errores críticos, limpia los bufer de registros y regresa el control al DOS. Esta función por lo regular seria colocada en el procedimiento principal y al salir del, CS contendría la dirección del PSP. La terminación preferida es por medio de la función 4CH de la INT 21H.

INT 21H: Petición de función al DOS. La principal operación del DOS necesita una función en el AH.

INT 22H: Dirección de terminación. Copia la dirección de esta interrupción en el PSP del programa (en el desplazamiento 0AH) cuando el DOS carga un programa para ejecución. A la terminación del programa, el DOS transfiere el control a la dirección de la interrupción. Sus programas no deben de emitir esta interrupción.

INT 23H: Dirección de Cltr + Break. Diseñada para transferir el control a una rutina del DOS (por medio del PSP desplazamiento 0EH) cuando usted presiona Ctlt + Break o Ctlr + c. La rutina finaliza la ejecución de un programa o de un archivo de procesamiento por lotes. Sus programas no deben de emitir esta interrupción.

INT 24H: Manejador de error crítico. Usada por el dos para transferir el control (por medio del PSP desplazamiento 12H) cuando reconoce un error critico (a veces

una operación de disco o de la impresora).Sus programas no deben de emitir esta interrupción.

INT 25H: Lectura absoluta de disco. Lee el contenido de uno o más sectores de disco.

INT 26H: Escritura absoluta de disco. Escribe información desde la memoria a uno o más sectores de disco.

INT 27H: Termina pero permanece residente (reside en memoria). Hace que un programa .COM al salir permanezca residente en memoria.

INT 2FH: Interrupción de multiplexion. Implica la comunicación entre programas, como la comunicación del estado de un spooler de la impresora, la presencia de un controlador de dispositivo o un comando del DOS tal como ASSIGN o APPEND.

INT 33H: Manejador del ratón. Proporciona servicios para el manejo del ratón.

ELEMENTOS BASICOS

COMENTARIOS EN LENGUAJE ENSAMBLADOR.

El uso de comentarios a lo largo de un programa puede mejorar su claridad, en especial en lenguaje ensamblador, donde el propósito de un conjunto de instrucciones con frecuencia no es claro. Un comentario empieza con punto y coma (;) y, en donde quiera que lo codifique, el ensamblador supone que todos los caracteres a la derecha de esa línea son comentarios. Un comentario puede contener cualquier carácter imprimible, incluyendo el espacio en blanco. Un comentario puede aparecer solo en una línea o a continuación de una instrucción en la misma línea, como lo muestran los dos ejemplos siguientes:

1. ; Toda esta línea es un comentario.

2. ADD AX, BX ; Comentario en la misma línea que la instrucción.

Ya que un comentario aparece solo en un listado de un programa fuente en ensamblador y no genera código de maquina, puede incluir cualquier cantidad de comentarios sin afectar el tamaño o la ejecución del programa ensamblado. Otra manera de proporcionar comentarios es por medio de la directiva COMMENT.

PALABRAS RESERVADAS.

Ciertas palabras en lenguaje ensamblador están reservadas para sus propósitos propios, y son usadas solo bajo condiciones especiales. Por categorías, las palabras reservadas incluyen:

Instrucciones, como MOV y ADD, que son operaciones que la computadora puede ejecutar.

Directivas como END o SEGMENT, que se emplean para proporcionar comandos al ensamblador.

Operadores, como FAR y SIZE, que se utilizan en expresiones.

Símbolos predefinidos, como @Data y @Model, que regresan información a su programa.

El uso de una palabra reservada para un propósito equivocado provoca que el ensamblador genere un mensaje de error.

Ver palabras reservadas.

IDENTIFICADORES.

Un identificador es un nombre que se aplica a elementos en el programa. Los dos tipos de identificadores son: nombre, que se refiere a la dirección de un elemento de dato. y etiqueta, que se refiere a la dirección de una instrucción. Las mismas reglas se aplican tanto para los nombres como para las etiquetas. Un identificador puede usar los siguientes caracteres:

1.- Letras del alfabeto:	Desde la A hasta la Z
2.- Dígitos:	Desde el 0 al 9 (no puede ser el primer carácter)
3.- Caracteres especiales	Signo de interrogación (?)
	Subrayado (_)
	Signo de pesos ($)
	Arroba (@)
	Punto (.) (no puede ser el primer carácter)

El primer carácter de un identificador debe ser una letra o un carácter especial, excepto el punto. Ya que el ensamblador utiliza algunos símbolos especiales en palabras que inician con el símbolo @, debe evitar usarlo en sus definiciones.

El ensamblador trata las letras mayúsculas y minúsculas como iguales. La longitud máxima de un identificador es de 31 caracteres (247 desde el MASM 6.0). Ejemplos de nombres validos son COUNT, PAGE25 y $E10. Se recomienda que los nombres sean descriptivos y con significado. Los nombres de registros, como AX, DI y AL, están reservados para hacer referencia a esos mismos registros. En consecuencia, en una instrucción tal como:

ADD AX, BX

el ensamblador sabe de forma automática que AX y BX se refieren a los registros. Sin embargo, en una instrucción como:

MOV REGSAVE, AX

el ensamblador puede reconocer el nombre REGSAVE solo si se define en algún lugar del programa.

INSTRUCCIONES.

Un programa en lenguaje ensamblador consiste en un conjunto de enunciados. Los dos tipos de enunciados son:

1. Instrucciones, tal como MOV y ADD, que el ensamblador traduce a código objeto.

2. Directivas, que indican al ensamblador que realiza una acción especifica, como definir un elemento de dato.

A continuación esta el formato general de un enunciado, en donde los corchetes indican una entrada opcional:

[identificador] operación [operando(s)] [;comentarios]

Un identificador (si existe), una operación y un operando (si existe) están separados por al menos un espacio en blanco o un carácter tabulador. Existe un máximo de 132 caracteres en una línea (512 desde el MASM 6.0), aunque la mayoría de los programadores prefiere permanecer en los 80 caracteres ya que es el numero máximo que cabe en la pantalla. A continuación se presentan dos ejemplos de enunciados:

IDENTIFICADOR	**OPERACION**	**OPERANDO**	**COMENTARIO**
Directiva: COUNT	DB	1	;Nom, Op, Operando
Instrucción:	MOV	AX, 0	;Operación, 2 Operand

Identificador, operación y operando pueden empezar en cualquier columna. Sin embargo, si de manera consistente se inicia en la misma columna para estas tres entradas se hace un programa mas legible.

IDENTIFICADOR

Como ya se explico, el termino nombre se aplica al nombre de un elemento o directiva definida, mientras que el termino etiqueta se aplica al nombre de una instrucción.

OPERACION

La operación, que debe ser codificada, es con mayor frecuencia usada para la definición de áreas de datos y codificación de instrucciones. Para un elemento de datos, una operación como DB o DW define un campo, área de trabajo o constante.

OPERANDO

El operando (si existe) proporciona información para la operación que actúa sobre el. Para un elemento de datos, el operando identifica su valor inicial. Por ejemplo, en la definición siguiente de un elemento de datos llamado COUNTER, la operación DB significa "definir byte", y el operando inicializa su contenido con un valor cero:

NOMBRE	OPERACION	OPERANDO	COMENTARIO
COUNTER	DB	0	;Define un byte con el valor 0

Para una instrucción, un operando indica en donde realizar la acción. Un operando de una instrucción puede tener una, dos o tal vez ninguna entrada. Aquí están tres ejemplos:

OPERACION	OPERANDO	COMENTARIO	OPERANDO
RET		;Regresa	Ninguno
INC	CX	;Incrementa CX	Uno
ADD	AX, 12	;Suma 12 al AX	Dos

DIRECTIVAS PARA LISTAR: PAGE Y TITLE

La directiva PAGE y TITLE ayudan a controlar el formato de un listado de un programa en ensamblador. Este es su único fin, y no tienen efecto sobre la ejecución subsecuente del programa.

PAGE. Al inicio de un programa, la directiva PAGE designa el numero máximo de líneas para listar en una pagina y el numero máximo de caracteres en una línea. Su formato general es:

PAGE [longitud][, ancho]

El ejemplo siguiente proporciona 60 líneas por pagina y 132 caracteres por línea:

PAGE 60, 132

El numero de líneas por pagina puede variar desde 10 hasta 255, mientras que el numero de caracteres por línea desde 60 hasta 132. La omisión de PAGE causa que el ensamblador tome PAGE 50, 80.

TITLE. Se puede emplear la directiva TITLE para hacer que un titulo para un programa se imprima en la línea 2 de cada pagina en el listado del programa. Puede codificar TITLE de una vez, al inicio del programa. Su formato general es:

TITLE Texto.

Para el operando texto, una técnica recomendada es utilizar el nombre del programa como se registra en el disco. Por ejemplo:

TITLE Prog1 Mi primer programa en ensamblador

DIRECTIVA SEGMENT

Un programa ensamblado en formato .EXE consiste en uno o mas segmentos. Un segmento de pila define el almacén de la pila, un segmento de datos define los elementos de datos y un segmento de código proporciona un código ejecutable. Las directivas para definir un segmento, SEGMENT y ENDS tienen el formato siguiente:

NOMBRE	OPERACION	OPERANDO	COMENTARIO
Nombre	SEGMENT	[Opciones]	;inicia el segmento
.			
.			
.			
Nombre	ENDS		;Fin del segmento

El enunciado SEGMENT define el inicio de un segmento. El nombre del segmento debe estar presente, ser único y cumplir las convenciones para nombres del lenguaje. EL enunciado ENDS indica el final del segmento y contiene el mismo nombre del enunciado SEGMENT. El tamaño máximo de un segmento es de 64K. El operando de un enunciado SEGMENT puede tener tres tipos de opciones: alineación, combinar y clase, codificadas en este formato:

nombre SEGMENT alineación combinar ' clase '

TIPO ALINEACION. La entrada alineación indica el limite en el que inicia el segmento. Para el requerimiento típico, PARA, alinea el segmento con el limite de un párrafo, de manera que la dirección inicial es divisible entre 16, o 10H. En ausencia de un operando hace que el ensamblador por omisión tome PARA.

TIPO COMBINAR. La entrada combinar indica si se combina el segmento con otros segmentos cuando son enlazados después de ensamblar. Los tipos de combinar son STACK, COMMON, PUBLIC y la expresión AT. Por ejemplo, el segmento de la pila por lo común es definido como:

nombre SEGMENT PARA STACK

Puede utilizar PUBLIC y COMMON en donde tenga el propósito de combinar de forma separada programas ensamblados cuando los enlaza. En otros casos, donde un programa no es combinado con otros, puede omitir la opción o codificar NONE.

TIPO CLASE. La entrada clase, encerrada entre apóstrofos, es utilizada para agrupar segmentos cuando se enlazan. Se utiliza la clase 'code' para el segmento

de códigos, 'data' por segmento de datos y 'stack' para el segmento de la pila. El ejemplo siguiente define un segmento de pila con tipos alineación, combinar y clase:

```
nombre    SEGMENT    PARA    STACK    'Stack'
```

DIRECTIVA ASSUME.

Un programa utiliza el registro SS para direccionar la pila, al registro DS para direccionar el segmento de datos y el registro CS para direccionar el segmento de código. Para este fin, usted tiene que indicar al ensamblador el propósito de cada segmento en el programa. La directiva para este propósito es ASSEME, codificada en el segmento de código como sigue:

```
OPERACION                    OPERANDO

ASSUME        SS:nompila, DS:nomsegdatos, CS: nomsegcodigo,. . .
```

Los operandos pueden aparecer en cualquier orden. Al igual que otras directivas, ASSUME es solo un mensaje que ayuda al ensamblador a convertir código simbólico a código maquina; aun puede tener que codificar instrucciones que físicamente cargan direcciones en registros de segmentos en el momento de la ejecución.

```
1                              PAGE                    60,132
2              TITLE P04ASM1 ESTRUCTURA DE UN PROGRAMA .EXE
```

```
3;--------------------------------------------------------------------------
4          STACKSG          SEGMENT          PARA          STACK          'Stack'
5                                                                                ...
6                                    STACKSG                                     ENDS
7;--------------------------------------------------------------------------
8          DATASG           SEGMENT          PARA          'Data'
9                                                                                ...
10                                   DATASG                                      ENDS
11;-------------------------------------------------------------------------
12         CODESG           SEGMENT          PARA          'Code'
13              BEGIN                PROC                FAR
14              ASSUME          SS:STACKSG,     DS:DATASG,CS:CODESG
15            MOV          AX, DATASG      ;Obtiene la dirección del segmento
de                                                                               datos
16            MOV          DS,  AX              ;Almacena dirección en DS
17                                                                               ...
18                MOV                AX,          4C00H              ;Peticion
19                INT                21H                 ;Salida al DOS
20                                   BEGIN                                       ENDP
21                                   CODESG                                      ENDS
22                                       END                                     BEGIN
```

DIRECTIVAS SIMPLIFICADAS DE SEGMENTOS

Los ensambladores de MicroSoft y de Borland proporcionan algunas formas abreviadas para definir segmentos. Para usar estas abreviaturas, inicialice el modelo de memoria antes de definir algún segmento. El formato general (incluyendo el punto inicial) es:

.MODEL modelo de memoria

El modelo de memoria puede ser TINY, SMALL, MEDIUM, COMPACT o LARGE. Los requisitos para cada modelo son:

MODELO	NUMERO DE SEGMENTOS DE CODIGO	NUMERO DE SEGMENTOS DE DATOS
TINY	*	*
SMALL	1	1
MEDIUM	MAS DE 1	1
COMPACT	1	MAS DE 1
LARGE	MAS DE 1	MAS DE 1

Puede utilizar cualquiera de estos modelos para un programa autónomo (esto es, un programa que no este enlazado con algún otro). El modelo TINY esta destinado para uso exclusivo de programas .COM, los cuales tienen sus datos, código y pila en un segmento. El modelo SMALL exige que el código quepa en un segmento de 64K y los datos en otro segmento de 64K. La directiva .MODELL genera automáticamente el enunciado ASSUME necesario.

Los formatos generales (incluyendo el punto inicial) para las directivas que define los segmentos de la pila, de datos y de código son:

.STACK [tamaño]
.DATA
.CODE [nombre]

Cada una de estas directivas hace que el ensamblador genere el enunciado SEGMENT necesario y su correspondiente ENDS. Los nombres por omisión de los segmentos (que usted no tiene que definir) son STACK, DATA y TEXT (para el segmento de código).

La figura 4.3 proporciona un ejemplo haciendo uso de las directivas simplificadas de segmento.

```
                                page                            60,132
TITLE       P04ASM2     (EXE)       Operaciones     de      mover       y       sumar
;-------------------------------------------------------------------------
                                .MODEL                          SMALL
            .STACK          64          ;Se         define      la          pila
          .DATA       ;Se definen los datos

FLDA                            DW                              250
FLDB                            DW                              125
FLDC                            DW                              ?
;-------------------------------------------------------------------------

            .CODE                   ;Se     define      el      segmento    de      código
BEGIN                           PROC                            FAR
          MOV       AX, @data   ;Se asigna la dirección de DATASG (Prog. anterior)

              MOV           AX,     FLDA        ;Mover      0250        a       AX
              ADD           AX,     FLDB        ;Sumar      0125        a       AX
          MOV       FLDC, AX    ;Almacenar suma en FLDC

              MOV           AX,     4C00H       ;Salida     a           DOS
                                INT                             21H
BEGIN       ENDP                        ;Fin        de          procedimiento
          END       BEGIN       ;Fin de programa
```

TRANSFERENCIA DE DATOS.

La instrucción de transferencia de datos por excelencia es:

MOV destino, fuente

Entendiendo por fuente el contenido que se va a transferir a una determinada zona o registro de memoria denominada destino.

Esta instrucción, por tanto, nos va a permitir transferir información entre registros y memoria, memoria y registros y entre los propios registros utilizando alguno de los diferentes modos de direccionamiento. Con la instrucción MOV diremos que se pueden realizar todo tipo de movimientos teniendo en cuenta las siguientes restricciones:

1.- No se puede realizar una transferencia de datos entre dos posiciones de memoria
directamente, por esta razón, siempre que queramos efectuarlas tendremos que utilizar un registro intermedio que haga de puente.

Por ejemplo, para hacer la operacion DATO1 <-- DATO2,

la instrucción MOV DATO2,DATO1

sería incorrecta. Lo que sí sería correcto sería utilizar el registro DX, u otro, como puente y hacer:

MOV DX,DATO1

MOV DATO2,DX

2.- Tampoco se puede hacer una transferencia directa entre dos registros de segmento. Por eso, como en el caso anterior, si fuera preciso se utilizaría un registro como puente.

3.- Asimismo, tampoco se puede cargar en los registros de segmento un dato utilizando direccionamiento inmediato, es decir, una constante, por lo que también habrá que recurrir a un registro puente cuando sea preciso.

Una instrucción util pero no imprescindible es:

```
XCHG DATO1, DATO2
```

que intercambia los contenidos de las posiciones de memoria o registros representadospor DATO1 y DATO2.

Por ejemplo, si queremos intercambiar los contenidos de los registros AX y BX, podemos hacer:

```
MOV AUX, AX
MOV AX, BX
MOV BX, AUX
```

en donde AUX es una variable auxiliar que hace de puente, o simplemente utilizar:

```
XCHG AX, BX
```

Las restricciones que presenta esta operación es que no se pueden efectuar intercambios directamente entre posiciones de memoria ni tampoco entre registros de segmento.

La instrucción XLAT tabla carga en el registro AL el contenido de la posición [BX][AL], en donde el registro BX ha de apuntar al comienzo de una tabla. Dichio de otra manera, AL hace de índice de la tabla y de almacén destino del contenido de la tabla.

Por ejemplo, el siguiente programa:

```
DATOS                                   SEGMENT
                TABLA           DB      2,3,5,8,16,23
DATOS                                   ENDS
CODIGO                                  SEGMENT
    MOVE BX, OFFSET TABLA ; Inicializa BX con la dirección donde comienza la tabla
                MOVE            AL,     5
                        XLAT            TABLA
CODIGO ENDS
```

hace que al final el contenido de AL se 16 ya que es el 5to. elemento de la tabla y AL antes de XLAT TABLA contenia el valor 5.

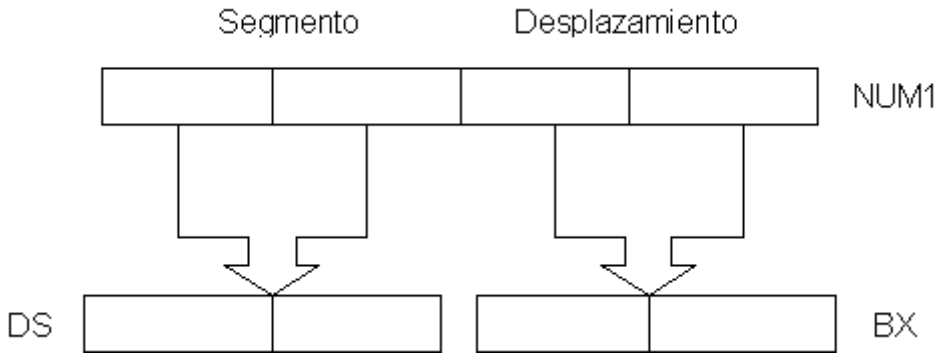
Para finalizar con las instrucciones de transferencia veremos un grupo de tres instrucciones:

- LEA o cargar dirección efectiva.
- LDS o cargar el puntero en DS.
- LES o cargar el puntero en ES.

denominadas de transferencia de direcciones.

La primera, LEA, carga el desplazamiento u OFFSET correspondiente al operando fuente en el operando destino. Por ejemplo, la instrucción MOVE BX, OFFSET TABLA del ejemplo anterior sería equivalente a LEA BX, TABLA.

La segunda, LDS, se utiliza para cargar el valor del segmento de una variable en el registro DS y el desplazamiento correspondiente en el registro o posición de memoria indicada en la instrucción. Por ejemplo, la instrucción LDS BX, NUM1 haría esquemáticamente lo siguiente:



La tercera y ultima de las instrucciones, LES, es similar a LDS, con la única salvedad de que el valor del segmento se carga sobre el registro de segmento ES en vez del DS.

SUMA Y RESTA

Las instrucciones ADD y SUB realizan sumas y restas sencillas de datos binarios. Los números binarios negativos están representados en la forma de complemento a dos: Invierta todos los bits del numero positivo y sume 1. Los formatos generales para las instrucciones ADD y SUB son:

[etiqueta]	ADD/SUB	{registro, registro}
[etiqueta]	ADD/SUB	{memoria, registro}
[etiqueta]	ADD/SUB	{registro, memoria}
[etiqueta]	ADD/SUB	{registro, inmediato}
[etiqueta]	ADD/SUB	{memoria, inmediato}

Como en otras instrucciones, no existen operaciones directas de memoria a memoria. El ejemplo siguiente utiliza el registro AX para sumar WORDA a WORDB:

```
WORDA    DW    123    ;Define WORDA
WORDB    DW    25     ;Define WORDB
                .          .          .
         MOV   AX,  WORDA    ;Mueve WORDA al AX
```

```
        ADD     AX,     WORDB       ;Suma WORDB al AX
    MOV WORDB, AX     ;Mueve AX a WORDB
```

La figura 6.1. proporciona ejemplos de ADD y SUB para el procesamiento de valores en un byte y en una palabra. El procedimiento B10ADD utiliza ADD para procesar bytes y el procedimiento C10SUB utiliza SUB para procesar palabras.

```
TITLE   P13ADD (COM) Operaciones ADD y SUB
                        .MODEL      SMALL
                                    .CODE
                          ORG       100H
BEGIN:          JMP       SHORT     MAIN
;-------------------------------------------------------------------------
BYTEA       DB          64H         ;DATOS
BYTEB                   DB          40H
BYTEC                   DB          16H
WORDA                   DW          4000H
WORDB                   DW          2000H
WORDC                   DW          1000H
;-------------------------------------------------------------------------
MAIN    PROC    NEAR    ;Procedimiento principal:
        CALL    B10ADD  ;Llama a la rutina ADD
        CALL    C10SUB  ;Llama a la rutina SUB
                INT     21H
MAIN                    ENDP
;       Ejemplos de suma (ADD) de bytes:
;-------------------------------------------------------------------------
B10ADD                  PROC
                MOV     AL,     BYTEA
                MOV     BL,     BYTEB
```

```
                ADD     AL, BL          ;registro a registro
                ADD     AL, BYTEC       ;memoria a registro
                ADD     BYTEA, BL       ;registro a memoria
                ADD     BL, 10H         ;inmediato a registro
                ADD     BYTEA, 25H      ;inmediato a memoria
                                        RET
B10ADD                                  ENDP
;       Ejemplos de resta (SUB) de palabras:
;-------------------------------------------------------
C10SUB                                  PROC
                        MOV             AX, WORDA
                        MOV             BX, WORDB
                SUB     AX,BX           ;Registro a registro
                SUB     AX,WORDC        ;Memora de registro
                SUB     WORDA, BX       ;Registro de memoria
                SUB     BX, 1000H       ;Inmediato de registro
                SUB     WORDA, 256H     ;Inmediato de memoria
                                        RET
C10SUB                                  ENDP
        END BEGIN
```

Desbordamientos

Este alerta con los desbordamientos en las operaciones aritméticas. Ya que un byte solo permite el uso de un bit de signo y siete de datos (desde -128 hasta +127), una operación aritmética puede exceder con facilidad la capacidad de un registro de un byte. Y una suma en el registro AL, que exceda su capacidad puede provocar resultados inesperados.

OPERANDOS LOGICOS.

La lógica booleana es importante en el diseño de circuitos y tiene un paralelo en la lógica de programación. Las instrucciones para lógica booleana son AND, OR, XOR, TEST y NOT, que pueden usarse para poner bits en 0 o en 1 y para manejar datos ASCII con propósitos aritméticos. El formato general para las operaciones booleanas es:

[etiqueta :] | operación | {registro/memoria}, {registro/memoria/inmediato}|

El primer operando se refiere a un byte o palabra en un registro o memoria y es el único valor que es cambiado. El segundo operando hace referencia a un registro o a un valor inmediato. La operación compara los bits de los dos operandos referenciados y de acuerdo con esto establece las banderas CF, OF, PF, SF y ZF.

AND. Si ambos bits comparados son 1, establece el resultado en 1. Las demás condiciones dan como resultado 0.

OR. Si cualquiera (o ambos) de los bits comparados es 1, el resultado es 1. Si ambos bits están en 0, el resultado es 0.

XOR. Si uno de los bits comparados es 0 y el otro 1, el resultado es 1. Si ambos bits comparados son iguales (ambos 0 o ambos 1), el resultado es 0.

TEST. Establece las banderas igual que lo hace AND, pero no cambia los bits de los operandos.

Las operaciones siguientes AND, OR y XOR ilustran los mismos valores de bits como operandos:

	AND	OR	XOR
	0101	0101	0101
	0011	0011	0011
Resultado:	0001	0111	0110

Es útil recordar la siguiente regla: el empleo de AND con bits 0 es 0 y el de OR con bits 1 es 1.

Ejemplos de operaciones booleanas.

Para los siguientes ejemplos independientes, suponga que AL contiene 11000101 y el BH contiene 01011100:

```
1.-  AND   AL,BH    ;Establece AL a 0100 0100
2.-  AND   AL,00H   ;Establece AL a 0000 0000
3.-  AND   AL,0FH   ;Establece AL a 0000 0101
4.-  OR    BH,AL    ;Establece BH a 1101 1101
5.-  OR    CL,CL    ;Pone en uno SF y ZF
6.-  XOR   AL,AL    ;Establece AL a 0000 0000
7.-  XOR   AL,0FFH  ;Establece AL a 0011 1010
```

Los ejemplos 2 y 6 muestran formas de limpiar un registro, y ponerlo a cero. El ejemplo 3 pone a cero los cuatro bits mas a la izquierda de AL.

TESt actúa igual que AND, pero solo establece las banderas. Aquí están algunos ejemplos :

```
1.- TEST BL, 11110000B ; Alguno de los bits de mas a la
    JNZ ... ; izquierda es BL en diferentes de cero?

2.- TEST AL, 00000001B ; AL contiene
    JNZ ... ; un numero impar?

3.- TEST DX, 0FFH ; El DX contiene
    JNZ ... ; un valor cero?
```

La instrucción NOT.

La instrucción NOT solo invierte los bits en un byte o palabra en un registro o en memoria; esto es, convierte los ceros en unos y los unos en ceros. El formato general es:

```
| [etiqueta:] | NOT | {registro/memoria} |
```

Por ejemplo si el AL contiene 11000101, la instrucción NOT AL cambia el AL a 00111010 (el resultado es el mismo de XOR AL, 0FFH). Las banderas no son afectadas.

1.3 La memoria principal (RAM)

La memoria principal o primaria**,**"Memoria Central ", es aquella memoria de un ordenador, donde se almacenan temporalmente tanto los datos como los programas que la CPU está procesando o va a procesar en un determinado momento. Por su función, es una amiga inseparable del microprocesador, con el cual se comunica a través de los buses de datos. Por ejemplo, cuando la CPU tiene que ejecutar un programa, primero lo coloca en la memoria y después lo

empieza a ejecutar. Lo mismo ocurre cuando necesita procesar una serie de datos; antes de poder procesarlos los tiene que llevar a la memoria principal.

Esta clase de memoria es volátil, es decir que, cuando se corta la energía eléctrica, se borra toda la información que estuviera almacenada en ella.

Por su función, la cantidad de memoria RAM de que disponga una computadora es una factor muy importante; hay programas y juegos que requieren una gran cantidad de memoria para poder usarlos. Otros andarán más rápido si el sistema cuenta con más memoria RAM.

La memoria Caché: dentro de la memoria RAM existe una clase de memoria denominada Memoria Caché que tiene la característica de ser más rápida que las otras, permitiendo que el intercambio de información entre el procesador y la memoria principal sea a mayor velocidad.

La estructura de la memoria principal ha cambiado en la historia de las computadoras. Desde los años 1980 es prevalentemente una unidad dividida en celdas que se identifican mediante una dirección. Está formada por bloques de circuitos integrados o chips capaces de almacenar, retener o "memorizar" información digital, es decir, valores binarios; a dichos bloques tiene acceso el microprocesador de la computadora.

La MP se comunica con el microprocesador de la CPU mediante el bus de direcciones. El ancho de este bus determina la capacidad que posea el microprocesador para el direccionamiento de direcciones en memoria.

En algunas oportunidades suele llamarse "memoria interna" a la MP, porque a diferencia de los dispositivos de memoria secundaria, la MP no puede extraerse tan fácilmente por usuarios no técnicos.

La MP es el núcleo del sub-sistema de memoria de una computadora, y posee una menor capacidad de almacenamiento que la memoria secundaria, pero una velocidad millones de veces superior. Si tienes más memoria almacenas más datos.

1.4 El concepto de interrupciones

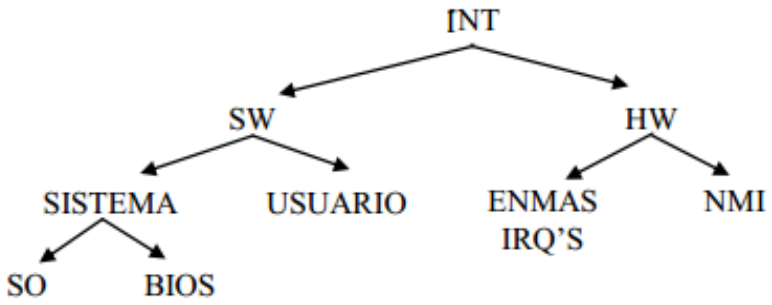
Definición: Una interrupción es el rompimiento en la secuencia de un programa para ejecutar un programa especial llamando una rutina de servicio cuya característica principal es que al finalizar regresa al punto donde se interrumpió el programa.

Dentro de una computadora existen dos clases de interrupciones:
Interrupciones por software: Son aquellas programadas por el usuario, es decir, el usuario decide cuando y donde ejecutarlas, generalmente son usadas para realizar entrada y salida.

Interrupciones por hardware: Son aquellas que son provocadas por dispositivos externos al procesador su característica principal es que no son programadas, esto es, pueden ocurrir en cualquier momento en el programa. Existen dos clases de interrupciones de este tipo:

- Interrupciones por hardware enmascarables: Aquellas en las que el usuario decide si quiere o no ser interrumpido.

- Interrupciones por hardware no enmascarables (NMI): Aquellas que siempre interrumpen al programa.



Las interrupciones por software se ejecutan con ayuda de las instrucciones: INT e IRET, además se tiene 256 interrupciones: de la 00 a la FF.

Asociado al concepto de interrupción se tiene un área de memoria llamada vector de interrupciones; la cual contiene las direcciones de las rutinas de servicio de cada interrupción. Esta área se encuentra en el segmento 0000:0000.

1.5 Llamadas a servicios del sistema

Es el mecanismo usado por una aplicación para solicitar un servicio al sistema operativo**.**

Las llamadas al sistema comúnmente usan una instrucción especial de la CPU que causa que el procesador transfiera el control a un código privilegiado. previamente especificado por el mismo código. Esto permite al código privilegiado especificar donde va a ser conectado así como el estado del procesador.

Cuando una llamada al sistema es invocada, la ejecución del programa que invoca es interrumpida y sus datos son guardados, normalmente en su PCB (Bloque de Control de Proceso del inglés Process Control Block), para poder continuar ejecutándose luego. El procesador entonces comienza a ejecutar las instrucciones de código de alto nivel de privilegio, para realizar la tarea requerida. Cuando esta finaliza, se retorna al proceso original, y continúa su ejecución. El retorno al proceso demandante no obligatoriamente es inmediato, depende del tiempo de ejecución de la llamada al sistema y del algoritmo de planificación de CPU.

1.6 Modos de direccionamiento

Los llamados modos de direccionamiento son las diferentes maneras de especificar en informática un operando dentro de una instrucción en lenguaje ensamblador.

Un modo de direccionamiento especifica la forma de calcular la dirección de memoria efectiva de un operando mediante el uso de la información contenida en registros y / o constantes, contenida dentro de una instrucción de la máquina o en otra parte.

Diferentes arquitecturas de computadores varían mucho en cuanto al número de modos de direccionamiento que ofrecen desde el hardware. Eliminar los modos de

direccionamiento más complejos podría presentar una serie de beneficios, aunque podría requerir de instrucciones adicionales, e incluso de otro registro. Se ha comprobado que el diseño de CPUs segmentadas es mucho más fácil si los únicos modos de direccionamiento que proporcionan son simples.

La mayoría de las máquinas RISC disponen de apenas cinco modos de direccionamiento simple, mientras que otras máquinas CISC tales como el DEC VAX tienen más de una docena de modos de direccionamiento, algunos de ellos demasiado complejos. El mainframe IBM System/360 disponía únicamente de tres modos de direccionamiento; algunos más fueron añadidos posteriormente para elSystem/390.

Cuando existen solo unos cuantos modos, estos van codificados directamente dentro de la propia instrucción (Un ejemplo lo podemos encontrar en el IBM/390, y en la mayoría de los RISC). Sin embargo, cuando hay demasiados modos, a menudo suele reservarse un campo específico en la propia instrucción, para especificar dicho modo de direccionamiento. El DEC VAX permitía múltiples operandos en memoria en la mayoría de sus instrucciones, y reservaba los primeros bits de cada operando para indicar el modo de direccionamiento de ese operando en particular.

Incluso en computadores con muchos modos de direccionamiento, algunas medidas realizadas a programas indican que los modos más simples representan cerca del 90% o más de todos los modos de direccionamiento utilizados. Dado que la mayoría de estas medidas son obtenidas a partir de códigos de alto nivel generados a partir de compiladores, nos da una idea de las limitaciones que presentan los compiladores que se utilizan.

1.7 Proceso de ensamblado y ligado

EDICION

Los archivos fuente de código ensamblador deben estar en formato ASCII standard. Para esto puede usarse cualquier editor que permita crear archivos sin formato, e.g. Edlin, Edit, Write, El editor del Turbo Pascal, Works, Word, WordStar,

etcétera. Las declaraciones pueden ser introducidas en mayúsculas y/o minúsculas. Una buena práctica de programación es poner todas las palabras reservadas (directivas e instrucciones) en mayúsculas y todo lo del usuario en minúsculas para fines de facilidad de lectura del código.

Las sentencias pueden comenzar en cualquier columna, no pueden tener más de 128 caracteres, no se permiten lineas múltiples ni códigos de control, y cada línea debe ser terminada con una combinación de line-feed y carriage-return. Los comentarios se declaran con; y terminan al final de la línea.

ENSAMBLADO

El ensamblado se lleva a cabo invocando al MASM. Este puese ser invocado, usando una línea de comando, de la siguiente manera:

MASM archivo [,[objeto][,[listado][,[cross]]]]][opciones][;]

Dónde:

Objeto.- Es el nombre para el archivo objeto.

Listado.- Nombre del archivo de listado de ensamblado. cross. Es un archivo de referencias cruzadas.

LINK

De la misma forma que el ensamblado, la fase de liga se lleva a cabo con el LINK. Este puede ser invocado de la misma forma que el MASM. Los parámetros que este requiere son:

LINK objeto [,[ejecutable][,[mapa][,[librería]]]]][opciones][;]

dónde:

Objeto.- Es el nombre para el archivo .OBJ

Ejecutable.- Nombre del archivo .EXE

Mapa.- Nombre del archivo mapa

Librería.- Nombre del archivo biblioteca de rutinas

EJECUCION

Para la ejecución del programa simplemente basta teclear su nombre en el prompt de MS-DOS y teclear ENTER. Con esto el programa será cargado en memoria y el sistema procederá a ejecutarlo. El proceso completo para poder crear un programa ejecutable con el Microsoft Macro Assembler se muestra abajo.

1.8 Desplegado de mensajes en el monitor

En este momento podemos comenzar a escribir las verdaderas instrucciones que le indicarán a la computadora que mensaje y como lo va a desplegar. Sugiero que comenzemos por borrar la pantalla. Esto se puede realizar de muy diversas formas, aquí lo haremos usando el BIOS, el cual es un microchip que se encuentra dentro de toda PC y controla las funciones básicas de entrada y salida (**B**asic **I**nput **O**utput **S**ystem). Lo que haremos es decirle al chip "¡Hey! dime en que modo está trabajando la tarjeta de video", cuando obtengamos la respuesta le diremos: "Dile a la tarjeta de video que deje de trabajar en ese modo y que comienze a trabajar en el modo de video que me diste". Una instrucción rara, pues lo que le estamos ordenando es que deje de trabajar en el modo en el que está trabajando !y que comienze a trabajar en ese mismo modo! Así se lo decimos en su propio lenguaje:

```
principio:
mov ah, 0fh
int 10h

mov ah, 0
int 10h
```

Lo primero que vemos es una "etiqueta", con ella le damos nombre a un punto dentro del código, si más tarde dentro del programa deseamos repetir esta parte del código solo tenemos que decir "salta a 'principio'" y ya está. El primer grupo de

instrucciones después de la etiqueta le dicen al BIOS que obtenga la modalidad en la que está trabajando el video. Aquí vemos por primera vez una interrupción (int 10h). Las interrupciones son funciones ya incorporadas dentro del BIOS y del sistema operativo MS-DOS que realizan tareas comunes como leer del disco, mostrar un mensaje en el monitor, o ¡borrar la pantalla!. Enseguida, mediante una función de la interrupción 10h, le decimos que cambie a la misma modalidad de video.

Bueno, ahora que la pantalla está limpia, podemos mostrar nuestro mensaje en el monitor. Aquí está el código:

```
lea dx, mensaje_a_mostrar

mov ah, 9h
int 21h
```

Con la primera instrucción le decimos al procesador "Carga en el registro DX, la dirección de memoria de la variable llamada 'mensaje_a_mostrar'". Enseguida le decimos que la despliegue en pantalla con la función 9h de la interrupción 21h.

Nuestra tarea está terminada, así que digamosle a la computadora que no hay más instrucciones que procesar.

```
int 20h
```

Las instrucciones están terminadas, pero todavía tenemos que decirle a la computadora que valor va a tener la variable 'mensaje_a_mostrar'.

```
mensaje_a_mostrar db "¡Hola Mundo!$",0
```

El signo de pesos al final de la cadena, es necesario para que el sistema operativo sepa en donde se acaba la cadena (una cadena es un grupo de caracteres) que va a desplegar.

Una vez que terminamos con las instrucciones y valores para la máquina, hay que marcar el archivo para que el compilador sepa que ya terminamos de darle instrucciones a la maquina.

```
CODE ENDS
end principio
```

¡Al fin! ¡Llegamos al final! Aquí está el código fuente completo:

```
CODE SEGMENT
ASSUME CS:CODE, DS:CODE, SS:CODE, ES:CODE
ORG 100h

principio:
mov ah, 0Fh
mov ah, 0
int 10h

lea dx, mensaje_a_mostrar
mov ah, 9h
int 21h

int 20h

mensaje_a_mostrar db "¡Hola Mundo!$",0

CODE ENDS
end principio
```