U.N.Ju. – F.I.

INGENIERÍA INFORMÁTICA

LABORATORIO DE COMPUTADORAS

ARQUITECTURA X86

TEMA: ARQUITECTURA X86 (8086) y PROGRAMACIÓN ASSEMBLER

Unidades Lógicas

El procesador se divide en dos unidades lógicas: la unidad de ejecución (EU) y la unidad de interfaz del bus (BIU).

La Unidad de ejecución es la encargada de ejecutar, propiamente, las instrucciones y operaciones aritméticas y lógicas. Contiene a la ALU, la unidad de control y varios registros.

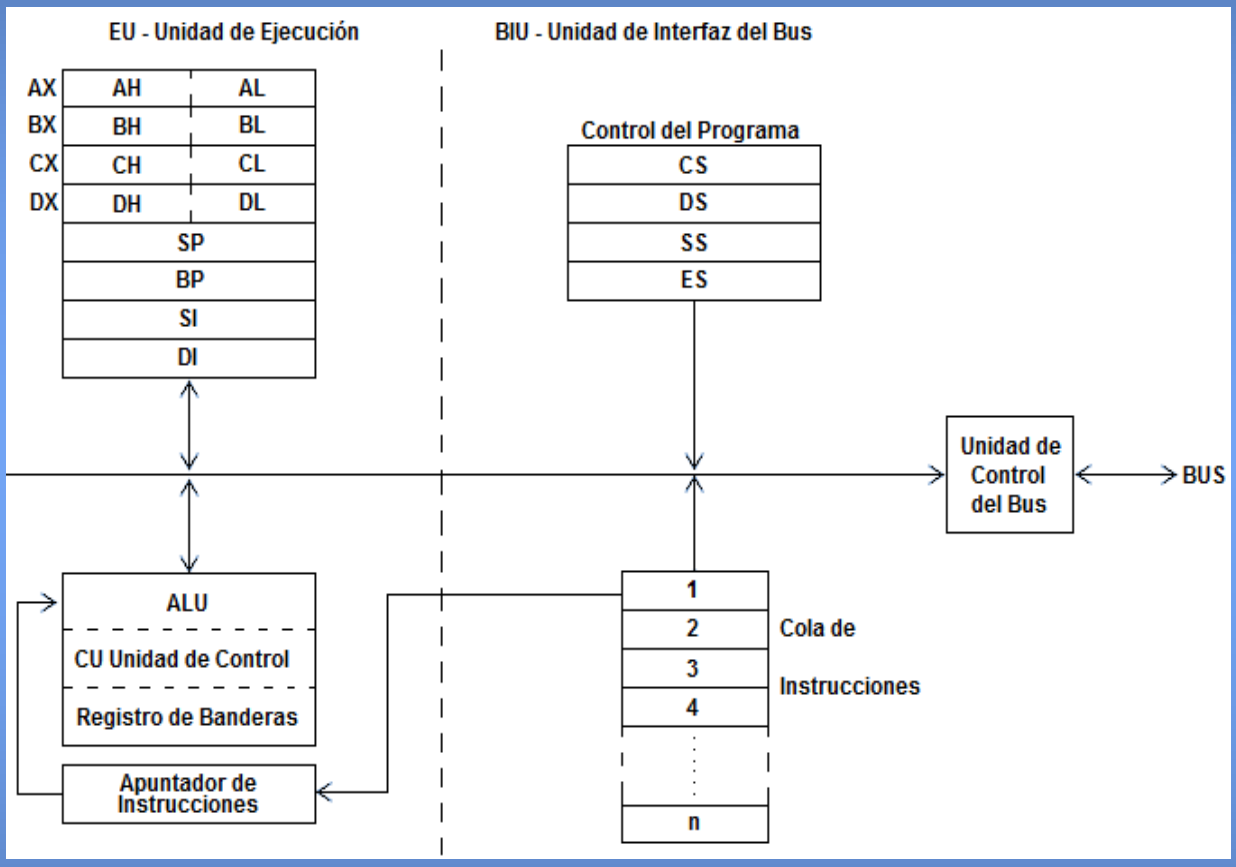
La unidad de interfaz del bus controla los buses que transfieren los datos a la EU, a la memoria y a los dispositivos de E/S externos, mientras que los registros de segmento controlan el direccionamiento de la memoria. La función más importantes de la BIU es manejar la unidad de control del bus, los registros de segmento y la cola de instrucciones. Otra función es permitir el acceso a las instrucciones desde la memoria y colocarlas en la cola de instrucciones. Dependiendo del procesador, la BIU es capaz de adelantarse y buscar con anticipación instrucciones de manera que siempre haya cola de instrucciones, para ser ejecutadas.

Unidades Lógicas

La EU y la BIU trabajan en paralelo, si bien la BIU se mantiene un paso adelante, la EU notifica a la BIU cuando necesita acceso a los datos en memoria o a un dispositivo de E/S. También la EU solicita instrucciones de máquina de la cola de instrucciones de la BIU. La instrucción que se encuentra adelante en la cola es la actualmente ejecutable y mientras la EU está ocupada ejecutando una instrucción, la BIU busca otra en memoria. Esta búsqueda se traslapa con la ejecución y aumenta la velocidad de procesamiento.

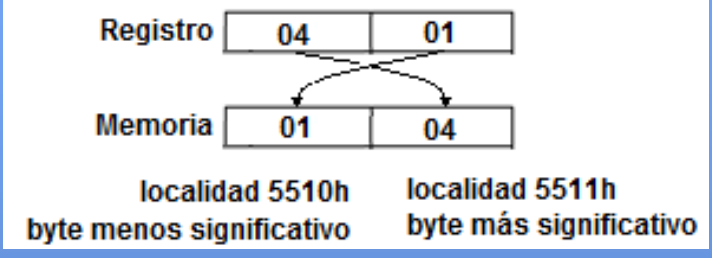
El principio de funcionamiento de ambas unidades lógicas y su relación con los distintos componentes de la arquitectura se refleja en el siguiente gráfico:

Unidades Lógicas



Conceptos Básicos

Direccionamiento de localidades de memoria: dependiendo del modelo, el procesador puede accesar uno o más bytes de memoria a la vez. Por ejemplo, el número 0401h requiere de dos bytes (o una palabra) de memoria. Consta de un byte de orden alto (más significativo) 04h y un byte de orden bajo (menos significativo) 01h. El sistema almacena en memoria estos bytes en secuencia inversa de bytes; el byte de orden bajo en la dirección baja de memoria y el byte de orden alto en la dirección alta de memoria. Por ejemplo, el procesador transferirá 0401h de un registro a las localidades de memoria 5510h y 5511h como:



Segmentos y direccionamiento: un segmento es un área especial en un programa que inicia en un límite de párrafo; esta es una localidad regularmente divisible entre 16 o 10h. Aunque un segmento puede estar ubicado casi en cualquier lugar de memoria, y en modo real puede ser de

Conceptos Básicos

hasta 64 kb, solo necesita tanto espacio como el programa requiera para su ejecución. Se puede tener cualquier número de segmentos. Y para direccionar un segmento particular basta cambiar la dirección en el registro de segmento apropiado. Los tres segmentos principales son los segmentos de código, de datos y de la pila.

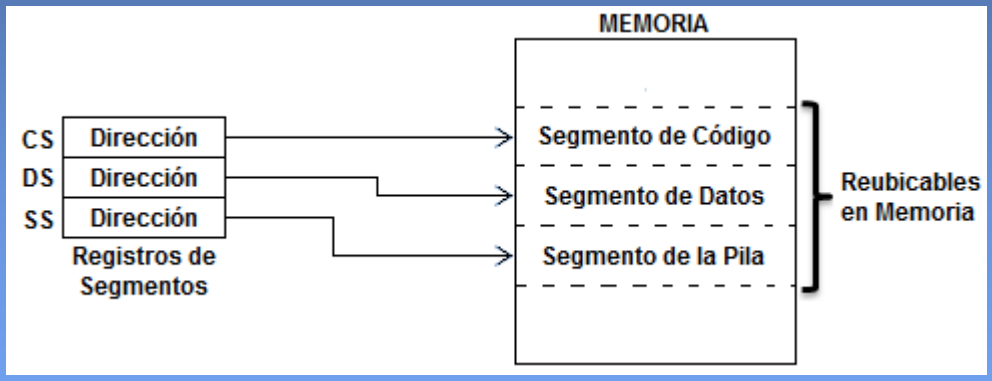
Segmento de Código: contiene las instrucciones de máquina que son ejecutadas. Por lo común, la primer instrucción ejecutable está en el inicio del segmento y el sistema operativo enlaza a esa localidad para iniciar la ejecución del programa. Como su nombre lo indica el CS es el registro de segmento de código y direcciona al segmento de código. Se pueden definir más de un segmento de código.

Segmento de Datos: contiene datos, constantes y áreas de trabajo definidos por el programa. El registro DS es el registro de segmento de datos y direcciona al segmento de datos. Se pueden definir más de un segmento de datos.

Segmento de la Pila: en términos sencillos, la pila contiene los datos y direcciones que se necesiten guardar temporalmente o para uso de sus llamadas subrutinas. El registro SS es el registro de segmento de la pila y direcciona al segmento de la pila.

Conceptos Básicos

Límite de los segmentos: los registros de segmento contienen la dirección inicial de cada segmento.



Otros registros de segmento son el ES (segmento extra) y, en procesadores 80386 y posteriores, los registros FS y GS; que tienen usos especializados.

Recordar que un segmento inicia en un límite de párrafo. Suponga que un segmento se inicia en la dirección 03450h. Como se aprecia, el último dígito hexadecimal de la derecha es 0; los diseñadores de computadoras decidieron que era innecesario almacenar el dígito 0 en el registro de segmento. Así 03450h se almacena como 0345h , con el 0 de la extrema derecha sobreentendido. En algunos casos se indica como 0345[0]h.

Conceptos Básicos

Desplazamiento de segmentos: en un programa, todas las localidades de memoria están referidas a una dirección inicial de segmento. La distancia, en bytes, desde la dirección inicial de segmento se define como el desplazamiento (OFFSET). Un desplazamiento de dos bytes puede estar entre el rango 0000h y FFFFh o bien, entre 0 hasta 65535. Es decir, que el primer byte del segmento tiene un desplazamiento 00, el segundo byte tiene un desplazamiento de 01 y así sucesivamente hasta el desplazamiento 65535.

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.

Por ejemplo, el registro DS contiene la dirección inicial del segmento de datos en 1234[0]h y una instrucción hace referencia a una localidad con un desplazamiento de 0032h dentro del segmento de datos.

Conceptos Básicos

Dirección del segmento DS: 12340h
Desplazamiento : + 0032h
Dirección Real : 12372h

Capacidad de direccionamiento: los registros del 8086/8088 proporcionan 16 bits. Ya que la dirección de un segmento se inicia en un límite de párrafo, los 4 bits de la extrema derecha de su dirección son 0. Por lo tanto solo se almacenan los 16 bits más significativos de la dirección real; en consecuencia se pueden direccionar hasta 1048560 bytes en estas arquitecturas.

Registros

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

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

Registros de segmentos: tienen 16 bits de longitud y facilitan un área de memoria para direccionamiento conocida como 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 más un valor de desplazamiento en el registro apuntador de instrucciones (IP), indica la dirección de una instrucción que se está buscando para su ejecución. Para propósitos de programación normal, no se necesita referenciar el registro CS.

Registros

Registro DS: la dirección inicial del segmento de datos de un programa es almacenada en el registro DS. En términos sencillos esta dirección de segmento más un valor de desplazamiento en una instrucción, genera una referencia a una 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 inicial del segmento de la pila de un programa en el registro SS. Esta dirección de segmento más un valor de desplazamiento en el registro apuntador de la pila (SP), indica la palabra actual en la pila que está siendo direccionada. Para propósitos de programación normal, no se necesita referenciar el registro SS.

Registro ES: algunas operaciones de cadenas de caracteres utilizan el registro extra de segmento para manejar el direccionamiento de memoria. En este contexto el segmento extra está asociado al registro DI (Índice).

Registros

Registro IP: 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, denominado EIP.

En el siguiente ejemplo, el registro CS contiene la dirección inicial del segmento de código en 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.

Dirección del segmento CS: 25A40h
Desplazamiento en el IP : + 0412h
Direc. de la sig. instrucción : 25E52h

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

Registros

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 SP ampliado de 32 bits, denominado ESP. El sistema maneja de manera automática estos registros.

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

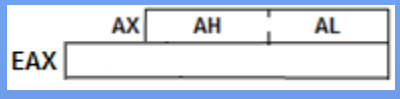
Registros de Propósito General: los registros AX, BX, CX y DX son los caballos de batalla del sistema. Son únicos en el sentido de que se puede direccionarlos como una palabra o como una parte de un byte. El último byte de la izquierda es la parte “alta” y el último byte de la derecha es la parte “baja”.

Registros

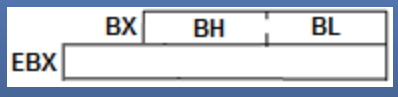
Por ejemplo, el registro CX consta de una parte CH (alta) y una parte CL (baja), y se puede referir a cualquier parte por su nombre: CX, CH o CL.

Los procesadores 80386 y posteriores permiten el uso de todos los registros de propósito general, más sus versiones ampliadas: EAX, EBX, ECX y EDX.

Registro AX: es el acumulador principal, es utilizado para operaciones que implican entrada/salida y la mayor parte de la aritmética.

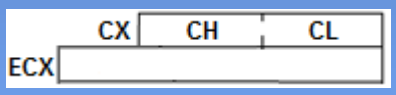


Registro BX: es conocido como registro base ya que es el único registro de propósito general que puede ser un índice para el direccionamiento indexado. También es común emplear el BX para cálculos.

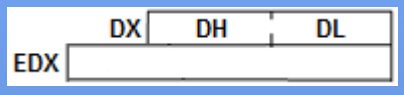


Registros

Registro CX: es conocido como registro contador. Puede contener un valor para controlar el número de veces que un ciclo se repite o un valor para corrimiento de bits, hacia la derecha o hacia la izquierda. También es usado para muchos cálculos.



Registro DX: es conocido como registro de datos. Algunas operaciones de entrada/salida requieren su uso. Las operaciones de multiplicación y división de grandes números suponen al DX y al AX trabajando juntos.



Puede usar los registros de propósito general para sumas y restas de cifras de 8, 16 y 32 bits.

Registros Índices: los registros SI y DI están disponibles para el direccionamiento indexado y para sumas y restas.

Registros

Registro SI: el registro índice fuente de 16 bits es requerido por algunas operaciones de cadena de caracteres.. En este contexto el SI esta asociado con el registro DS. Los procesadores 80386 y posteriores tienen un SI ampliado de 32 bits, denominado ESI.

Registro DI: el registro índice destino de 16 bits es requerido por algunas operaciones de cadena de caracteres.. En este contexto el DI esta asociado con el registro ES. Los procesadores 80386 y posteriores tienen un DI ampliado de 32 bits, denominado EDI.

Registro de Banderas: de los 16 bits del registro de banderas, 9 son comunes para toda la familia del 8086 y sirven para indicar el estado actual de la máquina y el resultado del procesamiento. Instrucciones que manejan comparaciones y aritmética cambian el estado de una o más banderas, y algunas de esas instrucciones realizan pruebas para determinar la acción subsecuente.

N° de Bit	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Bandera					O	D	I	T	S	Z		A		P		C

Registros

Descripción de las Banderas

OF (Overflow flag, Desbordamiento)

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

DF (Direction flag, Dirección)

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

IF (Interruption flag, Interrupción)

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

TF (Trap flag, Trampa)

Examina el efecto de una instrucción sobre los registros y la memoria. Los programas depuradores como DEBUG, activan esta bandera de manera que pueda avanzar en la ejecución de una sola interrupción a un tiempo.

SF (Sign flag, Signo)

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

Registros

ZF (Zero flag, Cero)

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

AF (Auxiliary carry flag, Acarreo auxiliar)

Contiene un acarreo externo del bit 3 en un dato de 8 bits, para aritmética especializada.

PF (Parity flag, Paridad)

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

CF (Carry flag, 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 rotación.

El Programa DEBUG

Para introducir al lector en la programación de bajo nivel se puede usar el DEBUG, un programa del DOS.

Este permite visualizar la memoria, introducir programas en ella, y rastrear su ejecución.

El rastreo de como se ejecutan estas instrucciones dan una idea de la operación de una computadora y la función de los registros.

Es utilizado para probar y depurar programas ejecutables. Ejecuta las instrucciones en modo de paso sencillo (un paso a la vez) de manera que se pueda ver el efecto de cada una sobre las localidades de memoria y sus registros.

Una característica del DEBUG es que despliega todo el código del programa y los datos en formato hexadecimal y cualquier dato que se introduzca a la memoria también está en formato hexadecimal.

Los programas generados con DEBUG son de tipo .COM.

Lenguaje Ensamblador

Ventajas:

1. Proporciona más control sobre el manejo particular de los requerimientos de hardware.
2. Genera módulos ejecutables más pequeños y más compactos.
3. Con mayor probabilidad tiene una ejecución más rápida.
4. Se programan módulos críticos.

Características

Comentarios: se pueden expresar en los siguientes formatos:

1. ; toda la línea es un comentario
2. INC CX; el resto de la línea es comentario

También se puede usar la directiva COMMENT.

Identificadores: es un nombre que se aplica a elementos en el programa.

- **Nombres**: se refiere a la dirección de un elemento de datos.
- **Etiquetas**: se refiere a la dirección de una instrucción.

Lenguaje Ensamblador

Para definir un identificador se puede usar:

1. Letras del alfabeto: desde A hasta la Z.
2. Dígitos. Desde o al 9 (no puede ser el 1° carácter).
3. Caracteres especiales: ?, _; $, @, . (punto, no puede ser el 1° carácter).

Palabras reservadas: tienen propósitos propios y son usadas bajo condiciones especiales.

Por categorías, las palabras reservadas incluyen:

- **Instrucciones**: son operaciones que la máquina puede ejecutar. Ej. ADD, MOV
- **Directivas**: se emplean para proporcionar comandos al ensamblador. Ej. PROC, END
- **Operadores**: se utilizan en la definición de expresiones. Ej. FAR, SIZE
- **Símbolos predefinidos**: regresan información a su programa. Ej. @DATA, @MODEL.

Formato de Enunciados

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

- Directivas: indican al ensamblador que realice una acción específica, como definir un elemento de datos. Se resuelven en tiempo de ensamblado y no generan código objeto.
- Instrucciones: tales como ADD y MUL, y que el ensamblador traduce a código objeto.

Formato General

[Identificador] operación [operando/s][;comentario]

Ejemplos:

```
Contador    DB      0
            Add     BX, 10
            Inc     Cx
            Ret
```

Principales Directivas

Definición de datos: DB (definir byte), DW (definir palabra), DD (definir doble palabra), DQ (definir cuádruple palabra), DT (definir 10 bytes): sirven para declarar las variables, asignándolas un valor inicial:

```
anno        DW   1991
mes         DB   12
numerazo    DD   12345678h
texto       DB   "Hola",13,10
```

Se pueden definir números reales de simple precisión (4 bytes) con DD, de doble precisión (8 bytes) con DQ y «reales temporales» (10 bytes) con DT; todos ellos con el formato empleado por el coprocesador. Para que el ensamblador interprete el número como real ha de llevar el punto decimal:

```
temperatura  DD   29.72
espanoles91  DQ   38.9E6
```

Con el operando DUP pueden definirse estructuras repetitivas.

Principales Directivas

Por ejemplo, para asignar 100 bytes a cero y 25 palabras de contenido indefinido (no importa lo que el ensamblador asigne):

```
ceros    DB    100 DUP (0)
basura   DW    25 DUP (?)
```

Se admiten también los anidamientos. El siguiente ejemplo crea una tabla de bytes donde se repite 50 veces la secuencia 1,2,3,7,7:

```
tabla    DB    50 DUP (1, 2, 3, 2 DUP (7))
```

Title: Título del listado, especifica un título que aparecerá en el listado como primera línea en cada página.

Ej. Title texto

Segment: inicio de un segmento.

EndS: fin de un segmento.

Ej. Datos Segment

.

.

.

Datos EndS

Principales Directivas

Proc: inicio de un procedimiento.

EndP: fin de un procedimiento.

Ej. Inicio Proc Far/Near

.

.

[ret]

Inicio EndP

Assume: hay que indicarle al ensamblador el propósito de cada segmento en el programa. La directiva para este propósito es ASSUME, codificada en el segmento de código.

Ej.

Assume SS:nompila, DS:nomdatos, CS:nomcodigo

End: esta directiva finaliza todo el programa.

Ej. End Inicio

Equ (EQUivalence): Asigna el valor de una expresión a un nombre simbólico fijo:

olimpiadas EQU 1992

Donde *olimpiadas* ya no podrá cambiar de valor en todo el programa. Se trata de un operador muy flexible.

Modos de Direccionamiento

De Registro: en esta forma de direccionamiento, el operando se obtiene de un registro o se coloca en este. Los registros deben ser de igual tamaño.

Ej. Sub AX, AX

Inmediato: el usuario especifica un byte o palabra como operando fuente. Esta constante se ensambla como parte de la instrucción.

Ej. Mov AX, 1234h

Directo (Absoluto): La dirección efectiva (EA) de 16 bits se toma directamente del campo de desplazamiento de la instrucción. El desplazamiento se coloca en la localidad siguiente al código de operación. Esta EA o desplazamiento es la distancia de la localidad de memoria al valor actual en el segmento de datos (DS) en el cual el dato está colocado.

Ej. Mov CX, START.

De Registro Indirecto: La dirección efectiva está especificada en un registro apuntador o un registro índice: BX, BP, SI o DI. Los registros deben indicarse entre corchetes. Como la EA es una dirección y no el contenido de una dirección, antes

Modos de Direccionamiento

de utilizar los registros mencionados; éstos deben contener direcciones.

Ej. Mov BX, OFFSET DATO1

Este modo de direccionamiento se puede usar para mover, por ej., el contenido de la dirección a la que apunta el registro BX.

Ej. Mov AX, [BX]

Relativo a la Base: la EA se obtiene sumando un desplazamiento (8 bits con signo o 16 bits sin signo) a los contenidos de BX o BP. En este caso, los registros deben contener la dirección de desplazamiento.

Ej. Mov AX, [BX + 2]

Indexado directo: la EA se calcula sumando un desplazamiento (8 o 16 bits) a los contenidos de SI o DI. Es común cargar una dirección en un registro índice primero y después la misma se combina con una localidad de memoria.

Ej. Mov SI, 2

Mov AH, Datos2 [SI]

Modos de Direccionamiento

Indexado de Base: la EA se calcula sumando los contenidos de los registros base (BX o BP), un registro índice (SI o DI) y un desplazamiento (opcional de 8 o 16 bits).

Ej. Mov BX, OFFSET Dato3

Mov SI, 8

Mov AX, [BX] [SI + 2]

Relativo: En este modo el operando se especifica como un desplazamiento de 8 bits con signo, relativo al IP.

Ej. JNC VOLVER. Si C=0, entonces bifurca a la instrucción con etiqueta VOLVER.

Implícito: Las instrucciones que usan este modo no tienen operandos.

Ej. CLC

Programación en Macroensamblador

Para poder crear un programa se requiere de un editor para crear el programa fuente. Segundo un ensamblador (compilador) que no es más que un programa que "traduce" el programa fuente a un programa objeto. Y tercero un enlazador o linker, que genere el programa ejecutable a partir del programa objeto.

Programación en Macroensamblador

- Escribir el código utilizando un generador de archivos de texto puro y guardarlo con extensión .asm. Por ejemplo: MiPrograma.asm. Este archivo se denomina archivo fuente.

- MASM MiPrograma; Enter

  Se obtiene el archivo objeto con extensión .obj. Por ejemplo: MiPrograma.obj

- TLINK MiPrograma Enter

  Se obtiene el archivo ejecutable con extensión .exe. Por ejemplo: MiPrograma.exe

Ej. C:\> MASM MiPrograma[.asm]; enter

C:\> TLINK MiPrograma[.obj] enter

C:\> MiPrograma[.exe] enter

Ejemplo de Programa

Se pide sumar dos elementos de datos en el registro AX.

```
Title Mi primer programa
;-------------------------------------------------------
Pila     Segment
          DW          32 DUP (0)
Pila      EndS
;-------------------------------------------------------
Datos    Segment
Valor1   DW          250
Valor2   DW          125
Resul    DW           ?
Datos    EndS
;-------------------------------------------------------
Codigo   Segment
Inicio     Proc        Far
          Assume SS:Pila, Ds:Datos, CS:Codigo

          Mov AX, DataSG
          Mov DS, AX
```

Ejemplo de Programa

```
        Mov AX, Valor1
        Add AX, Valor2
        Mov Resul, AX

        Mov AH, 4Ch
        Int 21h

Inicio  EndP
Codigo  EndS

        End Inicio
```