1. Modos de direccionamiento

Son medios que facilitan la tarea de programación, permitiendo el acceso a los datos de una manera natural y eficiente.

Estos indican al procesador como calcular la dirección absoluta (real o efectiva) donde se encuentran los datos.

Los modos de direccionamiento indican la manera de obtener los operandos y son:

1. Direccionamiento de registro
2. Direccionamiento inmediato
3. Direccionamiento directo
4. Direccionamiento indirecto mediante registro
5. Direccionamiento indirecto por registro base
6. Direccionamiento indexado
7. Direccionamiento indexado respecto a una base

El tipo de direccionamiento se determina en función de los operandos de la instrucción.

1.- Direccionamiento de Registro: Los operandos o datos se encuentran en registros No se necesita calcular la Dirección Absoluta: (DS+Dir_desplazamiento).
Ejemplos:

```
ADD BX, CX
MOV BX, AX
SUB DX, BX
```

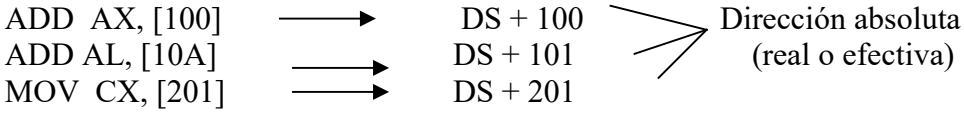
2.- Direccionamiento inmediato : El operando es un número que forma parte de la instrucción. No se necesita calcular la Dirección Absoluta: (DS+Dir_desplazamiento).
Ejemplos:

```
ADD BX, 2h
SUB  CX, 100h
MOV  dx, 30h.
```

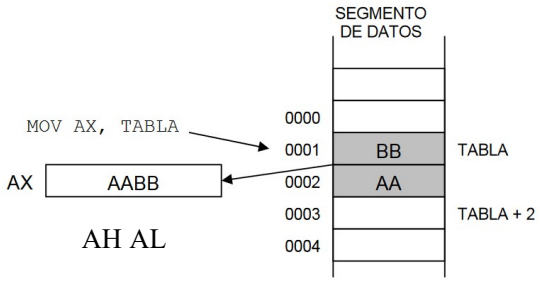
3.- Modo directo: Cuando el operando es una dirección de memoria. Ésta puede ser especificada con su valor entre [ ], o bien mediante una variable definida previamente. Aquí el procesador calcula la Dirección Absoluta: (DS+Dir_desplazamiento).

Ejemplo:

MOV BX,[1000] ; almacena en BX el contenido de la dirección de memoria DS:1000.



MOV AX,TABLA ; almacena en AX el contenido de la dirección de memoria ;DS:TABLA.,



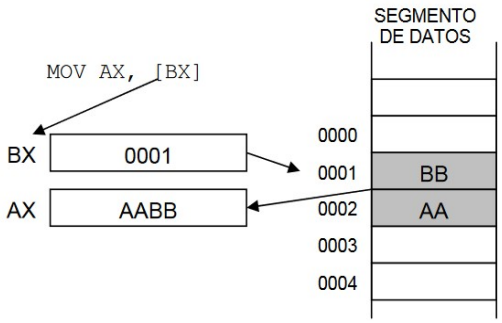
Suponga: **MOV AX, TABLA**
Al ensamblarlo se tiene la instrucción :
MOV AX, [xxxx]

4.- Direccionamiento indirecto mediante registro: Cuando el operando esta en memoria en una posición contenida en un registro (BX, BP, SI o DI). Aquí el procesador calcula la Dirección Absoluta: (DS+Dir_desplazamiento).

Ejemplos:

ADD AX, [DI] ⟶ ES + DI
ADD AL, [SI] ⟶ DS +SI
SUB CX, [BP] ⟶ SS + BP
MOV [BP],CX ⟶ SS + BP ; almacena en la dirección apuntada por BP el ;contenido de CX.

MOV AX, [BX] ⟶ DS + BX ;almacena en AX el contenido de la dirección ;de memoria DS:[BX].



Suponga:

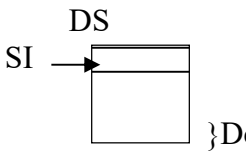
MOV BX, OFFSET DATO o (LEA BX, DATO)
MOV AX,[BX]

Al ensamblar se tiene : **MOV BX, xxxx**
MOV AX, [BX]

5.- Direccionamiento Indexado o indexado directo: Cuando la dirección del operando es obtenida como la suma de un desplazamiento más un índice (DI, SI). El procesador calcula la dirección efectiva (real o absoluta sumando a DS o a ES SI o DI respectivamente) más un desplazamiento.

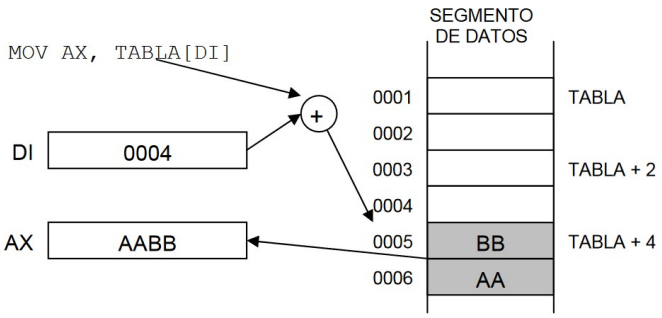
Ejemplos:

ADD AX, [SI + 70] ; Suma a AX el contenido de la posición de memoria apuntada por el resultado de sumarle DS + SI + Desplazamiento(70)



}Desplazamiento DS + SI + 70H

MOV AX, TABLA[DI] ; almacena en AX el contenido de la posición de memoria apuntada por el resultado de sumarle a TABLA el contenido de DI.



Suponga:

MOV SI,2
MOV AX, DATO [SI]

Al ensamblar:

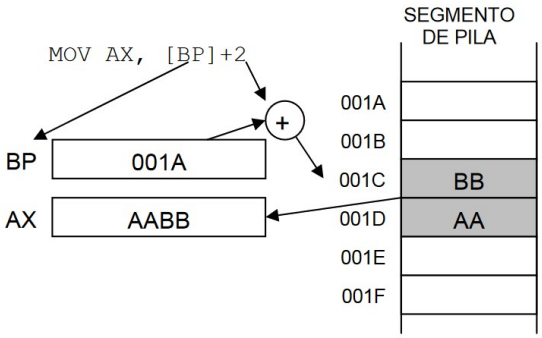
MOV SI,0002
MOV AX,[SI+xxxx]

6.- Direccionamiento de Base o por registro base: Se involucran los registros BX (asociado con DS) O BP (Asociado con SS) y un desplazamiento. Cuando el operando esta en memoria en una posición apuntada por el registro BX o BP al que se le añade un determinado desplazamiento. Aquí el procesador calcula la Dirección Absoluta: (DS+Dir_desplazamiento+desplazamiento).

Ejemplos:

```
ADD AL,  [BX+7]          ; Dirección real = DS+BX+7
```

```
MOV AX, [BP] + 2         ; almacena en AX el contenido de la posición de memoria
                         que resulte de sumar 2 al contenido de BP (dentro del
                         segmento de pila). Equivalente a MOV AX, [BP + 2]
```



Este tipo de direccionamiento permite acceder de una forma cómoda, a estructuras de datos que se encuentran en memoria.

Suponga:

```
MOV BX, OFFSET DATO o  (LEA BX, DATO)
MOV AX,[BX+2]
```

Al ensamblar se tiene :

```
MOV BX, xxxx
MOV AX, [BX+0002]
```

Suponga:

```
LEA SI, CONTENIDO+BX
```

Al ensamblar se tiene :

```
LEA SI, [BX+xxxx]
```

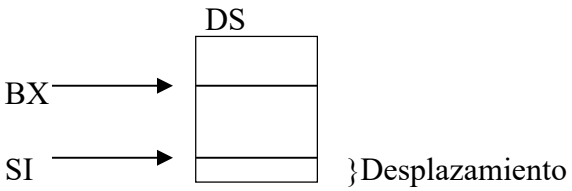
Suponga:

```
MOV BX,18
 MOV TABLAB[BX+1],'$'
```

Al ensamblar se tiene :

```
MOV BX,18
MOV [BX+xxxx+1],'$'
```

7.- Direccionamiento indexado respecto a una base o indexado con base. Cuando la dirección del operando se obtiene de la suma de un registro base (BP o BX), de un índice (DI, SI) y opcionalmente un desplazamiento.



MOV AL, [BX + SI +20]
Dir_ real = DS+BX+SI+20

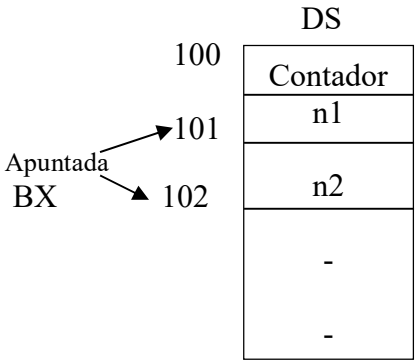
MOV AX, TABLA[BX][DI] ; almacena en AX el contenido de la posición de memoria apuntada por la suma de TABLA, el contenido de BX y el contenido de DI.

MOV BX, OFFSET DATO
MOV SI, 8
MOV AX, [BX][SI+2]

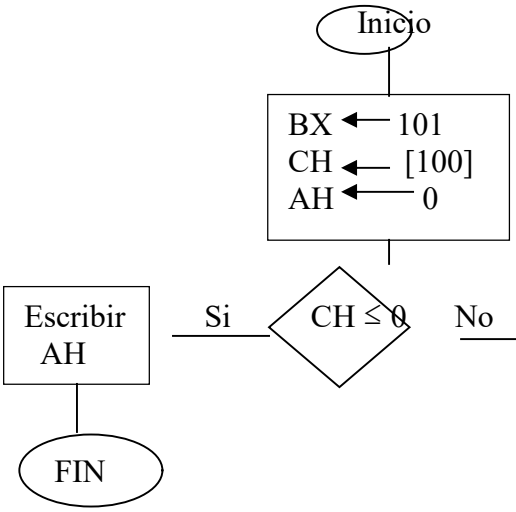
Al ensamblar se tiene:

MOV BX, xxxx
MOV SI, 0008
MOV AX,[BX+SI+xx]

Ejemplo 1: Suponga que se desea sumar n números que se encuentran almacenados en memoria a partir de la localidad 101 donde el contador también está en memoria en la localidad 100.



Apuntadores a DS = SI, BX
Contador = CH
Suma = AH
Apuntador = BX




```
        MOV BX, 101
        MOV CH, [100]
        MOV AH, 0
Ciclo:  CMP CH, 0
        JLE FIN
        ADD AH, [BX]
        INC BX,
        DEC CH
        JMP Ciclo
FIN:    NOP
```

Ejemplo 2:Ejemplo 3: Dados los números del l al A almacenado a partir de la localidad 110, obtener su tabla de multiplicar para cualquier numero del 1 al A, almacenar sus resultados a partir de la localidad 110+A

Respuesta ejercicio :

```
        Mov cl, num
        Mov ch,A
        Mov bx,110
RE:
        Mov al,[bx]
        Mul cl
        Mov [bx+A], AH ; el resultado es de un byte, aunque se alojo en 16 bits (AX)
        Inc bx
        Dec ch
        Cmp ch,0
        Jg RE
        NOP
```

Ejemplo 4: De un intervalo dado n y m (longitud máxima un byte: 00...FF) , almacenados en memoria en las localidades 100 y 101 respectivamente, cuente y clasifique cuales son pares e impares, almacene el total de pares en la localidad 102 y el total de impares en la localidad 103, los números pares a partir de la localidad 104 y los impares a partir de la 204.

Respuesta ejercicio:

```
            Mov cl ,0
            Mov al,[100]
            Mov ah,0
            Mov ch,2
            Mov bx,204          ;  dirección de almacenamiento impares
            Mov si,104          ; dirección de almacenamiento pares
RESG:       Mov dx,ax                ;resguardando
            Div ch
            Cmp ah,0
            Je PAR
            Mov [bx],dl,
            inc bx
            Jmp CONTINUA
PAR:
            Mov [si],dl
            inc si
            inc cl              ; contador de pares
CONTINUA:
            Mov ax,dx           ; recuperación del resguardo de ax
            inc ax
            Cmp al, [101]
            Jle RESG
            Mov ah, [100]
            Mov al,[101]
            Sub al,ah
            Inc al
            Sub al,cl           ; calculo de contador de impares
            Mov [102],cl
            Mov [103],al
            NOP
```

Ejemplo 2: Programa que lee caracteres hasta dar un punto, cuenta cuantos caracteres numéricos se insertan, almacena la cuenta en de la dirección 100 del DS y a partir de la dirección 101 almacena las mayúsculas leídas.

```
            MOV CH,0
            MOV BX,101
LEER:       CALL LEE_CAR
            CMP AL,30          ; Caracter igual a '0'
            JL ESPUNTO
            CMP AL,39          ; Caracter igual a '9'
            JG ESLETRA
            INC CH
            JMP  LEER
ESPUNTO:    CMP AL,2E ; Carácter igual a un punto '.'
            JNZ LEER
            MOV [100],CH
            JMP FIN
ESLETRA:    CMP AL,41          ; Carácter igual a 'A'
            JL LEER
            CMP AL,5A          ; Carcter igual a 'Z'
            JG LEER
            MOV [BX],AL
            INC BX
            JMP LEER
FIN:        NOP
```

1. Se tiene la siguiente fracción de programa en lenguaje ensamblador:

```
XOR AX, AX
MOV SI, 0000H
SUB AX, [SI]
ADC AX,  [SI + 2]
XCHG AX,  [SI + 4]
MOV BX, [0012H]
AND [BX], AX
CMP DX, CX
```

a) Dar el modo de direccionamiento de cada instrucción

2. Indicar el modo de direccionamiento de las siguientes instrucciones:

MOV BX, [BX + 0050H]

ADD [1000H], BP

SUB CX, 1020H

MOV [EBX + 8*EAX], EDX

3. Considerar el siguiente programa en lenguaje ensamblador.

```
PUSH DS
PUSHA
SUB AX, AX
MOV AX, CS
MOV DS, AX
MOV BX, 0100H
MOV AX, [BX]
ADD AX, [BX + 2]
ADC AX, [BX + 4]
MOV [BX + 6], AX
POPA
HLT
```

Se pide:

a) Deducir el modo de direccionamiento de cada instrucción