MICROPOCESADOR 8086

NOTAS DE CURSO

(Versión 2.2)

ÍNDICE

DIRECCIONAMIENTO DE MEMORIA

Los registros del 8086 son de 16 bits, por lo tanto el número de direcciones posibles a direccionar con 1 solo registro es:

$$2^{16} = 65536_{10} = 10000_{16}$$

lo cual representa un total de 64 Kbytes y los valores de direcciones se encuentran en el rango de 0 a FFFF.

Para superar este límite se utilizan 2 registros para direccionar memoria: Uno de SEGMENTO y otro de DESPLAZAMIENTO (offset) dentro del segmento. La notación utilizada para una dirección segmentada es:

SEGMENTO:DESPLAZAMIENTO

La relación entre la dirección de memoria real y la dirección segmentada es:

DIR = SEGMENTO * 16 + DESPLAZAMIENTO

Al multiplicar por 16 se obtienen 4 bits más con lo que ahora se tiene:

$$2^{20} = 1048576_{10} = 100000_{16}$$

con lo cual tenemos un total de 1024Kb = 1Mb de memoria direccionable. Los valores para las direcciones reales se encuentran en el rango 0 a FFFFFh.

Es importante hacer notar que una misma dirección de memoria puede ser direccionada con distintos valores de segmento y desplazamiento

Ej:

100:50 = 105:0 =0:1050, trabajando en base 16.

MODOS DE DIRECCIONAMIENTO

Se entiende por modos de direccionamiento a las diferentes formas que pueden tomar los parámetros de las instrucciones del procesador.

Diferentes autores clasifican en forma distinta los modos de direccionamiento del 8086.

Nosotros distinguiremos fundamentalmente cuatro modos diferentes:

REGISTRO

Un parámetro que direcciona a un registro está utilizando el modo de direccionamiento REGISTRO.

Ej: MOV Ax,Bx

En este ejemplo los dos parámetros direccionan un registro.

VALOR o INMEDIATO

El modo de direccionamiento INMEDIATO es utilizado cuando se hace referencia a un valor constante. Este se codifica junto con la instrucción. Es decir dicho parámetro representa a su valor y no a una dirección de memoria o un registro que lo contiene.

Ej: MOV Ax,500

En este ejemplo el número 500 es un parámetro inmediato.

DIRECTO

Se utiliza el modo directo cuando se referencia a una dirección de memoria y la misma esta codificada junto con la instrucción.

Ej:

MOV Al,[127]

En este ejemplo el desplazamiento de la dirección de memoria se codifica junto con la instrucción y el segmento se asume a DS.

Si MEMORIA es un array de bytes que representa a la memoria la instrucción anterior se puede poner como:

Al := MEMORIA[DS:127]

INDIRECTO

Se utiliza el modo directo cuando se referencia a una dirección de memoria a través de uno o varios registros

Ej: MOV Al,[Bx]

Aquí el offset de la dirección de memoria está contenido en el registro Bx y al igual que el caso anterior como no se especifica el segmento se asume DS.

Si MEMORIA es un array de bytes que representa a la memoria la instrucción anterior se puede poner como:

Al := MEMORIA[DS:Bx]

La especificación completa de las expresiones que van dentro de los paréntesis rectos es:

{ Bx | Bp } [+ { Si | Di }] [+ desplazamiento] |
{ Si | Di } [+ desplazamiento] |
desplazamiento

Donde la expresión entre {} es obligatoria, la expresión entre [] es opcional y el signo | implica opción entre dos expresiones.

Fuera del paréntesis recto se puede indicar cual es el registro de segmento usado para completar la dirección de memoria. En caso que este no se especifique siempre se asume uno por defecto.

Ejemplos:

Mov Ax, [Bp + 3]

Add [Bx + Si], 4

Sub Es:[Bx + Di + 5],Dx

En la Tabla 1 se indican las combinaciones posibles entre registros índice y los segmentos así como las asignaciones por defecto.

Tabla 1.Combinación entre registros de segmento e indices.

	CS	SS	DS	ES
IP	Si			
SP		si		
BP	Prefijo	por defecto	prefijo	prefijo
BX	Prefijo	prefijo	por defecto	prefijo
SI	Prefijo	prefijo	por defecto	prefijo
DI	Prefijo	prefijo	por defecto	por defecto (cadenas)

INSTRUCCIONES

Para describir a cada una de las instrucciones usaremos el siguiente formato:

CÓDIGO

Formato: CÓDIGO op_1, op_2

Tipo Args: Se indica qué forma puede tomar cada parámetro poniendo debajo del mismo una lista de códigos entre paréntesis que definimos así:

A	dirección absoluta inmediata (4 bytes)
a	dirección absoluta inmediata (2 bytes)
i	operando inmediato (1 o 2 bytes)
d	desplazamiento inmediato (1 byte)
r	registro de uso general (8 o 16 bits)
R	registro de uso general (16 bits)
m	palabra de memoria (1 o 2 bytes)
M	palabra de memoria (2 bytes)
W	doble palabra de memoria (4 bytes)
CL	el nombre de un registro en particular

Lógica: Se indica usando pseudo código cual es la lógica de la instrucción.

Descripción: Descripción de la semántica de la instrucción.

Banderas:

OF	DF	IF	TF	SF	ZF	AF	PF	CF
X	-	-	-	?	X	X	X	X

En la tabla de Banderas se indica cómo afecta a las flags del procesador la ejecución de la instrucción, usando el siguiente código:

- X → afecta siempre el valor de la flag colocando el valor apropiado según corresponda.
- ? → el valor de la flag luego de la ejecución de la instrucción es indeterminado
- - → la ejecución de la instrucción no afecta el valor de flag

ARITMÉTICAS

ADD

Formato: ADD op_1, op_2

Tipo Args: (r,m) (r,m,i)

Lógica: $op_1 \leftarrow op_1 + op_2$

Descripción: Suma los dos operandos y almacena el resultado en op_1, por lo tanto ambos deben tener el mismo tipo

Banderas:

OF	DF	IF	TF	SF	ZF	AF	PF	CF
X	-	-	-	X	X	X	X	X

ADC

Formato: ADC op_1, op_2

Tipo Args: (r,m) (r,m,i)

Lógica: $op_1 \leftarrow op_1 + op_2 + c$

Descripción: Idem a ADD pero además suma el valor del carry.

Banderas:

OF	DF	IF	TF	SF	ZF	AF	PF	CF
X	-	-	-	X	X	X	X	X

SUB

Formato: SUB op_1, op_2

Tipo Args: (r,m) (r,m,i)

Lógica: $op_1 \leftarrow op_1 - op_2$

Descripción: Idem a ADD pero realiza la resta en lugar de la suma.

Banderas:

OF	DF	IF	TF	SF	ZF	AF	PF	CF
X	-	-	-	X	X	X	X	X

SBB

Formato: SBB op_1, op_2

Tipo Args: (r,m) (r,m,i)

Lógica: $op_1 \leftarrow op_1 - op_2 - c$

Descripción: Idem a SUB pero además resta el carry.

Banderas:

OF	DF	IF	TF	SF	ZF	AF	PF	CF
X	-	-	-	X	X	X	X	X

MUL

Formato: MUL op

Tipo Args: (r,m)

Lógica:

si op es de tipo byte $\Rightarrow$
$\mathbf{Ax} = \mathbf{Al} * op$
sino si op es de tipo palabra $\Rightarrow$
$\mathbf{Dx, Ax} = \mathbf{Ax} * \mathbf{op}$
fin si
si la mitad superior del resultado es 0
$\mathbf{CF} = \mathbf{0}$
sino
$\mathbf{CF} = \mathbf{1}$
fin si
$\mathbf{OF} = \mathbf{CF}$

Descripción: Multiplica sin considerar el signo, el acumulador $\mathbf{(Ax\ o\ Ax)}$ por el operando op según este último sea de tipo byte o palabra.

En caso que op sea de tipo byte ls resultado se almacena en $\mathbf{Ax}$ si es de tipo palabra se almacena en el par de registros $\mathbf{Ax}$, $\mathbf{Dx}$ colocando la parte más significativa en $\mathbf{Dx}$.

Banderas:

OF	DF	IF	TF	SF	ZF	AF	PF	CF
X	-	-	-	?	?	?	?	X

DIV

Formato: DIV op

Tipo Args: (r,m)

Lógica:

si op es de tipo byte $\Rightarrow$
  si $\mathbf{Ax}\ \mathbf{div}\ op > \mathbf{FFh} \Rightarrow$
    $\mathbf{INT}\ 0$
  else
    $\mathbf{Al} = \mathbf{Ax}\ \mathbf{div}\ op$
    $\mathbf{Ah} = \mathbf{Ax}\ \mathbf{mod}\ op$
  fin si

fin si
sino si op es de tipo palabra $\Rightarrow$
  si $\mathbf{Dx}:\mathbf{Ax}\ \mathbf{div}\ op > \mathbf{FFh} \Rightarrow$
    $\mathbf{INT}\ 0$
  else
    $\mathbf{Ax} = \mathbf{Dx}:\mathbf{Ax}\ \mathbf{div}\ op$
    $\mathbf{Dx} = \mathbf{Ax}\ \mathbf{mod}\ op$
  fin si

Descripción: Divide sin considerar el signo, el acumulador $\mathbf{Ax}$ por el operando op si este último es de tipo byte. Si op es de tipo palabra el número dividido por éste es $\mathbf{Dx}:\mathbf{Ax}$.

En caso que op sea de tipo byte el cociente se almacena en $\mathbf{Al}$ y el resto en $\mathbf{Ah}$. Si es de tipo palabra el cociente se almacena en $\mathbf{Ax}$ y el resto en $\mathbf{Dx}$

Si el cociente es mayor que el número máximo representable (FFh o FFFFh según sea el caso), el resultado queda indefinido y se genera una interrupción tipo 0.

Banderas

OF	DF	IF	TF	SF	ZF	AF	PF	CF
?	-	-	-	?	?	?	?	?

NEG

Formato: NEG op

Tipo Args: (r,m)

Lógica:

si op es de tipo byte $\Rightarrow$
$op = \mathbf{FFh} - op$
$op = op + 1$
sino si op es de tipo palabra $\Rightarrow$
$op = \mathbf{FFFFh} - op$
$op = op + 1$
fin si

Descripción: Calcula el complemento a 2 de op

Banderas:

OF	DF	IF	TF	SF	ZF	AF	PF	CF
X	-	-	-	X	X	X	X	X

CBW

Formato: CBW

Lógica:

si **Al** < 80**h**
$\mathbf{Ah} = 00\mathbf{h}$
sino
$\mathbf{Ah} = \mathbf{FFh}$
fin si

Descripción: Copia el bit 7 del registro AL en todos los bits del registro AH; es decir expande el bit de signo de AL

Banderas:

OF	DF	IF	TF	SF	ZF	AF	PF	CF
-	-	-	-	-	-	-	-	-

INC

Formato: INC op

Tipo Args: (r,m)

Lógica: $op = op + 1$

Descripción: Incrementa el operando destino.

Banderas:

OF	DF	IF	TF	SF	ZF	AF	PF	CF
X	-	-	-	X	X	X	X	-

DEC

Formato: DEC op

Tipo Args: (r,m)

Lógica: $op = op - 1$

Descripción: Decrementa el operando destino.

Banderas:

OF	DF	IF	TF	SF	ZF	AF	PF	CF
X	-	-	-	X	X	X	X	-

LÓGICAS

AND

Formato: AND op_1, op_2

Tipo Args: (r,m) (r,m,i)

Lógica:

$$op_1 \leftarrow op_1 \wedge op_2$$

$$\mathbf{CF = OF = 0}$$

Descripción: Calcula el "y" lógico bit a bit entre op_1 y op_2 .

Banderas:

OF	DF	IF	TF	SF	ZF	AF	PF	CF
0	-	-	-	X	X	?	X	0

OR

Formato: OR op_1, op_2

Tipo Args: (r,m) (r,m,i)

Lógica:

$$op_1 \leftarrow op_1 \vee op_2$$

$$\mathbf{CF = OF = 0}$$

Descripción: Calcula el "o" lógico inclusivo bit a bit entre op_1 y op_2

Banderas:

OF	DF	IF	TF	SF	ZF	AF	PF	CF
0	-	-	-	X	X	?	X	0

XOR

Formato: XOR op_1, op_2

Tipo Args: (r,m) (r,m,i)

Lógica:

$$op_1 \leftarrow op_1 \otimes op_2$$
$$\mathbf{CF = OF = 0}$$

Descripción: Calcula el "o" lógico exclusivo bit a bit entre op_1 y op_2 .

Banderas:

OF	DF	IF	TF	SF	ZF	AF	PF	CF
0	-	-	-	X	X	?	X	0

NOT

Formato: NOT op

Tipo Args: (r,m)

Lógica: $op \leftarrow \neg op$

Descripción: Calcula el complemento a 1 de op. Es decir, cambia unos por cero y ceros por unos.

Banderas:

OF	DF	IF	TF	SF	ZF	AF	PF	CF
-	-	-	-	-	-	-	-	-

CMP

Formato: CMP op_1, op_2

Tipo Args: (r,m) (r,m,i)

Lógica: $op_1 - op_2$

Descripción: Resta op_1 de op_2 pero solo afecta las flags ignorando el resultado. Los operandos quedan inalterados pudiéndose consultar las flags mediante una operación de bifurcación condicional.

Banderas:

OF	DF	IF	TF	SF	ZF	AF	PF	CF
X	-	-	-	X	X	X	X	X

DESPLAZAMIENTO

SAL/SHL

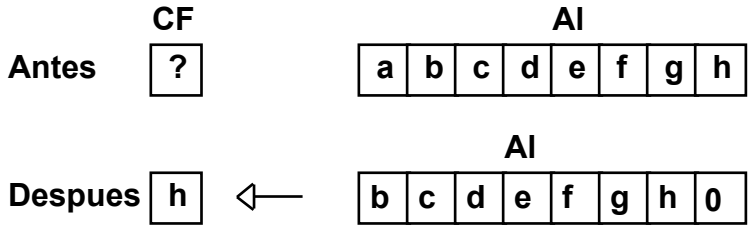
Formato: SAL op_1, op_2

SHL op_1, op_2

Tipo Args: (r,m) (1, CL)

Lógica: corre op_1, op_2 lugares a la izq.

Ej: SAL Al,1



Descripción: Desplaza a la izquierda los bits de op_1 el nro. de bits especificado por op_2. Los bits a la derecha se rellenan con ceros.

Si el nro. de bits a desplazar es 1, se puede especificar directamente. Si es mayor que 1 debe cargarse en CL.

CF contiene luego de la ejecución el ultimo bit de op_1 en ser desplazado "fuera" de la representación.

Banderas:

OF	DF	IF	TF	SF	ZF	AF	PF	CF
X	-	-	-	X	X	?	X	X

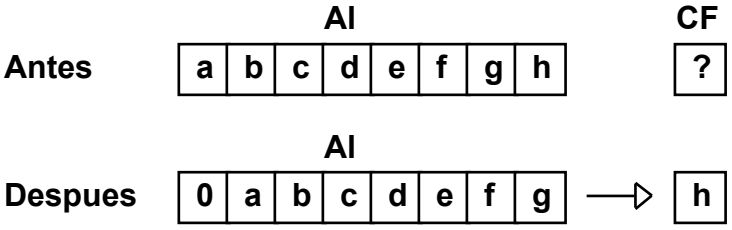
SHR

Formato: SHR op_1, op_2

Tipo Args: (r,m) (1, CL)

Lógica: corre op_1, op_2 lugares a la der.

Ej: SHR Al,1



Descripción: Desplaza a la derecha los bits de op_1 el nro. de bits especificado por op_2. Los bits a la derecha se rellenan con ceros.

Si el nro. de bits a desplazar es 1, se puede especificar directamente. Si es mayor que 1 debe cargarse en CL.

CF contiene luego de la ejecución el ultimo bit de op_1 en ser desplazado "fuera" de la representación.

Banderas:

OF	DF	IF	TF	SF	ZF	AF	PF	CF
X	-	-	-	X	X	?	X	X

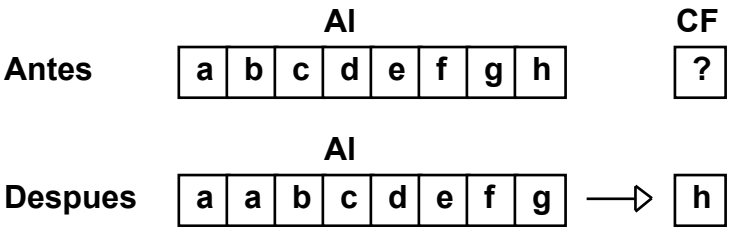
SAR

Formato: SAR op_1, op_2

Tipo Args: (r,m) (1, CL)

Lógica: corre op_1, op_2 lugares a la derecha expandiendo el signo.

Ej: SAR Al,1



Descripción: Desplaza a la derecha los bits de op_1 el nro. de bits especificado por op_2. Los bits a la derecha se rellenan con el signo del primero operando.

Si el nro. de bits a desplazar es 1, se puede especificar directamente. Si es mayor que 1 debe cargarse en CL.

CF contiene luego de la ejecución el ultimo bit de op_1 en ser desplazado "fuera" de la representación.

Banderas:

OF	DF	IF	TF	SF	ZF	AF	PF	CF
X	-	-	-	X	X	?	X	X

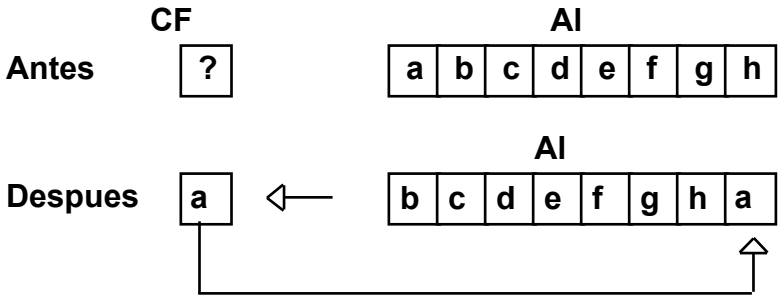
ROL

Formato: ROL op_1, op_2

Tipo Args: (r,m) (1, CL)

Lógica: rota op_1, op_2 lugares a la izquierda.

Ej: **ROL Al,1**



Descripción: Rota a la izquierda los bits de op_1 el numero de bits especificado por op_2

Si el nro. de bits a desplazar es 1, se puede especificar directamente. Si es mayor que 1 debe cargarse en CL.

CF contiene luego de la ejecución el ultimo bit de op_1 en ser desplazado "fuera" de la representación.

OF contiene el xor del CF con el bit más significativo del resultado.

Banderas:

OF	DF	IF	TF	SF	ZF	AF	PF	CF
X	-	-	-	-	-	-	-	X

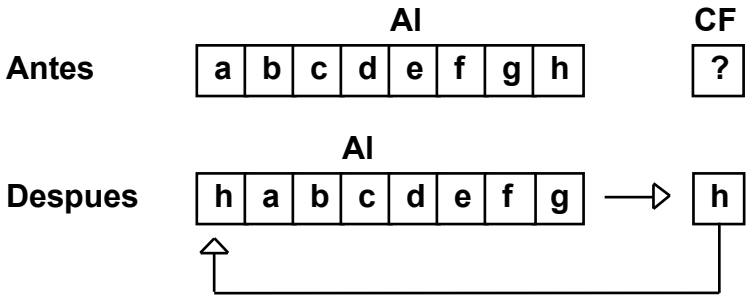
ROR

Formato: ROR op_1, op_2

Tipo Args: (r,m) (1, CL)

Lógica: rota op_1, op_2 lugares a la izquierda.

Ej: **ROR Al,1**



Descripción: Rota a la derecha los bits de op_1 el numero de bits especificado por op_2

Si el nro. de bits a desplazar es 1, se puede especificar directamente. Si es mayor que 1 debe cargarse en CL.

CF contiene luego de la ejecución el ultimo bit de op_1 en ser desplazado "fuera" de la representación.

OF contiene el xor del CF con el bit menos significativo del resultado.

Banderas:

OF	DF	IF	TF	SF	ZF	AF	PF	CF
X	-	-	-	-	-	-	-	X

MOVIMIENTO e I/O

MOV

Formato: MOV op_1, op_2

Tipo Args: (r,m) (r,m,i)

Lógica: $op_1 \leftarrow op_2$

Descripción: Transfiere un byte o una palabra desde el operando fuente al operando destino.
Ambos operandos deben ser del mismo tipo (byte o palabra).
El contenido especificado por el elemento fuente se copia sobre el elemento destino, quedando inalterado el elemento fuente.

Atención: **No se puede mover memoria a memoria.**

Banderas:

OF	DF	IF	TF	SF	ZF	AF	PF	CF
-	-	-	-	-	-	-	-	-

IN

Formato: IN op_1, op_2

Tipo Args: (AL, AX) (i,DX)

Lógica:

si $\mathrm{op}_1 = \mathbf{Al}$
$\mathbf{Al} = \mathrm{I/O}(\mathrm{op}_2)$
sino si $\mathrm{op}_1 = \mathbf{AX}$
$\mathbf{Ax} = \mathrm{I/O}(\mathrm{op}_2)$
fin si

Descripción: Transfiere un byte o una palabra de una puerta de entrada del procesador al registro Al o Ax, respectivamente.

El nro. de la puerta se puede especificar mediante:

- Un valor fijo (de o a 255).
- Un valor variable, el contenido en el registro Dx (de 0 a 65535), pudiéndose acceder a 64K puertas de entrada.

Banderas:

OF	DF	IF	TF	SF	ZF	AF	PF	CF
-	-	-	-	-	-	-	-	-

OUT

Formato: OUT op_1, op_2

Tipo Args: (i,DX) (Al, Ax)

Lógica:

si op_1 = **Al**
    I/O(op_2) = **Al**
sino si op_1 = **Ax**
    I/O(op_2) = **Ax**
fin si

Descripción: Transfiere un byte o una palabra desde el registro Al o Ax a una puerta de salida del procesador

El nro. de la puerta se puede especificar mediante:

- Un valor fijo (de o a 255).
- Un valor variable, el contenido en el registro Dx (de 0 a 65535), pudiéndose acceder a 64K puertas de salida.

Banderas:

OF	DF	IF	TF	SF	ZF	AF	PF	CF
-	-	-	-	-	-	-	-	-

MANEJO DE FLAGS

CLC

Formato: CLC

Lógica: CF := 0

Descripción: Borra la bandera de acarreo (CF) sin afectar a ninguna otra bandera.

Banderas:

OF	DF	IF	TF	SF	ZF	AF	PF	CF
-	-	-	-	-	-	-	-	0

STC

Formato: STC

Lógica: CF := 1

Descripción: Pone en 1 la bandera de acarreo (CF) sin afectar a ninguna otra bandera.

Banderas:

OF	DF	IF	TF	SF	ZF	AF	PF	CF
-	-	-	-	-	-	-	-	1

CLI

Formato: CLI

Lógica: IF := 0

Descripción: Borra la bandera de activación de interrupciones (IF) y desactiva las interrupciones enmascarables (las que aparecen sobre la linea INTR del procesador).

- Las interrupciones enmascarables se pueden activar o desactivar.
- Las interrupciones no enmascarables (las que aparecen sobre la linea NMI) no se pueden desactivar.

Banderas:

OF	DF	IF	TF	SF	ZF	AF	PF	CF
-	-	0	-	-	-	-	-	-

STI

Formato: STI

Lógica: IF := 1

Descripción: Pone en 1 la bandera de activación de interrupciones (IF) y activa las interrupciones enmascarables (las que aparecen sobre la linea INTR del procesador)

Una interrupción pendiente no será reconocida hasta que no se haya ejecutado la instrucción que sigue a STI.

Banderas:

OF	DF	IF	TF	SF	ZF	AF	PF	CF
-	-	1	-	-	-	-	-	-

BIFURCACIÓN INCONDICIONAL

CALL

Formato: CALL op_1

Tipo Args: (R,M,a,A,W)

Lógica:

Si llamada FAR
    PUSH CS
    PUSH IP
    CS := segmento op_1
    IP := offset op_1
sino { llamada NEAR }
    PUSH IP
    IP := op_1
fin si

Descripción: Modifica el flujo de control, cambiando el puntero de programa (CS:IP) a la dirección indicada por op_1, guardando previamente en la pila la dirección de la instrucción siguiente, para volver a esta instrucción una vez ejecutado el procedimiento.

El procedimiento llamado puede estar:

- Dentro del mismo segmento (llamada NEAR). En este caso, se almacena en la pila el desplazamiento de la instrucción siguiente.
- En otro segmento (llamada FAR). En este caso se almacena en la pila primero el segmento y segundo el desplazamiento de la instrucción siguiente.

La llamada puede ser a su vez:

- Directa, es decir op_1 es el valor a asignar a el puntero de instrucción. Se supone siempre un salto NEAR a menos que se indique lo contrario.
- Indirecta, es decir op_1 contiene la dirección de memoria donde se encuentra el valor a asignar al puntero de instrucción. Si se utiliza WORD PTR, el contenido es la nueva dirección o desplazamiento. Si en cambio se utiliza DWORD PTR la primera palabra contiene el desplazamiento y la segunda el segmento. De esta forma se pueden hacer llamadas NEAR o FAR, respectivamente.

Banderas:

OF	DF	IF	TF	SF	ZF	AF	PF	CF
-	-	-	-	-	-	-	-	-

JMP

Formato: JMP op_1

Tipo Args: (R,M,a,A,W)

Lógica:

Si llamada FAR
    CS := segmento op_1
    IP := offset op_1
sino
    IP := op_1
fin si

Descripción: Igual al CALL sin guardar la dirección de retorno en el stack:

Banderas:

OF	DF	IF	TF	SF	ZF	AF	PF	CF
-	-	-	-	-	-	-	-	-

RET

Formato: RET

Lógica:

POP IP
si procedimiento FAR
    POP CS
fin si

Descripción: Retorna de un procedimiento invocado por un CALL, utilizando como dirección de retorno el valor almacenado en el tope del stack.

El ensamblador genera un RET distinto según el procedimiento sea NEAR o FAR:

- SI es NEAR se asume que en el tope del STACK contiene el nuevo valor de IP.
- SI es FAR se asume que en el tope del STACK contiene el nuevo valor de IP y luego esta el nuevo valor de CS.

Banderas:

OF	DF	IF	TF	SF	ZF	AF	PF	CF
-	-	-	-	-	-	-	-	-

BIFURCACIÓN CONDICIONAL

Este conjunto de instrucciones se utilizan para efectuar un salto CONDICIONAL a una dirección de memoria ubicada en el segmento CS a una distancia menor a 128 bytes de la instrucción actual.

Todas verifican cierta condición que deben cumplir algunos bits del registro de flags para realizar la transferencia de control. Si dicha condición no se cumple entonces el salto no se realiza.

JA / JNBE (no considera signo)

Formato: JA op_1 (Jump Above)

JNBE op_1 (Jump Not Below or Equal)

Tipo Args: (d)

Lógica:

Si $CF = 0$ y $ZF = 0$

$IP \leftarrow op_1 + IP$

Descripción: Transfiere el control a la instrucción (IP + op_1) si se cumple la condición CF=0 y ZF=0.

El desplazamiento es un valor con signo de 8 bits, es decir esta comprendido entre -128 y 127.

Ej:

```
CMP a,b      ; comparar a y b
JA  DIR.     ; saltar a DIR si a > b
             ; sin considerar signo
```

Banderas:

OF	DF	IF	TF	SF	ZF	AF	PF	CF
-	-	-	-	-	-	-	-	-

JB / JNAE / JC (no considera signo)

Formato: JB op_1

JNAE op_1

JC op_1

Tipo Args: (d)

Lógica:

Si $\mathrm{CF} = 1$

$\mathrm{IP} \leftarrow op_1 + \mathrm{IP}$

Descripción: Transfiere el control a la instrucción (IP + op_1) si se cumple la condición CF=1 .

El desplazamiento es un valor con signo de 8 bits, es decir esta comprendido entre -128 y 127.

Ej:

```
CMP a,b        ; comparar a y b
JC  DIR.       ; saltar a DIR si a < b    ; sin considerar signo
```

Banderas:

OF	DF	IF	TF	SF	ZF	AF	PF	CF
-	-	-	-	-	-	-	-	-

JNB / JAE / JNC (no considera signo)

Formato: JNB op_1

JAE op_1

Tipo Args: (d)

Lógica:

Si $\mathrm{CF} = 0$

$\mathrm{IP} \leftarrow op_1 + \mathrm{IP}$

Descripción: Transfiere el control a la instrucción (IP + op_1) si se cumple la condición CF=0 .

El desplazamiento es un valor con signo de 8 bits, es decir esta comprendido entre -128 y 127.

Ej:

```
CMP a,b        ; comparar a y b
JNB  DIR.      ; saltar a DIR si a ≥ b
               ; sin considerar signo
```

Banderas:

OF	DF	IF	TF	SF	ZF	AF	PF	CF
-	-	-	-	-	-	-	-	-

JBE/JNA (no considera signo)

Formato: JBE op_1

JNA op_1

Tipo Args: (d)

Lógica:

Si $\mathrm{CF} = 1$ o $\mathrm{ZF} = 1$

$\mathrm{IP} \leftarrow op_1 + \mathrm{IP}$

Descripción: Transfiere el control a la instrucción (IP + op_1) si se cumple la condición CF=1 o ZF=1.

El desplazamiento es un valor con signo de 8 bits, es decir esta comprendido entre -128 y 127.

Ej:

```
CMP a,b      ; comparar a y b
JNA  DIR.    ; saltar a DIR si a ≤ b
             ; sin considerar signo
```

Banderas:

OF	DF	IF	TF	SF	ZF	AF	PF	CF
-	-	-	-	-	-	-	-	-

JE/JZ

Formato: JE op_1

JZ op_1

Tipo Args: (d)

Lógica:

Si $\mathrm{ZF} = 1$

$\mathrm{IP} \leftarrow op_1 + \mathrm{IP}$

Descripción: Transfiere el control a la instrucción (IP + op_1) si se cumple la condición ZF=1 .

El desplazamiento es un valor con signo de 8 bits, es decir esta comprendido entre -128 y 127.

Ej:

```
CMP a,b      ; comparar a y b
JC  DIR.     ; saltar a DIR si a = b
```

Banderas:

OF	DF	IF	TF	SF	ZF	AF	PF	CF
-	-	-	-	-	-	-	-	-

JG/JNLE (considera signo)

Formato: JG op_1

JNLE op_1

Tipo Args: (d)

Lógica:

Si $ZF = 0$ y $SF = OF$
$IP \leftarrow op_1 + IP$

Descripción: Transfiere el control a la instrucción (IP + op_1) si se cumple la condición ZF=0 y SF=OF .

El desplazamiento es un valor con signo de 8 bits, es decir esta comprendido entre -128 y 127.

Ej:

```
CMP a,b      ; comparar a y b
JG  DIR.     ; saltar a DIR si a > b
             ; considerando el signo
```

Banderas:

OF	DF	IF	TF	SF	ZF	AF	PF	CF
-	-	-	-	-	-	-	-	-

JNG/JLE (considera signo)

Formato: JNG op_1

JLE op_1

Tipo Args: (d)

Lógica:

Si $ZF = 1$ o $SF \neq OF$
$IP \leftarrow op_1 + IP$

Descripción: Transfiere el control a la instrucción (IP + op_1) si se cumple la condición ZF=1 y SF≠OF .

El desplazamiento es un valor con signo de 8 bits, es decir esta comprendido entre -128 y 127.

Ej:

```
CMP a,b      ; comparar a y b
JNG  DIR.    ; saltar a DIR si a ≤ b
             ; considerando el signo
```

Banderas:

OF	DF	IF	TF	SF	ZF	AF	PF	CF
-	-	-	-	-	-	-	-	-

JNE/JNZ

Formato: JNE op_1

JNZ op_1

Tipo Args: (d)

Lógica:

Si $\mathrm{ZF} = 0$

$\mathrm{IP} \leftarrow op_1 + \mathrm{IP}$

Descripción: Transfiere el control a la instrucción (IP + op_1) si se cumple la condición ZF=0 .

El desplazamiento es un valor con signo de 8 bits, es decir esta comprendido entre -128 y 127.

Ej:

```
CMP a,b      ; comparar a y b
JNE  DIR.    ; saltar a DIR si a ≠ b
```

Banderas:

OF	DF	IF	TF	SF	ZF	AF	PF	CF
-	-	-	-	-	-	-	-	-

JNO

Formato: JNO op_1

Tipo Args: (d)

Lógica:

Si $\mathrm{OF} = 0$

$\mathrm{IP} \leftarrow op_1 + \mathrm{IP}$

Descripción: Transfiere el control a la instrucción (IP + op_1) si se cumple la condición OF=0 .

El desplazamiento es un valor con signo de 8 bits, es decir esta comprendido entre -128 y 127.

Ej:

```
ADD a,b      ; a=a+b
JNO  DIR.    ; saltar a DIR si no hubo overflow
```

Banderas:

OF	DF	IF	TF	SF	ZF	AF	PF	CF
-	-	-	-	-	-	-	-	-

JNS

Formato: JNS op_1

Tipo Args: (d)

Lógica:

Si $\mathrm{SF} = 0$
$\mathrm{IP} \leftarrow op_1 + \mathrm{IP}$

Descripción: Transfiere el control a la instrucción (IP + op_1) si se cumple la condición SF=0 .

El desplazamiento es un valor con signo de 8 bits, es decir esta comprendido entre -128 y 127.

Ej:

```
CMP a,b        ; comparar a y b
JNS  DIR.      ; saltar a DIR si a ≥ b
```

Banderas:

OF	DF	IF	TF	SF	ZF	AF	PF	CF
-	-	-	-	-	-	-	-	-

JO

Formato: JO op_1

Tipo Args: (d)

Lógica:

Si $\mathrm{OF} = 1$
$\mathrm{IP} \leftarrow op_1 + \mathrm{IP}$

Descripción: Transfiere el control a la instrucción (IP + op_1) si se cumple la condición OF=1.

El desplazamiento es un valor con signo de 8 bits, es decir esta comprendido entre -128 y 127.

Ej:

```
ADD a,b        ; a=a+b
JO  DIR.       ; saltar a DIR si hubo overflow
```

Banderas:

OF	DF	IF	TF	SF	ZF	AF	PF	CF
-	-	-	-	-	-	-	-	-

JS

Formato: JS op_1

Tipo Args: (d)

Lógica:

Si $\mathrm{SF} = 1$
$\mathrm{IP} \leftarrow op_1 + \mathrm{IP}$

Descripción: Transfiere el control a la instrucción (IP + op_1) si se cumple la condición SF=0 .

El desplazamiento es un valor con signo de 8 bits, es decir esta comprendido entre -128 y 127.

Ej:

```
CMP a,b      ; comparar a y b
JNS  DIR.    ; saltar a DIR si a < b
```

Banderas:

OF	DF	IF	TF	SF	ZF	AF	PF	CF
-	-	-	-	-	-	-	-	-

MANEJO DE STACK

El stack del 8086 está implementado utilizando fundamentalmente las operaciones PUSH y POP conjuntamente con los registros SS y SP.

El tope del stack está apuntado por SS:SP y las operaciones PUSH y POP los actualizan de forma de obtener la semántica de stack como veremos más adelante.

PUSH

Formato: PUSH op_1

Tipo Args: (R,M)

Lógica:

SP=SP-2
MOV SS:[SP],op_1

Descripción: Decrementa el puntero del stack, SP y transfiere la palabra especificada por op_1 a la dirección SS:SP

Banderas:

OF	DF	IF	TF	SF	ZF	AF	PF	CF
-	-	-	-	-	-	-	-	-

POP

Formato: POP op_1

Tipo Args: (R,M)

Lógica:

MOV op_1,SS:[SP]
SP=SP+2

Descripción: Transfiere la palabra en tope del stack al operando op_1 y luego incrementa el puntero del stack, SP.

Banderas:

OF	DF	IF	TF	SF	ZF	AF	PF	CF
-	-	-	-	-	-	-	-	-

PUSHF

Formato: PUSHF

Lógica:

SP=SP-2
MOV SS:[SP],FLAGS

Descripción: Coloca el registro de flags en al tope del stack.

Banderas:

OF	DF	IF	TF	SF	ZF	AF	PF	CF
-	-	-	-	-	-	-	-	-

POPF

Formato: POPF

Lógica:

```
SP=SP-2
MOV FLAGS, SS:[SP]
```

Descripción: Restaura los registros de flags desde el tope del stack.

Banderas:

OF	DF	IF	TF	SF	ZF	AF	PF	CF
X	X	X	X	X	X	X	X	X

INTERRUPCIONES

La tabla de interrupciones del 8086 se encuentra en la dirección absoluta 0 y ocupa el primer kilobyte de memoria.

Dicha tabla posee 256 entradas de 4 bytes cada una. La entrada i-ésima de esta tabla posee la dirección de memoria del manejador de la interrupción i. Los primeros 2 bytes corresponden al offset del manejador y los últimos 2 corresponden al segmento del mismo.

INT

Formato: INT op_1

Tipo Args: (i)

Lógica:

```
PUSHF
IF=0
TF=0
CALL FAR tabla_int( op1 )
```

Descripción: Genera una interrupción software tipo op_1.

Banderas:

OF	DF	IF	TF	SF	ZF	AF	PF	CF
-	-	x	x	-	-	-	-	-

IRET

Formato: IRET

Lógica:

```
RET
POPF
```

Descripción: Retorna de una interrupción. Funciona en forma equivalente al RET utilizado para retornar de una llamada CALL con la diferencia que esta instrucción restaura las flags del stack antes de retornar.

Banderas:

OF	DF	IF	TF	SF	ZF	AF	PF	CF
X	x	x	x	x	x	x	x	x