CONJUNTO DE INSTRUCCIONES (Microprocesadores *8086*/*8088*)

Se pueden clasificar en los siguientes grupos:

Instrucciones de *Transferencia de Datos*.

Estas instrucciones mueven datos de una parte a otra del sistema; desde y hacia la memoria principal, de y a los registros de datos, puertos de ***E/S*** y registros de segmentación.

Las instrucciones de transferencia de datos son las siguientes:

- ***MOV*** transfiere
- ***XCHG*** intercambia
- ***IN*** entrada
- ***OUT*** salida
- ***XLAT*** traduce usando una tabla
- ***LEA*** carga la dirección efectiva
- ***LDS*** carga el segmento de datos
- ***LES*** carga el segmento extra
- ***LAHF*** carga los indicadores en ***AH***
- ***SAHF*** guarda ***AH*** en los indicadores
- ***PUSH*** **FUENTE** (***sp***) ***fuente***
- ***POP*** **DESTINO** ***destino*** (***sp***)

Control de *Bucles* (instrucciones simples)

Estas posibilitan el grupo de control más elemental de nuestros programas. Un bucle es un bloque de código que se ejecuta varias veces. Hay **4** tipos de bucles básicos:

- Bucles ***sin fin***
- Bucles por ***conteo***
- Bucles ***hasta***
- Bucles ***mientras***

Las instrucciones de control de bucles son las siguientes:

· ***INC*** **incrementar**
· ***DEC*** **decrementar**
· ***LOOP*** **realizar un bucle**
· ***LOOPZ,LOOPE*** **realizar un bucle *si es cero***
· ***LOOPNZ,LOOPNE*** **realizar un bucle *si no es cero***
· ***JCXZ*** **salta si *CX* es *cero***

Instrucciones de *Prueba, Comparación y Saltos*.

Este grupo es una continuación del anterior, incluye las siguientes instrucciones:

- ***TEST*** **verifica**
- ***CMP*** **compara**
- ***JMP*** **salta**

- ***JE***, ***JZ*** **salta** si ***es igual a cero***
- ***JNE***, ***JNZ*** **salta** si ***no igual a cero***
- ***JS*** **salta** si ***signo negativo***
- ***JNS*** **salta** si ***signo no negativo***
- ***JP***, ***JPE*** **salta** si ***paridad par***
- ***JNP***, ***JOP*** **salta** si ***paridad impar***
- ***JO*** **salta** si ***hay capacidad excedida***
- ***JNO*** **salta** si ***no hay capacidad excedida***
- ***JB***, ***JNAE*** **salta** si ***por abajo*** (no encima o igual)
- ***JNB***, ***JAE*** **salta** si ***no está por abajo*** (encima o igual)
- ***JBE***, ***JNA*** **salta** si ***por abajo o igual*** (no encima)
- ***JNBE***, ***JA*** **salta** si ***no por abajo o igual*** (encima)
- ***JL***, ***JNGE*** **salta** si ***menor que*** (no mayor o igual)
- ***JNL***, ***JGE*** **salta** si ***no menor que*** (mayor o igual)
- ***JLE***, ***JNG*** **salta** si ***menor que o igual*** (no mayor)
- ***JNLE***, ***JG*** **salta** si ***no menor que o igual*** (mayor)

Instrucciones de *Llamado y Retorno de Subrutinas.*

Para que los programas resulten eficientes y legibles tanto en lenguaje ensamblador como en lenguaje de alto nivel, resultan indispensables las subrutinas:

- ***CALL llamada*** a subrutina
- ***RET retorno*** al programa o subrutina que llamó

Instrucciones *Aritméticas.*

Estas instrucciones son las que realiza directamente el ***8086/8088***

a. Grupo de adición:

- ***ADD*** **suma**
- ***ADC*** **suma** con acarreo
- ***AAA*** ajuste **ASCII** para la **suma**
- ***DAA*** ajuste decimal para la **suma**

b. Grupo de sustracción:

- ***SUB*** **resta**
- ***SBB*** **resta** con acarreo negativo
- ***AAS*** ajuste **ASCII** para la **resta**
- ***DAS*** ajuste decimal para la **resta**

c. Grupo de multiplicación:

- ***MUL*** **multiplicación**
- ***IMUL*** **multiplicación** entera
- ***AAM*** ajuste **ASCII** para la **multiplicación**

d. Grupo de división:

- ***DIV*** **división**

- ***IDIV* división** entera
- ***AAD*** ajuste **ASCII** para la **división**

e. Conversiones:

- ***CBW* pasar** octeto a palabra
- ***CWD* pasar** palabra a doble palabra
- ***NEG* negación**

f. Tratamiento de cadenas:

Permiten el movimiento, comparación o búsqueda rápida en bloques de datos:

- ***MOVC* transferir *carácter*** de una cadena
- ***MOVW* transferir *palabra*** de una cadena
- ***CMPC* comparar *carácter*** de una cadena
- ***CMPW* comparar *palabra*** de una cadena
- ***SCAC* buscar *carácter*** de una cadena
- ***SCAW* buscar *palabra*** de una cadena
- ***LODC* cargar *carácter*** de una cadena
- ***LODW* cargar *palabra*** de una cadena
- ***STOC* guardar *carácter*** de una cadena
- ***STOW* guardar *palabra*** de una cadena
- ***REP* repetir**
- ***CLD* poner** a **0** el indicador de dirección
- ***STD* poner** a **1** el indicador de dirección

Instrucciones *Lógicas*.

Son operaciones **bit** a **bit** que trabajan sobre **octetos** o **palabras** completas:

- ***NOT* negación**
- ***AND* producto** lógico
- ***OR* suma** lógica
- ***XOR* suma** lógica exclusiva

Instrucciones de *Desplazamiento, Rotación y Adeudos*.

Básicamente permiten ***multiplicar*** y ***dividir*** por potencias de **2**

- ***SHL*, *SAL* desplazar** a la ***izquierda*** (desplazamiento aritmético)
- ***SHR* desplazar** a la ***derecha***
- ***SAR* desplazamiento** aritmético a la ***derecha***
- ***ROL* rotación** a la ***izquierda***
- ***ROR* rotación** a la ***derecha***
- ***RCL* rotación** con acarreo a la ***izquierda***
- ***RCR* rotación** con acarreo a la ***derecha***
- ***CLC* borrar** acarreo
- ***STC* poner** acarreo a **1**

Instrucciones de *Pila*.

Una de las funciones de la pila del sistema es la de salvaguardar (conservar) datos (la otra es la de salvaguardar las direcciones de retorno de las llamadas a subrutinas):

- ***PUSH* introducir**
- ***POP* extraer**
- ***PUSHF* introducir** indicadores
- ***POPF* extraer** indicadores

Instrucciones de *Control del microprocesador*.

Hay varias instrucciones para el control de la ***CPU***, ya sea a ella sola, o en conjunción con otros procesadores:

- ***NOP* no operación**
- ***HLT* parada**
- ***WAIT* espera**
- ***LOCK* bloquea**
- ***ESC* escape**

Instrucciones de *Interrupción*.

- ***STI* poner** a **1** el indicador de interrupción
- ***CLI* borrar** el indicador de interrupción
- ***INT* interrupción**
- ***INTO* interrupción** por capacidad excedida (desbordamiento)
- ***IRET* retorno** de interrupción

Las instrucciones de ***transferencia condicional del control*** del programa se pueden clasificar en **3** grupos:

1. Instrucciones usadas para comparar dos enteros sin signo:

a. ***JA*** o ***JNBE***. **Salta *si está arriba*** o **salta *si no está abajo o si no es igual*** (*jump if above* o *jump if not below or equal*) El salto se efectúa si la bandera ce ***CF* = 0** o si la bandera de ***ZF* = 0**

b. ***JAE*** o ***JNB***. **Salta *si está arriba o es igual*** o **salta *si no está abajo*** (*jump if above or equal* o *jump if not below*) El salto se efectúa si ***CF* = 0**.

c. ***JB*** o ***JNAE***. **Salta *si está abajo*** o **salta *si no está arriba o si no es igual*** (*jump if below or equal* o *jump if not above or equal*) El salto se efectúa si ***CF*=1**.

d. ***JBE*** o ***JNA***. **Salta *si está abajo o si es igual*** o **salta *si no está arriba*** (*jump if below or equa* o *jump if not above*) El salto se efectúa si ***CF* = 1**.

e. ***JE*** o ***JZ***. **Salta *si es igual*** o **salta *si es cero*** (*jump equal* o *jump if zero*) El salto se efectúa si ***ZF* = 1** (también se aplica a comparaciones de enteros con signo)

f. ***JNE*** o ***JNZ***. **Salta *si no es igual*** o **salta *si no es cero*** (*jump if not equal* o *jump if not zero*) El salto se efectúa si ***ZF* = 0** (también se aplica a comparaciones de enteros con signo)

2. Instrucciones usadas para comparar dos enteros con signo:

a. ***JG*** o ***JNLE***. **Salta *si es más grande*** o **salta *si no es menor o igual*** (*jump if greater* o *jump if not less or equal*) El salto se efectúa si ***ZF* = 0** o ***OF* = *SF***.

b. ***JGE*** o ***JNL***. **Salta *si es más grande o igual*** o **salta *si no es menor que*** (*jump if greater or equal* o *jump if not less*) El salto se efectúa si ***SF = OF***.

c. ***JL*** o ***JNGE***. **Salta *si es menor que*** o **salta *si no es mayor o igual*** (*jump if less* o *jump if not greater or equal*) El salto se efectúa si ***SF = OF***.

d. ***JLE*** o ***JNG***. **Salta *si es menor o igual*** o **salta *si no es más grande*** (*jump if less or equal* o *jump if not greater*) El salto se efectúa si ***ZF* = 1** o ***SF = OF***.

3. Instrucciones usadas según el estado de banderas:

a. ***JC*** **Salta *si hay acarreo*** (*jump if carry*) El salto se efectúa si ***CF* = 1**.

b. ***JNC*** **Salta *si no hay acarreo*** (*jump if not carry*) El salto se efectúa si ***CF* = 0**.

c. ***JNO*** **Salta *si no hay desbordamiento*** (*jump if not overflow*) El salto se efectúa si ***OF* = 0**.

d. ***JNP*** o ***JPO*** **Salta *si no hay paridad*** o **salta *si la paridad en non***. El salto se efectúa si ***PF*= 0**.

e. ***JNS*** **Salta *si el signo está apagado*** (*jump if not sign*) El salto se efectúa si ***SF* = 0**.

f. ***JO*** **Salta *si hay desbordamiento*** (*jump if overflow*) El salto se efectúa si ***OF* = 1**.

g. ***JP*** o ***JPE*** **Salta *si hay paridad*** o **salta *si la paridad es par*** (*jump if parity* o *jump if parity even*) El salto se efectúa si ***PF* = 1**.

h. ***JS*** **Salta *si el signo está prendido*** (*jump if sign set*) El salto se efectúa si ***SF* = 1**.

Las comparaciones con signo van de acuerdo con la interpretación que usted le quiera dar a los bytes o palabras de su programa. Por ejemplo, suponga que tiene un byte cuyo valor es **11111111** en binario y que desea compararlo con otro cuyo valor es **00000000**. ¿Es **11111111** mayor que **00000000**? ***SÍ*** y ***NO***, eso depende de la interpretación que usted le quiera dar. Si trabaja con números enteros ***sin signo SÍ LO SERÁ***, pues **255** es mayor que **0**. Por el contrario, si tiene ***signo*** entonces ***SERÁ MENOR*** puesto que **−1** es siempre menor que **0**.

Lo anterior lleva a seleccionar las instrucciones de comparación y de salto de acuerdo con la interpretación que se les dé a los bytes o palabras; reflexione sobre este punto.

Los saltos condicionales se encuentran limitados al rango de **−128** a **+127** bytes como máxima distancia, ya sea adelante o hacia atrás. Si desea efectuar un salto a mayores distancias es necesario crear una condición mixta entre saltos condicionales y no condicionales.

Iteraciones.

Con los saltos condicionales y no condicionales se pueden crear estructuras de iteración bastante complejas, aunque existen instrucciones específicas para ello tal como ***loop***.

Esta instrucción es muy útil cuando se va a efectuar cierto bloque de instrucciones un número finito de veces. He aquí un ejemplo:

CUENTA: DW, 100

.

.

.

MOV CX, CUENTA

ITERA:

.

.

LOOP ITERA

El bloque de instrucciones que se encuentra entre la etiqueta ***ITERA*** y la instrucción ***loop*** será ejecutado hasta que el registro ***CX*** sea igual a **0**. Cada vez que se ejecuta la instrucción ***loop***, el registro ***CX*** es decrementado en **1** hasta llegar a **0**.

Esta instrucción tiene la limitante de que debe encontrarse en el rango de **+128** a **−127** (máximo número de bytes entre ***ITERA*** y ***loop***)

Iteraciones condicionales

Existen otras dos variantes de la instrucción ***loop***. Las instrucciones ***loope*** y ***loopz*** decrementan ***CX*** e iteran si ***CX* = 0** y ***ZF* = 1**, mientras que ***loopne*** y ***looppnz*** iteran si ***CX"* 0** y ***ZF"* 0**. Un punto importante es que al decrementarse ***CX*** las banderas **NO RESULTAN AFECTADAS**. Por lo tanto, le corresponde a usted afectarlas

dentro del bloque de iteración.

FORMATO DE LAS INSTRUCCIONES

Cada instrucción en lenguaje ensamblador del ***8088*** está compuesta de **4** campos: **etiqueta operación operando comentario.**

El campo comentario se utiliza para propósitos de documentación y es opcional. **Campo *etiqueta***: Una ***etiqueta*** debe comenzar con un carácter alfabético y puede contener hasta **31** caracteres, incluyendo:

- Letras de la ***A*** a la ***Z***
- Números del ***0*** al ***9***
- Los símbolos especiales: **− $. @ %**

No se puede utilizar un nombre que coincida con una palabra reservada o directiva del ensamblador. Si el nombre incluye un ***punto***, entonces el ***punto*** debe ser el primer carácter.

Campo *operación*: Contiene el ***nemotécnico*** de la instrucción, que es de **2** a **6** caracteres.

Campo *operando*: Contiene la posición o posiciones donde están los ***datos*** que van a ser manipulados por la instrucción.

Campo *comentario*: Se utiliza para ***documentar*** el código fuente del ensamblador. Debe separarse del último campo por al menos un espacio e iniciar con **;**.

Cuando inicia un ***comentario*** en una línea ésta deberá tener en la primera columna el carácter **;**.

MODOS DE DIRECCIONAMIENTO Y GENERACIÓN DEL CÓDIGO OBJETO

Generación de la dirección de la instrucción.

Todos los registros internos del ***8086/8088*** son de **16** bits. El bus de ***dirección*** es de **20** bits, por lo que se usa más de un registro interno para generar la ***dirección*** de **20** bits.

Los **2** registros usados para la ***dirección*** de la instrucción son el ***IP*** y el ***CS***. Se combinan en una forma especial para generar la ***dirección*** de **20** bits. ***dirección* de 20 bits = 1610 * *CS* + *IP***

Por **ejemplo**: Si los registros ***CS*** e ***IP*** contienen los valores:

***CS* = 1000H**

***IP* = 0414 H**

La ***dirección*** de **20** bits es:

1610 * 1000H + 0414H = 10000H + 0414H = 10414H

Esta es la ***dirección*** en memoria desde la cual la nueva instrucción debe buscarse. Al registro ***IP*** se le refiere como ***offset***, el registro ***CS* * 1610** apunta a la ***dirección*** de nicio o segmento en memoria desde el cual se calcula el ***offset***. La ***Figura A*** muestra gráficamente cómo se calcula la ***dirección*** de **20** bits.

FIGURA A. Calculo de la dirección de 20 bits

Cada ***dirección*** generada por el ***8086/8088*** usa uno de los **4** registros de segmento. Este registro de segmento es recorrido **4** bits hacia la izquierda antes de ser sumado al ***offset***.

La instrucción del ***CPU*** especifica cuáles registros internos se usan para generar el ***offset***. Vamos a ver los diferentes modos de direccionamiento tomando como ejemplo la instrucción ***MOV***.

Instrucción *MOV*

Transfiere un byte desde el ***operando fuente*** al ***operando destin***o. Tiene el siguiente formato:

***MOV* destino, fuente**

Direccionamiento *Inmediato*

El ***operando fuente*** aparece en la instrucción. Un **ejemplo**, es el que mueve un valor constante a un registro interno.

***MOV* AX, 568**

Direccionamiento a *Registro*

Indica que el ***operando*** a ser usado está contenido en uno de los registros internos de propósito general del ***CPU***. En el caso de los registros ***AX***, ***BX***, ***CX*** o ***DX*** los registros pueden ser de **8** a **16** bits

Ejemplos:

***MOV* AX, BX ; AX . BX**

***MOV* AL, BL ; AL . BL**

Cuando usamos direccionamiento a registro, el ***CPU*** realiza las operaciones internamente, es decir, no se genera ***dirección*** de **20** bits para especificar el ***operando fuente***.

Direccionamiento *Directo*

Especifica en la instrucción la localidad de memoria que contiene al ***operando***. En este tipo de direccionamiento, se forma una ***dirección*** de **20** bits.

Ejemplo:

***MOV* CX, COUNT**

El valor de ***COUNT*** es una constante. Es usada como el valor ***offset*** en el cálculo de la ***dirección*** de **20** bits

El ***8086/8088*** siempre usa un registro de segmento cuando calcula una ***dirección*** física. **¿Cuál registro se debe usar para esta instrucción? Respuesta**: ***DS***

En la ***Figura B***, se muestra el cálculo de la ***dirección*** desde la cual se tomará el dato que se carga en ***CX***.

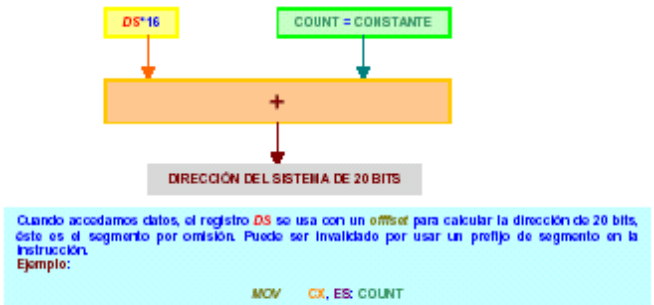


FIGURA B. Uso del segmento de datos y una constante para desplazamiento

Este es el segmento por omisión que se usa. Sin embargo, cualquiera de los **4** segmentos puede usarse. Esto se efectúa especificando el registro apropiado en la instrucción.

Por **ejemplo**, suponga que deseamos usar el registro ***ES*** en lugar del ***DS***:

***MOV* CX, ES: COUNT**

Direccionamiento de *Registro Indirecto*

Con el modo de direccionamiento de registro índice, la ***dirección offset*** de **16** bits está contenida en un registro base o registro índice. Esto es, la ***dirección*** reside en el registro ***BX***, ***BP***, ***SI*** o ***DI***.

Ejemplo:

MOV **AX, [SI]**

El valor de **16** bits contenido en el registro ***SI*** debe ser el ***offset*** usado para calcular la ***dirección*** de **20** bits.

Otra vez, debe usarse un registro de segmento para generar la ***dirección*** final. El valor de **16** bits en ***SI*** se combina con el segmento apropiado para generar la ***dirección***.

Direccionamiento de *Registro Indirecto con Desplazamiento*

Este tipo de direccionamiento incluye a los dos modos de direccionamiento anteriores. La ***dirección offset*** de **16** bits se calcula sumando el valor de **16** bits especificado en un registro interno y una constante.

Por **ejemplo**, si usamos el registro interno ***DI*** y el valor constante (desplazamiento), donde ***COUNT*** ha sido previamente definido, el ***nemotécnico*** para esta construcción es:

MOV **AX, COUNT [DI]**

Si: **COUNT = 0378H**

DI = 04FAH

0872H

Entonces, la ***dirección offset*** de **16** bits es **0872H**

Direccionamiento de *Registro Indirecto con un Registro Base y un Registro Índice*

Este modo de direccionamiento usa la suma de dos registros internos para obtener la ***dirección offset*** de **16** bits a usarse en el cálculo de la ***dirección*** de **20** bits.

Ejemplos:

MOV **[BP] [DI], AX** ; el ***offset*** es **BP + DI**

MOV **AX, [BX] [SI]** ; el ***offset*** es **BX + SI**

Direccionamiento de Registro Índice Indirecto con un Registro Base, un Registro Índice y un Registro Constante

Este es el modo de direccionamiento más complejo. Es idéntico al modo de direccionamiento anterior, excepto que se suma una constante.

Ejemplo: Suponga que tenemos los siguientes valores en los registros:

DI = 0367H

BX = 7890H

COUNT = 0012H

7C09H

Este modo de direccionamiento indica que el ***offset*** especificado por la suma de ***DI + BX***

+ *COUNT* sea usado para mover el dato en memoria en el registro ***AX***.

***MOV* AX, COUNT [BX] [DI]**

La ***dirección offset*** de **16** bits es **7C09H**. La ***dirección*** completa en **20** bits se calcula de la expresión:

1610*DS + 7C09H

Si el ***DS*** contiene **3000H**, la dirección completa de **20** bits es:

3000H + 7C09H = 37C09H

Código Objeto del *8086/8088*

Como programador, debes escribir los ***nemotécnicos***. El ***código objeto*** es generado por la computadora (son los bytes que ejecuta el ***CPU***) Con el conjunto de instrucciones del ***8086/8088***, cada tipo de modo de direccionamiento puede requerir un número diferente de bytes. En los ejemplos siguientes proporcionaremos el número de bytes requeridos por cada modo de direccionamiento.

Bit *W* y campo *REG*

La instrucción ***MOV* AX, 568H**

Indica mover inmediatamente al registro interno ***AX*** el valor **568H**. El registro interno puede ser de **1** byte o de una palabra. Esta instrucción requiere **2** o **3** bytes, como se indica en la ***Figura C***.

FIGURA C. Uso del bit *W* y del campo *REG*.

El primer byte contiene los *bits más significativos* (***MSB***) como **1011**. El próximo bit es ***W***. **W** indica: **1 para *word* 0 para *byte.*** Esto es, si el registro destino es de **16** bits o de **8** bits.

Los siguientes **3** bits del primer byte, campo ***REG***, determinan cuál registro está involucrado. La ***Figura D***, muestra el código de selección del registro.

REG	REGISTRO DE 16 BITS	REGISTRO DE 8 BITS
000	AX	AL
001	CX	CL
010	DX	DL
011	BX	BL
100	SP	AH
101	BP	CH

110	SI	DH
111	DI	BH

FIGURA D. Registro involucrado en la operación

Campo *DATA*. Si el registro de destino es de **1** byte, el dato debe estar en el segundo byte de la instrucción. Si el destino es de una palabra, el segundo byte de la instrucción son los **8** *bits menos significativos* (***lsb***) del ***dato***, el tercer byte de la instrucción son los **8** *bits más significativos* (***MSB***) del ***dato***. La siguiente tabla, muestra los ***nemotécnicos* 2** o **3** bytes

NEMOTÉCNICO	CÓDIGO OBJETO
MOV AX, 568	Instrucción de 3 bytes B8 68 05
MOV AL, 56	instrucción de 2 bytes B0 56

Bit *D, MOD* y *R/M*

En este ejemplo, moveremos ***datos*** desde memoria o moveremos un ***registro*** hacia o desde otro ***registro***. Usaremos una instrucción como:

***MOV* AX, BX**

Esta instrucción es de **2** bytes porque no nos referimos a memoria. Los bytes aparecerán

como lo muestra la ***Figura E***:

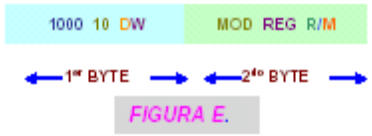


FIGURA E.

El primer byte contiene los 2 *bits menos significativos* como ***DW***. El bit ***W*** es para **word=1** o para **byte=0**. La ***D*** es para indicar si el ***dato*** será almacenado en el operando especificado por los campos ***MOD*** y ***R/M*** (***D* = 0**) o si va a ser almacenado en el registro especificado por el campo ***REG*** (***D* = 1**) La ***Figura F*** muestra las asignaciones para ***MOD*** y ***R/M***. Note en la descripción de ***MOD*=11**, el campo ***R/M*** es codificado con un formato de registro. Este formato se mostró en la ***Figura D***.

Registros base e índice especificados por *R/M* para operandos en memoria (*MOD* <> 11)

R/M	REGISTRO BASE	REGISTRO INDICE
000	BX	SI
001	BX	DI
010	BP	SI
011	BP	DI
100	NINGUNO	SI
101	NINGUNO	DI
110	BP	NINGUNO
111	BX	NINGUNO

MOD	DESPLAZAMIENTO	COMENTARIO
00	CERO	
01	8 BITS contenido del próximo byte de la instrucción, signo extendido a 16 bits	La instrucción contiene un byte adicional
10	16 bits contenidos en los próximos 2 bytes de la instrucción	La instrucción contiene 2 bytes adicionales
11	Registro R/M	

Si MOD = 00 y R/M = 110, entonces
1. Lo expuesto arriba no se aplica
2. La instrucción contiene 2 bytes adicionales
3. La dirección offset es contenida en esos bytes

FIGURA F. Definiciones para el código objeto del *8086/8088* de los campos MOD y R/M

Para esta instrucción deseamos almacenar el ***dato*** en el registro ***AX***. Por lo tanto el bit ***D*=0**. Esto significa que el dato debe ser almacenado en la localidad especificada por los campos ***MOD*** y ***R/M***. Por lo tanto, ***MOD* = 11**. El campo ***R/M* = 000**, indicando que el registro ***AX*** es el destino para los datos. El campo ***REG*** para el segundo byte de datos es **011**. indicando que el registro ***BX*** es el registro fuente a ser utilizado. El segundo byte de la instrucción es **11 011 000 = D8**. Por lo que el código objeto para la instrucción es:

***MOV* AX, BX es 89**

D8

Código Objeto para el uso de *Registro Base y Registro Índice*

Examinemos un último ejemplo para generar código objeto para el ***8086/8088***. En éste vamos a calcular el ***código objeto*** para la instrucción:

***MOV* CX, COUNT [BX] [SI]**

Esta instrucción es de **4** bytes, como se muestra en la ***Figura G***:

FIGURA G. Formato del *código objeto* para una instrucción como: *MOV* CX, COUNT [BX] [SI]

El primer byte de la ***Figura G***, debe tener el bit ***D*=1**. Esto es debido a que el destino para el dato debe ser especificado por el campo ***REG*** en el segundo byte. El bit ***W*=1**. porque es una transferencia de palabra. El primer byte es:

10001011 = 8B

En el segundo byte, ya que estamos usando una constante que requiere **16** bits, el campo ***MOD* = 10**. Refiriendo a la ***Figura F***, ésta indica que el desplazamiento debe ser formateado en **2** bytes y deben seguir a este segundo byte. El próximo campo para el segundo byte es el campo de registro (***REG***) Ya que debemos usar el registro ***CX***, este valor debe ser **001** (esto se obtiene de la ***Figura D***)

Finalmente, el campo ***R/M.*** Ya que el campo ***MOD*<> 11**, este campo debe especificar cuál registro base y cuál registro de índice están siendo usados para generar la dirección ***offset*** de **16** bits. En nuestro caso, usamos el campo **[*BX*+*SI*+DESPLAZAMIENTO]**

Esto corresponde a ***R/M* = 000**, ver ***Figura F***

El segundo byte es **1000 1000 = 88**

El tercer y cuarto byte corresponden al ***desplazamiento***

En este caso, el valor de ***COUNT* = 0345H**. Los últimos **2** bytes son **4503H**

Esto da el siguiente ***código objeto*** total para la instrucción:

MOV CX, COUNT [BX] [SI] 8BH

88H

45H

03H

Sumario del *Código Objeto*

Una pregunta que surge al programador ¿**Debo conformar los campos *D, W, REG, MOD* y *R/M*, en cada instrucción**? **NO**, la computadora lo hace (el lenguaje ensamblador lo genera) Esta sección se presentó para permitirle al programador un mejor entendimiento del trabajo interno del microprocesador ***8086/8088***

Interrupciones de los *Servicios Básicos de Entrada y Salida* (*BIOS, por sus siglas en inglés*)

FUNCIÓN INT 21

- **(AH)=1 ENTRADA DESDE EL TECLADO**

Esta función espera a que se digite un carácter en el teclado. Muestra el carácter en la pantalla (*eco*) y retorna el código **ASCII** en el registro ***AL***. (***AL***) = carácter leído desde el teclado

Ejemplo:

***MOV* AH, 1**

INT* 21h** ;AL*** = dato **ASCII** leído desde el teclado

- **(AH)=2 SALIDA EN EL EXHIBIDOR (*display*)**

Despliega un carácter en la pantalla. Algunos caracteres tienen un significado especial:

- **7** ***CAMPANA***: ***Suena*** durante un segundo
- **8** ***BACKSPACE***: ***Mueve*** el cursor hacia la izquierda un carácter
- **9 TABULADOR**: ***Mueve*** el tabulador a su próxima posición (cada 8 caracteres)
- **0Ah LF**: ***Mueve*** el cursor a la siguiente línea
- **0Dh CR**: ***Mueve*** el cursor al inicio de la línea corriente
- **(DL)**: Carácter a ***desplegar*** en la pantalla

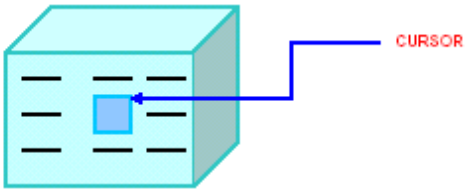
Ejemplo: **Desplegar un carácter**

MOV DL, 40 ; carácter a desplegar

MOV AH, 2

INT 21h ; aparece en la posición corriente del cursor

; el carácter contenido en ***DL***



Ejemplo: **Hacer que suene la campana 2 segundos**

MOV DL, 7 ; ***DL*** = campana

MOV AH, 02

INT 21h ; **1** segundo

INT 21h ; **1** segundo

- **(AH)=8 ENTRADA DESDE EL TECLADO SIN ECO**

Lee un carácter desde el teclado, pero no se despliega en la pantalla

(***AL***) = carácter leído desde el teclado

MOV **AH, 08**

INT **21h** ;***AL*** = carácter

- **(AH)=9 DESPLIEGA UNA CADENA DE CARACTERES**

Despliega en la pantalla la cadena apuntada por el par de registros ***DS:DX***.

Debemos marcar el fin de la cadena con el carácter **$**

DS:DX apuntan a la cadena que se va a desplegar

- **(AH)=0A h LEE UNA CADENA**

Lee una cadena de caracteres desde el teclado ¿**Dónde queda la información**?

- **(AH)=25h ACTIVA EL VECTOR DE INTERRUPCIÓN**

Activa un vector de interrupción, para que apunte a una nueva rutina

(***AL***) = número de interrupción

ES:BX dirección del manipulador de interrupciones

- **(AH)=35h CONSIGUE VECTOR DE INTERRUPCIÓN**

Consigue la dirección de la rutina de servicio para el número de interrupción dado en ***AL***

(***AL***) = número de interrupción

ES:BX dirección del manipulador de interrupción

- **(AH)=4Ch SALIDA AL DOS**

Retorna al **DOS**. Trabaja para ambos archivos ***.com** y ***.Exe**. Recuerde que ***INT***

20h trabaja solamente para archivos ***.com**

(***AL***) = código de retorno, normalmente activo a **0**, pero se puede activar a cualquier otro número y usar los comandos del **DOS**, ***IF*** y ***ERRORLEVEL***, para detectar errores