Multiplicación (MUL e IMUL)

La multiplicación se realiza en byte o palabra y puede ser:

- con signo (**IMUL**)
- sin signo (**MUL**).

El resultado después de la multiplicación es siempre un número de doble ancho. Si se multiplican **dos números de 8 bit se genera un producto de 16 bit**, y si multiplicamos **dos números de 16 bit se genera un producto de 32 bits**.

Algunas banderas (OF o CF) cambian cuando la instrucción multiplicación se ejecuta y produce resultados predecibles. Las otras banderas también cambian pero sus resultados son impredecibles y por tanto no son usados.

Multiplicación (MUL e IMUL)

Multiplicación de 8 bits:

Con la multiplicación de 8 bit, ya sea con signo o sin signo, el multiplicando está siempre en el registro **AL**. El multiplicador puede estar en cualquier registro de 8 bit o en cualquier localidad de memoria.

El resultado de la multiplicación se almacena en el registro **AX**.

Tabla 13. Instrucciones de Multiplicación de 8 bits.

Instrucciones	Comentarios
MUL CL	AX = AL * CL
IMUL DH	AX = AL * DH
IMUL BYTE PTR[BX]	AX = AL * DS:[BX]
MUL TEMP	AX = AL * DS:TEMP

Multiplicación (MUL e IMUL)

Multiplicación de 16 bits:

La multiplicación de palabra es similar a la multiplicación de byte. Las diferencias son que **AX** contiene el multiplicando en lugar de AL, y el resultado se almacena en **DX-AX** en lugar de AX. El registro DX siempre contiene los 16 bit más significativos del producto y AX los 16 bit menos significativos.

Tabla 14. Instrucciones de Multiplicación de 16 bits.

Instrucciones	Comentarios
MUL CX	DX-AX = AX * CX
IMUL DI	DX-AX = AX * DI
MUL WORD PTR[SI]	DX-AX=AX*DS:[SI]

División (DIV e IDIV)

La división se lleva a cabo con números de 8 y 16 bit que son números enteros con **signo** (**IDIV**) o **sin signo** (**DIV**).

El **dividendo es siempre de doble ancho**, el cual es dividido por el operando.

Esto quiere decir que una división de 8 bit, divide un número de 16 bit entre uno de 8 bit, la división de 16 bit divide un numero de 32 bit entre uno de 16 bit.

Ninguna de las banderas cambian en forma predecible con una división.

División (DIV e IDIV)

Una división puede resultar en dos tipos diferentes de errores. Uno de estos es intentar dividir entre cero y el otro es un sobreflujo por división.

Un ejemplo donde ocurriría un sobreflujo es el siguiente:

supongamos que AX= 3000 (decimal) y que lo dividimos entre dos.

Puesto que el cociente de una división de 8 bit se almacena en AL, y 1500 no cabe, entonces el resultado causa un sobreflujo por división.

División (DIV e IDIV)

División de 8 bit:

Una división de 8 bit emplea el registro **AX** para almacenar el dividendo que será dividido entre el contenido de un registro de 8 bit o una localidad de memoria.

Después de la división, el **cociente** se almacena en **AL** y el **residuo** en **AH**.

Para una división con signo el cociente es positivo o negativo, pero el residuo siempre es un número entero positivo.

Por ejemplo, si AX=0010H (+16) y BL=FDH (-3) y se ejecuta la instrucción IDIV BL, entonces AX queda AX=01FBH. Esto representa un cociente de -5 con un residuo de 1.

División (DIV e IDIV)

División de 8 bit:

Tabla 15. Instrucciones de División de 8 Bits.

Instrucciones	Comentarios	
DIV CL	AL= AX / CL,	AH=Residuo
IDIV BL	AL= AX / BL,	AH=Residuo
DIV BYTE PTR[BP]	AX=AX / SS:[BP],	AH=Residuo

División (DIV e IDIV)

División de 16 bit:

La división de 16 bit es igual que la división de 8 bit excepto que en lugar de dividir AX se divide **DX-AX**, es decir, se tiene un dividendo de 32 bit.

Después de la división, el **cociente** se guarda en **AX** y el **residuo** en **DX**.

Tabla 16. Instrucciones de División de 16 bits.

Instrucciones	Comentarios
DIV CX	AX=(DX-AX)/CX,DX=Residuo
IDIV SI	AX=(DX-AX)/SI,DX=Residuo
DIV NUMB	AX=(DX-AX)/DS:NUMB, DX=Residuo

XLAT

Se utiliza para realizar una técnica directa de conversión de un código a otro, por medio de una *lookup table*.

Una instrucción XLAT primero **suma** el contenido de **AL** con el contenido del registro **BX** para formar una dirección del segmento de datos, luego el dato almacenado en esta dirección es cargado en el registro **AL**.

El registro BX almacenaría la dirección de inicio de la tabla y AL almacenaría la posición del dato que se quiere cargar en este registro.

Esta instrucción no posee operando, ya que siempre opera sobre AL.

LEA

Se utiliza para cargar un registro con la dirección de un dato especificado por un operando (variable).

Ejemplo:

LEA AX,DATA

AX se carga con la dirección de DATA (16 bits), cabe notar que se almacena la dirección y no el contenido de la variable.

En una comparación de la instrucción LEA con MOV se puede observar lo siguiente:

LEA BX,[DI] carga la dirección especificada por [DI] en el registro BX. Es decir, estaría copiando el contenido de DI en BX.

MOV BX,[DI] carga el contenido de la localidad direccionada por DI en el registro BX.

LDS y LES

Estas instrucciones cambian el rango del segmento de Datos o del segmento Extra, y permiten cargar un nueva dirección efectiva.

Las instrucciones **LDS** y **LES** cargan **un registro de 16 bits** con un dato que representaría una dirección (un desplazamiento), y cargan el **registro de segmento DS** o **ES** con una nueva dirección de segmento.

Estas instrucciones utilizan cualquier modo de direccionamiento a memoria válido para seleccionar la localidad del nuevo desplazamiento y nuevo valor de segmento.

LDS y LES

En el diagrama se muestra la operación de la instrucción **LDS BX,[DI]**, la cual transfiere el contenido (2 bytes) de la localidad de memoria apuntada por DI, al registro BX; y transfiere los siguientes 2 bytes al registro de segmento DS.

Es decir, transfiere la dirección de 32 bits que esta almacenada en memoria y es apuntada por DI, a los registro BX y DS

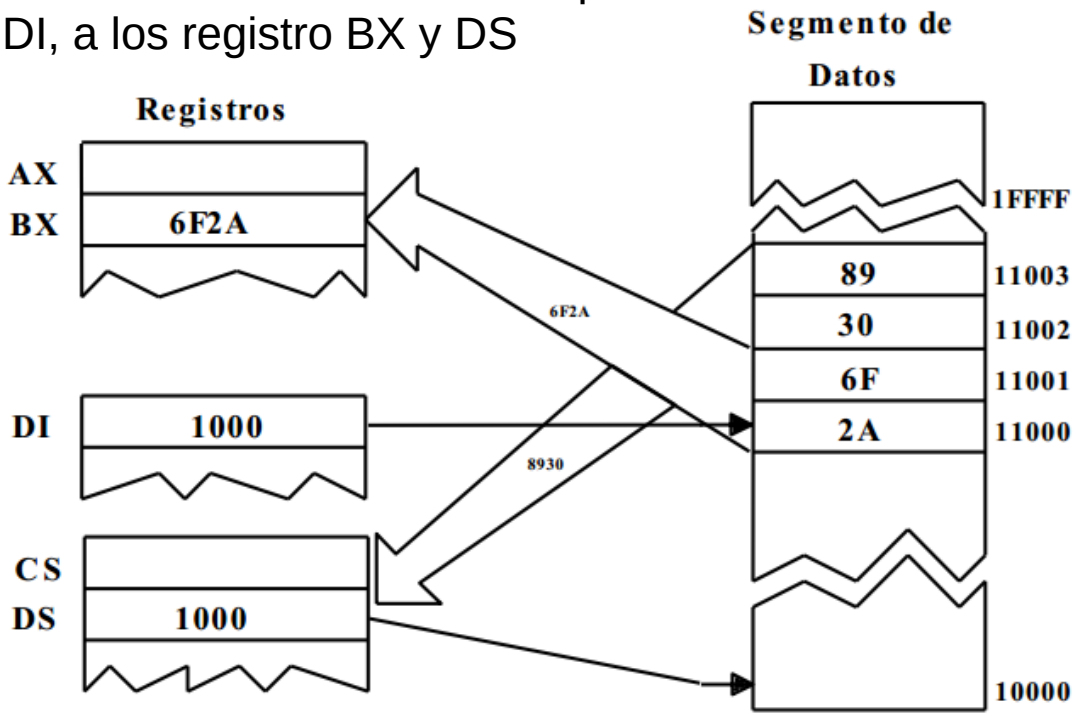

Figura 3. LDS BX,[DI] carga el registro BX con el contenido de la localidad 11000H y 11001H y el registro DS con el contenido de la localidad 11002H y 11003H. Esta instrucción cambia el rango de segmento 1000H-1FFFFH a 89300H-992FFH.

Ejemplos de LEA, LDS y LES

Tabla 4. Instrucción XCHG

Código de operación	Función
LEA AX,DATA	AX se carga con la dirección de DATA (16 bits)
LDS DI,LIST	DI y DS se cargan con la dirección de LIST (32 bits- DS:DI)
LES BX,CAT	BX y ES se cargan con la dirección de CAT (32 bits -ES:BX)