Transferencia de datos

Son las instrucciones encargadas de mover datos de un sitio a otro de la computadora como pueden ser la memoria, el espacio de E/S y los registros del CPU.

Instrucciones:

- MOV
- XCHG
- IN
- OUT
- PUSH
- POP
- PUSHF
- POPF
- LAHF
- SAHF
- XLAT
- LEA
- LDS

XCHG

Intercambia el contenido de cualquier registro con el contenido de cualquier otro registro o localidad de memoria, no incluyendo los registros de segmento o intercambios de memoria a memoria.

Ejemplo:

AX = 1F60 BX = 472D CX = 310B DX = 8472

XCHG AX,BX

AX = 472D BX = 1F60 CX = 310B DX = 8472

XCHG CL,DH

AX = 472D BX = 1F60 CX = 3184 DX = 0B72

IN y OUT

IN:

Lee un dato de un dispositivo de E/S y lo almacena en el registro AL o AX.

OUT:

Transfiere un dato del registro AL o AX a un puerto de E/S.

IN y OUT

Existen dos formas para el direccionamiento de puertos con las instrucciones IN y OUT, las cuales son: **puerto fijo** y **puerto variable**.

- **Puerto fijo:** Permite la transferencia de datos entre AL o AX y un puerto de E/S con dirección de 8 bits.

Ejemplos: `IN AX,85` `IN AL,3F` `OUT 42,AL` `OUT 2A,AX`

- **Puerto variable:** Permite la transferencia de datos entre AL o AX y un puerto de E/S con dirección de 16 bits, y el numero de puerto se almacena en DX.

Ejemplos: `IN AX,DX` `IN AL,DX` `OUT DX,AL` `OUT DX,AX`

IN y OUT

Tabla 2. Instrucciones IN y OUT.

Código de operación	Función
IN AL, pp	Un dato de 8 bits se transfiere del puerto pp a AL
IN AX,pp	Un dato de 16 bits se transfiere del puerto pp a AX
IN AL,DX	Un dato de 8 bits se transfiere del puerto DX a AL
IN AX,DX	Un dato de 16 bits se transfiere del puerto DX a AX
OUT pp,AL	Un dato de 8 bits se transfiere de AL al puerto DX
OUT pp,AX	Un dato de 16 bits se transfiere de AX al puerto pp
OUT DX,AL	Un dato de 16 bits se transfiere de AL al puerto DX
OUT DX,AX	Un dato de 16 bits se transfiere de AX al puerto DX

Nota: pp= un puerto de E/S de con dirección de 8 bits y DX = contiene la dirección de un puerto de E/S con dirección de 16 bits.

PUSH y POP

Se utilizan para empujar o sacar datos del **Segmento de Pila**.

El registro **SP** (Apuntador de Pila) almacena la dirección del segmento de Pila de donde se van a ingresar o sacar datos.

En el 8088 se dice que la Pila crece a direcciones mas pequeñas, ya que cuando se **ingresan** datos se **decrementa** el apuntador de pila, y cuando se **extraen** datos se **incrementa** el apuntador. Es decir, conforme se van ingresando datos SP apunta a direcciones cada vez mas pequeñas.

PUSH y POP

PUSH:

Ingresa 2 bytes de información a la pila.

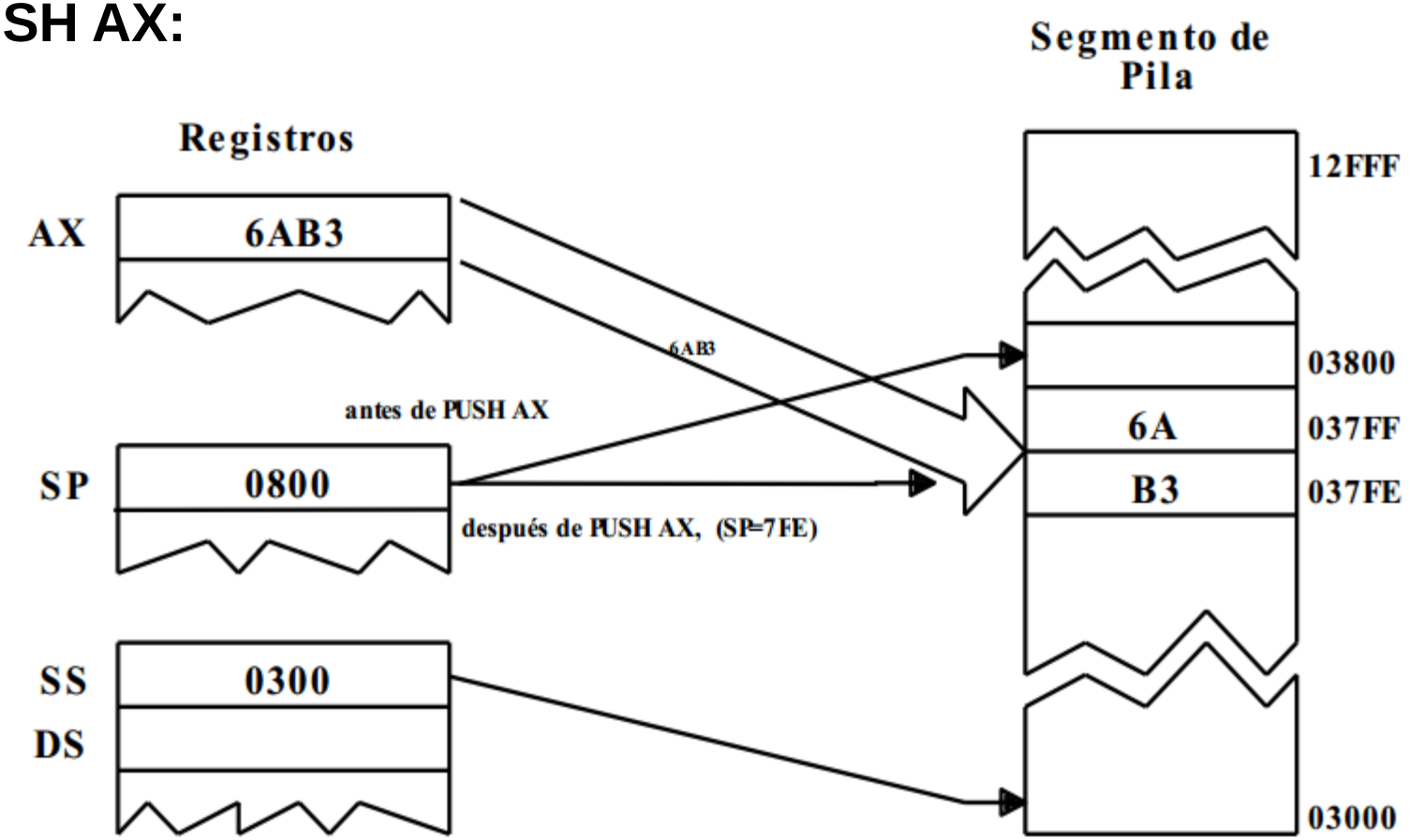
Cuando un dato es metido a la pila, el byte **mas significativo** es almacenado en la localidad direccionada por **SP-1**, y el byte **menos significativo** es almacenado en la localidad **SP-2**.

Después de que el dato se ha empujado a la pila, el registro **SP** es **decrementado en 2**.

PUSH y POP

Ejemplo:

PUSH AX:



PUSH y POP

POP:

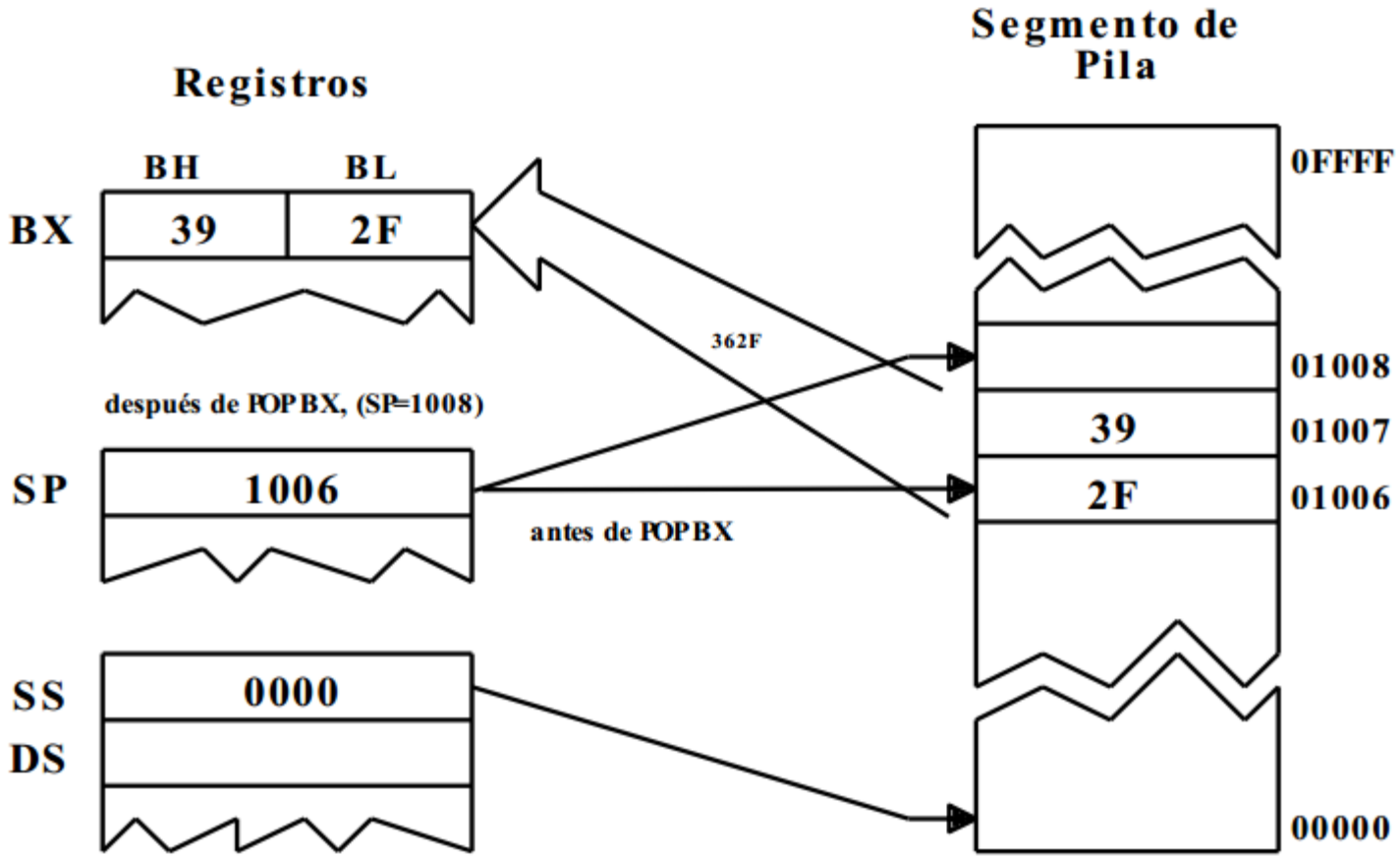
Realiza lo inverso a la instrucción PUSH. POP **remueve <u>2 bytes</u>** de la pila.

Suponga que la instrucción POP BX es ejecutada. El byte apuntado por **SP** es removido de la pila y almacenado en el registro BL. Después se remueve el byte apuntado por **SP+1** y se coloca en el registro BH.

Después de que ambos datos fueron extraídos de la pila, **SP** es **incrementado en 2**.

PUSH y POP

POP BX:



PUSH y POP

Las instrucciones PUSH y POP pueden operar sobre registros, memoria y registros de segmento.

Importante:

Sí se permite empujar a la pila el registro de segmento de Código (CS), pero no se permite extraer un valor de la pila y almacenarlo en CS, ya que no se permite la modificación de este registro.

PUSH CS ✓ (permitido)

POP CS ó **MOV CS,valor** , etc ✖

PUSHF y POPF

PUSHF:

Copia el contenido del registro de banderas a la pila.

POPF:

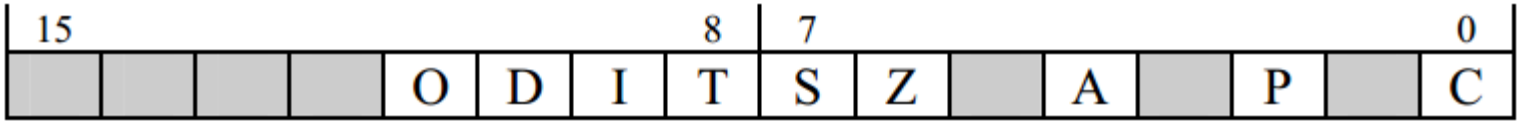
Realiza la operación inversa de PUSHF, remueve de la pila un dato de 16 bits que es cargado como el contenido del registro de banderas.

LAHF

Copia el byte menos significativo del registro de banderas en el registro AH.

Después de la ejecución de esta instrucción, los bits 7,6,4,2 y 0 de AH son iguales a las banderas S, Z, A, P y C respectivamente.

Registro de banderas:



SAHF

Copia el contenido de AH en el byte menos significativo del registro de banderas.

Después de la ejecución de esta instrucción, las banderas S, Z, A, P y C son iguales a los bits 7, 6, 4, 2 y 0 de AH, respectivamente.

Suma (ADD)

Siempre que una instrucción aritmética o lógica se ejecuta, cambia el contenido del registro de banderas. Las banderas denotan el resultado de la operación aritmética.

Cualquier instrucción ADD modifica el contenido de las banderas de signo, cero, acarreo, acarreo auxiliar, paridad y sobreflujo.

Importante:

No se permite la suma con registros de segmento

Suma (ADD)

Tabla 6. Instrucciones de Suma.

Instrucciones	Comentario
ADD AL,BL	AL = AL + BL
ADD CX,DI	CX = CX + DI
ADD CL,44H	CL = CL + 44H
ADD BX,35AFH	BX = BX+35AFH
ADD [BX],AL	DS:[BX] = DS:[BX]+AL
ADD CL,[BP]	CL=CL+SS:[BP]
ADD BX,[SI+2]	BX=BX+DS:[SI+2]
ADD CL,TEMP	CL=CL+[offset TEMP]
ADD BX,TEMP[DI]	BX=BX+[offset TEMP + DI]
ADD [BX+DI],DL	DS:[BX+DI]=DS:[BX+DI]+DL

Suma con acarreo (ADC)

Realiza la suma de dos operandos más el bit de acarreo.

Tabla 8. Instrucciones Suma con acarreo.

Instrucciones	Comentarios
ADC AL,AH	AL = AL + AH + CF
ADC CX,BX	CX = CX + BX + CF
ADC [BX],DH	DS:[BX]=DS:[BX] + DH + CF
ADC BX,[BP+2]	BX=BX+SS:[BP+2]+CF

Suma con acarreo (ADC)

Ejemplo:

Suponga que un número de 32 bit contenido en los registros BX y AX se suma a un número de 32 bit contenido en los registros DX y CX. Para que el resultado de la suma sea correcto, se tiene que considerar el acarreo que pudiera existir al sumar AX y CX, por lo tanto se tiene que hacer uso de la instrucción ADC.

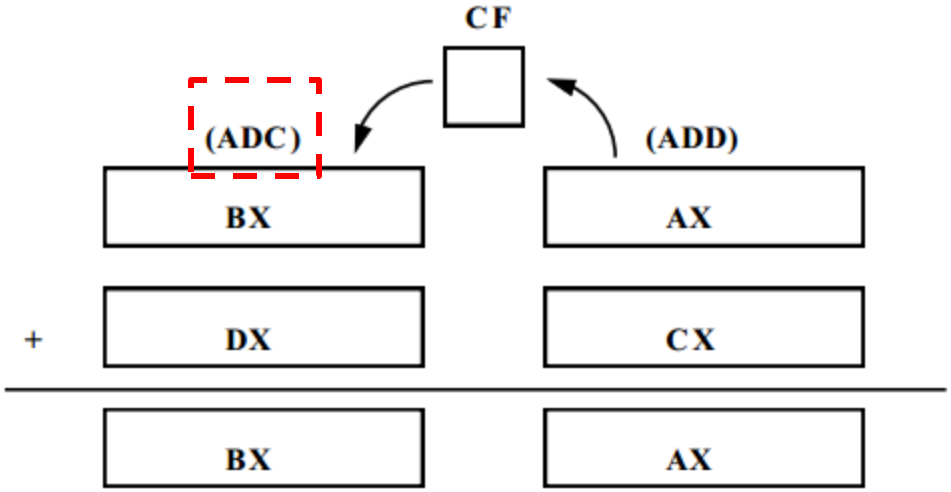


Figura 4. Suma con acarreo mostrando como la bandera de acarreo (CF) une dos sumas de 16 bits para formar suma de 32 bits.

Incremento (INC)

Incrementa en 1 el contenido de un registro o localidad de memoria. Al igual que la suma, no se permite el incremento de los registros de segmento.

Esta instrucción modifica el contenido del registro de banderas, a excepción de que no modifica el valor del bit de acarreo.

Ejemplo:

INC AX

Incremento (INC)

Tabla 7. Instrucciones de Incremento.

Instrucciones	Comentarios
INC BL	BL = BL + 1
INC SP	SP = SP + 1
INC BYTE PTR[BX]	DS:[BX]=DS:[BX] + 1
INC WORD PTR[SI]	DS:[SI]=DS:[SI] + 1
INC DATA1	DS:DATA1=DS:DATA1+1

Cuando se incrementa un dato que está almacenado en memoria, se tiene que especificar si el dato a incrementar es de 8 o de 16 bits.

BYTE PTR: Indica que el incremento va a ser de un dato de 8 bits.

WORD PTR: Indica que el incremento va a ser de un dato de 16 bits.

Resta (SUB)

Ejemplos:

Tabla 9. Instrucciones de Resta.

Instrucciones	Comentarios
SUB CL,BL	CL = CL - BL
SUB AX,SP	AX = AX - SP
SUB DH,6FH	DH = DH - 6FH
SUB AX,0CCCCH	AX=AX-DS:0CCCCH
SUB [DI],CH	DS:[DI]=DS:[DI]-CH
SUB CH,[BP]	CH=CH-SS:[BP]
SUB AH,TEMP	AH=AH-DS:TEMP
SUB DI,TEMP[BX]	DI=DI-TEMP[BX]

Resta con préstamo (SBB)

Una instrucción resta con préstamo (SBB) funciona como una resta regular, excepto que la bandera de acarreo (CF), la cual actúa como préstamo también se substrae de la diferencia.

Tabla 11. Instrucciones de Resta con Acarreo.

Instrucciones	Comentarios
SBB AH,AL	AH = AH - AL - CF
SBB AX,BX	AX = AX - BX - CF
SBB CL,3	CL = CL -3 - CF
SBB BYTE PTR[DI],3	DS:[DI]=DS:[I]-3-CF
SBB [DI],AL	DS:[DI]=DS:[DI]-AL-CF
SBB DI,[BP+2]	DI=DI-SS:[BP+2]-CF

Resta con préstamo (SBB)

Ejemplo:

Un número de 32 bit almacenado en BX y AX se substrae de un número de 32 bit almacenado en SI y DI, la bandera de acarreo propaga el préstamo entre las dos substracciones de 16 bit requeridas para realizar la operación.

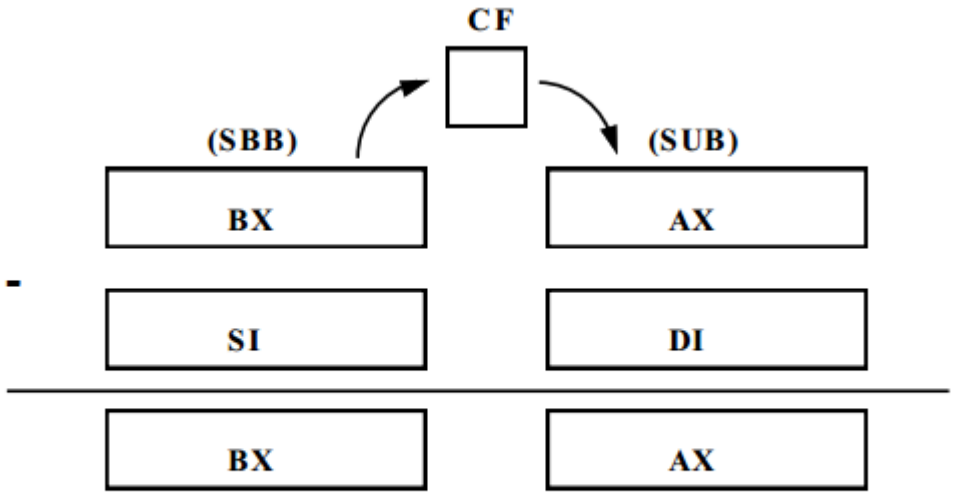


Figura 5. Resta con préstamo mostrando como la bandera de acarreo (CF) propaga el préstamo.

Decremento (DEC)

Decrementa en 1 el contenido de un registro o localidad de memoria.

Esta instrucción modifica el contenido del registro de banderas, a excepción de que no modifica el valor del bit de acarreo.

Ejemplo:

DEC AX

Decremento (DEC)

Tabla 10. Instrucciones de Decremento.

Instrucciones	Comentarios
DEC BH	BH = BH - 1
DEC SP	SP = SP - 1
DEC BYTE PTR[DI]	DS:[DI]=DS:[DI]-1
DEC WORD PTR[BP]	SS:[BP]=SS[BP]-1
DEC NUMB	DS:NUMB=DS:NUMB-1

Al igual que con la instrucción INC, cuando se va a decrementar un valor contenido en memoria se tiene que especificar por medio de las directivas BYTE PTR o WORD PTR si se esta trabajando con un dato de 8 o de 16 bits.

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**.

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 cambia 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=0010 (+16) y BL=FD (-3) y se ejecuta la instrucción IDIV BL, entonces AX queda AX=01FB. 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

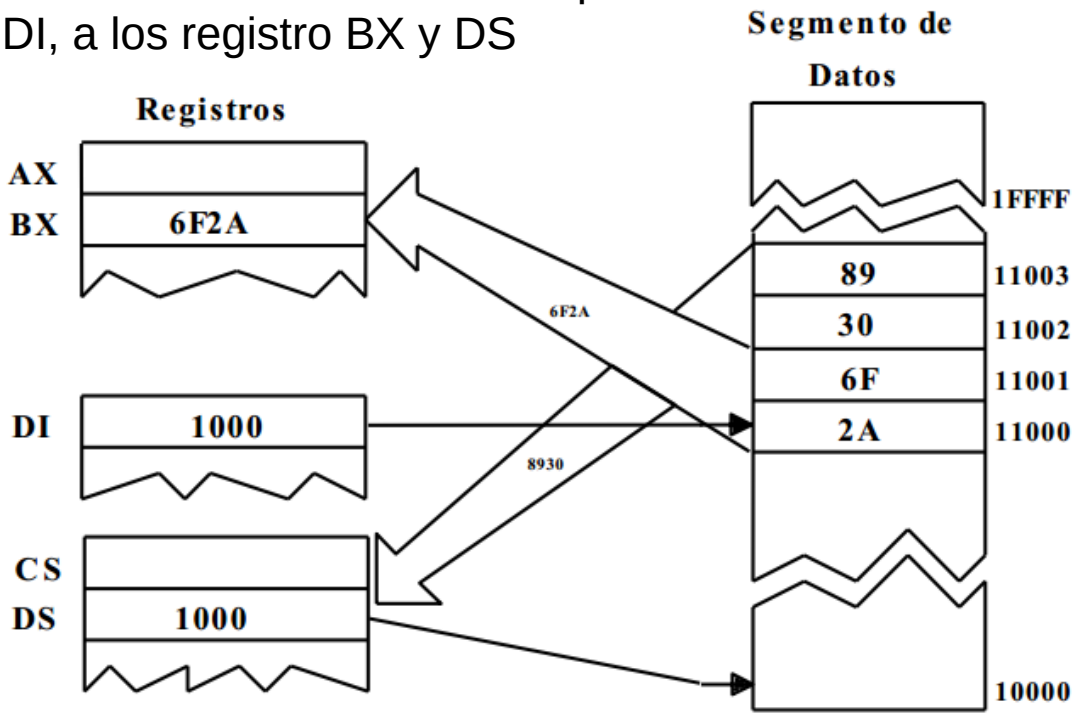


Figura 3. LDS BX,[DI] carga el registro BX con el contenido de la localidad 11000H y 11001H y el registro DS con el contenido de la localidad 11002H y 11003H. Esta instrucción cambia el rango de segmento 1000H-1FFFFH a 89300H-992FFH.

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)