UNIVERSIDAD AUTÓNOMA DE BAJA CALIFORNIA

Facultad de Ciencias Químicas e Ingeniería

Materia: Organización de Computadoras y Lenguaje Ensamblador
Docente: Lara Camacho Evangelina

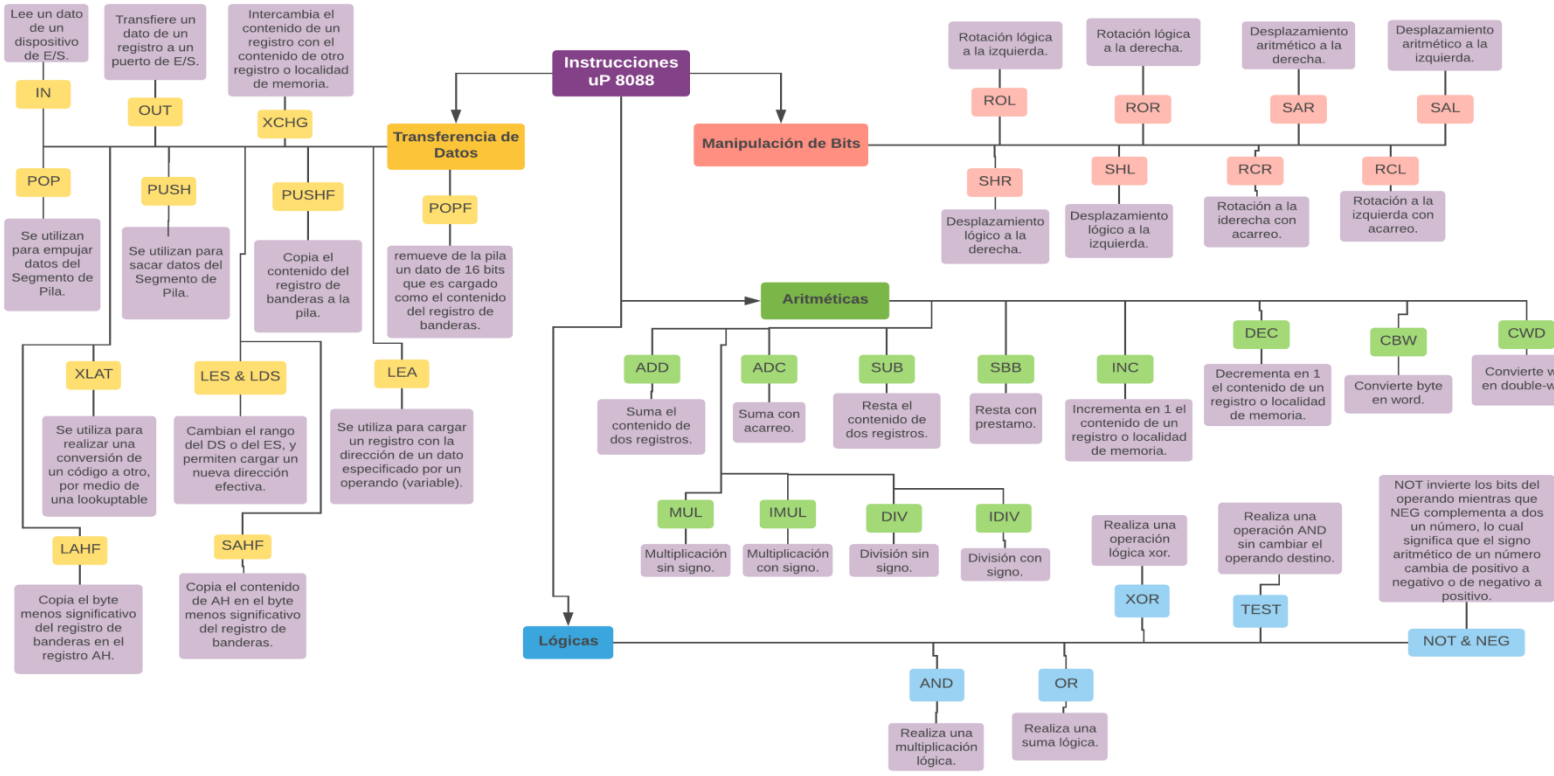
Practica 6 Instrucciones del procesador

Alumno:

Morales Rosales Iván A. 1231098

Teoría

Mapa conceptual.



Desarrollo

a) *Instrucciones lógicas y de manipulación de bits: NOT, AND, OR, XOR, TEST, SHL, SHR, SAR, ROL, ROR, RCL, RCR.*

NOT

Se carga el registro ax con el valor 1234 para después aplicar la operación lógica“NOT”, que es invertir los bits del registro.

```
073F:0100 mov ax,1234
073F:0103 not ax
AX=1234  BX=0000  CX=0000  DX=0000  SP=00FD  BP=0000  SI=0000  DI=0000
DS=073F  ES=073F  SS=073F  CS=073F  IP=0103   NV UP EI PL NZ NA PO NC
073F:0103 F7D0          NOT     AX
-t

AX=EDCB  BX=0000  CX=0000  DX=0000  SP=00FD  BP=0000  SI=0000  DI=0000
DS=073F  ES=073F  SS=073F  CS=073F  IP=0105   NV UP EI PL NZ NA PO NC
073F:0105 8099B07F98    SBB     BYTE PTR [BX+DI+7FB0],98          DS:7
```

AND

Se desactivan los bits 0, 3, 4, 7 y 15 del registro ax (AND AX,6D24).

1110 1101 1100 1011 → EDCB
0111 1111 0110 0110 → 7F66

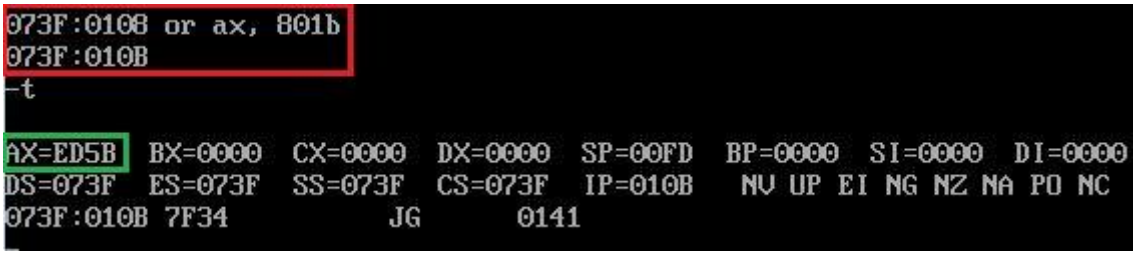
0110 1101 0100 0010→6D42

OR

Los bits 0, 1, 3, 4, y 15 del contenido del registro ax se activan mediante la operación OR AX, 801B.

0110 1101 0100 0010 → 6D42
1000 0000 0001 1011 → 801B

1110 1101 0101 1011 → ED5B



XOR

El contenido del registro ax, se le aplica un enmascaramiento mediante la instrucción XOR AX, D915.

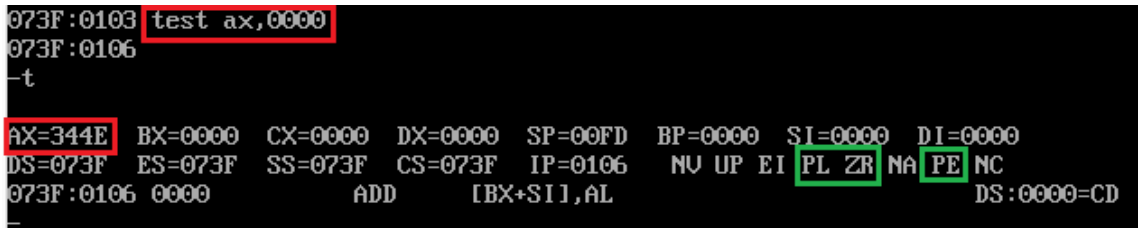
1110 1101 0101 1011 → ED5B
1101 1001 0001 0101 → D915

0011 0100 0100 1110 → 344E

TEST

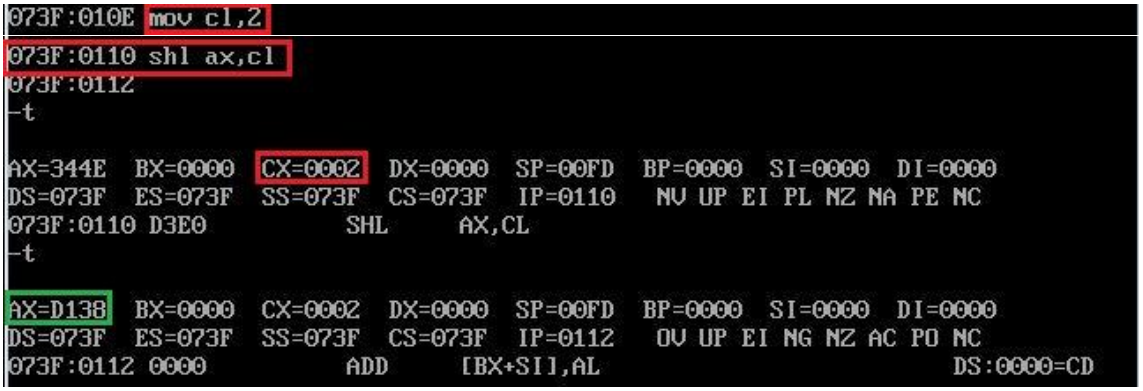
La instrucción **TEST** realiza una operación **AND**, pero sin afectar el operando, esta solo afecta las banderas de signo, zero y paridad.

Después de ejecutar la instrucción la única bandera afectada fue la de zero.



SHL

Para demostrar esta instrucción se ingresó el valor de 2 en el registro cl, ya que por medio de este se indica el total de corrimientos a la izquierda a realizar, tomando en cuenta que en este caso la bandera de carry quedo en cero.



SHR

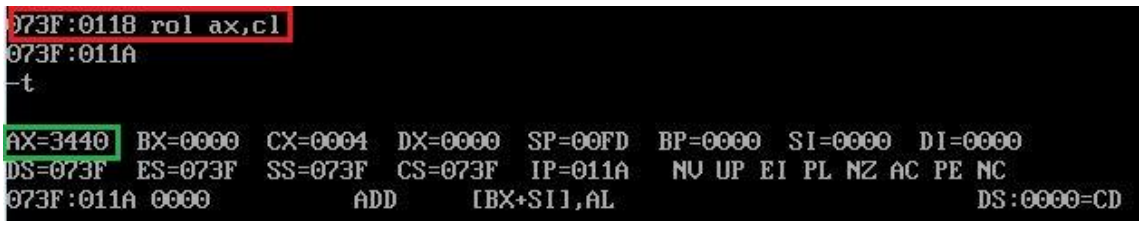
De igual manera que el caso anterior (SHL AX, CL), aplico dos corrimientos a la derecha y no quedo activado el carry.

SAR

Esta instrucción realiza corrimientos a la derecha, a diferencia de la instrucción **shr**, esta conserva el signo más significativo del registro. En este caso el carry se activo.

ROL

Con esta instrucción se realizan rotaciones a la izquierda. El bit más significativo es ingresado al carry al mismo tiempo que se copia hacia la parte menos significativa.

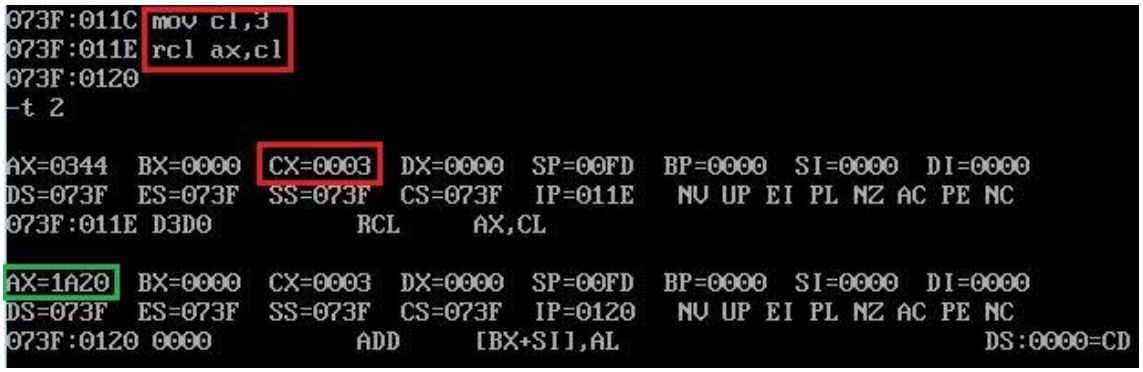


ROR

Realiza lo contrario de la instrucción **rol**, rotaciones hacia la derecha. El bit menos significativo es ingresado al carry al mismo tiempo que es copiado a la parte más significativa del valor del registro.

RCL

Con esta instrucción se realizan corrimientos hacia la izquierda tomando en cuenta el valor del carry. El valor actual del carry se ingresa en la parte menos significativa mientras que el MSB es ingresado en el carry.



RCR

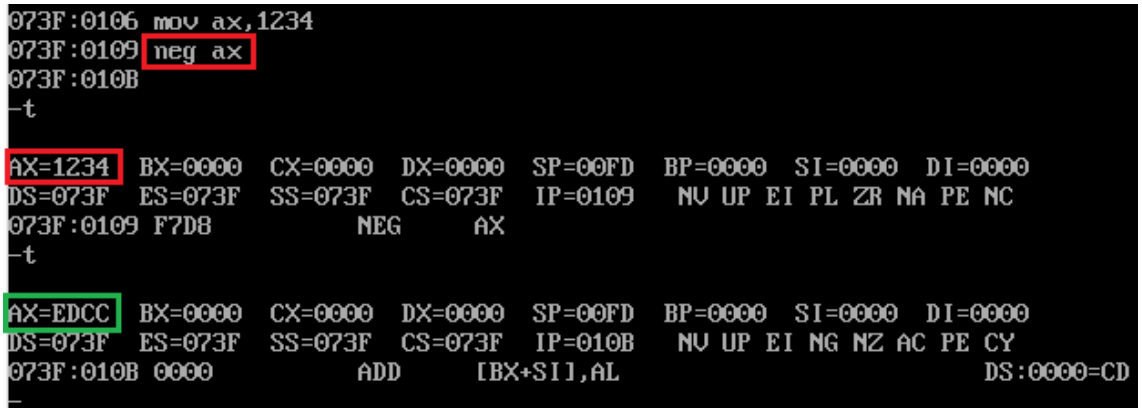
Con esta instrucción se realizan corrimientos hacia la derecha tomando en cuenta el valor del carry. El valor actual del carry se ingresa en la parte mas significativa mientras que el LSB es ingresado en el carry.

b) *Instrucciones aritméticas: NEG, MUL, IMUL, DIV, IDIV, CBW y CWD.*

NEG

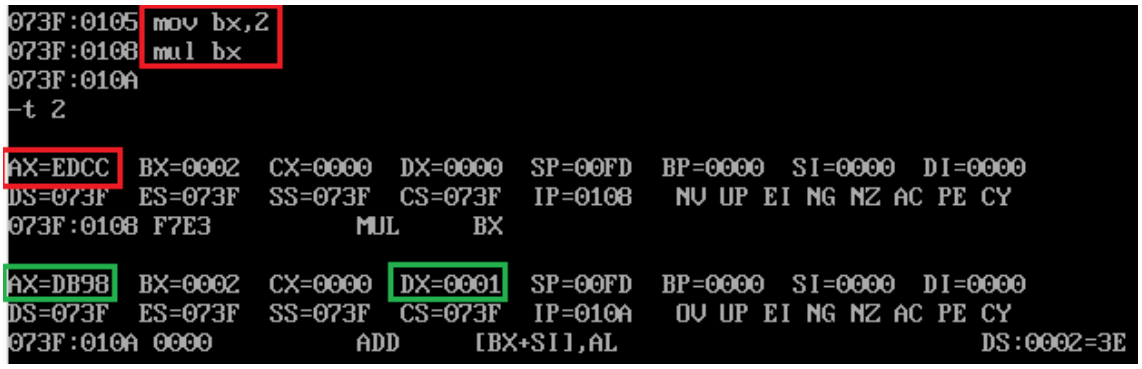
Invierte el signo del valor aplicando complemento a dos. Esta instrucción afecta las banderas carry, zero, signo, overflow, aux carry y paridad.

0001 0010 0011 0100
1110 1101 1100 1011 → not ax
\+ 1
---
110 1101 1100 1100 → EDCC



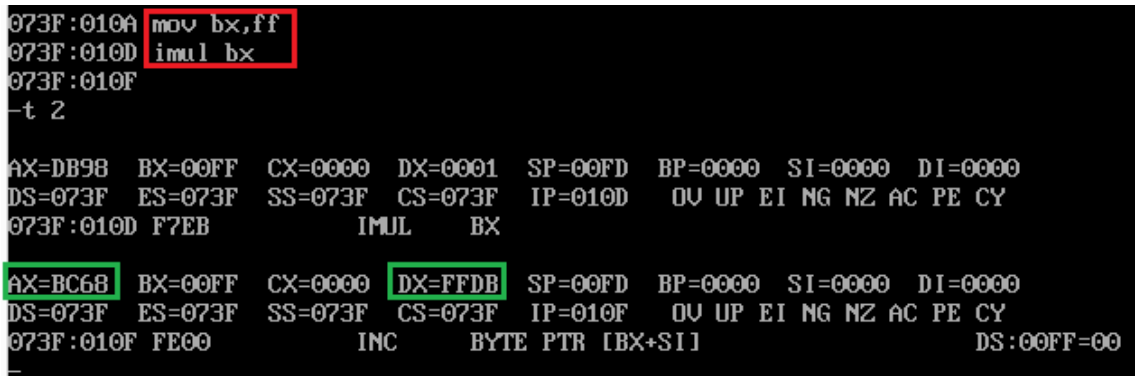
MUL

Se ingresa un valor al registro bx, el cual sera multiplicado por el registro ax, quien simpre contiene al multiplicando. EL resultado de la operación fue mayor de 2 bytes, por esta razón, se usa el registro dx.



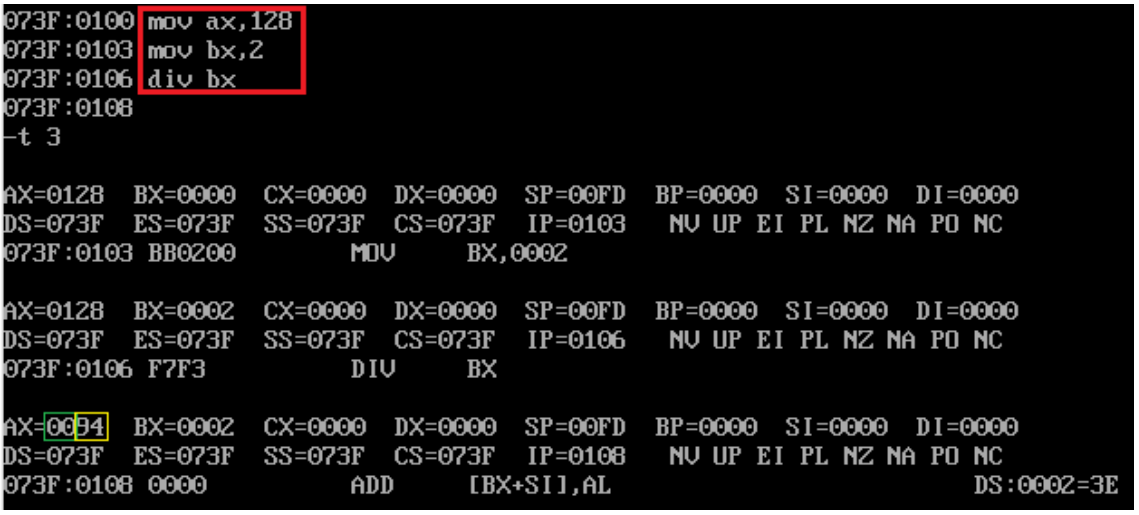
IMUL

Con este nemónico se multiplican números con singo. El valor del registro bx se multiplica con del registro ax. El producto es almacenado en registro dx (parte MSB) y registro bx(LSB).



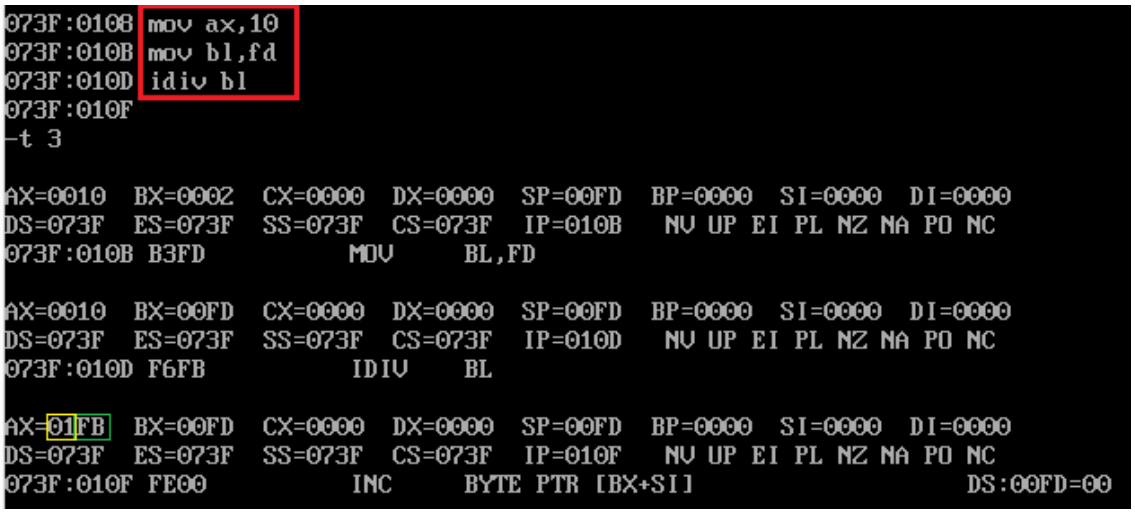
DIV

Se realizo la division de numeros positivos. El cociente es almacenado en AL y el resido en AH.



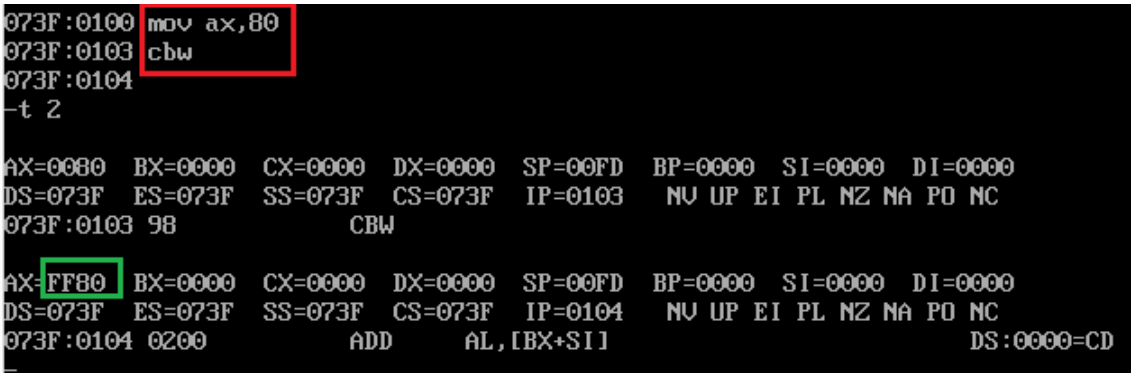
IDIV

Si la división se realizara con números con signo se utiliza IDIV.



CBW

Cuando se quiere convertir un byte a palabra se utiliza esta instrucción, a la vez, esta respeta el signo del byte.



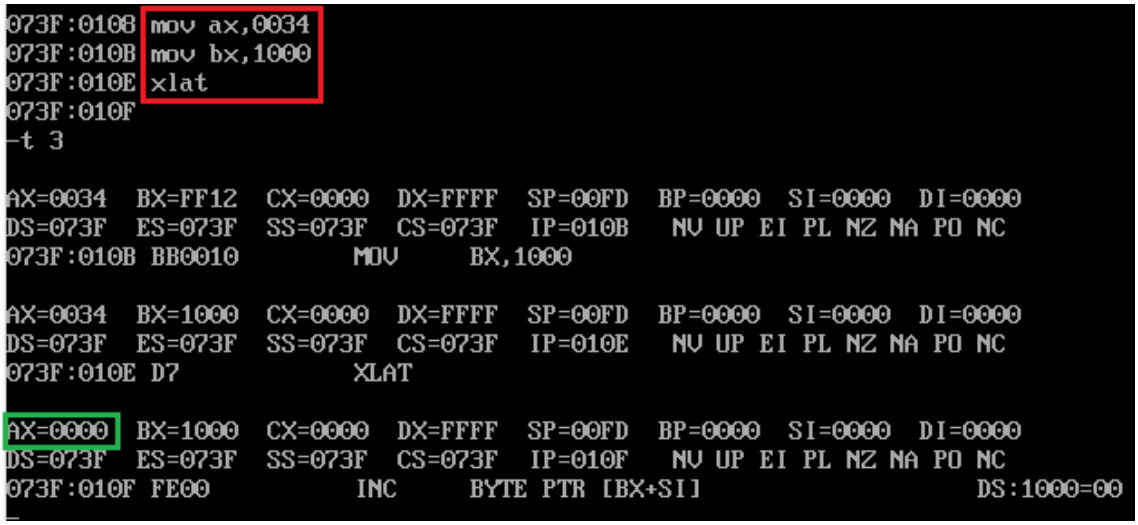
CWD

Para convertir una palabra a doble palabra, el cambio se puede realizar con esta instrucción, de igual manera respeta el signo. LA parte MSB se almacena en el registro dx mientras que la parte LSB en el registro origen.

c) *Instrucciones de movimiento de datos: XLAT, LEA, LDS y LES.*

XLAT

Esta instrucción suma el contenido de al con el contenido dl registro bx para formar una dirección del segmento de datos y almacenarla en el registro ax. La dirección resultante sería 34+1000 =1034, el valor contenido en esta dirección del segmento de datos se copia al registro ax.

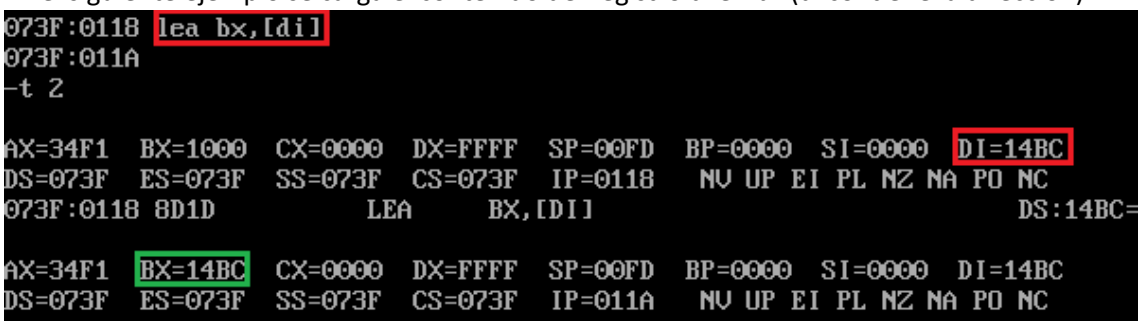


Usando el comando dump, se accede a la dirección para comprobar que el valor de ax corresponde al de la dirección.

LEA

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

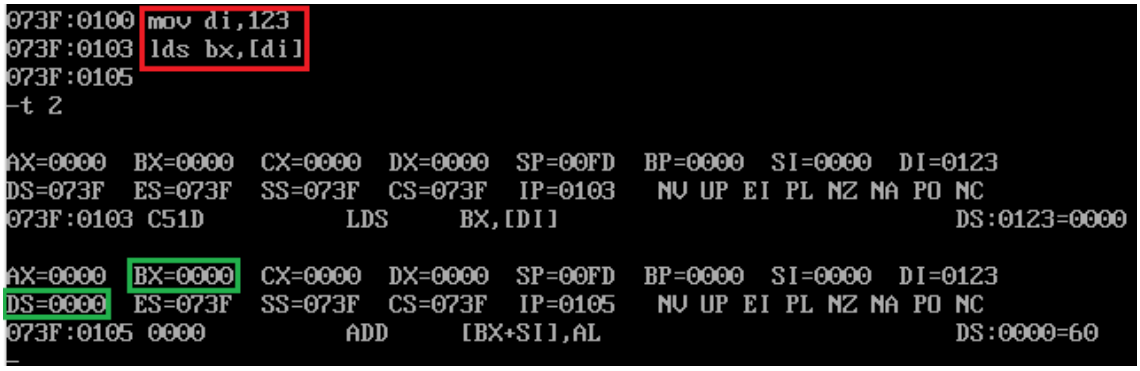
En el siguiente ejemplo se carga el contenido del registro di en bx (di contiene la dirección).



LDS y **LES**

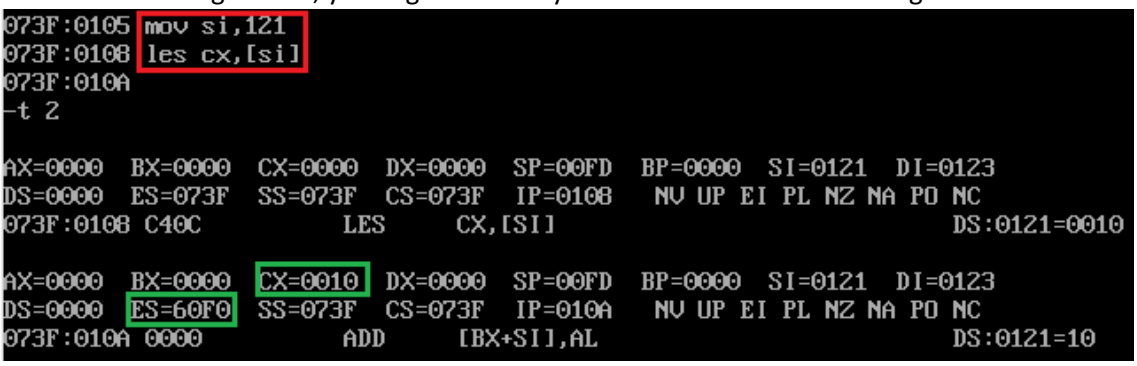
Estas instrucciones permiten cargar un registro de 2 bytes que representara una dirección y cargar DS o ES con una nueva dirección.

LDS: el valor de 2 bytes que se encuentra en memoria direccionada por el registro di, se almacena en el registro bx, y los siguientes 2 bytes son almacenados en el registro ds.



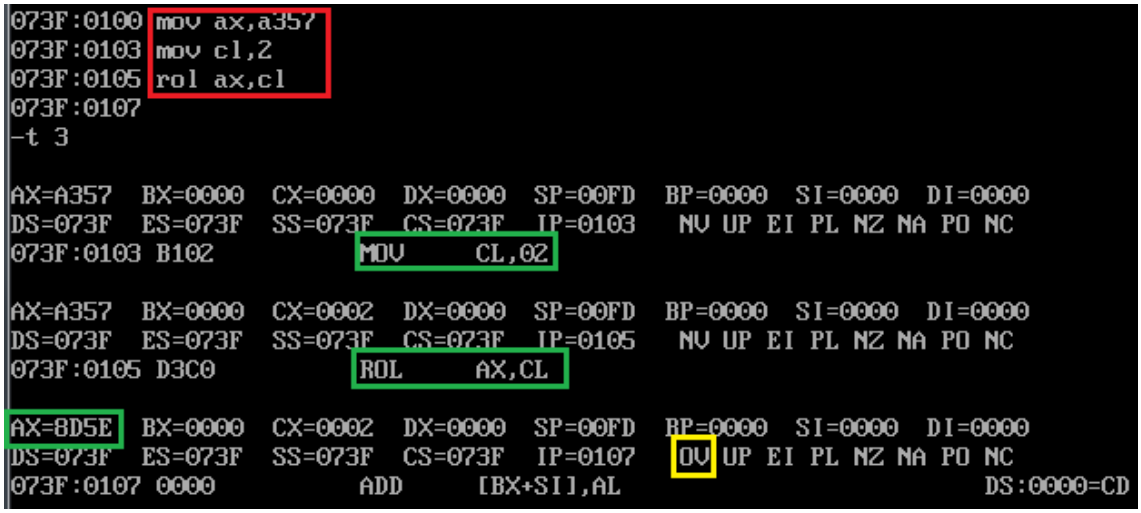
Para comprobar, se ejecuto el comando dump con la dirección obtenida (73F0H x 10H +123 = 7513).

LES: el valor de 2 bytes que se encuentra en memoria direccionada por el registro di, se almacena en el registro bx, y los siguientes 2 bytes son almacenados en el registro ds.

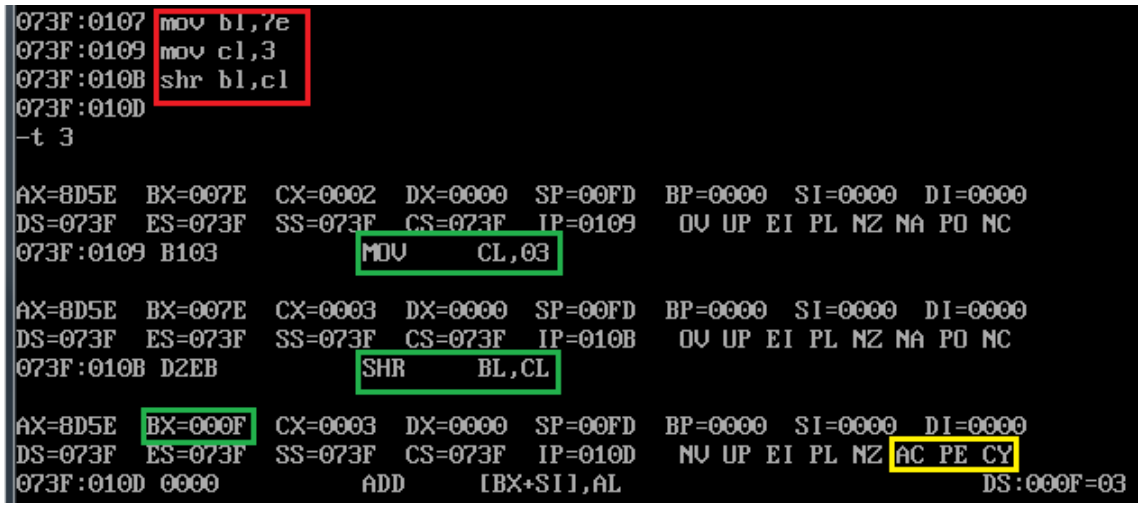


2. *Escriba y ejecute en Debug las instrucciones necesarias para:*

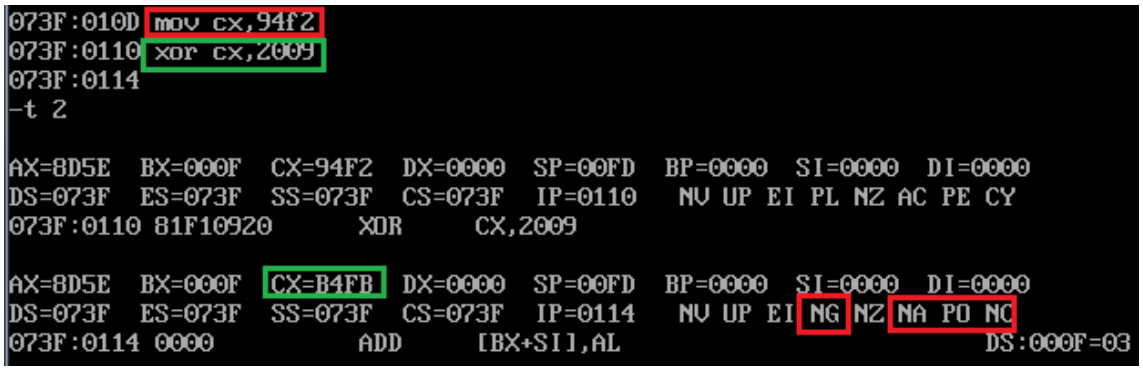
a) Colocar en el registro AX el valor 0xA357 y por medio de rotaciones obtener 0x8D5E.



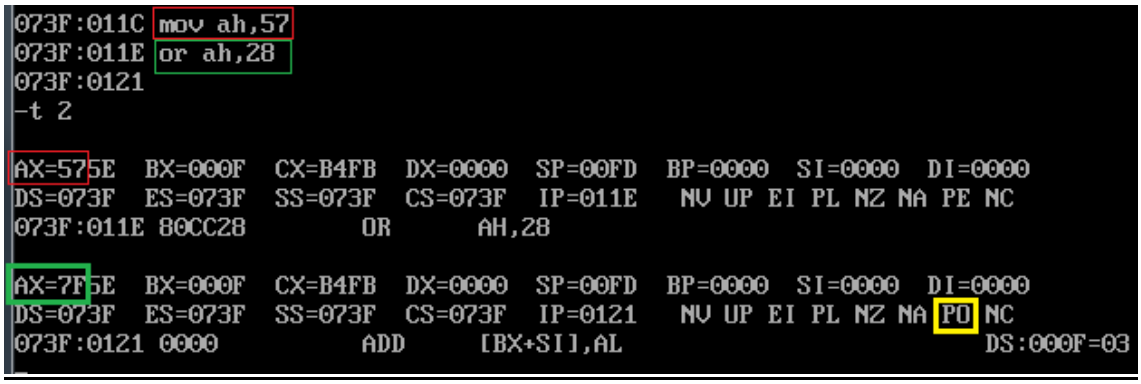
b) Colocar en el registro BL el valor 0x7E y por medio de corrimientos obtener 0xF.



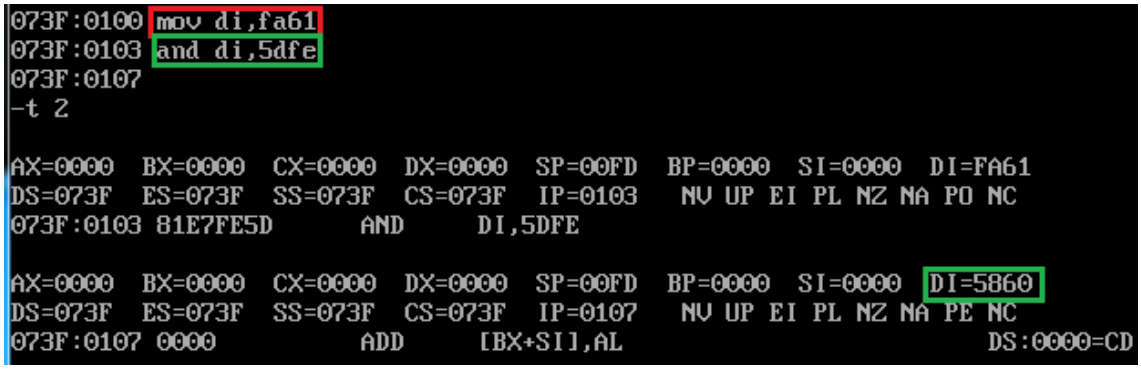
c) Colocar en el registro CX el valor 0x94F2 y por medio de enmascaramiento invertir los bits 0,3 y 13, sin modificar los demás.



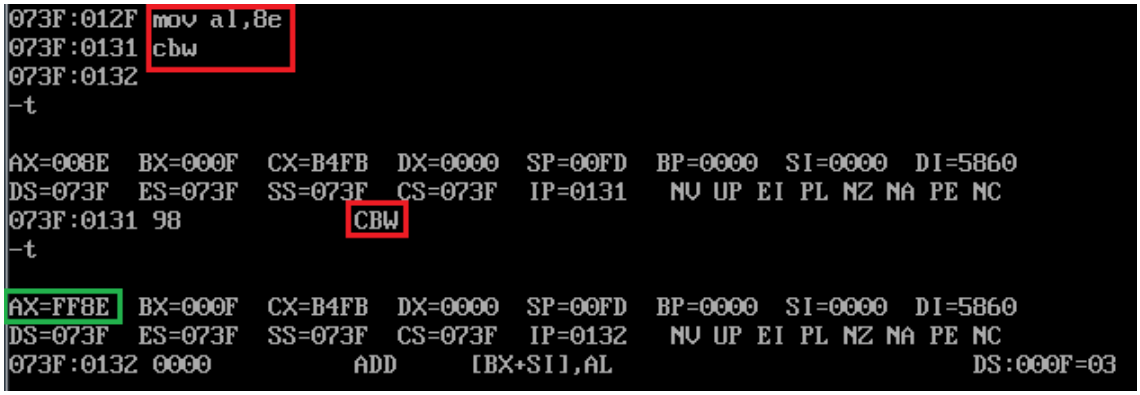
d) Colocar en el registro AH el valor 0x57 y por medio de enmascaramiento activar los bits 3 y 5, sin modificar los demás.



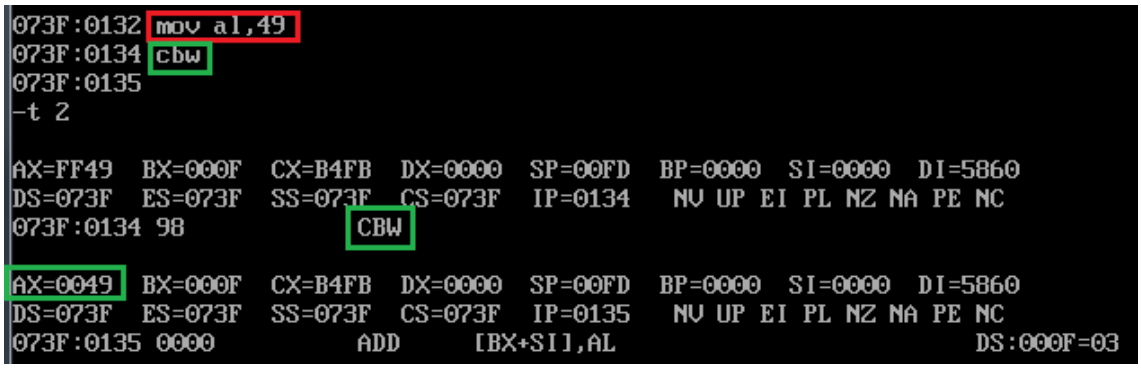
e) Colocar en el registro DI el valor 0xFA61 y por medio de enmascaramiento desactivar los bits 0, 9, 13 y 15, sin modificar los demás.



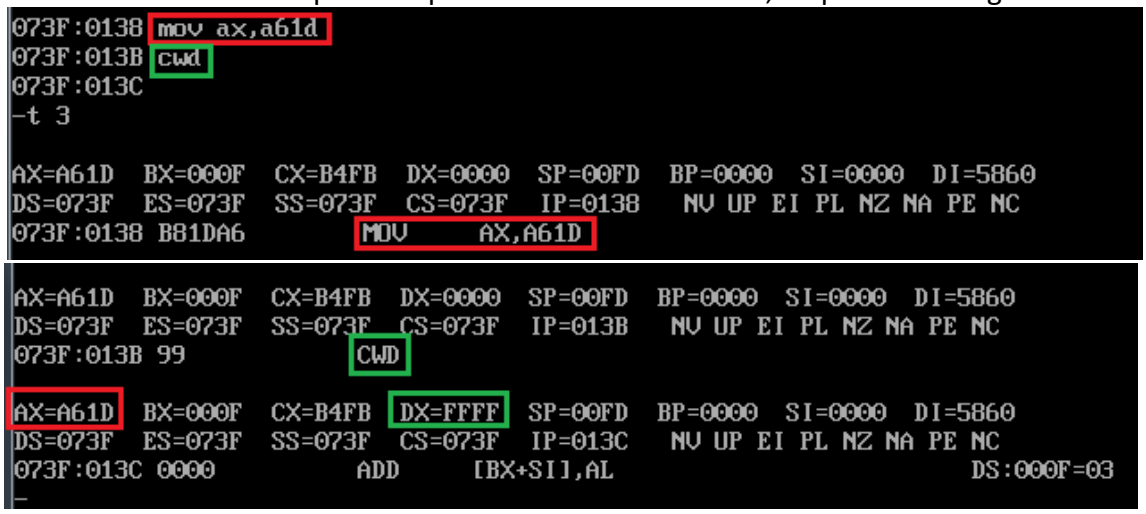
f) Colocar en el registro AL el valor 0x8E y por medio de la instrucción CBW convertirlo a una palabra que se almacene en AX, respetando el signo.



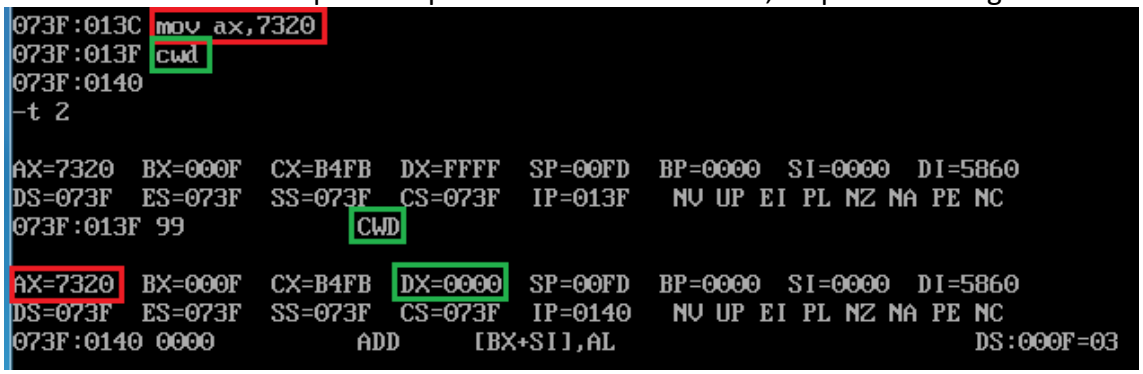
g) Colocar en el registro AL el valor 0x49 y por medio de la instrucción CBW convertirlo a una palabra que se almacene en AX, respetando el signo.



h) Colocar en el registro AX el valor 0xA61D y por medio de la instrucción CWD convertirlo a una doble palabra que se almacene en DX-AX, respetando el signo.



i) Colocar en el registro AX el valor 0x7320 y por medio de la instrucción CWD convertirlo a una doble palabra que se almacene en DX-AX, respetando el signo.



Conclusiones y comentarios

Sin duda alguna reafirme lo visto en clase. Agregue el diagrama con todas las instrucciones vistas anteriormente.

Dificultades en el desarrollo

Surgieron algunas dudas que resolví leyendo de nuevo los pdf de clase.

Referencias

PDF´s de clase.

Assembly Language For x86 Processors. Kip R. Irvine. 6th Edition