Actividad 2

Christian Geovany Muñoz Rodríguez

Ingeniería en computación

Código: 221350605

Seminario de Solución de Problemas de Traductores de lenguajes – D09 (Lunes y Miércoles de 1 a 3)

Maestro: José Juan Meza Espinosa

Universidad de Guadalajara

Centro Universitario de Ciencias Exactas e Ingenierías

10 de Febrero del 2023

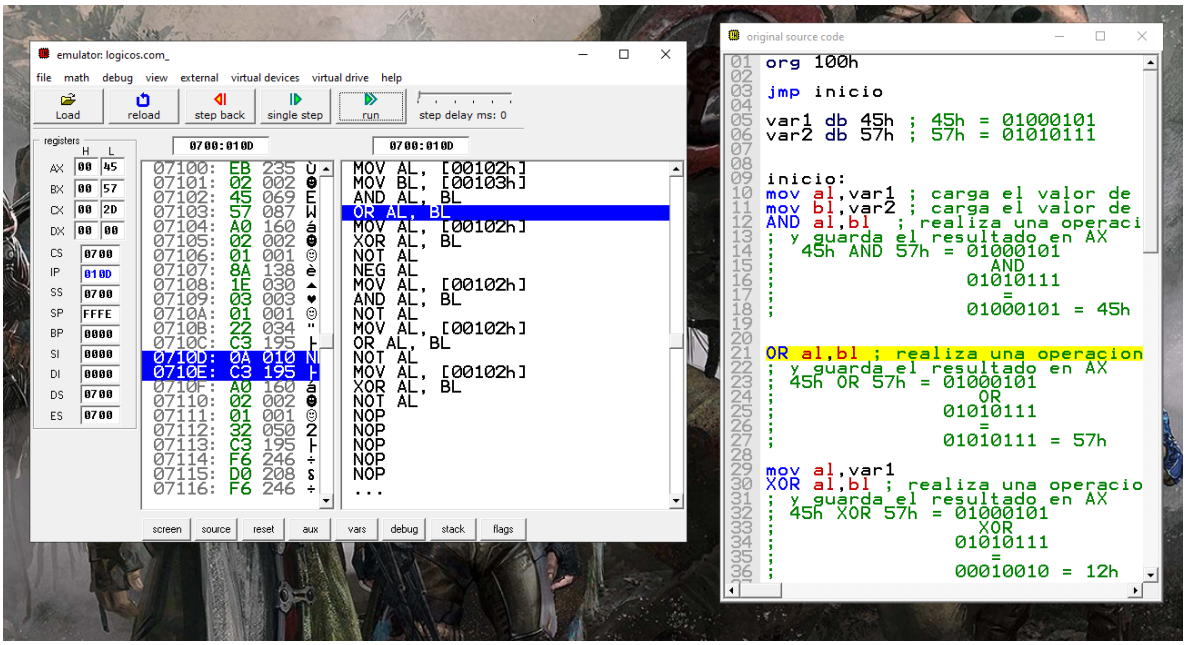
Código:

```
org 100h

jmp inicio

var1 db 45h ; 45h = 01000101
var2 db 57h ; 57h = 01010111


inicio:
  mov al,var1 ; carga el valor de var1 en el registro AX
  mov bl,var2 ; carga el valor de var2 en el registro BX
  AND al,bl  ; realiza una operacion logica AND entre los valores de AX y
BX
             ; y guarda el resultado en AX
             ;  45h AND 57h = 01000101
             ;                  AND
             ;                01010111
             ;                   =
             ;                01000101 = 45h


  OR al,bl ; realiza una operacion logica OR entre los valores de AX y BX
           ; y guarda el resultado en AX
           ; 45h OR 57h = 01000101
           ;                 OR
           ;              01010111
           ;                 =
```

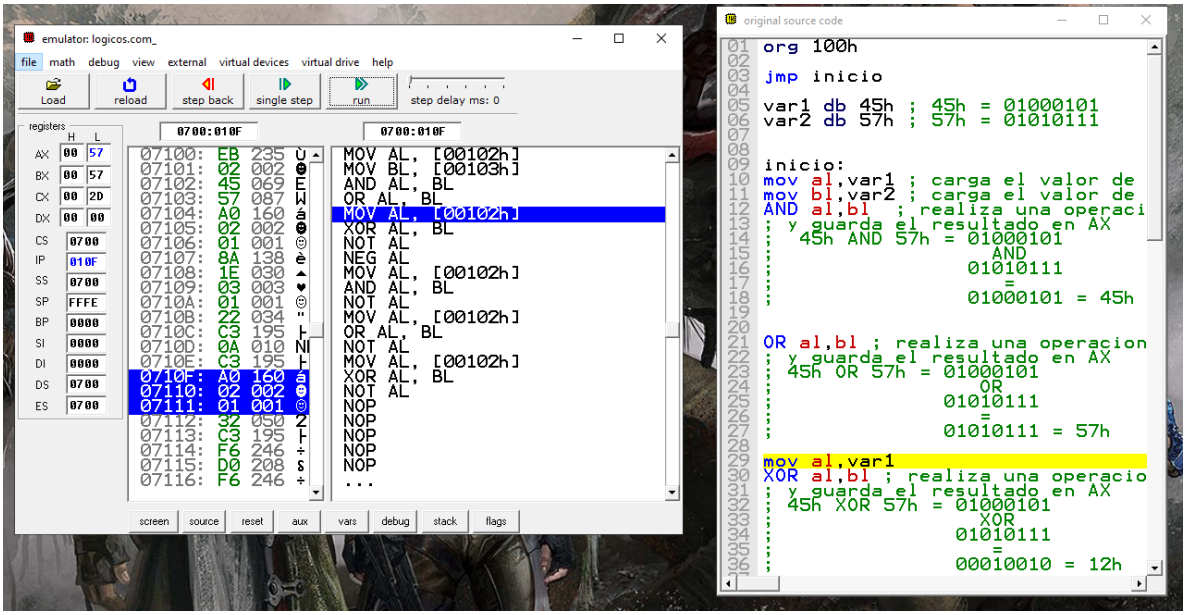
```
          ;              01010111 = 57h


  mov al,var1
  XOR al,bl ; realiza una operacion logica XOR entre los valores de AX y
BX
          ; y guarda el resultado en AX
          ; 45h XOR 57h = 01000101
          ;                 XOR
          ;               01010111
          ;                  =
          ;               00010010 = 12h


  NOT al ; realiza una operacion logica NOT en el valor de AX y guarda el
resultado en AX
         ; NOT 12h = 00010010 -> 11101101 = EDh


  NEG al ; realiza una operacion logica NOT en el valor de AX y guarda el
resultado en AX
         ; NEG EDh = 11101101 -> 00010011 = 13h


  ; NAND
  mov al, var1
  AND al,bl ; realiza la operación AND entre ambos registros
  NOT al; invierte el resultado obtenido con la operacion AND
       ; el resultado final es la negacion del resultado, la compuerta
NAND


  ;NOR
  mov al, var1
  OR al,bl; realiza la operacion OR entre ambos inputs
  NOT al; invierte el resultado obtenido con la operacion OR
```

```
        ; el resultado final es la negacion del resultado, la compuerta
NOR


  ;XNOR
  mov al, var1
  XOR al, bl; realizar operacion XOR entre AL y BL
  NOT al ; negar el resultado de la operacion XOR
```

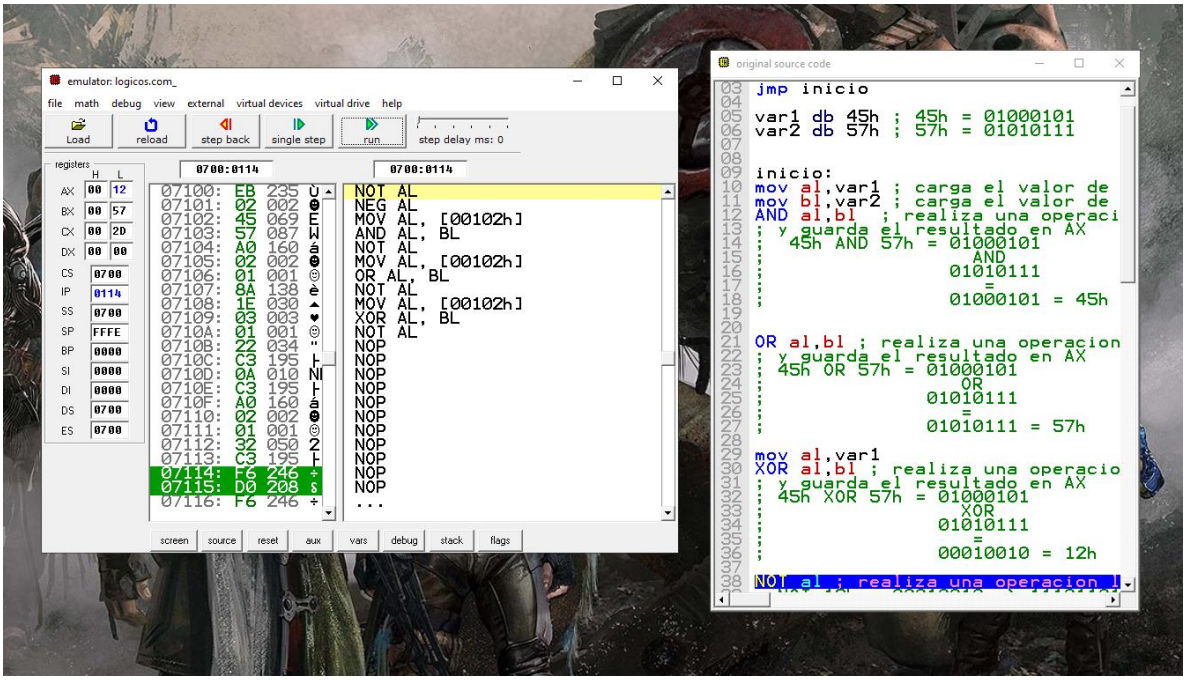
Pantallazos:



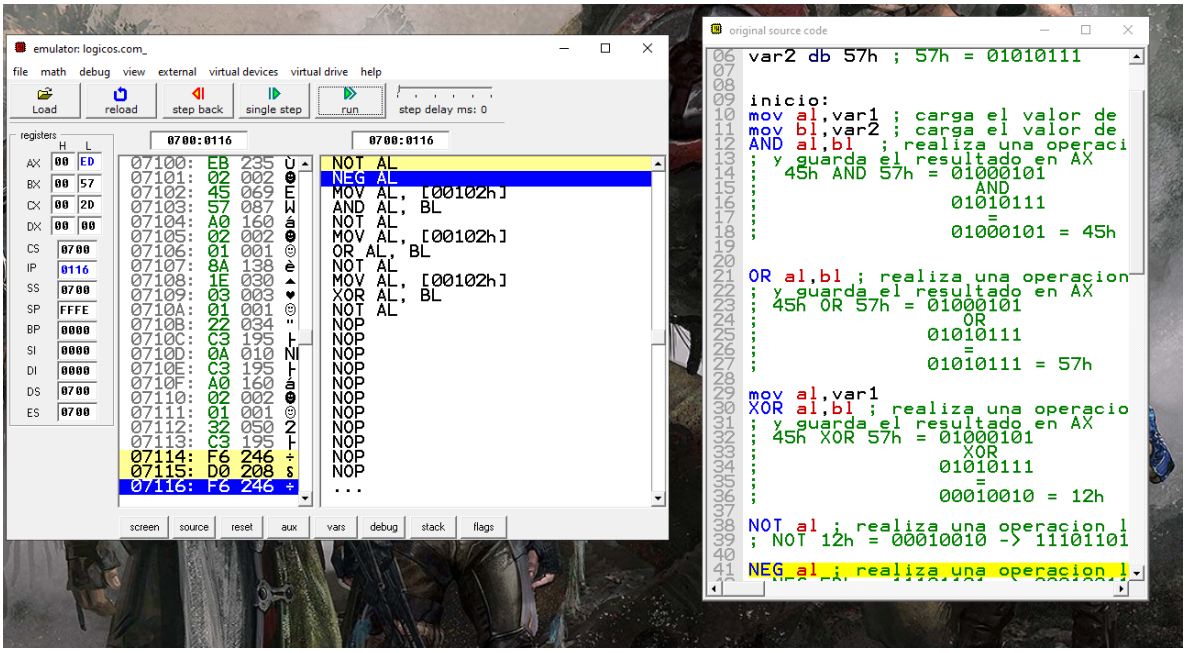
Aquí vemos como al aplicar el operador AND sobre AL y BL el registro AL conserva su valor (45h)



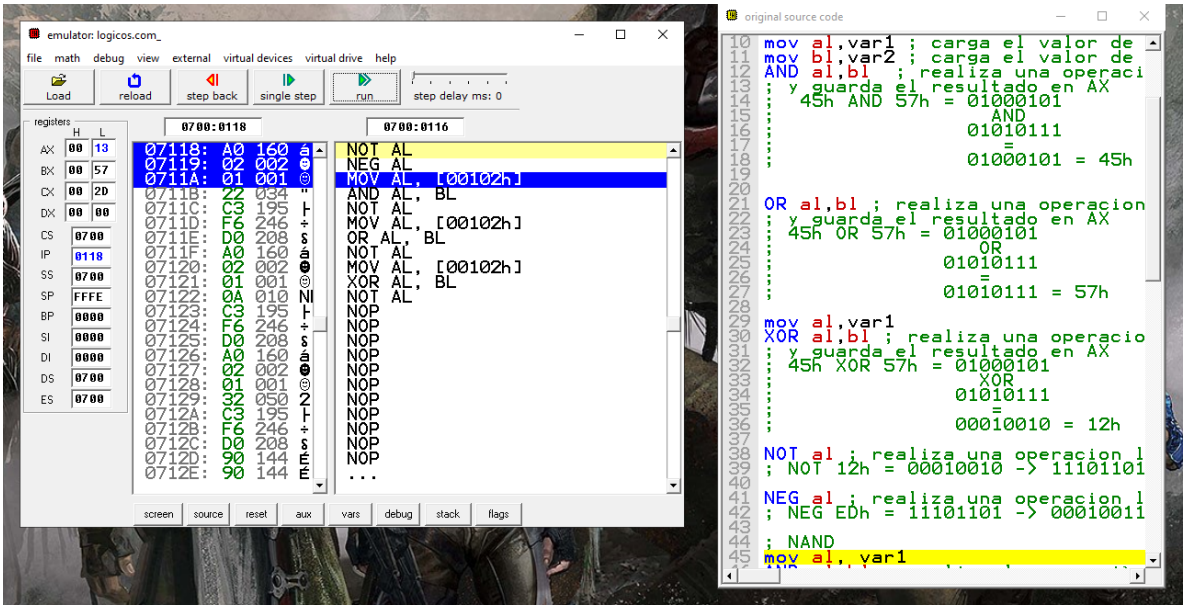
Aquí vemos como al aplicar el operador OR sobre AL y BL el registro AL toma el valor de 57h



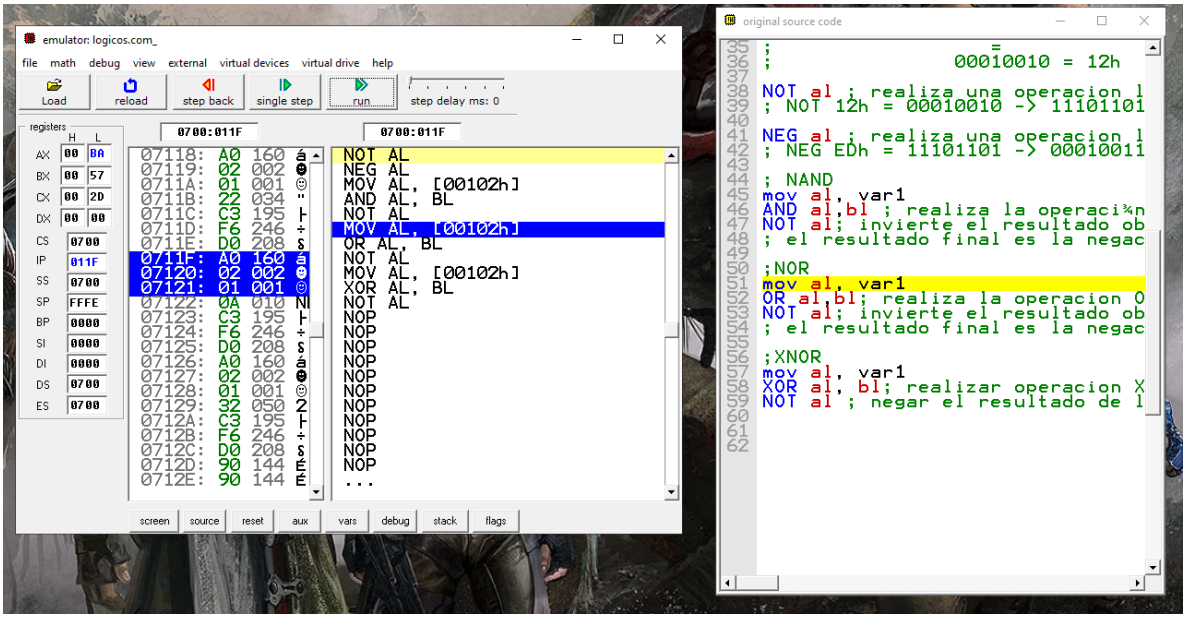
Aquí al aplicar el operador XOR entre AL y BL el registro AL toma el valor de 12h



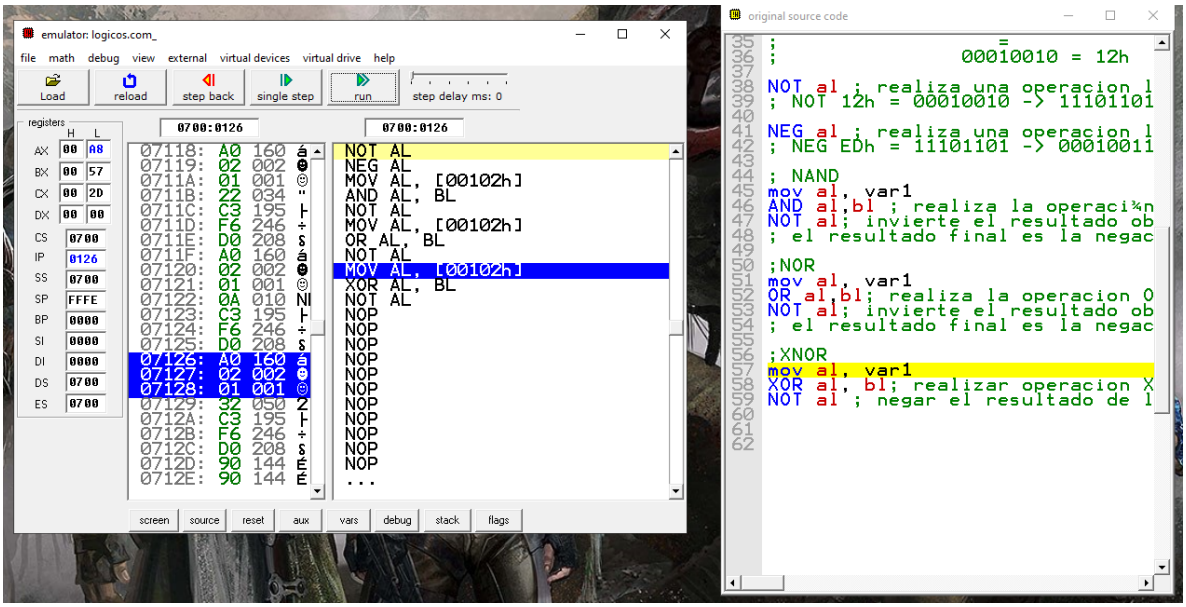
Al negar el registro AL toma el valor de EDh



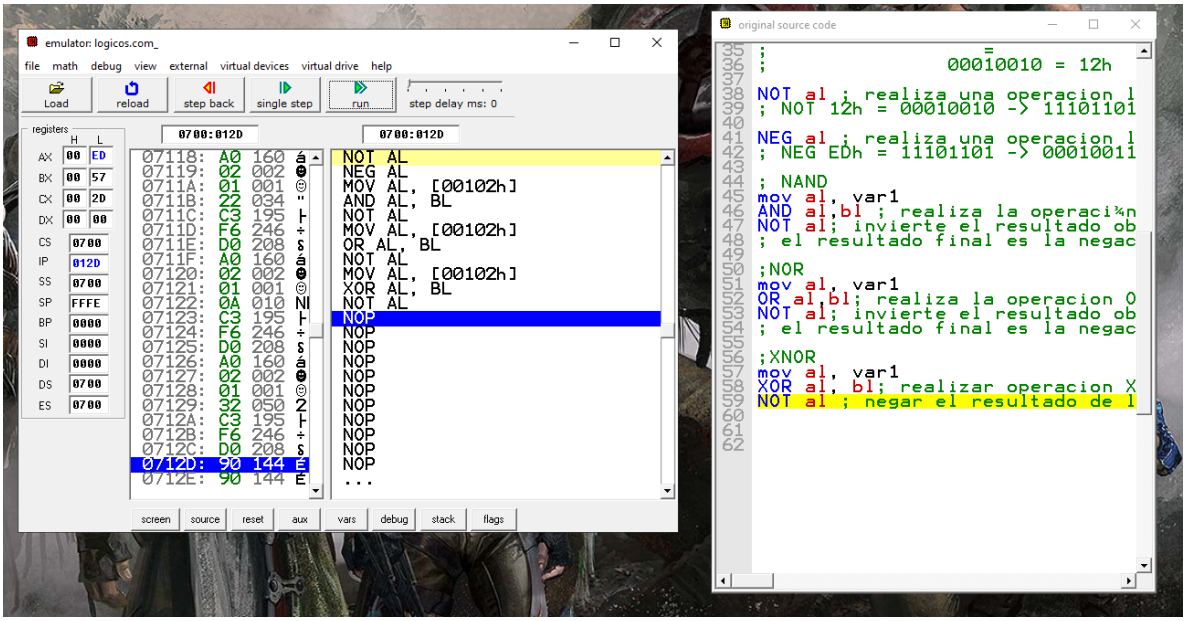
Al aplicar el operador NEG al registro AL vuelve a tomar el valor de 12h, pero sumado +1 (13h)



Al negar la operación AND entre al y bl obtenemos una operación NAND, el registro al toma el valor Bah



Al negar la operación OR entre al y bl obtenemos una operación NOR, al toma el valor A8h



Al negar la operación XOR entre al y bl obtenemos una operación XNOR, al toma el valor EDh

Desarrollo

Este código es un ejemplo de código en lenguaje ensamblador que ilustra el uso de algunas operaciones lógicas en el lenguaje de programación. En este código se utilizan las operaciones AND, OR, XOR, NOT, NEG y las compuertas lógicas NAND, NOR y XNOR.

Comenzando por el registro AX, el valor 45h es cargado en este registro y el valor 57h es cargado en el registro BX. Luego, se realiza una operación AND entre el valor de AX y BX, y se guarda el resultado en AX. La operación AND se realiza bit a bit comparando si ambos bits son 1, en cuyo caso el resultado es 1, de lo contrario el resultado es 0.

Luego, se realiza una operación OR entre el valor de AX y BX, y se guarda el resultado en AX. La operación OR se realiza bit a bit comparando si al menos uno de los bits es 1, en cuyo caso el resultado es 1, de lo contrario el resultado es 0.

A continuación, se realiza una operación XOR entre el valor de AX y BX, y se guarda el resultado en AX. La operación XOR se realiza bit a bit comparando si ambos bits son iguales o distintos, en cuyo caso el resultado es 0 o 1, respectivamente.

Luego, se realiza una operación NOT en el valor de AX y se guarda el resultado en AX. La operación NOT invierte el valor de cada bit, es decir, cambia los 0s por 1s y los 1s por 0s.

Por último, se realiza una operación NEG en el valor de AX y se guarda el resultado en AX. La operación NEG es equivalente a realizar una operación NOT seguida de sumar 1 al resultado.

En este código también se ilustra el uso de las compuertas lógicas NAND, NOR y XNOR. La compuerta NAND es equivalente a realizar una operación AND seguida de una operación NOT. La compuerta NOR es equivalente a realizar una operación OR seguida de una operación NOT. La compuerta XNOR es equivalente a realizar una operación XOR seguida de una operación NOT

Conclusiones

En conclusión, este código es un buen ejemplo de las operaciones lógicas y las compuertas que pueden ser realizadas en Ensamblador. Estas operaciones son esenciales en el mundo de la programación, ya que se utilizan en una gran variedad de aplicaciones, desde la manipulación de datos hasta la construcción de sistemas digitales complejos. Al comprender cómo funcionan y cómo se implementan estas operaciones en ensamblador, los estudiantes adquieren una comprensión profunda de los conceptos subyacentes y son capaces de aplicarlos en situaciones reales. En general, este código es un recurso valioso para cualquier persona interesada en aprender sobre programación en Ensamblador.

Bibliografía:

Brey, B. B. (2006). *Microprocesadores Intel : 8086/8088, 80186/80188, 80286, 80386 y 80486, Pentium, procesador Pentium Pro, Pentium II, Pentium III y Pentium 4: arquitectura, programación e interfaces*. Pearson Educación.