Universidad Autónoma de Baja California

Facultad de Ciencias Químicas e Ingeniería

ORGANIZACIÓN DE LAS COMPUTADORAS Y LENGUAJE ENSAMBLADOR

Practica 6

Introducción a los registros del procesador 8086, sus modos de direccionamiento e instrucciones básicas

Docente: Sanchez Herrera Mauricio Alonso

Alumno: Gómez Cárdenas Emmanuel Alberto

Matricula: 1261509

Contenido

TEORIA

Arquitectura interna del procesador 8086

El microprocesador 8086 es un circuito integrado con todas las funciones de un CPU, sin embargo, no puede ser simplemente utilizado como un microcontrolador ya que carece de memoria y periféricos. El 8086 está basado en la arquitectura CISC.

- Cuanto con dos bloques, BIU (Unidad de interfaz del bus) y EU (Unidad de ejecución).
- La BIU ejecuta todas las operaciones en bus (Obtener instrucciones, leer y escribir operandos de memoria y calcular las direcciones de los operandos de memoria).
- La EU ejecuta las instrucciones de la cola del byte del sistema de instrucciones.
- Ambas unidades operan asíncronamente para darle al 8086 un mecanismo de obtención y ejecución de instrucciones sobrepuestas denominado Pipelining. Esto genera un uso eficiente del bus del sistema y su rendimiento.
- La BIU contiene la cola de instrucciones, registro de segmentos, puntero de instrucción y el sumador de direcciones.
- La EU contiene el control de la circuitería, el decodificador de instrucciones, la ALU, los registros de índice y punteros y el registro de banderas.

Debug.exe

Es un ejecutable orientado a línea de comandos que puede ser utilizado como ensamblador, desensamblador o para volcar en hexadecimal un programa, permitiendo a los usuarios examinar los contenidos de memoria de una manera interactiva. El principal objetivo de un debugger es poder ejecutar un programa con condiciones controlados que permiten al programador seguir la operación y monitorear los cambios con el objetivo de determinar cuándo un programa está funcionando correctamente.

Procesador 8086

Estructura de un programa

Los programas de ensamblador en el 8086 cuentan con reglas, las cuales son:

- El código de programación a nivel ensamblador debe ser escrito en mayúsculas
- Las etiquetas deben ser seguidas por dos puntos.
- Todos los comentarios son escritos en minúsculas.
- La última línea del programa debe ser terminada con la directiva END
- 8086 cuanta con dos instrucciones extras para acceder a los datos; WORD PTR y BYTE PTR.

Modelo de memoria

Los modelos de memoria se refieren a un conjunto de seis diferentes modelos de memoria del CPU x86, esta especifica el tamaño de memoria que necesita el programa. Basada en esta directiva, el ensamblador asigna el monto de memoria requerida para los datos y el código. Estos modelos son:

- TINY: En este modelo el código y los datos ocupan u segmento físico y, por lo tanto, todos sus procedimientos y variables son direccionadas por defecto como NEAR (cercanos).
- SMALL: En este modelo, el código es puesto en un solo segmento y los datos en otro. Todos los procedimientos y variables son solo direccionadas como NEAR al apuntar a sus offsets.
- COMPACT: En este modelo, todos los elementos del código son puestos dentro de un segmento físico. Sin embargo, cada elemento de data puede ser puesto por defecto dentro de su propio segmento físico. Consecuentemente, los elementos de datos son direccionados al apuntar a la dirección del segmento y del offset.
- MEDIUM: Este modelo es contrario al compact. En este modelo los elementos de datos son tratados como NEAR y los elementos de códigos son direccionados como FAR (lejanos).
- LARGE: En este modelo, los elementos de código y datos son puestos en diferentes segmentos físicos. Procedimientos y variables son direccionados como FAR apuntando a las direcciones del segmento y del offset que contienen dicho elemento. Sin embargo, ningún Array de datos puede tener un tamaño que exceda un segmento físico (64 KB).
- HUGE: Este modelo es similar al modelo LARGE con la excepción que un Array de datos puede tener un tamaño que exceda un segmento físico (64KB).

Definición de segmentos

Un segmento es una unidad lógica de memoria que puede ser de hasta 64KB de grande. Cada segmento está hecho por locaciones contiguas de memoria. Es una unidad direccionable e independiente.

Declaración de variables

Una declaración de una variable inicia con la definición de la etiqueta o nombre de la variable, seguida de la directiva (DB) que indica el tamaño de memoria a reservar y por el ultimo el valor inicial de la variable. Ej. VARIABLE DB 12

Declaración de constantes

Para declarar una constante se utiliza la directiva EQU, esta directiva asigna un nombre simbólico al valor de una expresión. La sintaxis es parecida a la declaración de una variable, pero en vez de utilizar la directiva DB se utiliza la directiva EQU. La expresión a indicar puede ser una constante numérica (45), una referencia de dirección ([BX]), cualquier combinación de operaciones o símbolos que generen un valor numérico u otro nombre simbólico. Ej. CONSTANTE EQU 52

Modos de direccionamiento

- *Direccionamiento inmediato*

  Transfiere inmediatamente un dato del tamaño de una palabra o un byte a un registro destino o localidad de memoria (Ejemplo la instrucción MOV AL,22H copia la palabra de tamaño en bytes 22H en el registro AL).

- *Direccionamiento directo*

  En este modo en vez de dar el valor inmediato se utiliza la dirección de memoria donde se encuentra el operando. Ej. MOV BX, [1234h] (Copiar el valor que se encuentra en la dirección 1234.

- *Direccionamiento a registro*

  Se usa para transferir un byte o una palabra desde un registro fuente o desde una localidad de memora, hacia un registro destino o localidad de memoria. (Ejemplo: la instrucción MOV CX, DX copia el contenido del tamaño de la palabra del registro DX en el registro CX.

- *Direccionamiento de registro indirecto*

  Se usa para transferir un byte o palabra entre un registro y una localidad de memoria direccionada por el registro base o el registro índice. Los registros bases e índices son: BP, BX, DI, y SI (Ejemplo: la instrucción MOV AX, [BX] copia el dato de 16 bits (palabra) contenido en la localidad de memoria direccionada por el registro BX al registro AX).

- *Direccionamiento base más índice*

  Transfiere un byte o palabra entre un registro y la localidad de memoria direccionada por un registro base (BP o BX) más un registro índice (DI o SI) (Ejemplo: la instrucción MOV AX, [BX + SI] copia el contenido de 16-bits de la dirección apuntada por la suma de los registros BX y SI al registro AX).

- *Direccionamiento relativo a registro*

  Mueve un byte o palabra entre un registro y la localidad de memoria direccionada por un registro índice o base más un desplazamiento. (Ejemplo: la instrucción MOV AX, [BX+4] o MOV AX, ARRAY[BX]. La primera instrucción copia una palabra desde una dirección en el segmento de datos, formado por BX más 4 en el registro AX. La segunda instrucción transfiere el contenido de la localidad de memoria direccionada por ARRAY mas el contenido de BX en el registro AX.

- *Direccionamiento relativo a base más índice*

  Es usada para transferir una palabra o byte entre un registro y la localidad de memoria direccionada por una base y un registro índice más un desplazamiento. (Ejemplo: la instrucción MOV AX, ARRAY[BX+DI] o MOV AX, [BX+DI+4]. Ambas instrucciones copian una palabra de datos desde una localidad de memoria en el registro AX. La primera instrucción usa una dirección formada por la suma de ARRAY, BX, y DI y la segunda por la suma de BX, DI y 4).

Instrucciones:

De movimiento de datos:

- PUSH: Decrementa el puntero de pila por 2 y copia la palabra de la fuente especificada hacia donde apunta el puntero de pila. Esta instrucción no afecta ninguna bandera.

- POP: Copia una palabra de la pila al destino especificado en la instrucción, el destino puede ser un registro de propósito general, un registro de segmento o una locación de memoria. Después de copiar el puntero de pila se incrementa por 2 a la siguiente palabra. Esta instrucción no afecta ninguna bandera.

- PUSHF (PUSH FLAG REGISTER TO STACK): Esta instrucción decrementa el punto de pila por 2 y copia una palabra del registro de banderas a dos locaciones de memoria indicadas por el apuntador de pila. Esta instrucción afecta el registro banderas.

- POPF (POP WORD FROM TOP OF STACK TO FLAG REGISTER): Esta instrucción copia una palabra de dos locaciones del tope de la pila al registro de banderas e incrementa el puntero de pila por 2. Esta instrucción afecta el registro de banderas.

- XCHG: Intercambia el contenido de un registro con otro, o con el contenido en una locación(es) de memoria. No puede intercambiar contenido entre dos ubicaciones de memoria. Los registros o locaciones deben ser del mismo tipo (byte o Word (palabra)). Esta instrucción no afecta al registro de banderas. En este ejemplo intercambiamos los valores de AX y BX.

- XLAT (TRANSLATE A BYTE IN AL): Esta función traduce un byte de un código (8 bits o menos) a otro código. La instrucción reemplaza un byte en el registro AL. Esta instrucción no afecta el registro de banderas.

- IN: La función IN copia los datos de un puerto hacia el registro AL o el registro AX, dependiendo si es un puerto de 16 bits o de 8.

- OUT: Instrucción que copia un byte de AL o una palabra de AX al puerto especificado. Esta instrucción no afecta el registro de banderas.

Instrucciones aritméticas:

- ADD: Suma un numero de una fuente al número de un destino y pone el resultado en el destino especificado. La fuente puede ser un numero inmediato, un registro o una locación de memoria. La fuente y el destino deben de ser del mismo tipo (byte o palabra). Si se desea sumar un byte a una palabra se debe convertir el byte a palabra y llenar el byte con 0's antes de poder sumar. Las banderas afectadas en el registro son: AF (Auxiliary Carry), CF (Carry), OF (Overflow), SF (Sign), ZF (Zero).

- ADC: Funciona igual que la función ADD con la diferencia que esta instrucción incluye el estatus de la bandera de carry al resultado (ADD = X + Y; ADC = X + Y + CY)

- INC: Esta instrucción incrementa por 1 el registro o la ubicación en memoria especificada. Las banderas afectadas por la instrucción son: AF, OF, PF, SF y ZF. La bandera de carry no es afectada.

- SUB: Resta el número de la fuente al número del destino y pone el resultado en el destino. La fuente puede ser un numero inmediato, un registro o una locación de memoria, pero no dos locaciones de memoria. La fuente y el destino deben de ser del mismo tipo (byte o palabra). Si se desea restar un byte a una palabra se debe convertir el byte a palabra y llenar el byte con 0s antes de poder sumar. Las banderas afectadas en el registro son: AF, CF, OF, SF, ZF.

- SBB: Funciona igual que la función ADD con la diferencia que esta instrucción también resta el contenido de la bandera de carry al resultado.

- DEC: Esta instrucción decrementa por 1 el registro o la ubicación en memoria especificada. Las banderas afectadas por la instrucción son: AF, OF, PF, SF y ZF. La bandera de carry no es afectada.
  En este ejemplo podemos observar que la bandera de carry no es afectada a pesar de decrementar 1 desde 0h.

- MUL: Multiplica un byte sin signo de la fuente con un byte sin signo del registro AL o una palabra sin signo con el registro AX. Cuando un byte es multiplicado el resultado se guarda en AX. Cuando una palabra es multiplicada el resultado se guarda en AX y DX debido a que es un resultado demasiado grande para AX.
  Afecta las banderas CF y OF, mientras que las banderas AF, PF, SF y ZF son indefinidas.

- IMUL: Funciona igual que MUL con la excepción que los bytes o palabras a multiplicar cuentan con signo mientras que MUL solo trabaja con datos unsigned. Afecta las banderas CF y OF, mientras que las banderas AF, PF, SF y ZF son indefinidas.

- DIV: Esta instrucción divide una palabra entre un byte o una doble palabra (32 bits) entre una palabra. Cuando se divide una palabra esta debe estar en el registro AX y el divisor puede estar en un registro o en una ubicación de memoria. Después de la división AL contiene el cociente de 8 bits y AH el resultante de 8 bits.
  Cuando se divide una doble palabra (32 bits) la parte más significante de la palabra debe estar en el registro DX y la menos significante en AX. Después de la división AX contendrá el cociente de 16 bits y DX el resultante. Si se intenta dividir entre 0 se generará una interrupción.
  Todas las banderas son indefinidas después de una instrucción DIV.

- IDIV: Se utiliza parecido a la instrucción DIV, con la diferencia que este manipula datos con signo.
  Todas las banderas son indefinidas después de una instrucción IDIV.

- CMP: La instrucción compara un byte/palabra de la fuente con el del destino especificado. La fuente puede ser un número inmediato, un registro o una locación en memoria. El destino solo puede ser un registro o una locación en memoria. Ambos no pueden ser locaciones de memoria al mismo tiempo. La comparación se hace al restar el byte/palabra fuente del byte/palabra destino. La única bandera no afectada por la instrucción es DF.

- CBW (Convert signed Byte to signed Word): Copia el bit de signo en AL a todos los bits en AH. Esta instrucción no afecta el registro de banderas.

- CWD (Convert signed Word to signed Double word): Copia el bit de signo de una palabra en AX a todos los bits de DX. Esta instrucción no afecta al registro de banderas.

Instrucciones lógicas y de manipulación de bits:

- AND: Esta instrucción aplica un AND a cada bit de la palabra/byte fuente con el mismo bit de la palabra/byte. El resultado se coloca en el destino indicado. Las banderas CF y OF son 0 después de la instrucción. PF, SF y ZF son actualizadas y AF es indefinida. PF solo importa para los operandos de 8 bits.

- OR: Esta instrucción aplica un OR a cada bit de la palabra/byte fuente con el mismo bit de la palabra/byte. El resultado se coloca en el destino indicado. Las banderas CF y OF son 0 después de la instrucción. PF, SF y ZF son actualizadas y AF es indefinida. PF solo importa para los operandos de 8 bits.

- XOR: Esta instrucción aplica un XOR a cada bit de la palabra/byte fuente con el mismo bit de la palabra/byte. El resultado se coloca en el destino indicado. Las banderas CF y OF son 0 después de la instrucción. PF, SF y ZF son actualizadas y AF es indefinida. PF solo importa para los operandos de 8 bits.

- NOT: Invierte cada bit (con complemento a 1) del byte o la palabra especificada. El destino puede ser un registro o una locación de memoria. Esta instrucción no afecta el registro de banderas.

- TEST: Esta instrucción aplica ANDs a cada bit de la palabra o byte destino con la palabra o byte fuente. Ningún operando resulta modificado, solamente el registro de banderas se actualiza.

- SHL y SAL: Ambos mnemónicos indican la misma instrucción, esta instrucción desplaza cada bit un numero de posiciones a la izquierda. El MSB se desplaza hacia la bandera de carry y en el espacio del LSB se pone un 0. Afecta a las banderas ZF, PF, OF y CF

- SHR: Esta instrucción desplaza cada bit un numero de posiciones a la derecha. El LSB se desplaza hacia la bandera de carry y en el espacio del MSB se pone un 0. Afecta a las banderas ZF, PF, OF, AF y CF

- SAR: Esta instrucción desplaza cada bit un numero de posiciones a la derecha.
  El desplazamiento aritmético a comparación con los desplazamientos lógicos, toman en cuenta el signo. El LSB es desplazado hacia CF. Desplazar a la derecha equivale a dividir entre 2. Afecta a las banderas ZF, PF, OF, AF y CF

- ROL: Rota todos los bits a la izquierda de una palabra o bytes específicos un numero de posiciones de bit. El MSB es rotado hacia el LSB y también es copiado en CF. Solo afecta a las banderas CF y OF.

- RCL: Rota todos los bits a la izquierda de una palabra o bytes específicos. El MSB es rotado hacia el CF y CF es rotado hacia el LSB. Solo afecta a las banderas CF y OF.

- ROR: Rota todos los bits a la derecha de una palabra o bytes específicos. El LSB es rotado hacia el MSB y también es copiado en CF. Solo afecta a las banderas CF y OF.

- RCR: Rota todos los bits a la derecha de una palabra o bytes específicos. El LSB es rotado hacia el CF y CF es rotado hacia el MSB. Solo afecta a las banderas CF y OF.

DESARROLLO

Parte 1 Registros y banderas

Identificar dentro del programa debug.exe: los grupos de registros y las banderas.

Con la utilización del comando –r podemos observar claramente los de segmentos (DS, ES, SS, CS), y los registros de propósito general, que consisten en los registros de datos (AX, BX, CX, DX), los registros los registros de punteros (SP, BP, IP) y los registros de índice (SI, DI). Por ultimo nos muestra las banderas, estas son:

- Overflow (OV, Overflow NV, no overflow)
- Direction (DN, decrementa UP, incrementa)
- Interrupt (EI, activado DI, desactivado)
- Sign (NG, negativo PL, positivo)
- Zero (ZR, cero NZ, no cero)
- Auxiliary Carry (AC, carry auxiliar NA no existe carry auxiliar)
- Parity (PE, par PO, impar)
- Carry (CY, existe carry NC, no existe carry)

Parte 2 Modos de direccionamiento

Direccionamiento inmediato

MOV AX,1234h (Copiar el valor 1234h al registro AX)

```
-A
073F:0100 MOV AX,1234
073F:0103
-R
AX=0000  BX=0000  CX=0000  DX=0000  SP=00FD  BP=0000  SI=0000  DI=0000
DS=073F  ES=073F  SS=073F  CS=073F  IP=0100   NV UP EI PL NZ NA PO NC
073F:0100 B83412        MOV     AX,1234
-T

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 0000          ADD     [BX+SI],AL                      DS:0000=CD
-
```

Direccionamiento directo

MOV BX, [1234h] (Copiar el valor que se encuentra en la dirección 1234, que, en este caso es 3412 (Little endian) al registro BX).

```
-R
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=0109   NV UP EI PL NZ NA PO NC
073F:0109 8B1E3412      MOV     BX,[1234]                       DS:1234=1234
-D DS:1234
073F:1230             34 12 00 00-00 00 00 00 00 00 00 00      4...........
073F:1240  00 00 00 00 00 00 00 00-00 00 00 00 00 00 00 00   ................
073F:1250  00 00 00 00 00 00 00 00-00 00 00 00 00 00 00 00   ................
073F:1260  00 00 00 00 00 00 00 00-00 00 00 00 00 00 00 00   ................
073F:1270  00 00 00 00 00 00 00 00-00 00 00 00 00 00 00 00   ................
073F:1280  00 00 00 00 00 00 00 00-00 00 00 00 00 00 00 00   ................
073F:1290  00 00 00 00 00 00 00 00-00 00 00 00 00 00 00 00   ................
073F:12A0  00 00 00 00 00 00 00 00-00 00 00 00 00 00 00 00   ................
073F:12B0  00 00 00 00                                       ....
-T

AX=1234  BX=1234  CX=0000  DX=0000  SP=00FD  BP=0000  SI=0000  DI=0000
DS=073F  ES=073F  SS=073F  CS=073F  IP=010D   NV UP EI PL NZ NA PO NC
073F:010D 00AEFE00      ADD     [BP+00FE],CH                    SS:00FE=00
-s
```

Direccionamiento a registro

ADD AX, BX

```
-A
073F:0106 ADD AX,BX
073F:0108
-R
AX=0000  BX=01FF  CX=0000  DX=0000  SP=00FD  BP=0000  SI=0000  DI=0000
DS=073F  ES=073F  SS=073F  CS=073F  IP=0106   NV UP EI PL NZ AC PO NC
073F:0106 01D8          ADD     AX,BX
-T

AX=01FF  BX=01FF  CX=0000  DX=0000  SP=00FD  BP=0000  SI=0000  DI=0000
DS=073F  ES=073F  SS=073F  CS=073F  IP=0108   NV UP EI PL NZ NA PE NC
073F:0108 01D8          ADD     AX,BX
```

Direccionamiento de registro indirecto

ADD AX, [BX]. Se suma lo que se encuentre en la dirección BX a AX y el resultado se guarda en AX

$$ax = ax + [bx] \rightarrow ax = ax + [01FFh] \rightarrow ax = ax + CCDDh \rightarrow ax = AABBh + CCDDh$$

```
-R
AX=AABB  BX=01FF  CX=CCDD  DX=0000  SP=00FD  BP=0000  SI=0000  DI=0000
DS=073F  ES=073F  SS=073F  CS=073F  IP=010E   NV UP EI NG NZ AC PO NC
073F:010E 0307          ADD     AX,[BX]                         DS:01FF=CCDD
-A
073F:0110
-R
AX=AABB  BX=01FF  CX=CCDD  DX=0000  SP=00FD  BP=0000  SI=0000  DI=0000
DS=073F  ES=073F  SS=073F  CS=073F  IP=010E   NV UP EI NG NZ AC PO NC
073F:010E 0307          ADD     AX,[BX]                         DS:01FF=CCDD
-T

AX=7798  BX=01FF  CX=CCDD  DX=0000  SP=00FD  BP=0000  SI=0000  DI=0000
DS=073F  ES=073F  SS=073F  CS=073F  IP=0110   OV UP EI PL NZ AC PO CY
073F:0110 0000          ADD     [BX+SI],AL                      DS:01FF=DD
```

Direccionamiento base más índice

$$MOV\ AX, [BX + SI] \rightarrow AX, [AAAh + BBBh] \rightarrow AX, [1665] \rightarrow AX, AABBh$$

En este caso, en la ubicación de memoria 1665 se encuentra BB y AA en la 1666 guardados con Little endian en mente, por lo tanto, a la hora de copiarlos se transfieren como AABB.

```
-r
AX=0000  BX=0AAA  CX=0000  DX=0000  SP=00FD  BP=0000  SI=0BBB  DI=0000
DS=073F  ES=073F  SS=073F  CS=073F  IP=0100   NV UP EI PL NZ NA PO NC
073F:0100 8B00          MOV     AX,[BX+SI]                      DS:1665=AABB
-t

AX=AABB  BX=0AAA  CX=0000  DX=0000  SP=00FD  BP=0000  SI=0BBB  DI=0000
DS=073F  ES=073F  SS=073F  CS=073F  IP=0102   NV UP EI PL NZ NA PO NC
073F:0102 0000          ADD     [BX+SI],AL                      DS:1665=BB
-d ds:1665
073F:1660                BB AA 00-00 00 00 00 00 00 00 00        ..........
```

Direccionamiento relativo a registro

$MOV\ AX, [BX, 0Ah]$. $AX = [BX + 0A] \rightarrow AX = [AA0 + A] \rightarrow AX = [AAA] \rightarrow AX = CCDD$

```
-r
AX=0000  BX=0AA0  CX=0000  DX=0000  SP=00FD  BP=0000  SI=0BBB  DI=0000
DS=073F  ES=073F  SS=073F  CS=073F  IP=0105   NV UP EI PL NZ NA PO NC
073F:0105 8B470A        MOV     AX,[BX+0A]                      DS:0AAA=CCDD
-t

AX=CCDD  BX=0AA0  CX=0000  DX=0000  SP=00FD  BP=0000  SI=0BBB  DI=0000
DS=073F  ES=073F  SS=073F  CS=073F  IP=0108   NV UP EI PL NZ NA PO NC
073F:0108 0000          ADD     [BX+SI],AL                      DS:165B=00
-d ds:aaa
073F:0AA0                            DD CC 00 00 00 00              ......
```

Direccionamiento relativo a base más índice

$$MOV\ AX, [BX + DI + 04].\ AX = [BX + DI + 04] \rightarrow AX = [100 + 19 + 4]$$
$$AX = [11D] \rightarrow AX = 2E00$$

```
-R
AX=0000  BX=0100  CX=0000  DX=0000  SP=00FD  BP=0000  SI=0000  DI=0019
DS=073F  ES=073F  SS=073F  CS=073F  IP=0100   NV UP EI PL NZ NA PO NC
073F:0100 8B4104        MOV     AX,[BX+DI+04]                   DS:011D=2E00
-T

AX=2E00  BX=0100  CX=0000  DX=0000  SP=00FD  BP=0000  SI=0000  DI=0019
DS=073F  ES=073F  SS=073F  CS=073F  IP=0103   NV UP EI PL NZ NA PO NC
073F:0103 0000          ADD     [BX+SI],AL                      DS:0100=8B
-D DS:11D
073F:0110                                     00 2E 07                ...
073F:0120  00 00 00 00 00 00 00 00-00 00 00 00 00 00 00 00   ................
```

Parte 3 Instrucciones disponibles en el procesador 8086

De movimiento de datos:

- PUSH AX. Guarda el valor de AX en la pila y decrementa SP por 2.

```
-A
073F:0100 PUSH AX
073F:0101
-R
AX=AABB  BX=0000  CX=0000  DX=0000  SP=00FD  BP=0000  SI=0000  DI=0000
DS=073F  ES=073F  SS=073F  CS=073F  IP=0100   NV UP EI PL NZ NA PO NC
073F:0100 50            PUSH    AX
-T

AX=AABB  BX=0000  CX=0000  DX=0000  SP=00FB  BP=0000  SI=0000  DI=0000
DS=073F  ES=073F  SS=073F  CS=073F  IP=0101   NV UP EI PL NZ NA PO NC
073F:0101 0000          ADD     [BX+SI],AL                          DS:0000=CD
```

- POP BX. Copia el valor desde la pila hacia BX y aumenta SP por 2.

```
-A
073F:0101 POP BX
073F:0102
-R
AX=AABB  BX=0000  CX=0000  DX=0000  SP=00FB  BP=0000  SI=0000  DI=0000
DS=073F  ES=073F  SS=073F  CS=073F  IP=0101   NV UP EI PL NZ NA PO NC
073F:0101 5B            POP     BX
-T

AX=AABB  BX=AABB  CX=0000  DX=0000  SP=00FD  BP=0000  SI=0000  DI=0000
DS=073F  ES=073F  SS=073F  CS=073F  IP=0102   NV UP EI PL NZ NA PO NC
073F:0102 0000          ADD     [BX+SI],AL                          DS:AABB=00
```

- PUSHF decrementa el punto de pila por 2 y copia una palabra del registro de banderas a dos locaciones de memoria indicadas por el apuntador de pila.

```
073F:0102 PUSHF
073F:0103
-R
AX=AABB  BX=AABB  CX=0000  DX=0000  SP=00FD  BP=0000  SI=0000  DI=0000
DS=073F  ES=073F  SS=073F  CS=073F  IP=0102   NV UP EI PL NZ NA PO NC
073F:0102 9C            PUSHF
-T

AX=AABB  BX=AABB  CX=0000  DX=0000  SP=00FB  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 0000          ADD     [BX+SI],AL                          DS:AABB=00
```

- POPF. copia una palabra de dos locaciones del tope de la pila al registro de banderas e incrementa el puntero de pila por 2

```
-R
AX=FFFE  BX=F000  CX=0000  DX=0000  SP=00FB  BP=0000  SI=0000  DI=0000
DS=073F  ES=073F  SS=073F  CS=073F  IP=0109   NV UP EI NG NZ NA PO CY
073F:0109 9D            POPF
-T

AX=FFFE  BX=F000  CX=0000  DX=0000  SP=00FD  BP=0000  SI=0000  DI=0000
DS=073F  ES=073F  SS=073F  CS=073F  IP=010A   NV UP EI PL NZ NA PO NC
073F:010A 0000          ADD     [BX+SI],AL                      DS:F000=00
```

- XCHG AX, BX. En este ejemplo intercambiamos los valores de AX y BX.

```
-A
073F:010A XCHG AX,BX
073F:010C
-R
AX=AAAA  BX=BBBB  CX=0000  DX=0000  SP=00FD  BP=0000  SI=0000  DI=0000
DS=073F  ES=073F  SS=073F  CS=073F  IP=010A   NV UP EI PL NZ NA PO NC
073F:010A 87C3          XCHG    AX,BX
-T

AX=BBBB  BX=AAAA  CX=0000  DX=0000  SP=00FD  BP=0000  SI=0000  DI=0000
DS=073F  ES=073F  SS=073F  CS=073F  IP=010C   NV UP EI PL NZ NA PO NC
073F:010C 0000          ADD     [BX+SI],AL                      DS:AAAA=00
```

- XLAT. La instrucción reemplaza un byte en el registro AL.

```
-A100
073F:0100 XLAT
073F:0101
-R
AX=AADD  BX=01FF  CX=CCDD  DX=0000  SP=00FF  BP=0000  SI=0000  DI=0000
DS=073F  ES=0000  SS=073F  CS=073F  IP=0100   NV UP EI NG NZ NA PE NC
073F:0100 D7            XLAT
-T

AX=AA00  BX=01FF  CX=CCDD  DX=0000  SP=00FF  BP=0000  SI=0000  DI=0000
DS=073F  ES=0000  SS=073F  CS=073F  IP=0101   NV UP EI NG NZ NA PE NC
```

- IN AX, 0B. Copia los datos de un puerto hacia el registro AL o el registro AX, dependiendo si es un puerto de 16 bits o de 8.

```
-R
AX=0000  BX=073F  CX=0000  DX=0000  SP=00FD  BP=0000  SI=0000  DI=0000
DS=073F  ES=073F  SS=073F  CS=073F  IP=0102   NV UP EI PL NZ NA PO NC
073F:0102 E50B          IN      AX,0B
-T

AX=00FF  BX=073F  CX=0000  DX=0000  SP=00FD  BP=0000  SI=0000  DI=0000
DS=073F  ES=073F  SS=073F  CS=073F  IP=0104   NV UP EI PL NZ NA PO NC
073F:0104 0000          ADD     [BX+SI],AL                      DS:073F=00
```

- OUT 3B, AX. Instrucción que copia un byte de AL o una palabra de AX al puerto especificado.

```
-A
073F:0104 OUT 3B,AX
073F:0106
-R
AX=00FF  BX=073F  CX=0000  DX=0000  SP=00FD  BP=0000  SI=0000  DI=0000
DS=073F  ES=073F  SS=073F  CS=073F  IP=0104   NV UP EI PL NZ NA PO NC
073F:0104 E73B          OUT     3B,AX
-T

AX=00FF  BX=073F  CX=0000  DX=0000  SP=00FD  BP=0000  SI=0000  DI=0000
DS=073F  ES=073F  SS=073F  CS=073F  IP=0106   NV UP EI PL NZ NA PO NC
073F:0106 0000          ADD     [BX+SI],AL                          DS:073F=00
```

Instrucciones aritméticas:

- ADD AX, BX. => $AX + BX \rightarrow 00FF + FF01 = 0000$ *y Activa la bandera de carry*

$$banderas\ afectadas:$$

$$NA \rightarrow AC, \qquad NZ \rightarrow ZR, \qquad PO \rightarrow PE, \qquad NC \rightarrow CY$$

```
-A
073F:0106 ADD AX,BX
073F:0108
-R
AX=00FF  BX=FF01  CX=0000  DX=0000  SP=00FD  BP=0000  SI=0000  DI=0000
DS=073F  ES=073F  SS=073F  CS=073F  IP=0106   NV UP EI PL NZ NA PO NC
073F:0106 01D8          ADD     AX,BX
-T

AX=0000  BX=FF01  CX=0000  DX=0000  SP=00FD  BP=0000  SI=0000  DI=0000
DS=073F  ES=073F  SS=073F  CS=073F  IP=0108   NV UP EI PL ZR AC PE CY
073F:0108 0000          ADD     [BX+SI],AL                          DS:FF01=00
```

- ADC AX, BX. => $0000 + FF01 + CY \rightarrow FF01 + CY \rightarrow FF01 + 1 \rightarrow AX = FF02$

$$banderas\ afectadas$$

$$PL \rightarrow NG, \qquad ZR \rightarrow NZ, \qquad AC \rightarrow NA, \qquad PE \rightarrow PO, \qquad CY \rightarrow NC$$

```
-A
073F:0108 ADC AX,BX
073F:010A
-R
AX=0000  BX=FF01  CX=0000  DX=0000  SP=00FD  BP=0000  SI=0000  DI=0000
DS=073F  ES=073F  SS=073F  CS=073F  IP=0108   NV UP EI PL ZR AC PE CY
073F:0108 11D8          ADC     AX,BX
-T

AX=FF02  BX=FF01  CX=0000  DX=0000  SP=00FD  BP=0000  SI=0000  DI=0000
DS=073F  ES=073F  SS=073F  CS=073F  IP=010A   NV UP EI NG NZ NA PO NC
073F:010A 0000          ADD     [BX+SI],AL                          DS:FF01=00
```

- INC AX. En este ejemplo podemos observar que la bandera de carry no es afectada a pesar de incrementar 1 desde FFFFh.

```
073F:010A INC AX
073F:010B
-R
AX=FFFF  BX=FF01  CX=0000  DX=0000  SP=00FD  BP=0000  SI=0000  DI=0000
DS=073F  ES=073F  SS=073F  CS=073F  IP=010A   NV UP EI NG NZ NA PO NC
073F:010A 40            INC     AX
-T

AX=0000  BX=FF01  CX=0000  DX=0000  SP=00FD  BP=0000  SI=0000  DI=0000
DS=073F  ES=073F  SS=073F  CS=073F  IP=010B   NV UP EI PL ZR AC PE NC
073F:010B 0000          ADD     [BX+SI],AL                      DS:FF01=00
```

- SUB AX, BX => $AX = AX - BX \rightarrow AX = FF00 - 0F00 \rightarrow AX = F000$

```
-A
073F:0111 SUB AX,BX
073F:0113
-R
AX=FF00  BX=0F00  CX=0000  DX=0000  SP=00FD  BP=0000  SI=0000  DI=0000
DS=073F  ES=073F  SS=073F  CS=073F  IP=0111   NV UP EI NG NZ AC PE CY
073F:0111 29D8          SUB     AX,BX
-T

AX=F000  BX=0F00  CX=0000  DX=0000  SP=00FD  BP=0000  SI=0000  DI=0000
DS=073F  ES=073F  SS=073F  CS=073F  IP=0113   NV UP EI NG NZ NA PE NC
073F:0113 0000          ADD     [BX+SI],AL                      DS:0F00=00
```

- SBB AX, BX

  $AX = AX - BX - CY \rightarrow AX = 0F00 - 0F00 - 1 \rightarrow AX = 0000 - 1 \rightarrow AX = FFFF$

```
-A
073F:0115 SBB AX,BX
073F:0117
-R
AX=0F00  BX=0F00  CX=0000  DX=0000  SP=00FD  BP=0000  SI=0000  DI=0000
DS=073F  ES=073F  SS=073F  CS=073F  IP=0115   NV UP EI PL NZ NA PE CY
073F:0115 19D8          SBB     AX,BX
-T

AX=FFFF  BX=0F00  CX=0000  DX=0000  SP=00FD  BP=0000  SI=0000  DI=0000
DS=073F  ES=073F  SS=073F  CS=073F  IP=0117   NV UP EI NG NZ AC PE CY
073F:0117 0000          ADD     [BX+SI],AL                      DS:0F00=00
```

- DEC AX. En este ejemplo podemos observar que la bandera de carry no es afectada a pesar de decrementar 1 desde 0h.

```
073F:011A DEC AX
073F:011B
-R
AX=0000  BX=0F00  CX=0000  DX=0000  SP=00FD  BP=0000  SI=0000  DI=0000
DS=073F  ES=073F  SS=073F  CS=073F  IP=011A   NV UP EI PL NZ NA PE NC
073F:011A 48            DEC     AX
-T

AX=FFFF  BX=0F00  CX=0000  DX=0000  SP=00FD  BP=0000  SI=0000  DI=0000
DS=073F  ES=073F  SS=073F  CS=073F  IP=011B   NV UP EI NG NZ AC PE NC
073F:011B 0034          ADD     [SI],DH                         DS:0000=CD
```

- MUL BX => $AAAAh * BBBBh \rightarrow 7D26D82Eh \rightarrow AX = D82Eh\ y\ DX = 7D26$

```
-r
AX=AAAA  BX=BBBB  CX=0100  DX=0000  SP=00FD  BP=0000  SI=0000  DI=0000
DS=073F  ES=073F  SS=073F  CS=073F  IP=0100   OV UP EI PL NZ NA PO CY
073F:0100 F7E3          MUL     BX
-t

AX=D82E  BX=BBBB  CX=0100  DX=7D26  SP=00FD  BP=0000  SI=0000  DI=0000
DS=073F  ES=073F  SS=073F  CS=073F  IP=0102   OV UP EI PL NZ NA PO CY
073F:0102 EBFC          JMP     0100
```

- IMUL BX => $AABBh * CCDDh = -21829 * -13091 \rightarrow 1108676F \rightarrow AX = 676F\ \ y\ \ DX = 1108$

```
073F:0100 IMUL BX
073F:0102
-R
AX=AABB  BX=CCDD  CX=0000  DX=0000  SP=00FD  BP=0000  SI=0000  DI=0000
DS=073F  ES=073F  SS=073F  CS=073F  IP=0100   NV UP EI PL NZ NA PO NC
073F:0100 F7EB          IMUL    BX
-T

AX=676F  BX=CCDD  CX=0000  DX=1108  SP=00FD  BP=0000  SI=0000  DI=0000
DS=073F  ES=073F  SS=073F  CS=073F  IP=0102   OV UP EI PL NZ NA PO CY
073F:0102 0000          ADD     [BX+SI],AL                      DS:CCDD=00
```

- DIV BX => $\frac{AX}{BX} \rightarrow \frac{AAAA}{A} \rightarrow 1111$

```
-A
073F:0100 DIV BX
073F:0102
-R
AX=AAAA  BX=000A  CX=0000  DX=0000  SP=00FD  BP=0000  SI=0000  DI=0000
DS=073F  ES=073F  SS=073F  CS=073F  IP=0100   NV UP EI PL NZ NA PO NC
073F:0100 F7F3          DIV     BX
-T

AX=1111  BX=000A  CX=0000  DX=0000  SP=00FD  BP=0000  SI=0000  DI=0000
DS=073F  ES=073F  SS=073F  CS=073F  IP=0102   NV UP EI PL NZ NA PO NC
073F:0102 0000          ADD     [BX+SI],AL                      DS:000A=4F
```

- IDIV BX => $\frac{AX}{BX} -> \frac{1111}{AA} \rightarrow 0019.\ DX = RESULTANTE$

```
-R
AX=1111  BX=00AA  CX=0000  DX=0000  SP=00FD  BP=0000  SI=0000  DI=0000
DS=073F  ES=073F  SS=073F  CS=073F  IP=0100   NV UP EI PL NZ NA PO NC
073F:0100 F7FB          IDIV    BX
-T

AX=0019  BX=00AA  CX=0000  DX=0077  SP=00FD  BP=0000  SI=0000  DI=0000
DS=073F  ES=073F  SS=073F  CS=073F  IP=0102   NV UP EI PL NZ NA PO NC
073F:0102 EBFC          JMP     0100
```

- CMP AX, BX.

```
-A 100
073F:0100 CMP AX,BX
073F:0102
-R
AX=0019  BX=00AA  CX=0000  DX=0077  SP=00FD  BP=0000  SI=0000  DI=0000
DS=073F  ES=073F  SS=073F  CS=073F  IP=0102   NV UP EI PL NZ NA PO NC
073F:0102 EBFC          JMP     0100
-T

AX=0019  BX=00AA  CX=0000  DX=0077  SP=00FD  BP=0000  SI=0000  DI=0000
DS=073F  ES=073F  SS=073F  CS=073F  IP=0100   NV UP EI PL NZ NA PO NC
073F:0100 39D8          CMP     AX,BX
```

- CBW En la captura podemos observar como AH es afectada.

```
AX=0080  BX=00AA  CX=0000  DX=0000  SP=00FD  BP=0000  SI=0000  DI=0000
DS=073F  ES=073F  SS=073F  CS=073F  IP=0100   NV UP EI PL NZ NA PO NC
073F:0100 98            CBW
-T

AX=FF80  BX=00AA  CX=0000  DX=0000  SP=00FD  BP=0000  SI=0000  DI=0000
DS=073F  ES=073F  SS=073F  CS=073F  IP=0101   NV UP EI PL NZ NA PO NC
073F:0101 99            CWD
```

- CWD. En la captura podemos observar como DX es afectado.

```
-R
AX=FF80  BX=00AA  CX=0000  DX=0000  SP=00FD  BP=0000  SI=0000  DI=0000
DS=073F  ES=073F  SS=073F  CS=073F  IP=0101   NV UP EI PL NZ NA PO NC
073F:0101 99            CWD
-T

AX=FF80  BX=00AA  CX=0000  DX=FFFF  SP=00FD  BP=0000  SI=0000  DI=0000
DS=073F  ES=073F  SS=073F  CS=073F  IP=0102   NV UP EI PL NZ NA PO NC
073F:0102 EBFC          JMP     0100
```

Instrucciones lógicas y de manipulación de bits:

- AND AX,BX. $1111h\ AND\ 0101h \rightarrow AX = 101h$

```
-a
073F:0100 AND ax,bx
073F:0102
-a
073F:0102 jmp 100
073F:0104
-R
AX=1111  BX=0101  CX=0000  DX=0000  SP=00FD  BP=0000  SI=0000  DI=0000
DS=073F  ES=073F  SS=073F  CS=073F  IP=0100   NV UP EI PL NZ NA PO NC
073F:0100 21D8          AND     AX,BX
-T

AX=0101  BX=0101  CX=0000  DX=0000  SP=00FD  BP=0000  SI=0000  DI=0000
DS=073F  ES=073F  SS=073F  CS=073F  IP=0102   NV UP EI PL NZ NA PO NC
073F:0102 EBFC          JMP     0100
```

- OR AX,BX.=> $5555h\ OR\ AAAAh \rightarrow 5 = 0101;\quad A = 1010;\quad 5\ OR\ A = 1111 = F$

```
-R
AX=5555  BX=AAAA  CX=0000  DX=0000  SP=00FD  BP=0000  SI=0000  DI=0000
DS=073F  ES=073F  SS=073F  CS=073F  IP=0100   NV UP EI NG NZ NA PE NC
073F:0100 09D8          OR      AX,BX
-T

AX=FFFF  BX=AAAA  CX=0000  DX=0000  SP=00FD  BP=0000  SI=0000  DI=0000
DS=073F  ES=073F  SS=073F  CS=073F  IP=0102   NV UP EI NG NZ NA PE NC
073F:0102 EBFC          JMP     0100
-S
```

- XOR AX, BX. => $5555h\ XOR\ AAAAh \rightarrow$

$$5 = 0101;\quad A = 1010;$$
$$5\ XOR\ A = 1111 = F$$
$$5\ XOR\ 5 = 0$$

```
-R
AX=5555  BX=AA55  CX=0000  DX=0000  SP=00FD  BP=0000  SI=0000  DI=0000
DS=073F  ES=073F  SS=073F  CS=073F  IP=0100   NV UP EI NG NZ NA PE NC
073F:0100 31D8          XOR     AX,BX
-T

AX=FF00  BX=AA55  CX=0000  DX=0000  SP=00FD  BP=0000  SI=0000  DI=0000
DS=073F  ES=073F  SS=073F  CS=073F  IP=0102   NV UP EI NG NZ NA PE NC
073F:0102 EBFC          JMP     0100
-S
```

- NOT AX

```
AX=FF00  BX=AA55  CX=0000  DX=0000  SP=00FD  BP=0000  SI=0000  DI=0000
DS=073F  ES=073F  SS=073F  CS=073F  IP=0100   NV UP EI NG NZ NA PE NC
073F:0100 F7D0          NOT     AX
-T

AX=00FF  BX=AA55  CX=0000  DX=0000  SP=00FD  BP=0000  SI=0000  DI=0000
DS=073F  ES=073F  SS=073F  CS=073F  IP=0102   NV UP EI NG NZ NA PE NC
073F:0102 EBFC          JMP     0100
```

- TEST AX,BX => $00FFh\ AND\ FF00h = 0$

por lo tanto el registo de banderas indica que el resultado es cero

```
AX=00FF  BX=FF00  CX=0000  DX=0000  SP=00FD  BP=0000  SI=0000  DI=0000
DS=073F  ES=073F  SS=073F  CS=073F  IP=0100   NV UP EI NG NZ NA PE NC
073F:0100 85C3          TEST    AX,BX
-T

AX=00FF  BX=FF00  CX=0000  DX=0000  SP=00FD  BP=0000  SI=0000  DI=0000
DS=073F  ES=073F  SS=073F  CS=073F  IP=0102   NV UP EI PL ZR NA PE NC
073F:0102 EBFC          JMP     0100
```

- SHL AX,1 => $AAAAh \ll 1 \rightarrow 1010\ 1010\ 1010\ 1010b \ll 1$

$$\rightarrow CF = 1; 0101\ 0101\ 0100b = 5554h$$

```
AX=AAAA  BX=FF00  CX=0000  DX=0000  SP=00FD  BP=0000  SI=0000  DI=0000
DS=073F  ES=073F  SS=073F  CS=073F  IP=0100   NV UP EI NG NZ AC PE NC
073F:0100 D1E0          SHL     AX,1
-T

AX=5554  BX=FF00  CX=0000  DX=0000  SP=00FD  BP=0000  SI=0000  DI=0000
DS=073F  ES=073F  SS=073F  CS=073F  IP=0102   OV UP EI PL NZ AC PO CY
073F:0102 EBFC          JMP     0100
```

- SHR AX,1

$$5555 \gg 1 \rightarrow 0101\ 0101\ 0101\ 0101b \gg 1 \rightarrow$$
$$CF = 1; 0010\ 1010\ 1010\ 1010b = 2AAAh$$

```
-R
AX=5555  BX=FF00  CX=0000  DX=0000  SP=00FD  BP=0000  SI=0000  DI=0000
DS=073F  ES=073F  SS=073F  CS=073F  IP=0100   NV UP EI PL NZ AC PO NC
073F:0100 D1E8          SHR     AX,1
-T

AX=2AAA  BX=FF00  CX=0000  DX=0000  SP=00FD  BP=0000  SI=0000  DI=0000
DS=073F  ES=073F  SS=073F  CS=073F  IP=0102   NV UP EI PL NZ AC PE CY
073F:0102 EBFC          JMP     0100
```

- SAR AX,1

$$AAAAh \gg 1$$
$$1010\ 1010\ 1010\ 1010b \gg 1$$
$$CF = 0; 1101\ 0101\ 0101b = D555h$$

```
AX=AAAA  BX=FF00  CX=0000  DX=0000  SP=00FD  BP=0000  SI=0000  DI=0000
DS=073F  ES=073F  SS=073F  CS=073F  IP=0100   OV UP EI PL NZ AC PE NC
073F:0100 D1F8          SAR     AX,1
-T

AX=D555  BX=FF00  CX=0000  DX=0000  SP=00FD  BP=0000  SI=0000  DI=0000
DS=073F  ES=073F  SS=073F  CS=073F  IP=0102   NV UP EI NG NZ AC PE NC
073F:0102 EBFC          JMP     0100
```

- ROL AX,1

$$AAAAh \gg 1$$
$$1010\ 1010\ 1010\ 1010b \gg 1$$
$$CF = 1; 0101\ 0101\ 0101\ 0101b = 5555h$$

```
AX=AAAA  BX=FF00  CX=0000  DX=0000  SP=00FD  BP=0000  SI=0000  DI=0000
DS=073F  ES=073F  SS=073F  CS=073F  IP=0100   NV UP EI NG NZ AC PE NC
073F:0100 D1C0          ROL     AX,1
-T

AX=5555  BX=FF00  CX=0000  DX=0000  SP=00FD  BP=0000  SI=0000  DI=0000
DS=073F  ES=073F  SS=073F  CS=073F  IP=0102   OV UP EI NG NZ AC PE CY
073F:0102 EBFC          JMP     0100
```

- RCL AX,1

  Se le introduce el estado del carry en el lugar del LSB

$$CF = 1; 0000\ 0000\ 0000\ 0000b \ll 0001h$$
$$CF = 0; 0000\ 0000\ 0000\ 0001b = 8000h$$

```
AX=0000  BX=FF00  CX=0000  DX=0000  SP=00FD  BP=0000  SI=0000  DI=0000
DS=073F  ES=073F  SS=073F  CS=073F  IP=0100   OV UP EI PL NZ AC PO CY
073F:0100 D1D0          RCL     AX,1
-T

AX=0001  BX=FF00  CX=0000  DX=0000  SP=00FD  BP=0000  SI=0000  DI=0000
DS=073F  ES=073F  SS=073F  CS=073F  IP=0102   NV UP EI PL NZ AC PO NC
073F:0102 EBFC          JMP     0100
```

- ROR AX,1

$$0001h \gg 1$$
$$CF = 0; 0000\ 0000\ 0000\ 0001b \gg 0001h$$
$$CF = 1; 1000\ 0000\ 0000\ 0000b = 8000h$$

```
AX=0001  BX=FF00  CX=0000  DX=0000  SP=00FD  BP=0000  SI=0000  DI=0000
DS=073F  ES=073F  SS=073F  CS=073F  IP=0100   NV UP EI PL NZ AC PO NC
073F:0100 D1C8          ROR     AX,1
-T

AX=8000  BX=FF00  CX=0000  DX=0000  SP=00FD  BP=0000  SI=0000  DI=0000
DS=073F  ES=073F  SS=073F  CS=073F  IP=0102   OV UP EI PL NZ AC PO CY
073F:0102 EBFC          JMP     0100
```

- RCR AX,1

  Primer RCR

$$0001h \gg 1$$
$$CF = 1; 0000\ 0000\ 0000\ 0001b \gg 0001h$$
$$CF = 1; 1000\ 0000\ 0000\ 0000b = 8000h$$

  Segundo RCR

$$8000h \gg 1$$
$$CF = 1; 1000\ 0000\ 0000\ 0001b \gg 8000h$$
$$CF = 0; 1100\ 0000\ 0000\ 0000b = C000h$$

```
AX=0001  BX=FF00  CX=0000  DX=0000  SP=00FD  BP=0000  SI=0000  DI=0000
DS=073F  ES=073F  SS=073F  CS=073F  IP=0102   OV UP EI PL NZ AC PO CY
073F:0102 D1D8          RCR     AX,1
-T

AX=8000  BX=FF00  CX=0000  DX=0000  SP=00FD  BP=0000  SI=0000  DI=0000
DS=073F  ES=073F  SS=073F  CS=073F  IP=0104   OV UP EI PL NZ AC PO CY
073F:0104 D1D8          RCR     AX,1
-T

AX=C000  BX=FF00  CX=0000  DX=0000  SP=00FD  BP=0000  SI=0000  DI=0000
DS=073F  ES=073F  SS=073F  CS=073F  IP=0106   NV UP EI PL NZ AC PO NC
```

CONCLUSIONES

Gracias al debug.exe podemos tener un primer contacto con los registros y el comportamiento del procesador 8086 de una manera interactiva, lo cual nos ayuda a entender su comportamiento un poco más. El hecho de poder monitorear cada registro y ver como es afectado por una instrucción en específico es bastante interesante ya que nos prepararan para las siguientes prácticas que son mucho más complejas.

REFERENCIAS

2 Assembly Language Programming. Cs.unm.edu. (2020). Retrieved 5 December 2020, from https://www.cs.unm.edu/~maccabe/classes/341/labman/node2.html.

(2020). Retrieved 5 December 2020, from https://examradar.com/8086-microprocessor/.

Architecture of 8086 - GeeksforGeeks. GeeksforGeeks. (2020). Retrieved 5 December 2020, from https://www.geeksforgeeks.org/architecture-of-8086/.

Assembly - Constants - Tutorialspoint. Tutorialspoint.com. (2020). Retrieved 5 December 2020, from https://www.tutorialspoint.com/assembly_programming/assembly_constants.htm.

Debug (command). En.wikipedia.org. (2020). Retrieved 5 December 2020, from https://en.wikipedia.org/wiki/DEBUG_(DOS_Command).

Intel Memory Model. En.wikipedia.org. (2020). Retrieved 5 December 2020, from https://en.wikipedia.org/wiki/Intel_Memory_Model.

Intel.com. (2020). Retrieved 5 December 2020, from https://www.intel.com/content/dam/www/public/us/en/documents/manuals/64-ia-32-architectures-software-developer-instruction-set-reference-manual-325383.pdf.

Know Assembly Language Programming of 8086. ElProCus - Electronic Projects for Engineering Students. (2020). Retrieved 5 December 2020, from https://www.elprocus.com/8086-assembly-language-programs-explanation/.

Memory Segmentation in 8086 Microprocessor - GeeksforGeeks. GeeksforGeeks. (2020). Retrieved 5 December 2020, from https://www.geeksforgeeks.org/memory-segmentation-8086-microprocessor/.

MS-DOS DEBUG Program. Thestarman.pcministry.com. (2020). Retrieved 5 December 2020, from https://thestarman.pcministry.com/asm/debug/debug.htm.

Utm.mx. (2020). Retrieved 5 December 2020, from http://www.utm.mx/~jjf/le/LE_APENDICE_D.pdf.