TRABAJANDO CON LOS REGISTROS Y LA MEMORIA

Mover datos a Registros

-A 100

0CB6:0100 MOV AX,1234

0CB6:0103 MOV DX,ABCD

0CB6:0106 MOV AH,DL

0CB6:0108

-t

AX=1234 BX=0000 CX=0000 DX=0000 SP=FFEE BP=0000 SI=0000 DI=0000

DS=0CB6 ES=0CB6 SS=0CB6 CS=0CB6 IP=0103 NV UP EI PL NZ NA PO NC

0CB6:0103 BACDAB MOV DX,ABCD

-t

AX=1234 BX=0000 CX=0000 DX=ABCD SP=FFEE BP=0000 SI=0000 DI=0000

DS=0CB6 ES=0CB6 SS=0CB6 CS=0CB6 IP=0106 NV UP EI PL NZ NA PO NC

0CB6:0106 88D4 MOV AH,DL

-t

AX=CD34 BX=0000 CX=0000 DX=ABCD SP=FFEE BP=0000 SI=0000 DI=0000

DS=0CB6 ES=0CB6 SS=0CB6 CS=0CB6 IP=0108 NV UP EI PL NZ NA PO NC

0CB6:0108 094404 OR [SI+04],AX DS:0004=9A00

1. EL valor en el registro AX es **CD34h** y en el registro dx es **ABCDh**.

-a 108

0CB6:0108 MOV BX,AX

0CB6:010A

-t

AX=CD34 BX=CD34 CX=0000 DX=ABCD SP=FFEE BP=0000 SI=0000 DI=0000

DS=0CB6 ES=0CB6 SS=0CB6 CS=0CB6 IP=010A NV UP EI PL NZ NA PO NC

0CB6:010A 04BC ADD AL,BC

1. EL nuevo valor de AX y BX es **CD34h.**

-a 100

0CB6:0100 MOV AX,1000

0CB6:0103 MOV BX,1010

0CB6:0106 ADD AX,BX

0CB6:0108

-r ip

IP 0112

:100

-t

AX=1000 BX=CD34 CX=0000 DX=ABCD SP=FFEE BP=0000 SI=0001 DI=0000

DS=0CB6 ES=0CB6 SS=0CB6 CS=0CB6 IP=0103 NV UP EI PL NZ NA PO NC

0CB6:0103 BB1010 MOV BX,1010

-t

AX=1000 BX=1010 CX=0000 DX=ABCD SP=FFEE BP=0000 SI=0001 DI=0000

DS=0CB6 ES=0CB6 SS=0CB6 CS=0CB6 IP=0106 NV UP EI PL NZ NA PO NC

0CB6:0106 01D8 ADD AX,BX

-t

AX=2010 BX=1010 CX=0000 DX=ABCD SP=FFEE BP=0000 SI=0001 DI=0000

DS=0CB6 ES=0CB6 SS=0CB6 CS=0CB6 IP=0108 NV UP EI PL NZ NA PO NC

0CB6:0108 89C3 MOV BX,AX

* mov AX,1000 le asigna al registro AX el valor de 1000
* mov BX,1010 → le asigna al registro BX el valor de 1010
* add AX,BX → suma los valores de AX + BX poniendo el resultado en AX.

-e 1000

0CB6:1000 75.FF 65.00

-e 1010

0CB6:1010 76.00 6F.11

-a 100

0CB6:0100 MOV AX,[1000]

0CB6:0103 MOV BX,[1010]

0CB6:0107 ADD AX,BX

0CB6:0109 MOV [1020],AX

0CB6:010C

-r ip

IP 010C

:100

-t

AX=00FF BX=1010 CX=0000 DX=ABCD SP=FFEE BP=0000 SI=0001 DI=0000

DS=0CB6 ES=0CB6 SS=0CB6 CS=0CB6 IP=0103 NV UP EI PL NZ NA PO NC

0CB6:0103 8B1E1010 MOV BX,[1010]

DS:1010=1100

-t

AX=00FF BX=1100 CX=0000 DX=ABCD SP=FFEE BP=0000 SI=0001 DI=0000

DS=0CB6 ES=0CB6 SS=0CB6 CS=0CB6 IP=0107 NV UP EI PL NZ NA PO NC

0CB6:0107 01D8 ADD AX,BX

-t

AX=11FF BX=1100 CX=0000 DX=ABCD SP=FFEE BP=0000 SI=0001 DI=0000

DS=0CB6 ES=0CB6 SS=0CB6 CS=0CB6 IP=0109 NV UP EI PL NZ NA PE NC

0CB6:0109 A32010 MOV [1020],AX

DS:1020=7365

-

-d 1000,1030

0CB6:1000 FF 00 62 61 20 71 75 65-20 6C 6F 73 20 6E 75 65

0CB6:1010 00 11 73 20 61 72 63 68-69 76 6F 73 20 73 65 20

0CB6:1020 FF 11 63 72 69 62 61 6E-20 63 6F 72 72 65 63 74

0CB6:1030 61

1. Con ip en 100:

* mov AX,[1000] → en la dirección 0100 esta instrucción copia el valor en la dirección 1000 a AX.
* mov BX,[1010] → en la dirección 0103 esta instrucción copia el valor de la dirección 1010 en BX.
* add AX,BX → suma los valores de AX + BX colocando el resultado en AX.
* mov [1020],AX → se copia el valor en AX en la dirección 1020.

Interrupciones (INT)

-a 100

0CB6:0100 mov ax, 200

0CB6:0103 mov dx, 41

0CB6:0106 int 21

0CB6:0108

-g 108

A

AX=0241 BX=0000 CX=0000 DX=0041 SP=FFEE BP=0000 SI=0000 DI=0000

DS=0CB6 ES=0CB6 SS=0CB6 CS=0CB6 IP=0108 NV UP EI PL NZ NA PO NC

0CB6:0108 094404 OR [SI+04],AX

DS:0004=9A00

1. Los pasos a seguir para imprimir la letra b son:
2. Utilizando el comando A del Debug apuntando a 100, escribimos en el registro AX el valor 200h. para escribir un carácter en la salida estándar.
3. En el registro DX el valor 42h que es la letra B en código ASCII.
4. Escribir la interrupción INT 21 en la posición de memoria 0106h.
5. Apuntar el registro IP a la dirección 100h.
6. Utilizamos el comando G (go) del Debug, para ejecutar las instrucciones y a continuación de ella, la dirección 0108, que es en la que queremos parar.

Se imprime el carácter “B” por medio del DOS y luego devuelve el control a nuestro programa.

-a 100

-0CB6:0100 mov ax,200

-0CB6:0103 mov dx,42

-0CB6:0106 int21

-0CB6:0108

-r ip

IP 0108

:100

-g 108

B

AX=0242 BX=0000 CX=0000 DX=0042 SP=FFEE BP=0000 SI=0000 DI=0000

DS=0CB6 ES=0CB6 SS=0CB6 CS=0CB6 IP=0108 NV UP EI PL NZ NA PO NC

0CB6:0108 094404 OR [SI+04],AX

DS:0004=9A00

La Interrupción de salida (INT 20h)

-a 102

0CB6:0102 int 20

0CB6:0104

-g 102

El programa ha terminado de forma normal

1. Obtenemos en pantalla 'El programa ha terminado de forma normal'.
2. Lo pasos a seguir para escribir "\*" en la pantalla y DEVOLVER EL CONTROL A DOS son los siguientes:
3. Utilizando el comando A del Debug apuntando a 100, escribimos en el registro AX el valor 200h.
4. Escribimos en el registro BX el valor 2Ah.
5. Escribir la instrucción INT 21
6. Apuntar el registro IP a la dirección 0100h
7. Utilizamos el comando G para ejecutar las instrucciones y a continuación de ellas la dirección 108 que es en la que queremos parar.

-a 100

0D41:0100 mov ax, 200

0D41:0103 mov dx, 2a

0D41:0106 int 21  
0D41:0108

-g 108  
\*  
AX=022A BX=0000 CX=0000 DX=002A SP=FFEE BP=0000 SI=0000 DI=0000  
DS=0D41 ES=0D41 SS=0D41 CS=0D41 IP=0108 NV UP EI PL NZ NA PO NC  
0D41:0108 207974 AND [BX+DI+74],BH DS:0074=20

Escribir una cadena de caracteres

-a 100

1389:0100 mov ah,09

1389:0102 mov dx, 0200

1389:0105 int 21

1389:0107 int 20

1389:0109

-e 200

1389:0200 51.51 75.75 65.65 20.20 65.65 73.73 20.20 65.65

1389:0208 73.73 74.74 6F.6f 3F.3f 24.24 00.

-g 10a

Que es esto?

Program terminated normally

1. El valor 09 guardado en la parte alta del registro AX, y la INT 21, imprimen una cadena de caracteres que toman del registro DX.

-a 100

1389:0100 mov ah, 09

1389:0102 mov dx, 0200

1389:0105 int 21

1389:0107 int 20

1389:0109

-e 200

1389:0200 00.56 00.61 00.6c 00.65 00.72 00.69 00.61 00.0a

1389:0208 00.0d 00.47 00.69 00.75 00.6c 00.69 00.61 00.6e

1389:0210 00.61 00.0a 00.0d 00.53 00.61 00.6e 00.64 00.72

1389:0218 00.6f 00.0a 00.0d 00.45 00.75 00.67 00.65 00.6e

1389:0220 00.69 00.61 00.24

-g

Valeria

Giuliana

Sandro

Eugenia

Program terminated normally



-a 100

1389:0100 mov ax, 4100

1389:0103 mov bx, 0041

1389:0106 add ax, bx

1389:0108 mov dl, ah

1389:010A mov ah, 02

1389:010C int 21

1389:010E mov dl, al

1389:0110 int 21

1389:0112 int 20

1389:0114

-g

AA

Program terminated normally

Si lo “Corremos” por medio del comando G del Debug nos imprimirá los caracteres AA; esto se logra de la siguiente manera:

MOV AX ingresa al registro el valor 4100. Lo mismo hace con MOV BX y el valor 0041.   
ADD: suma los valores guardados en AX y BX y los guarda nuevamente en AX.   
MOV DL, AH: desplaza los datos almacenado en la parte alta del registro AX (AH), a la parte baja del registro DX (DL).

MOV AH, 02: imprime un carácter. La interrupción INT 21 hace que dicho carácter pueda ser impreso.

MOV DL, AL: desplaza los datos de la parte baja de AX, a la parte baja de DX, actuando la INT 21 para que se realice.

INT 20: finaliza el programa.

El resultado obtenido se debe a los valores 4141 que corresponden a las letras AA en el lenguaje ASCII.

LA ALU: DESPLAZAMIENTOS. OPERACIONES LÓGICAS Y COMPARACIONES

Rotaciones (Desplazamientos circulares) y el Indicador de Acarreo

1. Resultado: 0001h

Acarreo: CY → hay acarreo

Desbordamiento: NV → no hay desbordamiento



|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Acarreo** | **Registro BL en Binario** | | **BL en hexa** | **Observaciones** |
| NC | 0000 | 00001 | 0001 |  |
| NC | 0000 | 0010 | 0002 | Se realiza una rotación a la izquierda con acarreo por un bit, multiplicándose por 2 el número dado. |
| NC | 0000 | 0100 | 0004 |  |
| NC | 0000 | 1000 | 0008 |  |
| NC | 0001 | 0000 | 0010 |  |
| NC | 0010 | 0000 | 0020 |  |
| NC | 0100 | 0000 | 0040 |  |
| NC | 1000 | 0000 | 0080 |  |
| CY | 0000 | 0000 | 0000 | Se produjo un acarreo, resultando en la puesta a cero del registro. |
| NC | 0000 | 0001 | 0001 | La rotación con acarreo introdujo el valor del registro de acarreo en BL por izquierda. |

1. La operación anterior realiza una rotación a la izquierda con acarreo, multiplicando por dos el valor de BL. Luego de provocar un acarreo, la operación copia el valor del bit de acarreo en BL por izquierda.

Sumar usando el Indicador de Acarreo

1. La operación anterior realiza una rotación a la izquierda con acarreo, multiplicando por dos el valor de BL. Luego de provocar un acarreo, la operación copia el valor del bit de acarreo en BL por izquierda.
2. Siendo DL = 0, el resultado de la suma con acarreo incrementa en uno al haber acarreo y no se altera al no haberlo.
3. La razón de sumar 30h es que representa el número ‘0’, el cual es el primer código válido.

Bucles

1. El programa podría ser ejecutado de una sola vez mediante el comando del debug –G.

La conclusión que podemos sacar, es que al ejecutarlo con G, lo que hace el programa es volver al valor inicial.

Escribir un número binario

**MOV:** se utiliza para mover datos a los registros.  
**ADD:** realiza la suma aritmética de dos registros, almacenando el resultado en el primer registro dado.  
**ADC:** realiza la suma aritmética de dos registros y el valor del bit de acarreo, almacenando el resultado en el primer registro dado.  
**INT:** llama a una interrupción de software. Algunas de ellas son:  
**INT 20h:** termina el programa

* **INT 21h, AH=02:** imprime un carácter almacenado en el registro
* **INT 21h, AH=09:** imprime una cadena de caracteres almacenados en una dirección de memoria a la cual apunta el registro DX. Esta cadena debe finalizar con el signo ‘$’

**SHL:** realiza un desplazamiento lógico a la izquierda.  
**SHR:** realiza un desplazamiento lógico a la derecha.  
**SAL:** realiza un desplazamiento aritmético de un bit a la izquierda.  
**SAR:** realiza un desplazamiento aritmético de un bit a la derecha.  
**ROL:** rota un bit hacia la izquierda sin acarreo.  
**ROR:** rota un bit hacia la derecha sin acarreo.  
**RCL:** rota un bit hacia la izquierda con acarreo.



-r bx

BX 0000

:aa

-a 100

1389:0100 mov ah, 02 → Mueve el 02 al registro alto de AX

1389:0102 mov cx, 08 → Mueve el 08 al registro CX

1389:0105 mov dl, 00 → Mueve el 00 al registro bajo de DX

1389:0107 rcl bl, 1 → Rota con acarreo el valor 1 a la izquierda una sola vez.

1389:0109 adc dl, 30 → A DL se suma 30 junto con el acarreo.

1389:010C int 21 → imprime el acarreo de la operación anterior.

1389:010E loop 0105 → es un bucle que vuelve a la instrucción 105, 8 veces.

1389:0110 int 20 → finaliza el programa.

1389:0112

-t

AX=0200 BX=00AA CX=0000 DX=0000 SP=FFEE BP=0000 SI=0000 DI=0000

DS=1389 ES=1389 SS=1389 CS=1389 IP=0102 NV UP EI PL NZ NA PO NC

1389:0102 B90800 MOV CX,0008

-t

AX=0200 BX=00AA CX=0008 DX=0000 SP=FFEE BP=0000 SI=0000 DI=0000

DS=1389 ES=1389 SS=1389 CS=1389 IP=0105 NV UP EI PL NZ NA PO NC

1389:0105 B200 MOV DL,00

-t

AX=0200 BX=00AA CX=0008 DX=0000 SP=FFEE BP=0000 SI=0000 DI=0000

DS=1389 ES=1389 SS=1389 CS=1389 IP=0107 NV UP EI PL NZ NA PO NC

1389:0107 D0D3 RCL BL,1

-t

AX=0200 BX=0054 CX=0008 DX=0000 SP=FFEE BP=0000 SI=0000 DI=0000

DS=1389 ES=1389 SS=1389 CS=1389 IP=0109 OV UP EI PL NZ NA PO CY

1389:0109 80D230 ADC DL,30

-t

AX=0200 BX=0054 CX=0008 DX=0031 SP=FFEE BP=0000 SI=0000 DI=0000

DS=1389 ES=1389 SS=1389 CS=1389 IP=010C NV UP EI PL NZ NA PO NC

1389:010C CD21 INT 21

-p

1

AX=0231 BX=0054 CX=0008 DX=0031 SP=FFEE BP=0000 SI=0000 DI=0000

DS=139C ES=139C SS=139C CS=139C IP=010E NV UP EI PL NZ NA PO NC

139C:010E E2F5 LOOP 0105

[…]

-g

0

Program terminated normally

1. La modificación que debemos hacer es agregar el segmento de programa en la dirección 0110

**1389:0110 mov dl, 62**

**1389:0112 int 21**

-r bx

BX 00AA

:00aa

-a 100

1389:0100 mov ah,02

1389:0102 mov cx, 08

1389:0105 mov dl, 00

1389:0107 mov bl, 1

1389:0109 adc dl, 30

1389:010C int 21

1389:010E loop 0105

*1389:0110 mov dl, 62*

*1389:0112 int 21*

1389:0114 int 20

1389:0116

-g

00000000b

Program terminated normally

Comparación y bits de estado.

-a 100  
0CB6:0100 mov AX,0003  
0CB6:0103 sub AX,0001  
0CB6:0106 JNZ 0103  
0CB6:0108 int 20  
0CB6:010A  
-t

AX=0003 BX=0000 CX=0000 DX=0000 SP=FFEE BP=0000 SI=0000 DI=0000  
DS=0CB6 ES=0CB6 SS=0CB6 CS=0CB6 IP=0103 NV UP EI PL NZ NA PO NC  
0CB6:0103 2D0100 SUB AX,0001  
-t

AX=0002 BX=0000 CX=0000 DX=0000 SP=FFEE BP=0000 SI=0000 DI=0000  
DS=0CB6 ES=0CB6 SS=0CB6 CS=0CB6 IP=0106 NV UP EI PL NZ NA PO NC  
0CB6:0106 75FB JNZ 0103

-t

AX=0002 BX=0000 CX=0000 DX=0000 SP=FFEE BP=0000 SI=0000 DI=0000  
DS=0CB6 ES=0CB6 SS=0CB6 CS=0CB6 IP=0103 NV UP EI PL NZ NA PO NC  
0CB6:0103 2D0100 SUB AX,0001  
-t

AX=0001 BX=0000 CX=0000 DX=0000 SP=FFEE BP=0000 SI=0000 DI=0000  
DS=0CB6 ES=0CB6 SS=0CB6 CS=0CB6 IP=0106 NV UP EI PL NZ NA PO NC  
0CB6:0106 75FB JNZ 0103  
-t

AX=0001 BX=0000 CX=0000 DX=0000 SP=FFEE BP=0000 SI=0000 DI=0000  
DS=0CB6 ES=0CB6 SS=0CB6 CS=0CB6 IP=0103 NV UP EI PL NZ NA PO NC  
0CB6:0103 2D0100 SUB AX,0001  
--t

AX=0000 BX=0000 CX=0000 DX=0000 SP=FFEE BP=0000 SI=0000 DI=0000  
DS=0CB6 ES=0CB6 SS=0CB6 CS=0CB6 IP=0106 NV UP EI PL ZR NA PE NC  
0CB6:0106 75FB JNZ 0103

-t

AX=0000 BX=0000 CX=0000 DX=0000 SP=FFEE BP=0000 SI=0000 DI=0000  
DS=0CB6 ES=0CB6 SS=0CB6 CS=0CB6 IP=0108 NV UP EI PL ZR NA PE NC  
0CB6:0108 CD20 INT 20  
-p

El programa ha terminado de forma normal



|  |  |  |
| --- | --- | --- |
| IP | AX | Ind. Cero |
| 0103 | 0003 | NZ |
| 0106 | 0002 | NZ |
| 0103 | 0002 | NZ |
| 0106 | 0001 | NZ |
| 0103 | 0001 | NZ |
| 0106 | 0000 | ZR |
| 0108 | 0000 | ZR |

-a 100

1389:0100 mov ax, 00f5

1389:0103 mov bx, 00f5

1389:0106 cmp ax, bx

1389:0108 int 20

1389:010A

-t

AX=00F5 BX=00AA CX=0000 DX=0000 SP=FFEE BP=0000 SI=0000 DI=0000

DS=1389 ES=1389 SS=1389 CS=1389 IP=0103 NV UP EI PL NZ NA PO NC

1389:0103 BBF500 MOV BX,00F5

-t

AX=00F5 BX=00F5 CX=0000 DX=0000 SP=FFEE BP=0000 SI=0000 DI=0000

DS=1389 ES=1389 SS=1389 CS=1389 IP=0106 NV UP EI PL NZ NA PO NC

1389:0106 39D8 CMP AX,BX

-t

AX=00F5 BX=00F5 CX=0000 DX=0000 SP=FFEE BP=0000 SI=0000 DI=0000

DS=1389 ES=1389 SS=1389 CS=1389 IP=0108 NV UP EI PL ZR NA PE NC

1389:0108 CD20 INT 20

1. El indicador de ceros se activó.
2. Los registros AX y BX contienen los valores ingresados: 00F5

-a 100

1389:0100 mov al, ba

1389:0102 not al

1389:0104 mov ah, fc

1389:0106 or al, ah

1389:0108 mov bl, 14

1389:010A and bl, 19

1389:010D xor al, bl

1389:010F int 20

1389:0111

-t

AX=00BA BX=0000 CX=0000 DX=0000 SP=FFEE BP=0000 SI=0000 DI=0000

DS=1389 ES=1389 SS=1389 CS=1389 IP=0102 NV UP EI PL NZ NA PO NC

1389:0102 F6D0 NOT AL

-t

AX=0045 BX=0000 CX=0000 DX=0000 SP=FFEE BP=0000 SI=0000 DI=0000

DS=1389 ES=1389 SS=1389 CS=1389 IP=0104 NV UP EI PL NZ NA PO NC

1389:0104 B4FC MOV AH,FC

-t

AX=FC45 BX=0000 CX=0000 DX=0000 SP=FFEE BP=0000 SI=0000 DI=0000

DS=1389 ES=1389 SS=1389 CS=1389 IP=0106 NV UP EI PL NZ NA PO NC

1389:0106 08E0 OR AL,AH

-t

AX=FCFD BX=0000 CX=0000 DX=0000 SP=FFEE BP=0000 SI=0000 DI=0000

DS=1389 ES=1389 SS=1389 CS=1389 IP=0108 NV UP EI NG NZ NA PO NC

1389:0108 B314 MOV BL,14

-t

AX=FCFD BX=0014 CX=0000 DX=0000 SP=FFEE BP=0000 SI=0000 DI=0000

DS=1389 ES=1389 SS=1389 CS=1389 IP=010A NV UP EI NG NZ NA PO NC

1389:010A 80E319 AND BL,19

-t

AX=FCFD BX=0010 CX=0000 DX=0000 SP=FFEE BP=0000 SI=0000 DI=0000

DS=1389 ES=1389 SS=1389 CS=1389 IP=010D NV UP EI PL NZ NA PO NC

1389:010D 30D8 XOR AL,BL

-t

AX=FCED BX=0010 CX=0000 DX=0000 SP=FFEE BP=0000 SI=0000 DI=0000

DS=1389 ES=1389 SS=1389 CS=1389 IP=010F NV UP EI NG NZ NA PE NC

1389:010F CD20 INT 20

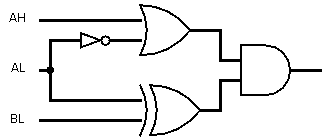
-g

Program terminated normally



|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **AL Hexa** | **AL Binario** | **AH Hexa** | **AH Binario** | **BL Hexa** | **BL Binario** |
| 00BA | 1011 1010 | 0000 | 0000 0000 | 0000 | 0000 0000 |
| 0045 | 0100 0101 | 0000 | 0000 0000 | 0000 | 0000 0000 |
| 0045 | 0100 0101 | 00FC | 1111 1100 | 0000 | 0000 0000 |
| 00FD | 1111 1101 | 00FC | 1111 1100 | 0000 | 0000 0000 |
| 00FD | 1111 1101 | 00FC | 1111 1100 | 0014 | 0001 0100 |
| 00FD | 1111 1101 | 00FC | 1111 1100 | 0010 | 0001 0000 |
| 00ED | 1110 1101 | 00fc | 1111 1100 | 0010 | 0001 0000 |



****



-a 100

00A7:0100 mov al, 01

00A7:0102 mov bl, 02

00A7:0104 oral, bl

00A7:0106 not al

00A7:0108 int 20

00A7:010A

-r ip

IP 0100

:100

-p

AX=0001 BX=0000 CX=0000 DX=0000 SP=FFEE BP=0000 SI=0000 DI=0000

DS=1389 ES=1389 SS=1389 CS=1389 IP=0102 NV UP EI PL NZ NA PO NC

1389:0102 B302 MOV BL,02

-p

AX=0001 BX=0002 CX=0000 DX=0000 SP=FFEE BP=0000 SI=0000 DI=0000

DS=1389 ES=1389 SS=1389 CS=1389 IP=0104 NV UP EI PL NZ NA PO NC

1389:0104 08D8 OR AL,BL

-p

AX=0003 BX=0002 CX=0000 DX=0000 SP=FFEE BP=0000 SI=0000 DI=0000

DS=1389 ES=1389 SS=1389 CS=1389 IP=0106 NV UP EI PL NZ NA PE NC

1389:0106 F6D0 NOT AL

-p

AX=0003 BX=0002 CX=0000 DX=0000 SP=FFEE BP=0000 SI=0000 DI=0000

DS=1389 ES=1389 SS=1389 CS=1389 IP=0106 NV UP EI PL NZ NA PE NC

1389:0106 F6D0 NOT AL

-p

AX=00FC BX=0002 CX=0000 DX=0000 SP=FFEE BP=0000 SI=0000 DI=0000

DS=1389 ES=1389 SS=1389 CS=1389 IP=0108 NV UP EI PL NZ NA PE NC

1389:0108 CD20 INT 20

-p

Program terminated normally

-r ax

AX 00FC

:0000

-r ax

AX 00FC

:0000

-r bx

BX 00FD

:0000

-a 100

1389:0100 mov al, 01

1389:0102 not al

1389:0104 mov bl, 02

1389:0106 not bl

1389:0108 and al, bl

1389:010A int 20

1389:010C

-r ip

IP 010A

:100

-p

AX=0001 BX=0000 CX=0000 DX=0000 SP=FFEE BP=0000 SI=0000 DI=0000

DS=1389 ES=1389 SS=1389 CS=1389 IP=0102 NV UP EI NG NZ NA PE NC

1389:0102 F6D0 NOT AL

-p

AX=00FE BX=0000 CX=0000 DX=0000 SP=FFEE BP=0000 SI=0000 DI=0000

DS=1389 ES=1389 SS=1389 CS=1389 IP=0104 NV UP EI NG NZ NA PE NC

1389:0104 B302 MOV BL,02

-p

AX=00FE BX=0002 CX=0000 DX=0000 SP=FFEE BP=0000 SI=0000 DI=0000

DS=1389 ES=1389 SS=1389 CS=1389 IP=0106 NV UP EI NG NZ NA PE NC

1389:0106 F6D3 NOT BL

-p

AX=00FE BX=0002 CX=0000 DX=0000 SP=FFEE BP=0000 SI=0000 DI=0000

DS=1389 ES=1389 SS=1389 CS=1389 IP=0106 NV UP EI NG NZ NA PE NC

1389:0106 F6D3 NOT BL

-p

AX=00FE BX=00FD CX=0000 DX=0000 SP=FFEE BP=0000 SI=0000 DI=0000

DS=1389 ES=1389 SS=1389 CS=1389 IP=0108 NV UP EI NG NZ NA PE NC

1389:0108 20D8 AND AL,BL

-p

AX=00FC BX=00FD CX=0000 DX=0000 SP=FFEE BP=0000 SI=0000 DI=0000

DS=1389 ES=1389 SS=1389 CS=1389 IP=010A NV UP EI NG NZ NA PE NC

1389:010A CD20 INT 20

-p

Program terminated normally

AX=00FC. Se comprueba que: -(a and b) = -a or –b.

Ejercicios de desplazamientos y rotaciones



**6Ah =** 01101010

RCL 4 bits

NC 10100011

SHR 2 bits

00101000 CY

Comprobación en debug (punto 22)

-a 100

0D41:0100 mov cx, 4

0D41:0103 mov al, 6a

0D41:0105 rcl al, cl

0D41:0107 mov cx, 2

0D41:010A shr al, cl

0D41:010C ^C

-r ip

IP 0107

:100

-g 10c

AX=0028 BX=0000 CX=0002 DX=0000 SP=FFEE BP=0000 SI=0000 DI=0000

DS=0D41 ES=0D41 SS=0D41 CS=0D41 IP=010C NV UP EI PL NZ NA PE CY



**B5h =** 10110101

ROL 4 bits

1 01011011

SAR 3 bits

00001011 NC

Comprobación en debug (punto 22)

-a 100

0D41:0100 mov cx, 4

0D41:0103 mov ax, b5

0D41:0106 rol al, cl

0D41:0108 mov cx, 3

0D41:010B sar al, cl

0D41:010D ^C

-r ip

IP 010C

:100

-g 10d

AX=000B BX=0000 CX=0003 DX=0000 SP=FFEE BP=0000 SI=0000 DI=0000

DS=0D41 ES=0D41 SS=0D41 CS=0D41 IP=010D NV UP EI PL NZ NA PO NC

0D41:010D 0D0A1D OR AX,1D0A



**C7h =** 11000111 CY

SAR 4 bits

11111100 NC

RCR 4bits

11000111 CY

Comprobación en debug (punto 22)

-a 100

0D41:0100 mov cx, 4

0D41:0103 mov ax, c7

0D41:0106 sar al, cl

0D41:0108 rcl al, cl

0D41:010A ^C

-r ip

IP 010D

:100

-g 10a

AX=00C7 BX=0000 CX=0004 DX=0000 SP=FFEE BP=0000 SI=0000 DI=0000

DS=0D41 ES=0D41 SS=0D41 CS=0D41 IP=010A NV UP EI NG NZ NA PE CY

0D41:010A 00D2 ADD DL,DL



**84h =** 10000100

SAR 2 bits

11100001 NC

RCL 4 bits

NC 00010111

Comprobación en debug (punto 22)

-a 100

0D41:0100 mov cx, 2

0D41:0103 mov al, 84

0D41:0105 sar al, cl

0D41:0107 mov cx, 4

0D41:010A rcl al, cl

-r ip

IP 010A

:100

-g 10c

AX=0017 BX=0000 CX=0004 DX=0000 SP=FFEE BP=0000 SI=0000 DI=0000

DS=0D41 ES=0D41 SS=0D41 CS=0D41 IP=010C NV UP EI NG NZ NA PE NC

0D41:010C F8 CLC



**87h =** 10000111 CY

SAR 2 bits

11100001 CY

SHR 4 bits

00001110 NC

RCL 5 bits

CY 11000001

Comprobación en debug (punto 22)

-a 100

0D41:0100 mov cx, 2

0D41:0103 mov al, 87

0D41:0105 sar al, cl

0D41:0107 mov cx, 4

0D41:010A rcl al, cl

0D41:010C mov cx, 5

0D41:010F rcl al, cl

0D41:0111 ^C

-r ip

IP 010C

:100

-g 0111

AX=00E1 BX=0000 CX=0005 DX=0000 SP=FFEE BP=0000 SI=0000 DI=0000

DS=0D41 ES=0D41 SS=0D41 CS=0D41 IP=0111 NV UP EI NG NZ NA PE CY

0D41:0111 7263 JB 0176

23.

A7h  1Dh NC

A7h 1Dh

10100111 00011101

RCL 4 bits NC 01110111

SHR 2 bits 00011101 NC

Comprobación en debug

-a 100

0D41:0100 B90400 MOV CX,4

0D41:0103 B0A7 MOV AL,A7

0D41:0105 D2D0 RCL AL,CL

0D41:0107 B90200 MOV CX,2

0D41:010A D2E8 SHR AL,CL

-r ip

IP 010C

:100

-rf

NV UP EI PL NZ NA PE NC -nc

-g 10c

AX=001D BX=0000 CX=0002 DX=0000 SP=FFEE BP=0000 SI=0000 DI=0000

DS=0D41 ES=0D41 SS=0D41 CS=0D41 IP=010C NV UP EI PL NZ NA PE NC

0D41:010C B90500 MOV CX,0005

24.

A7h  3Eh CY

A7h = 10100111

3Eh = 00111110

SHL 2 bits 11101000 NC

SAR 2 bits 11111010 NC

SHR 2 bits 00111110 CY

Comprobación en debug

-a 100

0D41:0100 mov cx, 2

0D41:0103 mov ax, 3a

0D41:0106 shl al, cl

0D41:0108 sar al, cl

0D41:010A shr al, cl

0D41:010C ^C

-r ip

IP 010C

:100

-rf

NV UP EI PL NZ NA PE NC -cy

-g 10c

AX=003E BX=0000 CX=0002 DX=0000 SP=FFEE BP=0000 SI=0000 DI=0000

DS=0D41 ES=0D41 SS=0D41 CS=0D41 IP=010C NV UP EI PL NZ NA PO CY

0D41:010C B90500 MOV CX,0005

COMPLEMENTO

25)

-a 100

00A7:0100 mov ah, 01

00A7:0102 int 21

00A7:0104 mov dl, al

00A7:0106 sub dl, 30

00A7:0109 cmp dl, 09

00A7:010C jle 0111

00A7:010E sub dl, 07

00A7:0111 mov cl, 04

00A7:0113 shl dl, cl

00A7:0115 int 21

00A7:0117 sub al, 30

00A7:0119 cmp al, 09

00A7:011B jle 011f

00A7:011D sub al, 07

00A7:011F add dl, al

00A7:0121 int 20

00A7:0123

-g121

**00**

AX=0100 BX=0000 CX=0004 DX=00**00**  SP=FFEE BP=0000 SI=0000 DI=0000

DS=1389 ES=1389 SS=1389 CS=1389 IP=0121 NV UP EI PL ZR NA PE NC

1389:0121 CD20 INT 20

-g121

**ff**

AX=012F BX=0000 CX=0004 DX=00**1F** SP=FFEE BP=0000 SI=0000 DI=0000

DS=1389 ES=1389 SS=1389 CS=1389 IP=0121 NV UP EI PL NZ NA PO CY

1389:0121 CD20 INT 20

El programa no funciona correctamente debido a que el segundo byte (de derecha a izquierda) del registro AL, queda un dato que luego interfiere con el resultado al ser sumado este registro con DL.

Si se realiza SHL y SHR de 4 bits al registro AL, se salva este inconveniente.

-a 100

1389:0100 mov ah, 01

1389:0102 int 21

1389:0104 mov dl, al

1389:0106 sub dl, 30

1389:0109 cmp dl, 09

1389:010C jle 0111

1389:010E sub dl, 07

1389:0111 mov cl, 04

1389:0113 shl dl, cl

1389:0115 int 21

1389:0117 sub al, 30

1389:0119 cmp al, 09

1389:011B jle 011f

1389:011D sub al, 07

1389:011F shl al, cl

1389:0121 shr al, cl

1389:0123 add al, dl

1389:0125 int 20

1389:0127

-g 125

**00**

AX=0100 BX=0000 CX=0004 DX=00**00** SP=FFEE BP=0000 SI=0000 DI=0000

DS=1389 ES=1389 SS=1389 CS=1389 IP=0125 NV UP EI PL ZR NA PE NC

1389:0125 CD20 INT 20

-r ip

IP 0125

:100

-g 125

**ff**

AX=010F BX=0000 CX=0004 DX=00**FF** SP=FFEE BP=0000 SI=0000 DI=0000

DS=0D05 ES=0D05 SS=0D05 CS=0D1B IP=0125 NV UP EI NG NZ NA PE NC

0D05:0125 CD20 INT 20

26)

**MOV AH, 01:** mueve el valor 01 a la parte alta del registro A, pues 01 es la función de la interrupción 21 que permite leer un carácter del teclado, que será colocado en la parte baja del registro A en formato de código ASCII.

**INT 21:** queda almacenado el carácter leído en código ASCII en la parte baja del registro A.

**MOV DL, AL:** mueve el valor hexadecimal del carácter leído de AL a DL ya que se lo utiliza de registro de trabajo.

**SUB DL, 30:** resta a DL el valor 30h, para saber si es un digito y cuál es, ya que estos se encuentran entre 30h y 39h en el código ASCII.

**CMP DL, 30:** resta DL y 09, para establecer los indicadores pero no cambia los registros DL. Se verifica si es un digito entre 0 y 9.

**JLE 0111:** Salta a 0111 si es menor o igual a 09. Si ya se determinó que es un digito, pasa a leer otro carácter; sino continúa con la siguiente sentencia.

**SUB DL, 07:** resta 07h al registro DL para determinar el carácter leído.

**MOV CL, 04:** mueve el valor 04 a la parte baja del contador C, para realizar el desplazamiento sobre DL.

**SHL DL, CL:** desplaza lógicamente a la izquierda 4 dígitos en DL, agrega un 0 en el ultimo byte de este registro, para que quede en la primera posición del número hexadecimal que está siendo leído, y no escribir el segundo carácter sobre el primero.

**INT 21:** permite ingresar el segundo carácter desde el teclado, y lo almacena en la parte baja del registro A.

**SUB AL, 30:** sustrae del registro AL el valor 30h, para determinar si es un digito y cuál es, ya que estos se encuentran entre 30h y 39h en el código ASCII.

**CMP AL, 09:** compara si DL es menor o igual a 09, restando los valores pero sin modificar el registro DL. Se verifica si es un digito entre 0 y 9.

**JLE 011F:** Salta a 011f si es menor o igual a 09. Si ya se estableció que es un digito continua leyendo otro carácter; sino sigue con la instrucción posterior.

**SUB AL, 07:** resta 07 al registro DL para determinar el carácter leído.

**SHL AL, CL:** desplaza lógicamente a la izquierda 4 dígitos en AL, agrega un 0 en el ultimo byte de este registro.

**SHR AL, CL:** desplaza lógicamente a la derecha 4 dígitos en AL, agrega un 0 en el primer byte de este registro.

**ADD AL, DL:** suma los dígitos hexadecimales contenidos en los registros DL y AL, para colocarlos en la forma en que se ingreso el número.

**INT 20:** finaliza el programa.

27)

mov ah, 01  
int 21  
mov dl, al  
sub dl, 30  
cmp dl, 09  
jle 0111  
sub dl, 07  
mov cl, 04  
chl dl, cl  
int 21  
sub al, 30  
cmp al, 09  
jle 011f  
sub al, 07  
shl al, cl  
shr al, cl  
add al, dl  
mov ah, 02  
int 21  
int 20