UTN.BA UNIVERSIDAD TECNOLÓGICA NACIONAL FACULTAD REGIONAL BUENOS AIRES	Arquitectura de Computadores	Curso:	K1092
	UNIDAD 10	Assembler	

TRABAJO PRÁCTICO 10

Assembler

GRUPO Nª 5

ALUMNO	LEGAJO
BUONAMICO, Leandro Elian	**1776721**
di NÁPOLI, Franco	**2047901**
PERALTA, Roque Damián	**1447580**
RODRIGUEZ, Iván Ángel	**1782630**

PRACTICO DE PROGRAMACIÓN ASSEMBLER BAJO DEBUG

- *Ingrese en el programa debug de su plataforma XP y transcriba los comandos y código tal cual aparecen abajo*

```
-a 0100
0CD7:0100 mov ah,8 {Mueve al registro Ah el valor 8}
0CD7:0102 add ah,3   {Al contenido del registro Ah (8) le suma el valor 3}
0CD7:0105 sub ah,4   {Al contenido del registro Ah (11) le resta 4}
0CD7:0108
-
-u 100 10a  {Muestra el programa entre los desplazamientos 100 y 10a}
0CD7:0100 B408          MOV     AH,08
0CD7:0102 80C403        ADD     AH,03
0CD7:0105 80EC04        SUB     AH,04
```

-r (muestra el valor de los registros antes de ejecutarse la primera instrucción)

```
AX=0000  BX=0000  CX=0000  DX=0000  SP=FFEE  BP=0000  SI=0000  DI=0000
DS=0CD7  ES=0CD7  SS=0CD7  CS=0CD7  IP=0100   NV UP EI PL NZ NA PO NC
0CD7:0100 B408        MOV AH,08
```

-t {Ejecuta un intruccion por vez (single step) y proporciona los contenidos de los registros y las banderas}

```
AX=0800  BX=0000  CX=0000  DX=0000  SP=FFEE  BP=0000  SI=0000  DI=0000
DS=0CD7  ES=0CD7  SS=0CD7  CS=0CD7  IP=0102   NV UP EI PL NZ NA PO NC
0CD7:0102 80C403        ADD     AH,03
-t

AX=0B00  BX=0000  CX=0000  DX=0000  SP=FFEE  BP=0000  SI=0000  DI=0000
DS=0CD7  ES=0CD7  SS=0CD7  CS=0CD7  IP=0105   NV UP EI PL NZ NA PO NC
0CD7:0105 80EC04        SUB     AH,04
-t

AX=0700  BX=0000  CX=0000  DX=0000  SP=FFEE  BP=0000  SI=0000  DI=0000
DS=0CD7  ES=0CD7  SS=0CD7  CS=0CD7  IP=0108   NV UP EI PL NZ NA PO NC
0CD7:0108 7415          JZ      011F
```

(En todos los casos AX= XX00 indica que en los 8 bits mas significativos se guardo un dato).

Las banderas mas importantes son:
NV = No hay desbordamiento
PL =El resultado es positivo
NZ = No es Cero

NC = No hay acarreo (lo cual indica indirectamente que no supero el limite, ya que es una suma de numeros positivos)

- *Indique el modo de direccionamiento de las tres instrucciones.*
- *El comando U de unassembler permite visualizar el código de maquina de las instrucciones ingresadas en –a (assembly) indique para cada instrucción los correspondientes códigos de maquina y para cada uno, el código de operación y el campo de referencia al DATA.*
- *Indique a que se denomina base y a que desplazamiento en una dirección segmentada y*
- *cuales son los valores que corresponden a la dirección de la tercera instrucción.*
- *El comando r de register permite visualizar e incluso modificar el valor de un registro, en el caso de este ejemplo con que valor se carga y para que.*
- *Indique la función de cada uno de los registros del procesador que visualiza debajo del comando –t.*
- *Exprese verbalmente a que banderas se hace referencia a continuación del registro IP.*
- *En la tercera línea aparece una dirección, un código de maquina y una instrucción assembler indique a que hace referencia cada uno de ellos en el primer trace.*

Practico 2: Suma y resta de constantes con resultado negativo, utilizando la parte alta (AH) del registro AX

Practico 2:

- *Ingrese nuevamente el programa con valores diferentes 8(10)+3(10)-12(10)*

-a 0100
0CD7:0100 mov ah,8
0CD7:0102 add ah,3
0CD7:0105 sub ah,c
0CD7:0108

Como este es su segundo programa el puntero de instrucción debe reiniciarse en el valor 0100 para ello se utiliza el mismo comando –r de la siguiente manera (al lado de 0108 ingrese 0100 y luego <enter>)

-r IP
IP 0108 0100 <enter>

Si quiere ver lo modificado, entonces hágalo nuevamente pero de un enter al lado del 0100

-r IP

IP 0100 <enter>

(por cada programa que haga debe reasignar el valor 0100 al IP)

Ahora comience la ejecución single step como en el ejercicio anterior

-r (muestra el valor de los registros antes de ejecutarse la primera instrucción)

AX=0000 BX=0000 CX=0000 DX=0000 SP=FFEE BP=0000 SI=0000 DI=0000
DS=0CD7 ES=0CD7 SS=0CD7 CS=0CD7 IP=0100 NV UP EI PL NZ NA PO NC
0CD7:0100 B408 MOV AH,08

-t

AX=0800 BX=0000 CX=0000 DX=0000 SP=FFEE BP=0000 SI=0000 DI=0000
DS=0CD7 ES=0CD7 SS=0CD7 CS=0CD7 IP=0102 NV UP EI PL NZ NA PO NC
0CD7:0102 80C403 ADD AH,03

-t

AX=0B00 BX=0000 CX=0000 DX=0000 SP=FFEE BP=0000 SI=0000 DI=0000
DS=0CD7 ES=0CD7 SS=0CD7 CS=0CD7 IP=0105 NV UP EI PL NZ NA PO NC
0CD7:0105 80EC0C SUB AH,0C

-t

AX=FF00 BX=0000 CX=0000 DX=0000 SP=FFEE BP=0000 SI=0000 DI=0000
DS=0CD7 ES=0CD7 SS=0CD7 CS=0CD7 IP=0108 NV UP EI NG NZ AC PE CY
0CD7:0108 7415 JZ 011F

Luego de la ejecución de esta ultima instrucción el registro AX=FF00 (indicado en Hexadecimal) nos muestra que el valor del AH=FF, para poder interpretarlo correctamente es necesario prestar atención a las banderas de estado que, para la ultima instrucción son:

NV= no hubo desbordamiento
NG= El resultado es negativo
NZ= No es Cero
CY= Hubo Acarreo

De donde el valor FF representa un resultado negativo, que acorde con el estado de la bandera de overflow no sobrepasa el limite de 8 bits.

- *Represente el valor de AH en binario y en el formato de 8 bits indicando quien representa el signo y quien la magnitud.*
  - *11111111*
  - *Siendo el bit marcado en rojo el más significativo, y el resto la magnitud*

- *Indique cual es el rango de valores signados permitidos para este registr AH.*
  - *(-128, 127) respondiendo a la formula (-2 ^ n - 1, 2 ^(n-1) –1)*

- *Compruebe que el resultado obtenido corresponde a -1.*
  - 1 1 1 1 1 1 1 1b = FF00h
  - 1 0 0 0 0 0 0 0
  - + 1
  - 1 0 0 0 0 0 0 1 b
  - - 1

- *Indique que banderas se modificaron entre la ejecución de la segunda instrucción y la de la primera.*
- PL => NG, NA => AC, PO => PE, NC => CY

- *Continue completando la tabla siguiente para que muestre la representación del mapa parcial de memoria donde se aloja el programa*

Instrucción	Base	Desplazamiento (inst)	Contenido (registro)
MOV AH,08	0CD7	0100	08
ADD AH, 3	0CD7	0102	0B
SUB AH, Ch	0CD7	0105	FF

La siguiente secuencia de instrucciones no sigue ninguna lógica, simplemente carguelas, haga el trace y represente el mapa parcial de memoria.

Reinicie el valor de IP e ingrese:

```
-a 0100
0CD7:0100 mov ax,8
0CD7:0103 mov [0200],ax
0CD7:0106 mov ax,3
0CD7:0109 mov [0203],ax
0CD7:010C mov ax,[0200]
0CD7:010F add ax,[0203]
0CD7:0113 mov [0205],ax
0CD7:0116
```

Reinicie el valor de IP a 0100 antes de ejecutar el trace

-t {ejecucion del MOV AX,08 / se carga la constante 8 en el registro AX}

```
AX=0008  BX=0000  CX=0000  DX=0000  SP=FFEE  BP=0000  SI=0000  DI=0000
DS=0CD7  ES=0CD7  SS=0CD7  CS=0CD7  IP=0103   NV UP EI PL NZ NA PO NC
0CD7:0103 A30002        MOV     [0200],AX                       DS:0200=0008
```

- *Que diferencia existe entre la ejecución de la instrucción MOV AH, 08 de los ejercicios anteriores y esta instrucción en cuanto a la modificación del registro AX? y en cuanto al código de operación? ¿Puede ser que el cambio de registro afecte el valor del código de operación? ¿Por qué?*
    - *En los ejercicios anteriores se guardaba 0800 en AX mientras que ejecutando la instrucción actual se guarda 0008 en AX.*
    - *El cambio de registro afecta al COP, debido a que cada instrucción tiene un COP distinto.*

- *La tercera línea de este trace detalla la próxima instrucción a ejecutarse indique que interpreta como DS:0200 0008*
    - *Es el valor de guardado en memoria de la posición 0200. En este caso se le asigna por referencia el valor 0008.*

Siguiente single step

-t {ejecución del MOV [0200], AX}

```
AX=0008  BX=0000  CX=0000  DX=0000  SP=FFEE  BP=0000  SI=0000  DI=0000
DS=0CD7  ES=0CD7  SS=0CD7  CS=0CD7  IP=0106   NV UP EI PL NZ NA PO NC
0CD7:0106 B80300        MOV     AX,0003
```

Esta instruccion escribe la direccion 0200 y 0201 con el valor del registro AX usaremos el un comando de debug para ver si efectivamente lo hizo, compruebelo

-E 0200

-E 0201

- *Que valores observa en cada desplazamiento y porque carga dos bytes?*
  - *En la posición 0200 se encuentra almacenado el valor 0008 debido que fue asignado en la segunda instrucción.*
  - *En la posición 0201 se encuentra almacenado el valor 0000 debido a que todavía no fue asignado ningún valor.*

Siguiente

-t {ejecucion del MOV AX,03}

AX=0003 BX=0000 CX=0000 DX=0000 SP=FFEE BP=0000 SI=0000 DI=0000
DS=0CD7 ES=0CD7 SS=0CD7 CS=0CD7 IP=0109 NV UP EI PL NZ NA PO NC
0CD7:0109 A30302 MOV [0203],AX DS:0203=0003

- *Que valor se borro de AX y cual es su nuevo contenido ?*
  - *Se borró el valor de 0008 y se le reasignó el valor 0003*
- *Cual era el valor del codigo de operacion de esta instruccion y cual era el valor de su campo data ?*
  - COP = 0109 DATA = A30302

Siguiente
-t

AX=0003 BX=0000 CX=0000 DX=0000 SP=FFEE BP=0000 SI=0000 DI=0000
DS=0CD7 ES=0CD7 SS=0CD7 CS=0CD7 IP=010C NV UP EI PL NZ NA PO NC
0CD7:010C A10002 MOV AX,[0200] DS:0200=0008

Esta ejecución correspondio al MOV [0203], AX según se indica en la ultima del trace anterior…..
0CD7:0109 A30302 MOV [0203],AX DS:0203=0003

- *Que bytes de memoria se modificaron y con que valor?*
  - *A la posición de memoria 0203 se le asignó el valor de AX el cual es 0003.*

- *Actualmente el registro AX contiene el valor 0003, cual será su valor luego de la ejecución de la próxima? Si ya lo respondió ahora ejecútela y verifique.*
  - *El valor del registro AX será 0008*

-t

AX=0008 BX=0000 CX=0000 DX=0000 SP=FFEE BP=0000 SI=0000 DI=0000
DS=0CD7 ES=0CD7 SS=0CD7 CS=0CD7 IP=010F NV UP EI PL NZ NA PO NC
0CD7:010F 03060302 ADD AX,[0203] DS:0203=0003

- *Si observa detenidamente este vuelco de registros cual sera el valor de AX luego de la ejecucion de la proxima instruccion ? Verifique con la siguiente ejecucion*
    - *El valor de AX va a ser:*
        - *0008 + 0003 = 000B*

-t

AX=000B BX=0000 CX=0000 DX=0000 SP=FFEE BP=0000 SI=0000 DI=0000
DS=0CD7 ES=0CD7 SS=0CD7 CS=0CD7 IP=0113 NV UP EI PL NZ NA PO NC
0CD7:0113 A30502 MOV [0205],AX DS:0205=000B

- *Que valor se cargara en 0205 y que valor en 0206? ejecute y verifique con el comando* -E
    - *En la 0205 se cargará el valor 000B*
    - *En la 0206 el valor 0000*

-t

AX=000B BX=0000 CX=0000 DX=0000 SP=FFEE BP=0000 SI=0000 DI=0000
DS=0CD7 ES=0CD7 SS=0CD7 CS=0CD7 IP=0116 NV UP EI PL NZ NA PO NC

- *Complete los siguientes contenidos*

Instrucción		Destino	Contenido
MOV AX,0008	0CD7	Registro AH	00
	0CD7	Registro AL	08
MOV [0200],AX	0CD7:	0201	00
	0CD7:	0200	08
MOV AX,0003	0CD7	Registro AH	00
	0CD7	Registro AL	03
MOV [0203],AX	0CD7:	0204	00
	0CD7:	0203	03
MOV AX,[0200]	0CD7	Registro AH	00
	0CD7	Registro AL	08
ADD AX,[0203]	0CD7	Registro AH	00
	0CD7	Registro AL	0B
MOV [0205],AX	0CD7:	0206	00
	0CD7:	0205	0B

Practico 3: Multiplicación de dos entidades enteras de 16 bits, con y sin signo

- *Ejecute el siguiente programa que realiza la multiplicación de dos entidades sin signo y que contienen el máximo valor permisible en 16 bits (65.535). El resultado deberá ser 4.294.836.225 en decimal, equivalente a FFFE0001 Hexadecimal, examine el contenido del par DX:AX para verificar que la instrucción deja el valor del resultado final en ellos.*

-r IP {reinicia el valor del puntero}
IP XXX 0100

-a 0100
0CD7:0100 mov ax , FFFF
0CD7:0103 mov cx , FFFF
0CD7:0106 mul , cx
0CD7:0108
-t

AX=FFFF BX=0000 CX=0000 DX=0000 SP=FFEE BP=0000 SI=0000 DI=0000
DS=0CD7 ES=0CD7 SS=0CD7 CS=0CD7 IP=0103   NV UP EI PL NZ NA PO NC
0CD7:0103 B9FFFF   MOV   CX,FFFF
-t

AX=FFFF BX=0000 CX=FFFF DX=0000 SP=FFEE BP=0000 SI=0000 DI=0000
DS=0CD7 ES=0CD7 SS=0CD7 CS=0CD7 IP=0106   NV UP EI PL NZ NA PO NC
0CD7:0106 F7E1   MUL   CX
-t

AX=0001 BX=0000 CX=FFFF DX=FFFE SP=FFEE BP=0000 SI=0000 DI=0000
DS=0CD7 ES=0CD7 SS=0CD7 CS=0CD7 IP=0108   OV UP EI PL NZ NA PO CY
0CD7:0108 7415   JZ   011F

- *Desarrolle el programa que permita la multiplicación de los siguientes valores de 16 bits con signo: 4* -1, esta vez deberá utilizar la instrucción IMUL Verifique el contenido del par DX:AX*

  MOV AX, -1
  MOV CX, 4
  IMUL CL

Reinicie IP e ingrese:

-a 0100
0CD7:0100 mov ax, 4
0CD7:0103 mov cx,-1
0CD7:0106 imul cl
0CD7:0108
-t

AX=0004 BX=0000 CX=0000 DX=0000 SP=FFEE BP=0000 SI=0000 DI=0000
DS=0CD7 ES=0CD7 SS=0CD7 CS=0CD7 IP=0103   NV UP EI PL NZ NA PO NC
0CD7:0103 B9FFFF   MOV   CX,FFFF
-t

AX=0004 BX=0000 CX=FFFF DX=0000 SP=FFEE BP=0000 SI=0000 DI=0000
DS=0CD7 ES=0CD7 SS=0CD7 CS=0CD7 IP=0106   NV UP EI PL NZ NA PO NC

```
0CD7:0106 F6E9          IMUL    CL
-t

AX=FFFC  BX=0000  CX=FFFF  DX=0000  SP=FFEE  BP=0000  SI=0000  DI=0000
DS=0CD7  ES=0CD7  SS=0CD7  CS=0CD7  IP=0108   NV UP EI NG NZ NA PE NC
0CD7:0108 7415          JZ      011F
```

Nuevamente el resultado se encuentra en el par DX:AX donde al ser DX=0000 y AX=FFFC, vemos que el resultado se encuentra en AX (por ser DX=0000 irrelevante) como estamos usando registros de 16 bits es y el flag NG nos esta indicando que es negativo, para obtener el resultado es necesario el complemento autentico de FFFC, El cual es -4 en decimal

Practico 4: División de dos entidades enteras

- *Ejecute el programa de este ejemplo con trace y analice los registros AL y AH*

Reinicie IP y luego ingrese

```
-a  0100
0CD7:0100 mov ax,7
0CD7:0103 mov bl,3
0CD7:0105 div bl
0CD7:0107
-t

AX=0007  BX=0000  CX=0000  DX=0000  SP=FFEE  BP=0000  SI=0000  DI=0000
DS=0CD7  ES=0CD7  SS=0CD7  CS=0CD7  IP=0103   NV UP EI PL NZ NA PO NC
0CD7:0103 B303          MOV     BL,03
-t

AX=0007  BX=0003  CX=0000  DX=0000  SP=FFEE  BP=0000  SI=0000  DI=0000
DS=0CD7  ES=0CD7  SS=0CD7  CS=0CD7  IP=0105   NV UP EI PL NZ NA PO NC
0CD7:0105 F6F3          DIV     BL
-t

AX=0102  BX=0003  CX=0000  DX=0000  SP=FFEE  BP=0000  SI=0000  DI=0000
DS=0CD7  ES=0CD7  SS=0CD7  CS=0CD7  IP=0107   NV UP EI PL NZ NA PO NC
0CD7:0107 61            DB      61
```

- *Que sucedió con la operación?*
    1) *A la posición AH le asignó el resto (01) y a AL el cociente (02) resultando AX = 0102*

La respuesta es que en AH=01 Tengo el resto de la division (1 decimal) y en AL=02 tengo el Cociente (2 decimal)

- *Desarrolle un programa donde el dividendo sea igual a 1C y el divisor a -1 y analice los registros AL y AH*

```
mov ax, 01Ch
mov bl, -1
div bl
```

Reinicie IP e ingrese:

```
-a 0100

0CD7:0100 Mov ax,1C
0CD7:0103 mov bl,ff
0CD7:0105 idiv bl
0CD7:0107

-t

AX=001C  BX=0000  CX=0000  DX=0000  SP=FFEE  BP=0000  SI=0000  DI=0000
DS=0CD7  ES=0CD7  SS=0CD7  CS=0CD7  IP=0103   NV UP EI PL NZ NA PO NC
0CD7:0103 B3FF          MOV     BL,FF
-t

AX=001C  BX=00FF  CX=0000  DX=0000  SP=FFEE  BP=0000  SI=0000  DI=0000
DS=0CD7  ES=0CD7  SS=0CD7  CS=0CD7  IP=0105   NV UP EI PL NZ NA PO NC
0CD7:0105 F6FB          IDIV    BL
-t

AX=00E4  BX=00FF  CX=0000  DX=0000  SP=FFEE  BP=0000  SI=0000  DI=0000
DS=0CD7  ES=0CD7  SS=0CD7  CS=0CD7  IP=0107   NV UP EI PL NZ NA PO NC
0CD7:0107 61            DB      61
```

Observe que en AH=00 tenemos el resto de la division y en AL=E4 el cociente, recuerde que IDIV considera los valores signados

- *Cual es el valor decimal del dividendo, del divisor y del cociente?*
  - El dividendo es 28
  - El divison -1
  - El cociente es E4 = -28

Practico 5: Desplazamientos y acarreos

- *Desplace aritméticamente a derecha el valor A0 en el registro AL y analice el contenido de los registros involucrados*

Reinicie el valor de IP e ingrese

```
-a 0100

0CD7:0100 mov al,A0
0CD7:0102 shr al,1
0CD7:0104
-t

AX=00A0  BX=0000  CX=0000  DX=0000  SP=FFEE  BP=0000  SI=0000  DI=0000
DS=0CD7  ES=0CD7  SS=0CD7  CS=0CD7  IP=0102   NV UP EI PL NZ NA PO NC
0CD7:0102 D0E8          SHR     AL,1
-t

AX=0050  BX=0000  CX=0000  DX=0000  SP=FFEE  BP=0000  SI=0000  DI=0000
DS=0CD7  ES=0CD7  SS=0CD7  CS=0CD7  IP=0104   OV UP EI PL NZ NA PE NC
0CD7:0104 7269          JB      016F
-t
```

En este caso el registro AL=A0 (160 en decimal) fue desplazado aritmeticamente hacia la derecha por lo que el bit menos significativo fue guardado en el carry (es por eso el flag NC esta activo ya que este bit es justamente cero) y no modifico el signo (el bit mas significativo no fue modificado, es decir que la bandera sigue estando en PL por que el bit era cero) por ultimo el resultado fue equivalente a dividir A0 por la base dos

- *Verifiquelo.*
  AL = 50h = 80d
  A0 / 2 = 160 /2 = 50h = 80h

- *idem anterior pero utilice rotación a izquierda*
  AL = 41h = 65d

Reinicie el valor de IP e ingrese

```
-a 0100
0CD7:0100 mov al,a0
0CD7:0102 rol al,1
0CD7:0104
-t

AX=00A0  BX=0000  CX=0000  DX=0000  SP=FFEE  BP=0000  SI=0000  DI=0000
DS=0CD7  ES=0CD7  SS=0CD7  CS=0CD7  IP=0102   NV UP EI PL NZ NA PO NC
0CD7:0102 D0C0          ROL     AL,1
-t

AX=0041  BX=0000  CX=0000  DX=0000  SP=FFEE  BP=0000  SI=0000  DI=0000
DS=0CD7  ES=0CD7  SS=0CD7  CS=0CD7  IP=0104   OV UP EI PL NZ NA PO CY
0CD7:0104 7269          JB      016F
```

El AL=41 es el resultado de la rotacion a izquierda de AL=A0 por lo que el bit mas significativo de A0 fue colocado en la bandera de carry (por eso cambio a CY por que el bit era 1 en binario A (HEX) = 1010 (BIN)) y en el bit menos significativo.

- *Realice el ejercicio anterior, pero utilice rotación a derecha*
  AL = 50h = 80d

Reinice el valor de IP e ingrese

```
-a
0CD7:0100 mov al,A0
0CD7:0102 ror al,1
0CD7:0104
-t

AX=00A0  BX=0000  CX=0000  DX=0000  SP=FFEE  BP=0000  SI=0000  DI=0000
DS=0CD7  ES=0CD7  SS=0CD7  CS=0CD7  IP=0102   NV UP EI PL NZ NA PO NC
0CD7:0102 D0C8          ROR     AL,1
-t

AX=0050  BX=0000  CX=0000  DX=0000  SP=FFEE  BP=0000  SI=0000  DI=0000
DS=0CD7  ES=0CD7  SS=0CD7  CS=0CD7  IP=0104   OV UP EI PL NZ NA PO NC
0CD7:0104 7269          JB      016F
```

El AL=50 es el resultado de la rotación a derecha de AL=A0 con lo que el bit menos significativo fue colocado en la bandera de carry (por eso el carry no cambio, ya que el bit menos significativo era cero).

- *Realice el ejercicio anterior, pero utilice Rotación con acarreo a izquierda*
  AL = 40h = 64d

Reinicie el valor de IP e ingrese

```
-a 0100

0CD7:0100 mov al,A0
0CD7:0102 rcl al,1
0CD7:0104
-t

AX=00A0  BX=0000  CX=0000  DX=0000  SP=FFEE  BP=0000  SI=0000  DI=0000
DS=0CD7  ES=0CD7  SS=0CD7  CS=0CD7  IP=0102   NV UP EI PL NZ NA PO NC
0CD7:0102 D0D0          RCL     AL,1
-t

AX=0040  BX=0000  CX=0000  DX=0000  SP=FFEE  BP=0000  SI=0000  DI=0000
DS=0CD7  ES=0CD7  SS=0CD7  CS=0CD7  IP=0104   OV UP EI PL NZ NA PO CY
0CD7:0104 7269          JB      016F
```

El resultado de la rotación con acarreo a izquierda de AL=A0 es AL=40 donde se utilizo el carry para “ampliar” el registro AL, De esta forma el bit mas significativo (en este caso 1) se roto al carry (por eso paso de NC a CY) y el bit de carry (0) paso al menos significativo, desplazando una posición a la izquierda al resto de los bits

- Realice el ejercicio anterior, pero utilice Rotacion con acarreo a derecha
  AL = 50h = 80d

Reinicie el valor de IP e ingrese:

-a 0100

0CD7:0100 mov al,A0
0CD7:0102 rcr al,1
0CD7:0104
-t

AX=00A0 BX=0000 CX=0000 DX=0000 SP=FFEE BP=0000 SI=0000 DI=0000
DS=0CD7 ES=0CD7 SS=0CD7 CS=0CD7 IP=0102 NV UP EI PL Z NA PO NC
0CD7:0102 D0D8 RCR AL,1
-t

AX=0050 BX=0000 CX=0000 DX=0000 SP=FFEE BP=0000 SI=0000 DI=0000
DS=0CD7 ES=0CD7 SS=0CD7 CS=0CD7 IP=0104 OV UP EI PL NZ NA PO NC
0CD7:0104 7269 JB 016F

En este caso el resultado del AL=A0 se transformo en AL=50 con lo cual el bit menos significativo paso al carry (esto es el bit 0, por eso NC se mantiene) y el bit de carry paso al mas significativo, desplazando a los demas un lugar hacia la derecha.

Práctico 6: Bifurcaciones condicionales, incondicionales e iteraciones

Despliega una cadena 7 veces, utilizando un salto condicional y un registro que lleva la cuenta de cuantas veces fue desplegada. No corre en emu8086

Reinicie el valor de IP e ingrese:

-a 0100

0CD5:0100 jmp 130
0CD5:0102
-e 102 "que tengas un feliz dia!" 0d 0a "$"
-a 130
0CD5:0130 mov bx,7
0CD5:0133 mov dx, 102
0CD5:0136 mov ah, 9
0CD5:0138 int 21
0CD5:013A dec bx
0CD5:013B jnz 138
0CD5:013D int 20
0CD5:013F

-g {este comando ejecuta las instrucciones una tras la otra}

que tengas un feliz dia!
que tengas un feliz dia!
que tengas un feliz dia!
que tengas un feliz dia!
que tengas un feliz dia!
que tengas un feliz dia!
que tengas un feliz dia!

El programa ha terminado de forma normal

En este programa se incluye el mensaje en un area reservada para el código, por esa razón la primera instrucción es JMP 0100, para que cuando se ejecuta salte sobre la zona reservada para el dato y asi pueda ejecutar la primera instrucción del algoritmo.
Investigue que es una interrupción INT 21 y para que sirve la función 9, describa como se convoca a la interrupción y que registros del procesador usa como parámetros.

La instrucción INT 21 hace el llamado a una función del sistema de acuerdo al valor de AH, las funciones son:

AH	Description	AH	Description
01	Read character from STDIN	02	Write character to STDOUT
05	Write character to printer	06	Console Input/Output
07	Direct char read (STDIN), no echo	08	Char read from STDIN, no echo
09	Write string to STDOUT	0A	Buffered input
0B	Get STDIN status	0C	Flush buffer for STDIN
0D	Disk reset	0E	Select default drive
19	Get current default drive	25	Set interrupt vector
2A	Get system date	2B	Set system date
2C	Get system time	2D	Set system time
2E	Set verify flag	30	Get DOS version
35	Get Interrupt vector		
36	Get free disk space	39	Create subdirectory
3A	Remove subdirectory	3B	Set working directory
3C	Create file	3D	Open file
3E	Close file	3F	Read file
40	Write file	41	Delete file
42	Seek file	43	Get/Set file attributes
47	Get current directory	4C	Exit program
4D	Get return code	54	Get verify flag
56	Rename file	57	Get/Set file date

En este caso, la función 09 (Write string to STDOUT) escribe una cadena en la salida estándar que generalmente es la pantalla.

- *Indique cual es la función del registro BX y porque se lo decrementa?*

El registro BX tiene asignada la cantidad de iteraciones restantes, a medida que se van ejecutando el registro decrementa hasta llegar al 0;

- *Indique como ejecuta la instrucción jnz 138 cuando BX = 4 y como ejecuta cuando BX=0, en ambos caso indique que valor asume el registro IP.*

JNZ = Jump if not zero
En el caso donde BX = 4, la ejecución saltar al IP 138 donde se ejecuta la función 09.
En el caso donde BX = 0, la ejecución seguirá y ejecutará la siguiente línea.

- *Busque con que otro nombre se conoce a las instrucciones de salto incondicional y de salto condicional.*

Se las conoce también como instrucciones de transferencia de control especiales.

- *Indique si las isntrucciones de tipo INT se incluyen dentro de la clasificación de instrucciones de ruptura de seuencia.*

Si, ya que permiten alterar la secuencia normal de ejecución del programa.

- *Indique la diferencia de ejecución entre una instrucción JMP, JNZ, CALL, RET, INT, IRET, LOOP.*

  JMP: se utiliza para desviar el flujo de un programa sin tomar en cuenta las condiciones actuales de las banderas ni de los datos.

  JNZ: es un salto condicional, realiza el salto si no es cero.

  CALL: sirve para invocar un procedimiento o subrutina.

  RET: sirve para dar el retorno a un procedimiento o subrutina.

  INT: invoca una subrutina de interrupción.

  IRET: da el retorno a una subrutina de interrupción.

  LOOP: Transfieren el flujo del proceso, condicional o incondicionalmente, a un destino repitiendose esta acción hasta que el contador sea cero.

- *Indique cual es la función de la instrucción INT 20.*

La instrucción INT 20 ordena provocar una interrupción que es 20h, que le cede el control a DOS, y DOS termina el programa. Una interrupción es una interrupción de la ejecución de un programa para realizar una determinada tarea, a veces el control no vuelve al programa original (Por ejemplo, en el caso de INT 20h en el DOS).

- *El siguiente código despliega una cadena 7 veces, pero utiliza el registro cx para controlar la iteración y el salto condicional JCXZ, que salta si cx=0.*

Reinicie el valor de IP e ingrese:

```
-a 0100

0CD5:0100 jmp 130
0CD5:0102
-e 102 "que tengas un feliz dia" 0d 0a "$"
-a 130
0CD5:0130 mov dx, 102
0CD5:0133 mov cx,7
0CD5:0136 mov ah, 9
0CD5:0138 int 21
0CD5:013A dec cx
0CD5:013B jcxz 13f
0CD5:013D jmp 138
0CD5:013F int 20
0CD5:0141
-g
```

que tengas un feliz dia
que tengas un feliz dia
que tengas un feliz dia
que tengas un feliz dia
que tengas un feliz dia
que tengas un feliz dia
que tengas un feliz dia

El programa ha terminado de forma normal

Despliega la cadena 7 veces, reemplazando el salto condicional y el registro contador por la instrucción LOOP.

Reinicie el valor de IP e ingrese:

```
-a 0100
0CD5:0141 jmp 130
0CD5:0143
-e 102 "que tengas un feliz dia!" 0d 0a "$"
-a 130
0CD5:0130 mov cx,7
0CD5:0133 mov dx,102
0CD5:0136 mov ah,9
0CD5:0138 int 21
0CD5:013A loop 138
0CD5:013C int 20
0CD5:013E
-g
que tengas un feliz dia!
que tengas un feliz dia!
que tengas un feliz dia!
que tengas un feliz dia!
que tengas un feliz dia!
que tengas un feliz dia!
que tengas un feliz dia!

El programa ha terminado de forma normal
```

Practico 7: Concepto de pila; manejo de pila: push y pop

-e1100
0CE7:1100 69.d8
-e1101
0CE7:1101 76.d7

Reinicie el valor de IP e ingrese:

-a 0100

0CE7:0100 mov ax,d5d6
0CE7:0103 mov bx,d3d4
0CE7:0106 mov cx,d1d2
0CE7:0109 push [1100] (o push word [1100])
0CE7:010D push ax
0CE7:010E push bx
0CE7:010F push cx
0CE7:0111 pop ax
0CE7:0112 pop bx
0CE7:0113 pop cx
0CE7:0114 pop dx
0CE7:0115 int 20
0CE7:0117

Cuando realice el trace de este codigo las instrucciones PUSH almacenan en el area de memoria de pila los registros AX, BX, CX y los contenidos de la direcciones de memoria 0100 y 0101 (para demostrar que en el STACK se pueden almacenar tanto el contenido de un registro como el contenido de una posición de memoria). El único cambio que muestra su ejecución el el decremento en 2 unidades del SP.

- *Indique en la tabla como se va cargando la pila*

Sucesivas Instrucciones de Carga	Contenido	Actualizacion del SP
MOV AX,D5D6	D5D6	00FD
MOV BX,D3D4	D3D4	00FD
MOV CX,D1D2	D1D2	00FD
PUSH [1100]	D7D8	00FB
PUSH AX	D5D6	00F9
PUSH BX	D3D4	00F7
PUSH CX	D1D2	00F5

- *Indique en la tabla como se va descargando la pila*

Sucesivas Instrucciones de Carga	Contenido	Actualizacion del SP	Contenido del registro destino
	D5		
POP AX	D6	00F7	AX=D1D2
	D3	..	
POP BX	D4	00F9	BX=D3D4
	D1	..	
POP CX	D2	00FB	CX=D5D6
	00		
POP DX	00	00FD	DX=D7D8