J.T.P. Ing. Steiner Guillermo.
Email:gsteiner@scdt.frc.utn.edu.ar

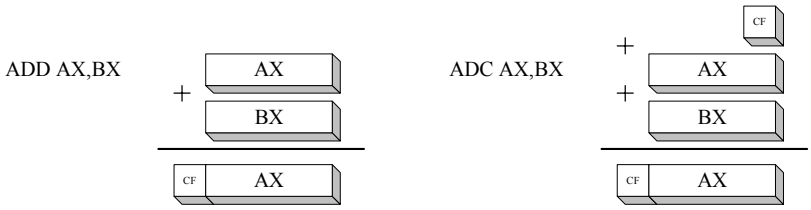
Trabajo Práctico Nro 3: Assembler

Banderas

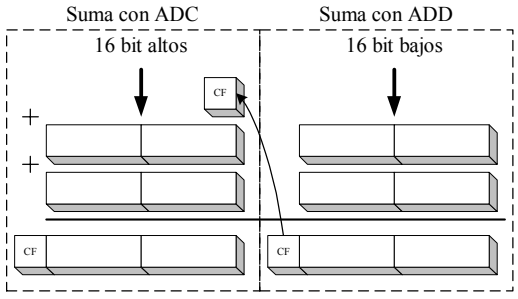
Nombre	**Estado = 0**	**Estado = 1**
Desbordamiento: Indica cuando el resultado de una operación con signo a excedido la capacidad del uP.	NV-No hubo desborde	OV-Desborde
Dirección: Controla la selección de incremento o decremento de los registros DI o SI en las instrucciones de cadenas.	UP-Incremento	DN-Decremento
Interrupción: habilita la entrada de interrupción.	DI-Interr. anulada	EI-Interrupción
Signo: indica el signo del resultado de la última operación aritmética.	PL-Positivo	NG-Negativo
Cero: indica si la última operación dio como resultado cero.	NZ-No Cero	ZR-Cero
Acarreo auxiliar: ocurre cuando en una operación aritmética, se produzco un acarreo o un préstamo entre el bit 3 y 4 del resultado.	NA-Sin Acarreo	AC-Acarreo
Paridad: es un conteo de unos en la última operación.	PO-Impar	PE-Par
Acarreo: indica un acarreo o un préstamo en la última operación aritmética, usado también en operaciones de corrimiento y como indicador de error en ciertos programas.	NC-Sin Acarreo	CY-Acarreo

Suma con acarreo

ADD: Esta instrucción suma el primer operador con el segundo y al resultado lo guarda en el primer operador, si hubo acarreo lo guarda en **Carry**.
ADC: Realiza la misma operación que ADD incorporando también el **Carry** en la suma de los operadores.



Para el caso de sumar números cuya longitud supera a los 16 bits del microprocesador o que su resultado sea mayor que 16 bits, se debe recurrir al uso de la suma con acarreo.
Esto significa, que en el caso por ejemplo de sumar dos números de 32 bits, se realiza la suma de los 16 bits menos significativos y luego los 16 bits mas significativos junto con el acarreo.



J.T.P. Ing. Steiner Guillermo.
Email:gsteiner@scdt.frc.utn.edu.ar

Ejercicio Nro 1

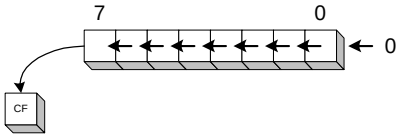
Sumar 2 nro de 32 bits contenidos en AX BX y CX DX respectivamente, guardando el resultado en AX BX.
AX BX = 0134A23Bh
CX DX = BD02E329h

```
; sumar dos números de 32 bits
name "suma32"
org 100h
mov ax,0134
mov bx,A23B
mov cx,BD02
mov dx,E329
add bx,dx
adc ax,cx
ret
```

AX BX = BE378564 (Resultado)

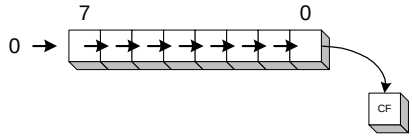
Movimiento (Shift) común y Aritmético

SHL (Shift Left)



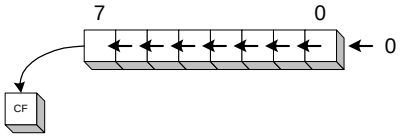
Multiplicar sin signo

SHR (Shift Right)



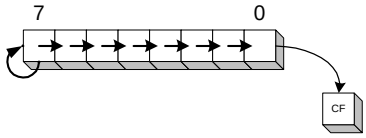
Dividir sin signo

SAL (Shift Arithmetic Left)



Multiplicar con signo

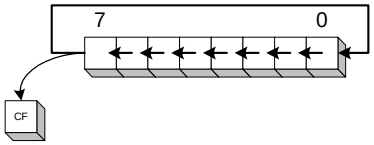
SAR (Shift Arithmetic Right)



Dividir con signo

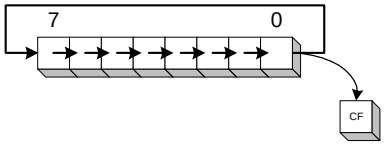
Rotación

ROL (Rotate Left)



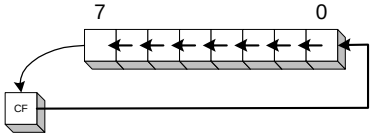
Rotación a la izquierda de 8 bits

ROR (Rotate Right)



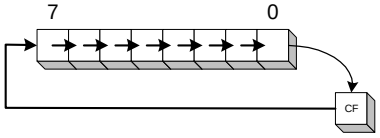
Rotación a la derecha de 8 bits

RCL (Rotate through Carry Left)



Rotación a la izquierda de 9 bits

RCR (Rotate through Carry Right)



Rotación a la derecha de 9 bits

J.T.P. Ing. Steiner Guillermo.
Email:gsteiner@scdt.frc.utn.edu.ar

Formato de las instrucciones

Si bien se pueden realizar corrimiento o rotaciones de 1 o más bits, solo está permitido el de 1 bit en inmediato, en estos casos para un corrimiento de más de un bits, se debe utilizar un registro auxiliar, el CL

Ejemplos

Shift a la derecha de un 1 bit

```
SHR BX,1
```

Shift a la derecha de 4 bits

```
MOV CL,4
SHR BX,CL
```

Ejercicio Nro 2

Usando instrucciones Shift ingresar un número en AX y multiplicarlo por 7

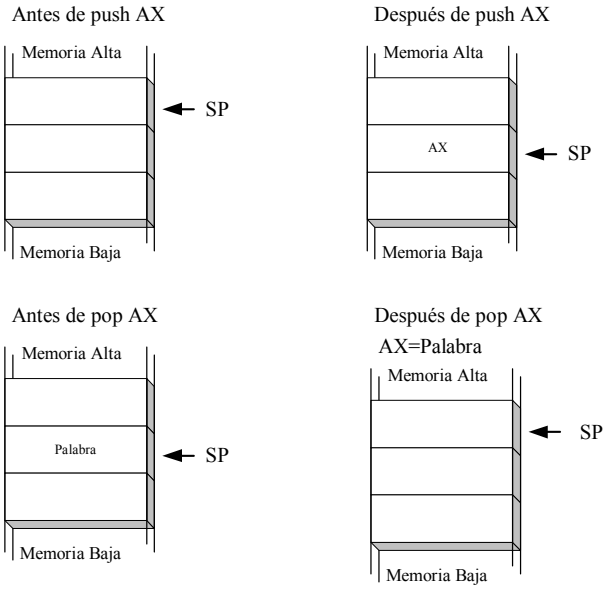
```
; multiplicar por 7
name "multip7"
org 100h
mov  ax,023Ah
mov  bx,ax
mov  cl,03
shl  bx,cl
sub  bx,ax
ret ; volver al sistema operativo
```

Resultado 0F96h

Stack

Lugar de memoria destinado a guardar datos temporales, es una pila (LIFO)
Los registros usados por el stack son:
SP: Stack Pointer – Puntero del stack
SS: Stack Segment – Segmento del stack
Quedando la dirección absoluta **SS:SP**
Funciones que usan el Stack
PUSH: Decrementa el SP en 2 y luego graba 2 bytes al stack.
POP: Lee los últimos 2 bytes grabados al Stack e incrementa el SP en 2.
También dan uso del Stack: las interrupciones de Hardware y Software y las llamadas a subrutinas.

Estado del Stack antes y después del uso de las instrucciones POP y PUSH



Ejercicio Nro 3

Salvar al los registros AX y BX, luego borrar los mismos para finalmente recuperar los datos del Stack

```
; manejo del stack
; NOTAR EL ORDEN EN QUE SE RECUPERA LOS REGISTROS
name "stack"
org  100h
mov  ax,10
mov  bx,20
push ax
push bx
sub  ax,ax
sub  bx,bx
pop  bx
pop  ax
ret ; volver al sistema operativo
```

Aritmética Entera
Instrucciones a usar MUL, IMUL, DIV, IDIV

Multiplicación

Tamaño Operador	**Multiplicando**	**Multiplicador**	**Resultado**
8 bits	AL	Byte registro o memoria	AX
16 bits	AX	Palabra registro o memoria	DX:AX

División

Tamaño Operador	**Dividendo**	**Divisor**	**Resultado**	**Resto**
16 bits	AX	Byte registro o memoria	AL	AH
32 bits	DX:AX	Palabra registro o memoria	AX	DX

Ejercicio Nro 4

Dividir AX por 7 y Multiplicar BX por 23

```
; dividir y multiplicar enteros
name "divymul"
org 100h
mov  ax,0100h
mov  bx,2000h

mov  cl,7     ; divisor
div  cl       ; dividendo = ax ; divisor = cl
mov  cx,ax    ; guardar resultado y resto en cx
mov  ax,bx    ; multiplicando
mov  bx,23    ; multiplicador
mul  bx       ; resultado en dx ax

ret ; volver al sistema operativo
```

Resultado: división 24h con resto 4h – multiplicación 2E000h

Saltos Condicionales e Incondicionales

En casi todo programa muchas veces es necesario interrumpir el flujo normal del mismo y saltar a otra porción del código, podemos distinguir dos saltos diferentes:

- Los condicionales, donde se evalúa una condición y al ser verdadera se produce el salto, es la forma que tiene el ensamblador de realizar IF o estructuras CASE que acostumbrábamos a usar en lenguajes de alto nivel, lo único que aquí lo hacemos en dos pasos, primero comparamos, quedando el resultado en las banderas (ver banderas al comienzo), luego utilizamos alguna de las instrucciones de salto condicional para realizar el salto.
- El salto incondicional, aquí no hay condición de salto y el mismo se efectúa siempre, es el equivalente al GOTO en lenguajes de alto nivel.

J.T.P. Ing. Steiner Guillermo.
Email:gsteiner@scdt.frc.utn.edu.ar

Salto Incondicional

```
JMP nn
```

El salto incondicional puede ser dividido en tres tipos.

- Salto corto, en este tipo de salto el parámetro nn es un byte, pudiendo saltar desde +127 hasta -128 posiciones a partir de la dirección próxima a JMP. (largo: 1 byte instrucción + 1 byte salto = 2 bytes)
- Salto cercano, el parámetro nn es de dos bytes indicando la dirección del puntero a donde va a saltar. (largo: 1 bytes instrucción + 2 bytes salto = 3 bytes)
- Salto lejano, nn es de 4 byte, indica la posición absoluta con segmento y puntero del salto. (largo: 1 byte instrucción + 4 byte salto = 5 bytes)

Generalmente el compilador elije la mejor forma (la más corta) para realizar el salto, no siendo necesario determinar el tipo de salto.

Salto condicionales

Este tipo de salto depende de las banderas del uP para realizar el salto, todos estos saltos son solamente corto, no pudiendo saltar con estas instrucciones mas allá de +127 – 128 bytes.

Saltos condicionales para número sin signo

Instrucción	Bandera/s	Descripción
JA	C = 0 y Z = 0	Saltar si está por arriba
JAE	C = 0	Saltar si está por arriba o es igual
JB	C = 1	Saltar si está por abajo
JBE	C = 1 o Z = 1	Saltar si está por debajo a igual

Saltos condicionales para número con signo

Instrucción	Bandera/s	Descripción
JG	Z = 0 y S = O	Saltar si es mayor
JGE	S = O	Saltar si es mayor o igual
JL	S ≠ O	Saltar si es menor
JLE	Z = 1 o S ≠ O	Saltar si es menor o igual
JNS	S = 0	Saltar si no hay signo
JS	S = 1	Saltar si hay signo

Otros saltos

Instrucción	Bandera/s	Descripción
JE o JZ	Z = 1	Saltar si es igual o cero
JNE o JNZ	Z = 0	Saltar si no es igual o cero
JC	C = 1	Saltar si hay acarreo
JNC	C = 0	Saltar si no hay acarreo
JNO	O = 0	Saltar si no hay overflow
JNP o JPO	P = 0	Saltar si no hay paridad o paridad impar
JO	O = 1	Saltar si hay overflow
JP o JPE	P = 1	Saltar si hay paridad o paridad par
JCXZ	CX = 0	Saltar si CX = 0

Ejercicio Nro. 4

Realizar un programa que: dado un vector de byte ya cargado, busque el final del mismo (byte = 0h) y termine dejando en BX la longitud del mismo.

```
; Vector
name "vector"
org 100h
    mov bx,0       ; en bx tenemos la posición dentro del vector
otro:
    mov al,vec[bx] ; cargamos en AL el elemento del vector indicado en BX
    inc bx         ; incrementamos BX
    cmp al,0       ; comparamos el elemento del vector con 0
    jz fin         ; si la comparación es 0 salimos del programa
    jmp otro       ; buscamos otro elemento
fin:
    ret ; volver al sistema operativo
vec db "abcdefghijklmnopqrstuvwxyz",0 ; cargar un vector con constantes
```

Loop

La instrucción loop se utiliza para realizar bucles, esta instrucción decrementa el contador CX y lo compara con cero en caso de no ser cero salta a la dirección indicada en el parámetro, de ser cero continua con la instrucción siguiente.

Ejercicio Nro. 5

Realizar un programa que: dado un vector de byte ya cargado de 10 elementos, sume los mismos y termine con el resultado de la suma en AX.

```
; suma
name "suma"
org 100h
    mov bx,0       ; en bx tenemos la posición dentro del vector
    mov cx,10      ; contador del programa para el loop
    mov ax,0       ; suma
    mov dx,0
otro:
    mov dl,vec[bx] ; cargamos en DL el elemento del vector indicado en BX
    inc bx         ; incrementamos BX
    add ax,dx
    loop otro
    ret            ; volver al sistema operativo
vec db 10,20,2,200,34,44,21,8,10,22; cargar un vector con constantes
```

Práctico de Aula no Desarrollados

Los programas pedidos deberán figurar en la carpeta con su código completo.

Ejercicio Nro 1

Realizar un programa ejemplo para cada tipo de direccionamiento, este programa ejemplo debe sumar dos números.

- Direccionamiento por registro.
- Direccionamiento inmediato.
- Direccionamiento directo.
- Direccionamiento base más índice.
- Direccionamiento relativo por registro.
- Direccionamiento relativo base más índice.

J.T.P. Ing. Steiner Guillermo.
Email:gsteiner@scdt.frc.utn.edu.ar

Ejercicio Nro 2

Realizar un programa que dada dos variables word previamente cargadas con sendos numero con signo, realice la suma de las mismas y la resta, el tamaño del resultado deben ser tal, que no puedan producirse desborde cualquiera sean los números a sumar o restar.
(Cuando nos referimos a una cadena o variable previamente cargada, significa que en el momento en que declaran el vector o variable en el programa, le ingresan los valores como en el caso de los ejercicios resueltos 4 y 5)

Ejercicio Nro 3

Realizar un programa que mediante corrimiento divida por 8 un número con signo previamente cargado en una variable word. (Explicar que instrucción uso para el corrimiento y porque)

Ejercicio Nro 4

Usando únicamente las instrucciones de corrimiento y rotación, realice la multiplicación de un número previamente cargado en el acumulador AX por 0Bh, sabiendo que el resultado no será mayor a 16 bits, realice el mismo ejemplo con la función de multiplicación MUL, calcule cuantos ciclos de reloj utiliza cada procedimiento.

Ejercicio Nro 5

Realizar un programa que mediante corrimiento cuente la cantidad de unos que hay en una cadena de 4 bytes previamente cargada, el resultado debe quedar en AL.

Ejercicio Nro 6

Realizar un programa que: Divida por 10 a cada elemento de un vector de Word previamente cargado, el vector no tiene longitud definida, se debe dividir hasta encontrar un 0h, los resultado se guardan en el mismo vector y deberá ser Word para evitar sobreflujo, el resto de la división de descarta.

Ejercicio Nro 7

Mostrar el “boot sector” de un disco de 3½” con formato FAT, el programa deberá reemplazar todos los bytes que no son imprimibles (menores a 20h) por “.”, mostrarlos en pantalla todos los bytes del sector en ASCII.
(El “boot sector” se encuentra en el cilindro = 0, head = 0 y sector = 1 y mide como todos los sectores 512 bytes.)

Ejercicio Nro 8

Realizar un programa que imprima un asterisco en el centro de la pantalla y que realice un scroll de una línea hacia arriba en un ventana de coordenadas (8,10), (16,70).
Para el scroll podemos utilizar un servicio de la BIOS.