Trabajo Práctico Nro. 4 Multiplicación de 32 bits

Introducción

Este ejemplo muestra el modelo para general un archivo .COM, el uso de funciones y del manejo de interrupciones del DOS, bajo el Masm 6.11 o compatible

Directivas utilizadas en el programa

.MODEL Ingresamos aquí el modelo a usar para nuestro proyecto, los modelos pueden ser:

- **Tiny:** modelo pequeño donde los CS, DS, SS, ES poseen la misma dirección, además este es el único modelo soportado para la generación de .COM, con este modelo aparte de cargar todos los registros de segmento con la misma dirección, se le indica al linker para que compile con .COM.
- **Small:** se separa en dos segmentos al programa uno para los datos y otro para el código esto permite tener todo saltos **near** por defectos.
- **Large:** este modelo permite múltiples segmentos tanto para datos como para código siendo por defecto saltos **far**.
- **Medium:** permite solo un segmento de dato y múltiples segmentos de código.
- **Compact:** permite múltiples segmentos de dato y un solo segmento de código.

.STACK *n* Establece el lugar para que se ubique el Stack al prescindir de parámetros el compilador deja un espacio de 1024 bytes, si se desea mas espacio se le debe incluir como parámetros.

.DATA En este bloque se colocan las variables a utilizar en el programa.

.CODE Comienza el código de programa.

.STARTUP Esta directiva será reemplazada, según el modelo con que se configure el programa, para el caso de .COM únicamente será que el programa empiece en la dirección de offset 100h. (reemplaza al org 100h).

nombre **PROC NEAR** / *nombre* **ENDP** De esta forma se declaran los procedimientos o funciones del assembler. La directiva NEAR indica un procedimiento cercano es decir que el salto será de 16 bit (únicamente el offset).

Ejemplo

```
MAX_TEXTO  EQU    30    ; tamaño máximo del titulo a leer
      .MODEL      tiny
      .STACK
      .DATA
cadena   DB MAX_TEXTO DUP('$')
lcadena  DB 0
contador DB 0

;     ================================================================
;     main del programa
;     ----------------------------------------------------------------
      .CODE
      .STARTUP
        mov     di, 80h                 ; puntero de comienzo del texto
        mov     bl, es:[di]
        mov     bh, 0                   ; carga longitud del texto
        or      bx, bx
        je      SParam                  ;

        mov     WORD PTR es:[bx+81h], 0 ; poner en cero el ultimo byte
                                        ; de la cadena
```

```
        mov     cx,bx
        mov     al, ' '                 ; buscar hasta que sea diferente
otro1:  inc     di                      ; a espacio, de esta forma elimino
        cmp     al,es:[di]              ; los espacios al comienzo
        jne     noesp
        loop    otro1

noesp:  mov     si, di                  ; cadena fuente (parámetro ingresado)
        mov     di, OFFSET cadena       ; cadena destino
        mov     cx, MAX_TEXTO - 1       ; Count = máximo tamaño permitido - 1
        mov     lcadena,0               ; inicializa lcadena

otro2:  mov     al,es:[si]              ; cargar el vector fuente en
        mov     [di],al                 ; el vector destino
        cmp     al,' '                  ; hasta encontrar un ' ' o 0
        je      listo                   ; o hasta llegar a cx bytes copiados
        cmp     al,0
        je      listo
        inc     si
        inc     di
        inc     lcadena
        loop    otro2
listo:  mov     BYTE PTR [di],'$'
SParam: mov dh,8                         ; Imprimir Linea
      mov   dl,21
      call  setcursor
      mov   al,'-'
      mov   cx,39
      call  impcaracter

      mov   contador,10
REP1: mov   dh,contador
      add   dh,8
      mov   dl,20
      mov   al,'|'
      call  caracter                     ; imprimir | en columna 20
      mov   dl,60
      call  caracter                     ; imprimir | en columna 60

      dec   contador
      jnz   REP1

      mov   dh,19                        ; Imprimir Linea
      mov   dl,21
      call  setcursor
      mov   al,'-'
      mov   cx,39
      call  impcaracter

      mov   dh,8                         ; Imprimir Titulo
      mov   dl,22
      call  setcursor
      mov   ah, 9h                       ; Int 21 func:09
      mov   dx, OFFSET cadena ;          ; dx = punt. de cadena a imprimir
      int   021h
      .EXIT
```

```
;     ================================================================
;     setcursor
;     mediante el serv. 2 de la int 10h ubica el cursor en la pantalla
;     parámetros: dh = y dl = x
setcursor PROC NEAR
      mov   ah,02h
      mov   bh,00h
      int   10h
      ret
setcursor ENDP
;     ================================================================
;     impcaracter
;     mediante el serv. Ah de la int 10h imp. el cursor en la pantalla
;     parámetros: al = ascii del carac.a imp.  cx = cant.de caracteres
impcaracter PROC NEAR
      mov   ah,0Ah
      mov   bh,00h
      int   10h
      ret
impcaracter ENDP

;     ================================================================
;     caracter
;     utiliza las funciones de setcursor y impcaracter para imprimir
;     un caracter dado en la pantalla
;     parámetros: dh = y  dl = x  al = caracter
caracter PROC NEAR
      push  ax
      call  setcursor
      pop   ax
      mov   cx,1
      call  impcaracter
      ret
caracter ENDP
      END
```

Multiplicación

El método mas difundido de multiplicación de dos números cuando estos exceden las posibilidades de ser realizado por el microprocesador, es una serie de corrimientos y suma.
El método puede reducirse a un par de pasos como los que siguen:

a) Se realiza un shift a la derecha del multiplicador.
b) Si el carry es 0 saltar al punto **d**.
c) Sumar el contenido del multiplicando al resultado.
d) Realizar un shift a la izquierda el multiplicando.
e) Saltar al punto **a.**

Esta rutina deberá repetirse **n** veces, donde **n** es la longitud en bits del multiplicador.

Ejemplo en Assempler de una multiplicación.

Multiplicando: 16 bits
Multiplicador: 8 bits
Resultado: 16bits

```
            mov WORD PTR mul1,003F
            mov BYTE PTR mul2,4A
            mov WORD PTR resul,0
            mov CX,8
OTRO        shr BYTE PTR mul2,1
            jnc NOSUMAR
            mov AX,mul1
            add resul,AX
NOSUMAR     shl WORD PTR mul1,1
            loop OTRO
```

Práctico a Desarrollar

Ejercicio Nro 1

Se pide crear un programa que multiplique dos números de 32 bits cada uno y devuelva el resultado en un número de 64 bits, pudiéndose usar para este programa solamente las instrucciones de corrimiento y suma.
El programa deberá recibir dos parámetros como entrada, los cuales serán los dos números a multiplicar, mostrando el resultado en pantalla. **(VER EJEMPLO).**
Los números a ingresar como los mostrados por la pantalla, pueden estar en base 10 o 16.

Ejemplo

```
C:\>mult 2341 231
El resultado es: 540771
```