Práctica No. 9

Procedimientos en el lenguaje ensamblador del procesador 8086

Objetivo:
Familiarizarse con la estructura de los procedimientos para programas en leguaje ensamblador del procesador 8086

Materiales:
TASM.exe, TLINK.exe, PCLIB06.lib, formato.asm, procs.inc

Teoría:
Hacer una reseña sobre:

- Conversiones numéricas, por ej. DEC a BIN, OCT a HEX, etc.

Desarrollo:
PARTE 1.
Usando la biblioteca PCLIB06.lib, crear el ejecutable a partir del código del ANEXO.

Actividad para validar el desarrollo de esta parte:

1. El programa deberá ejecutarse y funcionar tal como se pide.
2. Las conversiones mostradas del ANEXO son a partir del registro AL, probar con al menos dos valores más.

PARTE 2.
En base al código del ANEXO implementar un procedimiento **changeBase** bajo las siguientes restricciones:

1. Se captura el valor a ser convertido hacia AX
2. Se captura el valor de la base de conversión final hacia BX
3. Se captura el valor de la base de entrada, hacia un registro de elección libre

Nota. Se pueden limitar a base 2, 8, 10 y 16

Actividad para validar el desarrollo de esta parte:

1. El programa deberá ejecutarse y funcionar tal como se pide y bajo las restricciones dadas.

Nota. La biblioteca PCLIB06.lib no podrá ser utilizada para esta parte de la práctica.

ANEXO.

```
MODEL small
.STACK 100h
      ;----- Insert INCLUDE "filename" directives here
      ;----- Insert EQU and = equates here

INCLUDE procs.inc
LOCALS

.DATA
      mens_ascii db 10,13,"AL desplegado en ASCII:',0
      mens_bin db 10,13, "AL desplegado en Binario:',0
      mens_dec db 10,13, "AL desplegado en Decimal:',0
      mens_hex db 10,13, "AL desplegado en Hexadecimal:',0

.CODE ;----- Insert program, subrutine call, etc., here
Principal    PROC
             mov ax,@data              ;Inicializar DS al la direccion
             mov ds,ax                 ; del segmento de datos (.DATA)
             call clrscr
             mov al,3fh                ; dato a desplegar
             mov dx, offset mens_ascii
             call puts
             call putchar              ; imprime AL en ASCII
             mov dx, offset mens_bin
             call puts
             call printBin             ; desplegar AL en binario
             mov dx, offset mens_dec
             call puts
             call printDec             ; desplegar AL en decimal
             mov dx, offset mens_hex
             call puts
             call printHex             ; desplegar AL en decimal
             mov ah,04ch               ; fin de programa
             mov al,0
             int 21h
             ret
             ENDP
; --- procedimientos ----
printBin     PROC
             push ax      ; salvar registros a utilizar
             push cx
             mov cx,8     ; incializar conteo a 8
             mov ah,al    ; AH sera el registro a desplegar
    @@nxt: mov al,'0'     ; prerar a AL para imprimir ASCII
             shl ah,1     ; pasar el MSB de AH a la bandera de acarreo
             adc al,0     ; sumar a AL el valor del acarreo
             call putchar
             loop @@nxt   ; continuar con el proximo bit
             pop cx       ; recuperar registros utilizados
             pop ax
             ret
             ENDP
;************************************************
printDec     PROC
             push ax      ; salvar registro a utilizar
             push bx
             push cx
             push dx
             mov cx,3     ; inicializar conteo a 3 (cent-dec-unida)
             mov bx,100   ; iniciar con centenas
             mov ah,0     ; asegurar AX = AL
```

```
    @@nxt: mov dx,0        ; asegurar DX=0 para usar div reg16
           div bx          ; dividir DX:AX entre BX
           add al,'0'      ; convertir cociente a ASCII
           call putchar ; desplegar digito en pantalla
           mov ax,dx       ; pasar residuo (DX) a AX
           push ax         ; salvar temporalmente AX
           mov dx,0        ; ajustar divisor para nuevo digito
           mov ax,bx       ; la idea es:
           mov bx,10       ;            BX = BX/10
           div bx
           mov bx,ax       ; pasar cociente al BX para nuevo digito
           pop ax          ; recupera AX
           loop @@nxt      ; proximo digito
           pop dx
           pop cx
           pop bx
           pop ax
           ret
           ENDP
;************************************************
printHex   PROC
           push ax         ; salvar registros a utilizar
           push bx
           push cx
           mov ah,0        ; asegurar AX = AL
           mov bl,16
           div bl          ; dividir AX/16 --> cociente en AL y residuo AH
           mov cx,2        ; para imprimir dos digitos hex
    @@nxt: cmp al,10       ; verifica si cociente AL es menor a 10
           jb @@print
           add al,7
    @@print: add al,30h    ; si es menos a 10 sumar 30h de lo contrario 37h
           call putchar
           mov al,ah       ; pasa residul (AH) a AL para imprimirlo
           loop @@nxt      ; proximo digito
           pop cx
           pop bx
           pop ax          ; recupera registos utilizados
           ret
           ENDP
;************************************************
```