Macros y procedimientos[1]

Una macro es un conjunto de instrucciones que realizan una tarea específica. Las macros reciben parámetros y son llamadas como si fueran una instrucción, es decir, se comportan como funciones.

Un procedimiento es también un conjunto de instrucciones que hacen alguna tarea repetitiva, pero es llamada a través de una directiva y representa un salto en el listado de instrucciones. Su comportamiento depende de los valores almacenados en los registros y pila al momento de ser llamado.

El reúso de código es una práctica común de los programadores para aumentar su productividad y para tener control de su propiedad intelectual sobre el código. En el caso de ensamblador la definición de procedimientos y macros permite la creación de librerías con subrutinas y funciones auxiliares.

Estos procedimientos y macros permiten además que el código fuente del programa sea más legible y por consecuencia sea más sencillo hacer mantenimiento y encontrar errores, es decir, depurar. Permite, además, crear ejecutables más compactos y probablemente más veloces, pues al crear las subrutinas optimiza el manejo de memoria.

Macros

Las macros son subrutinas que pueden recibir parámetros. Pueden representar una función que se ejecuta en un programa en diversas ocasiones con parámetros diferentes. Si se define una macro sin parámetros de entrada se le llama simplemente con su nombre, como si fuera una nueva instrucción de ensamblador. Cuando una macro recibe parámetros, éstos se le pasan al momento de llamarla separados por coma. Se utilizan las directivas **MACRO**, **ENDM** y **LOCAL**. La estructura para definir una macro es la siguiente:

```
Nombre Macro MACRO parámetro_1, parámetro_2,…
LOCAL Etiqueta_interna_1, Etiqueta_Interna_2,…

** Conjunto de instrucciones de la función ***
ENDM
```

Por ejemplo, la función para esperar una tecla (getche) se puede implementar de manera muy sencilla como:

[1] Documento elaborado por el Dr. Carlos Jiménez Gallegos.

```
; *** getche: espera una tecla ****
getche MACRO
     MOV AH, 08H    ; espera tecla
     INT 21H
ENDM
;**********************************
```

La declaración de la macro puede ir al final o al principio del código, pero lo más usual es escribir todas las macros en un archivo separado al que se le pone la extensión ***.lib**. Con ello se define una librería de funciones que puede ser llamada con la directiva INCLUDE:

```
INCLUDE Milibreria.lib
getche         ; llamar a la macro
```

La creación de macros permite, además de construir de forma estructurada, macros complejas, pues una macro puede mandar llamar a otras macros permitiendo conservar la legibilidad del programa principal y de la librería y sin pérdida de velocidad en el procesamiento. El siguiente código compila algunas macros que podrían ser útiles para graficar cosas sencillas:

```
; **** LIBRERÍA DE MACROS *****
; **** GUARDAR como Librería.lib
;*** muestra: Muestra una cadena ***
muestra MACRO texto
     LEA DX,texto    ;Carga el texto
     MOV AH,9h       ;imprimir en pantalla
     INT 21h         ;Llama al sistema operativo
ENDM
;**********************************

;*** cls: borrar pantalla **********
cls MACRO
     MOV AH,00H     ;Espacios en blanco
     MOV AL,03H     ;Borrar
     INT 10H
ENDM
;**********************************

;*** gotoxy: coloca el cursor ******
gotoxy MACRO x,y
```

```
     pushr
     MOV BH,0H
     MOV DH,y  ; renglón
     MOV DL,x       ; columna
     MOV AH,02H     ; colocar cursor
     INT 10H
     popr
ENDM
;*********************************

; *** getche: espera una tecla ****
getche MACRO
     MOV AH, 08H    ;espera tecla
     INT 21H
ENDM
;*********************************

; *** marco: dibuja un marco ******
; *** en las coordenadas dadas ****
marco MACRO xi, xf, yi, yf
local inicio
  linea xi,xf,yi,0CDH,0BAH ; 1er Linea
  linea xi,xf,yf,0CDH,0BAH ; última Linea
     MOV CX, 0000H     ; Contador espacio
     MOV CL, yf
     SUB CL, yi
     MOV AL, yi
   inicio:          ; rellena el cuadro
     DEC CL
     INC AL
     linea xi,xf,AL,00H,0BAH
     CMP CL,1
     JNZ inicio
  ; coditos
     uncxy 201, xi, yi
     uncxy 200, xi, yf
     uncxy 187, xf, yi
     uncxy 188, xf, yf
ENDM
```

```
;*********************************

; *** linea: dibuja una linea de   ***
; *** caracteres (medio y orillas) ***
linea MACRO xi,xf,y,cm,co
     pushr
   ; Toda la linea
     MOV DL, y
     gotoxy xi ,DL
     MOV AH, 09H
     MOV AL, cm
     MOV BX, 001FH
     MOV CX, xf
     SUB CX, xi
     INT 10H
   ; orilla izquierda
     gotoxy xi, DL
     MOV AH,09H
     MOV AL, co
     MOV BX, 001FH
     MOV CX, 1
     INT 10H
   ; orilla derecha
     gotoxy xf, DL
     MOV AH, 09H
     MOV AL, co
     MOV BX, 001FH
     MOV CX, 1
     INT 10H

     popr
ENDM
;*********************************

; *** uncxy: escribe un caracter en **
; *** la posición x, y ***************
uncxy MACRO c, x, y
     pushr
     gotoxy x, y
```

```
      MOV AH, 09H
      MOV AL, c
      MOV BX, 001FH
      MOV CX, 1
      INT 10H
      popr
ENDM
; ***********************************

; *** pushr: guarda los registros
pushr MACRO
      PUSH AX
      PUSH BX
      PUSH CX
      PUSH DX
ENDM
;*********************************

; *** popr: recupera los registros
popr MACRO
      POP DX
      POP CX
      POP BX
      POP AX
ENDM
; ******************************

; *** macro unc: imprime un caracter ***
unc macro c
pushr
mov dl, C
MOV ah, 06h
int 21h
popr
endm
;*********************************

; *** FIN DE LAS MACROS ***
```

Con esta librería ya podríamos definir un programa compacto con un manejo de gráficos (en modo texto) básico:

```
INCLUDE libreria.lib

.MODEL SMALL                        ;Modelo pequeño
.STACK 100h                         ;Segmento de pila 256 posiciones
.DATA                               ;Segmento de datos

TEXTO1 DB 'Programas en ENSAMBLADOR $'
TEXTO2 DB 'UTEL 2014'
TEXTO3 DB 'Menu:$'


.CODE                               ;Código del programa

main PROC

; ** inicializar segmentos
MOV AX, @DATA
MOV DS,AX
MOV ES,AX

;  ******* AQUI SE INICIA EL PROGRAMA *******

cls
marco 0,79,0,24 ; marco grande
marco 1,38,6,22 ; marco medio izquierda
marco 39,78,6,22 ; marco medio derecha

gotoxy 14,2
muestra TEXTO1
gotoxy 26,3
muestra TEXTO2
gotoxy 35,5
muestra TEXTO3

getche
cls
```

```
;  ******* AQUI SE FINALIZA EL PROGRAMA *******

MOV Ax,4c00h                        ;Función 4ch
INT 21h
ret

main ENDP

;**************  FIN **********************
END                                 ;Fin del programa
```

Note como ahora el programa principal es totalmente compacto y legible

Procedimientos

Los procedimientos son subrutinas que contienen un bloque de comandos que se desea llamar como una sola instrucción. Los procedimientos son usados para partes de código que se repiten con mucha frecuencia, como los procesos de ajuste de valores o que representan tareas auxiliares, como mandar a la pantalla. Cuando se hace una llamada a un procedimiento en realidad el procesador hace un salto al segmento de memoria en el que radica el procedimiento y al finalizar regresa a la siguiente instrucción de la lista. Todo el contenido de los registros y la pila está accesible al procedimiento, pues éste es equivalente a tener sus instrucciones escritas en el punto en que se le llamó. Lo que quiere decir que los procedimientos también pueden modificar los valores en registros, variables y memoria en general, por ello se debe tener cuidado cuando se llama a un procedimiento que escribe o usa registros en los que posiblemente tengamos datos que queramos conservar.

El formato de un procedimiento usa las directivas **PROC**, **ENDP**, **RET** y **CALL**. La definición de un procedimiento tiene la estructura:

```
Nombre PROC
*
* Conjunto de instrucciones *
*
RET

Nombre ENDP
```

Donde nombre es la etiqueta con que se conocerá al procedimiento. En un programa los procedimientos pueden escribirse al principio o, como es costumbre, hasta el final.

La forma de llamar a un procedimiento es con el comando CALL:

```
CALL nombre
```

Donde *nombre* sea un procedimiento. Veamos ahora una segunda versión del programa que mostraba ciertas operaciones en pantalla. En este caso usando procedimientos. Note que ahora debemos definir al programa principal con **main PROC** para que el sistema sepa cual procedimiento es el que debe de ejecutar dado que los demás procedimientos pueden residir en el mismo archivo de código.

```
.MODEL SMALL                        ;Modelo pequeño

.STACK 100h                         ;Segmento de pila 256 posiciones

CR EQU 13                           ;Retorno de carro
LF EQU 10                           ;nueva linea

.DATA                               ;Segmento de datos

TEXTO1 DB 'Multiplicaci',163,'n de dos cifras   $',CR,LF  ;Texto
TEXTO2 DB CR, LF, 'Proporcione el primer n',162,'mero (dos cifras) x=
$',CR,LF
TEXTO3 DB CR, LF, 'Proporcione el segundo (dos cifras) y= $', CR,LF
TEXTO4 DB CR, LF, 'Los resultados x/y x+y y xy son: $', CR,LF

.CODE                               ;Código del programa

main PROC

MOV BL,TEXTO1

; **********   Texto de presentación ****
MOV AX,@DATA
MOV DS,AX                            ;Cargar la dirección en el segmento de
datos

LEA DX,texto1                       ;Carga el texto

MOV AH,9h                           ;imprimir en pantalla
INT 21h
                                    ;Llama al sistema operativo
```

```
;  ******* AQUI SE INICIA EL PROGRAMA *******


; ** 1) Pedir primer valor

LEA DX,TEXTO2                    ;Carga el texto
MOV AH,9h                        ;imprimir en pantalla
INT 21h
                                ;Llama al sistema operativo

call dosv             ; Recibe el primer valor
MOV BH,BL             ; se guarda en BH

; ** 2) Pedir segundo valor

LEA DX,TEXTO3                    ;Carga el texto
MOV AH,9h                        ;imprimir en pantalla
INT 21h                          ;Llama al sistema operativo

call dosv             ; Recibe el segundo valor
                      ; El resultado está en BH


; ** 3) Operaciones

; ** 3.1) multiplicación
MOV AL,BH
MOV CL,BL
MUL CL
PUSH AX

; ** 3.2) suma
MOV AX,0000H
MOV AL,BH
ADD AL,BL
PUSH AX
```

```
; ** 3.3) división
MOV AL,BH
MOV CL,BL
DIV CL
MOV AH, 00H


; ** 4) Mostrar resultados

; ** 4.1) letrero
MOV BX,AX           ; Guardar resultado de la división

LEA DX,TEXTO4                    ;Carga el texto
MOV AH,9h                        ;imprimir en pantalla
INT 21h                          ;Llama al sistema operativo

MOV AX, BX                   ;volver a colocar el resultado en AX

; **4.2) resultado entero de la división

call MosCu

; **4.3) resultado de la suma

POP AX
call MosCu

; **4.4) resultado de la multiplicación

POP AX
call MosCu

;  ******* AQUI SE FINALIZA EL PROGRAMA *******

MOV Ax,4c00h                     ;Función 4ch
INT 21h
ret

main ENDP
```

```
; ***********************************************
; **** PROCEDIMIENTOS UTILIZADOS ****************
; ***********************************************

; ************ Procedimiento imprime espacio *******
espacio  PROC
MOV DL,20H
MOV AH, 06H      ;06H imprimir un espacio
INT 21H
RET
espacio ENDP
; ************************************************

; ************ Procedimiento para imprimir 2 caracteres (BH BL)

dosc PROC
MOV DL,BH
MOV AH, 06H      ;06H imprimir ASCII que está en DL
INT 21H
MOV DL,BL
MOV AH, 06H      ;06H imprimir ASCII que está en DL
INT 21H
RET
dosc ENDP
; ************************************************************

; *********** Procedimiento para recibir dos cifras (guarda en AX)

dosv PROC

; ** a Y primer caracter (decenas) y multiplica por 10

MOV AH,01               ; Leer del teclado 1 caracter
INT 21H
SUB AL,30H                ;obtiene valor en decimal
MOV CL,0AH
MUL CL                    ;multiplica por 10
MOV BL,AL            ; se guarda en DH
```

```
; ** b Y segundo caracter (unidades) y lo suma al anterior
MOV AH,01               ; Leer del teclado 1 caracter
INT 21H
SUB AL,30H                ;obtiene valor en decimal
ADD BL,AL                 ;se suma a las decenas

RET
dosv ENDP
;*****************************************************


; ******** Ajuste Miles y Centenas (de AX a BX)*********************

AjuMyC PROC

MOV DX,0000H
MOV CX,100
DIV CX          ; AX/CX= AX/100 (centenas)
AAM             ;ajuste de las centenas
ADD AX,3030H  ; ajustes ASCII centenas
MOV BX,AX
MOV CX,DX

RET
AjuMyC ENDP
; ****************************************************

; ********** Ajuste decenas y unidades (de CX a AX) *****

AjuDyU PROC

XCHG AX,CX    ; intercambiar registros
AAM         ; Ajuste decenas y unidades
ADD AX, 3030H ; Ajuste ASCII DECENAS Y UNIDADES

RET
AjuDyU ENDP
; ****************************************************
```

```
; ******* Mostrar 4 cifras a partir de hexadecimal ******

MosCu PROC

call AjuMyC
call dosc
call AjuDyU
MOV BX,AX
call dosc
call espacio    ;imprime un espacio

RET
MosCu ENDP
; ********************************************************

END
```