ESCRITURA DE MACROS

Definición: son un conjunto de instrucciones en ensamblador agrupadas bajo un nombre simbólico que las sustituirá en aquellos puntos donde aparezcan.

- La definición de una macro dentro del programa fuente, debe aparecer *antes que cualquier definición de segmento*
- La definición de la macro no consume memoria, por lo que en la práctica es indiferente declarar cientos que ninguna macro.

Ejemplo:

- La macro se define por medio de la directiva MACRO.
- El nombre simbólico es el que permitirá en adelante hacer referencia a la macro, y se construye casi con las mismas reglas que los nombres de las variables y demás símbolos.
- La macro puede contener parámetros de manera opcional, seguida de las instrucciones y, finalmente, la directiva ENDM señala el final de la macro.
- No se debe repetir el nombre simbólico junto a la directiva ENDM

```
NOMBRE-SIMBOLICO    MACRO
                    MOV AX, @DATA
                    MOV DS, AX
                    MOV EX, AX
                ENDM
```

Llamado a la macro desde el programa:**NOMBRE-SIMBOLICO**

Referencia: Abel, Meter. LENGUAJE ENSAMBLADOR Y PROGRAMACIÓN PARA IBM-PC Y COMPATIBLES. Tercera Edición. Prentice Hall.

Diferencias con los procedimientos:

- En los procedimientos, el conjunto de instrucciones aparece una sola vez en todo el programa y luego se invoca con **CALL.**

- Cada vez que se referencia a una macro, el código que ésta representa se *expande* en el programa definitivo, duplicándose tantas veces como se use la macro. Por lo tanto consumen memoria.

- Los procedimientos economizan memoria y son más eficientes

- Las macros son más rápidas que las subrutinas (se ahorra un CALL y un RET) pero la diferencia es tan mínima que en la práctica es despreciable en el 99,99% de los casos.

USO DE PARÁMETROS EN MACROS

- Cuando se llama a una macro se le pueden pasar opcionalmente un cierto número de parámetros llamados *parámetros actuales*

- En la definición de la macro, los parámetros actuales aparecen asociados a ciertos nombres arbitrarios, cuya única misión es permitir distinguir unos parámetros de otros e indicar en qué orden son entregados: son los *parámetros formales*.

- Cuando el ensamblador expanda la macro al ensamblar, los parámetros formales serán sustituidos por sus correspondientes parámetros actuales.

Referencia: Abel, Meter. LENGUAJE ENSAMBLADOR Y PROGRAMACIÓN PARA IBM-PC Y COMPATIBLES. Tercera Edición. Prentice Hall.

```
; definición de la macro
SUMAR MACRO  a,b,total
          PUSH   AX
          MOV    AX,a
          ADD    AX,b
          MOV    total,AX
          POP    AX
          ENDM

;llamado a la macro
SUMAR positivos, negativos, total
```

- «a», «b» y «total» son los parámetros formales
- «positivos», «negativos» y «total» son los parámetros actuales. Tanto «a» como «b» pueden ser variables, etiquetas, etc. en otro punto del programa; sin embargo, dentro de la macro, se comportan de manera independiente.
- El parámetro formal «total» ha coincidido en el ejemplo y por casualidad con su correspondiente actual.

```
PROMPT  MACRO MENSAJE
        MOV AH, 09H
        LEA DX, MENSAJE
        INT 21H
        ENDM
```

Llamado : **PROMPT Saludo**
Donde : **Saludo DB 'Buenos Dias','$'**

DIRECTIVA LOCAL

Para definir etiquetas de instrucciones dentro de la definición de la macro y ésta es utilizada más de una vez en el mismo programa, es necesario utilizar esta directiva LOCAL, así cada nombre generado es único. Ejemplo:

```
MINIMO          MACRO  dato1, dato2, resultado
        LOCAL  ya_esta
        MOV    AX,dato1
               CMP    AX,dato2 ; ¿es dato1 el menor?
               JB     ya_esta  ; sí
        MOV    AX,dato2 ; no, es dato2
ya_esta:       MOV    resultado,AX
        ENDM
```

- Al invocar la macro dos veces el ensamblador no generará la etiqueta «ya_esta» sino las etiquetas ??0000, ??0001, ... y así sucesivamente.
- Se puede indicar un número casi indefinido de etiquetas con la directiva LOCAL.

```
DIVIDE    MACRO    DIV1, DIV2, COC
          LOCAL COMP
          LOCAL OUT
          MOV AX, DIV1   ; ax: dividendo, bx: divisor,
                         ; cx: cociente
          MOV BX, DIV2
          SUB CX, CX     ;  pone a cero el cociemte
COMP:
          CMP AX, BX     ; dividendo < divisor ?
          JB OUT         ; si, salir
          SUB AX, BX     ; dividendo - divisor
          INC CX         ; sumar al cociente
          JMP COMP
OUT:
          MOV COC, CX    ; almacenar cociente
          ENDM
```

; llamado a la macro
DIVIDE 4,2, res

Referencia: Abel, Meter. LENGUAJE ENSAMBLADOR Y PROGRAMACIÓN PARA IBM-PC Y COMPATIBLES. Tercera Edición. Prentice Hall.

OPERADORES DE MACROS.

Operador ;;

Indica que lo que viene a continuación es un comentario que no debe aparecer al expansionar la macro. Los comentarios relacionados con el funcionamiento interno de la macro deberían ir con (;;). Esto es además conveniente porque durante el ensamblaje son mantenidos en memoria los comentarios de macros (no los del resto del programa) que comienzan por (;), y no conviene desperdiciar memoria...

Operador &

Utilizado para concatenar texto o símbolos. Es necesario para lograr que el ensamblador sustituya un parámetro dentro de una cadena de caracteres o como parte de un símbolo:

```
SALUDO  MACRO cont
        MOV AL,&cont
            ENDM
```

Al ejecutar **SALUDO BL** se producirá la siguiente expansión:

MOV AL,BL

DIRECTIVAS ÚTILES PARA MACROS.

Estas directivas pueden ser empleadas también sin las macros, aumentando la comodidad de la programación, aunque abundan especialmente dentro de las macros.

1. **Repeat**

REPT veces ... ENDM

Permite repetir cierto número de veces una secuencia de instrucciones. El bloque de instrucciones se delimita con ENDM (no confundirlo con el final de una macro). Por ejemplo:

```
REPT   2
OUT  DX,AL
ENDM
```

Esta secuencia se transformará, al ensamblar, en lo siguiente:

```
OUT    DX,AL
OUT    DX,AL
```

Empleando símbolos definidos con (=) y apoyándose además en las macros se puede llegar a crear pseudo-instrucciones muy potentes:

```
SUCESION    MACRO n
        num = 0
         REPT n
          DB num
          num = num + 1
         ENDM  ; fin de REPT
        ENDM  ; fin de macro
```

La sentencia **SUCESION 3** provocará la siguiente expansión:

```
DB     0
DB     1
         DB     2
```

2. Indefinite Repeat

IRP simbolo_control, <arg1, arg2, ..., arg_n> ... ENDM

Es relativamente similar a la instrucción FOR de los lenguajes de alto nivel. Los ángulos (<) y (>) son obligatorios.

El símbolo de control va tomando sucesivamente los valores (no necesariamente numéricos) arg1, arg2, ... y recorre en cada pasada todo el bloque de instrucciones hasta alcanzar el ENDM (no confundirlo con fin de macro) sustituyendo simbolo_control por esos valores en todos los lugares en que aparece:

```
IRP i, <1,2,3>
   DB   0, i, i*i
ENDM
```

Este conjunto de instrucciones se convierte en lo siguiente:

```
DB     0, 1, 1
DB     0, 2, 4
DB     0, 3, 9
```

PUNTOS CLAVE

- Una definición de macro necesita una directiva MACRO, un bloque de una o más instrucciones y una directiva ENDM.
- El código de la macro se “expande” en el programa cada vez que se llama.
- La directiva LOCAL facilita el uso de nombres dentro de una definición de macro y debe aparecer inmediatamente después del enunciado de la macro y tabulado.
- El uso de argumentos en una definición de macro permite codificar parámetros con mayor facilidad.