TEORÍA PRÁCTICA 5

DEFINICIÓN DE MACROS. OPERACIONES ARITMÉTICAS CON NÚMEROS EN DISTINTAS REPRESENTACIONES

OBJETIVOS:

1. Introducir al alumno en el uso de macros, como herramienta que facilita la labor de programación y la legibilidad del código fuentes.

2. Definir procedimientos para la realización de operaciones aritméticas de números representados en los distintos formatos.

3. Practicar con llamadas a macros ya definidas en el emulador.

4. Definir macros propias para realizar operaciones aritméticas auxiliares que agilicen la programación.

Las macros estarán definidas dentro de un archivo que se incluye en el programa ensamblador mediante la directiva INCLUDE 'nombre_archivo.inc'

CONTENIDOS

1.Definición de macros y estructura. Análisis de las macros frente a los procedimientos

2.Uso del fichero de macros del emulador: emu8086.inc

3.Creación y uso de ficheros de macros propias

4.Ejemplos de procedimientos para operaciones aritméticas

EJEMPLOS

Se proponen en el enunciado y serán desarrollados en las 2 primeras sesiones prácticas correspondientes a la Practica 5

TEORÍA PRÁCTICA 6

1.Definición de macros y estructura. Macros frente a procedimientos

Una *Macro* es un trozo de código que se repite cada vez que se llama dentro del programa. La macro o macros están definidas dentro de un programa .inc, que se incluye en el programa ensamblador mediante la directiva *INCLUDE* 'nombre_macro.inc'

ESTRUCTURA DE UNA MACRO

nombre_macro **MACRO** *parámetros*

Cuerpo de la macro

ENDM

Para llamar a una macro es suficiente poner el nombre y los parámetros. La directiva MACRO sirve para definir una macro a través de su nombre. La directiva ENDM indica el fin de la macro. El cuerpo de la macro esta formado por una secuencia de instrucciones de ensamblador que se repite en cada punto del programa en el que la macro sea llamada.

MACROS FRENTE A PROCEDIMIENTOS

Aunque tanto las macros como los procedimientos pueden considerarse una referencia abreviada de un conjunto de instrucciones, no son lo mismo.

DIFERENCIAS

El código de un procedimiento solo aparece una vez en cada programa, aunque dicho procedimiento se llame muchas veces; el procesador transfiere el control a la dirección de comienzo del procedimiento cuando es necesario.

Por el contrario *el código de una macro se repite tantas veces dentro del programa como veces sea llamada dicha macro*; el ensamblador reemplaza el nombre de la macro por las instrucciones que dicha macro representa en el punto del programada donde la macro es llamada.

Las macros presentan algunas ventajas frente a los procedimientos:

1. Las macros son más flexibles. La macro puede ser llamada con diferentes parámetros en cada momento. Tanto para la entrada como para los datos de salida.

MACROS FRENTE A PROCEDIMIENTOS

2. La macros hacen que los programas se ejecuten más rápido, pues no deben ejecutar instrucciones de llamada y de retorno, como sucede en el caso de los procedimientos.

3. Las macros pueden incluirse en una biblioteca de macros, que podrá ser usada para distintas aplicaciones.

Sin embargo las macros tienen un inconveniente, puesto que su código se expande cada vez que se usa, *los programas que utilizan mucho las macros serán más grandes y ocuparán mucho espacio de memoria* con instrucciones repetidas.

Las macros agilizan la programación, ya que la macro se crea una vez y se puede utilizar en un programa tantas veces como sea necesario.

Las macros también agilizan la depuración de los programas, porque las macros se crean y depuran individualmente.

Las macros se usarán normalmente para la realización de operaciones sencillas que se utilizan mucho, no solo en un programa determinado, sino también en otros programas.

Cuando la tarea a realizar sea exclusiva de una aplicación o requiere una extensión de código considerable, es más recomendable el uso de procedimientos.

TEORÍA PRÁCTICA 6

2. Uso del fichero de macros del emulador: emu8086.inc

El emulador del i8086 incluye un fichero de macros denominado ***emu8086.inc***, donde vienen definidas macros para realizar operaciones de transferencia de información desde teclado o hacia pantalla similares a las vistas en la práctica anterior. Por ejemplo la macro *gotoxy* tiene la misma función que el procedimiento *cursor* .

```
cursor   PROC  near  ; posiciona el cursor en los valores en la variables fila , columna
                 PUSH    AX
                 PUSH    BX
                 PUSH  DX
                 MOV     DH, fila
                 MOV     DL, columna
                 MOV     BH,0
                 MOV     AH,2
                 INT     10H
                 POP     DX
                 POP     BX
                 POP     AX
               RET
cursor             endp
```

También incluye una macro para borrar la pantalla.

Si se quiere usar este fichero basta con poner en la cabecera del programa en ensamblador: **INCLUDE** **'emu8086.inc'**

3. Creación y uso de ficheros de macros propias

MACROS PARA EL TRATAMIENTO DE LOS SIGNOS

```
signo MACRO numero, signo     ; extrae el signo  y magnitud de un número en c-2
        local positivo, final
        MOV AX, [numero]
        SHL  AX, 1            ; desplaza a la izq. un bit CF=S
        JNC    positivo       ; salta si no acarreo, nº positivo
        MOV [signo],80h
        RCR    AX,1           ; desplaza a la derecha introduciendo  CF
        NEG    AX             ; realiza el complemento a 2
        JMP final
positivo: SHR AX,1
        MOV [signo], 00h
final:  MOV [numero], AX
          ENDM
```

3. Creación y uso de ficheros de macros propias

MACROS

```
signores MACRO s1,s2,sres          ;Obtener el signo del resultado
        local positivo, final
        MOV AL, [s1]
        MOV AH, [s2]
        XOR AL,AH          ; SI ES CERO EL SIGNO + Y SI ES 1 NEGATIVO
        CMP AL,0           ;POSITIVO
        JZ positivo
        MOV [sres], 2Dh                ; negativo
        JMP final
positivo: MOV [sres],2Bh               ; codigo ascii del +
final:   MOV [signprod], Al
      ENDM
```