Macro (in Turbo Assembler)

Le macro sono molto utili per raggruppare sequenze di istruzioni usate frequentemente, per le quali non è opportuno realizzare una procedura perché magari composte di poche istruzioni che si ripetono più volte nel programma. Le macro sono sostituite al momento della compilazione con il codice che contengono.

Questa è la sintassi per definire una macro:

```
Name_of_macro macro
;
;       a sequence of instructions
;
endm
```

```
Le macro vanno dichiarate all'inizio del codice ,e prima di qualsiasi direttiva.
```

Questi due esempi illustrano le macro per salvare e ripristinare gruppi di registri:

```
SaveRegs macro

push ax
push bx
push cx
push dx

endm

RestoreRegs macro

pop dx
pop cx
pop bx
pop ax

endm
```

Notate che i registri sono prelevati in ordine inverso a quello con cui sono stati salvati nello stack.

Per usare una macro in un programma è sufficiente utilizzare il nome della macro come una qualsiasi istruzione.

```
SaveRegs

; some other instructions

RestoreRegs
```

Questo esempio mostra come si può usare una macro per risparmiare righe di codice. Questa macro stampa semplicemente un messaggio sullo schermo.

```
OutMsg macro SomeText

local PrintMe,SkipData
```

```
jmp SkipData

PrintMe db SomeText,'$'

SkipData:

push ax dx cs

mov dx,OFFSET cs:PrintMe
mov ah,9
int 21h

pop cs dx ax

endm
```

L'unico problema con le macro è che usandole troppo spesso, la dimensione del codice del programma tende ad aumentare e si rischia di incontrare errori di duplicazione delle label e di nomi delle variabili.

Il modo corretto di risolvere questo problema è usare la direttiva LOCAL che consente di dichiarare nomi che valgono solo all'interno delle macro.

Sintassi:

```
LOCAL name
```
Dove name è il nome di una variabile locale o di una label.

Macro con parametri

Un'altra utile proprietà delle macro è che possono avere parametri. Il numero di parametri è limitato solo dalla lunghezza della linea di codice.

Sintassi:

```
Name_of_Macro macro par1,par2,par3
;
; commands go here
;
endm
```
Questo esempio somma il primo e il secondo parametro e pone il risultato nel terzo:
```
AddMacro macro num1,num2,result

push ax         ; save ax from being destroyed
mov ax,num1     ; put num1 into ax
add ax,num2     ; add num2 to it
mov result,ax   ; move answer into result
pop ax          ; restore ax

endm
```

Librerie di macro

La soluzione di collocare in una libreria un certo numero di macro di utilizzo frequente (che il programmatore può invocare tramite un semplice richiamo, evitando così di dover codificare ogni macro esplicitamente all'interno del programma stesso) ottimizza il tempo di sviluppo del programma. Una libreria di macro contiene solo codice sorgente, per cui l'unica operazione da eseguire è quella di salvare il file ASCII, al termine della scrittura delle macro. Per includere nel programma la libreria di macro si utilizza la direttiva:

Include:<path>

bisogna assemblare tutte le volte che si riassembla il programma principale, inserisce tutto il file anche se si usa una sola funzione, non esiste privatezza dei sorgenti.