9. Macroinstructiuni

9.1 Tipuri de macroinstrucţiuni

Macroinstrucţiunile reprezintă secvenţe de program (instrucţiuni, definiţii de date, directive) asociate unui nume.
Folosirea numelui macroinstrucţiunii în program are ca efect înlocuirea acestuia cu secvenţa de program asociată (expandare). Procesul are loc înaintea asamblării propriu-zise.
Un asamblor care implementează macrouri este Macroasamblor. Spre deosebire de proceduri folosirea macrourilor nu duce la micşorarea programului, ci textul sursă devine mai clar şi mai scurt. Folosirea macrourilor implică două etape :

- definirea macroinstrucţiunilor
- apelul (invocarea) macroinstrucţiunilor.

Macroinstrucţiunile pot fi:
- fără parametri
- cu parametri
- repetitive

9.1.1 Definirea macroinstrucţiunii fără parametri

```
Nume_macro MACRO
    ; corp macro
ENDM
```

Mod de utilizare: se scrie în textul sursă numele macroinstrucţiunii.

Exemple:

```
Save    MACRO                       Rest        MACRO
            PUSH AX                             POP     DI
            PUSH BX                             POP     SI
            PUSH CX                             POP     DX
            PUSH DX                             POP     CX
            PUSH SI                             POP     BX
            PUSH DI                             POP     AX
ENDM                                ENDM
```

Utilizare:

```
Some_proc       PROC            NEAR
                Save
                .....
                rest
                RET
Some_proc       ENDP
```

```
Go_dos  MACRO
        MOV     ax,4C00h
        INT 21h
ENDM
```

9.1.2 Definirea macroinstrucţiunilor cu parametri

```
Nume_macro      MACRO   P1,P2,…Pn
        ;
        corp macro
        ;
ENDM
```

Utilizare: `nume_macro  A1,A2,…An ,`

unde identificatorii Pi sunt parametri formali, iar Ai cei actuali.
La invocare pe lângă expandare are loc şi înlocuirea parametrilor formali cu cei actuali.

Exemple: apel de servicii DOS

```
Int_Dos         MACRO   N
                MOV     AH,N
                INT     21h
ENDM
```

Utilizare: `Int_Dos 9`

```
Aduna   MACRO   OP1,OP2,SUMA
        MOV     AX,OP1
        ADD     AX,OP2
        MOV     SUMA,AX
ENDM
```

Utilizare*:* aduna bx,cx,dx

9.1.3 Macroinstrucţiuni repetitive

Sunt predefinite şi generează secvenţe repetitive de program.

```
REPT    N
        ;corp macro
ENDM
```

Exemplu: Secvenţa următoare generează codurile ASCII pentru cifrele 0-9

```
        N=0
        Cifre Label Byte
REPT    10
        DB      '0' +n
        N=n+1
ENDM
```

Repetarea de un număr nedefinit de ori:

```
IRP     p_formal, <lista_param_actuali>
        ;corp macro
ENDM
```

Se repetă de un număr de ori egal cu numărul de elemente conţinut de lista de parametri actuali.

Exemplu:

```
IRP x,<'1','2','3'>    ;se va expanda în DB '1', DB '2', DB '3'
DB x
ENDM
```

Macroinstrucţiunile pot fi păstrate în fişiere separate (fişier de includere) şi folosite în diferite fişiere sursă prin includerea lor folosind directiva INCLUDE cu sintaxa:

INCLUDE identif_fisier

identif_fisier este un fişier de includere care conţine instrucţiuni corecte acceptate de asamblor.
De obicei fişierele de includere conţin macrouri, echivalări sau definiţiii standard de segmente.

```
INCLUDE fct.inc          ; specificator de fişier
INCLUDE c:\libs\seg.inc  ; specificator complet (calea)
INCLUDE ..\libs\tim.inc
```

Exemplu: c:\tasm\libs\ timer.inc conţine proceduri specifice timer

```
                ……
                include         ..\libs\timer.inc
seg_program     ends
                end     start
```

9.2 Exerciţii şi teme

1. Scrieţi un program complet care determină maximul dintr-un şir.

a. Definiţi un şir de 10 octeţi iniţializaţi cu numere aleatoare;
b. Definiţi o variabilă “maxim” pe octet;
c. Scrieţi o secvenţă de program care determină maximul dintr-un şir;
d. Scrieţi sursa (nume.asm);
e. Asamblaţi aplicaţia; generaţi şi listingul;
f. Linkeditaţi aplicaţia;
g. Executaţi programul cu td.exe.

2. După modelul prezentat, scrieţi un program care determină minimul dintr-un şir de 10 numere.

3. Scrieţi un program care calculează suma elementelor unui şir de numere. Parcurgeţi şirul în două moduri: prin adresare bazată indexată şi folosind instrucţiuni specifice şirurilor.

4. Dezvoltaţi programele pentru reuniunea, intersecţia şi diferenţa a două şiruri.

5. Scrieţi aplicaţiile care ordonează un şir de 10 elemente, definite ca octeţi în memorie, crescător (descrescător) considerând elementele ca numere fără semn şi cu semn.

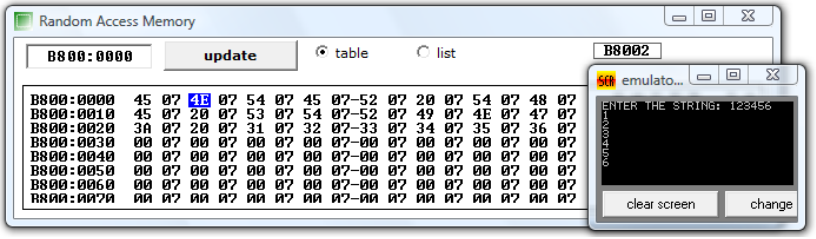
9.3 Probleme rezolvate, folosind macroinstructiuni

1. Sa se afiseze pe ecran valoarea din registru AL in formatele: zecimal fara semn, respectiv binar (de exemplu pentru AL=0FEH pe ecran se va tipari pe prima linie *254* iar pe cea de-a doua linie *11111110b*)

```
name "printAL"
org 100h

mov al, 0FEh
call print_al       ; afiseaza pe prima linie valoarea din Al
                    ; in format zecimal fara semn
call print_nl       ; trece la urmatoarea linie
call print_al_bin   ; afiseaza pe a doua linie valoarea din Al
                    ; in format binar
ret
print_al proc
    cmp al, 0
    jne print_al_r
    push ax
    mov al, '0'
    mov ah, 0eh
    int 10h
    pop ax
    ret
print_al_r:
    pusha
    mov ah, 0
    cmp ax, 0
    je pn_done
    mov dl, 10
    div dl
    call print_al_r
    mov al, ah
    add al, 30h
    mov ah, 0eh
    int 10h
    jmp pn_done
pn_done:
    popa
```

```
    ret
endp

print_al_bin proc
    pusha
    mov cx, 8
    mov bl, al
p1: mov ah, 2
    mov dl, '0'
    test bl, 10000000b
    jz zero
    mov dl, '1'
zero:  int 21h
    shl bl, 1
    loop p1
    mov dl, 'b'
    int 21h
    mov dl, 0Dh
    int 21h
    mov dl, 0Ah
    int 21h
    popa
    ret
endp

print_nl proc
    push ax
    push dx
    mov ah, 2
    mov dl, 0Dh
    int 21h
    mov dl, 0Ah
    int 21h
    pop dx
    pop ax
    ret
endp
```

2. Următorul program preia un şir de la tastatură si apoi afişează pe ecran elementele şirului, câte unul pe linie:



```
name "charchar"
org  100h

print_new_line macro
    mov dl, 13
    mov ah, 2
    int 21h
    mov dl, 10
    mov ah, 2
    int 21h
endm

    mov dx, offset msg1
    mov ah, 9
    int 21h
    mov dx, offset s1
    mov ah, 0ah
    int 21h

    xor cx, cx
    mov cl, s1[1]
    print_new_line

    mov bx, offset s1[2]
print_char:
    mov dl, [bx]
    mov ah, 2
    int 21h
    print_new_line
    inc bx
    loop print_char

    mov ax, 0
    int 16h

    ret

msg1    db  "ENTER THE STRING: $"
s1      db 100,?, 100 dup(' ')
end
```