SUPORT SEMINAR 2 - ASSEMBLER

MODURI DE ADRESARE

ADRESARE DIRECTĂ

Registrele sunt inițializate cu valori constante
Exemplu:

```
mov AX, 1234h
```

Numele variabilei este utilizat ca operand in instrucțiune.
Exemplu:

```
vector dw 10 dup(5)   ; definire vector cu 10 elemente inițializate cu valoarea 5
…
mov  AX, vector       ;copiază valoarea primului element de tip cuvânt din vector în AX
mov BX, vector[3]     ;copiază valoarea cuvântului care începe de la offset-ul al treilea
                      ;ATENTIE, cum elementele vectorului încep de la offset cu
                      ;valoare multiplu de 2, rezultă că în BX se copiază valoarea 0500h.
```

Vectorul numit *vector* se găseşte în segmentul de date memorat astfel:

DS:0000 **05 00 05 00 05 00 05 00 05 00**
DS:0008 **05 00** 00 00 00 00 00 00 00 00

iar instrucțiunea mov BX,vector[3] copiază în BX 2 octeți (deoarece vector conține elemente de tip *word*) însă începând de la offset-ul 0003 din DS. Deci BX va conține valoarea 0500h.

Valoarea offset-ului este utilizată pentru a citi date direct din segmentul DS.
Exemplu:

```
vector dw 10 dup(5)   ; definire vector cu 10 elemente inițializate cu valoarea 5
…
mov  AX, DS:[00h]     ;copiază valoarea primului element de tip cuvânt din vector în AX
mov BX,DS:[08h]       ;copiază valoarea cuvântului care începe de la offset-ul 08h
```

Când se specifică offset-ul este absolut necesar să se utilizeze expresia DS:[valoare_offset] pentru că, dacă se scrie instrucțiunea:

```
mov AX,[08h]
```

se copiază în AX doar valoarea 8h (Veți primi si un WARNING – *[Constant] assumed to mean immediate constant*.)

ADRESARE INDIRECTĂ

Se utilizează unul dintre registrele index SI, DI, BP sau registrul de bază BX. Atenție: utilizarea altor registre va genera eroare: **Illegal indexing mode**.
Se utilizează în general pentru parcurgerea masivelor şi pentru a accesa elemente din structuri de date de tip articol.

Utilizarea registrului index SI sau DI

- registrul SI este utilizat pentru a reține offset-ul elementului pe care dorim să-l accesăm;

```
vector dw 1234h,1235h,3222h,4343h,5455h
…
mov SI, offset vector  ;încarcă în SI offset-ul de început al vectorului
mov AX, [SI]           ;copiază în AX valoarea primului element, adică 1234h
mov CX, [SI+2]         ;copiază în CX valoarea elementului al doilea, adică 1235h
mov DX,[SI+3]          ;copiază în DX valoarea 2212h (octetul inferior din 3222h şi
                       octetul superior din 1235h)
```

acelaşi lucru se scrie şi:

```
mov SI, offset vector
mov AX, [SI]
mov CX, [SI][2]
mov DX,[SI][3]
```

- registrul SI sau DI este utilizat asemenea unui indice în parcurgerea masivului;

```
vector db 1,2,3,4,5,6
…
XOR SI,SI              ; ne asigurăm că SI are valoarea 0
mov AL, vector[SI]     ;copiază în AL valoarea primului element din vector
inc SI
mov AH, vector[SI]     ;copiază în AH valoarea elementului secund din vector
mov CL,vector[SI+1]    ;copiază în CL valoarea celui de al treilea element
mov AL,vector[SI][2]   ;copiază în AL valoarea celui de al patrulea element deoarece SI
                       conține valoarea 1
```

în cazul în care vectorul are elemente de tip *word* parcurgerea acestuia se face mărind SI cu valoarea 2 de fiecare dată:

```
vector dw 1243,2342,3342,44324,53242,63432
…
XOR SI,SI               ; ne asigurăm că SI are valoarea 0
mov AX, vector[SI]      ;copiază în AX valoarea primului element din vector
inc SI
inc SI
mov AX, vector[SI]      ;copiază în AX valoarea elementului secund din vector
mov CX,vector[SI+2]     ;copiază în CX valoarea celui de al treilea element
mov AX,vector[SI][4]    ;copiază în AX valoarea celui de al patrulea element deoarece SI
                         conține valoarea 2
```

Utilizarea registrului bază BX

- registrul BX este utilizat pentru adresare indirectă bazată;

```
vector db 1,2,3,4,5,6
…
XOR SI,SI               ; ne asigurăm că SI are valoarea 0
mov BX, offset vector;încărcăm în registrul de bază offset-ul de start al vectorului
mov AL, [BX][SI]        ;copiază în AL valoarea primului element din vector
inc SI
add AL, [BX+SI]         ;adună în AL valoarea elementului secund din vector
add AL, [BX+SI+1]       ;adună în AL valoarea celui de al treilea element
add AL,[BX][SI][2]      ;adună în AL valoarea celui de al patrulea element deoarece SI
                         conține valoarea 1
```

In exemplele anterioare, SI poate fi înlocuit de DI.
Registrul BX este utilizat şi pentru a transmite offset-ul unui articol sau masiv ce reprezintă parametru de intrare într-o procedură. Caracterul de generalitate al unei proceduri impune lucrul în interiorul ei cu offset-ul parametrilor de intrare şi nu cu numele lor.

Când se utilizează registrul BX ca registru de bază în adresările indirecte pentru a memora offset-ul primului octet al structurii respective, adresa segmentului este dată de registrul DS. Deci, fiecare element identificat prin [BX] se află memorat la adresa DS:BX (segment:offset).
In cazul în care se utilizează registrul BP pentru adresări indirecte, adresa segmentului este dată de registrul SS. Deci BP este utilizat pentru a accesa indirect locații de memorie din stivă. Din acest motiv, instrucțiunea

```
mov CX,[BP]
```

copiază în CX, valoarea din stivă aflată la offset-ul dat de BP, adică SS:BP.
Pentru a forța citirea din segmentul DS la offset-ul dat de BP, instrucțiunea devine

```
mov CX,DS:[BP]
```

Instrucțiunea LOOP

- forma analitică a instrucțiunii este:

```
LOOP nume_etichetă
```

- decrementează registrul CX şi sare la eticheta *nume_etichetă* cu condiția ca valoarea din CX să fie mai mare decât zero;
- nu afectează nici un flag;
- instrucțiunea memorează într-un octet offset-ul etichetei față de poziția sa în segmentul de cod; deci, eticheta trebuie să se găsească la o distanță de maxim128 de octeți în interiorul segmentului de cod față de poziția instrucțiunii LOOP;
- utilizată de obicei pentru a implementa instrucțiuni de control repetitive;

Exemplu: determinarea sumei elementelor unui vector.

```
.model small
.286
.stack 100h
.data
        vector db 1,2,3,4,5,6,7
        n db 7                  ;dimensiunea vectorului
        suma db 0               ;suma elementelor

.code
        mov AX,@data
        mov DS,AX

        xor SI,SI
        xor CX,CX               ;ne asigurăm că CX are valoarea 0
        mov CL,n                ;copiem în CL dimensiunea vectorului
repeat:                         ;definim eticheta
        mov AL,vector[SI]
        add suma,AL
        inc SI                  ;mărim valoarea din SI cu 1 pentru a trece la
                                ;elementul următor
        loop repeat             ;salt la etichetă cât timp CX diferit de 0
        mov AX,4c00h
        int 21h
end
```

Instrucțiunea LEA

- forma analitică a instrucțiunii este:

LEA registru, operand

- încarcă în registrul specificat offset-ul operandului;
- registrul nu poate fi unul dintre registrele de segment;
- exemplu:

```
.data
        vb dw 1234h,1235h
.code
        …
        mov SI,offset vb        ;încarcă registrul SI cu valoarea offest-ului variabilei vb
        lea DI,vb               ;încarcă registrul DI cu valoarea offest-ului variabilei vb
        …
```

CITIREA ŞI AFISAREA PE ECRAN

Citirea unui caracter de la tastatură cu ecou.

- se realizează utilizând rutina DOS 21h;
- registrul AH are valoarea 01h;
- codul ASCII al caracterului citit este pus în registrul AL;
- transformarea caracterului citit într-un număr (cu condiția ca acesta să reprezinte o cifră) cu valoarea cuprinsă între 0 şi 9 se face scăzând din AL valoarea 30h, ce reprezintă codul ASCII al caracterului 0;
- exemplu:

```
.data
        nr db ?

.code
        …
        mov ah,01h      ;inițializare AH cu codul funcției
        int 21h         ;apel rutină
        sub al,30h      ;transformare cod ASCII aferent cifrei în valoare întreagă
        mov nr,al       ;copiere valoare număr
        …
```

Afişarea unui caracter pe ecran.

- se realizează utilizând rutina DOS 21h;
- registrul AH are valoarea 02h;
- codul ASCII al caracterului citit este pus în registrul DL;
- afişarea unei valori numerice implică transformarea cifrelor în echivalentul lor ASCII (se adaugă la valoarea cifrei valoarea 30h, ce reprezintă codul ASCII al caracterului 0);
- exemplu:

```
.data
        nr db 5
        caracter db 'g'

.code
        …
        mov DL,caracter
        mov ah,02h      ;inițializare AH cu codul funcției
        int 21h         ;apel rutină şi afişare caracter g pe ecran
        mov DL,nr
        mov ah,02h
        int 21h         ;apel rutină şi afişare pe ecran caracter cu codul ASCII egal cu 5

        mov DL,nr
        add, DL,30h     ;obținere cod ASCII al cifrei din DL
        mov ah,02h
        int 21h         ;apel rutină şi afişare pe ecran a caracterului 5
        …
```

Afişarea unei valori numerice mai mari decât 9 pe ecran implică prelucrări prin intermediul cărora să se obțină codul ASCII al tuturor cifrelor din număr.

Afişarea unui şir pe ecran

- se realizează utilizând rutina DOS 21h;
- registrul AH are valoarea 09h;
- adresa de început a şirului de caractere trebuie să se găsească în DX; acest lucru se realizează prin intermediul instrucțiunii LEA sau a instrucțiunilor LES / LDS (descrise în Seminar 1)
- **Important**: ultimul caracter din şir trebuie să fie $ (24h); funcția afişează şirul până întâlneşte primul caracter $
- exemplu:

```
...
.data
      mesaj   db "Afisare mesaj !!!","$"
.code
      ...
   mov   AH,09
   lea   DX,mesaj
   int   21h
      ...
```

Citirea unui şir de la tastatură

- se realizează utilizând rutina DOS 21h;
- registrul AH are valoarea 0Ah;
- adresa din segmentul de date (offset-ul) în care se face memorarea şirului citit trebuie scrisă în DX; şirul citit de la tastatură se scrie la adresa DS:DX

- primul octet de la adresa DS:DX trebuie să conțină numărul maxim *n* de caractere al şirului (rutina permite citirea de la tastatură doar a *n-1* caractere); neinițializarea primului octet cu valoarea *n* poate conduce la situațiile: acesta are valoarea 0 şi nu se citesc caractere sau acesta are o valoarea reziduală mai mare / mică decât lungimea maximă dorită;
- introducerea şirului la tastatură se încheie cu Enter (0Dh);
- rutina va scrie în al doilea octet de la adresa DS:DX numărul de caractere efectiv citit de la tastatură;
- ultimul octet din şir este codul ASCII al tastei Enter (0Dh)
- exemplu:

```
.data
        vb dw 1234h,1234h,1234h
        sir db 5        ;valoare necesară pentru a indica numărul maxim de caractere de
                        citit
.code
        mov AX,@data
        mov DS,AX

        mov dx,0006h ;încarc în DX offset-ul la care începe scrierea şirului de caractere
        mov ah,0ah
        int 21h

        mov AX,4c00h
        int 21h
end
```