Platformă de e-learning și curriculă e-content pentru învățământul superior tehnic

Programare în limbaj de asamblare

43. Elaboraea de programe simple şi depanarea lor, utilizând instrucțiuni de transfer, sub Debug/ TD.

Editarea şi execuția unor programe simple în *Debug*.

Interschimbarea a două locații de memorie

(3000h:200h şi 3000h:300h, respectiv, de tip cuvânt).

Adresele absolute propuse, 3000h:200h, pot fi modificate dacă nu sunt disponibile (de fapt este vorba de modificarea adresei de segment, 3000h, cu o adresă superioară)

a) *Utilizarea utilitarei Debug*

Pentru exemple simple, compuse doar din câteva instrucțiuni, este mult mai simplu să utilizăm *Debug*-ul care permite atât editarea cât şi execuția acestor programe. Editorul este unul foarte redus şi simplu, întrucât nu permite corectarea liniilor scrise, decât prin reeditarea lor; mai mult, liniile următoare celor eronate, care au fost reeditate trebuie rescrise şi ele, întrucât pe masura editării liniile sunt translatate în cod maşină şi depuse în memorie peste codul anterior editat, astfel încât după liniile corectate vor urma 'fragmente' din liniile următoare celor corectate, în funcție de numărul de octeți ocupat de liniile reeditate.

Singura situație în care nu este necesară reeditarea liniilor următoare, celor corectate, este cea în care numărul de octeți al instrucțiunilor reeditate este egal cu numărul de octeți avut de aceste instrucțiuni, înainte de reeditare (dar probabilitatea este destul de mică).

Revenind la problema propusă, secvența de comenzi necesară este următoarea:
- se lansează, mai întâi utilitara Debug:

C:\> **Debug**

- după care se intră în modul de comandă al acesteia, şi se dă comanda de asamblare:

- a2000:100
2000:0100 mov ax,3000

Textul scris supraimprimat reprezintă comenzile introduse de utilizator, restul (adresa, scrisă cursiv) reprezentând răspunsurile utilitarei Debug. Editarea unei linii se termină prin apăsarea tastei RETURN; până atunci se pot face corecții ale textului scris, utilizând tastele de deplasare (săgețile) şi tastele *DEL, BS*. În continuare, după fiecare linie introdusă, care este imediat asamblată, asamblorul afişează locația de memorie a următoarei linii, pe care pentru a nu îngreuna urmărirea programului am omis-o (dar ea este permanent afişată). deci programul propriu-zis este următorul:

```
mov   ax,3000
mov   ds,ax
mov   ax,[200]
xchg  ax,[300]
mov   [200],ax
mov   ax,4c00
int   21
```

Reamintim ca pentru asamblorul *Debug* toate datele imediate sunt interpretate implicit în hexazecimal, deci nu este nevoie să punem după o valoare numerică tipul '*h'*, acesta fiind implicit; mai mult chiar dacă din greşeală tastaţi '*h*'-ul asamblorul vă va semnala o eroare pe linia respectivă şi continuă editarea de la aceeaşi adresă, ca de altfel pentru orice altă eroare.

Cât timp se afişează adresa locației următoare sunteți în modul 'editare'. Pentru a executa programul trebuie să părăsiți modul de editare (asamblare) şi să treceți în modul comandă, prin tastare RETURN pe o linie nouă, următoarea celei ce reprezintă ultima instrucțiune din program.

Pentru execuție se poate opta pentru execuția pas cu pas, dacă programul nu este prea lung şi doriți să vedeți ce se întâmplă după execuția fiecărei instrucțiuni, sau dacă doriți să depanați programul. Comanda pentru execuție pas cu pas este, în acest caz:

-T2000:100

care va executa doar prima instrucțiune, după care pentru execuția următoarelor instrucțiuni se va tasta, în mod repetat pentru fiecare instrucțiune, doar:

-T

Dacă se doreşte execuția integrală a programului se va utiliza comanda G:

- **G** = adresa de start (segment:offset), adresa de sfârşit (offset-ul)

adresa de sfârşit = prima adresă, furnizată de asamblor, după cea a ultimei instrucțiuni din programul editat.

Pentru vizualizarea rezultatelor şi introducerea datelor se vor utiliza comenzile: D, E, F, descrise în capitolul precedent.

b) Editare utilizând un editor de text.

Pentru editare se poate folosi orice editor de text disponibil, cum ar fi editoarele din mediile C, Pascal, editorul din NC (cel mai simplu şi mai direct de folosit), şi altele. Programul editat, cu un astfel de editor, este următorul:

```
prog    segment         word    public  'code'
        assume cs:prog
start:  mov     ax,3000h        ; adresa de segment unde vor fi datele introduse din Debug
        mov     ds,ax
        mov     ax,[200]        ; adresele relative ale celor doi operanzi
        xchg    ax,[300]
        mov     [200],ax
        mov     ax,4C00h        ; revenire in DOS
        int     21h
prog    ends
end     start
```

După editare programul va fi salvat într-un fişier cu extensia '*asm*': intersch.asm. Pentru obținerea fişierului obiect se va apela utilitara TASM:

C:\>**tasm intersch.asm**

care va furniza, dacă nu sunt erori, fişierul obiect: intersch.obj. Următoarea etapă este cea de editare de legături, utilizând utilitara TLINK:

C:\>**tlink intersch.obj**

care va furniza fişierul executabil: *intersch.exe*.

Acesta poate fi direct executat, dar pentru a vedea efectele sale el va fi rulat sub controlul Debugger-ului:

C:\>**debug intersch.exe**

Comanda de lansare este aceeaşi ca în cazul anterior. Pentru a vizualiza programul se va utiliza comanda U, descrisă în primul capitol.

Pentru exemplele următoare se vor prezenta doar programele, fără detaliile de editare şi etapele necesare execuției programelor, întrucât ele au fost prezentate pe larg pentru acest prim exemplu, dar şi în primul capitol.

Interschimbarea a două locații succesive de memorie

(3000h:200h şi 201h, respectiv, de tip octet).

Pentru astfel de situații (informații aflate la locații succesive de memorie) se va utiliza pentru adresare un registru index (SI sau DI) sau de bază (BX, de regulă sau BP, dar acesta face referire implicit la segmentul de stivă, referit de SS)

a) Utilizarea utilitarei Debug (aceasta nu permite utilizarea de comentarii).

```
mov   ax,3000
mov   ds,ax
mov   si,200
mov   al,[si]
inc   si
xchg  al,[si]
dec   si
mov   [si],ax
mov   ax,4c00
int   21
```

b) Editare utilizând un editor de text.

```
prog    segment         word    public  'code'
        assume cs:prog
start:  mov     ax,3000h        ; adresa de segment unde vor fi datele introduse din Debug
        mov     ds,ax
        mov     si,200h         ; adresa relativa a primului octet
        mov     al,[si]         ; citirea primului octet
        inc     si              ; pozitionare pe cel de-al doilea octet
        xchg    al,[si]         ; interschimbare (AL) <-> al doilea octet
        dec     si              ; revenire la adresa primului octet
        mov     [si],al         ; depunerea celui de-al doilea octet, deci interschimbarea
        mov     ax,4C00h        ; revenire in DOS
        int     21h
prog    ends
end     start
```

<u>Adunarea a două locaţii succesive de memorie</u>

(3000h:200h şi 202h, respectiv, şi depunerea rezultatului în continuare, după cei doi operanzi, de tip cuvânt, fără semn).

a) Utilizarea utilitarei Debug.

```
        mov     ax,3000
        mov     ds,ax
        mov     si,200
        mov     ax,[si]
        add     si,2
        add     ax,[si]
        add     si,2
        mov     [si],ax
        mov     ax,4c00
        int     21
```

b) Editare utilizând un editor de text.

```
prog    segment         word    public  'code'
        assume cs:prog, ds:date
start:  mov     ax,3000h        ; adresa de segment unde vor fi datele introduse din Debug
        mov     ds,ax
        mov     si,200h         ; adresa relativa a primului cuvant
        mov     ax,[si]         ; citirea primului cuvant
        add     si,2            ; pozitionare pe cel de-al doilea cuvant
```

```
        add     ax,[si]         ; (AX) = suma celor doi octeti
                                ; aici se poate testa indicatorul (CF) pentru depasire rezultat
        jc      afis_mesaj      ; (CF) = 1 ? , daca da -> mesaj depasire
                                ; numerele sunt fara semn, deci rezultatul este > 65535
                                ; daca consideram numerele cu semn, atunci rezultatul va fi
                                ; in intervalul [-32768,+32767], pentru detectarea depasirii
                                ; limitei superioare sau inferioare in locul instructiunii 'jc'
                                ; se va utiliza instructiunea 'jo' (Jump Overflow)
        add     si,2            ; adresa rezultatului
        mov     [si],ax         ; depunerea celui de-al doilea octet, deci interschimbarea
rev_DOS:
        mov     ax,4C00h        ; revenire in DOS
        int     21h
afis_mesaj:
        lea     dx,mes_err      ; afisarea mesajului de eroare
        mov     ah,9
        int     21h
        jmp     rev_DOS
prog    ends

date    segment
mes_err         db      'eroare: depasire dimensiune cuvant a rezultatului'
date    ends
end     start
```

<u>Adunarea elementelor unui sir de valori</u> (de tip cuvânt, cu semn).

a) Utilizarea utilitarei Debug

Se adună 16 cuvinte începând de la adresa 3000:200, cu rezultat de lungime 16 biți, fără detectarea depășirii.

```
        mov     ax,3000
        mov     ds,ax
        mov     si,200
        mov     cx,10
        mov     ax,0
        add     ax,[si]
        add     si,2
        loop    adresa_add_ax_si
        mov     si,300
        mov     [si],ax
        mov     ax,4c00
        int     21
```

Programul va depune rezultatul la adresa 3000h:300h. În locul adresei 'adresa_add_ax_si' se va pune adresa fizică a instrucțiunii '*add ax,[si]*', furnizată de asamblor, şi nu o etichtă simbolică, cum este prezentată (de altfel nici nu se poate în acest asamblor).

b) Editare utilizând un editor de text.

Se adună un număr oarecare de cuvinte şi se semnalează o eventuală depăşire a formatului inițial de 16 de biți.

```
date    segment
sir     dw      0f54h, 20000, 0ff56, 8000 ; si sirul poate continua
lung_sir        equ     ($-sir)/2
rezultatdw      2 dup (?)
mes_err         db      'eroare: depasire dimensiune cuvant a rezultatului'
date    ends
end     start

prog    segment         word    public  'code'
        assume cs:prog, ds:date
start:  mov     ax,date         ; adresa de segment pentru date
        mov     ds,ax
        lea     si,sir          ; adresa relativa a primului cuvant
        mov     ax,0            ; initializare rezultat
        mov     cx,lung_sir     ; contor numar de cuvinte de insumat
reia_add:
        add     ax,[si]         ; adunare cuvant curent
        jo      afis_mesaj      ; (CF) = 1 ? , daca da -> mesaj depasire superioara/inferioara
        add     si,2            ; pozitionare pe cel de-al doilea cuvant
        loop    reia_add        ; daca nu s-a terminat sirul, reia adunarea, cu urmatorul cuvant
        mov     rezultat,ax     ; memorare rezultat
rev_DOS:
        mov     ax,4C00h        ; revenire in DOS
        int     21h
afis_mesaj:
        lea     dx,mes_err      ; afisarea mesajului de eroare
        mov     ah,9
        int     21h
        jmp     rev_DOS
prog    ends
```