

AMP1 - Anexă Aplicații

Programul 1.

Fiind date două șiruri de elemente octeți cu semn, cu același număr de elemente, să se interschimbe elementele ce le alcătuiesc. Ambele șiruri sunt terminate cu 0h. Se vor folosi două proceduri, SLen și SXch.

Procedura SLen calculează lungimea unui șir terminat cu 0h :

primește: în ES :DI adresa logică de început a șirului ;

întoarce: CF = 1, caz în care în CX se va afla lungimea șirului ;

CF = 0 indică faptul că lungimea șirului depășește $2^{16}-2$.

Procedura SXch interschimbă element cu element elementele șirului :

primește: în DS:SI adresa logică de început a șirului 1;

în ES:DI adresa logică de început a șirului 2;

întoarce: CF = 1/0 semnifică interschimbare reușită/nereușită.

Soluție:

Procedura SLen: numărarea elementelor unui șir ce este terminat cu 0h constă în parcurgerea sa și incrementarea unui contor (ad-hoc) pentru fiecare element diferit de 0h. Aceasta se poate realiza prin folosirea grupului `repnz scasb`. Pentru aceasta, CX va fi registrul contor (implicit pentru `repnz`), AL va conține numărul care indică sfârșitul șirului (în cazul nostru, 0h), DI va conține adresa efectivă a primului element al șirului, iar DF trebuie resetat (pentru a parcurge crescător șirul). Primitiva `repnz` are ca efect decrementarea lui CX, pentru fiecare execuție a `scasb`. Valoarea contorului ce va fi folosit trebuie să plece din 0h și să fie incrementată. Transformarea `inc CX`, urmată de `not CX` echivalează acestei operații. La început CX este inițializat cu FFFFh, ceea ce prin succesiunea de transformări amintită mai sus ar însemna 0h, iar prin decrementarea lui FFFFh se obține incrementarea valorii `{inc CX | not CX}`.

Procedura SXch: apelează de două ori procedura SLen, salvând în BX și CX dimensiunile celor două șiruri, după care interschimbă elementele celor două șiruri.

Textul programului :

```
org      100h

jmp     start
```

```

sir1      db      'Exemplu de sir ASCII',0
sir2      db      'EXEMPLU DE SIR ASCII',0

start:    mov      ax,cs
          mov      ds,ax
          mov      es,ax
          mov      si,offset sir1
          mov      di,offset sir2
          call     SXch
          int      20h

SLen :    cld
          mov      cx,-1      ;CX ← 2^16-1 octeti
          mov      al,0      ;AL ←0, termiator sir
          repnz    scasb     ;cauta terminator sir
          jnz     SL1
          inc      cx
          not      cx        ;lungime sir in CX
          stc
          ret

SL1:      clc

SXch:     push     es
          push     di        ;salvare adresa sir2
          call     SLen

```

```

    jnc     Sxer     ;sir2 prea lung
    mov     bx,cx    ;lungime sir2
    mov     ax,ds
    mov     es,ax
    mov     di,si
    call    SLen     ;lungime sir1
    jnc     Sxer     ;sir1 prea lung
    cmp     bx,cx
    jne     Sxer     ;siruri de lungimi diferite
    pop     di
    pop     es       ;refacere adresa sir2
    jcxz    SX2     ;siruri vide
    cld

SX1:    lodsb
        xchg    al,[es:di]
        inc     di
        mov     [si-1],al
        loop   SX1     ;bucla

SX2:    stc

Sxer:   pop     di
        pop     es     ;descarcare stiva
        cld
        ret          ;terminare procedura

```

end

Programul 2.

Fiind dat un șir terminat cu 0h, să se elimine j elemente începând din poziția i . Se vor folosi procedurile SLen și SCut.

Procedura SLen calculează lungimea unui șir terminat cu 0h:

primește: în ES:DI adresa logică de început a șirului;

întoarce: CF = 1, caz în care în CX se va găsi lungimea șirului;

CF = 0 indică faptul că lungimea șirului $> 2^{16} - 2$;

alterează: AL, CX, DI.

Procedura SCut elimină elementele din șir:

primește: în DS:SI adresa logică de început a șirului;

în DX poziția i ;

în BX - numărul j de elemente;

întoarce: CF = 0 indică faptul că lungimea șirului $> 2^{16} - 2$.

Soluție:

Șirul fiind terminat în 0h, nu cunoaștem a priori lungimea lui. Îi vom calcula lungimea căutând simbolul terminal, 0h. Procedura SLen este descrisă în soluția programului anterior.

Procedura SCut apelează SLen pentru a afla dimensiunea șirului. Verifică dacă șirul are mai mult de i elemente ($CX-DX > 0$) și dacă mai rămân elemente de mutat ($CX-DX-BX > 0$). Prin `rep movsb` suprascriem elementele de la poziția i până la $i+j$. În final se copiază simbolul terminal 0h.

Textul programului:

```
org    100h

jmp    start

sir1   db    'Exempplu de sir de caractere ASCII',0h

start:  mov    ax,cs
```

```

mov     ds,ax

mov     si,offset sirl

mov     dx,3           ;incepand cu al treilea,
mov     bx,5           ;se elimina 5 elemente

call    SCut

int     20h

SLen:   cld

mov     cx,-1          ;CX<- 2^16-1 octeti
mov     al,0           ;AL<- 0, terminator sir
repnz   scasb          ;cauta terminator sir
jnz     SL1           ;sir prea lung

inc     cx

not     cx             ;lungime sir (v. ex. anterior)

stc

SL1:    clc

ret

SCut:   cld

push   ds

pop     es

mov     di,si          ;adresa de inceput a sirului

call    SLen           ;CX = lungime sir

jnc     SC2           ;sir prea lung

add     si,dx

dec     si

mov     di,si          ;adresa primului element ce va fi
                        ;eliminat

```

```

sub    cx,dx

jb     SC2          ;pozitia > lungimea sirului

add    si,bx       ;adresa primului element ce va fi
                    ;mutat

sub    cx,bx

jb     SC1          ;nu mai raman elemente de mutat

sub    byte ptr [bx],dh

                    ;corectie lungime sir

inc    cx          ;numarul de elemente de mutat

rep    movsb

SC1:   mov    al,0

        stosb          ;terminator sir

        stc            ;CF <- 1

SC2:   cld            ;CF <- 0

        ret

        end

```

Programul 3.

Fiind dat un șir de cuvinte, să se elimine elementele care se repetă din acest șir. Se va folosi procedura ELIM.

Procedura ELIM:

primește: în DS:SI - adresa logică a primului element din șir;

în CX - numărul elementelor din șir;

întoarce: în DX - numărul elementelor rămase în șir.

Soluție:

Procedura ELIM aduce un element din șir în acumulator și îl compară cu restul elementelor din șir. În momentul în care s-a găsit un alt element egal cu el se oprește comparația și îl elimină pe acesta prin mutarea elementelor șirului cu o poziție la stânga. Se mai caută o dată. Doar dacă nu a mai fost găsit, se trece la următorul element. La fiecare eliminare este decrementat DX (numărul de elemente din șir).

Programul principal pregătește contextul cerut de apelul procedurii ELIM (încarcă în DX numărul elementelor șirului și în SI adresa de început a șirului), apelează ELIM și întoarce în DX numărul elementelor ramase în șir.

Textul programului:

```
                org     100h

                jmp     start

sir             dw     1,1,2,0,3,0,3,3,7,3

ssir           equ     ($-sir)/2

lsir           dw     offset ssir

start:         mov     ax,cs

                mov     ds,ax

                mov     si,offset sir

                mov     cs,ssir

                call    elim

                mov     lsir,dx

                int     20h

elim:          push    ds

                pop     es

                mov     dx,cx           ;numarul de elemente ramase in sir

                dec     cx

                mov     bx,cx           ;numarul de elemente ramase de
                                        ;comparat

et1:           jcxz    et4

                lodsw

                mov     di,si

                repne   scasw           ;compara cu restul sirului

                jne     et3           ;trece la urmatorul element din sir
```

```

        jcxz    et2            ;nu mai sunt elemente de mutat
push    si
mov     si,di
sub     di,2
mutarea rep     movsw        ;elimina elementul repetat, prin
                                ;restului sirului
pop     si
et2:    sub     si,2
        dec     dx            ;numarul elementelor ramase in sir
et3:    dec     bx            ;numarul de elemente ramase de
                                ;comparat
        mov     cx,bx        ;devin contor al iteratiilor
        jmp     et1
et4:    ret
        ret
        end

```

Programul 4.

Fiind dată o matrice patrată, să se verifice dacă este simetrică. Se vor folosi procedurile ELEM și SIMM.

Procedura ELEM întoarce elementul (i,j) din matrice:

primește: în DL - dimensiunea matricii;

în CL, CH - numărul liniei (i) respectiv coloanei (j) ;

în SI - adresa efectivă a primului element al matricii;

întoarce: în AL - elementul (i,j) al matricii.

Procedura SIMM verifică dacă matricea este simetrică:

primește: în DL - dimensiunea matricii;

în SI - adresa efectivă a primului element al matricii;

întoarce: CF = 1/0 pentru matrice simetrică/nesimetrică.

Soluție:

Procedura ELEM: pentru că elementele matricii sunt dispuse pe linii și sunt scrise pe câte un octet, adresa relativă a elementului de pe linia i și coloana j față de adresa de început a matricii este $n*(i - 1) + j - 1$, n fiind dimensiunea matricii. Se încarcă în SI adresa efectivă a primului element al matricii și în CX dimensiunea matricii. Se salvează în stivă conținutul CX și se încarcă în CH indicele corespunzător coloanei. Se construiește în acumulator cu ajutorul lui CL și al lui CH adresa relativă necesară adresării elementului ce trebuie adunat și se transferă în acumulator acest element.

Procedura SIMM: apelează de două ori ELEM pentru a aduce din memorie elementele (i,j) și (j,i) , le compară, și în caz că nu sunt egale, întrerupe căutările resetând CF și revenind din procedură.

Programul principal pregătește contextul necesar apelului procedurii SIMM și transcrie în final rezultatul în funcție de starea fanionului CF în momentul revenirii din apelul lui SIMM.

Textul programului:

```
org 100h
jmp start
matr db 11h,12h,13h,14h
dim equ $-matr
db 12h,22h,23h,24h
db 13h,23h,55h,1
db 14h,24h,1,44h
simetrica db ? ;-1/0 pentru matrice
;simetrica/nesimetrica
start: mov ax,cs
mov ds,ax
mov si,offset matr
mov dl,dim
call SIMM
mov simetrica,-1
```

```

        jc      S1
        inc     simetrica
S1:     int     20h
ELEM:   push    dx
        mov     al,cl           ;i
        dec     al             ;i-1
        mul     dl             ;(i-1)*n
        dec     ax             ;(i-1)*n-1
        add     al,ch          ;(i-1)*n+j-1
        adc     ah,0
        mov     bx,ax
        mov     al,byte ptr [si][bx]
        pop     dx
        ret
SIMM:   mov     cl,0
SM1:    inc     cl             ;CL <- i
        cmp     cl,dl          ;bucla i de la 1 la n-1
        jae     SM3
        mov     ch,cl
SM2:    inc     ch             ;ch <- j
        cmp     ch,dl          ;bucla j de la i+1 la n
        ja     SM1
        call    ELEM
        push    ax             ;ax <- a[i,j]
        xchg   cl,ch
        call    ELEM           ;ax <- a[j,i]

```

```

        pop     bx             ;bx <- a[i,j]
        xchg   cl,ch
        cmp    al,bl
        jne    SM4           ;salt daca nu e simetrica
        jmp    SM2           ;reia bucla
SM3:    stc                 ;matrice simetrica
SM4:    clc                 ;matrice nesimetrica
        ret
        end

```

Programul 5.

Să se scrie un program care să copieze un șir de caractere ASCII, terminat cu 0h.

Se va folosi o procedură, STRCPY, care

primește: în SI - adresa efectivă a primului element al șirului;

în DI - adresa efectivă de început a șirului destinație.

Pentru accesarea elementelor șirului dat, de vor folosi instrucțiuni de lucru cu șiruri.

Soluție:

Programul va parcurge elementele șirului dat și la va copia începând de la adresa șirului destinație, sir2. Oprirea parcurgerii se va face în momentul întâlnirii octetului 0h.

Textul programului:

```

        org    100h
        jmp    start
sir1    db     'Exemplu de sir ASCII',0h
sir2    db     25     dup(?)
start:  mov    ax,cs
        mov    ds,ax

```

