

Seminar IV ASC

Instrucțiuni pentru șiruri. Probleme „complexe” pe șiruri

4. Conținut

4.1	Instrucțiuni pentru transferul datelor.....	1
4.2	Instrucțiuni pentru compararea datelor	2
4.3	Exemple.....	2

Instrucțiunile pentru șiruri au operanzi implicați și sunt folosite în felul următor: fac ceva cu elementul curent al șirului (șirurilor) și se mută la următorul element din șir (șiruri). Pentru a folosi aceste instrucțiuni, inițial trebuie:

- copiat offset-ul șirului sursă în registrul ESI (șirul sursă nu va fi modificat)
- copiat offset-ul șirului destinație în registrul EDI (șirul destinație este cel modificat)
- configurată direcția de parcurgere a șirului: dacă Direction Flag (DF) are valoarea 0 șirurile sunt parcurse de la stânga la dreapta, dacă DF = 1 șirurile sunt parcurse de la dreapta la stânga

Un set de instrucțiuni pentru șiruri folosește doar șirul sursă, alt set folosește doar șirul destinație iar alt set folosește atât șirul sursă cât și șirul destinație.

4.1 Instrucțiuni pentru transferul datelor

(Load String of Bytes)

1. LODSB

$AL \leftarrow \langle DS:ESI \rangle$
dacă DF=0 inc(ESI) altfel dec(ESI)

(Load String of Words)

2. LODSW

$AX \leftarrow \langle DS:ESI \rangle$
dacă DF=0 $ESI \leftarrow ESI+2$ altfel $ESI \leftarrow ESI-2$

(Store String of Bytes)

3. STOSB

$\langle ES:EDI \rangle \leftarrow AL$
dacă DF=0 inc(EDI) altfel dec(EDI)

(Store String of Words)

4. STOSW

$\langle ES:EDI \rangle \leftarrow AX$
dacă DF=0 $EDI \leftarrow EDI+2$ altfel $EDI \leftarrow EDI-2$

(Move String of Bytes)

5. MOVSB

$\langle ES:EDI \rangle \leftarrow \langle DS:ESI \rangle$
dacă DF=0 {inc(ESI); inc(EDI)} altfel {dec(ESI); dec(EDI)}

(Move String of Words)

6. MOVSW

$\langle ES:EDI \rangle \leftarrow \langle DS:ESI \rangle$
dacă DF=0 {ESI \leftarrow ESI+2; EDI \leftarrow EDI+2}
altfel {ESI \leftarrow ESI-2; EDI \leftarrow EDI-2}

4.2 Instrucțiuni pentru compararea datelor

(Scan String of Bytes)

7. SCASB

CMP AL, <ES:EDI>
dacă DF=0 inc(EDI) altfel dec(EDI)

(Scan String of Words)

8. SCASW

CMP AX, <ES:EDI>
dacă DF=0 EDI←EDI+2 altfel EDI←EDI-2

(Compare String of Bytes)

9. CMPSB

CMP <DS:ESI>, <ES:EDI>
dacă DF=0 {inc(ESI); inc(EDI)} altfel {dec(ESI); dec(EDI)}

(Compare String of Words)

10. CMPSW

CMP <DS:ESI>, <ES:EDI>
dacă DF=0 {ESI←ESI+2; EDI←EDI+2}
altfel {ESI←ESI-2; EDI←EDI-2}

Există și instrucțiunile LODSD, STOSD, MOVSD, SCASD, CMPSD care consideră elementele șirului ca fiind dublucuvinte, folosesc registrul EAX și incrementează/decrementează ESI și EDI cu patru octeți.

4.3 Exemple

Se rezolvă problema din seminarul 3 folosind instrucțiuni care manipulează șiruri.

Ex. 1. Se dă un șir de octeți care conține litere mici, să se construiască un șir nou de octeți care să conțină literele din șirul inițial transformate în majuscule.

```
bits 32
global start
extern exit
import exit msvcrt.dll

segment data use32 class=data
    s1 db 'abcdef'
    lenS1 equ $-s1      ; lungimea in octeti a sirului s1, 6 octeti
    s2 times lenS1 db 0 ; se rezerva lenS1 octeti pentru sirul s2

segment code use32 class=code
start:
    mov esi, s1      ; se copiaza offset-ul sirului s1 in ESI
    mov edi, s2      ; se copiaza offset-ul sirului ss in EDI
    mov ecx, lenS1   ; se repeta urmatoarele instructiuni de lenS1 ori
    cld

    repeat:
        lodsb                ; mov al, [esi]      +   inc esi
        sub al, 'a' - 'A'
        stosb                ; mov [edi], al      +   inc edi

        loop repeat
    ; instructiunea loop este echivalenta cu urmatoarele 3 instructiuni:
    ; dec ecx
    ; cmp ecx, 0
```

```

; ja repeat

push dword 0
call [exit]

```

Ex. 2. Se dă un șir de octeți. Sa se obțină șirul oglindit.

Exemplu: Fie următorul șir de octeți
s db 17, 20, 42h, 1, 10, 2ah
șirul oglindit este
t db 2ah, 10, 1, 42h, 20, 17.

Pentru a rezolva problema se parcurge șirul inițial „s” într-o buclă și se copiază fiecare octet în șirul „t”. Șirul „s” este parcurs de la stânga la dreapta (DF = 0) iar șirul „t” de la dreapta la stânga (DF = 1). Astfel primul octet din șirul „s” se va copia în ultimul octet din șirul „t”, etc.

```

bits 32
global start
extern exit
import exit msvcrt.dll

segment data use32 class=data
s db 17, 20, 42h, 1, 10, 2ah
len_s equ $-s
t times len_s db 0

segment code use32 class=code
start:
mov esi, s ; se copiaza offset-ul s in ESI
; ESI contine offset-ul primului octet din sirul s
mov edi, t ; se copiaza offsetul sirului t in EDI
; deoarece sirul t trebuie parcurs de la dreapta la stanga
; offset-ul din EDI trebuie sa fie cel al ultimului octet
; din sirul t (EDI = t + len_s - 1)
add edi, len_s-1
mov ecx, len_s
jecz theend ; daca ECX==0 salt la "theend"
repeat:
cld ; DF=0 (se parcurge sirul de la stanga la dreapta)
lodsb ; mov al, [esi] + inc esi

std ; DF=0 (se parcurge sirul de la dreapta la stanga)
stosb ; mov [edi], al + dec edi

loop repeat ;

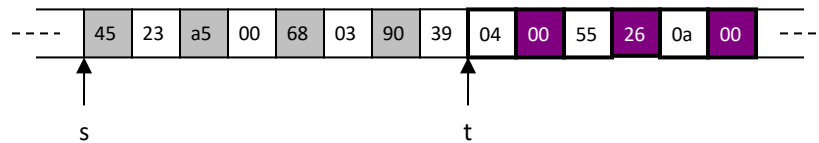
theend:
push dword 0
call [exit]

```

Ex.3. Se dau două șiruri de cuvinte, să se concateneze șirul octeților inferiori cuvintelor primului șir cu șirul octeților superiori cuvintelor din cel de-al doilea șir. Șirul rezultat trebuie sortat crescător în interpretare cu semn.

Exemplu: fie cele două șiruri de cuvinte:
 s dw 2345h, 0a5h, 368h, 3990h
 t dw 4h, 2655h, 10

aceste șiruri vor fi reprezentate în memorie în format little-endian în felul următor (octeții colorați sunt cei din cerința problemei)



Șirul rezultat este:

u: 90h, a5h, 0h, 0h, 26h, 45h, 68h

```
bits 32
global start
extern exit
import exit msvcrt.dll

segment data use32 class=data
s   dw  2345h, 0a5h, 368h, 3990h
len_s equ ($-s)/2 ; lungimea (în cuvinte) a șirului "s"
t   dw  4h, 2655h, 10
len_t equ ($-t)/2 ; lungimea (în cuvinte) a șirului "t"
len   equ len_s+len_t; lungimea șirului rezultat
u   times len db 0 ; șirul rezultat

segment code use32 class=code
start:
    ; se copiază octeții inferiori cuvintelor din s în șirul u
    mov esi, s ; offset pentru șirul sursă (offset-ul primului
                ; octet din șirul s)
    mov edi, u ; offset pentru șirul destinație (offset-ul primului
                ; octet din șirul u)
    cld      ; DF=0

    mov ecx, len_s ; o buclă cu len_s iterații
    jecxz theend

repeat:
    lodsw ; mov ax, [esi] + esi:=esi+2
    ; AL va stoca octetul inferior al cuvântului curent din s
    ; AH va stoca octetul superior al cuvântului curent din s
```

```

    stosb    ; mov [edi], al + edi:=edi+1
    ; trebuie copiat doar octetul inferior (AL) în şirul destinaţie
    loop repeat

    ; se copiază octeţii superiori cuvintelor din t în şirul u
    mov esi, t    ; offset-ul şirului sursă t
    mov ecx, len_t ; o buclă cu len_t iteraţii
    jecxz theend

repetal:
    lodsw    ; mov ax, [esi] + esi:=esi+2
    ; AL va stoca octetul inferior al cuvântului curent din t
    ; AH va stoca octetul superior al cuvântului curent din t

    xchg al, ah ; se interschimbă valorile din AL şi AH
    ; interesează octetul superior cuvintelor din t
    stosb    ; mov [edi], al + edi:=edi+1
    loop repetal; această buclă se mai poate scrie astfel:
        ; repetal:
        ;     lodsb
        ;     lodsb
        ;     stosb
        ;     loop repetal

    ; se sortează şirul u crescător în interpretare cu semn, se va
    ; implementa o variantă a bubble sort descrisă mai jos
; // u este un vector de lungime „len”
; changed = 1;
; while (changed = =1) {
;     changed = 0;
;     for (i=1; i<=len-1; i++) {
;         if (u[i+1]<u[i]) {
;             aux = u[i];
;             u[i] = u[i+1];
;             u[i+1] = aux;
;             changed = 1;
;         }
;     }
; }
; }

    mov dx, 1    ; changed=1
repeat2:
    cmp dx, 0
    je theend
    mov esi, u    ; offset u in ESI
    mov dx, 0    ; inițializare DX
    mov ecx, len-1 ; şirul u este prelucrat într-o bucla în
                  ; len-1 iteraţii

repeat3:
    mov al, byte [esi]    ; al = u[i].
    cmp al, byte [esi+1]  ; comparare al=u[i] cu u[i+1]
    jle next              ; dacă u[i]<=u[i+1] -> i++, altfel

```

```
                ; interschimba u[i] (byte [esi]) cu u[i+1]
                ; (byte [esi+1])
    mov ah, byte [esi+1]
    mov byte [esi], ah
    mov byte [esi+1], al
    mov dx, 1      ; changed = 1

next:
    inc esi        ; i++
    loop repeat3
    jmp repeat2

theend:
    push dword 0
    call [exit]
```