Laborator 7 - Suport teoretic

Operatii pe siruri de bytes/words/doublewords/quadwords

Operatii pe siruri

- Instructiunile pe siruri au operanzi impliciti.

Tipuri de instructiuni pe siruri:

- care folosesc un sir sursa si un sir destinatie (MOVSB, MOVSW, MOVSD, CMPSB, CMPSW, CMPSD)
- care folosesc numai un sir sursa (LODSB, LODSW, LODSD)
- care folosesc numai un sir destinatie (STOSB, STOSW, STOSD, SCASB, SCASW, SCASD)

Pentru a fi retinute mai usor

- Move String = MOVS
- Compare String = CMPS
- Load String = LODS
- Store String = STOS
- Scan String = SCAS

Un sir este caracterizat de:

- *tipul elementelor (octeti, cuvinte sau dublucuvinte)* => este indicat de ultima litera a instructiunii folosite (B=byte, W=word, D=doubleword), ambele siruri avand acelasi tip
- *adresa primului element* => este o adresa FAR memorata astfel:
    - in DS:ESI - pentru sirul sursa
    - in ES:EDI - pentru sirul destinatie
- *directia de parcurgere* => este indicata de valoarea din flagul DF (0 - de la adrese mai mici la adrese mai mari, 1 - de la adrese mai mari la adrese mai mici.)
- *numărul de elemente* => cand este nevoie de el, se pune în CX sau ECX

Instructiuni pentru transfer de date

LODSB	In AL se incarcă octetul de la adresa <DS:(E)SI> Daca DF=0 atunci inc((E)SI), altfel dec((E)SI)

LODSW	In AX se incarcă cuvantul de la adresa <DS:(E)SI> Daca DF=0 atunci (E)SI:=(E)SI+2, altfel (E)SI:=(E)SI-2
LODSD	In EAX se incarcă cuvantul de la adresa <DS:(E)SI> Daca DF=0 atunci (E)SI:=(E)SI+4, altfel (E)SI:=(E)SI-4
STOSB	La adresa <ES:(E)DI> se incarcă octetul din AL Daca DF=0 atunci inc((E)DI), altfel dec((E)DI)
STOSW	La adresa <ES:(E)DI> se incarcă cuvantul din AX Daca DF=0 atunci (E)DI:= (E)DI+2, altfel (E)DI:= (E)DI-2
STOSD	La adresa <ES:DI> se incarcă cuvantul din EAX Daca DF=0 atunci (E)DI:= (E)DI+4, altfel (E)DI:= (E)DI-4
MOVSB	La adresa <ES:(E)DI> se incarca octetul de la adresa <DS:(E)SI> Daca DF=0 atunci inc(SI), inc(DI), altfel dec(SI), dec(DI)
MOVSW	La adresa <ES:(E)DI> se incarca cuvantul de la adresa <DS:(E)SI> Daca DF=0 atunci (E)SI:= (E)SI+2, (E)DI:= (E)DI+2, altfel (E)SI:= (E)SI-2, (E)DI:= (E)DI-2
MOVSD	La adresa <ES:(E)DI> se incarca cuvantul de la adresa <DS:(E)SI> Daca DF=0 atunci (E)SI:= (E)SI+4, (E)DI:= (E)DI+4, altfel (E)SI:= (E)SI-4, (E)DI:= (E)DI-4

- **Obs. Avand in vedere utilizarea modelului de memorie flat, la orice inceput de executie a programului, SO va initializa cu aceeasi valoare registrii segment DS = ES, programatorul neavând nici o responsabilitate de incarcare/actualizare/modificare a acestor valori. In cadrul codului sursa ce utilizeaza instructiuni pe siruri programatorul va trebui sa gestioneze doar offset-urile acestor siruri.**

Exemplu:

```
;Avem un sir sursa (cuvinte). Sa se copieze intr-un sir destinatie. Stim cate
elemente avem.
mov ECX, dim_sir ; nr de elemente din sir
mov ESI, sir_sursa ; incarcare offset sir_sursa in ESI
mov EDI, sir_dest ; incarcare offset sir_dest in EDI
CLD
Again:
        LODSW
        STOSW
LOOP Again
```

- Avand in vedere ca LODS + STOS = MOVS bucla de mai sus se mai poate scrie:

```
Again:
MOVSW
LOOP Again
```

- sau (a se vedea sectiunea prefixelor de instructiune de mai jos)

```
rep MOVSW
```

Instructiuni pentru consultarea si compararea datelor

SCASB	CMP AL, <ES:(E)DI> Daca DF=0 atunci inc((E)DI), altfel dec((E)DI)
SCASW	CMP AX, <ES:(E)DI> Daca DF=0 atunci (E)DI:= (E)DI+2, altfel (E)DI:= (E)DI-2
SCASD	CMP EAX, <ES:(E)DI> Daca DF=0 atunci (E)DI:= (E)DI+4, altfel (E)DI:= (E)DI-4
CMPSB	CMP <DS:(E)SI>, <ES:(E)DI> Daca DF=0 atunci inc(SI), inc(DI), altfel dec(SI), dec(DI)
CMPSW	CMP <DS:(E)SI>, <ES:(E)DI> Daca DF=0 atunci (E)SI:= (E)SI+2, (E)DI:= (E)DI+2, altfel (E)SI:= (E)SI-2, (E)DI:= (E)DI-2
CMPSD	CMP <DS:(E)SI>, <ES:(E)DI> Daca DF=0 atunci (E)SI:= (E)SI+4, (E)DI:= (E)DI+4, altfel (E)SI:= (E)SI-4, (E)DI:= (E)DI-4

Exemplu:

```
;Se da un sir de octeti. Sa se găsească ultimul caracter "0".
;... se incarca toate datele despre sirul "destinatie"
MOV AL, '0'
MOV ECX, lung_sir
STD
Cont_caut: ;continui cautarea...
        SCASB
        JE Gasit
LOOP Cont_caut
;...
Gasit:
        INC EDI;ma intorc la caracterul gasit inainte sa se fi facut
decrementarea lui EDI
```

Prefixe de instructiune pentru executia repetata a unei instructiuni

```
prefix_de_instructiune instructiune_pe_sir
```

- echivalenta cu

```
Again:
       instructiune_pe_sir
LOOP Again
```

- unde *prefix_de_instructiune* poate fi REP, echivalent cu REPE (*Repeat While Equal*), REPZ (*Repeat While Zero*) - care provoaca executia repetata a instructiunilor SCAS sau CMPS pana cand ECX devine 0 sau pana cand apare o nepotrivire (=> ZF=0)
- sau poate fi REPNE (*Repeat While Not Equal*) sau REPNZ (*Repeat While Not Zero*) - care provoaca executia repetata a instructiunii SCAS sau CMPS pana cand ECX devine 0 sau pana cand apare o potrivire (=> ZF=1)

Observatii:

- instructiunile pe siruri nu afecteaza flagurile in urma actiunii asupra registrilor ESI, EDI sau ECX
- LODS, STOS, MOVS - nu afecteaza nici un flag, in timp ce SCAS si CMPS modifica flagurile doar ca rezultat al comparatiilor efectuate

---

Laborator 7 - Exemple

Operatii pe siruri de bytes/words/doublewords/quadwords

Exemplu

```
;Problema. Se da un sir de valori numerice intregi reprezentate pe
quadworduri.
;Sa se determine suma cifrelor numarului multiplilor de 8 din sirul octetilor
;inferiori ai cuvintelor superioare ai dublucuvintelor superioare din
elementele sirului de quadworduri.
bits 32
global start
extern exit,printf; tell nasm that exit exists even if we won't be defining it
import exit msvcrt.dll; exit is a function that ends the calling process. It
is defined in msvcrt.dll
import printf msvcrt.dll
; our data is declared here (the variables needed by our program)
segment data use32 class=data
        sir  dq  123110110abcb0h,1116adcb5a051ad2h,4120ca11d730cbb0h
        len equ ($-sir)/8;lungimea sirului (in dublucuvinte)
        opt db 8;variabila folosita pentru testarea divizibilitatii cu 8
        zece dw 10;variabila folosita pentru determinarea cifrelor unui numar
prin impartiri succesive la 10
        suma dd  0;variabila in care retinem suma cifrelor
        format  db "%x", 0
; our code starts here
segment code use32 class=code
    start:
        mov esi, sir
        cld;parcurgem sirul de la stanga la dreapta (DF=0).
        mov ecx, len;vom parcurge elementele sirului intr-o bucla loop cu len
iteratii.
        mov ebx, 0;in registrul ebx vom retine numarul multiplilor lui 8.
        repeta:
                lodsd;in eax vom avea dublucuvantul mai putin semnificativ al
quadword-ului curent din sir
                lodsd;in eax vom avea dublucuvantul cel mai semnificativ al
quadword-ului curent din sir
                shr eax, 16
                mov ah, 0;ne intereseaza doar octetul mai putin semnificativ
din acest cuvant (AL)

                div byte[opt];vedem daca al este divizibil cu 8
                cmp ah, 0;daca restul nu este 0, reluam ciclul repeta.
                                ;Altfel incrementam numarul multiplilor de 8
din registrul bx.
                jnz nonmultiplu
                inc ebx

                nonmultiplu:
```

```
        loop repeta;daca mai sunt elemente de parcurs(cx>0) reia ciclul.

        ;mai departe, obtinem cifrele numarului bx in baza 10 prin impartiri
succesive la 10 si calculam suma acestor cifre.

        mov eax, ebx
        mov edx, 0

        transf:
                div dword[zece];impartim la 10 numarul din registrul ca sa
aflam ultima cifra
                add dword[suma], edx;adunam cifra la suma.
                cmp eax, 0
        jz sfarsit;daca catul este 0 inseamna ca am obtinut toate cifrele si
putem parasi bucla transf
                                ;Altfel, il pregatim pentru o noua iteratie
        mov edx, 0
        jmp transf;reluam bucla pentru obtinerea unei noi cifre.

sfarsit:;incheiem programul.

        push dword suma; punem parametrii pe stiva de la dreapta la stanga
        push dword format
        call [printf];apelam functia printf
        add esp, 4 * 2; eliberam parametrii de pe stiva

        push dword 0; push the parameter for exit onto the stack
        call [exit]; call exit to terminate the program
```

Laborator 7 - Probleme propuse

Operatii pe siruri de bytes/words/doublewords/quadwords

Exerciții

Problemele din acest laborator trebuie rezolvate folosind instructiuni specifice lucrului cu siruri: LODSB, STOSB, MOVSB, SCASB, CMPSB, LODSW, STOSW, MOVSW, SCASW, CMPSW.

1. Se dau doua siruri de caractere ordonate alfabetic s1 si s2. Sa se construiasca prin interclasare sirul ordonat s3 care sa contina toate elementele din s1 si s2.
2. Se da un sir de dublucuvinte. Sa se ordoneze descrescator sirul cuvintelor inferioare ale acestor dublucuvinte. Cuvintele superioare raman neschimbate.

Exemplu:

dandu-se:

```
sir DD 12345678h 1256ABCDh, 12AB4344h
```

rezultatul va fi

```
1234ABCDh, 12565678h, 12AB4344h.
```

3. Se da un sir de dublucuvinte. Sa se ordoneze crescator sirul cuvintelor superioare ale acestor dublucuvinte. Cuvintele inferioare raman neschimbate.

Exemplu:

dandu-se:

```
sir DD 12AB5678h, 1256ABCDh, 12344344h
```

rezultatul va fi

```
12345678h, 1256ABCDh, 12AB4344h.
```

4. Dandu-se doua siruri de octeti sa se calculeze toate pozitiile unde al doilea sir apare ca subsir in primul sir.
5. Se da un sir de octeti reprezentand un text (succesiune de cuvinte separate de spatii). Sa se identifice cuvintele de tip palindrom (ale caror oglindiri sunt similare cu cele de plecare): "cojoc", "capac" etc.
6. Dandu-se un sir de cuvinte sa se obtina sirul (de octeti) cifrelor in baza zece ale fiecarui cuvant din acest sir.

Exemplu:

daca avem sirul:

```
sir DW 12345, 20778, 4596
```

obtinem rezultatul

```
1, 2, 3, 4, 5, 2, 0, 7, 7, 8, 4, 5, 9, 6.
```

7. Se da un sir de octeti 'input' si inca doua siruri de dimensiune N fiecare, 'src' si 'dst'. Sa se obtina un nou sir 'output' din sirul 'input' in care se vor inlocui toti octetii cu valoarea src[i] cu dst[i], unde i=1..N.
8. Dandu-se un sir de octeti sa se obtina un sir de cuvinte care sa contina in octetii inferiori multimea caracterelor din sirul de octeti, iar octetul superior al unui cuvant sa contina numarul de aparitii al octetului inferior din acel cuvant in sirul de octeti dat.

Exemplu:

se da sirul

```
sir DB 2, 4, 2, 5, 2, 2, 4, 4
```

se va obtine sirul

```
rez DW 0402h, 0304h, 0105h.
```

9. Dandu-se un sir de dublucuvinte, sa se obtina un alt sir de dublucuvinte in care se vor pastra doar dublucuvintele din primul sir care au un numar par de biti cu valoare 1.
10. Se da un sir de octeti. Sa se obtina sirul oglindit al reprezentarii binare a acestui sir de octesi.

Exemplu:

Se da sirul de octeti:

```
s DB 01011100b, 10001001b, 11100101b
```

Sa se obtina sirul

```
d DB 10100111b, 10010001b, 00111010b.
```

11. Se da un sir de dublucuvinte. Sa se obtina sirul format din octetii superiori ai cuvintelor inferioare din elementele sirului de dublucuvinte, care sunt multiplii de 10.

Exemplu:

Se da sirul de dublucuvinte:

```
s DD 12345678h, 1A2B3C4Dh, FE98DC76h
```

Sa se obtina sirul

```
d DB 3Ch, DCh.
```

12. Dandu-se un sir de cuvinte, sa se calculeze cel mai lung subsir de cuvinte ordonate crescator din acest sir.
13. Dandu-se un sir de octeti si un subsir al sau, sa se elimine din primul sir toate aparitiile subsirului.
14. Se dau doua siruri de octeti. Sa se parcurga cel mai scurt sir dintre cele doua siruri si sa se construiasca un al treilea sir care va contine cel mai mare element de acelasi rang din cele doua siruri, iar pana la lungimea celui mai lung sir, sirul al treilea se va completa alternativ cu valoarea 1 respectiv 0.
15. Se da un sir de cuvinte. Sa se construiasca doua siruri de octeti, s1 si s2, astfel: pentru fiecare cuvant,
    - daca numarul de biti 1 din octetul high al cuvantului este mai mare decat numarul de biti 1 din octetul low, atunci s1 va contine octetul high, iar s2 octetul low al cuvantului
    - daca numarul de biti 1 din cei doi octeti ai cuvantului sunt egali, atunci s1 va contine numarul de biti 1 din octet, iar s2 valoarea 0
    - altfel, s1 va contine octetul low, iar s2 octetul high al cuvantului.