Programarea microprocesorului Intel 8086.
Instructiuni pentru lucrul cu siruri

Scopul lucrării

a) Instructiuni de transfer pentru lucrul cu siruri.
b) Instructiuni de comparatie pentru lucrul cu siruri.

1. Instructiuni pentru lucrul cu siruri

Sirurile de date sunt declarate in limbajul de asamblare al microprocesorului Intel 8086 cu ajutorul directivelor DB, DW si DD (in functie de tipul datelor).

Exemplul 1:

```
cifra DB 0, 1, 2, 3, 4, 5, 6, 7, 8, 9, 0AH, 0BH, 0CH, 0DH, 0EH, 0FH
```

Exemplul 2:

```
mesaj  DB  'Introduceti datele $'
```

In primul caz este vorba de 16 octeti reprezentand cifrele in baza 16 iar in al doilea caz este vorba de 19 octeti reprezentand codurile ASCII ale literelor unui text.

Pentru realizarea unei operatii in mod repetat asupra elementelor consecutive ale unui sir sau a doua siruri se utilizeaza instructiuni specializate numite *instructiuni pentru lucrul cu siruri.*

1.1. Instructiuni pentru lucrul cu siruri cu operanzi expliciti

Instructiunile pentru lucrul cu siruri utilizeaza intotdeauna cate 2 operanzi. Daca acestia sunt specificati explicit in codul instructiunii dupa mnemonica atunci instructiunea se numeste *cu operanzi expliciti.*

1.2. Instructiuni pentru lucrul cu siruri cu operanzi impliciti

Instructiunile pentru lucrul cu siruri care nu are cei doi operanzi specificati explicit in codul instructiunii se numesc *cu operanzi impliciti.*

Deoarece sirurile de date se afla in memorie s-au stabilit *perechi de registre utilizate implicit pentru adresarea datelor curente ale sirurilor*. Daca sirul este de tip *destinatie*, atunci perechea de registre de adresare este ES : DI. Daca sirul este de tip *sursa*, atunci perechea de registre de adresare este DS : SI. De aceea registrul DI poarta numele *Destination Index* iar registrul SI numele *Source Index*.

Daca datele sunt octeti (*Bytes*), instructiunea utilizata va avea terminatia **B** iar daca datele sunt cuvinte (*Words*), instructiunea utilizata va avea terminatia **W**.

Actualizarea registrelor de adresare difera in functie de doi parametrii:

- tipul datelor sirului (exprimat in numar de octeti: **1** pentru *Byte* si **2** pentru *Word*), care determina valoarea absoluta a actualizarii (modulul actualizarii),
- valoarea flag-ului Direction (DF, **1** pentru sens *direct* si **0** pentru sens *invers*), care precizeaza sensul in care se face actualizarea.

Prin *conventia Intel* s-a stabilit sensul direct sensul crescator al adreselor iar sensul invers sensul descrescator al adreselor. De aceea putem spune ca valoarea flag-ului DF reprezinta semnul actualizarii registrelor SI si/sau DI.

Daca notam cu *a* actualizarea:

- pentru o instructiune care lucreaza cu octeti (terminata cu **B**),
  - daca DF = 0: $a = +1$
  - daca DF = 1: $a = -1$
- pentru o instructiune care lucreaza cu cuvinte (terminata cu **W**),
  - daca DF = 0: $a = +2$
  - daca DF = 1: $a = -2$

Instructiunile pentru lucrul cu siruri se pot imparti in instructiuni *de transfer* si instructiuni *de comparatie*. O instructiune de lucru cu siruri este *doua sarcini de realizat*:

- operatia propriu-zisa (transfer sau comparatie) asupra datelor curente,
- actualizarea registrelor de adresare corespunzatoare (SI si/sau DI) pentru urmatoarea executie a instructiunii (pregatirea urmatoarelor date).

2. Instructiuni de transfer pentru lucrul cu siruri

2.1. Instructiunile MOVS, LODS, STOS

Instructiunile de transfer pentru lucrul cu siruri cu operanzi expliciti sunt:

- transfer *intre doua siruri*:

```
MOVS SIRDEST, SIRSURSA
```

- transfer din sir in registru (*incarcare registru*):

```
LODS  AL, SIRSURSA        sau
LODS  AX, SIRSURSA
```

- transfer din registru in sir (*stocare in memorie*):

STOS SIRDEST, AL sau
STOS SIRDEST, AX

Instructiunile cu operanzi impliciti pentru siruri de octeti au formele:
- transfer *intre doua siruri*:

MOVSB

- transfer din sir in registru (*incarcare registru*):

LODSB

- transfer din registru in sir (*stocare in memorie*):

STOSB

iar *instructiunile cu operanzi impliciti pentru siruri de octeti* au formele:
- transfer *intre doua siruri*:

MOVSW

- transfer din sir in registru (*incarcare registru*):

LODSW

- transfer din registru in sir (*stocare in memorie*):

STOSW

1. Efectul instructiunii MOVSB este:

a) Daca DF = 0:

ES : DI <-- DS : SI	- transfer curent sir - sir
SI <-- SI + 1	- actualizare index sir sursa
DI <-- DI + 1	- actualizare index sir dest.

b) Daca DF = 1:

ES : DI <-- DS : SI	- transfer curent sir - sir
SI <-- SI - 1	- actualizare index sir sursa
DI <-- DI - 1	- actualizare index sir dest.

2. Efectul instructiunii MOVSW este:

a) Daca DF = 0:

ES:DI,DI+1 <-- DS:SI,SI+1 - transfer curent sir-sir
SI <-- SI + 2 - actualizare index sir sursa
DI <-- DI + 2 - actualizare index sir dest.

b) Daca DF = 1:

ES:DI,DI+1 <-- DS:SI,SI+1 - transfer curent sir-sir
SI <-- SI - 2 - actualizare index sir sursa
DI <-- DI - 2 - actualizare index sir dest.

3. Efectul instructiunii LODSB este:

a) Daca DF = 0:

AL <-- DS : SI - transfer curent sir-acumulator
SI <-- SI + 1 - actualizare index sir sursa
DI <-- DI + 1 - actualizare index sir dest.

b) Daca DF = 1:

AL <-- DS : SI - transfer curent sir-acumulator
SI <-- SI - 1 - actualizare index sir sursa
DI <-- DI - 1 - actualizare index sir dest.

4. Efectul instructiunii LODSW este:

a) Daca DF = 0:

AX <-- DS : SI,SI+1 - transfer curent sir-acumulator
SI <-- SI + 2 - actualizare index sir sursa
DI <-- DI + 2 - actualizare index sir dest.

b) Daca DF = 1:

AX <-- DS : SI,SI+1 - transfer curent sir-acumulator
SI <-- SI - 2 - actualizare index sir sursa
DI <-- DI - 2 - actualizare index sir dest.

5. Efectul instructiunii STOSB este:

a) Daca DF = 0:

ES : DI <-- **AL** - transfer curent acumulator-sir
SI <-- SI + 1 - actualizare index sir sursa
DI <-- DI + 1 - actualizare index sir dest.

b) Daca DF = 1:

ES : DI <-- **AL** - transfer curent acumulator-sir
SI <-- SI - 1 - actualizare index sir sursa
DI <-- DI - 1 - actualizare index sir dest.

6. Efectul instructiunii STOSW este:

a) Daca DF = 0:

ES : DI,DI+1 <-- **AX** - transfer curent acumulator-sir
SI <-- SI + 2 - actualizare index sir sursa
DI <-- DI + 2 - actualizare index sir dest.

b) Daca DF = 1:

ES : DI,DI+1 <-- **AX** - transfer curent acumulator-sir
SI <-- SI - 2 - actualizare index sir sursa
DI <-- DI - 2 - actualizare index sir dest.

2.2. Prefixul REP

Pentru a reduce timpul de executie in cazul repetarii de N>1 ori a unei instructiuni pentru lucrul cu siruri s-a introdus prefixul REP. Punand acest prefix in fata uneia dintre instructiunile pentru lucrul cu siruri, numita generic *instrsir*:

REP *instrsir*

efectul ei devine echivalent cu al secventei:

etich: *instrsir*
LOOP etich

(necesitand preincarcarea in registrul CX a numarului de repetari).

3. Instructiuni de comparatie pentru lucrul cu siruri

3.1. Instructiunile CMPS si SCAS

Instructiunile de comparatie pentru lucrul cu siruri cu operanzi expliciti sunt:
- comparatie intre doua siruri:

CMPS SIRDEST, SIRSURSA

- comparatie intre sir "destinatie" si registru:

SCAS SIRDEST, AL sau
SCAS SIRDEST, AX

Instructiunile cu operanzi impliciti pentru siruri de octeti au formele:
- comparatie *intre doua siruri*:

CMPSB

- comparatie *intre sir "destinatie" si registru*:

STOSB

iar *instructiunile cu operanzi impliciti pentru siruri de octeti* au formele:
- comparatie *intre doua siruri*:

CMPSW

- comparatie *intre sir "destinatie" si registru*:

STOSW

Aceste instructiuni au un *efect asemanator cu instructiunea de test CMP* (rezultatul nu este incarcat intr-un registru sau stocat in memorie). De aceea ele sunt *folosite in cazul in care se doreste testarea unor conditii.*

1. Efectul instructiunii CMPSB este:

a) Daca DF = 0:
CMP(ES : DI),(DS : SI)- comparatie valori curente sir-sir
SI <-- SI + 1 - actualizare index sir sursa
DI <-- DI + 1 - actualizare index sir destinatie

b) Daca DF = 1:
CMP(ES : DI),(DS : SI)- comparatie valori curente sir-sir
SI <-- SI - 1 - actualizare index sir sursa
DI <-- DI - 1 - actualizare index sir destinatie

2. Efectul instructiunii CMPSW este:

a) Daca DF = 0:
CMP(ES : DI,DI+1),(DS : SI,SI+1)
SI <-- SI + 2
DI <-- DI + 2
b) Daca DF = 1:
CMP(ES : DI,DI+1),(DS : SI,SI+1)

SI <-- SI - 2
DI <-- DI - 2

3. Efectul instructiunii SCASB este:

a) Daca DF = 0:

CMP(ES : DI),AL - comparatie valori curente sir-registru
SI <-- SI + 1 - actualizare index sir sursa
DI <-- DI + 1 - actualizare index sir destinatie

b) Daca DF = 1:

CMP(ES : DI), AL - comparatie valori curente sir-registru
SI <-- SI - 1 - actualizare index sir sursa
DI <-- DI - 1 - actualizare index sir destinatie

4. Efectul instructiunii SCASW este:

a) Daca DF = 0:

CMP(ES : DI,DI+1),AX
SI <-- SI + 2
DI <-- DI + 2

b) Daca DF = 1:

CMP(ES : DI,DI+1),AX
SI <-- SI - 2
DI <-- DI - 2

3.2. Prefixele REPZ, REPE, REPNZ si REPNE

Pentru a reduce timpul de executie in cazul unei repetari de N>1 ori a unei instructiuni de comparatie pentru lucrul cu siruri, se poate folosi unul dintre prefixele: REP**Z**, REP**E**, REP**NZ**, REP**NE**.

Punand unul dintre aceste prefixe, numit generic REP*conditie*, in fata uneia dintre instructiunile de comparatie pentru lucrul cu siruri, numita generic *instrsir*:

REP*conditie instrsir*

efectul ei devine echivalent cu al secventei:

etich: *instrsir*
LOOP*conditie* etich

4. Exemple de programe

4.1. Calcule in dubla precizie

1. Copierea unui sir de octeti *sirsursa* intr-un alt sir de octeti *sirdest*:

```
DATA SEGMENT
        sirsursa  DB  0, 10H, 20H, 30H, 40H, 50H, 60H, 70H
        sirdest   DB  8 DUP(?)
        N                 EQU 8
DATA ENDS
ASSUME CS: COD, DS: DATA
COD   SEGMENT
START:
        MOV  AX, DATA                           ; initializare DS
        MOV  DS, AX
        CLD                                     ; fortare DI = 0 (sens direct)
        MOV   SI, OFFSET sirsursa               ; pregatirea adresei primului
                                                ; octet din sirsursa
        MOV   DI, OFFSET sirdest                ; pregatirea adresei primului
                                                ; octet din sirdest
        MOV  CX, N                              ; numarul de elemente
                                                ; ale sirului sirsursa
et:     MOVSB                                   ; copiere octet curent
                                                ; si actualizare SI si DI
        LOOP  et                                ; repetare de N ori
        MOV  AH, 4CH                            ; exit
        INT  21H
COD   ENDS
        END  START
```

2. Copierea unui sir de cuvinte *sirsursa* intr-un alt sir de cuvinte *sirdest*:

```
DATA SEGMENT
        sirsursa  DW  0, 1000H, 2000H, 3000H, 4000H, 5000H
        sirdest   DW  6 DUP(?)
DATA ENDS
ASSUME CS: COD, DS: DATA
COD   SEGMENT
START:
        MOV  AX, DATA                           ; initializare DS
        MOV  DS, AX

        CLD                                     ; fortare DI = 1 (sens invers)
```

```
        MOV   SI, OFFSET sirsursa
        ADD   SI, M                     ; M = 2*(N-1) este deplasarea de
                                                ; la primul pana la ultimul
                                                ; cuvant din sirsursa
        MOV   DI, OFFSET sirdest        ; pregatirea adresei primului
                                                ; cuvant din sirdest
        ADD   DI, M                     ; M = 2*(N-1) este deplasarea de
                                                ; la primul pana la ultimul
                                                ; cuvant din sirdest
        MOV  CX, N                      ; numarul de elemente
                                                ; ale sirului sirsursa
et:     MOVSW                                   ; copiere cuvint curent
        LOOP  et                                ; repetare de N ori
        MOV  AH, 4CH                            ; exit
        INT  21H
COD     ENDS
        END  START
```

3. Incarcarea succesiva a octetilor unui sir *sirsursa* in registrul acumulator AL pentru negarea octetilor si stocarea lor in sirul *sirdest*:

```
DATA SEGMENT
        sirsursa  DB  0, 10H, 20H, 30H, 40H, 50H, 60H, 70H
        sirdest   DB  8 DUP(?)
        N                EQU 8
DATA ENDS
ASSUME CS: COD, DS: DATA
COD     SEGMENT
START:
        MOV  AX, DATA                           ; initializare DS
        MOV  DS, AX
        CLD                                     ; fortare DI = 0 (sens direct)
        MOV   SI,OFFSET sirsursa
        MOV   DI, OFFSET sirdest
        MOV  CX, N
et:     LODSB                                   ; incarcare octet curent in AL
        NEG  AL                                 ; negare continut AL
        STOSB                                   ; stocare continut AL in octet
                                                ; curent si actualizare DI
        LOOP  et                                ; repetare de N ori
        MOV  AH, 4CH                            ; exit
        INT  21H
COD     ENDS
        END  START
```

4. Copierea unui sir de cuvinte *sirsursa* intr-un alt sir de cuvinte *sirdest*:

```
DATA SEGMENT
        sirsursa  DW  0, 1000H, 2000H, 3000H, 4000H, 5000H
        sirdest   DW  6 DUP(?)
        N                 EQU 6
DATA ENDS
ASSUME CS: COD, DS: DATA
COD   SEGMENT
START:
        MOV  AX, DATA                           ; initializare DS
        MOV  DS, AX
        CLD                                     ; fortare DI = 0 (sens direct)
        MOV   SI, OFFSET sirsursa
        MOV   DI, OFFSET sirdest
        MOV  CX, N
        REP  MOVSW                              ; copiere cuvint curent
                                                ; repetata de N ori
        MOV  AH, 4CH                            ; exit
        INT  21H
COD   ENDS
        END  START
```

5. Compararea unui sir de octeti *sirsursa* cu un alt sir de octeti *sirdest* si semnalarea diferentei lor prin variabila *rez*:

Varianta 1.

```
DATA SEGMENT
        sirsursa  DB  0, 10H, 20H, 30H, 40H, 50H, 60H, 70H
        sirdest   DB  0, 10H, 20H, 30H, 40H, 50H, 50H, 70H
        N                 EQU 8
DATA ENDS
ASSUME CS: COD, DS: DATA
COD   SEGMENT
START:
        MOV  AX, DATA                           ; initializare DS
        MOV  DS, AX
        CLD                                     ; fortare DI = 0 (sens direct)
        MOV   SI, OFFSET sirsursa
        MOV   DI, OFFSET sirdest
        MOV  CX, N
        MOV  rez, 1                     ; initializare rez cu 1
                                                ; (siruri identice)
et1:    CMPSB                           ; comparare octet curent
        JE  et2                         ; continuare daca octetul curent
```

```
                                        ; este identic
        MOV  rez, 0                     ; modificare rez in 0
                                        ; (siruri diferite)
        JMP et3                         ; oprire daca octetul curent
                                        ; nu este identic
et2:    LOOP  et1                       ; repetare de N ori
et3:    MOV  AH, 4CH                    ; exit
        INT  21H
COD     ENDS
        END  START
```

Varianta 2.

```
DATA SEGMENT
        sirsursa  DB  0, 10H, 20H, 30H, 40H, 50H, 60H, 70H
        sirdest   DB  8 DUP(?)
        N                EQU 8
DATA ENDS
ASSUME CS: COD, DS: DATA
COD     SEGMENT
START:
        MOV  AX, DATA                   ; initializare DS
        MOV  DS, AX
        CLD                             ;  fortare DI = 0 (sens direct)
        MOV   SI, OFFSET sirsursa
        MOV   DI, OFFSET sirdest
        MOV  CX, N
        MOV  rez, 1                     ; initializare rez cu 1
                                        ; (siruri identice)
        REPZ  CMPSB                     ; repetare (de max. N ori) daca octetul
                                        ; curent al sirului sirsursa este identic
                ; cu octetul curent al sirului sirdest
        MOV  AX, ES:[DI]        ; copiere ultim octet comparat din sirdest
                                        ; in acumulator
        CMP  AX, [SI]                   ; comparatie cu oct. similar din sirsursa
        JE  et1                 ; continuare daca ultimii octeti comparati
                                        ; sunt identici
        MOV  rez, 0             ; rez = 0 (siruri diferite) daca ultimii
                                        ; octeti comparati sunt diferiti
et1:    MOV  AH, 4CH                    ; exit
        INT  21H
COD     ENDS
        END  START
```

6. Compararea octetilor unui sir de octeti *sirdest* cu un octet aflat in acumulator si semnalarea pozitiei primului octet care difera in *poz*:

```
DATA SEGMENT
        sirdest   DB  0, 10H, 20H, 30H, 40H, 50H, 60H, 70H
        N                 EQU 8
DATA ENDS
ASSUME CS: COD, DS: DATA
COD   SEGMENT
START:
        MOV  AX, DATA                           ; initializare DS
        MOV  DS, AX
        CLD                                     ; fortare DI = 0 (sens direct)
        MOV   DI,  OFFSET sirdest
        MOV  CX, N
        REPZ  SCASB                     ; comparare octet curent
        MOV  poz, AL                            ; stocare pozitie octet care
                                                ; difera
        MOV  AH, 4CH                            ; exit
        INT  21H
COD   ENDS
        END  START
```

7. Compararea cuvintelor unui sir de cuvinte *sirdest* cu un cuvant aflat in acumulator si semnalarea pozitiei primului cuvant care este identic in *poz*:

```
DATA SEGMENT
        sirdest   DW  0, 1000H, 2000H, 3000H, 4000H, 5000H, 6000H
        N                 EQU 7
DATA ENDS
ASSUME CS: COD, DS: DATA
COD   SEGMENT
START:
        MOV  AX, DATA                           ; initializare DS
        MOV  DS, AX
        CLD                                     ; fortare DI = 0 (sens direct)
        MOV   DI,  OFFSET sirdest
        MOV  CX, N
        REPNZ  SCASB                    ; comparare octet curent                MOV  poz, AL
                        ; stocare pozitie cuvant identic
                                                ; cu cel din acumulator
        MOV  AH, 4CH                            ; exit
        INT  21H
COD   ENDS
        END  START
```

5. Desfasurarea lucrarii

1. Se editeaza urmatorul *program de comparare a doua siruri de octeti sir1 si sir2, si de semnalare a diferentei lor prin variabila rez,* intr-un fisier cu numele *AP51.ASM*:

```
DATA SEGMENT
sir1    DB 'Triunghiul echilateral are toate laturile congruente.'
sir2    DB 'Triunghiul echilateral are toate laturile congruente.'
sir3    DB 'Triunghiul echilateral are toate unghiurile congruente.'
rez   DB ?
DATA ENDS
ASSUME CS: COD, DS: DATA
COD   SEGMENT
START:
        MOV  AX, DATA
        MOV  DS, AX
        MOV  CX, SIZE sir1
        MOV  SI, OFFSET sir1
        MOV  DI, OFFSET sir2
        MOV  rez, 'I'                ; siruri identice
        REPZ CMPSB
        JCXZ et1
        MOV  rez, 'D'                ; siruri diferite
et1:
        MOV  AH, 4CH
        INT  21H
COD   ENDS
        END  START
```

Se parcurg etapele 1.a) ... 1.f) de la lucrarea nr. 2 pentru acest program.

2. Se concepe si editeaza un *program care sa determine daca valoarea unui cuvant* cuv *se regaseste intr-un sir de cuvinte* sir *si sa semnaleze gasirea lui in variabila* rez, intr-un fisier cu numele *AP52.ASM.*

6. Teme si exercitii

1. Sa se scrie un *program care sa citeasca de la tastatura doua sir de caractere (incheiate de exemplu prin punct), sa determine daca sirurile sunt identice sau nu si sa afiseze un mesaj corespunzator pe ecran.*

2. Sa se scrie un *program care sa citeasca de la tastatura un sir de caractere (incheiat de exemplu prin punct) si inca un caracter pe care sa il compare cu sirul citit anterior, sa determine daca sirul contine caracterul si daca da pe ce pozitie, si sa afiseze mesaje rezultatele pe ecran*

7. Intrebari

1. Care sunt perechile de registre utilizate implicit la adresare de instructiunile pentru lucrul cu siruri ?

2. Care sunt registrele utilizate implicit de instructiunile pentru lucrul cu siruri LODS, STOS si SCAS ?

3. Care sunt asemanarile si care sunt deosebirile intre efectul instructiunii LOOP si efectul prefixului REP ?

4. Care sunt asemanarile si care sunt deosebirile intre efectul instructiunii LOOPE si efectul prefixului REPE ?