**Programarea microprocesorului Intel 8086.**

Instructiuni de transfer. Instructiuni aritmetice. Instructiuni logice, de deplasare si rotatie

Scopul lucrării

a) Elemente de limbaj de asamblare.

b) Instructiuni de transfer și aritmetice

c) Instructiuni logice, de deplasare si rotatie.

**1. Elemente de limbaj de asamblare pentru microprocesorul Intel 8086**

***1.1. Tipuri de date utilizate de microprocesorul Intel 8086***

Reprezentarea interna a informatiilor (instructiuni, date, adrese, comenzi, etc.) este realizata in binar. De aceea, tipurile de date elementare (grupurile de biti) utilizate de microprocesorul Intel 8086 au urmatoarele dimensiuni si denumiri:

- bitul (bit) = cifra binara,

- octetul (Byte) = grup de 8 biti,

- cuvantul (Word) = grup de 16 biti = 2 octeti (MSB = octetul superior, LSB = octetul inferior),

- dublul-cuvant (Double-word) = grup de 32 biti = 4 octeti = 2 cuvinte (MSW = cuvantul superior, LSW = cuvantul inferior).

Dublul-cuvint este tipul de data necesar pentru memorarea unei adrese logice, fapt pentru care se numeste si "pointer".

Un tip de date derivat este inregistrarea de biti, formata din 8 sau 16 biti (octet sau cuvant), compusa din campuri de biti de lungimi variabile, cu semnificatii diferite. Dintre tipurile de date compuse, necesare pentru structuri de date complexe, cel mai utilizat este sirul de date, care permite memorarea tablourilor (vectorilor, matricilor, etc.). Un alt tip compus este stuctura, care este o multime de date de tip heterogen (octeti, cuvinte, etc.).

***1.2. Structura unui program in limbaj de asamblare pentru microprocesorul Intel 8086***

Un exemplu de program in limbaj de asamblare pentru microprocesorul Intel 8086 este dat in continuare.

Programul efectueaza suma a N=5:

DSEG SEGMENT ; segmentul de date

TAB LABEL BYTE ; eticheta TAB

TABW DW 5 DUP(7000H) ; sirul de cuvinte (tabloul) TABW

REZ DB ? , ? , ? , ? ; sirul de octeti (tabloul) REZ

ZERO EQU 0 ; constanta ZERO

DSEG ENDS

STIVA SEGMENT ; segment de stiva

DW 40 DUP(?)

VIRF LABEL WORD ; eticheta VIRF

STIVA ENDS

CSEG SEGMENT ; segment de cod (program)

START: ; eticheta de inceput a programului

MOV AX, DSEG ; initializari

MOV DS, AX

MOV ES, AX

MOV AX, STIVA

MOV SS, AX

MOV SP, OFFSET VIRF ; initializare pointer de stiva

MOV AX, 0 ; initializare registre utilizate MOV DX, 0

MOV CX, LENGTH TABW ; CX = 5 = lungimea TABW

MOV BP, SIZE TABW ; BP = 10 = 5\*2 = dimensiunea TABW

NEXT:

SUB BP, TYPE TABW ; BP = BP - 2

ADD AX, DS: TABW [BP] ; AX = AX + cuvant curent din TABW

ADC DX, ZERO ; DX = DX + transportul anterior

LOOP NEXT ; CX = CX-1

; daca CX <> 0 salt la NEXT

; altfel trece la instruct. urmatoare

MOV WORD PTR REZ, AX ; stocarea LSW la adresa REZ

MOV WORD PTR REZ + 2 , DX ; stocare MSW la adresa REZ+2

MOV AH, 4CH

; functie DOS 4CH (terminare proces)

INT 21H ; intrerupere SW - apel fct. DOS 4CH

CSEG ENDS

END START ; sfarsitul programului

***1.3. Instructiunile limbajelor de asamblare pentru microprocesorul I8086***Un program in limbaj de asamblare este alcatuit din mai multe linii sursa. O linie sursa este  
alcatuita din urmatoarele cimpuri:

| Eticheta: | Codul operatiei Operanzi de tip imediat | ;Comentariu |  
 (mnemonica)

unde intre | | sunt cuprinse elementele optionale. Cimpul eticheta este facultativ si reprezinta numele simbolic al adresei din memorie la care se afla codul numeric al unei instructiuni. Mnemonica codului de operatie este numele simbolic al instructiunii. Cimpul operanzi poate contine doi operanzi, unul sau nici unul, in functie de tipul instructiunii. Operanzii pot fi datele asupra carora actioneaza instructiunea, adrese sau alte informatii auxiliare. Comentariul este optional. El serveste doar la marirea inteligibilitatii programului. La asamblare textul comentariului este ignorat de catre asamblor.

***1.4. Operatorii asamblorului***

Asupra operanzilor instructiunilor pot fi efectuate operatii si de catre programul asamblor. Acest fel de operatii sunt specificate prin operatori si sunt efectuate de catre asamblor odata cu asamblarea textului sursa. Operatorii utilizati de catre asamblorul TASM sunt de mai multe tipuri:

**Operatori modificatori de valori:**

a. Operatori aritmetici (pentru operatiile aritmetice elementare: +, -, \*, /, MOD).

Exemplu:

MOV WORD PTR REZ + 2 , DX

Asamblorul efectueaza operatia de adunare intre adresa REZ (componenta de tip offset) si 2, rezultatul fiind utilizat de catre codul obiect.

b. Operatorul de indexare [ ], care permite este utilizat la adresare.

Exemplu:

ADD AX, DS: TABW [BP]

Asamblorul utilizeaza registrul BP pentru a calcula adresa efectiva TABW + BP, necesara pentru obtinerea adresei fizice din adresa logica segment (DS) : offset (TABW + BP).

***1.5. Directivele asamblorului***

Un program scris in limbaj de asamblare contine alaturi de instructiunile propriu-zise si pseudo-instructiuni (directive), care se adreseaza programului asamblor modificindu-i modul de lucru sau permitind specificarea datelor.

1. Directivele care specifica segmentele utilizate sunt:

a. Directiva SEGMENT - folosita pentru a marca inceputul unui segment (ex. DSEG, CSEG, STIVA);

Exemple:

DSEG SEGMENT

defineste un segment de date numit DSEG,

STIVA SEGMENT

defineste un segment de stiva (neutilizat aici) numit STIVA,

CSEG SEGMENT

defineste un segment de cod (de program) numit CSEG.

b. Directiva ENDS - folosita pentru a marca sfirsitul segmentului;

Exemple:

DSEG ENDS

declara sfirsitul segmentului de date DSEG,

STIVA ENDS

declara sfirsitul segmentului de stiva STIVA,

CSEG ENDS

declara sfirsitul segmentului de cod CSEG.

2. Directiva EQU permite declararea constantelor. Constantele astfel declarate sunt inlocuite cu numere propriu-zise in momentul asamblarii. Pentru ele nu se aloca spatiu de memorie.

Exemplu:

ZERO EQU 0

Asamblorul inlocuieste in program constanta ZERO cu valoarea numerica 0.

3. Pentru declararea variabilelor se utilizeaza urmatoarele directive:

a. Directiva DB (Define Byte) declara octeti sau siruri de octeti.

Exemplu:

REZ DB ? , ? , ? , ?

declara o variabila REZ de tip sir de octeti formata din 4 octeti neinitializati

b. Directiva DW (Define Word) declara cuvinte sau siruri de cuvinte.

Exemple:

TABW DW 5 DUP(7000H)

declara o variabila TABW de tip sir de cuvinte formata din 5 cuvinte identice initializate cu

7000H DW 40 DUP(?)

declara un sir de 40 de cuvinte neinitializate fara nume (simpla alocare)

c. Directiva DD (Define Double-word) declara dublu-cuvinte sau siruri de dublu-cuvinte. Variabilele sint definite ca rezidente la o anumita adresa relativa (offset) in cadrul unui anumit segment si sunt caracterizate de tipul datelor. Se observa ca pentru rezervarea memoriei variabilelor neinitializate se utilizeaza operatorul:

? = rezervare zona de memorie pentru variabila neinitializata iar pentru precizarea valorilor multiple ale variabilelor se utilizeaza operatorul:

DUP = precizare numar replici (DUPlicate).

4. Declararea etichetelor utilizate pentru referirea la date din segmentele de date sau stiva (etichete adrese de variabile) se face cu directiva LABEL, eventual precizand si atributul.

Exemple:

TAB LABEL BYTE

defineste o eticheta cu numele TAB inaintea variabilei TABW, permitand accesul la variabila TABW octet cu octet (varianta pentru BYTE PTR !),

VIRF LABEL WORD

defineste eticheta VIRF.

Etichetele sunt (ca si variabilele) nume simbolice de adrese. Ele sunt caracterizate de un anumit offset in cadul unui segment. In general etichetele identifica instructiunile. In acest caz etichetele pot fi referite in alte instructiuni pentru executarea salturilor in program. Daca referirile la o eticheta sunt facute in cadrul segmentului in care ea este definita ("home segment") atunci se spune ca ea are atributul NEAR. O eticheta poate fi referita intr-o instructiune a altui segment logic daca poarta atributul FAR. Atributele etichetelor se stabilesc la definirea acestora. Pentru declararea etichetelor in segmentul de program (etichete adrese de instructiuni) se utilizeaza si operatorul: :

Exemple:

START:

declara eticheta de inceput a codului executabil

NEXT:

declara eticheta NEXT utilizata pentru iteratii

5. Directiva END, utilizata pentru declararea sfarsitului de program (cod).

Exemplu:

END START

indica asamblorului ca programul inceput la eticheta START se sfarseste.

6. Directiva RECORD, utilizata pentru definirea inregistrarilor de date. Formatul declaratiei de definire a unei inregistrari este:

nume\_inreg RECORD nume\_camp : expresie | = expresie' | |,..|

unde:

numele campurilor (nume\_camp, ...) sunt unice;

expresia expresie este evaluata la o constanta intreaga intre 1 si 16 si specifica numarul de biti (lungimea) ai campului; suma lungimilor trebuie sa fie mai mica sau egala cu 16, respectiv 8; expresie' este optionala si serveste ca valoare initiala a campului; |,... | indica repetarea optionala a lui nume\_camp : expresie | = expresie' |

Exemplul 1:

CHIPS RECORD RAM:7, EPROM:4, ROM:5

defineste o inregistrare cu 16 biti formata din 3 campuri cu numele RAM, EPROM si ROM avand lungimile de 7, 4, si respectiv 5 biti.

Exemplul 2:

CHIPS RECORD RAM:7=4, EPROM:4=2, ROM:5=0

defineste o inregistrare cu 16 biti formata din 3 campuri cu numele RAM, EPROM si ROM avand lungimile de 7, 4, si respectiv 5 biti, cu precizarea valorilor initiale ale campurilor.

7. Directiva STRUCT, utilizata pentru definirea structurilor. Formatul declaratiei de definire a unei structuri este:

nume\_structura STRUCT

| nume\_camp | {DB | DW | DD} expresie | |, ..|

nume\_structura ENDS

unde:

numele campurilor (nume\_camp, ...) sunt unice;

|,... | indica repetarea optionala a lui nume\_camp {DB | DW | DD} expresie

Exemplu:

PROCES STRUCT

STARE DB 0

VAL\_CRT DW ?

PROCES ENDS

**2. Operanzii instructiunilor microprocesorului Intel 8086**

Microprocesorul Intel 8086 utilizeaza mai multe moduri de adresare a datelor, numite generic operanzi (sursa sau destinatie), in functie de locul in care sunt plasati acestia. Operanzii pot fi continuti in registrii, in memorie, in codul instructiunii sau in porturile de intrare/iesire. Operanzii din registre permit, datorita plasarii lor interne, ca instructiunile care ii utilizeaza sa fie executate mai rapid, nemaifiind necesare transferuri cu memoria sau porturile. Registrele (de 8 sau 16 biti) pot fi operanzi sursa, operanzi destinatie sau operanzi sursa si destinatie. Operanzii de tip imediat sunt date constante de 8 sau 16 biti continute in codul instructiunii, dupa codul operatiei. Accesul la acesti operanzi este destul de rapid, deoarece ei sunt incarcati in coada de instructiuni de catre unitatea de interfata odata cu instructiunea. Limitarile operanzilor de tip imediat se datoreaza faptului ca ei pot fi doar valori constante si pot servi doar ca operanzi sursa. Operanzii din memorie sunt accesati mai lent, deoarece sunt necesare: mai intai calculul adresei efective a operandului (EA = offset-ul), apoi calculul adresei fizice si in final transferul lor.

Formula generala de calcul pentru EA (offset) este:

EA = (BX / BP)\* + (SI / DI)\* + (D8 / D16)\*

unde: \* indica un termen optional, iar

/ separa variantele unui termen.

Observatie: spatiul de adrese al porturilor este 0...FFFFH (64ko)

***2.1. Operanzi de tip imediat (adresarea imediata)***

Operandul este explicit in codul instructiunii, dupa codul operatiei, aflandu-se in coada de instructiuni.

Exemplul 1. In instructiunea:

ADD AL, 6

operandul al doilea este constanta numerica 6 de tip imediat.

Exemplul 2. In instructiunea:

ADC ALFA, AX

primul operand este constanta simbolica ALFA de tip imediat.

***2.2. Operanzi din registre (adresarea directa la registru)***

Operandul este in registrul specificat in codul instructiunii, in codul operatiei.

Exemplu. In instructiunea:

ADD AL, BL

ambii operanzi sunt de tip regsitru.

***2.3. Operanzi din memorie (adresarea la memorie)***

1. Adresarea directa la memorie.

Operandul este in memoria principala (MP), la offset-ul specificat in codul instructiunii (dupa codul operatiei). Adresa efectiva se afla in codul instructiunii.

Exemplu. In instructiunea:

ADD AL, VAR

al doilea operand este variabila VAR (inlocuita la asamblare cu offset-ul variabilei VAR) declarata cu o directiva. Operandul este continutul variabilei VAR din memorie.

2. Adresare indirecta la memorie prin registru.

Operandul este in memoria principala (MP), la offset-ul aflat in registru de baza (BX, BP) sau index (SI, DI) specificat in codul instructiunii, in codul operatiei.

Exemplul 1. In instructiunea:

ADD AX, [BX]

al doilea operand este perechea de octeti (cuvantul) din memoria principala, din segmentul curent de date (specificat de DS), aflat la offset-ul continut in registrul BX.

Exemplul 2. In instructiunea:

ADD [SI], AL

primul operand este octetul din MP, din segmentul de date curent (specificat de DS), aflat la offsetul continut in registrul SI.

3. Adresare indirecta la memorie prin registru cu deplasare.

Operandul este in memoria principala MP, la offset-ul calculat ca suma continutului registrului de baza sau index specificat in codul operatiei si deplasarea care urmeaza in codul instructiunii dupa codul operatiei.

Exemplul 1. In instructiunea:

ADD AX, DS: [BP+2]

al doilea operand este cuvantul din MP, din segmentul de date curent (specificat explicit de DS), aflat la offset-ul egal cu suma dintre continutul registrului BX si deplasarea 2 (de tip imediat).

Exemplul 2. In instructiunea:

ADD TAB [DI], AL

primul operand este cuvantul din MP, din segmentul de date curent (specificat de DS), aflat la offset-ul egal cu suma dintre continutul registrului DI si deplasarea TAB (de tip imediat) .

4. Adresare indirecta la memorie prin doua registre.

Operandul este in memoria principala MP, la offset-ul calculat ca suma continuturilor registrelor de baza si index specificate in codul operatiei.

Exemplu. In instructiunea:

ADD AX, [BX] [ SI]

al doilea operand este cuvantul din MP, din segmentul de date curent, aflat la offset-ul egal cu suma continuturilor registrelor BX si SI.

5. Adresare indirecta la memorie prin doua registre cu deplasare.

Operandul este in memoria principala MP, la offset-ul calculat ca suma continuturilor registrelor de baza si index specificate in codul operatiei si deplasarea care urmeaza codul operatiei

Exemplu. In instructiunea:

MOV MATR [BX] [SI], AX

primul operand este cuvantul din MP, din segmentul de date curent, aflat la offset-ul egal cu suma continuturilor registrelor BX si SI cu deplasarea MATR (de tip offset).

***2.4. Operanzi din porturi de intrare-iesire (adresarea porturilor)***

1. Adresarea directa a porturilor (pentru adrese pe 8 biti, in gama 0..255).

Operandul se afla in portul aflat la adresa specificata in instructiune dupa codul operatiei.

Exemplul 1. In instructiunea:

IN AX, 20H

al doilea operand este portul de intrare aflat la adresa numerica 20H (de tip imediat).

Exemplul 2. In instructiunea:

OUT PORT2, AL

primul operand este portul de iesire aflat la adresa simbolica PORT2 (de tip imediat).

2. Adresarea indirecta a porturilor prin registrul DX.

Operandul se afla in portul aflat la adresa specificata in registrul DX.

Exemplu. In instructiunea:

OUT DX, AL

primul operand este portul de iesire aflat in portul de la adresa continuta in DX.

**3. Instructiuni de transfer**

Instructiunile de transfer intre registre sau intre un registru si memorie realizeaza operatiile de atribuire (copiere) si de permutare. Operatii de transfer pot fi realizate si intre registre si porturi.

***3.1. Instructiunea de atribuire (copiere) are forma:***

MOV operand1, operand2

si efectul:

operand1 = operand2

Se observa ca sensul in care se face atribuirea este de la dreapta la stanga (sens utilizat in general pentru scrierea functiilor si a expresiilor matematice). Astfel, se poate spune ca forma instructiunii este:

MOV destinatie, sursa

Exemplu:

MOV AX, BX

copiaza in registrul AX valoarea continuta in registrul BX

***3.2 Instructiunea de permutare (inter-schimbare) are forma*:**

XCHG operand1, operand2

si efectul:

temp = operand1

operand1= operand2

operand2 = temp

Astfel, cei doi operanzi isi schimba continutul intre ei, operanzii fiind si sursa si destinatie. Exemplu:

MOV CX, BX

copiaza in registrul CX valoarea continuta in registrul BX, si in registrul BX valoarea initiala a lui CX

***3.3 Instructiunea de citire dintr-un port are forma****:*

IN acumulator, port

si efectul:

acumulator = port

***3.4 Instructiunea de scriere intr-un port are forma***:

OUT port, accumulator

si efectul:

port = accumulator

**4. Instructiuni aritmetice**

Instructiunile aritmetice ale microprocesorului Intel 8086 utilizeaza 1 sau 2 operanzi.

***4.1. Instructiuni aritmetice care utilizeaza 2 operanzi***

Instructiunile care utilizeaza 2 operanzi sunt: adunari (cu sau fara bitul Carry de la operatia anterioara), scaderi (cu sau fara bitul Carry de la operatia anterioara), inmultiri, impartiri. In cele ce urmeaza operand1 este sursa si destinatie iar operand2 este doar sursa.

1. Instructiunea de adunare are forma:

ADD operand1, operand2

si efectul: operand1 = operand1 + operand2

Exemplu (initial AX = 7FFFH si BX = 8000H):

ADD BX, DX

Efect:

1000 0000 0000 0000 B = 8000 H +

1100 0000 0000 0000 B = C000 H

--------------------------------------------------

(CF = 1) 0100 0000 0000 0000 B = 4000 H

Valori in urma executiei instructiunii: BX = 4000H, CF = 1 (transport).

2. Instructiunea de adunare cu bitul Carry (transport) de la operatia anterioara are forma:

ADC operand1, operand2

si efectul: operand1 = operand1 + operand2 + CF

Exemplu (initial AX = 7FFFH, CX = 4000H si CF = 1):

ADD AX, CX

Efect:

0111 1111 1111 1111 B = 7FFF H +

0100 0000 0000 0000 B = 4000 H +

1 B = 1 H

--------------------------------------------------

(CF = 0) 1100 0000 0000 0000 B = C000 H

Valori in urma executiei instructiunii: AX = C000H, CF = 0.

3. Instructiunea de scadere are forma:

SUB operand1, operand2

si efectul: operand1 = operand1 - operand2

Exemplu (initial BX = 8000H si DX = 0C000H):

SUB BX, DX

Efect:

1000 0000 0000 0000 B = 8000 H -

1100 0000 0000 0000 B = C000 H

--------------------------------------------------

(CF = 1) 1100 0000 0000 0000 B = C000 H

Valori in urma executiei instructiunii: BX = C000H, CF = 1 (imprumut).

4. Instructiunea de scadere cu bitul Carry (imprumut) de la operatia anterioara are forma:

SBB operand1, operand2

si efectul: operand1 = operand1 - operand2 – CF

Exemplu (initial AX = 7FFFH, CX = 4000H, si CF = 1):

SBB AX, CX

0111 1111 1111 1111 B = 7FFF H -

0100 0000 0000 0000 B = 4000 H -

1 B = 1 H

--------------------------------------------------

(CF = 0) 0011 1111 1111 1110 B = 3FFE H

Valori in urma executiei instructiunii: AX = 3FFEH, CF = 0.

5. Instructiunea de inmultire intre numere fara semn are forma:

MUL operand

si efectul: - pentru operanzi de 8 biti: AX = AL \* operand

- pentru operanzi de 16 biti: DX, AX = AX \* operand

6. Instructiunea de inmultire intre numere cu semn are forma:

IMUL operand

si efectul: - pentru operanzi de 8 biti: AX = AL \* operand

- pentru operanzi de 16 biti: DX, AX = AX \* operand

7. Instructiunea de impartire intre numere fara semn are forma:

DIV operand

si efectul: - pentru operanzi de 8 biti: AL = AX / operand (catul)

AH = AX MOD operand (restul)

- pentru operanzi de 16 biti: AX = DX, AX / operand

DX = DX, AX MOD operand

8. Instructiunea de impartire intre numere cu semn are forma:

IDIV operand

si efectul: - pentru operanzi de 8 biti: AL = AX / operand

AH = AX MOD operand

- pentru operanzi de 16 biti: AX = DX, AX / operand

DX = DX, AX MOD operand

***4.2. Instructiuni aritmetice care utilizeaza 1 operand***

Instructiunile care utilizeaza 1 operand (sursa si destinatie) sunt: incrementare, decrementare, negare (aritmetica, in complement fata de 2).

1. Instructiunea de incrementare are forma:

INC operand

si efectul: operand = operand + 1

Exemplu (initial DI = 0FFFFH):

INC DI

Efect:

1111 1111 1111 1111 B = FFFF H +

0000 0000 0000 0001 B = 1 H

--------------------------------------------------

(CF = 1) 0000 0000 0000 0000 B = 0000 H

Valori in urma executiei instructiunii: DI = 0000H, CF = 1 (transport).

2. Instructiunea de decrementare are forma:

DEC operand

si efectul: operand = operand – 1

Exemplu (initial SI = 0001H):

DEC SI

Efect:

0000 0000 0000 0001 B = 0001 H -

0000 0000 0000 0001 B = 1 H

--------------------------------------------------

(CF = 0) 0000 0000 0000 0000 B = 0000 H

Valori in urma executiei instructiunii: SI = 0000H, CF = 0.

3. Instructiunea de negare are forma:

NEG operand

si efectul: operand = (0) – operand

Exemplul 1 (initial SI = 0001H):

NEG SI

Efect:

(0000 0000 0000 0000 B = 0000 H)

- 0000 0000 0000 0001 B = 0001 H

--------------------------------------------------

(CF = 1) 1111 1111 1111 1111 B = FFFF H

Valori in urma executiei instructiunii: SI = FFFFH, CF = 1 (transport).

Exemplul 2 (initial DI = 0FFFFH):

NEG DI

Efect:

(0000 0000 0000 0000 B = 0000 H)

- 1111 1111 1111 1111 B = FFFF H

--------------------------------------------------

(CF = 1) 0000 0000 0000 0001 B = 0001 H

Valori in urma executiei instructiunii: DI = 0001H, CF = 1 (transport).

Exemplul 3 (initial AX = 7FFFH):

NEG AX

Efect:

(0000 0000 0000 0000 B = 0000 H)

- 0111 1111 1111 1111 B = 7FFF H

--------------------------------------------------

(CF = 1) 1000 0000 0000 0001 B = 8001 H

Valori in urma executiei instructiunii: AX = 8001H, CF = 1 (transport).

Exemplul 4 (initial BX = 8000H):

NEG BX

Efect:

(0000 0000 0000 0000 B = 0000 H)

- 1000 0000 0000 0000 B = 8000 H

--------------------------------------------------

(CF = 1) 1000 0000 0000 0000 B = 8000 H

Valori in urma executiei instructiunii: BX = 8000H, CF = 1 (transport).

Exemplul 5 (initial CX = 4000H):

NEG CX

Efect:

(0000 0000 0000 0000 B = 0000 H)

- 0100 0000 0000 0000 B = 4000 H

--------------------------------------------------

(CF = 1) 1100 0000 0000 0000 B = C000 H

Valori in urma executiei instructiunii: CX = C000H, CF = 1 (transport).

Exemplul 6 (initial DX = 0C000H):

NEG DX

Efect:

(0000 0000 0000 0000 B = 0000 H)

- 1100 0000 0000 0000 B = C000 H

--------------------------------------------------

(CF = 1) 0100 0000 0000 0000 B = 4000 H

Valori in urma executiei instructiunii: DX = 4000H, CF = 1 (transport).

**5. Instructiuni logice**

***5.1. Instructiuni logice care utilizeaza 2 operanzi***

Instructiunile care utilizeaza 2 operanzi sunt: functia si logic intre bitii operanzilor (AND),

functia sau logic intre bitii operanzilor (OR), functia sau-exclusiv logic intre bitii operanzilor

(XOR).

In cele ce urmeaza operand1 este sursa si destinatie iar operand2 este doar sursa.

1. Instructiunea AND are forma:

AND operand1, operand2

si efectul: operand1 = operand1 AND operand2

Exemplu (initial AX = 5555H si BX = 6666H):

AND AX, BX

Efect:

0101 0101 0101 0101 B = 5555 H x

0110 0110 0110 0110 B = 6666 H

-------------------------------------------------

0100 0100 0100 0100 B = 4444 H

Valoare in urma executiei instructiunii: AX = 4444H.

2. Instructiunea OR are forma:

OR operand1, operand2

si efectul: operand1 = operand1 OR operand2

Exemplu (initial AX = 5555H si BX = 6666H):

OR AX, BX

Efect:

0101 0101 0101 0101 B = 5555 H +

0110 0110 0110 0110 B = 6666 H

-------------------------------------------------

0111 0111 0111 0111 B = 7777 H

Valoare in urma executiei instructiunii: AX = 7777H.

3. Instructiunea XOR are forma:

XOR operand1, operand2

si efectul: operand1 = operand1 XOR operand2

Exemplu (initial AX = 5555H si BX = 6666H):

XOR AX, BX

Efect:

0101 0101 0101 0101 B = 5555 H +

0110 0110 0110 0110 B = 6666 H

-------------------------------------------------

0011 0011 0011 0011 B = 3333 H

Valoare in urma executiei instructiunii: AX = 3333H.

***5.2. Instructiuni logice care utilizeaza 1 operand***

Instructiunea care utilizeaza 1 operand (sursa si destinatie) este negarea logica (NOT) a

bitilor operandului (reprezentarea in complement fata de 1 - C1).

In cele ce urmeaza operand este sursa si destinatie.

Instructiunea NOT are forma:

NOT operand

si efectul: operand = NOT operand

Exemplul 1 (initial SI = 0001H):

NOT SI

Efect:

0000 0000 0000 0001 B = 0001 H \

--------------------------------------------------

1111 1111 1111 1110 B = FFFE H

Valoare in urma executiei instructiunii: SI = FFFEH.

Exemplul 2 (initial DI = 0FFFFH):

NOT DI

Efect: 1111 1111 1111 1111 B = FFFF H \

--------------------------------------------------

0000 0000 0000 0000 B = 0000 H

Valoare in urma executiei instructiunii: DI = 0000H.

Exemplul 3 (initial AX = 7FFFH):

NOT AX

Efect:

0111 1111 1111 1111 B = 7FFF H \

--------------------------------------------------

1000 0000 0000 0000 B = 8000 H

Valoare in urma executiei instructiunii: AX = 8000H.

Exemplul 4 (initial BX = 8000H):

NOT BX

Efect:

1000 0000 0000 0000 B = 8000 H \

--------------------------------------------------

0111 0111 0111 0111 B = 7FFF H

Valoare in urma executiei instructiunii: BX = 7FFFH.

Exemplul 5 (initial CX = 4000H):

NOT CX

Efect:

0100 0000 0000 0000 B = 4000 H \

--------------------------------------------------

1100 0000 0000 0000 B = C000 H

Valoare in urma executiei instructiunii: CX = C000H.

Exemplul 6 (initial DX = 0C000H):

NOT DX

Efect:

1100 0000 0000 0000 B = C000 H \

--------------------------------------------------

0011 1111 1111 1111 B = 3FFF H

Valoare in urma executiei instructiunii: DX = 3FFFH.

**6. Instructiuni de deplasare si rotatie**

***6.1. Instructiuni de deplasare***

Instructiunile de deplasare executa operatii asupra unui singur operand (sursa si destinatie).

In cele ce urmeaza operand este sursa si destinatie.

1. Instructiunea de deplasare logica la stanga are forma:

SHL operand, 1

si efectul: operand = operand deplasat la stanga cu o pozitie

sau SHL operand, CL

respectiv operand = operand deplasat la stanga cu (CL) poz.

Exemplul 1 (initial AX = AAAAH):

SHL AX, 1

Efect: 1

1010 1010 1010 1010 B = AAAA H (<-----)

------------------------------------------------------------

0101 0101 0101 0100 B = 5 5 5 4 H

Valori in urma executiei instructiunii: AX = 5554H, CF = 1.

Exemplul 2 (initial AX = AAAAH si CL = 4):

SHL AX, CL

Efect: (CL)=4

1010 1010 1010 1010 B = AAAA H (<-----)

------------------------------------------------------------

1010 1010 1010 0000 B = AAA0 H

Valori in urma executiei instructiunii: AX = AAA0H, CF = 0.

Se observa ca deplasarea logica la stanga cu n pozitii are acelasi efect cu inmultirea cu 2n.

Astfel, pentru primul exemplu avem (n = 1, 2n = 21 = 2):

AAAA H \* 2 H = 1 5 5 5 4 H

iar pentru al doilea (n = 4, 2n = 24 = 16 = 10 H):

AAAA H \* 10 H = AAAA0 H

Se observa de asemenea ca in primul caz rezultatul corect este continut in CF (1) si AX

(5554H), pe cand in al doilea caz se pierd primii biti ai rezultatului.

2. Instructiunea de deplasare logica la dreapta are forma:

SHR operand, 1

si efectul: operand = operand deplasat la dreapta cu o pozitie

sau SHR operand, CL

respectiv operand = operand deplasat la dreapta cu (CL) poz.

Exemplul 1 (initial AX = AAAAH):

SHR AX, 1

Efect: 1

1010 1010 1010 1010 B = AAAA H (----->)

------------------------------------------------------------

0101 0101 0101 0101 B = 5 5 5 5 H

Valori in urma executiei instructiunii: AX = 5555H, CF = 0.

Exemplul 2 (initial AX = AAAAH si CL = 4):

SHR AX, CL

Efect: (CL)=4

1010 1010 1010 1010 B = AAAA H (----->)

------------------------------------------------------------

0000 1010 1010 1010 B = 0AAA H

Valori in urma executiei instructiunii: AX = 0AAAH, CF = 1.

Se observa ca deplasarea logica la dreapta cu n pozitii are acelasi efect cu impartirea la 2n .

Astfel, pentru primul exemplu avem (n = 1, 2n = 21 = 2):

AAAA H : 2 H = 5 5 5 5 H

iar pentru al doilea (n = 4, 2n = 24 = 16 = 10 H):

AAAA H : 10 H = AAA H

3. Instructiunea de deplasare aritmetica la stanga are forma:

SAL operand, 1

si efectul: operand = operand deplasat la stanga cu o pozitie

sau SAL operand, CL

respectiv operand = operand deplasat la stanga cu (CL) poz.

Exemplul 1 (initial AX = AAAAH):

SAL AX, 1

Efect: 1

1010 1010 1010 1010 B = AAAA H (<-----)

-----------------------------------------------------------

0101 0101 0101 0100 B = 5 5 5 4 H

Valori in urma executiei instructiunii: AX = 5554H, CF = 1.

Exemplul 2 (initial AX = AAAAH si CL = 4):

SAL AX, CL

Efect: (CL)=4

1010 1010 1010 1010 B = AAAA H (<-----)

------------------------------------------------------------

1010 1010 1010 0000 B = AAA 0 H

Valori in urma executiei instructiunii: AX = AAA0H, CF = 0.

Se observa faptul ca efectul acestei instructiuni este identic cu cel al instructiunii de

deplasare logica la stanga !.

4. Instructiunea de deplasare aritmetica la dreapta (cu pastrarea semnului) are forma:

SAR operand, 1

si efectul: operand = operand depl. aritmetic la dr. cu o poz.

sau SAR operand, CL

respectiv operand = operand depl. aritmetic la dr cu (CL) poz

Exemplul 1 (initial AX = AAAAH):

SAR AX, 1

Efect: 1

1010 1010 1010 1010 B = AAAA H (----->)

------------------------------------------------------------

1101 0101 0101 0101 B = D5 5 5 H

Valori in urma executiei instructiunii: AX = D555H, CF = 0.

Exemplul 2 (initial AX = AAAAH si CL = 4):

SAR AX, CL

Efect: (CL)=4

1010 1010 1010 1010 B = AAAA H (----->)

------------------------------------------------------------

1111 1010 1010 1010 B = FAAA H

Valori in urma executiei instructiunii: AX = FAAAH, CF = 1.

***6.2. Instructiuni de rotatie***

Instructiunile de rotatie executa operatii asupra unui singur operand (sursa si destinatie).

In cele ce urmeaza operand este sursa si destinatie.

1. Instructiunile de rotatie la stanga fara CF au forma:

ROL operand, 1

si efectul: operand = operand rotit spre stanga cu o pozitie

sau ROL operand, CL

respectiv operand = operand rotit spre stanga cu (CL) poz.

Exemplul 1 (initial AX = AAAAH):

ROL AX, 1

Efect:

1010 1010 1010 1010 B = AAAA H (rot. stg. cu 1)

-------------------------------------------------------------------

0101 0101 0101 0101 B = 5 5 5 5 H

Valori in urma executiei instructiunii: AX = 5555H, CF = 1.

Exemplul 2 (initial AX = AAAAH si CL = 4):

ROL AX, CL

Efect:

1010 1010 1010 1010 B = AAAA H (rot. stg. cu CL)

----------------------------------------------------------------------

1010 1010 1010 1010 B = AAAA H

Valori in urma executiei instructiunii: AX = AAAAH, CF = 0.

2. Instructiunile de rotatie la stanga cu CF au forma:

RCL operand, 1

si efectul: operand = (operand,CF) rotit spre stanga cu o poz.

sau RCL operand, CL

respectiv operand = (operand,CF) rotit spre stg. cu (CL) poz.

Exemplul 1 (initial AX = AAAAH si CF = 1):

RCL AX, 1

Efect:

(CF = 1) 1010 1010 1010 1010 B = AAAA H (rot. stg. cu 1)

--------------------------------------------------------------------

(CF = 1) 0101 0101 0101 0101 B = 5 5 5 5 H

Valori in urma executiei instructiunii: AX = 5555H, CF = 1.

Exemplul 2 (initial AX = AAAAH si CF = 0):

RCL AX, 1

Efect:

(CF = 0) 1010 1010 1010 1010 B = AAAA H (rot. stg. cu 1)

--------------------------------------------------------------------

(CF = 1) 0101 0101 0101 0100 B = 5 5 5 4 H

Valori in urma executiei instructiunii: AX = 5554H, CF = 1.

Exemplul 3 (initial AX = AAAAH, CL = 4 si CF = 1):

RCL AX, CL

Efect:

(CF = 1) 1010 1010 1010 1010 B = AAAA H (rot. stg. cu CL)

----------------------------------------------------------------

(CF = 0) 1010 1010 1010 1101 B = AAAD H

Valori in urma executiei instructiunii: AX = AAADH, CF = 0.

Exemplul 4 (initial AX = AAAAH, CL = 4 si CF = 0):

RCL AX, CL

Efect:

(CF = 0) 1010 1010 1010 1010 B = AAAA H (rot. stg. cu CL)

----------------------------------------------------------------------

(CF = 0) 1010 1010 1010 0101 B = AAA5 H

Valori in urma executiei instructiunii: AX = AAA5H, CF = 0.

3. Instructiunile de rotatie la dreapta fara CF au forma:

ROR operand, 1

si efectul: operand = operand rotit spre dreapta cu o pozitie

sau ROR operand, CL

respectiv operand = operand rotit spre dreapta cu (CL) poz.

Exemplul 1 (initial AX = AAAAH):

ROR AX, 1

Efect:

1010 1010 1010 1010 B = AAAA H (rot. dr. cu 1)

-------------------------------------------------------------------

0101 0101 0101 0101 B = 5 5 5 5 H

Valori in urma executiei instructiunii: AX = 5555H, CF = 0.

Exemplul 2 (initial AX = AAAAH si CL = 4):

ROR AX, CL

Efect:

1010 1010 1010 1010 B = AAAA H (rot. dr. cu CL)

-------------------------------------------------------------------

1010 1010 1010 1010 B = AAAA H

Valori in urma executiei instructiunii: AX = AAAAH, CF = 1.

4. Instructiunile de rotatie la dreapta cu CF au forma:

RCR operand, 1

si efectul: operand = (operand,CF) rotit spre dreapta cu o poz.

sau RCR operand, CL

respectiv operand = (operand,CF) rotit spre dr. cu (CL) poz.

Exemplul 1 (initial AX = AAAAH si CF = 1):

RCR AX, 1

Efect:

1010 1010 1010 1010 (CF = 1) B = AAAA H (rot. dr. cu 1)

------------------------------------------------------------------------

1101 0101 0101 0101 (CF = 0) B = D 5 5 5 H

Valori in urma executiei instructiunii: AX = D555H, CF = 0.

Exemplul 2 (initial AX = AAAAH si CF = 0):

RCR AX, 1

Efect:

1010 1010 1010 1010 (CF = 0) B = AAAA H (rot. dr. cu 1)

------------------------------------------------------------------------

0101 0101 0101 0101 (CF = 0) B = 5 5 5 5 H

Valori in urma executiei instructiunii: AX = 5555H, CF = 0

Exemplul 3 (initial AX = AAAAH, CL = 4 si CF = 1):

RCR AX, CL

Efect:

1010 1010 1010 1010 (CF = 1) B = AAAA H (rot. dr. cu CL)

---------------------------------------------------------------------------

0101 1010 1010 1101 (CF = 1) B = 5AAA H

Valori in urma executiei instructiunii: AX = 5AAAH, CF = 1.

Exemplul 4 (initial AX = AAAAH, CL = 4 si CF = 0):

RCR AX, CL

Efect:

1010 1010 1010 1010 (CF = 0) B = AAAA H (rot. dr. cu CL)

--------------------------------------------------------

1010 1010 1010 0101 (CF = 1) B = AAAA H

Valori in urma executiei instructiunii: AX = AAAAH, CF = 1.

**7. Exemple de programe**

***7.1. Calcule in dubla precizie***

Rolul principal al instructiunilor de adunare cu transport si scadere cu imprumut este acela de a permite efectuarea calculelor in dubla precizie (pe 32 biti).

1. Adunarea a doua valori de 32 biti - perechile de registre (AX, BX), respectiv (CX, DX):

Initial: AX = 7FFFH, BX = 8000H, CX = 4000H si DX = 0C000H.

ADD BX, DX; adunarea LSW

ADC AX, CX ; adunarea MSW

Efectul secventei este:

AX BX +

CX DX

------------

AX BX

adica:

0111 1111 1111 1111 1000 0000 0000 0000 B = 7FFF 8000 H +

0100 0000 0000 0000 1100 0000 0000 0000 B = 4000 C000 H

(CF = 1)

--------------------------------------------------------------------------------------------

(CF = 0) 1100 0000 0000 0000 0100 0000 0000 0000 B = C000 4000 H

In urma executiei secventei: AX = C000H, BX = 4000H, CF = 0 (rezultat C000 4000 H).

2. Scaderea a doua valori de 32 biti - perechile de registre (AX, BX), respectiv (CX, DX):

Initial: AX = 7FFFH, BX = 8000H, CX = 4000H si DX = 0C000H.

SUB BX, DX ; scaderea LSW

SBB AX, CX ; scaderea MSW

Efectul secventei este:

AX BX –

CX DX

------------

AX BX

adica:

0111 1111 1111 1111 1000 0000 0000 0000 B = 7FFF 8000 H -

0100 0000 0000 0000 1100 0000 0000 0000 B = 4000 C000 H

(CF = 1)

--------------------------------------------------------------------------------------------

(CF = 0) 0011 1111 1111 1110 1100 0000 0000 0000 B = 3FFE C000 H

Valori in urma executiei instructiunii: AX = 3FFEH, BX = C000H, CF = 0 (rezultat 3FFE C000 H).

3. Calculul sumei in dubla precizie a variabilelor de tip cuvant a si b. Rezultatul este plasat in varibila c.

DAT SEGMENT

a dw 0a46fh

b dw 0dc89h

c dw ?, ?

DAT ENDS

COD SEGMENT

START:

MOV AX, DAT

MOV DS, AX

mov ax, a ; initializari

mov dx, 0

add ax, b ; suma in dubla precizie

adc dx, 0

mov c, ax ; memorare rezultat

mov [c+2], dx

MOV AH, 4CH

INT 21H

COD ENDS

END START

***7.2. Calcule in dubla precizie prin deplasare***

Rolul principal al bitului de transport in instructiunile de deplasare si rotatie este acela de a

permite efectuarea calculelor in dubla precizie (pe 32 biti).

1. Deplasarea cu o pozitie la stanga a unei valori de 32 biti - perechea de registre (AX, BX):

Initial: AX = AAAAH si BX = 8888H.

SHL AX, 1 ; deplasarea MSW

SHL BX, 1 ; deplasarea LSW

ADC AX, 0 ; adunarea bitului CF la LSW

Efectul secventei este:

1010 1010 1010 1010 1000 1000 1000 1000 B = AAAA 8 8 8 8 H

/ / / / / / / / / / / / / / / / / / / / / / / / / / / / / / / /

---------------------------------------------------------------------------------------------------------

(CF = 1)0101 0101 0101 0101 0001 0001 0001 0000 B = 5 5 5 5 1 1 1 0 H

In urma executiei secventei: AX = 5555H, BX = 1110H, CF = 1 (rezultat 1 5555 1110H)

2. Deplasarea cu o pozitie la dreapta a unei valori de 32 biti - perechea de registre (AX, BX):

Initial: AX = AAAAH si BX = 8888H.

SHR BX, 1 ; deplasarea LSW

SHR AX, 1 ; deplasarea MSW

ADC DX, 0 ; adunarea bitului CF la DX

MOV CL, 15 ; pregatire deplasare 15 biti

SHL DX, CL ; deplasare CF pe pozitia MSb

ADD BX, DX ; adunarea bitului CF la MSW

Efectul secventei este:

1010 1010 1010 1010 1000 1000 1000 1000 B = AAAA 8 8 8 8 H

\ \ \ \ \ \ \ \ \ \ \ \ \ \ \ \ \ \ \ \ \ \ \ \ \ \ \ \ \ \ \ \

---------------------------------------------------------------------------------------------------------

0101 0101 0101 0101 0100 0100 0100 0100 (CF = 0) B = 5 5 5 5 4 4 4 4 H

In urma executiei secventei: AX = 5555H, BX = 4444H, CF = 0 (rezultat 5555 4444H).

3. Rotatia cu o pozitie spre stanga a unei valori de 32 biti - perechea de registre (AX, BX):

Initial: AX = AAAAH, BX = 8888H, DX = 0000H, CL = 4 si CF = 1.

ROL AX, 1 ; rotirea MSW

RCL BX, 1 ; rotirea (LSW,CF)

ADC AX, 0 ; adunarea la MSW a bitului CF

; obtinut prin rotirea lui BX

Efectul secventei este:

1010 1010 1010 1010 1000 1000 1000 1000 B = AAAA 8 8 8 8 H

/ / / / / / / / / / / / / / / / / / / / / / / / / / / / / / / /

---------------------------------------------------------------------------------------------------------

(CF = 1)0101 0101 0101 0101 0001 0001 0001 0001 B = 5 5 5 5 1 1 1 1 H

In urma executiei secventei: AX = 5555H, BX = 1111H, CF = 1 (rezultat 1 5555 1111H).

4. Rotatia cu o pozitie spre dreapta a unei valori de 32 biti - perechea de registre (AX, BX):

Initial: AX = AAAAH, BX = 8888H, DX = 0000H, CL = 4 si CF = 1.

ROR AX, 1 ; rotirea cuvantului superior (MSW)

RCR BX, 1 ; rotirea cuvantului inferior (LSW) cu CF

; de la rotirea lui AX

ADC DX, 0 ; aducerea in DX a lui CF de la rotirea lui BX

ROR DX, 1 ; plasarea in DX a lui CF obtinut prin rotirea

; lui BX, pe cea mai semnificativa pozitie

ADD AX, DX ; adunarea la cuvantul superior a bitului

; CF obtinut prin rotirea lui BX

Efectul secventei este:

1010 1010 1010 1010 1000 1000 1000 1000 B = AAAA 8 8 8 8 H

\ \ \ \ \ \ \ \ \ \ \ \ \ \ \ \ \ \ \ \ \ \ \ \ \ \ \ \ \ \ \ \

---------------------------------------------------------------------------------------------------------

0101 0101 0101 0101 (CF = ) 0100 0100 0100 0100 (CF = 0) B = 5 5 5 5 4 4 4 4 H

In urma executiei secventei: AX = 5555H, BX = 4444H, CF = 0 (rezultat 5555 4444H).

5. Program de deplasare in dubla precizie a continutului unei variabile v cu 2 pozitii catre

dreapta.

data segment

v dw 0abcdh, 0ef12h

data ends

cod segment

st:

mov ax, data

mov ds, ax

mov ax, x ; initializari

mov bp, offset v

mov ax, ds:[bp]

mov bx, ds:[bp+2]

mov dx, 0 ; prima deplasare

shr ax, 1

shr bx, 1

adc dx, 0

ror dx, 1

adc ax, 0

mov dx, 0 ; a doua deplasare

shr ax, 1

shr bx, 1

adc dx, 0

ror dx, 1

adc ax, 0

mov ds:[bp], ax ; stocare rezultat

mov ds:[bp+2], bx

mov ah, 4ch

int 21h

cod ends

end st

**8. Alte programe**

1. Program de transformare a unui caracter litera mica citit de la tastatura in caracter litera mare afisat pe ecran.

data segment

numeprog db 25 dup(0ah),'Transformare caracter$'

citire db 2 dup(0ah),0dh,' Introduceti litera mica $'

afisare db 2 dup(0ah),0dh,' Litera mare este: $'

data ends

cod segment

start:

mov ax, data

mov ds, ax

mov dx, offset numeprog ; afisare sir caractere

mov ah, 9 ; (nume program)

int 21h

mov dx, offset citire ; afisare sir caractere

mov ah, 9 ; (mesaj citire)

int 21h

mov ah, 1 ; citire caracter cu ecou pe

int 21h ; ecran (litera mica)

sub al, 20h ; conversie litera mica -> litera mare

mov bl, al

mov dx, offset afisare ; afisare sir caractere

mov ah, 9 ; (mesaj afisare)

int 21h

mov dl, bl

mov ah, 2 ; afisare caracter

int 21h ; (litera mare)

mov ah, 8 ; citire caracter fara ecou pe

int 21h ; ecran (Enter)

mov ah, 4ch ; exit

int 21h

cod ends

end start

2. Program de transformare a unui caracter litera mare citit de la tastatura in caracter litera mica afisat pe ecran.

data segment

numeprog db 25 dup(0ah),'Transformare caracter$'

citire db 2 dup(0ah),0dh,' Introduceti litera mare $'

afisare db 2 dup(0ah),0dh,' Litera mica este: $'

data ends

cod segment

start:

mov ax, data

mov ds, ax

mov dx, offset numeprog ; afisare sir caractere

mov ah, 9 ; (nume program)

int 21h

mov dx, offset citire ; afisare sir caractere

mov ah, 9 ; (mesaj citire)

int 21h

mov ah, 1 ; citire caracter cu ecou pe

int 21h ; ecran (litera mare)

add al, 20h ; conversie litera mare -> litera mica

mov bl, al

mov dx, offset afisare ; afisare sir caractere

mov ah, 9 ; (mesaj afisare)

int 21h

mov dl, bl

mov ah, 2 ; afisare caracter

int 21h ; (litera mica)

mov ah, 8 ; citire caracter fara ecou pe

int 21h ; ecran (Enter)

mov ah, 4ch ; exit

int 21h

cod ends

end start

**9. Teme si exercitii**  
1. Sa se scrie un program care sa utilizeze calculul in dubla precizie pentru a calcula suma a  
doua cuvinte, aflate in registrele AX si DX, folosind numai registrele de tip octet (AH, AL, BH, BL,  
CH, CL, DH, DL).  
2. Sa se scrie un program care sa transforme caracterele 'a', ..., 'f' citite de la tastatura in  
valorile 10, ..., 15.  
3. Sa se scrie un program care sa transforme caracterele 'A', ..., 'F' citite de la tastatura in  
valorile 10, ..., 15.  
4. Sa se scrie un program care sa citeasca de la tastatura valorile 10, ..., 15 (doua cifre  
succesive), sa le transforme in caracterele 'A', ..., 'F' si sa le afiseze pe ecran.