Lucrarea de laborator numarul 7

Înmulţirea şi împărţirea numerelor binare

A.Înmulțirea a Doua Numere Binare de 8 Biți Fără Semn

Algoritmul dupa care se înmuțesc doua numere binare a fost prezentat în laboratorele precedente. Pe scurt algoritmul este urmatorul:

- se aliniază numerele unul sub altul de la dreapta la stânga, începând cu cel mai puțin semnificativ bit;
- pentru fiecare bit 1 de la înmulțitor se scrie sub acesta primul operand începând din dreptul locației respective de la dreapta spre stânga;
- se efectuează operația de adunare prezentată mai sus asupra operanzilor astfel rezultați.

Exemple:

```
        1 1 0 1  X      13  X
        0 1 0 1          5
        -------         ----
        1 1 0 1         65
    1 1 0 1
  -------------
1 0 0 0 0 0 1
```

```
        0 1 0 1  X       5  x
        1 1 1 1         15
        -------         ----
        0 1 0 1         75
      0 1 0 1
    0 1 0 1
  0 1 0 1
  -------------
1 0 0 1 0 1 1
```

Datorită limitorilor impuse de modelul simulatorului se pot face inmulțiri cu numere de maxim 4 biți.

În pseudocod algoritmul va arăta astfel:

```
begin
    *încarcă în registrul R1 înmulțitul
    *încarcă în registrul R2 înmulțitorul
    *încarcă în registrul R3 masca- 0x01
    *încarcă în registrul R4 pasul- 0x01
    for i=1 to 4
        R0=R2 and R3
        *dacă bitul înmulțitului de pe poziția i este 1 atunci
```

```
                    R5=R5+R1
                *deplasează la stânga cu o poziție îmulțitorului
            end for
        end
```

La implementarea acestui algoritm s-a folosi o "mască mobilă". Acesta este o mască pe un bit care este deplasat cu căte o poziție la stânga lşa fiecare ciclu. În acest fel putem analiza fiecare bit în parte.

După operația de mascare rezultatul şi-ului pe biți este pus în registrul R0. Dacă registrul R0 are aceeaşi valoare ca şi registrul R3 atunci înseamnă că pe poziția pe care se află 1-ul din mască se afla un 1 şi în numărul mascat.

În continuare se va explica algoritmul pas cu pas.

Pasul 1-Se încarcă în registrii termenii înmulțirii şi maştile

R0:	0000.0000		
R1:	0000.0010	-înmulțit	0x02
R2:	0000.1101	-înmulțitor	0x0D
R3:	0000.0001	-masca mobilă	0x01
R4:	0000.0001	-pasul	0x01
R5:	0000.0000		
RE:	0000.0000		

Pasul 2-Se maschează toți biții mai puțin bitul cel mai puțin semnificativ din înmulțitor

R0:	*0000.0001*		
R1:	0000.0010	-înmulțit	0x02
R2:	0000.1101	-înmulțitor	0x0D
R3:	0000.0001	-masca mobilă	0x01
R4:	0000.0001	-pasul	0x01
R5:	0000.0000		
RE:	0000.0000		

Pasul 3-Se compară rezultatul registrului R0 cu cel al registrului R3 (masca). Dacă cele două registre au acelaşi conținut se adună la conținutul registrului R5 conținutul registrului R1 (înmulțitul).

R0:	0000.0001		
R1:	0000.0010	-înmulțit	0x02
R2:	0000.1101	-înmulțitor	0x0D
R3:	0000.0001	-masca mobilă	0x01
R4:	0000.0001	-pasul	0x01

Lucrarea de laborator numarul 7

R5: *0000.0010*
RE: 0000.0000

Pasul 4-Se deplasează la stânga masca mobilă cu o poziție(se roteşte la drepta cu 7 poziții)

R0:	0000.0001		
R1:	0000.0010	-înmulțit	0x02
R2:	0000.1101	-înmulțitor	0x0D
R3:	*0000.0010*	-masca mobilă	0x02
R4:	0000.0001	-pasul	0x01
R5:	0000.0010		
RE:	0000.0000		

Pasul 5-Se incrementează cu 1 registrul R5

R0:	0000.0001		0x04
R1:	0000.0010	-înmulțit	0x02
R2:	0000.1101	-înmulțitor	0x0D
R3:	0000.0010	-masca mobilă	0x01
R4:	0000.0001	-pasul	0x01
R5:	*0000.0001*		0x01
RE:	0000.0000		

Pasul 6-Se încarcă în registrul R0, 0x04(lungimea în biți a numerelor care trebuie adunate) şi se compară cu registrul R5.
Dacă cele două numere sunt egale se termină programul.

R0:	0000.0001		0x04
R1:	0000.0010	-înmulțit	0x02
R2:	0000.1101	-înmulțitor	0x0D
R3:	0000.0001	-masca mobilă	0x01
R4:	0000.0001	-pasul	0x01
R5:	*0000.0001*		0x01
RE:	0000.0000		

Pasul 7- Se reia ciclul de la pasul 2, dar cu noua masca din R3

R0:	*0000.0000*		0x00
R1:	0000.0100	-înmulțit	0x04
R2:	0000.1101	-înmulțitor	0x0D
R3:	0000.0010	-masca mobilă	0x02

R4:	0000.0001	-pasul	0x01
R5:	0000.0001		0x01
RE:	0000.0000		

Pasul 8-Se compară registrul R0 cu registrul R3 şi observânduâ-se ca sunt diferite se sare peste adunare şi se deplasează masca mobilă şi înmulțitorul cu o poziție la stânga.

R0:	*0000.0000*		0x00
R1:	0000.1000	-înmulțit	0x04
R2:	0000.1101	-înmulțitor	0x0D
R3:	0000.0100	-masca mobilă	0x04
R4:	0000.0001	-pasul	0x01
R5:	0000.0001		0x01
RE:	0000.0000		

Pasul 9 – Se incrementează registrul R5 şi se compară cu 0x04. Incrementarea registrului are loc la fiecare ciclu de deplasări, indiferent dacă are loc sau nu adunare.

R0:	*0000.0100*		0x04
R1:	0000.1000	-înmulțit	0x04
R2:	0000.1101	-înmulțitor	0x0D
R3:	0000.0100	-masca mobilă	0x04
R4:	0000.0001	-pasul	0x01
R5:	0000.0010		0x02
RE:	0000.0000		

Paşii descrişi anterior se vor repeta până când R5 va avea valoarea 4, adică se vor parcurge toți biții.

```
        load R1,[inmultit]
        load R2,[inmultitor]

        load R3,0x01 ;masca mobila
        load R4,0x01 ;pasul

nextbit: and R0,R2,R3          ;se lasa ne mascat numai un bit
        jmpEQ R3=R0,shiftAndAdd ;dacă rezultatul operației
                                ;de mascare şi masca sunt la
                                ;fel salt la shiftAndAdd
```

```
                jmp shift
shiftAndAdd:addi R5,R5,R1       ;se adună înmulțitul la rezultat
      shift: ror R1,7           ;se deplasează cu 1 poziție la
                                ;stanga inmulțitul
                ror R3,7        ;se deplasează cu 1 poziție la
                                ;stanga masca
                addi RC,RC,R4   ;se incrementează indexul
                                ;poziției bitului care ;controlează
                                adunarea
                load R0,4       ;pregăteste compararea
                                ;indexului ;bitului de control cu 4
                jmpEQ RC=R0,stop    ;daca bitul de control are
                                ;valoarea 4 atunci opreste
                                ;programul
                jmp nextbit     ;salt la începutul ciclului

stop:           halt

inmultit:       db 0x02
inmultitor:     db 0x0F
```

Lucrarea de laborator numarul 7

B.Împartirea a Doua Numere În Baza 2 având 8 Biti

Împărțirea se efectuează similar cu împărțirea zecimală. Numerele sunt aliniate la bitul cel mai semnificativ. Dacă numărul format din biții de deasupra împărțitorului este mai mare decât împărțitorul se scade din acesta împărțitorul şi se concatenează 1 la cat. Dacă nu, se concatenează 0. În ambele cazuri împărțitorul se deplasează cu o poziție la dreapta. Procesul continuă până când ce rămâne în locul împărțitului este mai mic decât împărțitorului.

Exemplu:

1	*0*	*1*	*1*	1	1	1	1	1	0	1	0			
1	*0*	*1*	*0*	↓				1				Mai mare	Scad, Deplasaez şi inserez 1	
	0	0	*1*	*1*									Rezultat	
1	0	1	0	↓				1	0			Mai mic	Deplasez şi inserez 0	
	0	1	1	1										
	1	0	1	0	↓			1	0	0		Mai mic	Deplasez şi inserez 0	
		1	*1*	*1*	*1*									
		1	*0*	*1*	*0*	↓		1	0	0	1	Mai mare	Scad, Deplasaez şi inserez 1	
			1	0	1	1								
			1	0	1	0						Mai mare	Scad, Deplasaez şi inserez 1	
						1						Restul rămâne în locul câtului		

Pentru a efectua împărțirea trebuie să aliniem cei doi termeni la bitul cel mai semnificativ. Pentru aceasta trebuie să determinăm unde se afla bitul cel mai semnificativ al numărului cel mai mare. În implementarea de față împărțitul.

După ce determinăm pozitia acestui bit trebuie să bitul cel mai semnificativ al împărțitul la acesta. Pentru aceast folosim o masca care are 1 pe poziția bitului cel mai semnificativ al împărțitului.

Deplasăm la stânga împărțitorul şi testăm cu ajutorul măştii dacă bitul cel mai semnificativ a ajuns pe poziția dorită. Dacă rezultatul mascării este egal cu masca înseamnă că în împărțitor avem 1 pe poziția bitului de 1 din împărțitor.

Odata alinaite cele două numere începem să comparăm împărțitorul cu numărul format de biții împărțitului aflați deasupra

împărțitorului. Aceasta comparare este realizată prin mascarea biților rendundanți din împărțit.

Dacă împărțitorul este mai mic decât numărul obținut din împărțit atunci concatenăm 1 la sfârşitul câtului. Concatenarea se realizează prin deplasarea câtului cu o poziție la stânga şi adunarea cu 0x01 a câtului.

Dacă este mai mare doar deplasăm câtul cu o poziție la stânga. În ambele cazuri împărțitul va fi deplasat cu o poziție la dreapta.

Acest algoritm se repetă până când câtul este mai mic decât împărțitorul.

Algoritm în pseudocod:

```
Begin
    Begin *identificarea primului celui mai semnificativ 1 din
          Împărțitor
      *Se încarcă în R0 împărțitorul
      *Se încarcă în R1 împărțitul
      *Dacă împărțitorul este mai mare decât împărțitul câtul devine
          zero iar restul devine împărțitul
      *Se încarcă în R2 0x80 masca variabilă
      *Se încarcă în RB 0x80, folosit la inserarea lui 1 la stânga
    registrului
    do
          *copiază în registrul R0, conținutul registrului R1
          *and pe biți între biții din R0 şi R2
          *deplasează registrul cu o poziție la dreapta
    while R2!=R0
    End
    Begin *alinierea împărțitorului la primul bit al împărțitului
          *copiază masca folosită la aliniere din R2 în RE
          *încarcă în R2 împărțitorul
          do
          *and pe biți între biții din RE şi R2, iar rezultatul se aşează
          în R0
          * se deplasează R2 cu o poziție la stânga
          * se incrementeză contorul de poziții la stânga
          * se inserează în 1 la sfârşitul registrului RD şi se
                deplasează la stânga cu o poziție
          while RE != R0
    End
          *Se încarcă în registrul RC 0xFF
          * se inversează conținutul registrului RD(xor pe biți)
          * se copiază în R3 coținutul registrului R2
```

```
            * se inversează conținutul registrului R3
            * se adună 1 la registrul R3 şi rezultatul se pune în registrul
            R3-se crează complementul față de 2
      Begin *Împărțire propiuzisă
            do
            *se copiază conținutul registrului R2 în R0
            *se lasa în registrul R0 numai biții de deasupra
            împărțitorului
            if R2<=R0 then *scade din R1 complementul față de 2
                  aparținând R2 (R3)
                  *inserează prin adunarea 1 la sfârsitul câtului
            end if
            *deplasează câtul cu o poziție la stânga
            *deplasează împărțitorul cu o poziție la dreapta
            *deplasează complementul împărțitorul cu o poziție la
            dreapta
            *deplasează masca cu o poziție la dreapta
            *inserează 1 la stănga maştii pentru împărțit
            *scade 1 din contorul de poziții
            while contorul de poziții=0
      End
End
```

Exemplu:

Pasul 1 - Se încarcă împărțitorul şi împărțitul în R0 şi R1. Dacă împărțitorul este mai mic sau egal decât împărțitul împărțirea continuă. Dacă nu registrul câtului (R6) ia valoarea 0 iar regsitrul restului (R1) ia valoarea împărțitului şi se opreste execuția programului.

R0:	*0000.0111*	0x07
R1:	*0101.0110*	0x54
R2:	0000.0000	
R3:	0000.0000	
R4:	0000.0000	
R5:	0000.0000	
R6:	0000.0000	
R7:	0000.0000	
R8:	0000.0000	
R9:	0000.0000	
RA:	0000.0000	
RB:	0000.0000	
RC:	0000.0000	

RD: 0000.0000
RE: 0000.0000

Pasul 2 – Se inițializează registrii cu valorile necesare operațiilor

R0: 0000.0000
R1: 0101.0110 0x56 - Împărțitul
R2: 1000.0000 0x08 -Mască mobilă
R3: 0000.0000
R4: 0000.0000
R5: 0000.0001 0x01 -Pasul Adunării
R6: 0000.0000
R7: 0000.0000
R8: 0000.0000
R9: 0000.0000
RA: 1111.1111 0xFF- -1d
RB: 1000.0000 0x80 – Număr cu 1 pe poziția cea mai semnificativă, folosit pentru inserarea de 1 la dreapta numărului
RC: 0000.0000
RD: 0000.0000
RE: 0000.0000

Pasul 3- Se maschează împărțitul cu masca mobilă. Rezultatul se pune în R0.

R0: 0000.0000 0x00 - Rezultatul mascării registrului R1
R1: 0101.0110 0x56 - Împărțitul
R2: 1000.0000 0x08 -Mască mobilă
R3: 0000.0000
R4: 0000.0000
R5: 0000.0001 0x01 -Pasul Adunării
R6: 0000.0000
R7: 0000.0000
R8: 0000.0000
R9: 0000.0000
RA: 1111.1111 0xFF- -1
RB: 1000.0000 0x80 – Număr cu 1 pe poziția cea mai semnificativă, folosit pentru inserarea de 1 la dreapta numărului
RC: 0000.0000
RD: 0000.0000
RE: 0000.0000

Pasul 4- Dacă R0 este diferit de R2 atunci se deplasează R2 cu o poziție la dreapta şi se repetă pasul 3.

R0: 0100.0000 0x40 - Rezultatul mascării registrului R1
R1: 0101.0110 0x56 - Împărțitul
R2: 0100.0000 0x08 -Mască mobilă
R3: 0000.0000
R4: 0000.0000
R5: 0000.0001 0x01 -Pasul Adunării
R6: 0000.0000
R7: 0000.0000
R8: 0000.0000
R9: 0000.0000
RA: 1111.1111 0xFF- -1
RB: 1000.0000 0x80 – Număr cu 1 pe poziția cea mai semnificativă, folosit pentru inserarea de 1 la dreapta numărului
RC: 0000.0000
RD: 0000.0000
RE: 0000.0000

Pasul 5 – Se copiază masca folosită în aliniere în registrul RE. Se încarcă în registrul R2 împărțitorul. Se începe alinierea împărțitorul la cel mai semnificativ bit din împărțit.

R0: 0000.0000 0x00 - Rezultatul mascării registrului R1
R1: 0101.0110 0x56 - Împărțitul
R2: 0000.0111 0x07 -Mască mobilă
R3: 0000.0000
R4: 0000.0000
R5: 0000.0001 0x01 -Pasul Adunării
R6: 0000.0000
R7: 0000.0000
R8: 0000.0000
R9: 0000.0000
RA: 1111.1111 0xFF- -1
RB: 1000.0000 0x80 – Număr cu 1 pe poziția cea mai semnificativă, folosit pentru inserarea de 1 la dreapta numărului
RC: 0000.0000
RD: 0000.0000
RE: 0100.0000 0x40 – masca cu 1 pe poziția bitului cel mai semnificativ din împărțit

Pasul 6- Se maschează împărţitorul cu masca din registrul RE, iar rezultatul se aşează în registrul R0. Dacă R0 este diferit de RE atuncea se deplasează la stânga cu o poziţie, se incrementează regsitrul R7,folosit ca şi contor de poziţii, se inserează la dreapta registrului RD şi se repetă pasul 5. Registrul RD formează inversul maştii care va fi folosită la separarea biţilor.

R0: 0000.0000 0x00 - Rezultatul mascării registrului R1
R1: 0101.0110 0x56 - Împărţitul
R2: 0000.1110 0x0E - Împărţitorul
R3: 0000.0000
R4: 0000.0000
R5: 0000.0001 0x01 -Pasul Adunării
R6: 0000.0000
R7: 0000.0000
R8: 0000.0000
R9: 0000.0000
RA: 1111.1111 0xFF- -1
RB: 1000.0000 0x80 – Număr cu 1 pe poziţia cea mai semnificativă, folosit pentru inserarea de 1 la dreapta numărului
RC: 0000.0000
RD: 0000.0000
RE: 0100.0000 0x40 – Masca cu 1 pe poziţia bitului cel mai semnificativ din împărţit.

Pasul 7 - Dupa încheierea alinierii celor două numere la bitul cel mai semnificativ urmează obţinerea mastii care ne va permite găsirea biţilor de deasupra împărţitorului. Încărcăm în registrul RC 0xFF care ne v-a permite să inversăm conţinutul registrului RD. Vom folosi acelaşi registru pentru a afla compelementul faţă de 2 al registrului.

R0: 0000.0000 0x00 - Rezultatul mascării registrului R1
R1: 0101.0110 0x56 - Împărţitul
R2: 0111.0000 0x0E - Împărţitorul după aliniere
R3: 1001.0000 0x90 -Complementul faţă de 2 al împărţitorului
R4: 0000.0000
R5: 0000.0001 0x01 -Pasul Adunării
R6: 0000.0000
R7: 0000.0100 0x04 -Numărul de poziţii la stânga
R8: 0000.0000
R9: 0000.0000
RA: 1111.1111 0xFF- -1
RB: 1000.0000 0x80 – Număr cu 1 pe poziţia cea mai semnificativă,

folosit pentru inserarea de 1 la dreapta numărului
RC: 0000.0000
RD: 0000.1111 0x0F- Masca folosită la comparare şi scădere
RE: 0100.0000 0x40 – Masca cu 1 pe poziția bitului cel mai semnificativ din împărțit.

Pasul 8 – Se evidențiază cu ajutorul măştii din RD numai biții din împărțitor care sunt deasupra împărțitorului. Rezultatul se aşează în R0. Dacă R0 este mai mic sau egal decât împărțitul se adună la acesta complementul față de 2 al împărțitului din registrul R3 şi se inserează 1 sfârşitul câtului. Se va deplasa câtul cu o poziție la dreapta chiar în cazul în care nu se îndeplineste condiția precedentă.

R0: 0101.0000 0x00 - Rezultatul mascării registrului R1
R1: 0101.0110 0x56 - Împărțitul
R2: 0111.0000 0x0E - Împărțitorul după aliniere
R3: 1001.0000 0x90 - Complementul față de 2 al împărțitorului
R4: 0000.0000
R5: 0000.0001 0x01 -Pasul Adunării
R6: 0000.0000
R7: 0000.0100 0x04 -Numărul de poziții la stânga
R8: 0000.0000
R9: 0000.0000
RA: 1111.1111 0xFF- -1
RB: 1000.0000 0x80 – Număr cu 1 pe poziția cea mai semnificativă, folosit pentru inserarea de 1 la dreapta numărului
RC: 0000.0000
RD: 1111.0000 0xF0 - Mască folosită la pentru compararea împărțitorului cu biții din împărțit care sunt deasupra lui.
RE: 0100.0000 0x40 – Masca cu 1 pe poziția bitului cel mai semnificativ din împărțit.

Pasul 9 – După efectuarea adunării cu complementul față de 2 se deplasează masca din RD cu o poziție la dreapta şi se inserează 1 pe poziția cea mai din stânga. Se mai deplasează cu o poziție la stânga şi împărțitorul şi complementul față de 2 al împărțitorului.De asemenea se decrementează R7.

R0: 0101.0000 0x00 - Rezultatul mascării registrului R1
R1: 0101.0110 0x56 - Împărțitul
R2: 0011.1000 0x38 - Împărțitorul după aliniere
R3: 1100.1000 0xC8 - Complementul față de 2 al împărțitorului

Lucrarea de laborator numarul 7

R4: 0000.0000
R5: 0000.0001 0x01 -Pasul Adunării
R6: 0000.0000
R7: 0000.0011 0x04 -Numărul de poziţii la stânga
R8: 0000.0000
R9: 0000.0000
RA: 1111.1111 0xFF- -1
RB: 1000.0000 0x80 – Număr cu 1 pe poziţia cea mai semnificativă, folosit pentru inserarea de 1 la dreapta numărului
RC: 0000.0000
RD: 1111.1000 0xF8 - Mască folosită la pentru compararea împărţitorului cu biţii din împărţit care sunt deasupra lui.
RE: 0100.0000 0x40 – Masca cu 1 pe poziţia bitului cel mai semnificativ din împărţit.

Se efectuează paşii 8 şi 9 până când ce rămâne în locul împărţitului este mai mic decât împărţitorul sau R7 este zero.

```
            load R1,[Impartit] ;numar de prelucrat
            load R2,0x80;masca variabila
            load RB,0x80
            load R5,0x1; pas 1
            load RA,-1d
moveAndMask:and R0,R1,R2
            jmpEQ R2=R0,aliniere1
            ror R2,1
            jmp moveAndMask
                        ;incepe aliniaerea impartitorului-
aliniere1:  move RE,R2      ;in acest moment masca pentru
                            ;aliniere este in R0 si RE
            load R2,[Impartitor]
test:       move R0,R2
            and R0,R0,RE ;mascheaza bitii care nu sunt necesari
            jmpEQ RE=R0, aliniat2
            ror R2,7
            addi R7,R7,R5;incrementeaza contorul de pozitii la
                        ;stanga
            ror RD,7
            addi RD,RD,R5 ;creaza inversul mastii care permite
                        ;compararea impartitului cu impartitorul
            jmp test
```

```
aliniat2:   load RC, 0xFF
            xor RD,RD,RC
            move R3,R2
            xor R3,R3,RC; complementil fata de 1 al impartitorului
            addi R3,R3,R5;complementul fata de 2 al impartitorului
nextcicle:  move R0,R1
            and R0,R0,RD
            jmpLE R2<=R0, addOne
            jmp addZero
addOne:     addi R1,R1,R3;scade din impartit impartitorul
            addi R6,R6,R5;insereaza 1 la sfarsitul catului
addZero:    ror R6,7;deplaseaza catul cu o pozitie la stanga
            ror R2,1 ;deplaseaza impartitorul ci o pozitie la dreapta
            ror R3,1 ;deplaseaza complementul impartitorului cu o
                     ;pozitie la dreapta
            addi R3,R3,RB;insereaza 1 la stanga complementului
                        ;impartitorului
            ror RD,1 ;deplaseaza masca cu o pozitie la dreapta
            addi RD,RD,RB ;insereaza 1 la stanga in masca
            addi R7,R7,RA;scade 1 din contorul de pozitii
            load R0,0x00
            jmpEq R7=R0,stop
            jmp nextcicle
stop:       halt
Impartit:   db 54h
Impartitor: db 7h
```

Lucrarea de laborator numarul 7

Exerciții 1:

1. Se dau două numere pe 8 biti. Dacă cele două numere au prima jumătate egala să se înmulțească jumătățile cele mai putin semnificative.
Caz particular: FA si EF

2. Se da un numar pe 8 biti . Sa se inmultеasca jumatatea cea mai semnificativa cu jumatatea cea mai putin semnificativa.
Caz particular: CD

3. Se dau două numere pe 8 biți. Dacă primul număr este mai mic decât al doilea să se împartă al doilea număr la primul. Dacă nu invers (primul număr la al doilea).
Caz particular: F9 si D4

4. Se da un numar binar pe 8 biti. Sa se converteasca acest numar in format BCD.Indicatie: numarul maxim care poate fi reprezenta pe 8 biti este 255. Deci numarul va fi obtinut din resturile a trei impartiri repetate la 10.
Caz particular: D9

5. Să se scrie un program care să permită înmulțirea a două numere pe 8 biți.
Caz particular: AB si CD

Exerciții 2:

1. Se dau două numere pe 8 biti. Dacă cele două numere au prima jumătate egala să se înmulțească jumătățile cele mai putin semnificative.
Caz particular: D9 si 8C

2. Se da un numar pe 8 biti . Sa se inmultеasca jumatatea cea mai semnificativa cu jumatatea cea mai putin semnificativa.
Caz particular: A0

3. Se dau două numere pe 8 biți. Dacă primul număr este mai mic decât al doilea să se împartă al doilea număr la primul. Dacă nu invers (primul număr la al doilea).
Caz particular: BC si 89

4. Se da un numar binar pe 8 biti. Sa se converteasca acest numar in format BCD.Indicatie: numarul maxim care poate fi reprezenta pe 8

biti este 255. Deci numarul va fi obtinut din resturile a trei impartiri repetate la 10.

Caz particular: A1

5. Să se scrie un program care să permită înmulțirea a două numere pe 8 biți.

Caz particular: 56h si 89h

Exerciții 3:

1. Se dau două numere pe 8 biti. Dacă cele două numere au prima jumătate egala să se înmulțească jumătățile cele mai putin semnificative.

Caz particular: A7 si A9

2. Se da un numar pe 8 biti . Sa se inmulteasca jumatatea cea mai semnificativa cu jumatatea cea mai putin semnificativa.

Caz particular: C6

3. Se dau două numere pe 8 biți. Dacă primul număr este mai mic decât al doilea să se împartă al doilea număr la primul. Dacă nu invers (primul număr la al doilea).

Caz particular: D7 si 8C

4. Se da un numar binar pe 8 biti. Sa se converteasca acest numar in format BCD.Indicatie: numarul maxim care poate fi reprezenta pe 8 biti este 255. Deci numarul va fi obtinut din resturile a trei impartiri repetate la 10.

Caz particular: F8

5. Să se scrie un program care să permită înmulțirea a două numere pe 8 biți.

Caz particular: A7 si 09

Exerciții 4:

1. Se dau două numere pe 8 biti. Dacă cele două numere au prima jumătate egala să se înmulțească jumătățile cele mai putin semnificative.

Caz particular: D8 si 67

2. Se da un numar pe 8 biti . Sa se inmulteasca jumatatea cea mai semnificativa cu jumatatea cea mai putin semnificativa.

Caz particular: 76

Lucrarea de laborator numarul 7

3. Se dau două numere pe 8 biți. Dacă primul număr este mai mic decât al doilea să se împartă al doilea număr la primul. Dacă nu invers (primul număr la al doilea).

Caz particular: 89 si F5

4. Se da un numar binar pe 8 biti. Sa se converteasca acest numar in format BCD.Indicatie: numarul maxim care poate fi reprezenta pe 8 biti este 255. Deci numarul va fi obtinut din resturile a trei impartiri repetate la 10.

Caz particular: D3

5. Să se scrie un program care să permită înmulțirea a două numere pe 8 biți.

Caz particular: F4 si D8

Exerciții 5:

1. Se dau două numere pe 8 biti. Dacă cele două numere au prima jumătate egala să se înmulțească jumătățile cele mai putin semnificative.

Caz particular: D4 si D7

2. Se da un numar pe 8 biti . Sa se inmulteasca jumatatea cea mai semnificativa cu jumatatea cea mai putin semnificativa.

Caz particular: AE

3. Se dau două numere pe 8 biți. Dacă primul număr este mai mic decât al doilea să se împartă al doilea număr la primul. Dacă nu invers (primul număr la al doilea).

Caz particular: F6 si D8

4. Se da un numar binar pe 8 biti. Sa se converteasca acest numar in format BCD.Indicatie: numarul maxim care poate fi reprezenta pe 8 biti este 255. Deci numarul va fi obtinut din resturile a trei impartiri repetate la 10.

Caz particular: F5

5. Să se scrie un program care să permită înmulțirea a două numere pe 8 biți.

Caz particular: FF si FE