Lucrarea de laborator numarul 6

A.Continuarea descrierii setului de instructiuni pentru simulatorul calculatorului didactic

origin-*originea*-seteaza adresa de la care sa se inceapa scrierea codului. Are mnemonicul *org*, op-codul *7* si formatul *org adr* unde *adr* este un numar. Aceasta instructiune nu va apare in memoria calculatorului. Ea este folosita de intrepretorul de limbaj de asamblare pentru a faciliata organizarea programului in memorie.
Exemplu:
org 60h - seteza originea scrierii memorie la adresa 60h.
load R0 ,20- scrie instructiunea load in memorie la adresa 60h.

direct store- *stocare directa* – mnemonic *store,* op-codul *3* si formatul *strore reg, [addr].* Scrie in memorie la adresa *addr* continutul registrului *reg*.
Exemplu:
store R1,[0xA0]
store R2,[eticheta]

In memoria simulatorului instructiunea va fi reprezentata sub forma hexadecimala si va aparea astfel:

31 0A store(3) R1(1), 0x0A(10)

Sau sub forma binara
0011 0001 0000 1010

indirect store – *stocare indirecta* – mnemonicul *store,* op-codul *E* si formatul *strore reg1, [reg2]*. Scrie in memorie la adresa continuta in registrul *reg2.*
Exemplu:
store R1,[R2]

In memoria simulatorului instructiunea va fi reprezentata sub forma hexadecimala si va aparea astfel:

E0 12 store(E) R0(0), R1(1), R2(2)

Sau sub forma binara
1110 0000 0001 0010

floating point addition*-adunare in virgula mobila-* mnemonicul *addf,* op-codul *6* si formatul *addf reg3,reg1, reg2*. Aduna doua numere reprezentata in virgula mobila alfate in registrii *R1* si *R2* si scrie rezultatul in registrul *R3*.
Exemplu:
addf R7,R1,R2

In memoria simulatorului instructiunea va fi reprezentata sub forma hexadecimala si va aparea astfel:

67 12 addf(6) R7(7),R1(1), R2(2)

Sau sub forma binara
1010 0010 0010 1010

Numere reprezentate in virgula mobila au formatul 1+3+4. Adica primul bit reprezinta semnul 1 pentru numere negative, 0 pentru numere pozitive. Urmatorii trei biti reprezinta exponentul, iar ultimii patru reprezinta mantisa. Pentru reprezentarea numerelor negative se foloseste codul direct. In acest caz semnul unui numar va fi dat de bitul de semn.

Exemplu:

Numarul 10 va fi reprezentat in virgula mobila astfel.

0.100.1010

Numarul -10 va fi reprezentat in virgula mobila astfel.

1.100.1010

bitwise or *– sau logic pe biti-* are mnemonicul *or,* op-codul *7* si formatul *or reg3, reg1, reg2*. Efectueaza un or logic pe biti intre registrii *reg1* si *reg 2*, iar rezultatul se scrie in registrul *reg 3*.
Exemplu:
or R7,R1,R2

In memoria simulatorului instructiunea va fi reprezentata sub forma hexadecimala si va aparea astfel:

77 12 or(7) R7(7), R1(1), R2(2)

Sau sub forma binara
0111 0111 0001 0010

bitwise and *– si logic pe biti-* are mnemonicul and*,* op-codul *8* si formatul *and reg3, reg1, reg2*. Efectueaza un or logic pe biti intre registrii *reg1* si *reg 2*, iar rezultatul se scrie in registrul *reg 3*.
Exemplu:
and R7,R1,R2

In memoria simulatorului instructiunea va fi reprezentata sub forma hexadecimala si va aparea astfel:

87 12 and(8) R7(7) R1(1)R2(2)

Sau sub forma binara

1000 0111 0001 0010

bitwise exclusive or *– sau exclusiv logic pe biti-* are mnemonicul x*or,* op-codul *9* si formatul x*or reg3, reg1, reg2*. Efectueaza un sau exclusiv logic pe biti intre registrii *reg1* si *reg 2*, iar rezultatul se scrie in registrul *reg 3*.
Exemplu:
xor R7,R1,R2

In memoria simulatorului instructiunea va fi reprezentata sub forma hexadecimala si va aparea astfel:
97 12 xor(9) R1(2), R2(2)

Sau sub forma binara

1001 0111 0001 0010

rotate right *-rotire dreapra* – are mnemonicul *ror,* op-codul *A* si formatul *ror reg , num*. Efectuaza o operatie de rotire asupra registrului *reg*. Aceasta insemna ca bitii de la sfarsitul registrului sunt copiati la inceputul registrului iar acesta este mutat la dreapta cu un bit.
Exemplu:

ror R2, 10

In memoria simulatorului instructiunea va fi reprezentata sub forma hexadecimala si va aparea astfel:
A2 0A ror(A) R2(1), 0A(10)

Sau sub forma binara

1010 0010 0010 1010

jump when less or equal- salt daca mai mic sau egal – are mnemonicul *jmpLE,* op-codul *F* si formatul *jmpLE reg<=R0, addr* sau *jmpLE reg<=R0, [eticheta].*
Exemplu:

jmpLE R3<=R0 45h
jmpLE R3<=R0 sari_aici

In memoria simulatorului instructiunea va fi reprezentata sub forma hexadecimala si va aparea astfel:

F3 0A ror(A) R2(1), 0A(10)

Sau sub forma binara
1010 0010 0010 1010

B.Tehnici de programare

B1.Scrierea în limbaj de asamablare a unei instrucţiuni decizionale de tipul "if then else"

Singurul registru la care se poate compara este R0. Acest lucru va influența decisiv implementatrea algoritmilor.

```
        jmpEQ  R1=R0, then          ;if
else:   addi R2,R2,R3               ;else
        jmp endif
then:   addi R3,R2,R3               ;then
endif: halt
```

B2.Scrierea în limbaj de asamblare a unei instrucțiuni de tipul "*while do*"

```
do:     addi R5,R5,R6
        addi R4,R4,R6
        jmpEQ R1=R0  while
        jmp do
while:halt
```

B3.Scrierea în limbaj de asamblare a unei instrucțiuni de tipul "*for*"

```
for(i=0;i<10;i++){
        a=a+10
}
```

```
        load R0, 0xA            ;i<10
        load R1, 0x00           ;i=0
        load R2, 0x01           ;i++
        load R6, 0xA
for:    addi R5,R5,R6           ;a=a+10
        addi R1,R1,R2
        jmpLE R1<=R0,for
```

B4.Determinarea complementului fata de 1 al unui numar.

Aceasta operatie se realizeaza prin efectuarea unei operatii de sau exclusiv logic pe biti intre continutul registrului si o masca 11111111. Exprimandu-ne foarte simplist operatia xor pe un bit are ca rezultat 1daca cei doi operanzi sunt diferiti (unul 0 si celalalt 1) si 0 daca cei doi operanzi sunt identici (0 si 0 sau 1 si 1). In practica se poate obtine un inversor folosind o poarta *xor* care are pusa o intrare la 1 logic.

Exemplu:

1011 1010 (+)
1111 1111
0100 0101

Se observa ca rezultatul este complemntul fata de 1 al numarului 0xBA.

Mai jos se prezinta un program care obtine complementul fata de 1 al unui numar.

```
00:   load R1, 0xBA
02:   load R2, 0xFF
04:   xor R1,R1,R2
06:   halt
```

Daca dorim sa obtinem complementul fata de 2 de doi al unui numar la complementul fata de 1 al numarului vom adauga 1.Deci programul va arata astfel.

```
00:   load R1, 0xBA
02:   load R2, 0xFF
04:   xor R1,R1,R2
06:   load R3, 00000001b
08:   addi R1,R1,R3
0A:   halt
```

B5.Mascarea unui şir de biţi

Principiul de bază al acestei operaţii este efectuarea unui *and* logic pe biti între un numar binar, masca şi un numărul care trebuie sa fie mascat. Setarea la zero a unor biţi dintr-un numar se face prin efectuarea unui and pe biti între numărul care trebuie sa fie mascat şi un număr binar în care pe

poziția corespunzătoare biților care trebuie să îşi păstreze valoarea sunt biți cu valoarea 1, iar pe poziția care trebuie sa apară zero, zero.

Exemplu:

Avem numarull 0x0AD şi dorim să vedem dacă este un număr par sau impar. Pentru aceasta trebuie analizat ultimul bit. Dacă acesta este 0 atunci numărul este par, dacă nu numarul este impar. Pentru a putea compara ultimul bit trebuie să mascăm ceilați biți. Aceasta se realizează prin efectuarea unui and pe biti intre numărul 0x0AD şi 0x01.

0x0AC = 1010 1100
0x0AD = 1010 1101
0x01 = 0000 0001

1010 1100 $\wedge$ 0000 0001 = 0000 0000
1010 1101 $\wedge$ 0000 0001 = 0000 0001

O altă aplicație este verificarea existenței transportului în cazul adunării a două numere. Se consideră că se adună două numere de o cifră hexadecimală, 0x0A şi 0x0B.

0x0A+0x0B=0x15

B6.Scrierea unui operații de deplasare (shift) la dreapta

Acest tip de instructiune se foloseste in general pentru impartiri. O deplasare la dreapata cu x pozitii eschivaland cu impartirea la 2^x. Operatia este compusa din doua operatii, una de rotire la dreapta si una de mascare a bitilor in plus.

Presupunem ca avem un registru de 8 biti care contine numarul *0xBA*. Reprezentat in binar va arata astfel:

10111*010*

Dorim sa delplasam la dreapta acest numar cu 3 pozitii (impartim la 8). Prima data rotim registrul cu valoarea 3. Rezulatatul va arata astfel:

*010*10111

Observam ca ultimii cinci biti contin rezultatul care ne intereseaza, dar primii trei biti nu ar trebui sa fie acolo. Acestia vor fi mascati cu zero. Pentru cazul general se va folosi urmatoarea masca 0..x..01..n-x..1, unde n este lungimea registrului iar x numarul de poziții care trebuie deplasate. Dupa

aceasta vom efectua un şi pe biti intre continutul registrului şi 00011111. În urma mascării rezultatul va fi 00010111, ceace ne interesează pe noi.

Mai jos se dă un program scris în limbaj de asamblare care implementează algoritmul de mai sus.

```
00:load R1, 0x0BA
02:ror R1,3
04:load R2,00011111
06:and R1,R1,R2
08:halt
```

B7.Scrierea unui operaţii de deplasare(shift) la stânga

Această operaţie se implementează într-o maniera asemănătoare cu prima, adică printr-o rotire la dreapta si o mascare .

Se consideră acelaşi registru cu aceleaşi conţinut. Trebuie să deplasăm acest registrul la stânga cu x poziţii. Acest lucru echivalează cu înmulţirea cu 2^x . Pentru acesta va trebui sa rotim registrul cu 8-x pozitii, daca registrul este de un octet. Daca nu vom scădea numarul de poziţii de deplasat în registru din numarul de biti din lungimea registrului.

După aceasta conţinutul registrului va fi mascat cu o masca de forma 1..n-x..10..x..0.

Registrul va arata in decursul procesarii astfel.

Pasul 1. 101*1 1010* – continut initial
Pasul 2. *1101 0*101 - dupa rotire
Pasul 3. *1101 0*000 - dupa mascare

Programul în asamblare arată astfel:

```
00:load R1, 0x0BA
02:ror R1,5
04:load R2,11111000
06:and R1,R1,R2
08:halt
```

B8.Adunarea a doua numere intregi cu detectarea transportului

În cazul în care trebuie să adunăm doua numere al căror rezultat este pe mai mult de 8 biti simulatorul folosit de noi va trunchia rezultatul la lungimea unui registru, lucru care poate fi daunător. În continuare se va prezenta o metodă de calcul care compesează aceasta deficiență.

Considerăm că în memorie avem două numere întregi care trebuie adunate, 1100 1111(0x0CF) şi 1010 0101 (0x0A5).

Pasul 1-Se copiază primul număr în registrul R1 şi al doilea număr în registrul R2.

R1	1100 1111	CFh
R2	1010 0101	A5h
R3		
R4		
R5		
R6		

Pasul 2-Se setează valoarea registrului Rx la 0x0F şi Ry la 0x0F0.

R1	1100 1111	CFh
R2	1010 0101	A5h
R3		
R4		
R5	0000 1111	0Fh
R6	1111 0000	F0h

Pasul 3-Se maschează prima jumătate a celor doi regiştri.-Se face un şi pe biti între cei doi regiştrii şi Rx.

R1	0000 1111	0Fh
R2	0000 0101	05h
R3		
R4		
R5	0000 1111	0Fh
R6	1111 0000	F0h

Pasul 4-Se adună R1 şi R2 iar rezultatul se pune în R2.

R1	0000 1111	0Fh
R2	0001 0100	14h
R3		
R4		
R5	0000 1111	0Fh
R6	1111 0000	F0h

Pasul 5-Se copiază conținutul registrului R2 în R1.

R1	0001 0100	14h
R2	0001 0100	14h
R3		
R4		
R5	0000 1111	0Fh
R6	1111 0000	F0h

Pasul 6-Se deplasează la dreapta R1 cu 4 poziții.

R1	0000 0001	01h
R2	0001 0010	14h
R3		
R4		
R5	0000 1111	0Fh
R6	1111 0000	F0h

Pasul 7-Se maschează prima jumătate a registrului R2.

R1	0000 0001	01h
R2	0000 0100	04h
R3		
R4		
R5	0000 1111	0Fh
R6	1111 0000	F0h

Pasul 8-Se copiază primul număr în registrul R3 şi al doilea în R4.

R1	0000 0001	01h
R2	0000 0100	04h
R3	1100 1111	CFh
R4	1010 0101	A5h

R5	0000 1111	0Fh
R6	1111 0000	F0h

Pasul 9-Se maschează a doua jumătate a registrului R3 şi a registrului R4. Se face un şi pe biti între cei doi registri şi registrul Ry.

R1	0000 0001	01h
R2	0000 0100	04h
R3	1100 0000	C0h
R4	1010 0000	A0h
R5	0000 1111	0Fh
R6	1111 0000	F0h

Pasul10-Se deplasează cei doi registri cu patru poziții la dreapta.

R1	0000 0001	01h
R2	0000 0100	04h
R3	0000 1100	0Ch
R4	0000 1010	0Ah
R5	0000 1111	0Fh
R6	1111 0000	F0h

Pasul11-Se adună R3 cu R1, iar rezultatul se aşează în R3.

R1	0000 0001	01h
R2	0000 0100	04h
R3	0000 1101	0Dh
R4	0000 1010	0Ah
R5	0000 1111	0Fh
R6	1111 0000	F0h

Pasul12-Se adună R4 cu R3, iar rezultatul se aşează în R4.

R1	0000 0001	01h
R2	0000 0100	04h
R3	0000 1101	0Dh
R4	0001 0111	17h
R5	0000 1111	0Fh
R6	1111 0000	F0h

Pasul13-Se copiază conținutul registrului R4 în R3.

R1	0000 0001	01h
R2	0000 0100	04h
R3	0001 0111	17h
R4	0001 0111	17h
R5	0000 1111	0Fh
R6	1111 0000	F0h

Pasul14-Se deplasează conținutul registrului R3 cu patru poziții la dreapta.În registrul R3 vom avea transportul final.

R1	0000 0001	01h
R2	0000 0100	04h
R3	0000 0001	01h
R4	0001 0111	17h
R5	0000 1111	0Fh
R6	1111 0000	F0h

Pasul15-Se deplasează conținutul registrului R4 cu patru poziții la stânga.

R1	0000 0001	01h
R2	0000 0100	04h
R3	0000 0001	01h
R4	0111 0000	70h
R5	0000 1111	0Fh
R6	1111 0000	F0h

Pasul16- Se adună R4 cu R2 iar rezultatul se aşează în R4.

R1	0000 0001	01h
R2	0000 0100	04h
R3	0000 0001	01h
R4	0111 0100	74h
R5	0000 1111	0Fh
R6	1111 0000	F0h

Programul în limbaj de asamblare va arăta astfel:

```
load r1,[termen1]
load r2,[termen2]
load r5,0x0F
load r6,0xF0
and r1,r1,r5
and r2,r2,r5
addi r2,r2,r1
move r1,r2
ror r1,4
and r1,r1,r5
and r2,r2,r5
load r3,[termen1]
load r4,[termen2]
ror r3,4
and r3,r3,r5
ror r4,4
and r4,r4,r5
addi r3,r3,r1
addi r4,r4,r3
move r3,r4
and r3,r3,r6
ror r3,4
and r4,r4,r5
ror r4,4
addi r4,r4,r2
halt
termen1:db 0xCF
termen2:db 0xA5
```

Exerciţii 1.

1. Se dau două numere pozitive a şi b. Dacă primul este mai mare decât al doilea să se adune, iar rezultatul să se plaseze în memorie la adresa F0h. Dacă nu să se calculeze diferența celor două numere, iar rezultatul să se plaseze la aceeaşi adresă.
   Caz particular a= F312h, b=5678h.
2. Se dau două numere a şi b. Dacă primii trei biți ai acestor numre sunt identici cele două numere să se deplaseze la stânga cu trei biti,iar la dreapta numărului să se introducă 1 şi să se scadă numărul mai mare din numărul mai mic.
   Caz particular a= F3h, b=56h
3. Se dau trei numere a,b,c pe 8 biti, să se Determine dacă cele trei numere formează laturile unui triunghi(a+b>c,a+c>b,b+c>a). Dacă nu să se afişeze un mesaj de eroare.
   Caz particular a=9,b=4,c=16.
4. Se dă un numar pe 8 biti. Dacă prima jumătate este mai mică decât a doua jumătate cele două jumătăți să îşi schimbe locurile. Să se scadă prima din prima jumătate a doua jumătate, iar rezultatul să se afişeze în format decimal(în fereastra de afişare să apară echivalentul decimal al numărului).Caz particular AB
5. Se dau două numere a,b. Dacă cele două numere au acelaşi număr de biți să se adune printr-o metodă care permite detectarea transportului. Dacă nu să scadă din numărul mai mare numărul mai mic. Caz Particular A9h şi 16h

Exerciţii 2.

1. Se dau două numere pozitive a şi b. Dacă primul este mai mare decât al doilea să se adune, iar rezultatul să se plaseze în memorie la adresa F0h. Dacă nu să se calculeze diferența celor două numere, iar rezultatul să se plaseze la aceeaşi adresă.
   Caz particular a= D935 h, b=F911h.
2. Se dau două numere a şi b. Dacă primii trei biți ai acestor numre sunt identici cele două numere să se deplaseze la stânga cu trei biti,iar la dreapta numărului să se introducă 1 şi să se scadă numărul mai mare din numărul mai mic.
   Caz particular a= D9, b= F9h

3. Se dau trei numere a,b,c pe 8 biti, să se Determine dacă cele trei numere formează laturile unui triunghi(a+b>c,a+c>b,b+c>a). Dacă nu să se afişeze un mesaj de eroare.
   Caz particular a=7,b=12,c=10.
4. Se dă un numar pe 8 biti. Dacă prima jumătate este mai mică decât a doua jumătate cele două jumătăţi să îşi schimbe locurile. Să se scadă prima din prima jumătate a doua jumătate, iar rezultatul să se afişeze în format decimal(în fereastra de afişare să apară echivalentul decimal al numărului).Caz particular C9
5. Se dau dă un sir de caractere terminat cu caracterul 20. Să se adauge spaţii între cuvinte în astfel încât sirul sa aibă lungimea 80. Spaţiile se vor distribui cât mai echiatbil(se va simula justify din editoarele de text). Caz Particular "Ana are mere, pere şi gutui"

Exerciţii 3.

1. Se dau două numere pozitive a şi b. Dacă primul este mai mare decât al doilea să se adune, iar rezultatul să se plaseze în memorie la adresa F0h. Dacă nu să se calculeze diferenţa celor două numere, iar rezultatul să se plaseze la aceeaşi adresă. Caz particular a= BCDEh, b=0120h.
2. Se dau două numere a şi b. Dacă primii trei biţi ai acestor numre sunt identici cele două numere să se deplaseze la stânga cu trei biti,iar la dreapta numărului să se introducă 1 şi să se scadă numărul mai mare din numărul mai mic. Caz particular a= DEh, b=78h
3. Se dau trei numere a,b,c pe 8 biti, să se Determine dacă cele trei numere formează laturile unui triunghi(a+b>c,a+c>b,b+c>a). Dacă nu să se afişeze un mesaj de eroare.Caz particular a=12,b=20,c=15.
4. Se dă un numar pe 8 biti. Dacă prima jumătate este mai mică decât a doua jumătate cele două jumătăţi să îşi schimbe locurile. Să se scadă prima din prima jumătate a doua jumătate, iar rezultatul să se afişeze în format decimal(în fereastra de afişare să apară echivalentul decimal al numărului).Caz particular 90h
5. Se dau două numere a,b. Dacă cele două numere au acelaşi număr de biţi să se adune printr-o metodă care permite detectarea transportului. Dacă nu să scadă din numărul mai mare numărul mai mic. Caz Particular 81h şi A0h

Exerciţii 4.

1. Se dau două numere pozitive a şi b. Dacă primul este mai mare decât al doilea să se adune, iar rezultatul să se plaseze în memorie la adresa F0h.

Dacă nu să se calculeze diferența celor două numere, iar rezultatul să se plaseze la aceeaşi adresă. Caz particular a= F312h, b=5678h.

2. Se dau două numere a şi b. Dacă primii trei biți ai acestor numre sunt identici cele două numere să se deplaseze la stânga cu trei biti,iar la dreapta numărului să se introducă 1 şi să se scadă numărul mai mare din numărul mai mic. Caz particular a= F312h, b=5678h
3. Se dau trei numere a,b,c pe 8 biti, să se Determine dacă cele trei numere formează laturile unui triunghi(a+b>c,a+c>b,b+c>a). Dacă nu să se afişeze un mesaj de eroare. Caz particular a=9,b=4,c=12.
4. Se dă un numar pe 8 biti. Dacă prima jumătate este mai mică decât a doua jumătate cele două jumătăți să îşi schimbe locurile. Să se scadă prima din prima jumătate a doua jumătate, iar rezultatul să se afişeze în format decimal(în fereastra de afişare să apară echivalentul decimal al numărului).Caz particular 1Ah.
5. Să da un sir de octeți în memorie de tipul "a+b+c-d….", Unde a,b,c,… sunt numere reprezentate în formatul 1+4, numerele negative fiind reprezentate în cod complementar. Operanzii sunt reprezentați în memorie de caracterele "+" şi "-". Să se decodifice şi să se efectueze operațiile date. Rezultatul să se aşeze în memorie la începând cu adresa F0. Caz particular: "9+10+11+12"

Exerciții 5.

1. Se dau două numere pozitive a şi b. Dacă primul este mai mare decât al doilea să se adune, iar rezultatul să se plaseze în memorie la adresa F0h. Dacă nu să se calculeze diferența celor două numere, iar rezultatul să se plaseze la aceeaşi adresă. Caz particular a= F312h, b=5678h.
2. Se dau două numere a şi b. Dacă primii trei biți ai acestor numre sunt identici cele două numere să se deplaseze la stânga cu trei biti,iar la dreapta numărului să se introducă 1 şi să se scadă numărul mai mare din numărul mai mic. Caz particular a= F312h, b=5678h
3. Se dau trei numere a,b,c pe 8 biti, să se Determine dacă cele trei numere formează laturile unui triunghi(a+b>c,a+c>b,b+c>a). Dacă nu să se afişeze un mesaj de eroare.Caz particular a=100,b=5,c=9.
4. Se dă un numar pe 8 biti. Dacă prima jumătate este mai mică decât a doua jumătate cele două jumătăți să îşi schimbe locurile. Să se scadă prima din prima jumătate a doua jumătate, iar rezultatul să se afişeze în format decimal(în fereastra de afişare să apară echivalentul decimal al numărului).Caz particular FDh.
5. Să se scrie un program care să implementeze un program care să codifice decodifice şi să corecteze un număr pe 4 biti,folosind codul Hamming.