

4. Să se scrie primele N elemente ale șirului lui Fibonacci, înmulțite cu 8, la locații consecutive în memorie, pe 32 de biți, începând cu adresa A ($A \geq 8$). Valorile A și N se citesc din memorie de la adresele 0, respectiv 4. Pentru verificare, se poate adăuga o buclă de citire a elementelor șirului, la final.

Cod C

```
int mem[100];

mem[0] = 8;  // A = 8
mem[1] = 10; // N = 10


int A = mem[0];
int N = mem[1];
int f1 = 1, f2 = 1, f;
int i = 0;


while (i < N) {
    if (i < 2) {
        f = 1;
    } else {
        f = f1 + f2;
        f1 = f2;
        f2 = f;
    }

    int address = A + i * 4;
    mem[address / 4] = f * 8;

    i++;
}
```

Cod assembly

0: lw \$2, 0(\$0) # incarca A în \$2

1: lw \$4, 4(\$0) # incarca N în \$4

2: addi \$5, \$0, 1 # \$5 = f1 = 1

3: addi \$6, \$0, 1 # \$6 = f2 = 1

4: add \$1, \$0, \$0 # \$1 = i = 0

bucla

5: beq \$1, \$4, end # daca i == N, stop

6: slti \$9, \$1, 2 # daca i < 2 atunci \$9 = 1, altfel \$9 = 0

7: beq \$9, \$0, else # daca \$9 == 0 (i >= 2) sare la else

cazul de baza: i = 0 sau 1

8: addi \$7, \$0, 1 # f = 1

9: j save # sare la save

else:

10: add \$7, \$5, \$6 # f = f1 + f2

11: add \$5, \$6, \$0 # f1 = f2

12: add \$6, \$7, \$0 # f2 = f

save:

13: sll \$10, \$7, 3 # f * 8 -> \$10

14: add \$8, \$1, \$0 # \$8 = i

15: sll \$8, \$8, 2 # \$8 = i * 4

16: add \$8, \$2, \$8 # \$8 = A + i * 4 (adresa de scriere)

17: sw \$10, 0(\$8) # mem[(A+i*4)/4] = f * 8

18: addi \$1, \$1, 1 # i = i + 1

19: j 5 # sare inapoi la inceputul buclei

end:

20: noop

Cod masina

0 => B"100011_00000_00010_0000000000000000", -- X"8C020000" -- lw \$2, 0(\$0) --
incarca un cuvant din adresa \$0 in \$2

1 => B"100011_00000_00100_00000000000000100", -- X"8C040004" -- lw \$4, 4(\$0) --
incarca un cuvant din adresa \$0 + 4 in \$4

2 => B"001000_00000_00101_00000000000000001", -- X"20050001" -- addi \$5, \$0, 1 -- \$5
= 1

3 => B"001000_00000_00110_00000000000000001", -- X"20060001" -- addi \$6, \$0, 1 -- \$6
= 1

4 => B"000000_00000_00000_00001_00000_100000", -- X"00000820" -- add \$1, \$0, \$0 --
\$1 = \$0 + \$0 -- am initializat contorul

5 => B"000100_00001_00100_000000000000001101", -- X"1024000D" -- beq \$1, \$4, 13 --
daca \$1 = \$4, sare la adresa 13

6 => B"001010_00001_01001_00000000000000010", -- X"28290002" -- slti \$9, \$1, 2 -- daca
\$1 < 2 atunci \$9 <= 1 altfel \$9 <= 0

7 => B"000100_01001_00000_000000000000000011", -- X"11200003" -- beq \$9, \$0, 3 --
daca \$9 = 0 sare peste instructiuni

8 => B"001000_00000_00111_00000000000000001", -- X"20070001" -- addi \$7, \$0, 1 -- \$7
= 1

9 => B"000010_000000000000000000000010010", -- X"08000012" -- j 18 -- sare la adresa
18

10 => B"000000_00101_00110_00111_00000_100000",-- X"00A63820" -- add \$7, \$5, \$6 --
\$7 = \$5 + \$6

11 => B"000000_00110_00000_00101_00000_100000",-- X"00C02820" -- add \$5, \$6, \$0 --
\$5 = \$6 + 0

12 => B"000000_00111_00000_00110_00000_100000",-- X"00E03020" -- add \$6, \$7, \$0 --
\$6 = \$7 + 0

13 => B"000000_00000_00111_01010_00011_000000",-- X"000750C0" -- sll \$10, \$7, 3 --
\$10 = \$7 << 3 -- inmultesc cu 8

14 => B"000000_00001_00000_01000_00000_100000",-- X"00204020" -- add \$8, \$1, \$0 --
\$8 = \$1 + 0

15 => B"000000_00000_01000_01000_00010_000000",-- X"00084100" -- sll \$8, \$8, 2 -- \$8
= \$8 << 2 -- inmultesc cu 4

16 => B"000000_00010_01000_01000_00000_100000",-- X"00484020" -- add \$8, \$2, \$8 --
\$8 = \$2 + \$8 -- calculeaza adresa de memorie

17 => B"101011_01000_01010_0000000000000000", -- X"AD0A0000" -- sw \$10, 0(\$8) --
stocheaza \$10 la adresa \$8

18 => B"001000_00001_00001_00000000000000001", -- X"20210001" -- addi \$1, \$1, 1 -- \$1
= \$1 + 1 -- se incr contorul

19 => B"000010_0000000000000000000000000101", -- X"08000005" -- j 5 -- sare inapoi la
adresa 5

20 => B"000000_00000_00000_00000_00000_000000",-- X"00000000" -- noop

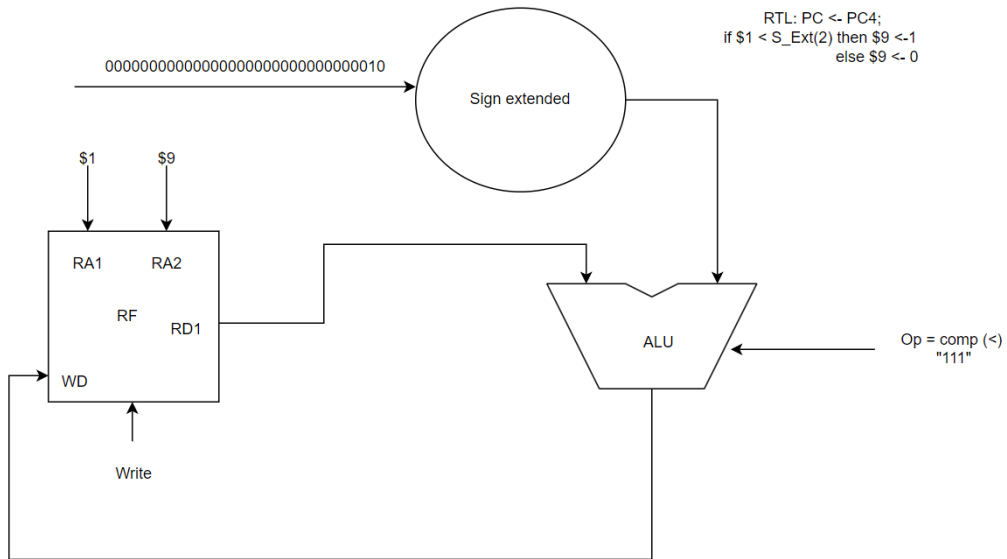
Probleme intampinate

Toate elementele sunt functionale. Problemele intampinate au fost la testarea codului pe placute, unde am observat ca ceva nu este in regula la codul in asamblare. Pe placuta se incrementa bine PC-ul, iar instructiunile se afisau si ele corespunzator, operatiile de salt fiind corecte. Singura problema este ca in MemData nu se afisau corect elemntele din sirul Fibonacci incrementate, se

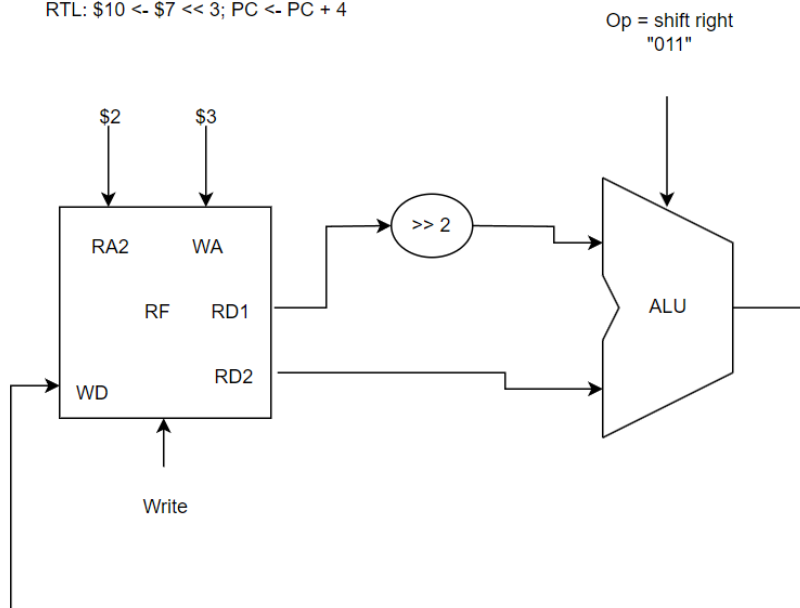
afiseaza doar 8 de 10 ori, dupa care se afiseaza 0, in loc sa se incrementeze si sa se afiseze 8 8 16 24 40 64 104 168 272 440.

Cele 4 comenzi

slli \$9, \$1, 2

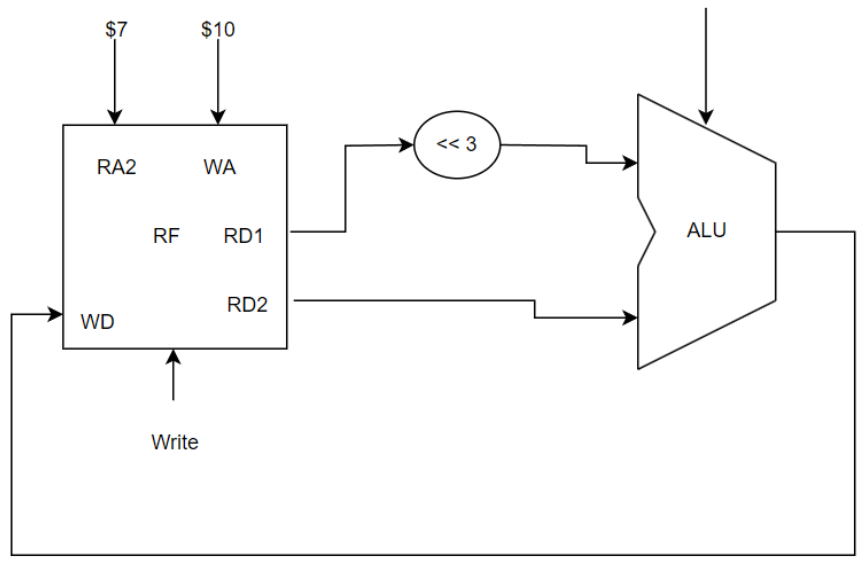


srl \$3, \$2, 2
 RTL: $\$10 \leftarrow \$7 \ll 3$; $PC \leftarrow PC + 4$



sll \$10, \$7, 3
 RTL: $\$10 \leftarrow \$7 \ll 3$; $PC \leftarrow PC + 4$

Op = shift left
 "010"



sw \$10, 0(\$3)

RTL: $M[\$3 + S_Ext(0)] \leftarrow \10

