RAPORT MIPS

-FPGA Basys 3-

Student: Lazarciuc Romeo

Grupa: 30227

Indrumator:Vlad Miclea

1. Instructiunile alese suplimentar

Pentru operatiile de tip R:

* xor : func = 110 => xor $rd, $rs, $rt
* sra : func = 111 => sra $rd, $rs, sa

Pentru operatiile de tip I :

* andi : opcode = 101 => andi $rt, $rs, imm
* xori : opcode = 110 => xorandi $rt, $rs, imm

1. Programul din memorie

Programul din memorie nu este implementat astfel incat sa calculeze ceva concret. In programul meu din memorie se afla toate functiile astfel incat sa poata fii testate.Incepand cu prima instructiune se folosesc 3 registrii ,unul pentru un termen al operatiei al doilea pentru urmatorul termen al operatiei iar al treilea este folosit pentru a pune rezultatul operatiei in el. Aproape la toate operatiile de tip R se aplica aceasta regula. La operatiile de tip I avem si operatii cu imediat care functioneaza aproape identic doar ca al doilea termen nu este luat din registu ci este privit ca o constanta. Aceasta constanta ( numita imediat ) poate fii folosita in mai multe feluri: ca un termen al unei operatii, ca o adresa la care trebuie sa ajungem(beq).

**In memoria rom, programul este sub forma:**

I0: 000\_001\_010\_011\_0\_000 add $1,$2,$3 rd1=3,rd2=1 RegWrite=1; RegDst=1,AluRes=4

I1: 000\_001\_010\_011\_0\_001 sub $1,$2,$3 rd1=3,rd2=1 RegWrite=1; RegDst=1,AluRes=2

I2: 000\_001\_000\_010\_1\_010 sll $1,$0,$2 rd1=3,rd2=1 RegWrite=1; RegDst=1,AluRes=6

I3: 000\_001\_000\_010\_1\_011 srl $1,$0,$2 rd1=3,rd2=1 RegWrite=1; RegDst=1,AluRes=1

I4: 000\_001\_010\_011\_0\_100 and $1,$2,$3 rd1=3,rd2=1 RegWrite=1; RegDst=1,AluRes=1

I5: 000\_001\_010\_011\_0\_101 or $1,$2,$3 rd1=3,rd2=1 RegWrite=1; RegDst=1,AluRes=3

I6: 000\_001\_010\_011\_0\_110 xor $1,$2,$3 rd1=3,rd2=1 RegWrite=1; RegDst=1,AluRes=2

I7: 000\_001\_000\_010\_1\_111sra $1,$0,$2 rd1=3,rd2=1 RegWrite=1; RegDst=1,AluRes=8001

I8: 001\_001\_010\_0000001 addi $1,$2,1 rd1=3,rd2=8001, RegWrite=1 ALUOp=001 AluSrc=1 AluRes=4 ExtendedImmediate=1

I9: 010\_001\_010\_0000001 lw $2,imm($1) MemtoReg=1 RegWrite=1 ALUOp=010 AluSrc=1 wd=134 ExtendedImmediate=1

I10: 011\_001\_010\_0000001 sw $2,imm($1) MemWrite<='1' ALUOp=011 AluSrc=1

I11: 100\_001\_010\_0000011 beq $1,$2,1 ALUOp<=100, Branch=1; ExtendedImmediate=3

I12: 101\_001\_010\_0000111 andi $1,$2,7 RegWrite=1, ALUOp=101,AluSrc=1, AluRes=3

ExtendedImmediate=7

I13: 110\_001\_010\_0000001 xori $1,$2,1 RegWrite=1 ALUOp=110, AluSrc=1

I15: 111\_0000000000001 jmp Jumpu=1 ExtendedImmediate=1

1. Semnalele de control

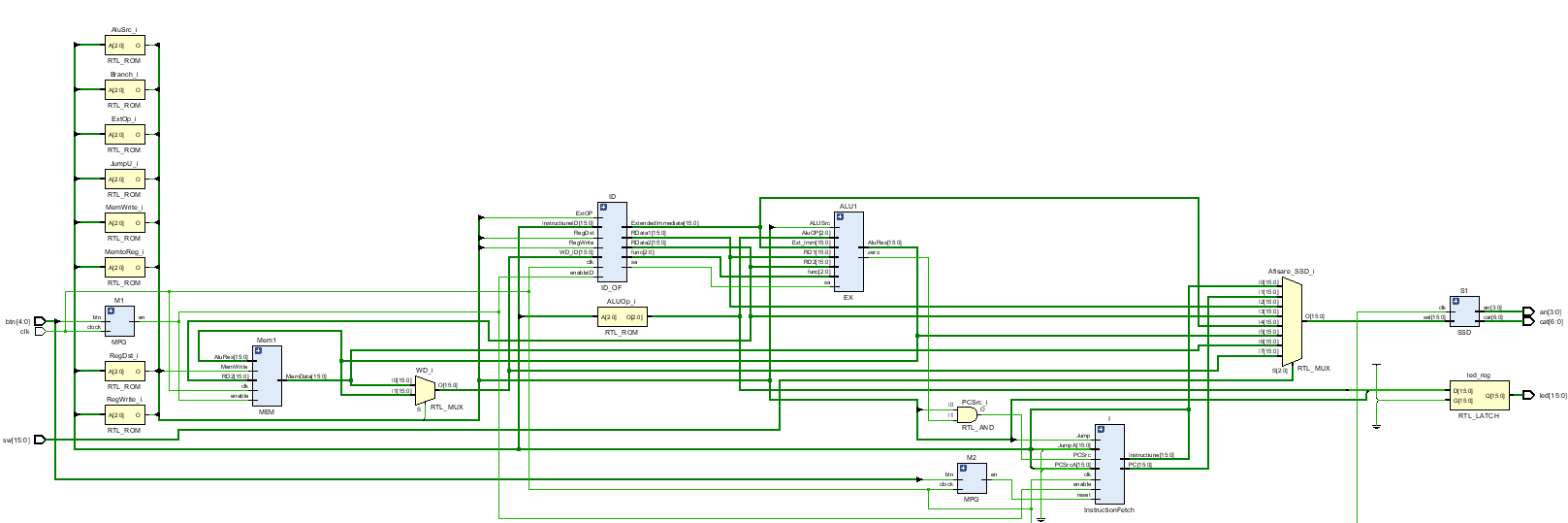
**Semnale control MIPS16 pentru Anexa 5**

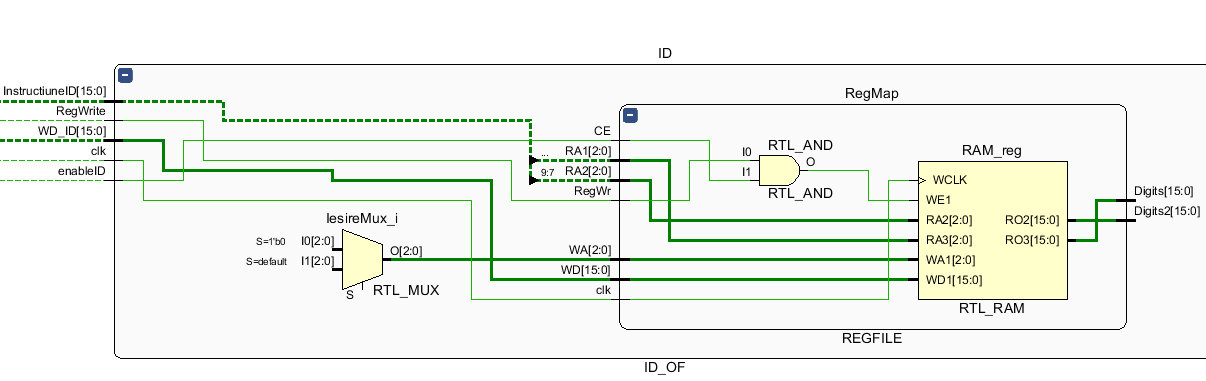
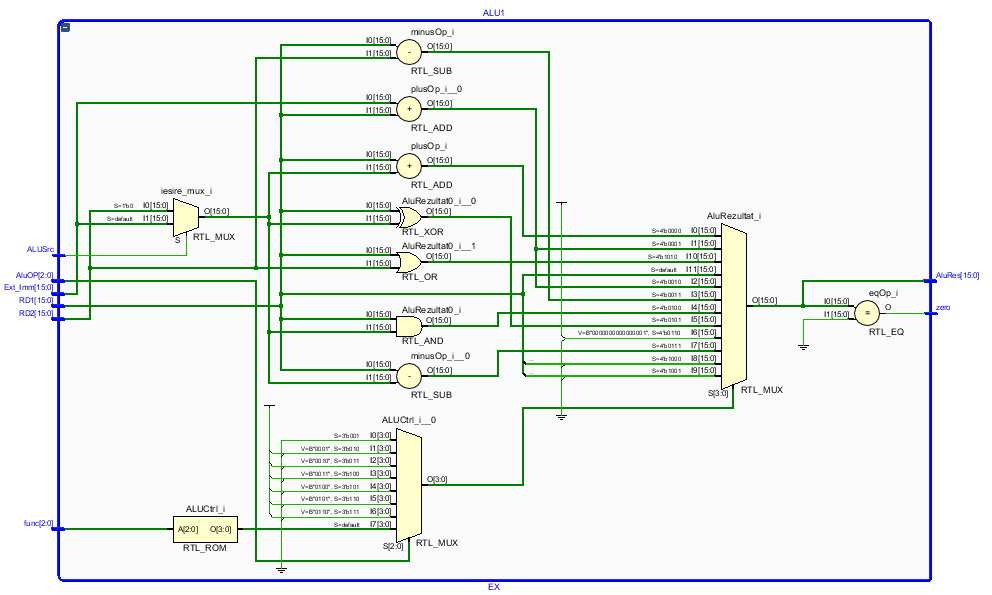
<?> ϵ {GEZ, NE, GTZ}

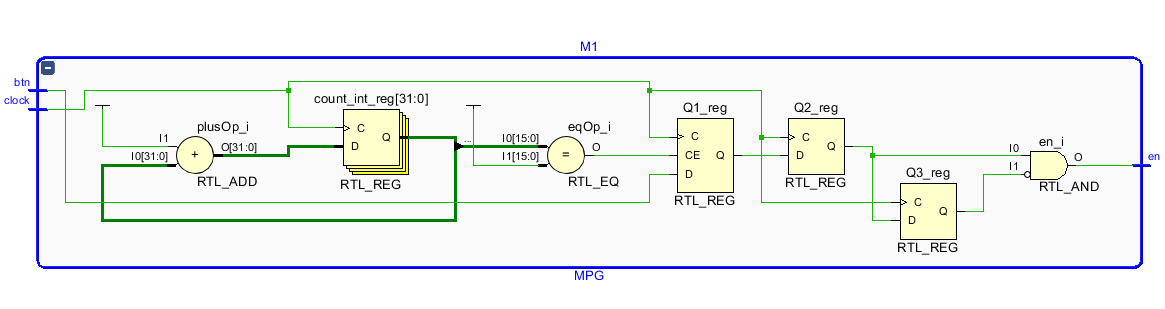
|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| **Instrucțiune** | **Opcode** *Instr(15-13)* | **RegDst** | **ExtOp** | **ALUSrc** | **Branch** | **Br<?>** (opțional) | **Jump** | **MemWrite** | **MemtoReg** | **RegWrite** | **ALUOp (2:0)** | **func**  *Instr(2-0)* | **ALUCtrl (3:0)** | **JmpR** (opțional) |
| add | 000 | 1 | 0 | 0 | 0 |  | 0 | 0 | 0 | 1 | 000 | 000 | 0000(+) |  |
| sub | 000 | 1 | 0 | 0 | 0 |  | 0 | 0 | 0 | 1 | 000 | 001 | 0111(-) |  |
| sll | 000 | 1 | 0 | 0 | 0 |  | 0 | 0 | 0 | 1 | 000 | 010 | 1000(<<l) |  |
| srl | 000 | 1 | 0 | 0 | 0 |  | 0 | 0 | 0 | 1 | 000 | 011 | 1001(>>l) |  |
| and | 000 | 1 | 0 | 0 | 0 |  | 0 | 0 | 0 | 1 | 000 | 100 | 0100(&) |  |
| or | 000 | 1 | 0 | 0 | 0 |  | 0 | 0 | 0 | 1 | 000 | 101 | 1010(^) |  |
| xor | 000 | 1 | 0 | 0 | 0 |  | 0 | 0 | 0 | 1 | 000 | 110 | 0101(^-) |  |
| sra | 000 | 1 | 0 | 0 | 0 |  | 0 | 0 | 0 | 1 | 000 | 111 | 1011(>>a) |  |
| addi | 001 | 0 | 1 | 1 | 0 |  | 0 | 0 | 0 | 1 | 001 | XXX | 0000(+v) |  |
| lw | 010 | 0 | 1 | 1 | 0 |  | 0 | 0 | 1 | 1 | 010 | XXX | 0001 |  |
| sw | 011 | 0 | 1 | 1 | 0 |  | 0 | 1 | 0 | 0 | 011 | XXX | 0010 |  |
| beq | 100 | 0 | 1 | 0 | 1 |  | 0 | 0 | 0 | 0 | 100 | XXX | 0011 |  |
| andi | 101 | 0 | 1 | 1 | 0 |  | 0 | 0 | 0 | 1 | 101 | XXX | 0100(&v) |  |
| xori | 110 | 0 | 1 | 1 | 0 |  | 0 | 0 | 0 | 1 | 110 | XXX | 0101(^v) |  |
| jmp | 111 | 0 | 0 | 0 | 0 |  | 1 | 0 | 0 | 0 | 111 | XXX | 0110 |  |

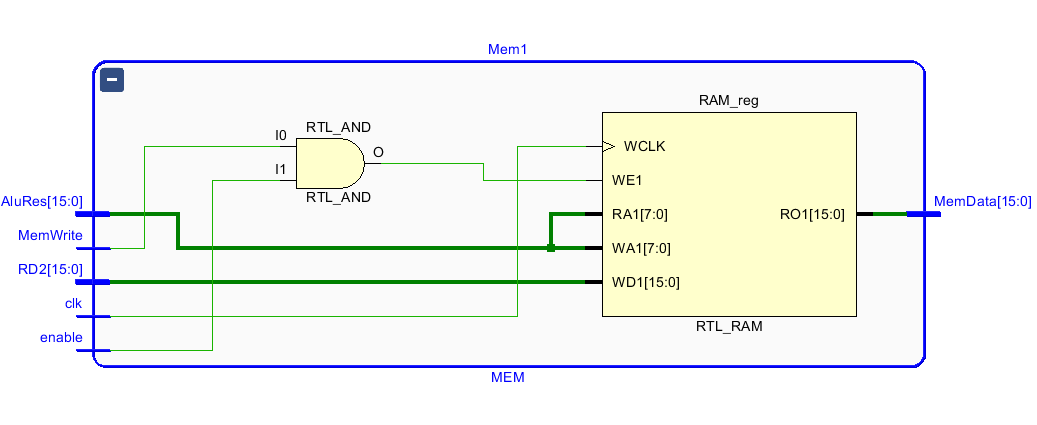
*Tipuri de operații care se pun în paranteză la ALUOp si ALUCtrl:* {(+), (-), (&), (I), (^), (<<l), (<<lv), (>>l), (>>a), (<)}, ^ *- XOR, l - logic, a - aritmetic, v - cu variabilă*

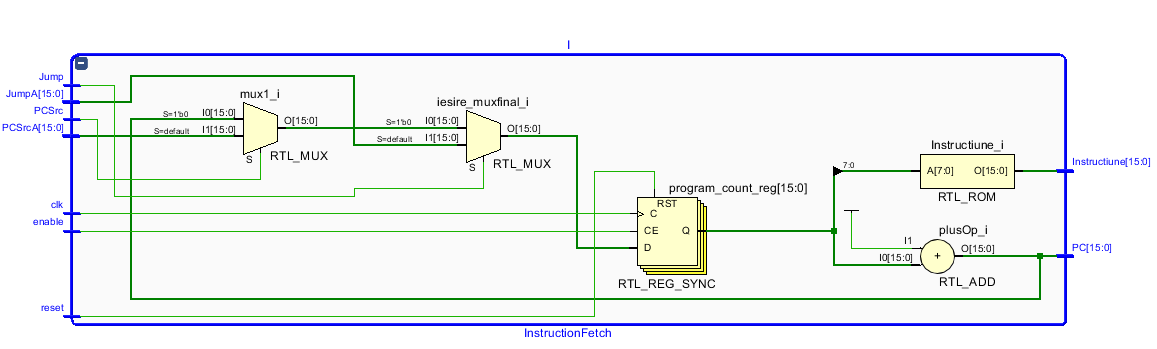
1. Schematic

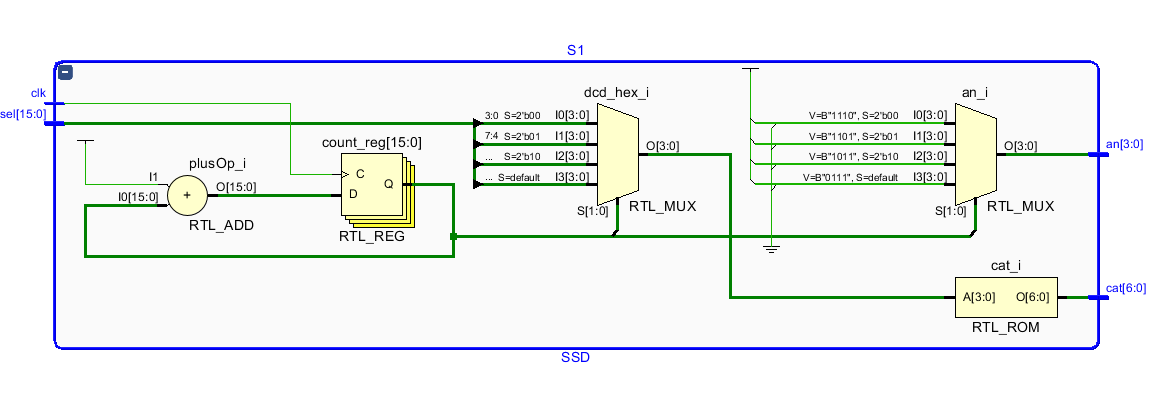




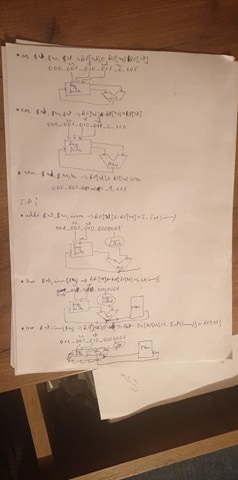
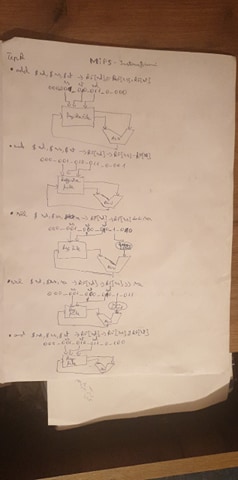
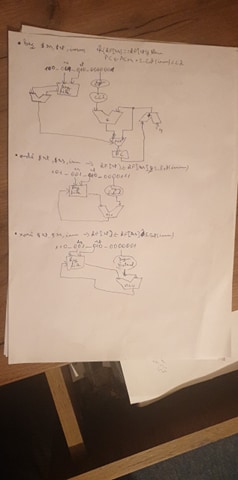
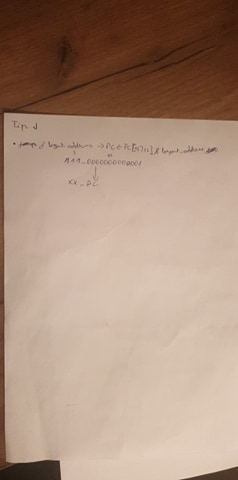








1. Codificarea operatiilor



1. Alte observatii

Partea de implementare si de scriere a fiecarei componente in parte au fost facute fara a intampina probleme. Fiecare laborator a fost dus pana la capat, chiar au fost testate si pe placa. Nu exista erori de cod sau erori de compilare. Dupa parerea mea MIPS-ul functioneaza corect, fiecare functie a acestuia a fost testata pe placa.