**RAPORT DE ACTIVITATE**

1. Cele 4 instructiuni alese suplimentar:

-XOR (de tipul R) : realizeaza operatia logica de xor pe doua intrari de 16 biti;

xor $rd, $rs, $rt

RF[RD] =RF[rs] xor RF[rt]

-NOR (de tipul R): realizeaza operatia logica de nor pe doua intrari de 16 biti;

nor $rd, $rs, $rt

RF[RD] =RF[rs] nor RF[rt]

-ANDI (de tipul I) : realizeaza operatia logica de AND pe doua intrari de 16 biti (cea de-a doua intrare se obtine dupa extinderea cu 0);

andi $rt, $rs, imm

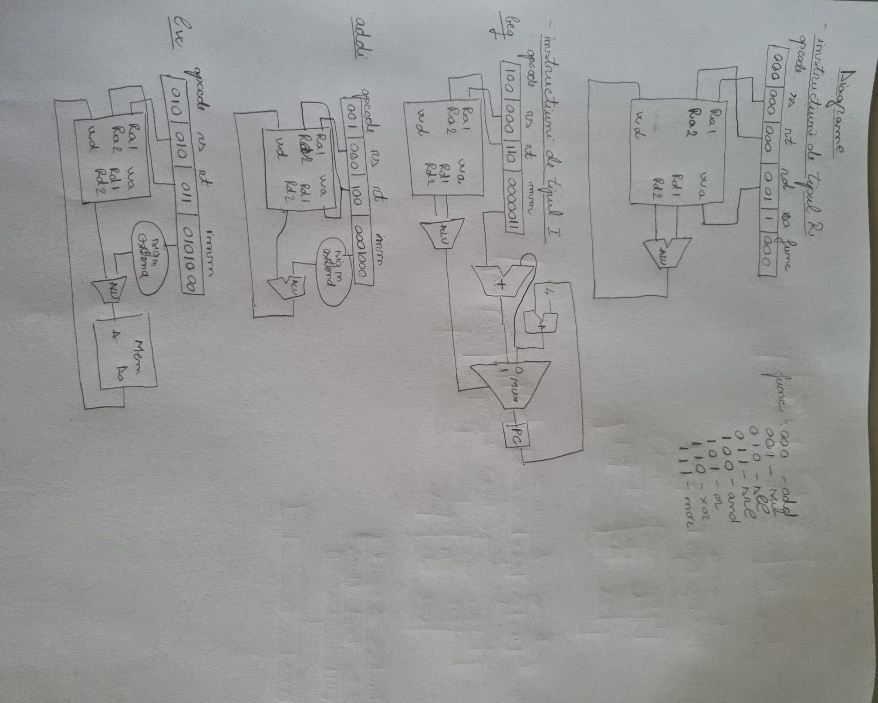
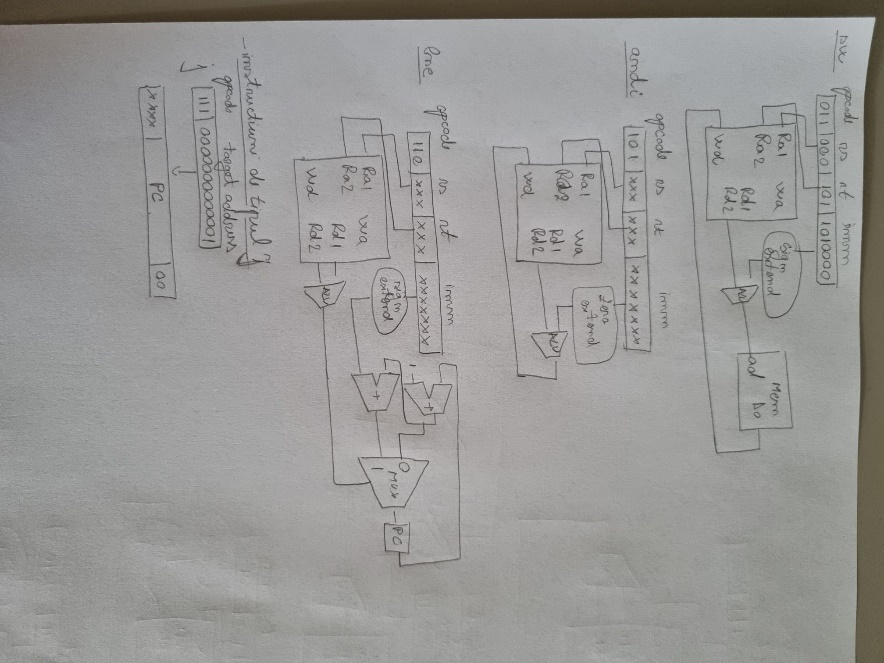
RF[rt] =RF[rs] andi Z\_ext(imm)

-BNE(de tipul I): realizeaza saltul conditional daca 2 registre sunt diferite;

bne $s, $t, offset

if $s != $t PC <= PC + 1 + (offset << 2); else PC <= PC + 1;

Toate instructiunile sunt descriese:

2)Tabelul cu semnalele de control

|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| **Instrucțiune** | **Opcode** *Instr(15-13)* | **RegDst** | **ExtOp** | **ALUSrc** | **Branch** | **<Br?>** (opțional) | **Jump** | **JmpR** (opțional) | **MemWrite** | **MemtoReg** | **Reg Write** | **ALUOp (1:0)** | **func**  *Instr(2-0)* | **ALUCtrl (2:0)** |
| add | 000 | 1 | X | 0 | 0 |  | 0 |  | 0 | 0 | 1 | 000 (+) | 000 | 000 (\_) |
| sub | 000 | 1 | X | 0 | 0 |  | 0 |  | 0 | 0 | 1 | 000(-) | 001 | 001 |
| sll | 000 | 1 | X | 0 | 0 |  | 0 |  | 0 | 0 | 1 | 000(<<) | 010 | 010 |
| srl | 000 | 1 | X | 0 | 0 |  | 0 |  | 0 | 0 | 1 | 000(>>) | 011 | 011 |
| and | 000 | 1 | X | 0 | 0 |  | 0 |  | 0 | 0 | 1 | 000(&) | 100 | 100 |
| Or | 000 | 1 | X | 0 | 0 |  | 0 |  | 0 | 0 | 1 | 000(|) | 101 | 101 |
| xor | 000 | 1 | X | 0 | 0 |  | 0 |  | 0 | 0 | 1 | 000(^) | 110 | 110 |
| nor | 000 | 1 | X | 0 | 0 |  | 0 |  | 0 | 0 | 1 | 000(not|) | 110 | 111 |
| addi | 001 | 0 | 1 | 1 | 0 |  | 0 |  | 0 | 0 | 1 | 001(+) | Xxx | 000 |
| Lw | 010 | 0 | 1 | 1 | 0 |  | 0 |  | 0 | 1 | 1 | 010(+) | Xxx | 000 |
| Sw | 011 | X | 1 | 1 | 0 |  | 0 |  | 1 | X | 0 | 011(+) | Xxx | 000 |
| beq | 100 | X | 1 | 0 | 1 |  | 0 |  | 0 | X | 0 | 100(-) | Xxx | 001 |
| andi | 101 | 0 | 1 | 1 | 0 |  | 0 |  | 0 | 0 | 1 | 101(&l) | Xxx | 010 |
| bne | 110 | X | 1 | 0 | 1 |  | 0 |  | 0 | X | 0 | 110(-) | Xxx | 001 |
| j | 111 | x | X | x | 0 |  | 1 |  | 0 | x | 0 | 111 | xxx | xxx |
|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |

3)Descrierea programului

Avand niste valri date in memorie, o sa calculam suma dublul primelor 8 numere de la adresa 10.

-cod C:

for(int i=0;i<8;i++){

sum+=v[i]\*2 ;

}

-programul executat de processor

0 add $1, $0, $0 -- I = 0, contorul buclei

1 addi $4, $0, 8 --se salveaza numarul de iteratii(8)

2 add $2, $0 , $0 –intializarea indexului locatiei de memorie

3 add $5, $0, $0 --sum=0

4 beq $1, $4, 7 –daca s-au facut 8 iteratii, salt in afara buclei

5 lw $3, 10($2) --in $3 se aduce elemental current din sir

6 sll $3, $3, 1 – se face inmultirea cu 2

7 sw $3, 10($2) –salvarea noii valori

8 add $5, $5, $3 --actualizarea sumei

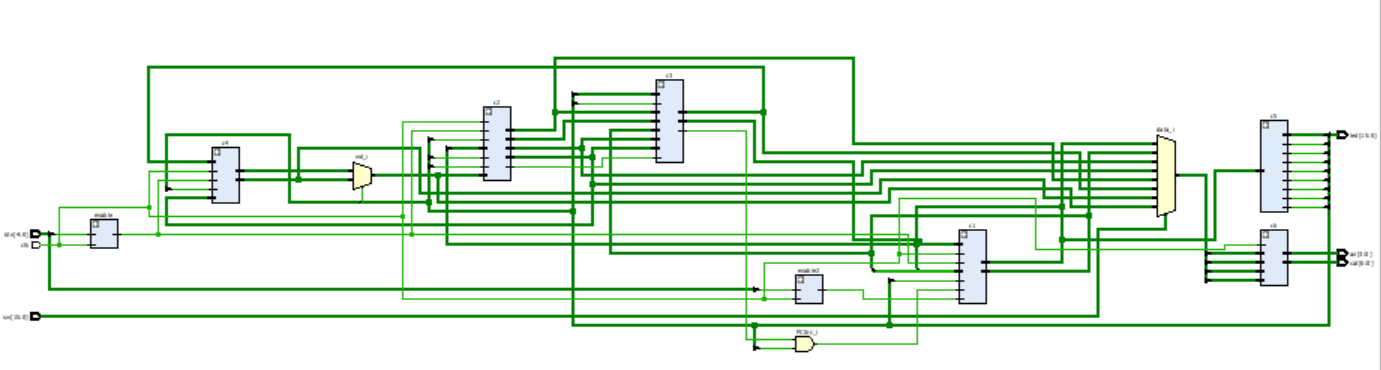
9 addi $2, $2, 1 – inexul urmatorului element din sir

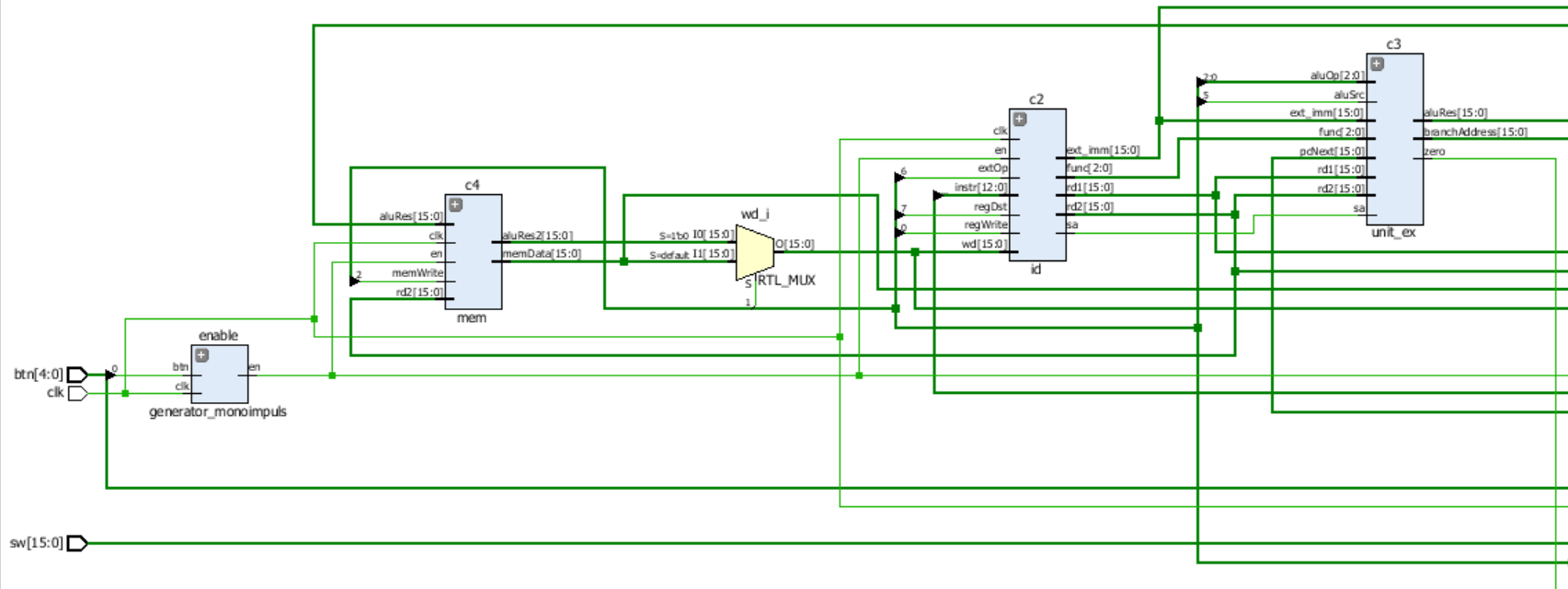
10 addi $1, $1, 1 –actualizarea contorului buclei

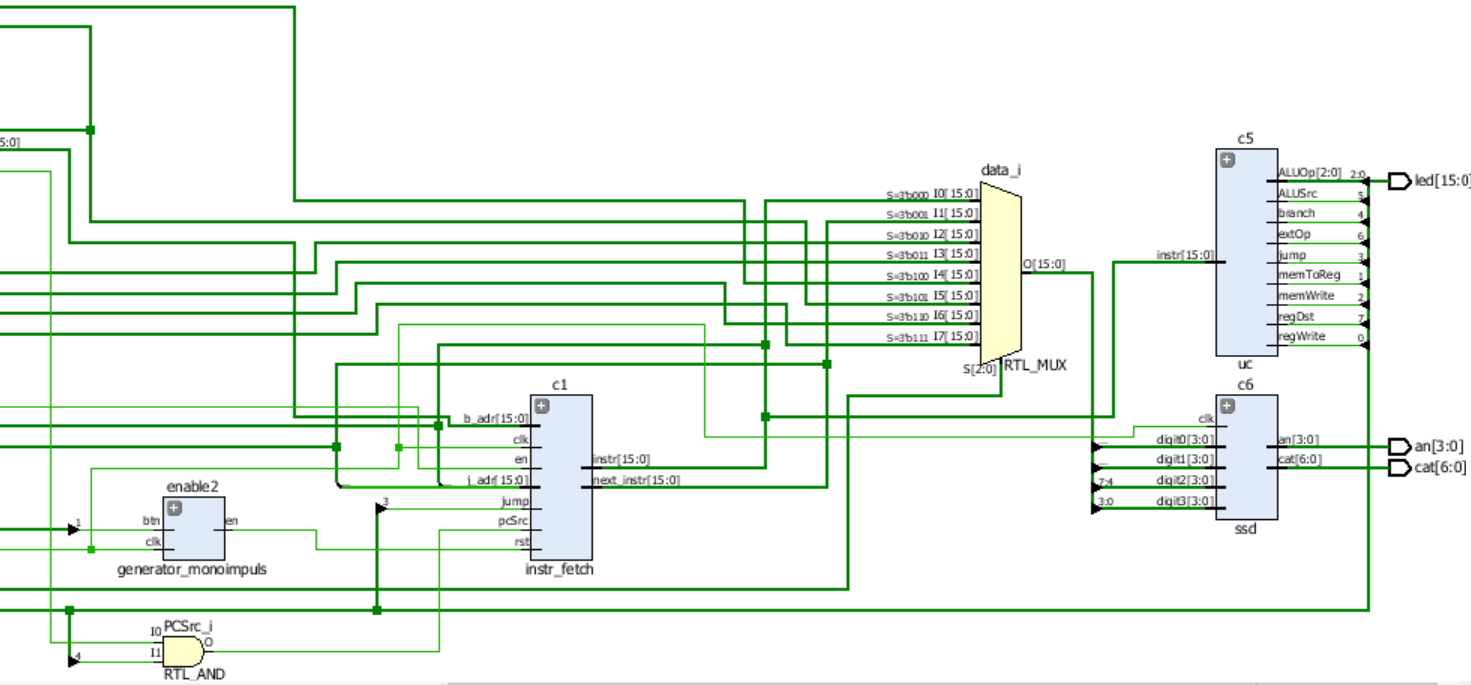
11 j 4 –salt inceputului buclei

12 sw $5, 18($0) –salvarea sumei in memorie la adresa 48

5)RTL schematic







6)Trasarea executiei

|  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| **Pas** | **SW(7:5)** | “000” | “001” | “010” | “011” | “100” | “101” | “110” | “111” | **De completat numai pentru instrucțiuni de salt** | |
| **Instr** (*în asamblare*) | **Instr** (*hexa*) | **PC+1** | **RD1** | **RD2** | **Ext\_Imm** | **ALURes** | **MemData** | **WD** | **BranchAddr** | **JumpAddr** |
| 0 | add $1, $0, $0 | x”0018” | x”0001” | x”0000” | x”0000” | x”0018” | x”0000” | x”0000” | x”0000” |  |  |
| 1 | Add $4, $0, 8 | x”2208” | x”0002” | x”0000” | x”1000” | x”2208” | x”1000” | x”0000” | x”1000” |  |  |
| 2 | add $2, $0 , $0 | x”0028” | x”0003” | x”0000” | x”0000” | x”0028” | x”0000” | x”0000” | x”0000” |  |  |
| 3 | add $5, $0, $0 | x”0058” | x”0004” | x”0000” | x”0000” | x”0058” | x”0000” | x”0000” | x”0000” |  |  |
| 4 | beq $1, $4, 7 | x”9087” | x”0005” | x”0100” | x”0001” | X”0007” | x”0111” | x”0000” | x”0001” |  |  |
| 5 | lw $3, 10($2) | x”498A” | x”0006” | x”0010” | x”0011” | x”000A” | x”000A” | x”0000” | x”0011” |  |  |
| 6 | sll $3, $3, 1 | x”01BA” | x”0007” | x”0011” | x”0001” | x”003A” | x”0018” | x”0000” | x”0001” |  |  |
| 7 | sw $3, 10($2) | x”698A” | x”0008” | x”0010” | x”0011” | x”000A” | x”001A” | x”001A” | x”0011” |  |  |
| 8 | add $5, $5, $3 | x”15D8” | x”0009” | x”0101” | x”0011” | x”0058” | x”1000” | x”0000” | x”0011” |  |  |
| 9 | addi $2, $2, 1 | x”2901” | x”000A” | x”0010” | x”0001” | x”0001” | x”0011” | x”0000” | x”0010” |  |  |
| 10 | addi $1, $1, 1 | x”2481” | x”000B” | x”0001” | x”0001” | x”0001” | x”0010” | x”0000” | x”0001” |  |  |
| 11 | j 4 | x”E004” | x”000C” | X”0000” | x”0000” | x”0000” | x”0000” | x”0000” | x”0000” | x”000C” | x”0004” |
| 12 | sw $5, 18($0) | x”6292” | x”000D” | x”0000” | x”0101” | x”0012” | x”0101” | x”0101” | x”0101” |  |  |

Am testat proiectul pe placuta, iar acesta a fost executat correct.