**Raport Procesor MIPS Pipeline**

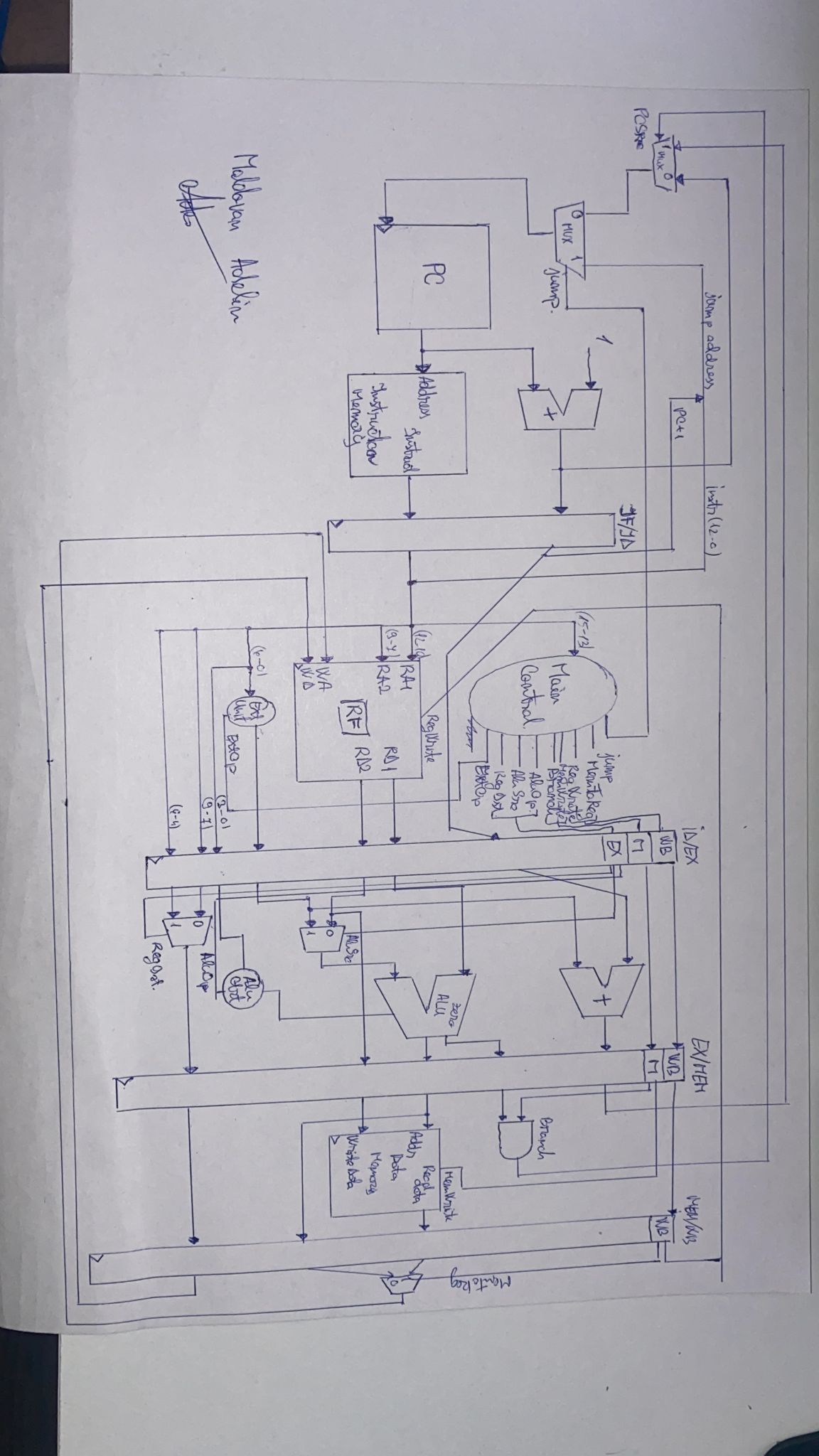
Moldovan Adelin

Grupa 302210

1. Nu au fost facute modificari particulare asupra instructiunilor suplimentare.
2. Tabelul cu descrierea registrilor de pipeline.

|  |  |  |  |
| --- | --- | --- | --- |
| **REG\_IF\_ID(31** – **0)** | **REG\_ID\_EX(82** – **0)** | **REG\_EX\_MEM(55** – **0)** | **REG\_MEM\_WB(36** – **0)** |
| IF.Instruction(31 – 16) | UC.MemtoReg(82) | ID/EX.MemtoReg(55) | EX/MEM.MemtoReg(36) |
| IF.PC + 1(15 – 0) | UC.RegWrite(81) | ID/EX.RegWrite(54) | EX/MEM.RegWrite(35) |
|  | UC.MemWrite(80) | ID/EX.MemWrite(53) | MEM.MemData |
|  | UC.Branch(79) | ID/EX.Branch(52) | EX/MEM.ALURes(18-3) |
|  | UC.ALUOp(78-76) | EX.BranchAddress(51-36) | EX/MEM.outmuxRegDst(2-0) |
|  | UC.ALUSrc(75) | EX.Zero(35) |  |
|  | UC.RegDst(74) | EX.ALURes(34-19) |  |
|  | IF/ID.PC+1(73-58) | ID/EX.RD2(18-3) |  |
|  | ID.RD1(57-42) | Test\_env.outmuxRegDst(2-0) |  |
|  | ID.RD2(41-26) |  |  |
|  | ID.Ext\_Imm(25-10) |  |  |
|  | IF/ID.Instruction(9-6) |  |  |
|  | IF/ID.Instruction(5-3) |  |  |
|  | IF/ID.Instruction(2-0) |  |  |

1. Schema procesorului mips pipeline.



s

1. Analiza procesorului mips pipeline. Identificarea hazardurilor. Programul final cu noop-urile inserate. Diagrama pipeline.

Programul original si programul cu “noop”:

|  |  |
| --- | --- |
| 1: addi $1,$0,1 ***pun in registrul 1 valoarea* 1**  2: ori $2,$0,10 ***in registrul 2 adaug 10***  3: addi $5,$0,0 ***cu ajutorul reg. $5 imi parcurg memoria***  4: ori $4,$0,0 ***in reg. $4- fac suma elementelor din memorie***  5: andi $7,$1,1 ***verific daca numarul din $1 e par sau impar***  6: beq $7,$0,3 ***daca e par sar la eticheta “par” pentru a-l inmulti cu “2”***  7: addi $7,$1,-1 ***daca e impar, atunci scad din acesta valoarea “1”***  8: sw $7,0($5) ***salvez valoarea actuala (par, “\*2”, impar,”-1”) in memorie***  9: j 11 ***sar la instructiunea 12 din program***  10: sll $6,$1,1 ***daca numarul e par il inmultesc cu “2”***  11: sw $6,0($5) ***scot valoarea actuala din memorie si o pun registrul $3***  12: lw $3,0($5) ***adun la suma($4) valoarea din memorie extrasa din instr precedenta***  13: add $4,$4,$3 ***trec la urmatorul element din sir***  14: addi $1,$1,1 ***cresc valoarea adresei memoriei***  15: addi $5,$5,1 ***cresc valoarea adresei memoriei***  16: beq $1,$2,1 ***daca am ajuns la limita superioara, merg la ultima instructiune***  17: j 4 ***daca nu am ajuns la limita superioara, imi parcurg iar pasii anteriori***  18: sw $4,0($5) ***la finalul programului, salvez si suma elementelor sirului “modificate” in memorie*** | 1: addi $1,$0,1  2: ori $2,$0,10  3: addi $5,$0,0  4: ori $4,$0,0  5: andi $7,$1,1  **6: NoOp**  **7: NoOp**  8: beq $7,$0,7  **9: NoOp**  **10: NoOp**  **11: NoOp**  12: addi $7,$1,-1  **13: NoOp**  14: j 19  15: sw $7,0($5)  16: sll $6,$1,1  **17: NoOp**  **18: NoOp**  19: sw $6,0($5)  20: lw $3,0($5)  **21: NoOp**  **22: NoOp**  23: add $4,$4,$3  24: addi $1,$1,1  **25: NoOp**  26: addi $5,$5,1  27: beq $1,$2,5  **28: NoOp**  **29: NoOp**  **30: NoOp**  31: j 4  **32: NoOp**  33: sw $4,0($5) |

Astfel intre instructiuniile **5 si 6($7); 7 si 8($7); 10 si 11($6); 12 si 13($3); 14 si 16($1);** exista hazard de tip RAW. Iar la instructiunile de salt **beq** si **j** exisa hazard de control.

Diagrama de pipleine:

|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| **Nr. Crt.** | **Instruction** | **CC1** | **CC2** | **CC3** | **CC4** | **CC5** | **CC6** | **CC7** | **CC8** | **CC9** | **CC10** | **CC11** | **CC12** | **CC13** | **CC14** | **CC15** | **CC16** | **CC17** | **CC18** |
| 5 | andi $7,$1,1 | IF | ID | EX | MEM | WB  ($7) |  |  |  |  |  |  |  |  |  |  |  |  |  |
| 6 | beq $7,$0,3 |  | IF | ID  ($7) | EX | MEM | WB |  |  |  |  |  |  |  |  |  |  |  |  |
| 7 | addi $7,$1,-1 |  |  | IF | ID | EX | MEM | WB ($7) |  |  |  |  |  |  |  |  |  |  |  |
| 8 | sw $7,0($5) |  |  |  | IF | ID ($7) | EX | MEM | WB |  |  |  |  |  |  |  |  |  |  |
| 9 | j 11 |  |  |  |  | IF | ID | EX | MEM | WB |  |  |  |  |  |  |  |  |  |
| 10 | sll $6,$1,1 |  |  |  |  |  | IF | ID | EX | MEM | WB ($6) |  |  |  |  |  |  |  |  |
| 11 | sw $6,0($5) |  |  |  |  |  |  | IF | ID ($6) | EX | MEM | WB |  |  |  |  |  |  |  |
| 12 | lw $3,0($5) |  |  |  |  |  |  |  | IF | ID | EX | MEM | WB ($3) |  |  |  |  |  |  |
| 13 | add $4,$4,$3 |  |  |  |  |  |  |  |  | IF | ID ($3) | EX | MEM | WB |  |  |  |  |  |
| 14 | addi $1,$1,1 |  |  |  |  |  |  |  |  |  | IF | ID | EX | MEM | WB ($1) |  |  |  |  |
| 15 | addi $5,$5,1 |  |  |  |  |  |  |  |  |  |  | IF | ID | EX | MEM | WB |  |  |  |
| 16 | beq $1,$2,1 |  |  |  |  |  |  |  |  |  |  |  | IF | ID ($1) | EX | MEM | WB |  |  |
| 17 | j 4 |  |  |  |  |  |  |  |  |  |  |  |  | IF | ID | EX | MEM | WB |  |
| 18 | sw $4,0($5) |  |  |  |  |  |  |  |  |  |  |  |  |  | IF | ID | EX | MEM | WB |

1. Nu exista activitati(parti din procesor) incomplete din laboratoarele 9-10. Este totul implementat.
2. Este totul descris in VHDL. Nu exista erori. RTL schematic:

Diagram

Description automatically generated

1. Nu a fost testat pe placa, dar consider ca este functional.