Raport MIPS – CICLU UNIC 16 biti

1. MIPS-ul are ca si componente urmatoarele:

* MPG
* SSD
* InstructionFetch
* RegisterFile
* Instruction Decode
* Memory
* Unit Control UC
* Unit Execute UEX
* ALU

Aceste componente sunt scrise in diferite surse vhdl.

Fisierul principal in care sunt adaugate ca si componente toate cele enumerate mai sus este denumit test\_env.vhdl.

In fisierul principal mai sunt incluse unele atribuiri si doua mux-uri necesare pentru a lega toate componentele microprocesorului.

Am generat fisierul bitstream si schema microprocesorului pe care o voi adauga ulterior in acest document.

1. a. Cele 4 instructiuni pe care le-am ales suplimentar sunt:

* cele de tip R sunt Logical XOR si Logical NXOR
* cele de tip I sunt Logical ORI si SUBI

Functia XOR efectueaza SAU-EXCLUSIV logic intre numerele memorate la adresele rs si rt si memoreaza rezultatul in rd.

XOR $rd, $rs, $rt

cod: 000\_rs\_rt\_rd\_110

Functia NXOR efectueaza NOT XOR logic intre numerele memorate la adresele rs, rt si memoreaza rezulatul la adresa rd.

NXOR $rd, $rs, $rt

cod: 000\_rs\_rt\_rd\_111.

Functia ORI face SAU logic intre valoarea de la adresa rt si o valoare imediata si memoreaza rezultatul la adresa rs.

ORI $rs, $rt, imm

PC <- PC + 1   
 cod:101\_rs\_rt\_imm

Functia SUBI efectueaza scaderea intre valoarea de la adresa rt si o valoare imediata si memoreaza rezultataul la adresa rs.

SUBi $rs, $rt, imm

Cod: 110\_rs\_rt\_imm

|  |  |  |  |
| --- | --- | --- | --- |
| **REG\_IF\_ID(31** – **0)** | **REG\_ID\_EX(82** – **0)** | **REG\_EX\_MEM(55** – **0)** | **REG\_MEM\_WB(36** – **0)** |
| Instruction(31 – 16) | RegDst(0) | MemtoReg(0) | MemtoReg(0) |
| PC + 1(15 – 0) | ALUSrc(1) | RegWrite(1) | RegWrite(1) |
|  | Branch(2) | MemWrite(2) | ReadData(17 – 2) |
|  | MemWrite(3) | Branch(3) | ALURes(33 – 18) |
|  | MemtoReg(4) | BranchAddress(19 – 4) | WriteAddress(36 – 34) |
|  | RegWrite(5) | Zero(20) |  |
|  | ALUOp(8 – 7) | ALURes(36 – 21) |  |
|  | sa(9) | RD2(52 – 37) |  |
|  | PC + 1(25 – 10) | WriteAddress(55 – 53) |  |
|  | RD1(41 – 26) |  |  |
|  | RD2(57 – 42) |  |  |
|  | ExtImm(73 – 58) |  |  |
|  | Func(76 – 74) |  |  |
|  | rt(79 – 77) |  |  |
|  | rd(82 – 80) |  |  |

1. **Tabel cu descrierea regiștrilor de pipeline**

b. **Semnale control MIPS16 pentru Anexa 5**

|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| **Instrucțiune** | **Opcode** *Instr(15-13)* | **RegDst** | **ExtOp** | **ALUSrc** | **Branch** | **<Br?>** (opțional) | **Jump** | **JmpR** (opțional) | **MemWrite** | | **MemtoReg** | **Reg Write** | **ALUOp (2:0)** | **func**  *Instr(2-0)* | **ALUCtrl (2:0)** |
| ADD | 000 | 1 | 0 | 0 | 0 |  | 0 |  | | 0 | 0 | 1 | 00 | 000 | 000 |
| SUB | 000 | 1 | 0 | 0 | 0 |  | 0 |  | | 0 | 0 | 1 | 00 | 001 | 001 |
| SLL | 000 | 1 | 0 | 0 | 0 |  | 0 |  | | 0 | 0 | 1 | 00 | 010 | 010 |
| SRL | 000 | 1 | 0 | 0 | 0 |  | 0 |  | | 0 | 0 | 1 | 00 | 011 | 011 |
| AND | 000 | 1 | 0 | 0 | 0 |  | 0 |  | | 0 | 0 | 1 | 00 | 100 | 100 |
| OR | 000 | 1 | 0 | 0 | 0 |  | 0 |  | | 0 | 0 | 1 | 00 | 101 | 101 |
| XOR | 000 | 1 | 0 | 0 | 0 |  | 0 |  | | 0 | 0 | 1 | 00 | 110 | 110 |
| NXOR | 000 | 1 | 0 | 0 | 0 |  | 0 |  | | 0 | 0 | 1 | 00 | 111 | 111 |
| ADDI | 001 | 0 | 1 | 1 | 0 |  | 0 |  | | 0 | 0 | 1 | 01 | - | 000 |
| LW | 010 | 0 | 1 | 1 | 0 |  | 0 |  | | 0 | 1 | 1 | 01 | - | 000 |
| SW | 011 | 0 | 1 | 1 | 0 |  | 0 |  | | 1 | 0 | 0 | 01 | - | 000 |
| BEG | 100 | 0 | 1 | 0 | 1 |  | 0 |  | | 0 | 0 | 0 | 10 | - | 001 |
| ORI | 101 | 0 | 0 | 1 | 0 |  | 0 |  | | 0 | 0 | 1 | 11 | - | 101 |
| SUBI | 110 | 0 | 1 | 1 | 0 |  | 0 |  | | 0 | 0 | 1 | 10 | - | 001 |
| J | 111 | 0 | 0 | 0 | 0 |  | 1 |  | | 0 | 0 | 0 | 00 | - | 000 |
|  |  |  |  |  |  |  |  |  | |  |  |  |  |  |  |

Codul pe care l-am incarcat in memoria ROM este suma primelor N numere pare mai mari decat 0, N fiind luat din registrul $5.

|  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- |
| Instructiune | Codificare | PC next | RD1 | RD2 | WD | EXT\_IMM | Alu Res | memData |
| XOR $0, $0 , $0 | 0006 | 1 | 0000 | 0000 | 0006 | 0000 | 0000 | 0000 |
| XOR $1, $1 , $1 | 0496 | 2 | 0001 | 0001 | 0016 | 0000 | 0000 | 0000 |
| XOR $2, $2 , $2 | 0926 | 3 | 0000 | 0000 | 0026 | 0000 | 0000 | 0000 |
| XOR $3, $3 , $3 | 0db6 | 4 | 0000 | 0000 | 0036 | 0000 | 0000 | 0000 |
| XOR $4, $4 , $4 | 1246 | 5 | 0000 | 0000 | 0046 | 0000 | 0000 | 0000 |
| ADD $0, $0, $5 | 1400 | 6 | 0005 | 0000 | 0000 | 0005 | 0000 | 0005 |
| ADDI $3, $3, 2 | 2d82 | 7 | 0000 | 0000 | 0002 | 0002 | 0000 | 0002 |
| ADD $2, $2, $3 | 0d20 | 8 | 0002 | 0000 | 0020 | 0002 | 0000 | 0002 |
| ADDI $4, $4, 1 | 3201 | 9 | 0000 | 0000 | 0001 | 0001 | 0001 | 0001 |
| BEQ $0, $4, 1 | 9001 | 10 | 0001 | 0005 | 0001 | FFFC | 0000 | FFFC |
| J 6 | E006 | 7 | 0005 | 0005 | 0006 | 0000 | 0000 | 0000 |
| SW $2, 0($1) | 6500 | 12 | 0000 | 001E | 0000 | 0000 | 0000 | 0000 |
| LW $2, 0($1) | 4500 | 13 | 0000 | 001E | 0000 | 0000 | 001E | 001E |

e. Nu exista parti din processor incomplete din laboratoarele 4-7.

1. **Rezolvarea hazardurilor folosind diagrama pipeline**

Programul original:

|  |  |
| --- | --- |
| **Instr. Nr.** | **Program** |
| 0 | xor $0, $0, $0 |
| 1 | xor $1, $1, $1 |
| 2 | xor **$2**, $2, $2 |
| 3 | xor $3, $3, $3 |
| 4 | xor $4, $4, $4 |
| 5 | add $0, $0, $5 |
| 6 | addi $3, $3, 2 |
| 7 | add $2, $2, $3 |
| 8 | addi $4, $4, 1 |
| 9 | beq $0, $4, 1 |
| 10 | j6 |
| 11 | sw $2, 0($1) |
| 12 | lw $2, 0($1) |

Identificarea hazardurilor:

Între **instrucțiunile 6 și 7** se identifică un **hazard de date** de tipul **Read After Write** (**RAW**) după **registrul $3**, acesta putând fi soluționat prin introducerea a două NoOp-uri între cele două instrucțiuni.

Între **instrucțiunile 8 și 9** se identifică, de asemenea, un **hazard de date** după **registrul $4**, acesta fiind rezolvat în același mod cu cel precedent.

Poate fi identificat câte un **hazard de control** în cazul **instrucțiunilor 9 și 10**. În ceea ce privește instrucțiunea 9, instrucțiuni de **salt condiționat**, hazardul este soluționat prin introducerea a trei operații de tip NoOp după instrucțiunea în cadrul căreia a fost regăsit hazardul, pentru instrucțiunea de **salt necondiționat** 10 fiind suficientă inserarea unei singure operații NoOp pentru a soluționa hazardul de control.

**Hazardul structural** a fost soluționat prin modificarea blocului de registre RF astfel încât scrierea să aibă loc în mijlocul perioadei de ceas (pe frontul descrescător al semnalului de ceas).

|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| **Instr** | CC1 | CC2 | CC3 | CC4 | CC5 | CC6 | CC7 | CC8 | CC9 | CC10 | CC11 | CC12 | CC13 | CC14 | CC15 | CC16 | CC17 | CC18 | CC19 | CC20 | CC21 | CC22 | CC23 | CC24 |
| xor $0,$0,$0 | IF | ID | EX | MM | WB |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| xor $1,$1,$1 |  | IF | ID | EX | MM | WB |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| xor $2,$2,$2 |  |  | IF | ID | EX | mm | WB |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| xor $3,$3,$3 |  |  |  | IF | ID | EX | mm | WB |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| xor $4,$4,$4 |  |  |  |  | IF | ID | EX | mm | WB |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| add $0,$0,$5 |  |  |  |  |  | IF | ID | EX | mm | WB |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| addi $3,$3,2 |  |  |  |  |  |  | IF | ID | EX | MM | WB |  |  |  |  |  |  |  |  |  |  |  |  |  |
| NoOp |  |  |  |  |  |  |  | IF | ID | EX | MM | WB |  |  |  |  |  |  |  |  |  |  |  |  |
| NoOp |  |  |  |  |  |  |  |  | IF | ID | EX | MM | WB |  |  |  |  |  |  |  |  |  |  |  |
| add $2,$2,$3 |  |  |  |  |  |  |  |  |  | IF | ID | EX | MM | WB |  |  |  |  |  |  |  |  |  |  |
| addi $4,$4,1 |  |  |  |  |  |  |  |  |  |  | IF | ID | EX | MM | WB |  |  |  |  |  |  |  |  |  |
| NoOp |  |  |  |  |  |  |  |  |  |  |  | IF | ID | EX | MM | WB |  |  |  |  |  |  |  |  |
| NoOp |  |  |  |  |  |  |  |  |  |  |  |  | IF | ID | EX | MM | WB |  |  |  |  |  |  |  |
| beq $0,$4,1 |  |  |  |  |  |  |  |  |  |  |  |  |  | IF | ID | EX | MM | WB |  |  |  |  |  |  |
| NoOp |  |  |  |  |  |  |  |  |  |  |  |  |  |  | IF | ID | EX | MM | WB |  |  |  |  |  |
| NoOp |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  | IF | ID | EX | MM | WB |  |  |  |  |
| NoOp |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  | IF | ID | EX | MM | WB |  |  |  |
| j6 |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  | IF | ID | EX | MM | WB |  |  |
| NoOp |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  | IF | ID | EX | MM | WB |  |
| sw $2,0($1) |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  | IF | ID | EX | MM | WB |
| lw $2,0($1) |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  | IF | ID | EX | MM |

f. Totul este descries in vhdl si nu exista errori. RTL schematic

A picture containing text, antenna, screenshot

Description automatically generated

![A picture containing text, whiteboard

Description automatically generated]()

g. Programul a fost testat pe placa si este complet functional