DOCUMENTATIE

MIPS

NUME STUDENT: Pântea Marius-Nicușor

GRUPA:30222

**Elemente Funcționale:**

Monopulse Generator (MPG):

* Generează semnale de control pentru afișarea pe un SSD.
* Verificări necesare: Corectitudinea generării semnalelor de control în funcție de intrările furnizate.

Display pe SSD (SEGMENTE):

* Implementează afișarea pe un display cu șapte segmente.
* Verificări necesare: Asigurarea că datele de afișare sunt corect prelucrate și transmise către display.

Unitate de Fetching a Instrucțiunilor (IFetch):

* Recuperează instrucțiuni din memorie și le furnizează către etapa de decodare.
* Verificări necesare: Asigurarea că instrucțiunile sunt citite corect și furnizate la ieșire în timpul ciclului de ceas.

Decodificator de Instrucțiuni (ID):

* Decodează instrucțiunile pentru a determina operația de execuție și operanzii implicați.
* Verificări necesare: Corectitudinea decodificării instrucțiunilor și identificarea corectă a operanzilor.

Unitate de Control (UC):

* Controlează operațiile efectuate de procesor în funcție de instrucțiunea curentă.
* Verificări necesare: Asigurarea că semnalele de control sunt corect generate în funcție de tipul de instrucțiune.

Unitate de Execuție (EX):

* Realizează operații aritmetice și logice pe baza instrucțiunilor primite.
* Verificări necesare: Corectitudinea operațiilor efectuate și a semnalelor de ieșire generate.

Unitate de Memorie (MEM):

* Accesează memoria pentru citirea și/sau scrierea datelor.
* Verificări necesare: Asigurarea corectitudinii accesului la memorie și a operațiilor de citire/scriere.

Registre de Memorare (Reg\_File):

* Implementează un fișier de registre pentru stocarea datelor.
* Verificări necesare: Asigurarea corectitudinii scrierii și citirii datelor în/din registre.

ROM (Read-Only Memory):

* Implementează o memorie program care stochează instrucțiuni.
* Verificări necesare: Asigurarea corectitudinii citirii instrucțiunilor din memorie.

test\_env:

* Mediu de test care conectează toate modulele.
* Verificări necesare: Asigurarea că toate modulele sunt conectate și funcționează împreună corect.

Precizari:

* Toate elementele componente au fost testate pe placa.

Probleme întâmpinate

* Erori în interconectarea modulelor sau a semnalelor
* Greșeli în logica de proiectare sau în implementarea modulelor
* Necorespunderea semnalelor între module
* Probleme în accesarea sau furnizarea instrucțiunilor
* Decodificarea incorectă a instrucțiunilor poate genera operații greșite sau poate afecta operațiile viitoare ale procesorului.
* Operații greșite sau semnale de ieșire incorecte
* Accesarea incorectă a memoriei
* Erori în procesul de scriere sau citire a datelor în/din registre
* Probleme în accesarea sau citirea corectă a instrucțiunilor din memorie ROM

**Cele 4 instrucțiuni adăugate**

XOR – bitwise eXclusive-OR

* SAU-Exclusiv logic între două registre, memorează rezultatul în alt registru
* $d 🡨 $s ^ $t; PC 🡨 PC + 4;
* xor $d, $s, $t
* 000000 sssss ttttt ddddd 00000 100110

SLT – Set on Less Than (signed)

* Dacă $s < $t, $d este inițializat cu 1, altfel cu 0
* PC🡨PC + 4; if $s < $t then $d 🡨 1 else $d 🡨 0;
* slt $d, $s, $t
* 000000 sssss ttttt ddddd 00000 101010

ORI – bitwise OR Immediate

* SAU logic între un registru și o valoare imediată, memorează rezultatul
* în alt registru
* $t 🡨$s | ZE(imm); PC 🡨PC + 4;
* ori $t, $s, imm
* 001101 sssss ttttt iiiiiiiiiiiiiiii

BGTZ – Branch on Greater Than Zero

* Descriere Salt condiționat dacă un registru este mai mare ca 0
* If $s > 0 then PC  PC + 4 + (SE(offset) << 2) else PC  PC + 4;
* bgtz $s, offset
* 000111 sssss 00000 oooooooooooooooo

Cele 4 instrucțiuni , nu au fost utilizate in programul executat de procesor , dar pentru o extindere a funcționalități pot fi foarte utile .

**Semnale de control MIPS32**

Semnale de Branch opționale: ? ϵ {gez, ne, gtz}, se va înlocui ? cu o valoare din paranteză, dacă e cazul

Tipuri de operații care se pun în paranteză la ALUOp si ALUCtrl:

(+), (-), (&), (|), (^), (<<*l*), (<<*lv*), (>>*l*), (>>*a*), (<)

Semnificații: & - AND, | - OR, ^ *- XOR, l* *- logic, a - aritmetic, v - cu variabilă*

|  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| **Instrucțiune** | **Opcode**  *Instr[31-26]* | **RegDst** | **ExtOp** | **ALUSrc** | **Branch** | **Br\_GTZ** | **Jump** | **MemWrite** | **MemtoReg** | **Reg Write** | **ALUOp[5:0]** | **function**  *Instr[5-0]* | **ALUCtrl[2:0]** |
| ADD | 000000 | 1 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 1 | 000000 | 100000 | 100000 |
| SUB | 000000 | 1 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 1 | 000000 | 100010 | 100010 |
| SLL | 000000 | 1 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 1 | 000000 | 000000 | 000000 |
| SRL | 000000 | 1 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 1 | 000000 | 000000 | 000000 |
| AND | 000000 | 1 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 1 | 000000 | 100100 | 100100 |
| OR | 000000 | 1 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 1 | 000000 | 100101 | 100101 |
| XOR | 000000 | 1 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 1 | 000000 | 100110 | 100110 |
| SLT | 000000 | 1 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 1 | 000000 | 101010 | 101010 |
| ADDI | 001000 | 0 | 1 | 1 | 0 | 0 | 0 | 0 | 0 | 1 | 000001 | X | 000001 |
| ORI | 001101 | 0 | 1 | 1 | 0 | 0 | 0 | 0 | 0 | 1 | 000010 | X | 000010 |
| LW | 100011 | 0 | 1 | 1 | 0 | 0 | 0 | 0 | 1 | 1 | 000001 | X | 000001 |
| SW | 101011 | 0 | 1 | 1 | 0 | 0 | 0 | 1 | 0 | 0 | 000001 | X | 000001 |
| BEQ | 000100 | 0 | 1 | 0 | 1 | 0 | 0 | 0 | 0 | 0 | 000100 | X | 000100 |
| BGTZ | 000001 | 0 | 1 | 0 | 0 | 1 | 0 | 0 | 0 | 0 | 000100 | X | 000100 |
| Jump | 000010 | 0 | 0 | 0 | 0 | 0 | 1 | 0 | 0 | 0 | 001000 | X | 001000 |
|  |  |  |  |  |  |  |  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |  |  |  |  |  |  |  |

URL: <https://drive.google.com/file/d/1SI7x2Gp_2m3SEkwnXuGt4ns4voYzpGBH/view?usp=sharing>

**Trasarea execuției programului de test pentru MIPS32**

Valorile se completează în hexazecimal așa cum trebuie să apară pe SSD. Succesiunea pașilor reprezintă ordinea de execuție în timp la apăsarea butonului ENable. **Pasul 0 corespunde stării inițiale a circuitului (PC = 0), iar** **pasul *N* caracterizează starea după apăsarea de *N* ori a butonului ENable**. Inițial registrele vor avea valoarea 0 (care se atribuie automat în lipsa unei inițializări explicite a RF), iar memoria de date RAM poate fi inițializată cu valori dorite. Tabelul se completează pentru tot programul sau, dacă are buclă, până la finalul primei iterații. *Buclă = revenirea execuției la o instrucțiune care a mai fost executată anterior.*

|  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| **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+4** | **RD1** | **RD2** | **Ext\_Imm** | **ALURes** | **MemData** | **WD** | **BranchAddr** | **JumpAddr** |
| 0 | xor $r10, $r10, $r10 | X"014A5026" | X"00000001" | X"0000010" | X"0000010" | X"" | X"0000000" | X"0000008" | X"0000000" | X"0000000" | X"0000000" |
| 1 | xor $r0, $r0, $r0 | X"00000026" | X"00000002" | X"0000000" | X"0000000" | X"" | X"0000000" | X"0000008" | X"0000000" |  |  |
| 2 | xor $r30, $r30, $r30 | X"03DEF026" | X"00000003" | X"0000030" | X"0000030" | X"" | X"0000000" | X"0000008" | X"0000000" |  |  |
| 3 | xor $r3, $r3, $r3 | X"00631826" | X"00000004 | X"0000003" | X"0000003" | X"" | X"0000000" | X"0000008" | X"0000000" |  |  |
| 4 | addi $r3, $r3, 0xFFFF | X"2063FFFF" | X"00000005" | X"0000000" | X"0000000" | X"0000FFFF" | X"0000FFFF" | X"0000000" | X"0000FFFF" |  |  |
| 5 | addi $r30,$r30,4 | X"23DE0004" | X"00000006" | X"0000000" | X"0000000" | X"0000004" | X"0000004" | X"0000000" | X"0000004" |  |  |
| 6 | lw $r1, 0($r10) | X"8D410000" | X"00000007" | X"0000000" | X"0000001" | X"0000000" | X"000008" | X"0000008" | X"0000008" |  |  |
| 7 | addi $r0, $r0, 8 | X"20000008" | X"00000008" | X"0000000" | X"0000000" | X"0000008" | X"0000008" | X"0000000" | X"0000008" |  |  |
| 8 | addi $r11, $r1, 32 | X"202B0020" | X"00000009" | X"0000000" | X"0000008" | X"000001F" | X"0000028" | X"0000000" | X"0000028" |  |  |
| 9 | beq $r1, $r11, offset 5 | X"102B0005" | X"0000000A" | X"0000008" | X"0000028" | X"0000005" | X"0000008" | X"0000028" | X"0000008" | X"000000F" |  |
| 10 | lw $r2, 0($r1) | X"8C220000" | X"0000000B" | X"0000000" | X"0000009" | X"0000000" | X"0000008" | X"0000004" | X"0000004" |  |  |
| 11 | and $r2, $r2, $r3 | X"00431024" | X"0000000C" | X"00000004" | X"0000FFFF" | X"" | X"0000000" | X"0000000" | X"0000004" |  |  |
| 12 | add $r0, $r0, $r2 | X"00020020" | X"0000000D" | X"00000000" | X"00000004" | X"" | X0000000"" | X"0000000" | X"0000004" |  |  |
| 13 | addi $r1, $r1, 4 | X"20210004" | X"0000000E" | X"0000008" | X"0000008" | X"0000004" | X"000000C" | X"0000002" | X"000000C" |  |  |
| 14 | j offset 9 | X"08000009" | X"0000000F" | X"0000000" | X"0000000" | X"0000000" | X"0000009" | X"0000002" | X"0000000" |  | X"0000009" |
| 15 | sw $r0, offset($r30) | X"AFC00000" | X"00000010" | X"0000060" | X"0000004" | X"0000000" | X"0000060" | X"0000000" | X"0000060" |  |  |
| 16 |  |  |  |  |  |  |  |  |  |  |  |
| 17 |  |  |  |  |  |  |  |  |  |  |  |
| 18 |  |  |  |  |  |  |  |  |  |  |  |
| 19 |  |  |  |  |  |  |  |  |  |  |  |
| 20 |  |  |  |  |  |  |  |  |  |  |  |
| 21 |  |  |  |  |  |  |  |  |  |  |  |
| 22 |  |  |  |  |  |  |  |  |  |  |  |
| 24 |  |  |  |  |  |  |  |  |  |  |  |

URL: <https://drive.google.com/file/d/1OgoST1-tEe1cbUdNk_VKr6NHq3zVfs83/view?usp=sharing>