DOCUMENTATIE

MIPS16 CICLU UNIC

Logo

Description automatically generated

NUME STUDENT: VOICU LAURA-LUISA

GRUPA: 30226

***CUPRINS***

1. **Instructiuni Suplimentare**
2. **Tabel cu valorile semnalelor de control**
3. **Descrierea codului**
4. **Trasarea Executiei programului**
5. **Precizare activitati incomplete din laboratoarele 4-7**
6. **RTL schematic**
7. **Testare placuta**

***Instructiuni suplimentare alese***

**🡪 Tip R**

* **Exclusive OR (XOR)**
  + Generic: **xor $rd, $rs, $rt**
  + Operatie de baza: **RF[rd] 🡨 RF[rs] XOR RF[rt]**
  + PC la urmatoarea instructiune: **PC🡨 PC + 1**
  + OPCODE: **000**
  + Func: **110**
  + Reprezentare:

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **OPCODE** | **RS** | **RT** | **RD** | **SA** | **FUNC** |
| **000** |  |  |  |  | **110** |
| **15 14 13** | **12 11 10** | **9 8 7** | **6 5 4** | **3** | **2 1 0** |

* **Shift Right Arithmetic (SRA)**
  + Generic: sra $rd, $rs,sa
  + Operatie de baza: **RF🡨RF[rs] XOR RF[rt]**
  + PC la urmatoarea instructiune: **PC 🡨 PC +1**
  + OPCODE: **110**
  + Func: **111**

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **OPCODE** | **RS** | **RT** | **RD** | **SA** | **FUNC** |
| **000** |  |  |  |  | **111** |
| **15 14 13** | **12 11 10** | **9 8 7** | **6 5 4** | **3** | **2 1 0** |

**🡪 Tip I**

* **Branch if not equal (BNE)**
  + Adresare: Relativa la PC
  + RTL Abstract:
    - **if(RF[rs] == RF[rt]) then**
    - **PC 🡨 PC + 4 + S\_EXT(imm)**
    - **else**
    - **PC 🡨 PC + 1**

|  |  |  |  |
| --- | --- | --- | --- |
| **OPCODE** | **RS** | **RT** | **IMMEDIATE** |
| **101** |  |  |  |
| **15 14 13** | **12 11 10** | **9 8 7** | **6 5 4 3 2 1 0** |

* + Resurse necesare:
    - **[IF]** PC,Memorie de instuctiuni,sumator
    - **[ID]** Bloc de registre, Extensie, UC
    - **[EX]** ALU, UC Alu 🡪 sumator,circuit de deplasare, MUX

**Semnale de constol MIPS16 ciclu unic**

|  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| Instr | OPCODE | RegDst | ExtOp | | ALUSRC | Branch | Jump | MemWrite | MemToReg | RegWrite | ALUOp | Func | ALUCtrl |
| Tip R | | | |
| ADD | 000 | 1 | X | | 0 | 0 | 0 | 0 | 0 | 1 | 000 | 000 | 0000 |
| SUB | 000 | 1 | X | | 0 | 0 | 0 | 0 | 0 | 1 | 001 | 001 | 0001 |
| SLL | 000 | X | X | | X | 0 | 0 | 0 | 0 | 1 | 010 | 010 | 0010 |
| SRL | 000 | X | X | | X | 0 | 0 | 0 | 0 | 1 | 011 | 011 | 0011 |
| AND | 000 | 1 | X | | 0 | 0 | 0 | 0 | 0 | 1 | 100 | 100 | 0100 |
| OR | 000 | 1 | X | | 0 | 0 | 0 | 0 | 0 | 1 | 101 | 101 | 0101 |
| XOR | 000 | 1 | X | | 0 | 0 | 0 | 0 | 0 | 1 | 110 | 110 | 0110 |
| SRA | 000 | 1 | X | | 0 | 0 | 0 | 0 | 0 | 1 | 111 | 111 | 0111 |
| Tip I | | | |
| ADDI | 001 | 0 | 1 | | 1 | 0 | 0 | 0 | 0 | 1 | 001 | X | 0000 |
| LW | 010 | 0 | 1 | | 1 | 0 | 0 | 0 | 1 | 1 | 010 | X | 0000 |
| SW | 011 | X | 1 | | 1 | 0 | 0 | 1 | X | 0 | 011 | X | 0000 |
| BEQ | 100 | X | 1 | | 0 | 1 | 0 | 0 | X | 0 | 100 | X | 0001 |
| BNE | 101 | 0 | 1 | | 0 | 0 | 0 | 0 | X | 0 | 101 | X | 0001 |
| SLTI | 110 | 0 | 0 | | 1 | 0 | 0 | 0 | X | 0 | 110 | x | 1000 |
| Tip J | | | |
| J | 111 | X | X | | x | x | 1 | 0 | x | 0 | 111 | 0 | 1111 |

***Descriere cod***

* **Este implementat un algoritm ce foloseste un loop pentru a calcula suma elementelor de pe indicii pari si cea de pe indicii impari, la final verificand daca acestea sunt egale**
* b"001\_000\_001\_0000001", -- 2081 0. addi $1,$0,1 --i=1

int i = 1;

int j = 5;

int sum1=0;

int sum2=0;

int n = (i+j)/2; -- nu e folosit la nimic

for(i = i ; i < j; i++)

{

if(i&”0001”==0)

{

sum2+=v[i];

}

else

{

sum3+=v[i];

} }

If(sum1==sum2)

{

result = 1 ;

}

else

{

result=0;

}

* b"001\_000\_010\_0000101" ,-- 2105 1. addi $2,$0,5 --j=5 -- la 5 se opreste ( functioneaza ca while)
* b"001\_000\_011\_0000000", -- 2180 2. addi $3,$0,0 --sum1=0
* b"001\_000\_100\_0000000", -- 2200 3. addi $4,$0,0 --sum2=0
* b"000\_010\_001\_101\_0\_000", -- 08D0 4. add $5,$2,$1 --n = i+j
* b"000\_000\_101\_101\_1\_011", -- 02DB 5. srl $5,$5,1 --n/2 -- marchez jumatatea vectorului
* b"100\_010\_001\_0001100", -- 8888 6. loop\_st: beq $1,$2,loop\_end = 8 -- deschide bucla
* b"001\_000\_110\_0000001",-- 2301 13. addi $6,$0,1 -- $6=0001
* b"000\_110\_001\_111\_0\_100", -- 18F4 4. and $7,$6,$1 $7 = 1 daca e i impar, 0 altfel
* b"100\_111\_000\_0000100", -- 9C82 9. beq $7,$0,par=2 -- verifica daca i par
* b"010\_001\_101\_0000000", --4780 lw $5, $1
* b"000\_101\_000\_101\_0\_000", -- 1C70 4. add $5,$5,$0 --n = i+j
* b"000\_011\_101\_011\_0\_000", -- eb0 12. impar: add $3,$3,$5
* b"111\_0000000010001", -- E011 14. jmp incr = 17
* b"010\_001\_101\_0000000", --4780 lw $5, $1
* b"000\_101\_000\_101\_0\_000", -- 1C70 4. add $5,$5,$0 --n = i+j
* b"000\_100\_101\_100\_0\_000", -- 12C0 12. par: add $4,$4,$5
* b"001\_001\_001\_0000001",-- 2481 13. incr: addi $1,$1,1
* b"111\_0000000000110", -- E006 14. jmp 6 --loop\_st
* b"100\_011\_100\_0000010", -- 8E01 15. loop\_end: beq $3,$4, equal
* b"001\_000\_001\_0000000",-- 2080 13. addi $1,$0,0 -- $6=0001
* b"111\_0000000010111", -- E017 14. jmp done
* b"001\_000\_001\_0000001",-- 2081 13. addi $1,$0,1 -- $6=0001
* b"000\_001\_000\_001\_0\_000", -- 410 4. --done add $1,$1,$0 --n = i+j
* b"100\_001\_001\_0000100", --8484 sw $1, $1

**Tip I : op $rt, $rs, imm op $rd,$rs,$rt**

Diagram

Description automatically generatedChart

Description automatically generated

***Testarea executiei***

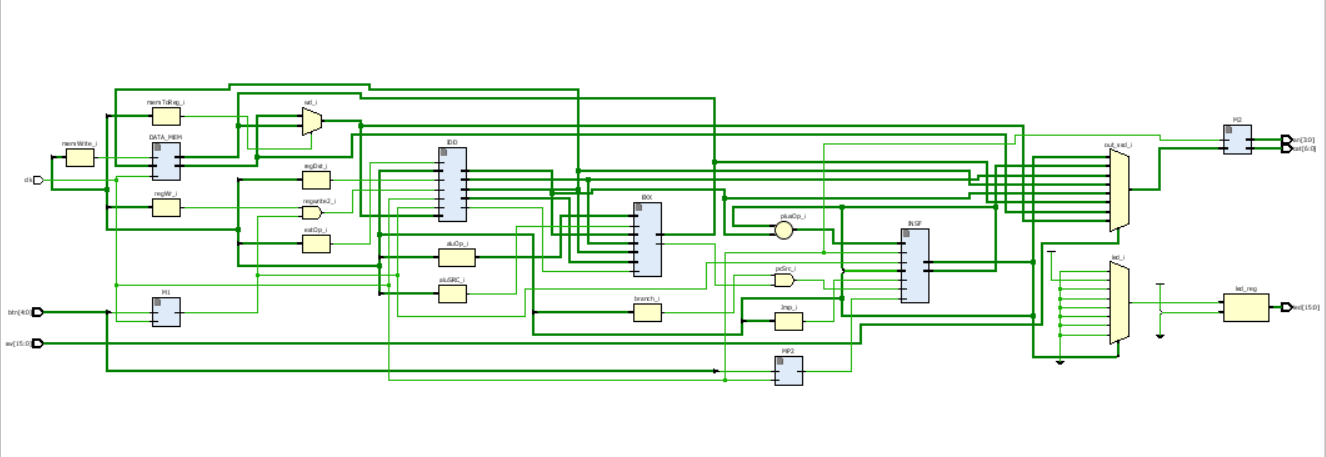
|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| **Instructiune** | **Instr** | **AluRes** | **RD1** | **RD2** | **Ext\_imm** | **PC** | **WD** |
| **addi $1,$0,1** | **2180** | **1** | **0** | **0** | **1** | **1** | **1** |
| **addi $2,$0,5 --j=5** | **2105** | **5** | **0** | **0** | **5** | **2** | **5** |
| **addi $3,$0,0** | **2180** | **0** | **0** | **0** | **0** | **3** | **0** |
| **addi $4,$0,0** | **2200** | **0** | **0** | **0** | **0** | **4** | **0** |
| **add $5,$2,$1** | **08D0** | **6** | **5** | **1** | **50** | **5** | **6** |
| **srl $5,$5,1** | **02DB** | **3** | **0** | **6** | **5b** | **6** | **3** |
| **beq $1,$2,loop\_end = 8** | **888C** | **4** | **5** | **1** | **c** | **7** | **4** |
| **addi $6,$0,1** | **2301** | **1** | **0** | **0** | **1** | **8** | **1** |
| **and $7,$6,$1** | **18F4** | **1** | **1** | **1** | **74** | **9** | **1** |
| **beq $7,$0,par=2** | **9C04** | **1** | **1** | **0** | **4** | **a** | **1** |
| **lw $5, $1** | **4680** | **1** | **1** | **3** | **0** | **b** | **4** |
| **add $5,$5,$0** | **1C70** | **4** | **4** | **0** | **50** | **C** | **4** |
| **impar: add $3,$3,$5** | **0eb0** | **3** | **0** | **4** | **30** | **d** | **4** |
| **jmp incr = 17** | **E011** | **0** | **0** | **0** | **11** | **e** | **0** |
| **lw $5, $1** | **4680** | **2** | **2** | **4** | **0** | **f** | **5** |
| **add $5,$5,$0** | **1450** | **5** | **5** | **0** | **50** | **10** | **5** |
| **par: add $4,$4,$5** | **12C0** | **5** | **0** | **5** | **40** | **11** | **5** |
| **incr: addi $1,$1,1** | **2481** | **2** | **1** | **1** | **1** | **12** | **1** |
| **jmp 6 --loop\_st** | **E006** | **0** | **0** | **0** | **6** | **13** | **0** |
| **loop\_end: beq $3,$4, equal** | **8E01** | **3** | **a** | **7** | **2** | **14** | **3** |
| **addi $1,$0,0** | **2080** | **0** | **0** | **5** | **0** | **15** | **0** |
| **jmp done** | **E017** | **0** | **0** | **0** | **17** | **16** | **0** |
| **addi $1,$0,1** | **2081** | **0** | **0** | **0** | **10** | **17** | **0** |
| **done add $1,$1,$0** | **0410** | **0** | **0** | **0** | **18** | **18** | **0** |
| **sw $1, $1** | **8484** | **0** | **0** | **0** | **4** | **19** | **0** |
|  |  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |  |

**e. Parti incomplete din laboratoarele 4-7**

Nu exista parti incomplete din laboratoarele 4-7

**f. Corectitudinea descrierii vhdl – RTL schematic**

Nu exista erori.



**g. Testare pe placa**

Programul a fost testat pe placuta si functioneaza.

1. **b"001\_000\_001\_0000001", -- 2081 0. addi $1,$0,1 --i=1**
2. **b"001\_000\_010\_0000100" ,-- 2104 1. addi $2,$0,4 --j=4**
3. **b"001\_000\_011\_0000000", -- 2180 2. addi $3,$0,0 --sum1=0**
4. **b"001\_000\_100\_0000000", -- 2200 3. addi $4,$0,0 --sum2=0**
5. **b"000\_010\_001\_101\_0\_000", -- 0CD0 4. add $5,$2,$1 --n = i+j**
6. **b"000\_000\_101\_101\_1\_011", -- 02DB 5. srl $5,$5,1 --n/2 -- marchez jumatatea vectorului**
7. **b"100\_010\_001\_0001000", -- 8888 6. loop\_st: beq $1,$2,loop\_end = 8 -- deschide bucla**
8. **b"000\_001\_101\_110\_0\_001", -- 06E1 7. sub $6,$5,$1 -- am trecut pe prima jumatate?**
9. **b"110\_110\_111\_0000000", -- DB80 8. slti $7,$6,0 -- daca da, retin in &7 = 1**
10. **b"100\_000\_111\_0000000", -- 8380 9. beq $7,$0,mare=2**
11. **b"000\_011\_001\_011\_0\_000", -- 0CB0 10. add $3,$3,$1**
12. **b"111\_0000000000110", -- E006 14. jmp 6 --loop\_st**
13. **b"000\_100\_001\_100\_0\_000", -- 10C0 12. mare: add $4,$4,$1**
14. **b"001\_001\_001\_0000001",-- 2481 13. incr: addi $1,$1,1**
15. **b"111\_0000000000110", -- E006 14. jmp 6 --loop\_st**
16. **b"000\_011\_100\_001\_0\_001", -- 0E11 15. loop\_end: sub $1,$3,$4**
17. **b"001\_000\_111\_0000001", --2381 16. 18. addi $7,$0,1**

**others =>x"1110000000000011"**

**🡪 video ce demonstreaza functionalitatea programului ( watermark-ul de jos e pus automat de o aplicatie ce inverseaza un video – am filmat invers 😊 )**

****