Logo

Description automatically generated

**Procesorul MIPS Pipeline- 16 biti**

**Student: Miruna Muresan**

**Grupa:302210**

1. Nu au fost facute schimbari la cele patru instructiuni alese pentru Mips pipeline pe 16 biti, descrierea acestora ramanand la fel ca la Mips ciclu unic pe 16 biti.

b)

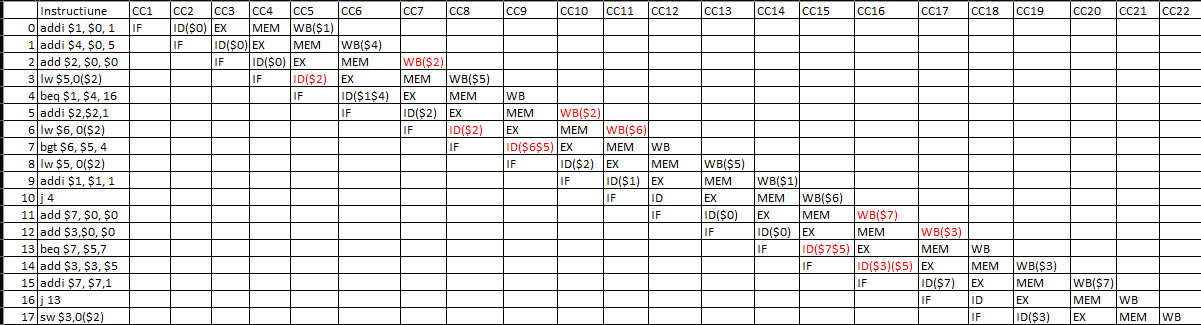
|  |  |  |  |
| --- | --- | --- | --- |
| **REG\_IF\_ID(31** – **0)** | **REG\_ID\_EX(<msb>** – **0>)** | **REG\_EX\_MEM(<msb>** – **0)** | **REG\_MEM\_WB(<msb>** – **0)** |
| Instruction (IF\_ID(15 DOWNTO 0)) | RegDst (ID\_EX(0)) | REGWRITE (EX\_MEM(0)) | REGWRITE (MEM\_WB(0)) |
| PC PLUSUNU(IF\_ID(31 DOWNTO 16)) | AluSrc (Id\_EX(1)) | MEMWRITE( EX\_MEM(1)) | MEMREG (MEM\_WB(1)) |
|  | ALUOP (ID\_EX(3 DOWNTO 2)) | MEMREG (EX\_MEM(2)) | WA (MEM\_WB(4 DOWNTO 2)) |
|  | ALUOP (ID\_EX(5 DOWNTO 4)) | BRANCH(EX\_MEM(3)) | ALURES (MEM\_WB(20 DOWNTO 5)) |
|  | BRANCH(ID\_EX(4)) | BGT (EX\_MEM(4)) | MEMDATA (MEM\_WB(36 DOWNTO 21) |
|  | BGT(ID\_EX(5)) | ZERO (EX\_MEM(5)) |  |
|  | REGWRITE (ID\_EX(6)) | ZERO BRANCH BGT(EX\_MEM(6)) |  |
|  | MEMWRITE(ID\_EX(7)) | ZEROBRANCHBGT (EX\_MEM(57)) |  |
|  | MEMREG(ID\_EX(8)) | WA( EX\_MEM(9 DOWNTO 7)) |  |
|  | RT (ID\_EX(11 DOWNTO 9)) | RDATA2 (EX\_MEM((25 DOWNT0 10)) |  |
|  | RTD (ID\_EX(14 DOWNTO 12)) | ALURES (EX\_MEM(41 DOWNTO 26)) |  |
|  | FUNC (ID\_EX(18 DOWNTO 16)) | BRANCHADDRESS (EX\_MEM (57 DOWNTO 42)) |  |
|  | SA (ID\_EX(15)) |  |  |
|  | EXT\_IMM(ID\_EX(34 DOWNTO 19)) |  |  |
|  | RDATA1(ID\_EX(50 DOWNTO 35)) |  |  |
|  | RDATA2(ID\_EX(66 DOWNTO 51)) |  |  |
|  | PC PLUS UNU (ID\_EX(82 DOWNTO 67)) |  |  |
|  |  |  |  |

c)

Diagram

Description automatically generated

d)



Programul original

0. addi $1, $0, 1 #i=0, contorul buclei

1. addi $4, $0, 5 #se salveaza numarul maxim de iteratii

2. add $2, $0, $0 #initializarea indexului locatiei de memorie

3. lw $5,0($2) #min=arr[0]

4. beq $1, $4, 6 #s-au facut 10 iteratii? Daca da, sari la

5. addi $2,$2,1 #creste indexul

6. lw $6, 0($2) #salveaza in $6 urmatorul elem

7. bgt $6, $5, 1 #arr[i]>min? Daca da, sari

8. lw $5, 0($2) #daca nu, salveaza in $5 arr[i]

9. addi $1, $1, 1 #creste contorul

10. j 4 #jump

11. add $7, $0, $0 #pune in $7 0

12. add $3,$0, $0 #pune in $3 valoarea 0

13. beq $7, $5,3 #este $7=$5? daca da sari

14. add $3, $3, $5 #daca nu adauga in $3 valoarea lui $5

15. addi $7, $$7,1 #creste contorul

16. j 13 #jump

17. sw $3,0($2) #salveaza minimul la patrat pe ultimul element

Identificare hazardurilor

* Între instrucțiunile 2 și 3 se identifică un hazard de date de tipul Read After Write (RAW) după registrul $2, acesta putând fi soluționat prin introducerea a două NoOp-uri între cele două instrucțiuni.
* Instructiunile 4, 7 si 13 sunt de salt conditionat, astfel apare un hazard de control. Dupa fiecare instructiune de salt conditionat, am adaugat 3 NoOp-uri. Instructiunile 10 si 16 sunt de jump, la finalul fiecaruia am adaugat un singur NoOp.
* Între instrucțiunile 5 și 6 se identifică un hazard de date de tipul Read After Write (RAW) după registrul $2, acesta putând fi soluționat prin introducerea a două NoOp-uri între cele două instrucțiuni.
* Între instrucțiunile 6 și 7 se identifică un hazard de date de tipul Read After Write (RAW) după registrul $6, acesta putând fi soluționat prin introducerea a două NoOp-uri între cele două instrucțiuni.
* Între instrucțiunile 11 și 13 se identifică un hazard de date de tipul Read After Write (RAW) după registrul $7, acesta putând fi soluționat prin introducerea unui singur NoOp între cele două instrucțiuni.
* Între instrucțiunile 12 și 14 se identifică un hazard de date de tipul Read After Write (RAW) după registrul $3, acesta putând fi soluționat prin introducerea unui singur NoOp între cele două instrucțiuni.

Programul cu NoOp-uri

0. addi $1, $0, 1 #i=0, contorul buclei

1. addi $4, $0, 5 #se salveaza numarul maxim de iteratii

2. add $2, $0, $0 #initializarea indexului locatiei de memorie

3. add $0, $0, $0

4. add $0, $0, $0

5. lw $5,0($2) #min=arr[0]

6. beq $1, $4, 14 #s-au facut 5 iteratii? Daca da, sari la

7. add $0, $0, $0

8. add $0, $0, $0

9. add $0, $0, $0

10. addi $2,$2,1 #creste indexul

11. add $0, $0, $0

12. add $0, $0, $0

13. lw $6, 0($2) #salveaza in $6 urmatorul elem

14. add $0, $0, $0

15. add $0, $0, $0

16. bgt $6, $5, 4 #arr[i]>min? Daca da, sari

17. add $0, $0, $0

18. add $0, $0, $0

19. add $0, $0, $0

20. lw $5, 0($2) #daca nu, salveaza in $5 arr[i]

21. addi $1, $1, 1 #creste contorul

22. j 6 #jump

23. add $0, $0, $0

23. add $7, $0, $0 #pune in $7 0

24. add $0, $0, $0

25. add $3,$0, $0 #pune in $3 valoarea 0

26. add $0, $0, $0

27. beq $7, $5,6 #este $7=$5? daca da sari

28. add $0, $0, $0

29. add $0, $0, $0

30. add $0, $0, $0

31. add $3, $3, $5 #daca nu adauga in $3 valoarea lui $5

32. addi $7, $7,1 #creste contorul

33. j 25 #jump

34. add $0, $0, $0

35. sw $3,0($2) #salveaza minimul la patrat pe ultimul element

e) Nu exsita componente incomplete

f) Fisierul bitstream a fost generat cu success, totul a fost implementat in vhdl.

Diagram, schematic

Description automatically generated

Diagram, schematic

Description automatically generated

Diagram, schematic

Description automatically generated

g) Programul a fost testat pe placa, are mici defecte de functionare