**Procesor MIPS pipeline 16bit (Raport)**

Stef Paula-Elena gr. 30221

1. Modificari facute:

Pentru cele 4 instructiuni suplimentare nu a fost necesara realizarea unor modificari aditionale inafara de cele necesare pentru definirea procesorului MIPS Pipeline:

- Scrierea sincrona in registrul de fisiere pe frontul descendente de ceas (falling\_edge(clk)).

- Modificarea unitatii ID, prin adaugarea campului care reprezinta datele (WriteData) care vor fi scrise

in registru la momentul WriteBack si excluderea mux-ului din ID.

- Modificarea algoritmului de testare pentru a evita hazardurile.

1. Registrele pipeline

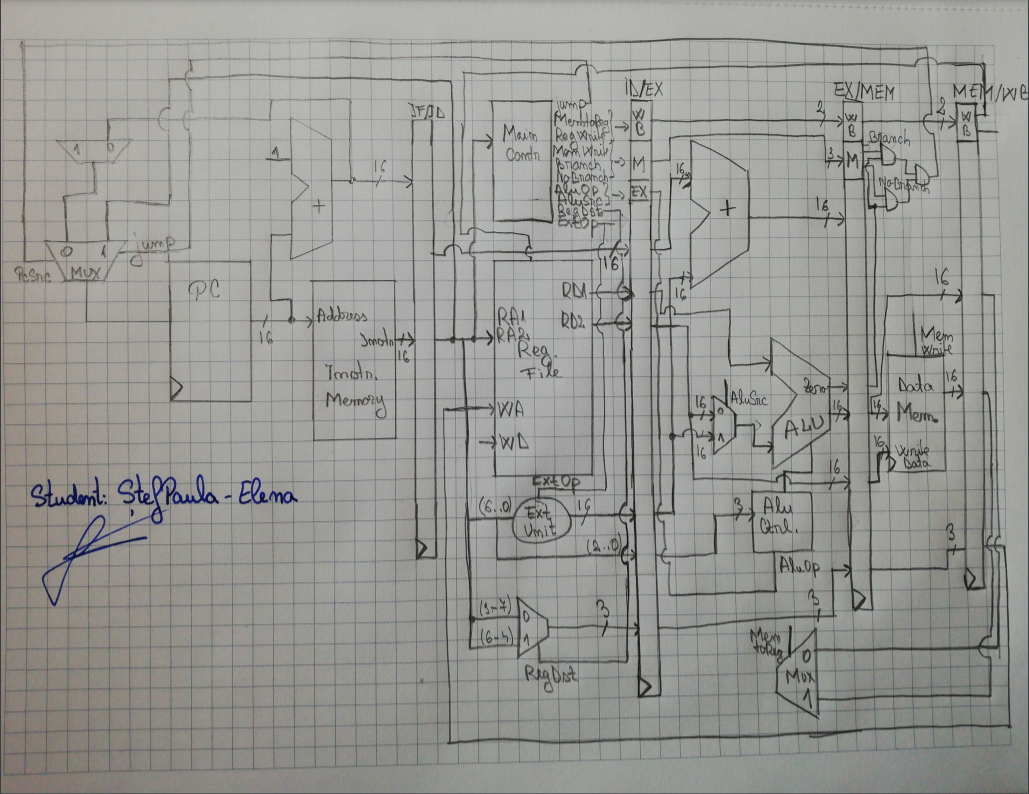
|  |
| --- |
| **REG\_IF\_ID(31** – **0)** |
| PC\_plus\_one(31 – 16) |
| Instruction(15 – 0) |
|  |
|  |
|  |
|  |

|  |
| --- |
| **REG\_ID\_EX(78** – **0>)** |
| MemtoReg (78) |
| RegWrite (77) |
| MemWrite (76) |
| Branch (75) |
| NoBranch (74) |
| AluOp (73-72) |
| AluSrc (71) |
| IF/ID (31-16) /PC\_plus\_one-> (70-55) |
| RD1\_rs (54-39) |
| RD2\_rt (38-23) |
| Ext\_imm(22-7) |
| Func\_sign(6-4) |
| sa (3) |
| Write\_address (2-0) |

|  |
| --- |
| **REG\_EX\_MEM(56** – **0)** |
| MemtoReg (56) – ID\_EX |
| RegWrite (55) -ID\_EX |
| MemWrite (54) – ID/EX |
| Branch (53) – ID\_EX |
| NoBranch (52) – ID\_EX |
| Branch\_address (51-36) |
| Zero (35) |
| AluRes (34-19) |
| Rd2\_rt (18-3)-ID\_EX |
| Write\_adress (2-0) – ID\_EX |

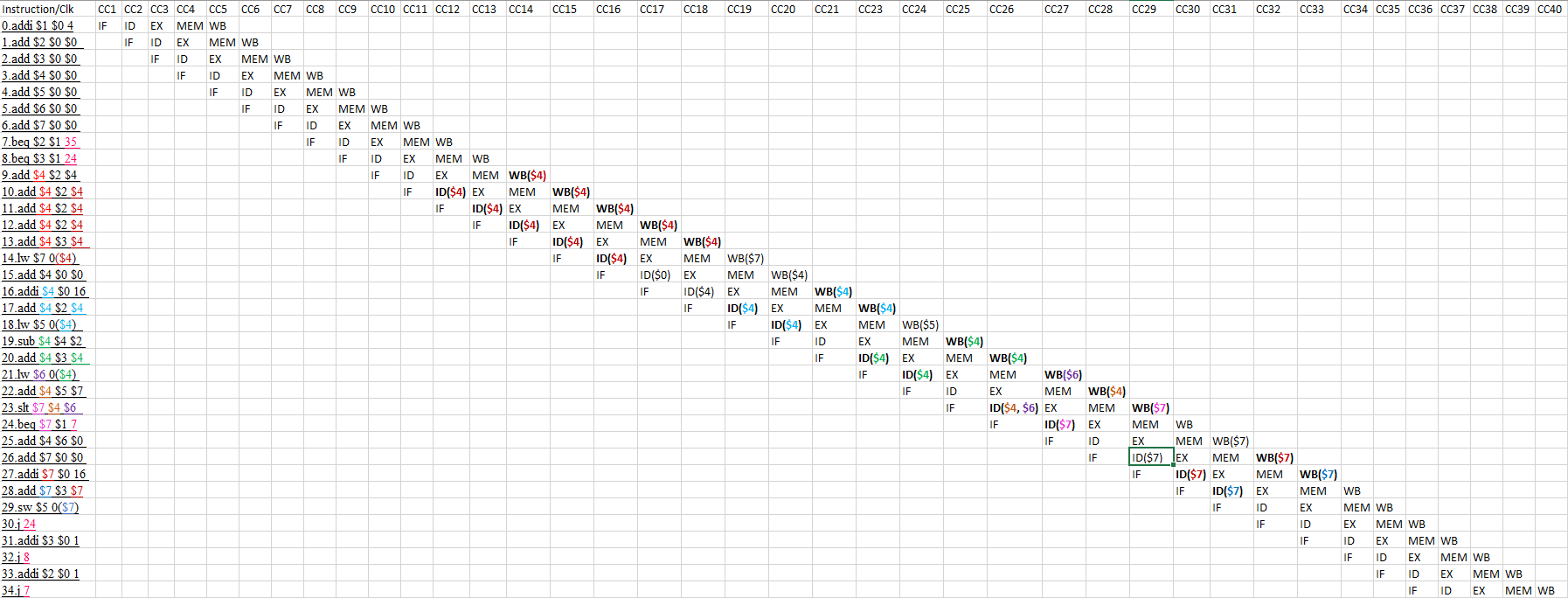
|  |
| --- |
| **REG\_MEM\_WB(36** – **0)** |
| MemtoReg(36) |
| RegWr(35) |
| MemData(34-19) |
| AluRes(18-3) – EX\_MEM |
| Write\_adress(2-0) – EX\_MEM |

1. Schema procesorulu



1. Analiza programului

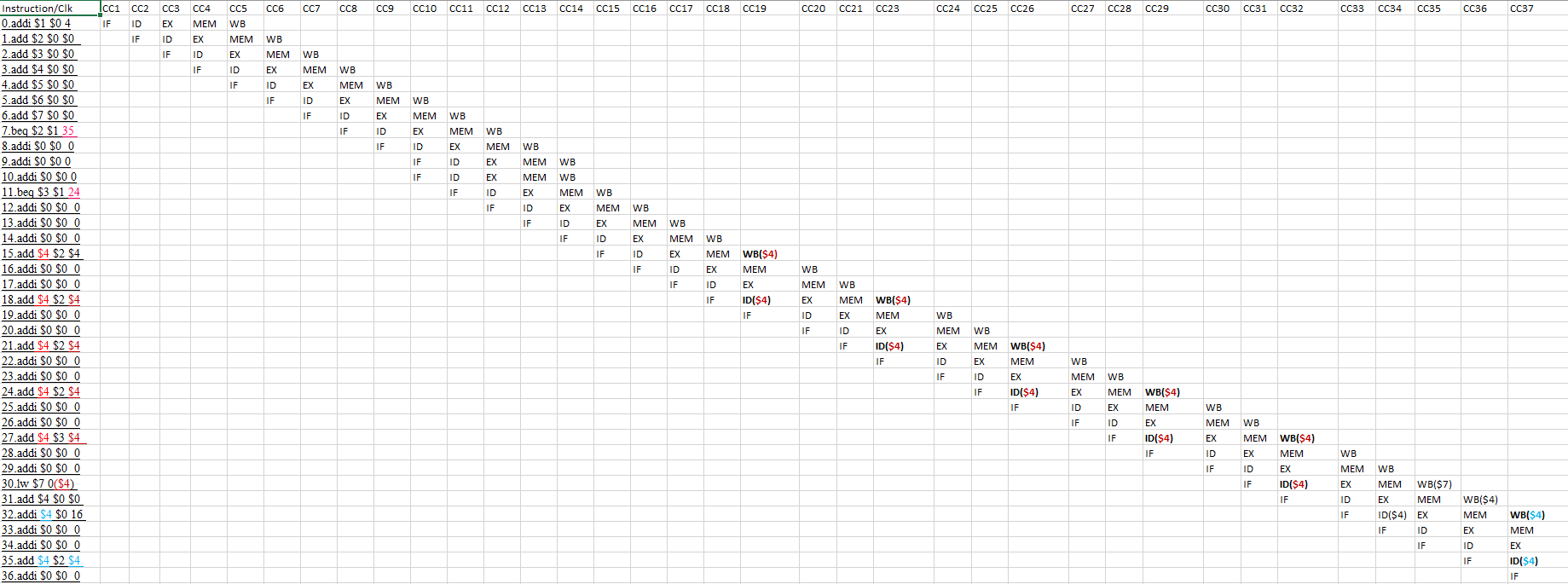
Programul original scris instructiune cu instructiune (cu culori am identitficat posibilele hazard-uri ) :

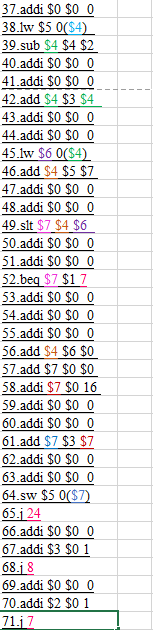
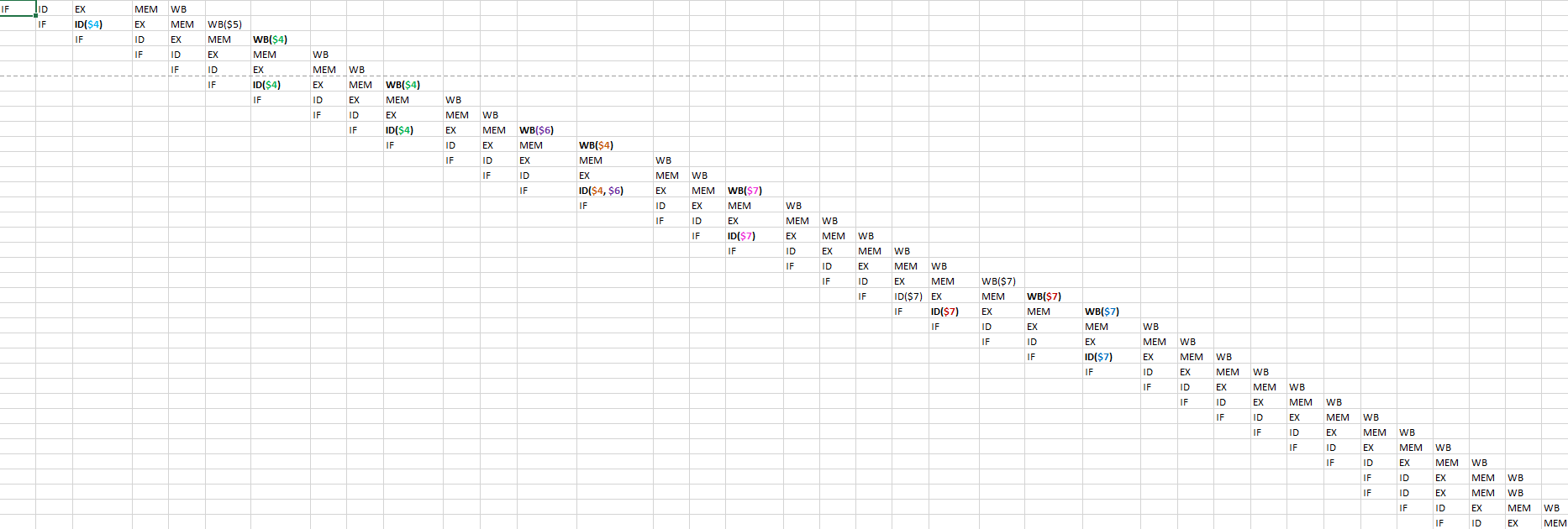


Am identificat urmatoarele hazard-uri:

1. La instructiunea 7 avem un “hazard de control” pentru beq
2. La instructiunea 8 avem un “hazard de control” pentru beq
3. Intre instructiuniile 9 si 10 avem un “hazard de date” (Read after write) pentru registrul $4
4. Intre instructiuniile 10 si 11 avem un “hazard de date” (Read after write) pentru registrul $4
5. Intre instructiuniile 11 si 12 avem un “hazard de date” (Read after write) pentru registrul $4
6. Intre instructiuniile 12 si 13 avem un “hazard de date” (Read after write) pentru registrul $4
7. Intre instructiuniile 13 si 14 avem un “hazard de date” (Read after write) pentru registrul $4
8. Intre instructiuniile 16 si 17 avem un “hazard de date” (Read after write) pentru registrul $4
9. Intre instructiuniile 17 si 18 avem un “hazard de date” (Read after write) pentru registrul $4
10. Intre instructiuniile 19 si 20 avem un “hazard de date” (Read after write) pentru registrul $4
11. Intre instructiuniile 20 si 21 avem un “hazard de date” (Read after write) pentru registrul $4
12. Intre instructiuniile 21 si 23 avem un “hazard de date” (load data) pentru registrul $6
13. Intre instructiuniile 22 si 23 avem un “hazard de date” (Read after write) pentru registrul $4
14. Intre instructiuniile 23 si 24 avem un “hazard de date” (Read after write) pentru registrul $7
15. La instructiunea 24 avem un “hazard de control” pentru beq
16. Intre instructiuniile 27 si 28 avem un “hazard de date” (Read after write) pentru registrul $7
17. Intre instructiuniile 28 si 29 avem un “hazard de date” (Read after write) pentru registrul $7
18. La instructiunea 30 avem un “hazard de control” pentru jump
19. La instructiunea 32 avem un “hazard de control” pentru jump
20. La instructiunea 8 avem un “hazard de control” pentru jump

Programul dupa rezolvarea hazard-urilor:





Codul in asamblare:

|  |
| --- |
| 0.addi $1 $0 4 |
| 1.add $2 $0 $0 |
| 2.add $3 $0 $0 |
| 3.add $4 $0 $0 |
| 4.add $5 $0 $0 |
| 5.add $6 $0 $0 |
| 6.add $7 $0 $0 |
| 7.beq $2 $1 65 |
| 8.addi $0 $0 0 |
| 9.addi $0 $0 0 |
| 10.addi $0 $0 0 |
| 11.beq $3 $1 57 |
| 12.addi $0 $0 0 |
| 13.addi $0 $0 0 |
| 14.addi $0 $0 0 |
| 15.add $4 $2 $4 |
| 16.addi $0 $0 0 |
| 17.addi $0 $0 0 |
| 18.add $4 $2 $4 |
| 19.addi $0 $0 0 |
| 20.addi $0 $0 0 |
| 21.add $4 $2 $4 |
| 22.addi $0 $0 0 |
| 23.addi $0 $0 0 |
| 24.add $4 $2 $4 |
| 25.addi $0 $0 0 |
| 26.addi $0 $0 0 |
| 27.add $4 $3 $4 |
| 28.addi $0 $0 0 |
| 29.addi $0 $0 0 |
| 30.lw $7 0($4) |
| 31.add $4 $0 $0 |
| 32.addi $4 $0 16 |
| 33.addi $0 $0 0 |
| 34.addi $0 $0 0 |
| 35.add $4 $2 $4 |
| 36.addi $0 $0 0 |
| 37.addi $0 $0 0 |
| 38.lw $5 0($4) |
| 39.sub $4 $4 $2 |
| 40.addi $0 $0 0 |
| 41.addi $0 $0 0 |
| 42.add $4 $3 $4 |
| 43.addi $0 $0 0 |
| 44.addi $0 $0 0 |
| 45.lw $6 0($4) |
| 46.add $4 $5 $7 |
| 47.addi $0 $0 0 |
| 48.addi $0 $0 0 |
| 49.slt $7 $4 $6 |
| 50.addi $0 $0 0 |
| 51.addi $0 $0 0 |
| 52.beq $7 $1 13 |
| 53.addi $0 $0 0 |
| 54.addi $0 $0 0 |
| 55.addi $0 $0 0 |
| 56.add $4 $6 $0 |
| 57.add $7 $0 $0 |
| 58.addi $7 $0 16 |
| 59.addi $0 $0 0 |
| 60.addi $0 $0 0 |
| 61.add $7 $3 $7 |
| 62.addi $0 $0 0 |
| 63.addi $0 $0 0 |
| 64.sw $5 0($7) |
| 65.j 52 |
| 66.addi $0 $0 0 |
| 67.addi $3 $0 1 |
| 68.j 11 |
| 69.addi $0 $0 0 |
| 70.addi $2 $0 1 |
| 71.j 7 |
|  |

1. Corectitudinea in vhdl

* Toate componentele descries in vhdl au fost implementate fara erori si a fost generat fisierul .bit cu success

Schema rezultata:

