Elemente funcționale:

Programul pare să afișeze valorile corespunzatoare pe SSD pentru primele linii de cod, și fiind încarcat pe plăcuță acestea au fost observațiile:

1. Preia instructiunea cu IFETCH corect.
2. PC este incrementat corespunzător, fiind incrementat și decrementat în funcție de branchuri și jumpuri sau doar cu +4 în cazul în care nu sunt active semnalele de jump sau branch.
3. Cei doi registrii RD1 si RD2 sunt afisați corect inițial, însă consider că din cauza unei erori în cod, aceștia nu sunt modificați corespunzător.
4. Ext\_Imm a reusit să fie extras corespunzător.
5. Alu res nu a putut fi determinat datorită registrilor care nu erau preluați corespunzător.
6. MemData era afisat corespunzător.
7. Portul de scriere WD nu a fost verificat.

Așadar, elementul care a fost clasificat drept nefuncțional a fost defapt parcursul instrucțiunilor, deoarece dintr-un jump eronat a fost dereglat parcursul firesc al codului mașină. Între timp a fost modificat codul și regiștrii pe care se lucrează, însă proiectul nu a putut fi reverificat pentru codul nou. Eroarea identificată a fost identificarea eronata a registrilor, in loc ca registrul t2 sa fie identificat cu numarul 10 (1010), l-am identificat cu 2 (0010). Am încercat modificarea codului pentru a repara greșeala însă nu am reusit să reparcurg codul pentru scăpări.

Prin încercarea de a modifica codul pentru a repara această eroare și altele aparute pe parcurs, se prea poate să existe neconcordanțe între zonele din tabelul cu trasarea execuției. Am încercat să il modific o dată cu modificările din cod, însă există zone nedeterminate sau lăsate din parcurgerea codului anterior.

Am utilizat acest IDE pentru a lucra cu codul mips assembly <https://courses.missouristate.edu/kenvollmar/mars/> și am încercat pe baza rezultatelor din interfață să modific codul astfel încât să se îndeplinească cerințele. În ide codul se efectuează până la final, afișând șirul sortat cu o poziție la final (executând o iterație din loop), însă pe placuță (la momentul testării, înainte de modificari), se realiza jumpul intr-o zonă nedefinită și valorile din registrii nu erau cei așteptați.

Instructiunile pe care le-am adaugat eu sunt:

De tip r:

Sll – Shift Left Logical

Aceasta instructiune deplasează logic la stânga pentru un registru ($t) , cu un număr de poziții indicat de celalalt registru ($s), si salvat în al treilea registru ($d).

|  |  |
| --- | --- |
| Format asamblare: | sll $d, $t, $s |
| RTL: | $d <- $t << $s; PC<-PC+4; |
| Format binar: | 000000 sssss ttttt ddddd 00000 000100 |
| Valorile semnalelor de control: | AluSrc <= '0';  extOp <= '0';  MemWrite <= '0';  MemToReg <= '0';  branch <= '0';  jump <= '0';  regDest <= '1';  regWrite <= '1';  AluOp <= "010"; |

Slt – set on less than

Daca registrul $s este mai mic decât registrul $t, atunci registrul $d este inițializat cu 1, altfel cu 0.

|  |  |
| --- | --- |
| Format asamblare: | slt $d, $s, $t |
| RTL: | If $s < $t then $d <- 1 else $d <- 0; |
| Format binar: | 000000 sssss ttttt ddddd 00000 000100 |
| Valorile semnalelor de control: | AluSrc <= '0';  extOp <= '0';  MemWrite <= '0';  MemToReg <= '0';  branch <= '0';  jump <= '0';  regDest <= '1';  regWrite <= '1';  AluOp <= "010"; |

De tip i

Lui

|  |  |
| --- | --- |
| Format asamblare: | lui $t, imm |
| RTL: | $t <- imm << 16 |
| Format binar: | 001111 sssss ttttt oooooooooooooooo |
| Valorile semnalelor de control: | AluSrc <= '0';  extOp <= '1';  MemWrite <= '0';  MemToReg <= '0';  branch <= '0';  jump <= '0';  regDest <= '0';  regWrite <= '1';  AluOp <= "010"; |

Addiu – Add Immediate Unsigned

|  |  |
| --- | --- |
| Format asamblare: | addiu $t,$s,imm |
| RTL: | rt=rs+imm |
| Format binar: | 001001 sssss ttttt oooooooooooooooo |
| Valorile semnalelor de control: | AluSrc <= '1';  extOp <= '1';  MemWrite <= '0';  MemToReg <= '0';  branch <= '0';  jump <= '0';  regDest <= '0';  regWrite <= '1';  AluOp <= "000"; |

Referințe:

<https://opencores.org/projects/plasma/opcodes>

<https://inst.eecs.berkeley.edu/~cs61c/resources/MIPS_help.html>