# RISC PLA/FSM Lab – MIPS

**Description:**

In this lab, we were asked to design an FSM controller for a MIPS process with more than six RISC instructions. We also needed to include control signals for the states and Boolean equations, state equations, and a PLA.

**Equipment Used:**

- MS Paint

**Procedure:**

I started by drawing the FSM in MS Paint, following the lectures located in the Week 11 folder. Then I noted the control signals for each state, which I used to create the state equations. After that, I created a truth table for the NS bits and wrote down the Boolean equations for all the variables. Finally, I utilized all this information to create a PLA diagram in MS paint.

**Results:**

*Control Signals for Each State*

- State 0:
    - MemRead
    - ALUSrcA = 0
    - IorD = 0
    - IRWrite
    - ALUSrcB = 01
    - ALUOp = 00

    - PCWrite
    - PCSource = 00
- State 1:
    - ALUSrcA = 0
    - ALUSrcB = 11
    - ALUOp = 00
- State 2:
    - ALUSrcA = 1
    - ALUSrcB = 10
    - ALUOp = 00
- State 3:
    - MemRead
    - IorD = 1
- State 4:
    - RegDst=0
    - RegWrite
    - MemtoReg=1
- State 5:
    - MemWrite
    - IorD = 1
- State 6:
    - ALUSrcA =1
    - ALUSrcB = 00

    - ALUOp= 10
- State 7:
    - RegDst = 1
    - RegWrite
    - MemtoReg = 0
- State 8:
    - ALUSrcA = 1
    - ALUSrcB = 00
    - ALUOp = 01
    - PCWriteCond
    - PCSource = 01
- State 9:
    - PCWrite
    - PCSource = 10

*Control Signals to Boolean Equations:*

- PCWrite = state0 + state9
- PCWriteCond = state8
- PCSource1 = state9
- PCSource0 = state8
- IorD = state3 + state5
- MemRead = state0 + state3
- MemWrite = state5
- MemToReg = state4

- ALUOp1 = state6
- ALUOp0 = state8
- ALUSrcA = state2 + state6 + state8
- ALUSrcB1 = state1 + state2
- ALUSrcB0 = state0 + state1
- RegWrite = state4 + state7
- RegDst = state7
- IRWrite = state0

*State Equations:*

- Next State 0 = state4 + state5 + state7 + state8 + state9
- Next State 1 = state0
- Next State 2 = state1(lw + sw)
- Next State 3 = state 2(lw)
- Next State 4 = state3
- Next State 5 = state2(sw)
- Next State 6 = state1(R-type)
- Next State 7 = state6
- Next State 8 = state1(beq)
- Next State 9 = state1(jump)
- NS0 = Next State 1 + Next State 3 + Next State 5 + Next State 7 + Next State 9 = state0 + state2(lw) + state2(sw) + state6 + state1(jump)

- NS1 = Next State 2 + Next State 3 + Next State6 + Next State 7 = state1(lw+sw) + stae2(lw) + state1(R-type) + state6
- NS2 = Next State 4 + Next State 5 + Next State 6 + Next State 7 = state3 + state2(sw) + state1(R-type) + state6
- NS3 = Next State 8 + Next State 9 = state1(beq) +state1(jump)

*Table 1 A truth table for the NS bits.*

| NS3 | NS2 | NS1 | NS0 |
|---|---|---|---|
| 0 | 0 | 0 | 0 |
| 0 | 0 | 0 | 1 |
| 0 | 0 | 1 | 0 |
| 0 | 0 | 1 | 1 |
| 0 | 1 | 0 | 0 |
| 0 | 1 | 0 | 1 |
| 0 | 1 | 1 | 0 |
| 0 | 1 | 1 | 1 |
| 1 | 0 | 0 | 0 |
| 1 | 0 | 0 | 1 |

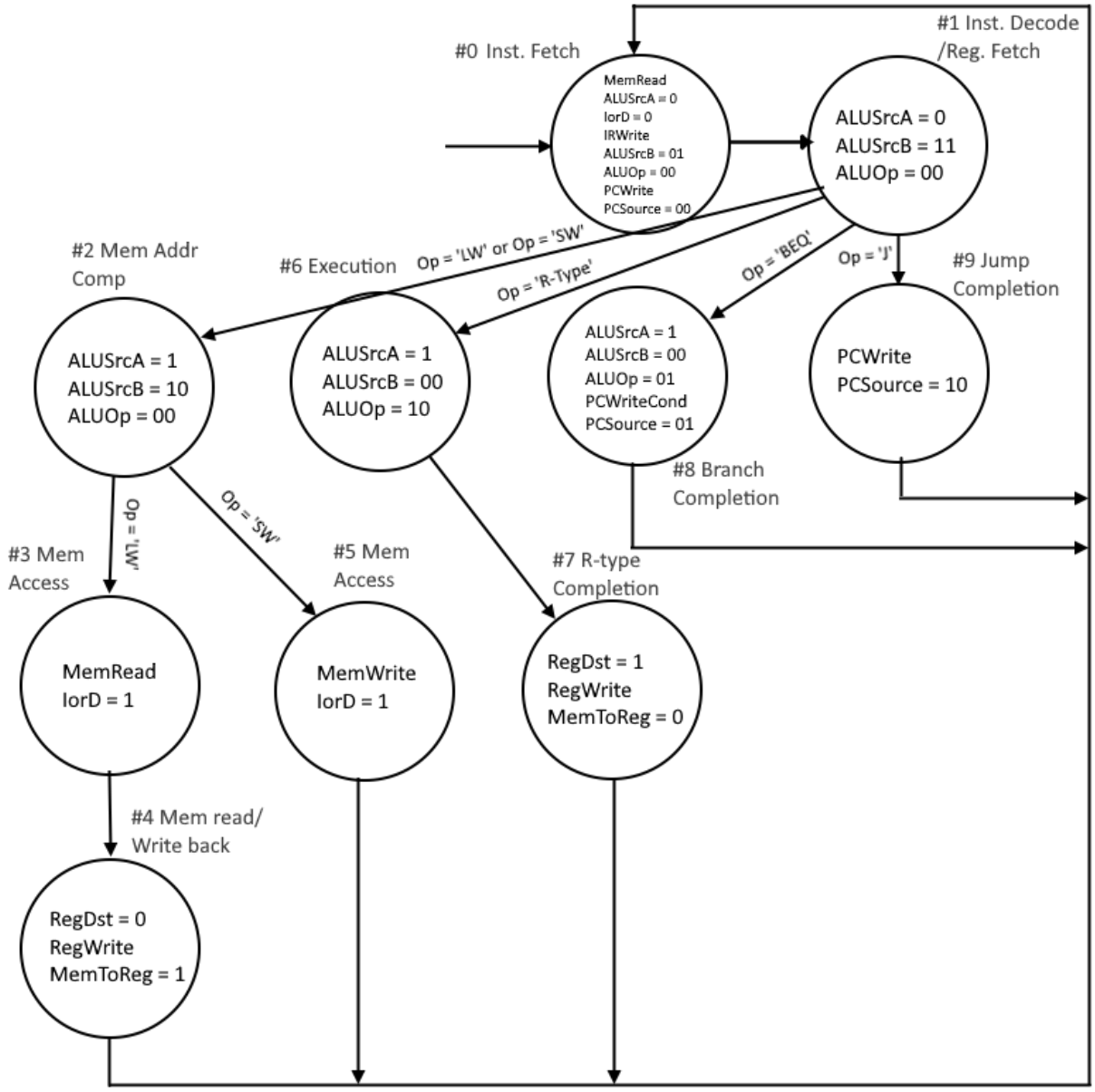


*Figure 1 An FSM controller for the MIPS processor*.

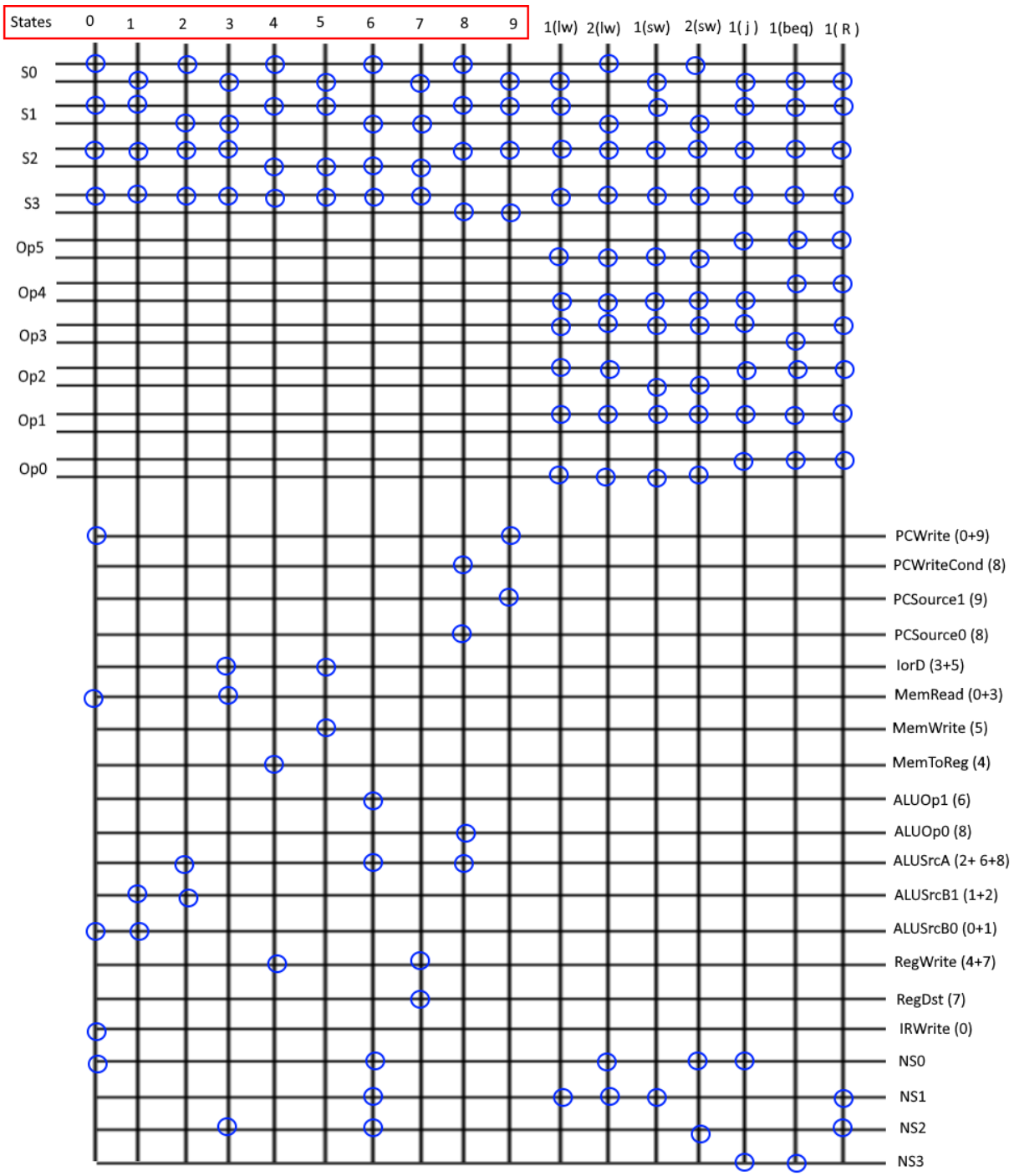


*Figure 2 A PLA diagram for the MIPS processor.*

**Conclusions:**

This lab helped me understand the process of designing an FSM and PLA for a processor datapath. Although it looked intimidating at first, I realized that it’s actually relatively simple. This knowledge will help me in the future when I have to add additional instructions to a datapath.