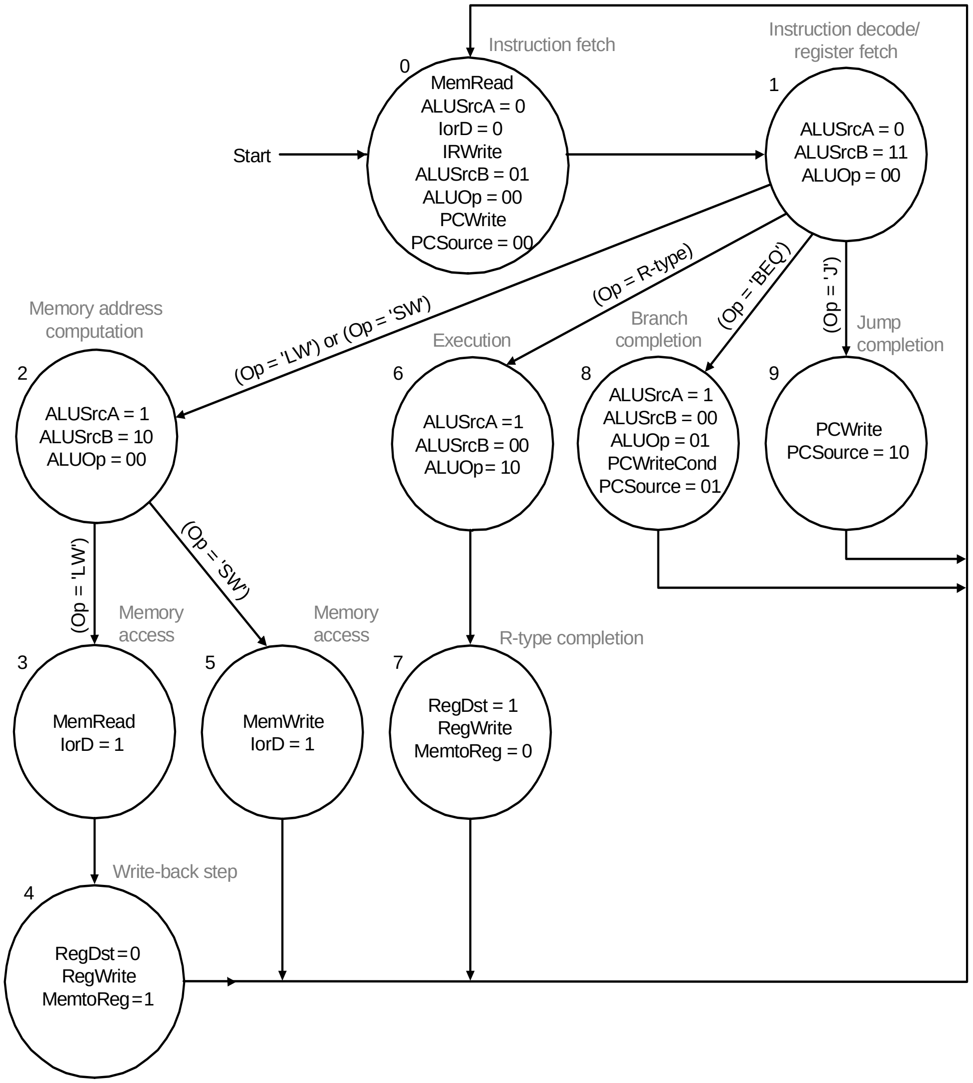
CprE 381 Lab 8
Multi-Cycle Datapath Implementation

In this lab you will construct a multi-cycle datapath. You have learned that a single-cycle implementation is an inefficient use of resources and a multi-cycle datapath is an alternate implementation that increases the efficiency.

Part 1 – Control

The major difference between the single cycle datapath and the multi-cycle datapath is the control unit. In the previous lab, the control unit was entirely combinational logic. The multi-cycle datapath requires sequential logic to implement a state machine. The control signals are now dependant on both the instruction being executed and the stage of execution.

1. Implement a finite state machine shown in the following diagram to control the multi-cycle datapath. A shell of a verilog FSM is provided to get you started.



```
// Incomplete FSM implementation for above diagram
module CTRL(clk, rst, opcode,
          IRWrite, IorD, //…others omitted);
    // Input signals for control unit
    input clk, rst; input [5:0] opcode;
    // Output control signals
    output IRWrite, IorD;
    // State variables
    reg [3:0] state, next_state;
    // Registered output signals based on state
    reg IRWrite, IorD;

    // State assignment
    parameter [3:0]
       IFETCH = 0,
       DECODE = 1,
       MEMACCESS = 2,
       LW = 3,
       SW = 5,
       RTYPE_EXEC = 6,
       RTYPE_2 = 7;
       // other states omitted

// Change state when clock arrives
always @ (posedge clk or negedge rst)
begin
   if (!rst)
      state = IFETCH; // asynchronous reset
   else
      state = next_state;  // normal state transition
end
```

```
// Determine next state from current state and opcode
always @ (opcode or state)
begin
   case (state)
      IFETCH: state_next = DECODE;
      DECODE: if (opcode == 6’h00) state_next =
RTYPE_EXEC;
           else if (opcode == 6’h35 || opcode == 6’h43)
state_next = MEMACCESS;
      MEMACCESS: if(opcode == 6’h35) state_next = LW;
         else if(opcode == 6’h43) state_next = SW;
      RTYPE_EXEC: state_next = RTYPE_COMP;
      RTYPE_COMP: state_next = IFETCH;
    // other states omitted
   endcase
end

// Assert output signals based on current state
assign IRWrite = (state == IFETCH) ? 1 : 0;

 assign IorD = (state == LW || state == SW) ? 1 : 0;

//Other signals omitted

end module
```

More information on the meaning of the various signals can be found in your book in Figure 5.29 on page 324.

2. Test your finite state machine for all state transitions to ensure that the proper control signals are asserted.

Part 2 – Complete Multi-Cycle Datapath

1. Combine the control you developed in Part 1 with the remaining components necessary to construct a complete multi-cycle datapath. Use the diagram in Figure 5.28 on page 323 in your book as a guide.
2. Test and evaluate the functionality of your multi-cycle datapath.