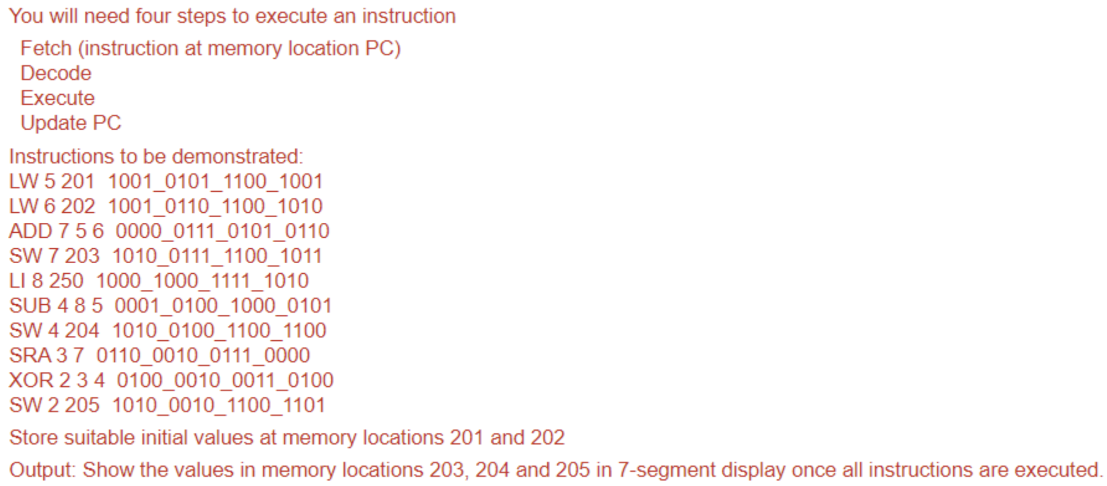
Oscar Osornio

Final Project EE 5193 (16-bit RISC MIPS Processor)

Problem Statement: Create a 16-bit RISC MIPS Processor. The project will be able to properly execute 10 instructions from memory and store the values in memory, then display them on the seven-segment display on the Nexys board. Below are the layout and instructions given.



You will need four steps to execute an instruction

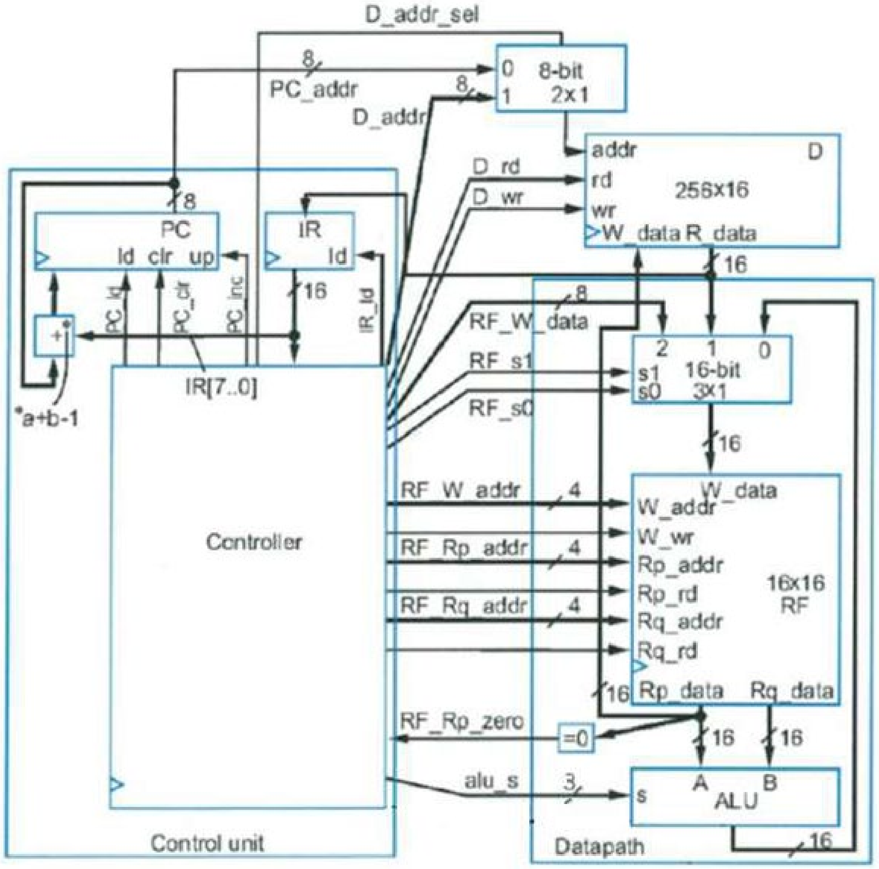
Fetch (instruction at memory location PC)
Decode
Execute
Update PC

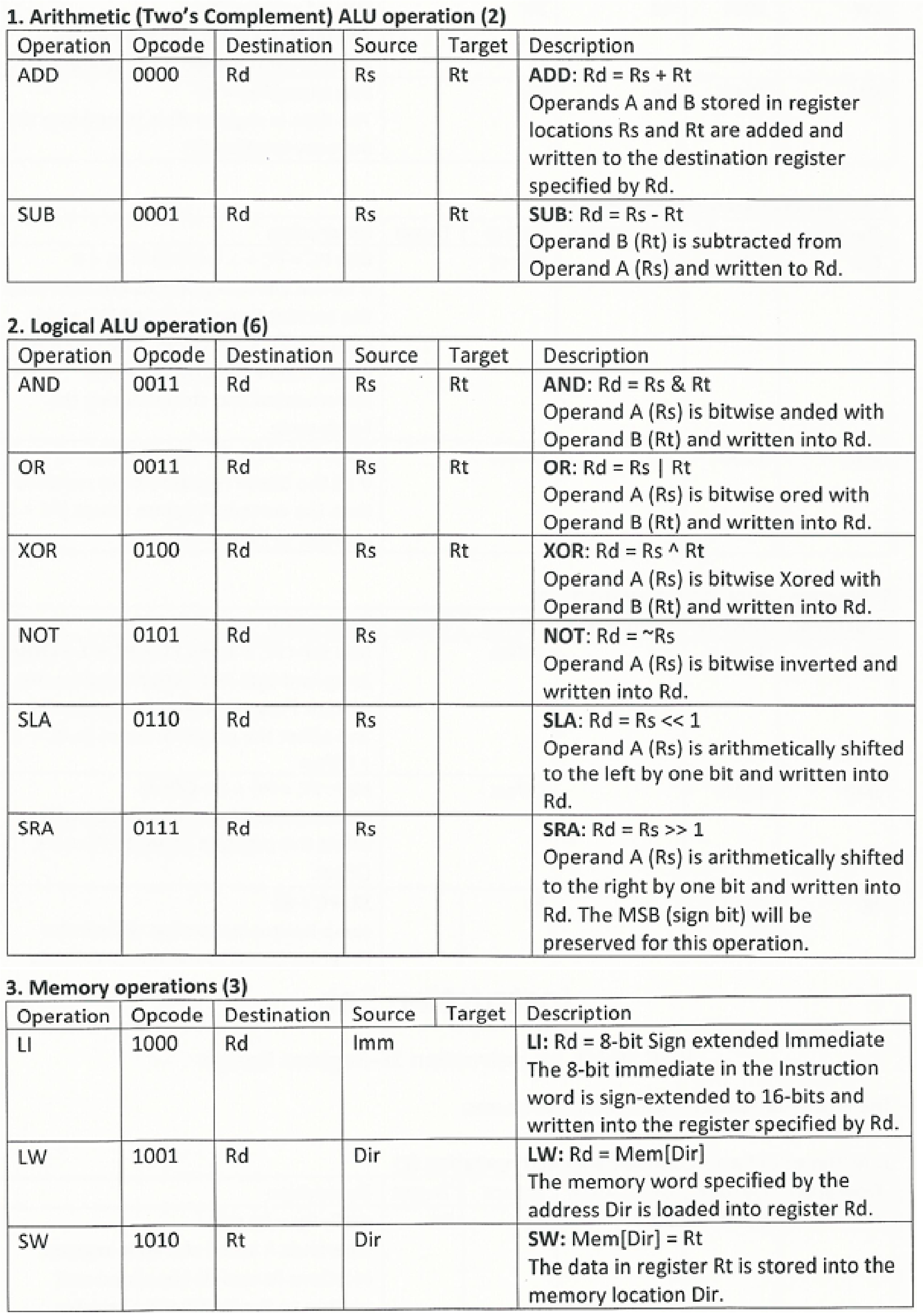
Instructions to be demonstrated:

LW 5 201	1001_0101_1100_1001
LW 6 202	1001_0110_1100_1010
ADD 7 5 6	0000_0111_0101_0110
SW 7 203	1010_0111_1100_1011
LI 8 250	1000_1000_1111_1010
SUB 4 8 5	0001_0100_1000_0101
SW 4 204	1010_0100_1100_1100
SRA 3 7	0110_0010_0111_0000
XOR 2 3 4	0100_0010_0011_0100
SW 2 205	1010_0010_1100_1101

Store suitable initial values at memory locations 201 and 202

Output: Show the values in memory locations 203, 204 and 205 in 7-segment display once all instructions are executed.





1. Arithmetic (Two's Complement) ALU operation (2)

Operation	Opcode	Destination	Source	Target	Description
ADD	0000	Rd	Rs	Rt	**ADD**: Rd = Rs + Rt Operands A and B stored in register locations Rs and Rt are added and written to the destination register specified by Rd.
SUB	0001	Rd	Rs	Rt	**SUB**: Rd = Rs - Rt Operand B (Rt) is subtracted from Operand A (Rs) and written to Rd.

2. Logical ALU operation (6)

Operation	Opcode	Destination	Source	Target	Description
AND	0011	Rd	Rs	Rt	**AND**: Rd = Rs & Rt Operand A (Rs) is bitwise anded with Operand B (Rt) and written into Rd.
OR	0011	Rd	Rs	Rt	**OR**: Rd = Rs | Rt Operand A (Rs) is bitwise ored with Operand B (Rt) and written into Rd.
XOR	0100	Rd	Rs	Rt	**XOR**: Rd = Rs ^ Rt Operand A (Rs) is bitwise Xored with Operand B (Rt) and written into Rd.
NOT	0101	Rd	Rs		**NOT**: Rd = ~Rs Operand A (Rs) is bitwise inverted and written into Rd.
SLA	0110	Rd	Rs		**SLA**: Rd = Rs << 1 Operand A (Rs) is arithmetically shifted to the left by one bit and written into Rd.
SRA	0111	Rd	Rs		**SRA**: Rd = Rs >> 1 Operand A (Rs) is arithmetically shifted to the right by one bit and written into Rd. The MSB (sign bit) will be preserved for this operation.

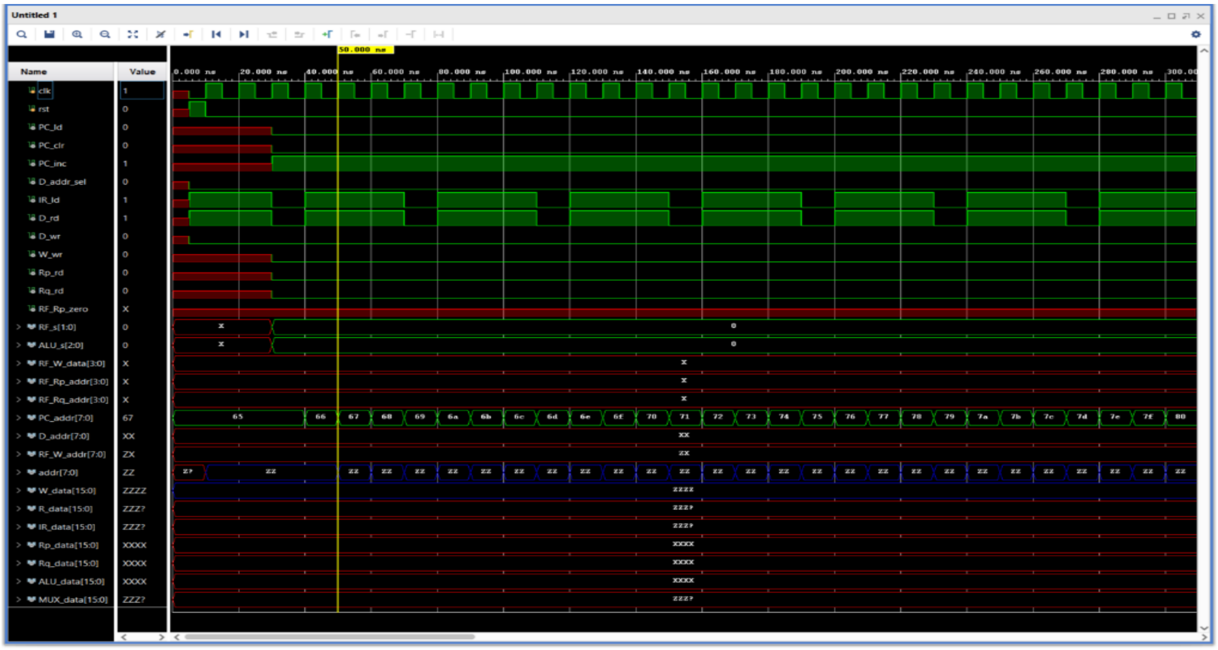
3. Memory operations (3)

Operation	Opcode	Destination	Source	Target	Description
LI	1000	Rd	Imm		**LI:** Rd = 8-bit Sign extended Immediate The 8-bit immediate in the Instruction word is sign-extended to 16-bits and written into the register specified by Rd.
LW	1001	Rd	Dir		**LW:** Rd = Mem[Dir] The memory word specified by the address Dir is loaded into register Rd.
SW	1010	Rt	Dir		**SW:** Mem[Dir] = Rt The data in register Rt is stored into the memory location Dir.

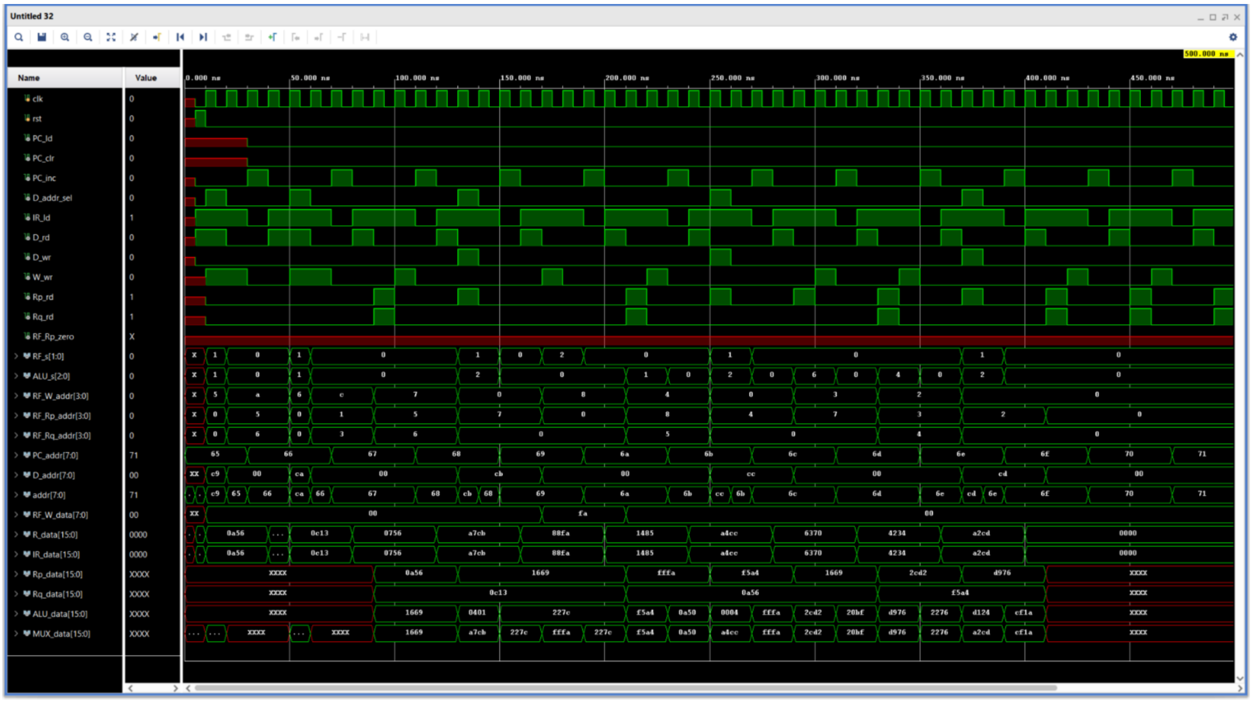
Approach: My approach involved using Assignment 3 (ASM+D) to properly divide up the sections of the project into modules and have them all initialized on a single top module that would just merge all the different pieces.

Problems: I encountered several problems during the simulation step. I had disconnected wires (showing Z as an output), failed signals (showing x as an output), and failing modules. I corrected the modules by making sure that all the proper connections were correct in the top module. That helped as well with the high impedance outputs. I also went ahead and checked the reg signals in each individual module to make sure that they had the right size. For failing signals, I made sure that I was using non-blocking signals through most of the modules to make sure that the signals were being transmitted at the right time. I was able to properly get all the information transmitted throughout the different steps of the instruction cycle. Below are the before and after of the simulation waves.

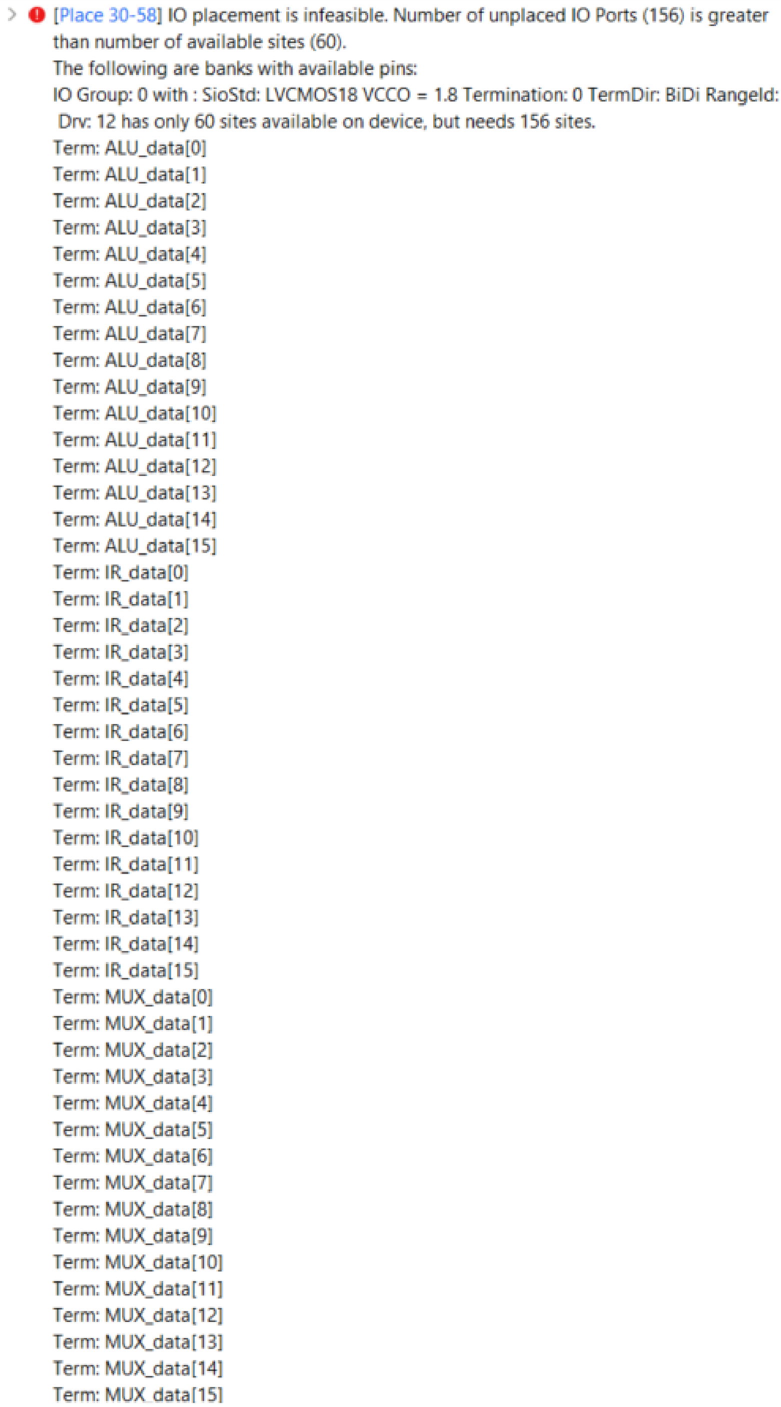
Before:



After:



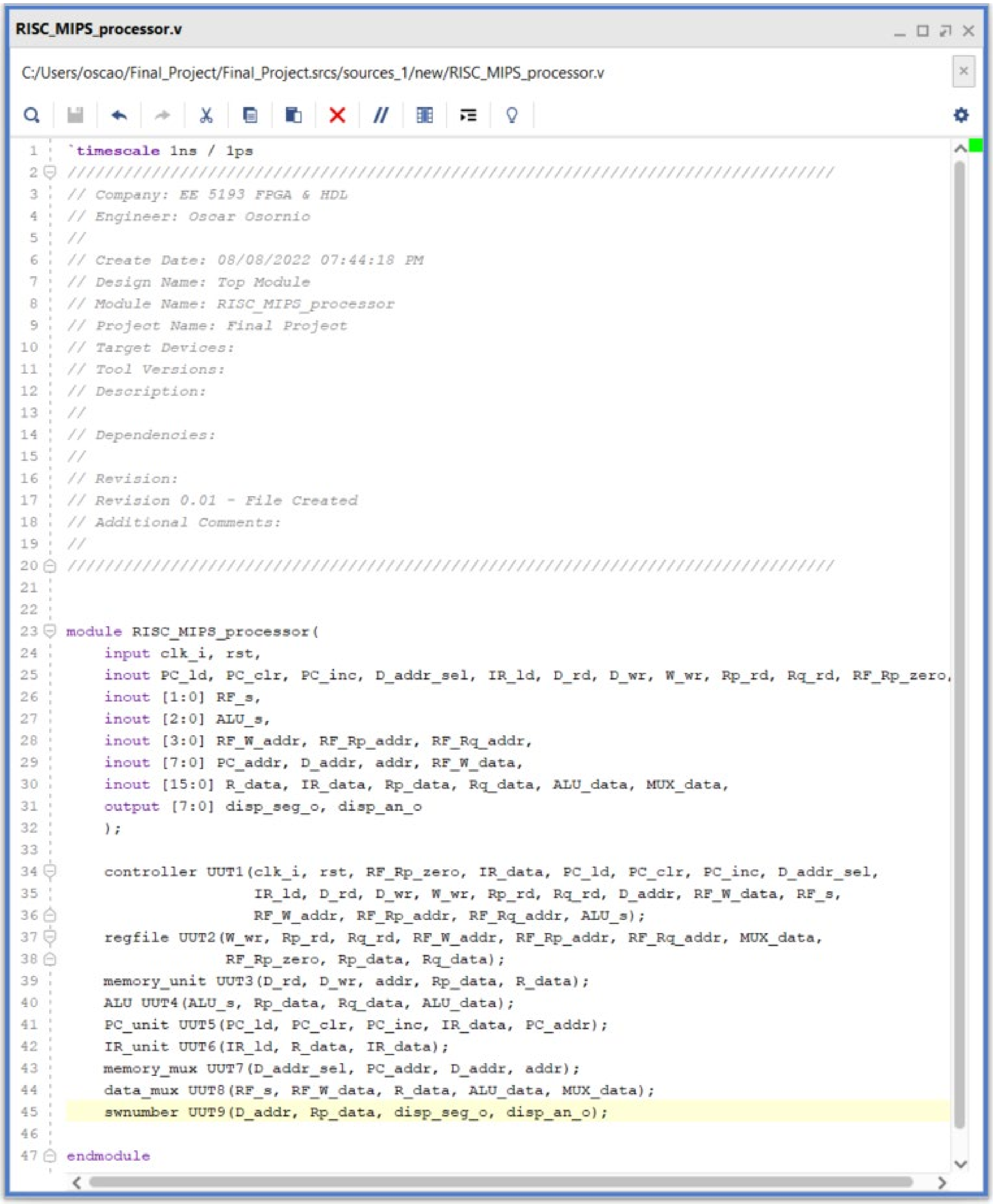
Implementation, on the other hand, was not able to work properly. I was able to get past the synthesis step after a small correction but was not able to properly implement it. I got the error below:



```
[Place 30-58] IO placement is infeasible. Number of unplaced IO Ports (156) is greater
than number of available sites (60).
The following are banks with available pins:
IO Group: 0 with : SioStd: LVCMOS18 VCCO = 1.8 Termination: 0 TermDir: BiDi RangeId:
 Drv: 12 has only 60 sites available on device, but needs 156 sites.
Term: ALU_data[0]
Term: ALU_data[1]
Term: ALU_data[2]
Term: ALU_data[3]
Term: ALU_data[4]
Term: ALU_data[5]
Term: ALU_data[6]
Term: ALU_data[7]
Term: ALU_data[8]
Term: ALU_data[9]
Term: ALU_data[10]
Term: ALU_data[11]
Term: ALU_data[12]
Term: ALU_data[13]
Term: ALU_data[14]
Term: ALU_data[15]
Term: IR_data[0]
Term: IR_data[1]
Term: IR_data[2]
Term: IR_data[3]
Term: IR_data[4]
Term: IR_data[5]
Term: IR_data[6]
Term: IR_data[7]
Term: IR_data[8]
Term: IR_data[9]
Term: IR_data[10]
Term: IR_data[11]
Term: IR_data[12]
Term: IR_data[13]
Term: IR_data[14]
Term: IR_data[15]
Term: MUX_data[0]
Term: MUX_data[1]
Term: MUX_data[2]
Term: MUX_data[3]
Term: MUX_data[4]
Term: MUX_data[5]
Term: MUX_data[6]
Term: MUX_data[7]
Term: MUX_data[8]
Term: MUX_data[9]
Term: MUX_data[10]
Term: MUX_data[11]
Term: MUX_data[12]
Term: MUX_data[13]
Term: MUX_data[14]
Term: MUX_data[15]
```

I believe the error is not resolvable in the time left to submit the project. Due to this, I will be submitting the assignment as is. If there was more time to go ahead and work on this project, then I would redesign the project to make sure that it could properly work in the Nexys board.

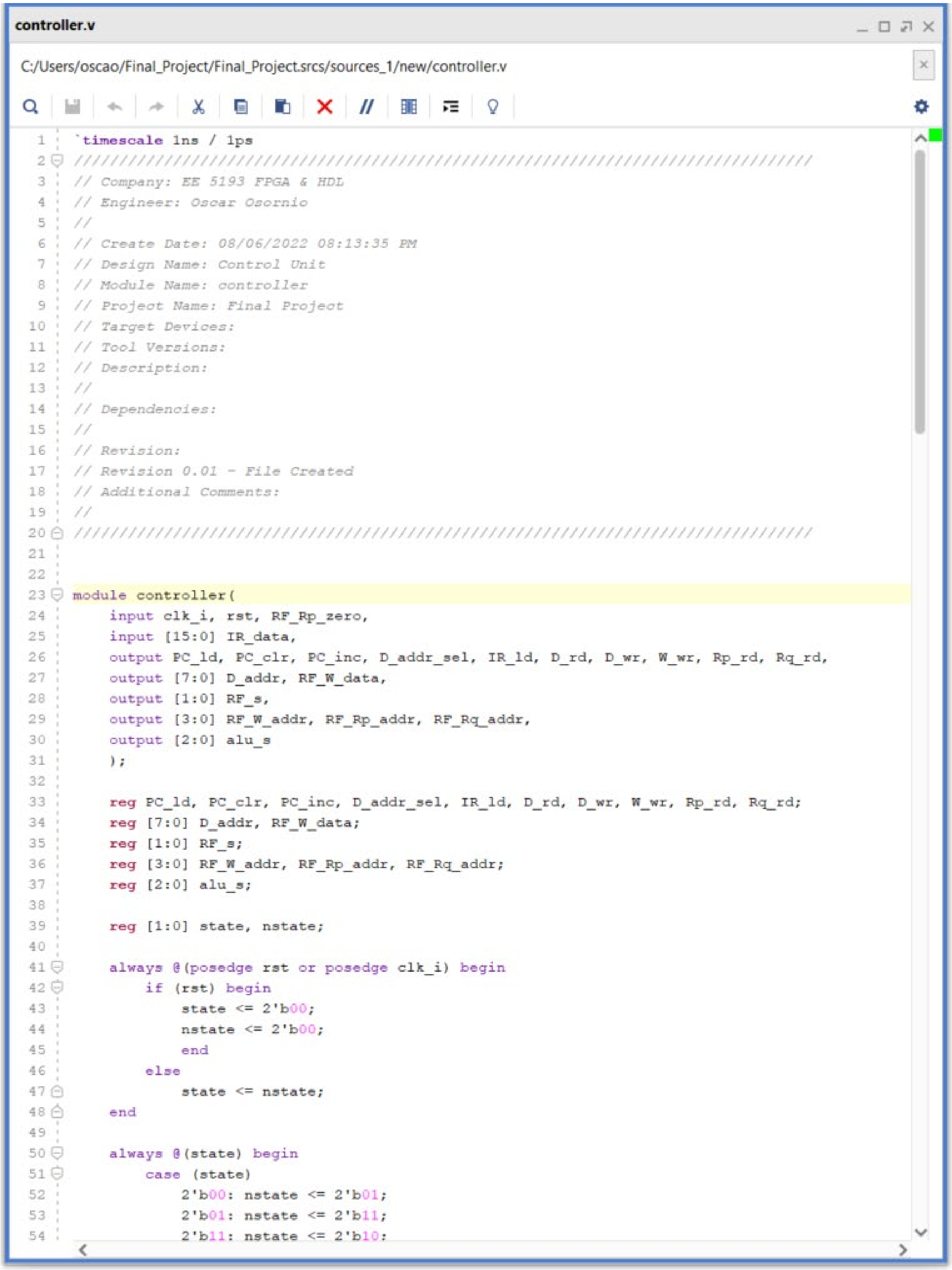
Top Module:



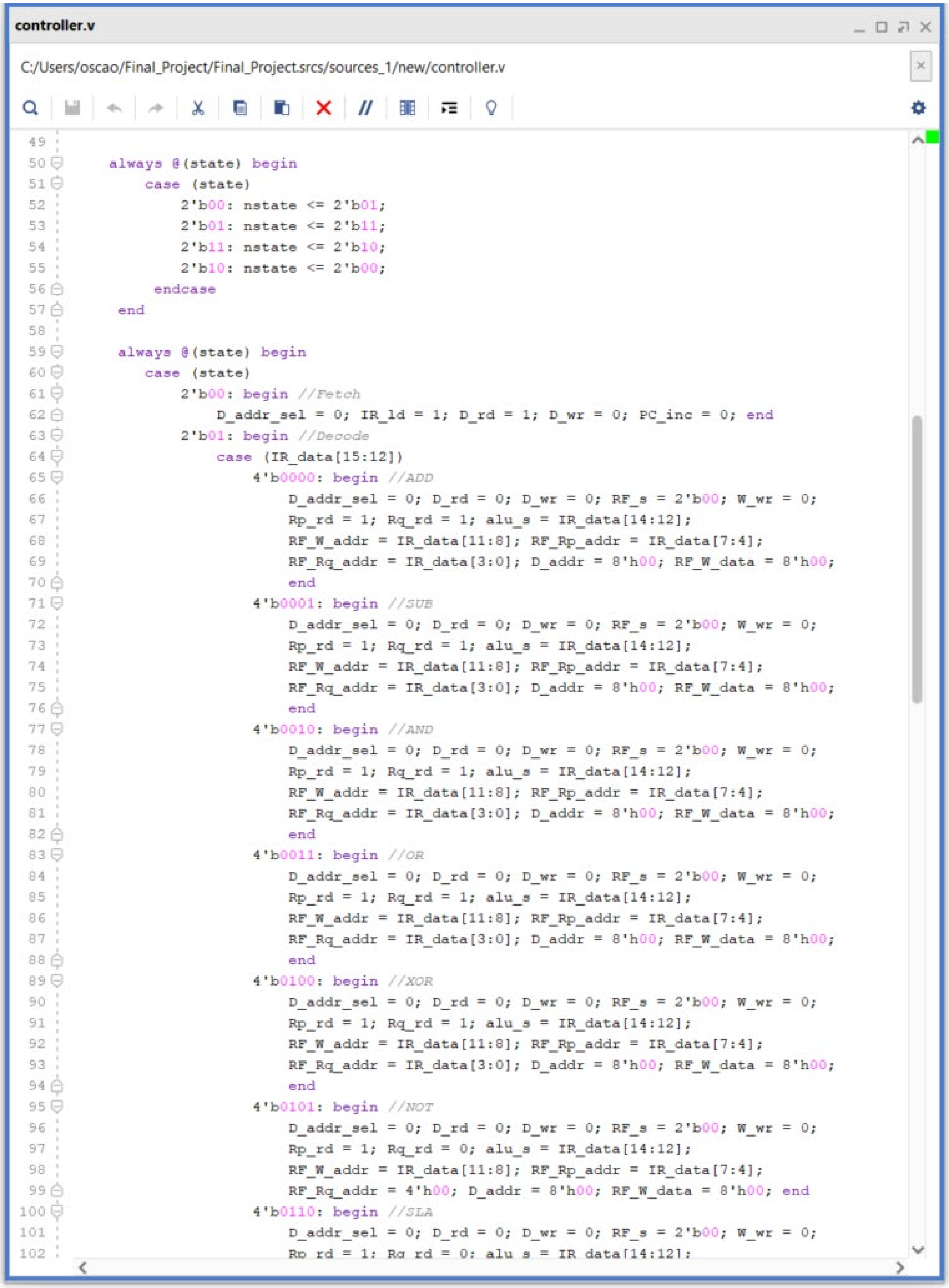
C:/Users/oscao/Final_Project/Final_Project.srcs/sources_1/new/RISC_MIPS_processor.v

```verilog
`timescale 1ns / 1ps
////////////////////////////////////////////////////////////////////////////////
// Company: EE 5193 FPGA & HDL
// Engineer: Oscar Osornio
//
// Create Date: 08/08/2022 07:44:18 PM
// Design Name: Top Module
// Module Name: RISC_MIPS_processor
// Project Name: Final Project
// Target Devices:
// Tool Versions:
// Description:
//
// Dependencies:
//
// Revision:
// Revision 0.01 - File Created
// Additional Comments:
//
////////////////////////////////////////////////////////////////////////////////


module RISC_MIPS_processor(
    input clk_i, rst,
    inout PC_ld, PC_clr, PC_inc, D_addr_sel, IR_ld, D_rd, D_wr, W_wr, Rp_rd, Rq_rd, RF_Rp_zero,
    inout [1:0] RF_s,
    inout [2:0] ALU_s,
    inout [3:0] RF_W_addr, RF_Rp_addr, RF_Rq_addr,
    inout [7:0] PC_addr, D_addr, addr, RF_W_data,
    inout [15:0] R_data, IR_data, Rp_data, Rq_data, ALU_data, MUX_data,
    output [7:0] disp_seg_o, disp_an_o
    );

    controller UUT1(clk_i, rst, RF_Rp_zero, IR_data, PC_ld, PC_clr, PC_inc, D_addr_sel,
                    IR_ld, D_rd, D_wr, W_wr, Rp_rd, Rq_rd, D_addr, RF_W_data, RF_s,
                    RF_W_addr, RF_Rp_addr, RF_Rq_addr, ALU_s);
    regfile UUT2(W_wr, Rp_rd, Rq_rd, RF_W_addr, RF_Rp_addr, RF_Rq_addr, MUX_data,
                 RF_Rp_zero, Rp_data, Rq_data);
    memory_unit UUT3(D_rd, D_wr, addr, Rp_data, R_data);
    ALU UUT4(ALU_s, Rp_data, Rq_data, ALU_data);
    PC_unit UUT5(PC_ld, PC_clr, PC_inc, IR_data, PC_addr);
    IR_unit UUT6(IR_ld, R_data, IR_data);
    memory_mux UUT7(D_addr_sel, PC_addr, D_addr, addr);
    data_mux UUT8(RF_s, RF_W_data, R_data, ALU_data, MUX_data);
    swnumber UUT9(D_addr, Rp_data, disp_seg_o, disp_an_o);

endmodule
```

Controller:


controller.v
C:/Users/oscao/Final_Project/Final_Project.srcs/sources_1/new/controller.v

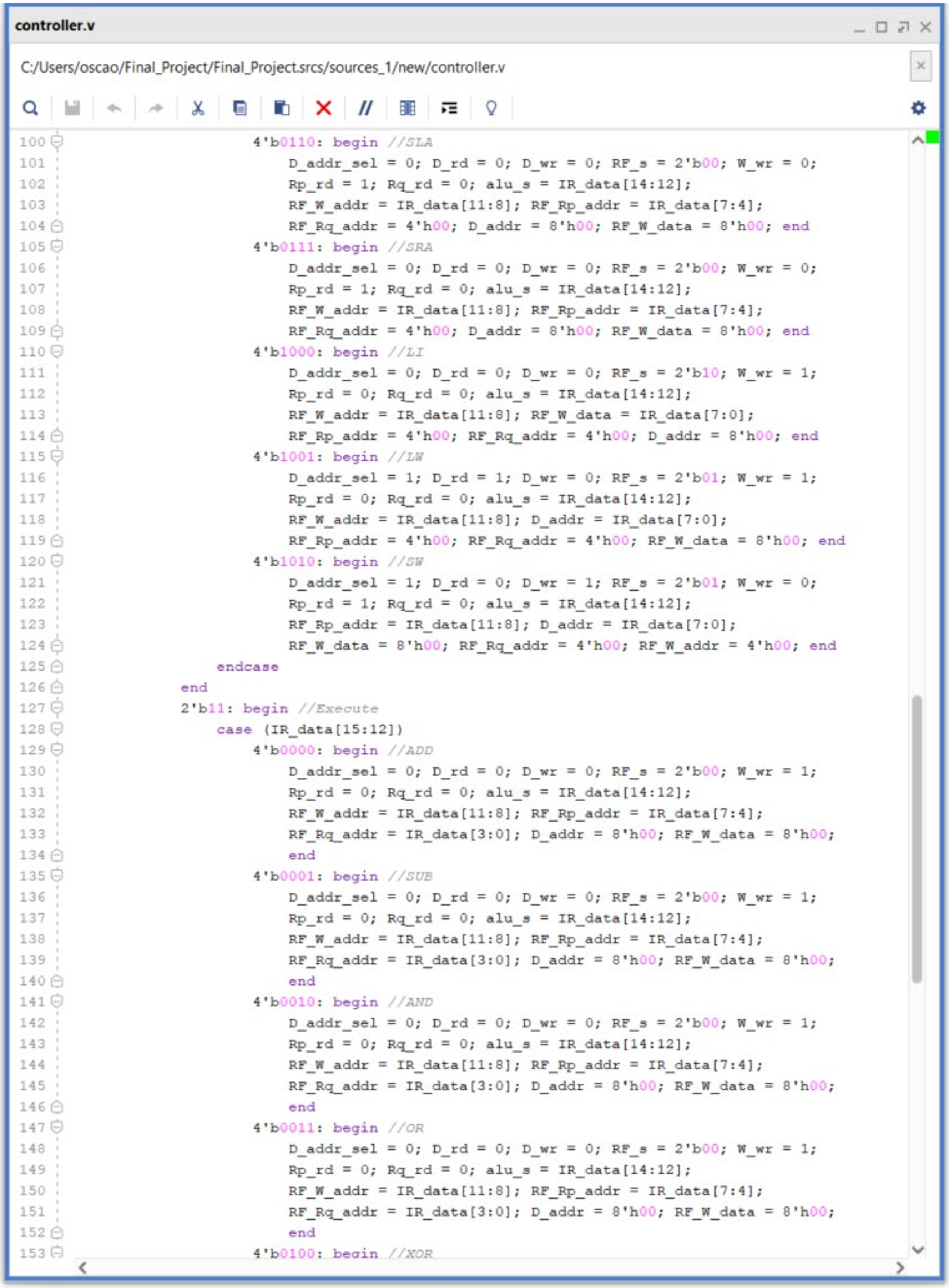
```
`timescale 1ns / 1ps
//////////////////////////////////////////////////////////////////////////////////
// Company: EE 5193 FPGA & HDL
// Engineer: Oscar Osornio
//
// Create Date: 08/06/2022 08:13:35 PM
// Design Name: Control Unit
// Module Name: controller
// Project Name: Final Project
// Target Devices:
// Tool Versions:
// Description:
//
// Dependencies:
//
// Revision:
// Revision 0.01 - File Created
// Additional Comments:
//
//////////////////////////////////////////////////////////////////////////////////


module controller(
    input clk_i, rst, RF_Rp_zero,
    input [15:0] IR_data,
    output PC_ld, PC_clr, PC_inc, D_addr_sel, IR_ld, D_rd, D_wr, W_wr, Rp_rd, Rq_rd,
    output [7:0] D_addr, RF_W_data,
    output [1:0] RF_s,
    output [3:0] RF_W_addr, RF_Rp_addr, RF_Rq_addr,
    output [2:0] alu_s
    );

    reg PC_ld, PC_clr, PC_inc, D_addr_sel, IR_ld, D_rd, D_wr, W_wr, Rp_rd, Rq_rd;
    reg [7:0] D_addr, RF_W_data;
    reg [1:0] RF_s;
    reg [3:0] RF_W_addr, RF_Rp_addr, RF_Rq_addr;
    reg [2:0] alu_s;

    reg [1:0] state, nstate;

    always @(posedge rst or posedge clk_i) begin
        if (rst) begin
            state <= 2'b00;
            nstate <= 2'b00;
            end
        else
            state <= nstate;
    end

    always @(state) begin
        case (state)
            2'b00: nstate <= 2'b01;
            2'b01: nstate <= 2'b11;
            2'b11: nstate <= 2'b10;
```



controller.v

C:/Users/oscao/Final_Project/Final_Project.srcs/sources_1/new/controller.v

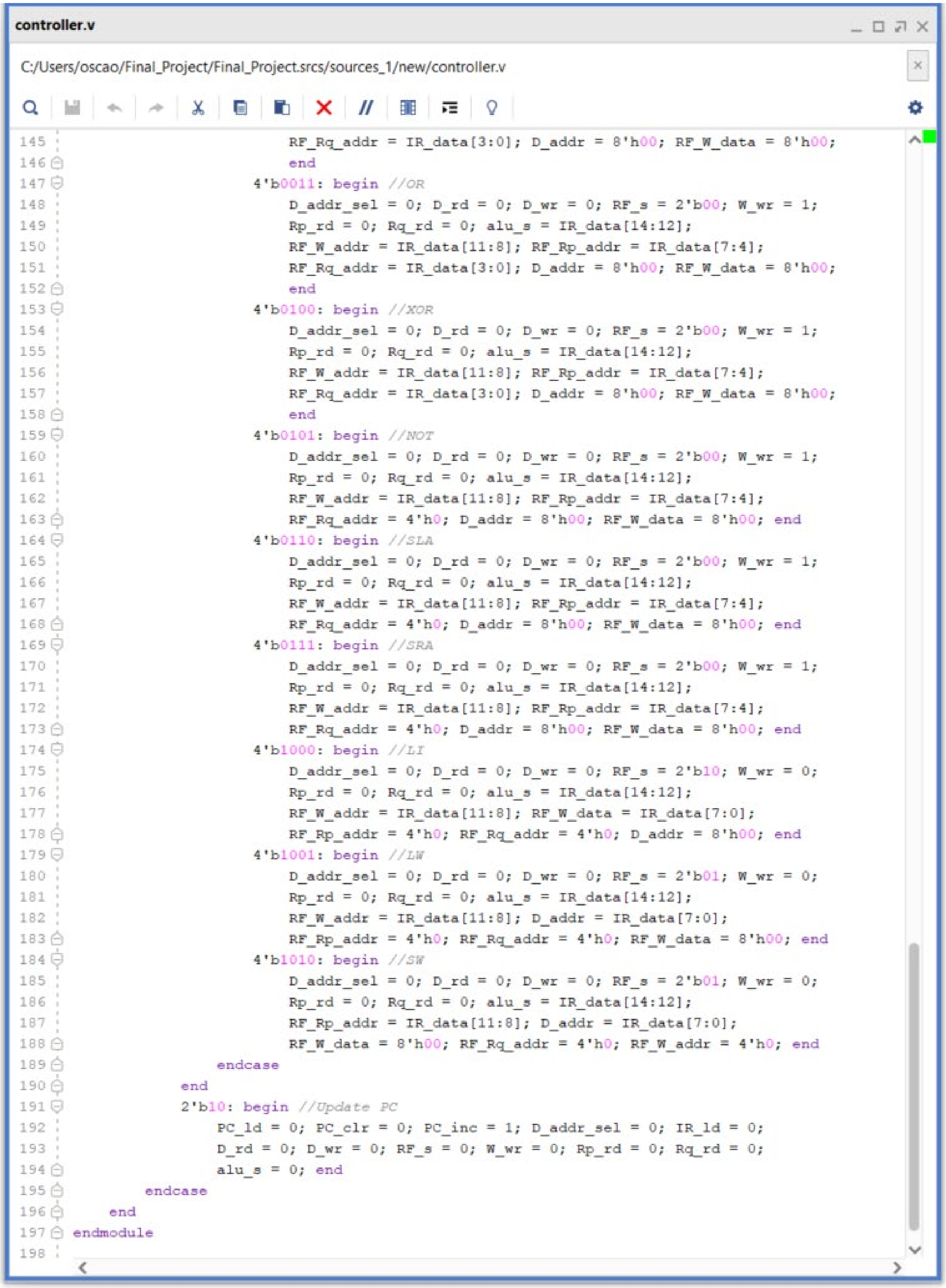
```
    always @(state) begin
        case (state)
            2'b00: nstate <= 2'b01;
            2'b01: nstate <= 2'b11;
            2'b11: nstate <= 2'b10;
            2'b10: nstate <= 2'b00;
         endcase
     end

     always @(state) begin
        case (state)
            2'b00: begin //Fetch
                D_addr_sel = 0; IR_ld = 1; D_rd = 1; D_wr = 0; PC_inc = 0; end
            2'b01: begin //Decode
                case (IR_data[15:12])
                    4'b0000: begin //ADD
                        D_addr_sel = 0; D_rd = 0; D_wr = 0; RF_s = 2'b00; W_wr = 0;
                        Rp_rd = 1; Rq_rd = 1; alu_s = IR_data[14:12];
                        RF_W_addr = IR_data[11:8]; RF_Rp_addr = IR_data[7:4];
                        RF_Rq_addr = IR_data[3:0]; D_addr = 8'h00; RF_W_data = 8'h00;
                        end
                    4'b0001: begin //SUB
                        D_addr_sel = 0; D_rd = 0; D_wr = 0; RF_s = 2'b00; W_wr = 0;
                        Rp_rd = 1; Rq_rd = 1; alu_s = IR_data[14:12];
                        RF_W_addr = IR_data[11:8]; RF_Rp_addr = IR_data[7:4];
                        RF_Rq_addr = IR_data[3:0]; D_addr = 8'h00; RF_W_data = 8'h00;
                        end
                    4'b0010: begin //AND
                        D_addr_sel = 0; D_rd = 0; D_wr = 0; RF_s = 2'b00; W_wr = 0;
                        Rp_rd = 1; Rq_rd = 1; alu_s = IR_data[14:12];
                        RF_W_addr = IR_data[11:8]; RF_Rp_addr = IR_data[7:4];
                        RF_Rq_addr = IR_data[3:0]; D_addr = 8'h00; RF_W_data = 8'h00;
                        end
                    4'b0011: begin //OR
                        D_addr_sel = 0; D_rd = 0; D_wr = 0; RF_s = 2'b00; W_wr = 0;
                        Rp_rd = 1; Rq_rd = 1; alu_s = IR_data[14:12];
                        RF_W_addr = IR_data[11:8]; RF_Rp_addr = IR_data[7:4];
                        RF_Rq_addr = IR_data[3:0]; D_addr = 8'h00; RF_W_data = 8'h00;
                        end
                    4'b0100: begin //XOR
                        D_addr_sel = 0; D_rd = 0; D_wr = 0; RF_s = 2'b00; W_wr = 0;
                        Rp_rd = 1; Rq_rd = 1; alu_s = IR_data[14:12];
                        RF_W_addr = IR_data[11:8]; RF_Rp_addr = IR_data[7:4];
                        RF_Rq_addr = IR_data[3:0]; D_addr = 8'h00; RF_W_data = 8'h00;
                        end
                    4'b0101: begin //NOT
                        D_addr_sel = 0; D_rd = 0; D_wr = 0; RF_s = 2'b00; W_wr = 0;
                        Rp_rd = 1; Rq_rd = 0; alu_s = IR_data[14:12];
                        RF_W_addr = IR_data[11:8]; RF_Rp_addr = IR_data[7:4];
                        RF_Rq_addr = 4'h00; D_addr = 8'h00; RF_W_data = 8'h00; end
                    4'b0110: begin //SLA
                        D_addr_sel = 0; D_rd = 0; D_wr = 0; RF_s = 2'b00; W_wr = 0;
                        Rp_rd = 1; Rq_rd = 0; alu_s = IR_data[14:12];
```



controller.v

C:/Users/oscao/Final_Project/Final_Project.srcs/sources_1/new/controller.v

```
                    4'b0110: begin //SLA
                        D_addr_sel = 0; D_rd = 0; D_wr = 0; RF_s = 2'b00; W_wr = 0;
                        Rp_rd = 1; Rq_rd = 0; alu_s = IR_data[14:12];
                        RF_W_addr = IR_data[11:8]; RF_Rp_addr = IR_data[7:4];
                        RF_Rq_addr = 4'h00; D_addr = 8'h00; RF_W_data = 8'h00; end
                    4'b0111: begin //SRA
                        D_addr_sel = 0; D_rd = 0; D_wr = 0; RF_s = 2'b00; W_wr = 0;
                        Rp_rd = 1; Rq_rd = 0; alu_s = IR_data[14:12];
                        RF_W_addr = IR_data[11:8]; RF_Rp_addr = IR_data[7:4];
                        RF_Rq_addr = 4'h00; D_addr = 8'h00; RF_W_data = 8'h00; end
                    4'b1000: begin //LI
                        D_addr_sel = 0; D_rd = 0; D_wr = 0; RF_s = 2'b10; W_wr = 1;
                        Rp_rd = 0; Rq_rd = 0; alu_s = IR_data[14:12];
                        RF_W_addr = IR_data[11:8]; RF_W_data = IR_data[7:0];
                        RF_Rp_addr = 4'h00; RF_Rq_addr = 4'h00; D_addr = 8'h00; end
                    4'b1001: begin //LW
                        D_addr_sel = 1; D_rd = 1; D_wr = 0; RF_s = 2'b01; W_wr = 1;
                        Rp_rd = 0; Rq_rd = 0; alu_s = IR_data[14:12];
                        RF_W_addr = IR_data[11:8]; D_addr = IR_data[7:0];
                        RF_Rp_addr = 4'h00; RF_Rq_addr = 4'h00; RF_W_data = 8'h00; end
                    4'b1010: begin //SW
                        D_addr_sel = 1; D_rd = 0; D_wr = 1; RF_s = 2'b01; W_wr = 0;
                        Rp_rd = 1; Rq_rd = 0; alu_s = IR_data[14:12];
                        RF_Rp_addr = IR_data[11:8]; D_addr = IR_data[7:0];
                        RF_W_data = 8'h00; RF_Rq_addr = 4'h00; RF_W_addr = 4'h00; end
                endcase
            end
            2'b11: begin //Execute
                case (IR_data[15:12])
                    4'b0000: begin //ADD
                        D_addr_sel = 0; D_rd = 0; D_wr = 0; RF_s = 2'b00; W_wr = 1;
                        Rp_rd = 0; Rq_rd = 0; alu_s = IR_data[14:12];
                        RF_W_addr = IR_data[11:8]; RF_Rp_addr = IR_data[7:4];
                        RF_Rq_addr = IR_data[3:0]; D_addr = 8'h00; RF_W_data = 8'h00;
                        end
                    4'b0001: begin //SUB
                        D_addr_sel = 0; D_rd = 0; D_wr = 0; RF_s = 2'b00; W_wr = 1;
                        Rp_rd = 0; Rq_rd = 0; alu_s = IR_data[14:12];
                        RF_W_addr = IR_data[11:8]; RF_Rp_addr = IR_data[7:4];
                        RF_Rq_addr = IR_data[3:0]; D_addr = 8'h00; RF_W_data = 8'h00;
                        end
                    4'b0010: begin //AND
                        D_addr_sel = 0; D_rd = 0; D_wr = 0; RF_s = 2'b00; W_wr = 1;
                        Rp_rd = 0; Rq_rd = 0; alu_s = IR_data[14:12];
                        RF_W_addr = IR_data[11:8]; RF_Rp_addr = IR_data[7:4];
                        RF_Rq_addr = IR_data[3:0]; D_addr = 8'h00; RF_W_data = 8'h00;
                        end
                    4'b0011: begin //OR
                        D_addr_sel = 0; D_rd = 0; D_wr = 0; RF_s = 2'b00; W_wr = 1;
                        Rp_rd = 0; Rq_rd = 0; alu_s = IR_data[14:12];
                        RF_W_addr = IR_data[11:8]; RF_Rp_addr = IR_data[7:4];
                        RF_Rq_addr = IR_data[3:0]; D_addr = 8'h00; RF_W_data = 8'h00;
                        end
                    4'b0100: begin //XOR
```

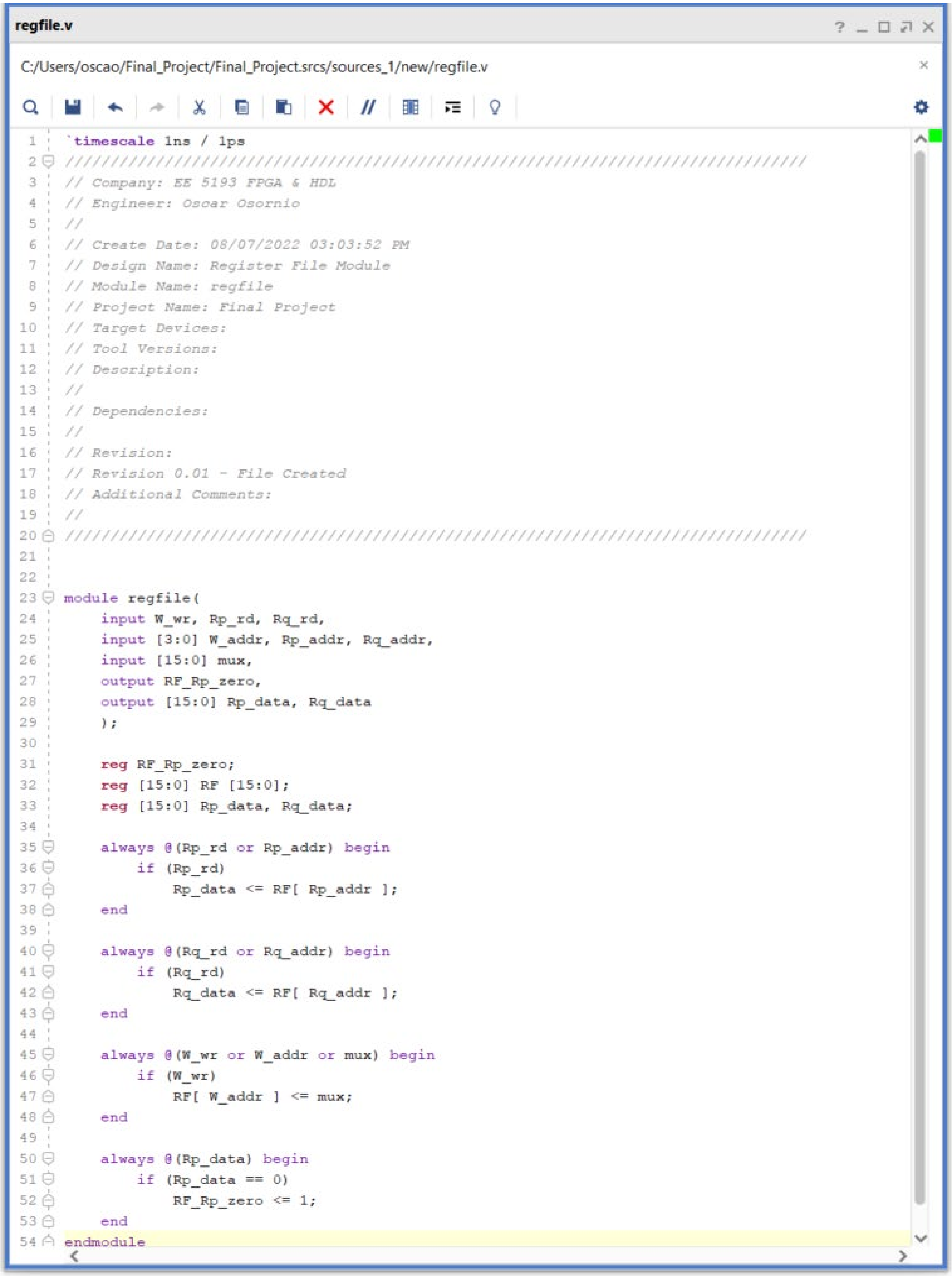


controller.v

C:/Users/oscao/Final_Project/Final_Project.srcs/sources_1/new/controller.v

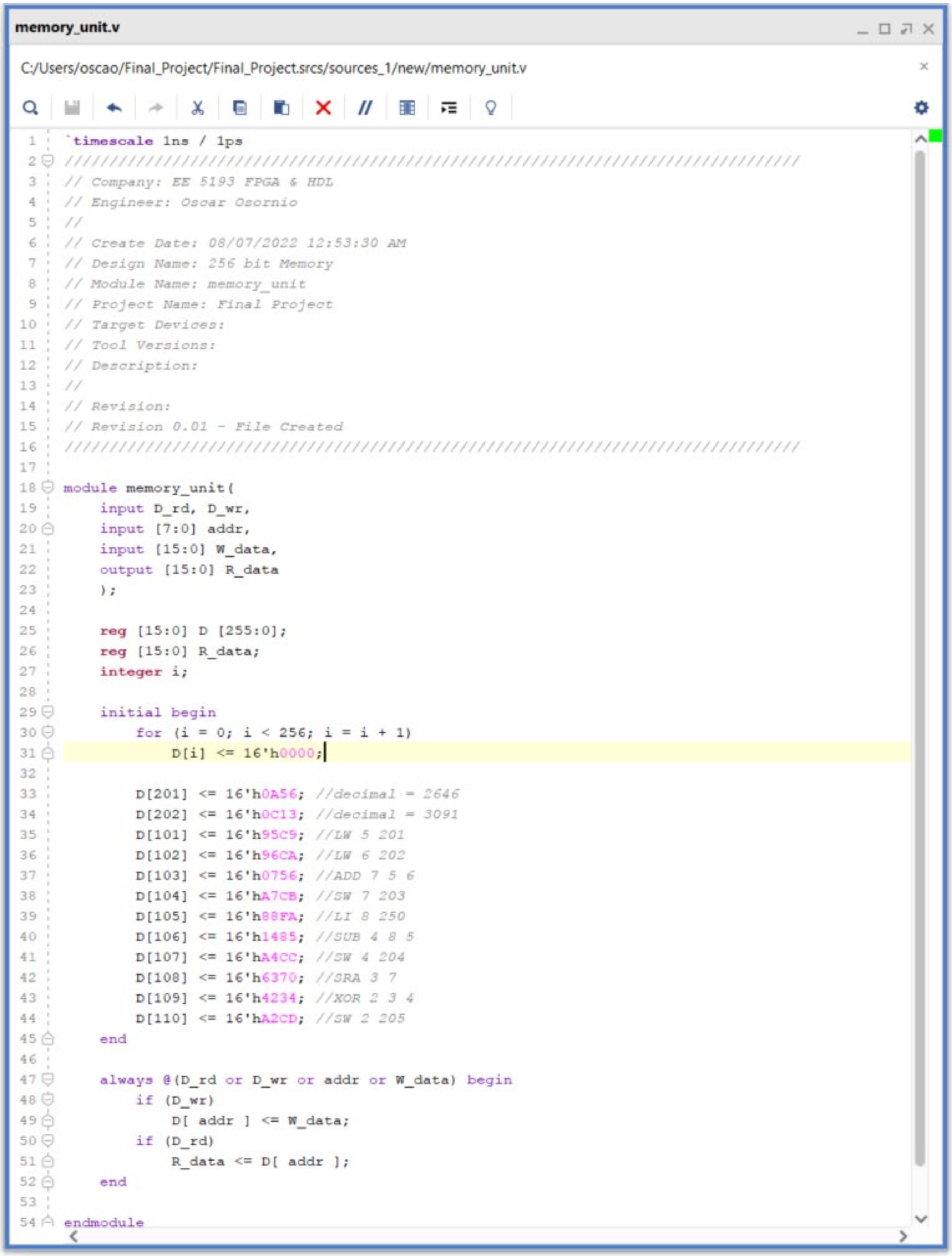
```
                        RF_Rq_addr = IR_data[3:0]; D_addr = 8'h00; RF_W_data = 8'h00;
                        end
                    4'b0011: begin //OR
                        D_addr_sel = 0; D_rd = 0; D_wr = 0; RF_s = 2'b00; W_wr = 1;
                        Rp_rd = 0; Rq_rd = 0; alu_s = IR_data[14:12];
                        RF_W_addr = IR_data[11:8]; RF_Rp_addr = IR_data[7:4];
                        RF_Rq_addr = IR_data[3:0]; D_addr = 8'h00; RF_W_data = 8'h00;
                        end
                    4'b0100: begin //XOR
                        D_addr_sel = 0; D_rd = 0; D_wr = 0; RF_s = 2'b00; W_wr = 1;
                        Rp_rd = 0; Rq_rd = 0; alu_s = IR_data[14:12];
                        RF_W_addr = IR_data[11:8]; RF_Rp_addr = IR_data[7:4];
                        RF_Rq_addr = IR_data[3:0]; D_addr = 8'h00; RF_W_data = 8'h00;
                        end
                    4'b0101: begin //NOT
                        D_addr_sel = 0; D_rd = 0; D_wr = 0; RF_s = 2'b00; W_wr = 1;
                        Rp_rd = 0; Rq_rd = 0; alu_s = IR_data[14:12];
                        RF_W_addr = IR_data[11:8]; RF_Rp_addr = IR_data[7:4];
                        RF_Rq_addr = 4'h0; D_addr = 8'h00; RF_W_data = 8'h00; end
                    4'b0110: begin //SLA
                        D_addr_sel = 0; D_rd = 0; D_wr = 0; RF_s = 2'b00; W_wr = 1;
                        Rp_rd = 0; Rq_rd = 0; alu_s = IR_data[14:12];
                        RF_W_addr = IR_data[11:8]; RF_Rp_addr = IR_data[7:4];
                        RF_Rq_addr = 4'h0; D_addr = 8'h00; RF_W_data = 8'h00; end
                    4'b0111: begin //SRA
                        D_addr_sel = 0; D_rd = 0; D_wr = 0; RF_s = 2'b00; W_wr = 1;
                        Rp_rd = 0; Rq_rd = 0; alu_s = IR_data[14:12];
                        RF_W_addr = IR_data[11:8]; RF_Rp_addr = IR_data[7:4];
                        RF_Rq_addr = 4'h0; D_addr = 8'h00; RF_W_data = 8'h00; end
                    4'b1000: begin //LI
                        D_addr_sel = 0; D_rd = 0; D_wr = 0; RF_s = 2'b10; W_wr = 0;
                        Rp_rd = 0; Rq_rd = 0; alu_s = IR_data[14:12];
                        RF_W_addr = IR_data[11:8]; RF_W_data = IR_data[7:0];
                        RF_Rp_addr = 4'h0; RF_Rq_addr = 4'h0; D_addr = 8'h00; end
                    4'b1001: begin //LW
                        D_addr_sel = 0; D_rd = 0; D_wr = 0; RF_s = 2'b01; W_wr = 0;
                        Rp_rd = 0; Rq_rd = 0; alu_s = IR_data[14:12];
                        RF_W_addr = IR_data[11:8]; D_addr = IR_data[7:0];
                        RF_Rp_addr = 4'h0; RF_Rq_addr = 4'h0; RF_W_data = 8'h00; end
                    4'b1010: begin //SW
                        D_addr_sel = 0; D_rd = 0; D_wr = 0; RF_s = 2'b01; W_wr = 0;
                        Rp_rd = 0; Rq_rd = 0; alu_s = IR_data[14:12];
                        RF_Rp_addr = IR_data[11:8]; D_addr = IR_data[7:0];
                        RF_W_data = 8'h00; RF_Rq_addr = 4'h0; RF_W_addr = 4'h0; end
                endcase
            end
            2'b10: begin //Update PC
                PC_ld = 0; PC_clr = 0; PC_inc = 1; D_addr_sel = 0; IR_ld = 0;
                D_rd = 0; D_wr = 0; RF_s = 0; W_wr = 0; Rp_rd = 0; Rq_rd = 0;
                alu_s = 0; end
        endcase
    end
endmodule
```

Regfile:


regfile.v
C:/Users/oscao/Final_Project/Final_Project.srcs/sources_1/new/regfile.v

```verilog
`timescale 1ns / 1ps
//////////////////////////////////////////////////////////////////////////////////
// Company: EE 5193 FPGA & HDL
// Engineer: Oscar Osornio
//
// Create Date: 08/07/2022 03:03:52 PM
// Design Name: Register File Module
// Module Name: regfile
// Project Name: Final Project
// Target Devices:
// Tool Versions:
// Description:
//
// Dependencies:
//
// Revision:
// Revision 0.01 - File Created
// Additional Comments:
//
//////////////////////////////////////////////////////////////////////////////////


module regfile(
    input W_wr, Rp_rd, Rq_rd,
    input [3:0] W_addr, Rp_addr, Rq_addr,
    input [15:0] mux,
    output RF_Rp_zero,
    output [15:0] Rp_data, Rq_data
    );

    reg RF_Rp_zero;
    reg [15:0] RF [15:0];
    reg [15:0] Rp_data, Rq_data;

    always @(Rp_rd or Rp_addr) begin
        if (Rp_rd)
            Rp_data <= RF[ Rp_addr ];
    end

    always @(Rq_rd or Rq_addr) begin
        if (Rq_rd)
            Rq_data <= RF[ Rq_addr ];
    end

    always @(W_wr or W_addr or mux) begin
        if (W_wr)
            RF[ W_addr ] <= mux;
    end

    always @(Rp_data) begin
        if (Rp_data == 0)
            RF_Rp_zero <= 1;
    end
endmodule
```

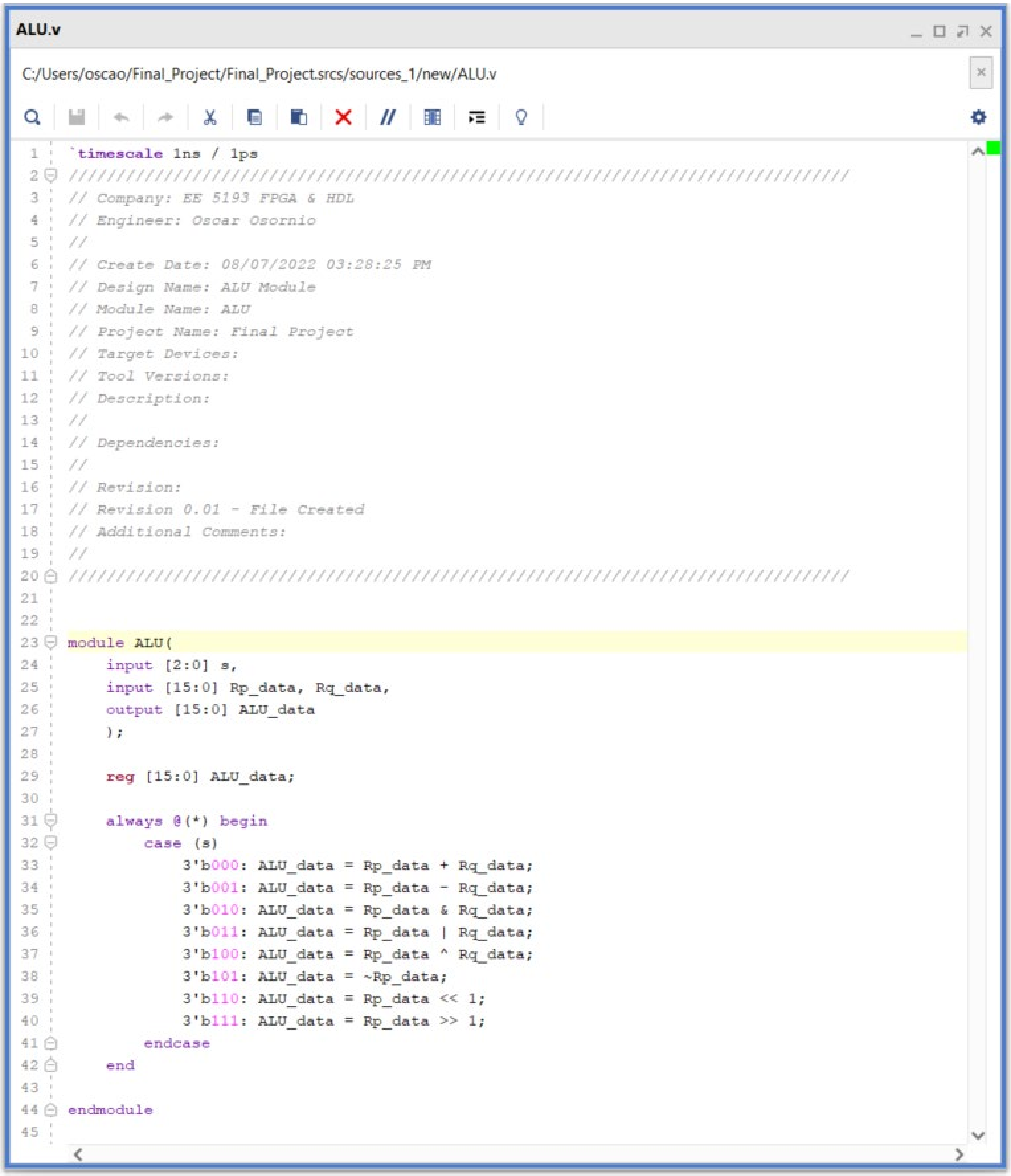
Memory:



C:/Users/oscao/Final_Project/Final_Project.srcs/sources_1/new/memory_unit.v

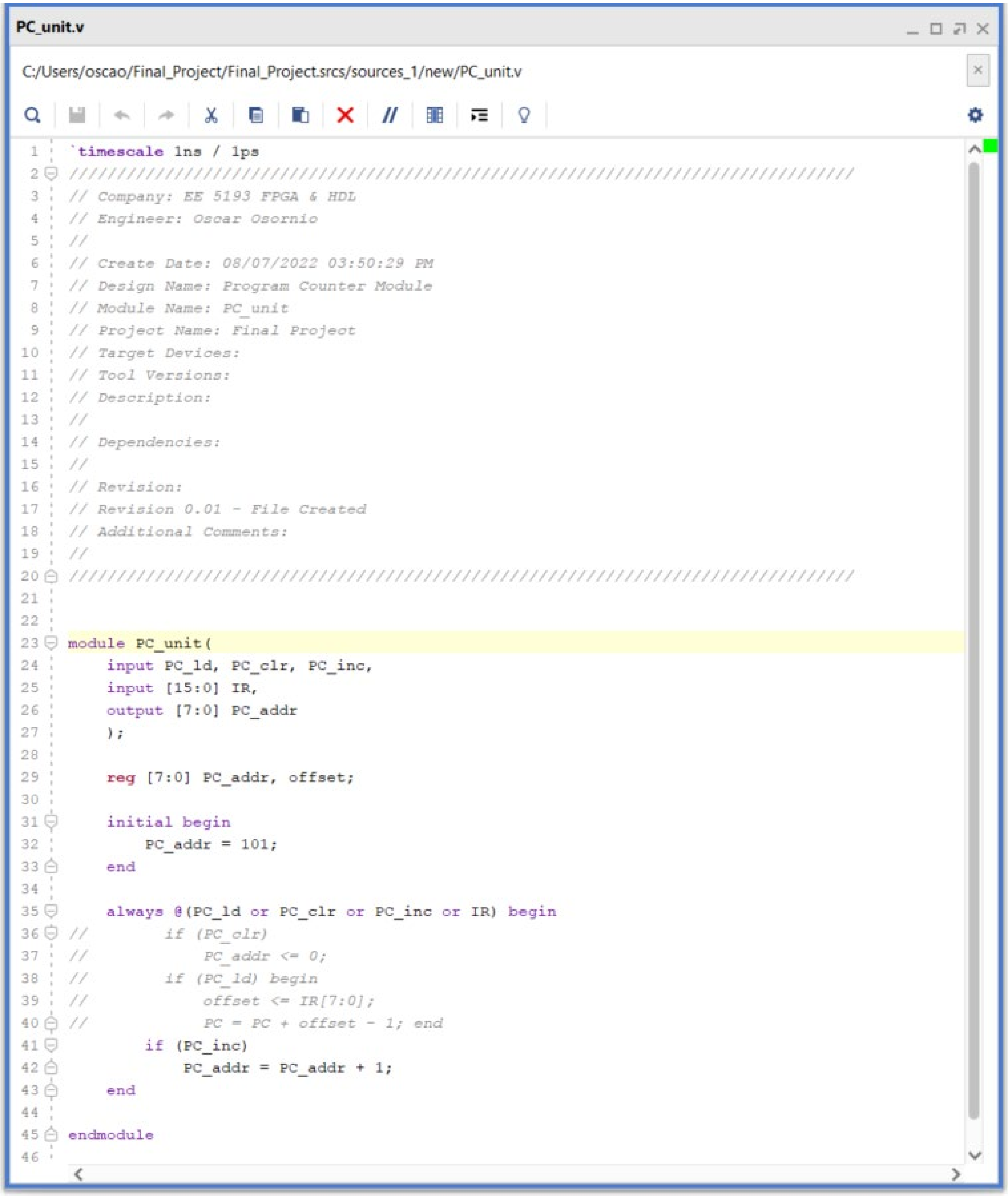
```verilog
`timescale 1ns / 1ps
//////////////////////////////////////////////////////////////////////////////////
// Company: EE 5193 FPGA & HDL
// Engineer: Oscar Osornio
//
// Create Date: 08/07/2022 12:53:30 AM
// Design Name: 256 bit Memory
// Module Name: memory_unit
// Project Name: Final Project
// Target Devices:
// Tool Versions:
// Description:
//
// Revision:
// Revision 0.01 - File Created
//////////////////////////////////////////////////////////////////////////////////

module memory_unit(
    input D_rd, D_wr,
    input [7:0] addr,
    input [15:0] W_data,
    output [15:0] R_data
    );

    reg [15:0] D [255:0];
    reg [15:0] R_data;
    integer i;

    initial begin
        for (i = 0; i < 256; i = i + 1)
            D[i] <= 16'h0000;

        D[201] <= 16'h0A56; //decimal = 2646
        D[202] <= 16'h0C13; //decimal = 3091
        D[101] <= 16'h95C9; //LW 5 201
        D[102] <= 16'h96CA; //LW 6 202
        D[103] <= 16'h0756; //ADD 7 5 6
        D[104] <= 16'hA7CB; //SW 7 203
        D[105] <= 16'h88FA; //LI 8 250
        D[106] <= 16'h1485; //SUB 4 8 5
        D[107] <= 16'hA4CC; //SW 4 204
        D[108] <= 16'h6370; //SRA 3 7
        D[109] <= 16'h4234; //XOR 2 3 4
        D[110] <= 16'hA2CD; //SW 2 205
    end

    always @(D_rd or D_wr or addr or W_data) begin
        if (D_wr)
            D[ addr ] <= W_data;
        if (D_rd)
            R_data <= D[ addr ];
    end

endmodule
```

ALU:


ALU.v
C:/Users/oscao/Final_Project/Final_Project.srcs/sources_1/new/ALU.v

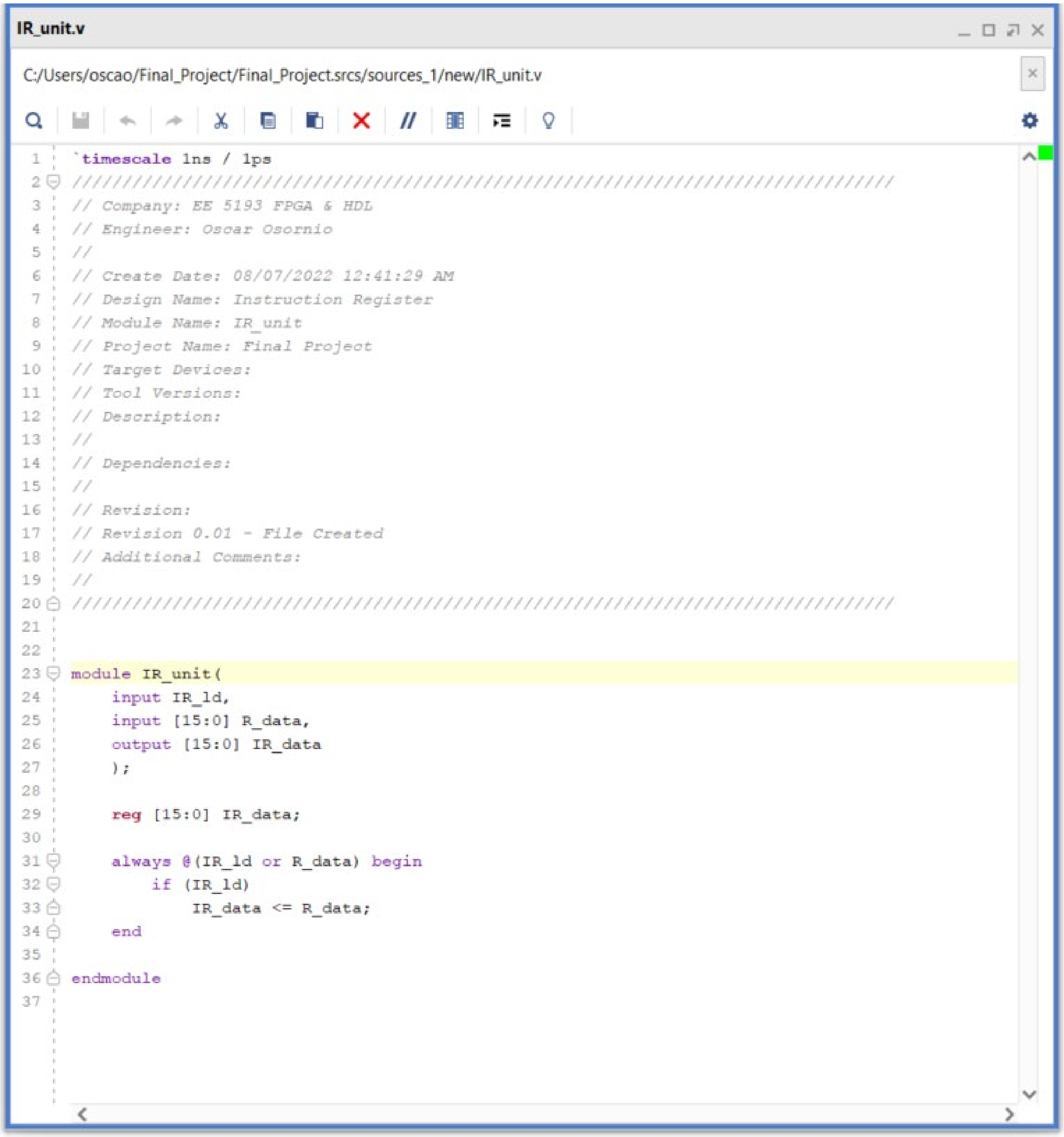
```verilog
`timescale 1ns / 1ps
//////////////////////////////////////////////////////////////////////////////////
// Company: EE 5193 FPGA & HDL
// Engineer: Oscar Osornio
//
// Create Date: 08/07/2022 03:28:25 PM
// Design Name: ALU Module
// Module Name: ALU
// Project Name: Final Project
// Target Devices:
// Tool Versions:
// Description:
//
// Dependencies:
//
// Revision:
// Revision 0.01 - File Created
// Additional Comments:
//
//////////////////////////////////////////////////////////////////////////////////


module ALU(
    input [2:0] s,
    input [15:0] Rp_data, Rq_data,
    output [15:0] ALU_data
    );

    reg [15:0] ALU_data;

    always @(*) begin
        case (s)
            3'b000: ALU_data = Rp_data + Rq_data;
            3'b001: ALU_data = Rp_data - Rq_data;
            3'b010: ALU_data = Rp_data & Rq_data;
            3'b011: ALU_data = Rp_data | Rq_data;
            3'b100: ALU_data = Rp_data ^ Rq_data;
            3'b101: ALU_data = ~Rp_data;
            3'b110: ALU_data = Rp_data << 1;
            3'b111: ALU_data = Rp_data >> 1;
        endcase
    end

endmodule

```

Program Counter:


PC_unit.v
C:/Users/oscao/Final_Project/Final_Project.srcs/sources_1/new/PC_unit.v

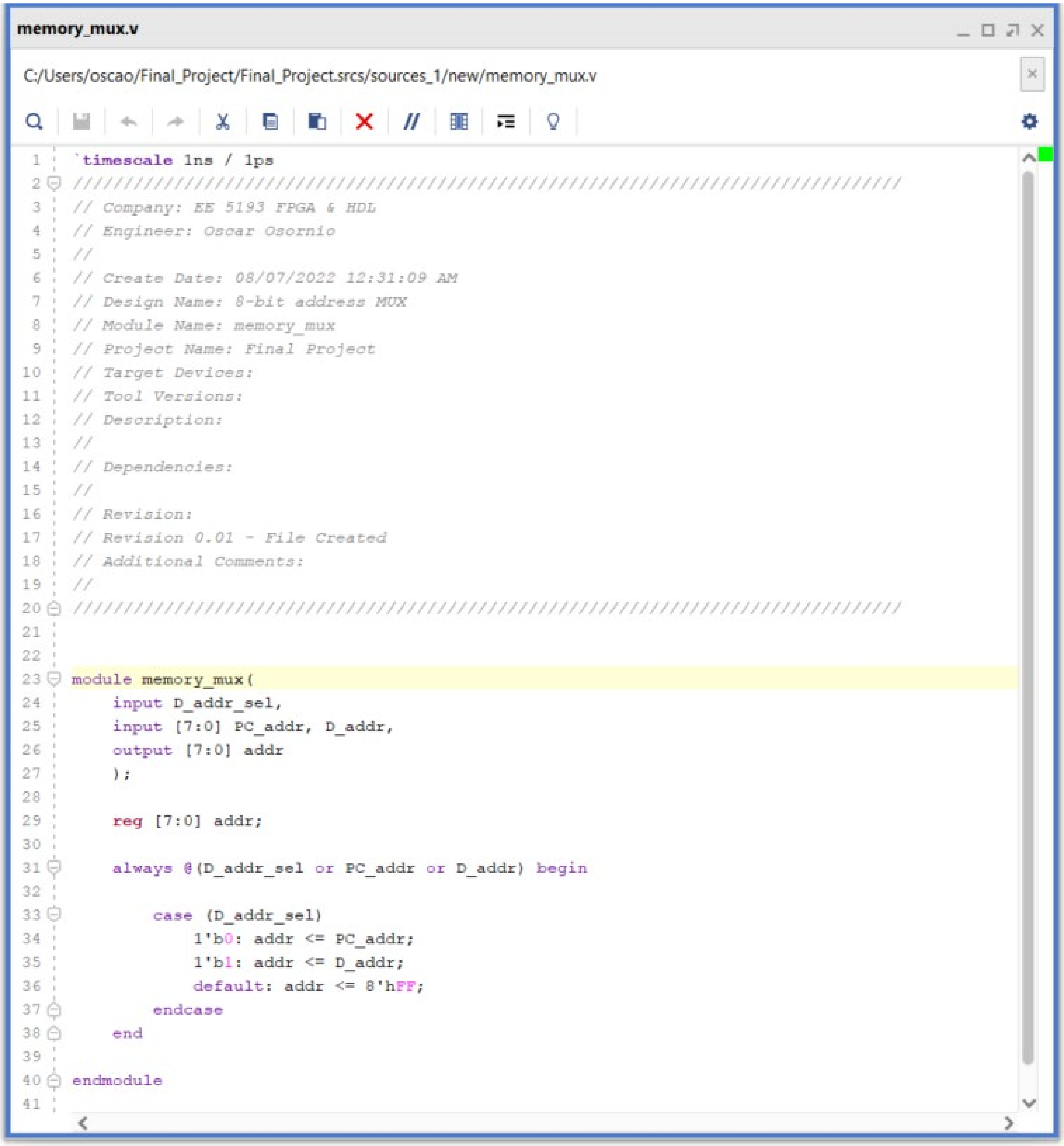
```
`timescale 1ns / 1ps
//////////////////////////////////////////////////////////////////////////////////
// Company: EE 5193 FPGA & HDL
// Engineer: Oscar Osornio
//
// Create Date: 08/07/2022 03:50:29 PM
// Design Name: Program Counter Module
// Module Name: PC_unit
// Project Name: Final Project
// Target Devices:
// Tool Versions:
// Description:
//
// Dependencies:
//
// Revision:
// Revision 0.01 - File Created
// Additional Comments:
//
//////////////////////////////////////////////////////////////////////////////////


module PC_unit(
    input PC_ld, PC_clr, PC_inc,
    input [15:0] IR,
    output [7:0] PC_addr
    );

    reg [7:0] PC_addr, offset;

    initial begin
        PC_addr = 101;
    end

    always @(PC_ld or PC_clr or PC_inc or IR) begin
//        if (PC_clr)
//            PC_addr <= 0;
//        if (PC_ld) begin
//            offset <= IR[7:0];
//            PC = PC + offset - 1; end
        if (PC_inc)
            PC_addr = PC_addr + 1;
    end

endmodule

```

Instruction Register Unit:


IR_unit.v
C:/Users/oscao/Final_Project/Final_Project.srcs/sources_1/new/IR_unit.v

```
`timescale 1ns / 1ps
//////////////////////////////////////////////////////////////////////////////////
// Company: EE 5193 FPGA & HDL
// Engineer: Oscar Osornio
//
// Create Date: 08/07/2022 12:41:29 AM
// Design Name: Instruction Register
// Module Name: IR_unit
// Project Name: Final Project
// Target Devices:
// Tool Versions:
// Description:
//
// Dependencies:
//
// Revision:
// Revision 0.01 - File Created
// Additional Comments:
//
//////////////////////////////////////////////////////////////////////////////////


module IR_unit(
    input IR_ld,
    input [15:0] R_data,
    output [15:0] IR_data
    );

    reg [15:0] IR_data;

    always @(IR_ld or R_data) begin
        if (IR_ld)
            IR_data <= R_data;
    end

endmodule
```

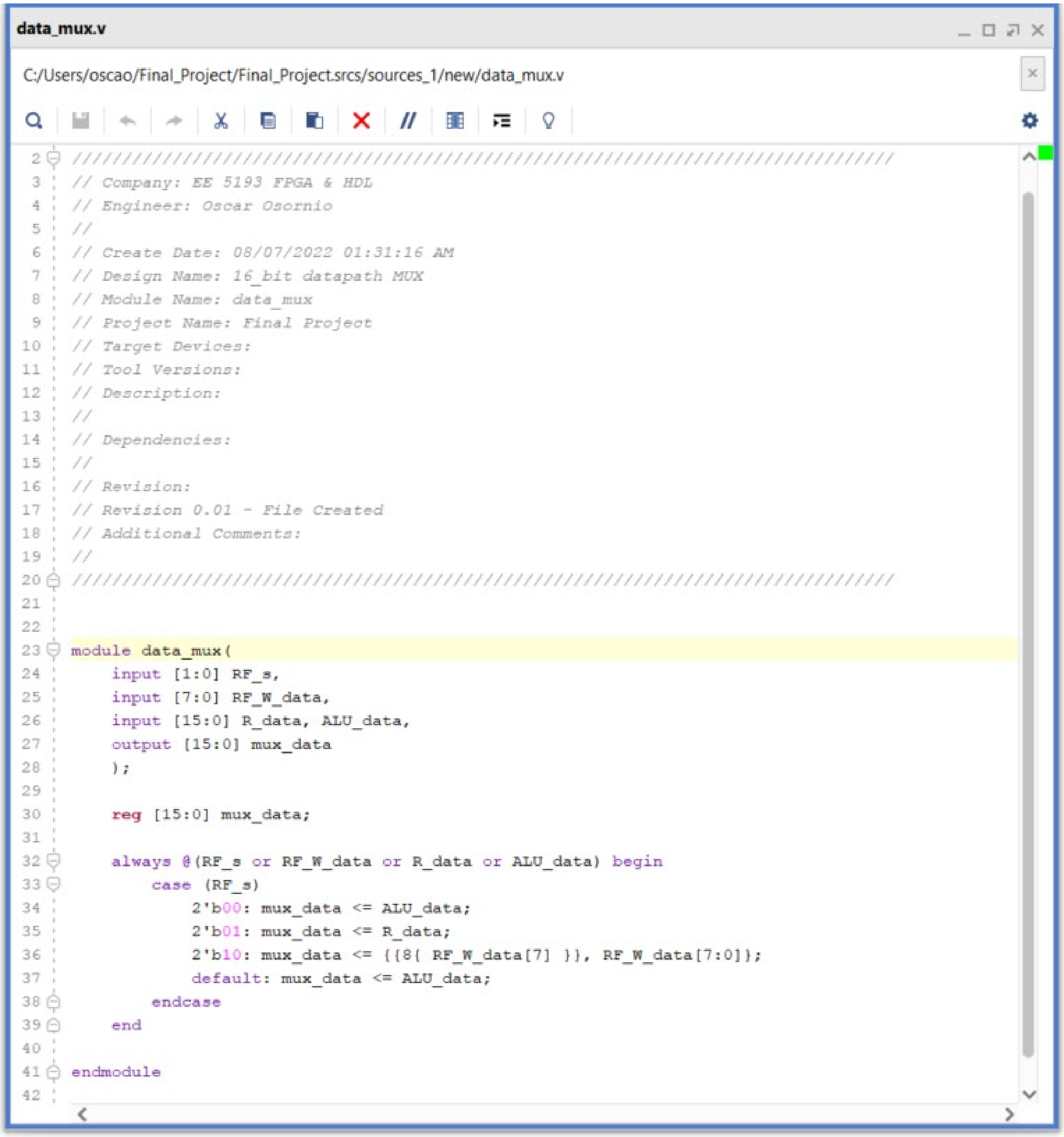
Memory MUX:



C:/Users/oscao/Final_Project/Final_Project.srcs/sources_1/new/memory_mux.v

```
`timescale 1ns / 1ps
//////////////////////////////////////////////////////////////////////////////////
// Company: EE 5193 FPGA & HDL
// Engineer: Oscar Osornio
//
// Create Date: 08/07/2022 12:31:09 AM
// Design Name: 8-bit address MUX
// Module Name: memory_mux
// Project Name: Final Project
// Target Devices:
// Tool Versions:
// Description:
//
// Dependencies:
//
// Revision:
// Revision 0.01 - File Created
// Additional Comments:
//
//////////////////////////////////////////////////////////////////////////////////


module memory_mux(
    input D_addr_sel,
    input [7:0] PC_addr, D_addr,
    output [7:0] addr
    );

    reg [7:0] addr;

    always @(D_addr_sel or PC_addr or D_addr) begin

        case (D_addr_sel)
            1'b0: addr <= PC_addr;
            1'b1: addr <= D_addr;
            default: addr <= 8'hFF;
        endcase
    end

endmodule

```

Data MUX:

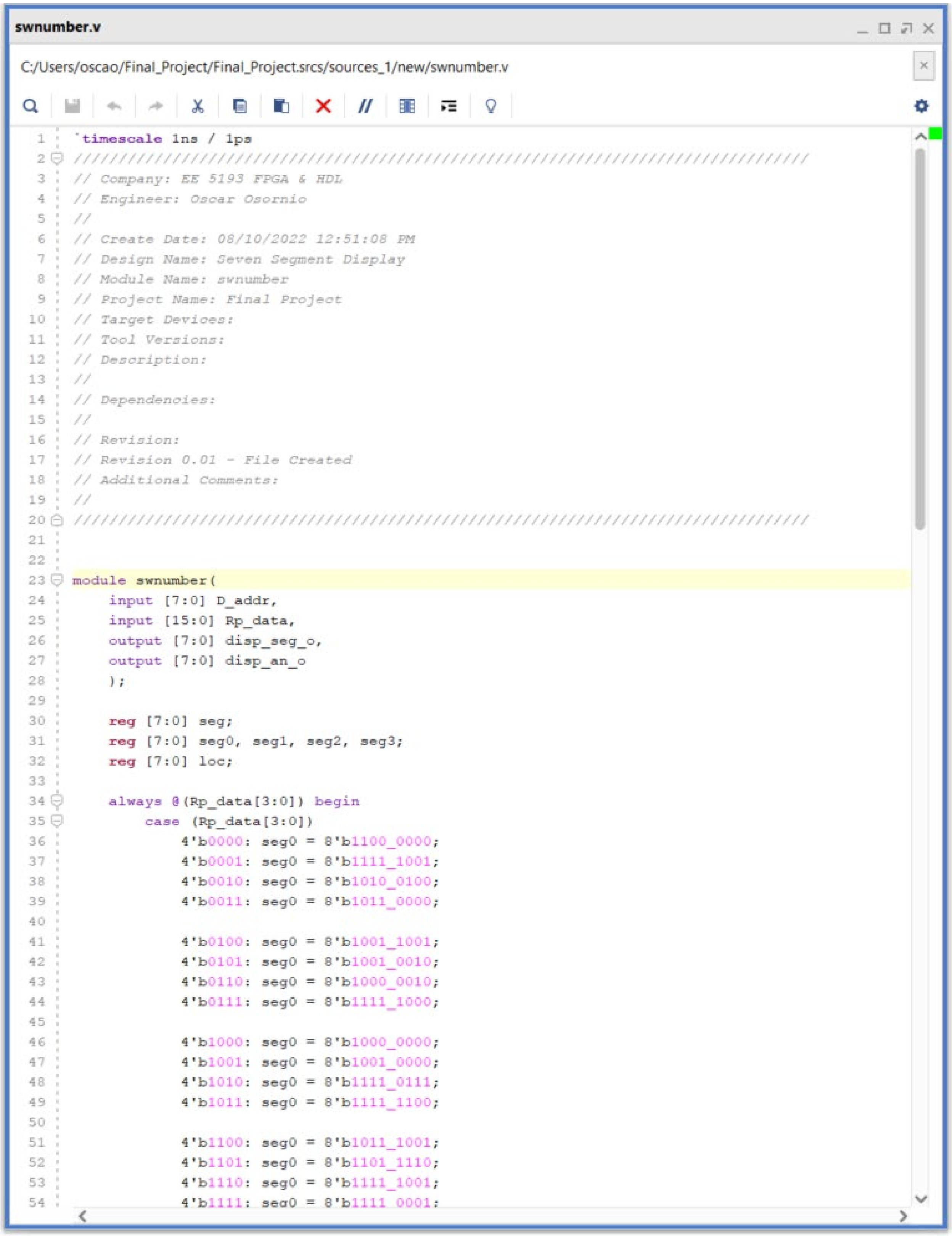


data_mux.v

C:/Users/oscao/Final_Project/Final_Project.srcs/sources_1/new/data_mux.v

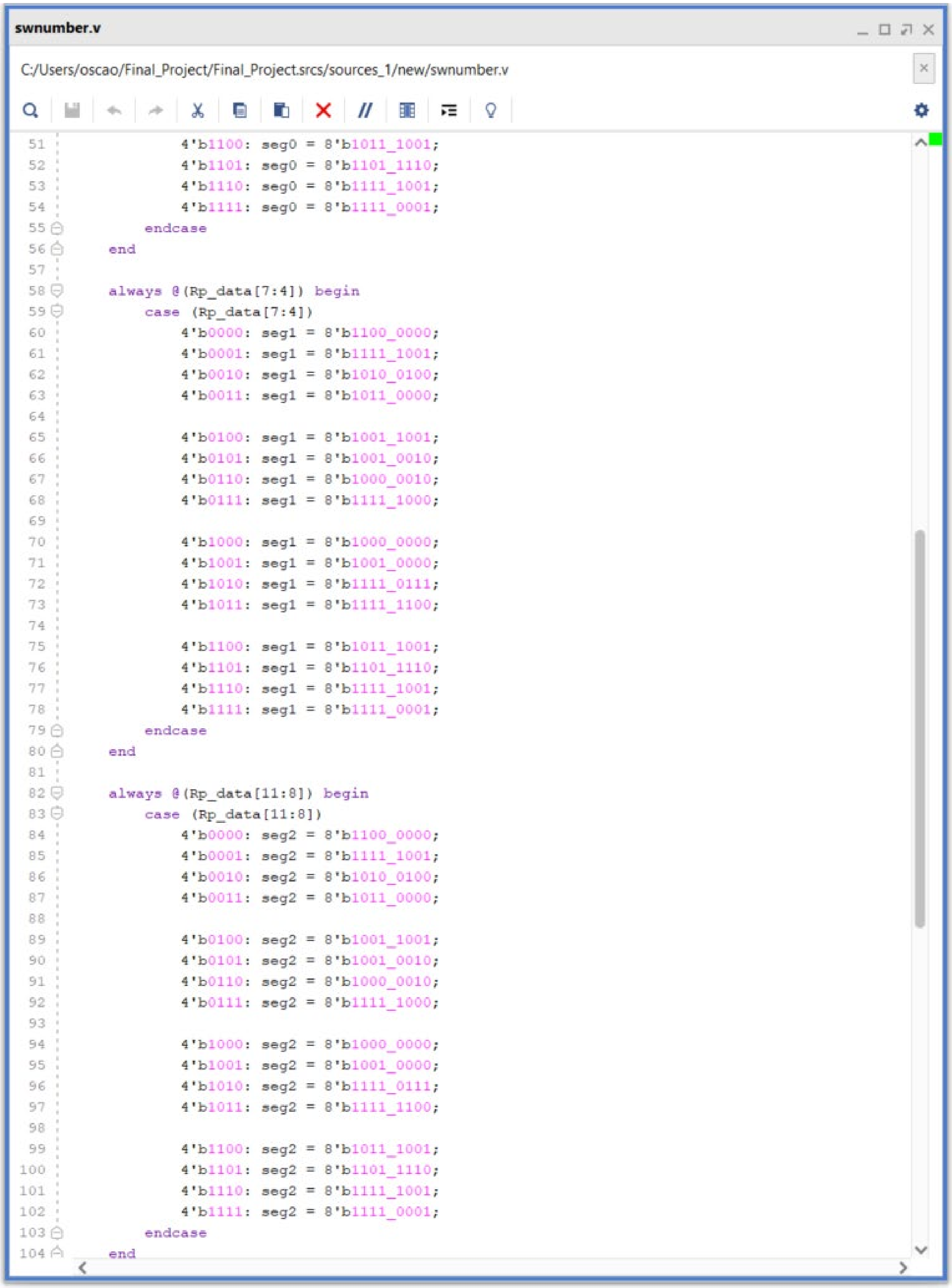
```
//////////////////////////////////////////////////////////////////////////////////
// Company: EE 5193 FPGA & HDL
// Engineer: Oscar Osornio
//
// Create Date: 08/07/2022 01:31:16 AM
// Design Name: 16_bit datapath MUX
// Module Name: data_mux
// Project Name: Final Project
// Target Devices:
// Tool Versions:
// Description:
//
// Dependencies:
//
// Revision:
// Revision 0.01 - File Created
// Additional Comments:
//
//////////////////////////////////////////////////////////////////////////////////


module data_mux(
    input [1:0] RF_s,
    input [7:0] RF_W_data,
    input [15:0] R_data, ALU_data,
    output [15:0] mux_data
    );

    reg [15:0] mux_data;

    always @(RF_s or RF_W_data or R_data or ALU_data) begin
        case (RF_s)
            2'b00: mux_data <= ALU_data;
            2'b01: mux_data <= R_data;
            2'b10: mux_data <= {{8{ RF_W_data[7] }}, RF_W_data[7:0]};
            default: mux_data <= ALU_data;
        endcase
    end

endmodule

```

Seven Segment Display:



swnumber.v
C:/Users/oscao/Final_Project/Final_Project.srcs/sources_1/new/swnumber.v

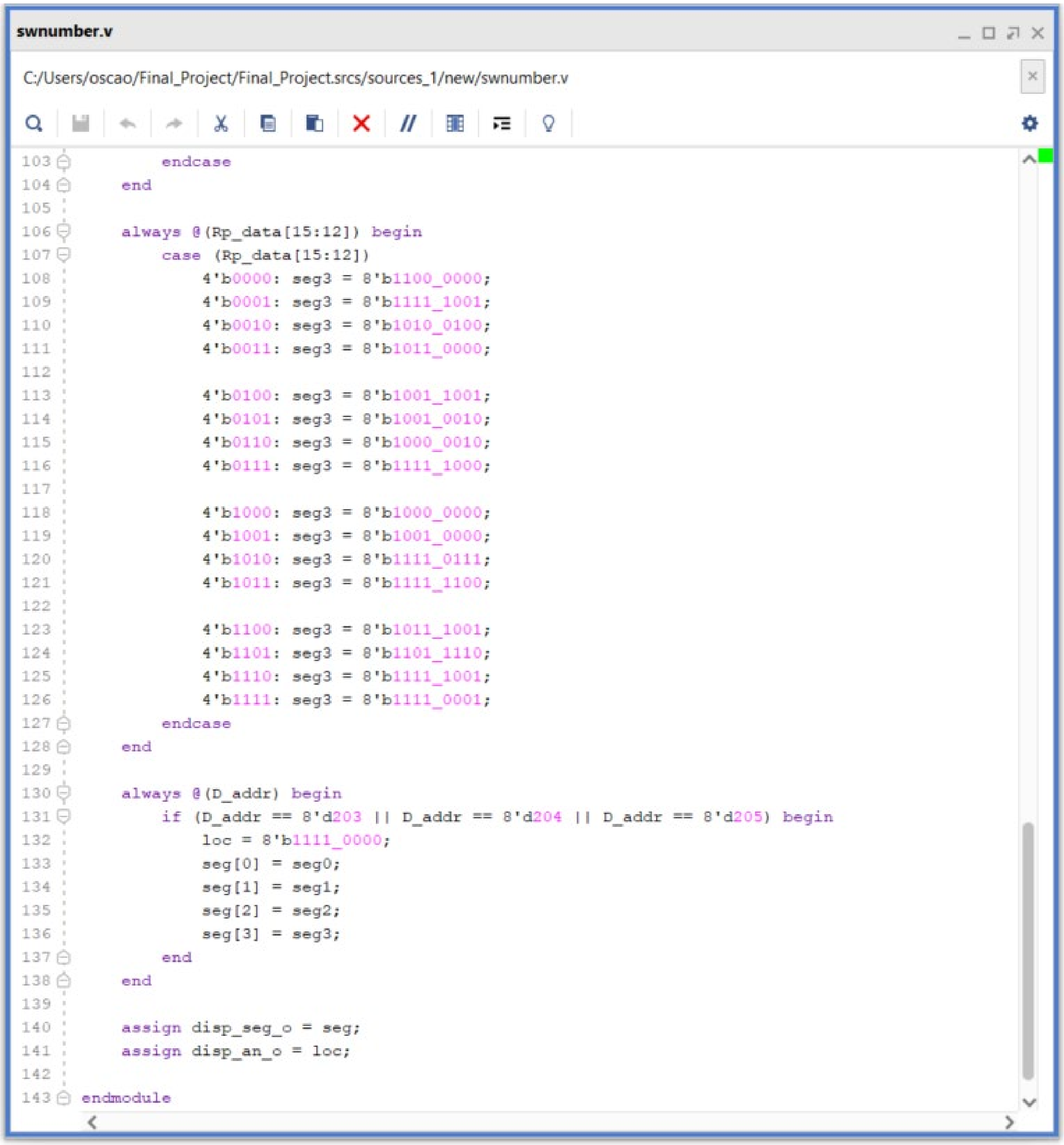
```verilog
`timescale 1ns / 1ps
//////////////////////////////////////////////////////////////////////////////////
// Company: EE 5193 FPGA & HDL
// Engineer: Oscar Osornio
//
// Create Date: 08/10/2022 12:51:08 PM
// Design Name: Seven Segment Display
// Module Name: swnumber
// Project Name: Final Project
// Target Devices:
// Tool Versions:
// Description:
//
// Dependencies:
//
// Revision:
// Revision 0.01 - File Created
// Additional Comments:
//
//////////////////////////////////////////////////////////////////////////////////


module swnumber(
    input [7:0] D_addr,
    input [15:0] Rp_data,
    output [7:0] disp_seg_o,
    output [7:0] disp_an_o
    );

    reg [7:0] seg;
    reg [7:0] seg0, seg1, seg2, seg3;
    reg [7:0] loc;

    always @(Rp_data[3:0]) begin
        case (Rp_data[3:0])
            4'b0000: seg0 = 8'b1100_0000;
            4'b0001: seg0 = 8'b1111_1001;
            4'b0010: seg0 = 8'b1010_0100;
            4'b0011: seg0 = 8'b1011_0000;

            4'b0100: seg0 = 8'b1001_1001;
            4'b0101: seg0 = 8'b1001_0010;
            4'b0110: seg0 = 8'b1000_0010;
            4'b0111: seg0 = 8'b1111_1000;

            4'b1000: seg0 = 8'b1000_0000;
            4'b1001: seg0 = 8'b1001_0000;
            4'b1010: seg0 = 8'b1111_0111;
            4'b1011: seg0 = 8'b1111_1100;

            4'b1100: seg0 = 8'b1011_1001;
            4'b1101: seg0 = 8'b1101_1110;
            4'b1110: seg0 = 8'b1111_1001;
            4'b1111: seg0 = 8'b1111_0001;
```



swnumber.v

C:/Users/oscao/Final_Project/Final_Project.srcs/sources_1/new/swnumber.v

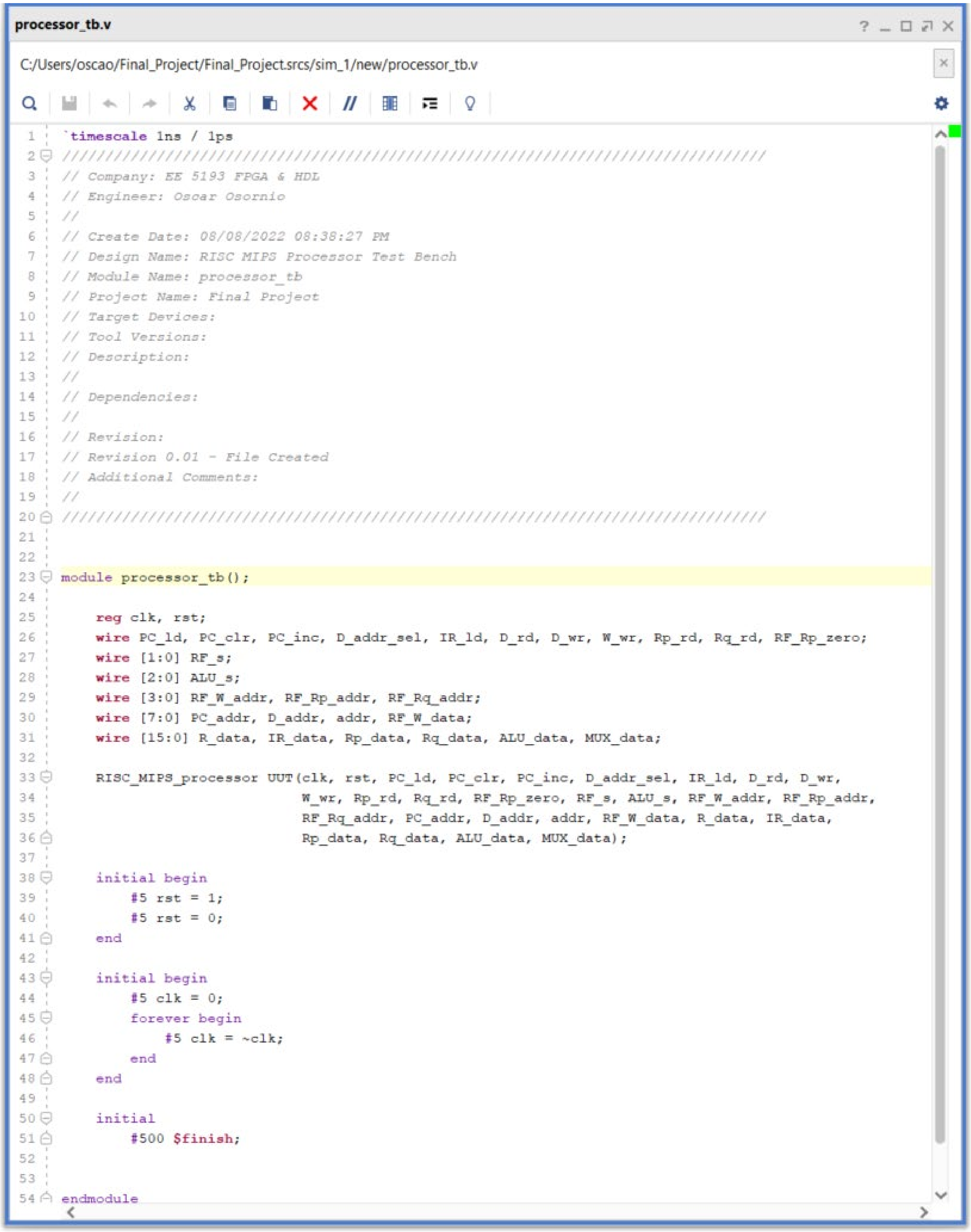
```
                4'b1100: seg0 = 8'b1011_1001;
                4'b1101: seg0 = 8'b1101_1110;
                4'b1110: seg0 = 8'b1111_1001;
                4'b1111: seg0 = 8'b1111_0001;
            endcase
        end

        always @(Rp_data[7:4]) begin
            case (Rp_data[7:4])
                4'b0000: seg1 = 8'b1100_0000;
                4'b0001: seg1 = 8'b1111_1001;
                4'b0010: seg1 = 8'b1010_0100;
                4'b0011: seg1 = 8'b1011_0000;

                4'b0100: seg1 = 8'b1001_1001;
                4'b0101: seg1 = 8'b1001_0010;
                4'b0110: seg1 = 8'b1000_0010;
                4'b0111: seg1 = 8'b1111_1000;

                4'b1000: seg1 = 8'b1000_0000;
                4'b1001: seg1 = 8'b1001_0000;
                4'b1010: seg1 = 8'b1111_0111;
                4'b1011: seg1 = 8'b1111_1100;

                4'b1100: seg1 = 8'b1011_1001;
                4'b1101: seg1 = 8'b1101_1110;
                4'b1110: seg1 = 8'b1111_1001;
                4'b1111: seg1 = 8'b1111_0001;
            endcase
        end

        always @(Rp_data[11:8]) begin
            case (Rp_data[11:8])
                4'b0000: seg2 = 8'b1100_0000;
                4'b0001: seg2 = 8'b1111_1001;
                4'b0010: seg2 = 8'b1010_0100;
                4'b0011: seg2 = 8'b1011_0000;

                4'b0100: seg2 = 8'b1001_1001;
                4'b0101: seg2 = 8'b1001_0010;
                4'b0110: seg2 = 8'b1000_0010;
                4'b0111: seg2 = 8'b1111_1000;

                4'b1000: seg2 = 8'b1000_0000;
                4'b1001: seg2 = 8'b1001_0000;
                4'b1010: seg2 = 8'b1111_0111;
                4'b1011: seg2 = 8'b1111_1100;

                4'b1100: seg2 = 8'b1011_1001;
                4'b1101: seg2 = 8'b1101_1110;
                4'b1110: seg2 = 8'b1111_1001;
                4'b1111: seg2 = 8'b1111_0001;
            endcase
        end
```



C:/Users/oscao/Final_Project/Final_Project.srcs/sources_1/new/swnumber.v

```verilog
        endcase
    end

    always @(Rp_data[15:12]) begin
        case (Rp_data[15:12])
            4'b0000: seg3 = 8'b1100_0000;
            4'b0001: seg3 = 8'b1111_1001;
            4'b0010: seg3 = 8'b1010_0100;
            4'b0011: seg3 = 8'b1011_0000;

            4'b0100: seg3 = 8'b1001_1001;
            4'b0101: seg3 = 8'b1001_0010;
            4'b0110: seg3 = 8'b1000_0010;
            4'b0111: seg3 = 8'b1111_1000;

            4'b1000: seg3 = 8'b1000_0000;
            4'b1001: seg3 = 8'b1001_0000;
            4'b1010: seg3 = 8'b1111_0111;
            4'b1011: seg3 = 8'b1111_1100;

            4'b1100: seg3 = 8'b1011_1001;
            4'b1101: seg3 = 8'b1101_1110;
            4'b1110: seg3 = 8'b1111_1001;
            4'b1111: seg3 = 8'b1111_0001;
        endcase
    end

    always @(D_addr) begin
        if (D_addr == 8'd203 || D_addr == 8'd204 || D_addr == 8'd205) begin
            loc = 8'b1111_0000;
            seg[0] = seg0;
            seg[1] = seg1;
            seg[2] = seg2;
            seg[3] = seg3;
        end
    end

    assign disp_seg_o = seg;
    assign disp_an_o = loc;

endmodule
```

Test Bench (Without Seven Segment Display):

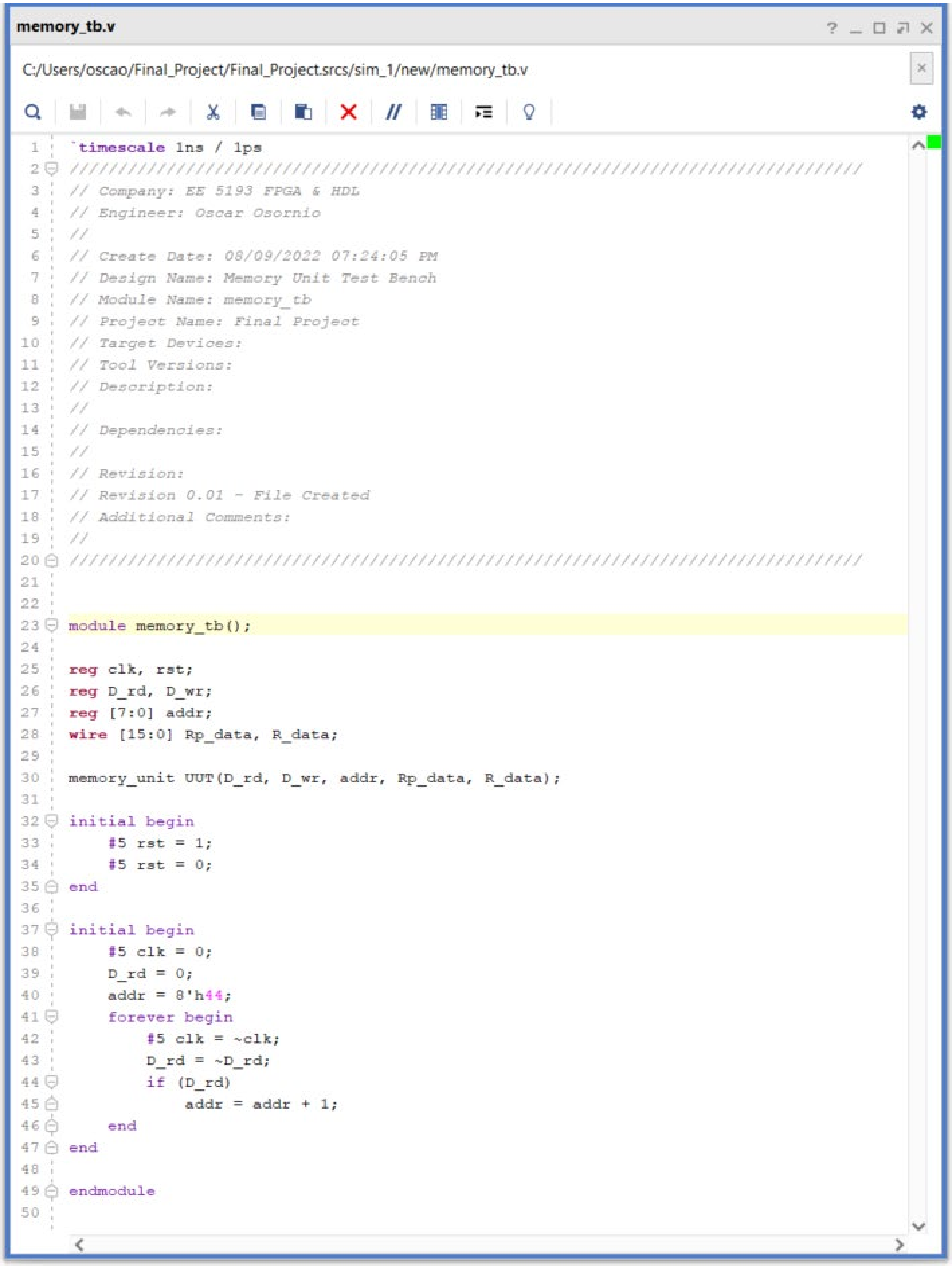


processor_tb.v

C:/Users/oscao/Final_Project/Final_Project.srcs/sim_1/new/processor_tb.v

```
`timescale 1ns / 1ps
//////////////////////////////////////////////////////////////////////////////////
// Company: EE 5193 FPGA & HDL
// Engineer: Oscar Osornio
//
// Create Date: 08/08/2022 08:38:27 PM
// Design Name: RISC MIPS Processor Test Bench
// Module Name: processor_tb
// Project Name: Final Project
// Target Devices:
// Tool Versions:
// Description:
//
// Dependencies:
//
// Revision:
// Revision 0.01 - File Created
// Additional Comments:
//
//////////////////////////////////////////////////////////////////////////////////


module processor_tb();

    reg clk, rst;
    wire PC_ld, PC_clr, PC_inc, D_addr_sel, IR_ld, D_rd, D_wr, W_wr, Rp_rd, Rq_rd, RF_Rp_zero;
    wire [1:0] RF_s;
    wire [2:0] ALU_s;
    wire [3:0] RF_W_addr, RF_Rp_addr, RF_Rq_addr;
    wire [7:0] PC_addr, D_addr, addr, RF_W_data;
    wire [15:0] R_data, IR_data, Rp_data, Rq_data, ALU_data, MUX_data;

    RISC_MIPS_processor UUT(clk, rst, PC_ld, PC_clr, PC_inc, D_addr_sel, IR_ld, D_rd, D_wr,
                            W_wr, Rp_rd, Rq_rd, RF_Rp_zero, RF_s, ALU_s, RF_W_addr, RF_Rp_addr,
                            RF_Rq_addr, PC_addr, D_addr, addr, RF_W_data, R_data, IR_data,
                            Rp_data, Rq_data, ALU_data, MUX_data);

    initial begin
        #5 rst = 1;
        #5 rst = 0;
    end

    initial begin
        #5 clk = 0;
        forever begin
            #5 clk = ~clk;
        end
    end

    initial
        #500 $finish;


endmodule
```

Test Bench (Memory Module):



memory_tb.v
C:/Users/oscao/Final_Project/Final_Project.srcs/sim_1/new/memory_tb.v

```
`timescale 1ns / 1ps
//////////////////////////////////////////////////////////////////////////////////
// Company: EE 5193 FPGA & HDL
// Engineer: Oscar Osornio
//
// Create Date: 08/09/2022 07:24:05 PM
// Design Name: Memory Unit Test Bench
// Module Name: memory_tb
// Project Name: Final Project
// Target Devices:
// Tool Versions:
// Description:
//
// Dependencies:
//
// Revision:
// Revision 0.01 - File Created
// Additional Comments:
//
//////////////////////////////////////////////////////////////////////////////////


module memory_tb();

reg clk, rst;
reg D_rd, D_wr;
reg [7:0] addr;
wire [15:0] Rp_data, R_data;

memory_unit UUT(D_rd, D_wr, addr, Rp_data, R_data);

initial begin
    #5 rst = 1;
    #5 rst = 0;
end

initial begin
    #5 clk = 0;
    D_rd = 0;
    addr = 8'h44;
    forever begin
        #5 clk = ~clk;
        D_rd = ~D_rd;
        if (D_rd)
            addr = addr + 1;
    end
end

endmodule

```