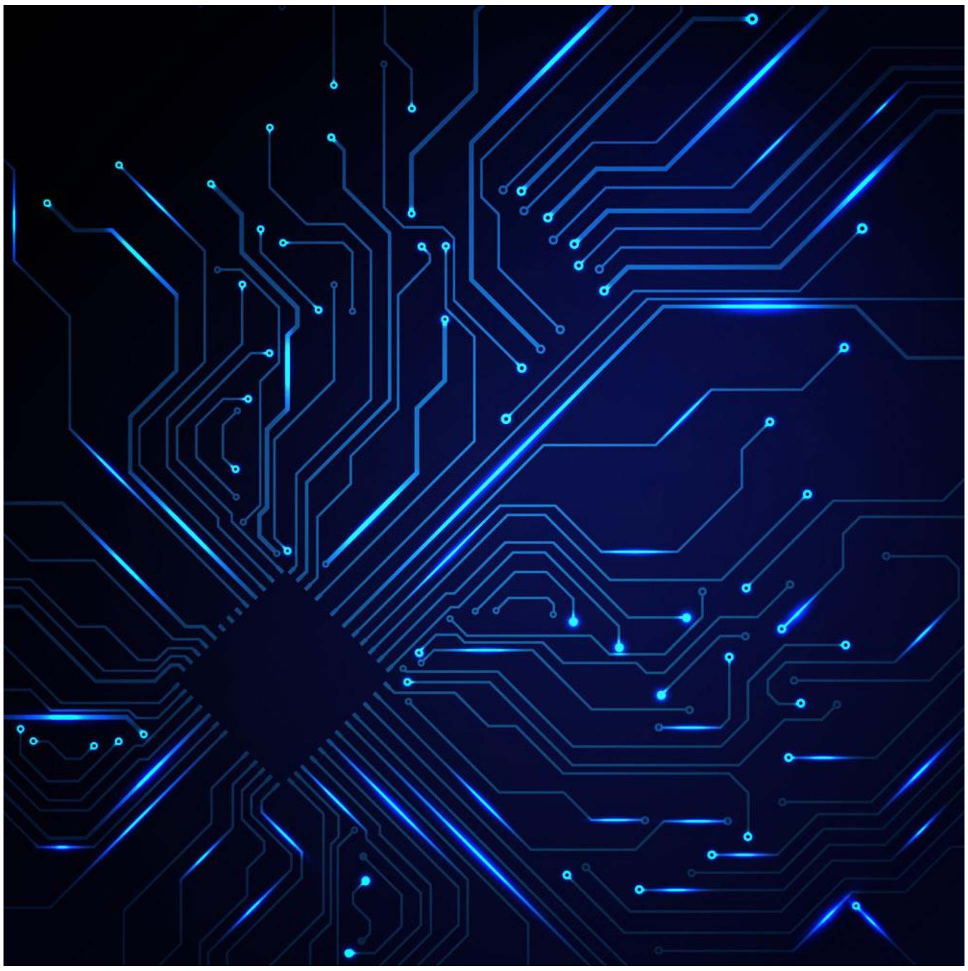


注意

1. 繳交時一律轉 PDF 檔
2. 繳交期限為
   上完課後
   當週五晚上 12 點前
3. 一人繳交一份
4. 檔名：學號_HW?.pdf
   檔名請按照作業檔名格式進行填寫
   未依照格式不予批改

2022/11/21

實驗九

暫存器定址

姓名：張銀軒　　　學號：00957050

班級：資工 3A

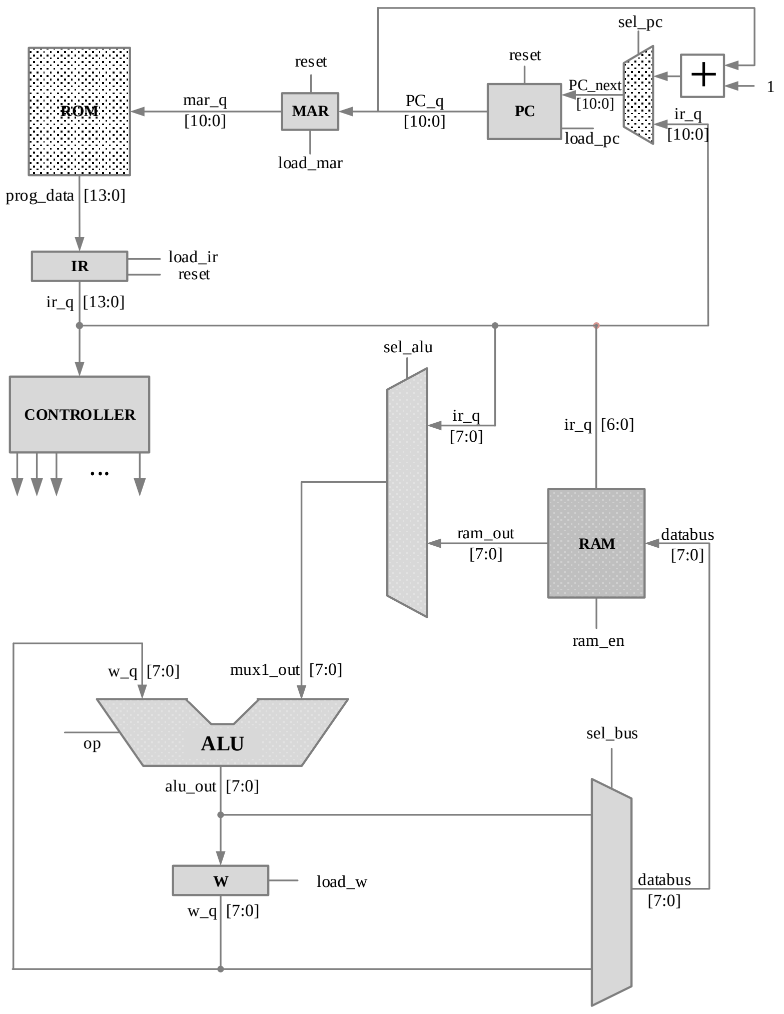
E-mail：00957050@mail.ntou.edu.tw

● 實驗説明：

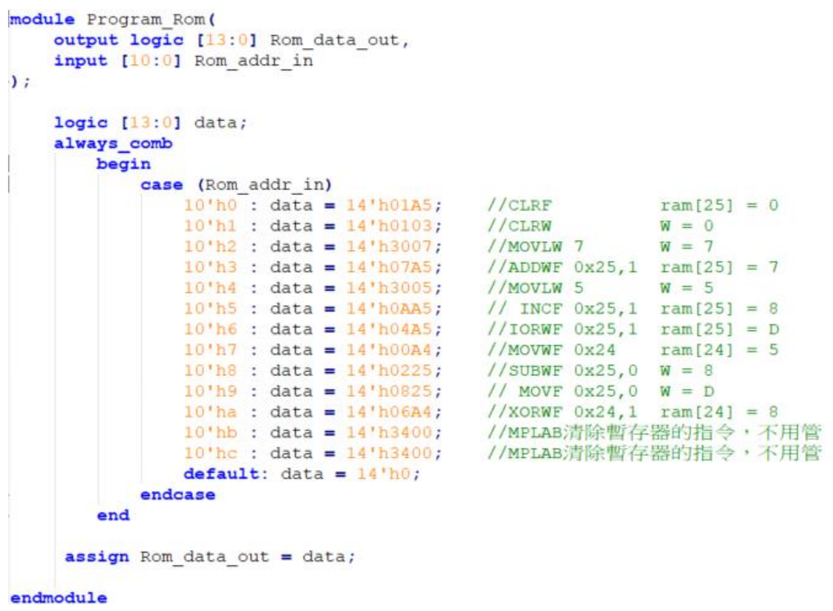
1. 如圖所示，設計一個架構實現暫存器定址的指令
2. 輸入：clk, reset
3. 輸出：w_q[7:0]

   下方有附 Rom 的截圖，請務必按照規定的 input 及 output 來做

● 系統硬體架構方塊圖（接線圖）：



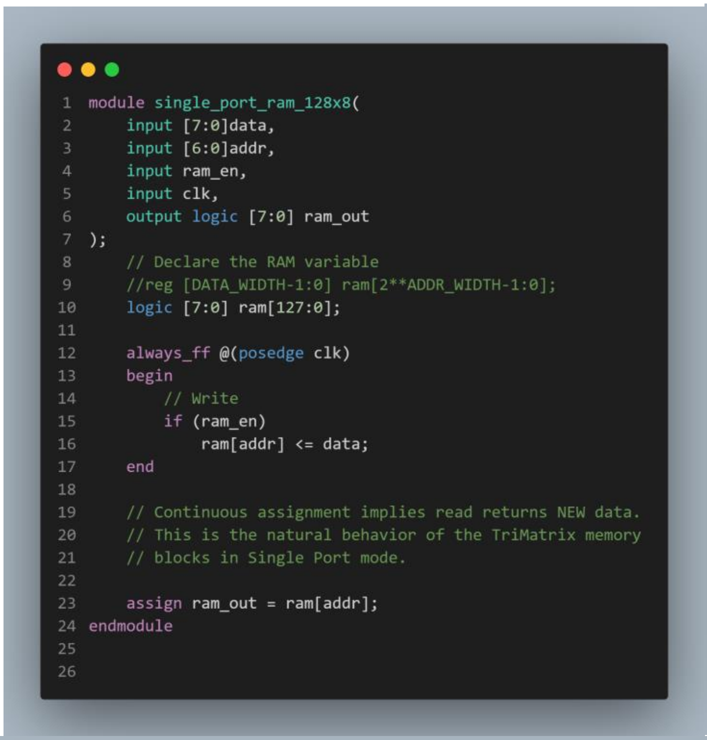
架構圖



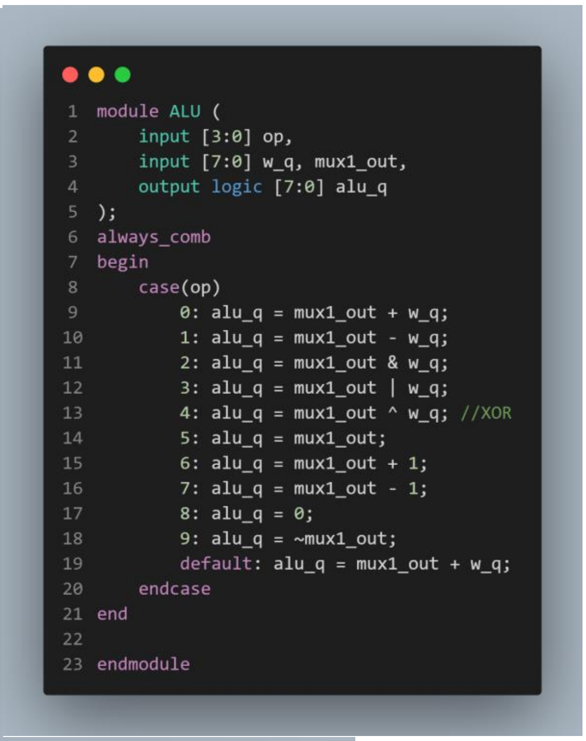
```
module Program_Rom(
    output logic [13:0] Rom_data_out,
    input [10:0] Rom_addr_in
);

    logic [13:0] data;
    always_comb
        begin
            case (Rom_addr_in)
                10'h0 : data = 14'h01A5;    //CLRF          ram[25] = 0
                10'h1 : data = 14'h0103;    //CLRW          W = 0
                10'h2 : data = 14'h3007;    //MOVLW 7       W = 7
                10'h3 : data = 14'h07A5;    //ADDWF 0x25,1  ram[25] = 7
                10'h4 : data = 14'h3005;    //MOVLW 5       W = 5
                10'h5 : data = 14'h0AA5;    // INCF 0x25,1  ram[25] = 8
                10'h6 : data = 14'h04A5;    //IORWF 0x25,1  ram[25] = D
                10'h7 : data = 14'h00A4;    //MOVWF 0x24    ram[24] = 5
                10'h8 : data = 14'h0225;    //SUBWF 0x25,0  W = 8
                10'h9 : data = 14'h0825;    // MOVF 0x25,0  W = D
                10'ha : data = 14'h06A4;    //XORWF 0x24,1  ram[24] = 8
                10'hb : data = 14'h3400;    //MPLAB清除暫存器的指令，不用管
                10'hc : data = 14'h3400;    //MPLAB清除暫存器的指令，不用管
                default: data = 14'h0;
            endcase
        end

     assign Rom_data_out = data;

endmodule
```

Program_Rom

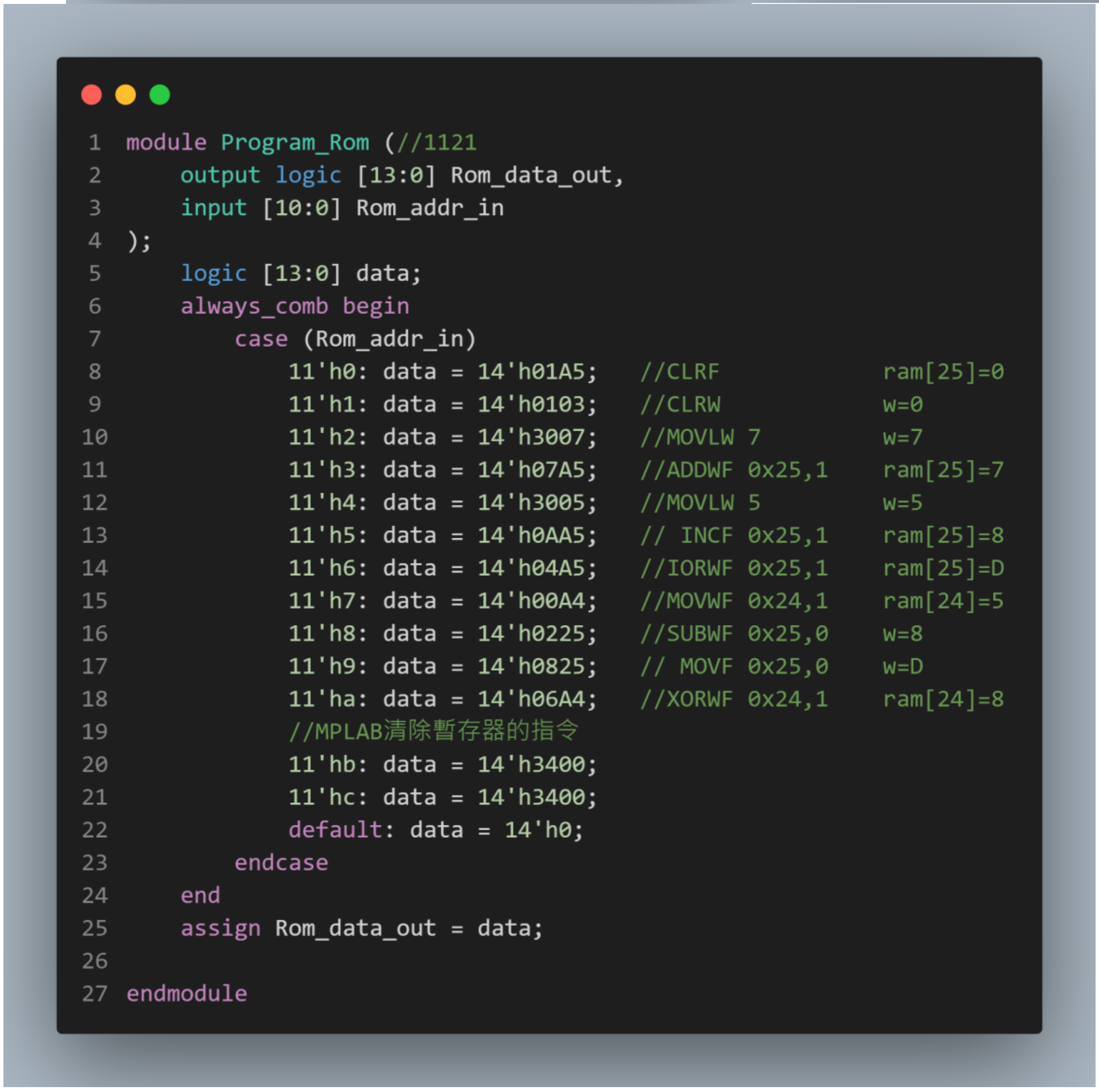
- 系統架構程式碼、測試資料程式碼與程式碼說明(.sv 檔及.do 檔都要截圖)
  截圖請善用 win+shift+S



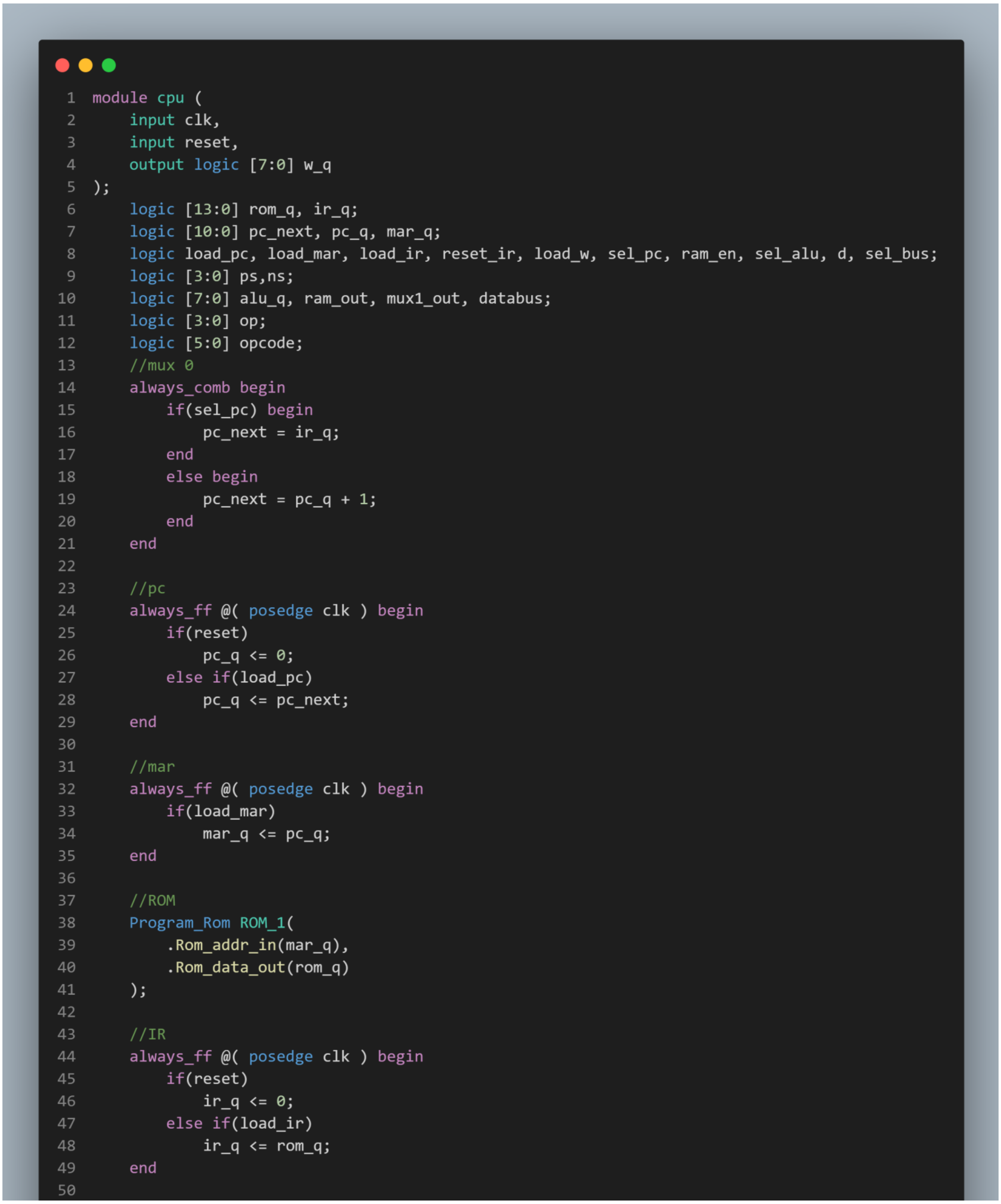
```
module single_port_ram_128x8(
    input [7:0]data,
    input [6:0]addr,
    input ram_en,
    input clk,
    output logic [7:0] ram_out
);
    // Declare the RAM variable
    //reg [DATA_WIDTH-1:0] ram[2**ADDR_WIDTH-1:0];
    logic [7:0] ram[127:0];

    always_ff @(posedge clk)
    begin
        // Write
        if (ram_en)
            ram[addr] <= data;
    end

    // Continuous assignment implies read returns NEW data.
    // This is the natural behavior of the TriMatrix memory
    // blocks in Single Port mode.

    assign ram_out = ram[addr];
endmodule


```



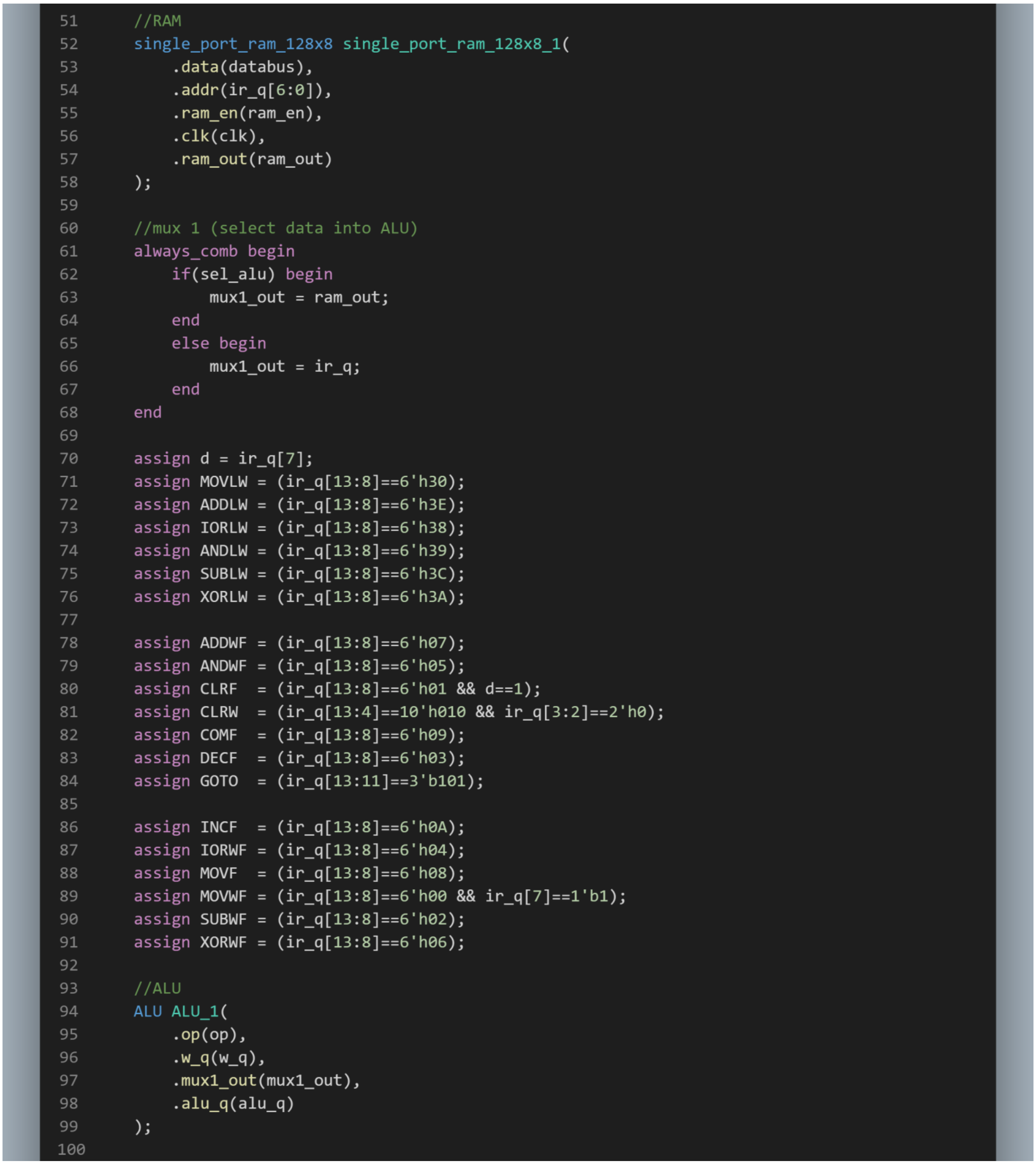
```
module ALU (
    input [3:0] op,
    input [7:0] w_q, mux1_out,
    output logic [7:0] alu_q
);
always_comb
begin
    case(op)
        0: alu_q = mux1_out + w_q;
        1: alu_q = mux1_out - w_q;
        2: alu_q = mux1_out & w_q;
        3: alu_q = mux1_out | w_q;
        4: alu_q = mux1_out ^ w_q; //XOR
        5: alu_q = mux1_out;
        6: alu_q = mux1_out + 1;
        7: alu_q = mux1_out - 1;
        8: alu_q = 0;
        9: alu_q = ~mux1_out;
        default: alu_q = mux1_out + w_q;
    endcase
end

endmodule
```



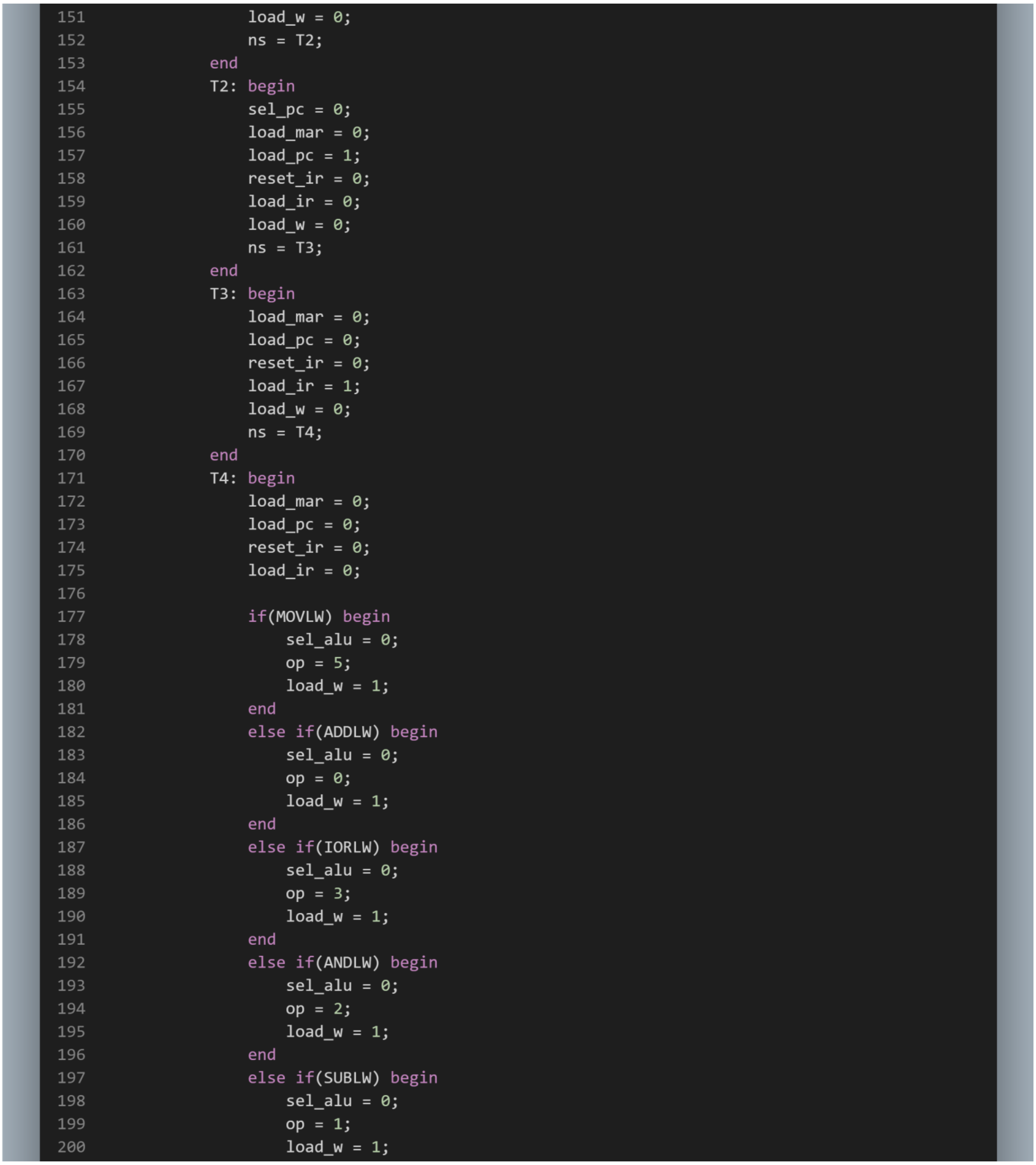
```
module Program_Rom (//1121
    output logic [13:0] Rom_data_out,
    input [10:0] Rom_addr_in
);
    logic [13:0] data;
    always_comb begin
        case (Rom_addr_in)
            11'h0: data = 14'h01A5;   //CLRF          ram[25]=0
            11'h1: data = 14'h0103;   //CLRW          w=0
            11'h2: data = 14'h3007;   //MOVLW 7       w=7
            11'h3: data = 14'h07A5;   //ADDWF 0x25,1  ram[25]=7
            11'h4: data = 14'h3005;   //MOVLW 5       w=5
            11'h5: data = 14'h0AA5;   // INCF 0x25,1  ram[25]=8
            11'h6: data = 14'h04A5;   //IORWF 0x25,1  ram[25]=D
            11'h7: data = 14'h00A4;   //MOVWF 0x24,1  ram[24]=5
            11'h8: data = 14'h0225;   //SUBWF 0x25,0  w=8
            11'h9: data = 14'h0825;   // MOVF 0x25,0  w=D
            11'ha: data = 14'h06A4;   //XORWF 0x24,1  ram[24]=8
            //MPLAB清除暫存器的指令
            11'hb: data = 14'h3400;
            11'hc: data = 14'h3400;
            default: data = 14'h0;
        endcase
    end
    assign Rom_data_out = data;

endmodule
```



```
module cpu (
    input clk,
    input reset,
    output logic [7:0] w_q
);
    logic [13:0] rom_q, ir_q;
    logic [10:0] pc_next, pc_q, mar_q;
    logic load_pc, load_mar, load_ir, reset_ir, load_w, sel_pc, ram_en, sel_alu, d, sel_bus;
    logic [3:0] ps,ns;
    logic [7:0] alu_q, ram_out, mux1_out, databus;
    logic [3:0] op;
    logic [5:0] opcode;
    //mux 0
    always_comb begin
        if(sel_pc) begin
            pc_next = ir_q;
        end
        else begin
            pc_next = pc_q + 1;
        end
    end

    //pc
    always_ff @( posedge clk ) begin
        if(reset)
            pc_q <= 0;
        else if(load_pc)
            pc_q <= pc_next;
    end

    //mar
    always_ff @( posedge clk ) begin
        if(load_mar)
            mar_q <= pc_q;
    end

    //ROM
    Program_Rom ROM_1(
        .Rom_addr_in(mar_q),
        .Rom_data_out(rom_q)
    );

    //IR
    always_ff @( posedge clk ) begin
        if(reset)
            ir_q <= 0;
        else if(load_ir)
            ir_q <= rom_q;
    end

```

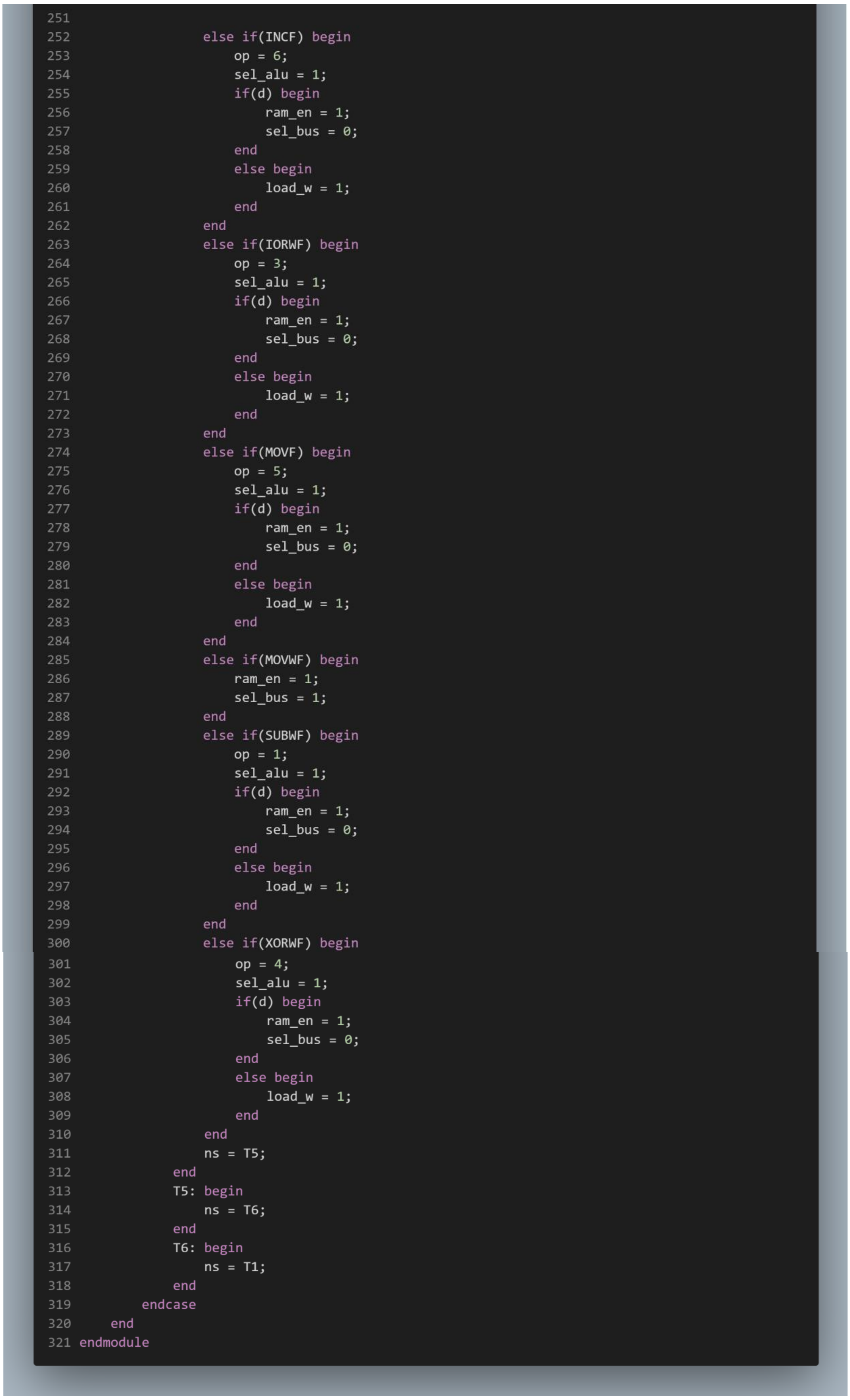


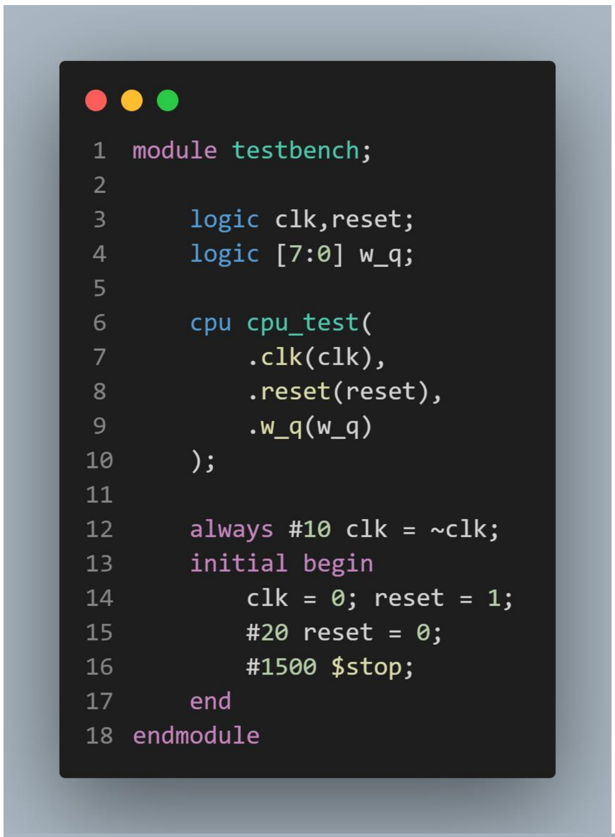
```verilog
    //RAM
    single_port_ram_128x8 single_port_ram_128x8_1(
        .data(databus),
        .addr(ir_q[6:0]),
        .ram_en(ram_en),
        .clk(clk),
        .ram_out(ram_out)
    );

    //mux 1 (select data into ALU)
    always_comb begin
        if(sel_alu) begin
            mux1_out = ram_out;
        end
        else begin
            mux1_out = ir_q;
        end
    end

    assign d = ir_q[7];
    assign MOVLW = (ir_q[13:8]==6'h30);
    assign ADDLW = (ir_q[13:8]==6'h3E);
    assign IORLW = (ir_q[13:8]==6'h38);
    assign ANDLW = (ir_q[13:8]==6'h39);
    assign SUBLW = (ir_q[13:8]==6'h3C);
    assign XORLW = (ir_q[13:8]==6'h3A);

    assign ADDWF = (ir_q[13:8]==6'h07);
    assign ANDWF = (ir_q[13:8]==6'h05);
    assign CLRF  = (ir_q[13:8]==6'h01 && d==1);
    assign CLRW  = (ir_q[13:4]==10'h010 && ir_q[3:2]==2'h0);
    assign COMF  = (ir_q[13:8]==6'h09);
    assign DECF  = (ir_q[13:8]==6'h03);
    assign GOTO  = (ir_q[13:11]==3'b101);

    assign INCF  = (ir_q[13:8]==6'h0A);
    assign IORWF = (ir_q[13:8]==6'h04);
    assign MOVF  = (ir_q[13:8]==6'h08);
    assign MOVWF = (ir_q[13:8]==6'h00 && ir_q[7]==1'b1);
    assign SUBWF = (ir_q[13:8]==6'h02);
    assign XORWF = (ir_q[13:8]==6'h06);

    //ALU
    ALU ALU_1(
        .op(op),
        .w_q(w_q),
        .mux1_out(mux1_out),
        .alu_q(alu_q)
    );

    //register
    always_ff @( posedge clk ) begin
        if(load_w)
            w_q <= alu_q;
    end

    //mux 2 (select data into RAM)
    always_comb begin
        if(sel_bus) begin
            databus = w_q;
        end
        else begin
            databus = alu_q;
        end
    end

    //controller
    parameter T0 = 0;
    parameter T1 = 1;
    parameter T2 = 2;
    parameter T3 = 3;
    parameter T4 = 4;
    parameter T5 = 5;
    parameter T6 = 6;

    always_ff @( posedge clk ) begin
        if(reset) ps <= 0;
        else ps <= ns;
    end

    always_comb begin
        sel_alu = 0;
        sel_pc = 0;
        load_mar = 0;
        load_pc = 0;
        reset_ir = 1;
        load_ir = 0;
        load_w = 0;
        ram_en=0;
        op =0;
        ns=0;
        case(ps)
            T0: begin
                ns = T1;
            end
            T1: begin
                load_mar = 1;
                load_pc = 0;
                reset_ir = 0;
                load_ir = 0;
```

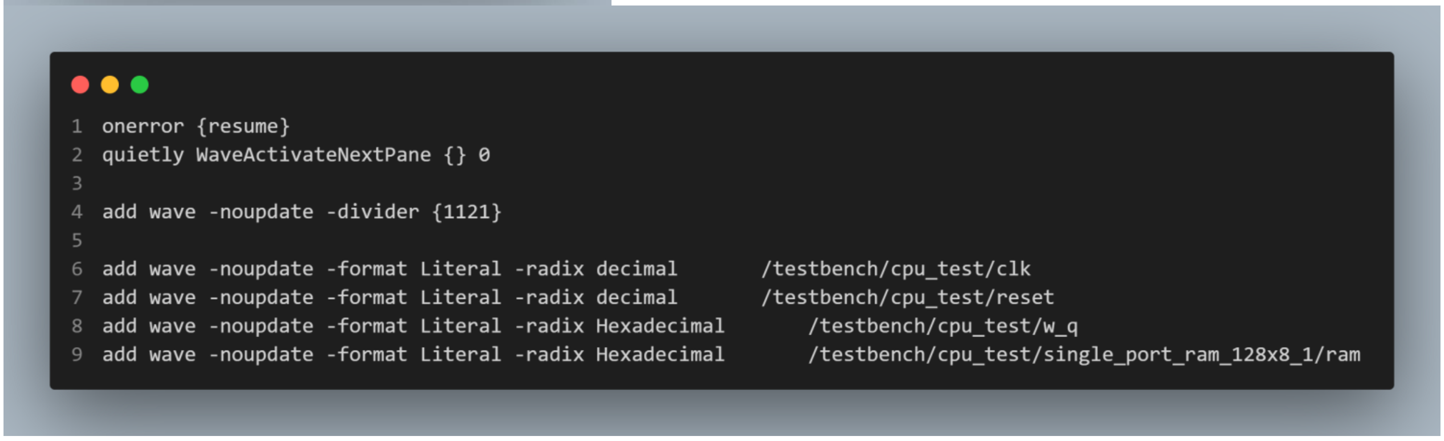


```
                load_w = 0;
                ns = T2;
            end
            T2: begin
                sel_pc = 0;
                load_mar = 0;
                load_pc = 1;
                reset_ir = 0;
                load_ir = 0;
                load_w = 0;
                ns = T3;
            end
            T3: begin
                load_mar = 0;
                load_pc = 0;
                reset_ir = 0;
                load_ir = 1;
                load_w = 0;
                ns = T4;
            end
            T4: begin
                load_mar = 0;
                load_pc = 0;
                reset_ir = 0;
                load_ir = 0;

                if(MOVLW) begin
                    sel_alu = 0;
                    op = 5;
                    load_w = 1;
                end
                else if(ADDLW) begin
                    sel_alu = 0;
                    op = 0;
                    load_w = 1;
                end
                else if(IORLW) begin
                    sel_alu = 0;
                    op = 3;
                    load_w = 1;
                end
                else if(ANDLW) begin
                    sel_alu = 0;
                    op = 2;
                    load_w = 1;
                end
                else if(SUBLW) begin
                    sel_alu = 0;
                    op = 1;
                    load_w = 1;
```

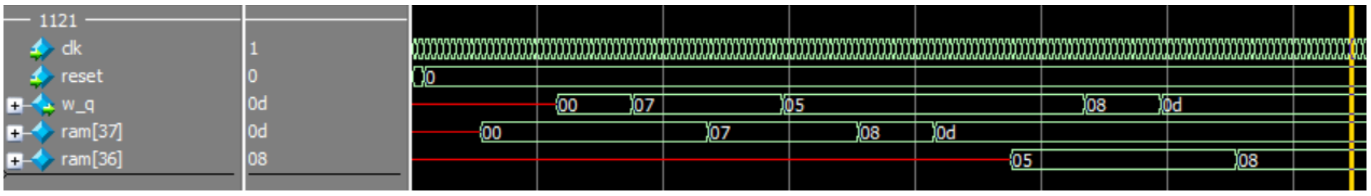
```verilog
                end
                else if(XORLW) begin
                    sel_alu = 0;
                    op = 4;
                    load_w = 1;
                end


                else if(GOTO) begin
                    sel_pc = 1;
                    load_pc = 1;
                end
                else if(ADDWF) begin
                    op = 0;
                    sel_alu = 1;
                    if(d) begin
                        ram_en = 1;
                    end
                    else begin
                        load_w = 1;
                    end
                end
                else if(ANDWF) begin
                    op = 2;
                    sel_alu = 1;
                    if(d) begin
                        ram_en = 1;
                    end
                    else begin
                        load_w = 1;
                    end
                end
                else if(CLRF) begin
                    op = 8;
                    ram_en = 1;
                end
                else if(CLRW) begin
                    op = 8;
                    load_w = 1;
                end
                else if(COMF) begin
                    op = 9;
                    sel_alu = 1;
                    ram_en = 1;
                end
                else if(DECF) begin
                    op = 7;
                    sel_alu = 1;
                    ram_en = 1;
                end
```


```verilog

                else if(INCF) begin
                    op = 6;
                    sel_alu = 1;
                    if(d) begin
                        ram_en = 1;
                        sel_bus = 0;
                    end
                    else begin
                        load_w = 1;
                    end
                end
                else if(IORWF) begin
                    op = 3;
                    sel_alu = 1;
                    if(d) begin
                        ram_en = 1;
                        sel_bus = 0;
                    end
                    else begin
                        load_w = 1;
                    end
                end
                else if(MOVF) begin
                    op = 5;
                    sel_alu = 1;
                    if(d) begin
                        ram_en = 1;
                        sel_bus = 0;
                    end
                    else begin
                        load_w = 1;
                    end
                end
                else if(MOVWF) begin
                    ram_en = 1;
                    sel_bus = 1;
                end
                else if(SUBWF) begin
                    op = 1;
                    sel_alu = 1;
                    if(d) begin
                        ram_en = 1;
                        sel_bus = 0;
                    end
                    else begin
                        load_w = 1;
                    end
                end
                else if(XORWF) begin
                    op = 4;
                    sel_alu = 1;
                    if(d) begin
                        ram_en = 1;
                        sel_bus = 0;
                    end
                    else begin
                        load_w = 1;
                    end
                end
                ns = T5;
            end
            T5: begin
                ns = T6;
            end
            T6: begin
                ns = T1;
            end
        endcase
    end
endmodule
```


```
module testbench;

    logic clk,reset;
    logic [7:0] w_q;

    cpu cpu_test(
        .clk(clk),
        .reset(reset),
        .w_q(w_q)
    );

    always #10 clk = ~clk;
    initial begin
        clk = 0; reset = 1;
        #20 reset = 0;
        #1500 $stop;
    end
endmodule
```


```
onerror {resume}
quietly WaveActivateNextPane {} 0

add wave -noupdate -divider {1121}

add wave -noupdate -format Literal -radix decimal       /testbench/cpu_test/clk
add wave -noupdate -format Literal -radix decimal       /testbench/cpu_test/reset
add wave -noupdate -format Literal -radix Hexadecimal       /testbench/cpu_test/w_q
add wave -noupdate -format Literal -radix Hexadecimal       /testbench/cpu_test/single_port_ram_128x8_1/ram
```

● 模擬結果與結果說明：



能夠依照 Program_Rom 的指令正確進行運算及寫入 w 暫存器和記憶體

● 結論與心得：

本周與上周相比又增加了一條從 w 暫存器接到 RAM 的線，可以把 w 暫存器的值放進 RAM 裡面，除此之外也增加了 6 個新的指令，目前架構看起來相當完整。老師在影片中也提到，之後會教學 MPLAB 以及把組合語言轉譯成 System Verilog，非常期待，也祝補考的同學順利!