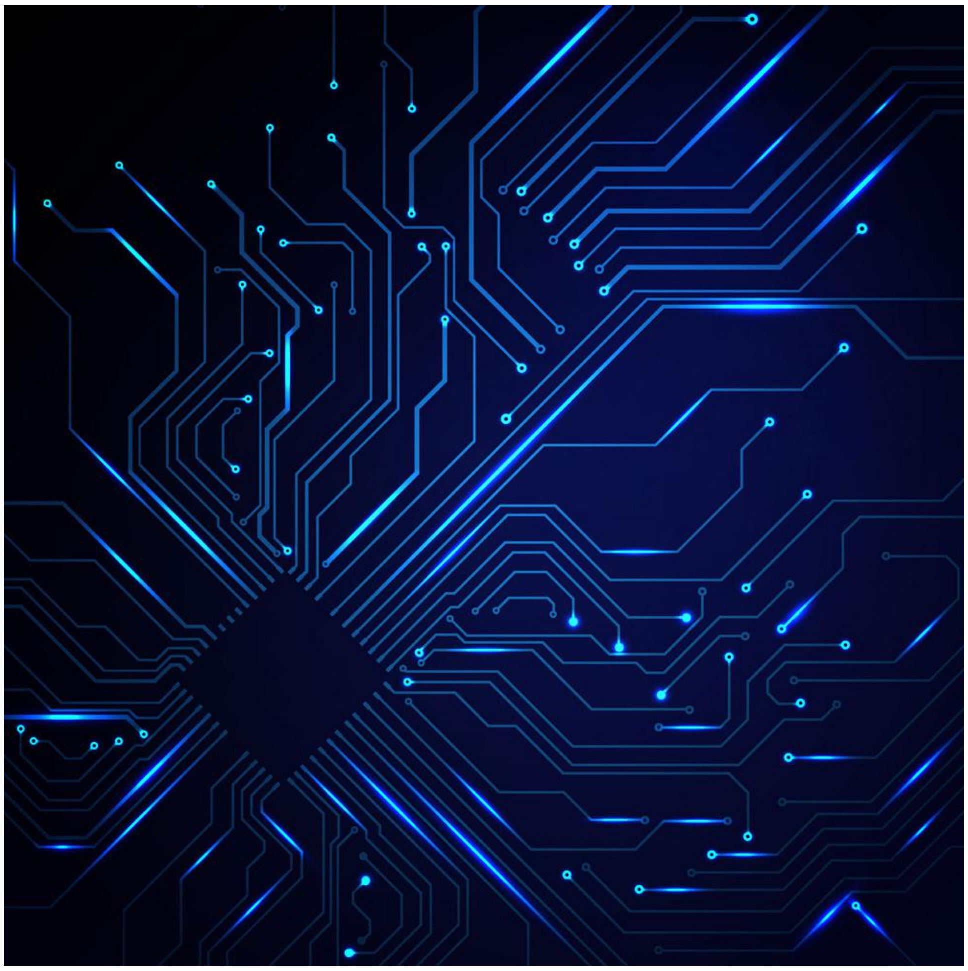


2022/11/14

實驗八

暫存器定址

姓名：王嘉羽　　　學號：00957116

班級：資工 3B

E-mail：vayne20011125@gmail.com

注意

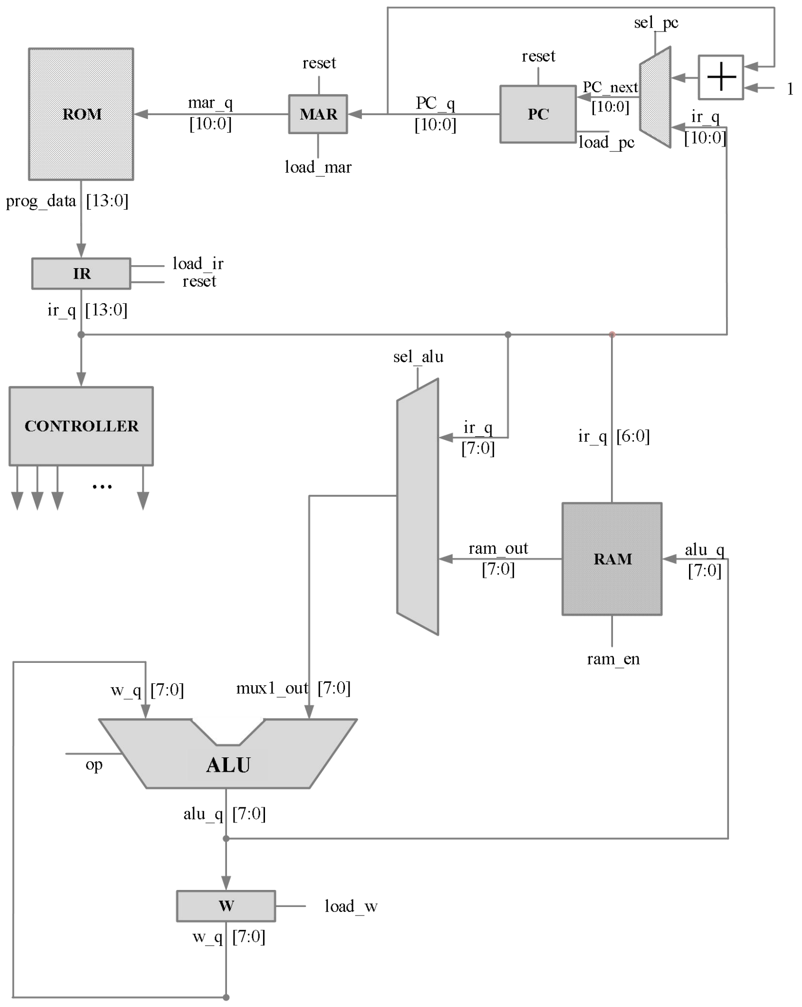
1. 繳交時一律轉 PDF 檔
2. 繳交期限為
   上完課後
   當週五晚上 12 點前
3. 一人繳交一份
4. 檔名：學號_HW?.pdf
   檔名請按照作業檔名格
   式進行填寫
   未依照格式不予批改

● 實驗說明：

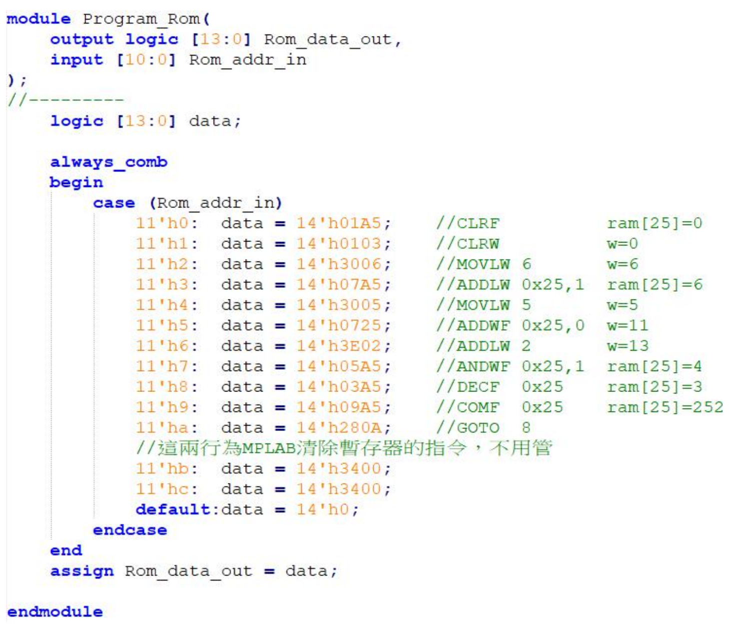
1. 如圖所示，設計一個架構實現暫存器定址的指令
2. 輸入：clk, reset
3. 輸出：w_q[7:0]

   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

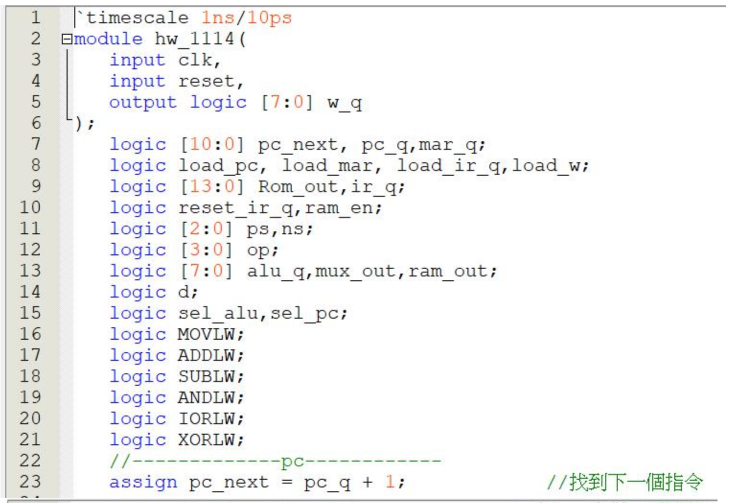


```
module Program_Rom(
    output logic [13:0] Rom_data_out,
    input [10:0] Rom_addr_in
);
//---------
    logic [13:0] data;

    always_comb
    begin
        case (Rom_addr_in)
            11'h0:  data = 14'h01A5;    //CLRF          ram[25]=0
            11'h1:  data = 14'h0103;    //CLRW          w=0
            11'h2:  data = 14'h3006;    //MOVLW 6       w=6
            11'h3:  data = 14'h07A5;    //ADDLW 0x25,1  ram[25]=6
            11'h4:  data = 14'h3005;    //MOVLW 5       w=5
            11'h5:  data = 14'h0725;    //ADDWF 0x25,0  w=11
            11'h6:  data = 14'h3E02;    //ADDLW 2       w=13
            11'h7:  data = 14'h05A5;    //ANDWF 0x25,1  ram[25]=4
            11'h8:  data = 14'h03A5;    //DECF  0x25    ram[25]=3
            11'h9:  data = 14'h09A5;    //COMF  0x25    ram[25]=252
            11'ha:  data = 14'h280A;    //GOTO  8
            //這兩行為MPLAB清除暫存器的指令，不用管
            11'hb:  data = 14'h3400;
            11'hc:  data = 14'h3400;
            default:data = 14'h0;
        endcase
    end
    assign Rom_data_out = data;

endmodule
```

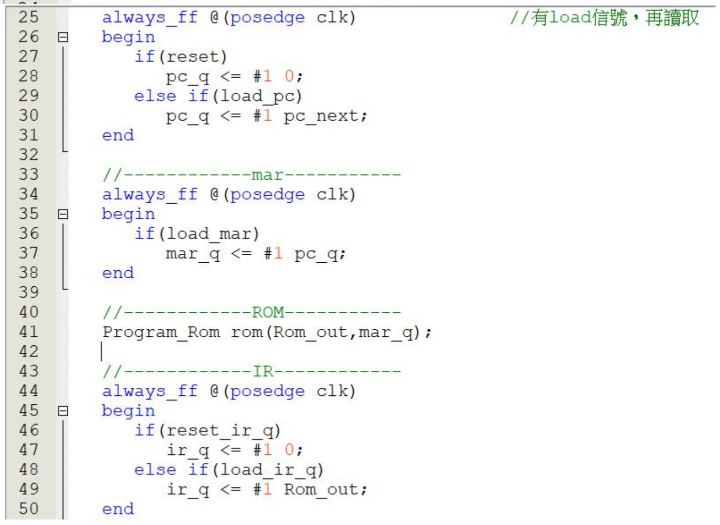
Program_Rom

- **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**

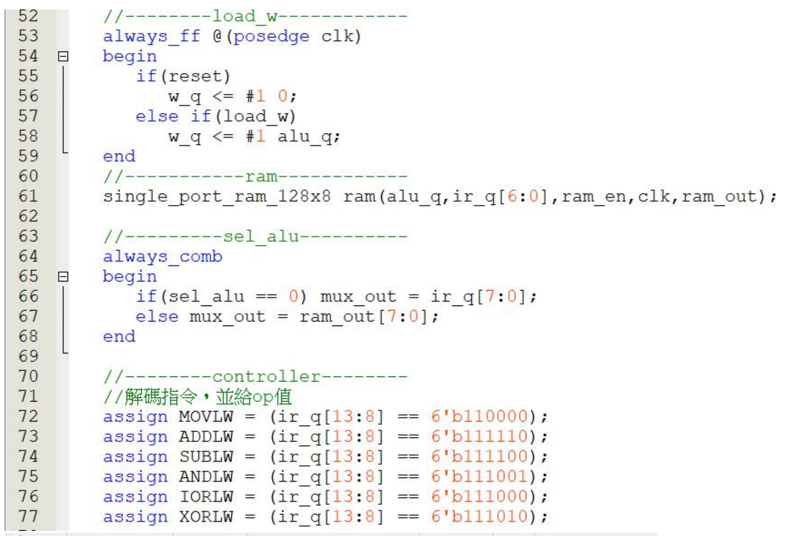
✧ hw_1114.sv



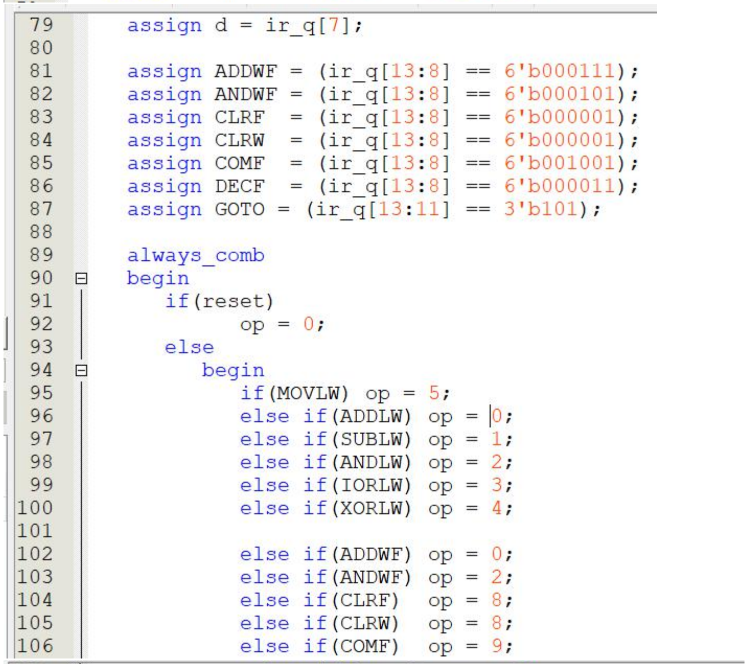
```
`timescale 1ns/10ps
module hw_1114(
    input clk,
    input reset,
    output logic [7:0] w_q
);
    logic [10:0] pc_next, pc_q,mar_q;
    logic load_pc, load_mar, load_ir_q,load_w;
    logic [13:0] Rom_out,ir_q;
    logic reset_ir_q,ram_en;
    logic [2:0] ps,ns;
    logic [3:0] op;
    logic [7:0] alu_q,mux_out,ram_out;
    logic d;
    logic sel_alu,sel_pc;
    logic MOVLW;
    logic ADDLW;
    logic SUBLW;
    logic ANDLW;
    logic IORLW;
    logic XORLW;
    //-------------pc------------
    assign pc_next = pc_q + 1;                  //找到下一個指令
```



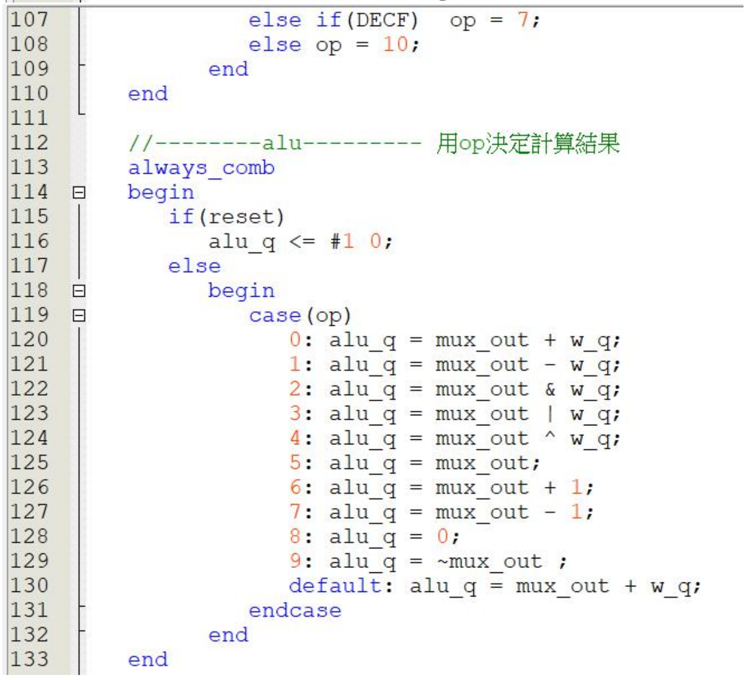
```
    always_ff @(posedge clk)                  //有load信號，再讀取
    begin
        if(reset)
            pc_q <= #1 0;
        else if(load_pc)
            pc_q <= #1 pc_next;
    end

    //------------mar-----------
    always_ff @(posedge clk)
    begin
        if(load_mar)
            mar_q <= #1 pc_q;
    end

    //------------ROM-----------
    Program_Rom rom(Rom_out,mar_q);

    //------------IR------------
    always_ff @(posedge clk)
    begin
        if(reset_ir_q)
            ir_q <= #1 0;
        else if(load_ir_q)
            ir_q <= #1 Rom_out;
    end
```



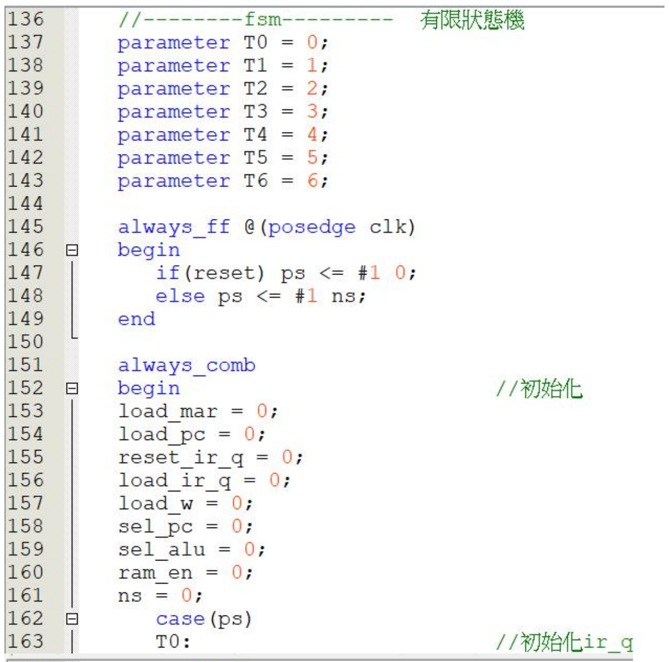
```
//--------load_w------------
always_ff @(posedge clk)
begin
   if(reset)
      w_q <= #1 0;
   else if(load_w)
      w_q <= #1 alu_q;
end
//-----------ram------------
single_port_ram_128x8 ram(alu_q,ir_q[6:0],ram_en,clk,ram_out);

//---------sel_alu----------
always_comb
begin
   if(sel_alu == 0) mux_out = ir_q[7:0];
   else mux_out = ram_out[7:0];
end

//--------controller--------
//解碼指令，並給op值
assign MOVLW = (ir_q[13:8] == 6'b110000);
assign ADDLW = (ir_q[13:8] == 6'b111110);
assign SUBLW = (ir_q[13:8] == 6'b111100);
assign ANDLW = (ir_q[13:8] == 6'b111001);
assign IORLW = (ir_q[13:8] == 6'b111000);
assign XORLW = (ir_q[13:8] == 6'b111010);
```



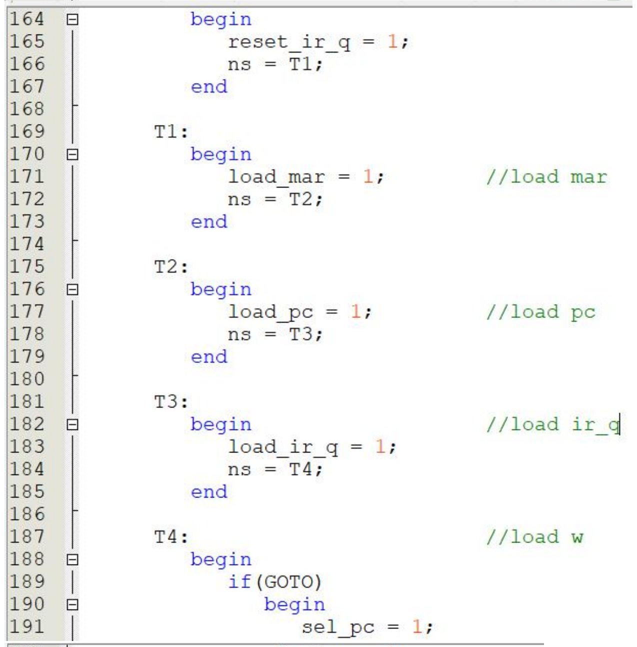
```
assign d = ir_q[7];

assign ADDWF = (ir_q[13:8] == 6'b000111);
assign ANDWF = (ir_q[13:8] == 6'b000101);
assign CLRF  = (ir_q[13:8] == 6'b000001);
assign CLRW  = (ir_q[13:8] == 6'b000001);
assign COMF  = (ir_q[13:8] == 6'b001001);
assign DECF  = (ir_q[13:8] == 6'b000011);
assign GOTO = (ir_q[13:11] == 3'b101);

always_comb
begin
   if(reset)
         op = 0;
   else
      begin
         if(MOVLW) op = 5;
         else if(ADDLW) op = 0;
         else if(SUBLW) op = 1;
         else if(ANDLW) op = 2;
         else if(IORLW) op = 3;
         else if(XORLW) op = 4;

         else if(ADDWF) op = 0;
         else if(ANDWF) op = 2;
         else if(CLRF)  op = 8;
         else if(CLRW)  op = 8;
         else if(COMF)  op = 9;
```



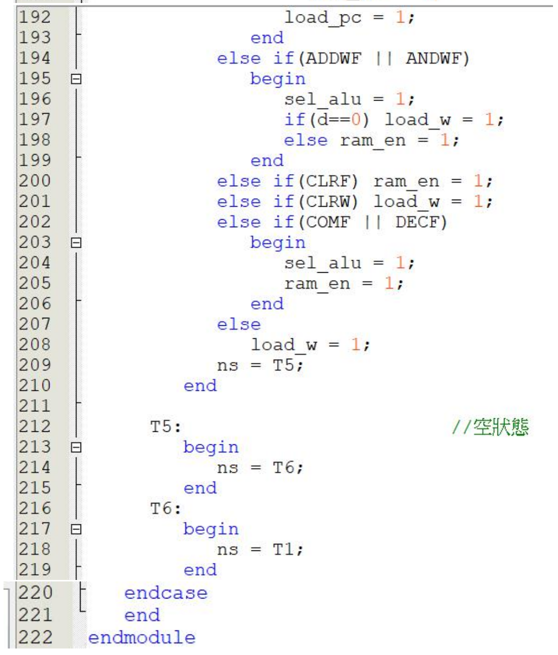
```
         else if(DECF)   op = 7;
         else op = 10;
      end
end

//--------alu--------- 用op決定計算結果
always_comb
begin
   if(reset)
      alu_q <= #1 0;
   else
      begin
         case(op)
            0: alu_q = mux_out + w_q;
            1: alu_q = mux_out - w_q;
            2: alu_q = mux_out & w_q;
            3: alu_q = mux_out | w_q;
            4: alu_q = mux_out ^ w_q;
            5: alu_q = mux_out;
            6: alu_q = mux_out + 1;
            7: alu_q = mux_out - 1;
            8: alu_q = 0;
            9: alu_q = ~mux_out ;
            default: alu_q = mux_out + w_q;
         endcase
      end
end
```



```
//--------fsm---------   有限狀態機
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 <= #1 0;
   else ps <= #1 ns;
end

always_comb
begin                              //初始化
load_mar = 0;
load_pc = 0;
reset_ir_q = 0;
load_ir_q = 0;
load_w = 0;
sel_pc = 0;
sel_alu = 0;
ram_en = 0;
ns = 0;
   case(ps)
   T0:                             //初始化ir_q
```

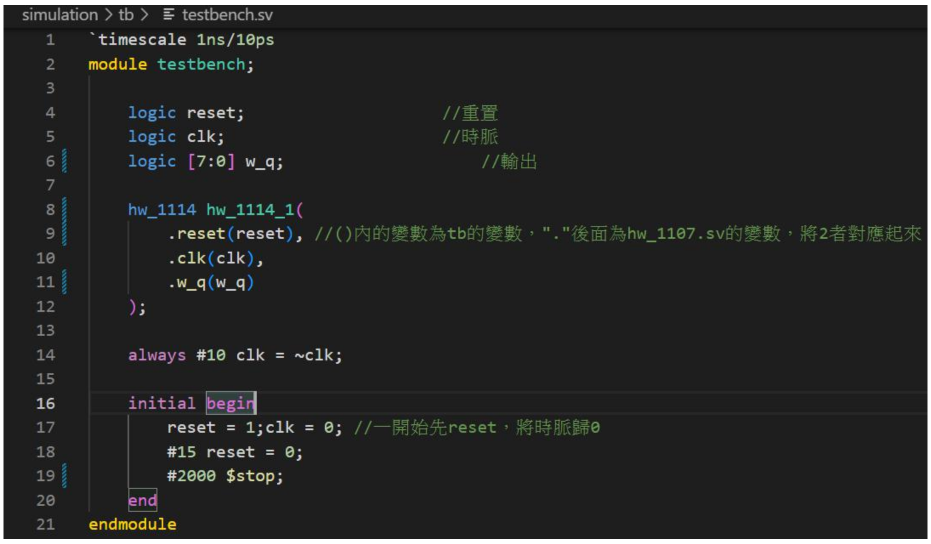


```
      begin
         reset_ir_q = 1;
         ns = T1;
      end

   T1:
      begin
         load_mar = 1;          //load mar
         ns = T2;
      end

   T2:
      begin
         load_pc = 1;           //load pc
         ns = T3;
      end

   T3:
      begin                     //load ir_q
         load_ir_q = 1;
         ns = T4;
      end

   T4:                          //load w
      begin
         if(GOTO)
            begin
               sel_pc = 1;
```



```
               load_pc = 1;
            end
         else if(ADDWF || ANDWF)
            begin
               sel_alu = 1;
               if(d==0) load_w = 1;
               else ram_en = 1;
            end
         else if(CLRF) ram_en = 1;
         else if(CLRW) load_w = 1;
         else if(COMF || DECF)
            begin
               sel_alu = 1;
               ram_en = 1;
            end
         else
            load_w = 1;
         ns = T5;
      end

   T5:                          //空狀態
      begin
         ns = T6;
      end
   T6:
      begin
         ns = T1;
      end
   endcase
   end
endmodule
```

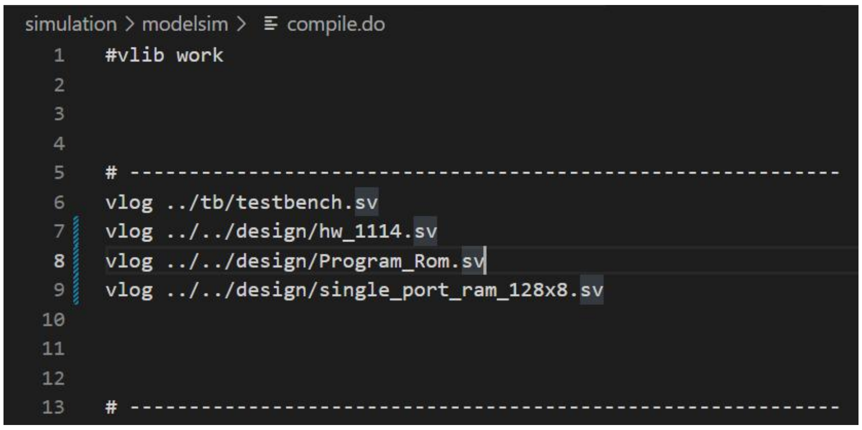
✧ testbench.sv



simulation > tb > testbench.sv

```
`timescale 1ns/10ps
module testbench;

    logic reset;                    //重置
    logic clk;                      //時脈
    logic [7:0] w_q;                    //輸出

    hw_1114 hw_1114_1(
        .reset(reset), //()內的變數為tb的變數，"."後面為hw_1107.sv的變數，將2者對應起來
        .clk(clk),
        .w_q(w_q)
    );

    always #10 clk = ~clk;

    initial begin
        reset = 1;clk = 0; //一開始先reset，將時脈歸0
        #15 reset = 0;
        #2000 $stop;
    end
endmodule
```

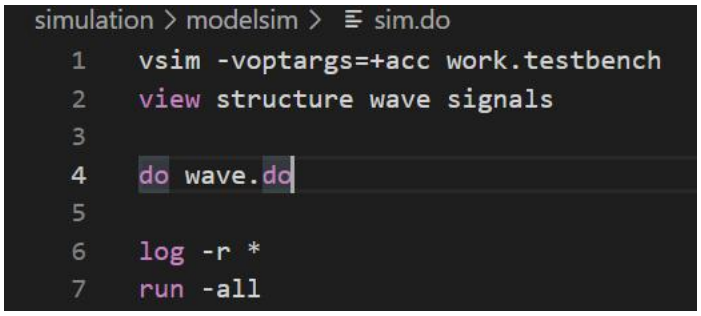
✧ compile.do



simulation > modelsim > compile.do

```
#vlib work



# -----------------------------------------------------------
vlog ../tb/testbench.sv
vlog ../../design/hw_1114.sv
vlog ../../design/Program_Rom.sv
vlog ../../design/single_port_ram_128x8.sv



# -----------------------------------------------------------
```

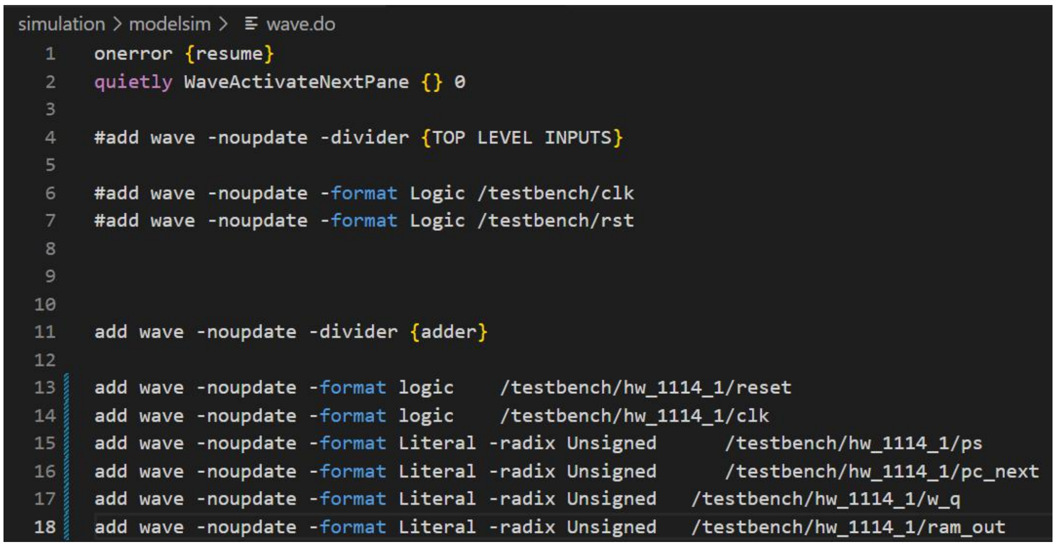
✧ sim.do



simulation > modelsim > sim.do

```
vsim -voptargs=+acc work.testbench
view structure wave signals

do wave.do

log -r *
run -all
```

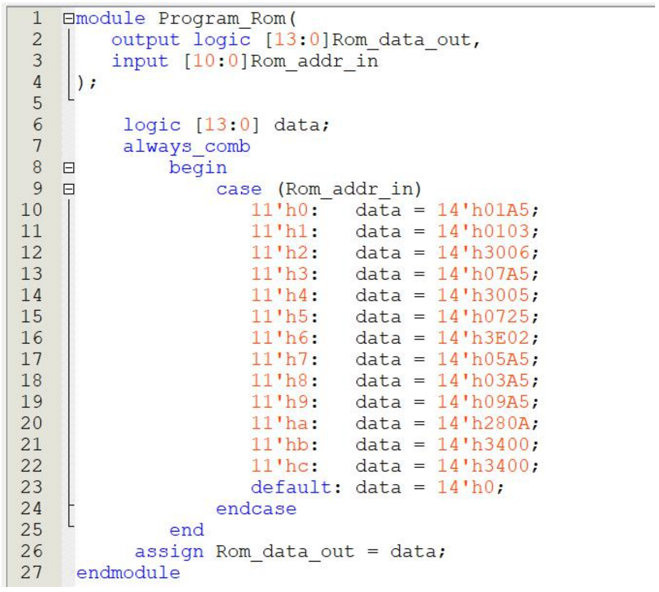
✧ wave.do



simulation > modelsim > wave.do

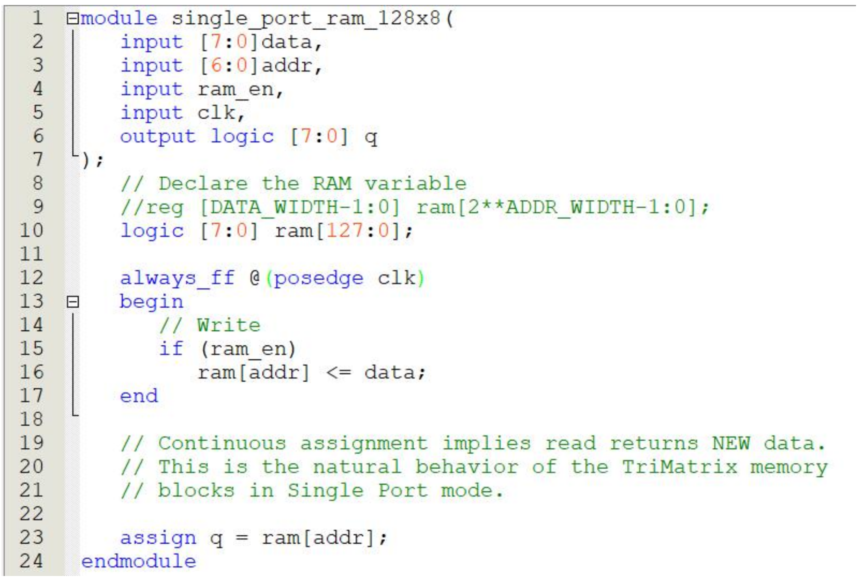
```
onerror {resume}
quietly WaveActivateNextPane {} 0

#add wave -noupdate -divider {TOP LEVEL INPUTS}

#add wave -noupdate -format Logic /testbench/clk
#add wave -noupdate -format Logic /testbench/rst



add wave -noupdate -divider {adder}

add wave -noupdate -format logic    /testbench/hw_1114_1/reset
add wave -noupdate -format logic    /testbench/hw_1114_1/clk
add wave -noupdate -format Literal -radix Unsigned      /testbench/hw_1114_1/ps
add wave -noupdate -format Literal -radix Unsigned      /testbench/hw_1114_1/pc_next
add wave -noupdate -format Literal -radix Unsigned   /testbench/hw_1114_1/w_q
add wave -noupdate -format Literal -radix Unsigned   /testbench/hw_1114_1/ram_out
```

✧ program_Rom.sv



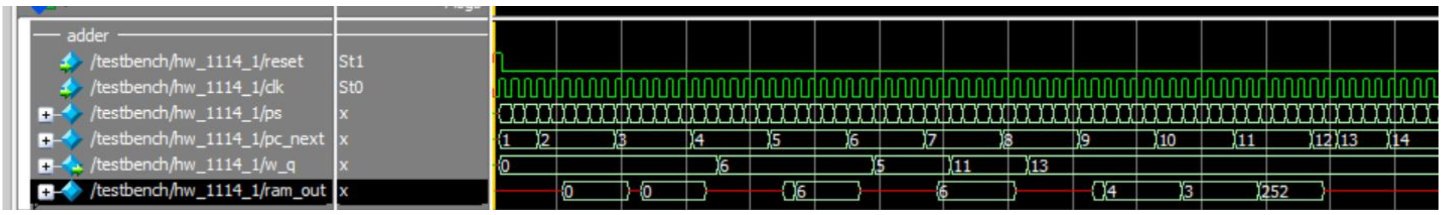
```
module Program_Rom(
    output logic [13:0]Rom_data_out,
    input [10:0]Rom_addr_in
);

     logic [13:0] data;
     always_comb
         begin
             case (Rom_addr_in)
                11'h0:    data = 14'h01A5;
                11'h1:    data = 14'h0103;
                11'h2:    data = 14'h3006;
                11'h3:    data = 14'h07A5;
                11'h4:    data = 14'h3005;
                11'h5:    data = 14'h0725;
                11'h6:    data = 14'h3E02;
                11'h7:    data = 14'h05A5;
                11'h8:    data = 14'h03A5;
                11'h9:    data = 14'h09A5;
                11'ha:    data = 14'h280A;
                11'hb:    data = 14'h3400;
                11'hc:    data = 14'h3400;
                default: data = 14'h0;
             endcase
         end
      assign Rom_data_out = data;
endmodule
```

✧ single_port_ram_128x8.sv



```
module single_port_ram_128x8(
    input [7:0]data,
    input [6:0]addr,
    input ram_en,
    input clk,
    output logic [7:0] q
);
    // 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 q = ram[addr];
endmodule
```

● 模擬結果與結果說明：



```
adder
/testbench/hw_1114_1/reset     St1
/testbench/hw_1114_1/clk       St0
/testbench/hw_1114_1/ps        x
/testbench/hw_1114_1/pc_next   x    1 2 3 4 5 6 7 8 9 10 11 12 13 14
/testbench/hw_1114_1/w_q       x    0 6 5 11 13
/testbench/hw_1114_1/ram_out   x    0 0 6 6 4 3 252
```

● 結論與心得：

經過這次的作業我好像稍微了解之前計算機組織課在教的東西，就是組合語言的 r,i,j-format 指令，每一個的格式都不一樣，以前還不懂為甚麼要分格式，現在懂了!也大致了解每個在表達的東西!這次收穫良多!

不過這次在實作上也遇到奇怪問題，一開始可以編譯，然後過一下他突然不能編譯了!!不過後來發現是 if 後面要寫一個 else，不然他會不讓我過 qq