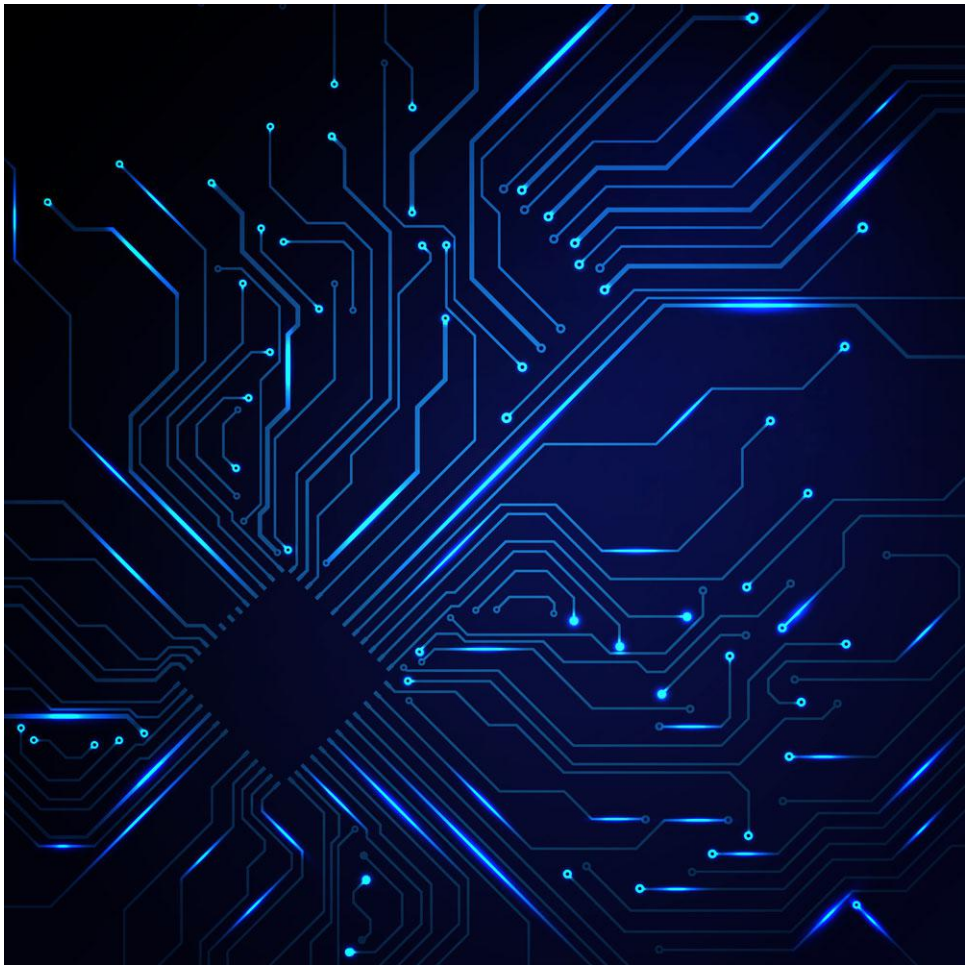


# 2022/11/21

# 實驗九

## 暫存器定址

姓名：王嘉羽　　學號：00957116

班級：資工 3B

E-mail：vayne20011125@gmail.com

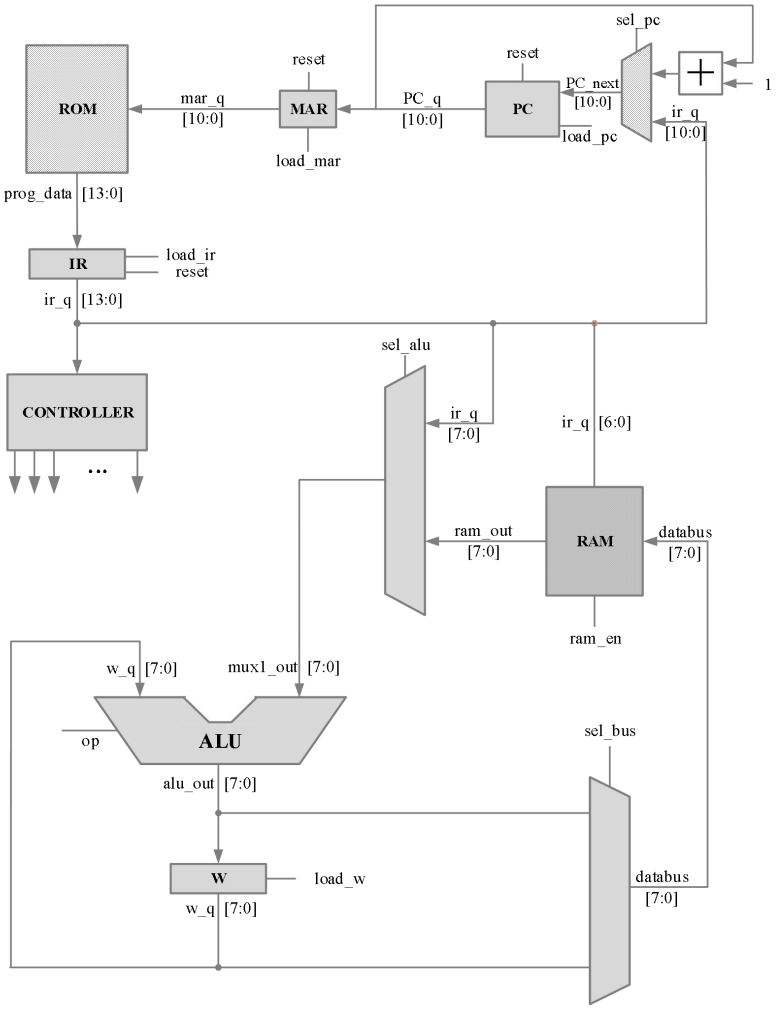
## 注意

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)
                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**

◆ hw_1121.sv

design > hw_1121.sv

```
1   `timescale 1ns/10ps
2   module hw_1121(
3       input clk,
4       input reset,
5       output logic [7:0] w_q
6   );
7       logic [10:0] pc_next, pc_q,mar_q;
8       logic load_pc, load_mar, load_ir_q,load_w;
9       logic [13:0] Rom_out,ir_q;
10      logic reset_ir_q,ram_en;
11      logic [2:0] ps,ns;
12      logic [3:0] op;
13      logic [7:0] alu_q,mux_out,ram_out,databus;
14      logic d;
15      logic sel_alu,sel_pc,sel_bus;
16      logic MOVLW;
17      logic ADDLW;
18      logic SUBLW;
19      logic ANDLW;
20      logic IORLW;
21      logic XORLW;
22      //-------------pc------------
23      assign pc_next = pc_q + 1;            //找到下一個指令
24
25      always_ff @(posedge clk)              //有load信號，再讀取
26      begin
27          if(reset)
28              pc_q <= #1 0;
29          else if(load_pc)
30              pc_q <= #1 pc_next;
31      end
32
33      //------------mar-----------
34      always_ff @(posedge clk)
35      begin
36          if(load_mar)
37              mar_q <= #1 pc_q;
38      end
39
40      //------------ROM-----------
41      Program_Rom rom(Rom_out,mar_q);
42
43      //------------IR------------
44      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(databus,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

    //---------sel_bus----------
    always_comb
    begin
        if(sel_bus == 0) databus = alu_q;
        else databus = w_q;
    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);

    assign INCF  = (ir_q[13:8] == 6'b001010);
    assign IORWF = (ir_q[13:8] == 6'b000100);
    assign MOVF  = (ir_q[13:8] == 6'b001000);
    assign MOVWF = (ir_q[13:8] == 6'b000000);
    assign SUBWF = (ir_q[13:8] == 6'b000010);
    assign XORWF = (ir_q[13:8] == 6'b000110);

    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 if(INCF)  op = 6;
                else if(IORWF) op = 3;
                else if(MOVF)  op = 5;
                else if(SUBWF) op = 1;
                else if(XORWF) op = 4;
                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;
    sel_bus = 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 || INCF || IORWF || MOVF || SUBWF || XORWF)
                    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 if(MOVWF)
                    begin
                        sel_bus = 1;
                        ram_en = 1;
                    end
                else
                    load_w = 1;
                ns = T5;
            end

        T5:                          //空状态
            begin
                ns = T6;
            end
        T6:
            begin
                ns = T1;
            end
    endcase
    end
endmodule
```

◆ 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'h3007;
                 11'h3:   data = 14'h07A5;
                 11'h4:   data = 14'h3005;
                 11'h5:   data = 14'h0AA5;
                 11'h6:   data = 14'h04A5;
                 11'h7:   data = 14'h00A4;
                 11'h8:   data = 14'h0225;
                 11'h9:   data = 14'h0825;
                 11'ha:   data = 14'h06A4;
                 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
```

◆ testbench.sv

testbench.sv M | sim.do | wave.do M

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_1121 hw_1121_1(
        .reset(reset), //()內的變數為tb的變數，"."後面為hw_1121.sv的變數
        .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

testbench.sv M | sim.do | wave.do M | compile.do M

simulation > modelsim > compile.do

```
#vlib work



# ----------------------------------------------------------------
vlog ../tb/testbench.sv
vlog ../../design/hw_1121.sv
vlog ../../design/Program_Rom.sv
vlog ../../design/single_port_ram_128x8.sv
```

◆ sim.do

simulation > modelsim > sim.do

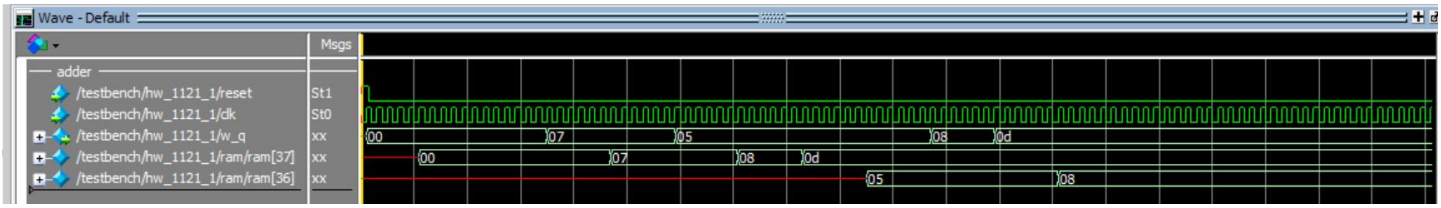
```
vsim -voptargs=+acc work.testbench
view structure wave signals

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_1121_1/reset
add wave -noupdate -format logic    /testbench/hw_1121_1/clk
add wave -noupdate -format Literal -radix Hexadecimal   /testbench/hw_1121_1/w_q
add wave -noupdate -format Literal -radix Hexadecimal   /testbench/hw_1121_1/ram/ram\[37\]
add wave -noupdate -format Literal -radix Hexadecimal   /testbench/hw_1121_1/ram/ram\[36\]
```

## ● 模擬結果與結果說明：




與結果相符~

## ● 結論與心得：

今天是看錄影帶，我想說可以晚上寫功課前再看(我都習慣禮拜一晚上寫硬體作業)，然後一點開影片才發現老師說，先交的分數高，嗚嗚嗚，以後應該要乖乖上課時間看的嗚嗚。這次我截圖做了革新，我之前是截 quartus，然後一次只能截圖 40 行左右，我現在改用 vscode 打開，一次可以截圖 80 行，大大的減少我的截圖時間!!然後今天遇到一個小 bug，我以為我寫錯，之後才發現測試資料是 08xx 我看成 0Bxx 好難過，我找好久嗚嗚。