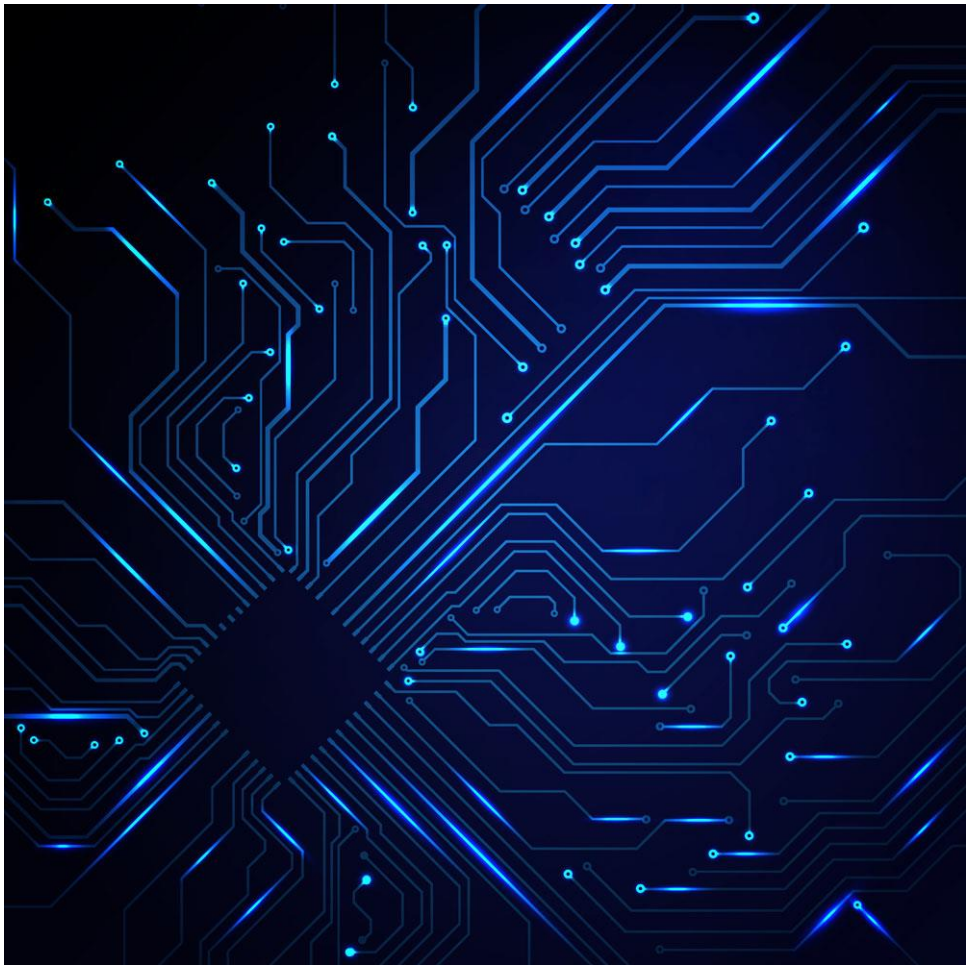


2022/11/14

實驗八

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

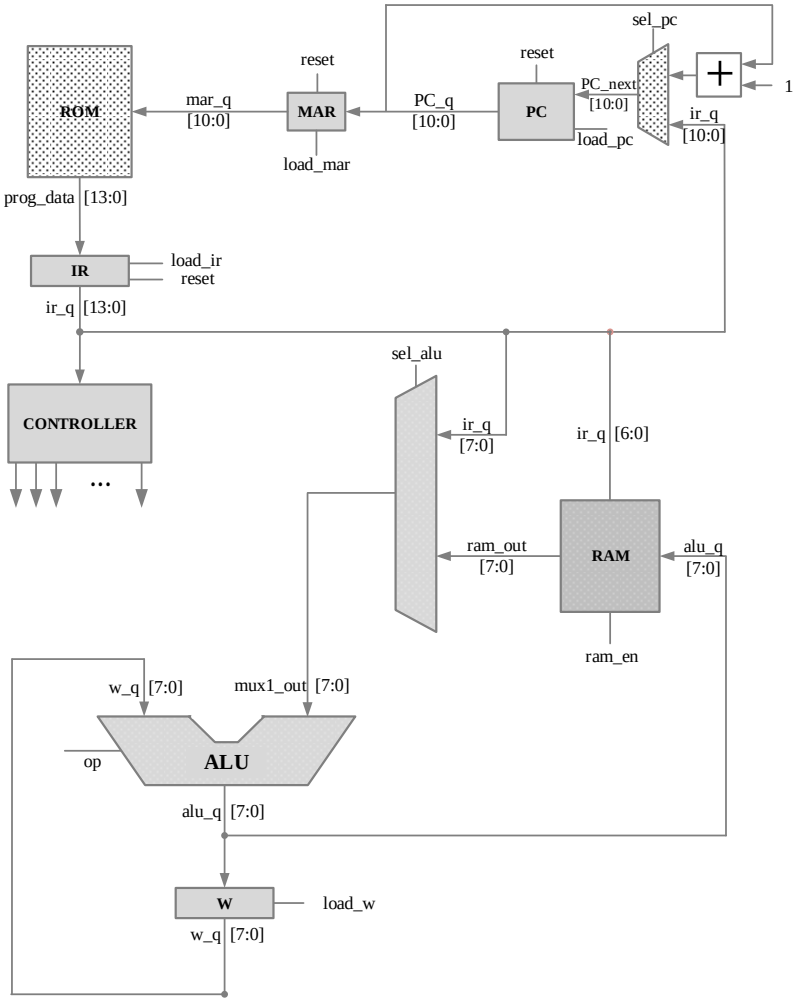
注意

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

```
module Program_Rom (//1114 class lesson
    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
            //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 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 cpu (
    input clk,
    input reset,
    //output logic [13:0] IR
    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;
    logic [3:0] ps,ns;
    //logic [7:0] w_q
    logic [7:0] alu_q, ram_out, mux1_out;
    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;
```

```
51      end
52
53      //RAM
54      single_port_ram_128x8 single_port_ram_128x8_1(
55          .data(alu_q),
56          .addr(ir_q[6:0]),
57          .ram_en(ram_en),
58          .clk(clk),
59          .ram_out(ram_out)
60      );
61
62      //mux 1
63      always_comb begin
64          if(sel_alu) begin
65              mux1_out = ram_out;
66          end
67          else begin
68              mux1_out = ir_q;
69          end
70      end
71
72      assign d = ir_q[7];
73      assign MOVLW = (ir_q[13:8]==6'h30);
74      assign ADDLW = (ir_q[13:8]==6'h3E);
75      assign IORLW = (ir_q[13:8]==6'h38);
76      assign ANDLW = (ir_q[13:8]==6'h39);
77      assign SUBLW = (ir_q[13:8]==6'h3C);
78      assign XORLW = (ir_q[13:8]==6'h3A);
79
80      assign ADDWF = (ir_q[13:8]==6'h07);
81      assign ANDWF = (ir_q[13:8]==6'h05);
82      assign CLRF  = (ir_q[13:8]==6'h01 && d==1);
83      assign CLRW  = (ir_q[13:4]==10'h010 && ir_q[3:2]==2'h0);
84      assign COMF  = (ir_q[13:8]==6'h09);
85      assign DECF  = (ir_q[13:8]==6'h03);
86      assign GOTO  = (ir_q[13:11]==3'b101);
87
88      //ALU
89      ALU ALU_1(
90          .op(op),
91          .w_q(w_q),
92          .mux1_out(mux1_out),
93          .alu_q(alu_q)
94      );
95
96      //register
97      always_ff @( posedge clk ) begin
98          if(load_w)
99              w_q <= alu_q;
100     end
```

```
101
102     //controller
103     parameter T0 = 0;
104     parameter T1 = 1;
105     parameter T2 = 2;
106     parameter T3 = 3;
107     parameter T4 = 4;
108     parameter T5 = 5;
109     parameter T6 = 6;
110
111     always_ff @( posedge clk ) begin
112         if(reset) ps <= 0;
113         else ps <= ns;
114     end
115
116     always_comb begin
117         sel_alu = 0;
118         sel_pc = 0;
119         load_mar = 0;
120         load_pc = 0;
121         reset_ir = 1;//////////////////////////////
122         load_ir = 0;
123         load_w = 0;
124         ram_en=0;
125         op =0;
126         ns=0;
127         case(ps)
128             T0: begin
129                 ns = T1;
130             end
131             T1: begin
132                 load_mar = 1;
133                 load_pc = 0;
134                 reset_ir = 0;
135                 load_ir = 0;
136                 load_w = 0;
137                 ns = T2;
138             end
139             T2: begin
140                 sel_pc = 0;
141                 load_mar = 0;
142                 load_pc = 1;
143                 reset_ir = 0;
144                 load_ir = 0;
145                 load_w = 0;
146                 ns = T3;
147             end
148             T3: begin
149                 load_mar = 0;
150                 load_pc = 0;
```

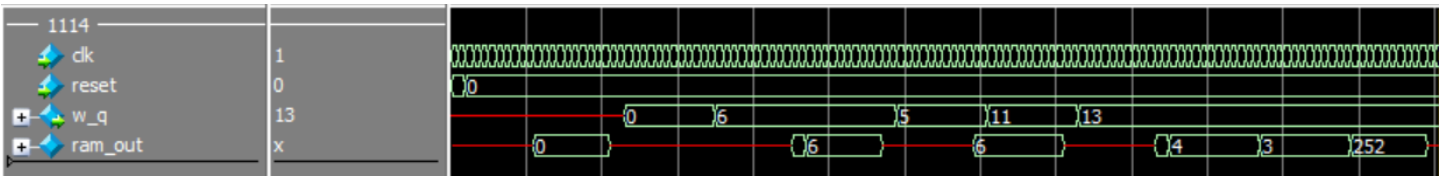
```
151                 reset_ir = 0;
152                 load_ir = 1;
153                 load_w = 0;
154                 ns = T4;
155             end
156             T4: begin
157                 load_mar = 0;
158                 load_pc = 0;
159                 reset_ir = 0;
160                 load_ir = 0;
161 
162                 if(MOVLW) begin
163                     sel_alu = 0;
164                     op = 5;
165                     load_w = 1;
166                 end
167                 else if(ADDLW) begin
168                     sel_alu = 0;
169                     op = 0;
170                     load_w = 1;
171                 end
172                 else if(IORLW) begin
173                     sel_alu = 0;
174                     op = 3;
175                     load_w = 1;
176                 end
177                 else if(ANDLW) begin
178                     sel_alu = 0;
179                     op = 2;
180                     load_w = 1;
181                 end
182                 else if(SUBLW) begin
183                     sel_alu = 0;
184                     op = 1;
185                     load_w = 1;
186                 end
187                 else if(XORLW) begin
188                     sel_alu = 0;
189                     op = 4;
190                     load_w = 1;
191                 end
192 
193 
194                 else if(GOTO) begin
195                     sel_pc = 1;
196                     load_pc = 1;
197                 end
198                 else if(ADDWF) begin
199                     op = 0;
200                     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
                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;
        #1300 $stop;
    end
endmodule
```

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

add wave -noupdate -divider {1114}

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 decimal       /testbench/cpu_test/w_q
add wave -noupdate -format Literal -radix Unsigned       /testbench/cpu_test/ram_out
```

- **模擬結果與結果說明：**



一開始 清空 ram[25]和 w，接下來把 w 設成 6.....

- **結論與心得：**

本周的教學內容和作業又更上一層樓，上週只是從指令裡面去做運算，本周又增加了暫存器，逐漸越來越像一個 CPU 的樣子。除此之外，老師也把期中考的成績加 5 分，非常謝謝老師，也謝謝助教在期中考時在最後一刻讓我 demo，成功過關。