

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

注意

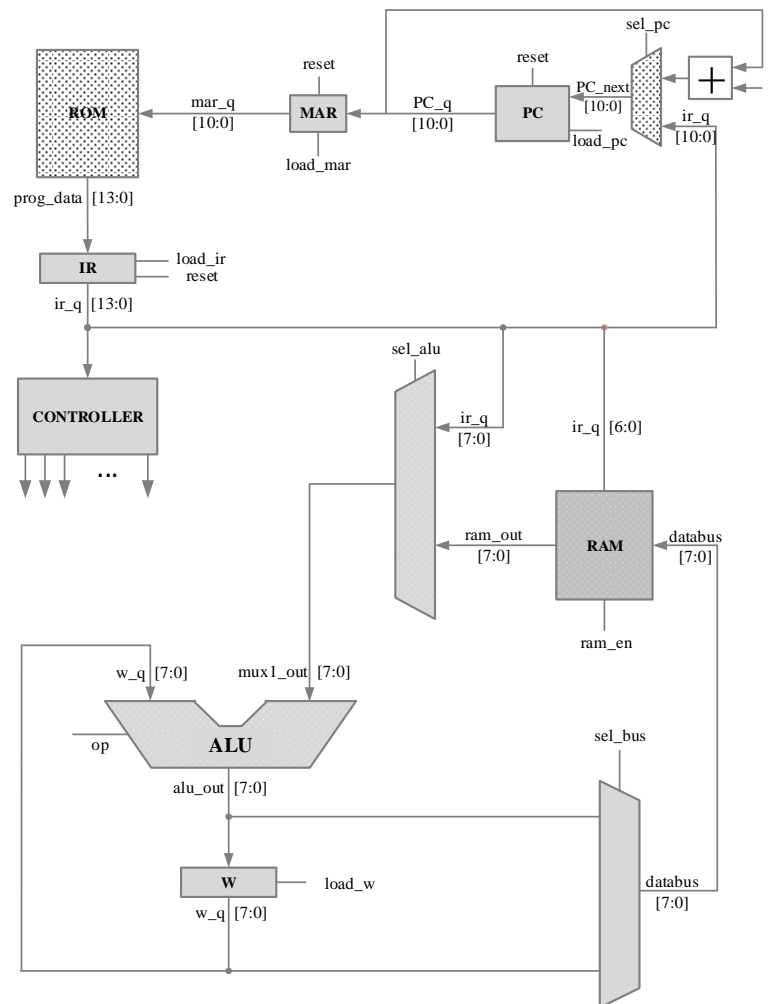
1. 繳交時一律轉 PDF 檔
2. 繳交期限為
上完課後
當週五晚上 12 點前
3. 一人繳交一份
4. 檔名：學號_HW?.pdf
檔名請按照作業檔名格式進行填寫
未依照格式不予批改

● 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; //CLRWF       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

```
1 module single_port_ram_128x8(  
2     input [7:0]data,  
3     input [6:0]addr,  
4     input ram_en,  
5     input clk,  
6     output logic [7:0] ram_out  
7 );  
8     // Declare the RAM variable  
9     //reg [DATA_WIDTH-1:0] ram[2**ADDR_WIDTH-1:0];  
10    logic [7:0] ram[127:0];  
11  
12    always_ff @(posedge clk)  
13    begin  
14        // Write  
15        if (ram_en)  
16            ram[addr] <= data;  
17    end  
18  
19    // Continuous assignment implies read returns NEW data.  
20    // This is the natural behavior of the TriMatrix memory  
21    // blocks in Single Port mode.  
22  
23    assign ram_out = ram[addr];  
24 endmodule  
25  
26
```

```
1 module ALU (  
2     input [3:0] op,  
3     input [7:0] w_q, mux1_out,  
4     output logic [7:0] alu_q  
5 );  
6 always_comb  
7 begin  
8     case(op)  
9         0: alu_q = mux1_out + w_q;  
10        1: alu_q = mux1_out - w_q;  
11        2: alu_q = mux1_out & w_q;  
12        3: alu_q = mux1_out | w_q;  
13        4: alu_q = mux1_out ^ w_q; //XOR  
14        5: alu_q = mux1_out;  
15        6: alu_q = mux1_out + 1;  
16        7: alu_q = mux1_out - 1;  
17        8: alu_q = 0;  
18        9: alu_q = ~mux1_out;  
19        default: alu_q = mux1_out + w_q;  
20    endcase  
21 end  
22  
23 endmodule
```

```
1 module Program_Rom (//1121  
2     output logic [13:0] Rom_data_out,  
3     input [10:0] Rom_addr_in  
4 );  
5     logic [13:0] data;  
6     always_comb begin  
7         case (Rom_addr_in)  
8             11'h0: data = 14'h01A5; //CLRF          ram[25]=0  
9             11'h1: data = 14'h0103; //CLRWF         w=0  
10            11'h2: data = 14'h3007; //MOVLW 7        w=7  
11            11'h3: data = 14'h07A5; //ADDWF 0x25,1    ram[25]=7  
12            11'h4: data = 14'h3005; //MOVLW 5        w=5  
13            11'h5: data = 14'h0AA5; // INCF 0x25,1    ram[25]=8  
14            11'h6: data = 14'h04A5; //IORWF 0x25,1    ram[25]=D  
15            11'h7: data = 14'h00A4; //MOVWF 0x24,1    ram[24]=5  
16            11'h8: data = 14'h0225; //SUBWF 0x25,0    w=8  
17            11'h9: data = 14'h0825; // MOVF 0x25,0    w=D  
18            11'ha: data = 14'h06A4; //XORWF 0x24,1    ram[24]=8  
19            //MPLAB清除暫存器的指令  
20            11'hb: data = 14'h3400;  
21            11'hc: data = 14'h3400;  
22            default: data = 14'h0;  
23        endcase  
24    end  
25    assign Rom_data_out = data;  
26  
27 endmodule
```



```
1 module cpu (
2     input clk,
3     input reset,
4     output logic [7:0] w_q
5 );
6     logic [13:0] rom_q, ir_q;
7     logic [10:0] pc_next, pc_q, mar_q;
8     logic load_pc, load_mar, load_ir, reset_ir, load_w, sel_pc, ram_en, sel_alu, d, sel_bus;
9     logic [3:0] ps,ns;
10    logic [7:0] alu_q, ram_out, mux1_out, databus;
11    logic [3:0] op;
12    logic [5:0] opcode;
13    //mux 0
14    always_comb begin
15        if(sel_pc) begin
16            pc_next = ir_q;
17        end
18        else begin
19            pc_next = pc_q + 1;
20        end
21    end
22
23    //pc
24    always_ff @( posedge clk ) begin
25        if(reset)
26            pc_q <= 0;
27        else if(load_pc)
28            pc_q <= pc_next;
29    end
30
31    //mar
32    always_ff @( posedge clk ) begin
33        if(load_mar)
34            mar_q <= pc_q;
35    end
36
37    //ROM
38    Program_Rom ROM_1(
39        .Rom_addr_in(mar_q),
40        .Rom_data_out(rom_q)
41    );
42
43    //IR
44    always_ff @( posedge clk ) begin
45        if(reset)
46            ir_q <= 0;
47        else if(load_ir)
48            ir_q <= rom_q;
49    end
50
```

```

51 //RAM
52 single_port_ram_128x8 single_port_ram_128x8_1(
53     .data(databus),
54     .addr(ir_q[6:0]),
55     .ram_en(ram_en),
56     .clk(clk),
57     .ram_out(ram_out)
58 );
59
60 //mux 1 (select data into ALU)
61 always_comb begin
62     if(sel_alu) begin
63         mux1_out = ram_out;
64     end
65     else begin
66         mux1_out = ir_q;
67     end
68 end
69
70 assign d = ir_q[7];
71 assign MOVLW = (ir_q[13:8]==6'h30);
72 assign ADDLW = (ir_q[13:8]==6'h3E);
73 assign IORLW = (ir_q[13:8]==6'h38);
74 assign ANDLW = (ir_q[13:8]==6'h39);
75 assign SUBLW = (ir_q[13:8]==6'h3C);
76 assign XORLW = (ir_q[13:8]==6'h3A);
77
78 assign ADDWF = (ir_q[13:8]==6'h07);
79 assign ANDWF = (ir_q[13:8]==6'h05);
80 assign CLRF = (ir_q[13:8]==6'h01 && d==1);
81 assign CLRW = (ir_q[13:4]==10'h010 && ir_q[3:2]==2'h0);
82 assign COMF = (ir_q[13:8]==6'h09);
83 assign DECF = (ir_q[13:8]==6'h03);
84 assign GOTO = (ir_q[13:11]==3'b101);
85
86 assign INCF = (ir_q[13:8]==6'h0A);
87 assign IORWF = (ir_q[13:8]==6'h04);
88 assign MOVF = (ir_q[13:8]==6'h08);
89 assign MOVWF = (ir_q[13:8]==6'h00 && ir_q[7]==1'b1);
90 assign SUBWF = (ir_q[13:8]==6'h02);
91 assign XORWF = (ir_q[13:8]==6'h06);
92
93 //ALU
94 ALU ALU_1(
95     .op(op),
96     .w_q(w_q),
97     .mux1_out(mux1_out),
98     .alu_q(alu_q)
99 );
100

```

```

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

```

```

151         load_w = 0;
152         ns = T2;
153     end
154     T2: begin
155         sel_pc = 0;
156         load_mar = 0;
157         load_pc = 1;
158         reset_ir = 0;
159         load_ir = 0;
160         load_w = 0;
161         ns = T3;
162     end
163     T3: begin
164         load_mar = 0;
165         load_pc = 0;
166         reset_ir = 0;
167         load_ir = 1;
168         load_w = 0;
169         ns = T4;
170     end
171     T4: begin
172         load_mar = 0;
173         load_pc = 0;
174         reset_ir = 0;
175         load_ir = 0;
176
177         if(MOVLW) begin
178             sel_alu = 0;
179             op = 5;
180             load_w = 1;
181         end
182         else if(ADDLW) begin
183             sel_alu = 0;
184             op = 0;
185             load_w = 1;
186         end
187         else if(IORLW) begin
188             sel_alu = 0;
189             op = 3;
190             load_w = 1;
191         end
192         else if(ANDLW) begin
193             sel_alu = 0;
194             op = 2;
195             load_w = 1;
196         end
197         else if(SUBLW) begin
198             sel_alu = 0;
199             op = 1;
200             load_w = 1;

```

```
201         end
202     else if(XORLW) begin
203         sel_alu = 0;
204         op = 4;
205         load_w = 1;
206     end
207
208
209     else if(GOTO) begin
210         sel_pc = 1;
211         load_pc = 1;
212     end
213     else if(ADDWF) begin
214         op = 0;
215         sel_alu = 1;
216         if(d) begin
217             ram_en = 1;
218         end
219         else begin
220             load_w = 1;
221         end
222     end
223     else if(ANDWF) begin
224         op = 2;
225         sel_alu = 1;
226         if(d) begin
227             ram_en = 1;
228         end
229         else begin
230             load_w = 1;
231         end
232     end
233     else if(CLRF) begin
234         op = 8;
235         ram_en = 1;
236     end
237     else if(CLRW) begin
238         op = 8;
239         load_w = 1;
240     end
241     else if(COMF) begin
242         op = 9;
243         sel_alu = 1;
244         ram_en = 1;
245     end
246     else if(DECF) begin
247         op = 7;
248         sel_alu = 1;
249         ram_en = 1;
250     end
```



```

251
252         else if(INCF) begin
253             op = 6;
254             sel_alu = 1;
255             if(d) begin
256                 ram_en = 1;
257                 sel_bus = 0;
258             end
259             else begin
260                 load_w = 1;
261             end
262         end
263         else if(IORWF) begin
264             op = 3;
265             sel_alu = 1;
266             if(d) begin
267                 ram_en = 1;
268                 sel_bus = 0;
269             end
270             else begin
271                 load_w = 1;
272             end
273         end
274         else if(MOVF) begin
275             op = 5;
276             sel_alu = 1;
277             if(d) begin
278                 ram_en = 1;
279                 sel_bus = 0;
280             end
281             else begin
282                 load_w = 1;
283             end
284         end
285         else if(MOVWF) begin
286             ram_en = 1;
287             sel_bus = 1;
288         end
289         else if(SUBWF) begin
290             op = 1;
291             sel_alu = 1;
292             if(d) begin
293                 ram_en = 1;
294                 sel_bus = 0;
295             end
296             else begin
297                 load_w = 1;
298             end
299         end
300         else if(XORWF) begin
301             op = 4;
302             sel_alu = 1;
303             if(d) begin
304                 ram_en = 1;
305                 sel_bus = 0;
306             end
307             else begin
308                 load_w = 1;
309             end
310         end
311         ns = T5;
312     end
313     T5: begin
314         ns = T6;
315     end
316     T6: begin
317         ns = T1;
318     end
319 endcase
320 end
321 endmodule

```

```

1 module testbench;
2
3     logic clk,reset;
4     logic [7:0] w_q;
5
6     cpu cpu_test(
7         .clk(clk),
8         .reset(reset),
9         .w_q(w_q)
10    );
11
12    always #10 clk = ~clk;
13    initial begin
14        clk = 0; reset = 1;
15        #20 reset = 0;
16        #1500 $stop;
17    end
18 endmodule

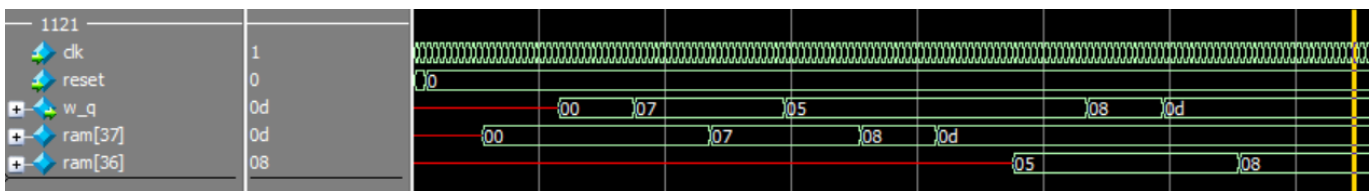
```

```

1 onerror {resume}
2 quietly WaveActivateNextPane {} 0
3
4 add wave -noupdate -divider {1121}
5
6 add wave -noupdate -format Literal -radix decimal      /testbench/cpu_test/clk
7 add wave -noupdate -format Literal -radix decimal      /testbench/cpu_test/reset
8 add wave -noupdate -format Literal -radix Hexadecimal  /testbench/cpu_test/w_q
9 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!