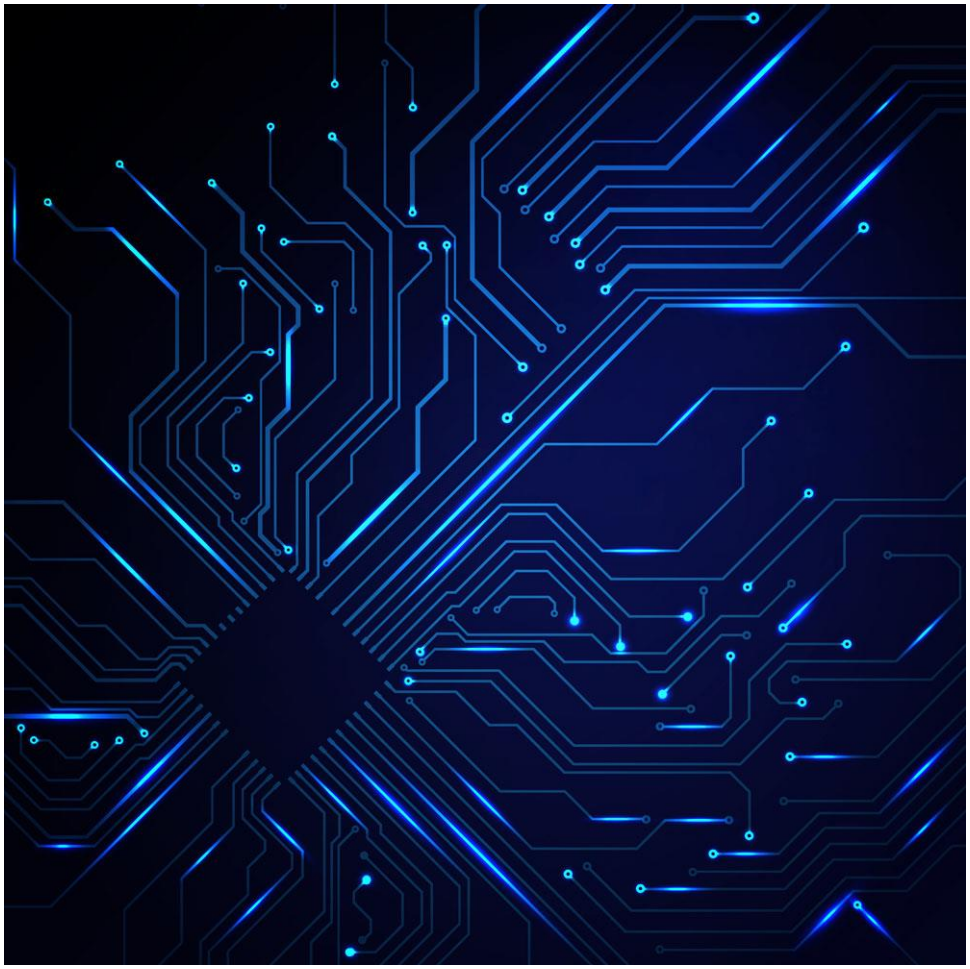


2022/11/07

實驗七

立即數定址

姓名：張銀軒　　學號：00957050

班級：資工 3A

E-mail：00957050@mail.ntou.edu.tw

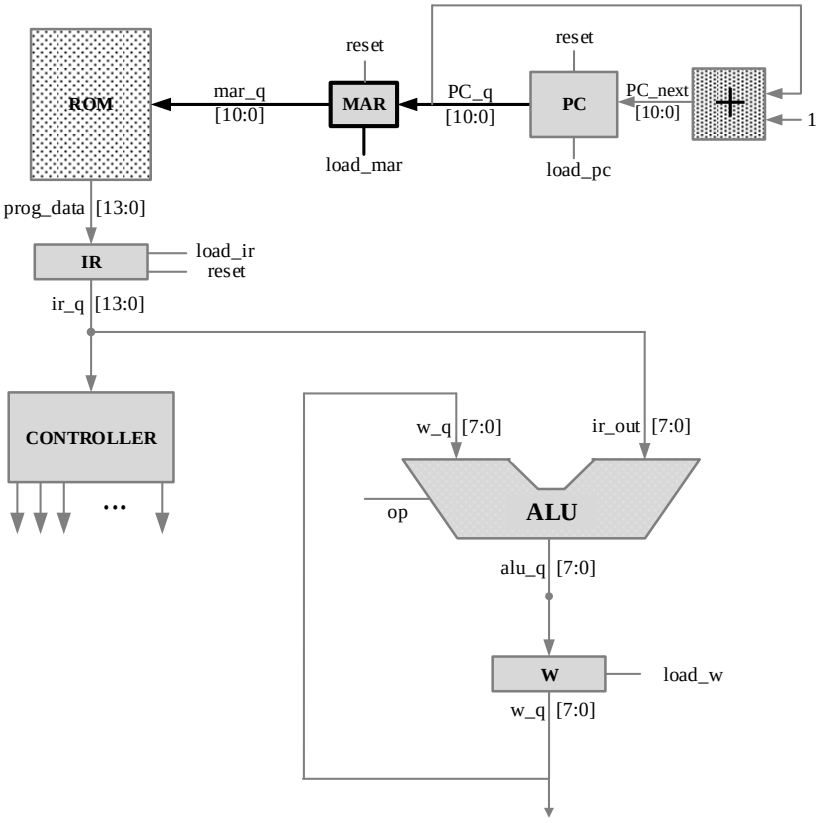
注意

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，設計一個架構實現立即定址的指令
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'h3044;        //MOVLW
            11'h1:  data = 14'h3E01;        //ADDLW
            11'h2:  data = 14'h3802;        //IORLW
            11'h3:  data = 14'h39FE;        //ANDLW
            11'h4:  data = 14'h3C47;        //SUBLW
            11'h5:  data = 14'h3A55;        //XORLW
            11'h6:  data = 14'h3AAA;        //XORLW
            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 Program_Rom (//1107 class lesson
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'h3044;   //MOVLW
9              11'h1: data = 14'h3E01;   //ADDLW
10             11'h2: data = 14'h3802;   //IORLW
11             11'h3: data = 14'h39FE;   //ANDLW
12             11'h4: data = 14'h3C47;   //SUBLW
13             11'h5: data = 14'h3A55;   //XORLW
14             11'h6: data = 14'h3AAA;   //XORLW
15             default: data = 14'h0;
16         endcase
17     end
18
19     assign Rom_data_out = data;
20
21 endmodule
```

```
module cpu (
    input clk,
    input reset,
    output logic [13:0] IR
);
    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;
    logic [3:0] ps,ns;
    logic [7:0] w_q,alu_q;
    logic [3:0] op;
    logic [5:0] opcode;
    //pc
    assign pc_next = pc_q + 1;

    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 <= 0;
        else if(load_ir)
            IR <= rom_q;
    end

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

```

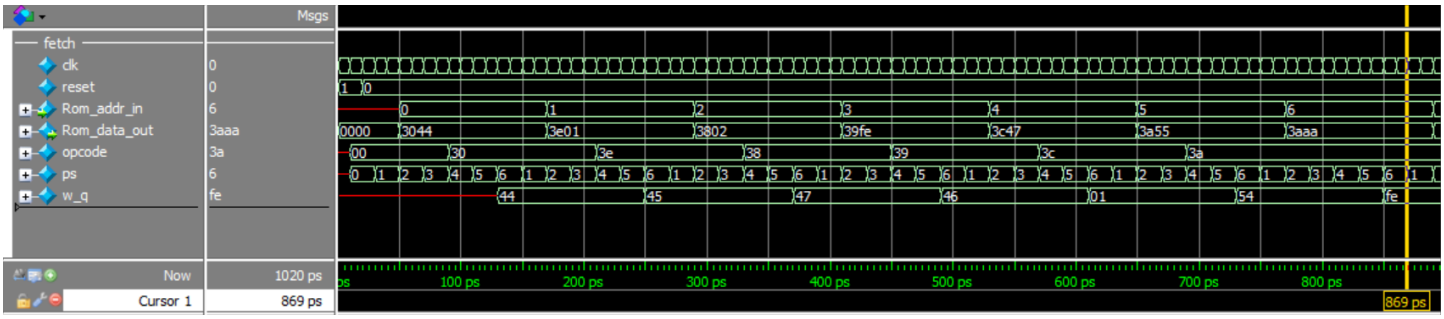
```
51      //ALU
52      ALU ALU_1(
53      .op(op),
54      .w_q(w_q),
55      .ir_q(IR[7:0]),
56      .alu_q(alu_q)
57  );
58
59      always_ff @( posedge clk ) begin
60          if(load_w)
61              w_q <= alu_q;
62      end
63
64      //controller
65      parameter T0 = 0;
66      parameter T1 = 1;
67      parameter T2 = 2;
68      parameter T3 = 3;
69      parameter T4 = 4;
70      parameter T5 = 5;
71      parameter T6 = 6;
72
73      always_ff @( posedge clk ) begin
74          if(reset) ps <= 0;
75          else ps <= ns;
76      end
77
78      always_comb begin
79          load_mar = 0;
80          load_pc = 0;
81          reset_ir = 1;//////////////////////////////
82          load_ir = 0;
83          load_w = 0;
84          op =0;
85          ns=0;
86          case(ps)
87              T0: begin
88                  load_mar = 0;
89                  load_pc = 0;
90                  reset_ir = 0;
91                  load_ir = 0;
92                  load_w = 0;
93                  ns = T1;
94              end
95              T1: begin
96                  load_mar = 1;
97                  load_pc = 0;
98                  reset_ir = 0;
99                  load_ir = 0;
100                 load_w = 0;
```

```
                ns = T2;
            end
            T2: begin
                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;
                load_w = 0;
                ns = T5;
            end
            T5: begin
                load_w = 1;
                if(MOVLW)
                    op = 5;
                else if(ADDLW)
                    op = 0;
                else if(IORLW)
                    op = 3;
                else if(ANDLW)
                    op = 2;
                else if(SUBLW)
                    op = 1;
                else if(XORLW)
                    op = 4;
                ns = T6;
            end
            T6: begin
                ns = T1;
            end
        endcase
    end
endmodule
```

```
module testbench;

    logic clk,reset;
    logic [13:0] ir_out;

    cpu cpu_test(
        .clk(clk),
        .reset(reset),
        .IR(ir_out)
    );

    always #10 clk = ~clk;
    initial begin
        clk = 0; reset = 1;
        #20 reset = 0;
        #1000 $stop;
    end
endmodule
```

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

add wave -noupdate -divider {fetch}

add wave -noupdate -format Literal -radix Unsigned      /testbench/clk
add wave -noupdate -format Literal -radix Unsigned      /testbench/reset
add wave -noupdate -format Literal -radix Unsigned      /testbench/cpu_test/ROM_1/Rom_addr_in
add wave -noupdate -format Literal -radix Hexadecimal       /testbench/cpu_test/ROM_1/Rom_data_out
add wave -noupdate -format Literal -radix Hexadecimal       /testbench/cpu_test/opcode
add wave -noupdate -format Literal -radix Unsigned      /testbench/cpu_test/ps
add wave -noupdate -format Literal -radix Hexadecimal       /testbench/cpu_test/w_q
```

● 模擬結果與結果說明：



能成功從 program_rom 讀出指令並加以執行(44+1=45, 45|2=47...)

● 結論與心得：

這次的作業是從期中考前一週就開始教的內容，從一開始的讀取指令，到期中考的讀取指令，並且用很多邏輯閘去實現指令內容，到這一次的把實現邏輯閘的部份用 ALU 整合起來，一步一腳印，越來越接近真實的 CPU，也謝謝助教在上課時的耐心指導，讓我能夠搞懂整個硬體架構與流程。