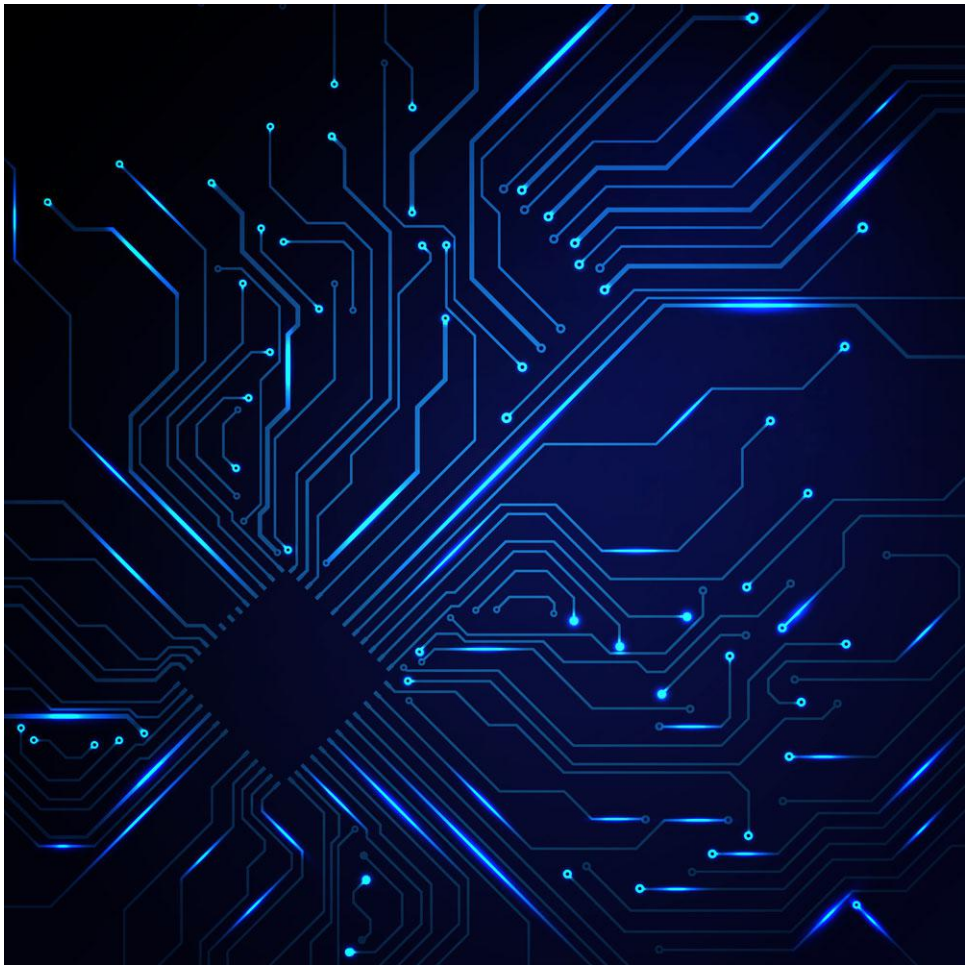


2022/10/24

實驗六

FETCH CYCLE

姓名：王嘉羽　　學號：00957116

班級：資工 3B

E-mail：vayne20011125@gmail.com

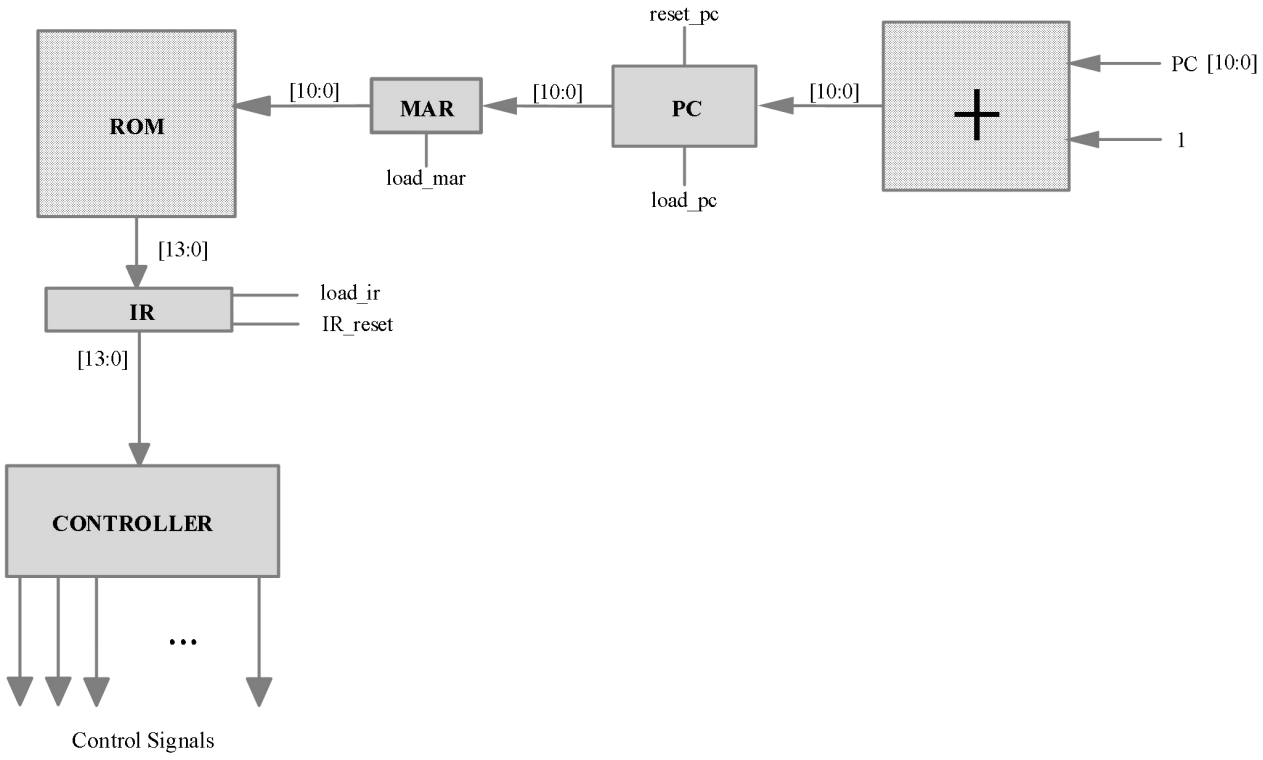
注意

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「提取週期」及「控制單元」
2. 輸入：clk, reset
3. 輸出：IR[13:0]
4. Program_Rom 已提供，其餘部分請自行撰寫

● 系統硬體架構方塊圖（接線圖）：




```
module Program_Rom(Rom_data_out, Rom_adr_in);

//---------
    output  [13:0] Rom_data_out;
//---------
    input   [10:0] Rom_adr_in;
//---------
    reg     [13:0] data;
    wire    [13:0] Rom_data_out;

        always @(Rom_adr_in)
            begin
                case (Rom_adr_in)
                    15'h0 : data = 14'h3044; //MOVLW
                    15'h1 : data = 14'h3E01; //ADDLW
                    15'h2 : data = 14'h3E02; //ADDLW
                    15'h3 : data = 14'h3E03; //ADDLW
                    15'h4 : data = 14'h3E04; //ADDLW
                    15'h5 : data = 14'h3E05; //ADDLW
                    15'h6 : data = 14'h3E06; //ADDLW
                    15'h7 : data = 14'h3E07; //ADDLW
                    default:data = 14'h0;
                endcase
            end

        assign Rom_data_out = data;
endmodule
```

● 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

◆ CPU.sv

```
`timescale 1ns/10ps
module CPU(
    input clk,
    input reset,
    output logic [13:0] IR
);
    logic [10:0] pc_next, pc_q,mar_q;
    logic load_pc, load_mar, load_IR;
    logic [13:0] Rom_out;
    logic reset_IR;
    logic [2:0] ps,ns;
    //-------------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-----------
    ROM rom1(Rom_out,mar_q);

    //------------IR------------
    always_ff @(posedge clk)
    begin
       if(reset_IR)
          IR <= #1 0;
       else if(load_IR)
          IR <= #1 Rom_out;
    end

    //--------controller--------

    parameter T0 = 0;
    parameter T1 = 1;
    parameter T2 = 2;
    parameter T3 = 3;

    always_ff @(posedge clk)
    begin
       if(reset) ps <= #1 0;
       else ps <= #1 ns;
    end

    always_comb
    begin
    load_mar = 0;
    load_pc = 0;
    reset_IR = 0;
    load_IR = 0;
    ns = 0;
       case(ps)
       T0:                          //初始化IR
          begin
             reset_IR = 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
                load_IR = 1;
                ns = T1;                //一直重複T1,T2,T3
             end
     endcase
     end
endmodule
```

◆ ROM.sv

```
`timescale 1ns/10ps
module ROM(
    output [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'h3E02;      //ADDLW
            11'h3:   data = 14'h3E03;      //ADDLW
            11'h4:   data = 14'h3E04;      //ADDLW
            11'h5:   data = 14'h3E05;      //ADDLW
            11'h6:   data = 14'h3E06;      //ADDLW
            11'h7:   data = 14'h3E07;      //ADDLW
            default:data = 14'hX;
            endcase
        end
     assign Rom_data_out = data;

endmodule
```

◆ seven_segment.sv

```
module seven_segment(
    input [3:0] A,          //輸入
    output logic [6:0] y  //輸出
);
always_comb
begin
    //當A改變時，y會對應到七段顯示器，右邊的註解表示他在7段顯示器上的效果
    case(A)
       4'h0 : y = 7'b1000000; //0
       4'h1 : y = 7'b1111001; //1
       4'h2 : y = 7'b0100100; //2
       4'h3 : y = 7'b0110000; //3
       4'h4 : y = 7'b0011001; //4
       4'h5 : y = 7'b0010010; //5
       4'h6 : y = 7'b0000010; //6
       4'h7 : y = 7'b1111000; //7
       4'h8 : y = 7'b0000000; //8
       4'h9 : y = 7'b0010000; //9
       4'hA : y = 7'b0001000; //A
       4'hB : y = 7'b0000011; //b
       4'hC : y = 7'b1000110; //C
       4'hD : y = 7'b0100001; //d
       4'hE : y = 7'b0000110; //E
       4'hF : y = 7'b0001110; //F
    endcase
end
endmodule
```

◆ testbench.sv

simulation > tb > testbench.sv

```
1   `timescale 1ns/10ps
2   module testbench;
3
4       logic reset;                    //重置
5       logic clk;                      //時脈
6       logic [14:0] IR;                    //輸出
7
8       CPU CPU1(
9           .reset(reset), //()內的變數為tb的變數，"."後面為CPU.sv的變數，將2者對應起來
10          .clk(clk),
11          .IR(IR)
12      );
13
14      always #10 clk = ~clk;
15
16      initial begin
17          reset = 1;clk = 0; //一開始先reset，將時脈歸0
18          #15 reset = 0;
19          #1000 $stop;
20      end
21  endmodule
```

◆ mcu.sv(課堂 demo)

```
1
2   //=======================================================
3   //  This code is generated by Terasic System Builder
4   //=======================================================
5
6   module mcu(
7
8       ////////////// CLOCK //////////
9       input CLK,
10
11      ////////////// 7-segment decoder //////////
12      output [6:0] HEX0,
13      output [6:0] HEX1,
14      output [6:0] HEX2,
15      output [6:0] HEX3,
16
17      ////////////// LED //////////////
18      output [9:0] LED,
19
20      ////////////// SWITCH //////////
21      input [9:0] SW,
22
23      ////////////// BUTTON //////////
24      input [2:0] BTN
25
26  );
27  //////////////////////////////////
31  // add module here
32  CPU CPU1(
33      .reset(~BTN[0]),
34      .clk(BTN[2]),
35      .IR({H3,H2,H1,H0})          //將結果存入變數
36  );
37
38  seven_segment s0(
39      .A(H0),         //輸入
40      .y(HEX0)  //輸出
41  );
42
43  seven_segment s1(
44      .A(H1),         //輸入
45      .y(HEX1)  //輸出
46  );
47
48  seven_segment s2(
49      .A(H2),         //輸入
50      .y(HEX2)  //輸出
51  );
52
53  seven_segment s3(
54      .A(H3),         //輸入
55      .y(HEX3)  //輸出
56  );
57  //////////////////////////////////
58  endmodule
```

◆ 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/CPU1/reset
add wave -noupdate -format logic    /testbench/CPU1/clk
add wave -noupdate -format Literal -radix Unsigned      /testbench/CPU1/ps
add wave -noupdate -format Literal -radix Unsigned      /testbench/CPU1/pc_next
add wave -noupdate -format logic    /testbench/CPU1/load_pc
add wave -noupdate -format Literal -radix Unsigned      /testbench/CPU1/pc_q
add wave -noupdate -format logic    /testbench/CPU1/load_mar
add wave -noupdate -format Literal -radix Unsigned      /testbench/CPU1/mar_q
add wave -noupdate -format logic    /testbench/CPU1/load_IR
add wave -noupdate -format Literal -radix Hexadecimal       /testbench/CPU1/IR
```

◆ compile.do

simulation > modelsim > compile.do

```
#vlib work



# ------------------------------------------------------------
vlog ../tb/testbench.sv
vlog ../../design/CPU.sv
vlog ../../design/ROM.sv

```

◆ sim.do

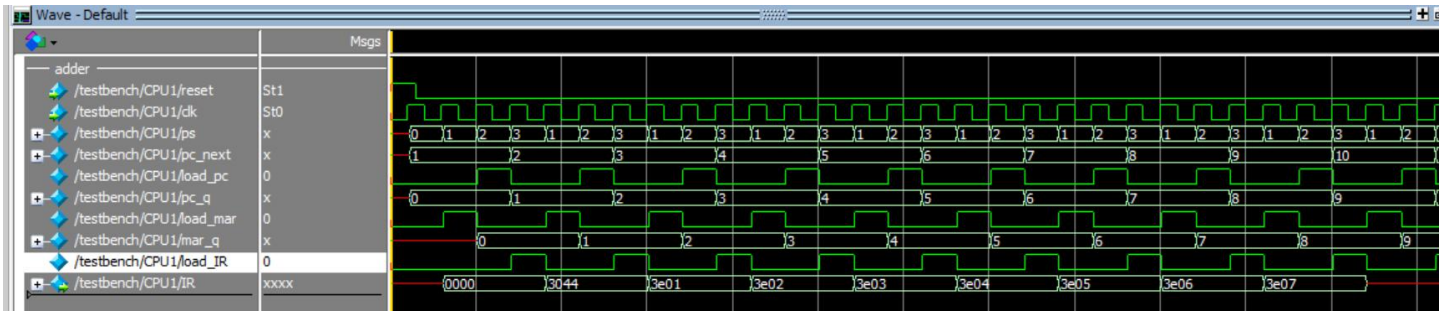
simulation > modelsim > sim.do

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

do wave.do

log -r *
run -all
```

● 模擬結果與結果說明：



跟 ppt 圖幾乎一樣~

● 結論與心得：

今天教的東西，感覺很複雜，但寫起來又還好；寫起來很好寫，但又感覺似懂非懂的 qq。

一定是因為助教有先示範寫，才會感覺好寫，而且有給波形圖，就更好寫了。下禮拜就要段考了，希望考試可以順順利利，有時候作業都寫好久，不確定考試能不能很快寫完...也希望老師能早日康復。