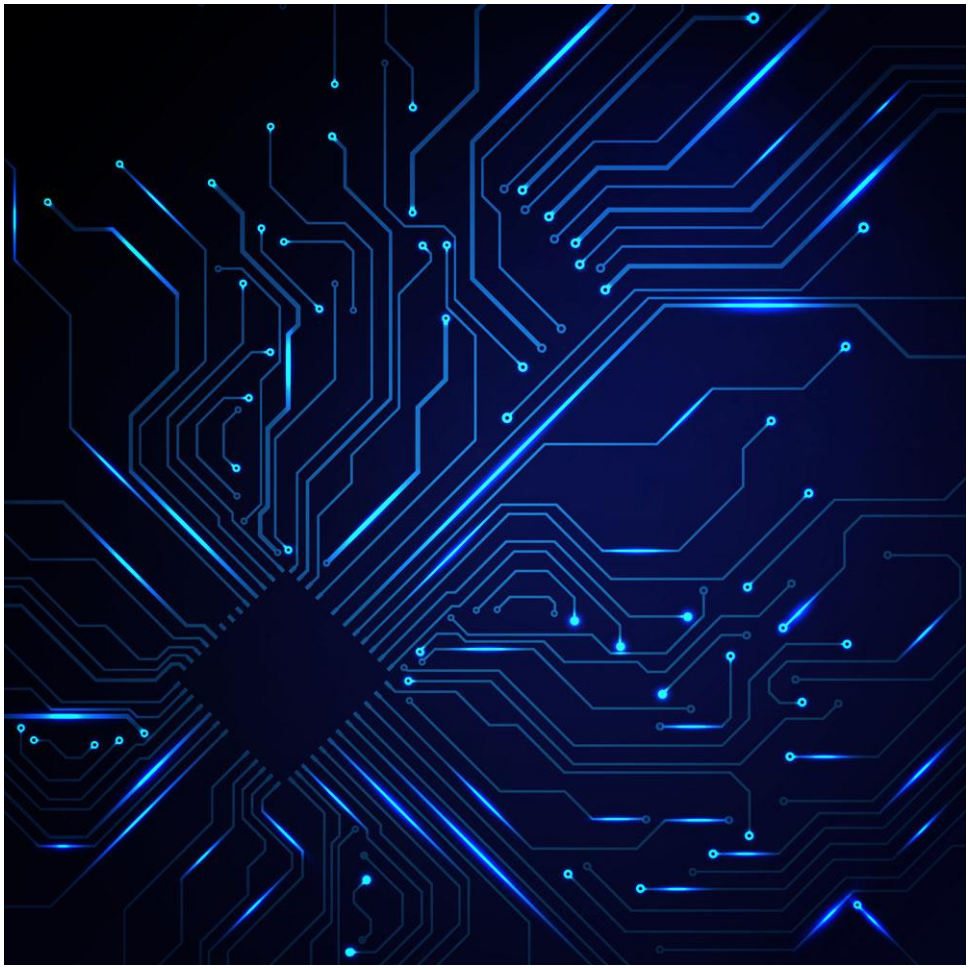


注意

1. 繳交時一律轉 PDF 檔
2. 一人繳交一份
3. 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

2022/12/19

實驗十四

進階組合語言

姓名：張銀軒　　學號：00957050

班級：資工 3A

E-mail：00957050@mail.ntou.edu.tw

2022/12/19

● 實驗說明：

第一題:

■ 設計組合語言，用來顯示自己的學號，運行於PIC MCU上。

■ 輸出到PORT_B

C語言:

```
while(1)
{
    cout<< 0;
    cout<< 0;
    cout<< 6;
    cout<< 5;
    cout<< 7;
    cout<< 0;
    cout<< 0;
    cout<< 3;
}
```

第二題:

```
a 0x25 c 0x24  answer 0x23
answer = a * c;
```

■ 把answer輸出到PORT_B

C語言:

```
int a = 5;
int c = 3;
int count = c;
int answer = 0;
while(1)
{
    answer = answer + a;
    count --;
    if(count==0)    //decfsz
    {
        break;
    }

}
cout << answer;
```

第三題:

```
a 0x25 c 0x24  answer 0x23
answer = a / c;
```

■ 把answer輸出到PORT_B

C語言:

```
int count =0;
int a = 21;
int c = 3;
int temp =0;
while(1)
{
    a = a-c;
    count++;
    if(a<0)          //btfss a 7 判斷第8個bit是不是1
    {
        break;
    }
}
    count --;
    cout << count    //count = a/c
    temp =0-a;
    mod = c-temp
    cout << mod      //a%c;
```

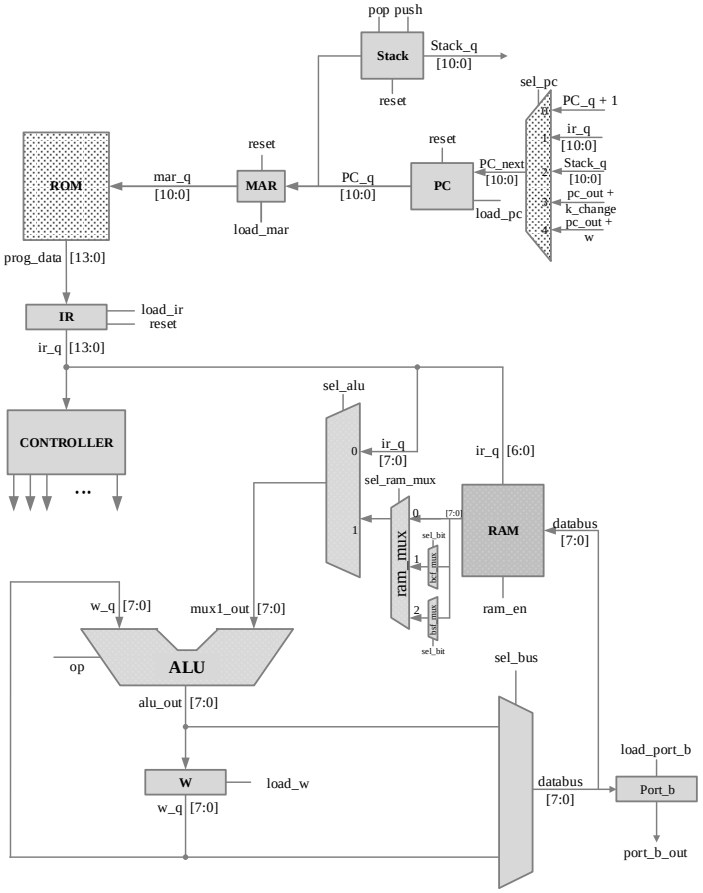
第四題:

輾轉相除法(可以用迴圈暴力解)

```
a 0x25 b 0x24  answer(最大公因數) 0x23
```

```
int a = 36;
int c = 21;
int temp;
while(1)
{
    temp = a%c;
    a=c;
    c=temp;
    if(c<=0)          //不能小於0，也不能等於0
    {                 //先做btfss c,7 跳過代表小於0 再做 w = b; decf w;  btfss w,7
                      //跳過代表等於0 都沒跳過則都是goto同一個地方
        break;
    }

}

cout <<a;
```

● 系統硬體架構方塊圖（接線圖）：



● 系統架構程式碼、測試資料程式碼與程式碼說明

截圖請善用 win+shift+S

```
module ALU (
    input [3:0] op,
    input [7:0] w_q, mux1_out,
    output logic [7:0] alu_q
);
always_comb
begin
    case(op)
        4'h0: alu_q = mux1_out[7:0] + w_q;
        4'h1: alu_q = mux1_out[7:0] - w_q;
        4'h2: alu_q = mux1_out[7:0] & w_q;
        4'h3: alu_q = mux1_out[7:0] | w_q;
        4'h4: alu_q = mux1_out[7:0] ^ w_q; //XOR
        4'h5: alu_q = mux1_out[7:0];
        4'h6: alu_q = mux1_out[7:0] + 1;
        4'h7: alu_q = mux1_out[7:0] - 1;
        4'h8: alu_q = 0;
        4'h9: alu_q = ~mux1_out[7:0];
        4'hA: alu_q = { mux1_out[7],mux1_out[7:1] };    //ASRF arithmetic right shift
        4'hB: alu_q = { mux1_out[6:0], 1'b0 };          //LSLF logical left shift
        4'hC: alu_q = { 1'b0, mux1_out[7:1] };          //LSRF logical right shift
        4'hD: alu_q = { mux1_out[6:0], mux1_out[7] };   //RLF rotate left f
        4'hE: alu_q = { mux1_out[0], mux1_out[7:1] };   //RRF rotate right f
        4'hF: alu_q = { mux1_out[3:0], mux1_out[7:4] }; //{m7, m6,...m4, m3,...m0} => {m3,...m0, m7, m6,...m4}
        default: alu_q = mux1_out[7:0] + w_q;
    endcase
end

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 Stack (
    output logic [10:0] stack_out,
    input [10:0] stack_in,
    input push,
    input pop,
    input reset,
    input clk
);
    logic [3:0] stk_ptr;
    logic [10:0] stack [15:0];
    logic [3:0] stk_index;

    assign stk_index = stk_ptr + 1;
    assign stack_out = stack[stk_ptr];

    always_ff @( posedge clk ) begin
        if(reset) begin
            stk_ptr <= 4'b1111;
        end
        else if(push) begin
            stack[stk_index] <= stack_in;
            stk_ptr <= stk_ptr + 1;
        end
        else if(pop) begin
            stk_ptr <= stk_ptr - 1 ;
        end
    end

endmodule
```

```
module cpu (
    input clk,
    input reset,
    output logic [7:0] port_b_out
);
    logic [13:0] rom_q, ir_q;
    logic [10:0] pc_next, pc_q, mar_q, k_change;
    logic load_pc, load_mar, load_ir, reset_ir, load_w, ram_en, sel_alu, d, sel_bus;
    logic load_port_b;
    logic [2:0] sel_pc;
    logic [3:0] ps,ns;
    logic [7:0] w_q, alu_q, ram_out, mux1_out, bcf_mux, bsf_mux, ram_mux;
    logic [7:0] databus;
    logic [3:0] op;
    logic [5:0] opcode;
    logic [2:0] sel_bit;
    logic [1:0] sel_ram_mux;
    //for Stack
    logic pop, push;
    logic [10:0] stack_out;

    //Stack
    Stack Stack_1(
        .stack_out(stack_out),
        .stack_in(pc_q),
        .push(push),
        .pop(pop),
        .reset(reset),
        .clk(clk)
    );

    assign w_change = {3'b0, w_q} - 1;
    assign k_change = {ir_q[8], ir_q[8], ir_q[8:0]}-1 ;

    //mux 0 (select next PC address)
    always_comb begin
        if(sel_pc == 4) begin
            pc_next = pc_q + w_change;
        end
        else if(sel_pc == 3) begin
            pc_next = pc_q + k_change;
        end
        else if(sel_pc == 2) begin
            pc_next = stack_out;
        end
        else if(sel_pc == 1) 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)
            ir_q <= 0;
        else if(load_ir)
            ir_q <= rom_q;
    end

    //RAM
    single_port_ram_128x8 single_port_ram_128x8_1(
        .data(databus),
        .addr(ir_q[6:0]),
        .ram_en(ram_en),
        .clk(clk),
        .ram_out(ram_out)
    );

    assign sel_bit = ir_q[9:7];

    //BCF mux
    always_comb begin
        case (sel_bit)
            3'b000: bcf_mux = ram_out & 8'b1111_1110;
            3'b001: bcf_mux = ram_out & 8'b1111_1101;
            3'b010: bcf_mux = ram_out & 8'b1111_1011;
            3'b011: bcf_mux = ram_out & 8'b1111_0111;
            3'b100: bcf_mux = ram_out & 8'b1110_1111;
```

```
            3'b101: bcf_mux = ram_out & 8'b1101_1111;
            3'b110: bcf_mux = ram_out & 8'b1011_1111;
            3'b111: bcf_mux = ram_out & 8'b0111_1111;
        endcase
    end

    //BSF mux
    always_comb begin
        case (sel_bit)
            3'b000: bsf_mux = ram_out | 8'b0000_0001;
            3'b001: bsf_mux = ram_out | 8'b0000_0010;
            3'b010: bsf_mux = ram_out | 8'b0000_0100;
            3'b011: bsf_mux = ram_out | 8'b0000_1000;
            3'b100: bsf_mux = ram_out | 8'b0001_0000;
            3'b101: bsf_mux = ram_out | 8'b0010_0000;
            3'b110: bsf_mux = ram_out | 8'b0100_0000;
            3'b111: bsf_mux = ram_out | 8'b1000_0000;
        endcase
    end

    //ram mux (select data into mux1)
    always_comb begin
        case (sel_ram_mux)
            0: ram_mux = ram_out;
            1: ram_mux = bcf_mux;
            2: ram_mux = bsf_mux;
        endcase
    end

    //mux 1 (select data into ALU)
    always_comb begin
        if(sel_alu) begin
            mux1_out = ram_mux;
        end
        else begin
            mux1_out = ir_q;
        end
    end

    assign d = ir_q[7];
    assign  MOVLW = (ir_q[13:8]==6'h30);
    assign  ADDLW = (ir_q[13:8]==6'h3E);
    assign  IORLW = (ir_q[13:8]==6'h38);
    assign  ANDLW = (ir_q[13:8]==6'h39);
    assign  SUBLW = (ir_q[13:8]==6'h3C);
    assign  XORLW = (ir_q[13:8]==6'h3A);

    assign  ADDWF = (ir_q[13:8]==6'h07);
    assign  ANDWF = (ir_q[13:8]==6'h05);
    assign   CLRF = (ir_q[13:8]==6'h01 && d==1);
```

```
assign   CLRW = (ir_q[13:4]==10'h010 && ir_q[3:2]==2'h0);
assign   COMF = (ir_q[13:8]==6'h09);
assign   DECF = (ir_q[13:8]==6'h03);
assign   GOTO = (ir_q[13:11]==3'b101);

assign   INCF = (ir_q[13:8]==6'h0A);
assign  IORWF = (ir_q[13:8]==6'h04);
assign   MOVF = (ir_q[13:8]==6'h08);
assign  MOVWF = (ir_q[13:8]==6'h00 && ir_q[7]==1'b1);
assign  SUBWF = (ir_q[13:8]==6'h02);
assign  XORWF = (ir_q[13:8]==6'h06);

assign    BCF = (ir_q[13:12]==2'b01 && ir_q[11:10]==2'b00);
assign    BSF = (ir_q[13:12]==2'b01 && ir_q[11:10]==2'b01);
assign  BTFSC = (ir_q[13:12]==2'b01 && ir_q[11:10]==2'b10);
assign  BTFSS = (ir_q[13:12]==2'b01 && ir_q[11:10]==2'b11);
assign DECFSZ = (ir_q[13:8]==6'h0B);
assign INCFSZ = (ir_q[13:8]==6'h0F);

assign btfsc_skip_bit = (ram_out[ir_q[9:7]]==0);
assign btfss_skip_bit = (ram_out[ir_q[9:7]]==1);
assign btfsc_btfss_skip_bit = (BTFSC&btfsc_skip_bit) |
                              (BTFSS&btfss_skip_bit);
assign   ASRF = (ir_q[13:8]==6'h37);        // arithmetic right shift
assign   LSLF = (ir_q[13:8]==6'h35);        // logical left shift
assign   LSRF = (ir_q[13:8]==6'h36);        // logical right shift
assign    RLF = (ir_q[13:8]==6'h0D);        // rotate left f
assign    RRF = (ir_q[13:8]==6'h0C);        // rotate right f
assign   SWAP = (ir_q[13:8]==6'h0E); //{m7, m6,...m4, m3,...m0} => {m3,...m0, m7, m6,...m4}

assign   CALL = (ir_q[13:12]==2'b10 && ir_q[11]==0);
assign RETURN = (ir_q == 8);

assign    BRA = (ir_q[13:12]==2'b11 && ir_q[11:9]==3'b001);// relative branch
assign    BRW = (ir_q[13:0]==14'h000B);      // relative branch with W
assign    NOP = (ir_q[13:0]==14'h0000);      // no operation

//ALU
ALU ALU_1(
    .op(op),
    .w_q(w_q),
    .mux1_out(mux1_out),
    .alu_q(alu_q)
);
assign aluout_zero = (alu_q == 0);

//register
always_ff @( posedge clk ) begin
    if(load_w)
        w_q <= alu_q;
```

```
    end

    //mux 2 (select data into RAM)
    always_comb begin
        if(sel_bus) begin
            databus = w_q;
        end
        else begin
            databus = alu_q;
        end
    end

    //Port_b
    always_ff @( posedge clk ) begin
        if(reset) begin
            port_b_out <= 0;
        end
        else if(load_port_b) begin
            port_b_out <= databus;
        end
    end

    logic [3:0] ten,unit;
    always_comb begin
        ten = port_b_out / 10;
        unit = port_b_out - ten * 10;
    end

    assign addr_port_b = (ir_q[6:0] == 7'h0D);

    //controller
    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 <= 0;
        else ps <= ns;
    end

    always_comb begin
        sel_alu = 0;
        sel_pc = 0;
        load_mar = 0;
        load_pc = 0;
        reset_ir = 0;
```

```
        load_ir = 0;
        load_w = 0;
        ram_en = 0;
        op = 0;
        sel_ram_mux = 0;
        sel_bus = 0;
        load_port_b = 0;
        ns=0;
        //for Stack
        push = 0;
        pop = 0;
        case(ps)
            T0: begin
                ns = T1;
            end
            T1: begin
                load_mar = 1;
                sel_pc = 0;
                load_pc = 1;
                ns = T2;
            end
            T2: begin
                ns = T3;
            end
            T3: begin
                load_ir = 1;
                ns = T4;
            end
            T4: begin
                load_mar = 1;
                sel_pc = 2'b00;
                load_pc = 1;
                if(GOTO) begin//to skip the following instruction in T4
                    load_mar = 1;
                end
                else if(MOVLW) begin
                    sel_alu = 0;
                    op = 5;
                    load_w = 1;
                end
                else if(ADDLW) begin
                    sel_alu = 0;
                    op = 0;
                    load_w = 1;
                end
                else if(IORLW) begin
                    sel_alu = 0;
                    op = 3;
                    load_w = 1;
                end
```

```
                else if(ANDLW) begin
                    sel_alu = 0;
                    op = 2;
                    load_w = 1;
                end
                else if(SUBLW) begin
                    sel_alu = 0;
                    op = 1;
                    load_w = 1;
                end
                else if(XORLW) begin
                    sel_alu = 0;
                    op = 4;
                    load_w = 1;
                end

                else if(ADDWF) begin
                    op = 0;
                    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

            else if(INCF) begin
                op = 6;
                sel_alu = 1;
                if(d) begin
                    ram_en = 1;
                    sel_bus = 0;
                end
                else begin
                    load_w = 1;
                end
            end
```

```
367                 else if(IORWF) begin
368                     op = 3;
369                     sel_alu = 1;
370                     if(d) begin
371                         ram_en = 1;
372                         sel_bus = 0;
373                     end
374                     else begin
375                         load_w = 1;
376                     end
377                 end
378                 else if(MOVF) begin
379                     op = 5;
380                     sel_alu = 1;
381                     if(d) begin
382                         ram_en = 1;
383                         sel_bus = 0;
384                     end
385                     else begin
386                         load_w = 1;
387                     end
388                 end
389                 else if(MOVWF) begin
390                     sel_bus = 1;
391                     if(addr_port_b)begin
392                         load_port_b = 1;
393                     end
394                     else begin
395                         ram_en = 1;
396                     end
397                 end
398                 else if(SUBWF) begin
399                     op = 1;
400                     sel_alu = 1;
```

```
                    if(d) begin
                        ram_en = 1;
                        sel_bus = 0;
                    end
                    else begin
                        load_w = 1;
                    end
                end
                else if(XORWF) begin
                    op = 4;
                    sel_alu = 1;
                    if(d) begin
                        ram_en = 1;
                        sel_bus = 0;
                    end
                    else begin
                        load_w = 1;
                    end
                end
                else if(BCF)begin
                    sel_alu = 1;
                    sel_ram_mux = 1;
                    op = 5;
                    sel_bus = 0;
                    ram_en = 1;
                end
                else if(BSF)begin
                    sel_alu = 1;
                    sel_ram_mux = 2;
                    op = 5;
                    sel_bus = 0;
                    ram_en = 1;
                end
```

```
434
435                 else if(ASRF) begin
436                     sel_alu = 1;
437                     sel_ram_mux = 0;
438                     op = 4'hA;
439                     if(d)begin
440                         sel_bus = 0;
441                         ram_en = 1;
442                     end
443                     else begin
444                         load_w = 1;
445                     end
446                 end
447                 else if(LSLF) begin
448                     sel_alu = 1;
449                     sel_ram_mux = 0;
450                     op = 4'hB;
451                     if(d)begin
452                         sel_bus = 0;
453                         ram_en = 1;
454                     end
455                     else begin
456                         load_w = 1;
457                     end
458                 end
459                 else if(LSRF) begin
460                     sel_alu = 1;
461                     sel_ram_mux = 0;
462                     op = 4'hC;
463                     if(d)begin
464                         sel_bus = 0;
465                         ram_en = 1;
466                     end
```

```
                    else begin
                        load_w = 1;
                    end
                end
                else if(RLF) begin
                    sel_alu = 1;
                    sel_ram_mux = 0;
                    op = 4'hD;
                    if(d)begin
                        sel_bus = 0;
                        ram_en = 1;
                    end
                    else begin
                        load_w = 1;
                    end
                end
                else if(RRF) begin
                    sel_alu = 1;
                    sel_ram_mux = 0;
                    op = 4'hE;
                    if(d)begin
                        sel_bus = 0;
                        ram_en = 1;
                    end
                    else begin
                        load_w = 1;
                    end
                end
                else if(SWAP) begin
                    sel_alu = 1;
                    sel_ram_mux = 0;
                    op = 4'hF;
                    if(d)begin
                        sel_bus = 0;
```

```
                    ram_en = 1;
                end
                else begin
                    load_w = 1;
                end
            end

            else if(CALL) begin
                push = 1;
            end
            ns = T5;
        end
        T5: begin
            if(GOTO) begin
                sel_pc = 2'b01;
                load_pc = 1;
            end
            else if(CALL)begin
                sel_pc = 2'b01;
                load_pc = 1;
            end
            else if(RETURN) begin
                pop = 1;
                sel_pc = 2'b10;
                load_pc = 1;
            end
            else if(BRA) begin
                load_pc = 1;
                sel_pc = 3;
            end
            else if(BRW) begin
                load_pc = 1;
                sel_pc = 4;
```

```
                end
                ns = T6;
            end
            T6: begin
                load_ir = 1;
                if(GOTO || CALL || RETURN || BRA || BRW) begin
                    reset_ir = 1;
                end
                else if(DECFSZ)begin
                    op = 7;
                    sel_alu = 1;
                    if(aluout_zero)begin
                        reset_ir = 1;
                    end
                    if(d)begin
                        sel_bus = 0;
                        ram_en = 1;
                    end
                    else begin
                        load_w = 1;
                    end
                end
                else if(INCFSZ) begin
                    op = 6;
                    sel_alu = 1;
                    if(aluout_zero)begin
                        reset_ir = 1;
                    end
                    if(d)begin
                        sel_bus = 0;
                        ram_en = 1;
                    end
                    else begin
```

```
                                load_w = 1;
                            end
                        end
                        else if(BTFSC || BTFSS)begin
                            if( btfsc_btfss_skip_bit )begin
                                reset_ir = 1;
                            end
                        end
                        ns = T4;
                    end
                endcase
            end
        endmodule
```

第 1 題

C:\...\Program_Rom.asm

```
#include    <p16Lf1826.inc>

temp    equ 0x25

;_______Program start
    org 0x00

        clrw
loop    movlw   .0
        movwf   PORTB
        movlw   .0
        movwf   PORTB
        movlw   .9
        movwf   PORTB
        movlw   .5
        movwf   PORTB
        movlw   .7
        movwf   PORTB
        movlw   .0
        movwf   PORTB
        movlw   .5
        movwf   PORTB
        movlw   .0
        movwf   PORTB
        goto    loop

        end
```

```
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'h0103;
                10'h1 : data = 14'h3000;
                10'h2 : data = 14'h008D;
                10'h3 : data = 14'h3000;
                10'h4 : data = 14'h008D;
                10'h5 : data = 14'h3009;
                10'h6 : data = 14'h008D;
                10'h7 : data = 14'h3005;
                10'h8 : data = 14'h008D;
                10'h9 : data = 14'h3007;
                10'ha : data = 14'h008D;
                10'hb : data = 14'h3000;
                10'hc : data = 14'h008D;
                10'hd : data = 14'h3005;
                10'he : data = 14'h008D;
                10'hf : data = 14'h3000;
                10'h10 : data = 14'h008D;
                10'h11 : data = 14'h2801;
                10'h12 : data = 14'h3400;
                10'h13 : data = 14'h3400;
                default: data = 14'h0;
            endcase
        end

     assign Rom_data_out = data;

endmodule
```

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

add wave -noupdate -divider {student number to port_b_out}

add wave -noupdate -format Literal -radix Unsigned      /testbench/clk
add wave -noupdate -format Logic -radix decimal         /testbench/reset
add wave -noupdate -format Literal -radix Unsigned      /testbench/cpu_test/port_b_out
add wave -noupdate -format Literal -radix Hexadecimal       /testbench/cpu_test/ir_q
```

第 2 題

C:\...\Program_Rom.asm

```
#include    <p16Lf1826.inc>

a       equ 0x25
c       equ 0x24
answer  equ 0x23
count   equ 0x22

;________Program start
        org 0x00

        movlw   .5  ; int a=5
        movwf   a

        movlw   .3  ; int c=3
        movwf   c

        ; int count=c
        movf    c,0     ; move c to w
        movwf   count   ; move w to count

        ; int answer = 0
        clrw            ; w=0
        movwf   answer  ; answer=w

loop    movf    a,0         ; w=a
        addwf   answer,1    ; answer= answer+w
        decfsz  count
        goto    loop

        movf    answer,0    ; w=answer
        movwf   PORTB
        end
```

```
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'h3005;
                10'h1 : data = 14'h00A5;
                10'h2 : data = 14'h3003;
                10'h3 : data = 14'h00A4;
                10'h4 : data = 14'h0824;
                10'h5 : data = 14'h00A2;
                10'h6 : data = 14'h0103;
                10'h7 : data = 14'h00A3;
                10'h8 : data = 14'h0825;
                10'h9 : data = 14'h07A3;
                10'ha : data = 14'h0BA2;
                10'hb : data = 14'h2808;
                10'hc : data = 14'h0823;
                10'hd : data = 14'h008D;
                10'he : data = 14'h3400;
                10'hf : data = 14'h3400;
                default: data = 14'h0;
            endcase
        end

     assign Rom_data_out = data;

endmodule

```

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

add wave -noupdate -divider {multiple two numbers}

add wave -noupdate -format Literal -radix Unsigned      /testbench/clk
add wave -noupdate -format Logic -radix decimal         /testbench/reset
add wave -noupdate -format Literal -radix Unsigned      /testbench/cpu_test/port_b_out
```

第 3 題

C:\...\Program_Rom.asm

```
#include    <p16Lf1826.inc>

a       equ 0x25
c       equ 0x24
answer  equ 0x23    ; ram[35]
count   equ 0x22    ; ram[34]
temp    equ 0x21
mod     equ 0x20

;________Program start_________
        org 0x00
        ; int count = 0
        clrw            ; w = 0
        movwf   count

        movlw   .21 ; int a = 21
        movwf   a

        movlw   .3  ; int c=3
        movwf   c

        ; int temp = 0
        clrw
        movwf   temp

loop    movf    c,0     ; w = c
        subwf   a,1     ; a = a - w

        incf    count,1 ; count++

        btfss   a,7     ; if(a<0) break;
        goto loop

        decf    count,1 ; count--

        ; answer = a / c
        movf    count,0 ; w = count
        movwf   answer  ; answer = w

        ; cout << count
        movf    count,0 ; w = count
        movwf   PORTB   ; port_b = w

        ; temp = 0 - a
        movf    a,0     ; w = a
        sublw   0       ; w = 0 - w
        movwf   temp

        ; mod = c - temp
        movf    temp,0  ; w = temp
        subwf   c,0     ; w = c - w
        movwf   PORTB   ; port_b = w
        end
```

```
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'h0103;
                10'h1 : data = 14'h00A2;
                10'h2 : data = 14'h3015;
                10'h3 : data = 14'h00A5;
                10'h4 : data = 14'h3003;
                10'h5 : data = 14'h00A4;
                10'h6 : data = 14'h0103;
                10'h7 : data = 14'h00A1;
                10'h8 : data = 14'h0824;
                10'h9 : data = 14'h02A5;
                10'ha : data = 14'h0AA2;
                10'hb : data = 14'h1FA5;
                10'hc : data = 14'h2808;
                10'hd : data = 14'h03A2;
                10'he : data = 14'h0822;
                10'hf : data = 14'h00A3;
                10'h10 : data = 14'h0822;
                10'h11 : data = 14'h008D;
                10'h12 : data = 14'h0825;
                10'h13 : data = 14'h3C00;
                10'h14 : data = 14'h00A1;
                10'h15 : data = 14'h0821;
                10'h16 : data = 14'h0224;
                10'h17 : data = 14'h008D;
                10'h18 : data = 14'h3400;
                10'h19 : data = 14'h3400;
                default: data = 14'h0;
            endcase
        end

     assign Rom_data_out = data;

endmodule

```

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

add wave -noupdate -divider {divide two numbers}

add wave -noupdate -format Literal -radix Unsigned      /testbench/clk
add wave -noupdate -format Logic -radix decimal         /testbench/reset
add wave -noupdate -format Literal -radix Unsigned      /testbench/cpu_test/single_port_ram_128x8_1/ram\[35\]
add wave -noupdate -format Literal -radix Unsigned      /testbench/cpu_test/port_b_out
```

第 4 題

C:\...\Program_Rom.asm

```
#include    <p16Lf1826.inc>
aTwo    equ 0x26    ; ram[38]
a       equ 0x25    ; ram[37]
c       equ 0x24    ; ram[36]
answer  equ 0x23    ; ram[35]
count   equ 0x22    ; ram[34]
temp    equ 0x21    ; ram[33]
mod     equ 0x20    ; ram[32]
cmOne   equ 0x19    ; ram[31]

;________Program start_________
        org 0x00

        movlw   .36 ; int a = 36
        movwf   a

        movlw   .21 ; int c = 21
        movwf   c

        ; int count = 0
loop1   clrw            ; w = 0
        movwf   count

        ; aTwo = a
        movf    a,0     ; w = a
        movwf   aTwo    ; aTwo = w

; loop for calculating mod
loop2   movf    c,0     ; w = c
        subwf   aTwo,1  ; aTwo = aTwo - w

        incf    count,1 ; count++
        btfss   aTwo,7  ; if(aTwo<0) break;
        goto loop2

        ; temp = a % c = aTwo + c
        movf    aTwo,0  ; w = aTwo
        movwf   temp    ; temp = w

        movf    c,0     ; w = c
        addwf   temp,1  ; temp = temp + w

        ; a = c
        movf    c,0 ; w = c
        movwf   a       ; a = w

        ; c = temp
        movf    temp,0  ; w = temp
        movwf   c       ; c = w

        ; if (c<=0) break
        btfss   c,7     ; skip if less than 0
        goto check
        goto final

check   movf    c,0     ; w = c
        movwf   cmOne   ; cmOne = w
        decf    cmOne,1
        btfss   cmOne,7
        goto loop1

final   movf    a,0     ; w = a
        movwf   answer  ; answer = a
        end
```

```
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'h3024;
                10'h1 : data = 14'h00A5;
                10'h2 : data = 14'h3015;
                10'h3 : data = 14'h00A4;
                10'h4 : data = 14'h0103;
                10'h5 : data = 14'h00A2;
                10'h6 : data = 14'h0825;
                10'h7 : data = 14'h00A6;
                10'h8 : data = 14'h0824;
                10'h9 : data = 14'h02A6;
                10'ha : data = 14'h0AA2;
                10'hb : data = 14'h1FA6;
                10'hc : data = 14'h2808;
                10'hd : data = 14'h0826;
                10'he : data = 14'h00A1;
                10'hf : data = 14'h0824;
                10'h10 : data = 14'h07A1;
                10'h11 : data = 14'h0824;
                10'h12 : data = 14'h00A5;
                10'h13 : data = 14'h0821;
                10'h14 : data = 14'h00A4;
                10'h15 : data = 14'h1FA4;
                10'h16 : data = 14'h2818;
                10'h17 : data = 14'h281D;
                10'h18 : data = 14'h0824;
                10'h19 : data = 14'h0099;
                10'h1a : data = 14'h0399;
                10'h1b : data = 14'h1F99;
                10'h1c : data = 14'h2804;
                10'h1d : data = 14'h0825;
                10'h1e : data = 14'h00A3;
                10'h1f : data = 14'h3400;
                10'h20 : data = 14'h3400;
                default: data = 14'h0;
            endcase
        end

     assign Rom_data_out = data;

endmodule

```

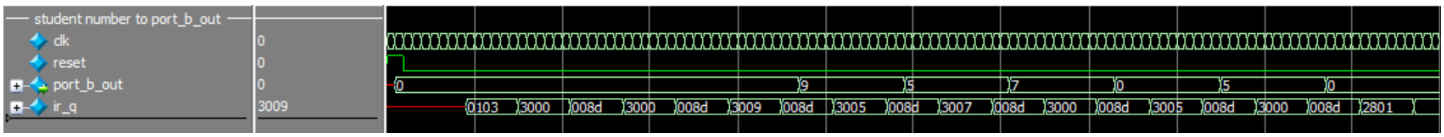
```
onerror {resume}
quietly WaveActivateNextPane {} 0

add wave -noupdate -divider {Euclidean}

add wave -noupdate -format Literal -radix Unsigned      /testbench/clk
add wave -noupdate -format Logic -radix decimal         /testbench/reset
add wave -noupdate -format Literal -radix Unsigned      /testbench/cpu_test/single_port_ram_128x8_1/ram\[37\]
add wave -noupdate -format Literal -radix Unsigned      /testbench/cpu_test/single_port_ram_128x8_1/ram\[36\]
add wave -noupdate -format Literal -radix Unsigned      /testbench/cpu_test/single_port_ram_128x8_1/ram\[35\]
```

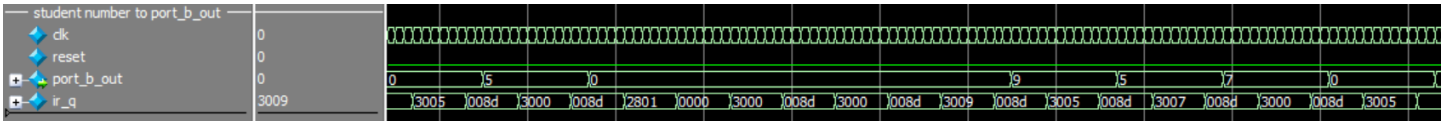
● 模擬結果與結果說明：

第 1 題

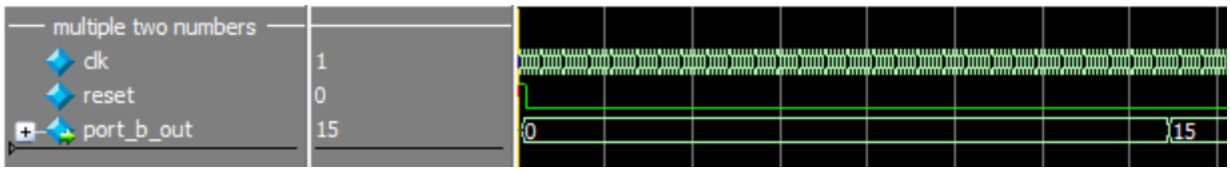
因為我的學號頭尾都是 0，所以輸出時比較看不出差別，但時實際上的確有依照我的學號 0->0->9->5->7->0->5->0 一直循環

5 後面都是 0，當中在做了 goto、因為 goto 的關係 ir_q 是被 reset、把 w 的值改成 0，把 w 的值 load 到 port_b_out、把 w 的值改成 0，把 w 的值 load 到 port_b_out、把 w 的值改成 9



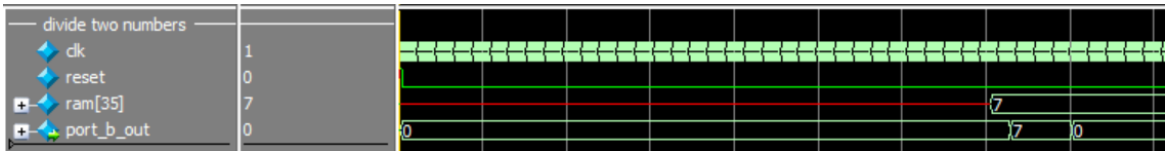


第 2 題



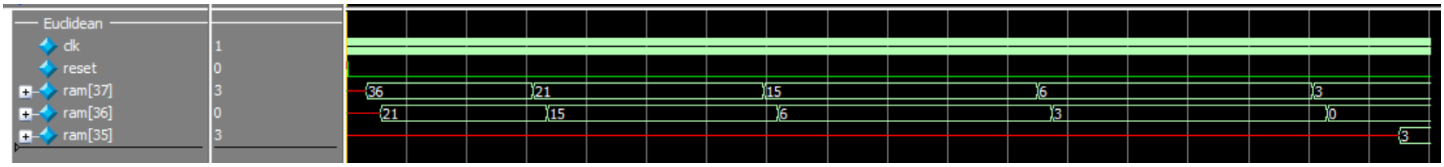
如同題目要求，計算 5*3，port_b_out 成功輸出結果

第 3 題



如同題目要求，計算 21/3 的商(7)和餘數(0)，並且輸出在 port_b_out。因為題目上方有 answer = a/c; 但是下方程式碼裡面卻沒有，所以我把 answer = a/c 的實作加在 count—後面，同時也在波型圖上呈現 answer(0x23=>ram[35])的值

第 4 題



a :　　ram[37]

c :　　ram[36]

answer: ram[35]

1. a = 36, c =21, temp =15
2. a = 21, c = 15, temp = 6
3. a = 15, c = 6, temp = 3
4. a = 6, c = 3, temp = 0
5. a = 3, c = 0, temp = 3，因為這時候 c=0 所以結束計算並且把答案(a)輸出在 answer

- **結論與心得：**

這次的作業很特別，雖然有 4 個小題，但是實際上是按部就班，從乘法、除法...一步一步慢慢讓我學會用組合語言模擬輾轉相除法。前 3 題都沒有遇到甚麼狀況，第 4 題的 if(c<=0) break; 真的讓我費煞苦心，想方設法都無法達到做完 2 次判斷之後再決定是否 break，所以最終只好先判斷 <0，true 就直接 break，false 再去做=0 的判斷。