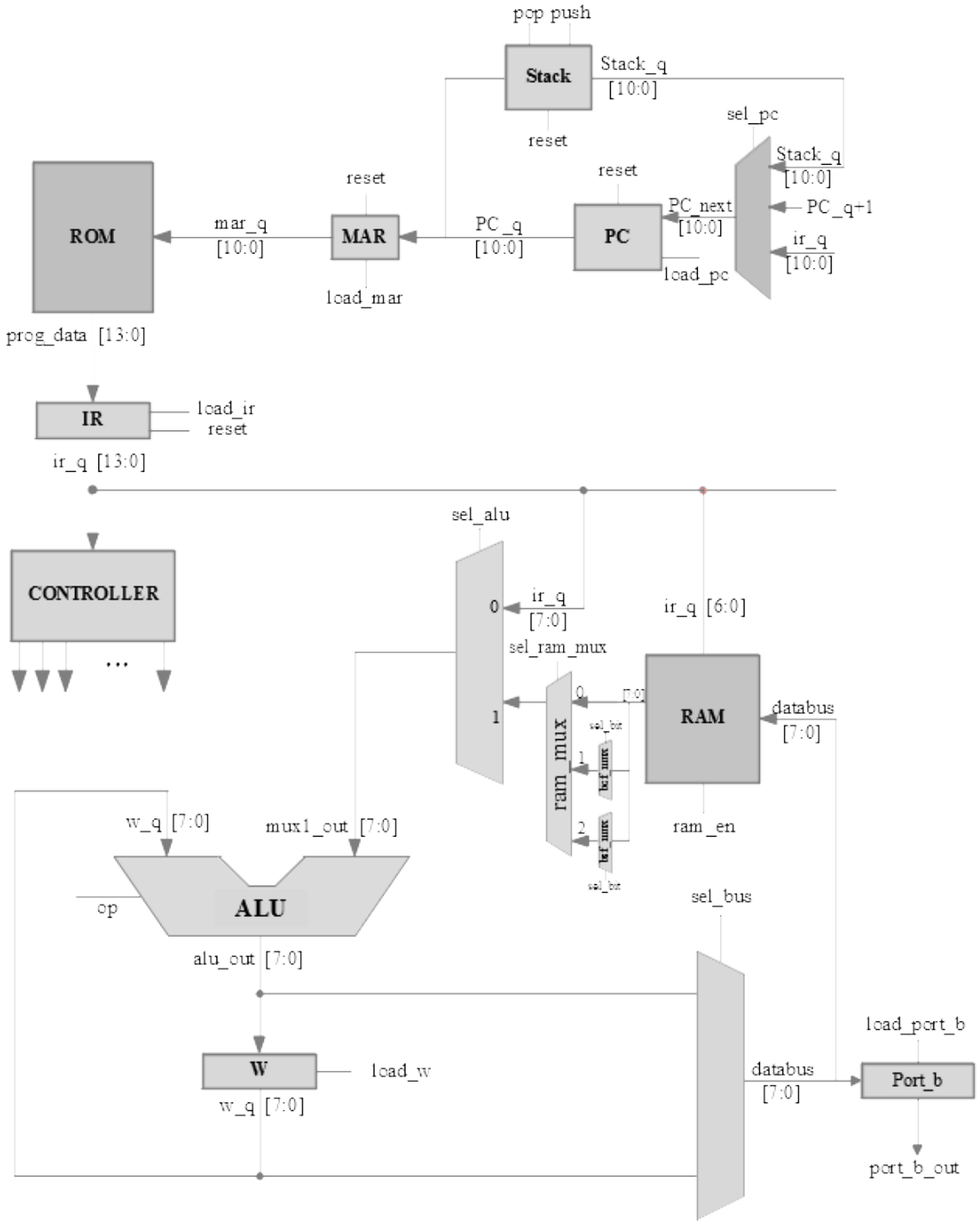
● 實驗說明：

用 MPLAB 設計一個 Rom，使 0x21 和 0x22 兩個位址的 16 進制分別表示時鐘的分及秒，即 0x22(秒)的 16 進制會由 1 數到 59 後歸零，每當 0x22(秒)歸零 0x21(分)就會加 1

請交 MPLAB 專案及程式碼截圖，存分跟秒的暫存器請分別設定為 0x21 跟 0x22

● 系統硬體架構方塊圖（接線圖）：



架構圖

● 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

檔案:cpu.V:

```
module CPU(
  input clk,
  input reset,
  output reg [7:0] port_b_out
);
  reg [13:0] ir_out;
  reg [10:0] pc_out;
  reg [10:0] mar_out;
  wire [13:0] ir_in;
  reg [10:0] pc_in;
  reg [7:0] alu_q;

  reg [3:0]op;
  reg load_w;
  reg load_pc;
  reg load_mar;
  reg load_ir;
  reg [2:0] ps,ns;

  reg [7:0]databus;
  reg ram_en;
  wire [7:0]ram_out;
  reg sel_alu;
  reg sel_bus;
  reg [2:0]sel_pc;
  reg [7:0]mux1_out;

  wire [2:0]sel_bit;
  reg [1:0]sel_RAM_mux;
  reg [7:0]bsf_mux;
  reg [7:0]bcf_mux;
  reg [7:0]RAM_mux;
  wire alu_zero;

  reg [7:0]w_q;
  reg load_port_b;

  reg pop;
  reg push;
  wire [10:0]stack_q;


  always @(*)
  begin
    case(sel_pc)
    0:pc_in = pc_out + 1;
    1:pc_in = ir_out[10:0];
    2:pc_in = stack_q;
    endcase
  end
  //mux_pc
  always @(posedge clk)
  begin
    if(reset)
      pc_out <= 0;
    else if(load_pc)
      pc_out <= pc_in;
  end
  //pc
  Stack st(.stack_out(stack_q),.stack_in(pc_c
  //Stack
  always @(posedge clk)
  begin
    if(reset)
      mar_out <= 0;
    else if(load_mar)
      mar_out <= pc_out;
  end
  //mar
  Program_Rom Rom(.Rom_data_out(ir_in),.Rom_a
  //ROM
  always @(posedge clk)
  begin
    if(reset)
      ir_out <= 0;
    else if(load_ir)
      ir_out <= ir_in;
  end
  //IR
  single_port_ram_128x8 Ram(
    .data(databus),
    .addr(ir_out[6:0]),
    .en(ram_en),
    .clk(clk),
    .q(ram_out)
    );

  assign btfsc_skip_bit = ram_out[ir_out[9:7]]==0
  assign btfss_skip_bit = ram_out[ir_out[9:7]]==1


  //RAM
  assign sel_bit = ir_out[9:7];
  //sel_bit
  always@(*)
    begin
        case(sel_RAM_mux)
            0:RAM_mux = ram_out;
            1:RAM_mux = bcf_mux;
            2:RAM_mux = bsf_mux;
            default : RAM_mux = 8'bx;
        endcase
    end
  //RAM_mux
  always@(*)
    begin
        case(sel_bit)
            3'b000 : bcf_mux = ram_out & 8'hFE;
            3'b001 : bcf_mux = ram_out & 8'hFD;
            3'b010 : bcf_mux = ram_out & 8'hFB;
            3'b011 : bcf_mux = ram_out & 8'hF7;
            3'b100 : bcf_mux = ram_out & 8'hEF;
            3'b101 : bcf_mux = ram_out & 8'hDF;
            3'b110 : bcf_mux = ram_out & 8'hBF;
            3'b111 : bcf_mux = ram_out & 8'h7F;
        endcase
    end
  //BCF_MUX
    always@(*)
    begin
        case(sel_bit)
            3'b000 : bsf_mux = ram_out | 8'h01;
            3'b001 : bsf_mux = ram_out | 8'h02;
            3'b010 : bsf_mux = ram_out | 8'h04;
            3'b011 : bsf_mux = ram_out | 8'h08;
            3'b100 : bsf_mux = ram_out | 8'h10;
            3'b101 : bsf_mux = ram_out | 8'h20;
            3'b110 : bsf_mux = ram_out | 8'h40;
            3'b111 : bsf_mux = ram_out | 8'h80;
        endcase
    end
  //BCF_MUX
  always @(*)
  begin
    case(sel_alu)
    0: mux1_out = ir_out[7:0];
    1: mux1_out = RAM_mux;
    endcase
  end
  //mux1
  always @(*)
  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;
    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;
    10: alu_q = {mux1_out[7],mux1_out[7:1]};
    11: alu_q = {mux1_out[6:0],1'b0};
    12: alu_q = {1'b0,mux1_out[7:1]};
    13: alu_q = {mux1_out[6:0],mux1_out[7]};
```

```
    13: alu_q = {mux1_out[6:0],mux1_out[7]};
    14: alu_q = {mux1_out[0],mux1_out[7:1]};
    15: alu_q = {mux1_out[3:0],mux1_out[7:4]};
    default: alu_q = mux1_out[7:0] + w_q;
    endcase
  end

  assign alu_zero = (alu_q == 0)? 1'b1: 1'b0;

  //alu
  always @(*)
  begin
    case(sel_bus)
    0: databus = alu_q;
    1: databus = w_q;
    endcase
  end
  //bus
  always @(posedge clk)
    if(reset) port_b_out <= 0;
    else if(load_port_b) port_b_out <= databus;

  assign addr_port_b = (ir_out[6:0] == 7'h0d);
  //port_b
  always @(posedge clk)
  begin
    if(load_w)
        w_q <= alu_q;
  end
  //w
  assign MOVLW = (ir_out[13:8] == 6'b11_0000);
  assign ADDLW = (ir_out[13:8] == 6'b11_1110);
  assign SUBLW = (ir_out[13:8] == 6'b11_1100);
  assign ANDLW = (ir_out[13:8] == 6'b11_1001);
  assign IORLW = (ir_out[13:8] == 6'b11_1000);
  assign XORLW = (ir_out[13:8] == 6'b11_1010);

  assign ADDWF = (ir_out[13:8] == 6'b00_0111);
  assign ANDWF = (ir_out[13:8] == 6'b00_0101);
  assign CLRF = (ir_out[13:8] == 6'b00_0001);
  assign CLRW = (ir_out[13:2] == 12'b00_0001000000);
  assign COMF = (ir_out[13:8] == 6'b00_1001);
  assign DECF = (ir_out[13:8] == 6'b00_0011);
  assign GOTO = (ir_out[13:11] == 3'b10_1);

  assign INCF = (ir_out[13:8] == 6'b00_1010);
  assign IORWF = (ir_out[13:8] == 6'b00_0100);
  assign MOVF = (ir_out[13:8] == 6'b00_1000);
  assign MOVWF = (ir_out[13:7] == 7'b00_00001);
  assign SUBWF = (ir_out[13:8] == 6'b00_0010);
  assign XORWF = (ir_out[13:8] == 6'b00_0110);

  assign DECFSZ = (ir_out[13:8] == 6'b00_1011);
  assign INCFSZ = (ir_out[13:8] == 6'b00_1111);

  assign BCF = (ir_out[13:10] == 4'b01_00);
  assign BSF = (ir_out[13:10] == 4'b01_01);
  assign BTFSC = (ir_out[13:10] == 4'b01_10);
  assign BTFSS = (ir_out[13:10] == 4'b01_11);
  assign btfsc_btfss_skip_bit = (BTFSC&btfsc_skip_bit)|(BTFSS&btfss_skip_bit);

  assign ASRF = (ir_out[13:8] == 6'b11_0111);
  assign LSLF = (ir_out[13:8] == 6'b11_0101);
  assign LSRF = (ir_out[13:8] == 6'b11_0110);
  assign RLF = (ir_out[13:8] == 6'b00_1101);
  assign RRF = (ir_out[13:8] == 6'b00_1100);
  assign SWAPF = (ir_out[13:8] == 6'b00_1110);

  assign CALL = (ir_out[13:11] == 3'b10_0);
  assign RETURN = (ir_out[13:0] == 14'b00_0000000001000);

  always @(posedge clk)
  begin
    if(reset)
      ps <= 0;
    else
      ps <= ns;
  end
  always@(*)
```

```
235     always@(*)
236     begin
237       load_pc=0;
238       load_ir=0;
239       load_mar=0;
240       load_w=0;
241       ram_en=0;
242       sel_RAM_mux=0;
243       load_port_b=0;
244       sel_pc=0;
245       pop=0;
246       push=0;
247
248       ns=0;
249       case(ps)
250         0: ns = 1;
251         1:
252         begin
253           load_mar=1;
254           ns = 2;
255         end
256         2:
257         begin
258           sel_pc=0;
259           load_pc=1;
260           ns = 3;
261         end
262         3:
263         begin
264           load_ir=1;
265           ns = 4;
266         end
267         4:
268         begin
269           if (MOVLW|ADDLW|SUBLW|ANDLW|IORLW|XORLW)
270               begin
271                   sel_alu=0;
272                   load_w = 1;
273                   if (MOVLW)
274                       op=5;
275                   else if (ADDLW)
276                       op=0;
277                   else if (IORLW)
278                       op=3;
279                   else if (SUBLW)
280                       op=1;
281                   else if (ANDLW)
282                       op=2;
283                   else if (XORLW)
284                       op=4;
285               end
286           ns = 1;
287           if (GOTO)
288               begin
289                   sel_pc=1;
290                   load_pc=1;
291               end
292           if (ADDWF)
293               begin
294               op=0;
295               sel_alu=1;
296               if (ir_out[7]==0)
297                   load_w=1;
298               else
299                   begin
300                   ram_en=1;
301                   sel_bus=0;
302                   end
303               end
304           if (ANDWF)
305               begin
306               op=2;
307               sel_alu=1;
308               if (ir_out[7]==0)
309                   load_w=1;
310               else
311                   begin
312                   ram_en=1;
313                   sel_bus=0;
```

```
314             end
315         end
316         if (CLRF)
317             begin
318             op=8;
319             ram_en=1;
320             sel_bus=0;
321             end
322         if (CLRW)
323             begin
324             op=8;
325             load_w=1;
326             end
327         if (COMF)
328             begin
329             op=9;
330             sel_alu=1;
331             if (ir_out[7]==0)
332                 load_w=1;
333             else
334                 begin
335                 ram_en=1;
336                 sel_bus=0;
337                 end
338             end
339         if (DECF)
340             begin
341             op=7;
342             sel_alu=1;
343             if (ir_out[7]==0)
344                 load_w=1;
345             else
346                 begin
347                 ram_en=1;
348                 sel_bus=0;
349                 end
350             end
351         if (INCF)
352             begin
353             op=6;
354             sel_alu=1;
355             if (ir_out[7]==0)
356                 load_w=1;
357             else
358                 begin
359                 ram_en=1;
360                 sel_bus=0;
361                 end
362             end
363         if (IORWF)
364             begin
365             op=3;
366             sel_alu=1;
367             if (ir_out[7]==0)
368                 load_w=1;
369             else
370                 begin
371                 ram_en=1;
372                 sel_bus=0;
373                 end
374             end
375         if (MOVF)
376             begin
377             op=5;
378             sel_alu=1;
379             if (ir_out[7]==0)
380                 load_w=1;
381             else
382                 begin
383                 ram_en=1;
384                 sel_bus=0;
385                 end
386             end
387         if (MOVWF)
388             begin
389             ram_en = 1;
390             sel_bus = 1;
391             if (addr_port_b == 1)
```

```
390             sel_bus = 1;
391             if (addr_port_b == 1)
392                 load_port_b = 1;
393             else
394                 ram_en = 1;
395             end
396         if (SUBWF)
397             begin
398             op=1;
399             sel_alu=1;
400             if (ir_out[7]==0)
401                 load_w=1;
402             else
403                 begin
404                 ram_en=1;
405                 sel_bus=0;
406                 end
407             end
408         if (XORWF)
409             begin
410             op=4;
411             sel_alu=1;
412             if (ir_out[7]==0)
413                 load_w=1;
414             else
415                 begin
416                 ram_en=1;
417                 sel_bus=0;
418                 end
419             end
420         if (DECFSZ)
421             begin
422             if (ir_out[7]==0)
423                 begin
424                 sel_alu=1;
425                 op=7;
426                 load_w=1;
427                 if (alu_zero==1)
428                     begin
429                     load_pc=1;
430                     sel_pc=0;
431                     end
432                 end
433             else
434                 begin
435                 sel_alu=1;
436                 op=7;
437                 ram_en=1;
438                 sel_bus=0;
439                 if (alu_zero==1)
440                     begin
441                     load_pc=1;
442                     sel_pc=0;
443                     end
444                 end
445             end
446         if (INCFSZ)
447             begin
448             if (ir_out[7]==0)
449                 begin
450                 sel_alu=1;
451                 op=6;
452                 load_w=1;
453                 if (alu_zero==1)
454                     begin
455                     load_pc=1;
456                     sel_pc=0;
457                     end
458                 end
459             else
460                 begin
461                 sel_alu=1;
462                 op=6;
463                 ram_en=1;
464                 sel_bus=0;
465                 if (alu_zero==1)
466                     begin
```

```
460                 begin
461                 sel_alu=1;
462                 op=6;
463                 ram_en=1;
464                 sel_bus=0;
465                 if (alu_zero==1)
466                     begin
467                     load_pc=1;
468                     sel_pc=0;
469                     end
470                 end
471             end
472         if (BCF)
473             begin
474             sel_alu=1;
475             sel_RAM_mux=1;
476             op=5;
477             sel_bus=0;
478             ram_en=1;
479             end
480         if (BSF)
481             begin
482             sel_alu=1;
483             sel_RAM_mux=2;
484             op=5;
485             sel_bus=0;
486             ram_en=1;
487             end
488         if (BTFSC|BTFSS)
489             begin
490             if (btfsc_btfss_skip_bit ==1)
491             begin
492             load_pc=1;
493             sel_pc=0;
494             end
495             end
496         if (ASRF)
497             begin
498             sel_alu=1;
499             sel_RAM_mux=0;
500             op=10;
501             if (ir_out[7]==0)
502                 load_w=1;
503             else
504                 begin
505                 ram_en=1;
506                 sel_bus=0;
507                 end
508             end
509         if (LSLF)
510             begin
511             sel_alu=1;
512             sel_RAM_mux=0;
513             op=11;
514             if (ir_out[7]==0)
515                 load_w=1;
516             else
517                 begin
518                 ram_en=1;
519                 sel_bus=0;
520                 end
521             end
522         if (LSRF)
523             begin
524             sel_alu=1;
525             sel_RAM_mux=0;
526             op=12;
527             if (ir_out[7]==0)
528                 load_w=1;
529             else
530                 begin
531                 ram_en=1;
532                 sel_bus=0;
533                 end
534             end
535         if (RLF)
536             begin
537             sel_alu=1;
538             sel_RAM_mux=0;
```

```
538                 sel_RAM_mux=0;
539                 op=13;
540                 if (ir_out[7]==0)
541                     load_w=1;
542                 else
543                     begin
544                     ram_en=1;
545                     sel_bus=0;
546                     end
547                 end
548             if (RRF)
549                 begin
550                 sel_alu=1;
551                 sel_RAM_mux=0;
552                 op=14;
553                 if (ir_out[7]==0)
554                     load_w=1;
555                 else
556                     begin
557                     ram_en=1;
558                     sel_bus=0;
559                     end
560                 end
561             if (SWAPF)
562                 begin
563                 sel_alu=1;
564                 sel_RAM_mux=0;
565                 op=15;
566                 if (ir_out[7]==0)
567                     load_w=1;
568                 else
569                     begin
570                     ram_en=1;
571                     sel_bus=0;
572                     end
573                 end
574             if (CALL)
575                 begin
576                 sel_pc=2'b01;
577                 load_pc=1;
578                 push=1;
579                 end
580             if (RETURN)
581                 begin
582                 sel_pc=2'b10;
583                 load_pc=1;
584                 pop=1;
585                 end
586         end
587       endcase
588     end
589     //controller
590   endmodule
```

stack.v:

```
module Stack(
    stack_out,
    stack_in,
    push,
    pop,
    reset,
    clk
);

output [10:0] stack_out;
input  [10:0] stack_in;
input  push,pop,reset,clk;

reg [3:0] stk_ptr;
reg [10:0] stack [15:0];
wire [10:0] stack_out;
wire [3:0] stk_index;

assign stk_index = stk_ptr + 1;
assign stack_out = stack[stk_ptr[3:0]];

always @(posedge clk)
    begin
        if(reset)
            stk_ptr <= 4'b1111;

        else if(push)
            begin
            stack[stk_index[3:0]] <= stack_in;
            stk_ptr <= stk_ptr + 1;
            end
        else if(pop)
            stk_ptr <= stk_ptr - 1;
    end
endmodule
```

● 結論與心得：

這次用 MPLAB 設計一個 Rom 跟上次比較起來更難一點,花的時間也比上次久
後來去補強解決問題了