

# 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:

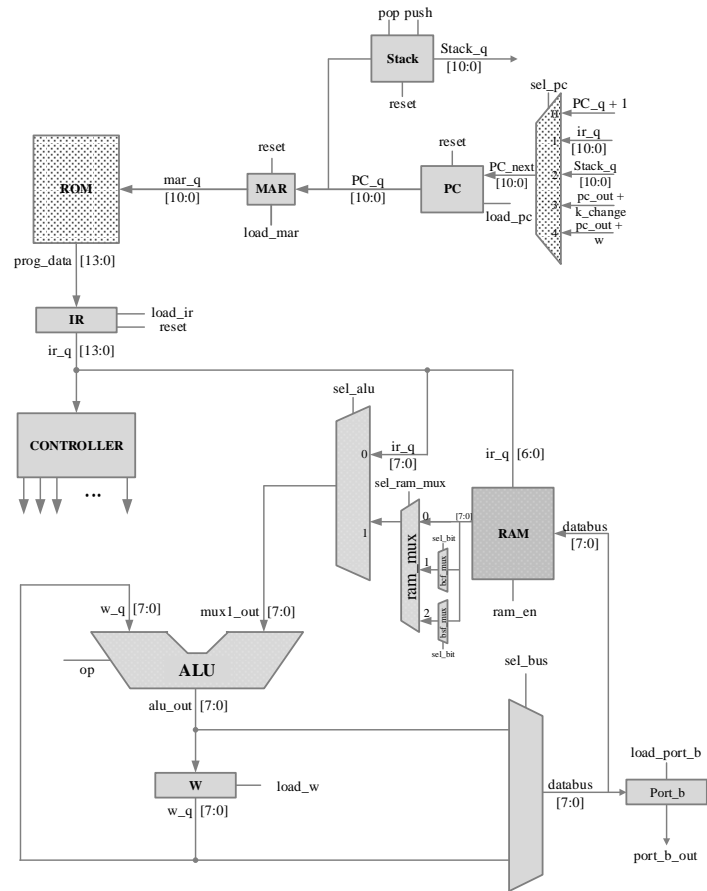
輾轉相除法(可以用迴圈暴力解)

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
        break;            //跳過代表等於0 都沒跳過則都是goto同一個地方
    }
}

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

```

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         4'h0: alu_q = mux1_out[7:0] + w_q;
10        4'h1: alu_q = mux1_out[7:0] - w_q;
11        4'h2: alu_q = mux1_out[7:0] & w_q;
12        4'h3: alu_q = mux1_out[7:0] | w_q;
13        4'h4: alu_q = mux1_out[7:0] ^ w_q; //XOR
14        4'h5: alu_q = mux1_out[7:0];
15        4'h6: alu_q = mux1_out[7:0] + 1;
16        4'h7: alu_q = mux1_out[7:0] - 1;
17        4'h8: alu_q = 0;
18        4'h9: alu_q = ~mux1_out[7:0];
19        4'hA: alu_q = { mux1_out[7],mux1_out[7:1] }; //ASRF arithmetic right shift
20        4'hB: alu_q = { mux1_out[6:0], 1'b0 }; //LSLF logical left shift
21        4'hC: alu_q = { 1'b0, mux1_out[7:1] }; //LSRF logical right shift
22        4'hD: alu_q = { mux1_out[6:0], mux1_out[7] }; //RLF rotate left f
23        4'hE: alu_q = { mux1_out[0], mux1_out[7:1] }; //RRF rotate right f
24        4'hF: alu_q = { mux1_out[3:0], mux1_out[7:4] }; //{m7, m6,...m4, m3,...m0} => {m3,...m0, m7, m6,...m4}
25        default: alu_q = mux1_out[7:0] + w_q;
26    endcase
27 end
28
29 endmodule

```

```

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 Stack (
2     output logic [10:0] stack_out,
3     input [10:0] stack_in,
4     input push,
5     input pop,
6     input reset,
7     input clk
8 );
9     logic [3:0] stk_ptr;
10    logic [10:0] stack [15:0];
11    logic [3:0] stk_index;
12
13    assign stk_index = stk_ptr + 1;
14    assign stack_out = stack[stk_ptr];
15
16    always_ff @(posedge clk) begin
17        if(reset) begin
18            stk_ptr <= 4'b1111;
19        end
20        else if(push) begin
21            stack[stk_index] <= stack_in;
22            stk_ptr <= stk_ptr + 1;
23        end
24        else if(pop) begin
25            stk_ptr <= stk_ptr - 1 ;
26        end
27    end
28
29 endmodule

```

```

1 module cpu (
2     input clk,
3     input reset,
4     output logic [7:0] port_b_out
5 );
6     logic [13:0] rom_q, ir_q;
7     logic [10:0] pc_next, pc_q, mar_q, k_change;
8     logic load_pc, load_mar, load_ir, reset_ir, load_w, ram_en, sel_alu, d, sel_bus;
9     logic load_port_b;
10    logic [2:0] sel_pc;
11    logic [3:0] ps,ns;
12    logic [7:0] w_q, alu_q, ram_out, mux1_out, bcf_mux, bsf_mux, ram_mux;
13    logic [7:0] databus;
14    logic [3:0] op;
15    logic [5:0] opcode;
16    logic [2:0] sel_bit;
17    logic [1:0] sel_ram_mux;
18    //for Stack
19    logic pop, push;
20    logic [10:0] stack_out;
21
22    //Stack
23    Stack Stack_1(
24        .stack_out(stack_out),
25        .stack_in(pc_q),
26        .push(push),
27        .pop(pop),
28        .reset(reset),
29        .clk(clk)
30    );
31
32    assign w_change = {3'b0, w_q} - 1;
33    assign k_change = {ir_q[8], ir_q[8], ir_q[8:0]}-1 ;
34
35    //mux 0 (select next PC address)
36    always_comb begin
37        if(sel_pc == 4) begin
38            pc_next = pc_q + w_change;
39        end
40        else if(sel_pc == 3) begin
41            pc_next = pc_q + k_change;
42        end
43        else if(sel_pc == 2) begin
44            pc_next = stack_out;
45        end
46        else if(sel_pc == 1) begin
47            pc_next = ir_q;
48        end
49        else begin
50            pc_next = pc_q + 1;

```

```

51     end
52 end
53
54 //pc
55 always_ff @( posedge clk ) begin
56     if(reset)
57         pc_q <= 0;
58     else if(load_pc)
59         pc_q <= pc_next;
60 end
61
62 //mar
63 always_ff @( posedge clk ) begin
64     if(load_mar)
65         mar_q <= pc_q;
66 end
67
68 //ROM
69 Program_Rom ROM_1(
70     .Rom_addr_in(mar_q),
71     .Rom_data_out(rom_q)
72 );
73
74 //IR
75 always_ff @( posedge clk ) begin
76     if(reset_ir)
77         ir_q <= 0;
78     else if(load_ir)
79         ir_q <= rom_q;
80 end
81
82 //RAM
83 single_port_ram_128x8 single_port_ram_128x8_1(
84     .data(databus),
85     .addr(ir_q[6:0]),
86     .ram_en(ram_en),
87     .clk(clk),
88     .ram_out(ram_out)
89 );
90
91 assign sel_bit = ir_q[9:7];
92
93 //BCF mux
94 always_comb begin
95     case (sel_bit)
96         3'b000: bcf_mux = ram_out & 8'b1111_1110;
97         3'b001: bcf_mux = ram_out & 8'b1111_1101;
98         3'b010: bcf_mux = ram_out & 8'b1111_1011;
99         3'b011: bcf_mux = ram_out & 8'b1111_0111;
100        3'b100: bcf_mux = ram_out & 8'b1110_1111;

```



```

101         3'b101: bcf_mux = ram_out & 8'b1101_1111;
102         3'b110: bcf_mux = ram_out & 8'b1011_1111;
103         3'b111: bcf_mux = ram_out & 8'b0111_1111;
104     endcase
105 end
106
107 //BSF mux
108 always_comb begin
109     case (sel_bit)
110         3'b000: bsf_mux = ram_out | 8'b0000_0001;
111         3'b001: bsf_mux = ram_out | 8'b0000_0010;
112         3'b010: bsf_mux = ram_out | 8'b0000_0100;
113         3'b011: bsf_mux = ram_out | 8'b0000_1000;
114         3'b100: bsf_mux = ram_out | 8'b0001_0000;
115         3'b101: bsf_mux = ram_out | 8'b0010_0000;
116         3'b110: bsf_mux = ram_out | 8'b0100_0000;
117         3'b111: bsf_mux = ram_out | 8'b1000_0000;
118     endcase
119 end
120
121 //ram mux (select data into mux1)
122 always_comb begin
123     case (sel_ram_mux)
124         0: ram_mux = ram_out;
125         1: ram_mux = bcf_mux;
126         2: ram_mux = bsf_mux;
127     endcase
128 end
129
130 //mux 1 (select data into ALU)
131 always_comb begin
132     if(sel_alu) begin
133         mux1_out = ram_mux;
134     end
135     else begin
136         mux1_out = ir_q;
137     end
138 end
139
140 assign d = ir_q[7];
141 assign MOVLW = (ir_q[13:8]==6'h30);
142 assign ADDLW = (ir_q[13:8]==6'h3E);
143 assign IORLW = (ir_q[13:8]==6'h38);
144 assign ANDLW = (ir_q[13:8]==6'h39);
145 assign SUBLW = (ir_q[13:8]==6'h3C);
146 assign XORLW = (ir_q[13:8]==6'h3A);
147
148 assign ADDWF = (ir_q[13:8]==6'h07);
149 assign ANDWF = (ir_q[13:8]==6'h05);
150 assign CLRF = (ir_q[13:8]==6'h01 && d==1);

```



```

151 assign CLRW = (ir_q[13:4]==10'h010 && ir_q[3:2]==2'h0);
152 assign COMF = (ir_q[13:8]==6'h09);
153 assign DECF = (ir_q[13:8]==6'h03);
154 assign GOTO = (ir_q[13:11]==3'b101);
155
156 assign INCF = (ir_q[13:8]==6'h0A);
157 assign IORWF = (ir_q[13:8]==6'h04);
158 assign MOVF = (ir_q[13:8]==6'h08);
159 assign MOVWF = (ir_q[13:8]==6'h00 && ir_q[7]==1'b1);
160 assign SUBWF = (ir_q[13:8]==6'h02);
161 assign XORWF = (ir_q[13:8]==6'h06);
162
163 assign BCF = (ir_q[13:12]==2'b01 && ir_q[11:10]==2'b00);
164 assign BSF = (ir_q[13:12]==2'b01 && ir_q[11:10]==2'b01);
165 assign BTFSC = (ir_q[13:12]==2'b01 && ir_q[11:10]==2'b10);
166 assign BTFSS = (ir_q[13:12]==2'b01 && ir_q[11:10]==2'b11);
167 assign DECFSZ = (ir_q[13:8]==6'h0B);
168 assign INCFSZ = (ir_q[13:8]==6'h0F);
169
170 assign btfsc_skip_bit = (ram_out[ir_q[9:7]]==0);
171 assign btfss_skip_bit = (ram_out[ir_q[9:7]]==1);
172 assign btfsc_btfss_skip_bit = (BTFSC&btfsc_skip_bit) |
173                               (BTFSS&btfss_skip_bit);
174 assign ASRF = (ir_q[13:8]==6'h37); // arithmetic right shift
175 assign LSLF = (ir_q[13:8]==6'h35); // logical left shift
176 assign LSRF = (ir_q[13:8]==6'h36); // logical right shift
177 assign RLF = (ir_q[13:8]==6'h0D); // rotate left f
178 assign RRF = (ir_q[13:8]==6'h0C); // rotate right f
179 assign SWAP = (ir_q[13:8]==6'h0E); //{m7, m6,...m4, m3,...m0} => {m3,...m0, m7, m6,...m4}
180
181 assign CALL = (ir_q[13:12]==2'b10 && ir_q[11]==0);
182 assign RETURN = (ir_q == 8);
183
184 assign BRA = (ir_q[13:12]==2'b11 && ir_q[11:9]==3'b001); // relative branch
185 assign BRW = (ir_q[13:0]==14'h000B); // relative branch with W
186 assign NOP = (ir_q[13:0]==14'h0000); // no operation
187
188 //ALU
189 ALU ALU_1(
190     .op(op),
191     .w_q(w_q),
192     .mux1_out(mux1_out),
193     .alu_q(alu_q)
194 );
195 assign aluout_zero = (alu_q == 0);
196
197 //register
198 always_ff @( posedge clk ) begin
199     if(load_w)
200         w_q <= alu_q;

```

```

201     end
202
203     //mux 2 (select data into RAM)
204     always_comb begin
205         if(sel_bus) begin
206             databus = w_q;
207         end
208         else begin
209             databus = alu_q;
210         end
211     end
212
213     //Port_b
214     always_ff @( posedge clk ) begin
215         if(reset) begin
216             port_b_out <= 0;
217         end
218         else if(load_port_b) begin
219             port_b_out <= databus;
220         end
221     end
222
223     logic [3:0] ten,unit;
224     always_comb begin
225         ten = port_b_out / 10;
226         unit = port_b_out - ten * 10;
227     end
228
229     assign addr_port_b = (ir_q[6:0] == 7'h0D);
230
231     //controller
232     parameter T0 = 0;
233     parameter T1 = 1;
234     parameter T2 = 2;
235     parameter T3 = 3;
236     parameter T4 = 4;
237     parameter T5 = 5;
238     parameter T6 = 6;
239
240     always_ff @( posedge clk ) begin
241         if(reset) ps <= 0;
242         else ps <= ns;
243     end
244
245     always_comb begin
246         sel_alu = 0;
247         sel_pc = 0;
248         load_mar = 0;
249         load_pc = 0;
250         reset_ir = 0;

```

```

251     load_ir = 0;
252     load_w = 0;
253     ram_en = 0;
254     op = 0;
255     sel_ram_mux = 0;
256     sel_bus = 0;
257     load_port_b = 0;
258     ns=0;
259     //for Stack
260     push = 0;
261     pop = 0;
262     case(ps)
263         T0: begin
264             ns = T1;
265         end
266         T1: begin
267             load_mar = 1;
268             sel_pc = 0;
269             load_pc = 1;
270             ns = T2;
271         end
272         T2: begin
273             ns = T3;
274         end
275         T3: begin
276             load_ir = 1;
277             ns = T4;
278         end
279         T4: begin
280             load_mar = 1;
281             sel_pc = 2'b00;
282             load_pc = 1;
283             if(GOTO) begin//to skip the following instruction in T4
284                 load_mar = 1;
285             end
286             else if(MOVLW) begin
287                 sel_alu = 0;
288                 op = 5;
289                 load_w = 1;
290             end
291             else if(ADDLW) begin
292                 sel_alu = 0;
293                 op = 0;
294                 load_w = 1;
295             end
296             else if(IORLW) begin
297                 sel_alu = 0;
298                 op = 3;
299                 load_w = 1;
300             end

```

```
301         else if(ANDLW) begin
302             sel_alu = 0;
303             op = 2;
304             load_w = 1;
305         end
306         else if(SUBLW) begin
307             sel_alu = 0;
308             op = 1;
309             load_w = 1;
310         end
311         else if(XORLW) begin
312             sel_alu = 0;
313             op = 4;
314             load_w = 1;
315         end
316
317         else if(ADDWF) begin
318             op = 0;
319             sel_alu = 1;
320             if(d) begin
321                 ram_en = 1;
322             end
323             else begin
324                 load_w = 1;
325             end
326         end
327         else if(ANDWF) begin
328             op = 2;
329             sel_alu = 1;
330             if(d) begin
331                 ram_en = 1;
332             end
333             else begin
```

```

334         load_w = 1;
335     end
336 end
337 ✓ else if(CLRF) begin
338     op = 8;
339     ram_en = 1;
340 end
341 ✓ else if(CLRW) begin
342     op = 8;
343     load_w = 1;
344 end
345 ✓ else if(COMF) begin
346     op = 9;
347     sel_alu = 1;
348     ram_en = 1;
349 end
350 ✓ else if(DECF) begin
351     op = 7;
352     sel_alu = 1;
353     ram_en = 1;
354 end
355
356 ✓ else if(INCF) begin
357     op = 6;
358     sel_alu = 1;
359 ✓ if(d) begin
360     ram_en = 1;
361     sel_bus = 0;
362 end
363 ✓ else begin
364     load_w = 1;
365 end
366 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;
```

```
401         if(d) begin
402             ram_en = 1;
403             sel_bus = 0;
404         end
405         else begin
406             load_w = 1;
407         end
408     end
409     else if(XORWF) begin
410         op = 4;
411         sel_alu = 1;
412         if(d) begin
413             ram_en = 1;
414             sel_bus = 0;
415         end
416         else begin
417             load_w = 1;
418         end
419     end
420     else if(BCF)begin
421         sel_alu = 1;
422         sel_ram_mux = 1;
423         op = 5;
424         sel_bus = 0;
425         ram_en = 1;
426     end
427     else if(BSF)begin
428         sel_alu = 1;
429         sel_ram_mux = 2;
430         op = 5;
431         sel_bus = 0;
432         ram_en = 1;
433     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
```

```
467         else begin
468             load_w = 1;
469         end
470     end
471     else if(RLF) begin
472         sel_alu = 1;
473         sel_ram_mux = 0;
474         op = 4'hD;
475         if(d)begin
476             sel_bus = 0;
477             ram_en = 1;
478         end
479         else begin
480             load_w = 1;
481         end
482     end
483     else if(RRF) begin
484         sel_alu = 1;
485         sel_ram_mux = 0;
486         op = 4'hE;
487         if(d)begin
488             sel_bus = 0;
489             ram_en = 1;
490         end
491         else begin
492             load_w = 1;
493         end
494     end
495     else if(SWAP) begin
496         sel_alu = 1;
497         sel_ram_mux = 0;
498         op = 4'hF;
499         if(d)begin
500             sel_bus = 0;
```

```
501         ram_en = 1;
502     end
503     else begin
504         load_w = 1;
505     end
506 end
507
508     else if(CALL) begin
509         push = 1;
510     end
511     ns = T5;
512 end
513 T5: begin
514     if(GOTO) begin
515         sel_pc = 2'b01;
516         load_pc = 1;
517     end
518     else if(CALL)begin
519         sel_pc = 2'b01;
520         load_pc = 1;
521     end
522     else if(RETURN) begin
523         pop = 1;
524         sel_pc = 2'b10;
525         load_pc = 1;
526     end
527     else if(BRA) begin
528         load_pc = 1;
529         sel_pc = 3;
530     end
531     else if(BRW) begin
532         load_pc = 1;
533         sel_pc = 4;
```

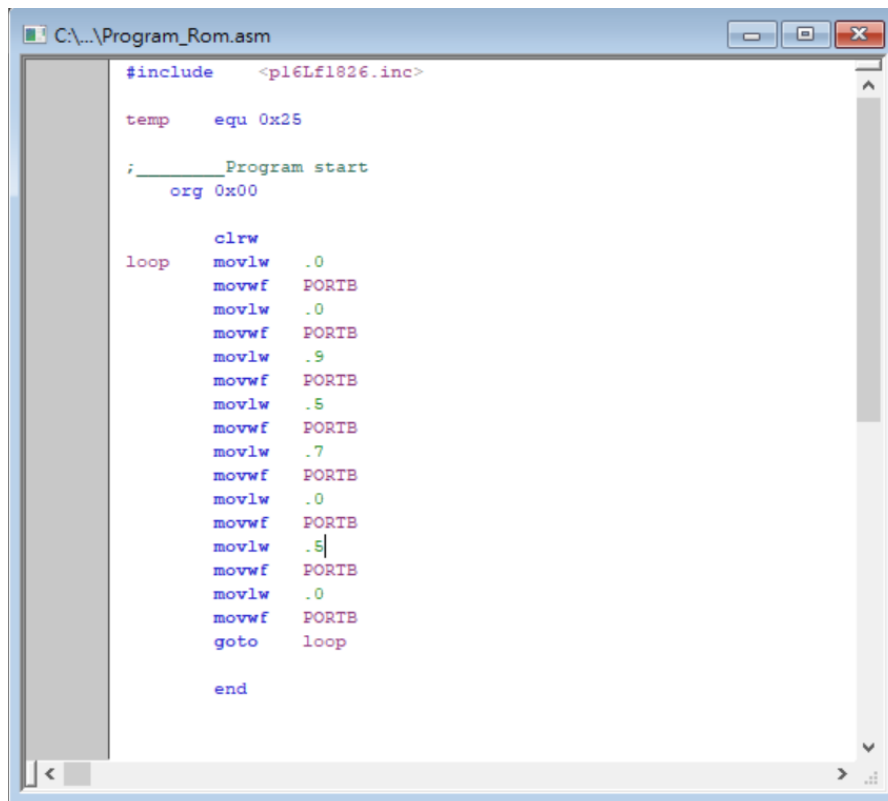
```
534         end
535         ns = T6;
536     end
537     T6: begin
538         load_ir = 1;
539         if(GOTO || CALL || RETURN || BRA || BRW) begin
540             reset_ir = 1;
541         end
542         else if(DECFSZ)begin
543             op = 7;
544             sel_alu = 1;
545             if(aluout_zero)begin
546                 reset_ir = 1;
547             end
548             if(d)begin
549                 sel_bus = 0;
550                 ram_en = 1;
551             end
552             else begin
553                 load_w = 1;
554             end
555         end
556         else if(INCFSZ) begin
557             op = 6;
558             sel_alu = 1;
559             if(aluout_zero)begin
560                 reset_ir = 1;
561             end
562             if(d)begin
563                 sel_bus = 0;
564                 ram_en = 1;
565             end
566             else begin
```

```

567         load_w = 1;
568     end
569 end
570 else if(BTFSC || BTFSS)begin
571     if( btfsc_btfss_skip_bit )begin
572         reset_ir = 1;
573     end
574 end
575 ns = T4;
576 end
577 endcase
578 end
579 endmodule

```

## 第 1 題



```

C:\...Program_Rom.asm
#include    <pic16lf1826.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

```

```

1  module Program_Rom(
2      output logic [13:0] Rom_data_out,
3      input [10:0] Rom_addr_in
4  );
5
6      logic [13:0] data;
7      always_comb
8      begin
9          case (Rom_addr_in)
10             10'h0 : data = 14'h0103;
11             10'h1 : data = 14'h3000;
12             10'h2 : data = 14'h008D;
13             10'h3 : data = 14'h3000;
14             10'h4 : data = 14'h008D;
15             10'h5 : data = 14'h3009;
16             10'h6 : data = 14'h008D;
17             10'h7 : data = 14'h3005;
18             10'h8 : data = 14'h008D;
19             10'h9 : data = 14'h3007;
20             10'ha : data = 14'h008D;
21             10'hb : data = 14'h3000;
22             10'hc : data = 14'h008D;
23             10'hd : data = 14'h3005;
24             10'he : data = 14'h008D;
25             10'hf : data = 14'h3000;
26             10'h10 : data = 14'h008D;
27             10'h11 : data = 14'h2801;
28             10'h12 : data = 14'h3400;
29             10'h13 : data = 14'h3400;
30             default: data = 14'h0;
31         endcase
32     end
33
34     assign Rom_data_out = data;
35
36 endmodule
37

```

```

1  onerror {resume}
2  quietly WaveActivateNextPane {} 0
3
4  add wave -noupdate -divider {student number to port_b_out}
5
6  add wave -noupdate -format Literal -radix Unsigned      /testbench/clock
7  add wave -noupdate -format Logic -radix decimal        /testbench/reset
8  add wave -noupdate -format Literal -radix Unsigned     /testbench/cpu_test/port_b_out
9  add wave -noupdate -format Literal -radix Hexadecimal  /testbench/cpu_test/ir_q

```

```
C:\...\Program_Rom.asm

#include    <pl6Lf1826.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
```





```
1 module Program_Rom(  
2     output logic [13:0] Rom_data_out,  
3     input [10:0] Rom_addr_in  
4 );  
5  
6     logic [13:0] data;  
7     always_comb  
8         begin  
9             case (Rom_addr_in)  
10                10'h0 : data = 14'h3005;  
11                10'h1 : data = 14'h00A5;  
12                10'h2 : data = 14'h3003;  
13                10'h3 : data = 14'h00A4;  
14                10'h4 : data = 14'h0824;  
15                10'h5 : data = 14'h00A2;  
16                10'h6 : data = 14'h0103;  
17                10'h7 : data = 14'h00A3;  
18                10'h8 : data = 14'h0825;  
19                10'h9 : data = 14'h07A3;  
20                10'ha : data = 14'h0BA2;  
21                10'hb : data = 14'h2808;  
22                10'hc : data = 14'h0823;  
23                10'hd : data = 14'h008D;  
24                10'he : data = 14'h3400;  
25                10'hf : data = 14'h3400;  
26                default: data = 14'h0;  
27            endcase  
28        end  
29  
30        assign Rom_data_out = data;  
31  
32 endmodule  
33
```



```
1 onerror {resume}  
2 quietly WaveActivateNextPane {} 0  
3  
4 add wave -noupdate -divider {multiple two numbers}  
5  
6 add wave -noupdate -format Literal -radix Unsigned      /testbench/clock  
7 add wave -noupdate -format Logic -radix decimal         /testbench/reset  
8 add wave -noupdate -format Literal -radix Unsigned      /testbench/cpu_test/port_b_out
```

### 第 3 題

```
C:\Program_Rom.asm

#include    <pl6Lf1826.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
```

```

1  module Program_Rom(
2      output logic [13:0] Rom_data_out,
3      input [10:0] Rom_addr_in
4  );
5
6      logic [13:0] data;
7      always_comb
8      begin
9          case (Rom_addr_in)
10             10'h0 : data = 14'h0103;
11             10'h1 : data = 14'h00A2;
12             10'h2 : data = 14'h3015;
13             10'h3 : data = 14'h00A5;
14             10'h4 : data = 14'h3003;
15             10'h5 : data = 14'h00A4;
16             10'h6 : data = 14'h0103;
17             10'h7 : data = 14'h00A1;
18             10'h8 : data = 14'h0824;
19             10'h9 : data = 14'h02A5;
20             10'ha : data = 14'h0AA2;
21             10'hb : data = 14'h1FA5;
22             10'hc : data = 14'h2808;
23             10'hd : data = 14'h03A2;
24             10'he : data = 14'h0822;
25             10'hf : data = 14'h00A3;
26             10'h10 : data = 14'h0822;
27             10'h11 : data = 14'h008D;
28             10'h12 : data = 14'h0825;
29             10'h13 : data = 14'h3C00;
30             10'h14 : data = 14'h00A1;
31             10'h15 : data = 14'h0821;
32             10'h16 : data = 14'h0224;
33             10'h17 : data = 14'h008D;
34             10'h18 : data = 14'h3400;
35             10'h19 : data = 14'h3400;
36             default: data = 14'h0;
37          endcase
38      end
39
40      assign Rom_data_out = data;
41
42  endmodule
43

```

```

1  onerror {resume}
2  quietly WaveActivateNextPane {} 0
3
4  add wave -noupdate -divider {divide two numbers}
5
6  add wave -noupdate -format Literal -radix Unsigned      /testbench/clk
7  add wave -noupdate -format Logic -radix decimal         /testbench/reset
8  add wave -noupdate -format Literal -radix Unsigned      /testbench/cpu_test/single_port_ram_128x8_1/ram\[35\]
9  add wave -noupdate -format Literal -radix Unsigned      /testbench/cpu_test/port_b_out

```

## 第 4 題

```

C:\...\Program_Rom.asm
#include <pl6Lf1826.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

```

```

1 module Program_Rom(
2     output logic [13:0] Rom_data_out,
3     input [10:0] Rom_addr_in
4 );
5
6     logic [13:0] data;
7     always_comb
8     begin
9         case (Rom_addr_in)
10             10'h0 : data = 14'h3024;
11             10'h1 : data = 14'h00A5;
12             10'h2 : data = 14'h3015;
13             10'h3 : data = 14'h00A4;
14             10'h4 : data = 14'h0103;
15             10'h5 : data = 14'h00A2;
16             10'h6 : data = 14'h0825;
17             10'h7 : data = 14'h00A6;
18             10'h8 : data = 14'h0824;
19             10'h9 : data = 14'h02A6;
20             10'ha : data = 14'h0AA2;
21             10'hb : data = 14'h1FA6;
22             10'hc : data = 14'h2808;
23             10'hd : data = 14'h0826;
24             10'he : data = 14'h00A1;
25             10'hf : data = 14'h0824;
26             10'h10 : data = 14'h07A1;
27             10'h11 : data = 14'h0824;
28             10'h12 : data = 14'h00A5;
29             10'h13 : data = 14'h0821;
30             10'h14 : data = 14'h00A4;
31             10'h15 : data = 14'h1FA4;
32             10'h16 : data = 14'h2818;
33             10'h17 : data = 14'h281D;
34             10'h18 : data = 14'h0824;
35             10'h19 : data = 14'h0099;
36             10'h1a : data = 14'h0399;
37             10'h1b : data = 14'h1F99;
38             10'h1c : data = 14'h2804;
39             10'h1d : data = 14'h0825;
40             10'h1e : data = 14'h00A3;
41             10'h1f : data = 14'h3400;
42             10'h20 : data = 14'h3400;
43             default: data = 14'h0;
44         endcase
45     end
46
47     assign Rom_data_out = data;
48
49 endmodule
50

```

```

1 onerror {resume}
2 quietly WaveActivateNextPane {} 0
3
4 add wave -noupdate -divider {Euclidean}
5
6 add wave -noupdate -format Literal -radix Unsigned      /testbench/clock
7 add wave -noupdate -format Logic -radix decimal        /testbench/reset
8 add wave -noupdate -format Literal -radix Unsigned     /testbench/cpu_test/single_port_ram_128x8_1/ram\[37\]
9 add wave -noupdate -format Literal -radix Unsigned     /testbench/cpu_test/single_port_ram_128x8_1/ram\[36\]
10 add wave -noupdate -format Literal -radix Unsigned     /testbench/cpu_test/single_port_ram_128x8_1/ram\[35\]

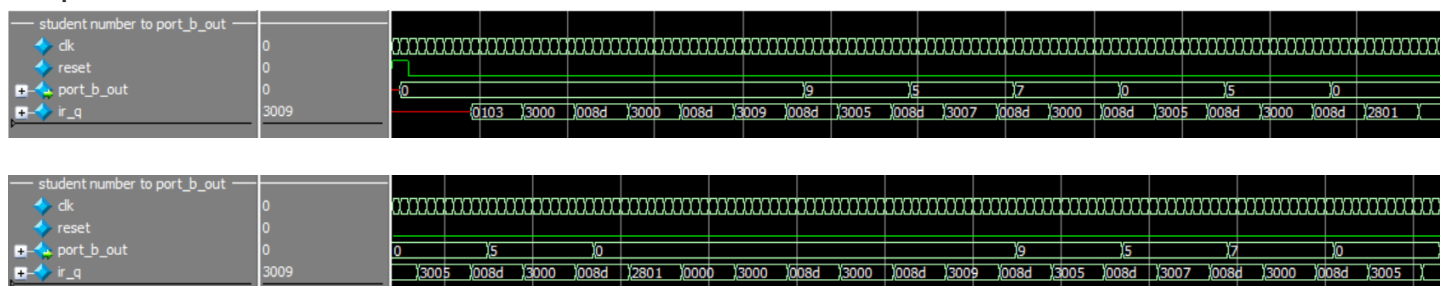
```

## ● 模擬結果與結果說明：

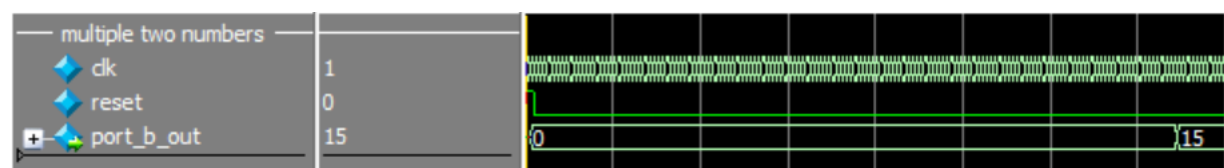
### 第 1 題

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

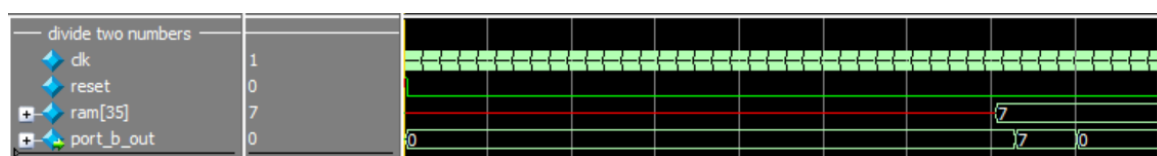


### 第 2 題



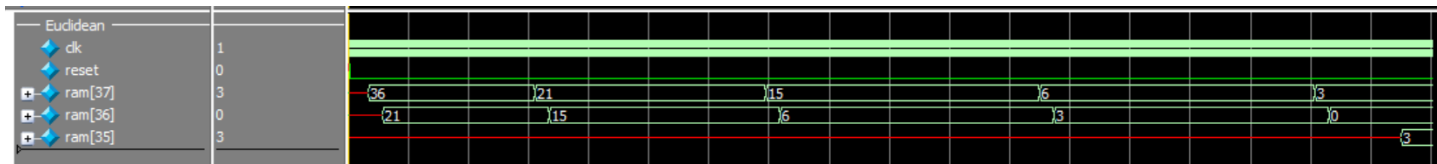
如同題目要求，計算  $5 \times 3$ ，port\_b\_out 成功輸出結果

### 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  $\text{answer} = a/c$ ；但是下方程式碼裡面卻沒有，所以我把  $\text{answer} = a/c$  的實作加在 count—後面，同時也在波型圖上呈現  $\text{answer}(0x23 \Rightarrow \text{ram}[35])$  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。