微算機實驗報告

期末報告

姓名：洪巧芸

系級：資工 **114**

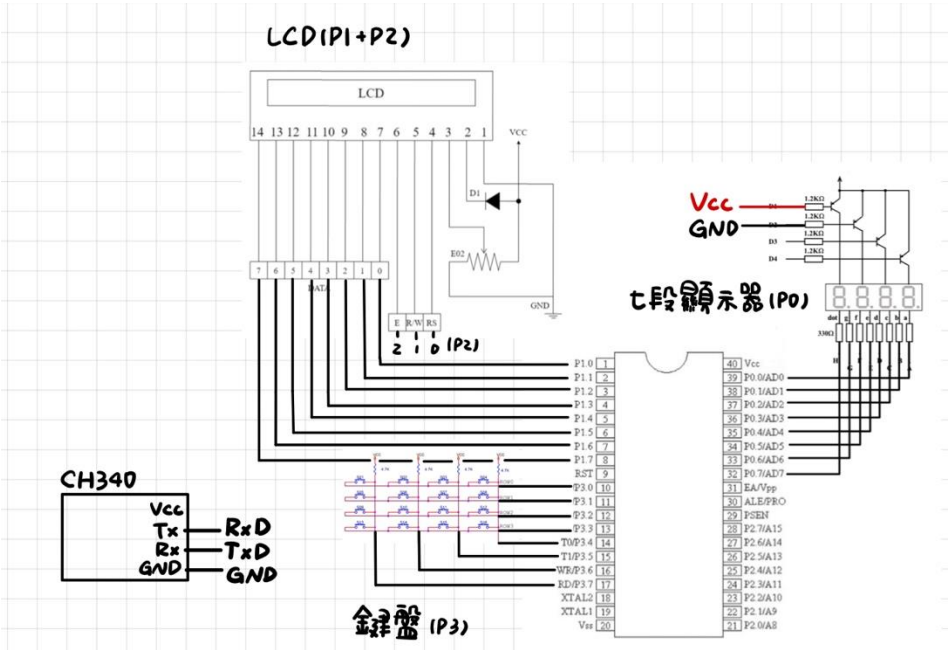
學號：**110550143**

一、實現功能：

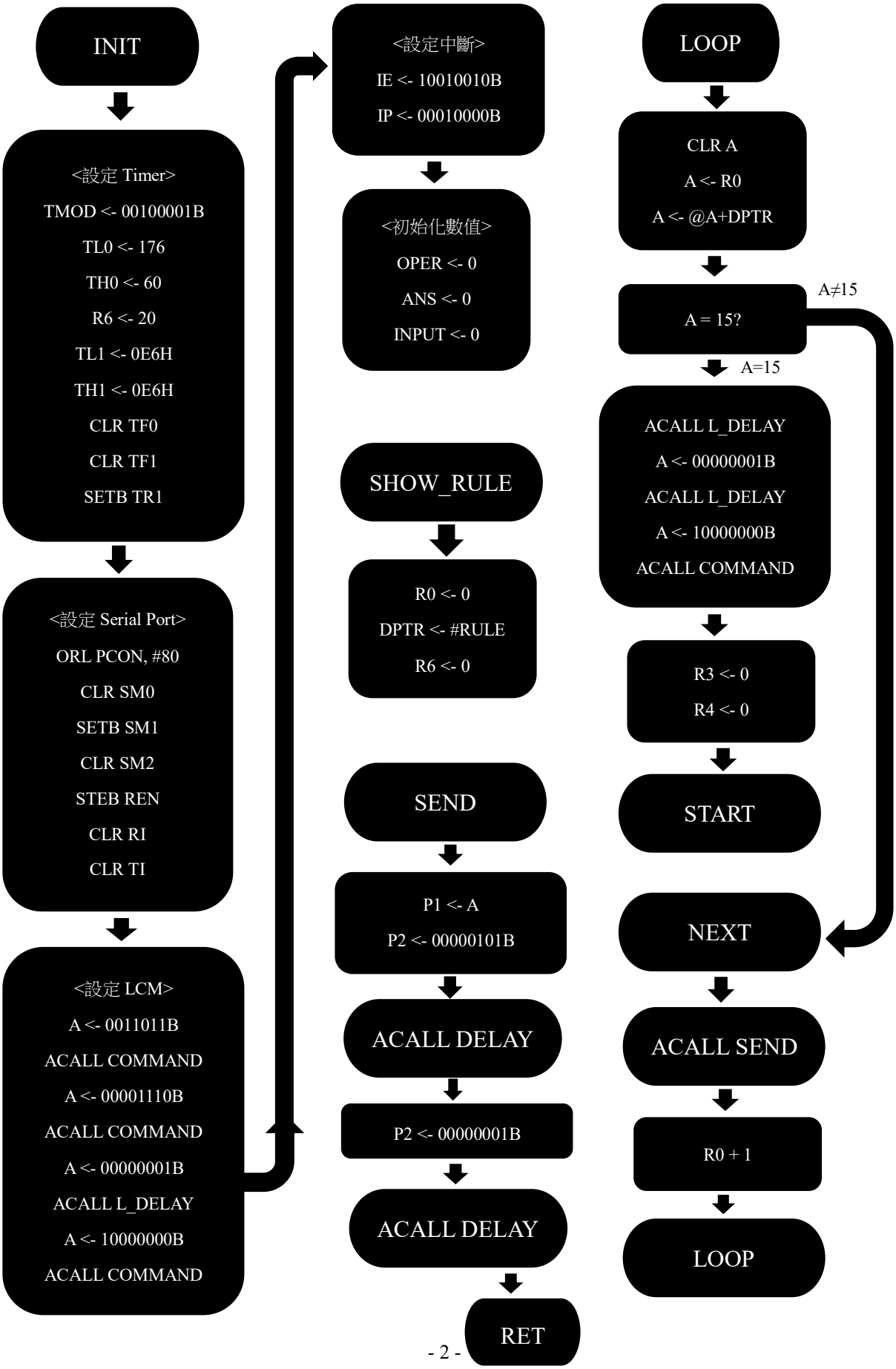
小型的加減乘除題目產生器，一開始 LCD 會顯示「Please Press 2 numbers」，接下來再由我們主動按鍵盤(00~15)決定要被計算的兩個數字，之後 LCD 會顯示我們所按的題目「數字 + / - / * / / 數字 =」，七段顯示器顯示解題所花的秒數，然後再到電腦視窗輸入「答案#」，輸出「o / x」。

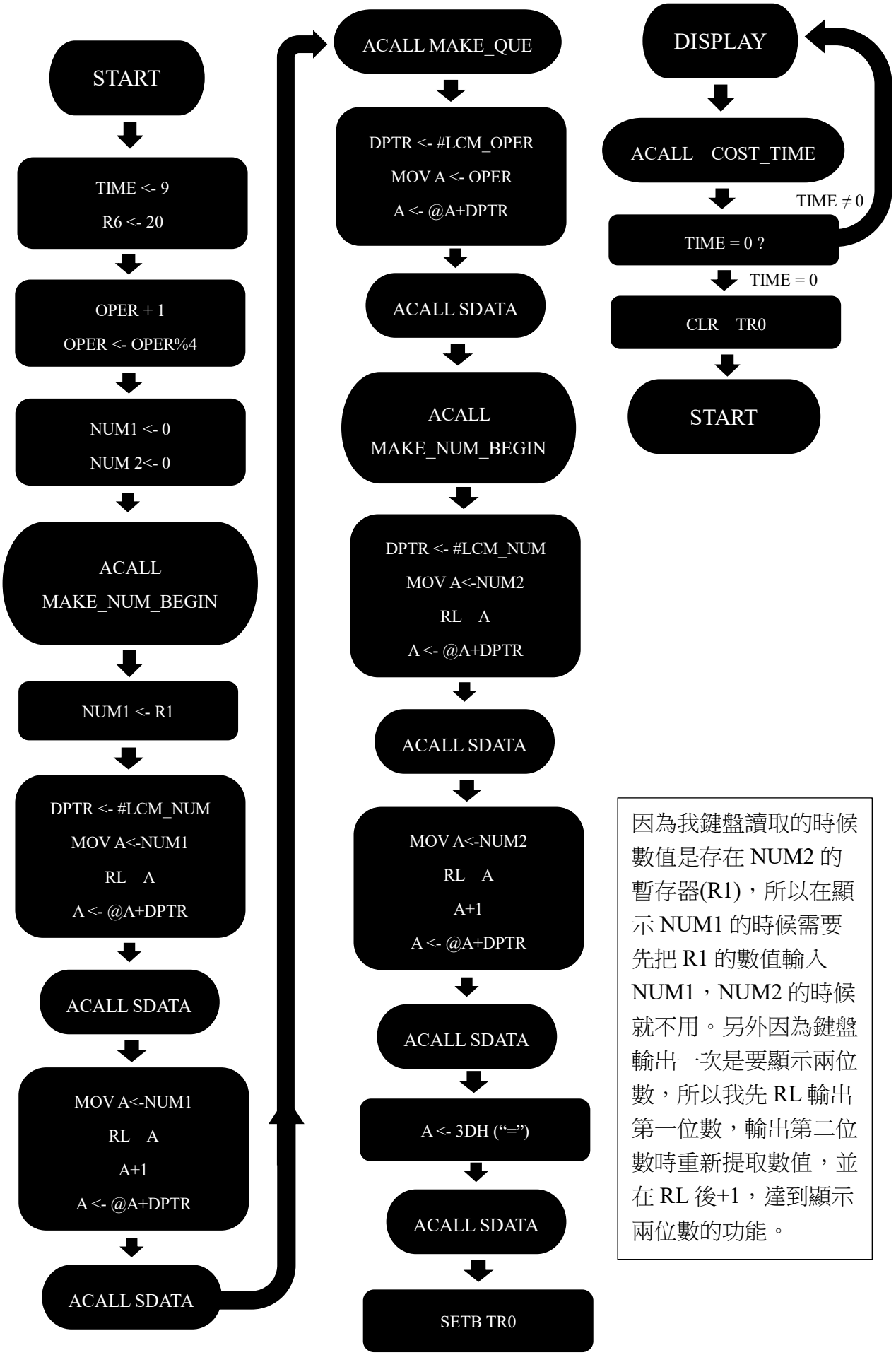
二、硬體架構：

1. LCD：D0~D7 接 P1，E,R/W,RS 接 P2
2. 鍵盤：JP03 接 P3
3. CH340：Tx 接 Rx，Rx 接 Tx，地接地
4. 七段顯示器：A~H 接 P0，D1 接 Vcc，D2 接地

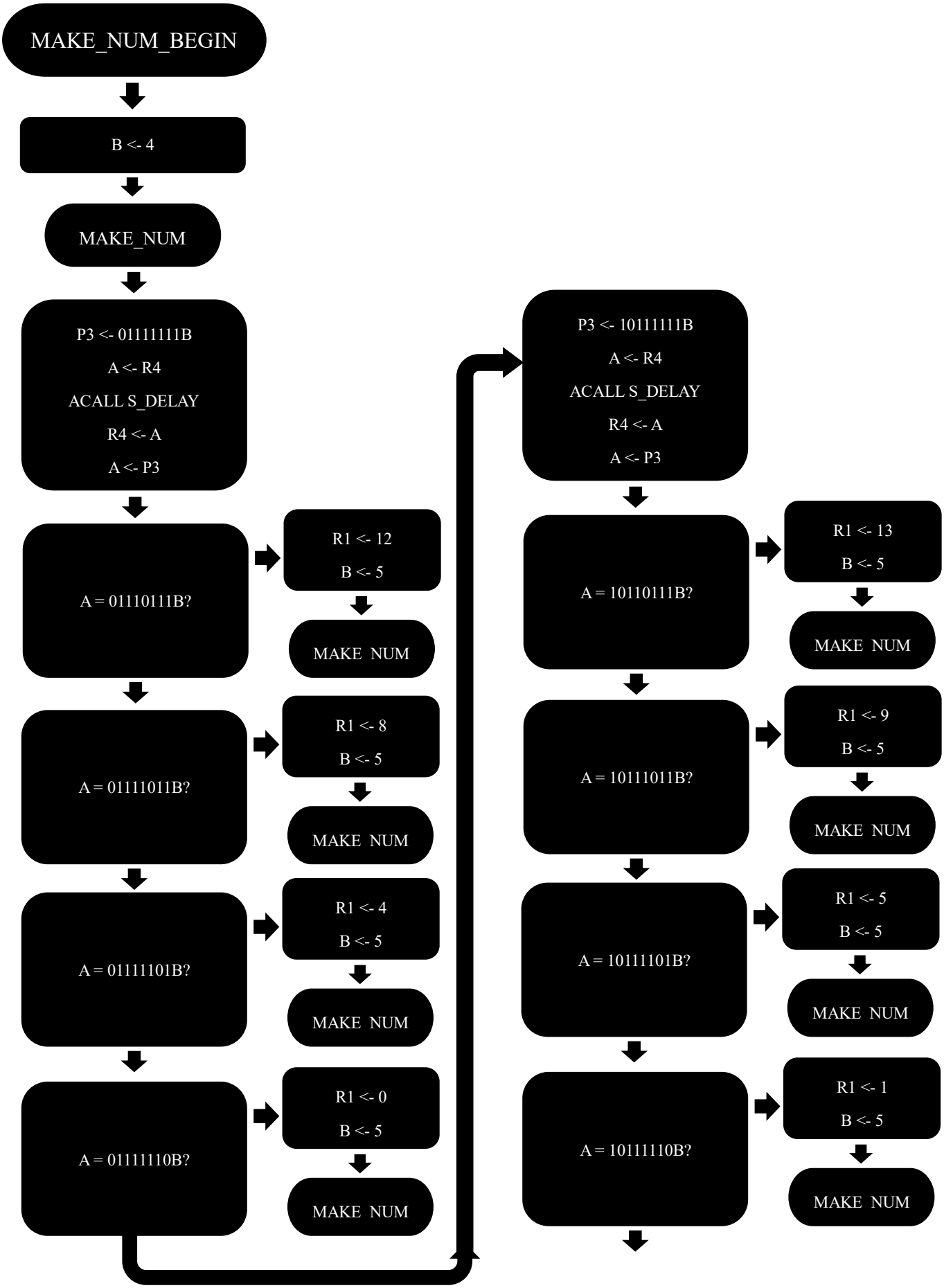


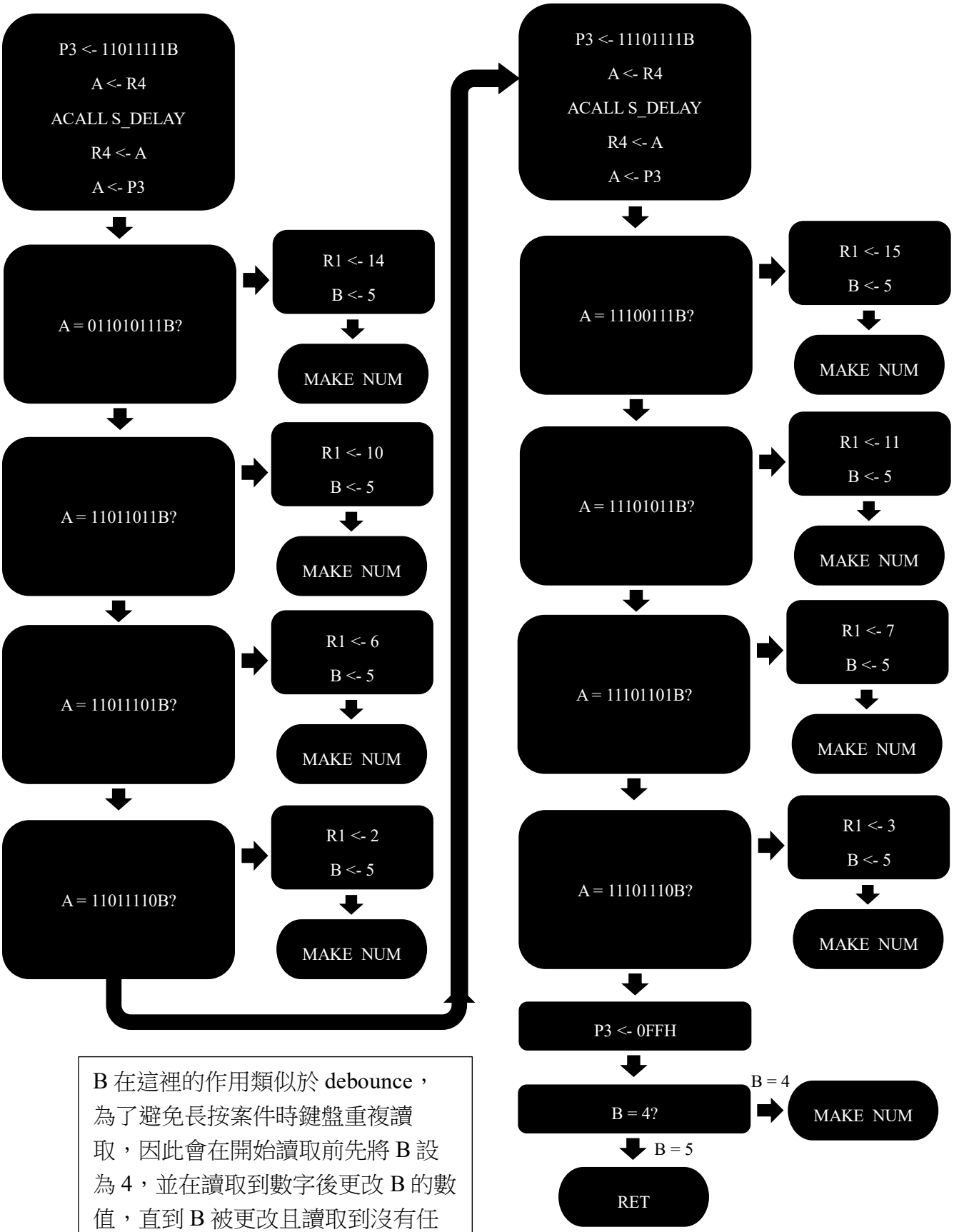
三、程式流程圖：


INIT
<設定 Timer>
TMOD <- 00100001B
TL0 <- 176
TH0 <- 60
R6 <- 20
TL1 <- 0E6H
TH1 <- 0E6H
CLR TF0
CLR TF1
SETB TR1
<設定 Serial Port>
ORL PCON, #80
CLR SM0
SETB SM1
CLR SM2
STEB REN
CLR RI
CLR TI
<設定 LCM>
A <- 0011011B
ACALL COMMAND
A <- 00001110B
ACALL COMMAND
A <- 00000001B
ACALL L_DELAY
A <- 10000000B
ACALL COMMAND
<設定中斷>
IE <- 10010010B
IP <- 00010000B
<初始化數值>
OPER <- 0
ANS <- 0
INPUT <- 0
SHOW_RULE
R0 <- 0
DPTR <- #RULE
R6 <- 0
SEND
P1 <- A
P2 <- 00000101B
ACALL DELAY
P2 <- 00000001B
ACALL DELAY
RET
LOOP
CLR A
A <- R0
A <- @A+DPTR
A = 15?
A≠15
A=15
ACALL L_DELAY
A <- 00000001B
ACALL L_DELAY
A <- 10000000B
ACALL COMMAND
R3 <- 0
R4 <- 0
START
NEXT
ACALL SEND
R0 + 1
LOOP

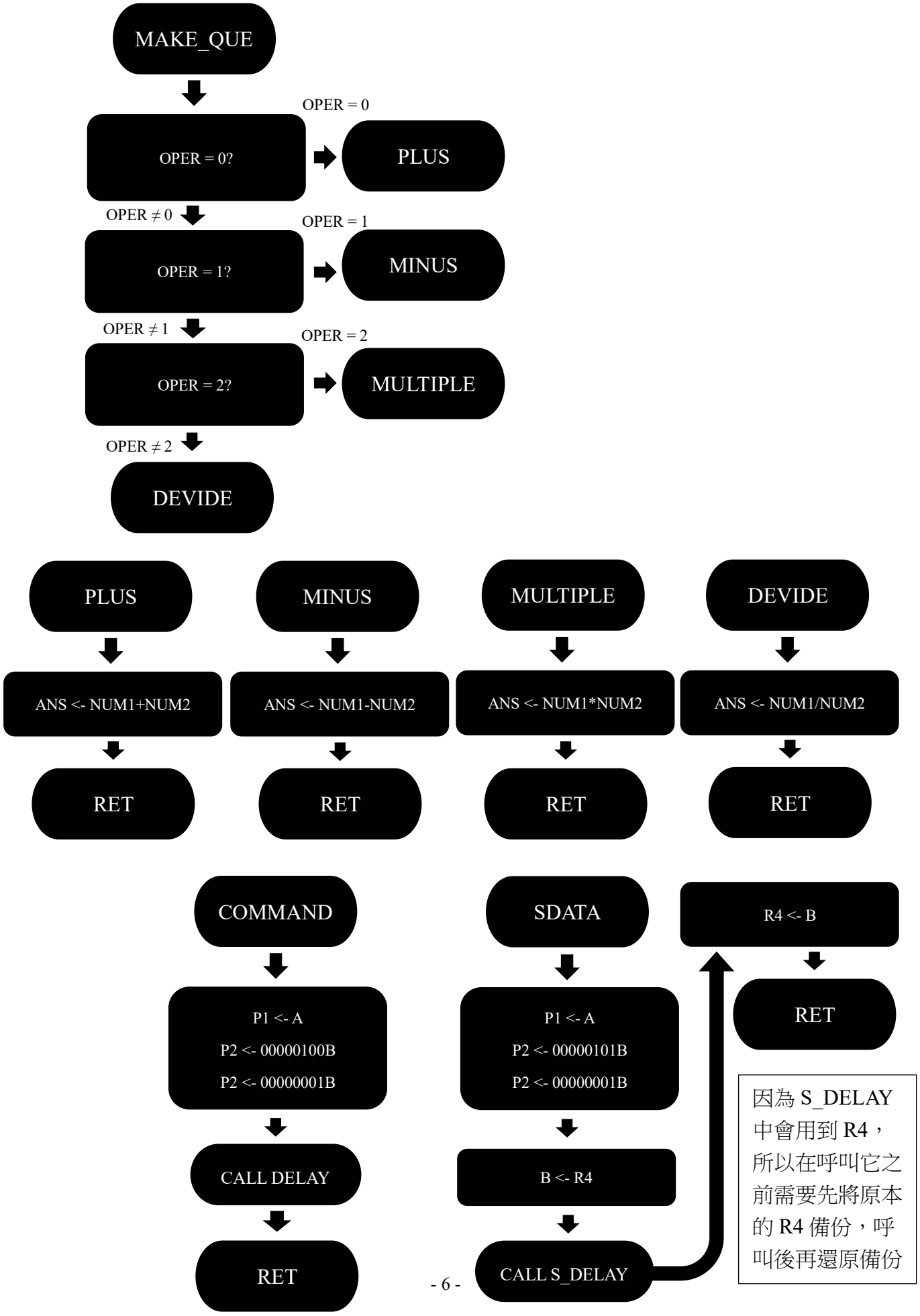


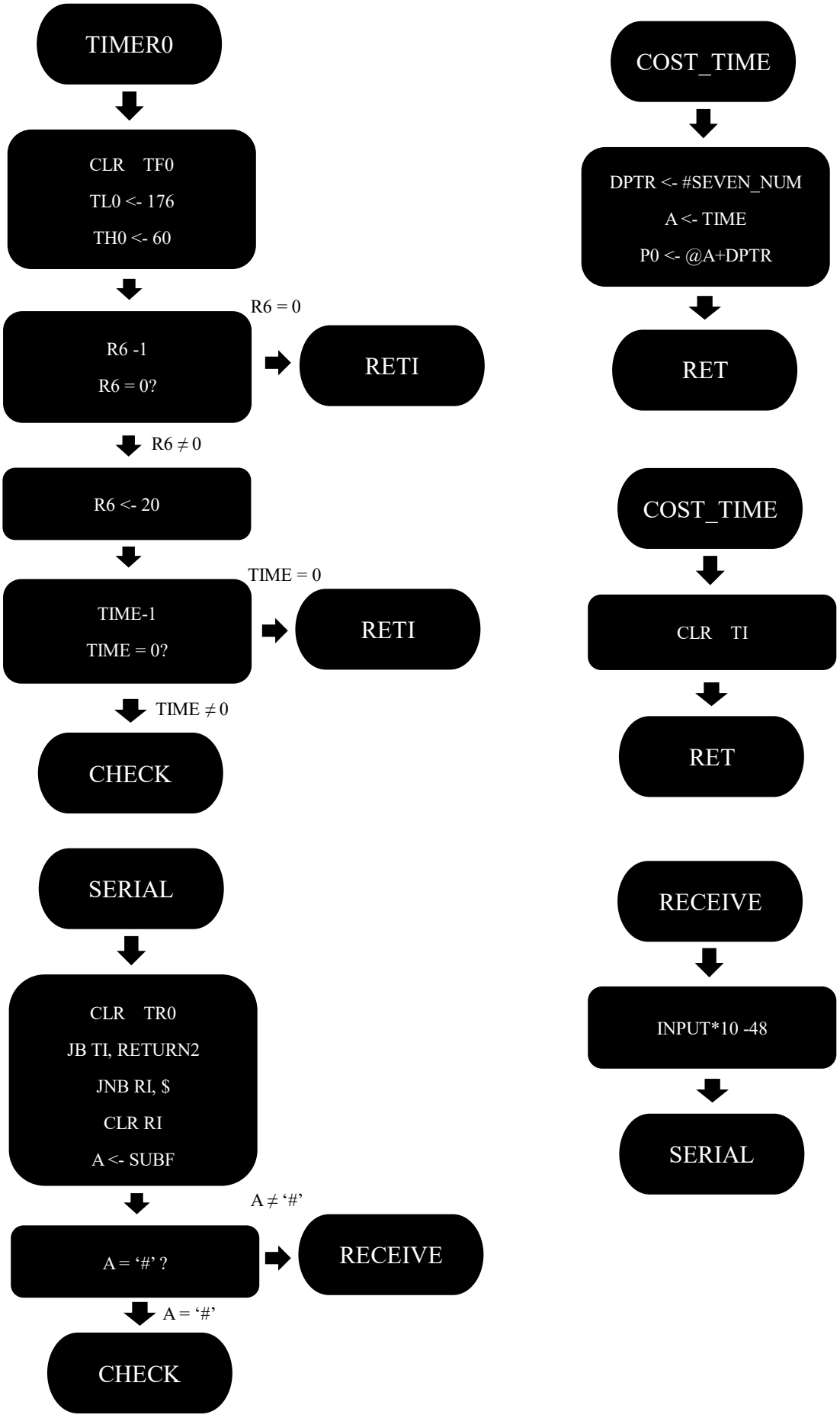
因為我鍵盤讀取的時候數值是存在 NUM2 的暫存器(R1)，所以在顯示 NUM1 的時候需要先把 R1 的數值輸入 NUM1，NUM2 的時候就不用。另外因為鍵盤輸出一次是要顯示兩位數，所以我先 RL 輸出第一位數，輸出第二位數時重新提取數值，並在 RL 後+1，達到顯示兩位數的功能。

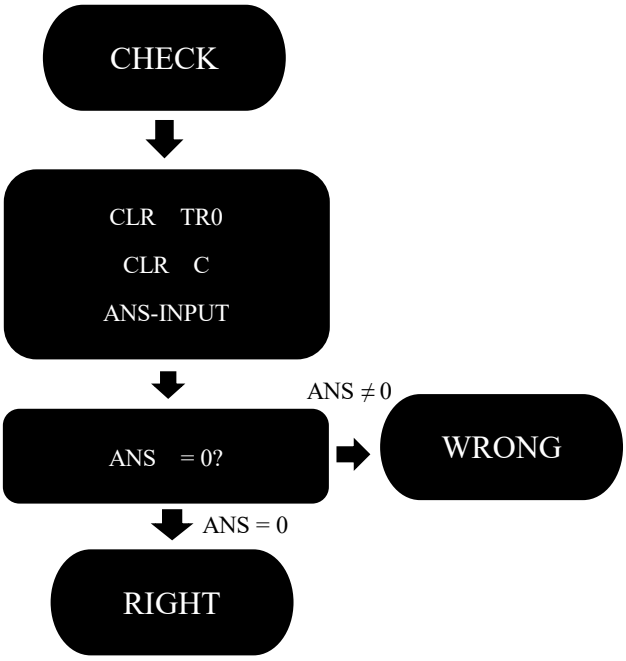

MAKE_NUM_BEGIN
B <- 4
MAKE_NUM
P3 <- 01111111B
A <- R4
ACALL S_DELAY
R4 <- A
A <- P3
A = 01110111B?
R1 <- 12
B <- 5
MAKE_NUM
A = 01111011B?
R1 <- 8
B <- 5
MAKE_NUM
A = 01111101B?
R1 <- 4
B <- 5
MAKE_NUM
A = 01111110B?
R1 <- 0
B <- 5
MAKE_NUM
P3 <- 10111111B
A <- R4
ACALL S_DELAY
R4 <- A
A <- P3
A = 10110111B?
R1 <- 13
B <- 5
MAKE_NUM
A = 10111011B?
R1 <- 9
B <- 5
MAKE_NUM
A = 10111101B?
R1 <- 5
B <- 5
MAKE_NUM
A = 10111110B?
R1 <- 1
B <- 5
MAKE_NUM

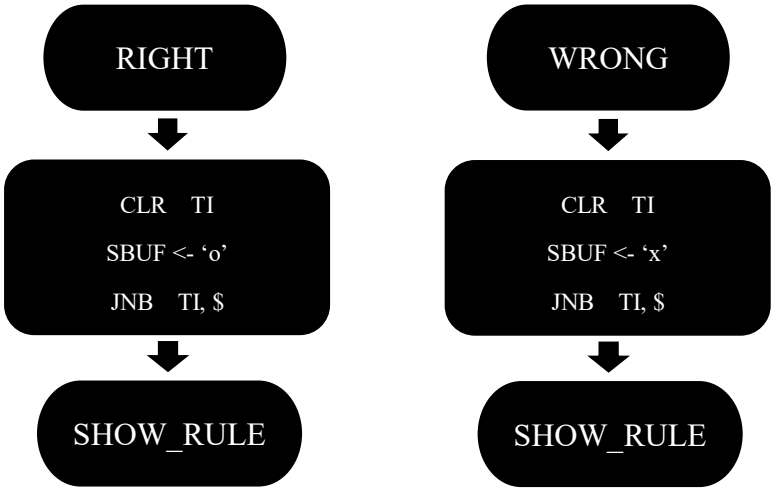


B 在這裡的作用類似於 debounce，為了避免長按案件時鍵盤重複讀取，因此會在開始讀取前先將 B 設為 4，並在讀取到數字後更改 B 的數值，直到 B 被更改且讀取到沒有任何鍵盤按鈕被按下才會吃到數值。


MAKE_QUE
OPER = 0?
OPER = 0
PLUS
OPER ≠ 0
OPER = 1?
OPER = 1
MINUS
OPER ≠ 1
OPER = 2?
OPER = 2
MULTIPLE
OPER ≠ 2
DEVIDE
PLUS
ANS <- NUM1+NUM2
RET
MINUS
ANS <- NUM1-NUM2
RET
MULTIPLE
ANS <- NUM1*NUM2
RET
DEVIDE
ANS <- NUM1/NUM2
RET
COMMAND
P1 <- A
P2 <- 00000100B
P2 <- 00000001B
CALL DELAY
RET
SDATA
P1 <- A
P2 <- 00000101B
P2 <- 00000001B
B <- R4
CALL S_DELAY
R4 <- B
RET
因為 S_DELAY 中會用到 R4，所以在呼叫它之前需要先將原本的 R4 備份，呼叫後再還原備份


TIMER0
CLR TF0
TL0 <- 176
TH0 <- 60
R6 -1
R6 = 0?
R6 = 0
RETI
R6 ≠ 0
R6 <- 20
TIME-1
TIME = 0?
TIME = 0
RETI
TIME ≠ 0
CHECK
SERIAL
CLR TR0
JB TI, RETURN2
JNB RI, $
CLR RI
A <- SUBF
A = '#' ?
A ≠ '#'
RECEIVE
A = '#'
CHECK
COST_TIME
DPTR <- #SEVEN_NUM
A <- TIME
P0 <- @A+DPTR
RET
COST_TIME
CLR TI
RET
RECEIVE
INPUT*10 -48
SERIAL


CHECK
CLR TR0
CLR C
ANS-INPUT
ANS = 0?
ANS ≠ 0
WRONG
ANS = 0
RIGHT


RIGHT
CLR TI
SBUF <- 'o'
JNB TI, $
SHOW_RULE
WRONG
CLR TI
SBUF <- 'x'
JNB TI, $
SHOW_RULE

四、程式碼與註解：

```
    //定義會用到的變數名稱
    NUM1 EQU R0
    NUM2 EQU R1
    ANS EQU R2
    INPUT EQU R3
    TIME EQU R4
    OPER EQU R5

    ORG 00H
    AJMP INIT
    ORG 0BH
    AJMP TIMER0
    ORG 23H
    AJMP SERIAL
    ORG 50H

INIT:
    //設定Timer
    MOV TMOD, #00100001B
    MOV TL0, #176
    MOV TH0, #60
    MOV R6, #20     //20*50ms=15
    MOV TL1, #0E6H  //鮑率2400Hz
    MOV TH1, #0E6H
    CLR TF0
    CLR TF1
    SETB TR1

    //設定serial port mode 1
    ORL PCON, #80H
    CLR SM0
    SETB SM1
    CLR SM2
    SETB REN
    CLR RI
    CLR TI

    //設定LCM
    MOV     A, #00111011B //8BIT, 2ROW, 5X7
    ACALL   COMMAND //寫入指令
    MOV     A, #00001110B //DISPLAY DDRAM, CURSOR, BLINK
    ACALL   COMMAND //寫入指令
    MOV     A, #00000001B //清除顯示
    ACALL   COMMAND //寫入指令
    ACALL   L_DELAY
    MOV     A, #10000000B //DDRAM歸0
    ACALL   COMMAND //寫入指令

    //設定中斷
    MOV IE, #10010010B
    MOV IP, #00010000B
```

```
53      //初始化數值
54      MOV OPER, #0
55      MOV ANS, #0
56      MOV INPUT, #0
57
58  //顯示規則
59  SHOW_RULE:
60      MOV R0, #0
61      MOV DPTR, #RULE
62      MOV R6, #0
63
64  LOOP:
65      CLR     A //清除A
66      MOV     A, R0
67      MOVC    A, @A+DPTR
68      CJNE    A, #15, NEXT
69      ACALL   L_DELAY
70      MOV     A, #00000001B //清除顯示
71      ACALL   COMMAND //寫入指令
72      ACALL   L_DELAY
73      MOV     A, #10000000B //DDRAM歸0
74      ACALL   COMMAND //寫入指令
75      MOV     R3, #0
76      MOV     R4, #0
77      AJMP    START
78  NEXT:
79      ACALL   SEND
80      INC     R0
81      AJMP    LOOP
```

```
82  SEND:
83      MOV     P1, A
84      MOV     P2, #00000101B   //ENABLE, WRITE, DR
85      ACALL   DELAY
86      MOV     P2, #00000001B   //DISABLE, WRITE, DR
87      ACALL   DELAY
88      RET
89  RULE:
90      DB      "Press 2 numbers", 15
91
92  START:
93      MOV TIME, #9
94      MOV R6, #20
95
96      //+_*/
97      INC OPER
98      MOV A, OPER
99      MOV B, #4
100     DIV AB
101     MOV OPER, B
102
103     MOV NUM1, #0
104     MOV NUM2, #0
105
106     //按NUM1
107     ACALL MAKE_NUM_BEGIN
108     MOV A, R1
109     MOV NUM1, A
```

```
111     //顯示NUM1
112     MOV DPTR, #LCM_NUM
113     MOV A, NUM1
114     RL  A
115     MOVC A, @A+DPTR
116     ACALL SDATA
117     MOV A, NUM1
118     RL  A
119     INC A
120     MOVC A, @A+DPTR
121     ACALL SDATA
122
123     ACALL MAKE_QUE
124     MOV DPTR,#LCM_OPER
125     MOV A, OPER
126     MOVC A, @A+DPTR
127     ACALL SDATA
128
129     //按NUM2
130     ACALL MAKE_NUM_BEGIN
131
132     //顯示NUM2
133     MOV DPTR, #LCM_NUM
134     MOV A, NUM2
135     RL A
136     MOVC A, @A+DPTR
137     ACALL SDATA
138     MOV A, NUM2
139     RL A
140     INC A
141     MOVC A, @A+DPTR
142     ACALL SDATA
```

```
144     //顯示=
145     MOV A, #3DH
146     ACALL SDATA
147
148     SETB TR0//啟動計時
149
150 DISPLAY:
151     //ACALL SHOW_QUE //顯示題目
152     ACALL COST_TIME //顯示倒計時
153     CJNE TIME, #0, DISPLAY //時間還沒到就重複顯示
154
155     CLR TR0 //停止計時
156     AJMP START
157
158 //按鍵盤產生數字
159 MAKE_NUM_BEGIN:
160     MOV B, #4
```

```
MAKE_NUM:
    MOV P3, #01111111B
    MOV A, R4
    ACALL S_DELAY
    MOV R4, A
    MOV A, P3
    //12
    CJNE    A, #01110111B, NEXT_8 //從最下面一列開始檢查
    MOV     R1, #12 //個位數是2
    MOV     B, #5
    AJMP    MAKE_NUM
NEXT_8:
    CJNE     A, #01111011B, NEXT_4 //換到倒數第二列
    MOV     R1, #8
    MOV     B, #5
    AJMP    MAKE_NUM
NEXT_4:
    CJNE    A, #01111101B, NEXT_0 //換到倒數第三列
    MOV     R1, #4
    MOV     B, #5
    AJMP    MAKE_NUM
NEXT_0:
    CJNE    A, #01111110B, NEXT_13 //換到最上面一列
    MOV     R1, #0
    MOV     B, #5
    AJMP    MAKE_NUM
```

```
NEXT_13:
    MOV     P3, #10111111B //換到第二行
    MOV A, R4
    ACALL S_DELAY
    MOV R4, A
    MOV     A, P3
    CJNE    A, #10110111B, NEXT_9 //從最下面一列開始檢查
    MOV     R1, #13 //E
    MOV     B, #5
    AJMP    MAKE_NUM
NEXT_9:
    CJNE    A, #10111011B, NEXT_5 //換到倒數第二列
    MOV     R1, #9
    MOV     B, #5
    AJMP    MAKE_NUM
NEXT_5:
    CJNE    A,#10111101B, NEXT_1 //換到倒數第三列
    MOV     R1,#5
    MOV     B, #5
    AJMP    MAKE_NUM
NEXT_1:
    CJNE    A, #10111110B, NEXT_14 //換到最上面一列
    MOV     R1,#1
    MOV     B, #5
    AJMP    MAKE_NUM
```

```
NEXT_14:
    MOV     P3, #11011111B //換到第三行
    MOV A, R4
    ACALL S_DELAY
    MOV R4, A
    MOV     A, P3
    CJNE    A, #11010111B, NEXT_10 //從最下面一列開始檢查
    MOV     R1, #14
    MOV     B, #5
    AJMP    MAKE_NUM
NEXT_10:
    CJNE    A, #11011011B, NEXT_6 //換到倒數第二列
    MOV     R1, #10
    MOV     B, #5
    AJMP    MAKE_NUM
NEXT_6:
    CJNE    A, #11011101B, NEXT_2  //換到倒數第三列
    MOV     R1, #6
    MOV     B, #5
    AJMP    MAKE_NUM
NEXT_2:
    CJNE    A, #11011110B, NEXT_15  //換到最上面一列
    MOV     R1,#2
    MOV     B, #5
    AJMP    MAKE_NUM
NEXT_15:
    MOV     P3, #11101111B //換到第四行
    MOV A, R4
    ACALL S_DELAY
    MOV R4, A
    MOV     A, P3
    CJNE    A, #11100111B, NEXT_11  //從最下面一列開始檢查
    MOV     R1, #15
    MOV     B, #5
    AJMP    MAKE_NUM
```

```
NEXT_11:
    CJNE    A, #11101011B, NEXT_7  //換到倒數第二列
    MOV     R1, #11
    MOV     B, #5
    AJMP    MAKE_NUM
NEXT_7:
    CJNE    A, #11101101B, NEXT_3  //換到倒數第三列
    MOV     R1, #7
    MOV     B, #5
    AJMP    MAKE_NUM
NEXT_3:
    CJNE    A,#11101110B, NEXT_NO  //換到最上面一列
    MOV     R1,#3
    MOV     B, #5
    AJMP    MAKE_NUM
```

```
262  //都沒有按按鍵
263  NEXT_NO:
264      MOV     P3, #0FFH   //no
265      MOV     A, B
266      CJNE    A, #4, RETU
267      AJMP    MAKE_NUM
268  RETU:
269      RET
270
271
272  //+-*/
273  MAKE_QUE:
274      CJNE OPER, #0,NOT_PLUS
275      JMP PLUS
276  NOT_PLUS:
277      CJNE OPER, #1, NOT_MINUS
278      JMP MINUS
279  NOT_MINUS:
280      CJNE OPER, #2, NOT_MULTIPLE
281      JMP MULTIPLE
282  NOT_MULTIPLE:
283      JMP DIVIDE
284
285  PLUS:
286      MOV A, NUM1
287      ADD A, NUM2
288      MOV ANS, A
289
290      RET
```

```
292  MINUS :
293      MOV A, NUM1
294      SUBB A, NUM2
295      MOV ANS,A
296
297      RET
298
299  MULTIPLE:
300      MOV A, NUM1
301      MOV B, NUM2
302      MUL AB
303      MOV ANS,A
304
305      RET
306
307  DIVIDE:
308      MOV A, NUM1
309      MOV B, NUM2
310      DIV AB
311      MOV ANS, A
312
313      RET
```

```
COMMAND:
    MOV P1,A
    MOV P2, #00000100B
    //CALL DELAY
    MOV P2, #00000000B
    CALL DELAY
    RET
SDATA:
    MOV P1,A
    MOV P2, #00000101B
    MOV B, R4
    //CALL S_DELAY
    MOV R4, B
    MOV P2, #00000001B
    MOV B, R4
    CALL S_DELAY
    MOV R4, B
    RET

TIMER0:
    CLR TF0
    MOV TL0, #176
    MOV TH0,#60
    DJNZ R6, RETURN
    MOV R6, #20
    DJNZ TIME, RETURN
    ACALL CHECK
RETURN:
    RETI
```

```
COST_TIME:
    MOV DPTR, #SEVEN_NUM
    MOV A, TIME
    MOVC A,@A+DPTR
    MOV P0,A
    RET

//串列傳輸中斷
SERIAL:
    CLR TR0
    JB TI, RETURN2
    JNB RI, $ //接收資料
    CLR RI
    MOV A, SBUF
    CJNE A, #23H, RECEIVE //接收到'#'就停
    ACALL CHECK
RETURN2:
    CLR TI
    RETI
```

```
364  //一位一位接收
365  RECEIVE:
366      CLR C
367      SUBB A, #48
368      MOV 22H,A
369      MOV A, INPUT
370      MOV B, #10
371      MUL AB
372      ADD A, 22H
373      MOV INPUT, A
374      JMP SERIAL
375
376  CHECK: //檢查答索是否正礁
377      CLR TR0
378      MOV TIME, #0
379
380      CLR C
381      MOV A, INPUT
382      SUBB A,ANS
383      MOV ANS, A
384      MOV INPUT, #0
385
386      CJNE ANS, #0, WRONG
387      JMP RIGHT
```

```
389  RIGHT: //正確回傳'o'
390      CLR TI
391      MOV A, #4FH
392      MOV SBUF, A
393      JNB TI,$
394      JMP SHOW_RULE
395  WRONG: //錯誤回傳'x'
396      CLR TI
397      MOV A, #58H
398      MOV SBUF,A
399      JNB TI,$
400      JMP SHOW_RULE
401
402  DELAY:
403      MOV R7, #255
404      DJNZ R7, $
405      RET
406
407  S_DELAY:
408      MOV R7, #100
409  S_DELAY1:
410      MOV R4, #150
411  S_DELAY2:
412      DJNZ R4, S_DELAY2
413      DJNZ R7, S_DELAY1
414      RET
```

```
416  //1s
417  L_DELAY:
418      MOV R7, #71
419  L_DELAY1:
420      MOV R4, #168
421  L_DELAY2:
422      MOV R3, #250
423  L_DELAY3:
424      DJNZ  R3, L_DELAY3
425      DJNZ  R4, L_DELAY2
426      DJNZ  R7, L_DELAY1
427      RET
428
429  LCM_NUM:
430      DB "00"
431      DB "01"
432      DB "02"
433      DB "03"
434      DB "04"
435      DB "05"
436      DB "06"
437      DB "07"
438      DB "08"
439      DB "09"
440      DB "10"
441      DB "11"
442      DB "12"
443      DB "13"
444      DB "14"
445      DB "15"
446  LCM_OPER:
447      DB "+-*/"
```

```
449  //9-0
450  SEVEN_NUM:
451      DB 090H
452      DB 080H
453      DB 0F8H
454      DB 082H
455      DB 092H
456      DB 099H
457      DB 0B0H
458      DB 0A4H
459      DB 0F9H
460      DB 0C0H
461
462      END
```

五、心得：

期末專題的時間剛好是期末考週跟其他科最後作業的繳交期限，所以本來的時間就有限，再加上最近可能天氣變化比較大所以我一直發燒，導致有些原本預計要實現的部分都沒有辦法實現（比如說加入 score 的計分功能）。這也是我第一次把 R0~R7 用完。印象最深刻的是在實驗過程中 LCD 一直出現問題，有時候無法顯示有時候在燒錄完會變成全黑，後來有同學建議我檢查自己的線路，看是不是接觸的問題，所以我最後沒有把 LCD 的板子直接接在它的銜接板上，而是另外拉線出來在街上銜接版，之後就好很多了，另外一個則是 LCM 在燒錄後插拔會導致全黑，問過同學才知道是為電位改變導致混亂所以才會再插拔的時後出線錯誤。這次也因為用到很多不同的外接模組，所以在 debug 的時候格外麻煩，找了很多同學幫忙。然後最後在錄製 Demo 影片時我的 LCD 銜接版上的 GND 接腳直接掉了，所以 Demo 影片就沒有錄好。