微介實驗四

算術及邏輯運算指令

日期：10月22日

報告者：蕭力文

Outline

- 學習重點
- 實驗內容
- 實驗原理
- 算術運算實驗
- 邏輯運算實驗

Outline

- 學習重點
- 實驗內容
- 實驗原理
- 算術運算實驗
- 邏輯運算實驗

學習重點

- 熟悉組合語言中算術運算指令(如：ADD、SUBB、MUL、DIV等)。
- 熟悉組合語言中邏輯運算指令(如：ANL、ORL、CPL等)。
- 了解8051與運算相關的特殊功能暫存器 (如：PSW、ACC、B等)。

Outline

- 學習重點
- 實驗內容
- 實驗原理
- 算術運算實驗
- 邏輯運算實驗

實驗內容

- 算術運算實驗：

  計算(32H – 06H) × 07H 與(64H + 0DH) ÷ 06H ，觀察Debug Mode 中的特殊功能暫存器，分別找出A、B暫存器中的值。

- 邏輯運算實驗：

  先使用AND邏輯運算將#10111101B 中的第4位元變為0，存入暫存器B，再使用OR邏輯運算將 #01000010B 中的第4位元變為1，最後透過NOT邏輯運算，使第二個值與第一個值相同。

Outline

- 學習重點
- 實驗內容
- 實驗原理
- 算術運算實驗
- 邏輯運算實驗

實驗原理

- 特殊功能暫存器(Special Function Register，簡稱SFR)
  - 位址0080H到00FFH之間。
  - 有21 (8051) / 26 (8052)個特殊功能暫存器，如P0、P1、SBUF、IE等。8052多一個timer。
  - 與算術及邏輯運算相關的為PSW、ACC、B。

F8									FF
F0	B								F7
E8									EF
E0	ACC								E7
D8									DF
D0	PSW								D7
C8	T2CON		RCAP2L	RCAP2H	TL2	TH2			CF
C0									C7
B8	IP								BF
B0	P3								B7
A8	IE								AF
A0	P2								A7
98	SCON	SBUF							9F
90	P1								97
88	TCON	TMOD	TL0	TL1	TH0	TH1			8F
80	P0	SP	DPL	DPH				PCON	87

實驗原理— PSW

- PSW (Program Status Word程式狀態字組)

  為8位元的暫存器，其位址在00D0H。

  功能為顯示8051在執行過程中的狀態。

	7	6	5	4	3	2	1	0
PSW	CY	AC	F0	RS1	RS0	OV	–	P

實驗原理— PSW

- PSW.7　CY (Carry Flag進位旗標)

  在執行加法/減法運算時，若第7位元有進位/借位時，CY = 1；
  沒有進位/借位，CY = 0。

- PSW.6　AC (Auxiliary Carry Flag輔助進位旗標)

  在執行加法/減法運算時，若第3位元有進位/借位時，AC = 1；
  沒有進位/借位，AC = 0。

- PSW.5　F0 (Flag 0)

  為使用者自訂旗標，由使用者自行設定的位元。

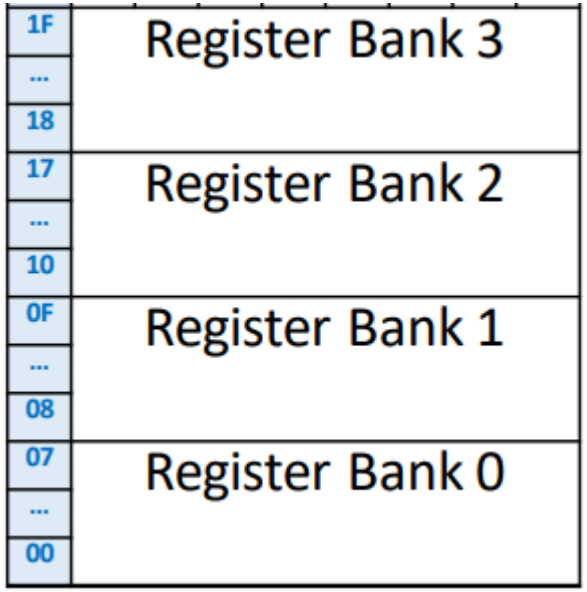
實驗原理— PSW

- PSW.3與PSW.4

RS0 (Register Bank Selector 0暫存器庫選擇位元0) 與
RS1 (Register Bank Selector 1暫存器庫選擇位元1)

功能為選擇控制哪一組暫存器庫工作。

RS1	RS0	暫存器庫	位址
0	0	RB0	0000H ~ 0007H
0	1	RB1	0008H ~ 000FH
1	0	RB2	0010H ~ 0017H
1	1	RB3	0018H ~ 001FH



實驗原理— PSW

- PSW.2 OV (Overflow Flag溢位旗標)

  考慮有符號的運算，若運算結果發生溢位，則OV = 1；反之，則OV = 0。

- PSW.1 Reserved Flag保留旗標

  沒有提供服務。

- PSW.0 P (Parity Flag同位旗標)

  8051採偶同位(Even Parity Check)，若ACC裡1的個數為奇數個，則P = 1；有偶數個，則P = 0。

實驗原理

- ACC (Accumulator累加器)

  又稱為A暫存器，為8位元暫存器，其位址在00E0H。是CPU主要運作的位置，是使用頻率最高的暫存器。

- B (Base Register基底暫存器)

  為8位元暫存器，其位址在00F0H。主要功能為乘法及除法運算時， 和ACC搭配使用，也可作為一般暫存器使用。

算術運算指令

- ADD A, <src-byte>

  將ACC的值與 src-byte 的值相加，並且將結果存回ACC。

- ADDC A, <src-byte>

  將ACC的值與 src-byte 的值、進位位元(CY)的值相加，並將結果存回ACC。

- SUBB A, <src-byte>

  將ACC的值減去 src-byte 的值、進位位元(CY)的值，並將結果存回ACC。

算術運算指令— 減法

- SUBB A, #09H

執行前	→	運算過程	→	執行後
A = 05H、CY = 1		A = A - CY - #08H		A = 0FCH

CY	AC	OV	P
1	0	0	0

CY	AC	OV	P
1	1	0	1

借位，設定CY=1　　借位，設定AC=1

```
      0   0   0   0 | 0   1   0   1   → 05H  → 5
  -   0   0   0   0 | 1   0   0   1   → 09H  → 9
  ------------------+------------------------------
      1   1   1   1 | 1   1   0   0   → 0FCH  → -4
```

註：這只是驗證旗標的狀態，實際上只需知道各個旗標的意思就可以了。

算術運算指令— 乘法

- MUL AB

		A	→ 被乘數(無符號8位元)
X		B	→ 乘數(無符號8位元)
	B	A	高8位元→ 存回B、低8位元→ 存回ACC

- 乘積大於0FFH(255)時，OV = 1。
- 根據ACC的值，同位旗標P 會有變化。

算術運算指令— 乘法

- MUL AB

執行前：A = 030H、B = 012H ; A = 48D、B = 18D

執行後：A = 060H、B = 003H ; A = 96D、B = 03D (3×256=768)

A+B=96+768=864

					0	0	1	1	0	0	0	0
				x	0	0	0	1	0	0	1	0
				0	0	1	1	0	0	0	0	
			+	0	0	1	1	0	0	0	0	
			0	0	1	1	0	1	1	0	0	0	0	0
			3				6			0		
			B				A					

OV	P
1	0

算術運算指令— 除法

- DIV AB

除數(無符號8位元) ← B | A → 被除數(無符號8位元)

商數 ← A …B → 餘數

- 除數為0時，OV = 1。
- 根據ACC的值，同位旗標P 會有變化。

算術運算指令— 除法

- DIV AB

  執行前：A = 030H、B = 012H ; A = 48D、B = 18D

  執行後：A = 002H、B = 00CH ; A = 2D、B = 12D

```
      2    → A
   ┌─────
18 │ 48
     36
   ──────
     12    → B
```

OV	P
0	1

邏輯運算指令

- ANL <dest-byte>,<src-byte>

  將 dest-byte 與 src-byte 執行 AND 邏輯運算，並將結果存回 dest-byte 。

- ORL <dest-byte>,<src-byte>

  將 dest-byte 與 src-byte 執行 OR 邏輯運算，並將結果存回 dest-byte 。

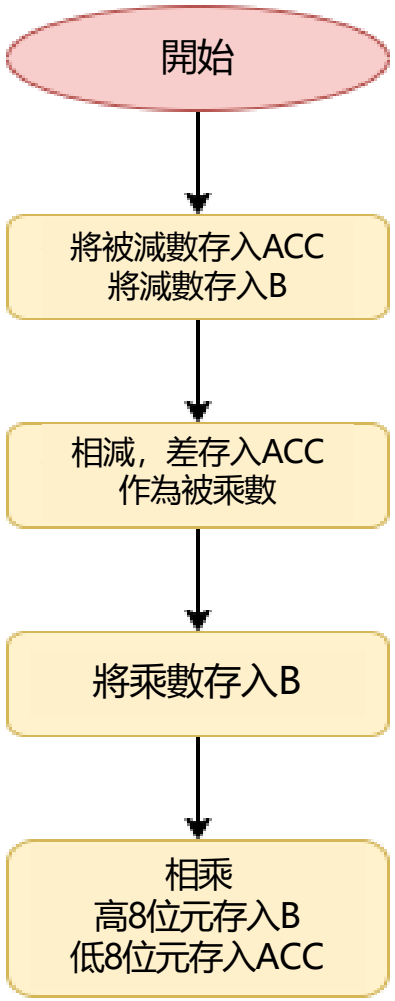
- CPL

  將運算元執行NOT邏輯運算，並將結果存回運算元中。

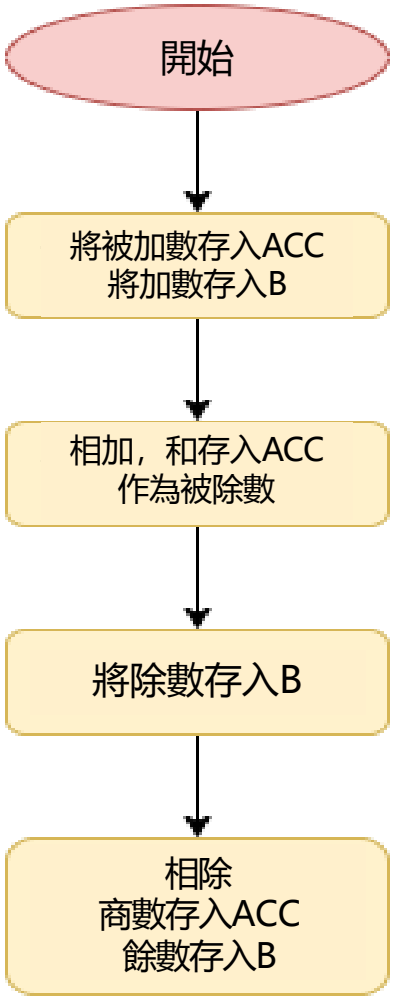
Outline

- 學習重點
- 實驗內容
- 實驗原理
- 算術運算實驗
  - 軟體流程圖
  - 範例程式碼
  - 模擬結果
- 邏輯運算實驗

算術運算實驗—軟體流程圖



（32H-06H）× 07H



（64H + 0DH） ÷06H

Outline

- 學習重點
- 實驗內容
- 實驗原理
- 算術運算實驗
  - 軟體流程圖
  - 範例程式碼
  - 模擬結果
- 邏輯運算實驗

算術運算實驗—範例程式碼

- (32H – 6H) × 7H

```
1. ORG 0              ; code start from 0000H
2. MOV A, #32H        ; move 32H into ACC
3. MOV B, #06H        ; move 06H into B
4. SUBB A, B          ; A-B, then save into ACC
5. MOV B, #07H        ; move multiplier 07H into B
6. MUL AB             ; A×B
7. END
```

算術運算實驗—範例程式碼

- (64H + DH) ÷ 6H

```
1. ORG 0              ; code start from 0000H
2. MOV A, #64H        ; move 64H into ACC
3. MOV B, #0DH        ; move 0DH into B
4. ADD A, B           ; A+B, then save into ACC
5. MOV B, #06H        ; move divisor 06H into B
6. DIV AB             ; A÷B
7. END
```

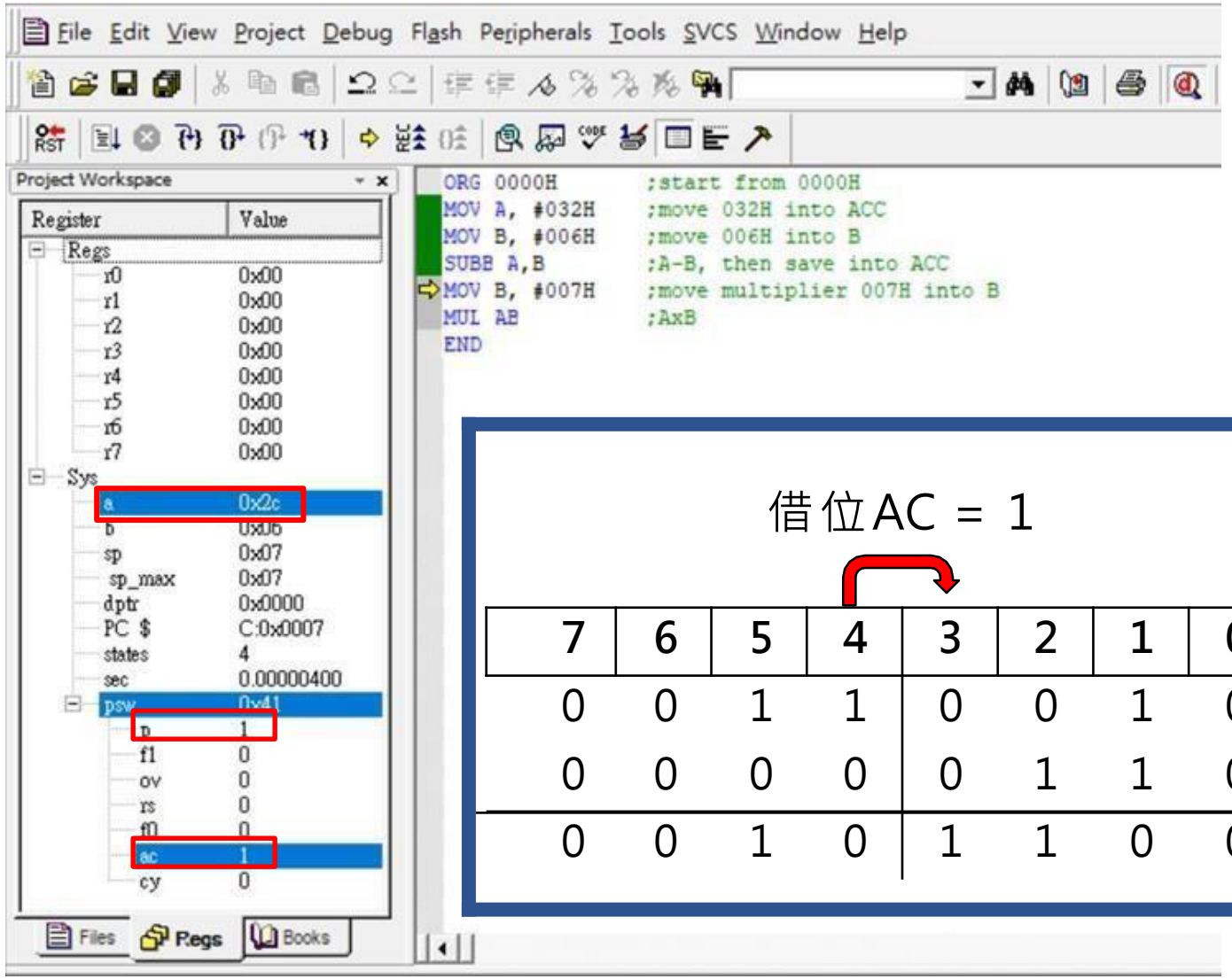
Outline

- 學習重點
- 實驗內容
- 實驗原理
- 算術運算實驗
  - 軟體流程圖
  - 範例程式碼
  - 模擬結果
- 邏輯運算實驗

算術運算實驗—模擬結果

- (32H – 6H) × 7H
  - SUBB A, B

  32-6 存回ACC



借位AC = 1

7	6	5	4	3	2	1	0	
0	0	1	1	0	0	1	0	→ 032H
0	0	0	0	0	1	1	0	→ 006H
0	0	1	0	1	1	0	0	→ 02CH

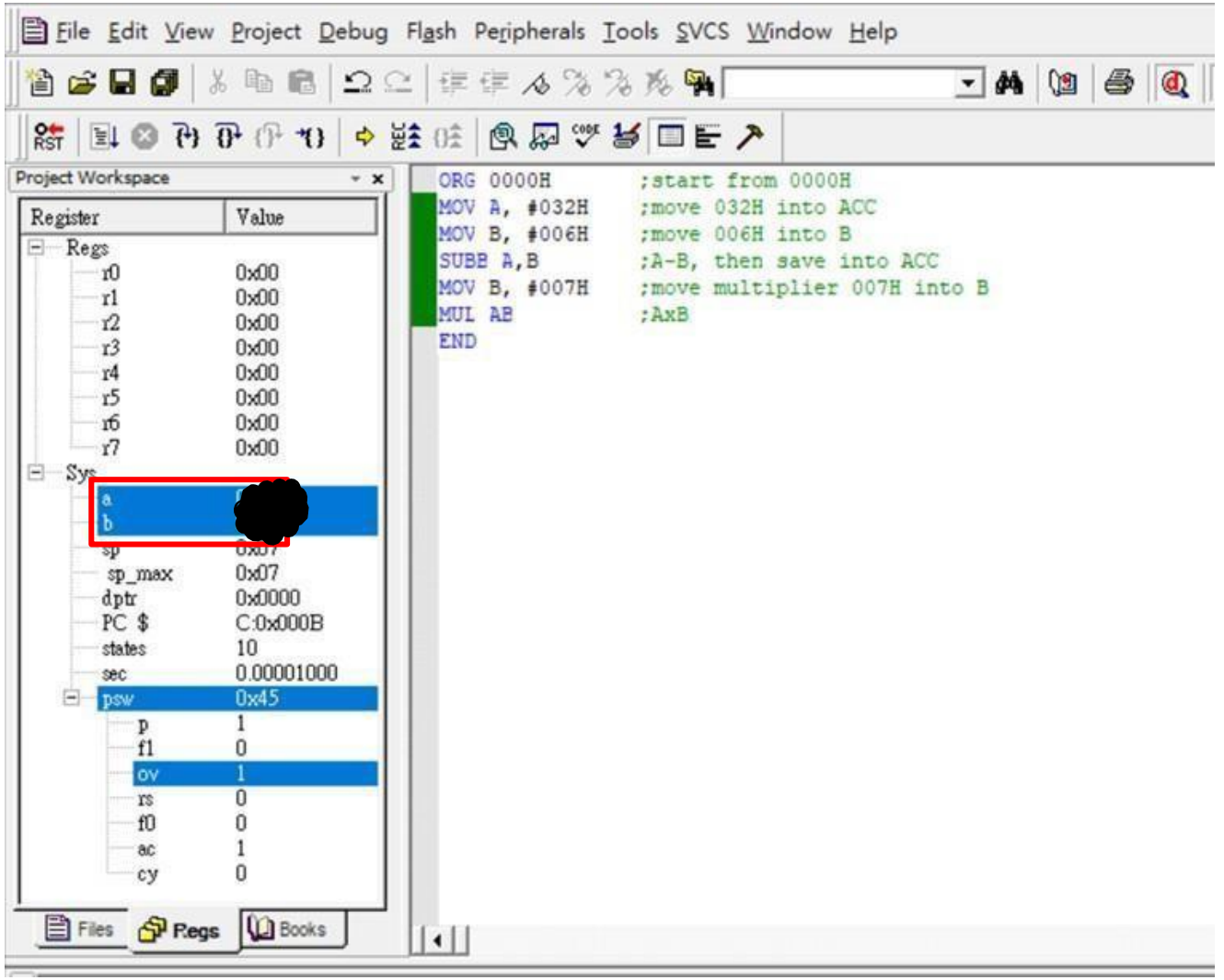
算術運算實驗—模擬結果

- (32H – 6H) × 7H
  - MUL AB

高8位元存回B，
低8位元存回ACC

A = ?

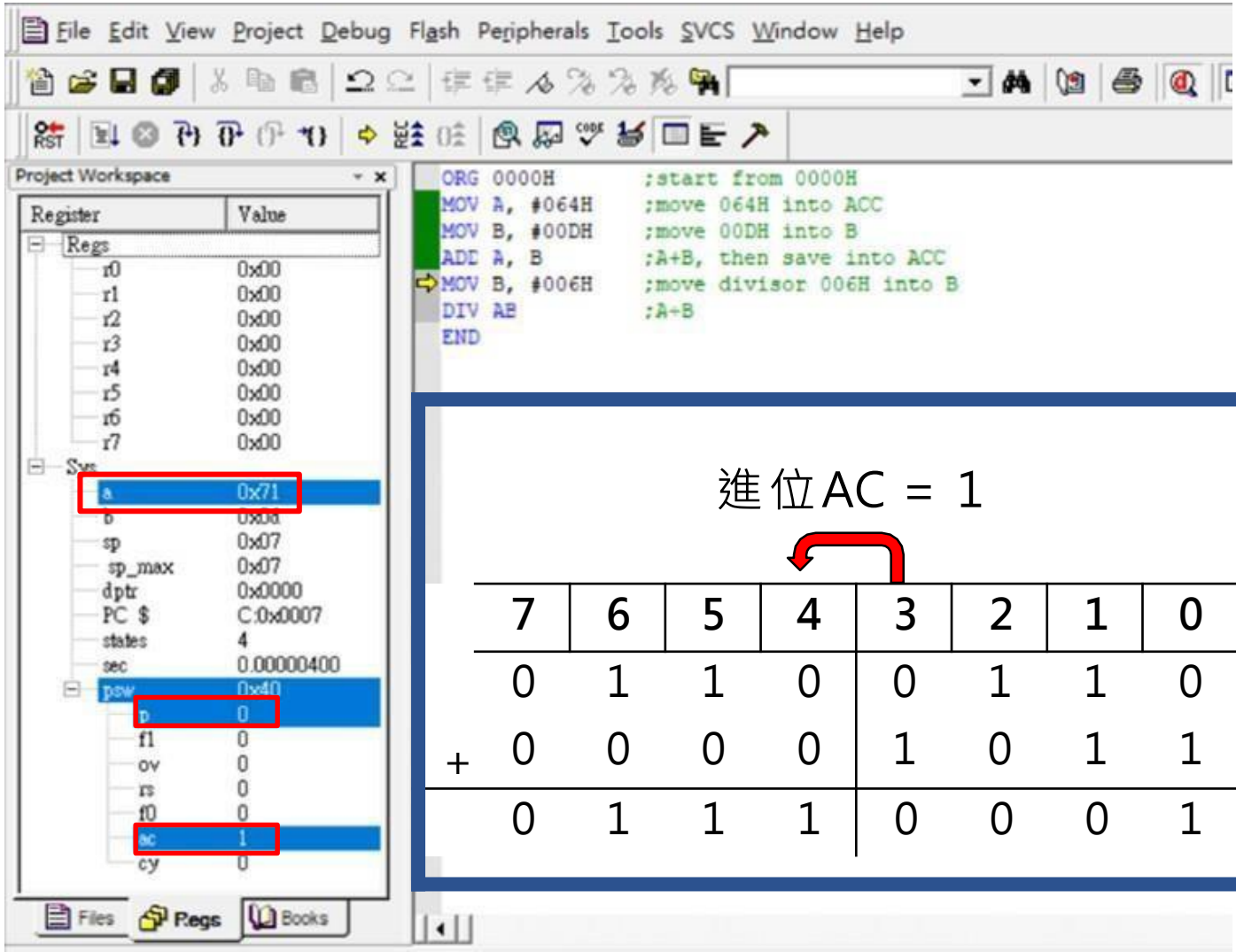
B = ?



算術運算實驗—模擬結果

- (64H + DH) ÷ 6H
  - ADD A, B

  64+D 存回 ACC



進位AC = 1

	7	6	5	4	3	2	1	0		
	0	1	1	0	0	1	1	0	→	064H
+	0	0	0	0	1	0	1	1	→	00DH
	0	1	1	1	0	0	0	1	→	071H

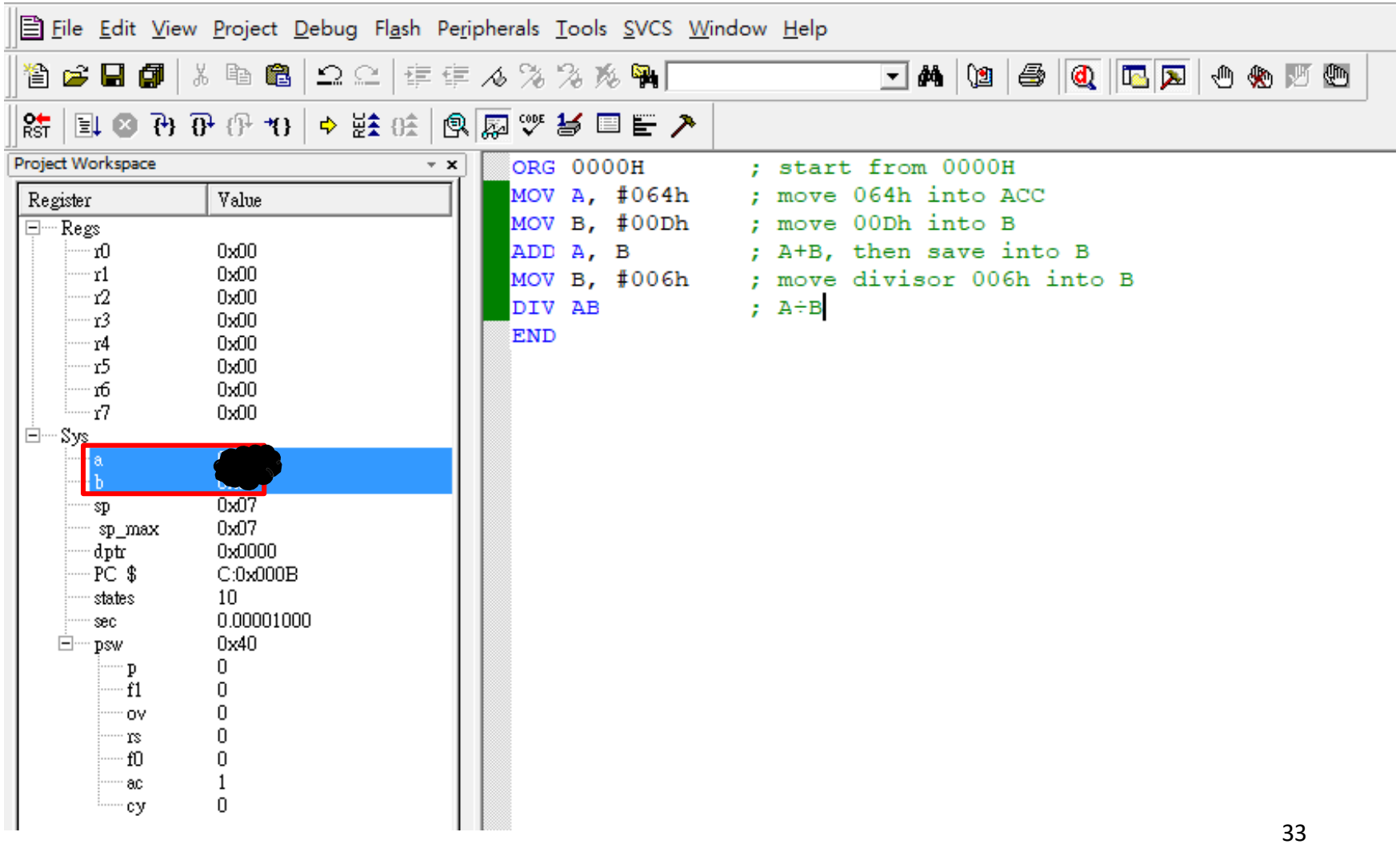
算術運算實驗—模擬結果

- (64H + DH) ÷ 6H
  - DIV AB

餘數存回B，商數存回ACC

A = ?

B = ?

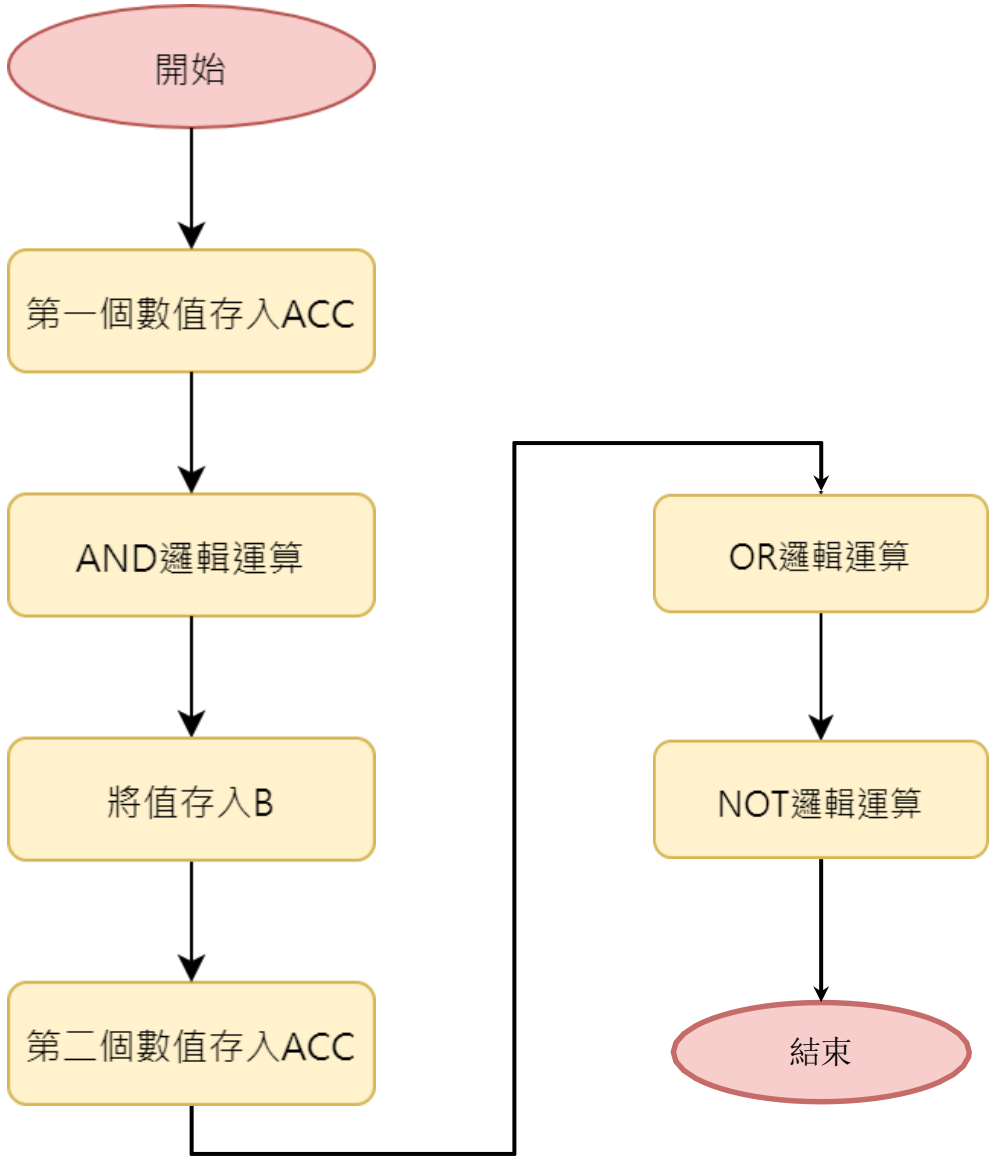


Outline

- 學習重點
- 實驗內容
- 實驗原理
- 算術運算實驗
- 邏輯運算實驗
  - 軟體流程圖
  - 範例程式碼
  - 模擬結果

邏輯運算實驗

—軟體流程圖



Outline

- 學習重點
- 實驗內容
- 實驗原理
- 算術運算實驗
- 邏輯運算實驗
  - 軟體流程圖
  - 範例程式碼
  - 模擬結果

邏輯運算實驗—範例程式碼

```
1. ORG 0                    ; code start from 0000H
2. MOV A, #10111101B        ; move 10111101B into ACC
3. ANL A, #11101111B        ; use AND Gate to turn Bit 4 into 0
4. MOV B, A                 ; move the value into B
5. MOV A, #01000010B        ; move 01000010B into ACC
6. ORL A, #00010000B        ; use OR Gate to turn Bit 4 into 1
7. CPL A                    ; use NOT Gate
8.  END
```

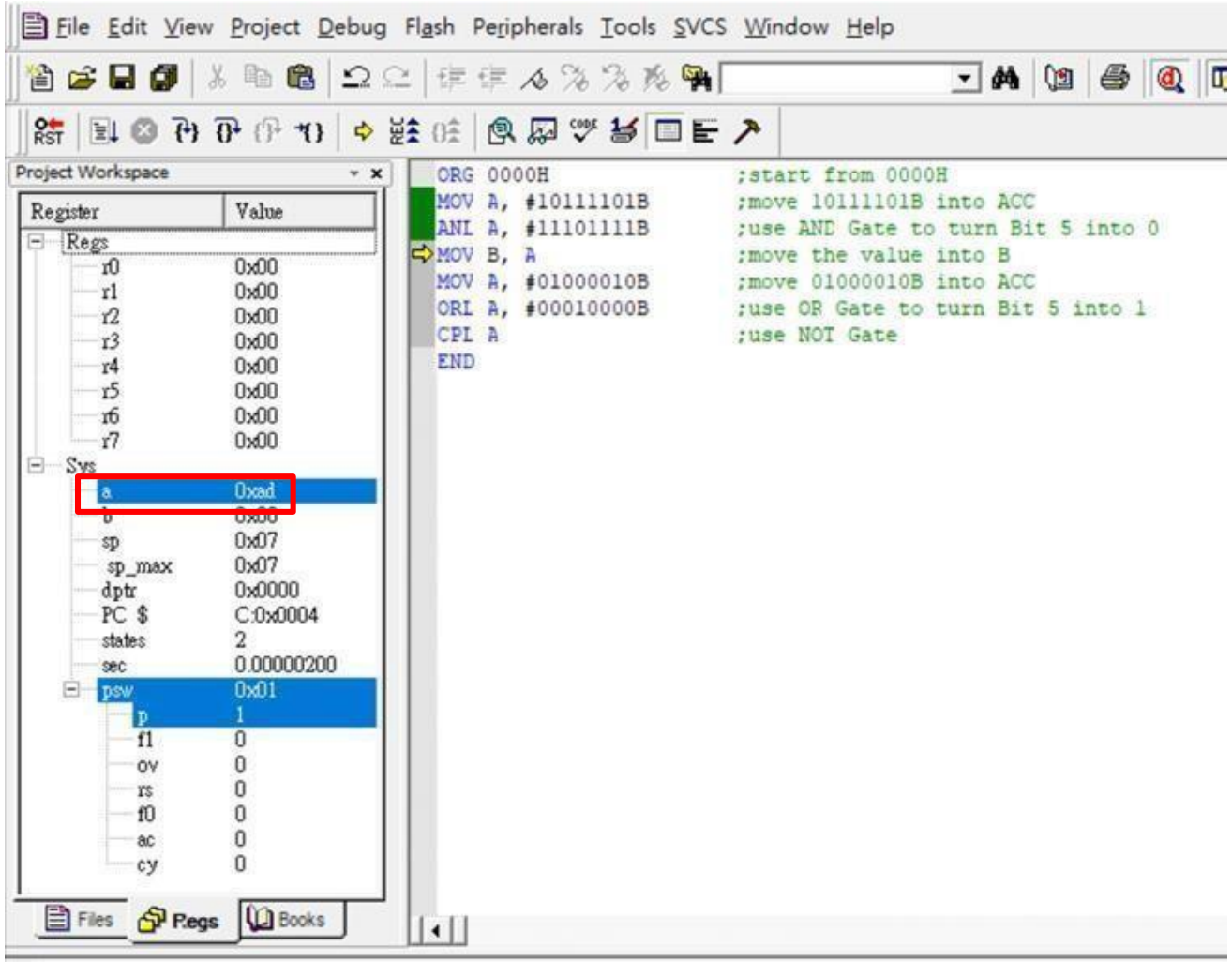
Outline

- 學習重點
- 實驗內容
- 實驗原理
- 算術運算實驗
- 邏輯運算實驗
  - 軟體流程圖
  - 範例程式碼
  - 模擬結果

邏輯運算實驗—模擬結果

- ANL A, #11101111B

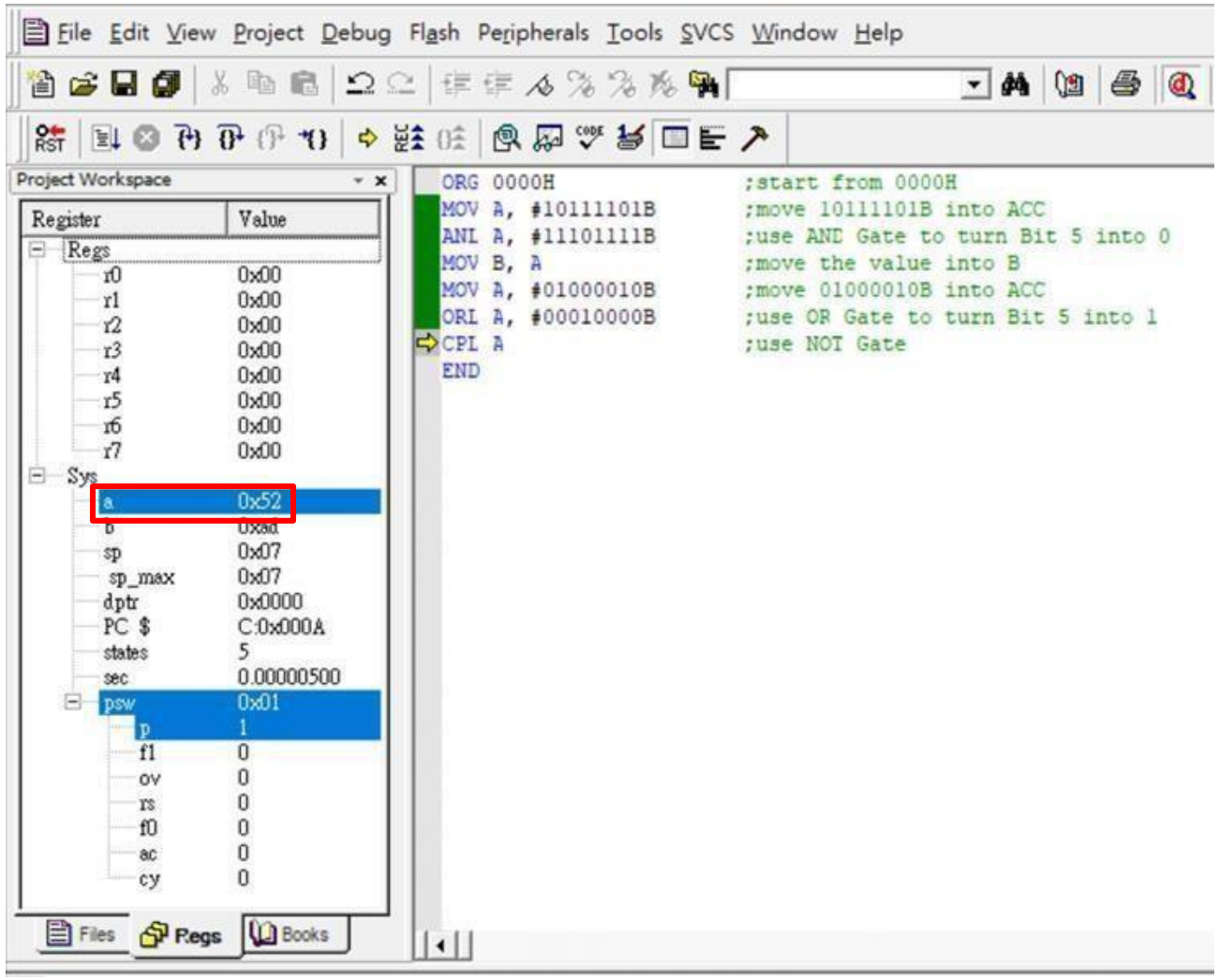
```
  10111101B
AND 11101111B
-------------
  10101101B
```



邏輯運算實驗—模擬結果

- ORL A, #00010000B

```
      01000010B
OR    00010000B
-------------------
      01010010B
```

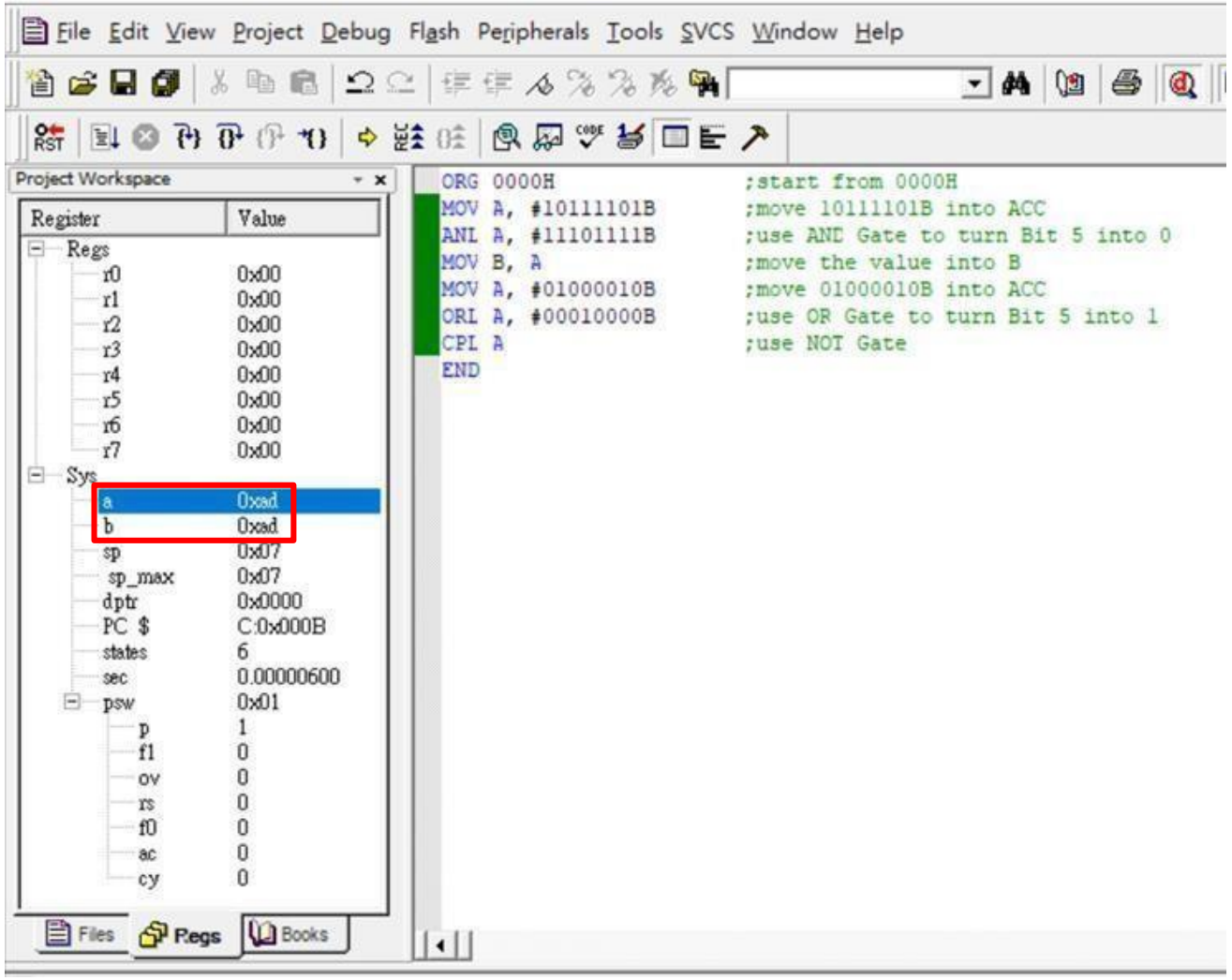


邏輯運算實驗—模擬結果

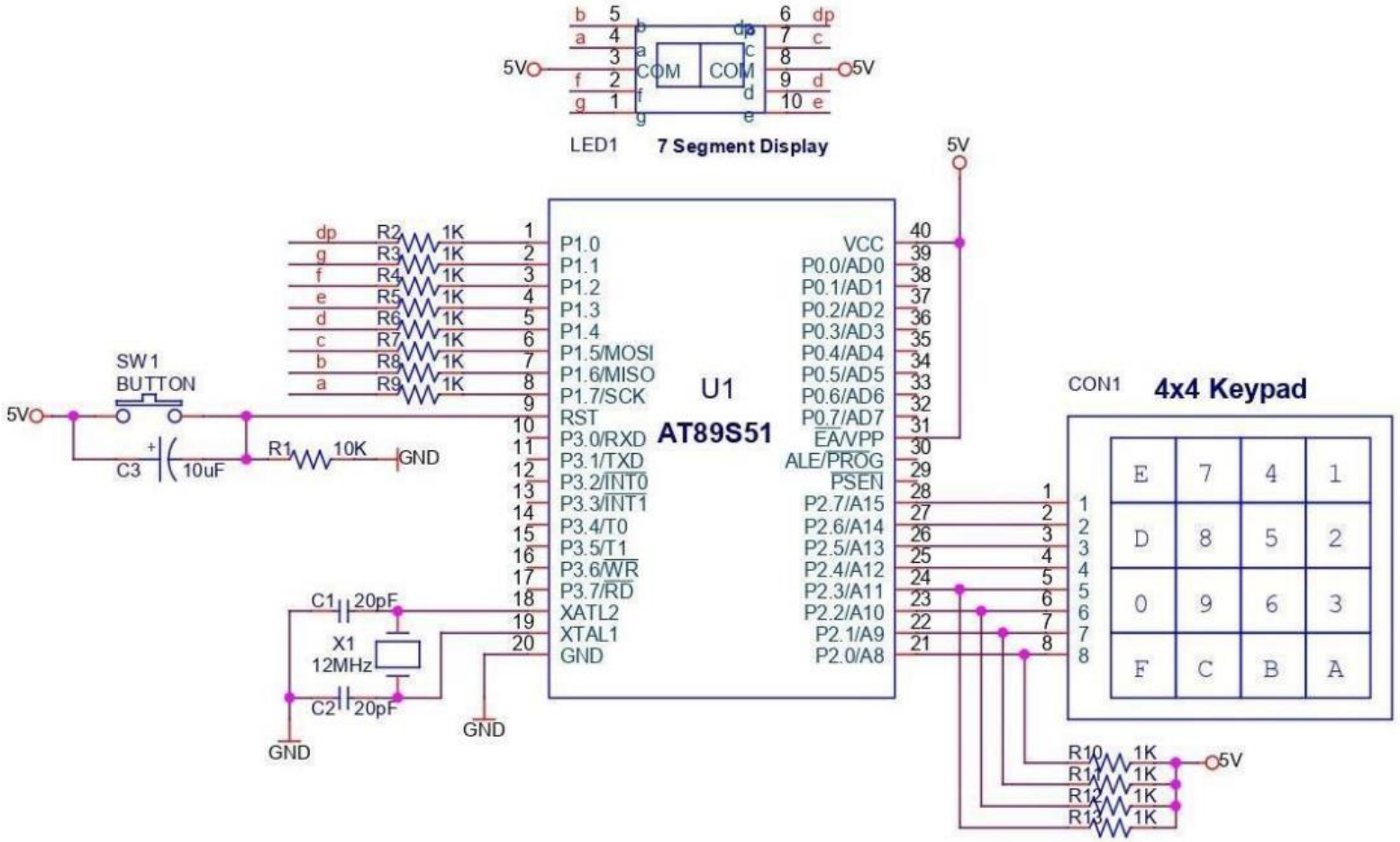
- CPL A

NOT 01010010B

10101101B



進階題參考電路 (同實驗3電路)



Q & A