實驗一 內部 RAM 與 Flash 空間定址與存取

1. 進階題

請利用 Keil C51 的各種功能,找出問題的答案,總共 12 題。

```
A.
 1 ORG 0
 2 MOV RO, #01H
 3 MOV R1, #0FFH
 4 SETB RSO
             ; switch from RB0 to RB1
 5 MOV RO, 00H
                         oFFH
 6 MOV A, 01H → R\
                         01H
 7 MOV A, RO
 8 MOV A, 08H → R<sup>0</sup>
                        01 H
 9 MOVA, @RO→OIH→RI OFFH
10 CLR RSO
            ; switch from RB1 to RB0
                         HIO
11 MOV A, RO
12 SJMP $
13 END
```

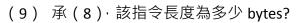
(1) 第6行·A=? Hint: 01H 為暫存器庫 0 R1 的位址。 OFFH
(2) 第7行·A=? Hint: R0 此時為暫存器庫 1 的 R0。 OI H
(3) 第8行·A=? Hint: 08H 為暫存器庫 1 R0 的位址。 OI H
(4) 第9行·A=? Hint: R0 內數值位址所儲存的資料。 OFFH
(5) 第11行·A=? Hint: R0 此時為暫存器庫 0 的 R0。 OI H

Hint: 暫存器庫 0 的 R0~R7 對應的內部 RAM 位址為 00H~07H·暫存器庫 1 的 R0~R7 對應的內部 RAM 位址為 08H~0FH。

```
В.
```

- 1 ORG 0
 2 <u>MOV A</u>, #<u>11H</u>
 3 MOV 0E0H, #22H
 4 SJMP \$
 5 END
- (6) 第2行·A=? | | 「
- (7) 承(6)·該指令長度為多少 bytes?
- (8) 第3行,A=? 72 [-]

2 bytes





- (10) 承 (7)·(9)·為何此兩種指令達成的功能相同·但指令長度卻不一致? 沒者用 兌 赴
 - (A) 因為兩者根本是不同指令,只是兩種指令達成同樣的效果。
 - (B) 因為兩指令在 Flash 空間的位址不同。
 - (C) 因為 MOV 0E0H, #22H 的字數比 MOV A, #11H 還多。

Hint:指令長度可以觀察 disassembly window 將前後位址相減,或者是查詢 8051 指令集獲得,由於 8051 為複雜指令集,代表著每個指令的長度可能不同,通常要用到直接位址的指令會比用暫存器的指令還長,assembler 會將指令轉換後長度不一的 machine code 從 ORG 指定的位址依序填入至 Flash 當中。

C.

- 1 ORG 0
- 2 MOV 20H, #10H
- 3 SETB 20H.2
- 4 MOV 30H, #20H
- 5 **SETB 30H.1**
- 6 *SJMP* \$
- 7 END
- (11) 為何此段程式碼無法成功編譯?
- (A) 因為不存在 SETB 指令。
- (B) 因為 SETB XXH.X 的位址表示形式不合乎語法。
- (C) 因為 30H 位址不是 bit addressable area。

Hint: 觀察錯誤的行數以及 Build Output 視窗的內容。

D.

```
1
            ORG 0
 2
            MOV DPTR, #TABLE
 3
            MOV R7, #04H
   START:
 4
  LOOP:
            MOV A, R7
                                  R7 X 1
 5
            MOVC A, @A+DPTR
 6
            DJNZ R7, LOOP
 7
            SJMP START
 8
   TABLE:
            DB 5BH
 9
            DB 7CH
10
            DB 1CH
11
            DB 1CH
12
            DB 3FH
13
            END
```

(12) 請依照程式碼順序,寫出 A+DPTR 位址在程式記憶體所儲存的數值。 Hint:該數值將會被儲存至 A 暫存器,只須按照順序寫一次循環即可,可能出現的數值有 5BH、7CH、1CH、3FH。

3FH -> 1CH -> 1CH -> 1CH