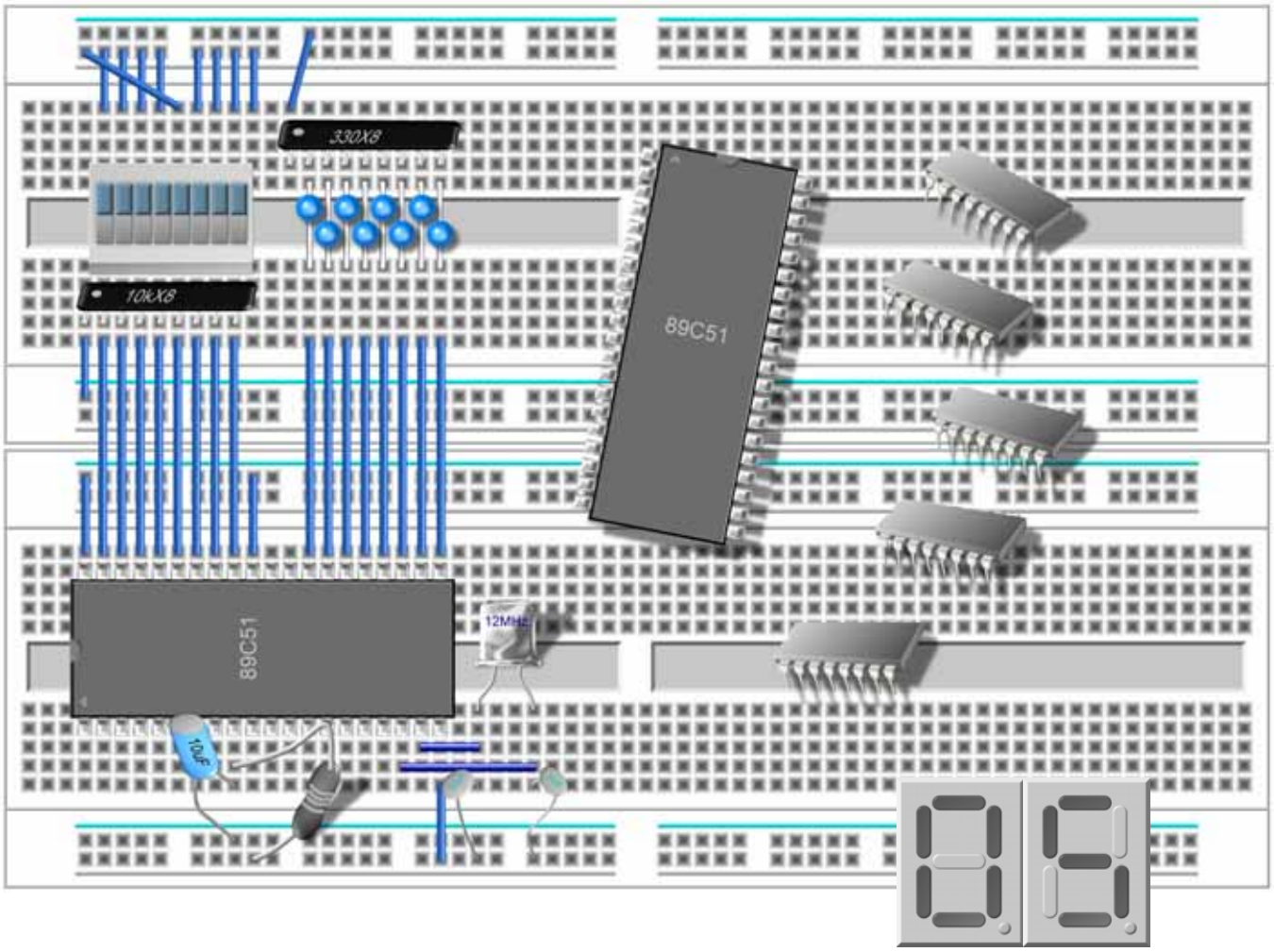


中斷之應用

本章內容豐富，主要包括三部分：

- 硬體部分：

  認識 8051 的中斷功能。

- 指令部分：

  詳細說明邏輯運算指令。

- 程式與實作部分：

  單一中斷的應用程式、兩個中斷的應用程式、鍵盤中斷程式等。

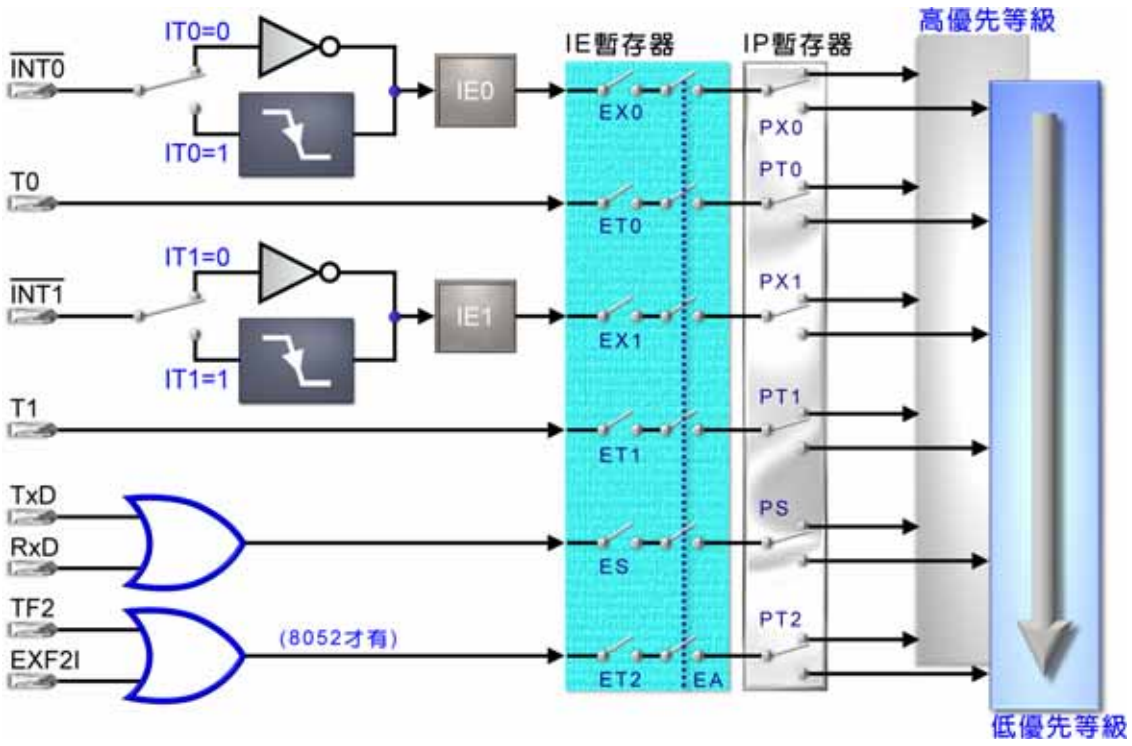
5-1 8051 之中斷

中斷(interrupt)是暫時放下目前所執行的程式，先去執行特定的程式，待完成特定的程式後，再返回剛才放下的程式。譬如說，老師正在講課，而同學有疑問，隨時都可舉手發問，老師將立即暫停課程進度，先爲同學解惑，再繼續剛才暫停的課程。這樣的動作就是「中斷」。

*好端端的幹嘛中斷？*就是爲了提升效率！*中斷能提升效率？*試想若不立即提出問題、立即得到適切答覆，待老師授課完畢，這位同學早就忘光光了，同時也失去興趣了！當然，老師也不能整天待在教室，課也不上，大家大眼瞪小眼，等待同學提問題！所以，以「中斷」的方式，既能保持進度，又兼顧與滿足同學的需求，當然是比較有效率！

5-1-1 MCS-51 之中斷

8051 提供五個中斷服務，即外部中斷 INT0、外部中斷 INT1、計時計數器中斷 T0、計時計數器中斷 T1 與串列埠 RI/TI，如下圖所示：



（圖1） MCS-51 中斷控制系統

8052 提供六個中斷服務，除了 8051 的五個中斷外，還包括第三個計時計數器(Timer 2)的中斷。不過怎樣，其中可概分爲三類，如下說明：

外部中斷

外部中斷有 INT0 與 INT1 兩個，CPU 透過 INT0 接腳(即 12 腳，也就是 P3.2 共用接腳)及 INT1 接腳(即 13 腳，也就是 P3.3 共用接腳)，即可接受外部中斷的請求。

外部中斷信號的採樣方式，可爲準位觸發(低準位觸發)及邊緣觸發(負緣觸發)兩種。若要採用準位觸發，須將 TCON 暫存器中的 IT0(或 IT1)設定爲 0，則只要 INT0 接腳(INT1 接腳)爲低態，即視爲外部中斷需求。若要採用邊緣觸發，須將 TCON 暫存器中的 IT0(或 IT1)設定爲 1，則只要 INT0 接腳(INT1 接腳)的信號由高態轉爲低態瞬間，將視爲外部中斷需求。這些中斷需求將反應在 IE0(或 IE1)裡，若 IE 暫存器的 EX0(或 EX1)=1、且 EA=1，CPU 將進入該中斷的服務。至於，中斷優先等級暫存器(IP 暫存器)只是安排多個中斷發生時，中斷服務執行的順序而已，若只有一個中斷，將不會有所影響。

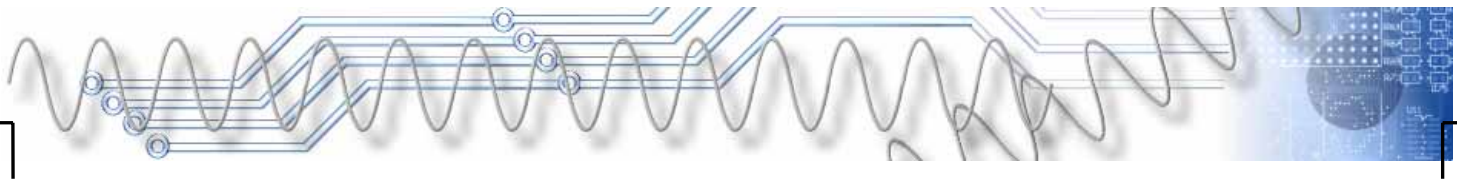
計時計數器中斷

計時計數器中斷有 TF0 與 TF1 兩個(8052 則還有 TF2)，若是計時器，CPU 將計數內部的時鐘脈波，而提出內部中斷；若是計數器，CPU 將計數外部的脈波，而提出內部中斷。至於外部脈波的輸入，則是透過 T0 接腳(即 14 腳，也就是 P3.4 共用接腳)及 T1 接腳(即 15 腳，也就是 P3.5 共用接腳)。關於計時計數器，待第六章再詳細說明。

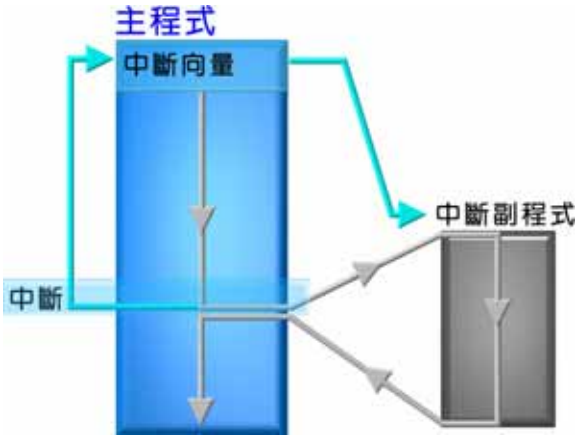
串列埠中斷

串列埠中斷則爲 RI 或 TI 兩個，CPU 透過 RXD 接腳(即 10 腳，也就是 P3.0 共用接腳)及 TXD 接腳(即 11 腳，也就是 P3.1 共用接腳)，要求接收(RI)中斷需求或傳送(TI)中斷需求。關於計時計數器，待第七章再詳細說明。

當中斷發生時，CPU 將暫停當時所執行的程式，立即按中斷的種類，執行其中斷向量，例如 INT0 中斷時，程式將跳到 03H 位址(INT0 的中斷向量)，而 03H 位址可能只有一個命令，即「JMP　XXX」，其中的

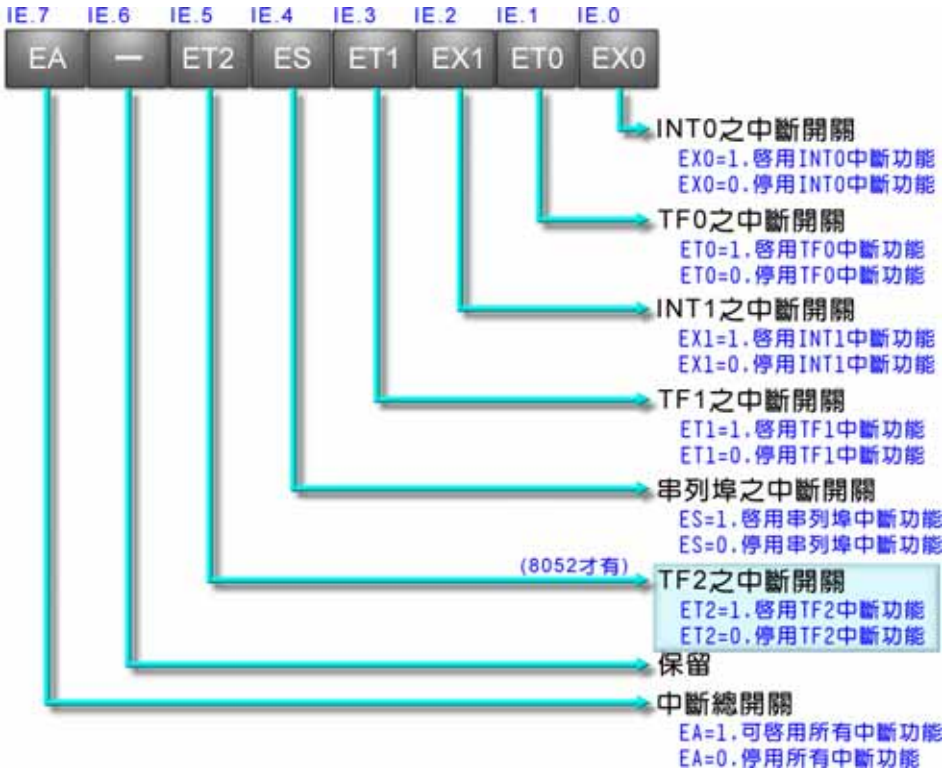


XXX 就是該中斷的服務程式的位址。所以，CPU 將執行 XXX 中斷服務程式，在中斷服務程式的最後一道指令爲「RETI」，執行這道指令，即可返回主程式，繼續執行剛才中斷時的下一個指令，如下圖所示：



（圖2） **中斷流程**

5-1-2 中斷致能暫存器



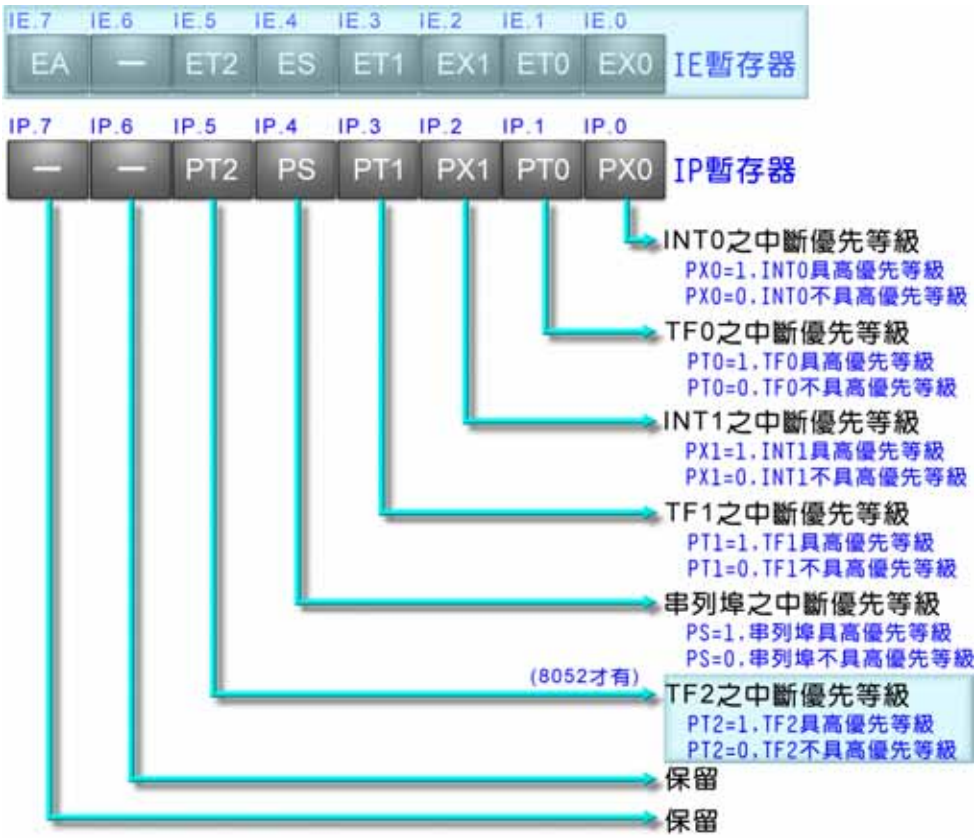
（圖3） **IE 暫存器**

如圖(1)所示，我們可將中斷致能暫存器(即 IE 暫存器)視爲啓閉中斷

功能的開關，實際上，IE 暫存器是一個 8 位元的可位元定址暫存器，其中各位元如圖(3)所示。

5-1-3 中斷優先等級暫存器

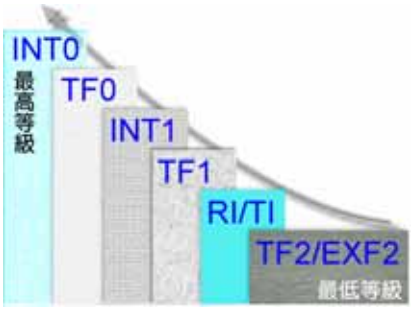
如圖(1)所示，中斷優先等級暫存器(IP 暫存器)判斷各中斷優先等級的開關，實際上，IP 暫存器是一個 8 位元的可位元定址暫存器，其中各位元如下圖所示：



（圖4） IP 暫存器

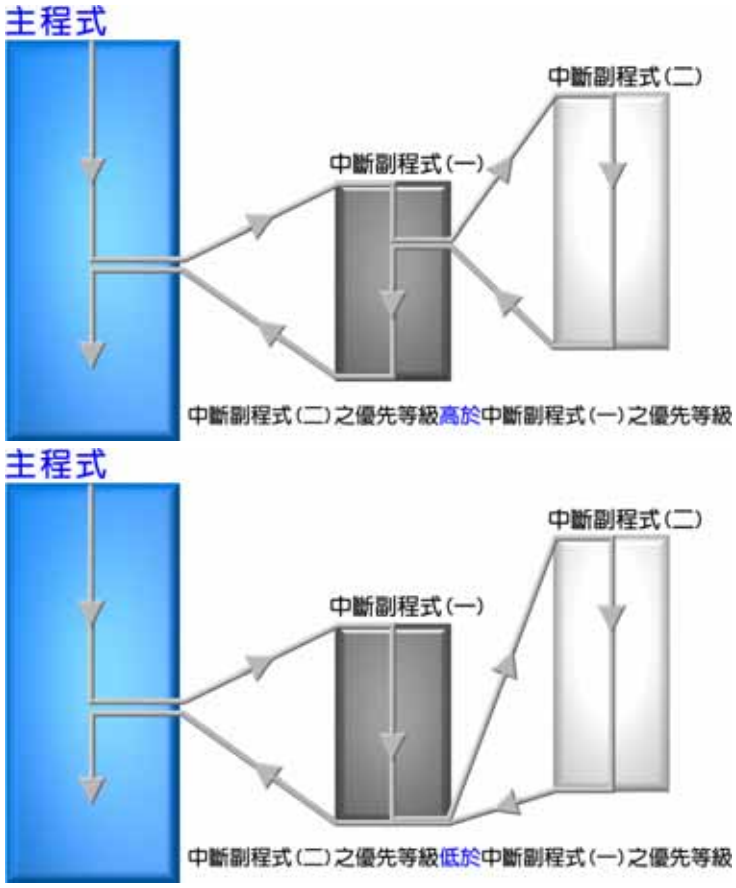
在此特將 IE 暫存器也放在上面，直接與 IP 暫存器比對，我們將可發現其中各位元幾乎是相對應的！所以，記住 IE 暫存器，也就記住 IP 暫存器了！

在圖(1)之中，很明顯地，IP 暫存器只是決定中斷屬「高優先等級」族群，還是「低優先等級」族群而已。原本各個中斷已有先後之分，其順序爲：



（圖5） **預置優先等級**

如上圖所示，若都沒有在 IP 暫存器裡設定優先等級，則中斷的優先等級爲「INT0>TF0>INT1>TF1>RI/TI>TF2/EXF2」；若將其中任一個中斷設爲高優先等級，例如讓 TF=1，則中斷的優先等級變爲「TF1>INT0>TF0 >INT1>RI/TI>TF2/EXF2」；若讓 TF=1、IE1=1，則中斷的優先等級變爲「INT1>TF1>IE0>TF0>RI/TI>TF2/EXF2」，以此類推。如下圖所示，分別是不同優先等級下，程式執行的流程：



（圖6） **不同優先等級下，程式執行的流程**

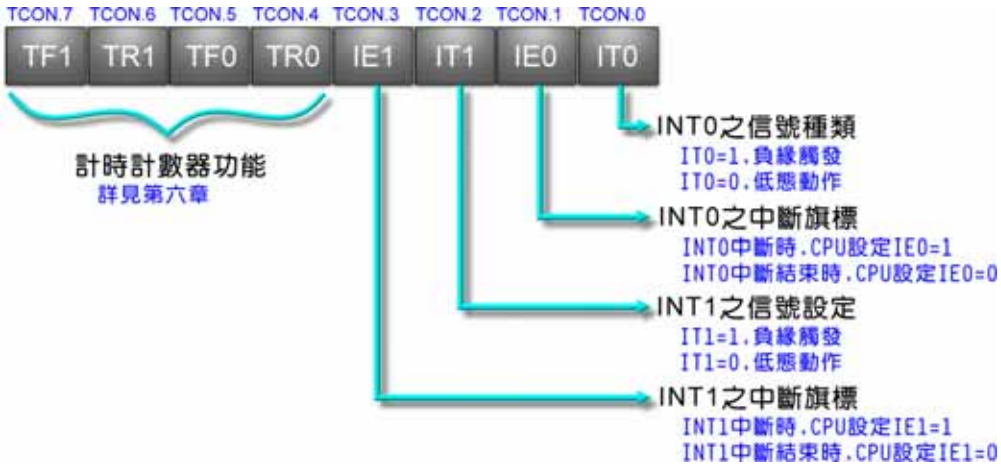
如上圖所示，簡單講，就是「優先等級較低者站一邊等」。如下兩個例子，正好說明「優先等級」這回事。

另類思考

老師上課中，教務主任要求張無忌同學到教務處說明科展的進度，於是張無忌就到教務處向教務主任報告科展的進度；不一會兒，校長又來要求張無忌同學到校長室說明技能競賽的事。於是，狗腿的值日生一健步，飛奔至教務處，通知張無忌立即到校長室。這時候，張無忌只好暫停向教務主任報告，而先去跟校長報告，報告完畢後，再回教務處繼續向教務主任報告。全部都報告完畢後，才回教室。

同樣的場景，老師上課中，校長要求張無忌同學到校長室說明技能競賽的事，於是張無忌就到校長室向校長報告技能競賽的事；不一會兒，教務主任又來要求張無忌同學到教務處說明科展的進度，只見老師冷冷地說「張無忌在校長室」，教務主任只好摸摸鼻子，回教務處等候。待張無忌結束校長室的報告後，再到教務處向教務主任報告。全部都報告完畢後，才回教室。

5-1-4 計時計數器控制暫存器



（圖7） TCON 暫存器

在計時計數器控制暫存器 TCON 裡，有部分設定與外部中斷信號的

取樣方式有關，如圖(7)所示，TCON 暫存器是一個 8 位元的可位元定址暫存器。其中 IT0 與 IT1 分別爲 INT0 與 INT1 的取樣信號設定位元，若要採用負緣觸發信號，則可將它設定爲 1；若要採用低態動作信號，則可將它設定爲 0。至於 IE0 與 IE1 兩個位元，則是由 CPU 所操作的中斷旗標，當中斷發生時，將被設定爲 1；結束中斷時，將恢復爲 0。

5-1-5 中斷向量

8051 的中斷向量，如下表所示：

中 斷 源	中 斷 向 量
INT0	03H
TF0	0BH
INT1	13H
TF1	1BH
RI/TI	23H
TF2/EXF2	2BH

上述中斷向量就是程式記憶體的位址，而 CPU 是從 00H 開始執行，是不是一開始就執行這些位址的指令呢？實際上，我們會在「ORG　0」之後，即執行「JMP　START」指令，而 START 標記是在所有中斷向量之後，即可避開這些中斷向量，如下所示：

```
                ORG     0               ;程式從 0 位址開始
                JMP     START           ;跳至 START
                ORG     03H             ;INT0 中斷向量
                JMP     XXX1            ;執行 XXX1 中斷副程式
                ORG     13H             ;INT1 中斷向量
                JMP     XXX2            ;執行 XXX2 中斷副程式
                :
                :
START:
                :
                :
```

INT0 的中斷向量爲 03H、TF0 的中斷向量爲 0BH，兩個中斷向量之間，只有 8 個位址的記憶體空間，很難寫出漂亮的中斷副程式，通常只在這個位址放置「JMP　XXX1」之類的指令，而 XXX1 才是真正的中斷副程式。

5-1-5 中斷的應用

中斷的應用包括三項措施，即中斷向量的設置、中斷的設定，以及中斷副程式的撰寫，如下說明：

設置中斷向量

MCS-51 的中斷向量分別是 03H、0BH、13H、1BH、23H 及 2BH，由於各中斷向量之間只有 8 個位址的記憶體空間，除非是中斷時，只是要提供簡單的服務，否則不會在這短短的 8 個記憶體位置之中，撰寫相關的服務程式，大都以「JMP　XXX1」指令，跳至特定的中斷副程式，然後在該中斷副程式之中，才提供真正的服務。中斷向量的設置是應需要而設，有使用到的中斷，才在其中斷向量設置所要中斷時操作的指令，而中斷向量設置需利用「ORG」虛擬指令來定位，例如「ORG　03H」指令代表其下一列指令將存放於程式記憶體的 03H 位置。

中斷設定

中斷的設定包括開啓中斷開關(即 IE 暫存器的設定)、中斷優先等級的設定(即 IE 暫存器的設定)、中斷信號的設定(即 TCON 暫存器的設定)、設定新的堆疊位置等。基本上，IE 暫存器、IP 暫存器及 TCON 暫存器的設定，可以利用 MOV 指令、SETB 指令或 CLR 指令，例如要開啓「總開關」、「INT0 開關」，則可以下列命令：

```
MOV     IE, #10000001B
```

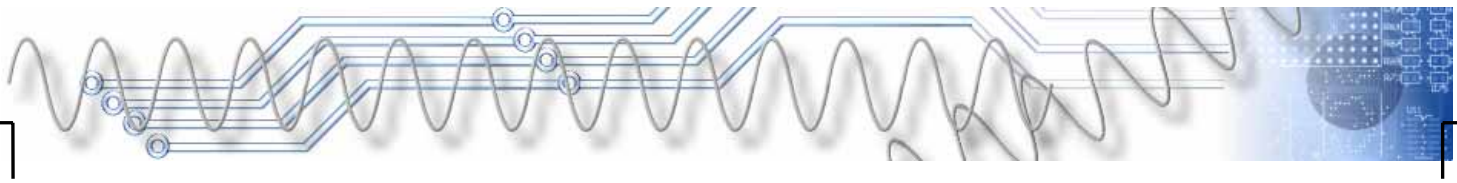
也可以使用下列命令：

```
SETB    IE.7
SETB    IE.0
```

若位置不清楚，則可直接指定「開關」的名稱，即：

```
SETB    EA
SETB    EX0
```

同理，IP 暫存器、TCON 暫存器的設定，也可以使用 MOV 指令或 SETB 指令，不過，使用 SETB 指令比較簡單，且具可讀性，例如要把 INT1 中斷的優先等級提高，則：



```
SETB    PX1
```

若要 INT1 中斷採負緣觸發的中斷信號，則：

```
SETB    IT1
```

至於堆疊的位置問題，程式預置堆疊指標指向 07H 位置，也就是從 08H 開始存放堆疊資料，而 08H 正是暫存器庫 1(即 RB1)的位置，爲避免衝突而破壞資料，因此，將堆疊指標改到其它地方，例如要把堆疊移到 30H，則：

```
MOV     SP, #30H
```

中斷副程式

「中斷副程式」就是一種副程式，而這種副程式與一般副程式之最大差異是中斷副程式的最後一道指令是「RETI」，一般副程式的最後一道指令是「RET」。

5-2 邏輯運算指令

邏輯運算指令的功能是將來源運算元的資料與目的運算元的資料，進行 AND、OR、NOT 等邏輯運算。邏輯運算指令包括 25 個指令，在此將它們分爲 7 大類來介紹：

AND 運算指令

AND 運算指令的功能是將來源運算元的資料與目的運算元的資料進行 AND 運算，而其結果存回目的運算元，如下圖所示：

其中的來源運算元可爲記憶體(RAM)位址 *direct* 的資料、暫存器 R*n* 的內容、以索引暫存器 R*i* 內容爲地址(@R*i*)的資料、立即值# *data* 或 ACC 等；目的運算元可爲 ACC 或記憶體(RAM)位址 *direct*。而 AND 運算的基本法則就是「**全部為 1，輸出為 1**」。「•」爲 AND 運算符號，若 A、B 爲輸入，Y 爲輸出，則 A 與 B 的 AND 運算可標記爲「Y=A•B」，

其真值表爲：

A	B	Y
0	0	0
0	1	0
1	0	0
1	1	1

- ANL　A, *direct*
  - **說明**：將 ACC 的內容與記憶體(RAM)位址(*direct*)的內容進行 AND 運算，而其結果存回 ACC，即 ACC • (*direct*)→ACC。
  - **組譯後大小**：2 bytes　　**執行時間**：12 個時鐘脈波
  - **範例**：

指令：ANL　A, 20H
若執行前：ACC=ABH、記憶體(20H)位址的內容爲 F0H
執行後：ACC=A0H、記憶體(20H)位址的內容爲 F0H

- ANL　A, R*n*
  - **說明**：將 ACC 的內容與暫存器 R*n* 的內容進行 AND 運算，而其結果存回 ACC，即 ACC • R*n* →ACC。
  - **組譯後大小**：1 byte　　**執行時間**：12 個時鐘脈波
  - **範例**：

指令：ANL　A, R7
若執行前：ACC=12H、R1=10101010B
執行後：ACC=02H、R1=10101010B

- ANL　A, @R*i*
  - **說明**：將 ACC 的內容與(R*i*)位址的內容進行 AND 運算，而其結果存回 ACC，即 ACC • (R*i*)→ACC。
  - **組譯後大小**：1 byte　　**執行時間**：12 個時鐘脈波
  - **範例**：

指令：ANL　A, @R0
若執行前：ACC=35H、R0=21H、記憶體(21H)位址的內容爲 A3H

執行後：ACC=21H、R0=21H、記憶體(21H)位址的內容爲 A3H

- ANL A, #*data*
  - **說明**：將 ACC 的內容與立即值 *data* 進行 AND 運算，而其結果存回 ACC，即 ACC • *data*→ACC。
  - **組譯後大小**：2 bytes **執行時間**：12 個時鐘脈波
  - **範例**：

指令：ANL A, #62H
若執行前：ACC=35H
執行後：ACC=20H

- ANL *direct*, A
  - **說明**：將 ACC 的內容與記憶體(RAM)位址(*direct*)的內容進行 AND 運算，而其結果存回(*direct*)記憶體位址，即 ACC • (*direct*)→ (*direct*)
  - **組譯後大小**：2 bytes **執行時間**：12 個時鐘脈波
  - **範例**：

指令：ANL 20H, A
若執行前：ACC=35H、記憶體(20H)位址的內容爲 22H
執行後：ACC=35H、記憶體(20H)位址的內容爲 20H

- ANL *direct*, #*data*
  - **說明**：將記憶體(RAM)位址(*direct*)的內容與立即值 *data* 進行 AND 運算，而其結果存回(*direct*)記憶體位址，即 (*direct*) • *data* → (*direct*)
  - **組譯後大小**：3 bytes **執行時間**：24 個時鐘脈波
  - **範例**：

指令：ANL 30H, #3EH
若執行前：記憶體(30H)位址的內容爲 27H
執行後：記憶體(30H)位址的內容爲 26H

OR 運算指令

OR 運算指令的功能是將來源運算元的資料與目的運算元的資料進

行 OR 運算，而其結果存回目的運算元，如下圖所示：

其中的來源運算元可爲記憶體(RAM)位址 *direct* 的資料、暫存器 R*n* 的內容、以索引暫存器 R*i* 內容爲地址(@R*i*)的資料、立即值# *data* 或 ACC 等；目的運算元可爲 ACC 或記憶體(RAM)位址 *direct*。而 OR 運算的基本法則就是「**全部為 0，輸出為 0**」。「+」爲 OR 運算符號，若 A、B 爲輸入，Y 爲輸出，則 A 與 B 的 OR 運算可標記爲「Y=A+B」，其真值表爲：

A	B	Y
0	0	0
0	1	1
1	0	1
1	1	1

- ORL　A, *direct*
  - **說明**：將 ACC 的內容與記憶體(RAM)位址(*direct*)的內容進行 OR 運算，而其結果存回 ACC，即 ACC + (*direct*)→ACC。
  - **組譯後大小**：2 bytes　　**執行時間**：12 個時鐘脈波
  - **範例**：

指令：ORL　A, 20H
若執行前：ACC=ABH、記憶體(20H)位址的內容爲 F0H
執行後：ACC=FBH、記憶體(20H)位址的內容爲 F0H

- ORL　A, R*n*
  - **說明**：將 ACC 的內容與暫存器 R*n* 的內容進行 OR 運算，而其結果存回 ACC，即 ACC + R*n* →ACC。
  - **組譯後大小**：1 byte　　**執行時間**：12 個時鐘脈波
  - **範例**：

指令：ORL　A, R7

若執行前：**ACC=12H、R1=10101010B**

執行後：**ACC=BAH、R1=10101010B**

- ORL A, @R*i*
  - **說明**：將 ACC 的內容與(R*i*)位址的內容進行 OR 運算，而其結果存回 ACC，即 ACC + (R*i*)→ACC。
  - **組譯後大小**：1 byte **執行時間**：12 個時鐘脈波
  - **範例**：

指令：ORL A, @R0

若執行前：ACC=35H、R0=21H、記憶體(21H)位址的內容爲 A3H

執行後：ACC=B7H、R0=21H、記憶體(21H)位址的內容爲 A3H

- ORL A, #*data*
  - **說明**：將 ACC 的內容與立即值 *data* 進行 OR 運算，而其結果存回 ACC，即 ACC + *data*→ACC。
  - **組譯後大小**：2 bytes **執行時間**：12 個時鐘脈波
  - **範例**：

指令：ORL A, #62H

若執行前：ACC=35H

執行後：ACC=77H

- ORL *direct*, A
  - **說明**：將 ACC 的內容與記憶體(RAM)位址(*direct*)的內容進行 OR 運算，而其結果存回(*direct*)記憶體位址，即 ACC + (*direct*)→ (*direct*)
  - **組譯後大小**：2 bytes **執行時間**：12 個時鐘脈波
  - **範例**：

指令：ORL 20H, A

若執行前：ACC=35H、記憶體(20H)位址的內容爲 22H

執行後：ACC=35H、記憶體(20H)位址的內容爲 37H

- ORL *direct*, #*data*
  - **說明**：將記憶體(RAM)位址(*direct*)的內容與立即值 *data* 進行 OR

運算，而其結果存回(*direct*)記憶體位址，即
(*direct*) + *data* → (*direct*)

組譯後大小：3 bytes　　**執行時間**：24 個時鐘脈波

範例：

指令：ORL　30H, #3EH
若執行前：記憶體(30H)位址的內容爲 27H
執行後：記憶體(30H)位址的內容爲 3FH

XOR 運算指令

XOR 運算指令的功能是將來源運算元的資料與目的運算元的資料進行 XOR(互斥或)運算，而其結果存回目的運算元，如下圖所示：

其中的來源運算元可爲記憶體(RAM)位址 *direct* 的資料、暫存器 R*n* 的內容、以索引暫存器 R*i* 內容爲地址(@R*i*)的資料、立即值# *data* 或 ACC 等；目的運算元可爲 ACC 或記憶體(RAM)位址 *direct*。而 XOR 運算的基本法則就是「**輸入相同、輸出為 0，輸入不相同、輸出為 1**」。「⊕」爲 OR 運算符號，若 A、B 爲輸入，Y 爲輸出，則 A 與 B 的 XOR 運算可標記爲「Y=A⊕B」，其真值表爲：

A	B	Y
0	0	0
0	1	1
1	0	1
1	1	0

- XRL　A, *direct*

說明：將 ACC 的內容與記憶體(RAM)位址(*direct*)的內容進行 XOR(互斥或)運算，而其結果存回 ACC，即 ACC ⊕ (*direct*)→ACC

組譯後大小：2 bytes　　**執行時間**：12 個時鐘脈波

範例：

指令：XRL　A, 20H

若執行前：**ACC=ABH**、記憶體**(20H)**位址的內容爲 **F0H**
執行後：**ACC=5BH**、記憶體**(20H)**位址的內容爲 **F0H**

- XRL A, R*n*
  - **說明**：將 ACC 的內容與暫存器 R*n* 的內容進行 XOR(互斥或)運算，而其結果存回 ACC，即 ACC ⊕ R*n* →ACC。
  - **組譯後大小**：1 byte **執行時間**：12 個時鐘脈波
  - **範例**：

指令：XRL A, R7
若執行前：**ACC=12H**、**R1=10101010B**
執行後：**ACC=B8H**、**R1=10101010B**

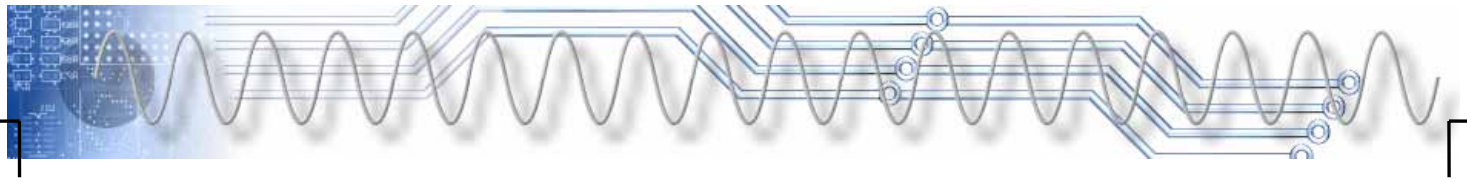
- XRL A, @R*i*
  - **說明**：將 ACC 的內容與(R*i*)位址的內容進行 XOR(互斥或)運算，而其結果存回 ACC，即 ACC ⊕ (R*i*)→ACC。
  - **組譯後大小**：1 byte **執行時間**：12 個時鐘脈波
  - **範例**：

指令：XRL A, @R0
若執行前：**ACC=35H**、**R0=21H**、記憶體**(21H)**位址的內容爲 **A3H**
執行後：**ACC=96H**、**R0=21H**、記憶體**(21H)**位址的內容爲 **A3H**

- XRL A, #*data*
  - **說明**：將 ACC 的內容與立即值 *data* 進行 XOR(互斥或)運算，而其結果存回 ACC，即 ACC ⊕ *data*→ACC。
  - **組譯後大小**：2 bytes **執行時間**：12 個時鐘脈波
  - **範例**：

指令：XRL A, #62H
若執行前：**ACC=35H**
執行後：**ACC=57H**

- XRL *direct*, A
  - **說明**：將 ACC 的內容與記憶體(RAM)位址(*direct*)的內容進行



XOR(互斥或)運算，而其結果存回(*direct*)記憶體位址，即
ACC ⊕ (*direct*)→ (*direct*)

組譯後大小：2 bytes　　**執行時間**：12 個時鐘脈波

範例：

指令：XRL　20H, A
若執行前：ACC=35H、記憶體(20H)位址的內容爲 22H
執行後：ACC=35H、記憶體(20H)位址的內容爲 17H

- XRL　*direct*, #*data*

說明：將記憶體(RAM)位址(*direct*)的內容與立即值 *data* 進行 XOR(互斥或)運算，而其結果存回(*direct*)記憶體位址，即
(*direct*) ⊕ *data* → (*direct*)

組譯後大小：3 bytes　　**執行時間**：24 個時鐘脈波

範例：

指令：XRL　30H, #3EH
若執行前：記憶體(30H)位址的內容爲 27H
執行後：記憶體(30H)位址的內容爲 19H

NOT 運算指令

NOT 運算指令的功能是將運算元反相，也就是取補數的意思。

- CPL　A

說明：將 ACC 的內容反相，也就是取補數，即
ACC → ACC = $\overline{ACC}$

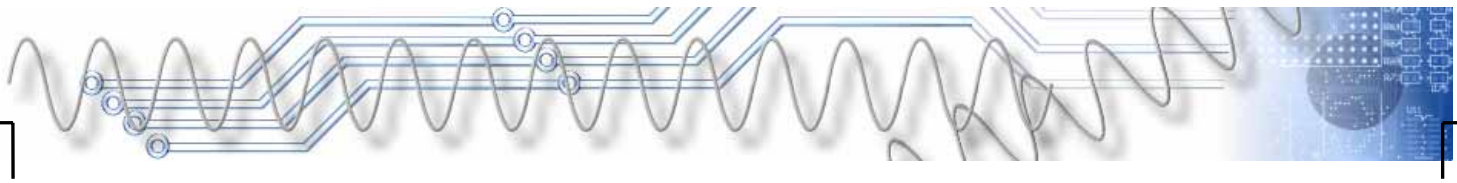
組譯後大小：1 byte　　**執行時間**：12 個時鐘脈波

範例：

指令：CPL　A
若執行前：ACC=10110100B
執行後：ACC=01001011B

清除指令

8051 只提供對 ACC 的清除指令，也就是讓 ACC 的內容變爲 0。



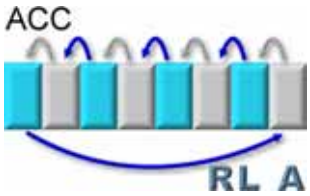
- CLR A
  - **說明**：將 ACC 的內容變成 00H，即 00H → ACC。
  - **組譯後大小**：2 bytes **執行時間**：12 個時鐘脈波
  - **範例**：

指令：CLR A
若執行前：ACC=35H
執行後：ACC=00H

旋轉指令

旋轉指令的功能是將 ACC 內的每個位元左移或右移一個位元，此外，參與移位的也可以包含進位位元 CY。

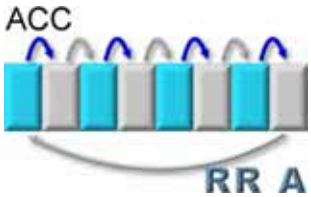
- RL A
  - **說明**：將 ACC 裡的每一位元左移一個位元，即



  - **組譯後大小**：1 byte **執行時間**：12 個時鐘脈波
  - **範例**：

指令：RL A
若執行前：ACC=10110100B
執行後：ACC=01101001B

- RR A
  - **說明**：將 ACC 裡的每一位元右移一個位元，即



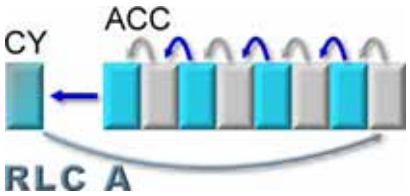
  - **組譯後大小**：1 byte **執行時間**：12 個時鐘脈波
  - **範例**：

指令：**RR　A**

若執行前：**ACC=10110100B**

執行後：**ACC=01011010B**

- RLC　A
  - **說明**：將 ACC 裡的每一位元左移一個位元，最左邊位元移入 CY，CY 移入最右邊位元，即



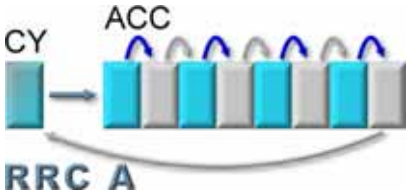
  - **組譯後大小**：1 byte　　**執行時間**：12 個時鐘脈波
  - **範例**：

指令：**RLC　A**

若執行前：**ACC=10110100B、CY=0**

執行後：**ACC=01101000B、CY=1**

- RRC　A
  - **說明**：將 ACC 裡的每一位元右移一個位元，最右邊位元移入 CY，CY 移入最左邊位元，即



  - **組譯後大小**：1 byte　　**執行時間**：12 個時鐘脈波
  - **範例**：

指令：**RRC　A**

若執行前：**ACC=10110100B、CY=1**

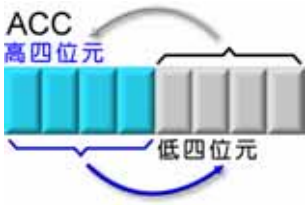
執行後：**ACC=11011010B、CY=0**

互換指令

互換指令的功能是將 ACC 的內容分爲低四位元與高四位元，而將這

兩部分的內容互換位置。

- SWAP A
  - **說明**：，即



  - **組譯後大小**：1 byte **執行時間**：12 個時鐘脈波
  - **範例**：

指令：SWAP A
若執行前：ACC=CDH
執行後：ACC=DCH

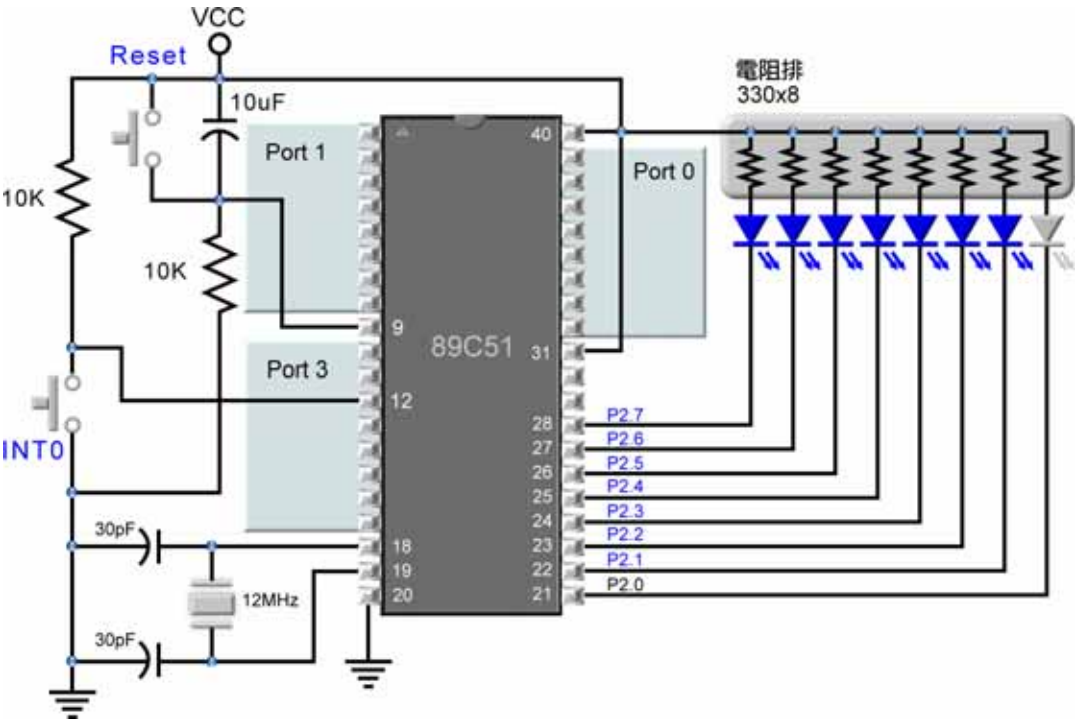
5-3 實例演練

在本單元裡提供四個範例，以展示 8051 的外部中斷功能。

5-3-1 外部中斷 INT0 實例演練

- **功能說明**

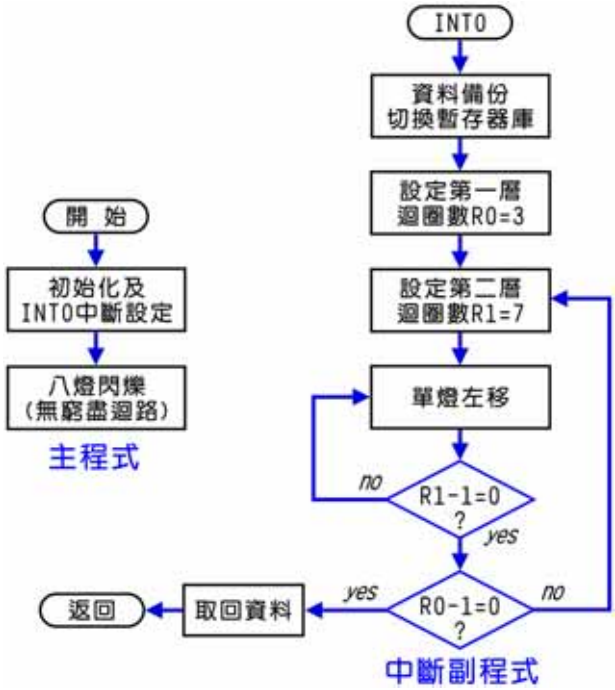
如圖 8 所示，P2 連接 8 個 LED、第 12 腳(即 INT0 接腳)連接一個 10K 歐姆的提升電阻，讓該接腳保持爲 High，另外再連接一個按鈕開關(INT0)。當主程式正常執行時，P2 所連接的八個 LED 將閃爍。若按 INT0 按鈕開關，則進入中斷狀態，P2 所連接的八個 LED 將變成單燈左移，而左移 3 圈(從最左邊到最右邊爲 1 圈)後，恢復中斷前的狀態，程式將繼續執行八燈閃爍的功能。



（圖8）　**電路圖**

參考程式

依功能需求與電路結構得知，首先必須設定中斷向量，而 INT0 的中斷向量爲 03H，在此希望中斷後，即執行單燈左移的中斷副程式(在此定名爲 INT0 中斷副程式)。在宣告中斷向量之後，立即設定中斷，包括打開中斷的總開關(EA)及 INT0 開關(EX0)，另外也把堆疊指標移至安全的位置(30H)。主程式的爲簡單的八燈閃爍功能，中斷副程式裡，一開始先把主程式的資料存入堆疊，包括程式狀態字組暫存器(PSW)及 ACC，然後將暫存器庫切換到 RB1，以避免影響到返回主程式後的結果。至於三圈單燈左移的功能，則由兩個暫存器(R0、R1)充當計次迴圈的計數器，最左邊移到最右邊須移動 7 次，在此以 R1 做爲其計數器，而這樣的動作總共要做 3 次，所以利用 R0 來計次，以構成巢狀迴圈。




```
                ORG     0               ;程式從 0 位址開始
                JMP     START           ;跳過中斷向量
                ORG     03H             ;INT0 中斷向量
                JMP     INT0            ;執行 INT0 中斷副程式
;=============================================================
START:          MOV     IE, #10000001B  ;打開總開關與 EX0 分路開關
                MOV     SP, #30H        ;設定堆疊位置
                SETB    IT0             ;採用負緣觸發信號
                MOV     A, #0           ;將 ACC 設定為 00000000B
LOOP:           MOV     P2, A           ;輸出到 LED
                CALL    DELAY           ;呼叫延遲副程式
                CPL     A               ;將 A 的內容反相
                JMP     LOOP            ;跳至 LOOP 形成一個迴圈
;
;=============INT 0 中斷副程式===開始=======================
INT0:           PUSH    PSW             ;將 PSW 存入堆疊
                PUSH    A               ;將 ACC 存入堆疊
                SETB    RS0             ;切換到 RB1
;===第一層迴圈開始
                MOV     R0, #3          ;設定三次迴圈
INT_LOOP0:      MOV     A, #FEH         ;單燈左移初始值
                MOV     R1, #7          ;設定七次左移
;========第二層迴圈開始
INT_LOOP1:      MOV     P2, A           ;輸出到 LED
                CALL    DELAY           ;呼叫延遲副程式
                RL      A               ;將 A 的內容左移
                DJNZ    R1, INT_LOOP1   ;跳至 INT_LOOP0 形成一個迴圈
```

```
;========第二層迴圈結束
                DJNZ    R0, INT_LOOP0   ;跳至 INT_LOOP1 形成一個迴圈
;===第一層迴圈結束
                POP     A               ;取回 ACC 內容
                POP     PSW             ;取回 PSW 內容
                RETI                    ;返回主程式
;=============INT 0 中斷副程式====結束=====================
;
;=============0.1 秒 DELAY 副程式===開始====================
DELAY:          MOV     R7, #200
D1:             MOV     R6, #250
                DJNZ    R6, $
                DJNZ    R7, D1
                RET
;============= 0.1 秒 DELAY 副程式====結束==================
                END
```

INT0 中斷實驗(ch5-1.asm)

操作

1. 依功能需求與電路結構撰寫程式，然後將該程式組譯與連結，以產生*.HEX 檔。
2. 請利用 AVSIM51 之類的模擬軟體，模擬其功能(先將延遲時間縮短，大概是 R7=10、R6=10)。若有非預期的狀況，則檢視原始程式，看看哪裡出問題？並將它記錄在實驗報告裡。
3. 將延遲時間恢復正常，重新組譯連結，再按圖 8 連接線路，使用實體模擬器，載入新的程式(*.HEX)，以模擬該電路的動作。若有非預期的狀況，則檢視線路的連接狀況，看看哪裡出問題？並將它記錄在實驗報告裡。
4. 若實體模擬功能正常，請將程式燒錄到 89C51，再把該 89C51 放入實體電路，以取代剛才的實體模擬器，然後直接送電，看看是否正常？
5. 撰寫實驗報告。

思考一下

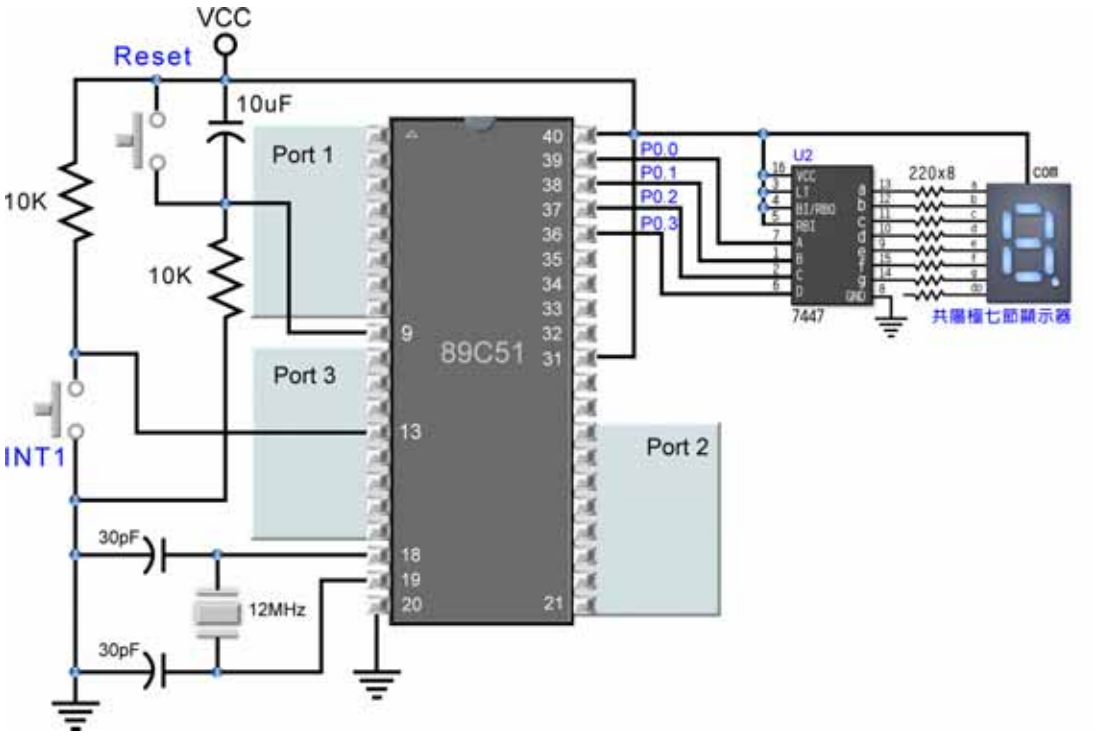
1. 在本實驗裡，中斷副程式是以巢狀迴圈方式，達到巢狀單燈左移三圈的目的；若不要以巢狀迴圈方式，請試著以別種方式，達到同樣的效果？

2. 在本實驗裡，若希望中斷時，這八個 LED 變成是霹靂燈，來回各三圈，才返回主程式，程式應如何更改？

5-3-2 外部中斷 INT1 實例演練

功能說明

如圖 9 所示，P0 之低四位元連接 7447，再驅動共陽極七節顯示器，所以只要在 P0.0 到 P0.3 輸出 BCD 碼，即可在七節顯示器上顯示該數字，而第 13 腳(即 INT1 接腳)連接一個 10K 歐姆的提升電阻，讓該接腳保持爲 High，另外再連接一個按鈕開關(INT1)。當主程式正常執行時，七節顯示器將從 0 開始正數到 9(循環)，每 0.5 秒增加 1。若按 INT1 按鈕開關，則進入中斷狀態，則七節顯示器將從 9 開始倒數到 0(一圈後結束中斷)，每 0.5 秒減少 1。

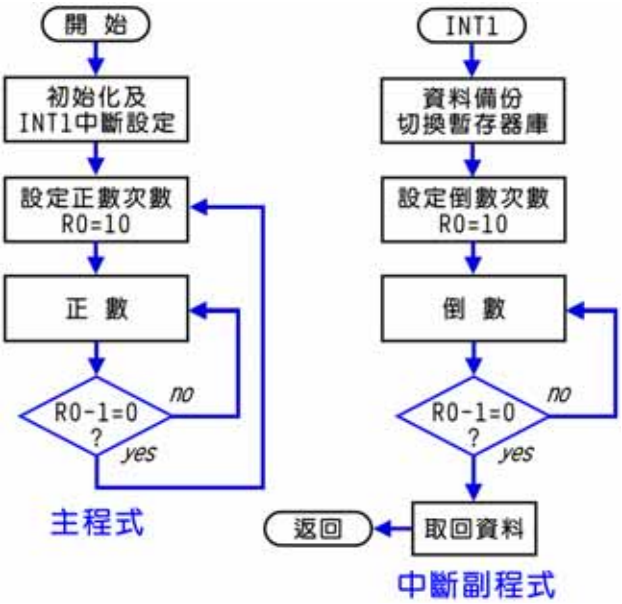


（圖9） **電路圖**

參考程式

依功能需求與電路結構得知，首先設定中斷向量，而 INT1 的中斷向量爲 13H，在此希望中斷後，即執行倒數的中斷副程式(在此定名

爲 INT1 中斷副程式)。在宣告中斷向量之後，立即設定中斷，包括打開中斷的總開關(EA)及 INT1 開關(EX1)，另外也把堆疊指標移至安全的位置(30H)。主程式的爲簡單的正數功能，而在中斷副程式裡，一開始先把主程式的資料存入堆疊，包括程式狀態字組暫存器(PSW)及 ACC，然後將暫存器庫切換到 RB1，以避免影響到返回主程式後的結果。至於倒數部分的功能，則可複製主程式的正數部分程式，再將其初始值改爲 9，INC 指令改爲 DEC 指令即可。




```
                ORG     0               ;程式從 0 位址開始
                JMP     START           ;跳過中斷向量
                ORG     13H             ;INT1 中斷向量
                JMP     INT1            ;執行 INT1 中斷副程式
;=============================================================
START:          MOV     IE, #10000100B ;打開總開關與 EX1 分路開關
                MOV     SP, #30H        ;設定堆疊位置
                SETB    IT1             ;採用負緣觸發信號
RENEW:          MOV     R0, #10         ;R0 設定為 10，即輸出次數
                MOV     A, #0           ;設定輸出初始值
LOOP:           MOV     P0, A           ;輸出到七節顯示器
                CALL    DELAY           ;呼叫延遲副程式
                INC     A               ;將 A+1
                DJNZ    R0, LOOP        ;跳至 LOOP 形成一個迴圈
                JMP     RENEW           ;重新開始
;
;=============INT 1 中斷副程式===開始=======================
INT1:           PUSH    PSW             ;將 PSW 存入堆疊
```

```
                PUSH    A               ;將 ACC 存入堆疊
                SETB    RS0             ;切換到 RB1
RENEW:          MOV     R0, #10         ;R0 設定為 10，即輸出次數
                MOV     A, #9           ;設定輸出初始值
INT_LOOP:       MOV     P0, A           ;輸出到七節顯示器
                CALL    DELAY           ;呼叫延遲副程式
                DEC     A               ;將 A-1
                DJNZ    R0, INT_LOOP    ;跳至 INT_LOOP 形成一個迴圈
                POP     A               ;取回 ACC 內容
                POP     PSW             ;取回 PSW 內容
                RETI                    ;返回主程式
;=============INT 0 中斷副程式====結束=====================
;=============0.5 秒 DELAY 副程式===開始====================
DELAY:          MOV     R7, #5
D1:             MOV     R6, #200
D2:             MOV     R5, #250
                DJNZ    R5, $
                DJNZ      R6, D2
                DJNZ    R7, D1
                RET
;============= 0.5 秒 DELAY 副程式====結束==================
                END
```

INT1 中斷實驗(ch5-2.asm)

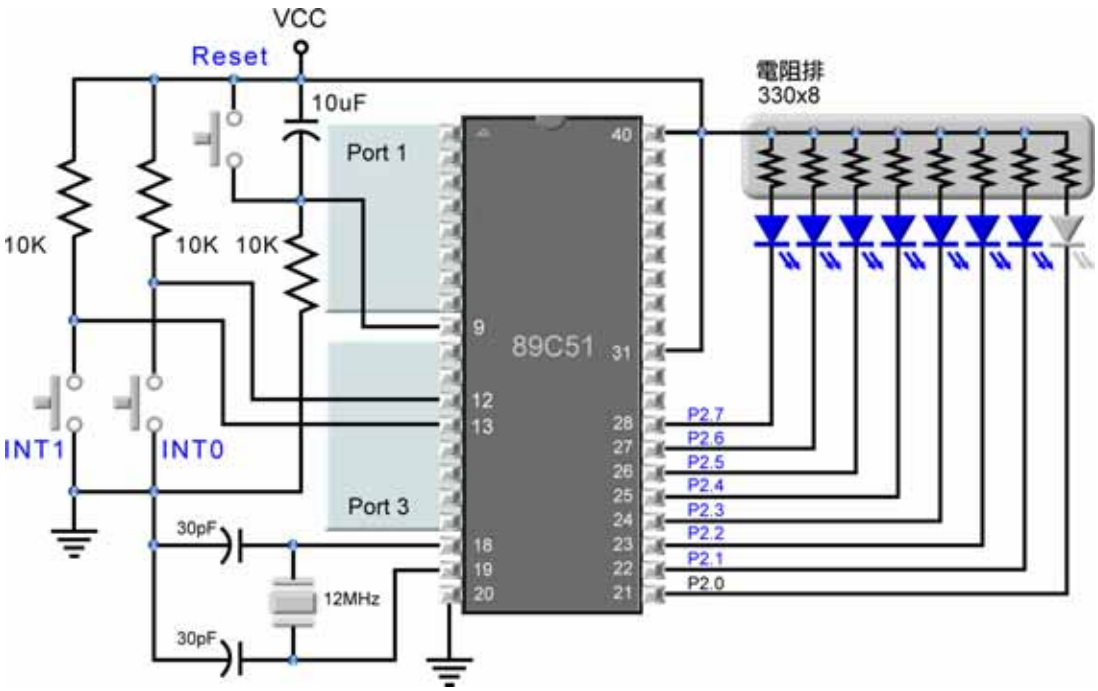
操作

1. 依功能需求與電路結構撰寫程式，然後將該程式組譯與連結，以產生*.HEX 檔。
2. 請利用 AVSIM51 之類的模擬軟體，模擬其功能(先將延遲時間縮短，大概是 R7=10、R6=10)。若有非預期的狀況，則檢視原始程式，看看哪裡出問題？並將它記錄在實驗報告裡。
3. 將延遲時間恢復正常，重新組譯連結，再按圖 9 連接線路，使用實體模擬器，載入新的程式(*.HEX)，以模擬該電路的動作。若有非預期的狀況，則檢視線路的連接狀況，看看哪裡出問題？並將它記錄在實驗報告裡。
4. 若實體模擬功能正常，請將程式燒錄到 89C51，再把該 89C51 放入實體電路，以取代剛才的實體模擬器，然後直接送電，看看是否正常？
5. 撰寫實驗報告。

思考一下

1. 在本實驗裡，使用 7447 做爲 BCD 對七節顯器碼的解碼器，若不使用 7447，程式與電路應如何更改？
2. 若 INT1 採用低態動作信號(即將「SETB　IT1」改爲「CLR　IT1」)，軟體模擬時，有何影響？硬體模擬時，有何影響？

5-3-3 兩個外部中斷實例演練



（圖10）**電路圖**

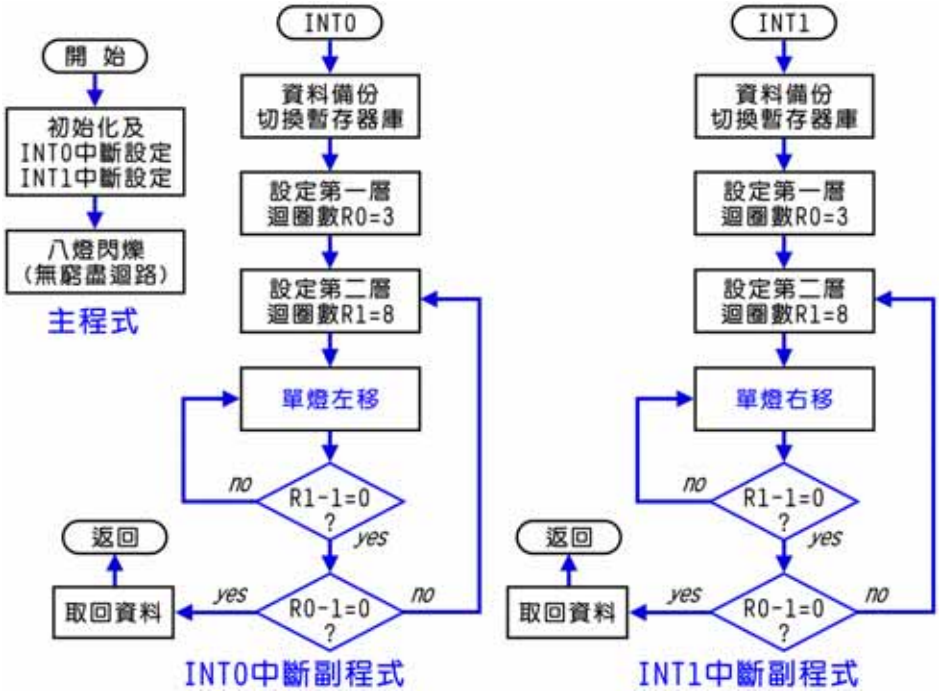
功能說明

如圖 10 所示，P2 連接 8 個 LED、第 12 腳(即 INT0 接腳)連接一個 10K 歐姆的提升電阻，讓該接腳保持爲 High，另外再連接一個按鈕開關(INT0)、第 13 腳(即 INT1 接腳)連接一個 10K 歐姆的提升電阻，讓該接腳保持爲 High，另外再連接一個按鈕開關(INT1)。當主程式正常執行時，P2 所連接的八個 LED 將閃爍。若按 INT0 按鈕開關，則進入 INT0 中斷狀態，P2 所連接的八個 LED 將變成單燈左移，而左移 3 圈(從最左邊到最右邊爲 1 圈)後，恢復中斷前的狀態，程式將繼續執行八燈閃爍的功能。若按 INT1 按鈕開關，則進入 INT1 中斷狀態，P2 所連接的八個 LED 將變成單燈右移，而左移 3 圈(從最

左邊到最右邊爲 1 圈)後，恢復中斷前的狀態，程式將繼續執行八燈閃爍的功能。另外，在此要求單燈左移(INT0)中斷的優先等級較單燈右移(INT1)中斷的優先等級高。

參考程式

依功能需求與電路結構得知，首先必須設定中斷向量，而 INT0 的中斷向量爲 03H，在此希望中斷後，即執行單燈左移的中斷副程式(在此定名爲 INT0 中斷副程式)；INT1 的中斷向量爲 13H，在此希望中斷後，即執行單燈右移的中斷副程式(在此定名爲 INT1 中斷副程式)。在宣告中斷向量之後，立即設定中斷，包括打開中斷的總開關(EA)、INT0 開關(EX0)及 INT1 開關(EX1)，另外也把堆疊指標移至安全的位置(30H)。而由於本實驗要求 INT0 中斷的優先等級高於 INT1 中斷的優先等級，與 8051 的預置狀態相同，所以不要額外的設定。基本上，主程式及單燈左移中斷副程式，與 5-3-1 節相同，在此只要增加單燈右移中斷副程式即可。



```
                ORG     0               ;程式從 0 位址開始
                JMP     START           ;跳過中斷向量
                ORG     03H             ;INT0 中斷向量
                JMP     INT0            ;執行 INT0 中斷副程式
                ORG     13H             ;INT1 中斷向量
                JMP     INT1            ;執行 INT1 中斷副程式
;=============================================================
START:          MOV     IE, #10000101B ;打開總開關、EX0 與 EX1 開關
                MOV     SP, #30H        ;設定堆疊位置
                SETB    IT0             ;採用負緣觸發信號
                SETB    IT1             ;採用負緣觸發信號
                MOV     A, #0           ;將 ACC 設定為 00000000B
LOOP:           MOV     P2, A           ;輸出到 LED
                CALL    DELAY           ;呼叫延遲副程式
                CPL     A               ;將 A 的內容反相
                JMP     LOOP            ;跳至 LOOP 形成一個迴圈
;
;=============INT 0 中斷副程式===開始=======================
INT0:           PUSH    PSW             ;將 PSW 存入堆疊
                PUSH    A               ;將 ACC 存入堆疊
                SETB    RS0             ;切換到 RB1
;===第一層迴圈開始
                MOV     R0, #3          ;設定三次迴圈
INT0_LOOP0:     MOV     A, #FEH         ;單燈左移初始值
                MOV     R1, #8          ;設定八次左移
;========第二層迴圈開始
INT0_LOOP1:     MOV     P2, A           ;輸出到 LED
                CALL    DELAY           ;呼叫延遲副程式
                RL      A               ;將 A 的內容左移
                DJNZ    R1, INT0_LOOP1 ;跳至 INT0_LOOP0 形成一個迴圈
;========第二層迴圈結束
                DJNZ    R0, INT0_LOOP0 ;跳至 INT0_LOOP1 形成一個迴圈
;===第一層迴圈結束
                POP     A               ;取回 ACC 內容
                POP     PSW             ;取回 PSW 內容
                RETI                    ;返回主程式
;=============INT 0 中斷副程式====結束=====================
;
;=============INT 1 中斷副程式===開始=======================
INT1:           PUSH    PSW             ;將 PSW 存入堆疊
                PUSH    A               ;將 ACC 存入堆疊
                CLR     RS0             ;
                SETB    RS1             ;切換到 RB2
;===第一層迴圈開始
                MOV     R0, #3          ;設定三次迴圈
INT1_LOOP0:     MOV     A, #7FH         ;單燈右移初始值
                MOV     R1, #8          ;設定八次左移
;========第二層迴圈開始
```

```
INT1_LOOP1:   MOV    P2, A            ;輸出到 LED
              CALL   DELAY            ;呼叫延遲副程式
              RR     A                ;將 A 的內容右移
              DJNZ   R1, INT1_LOOP1 ;跳至 INT1_LOOP0 形成一個迴圈
;========第二層迴圈結束
              DJNZ   R0, INT1_LOOP0 ;跳至 INT1_LOOP1 形成一個迴圈
;===第一層迴圈結束
              POP    A                ;取回 ACC 內容
              POP    PSW              ;取回 PSW 內容
              RETI                    ;返回主程式
;=============INT 0 中斷副程式====結束=====================
;
;=============0.1 秒 DELAY 副程式===開始====================
DELAY:        MOV    R7, #200
D1:           MOV    R6, #250
              DJNZ   R6, $
              DJNZ   R7, D1
              RET
;============= 0.1 秒 DELAY 副程式====結束==================
              END
```

兩個外部中斷實驗(ch5-3.asm)

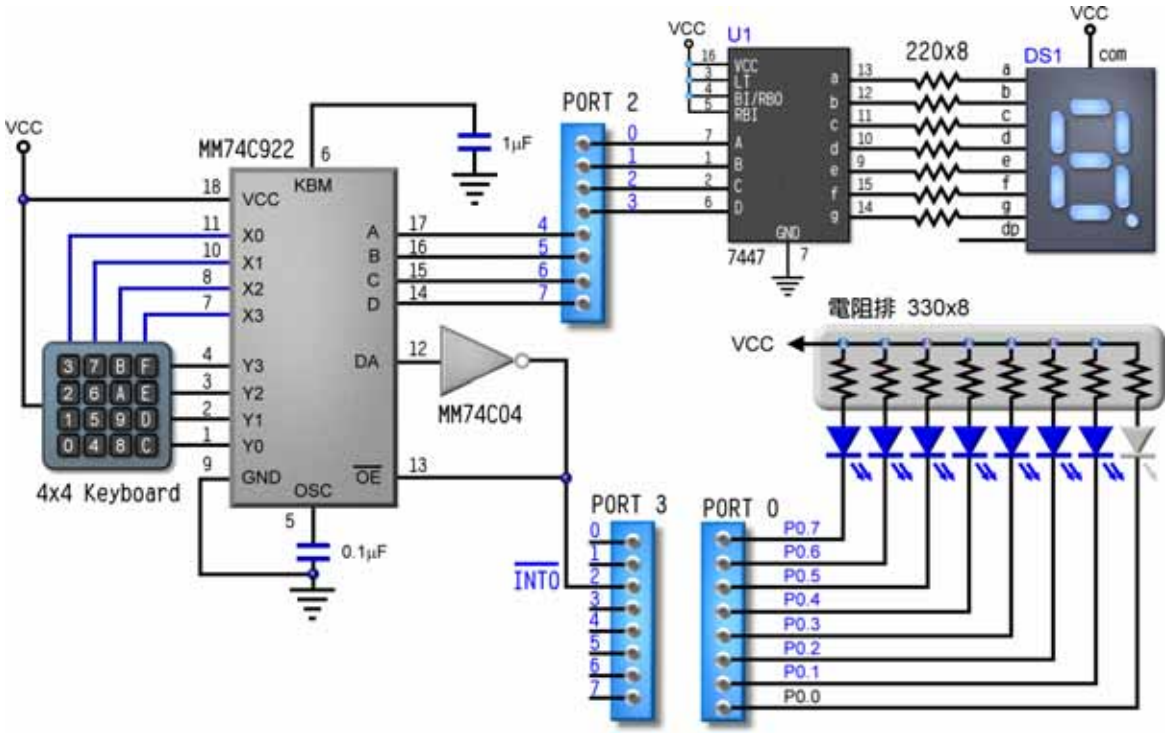
操作

1. 依功能需求與電路結構撰寫程式，然後將該程式組譯與連結，以產生*.HEX 檔。
2. 請利用 AVSIM51 之類的模擬軟體，模擬其功能(先將延遲時間縮短，大概是 R7=10、R6=10)。若有非預期的狀況，則檢視原始程式，看看哪裡出問題？並將它記錄在實驗報告裡。
3. 將延遲時間恢復正常，重新組譯連結，再按圖 10 連接線路，使用實體模擬器，載入新的程式(*.HEX)，以模擬該電路的動作。若有非預期的狀況，則檢視線路的連接狀況，看看哪裡出問題？並將它記錄在實驗報告裡。
4. 若實體模擬功能正常，請將程式燒錄到 89C51，再把該 89C51 放入實體電路，以取代剛才的實體模擬器，然後直接送電，看看是否正常？
5. 撰寫實驗報告。

思考一下

1. 利用 AVSIM51 進行模擬時，當 INT1 中斷中，再按 INT0 按鈕時，是否能正確模擬出 INT0 中斷具有較高優先等級？
2. 在本實驗裡，若希望 INT1 中斷的優先等級高於程式 INT0 中斷的優先等級，應如何修改？

5-3-4 鍵盤中斷實例演練



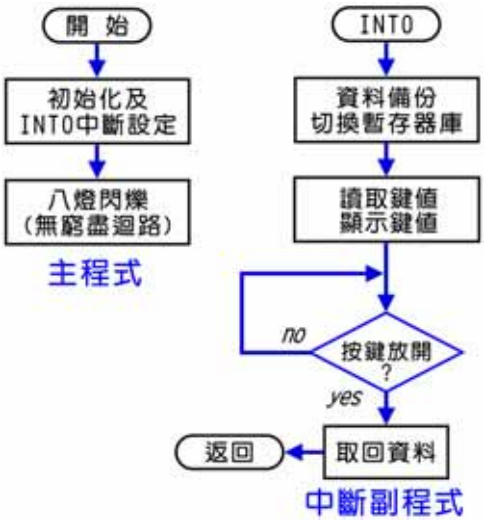
（圖11）**電路圖**

功能說明

如圖 11 所示，MM74C922 的資料 ABCD 連接到 8051 的 P2.4 到 P2.7，而 P2.0 到 P2.3 連接 7447，以輸出數字資料。MM74C922 的 DA 接腳經過反閘(MM74C04)連接到 8051 的 P3.2，也就是 INT0 接腳。另外，8051 的 PORT 0 連接 8 個 LED。當主程式正常執行時，8 個 LED 閃爍(每 0.1 秒切換一次)，若按下 4X4 鍵盤上的任一個鍵，則該鍵的數字將顯示在七節顯示器上，而 PORT 0 所連接的 LED，仍保持正常的閃爍。

參考程式

依功能需求與電路結構得知，首先必須設定中斷向量，而 INT0 的中斷向量爲 03H，在此希望中斷後，即執行讀取鍵盤資料，並將讀取到的資料，送到七節顯示器上，這個動作將由名爲 KEYBOARD 中斷副程式來執行。在宣告中斷向量之後，立即設定中斷，包括打開中斷的總開關(EA)及 INT0 開關(EX0)，同樣地，把堆疊指標移至安全的位置(30H)。而主程式爲八燈閃爍，在前面幾個實驗中已介紹過了，在此不贅述。




```
                ORG     0               ;程式從 0 位址開始
                JMP     START           ;跳過中斷向量
                ORG     03H             ;INT0 中斷向量
                JMP     KEYBOARD        ;執行 KEYBOARD 中斷副程式
START:          MOV     IE, #10000001B  ;打開總開關、EX0 與 EX1 開關
                MOV     SP, #30H        ;設定堆疊位置
                CLR     IT0             ;採用低態動作信號
                MOV     P2, #FFH        ;關閉七節顯示器，
                                        ;並將 P2.4 至 P2.7 設定為輸入模式
                MOV     A, #0           ;八燈初始值
LOOP:           MOV     P0, A           ;輸出到 LED
                CALL    DELAY           ;呼叫延遲副程式
                CPL     A               ;將 A 的內容反相
                JMP     LOOP            ;跳至 LOOP 形成一個迴圈
;============= KEYBOARD 中斷副程式===開始==================
KEYBOARD:       PUSH    PSW             ;將 PSW 存入堆疊
                PUSH    A               ;將 ACC 存入堆疊
                SETB    RS0             ;切換到 RB1
                MOV     P2, #FFH        ;關閉七節顯示器，
                                        ;並將 P2.4 至 P2.7 設定為輸入模式
                MOV     A, P2           ;讀入鍵盤資料
```

```
                SWAP    A               ;將 ACC 的高四位元與低四位元互換
                ORL     A, #F0H         ;遮蓋高四位元
                MOV     P2, A           ;顯示鍵盤資料
RELEASE:        JB      P3.2, RELEASE   ;判讀是否放開按鍵
                POP     A               ;取回 ACC 內容
                POP     PSW             ;取回 PSW 內容
                RETI                    ;返回主程式
;=============0.1 秒 DELAY 副程式===開始====================
DELAY:          MOV     R7, #200
D1:             MOV     R6, #250
                DJNZ    R6, $
                DJNZ    R7, D1
                RET
                END
```

4X4 鍵盤實驗(ch5-4.asm)

操作

1. 依功能需求與電路結構撰寫程式，然後將該程式組譯與連結，以產生*.HEX 檔。
2. 請按圖 11 連接線路，再使用實體模擬器，載入該程式(*.HEX)，以模擬該電路的動作。若有非預期的狀況，則檢視線路的連接狀況，看看哪裡出問題？並將它記錄在實驗報告裡。
3. 若實體模擬功能正常，請將程式燒錄到 89C51，再把該 89C51 放入實體電路，以取代剛才的實體模擬器，然後直接送電，看看是否正常？
4. 撰寫實驗報告。

思考一下

1. 在本實驗裡，有沒有「彈跳」的困擾？
2. 在本實驗裡，若按住按鍵不放，會怎樣？

5-4 即時練習

在本章裡探討 8051 的中斷，包括中斷向量、與中斷相關的暫存器、中斷程式的撰寫等，以及邏輯運算指令，可說是微電腦系統中不可或缺的部分。在此請試著回答下列問題，以確認對於此部分的認識程度。

1. 試說明 IE 暫存器及 IP 暫存器中，各位元功能？
2. 試說明 MCS-51 提供哪些中斷？中斷向量爲何？
3. 試說明如何設定 IE 暫存器？
4. 具有中斷功能的程式，必須包含那些宣告或設定？
5. 如何設定外部中斷信號的種類？
6. 試述 MCS-51 所提供的邏輯運算指令有哪幾種？

和自己賽跑！