微算機實驗報告

Lab # 6

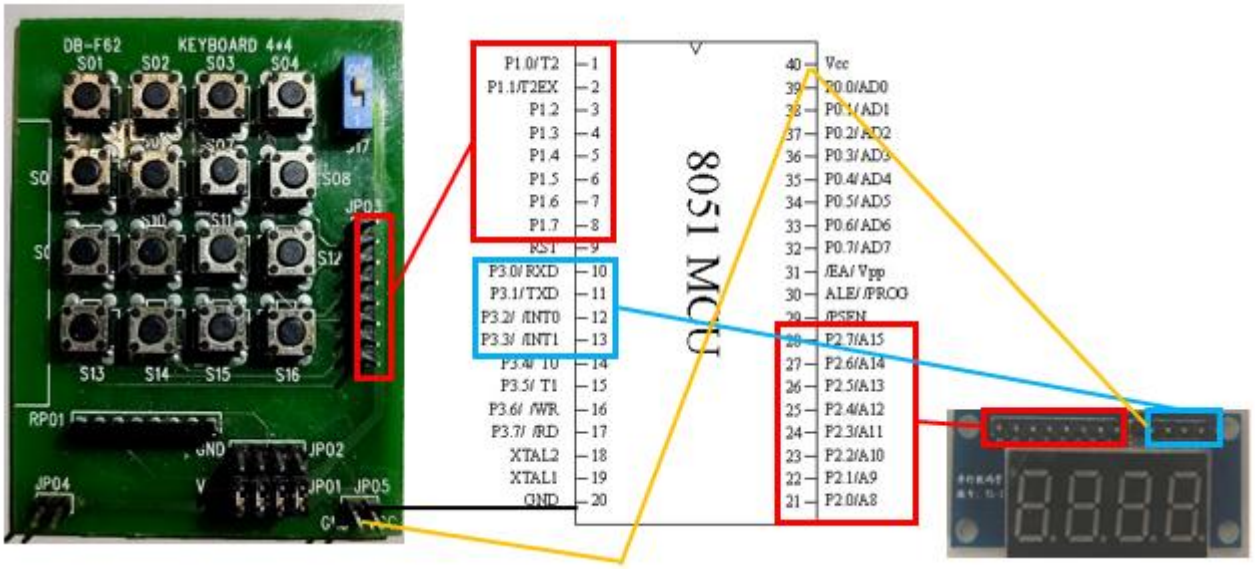
姓名：仇健安
系級：電機系
學號：111511239
上課時間：2025/04/01

一、實驗目的：

了解矩陣式鍵盤之電路架構與其工作原理，並學習以掃描方式驅動與讀取鍵盤輸入。透過操作 4×4 鍵盤模組，實作鍵值偵測與顯示控制，熟悉行列掃描技術。

二、硬體架構：

功能類別:	說明:	8051 腳位:	連接對象腳位:
鍵盤輸出	Row 1	P1.7	JP03.7
鍵盤輸出	Row 2	P1.6	JP03.6
鍵盤輸出	Row 3	P1.5	JP03.5
鍵盤輸出	Row 4	P1.4	JP03.4
鍵盤輸入	Column 1	P1.3	JP03.3
鍵盤輸入	Column 2	P1.2	JP03.2
鍵盤輸入	Column 3	P1.1	JP03.1
鍵盤輸入	Column 4	P1.0	JP03.0
七段顯示輸出	段碼資料 (A~G, DP)	P2.0–P2.7	SEG_A ~ SEG_DP (JP02.0-JP02.7)
顯示位元選通	顯示器選擇控制	P3.3	SEG1 (JP04.3)
顯示位元選通	顯示器選擇控制	P3.2	SEG2 (JP04.2)
顯示位元選通	顯示器選擇控制	P3.1	SEG3 (JP04.1)
顯示位元選通	顯示器選擇控制	P3.0	SEG4 (JP04.0)



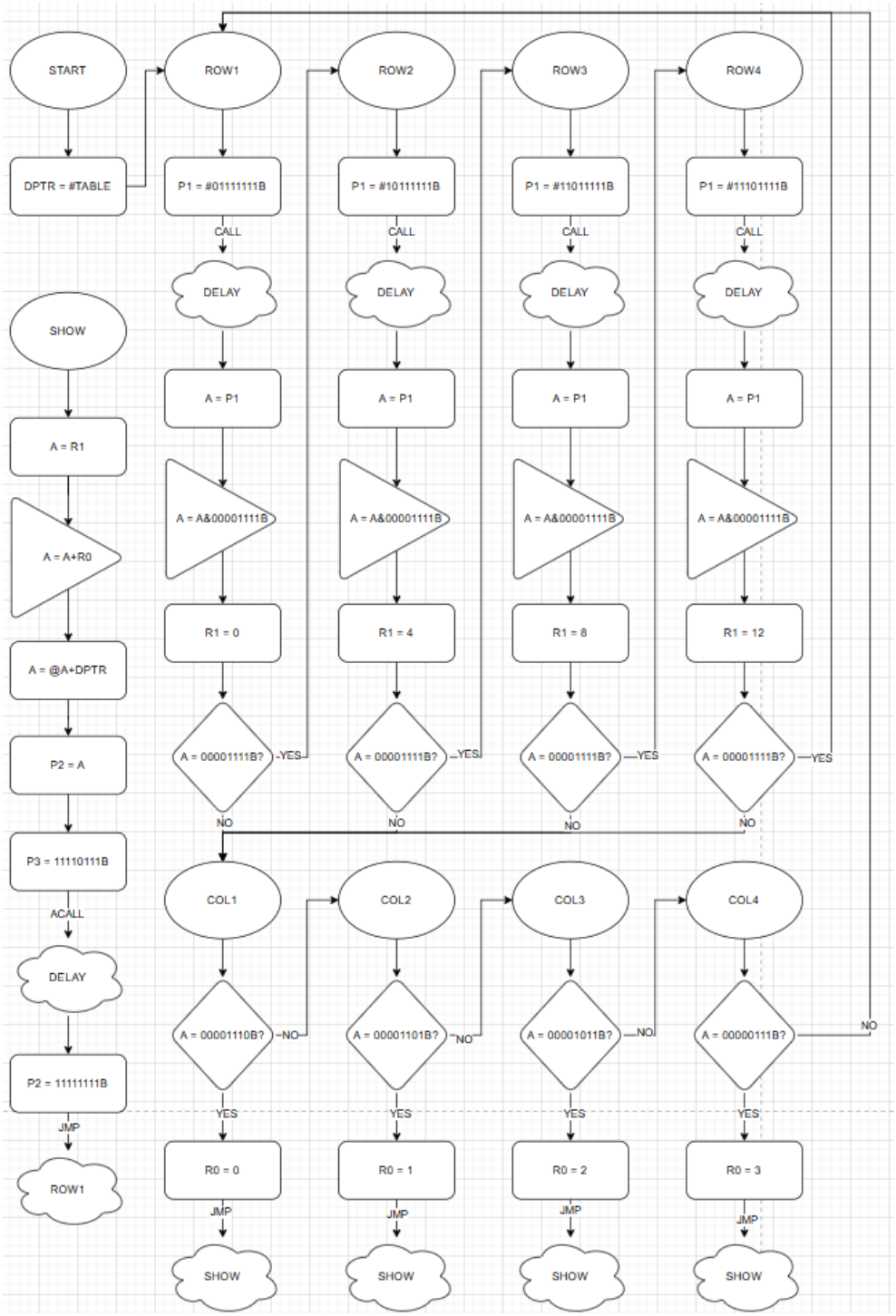
此 4×4 鍵盤模組由 16 個按鍵組成，按鍵以行（ROW0~R0W3）與列(~~ROW3-）與列（COL0~~COL0~COL3）交錯排布。每個按鍵的其中一端接至 Row 線，另一端接至 Column 線，因此鍵盤的實體結構是一個 4 行 ×4 列的矩陣型開關網路。

1. 每一列（ROW0~ROW3） 分別接至 JP03 的第 4 至第 7 腳（JP03.4~JP03.7），由微控制器輸出控制，透過程式設定逐行拉低進行掃描。
2. 每一行（COL0~COL3） 分別接至 JP03 的第 0 至第 3 腳（JP03.0~JP03.3），這些腳位會接上 4.7kΩ 的上拉電阻到 VCC，以確保在無按鍵按下時為高電位。
3. 按鍵閉合時導通：當某行被拉低，若對應按鍵被按下，則該行與該列形成閉路，微控制器便可從 Column 偵測到低電位，判斷按鍵位置。

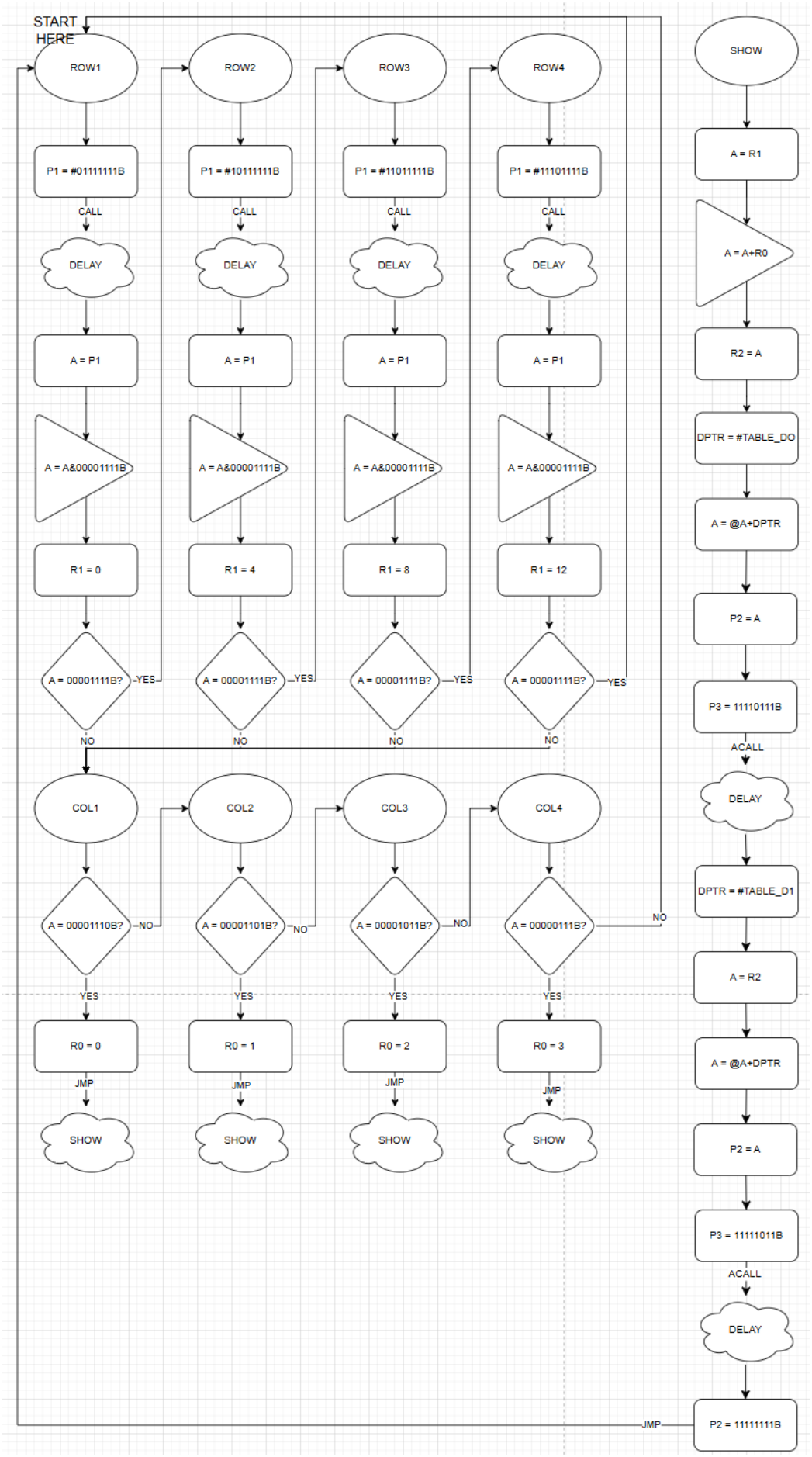
此設計透過上拉電阻保持穩定輸入狀態，搭配行列掃描程式邏輯，能有效偵測多個按鍵的輸入狀態，並減少 I/O 腳位的使用數量。

三、程式流程圖：

基本題:



進階題:


START HERE
ROW1
ROW2
ROW3
ROW4
P1 = #01111111B
P1 = #10111111B
P1 = #11011111B
P1 = #11101111B
CALL
DELAY
A = P1
A = A&00001111B
R1 = 0
R1 = 4
R1 = 8
R1 = 12
A = 00001111B?
YES
NO
COL1
COL2
COL3
COL4
A = 00001110B?
A = 00001101B?
A = 00001011B?
A = 00000111B?
R0 = 0
R0 = 1
R0 = 2
R0 = 3
JMP
SHOW
A = R1
A = A+R0
R2 = A
DPTR = #TABLE_DO
A = @A+DPTR
P2 = A
P3 = 11110111B
ACALL
DPTR = #TABLE_D1
A = R2
P3 = 11111011B
P2 = 11111111B

四、問題與討論：

(1) 一般開關在按下之後，必然有機械振動使接點開(open)、閉(close)多次才穩定觸合，如下圖 3 所示為開關彈跳波形，如果產生以下波形時，應如何消除彈跳？請就軟體面（程式）詳細說明你解決開關彈跳的方法。

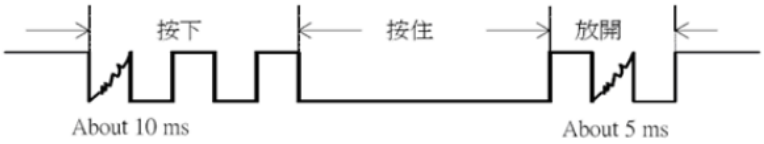


圖 3　開關彈跳波形

當開關被按下或放開時，會因機械接點彈性而產生數毫秒內的抖動（彈跳），導致電位在高低間快速切換。若不處理，將導致單次按鍵被誤判為多次輸入。為此，我們可透過以下軟體方式消除彈跳：

方法一：延遲確認法（Delay Debounce）

在偵測到按鍵被按下時，先延遲約 10～20 毫秒，再次讀取按鍵狀態；若仍為按下狀態，才視為有效輸入，否則忽略。

方法二：多次一致判定法（Majority Voting）

重複讀取該按鍵狀態數次（如每隔 1ms 取 10 次），若其中有超過 7 次以上為相同狀態，則視為穩定按下，能有效過濾雜訊。

方法三：狀態變化穩定計時法（Software Flag）

當偵測到與前一狀態不同時，啟動計時器（如 15ms）；若在此期間狀態維持不變，才承認為有效輸入，否則重設判斷。

(2) 在實際使用上，我們會賦予鍵盤每一個按鍵相對應的定義，而在應用上可能會需要同時按兩個按鍵來啟動某一個功能，例如: 通過 CTRL+V 來複製資料，請問我們該如何以掃描讀取資料的方式為基礎，來讀取兩個按鍵?請詳細說明你對該問題的解析及相對應的解決方法，不需要附程式碼。

要實作多鍵同時按下的偵測，需改變一般矩陣鍵盤的掃描方式與處理邏輯：

步驟一：掃描整個矩陣

傳統只偵測到第一顆被按下的鍵就跳出掃描；多鍵輸入需完整掃描所有行列，將所有有按下的鍵紀錄下來

步驟二：對應多鍵輸入

透過行列位置轉換為對應的鍵值，並同時紀錄兩顆以上被按下的鍵。例如：偵測到 [Row1, Col0] 與 [Row2, Col1] 同時為低電位，則對應兩個按鍵被按下

步驟三：設定同時性條件

為避免使用者兩鍵放開間隔過久，可設定兩按鍵需在 50ms 以內同時偵測放開，才視為有效的組合輸入。

五、程式碼與註解：

基本題:

```
    ORG 00H                 ; 程式起始地址
    JMP START               ; 開機後跳轉至 START 執行主程式
    ORG 50H                 ; 主程式從地址 50H 開始

START:
    MOV DPTR, #TABLE        ; 將資料表 TABLE 的地址載入資料指標 DPTR（用於查七段碼）

; ===== 行列掃描鍵盤部分 =====

ROW1:
    MOV P1, #01111111B      ; 啟用 Row1（P1.7=0 其餘為 1），開始掃描第一行
    CALL DELAY              ; 延遲去彈跳
    MOV A, P1               ; 讀取 P1 狀態（讀取 Column 狀態）
    ANL A, #00001111B       ; 清除高 4 位，只保留 P1.0~P1.3（Column）
    MOV R1, #0              ; 記錄當前掃描的行編號（Row1）
    CJNE A, #00001111B, COL1; 若有 Column 變低（按鍵按下），跳至 COL1 判斷列編號

ROW2:
    MOV P1, #10111111B      ; 啟用 Row2（P1.6=0）
    CALL DELAY
    MOV A, P1
    ANL A, #00001111B
    MOV R1, #4              ; 記錄行編號（Row2 起始偏移為 4）
    CJNE A, #00001111B, COL1

ROW3:
    MOV P1, #11011111B      ; 啟用 Row3（P1.5=0）
    CALL DELAY
    MOV A, P1
    ANL A, #00001111B
    MOV R1, #8              ; 記錄行編號（Row3 起始偏移為 8）
    CJNE A, #00001111B, COL1

ROW4:
    MOV P1, #11101111B      ; 啟用 Row4（P1.4=0）
    CALL DELAY
    MOV A, P1
    ANL A, #00001111B
```

```
    MOV R1, #12              ; 記錄行編號（Row4 起始偏移為 12）
    CJNE A, #00001111B, COL1

    JMP ROW1                 ; 若沒有偵測到按鍵，重新從 Row1 掃描

; ===== 判斷按下的是哪一列（Column） =====

COL1:
    CJNE A, #00001110B, COL2 ; 判斷是否為 Column0（只有 P1.0=0）
    MOV R0, #0               ; Column0 → 編號 0
    JMP SHOW

COL2:
    CJNE A, #00001101B, COL3 ; 判斷是否為 Column1（P1.1=0）
    MOV R0, #1               ; Column1 → 編號 1
    JMP SHOW

COL3:
    CJNE A, #00001011B, COL4 ; 判斷是否為 Column2（P1.2=0）
    MOV R0, #2               ; Column2 → 編號 2
    JMP SHOW

COL4:
    CJNE A, #00000111B, ROW1 ; 判斷是否為 Column3（P1.3=0）
    MOV R0, #3               ; Column3 → 編號 3

; ===== 顯示按鍵對應的七段碼 =====

SHOW:
    MOV A, R1                ; A = 行偏移值（如 Row2 開頭為 4）
    ADD A, R0                ; A = 行 + 列 → 鍵盤矩陣位置（0~15）
    MOVC A, @A+DPTR          ; 從 TABLE 中讀取對應的七段顯示碼
    MOV P2, A                ; 將七段碼送到 P2（控制七段顯示器）

    MOV P3, #11110111B       ; 啟用第 1 位顯示器（P3.3=0，其餘為 1）
    ACALL DELAY              ; 延遲（供人眼辨識）
    MOV P2, #11111111B       ; 清空七段顯示器（關閉顯示）
    JMP ROW1                 ; 回到掃描鍵盤的流程

; ===== 延遲副程式（簡單的雙迴圈 delay） =====
```

```
DELAY:
    MOV R5, #100              ; 外層迴圈次數
DELAY1:
    MOV R6, #150              ; 內層迴圈次數
DELAY2:
    DJNZ R6, DELAY2           ; 內層倒數
    DJNZ R5, DELAY1           ; 外層倒數
    RET

; ===== 七段顯示碼表（對應 0~F） =====

TABLE:
    DB 0C0H, 0F9H, 0A4H, 0B0H ; 0~3
    DB 099H, 092H, 082H, 0F8H ; 4~7
    DB 080H, 090H, 088H, 083H ; 8~B
    DB 0C6H, 0A1H, 086H, 08EH ; C~F

    END                       ; 程式結束

進階題:
    ORG 00H                   ; 程式起始位置
    JMP MAIN                  ; 跳至 MAIN 開始主程式
    ORG 50H                   ; 主程式從記憶體地址 50H 開始

MAIN:
; === 掃描 ROW1 ===
ROW1:
    MOV   P1, #01111111B      ; 啟用 Row1（P1.7 = 0，其餘為 1）
    CALL DELAY                ; 延遲去彈跳
    MOV A, P1                 ; 讀取鍵盤輸入
    ANL A, #00001111B         ; 保留低 4 bit（P1.0~P1.3），代表 Column 狀態
    MOV R1, #0                ; R1 儲存目前掃描行的偏移值
    CJNE A, #00001111B, COL1 ; 若有任何按鍵被按下（某列為 0），跳至 COL 判斷

; === 掃描 ROW2 ===
ROW2:
    MOV   P1, #10111111B      ; 啟用 Row2（P1.6 = 0）
    CALL DELAY
    MOV A, P1
    ANL A, #00001111B
    MOV R1, #4                ; 第二行偏移值為 4
```

```
    CJNE A, #00001111B, COL1

; === 掃描 ROW3 ===
ROW3:
    MOV   P1, #11011111B        ; 啟用 Row3（P1.5 = 0）
    CALL DELAY
    MOV A, P1
    ANL A, #00001111B
    MOV R1, #8                  ; 第三行偏移值為 8
    CJNE A, #00001111B, COL1

; === 掃描 ROW4 ===
ROW4:
    MOV   P1, #11101111B        ; 啟用 Row4（P1.4 = 0）
    CALL DELAY
    MOV A, P1
    ANL A, #00001111B
    MOV R1, #12                 ; 第四行偏移值為 12
    CJNE A, #00001111B, COL1

    JMP ROW1                    ; 若沒偵測到按鍵，回到 Row1 繼續掃描

; === 判斷是哪一列（Column）被按下 ===
COL1:
    CJNE A, #00001110B, COL2 ; 判斷是否 Column 0（P1.0 = 0）
    MOV R0, #0                  ; 記錄列編號為 0
    JMP SHOW

COL2:
    CJNE A, #00001101B, COL3 ; 判斷是否 Column 1（P1.1 = 0）
    MOV R0, #1
    JMP SHOW

COL3:
    CJNE A, #00001011B, COL4 ; 判斷是否 Column 2（P1.2 = 0）
    MOV R0, #2
    JMP SHOW

COL4:
    CJNE A, #00000111B, ROW1 ; 判斷是否 Column 3（P1.3 = 0），若無回掃描
    MOV R0, #3
```

```
; === 顯示對應的七段碼 ===
SHOW:
    MOV A, R1                   ; 將行偏移（R1）存入 A
    ADD A, R0                   ; 加上列編號（R0），得出按鍵編號 0~15
    MOV R2, A                   ; 將總鍵碼存入 R2

; 顯示第一位（TABLE_D0 控制字型）
    MOV DPTR, #TABLE_D0         ; 指向七段碼資料表 0（低位）
    MOVC A, @A+DPTR             ; 讀出七段碼資料
    MOV P2, A                   ; 顯示在七段顯示器
    MOV P3, #11110111B          ; 選通第 1 顆七段顯示器
    ACALL DELAY                 ; 顯示延遲

; 顯示第二位（TABLE_D1 控制字型）
    MOV DPTR, #TABLE_D1         ; 指向七段碼資料表 1（高位）
    MOV A, R2
    MOVC A, @A+DPTR             ; 讀取高位顯示資料
    MOV P2, A
    MOV P3, #11111011B          ; 選通第 2 顆七段顯示器
    ACALL DELAY

    MOV P2, #11111111B          ; 清除顯示器
    JMP ROW1                    ; 繼續掃描按鍵

; === 延遲副程式（雙層延遲迴圈） ===
DELAY:
    MOV R5, #100
DELAY1:
    MOV R6, #150
DELAY2:
    DJNZ R6, DELAY2
    DJNZ R5, DELAY1
    RET

; === 七段碼對照表（低位） ===
TABLE_D0:
    DB 0C0H,0F9H,0A4H,0B0H   ; 0~3
    DB 099H,092H,082H,0F8H   ; 4~7
    DB 080H,090H,0C0H,0F9H   ; 8~B
    DB 0A4H,0B0H,099H,092H   ; C~F
```

```
; === 七段碼對照表（高位） ===
TABLE_D1:
    DB 0FFH,0FFH,0FFH,0FFH   ; 全熄
    DB 0FFH,0FFH,0FFH,0FFH
    DB 0FFH,0FFH,0F9H,0F9H   ; 顯示 "1"
    DB 0F9H,0F9H,0F9H,0F9H

END                          ; 程式結束
```

六、心得：

上課心得

這次課堂內容主要講解矩陣鍵盤的結構與掃描原理，透過老師詳細地介紹鍵盤的行列組成、如何以程式控制方式掃描按鍵，讓我更深入理解實際硬體的架構。老師在講解過程中也搭配實體範例與圖解，使我能夠清楚掌握鍵盤按在按鈕按下後實際電路的反饋是如何。

實驗心得

這次實驗實作了鍵盤掃描程式，並將按下的數字顯示在七段顯示器上。透過實際撰寫與上傳程式，我學會了如何控制輸出腳位進行逐行掃描、如何使用邏輯運算來判斷哪個按鍵被觸發，並運用查表法將對應值轉換為七段碼輸出顯示。實作過程中，我也理解了開關彈跳的影響與消除方式，並學習到如何延遲與消除錯誤觸發。

Notes:

1. 內容字體大小為 12，間距為單行間距
2. 中文字字體為標楷體
3. 英文字和阿拉伯數字為 Times New Roman
4. 嚴禁抄襲，抄襲者以 0 分計算
5. 請於報告左上角附上照片
6. 每次實驗課繳交上次實驗結報