微處理機 lab2 結報

0413335 郭逸琳 0416039 李佳燕

實驗主題與訓練目的:

此次實驗主要是要透過程式使 LED 燈能按照特定的規則發亮，並透過更改程式碼讓自己更加熟悉組合語言。

實驗過程

將以下程式碼放入 uvision 的檔案中並依照 lab1 的方法執行他：

```
org 0
    mov sp, #50H
    clr c
    mov a, #0feH
    mov R7, a
    mov a, #0fH
mk1:
    cpl a
    mov r6, a
    mov p1, A
    mov a, r7
    mov p0, a
    call delay
    rlc a
    mov r7, a
    mov a, r6
```

```
        jc mk1
        mov a, #0ccH
mk2:
        cpl a
        mov r6, a
        mov p1, a
        mov a, r7
        mov p0, a
        call delay
        rrc a
        mov r7,a
        mov a, r6
        jc mk2
        mov a, #0f0H
mk3:
        cpl a
        mov r6, a
        mov p1, a
        mov a, r7
        mov p0, a
        call delay
        rlc a
        mov r7, a
        mov a, r6
```

```
        jc mk3
        mov a, #0ccH
mk4:
        cpl a ; XXX
        mov r6, a
        mov p1, a
        mov a, r7
        mov p0, a
        call delay
        rrc a
        mov r7,a
        mov a, r6
        jc mk4
        mov a, #0fH
        jmp mk1
delay:
        push 5 ; push R5
        push 6
        push 7
        mov r5, #20
dd1:
        mov r6, #200
dd2:
        mov r7, #250
```

```
    djnz r7, $
    djnz r6, dd2
    djnz r5, dd1
    pop 7
    pop 6
    pop 5
    ret
end
```

(因 Lab2 指示的程式碼並不符合他要求的 LED 發亮規則，故稍作修改)

程式說明：(原本 spec 上的 code)
一、一開始把 0FH(00001111)給 r6，feH 給 R7

1. mk1 會先讓 R6 的值接到 p1，complement 之後讓 R7 的值接到 p0，call 完 delay 後對 R7 的值做 rlc 然後再跳回 mk1 總共 8 個 cycle(因為 jc 的關係)。
2. mk1 會讓接 p1 的 LED 右半左半閃爍，讓接 p0 的 LED 從最右邊的燈開始亮，之後一次亮兩顆，直到接 p0 的 LED 的最左邊的燈亮了為止。

二、然後 R7 的值維持不變(依然是 mk1 結束時的值)，R6 的值改為 0ccH(11001100)

1. mk2 會先讓 R6 的值接到 p1，complement 之後讓 R7 的值接到 p0，call 完 delay 後對 R7 的值做 rrc 然後再跳回 mk2 總共 8 個 cycle(因為 jc 的關係)。
2. mk2 會讓接 p1 的 LED"亮亮暗暗亮亮暗暗"變"暗暗亮亮暗暗亮亮"，讓接 p0 的 LED 從最左邊的燈開始亮，之後一次亮兩顆，直到接 p0 的 LED 的最右邊的燈亮了為止。

三、再來 R7 的值還是維持不變(依然是 mk2 結束時的值)，R6 的值改為 0F0H(11110000)

1. mk3 會先讓 R6 的值接到 p1，complement 之後讓 R7 的值接到 p0，call 完 delay 後對 R7 的值做 rlc 然後再跳回 mk3 總共 8 個 cycle(因為 jc 的關係)。
2. mk3 會讓接 p1 的 LED 左半右半閃爍，讓接 p0 的 LED 從最右邊的燈開始亮，之後一次亮兩顆，直到接 p0 的 LED 的最左邊的燈亮了為止。

四、最後 R7 的值依然維持不變(依然是 mk3 結束時的值)，R6 的值改為 0ccH(00110011)

1. mk4 會先讓 R6 的值接到 p1，complement 之後讓 R7 的值接到 p0，call 完 delay 後對 R7 的值做 rrc 然後再跳回 mk4 總共 8 個 cycle(因為 jc 的關係)。
2. mk4 會讓接 p1 的 LED"亮亮暗暗亮亮暗暗"變"暗暗亮亮暗暗亮亮"，讓接 p0 的 LED 從最左邊的燈開始亮，之後一次亮兩顆，直到接 p0 的 LED 的最右邊的燈亮了為止。

註：delay function 跟 LAB1 相同，在更改 R5、R6、R7 的值以前記得要把 R5、R6、R7 的值 push 到 stack，跑完 delay function 後再把 R5、R6、R7 的值 push 回去，裡面以免更改到他們原本的值。

實驗結果:

(改完 code 之後)
LED 燈會依照以下規則發亮：
接 p0 的 LED 為以兩顆燈泡為單位，依序來回發亮。
接 p1 的 LED 則是以以下兩種模式交替輪迴執行：

1. 一半亮一半暗並互相交換
2. 以兩顆燈泡為單位，亮暗相間並互相交換

p1 的 LED 燈兩種模式交替的時機是在接 p0 的 LED 跑完一排後發生的。

實驗延伸：

一、因此次實驗的組合語言稍長，於是我們思考是否可以簡化程式。而簡化的結果如下：

```
org 0

    mov sp, #50H

    clr c

    mov a, #0feH

    mov R7, a

    mov a, #0fH

    mov R5, a

    mov a, r5

mk1:
```

```
        cpl a
        mov r6, a
        mov p1, A
        mov a, r7
        mov p0, a
        call delay
        rlc a
        mov r7, a
        mov a, r6
        jc mk1
        mov a, #0ccH
mk2:
        cpl a
        mov r6, a
        mov p1, a
        mov a, r7
        mov p0, a
        call delay
        rrc a
        mov r7,a
        mov a, r6
        jc mk2
        mov p2, r5
        jb p2.1, mk3
```

```
        mov a, #0fH
        mov R5, a
        jmp mk4
mk3:
        mov a, #0f0H
        mov R5, a
mk4:
        mov a, r5
        jmp mk1
delay:
        push 5 ; push R5
        push 6
        push 7
        mov r5, #20
dd1:
        mov r6, #200
dd2:
        mov r7, #250
        djnz r7, $
        djnz r6, dd2
        djnz r5, dd1
        pop 7
        pop 6
        pop 5
```

ret

end

因我們發現 mk3 與 mk4 執行的程式與 mk1、mk2 相似，故把 mk3、mk4 砍掉，並加上一個判斷式判斷下一個要執行的是 mk1 還是 mk3。因網路上查到 jb 是用板子上的某個 pin 角中特定的 bit 判斷的，故用一個不會影響 LED 的 pin(p2)，對此輸入特定的值來判斷。

二、For the code line marked with ;XXX, how would the display pattern sequence changed if it is removed?
(原本 spec 上的 code)

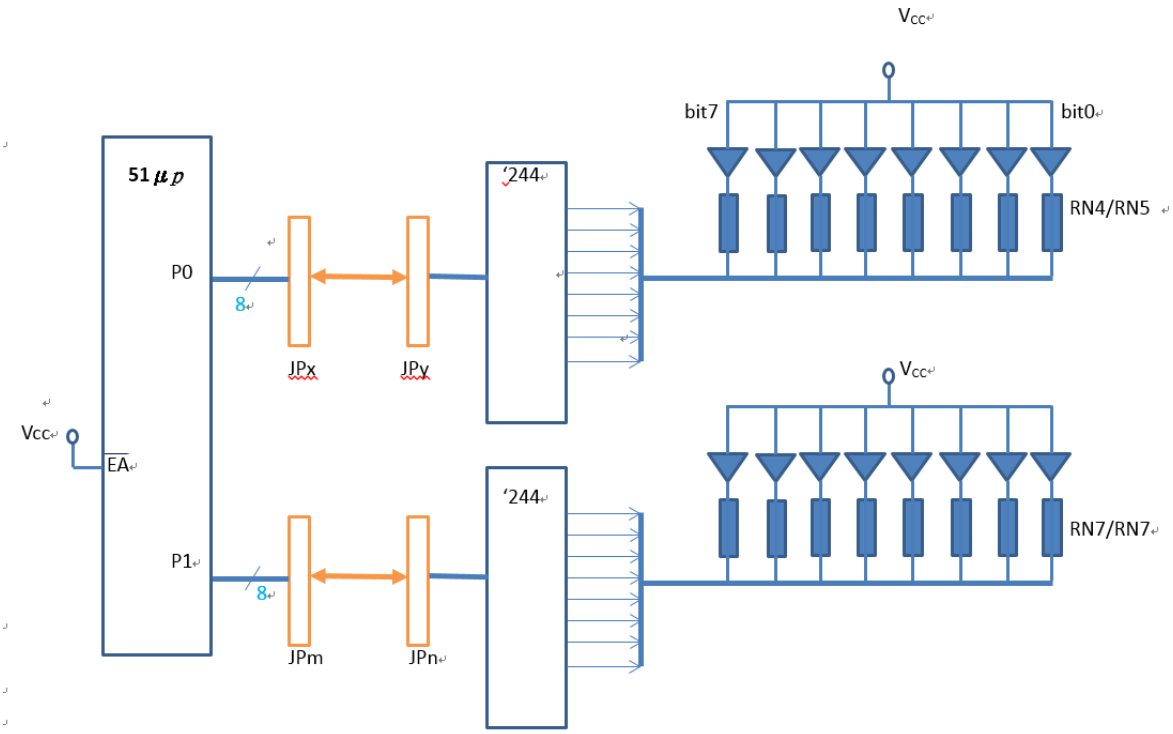
```
        mov     a, #0ccH        a:11001100
mk4:
        mov     r6, a           r6:11001100 11001100
        mov     p0, a           p0:11001100 11001100
        mov     a, r7           a:01111111 00111111
        mov     p1, a           p1:01111111 00111111
        call    delay
        rrc     a               c:1. a:00111111 10011111
        mov     r7,a            r7:00111111 10011111
        mov     a, r6           a:11001100 11001100
        jc      mk4
```

假設不要 complement a, 則會發現接 p0 的 LED 在 mk4 裡面永遠只會呈現"暗暗亮亮暗暗亮亮",不會有閃爍的效果; 但接 p1 的 LED 的功能還是正常的, 從左邊亮到右邊

三、Can you identify the stack status (where SP is pointing to, contents of the stack, etc.) at any instance during the task execution?
一開始 sp 是 50H
在 delay function 裡面先 push 5(push R5), sp 目前會指到 51H, 所以會在 51H 的地方放入 R5 的內容;
接下來 push 6(push R6), sp 目前會指到 52H, 所以會在 52H 的地方放入 R6 的內容;
再來 push 7(push R7), sp 目前會指到 53H, 所以會在 53H 的地方放入 R7 的內容
… 接下來做 dd1 & dd2
Pop 7(pop R7), sp 目前指到 53H, 所以會在 53H 的地方拿內容走, 並放到 R7, sp--變成 52H
Pop 6(pop R6) , sp 目前指到 52H, 所以會在 52H 的地方拿內容走, 並放到 R6, sp--

變成 51H
Pop 5(pop R5) , sp 目前指到 51H, 所以會在 51H 的地方拿內容走, 並放到 R7, sp--
變成 50H

實驗電路與驅動程式:



實驗心得:

此次實驗主要是透過更改程式碼讓自己更加熟悉組合語言。透過這次實驗，我們更加了解了 89c51 的運作方式，並更加熟悉組合語言該怎麼使用，希望下次實驗能讓我們更加地了解要如何利用組合語言做出更有趣的東西。

實際去 trace code 一次:

```
        org     0
        mov     sp, #50H
        clr     c
        mov     a, #0feH
        mov     R7, a

        mov     a, #0fH
mk1:
        cpl     a
        mov     r6, a
        mov     p1, A
        mov     a, r7
        mov     p0, a
        call    delay
        rlc     a
        mov     r7, a
        mov     a, r6
        jc      mk1

        mov     a, #0ccH
mk2:
        cpl     a
        mov     r6, a
        mov     p1, a
        mov     a, r7
        mov     p0, a
        call    delay
        rrc     a
        mov     r7,a
        mov     a, r6
        jc      mk2

        mov     a, #0f0H
mk3:
        cpl     a
        mov     r6, a
        mov     p1, a
        mov     a, r7
        mov     p0, a
        call    delay
        rrc     a
        mov     r7,a
        mov     a, r6
        jc      mk3

        mov     a, #0ccH
mk4:
        cpl     a       ; XXX
        mov     r6, a
        mov     p1, a
        mov     a, r7
        mov     p0,a
        call    delay
        rlc     a
        mov     r7,a
        mov     a, r6
        jc      mk4

        mov     a, #0fH
        jmp     mk1
delay:
        push    5       ; push R5???
        push    6
        push    7
        mov     r5, #2
dd1:
        mov     r6, #200
dd2:
        mov     r7, #250
        djnz    r7, $
        djnz    r6, dd2
        djnz    r5, dd1
        pop     7
        pop     6
        pop     5
        ret
```