D組員: 0616318蔡承恩

實驗目的:

學習程式與電路板的關係，利用組合語言控制7段顯示器閃爍

設計圖:

實驗環境:

u-vision

實驗要點

1. 本次lab的51μp控制兩個port: P1, P2，P1控制的是七段顯示器的開關，本次lab只用到最低位的6bit，分別控制由左至右P1[5]~P1[0]分別控制由左至右6個七段顯示器，當該bit被設為0時，則七段顯示器為開，反之則為關；P2控制的是七段顯示器要顯示的數字，因為要表示0~9十個數字只需要4bit，所以本次lab只要控制P2的最低4位，就可以顯示數字。

2. 在同一瞬間，這6個七段顯示器能顯示的數字都是同一個，但是透過人眼視覺暫留的現象可以感覺達到同時顯示不同數字的效果，以這次lab為例，在phase 1中，要求6個七段顯示器由左至右依序顯示5~0，實作上是讓6個七段顯示器輪流顯示，一直快速循環，對人眼來說就會有同時顯示不同數字的效果:

	Display1	Display2	Display3	Display4	Display5	Display6
phase1	Off	Off	Off	Off	Off	0
phase2	Off	Off	Off	Off	1	Off
phase3	Off	Off	Off	2	Off	Off
phase4	Off	Off	3	Off	Off	Off
phase5	Off	4	Off	Off	Off	Off
phase6	5	Off	Off	Off	Off	Off

CODE:

```
        org     0
        mov     SP, #50H
start:  mov     R7, #6
next1:
        mov     R5, #250
next11:
        mov     R6, #6
```

```
        mov     R1, #0FEH
        mov     R2, #0
next12:
        mov     A, R1
        mov     P1, A
        RL      A
        mov     R1, A
        mov     A, R2
        inc     R2
        mov     P2, A
        call    delay1      ; ===KKK===
        djnz    R6, next12
        djnz    R5, next11   ;DALAY = R5*delay1
        mov     P1, #0FFH
        call    delay2      ; ===LLL===
        djnz    R7, next1
        mov     R7, #6
next2:
        mov     P1, #0F8H
        mov     P2, #5
        call    delay2      ; ===III===
        mov     P1, #0c7H
        mov     P2, #1
        call    delay2      ; ===JJJ===
        djnz    R7, next2
```

```
        jmp     start
  delay1:
push    1
mov     R1, #255
djnz    R1, $
pop     1
ret
      delay2:           ; appx. 0.5sec delay, why? O
;push    1
;push    2
;push    3
mov     R1, #200
dd22:  mov     R2, #250
dd21:  mov     R3, #10
djnz    R3, $
djnz    R2, dd21
djnz    R1, dd22
;pop     3
;pop     2
;pop     1
ret
end
```

CODE解釋:

初始: SP指向50H做為stack的起始位置

R1： 1.儲存數字應該要顯示的位置 2.作為delay計時的暫存器

R2： 1.儲存數字為多少 2作為delay計時的暫存器

R3： 作為delay計時的暫存器

R6 : 共有6個LED顯示器，一次循環代表個別LED的顯示

R7: 控制 pattern 出現的次數，一個完整的pattern 分別為next1 next2

next1:

顯示 “5 4 3 2 1 0” 和 “x x x x x x”(led不顯示)，循環共6次，跳到next1

圖例:

part1 : 初始R1為#0FEH傳送到P1，控制LED顯示位置(第一盞LED);R2為#0傳送到P2，控制LED的顯示數字。call delay1, 接著透過指令rr R1，inc R2,顯示下一個LED的位置與數字，總共做6次，(順序由右向左顯示，由0到5)。做完part1，重新回到part1，共250次。

part2: 設R1為#FFH傳送P1，控制LED6個全暗，call delay2，跳回part1。

在part1、part2循環6次再進入next2

next2:

顯示 "x x x 5 5 5” 和 "3 3 3 x x x”，循環共6次，跳到next2

圖例:

part3 : 初始R1為#0F8H傳送到P1，控制LED顯示位置(第一盞LED);R2為#5傳送到P2，控制

LED的顯示數字。(傳送p1 #1111 1000控制右邊3個LED顯示，傳送p2#5顯示LED的數字為"5")，call delay2，跳到part4

part4: 初始R1為#0C7H傳送到P1，控制LED顯示位置(第一盞LED);R2為#1傳送到P2，控制

LED的顯示數字。(傳送p1 #1100 011控制左邊3個LED顯示，傳送p2#1顯示LED的數字為"1")，call delay2，跳到part3

part3 part4 循環共6次，再跳回到start

* difficulties encountered and resolving measures:

原本把P1接到七段顯示器那區的下排，把P2接到七段顯示器那區的上排，發現顯示怪怪的，去檢查一下code認為應該總要有個東西對才對，所以懷疑是接線的問題，後來把P1接到七段顯示器那區的上排，把P2接到七段顯示器那區的下排就成功了。

* phenomena observed and explanations:

Code上發現:

按照lab2.2給的code可以執行，而結果會是543210閃6次，接著是右邊3個5(XXX555)和左邊3個1(111XXX)輪著閃6次。透過code，發現R1 is position, R2 is the number。

其實一開始的543210並不是同時閃，而是先用R1設定是0，用R2設定位置為最右邊那一格，顯示0了之後，把R1從0變成1，再用R2設定位置為右邊第二

格，以此類推，便能顯示543210。之所以能同時顯示是因為instruction執行的速度很快，因為數字間顯示的時間差的極短所以我們會認為是同時顯示。

Delay問題:

===III===：3個5(XXX555)會顯示的時間，要是沒有這一行，會幾乎看不到右邊的那3個5(XXX555)

===JJJ===：3個1(111XXX)會顯示的時間，要是沒有這一行，會幾乎看不到左邊的那3個1(111XXX)

===LLL===：543210與下次5432110中間間隔全暗的時間，要是沒有這一行，會幾乎看不到閃爍而是認為543210一直亮著

七段顯示器:

七段顯示器那區的上排的8bits中，只有7bits是控制位置，分別是控制小數點，第一個數字，第二個數字，.....，第六個數字。第幾個bit為0的時候第幾個位置才會亮，1為不亮，MSB似乎什麼都不影響，像是FFH(11111111)代表全暗

七段顯示器那區的下排的8bits中，只有4bits會控制數字

push-pop問題:

delay1的push-pop不可以省略，因為R1等等還會用到，而不是先被重新賦值，所以必須要特別先push之後再pop回來

delay2的push-pop可以省略，因為R3不會被用到，所以不需要特別先push之後再pop回來，而R1和R2因為很快就會被重新賦值，重新賦值完才會用到，所以不用特別用push-pop保存

delay問題2:

===KKK===：讓543210可以同時出現的關鍵，要是沒有這一行，會幾乎看不到543210因為閃太快

要是把===KKK===的delay1的時間乘上1000倍，543210看起來不太像是同時出現，而是像接續出現

D、C、B、A：為BCD碼輸入端。輸入值應為0000~1001(0~9)。

a , b , c , d , e , f 為輸出端，可以控制輸出位置。

LT為燈測試輸入，當BI/RBO為1，而LT為0時，所有LED均發亮，若顯示器上有某數字不亮，則可判定期損壞或沒有接妥。

BI/RBO前導零遮沒輸出腳，當其為0時，不管它輸入狀態為何，所有LED將不亮，一般此接腳是街道下一級(低一位數)之RBI接腳。

self-evaluation

* things learnt

學會最基本的流程，了解如何用code去控制七段顯示器(數字和位置)，還有要怎麼利用delay和視覺暫留以達到同時顯示543210，已及了解到更多push-pop的用法

* retrospections raised to oneself

在開始用code前記得hardware已經都準備好了，不要在忘記接電源了。要debug的時候不要一次改太多東西，一次變動一點就好比較好之道bug發生在哪。還有有些事情不要太直覺，像是不要總認為1是亮0是暗，七段顯示器就是相反。記得要保存直的時候用push-pop