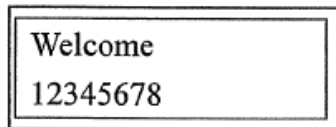


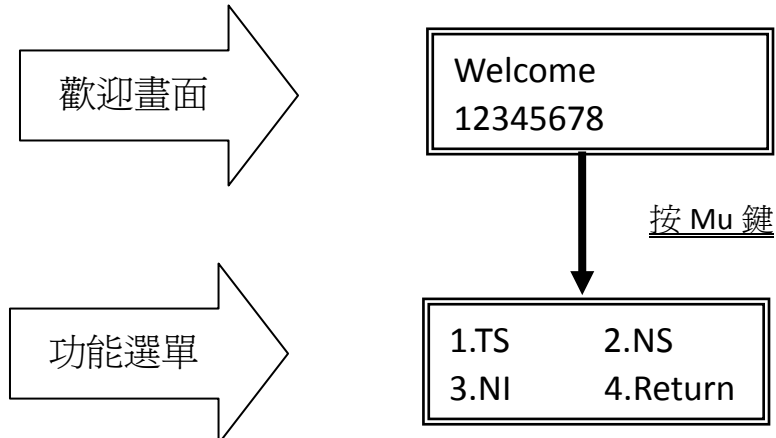
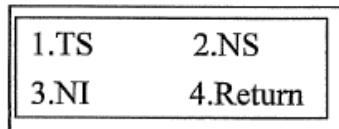
4-1 系統初始設計

4-1-1 題目：

1. 當開啓電源或重置系統之後，在 LCM 上必須先進行『歡迎畫面』的顯示，LCM 的第一列呈現” Welcome”、第二列則呈現” 准考證號碼” (假設准考證號碼爲 12345678)等內容，實際畫面應該如下所示。



2. 當系統在歡迎畫面之時，若操作者按下” Mu” 鍵，LCM 應該立即出現『功能選單』，功能選單的畫面必須在第一列呈現出” 1.TS 2.NS”、第二列則呈現” 3.NI 4.Return” 等內容，實際畫面如下所示；其中 TS 代表 Team Set 功能、NS 代表 Number Set 功能、NI 表示 Number Input 功能，而 Return 則代表畫面返回歡迎畫面。



4-1-2 程式設計說明：

1. 首先利用考場提供之 8255-lcm-kb.a 這個電子檔來修改，在考場提供之隨身碟建新資料夾 work2，並將 8255-lcm-kb.a 拷貝至此資料夾。
2. 在 work2 資料夾中建立新專案 ex2，並將 8255-lcm-kb.a 檔案包含至此專案中。
3. 8255-lcm-kb.a 程式功能說明：

此程式包含 3 個主要段落

a. 接腳和暫存器宣告區

新增加的接腳定義和暫存器命名宣告都拷貝到這裡，記得不要重複定義，某些暫存器宣告記得要改到可位元定址位 20H~2FH 才可以正常動作。

b. 主程式區

所有新增加的程式都由這裏往下加。

c. 副程式區

原則上這裡都不需要修改，只要把其他程式需要用到的副程式拷貝到這個地方，位置不拘，原則上放在附程式區前面即可，已經有的副程式千萬不要再拷貝過來。

以下是程式 8255-lcm-kb.a 架構說明

```
;-----8255 registers define-----
ctrl55    equ  0x5f
pa55      equ  0x5e
pb55      equ  0x5d
pc55      equ  0x5c
pcbit     equ  0x5b
;
a0_8255   equ  p2.0
a1_8255   equ  p2.1
;*****keyboard rigisters define*****
keynum    equ  0x5a          ;keynumber
keydbl    equ  0x59          ;debug bounce,double check
counter   equ  0x58          ;scan keyboard array
bounce0    equ  0x57
bounce1    equ  0x56
key_f     equ      22h.0
;*****lcm rigisters define*****
abuf      equ  0x55          ;the data write to lcm
atmp      equ  0x54          ;the data read from lcm
;
;
```

接腳和暫存器宣告都在這裡

主程式從這裡開始

```
org 00h

Initial:
    mov ctrl55,#82h      ;8255 initial status setting
    call wr_ctrl         ;pa:output,pb:input,pc:output
; ::::: Initial LCD :::::
;-----lcm function setting
    mov abuf,#38h        ; Set for 8 bit data transfer and
    call wr_lcdcmd       ; address type b,font=5*7
;-----lcm display control
    mov abuf,#0eh        ; Ture on the display and cursor.
    call wr_lcdcmd       ; cursor no twinkle
;-----lcm entry model setting
    mov abuf,#06h        ; Set the LCD unit to increment
    call wr_lcdcmd       ; the address counter.
;
; 20h store in all dd ram address
sho_main:
    mov dptr,#main_scr1  ; Get the 1st row data(address)
    call sho_lcm1        ; Call lcm display sub-program2
    mov dptr,#main_scr2  ; Get the 2nd row data(address)
    call sho_lcm2        ; Call lcm display sub-program2
;++++++main program
start++++++
loop:
    call scan_key
    mov a,keynum
    cjne a,#0ah,loop     ;判斷是否為 Menu 鍵
;
sho_menu:
    mov dptr,#menu_scr1
    call sho_lcm1
    mov dptr,#menu_scr2
    call sho_lcm2
;-----
    jmp sho_main
;*****lcm display data sub-program*****
sho_lcm1:
    call lcd_cls         ;clear display and cursor return
    call lcd_printstring
```

這裡需要顯示字串 menu_scr1
和 menu_scr2，後面沒有宣告
這 2 個字串，組譯會有錯誤

改成 JMP \$

主程式到此為止，要加的主程式
從這裡開始加

從這裡開始是副程式，要加的副
程式都拷貝到這裡

```

        ret
sho_lcm2:                ;=====cursor to second line
        clr a            ;homing to left
        setb b.0         ;Cursor to second line(1)
        call lcd_locate   ;b.0=0:line0,b.0=1:line1
        call lcd_printstring
        ret

;=====display data array=====
main_scr1:  db  "TEMI-uC",0
main_scr2:  db  "2223-9560",0
;=====

lcd_cls:
        mov abuf, #01h    ;Clear display
        call wr_lcdcmd
        ret

; -----
lcd_home:
        mov abuf, #02h    ;Cursor home
        call wr_lcdcmd
        ret

; -----
lcd_locate:
        anl a, #00001111b ;00-0fh or 40-4fh
        jnb b.0,setb7
        setb acc.6         ;second address 40-4fh
setb7:   setb acc.7         ;data to ddram not to cgram
        mov abuf,a
        call wr_lcdcmd
        ret

; -----
lcd_printstring:
        clr a
        movc a,@a+dptr
        jz  endps          ;if a=0 is ending
        mov abuf,a
        call wr_lcdmdat
        inc dptr
        jmp lcd_printstring
endps:   ret

```

這三行爲控制游標移到第二行開頭的指令，後面會用到，請另外寫成副程式，命名爲 **line2**

這二行控制將累加器內的值顯示於 **LCD**，後面會用到

以下還很多副程式，都不需要動它，這裡就不全部列出來

```

scan_key:    mov keynum,#0      ;key counter=0
             clr key_f          ;initial
;
;row-0
             ;74139=>q0=0
             mov pcbit,#4 ;pc4=0
             call cl_pc
             mov pcbit,#5 ;pc5=0
             call cl_pc
             call rd_pb          ;read pb,data store at pb55
             call check_key
             jbc key_f,have_state
;row-1
             ;74139=>q1=0
             mov pcbit,#4 ;pc4=1
             call st_pc
             mov pcbit,#5 ;pc5=0
             call cl_pc
             call rd_pb          ;read pb,data store at pb55
             call check_key
             jbc key_f,have_state
;row-2
             ;74139=>q2=0
             mov pcbit,#4 ;pc4=0
             call cl_pc
             mov pcbit,#5 ;pc5=1
             call st_pc
             call rd_pb          ;read pb,data store to pb55
             call check_key
             jbc key_f,have_state
;row-3
             ;74139=>q3=0
             mov pcbit,#4 ;pc4=1
             call st_pc
             mov pcbit,#5 ;pc5=1
             call st_pc
             call rd_pb          ;read pb,data store to pb55
             call check_key
             jbc key_f,have_state
;
             ret
;
have_state:

```

```

mov a, keynum
cjne a, #0ffh, is_key
ret

```

修改步驟：

1. 將 `jmp sho_main` 改成 `jmp $`，讓程式停留在此行
不然按完 Mu 鍵顯示「主功能畫面」後又馬上跳回去顯示「初始畫面」，只會一直看到「初始畫面」，會以為按了 Mu 鍵完全沒反應。
2. 修改顯示字串，`main_scr1` 和 `main_scr2` 這 2 行內容依規定要改成 Welcome 和 准考證號碼，另外按 Menu 鍵要顯示的選單，原來附的程式並沒有，請 `main_scr1` 和 `main_scr2` 這 2 行，並把要顯示的內容依題意打好。

解法如下：

將

```

main_scr1:    db    "TEMI-uC", 0
main_scr2:    db    "2223-9560", 0

```

改成

```

main_scr1:    db    "Welcome", 0
main_scr2:    db    "12345678", 0    ; 數字的部份請改成 准考證號碼
menu_scr1:    db    "1. TS   2. NS", 0    ; 這二行字串的內容是固定的
menu_scr2:    db    "3. NI   4. Return", 0

```

3. 後面程式常需要控制游標移到第二行開頭，再寫入資料到 LCM，在此我們順便設計控制游標到第二行開頭的副程式，以方便未來使用。

A. 先找到副程式 `sho_lcm2`，將其完整重新複製一次

```

sho_lcm2:                                ;=====cursor to second line
clr      a                                ;homing to left
setb     b.0                             ;Cursor to second line(1)
call     lcd_locate                      ;b.0=0:line0, b.0=1:line1
call     lcd_printstring
ret

```

B. 更改 `sho_lcm2` 為 `line2`，並將 `call lcd_printstring` 那一行刪除，完成結果如下

```

line2:
clr      a                                ;homing to left
setb     b.0                             ;Cursor to second line(1)
call     lcd_locate                      ;b.0=0:line0, b.0=1:line1
ret

```

4-1-3 重要副程式說明

以下內容會根據考場所提供的原始程式 55-lcm-kb-new.a 內的幾個主要副程式的功能做說明，要完全明白程式的寫法可能要花比較多的時間，但還是必須要了解此副程式的功能，以及使用使副程式需要傳遞的參數是哪些，這樣才可避免修改程式時出錯。

1. scan_key 副程式

此程式主要在掃描周邊板上的鍵盤，考場使用的週邊板上一共有 16 個按鍵，work2 考試所需要所有功能，都需要靠這幾個按鍵做輸入控制。鍵盤的掃描，是以 74139 的輸出控制要選擇哪一行，當 74139 的輸入 B(PC5)、A(PC4) 依序從 00(二進制)變化到 11 時，輸出 Q0~Q3 同一時間會有一組輸出為 0，其餘為 1，依序從左到右掃描，並從 PB3~PB0 將按鍵值讀回。

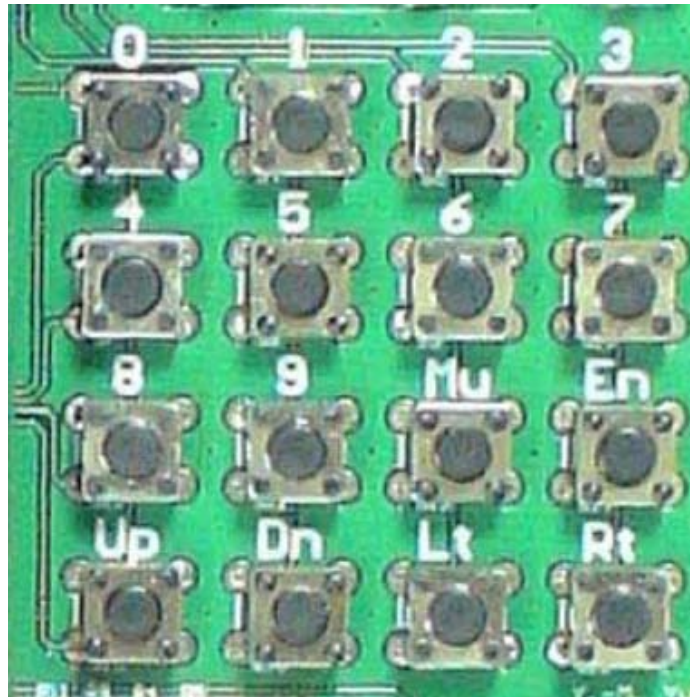


圖 1 自走車上的按鍵

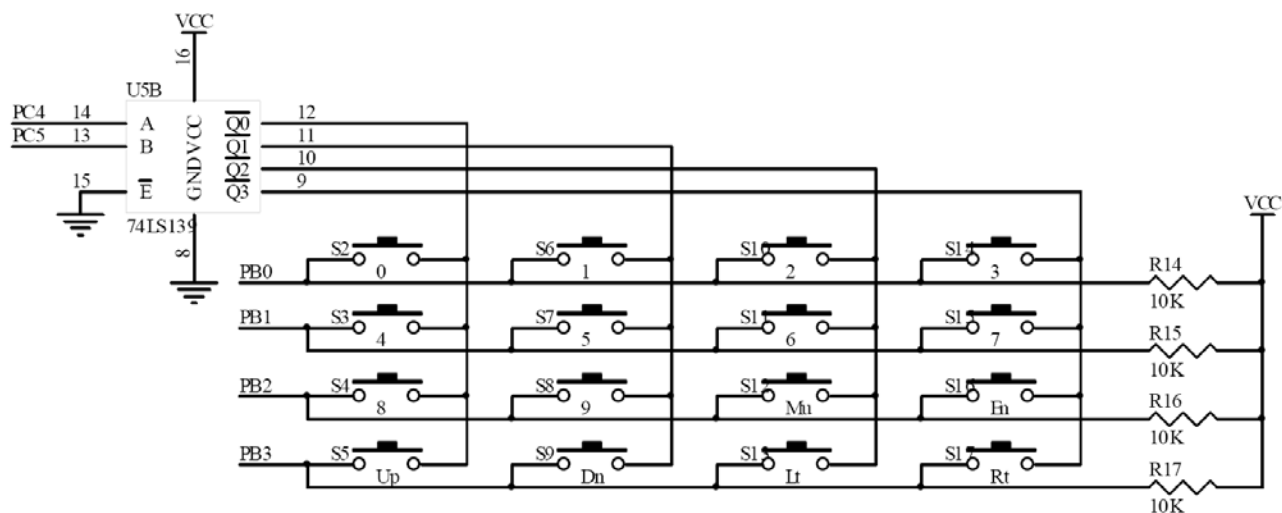


圖 2 按鍵對應的電路圖

(這個電路圖有錯，S2、S3、S4、S9 對應的鍵應該是 0、1、2、3，以下程式僅供參考，如果看不懂就先不要管它，考試不需要修改這部分)

```

scan_key:    mov     keynum,#0 ;key counter=0
             clr     key_f      ;initial
;
;row-0              ;74139=>q0=0
             mov     pcbit,#4   ;pc4=0
             call    cl_pc
             mov     pcbit,#5   ;pc5=0
             call    cl_pc
             call    rd_pb      ;read pb,data store at pb55
             call    check_key
             jbc     key_f,have_state
;row-1              ;74139=>q1=0
             mov     pcbit,#4   ;pc4=1
             call    st_pc
             mov     pcbit,#5   ;pc5=0
             call    cl_pc
             call    rd_pb      ;read pb,data store at pb55
             call    check_key
             jbc     key_f,have_state
;row-2              ;74139=>q2=0
             mov     pcbit,#4   ;pc4=0
             call    cl_pc
             mov     pcbit,#5   ;pc5=1
             call    st_pc
             call    rd_pb      ;read pb,data store to pb55

```

```

        call    check_key
        jbc     key_f, have_state
;row-3          ;74139=>q3=0
        mov     pcbit, #4    ;pc4=1
        call    st_pc
        mov     pcbit, #5    ;pc5=1
        call    st_pc
        call    rd_pb        ;read pb, data store to pb55
        call    check_key
        jbc     key_f, have_state
;
        ret
;
have_state:
        mov     a, keynum
        cjne    a, #0ffh, is_key
        ret
is_key:
        ret
;*****check key whether true and count key number
check_key:
        anl     pb55, #0x0f    ;保留 pb0-pb3，其餘清為 0
        mov     a, pb55
        mov     counter, #4
nextk:
        rrc     a
        jnc     dblchk        ;判斷讀到的是否有 0，則跳去除彈跳
        inc     keynum        ;keynum 數值加 1，計算是哪一個鍵
        djnz    counter, nextk
        ret
;
dblchk:
        mov     keydbl, pb55
        mov     bounce0, #200 ;about delay time=80mS
chk1p:
        mov     bounce1, #250
        djnz    bounce1, $
        djnz    bounce0, chk1p
        call    rd_pb        ;read pb, data store to pb55
        anl     pb55, #0x0f    ;pb0-pb3
        mov     a, pb55
        cjne    a, keydbl, not_key ;判斷第一次和第二次讀到的結果是否相

```

同

```

        setb    key_f      ;have state
        ret
not_key:  setb    key_f
        mov     keynum, #0ffh ;不同就設 keynum = ff
        ret

```

掃描的結果會儲存在 **keynum** 暫存器，按數字鍵"0"~"9"，keynum 儲存的值是 0~9，"Mu"鍵，儲存的值是 10，"En"鍵是 11，"Up""Dn""Lt""Rt"依序是 12、13、14、15。

掃描完按鍵後，最重要的是判斷按那個鍵，所以程式中最常出現的是以下幾行，因為功能常要求按"Mu"鍵回到功能選單

```

        call    scan_key
        mov     a, keynum
        cjne    a, #0ah, loop      ;判斷是否為 Menu 鍵

```

此外，如果很多功能鍵，要判斷不同的鍵做不同的事，則需要進行多次的比較，以專業級認證之初始選單為例：

```

scan:  call     scan_key
       mov      a, keynum
       cjne     a, #1, s1          ;判斷是否為"1"鍵
       call     TS_menu            ;執行 Team Set
       jmp      sho_menu          ;跳回顯示功能表畫面
s1:
       cjne     a, #2, s2          ;判斷是否為"2"鍵
       call     NS_menu            ;執行 Number Set
       jmp      sho_menu
s2:
       cjne     a, #3, s3          ;判斷是否為"3"鍵
       call     NI_menu            ;執行 Number Input
       jmp      sho_menu
s3:
       cjne     a, #4, s4          ;判斷是否為"4"鍵
       jmp      sho_main          ;跳回顯示歡迎畫面(注意跳躍的地方與前 3 個不同)
s4:
       jmp      scan              ;重新掃描按鍵

```

2. 程式中常需要將游標移到第二行開頭顯示目前執行的結果。

考場提供的程式提供了幾行可以控制游標位址的程式

```
clr    a                ;homing to left
setb   b.0              ;Cursor to second line(1)
call   lcd_locate
```

lcd_locate 副程式在控制游標位址，兩個重要參數 b 的第 0 位元決定第幾行，0 第一行，1 第二行，累加器 a 則控制在該行第幾個字，因為常需要將游標移到第二行開頭，建議將這 3 行包成一個副程式，叫做 line2

line2:

```
clr    a                ;a = 0，移到做開頭
setb   b.0              ;游標移到第二行
call   lcd_locate
ret
```

想要游標移到第二行開頭，只需要 call line2 即可。

如果對 LCM 控制很熟，也可以直接寫

```
mov     abuf, #11000000b ;控制游標位址命令
call    wr_lcdcmd        ;寫命令到 LCM
```

3. 在 LCM 上顯示文字

要在 LCM 顯示文字，考場提供了 wr_lcddat 副程式，只要將想要顯示文字的 ASCII 碼寫入 abuf 暫存器，再 CALL 此副程式即可。

例如：

```
mov     abuf, #'0'
call    wr_lcddat
```

這樣就可以在 LCM 上顯示"0"，LCM 尚可顯示的文字和對應的碼如下圖所示：

高四位元

0000 0001 0010 0011 0100 0101 0110 0111 1000 1001 1010 1011 1100 1101 1110 1111

低四位元

0000	CG RAM (1)			0	0	P	\	P						-	9	E	0	P	
0001	CG RAM (2)			!	1	A	Q	a	4					.	7	4	ä	q	
0010	CG RAM (3)			"	2	B	R	b	r					"	イ	ツ	×	ε	θ
0011	CG RAM (4)			#	3	C	S	c	s					」	ウ	テ	ε	ε	ω
0100	CG RAM (5)			\$	4	D	T	d	t					、	エ	ト	ト	μ	Ω
0101	CG RAM (6)			%	5	E	U	e	u					.	オ	ナ	1	ε	Ü
0110	CG RAM (7)			&	6	F	V	f	v					ヲ	カ	ニ	ヨ	ρ	Σ
0111	CG RAM (8)			'	7	G	W	g	w					ア	キ	ヌ	ラ	g	π
1000	CG RAM (1)			(	8	H	X	h	x					ィ	ク	ネ	リ	ル	ア
1001	CG RAM (2)			)	9	I	V	i	v					ウ	ク	リ	ル	リ	ウ
1010	CG RAM (3)			*	:	J	Z	j	z					エ	コ	ハ	レ	ジ	チ
1011	CG RAM (4)			+	:	K	L	k	l					オ	サ	ヒ	ロ	×	ア
1100	CG RAM (5)			,	<	L	*	1	l					カ	シ	フ	ワ	φ	π
1101	CG RAM (6)			-	=	M	J	m	j					ユ	ズ	ハ	ン	ト	÷
1110	CG RAM (7)			.	>	N	^	n	^					ヨ	セ	ホ	ノ	ン	
1111	CG RAM (8)			/	?	O	_	o	+					ウ	リ	マ	ノ	ö	■

LCM 字型編碼表

(此圖版權屬張義和，例說 89S51-C 語言，新文京開發出版社所有)

在 ADC 控制中，要在 LCM 上顯示二位數電壓 3.8V，因為數字和對應的 ASCII 碼剛好差 30H，所以可以將計算得到的數值加上 30H，就可以顯示在 LCM 上了，

例如：

```

mov    a,b
mov    b,#10
div    ab                ;執行 a/b = a(商) ... b(餘數)
add    a,#30h            ;將整數部份轉成 ASCII 碼
mov    abuf,a
call   wr_lcd dat       ;寫入 LCM
mov    abuf,#'.'
call   wr_lcd dat       ;顯示小數點' .'
mov    a,b
add    a,#30h            ;將小數部份轉成 ASCII 碼
mov    abuf,a

```

```
call    wr_lcddat    ; 寫入 LCM
mov     abuf, #'V'
call    wr_lcddat    ; 顯示電壓符號 'V'
```