**;------------------------------------------------------------------**

**;【Define】 pseudo name**

**//用來自定義新的名稱或位置去存取,紀錄其他東西**

**;------------------------------------------------------------------**

**port4 EQU 0E8h //定義port4為位置0E8h**

**shift7LED EQU 0FEh //目的使七字節每次只有一個點亮**

**dispBuf EQU 40h //最左LED為40h,往右為41h、42h、43h**

**keyBuf1 EQU 44h //存取按鍵0～7的輸入，若有輸入顯示0**

**keyBuf2 EQU 45h //存取按鍵8～F的輸入，若有輸入顯示0**

**keyCode EQU 46h //用來存取按鍵碼**

**keyCheck EQU 47h //檢查是否和keyCode一樣，有無取到跳躍信號**

**debCounter EQU 48h //用來檢查是否過了跳躍信號**

**debTimes EQU 60 //用來延遲時間**

**Clear EQU 49h //判斷是否要執行clear**

**Save\_position EQU 4Ah //於副程式Clear\_LED中存R0原本所擁有的內容**

**First\_dot EQU 4Bh //用來記錄輸入的第一個數值小數點前有幾位**

**Sec\_dot EQU 4Ch //用來記錄輸入的第二個數值小數點前有幾位**

**Count\_dot EQU 4Dh //用來數小數點所在位置**

**First\_num EQU 4Eh //4Eh~51h用來存輸入的第一個數值**

**Sec\_num EQU 52h //52h~55h用來存輸入的第二個數值**

**Position EQU 57h //用來記住現在輸入哪個數值哪個位置**

**Caculate1 EQU 58h //用來對齊記住輸入的第一個數值的整數+小數**

**Caculate2 EQU 60h //用來對齊記住輸入的第二個數值的整數+小數**

**Count4 EQU 68h //在副程式Adjust中輔助 ,數數字4個對齊完沒**

**Which\_Big EQU 69h //用來記住輸入數值1大還是輸入數值2大**

**Sum\_tail EQU 71h //6Ah~71h存Sum**

**Borrow BIT 0 //用來在副程式Subtraction中記住是否有CY**

**PrintFrom EQU 72h //用來記住最低位要從哪裡開始輸出數字**

**Blank BIT 1 //用來記住相減後答案是否為0**

**WhereSave EQU 73h //副程式Subtraction中用來記住差值儲存位置**

**What\_num EQU 74h //在Adjust中因為不能MOV @R1,@R0**

**//所以用What\_num在中間過度傳數字過去**

**ORG 0000h //從位置0000h開始**

**;------------------------------------------------------------------**

**;【Main】 program**

**//主程式,包含許多數值的初始化以及各個副程式呼叫運行**

**;------------------------------------------------------------------**

**MOV A, #0FFh //初始化A為0FFh(表無輸入)**

**MOV P2, A //將P2給值為0FFh(表0～7無輸入)**

**MOV P3, A //將P3給值為0FFh(表8～F無輸入)**

**MOV Clear, #0 //先設定Clear為0，為0時不呼叫副程式Clear**

**MOV First\_dot,#0 //初始化第一個數值小數點前有0位數**

**MOV Sec\_dot,#0 //初始化第二個數值小數點前有0位數**

**MOV R1,#First\_do //讓R1記住First\_dot的地址**

**MOV Count\_dot,#0 //初始化數小數點前有0位數**

**ACALL Reset\_all //初始化存取數值1.2和對齊後數值1.2的地方**

**MOV Position,#First\_num //將Position給值為First\_num的地址**

**MOV keyBuf1, #0FFh //用來存取P2，設定為0輸入**

**MOV keyBuf2, #0FFh //用來存取P3，設定為0輸入**

**MOV keyCode,#0FFH //初始化keyCode為無輸入**

**MOV keyCheck,#0FFH //初始化keyCheck為無輸入**

**MOV debCounter,#0 //初始化“數跳躍信號是否結束“為0**

**MOV dispBuf, #14h //初始化最左LED為無字型**

**MOV dispBuf+1, #14h //初始化左二LED為無字型**

**MOV dispBuf+2, #14h //初始化最右LED為無字型**

**MOV dispBuf+3, #14h //初始化右二LED為無字型**

**ACALL Clear\_LED //呼叫副程式Clear\_LED**

**//使一開機顯示空空空0**

**MOV DPTR, #Led\_table //將DPTR設為LED\_table**

**Reset: MOV R0, #dispBuf //初始化R0記住dispBuf的地址(最左LED)**

**MOV A, #shift7LED //初始化A為ShiftLED**

**Loop: MOV port4, A //控制port4只亮一個LED**

**ACALL Covled //呼叫Covled輸出字型**

**SJMP Going //跳至標籤Going繼續**

**Clearing: ACALL Clear\_LED //若Clear=1會跳至這行**

**Going: ACALL Getkeycode //取得按鍵碼**

**ACALL DelaykeyDebounce //確認取的值是否為跳躍信號**

**//若否,則該按鍵之功能為何?**

**MOV R4, Clear //將Clear值給R4**

**CJNE R4,#0, Clearing //若Clear=0，表非按CLR鍵，不用清除**

**INC R0 //換處理下一個LED**

**RL A //換亮下一個LED**

**JB ACC.4, loop //因LED只有四個，若ACC.4=1繼續執行**

**SJMP Reset //若ACC.4=0，要重置**

**;-------------------------------------------**

**;【Subroutine】 Get key code**

**//藉由P2(存取0~7按鍵輸入).P3(存取8~F按鍵輸入)**

**//去利用rotate檢查按鍵碼為何,找到後存於keyCode**

**;-------------------------------------------**

**Getkeycode:**

**PUSH 0E0h //PUSH A保留原本A值**

**MOV A, P2 //將P2存取的按鍵輸入給A**

**MOV keyBuf1, A //keyBuf1存取P2(0~7)按鍵輸入之值**

**MOV A, P3 //將P3存取的按鍵輸入給A**

**MOV keyBuf2, A //keyBuf2存取P2(8~F)按鍵輸入之值**

**;=============================**

**MOV A, keyBuf2 //將 keyBuf2(存取 8~F 輸入)給 A 做以下處理**

**CJNE A,#0FFh,nextcode2 //若 A 不是 FF(零輸入),跳至 nextcode2**

**MOV A,keyBuf1 //將 keyBuf1(存取 0~7 輸入)給 A 做以下處理**

**CJNE A,#0FFh,nextcode1 //若 A 不是 FF(零輸入),跳至 nextcode1**

**SJMP exitgetkeycode //若走到這行代表 0~F 都無輸入,離開副程式**

**nextcode1:**

**MOV R2,#0 //R2 用來數按鍵碼為何**

**nextcheck1:**

**JNB 0E0h,gotkeycode //若 ACC.0=0,跳至 gotkeycode(找到碼)**

**INC R2 //若沒找到按鍵碼,則 R2+1**

**RR A //將 A 的所有 bit 向右轉一個**

**CJNE R2,#8,nextcheck1//若 R2 還沒+到 8,回去繼續檢查 0~7**

**SJMP exitgetkeycode //當 R2=8 時,會走到這步,代表按鍵=7,離開**

**nextcode2:**

**MOV A, keyBuf2 //將 keyBuf2 所存取的按鍵 8~F 輸入值給 A**

**MOV R2, #8 //將 R2 給值為 8 開始找按鍵碼**

**nextcheck2:**

**JNB 0E0h,gotkeycode//若 ACC.0=0,跳至 gotkeycode(找到碼)**

**INC R2 //若沒找到按鍵碼,則 R2+1**

**RR A //將 A 的所有 bit 向右轉一個**

**CJNE R2,#16,nextcheck2//若 R2 還沒+到 16,回去繼續檢查 8~16**

**SJMP exitgetkeycode //當 R2=16 時,會走到這步,代表按鍵=16,離開**

**gotkeycode:**

**MOV keyCode, R2 //若找到 keyCode,將 R2 存的按鍵給 keyCode**

**exitgetkeycode:**

**POP 0E0h //因先前有 PUSH A,所以出副程式前必須 POP A**

**RET //返回**

**;----------------------------------------------------------------**

**;【Subroutine】 Posi\_Nega**

**//用來判定相減數值相減後為正數還是負數**

**//若為正數,記住輸入數值1較大**

**//若為負數,記住輸入數值2較大,且P1.7亮**

**;----------------------------------------------------------------**

**Posi\_Nega:PUSH 0E0h //進副程式先 PUSH A**

**PUSH 1 //因為值會動到,所以PUSH R1**

**PUSH 0 //因為值會動到,所以PUSH R0**

**MOV Which\_big,#1 //先假設數值1較大**

**MOV A,First\_dot //將First\_dot內容給A**

**CJNE A,Sec\_dot,Not\_equal//First\_dot與Sec\_dot若不同**

**//跳至Not\_equal**

**MOV R0,#4Dh //若相等,執行以下動作,給R0值4Dh**

**MOV R1,#51h //給R1值51h**

**Againn: INC R0 //R0++**

**INC R1 //R1++**

**CLR C //清除CY**

**MOV A,@R0 //將R0所存取的地址(數值1)給A**

**SUBB A,@R1 //將數值1與數值2做相減**

**JC Not\_equal //若有carry表示為負數,跳至Not\_equal**

**JNC Examine //若無carry表示為正數,跳至Examine**

**Examine: CJNE A,#0,ExitPN //若A!=0,跳至ExitPN(若正數,出副程式)**

**SJMP Againn //若A=0,跳至Againn繼續**

**Not\_equal:JNC ExitPN //若無carry表示為正數,跳至ExitPN**

**MOV P1,#01111111B //確認為負數,P1.7亮**

**MOV Which\_big,#2 //若為負數,表數值2較大,紀錄2**

**ExitPN: POP 0 //POP R0(配合前面的PUSH)**

**POP 1 //POP R1(配合前面的PUSH)**

**POP 0E0h //POP A(配合前面的PUSH)**

**RET //返回**

**;-------------------------------------------**

**;【Subroutine】 Save**

**//每輸入一個按鍵,就存一個數字Ex:0.123,存0123**

**//而Position記住現在該存數值1的位置還是數值2的位置**

**;-------------------------------------------**

**Save: PUSH 0E0h //進副程式先 PUSH A**

**PUSH 1 //因為值會動到,所以PUSH R1**

**MOV A,R3 //將R3(按鍵碼)給A**

**MOV R1,Position //從Position取現在該存的地址給R1**

**MOV @R1,A //將A存於該地址**

**INC Position //Position++(換下一個位置)**

**POP 1 //POP R1(配合前面的PUSH)**

**POP 0E0h //POP A(配合前面的PUSH)**

**RET //返回**

**;-------------------------------------------**

**;【Subroutine】 DelaykeyDebounce:**

**//檢查得到的keyCode是否為穩定訊號,而非跳躍訊號的keyCode**

**//檢查完後,判斷按鍵碼是否為功能鍵C或功能鍵F**

**//若都不是,則+246去取按鍵碼**

**// 若超過255(按鍵碼可能為A.B.D.E)會有carry,直接重置數值出副程式**

**;-------------------------------------------**

**DelaykeyDebounce:**

**PUSH 0E0H //進副程式前先PUSH A**

**ACALL Delay //呼叫副程式Delay拖延時間**

**INC debCounter //decCounter++**

**MOV A,debCounter //將debCounter給A以利做下一步判斷**

**CJNE A,#debCounter,exit //若A還未達到#debCounter,離開副程式**

**MOV debCounter,#0 //若達到,先重置debCounter=0**

**MOV A,keycode //將得到之keyCode給A做其他判斷**

**CJNE A,#0FFH,Here //若A!=#0FFh跳至Here**

**exit: LJMP exitkey //若A=#0FFh表無輸入,跳至exitkey離開**

**Here: CJNE A,keycheck,There //若A!=keycheck,表示取到跳躍信號**

**//跳至There**

**CJNE A, #0, Not\_zero //按鍵碼A若!=0跳至Not\_zero**

**MOV A, #0 //轉換鍵盤,keyCode=0時,為數字0**

**LJMP Get\_num\_word //跳至Get\_num\_word**

**There: LJMP Again //跳至Again**

**Not\_zero: CJNE A, #4, Not\_one //按鍵碼A若!=4跳至Not\_one**

**MOV keyCode, #1 //若A=4,keyCode給值為1(4號位為按鍵1)**

**LJMP Get\_num\_word //跳至Get\_num\_word去取得字型**

**Not\_one: CJNE A, #5, Not\_two //按鍵碼A若!=5跳至Not\_two**

**MOV keyCode, #2 //若A=5,keyCode給值為2(5號位為按鍵2)**

**LJMP Get\_num\_word //跳至Get\_num\_word去取得字型**

**Not\_two: CJNE A, #6, Not\_three //按鍵碼A若!=6跳至Not\_three**

**MOV keyCode, #3 //若A=6,keyCode給值為3(6號位為按鍵3)**

**LJMP Get\_num\_word //跳至Get\_num\_word去取得字型**

**Not\_three: CJNE A ,#8, Not\_four //按鍵碼A若!=8跳至Not\_four**

**MOV keyCode, #4 //若A=8,keyCode給值為4(8號位為按鍵4)**

**LJMP Get\_num\_word //跳至Get\_num\_word去取得字型**

**Not\_four: CJNE A, #9, Not\_five //按鍵碼A若!=9跳至Not\_five**

**MOV keyCode, #5 //若A=9,keyCode給值為5(9號位為按鍵5)**

**LJMP Get\_num\_word //跳至Get\_num\_word去取得字型**

**Not\_five: CJNE A, #0Ah, Not\_six //按鍵碼A若!=0Ah跳至Not\_six**

**MOV keyCode, #6 //若A=0Ah,keyCode給值為6**

**//(A號位為按鍵6)**

**LJMP Get\_num\_word //跳至Get\_num\_word去取得字型**

**Not\_six: CJNE A, #0Ch, Not\_seven //按鍵碼A若!=0Ch跳至Not\_seven**

**MOV keyCode, #7 //若A=0Ch,keyCode給值為7**

**//(C號位為按鍵7)**

**SJMP Get\_num\_word //跳至Get\_num\_word去取得字型**

**Not\_seven:CJNE A,#0Dh,Not\_eight //按鍵碼A若!=0Dh跳至Not\_eight**

**MOV keyCode, #8 //若A=0Dh,keyCode給值為8**

**//(D號位為按鍵8)**

**SJMP Get\_num\_word //跳至Get\_num\_word去取得字型**

**Not\_eight:CJNE A,#0Eh,Not\_nine //按鍵碼A若!=0Eh跳至Not\_nine**

**MOV keyCode, #9 //若A=0Eh,keyCode給值為9**

**//(E號位為按鍵9)**

**SJMP Get\_num\_word //跳至Get\_num\_word去取得字型**

**Not\_nine: CJNE A, #1, Not\_dot //按鍵碼A若!=1跳至Not\_dot**

**MOV A, R3 //若A=1,將R3(最後一次數字值)給A**

**ADD A,#10 //A=A+10(目的為取小數Table)**

**MOV dispBuf+3,A //將A值給dispBuf+3做輸出**

**MOV @R1,Count\_dot //@R1(First或Sec\_dot)存Count\_dot**

**SJMP Res //跳至Res重設按鍵初值**

**Not\_dot: CJNE A ,#2, Not\_clear //按鍵碼A若!=2跳至Not\_cear**

**ACALL Clear\_LED //呼叫副程式Clear\_LED進行Clear**

**SJMP Res //跳至Res重設按鍵初值**

**Not\_clear:CJNE A,#0Fh,Not\_op //按鍵碼A若!=0Fh跳至Not\_op**

**MOV R4,First\_dot //將First\_dot值給R4**

**CJNE R4,#0,Have\_dot //若R4!=0,跳至Have\_dot**

**MOV First\_dot,Count\_dot //若R4=0,將Count\_dot值給First\_dot**

**MOV R1,#Sec\_dot //R1記住Sec\_dot的地址**

**Have\_dot: MOV R1,#Sec\_dot //R1記住Sec\_dot的地址**

**MOV Position,#Sec\_num //Position記住Sec\_num的地址**

**ACALL Clear\_LED //呼叫Clear副程式**

**SJMP Res //跳至Res重設按鍵初值**

**Not\_op: CJNE A,#3,Res //按鍵碼A!=3,跳至Res(最後1個按鍵檢查)**

**MOV R4,Sec\_dot //將Sec\_dot值給R4**

**CJNE R4,#0,Have\_secdot //若R4!=0,跳至Have\_secdot**

**MOV Sec\_dot,Count\_dot //若R4=0,將Count\_dot值給Sec\_dot**

**Have\_secdot:ACALL Posi\_Nega //呼叫副程式Posi\_Nega將兩數值比大小**

**PUSH 1 //因為會變更R1值,先PUSH R1**

**ACALL Adjust //呼叫副程式Adjust整理並存取對齊後數值**

**ACALL Subtraction //呼叫副程式Subtraction去做減法運算**

**ACALL ConvertBCD //因會有借位問題,呼叫ConvertBCD做轉換**

**ACALL Print\_who //呼叫副程式Print\_who去找輸出哪4位數**

**ACALL Oh\_Ya\_Print //呼記副程式Oh\_Ya\_Print做輸出**

**POP 1 //因前面有PUSH R1,因此這裡要POP R1**

**MOV R1,#First\_dot //重設R1存取First\_dot的地址**

**MOV First\_dot,#0 //重設First\_dot為0**

**MOV Sec\_dot,#0 //重設Sec\_dot為0**

**ACALL Reset\_all //呼叫副程式Reset\_all**

**//重設存取數值的各個地址內容為0**

**MOV Position,#First\_num //重設Position存First\_num的地址**

**SJMP Res //跳至Res重設按鍵初值**

**Get\_num\_word:MOV R3, keyCode //若按鍵為數字,將keyCode給R3存**

**ACALL Save //呼叫副程式Save將數字存起來**

**INC Count\_dot //Count\_dot++**

**MOV Clear,#0 //重設Clear值為0(不做Clear)**

**ACALL Displaykeypattern //呼叫Displaykeypattern做輸出顯示**

**Res: MOV keycode,#0FFH //重設keyCode值為0FFh**

**MOV keycheck,#0FFH //重設keyCheck值為0FFh**

**SJMP exitkey //跳至exitkey**

**Again: MOV keycheck,keycode //若按鍵為跳躍信號,keyCode給keyCheck**

**exitkey: POP 0E0H //因先前有PUSH A,出副程式前必須POP A**

**RET //返回**

**;-------------------------------------------**

**;【Subroutine】 Adjust**

**;用來將兩數值存到對齊後存到新的8byte空間**

**;以置中的方式對齊個位數**

**;-------------------------------------------**

**Adjust: PUSH 0E0h //進入副程式前先PUSH A**

**PUSH 1 //因為會用到R1,因此先PUSH R1**

**PUSH 0 //因為會用到R0,因此先PUSH R0**

**PUSH 2 //因為會用到R2,因此先PUSH R2**

**MOV R2,#1 //R2用來記住現在處理數值1還是數值2**

**MOV A,First\_dot //將First\_dot(記住數值1小數點前有幾位)給A**

**MOV R0,#4Eh //將R0給值為4Eh(數值1 儲存位置4Eh~51h)**

**MOV Count4,#0 //給Count4值0,輸入最多4個**

**//(數數字4個對齊完沒)**

**CJNE A,#4,Not4 //小數點前有4位數?**

**MOV R1,#58h //若4位數,從58h存(58h~5Bh存對齊後數值1)**

**SJMP Enter\_num //跳至Enter\_num存對齊的數值,**

**//(存於58.59.5A.5B)**

**Not4: CJNE A,#3,Not3 //小數點前有3位數?**

**MOV R1,#59h //若3位數,從59h存(59h~5Ch存對齊後數值1)**

**SJMP Enter\_num //跳至Enter\_num存對齊的數值,**

**//(存於59.5A.5B.5C)**

**Not3: CJNE A,#2,Not2 //小數點前有2位數?**

**MOV R1,#5Ah //若2位數,從5Ah存(5Ah~5Dh存對齊後數值1)**

**SJMP Enter\_num //跳至Enter\_num存對齊的數值,**

**//(存於5A.5B.5C.5D)**

**Not2: CJNE A,#1,Exit\_adjust//小數點前有1位數?**

**MOV R1,#5Bh //若1位數,從5Bh存(5Bh~5Eh存對齊後數值1)**

**SJMP Enter\_num //跳至Enter\_num存對齊的數值,**

**//(存於5B.5C.5D.5E)**

**//以下迴圈為處理數值2**

**Num2: MOV A,Sec\_dot //將Sec\_dot(記住數值2小數點前有幾位)給A**

**MOV R0,#52h //將R0給值為52h(數值2 儲存位置52h~55h)**

**MOV Count4,#0 //給Count4值0,輸入最多4個**

**//(數數字4個對齊完沒)**

**CJNE A,#4,Not44 //小數點前有4位數?**

**MOV R1,#60h //若4位數,從60h存(60h~63h存對齊後數值2)**

**SJMP Enter\_num //跳至Enter\_num存對齊的數值,**

**//(存於60.61.62.63)**

**Not44: CJNE A,#3,Not33 //小數點前有3位數?**

**MOV R1,#61h //若3位數,從61h存(61h~64h存對齊後數值2)**

**SJMP Enter\_num //跳至Enter\_num存對齊的數值,**

**//(存於61.62.63.64)**

**Not33: CJNE A,#2,Not22 //小數點前有2位數?**

**MOV R1,#62h //若2位數,從62h存(62h~65h存對齊後數值2)**

**SJMP Enter\_num //跳至Enter\_num存對齊的數值,**

**//(存於62.63.64.65)**

**Not22: CJNE A,#1,Exit\_adjust//小數點前有1位數?**

**MOV R1,#63h //若1位數,從63h存(63h~66h存對齊後數值2)**

**SJMP Enter\_num //跳至Enter\_num存對齊的數值,**

**//(存於63.64.65.66)**

**//因小數點前至少有1位,R1一定存有對齊位置**

**// R1一定存有對齊位置的起始**

**Enter\_num:MOV What\_num,@R0 //給What\_num值為@R0(原輸入數字)**

**MOV @R1, What\_num //再將該數字存到@R1(R1記對齊要從哪裡開始存)**

**INC Count4 //Count4++(已存入幾個數值)**

**INC R1 //R1++(換下一個新地址)**

**INC R0 //R0++(換下一個舊地址)**

**MOV A,Count4 //將Count4值給A去做判定**

**CJNE A,#4,Enter\_num //A(Count4)是否等於4**

**INC R2 //若A=4,表數值對齊轉換完畢,R2++**

**CJNE R2,#3,Num2 //R2是否等於3若等於3表兩數值對齊轉換完畢**

**Exit\_adjust: POP 2 //因先前有PUSH R2,所以要POP R2**

**POP 0 //因先前有PUSH R0,所以要POP R0**

**POP 1 //因先前有PUSH R1,所以要POP R1**

**POP 0E0h //因先前有PUSH A,所以要POP A**

**RET //返回**

**;-----------------------------------------------**

**;【Subroutine】 Subtraction**

**//將對齊後的兩數值從最低位(最右邊)開始用SUBB連續相減**

**;-----------------------------------------------**

**Subtraction: PUSH 0E0h //進入副程式前先PUSH A**

**PUSH 1 //因為會用到R1,因此先PUSH R1**

**PUSH 0 //因為會用到R0,因此先PUSH R0**

**PUSH 2 //因為會用到R2,因此先PUSH R2**

**MOV WhereSave,#Sum\_tail //WhereSave用來記住結果的最低位**

**MOV A,Which\_Big //將A給值為Which\_big**

**CJNE A,#1,Bigger2 //A!=1,表輸入數值2較大,跳至Bigger2**

**MOV R1,#5Fh //R1用來記住轉換後數值1的最低位的位置(大)**

**MOV R0,#67h //R0用來記住轉換後數值2的最低位的位置(小)**

**SJMP Howmany //跳至Howmany進行減法運算**

**Bigger2: MOV R1,#67h // R1用來記住轉換後數值2的最低位的位置(大)**

**MOV R0,#5Fh // R0用來記住轉換後數值1的最低位的位置(大)**

**Howmany: CLR Borrow //先清楚Borrow**

**Cacu: MOV C,Borrow //在將清楚後的Borrow值給C**

**MOV A,@R1 //將R1所存的地址去取內容給A**

**SUBB A,@R0 //A-@R0,相減之後的數值會存在A**

**PUSH 1 //因為會用到R1,所以先PUSH R1**

**MOV Borrow,C //怕CY值在下次減法前受影響,先用Borrow存**

**MOV R1,WhereSave //讓R1記住相減後結果要存的位置**

**MOV @R1,A //將相減後的值A給@R1存**

**POP 1 //POP R1回來**

**DEC R0 //R0--,換下一位數**

**DEC R1 //R1--,換下一位數**

**DEC WhereSave //WhereSave,換下一個位置**

**MOV R2,WhereSave //將WhereSave值給R2**

**CJNE R2,#69h,Cacu //R2!=69h,跳至Cacu繼續算(結果存6Ah~71h)**

**Over: POP 2 //因先前有PUSH R2,所以要POP R2**

**POP 0 //因先前有PUSH R0,所以要POP R0**

**POP 1 //因先前有PUSH R1,所以要POP R1**

**POP 0E0h //因先前有PUSH A,所以要POP A**

**RET //返回**

**;-------------------------------------------**

**;【Subroutine】 ConvertBCD**

**//將相減後的數值，可能會出現F7~FF之間的數值**

**//需要將該數值and 0Fh後減06h或是直接做對應取值**

**//我是直接對應取值,Ex:FF=9**

**;-------------------------------------------**

**ConvertBCD:PUSH 0E0h //進入副程式前先PUSH A**

**PUSH 1 //因為會用到R1,因此先PUSH R1**

**PUSH 2 //因為會用到R2,因此先PUSH R2**

**MOV R1,#Sum\_tail //R1用來記住結果的最低位(Sum\_tail)**

**NotFinish:MOV A,@R1 //將@R1(結果數值)給A**

**MOV R2,A //再將結果數值由A給R2**

**ANL A,#0F0h //A去and F0,留下high byte**

**CJNE A,#0F0h,NoF //若A!=0F0h,跳至NoF,表數值不用轉換**

**MOV A,R2 //若A=0F0h,表數值要做轉換,將原數值R2給A**

**ANL A,#0Fh //之後A再去and 0Fh,留下low byte**

**SUBB A,#6 //將A的low byte減6**

**MOV @R1,A //將轉換後數值給@R1(該結果所要存的位置)**

**NoF: DEC R1 //R1--,換下一個位置**

**CJNE R1,#69h,NotFinish //R1!=69h,表尚未處理完,跳至NotFinish**

**POP 2 //因先前有PUSH R2,所以要POP R2**

**POP 1 //因先前有PUSH R1,所以要POP R1**

**POP 0E0h //因先前有PUSH A,所以要POP A**

**RET //返回**

**;-------------------------------------------**

**;【Subroutine】 Print\_who**

**//減法執行完且數值轉換完後,6Ah~71h存有結果**

**//從最低位(71h)開始找不為0的數**

**//PrintFrom記住該從哪一位開始印(結果的最低位)**

**;-------------------------------------------**

**Print\_who:PUSH 0E0h //進入副程式前先PUSH A**

**PUSH 1 //因為會用到R1,因此先PUSH R1**

**CLR Blank //先清空Blank,運算後假設結果不為0**

**MOV R1,#Sum\_tail //R1用來記住結果的最低位(Sum\_tail)**

**Gogo: MOV A,@R1 //將@R1(結果數值)給A**

**CJNE A,#0,Print //若找到最低位結果數值不為0,跳至Print處理**

**DEC R1 //R1--,換下一個位數檢查**

**CJNE R1,#6Dh,Gogo //R1!=6Dh(個位數),跳至Gogo繼續找**

**MOV PrintFrom,R1 //若R1=6Dh,從個位數開始印,PrintFrom記6Dh**

**Check: CJNE @R1,#0,Exit\_print //@R1!=0,表結果不為0,跳至Exit\_print**

**DEC R1 //若R1=0,繼續檢查其他位數,R1--**

**CJNE R1,#69h,Check //若R1!=69h,跳至Check繼續檢查**

**SETB Blank //若R1=69h,表差值為0,Blank設為1**

**SJMP Exit\_print //跳至Exit\_print**

**Print: MOV PrintFrom,R1 //若找到要從哪裡印,將R1位置給PrintFrom記**

**Exit\_print:POP 1 //因先前有PUSH R1,所以要POP R1**

**POP 0E0h //因先前有PUSH A,所以要POP A**

**RET //返回**

**;-------------------------------------------**

**;【Subroutine】 Oh\_Ya\_Print**

**//此副程式在Print\_who找到由誰開始印時**

**//因PrinrFrom記住要從哪裡開始印**

**//在這裡做輸出顯示**

**;-------------------------------------------**

**Oh\_Ya\_Print:**

**PUSH 0E0h //進入副程式前先PUSH A**

**PUSH 0 //因為會用到R0,因此先PUSH R0**

**JNB Blank,Sum\_not0 //先看看差值是否為0,若非0，跳至Sum\_not0**

**MOV Port4,#11110111B//若差值為0,只有LED最右會亮**

**MOV P1,#0FFh //字型輸出顯示給0FFh(全關)**

**Mov dispBuf, #14h //給左一LED#14h無字型**

**Mov dispBuf+1, #14h //給左二LED#14h無字型**

**Mov dispBuf+2, #14h //給左三LED#14h無字型**

**MOV dispBuf+3, #0 //給右一LED#0字型0**

**Sum\_not0: PUSH 1 //因為會用到R1,所以先PUSH R1**

**ACALL Delete\_zero //呼叫副程式Delete\_zero清除高位無用處之0**

**MOV R1, PrintFrom //將PrintFrom值給R1**

**MOV dispBuf+3,@R1 //取R1內容為地址,取該地址內容給dispBuf+3**

**DEC R1 //R1--,換下一位**

**CJNE R1,#6Dh,NoDot1 //R1!=6D,跳至NoDot1**

**MOV A,@R1 //若R1=6D,將@R1值給A**

**ADD A,#10 //A=A+10**

**MOV @R1,A //再將A值還給@R1**

**NoDot1: MOV dispBuf+2,@R1 //取R1內容為地址,取該地址內容給dispBuf+2**

**DEC R1 //R1--,換下一位**

**CJNE R1,#6Dh,NoDot2 //R1!=6D,跳至NoDot2**

**MOV A,@R1 //若R1=6D,將@R1值給A**

**ADD A,#10 // A=A+10**

**MOV @R1,A //再將A值還給@R1**

**NoDot2: MOV dispBuf+1,@R1 //取R1內容為地址,取該地址內容給dispBuf+1**

**DEC R1 //R1--**

**CJNE R1,#6Dh,NoDot3 //R1!=6D,跳至NoDot3**

**MOV A,@R1 //若R1=6D,將@R1值給A**

**ADD A,#10 // A=A+10**

**MOV @R1,A //再將A值還給@R1**

**NoDot3: MOV dispBuf,@R1 //取R1內容為地址,取該地址內容給dispBuf**

**POP 1 // POP R1回來**

**Show: MOV R0,#40h //R0給值為40h**

**ACALL Covled //呼叫副程式Covled做輸出顯示**

**MOV R0,#41h //R0給值為41h**

**ACALL Covled //呼叫副程式Covled做輸出顯示**

**MOV R0,#42h //R0給值為42h**

**ACALL Covled //呼叫副程式Covled做輸出顯示**

**MOV R0,#43h //R0給值為43h**

**ACALL Covled //呼叫副程式Covled做輸出顯示**

**Exit\_oh: CLR Blank //重設Blank為0**

**POP 0 //因先前有PUSH R0,所以要POP R0**

**POP 0E0h //因先前有PUSH A,所以要POP A**

**RET //返回**

**;-------------------------------------------**

**;【Define】 Delete\_zero**

**//將高位沒用的0刪除**

**//Ex0012.3刪除最左邊的兩個0**

**;-------------------------------------------**

**Delete\_zero: PUSH 0E0h //進入副程式前先PUSH A**

**PUSH 1 //因為會用到R1,因此先PUSH R1**

**MOV R1,#6Ah //將R1給值為6Ah,結果的最高位**

**Againnn: CJNE @R1,#0,Exit\_delete //若R1!=0,跳至Exit\_delete**

**MOV @R1,#0FFh //若R1=0,將@R1給值為0FFh(無字型)**

**INC R1 //R1++,換下一位**

**CJNE R1,#6Dh,Againnn//若R1!=6Dh表尚未處理完,跳至Againnn繼續**

**Exit\_delete:POP 1 //因先前有PUSH R1,所以要POP R1**

**POP 0E0h //因先前有PUSH A,所以要POP A**

**RET //返回**

**;-------------------------------------------**

**;【Subroutine】 time delay**

**//用在時間延遲,透過迴圈去浪費時間造成delay效果**

**;-------------------------------------------**

**Delay:**

**mov r4, #1 //R4給值1**

**delay0: mov r5, #2 //R5給值2**

**delay1: mov r6, #100 //R6給值100**

**delay2: mov r7, #100 //R7給值100**

**delay3: djnz r7, delay3 //若R7!=0,跳至delay3**

**djnz r6, delay2 //若R6!=0,跳至delay2**

**djnz r5, delay1 //若R5!=0,跳至delay1**

**djnz r4, delay0 //若R4!=0,跳至delay0**

**ret //返回**

**;-------------------------------------------**

**;【Subroutine】 7-seg LED pattern conversion**

**//主要是向Led\_teble取字形並交給P0去做顯示**

**;-------------------------------------------**

**Covled:**

**PUSH 0E0h //進入副程式前先PUSH A**

**mov a, @R0 //取R0內容當地址,到該地址取內容給A**

**movc a, @a+dptr //將dptr+A(偏移量)到Led\_table取字型**

**mov p0, a //取到的字形傳至P0做輸出**

**POP 0E0h //因先前有PUSH A,所以要POP A**

**Ret //返回**

**;------------------------------------------------**

**;【Subroutine】 Reset\_all**

**//將所有存數值的地方都重設為0**

**//輸入數值1存於:4Eh~51h,輸入數值2存於:52h~55h**

**//對齊後數值1存於58h~5Fh,對齊後數值2存於60h~67h**

**//運算後結果(差值)存於6Ah~71h,(以上空間都要重設為0)**

**;------------------------------------------------**

**Reset\_all:PUSH 0E0h //進入副程式前先PUSH A**

**PUSH 1 //PUSH R1**

**MOV R1,#4Dh //給R1為4Dh(為了+1=4Eh,重設輸入數值1.2)**

**Initial\_1:INC R1 //R1++,換下一位**

**MOV @R1,#0 //取R1內容當地址，該地址內容設為0**

**CJNE R1,#55h,Initial\_1 //若R1!=56h，跳至Initial\_1繼續初始化**

**MOV R1,#57h //給R1為57h(為了+1=58h,重設對齊數值1.2)**

**Inintial\_2:INC R1 //R1++,換下一位**

**MOV @R1,#0 //取R1內容當地址，該地址內容設為0**

**CJNE R1,#67h,Inintial\_2 //若R1!=67h，跳至Initial\_2繼續初始化**

**MOV R1,#69h //給R1為69h(為了+1=58h,重設結果差值數值)**

**Inintial\_3:INC R1 // R1++,換下一位**

**MOV @R1,#0 //取R1內容當地址，該地址內容設為0**

**CJNE R1,#71h, Inintial\_3//若R1!=71h，跳至Initial\_3繼續初始化**

**POP 1 //因先前有PUSH R1,所以要POP R1**

**POP 0E0h //因先前有PUSH A,所以要POP A**

**RET //返回**

**;-------------------------------------------**

**;【Subroutine】 Clear\_LED**

**//Clear鍵按下後,需顯示空空空0**

**//其中有些數值也要隨之重設**

**;-------------------------------------------**

**Clear\_LED:**

**PUSH 0E0h //進入副程式前先PUSH A**

**MOV Clear, #0FFh //Clear給值為0FFh代表啟動Clear**

**MOV P1,#0FFh //將P1給值為0FFh,表指示燈全關**

**MOV Port4,#11110111B//Port4給值為1111011B,表只亮最右的LED**

**MOV Count\_dot,#0 //將Count\_dot給值為0,表小數點前有0位**

**MOV A,#3 //將A給值為3**

**MOV dispBuf+3, #0 //dispBuf++3給值為0**

**MOV Save\_position,R0//先將R0值給Save\_position儲存原值**

**MOV R0,#43h //R0給值為43h**

**ACALL Covled //呼叫副程式Covled做顯示輸出**

**MOV R0,Save\_position//再將原本R0的值還給R0**

**Mov dispBuf, #14h //給左一LED#14h無字型**

**Mov dispBuf+1, #14h //給左二LED#14h無字型**

**Mov dispBuf+2, #14h //給右二LED#14h無字型**

**Mov dispBuf+3, #14h //給右一LED#14h無字型**

**POP 0E0h //因先前有PUSH A,所以要POP A**

**Ret //返回**

**;-------------------------------------------**

**;【Subroutine】Displaykeypattern**

**//將LED座向左傳遞的顯示**

**;-------------------------------------------**

**Displaykeypattern:**

**PUSH 0E0H //進入副程式前先PUSH A**

**MOV dispBuf ,dispBuf+1 //先將左二LED數值給左一LED**

**MOV dispBuf+1,dispBuf+2 //再將左三LED數值給左二LED**

**MOV dispBuf+2,dispBuf+3 //然後左一LED數值給右二LED**

**MOV dispBuf+3,KeyCode //將取得隻按鍵碼給右一的LED**

**POP 0E0H //因先前有PUSH A,所以要POP A**

**RET //返回**

**;-------------------------------------------**

**;【Fixed data】 for table lookup**

**//用在事先存取好使LED亮0~9,0.~9.時該給何值  
//方便直接取值出去做輸出**

**;-------------------------------------------**

**LED\_table:DB 0C0h //當偏移值為0時,取0c0h,字型0,回去給A**

**DB 0F9h //當偏移值為1時,取0F9h,字型1,回去給A**

**DB 0A4h //當偏移值為2時,取0A4h,字型2,回去給A**

**DB 0B0h //當偏移值為3時,取0B0h,字型3,回去給A**

**DB 99h //當偏移值為4時,取99h,字型4,回去給A**

**DB 92h //當偏移值為5時,取92h,字型5,回去給A**

**DB 82h //當偏移值為6時,取82h,字型6,回去給A**

**DB 0D8h //當偏移值為7時,取0D8h,字型7,回去給A**

**DB 80h //當偏移值為8時,取80h,字型8,回去給A**

**DB 90h //當偏移值為9時,取90h,字型9,回去給A**

**DB 40h //當偏移值為10時,取40h,字型0.,回去給A**

**DB 79h //當偏移值為11時,取79h,字型1.,回去給A**

**DB 24h //當偏移值為12時,取24h,字型2.,回去給A**

**DB 30h //當偏移值為13時,取30h,字型3.,回去給A**

**DB 19h //當偏移值為14時,取19h,字型4.,回去給A**

**DB 12h //當偏移值為14時,取12h,字型5.,回去給A**

**DB 2h //當偏移值為15時,取2h,字型6.,回去給A**

**DB 58h //當偏移值為16時,取58h,字型7.,回去給A**

**DB 0h //當偏移值為17時,取0h,字型8.,回去給A**

**DB 10h //當偏移值為18時,取10h,字型9.,回去給A**

**DB 0FFh //當偏移值為19時,取0FFh,字型無,回去給A**

**END //結束程式**