

计算机工程学院

**汇编语言与接口技术**

**实验报告**

学年学期：  **2022 - 2023 学年 第1学期**

实验名称：  **实验七 算术运算程序设计**

班级： 计算2114

学号： 202121331104

姓名： 庄佳强

日期： 2022.11.26

成绩：

**一、 实验要求**

1. 掌握数码管显示方法
2. 掌握软件延时方法
3. 掌握键盘扫描及去抖动方法

**二、 实验设计**

**1.整体思路**

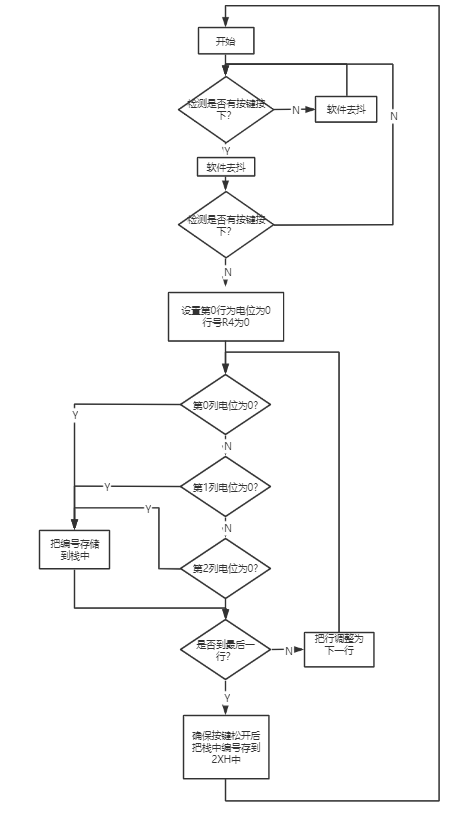
先设计好矩阵按键部分。通过先判断是否有按键按下，在通过控制变量，先控制其中一行，在一次检测这一行中的列电位信号是否为0，否则跳到下一行了后再检测列。检测处理后把行号加上列号就是这个数。全部按键检测完后跳到LED显示部分。

LED显示中,把检测后的数依次存到20H~27H，依次为LED从左到右显示。

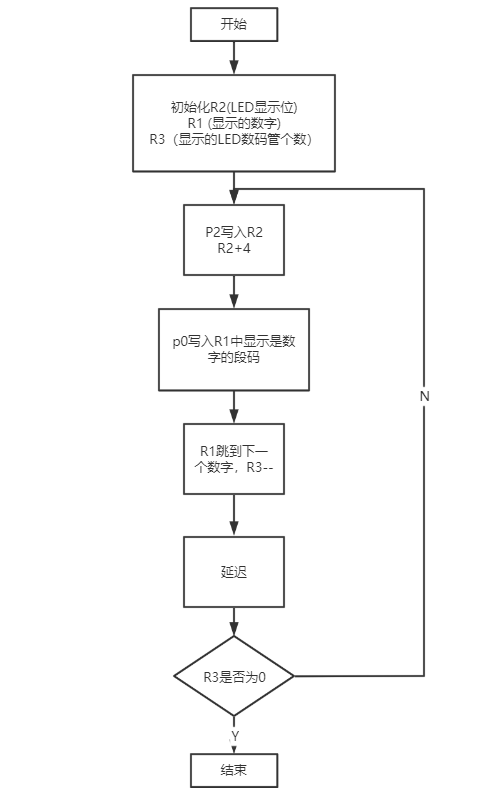
LED显示中，先把p2中写入显示位，再把显示位移到下一个LED灯，在p0中写入要显示的数据段号，延迟后再循环到下一位。

1. **流程图**

**矩阵按键：**



**LED数码管显示部分：**



**3.主要模块设计思路及分析**

1. 矩阵按键模块

(1)首先判别整个键盘有无按键按下,方法为单片机驱动列线P1.0～P13输出全“0”,然后读行线P1.4～P1.7的状态若为全“1”,则键盘上没有闭合键,若P1.4～P1.7不全为“1”,则有键按下。

(2）去除键的抖动。当判别出可能有键按下时,软件延时一段时间(10ms左右)再判别键盘的状态,若仍有键闭合，则认为键盘上有确定的键按下,否则是键抖动。

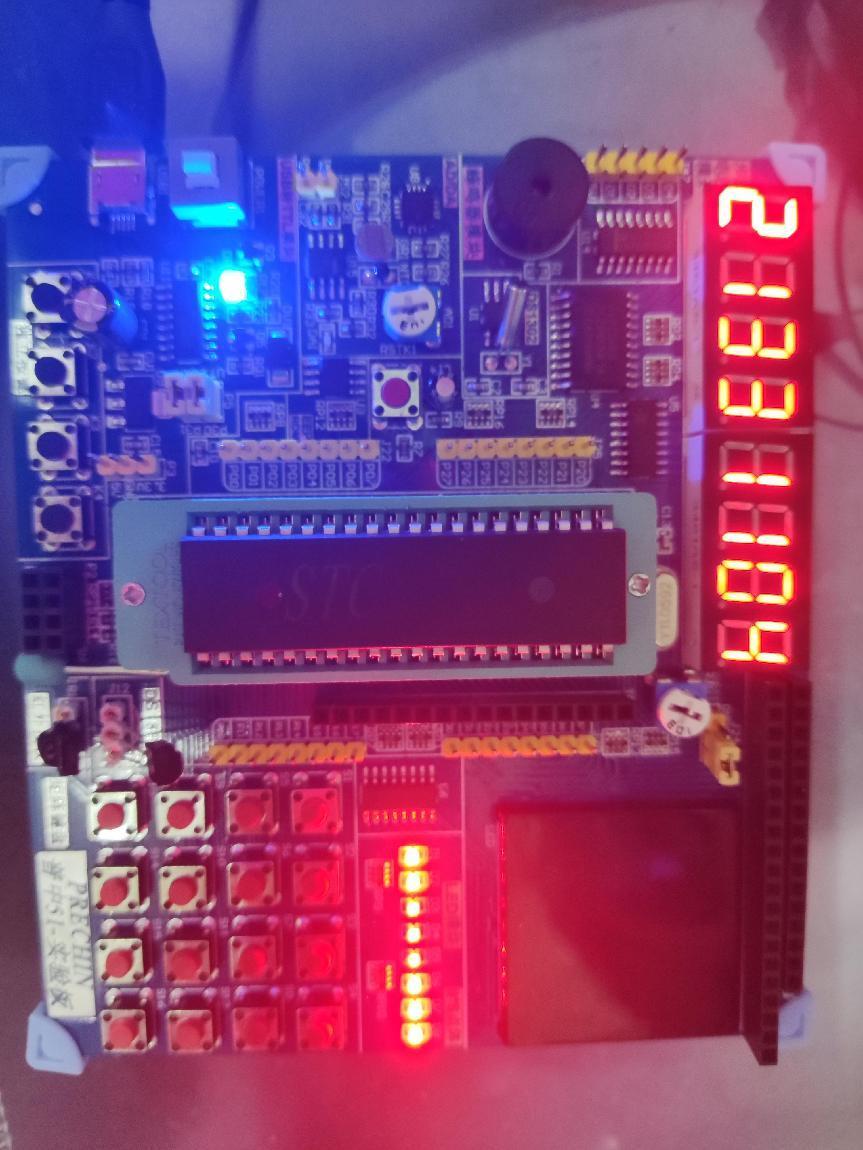
(3）求出按下键的键号,16个按键键号依次为0，1,…,15。各行的首键号分别为0,4,8,12,列号;依次为0，1,2,3。行通过上拉电阻接+5V,当无键按下时，行线为高电平,当有按键按下时,对应的行线与列线短接，行线的电平将由此行线相连的列线电平决定。如果把行线设置为单片机的输入口线，列线设置为单片机的输出口线,则按键号的识别过程是:先令0列线P1.3少与低电平“0”,其余3根列线都为高电平,逐行检查行线状态。若P1.4～P1.7中有低电平，表示有按键按下1，跳到求键号。若P1.4～P1.7都为高电平上没有按键闭合,而P1.3这一行中没有按键闭合，接着再使P1.2为低电平以此类推。求出按下键的键号N为:N=行描(只有一列为低)。存到栈中。

(4）判闭合键是否松开,如果松开则将键号送键的一次闭合仅作—次键功能处理。

1. LED显示模块

先初始化个个数值，再把p2中写入显示位，在把存储在20H~27H中数写入到p0中，通过延时来实现视觉暂留效果。再把显示位+4来跳到下一个LED数码管你显示位2XH加一来跳到下一个数字。

1. **实现效果**

****

1. **总结**
2. 本次实验和书上高度相同，自需要更改一些就可以完成。

2. 一开始写完后发现LED数码管的动态显示有问题，无法显示完全。后面通过询问同学得知本实验的LED动态控制和书上不相同，是通过74HC138译码器来控制，所以p2每次加4来到下一个LED数码管。

3. 一开始在延迟中加显示的时候，一开始一直无法显示按下的按键，通过反复遇到代码后得知是寄存器R7到R5复用了，导致一直没有循环出来，后面变成内部存储后就可以了。

**附录：**

ORG 0000H

AJMP KEY1

ORG 0100H

KEY1:

MOV R0,#20H //存储号码的存储地址(从左到右)

KEY:

LCALL JCK

JNZ KH //当有按键按下时跳转

LCALL D10ms //软件去抖

AJMP KEY //无按键按下时，进行循环检测

KH:

LCALL D10ms

LCALL JCK //再次检测确保按键按下，

JNZ AKH //A中不为0，跳到AKH

AJMP KEY

AKH:

MOV R2,#0F7H //11110111 把列扫描p1.3设置为0

MOV R4,#00H //将第0列号 送到R4暂存

SK:

MOV P1,R2 //把R2中送到P1，实现p1的行电位低，

L0:

JB P1.7,L1 //判断行线电平，p1.4==1表示0行没有按键按下转到L1.检测第一行。

MOV A,#00H //将第一行的首键号送到Acc

AJMP LK

L1:

JB P1.6,L2

MOV A,#04H

AJMP LK

L2:

JB P1.5,L3

MOV A,#08H

AJMP LK

L3:

JB P1.4,NEXT

MOV A,#0cH

AJMP LK

LK:

ADD A,R4 //累加

PUSH Acc

NEXT:

INC R4 //把行数加一

MOV A,R2

JNB ACC.0,Light //判断扫描是否完成，完成跳转到Light处理 中

RR A //左移到一个列进行检测

MOV R2,A

AJMP SK

JCK:

MOV P1,#0FH //将p1的0~4设置为1，当按下后变为0,通过比较后说明有按键按入

MOV A,P1 //读入第四位，就是行线状态

XRL A, #0FH //和0FH异或，如果为按键按下A为0

LCALL Lightchange

RET

Light:

LCALL JCK //调用检测程序

JNZ Light //当未松开按键时循环

POP Acc //释放保存的值

MOV @R0,A //存储按下值到字节地址

INC R0

AJMP KEY

Lightchange:

PUSH PSW

PUSH Acc

MOV R2,#0 //初始化R2为0

MOV R1,#27H //R1初始化存储地址

MOV R3,#08H

MOV DPTR,#TABLE

START:

MOV A,R2

MOV P2,A

ADD A,#4

MOV R2,A

MOV A,@R1

DEC R1

MOVC A,@A+DPTR

MOV p0,A

LCALL DELAY

DJNZ R3,START

POP Acc

POP PSW

RET

D10ms: //延时10ms

PUSH PSW

PUSH Acc

MOV 30H,#5

D0: MOV 31H,#4

D2:

MOV 32H,#250 //250\*2\*4\*5=10000纳秒=10ms

D1:

DJNZ 32H,D1

DJNZ 31H,D2

LCALL Lightchange

DJNZ 30H,D0

POP Acc

POP PSW

RET

DELAY:

MOV R5,#20

DEL0: MOV R6,#1

DEL1: MOV R7,#10

DEL2: DJNZ R7,DEL2

DJNZ R6,DEL1

DJNZ R5,DEL0

RET

TABLE:

DB 3FH, 06H, 5BH, 4FH, 66H, 6DH //0 1 2 3 4 5

DB 7DH, 07H, 7FH, 6FH, 77H, 7CH //6 7 8 9 A b

DB 39H, 5EH, 79H, 71H //C d E F

END