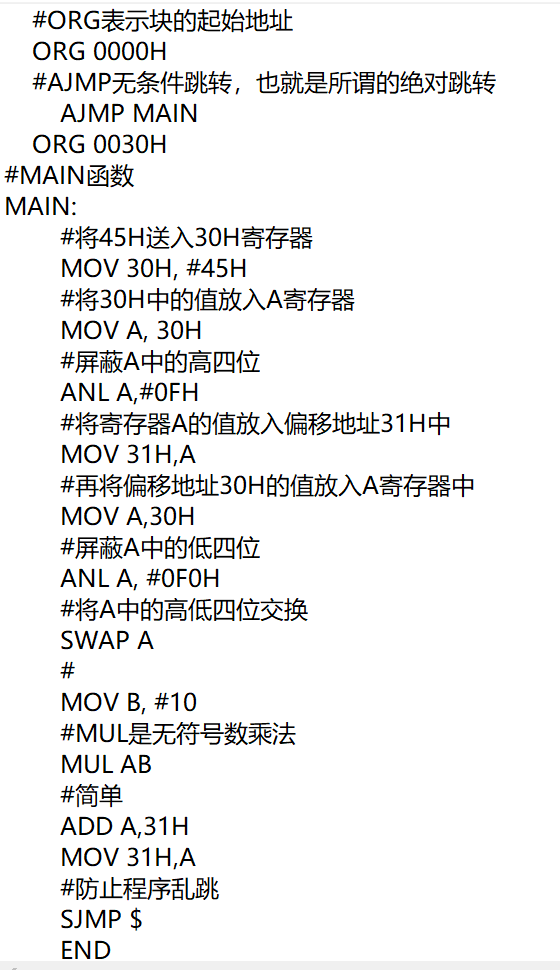
单片机实验报告

**一、实验2.2**

**（1）实验2.2.1：**

实验作业：

1.如下图所示：



2.实现功能：将16进制的45变成10进制的45

3.结果截图：

实验2.2.2：

1.代码（带注释如下）

ORG 0000H

AJMP MAIN

ORG 0100H

MAIN:

MOV SP,#62H

MOV A,#0H

MOV R1,#30H

;换成十进制的31

MOV R7,#20H

MOV DPTR,#TABLE

;疑惑：这R4干嘛的

MOV R4,#1FH

LOOP1:

;MOVC是查表指令

MOVC A,@A+DPTR

MOV @R1,A

INC R1

INC DPTR

MOV A,#0H

;输入32个数

DJNZ R7,LOOP1

LOOP2:

;清除借位标志位

CLR C

MOV A,#1FH

SUBB A,R4

ADD A,#30H

;令R1从该循环第一个数开始计算

MOV R1,A

MOV A,R4

;使R2为计算次数

MOV R2,A

MOV A,R1

;设置该循环初始值

MOV R0,A

LOOP3:

CLR C

MOV A,@R1

;R3中存入R1所指数据

MOV R3,A

MOV A,R0

;比较R0与R1的大小

SUBB A,R1

;执行流程图第一步

JNZ X

CLR C

INC R1

X:

MOV A,@R0

;比较R1与R0+1的大小

SUBB A,@R1

JC LOOP4

MOV A,R0

INC A

SUBB A,R1

JNZ Y

MOV A,R3

XCH A,@R1

MOV @R0,A

SJMP LOOP4

Y:

MOV A,R3

XCH A,@R0

MOV @R1,A

LOOP4:

INC R1

;持续循环直至结束

DJNZ R2,LOOP3

INC R0

;持续循环直至结束

DJNZ R4,LOOP2

SJMP $

ORG 0030H

TABLE: DB 11,3,9,2,17,4,11,6

DB 5,20,100,64,21,14,79,35

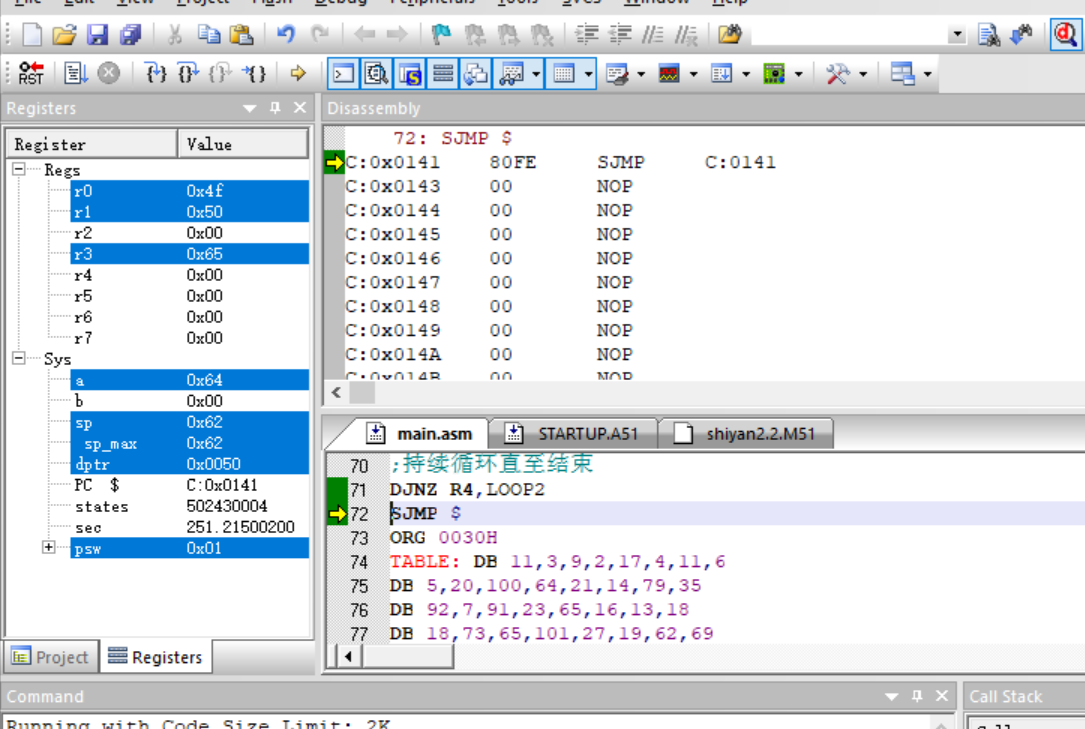
DB 92,7,91,23,65,16,13,18

DB 18,73,65,101,27,19,62,69

END

2.实验步骤：参照老师的流程图进行相应的编程

3.实验结果如下：

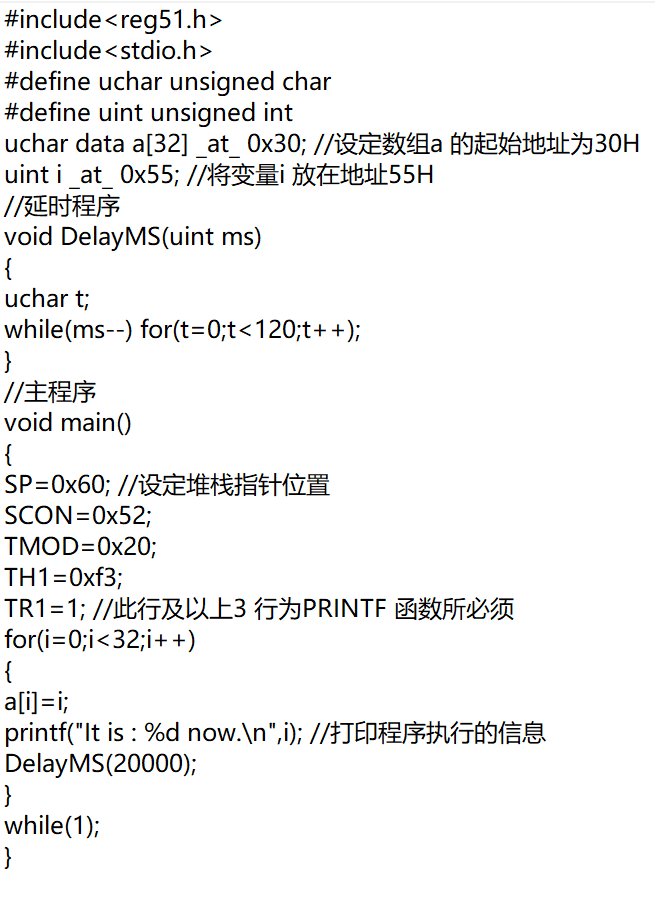


**二、实验2.3**

**（1）实验2.3.1**

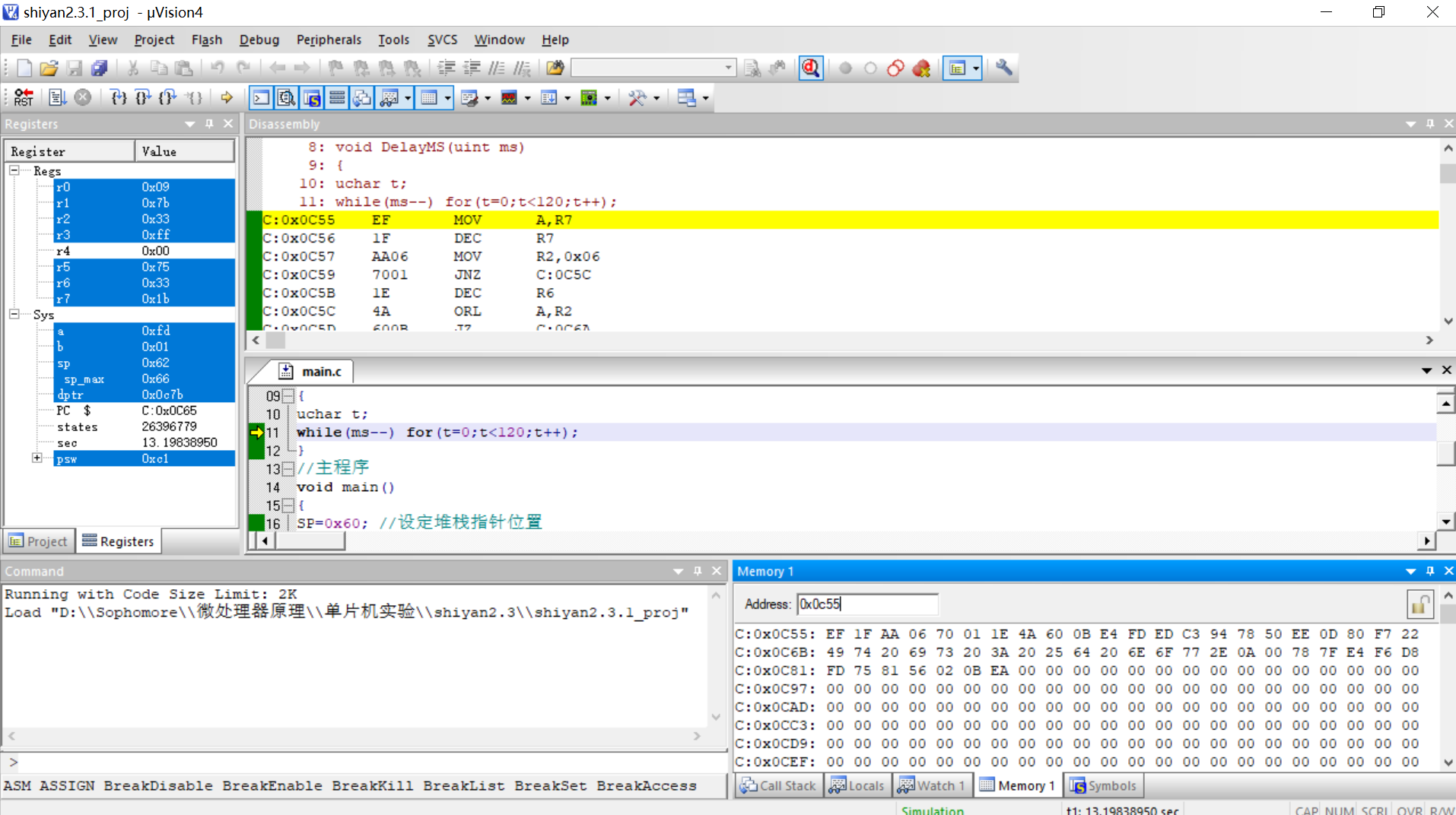
实验作业：

1.感觉没有什么可添加注释的，老师写的已经很详尽了



2.功能就是每20000ms打印一个数组a中的元素

3.实验结果

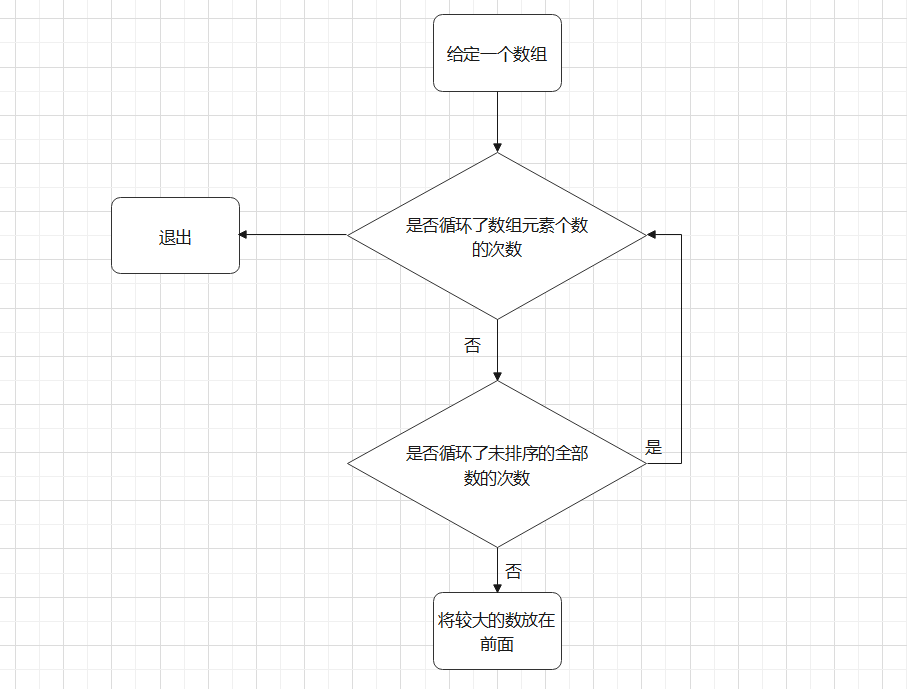


4.while（1）的作用

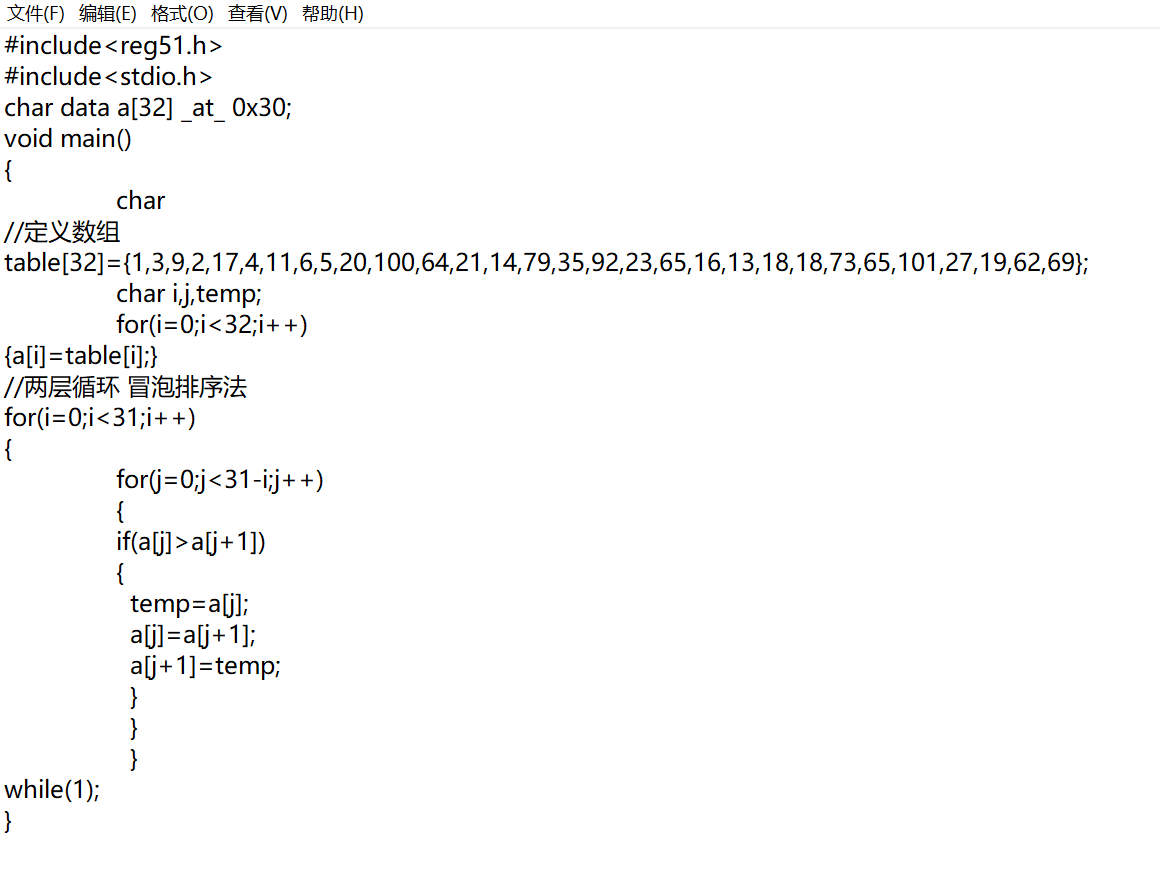
是一个死循环，防止程序自动结束

**（2）实验2.3.2**

1. 流程图如下：

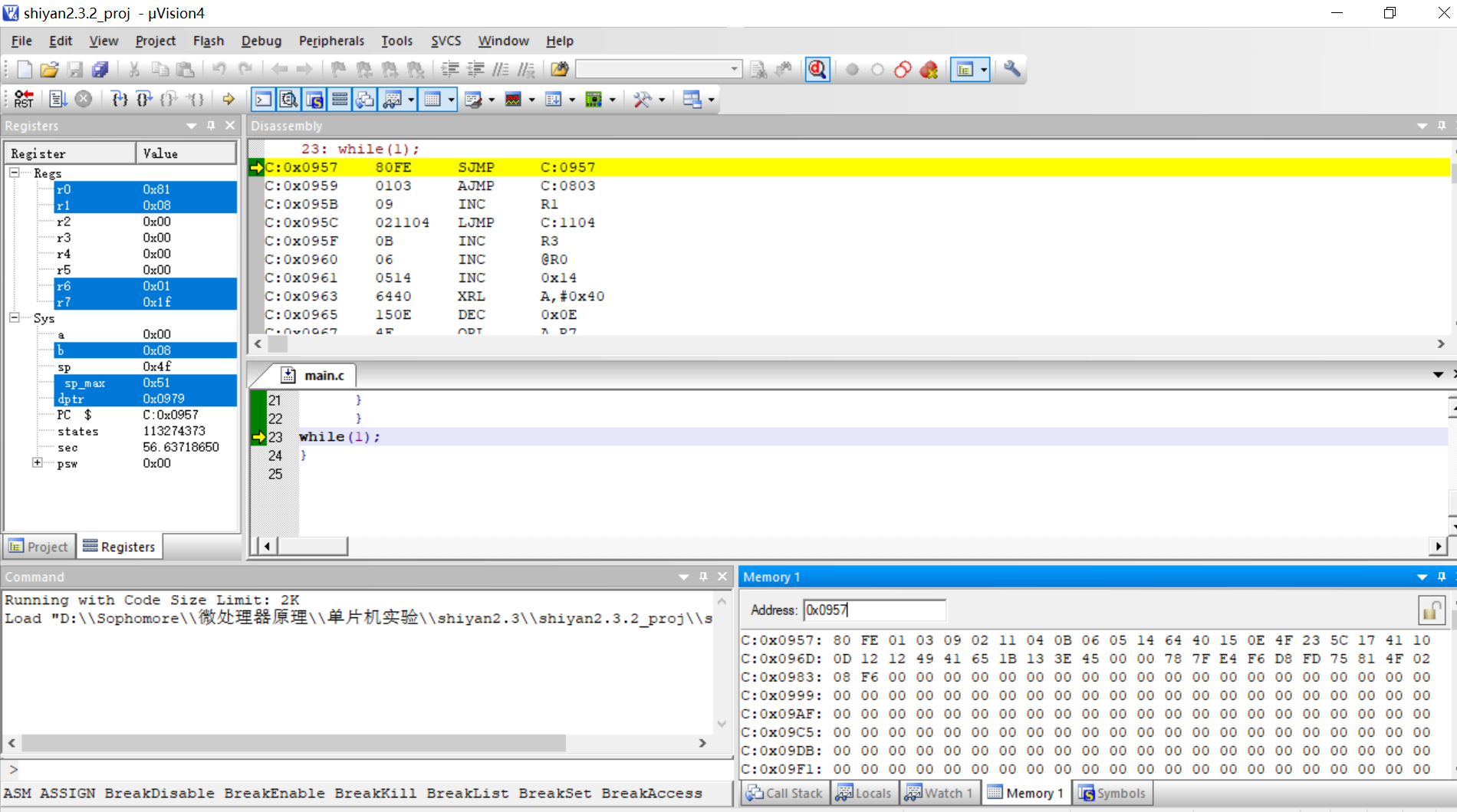


2.实验程序如下：



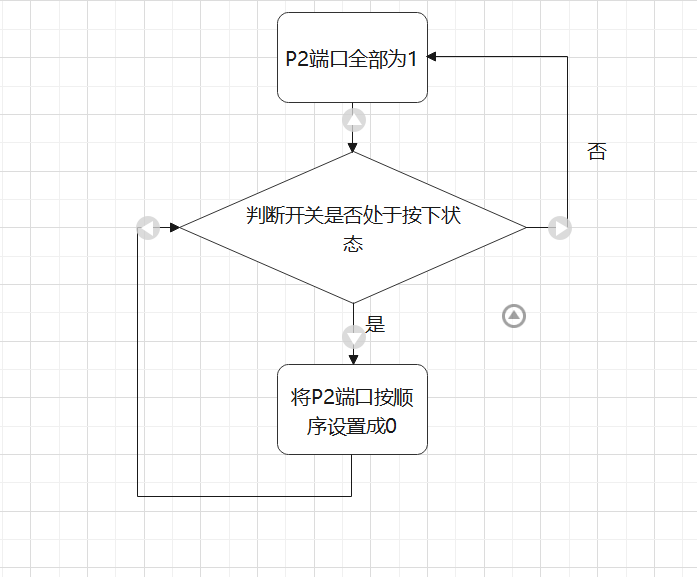
3.实验过程：参照流程图进行编写程序

4.实验结果如下



**三、实验3.1**

1.流程图：



2.程序以及注释：

ORG 0000H

AJMP MAIN

ORG 0030H

MAIN:

MOV SP,#60H

MOV P2,#0FEH

LOOP1:

JB P1.0, LOOP1 ；判断条件位P1.0是1的话跳转不是1的话就是低电平，表示开关按下了

；这个是换IO口进行点亮灯的情况

MOV A, P2

RL A ；左移一位

MOV P2, A

acall DL100MS

SJMP LOOP1

DL100MS: MOV R7,#200

DL: MOV R6,#250

DJNZ R6,$

DJNZ R7,DL

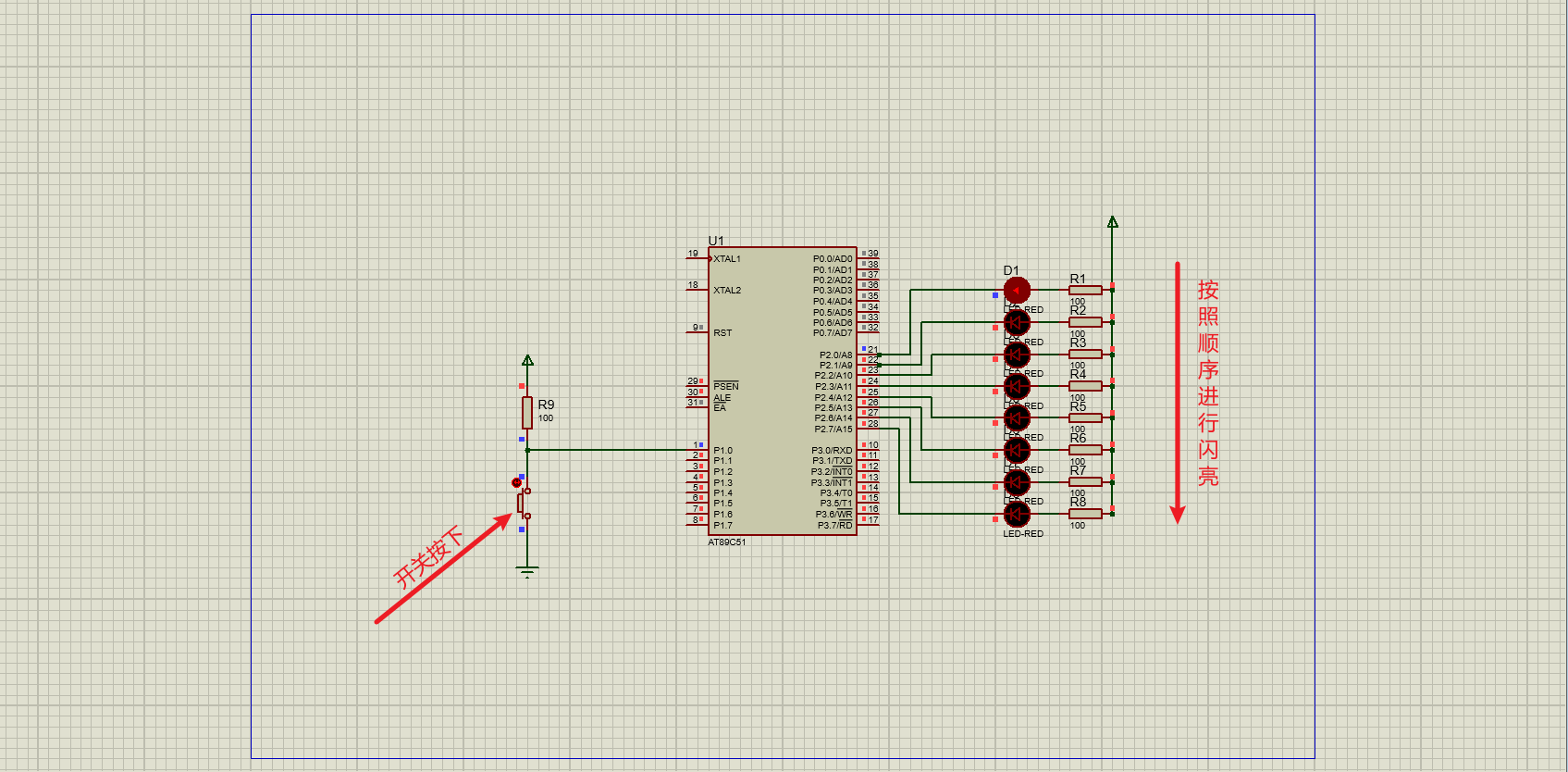
RET

END

3.实验过程：

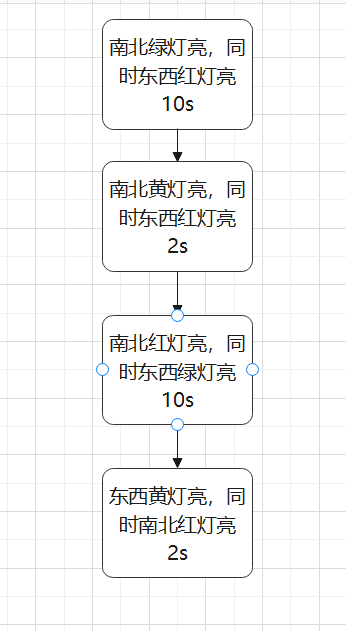
在Proteus中按照PDF连接相应的电路，根据需要实现的相应功能完成程序的编写，不断调试达到最优化的结果。

4.实验结果抓图：



**四、实验3.2**

1.流程图：通过引脚设计即可，并无较为复杂流程



2.实验程序：

ORG 0000H

AJMP START

ORG 0030H

；程序入口

START:

MOV SP,#60H

；调用一堆子程序

LOOP:

LCALL START1

LCALL DL10S

LCALL START2

LCALL DL2S

LCALL START3

LCALL DL10S

LCALL START4

LCALL DL2S

SJMP LOOP

;第一部分的亮的情况

START1:

MOV A,#0F3H

MOV DPTR,#U4

MOVX @DPTR,A

MOV A,#0CH

MOV DPTR,#U5

MOVX @DPTR,A

RET

；第二部分亮的情况

START2:

MOV A,#0C3H

MOV DPTR,#U4

MOVX @DPTR,A

MOV A,#0FH

MOV DPTR,#U5

MOVX @DPTR,A

RET

；第三部分亮的情况

START3:

MOV A,#0FCH

MOV DPTR,#U4

MOVX @DPTR,A

MOV A,#03H

MOV DPTR,#U5

MOVX @DPTR,A

RET

；第四部分亮的情况

START4:

MOV A,#3CH

MOV DPTR,#U4

MOVX @DPTR,A

MOV A,#0FH

MOV DPTR,#U5

MOVX @DPTR,A

RET

DL10S:

MOV R1,#5

AA1:LCALL DL2S

DJNZ R1,AA1

RET

DL2S: MOV R7,#20

DL2: MOV R6,#200

DL1: MOV R5,#250

DJNZ R5,$

DJNZ R6,DL1

DJNZ R7,DL2

RET

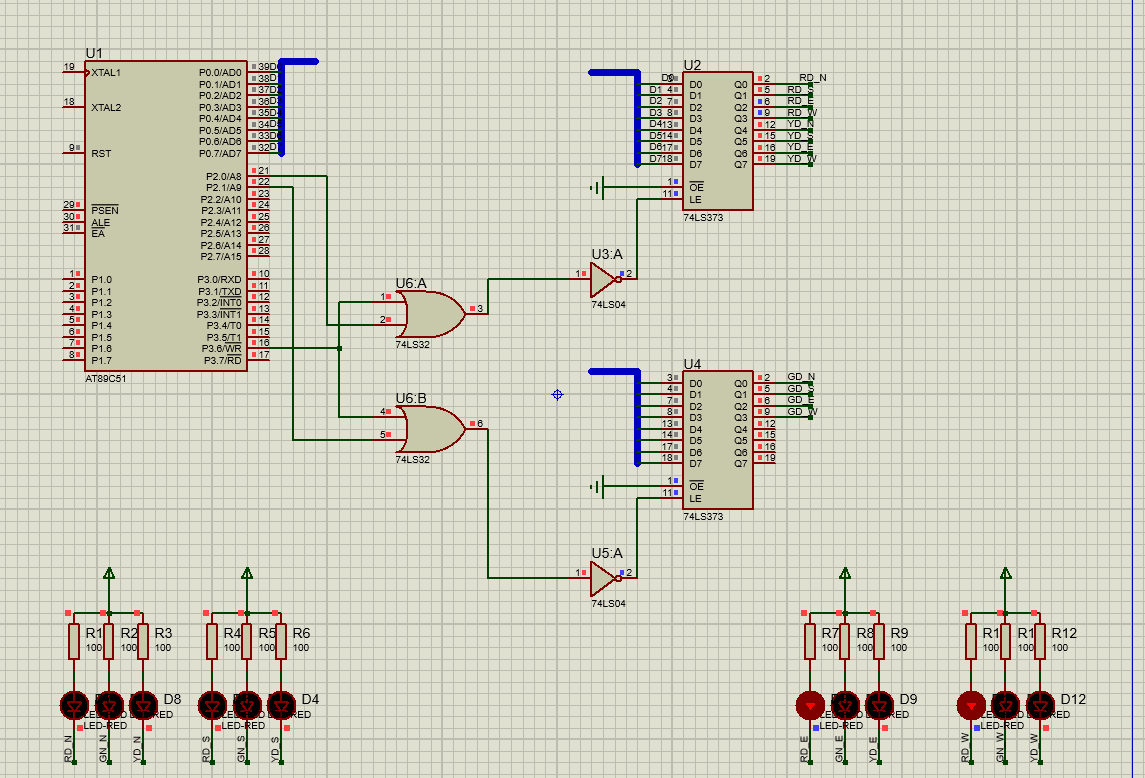
U4 EQU 0FE00H

U5 EQU 0FD00H

END

3.实验过程：根据PDF文档以及老师给定的要求编写程序，在Proteus中按照示例连接原理图

4.实验抓图：



**五、实验3.3**

1.实验程序：

ORG 0000H

AJMP MAIN

ORG 0030H

MAIN:

MOV SP,#60H

MOV R0,#0

MOV R1,#1

SETB P3.0

CLR F0

LOOP:

；判断P1.1是不是1，是的话跳转到LOOP2

JB P1.1,LOOP2

LOOP1:

CLR F0

MOV 30H,R0

MOV 31H,R1

SETB P3.0

LCALL DISPLAY

LOOP2:

JB P1.0,LOOP

LOOP3:

LCALL DISPLAY

LCALL DELAY1S

LCALL ADJUST2

JB F0,LOOP4

LJMP LOOP3

LOOP4:

CLR P3.0

LCALL DELAY100MS

SETB P3.0

LCALL DELAY100MS

JB P1.1,LOOP4

LJMP LOOP1

DISPLAY:

MOV A,30H

；通过TABLE表传进去的

MOV DPTR,#TABLE

；MOVC是查表指令

MOVC A,@A+DPTR

MOV DPTR,#D1ADD

；MOVX是访问外部内存的指令

MOVX @DPTR,A

MOV A,31H

MOV DPTR,#TABLE

MOVC A,@A+DPTR

MOV DPTR,#D10ADD

MOVX @DPTR,A

RET

ADJUST2:

DEC 30H

MOV A,30H

；比较两个操作数如果相等就跳转

CJNE A,#-1,GOTORET

MOV 30H,#9

DEC 31H

MOV A,31H

CJNE A,#-1,GOTORET

SETB F0

RET

GOTORET:

RET

DELAY1S:MOV R7,#10

DL2:MOV R6,#200

DL1:MOV R5,#250

DJNZ R5,$

DJNZ R6,DL1

DJNZ R7,DL2

RET

DELAY100MS:MOV R7,#200

DL:MOV R6,#250

；将寄存器的数减1看是否为0，为0就不转移

DJNZ R6,$

DJNZ R7,DL

RET

；7段LED共阳显示码

TABLE:DB 0C0H,0F9H,0A4H,0B0H,99H,0F8H,80H,90H

；这是给定的两个锁存器的地址

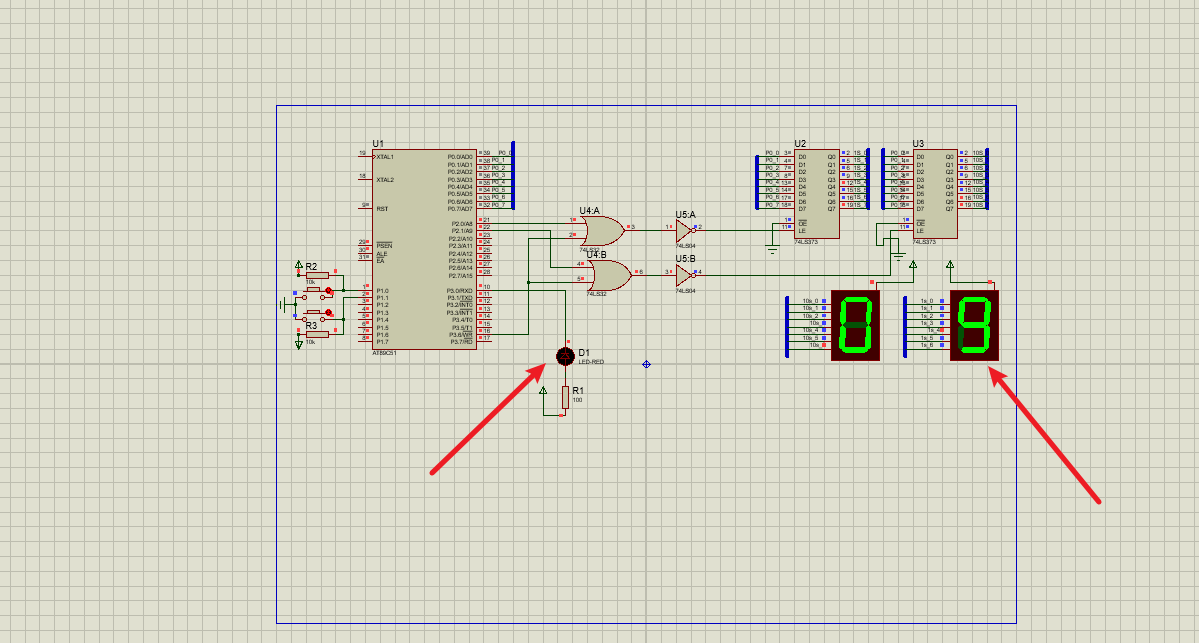
D1ADD EQU 0FE00H

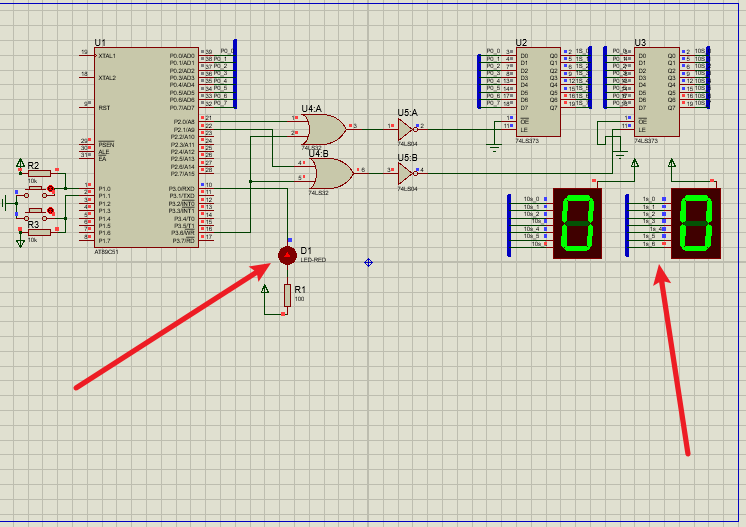
D10ADD EQU 0FD00H

END

2.实验过程：借鉴3.3.1中的数码管显示的实验示例，进行改编，根据流程图编写出正确的程序

3.实验结果：





**六、实验3.4**

1.实验代码加注释：

#include <reg51.h>

#include <stdio.h>

#include <intrins.h>

#define unit unsigned int

#define uchar unsigned char

//这个是C51的关键字，进行外部声明的

sbit D5=P0^0;

sbit D4=P0^1;

sbit D3=P0^2;

sbit D2=P0^3;

sbit D1=P0^4;

uchar data key \_at\_ 0x30;

uchar code KEY\_TABLE[]={0x22,0x12,0x21,0x11};

uchar code OUT\_TABLE[]={0x7f,0xbf,0xdf,0xef,0xf7,0xfb,0xfd,0xfe};

//这里是延迟函数的编写

void delay(uchar n)

    {

     uchar t,i;

     uchar m=200;

     while(m--)

     for(i=0;i<n;i++)

     {

     for(t=0;t<120;t++);

     }

    }

void delays()

    {

      uchar t,ms;

      ms = 10;

      while(ms--)for(t=0;t<120;t++);

    }

void main()

    {

      uchar DIR=0;

      uchar speed=3;

      char m,n;

      uchar i,num;

      uchar temp;

      P1=0x7f;

      while(1)

      {

          //这里0xf0就是1的意思，0x30就是0的意思

        P3=0xf0;

        temp=P3;

        temp=temp&0x30;

        if(temp!=0x30)

        {

            //这是按键扫描程序

          delays();

          P3=0xf0;

          temp=P3;

          temp=temp&0x30;

          if(temp!=0x30)

          {

            key=temp;

            temp=temp|0x03;

            P3=temp;

            temp=P3&0x03;

            key=temp|key;

            key=0x33-key;

            P3=0xf0;

            for(i=0;i<4;i++)

            {

                if(key==KEY\_TABLE[i])

            {

                    num=i;

                    break;

            }

            }

        switch(num)

        {

            case 0:

            {

            speed++;

            //对于溢出的处理

            if(speed==6){speed=1;}break;

            }

            case 1:

            {

            speed--;

            if(speed==0){speed=5;}break;

            }

            case 2:DIR;break;

            case 3:DIR;break;

        }

        switch(speed)

        {

            //将这数段按照数组的形式放入P0各个口中

            case 1:P0=0xFE;break;

            case 2:P0=0xFD;break;

            case 3:P0=0xFB;break;

            case 4:P0=0xF7;break;

            case 5:P0=0xEF;break;

        }

        if(DIR==1)

            {

               for(;m<8;m++)

               {P1=OUT\_TABLE[m];

               //用这个来控制速度

               delay(speed);

               if(m==7){m=-1;}

               temp=P3&0x30;

               if(temp!=0x30){break;}

               }

           }

       if(DIR==0)

        {

            for(;m<8;m--)

            {

                P1=OUT\_TABLE[m];

                delay(speed);

                if(m==0){m=8;}

                temp=P3&0x30;

                if(temp!=0x30){break;}

            }

        }

    }

}

}

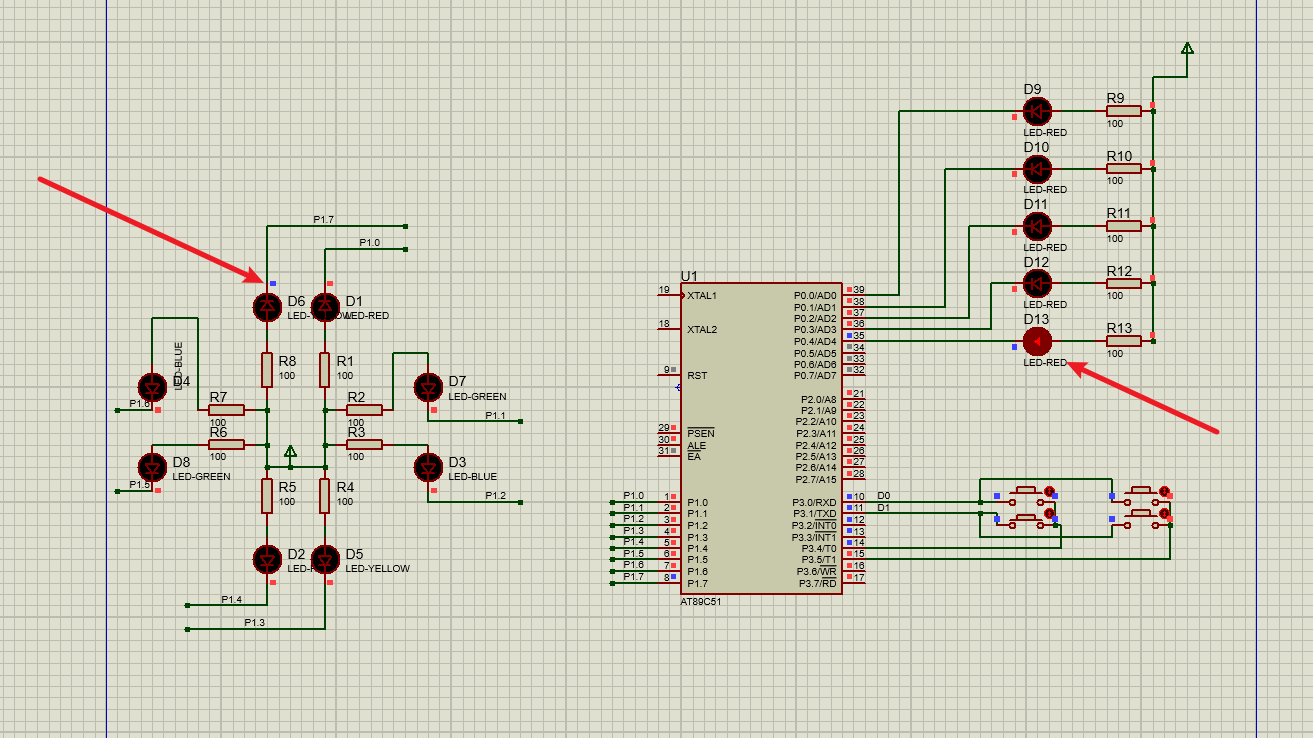
}

2. 实验过程：

在Proteus中按照PDF连接相应的电路，根据需要实现的相应功能完成程序的编写，不断调试达到最优化的结果。

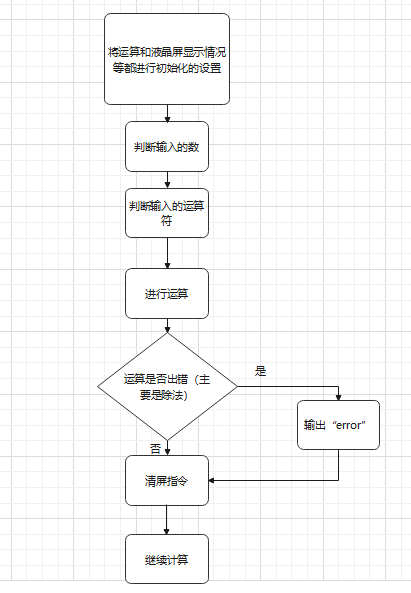
3.

实验结果：



**七、实验3.7**

1.流程图：



2.代码带注释：

#include <reg51.h>        //头文件

#include <intrins.h>

#define uint unsigned int

#define uchar unsigned char

sbit lcden = P1 ^ 5; //定义引脚

sbit rs = P1 ^ 7;

sbit rw = P1 ^ 6;

sbit busy = P0 ^ 7;

char i, j, temp, num;

long a, b, c, d;

float a\_c, b\_c;

uchar flag, op, p2\_tmp; //flag表示是否有符号键按下，op表征按下的是哪个符号

uchar code table[] = {

    0, 1, 2, 3,

    4, 5, 6, 7,

    8, 9, 0, 0,

    0, 0, 0, 0};

uchar code table1[] = {

    0, 1, 2, 3,

    4, 5, 6, 7,

    8, 9, 0x2b - 0x30, 0x2d - 0x30,

    0x2a - 0x30, 0x2f - 0x30, 0x01 - 0x30, 0x3d - 0x30};

uchar code table2[] = "!rorrE"; //第二行从右向左输出，因此用倒序

void delay(uchar z) // 延迟函数

{

    uchar y;

    for (z; z > 0; z--)

        for (y = 0; y < 110; y++)

            ;

}

void check() // 判断忙或空闲

{

    do

    {

        P2 = 0xFF;

        rs = 0;          //指令

        rw = 1;          //读

        lcden = 0;       //禁止读写

        delay(1);        //等待，液晶显示器处理数据

        lcden = 1;       //允许读写

    } while (busy); //判断是否为空闲，1为忙，0为空闲

}

void write\_com(uchar com) // 写指令函数

{

    P2 = com; //com指令付给P0口

    rs = 0;

    rw = 0;

    lcden = 0;

    check();

    lcden = 1;

}

void write\_data(uchar data\_) // 写数据函数

{

    P2 = data\_;

    rs = 1;

    rw = 0;

    lcden = 0;

    check();

    lcden = 1;

}

void init() //初始化

{

    num = -1;

    lcden = 1;       //使能信号为高电平

    write\_com(0x38); //8位，2行

    write\_com(0x0c); //显示开，光标关，不闪烁

    write\_com(0x06); //增量方式不移位

    write\_com(0x80); //检测忙信号

    write\_com(0x01); //显示开，光标关，不闪烁

    i = 0;

    j = 0;

    a = 0; //第一个参与运算的数

    b = 0; //第二个参与运算的数

    c = 0; //运算结果

    d = 0; //整除的余数

    flag = 0;  //flag表示是否有符号键按下，

    op = 0; // op表征按下的是哪个符号

}

void keyscan() // 键盘扫描程序

{

    P3 = 0xfe;

    if (P3 != 0xfe)

    {

        delay(20);

        if (P3 != 0xfe)

        {

            temp = P3 & 0xf0;

            switch (temp)

            {

            case 0xe0:

                num = 0;

                break;

            case 0xd0:

                num = 1;

                break;

            case 0xb0:

                num = 2;

                break;

            case 0x70:

                num = 3;

                break;

            }

        }

        while (P3 != 0xfe)

            ;

        if (flag == 0) //没有按过符号键

        {

            a = a \* 10 + table[num];

        }

        else //如果按过符号键

        {

            b = b \* 10 + table[num];

        }

        i = table1[num];

        write\_data(0x30 + i);

    }

    P3 = 0xfd;

    if (P3 != 0xfd)

    {

        delay(5);

        if (P3 != 0xfd)

        {

            temp = P3 & 0xf0;

            switch (temp)

            {

            case 0xe0:

                num = 4;

                break;

            case 0xd0:

                num = 5;

                break;

            case 0xb0:

                num = 6;

                break;

            case 0x70:

                num = 7;

                break;

            }

        }

        while (P3 != 0xfd)

            ;

        if (flag == 0) //没有按过符号键

        {

            a = a \* 10 + table[num];

        }

        else //如果按过符号键

        {

            b = b \* 10 + table[num];

        }

        i = table1[num];

        write\_data(0x30 + i);

    }

    P3 = 0xfb;

    if (P3 != 0xfb)

    {

        delay(5);

        if (P3 != 0xfb)

        {

            temp = P3 & 0xf0;

            switch (temp)

            {

            case 0xe0:

                num = 8;

                break;

            case 0xd0:

                num = 9;

                break;

            case 0xb0:

                num = 10;

                break;

            case 0x70:

                num = 11;

                break;

            }

        }

        while (P3 != 0xfb)

            ;

        if (num == 8 || num == 9) //如果按下的是'8','9'

        {

            if (flag == 0) //没有按过符号键

            {

                a = a \* 10 + table[num];

            }

            else //如果按过符号键

            {

                b = b \* 10 + table[num];

            }

        }

        else if (num == 10) //如果按下的是'+'

        {

            flag = 1;

            op = 1; //2表示加号已按

        }

        else if (num == 11) //如果按下的是'-'

        {

            flag = 1;

            op = 2; //2表示减号已按

        }

        i = table1[num];

        write\_data(0x30 + i);

    }

    P3 = 0xf7;

    if (P3 != 0xf7)

    {

        delay(5);

        if (P3 != 0xf7)

        {

            temp = P3 & 0xf0;

            switch (temp)

            {

            case 0xe0:

                num = 12;

                break;

            case 0xd0:

                num = 13;

                break;

            case 0xb0:

                num = 14;

                break;

            case 0x70:

                num = 15;

                break;

            }

        }

        while (P3 != 0xf7)

            ;

        switch (num)

        {

            case 12:

            {

                write\_data(0x30 + table1[num]);

                flag = 1;

                op = 3;

            }

            break;

            case 13:

            {

                write\_data(0x30 + table1[num]);

                flag = 1;

                op = 4;

            }

            break;

            case 14:

            {

                write\_com(0x01);

                i = 0;

                j = 0;

                a = 0;

                b = 0;

                c = 0;

                d = 0;

                flag = 0;

                op = 0;

            } //按下的是"清零"

            break;

            case 15:

            {

                j = 1;

                if (op == 1)

                {

                    write\_com(0x80 + 0x4f); //按下等于键，光标前进至第二行最后一个显示处

                    write\_com(0x04);        //设置从后往前写数据，每写完一个数据，光标后退一格

                    c = a + b;

                    while (c != 0)

                    {

                        write\_data(0x30 + c % 10);

                        c = c / 10;

                    }

                    write\_data(0x3d); //再写"="

                    a = 0;

                    b = 0;

                    flag = 0;

                    op = 0;

                }

                else if (op == 2)

                {

                    write\_com(0x80 + 0x4f); //光标前进至第二行最后一个显示处

                    write\_com(0x04);        //设置从后往前写数据，每写完一个数据，光标后退一格)

                    if (a - b > 0)

                        c = a - b;

                    else

                        c = b - a;

                    while (c != 0)

                    {

                        write\_data(0x30 + c % 10);

                        c = c / 10;

                    }

                    if (a - b < 0)

                        write\_data(0x2d);

                    write\_data(0x3d); //再写"="

                    a = 0;

                    b = 0;

                    flag = 0;

                    op = 0;

                }

                else if (op == 3)

                {

                    write\_com(0x80 + 0x4f);

                    write\_com(0x04);

                    c = a \* b;

                    while (c != 0)

                    {

                        write\_data(0x30 + c % 10);

                        c = c / 10;

                    }

                    write\_data(0x3d);

                    a = 0;

                    b = 0;

                    flag = 0;

                    op = 0;

                }

                else if (op == 4)

                {

                    write\_com(0x80 + 0x4f);

                    write\_com(0x04);

                    if (b == 0) //除数为零

                    {

                        i = 0;

                        while (table2[i] != '\0')

                        {

                            write\_data(table2[i]);

                            i++;

                        } //输出错误信息

                        a = 0;

                        b = 0;

                        flag = 0;

                        op = 0;

                    }

                    else if ((a % b == 0) && (b != 0)) //可以整除

                    {

                        c = a / b;

                        while (c != 0)

                        {

                            write\_data(0x30 + c % 10);

                            c = c / 10;

                        } //输出商

                        if (a / b <= 0)

                            write\_data(0x30);

                        write\_data(0x3d);

                        a = 0;

                        b = 0;

                        flag = 0;

                        op = 0;

                    }

                    else if ((a % b != 0) && (b != 0)) //不能整除

                    {

                        d = a % b;

                        while (d != 0)

                        {

                            write\_data(0x30 + d % 10);

                            d = d / 10;

                        } //输出余数

                        write\_data(0x2e);

                        write\_data(0x2e);

                        write\_data(0x2e); //三个点分隔

                        c = a / b;

                        while (c != 0)

                        {

                            write\_data(0x30 + c % 10);

                            c = c / 10;

                        }

                        if (a / b <= 0)

                            write\_data(0x30);

                        write\_data(0x3d);

                        a = 0;

                        b = 0;

                        flag = 0;

                        op = 0;

                    }

                }

            }

            break;

        }

    }

}

int main(void)

{

    init();

    while (1)

    {

        keyscan();

    }

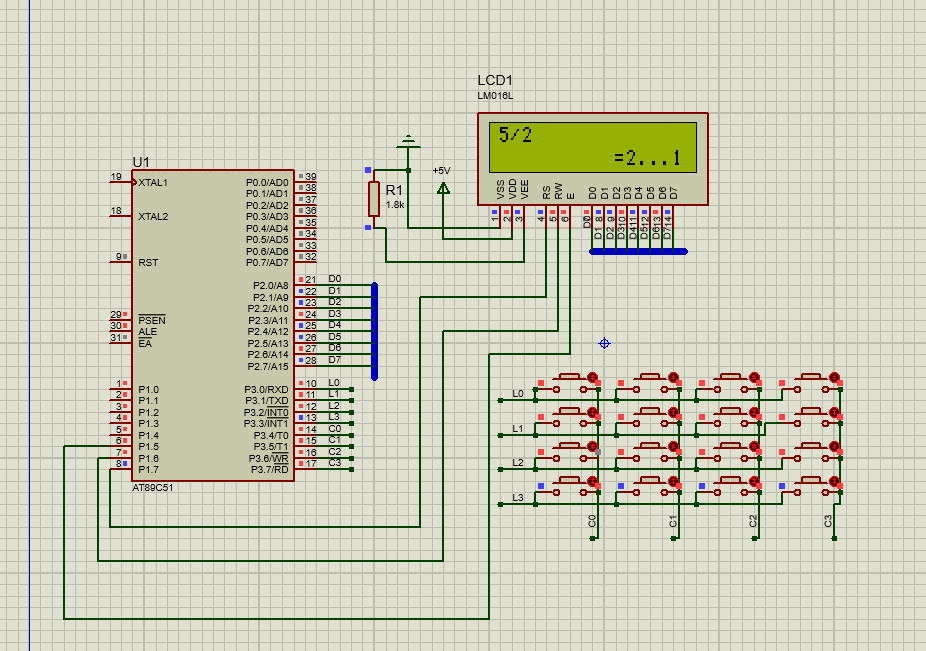
    return 0;

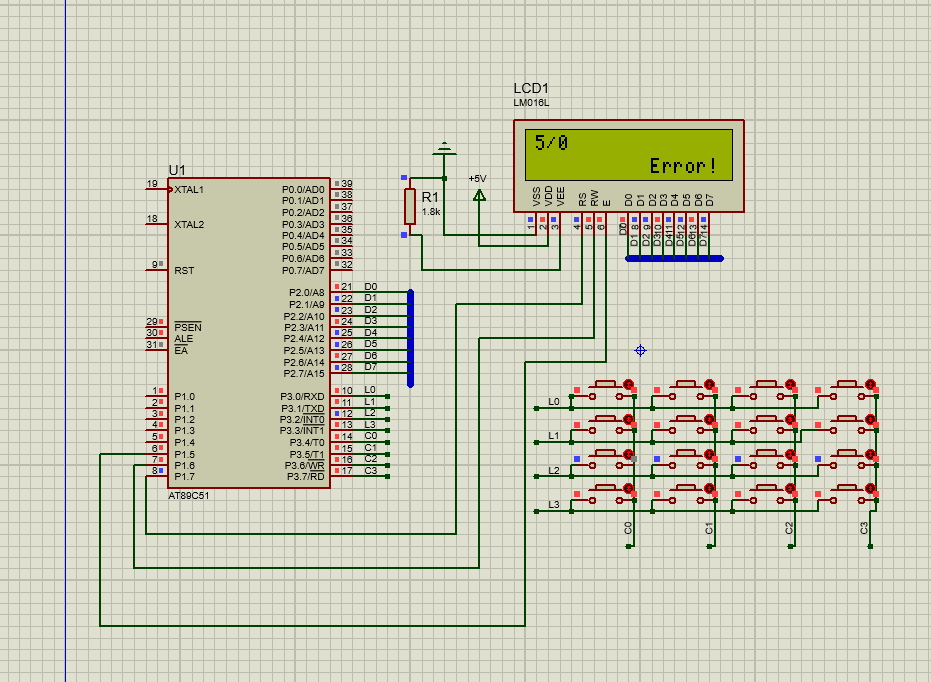
}

3.实验过程：

首先按照老师所给的要求画出相应的流程图，然后根据流程图将程序编写出来，通过proteus编写程序得到PDF中给出的电路图示例，通过不断的调试最终得到一个较为成功的展示结果。

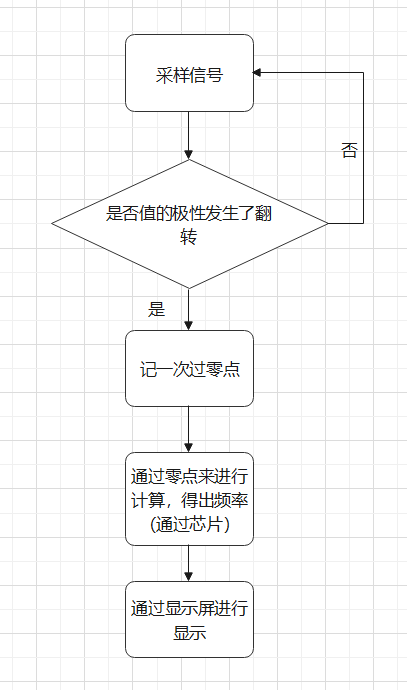
4.实验结果抓图：





**八、实验3.8**

1.流程图



2.编程代码：#include <reg51.h>

#include <intrins.h>

#define uint unsigned int

#define uchar unsigned char

uchar data line\_data[16];

uchar code numchar[] = "0123456789";

sbit LCD\_RS = P2 ^ 0;

sbit LCD\_RW = P2 ^ 1;

sbit LCD\_EN = P2 ^ 2;

sbit AD\_CLK = P2 ^ 3;

sbit Start = P2 ^ 4;

sbit OE = P2 ^ 5;

sbit Out\_pulse = P2 ^ 6;

sbit EOC = P2 ^ 7;

uint tmpint, number, c;

uchar samp\_data, tmpchar, n, t, a, b, d, p1\_tmp;

void delay\_ms(uint xms)

{ //延时子程序

    uint j;

    while (xms--)

        for (j = 110; j > 0; j--)

            ;

}

bit lcd\_busy()

{ //LCD忙检查子程序，最高位BF=1表示液晶显示器忙，暂时无法接收单片机送来的数据或指令

    bit flag;

    p1\_tmp = P1;//保护P1口的值，防止其被修改（推测是Proteus8的问题）

    LCD\_RS = 0;

    LCD\_RW = 1;

    LCD\_EN = 1;

    \_nop\_(); //如果单片机的晶振是12M，则调用\_nop\_()用于延时1us

    \_nop\_();

    \_nop\_();

    \_nop\_();

    flag = (bit)(P1 & 0x80); //返回最高位BF代表是否忙，忙flag=1，不忙flag=0

    LCD\_EN = 0;

    P1 = p1\_tmp;

    return flag;

}

//写命令子程序

void lcd\_wcmd(uchar cmd)

{

    while (lcd\_busy());

    LCD\_RS = 0;

    LCD\_RW = 0;

    LCD\_EN = 0;

    \_nop\_();

    \_nop\_();

    P1 = cmd; //P1口传输命令到LCD1602

    \_nop\_();

    \_nop\_();

    \_nop\_();

    \_nop\_();

    LCD\_EN = 1;

    \_nop\_();

    \_nop\_();

    \_nop\_();

    \_nop\_();

    LCD\_EN = 0; //下降沿脉冲

}

void lcd\_clr()

{

    lcd\_wcmd(0x01); //清屏命令

    delay\_ms(2);    //清屏命令需要1.64ms以上

}

//写数据子程序

void lcd\_wdat(uchar dat)

{

    while (lcd\_busy());

    LCD\_RS = 1;

    LCD\_RW = 0;

    LCD\_EN = 0;

    \_nop\_();

    \_nop\_();

    P1 = dat; //P1口传输数据到LCD1602

    \_nop\_();

    \_nop\_();

    \_nop\_();

    \_nop\_();

    LCD\_EN = 1;

    \_nop\_();

    \_nop\_();

    \_nop\_();

    \_nop\_();

    LCD\_EN = 0; //下降沿脉冲

}

void lcd\_init()

{

    delay\_ms(5);    //等待LCD电源稳定

    lcd\_wcmd(0x01); //清除LCD显示内容

    delay\_ms(5);

    lcd\_wcmd(0x06); //移动光标

    delay\_ms(5);

    lcd\_wcmd(0x0c); //显示开，关光标，不闪烁

    delay\_ms(5);

    lcd\_wcmd(0x38); //8位数据总线，2行，5\*7点阵每字符

    delay\_ms(15);

}

void showstring()

{

    uchar i = 0;

    lcd\_wcmd(0x80); //设置显示位置为第1行第0列

    while (line\_data[i] != '\0')

    {

        lcd\_wdat(line\_data[i]);

        delay\_ms(5);

        i++;

    }

}

void s\_timer0() interrupt 1

{ //定时器T0实现中断，完成50ms定时

    t++;

    TH0 = 0x3c; //T0采用方式1，需要重装初值

    TL0 = 0xb0;

}

void s\_timer1() interrupt 3

{ //定时器T1实现中断，为50kHz

    AD\_CLK = ~AD\_CLK; //取反

}

void sample()

{

    TR0 = 0; //关闭定时器T0中断，避免该子程序对采集精度的影响，因为此子程序是用用于处理采集结果的

    if (samp\_data > 128 && b == 0)

    {   //此处引入变量a,b用来找出零点，假如第一次采集结果samp\_data大于128，则采集结果为正数，则b赋1，当下一次采集结果用于此处判断时一定不会执行判断后

        //的指令，因为只有当b = 0时才会执行这个判断；而下一个判断却可以将b赋为0，执行下一个判断，采集结果必须为负，由此实现对零点的计数

        number++;

        a = 0;

        b = 1;

        Out\_pulse = 1; //信号过零点，产生5us脉冲信号

        \_nop\_();

        \_nop\_();

        \_nop\_();

        \_nop\_();

        \_nop\_();

        Out\_pulse = 0;

    }

    else if (samp\_data <= 128 && a == 0)

    {

        number++;

        a = 1;

        b = 0;

        Out\_pulse = 1; //信号过零点，产生5us脉冲信号

        \_nop\_();

        \_nop\_();

        \_nop\_();

        \_nop\_();

        \_nop\_();

        Out\_pulse = 0;

    }

    TR0 = 1; //打开定时器T0中断

}

int main(void)

{

    delay\_ms(10);

    lcd\_init(); //初始化

    lcd\_clr();  //清屏

    delay\_ms(2);

    t = 0;

    a = 0;

    b = 0;

    n = 0;

    line\_data[n++] = 'f'; //先将要在LCD1602上输出的字符串的前三个值赋为'f：'

    line\_data[n++] = ':';

    line\_data[n++] = ' ';

    Out\_pulse = 0;

    number = 0;

    AD\_CLK = 0;

    OE = 0;      //关闭ADC0808的输出使能

    TMOD = 0x21; //T0工作在方式1，T1工作在方式2,T1不需要重装初值

    TH1 = 0xF6;  //T1 50kHz

    TL1 = 0xF6;

    TH0 = 0x3C; //T050ms定时

    TL0 = 0xB0;

    EA = 1; //打开中断总开关，下面就是打开定时器T0中断，定时器T1中断

    ET0 = 1;

    TR0 = 1;

    ET1 = 1;

    TR1 = 1;

    while (1)

    {

        Start = 1; //启动AD

        Start = 0; //启动信号结束

        while (EOC != 1)

            ;   //EOC转换结束为1

        OE = 1; //打开ADC0808的输出使能

        P0 = 0xff;

        samp\_data = P0; //采集

        OE = 0;         //关闭ADC0808的输出使能

        sample();       //处理采集结果

        if (t >= 9)

        {            //频率等于1s内的零点数除2，但是考虑到除2可能会产生余数，所以此处记录的0.5s的零点数，频率就直接等于0.5s的零点数

            TR0 = 0; //关闭定时器T0中断，避免该步骤对采集精度的影响，因为此步骤是用于计算频率

            t = 0;

            tmpint = number;

            number = 0;

            n = 3; //前面输出字符串中已经先赋值了前三个字符，此处要从第四个字符开始赋值

            d = 0;

            a = 0;

            b = 0;

            c = 100;

            while (c != 0)

            {

                tmpchar = (uchar)(tmpint / c);

                tmpint = tmpint - (uint)tmpchar \* c;

                c /= 10;

                if (tmpchar != 0) //输出计算得到的频率，此处tmpchar为显示的数据，为了避免输出多余的零在最终输出数据前面，所以引入了d用于标记有效数字

                    d = 1;

                if (d != 0)

                    line\_data[n++] = numchar[tmpchar];

            }

            line\_data[n++] = 'H';

            line\_data[n++] = 'z';

            line\_data[n] = '\0'; //将字符串补充完整

            showstring();

            TR0 = 1; //打开定时器T0中断

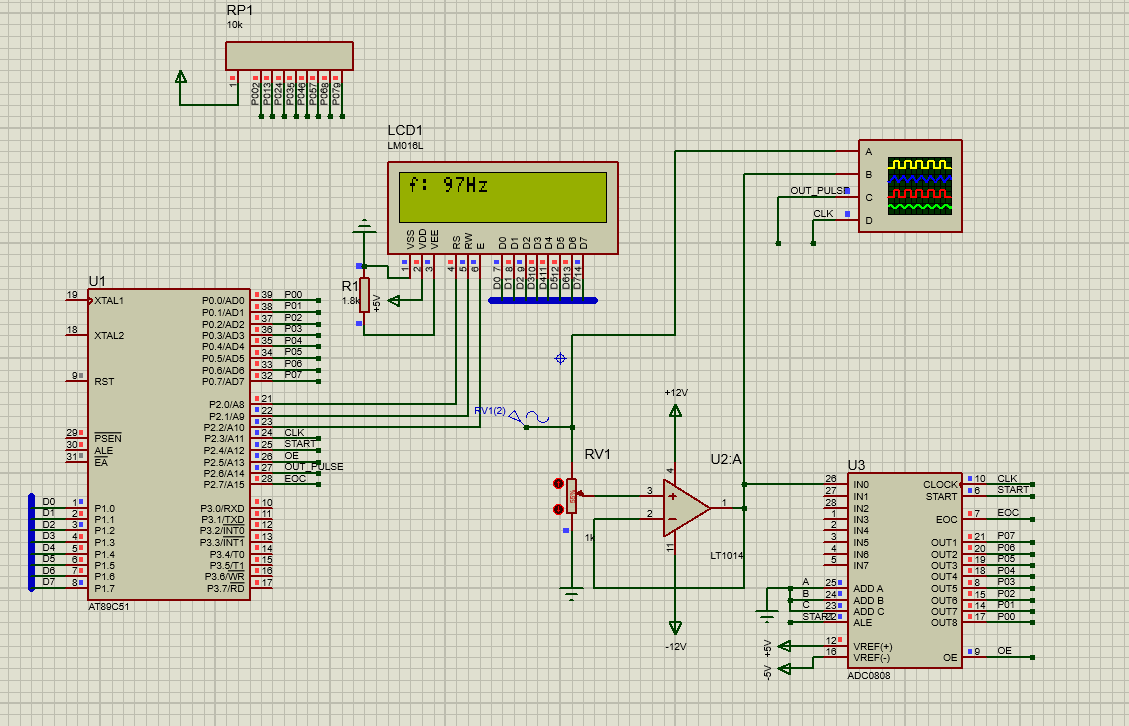
        }

    }

}

3.实验过程：根据老师的提示编写数采集点的方法，然后通过Proteus仿真按照PDF画出相应的图，将程序调整，提高精度

4.实验结果：100Hz的时候



**九、实验4.1**

1.实验程序：

#include<reg52.h>

#define uint unsigned int

#define uchar unsigned char

sbit bell=P3^1;    //用p3.1口控制bell

sbit led=P3^0;    //用P3^0来控制led

sbit key0=P3^2;   //控制报警的键

uchar count;

uchar i;

/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

                     1ms延时程序

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

void delay(uchar count)

  {uint x,y;

   for(x=count;x>0;x--)

     for(y=110;y>0;y--);

  }

/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

                 铃响，灯亮子程序

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

void show(uchar count)

  {i=count;

    for(;i!=0;i--)       //函数循环i次

      {led=0;bell=0;          //灯亮，铃响

        delay(500);        //延时0.5s

        led=1;bell=1;          //灯灭，铃停

        delay(500);             //延时0.5s

      }

  }

/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

                    0中断子程序

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

void s\_timer0() interrupt 0 using 0    //设置外部中断0

  {EA=0;

    if(count!=10)

    count=count+1;         //小于10，每次触发计数加一

    show(count);

    delay(50);

  if(count==10)                //若count为十则结束

    count=0;

    EA=1;

  }

/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

                         主程序

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

void main()

  {EA=1;       //打开总中断允许开关

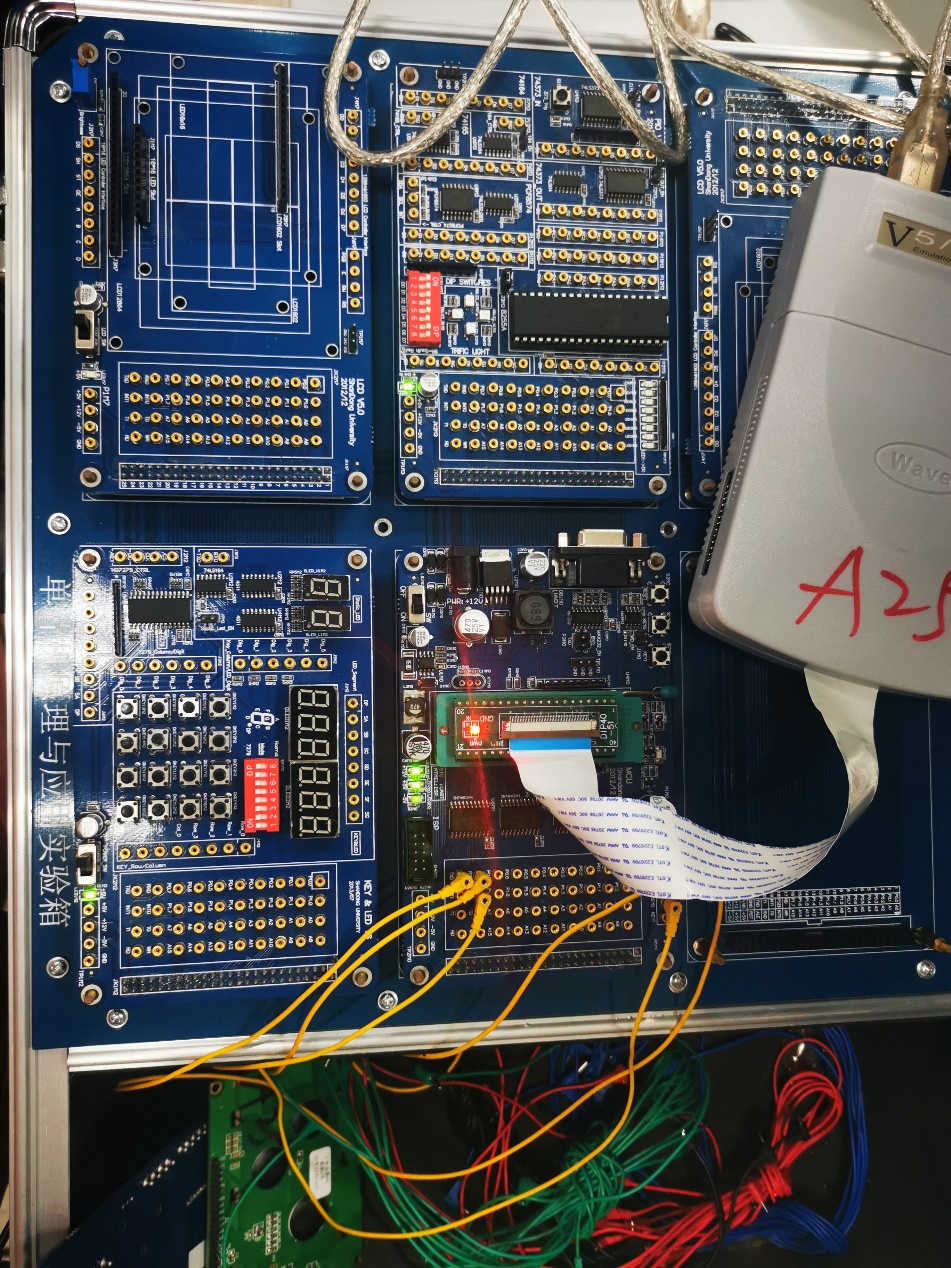
  EX0=1;       //开外中断0允许开关

  IT0=0;       //设置外中断0为电平触发方式

  while(1);

  }            //循环执行，等待

2.实现结果：



C语言编写C51单片机中断的时候，直接使用interrupt加上向量号进行编写就可以了

**十、实验4.5**

1.实验程序：

#include<reg51.h>

#include<stdio.h>

#include<absacc.h>  //使用其中定义的宏访问绝对地址

#define uchar unsigned char

#define uint unsigned int

#define PA XBYTE[0x7FFC]  //指代外部数据存储器7FFCh的地址

#define COM XBYTE[0x7FFF] //由P0 P2口联合输出地址

sbit P30=P3^0;

sbit P31=P3^1;

uint temp,shi,ge,t;

uchar a;

void show(uint temp);

void delay();

void delays();

void init\_8255();

void init\_8255()

{

    COM=0x80;  //c=out,a=out,b=out;

}

void delay()     //延时500ms子程序

{

    uint i,j;

    for(i=500;i>0;i--)

    for(j=120;j>0;j--);

}

void delays()  //延时1S子程序

{

    uint i,j;

    for(i=1000;i>0;i--)

    for(j=120;j>0;j--);

}

void main()

{

    init\_8255();

    TMOD=0x01;   //设定工作模式为1

    EA=1;        //开放中断

    ET0=1;       //允许T0中断

    TF0=0;       //定时器0溢出标志位

    while(1)

    {

        a=0x26;  //变量a初值

        for(temp=25;temp>19;temp--)   //绿灯倒计时25秒

        {

            a=a-1;

            PA=0X69;           //东西绿灯亮

            show(a);   //调用数码管显示子程序

            delays();  //延时1s

        }

        a=0x1a;

        for(temp=19;temp>9;temp--)

        {

            a=a-1;

            PA=0x69;  //东西绿灯亮

            show(a);

            delays();

        }

        a=0x0a;

        for(temp=9;temp>0;temp--)

        {

            a=a-1;

            PA=0x69;

            show(a);

            delays();

        }

        for(t=5;t>0;t--)

        {

            a=0;

            show(a);

            PA=0xeb;  //东西绿灯灭

            delay();

            PA=0x69;  //东西绿灯亮

            delay();

        }

        a=0x26;

        for(temp=25;temp>19;temp--) //绿灯倒计时25s

        {

            a=a-1;

            PA = 0x96;

            show(a);

            delays();

        }

        a=0x1a;

        for(temp=19;temp>9;temp--)

        {

            a=a-1;

            PA=0X96;

            show(a);

            delays();

        }

        a = 0x0a;

        for(temp=9;temp>0;temp--)

        {

            a=a-1;

            PA=0X96;

            show(a);

            delays();

        }

        for(t=5;t>0;t--);

        {

            a=  0;

            show(a);

            PA=0xbe;

            delay();

            PA=0x96;

            delay();

        }

    }

}

void show(uint a)//显示子程序

{

    SCON = 0X00;     //工作模式为0，13位定时器模式

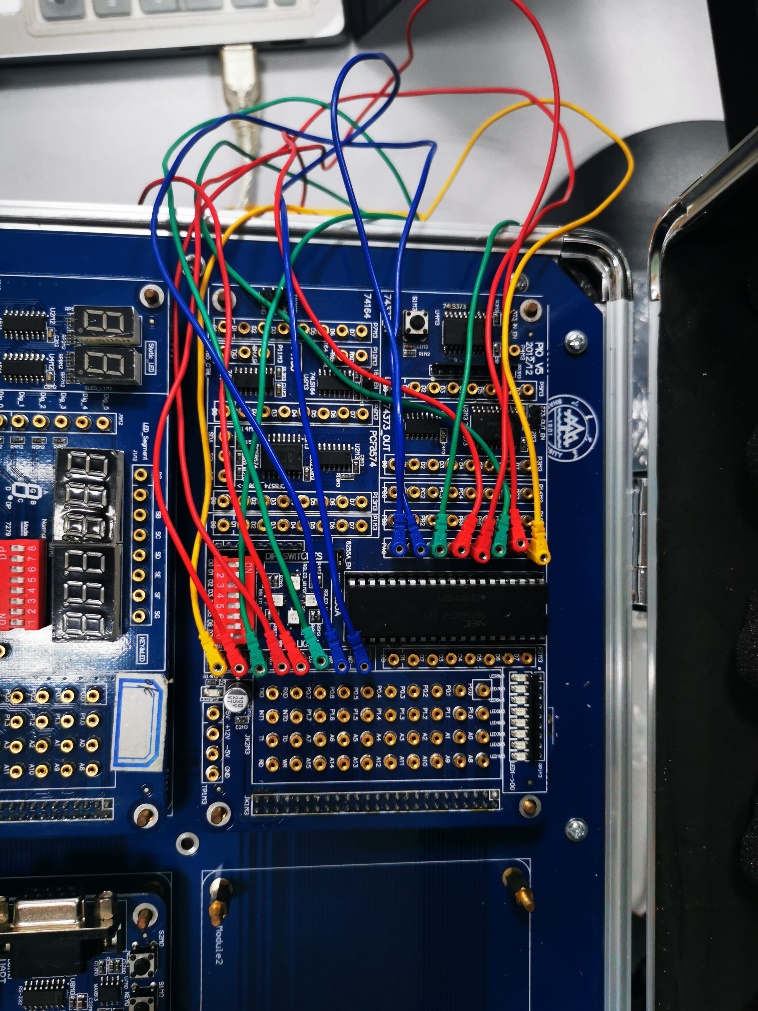
    SBUF = a;        //将变量a的值送给缓存寄存器

    while(!TI)       //等待发送是否完成

    TI=0;            //中断标志位清零

}

2.实验结果：



51 单片机利用可编程IO 扩展芯片8255 的使用方法

① 选择工作模式

② 确定好地址，就和内部IO是一样的

**十一、实验4.9**

1.实验程序：

#include <reg51.h>

#include <intrins.h>

#define uint unsigned int

#define uchar unsigned char

#define LCD\_Data P1

sbit LCD\_RS = P0 ^ 3;

sbit LCD\_WR = P0 ^ 2;

sbit LCD\_EN = P0 ^ 1;

sbit PSB = P0 ^ 0;

uchar code name[] = "李禹申";

uchar code number[] = "201900120123";

uchar code a[] = "123";

/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*延时子程序\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

void delay\_ms(uint xms)

{

    uint j;

    while (xms--)

        for (j = 110; j > 0; j--);

}

void Write\_Data(uchar dat)

{

    LCD\_EN = 0;

    LCD\_WR = 0; //选择写模式

    LCD\_RS = 1; //选择数据模式

    //延时

    delay\_ms(1);

    //将数据准备

    LCD\_Data = dat;

    LCD\_EN = 1;

    \_nop\_();

    \_nop\_();

    LCD\_EN = 0;  //表示开始写入数据,下降沿写入数据

    delay\_ms(2); //延时等待数据全部发送完成

}

//向12864中写入命令

void Write\_Com(uchar com)

{

    //首先我们设置片选信号

    LCD\_WR = 0; //选择写模式

    LCD\_RS = 0; //选择命令模式

    //延时

    delay\_ms(1);

    //将数据准备

    LCD\_Data = com;

    LCD\_EN = 1; //片选信号一般设置为0表示片选

    \_nop\_();

    \_nop\_();

    LCD\_EN = 0;  //表示开始写入数据,下降沿写入命令

    delay\_ms(2); //延时等待数命令全部发送完成

}

//对LCD12864进行初始化

void InitLCD()

{

    PSB = 1; //并口方式

    LCD\_WR = 0;

    LCD\_EN = 0;

    Write\_Com(0x30); //设置为8位并行口，基本指令集动作

    Write\_Com(0x08); //整体显示关，游标关，游标位置关

    Write\_Com(0x01); //清除LCD的显示内容

    Write\_Com(0x06); //显示光标右移，DDRAM位地址加一，画面不移动

    Write\_Com(0x0c); //开显示，关光标

}

//汉字一行能显示8个，字符能显示16个

/\*

第一行：0x80 -> 0x87

第二行：0x90 -> 0x97

第三行：0x88 -> 0x8f

第四行：0x98 -> 0x9f

\*/

unsigned char pos[] = {0x80, 0x90, 0x88, 0x98};

void Display(uchar x, uchar y, uchar code \*str)

{

    uchar i;

    //写入地址

    Write\_Com(pos[x] + y);

    //写入数据

    for (i = 0; str[i] != '\0'; i++)

        Write\_Data(str[i]);

}

int main(void)

{

    //液晶初始化

    InitLCD();

    while (1)

    {

        Display(0, 0, name); //名字写在第一行

        Display(1, 0, number); //学号写在第二行

        Display(2, 0, a);

    }

}

2.实验结果：

LED12864如何显示：

选择写模式，然后找到每一行的地址，将要显示的字符放入到相应的地址中，进行显示。