西安电子科技大学

自主可控嵌入式系统课程实验报告

实验名称 ARMv8 汇编实验

计科 院 2103012 班

姓名 刘一喆 学号 21009201180

同作者 无

实验日期 2023 年 11 月 12 日

实验地点 **线上** 实验批次 **第二批**

成　绩

指导教师评语：

指导教师：

年 月 日

实验报告内容基本要求及参考格式

一、实验目的

二、实验所用仪器（或实验环境）

三、实验基本原理及步骤（或方案设计及理论计算）

四、实验数据记录（或仿真及软件设计）

五、实验结果分析及回答问题（或测试环境及测试结果）

实验一　ARMv8 汇编实验

一、实验目的

1. 尝试 ARMv8 汇编实验环境的搭建
2. 掌握 ARM 汇编的编程方法
3. 掌握 c 语言中嵌入 ARM 汇编的方法
4. 掌握汇编语言程序的编译运行过程

二、实验内容

1.在华为鲲鹏云服务器中搭建 ARMv8 汇编实验环境。

2.编写 hello-world 程序，编译运行。

3.测试样例程序实现通过 C 语言源码来调用汇编源码中的代码。

4.实现在 ARM 平台上通过 C 语言代码内嵌汇编代码的方式，将一个整数类型值，以字节为单位从小端存储转为大端存储。

三、实验步骤

以编写 hello-world 程序为例，简要阐述一下实验的基本流程：

1. 创建存放程序所有文件的目录 hello，并进入该目录，具体命令使用的是 Linux 的基本命令；
2. 创建汇编程序代码文件 hello.s，如果是 c 代码，则后缀名应该为.c。使用 vim 命令创建并进行文本编辑，编写程序；
3. 编译运行。保存并退出 vim 编辑器，按照实验手册对应的命令格式进行编译运行即可。

四、实验结果

三个程序的运行结果如下：

```
[root@ecs-lyzeeel ~]# vim hello.s
[root@ecs-lyzeeel ~]# as hello.s -o hello.o
[root@ecs-lyzeeel ~]# ld hello.o -o hello
ld: warning: cannot find entry symbol _start; defaulting to 00000000004000b0
[root@ecs-lyzeeel ~]# ls
hello  hello.o  hello.s
[root@ecs-lyzeeel ~]# ./hello
Hello World!
[root@ecs-lyzeeel ~]#
```

```
[root@ecs-lyzeeel called]# vim globalCalled.S
[root@ecs-lyzeeel called]# vim globalCalling.c
[root@ecs-lyzeeel called]# ls
globalCalled.S  globalCalling.c
[root@ecs-lyzeeel called]# gcc globalCalling,c globalCalled.S -o called
gcc: error: globalCalling,c: No such file or directory
[root@ecs-lyzeeel called]# gcc globalCalling.c globalCalled.S -o called
[root@ecs-lyzeeel called]# ./called
Original Status:Source string Destination string
Modified Status:Source string Source string
[root@ecs-lyzeeel called]#
```

```
[root@ecs-lyzeeel builtin]# vim globalBuiltin.c
[root@ecs-lyzeeel builtin]# vim globalBuiltin.c
[root@ecs-lyzeeel builtin]# gcc -E globalBuiltin.c -o globalBuiltin.i
[root@ecs-lyzeeel builtin]# gcc -S globalBuiltin.i -o globalBuiltin.s
[root@ecs-lyzeeel builtin]# gcc -c globalBuiltin.s -o globalBuiltin.o
[root@ecs-lyzeeel builtin]# gcc globalBuiltin.o -o globalBuiltin
[root@ecs-lyzeeel builtin]# ./globalBuiltin
out is 78563412
[root@ecs-lyzeeel builtin]#
```

五、心得体会

1.通过本次实验，巩固我对于 ARM 汇编编程的理论学习。具体包括汇编语言编程结构，字符串的基本声明方法，常用指令的使用方法，以及 c 语言调用和内嵌汇编语言的方法。

2.本次实验过程中，由于对汇编语法不熟悉，经常遇到一些语法错误。解决方案是仔细检查代码，逐行排查错误。

3.本次实验，加深了我对嵌入式系统课程内容的理解，还提高了我问题分析和解决的能力。汇编语言的学习是一个渐进的过程，需要坚持不懈的努力和不断的实践。通过这次实验，我为今后更深入的汇编语言编程学习打下了坚实的基础。

六、实验程序

1.Hello-world 实验

```
.text
.global tart1
tart1:
   mov x0,#0
   ldr x1,=msg
   mov x2,len
   mov x8,64
   svc #0


   mov x0,123
```

```
    mov x8,93
    svc #0

.data
msg:
    .ascii "Hello World!\n"
len=.-msg
```

2.c 语言调用汇编程序实验：

```
/************************ globalCalling.c*****************************/
    #include <stdio.h>
    extern void strcpy1(char *d, const char *s);
    int main()
    {
          const char *srcstring="Source string";
          char dststring[]="Destination string";

          printf("Original Status: %s      %s\n",srcstring,dststring);
        strcpy1(dststring,srcstring);
        printf("Modified Status: %s      %s\n",srcstring,dststring);
          return 0;
    }
/************************ globalCalled.S ****************************/
.global strcpy1
# Start the function: strcpy1
strcpy1:
LDRB w2,[X1],#1
STR w2,[X0],#1
CMP w2,#0      //ascii code "NUL" is the last character of a string;
BNE strcpy1
RET
```

3.c 语言嵌入汇编程序实验：

```
    #include <stdio.h>
    int main()
    {
    int val=0x12345678;
```

```
  __asm__ __volatile__(
       "mov x3,%1\n"
   "mov w3,w3, ror #8\n"
   "bic w3,w3, #0x00ff00ff\n"

   "mov x4,%1\n"
   "mov w4,w4, ror #24\n"
   "bic w4,w4, #0xff00ff00\n"

   "add w3,w4,w3\n"
   "mov %0,x3\n"

   :"=r"(val)
   :"0"(val)
   :"w3","w4","cc"
   );

printf("out is %x \n",val);
return 0;
}
```

西安电子科技大学

自主可控嵌入式系统课程实验报告

实验名称 基本接口实验

计科 院 2103012 班

姓名 刘一喆 学号 21009201180

同作者 无

实验日期 2023 年 12 月 9 日

成 绩

实验地点 E-II504 实验批次 **第二批**

指导教师评语：

指导教师：

年 月 日

实验报告内容基本要求及参考格式

一、实验目的

二、实验所用仪器（或实验环境）

三、实验基本原理及步骤（或方案设计及理论计算）

四、实验数据记录（或仿真及软件设计）

五、实验结果分析及回答问题（或测试环境及测试结果）

实验二 基本接口实验

一、实验目的

1.通过实验掌握 ARM 芯片使用 I/O 口控制 LED 显示；

2.了解 ARM 芯片中复用 I/O 口的使用方法；

3.通过实验掌握键盘控制与设计方法。

4.能够编写 ARM 核处理器 S3C2410X 中断处理程序。

二、实验内容

1.运行示例代码，了解键盘和 LED 的程序控制机制。

2.自己编写程序，控制实验平台的发光二极管 LED1、LED2、LED3、LED4，使它们有规律的点亮和熄灭；使用实验板上 5x4 用户键盘，编写程序接收键盘中断；使用键盘控制发光二极管，按照不同模式点亮。

三、实验原理

1. S3C2410X 芯片

S3C2410X 芯片是一种并行输入输出接口，其上共有 71 个多功能的输入输出管脚，他们分为 7 组 I/O 端口。

一个 23 位的输出端口（端口 A）；

两个 11 位的输入/输出端口（端口 B、H）；

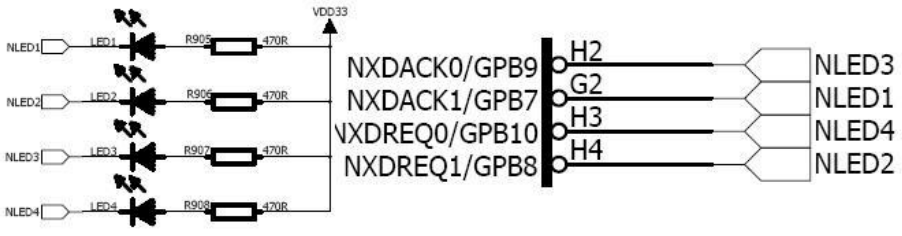
四个 16 位的输入/输出端口（端口 C、D、E、G）；

一个 8 位的输入/输出端口（端口 F）；

由于 S3C2410X 中，大多数的管脚都复用，因此在运行程序之前必须先写相关的控制字，对每个用到的管脚功能进行设置。

2.LED

将 LED1-4 与 S3C2410X 相连，试验箱连的是 GPB7-10（作为输出接口），通过 GPB7-10 引脚的高低电平来控制发光二极管的亮与灭。当这几个管脚输出高电平的时候发光二极管熄灭，反之，发光二极管点亮。连接电路图如下：



3.键盘

试验箱上的键盘使用的是矩阵式键盘，采用的是中断式来读取键值。在键盘按下时产生一个外部中断通知 CPU，并由中断处理程序通过不同的地址读取数据线上的状态，判断哪个按键被按下。

四、实验步骤

1、验证示例源码

验证 02_led_test 子目录下的 led_test. Uv2 例程，编译链接工程；下载到试验箱上，观察运行结果；

验证 12_KeyBoardTest 子目录下的 KeyBoardTest.Uv2 例程，编译链接工程；下载到试验箱上，观察超级终端上返回的结果。

2、设计实现自己的 I/O 控制程序

拷贝示例实验源码工程；

设计程序，实现使用键盘控制发光二极管按照不同模式点亮。

五、实验结果

当按下按键“0”，四个 LED 全灭；当按下按照“1”时，四个 LED 全亮；当按下数字“2”时，四个 LED 为流水灯，循环亮灭。

六、心得体会

1.通过本次实验，学习了嵌入式相关基本硬件的控制原理和操作方法；S3C2410X 芯片是一种输入输出接口，可以在上面接输入输出设备；它的功能与微机课上的 8255 类似，但是它的接口数更多，功能更复杂，配置也更麻烦。键盘是矩阵式键盘，通过中断方式来获取键值。

2.通过本次实验，第一次真正体会到软件程序是如何与硬件联系在一起，也更深刻地理解了 CPU 和接口交互的底层原理，也让我明白了理论与实践结合的重要性，这对于后续的学习将有很大帮助。

七、程序说明

基本接口实验
预先得把 led 相关这三个函数体制到 keyboard 这个程序中。

//核心代码：在 keyboard_test ( ) 函数中添加如下判断语句
if(ucChar == 0)
{ led_off();
}
else if(ucChar == 1)
{led_on();
}
else if(ucChar == 2)
{led_on_off() ;
}

西安电子科技大学

自主可控嵌入式系统课程实验报告

实验名称 人机接口实验

计科 院 2103012 班

姓名 刘一喆 学号 21009201180

同作者 无

实验日期 2023 年 12 月 9 日

成 绩

实验地点 E-II504 实验批次 **第二批**

指导教师评语：

指导教师：

年 月 日

实验报告内容基本要求及参考格式

一、实验目的

二、实验所用仪器（或实验环境）

三、实验基本原理及步骤（或方案设计及理论计算）

四、实验数据记录（或仿真及软件设计）

五、实验结果分析及回答问题（或测试环境及测试结果）

实验三 人机接口实验

一、实验目的

1.通过实验掌握液晶显示文本及图形的方法与程序设计；
2.通过实验掌握触摸屏（TSP）的设计与控制方法。
3.尝试将触摸屏作为输入设备，LCD 作为输出设备的交互应用。

二、实验内容

1.掌握液晶屏作为人机接口界面的设计方法，并编写程序实现；
2.编程实现触摸屏坐标转换为液晶对应坐标；
3.编程实现由液晶屏和触摸屏构成的可以互动的人机界面，至少实现 3 屏。

三、实验原理

1.液晶屏（LCD）

液晶屏（LCD）主要用于显示文本及图形信息。液晶显示屏具有轻薄、体积小、低耗电量、无辐射危险、平面直角显示以及影像稳定不闪烁等特点，因此在许多电子应用系统中，常使用液晶屏作为人机界面。

实验用的 LCD 屏像素：320*240,具体解释如下：

横坐标范围：从最左边的像素位置 (0) 到最右边的像素位置 (320 - 1)。

纵坐标范围：从最上方的像素位置 (0) 到最下方的像素位置 (240 - 1)。

像素点颜色可以通过（R,G,B）三元组来表示，其中 R 代表红色通道的亮度,G 代表绿色通道的亮度，B 代表蓝色通道的亮度。在 16 位 TFT 液晶屏中，通常使用 565 格式来表示颜色。在 RGB565 格式中，一个像素用 16 位来表示，其中 5 位用于红色（R），6 位用于绿色（G），5 位用于蓝色（B）。示意图如下：

VD Pin Connections at 16BPP

(5:6:5)

VD	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
RED	4	3	2	1	0	NC									NC							NC		
GREEN									5	4	3	2	1	0										
BLUE																	4	3	2	1	0			

从图中不难推出一些纯色的 RGB 值如下：

红色：0xF800（二进制：11111 000000 00000）

绿色：0x07E0（二进制：00000 111111 00000）

蓝色：0x001F（二进制：00000 000000 11111）

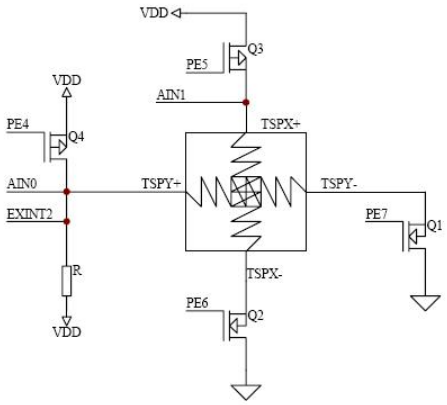
白色：0xFFFF（二进制：11111 111111 11111）

黑色：0x0000（二进制：00000 000000 00000）

2. 触摸屏（TSP）

触摸屏（TSP）按其技术原理可分为五类：矢量压力传感式、电阻式、电容式、红外线式和表面声波式，其中电阻式触摸屏在嵌入式系统中用的较多。

实验系统由触摸屏、触摸屏控制电路和数据采集处理三部分组成。根据按压点的相关电路参数，可以转换出该点对应的坐标。



触摸屏等效电路

四、实验步骤

1、验证示例源码

验证实验例程目录 11_LCD_Test 子目录下的 LCD_Test.Uv2 例程，编译链接工程，下载程序到试验箱，观察运行结果；

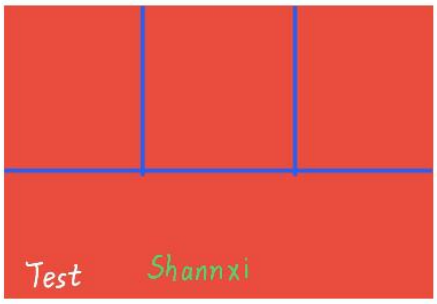
验证实验例程目录 07_TSP_Test 子目录下的 TSP_Test.Uv2 例程，编译链接工程，下载程序到试验箱，观察超级终端返回的结果。

2、设计实现自己的人机互动界面程序

在示例实验源码 TSP_Test.Uv2 的基础上，设计程序，实现由液晶屏和触摸屏构成的可以互动的人机界面，至少实现 3 屏。

五、实验结果

最终若按在区域 1，则显示 Shannxi；若按在区域 2，则显示 Xi’an，若按在区域 3，则显示 Xidian；示意图如下（从左至右依次为区域 1,2,3）：



六、心得体会

通过本次实验，理解了触摸屏和液晶屏的基本工作原理，对于二者的协同工作进行了一定的尝试，设计了一个简易的人机交互程序。

本次实验遇到了一些 c 程序语法上不熟悉的问题，例如对字符串的声明和赋值不清楚，造成编译报错，后续得多复习巩固。

七、程序说明

人机接口使用

预先得把相关的 lcd 相关代码导入到 tsp 例程代码目录下，然后在相应的源文件下用 extern 声明外部函数来使用

```
//核心代码：修改 color_lcd_test()如下：
void color_lcd_test(int x,int y){
    Lcd_Clear(0xf800); // 刷新屏幕，设置背景颜色为红色
    /*下面三行代码为用蓝色线限定三个方形区域*/
     Lcd_Draw_Line(0, 140, 320, 140, 0x001F, 5);
     Lcd_Draw_Line(120, 0, 120, 140, 0x001F, 5);
     Lcd_Draw_Line(220, 0, 220, 140, 0x001F, 5);
     Lcd_DspAscII8x16(20, 220, 0xFFFF, "Test"); // 在左下角显示字符串"Test"，颜色为白色
if (x < 500 && y < 300 && y > 0) {
     Lcd_DspAscII8x16(120, 200, 0x7e0, "Shannxi"); // 显示字符串“Shannxi”，颜色为绿色
} else if (x < 500 && y < 600 && y > 300) {
     Lcd_DspAscII8x16(120, 200, 0x7e0, "Xi'an"); // 显示字符串“Xi'an”，颜色为绿色
} else if (x < 500 && y < 900 && y > 600) {
     Lcd_DspAscII8x16(120, 200, 0x7e0, "Xidian"); // 显示字符串“Xidian”，颜色为绿色
}
}
//最后在 tsp_test()中加入 color_lcd_test(g_nPosX,g_nPosY);即可
```

西安电子科技大学

自主可控嵌入式系统课程实验报告

实验名称_μCOS-II 系统原理实验

计科 院 2103012 班

姓名 刘一喆 学号 21009201180

同作者 无

实验日期 2023 年 12 月 23 日

成 绩

实验地点 E-II504 实验批次 第二批

指导教师评语：

指导教师：

年 月 日

实验报告内容基本要求及参考格式

一、实验目的

二、实验所用仪器（或实验环境）

三、实验基本原理及步骤（或方案设计及理论计算）

四、实验数据记录（或仿真及软件设计）

五、实验结果分析及回答问题（或测试环境及测试结果）

实验四　μCOS-II 系统原理实验

一、实验目的

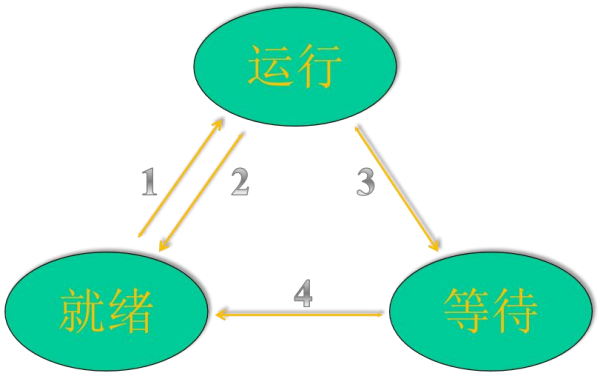
1.理解任务管理的基本原理，了解任务的各个基本状态及其变迁过程；
2.掌握μCOS-II 中任务管理的基本方法，包括创建、启动、挂起、解挂任务等；
3.掌握μCOS-II 中任务使用信号量的一般原理。

二、实验内容

uC/OS 中任务管理函数的正常运行，并通过实现 "哲学家就餐" 问题，验证信号量对任务间互斥访问的支持。另外，应用信号量实现任务间的同步，设计并顺序执行七个任务。

三、实验原理

1. μCOS-II 中任务的状态
就绪态（Ready）： 任务已经准备好执行，但还未被调度执行。
阻塞态（Blocked）： 任务由于某些原因（例如等待某个事件、信号量或消息）而暂时无法执行，进入阻塞状态。
运行态（runnning）：该任务以及获得了运行所必要的资源，它的程序正在处理机上运行。
挂起态（Suspended）： 任务被显式地挂起，不参与调度。



任务状态转换示意图

2. μCOS-II 中的相关函数

```
OSTaskCreate();//创建任务
OSTaskSuspend(task_id);   // 挂起任务
OSTaskResume(task_id);    // 恢复任务
OS_EVENT *OSSemCreate(INT16U cnt);//创建信号量
```

```
OSSemPend(sem, timeout, &err);       // P 操作
OSSemPost(sem);                      //V 操作
OSTimeDly(delay);     //  使任务延时一段时间
OSStart();                  //多任务的启动
```

四、实验步骤

1、验证示例源码

拷贝整个实验例程源码目录到自己的实验文件目录下；

使用 μ Vision IDE for ARM 通过 ULINK2 仿真器连接实验板，打开实验例程目录 04-uCOS\2.1_Task_test 子目录下的 ucos2.Uv2 例程，编译链接工程；

将程序烧写到实验平台的 NorFlash 中，观察串口输出； 打开实验例程目录\04-uCOS\2.3_Semaphore_test 子目录下的 ucos2.Uv2 例程，编译链接工程；

烧写调试，观察结果。

2. 设计实现一个多任务应用程序

拷贝示例实验源码工程，在示例代码的基础上，设计 7 个任务，并用信号量实现 7 个任务顺序执行，将执行结果在串口上打印出来。

五、实验结果

从超级终端上可以看到回馈的结果：可知七个任务顺序执行。

六、心得体会

本次实验对μCOS-II 操作系统的多任务处理（创建、删除、挂起等）、多任务调度、信号量等机制有了更深刻的理解，巩固了我理论课所学知识，对后续基于多任务处理的程序设计有很大的帮助。

西安电子科技大学

自主可控嵌入式系统课程实验报告

实验名称 简易计算器的设计

计科 院 2103012 班

姓名 刘一喆 学号 21009201180

同作者 无

实验日期 2023 年 12 月 23 日

成 绩

实验地点 E-II504 实验批次 **第二批**

指导教师评语：

指导教师：

年 月 日

实验报告内容基本要求及参考格式

一、实验目的

二、实验所用仪器（或实验环境）

三、实验基本原理及步骤（或方案设计及理论计算）

四、实验数据记录（或仿真及软件设计）

五、实验结果分析及回答问题（或测试环境及测试结果）

实验五 简易计算器的设计

一、实验目的

1.理解任务管理的基本原理，掌握μCOS-II 中任务管理的基本方法；
2.掌握μCOS-II 中任务间通信的一般原理和方法；
3.掌握嵌入式系统中 LCD 与键盘控制的一般方法。
4.掌握嵌入式系统中程序设计的基本格式与规范。

二、实验内容

本实验旨在通过对嵌入式系统设计和μCOS 操作系统的理解，实现一个简单的计算器功能。从键盘上输入的运算表达式中获取对应的操作数和运算符，并实现基本的加减乘运算，最终在屏幕上显示运算结果。

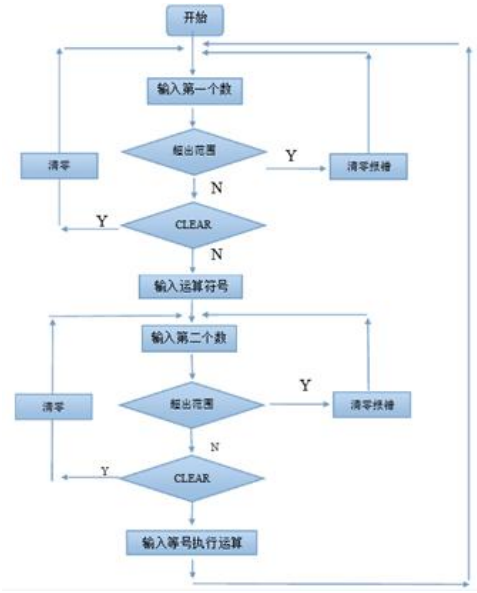
三、实验原理

1 系统架构

实验系统采用了μCOS-II 操作系统，通过任务调度实现多任务处理。主要包括两个任务：一个任务用于接收键盘输入的字符并将其转换为数字，计算运算结果；另一个任务负责将计算结果转为字符串送至 LCD 屏幕显示。

2 任务 1：字符接收与运算任务

该任务负责接收来自键盘的输入，根据输入的字符判断是操作数还是操作符，并将其保存下来。通过状态机或类似的方法实现运算或转换过程。实验基本流程图如下：



3 任务 2：显示任务

该任务在接收到等号操作符时，执行相应的运算，并将操作数，运算符，结果转为字符串，进而显示在屏幕上。

四、实验过程

1 预先准备

拷贝示例实验源码工程 3.3_keyboard_test 到自己实验所在的路径下，接好必要的线，将示例代码烧录到 Flash 上，在实验箱上检查 LCD 屏是否正常工作，通过观察超级终端上的反馈值确定串口和键盘是否正常工作。

2 程序设计与运行

自行设计字符接收与运算任务和显示任务的代码，编译程序。将程序代码烧录到 Flash 中，在试验箱上测试，确保程序可以正常运行。

五、实验结果

通过实验，成功实现了基本的计算器功能。能够执行简单的运算，并将结果显示在屏幕上。加法、减法和乘法的测试样例如下图所示：

六、实验心得

1.本实验成功地实现了一个简单的嵌入式系统计算器功能，利用μCOS-II 操

作系统实现了字符转换和运算显示的多任务处理。在实现过程中，不仅考虑了实现计算器所要有的逻辑功能，对任务同步、互斥访问等问题也进行了充分考虑。

2.本次实验多个任务之间的信息交互是通过全局变量实现的。即定义几个全局变量例如操作数，结果等，在两个任务之间通过 extern 关键字来引用其他任务中已经定义的全局变量。但由于全局变量被两个任务所共享，因此对其进行一定的互斥和同步处理可能会更好。

3.本次实验，锻炼了我分析问题和解决问题的能力。在实验过程中，我遇到了一些调试方面的问题。对于编译不可通过的问题，比如我遇到了数据声明必须出现在函数的开始这样的提示，那么根据提示分析和修改代码即可；如果编译通过能在实验板上运行，说明代码语法没问题，就是逻辑功能上的问题，比如我的计算器计算结果如果为 0 时不会显示，检查代码发现在我进行数字转换为字符串时，运算结果为 0 无法进入循环，因此这部分代码就需要改进。

4.实验改进思路：我的计算器只能实现两个数字的运算，如果设计多个数字的运算，本质上即计算一个中缀表达式，可以考虑使用栈来实现。

5.通过这次实验，我不仅巩固了理论课上关于嵌入式 c 编程和μCOS 操作系统相关知识，也体会到了嵌入式应用设计的挑战与乐趣。能够亲自动手设计、编程并调试这个简易计算器，让我收获良多。

七、程序说明

计算器设计

全局变量介绍

```
int x;//操作数 1
int y;//操作数 2
unsigned char operation;    //  操作符
unsigned char resultSign; //  结果符号
int result0;//运算结果
```

/**********字符接收与运算任务**********/

```
/*typedef unsigned int UINT32T;        typedef unsigned char UINT8T;*/
//定义的变量
UINT8T flag;
//下述代码加在 keyboard_test()中
      if (ucChar >= '0' && ucChar <= '9') {
            int digit = ucChar - '0';
            if (flag== 0) {
                  x = x * 10 + digit; //  输入第一个数
            } else {
                  y = y * 10 + digit; //  输入第二个数
            }
```

```
        } else if (ucChar == '+' || ucChar == '-' || ucChar == '*') {
            operation = ucChar;
            flag = 1;
    } else if (ucChar == '=') {
    resultSign = '+';
            switch (operation) {
                case '+':
                    result0 = x + y;
                    break;
                case '-':
                    resultSign = (x < y) ? '-' : '+';
                    result0 = (x < y) ? y - x : x - y;
                    break;
                case '*':
                    result0 = x * y;
                    break;
                default:
                    break;
            }
        }
```

/*****************************显示任务*****************************/

```
    //定义的变量与函数
    unsigned char resultStr[10];
    unsigned char opStr[1];
    UINT8T p = 0;
    UINT32T t;
    UINT8T i;
    //  结果转换为字符串函数
    unsigned char* exchangeResult(UINT32T number) {
        if (resultSign == '-') {
            if (number == 0) {
                //  处理 number 为 0 的情况
                resultStr[0] = '0';
                resultStr[1] = '\0';
                p = 0;
            } else {
                resultStr[p] = resultSign;
```

```
                p++;

                // 将数字转换为字符串
                while (number != 0) {
                    resultStr[p] = number % 10 + '0';
                    number /= 10;
                    p++;
                }
                resultStr[p] = '\0';
                for (i = 0; i < p / 2; i++) {
                    t = resultStr[i];
                    resultStr[i] = resultStr[p - i - 1];
                    resultStr[p - i - 1] = t;
                }
                p = 0;
            }
        } else if (mark == '+') {
        while (num != 0) {
            str[p] = num % 10 + '0';
            num /= 10;
            p++;
        }
        str[p] = '\0';
        for (i = 0; i < p / 2; i++) {
            temp = str[i];
            str[i] = str[p - i - 1];
            str[p - i - 1] = temp;
        }
        p = 0;
}
        return resultStr;
}

// 操作数转换为字符串函数
unsigned char* exchangeInput(UINT32T number) {
        // 将数字转换为字符串
        while (number != 0) {
            resultStr[p] = number % 10 + '0';
```

```
            number /= 10;
            p++;
        }
        // 在字符串末尾添加字符串终止标记
        resultStr[p] = '\0';
        for (i = 0; i < p / 2; i++) {
            t = resultStr[i];
            resultStr[i] = resultStr[p - i - 1];
            resultStr[p - i - 1] = t;
        }
        p = 0;
        return resultStr;
}

/*下述代码加 color_lcd_test()中*/
/*前两项是坐标，第三项为颜色是绿色，最后一项为要输出的字符串*/
sprintf(opStr,"%c",operation);        //将运算符转为字符串，以便于输出
Lcd_DspAscII8X16(10,100,0x7e0, exchangeInput(x));
Lcd_DspAscII8X16(50,100,0x7e0, opStr);
Lcd_DspAscII8X16(100,100,0x7e0, exchangeInput (y));
Lcd_DspAscII8X16(150,100,0x7e0,"=");
Lcd_DspAscII8X16(200,100,0x7e0, exchangeResult (result0));
```