**中山大学数据科学与计算机学院本科生实验报告**

**（2016学年秋季学期）**

课程名称：操作系统实验 任课教师：凌应标 教学助理（TA）：

|  |  |  |  |
| --- | --- | --- | --- |
| 年级 | **15级** | 专业（方向） | **软件工程（移动信息工程）** |
| 学号 | **15352461** | 姓名 | **宗嘉希（组长）** |
| 学号 | **15352443** | 姓名 | **钟凌山** |
| 学号 | **15352448** | 姓名 | **周禅城** |
| 电话 | **18022724490** | Email | **zongjx@mail2.sysu.edu.cn** |
| 开始日期 | **2017.04.07** | 完成日期 | **2017.04.17** |

# 【实验题目】

用 C 和汇编设计内核

# 【实验目的】

1. 将实验二的原型操作系统分离为引导程序和 OS 内核，由引导程序加载内核， 用 C 和汇编实现操作系统内核

2. 扩展内核汇编代码，增加一些有用的输入输出函数，供 C 模块中调用

3. 提供用户程序返回内核的一种解决方案

4. 在内核的 C 模块中实现增加批处理能力

# 【实验要求】

1. 在磁盘上建立一个表，记录用户程序的存储安排

2. 可以在控制台命令查到用户程序的信息，如程序名、字节数、在磁盘映像文件中的 位置等

3. 设计一种命令，命令中可加载多个用户程序，依次执行，并能在控制台发出命令

4. 在引导系统前，将一组命令存放在磁盘映像中，系统可以解释执行

# 【实验方案】

## 一、 硬件及虚拟机配置

## 硬件：操作系统为 win10 的笔记本电脑

## 虚拟机配置：无操作系统，10MB 硬盘，4MB 内存，启动时连接软盘

## 二、 软件工具及作用

## Nasm:用于编译汇编程序，生成.bin 文件

## WinHex:用于向软盘写入程序

## VMware Workstation 12 Player：用于创建虚拟机，模拟裸机环境

## Notepad++: 用于编辑汇编语言文件

TCC：用于编译C文件，生成OBJ文件

## TASM：用于编译ASM文件，生成OBJ文件

TLINK：用于把两个OBJ文件连接，生成COM文件

Dosbox：用于提供16位运行环境，为TCC+TASM+TLINK编译链接提供一个环境

# 【实验过程】

## 首先，配置好虚拟机，创建好一个1.44MB的虚拟软盘并连接到虚拟机。

## 然后我们首先设计了四个不一样的有输出的用户可执行程序的代码，分别编译出四份可执行程序。四个程序的内容是：

1、程序显示在屏幕的左上角的四分之一的位置，用字符 ‘Z’从屏幕左上角位置 45 度角下斜射出，保持一个可观察的适当速度直线运动，碰到屏幕相应 1/4 区域的边后产生反射，改变方向运动，如此类推，不断运动。在最左上角打印出组员1的姓名和学号。

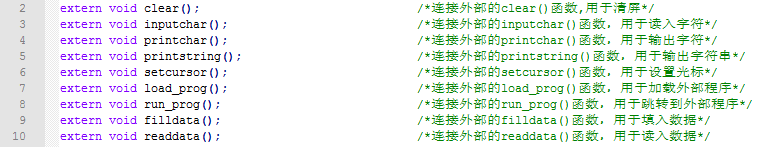
2、程序显示在屏幕的右上角的四分之一的位置，用字符 ‘Z’从屏幕右上角位置 45 度角下斜射出，保持一个可观察的适当速度直线运动，碰到屏幕相应 1/4 区域的边后产生反射，改变方向运动，如此类推，不断运动。在最右上角打印出组员2的姓名和学号。

3、程序显示在屏幕的左下角的四分之一的位置，用字符 ‘Z’从屏幕左上角位置 45 度角上斜射出，保持一个可观察的适当速度直线运动，碰到屏幕相应 1/4 区域的边后产生反射，改变方向运动，如此类推，不断运动。在最左下角打印出组员3的姓名和学号。

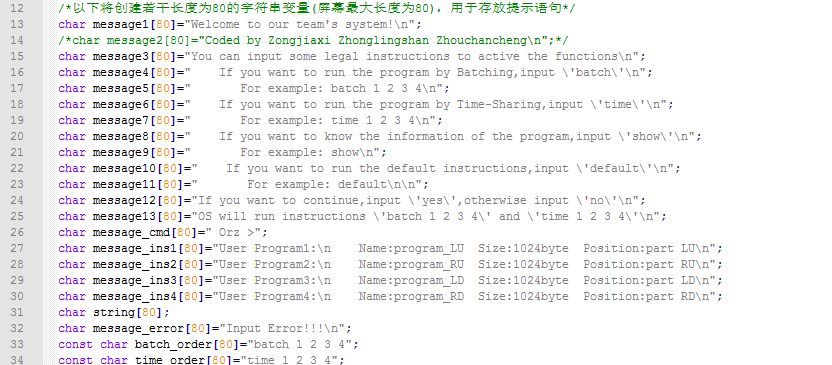
4、程序显示在屏幕的右下角的四分之一的位置，用字符 ‘Z’从屏幕左上角位置 45 度角上斜射出，保持一个可观察的适当速度直线运动，碰到屏幕相应 1/4 区域的边后产生反射，改变方向运动，如此类推，不断运动。在最右下角打印出所有组员的姓名和学号。

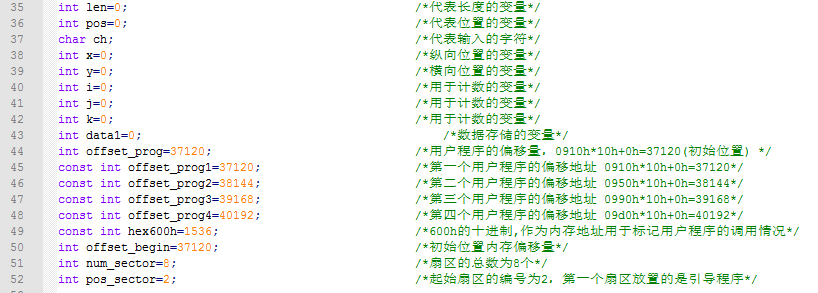
这四个程序跟实验二的程序内容上是差不多的，但是由于我们将要用到分时运行，所以会在程序里做出一些改动，这将会在实验报告后面讲到。

下一步我们就要开始设计c程序和os内核了 ，首先要确定要做什么功能，由于汇编学的不是很好所以我们就决定模仿师兄的程序里的功能，实现批处理功能和分时功能，虽然不是很懂这些名词是什么意思，但是我经过自己的理解后，认为批处理是按顺序运行几个程序，当一个程序运行完了，再运行下一个程序；分时是同时运行几个程序。

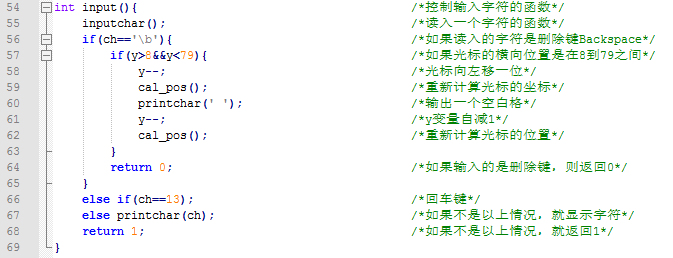
 首先决定实现程序要使用哪些功能，一共有以下几个函数，先把声明写在c语言文件的开头，使用extern标志来定义，因为这些函数的实现是写在与此c文件关联的汇编文件中的。

把以上函数声明好后就要声明出要显示的字符串：

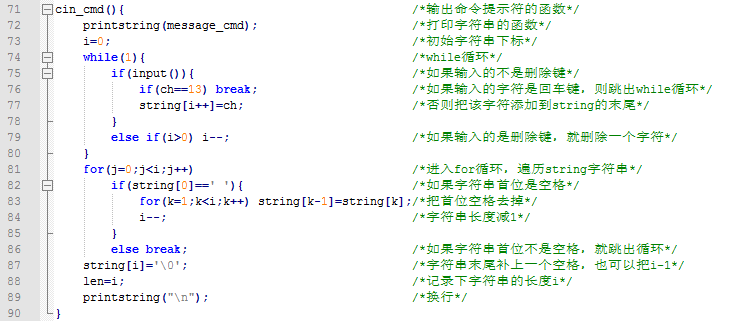


还有需要用到的变量：

接下来就要定义一些在c语言里面比较容易实现的函数了：

 输入数据的函数：

调用输入命令的函数：



计算光标的坐标的函数：

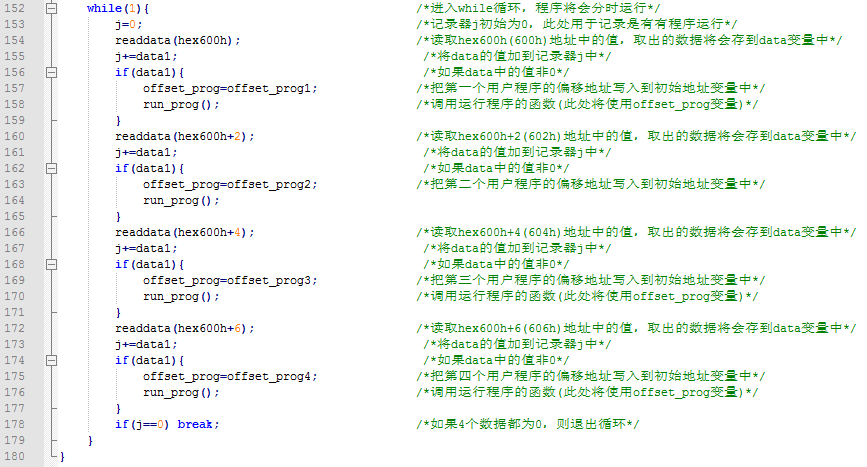
## QQ截图20170416210603

实现批处理功能命令的函数：

## QQ截图20170416210658

实现分时功能命令的函数：

## QQ截图20170416210807



## QQ截图20170416211022主函数：

## QQ截图20170416211047

## QQ截图20170416211100

## 以上为止，c函数的编写就结束了，每一条语句的具体功能都在代码上有注释（如上）。

## 在这一个c程序中实现的其实是等于前一次实验的引导扇区的功能，输出提示字符，然后提供输入命令的功能和显示信息的功能。

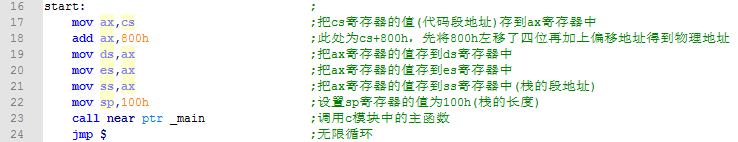
## 完成c文件后，就要着手去写和它相呼应的汇编文件了。

## QQ截图20170416212219首先还是先要用extrn标识符去定义一些在c文件里面的变量，作用是与c文件里面一样的，都是关联两个文件，使这些变量可以互用。要注意，在这里定义变量的时候要在变量名前加一个下划线“\_”，因为c文件编译成obj文件的时候，变量名前都会加上“\_”（具体原因我不清楚），因此要加上才能关联上对应的变量。

## 

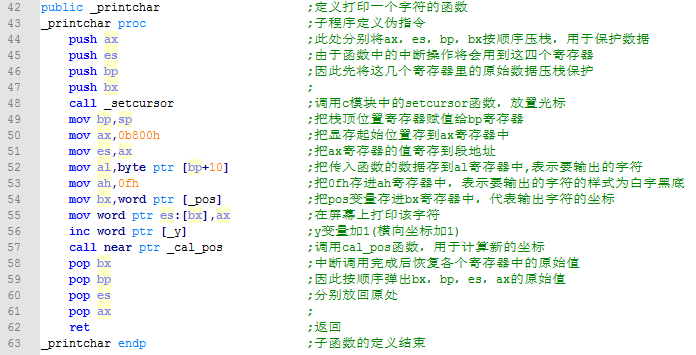
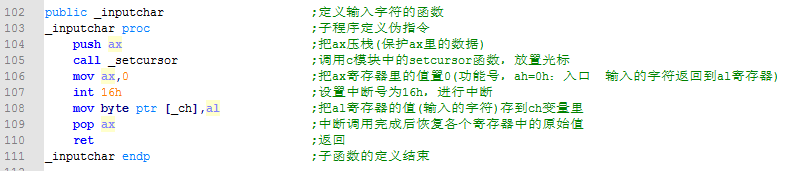
## QQ截图20170416212819设置好段定义：

## 

 开始程序，把程序放置到合适的地址：

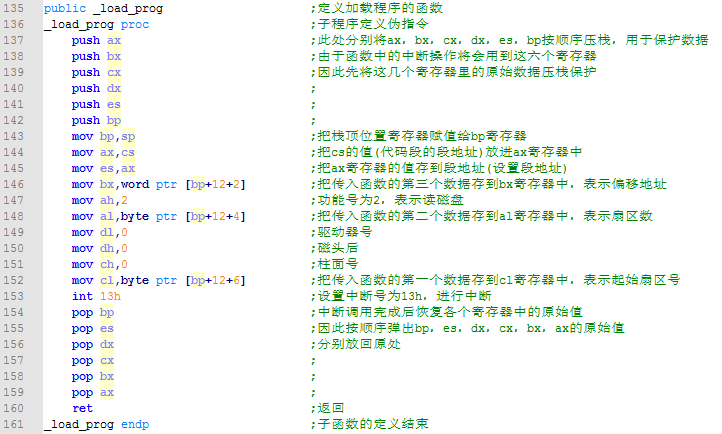
## QQ截图20170416213109 下面是C语言中关联的各个函数的定义：

## 



## QQ截图20170416213322

## QQ截图20170416213354

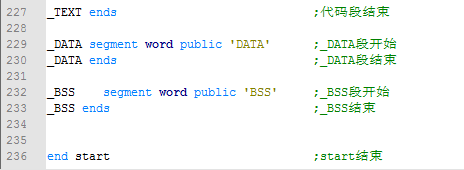


## QQ截图20170416213520

## QQ截图20170416213540

## QQ截图20170416213548

以上就是所有函数的定义了，定义完成后结束数据段：

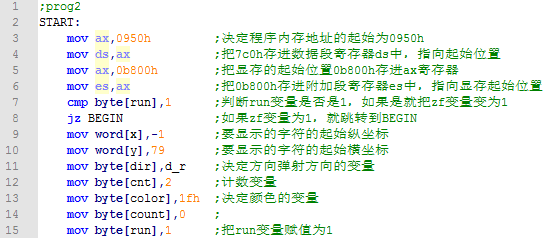


## 至此，与c文件关联的汇编文件也完成了，这两个文件编译并关联后就是这次实验的“内核”了。

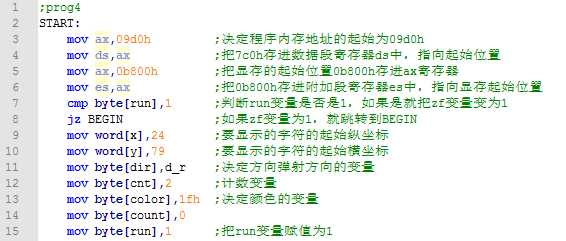
## QQ截图20170416214227然后就要编写引导程序了，这一次的引导程序只需要把“内核”放在正确的内存地址中就可以了，还要把四个用户程序的bin文件和内核com文件包入其中：

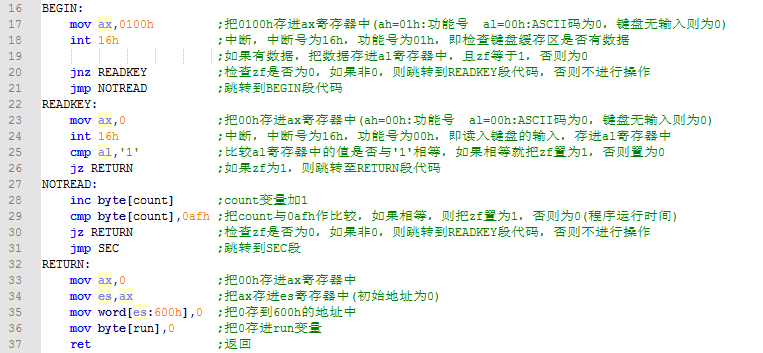
至此，所有的程序就已经完成了，真是一个很痛苦的过程！

## QQ截图20170416214545如果要实现批处理和分时功能，用户程序上也要动一些手脚，首先是程序存放的地址，由于要分时运行，所以要分配不同的内存地址。



## QQ截图20170416214829



还有判断是否分时运行的代码段：

其中count变量用作计时，程序运行一定时间就会自动停止。

至此，代码部分的内容全部结束，接下来我们就进入到编译过程的部分：

代码都放在d盘的exp3文件夹，首先把四个用户程序使用nasm编译成bin文件：

## QQ截图20170416215435

编译完成！

## QQ截图20170416215719接下来就要使用TCC和TASM分别去编译c文件和相关联的汇编文件了，在使用TLINK去连接两个OBJ文件。但是我发现在我的64位系统里面不能够运行TCC和TASM 和TLINK编译器，因此我们使用一个叫做DOSBOX的工具去模拟16位的计算机环境，从而进行编译。

## QQ截图20170416215910首先把我的放置编译器和代码的文件地址导入到DOSBOX的虚拟盘中，再把DOSBOX的当前地址切换到该地址：

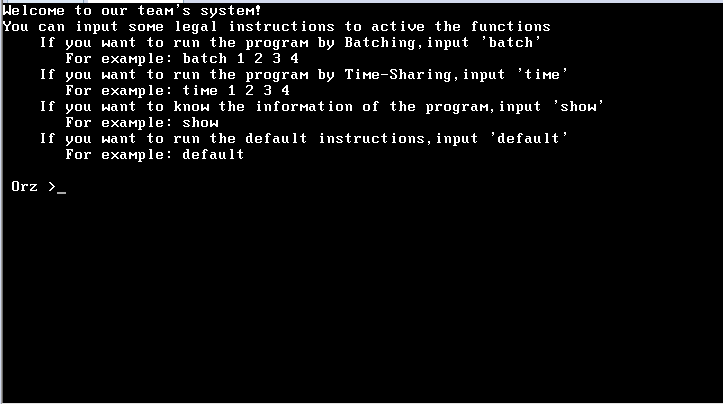
## QQ截图20170416220146然后就可以进行编译了：

从以上图片我们可以看到，c文件和汇编文件都编译出OBJ文件了，并且两个OBJ文件都连接起来了，生成了“内核”OS1.COM文件。然后再使用nasm编译loader1.asm文件，最后一步完成！

## QQ截图20170416220417

## 得到的loader1.bin就是最后的文件了！把里面的内容复制到1.44mb的虚拟软盘中，链接到虚拟机上，就可以运行了！！！

下面为运行结果的实现：

初始引导界面：

## QQ截图20170416220639输入show命令，显示用户程序的属性内容：

## 输入批处理指令（运行1，4程序）：QQ截图20170416220859

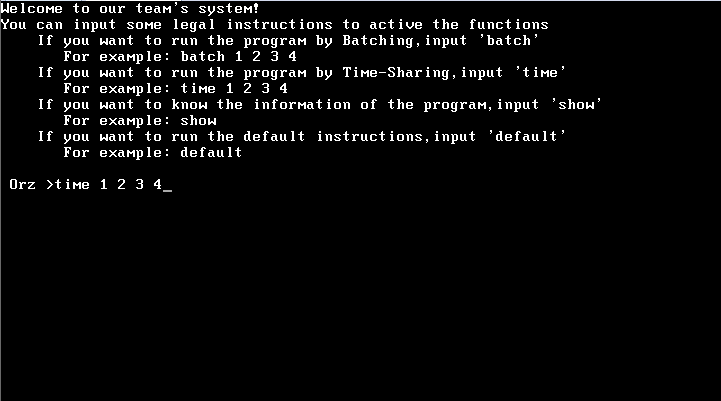
## QQ截图20170416220910刚开始运行时：

运行第二个程序：

## QQ截图20170416220919

# QQ截图20170416220929运行完毕，输出提示是否继续运行（其实输入yes或no都一样，返回主界面）：

## 输入分时指令（运行1，2，3，4程序）：



## QQ截图20170416221453刚开始运行时：

运行结束：

## QQ截图20170416221559

## QQ截图20170416220838如果输入的命令有误，输出错误提示，弹出重新输入的命令：

输入default命令，先运行批处理，后运行分时：

## QQ截图20170416221739

## QQ截图20170416222012运行时：

# QQ截图20170416222038

# 至此，程序演示结束！

# 【实验总结】

宗嘉希 15352461： 这一次的实验让我学习到了很多东西，也解决了我上一节课的一个疑问：既然引导扇区放置引导程序，而引导扇区的大小却只有512字节，那么如果我想在引导扇区里面实现更多的功能那该要怎么办？引导程序输出几个字符串就差不多写满了，那要怎么样去实现更多的功能呢？这一次实验给出的答案就是把引导程序写成一个“内核”，再写一个引导程序把这一个内核导入到内存中运行就可以了。那么这个内核必然是要实现很多的功能的，但是学习了汇编之后，都知道要是是用汇编去实现这么多功能是有难度的。因此我们就可以使用c语言和汇编交叉调用的方式去写这一个“内核”，这跟用纯汇编语言编写程序比起来要简单一点。

为什么要用两种语言呢？老师在课堂上已经解释了，首先，使用汇编语言是必须的，因为它有几个必备功能：①设置自身运行模式和环境，需要设置硬件寄存器，②设置I/O端口实现I/O操作，③初始化中断向量表和实现中断处理，④实现控制原语。其次，使用c语言也是有它的好处的：①适合构造复杂的数据结构和相关数据结构的管理，②实现复杂的功能或算法。

因此，函数的声明和调用就要在实现复杂功能的c语言模块里面，而这些函数的实现就要在汇编语言模块里面了，因为要使用中断，I/O操作等等。

根据以上这一些原理，还有对师兄的代码的参考，我们编写出了一个像师兄一样的除了能实现实验的基本要求功能，还能实现批处理与分时功能的程序。这一个过程真的是很辛苦，因为对代码的不熟悉和粗心大意，导致要花很多时间去学习，后期调试的时候还出现了很多的错误要去排除，比如说输出了乱码，才发现自己的函数的调用出现了错误，导致输出的字符错误，还有段地址设置错误等问题导致用户程序不能够正确地运行。在输入命令的时候也出现了一些小问题，不过并不影响整体程序的正确运行。

总的来说，这一次的实验虽然挺难的，知识点也挺多的，但是学习到的东西也很多。原来c语言和汇编交叉编译的功能真的挺大的，可惜我的时间与能力不足，不能写出更多的功能。希望能在以后继续努力吧！

## 钟凌山 15352443：这次的实验要求我们用汇编和 C 编写一个引导程序，引导操作系统内核运行。操作系统要求能够根据输入的指令来调用不同的用户程序来运行。

## 首次运行的时候就遇到了不小的问题。运行时屏幕上出现了几行乱码，进行输入后乱码消失，但是并没有进入系统。后来仔细检查所有代码后发现是因为段定义处少了一句（DGROUP）。

## 本次实验需要更深入地了解磁盘扇区的概念，以及熟悉读取用户程序的方法。

周禅城 15352448：不得不说，相比上一次实验，这次实验的难度真可谓是飞跃啊！！！！！这次实验主要是要设计操作系统的内核，除了要实现一些基本功能以外，还要拓展内核代码，增加一些有用的输入输出函数以及实现批处理能力和分时能力。与很多其他同学一样，我们看到实验题目的第一感觉也是：题目要我们做什么？实话实说，一开始我们真的无从下手，完全不知道要做什么。后来我们觉得非常有必要先学习一下。于是我们找到了王烁辰师兄这次实验的实验报告，对他的实验报告进行了深入的学习。通过王师兄的报告，我们知道了王师兄是如何在内核里实现批处理和分时功能的。基于对王师兄报告的学习和我们对本次实验的理解，我们便着手编写我们自己的操作系统内核。虽然是用我们比较熟悉的C语言编写，但是在实验过程中我们遇到了非常多的问题，我们一边继续学习王师兄的报告，一边讨论，完成代码的修改，解决一个又一个问题，其中的辛苦，只有真正去做的人才知道。这次实验虽然耗费了我们很多的时间和精力，但是对于加深对操作系统内核的理解可谓是大有裨益。

## 附录：

**asm文件：**

## loader1.asm

prog1.asm

## prog2.asm

prog3.asm

## prog4.asm

os1.asm

## **c文件：**

cpro.c

## **bin文件：**

loader1.bin

## prog1.bin

prog2.bin

## prog3.bin

Prog4.bin

## **obj文件：**

os1.obj

## cpro.obj

**com文件：**

## cpro.com

**img文件：**

## 144mb.img