**《操作系统实验》**

**实验报告**

**（实验四五）**

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| **学院名称** | **：** | 数据科学与计算机学院 | | | | | |
| **专业（班级）** | **：** | 16计科2班 | | | | | |
| **学生姓名** | **：** | 杨志成 | | | | | |
| **学号** | **：** | 16337281 | | | | | |
| **时间** | **：** | 2018 | 年 | 3 | 月 | 17 | 日 |

**目录**

**一，实验目的-----------------------------------3**

**二，实验要求-----------------------------------3**

**三，实验方案-----------------------------------4**

**（1）基础原理----------------------------4**

**（2）实验工具与环境----------------------7**

**（3）程序流程----------------------------8**

**（4）程序模块功能------------------------9**

**（5）代码文档组成------------------------9**

**（6）实现效果----------------------------10**

**四，实验过程及实验结果-------------------------12**

**五，实验总结-----------------------------------15**

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
|  |  |  | **成绩** | **:** |  |
| **实验五** | **：** | 中断机制编程技术 | | | |

**一 实验目的**

**1在内核实现多进程的二状态模，理解简单进程的构造方法和时间片轮转调度过程。**

**2解释多进程的控制台命令，建立相应进程并能启动执行。**

**3至少一个进程可用于测试前一版本的系统调用，搭建完整的操作系统框架，为后续实验项目打下扎实基础。**

**二 实验四/五的目的和要求(合并):**  
**1掌握pc微机的实模式硬件中断系统原理和中断服务程序设计方法，实现对时 钟、键盘/鼠标等硬件中断的简单服务处理程序编程和调试，让你的原型操作系统在运行以前己有的用户程序时，能对异步事件正确捕捉和响应。**

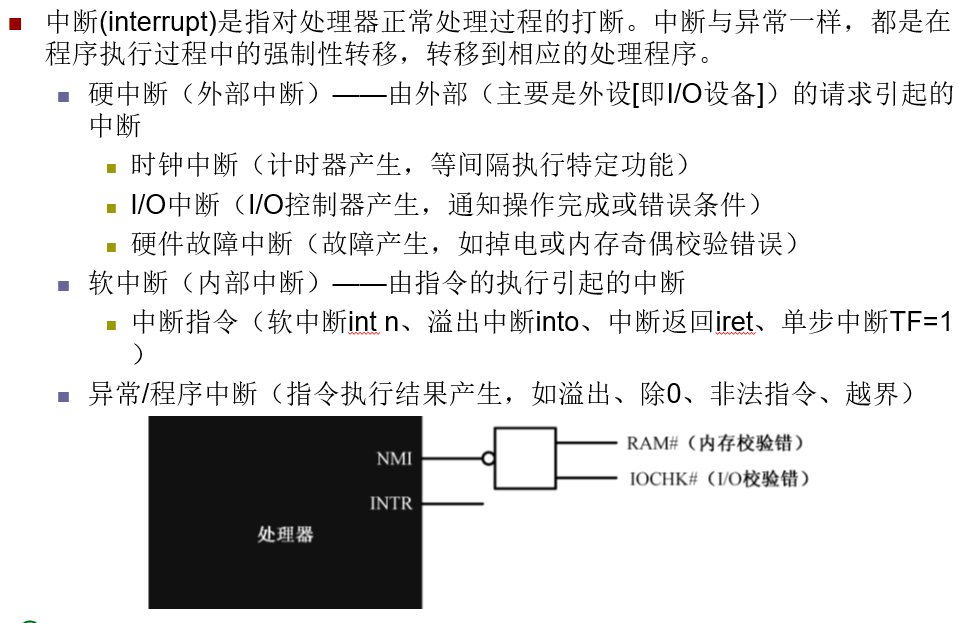
**2掌握操作系统的系统调用原理，实现原型操作系统中的系统调用框架，提供若干简单功能的系统调用。**

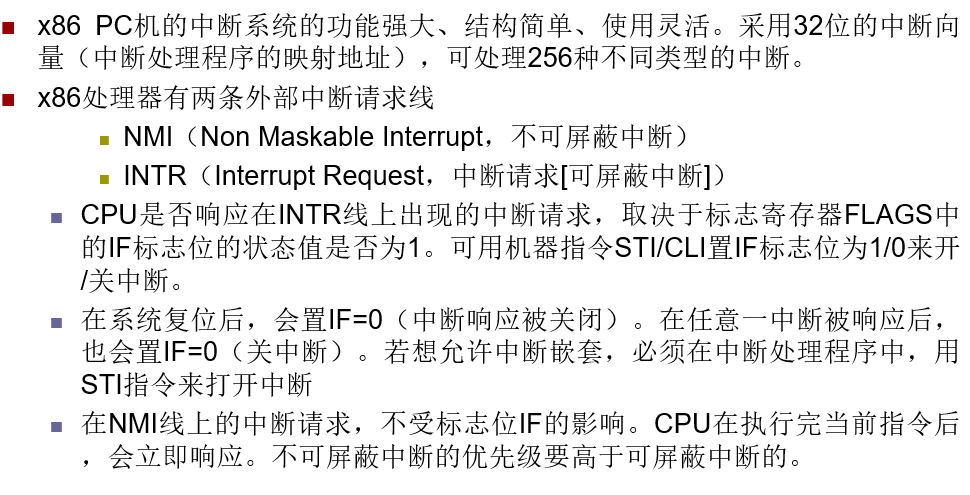
**3学习握掌c语言库的设计方法，为自己的原型操作系统配套一个c程序开发环境，实现用自建的c语言开发简单的输入/输出的用户程序，展示封装的系统调用。**

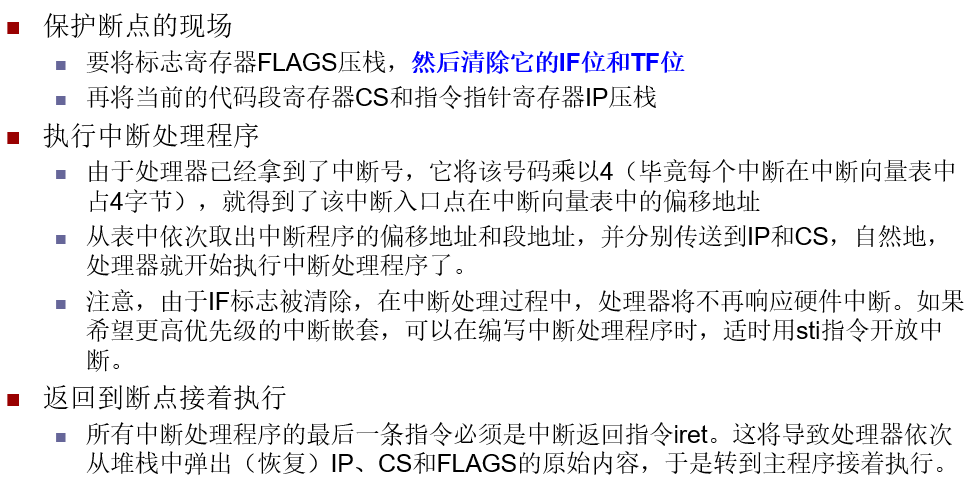
**三 实验方案**

**（1）基础原理**

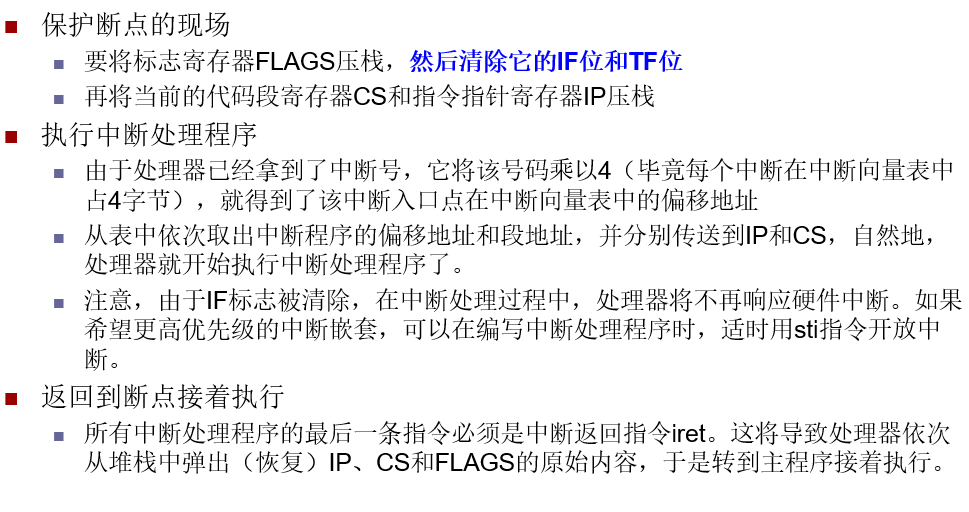
**什么是中断？**



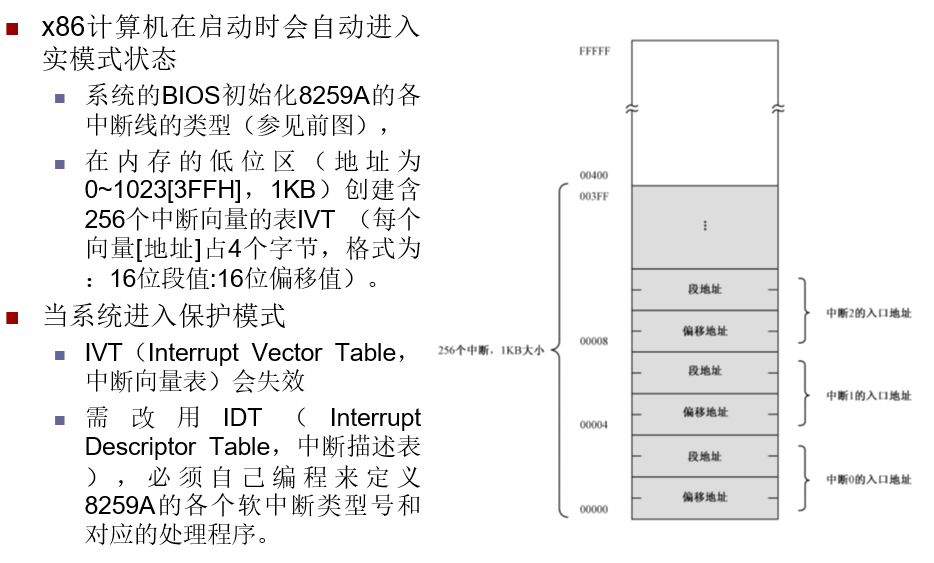




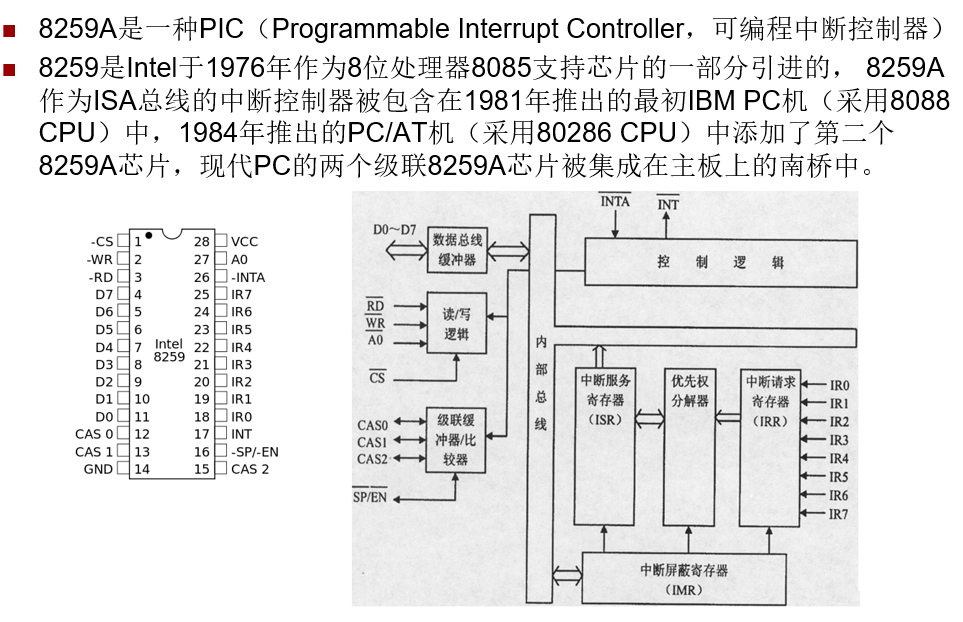
**中断的处理过程：**

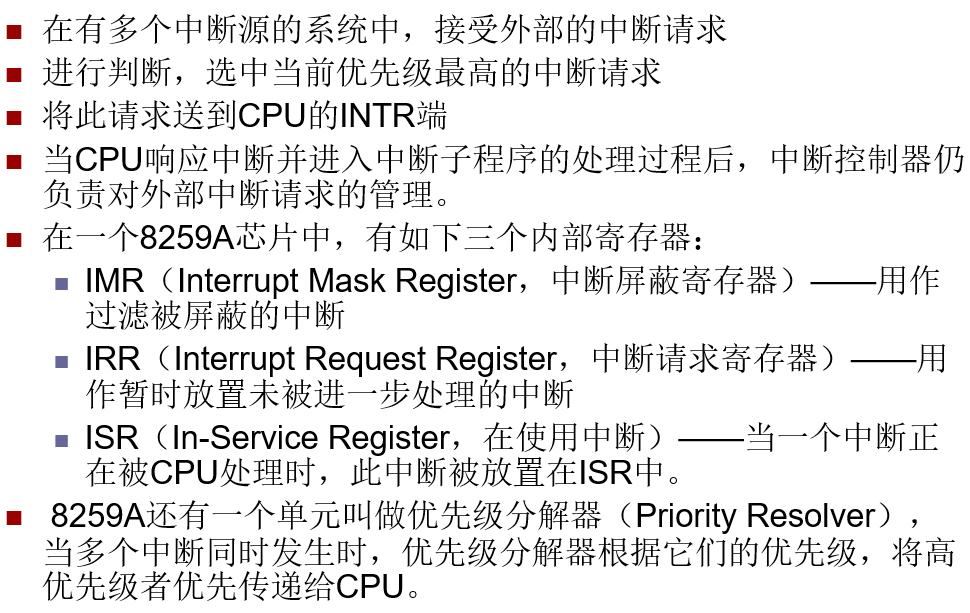


**中断向量表在内存中的位置：**

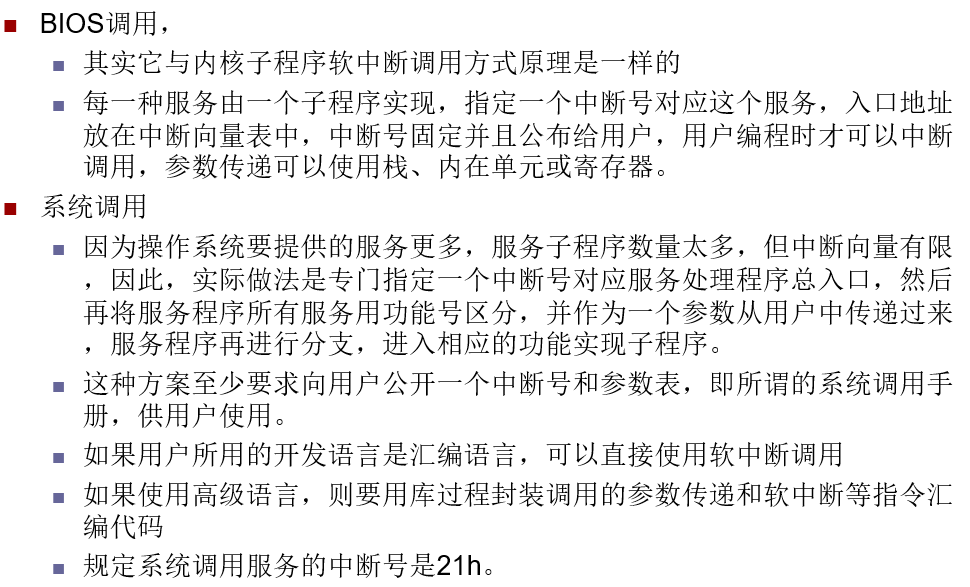


**8259A对终端的控制作用：**





**软中断实现系统调用：**



**（2）实验工具环境**

实验支撑环境

硬件：个人计算机

主机操作系统：Windows/Linux/Mac OS/其它

虚拟机软件：VMware/VirtualPC/Bochs/其它

PC虚拟机裸机/DOS虚拟机/其它

实验开发工具

汇编语言工具：x86汇编语言

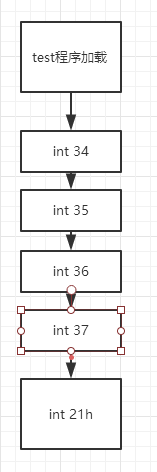
高级语言工具：标准c语言

磁盘映像文件浏览编辑工具

调试工具：Bochs

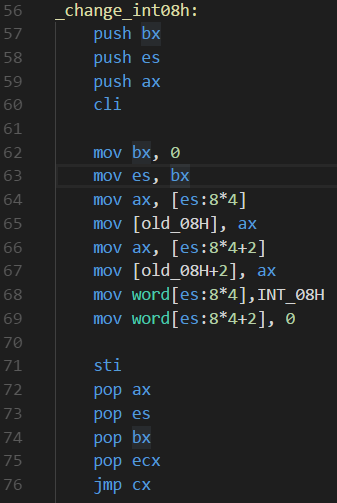
**（3）程序流程：**

测试程序流程图



(4) 代码模块组成：

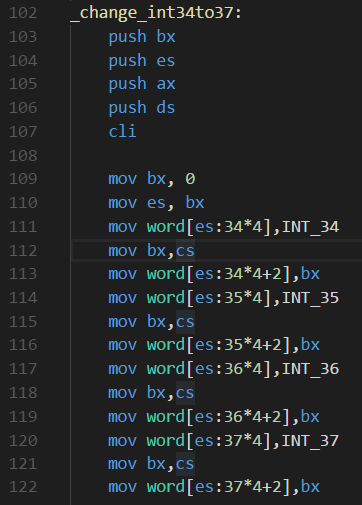
修改中断向量表模块：



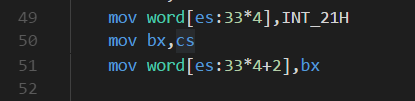
一开始写这个模块时，我的思路是按照老师的思路重修INT 08H 但随后我发现，我们重写的INT 08H会修改时钟当前的计数‘滴答’值，经过网上查阅资料我得知，int 08H 有一部分和系统时钟有关，因此如果重写INT 08H ，我们的系统时钟有可能改变。在实验三中，我利用gettime函数，通过获取当前的系统时钟实现了随机数程序，但在本次实验中，如果我毫无顾忌的直接重写INT 08H，会导致系统时钟被不断重新设置，随机数程序就会变成彻彻底底的伪随机。为了修改这个BUG，我采取了‘重载’ bios 中断的方式：将原终端地址保存到一个内存中，等运行完我的程序后，再将它调到原中断程序中执行。

修改键盘响应中断模块和上述模块类似

修改int 34—37中断模块：

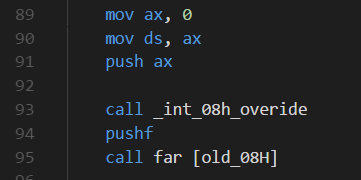


修改系统中断模块：（与上述模块相似）

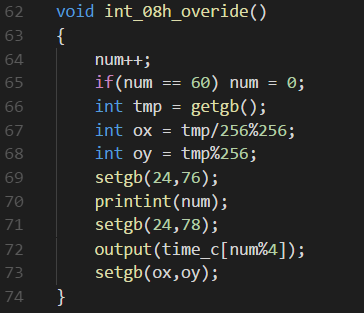


新的中断响应：

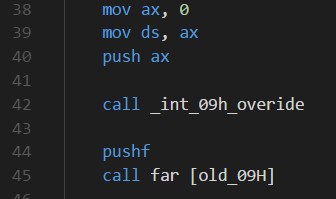
时钟中断：



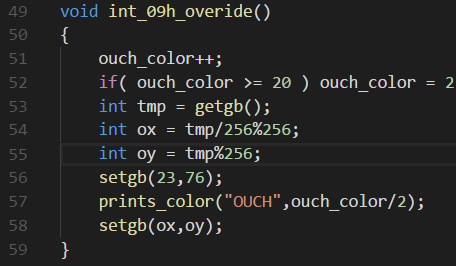
Int\_08h\_overide是我实现的c语言函数



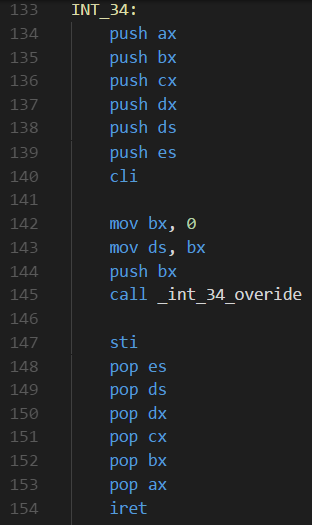
键盘响应中断：



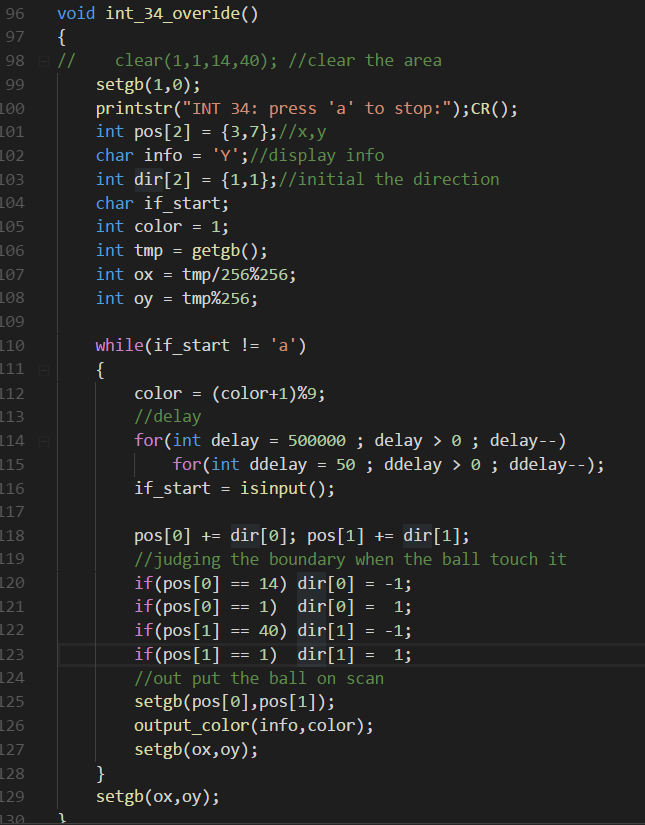
Int\_09h\_overide是我实现的c语言函数



重写INT 34,35,36,37 ：



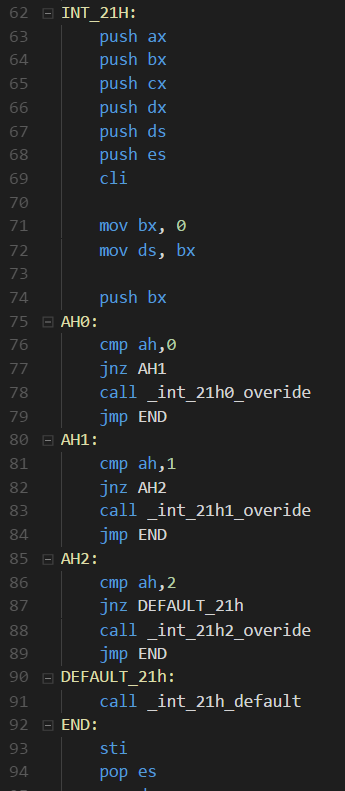
Int\_34\_overide是我写的c语言程序



（34，35，36，37中断类似）

这段c语言程序通过调用我自己实现的stdio库，实现了在屏幕四分之一区域控制字符的弹跳，33号中断控制左上方屏幕，34号中断控制右上方屏幕，35号中断控制左下方屏幕，36号中断控制右下方屏幕。分别通过输入字符‘a’，‘b’，‘c’，‘d’来结束程序的弹跳。

新的系统中断：



我们通过比较ah中的值来进行判断，将程序跳到哪一个我们实现的c语言程序中执行。其实此处有很多中方法来进行比较与跳转。我这里使用的时CMP指令，在汇编语言中不断比较再进行跳转。

这里给出我实现的INT 21H终端号参数表：

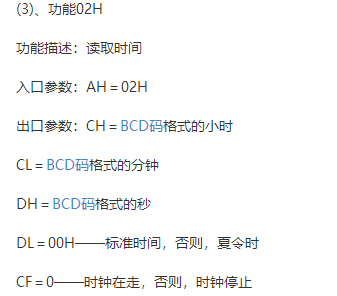
|  |  |
| --- | --- |
| 参数AH | 功能 |
| 0 | 获取并打印当前时间（年月日小时分钟） |
| 1 | 输入字符串并打印之 |
| 2 | 在频幕上输出“I love OS” |
| 其他 | 在屏幕上提示系统调用错误信息 |

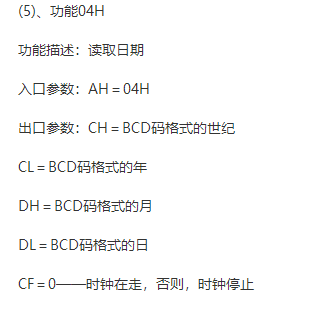
系统功能调用中的c语言函数：



该段程序通过我实现在stdio中的函数来进行输出时间的操作

这里先介绍一下INT 1aH：

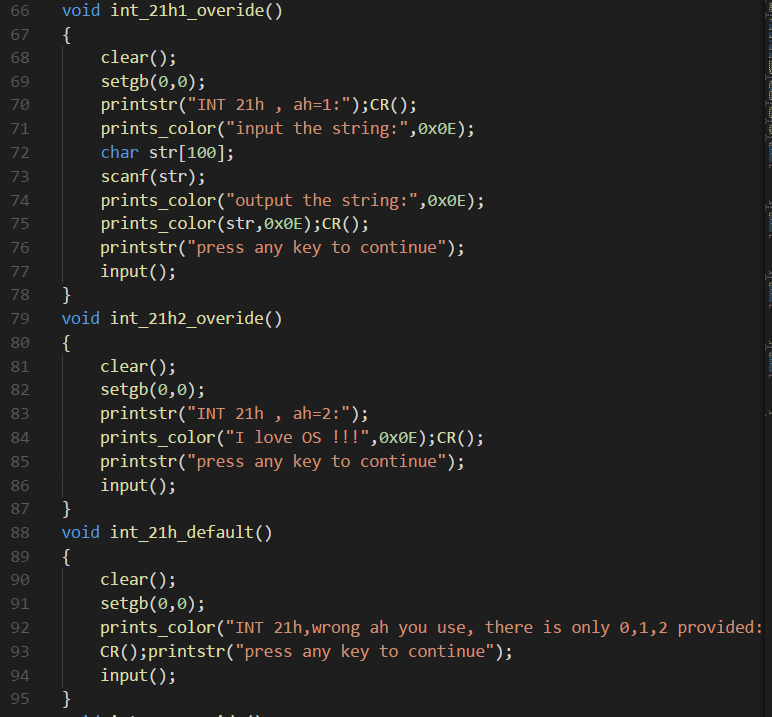




接下来我将BCD码转为十进制的操作：

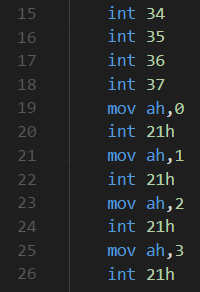
C:\Users\DELL\AppData\Roaming\Tencent\Users\1160537814\QQ\WinTemp\RichOle\)1I)U~SG)0ZBQR(XX3SLSNG.png

该操作确保了将一个int类型的BCD码转化为十进制，并储存在int中



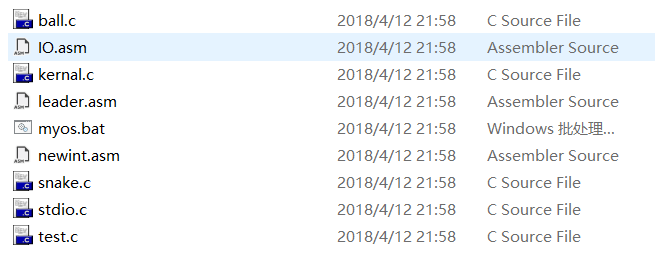
这三个函数分别对应着ah = 0，1，2的情况

中断响应测试程序：



该模块思路比较简单，明了，保护完段寄存器之后，直接测试即可

**（5）代码文档组成**



Kernal.c c语言内核

Stdio.c 实现的c语言库

IO.asm 汇编语言中的IO接口

Leader.asm 引导程序

Snake.c 贪吃蛇游戏

Ball.c 小球弹跳程序

Test.c 测试中断程序

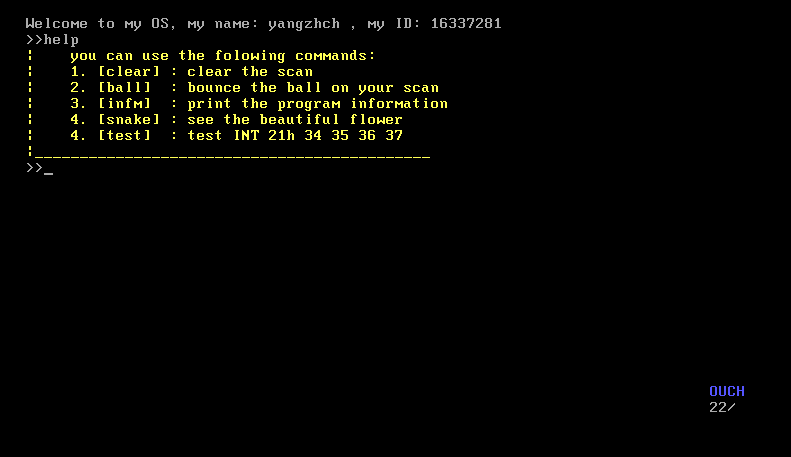
Newint.asm 中断重写的asm文件

Myos.bat 批处理文件

（**6）实现效果：**

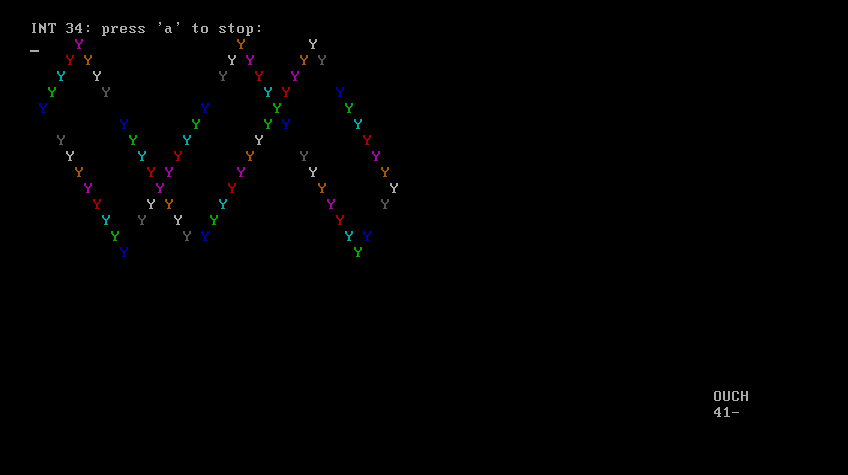
此为刚进入虚拟机时，可以看见右下角的‘摩天轮’，计数器，以及键盘输出响应的ouch

在本次实验中，我的键盘输出响应变化是变色，即每按一次键盘变依次ouch颜色



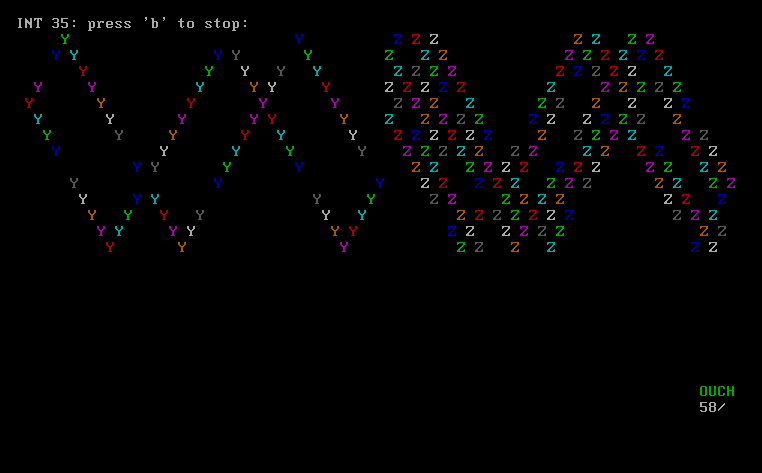
运行test程序效果：此为

INT 34



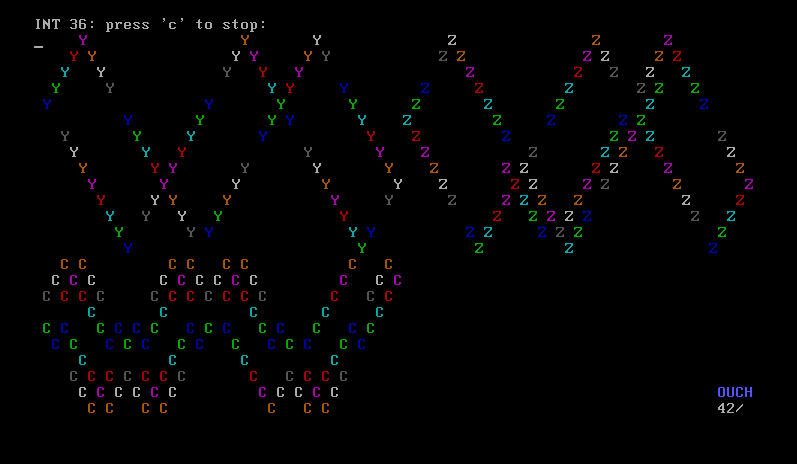
按下a

INT 35：



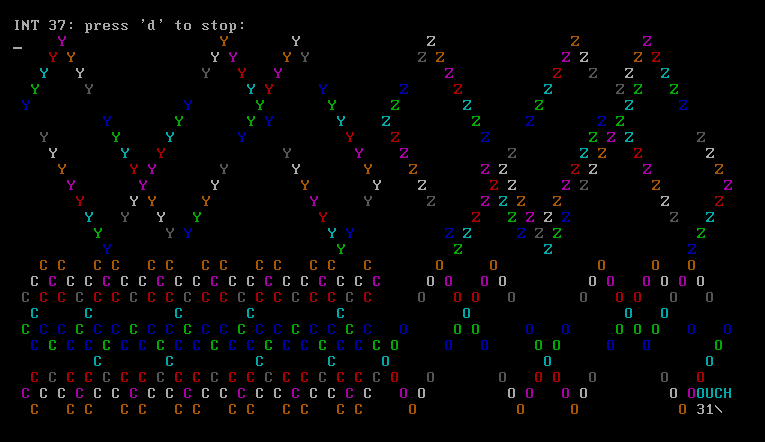
按下b

INT 36：



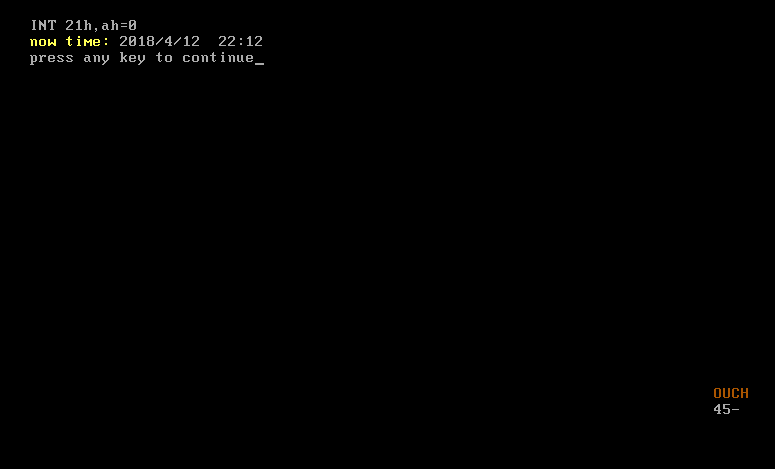
按下c

INT 37



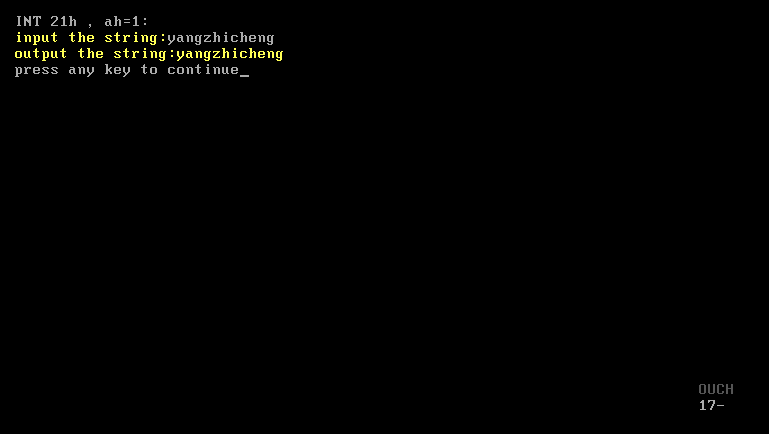
按下d

INT 21h（ah = 0）

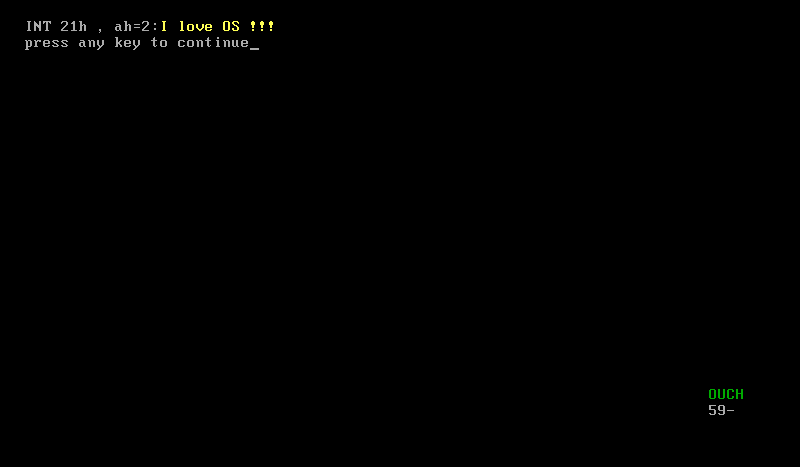
****

**如图可见时间显示是正确的**

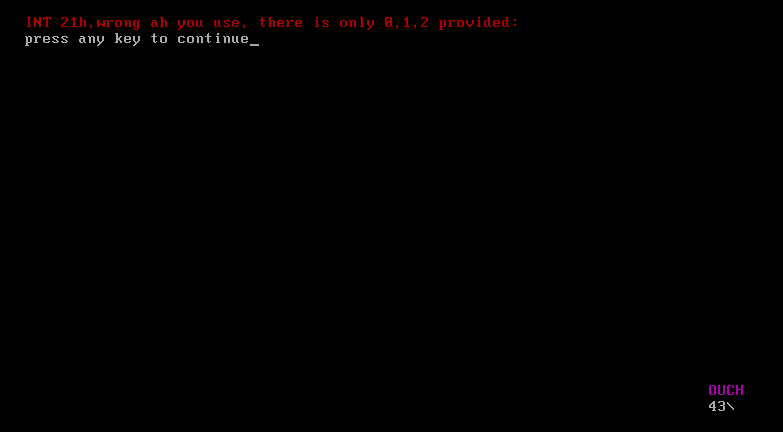
INT 21H(ah = 1)：



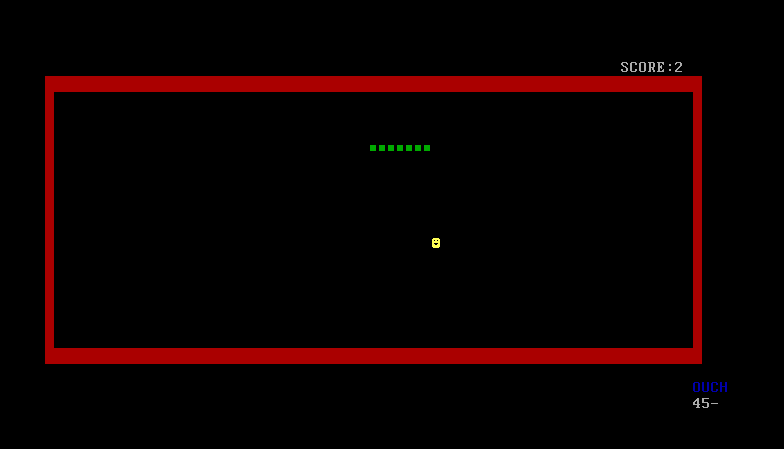
INT 21H(ah = 1)



此外，在我设计的系统调用终端中，若错误使用端口号，我会进行提醒：



此外也可以看见我的时钟和键盘中断在用户程序里也能运行：



**五 实验过程及其总结**

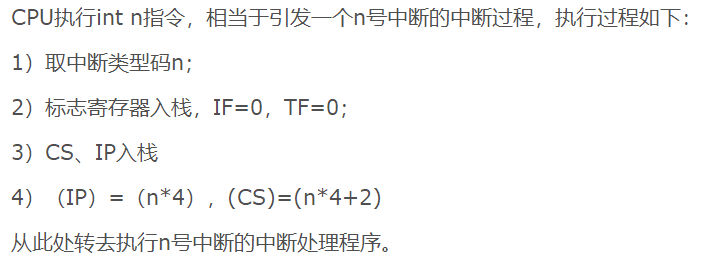
在本次实验中，虽然更改终端的代码量并不大，但是debug却不一定是一件容易的事情，因为BIOS终端对我们而言更多的像是一个黑箱，我们不知道黑箱里究竟发生了什么。

接下来列举一下我在本次实验中遇到的问题，以及我自己的思考：

**（1）为什么跳入原中断底之前，要使用pushf ?**

C:\Users\DELL\AppData\Roaming\Tencent\Users\1160537814\QQ\WinTemp\RichOle\TPBC9~74V])3E}{LUBQHD@4.png

这里就涉及到了int指令与iret指令



而iret指令相当于此指令的逆过程，由于在此处我们是call到了[old\_09h]的旧中断地址，而call指令的执行过程中，并没有pushf的操作，因此为了中断程序能够正常返回，我们需要加上一条pushf指令，将标志寄存器压栈。

**（2）如何进行保护断点的操作？**

这个问题并没有困扰我很久，我们只需要将段寄存器ds，es，以及某些通用寄存器压栈即可，当然也不排除某些特殊的情况，导致我们需要将bp，sp等偏移寄存器压栈。

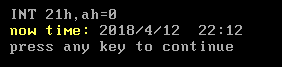
**（3）如何获取时间？**

个人认为在本次实验中，获取时间这个操作是我的一个小创新点，我通过调用BIOS的int 1Ah准确获取了系统时间，在这个过程中我运用了BCD转十进制的知识，技术细节如下：

C:\Users\DELL\AppData\Roaming\Tencent\Users\1160537814\QQ\WinTemp\RichOle\)1I)U~SG)0ZBQR(XX3SLSNG.png

因为BCD每4位代表一个十进制位，因此通过将tmp右移四位来实现然后与0x000F进行与运算来获得正确的十进制数。

然后后将其打印在屏幕上。



其实，**在本次实验中，我可以将时钟中断（INT 08H）修改一下，让他不断的在屏幕右下角刷新显示当前的系统时钟，**这是轻而易举的事情，但是老师本次的实验并没有这个要求，我就把这个功能放到了系统调用中实现，**若老师认为这个策略可行，我会在下次的实验中把该功能放到时钟中断上。**

**（4）一些低级错误**

在此次中断编写过程中，我曾犯下了一个低级错误：把INT 33和INT 33h搞混淆了，虽然我修改对了中断向量表，但是响应中断时，用的时INT 33H，将十六进制和十进制搞混淆是一个十分低级的错误，下次一定不会再犯。

（**5）时钟中断和键盘响应中断修改后，是否需要还原？**

在我的这个操作系统里，是不需要还原的，我在微信群看到老师和同学们讨论时，认为跳入到用户程序前，应该将中断向量表还原，我觉得没有这个必要，让ouch和时钟计数相应在特定的位置即可。而且我的时钟中断和键盘响应中断进入用户程序之后依然正常运行，没有任何异处。

**（6）系统中断和34到37中断写在哪个文件？**

本次实验中我写在了一个新的文件，要测试系统终端时，直接让内核加载这个程序即可。但是我的08和09中断（时钟中断和键盘响应中断）直接写在了c语言函数库中，这样，这两个终端就可以从头到尾一直存在；且可在用户程序中一直正常运行。

**六 实验感想**

本次实验不难，但还是花费了我的一些时间。

中断机制是我们在学写操作系统中必不可少的一个知识点。他是我们进行多进程操作（而状态进程模型）的基础。也是我们想要掌握操作系统所必经的路。

这么多次操作系统实验以来，我总结了不少经验，以后的操作系统实验想

必依然不会顺畅，但我仍会保持我的好奇心，迎难而上。

争取要在这个学期，写出一个令老师也令我自己满意的操作系统。