操作系统实验报告

实验四&五：中断技术编程的设计&&简单的系统调用

学院：　　数据科学与计算机学院

班级：　　16级计算机科学与技术 教务2班

姓名：　　郑映雪

学号：　　16337327

完成时间：　　2018.4.11

一、实验目的

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已有的用户程序时，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.学习掌握 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。

二、实验要求

1、操作系统工作期间，利用时钟中断，在屏幕最边缘处动态画框，第一次用字母 A，第二次画用字母 B，如此类推，还可加上变色闪耀等效果。适当控制显示速度，以方便观察效果。

2、编写键盘中断响应程序，原有的你设计的用户程序运行时，键盘事件会做出有事反应：当键盘有按键时，屏幕适当位置显示”OUCH! OUCH!”。

3、在内核中，对 34 号、35 号、36 号和 37 号中断编写中断服务程序，分别在屏幕 1/4 区域内显示一些个性化信息。再编写一个汇编语言的程序，作为用户程序，利用 int 34、int 35、int 36 和 int 37 产生中断调用你这 4 个服务程序。

4、扩充系统调用，实现三项以上新的功能，并编写一个测试所有系统调用功能的用户程序。

三、实验方案

1、相关原理

a.硬中断与软中断

当设备有某种事情发生时，它就会产生中断，并通过总线把电信号发送给中断控制器，中断控制器把电信号发送给处理器的某个特定引脚，处理器进行中断处理。在本实验中要实现两种中断——硬中断与软中断。

硬中断是由外设产生的，可以通知操作系统外设状态的变化。在这个实验中我们要实现的硬中断是键盘中断（即实验要求中的按下键盘就会在用户程序中显示“OUCH”）。软中断是服务程序对内核的中断，并不会直接中断 CPU。

b.时钟中断和键盘中断

时钟中断和键盘中断发生时，调用 08h 号中断，注意需要发送 EOI 到主 8529A 和从 8529A（这一步非常重要）。键盘中断发生时，调用 09h 号中断，但需要注意的是，键盘中断并不是按下键中断一次，而是按下键和释放键都会引发中断，也就是说在按键过程中会引发两次中断。

c.安装中断

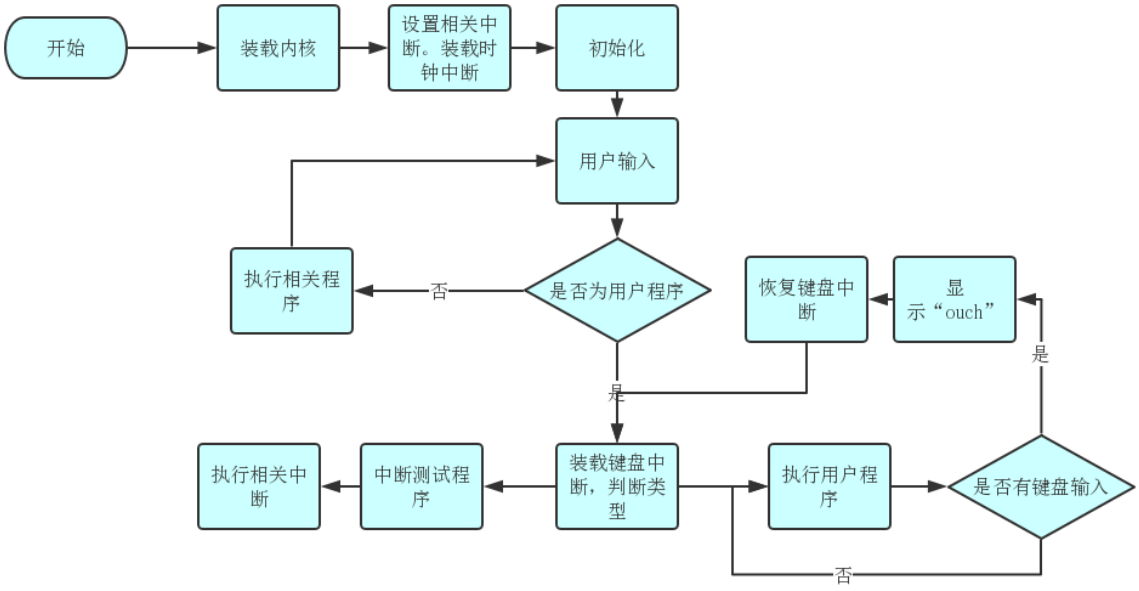
中断是在 0000h 的前 256 个字节中安装。Intel 预留了前二十个中断，当我们改写它们时要注意把原中断保存在栈中，结束后要恢复原中断。软中断可以在 21h 之后设计，同样也是安装在 4*中断序号 的地址。在本实验中，要求第三项的 34、35、36、37 号中断与作为系统调用的 33 号中断都是软中断，通过安装中断的函数安装在对应地址中。比如，我要安装 33 号中断，则安装的入口地址为：cs：33*4+2，ip：33*4。安装这些中断是一个重复的过程，所以可以将中断号作为参数压栈，利用一个安装向量的函数统一安装。

d.系统调用

系统调用提供了用户程序与操作系统内核之间的接口，系统调用是用户不可见的，由内核来完成相关的操作。在本实验中设计了内核子程序软中断调用，即采用软中断的方式，用 int 指令调用操作系统服务的子程序，类似于我们之前学习的设置 ah 功能号再调用中断。做法是指定一个中断号对应服务处理程序总入口，再将功能号相区分，作为参数从用户程序中传递过来，通过判断参数来执行

哪一项功能。这个原理是我们之前大量使用 BIOS 中断时所熟悉的了。

2、程序流程图



3、程序关键模块

本次实验报告中我只放上相对于上一个实验新增的代码，即几个中断的实现代码。

a. 安装中断向量的函数

通过资料得知，中断向量的安装方法相同，不同的只是根据序号而不同的存放地址，所以可以写一个函数，将向量地址和标号传参，就可以不用每次写中断都单独设置了。但是要在 os 中调用设置中断向量的函数。关键部分如下：

```
.....
mov ax, 0
mov es, ax    ;将段地址设置为 0000h
mov al, 4
mov bl, [bp+10]
mul bl     ;中断序号*4 的处理
mov di, ax
mov ax, [bp+8]
mov word ptr es:[di], ax ; 设置中断向量的偏移地址
add di, 2
mov word ptr es:[di], cs;  设置中断向量的段地址
mov sp, bp
.....
```

b. 时钟中断

这个中断来源于老师给的时钟中断示例，将显示的模块进行修改，利用分支在右下角实现“LOVE”的轮次闪动。（老师的原要求为显示边框，但后期大家讨论时老师同意改用显示其它内容，只要实现时钟中断即可）

```
……
mov ax, 8   ;安装 8 号中断
push ax
mov ax, offset MyClock
push ax
call _setNewInt   ;调用安装中断的函数安装中断
……

mov ax,0B800h                    ; 文本窗口显存起始地址
mov es,ax                        ; GS = B800h
mov ax, 1
mov di, (24*80+79)*2          ;右下角
cmp ax, word ptr[chx]
jz show1
mov ax, 2
cmp ax, word ptr[chx]
jz show2
mov ax, 3
cmp ax, word ptr[chx]
jz show3
mov ax, 4
cmp ax, word ptr[chx]
jz show4
show1:                           ;轮流显示“LOVE”
mov ah,0FAh
mov al, 'L'
mov word ptr es:[di], ax
mov word ptr [chx], 2
ret
……此处省略另外 3 个字母显示的代码展示
endd@:                           ; 从中断返回
pop es
pop ax
mov al,20h                       ; AL = EOI
out 20h,al                            ; 发送 EOI 到主 8529A
out 0A0h,al                      ; 发送 EOI 到从 8529A
iret ;中断返回
```

c. 键盘中断

首先在读取用户程序之前，将 9 号中断保存起来，然后设置键盘中断向量的偏移地址，在显示 ouch 之后要记得在原处填充空格以消除，另外要清除键盘数据锁存器以能多次调用该中断。最后再调取用户程序之后，要恢复 9 号中断。这样就可以实现进入用户程序则会按下键盘和释放键盘响应 ouch。具体代码如下：

```
        ……
        xor ax,ax
        mov es,ax
        push word ptr es:[9*4]                          ;将 9h 中断入栈，再出栈保存在数据中
        pop word ptr ds:[0]
        push word ptr es:[9*4+2]
        pop word ptr ds:[2]
        mov word ptr es:[24h],offset keyboard       ; 设置键盘中断向量的偏移地址
        mov ax,cs
        mov word ptr es:[26h],ax
        ……
        xor ax,ax
        mov es,ax
        push word ptr ds:[0]      ;将存有原中断的数据入栈再出栈到出栈保存到原中断地址
        pop word ptr es:[9*4]
        push word ptr ds:[2]
        pop word ptr es:[9*4+2]
        int 9h
        ……
keyboard:
        ……
        mov ax,0B800h
        mov es,ax
        mov ah, 0Fh
        mov al, 'O'
        mov word ptr es:[((12*80+40)*2)],ax
        mov al, 'U'
        mov word ptr es:[((12*80+41)*2)],ax
        mov al, 'C'
        mov word ptr es:[((12*80+42)*2)],ax
        mov al, 'H'
        mov word ptr es:[((12*80+43)*2)],ax
        mov al, '!'
        mov word ptr es:[((12*80+44)*2)],ax

        ; 延迟微小的时间再让 ouch 消失
        mov cx,delayTime
```

```
（接上文）
    ……此处省略延时代码
    mov al,' ';用空格填充原本的地方
    mov word ptr es:[((12*80+40)*2)],ax
    mov al,' '
    mov word ptr es:[((12*80+41)*2)],ax
    mov al,' '
    mov word ptr es:[((12*80+42)*2)],ax
    mov al,' '
    mov word ptr es:[((12*80+43)*2)],ax
    mov al,' '
    mov word ptr es:[((12*80+44)*2)],ax

    pop es
    pop ax

    in al,60h  ;清除键盘数据锁存器，以便下次还可以触发
    mov al,20h                          ; AL = EOI
    out 20h,al                          ; 发送 EOI 到主 8529A
    out 0A0h,al                         ; 发送 EOI 到从 8529A
    ……
```

d. 四个软中断

按照实验要求，设计了 34、35、36、37 四个中断，分别在屏幕的四分之一处显示一些内容，并用用户程序调用。这些软中断只要实现了一个，另外三个只要代码重用就可以了，比较简单。

首先用户测试程序就非常简单，代码如下：

```
    org 7e00h
    int 34
    int 35
    int 36
    int 37
```

上面这个用户程序，分别调用了这 4 个软中断，由于这 4 个软中断非常相似，所以我只放上其中一个中断的代码：

```
        mov ax, 34
        push ax
        mov ax, offset int34
        push ax
        call _setNewInt    ;安装 34 号向量
        pop ax
        pop ax
        pop es
        pop ax
        ret

int34:
        push bp
        push ax
        push bx
        push cx
        push dx
        push es
        push ds
        mov ax, 0b800h                          ;显示'INT34'
        mov es,ax
        mov di, (80*1+0)*2
        mov ah, 0Fh
        mov al, 'I'
        mov word ptr es:[di], ax
        mov al, 'N'
        mov word ptr es:[di+2], ax
        mov al, 'T'
        mov word ptr es:[di+4], ax
        mov al, '3'
        mov word ptr es:[di+6], ax
        mov al, '4'
        mov word ptr es:[di+8], ax

        mov ax,cs
        mov ds,ax
        mov es,ax
        mov ss,ax
        mov ah,13h
        mov al,1
        mov bl,0ah
        mov bh,0
        mov dh,6
        mov dl, 0
```

```
    mov bp,offset str1
    mov cx,23
    int 10h ; 调用 10H 号中断显示字符
    pop ds
    pop es
    pop dx
    pop cx
    pop bx
    pop ax
    pop bp
    iret
str1 db "The experiment is hard!"
setint34 endp
```

36、37、38 号中断同理，就不重复贴上了。

e. 系统调用中断

我将 33 号中断作为系统调用中断，第一个功能只是一个简单的显示 hello，第二个功能是之前实现了的显示时间，第三个功能我又加上了显示日期。老师将两个实验在一起布置应该是系统中断的做法也只是之前的软中断多了个 ah 里的功能号的跳转而已，因此原理上是相同的。代码如下：

```
    ……
    mov ax, 33
    push ax
    mov ax, offset int33
    push ax
    call _setNewInt
    pop ax
    pop ax
    pop ax
    pop es
    pop ax
    ret

int33:
    push bp
    push bx
    push cx
    push dx
    push es
    push ds
```

```
    cmp ah, 0                   ;根据功能号实现对应的功能
    je int330
    cmp ah, 1
    je int331
    cmp ah, 2
    je int332
int330:                 ;0 号功能：在屏幕中间显示"HELLO"
    push es
    push ax
    mov ax, 0b800h
    mov es,ax
    mov di, (80*13+38)*2
    mov ah, 0ah
    mov al, 'H'
    mov word ptr es:[di], ax
    mov ah, 0Bh
    mov al, 'E'
    mov word ptr es:[di+2], ax
    mov ah, 0Ch
    mov al, 'L'
    mov word ptr es:[di+4], ax
    mov ah, 0Dh
    mov al, 'L'
    mov word ptr es:[di+6], ax
    mov ah, 0eh
    mov al, 'O'
    mov word ptr es:[di+8], ax
    pop ax
    pop es
    jmp int33end

int331:                 ;1 号功能，在屏幕下方显示当前时间
    mov bh,0            ;读光标位置
    mov ah,3
    int 10h
    push dx             ;保存光标位置
    mov dh, 24
    mov dl, 38          ;设置光标位置
    mov bh,0
    mov ah,2
    int 10h
    call _showTime      ;调用 C 文件中的函数，在屏幕下方显示当前的时间
    pop dx
    mov bh,0            ;恢复光标位置(dh,dl)
```

```
    mov ah,2
    int 10h
    jmp int33end

int332:         ;2 号功能，在屏幕偏上方显示时间
    mov bh,0            ;读光标位置，(dh,dl) = (行，列)
    mov ah,3
    int 10h
    push dx             ;保存光标位置
    mov dh, 12
    mov dl, 38          ;设置光标位置(dh,dl) = (12， 38)
    mov bh,0
    mov ah,2
    int 10h
    call _showDate      ;调用 C 文件中的函数，在屏幕下方显示当前的时间
    pop dx
    mov bh,0            ;恢复光标位置(dh,dl)
    mov ah,2
    int 10h
    jmp int33end

int33end:
    pop ds
    pop es
    pop dx
    pop cx
    pop bx
    pop bp
    iret
    ……
```

f. 系统调用中断的用户程序

这个程序由 C 语言和汇编语言链接而成，用的相关输入输出的函数都是内核相关的函数，故在这里不贴上库的代码了。以下是 C 语言部分代码：

```
extern int strlen(char *str);
extern void printchar(char ch);
extern void showhello();
extern void showtimeint();
extern void showdateint();
char a[100];
int n;
int flag;
```

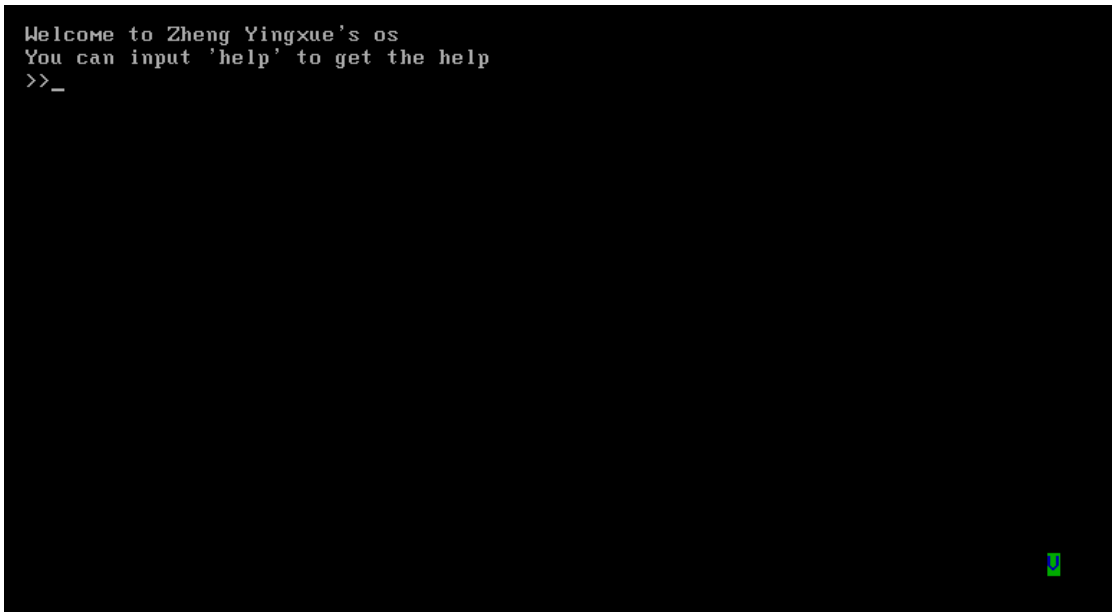
```
void printf(char *ch){
	while (*ch != '\0'){
		printchar(*ch);
		ch++;
	}
}
int i =0;
int strcmp(char str1[], char str2[], int len1, int len2) /*实现两个字符串的比较*/
{
	if(len1 < len2) return 0;
	for(i = 0; i < len2; i++)
		if(str1[i] != str2[i]) return 0;
	return 1;
}
void main()
{
	while(1){
		printf("\r\n");
		printf("Input 0/1/2 to get the system call\r\n");
		printf("0: show 'hello' \r\n1: show the current time\r\n2: show the current
date\r\n");
		printf(">> ");		n=0;
		n = strlen(a);
		if (strcmp(a,"0",n,1)) flag=0;
		else if (strcmp(a,"1",n,1)) flag=1;
		else if (strcmp(a,"2",n,1)) flag=2;
		else continue;
		switch (flag)
		{
			case 0:
				showhello();
				printf("\r\n");
				break;
			case 1:
				showtimeint();
				printf("\r\n");
				break;
			case 2:
				showdateint();
				printf("\r\n");
				break;
		}
	}
}
```

关于系统调用中断部分的代码如下，都是简单的 ah 设置功能号，调用 33 号中断。

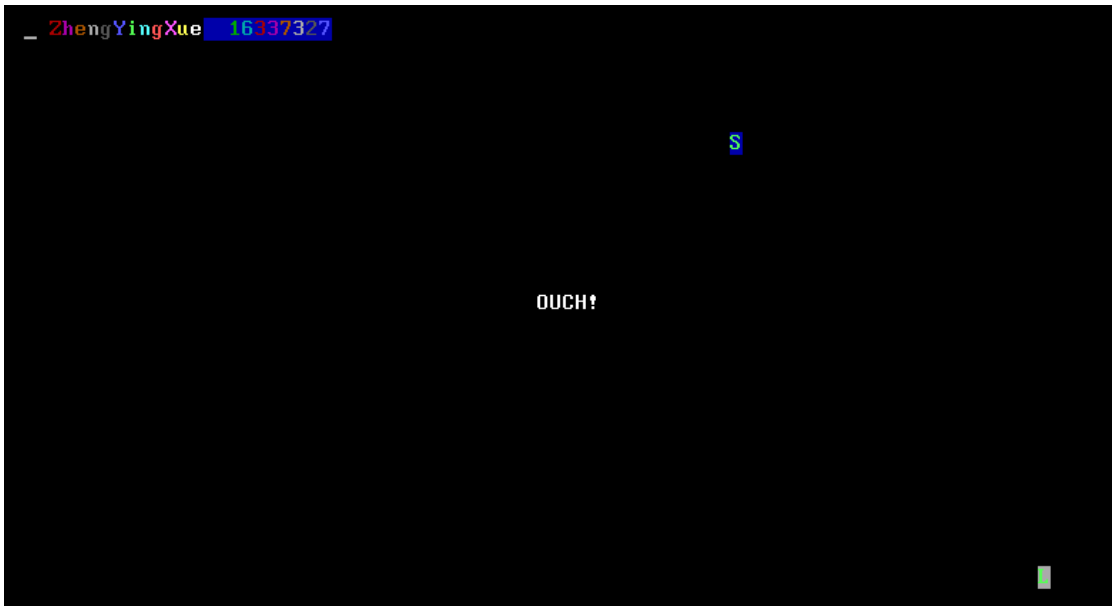
```
public _showhello
_showhello proc
	push ax
	mov ah, 0
	int 33
	pop ax
	ret
_showhello endp

public _showtimeint
_showtimeint proc
	push ax
	mov ah, 1
	int 33
	pop ax
	ret
_showtimeint endp

public _showdateint
_showdateint proc
	push ax
	mov ah, 2
	int 33
	pop ax
	ret
_showdateint endp
```

四、实验过程

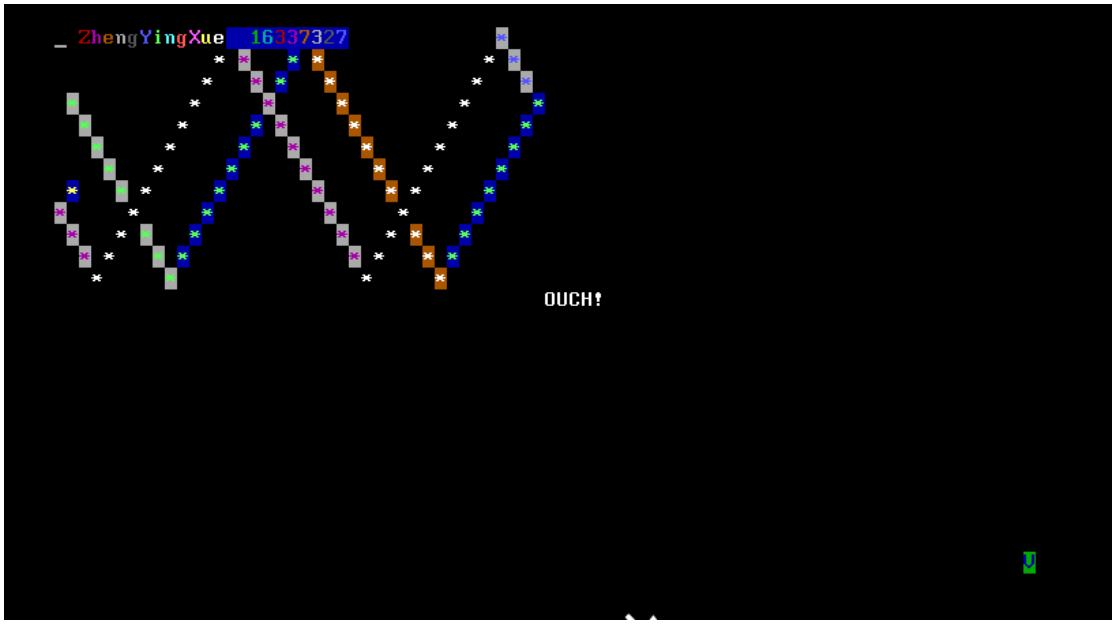
1、进入内核界面，可以看到右下角有轮回闪烁的 LOVE 字母并更换颜色。由于截图比较困难，所以这里只截图一张：



2、输入 run a 进入第一个用户程序，即在第一象限内飞翔的 S 字符，此时按下键盘会有 ouch 出现。（因为闪动的 ouch 太难截图，特地用了截图的延迟功能，真好用嘻嘻）但是由于键盘中断，所以无法调取原本的按下 q 返回的代码，询问过老师，老师说无碍，可是这样就得重新开虚拟机运行之后的程序了。我已经查资料得知键盘中断后可以将按键扫描码发送至端口，但由于时间问题没有实现，得空时我会将这个功能完善一下。



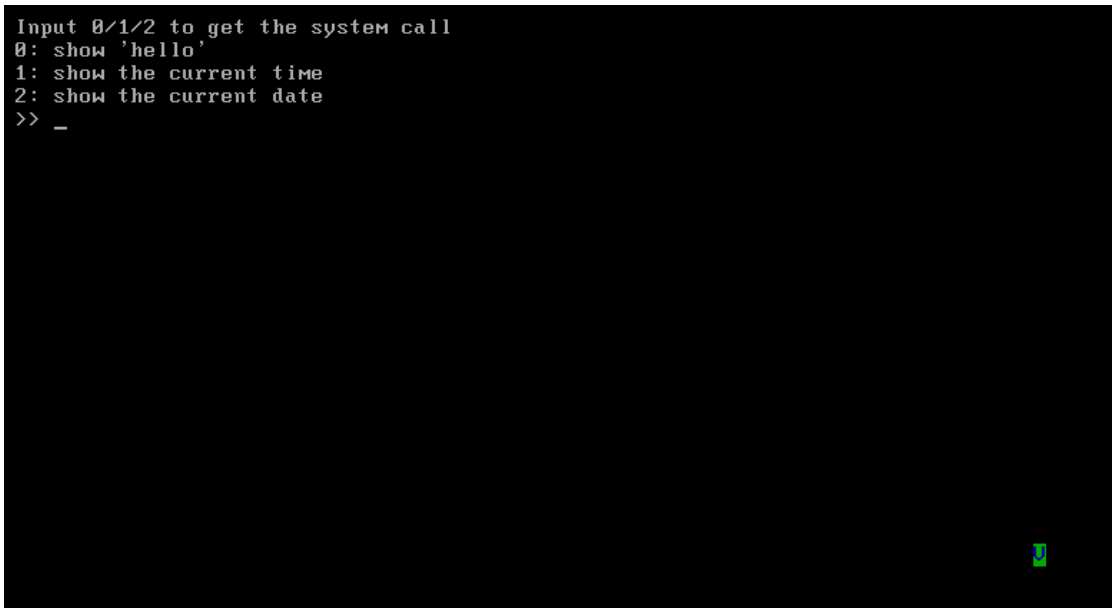
3、同理，关机再进入，输入 run b 我们可以看到用户程序 B，并且有硬件中断响应。



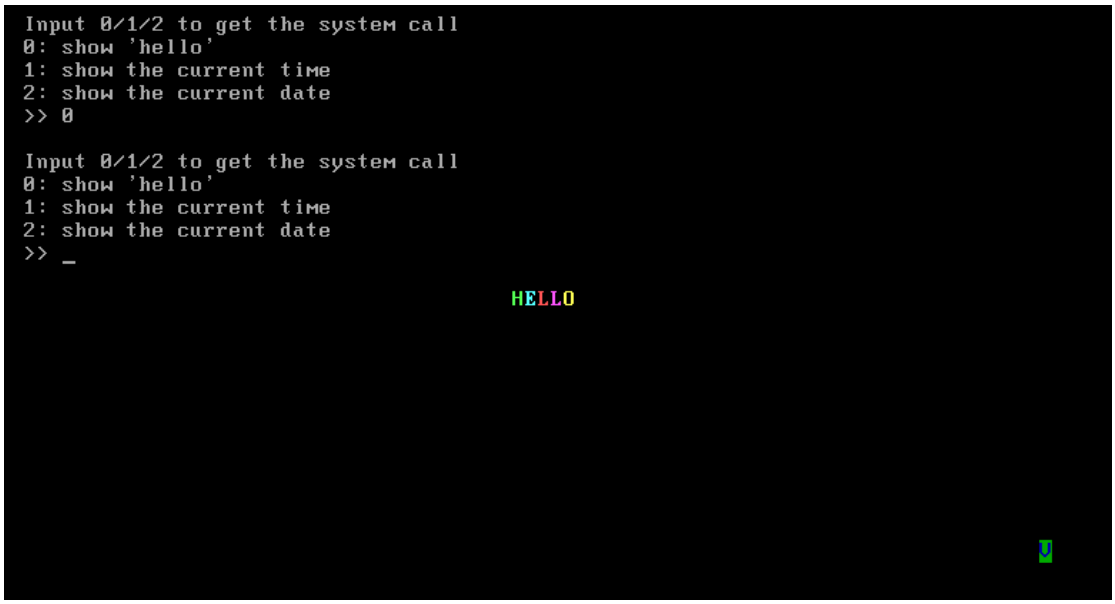
4、同理，关机再进入，输入 run c，第三象限和第四象限的用户程序因为之前运行过，所以我就没有采用，我将第三个用户程序设置为老师 PPT 中的要求，调用 34、35、36、37 中断在各四分之一屏幕显示字符：

5、关机再进入虚拟机，用户程序 D 我设置的是系统调用的程序，输入 0/1/2 可以看到三个不同功能的 33 号中断。具体操作如下：

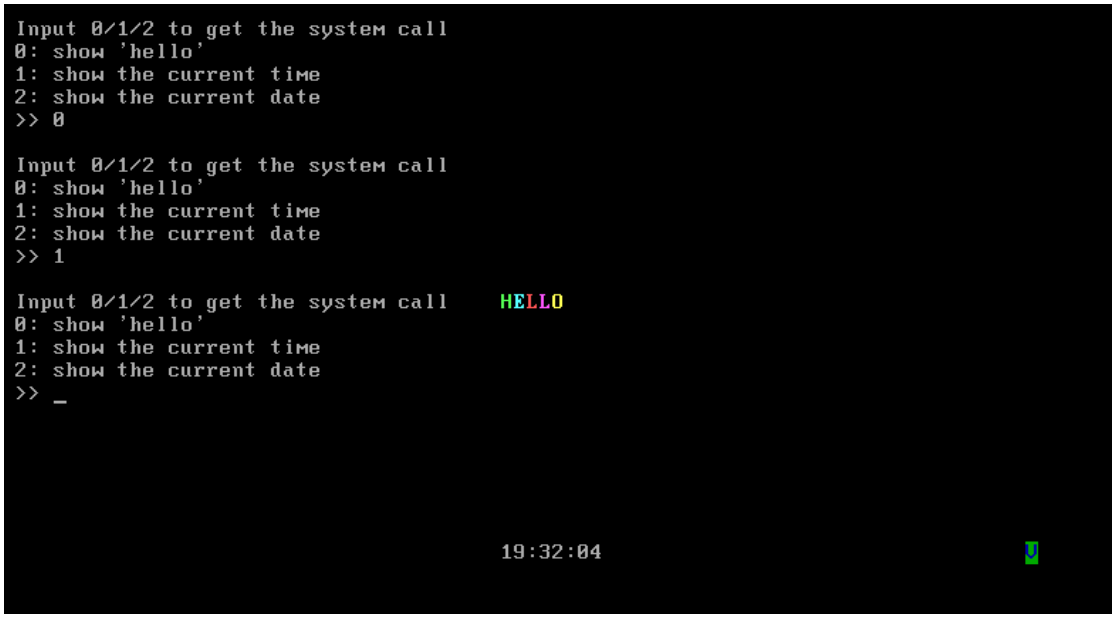
输入 run d，进入显示系统调用的界面；



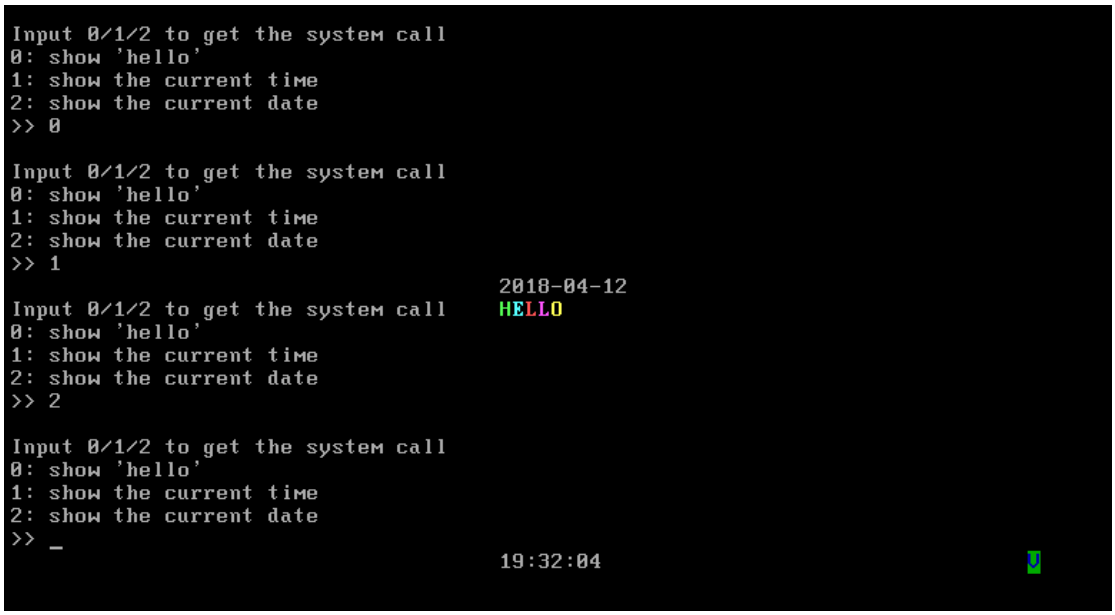
根据提示，输入 0，可以看到屏幕中央显示的“HELLO”。



再根据提示，输入 1，可以看到屏幕下方显示的当前时间：



再根据提示，输入 2，可以看到在 HELLO 的上方显示了当前的日期：



五、实验总结

这个实验又是一次花了我大量时间的实验。

首先是时钟中断的问题，本次实验中的时钟中断有老师给的范例可以做为套用，但是我将显示的内容扩充之后，发现只显示了前面三个字母。后来我发现是

延时计数的变量出了问题，修改延时计数变量后可以轮流显示 4 个字母了。这个问题解决得还是蛮快的。

后来再写硬盘中断的时候，又出现了问题。硬盘中断老师没有给范例代码，只得查资料再慢慢摸索……一开始是一进入用户程序就显示了 ouch，而且不会消失。查阅资料得知，键盘中断的触发不仅仅是按下键时，释放键时也会触发。这就解释了为什么一进入用户程序就会显示 ouch 了。但是迟迟不消失的原因在于键盘数据锁存器没有清除，导致中断只能执行一次。加上了 in al,60h 这条指令就可以多次执行中断了。

后来出现了一个概念上的错误……我没有正确理解老师的意思，把键盘中断写进了用户程序里，后来被老师指正，明白这次的所有中断都是在内核中实现，用户程序无法发现中断的实现方法，只可以调用，不管运行什么用户程序，按下键盘都会 ouch。将键盘中断写在内核里供用户程序调用也不难，就是把写好的键盘中断改写进原本的 run 函数（读取用户程序）里，就可以实现“一进入用户程序，按下键盘就 ouch”的效果了。

随后紧接而来的问题都与栈有关。因为随着实验难度的增加，有大量的问题需要用到栈，随着一大堆 push 和 pop 的到来，一个不小心就会把数据弄错，之前几个实验“用眼 debug”的我终于用到了 boches，在几次遇到错误时可以看看目前栈的情况，很容易发现哪里少了一个 pop 或者 pop 到不正确的地方了等等问题。

前两个用户程序我用的是旧程序，第三和第四个用户程序我不再使用重复的三、四象限飞翔的字符，而是将第三个用户程序设置为调用 34 35 36 37 号中断，第四个用户程序为调用系统调用中断的 C 语言和汇编语言链接的程序。

第三个用户程序测试只是直接 int 34 35 36 37 就可以了，但是第四个用户程序出现了问题。

因为这个用户程序完全可以仿照之前写内核的相关规则写，所以我写得很快（就当 tasm+tcc 的又一次练手!），但是链接后的.com 文件显示为乱码。但是单独在 windows7 32 位虚拟机下执行该.com 文件时，是完全可以正常显示的，可见问题出现在调用用户程序上。突然我一拍脑袋，觉得自己真是傻呀，tasm 生成.com 文件是要 org 100h 的，但是用户程序我设置的是 7e00h，直接把这个.com

文件作为用户程序从扇区读进内存肯定会出错呀!于是我将内核的引导程序微微改动作为该用户程序的引导程序，再用引导程序读这个程序，果然可以正常显示了。对程序偏移量这个方面看来还是我不够敏感呀。

在系统调用的实现功能方面，老师考虑到两个实验合一时间不够的问题，只让我们实现了 3 个功能，其中第 0 号功能差不多可以是一个测试功能，当调用 0 号功能的 33 号中断时，在屏幕中央显示一行 hello。第二个功能我调用了上个实验中实现了的显示时间的功能。还有一个功能没有着落，我索性就添写了内核代码，增加了一个获取系统日期的模块。其实获取日期也是 1Ah 号中断的一个功能，与获取时间异曲同工，所以我便没有把这个功能作为一个创新点。3 个功能完成后，如果我们以后的实验需要更多的功能，大致框架只要仿照这个实验中实现的功能就可以了。

在这个实验中我还有不足的地方，最主要的是不知道如何实现从用户程序的返回，（在之前我是按键返回的，结果加入键盘中断后这个功能就无效了）几个用户程序测试时每次得重新进虚拟机测试，这是我本次实验的一个不足。

在这个实验中我觉得我离操作系统的基层更近了一步，之前最多也就是调用 BIOS 中断，这个实验开始，我们开始勘探中断的原理了，也涉及到了一点端口方面的知识，虽然能力有限没有实现很多很牛的功能，但是我对基本的原理有了认识，在这个实验里学到了很多东西。

六、参考资料

1、老师的课件内容

2、https://wenku.baidu.com/view/fd0ee4d428ea81c758f5788e.html BIOS 中断大全

3、https://blog.csdn.net/lulipeng_cpp/article/details/8178672 第六篇 键盘中断与应用程序读取键盘缓冲区

4、https://blog.csdn.net/zhangskd/article/details/21992933 硬中断与软中断

5、《x86 汇编语言-从实模式到保护模式》 李忠 王晓波 余洁 著