

# 操作系统 实验报告

院系：计算机学院

班级：计科 2 班

学号：19335174

姓名：施天予

指导老师：凌应标

## 一、实验题目

具有中断处理的内核

## 二、实验目的

- 1、PC 系统的中断机制和原理
- 2、理解操作系统内核对异步事件的处理方法
- 3、掌握中断处理编程的方法
- 4、掌握内核中断处理代码组织的设计方法
- 5、了解查询式 I/O 控制方式的编程方法

## 三、实验要求

- 1、知道 PC 系统的中断硬件系统的原理
- 2、掌握 x86 汇编语言对时钟中断的响应处理编程方法
- 3、重写和扩展实验三的内核程序，增加时钟中断的响应处理和键盘中断响应。
- 4、编写实验报告，描述实验工作的过程和必要的细节，如截屏或录屏，以证实实验工作的真实性

## 四、实验内容

- 1、编写 x86 汇编语言对时钟中断的响应处理程序：设计一个汇编程序，在一段时间内系统时钟中断发生时，屏幕变化显示信息。在屏幕 24 行 79 列位置轮流显示 ' | '、' / ' 和 ' \ '（无敌风火轮），适当控制显示速度，以方便观察效果，也可以屏幕上画框、反弹字符等，方便观察时钟中断多次发生。将程序生成 COM 格式程序，在 DOS 或虚拟环境运行。
- 2、重写和扩展实验三的内核程序，增加时钟中断的响应处理和键盘中断响应。，在屏幕右下角显示一个转动的无敌风火轮，确保内核功能不比实验三的程序弱，展示原有功能或加强功能可以工作。
- 3、扩展实验三的内核程序，但不修改原有的用户程序，实现在用户程序执行期间，若触碰键盘，屏幕某个位置会显示 "OUCH!OUCH!"。

4、编写实验报告，描述实验工作的过程和必要的细节，如截屏或录屏，以证实实验工作的真实性

## 五、实验方案

### 【实验环境】

- 1、实验运行环境：Windows10
- 2、虚拟机软件：VirtualBox 和 DOSBox
- 3、NASM 编译器
- 4、TCC+TASM+TLINK 混合编译

### 【实验工具】

- 1、汇编语言：NASM，TASM
- 2、文本编辑器：Notepad++
- 3、相关软件：WinHex

### 【实验思想】

#### 1、中断技术

中断是指对处理器正常处理过程的打断。中断与异常一样，都是在程序执行过程中的强制性转移，转移到相应的处理程序。

1) 硬中断（外部中断）——由外部（主要是 I/O 设备）的请求引起的中断

时钟中断（计时器产生，等间隔执行特定功能）

I/O 中断（I.O 控制器产生，通知操作完成或错误条件）

硬件故障中断（故障产生，如掉电或内存奇偶校验错误）

2) 软中断（内部中断）——由指令的执行引起的中断

中断指令（软中断 `int n`、溢出中断 `into`、中断返回 `iret`、单步中断 `TF=1`）

3) 异常/程序中断（指令执行结果产生，如溢出、除 0、非法指令、越界）

#### 2、PC 中断的处理过程

1) 硬件实现的保护断点现场

- 要将标志寄存器 `FLAGS` 压栈，然后清除它的 `IF` 位和 `TF` 位
- 再将当前的代码段寄存器 `CS` 和指令指针寄存器 `IP` 压栈

## 2) 执行中断处理程序

- 处理器获得中断号，该号码乘以 4（每个中断在中断向量表中占 4 字节），就得到了该中断处理程序入口点在中断向量表中的偏移地址
- 表中取出中断程序的偏移地址和段地址，并分别传送到 IP 和 CS，自然地，处理器就开始执行中断处理程序了。
- 由于 IF 标志被清除，在中断处理过程中，处理器将不再响应硬件中断。如果希望更高优先级的中断嵌套，可以在编写中断处理程序时，适时用 sti 指令开放中断。

## 3) 返回到断点接着执行

- 中断处理程序的最后一条指令通常是中断返回指令 iret。这将导致处理器依次从堆栈中弹出（恢复）IP、CS 和 FLAGS 的原始内容，于是转到被中断的程序接着执行

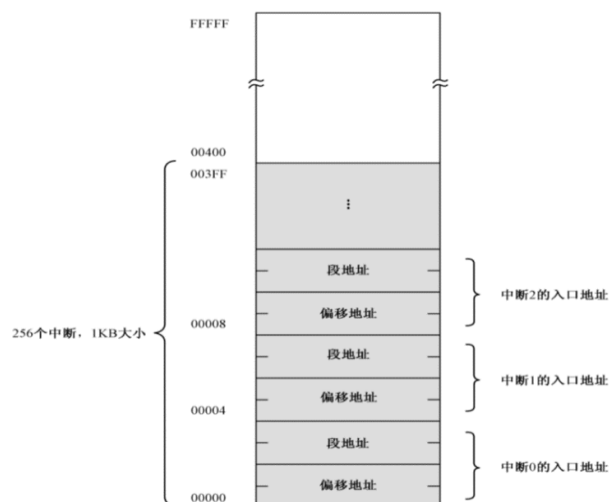
## 3、中断向量

### 1) x86 计算机在启动时会自动进入实模式状态

- 系统的 BIOS 初始化 8259A 的各中断线的类型（参见前图）
- 在内存的低位区（地址为 0~1023[3FFH]，1KB）创建含 256 个中断向量的表 IVT（每个向量[地址]占 4 个字节，格式为 16 位段值:16 位偏移值）。

### 2) 当系统进入保护模式

- IVT（Interrupt Vector Table，中断向量表）会失效
- 需改用 IDT（Interrupt Descriptor Table，中断描述表），必须自己编程来定义 8259A 的各个软中断类型号和对应的处理程序。



#### 4、中断向量表的修改

中断向量表的修改其实就是将目标中断向量的地址改成我们自己实现的中断服务程序的入口地址，根据前面的讲述，如果要修改第  $n$  号中断的中断向量，首先我们找到  $0:4n$  这个地址，然后将  $[0:4n]$  开始的两个字节修改为我们实现的中断服务程序的 IP，将  $[0:4n+2]$  开始的两个字节修改为中断服务程序的 CS。因此要修改中断向量表，要先将中断向量表的某一个中断向量转移到中断向量表的另外一个地方，再将中断服务程序写入中断向量表指定的位置。

int 指令的格式为：int  $n$ ， $n$  为中断类型码，它的功能是引发中断过程。

CPU 执行 int  $n$  指令，相当于引发一个  $n$  号中断的中断过程，执行过程如下。

- (1) 取中断类型码  $n$ ;
- (2) 标志寄存器入栈，IF=0，TF=0;
- (3) CS、IP 入栈;
- (4)  $(IP)=(n*4)$ ， $(CS)=(n*4+2)$ 。

从此处转去执行  $n$  号中断的中断处理程序。

## 六、实验过程

### 1、“无敌风火轮”汇编程序

#### 【设计分析】

“无敌风火轮”只要在右下角 (24, 79) 循环输出字符就可以了，比较简单。因为时钟中断频率太快，为防止运行速度过快导致看不清效果，我用一个 delay 变量，初始值为 5，当 delay 不为 0 时自减并中断返回，当 delay 为 0 时打印下一个字符，并将 delay 复原。旋转过程中有 3 种状态，我设定了一个计数器从 0 开始计数，当它到 3 时归 0，反复循环，“无敌风火轮”即可实现。

#### 【源代码】

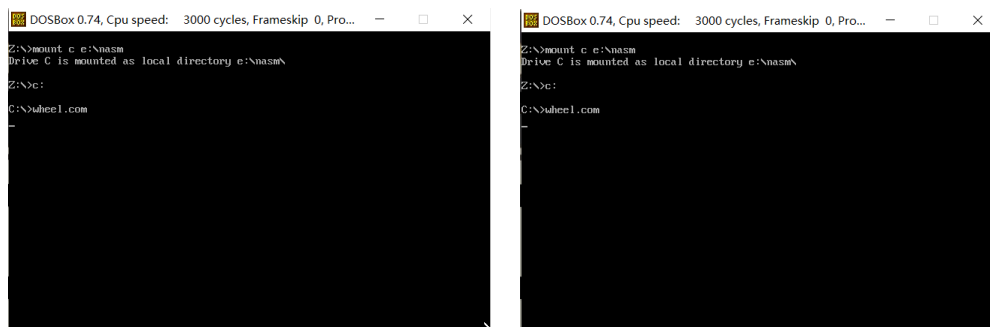
```
1      org 100h                ; 程序加载到100h, 可用于生成COM
2      ; 设置时钟中断向量 (08h), 初始化段寄存器
3      xor ax, ax              ; AX = 0
4      mov es, ax              ; ES = 0
5      mov word [es:20h], Timer ; 设置时钟中断向量的偏移地址
6      mov ax, cs
7      mov word [es:22h], ax   ; 设置时钟中断向量的段地址=CS
8      mov ds, ax              ; DS = CS
9      mov es, ax              ; ES = CS
10
11     mov ax, 0B800h           ; 文本窗口显存起始地址
12     mov gs, ax               ; GS = B800h
13     mov ah, 0Fh              ; 0000: 黑底、1111: 亮白字 (默认值为07h)
14     jmp $                    ; 死循环
```

```

15 ; 时钟中断处理程序
16 Timer:
17     dec byte [count]      ;递减计数变量
18     jnz end              ;非0则跳转返回, 5次中断和中断返回做一次切换
19     mov byte[count],delay;BS恢复计数变量
20
21     mov si,wheel
22     add si,[offset]
23     mov al,byte [si]
24     mov [gs:((80*24+79)*2)],ax ;打印
25
26     add byte[offset],1
27     cmp byte[offset],3
28     jne end
29     mov byte[offset],0
30
31 end:
32     mov al,20h            ; AL = EOI
33     out 20h,al            ; 发送EOI到主8529A
34     out 0A0h,al          ; 发送EOI到从8529A
35     iret                  ;从中断返回
36
37 datadef:
38     delay equ 5
39     count db delay
40     wheel db '|/\ '
41     offset db 0

```

## 【运行结果】



## 2、给内核添加时钟中断

根据之前的“无敌风火轮”程序，我在 kliba.asm 中加了一个 Timer 标号作为时钟中断的入口。在 monitor.asm 中调用它。

```

xor ax, ax
mov es, ax
mov word ptr es:[20h],offset Timer
mov word ptr es:[22h],cs

```

monitor.asm 调用时钟中断

```

; 时钟中断处理程序
Timer:
    push ax
    push bx
    push ds      ;保护寄存器
    push es

```

```

mov ax,cs
mov ds,ax
lea bx,count      ;判断count值是否为0，为0显示下一个"风火轮"
mov al,[bx]
dec al
mov [bx],al
cmp al,0
jnz exit
mov al,Delay
mov [bx],al
lea bx,wheel
mov al,[bx]
xor ah,ah
inc ax
cmp ax,4
jnz next
mov ax,1
next:
mov [bx],al
add bx,ax          ;[bx]是要显示的字符，加上偏移量获取
mov ax,0b800h
mov es,ax
mov al,[bx]
mov ah,0Fh         ;黑底白字
mov bx,((24*80+79)*2) ;24行79列
mov es:[bx],ax
exit:
mov al,20h         ; AL = EOI
out 20h,al         ; 发送EOI到主8529A
out 0A0h,al        ; 发送EOI到从8529A

pop es
pop ds
pop bx            ;恢复寄存器
pop ax
iret              ;中断返回

Delay equ 5
count label byte
    db Delay
wheel label byte
    db 1
    db '|'
    db '/'
    db '\'

```

kliba.asm 中“无敌风火轮”的实现，注意对寄存器的保护。

### 3、给内核添加键盘中断

当用户按下键盘的某个按键之后，键盘会通过 8259A 中断控制器向 CPU 申请中断，这是一个硬中断。中断相应后，中断控制器送出 int 09h 中断号，然后进入中断服务程序进行下一步的操作。

在这里为了实现目标，我考虑将在 int 09h 中断服务程序中添加上打印“OUCH! OUCH!”的功能。而由于中断可以嵌套，所以可以采用先将原来的 int 09h 号中断服务程序的地址转移到中断向量表其它位置，然后再由新的中断服务程序在打印“OUCH! OUCH!”后调用原来的中断服务程序。

所以具体的过程变成了：当执行用户程序的时候，当敲击键盘时，键盘通过中断控制器向 CPU 申请中断，中断响应后，中断控制器送出 int 09h 号中断向量号，然后调用中断服务程序，中断服务程序在打印完毕中“OUCH! OUCH!”调用原来的中断服务程序，然后发送 EOI 并中断返回。

```
;键盘中断
OUCH:
    push ax          ;保护寄存器
    push bx
    push cx
    push dx
    push bp
    push es
    push ds
    push si

    mov ax,cs
    mov ds,ax
    mov ax,0b800h
    mov es,ax
    lea si,len
    mov cx,[si]
    lea si,Message
    mov bx,(24*80+69)*2
    mov ah,07h

printOUCH:          ;打印字符
    mov al,[si]
    mov es:[bx],ax
    inc si
    inc bx
    inc bx
    loop printOUCH

    mov ah,6         ;清除该ouch
    mov al,0
    mov ch,24
    mov cl,69
    mov dh,24
    mov dl,79
    mov bh,7
    int 10h

    in al,60h
    mov al,20h       ; AL = EOI
    out 20h,al       ; 发送EOI到主8529A
    out 0A0h,al      ; 发送EOI到从8529A

    pop si           ;恢复寄存器
    pop ds
    pop es
    pop bp
    pop dx
    pop cx
    pop bx
    pop ax
    iret

Message db "OUCH! OUCH!"
len db 11
```

为了保证在用户程序结束后，主程序还能正常输入字符，我们在调用 loadUser() 函数开头要对原来的键盘 int 09 号中断做一个保存，在结束的地方再将它恢复即可。

```
;加载用户程序
OffsetOfUserPrg equ 8100h
public _loadUser
_loadUser proc
```



```

push ax
push bx
push cx
push dx
push ds
push es
push bp
;保护键盘中断
xor ax, ax
mov es, ax
mov bp, offset saveplace
mov word ptr ax, es: [4*9]
mov word ptr [bp], ax
mov word ptr ax, es: [4*9+2]
mov word ptr [bp+2], ax
;修改键盘中断
mov word ptr es: [4*9], offset OUCH
mov word ptr es: [4*9+2], cs

mov bp, sp
mov ax, cs ;段地址 ; 存放数据的内存基地址
mov es, ax ;设置段地址 (不能直接mov es, 段地址)
mov bx, OffSetOfUserPrg ;偏移地址; 存放数据的内存偏移地址
mov ah, 2 ; 功能号
mov al, 1 ;扇区数
mov dl, 0 ;驱动器号 ; 软盘为0, 硬盘和U盘为80H
mov dh, 0 ;磁头号 ; 起始编号为0
mov ch, 0 ;柱面号 ; 起始编号为0
mov cl, [bp+16] ;起始扇区号 ; 起始编号为1
int 13H ; 调用读磁盘BIOS的13h功能
;用户程序a.com已加载到指定内存区域
call bx ;执行用户程序

;恢复键盘中断
xor ax, ax
mov es, ax
mov bp, offset saveplace
mov word ptr ax, [bp]
mov word ptr es: [4*9], ax
mov word ptr ax, [bp+2]
mov word ptr es: [4*9+2], ax

pop bp
pop es
pop ds
pop dx
pop cx
pop bx
pop ax
ret ;中断用iret
_loadUser endp
saveplace dw 0, 0

```

#### 4、其他模块的改进与新功能

##### 1) 引导程序 (bootloader.asm)

与上次实验相同，引导程序在首扇区，4 个用户程序放在 2、3、4、5 扇区，内核从第 6 扇区开始。因为这次内核的功能增加了，所以内核代码一共占了 5 个扇区，引导扇区仅在读扇区数上改变即可。

## 2) 内核

内核的功能与上次实验基本相同，我在网上又学到了一些底层实现关机、重启等功能的知识，于是在内核中新增了这两个功能。

```
;关机
public _PowerOff
_PowerOff proc
    mov ax,5307H ;高级电源管理功能,设置电源状态
    mov bx,0001H ;设备ID1:所有设备
    mov cx,0003H ;状态3:表示关机
    int 15H
_PowerOff endp

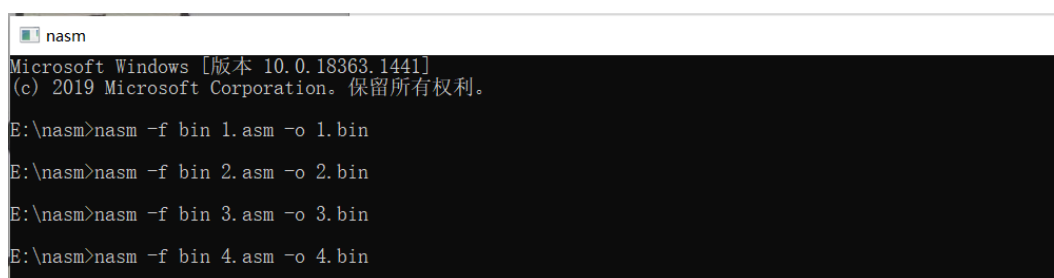
;重启
public _Reboot
_Reboot proc
    mov al, 0FEh
    out 64h, al
    ret
_Reboot endp
```

## 3) 用户程序

用户程序与之前基本一样，不过这次实验我把用户程序 3 和 4 也改成彩色的了，这样美观一些，之前偷懒一直没改。。。因为 int 09 号中断的更改会影响输入功能，所以在用户程序中不能再使用“空格”返回了，我设置了一个计数器自减到 0 后返回主程序。

## 4) NASM 编译与 TCC-TASM-TLINK 混合编译链接

用户程序和引导程序用 NASM 编译



```
nasm
Microsoft Windows [版本 10.0.18363.1441]
(c) 2019 Microsoft Corporation。保留所有权利。

E:\nasm>nasm -f bin 1.asm -o 1.bin
E:\nasm>nasm -f bin 2.asm -o 2.bin
E:\nasm>nasm -f bin 3.asm -o 3.bin
E:\nasm>nasm -f bin 4.asm -o 4.bin
```

```
E:\nasm>nasm -f bin bootloader.asm -o bootloader.bin
```

内核的三个程序用 TCC-TASM-TLINK 混合编译链接

```
DOSBox 0.74, Cpu speed: 3000 cycles, Frameskip 0, Pro...
C:\>tcc kernal.c
Turbo C Version 2.01 Copyright (c) 1987, 1988 Borland International
kernal.c:
Turbo Link Version 2.0 Copyright (c) 1987, 1988 Borland International
c0s.obj : unable to open file

        Available memory 448686

C:\>tasm monitor.asm
Turbo Assembler Version 4.1 Copyright (c) 1988, 1996 Borland International

Assembling file:  monitor.asm
Error messages:  None
Warning messages: None
Passes:         1
Remaining memory: 466k

C:\>tlink /3 /t monitor.obj kernal.obj,kernal.com
Turbo Link Version 2.0 Copyright (c) 1987, 1988 Borland International
```

## 5) 使用 WinHex 修改软盘

使用 WinHex 打开所有的 COM 文件和 BIN 文件。将引导程序 bootloader.bin 放入首扇区，将 4 个用户程序分别放入 2-5 扇区，将内核 kernal.com 放入第 6 扇区。

## 6) 用软盘启动裸机

操作系统界面

```
*****
*                               Welcome to my OS!                               *
*                               Enjoy yourself!                                *
*                               STY 19335174                                  *
*****

Enter help to get the command list

>>help
help:  get the command list
cls:   clear the screen
u1:    run the user program 1
u2:    run the user program 2
u3:    run the user program 3
u4:    run the user program 4
runall: run all the user programs
ls:    get the file list
restart: restart the OS
quit:  exit the OS

>>_
```

文件列表

```
>>ls
The Number Of Files: 4
1.Number      402B
2.Name        397B
3.Rectangle   298B
4.Stone        393B
>>
```

用户程序 1

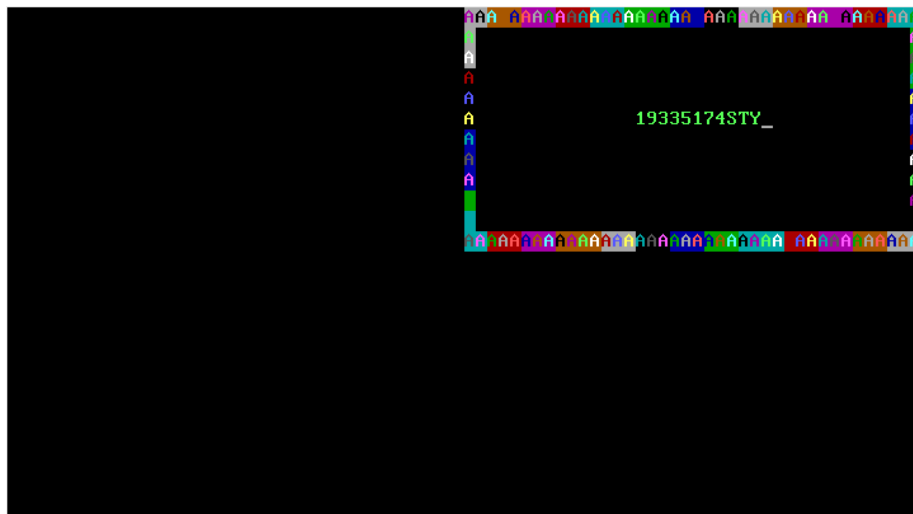
```
19335174      193319335174
19335174      19335119335174
19335174      1933517419335174
19335174      19335174 19335174
19335174      19335174 1933517474
19335174      19335174 1933517474
19335174 19335174 1933517474
1933517419335174 1933517474
19335119335174 1933517474
193319335174 1933517474
1919335174 1933517474
19335174 19335174
-
OUCH! OUCH!
```

按下任意按键，出现“OUCH! OUCH!”

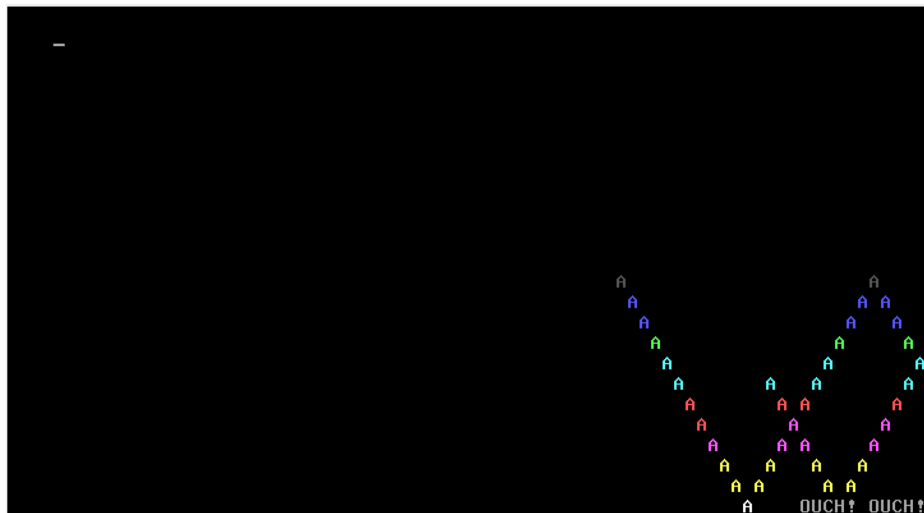
用户程序 2

```
STY      STY      STY      STY
STY      STYTY      STSTY      STYTY
STY STY STY STY STY STY STY STY
STY STY STY STY STY STY STY
STY STYTY STYTY STY STY
STYTY STY STYTY STY STY
STY STY STSTY STY STY STY
STY STY STY STY STY STY STY
STY STY STY STY STY STY
STY STSTY STYTY STSTY
STY STY STY STY
```

### 用户程序 3

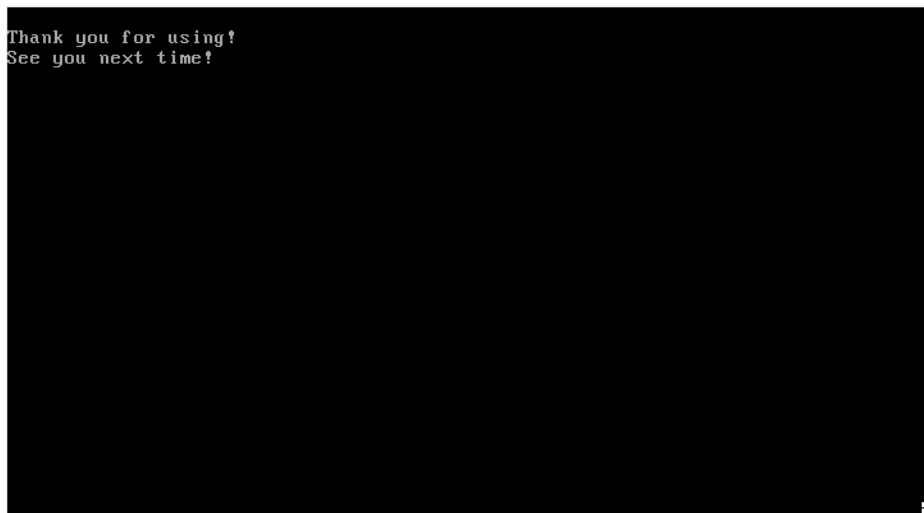


### 用户程序 4



按下任意按键，出现“OUCH! OUCH!”

### 退出界面



按下任意按键后关闭电源！

**Runall 命令**是依次执行四个用户程序，截图很难显示动画的效果，我在“运行视频”中有演示。

## 七、实验总结

这次实验中没有上次那么复杂，但也十分麻烦，也让我学到了很多東西。我对中断的含义和处理更深刻了，清楚了 COM 文件的组织，也学会了如何修改中断向量表。

时钟中断相对比较简单，我直接对老师的模板进行修改就完成了。但当我想把开始写的程序直接加入内核时却遇到了问题，因为我第一个在 DOSBox 的“无敌风火轮”是使用 NASM 编译的，而我的内核是 TCC-TASM-TLINK 混合编译，所以语法有一些不同，我做了一定修改才完成。

键盘中断比时钟中断要难一些，因为 int 09h 号中断改了，我的用户程序也不能使用“空格”返回了，所以我改成了延时退出。要把原来的 int 09h 号中断保存到一个地址，在用户程序结束后再将键盘中断改回来。在显示字符串“OUCH! OUCH!”时我本来想用 int 10h 的输出字符串功能，但我意外地发现除非我的用户程序进入失败，“OUCH! OUCH!”会显示，否则进入用户程序后，按下任意按键，只会停顿一下而不会显示“OUCH! OUCH!”。这个问题困扰了我很久，也不知道是什么原因。最后我只好使用在显存 B800h 循环输出字符的方法显示“OUCH! OUCH!”。

总而言之，这次实验也花了我很多时间，但也让我受益匪浅，经过这几次实验我的汇编编程水平也提升了不少。希望在后面的实验中，我也能再接再厉，勇往直前！

## 八、参考文献

《汇编语言（第3版）》

《X86 汇编语言：从实模式到保护模式》

[https://blog.csdn.net/csdn\\_blog\\_lcl/article/details/54926726](https://blog.csdn.net/csdn_blog_lcl/article/details/54926726)

<http://www.myexceptions.net/assembly-language/288142.html>