

实验四：中断与系统调用

实验者：张海涛，张晗宇，张浩然，张镓伟
学号：15352405, 15352406, 15352407, 15352408
院系：数据科学与计算机学院
专业：15 级软件工程（移动信息工程）
指导老师：凌应标

【实验题目】

在设计的操作系统中添加中断和系统调用

【实验目的】

1. 学习中断机制知识，使用中断处理程序设计。
2. 设计一个汇编程序，实现时钟中断处理程序。
3. 扩展 MyOS3，增加时钟中断服务，利用时钟中断实现与时间有关的操作
4. 实现简单的系统调用

【实验要求】

1. 操作系统工作期间，利用时钟中断，在屏幕 24 行 79 列位置轮流显示 ' | '、' / ' 和 ' \ '，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 33 号、34 号、35 号和 36 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 33、int 34、int 35 和 int 36 产生中断调用你这 4 个服务程序。

【实验方案】

一. 虚拟机配置方法

无操作系统，10M 硬盘，4MB 内存，启动时连接软盘

二. 软件工具和作用

Notepad++: 用于生成. 汇编语言文件

Nasm: 用于编译. asm 类型的汇编语言文件，生成 bin 文件

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

TCC: 用于由. c 文件生成. obj 文件

TNASM: 用于由. asm 文件生成. obj 文件。

Tlink: 用于链接 obj 文件生产. com 可执行文件

三. 实验原理

(1) 中断技术

计算机硬件系统提供中断技术，支持 CPU 与外部设备的并发工作，也利用中断技术处理硬件错误、支持程序调试、实现软件保护和信息安全等。

中断(interrupt)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。

硬中断（外部中断）——由外部（主要是外设[即 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）

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`）

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）系统调用

操作系统提供的服务可以用多种方式供用户程序使用

①子程序库静态链接：采用子程序调用方式，如汇编语言中用 `call` 指令调用操作系统提供服务的子程序，静态链接到用户程序代码中，这种方式优点是程序执行快，最大缺点是用户程序内存和外存空间占用多，子程序库管理维护工作复杂。

②内核子程序软中断调用：采用软中断方式，如汇编语言中用 `int` 指令调用操作系统提供服务的子程序，系统服务的子程序在内核，这种方式的优点是服务由系统提供，程序效率较高，且被所有用户程序代码共享，有利于节省内存，最大缺点是需要防止内核再入或内核设计为可再入，且用户程序陷入内核和内核返回用户程序的开销较大。

③子程序库动态链接：采用动态链接技术，操作系统在运行时响应子程序调用，加载相应的子服务程序并链接致用户地址空间，这种方式优点是可由多方提供服务程序，服务更内容丰富，增加和变更服务方便，最大缺点是链接耗时多，程序响应变慢，实现复杂。

BIOS 调用

①与内核子程序软中断调用方式原理是一样的

②每一种服务由一个子程序实现，指定一个中断号对应这个服务，入口地址放在中断向量表中，中断号固定并且公布给用户，用户编程时才可以中断调用，参数传递可以使用栈、内在单元或寄存器。

系统调用

①因为操作系统要提供的服务更多，服务子程序数量太多，但中断向量有限，因此，实际做法是专门指定一个中断号对应服务处理程序总入口，然后再将服务程序所有服务用功能号区分，并作为一个参数从用户中传递过来，服务程序再进行分支，进入相应的功能实现子程序。

②这种方案至少要求向用户公开一个中断号和参数表，即所谓的系统调用手册，供用户使用。

③如果用户所用的开发语言是汇编语言，可以直接使用软中断调用

④如果使用高级语言，则要用库过程封装调用的参数传递和软中断等指令汇编代码

⑤规定系统调用服务的中断号是 21h。

四. 程序功能

开机进入内核，内核提供 3 种指令输入，例如：

```

Welcome to MYOS
Please input your instructions
A.Check the information of the OS
B.Run the programs in sequence(you can choose 1~5 to run)
  e.g B 1 2 3 4 5(Then programs will be run in such a order)
C.Run the programs simultaneously(you can choose 1~4 to run)
  e.g C 1 2 3 (Then program 1,2,3 will be run at the same time)
D.Run as default
MYOS> _
```

1. **时钟中断：** 在内核运行期间， 会在屏幕右下角 24 行 79 列位置循环变色显示 ‘|’ 、 ‘/’ 、 ‘\’ 。

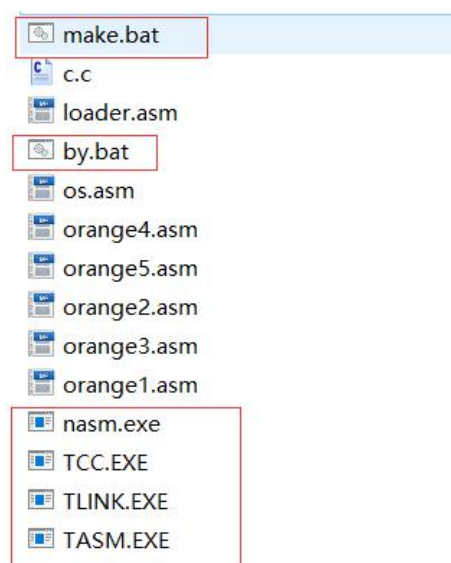
在进入用户程序之前， 恢复默认中断向量表以停止字符显示；在从用户程序返回内核后， 重新设置中断向量表继续循环变色显示三个字符。

2. **键盘中断：** 在内核进入用户程序前， 修改当前键盘中断向量表， 使用户程序中每次按键盘都会变色显示 “Ouch!Ouch!” ， 且显示位置会不断变成上次显示的位置之后。 从用户程序返回内核后， 恢复默认中断向量表， 以保证内核正常的输入功能。

3. **系统调用功能：**提供 INT 33H, INT 34H, INT 35H, INT 36H 的系统调用，可分别在屏幕左上方、右上方、左下方、右下方显示特定信息。
4. **显示用户程序的信息：**输入指令 A，则屏幕上会显示每个用户程序的相关信息。
5. **批处理功能：**指令 B 之后，依次输入想执行的用户程序的编号，范围 1~5，之后程序会依次调用相应的用户程序去执行。其中前 4 个用户程序为实验 3 中的 4 个用户程序，第 5 个用户程序为新增的准们测试 INT 33H, INT 34H, INT 35H, INT 36H 4 个系统调用的程序。输入指令 D 会默认依次执行 1、2、3、4 这 4 个用户程序。
6. **分时功能：**指令 C 之后，依次输入想执行的用户程序的编号，范围 1~4，则程序会同时调用这些用户程序去执行。这里没有程序 5 是因为程序 5 会霸占整个屏幕。
7. **命令行输入：**左下的 MYOS>处输入命令，此处模仿 cmd，在输入过程中，若输入错误，可以进行删除一定字符以修正输入。如果输入了错误的命令并按了回车，则会有报错信息。

五. 编译流程

由于之前几次实验每次编译都要按好多命令，大家深感其烦，所以这次我们使用了脚本完成自动化编译。首先在含代码的文件夹中将新建 by.bat 和 make.bat。并将 nasm.exe, TCC.exe, TLINK.exe, TASM.exe 复制到该文件夹里。完成后如下图：



在 make.bat 文件中键入如下内容：

```
del *.obj
del *.bin
del *.img
del *.com
del *.map
tcc -c c.c
tasm os.asm os.obj
tlink/3/t os.obj c.obj, os.com
```

前面几句 del 的几句诗删除以*后面位后缀结尾的那些编译生成的文件，然后是内核的 C 与汇编的混合编译。

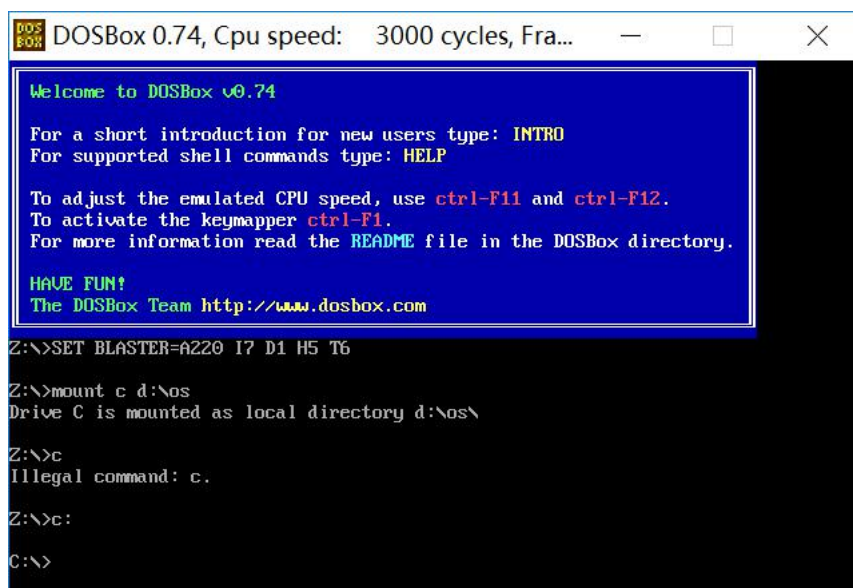
在 by.bat 中键入如下内容：

```
nasm orange1.asm -o orange1.bin
nasm orange2.asm -o orange2.bin
nasm orange3.asm -o orange3.bin
nasm orange4.asm -o orange4.bin
nasm orange5.asm -o orange5.bin
nasm loader.asm -o myos.img
pause
```

这是用来编译用户程序和引导程序的，最后的 pause 是为了防止命令完成后 cmd 框消失而看不到如果代码有错而产生的错误信息。

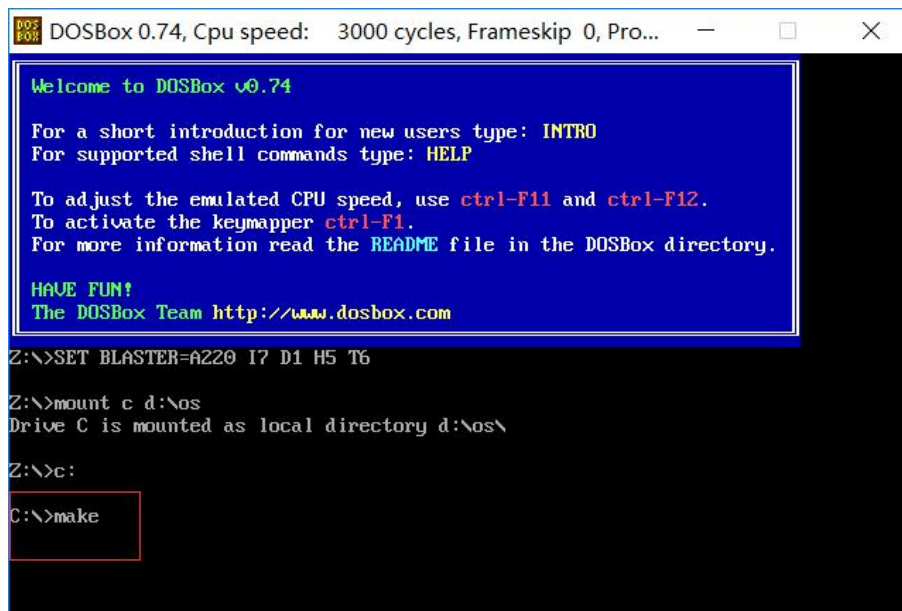
之所以不把 make.bat 和 by.bat 的内容写在一起是因为, make.bat 是要在 DOSBOX 中运行的，而在 DOSBOX 中无法运行 nasm，所以 nasm 之内另外写在 by.bat 中在本机环境下执行。

打开 DOSBOX，将这个文件夹创建为一个虚拟盘，在这个虚拟盘下便于操作。



```
DOSBox 0.74, Cpu speed: 3000 cycles, Fra...
Welcome to DOSBox v0.74
For a short introduction for new users type: INTRO
For supported shell commands type: HELP
To adjust the emulated CPU speed, use ctrl-F11 and ctrl-F12.
To activate the keymapper ctrl-F1.
For more information read the README file in the DOSBox directory.
HAVE FUN!
The DOSBox Team http://www.dosbox.com
Z:\>SET BLASTER=A220 I7 D1 H5 T6
Z:\>mount c d:\os
Drive C is mounted as local directory d:\os\
Z:\>c
Illegal command: c.
Z:\>c:
C:\>
```

输入命令 make 并回车执行自动编译。



DOSBox 0.74, Cpu speed: 3000 cycles, Frameskip 0, Pro...

```
Welcome to DOSBox v0.74

For a short introduction for new users type: INTRO
For supported shell commands type: HELP

To adjust the emulated CPU speed, use ctrl-F11 and ctrl-F12.
To activate the keymapper ctrl-F1.
For more information read the README file in the DOSBox directory.

HAVE FUN!
The DOSBox Team http://www.dosbox.com

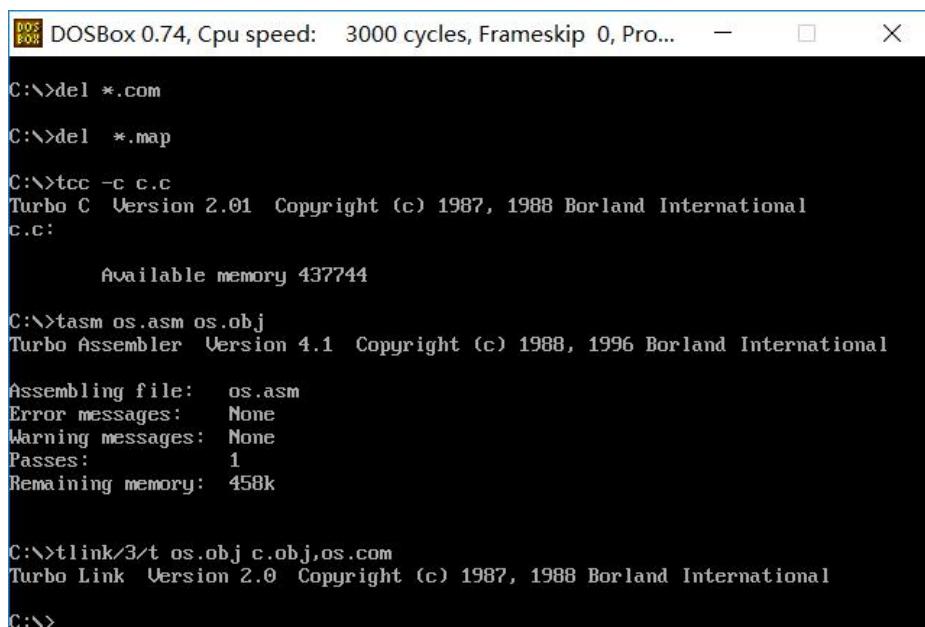
Z:\>SET BLASTER=A220 I7 D1 H5 T6

Z:\>mount c d:\os
Drive C is mounted as local directory d:\os\

Z:\>c:

C:\>make
```

自动编译效果:



DOSBox 0.74, Cpu speed: 3000 cycles, Frameskip 0, Pro...

```
C:\>del *.com

C:\>del *.map

C:\>tcc -c c.c
Turbo C Version 2.01 Copyright (c) 1987, 1988 Borland International
c.c:

        Available memory 437744

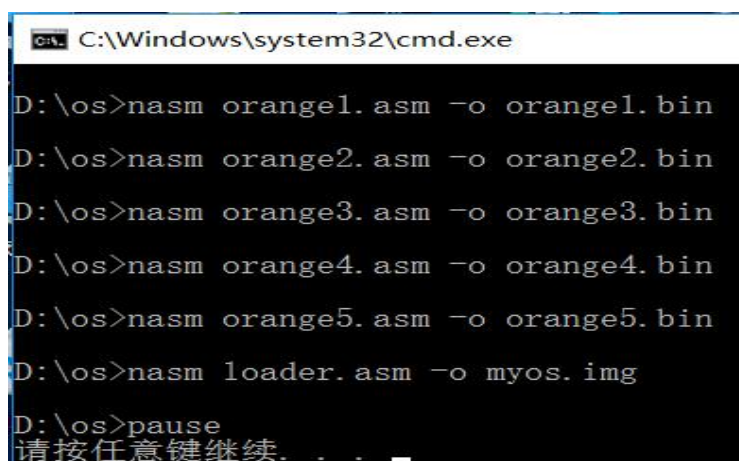
C:\>tasm os.asm os.obj
Turbo Assembler Version 4.1 Copyright (c) 1988, 1996 Borland International

Assembling file:  os.asm
Error messages:  None
Warning messages: None
Passes:         1
Remaining memory: 458k

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

C:\>
```

在代码文件目录下双击 by. bat 继续完成剩下的编译。



C:\Windows\system32\cmd.exe

```
D:\os>nasm orange1.asm -o orange1.bin
D:\os>nasm orange2.asm -o orange2.bin
D:\os>nasm orange3.asm -o orange3.bin
D:\os>nasm orange4.asm -o orange4.bin
D:\os>nasm orange5.asm -o orange5.bin
D:\os>nasm loader.asm -o myos.img
D:\os>pause
请按任意键继续. . .
```

六.程序关键代码及解释

【内核】

内核在上一次实验基础上修改，并且增加了时间中断、键盘中断、自己编写的中断程序的内容以及需要用到的一些新函数。

(1) 时钟中断

①设置时间中断

首先使用 cli 关闭中断，然后压栈保存 es 和 ax。将 es 置为 0。先保存原本的时钟中断向量到 clock_vector，然后将 Paint 的偏移量和段地址存入时钟中断，出栈，打开中断。

```
set_clock_interrupt proc
;关闭中断
cli
;入栈保存
push es
push ax
;将es置0
xor ax,ax
mov es,ax
;保存原本的中断向量
mov ax,word ptr es:[20h]
mov word ptr [clock_vector],ax
mov ax,word ptr es:[22h]
mov word ptr [clock_vector+2],ax
;将Paint的偏移和段地址填入
mov word ptr es:[20h],offset Paint
mov ax,cs
mov word ptr es:[22h],ax
;出栈
pop ax
pop es
;打开中断
sti
ret
set_clock_interrupt endp
```

②恢复时间中断

关闭中断，入栈保存，es 置 0，从 clock_vector 恢复原本的时钟中断向量，出栈，打开中断。

```

re_clock_interrupt proc
    cli
    push es
    push ax
    ;es置零
    xor ax,ax
    mov es,ax
    ;恢复原本的时钟中断的中断向量
    mov ax,word ptr [clock_vector]
    mov word ptr es:[20h],ax
    mov ax,word ptr [clock_vector+2]
    mov word ptr es:[22h],ax
    pop ax
    pop es
    sti
    ret
re_clock_interrupt endp

```

③时钟中断服务程序

关闭中断，入栈保存，调用 c 中的程序来绘制 ‘\’、‘|’、‘/’ 轮流变化的三个字符。发送 EOI 到主 8529A 和从 8529A，出栈，打开中断，然后从中断返回。

```

Paint proc
    cli
    push es
    push si
    push di
    push ax
    push bx
    push cx
    push dx
    push bp
    push ds

    call near ptr _paint

    mov al,20h
    out 20h,al
    out 0a0h,al

    pop ds
    pop bp
    pop dx
    pop cx
    pop bx
    pop ax
    pop di
    pop si
    pop es
    sti
    iret
Paint endp

```

④c 中 Paint 函数

函数本身不是很难，不过需要注意一下 c 中显示需要两个\\才能表示\。


```

int change,begin,px,py,pcolor;
char pch;
int px,py,pcolor,pdir,pbegin;
char pch;
paint()
{
    if(pbegin!=1)
    {
        px=24;
        py=79;
        pcolor=1;
        change=0;
        pdir=0;
        pch='|';
        pbegin=1;
    }
    change=(change+1)%45;
    if (change==0)pch='|';
    if (change==15)pch='/';
    if (change==30)pch='\\';
    put_color_char(pch,px,py,pcolor);
    pcolor=(pcolor+1)%15;
}

```

(2) 键盘中断

键盘中断本身与时钟中断是类似的，区别除了中断号不同主要在键盘中断服务程序和时间中断不同。

①设置键盘中断

```

set_keyboard_interrupt proc
    cli
    push es
    push ax
    xor ax,ax
    mov es,ax
    ;save the vector
    mov ax,word ptr es:[24h]
    mov word ptr [keyboard_vector],ax
    mov ax,word ptr es:[26h]
    mov word ptr [keyboard_vector+2],ax
    ;fill the vector
    mov word ptr es:[24h],offset Ouch
    mov ax,cs
    mov word ptr es:[26h],ax
    ;用于中断服务c程序的变量
    mov word ptr [_ifouch],1
    pop ax
    pop es
    sti
    ret
set_keyboard_interrupt endp

```

②恢复键盘中断

```

;恢复键盘中断
re_keyboard_interrupt proc
    cli
    push es
    push ax
    xor ax,ax
    mov es,ax
    mov ax,word ptr [keyboard_vector]
    mov word ptr es:[24h],ax
    mov ax,word ptr [keyboard_vector+2]
    mov word ptr es:[26h],ax
    ;用于中断服务c程序的变量
    mov word ptr [_ifouch],0
    pop ax
    pop es
    sti
    ret
re_keyboard_interrupt endp

```

③键盘中断服务程序

```

Ouch proc
    cli
    push es
    push si
    push di
    push ax
    push bx
    push cx
    push dx
    push bp
    push ds

    call near ptr _ouch
    ;读缓冲区
    in al,60h

    mov al,20h
    out 20h,al
    out 0a0h,al

    pop ds
    pop bp
    pop dx
    pop cx
    pop bx
    pop ax
    pop di
    pop si
    pop es
    sti
    iret
Ouch endp

```

④c 中 ouch 函数

```

int ifouch;
int ouchcolor;
ouch()
{
    if(ifouch){
        x=y=pos=0;
        ifouch=0;
        ouchcolor=1;
    }
    printstr(ouchcolor,"ouch!!!");
    ouchcolor=(ouchcolor+1)%15;
}

```

(3) 自己编写系统中断

四个系统中断分别是在四个 1/4 屏幕显示文字。

以 int 33 为例

①填充中断向量表

```

int33h proc
    cli
    push es
    push ax
    ;es置零
    xor ax,ax
    mov es,ax
    ;填充中断向量
    mov word ptr es:[0cch],offset interrupt33h
    mov ax,cs
    mov word ptr es:[0ceh],ax
    pop ax
    pop es
    sti
    ret
int33h endp

```

②中断处理程序

```

message33h1 db 'This is program of INT 33H'
message33h1length equ $-message33h1
message33h2 db '-----Zhang Haitao-----'
message33h2length equ $-message33h2
message33h3 db 'Head of the dormitory 714'
message33h3length equ $-message33h3
;中断处理程序33h
interrupt33h proc
    cli
    push es
    push si
    push di
    push ax
    push bx
    push cx
    push dx
    push bp
    push ds

    mov ax,cs
    mov es,ax
    mov ax,1301h
    mov bx,0001h;b1颜色
    mov dx,0408h;行,列
    mov cx,message33h1length
    mov bp,offset message33h1
    int 10h

```

```

mov ax,cs
mov es,ax
mov ax,1301h
mov bx,0001h;b1颜色
mov dx,050dh;行,列
mov cx,message33h2length
mov bp,offset message33h2
int 10h

mov ax,cs
mov es,ax
mov ax,1301h
mov bx,0001h;b1颜色
mov dx,0608h;行,列
mov cx,message33h3length
mov bp,offset message33h3
int 10h

mov al,20h
out 20h,al
out 0a0h,al

pop ds
pop bp
pop dx
pop cx
pop bx
pop ax
pop di
pop si
pop es
sti
iret
interrupt33h endp

```

(4) 中断的位置

在开始时调用四个新写的中断和设置时钟中断的过程，然后调用 c 中主函数。

```

start:
    mov ax, cs
    mov ds, ax
    mov es, ax
    mov ss, ax
    mov sp, 100h

    call int33h
    call int34h
    call int35h
    call int36h
    call set_clock_interrupt
    call near ptr _cmain
    jmp $

```

在处理跳转时，先恢复原本时钟中断，再设置键盘中断，接着再跳转到用户程序，然后恢复键盘中断，设置时钟中断。实现了在运行用户程序前运用时钟中断显示变化字符，运行用户程序时运用键盘中断显示 ouch 的效果。

```

public _jmp
_jmp proc
    push ax
    push bx
    push cx
    push dx
    push es
    push ds
    call re_clock_interrupt
    call set_keyboard_interrupt
    call word ptr [_offset_user]
    call re_keyboard_interrupt
    call set_clock_interrupt
    pop ds
    pop es
    pop dx
    pop cx
    pop bx
    pop ax
    ret
_jmp endp

```

【引导程序】

引导程序与上次变化不大，主要的区别在于扇区数和起始扇区号的更新，以及将段地址与偏移地址分开。

```

org 7c00h ; BIOS将把引导扇区加载到0:7c00h处，并开始执行
SegOfKernal equ 8000h ;内核加载到的段地址
OffsetOfKernal equ 100h;内核加载到的偏移地址
Start:
    ;读软盘或硬盘上的若干物理扇区到内存的ES:BX处:
    mov ax,SegOfKernal ;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,OffsetOfKernal ;偏移地址; 存放数据的内存偏移地址
    mov ah,2 ; 功能号
    mov al,10 ;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,12 ;起始扇区号 ; 起始编号为1

    int 13H ; 调用读磁盘BIOS的13h功能
    jmp SegOfKernal:OffsetOfKernal

```

【用户程序】

前面四个用户程序基本不变，仅把开头确定位置改为使用 `org` 来完成。

新加一个用户程序来处理四个自己写的中断。

```

org 0A100h
Start:
    mov ax,cs
    mov ds,ax
    mov es,ax
    int 33h
    int 34h
    int 35h
    int 36h
    call DELAY
    ret
DELAY:
DELAY1:
    push cx
    mov cx,0ffffh
DELAY2:
    loop DELAY2
    pop cx
    loop DELAY1
    ret
Define:
    times 1022-($-$$) db 0
    db 0x55,0xaa

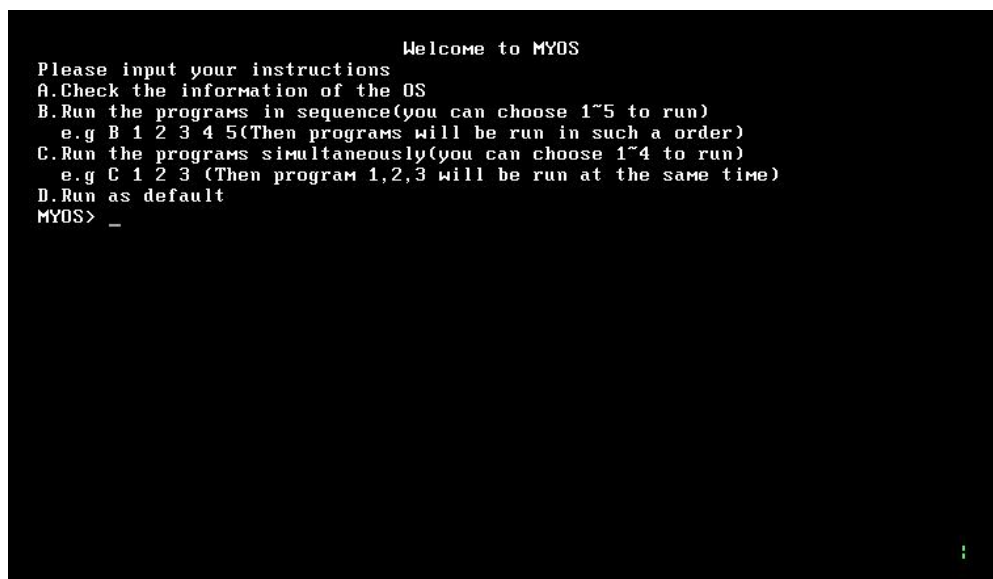
```

七.程序运行流程

以下为连续的依次操作。

内核界面：

可以看到除了在实验三中已有的内容外，右下角还有一个彩色的字符。



```

                                Welcome to MYOS
Please input your instructions
A.Check the information of the OS
B.Run the programs in sequence(you can choose 1~5 to run)
  e.g B 1 2 3 4 5(Then programs will be run in such a order)
C.Run the programs simultaneously(you can choose 1~4 to run)
  e.g C 1 2 3 (Then program 1,2,3 will be run at the same time)
D.Run as default
MYOS> _

```

输入命令 A，显示程序信息。

```

Welcome to MYOS
Please input your instructions
A.Check the information of the OS
B.Run the programs in sequence(you can choose 1~5 to run)
  e.g B 1 2 3 4 5(Then programs will be run in such a order)
C.Run the programs simultaneously(you can choose 1~4 to run)
  e.g C 1 2 3 (Then program 1,2,3 will be run at the same time)
D.Run as default
MYOS> A _

```

```

USER PROGRAM1-Name:function1 Size:1024byte Position:section 2
USER PROGRAM2-Name:function2 Size:1024byte Position:section 4
USER PROGRAM3-Name:function3 Size:1024byte Position:section 6
USER PROGRAM4-Name:function4 Size:1024byte Position:section 8
USER PROGRAM5-Name:function5 Size:1024byte Position:section 10
Input anything and return
MYDS> _

```

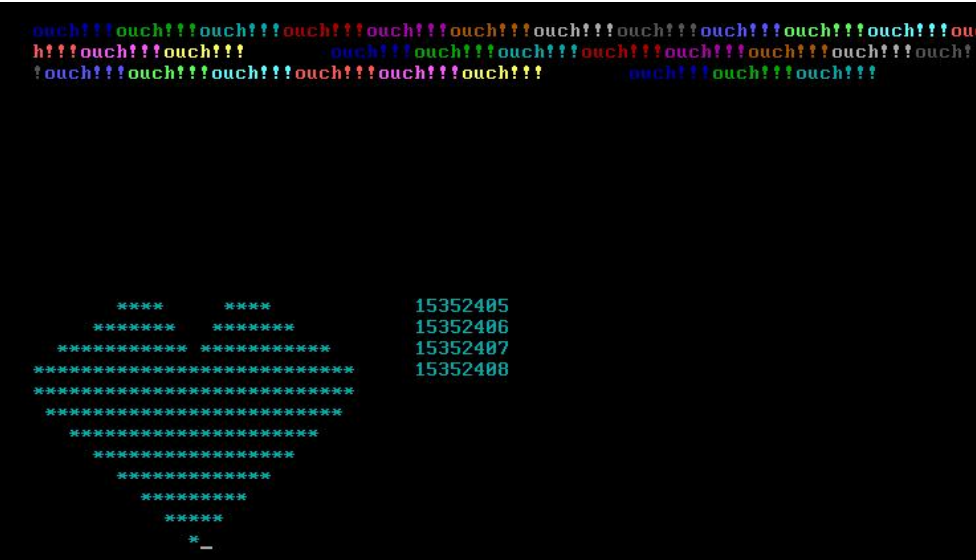
按任意键返回再输入 B1 运行用户程序 1，可以 OUCH!

```
ouch!!!ouchB!C!ouch!!!ouchC!!!ouchB!Couch!!!ouch!!!ouch!!!ouch!!!ouch!!!ouch
h!!!ouch!B!AuCh!!!      D Cuch!BoAch!!!ouch!!!ouch!!!ouch!!!ouch!!!ouch!!!ouch!!!
!ouch!!!BuDhA!Couch!!!DuchC!oBch!A!ouch!!!      ouch!!!ouch!!!ouch!!!ouch!!!ou
ch!!!ouBhD!oAch!!!ouDh!!!oCcB!!!ouAh!!!ouch!!!ouch!!!ouch!!!_
  A B D      A C B D      A C D      A C
    B D      15352405      15352406      A B
    B D      15352407      15352408      C      A
    B D A      B D C      B D A C      B A
    B D      A      B D A C      B D      A C      B D
    B D      A      B      A C B D      A C B D
C D      A B      A B D      A C D
  D      A      A D      A D
```

用户程序1结束后输入任意命令返回再输入B2 运行用户程序2,可以OUCH!



用户程序 2 结束后输入任意命令返回再输入 B 3 运行用户程序 3, 可以 OUCH!



用户程序 3 结束后输入任意命令返回再输入 B 4 运行用户程序 4, 由于这个程序只是显示一个日期, 一瞬间就执行完了, 所以是没有机会 ouch!的




```
ouch!!!!ouch!!!!ouch!!!!ouch!!!!ouch!!!!ouch!!!!ouch!!!!ouch!!!!ouch!!!!ouc  
h!!!!ouch!!!!ouch!!!!      ouch!!!!ouch!!!!ouch!!!!ouch!!!!ouch!!!!ouch!!!!ouch!!!!ouch!!!!ouch!!!!  
!ouch!!!!ouch!!!!ouch!!!!ouch!!!!_
```

| | |
|---|--|
| This is program of INT 33H -----Zhang Haitao----- Head of the dormitory 714 | This is program of INT 34H -----Zhang Hanyu----- Member of the dormitory 714 |
| This is program of INT 35H -----Zhang Haoran----- Member of the dormitory 714 | This is program of INT 36H -----Zhang Jiawei----- Member of the dormitory 714. |

The image shows a terminal window with a complex ASCII art pattern. The pattern is composed of various characters including 'ouch!', 'B!', 'C!', 'A', 'B', 'C', 'D', and 'AuCh!!!'. The characters are arranged in a grid-like structure, with some characters appearing in multiple colors (red, green, blue, yellow, and white) against a black background. The pattern is symmetrical and has a central area that appears to be a placeholder or a specific data field.

[illegible]

[illegible][illegible][illegible]

```
*****
* DATE:2017/05/05 *
*****
```

内核界面输入错误命令，会有报错信息：

```
Welcome to MYOS
Please input your instructions
A.Check the information of the OS
B.Run the programs in sequence(you can choose 1~5 to run)
  e.g B 1 2 3 4 5(Then programs will be run in such a order)
C.Run the programs simultaneously(you can choose 1~4 to run)
  e.g C 1 2 3 (Then program 1,2,3 will be run at the same time)
D.Run as default
MYOS> op_

Your instructions cannot be recognized.Try again.
Input anything and return
MYOS> _
```

【实验总结】

张浩然的实验总结：

这次实验目前为止最花时间的一次。王师兄的代码量很多，一开始很没底，不过由于有不少内容本质上是重复的，并且我们的进度还没到文件系统，所以实际上工作量比想象的要好很多。

这次的实验主要是中断。除了对于写中断的学习，我还有很多概念和规则上查漏补缺的地方。和很多同学一样，我们这次在实验中也遇到了很多同学都遇到的问题，那就是字符串的显示有问题，不管是用 bios 中断写的字符串显示，还是用循环写出来的字符串显示都无法显示。虽然我是知道是段地址与偏移位置出的问题，但是具体是哪里错了，该怎么修改并没有概念，最后在一个同学的指导下修改了用户程序确认开始位置的方式——使用 org 来确定位置。这个问题修改完，四个子程序的字符串都没有问题。

此外我们还遇到了在内核部分场合显示字符串会多出一些乱码的内容，因为只会在两个固定的情况下会出现这种情况，并没有看出和内核其他显示字符串的

地方有什么不同会导致乱码，所以我们采用了清屏来避开了乱码，虽然没有彻底解决，不过比较好地用来不错的方法规避这个问题。

张海涛的实验总结：

这次实验主要针对设置中断及系统调用，其中最大的难点在用段地址的设置和程序的跳转。我们在实验中遇到许多问题，其中最困难的一个问题就是：字符显示。我们一直各种尝试，都以失败告终，最后我们了解到别人也遇到了这样的问题，交流之后，我们得知问题在于段地址的设置，通过计算出子程序正确的段地址，我们解决了这个问题。

除了实验过程中对问题的解决，我们还加深了操作系统的理解。内核提供了程序之间的调度功能，而操作系统对进行中的程序的管理是最关键的部分。而通过时间中断的管理是重要方法之一，通过时钟中断和 BIOS 提供的服务，能处理计算机在运行过程中遇到的情况。

还有就是对操作系统中对寄存器族的使用，这是操作系统中实现快速跳转的关键。

四个通用寄存器的使用，还有段寄存器，标志寄存器。这是汇编语言和高级语言的重要区别，也是我们容易弄错的地方。因为运算过程中所需要的数据都是从这些有限的寄存器中获取，在代码中不用看到寄存器的变换，这使得程序运行的状态变得难以捕捉。

张晗宇的实验总结：

这次的实验虽然是很多周来实现，但是一开始被王师兄的 1200 行代码吓到了，这次的实验内容主要还是中断程序的调用，在 C 语言框架部分，加入两个函数，一个用来在右下角显示字符，一个用来在输入时输出 `ouch`，由于对段地址和偏移地址的理解不够到位，使得在最初的尝试中出现乱码现象，不仅仅是用户程序的字符无法显示，时钟中断以及键盘中断的字符显示都出现乱码，最后修改了 `os` 的偏移地址以及改变了 `loader` 引导程序中的段地址和偏移地址的赋值方式才使字符显示完整，但是用户程序依旧不知如何修改，虽然一开始想用暴力的方法输出字符，但是由于这种做法实质上并没有解决问题，所以一直寻找着其他方法，最后在同学的帮助下，我们修改了用户程序开始的偏移地址才使字符正常显示。由于这次实验涉及到的文件较多，也出现了编译时用错文件的尴尬场景，虽然不是很难，但是由于对偏移地址以及段地址的不熟悉不理解，导致出现了很多的错误，花费了很多的时间。

张稼伟的实验总结：

这次的实验过程可以说是一波三折。这次的实验内容是实现中断和系统调用，有时钟中断，键盘中断和自己编写的 4 个系统调用。一开始我们选择先完成时钟中断和键盘中断。作为小白我们依然选择先将王师兄的代码移植过来再做修改。但是没想到加入这两个中断之后用户程序的字符显示不全了，而且时钟中断和键盘中断都出现了乱码。我们调试了一个晚上都没有得出结果。后来我们四处求教，有些人说他们是在用户程序直接暴力再输出字符的，然而这种方法并非从本质上解决了问题，我们没有采纳。后来看到微信群中大家说是段地址偏移量的问题，于是我们修改了内核和引导程序的段地址和偏移量以及用户程序中的偏移地址才解决了问题。接着我试着加入第 5 个用户程序去测试 4 个中断的内容，然

而问题又出现了，加进去之后虚拟机一加载直接就死机了后面发现了是我们增加了扇区数却没有修改起始扇区号导致的，归根究底还是对这些知识不够熟悉导致的。最后程序中其实还有一个小 bug，就是我们显示用户程序信息和输错命令的话，本应是在当前界面接着输出相关内容，但不知道为什么现在会先输出一段乱码，调试很久无果，最终我灵机一动，在输出内容前先执行清屏操作不就看不见乱码了吗，一试之下果然可行。这次试验总体来说是同既艰苦又快乐的，苦在 debug 上，乐在成功后。