中山大學
SUN YAT-SEN UNIVERSITY

本科生实验报告

实验课程:_____操作系统_____

实验名称:_____操作系统_____

专业名称:_____网络空间安全_____

学生姓名:_____陈浚铭_____

学生学号:_____20337021_____

实验地点:_____我的家____

实验成绩:_____

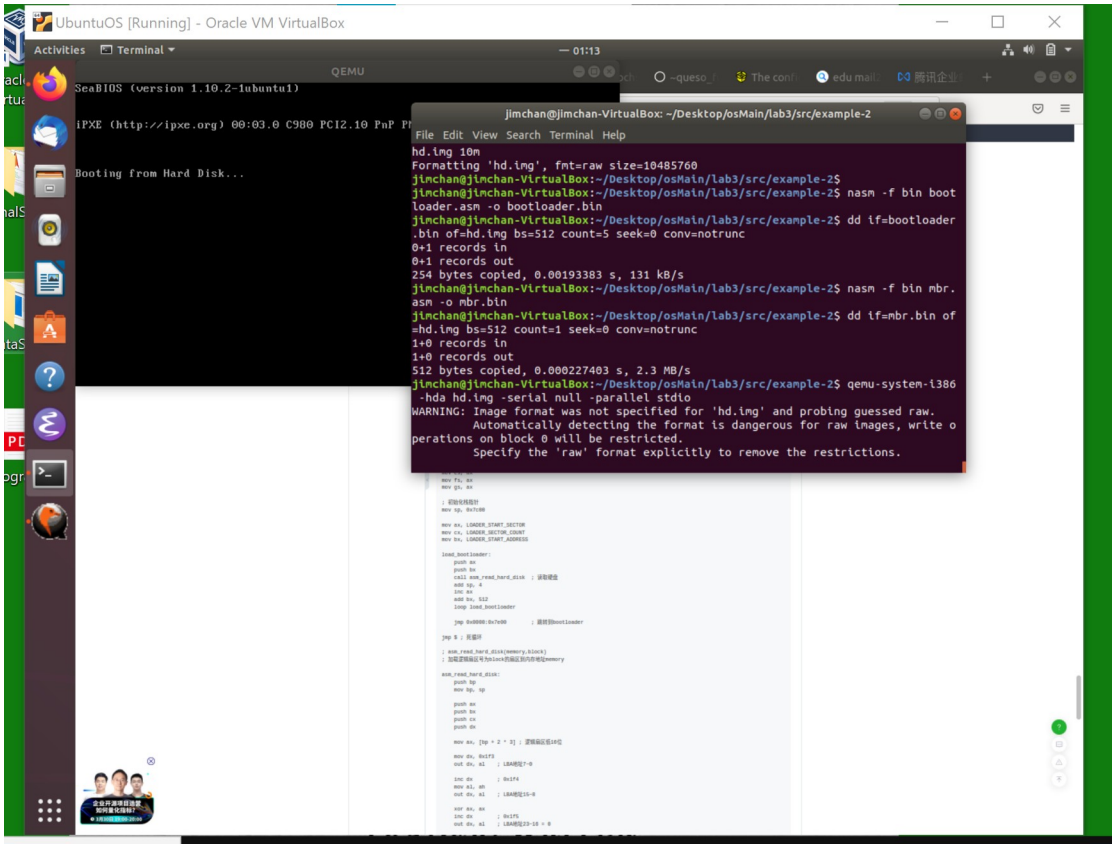
报告时间:_____

1. 实验要求

深入理解实模式和保护模式

2. 实验过程

################我在实验遇到的一个根本问题####



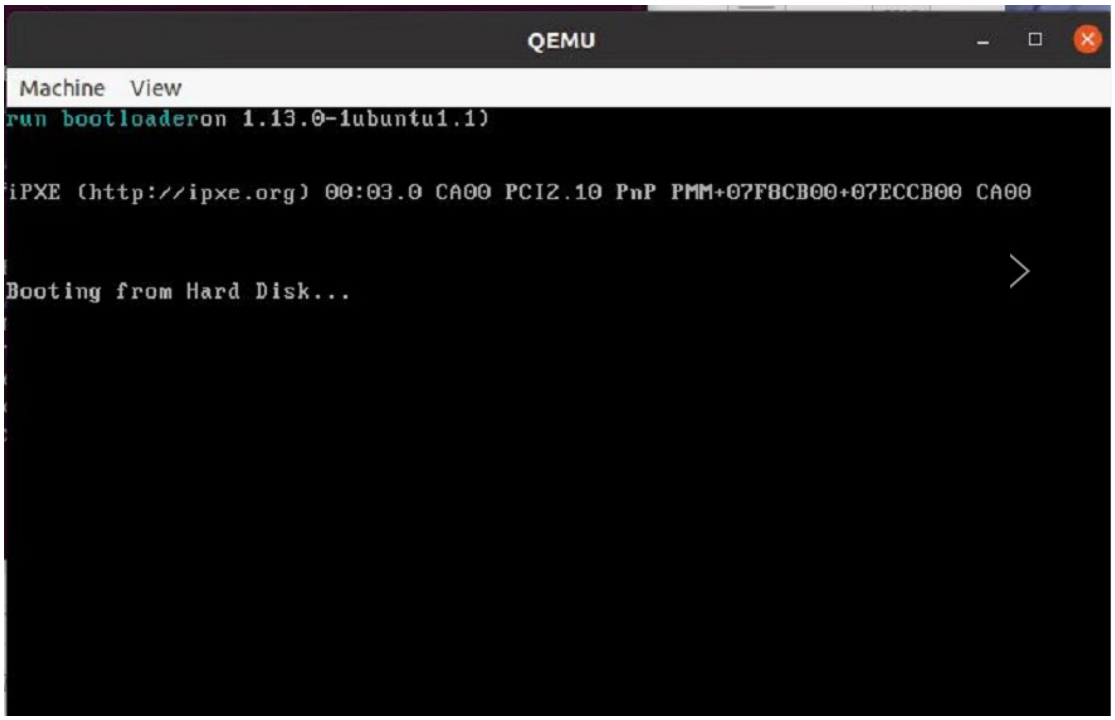
在上图，可见我运行 example-2 的代码后，并没有显示任何结果，而 example-1 的情况也是一样。这证明我的操作系统上运行 qemu 是不能够成功复现代码的。为了解决这个问题，我在实验 1，2 都使用 bochs 来取代 qemu 来实现。

实验 1：

1.1

```
jimchan@jimchan-VirtualBox:~/Desktop/osMain/lab3/src/example-1$ qemu-img create hd.img 10m
Formatting 'hd.img', fmt=raw size=10485760
jimchan@jimchan-VirtualBox:~/Desktop/osMain/lab3/src/example-1$ nasm -f bin bootloader.asm -o bootloader.bin
jimchan@jimchan-VirtualBox:~/Desktop/osMain/lab3/src/example-1$ dd if=bootloader.bin of=hd.img bs=512 count=5 seek=1 conv=notrunc
0+1 records in
0+1 records out
52 bytes copied, 0.000191163 s, 272 kB/s
jimchan@jimchan-VirtualBox:~/Desktop/osMain/lab3/src/example-1$ nasm -f bin mbr.asm -o mbr.bin
jimchan@jimchan-VirtualBox:~/Desktop/osMain/lab3/src/example-1$ nasm -f bin mbr.asm -o mbr.bin
jimchan@jimchan-VirtualBox:~/Desktop/osMain/lab3/src/example-1$ dd if=mbr.bin of=hd.img bs=512 count=5 seek=1 conv=notrunc
1+0 records in
1+0 records out
512 bytes copied, 0.000228976 s, 2.2 MB/s
jimchan@jimchan-VirtualBox:~/Desktop/osMain/lab3/src/example-1$ qemu-system-i386 -hda hd.img -serial null -parallel stdio
WARNING: Image format was not specified for 'hd.img' and probing guessed raw.
         Automatically detecting the format is dangerous for raw images, write operations on block 0 will be restricted.
         Specify the 'raw' format explicitly to remove the restrictions.
```

结果：



解释：

我们在这里分析一下代码：

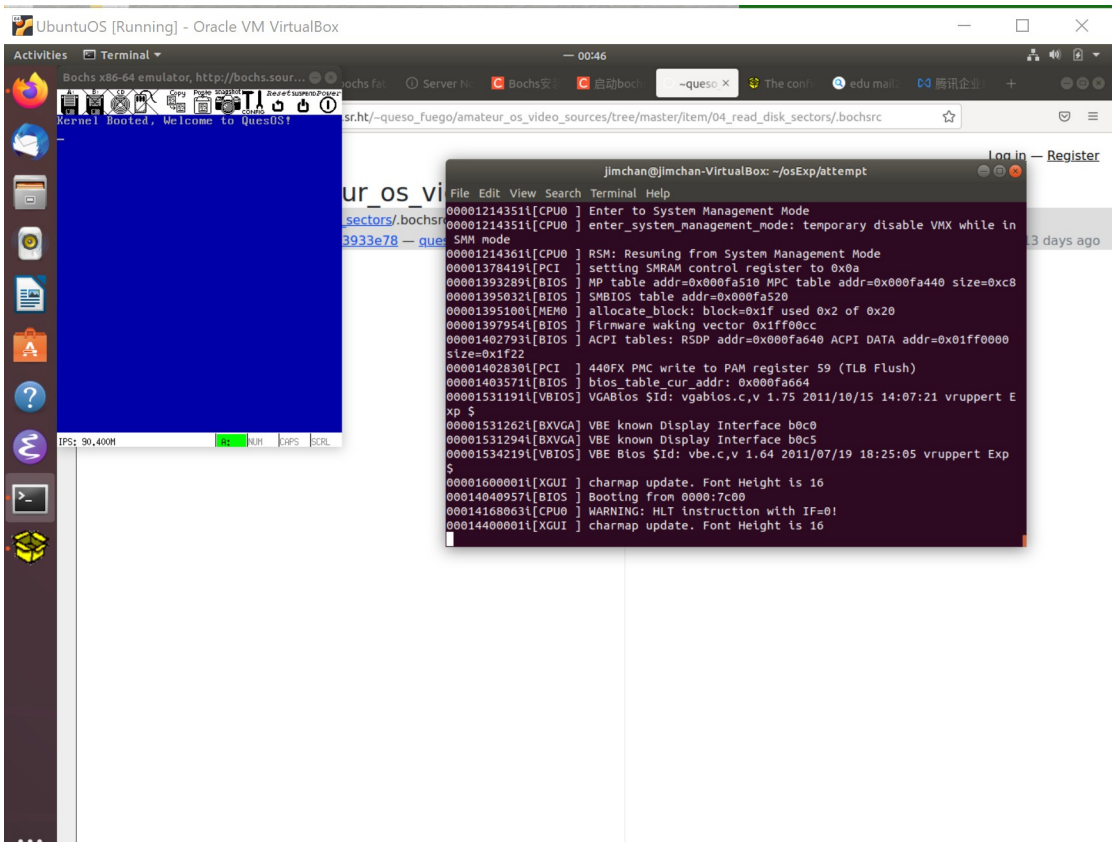
```
asm_read_hard_disk:
; 从硬盘读取一个逻辑扇区

; 参数列表
; ax=逻辑扇区号0~15位
; cx=逻辑扇区号16~28位
; ds:bx=读取出的数据放入地址

; 返回值
; bx=bx+512

    mov dx, 0x1f3
    out dx, al    ; LBA地址7~0

    inc dx        ; 0x1f4
    mov al, ah
    out dx, al    ; LBA地址15~8

    mov ax, cx

    inc dx        ; 0x1f5
    out dx, al    ; LBA地址23~16

    inc dx        ; 0x1f6
    mov al, ah
    and al, 0x0f
    or al, 0xe0   ; LBA地址27~24
    out dx, al
```

这里的代码的功能是把 ax (0-15 位的逻辑扇区的编号), 和 cx(16-28 位的逻辑扇区的编号), 分别读入到 0x1f3 – 0x1f6 的地址， 因此能够初始化逻辑扇区的编号。

```
    mov dx, 0x1f2
    mov al, 1
    out dx, al   ; 读取1个扇区

    mov dx, 0x1f7    ; 0x1f7
    mov al, 0x20     ;读命令
    out dx,al

    ; 等待处理其他操作
  .waits:
    in al, dx        ; dx = 0x1f7
    and al,0x88
    cmp al,0x08
    jnz .waits


    ; 读取512字节到地址ds:bx
    mov cx, 256   ; 每次读取一个字，2个字节，因此读取256次即可
    mov dx, 0x1f0
  .readw:
    in ax, dx
    mov [bx], ax
    add bx, 2
    loop .readw

    ret
```

之后，我们需要判断有没有其他的操作正在运行而需要等待，才能够进行读取。这里就通过 .wait 的循环来执行， 具体来说， 这里透过在 0x1f7 读入硬盘的状态值， 从而判断是否能够在处理其他操作。
最后， 我们就能够从硬盘读取数据了。在.readw 循环里， 注意到我们每一次读取为一个字（16 位）， 2x 字节（8 位），因此需要读取 256 次，这是因为 MBR 的大小为 512 字节。这样我们就能够把 MBR 加载到内存里面运行。

1.2

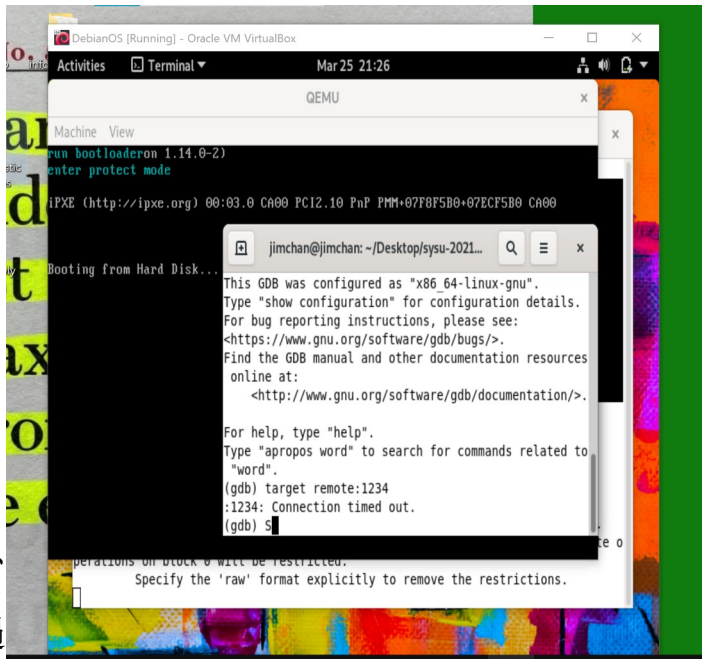


基本上， 这里跟上面一样， 使用 mbr.asm 把 bootloader.asm 的代码加载到内存运行。 但是在这里，使用了 C/H/S 的方法加载， 也就是通过 Cylinder, Head, Sector 这三个数来提 bootloader 在硬盘的位置。

LBA 到 CHS 的方程：

```
 CYL = LBA / (HPC * SPT)
TEMP = LBA % (HPC * SPT)
HEAD = TEMP / SPT
SECT = TEMP % SPT + 1
```

实验 2 和实验 3：



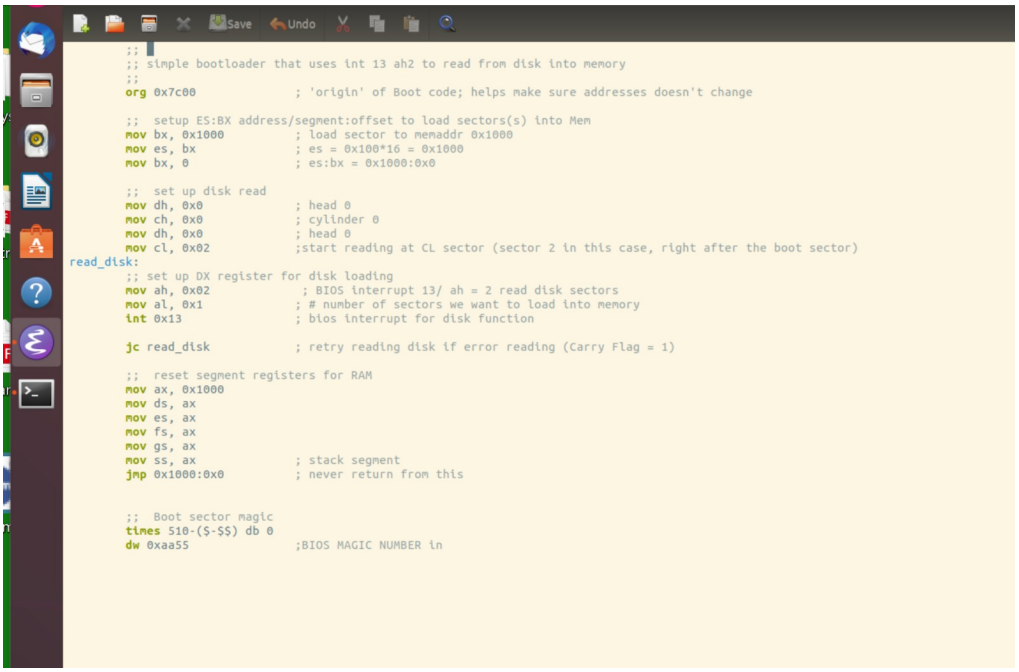
在这里可以看到我所遇到的问题： 我的 gdb 无法链接到程序来进行调试，所以无法成功实现实验。实验 3 同样原因不能做。

我会在之后补交这些实验， 拜托助教记录以下，感谢！！！！！

3. 关键代码

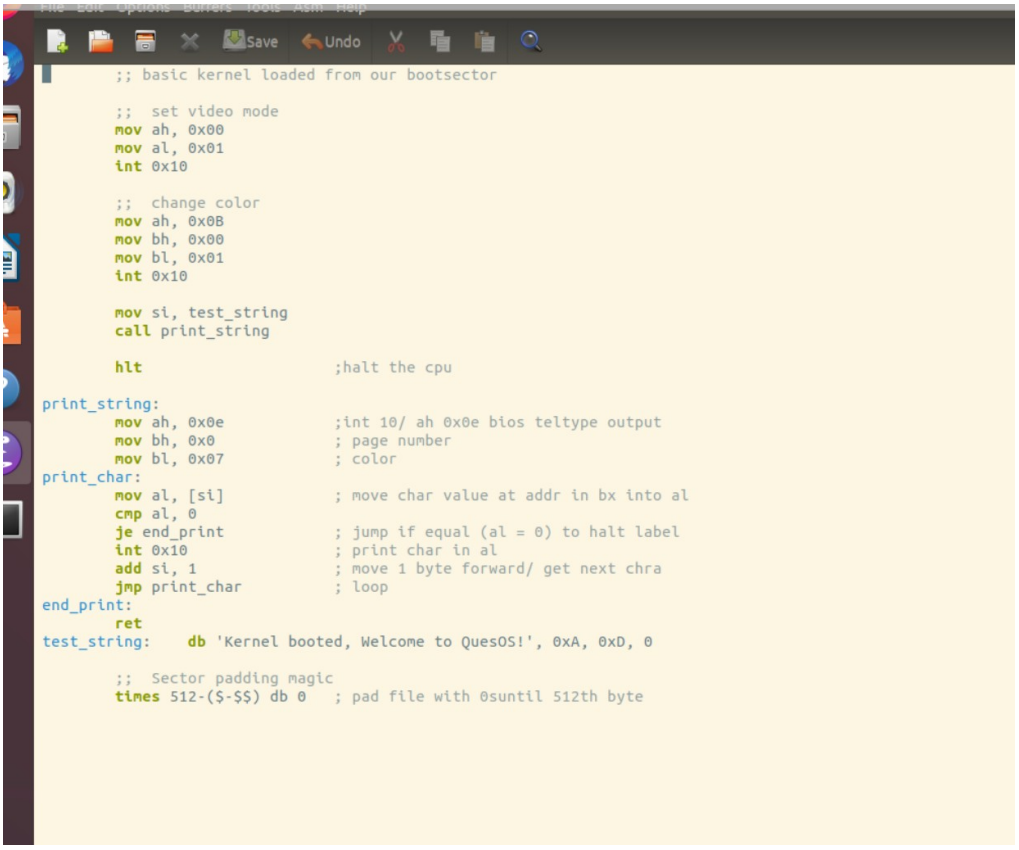
实验 1.2 的关键代码：

bootSect.asm



```
;;
;; simple bootloader that uses int 13 ah2 to read from disk into memory
;;
org 0x7c00              ; 'origin' of Boot code; helps make sure addresses doesn't change

;;  setup ES:BX address/segment:offset to load sectors(s) into Mem
mov bx, 0x1000          ; load sector to memaddr 0x1000
mov es, bx              ; es = 0x100*16 = 0x1000
mov bx, 0               ; es:bx = 0x1000:0x0

;;  set up disk read
mov dh, 0x0             ; head 0
mov ch, 0x0             ; cylinder 0
mov dh, 0x0             ; head 0
mov cl, 0x02            ;start reading at CL sector (sector 2 in this case, right after the boot sector)
read_disk:
;; set up DX register for disk loading
mov ah, 0x02             ; BIOS interrupt 13/ ah = 2 read disk sectors
mov al, 0x1             ; # number of sectors we want to load into memory
int 0x13                ; bios interrupt for disk function

jc read_disk            ; retry reading disk if error reading (Carry Flag = 1)

;;  reset segment registers for RAM
mov ax, 0x1000
mov ds, ax
mov es, ax
mov fs, ax
mov gs, ax
mov ss, ax              ; stack segment
jmp 0x1000:0x0          ; never return from this


;;  Boot sector magic
times 510-($-$$) db 0
dw 0xaa55               ;BIOS MAGIC NUMBER in
```

Kernel.asm



```
;; basic kernel loaded from our bootsector

;;  set video mode
mov ah, 0x00
mov al, 0x01
int 0x10

;;  change color
mov ah, 0x0B
mov bh, 0x00
mov bl, 0x01
int 0x10

mov si, test_string
call print_string

hlt                     ;halt the cpu

print_string:
mov ah, 0x0e            ;int 10/ ah 0x0e bios teltype output
mov bh, 0x0             ; page number
mov bl, 0x07            ; color
print_char:
mov al, [si]            ; move char value at addr in bx into al
cmp al, 0
je end_print            ; jump if equal (al = 0) to halt label
int 0x10                ; print char in al
add si, 1               ; move 1 byte forward/ get next chra
jmp print_char          ; loop
end_print:
ret
test_string:    db 'Kernel booted, Welcome to QuesOS!', 0xA, 0xD, 0

;;  Sector padding magic
times 512-($-$$) db 0   ; pad file with 0suntil 512th byte
```

4. 总结

在技术上遇到很多问题，而在解决问题中（实验 1.2）得到丰富经验