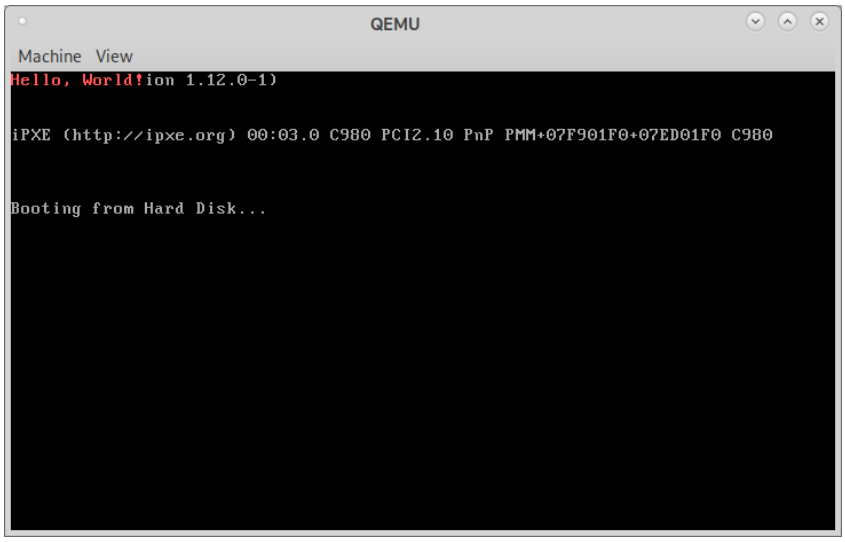
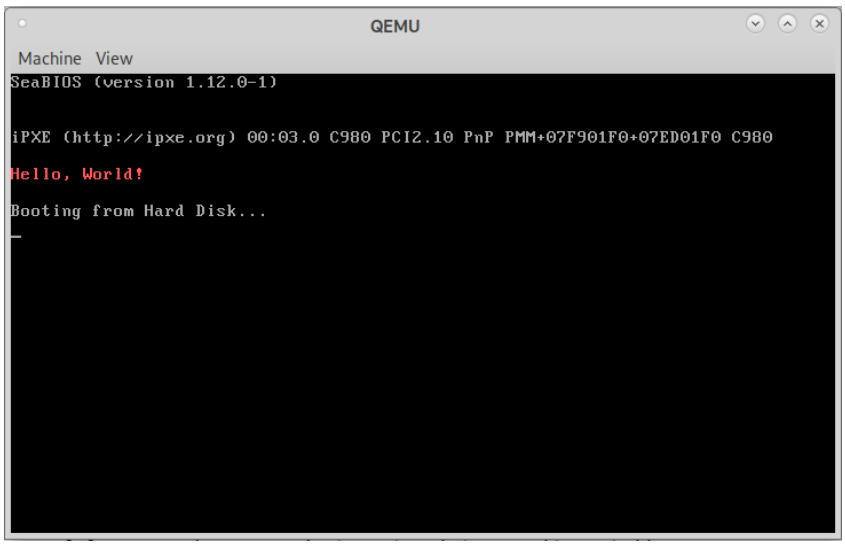
姓名：张涵之　　学号：191220154　　邮箱：1683762615@qq.com

实验进度：我完成了所有内容

实验结果：下图分别为实模式和保护模式下的输出效果





实验修改的代码位置：

实模式：仿照第一节实验课提供的参考代码实现

```
/* Real Mode Hello World */
.code16

.global start
start:
    movw %cs, %ax
    movw %ax, %ds
    movw %ax, %es
    movw %ax, %ss
    movw $0x7d00, %ax
    movw %ax, %sp # setting stack pointer to 0x7d00
    pushw $13 # pushing the size to print into stack
    pushw $message # pushing the address of message into stack
    callw displayStr # calling the display function
```

实模式下完成了 displayStr 函数入口参数的准备和函数调用

```
loop:
    jmp loop

message:
   .string "Hello, World!\n\0"

displayStr:
   pushw %bp
   movw 4(%esp), %ax
   movw %ax, %bp
   movw 6(%esp), %cx
   movw $0x1301, %ax
   movw $0x000c, %bx
   movw $0x0000, %dx
   int $0x10
   popw %bp
   ret
```

观察可知 displayStr 通过 int $0x10 系统调用实现打印字符串
循环的作用为完成打印后保持窗口不关闭供用户观看

保护模式下根据 PPT 介绍可知需要完成的步骤依次为：
在 start.s 文件中实现实模式到保护模式的切换：
1）通过 cli 指令实现关闭中断
2）打开 A20 地址线（此处通过网络搜索“什么是地址线”，“如何使用汇编指令打开 A20 地址线”，并阅读了几篇介绍 80286 和 A20 地址线的相关博文）；
3）通过 lgdt 指令加载 GDTR 寄存器（ICS 课程中实现过相关指令）；
4）设置 CR0 寄存器的 PE 位（第 0 位）为 1（表示进入保护模式）；
5）长跳转进入保护模式代码；

```
    #TODO: Protected Mode Here
    cli # closing interuption
    inb $0x92, %al # setting up A20 bus
    orb $0x02, %al
    outb %al, $0x92
    data32 addr32 lgdt gdtDesc # loading GDTR
    movl %cr0, %eax
    orb $0x01, %al
    movl %eax, %cr0 # setting CR0 to switch to protected mode
    data32 ljmp $0x08, $start32 # long jump to execute start32
```

6）初始化 DS，SS，ES，FS，GS 这些段寄存器初始化 ESP（这部分框架已提供）；
7）设置 GDT 表项，其中代码段与数据段基址都为 0x0，视频段基址为 0xb8000；

```
gdt:
    #GDT definition here
    .word 0,0
    .byte 0,0,0,0

    .word 0xffff,0
    .byte 0,0x9a,0xcf,0

    .word 0xffff,0
    .byte 0,0x92,0xcf,0

    .word 0xffff,0x8000
    .byte 0x0b,0x92,0xcf,0

gdtDesc:
    #gdtDesc definition here
    .word (gdtDesc - gdt - 1)
    .long gdt
```

8）保护模式下中断关闭，无法通过陷入磁盘中断调用 BIOS 进行磁盘读取，代码框架中实现的 readSec(void *dst, int offset)接口通过读写磁盘的相应端口来实现特定扇区的读取
在 bootMain 函数中通过上述接口读取磁盘 MBR 之后扇区中的程序至内存的特定位置（注意到代码框架 app/Makefile 中设置的该 Hello World 程序入口地址为 0x8c00，则该位置的地址即为 0x8c00，通过 elf 函数指针的调用跳转执行，便实现了保护模式下的输出

```
app.bin: app.s
    gcc -c -m32 app.s -o app.o
    ld -m elf_i386 -e start -Ttext 0x8c00 app.o -o app.elf
    objcopy -S -j .text -O binary app.elf app.bin
```

```
void bootMain(void) {
    void (*elf)(void);
    elf = (void(*)(void))0x8c00;
    readSect((void*)elf, 1); // loading sector 1 to 0x8c00
    elf();
}
```

思考和总结:
本次实验帮助我更好地理解了 ICS 中对实模式、保护模式、中断和函数调用的介绍。