

2018 OS lab1 实验报告

姓名：王志邦
学号：161220131
日期：18.3.25

一、 实验目的

本实验通过实现一个简单的引导程序，了解系统启动的基本过程，包括如何从实模式进入保护模式，并从磁盘中加载程序并运行。

二、 实验内容

从实模式切换至保护模式，在保护模式下读取磁盘 1 号扇区中的 Hello World 程序至内存中的相应位置，跳转执行该 Hello World 程序，并在终端中打印 Hello, World!

三、 分步实现

a) 从实模式进入保护模式

8086 的时候仅有实模式，但是实模式下，物理地址对程序员是可见的，因此无法支持多任务，而且程序的安全性无法得到保证。

通过虚拟存储，分页机制等，是系统可以支持多道任务，并且提高了程序执行的安全度。

在我们的引导程序中，在 start.s 中模拟了从实模式切换到保护模式的过程。

```
cli                                #关闭中断
inb $0x92, %al                    #启动A20总线
orb $0x02, %al
outb %al, $0x92
data32 addr32 lgdt gdtDesc        #加载GDTR
movl %cr0, %eax                   #启动保护模式
orb $0x01, %al
movl %eax, %cr0
data32 ljmp $0x08, $start32      #长跳转切换至保护模式
```

实模式下首先关闭终端，将 IF 寄存器清零，启动 A20 总线，加载 GDTR，然后启动保护模式，long jmp 中设置 CS 寄存器，开始保护模式的运行。

b) 保护模式下的准备阶段

进入保护模式，然后就是需要初始化 6 个段寄存器和 esp 指针，然后才能够执行功能。

```
# Set up the protected-mode data segment registers
movw    $0x10, %ax
movw    %ax, %ds      # %DS = %AX
movw    %ax, %es      # %ES = %AX
movw    %ax, %ss      # %SS = %AX
movw    %ax, %fs      # %FS = %AX
movw    $0x18, %ax
movw    %ax, %gs      # %GS = %AX

# Set up a stack for C code.
movl    $0, %ebp
movl    $(128 << 20), %esp
sub     $16, %esp      #初始化DS ES FS GS SS 初始化栈顶指针ESP
jmp     bootMain       #跳转至bootMain函数 定义于boot.c
```

通过上学期 pa 的学习，利用 movw sreg, r/m16 可以初始化段寄存器。

```

gdt:
    .word 0,0                                #GDT第一个表项必须为空
    .byte 0,0,0,0

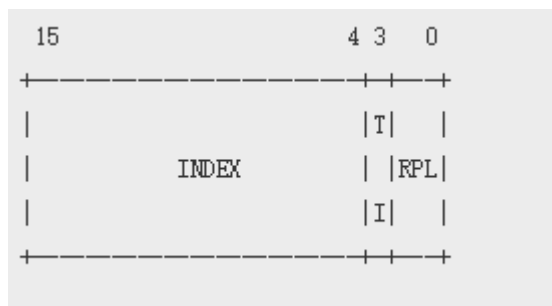
    .word 0xffff,0                            #代码段描述符
    .byte 0,0x9a,0xcf,0

    .word 0xffff,0                            #数据段描述符
    .byte 0,0x92,0xcf,0

    .word 0xffff,0x8000                       #视频段描述符
    .byte 0x0b,0x92,0xcf,0

```

根据提供的 gdt 表，数据段的描述符为表的第 2 项，而视频段的描述符为表的第三项，由于段寄存器的结构：



Index 为 15 到 4，所以在给段寄存器赋值时应该根据对应描述符在表中的位置对段寄存器进行赋值，比如 ds 的 visible portion 应为 0x0010，而 gs 的 visible portion 为 0x18

c) 从磁盘中加载用户程序

由于中断关闭，无法通过陷入磁盘中断调用 BIOS 进行磁盘读取，但是框架代码中提供了 readSect 函数，通过 in/out 指令通过对应端口对磁盘进行读取，根据提示 loading sector 1 to memory，调用 readSect 函数进行加载。

```

void bootMain(void) {
    void (*elf)(void);
    // Loading sector 1 to memory
    readSect(elf, 1);
    elf();
}

```

d) Hello 程序的编写——写显存的应用

```

movl $((80*9+0)*2), %edi      #在第9行第0列打印
movb $0x0c, %ah               #黑底红字
movb $72, %al                 #72为H的ASCII码
movw %ax, %gs:(%edi)          #写显存

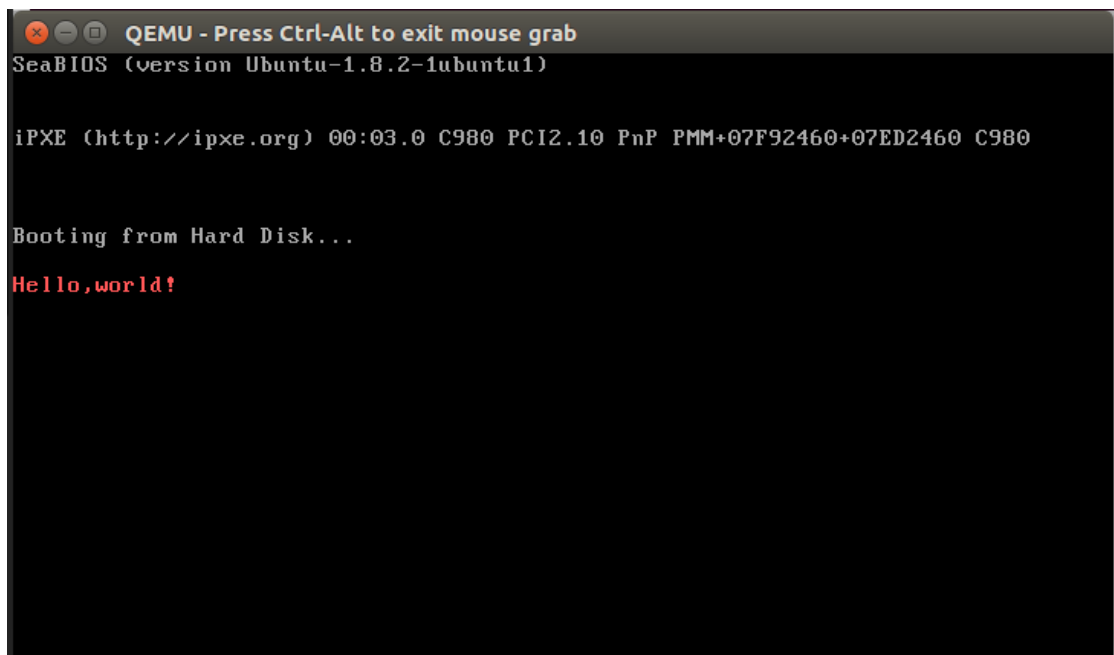
```

```

movl $((80*9+11)*2), %edi     #在第9行第11列打印
movb $33, %al                 #33为!的ASCII码
movw %ax, %gs:(%edi)          #写显存
movl $((80*9+12)*2), %edi     #在第9行第12列打印
movb $00, %al                 #00为\0的ASCII码
movw %ax, %gs:(%edi)          #写显存
ret

```

四、 实验结果



```

QEMU - Press Ctrl-Alt to exit mouse grab
SeaBIOS (version Ubuntu-1.8.2-1ubuntu1)

iPXE (http://ipxe.org) 00:03.0 C980 PCI2.10 PnP PMM+07F92460+07ED2460 C980

Booting from Hard Disk...
Hello,world!

```

五、 遇到的问题

栈顶指针寄存器 esp 的初始化的方法，根据 pa 的代码得到的，并未理解这样初始化的原因。

在 app.s 的末尾如果不加 ret 或者 loop : jmp loop 就会屏幕不断闪烁，但是并未理解闪烁的原因，并且不知道 ret 或者写一个死循环，二者究竟是哪一个的实现比较好。