Lab1 系统引导

141180016 丁俊峰

一、实验步骤

1.系统启动

系统启动时，工作在实模式的 BIOS 程序将主引导扇区加载至内存 0x7c00 处（被加载的程序一般称为 Bootloader），紧接着执行一条跳转指令，将 CS 设置为 0x0000，IP 设置为 0x7c00，运行被装入的 Bootloader。

代码：

```
#编译 start.s 和 boot.c
gcc -c -m32 start.s -o start.o
gcc -c -m32 -O1 -fno-stack-protector boot.c -o boot.o

#将 start.o 和 boot.o 链接为 bootloader 后放入内存 0x7c00 处
ld -m elf_i386 -e start -Ttext 0x7c00 start.o boot.o -o bootloader.elf
```

2.实模式切换保护模式

关闭中断，打开 A20 数据总线，加载 GDTR，设置 CR0 的 PE 位（第 0 位）为 1b，通过长跳转设置 CS 进入保护模式，初始化 DS，ES，SS。

代码：

```
#初始化 DS ES SS
xor    %ax,%ax
movw %ax,%ds
movw %ax,%es
movw %ax,%ss

Cli                                    #关中断
```

```
inb $0x92, %al                  #启动 A20 总线
orb $0x02, %al
outb %al, $0x92
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
```

下面设置了三个 GDT 表项，其中代码段与数据段的基地址都为 0x0，视频段的基地址为 0xb8000，为 32 位模式下的寻址提供基础。

代码：

```
.p2align 2
gdt:
    .word 0,0                   # GDT 第一个表项必须为空
    .byte 0,0,0,0
LABEL_DESC_CODE:
    .word 0xffff,0              #代码段描述符
    .byte 0,0x9a,0xcf,0
LABEL_DESC_DATA:
    .word 0xffff,0              #数据段描述符
    .byte 0,0x92,0xcf,0
LABEL_DESC_VIDEO:
    .word 0xffff,0x8000         #视频段描述符
    .byte 0x0b,0x92,0xcf,0

gdtDesc:
    .word (gdtDesc - gdt -1)
    .long gdt

SelectorCode=LABEL_DESC_CODE-gdt
SelectorData=LABEL_DESC_DATA-gdt
SelectorVideo=LABEL_DESC_VIDEO-gdt
```

3.保护模式下打印字符

由于中断关闭，保护模式下采用写显存方式打印字符。

代码：

```
mov   $SelectorVideo,%ax          #将视频段选择子赋值给 gs
mov   %ax,%gs
movl $((80*5+0)*2), %edi          #在第 5 行第 0 列打印
movb $0x0c, %ah                   #黑底红字
movw $message, %ax                 #42 为 H 的 ASCII 码
movw %ax, %gs:(%edi)               #写显存
```

二、实验结果

由于对于 Makefile 中的 app.s 放置在内存还是磁盘中以及如何利用 readSect 函数读取 app.s 存有疑惑，没能完成在保护模式下读取磁盘程序的任务，只能使用 bootloader 切换保护模式并在保护模式下显示字符，效果如图所示：