

lab1 实验报告

161220124 王明

1.1 在实模式下实现一个 Hello World!程序

观察提供的代码框架，发现程序执行时首先进入的是 bootLoader 文件夹中的 start.s 中的 code16 的部分，这是刚刚开机时 BIOS 实模式下的首先执行部分。于是就相应地使用屏幕中断的方法来实现打印，具体的参数参考的是助教提供的 BIOS 中断表，代码如下：

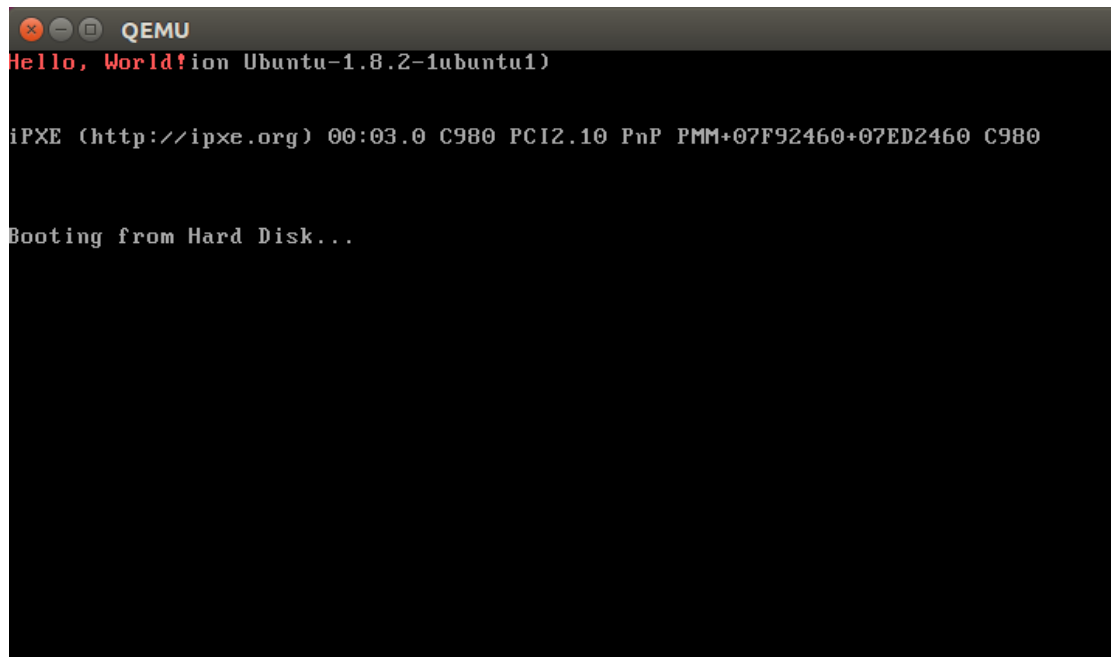
```
.code16

.global start
start:
|
movw $message, %ax
movw %ax, %bp
movw $13, %cx
movw $0x1301, %ax
movw $0x000c, %bx
movw $0x0000, %dx
int $0x10

message:
.string "Hello, World!"
```

这时候还需在 app.s 里随便加一条指令，不能为空，否则编译无法通过。

运行结果如下：



1.2.在保护模式下实现一个 Hello World 程序

先实现实模式向保护模式的切换，即在 start.s 的 code16 中完成关闭中断、开启 A20 总线、加载 GDTR，将 cr0 寄存器的 PE 位由 0 变成 1、再用 ljmp 指令跳转至保护模式这些操作完成，这部分代码使用的是助教给的示例代码，代码如下：

```
.global start
start:

    cli
    inb $0x92, %al
    orb $0x02, %al
    outb %al, $0x92
    data32 addr32 lgdt gdtDesc
    movl %cr0, %eax
    orb $0x01, %al
    movl %eax, %cr0
    data32 ljmp $0x08, $start32
```

进入保护模式后，首先要做的就是设置段寄存器，根据 386 的规定，16 位的段寄存器的前 13 位是该段寄存器的编号，第 1 个是 CS，那么本实验用到的 DS 和 GS 自然是第 2 个和第 3 个，因此分别赋值 0x0010 和 0x0018，然后设置段的初始栈顶 %esp，这个值不能过大，否则会发生越界，代码如下：

```
start32:

    movw $0x0010, %ax    #DS
    movw %ax, %ds

    movw $0x0018, %ax    #GS
    movw %ax, %gs

    movl $0x2000, %esp

    jmp bootMain
```

再接下来就是填充段描述符表，此部分参照了助教给的示例代码，如下：

```
gdt:

    .word 0,0            #EMPTY
    .byte 0,0,0,0

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

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

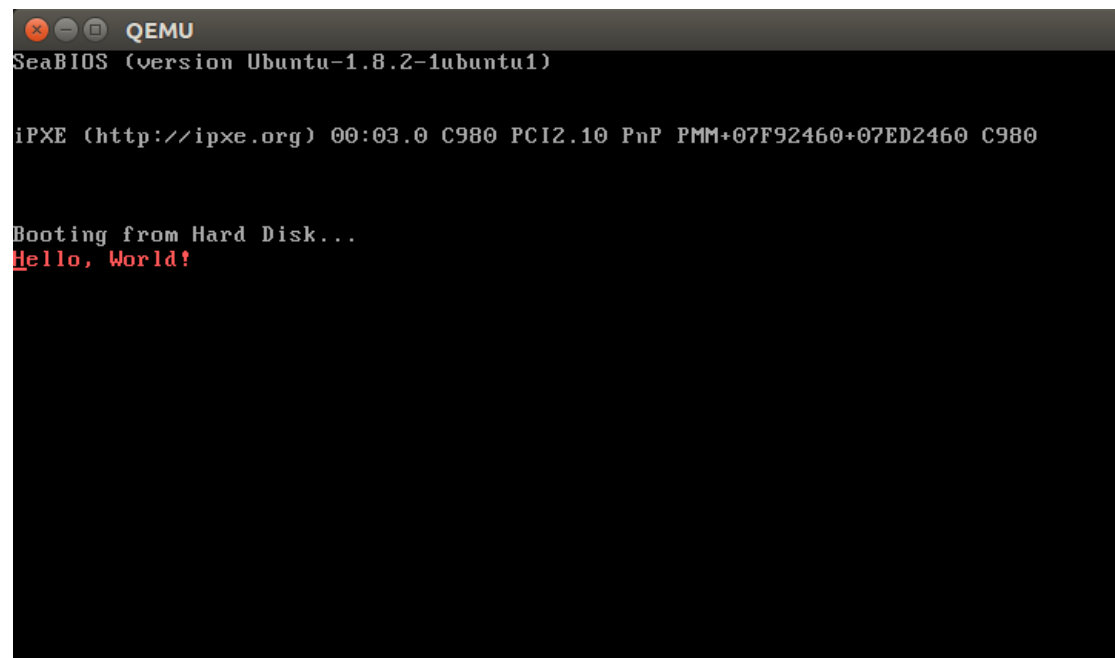
    .word 0xffff,0x8000   #GS
    .byte 0x0b,0x92,0xcf,0
```

此时保护模式才算设置完成。接下来就是实现 hello world 程序了。

我一开始以为是要进入到 bootMain 函数里使用 printf 进行打印，后来发现只需要在 start.s 的 code32 里直接使用显存输出的方式就可以了，代码如下：

```
#jmp bootMain
movl $((80*8+0)*2), %edi
movb $0x0c, %ah
movb $72, %al
movw %ax, %gs:(%edi)
addl $2, %edi
movb $101, %al
movw %ax, %gs:(%edi)
addl $2, %edi
movb $108, %al
movw %ax, %gs:(%edi)
addl $2, %edi
movb $108, %al
movw %ax, %gs:(%edi)
addl $2, %edi
movb $111, %al
movw %ax, %gs:(%edi)
addl $2, %edi
movb $44, %al
movw %ax, %gs:(%edi)
addl $2, %edi
movb $32, %al
movw %ax, %gs:(%edi)
```

运行结果如下：



```
QEMU
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!
```

1.3. 在保护模式下加载磁盘中的 Hello World 程序运行

在 1.2 中已经说明了如何从实模式切换到保护模式，在此不重复。

示例代码提供了读取磁盘的相关接口和调用函数，于是首先使用 `jmp bootMain` 进入到读取磁盘的函数，然后利用 `void readSect(void *dst, int offset)`，来读取磁盘的第一个扇区。注意到 `app` 文件夹中的 `Makefile` 里的首地址，如下：

```
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
```

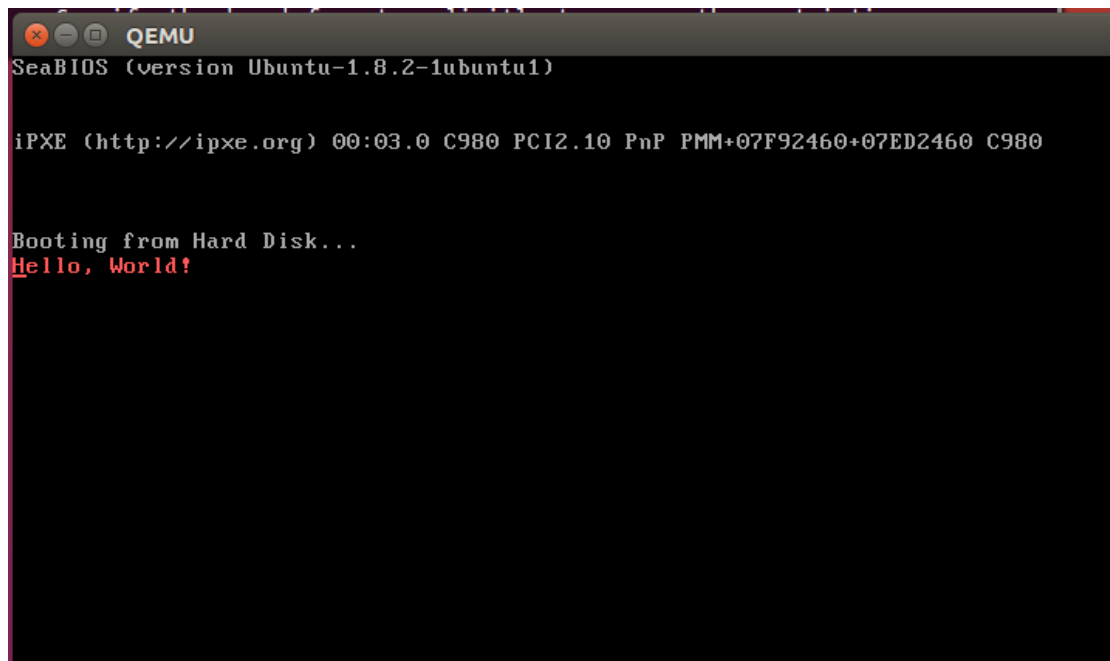
`-Ttext 0x8c00`，因此 `ReadSect` 的第一个参数 `dst` 应该为 `0x8c00`，由于是第一个扇区，所以 `offset` 设置为 1，代码如下：

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

此时加载的程序便是 `app.s` 里的用户程序，所以将用显存输出 `helloworld` 的程序放到 `app.s` 里，同时还需要设置一个死循环避免屏幕突然跳转，如下：

```
loop:
    jmp loop
```

运行结果如下：



```
QEMU
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!
```