OSLab1

191220083 蒙芷露 mzl0830@sina.com

3.1实模式hello world程序

回答问题：CPU、内存、BIOS、磁盘、主引导扇区、加载程序、 操作系统这些名词之间的关系?

CPU会在开机后到内存寻找需要执行的第一条指令。内存在刚开机时，640k到1MB的位置存储了BIOS固件，BIOS可以完成开机自检，然后BIOS会在磁盘的主引导扇区（MBR）加载到内存0x7c00的位置，并且检查末尾两个字节是否是魔数。是魔数则则BIOS跳转到内存0x7c00位置，执行加载到此处的启动代码，否则会继续寻找启动设备直到确认所有设备都不可启动而返回相关错误信息。启动代码会将操作系统的代码和数据从磁盘装载入内存，并跳转到操作系统的首地址，此后操作系统开始工作。

3.1已经在index的指导下完成，为了实现3.2，需要看懂3.1中的一些操作：

mbr.s 是一段在实模式下运行的AT&T汇编代码，在借用%ax完成%ds %ss %es置成与%cs一样的初始化操作之后，%ax设置为栈顶指针，并且把字符串长度和message压入栈中作为函数displayStr的参数（传参） 之后进入循环

message中保存了string型的数据，是要输出的内容；

displayStr部分是函数的汇编代码，先进行保存调用者状态下的栈顶指针，然后对保存一些参数，调用软中断指令int $0x10 在实模式下直接对屏幕进行输出

编译链接过后得到一个可执行文件，在缩减大小后，利用已经编写好的genboot.pl文件把mbr.bin制作成为MBR,genboot.pl文件应当对其他.bin文件也有一样的功能 之后编写、编译链接之后可以复用。

3.2 实模式切换保护模式

填写start.s

cli就可以完成关中断

A20的开启参考了网络上的博客 有多种实现 主要都是通过和端口的交互完成。

开启保护模式：在苦恼如何只设置CR0的PE位，发现只需要设置对应的特定bit就可以了，不需要特意找到表示的对应符号。实际实现不同于ICS的实验。

GDT表的初始化：参考给出的段结构符的描述和属性，以及网上搜到的各段寄存器代表的权限完成填写。根据各部分的信息进行了分段填写 大致如下：

```
.word 0x0000
.word 0x0000
.word 0x0000
.word 0x0000
 #empty

.word 0xffff #15~0
.word 0x0000 #32~16
.byte 0x00 #39~32
.byte 0x9a #47~40
.word 0x00cf #63~48
 #cs

.word 0xffff #15~0
.word 0x0000 #32~16
.byte 0x00 #39~32
.byte 0x92 #47~40
.word 0x00cf #63~48
#ds

.word 0xffff #15~0
.word 0x8000 #32~16
.byte 0x0b #39~32
.byte 0x92 #47~40
.word 0x00cf #63~48
#tss
```

> 注：视频段是数据；limit应该是全1
>
> 小心大小端，一开始填反了

实模式下段描述符高速缓冲寄存器的内容	段寄存器	段基地址	段界限(固定)	存在性	特权级	已存取	粒度	扩展方向	可读性	可写性	可执行	堆栈大小	一致特权
	CS	当前CS*16	0000FFFFH	Y	0	Y	B	U	Y	Y	Y	-	N
	SS	当前SS*16	0000FFFFH	Y	0	Y	B	U	Y	Y	N	W	-
	DS	当前DS*16	0000FFFFH	Y	0	Y	B	U	Y	Y	N	-	-
	ES	当前ES*16	0000FFFFH	Y	0	Y	B	U	Y	Y	N	-	-
	FS	当前FS*16	0000FFFFH	Y	0	Y	B	U	Y	Y	N	-	-
	GS	当前GS*16	0000FFFFH	Y	0	Y	B	U	Y	Y	N	-	-

lab手册中对conforming的描述我不太能理解 找到这样一段解释：

- Conforming bit for code selectors:
  - If **1** code in this segment can be executed from an equal or lower privilege level. For example, code in ring 3 can far-jump to *conforming* code in a ring 2 segment. The `privl`-bits represent the highest privilege level that is allowed to execute the segment. For example, code in ring 0 cannot far-jump to a conforming code segment with `privl==0x2`, while code in ring 2 and 3 can. Note that the privilege level remains the same, ie. a far-jump form ring 3 to a `privl==2`-segment remains in ring 3 after the jump.
  - If **0** code in this segment can only be executed from the ring set in `privl`.

为了填写GDT主要参考了：

https://wiki.osdev.org/GDT_Tutorial

https://wiki.osdev.org/GDT_Tutorial#What_should_I_put_in_my_GDT.3F

https://bbs.huaweicloud.com/blogs/144130

修改boot.c

主要编写bootMain函数，关键在于读取磁盘中的扇区，把程序装载到内存特定位置并跳转执行。根据app的makefile可以知道特定开始位置是 **0x8c00**

利用已给的函数读出磁盘中的内容到上述地址，再把上述地址转成函数指针然后调用

调试和运行

在保护模式下直接输出helloworld

参考了app.s装载数据和设置函数的方式，前面启动保护模式的代码不变，将jump bootMain注释后代码修改如下：

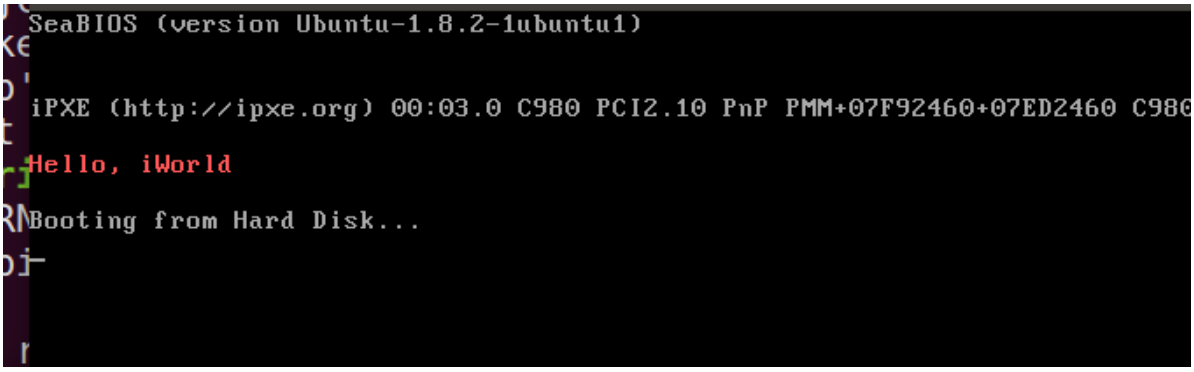
（其中为了与从磁盘加载的程序区分，对输出内容做了一些修改）

```
        movl $0x8000, %eax # setting esp
        movl %eax, %esp
        #jmp bootMain # jump to bootMain in boot.c
        pushl $13
        pushl $message
        calll displayStr
loop:
        jmp loop

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

displayStr:
        movl 4(%esp), %ebx
        movl 8(%esp), %ecx
        movl $((80*5+0)*2), %edi
        movb $0x0c, %ah
nextChar:
        movb (%ebx), %al
        movw %ax, %gs:(%edi)
        addl $2, %edi
        incl %ebx
        loopnz nextChar # loopnz decrease ecx by 1
        ret
.p2align 2
```

结果如下：



一开始以为Booting from Hard Disk是从磁盘加载应用程序的标志，后来修改直接输出的内容后发现此处应是指成功加载了导引程序，所以对输出进行微调是有必要的

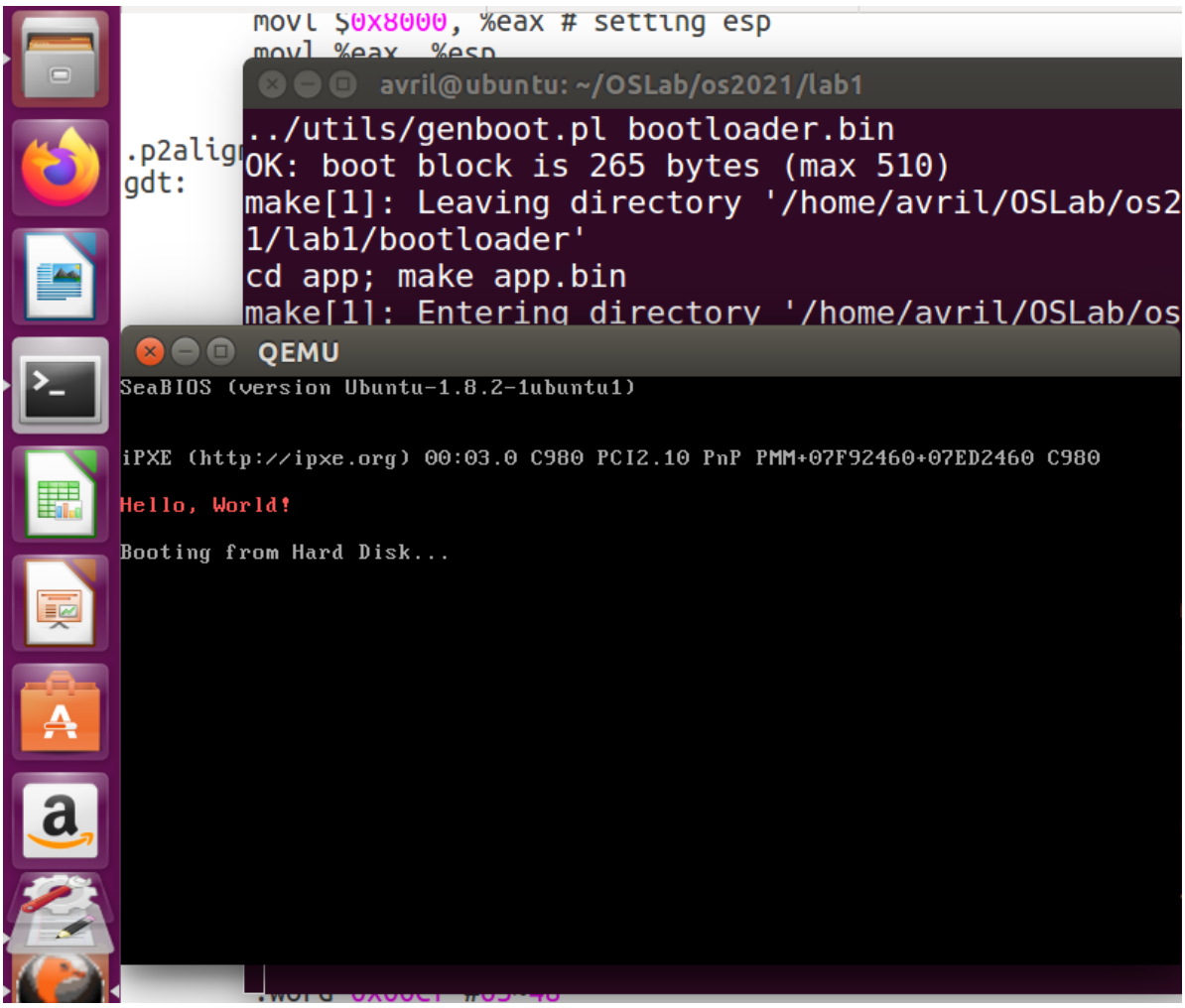
从磁盘加载程序以输出helloworld

gdb可以调试qemu 见index

app的bin在makefile中已经生成

调试：

```
b *0x7c00 #可以在编写的start.s中观察程序运行了
layout asm #查看汇编代码
```



> 最终提交的版本以磁盘加载应用程序输出为准

一点感想

说实话一开始觉得自己可能根本完不成。因为虽然知道大致的原理和实现，但是对于具体要填写什么样的数据，怎样修改代码完全没有头绪。实验手册只是大概重温了一下原理，翻看最多的部分应该是框架代码和提供的关于段寄存器的信息。Intel手册也是类似于伪代码的实现。在google的过程中发现有很多人尝试自制os，他们用不同的方式（比如内联汇编）完成了同样的功能，虽然并没有直接的实现，但是也帮助我理清了思路。

我认为的我开始有所进展的转折点应该在搜索怎样打开A20总线，查到了很多不同的方法，也看到了具体的原理解释。主要是突然意识到如何使用汇编这个工具来填写修改数据。

第二个大关是填写GDT，查了很多资料才搞明白GDT到底要我填什么，因为一开始把段寄存器和GDT搞混了。我在对着GDT的结构一个个比特确认的时候，对GDT中的信息有了更深刻的了解。同时也查到了一个对OS原理和实现有很详细解释的网站。

总体来说，ICS课程省略了很多细节的实现，让人有一种"我也懂OS"的错觉，OS的实验应了那句话：Talk is cheap,show me the code.

> 实验用时：两个晚上（19：00~00：00）约10h