

OS-lab1-report

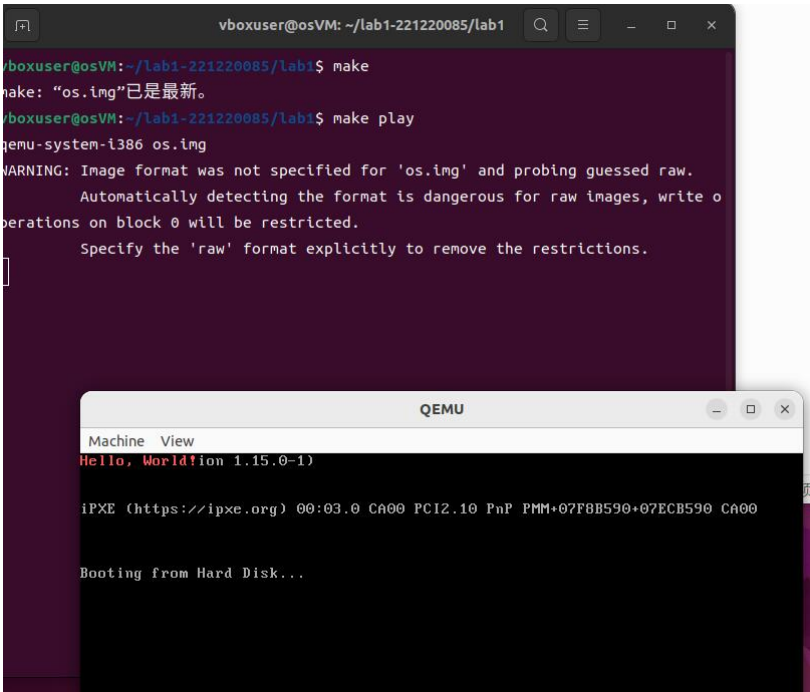
姓名	时昌军
学号	221220085
邮箱	221220085@smail.nju.edu.cn

一、实验进度

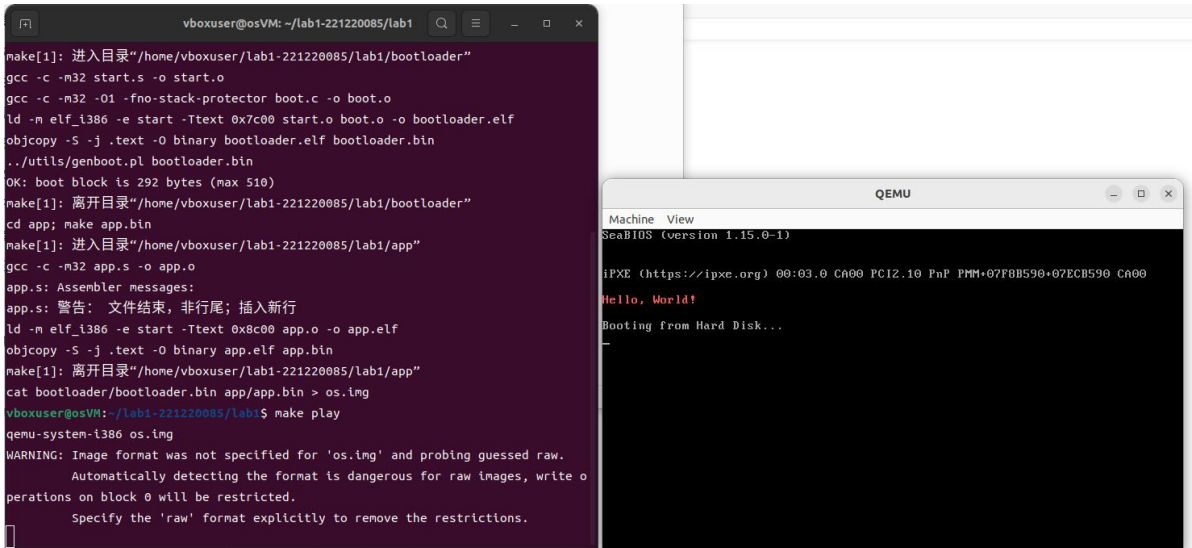
3个实验要求全部完成

二、实验结果

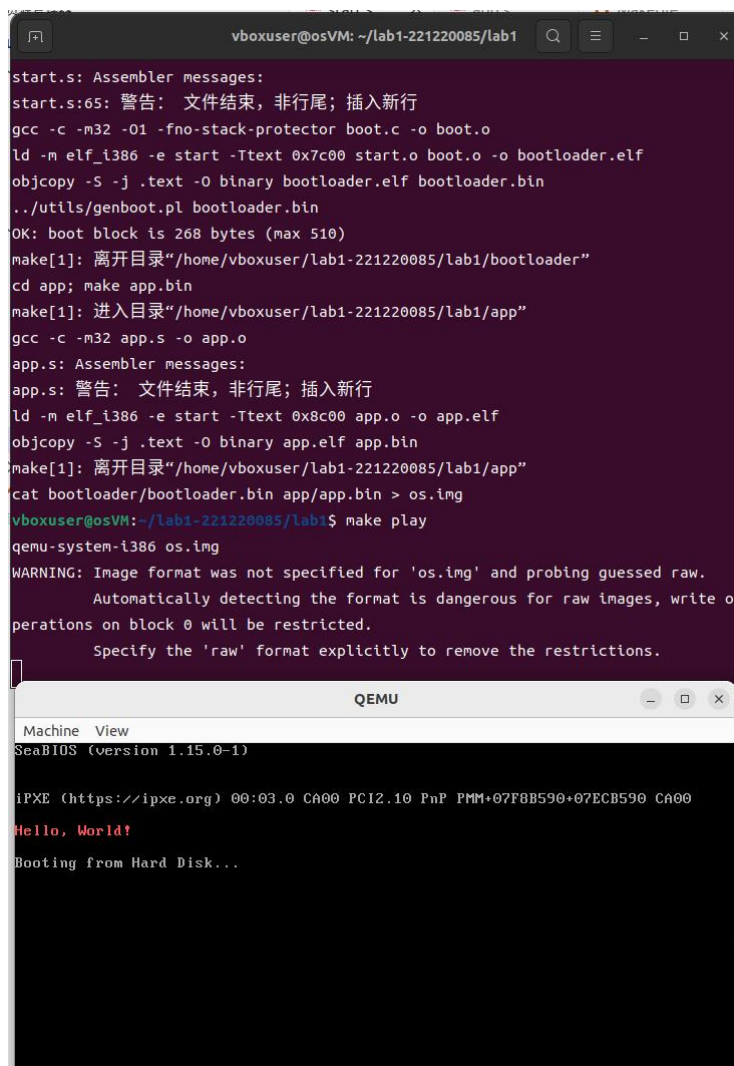
1.1. 在实模式下实现一个Hello World程序



1.2. 在保护模式下实现一个Hello World程序



1.3. 在保护模式下加载磁盘中的Hello World程序运行



```
vboxuser@osVM: ~/lab1-221220085/lab1
start.s: Assembler messages:
start.s:65: 警告: 文件结束, 非行尾; 插入新行
gcc -c -m32 -O1 -fno-stack-protector boot.c -o boot.o
ld -m elf_i386 -e start -Ttext 0x7c00 start.o boot.o -o bootloader.elf
objcopy -S -j .text -O binary bootloader.elf bootloader.bin
../utils/genboot.pl bootloader.bin
OK: boot block is 268 bytes (max 510)
make[1]: 离开目录"/home/vboxuser/lab1-221220085/lab1/bootloader"
cd app; make app.bin
make[1]: 进入目录"/home/vboxuser/lab1-221220085/lab1/app"
gcc -c -m32 app.s -o app.o
app.s: Assembler messages:
app.s: 警告: 文件结束, 非行尾; 插入新行
ld -m elf_i386 -e start -Ttext 0x8c00 app.o -o app.elf
objcopy -S -j .text -O binary app.elf app.bin
make[1]: 离开目录"/home/vboxuser/lab1-221220085/lab1/app"
cat bootloader/bootloader.bin app/app.bin > os.img
vboxuser@osVM: ~/lab1-221220085/lab1$ make play
qemu-system-i386 os.img
WARNING: Image format was not specified for 'os.img' and probing guessed raw.
Automatically detecting the format is dangerous for raw images, write o
perations on block 0 will be restricted.
Specify the 'raw' format explicitly to remove the restrictions.

QEMU
Machine View
SeaBIOS (version 1.15.0-1)

iPXE (https://ipxe.org) 00:03.0 CA00 PCI2.10 PnP PMM+07F8B590+07ECB590 CA00
Hello, World!
Booting from Hard Disk...
```

三、实验修改的代码

1.1. 在实模式下实现一个Hello World程序

在 `start.s` 中实现, 代码内容同 `index` 中的 `mbr.s`

1.2. 在保护模式下实现一个Hello World程序

```
1  # TODO:关闭中断
2      cli
3      ...
4  # TODO: 设置CR0的PE位(第0位)为1
5      movl %cr0,%eax
6      orb $0x01,%al
7      movl %eax,%cr0
8      ...
9  # TODO:输出Hello world
10     pushl $13
11     pushl $message
12     calll displayStr
13     ...
14 displayStr:
15     movl 4(%esp), %ebx
16     movl 8(%esp), %ecx
```

```

17     movl $((80*5+0)*2), %edi
18     movb $0x0c, %ah
19 nextChar:
20     movb (%ebx), %al
21     movw %ax, %gs:(%edi)
22     addl $2, %edi
23     incl %ebx
24     loopnz nextChar # loopnz decrease ecx by 1
25     ret
26 ...
27 # TODO: code segment entry
28     .word 0xffff,0
29     .byte 0,0x9a,0xcf,0
30
31     # TODO: data segment entry
32     .word 0xffff,0
33     .byte 0,0x92,0xcf,0
34
35     # TODO: graphics segment entry
36     .word 0xffff,0x8000
37     .byte 0x0b,0x92,0xcf,0
38 ...

```

1.3. 在保护模式下加载磁盘中的Hello World程序运行

start.s 要实现的部分和 1.2 大致相同，这里给出 boot.c 中的代码

```

1 void bootMain(void) {
2     //TODO
3     void (*addr)(void);
4     addr=(void(*) (void))0x8c00;
5     readSect((void*)addr,1); //loadint 1 to 0x8c00
6     addr(); //jump to the specified addr
7 }

```

四、思考题

1. 你弄清楚本小结标题中各种名词的含义和他们间的关系了吗？请在实验报告中阐述。（CPU、内存、BIOS、磁盘、主引导扇区、加载程序、操作系统）

答：CPU是计算机中央处理器，内存是计算机主存储器，BIOS是基本输入输出系统，磁盘是计算机的存储介质，加载程序是主引导扇区中负责将操作系统的代码和数据从磁盘加载到内存中的一组程序，操作系统是管理、控制计算机操作，运行各种软硬件及组织用户交互等的系统软件程序。

五、实验心得

1. 在做1.1时，make 和 make play之后没有得到正确的输出。解决办法是在更改代码之后清除make的内容。在make play之前clean就行。感谢助教哥哥的帮助(●'◡'●)，要不然实验刚开始就被卡住了😭。
2. 做1.2时，code,data,graphics segment的type的判断还是花了一些时间的。查了一些资料去判断。
3. 做1.3时，主要就是读取加跳转。读取用给定的readSect接口，跳转查了资料：

```
1 程序的跳转是通过寻找函数名(函数指针)指向的地址来完成的，
2 因此可以使用如下代码来实现让程序跳转到0x100000000处执行
3 *((void (*)( ))0x100000000)();
4 通过typedef更加直观：
5 typedef (void (*)( )) func //返回值为void 参数为空的函数指针
6 *(func 0x100000000)();
```

总的来说这次试验“有惊无险”，希望下次实验能顺利一些。(•'◡'•)