

Lab1系统引导

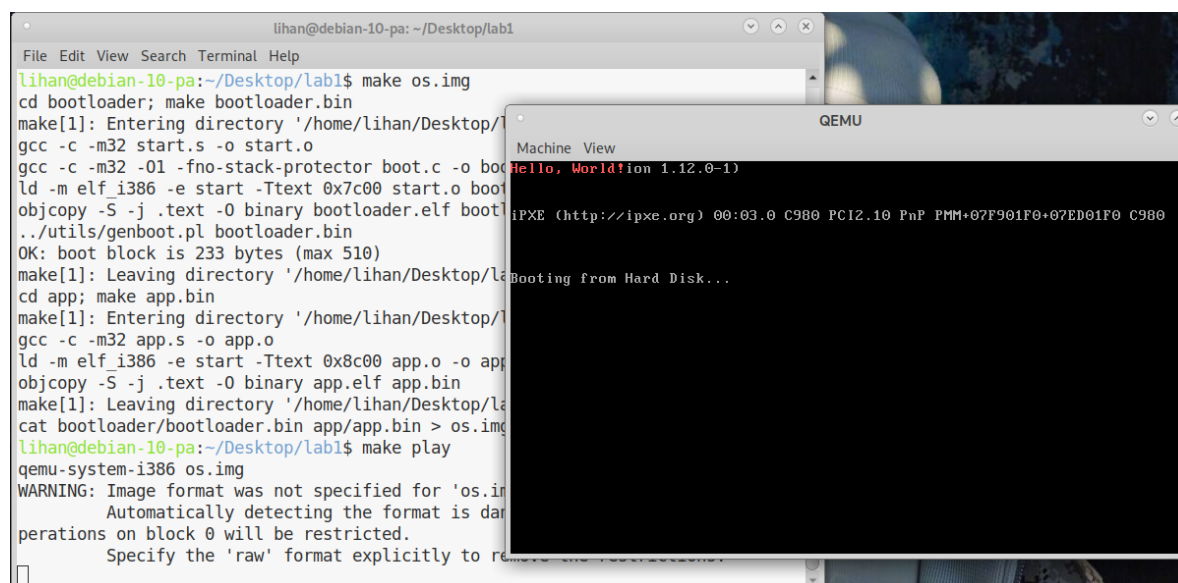
181860044 李翰

邮箱: 181860044@smail.nju.edu.cn

1. 实验进度：完成了所有内容

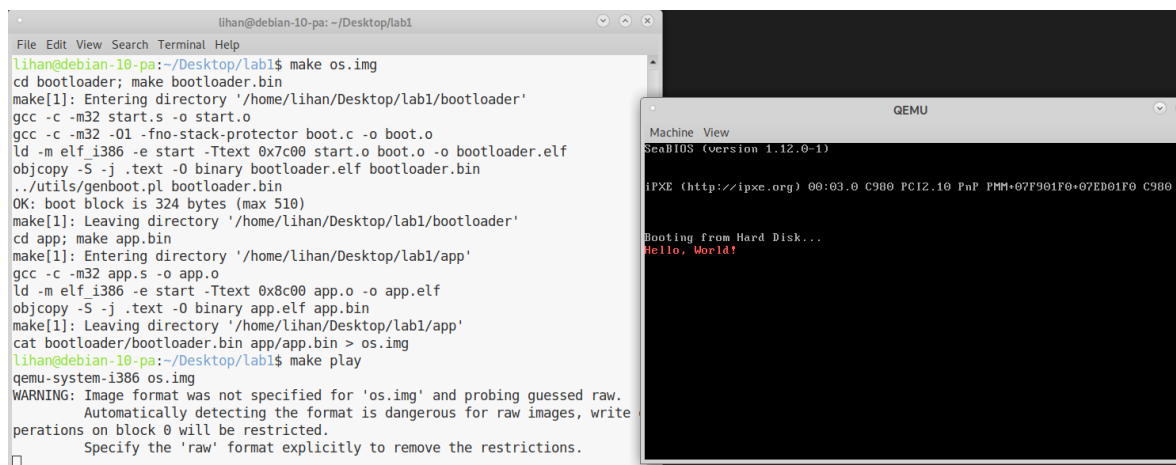
2. 实验结果：

2.1 实模式

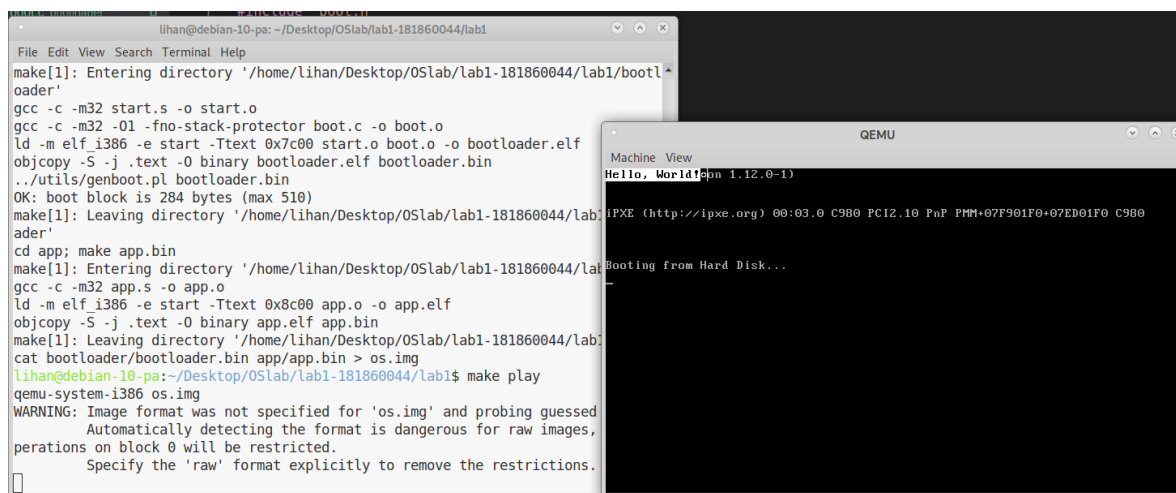


```
lihan@debian-10-pa: ~/Desktop/lab1
File Edit View Search Terminal Help
lihan@debian-10-pa:~/Desktop/lab1$ make os.img
cd bootloader; make bootloader.bin
make[1]: Entering directory '/home/lihan/Desktop/lab1/bootloader'
gcc -c -m32 start.s -o start.o
gcc -c -m32 -O1 -fno-stack-protector boot.c -o 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 233 bytes (max 510)
make[1]: Leaving directory '/home/lihan/Desktop/lab1/bootloader'
cd app; make app.bin
make[1]: Entering directory '/home/lihan/Desktop/lab1/app'
gcc -c -m32 app.s -o app.o
ld -m elf_i386 -e start -Ttext 0x8c00 app.o -o app.bin
objcopy -S -j .text -O binary app.elf app.bin
make[1]: Leaving directory '/home/lihan/Desktop/lab1/app'
cat bootloader/bootloader.bin app/app.bin > os.img
lihan@debian-10-pa:~/Desktop/lab1$ make play
qemu-system-i386 os.img
WARNING: Image format was not specified for 'os.img'.
Automatically detecting the format is dangerous.
Operations on block 0 will be restricted.
Specify the 'raw' format explicitly to remove the warning.
QEMU
Machine View
Hello, World!
ion 1.12.0-1)
iPXE (http://ipxe.org) 00:03.0 C980 PC12.10 PnP FMM+07F901F0+07ED01F0 C980
Booting from Hard Disk...
```

2.2 保护模式



2.3 保护模式下加载磁盘



3. 代码修改

3.1 实模式

提供的代码已经实现好，无需修改

3.2 保护模式

系统启动时首先进入8086的实模式，然后通过 `cli` 指令关中断，接着启动 `A20` 数据总线，然后加载 `GDTR` 寄存器，设置 `CR0` 的 `PE` 位为 1 启动保护模式，最后长跳转切换到保护模式。

上述过程所涉及的代码讲义中已经提供，需要增加的内容有两部分：

①初始化 `DS` 等段寄存器以及栈顶指针 `ESP`（位于 `/bootloader/start.s` 文件中 `start32` 部分）

②保护模式下文字的显示（同样位于 `/bootloader/start.s` 文件中 `start32` 部分）

这一部分的实现主要模仿了 `/app/app.s` 中的代码，将 `%edi` 设置为 `$((80*8+0)*2)`，在第八行第一列显示字符，将 `%ah` 设置为 `$0x0c` 即黑底红字，然后通过循环控制将 `%al` 设置为待显示的 ASCII 字符，再通过 `movw %ax, %gs:(%edi)` 写入显存。代码大致如下：

```
28     pushl $13 # pushing the size to print into stack
29     pushl $message # pushing the address of message into stack
30     calll displayStr # calling the display function
31 loop:
32     jmp loop
33
34 message:
35     .string "Hello, World!\n\0"
36
37 displayStr:
38     movl 4(%esp), %ebx
39     movl 8(%esp), %ecx
40     movl $((80*8+0)*2), %edi
41     movb $0x0c, %ah
42 nextChar:
43     movb (%ebx), %al
44     movw %ax, %gs:(%edi)
45     addl $2, %edi
46     incl %ebx
47     loopnz nextChar # loopnz decrease ecx by 1
48     ret
```

3.3 保护模式下加载硬盘

在上一步的基础上，修改文字显示的方式

（`/bootloader/start.s` 文件中 `start32` 部分代码）——跳转至 `/bootloader/boot.c` 中的 `bootMain` 函数，并在该函数中增加跳转到加载程序的相关代码，便可通过该函数读取磁盘特定扇区中的程序至内存特定位置并跳转执行。

4.思考&心得体会

1.三个GDT表项的基址，段限，权限

代码段描述符 (0x00cf 9a00 0000 ffff) : 基址为 0x0 , 段限 0xffffffff , 特权级 0

数据段描述符 (0x00cf 9200 0000 ffff) : 基址为 0x0 , 段限 0xffffffff , 特权级 0

视频段描述符 (0x00cf 920b 8000 ffff) : 基址为 0xb8000 , 段限 0xffffffff , 特权级 0

2.切换到保护模式时,之所以要打开 A20 数据总线,是因为如果该地址线不打开,则地址的第20位会一直是0,导致无法访问到所有的内存。

3.设置 CS 段寄存器时是通过长跳转指令实现,原因是无法直接修改 CS 段寄存器。

4.在修改完成代码后,运行结果在输出 Hello World! 之后还出现了一个奇怪的字符,经检查发现是换行符'\n'导致,即实际上并没有换行而是输出了一个奇怪的字符,个人觉得可能是因为'\n'无法识别或者被识别成了其他字符。

5.通过本次实验,本人回顾了上学期计算机系统基础课上曾讲过的诸如段寄存器、段描述符等知识点,同时也更加深入的理解了一个计算机系统开机后引导启动的过程。