

OSLAB2 实验报告

计算机科学与技术系

李杨 161220071
1103266550@qq.com

实验要求及实现

1. 要求：在 bootloader 中实现从实模式进入保护模式，将内核内的内容复制至内存，并跳转至 kernel

实现：由于 lab1 已经实现了从实模式进入保护模式，所以我们只需要在 start.s 中加入域 lab1 相似的汇编代码即可。至于将内容从内核读取并存储到内存，这里从 ics 的实验中可以知道，可以采用 memcpy 和 memset 两个函数。这里并没有直接调用这两个函数，而是采用与这两个函数相同意义的函数加以代替。具体操作是，先将内核里面的内容复制到一个独立声明的 buf 数组里，然后在将内容从 buf 数组移植到内存之中。在 pa 中我们就已经知道了具体操作，所以就是将内核中的内容复制到[VirtAddr, VirtAddr + FileSiz]这个区间，并将[VirtAddr + FileSiz, VirtAddr + MemSiz]这个区间的内容给置零。然后在跳转至 kernel。

2. 要求：内核初始化 IDT (Interrupt Descriptor Table, 中断描述符表)，初始化 GDT，初始化 TSS (Task State Segment, 任务状态段)

实现：初始化 IDT,照抄 pa 的函数声明与中断处理函数，然后在 dolrp.s 中对这 15 个中断函数进行填充内容。

初始化 gdt 时大多数表项已初始化完成，只需要参照框架初始化视频段即可。初始化 TSS 时，只需要初始化 esp0 和 ss0，将 esp0 置为 0x200000(用户程序的入口地址)，ss0 利用宏 KSEL(kDATA)初始化 (其中 kDATA=2,内核代码段在 GDT 中的第二个表项)。此处还要设置正确的段寄存器，仿照 lab1，将 ds 赋值为 KSEL(kDATA),gs 赋值为 KSEL(6) (视频段在 GDT 中的第六个表项)。

3. 要求：内核加载用户程序至内存，对内核堆栈进行设置，通过 iret 切换至用户空间，执行用户程序

实现：与 1 实现方法一致，然后将用户程序加载到指定内存地址，设置对应的寄存器，通过 iret 跳转至用户程序的入口地址，执行用户程序。

4. 要求：用户程序调用自定义实现的库函数 printf 打印字符串

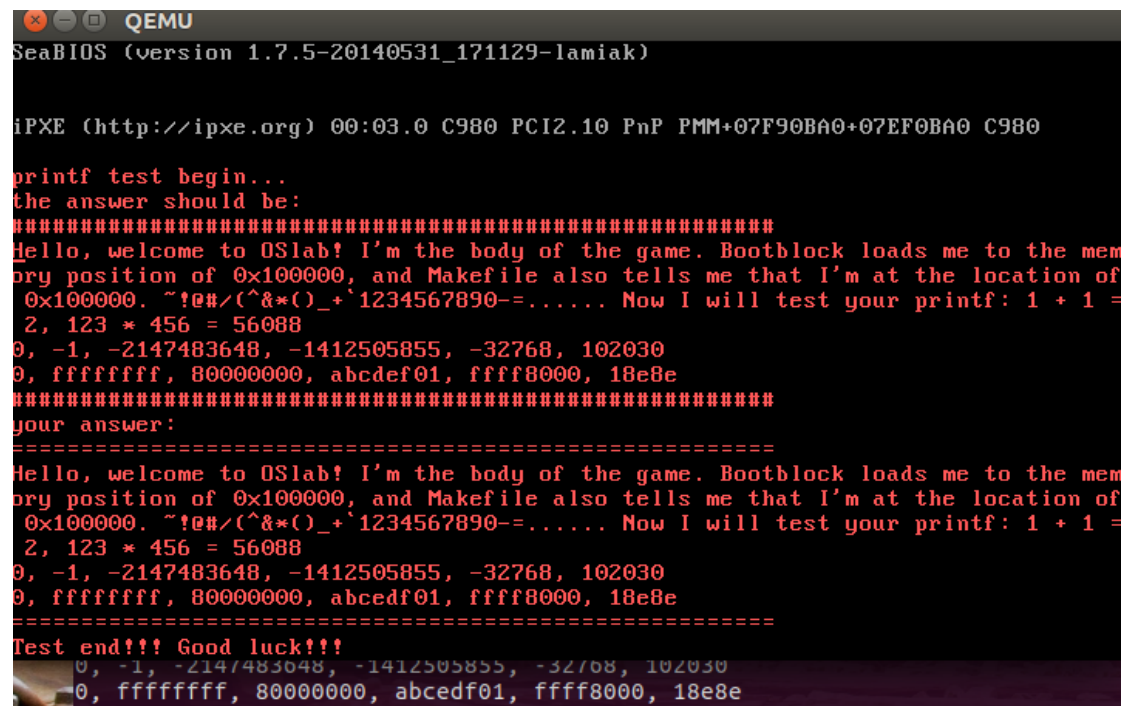
实现：要考虑的有以下几种可能情况：%d,%c,%x,%s,以及各种可能的%%情况。这里可以声明一个 1024 长度的 char 类型的数组用于存放需要打印的内容。这里可以分为 str2buf (将 string 存入 buf 中)，ch2buf(将一个字符存入 buf 中)，minus2buf(将整数存入 buf 中，这里面考虑了负数的情况，其实就是加了一个-号)，以及 hex2buf (将数字转换为 16 进制，存放在 buf 中)

5. 要求 printf 基于中断陷入内核，由内核完成在视频映射的显存地址中写入内容，完成字

字符串的打印

实现：在 Lab1 的基础上，使程序陷入内核很容易，至于完成在视频映射的显存地址中写入内容，只要以上的每一步操作正确，然后保证自己自定义的 printf 正确的话就可以实现了。

实现：



```
QEMU
SeaBIOS (version 1.7.5-20140531_171129-lamiak)

iPXE (http://ipxe.org) 00:03.0 C980 PCI2.10 PnP PMM+07F90BA0+07EF0BA0 C980

printf test begin...
the answer should be:
#####
Hello, welcome to OSlab! I'm the body of the game. Bootblock loads me to the mem
ory position of 0x100000, and Makefile also tells me that I'm at the location of
0x100000. ~!@#/(^&*()_+`1234567890-=..... Now I will test your printf: 1 + 1 =
2, 123 * 456 = 56088
0, -1, -2147483648, -1412505855, -32768, 102030
0, ffffffff, 80000000, abcedf01, ffff8000, 18e8e
#####
your answer:
=====
Hello, welcome to OSlab! I'm the body of the game. Bootblock loads me to the mem
ory position of 0x100000, and Makefile also tells me that I'm at the location of
0x100000. ~!@#/(^&*()_+`1234567890-=..... Now I will test your printf: 1 + 1 =
2, 123 * 456 = 56088
0, -1, -2147483648, -1412505855, -32768, 102030
0, ffffffff, 80000000, abcedf01, ffff8000, 18e8e
=====
Test end!!! Good luck!!!
0, -1, -2147483648, -1412505855, -32768, 102030
0, ffffffff, 80000000, abcedf01, ffff8000, 18e8e
```