**PartA**

**Exercise 2.**

**Use GDB's si (Step Instruction) command to trace into the ROM BIOS for a few more instructions, and try to guess what it might be doing. You might want to look at**[**Phil Storrs I/O Ports Description**](http://web.archive.org/web/20040404164813/members.iweb.net.au/~pstorr/pcbook/book2/book2.htm)**, as well as other materials on the**[**Jos reference materials page**](https://ipads.se.sjtu.edu.cn/courses/os/2015/reference.html)**. No need to figure out all the details - just the general idea of what the BIOS is doing first.**

1. 第一条指令



CPU加电后会将CS和IP的初始值传给对应的寄存器，因此根据16\*cs+ip可以得知第一条指令从0x000ffff0处开始，BIOS指令区域为0x000f0000-0x00100000，即第一条指令从距离BIOS指令区顶端16bytes处开始执行。第一条指令是跳转，后面的地址格式应该分别是cs和ip指令，按理说两个地址应该分别为0xf0000，0xe05b从而得到下一步的有效地址是0xfe05b，但是从gdb调试中却发现ip的地址为0xf000e05b，是cs和ip拼接而成的一个指令，猜测应该是用eip充当了cs+ip，前面的那个值是无用，因此是直接分离eip中的cs和ip计算得到了新的指令地址。

1. 完成跳转之后的执行的指令如下，作用如注释中所示：

**[f000:e05b] 0xfe05b: cmpw $0x48,%cs:(%esi) //比较两处的值**

**[f000:e062] 0xfe062: jne 0xd241d09e //如果不相等则跳转**

**[f000:e066] 0xfe066: xor %edx,%edx**

**[f000:e068] 0xfe068: mov %edx,%ss**

**[f000:e06a] 0xfe06a: mov $0x7000,%sp //设置栈段寄存器和栈指针寄存器**

**[f000:e070] 0xfe070: mov $0x1e0a,%dx //寄存器dx赋值**

**[f000:e076] 0xfe076: jmp 0x5576cf15 //跳转**

**[f000:cf13] 0xfcf13: cli //关闭中断**

**[f000:cf14] 0xfcf14: cld //复位方向标志位**

**[f000:cf15] 0xfcf15: mov %ax,%cx //保存ax值到cx中**

**[f000:cf18] 0xfcf18: mov $0x8f,%ax**

**[f000:cf1e] 0xfcf1e: out %al,$0x70 //将al值输出到端口0x70**

**[f000:cf20] 0xfcf20: in $0x71,%al //将0x71端口值输入到al(三条指令修改CMOS端口寄存器内容，屏蔽NMI终端，最后读入的值并没有被用到)**

**[f000:cf22] 0xfcf22: in $0x92,%al //将0x92端口的值输入到al**

**[f000:cf24] 0xfcf24: or $0x2,%al //设置al中第二个bit为1**

**[f000:cf26] 0xfcf26: out %al,$0x92 //写回0x92**

**[f000:cf28] 0xfcf28: mov %cx,%ax //恢复ax的值**

**[f000:cf2b] 0xfcf2b: lidtl %cs:(%esi) //加载值到中断描述符表寄存器**

**[f000:cf31] 0xfcf31: lgdtl %cs:(%esi) //加载值到全局描述符表格寄存器**

**[f000:cf37] 0xfcf37: mov %cr0,%ecx**

**[f000:cf3a] 0xfcf3a: and $0xffff,%cx**

**[f000:cf41] 0xfcf41: or $0x1,%cx**

**[f000:cf45] 0xfcf45: mov %ecx,%cr0 //cr0中第一位bit置1进入保护模式**

**[f000:cf48] 0xfcf48: ljmpw $0xf,$0xcf50 //跳转至0xfcf50**

之后又执行了一系列的修改寄存器值得指令，是加载完全局描述符表格寄存器之后要求执行得指令，便不继续深入了。