

qemu 的使用与软件调试

单步跟踪 BIOS，从 0x7c00 开始跟踪代码运行,将单步跟踪反汇编得到的代码与 bootasm.S 和 bootblock.asm 进行比较。

用 gdb 调试时源码显示不出来，但是通过 layout asm 能看到汇编代码，再用 layout regs 查看寄存器的值，结果如下：



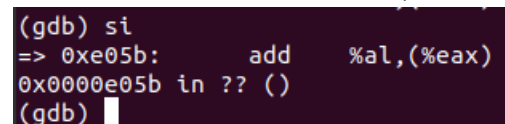
```
终端
文件(F) 编辑(E) 查看(V) 搜索(S) 终端(T) 帮助(H)

Register group: general
eax      0x0      0          ecx      0x0      0
edx      0x663    1635       ebx      0x0      0
esp      0x0      0x0         ebp      0x0      0x0
esi      0x0      0          edi      0x0      0
eip      0xffff0  0xffff0    eflags   0x2      [ ]
cs       0xf000   61440      ss       0x0      0

> 0xffff0 add %al, (%eax)
0xffff2 add %al, (%eax)
0xffff4 add %al, (%eax)
0xffff6 add %al, (%eax)
0xffff8 add %al, (%eax)
0xffffa add %al, (%eax)

remote Thread 1 In: L?? PC: 0xffff0
Source directories searched: /home/lsy/Desktop/OS/labcodes/lab1/kern/init:$cd
$cd
(gdb) show src
Undefined show command: "src". Try "help show".
(gdb) layout src
(gdb) layout asm
(gdb) layout regs
(gdb)
```

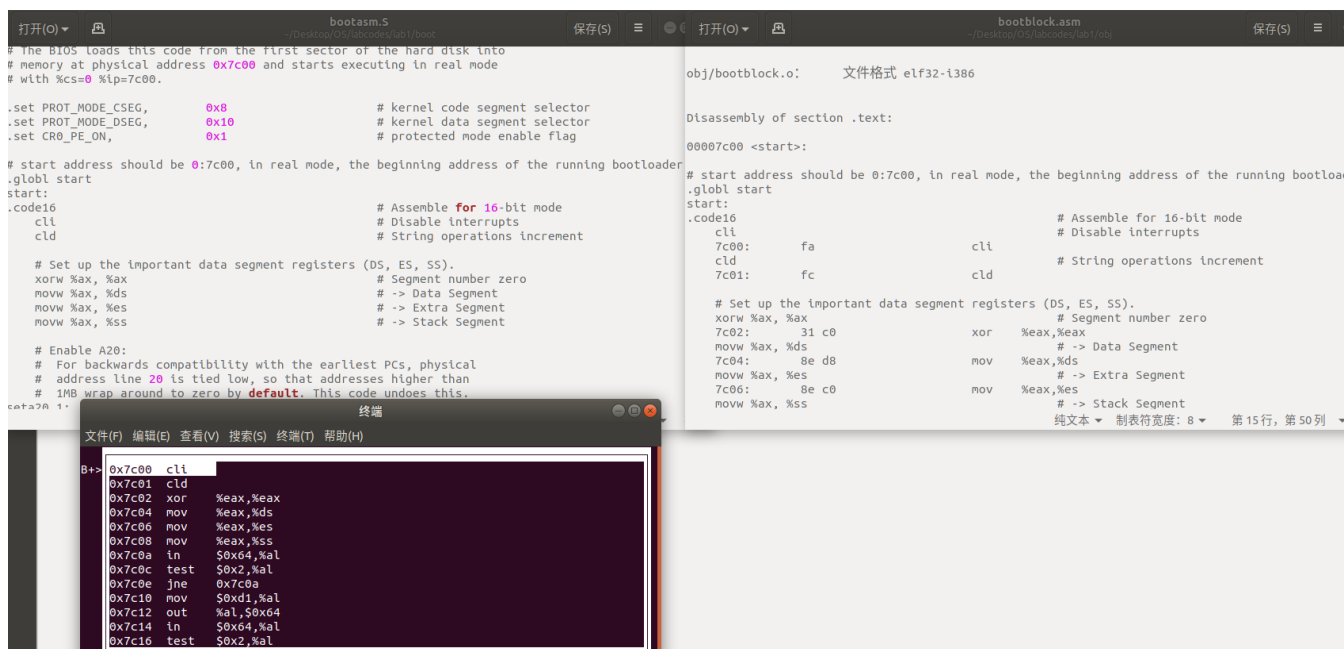
输入 si 命令查看下一条指令，eip 的值变为了 0xe05b，这应该就是 BIOS 开始的地址了



```
(gdb) si
=> 0xe05b:      add    %al, (%eax)
0x0000e05b in ?? ()
(gdb)
```

在 gdbinit 文件中加入：

```
b *0x7c00          # 在 0x7c00 处设置断点
c
x /2i $pc          # 显示当前 eip 处的汇编指令
set architecture i386 # 设置当前调试得 CPU 是 80386
然后再次 make debug
```



用 `layout asm` 命令查看此时的汇编代码，与 `bootasm.S` 和 `bootblock.asm` 比较，三者代码指令完全相同，