

Report for Lab3——实模式下的 Hello World

Stu：金泽文

No.PB15111604

实验内容：

1. 编写一段 16 进制的汇编程序，用作启动扇区代码
2. 编写一个链接脚本，将启动代码按照启动扇区进行链接
3. 制作启动软盘，将启动扇区代码安装到软盘的启动扇区(也可以选择其他介质)
4. 启动运行启动扇区代码，最后进入死循环
5. 编写一个 shell 脚本文件，将 make，制作启动软盘，启动运行等步骤放在这个脚本文件中

实验步骤：

·学习 x86 汇编，并编写一段 16 进制的汇编程序，用作启动扇区代码。

参考 [这篇博客](#)，以及《Computer System: A Programmer's Perspective》学习简单的 x86 汇编（AT&T 格式）。

·学习并编写一个链接脚本，将启动代码按照启动扇区进行链接。

参考[这个博客](#)（写完意外地发现老师在群里发了一个 ld 脚本，嚶嚶嚶）

·同时，为了方便之后 n 次的反复操作，编写了整个 shell 脚本。

刚开始的时候，不知道有 lods b，movsb 等 routine，所以暴力存取，并且当时理解错了 data section text section 的用法，没能正确读取字符串的地址，所以只好手动将字符存入到 vga 中。当时的代码是这样的。

在听送小牛说有 movsb 这种好东

```
→ Documents ./boot1.sh
gcc -c t2.s -o t2.o -m32
ld -Tt1.ld -o t1.elf t2.o

objcopy -O binary t1.elf t1.bin
Hello World!PB15111604
记录了2880+0 的读入
记录了2880+0 的写出
1474560 bytes (1.5 MB, 1.4 MiB) copied, 0.0074 s, 147.4 MB/s
dd if=/dev/zero of=a.img bs=512 count=2880

sudo losetup /dev/loop4 a.img
记录了1+0 的读入
记录了1+0 的写出
512 bytes copied, 0.000488633 s, 1.0 MB/s
sudo dd if=t1.bin of=/dev/loop4 bs=512 count=1

qemu -fda a.img
sudo losetup -d /dev/loop4

WARNING: Image format was not specified for 'a.img' and probing guessed raw.
Automatically detecting the format is dangerous for raw images, write operations on block 0 will be restricted.
Specify the 'raw' format explicitly to remove the restrictions.

→ Documents
```

```
31     mov $0, %di
32     mov $0xb800, %ax
33     mov %ax, %es
34     print:
35     movw $0, %dx
36     movb %ds:(%si), %dl
37     movb %dl, %es:(%di)
38     inc %di
39     movb $0x1e, %es:(%di)
40     inc %di
41     movw $0, %dx
42     movb $'e', %dl
43     movb %dl, %es:(%di)
44     inc %di
45     movb $0x1e, %es:(%di)
46     inc %di
47     movw $0, %dx
48     movb $'l', %dl
49     movb %dl, %es:(%di)
50     inc %di
51     movb $0x1e, %es:(%di)
52     inc %di
53     movw $0, %dx
54     movb $'l', %dl
55     movb %dl, %es:(%di)
56     inc %di
57     movb $0x1e, %es:(%di)
58     inc %di
59     movw $0, %dx
60     movb $'o', %dl
61     movb %dl, %es:(%di)
62     inc %di
63     movb $0x1e, %es:(%di)
64     inc %di
65     movw $0, %dx
66     movb $' ', %dl
67     movb %dl, %es:(%di)
68     movb $' ', %dl
69     movb %dl, %es:(%di)
70     inc %di
71     movw $0, %dx
72     movb $'W', %dl
73     movb %dl, %es:(%di)
74     inc %di
75     movb $0x1e, %es:(%di)
76     inc %di
77     movw $0, %dx
78     movb $'o', %dl
79     movb %dl, %es:(%di)
80     inc %di
81     movb $0x1e, %es:(%di)
82     inc %di
83     movw $0, %dx
84     movb $'r', %dl
85     movb %dl, %es:(%di)
86     inc %di
87     movb $0x1e, %es:(%di)
88     inc %di
89     movw $0, %dx
90     movb $'l', %dl
91     movb %dl, %es:(%di)
92     inc %di
93     movb $0x1e, %es:(%di)
94     inc %di
95     movw $0, %dx
96     movb $'d', %dl
97     movb %dl, %es:(%di)
98     inc %di
99     movb $0x1e, %es:(%di)
100    inc %di
101    movw $0, %dx
102    movb $'!', %dl
103    movb %dl, %es:(%di)
```

西并且调试 n 次之后，得到以下结果。（当时要求的明明是输出这个东西 “hello world pb15111604 jinzewen”。产品经理竟然改需求。）
之后更改得到如下图所示。

```
→ Documents WARNING: Image format was not specified for 'a.img' and probing gue
ssed 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.
./boot1.sh
gcc -c final.s -o t2.o -m32
ld -Tt1.ld -o t1.elf t2.o
objcopy -O binary t1.elf t1.bin
记录了2880+0 的读入
记录了2880+0 的写出
1474560 bytes (1.5 MB, 1.4 MiB) copied, 0.0236683 s, 62.3 MB/s
dd if=/dev/zero of=a.img bs=512 count=2880
[sudo] zevin 的密码:
#关中断后的初始化
sudo losetup /dev/loop4 a.img
HelloPB15111604
记录了1+0 的读入
mov $0xb800, %eax #赋值es, 由于段基
记录了1+0 的写出
mov %eax, %es
512 bytes copied, 0.0273327 s, 18.7 kB/s
sudo dd if=t1.bin of=/dev/loop4 bs=512 count=1
qemu -fda a.img
sudo losetup -d /dev/loop4
→ Documents WARNING: Image format was not specified
```

以下是调试过程的截图：

```
→ Documents ./boot2.sh
gcc -c t3.s -o t2.o -m32
ld -Tt1.ld -o t1.elf t2.o
objcopy -O binary t1.elf t1.bin
记录了2880+0 的读入
记录了2880+0 的写出 #关中断
1474560 bytes (1.5 MB, 1.4 MiB) copied, 0.123498 s, 11.9 MB/s
dd if=/dev/zero of=a.img bs=512 count=2880
[sudo] zevin 的密码:
#关中断后的初始化
sudo losetup /dev/loop4 a.img
HelloPB15111604
boot2.sh
rootfs
rootfs.img
t1.s t.s
t2.o tt.s
Reading symbols from t1.elf...done.
(gdb) target remote :1234
Remote debugging using :1234
0x0000ffff in ?? ()
(gdb) l
1 # generate 16-bit code
2 .code16 #关中断
3
4 # hint the assembler that here is the executable
5 section .text
6
7 .globl _start;
8 mov $0xb800, %eax #赋值es, 由于段基
9 _start:
10 mov %eax, %es
11 cli #由于vga有25行80列
(gdb) b 10
Breakpoint 1 at 0x7c00: file t3.s, line 10.
(gdb) c
Continuing.
Breakpoint 1, _start () at t3.s:10
10 cli
(gdb) 
```

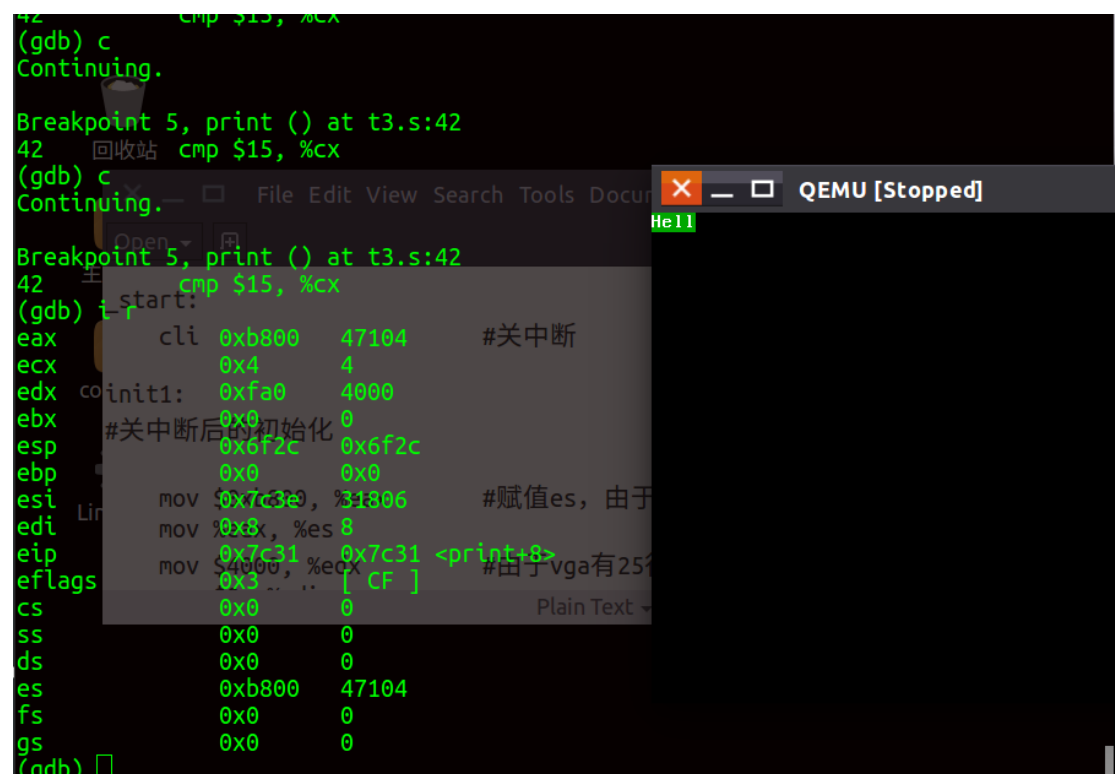
清屏之后



```
21
22 movb $0, %es:(%di)
(gdb) _start:
23 inc %di
24 cli #关中断
25 cmp %dx, %di
26 jne clear_loop
27 #关中断后的初始化
28 init2:
29
30 lea message, %si #赋值es, 由于段基址
31 mov %si, %di
32 mov $4000, %edx #由于vga有25行80列
(gdb) b 30
Breakpoint 4 at 0x7c1f: file t3.s, line 30.
(gdb) c
Continuing.

Breakpoint 4, init2 () at t3.s:30
30 lea message, %si
(gdb) 
```

Print 中间查看寄存器



```
42
(gdb) c
Continuing.

Breakpoint 5, print () at t3.s:42
42 回收站 cmp $15, %cx
(gdb) c
Continuing.

Breakpoint 5, print () at t3.s:42
42 cmp $15, %cx
(gdb) i
eax cli 0xb800 47104 #关中断
ecx 0x4 4
edx coinit1: 0xfa0 4000
ebx #关中断后的初始化 0x0 0
esp 0x6f2c 0x6f2c
ebp 0x0 0x0
esi mov $0x7c31, %si #赋值es, 由于
edi mov %si, %es 8
eip 0x7c31 0x7c31 <print+8>
eflags mov $4000, %ecx #由于vga有25行
0x3 [ CF ]
cs 0x0 0
ss 0x0 0
ds 0x0 0
es 0xb800 47104
fs 0x0 0
gs 0x0 0
(gdb) 
```

实验步骤部分结束。

回答问题：

1. 什么是 BIOS ? BIOS 和启动软盘之间的启动协议是什么？

软盘的启动扇区最后两字节如果是 0xaa55，那么软盘即为启动设备，否则识别软盘为 unbootable device。

2. 启动时为什么要关中断

关中断，能够避免在启动时被意料之外的中断所干扰。保证启动过程安全进行。同时，不让我们用中断写入到 vga 中，而是自己直接写入显存。

3. 什么是实模式？实模式初始化哪些寄存器？

实模式就是在位宽只有 16 位的情况下实现 20 位的寻址空间（即 1MB）。实模式通过段：段内偏移来实现 20 位。需要初始化 cs、ds、ss、si、di 等寄存器。

4. VGA 的接口是什么，如何编程实现 VGA 输出

Vga 的接口即为 vga 显存地址内容。（或者说没有接口，直接存取？）实模式下，由于 vga 显存从 0xb8000 开始，每个字符对应两个字节，分别存放属性和字符，所以只要对应赋值即可。

5. 什么是链接脚本？启动软盘的链接脚本有哪些注意的地方？

链接脚本用于规定如何把输入文件的 section 对应地放入到输出文件中，并控制输出文件各部分在内存空间中的布局。启动软盘的 OUTPUT_FORMAT 应该对应我们操作系统下正确的 elf 格式，即 elf32-i386，OUTPUT_ARCH 对应正确的 i386，并且 ENTRY 要对应正确的汇编代码中的 global 起始常量，在我的汇编代码中对应的是 _start 部分。由于软盘空间从 0x7c00 开始，所以要把输出文件起始地址对应到内存空间的 0x7c00 处。另外，为了最后两个 0xaa55，可以设置 signature。（也可以不在这里设置，而直接在汇编代码中设置。）

6. 从 Power-on 开始，说清楚计算机是如何一步一步启动，直到开始运行你写的代码

① BIOS

从通电开始，ROM 内的 BIOS 程序被读取并执行。BIOS 先检查硬件是否满足启动条件。硬件自检之后，开始按照启动顺序依次访问各个设备对应的 MBR，如果最后两个字节为 0xaa55，则认定为启动程序，否则继续检查。

② MBR

正常情况下，MBR 包含操作系统的调用信息，以便计算机继续启动。MBR 还包括分区表，以及最后两个上面提到的 0xaa55。

而在我们这次实验中，我们的汇编代码转化的二进制代码直接存放在启动软盘上面，所以直接被计算机所执行。

7. 给出代码运行关键时刻的截屏并加以说明。

这一步在最前面都有详细的交代。

【感谢】感谢积极回答我的问题并丝毫没有嫌弃我的陈香兰老师，以及与跟我一起积极讨论

并给予帮助的宋小牛同学。

参考

【1】 <http://www.cnblogs.com/orlion/p/5765339.html>

【2】 <http://blog.csdn.net/mrwangwang/article/details/9097411>