

实验一

李卓 pb19000064

一， 实验内容

建立 Ubuntu 和主机的共享文件夹

了解必要的 multiboot 协议相关内容

安装 QEMU，了解 QEMU 对 multiboot 的支持情况

遵守 multiboot 协议，写 multiboot header

直接在 VGA 的显存中写“helloworld”

使用串口输出“HELLOWORLD”

了解并掌握必要的汇编

了解 Makefile 和链接描述文件 1

二， 实验原理

利用 Ubuntu 和 x86 汇编 编写支持 multiboot 启动协议的内核，.

运行 make 指令能够成功生成 multibootHeader.bin，然后通过

qemu 运行，将 helloworld 显示在界面上，输出方式有 vga 输出

（直接将字符以指定格式写到 vga 显存位置），串口输出（将字符直接写到端口地址）

三， 实验过程及代码

电脑上之前装过 VMware，直接编写 multibootHeader.S 文件

以下是对 multiboot Header 特有数据段进行设置，以便引导程序能够识别

```

.globl start

MAGIC_ITEM_NAME = 0x1BADB002 ;
FLAGS_ITEM_NAME = 0x00 ;
CHECKSUM_ITEM_NAME = - 0x1BADB002 ;

.section ".multiboot_header"

    .long MAGIC_ITEM_NAME
    .long FLAGS_ITEM_NAME
    .long CHECKSUM_ITEM_NAME
    .long start

.text
.code32

```

然后是输出部分, vga 输出: 使用指令 `movl $0x12345678, 0xB8000`

输出两个字符。串口输出: 初始化串口后 将字符依次写入串口。

查 ASCII 对照表将结果填上。最后 `hlt` 停机

start:

```

    movl $0x2f452f48, 0xB8000
    movl $0x2f4c2f4c, 0xB8004
    movl $0x2f572f4f, 0xB8008
    movl $0x2f522f4f, 0xB800c
    movl $0x2f442f4c, 0xB8010
    movl $0x2f202f21, 0xB8014

    movl $0x2f422f50, 0xB8018
    movl $0x2f392f31, 0xB801c
    movl $0x2f302f30, 0xB8020
    movl $0x2f302f30, 0xB8024
    movl $0x2f342f36, 0xB8028

    movb $0x46, %al
    movw $0x3F8, %dx
    outb %al, %dx
    nop
    nop
    hlt

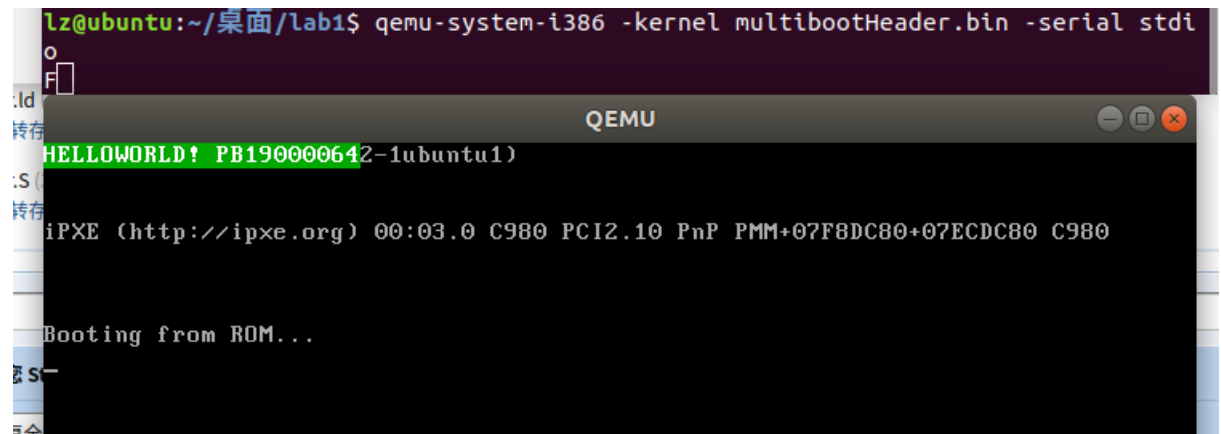
```

这样查表很蠢, 也可以写个 C 函数, 先对整个字符串做二进制转换, 再输出。有 bug 懒得改了

内存上, vga 地址 0xB8000, 每个字符占用四个字节, 地址依次加

4。串口地址为 0x3F8, , 存入%dx, 字符先存入%al 再从该地址输出。

实验结果：串口输出 F , vga 输出 HELLOWORLD!+学号



The screenshot shows a terminal window with the command `qemu-system-i386 -kernel multibootHeader.bin -serial stdio` being executed. The output in the terminal window is as follows:

```
o
F
HELLOWORLD! PB190000642-1ubuntu1)
iPXE (http://ipxe.org) 00:03.0 C980 PCI2.10 PnP PMM+07F8DC80+07ECDC80 C980

Booting from ROM...
S-
```