

# Lab1 实验报告

PB21020552\_张易

## 一.实验目标

本实验用来实现一个最简单的multiboot header同时具有以下功能

- 在屏幕上输出特定的内容.
- 在串口输出特定的内容.

## 二.实验原理

- **multiboot介绍** 当前设计的每个操作系统可能都有其自己的引导程序,在一台机器上安装有多个操作系统,如果用典型的链式引导机制让这些操作系统可靠地共存,这是不现实的。对于一个特定的操作系统来说,可选的引导加载器不是很多,甚至根本没有可选的解决方案。基本上, multiboot协议指定了引导加载程序与操作系统之间的接口,这样任何符合规范的引导加载程序都应该能够加载任何符合的操作系统。本规范没有指定引导加载程序应该如何工作,只是规定了它们必须如何与正在加载的操作系统一起使用该接口。本实验中采取最简单的multiboot启动,同时具有以下规定.magic必须被设置为0x1BADB002 ,flag设置成0,checksum=-(magic+flag)
- **qemu** QEMU是一套可以在Windows操作系统中仿真出另一套操作系统的仿真软件。因为它可以在系统中仿真出一颗虚拟的CPU,然后将要仿真的操作系统交给这颗虚拟CPU去处理,所以能够达到同一平台却能同时执行两种操作系统的目的。在本实验中,把写好的multiboot header放在QEMU环境中实验
- **vga介绍** VGA (Video Graphics Array), 是模拟信号,只能传输视频信号。它传输红、绿、蓝模拟信号以及同步信号(水平和垂直信号)。在本实验中,通过给vga起始地址出用movl指令,在vga起始地址处逐渐放入字节,同时对于一个字符需要两个字节,一个用来存放字符的ascii码,一个用来存放字符的显示属性
- **UART介绍** UART(Universal Asynchronous Receive),由于在QEMU中不需要对uart进行波特率初始化,其端口地址为0x3f8,可以直接通过outb指令,先给特定的寄存器放置对应的ASCII码值,再在串口处输出.

## 三.源代码说明

- **multiboot header构建**

- 定义参数

```
magic = 0x1BADB002
flags = 0x00000000
checksum = -(magic + flags)
```

按照Multiboot Header协议定义 magic必须定义成0x1BADB002,同时flags可以设定成0,checksum需要设置成checksum+magic+flags=0的形式.

- 声明参数

```
.section ".multiboot_header"      #先起一个section的名字
.align 4
.long magic
.long flags
.long checksum
```

本段开辟出语句端段,.align 4 使得位对齐,.long声明出Multiboot的三个域.

- VGA输出

- 原有字符清除

```
movl $0xB8000, %ebx # 给予显存地址
clean:
movl $0, (%ebx,%eax,4) # 每一次操作四个字节
addl $1, %eax
cmpl $0x03e8, %eax # 查询资料知一屏幕内容有4000个字节,选择大于4000字节的内存空间完成清理
jle clean
```

在最初的输出中,发现了大量空余字符的输出,所以选择对于这些字符进行清除,同时查阅资料发现,一页显存至多存在4000个字节,利用movl进行清除,通过%eax寄存器的循环清楚4000个字节范围内的字节

- VGA字符输出 按照实验文档的方法逐渐在vga增加位置放置对应的属性以及所需字符的ascii码值.

```
vgaoutput:
movl $0x2f652f48, 0xB8000 # movl指令,前面的地址放在后面,也就是地址从高到低进行分布
movl $0x2f6C2f6C, 0xB8004
movl $0x2f202f6F, 0xB8008
movl $0x2f6F2f77, 0xB800C
movl $0x2f6C2f72, 0xB8010
movl $0x2f212f64, 0xB8014
movl $0x2f502f20, 0xB8018
movl $0x2f322f42, 0xB801C
movl $0x2f302f31, 0xB8020
movl $0x2f302f32, 0xB8024
movl $0x2f352f35, 0xB8028
movl $0x2f5f2f32, 0xB802C
movl $0x2f682f7a, 0xB8030
movl $0x2f6e2f61, 0xB8034
movl $0x2f792f67, 0xB8038
movl $0x00002f69, 0xB803c
```

- Uart\_output输出 按照实验文档,把地址0x3F8存入%dx,movb \$0x7a, %al 将字符的ASCII码存入%al,利用outb输出

```
Uart_output:  # 一下是利用串口输出helloworld的内容
    movw $0x3f8, %dx
    nop
    movb $0x48, %al
    outb %al, %dx
    movb $0x45, %al
    outb %al, %dx
    movb $0x4c, %al
    outb %al, %dx
    movb $0x4c, %al
    outb %al, %dx
    movb $0x4f, %al
    outb %al, %dx
    movb $0x57, %al
    outb %al, %dx
    movb $0x4f, %al
    outb %al, %dx
    movb $0x52, %al
    outb %al, %dx
    movb $0x4c, %al
    outb %al, %dx
    movb $0x44, %al
    outb %al, %dx
    movb $0x5f, %al
    outb %al, %dx
    movb $0x7a, %al
    outb %al, %dx
    movb $0x68, %al
    outb %al, %dx
    movb $0x61, %al
    outb %al, %dx
    movb $0x6E, %al
    outb %al, %dx
    movb $0x67, %al
    outb %al, %dx
    movb $0x79, %al
    outb %al, %dx
    movb $0x69, %al
    outb %al, %dx
```

- hlt 直接用hlt指令停机即可.

## 代码布局说明

代码布局（地址空间）由链接描述文件（.ld文件）中的SECTIONS部分决定。

```
SECTIONS {
    . = 1M;
    .text : {
        *(.multiboot_header)
        . = ALIGN(8);
        *(.text)
    }
}
```

.S中 multiboot\_header在内存占用12个字节3\*4(.long)

## 编译过程说明

生成.bin文件

```
ASM_FLAGS = -m32 --pipe -Wall -fasm -g -O1 -fno-stack-protector

multibootHeader.bin: multibootHeader.S
    gcc -c ${ASM_FLAGS} multibootHeader.S -o multibootHeader.o
    ld -n -T multibootHeader.ld multibootHeader.o -o multibootHeader.bin

clean:
    rm -rf ./multibootHeader.bin ./multibootHeader.o
```

从实验文档中可知,利用make指令可以直接生成.bin文件.从Makefile文件, 第一条指令用gcc从multiboot.S中编译生成multiboot.o文件,再用ld文件把.o文件链接生成.bin文件.,而其中的\${ASM\_FLAGS}是一个变量, 其中包含一个或多个选项或标志, 用于传递给汇编器. 下面的clean说明,输入make clean可以删除生成的.bin与.o文件.

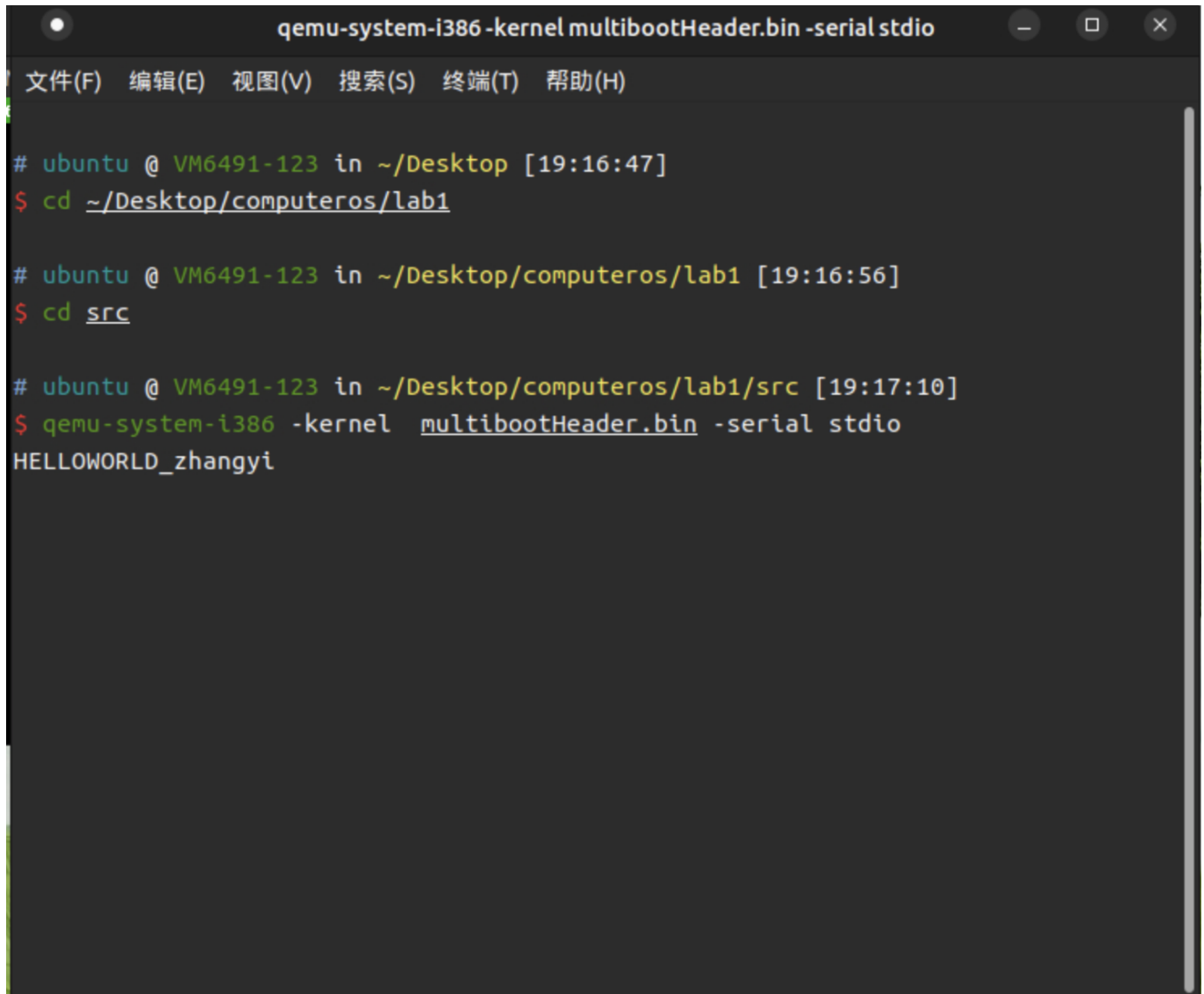
运行qemu

输入:

```
qemu-system-i386 -kernel multibootHeader.bin -serial stdio
```

## 六.运行结果

终端结果:



```
qemu-system-i386 -kernel multibootHeader.bin -serial stdio
文件(F) 编辑(E) 视图(V) 搜索(S) 终端(T) 帮助(H)

# ubuntu @ VM6491-123 in ~/Desktop [19:16:47]
$ cd ~/Desktop/computeros/lab1

# ubuntu @ VM6491-123 in ~/Desktop/computeros/lab1 [19:16:56]
$ cd src

# ubuntu @ VM6491-123 in ~/Desktop/computeros/lab1/src [19:17:10]
$ qemu-system-i386 -kernel multibootHeader.bin -serial stdio
HELLOWORLD_zhangyi
```

qemu输出结果

