OS lab1 报告

PB21051012 刘祥辉

实验原理

“QEMU”可以模拟不同的CPU体系结构、操作系统和硬件平台，使得用户可以在同一台计算机上运行不同的操作系统和应用程。 Multiboot 是一种用于在单个计算机上引导多个操作系统的规范。这种规范定义了操作系统启动信息的格式、传递方式以及如何在内存中布局多个操作系统。

VGA显存输出的起始地址为 0xB8000,每个字符需要两个字节，一个用于存放字符的ASCII码，另一个用于存放该字节的属性。

端口输出的地址为0x3F8, QEMU中，即使 MyOS 不对 uart 进行初始化，也可以正确输出。

编译运行

Makefile文件中:

- `ASM_FLAGS= -m32 --pipe -Wall -fasm -g -O1 -fno-stack-protector` 是GNU Make 编译器的参数设置
- `gcc -c ${ASM_FLAGS} multibootHeader.S -o multibootHeader.o` 是使用 GCC 编译器编译汇编语言文件的命令

`-c` 选项指定只编译、汇编，不链接；`-o multibootHeader.o` 指定输出文件的文件名为"multibootHeader.o"

- `ld -n -T multibootHeader.ld multibootHeader.o -o multibootHeader.bin` 使用 GNU 链接器将目标文件链接成可执行文件
- `rm -rf ./multibootHeader.bin ./multibootHeader.o` 用于删除当前目录下的"multibootHeader.bin" 和"multibootHeader.o" 文件。

multibootHeader.ld文件中:

- `OUTPUT_FORMAT("elf32-i386", "elf32-i386", "elf32-i386")`: 指定输出格式为ELF 32位x86架构。
- `OUTPUT_ARCH(i386)`: 指定输出架构为i386。
- `ENTRY(start)`: 指定程序入口地址为标号"start"的地址。
- `SECTIONS`: 定义输出文件中各个节的位置和属性。
- `. = 1M;`: 设置当前位置为1M。
- `.text`: 定义一个节名为".text"的节。
- `*(.multiboot_header)`: 将所有标记为".multiboot_header"的输入节拷贝到输出文件中的".text"节中。
- `. = ALIGN(8);`: 设置当前位置按8字节对齐。
- `*(.text)`: 将所有标记为".text"的输入节拷贝到输出文件中的".text"节中。

源代码解释

```
/* 声明入口点 */
.globl start

/* 定义Multiboot header */
.magic_ITEM_NAME:     /* Multiboot magic number */
    .long   0x1BADB002
```

```
.flags_ITEM_NAME:     /* Multiboot flags */
    .long   0x0
.checksum_ITEM_NAME:  /* Checksum (magic + flags + checksum must equal 0) */
    .long   -(0x1BADB002 + 0x0)

/* .text 段，存放代码 */
.section .text
.code32
start:
    /* VGA 输出 "PB21051012" */
    movl $0x2F422F50, 0xB8000
    movl $0x2F312F32, 0xB8004
    movl $0x2F352F30, 0xB8008
    movl $0x2F302F31, 0xB800C
    movl $0x2F322F31, 0xB8010

    /* 串口输出 "LXH" */
    movw $0x3F8, %dx      /* 把串口地址 0x3F8 装入 DX 寄存器 */
    movb $0x4C, %al       /* 把 'L' 存入 AL 寄存器 */
    outb %al, %dx         /* 输出 AL 寄存器的值到 DX 寄存器指定的串口端口 */
    movb $0x58, %al       /* 把 'X' 存入 AL 寄存器 */
    outb %al, %dx         /* 输出 AL 寄存器的值到 DX 寄存器指定的串口端口 */
    movb $0x48, %al       /* 把 'H' 存入 AL 寄存器 */
    outb %al, %dx         /* 输出 AL 寄存器的值到 DX 寄存器指定的串口端口 */

    /* 停机 */
    hlt

/* 声明代码结束点 */
end:
```

- `.globl start`：定义了一个名为`start`的全局标签，表示这是程序的入口点。
- `.magic_ITEM_NAME`、`.flags_ITEM_NAME`和`.checksum_ITEM_NAME`：这些是Multiboot header的三个字段，用于指示引导程序使用的操作系统类型和加载地址等信息。
- `.section .text`：指定下面的代码将存储在程序的代码段（`.text`）中。
- `movl $0x2F422F50, 0xB8000`：将16进制数0x2F422F50存储到地址0xB8000中，该地址是显存中第一个字符的位置。这里使用了MOV指令将数据从寄存器传输到内存。
- `movw $0x3F8, %dx`：将串口端口地址0x3F8传输到DX寄存器中，用于输出后面的信息到串口。
- `movb $0x4C, %al`：将字符'L'存储到AL寄存器中。
- `outb %al, %dx`：将AL寄存器的值输出到DX寄存器指定的串口端口。
- `hlt`：让CPU暂停执行，直到有中断或者复位信号发生为止。这里的作用是让引导程序停止执行，等待操作系统的启动。
- `end`：表示代码结束。

该段代码在启动地址空间引导程序中位于Bootloader区域

显示效果

- 生成`.bin`文件

- 使用 `qemu-system-i386 -kernel multibootHeader.bin -serial stdio` 来进行运行