卡通人物

描述已自动生成

**操作系统**

**课程实验报告**

|  |  |  |  |
| --- | --- | --- | --- |
| 实验名称 | 操作系统的引导 | | |
| 姓名 | 常添 | 院系 | 计算学部 |
| 班级 | 2203102 | 学号 | 2022111699 |
| 任课教师 | 郑铁然 | 指导教师 | 郑铁然 |
| 实验地点 | G709 | 实验时间 | 2024.12.1 |

****

**实验3：操作系统的引导**

**1 实验目的**

1. 熟悉操作系统的引导过程及其实现原理。
2. 探究 x86 架构计算机的启动机制，掌握 BIOS 加载引导扇区的过程。
3. 通过修改和优化 bootsect.s 程序，深入了解 Linux 0.11 的引导过程并实现简单功能扩展。

**2 实验环境**

* **操作系统**：Ubuntu 22.04
* **开发工具**：as86 和 ld86 用于汇编和链接，dd 用于生成镜像。
* **调试工具**：Bochs 或 QEMU 虚拟机。

**3 实现功能**

1. 编写 bootsect.s 实现引导功能：
   * 显示自定义提示信息：changtian's os is running...
   * 加载 setup.s 代码到内存中并跳转执行。
2. 确保生成的引导扇区文件符合 512 字节的大小限制，并设置引导标志 0xAA55。
3. 使用 BIOS 中断获取硬件参数并打印结果。

**4 代码流程解析**

**4.1 初始化引导扇区**

* **BIOS 加载引导扇区**： 当计算机启动时，BIOS 将硬盘的第一个扇区（512 字节）加载到内存地址 0x7C00，并将控制权交给它。
* **将引导代码移至内存高地址 0x90000**：

mov ax,#BOOTSEG ; 原始引导扇区地址 0x07C0

mov ds,ax ; 设置数据段寄存器为 BOOTSEG

mov ax,#INITSEG ; 目标地址为 0x9000

mov es,ax ; 设置目标段寄存器

mov cx,#256 ; 复制 256 字（512 字节）

sub si,si

sub di,di

rep movw ; 执行批量复制

jmpi go,INITSEG ; 跳转到新的地址执行

**4.2 加载 setup.s**

* **加载扇区至内存**： 使用 BIOS 中断 int 0x13，将 setup.s 加载到内存地址 0x90200。

mov dx,#0x0000 ; 驱动器 0，磁头 0

mov cx,#0x0002 ; 扇区 2，磁道 0

mov bx,#0x0200 ; 加载地址 = 512（INITSEG 偏移量）

mov ax,#0x0200+SETUPLEN ; 服务号 2，加载 4 个扇区

int 0x13 ; 调用 BIOS 读取扇区

jnc ok\_load\_setup ; 如果读取成功，则跳转

* **读取失败重试机制**： 如果读取失败，使用 int 0x13 复位磁盘控制器并重试。

mov dx,#0x0000

mov ax,#0x0000 ; 重置磁盘控制器

int 0x13

jmp load\_setup ; 重新尝试加载

**4.3 显示提示信息**

* **打印自定义信息**： 使用 BIOS 中断 int 0x10，显示提示信息 changtian's os is running...。

mov ah,#0x03 ; 获取光标位置

xor bh,bh

int 0x10

mov cx,#msg1\_len ; 设置字符串长度

mov bx,#0x0007 ; 设置显示属性

mov bp,#msg1 ; 指向字符串地址

mov ax,#0x1301 ; 显示字符串并移动光标

int 0x10

**4.4 跳转到 setup.s**

* **跳转到 setup.s 执行**：

jmpi 0,SETUPSEG ; 跳转到加载的 setup.s 起始地址

**4.5 数据布局**

* **字符串定义**：

msg1:

.byte 13,10

.ascii "changtian's os is running..."

.byte 13,10,13,10

.byte 0 ; 字符串结束符

* **引导标志**： 引导扇区的最后两个字节必须是 0xAA55，用于 BIOS 验证。

.org 508

root\_dev:

.word ROOT\_DEV

boot\_flag:

.word 0xAA55 ; 必须存在的引导标志

运行结果如下，bootsect 显示正确，第一个任务已完成。

文本

描述已自动生成**5 setup.s 的代码解析与补充**

**5.1 修改初始段寄存器**  
设置 es 和 ds 为 cs，确保使用相同的数据段和代码段：

mov ax,cs

mov ds,ax

mov es,ax

**5.2 显示提示信息**  
使用 BIOS 中断 int 0x10 输出 "Now we are in setup..."：

mov ah,#0x03 ; 读取光标位置

xor bh,bh

int 0x10

mov cx,#28 ; 设置字符串长度

mov bx,#0x000c ; 设置显示属性

mov bp,#msg1 ; 设置字符串地址

mov ax,#0x1301 ; 写字符串并移动光标

int 0x10

**5.3 定义字符串**：

msg1:

.byte 13,10

.ascii "Now we are in setup..."

.byte 13,10,13,10

**5.4 build.c 的改造**

找到 build.c 的 argv[3] 检查逻辑部分，并直接返回以跳过 system 的处理：

if (strcmp(argv[3],"none") == 0)

return 0;

文本

描述已自动生成此时运行结果如下，bootsect 正确读入 setup，第二个任务已完成。

**6获取硬件参数**  
以读取内存大小为例，使用 BIOS 中断 int 0x15：

mov ah,#0x88 ; 功能号 0x88，读取内存大小

int 0x15

mov [0x9000],ax ; 将结果存储到内存地址 0x9000

将读取的硬件参数转换为十六进制格式并输出：

print\_memory\_size:

mov ax,[0x9000] ; 从内存地址 0x9000 获取内存大小

call print\_hex ; 调用十六进制打印函数

ret

**6.1 打印十六进制数的函数**：

print\_hex:

push ax

mov cx,#4 ; 4 个十六进制数字

print\_digit:

rol ax,#4 ; 左移 4 位，获取最高 4 位

mov bl,al ; 保存结果到 bl

and bl,#0x0f ; 提取低 4 位

cmp bl,#9

jg hex\_letter

add bl,#0x30 ; 转换为 ASCII 数字字符

jmp output\_digit

hex\_letter:

add bl,#0x37 ; 转换为 ASCII 字母字符

output\_digit:

mov ah,#0x0e

int 0x10

loop print\_digit

pop ax

ret

**6.2 打印信息字符串：print\_msg**

print\_msg:

mov bx, #0x0007 ; 设置显示属性（页面0，普通属性）

mov ax, #0x1301 ; BIOS 中断功能号

int 0x10

ret

**功能**：打印 bp 指向的字符串，字符串长度通过 cx 传入。

**6.3 获取和打印硬件参数**

**获取光标位置**：

mov ah, #0x03 ; 功能号 0x03 表示读取光标位置

xor bh, bh ; 指定页面号为 0

int 0x10

mov [0], dx ; 将光标位置保存到内存地址 ds:0

**打印光标位置**：

mov cx, #14

mov bp, #msg2 ; 光标位置提示信息

call print\_msg

call print\_hex ; 打印光标位置值

call print\_nl ; 换行

**6.4获取和打印扩展内存大小**：

mov ah, #0x88 ; 功能号 0x88 表示读取扩展内存大小

int 0x15

mov [2], ax ; 将扩展内存大小保存到内存地址 ds:2

**打印扩展内存大小**：

mov cx, #15

mov bp, #msg3 ; 内存大小提示信息

call print\_msg

mov dx, [2] ; 读取扩展内存大小

call print\_hex ; 打印内存大小值

call print\_nl ; 换行

**6.5读取磁盘参数表**：

mov ax, #0x000

mov ds, ax

lds si, [4\*0x41] ; 获取第一个磁盘参数表的入口地址

mov ax, #INITSEG

mov es, ax

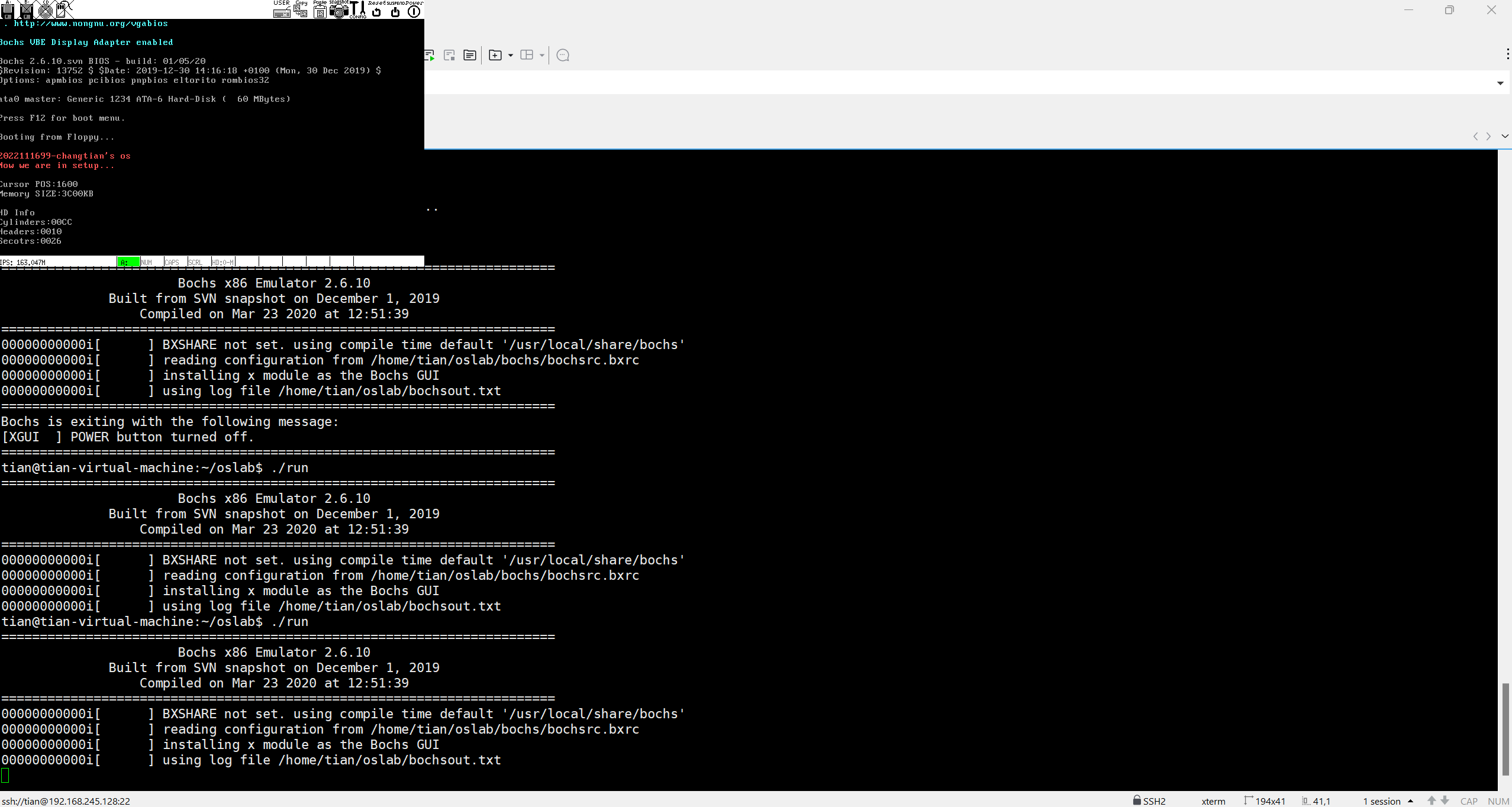
mov di, #0x0004 ; 存储位置 es:0x0004

mov cx, #0x10 ; 复制 16 字节

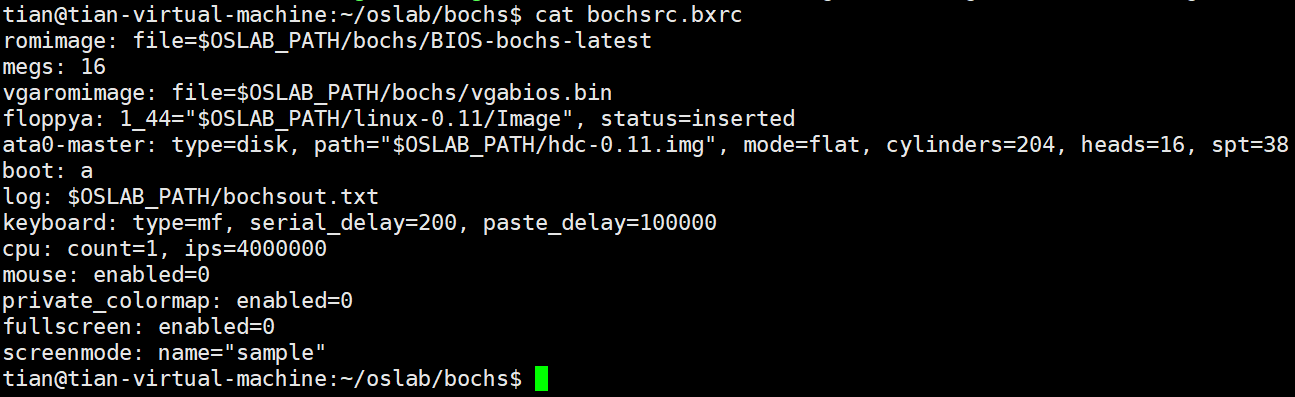
rep

movsb

最终的运行结果如下，setup 获取硬件参数正确，第三个任务已完成。



16进制转换成10进制：0010=16，00CC=204，0026=38



查看bochs/bochsrc.bxrc的文件，二者结果一致，说明setup 正确显示硬件参数。本实验任务完成。