Lab1 系统引导实验报告

姓名：高心尧 学号：221220139 邮箱：1612759455@qq.com

实验进度：完成了所有内容

一、实验要求

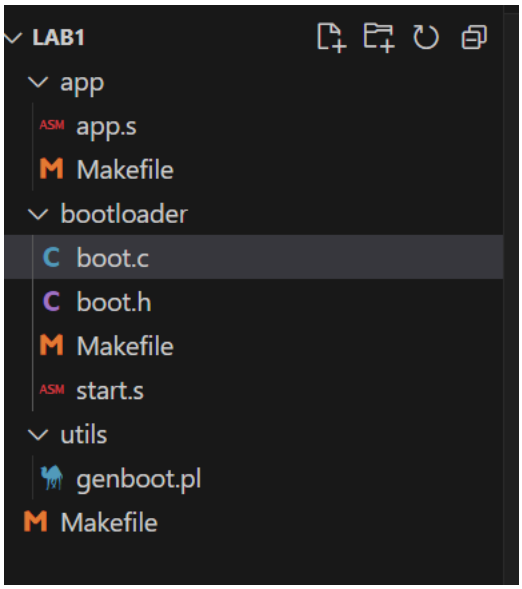
本实验通过实现一个简单的引导程序，介绍系统启动的基本过程

1.1. 在实模式下实现一个 Hello World 程序 在实模式下在终端中打印 Hello, World!

1.2. 在保护模式下实现一个 Hello World 程序 从实模式切换至保护模式，并在保护模式下在终端中打印 Hello, World!

1.3. 在保护模式下加载磁盘中的 Hello World 程序运行 从实模式切换至保护模式，在保护模式下读取磁盘 1 号扇区中的 Hello World 程序至内存中的相应位置， 跳转执行该 Hello World 程序，并在终端中打印 Hello, World!

二、代码框架



主文件夹中的 Makefile 提供了 make os.img,make clean,make play,make debug 这些指令，编译 app.s 生成 app,bin,编译链接 bootloader 中的文件生成 bootloader.bin，将 app.bin 与 bootloader.bin 拼接得到 os.img。本次实验中修改的文件为 boot.c 与 start.s。

三、实验过程

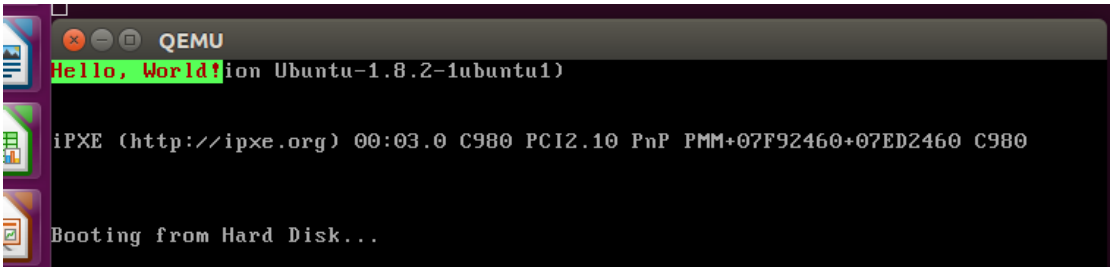
1.1. 该阶段需要修改 start.s 的 start 函数，代码在 index.pdf 中已经给出，解释在注释中。代码如下：

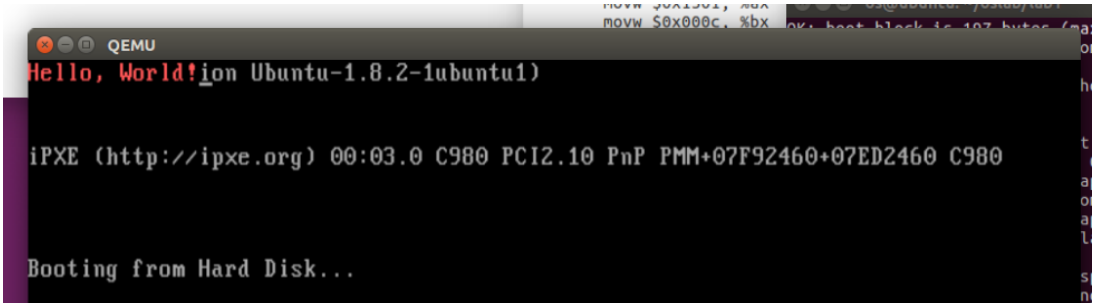
```
# TODO: This is lab1.1
/* Real Mode Hello World */
.code16

.global start
start:
    movw %cs, %ax
    movw %ax, %ds
    movw %ax, %es
```

```
10.     movw %ax, %ss  #在实模式下，这样做是为了确保 ds，es，ss 这些段寄存器指向相同的段，以便程序可以正确访问内存。
11.     movw $0x7d00, %ax
12.     movw %ax, %sp # setting stack pointer to 0x7d00
13.     # TODO:通过中断输出 Hello World
14.     pushw $13 # pushing the size to print into stack
15.     pushw $message # pushing the address of message into stack
16.     callw displayStr # calling the display function
17. loop:
18.     jmp loop  #形成一个无限循环
19.
20. message:
21.     .string "Hello, World!\n\0"  #输出字符串
22.
23. displayStr:
24.     pushw %bp
25.     movw 4(%esp), %ax
26.     movw %ax, %bp   #将字符串首地址存入寄存器 bp
27.     movw 6(%esp), %cx   #将字符串长度 13 存入寄存器 cx
28.     movw $0x1301, %ax  #ax 中 AH 中 0x13 控制输出字符串，AL 中 0x1 控制写模式
29.     movw $0xa4, %bx  #bx 中 BL 高 4 位控制背景颜色，BL 低四位控制字体颜色，当前为绿色背景红色字体（index 中 0xc 为黑色背景红色字体）
30.     movw $0x0000, %dx  #dx 中 DH，DL 控制输出的起始行列
31.     int $0x10  #执行 BIOS 中断 0x10，显示消息。
32.     popw %bp
33.     ret
```

实验结果：





2.2.该阶段修改 start.s 的 start 函数，部分代码在 app.s 中已经给出。

```
# TODO:关闭中断
    cli
```

```
# TODO：设置 CR0 的 PE 位（第 0 位）为 1
    movl %cr0,%eax
    or $0x1,%eax  #cr0 寄存器中的值与 1 进行或运算再存回 cr0 寄存器完成任务
    movl %eax,%cr0
```

```
# TODO:输出 Hello World
    pushl $13  #字符串长度为 13
    pushl $message  #字符串地址
    calll displayStr  #调用函数 displayStr
loop32:
    jmp loop32  #无限循环

message:
    .string "Hello, World!\n\0"

displayStr:
    movl 4(%esp), %ebx  #字符串地址
    movl 8(%esp), %ecx  #字符串的长度 13 存入 ecx，后面 loopnz 循环 13 次
    movl $((80*5+0)*2), %edi  #将屏幕缓冲区的偏移量加载到%edi 寄存器，此处为第 5 行第 0 列
    movb $0x0c, %ah  #AH 中的 0x0c 控制打印的字符串为红色字体黑色背景
nextChar:
    movb (%ebx), %al  #将待打印字符依次存入 AL 中
    movw %ax, %gs:(%edi)  #每个待打印字符两个字节信息，AH 控制颜色 AL 控制字符。将这个待打印字符的信息存入屏幕缓冲区，屏幕缓冲区基址由段寄存器 gs 中的内容得到。
    addl $2, %edi
    incl %ebx  #处理下一个字符
    loopnz nextChar # loopnz decrease ecx by 1
    ret
```

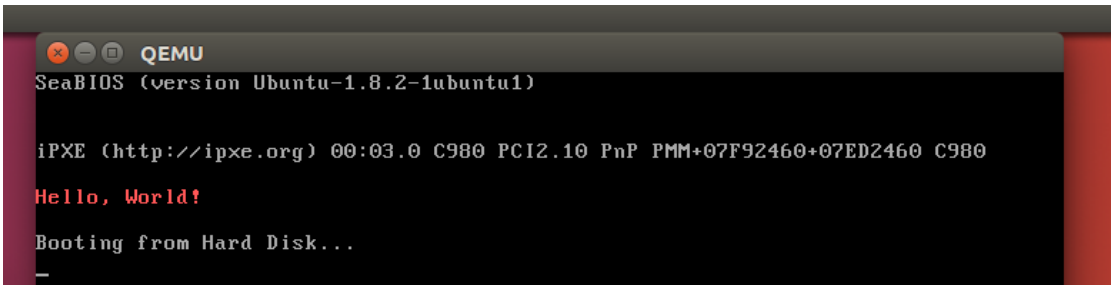
```
# TODO: code segment entry
.word 0xffff,0
.byte 0,0x9a,0xcf,0x00
# TODO: data segment entry
.word 0xffff,0
.byte 0,0x92,0xcf,0
# TODO: graphics segment entry
.word 0xffff,0x8000
.byte 0x0b,0x92,0xcf,0
```

.word limit[15:0],base[15:0]

.byte base[23:16],(0x90|(type)),(0xc0|(limit[19:16])),base[31:24]

这里初始化了描述符的段界限（Limit），段描述符的基址（Base），以及类型（type），代码段可读可执行，数据段可读写，图形段可读写。图形段首地址为 0xb800。

实验结果：



2.3. 该阶段修改 start.s 的 start 函数，与 2.2 基本相同，但是 helloworld 程序的实现通过跳转到 bootMain 函数实现，所以还要完成 boot.c 中 bootMain 函数。

```
1.  void bootMain(void) {
2.      //TODO
3.      void (*app)(void);
4.      app=(void*)0x8c00;
5.      readSect(app, 1);
6.      app();
7.  }
```

阅读 app 中的 Makefile 知道该文件的代码段在 0x8c00 地址处。

该函数首先让函数指针 app 指向 0x8c00 位置，然后将磁盘 1 号扇区的程序 app（输出 helloworld）拷贝到 0x8c00 的位置，最后运行拷贝到该位置的程序，完成实验。

实验结果：

四、实验心得

通过本次实验，我对汇编语言和操作系统有了更深入的了解。本次实验中看完手册我没有任何头绪不知道该从何入手，但是仔细阅读实验文件后，通过 Makefile 文件我大致明白这次实验的结构与逻辑，然后在仔细阅读手册和实验文件后找到了 1.1 和 1.2 所需要的部分代码。这次实验很好锻炼了我阅读理解汇编代码的能力，让我对计算机系统底层原理有了更深入的了解。

同时本次实验报告中，我对代码逐行解释是因为，我通过查资料弄清楚了它们的含义，但是我怕过几天又忘了所以把实验报告当笔记。

五、思考题

1. **你弄清楚本小结标题（CPU、内存、BIOS、磁盘、主引导扇区、加载程序、 操作系统）中各种名词的含义和他们间的关系了吗？请在实验报告中阐述。**

答：

CPU: 中央处理器，负责执行各种指令以及控制其他硬件和软件。

内存: 内存是计算机用来存储数据和程序的地方，用于临时存储正在执行的程序和数据。

BIOS: 基本输入输出系统，负责在系统启动时进行硬件初始化和自检，并加载操作系统。

磁盘: 磁盘是用于永久存储数据的设备，包括硬盘驱动器和固态硬盘等。

主引导扇区:主引导扇区是硬盘上的一个特殊区域，存储着引导加载程序（bootloader）。主引导扇区位于磁盘的第一个扇区，用于引导计算机系统启动过程。在 BIOS 成功加载主引导扇区后，将会运行加载程序。

加载程序: 加载程序是一个小型程序，位于主引导扇区中，负责加载操作系统的核心部分（内核）到内存中，并将控制权转移给内核。

操作系统: 操作系统负责管理内存、文件系统、设备驱动程序等，以及提供用户界面和应用程序执行环境，负责管理计算机的硬件资源并向其他用户程序提供服务。

关系：

CPU 从内存中读取指令和数据，并执行这些指令来完成各种计算和操作。磁盘通常用于存储操作系统、应用程序和用户数据等。在计算机启动时，BIOS 负责初始化硬件，并从磁盘的主引导扇区加载引导加载程序。引导加载程序负责加载操作系统的内核到内存中，并将控制权转移到内核，从而启动操作系统。操作系统在内存中运行，管理系统资源并提供各种服务，使用户能够运行应用程序和执行各种任务。

六、问题

对 readSect 函数仍然不完全理解，看不懂 genboot.pl 文件。