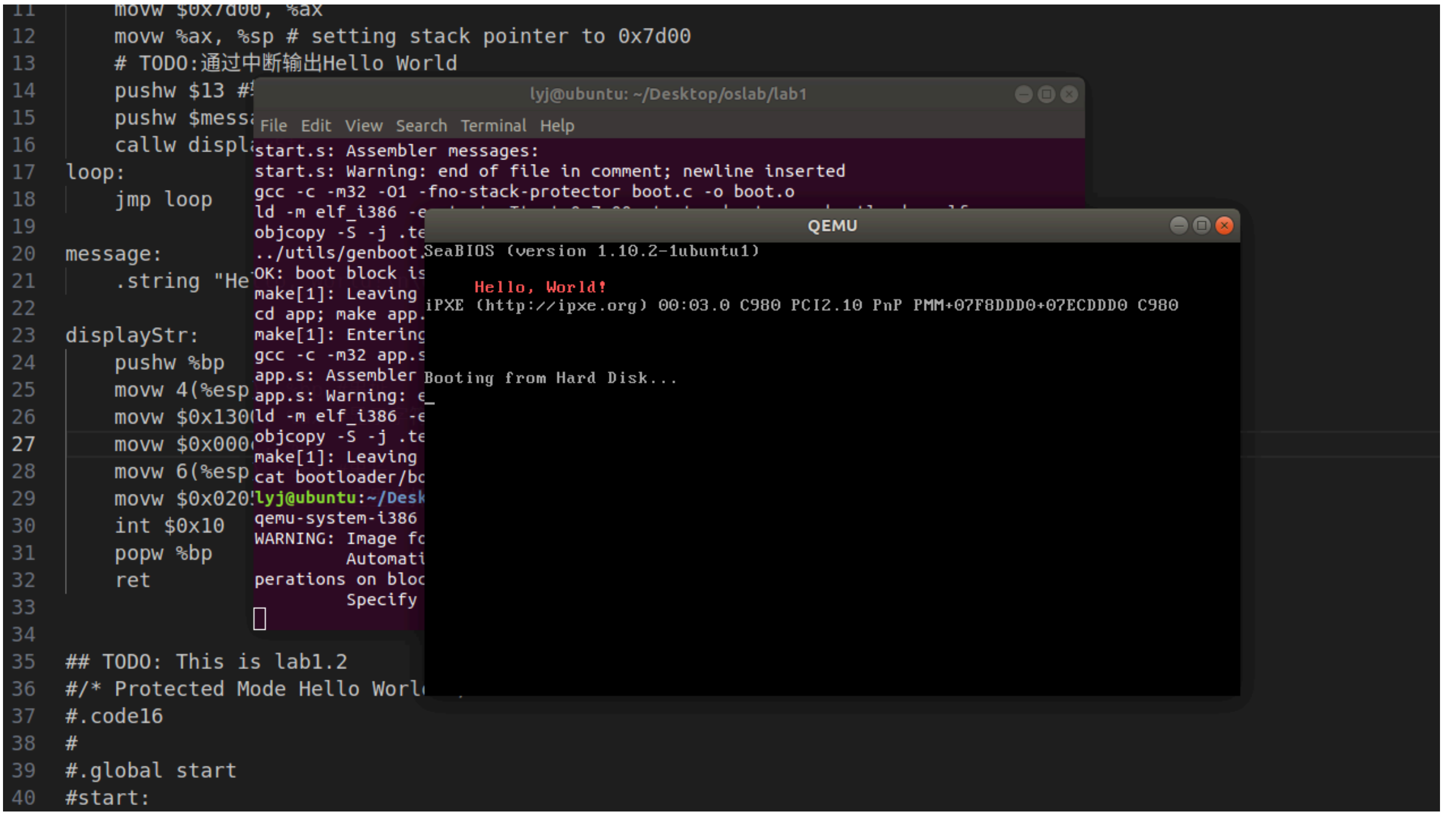
Lab1 实验报告

实验进度

实验结果

1.实模式Hello World程序



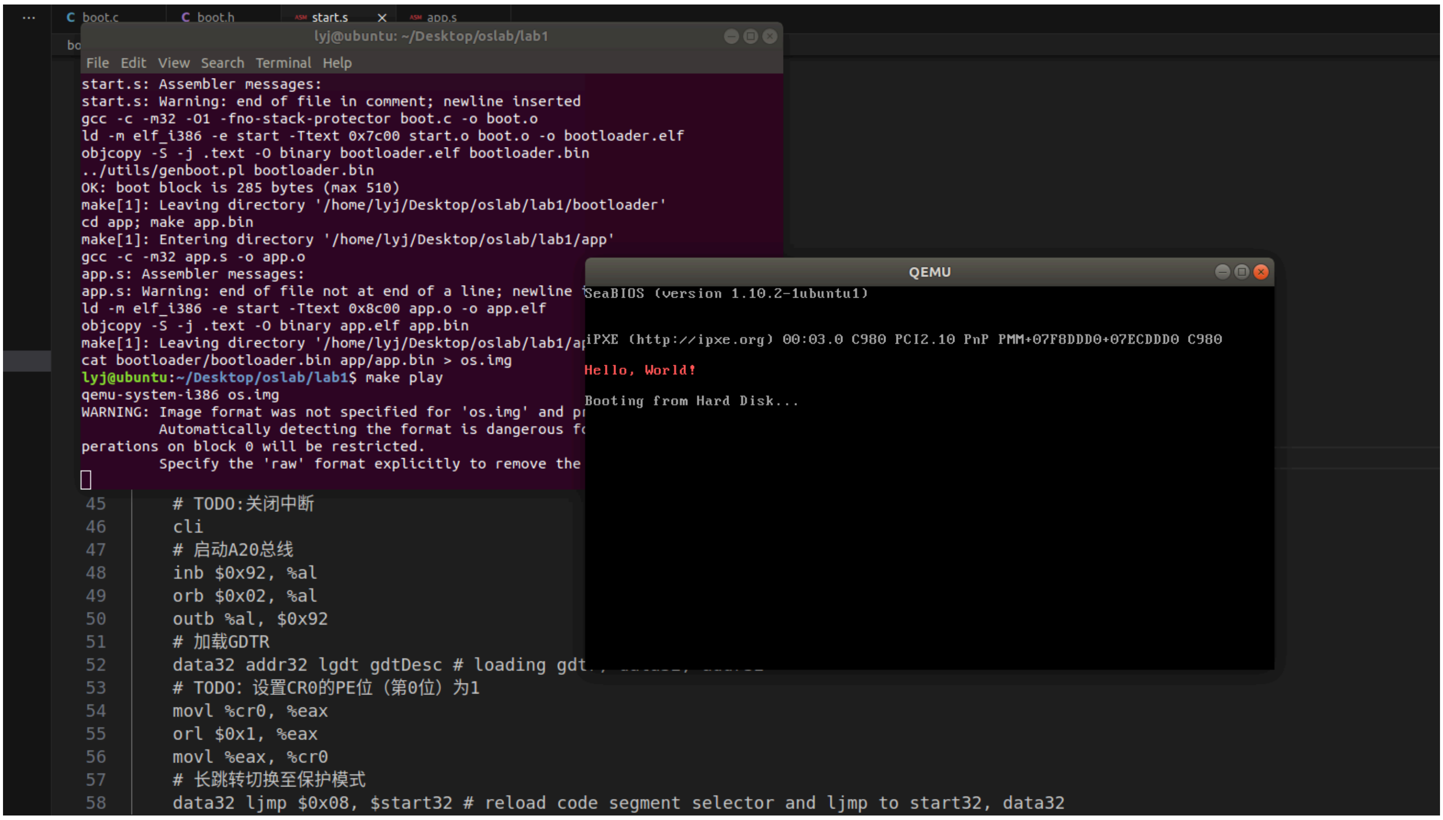
```
    movw $0x7d00, %ax
    movw %ax, %sp # setting stack pointer to 0x7d00
    # TODO:通过中断输出Hello World
    pushw $13 #
    pushw $mess
    callw displ
loop:
    jmp loop

message:
    .string "He

displayStr:
    pushw %bp
    movw 4(%esp
    movw $0x130
    movw $0x000
    movw 6(%esp
    movw $0x020
    int $0x10
    popw %bp
    ret


## TODO: This is lab1.2
#/* Protected Mode Hello Worl
#.code16
#
#.global start
#start:
```

lyj@ubuntu: ~/Desktop/oslab/lab1
File Edit View Search Terminal Help

```
start.s: Assembler messages:
start.s: Warning: end of file in comment; newline inserted
gcc -c -m32 -O1 -fno-stack-protector boot.c -o boot.o
ld -m elf_i386 -e
objcopy -S -j .te
../utils/genboot.
OK: boot block is
make[1]: Leaving
cd app; make app.
make[1]: Entering
gcc -c -m32 app.s
app.s: Assembler
app.s: Warning: e
ld -m elf_i386 -e
objcopy -S -j .te
make[1]: Leaving
cat bootloader/bo
lyj@ubuntu:~/Desk
qemu-system-i386
WARNING: Image fo
         Automati
perations on bloc
         Specify
```

```
SeaBIOS (version 1.10.2-1ubuntu1)

     Hello, World!
iPXE (http://ipxe.org) 00:03.0 C980 PCI2.10 PnP PMM+07F8DDD0+07ECDDD0 C980


Booting from Hard Disk...
_
```

2. 实模式切换保护模式

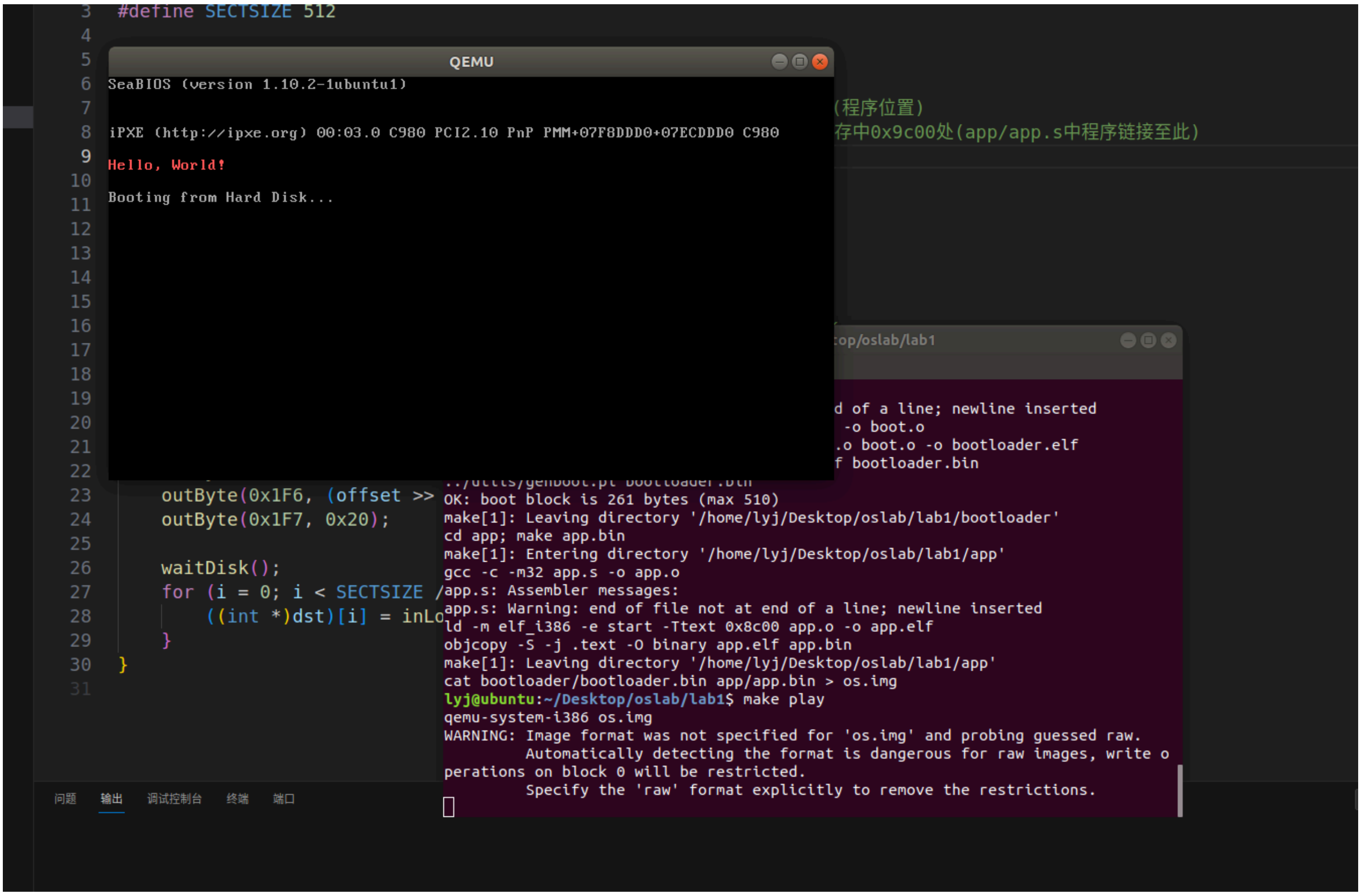

boot.c
boot.h
start.s
app.s
lyj@ubuntu: ~/Desktop/oslab/lab1
File Edit View Search Terminal Help
QEMU

```
start.s: Assembler messages:
start.s: Warning: end of file in comment; newline inserted
gcc -c -m32 -O1 -fno-stack-protector boot.c -o boot.o
ld -m elf_i386 -e start -Ttext 0x7c00 start.o boot.o -o bootloader.elf
objcopy -S -j .text -O binary bootloader.elf bootloader.bin
../utils/genboot.pl bootloader.bin
OK: boot block is 285 bytes (max 510)
make[1]: Leaving directory '/home/lyj/Desktop/oslab/lab1/bootloader'
cd app; make app.bin
make[1]: Entering directory '/home/lyj/Desktop/oslab/lab1/app'
gcc -c -m32 app.s -o app.o
app.s: Assembler messages:
app.s: Warning: end of file not at end of a line; newline
ld -m elf_i386 -e start -Ttext 0x8c00 app.o -o app.elf
objcopy -S -j .text -O binary app.elf app.bin
make[1]: Leaving directory '/home/lyj/Desktop/oslab/lab1/ap
cat bootloader/bootloader.bin app/app.bin > os.img
lyj@ubuntu:~/Desktop/oslab/lab1$ make play
qemu-system-i386 os.img
WARNING: Image format was not specified for 'os.img' and pr
         Automatically detecting the format is dangerous fo
perations on block 0 will be restricted.
         Specify the 'raw' format explicitly to remove the
```

```
SeaBIOS (version 1.10.2-1ubuntu1)

iPXE (http://ipxe.org) 00:03.0 C980 PCI2.10 PnP PMM+07F8DDD0+07ECDDD0 C980

Hello, World!

Booting from Hard Disk...
```

```
    # TODO:关闭中断
    cli
    # 启动A20总线
    inb $0x92, %al
    orb $0x02, %al
    outb %al, $0x92
    # 加载GDTR
    data32 addr32 lgdt gdtDesc # loading gdt
    # TODO: 设置CR0的PE位（第0位）为1
    movl %cr0, %eax
    orl $0x1, %eax
    movl %eax, %cr0
    # 长跳转切换至保护模式
    data32 ljmp $0x08, $start32 # reload code segment selector and ljmp to start32, data32
```

3.加载磁盘中的程序并运行



```
#define SECTSIZE 512
```

```
(程序位置)
存中0x9c00处(app/app.s中程序链接至此)
```

```
SeaBIOS (version 1.10.2-1ubuntu1)

iPXE (http://ipxe.org) 00:03.0 C980 PCI2.10 PnP PMM+07F8DDD0+07ECDDD0 C980

Hello, World!

Booting from Hard Disk...
```

```
    outByte(0x1F6, (offset >>
    outByte(0x1F7, 0x20);

    waitDisk();
    for (i = 0; i < SECTSIZE /
        ((int *)dst)[i] = inLo
    }
}
```

```
d of a line; newline inserted
 -o boot.o
.o boot.o -o bootloader.elf
f bootloader.bin
OK: boot block is 261 bytes (max 510)
make[1]: Leaving directory '/home/lyj/Desktop/oslab/lab1/bootloader'
cd app; make app.bin
make[1]: Entering directory '/home/lyj/Desktop/oslab/lab1/app'
gcc -c -m32 app.s -o app.o
app.s: Assembler messages:
app.s: Warning: end of file not at end of a line; newline inserted
ld -m elf_i386 -e start -Ttext 0x8c00 app.o -o app.elf
objcopy -S -j .text -O binary app.elf app.bin
make[1]: Leaving directory '/home/lyj/Desktop/oslab/lab1/app'
cat bootloader/bootloader.bin app/app.bin > os.img
lyj@ubuntu:~/Desktop/oslab/lab1$ make play
qemu-system-i386 os.img
WARNING: Image format was not specified for 'os.img' and probing guessed raw.
         Automatically detecting the format is dangerous for raw images, write o
perations on block 0 will be restricted.
         Specify the 'raw' format explicitly to remove the restrictions.
```

实验代码修改位置

1.实模式Hello World程序

修改/lab1/bootloader/start.s中代码

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

.global start
start:
        movw %cs, %ax
        movw %ax, %ds
        movw %ax, %es
        movw %ax, %ss
        movw $0x7d00, %ax
        movw %ax, %sp # setting stack pointer to 0x7d00
        # TODO:通过中断输出Hello World
        pushw $13 #输入字符串长度为13
        pushw $message #字符串地址入栈
        callw displayStr
loop:
        jmp loop

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

displayStr:
        pushw %bp
        movw 4(%esp), %bp #串址
        movw $0x1300, %ax #显示字符串模式,光标跟随移动
        movw $0x000c, %bx
        movw 6(%esp), %cx #串长
        movw $0x0205, %dx #这里选择从第2行第5列开始
        int $0x10
        popw %bp
        ret
```

2. 实模式切换保护模式

其中显示"Hello, World!"部分借用app/app.s里的代码

```
# TODO: This is lab1.2
/* Protected Mode Hello World */
.code16

.global start
start:
        movw %cs, %ax
        movw %ax, %ds
        movw %ax, %es
        movw %ax, %ss
        # TODO:关闭中断
        cli
        # 启动A20总线
        inb $0x92, %al
        orb $0x02, %al
        outb %al, $0x92
        # 加载GDTR
        data32 addr32 lgdt gdtDesc # loading gdtr, data32, addr32
        # TODO：设置CR0的PE位（第0位）为1
        movl %cr0, %eax
        orl $0x1, %eax
        movl %eax, %cr0
        # 长跳转切换至保护模式
        data32 ljmp $0x08, $start32 # reload code segment selector and ljmp to start32, data32

.code32
start32:
        movw $0x10, %ax # setting data segment selector
        movw %ax, %ds
        movw %ax, %es
        movw %ax, %fs
        movw %ax, %ss
        movw $0x18, %ax # setting graphics data segment selector
        movw %ax, %gs

        movl $0x8000, %eax # setting esp
        movl %eax, %esp
        # TODO:输出Hello World
        pushl $13 #字符串长
        pushl $message #字符串地址入栈
        call displayStr

loop32:
        jmp loop32

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

#displayStr仿照app.s里的写法
displayStr:
        movl 4(%esp), %ebx
        movl 8(%esp), %ecx
        movl $((80*5+0)*2), %edi
        movb $0x0c, %ah
nextChar:
        movb (%ebx), %al
        movw %ax, %gs:(%edi)
        addl $2, %edi
        incl %ebx
        loopnz nextChar # loopnz decrease ecx by 1
        ret



.p2align 2
gdt: # 8 bytes for each table entry, at least 1 entry
        # .word limit[15:0],base[15:0]
        # .byte base[23:16],(0x90|(type)),(0xc0|(limit[19:16])),base[31:24]
        # GDT第一个表项为空
        .word 0,0
        .byte 0,0,0,0

        # TODO：code segment entry
        .word 0xffff, 0
        .byte 0, 0x9a, 0xcf, 0 #type为代码段,(1010B)，可读，未被访问。段限为fffffH,即最大段限

        # TODO：data segment entry
        .word 0xffff, 0
        .byte 0, 0x92, 0xcf, 0 #type为数据段,(0010B),可读可写未被访问。段限为fffffH,即最大段限

        # TODO：graphics segment entry
        .word 0xffff, 0x8000
        .byte 0x0b, 0x92, 0xcf, 0 #视频段基址为0xb8000。段限为fffffH,即最大段限

gdtDesc:
        .word (gdtDesc - gdt -1)
        .long gdt
```

3.加载磁盘中的程序并运行

start.s:

```
#TODO: This is lab1.3
/* Protected Mode Loading Hello World APP */
.code16

.global start
start:
        movw %cs, %ax
        movw %ax, %ds
        movw %ax, %es
        movw %ax, %ss
        # TODO:关闭中断
        cli

        # 启动A20总线
        inb $0x92, %al
        orb $0x02, %al
        outb %al, $0x92

        # 加载GDTR
        data32 addr32 lgdt gdtDesc # loading gdtr, data32, addr32

        # TODO：设置CR0的PE位（第0位）为1
        movl %cr0, %eax
        orl $0x1, %eax
        movl %eax, %cr0

        # 长跳转切换至保护模式
        data32 ljmp $0x08, $start32 # reload code segment selector and ljmp to start32, data32

.code32
start32:
        movw $0x10, %ax # setting data segment selector
        movw %ax, %ds
        movw %ax, %es
        movw %ax, %fs
        movw %ax, %ss
        movw $0x18, %ax # setting graphics data segment selector
        movw %ax, %gs

        movl $0x8000, %eax # setting esp
        movl %eax, %esp
        jmp bootMain # jump to bootMain in boot.c

.p2align 2
gdt: # 8 bytes for each table entry, at least 1 entry
        # .word limit[15:0],base[15:0]
        # .byte base[23:16],(0x90|(type)),(0xc0|(limit[19:16])),base[31:24]
        # GDT第一个表项为空
        .word 0,0
        .byte 0,0,0,0

        # TODO：code segment entry
        .word 0xffff, 0
        .byte 0, 0x9a, 0xcf, 0 #type为代码段,(1010B)，可读，未被访问。段限为fffffH,即最大段限

        # TODO：data segment entry
        .word 0xffff, 0
        .byte 0, 0x92, 0xcf, 0 #type为数据段,(0010B),可读可写未被访问。段限为fffffH,即最大段限

        # TODO：graphics segment entry
        .word 0xffff, 0x8000
        .byte 0x0b, 0x92, 0xcf, 0 #视频段基址为0xb8000。段限为fffffH,即最大段限

gdtDesc:
        .word (gdtDesc - gdt - 1)
        .long gdt
```

boot.c:

```
void bootMain(void) {
        //TODO
        void (*app)(void) = (void(*)(void))0x8c00;//app函数指向0x8c00处(程序位置)
        readSect((void*)app, 1);
        //读取磁盘第1扇区中的Hello, World!程序至内存中0x9c00处(app/app.s中程序链接至此)
        app();//执行程序
}
```

自由报告

本次实验过程中遇到了一个问题——段限应该是多少。为此翻80386手册翻了n久没找到(应该是有的，可能只是我没找到)。之后突然想起来上学期学ics的时候讲过lilux内存管理采用段页式，但分段是个“假的”分段。代码段、数据段都是取最大的，故为fffffH。

本次实验让我更深入地体会到了系统启动第一步引导程序是如何运行的。