从实模式进入保护模式 实验报告

PB15000102 王嵩超

如何在实模式利用BIOS实现任意键的获取

通过BIOS中断调用int 0x16。

首先将ax寄存器的高位赋为0，再引发中断即可。

```
readkey:
    movb $0, %ah # function code: read a character
    int $0x16 # read a key
```

Protection Mode

来自维基百科的定义：

In computing, **protected mode**, also called **protected virtual address mode**, is an operational mode of x86-compatible CPUs. It allows system software to use features such as virtual memory, paging and safe multi-tasking designed to increase an operating system's control over application software.

保护模式与实模式最大的不同是：各个段的信息存储在GDT（Global Descriptor Table）中，每个段寄存器仅提供段的索引。

What to do first

1. 关中断。（在建好中断描述表IDT之后才可打开，但本实验未涉及此）
2. 将内存中准备好GDT表。在内存中建立一块特定格式的指向该GDT表的指针，用LGDT指令将该指针内容送至GDTR寄存器。

   > 在此步之前需要做的是将GDT表的位置填入该指针。也可以在汇编前就计算好。

3. 更改CR0寄存器，使其最后一位置1，这就打开了保护模式的开关。
4. 通过jmp指令更改CS寄存器，跳转至32位代码执行。
5. 在保护模式下初始化各段寄存器。本实验中需要初始化数据段寄存器（用于取得数据段内的字符常量）、和指向显存区的段寄存器（GDT表里设有指向显存的段）。

实模式下利用BIOS实现VGA显示

首先把字符串地址存入si寄存器，再反复用lodsb指令装载字符串，使用BIOS1中断int 0x10，即可在光标的显示字符。

代码如下：

```
disp:
    movw    $0x0001, %bx
    movb    $0xe, %ah # This means writing characters in TTY(Teletype output) mode
    int $0x10       /* display a byte */
message: #BIOS text output
    lodsb
    cmpb $0, %al
    jne disp
    ret

```

不能在保护模式下利用BIOS实现IO。理由是：保护模式下BIOS中断不起作用。

保护模式写VGA缓存与实模式有何不同？

主要的不同是：保护模式中段寄存器所存的是一个索引，该索引指向显存所在的段。

实模式中，段寄存器直接存储显存所在段的段地址。

回车功能的实现

上一行末尾的显存地址（以开头为0，线性往后）：

2*(11*80+14) 即11行14列

其中乘2是因为每两个字节表示一位地址。（两个字节分别说明属性、ASCII码值）

则换行后的显存地址：

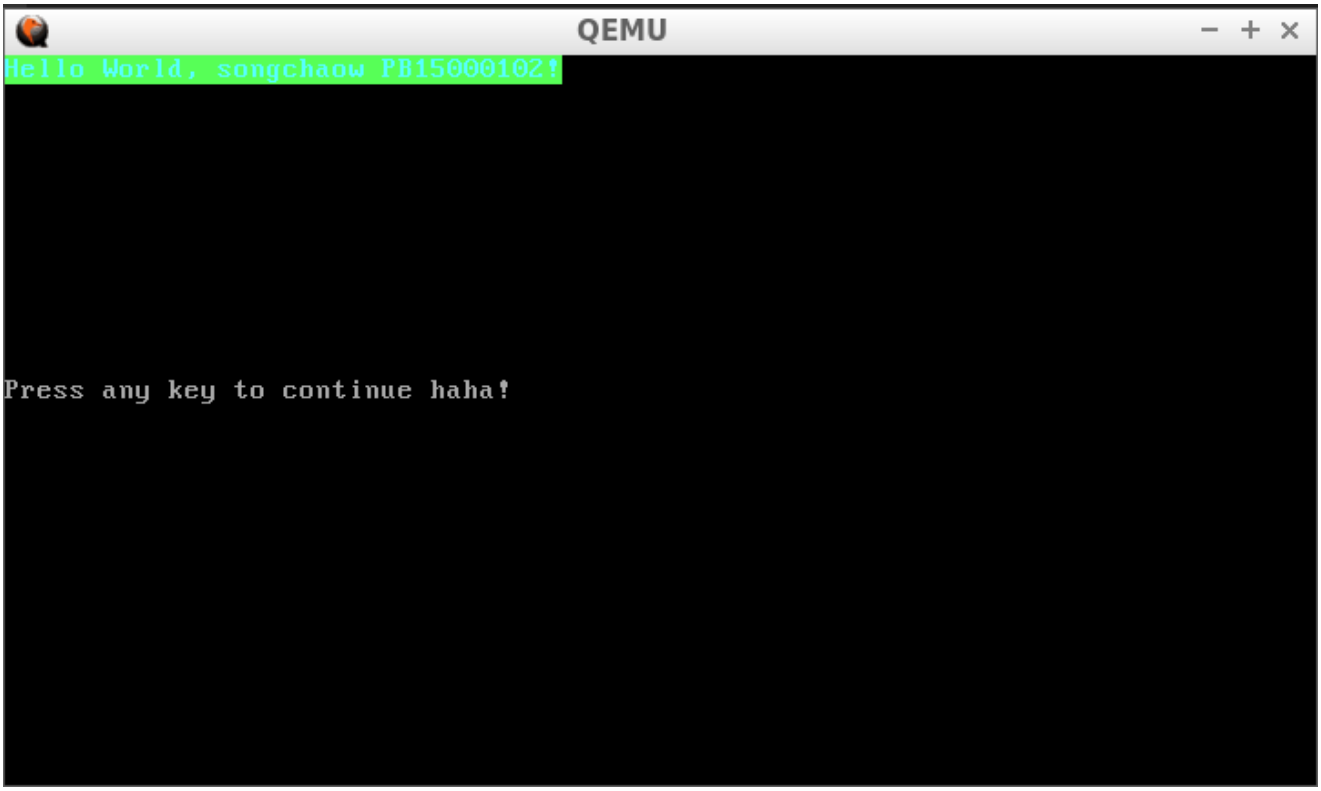
2*(12*80)

汇编代码的输出部分如下：

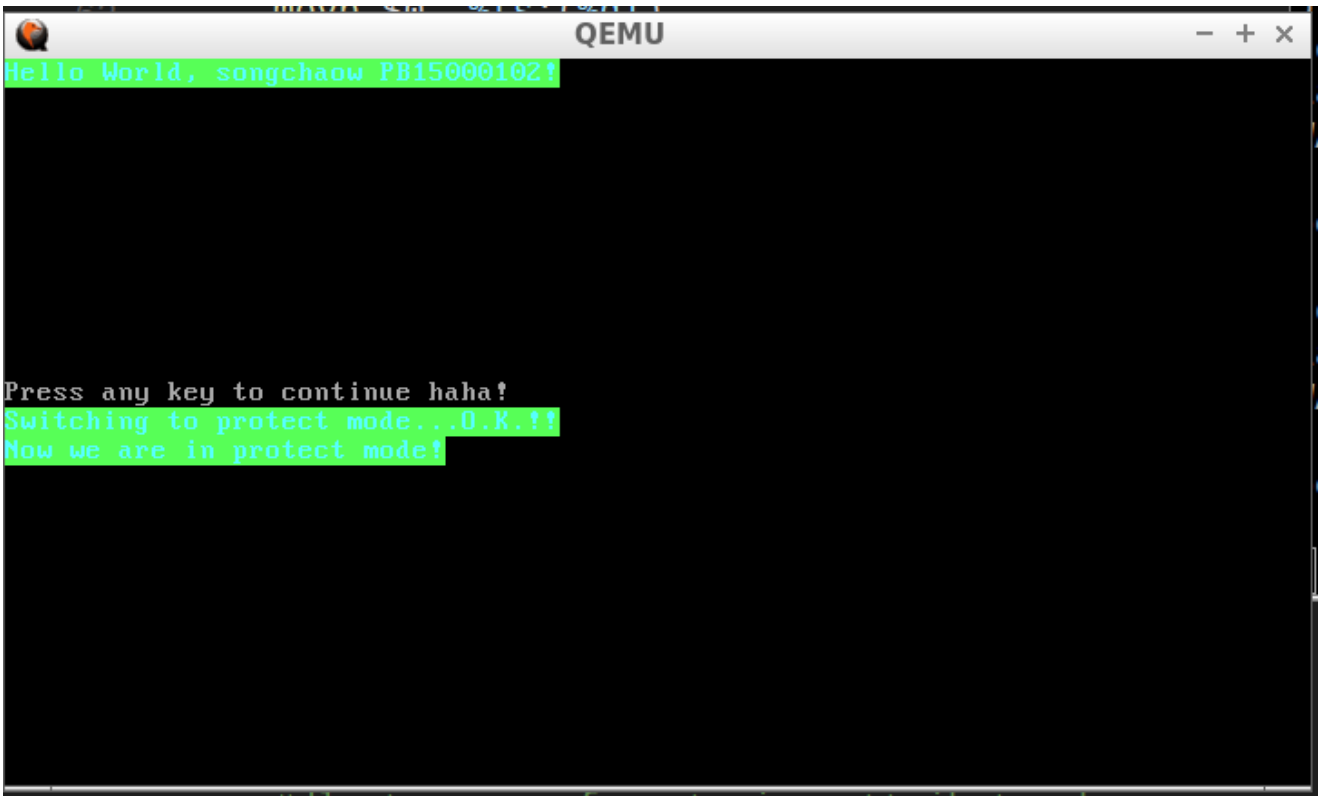
```
movl $(2*(12*80)),%edi
movw $okdisp, %si
call writetovideo_mem
#a new line and output prompt 2
movl $(2*(13*80)),%edi
movw $succ, %si
call writetovideo_mem

```

运行截图

在等待任意键：



保护模式切换：



代码流程图

输出helloworld字符串->显示Press any key to continue，等待输入->显示进入保护模式->进入保护模式，步骤按照前述，之后初始化各段寄存器->进入死循环结束