

本科生实验报告

实验课程:\_\_\_\_\_\_\_\_\_\_\_操作系统实验 \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

实验名称:\_\_\_\_\_\_\_\_\_\_\_\_\_\_lab3 \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

专业名称:\_\_\_\_\_\_\_\_\_\_\_\_ 计科 \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

学生姓名:\_\_\_\_\_\_\_\_\_\_ \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

学生学号:\_\_\_\_\_\_\_\_\_\_ \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

实验地点:\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

实验成绩:\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

报告时间:\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

Assignment1

1. **实验要求**
2. 复现Example 1
3. 将LBA28读取硬盘的方式换成CHS读取，同时给出逻辑扇区号向CHS的转换公式。最后说说你是怎么做的并提供结果截图。
4. **实验过程**
5. 复现example1加载bootloader
   1. 定义bootloader文件：打印字符串
   2. 编写mbr文件：
      1. 定义从硬盘中读取一个逻辑扇区的函数（asm\_read\_disk）: 将要读取逻辑扇区号28位写入0X1F3-0X1F6端口（8bit）；将读取的扇区数量写入0X1F2端口；将硬盘读请求（0X20）写入0X1F7端口；判断状态（wait）；从0X1F7端口（16bit）连续读取一个扇区数据出来（readw）
      2. 在循环中读取五个扇区到bootloader的起始地址（0x7e00）
      3. 跳转到bootloader
6. CHS方式读取硬盘：
7. **LBA和CHS的转化公式：**
8. **CHS->LBA:**

**LBA（逻辑扇区数）=磁头数\*每磁道的扇区数\*当前所在柱面号+每磁道扇区数\*当前所在磁头号+当前所在扇区数-1**

1. **LBA->CHS:**

* **柱面号：LBA/(每个磁道的扇区数\*每个柱面的磁头数)**

**补充：每个磁道的扇区数\*每个柱面的磁头数**

**=每个柱面的扇区数**

* **磁头号：（LBA/每个磁道的扇区数）%每个**

**柱面的磁头数**

**补充：LBA/每个磁道的扇区数=当前扇区前面**

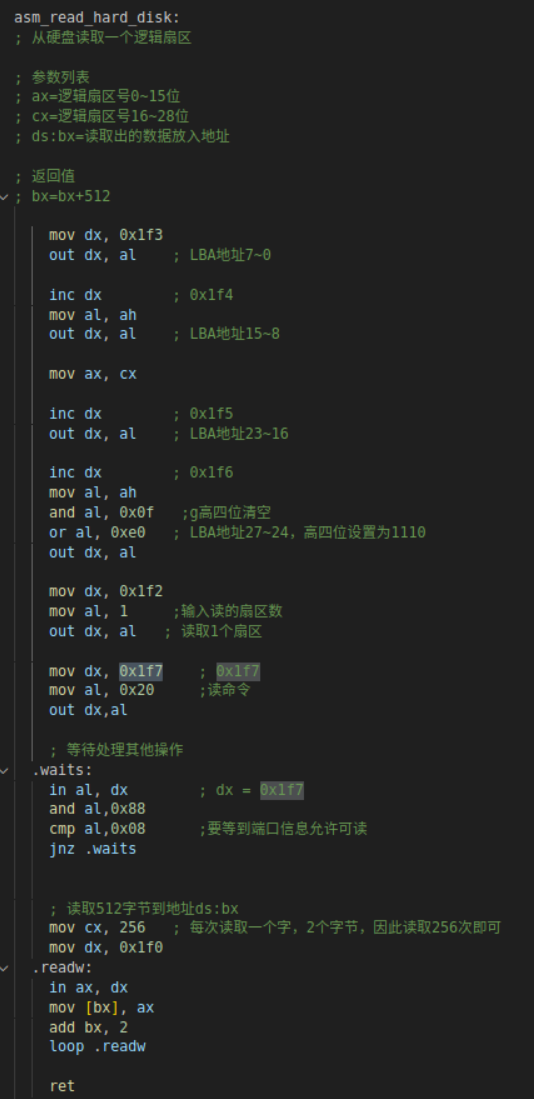
**包含了多少个磁道**

* **扇区号：LBA%每个磁道的扇区数+1**

1. 调用**int 30h中断，02功能号**读五个扇区（具体细节看关键代码实现）
2. **关键代码**
   1. 复现example1：

Mbr的编写：

Asm\_read\_hard\_disk:

­

* 1. CHS读硬盘：

1. **要读取硬盘的扇区LBA号为1-5，则可以根据公署算出，要从柱面号为0，磁道号为0，扇区号为2（编号从1开始）开始读取5个扇区**
2. **Int 13h的参数（寄存器号/内容/注释/）：**bx/0x7e00/指向要接受的扇区的内容存在内存区的起始地址；

al/5/读入扇区数；

ch/0/磁道号；

cl/2/起始扇区号；

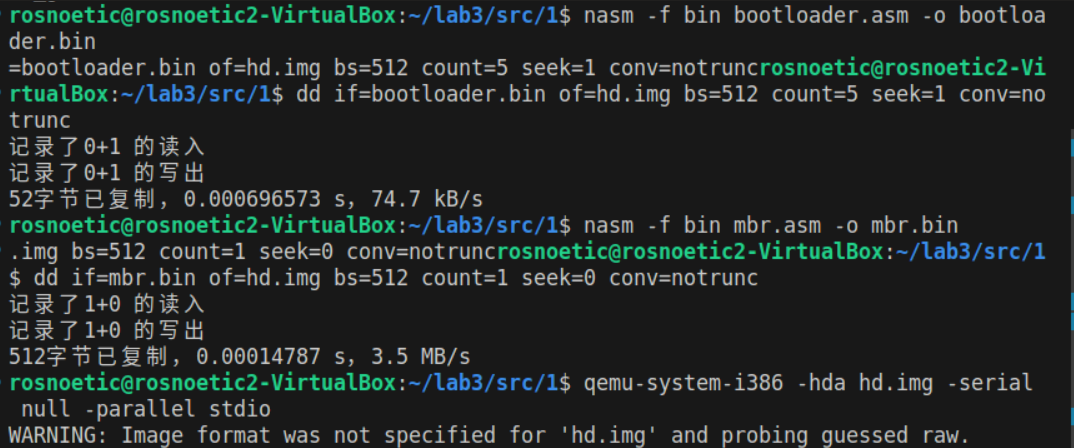
dh/0/磁头号；

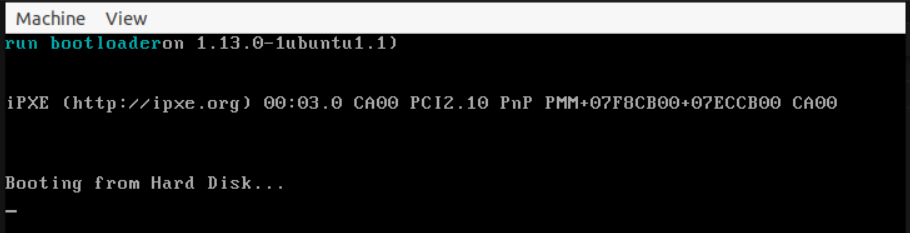
dl/80h/驱动器号：硬盘；ah/2/读硬盘的功能号

注意不需要循环读五个扇区，直接连续读五个扇区

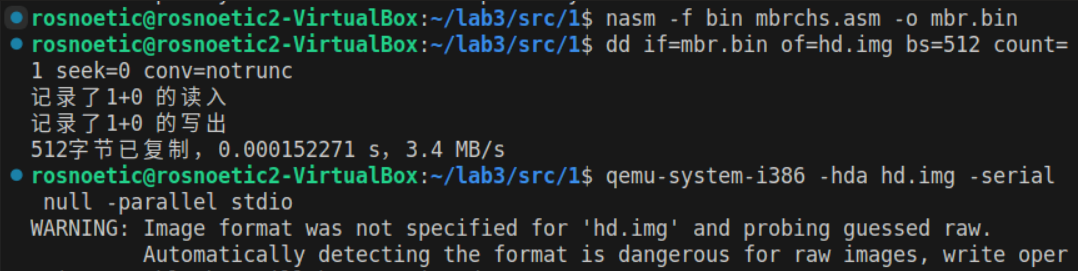
****

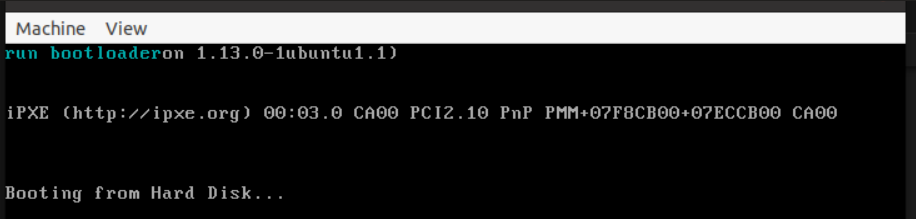
1. **实验结果**
2. 复现eaxmple1：





1. CHS读硬盘：





1. **总结**

问题：CHS读取硬盘时在函数里，尝试将bx赋值为

0x7e00，且将bx置为0（在int 13h中断02功能号中，es:bx指向接收/读写扇区的内存区起始地址。）:出现错误，无法正确读取硬盘

原因：不能随便修改es寄存器，直接将bx赋值为

0x7e00即可

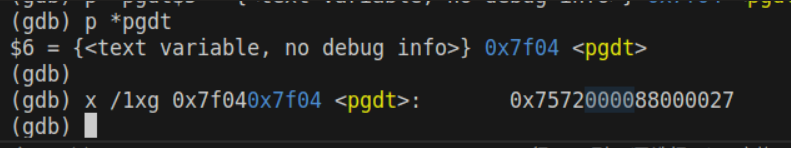
Assignment2：

1. **实验要求**

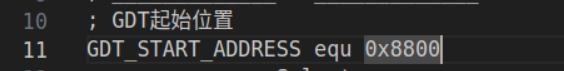
复现Example 2，使用gdb或其他debug工具在进入保护模式的4个重要步骤上设置断点，并结合代码、寄存器的内容等来分析这4个步骤

1. **实验过程：**
2. **准备GDT，用lgdt指令加载GDTR信息。**
   1. **创建空描述符**
   2. **创建数据段描述符**
   3. **创建堆栈描述符**
   4. **创建显存描述符**
   5. **创建平坦模式下的代码段描述符**
3. **打开第21根地址线。**
4. **开启cr0的保护模式标志位。**
5. **远跳转，进入保护模式。**
6. **关键代码**
   1. **准备GDT，用lgdt指令加载GDTR信息。**
7. **在lgdt[pgdt]设置断点，查看pgdt（GDTR）初始化的内容：88000027(只看32位)**

* **全局描述符表基地址：0x8800（头文件定义）**
* **全局描述符表界限：27h=39个字节（前面5个描述符，每个是64bit8个字节，则GDT长度为40个字节，界限=长度-1=39）**







1. **在in al，0x92处设置断点，查看GDT的五个描述符的加载情况**

* **0x8800：空描述符，64bit全为0**
* **0x8808：数据段描述符，0x00cf92000000ffff**

**基地址为0x00，段界限为0xFFFFF**

**粒度为4KB**

* **0x8810：堆栈段描述符，0x040960000000000**
* **0x8818显存段描述符，0x0040920b80007fff**

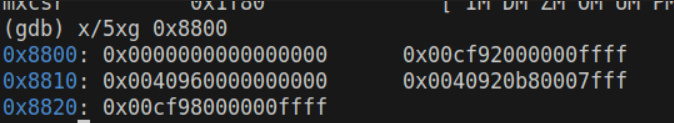
**基地址为0x000B8000，界限0x07FFF**

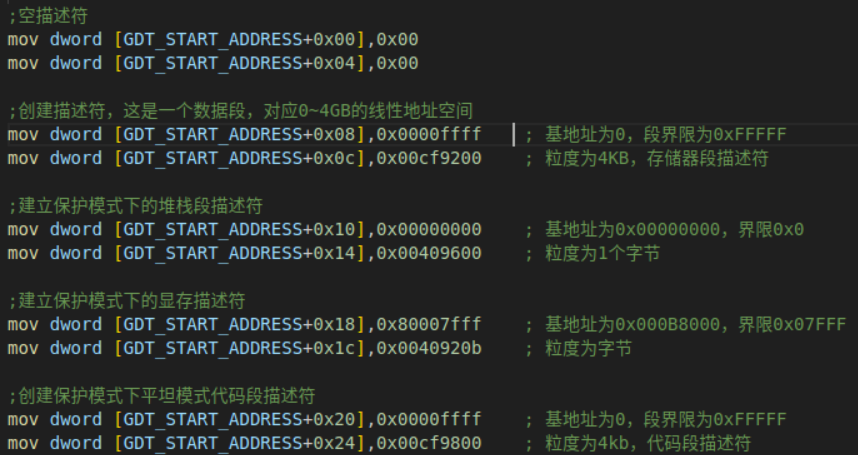
**粒度为字节（高32bit的23位为0）**

* **0x8820：代码段描述符，0x00cf98000000ffff**

**基地址为0，段界限为0xFFFFF**

**粒度为4kb**



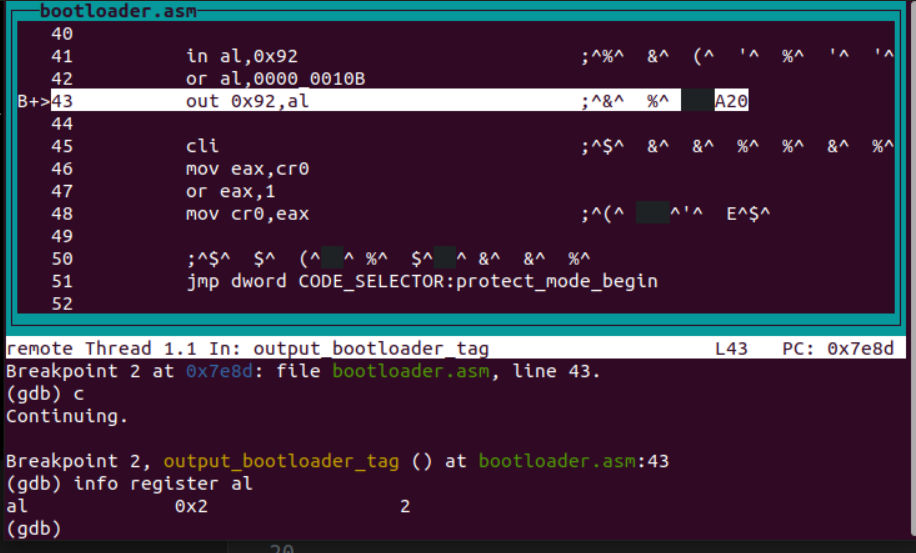


* 1. **打开第21根地址线：**

打开0x92端口的低位第二位bit（置1）

在out 0x92,al处设置断点，查看要写入0x92端口的al值：0x2 打开低位第二位bit

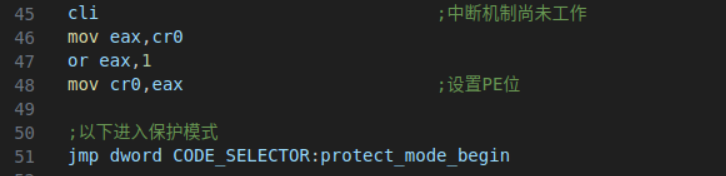


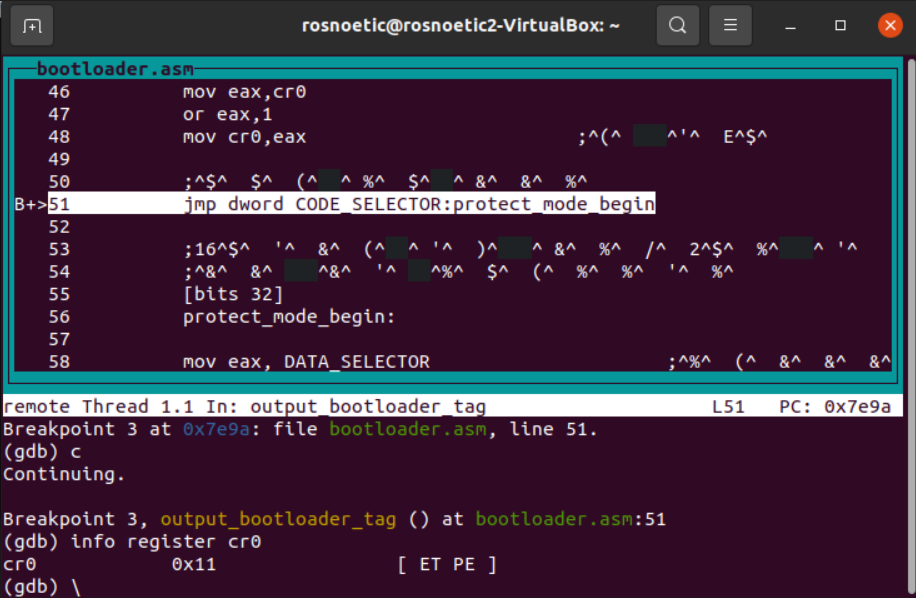


1. **开启cr0的保护模式标志位。**

**设置cr0的PE位为1，进入保护模式**

**在跳转到保护模式的代码处设置断点，查看cr0值，其中最低位PE位为1**





1. 远跳转，进入保护模式

进入后查看所有寄存器，看寄存器中是否正确加载段选择子（低三位RPL和TI都为0）

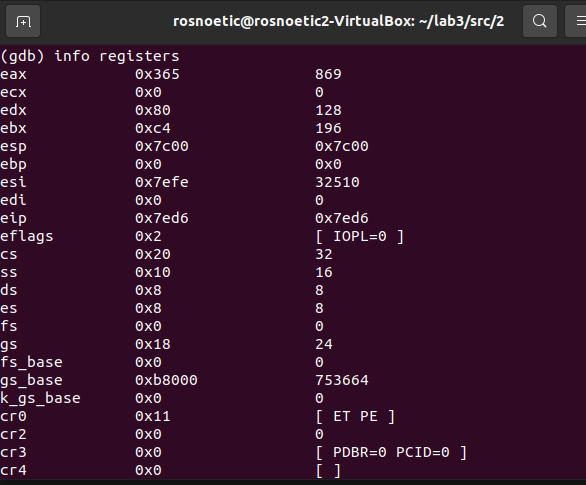
ds：代码段选择子，第一个段描述符，则内容为1000b

ss：堆栈段选择子，第二个段描述符，内容为10000b

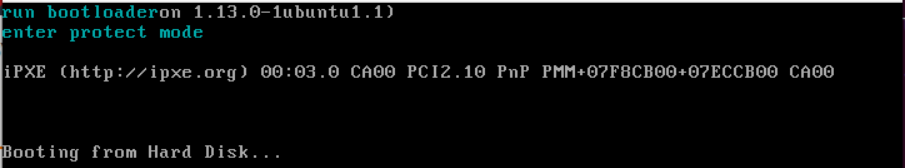
gs：显存段描述符，第三个段描述符，内容为11000b

gs\_base:成功加载显存段的基地址：0xb8000





1. **实验结果**

****

1. **总结**

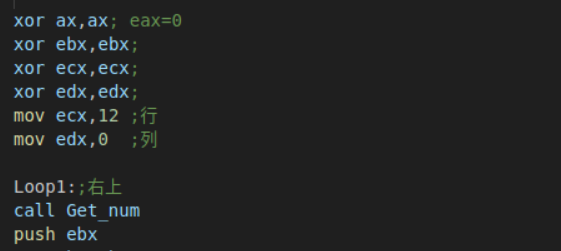
尚未找到如何在gdb调试中查看端口值的方法

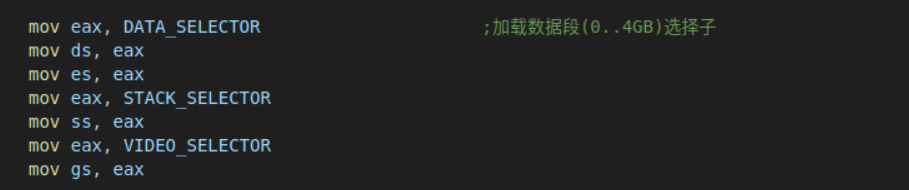
Assignment3

1. **实验要求**

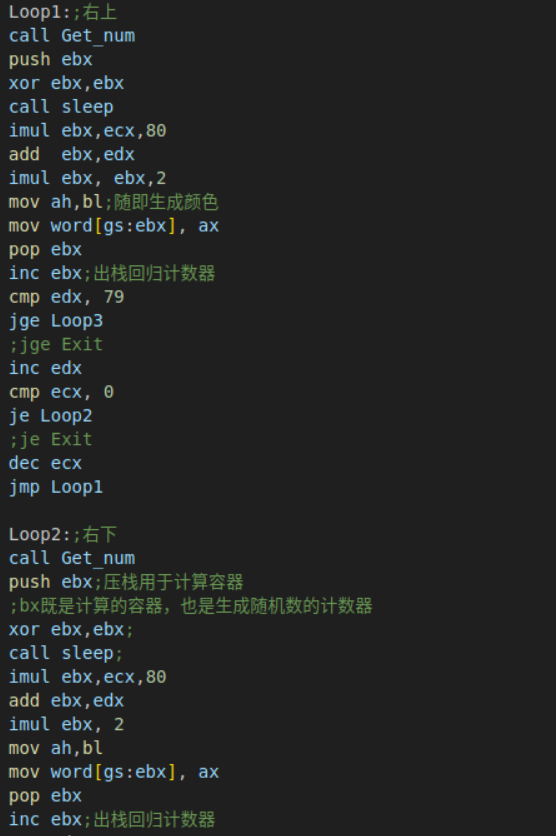
改造“Lab2-Assignment 4”为32位代码，即在保护模式后执行自定义的汇编程序。

1. **实验过程**
2. 复现字符弹射程序的代码
3. 根据32bit的特性修改代码
   1. 将用到的寄存器根据具体情况改为32bit，如bx改成ebx（避免未用到的高位16bit在运算过程中对低位16bit产生奇怪的影响，使用32bit安全）
   2. 寄存器的初始化：gs寄存器不需要特地赋值，前面的部分已经为gs赋值显存段选择子，其他段寄存器ss，es等不需要重新初始化
4. **关键代码**
5. 寄存器初始化：将使用的寄存器初始化而不需要再初始化段寄存器，前面已经为段寄存器赋值了段选择子

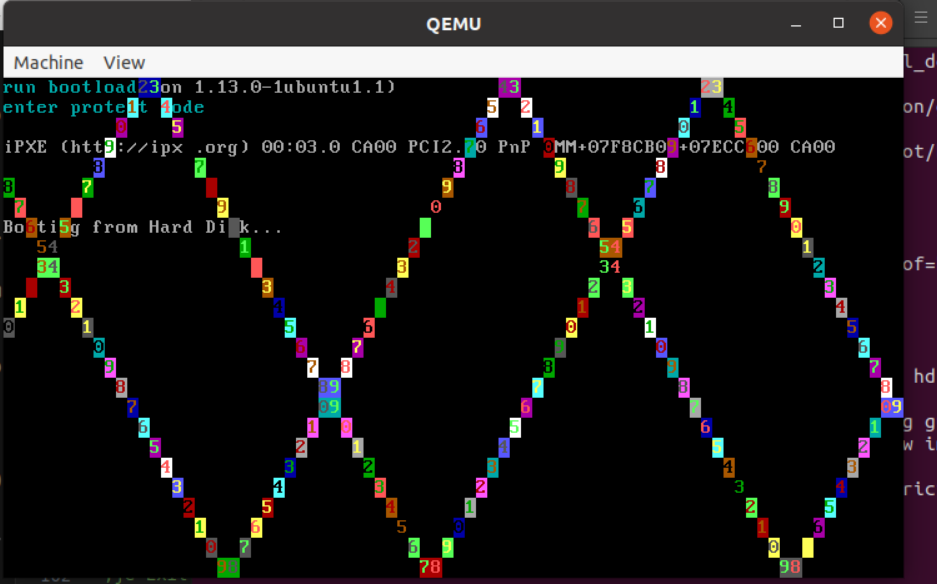




1. 将用到的寄存器，除了al，ah必须为8bit的，其他改为32bit



1. **实验结果**



1. **总结**

问题：将字符弹射程序放在字符串标识定义下面时，无法正确显示，经过gdb调试，在跳转到保护模式时，无法找到保护模式代码段的位置

解决：数据段和代码段不能混着放在一起，不然会引起错误

