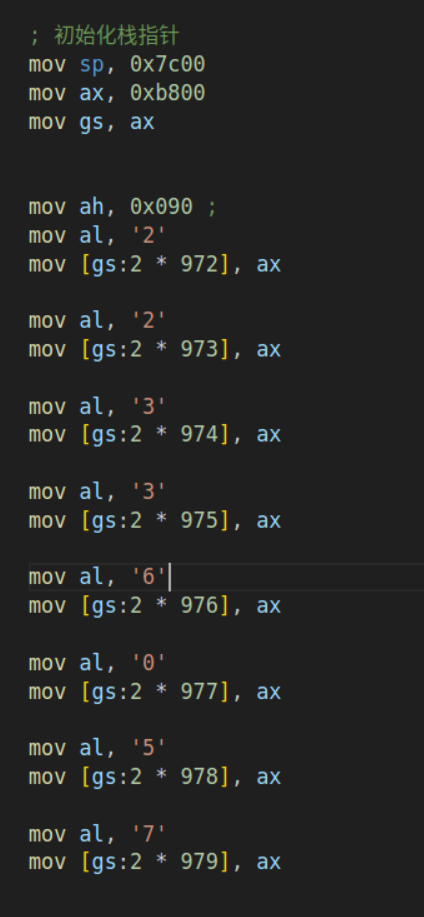


本科生实验报告

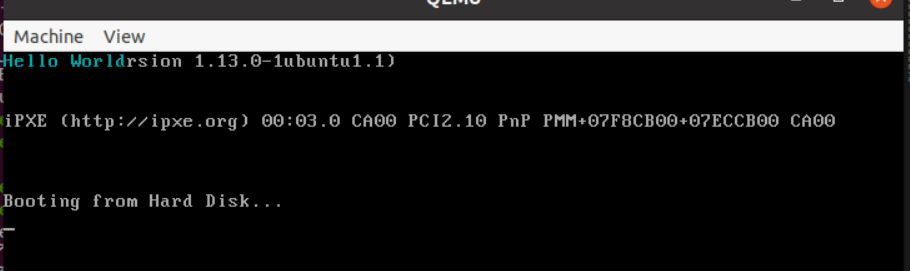
1. **Assignment 1**
2. **实验要求**
3. 复现example 1。说说你是怎么做的，并将结果截图。
4. 请修改example 1的代码，使得MBR被加载到0x7C00后在(12,12)(12,12)处开始输出你的学号。注意，你的学号显示的前景色和背景色必须和教程中不同。说说你是怎么做的，并将结果截图。
5. **实验过程**
6. **编写汇编代码**
7. **将汇编代码编译成二进制文件，写入虚拟硬盘中**
8. **启动qemu模拟器，执行mbr程序（流程同assignment2）**
9. **关键代码**

**学号显示在（12，12）：偏移量=2\*（80\*12+12）=2\*972**

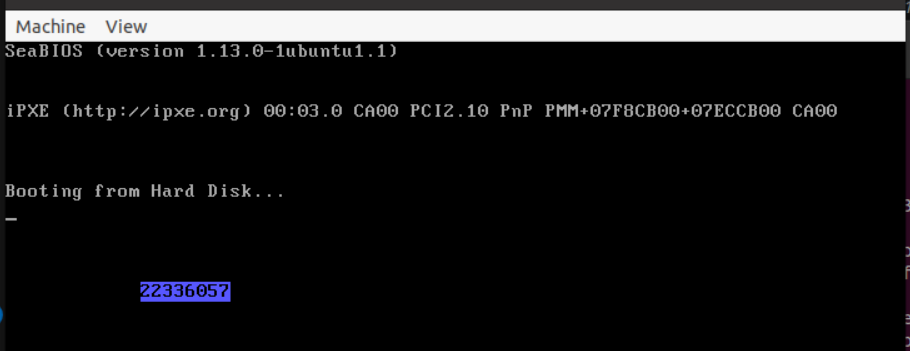
****

1. **实验结果**

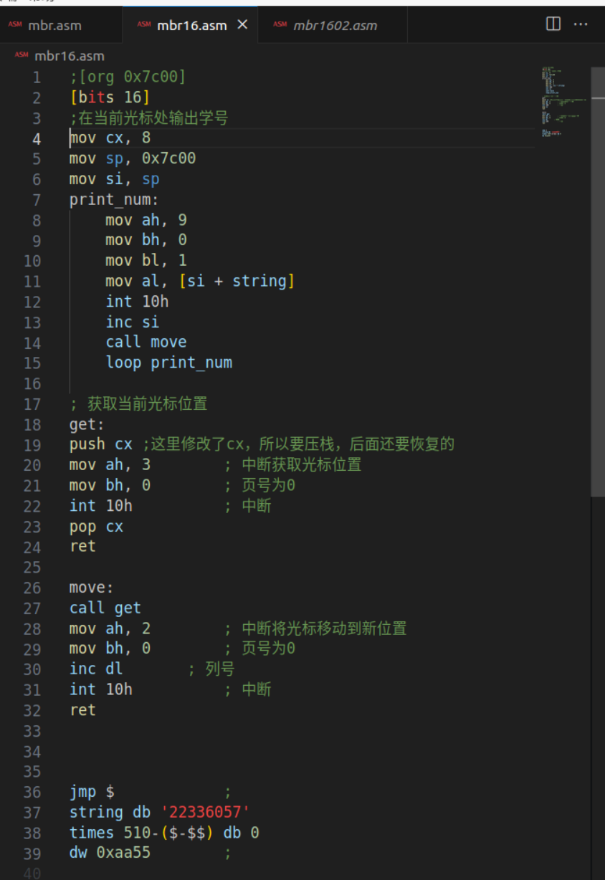
**复现：**

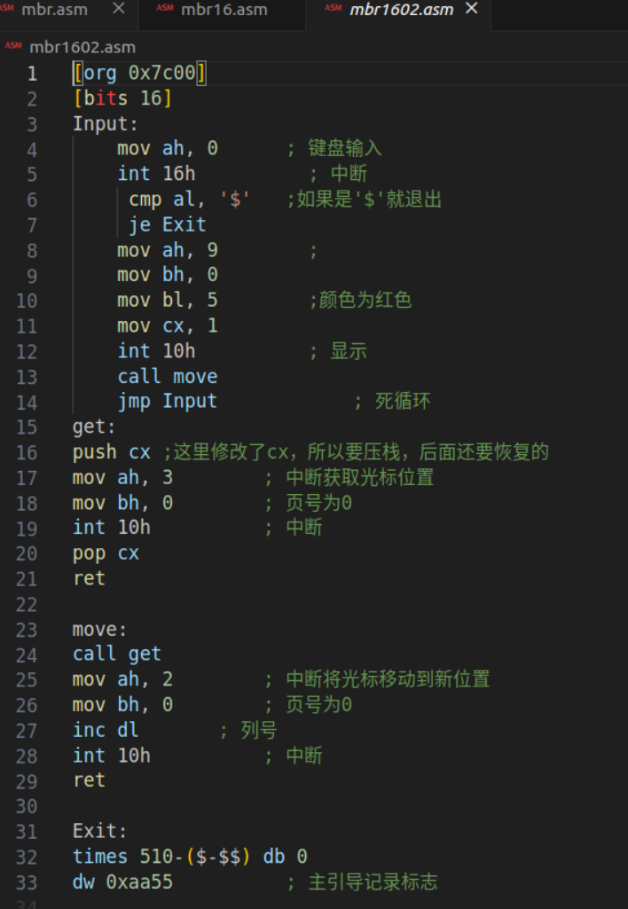
****

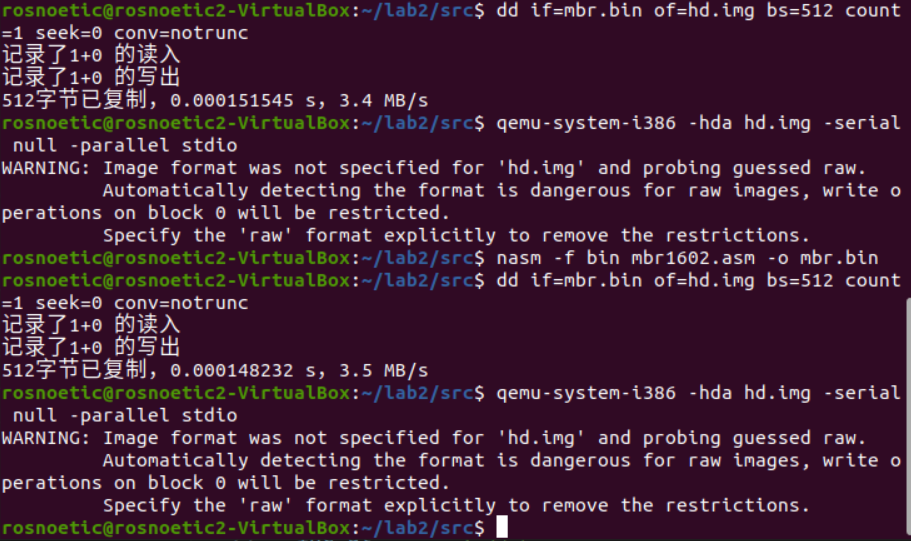
**学号显示**

****

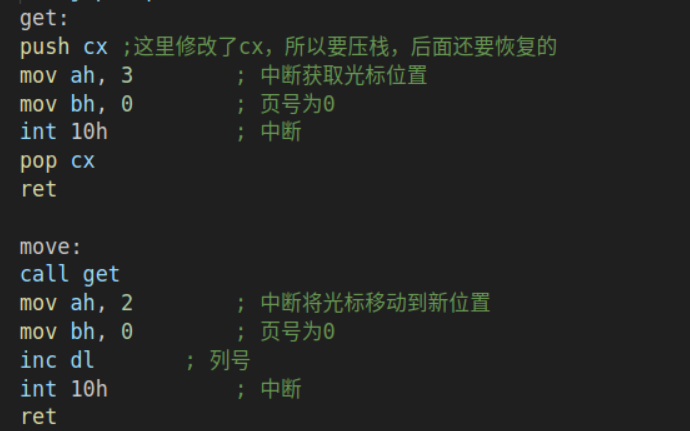
1. **Assignment 2**
2. **实验要求**
3. 请探索实模式下的光标中断，**利用中断实现光标的位置获取和光标的移动**。说说你是怎么做的，并将结果截图。
4. 请修改1.2的代码，**使用实模式下的中断来输出你的学号**。说说你是怎么做的，并将结果截图。
5. 在2.1和2.2的知识的基础上，探索实模式的键盘中断，**利用键盘中断实现键盘输入并回显**
6. **实验过程**
7. **利用int 10h(屏幕显示)，和int 16h（键盘中断）编写汇编代码**
8. **将汇编代码编译成二进制文件，写入虚拟硬盘中**
9. **启动qemu模拟器，执行mbr程序**

****

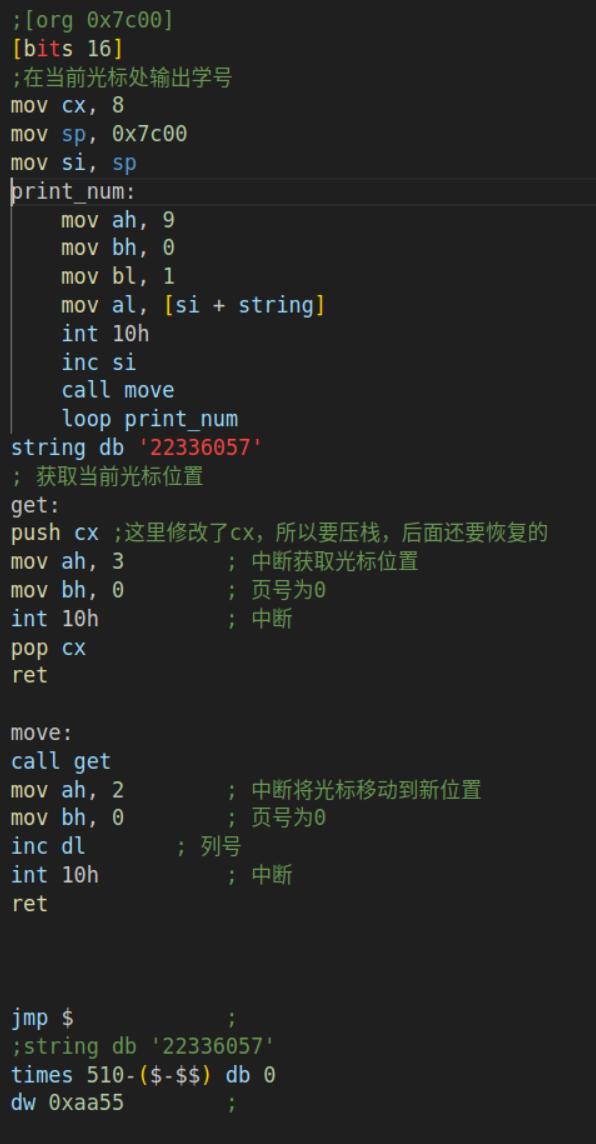
****

****

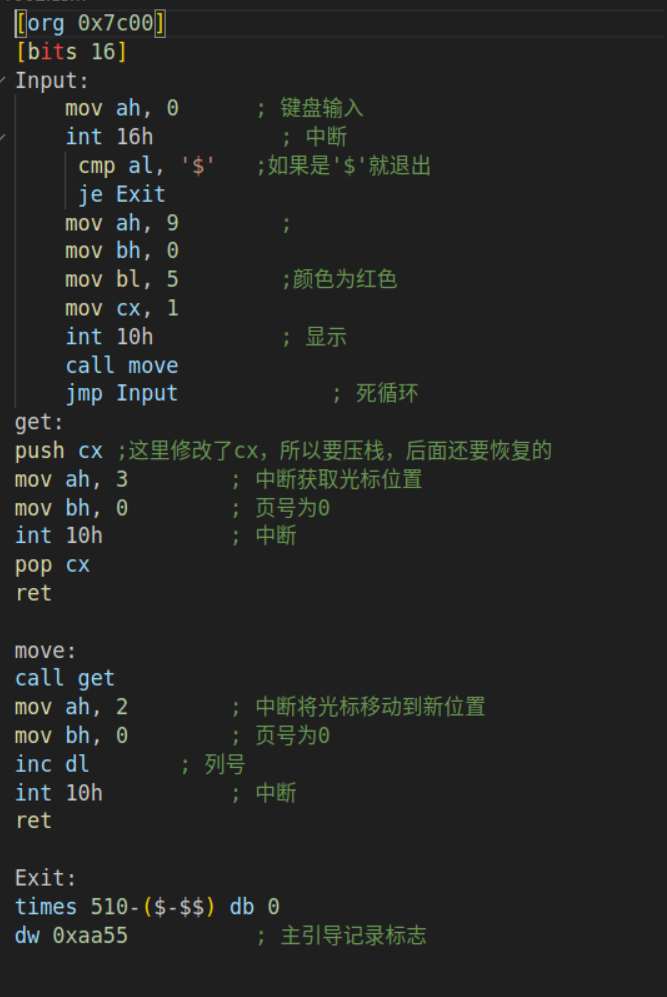
1. **关键代码**
2. **获得光标位置和移动：**
   1. **获得光标位置：使用int 10h中断和03功能号，bh寄存器中放入页号，中断后获得的结果存储在dx寄存器中**
   2. **移动光标：使用 int 10h中断和02功能号，先调用获取光标位置的函数，将行列信息存储在dx中再中断**

****

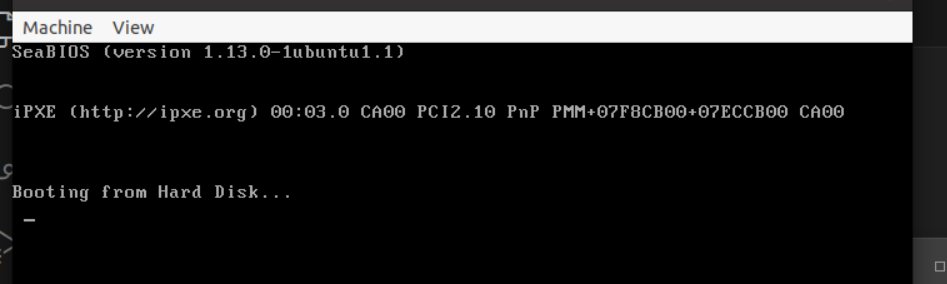
1. **利用中断输出学号：**
2. **定义字符string，注意string标识符里面存储的是字符串对0x7c00的相对偏移，如果要取出字符要间接段寻址**
3. **在光标处输出字符，利用了int 10h中断的09h功能号，和循环遍历输出，光标的移动实现调用2.1的实现**

****

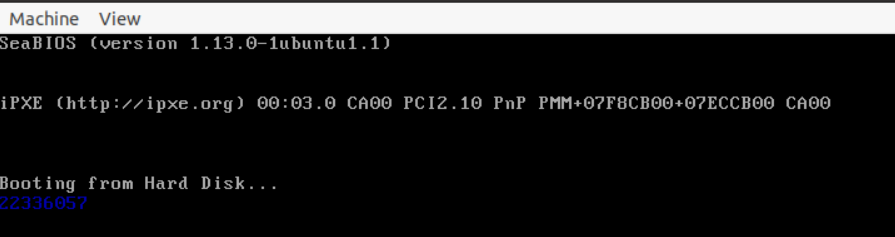
1. **键盘中断和回显：**

**先利用int 16h中断0功能号进行键盘中断，接收的字符存在al寄存器中，再调用int 10h中断09功能号显示在屏幕上，调用2.1实现光标移动**

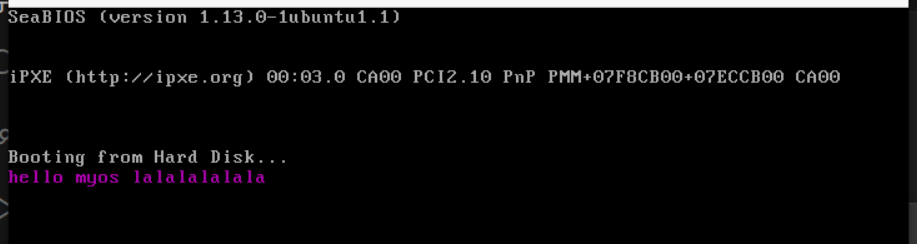
1. **实验结果**
2. **光标位置和移动（光标移动一位）**

****

1. **学号输出**

****

**键盘中断和回显**

****

1. **总结**

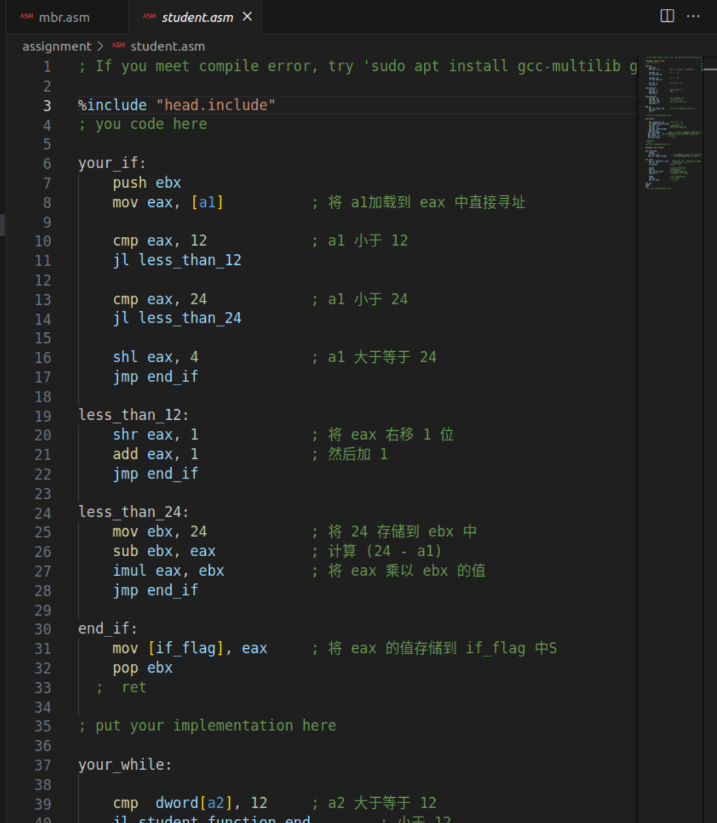
**如何获取string中的字符的寻址方式：段间接寻址**

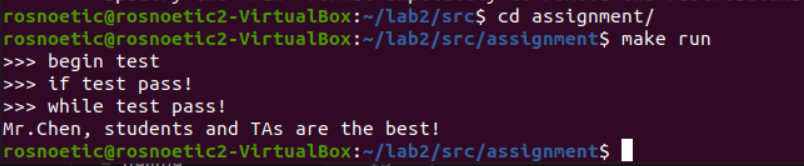
1. **Assignment 3**
2. **实验要求**

**分支逻辑，循环逻辑，函数的实现**

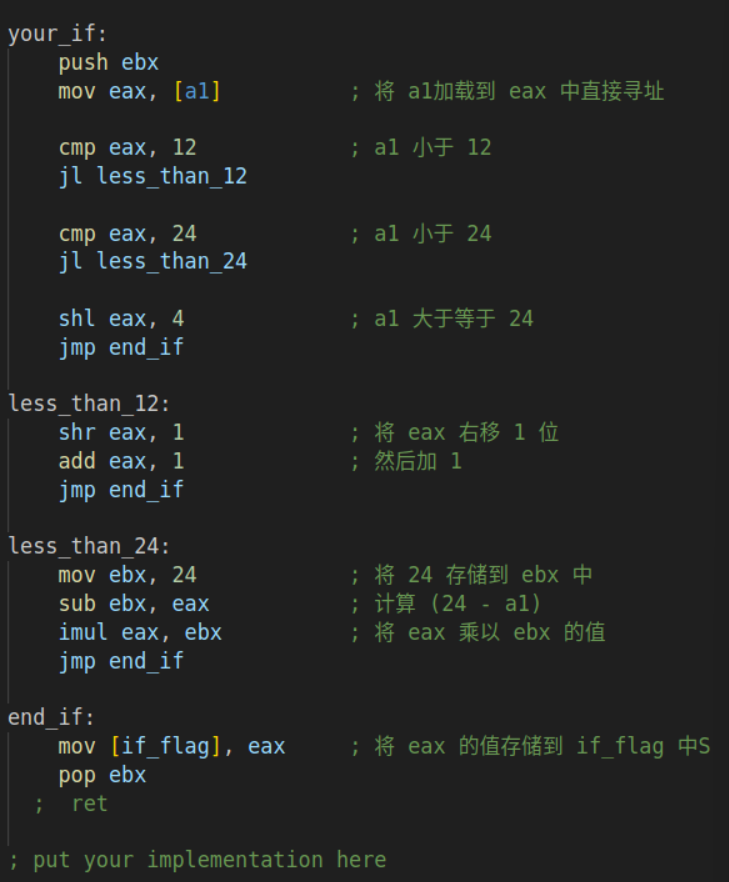
1. **实验过程**

**编写汇编代码，运行**

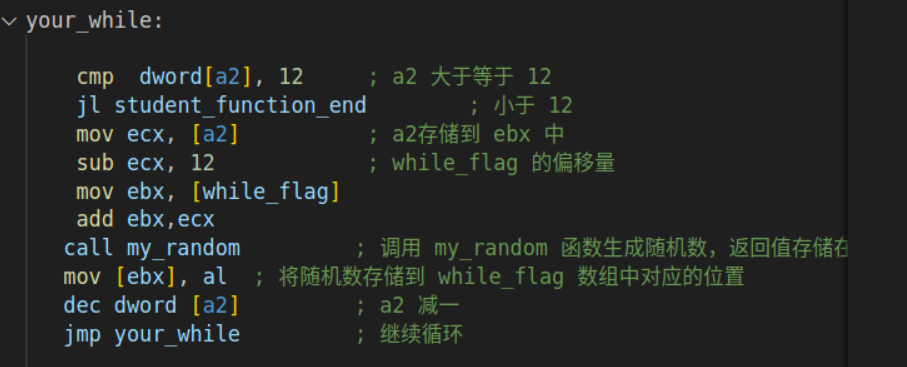




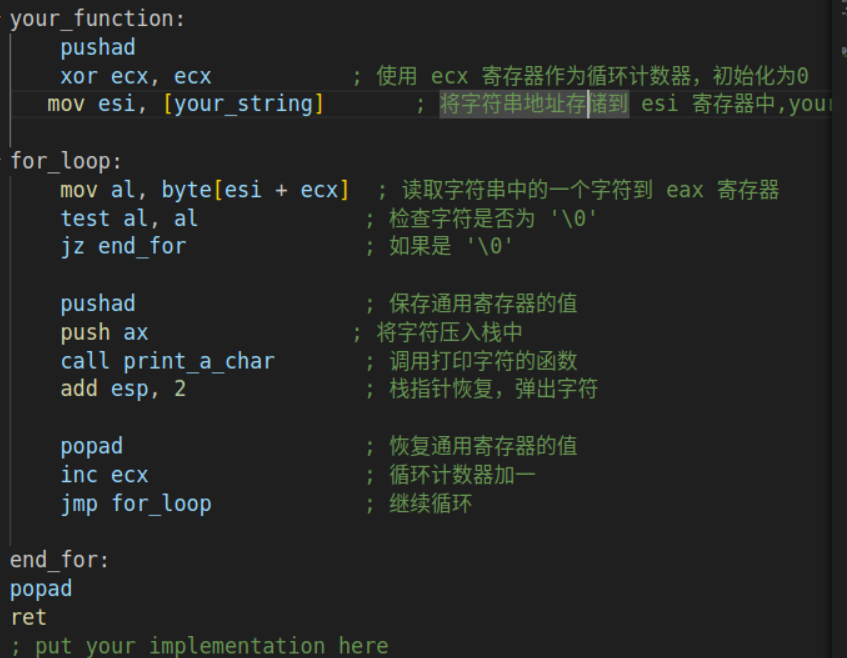
1. **关键代码**
2. **Your\_if:主要体现汇编的标签的使用**
   1. **注意要保存寄存器压栈和返回前的出栈恢复寄存器，变量的改变**
   2. **对标识符a1的使用：a1是存放着变量的地址，要用间接寻址**
   3. **运算符的使用：shr右移算除法，shl左移算乘法**

****

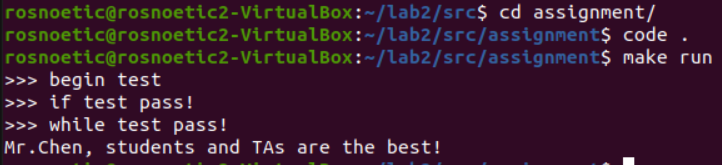
1. **Your\_while:主要体现汇编的循环逻辑**
2. **标识符的使用：这里的a2和while\_flag都是地址的地址，a2中存放的是a2变量的地址，while\_flag中存放的是while\_flag数组的开头地址，所以要用间接寻址**
3. **将生成的随机数放入数组：虽然生成的随机数存放在eax，但是只用放进al，因为随机生成字符只占8bit，如果放入eax（32bit）会改变后面的内存空间，会造成无论放入多少个字符，最终实际放入的只有一个的错误情况**
4. **函数的返回：这里的返回不用也不能自己ret，因为包含了end.include，在这里面已经做了返回和恢复寄存器工作**

****

1. **Your\_function:**
   1. **进入函数调用前要保存所有寄存器（pushad），函数返回前要恢复所有寄存器（popad）**
   2. **标识符your\_string:里面存储的是指向your\_string的首地址的指针，要用间接寻址法将首地址提取出到esi中**
   3. **根据首地址加偏移，要一个一个字节的将字符取出放入到寄存器中**
   4. **Add esp, 2：传入函数print\_a­\_char的参数ax传入前压栈了，则函数返回时，调整栈指针esp，将参数弹出栈**

****

1. **实验结果**

**全部运行成功**  
 

1. **总结**

**要注意标识符的意思和寻址方式：一般是是地址的地址，间接寻址。**

**当写入寄存器和内存时，根据不同的要求，要指定写入不同的长度：如写入一个字符就是八个bit，一个字节**

1. **Assignment4**
2. **实验要求**

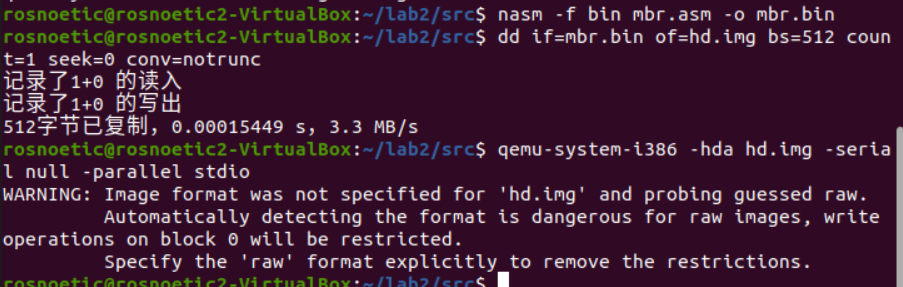
字符弹射程序。请编写一个字符弹射程序，其从点$(2,0)$处开始向右下角45度开始射出，遇到边界反弹，反弹后按45度角射出，方向视反弹位置而定。同时，你可以加入一些其他效果，如变色，双向射出等。注意，你的程序应该不超过510字节，否则无法放入MBR中被加载执行。

（由于之前做的时候没有看清实验要求，特此说明，在我的字符弹射程序中从（12，0）发射，然后向右上方45度弹射，碰壁反弹）

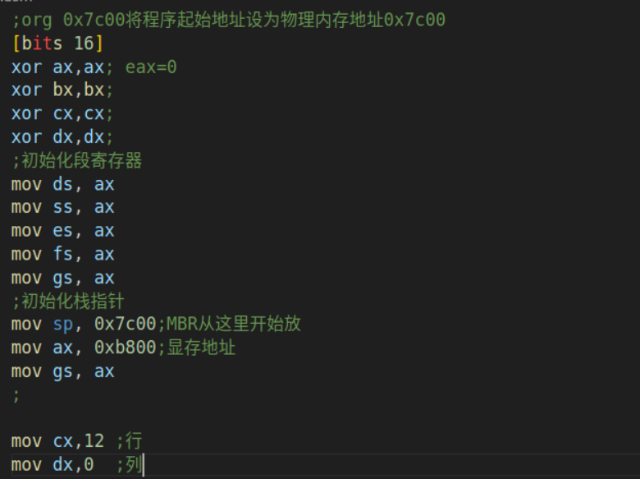
1. **实验过程**
2. **编写mbr程序**



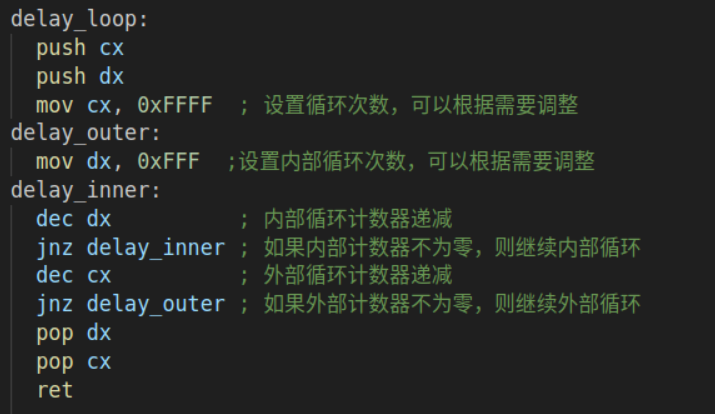
1. **将汇编文件 mbr.asm编译成二进制文件 mbr.bin**
2. **创建虚拟硬盘，在图上该步骤省略**
3. **将mbr.bin写入虚拟硬盘的首扇区**
4. **启动qemu模拟计算机启动**



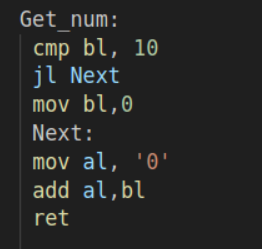
1. **关键代码**
2. **初始化寄存器:gs里面放显存地址，dx寄存器的值相当于列号，cx寄存器相当于行号**



1. **函数：**
   1. **延迟函数：为了做到弹射效果，每显示一个字符就要延迟**

****

* 1. **Get\_num函数：将数字字符放入ax的低位地址al中，为了能够显示出数字字符**

****

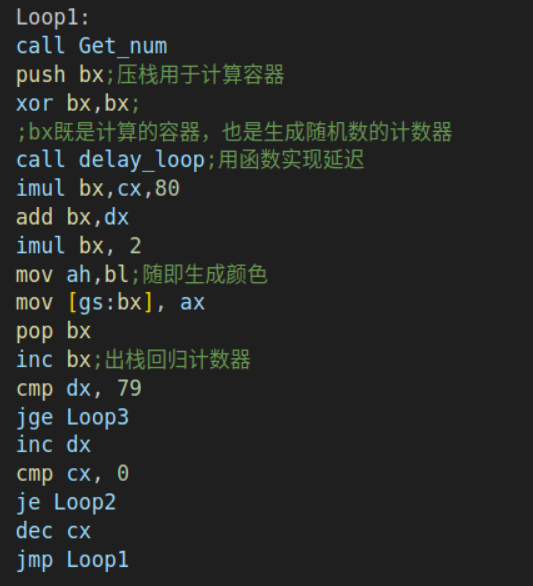
* 1. **Loop1：实现向右上方的显示位置移动和字符显示**
     1. **bx：bx在loop1中既是全局计数器（传入Get\_num中获得数字字符），又是计算显示字符在显存中的偏移位置，显示完后恢复成计数器**
     2. **bx计算：**

****

**这里的x是cx，y是dx**

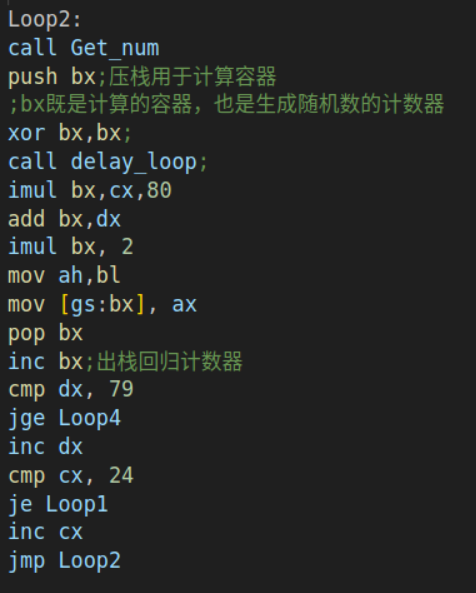
* + 1. **控制跳转逻辑：**

**如果碰到右边的壁（dx=79），则到loop3反弹，碰到上边的壁（cx=0），则到loop2反弹，否则回到loop1继续向右上弹射**

****

* 1. **Loop2：只有控制逻辑与loop1不同**

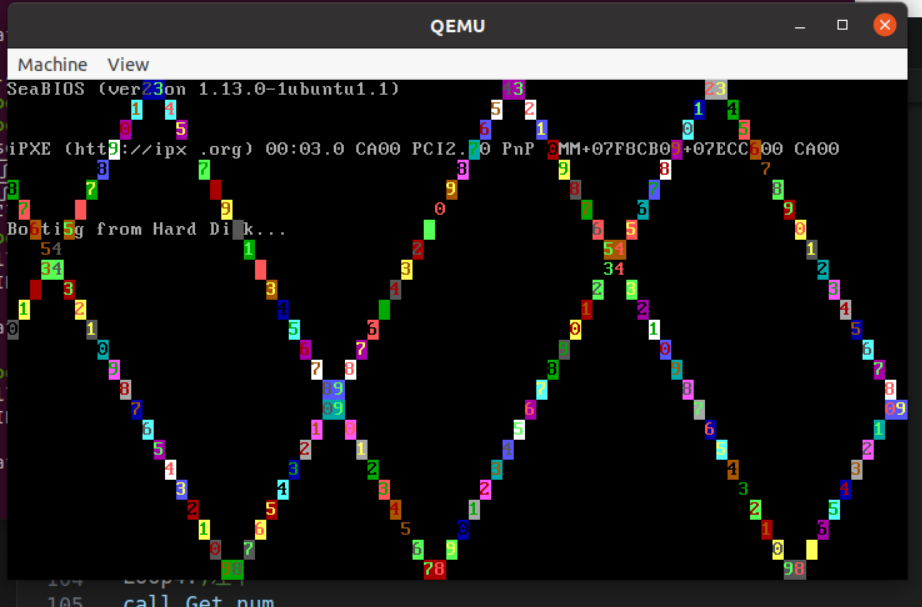
**如果碰到右边的壁（dx=79），则到loop4反弹，碰到下边的壁（cx=24），则到loop1反弹，否则回到loop2继续向右下弹射**

****

* 1. **Loop3，Loop4同理（Loop3左上，Loop4左下）**

1. **实验结果**

**在字符弹射程序中从（12，0）发射，然后向右上方45度弹射，碰壁反弹**



1. **总结**

Get\_num函数中 如果bl满10了应该mov bl，0而不是sub bl，10因为直接sub 10有sub完后溢出或者其他效果：