

课程实践报告



**课程 网络攻防技术(314006040)**

**课 序 号 3**

**作业名称 shellcode编写实验**

**评分**

**姓名** 邓嘉怡 **学号** 2022141530010

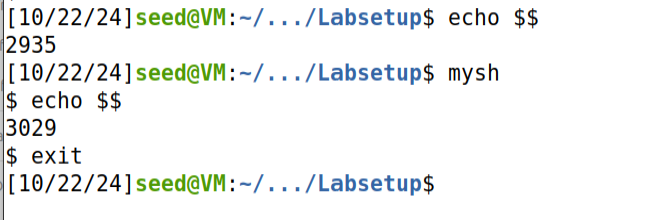
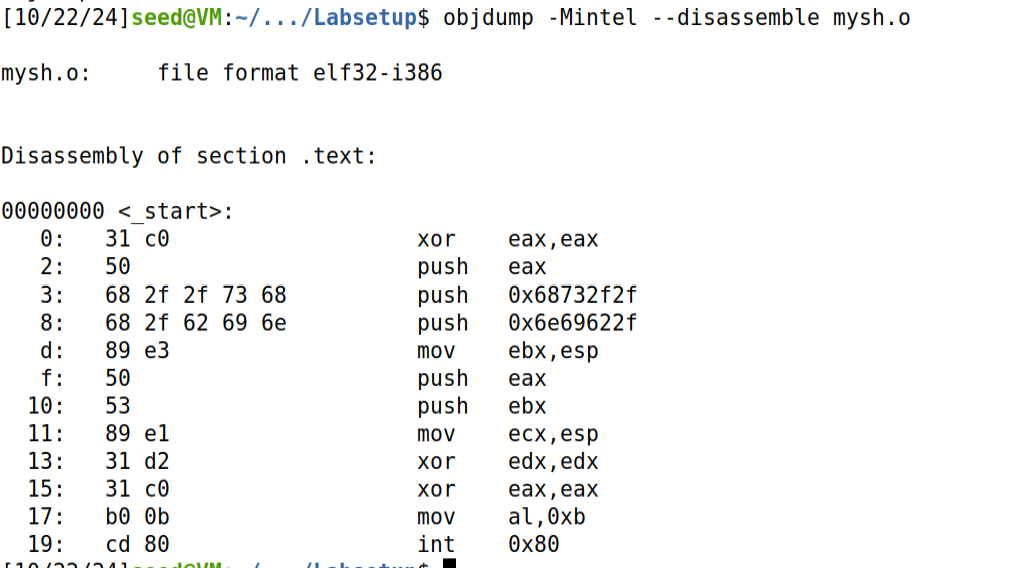
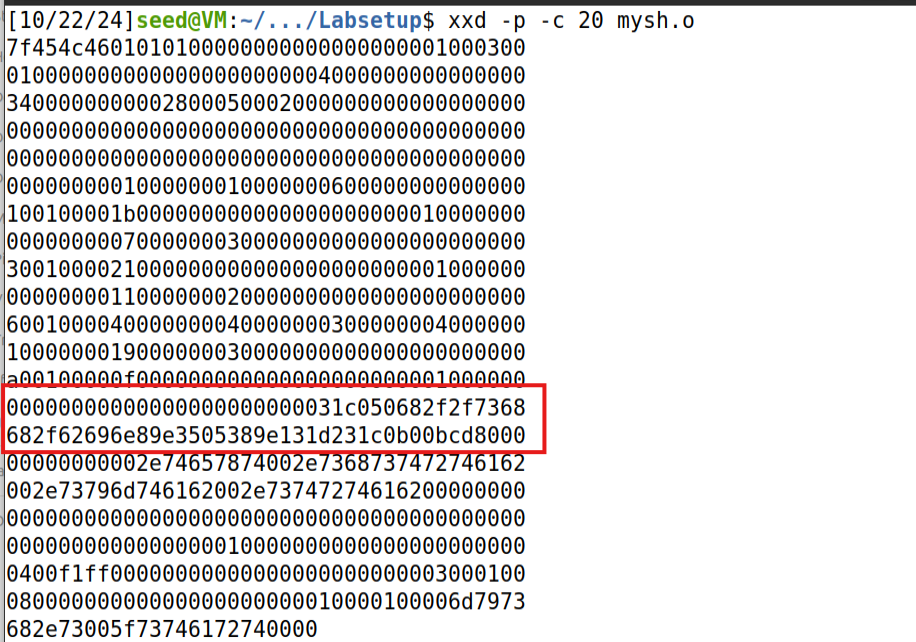
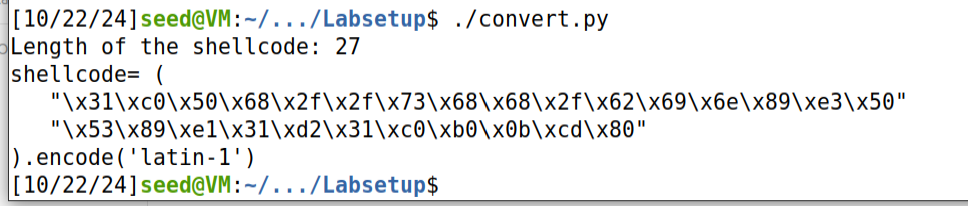
**评阅意见**

# 作业题目

shellcode广泛用于许多涉及代码注入的攻击中。编写shellcode是相当有挑战性的。虽然我们可以很容易地从互联网上找到现有的shellcode，但是能够从头开始编写我们自己的shellcode总是令人兴奋的。shellcode中涉及到几种有趣的技术。本实验室的目的是帮助学生理解这些技术，以便他们能够编写自己的shellcode。

编写shellcode有几个挑战，一个是确保二进制文件中没有0x00，另一个是找出命令中使用的数据的地址。第一个挑战不是很难解决，有几种方法可以解决它。第二个挑战的解决方案导致了编写外壳代码的两种典型方法。在一种方法中，数据在执行期间被推入堆栈，因此可以从堆栈指针获得它们的地址。在第二种方法中，数据存储在代码区域中，就在调用指令之后，因此在调用调用函数时，其地址被推入堆栈（作为返回地址）。两种解决方案都非常优雅，我们希望学生能够学习这两种技术。

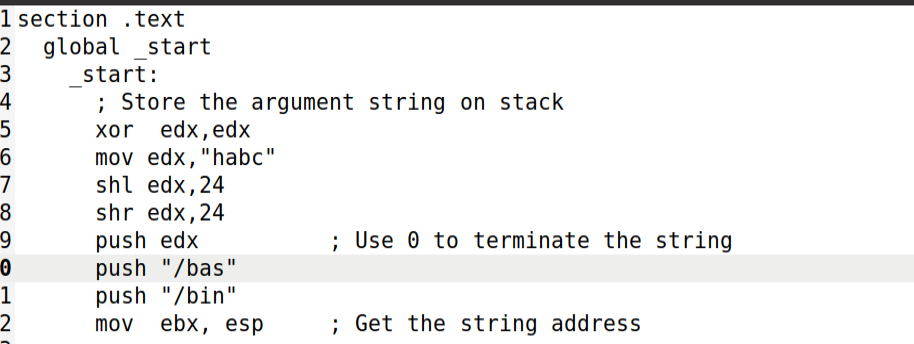
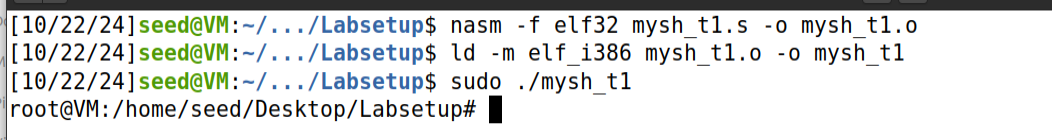
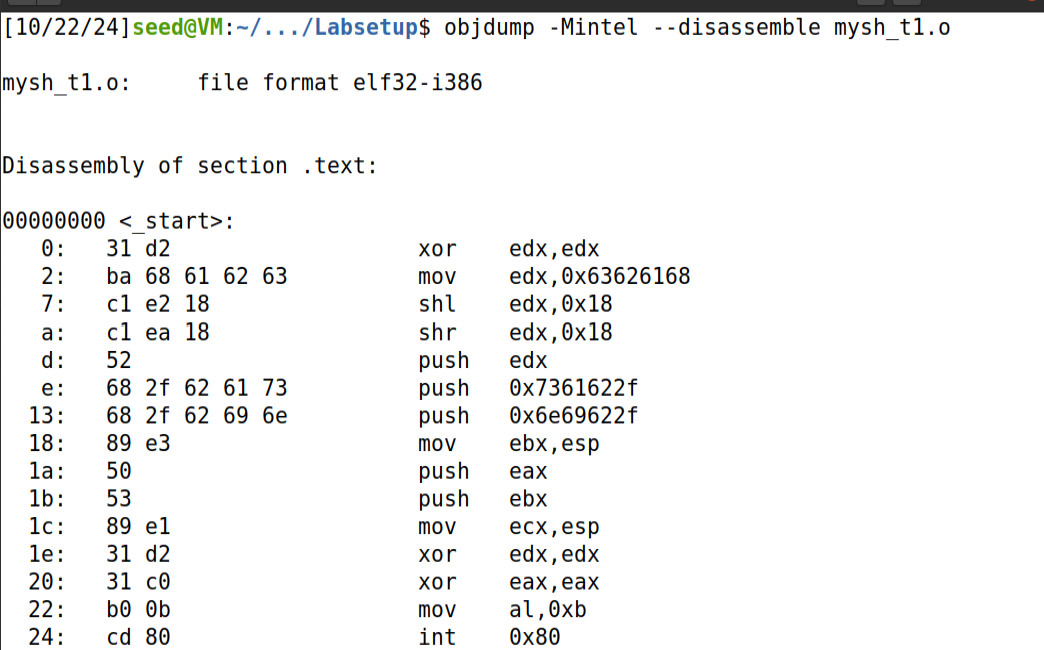
# 实验步骤及结果 任务一.a：写shellcode

1. 编译mysh.s并得到二进制文件  
   
2. 执行二进制文件  
   发现运行mysh之前的pid和运行mysh之后的pid不相同，证明我们通过mysh启动了一个新的shell  
   
3. 获取机器码  
   
4. 查看二进制文件  
     
   截取：31c050682f2f7368682f62696e89e3505389e131d231c0b00bcd80  
   复制到convert.py中  
   
5. 执行convert.py，得到包含16进制机器码  
   

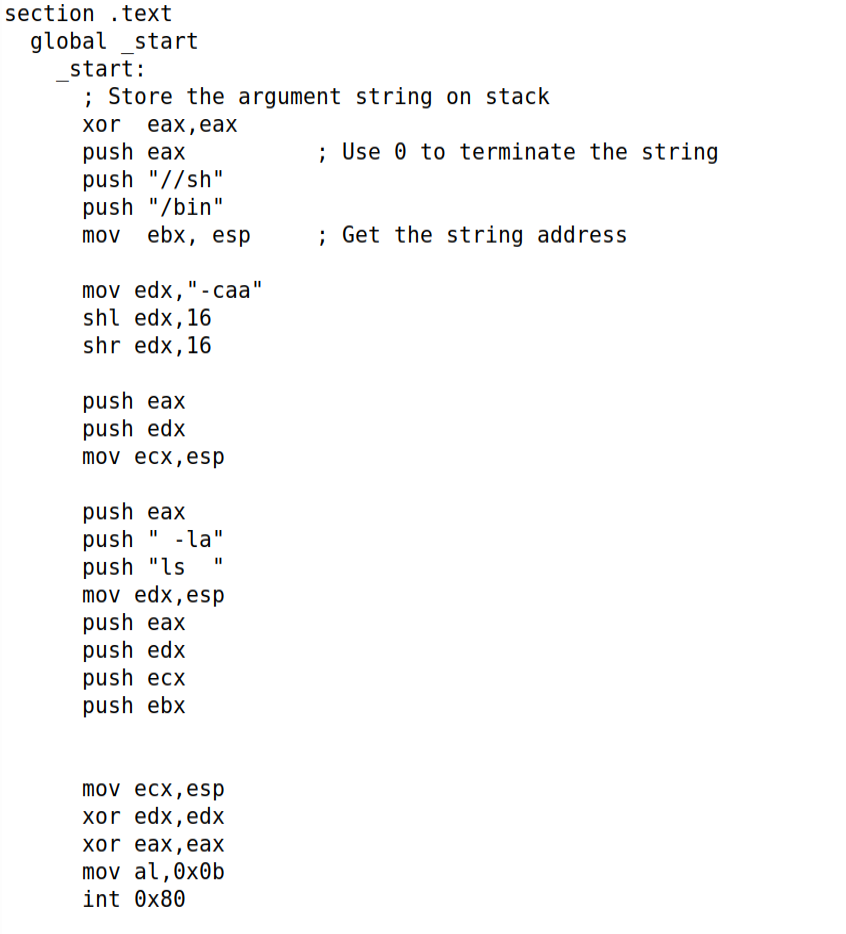
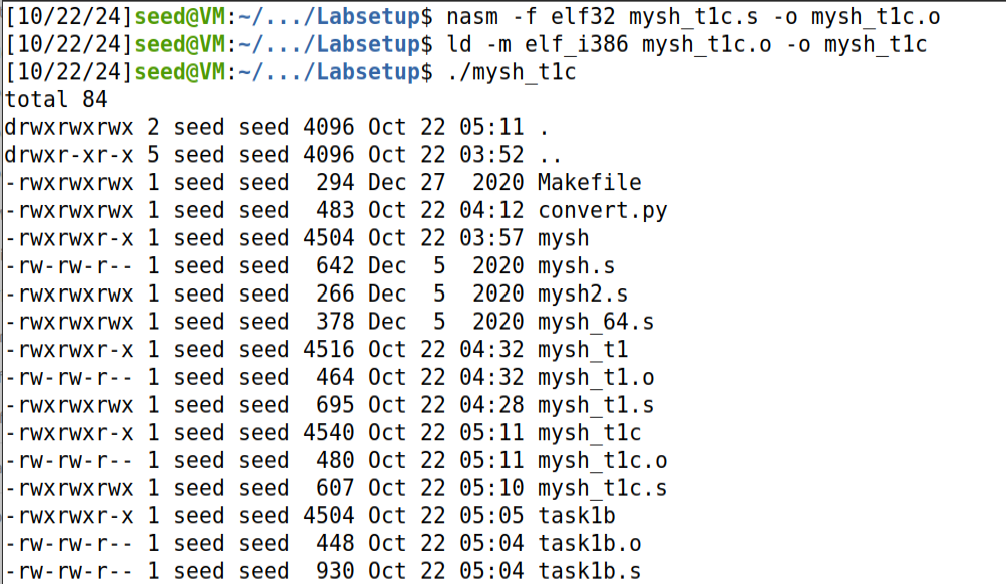
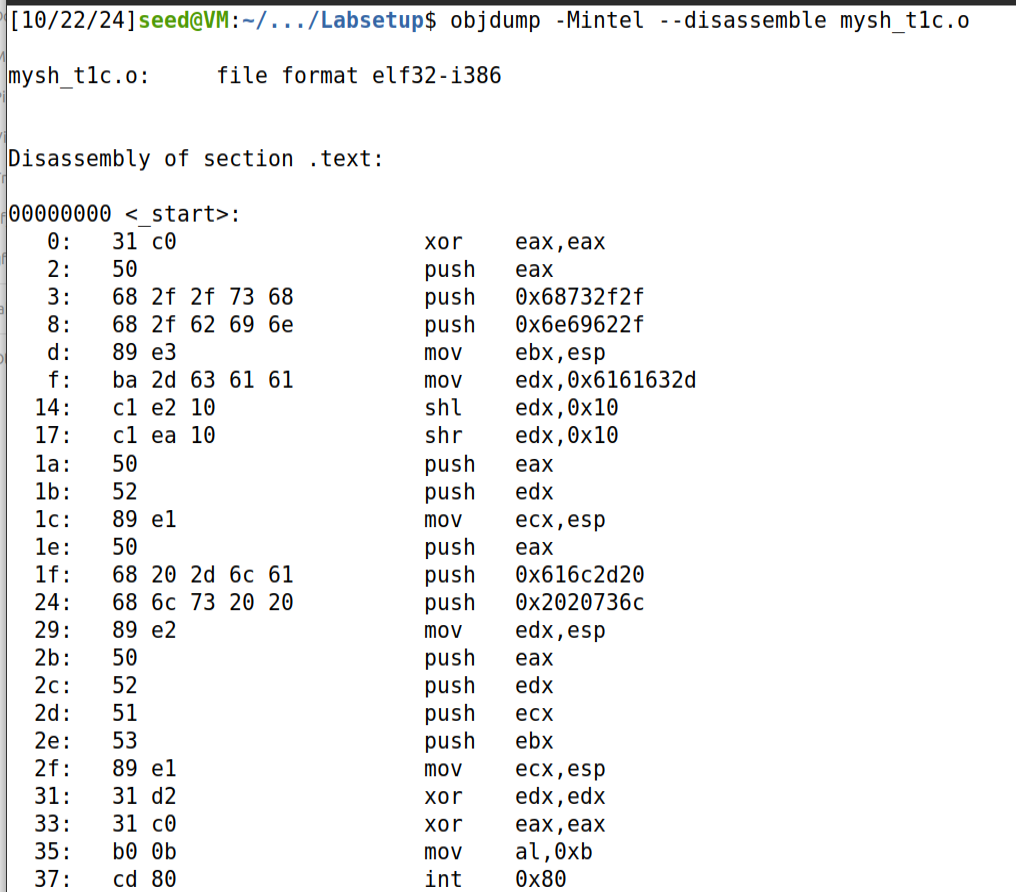
**任务一.b：消除代码中的0**

1. 解释为什么”xor eax, eax”会有效的将0分配给eax，并且不会在机器

码中获得一个0？  
move 指令为5个字节，会将eax后续的机器码中获得0，而xor指 令为2个字节，机器码中不会有多余的0

1. 修改mysh代码如下，用小端的方法，把abc丢弃，h压入栈中  
   
2. 编译链接并执行  
   
3. 查看机器码  
   

**任务一.c：**

1. 修改mysh.s的代码如下  
   
2. 编译运行  
   
3. 查看机器码  
   

**任务二：**

1. 解释mysh2.s代码

section .text

global \_start

\_start:

BITS 32 指定NASM产生的代码被运行在32位模式处理器下

jmp short two 跳转到two位置

one:

pop ebx 将堆栈中一个字节弹出到ebx中，这里是db语句的ip地址

xor eax, eax eax和eax异或

mov [ebx+7], al 将al（就是eax的低八位）的数据传送给地址ebx+7

mov [ebx+8], ebx 将ebx的数据传送给地址ebx+8

mov [ebx+12], eax 将eax的数据传给地址ebx+12

lea ecx, [ebx+8] 将ebx+8 的有效地址传送给ecx

xor edx, edx edx 和 edx 异或

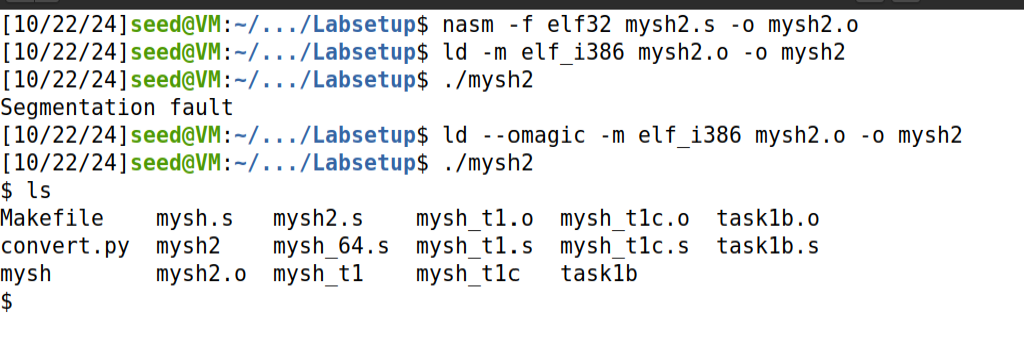
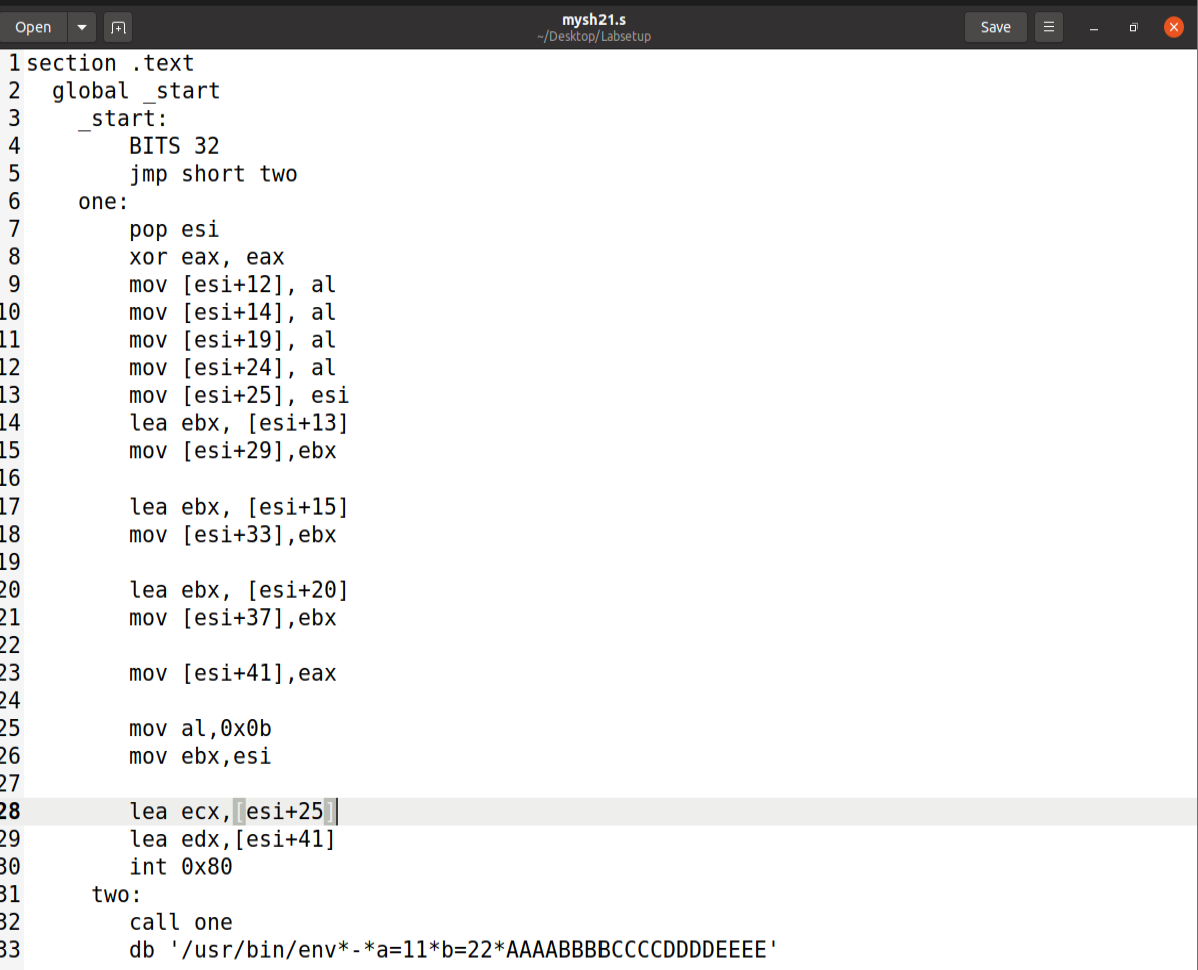
mov al, 0x0b 将 0x0b 传送给al(eax 的低8 位)

int 0x80

two:

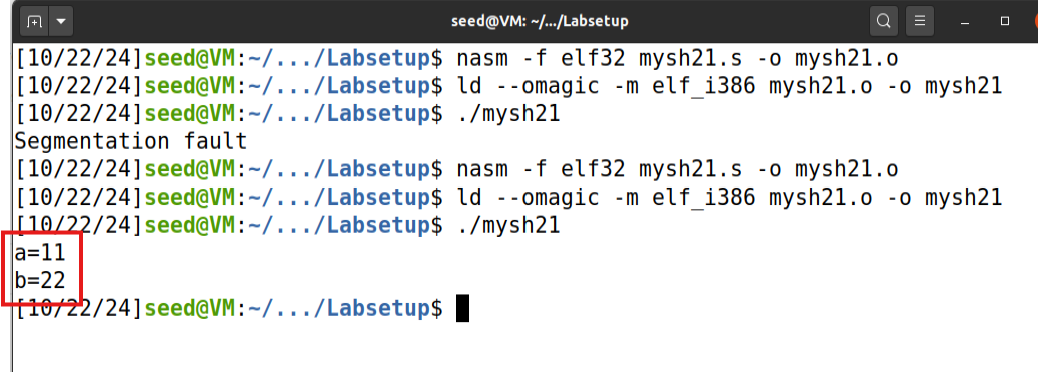
call one 先将 db 语句的IP压入栈中，再使当前的IP+16位位移，再跳转到one处

db '/bin/sh\*AAAABBBB' 定义指令

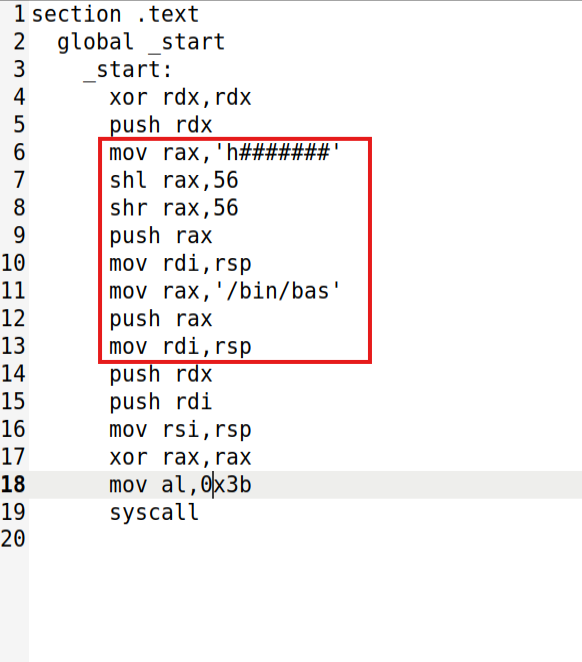
1. 编译运行  
   
2. 为什么代码能成功  
   mov[ebx+7],al，将\*替换为0x00，mov[ebx+8],ebx，将 AAAA 替换 为”/bin/sh”的地址，将BBBB替换为“0”；lea ecx, [ebx+8]； argv 此时为[“/bin/sh”]，所以可以执行成功
3. 修改代码  
     
   最终，修改后的mysh2.s可用c语言总结为：

char \*command[] = {"/usr/bin/env", "-"， "a=11", "b=22" NULL};

execve(command[0], command, NULL);

1. 编译并执行  
   

**任务三：写64bit的shellcode**

1. 将占位符替换为%0的操作修改mysh\_64，代码如下：  
   
2. 编译并运行，成功  
   