

**课 程 实 验 报 告**

**课程名称： 计算机系统基础实验**

**实验名称： ELF文件与程序链接**

**院 系 ： 计算机科学与技术**

**专业班级 ： 计算机本硕博2301班**

**学 号 ： U202315763**

**姓 名 ： 王家乐**

**指导教师 ： 李海波**

**2024 年 10 月 25 日**

**一、实验目的与要求**

通过修改给定的可重定位的目标文件（链接炸弹），加深对可重定位目标文件格式、目标文件的生成、以及链接的理论知识的理解。

实验环境：Ubuntu。

工具：GCC、GDB、readelf、hexdump、hexedit、od等。

**二、实验内容**

**在二进制层面，逐步修改构成目标程序“linkbomb”的多个二进制模块（“.o文件”），然后链接生成可执行程序，要求可执行程序运行能得到指定的效果。修改目标包括可重定位目标文件中的数据、机器指令、重定位记录等。**

**第1关 静态数据与ELF数据节**

修改二进制可重定位目标文件 phase1.o 的数据节中的内容（不允许修改其他节），使其与main.o链接后，生成的执行程序，可以输出自己的学号。

**第2关 简单的机器指令修改**

修改二进制可重定位目标文件 phase2.o 的代码节中的内容（不允许修改其他节），使其与main.o链接后，生成的执行程序。在phase\_2.c 中，有一个静态函数 static void myfunc( ) ，要求在 do\_phase 函数中调用myfunc( )，显示信息myfunc is called. Good!。

**第3关 有参数的函数调用的机器指令修改**

修改二进制可重定位目标文件 phase3.o 的代码节中的内容（不允许修改其他节），使其与main.o链接后，生成的执行程序。在phase\_3.c 中，有一个静态函数 static void myfunc(int offset) ，要求在 do\_phase函数中调用myfunc(pos )，将do\_phase的参数pos直接传递myfunc，显示相应的信息。

**第4关 有局部变量的机器指令修改**

修改二进制可重定位目标文件 phase4.o 的代码节中的内容（不允许修改其他节），使其与main.o链接后，生成的执行程序。在phase\_4.c 中，有一个静态函数 static void myfunc(char \*s) ，要求在 do\_phase 函数中调用myfunc(s )，显示出自己的学号。

**第5关 重定位表的修改**

修改二进制可重定位目标文件 phase5.o 的重定位节中的内容（不允许修改代码节和数据节），使其与main.o链接后，生成的执行程序运行时，显示Class Name : Computer Foundation. Teacher Name : Xu Xiangyang。

**第6关 强弱符号**

不准修改 main.c 和phase6.o，通过增补一个文件，使得程序链接后，能够输出自己的学号。 #gcc -no-pie -o linkbomb6 main.o phase6.o phase6\_patch.o

**第7关 只读数据节的修改**

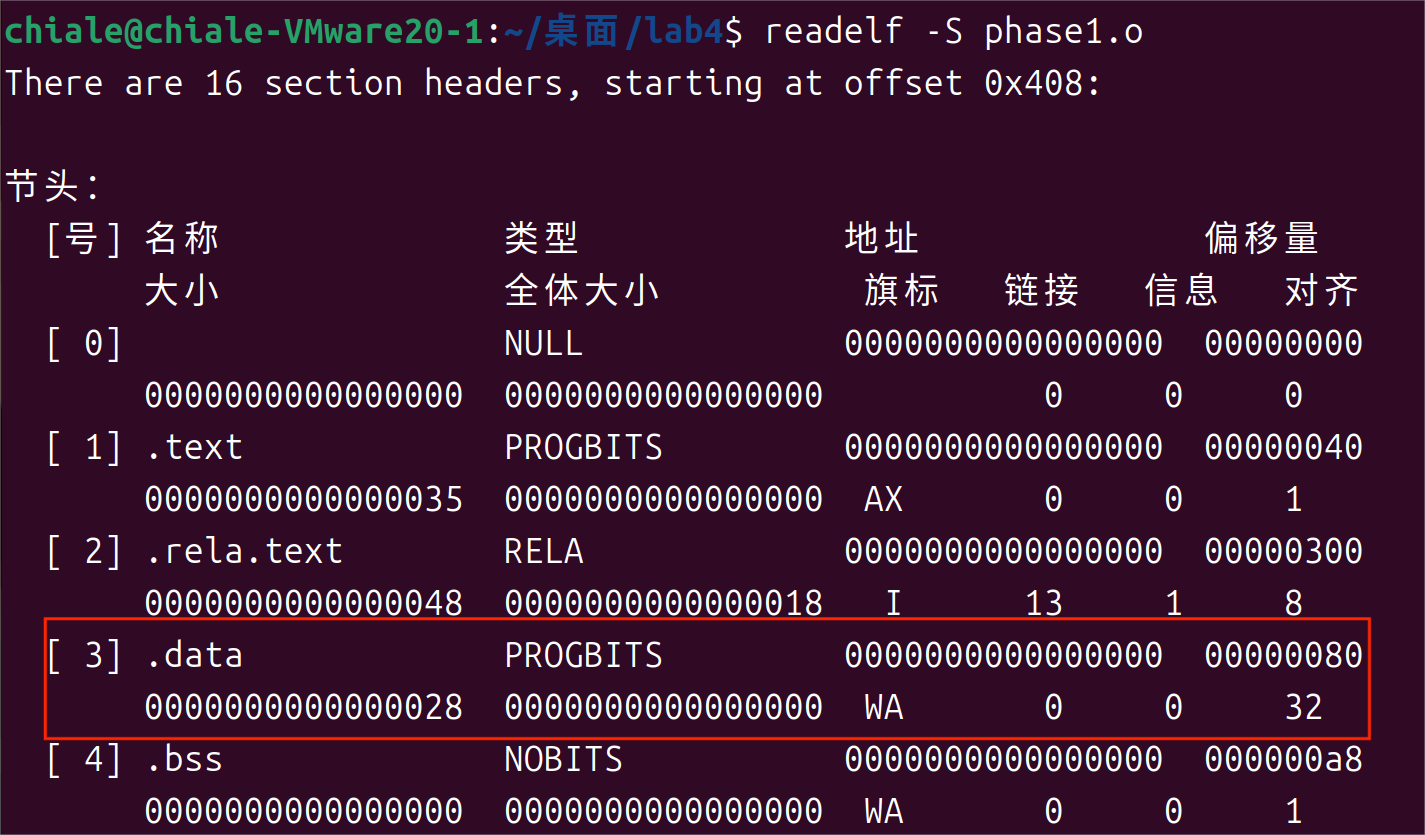
修改 phase7.o 中只读数据节（不准修改代码节），使其与main.o链接后，能够输出自己的学号。

**三、实验记录及问题回答**

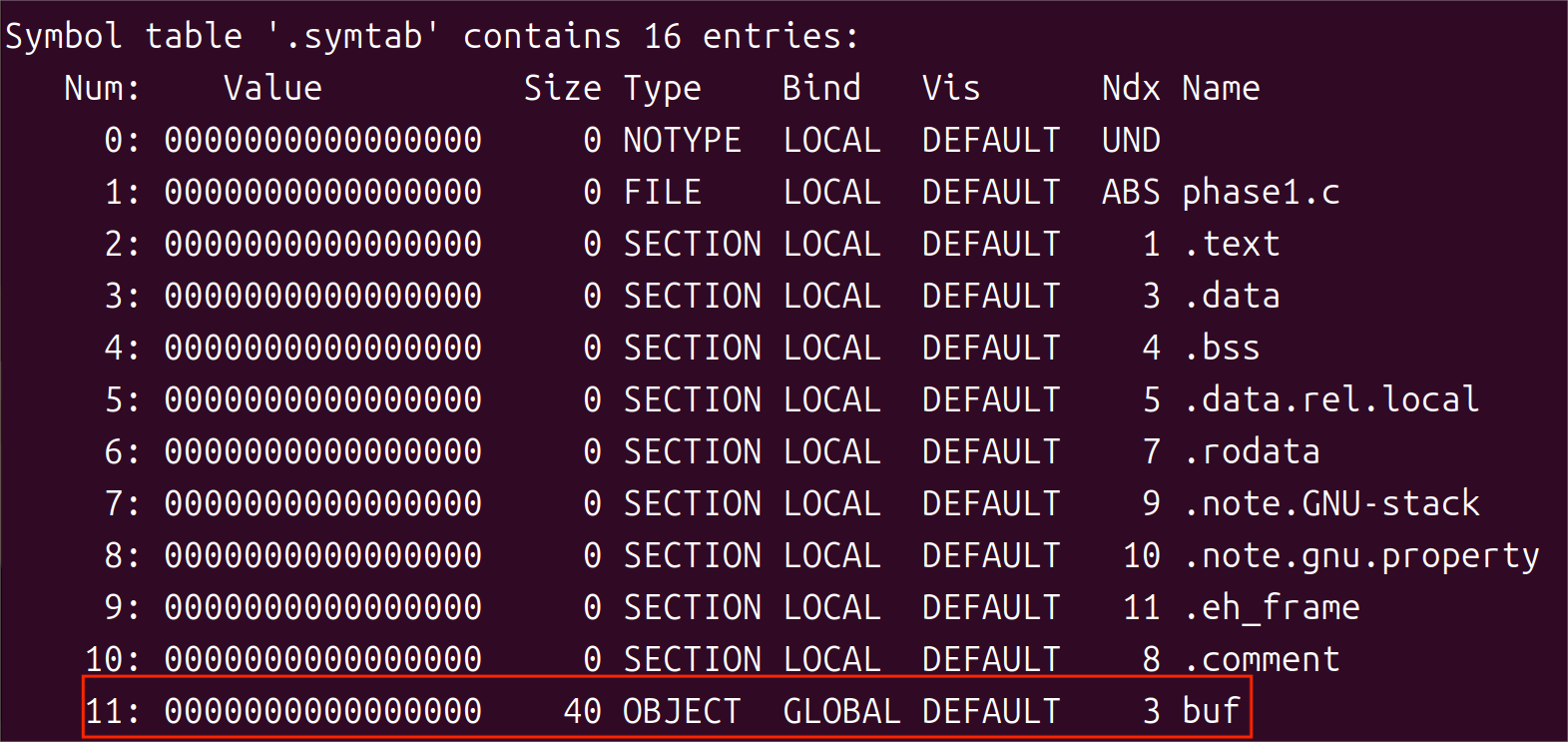
**（1）实验结果及操作过程记录**

**1.静态数据与ELF数据节**

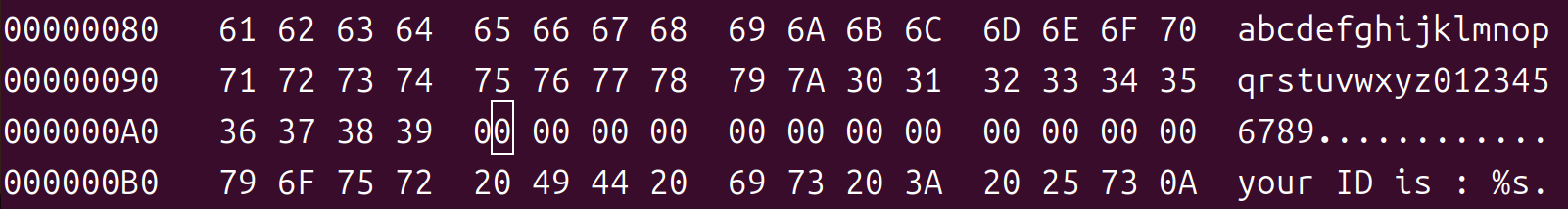
数据节在节头表中的第三号，偏移量为0x80,大小为0x28



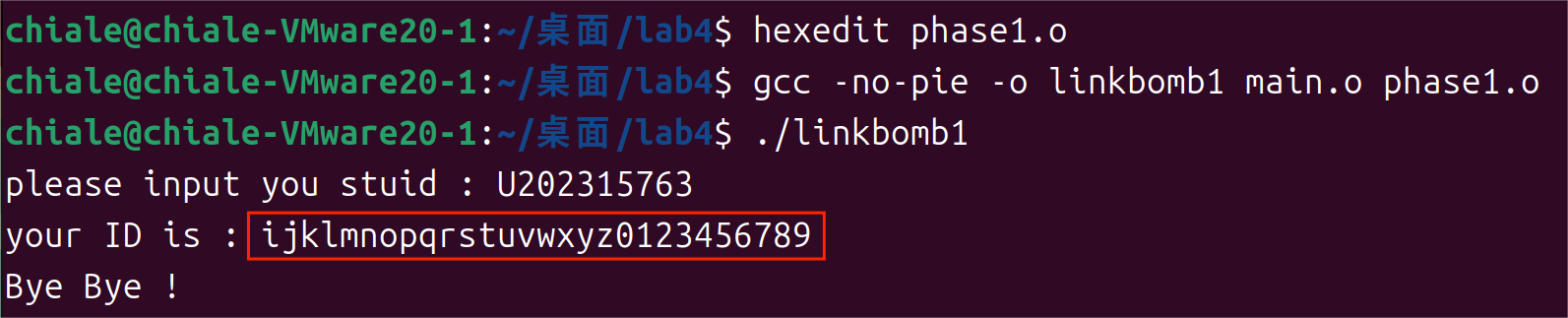
Buf数组在数据节，大小为40个字节



原phase1.o文件中偏移量为80处



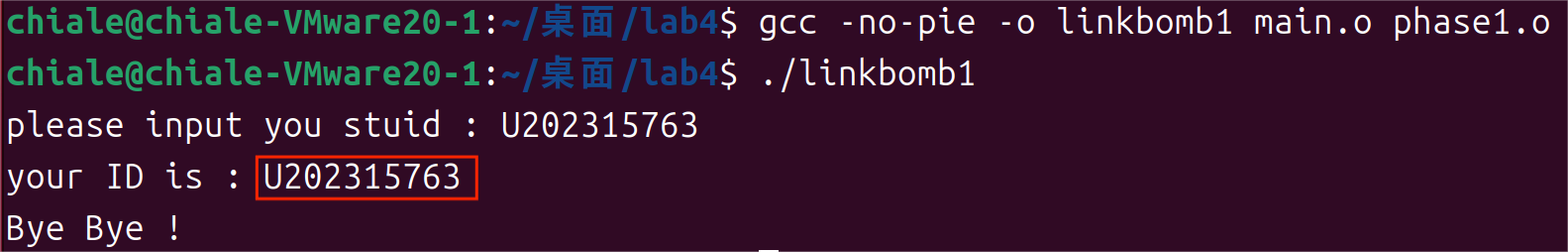
链接并运行，可发现从偏移量88处开始输出



phase1.0文件中可修改为



链接并运行，正确输出学号

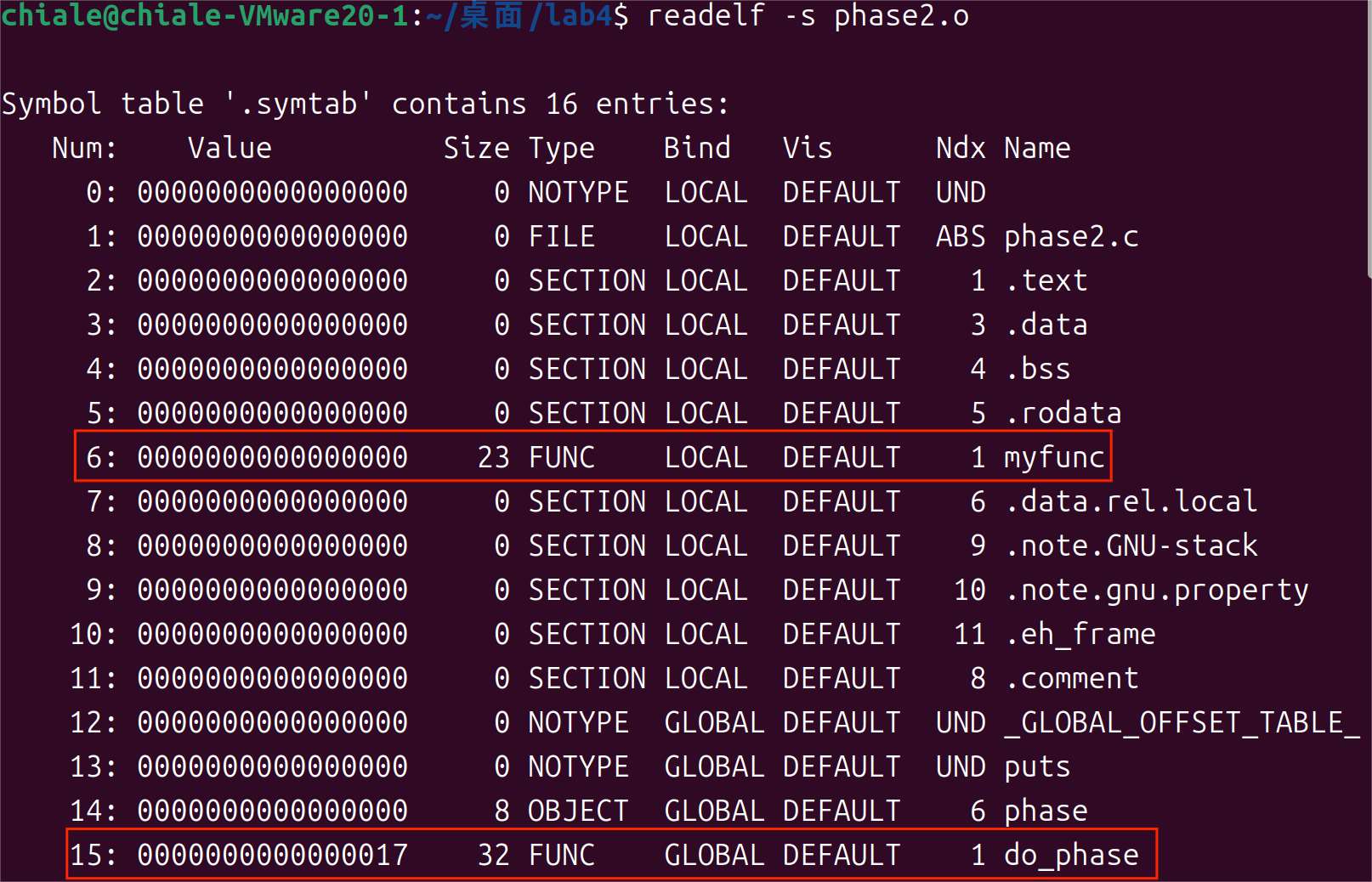


**2.简单的机器指令修改**

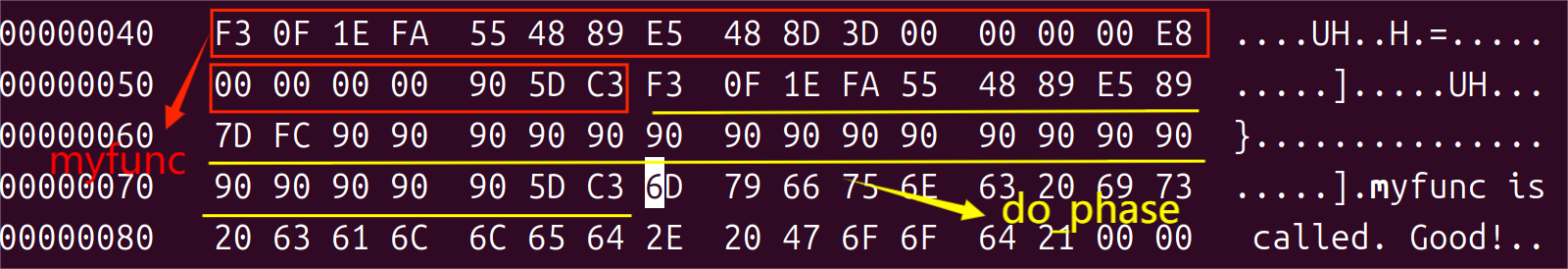
代码节在节头表中的第一号，偏移量为0x40，大小为0x37



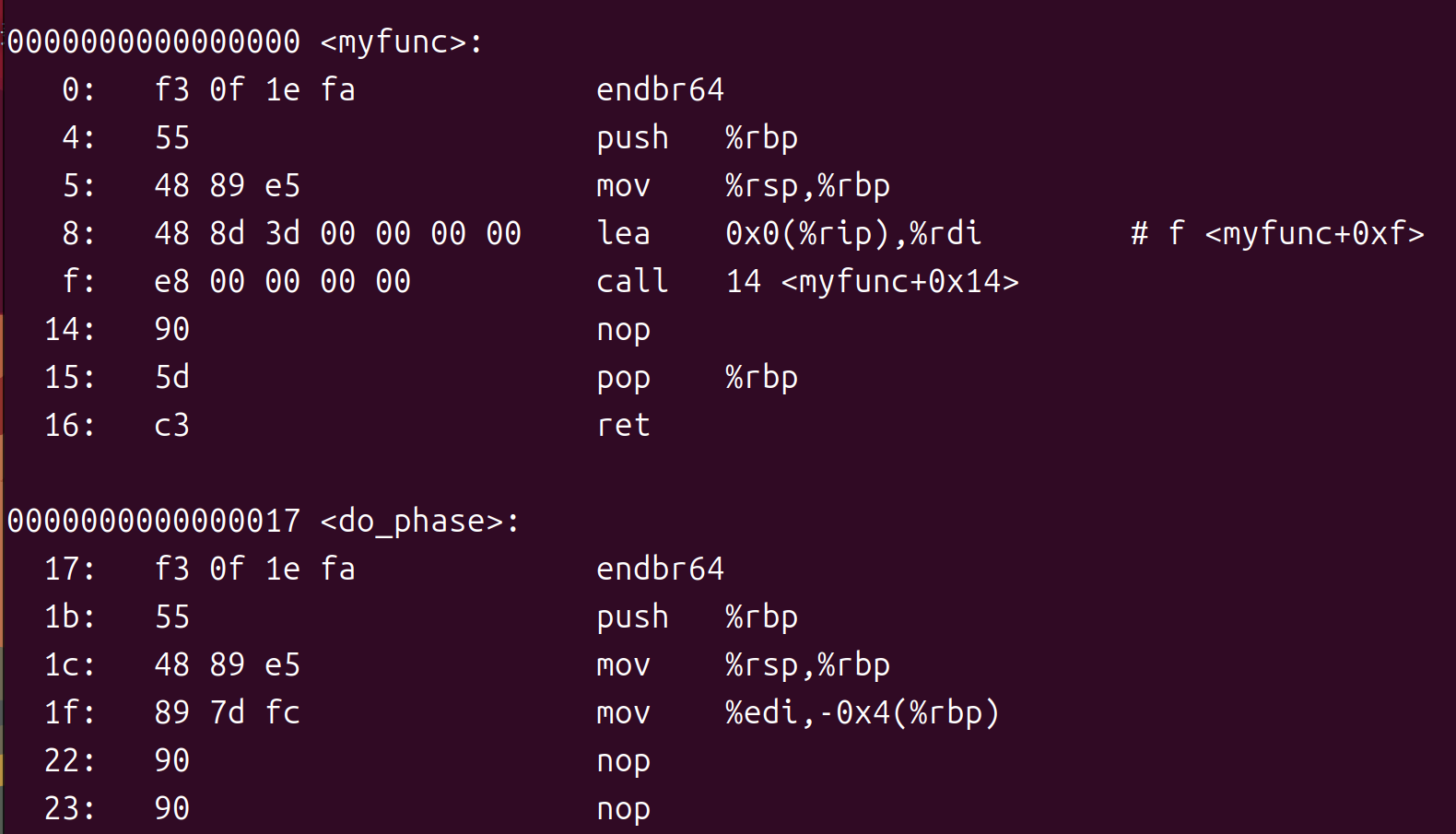
myfunc函数23个字节，偏移量为0；do\_phase函数32个字节，偏移量为0x17



因此两函数在phase2.o文件中的位置如下图



使用objdump -d phase2.o指令反汇编



要让do\_phase调用myfunc，我们需要在do\_phase函数内部插入一个call**指令，假**

**设我们要在地址0x22处插入call，那么call的目标是0x0，而call使用的是相对**

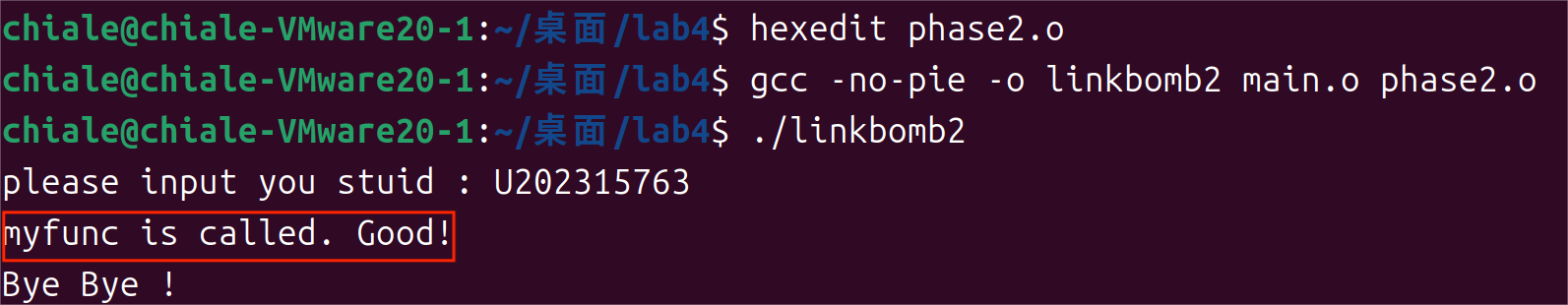
**地址,call下一条指令的地址是0x27(call指令为5个字节)，**跳转的相对偏移量是：

0x0 - 0x27 = -0x27，转换为十六进制：0xffffffd9,因为是小端序存储，且call指

令的机器码为e8，所以添加的指令为e8 d9 ff ff ff



链接并运行，正确调用myfunc函数

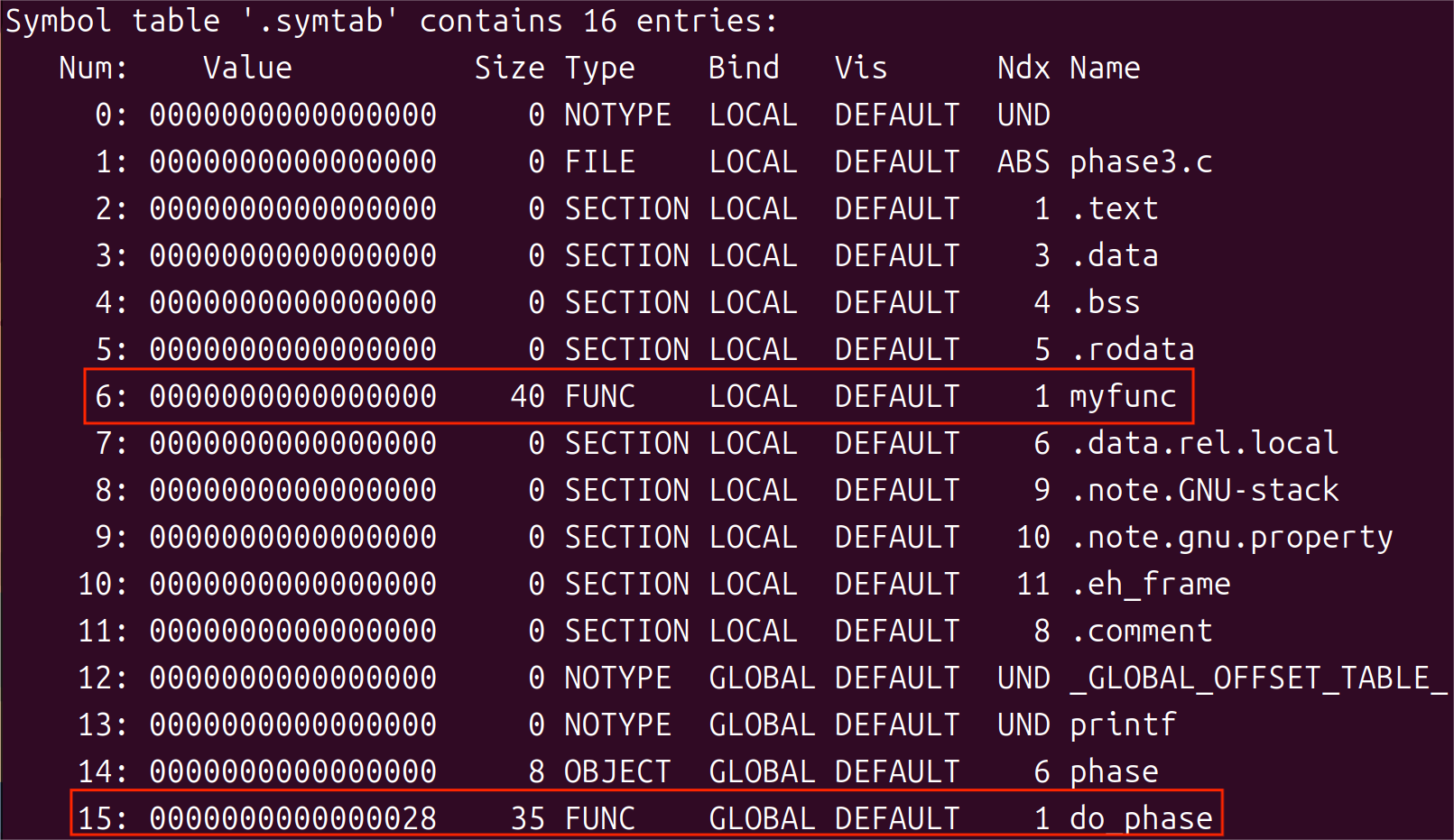


**3.有参数的函数调用的机器指令修改**

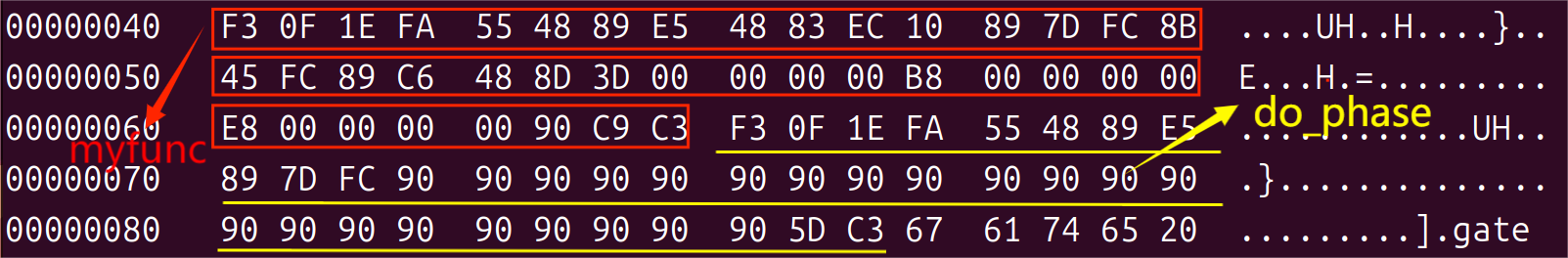
代码节在节头表中的第一号，偏移量为0x40，大小为0x4b



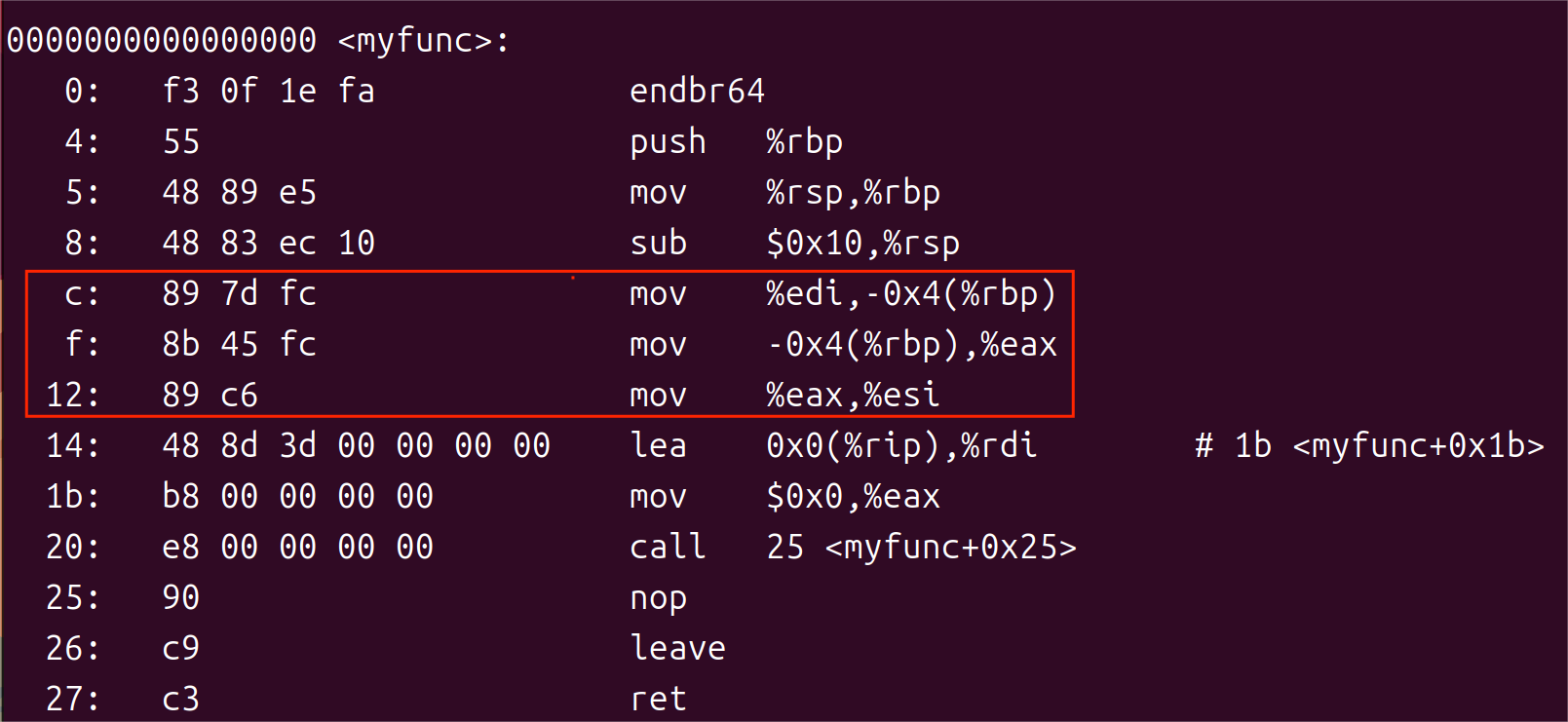
myfunc函数40个字节，偏移量为0；do\_phase函数35个字节，偏移量为0x28



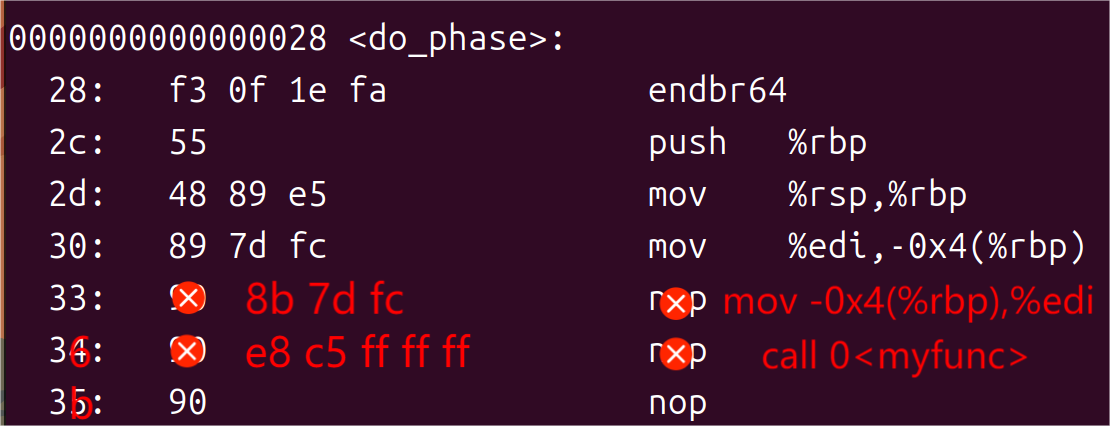
因此两函数在phase2.o文件中的位置如下图



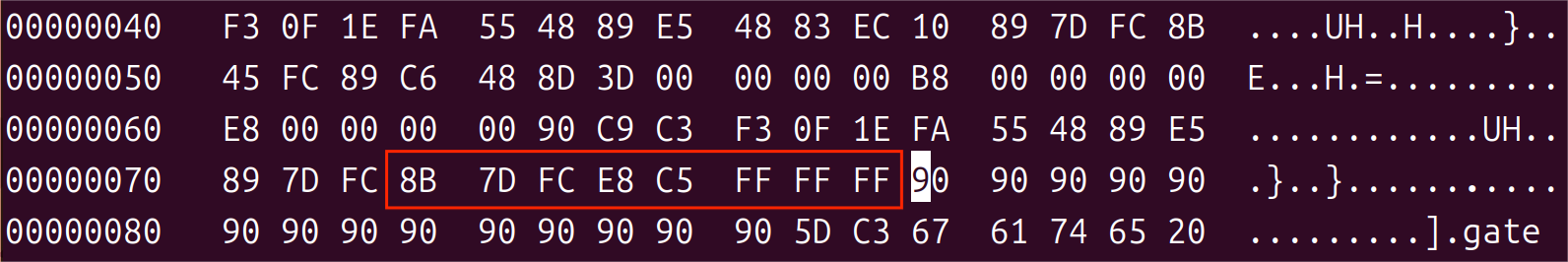
使用objdump -d phase3.o指令反汇编，myfunc的参数由%edi传递



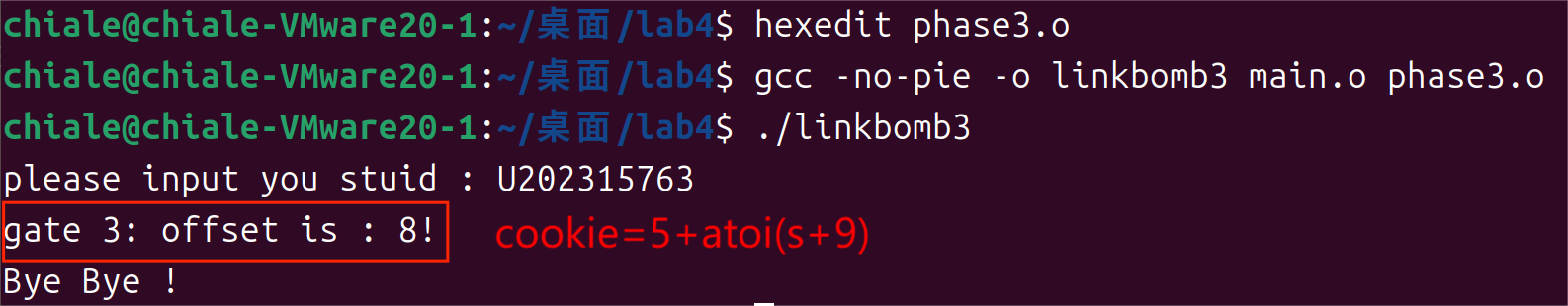
我们需要在do\_phase函数中将参数传到%edi并调用myfun函数



将相应机器码添加到phase3.o文件



链接并运行，正确调用myfunc函数并输出学号尾号+5

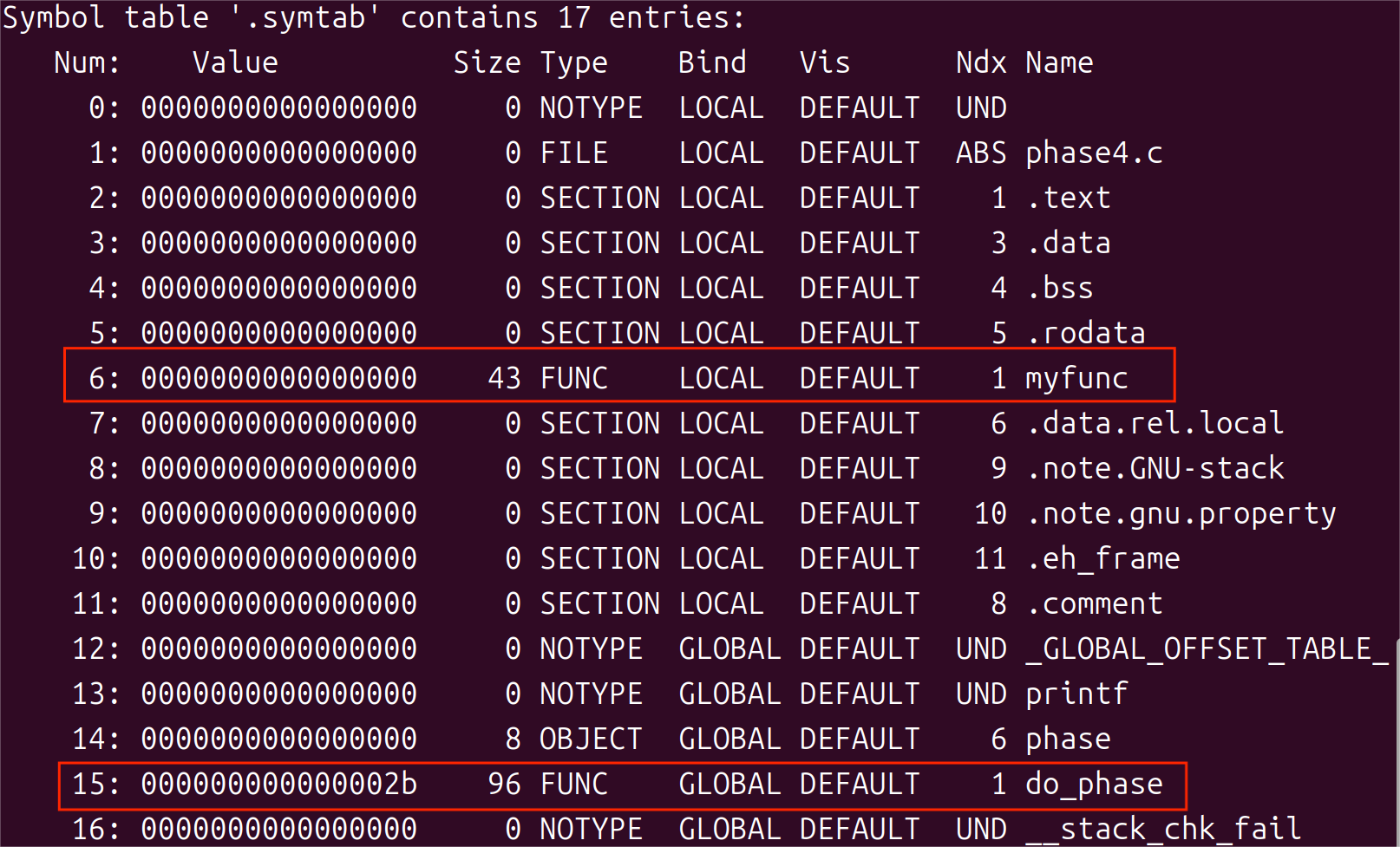


**4.有局部变量的机器指令修改**

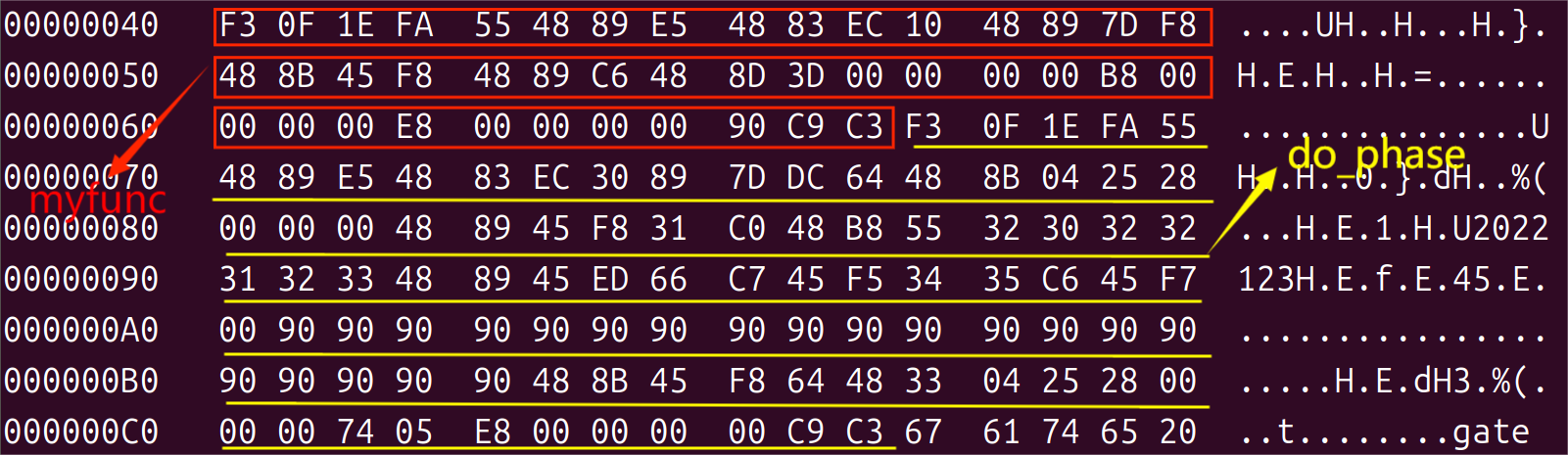
代码节在节头表中的第一号，偏移量为0x40，大小为0x8b



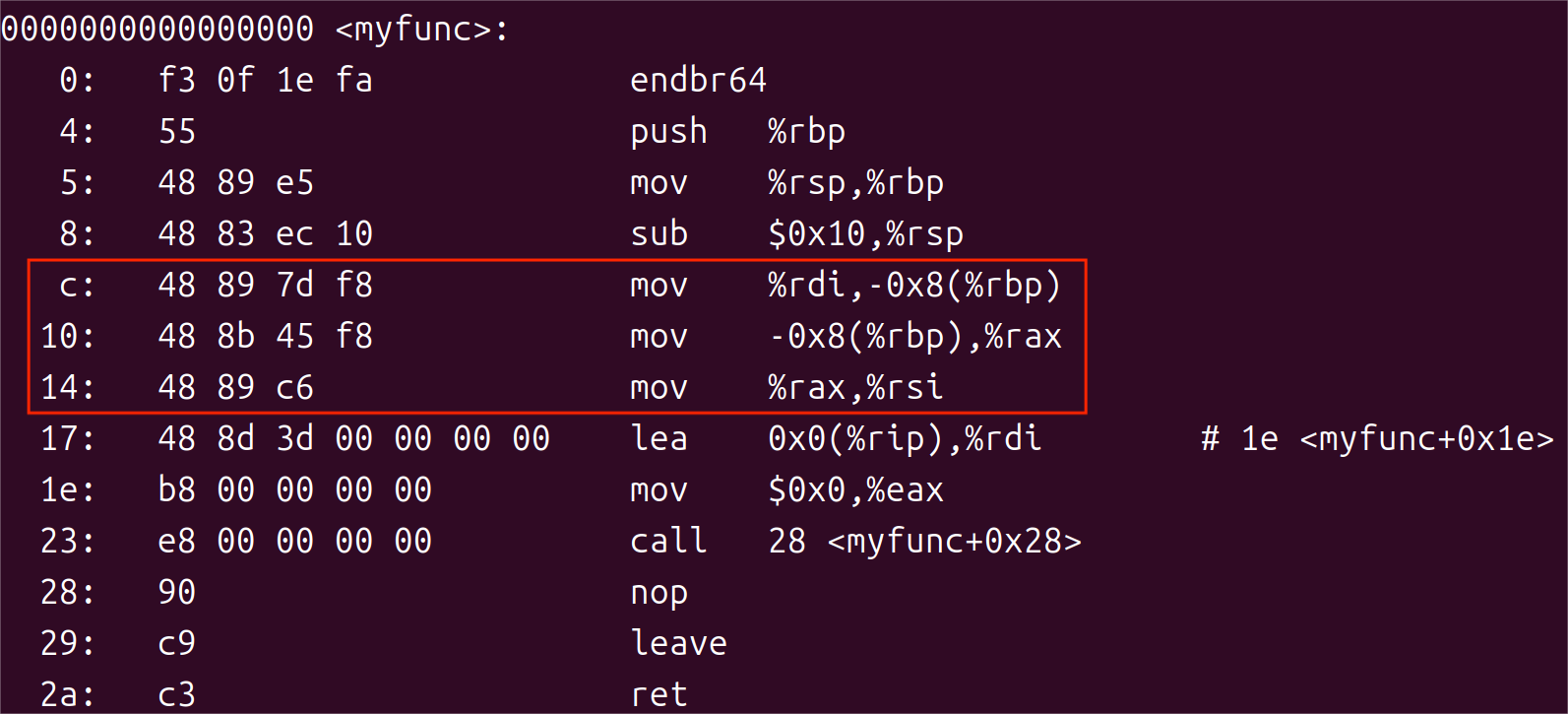
myfunc函数43个字节，偏移量为0；do\_phase函数96个字节，偏移量为0x2b



因此两函数在phase2.o文件中的位置如下图

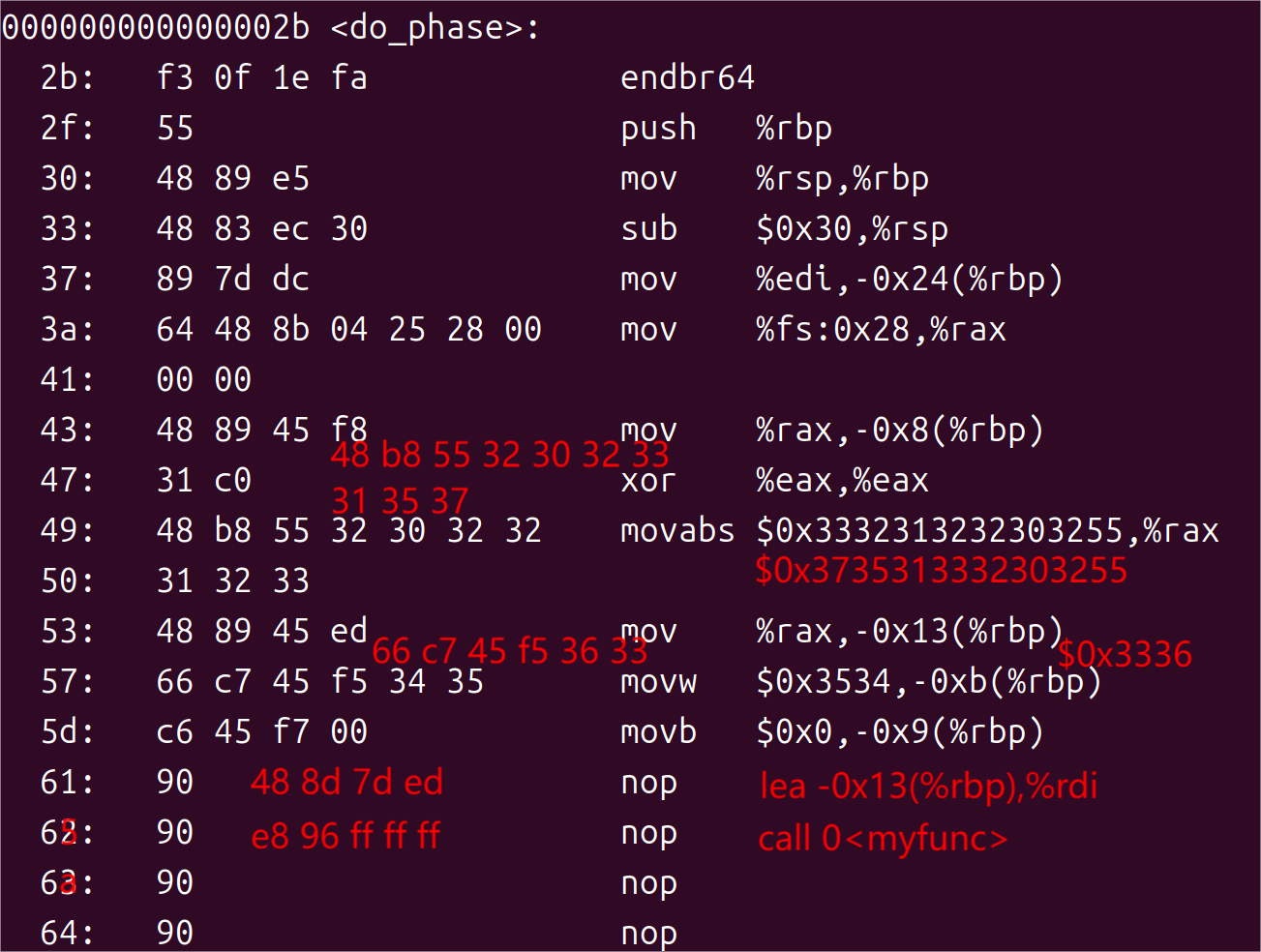


使用objdump -d phase4.o指令反汇编，学号字符串的地址由%rdi传递

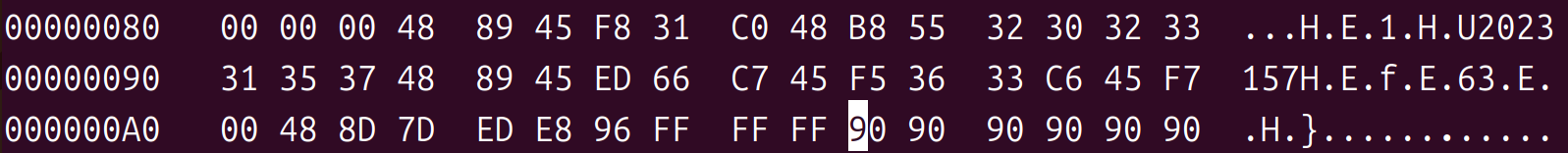


因此需要在do\_phase函数中修改字符串为自己的学号，并将字符串首地址放到%rdi，

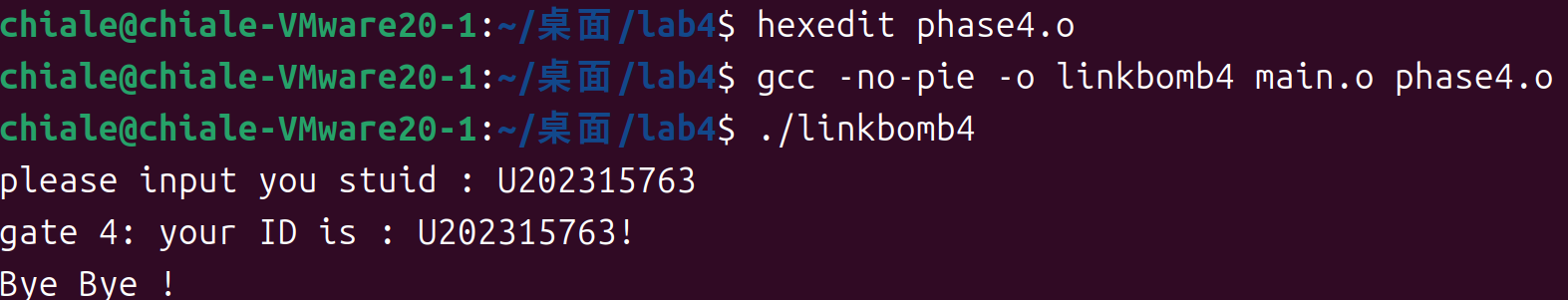
再调用myfunc函数



修改phase4.o中相应的机器码

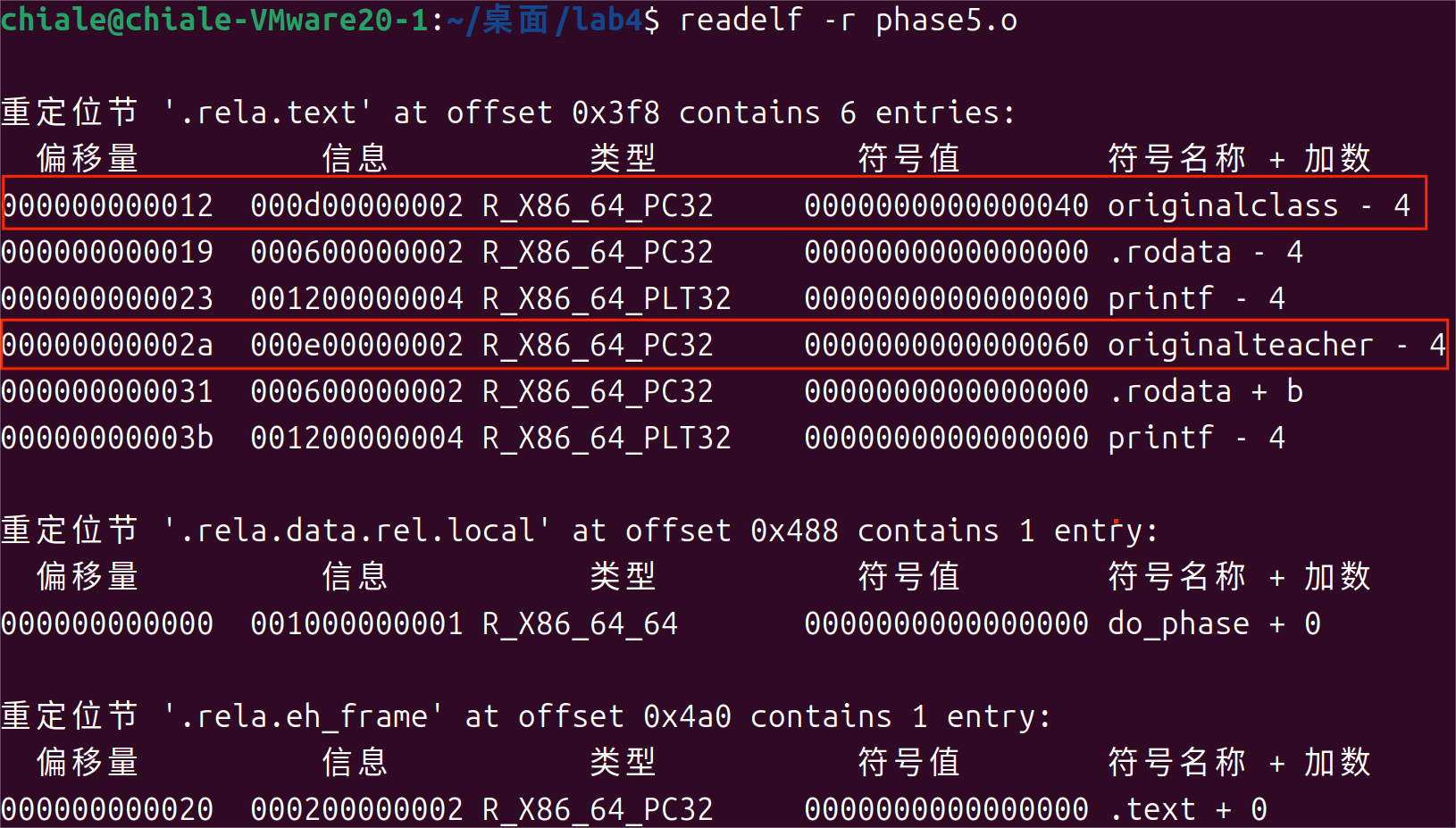


链接并运行，正确调用myfunc函数并输出学号

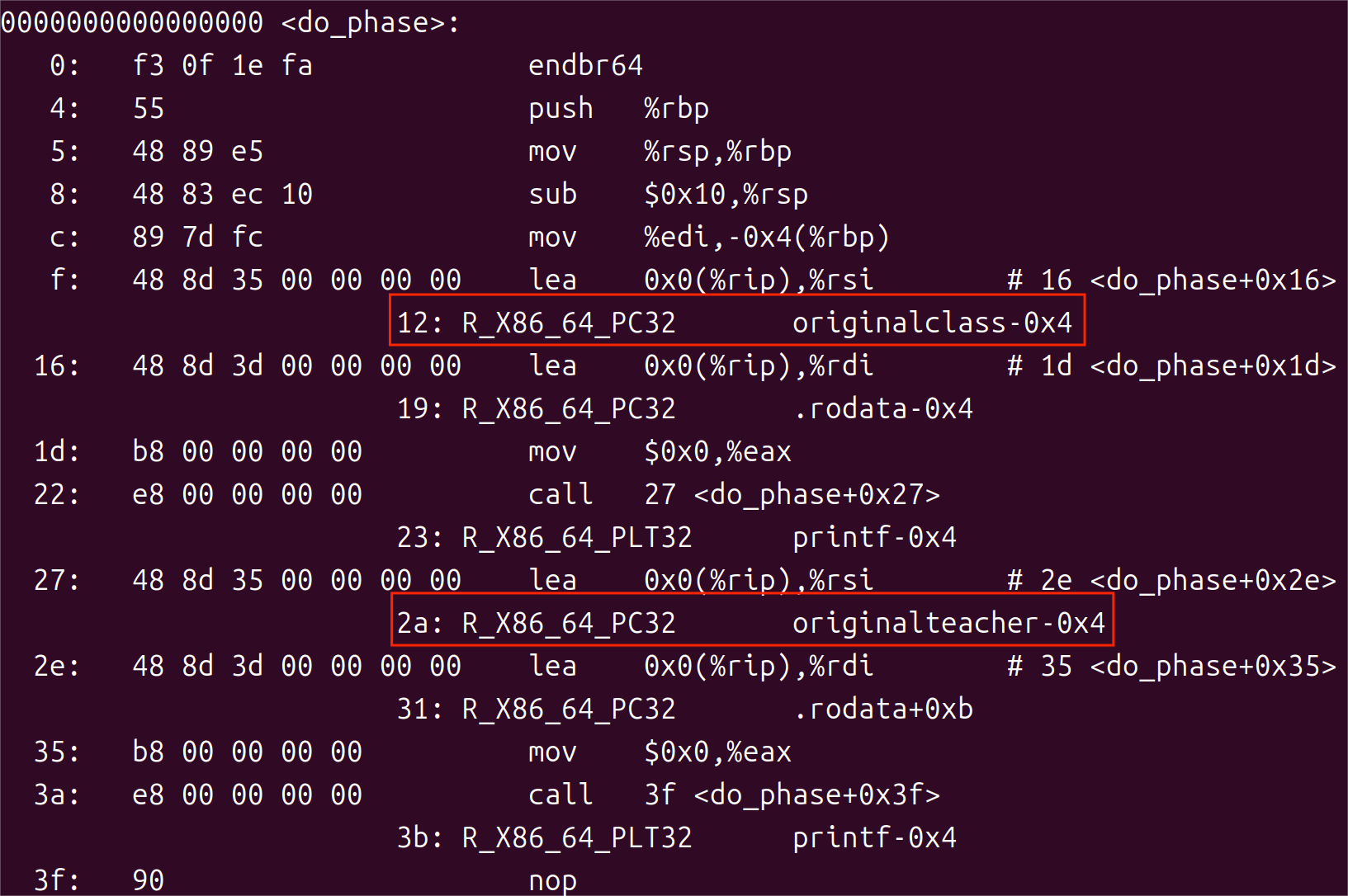


**5.重定位表的修改**

查看重定位节的信息，需要对.text节中偏移量为0x12和0x2a的地方按照PC相对地址 方式进行重定位，所引用符号为phase5.o的符号表中value值为0x40和0x60 的符号

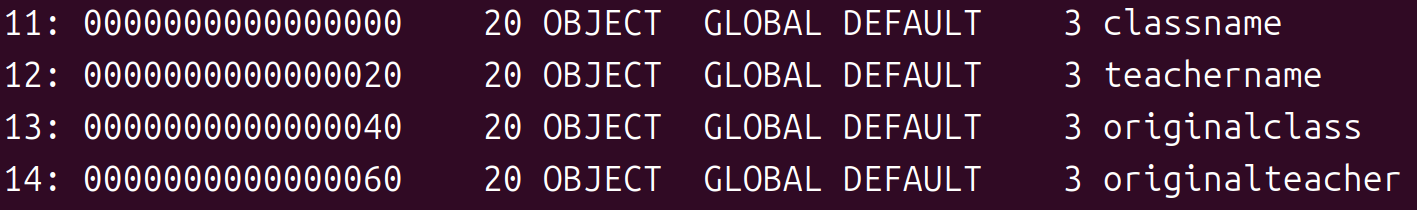


使用objdump -r -d phase5.o指令显示带重定位信息的反汇编

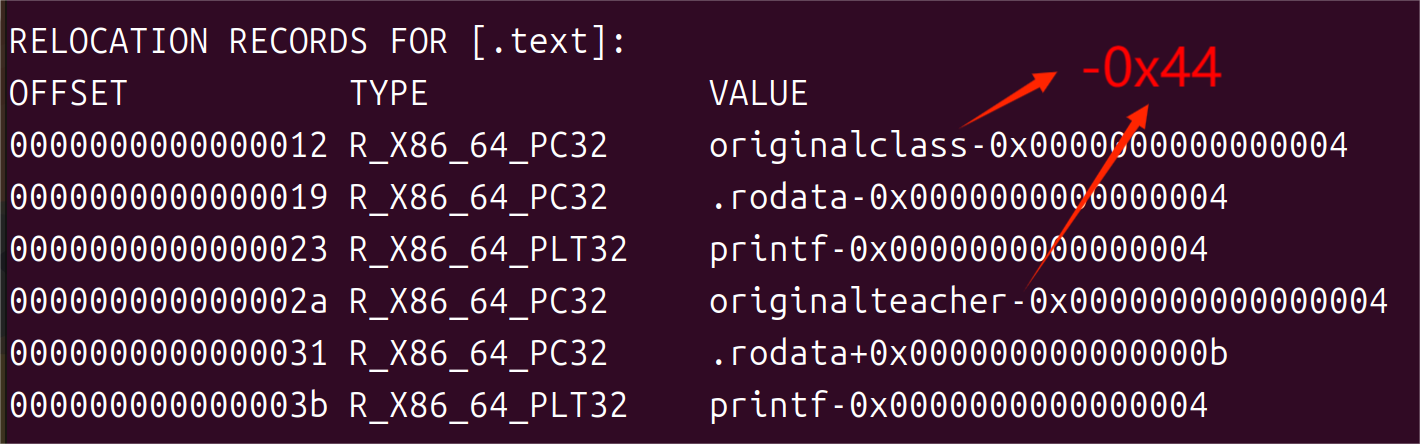


查看originalclass和originalteacher的信息，他们在数据节偏移量为0x40和

0x60的地方



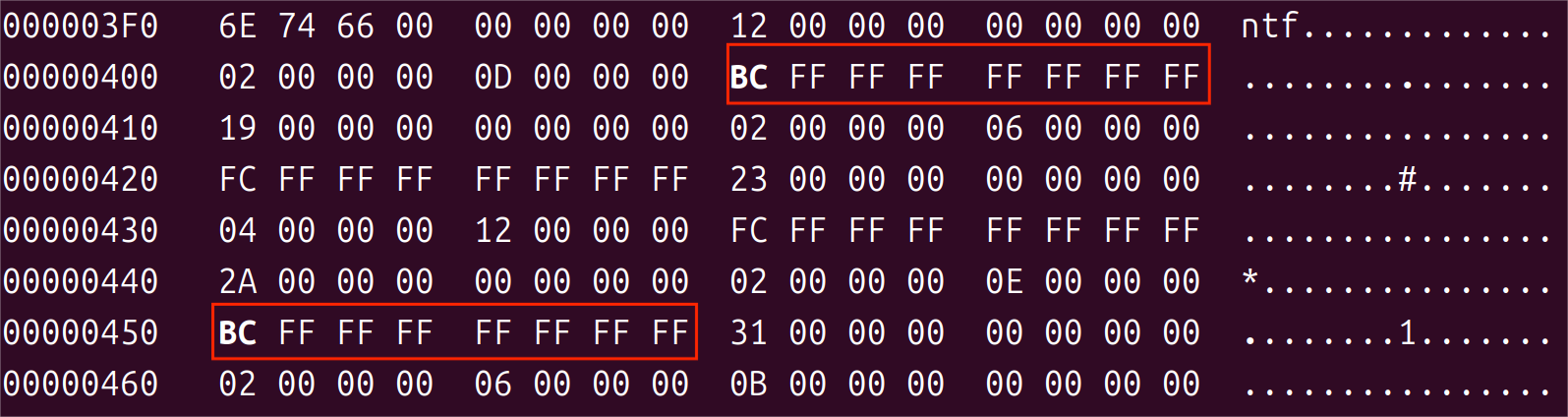
只需修改重定位节的重定位地址即可



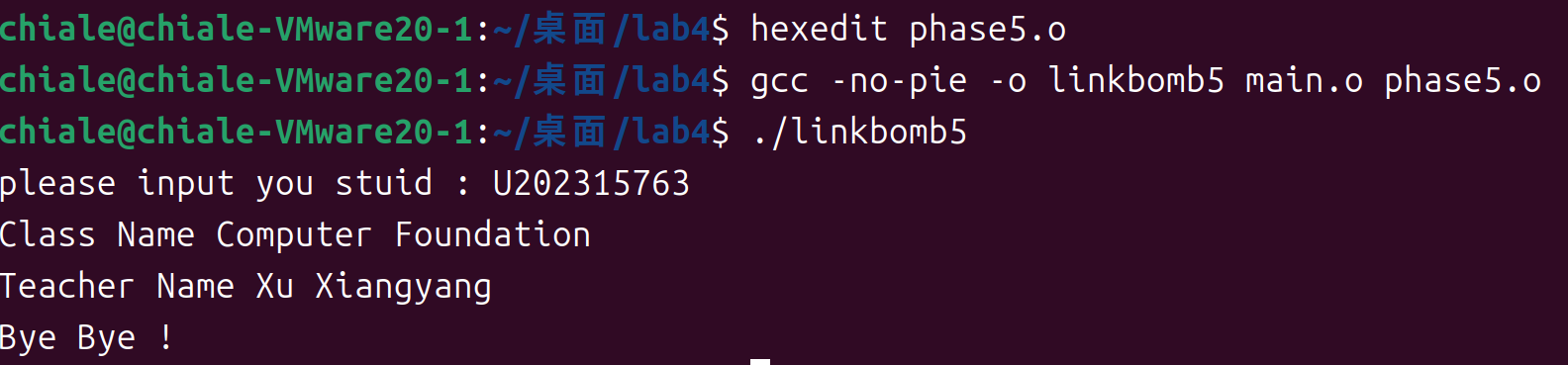
.rela.text节在phase5.o的偏移地址为0x3f8，大小为0x90



-0x44的补码为0xffffffbc修改phase5.o中相应的机器码

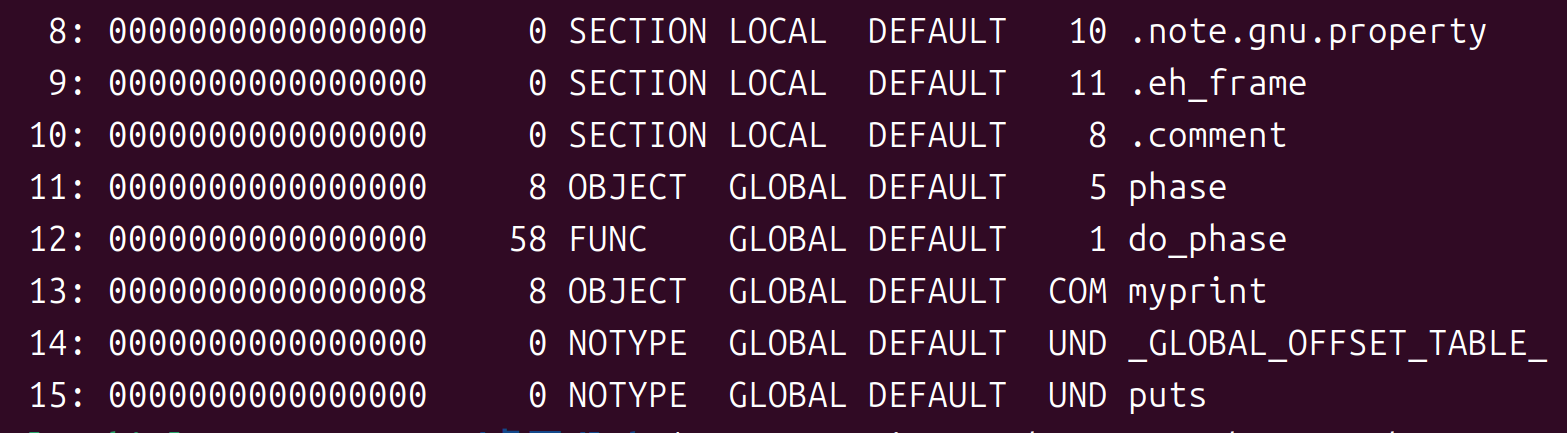


链接并运行，正确输出相应内容

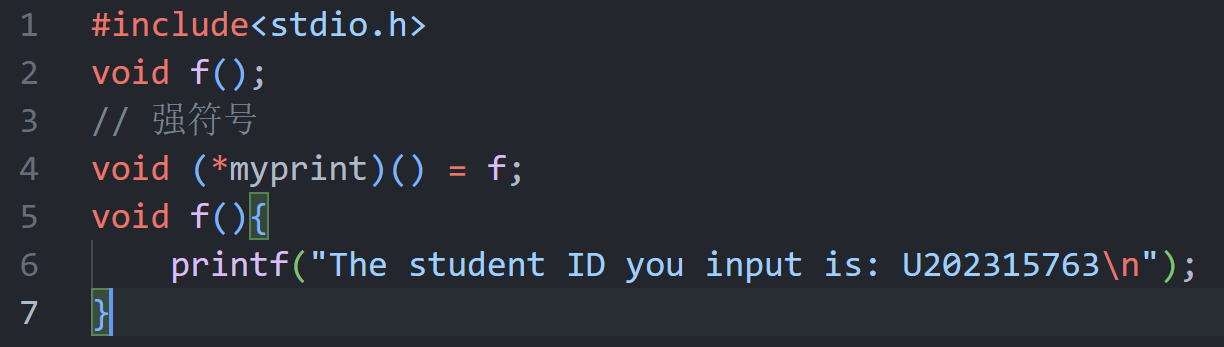


**6.强弱符号**

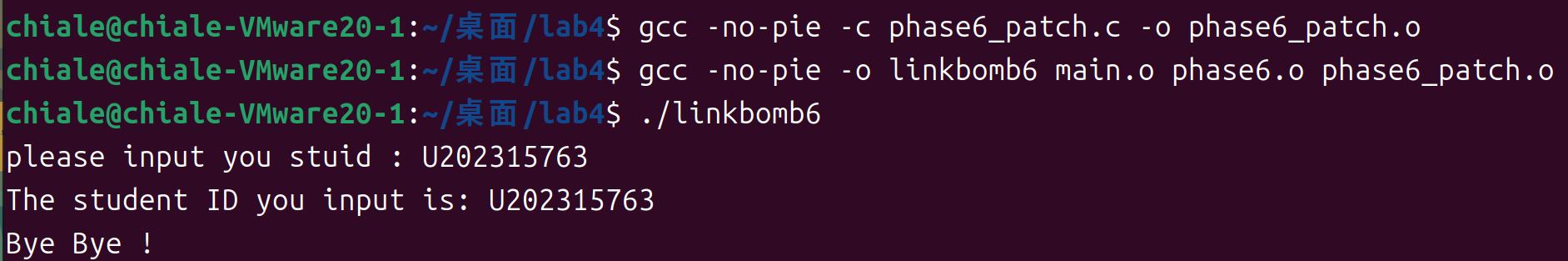
phase6.o中又有一个未定义的全局符号myprint



phase6\_patch.c如下



链接并运行，正确输出学号

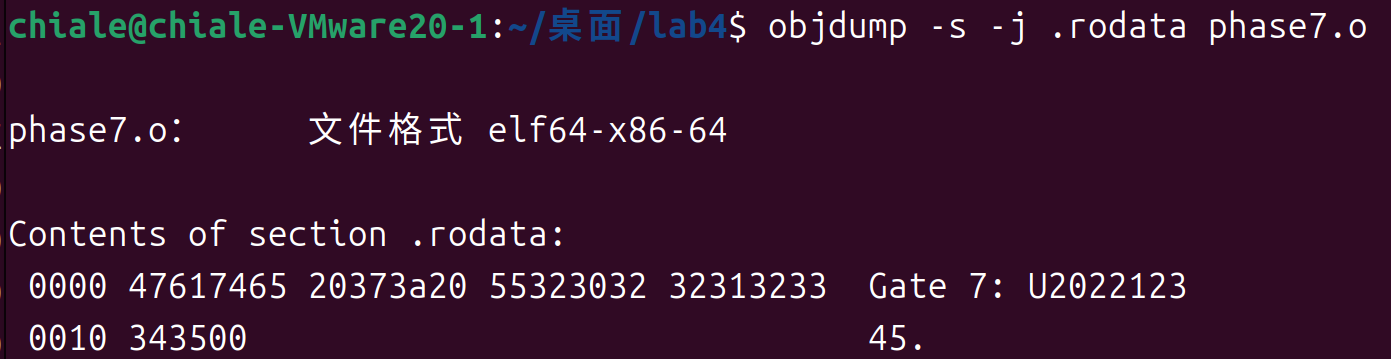


**7.只读数据节的修改**

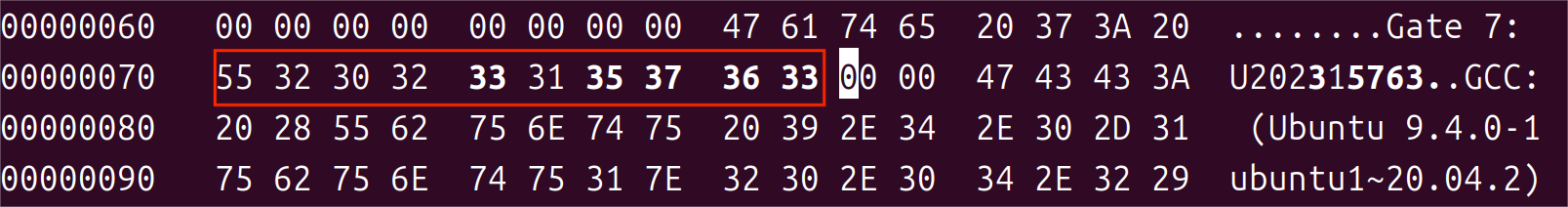
只读数据节偏移量为0x68，大小为0x13



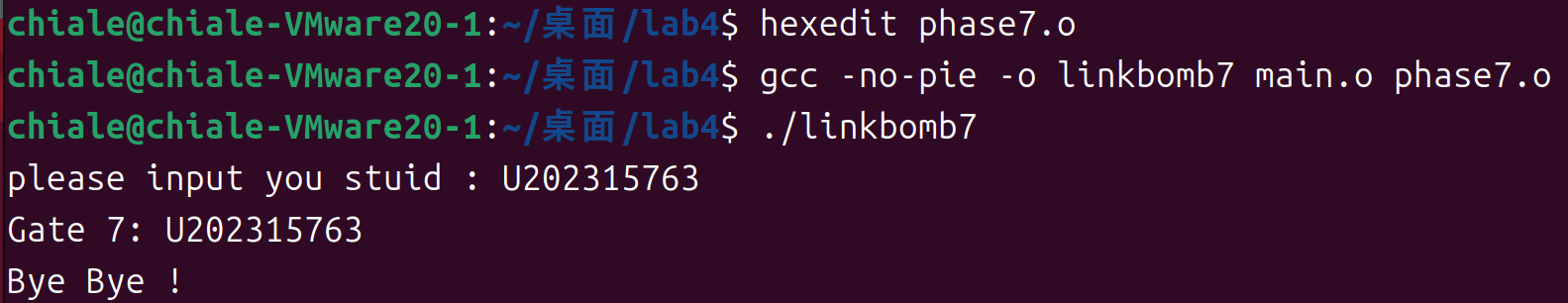
查看.rodata内容



修改phase7.o



链接并运行，正确输出学号



1. **描述修改各个文件的基本思想**

**1.静态数据节的修改**：修改 .data 节中的内容以包含指定的静态数据（学号）。在这种情况下，直接修改 .data 节的内容即可，不会涉及到代码段或重定位项的修改。

**2.机器指令修改**：在 phase2.o 中，通过修改 .text 节的代码，使得函数 do\_phase 可以调用 myfunc 函数。添加 call 指令跳转到 myfunc 的地址，这涉及到相对偏移量的计算和代码节内容的修改。

**3.带参数的函数调用**：修改 .text 节的代码，使得 do\_phase 可以将参数传递给 myfunc。通过调整寄存器（%edi），将参数传递到 myfunc 中。这种方法需要修改机器指令，并调用 myfunc，确保参数的正确传递。

**4.局部变量和字符串传递**：在 phase4.o 中，修改 .text 节代码，使得 do\_phase 能调用 myfunc 并传递字符串（学号）。这种修改需要将字符串地址加载到合适的寄存器中，并调用 myfunc。

**5.重定位表的修改**：通过修改重定位表中的条目，使符号 originalclass 和 originalteacher 指向 .rodata 节中的新字符串地址，从而在运行时正确显示“Class Name: Computer Foundation. Teacher Name: Xu Xiangyang”。

**6.强弱符号的使用**：在 phase6.o 中使用强符号 myprint 替代未定义的全局符号，通过增补一个新文件 phase6\_patch.o，实现输出学号。

**7.只读数据节的修改**：修改 .rodata 节内容，将学号写入该节中，确保只读数据在运行时正确显示。在这种情况下，不涉及代码段的修改，仅更改 .rodata 数据即可。

1. **体会**

在进行ELF文件与程序链接的实验后，我获得了深刻的见解和宝贵的经验。以下是我对这次实验的心得体会：

首先，我深入了解了ELF文件的结构和原理，这使我对程序的编译和链接过程有了全新的认识。ELF文件作为可执行文件的一种标准格式，其内部复杂的组织结构，包括文件头、节表和段表等，为我揭示了程序如何被计算机所识别和执行。通过亲手操作和分析ELF文件，我能够更加清晰地理解程序的构成和运行机制。

其次，在实验中，我掌握了多种强大的工具来解析和编辑ELF文件，如objdump、readelf和hexedit等。这些工具不仅帮助我直观地查看了ELF文件的汇编代码、头部信息和节表内容，还让我能够亲手尝试修改文件内容，从而更深入地了解程序的内部结构。特别是将源代码编译成可重定位文件（.o文件），并进一步链接成可执行文件的过程，让我对C语言程序的编译链接流程有了更加全面的认识。

同时，我也深刻体会到了修改ELF文件可能带来的风险。在实验过程中，我尝试修改了节表的某些属性或大小，结果导致程序无法正常运行。这让我意识到，在修改ELF文件时需要格外小心，必须确保对修改的后果有充分的认识和准备。这也让我更加明白，在编程和软件开发中，每一个细节都可能影响到程序的稳定性和正确性。

此外，与同学们的交流讨论让我受益匪浅。通过互相学习和分享经验，我们共同解决了实验中遇到的难题，也从彼此的观点中获得了新的启示。这种团队合作和互助精神不仅让我感受到了学习的乐趣，也让我更加珍惜与同学们共度的时光。

总的来说，这次ELF文件与程序链接的实验不仅加深了我对计算机系统基础知识的理解，还提高了我的实践能力和问题解决能力。我相信这些宝贵的经验和收获将对我未来的学习和工作产生深远的影响。