

**课 程 实 验 报 告**

**课程名称： 计算机系统基础**

**实验名称： ELF文件与程序链接**

**院 系 ：**

**专业班级 ：**

**学 号 ：**

**姓 名 ：**

**指导教师 ：**

**年 月 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**

**任务** 链接炸弹的拆除

**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。**

**1、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。

**2、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!。

**3、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。

**4、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。

**5、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。

**6、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。

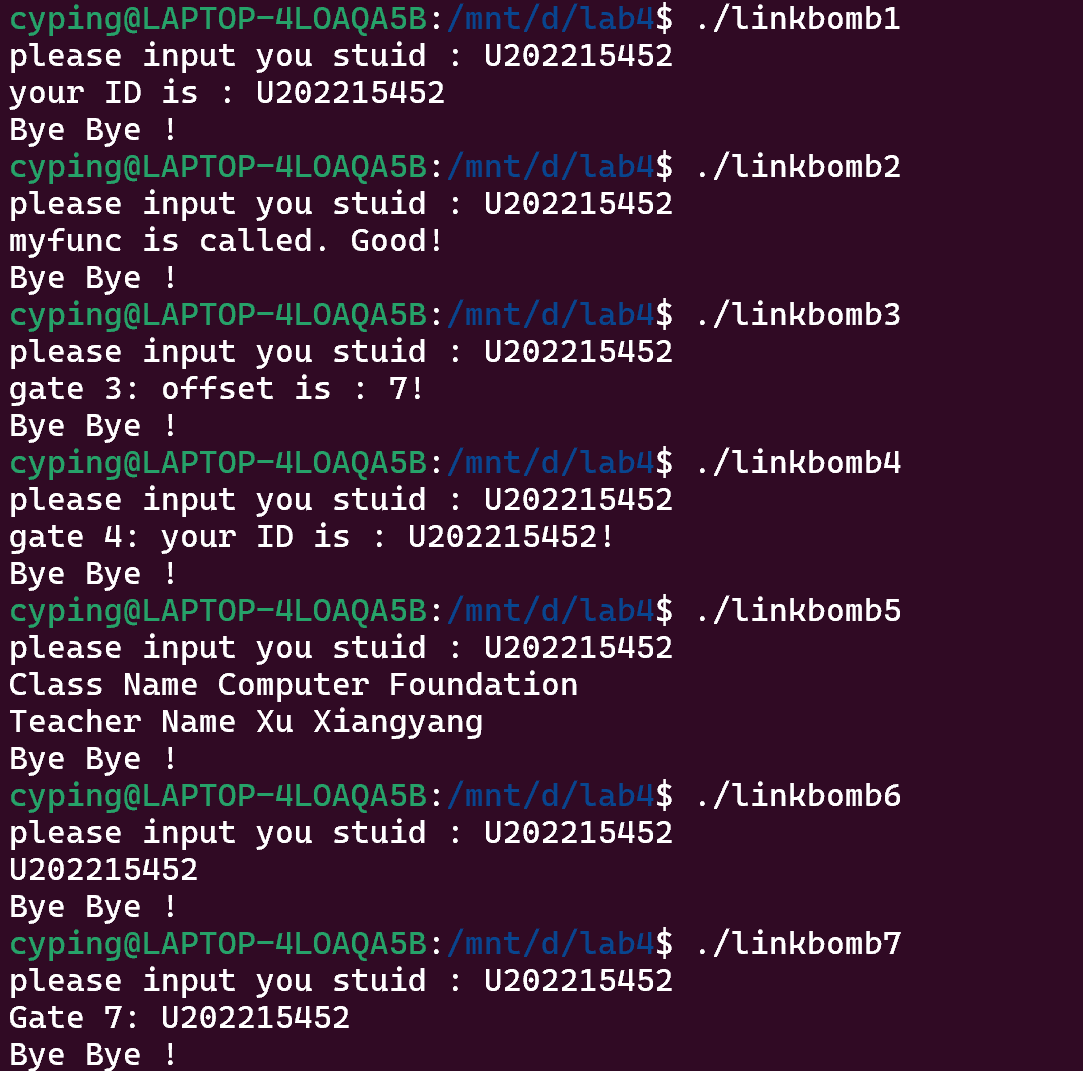
#gcc -no-pie -o linkbomb6 main.o phase6.o phase6\_patch.o

**7、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）实验任务的实验记录**



完整演示结果

**（2） 描述修改各个文件的基本思想**

**第1关 静态数据与ELF数据节**

如果不对phase1.o进行修改，直接链接并运行linkbomb1，会得到如下结果：

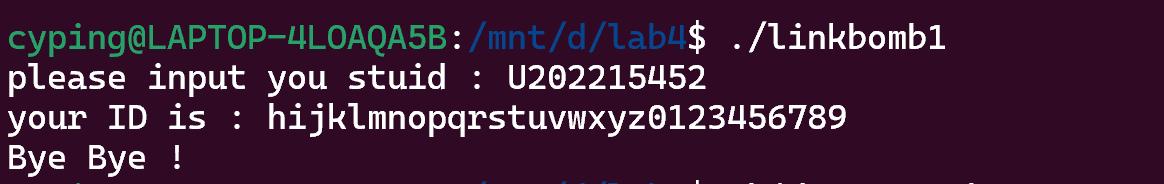


图1-1. 直接运行结果

我们使用hexedit进入phase1.o文件，红框对应的地方为原本输出的字符串hijklmnopqrstuvwxyz0123456

789，修改其对应位置的ASCII码使得输出字符串为U202215452。

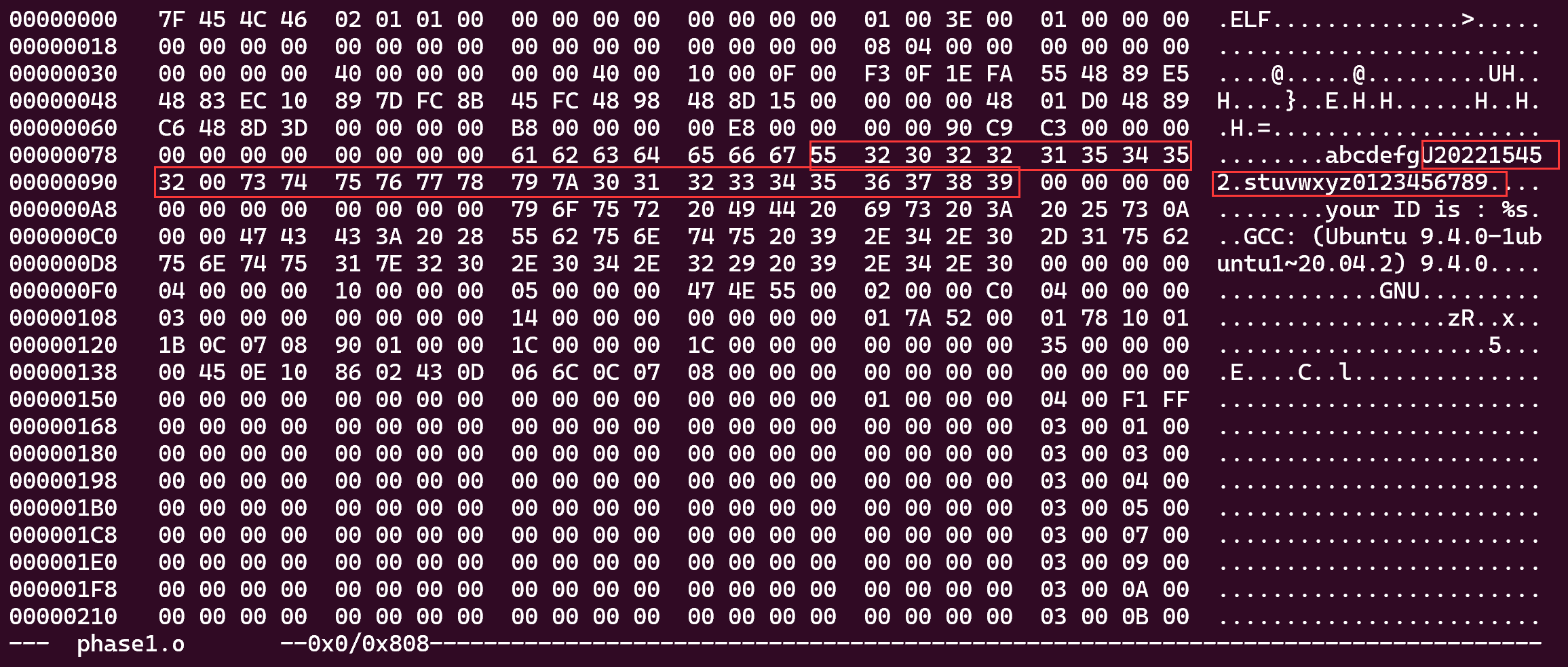


图1-2. hexedit内页面

Ctrl+W保存并退出后再次用gcc指令链接编译就可以获得我们想要的结果。

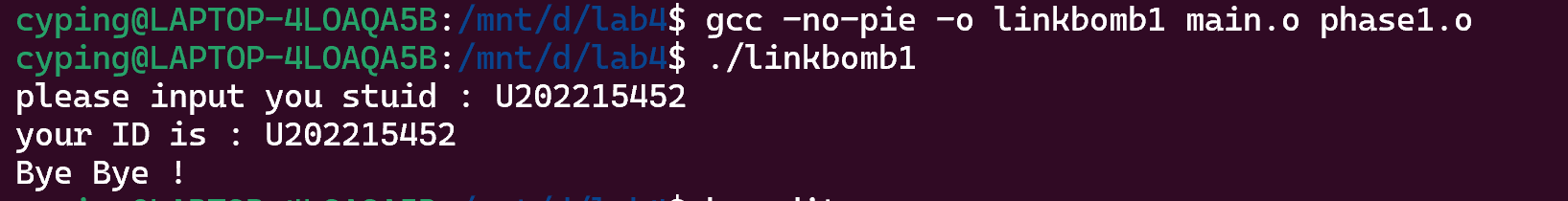


图1-3. 最终结果

**第2关 简单的机器指令修改**

首先查看反汇编代码，发现<do\_phase>函数后执行了很多空操作，我们只需要将22: nop语句改成callq <myfunc>即可。

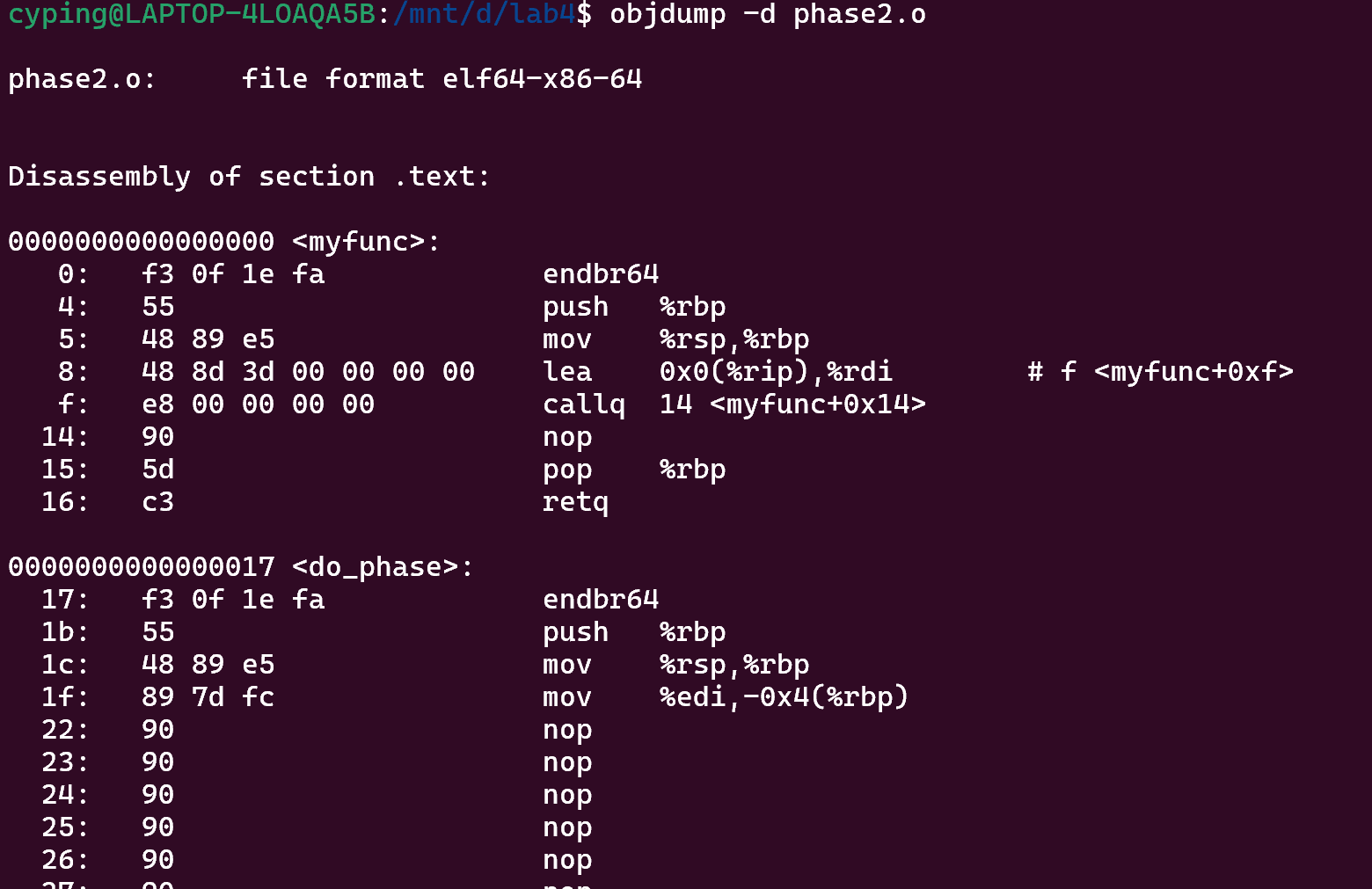


图2-1. phase2.o反汇编

由于call指令占五个字节，且指令对应机器码为E8，所以下一条指令位置为27，又有<myfunc>首地址为0，则填入-27补码即0xFFFFFFD9。通过hexedit修改相应位置十六进制数，结果如图：

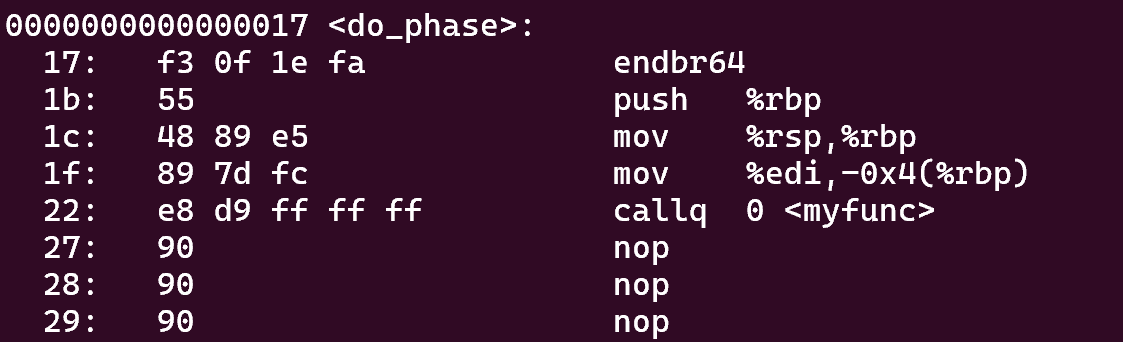


图2-2. phase2.o修改后反汇编

然后链接编译就可以获得我们想要的结果。

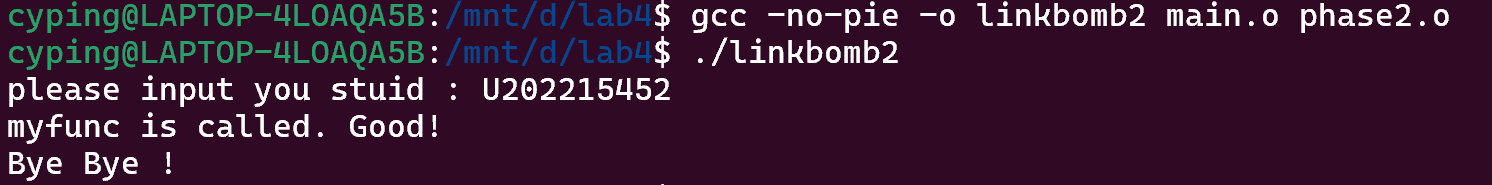


图2-3. 最终结果

**第3关 有参数的函数调用的机器指令修改**

与前一关类似，首先查看phase3.o的反汇编代码：

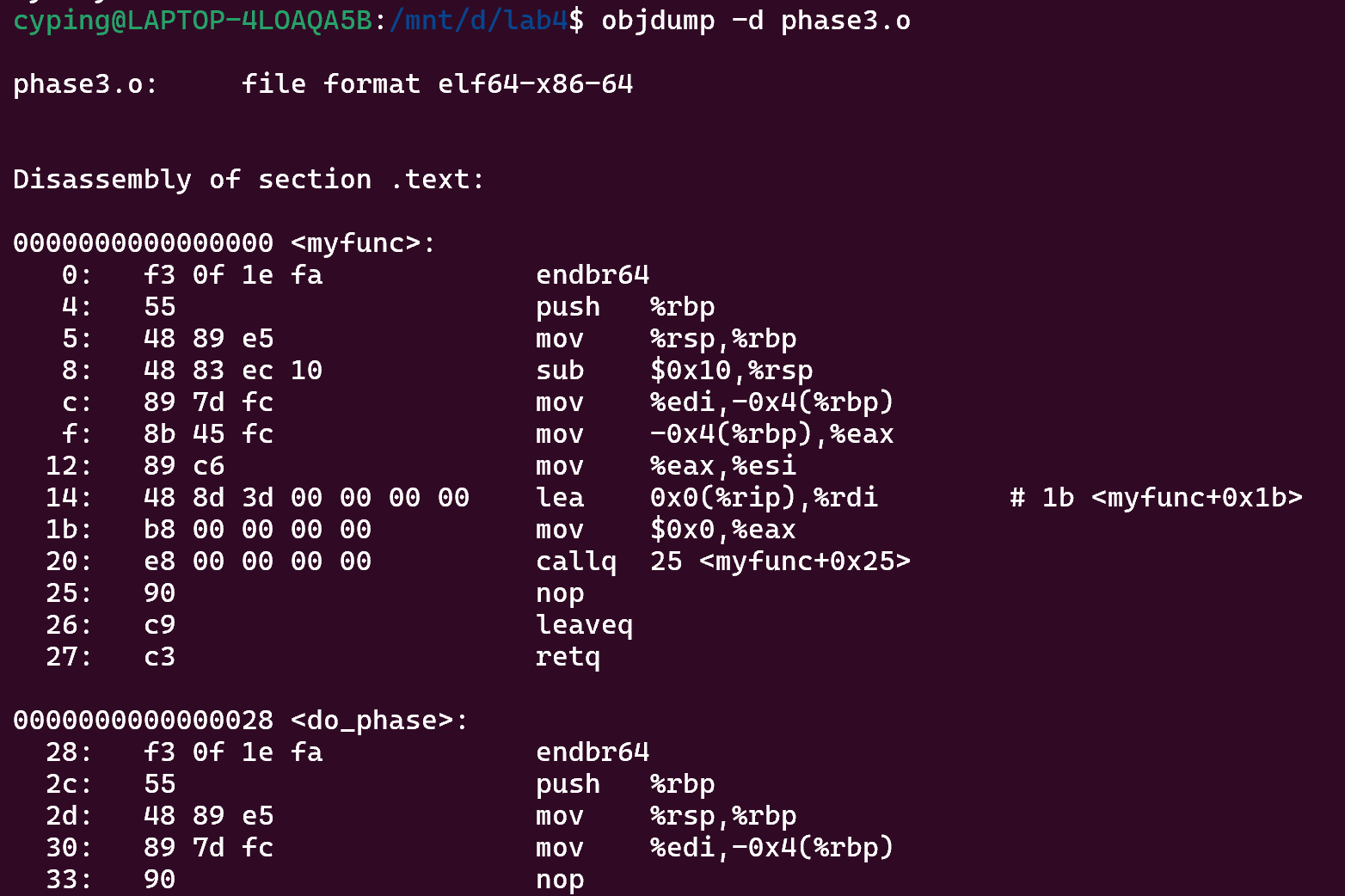


图3-1. phase3.o反汇编

<do\_phase>函数后面依然有许多空操作。观察代码，我们可以发现，-0x4(%rbp)内应该存储的是offset的值，在<myfunc>函数中已经将这个值移入参数位%eax，所以我们只需要在33的位置加上一个callq <myfunc>指令即可。过程与上一关类似。

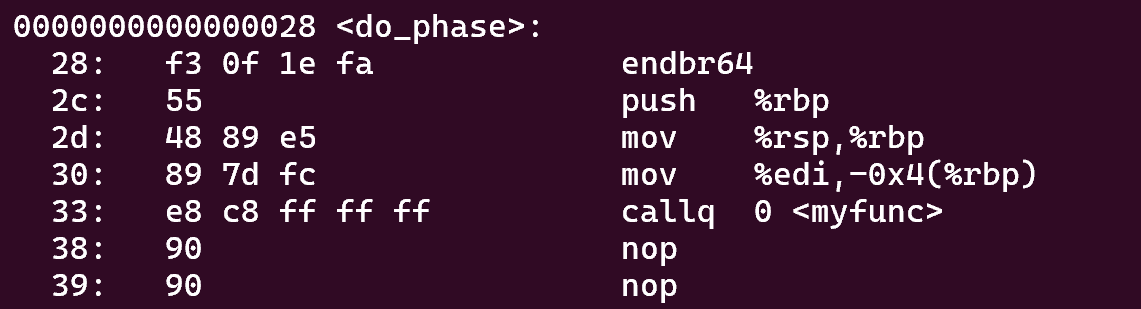


图3-2. phase3.o修改后反汇编

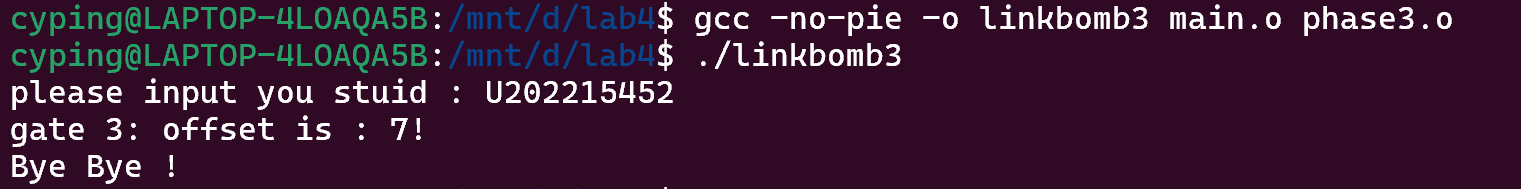


图3-3. 最终结果

**第4关 有局部变量的机器指令修改**

首先查看<main>函数的反汇编代码，可以看到输入的学号信息存入了-0x14(%rbp)的位置。通过gdb调试可以发现，在进入<do\_phase>函数后，%rbp的值减少了0x40，此时学号信息位置应该为0x2c(%rbp)。

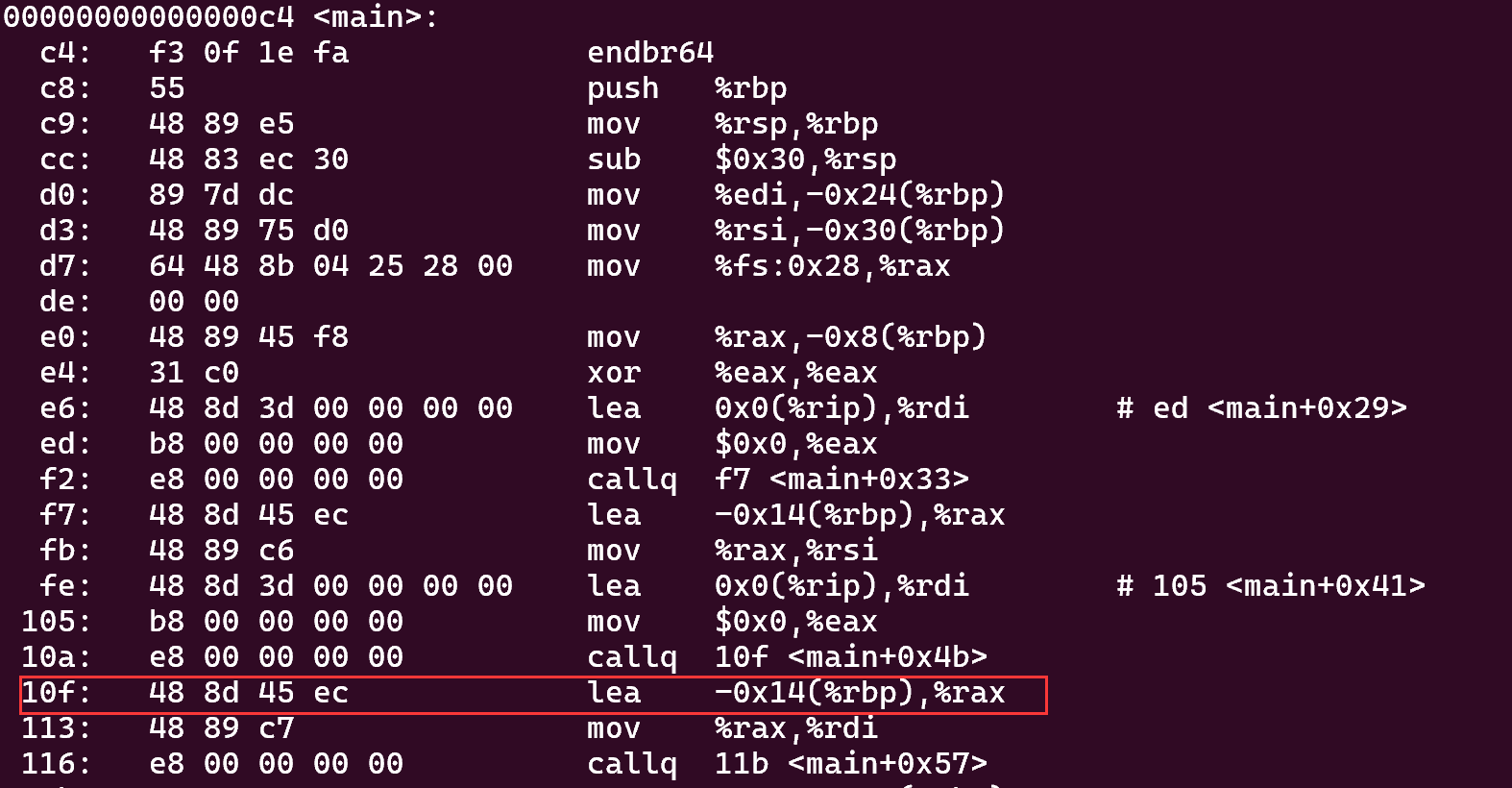


图4-1. main.o反汇编

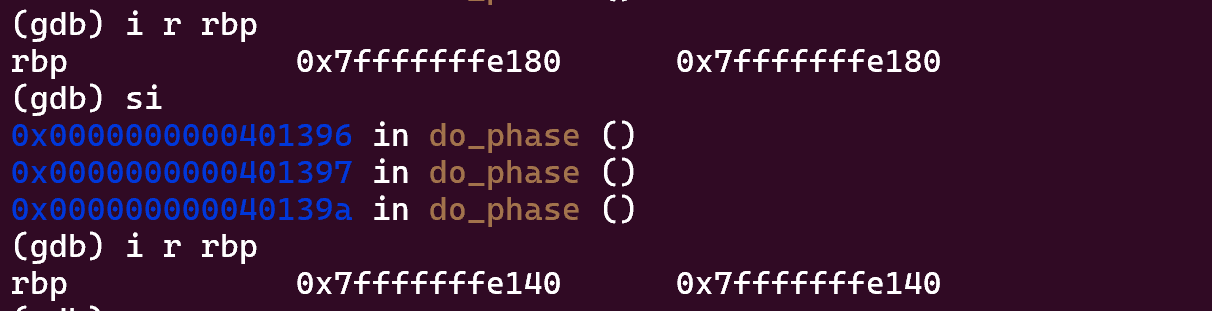


图4-2. 查看rbp寄存器信息

在<myfunc>函数的反汇编中，我们可以看到输入的学号首地址应该存入%rdi中，所以可以编写一个.S文件，内容为lea 0x2c(%rbp),%rdi，然后生成可执行文件再反汇编，可以看到其机器指令码。

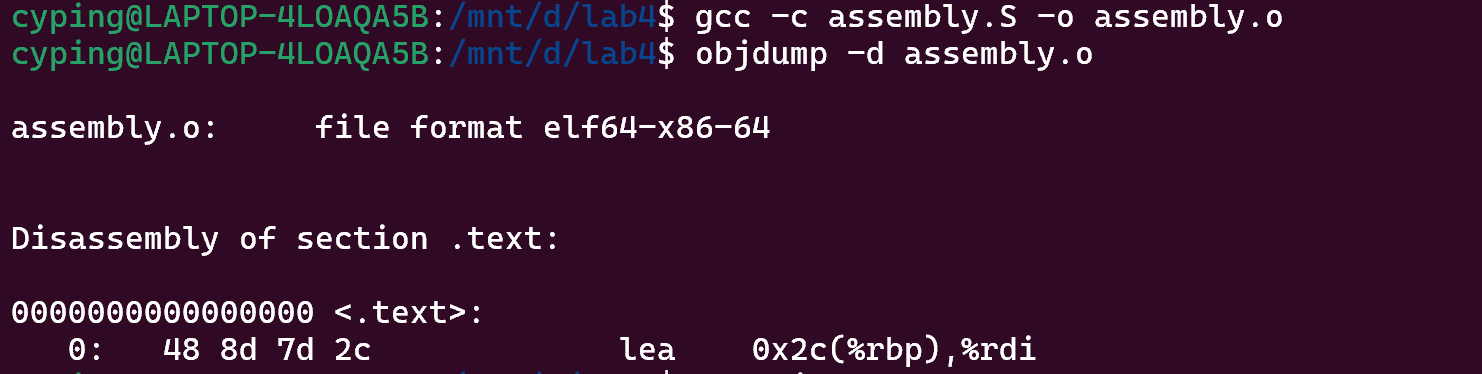


图4-3. 查看机器指令码

最后，再加上与前几关类似的call指令的机器码，通过hexedit添加到<do\_phase>函数后面，如下图所示：

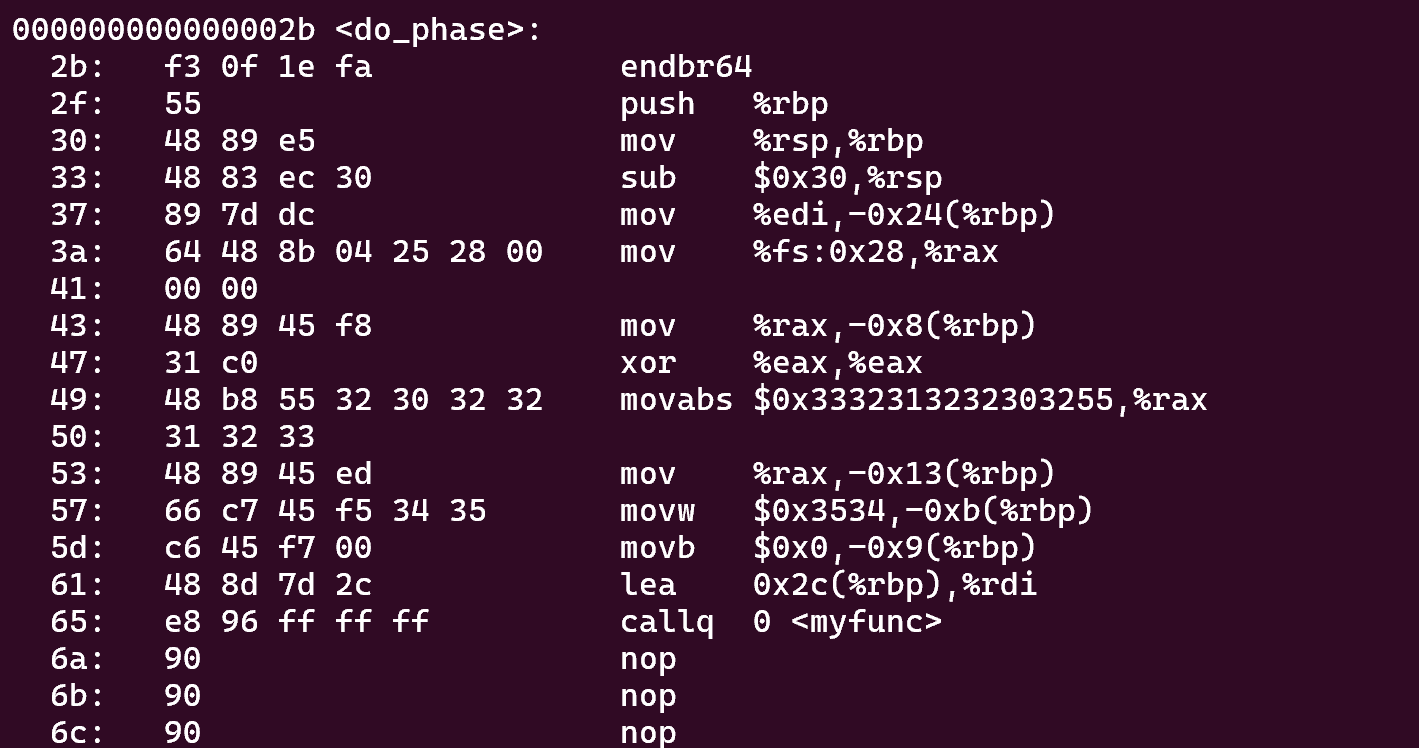


图4-4. phase4.o修改后反汇编

最后再编译生成可执行文件，结果如下：

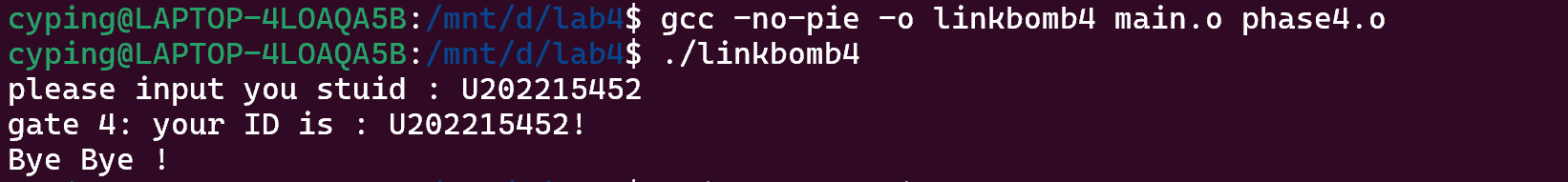


图4-5. 最终结果

**第5关 重定位表的修改**

我们可以尝试直接编译运行，会发现可以输出结果，但是课程名和教师名并不符合需要。

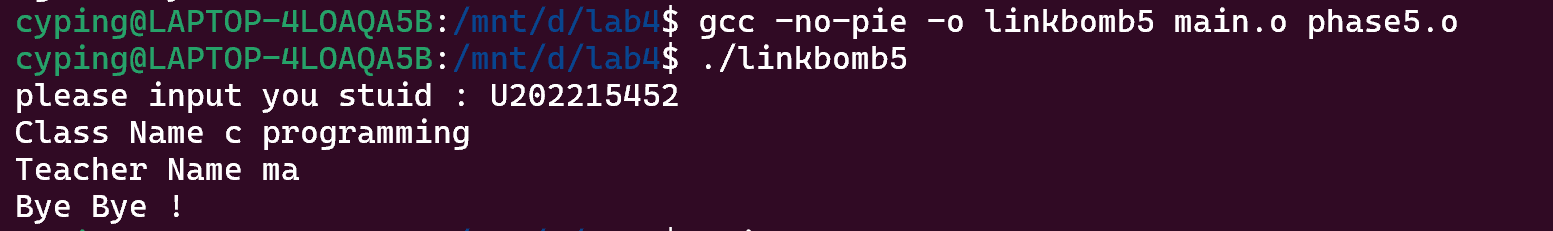


图5-1. 直接运行结果

查看重定位信息，originalclass和originalteacher的信息错误，分别指向了“c programming”和“ma”。这时再查看数据节内容信息，发现正确的信息地址与错误的相隔-0x40，所以应该将重定位的信息修改为 - 44和 - 44。

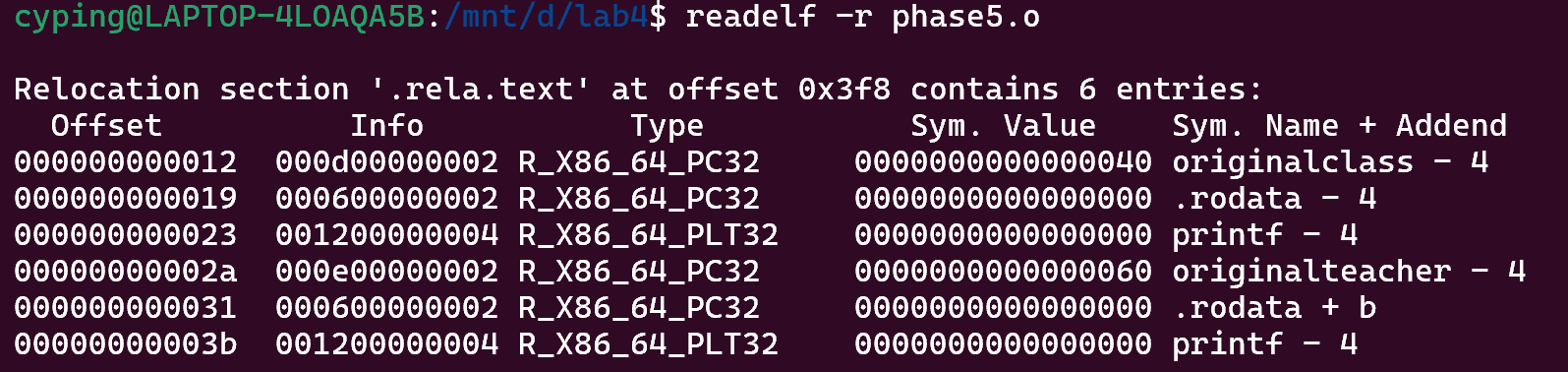


图5-2. 重定位信息

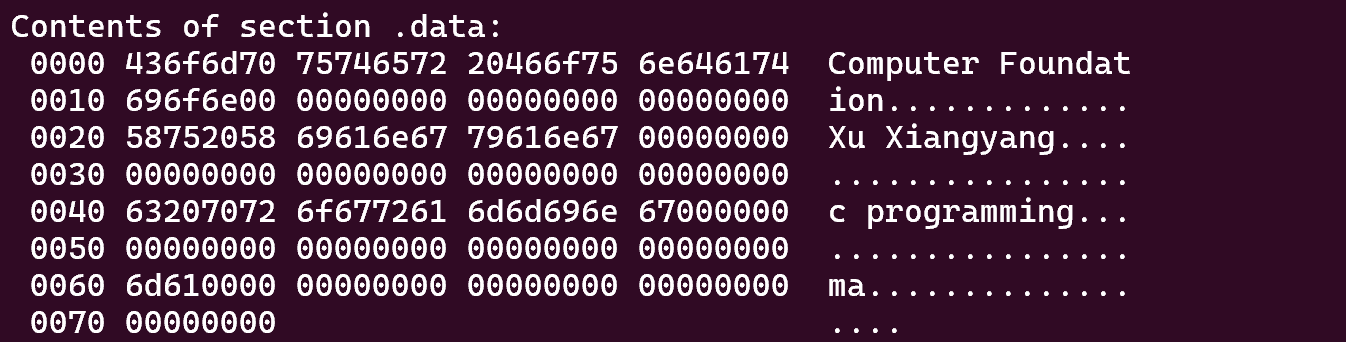


图5-3. 数据节内容

在hexedit中找到相对应的位置，然后修改为-44的补码，如下图所示：

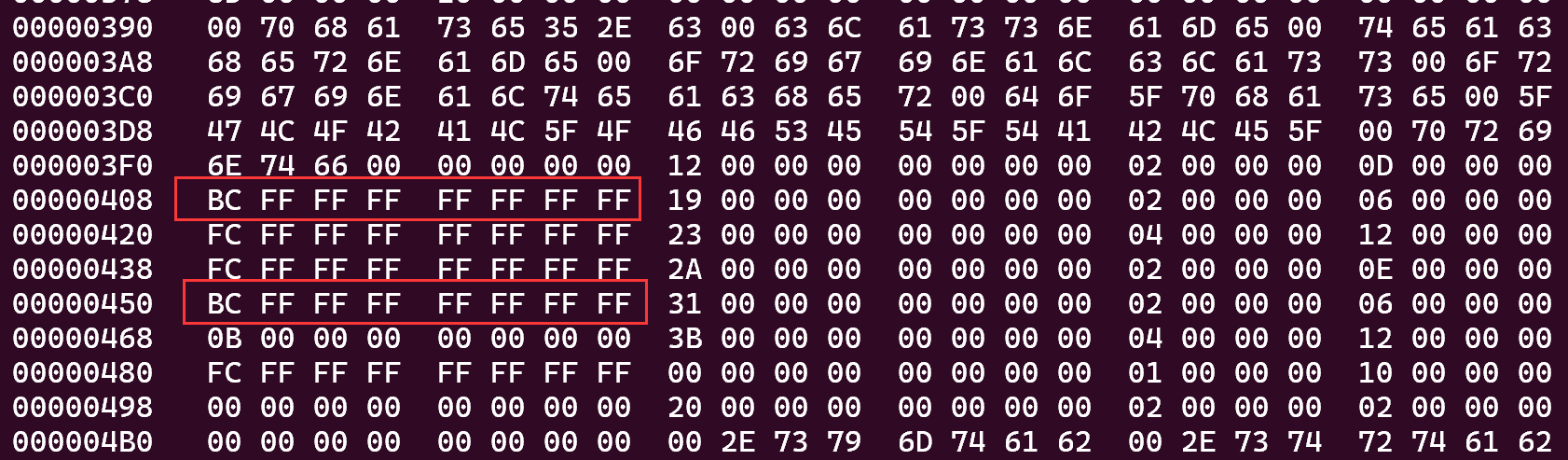


图5-4. hexedit修改内容

最后再次编译运行，即可输出正确结果。

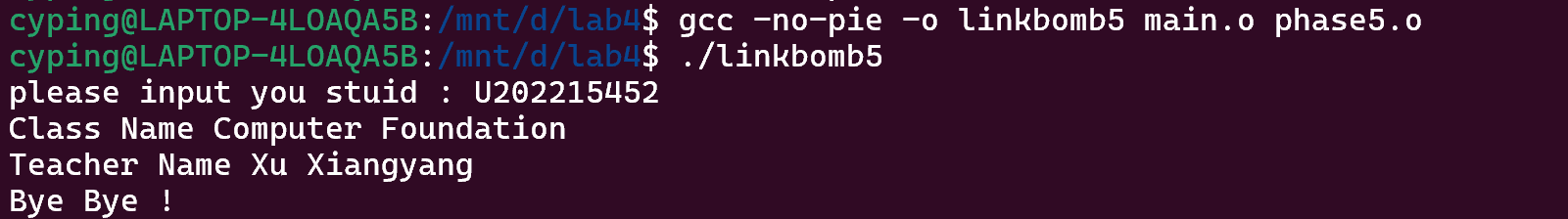


图5-5. 最终结果

**第6关 强弱符号**

首先编译链接，然后查看linkbomb6.o的反汇编代码，发现提到了一个名为<myprint>的函数指针，使用方法与phase0.c内的phase相似。

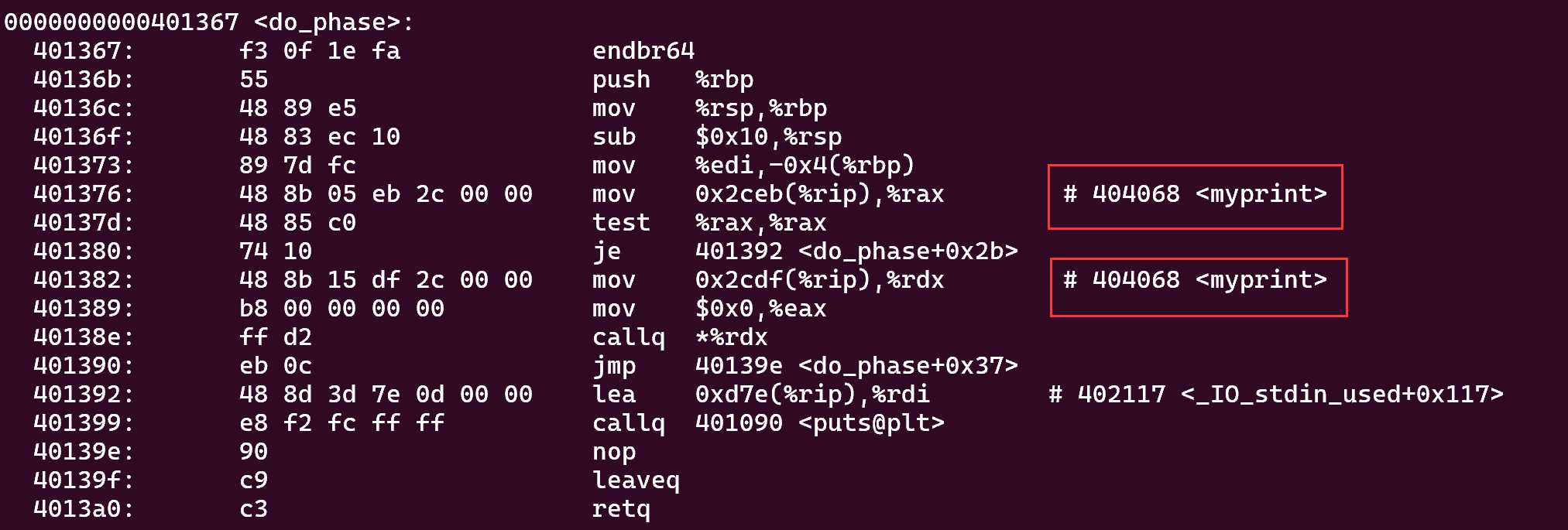


图6-1. linkbomb6反汇编

因此，我们需要定义一个新的函数。这里我将其命名为<myprintf>，函数实现的功能是输出自己的学号，并使得myprint这个函数指针指向这个函数。将以上内容写进一个新的.c文件内，然后链接编译，就可以实现我们想要的功能了。

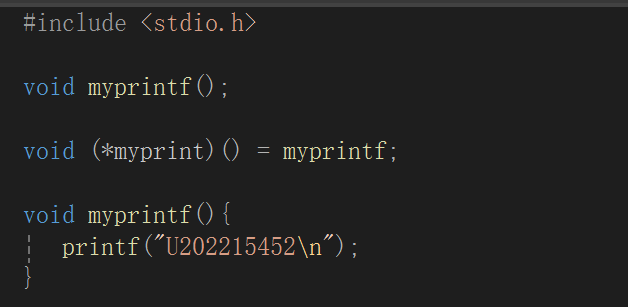


图6-2. phase6\_patch.c文件内容

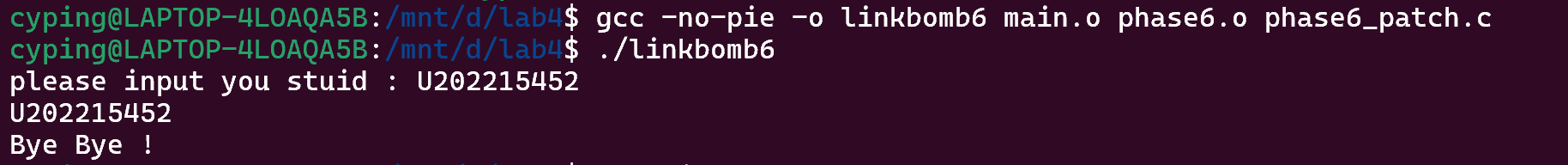


图6-3. 最终结果

**第7关 只读数据节的修改**

首先尝试一下直接运行，发现输出学号为U202212345。

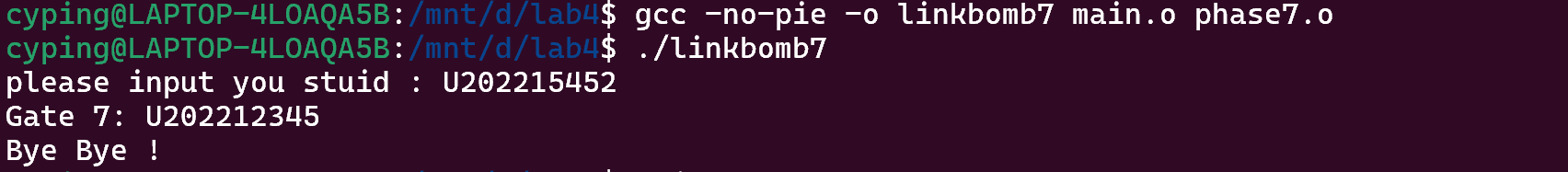


图7-1. 直接运行结果

然后查看只读数据节的内容，发现输出字符串是直接储存在只读数据节中的，只需要将其修改为自己的学号即可。

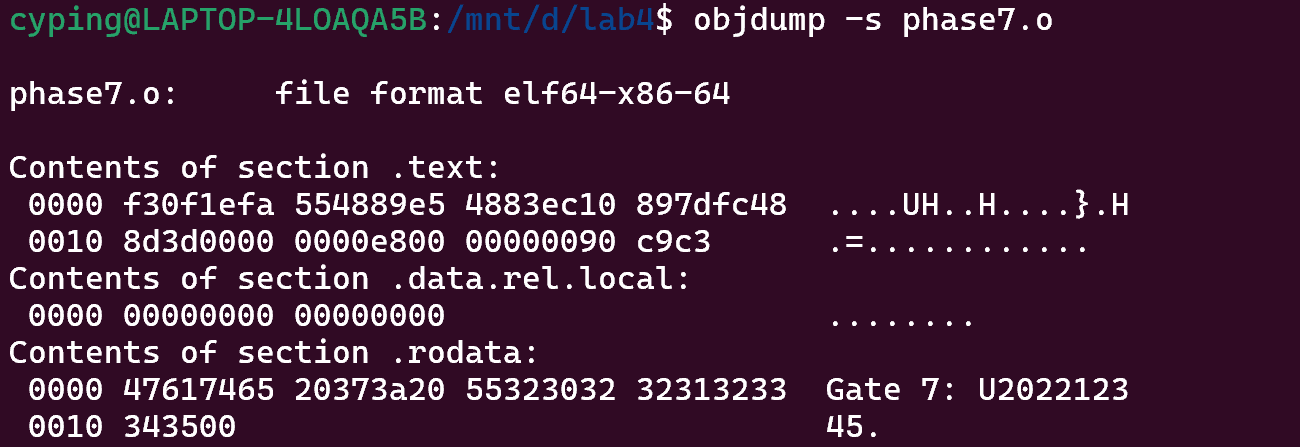


图7-2. 只读数据节内容

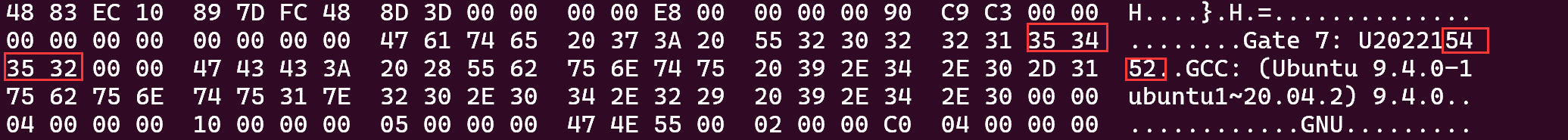


图7-3. 使用hexedit进行修改

再用hexedit对相应位置进行修改后，重新链接编译运行，结果就可以正确输出我们自己的学号了。

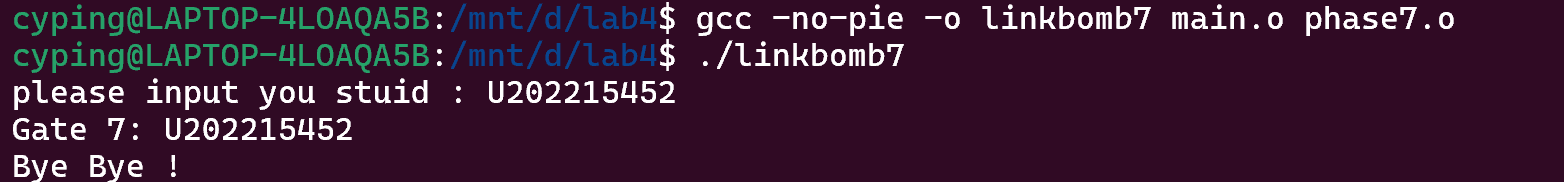


图7-4. 最终结果

1. **体会**

在本次实验中，我们通过几个简单的关卡进一步熟悉了objdump、gcc等工具的使用，了解了一些基本操作的命令语句格式，并通过实践了解了可重定位目标文件中各个节的内容和含义。在逐一破解关卡的过程中，我们也能够更加深入地了解可重定位目标文件的组成结构以及编译链接运行的基本原理。