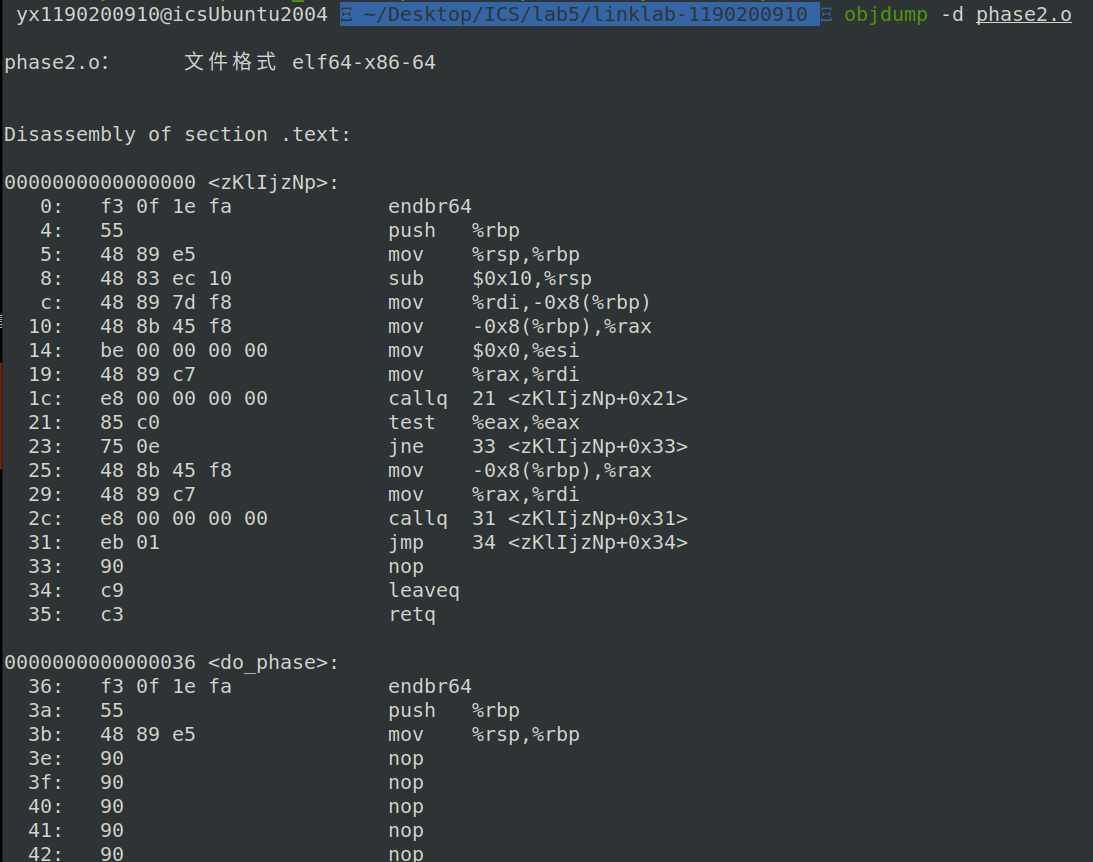
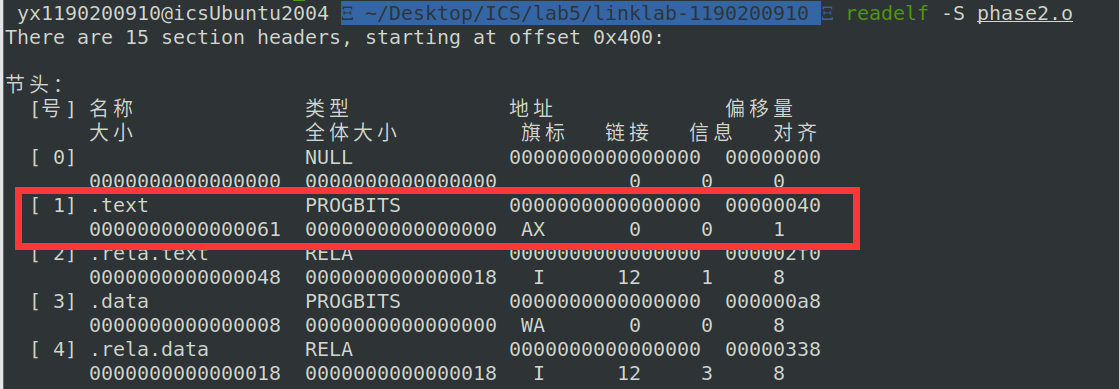
先通过objdump -d看一下phase2.o的反汇编：



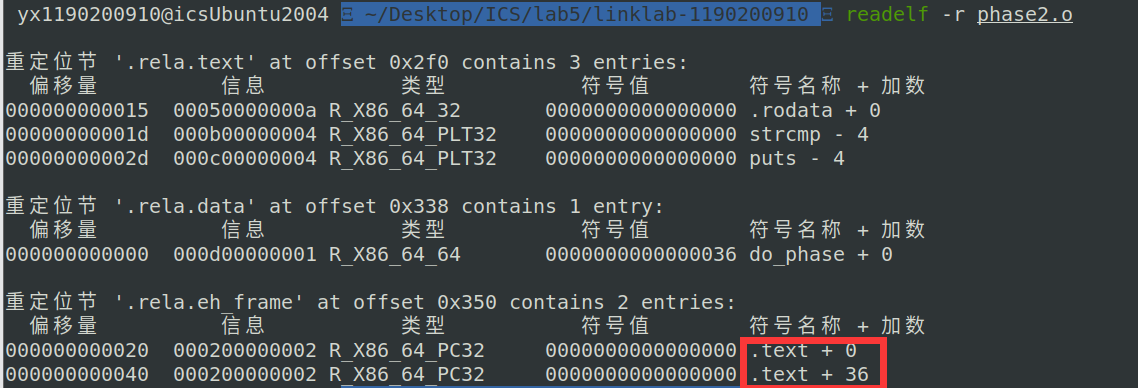
我们需要在do\_phase内部调用上面的zKlIjzNp函数从而完成学号的输出。因此我们需要编辑.text段的do\_phase代码。

用readelf -S看一下phase2.o的节头信息：



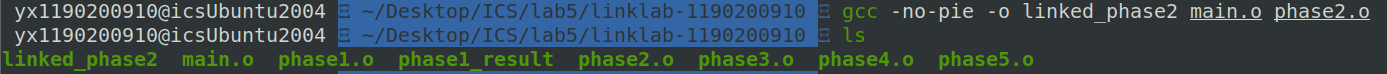
可以看到.text节的文件偏移是0x40字节，也就是说.text节从文件的第0x40个字节开始。

然后用readelf -r看一下phase2.o的重定位信息：

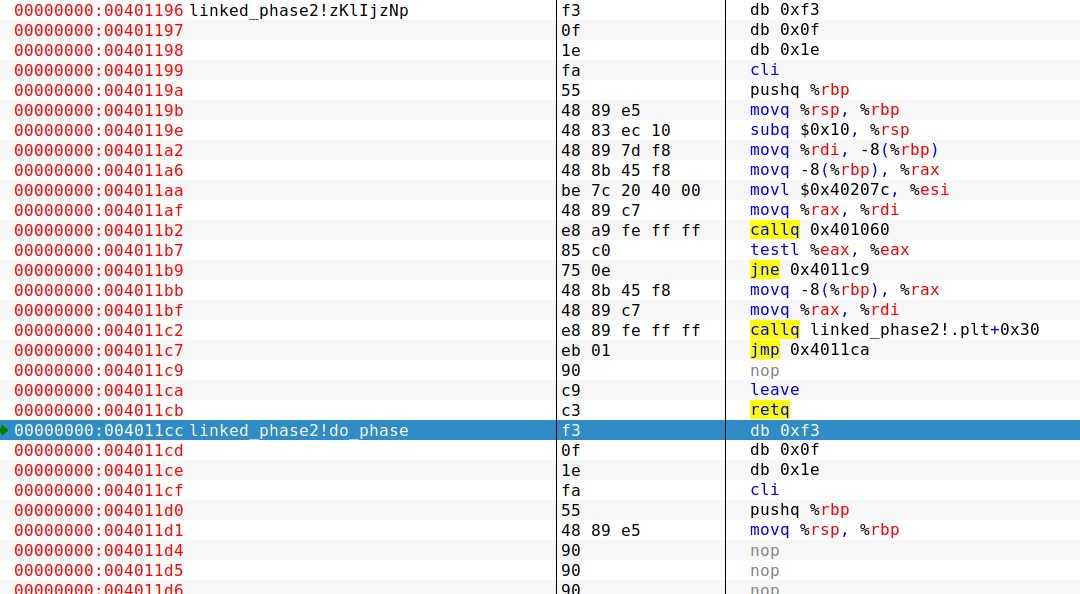


下方的.text+0和.text+36应该就是phase2.o中的两个函数，其中第一个函数的入口在.text节的偏移0的位置，第二个函数的入口在.text节偏移0x36的位置，结合刚刚objdump -d的结果，可以判断出do\_phase函数的入口就是位于.text节偏移0x36位置。

但我们还不知道如何编写在do\_phase函数中填入的机器代码，我们先把原始的phase2.o和main.o链接，然后用EDB调试，在EDB中编写汇编代码并编译成机器语言，我们复制对应的机器码即可。

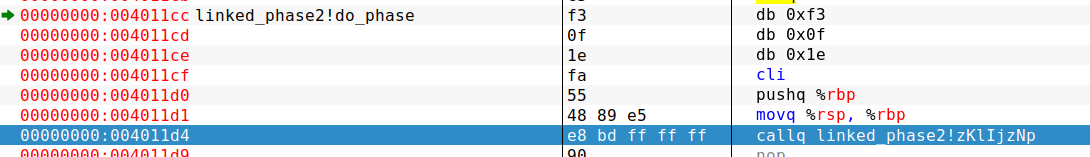


链接phase2.o，然后用EDB运行上一步的结果linked\_phase2。

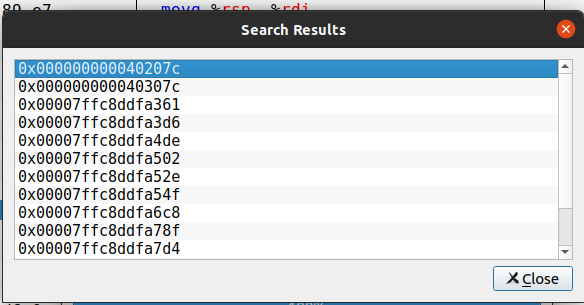


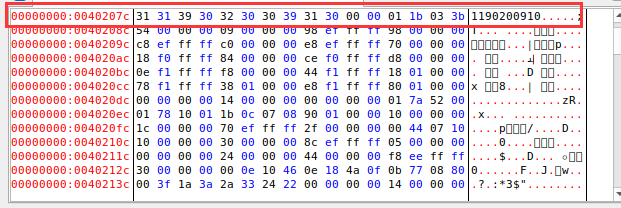
可以看到，do\_phase中有一系列的nop指令，我们需要在其中call zKlIjzNp函数。

利用EDB的assemble功能编写汇编代码并编译，得到这样的结果：



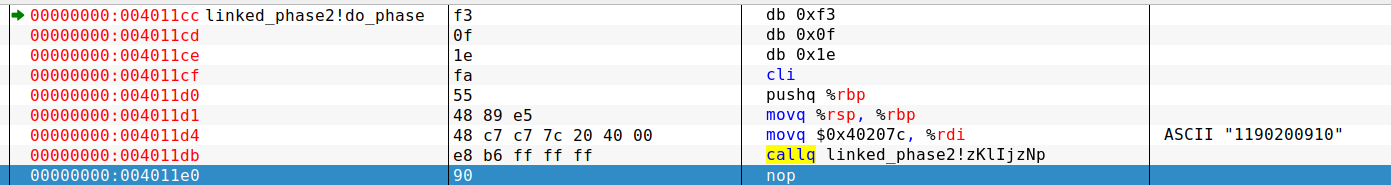
但是，zKlIjzNp需要接收一个参数，该参数是输出字符串的所在地址，因此在程序中需要找到自己学号字符串的地址。我们在EDB中直接Ctrl+F查找1190200910，可以查到以下结果。





说明程序中0x40207c的地址处就存放着我学号的字符串。为了验证这个地址是否会动态变化，多次运行EDB，发现这个地址不会改变，每次都是0x40207c。

因此重新编写do\_phase的汇编代码：



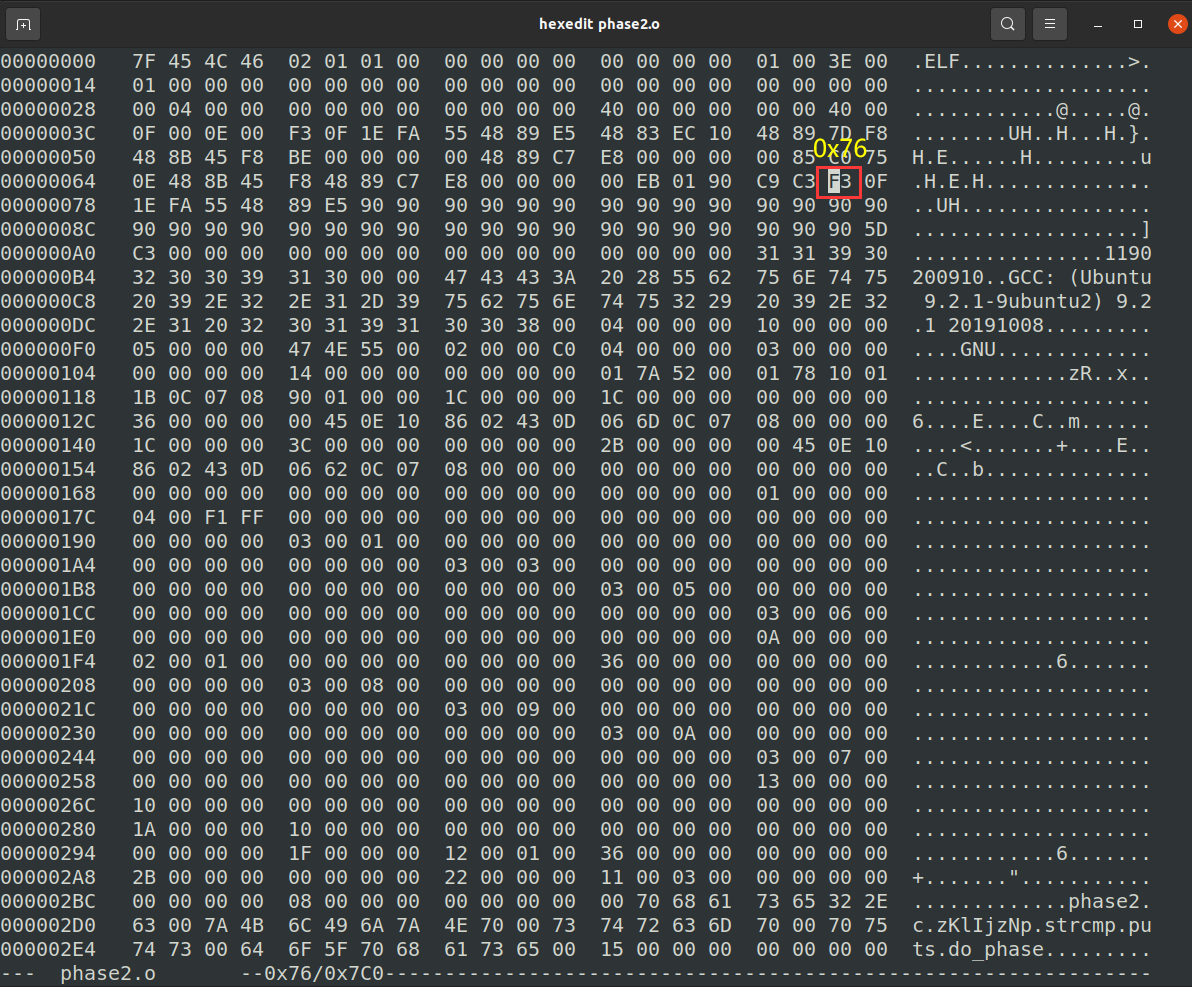
经过测试这段代码可以让do\_phase调用zKlIjzNp函数并输出学号。

因此我们需要在phase2.o这个ELF文件中找到.text段(文件偏移为0x40)，在.text段中找到do\_phase函数入口(段内偏移为0x36)，然后将do\_phase函数的nop指令(机器码90)改为下面的机器码：

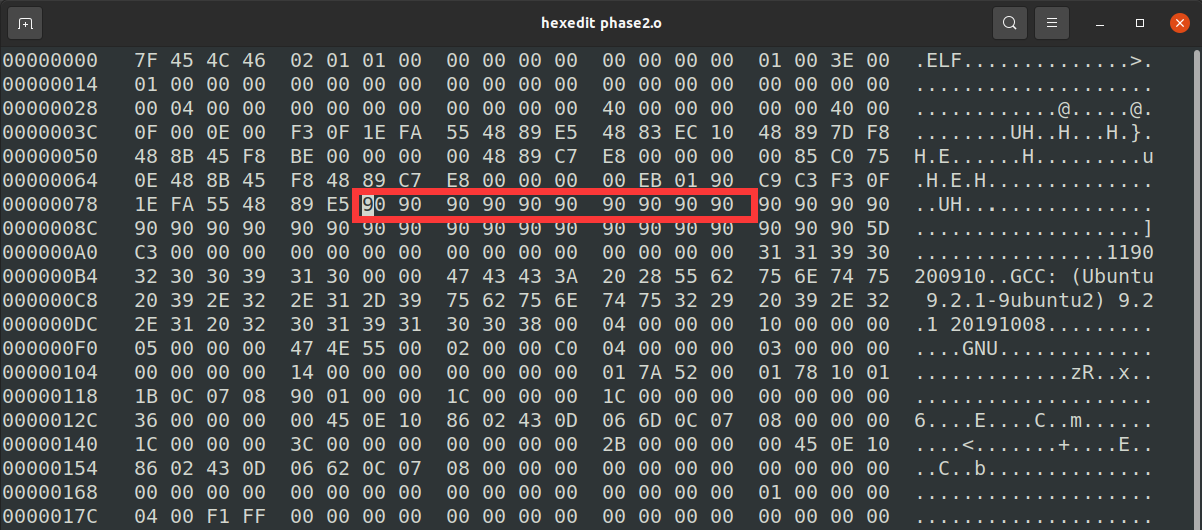
48 c7 c7 7c 20 40 00

e8 b6 ff ff ff

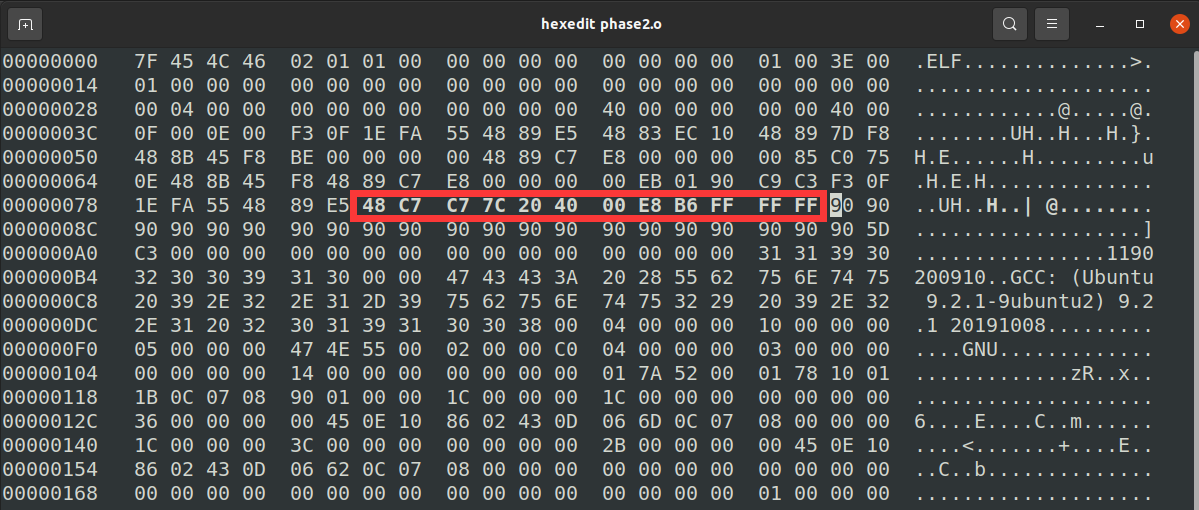
利用hexedit编辑phase2.o。经过计算，phase2.o中do\_phase函数入口与文件开头偏移量为0x40+0x36=0x76。



图中框起的部分即为文件偏移0x76处，也就是do\_phase函数的入口，这部分机器码“F3 0F 1E FA……”与在EDB调试中看到的机器码是一致的。我们需要将机器码“48 89 E5”后面的90(nop指令的机器码为90)改为上面那串机器码。



将框起来的那部分90机器码改为：48 c7 c7 7c 20 40 00 e8 b6 ff ff ff



修改后的结果如上图所示。按Ctrl+W保存，按Ctrl+X退出。

然后再次编译运行，可以看到程序正确输出了学号。

