

**课 程 实 验 报 告**

**课程名称： 计算机系统基础实验**

**实验名称： 二进制程序分析**

**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 8 日**

**一、实验目的与要求**

通过逆向分析一个二进制程序（称为“二进制炸弹”）的构成和运行逻辑，加深对理论课中关于程序的机器级表示各方面知识点的理解，增强反汇编、跟踪、分析、调试等能力。

实验环境：Ubuntu，GCC，GDB等。

**二、实验内容**

作为实验目标的二进制炸弹（binary bombs）可执行程序由多个“关”组成。每一个“关”（阶段）要求输入一个特定字符串，如果输入满足程序代码的要求，该阶段即通过，否则程序输出失败。实验的目标是设法得到得出解除尽可能多阶段的字符串。

为了完成二进制炸弹的拆除任务，需要通过反汇编和分析跟踪程序每一阶段的机器代码，从中定位和理解程序的主要执行逻辑，包括关键指令、控制结构和相关数据变量等等，进而推断拆除炸弹所需要的目标字符串。

实验源程序及相关文件：

bomb.c 主程序

phases.o 各个阶段的目标程序

support.c 完成辅助功能的目标程序

phases.h support.h 公共头文件

**阶段1： 串比较 phase\_1(char \*input);**

要求输出的字符串(input) 与程序中内置的某一特定字符串相同。提示：找到与input串相比较的特定串的地址，查看相应单元中的内容，从而确定input 应输入的串。

**阶段2：循环 phase\_2(char \*input);**

要求在一行上输入 6个整数数据，与程序自动产生的 6个数据进行比较，若一致，则过关。提示：将输入串input拆分成 6个数据由函数 read\_six\_numbers(input, numbers) 完成。之后是各个数据与自动产生的数据的比较，在比较中使用了循环语句。

**阶段3：条件分支 phase\_3(char \*input);**

要求输入一个整数数据，该数据与程序自动生成的 一个数据比较，相等则过关。提示：在自动生成数据时，使用了 switch … case 语句。

**阶段4：递归调用和栈 phase\_4(char \*input);**

要求在一行中输入两个数，第一个数表示在一个有序的数组（或者binary search tree）中需要搜索到的数，该数是在一定范围之内的；第二个数表示找到搜索数的路径（在树的左边搜索编码为二进制位0，在树的右边搜索编码为二进制位1）。

**阶段5：指针和数组访问 phase\_5(char \*input);**

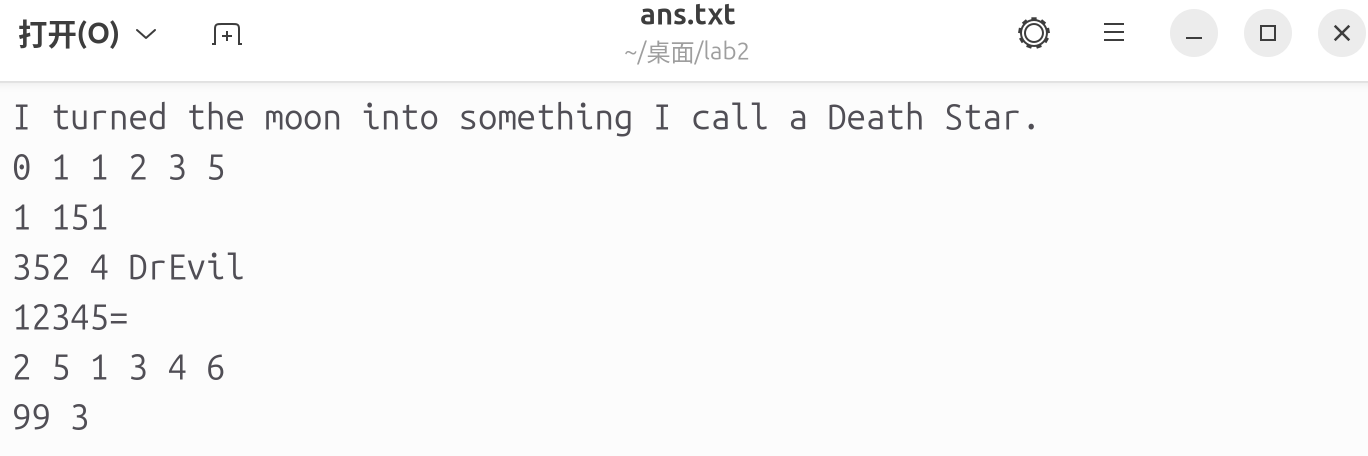
要求在一行中输入一个串，该串与程序自动生成的串相同。在生成串和比较串时，使用了数组和指针。

**阶段6：链表、结构、指针的访问 phase\_6(char \*input);**

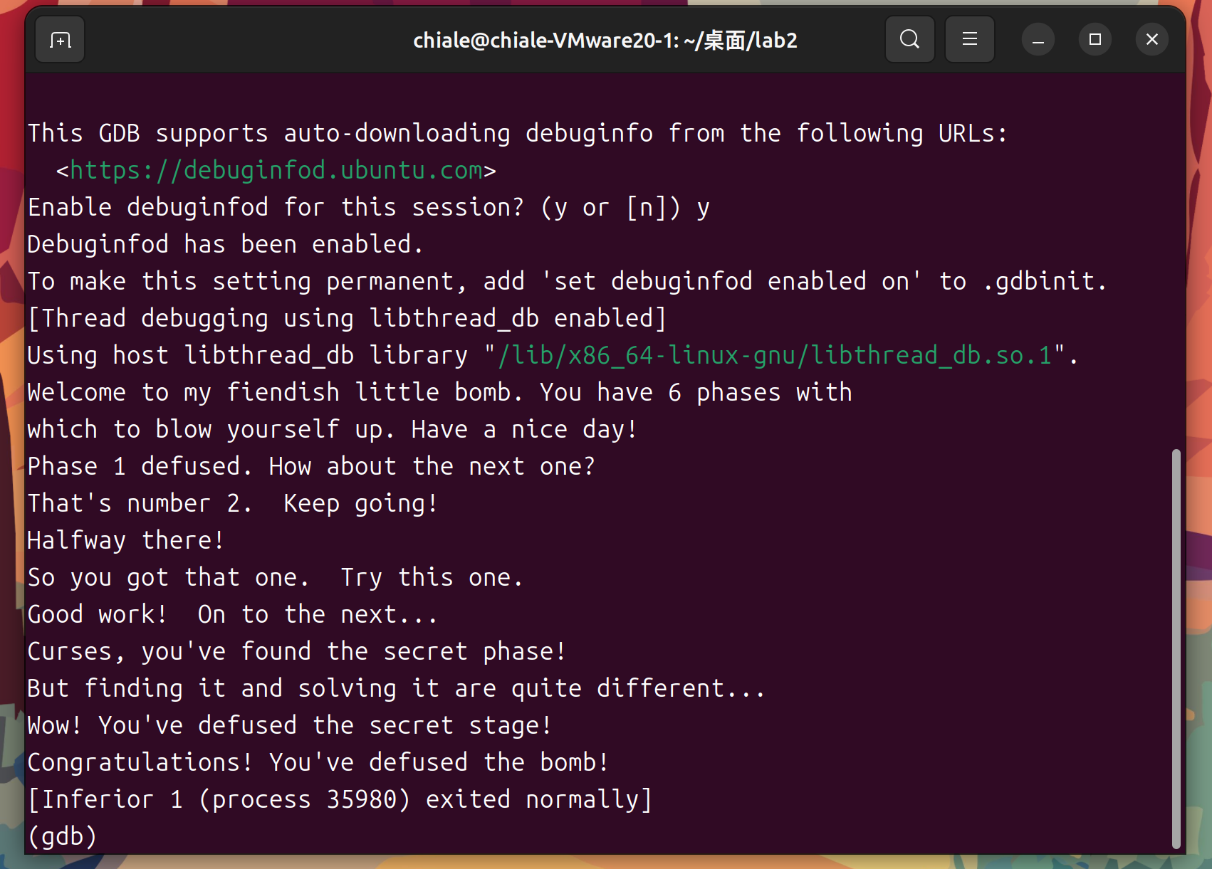
要求在一行中输入6个数，这6个数是一个链表中结点的顺序号（从 1 到6）。按照输入的顺序号，将对应链表结点中的值形成一个数组。若该数组是按照降序排列的，则过关。

1. **实验记录及问题回答**
2. **实验结果**

采用文件输入的方式，结果保存在ans.txt中

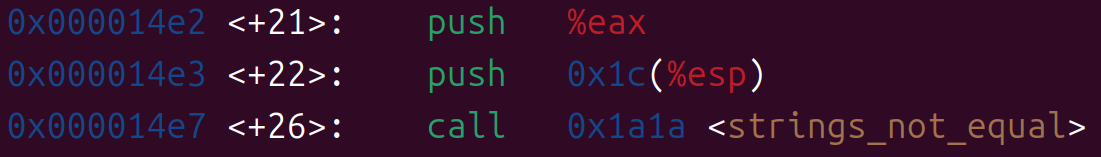
****

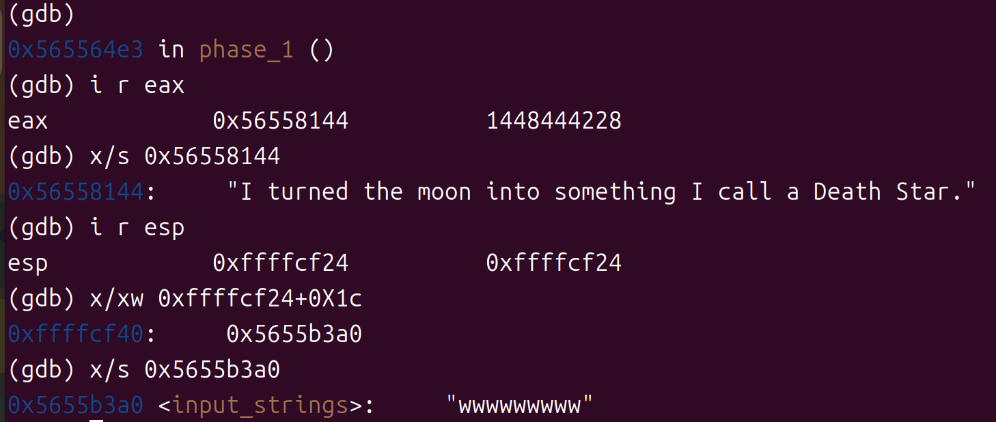
运行结果如下图



1. **实验过程记录**
2. **phase\_1：**

输入wwwwwwwww测试，比较两个字符串是否相等

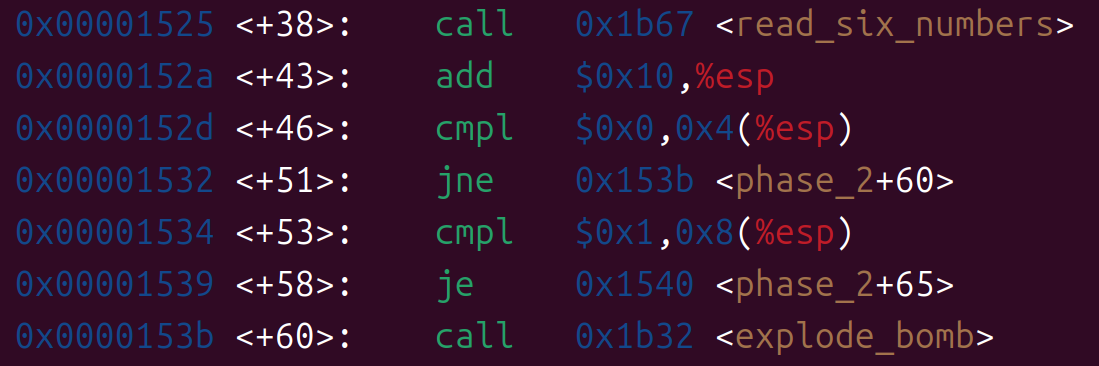
****

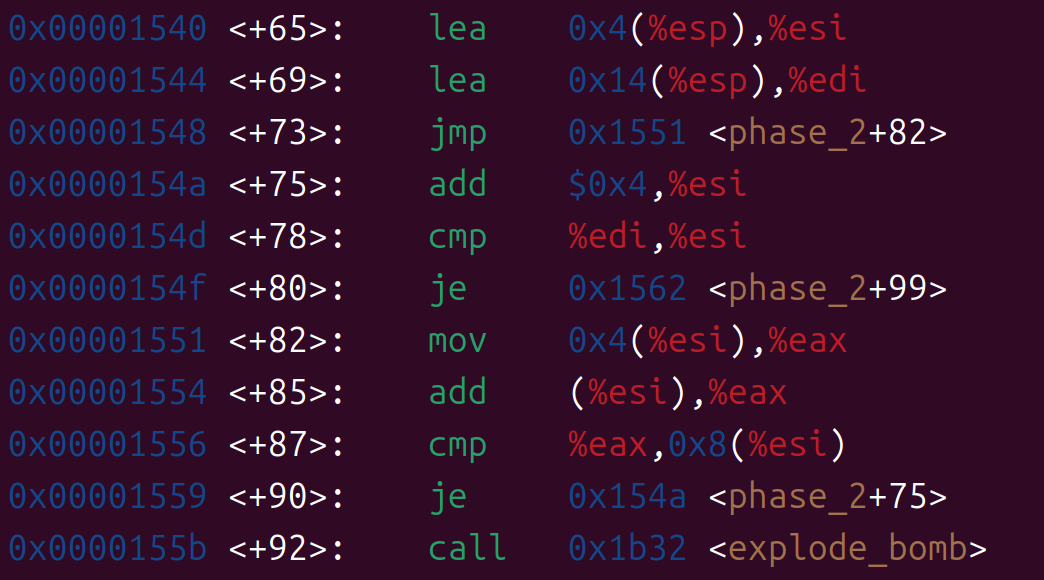
两个字符串的地址分别存在%eax和%esp+0x1c中

“I turned the moon into something I call a Death Star.”

1. **phase\_2：**

读入6个整数，前两个数为0和1



接下来每个数都等于前面的两个数之和

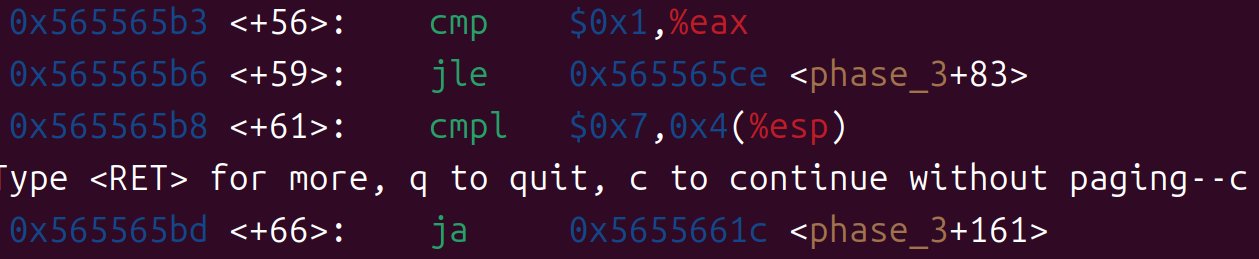
所以最终结果为：0 1 1 2 3 5

1. **phase\_3：**

这是一个switch-case语句，需要输入两个整数



<+56>检查是否输入两个数，<+61>检查第一个数是否小于等于7

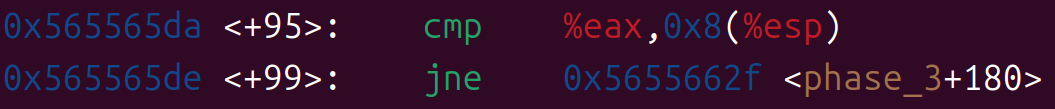


根据第一个数，计算%edx并跳转，给%eax赋值





%eax再与第二个数比较，不同则爆炸

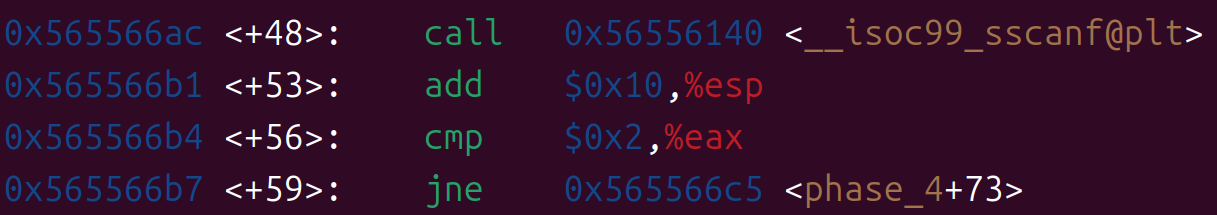


最终结果可以为

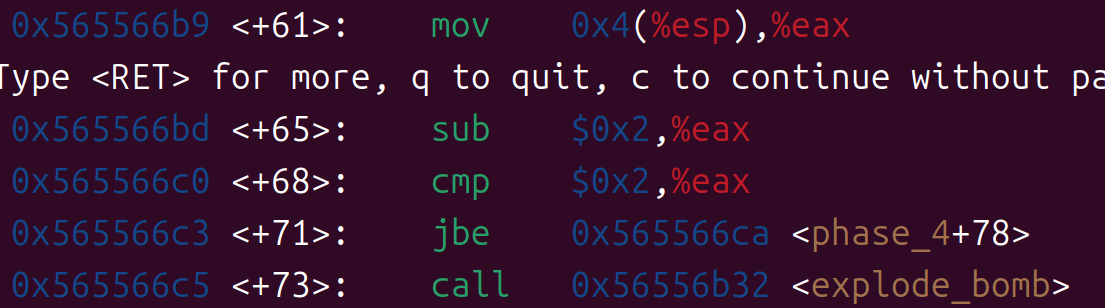
|  |  |  |
| --- | --- | --- |
| 第一个数 | 第二个数 | %edx |
| 1 | 151 | 0x565565d5 |
| 2 | 478 | 0x565565f2 |
| 3 | 53 | 0x565565f9 |
| 4 | 683 | 0x56556600 |
| 5 | 231 | 0x56556607 |
| 6 | 266 | 0x5655660e |
| 7 | 129 | 0x56556615 |

1. **phase\_4：**

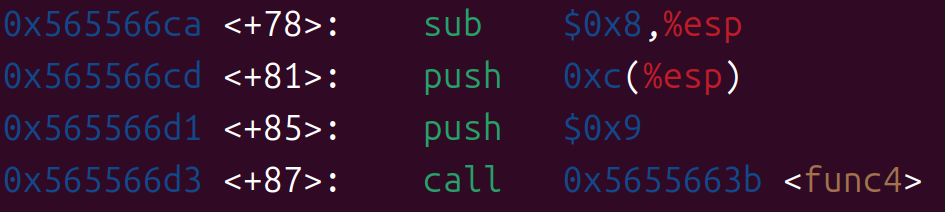
输入两个数x、y，x地址为：%esp+0x8，y地址为：%esp+0x4



检查2<=y<=4

****

y和9压栈，需要计算func4(9,y)

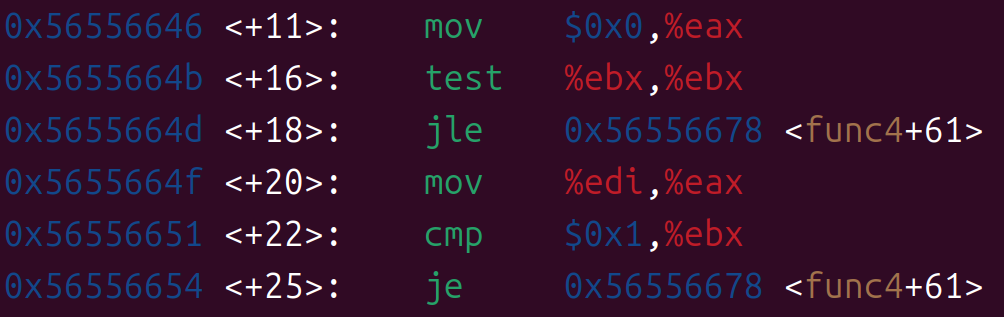


以下步骤进入到func4函数

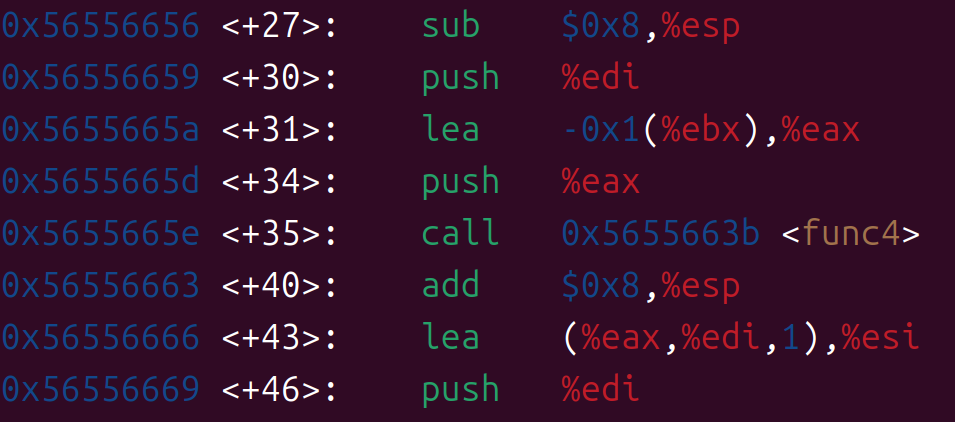
%ebx为传入的第一个参数，%edi始终为y

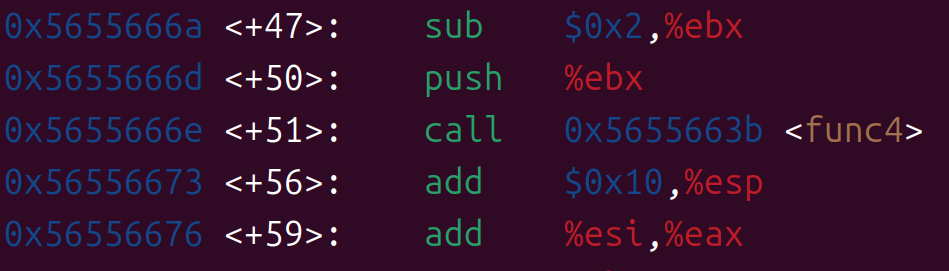


函数返回值为%eax，**f(0,y)=0**，**f(1,y)=y**

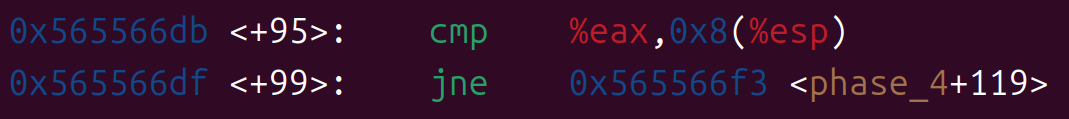


递归计算**f(n,y)=f(n-1,y)+f(n-2,y)+y**

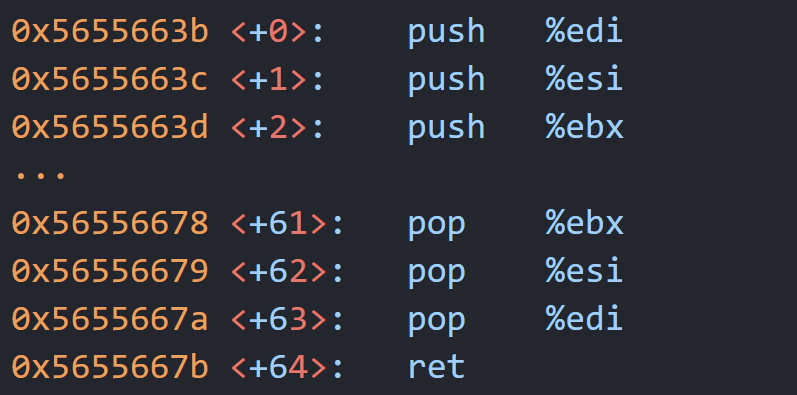




返回到phase\_4，判断x是否等于f(9,y)



每次调用func4不改变%edi，%esi，%ebx的值

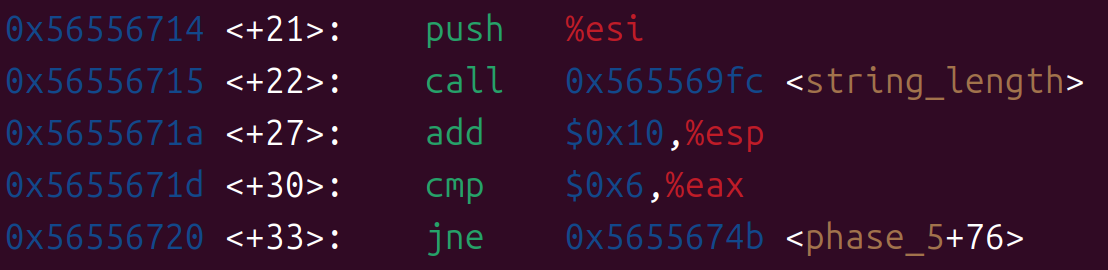


最终结果可以为

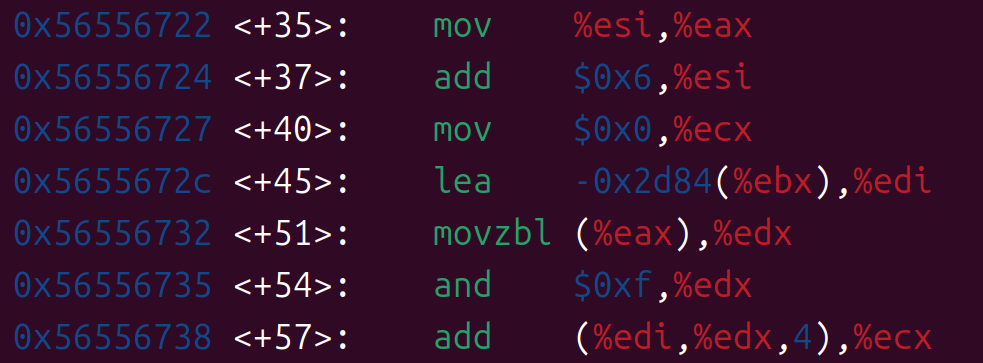
|  |  |
| --- | --- |
| 第二个数y | 第一个数x=func4(9,y) |
| 2 | func4(9,2)=176 |
| 3 | func4(9,3)=264 |
| 4 | func4(9,4)=352 |

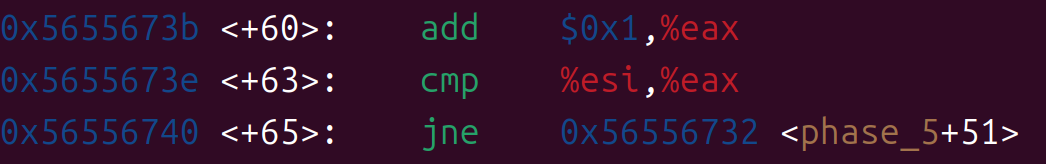
1. **phase\_5：**

输入字符串长度为6，%esi为字符串首址,用“abcdef”测试

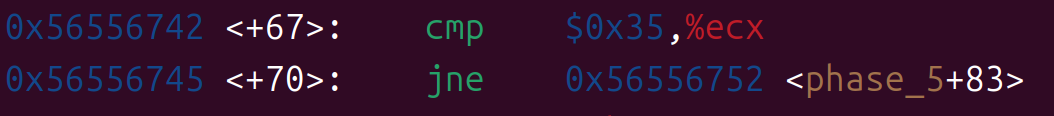


每个字符的ASCII码取低四位再乘4作为以%edi为首址的int

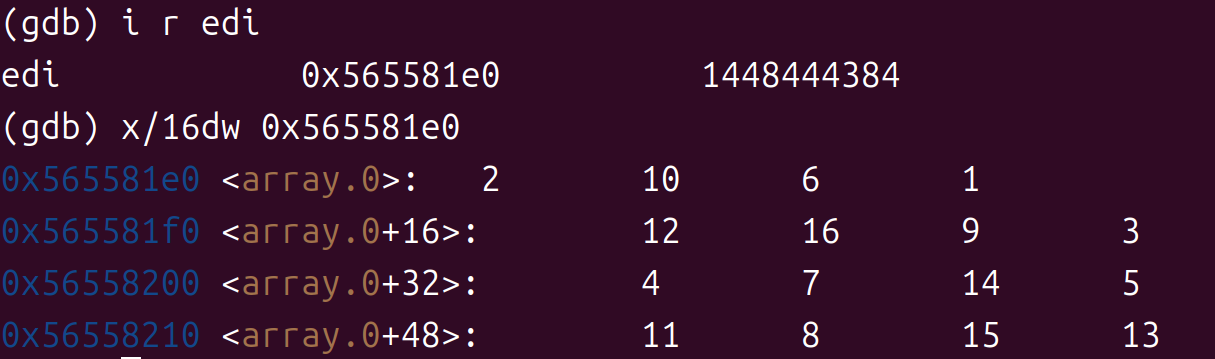
型数组的偏移量，并将该偏移地址中的值加到%ecx中



最终%ecx的值等于53(0x35)



查看以%edi为首址的int型数组信息



可以发现：10+6+1+12+16+8=53

因此偏移地址分别为：0x4 0x8 0x12 0x16 0x20 0x34

对应的字符低四位为：0x1 0x2 0x3 0x4 0x5 0xd

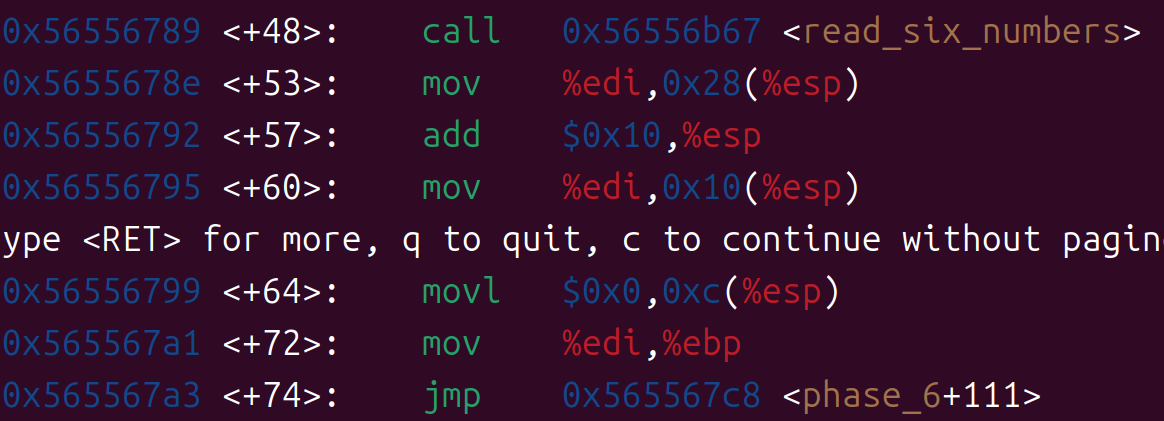
对应字符ASCII码可以为：0x31 0x32 0x33 0x34 0x35 0x3d

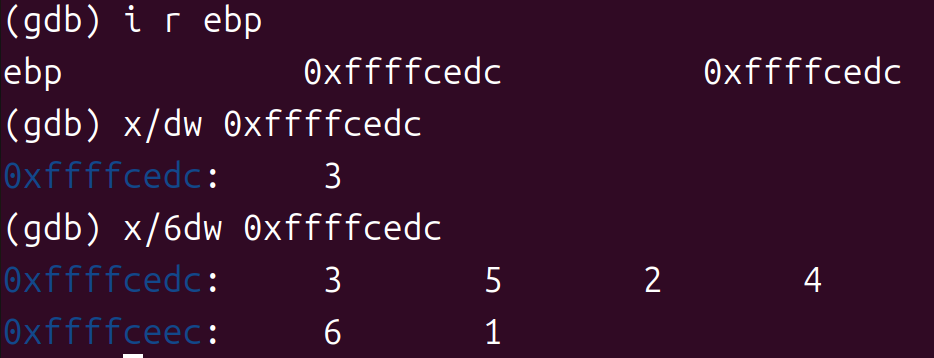
对应字符串为：12345=

1. **phase\_6：**

需要读入6个整数，输入3 5 2 4 6 1测试，6个数放在一个数

组中，数组首地址存放在%edi和%ebp，0xc(%esp)为当前索引值0

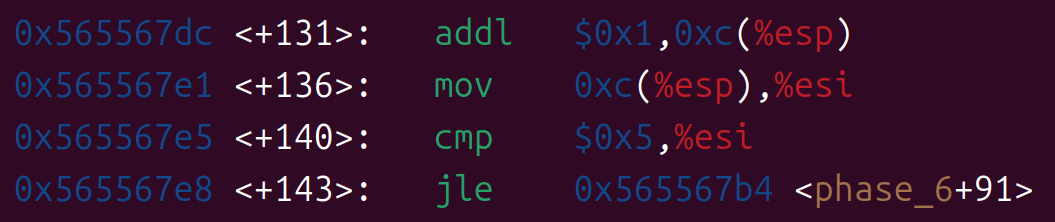




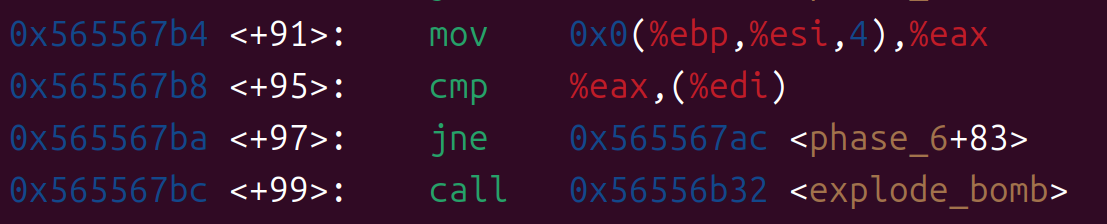
0x10(%esp)为数组首地址，0x14(%esp)为第一个数，第一个数<=6



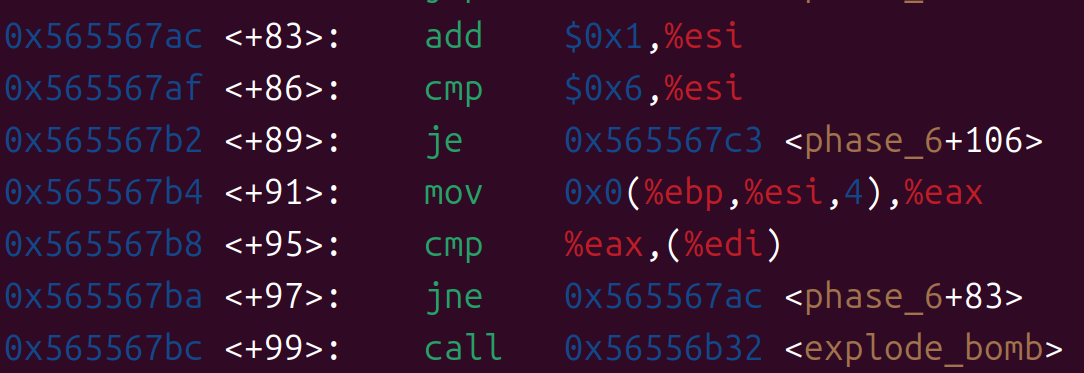
索引加1(为1)，用%esi(当前为1)保存索引，%esi<=5则跳转



a[1]!=a[0],否则爆炸



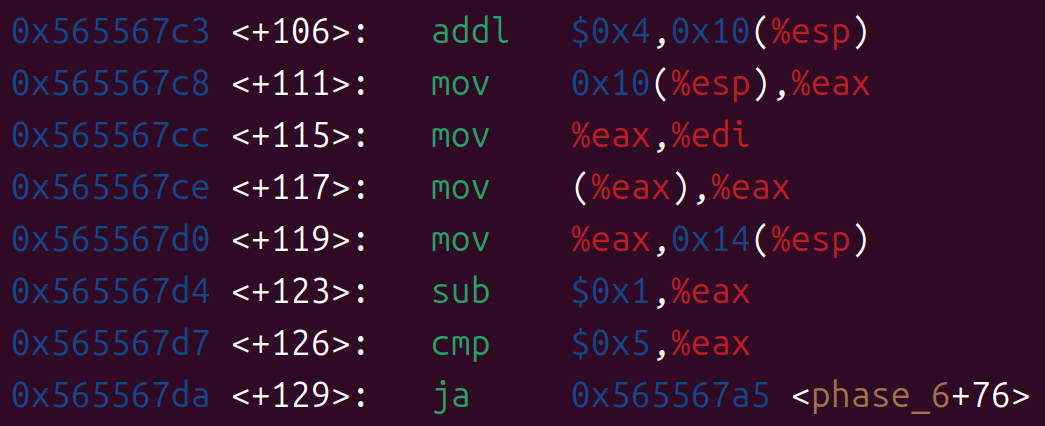
%esi(1)加1，%esi=6则跳转，a[0]以后的所有数都不等于a[0]



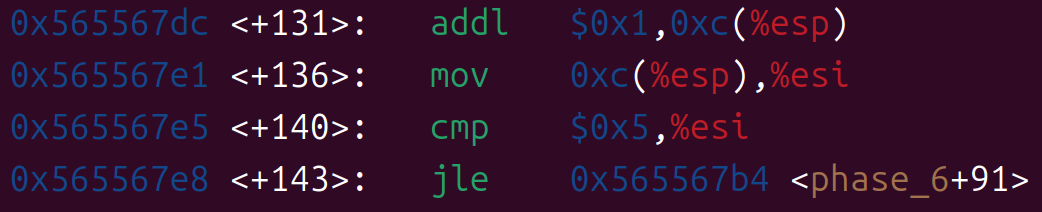
跳转至106行时，%esi=6，0x10(%esp)存放首地址；执行完106

行，0x10(%esp)存放第二个数地址，%edi也更新；0x14(%esp)

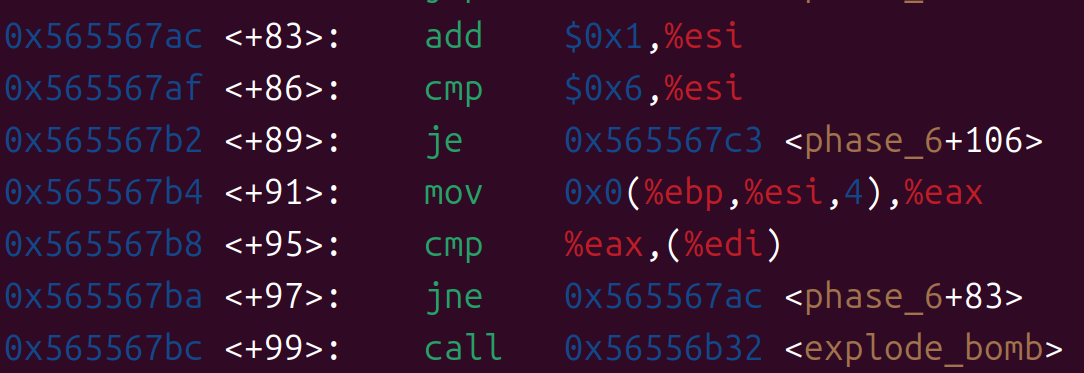
存放第二个数，第二个数<=6



索引加1(为2)，用%esi(当前为2)保存索引，%esi<=5则跳转



%esi(2)加1，%esi=6则跳转，a[1]以后的所有数都不等于a[1]

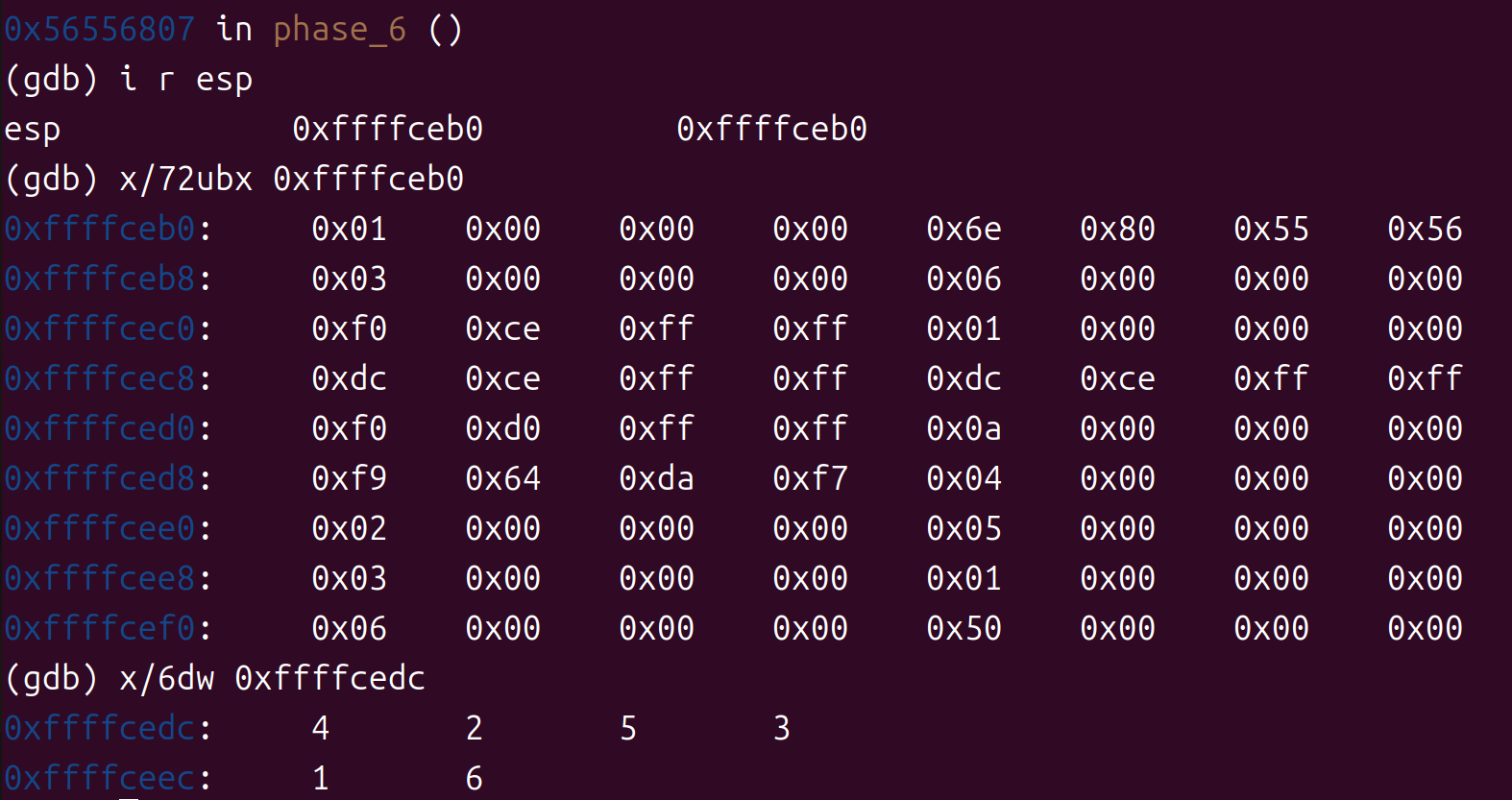


继续循环下去，6个数互不相等且均<=6，循环完后运行至145行

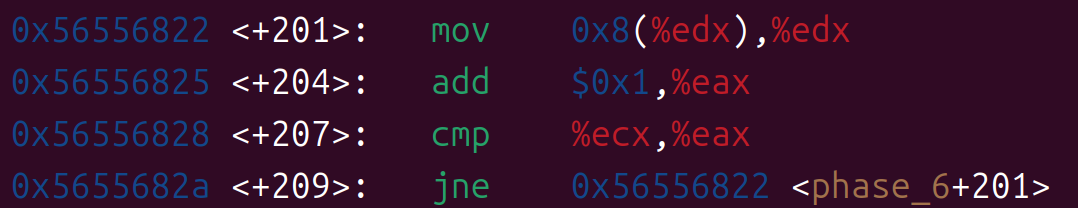
0x18(%esp)和0x1c(%esp)为数组首地址，a[n]=7-a[n]

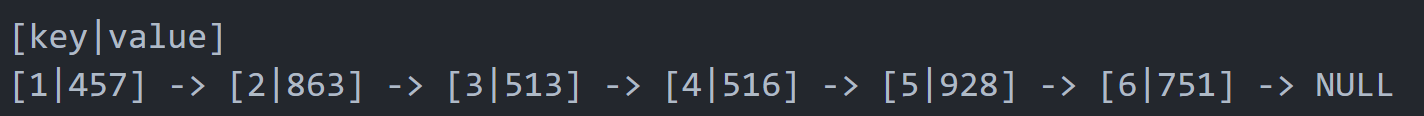
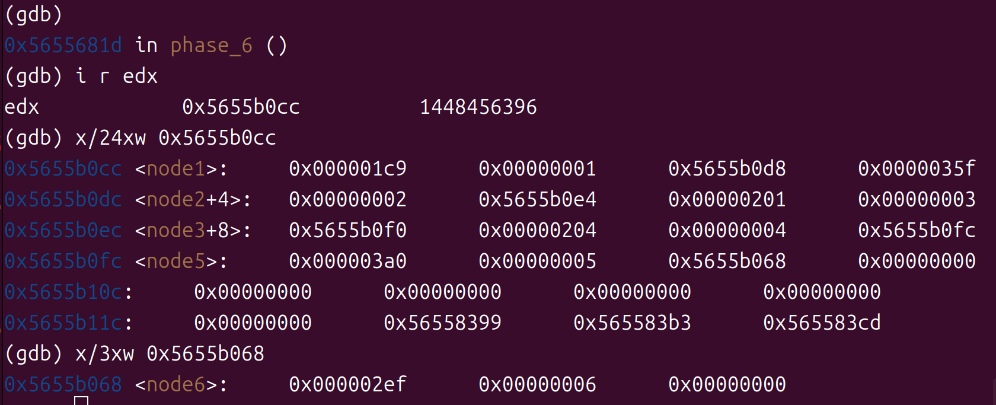
%esp+0x2c即为数组首地址



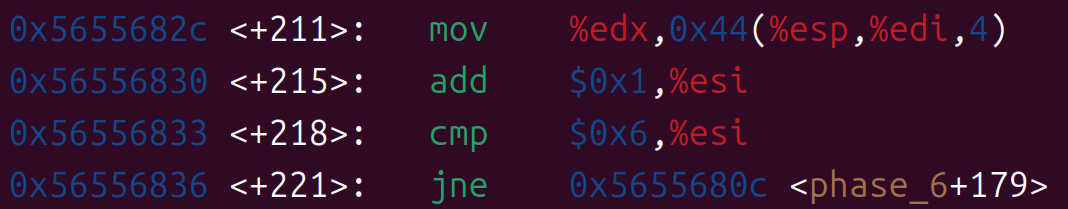


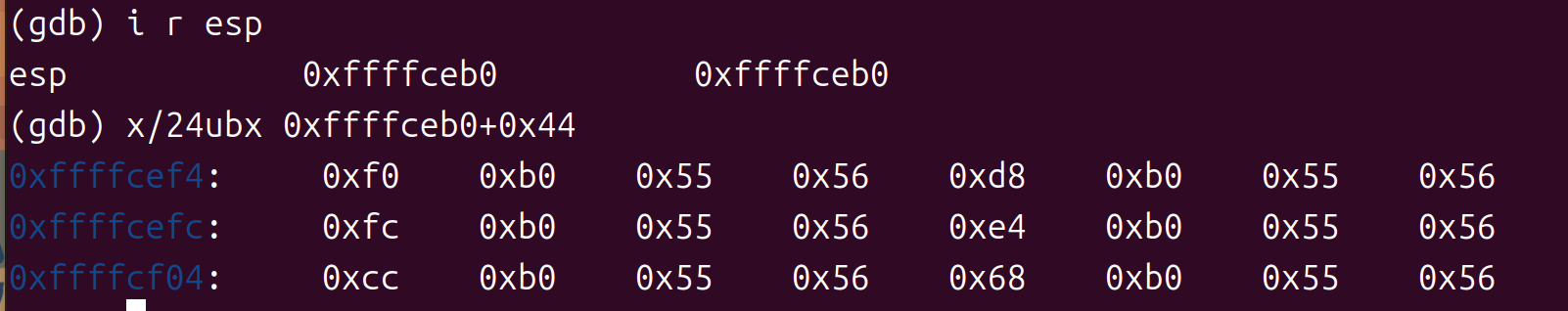
初始时，%edx为链表首地址，1个结点占12个字节



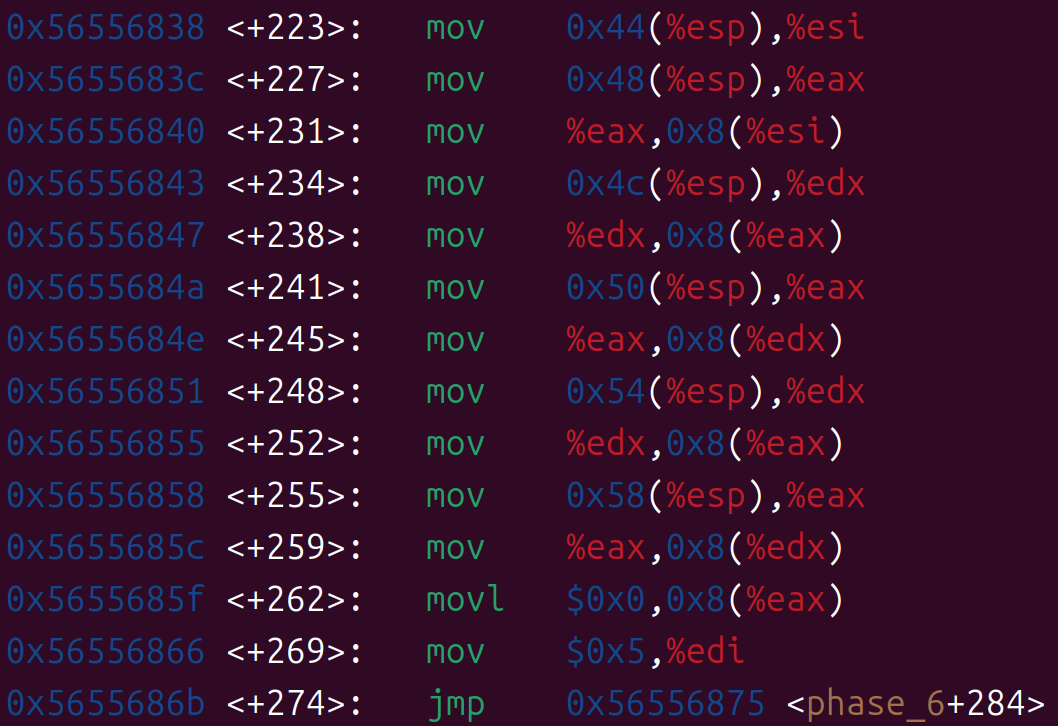


%esp+0x44处依次存放链表第a[n]个结点的地址(n:0~5)

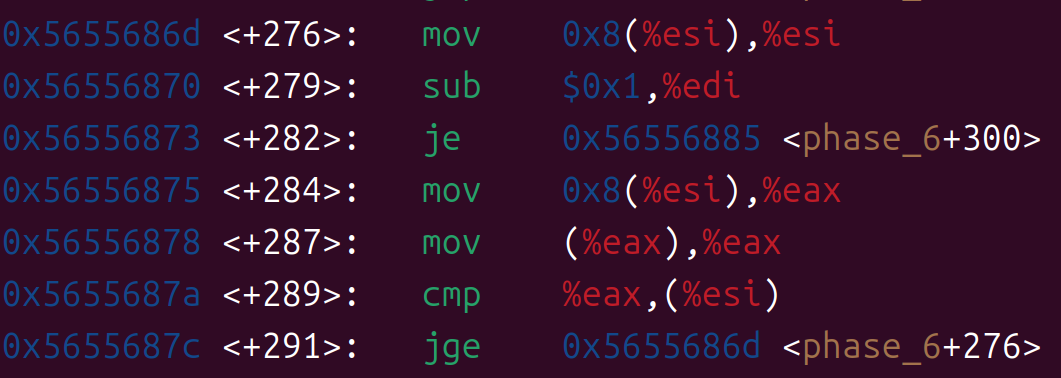




重置链表(改变指针的连接)，其key值顺序与a[n]一致



%esi为链表重置后的首地址，重置后的链表应降序排序



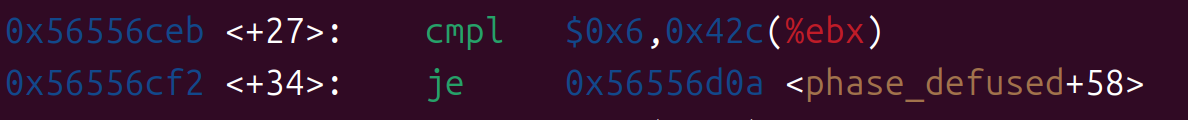
所以链表key值顺序：5 2 6 4 3 1

a[n]：5 2 6 4 3 1

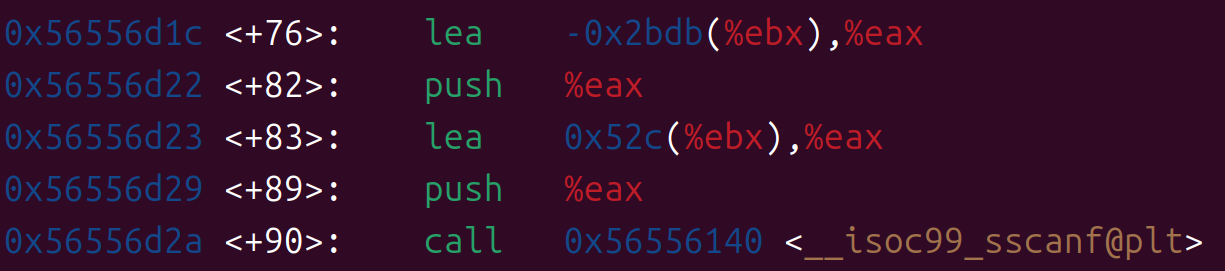
7-a[n]：2 5 1 3 4 6 (最终结果)

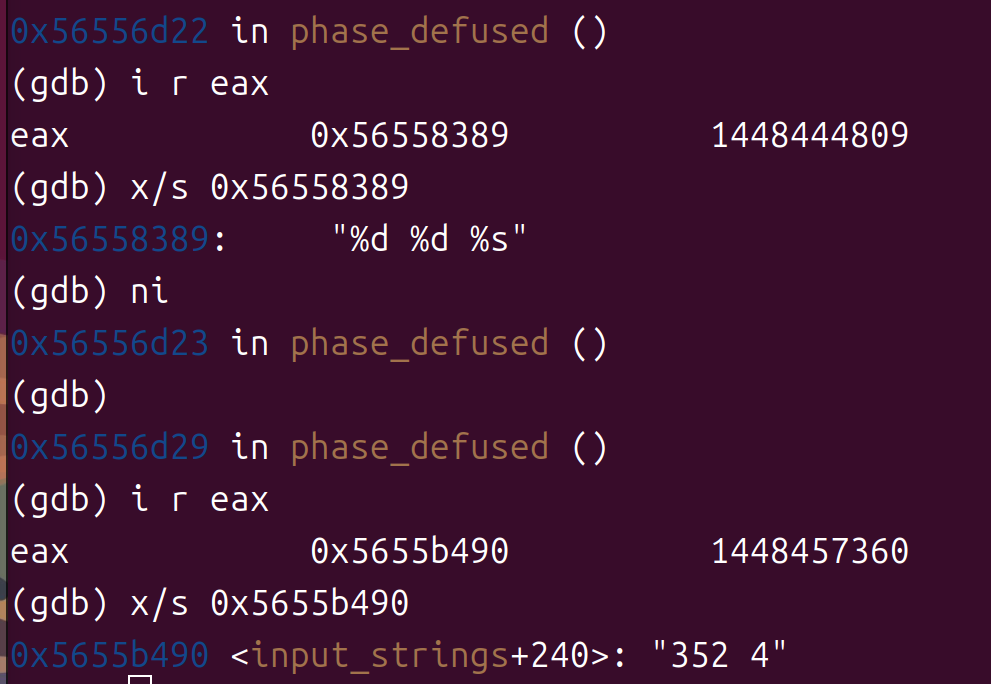
1. **secret\_phase：**

每次通过一关后调用phase\_defused()函数，检查是否通过六关

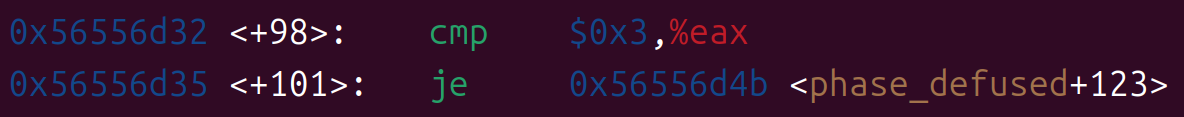


需要在第四关添加特定字符串



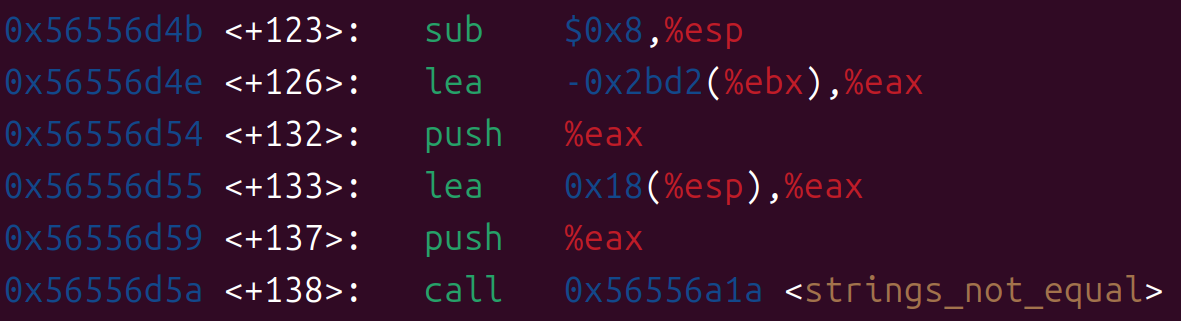


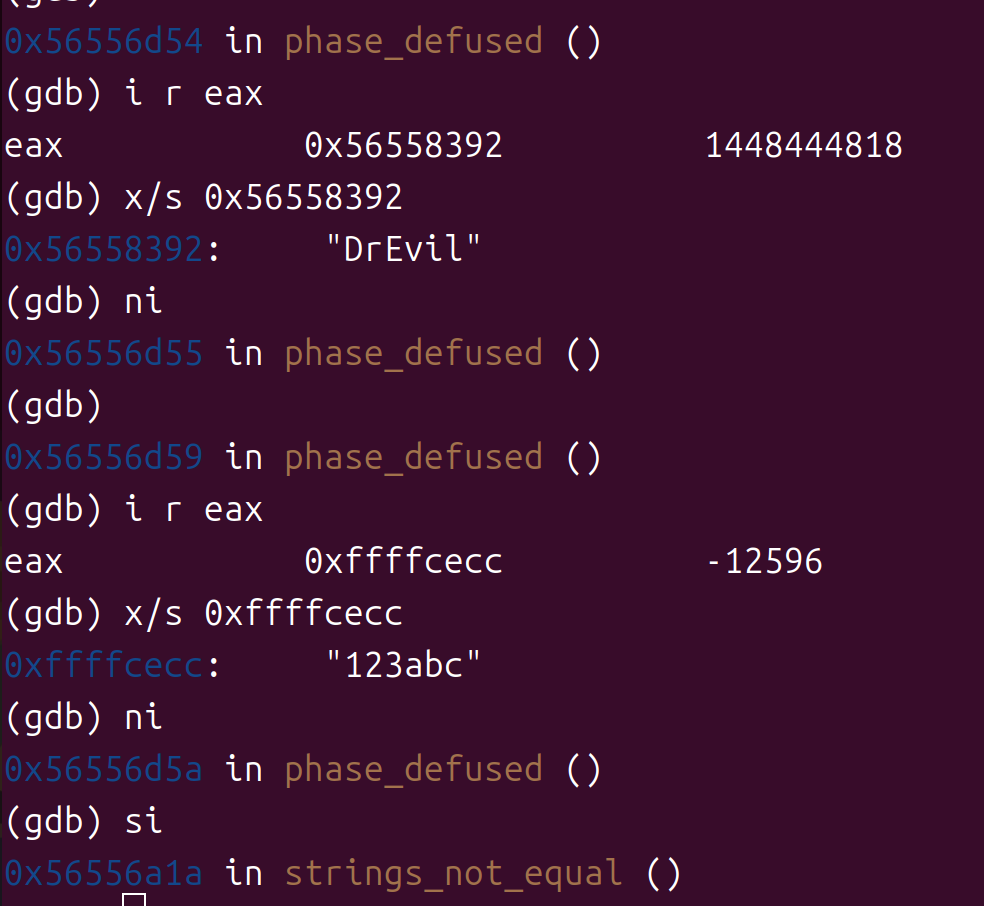
%eax检查第四关是否输入了字符串，所以先输入123abc测试



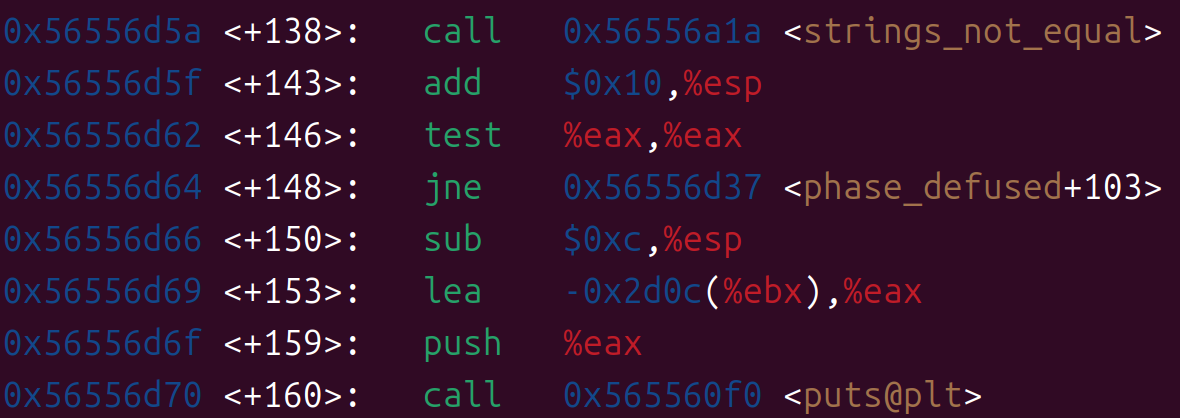
调用strings\_not\_equal前依次将比较的两个字符串地址压栈

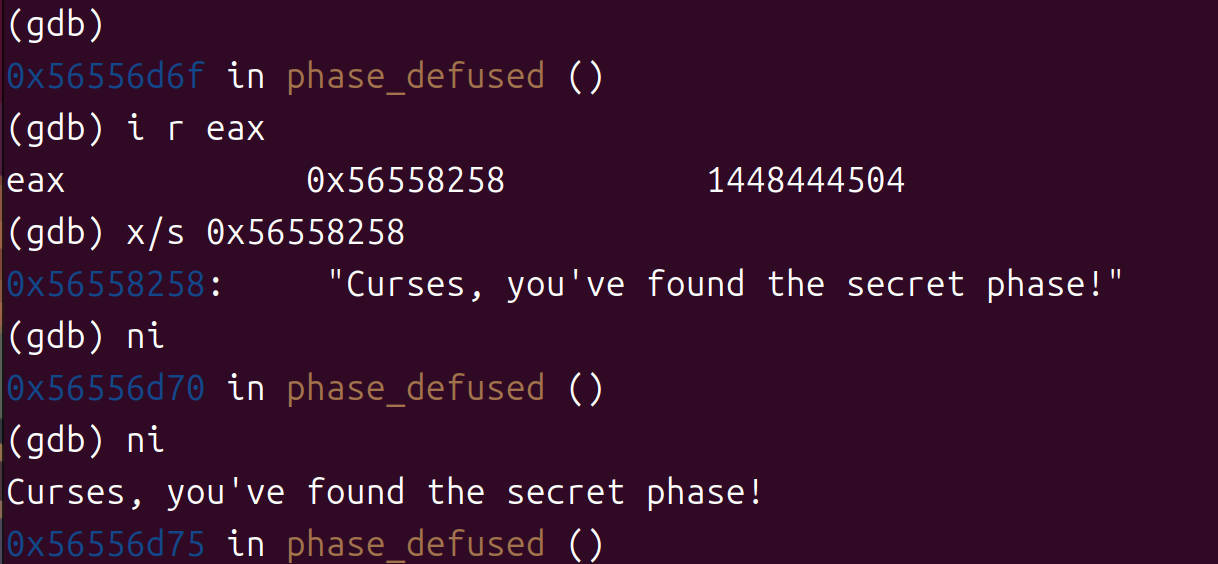
可知需要在第四关添加"DrEvil"



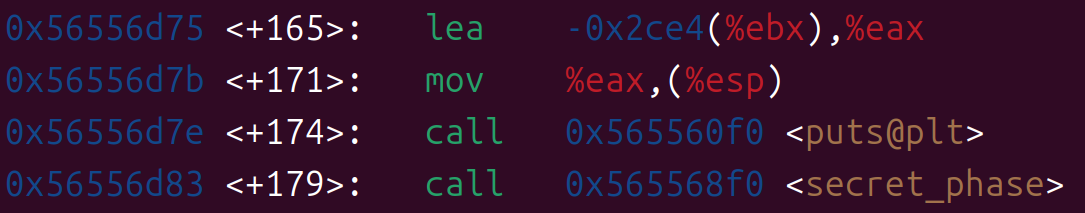


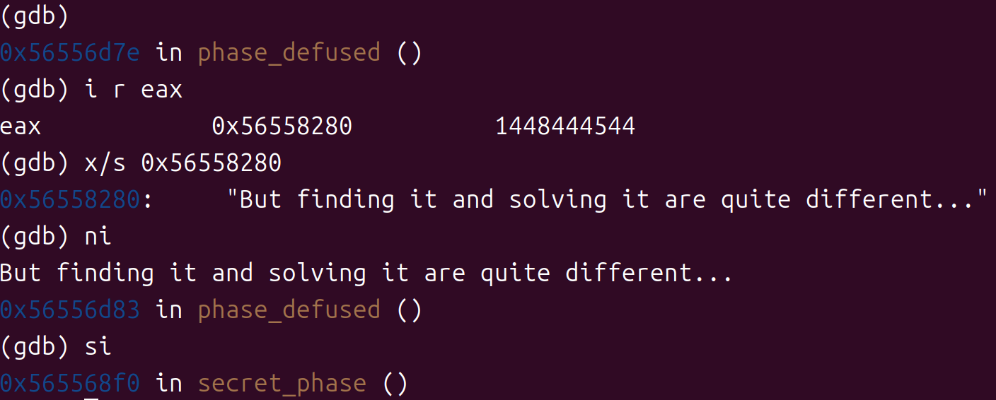
添加"DrEvil"后



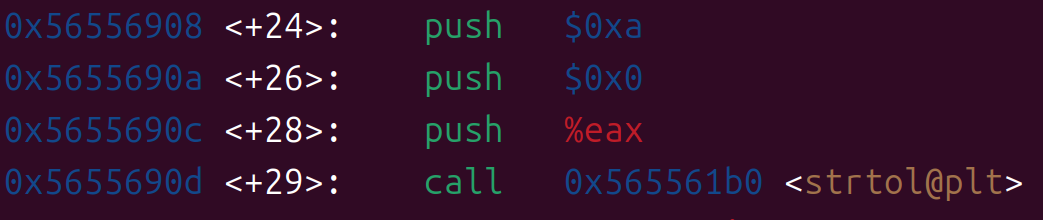


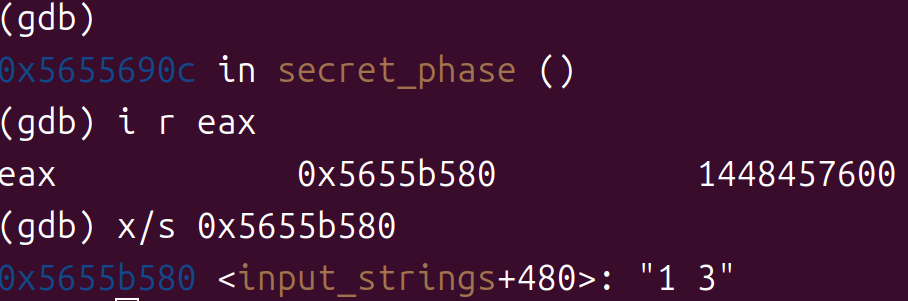
接下来进入secret\_phase



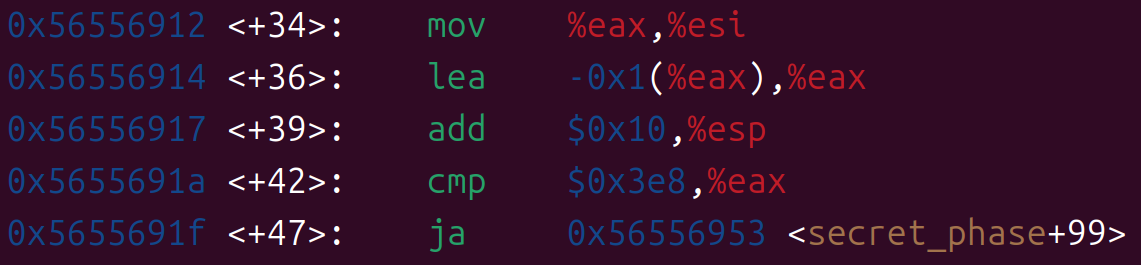


输入1 3进行测试

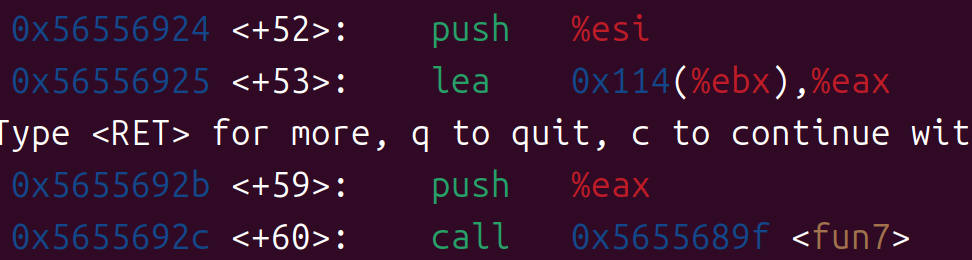


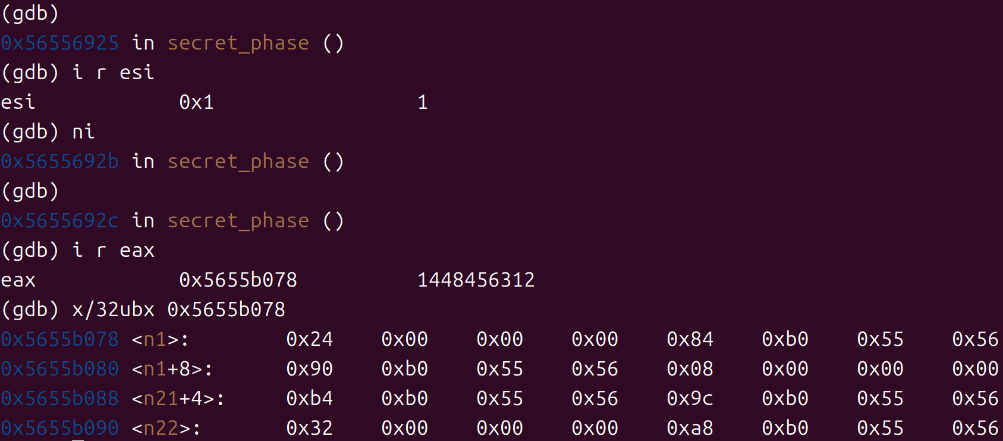


第一个数<=1001



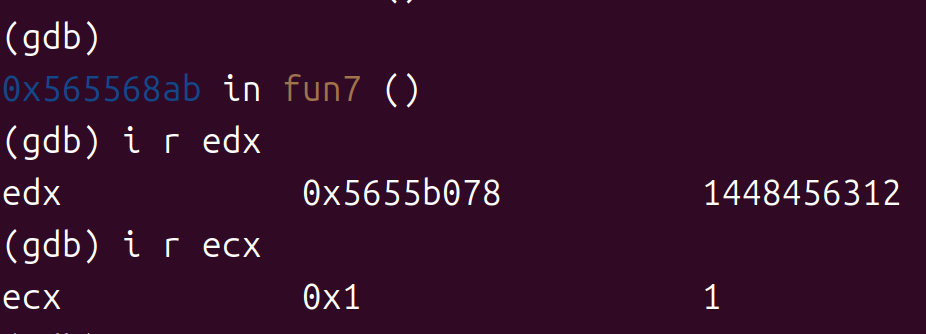
传入fun7函数的是第一个数和二叉树的根结点的地址

****

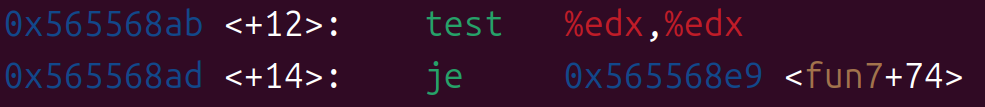


进入fun7函数，%edx为根结点地址，%ecx为第一个数



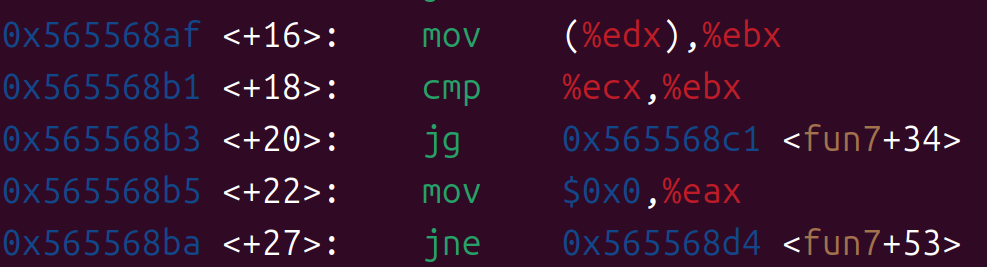


%edx为0表示没找到，返回-1

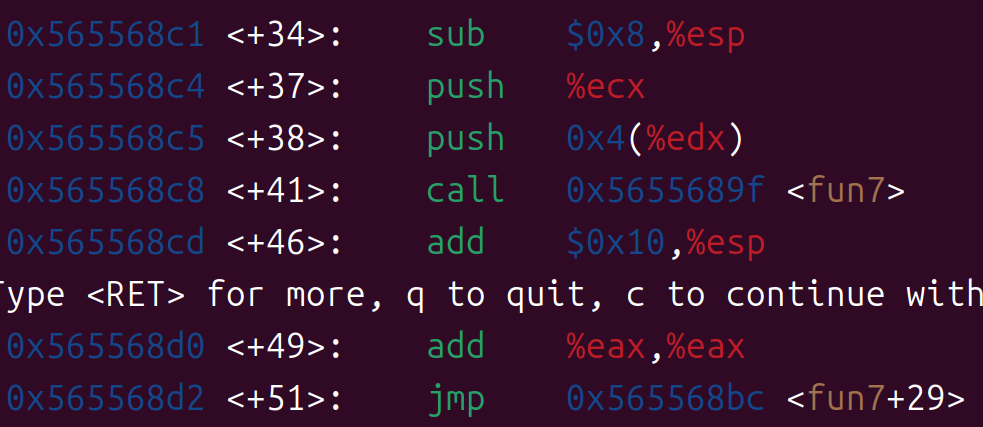


第一个数小于结点值则进入左子树查找，大于则进入右子树查找

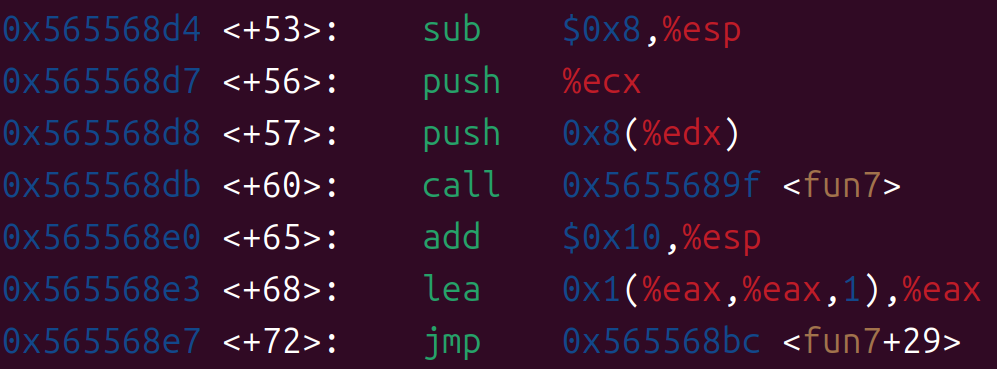
查找到或进入右子树，%eax赋为0

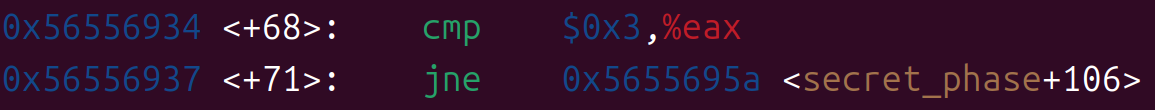


查找完左子树，%eax=2\*%eax

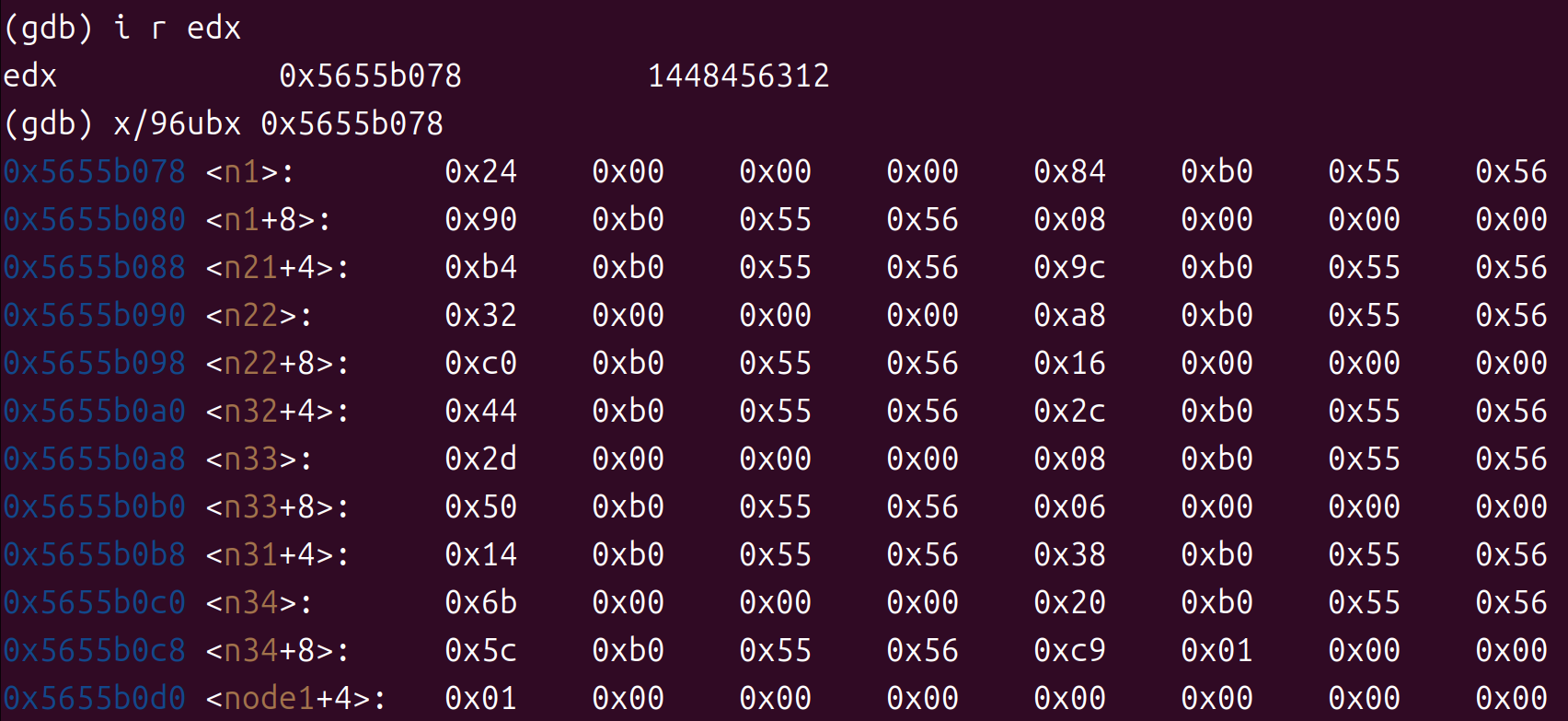
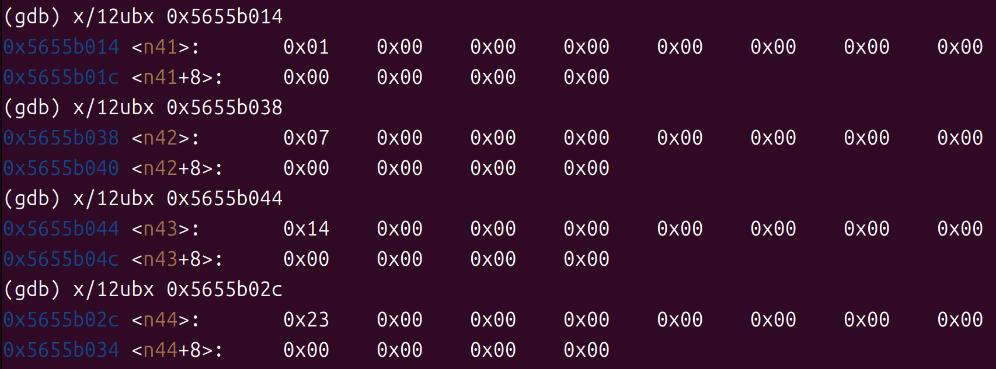
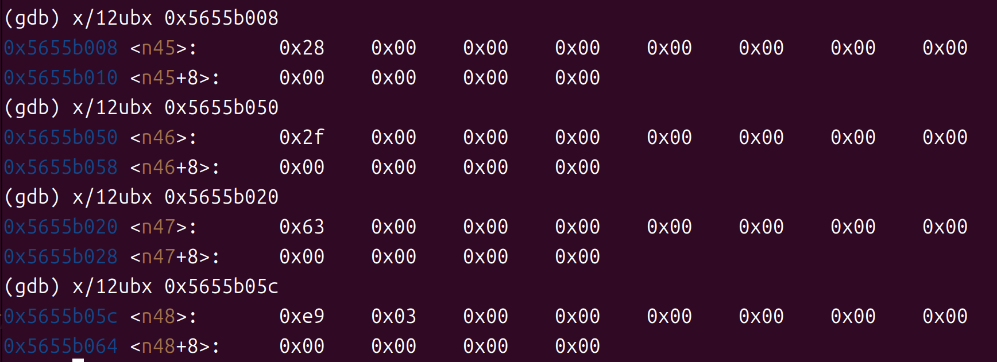


查找完右子树，%eax=%eax\*2+1



返回到secret\_phase后，%eax的值需为3

二叉树仅有4行

所以该结点为第3行第4个107,或第4行第7个99

**四、体会**

1. 加深对汇编语言和调试工具的理解

二进制炸弹实验要求深入理解汇编语言和程序的执行过程。在这个过程中，我通过阅读汇编代码、分析控制流、理解函数调用以及处理数据的方式，极大地加深了对计算机底层原理的认识。此外，实验中大量使用调试工具（gdb）来逐步检查代码执行过程，也让我更熟练地掌握了gdb的使用。

2. 培养耐心与细致的习惯

每个阶段的“炸弹”都需要细致的分析和耐心的调试。如果稍有不慎，输入错误的数据就可能导致炸弹“爆炸”，从而增加挑战。在实验过程中，我学会了如何有条不紊地分析问题、记录调试过程，并从错误中吸取经验，逐渐形成了细致入微的习惯。

3. 增强了解决问题的能力

实验中不同的炸弹阶段设计了复杂的挑战，涉及到字符串处理、算术运算、条件判断等多种编程技巧。这不仅让我对如何处理程序中的复杂逻辑有了新的认识，也让我提升了分析和解决问题的能力。当面对一个陌生的汇编代码文件时，我学会了从全局分析着手，逐步拆解问题，最终找到正确的解法。