

**课 程 实 验 报 告**

**课程名称： 计算机系统基础**

**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 ： 大数据202302班**

**学 号 ： U202315557**

**姓 名 ： 李润怿**

**指导教师 ： 许向阳**

**2024 年 10 月 20 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.rar

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

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，在树的有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。

**三、实验记录及问题回答**

**（1）实验任务1 的实验记录**

对于该二进制拆弹实验，一共分为六个炸弹，下面将分别进行简述

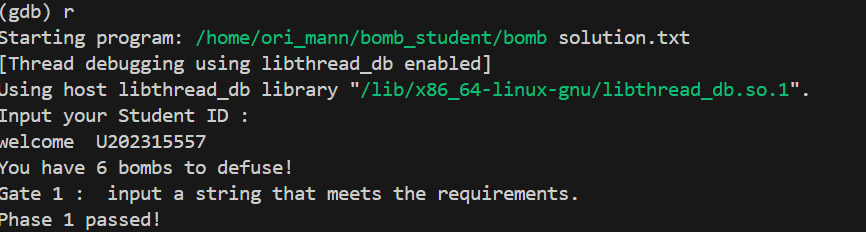
**第一个**

首先先观看汇编语言会发现，该炸弹与学号的最后一位有关，然后经过一系列变化来确定炸弹选择的答案，最后在函数 strings\_not\_equal中判断输入的字符串与答案是否相同，相同则跳过炸弹爆炸的部分，不相同则引爆爆炸，程序结束运行。



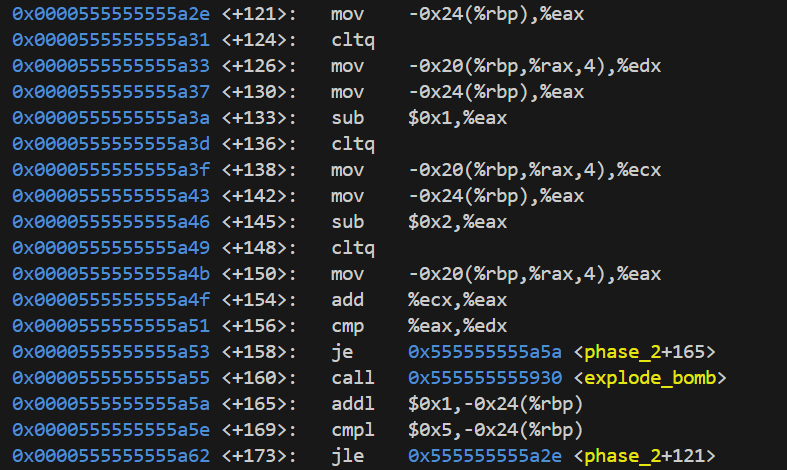
因此如果想要拆弹，我们需要查看在寄存器rdi和rax中的字符串，一个是输入的字符串，另一个存储了炸弹的答案，查看之后即可拆弹成功，答案为

Instructions set architecture



**第二个**

观察汇编语言会发现，在其中存在循环操作，即在<121>至<173>语句中，如下

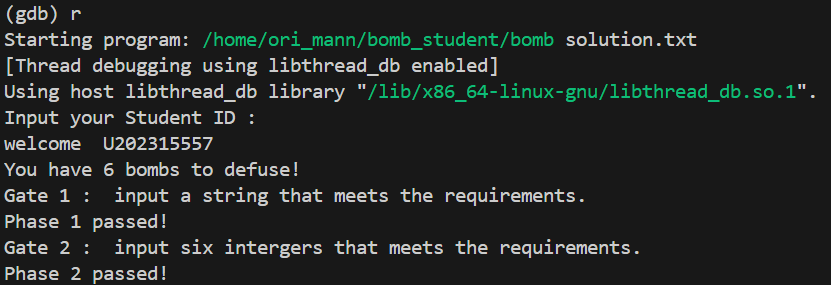


这个炸弹需要我们输入六个数字，前两个数字会以倒序的方式读取学号的后两位，我的序号后两位分别是7和5，之后的操作通过循环一次会发现为两者相加，因此我们可以破译出该炸弹的循环操作为如下公式

a3=a1+a2

即第三个数为前两个数字相加，因此破译答案为

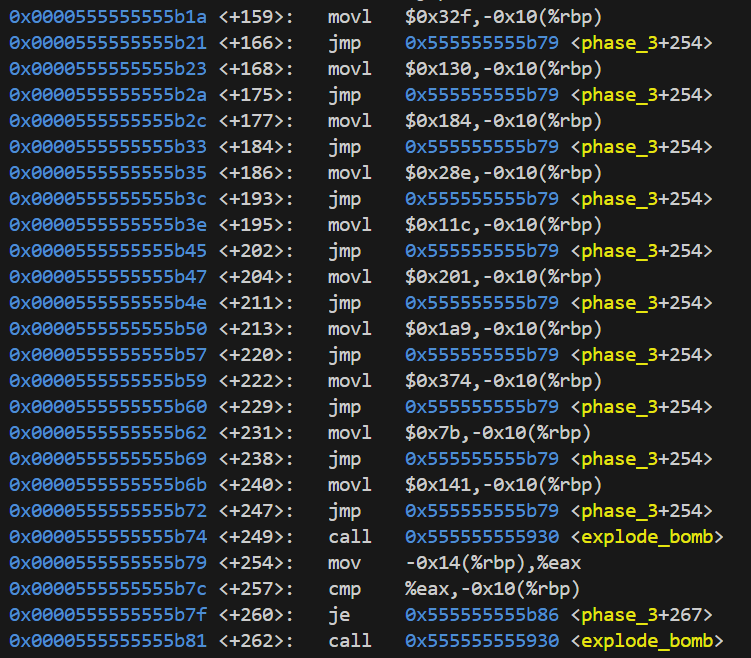
7 5 12 17 29 46



**第三个**

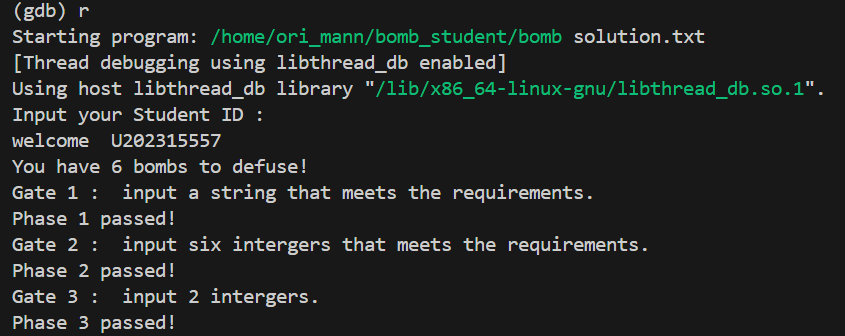
分析会发现第三个炸弹考察的知识点是条件分支判断，我们需要输入两个数字，第一个数字很容易发现为学号倒数第三位

而通过第一个数字按照条件分支确定第二个数字，汇编语言块具体如下

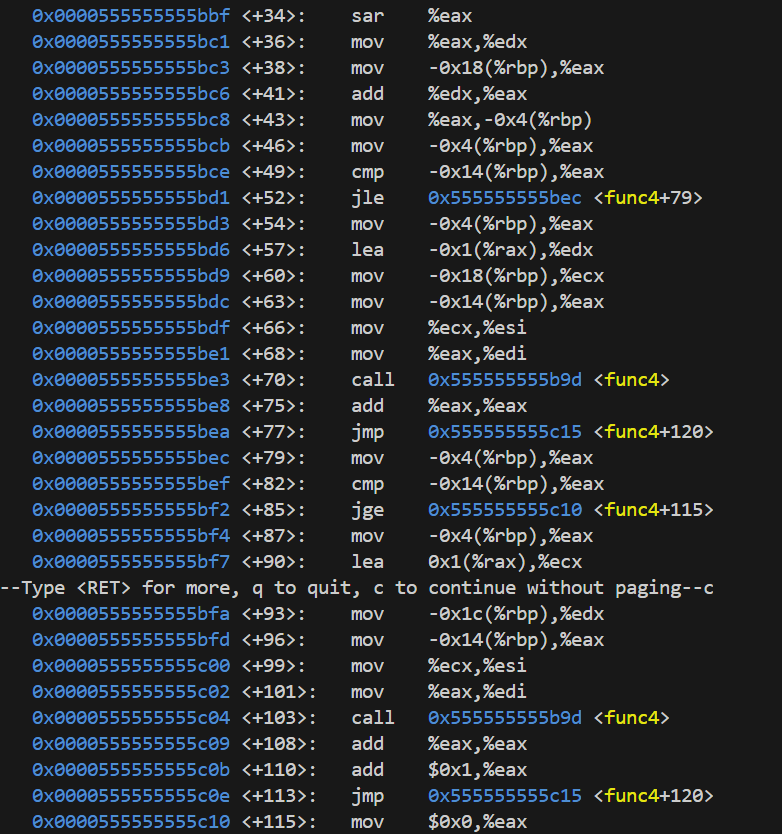


我们需要找到在炸弹爆炸之前比较的两个数字，因此我们寻找寄存器的值，发现答案拆弹成功，如下：

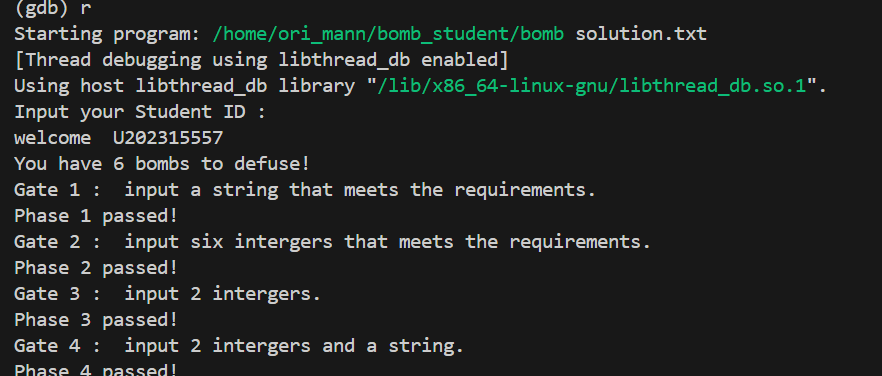
513



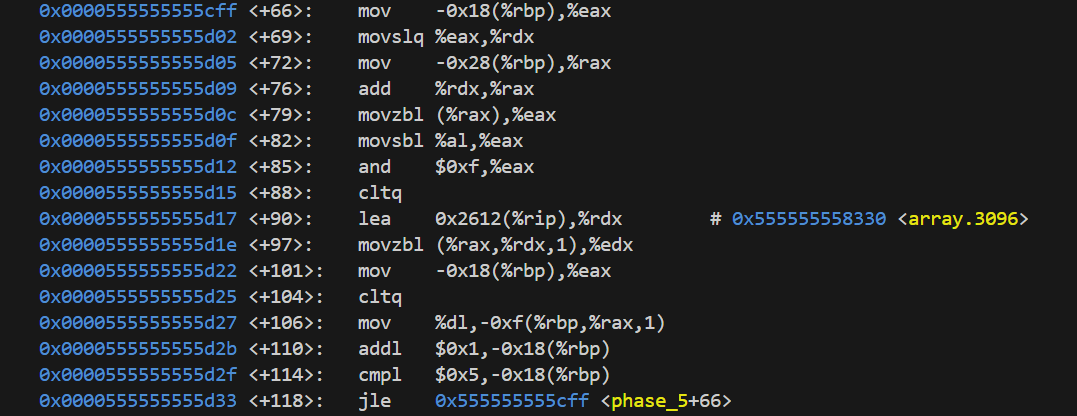
**第四个**

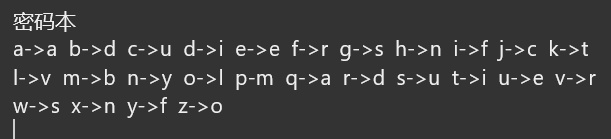
这个需要我们输入两个数字，而具体进行的操作集成到了函数func4中，通过打开func4的汇编语言，发现为递归函数

通过在炸弹函数phase\_4发现如果需要防止炸弹爆炸，我们需要让寄存器eax运行结束func4之后值为7，因此这时我们重新进入func4中观察该递归函数，发现其中的运算操作为除法和加法，在代码<34>和<41>处，为两者相加除以2，类似于寻找平均值的操作，然后判断，通过运行我们发现递归函数首次运行后寄存器eax值为0，而后递归一次add $0x1,%eax操作将eax+1，我们如果要得到答案7，需要递归7次，而后我们发现，在操作之前一共有三个值，分别为输入的值、0、14，因此如果想要递归循环7次，我们需要输入14，这样才能保证在递归7次之前不会跳出函数，因此第一个数字为14，第二个数字直接读取会发现为7，因此拆弹成功，答案为14、7



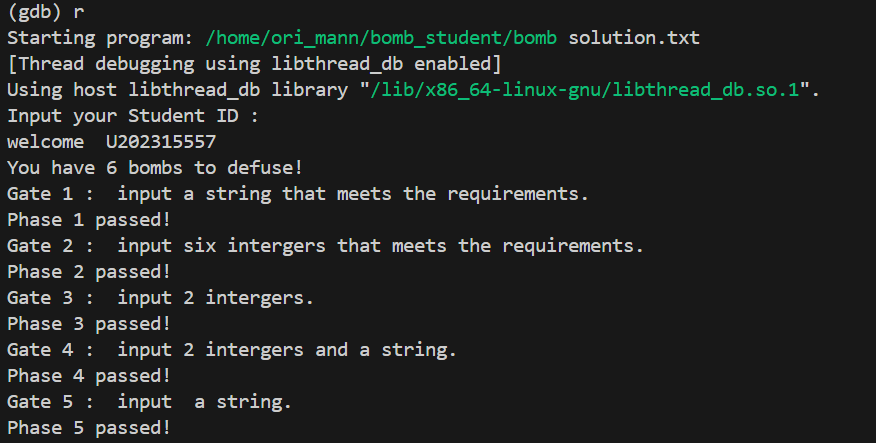
**第五关**

这关考察的内容为指针和数组的访问，通过分析函数phase\_5发现最后判定炸弹时调用了判断字符串是否相同的函数，因此我们需要得到与提供的字符串相同的字符串，但是这关与第一关不同的点在于，第一关是输入的字符串直接进行比较，但是这一关进行了对应的转换，相当于有一个密码本存在于函数之中，我们输入的字符串会先进行一步转化，因此我们要找到数组存储的位置，这时候也用到了指针

通过读取array数组我们会发现密码本如下

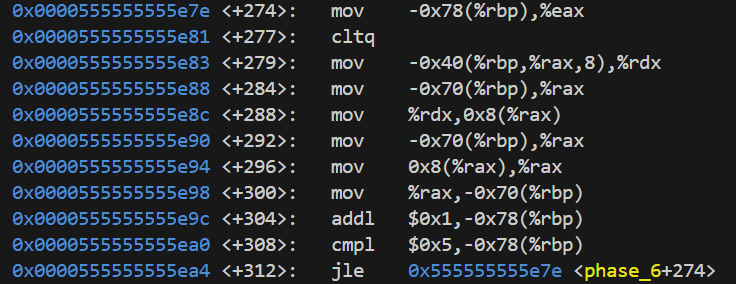
我们找到对应的字符串即可，答案如下

mvcdxw

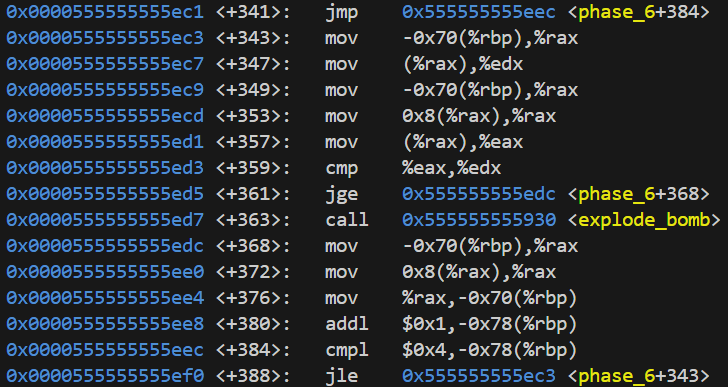


**第六关**

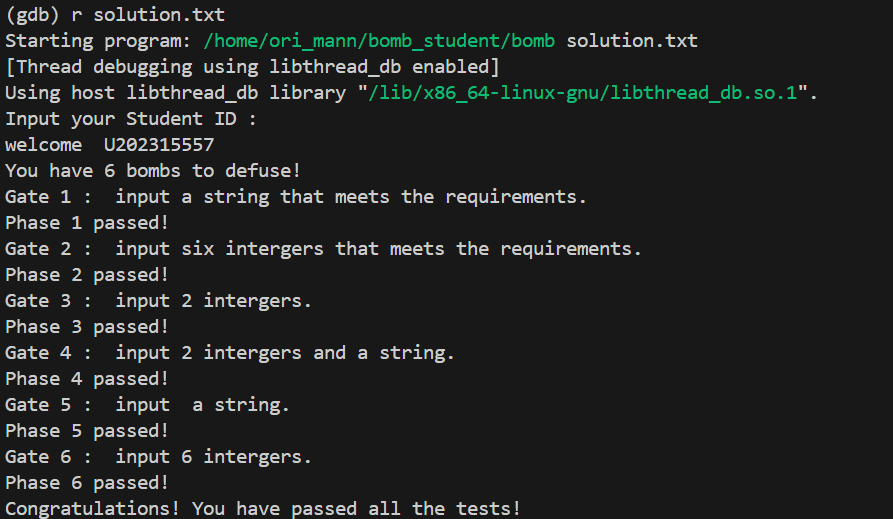
根据提示以及c语言挖空代码，我们可以知道这一关的炸弹答案有几个要求，首先排序之后是按照降序进行排列，然后一共有六个数字，最后是找出他们正确的逻辑顺序，从而拆除炸弹，知道这些之后我们最关键的拆弹步骤就是找到指针的指向顺序即可，我们可以在代码中进行分析，找到关键的循环操作，看他们之中指针的指向的值即可判断他们的顺序



这一部分是进行的循环验证操作



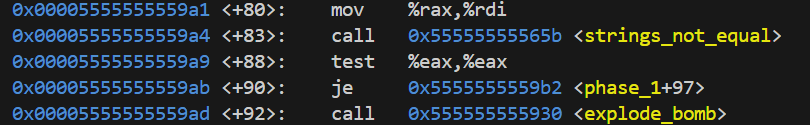
从这里我们直接通过gdb查看指针的值即可，可以得到炸弹的指针顺序为3 2 6 4 1 5，因此我们按照降序排列，答案为2 5 6 3 1 4



**（2）拆除炸弹的过程中关键操作**

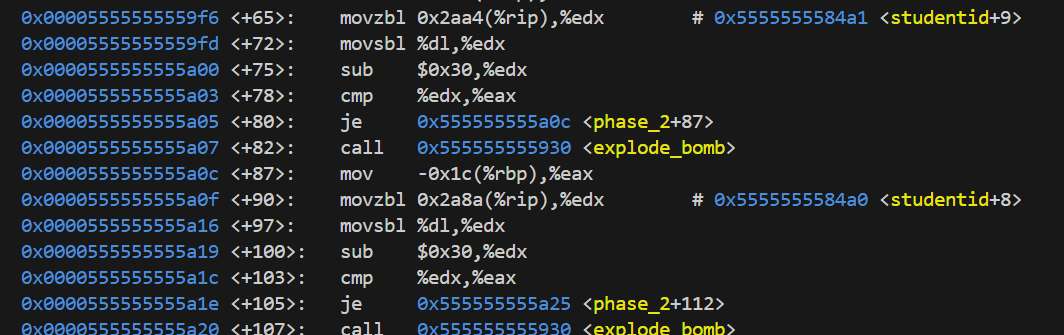
我认为在拆弹过程中最重要的操作是寻找炸弹爆炸的条件，通过条件来反推我们需要的答案，比如说第一关，首先定位到<explode\_bomb>函数的所在位置，然后看他的前几句或者跳转的语句，这些语句就是拆弹的关键，下面将会具体分析

**第一关**

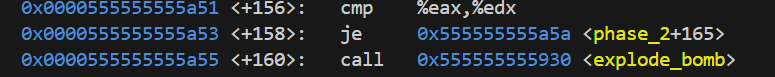
****

拆弹关键在语句<+80>和<+83>，因此我们只需要查看寄存器的值即可

**第二关**

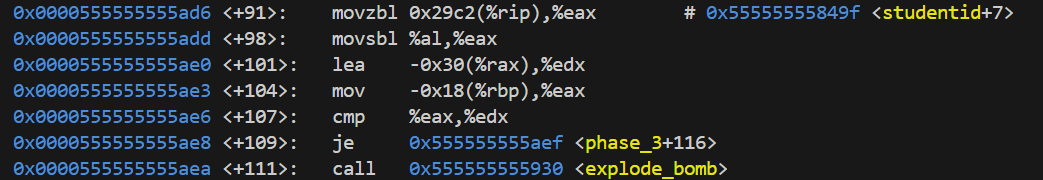
****

前两个数字与学号有关

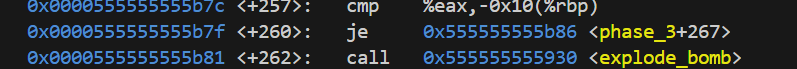


后面为递增的公式操作，如果我们没有分析出来公式，也可以通过查看寄存器eax和edx也可以得到答案

**第三关**



第一个数字为学号第七位

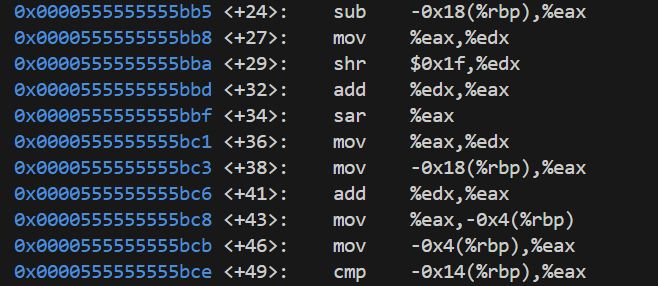
速通的话，跳转之后读取内存值即可

**第四关**

首先找到炸弹爆炸的地方

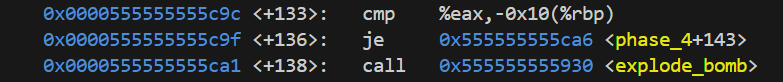


在这里我们需要让eax为7才可以避免爆炸，发现在func4中确定eax的值，关键在于找到运算的语句，在func4中寻找可知二分法计算在



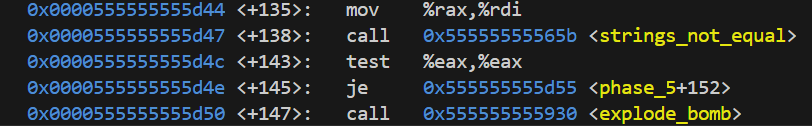
因此分析运算可知二分法，找到需要填入的数和需要的数字即可

第二个数字与第一个数字是独立的关系，因此只需要找到答案即可

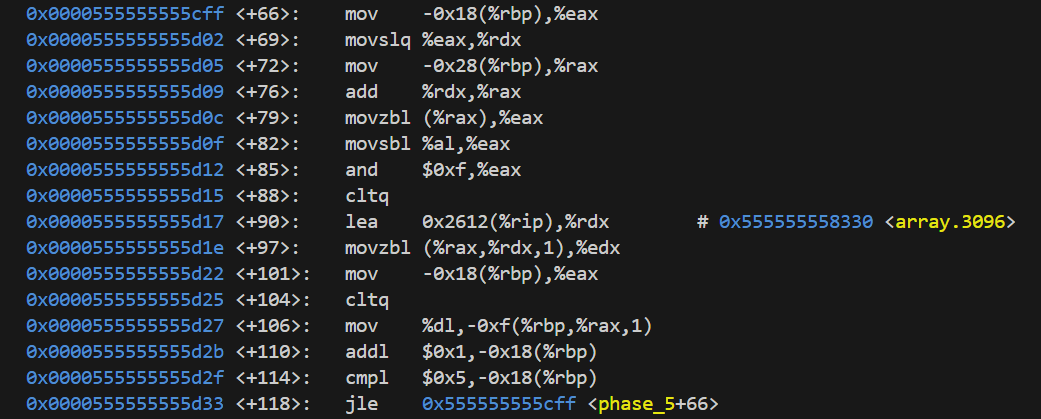


**第五关**

找到炸弹



可知我们需要找到rdi和rax的值，我们输入的值为rdi，然而我们发现这里与我们输入的值存在变化，因此我们反推，找到密码本存在的位置



这里存在循环，即按照密码本将我们输入的值加密成另外的值，因此破译array密码本即可

**四、体会**

在二进制炸弹实验中，我通过分析和反汇编程序代码，逐步破解了各个阶段的输入要求，获得了宝贵的编程经验。这次实验让我深入理解了程序的底层执行机制，尤其是在涉及汇编、指针、递归、数据结构等方面的内容时。通过对反汇编代码的分析，我学会了如何从机器指令中推断程序逻辑，并通过调试工具验证我的猜想和假设。

实验的各个阶段涉及字符串比较、循环结构、条件分支、递归调用、指针操作、数组访问和链表处理等不同编程概念。在应对这些挑战时，我掌握了如何通过解析汇编代码中的关键指令，确定数据流和控制流的走向，最终推断出每个阶段所需的正确输入。特别是在链表、指针和结构体的操作上，我的理解有了进一步的加深。

整体而言，这次实验极大地增强了我对底层编程的理解，提高了我对复杂问题的分析能力，同时也让我掌握了如何高效地运用调试工具来解决实际的编程挑战。这为我今后深入研究计算机系统和底层代码奠定了坚实的基础。