

**课 程 实 验 报 告**

**课程名称： 计算机系统基础**

**实验名称： 机器级语言理解**

**院 系 ：计算机科学与技术**

**专业班级 ： 计算机2305班**

**学 号 ： U202315677**

**姓 名 ： 岳皓**

**指导教师 ： 班鹏新**

**2025 年 4 月 3 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**

**任务1** 二进制炸弹拆除

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);**

要求输入两个整数数据，与程序中给定的数据比较，相等则过关。提示：在自动生成数据时，使用了 switch … case 语句。

**三、实验记录及问题回答**

**（1）实验任务1 的实验记录**

根据分析和运算，最终得到前面三个阶段的总答案并最终进行运行，运行结果如图1。

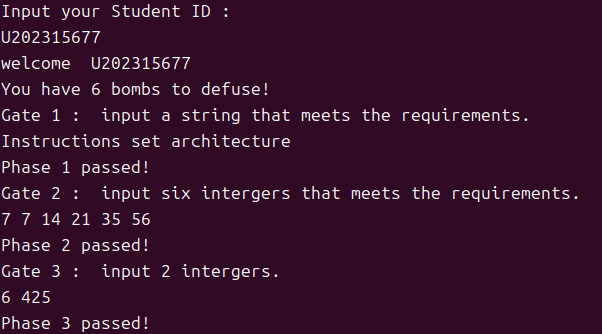


图 1 运行结果

1. **拆除炸弹的过程中关键操作**

**阶段一：**

在终端中通过设置断点（break phase\_1）然后来运行（disas phase\_1）得到阶段一的的反汇编代码如下图：

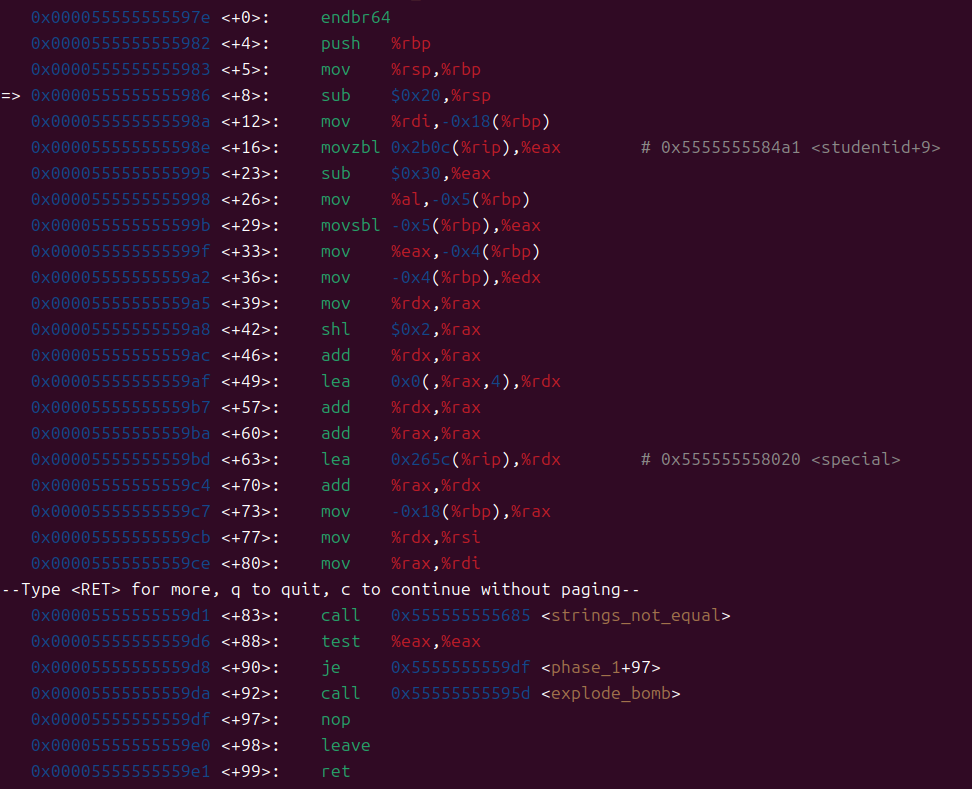
****

图 2 phase1反汇编代码

通过分析，代码前面先执行设置栈帧、分配32字节空间、保存参数、从参数中加载一个字节到eax、字符转换为数值等操作，从<+39>行到<+60>行，对于edx保存的值进行操作，利用rdx和rax，对值进行乘以50的操作（即先通过右移两位进行乘4操作，再相加原来的值得到原来5倍的值，接着进行乘4操作来得到原来20倍的值，最后与之前五倍的值相加得到原来25倍的值后自身相加得到原来50倍的值并存在rax），后面的代码则为获取指定地址（rdx=special+input\_value\*50），其中special为0x265c（%rip）的地址，其值为0x555555558020。

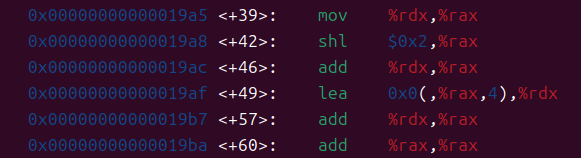


图 3 值处理过程

已知学号最后一位为7，故根据上述分析，指定字符地址0x0x55555555817e，通过指令搜索相关地址的内容如下图：

QQ20250403-192738

图 4 相应位置存储的字符

所以阶段一的字符答案为Instructions set architecture。

**阶段二：**

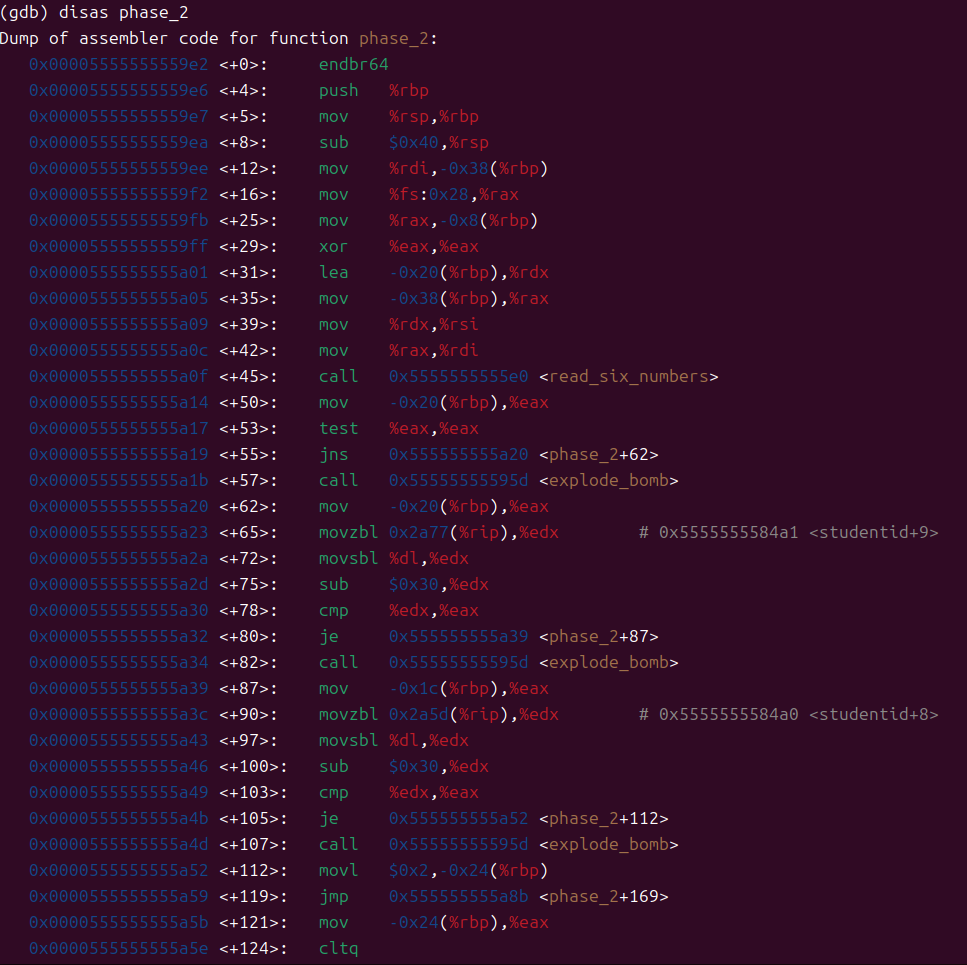


图 5 phase\_2反汇编代码上半部分

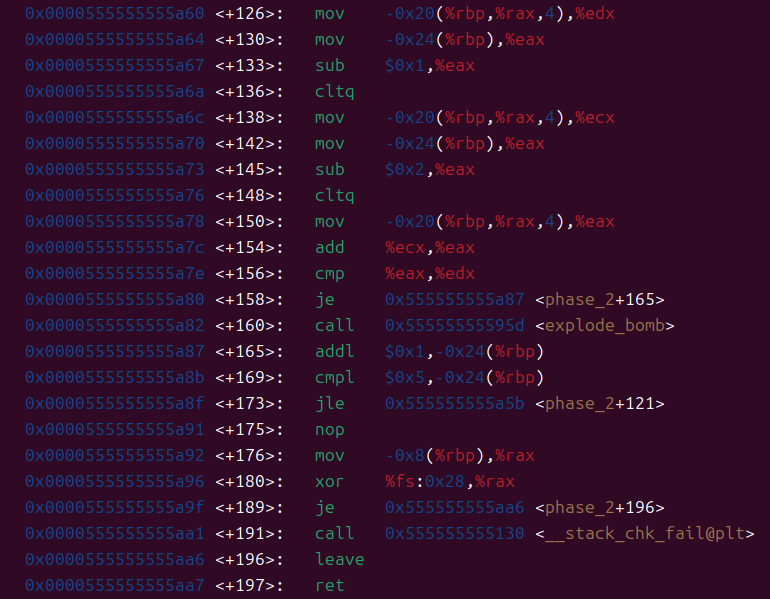


图 6 phase\_2反汇编代码下半部分

在终端中通过设置断点（break phase\_2）然后来运行（disas phase\_2）得到阶段二的的反汇编代码如图5和图6。

通过分析图5和图6的反汇编代码，可知前面先分配栈空间、清零eax等操作，然后调用read\_six\_numbers函数，其中函数需要读入6个数字，如果未满6个则会直接bomb，通过分析发现，对于edx，前面两次分别读入studentid+9和studentid+8处的数字，对于本人学号而言，其按照排序两个数字均对应7，然后再与输入的数进行比较，如不同则bomb。

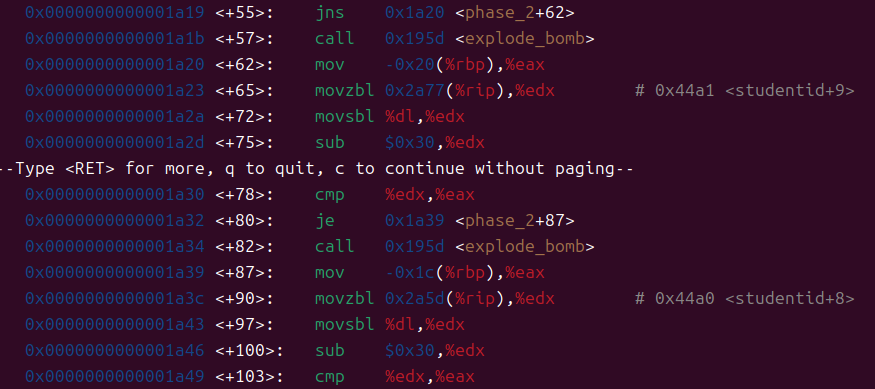


图 7 对前面两个数的处理相对应的代码

后面的代码为循环检查第3至第6个数，而其逻辑为每个数字须为前两个数字之和，即满足斐波那契数列的规则，所以对于前面两个数均为7，答案应为7 7 14 21 35 56。

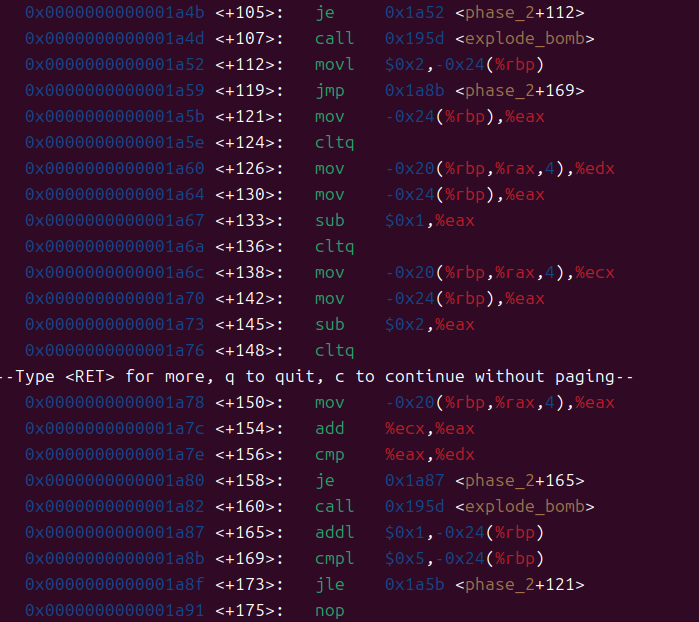


图 8 第3~第6个数的比较

**阶段三：**

在终端中通过设置断点，然后来运行（disas phase\_3）得到阶段三的的反汇编代码如图9和图10。

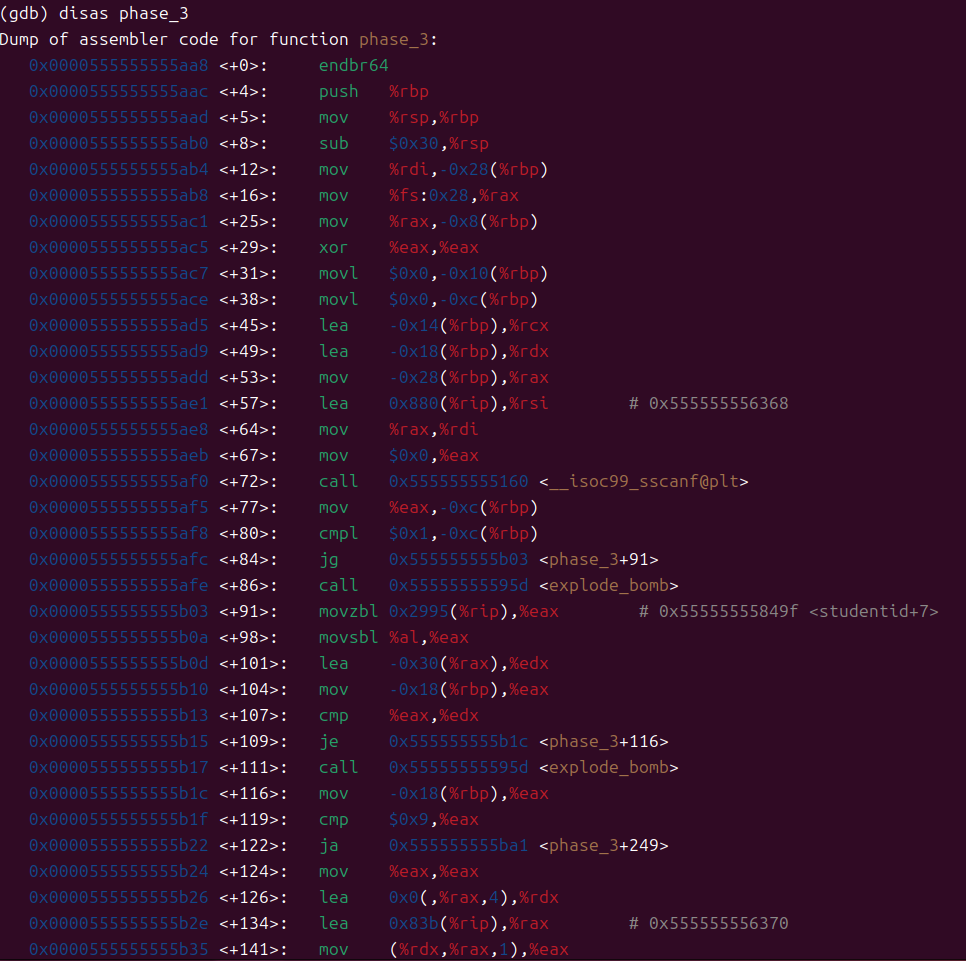


图 9 phase\_3反汇编代码上半部分

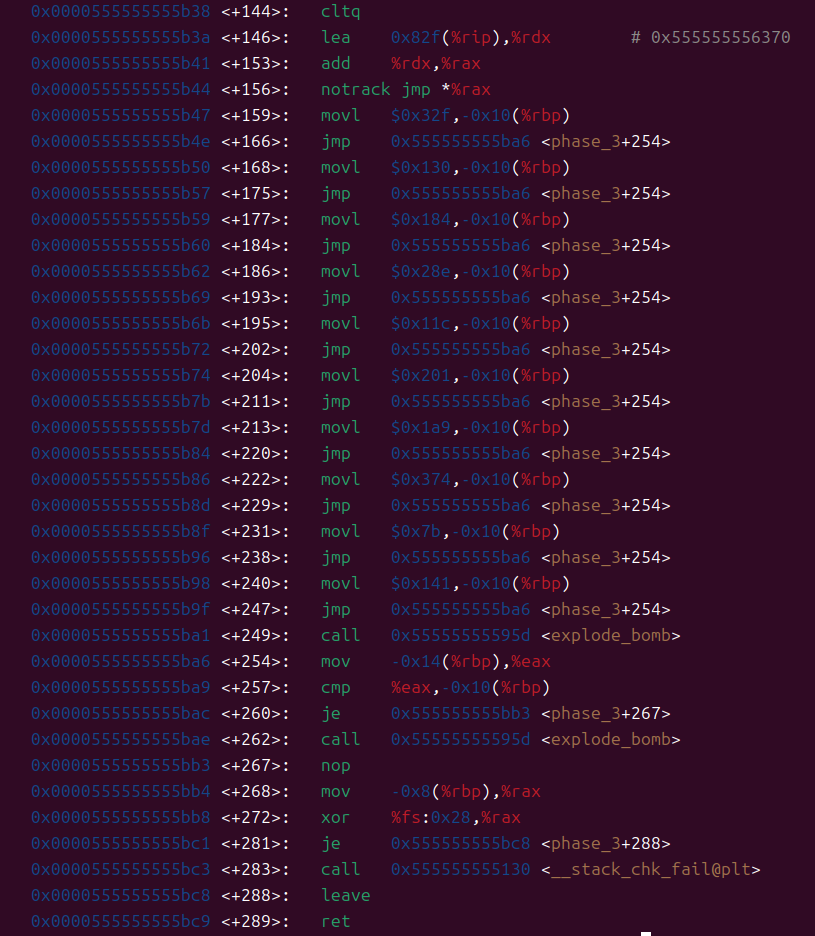


图 10 phase\_3反汇编代码下半部分

通过分析代码，可知前面为分配48字节栈空间、清零eax等操作，然后调用sscanf函数来从输入中读取两个整数，其中，第一个数存入-0x18(%rbp)，第二个数存入-0x14(%rbp)，然后检查是否有两个参数，无则直接bomb。

对后续代码进行分析，可知第一个数与学号的倒数第三位进行比较，所以第一个数应为6。

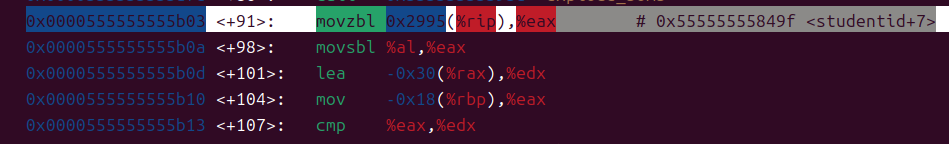


图 11 第一个数的比较代码

后面根据第一个数计算偏移量并将该偏移量存放在rax，根据rax中存放的值从而可以定位到指定值，即为switch...case语句的判断，输入第一个数6后，可以通过rax的值定位到第二个数的执行位置，所以通过重新设置断点，再通过break \*phase\_3+156语句，将程序运行到rax进行比较，然后查看rax的值得到将要进行比较的语句的位置，通过该位置可知相比较的数具体为多少，通过info指令查看寄存器rax保存的值，如图12。

然后根据值定位到相对应的位置，如图13，可知第二个比较的数应为0x1a9=256\*1+16\*10+9=425。

故答案为6 425。

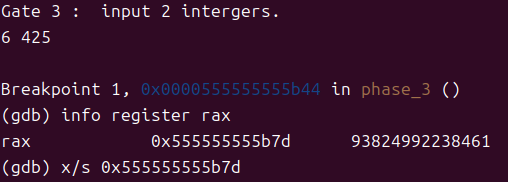


图 12 查看rax的值

QQ20250403-210417

图 13 第二个数的相应比较位置

1. **体会**

进行本次实验的一开始，因为对于Ubuntu环境的不熟悉，导致起手较慢，在翻找一些文章后才渐渐熟悉操作，对于前面三关，难度较为容易，只需对反汇编代码进行分析即可知道运行逻辑，在经过一些简单的调试后，得到了前三关的答案。在这个过程中，加强了对于反汇编代码的理解和指令的运用，收益颇多。