

**实验报告**

**实 验（三）**

题 目 Binary Bomb

二进制炸弹

专 业 计算机类

学　　 号 1190200526

班　　 级 1903002

学 生 沈城有

指 导 教 师 郑贵滨

实 验 地 点 G709

实 验 日 期 2021.04.23

**计算机科学与技术学院**

**目 录**

[第1章 实验基本信息 - 3 -](#_Toc70026049)

[1.1 实验目的 - 3 -](#_Toc70026050)

[1.2 实验环境与工具 - 3 -](#_Toc70026051)

[1.2.1 硬件环境 - 3 -](#_Toc70026052)

[1.2.2 软件环境 - 3 -](#_Toc70026053)

[1.2.3 开发工具 - 3 -](#_Toc70026054)

[1.3 实验预习 - 3 -](#_Toc70026055)

[第2章 实验环境建立 - 4 -](#_Toc70026056)

[2.1 Ubuntu下CodeBlocks反汇编（10分） - 4 -](#_Toc70026057)

[2.2 Ubuntu下EDB运行环境建立（10分） - 4 -](#_Toc70026058)

[第3章 各阶段炸弹破解与分析 - 5 -](#_Toc70026059)

[3.1 阶段1的破解与分析 - 5 -](#_Toc70026060)

[3.2 阶段2的破解与分析 - 6 -](#_Toc70026061)

[3.3 阶段3的破解与分析 - 7 -](#_Toc70026062)

[3.4 阶段4的破解与分析 - 8 -](#_Toc70026063)

[3.5 阶段5的破解与分析 - 10 -](#_Toc70026064)

[3.6 阶段6的破解与分析 - 11 -](#_Toc70026065)

[3.7 阶段7的破解与分析(隐藏阶段) - 14 -](#_Toc70026066)

[第4章 总结 - 15 -](#_Toc70026067)

[4.1 请总结本次实验的收获 - 15 -](#_Toc70026068)

[4.2 请给出对本次实验内容的建议 - 15 -](#_Toc70026069)

[参考文献 - 16 -](#_Toc70026070)

# 第1章 实验基本信息

## 1.1 实验目的

熟练掌握计算机系统的ISA指令系统与寻址方式；

熟练掌握Linux下调试器的反汇编调试跟踪分析机器语言的方法；

增强对程序机器级表示、汇编语言、调试器和逆向工程等的理解。

## 1.2 实验环境与工具

### 1.2.1 硬件环境

X64 CPU；2GHz；2G RAM；256GHD Disk 以上

### 1.2.2 软件环境

Windows7 64位以上

VirtualBox/Vmware 11以上

Ubuntu 16.04 LTS 64位/优麒麟 64位

### 1.2.3 开发工具

GDB/OBJDUMP；EDB；KDD等

## 1.3 实验预习

上实验课前，必须认真预习实验指导书（PPT或PDF）。

了解实验的目的、实验环境与软硬件工具、实验操作步骤，复习与实验有关的理论知识。

请写出C语言下包含字符串比较、循环、分支（含switch）、函数调用、递归、指针、结构、链表等的例子程序sample.c，生成执行程序sample.out。

用gcc -S或CodeBlocks或GDB或OBJDUMP等，反汇编，比较。

列出每一部分的C语言对应的汇编语言。

修改编译选项-O (缺省2)、O0、O1、O2、O3，-m32/m64。再次查看生成的汇编语言与原来的区别。

注意O1之后无栈帧，EBP做别的用途。-fno-omit-frame-pointer加上栈指针。

GDB命令详解 –tui模式 ^XA切换 layout改变等等

有目的地学习: 看VS的功能GDB命令用什么？

# 第2章 实验环境建立

## 2.1 Ubuntu下CodeBlocks反汇编（10分）

CodeBlocks运行hellolinux.c。反汇编查看printf函数的实现。

要求：C、ASM、内存(显示hello等内容)、堆栈（call printf前）、寄存器同时在一个窗口。

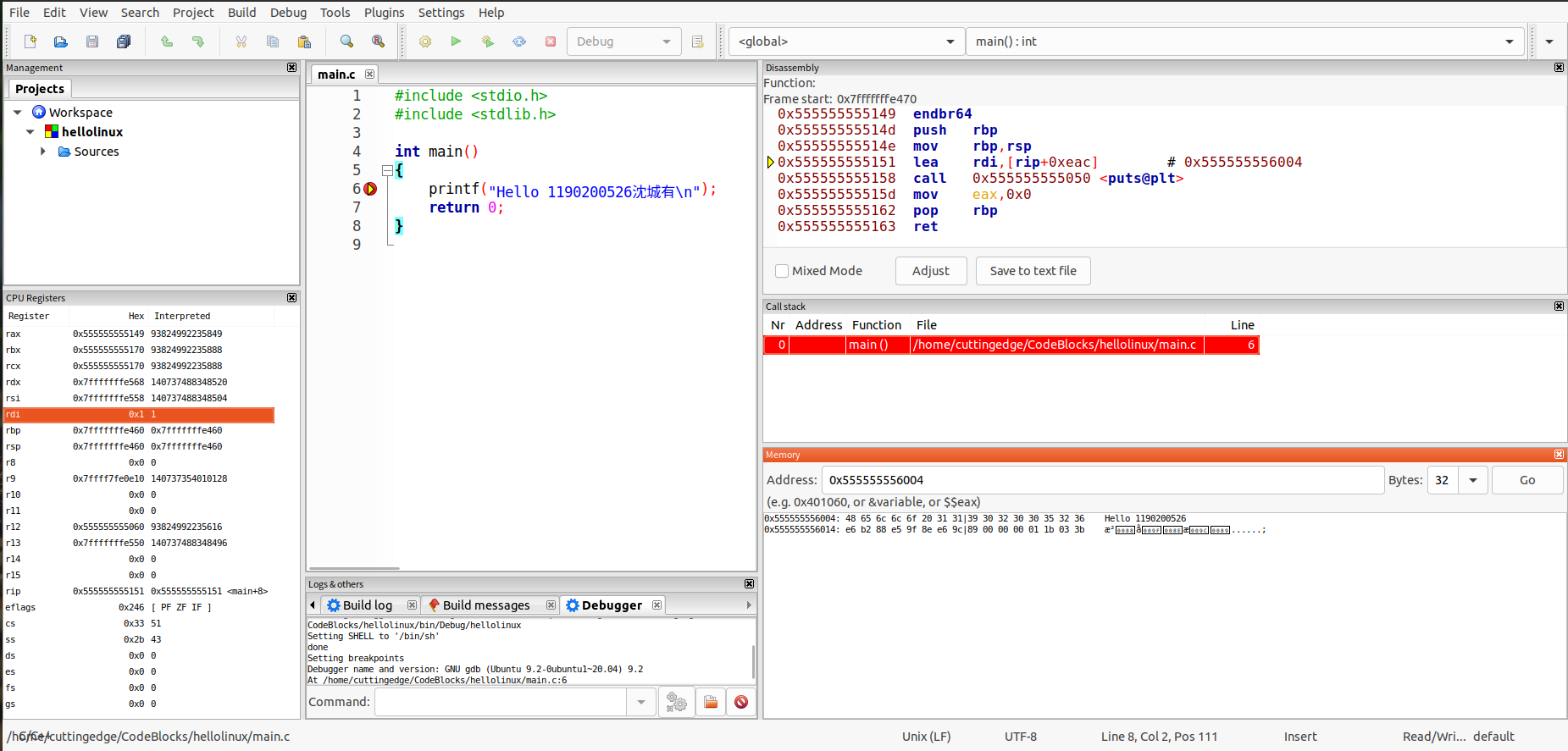


图2-1 Ubuntu下CodeBlocks反汇编截图

## 2.2 Ubuntu下EDB运行环境建立（10分）

用EDB调试hellolinux.c的执行文件，截图，要求同2.1

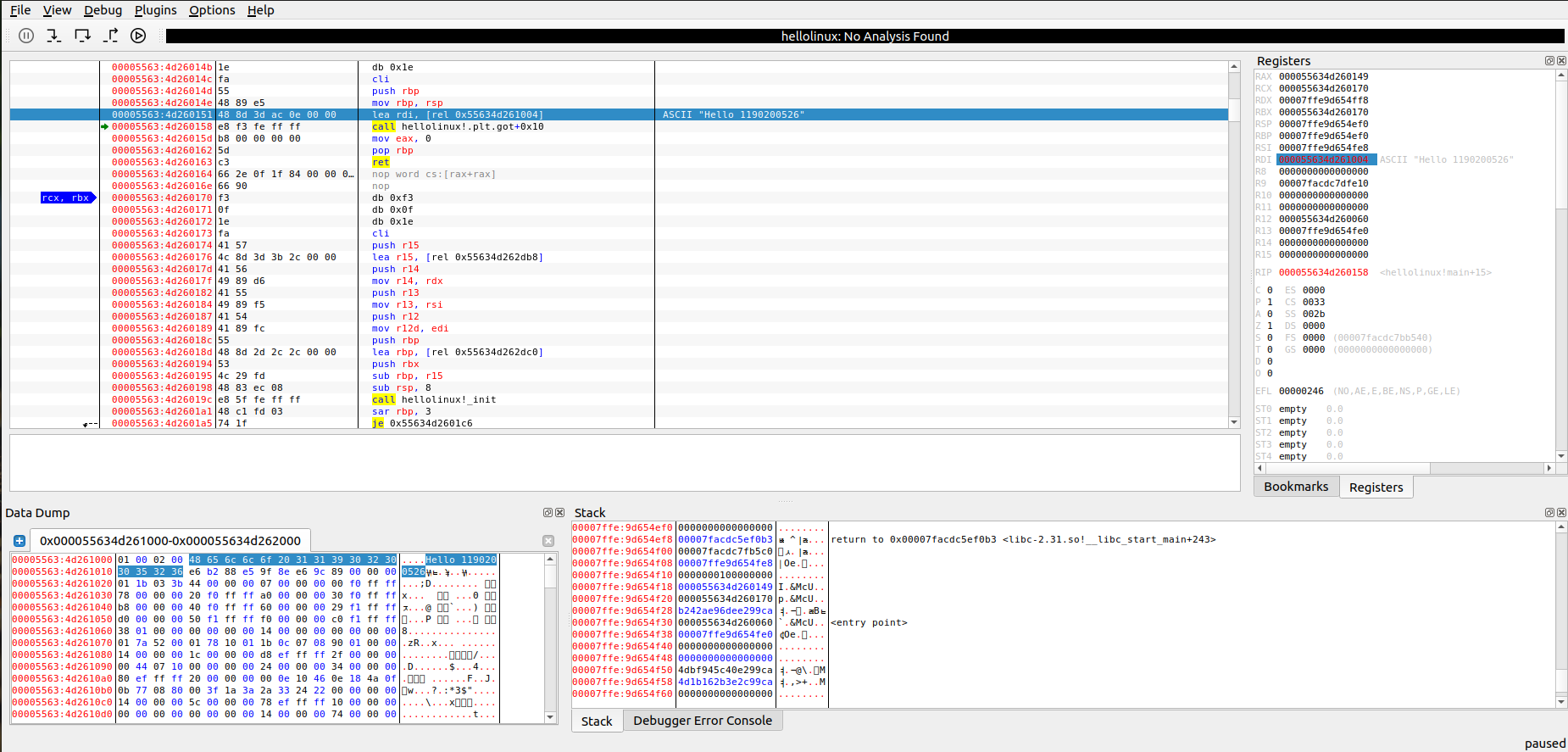
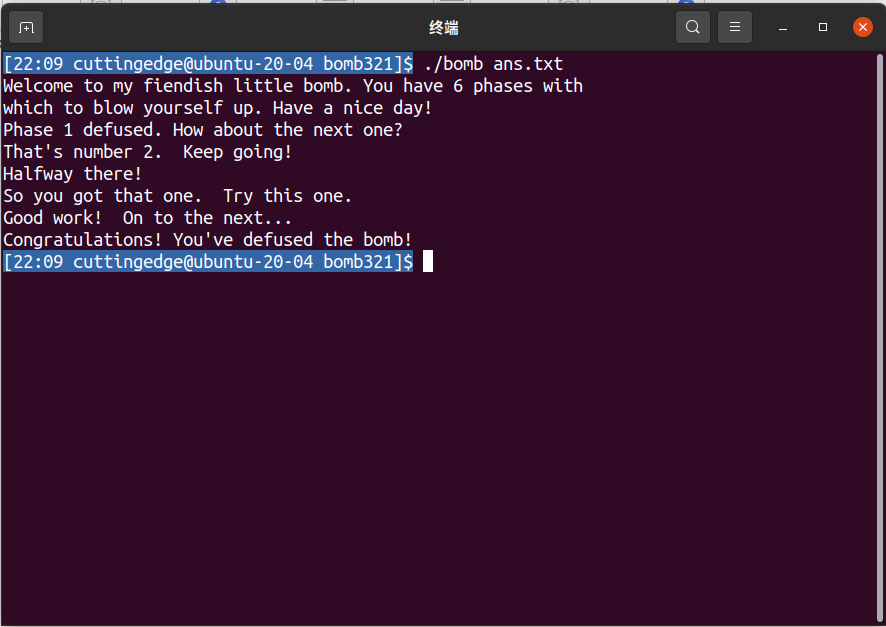


图2-2 Ubuntu下EDB截图

# 第3章 各阶段炸弹破解与分析

每阶段15分（密码10分，分析5分），总分不超过80分

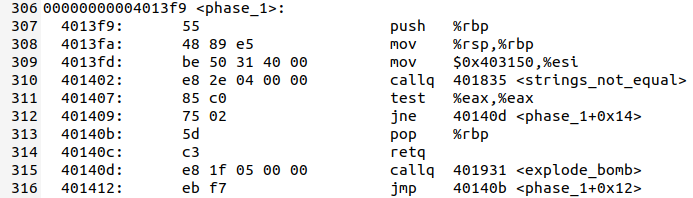


## 3.1 阶段1的破解与分析

密码：I am for medical liability at the federal level.

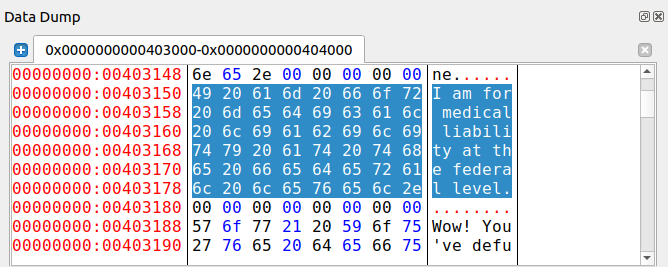
破解过程：

下图为phase\_1函数的汇编指令：



可知此函数将输入字符串（寄存器RDI指向的字符串）与寄存器RSI指向的字符串进行比较（利用string\_not\_equal函数），不相同（返回值不为0）则调用explode\_bomb函数，导致炸弹爆炸。

通过EDB查看对应内存位置答案字符串如下：

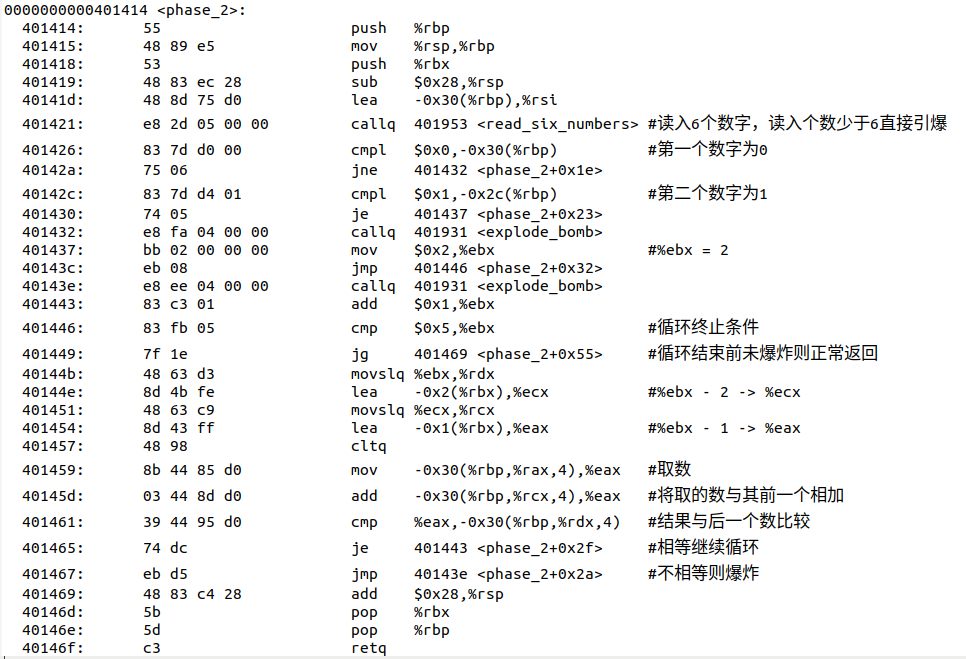


## 3.2 阶段2的破解与分析

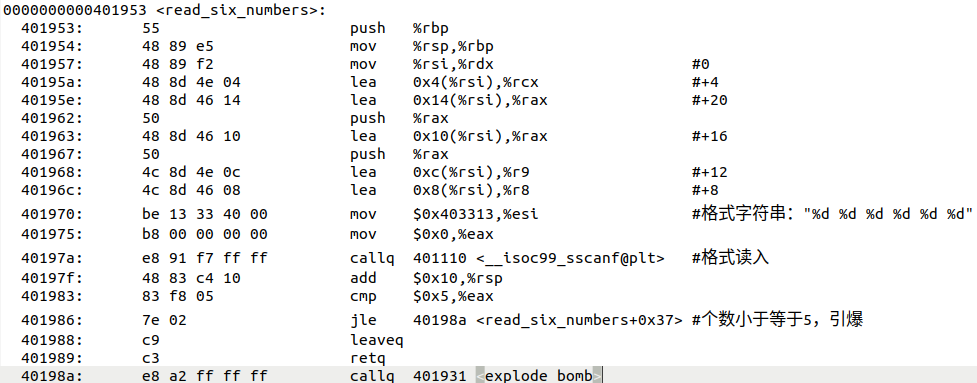
密码：0 1 1 2 3 5

破解过程：

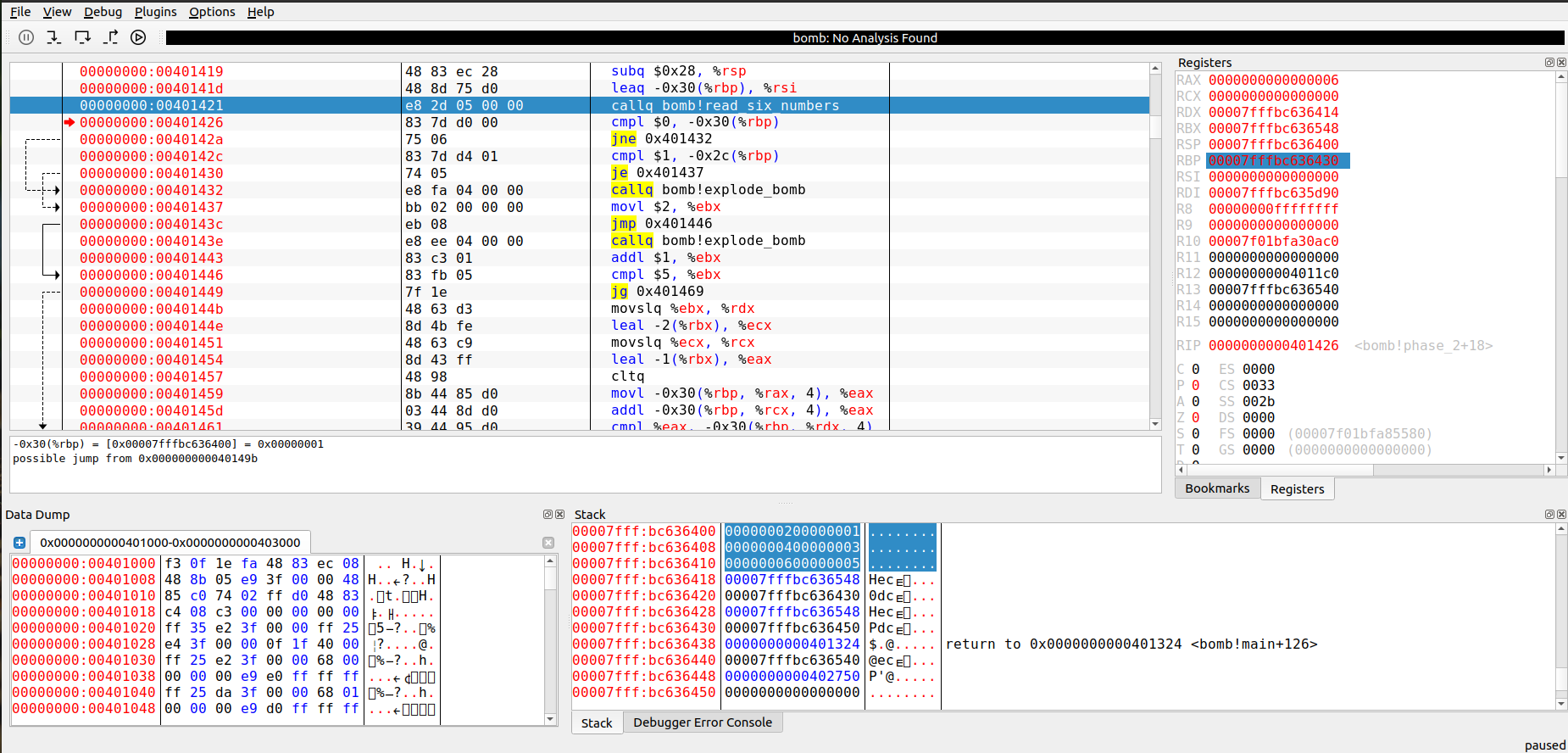
下图为phase\_2函数的汇编指令及分析：



下图为read\_six\_numbers函数的汇编指令及分析：



如下图，通过简单的输入测试可确定读入后的具体保存位置：



第1个数→第6个数：(%rbp-0x30)→(%rbp)

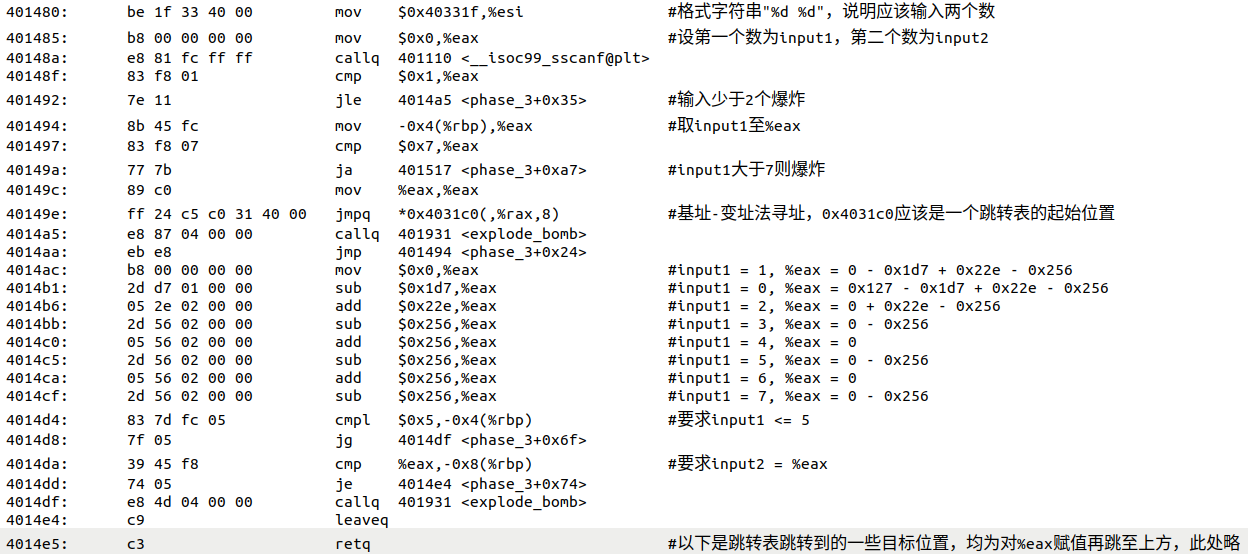
由以上分析可得，程序中有一个类似于检测斐波那契数列的循环体，且可知前两个数字分别为0、1，之后四个数字即可推出，分别为0+1=1，1+1=2，1+2=3，2+3=5。

## 3.3 阶段3的破解与分析

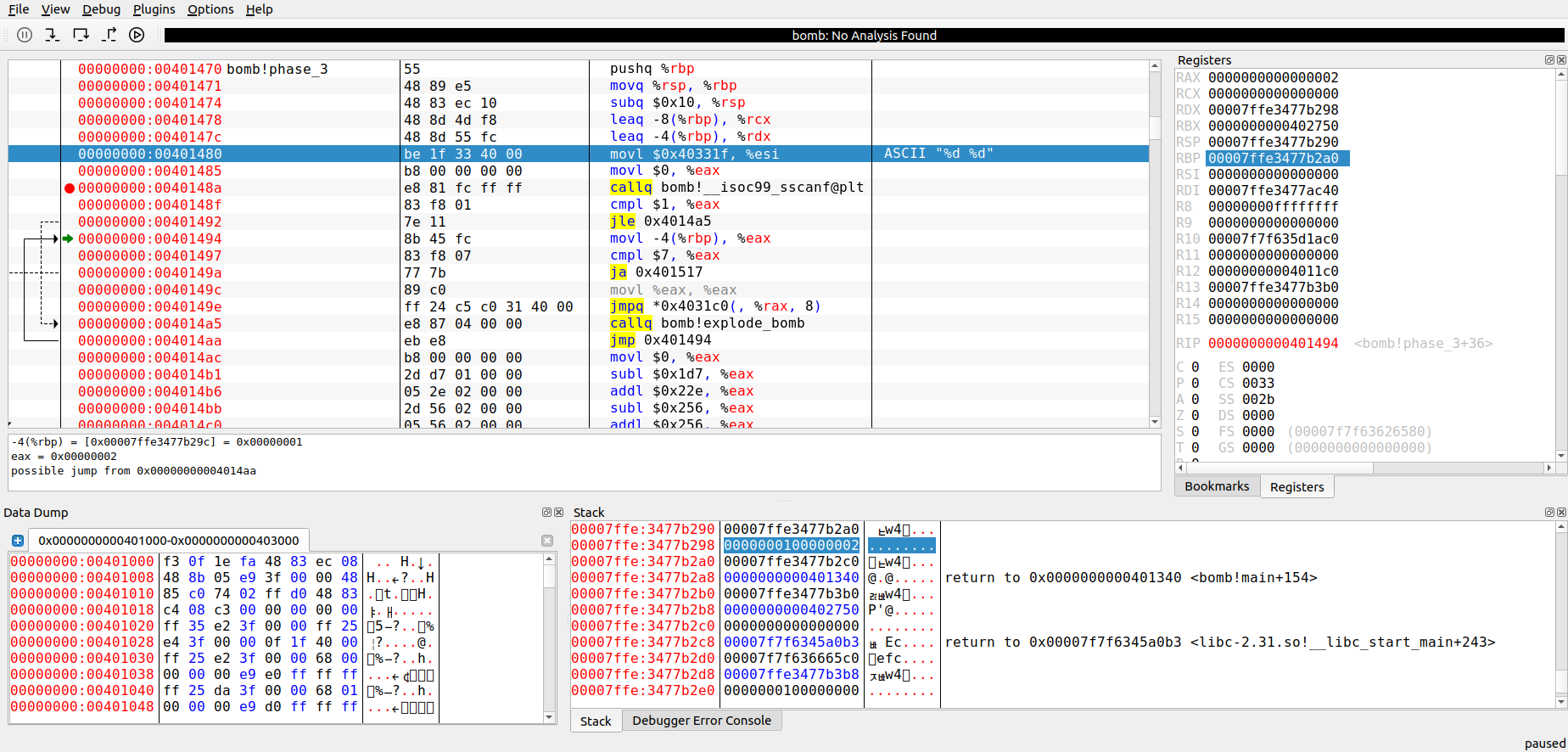
密码：4 0 （注：其中一种可行数字组合）

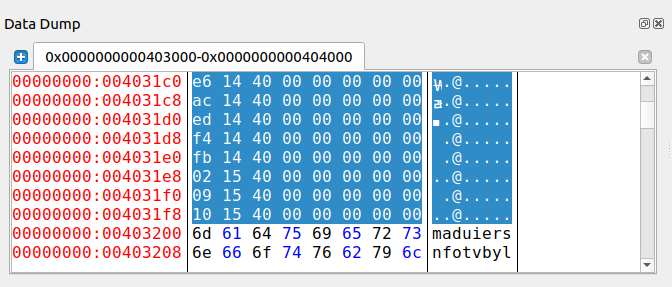
破解过程：

下图为phase\_3函数部分汇编代码及分析：



如下图，输入1 2进行测试，可知阶段3读入两个数，第一个数保存于(%rbp-4)指向的栈位置，第二个数保存于(%rbp-8)指向的栈位置。





上图为汇编代码分析中所提到的跳转表。结合汇编代码，能够确定两个输入数字之间的对应关系，且要求第二个数不大于5，故正确的输入数字组合共有6组，分别为：

[0 -216] [1 -511] [2 -40] [3 -598] [4 0] [5 -598]。

## 3.4 阶段4的破解与分析

密码：8 35

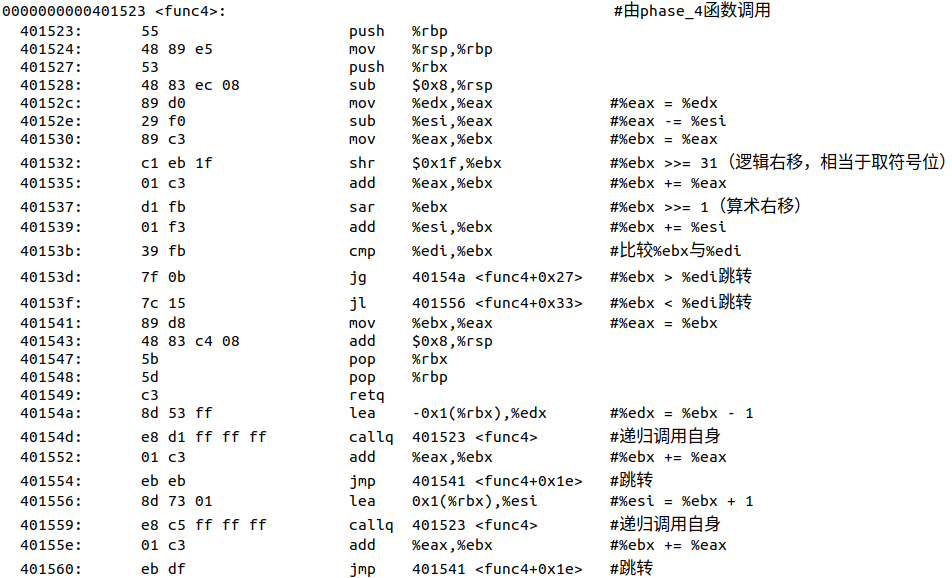
破解过程：

读入过程同阶段三，保存顺序也同理。

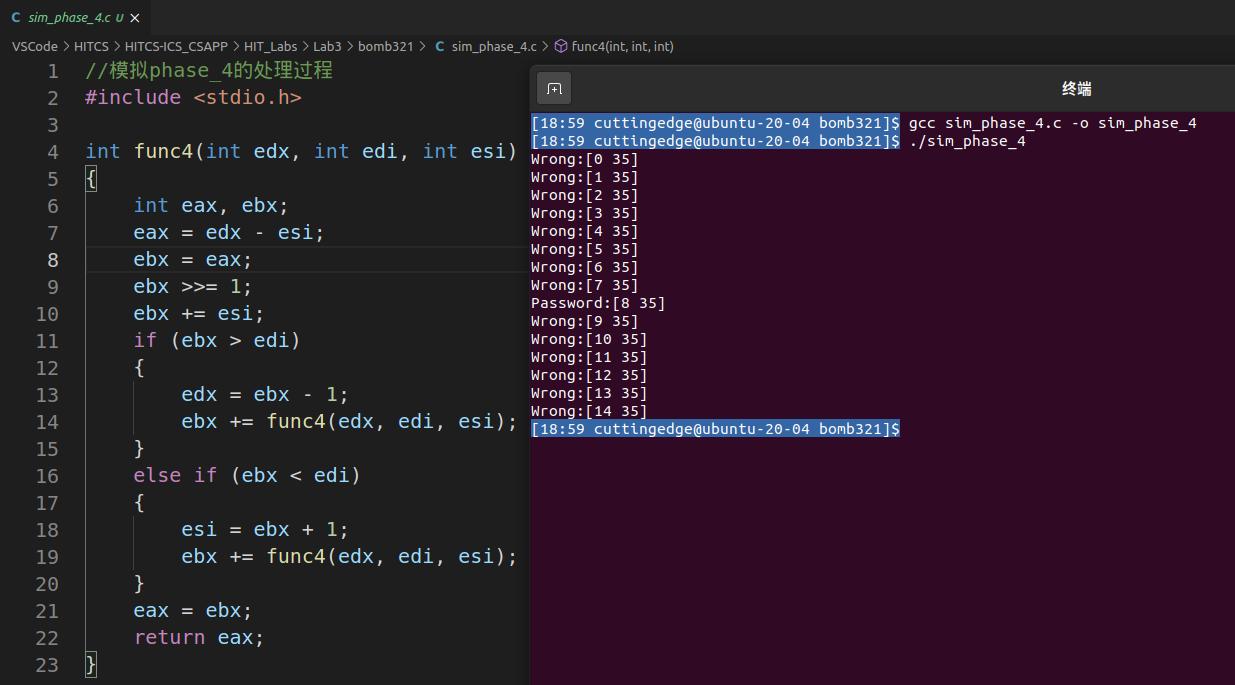
下图为phase\_4函数部分汇编代码及分析：



下图为递归调用的func4函数汇编代码及分析：



根据分析，我编写了C语言程序模拟这一过程并最终得到密码，模拟程序部分代码及运行结果见下图：

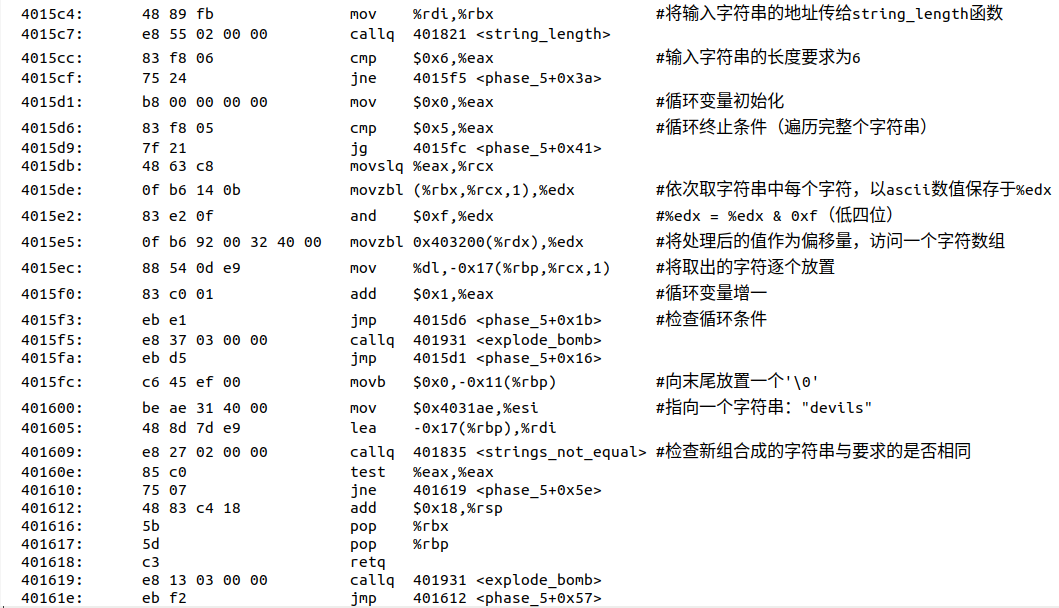


## 3.5 阶段5的破解与分析

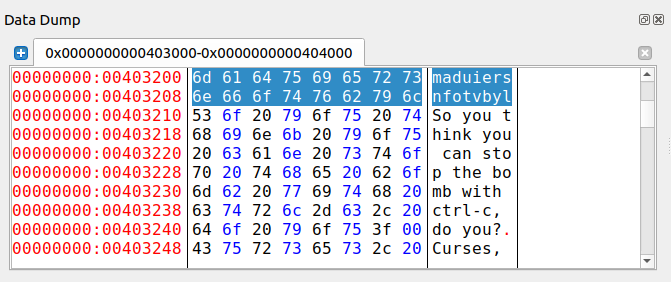
密码：BELDOG

破解过程：

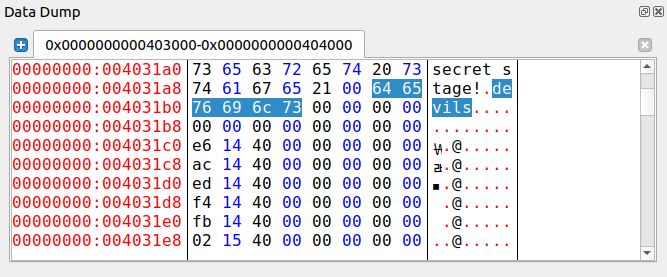
下图为phase\_5函数部分汇编代码及分析：



下图为用于组合要求字符串所用到的字符数组：



下图为要求组合得到的字符串：



由以上可知，我们需要构造一个6个字符的字符串，用其ascii码与0xf做与运算后获得的偏移量取出字符数组里的字符构成目标字符串。

过程如下：

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| 目标字符串字符 | d | e | v | i | l | s |
| 偏移量 | 2 | 5 | 12 | 4 | 15 | 7 |
| 输入字符串字符编码低四位 | 0010 | 0101 | 1100 | 0100 | 1111 | 0111 |
| 一种可行的编码 | 64+2 | 64+5 | 64+12 | 64+4 | 64+15 | 64+7 |
| 对应字符 | B | E | L | D | O | G |

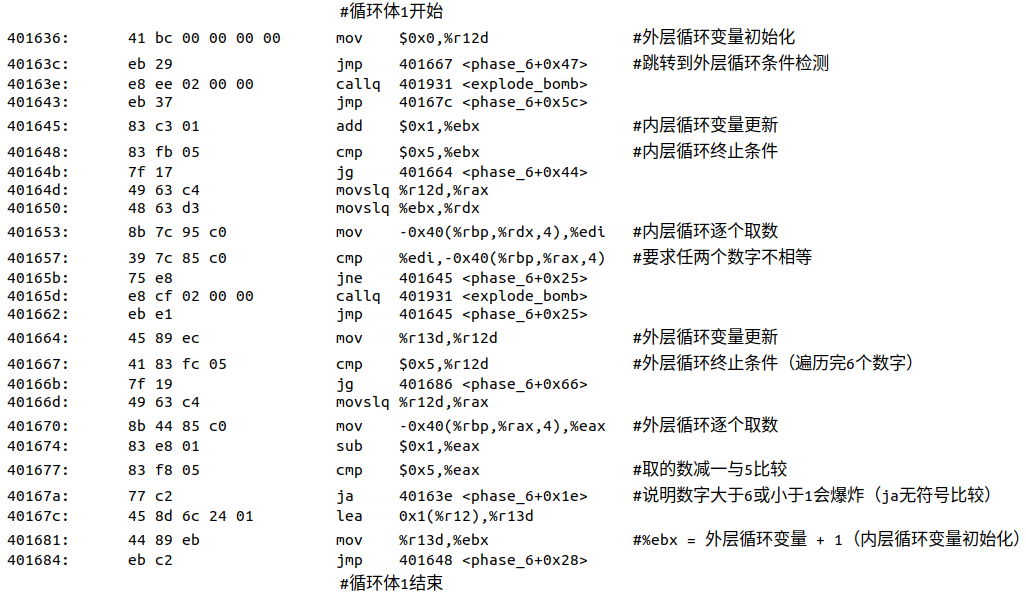
## 3.6 阶段6的破解与分析

密码：2 3 5 6 1 4

破解过程：

由phase\_6函数汇编代码开头部分调用read\_six\_numbers函数可以看出，输入格式类似于phase\_2，即6个整数。

分析phase\_6函数的汇编代码，发现其主要由四个循环体构成，下图为第一个循环体的汇编代码及分析：



类似于以下的C代码功能：

for (int i = 0; i < 6; ++i)

{

if (num[i] > 6 || num[i] < 0)

return -1; //炸弹爆炸

for (int j = i + 1; j < 6; ++j)

{

if (num[i] == num[j])

return -1; //炸弹爆炸

}

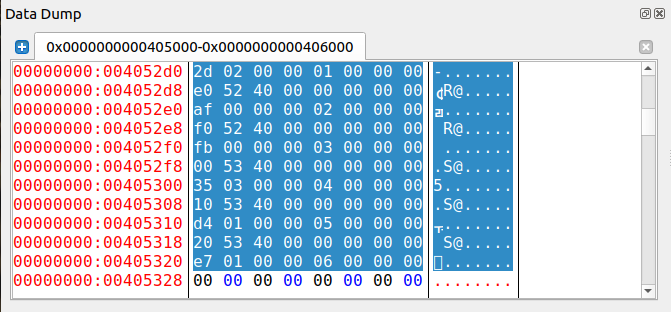
}

故这个循环体的作用为判断6个数是否互不相同，且取值范围是否为 ，若不是则爆炸。

下图为phase\_6函数的第二个循环体的汇编代码及分析：



下图为以上分析中所提到的链表结构：



可以看成4字节数值+4字节序号（1~6）+8字节地址（指向下一个结点）。故此链表可如下表示：

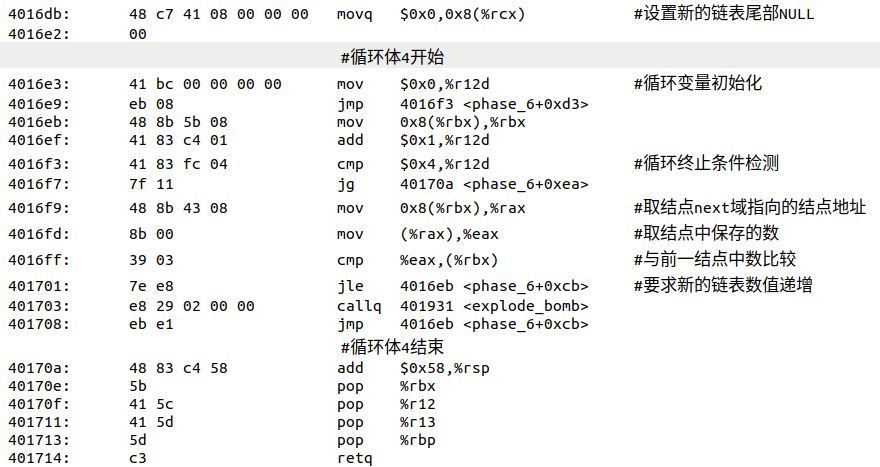
由此分析，第二个循环体的作用在于按照输入的6个数取对应标号的链表结点next地址，重新排列放置于栈的另一块连续空间里。

下图为phase\_6函数的第三个循环体的汇编代码及分析：



循环体3主要功能是重新链接整个链表。

下图为phase\_6函数的第四个循环体的汇编代码及分析：



循环体4的主要功能是检测新链接的链表数值部分是否递增，函数至此主要功能已经结束。

根据以上分析，将链表各点按数值升序得序号：2 3 5 6 1 4。

此数字序列即为密码。

## 3.7 阶段7的破解与分析(隐藏阶段)

密码：

破解过程：

# 第4章 总结

## 4.1 请总结本次实验的收获

1. 本次实验学会了Code::Blocks、EDB的反汇编调试方法；
2. 深化了对汇编指令的理解；
3. 锻炼了分析汇编指令和汇编程序过程的能力；
4. 理解了汇编语言中链表、数组递归、循环等结构的表示及实现。

## 4.2 请给出对本次实验内容的建议

建议课件补充EDB使用和配置的入门知识。例如在完成EDB的安装后，如何将EDB默认汇编指令格式从Intel修改至AT&T，以及如何修改字号、如何查找函数名、如何设置断点及断点的类型与区别等等。

注：本章为酌情加分项。

# 参考文献

**为完成本次实验你翻阅的书籍与网站等**

[1] 林来兴. 空间控制技术[M]. 北京：中国宇航出版社，1992：25-42.

[2] 辛希孟. 信息技术与信息服务国际研讨会论文集：A集[C]. 北京：中国科学出版社，1999.

[3] 赵耀东. 新时代的工业工程师[M/OL]. 台北：天下文化出版社，1998 [1998-09-26]. http://www.ie.nthu.edu.tw/info/ie.newie.htm（Big5）.

[4] 谌颖. 空间交会控制理论与方法研究[D]. 哈尔滨：哈尔滨工业大学，1992：8-13.

[5] KANAMORI H. Shaking Without Quaking[J]. Science，1998，279（5359）：2063-2064.

[6] CHRISTINE M. Plant Physiology: Plant Biology in the Genome Era[J/OL]. Science，1998，281：331-332[1998-09-23]. http://www.sciencemag.org/cgi/ collection/anatmorp.