

**实验报告**

**实 验（三）**

题 目 Binary Bomb

二进制炸弹

专 业 计算机

学　　 号 1190201423

班　　 级 1903004

学 生 顾海耀

指 导 教 师 史先俊

实 验 地 点 G709

实 验 日 期 2021/4/21

**计算机科学与技术学院**

**目 录**

[第1章 实验基本信息 - 3 -](#_Toc69908635)

[1.1 实验目的 - 3 -](#_Toc69908636)

[1.2 实验环境与工具 - 3 -](#_Toc69908637)

[1.2.1 硬件环境 - 3 -](#_Toc69908638)

[1.2.2 软件环境 - 3 -](#_Toc69908639)

[1.2.3 开发工具 - 3 -](#_Toc69908640)

[1.3 实验预习 - 3 -](#_Toc69908641)

[第2章 实验环境建立 - 5 -](#_Toc69908642)

[2.1 Ubuntu下CodeBlocks反汇编（10分） - 5 -](#_Toc69908643)

[2.2 Ubuntu下EDB运行环境建立（10分） - 5 -](#_Toc69908644)

[第3章 各阶段炸弹破解与分析 - 8 -](#_Toc69908645)

[3.1 阶段1的破解与分析 - 8 -](#_Toc69908646)

[3.2 阶段2的破解与分析 - 8 -](#_Toc69908647)

[3.3 阶段3的破解与分析 - 8 -](#_Toc69908648)

[3.4 阶段4的破解与分析 - 8 -](#_Toc69908649)

[3.5 阶段5的破解与分析 - 8 -](#_Toc69908650)

[3.6 阶段6的破解与分析 - 8 -](#_Toc69908651)

[3.7 阶段7的破解与分析(隐藏阶段) - 8 -](#_Toc69908652)

[第4章 总结 - 10 -](#_Toc69908653)

[4.1 请总结本次实验的收获 - 10 -](#_Toc69908654)

[4.2 请给出对本次实验内容的建议 - 10 -](#_Toc69908655)

[参考文献 - 11 -](#_Toc69908656)

# 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。

## 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)、Og、O0、O1、O2、O3、Og、-m32/m64。再次查看生成的汇编语言与原来的区别。

注意O1之后缺省无栈帧，RBP为普通寄存器。用-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。

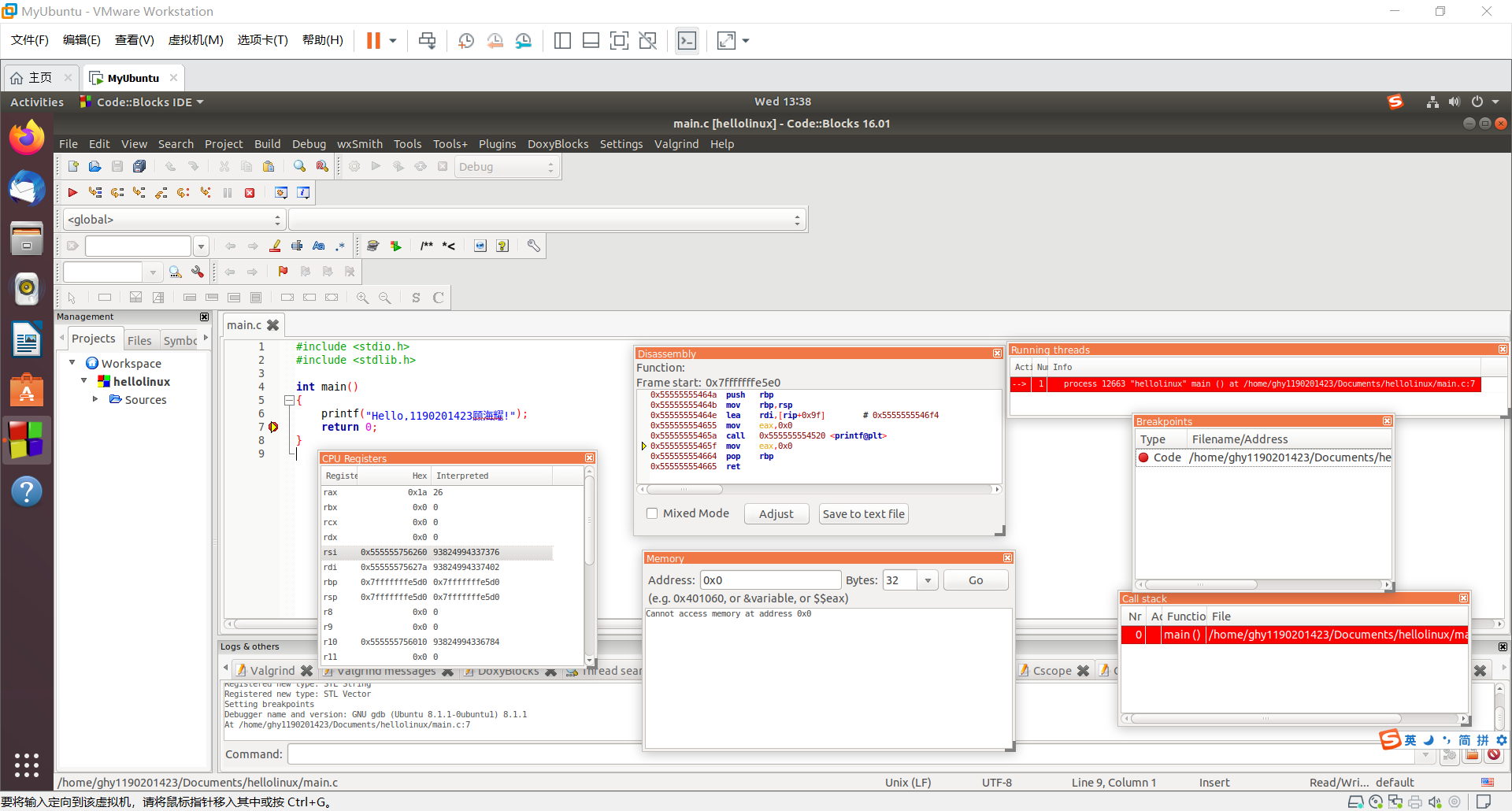
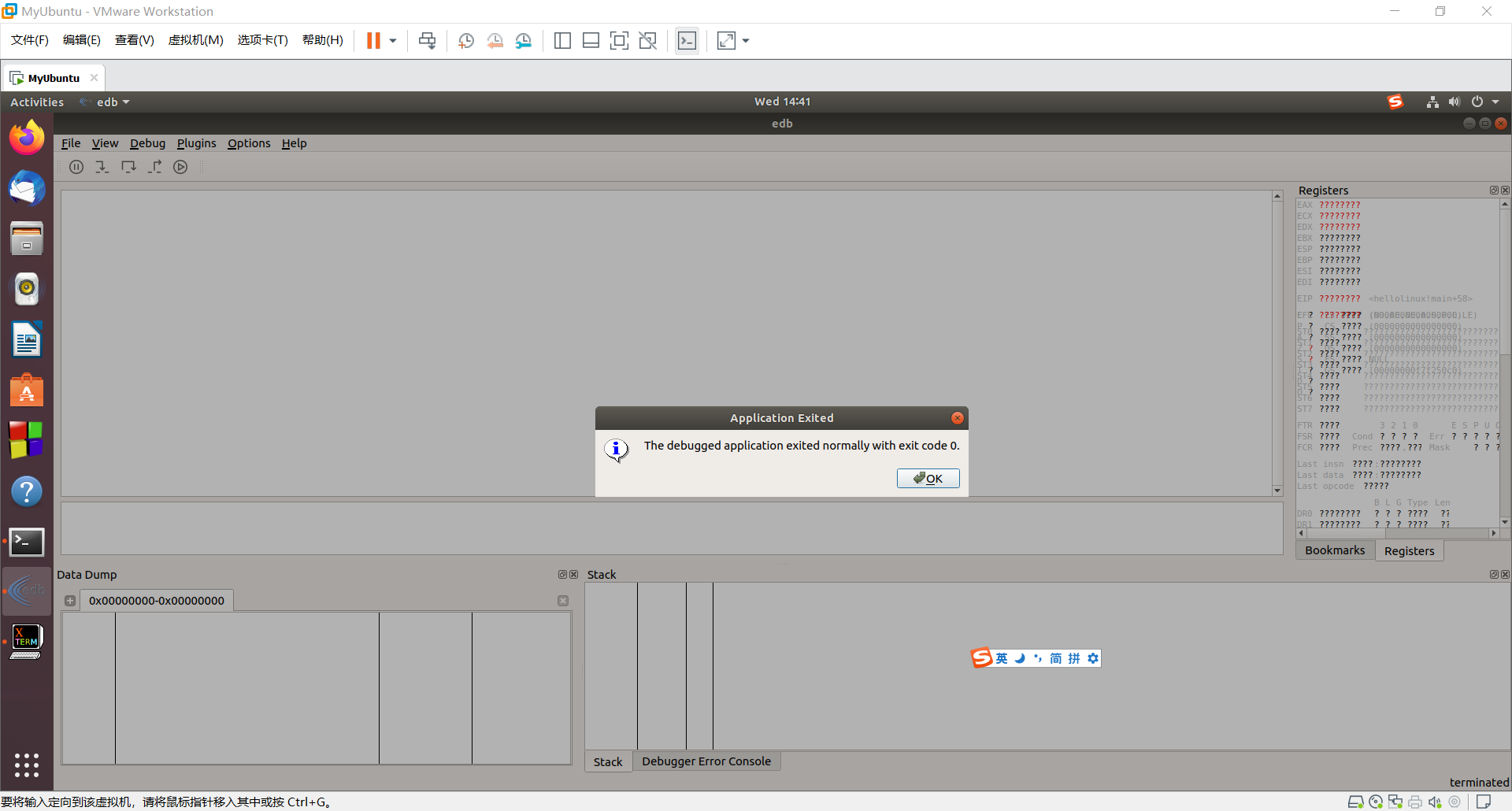


图2-1 Ubuntu下CodeBlocks反汇编截图

## 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





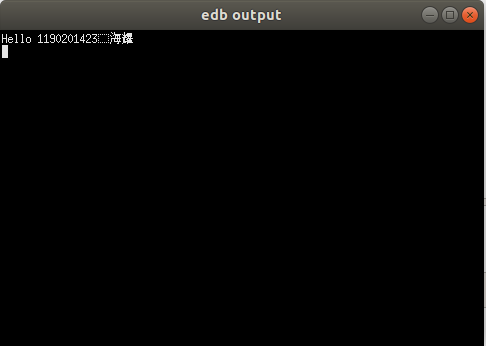


图2-2 Ubuntu下EDB截图

# 第3章 各阶段炸弹破解与分析

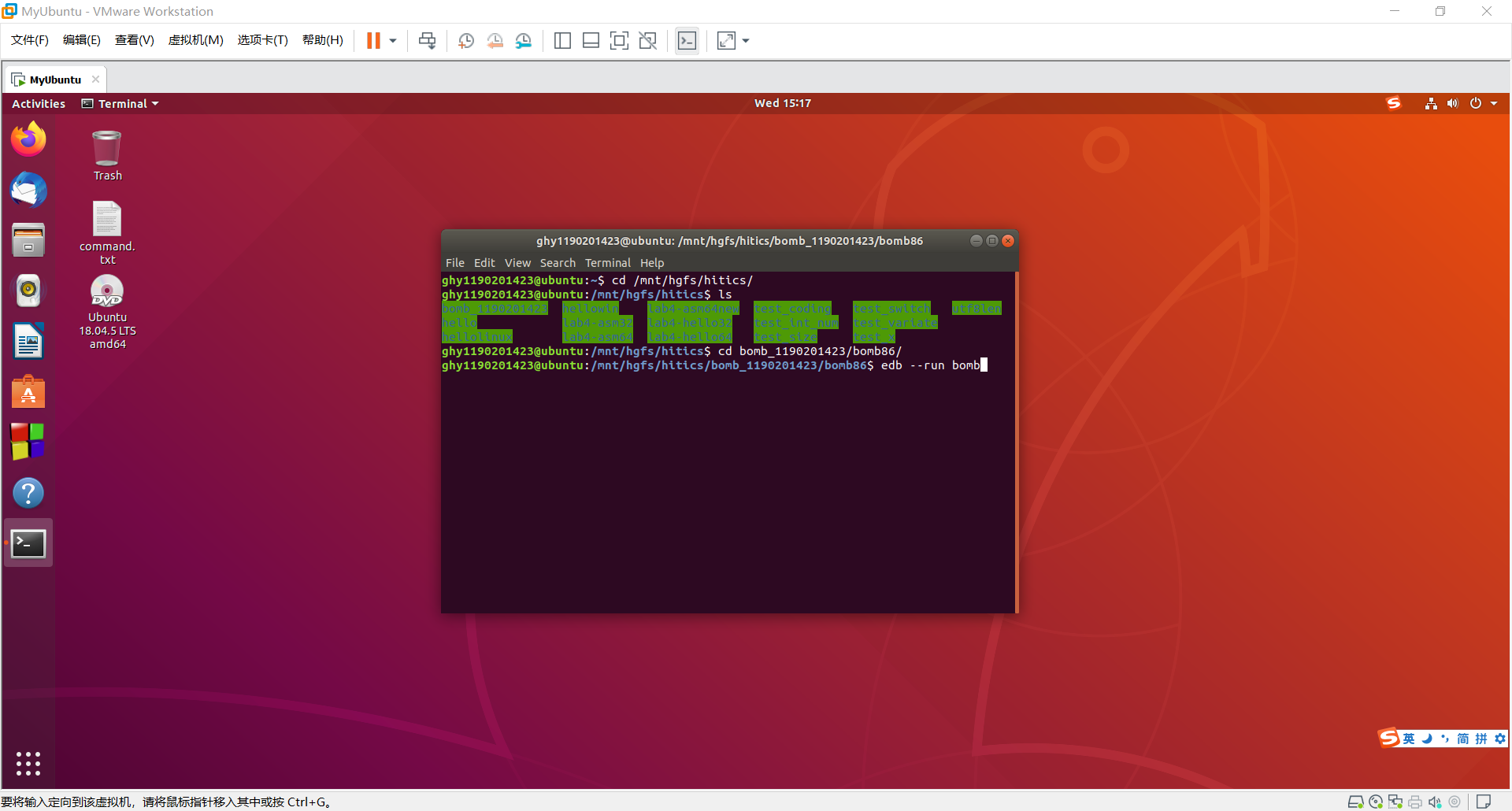
每阶段40分，密码20分，分析20分，总分不超过80分

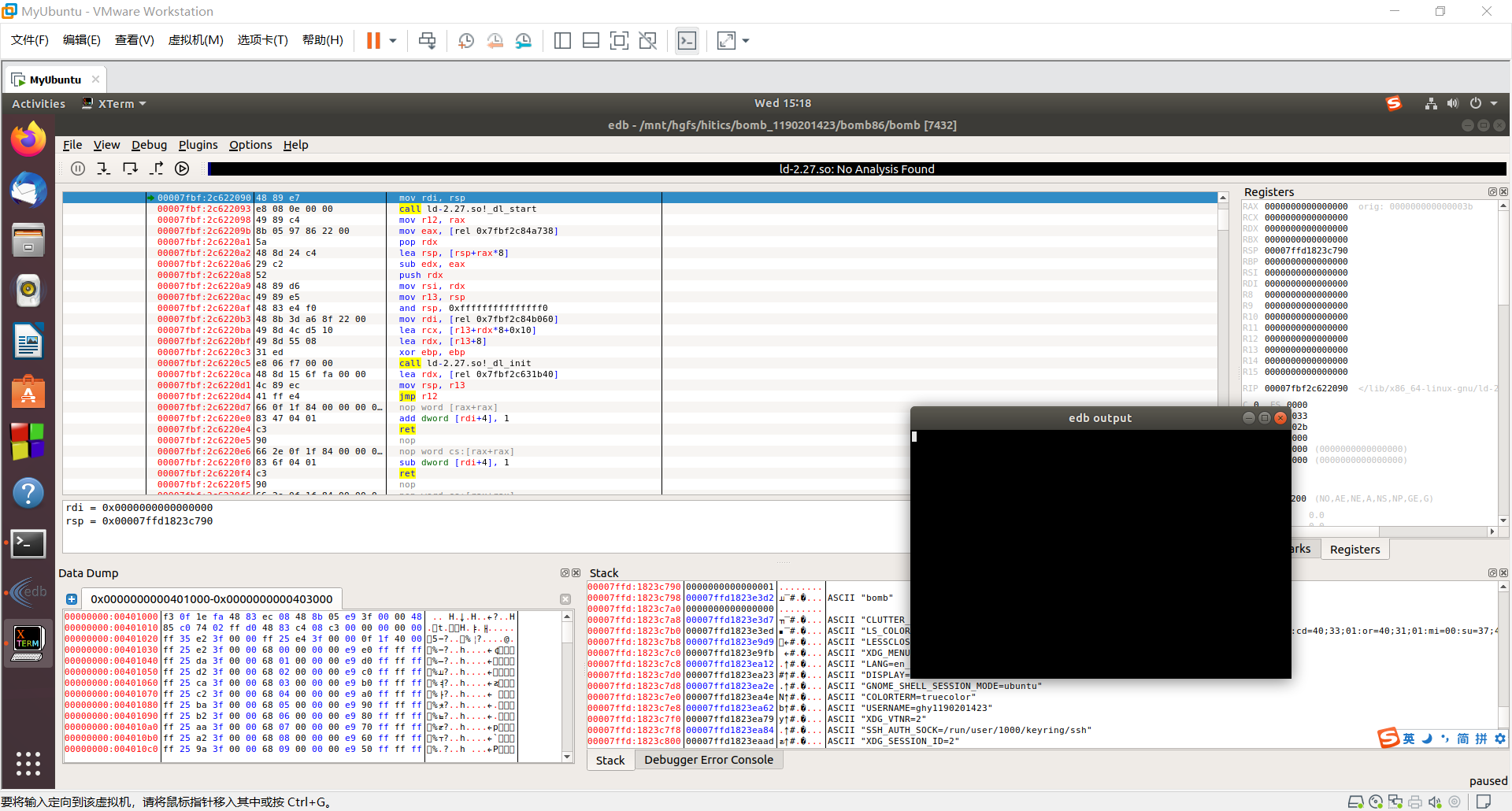
## 3.1 阶段1的破解与分析

密码如下：The moon unit will be divided into two divisions.

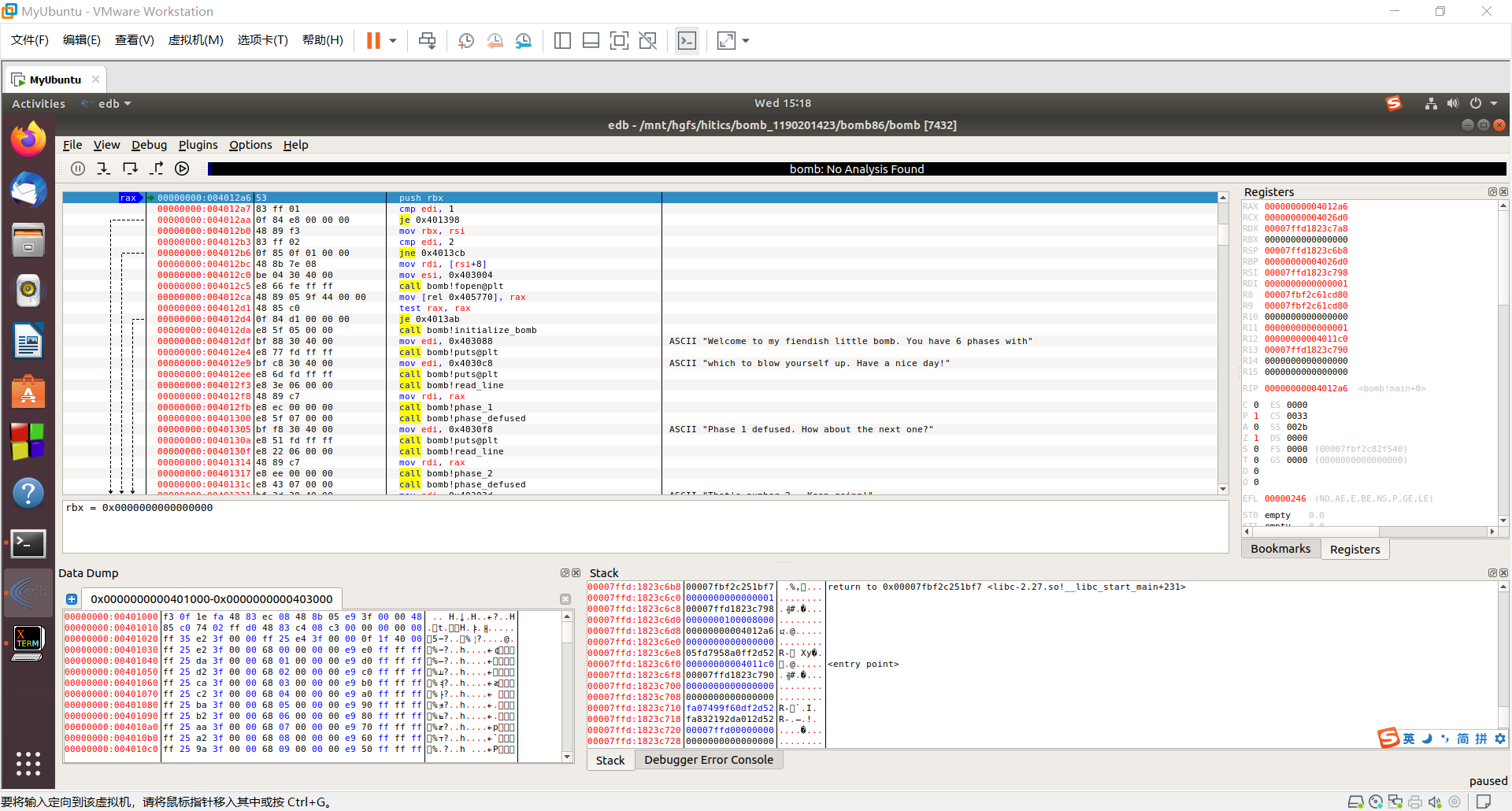
破解过程：

首先使用edb打开bomb

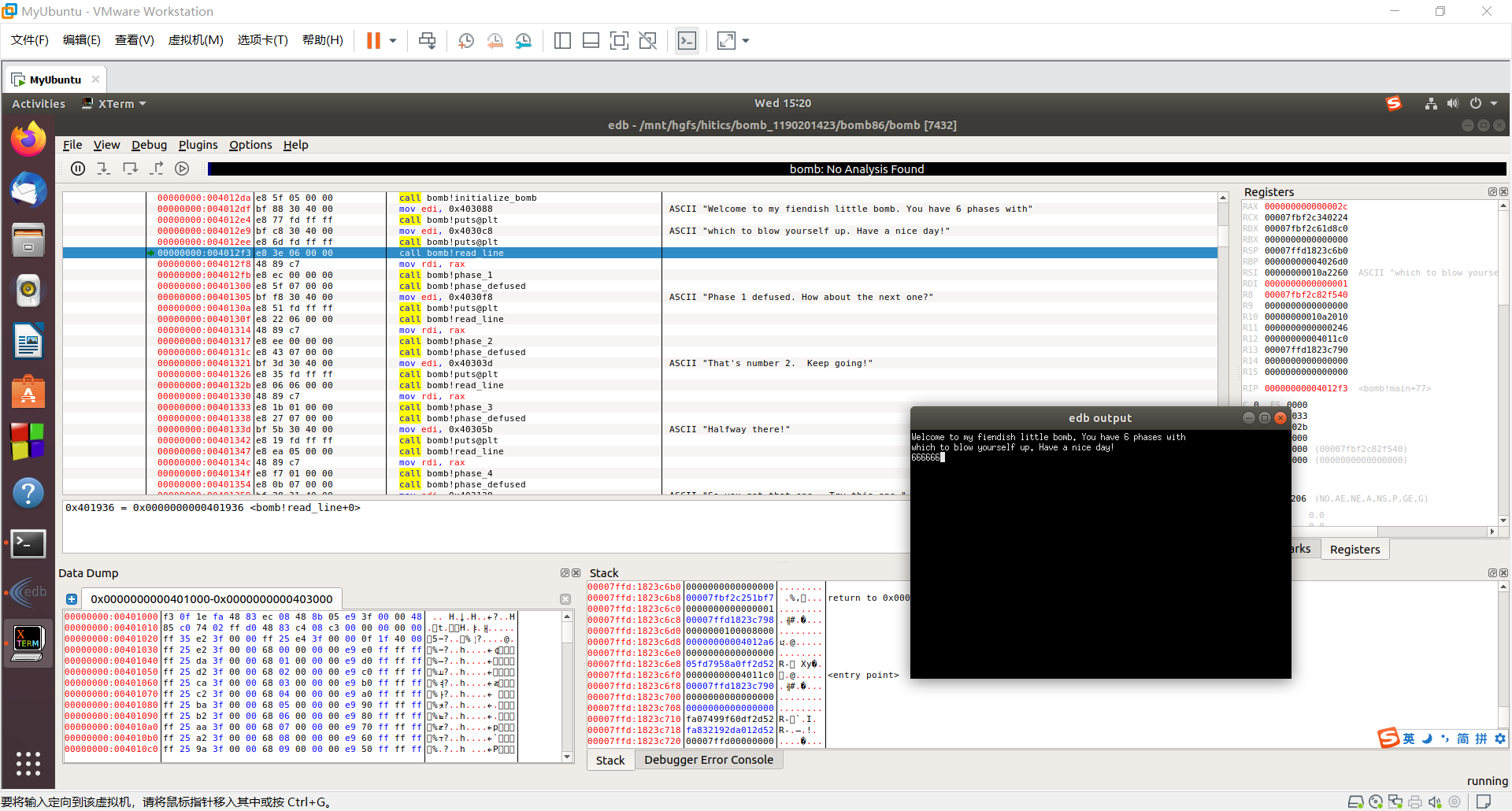




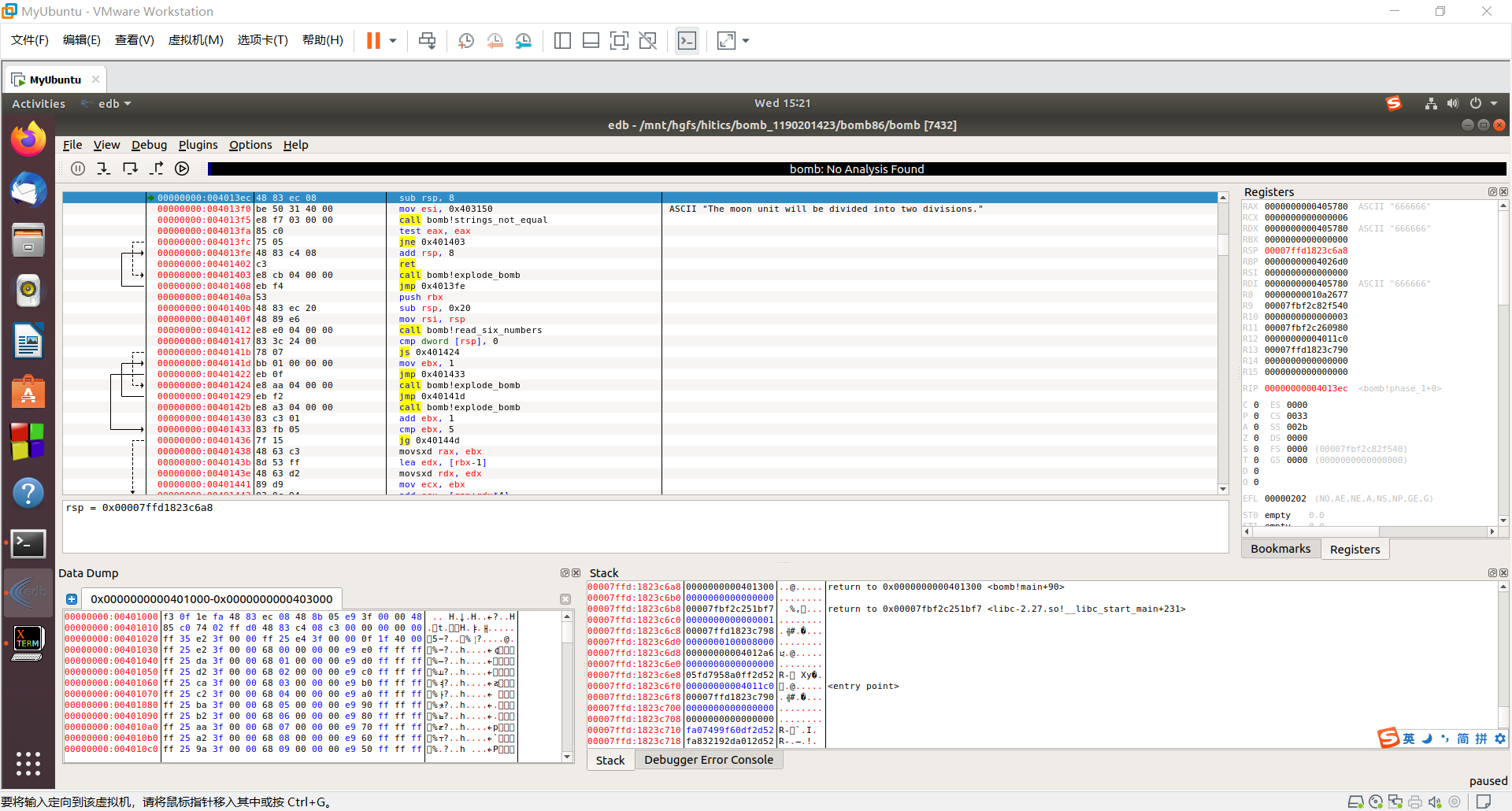
接下来，开始调试



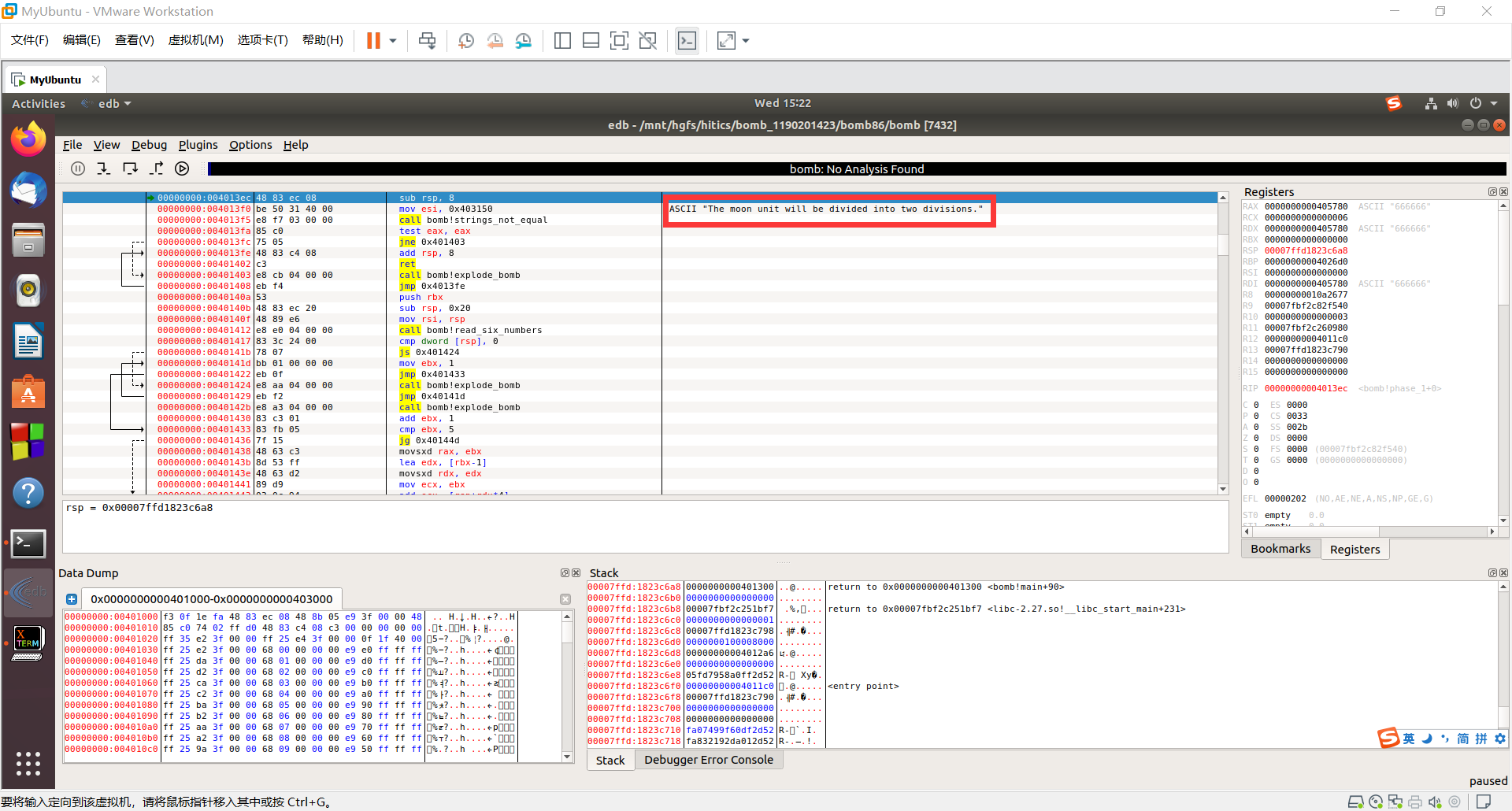
调试到输入phase1,随便输入一个密码



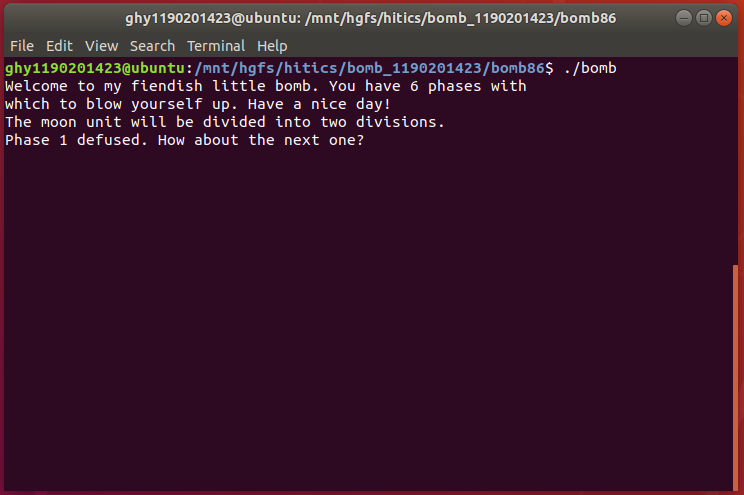
进入bomb!phase\_1



观察反汇编，即可看出答案是: The moon unit will be divided into two divisions.



最后，进行测试



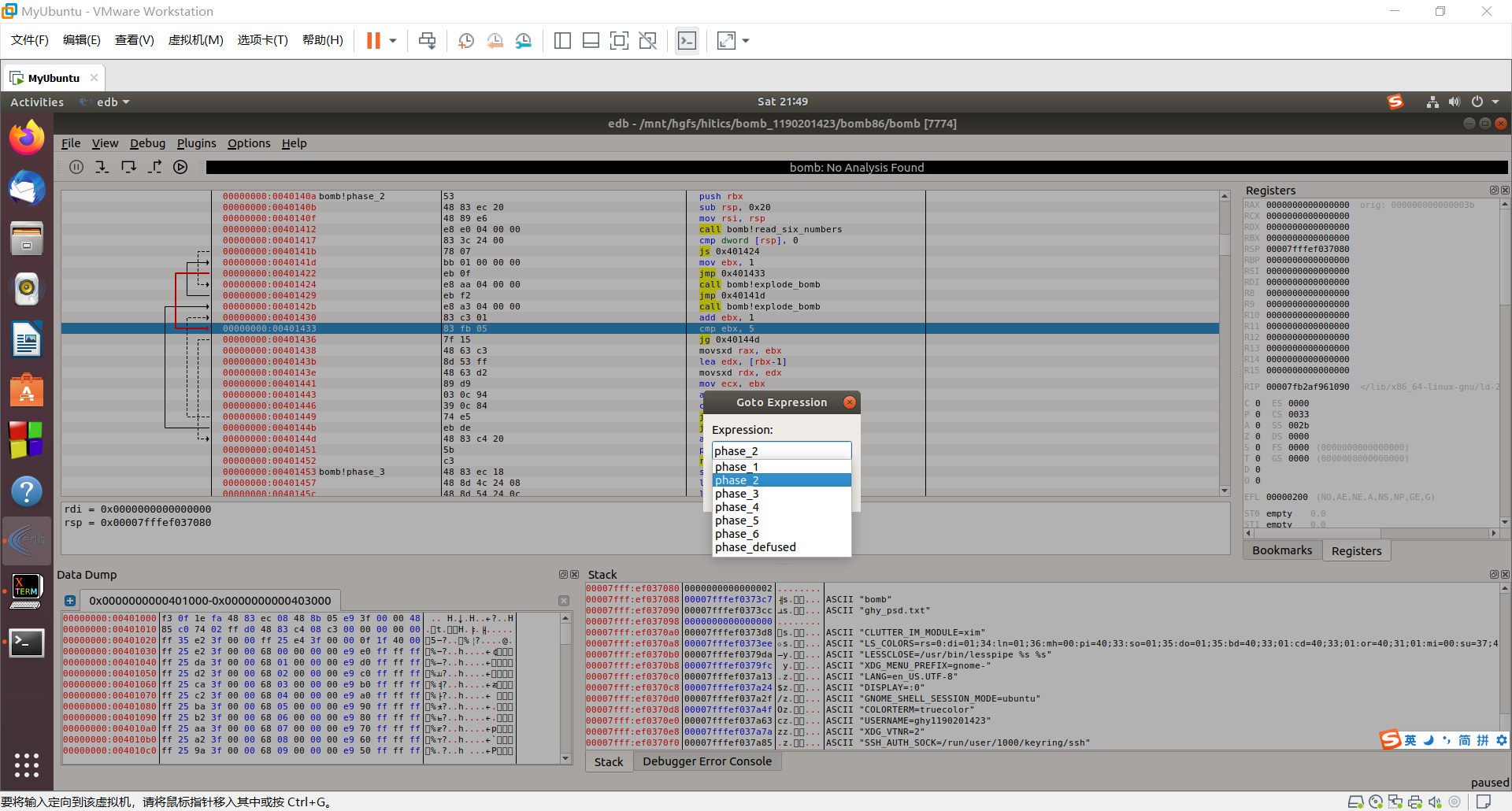
## 3.2 阶段2的破解与分析

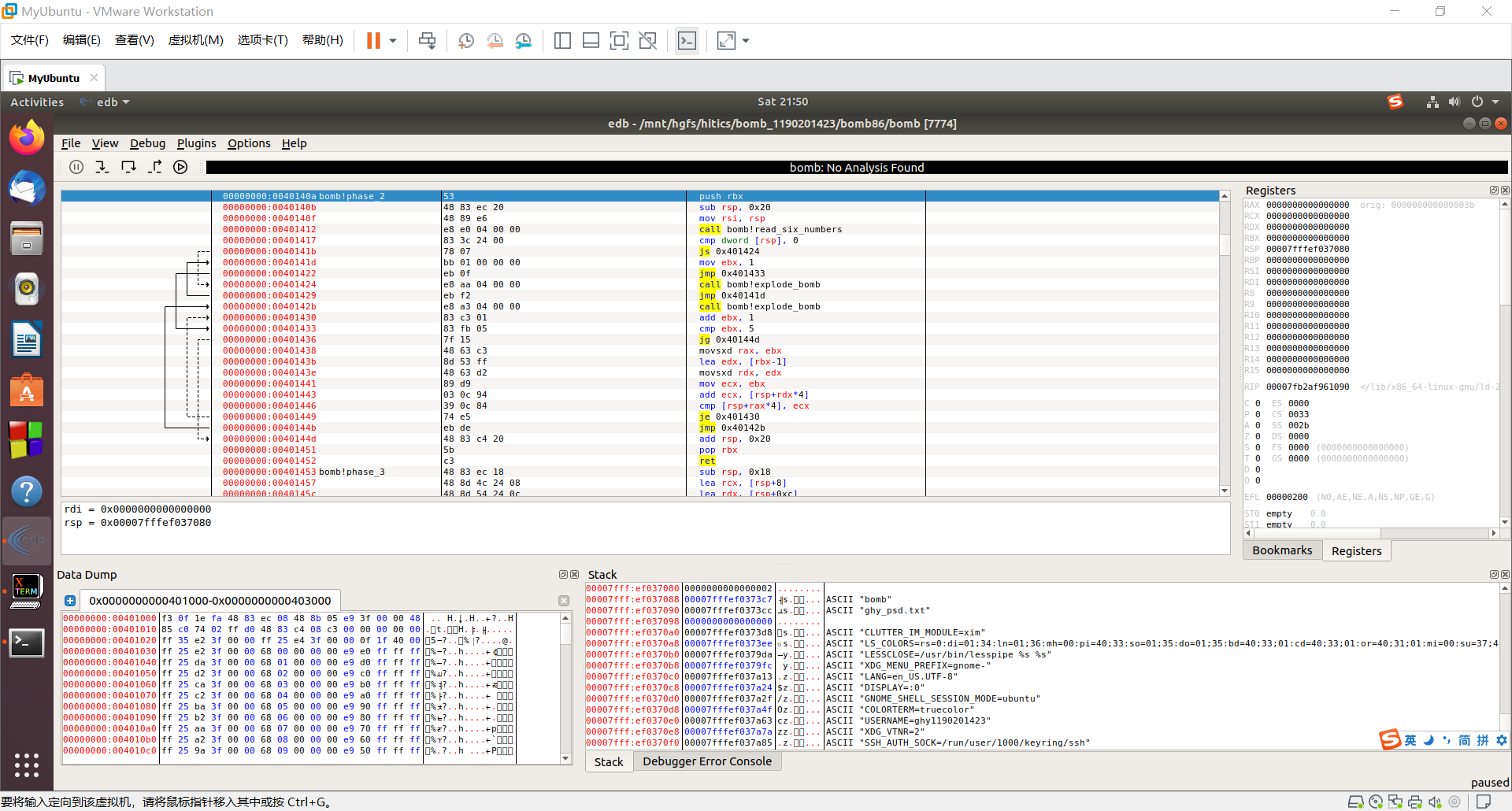
密码如下：0 1 3 6 10 15

[n n+1 n+2 n+3 n+4 n+5]即可

破解过程：

首先使用edb调用Goto Expression输入phase\_2





我们开始分析

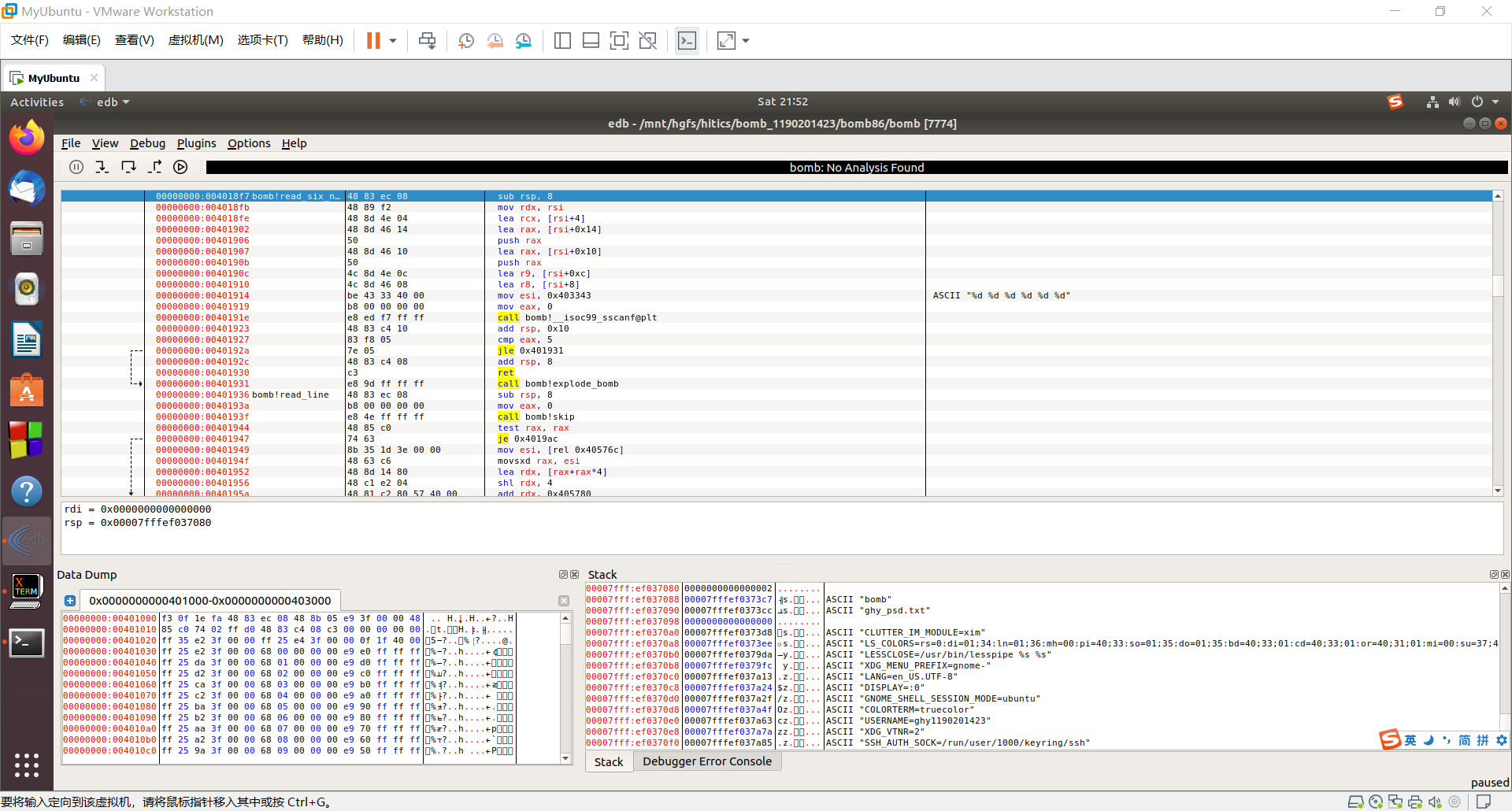
首先的

push rbx

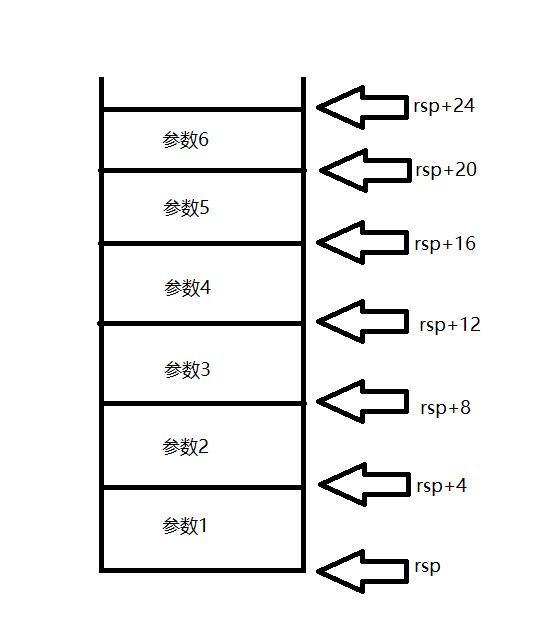
sub rsp,0x20

mov rsi,rsp

对栈进行初始化，接下来调用bomb!read\_six\_numbers,我们进入这个函数：

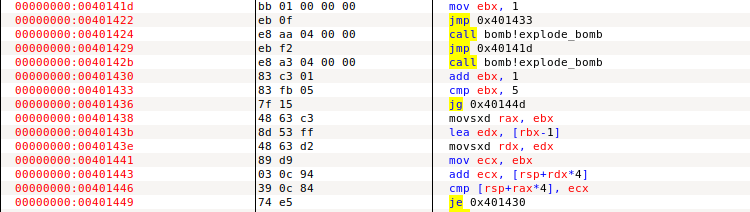


这个函数就是读取6个int类型的数字并用空格隔开，这应该就是第二关密码的格式，我们再接下来分析，因为我们现在读入了6个int类型的参数，所以我们的栈里应该是这样的：



先比较rsp和0，这说明读取的第一个值一定是大于0的。如果小于0，会进入一个函数：bomb!explode\_bomb，我们进入函数发现这个函数会引爆炸弹，

接下来进入最关键的部分：



首先给ebx赋值为1，然后edb与5进行比较，如果大于5，会直接恢复栈指针，如果小于5，先给rax赋值为edx的值并进行符号扩展，此时rax=1，然后把rbx-1的值赋值给edx，edx再符号扩展为rbx，这一步相当于rbx自减1，此时rbx=0,再把edx的值赋值给ecx，此时ecx为0，ecx再加上rsp+4\*rdx的值，即为初始数值，与第一个参数作比较，假设此初始值为X，那么第一次比较的就是X与第一个参数，暂记为X1，一定相等，再进行第二轮，此时edx=2，再经过上述步骤，ecx=1,rdx=edx=1,rax=2，此时比较的是X+0+1与X2……依次进行比较会发现规律：

X+0=X1

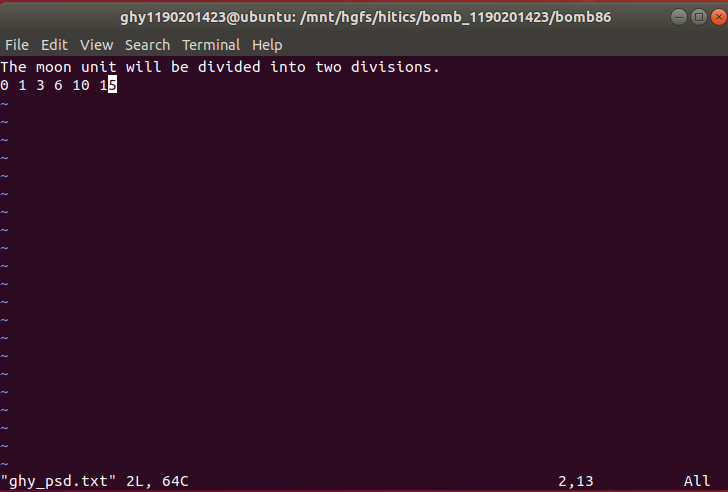
X+0+1=X2

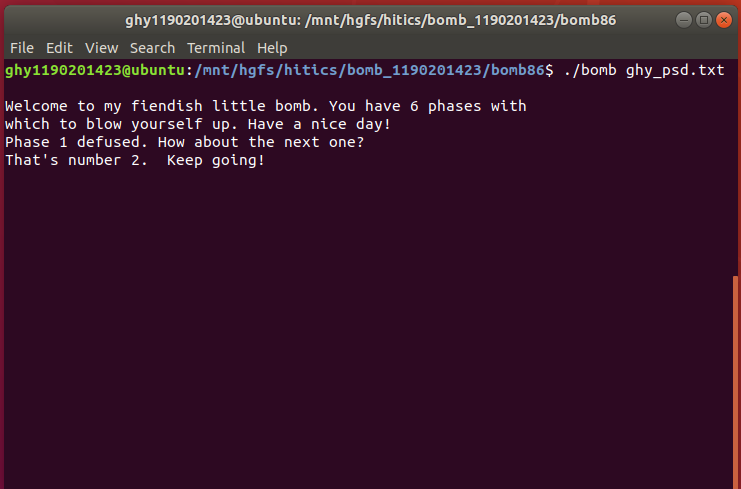
X+0+1+2=X3

X+0+1+2+3=X4

X+0+1+2+3+4=X5

X+0+1+2+3+4+5=X6



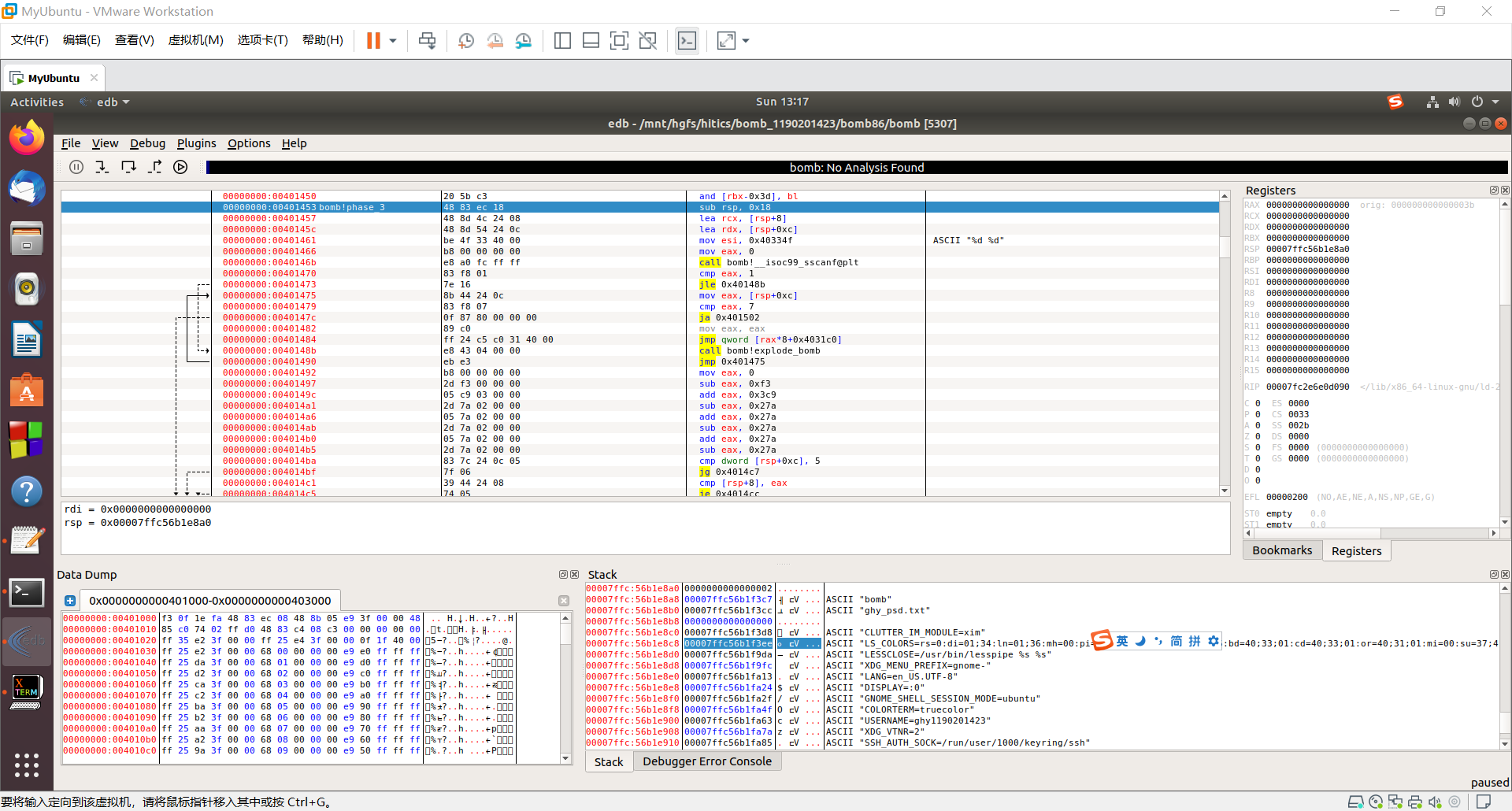


## 3.3 阶段3的破解与分析

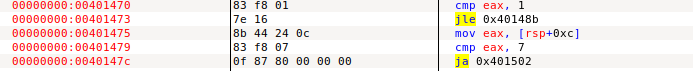
密码如下：3 -634

破解过程：

首先使用edb调用Goto Expression输入phase\_3



首先调用了bomb!\_\_isoc99\_sscanf@plt，并且观察可得这个函数读取两个int类型的整数，这应该就是密码的格式，看下面这些代码，可以发现第一个参数的值应该的在[0,7]这个范围内：

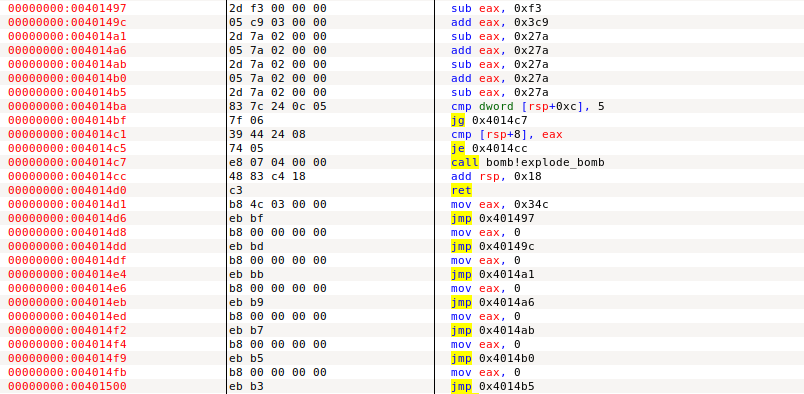


又看到下面这两行代码：



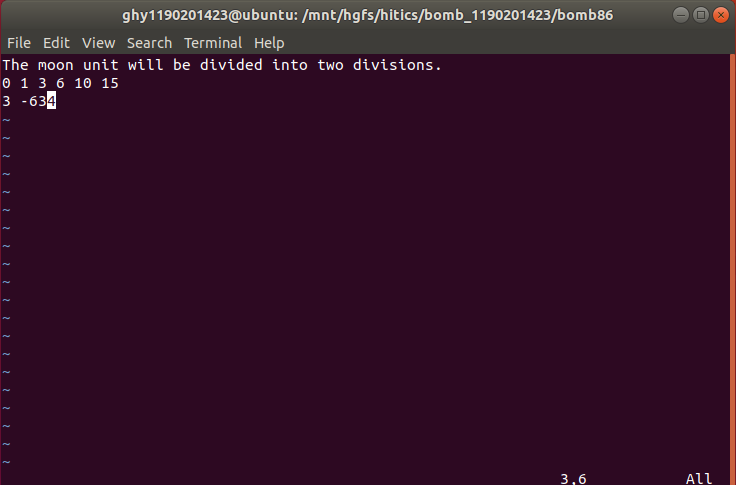
可以将第一个参数的范围确定在[0,5]。

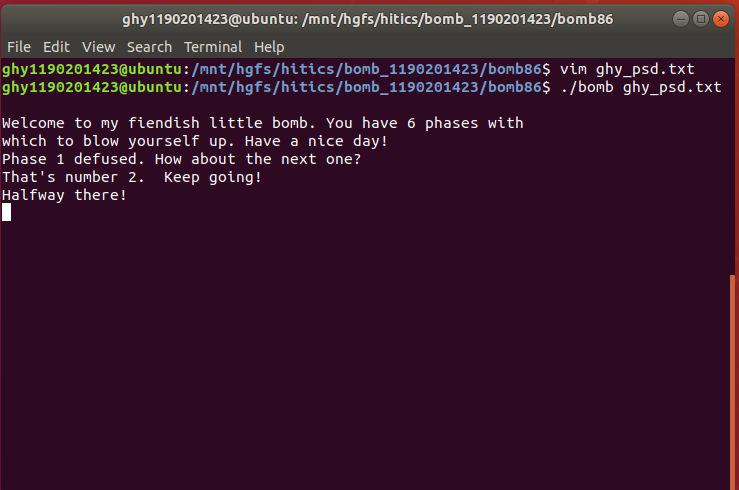
看下面这几行代码，可以发现这是一个switch分支结构：



于是选取3，计算可得第二个参数为-634

运行可得：



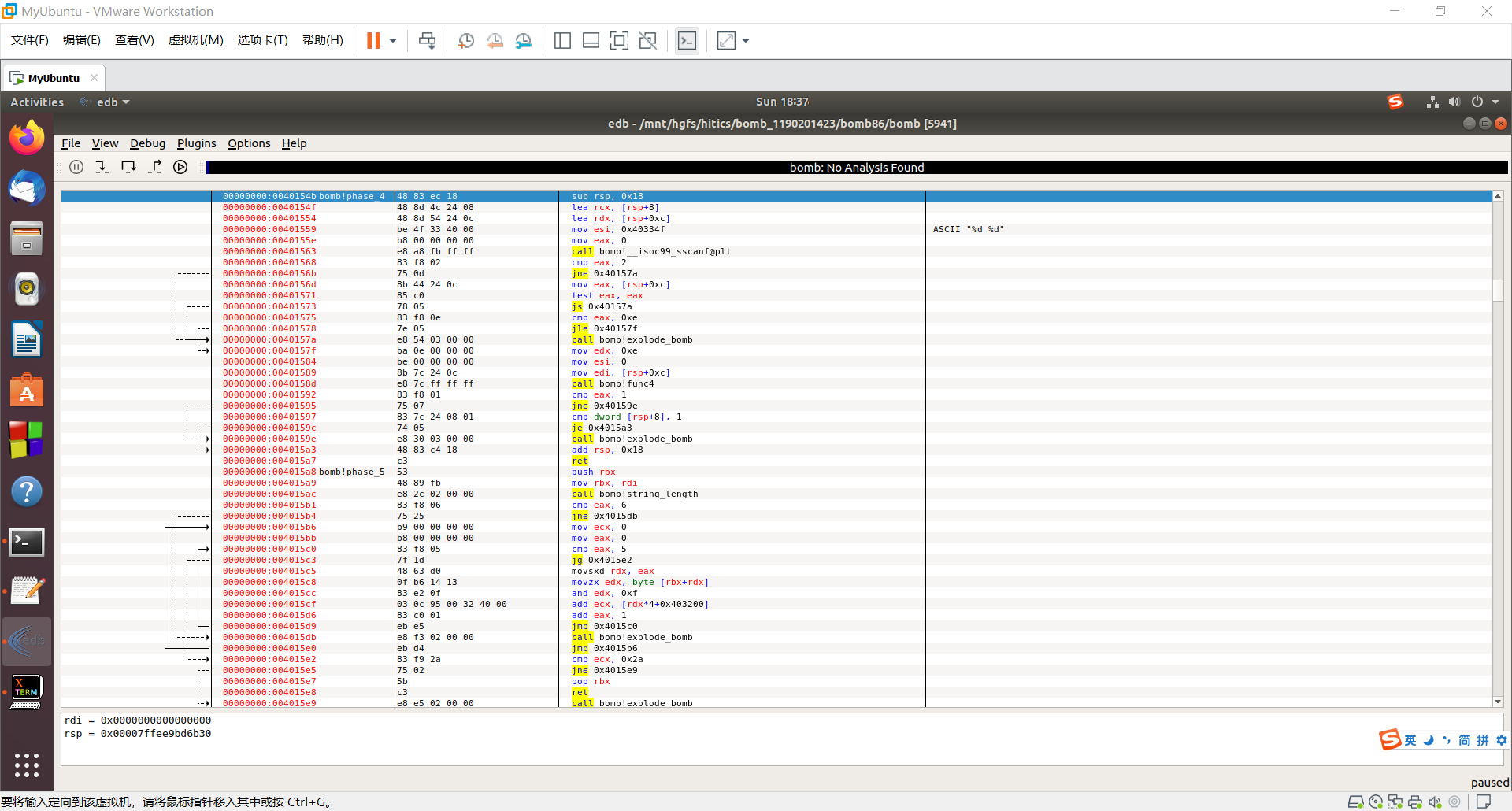


## 3.4 阶段4的破解与分析

密码如下：8 1

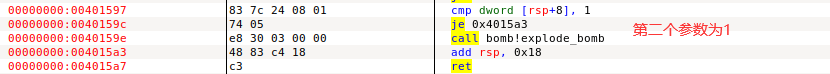
破解过程：

首先使用edb调用Goto Expression输入phase\_4

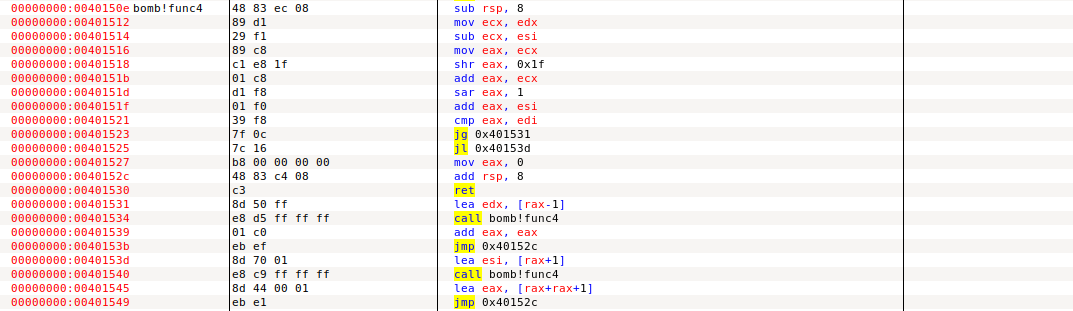


看调用的bomb!\_\_isoc99\_sscanf@plt函数可以看出密码的格式，是两个空格分隔的int类型常数。

往下看可以看出第一个整数的范围是0~14，由bomb!fun4函数可以求出第一个参数，而第二个参数为1可以很显然的看出。



于是看bomb!fun4函数



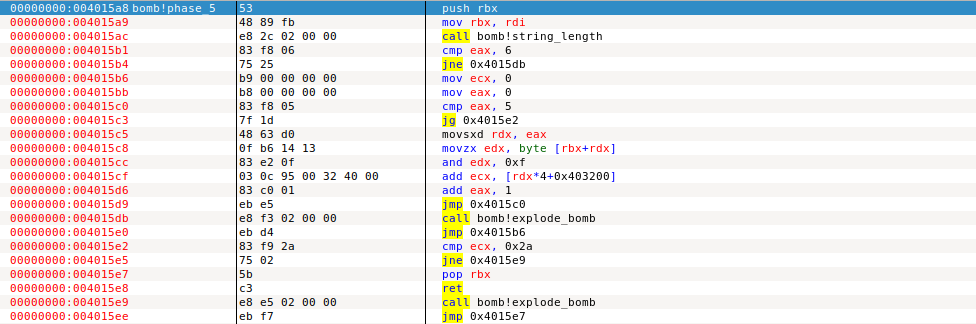
接下来使用穷举法，可以得出答案为8 1。

## 3.5 阶段5的破解与分析

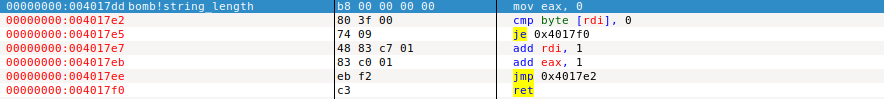
密码如下：444000

破解过程：

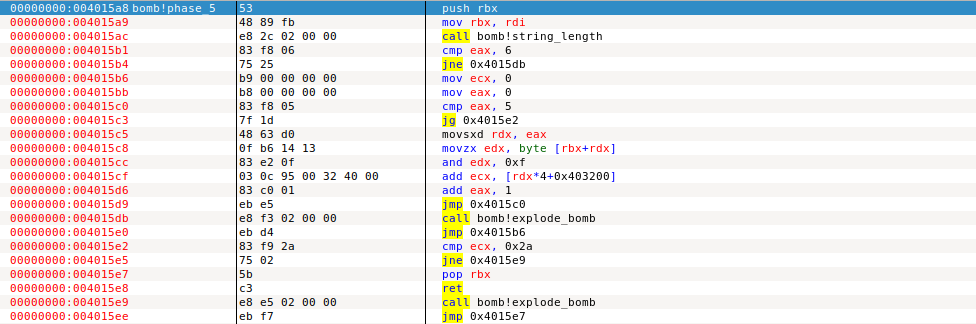
首先使用edb调用Goto Expression输入phase\_5



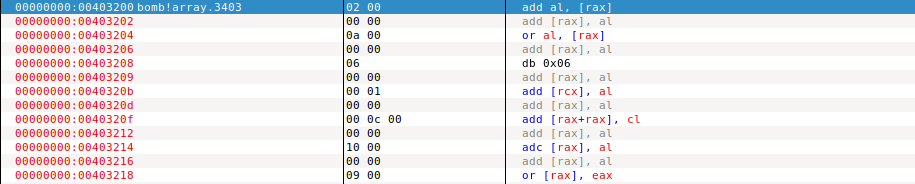
进入bomb!string\_length函数，可以看到是在判断密码的位数：



然后回到phase\_5,可以看到eax与6进行比较，所以密码应该就是一个6位字符串。



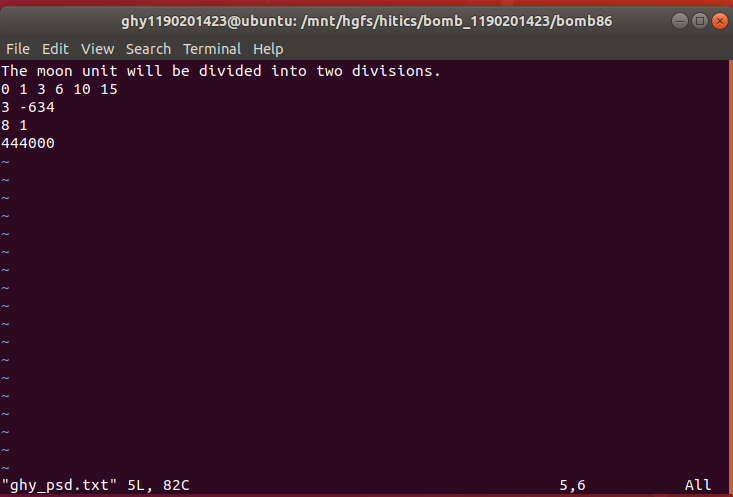
接下来分析，进行一个循环，每一次循环，都会让ecx加上一个值，而这个值是rdx\*4+0x403200得出来的，最后，ecx与0x2c（42）进行比较，如果相等则为答案，于是进入0x403200：

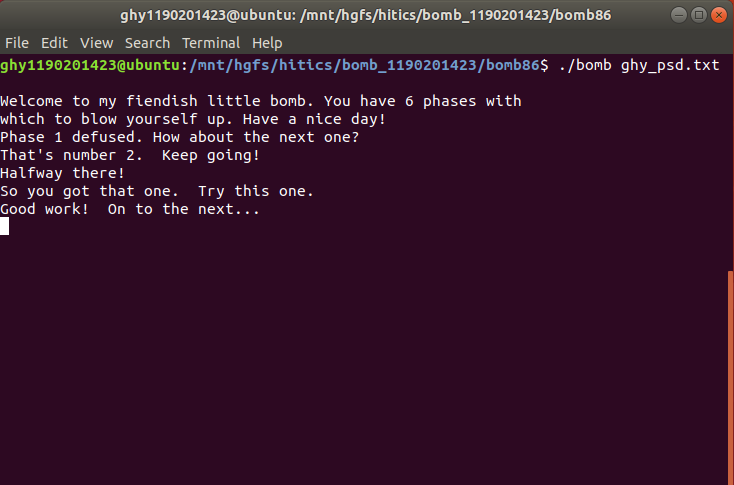


通过观察可以发现如下规律：

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| rdx10 | rdx\*410 | rdx\*416 | 对应的值16 | 对应的值10 |
| 0 | 0 | 0x0 | 0x02 | 2 |
| 1 | 4 | 0x4 | 0x0a | 10 |
| 2 | 8 | 0x8 | 0x06 | 6 |
| 3 | 12 | 0xc | 0x01 | 1 |
| 4 | 16 | 0x10 | 0x0c | 12 |

找到一种组合方式，可以让6个数的和为42，于是可以可以选择12+12+12+2+2+2=42的组合方式，即为444000



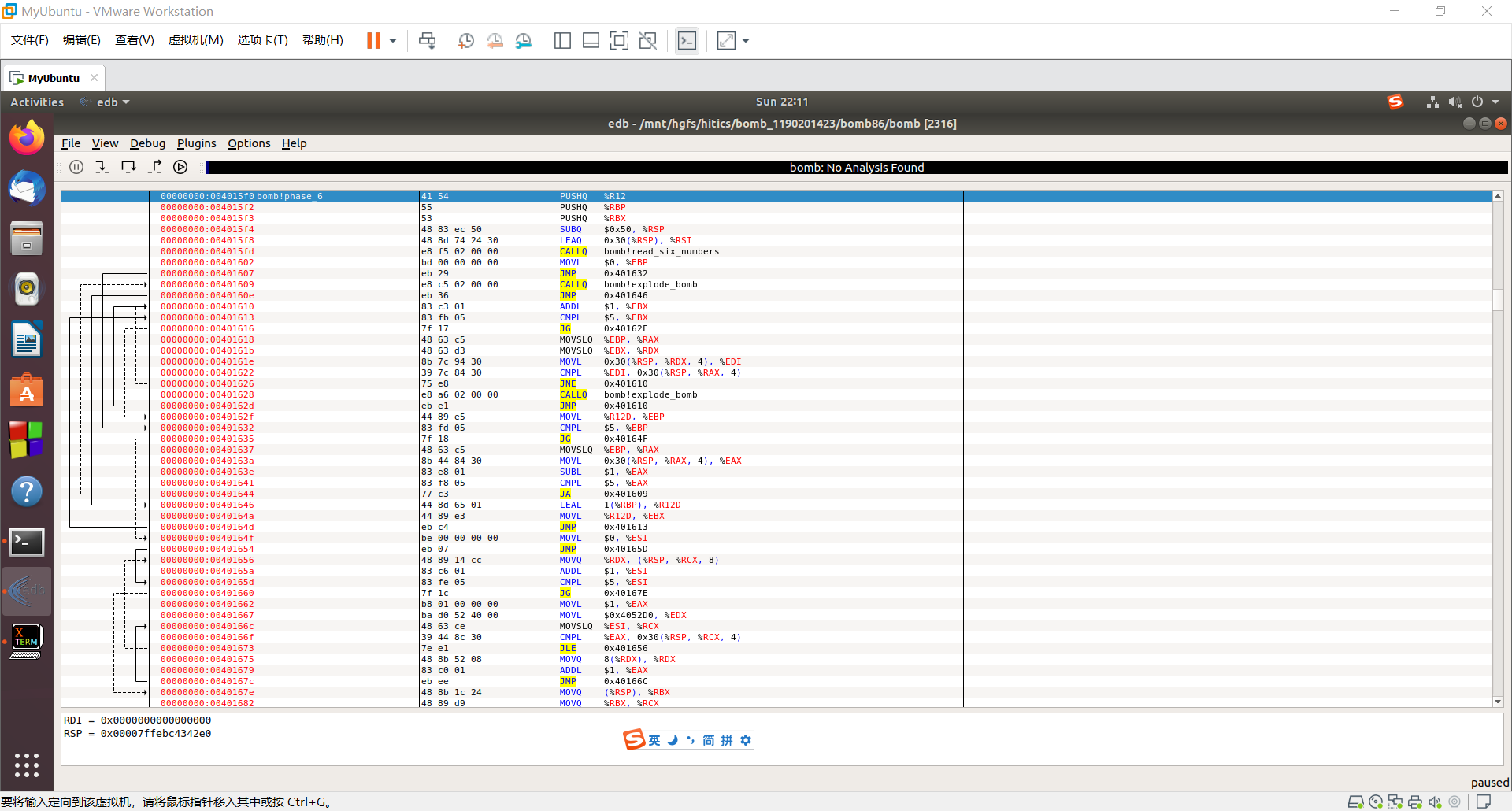


## 3.6 阶段6的破解与分析

密码如下：5 2 1 4 3 6

破解过程：

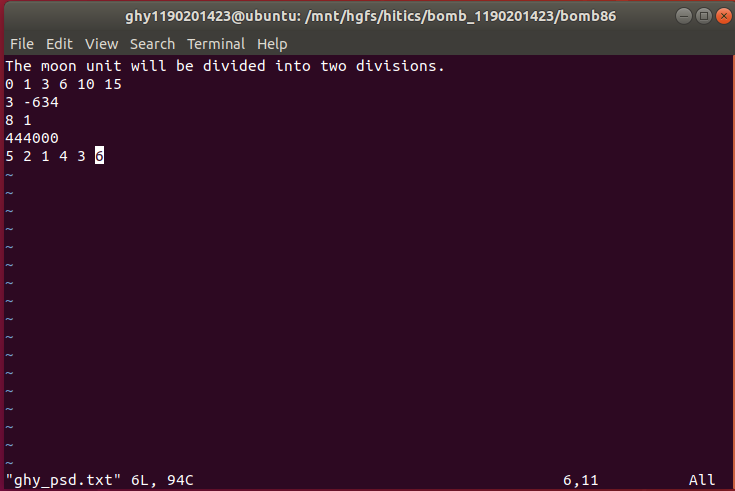
首先使用edb调用Goto Expression输入phase\_6

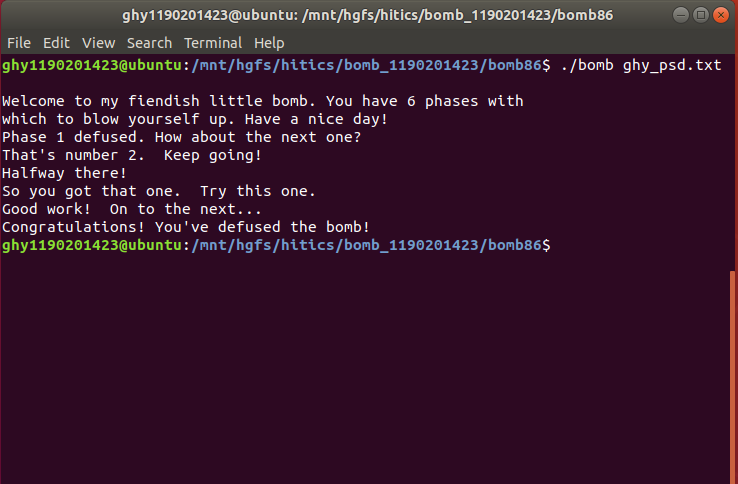


观察代码可以发现调用了bomb!read\_six\_numbers函数，说明了密码的格式为6个int类型整数，观察代码后，调用Goto Expression依次输入node\_1~node\_6,观察这6个节点所存储的值，可以得到下面的表格：

|  |  |  |
| --- | --- | --- |
| 节点名称 | 节点存储的值(节点所在位置后1字节) | 节点标号(节点所在位置后2字节) |
| node1 | 0000 0266 | 0000 0001 |
| node2 | 0000 02c9 | 0000 0002 |
| node3 | 0000 0235 | 0000 0003 |
| node4 | 0000 0249 | 0000 0004 |
| node4 | 0000 02da | 0000 0005 |
| node5 | 0000 00ff | 0000 0006 |

再观察发现降序排列，于是便可得到：5 2 1 4 3 6:





## 3.7 阶段7的破解与分析(隐藏阶段)

密码如下：

破解过程：

# 第4章 总结

## 4.1 请总结本次实验的收获

本次实验学会了 edb 的调试方法，学会了分析反汇编代码来寻找其中储存的某些关键信息进而破解密码，对反汇编语句运用更加熟练，明白了汇编语言中各个函数之间的调用关系，对堆栈有了更深的理解，学会了对教材知识熟练运用。

## 4.2 请给出对本次实验内容的建议

建议老师可以多讲一些其他的edb指令的使用，感觉对edb这个工具的使用还不是很熟悉。

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.