

**实验报告**



**题目： 拆解二进制炸弹（高阶）**

**班 级： 2023211301**

**学 号： 2023210896**

**姓 名： 王书翰**

**学 院： 计算机学院（国家示范性软件学院）**

**2024年 11 月12日**

一、实验目的  
1.理解C语言程序的机器级表示。  
2.初步掌握GDB调试器的用法。  
3.阅读C编译器生成的x86-64机器代码，理解不同控制结构生成的基本指令模式，过程的实现。

1. 实验环境
2. Windows PowerShell（10.120.11.12）
3. Linux
4. Objdump命令反汇编
5. GDB调试工具

5. Vim 8.2.2121

三、实验内容

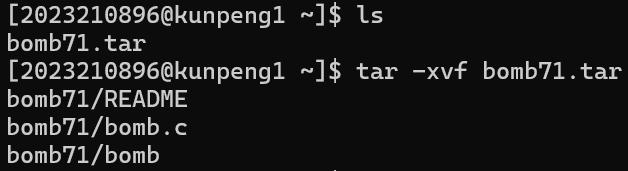
登录bupt1服务器，在home目录下可以找到Evil博士专门为你量身定制的一个bomb，当运行时，它会要求你输入一个字符串，如果正确，则进入下一关，继续要求你输入下一个字符串；否则，炸弹就会爆炸，输出一行提示信息并向计分服务器提交扣分信息。因此，本实验要求你必须通过反汇编和逆向工程对bomb执行文件进行分析，找到正确的字符串来解除这个的炸弹。

本实验通过要求使用课程所学知识拆除一个“binary bombs”来增强对程序的机器级表示、汇编语言、调试器和逆向工程等方面原理与技能的掌握。 “binary bombs”是一个Linux可执行程序，包含了5个阶段（或关卡）。炸弹运行的每个阶段要求你输入一个特定字符串，你的输入符合程序预期的输入，该阶段的炸弹就被拆除引信；否则炸弹“爆炸”，打印输出 “BOOM!!!”。炸弹的每个阶段考察了机器级程序语言的一个不同方面，难度逐级递增。

为完成二进制炸弹拆除任务，需要使用gdb调试器和objdump来反汇编bomb文件，可以单步跟踪调试每一阶段的机器代码，也可以阅读反汇编代码，从中理解每一汇编语言代码的行为或作用，进而设法推断拆除炸弹所需的目标字符串。实验2的具体内容见实验2说明。

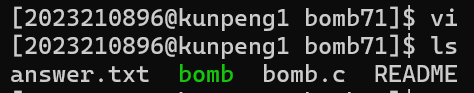
四、实验步骤及实验分析

0、准备阶段

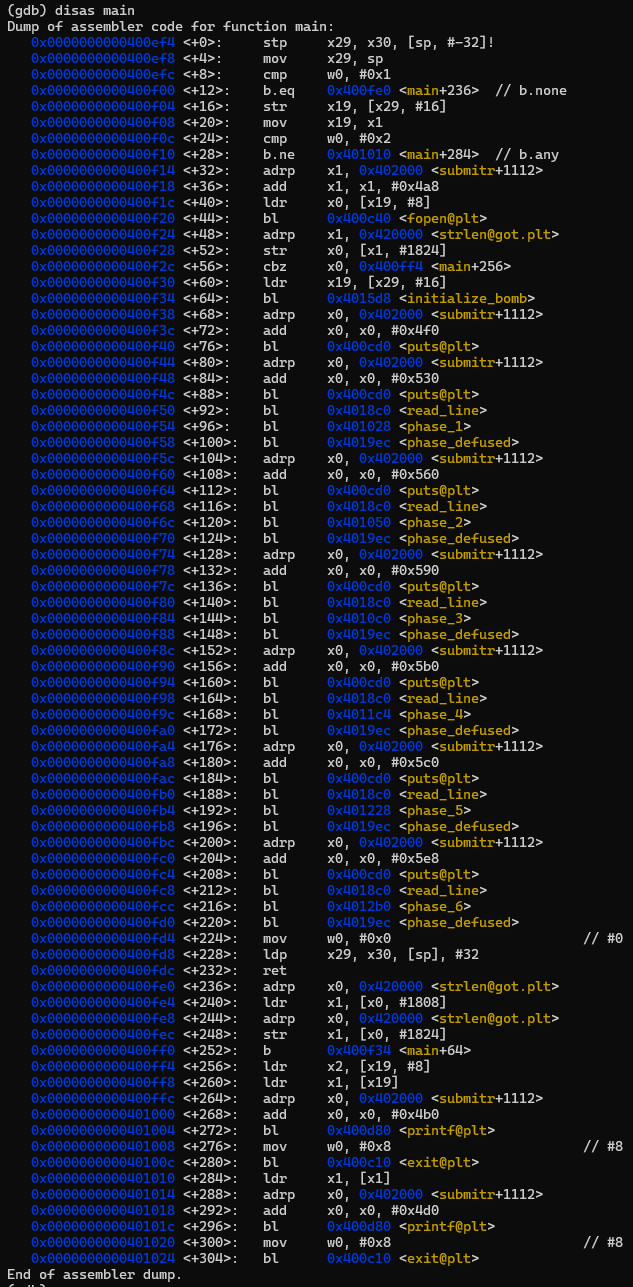
首先使用ls指令，查看压缩文件使用tar -xvf bomb71.tar 指令解压文件，得到3个文件

使用cd bomb71指令将目录更改为bomb71文件夹

用vi 指令启动vim编辑器，新建answer.txt文件用来储存答案



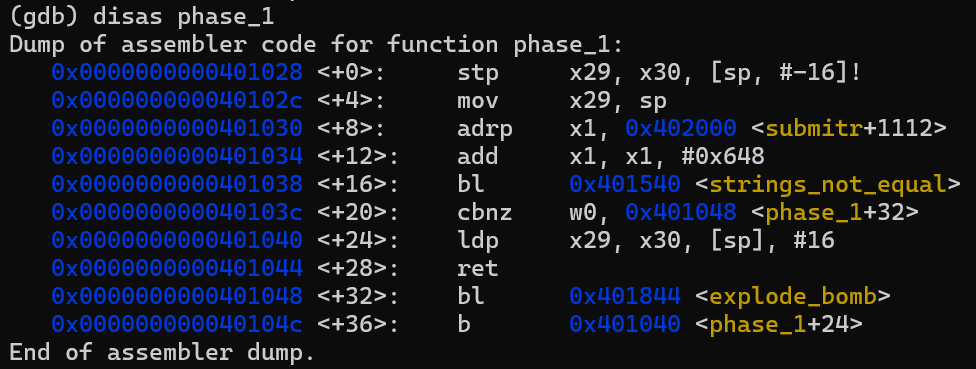
使用gdb 指令启动gdb调试器使用file bomb加载文件bomb，使用disas main指令查看main函数的反汇编语言，大概了解执行过程



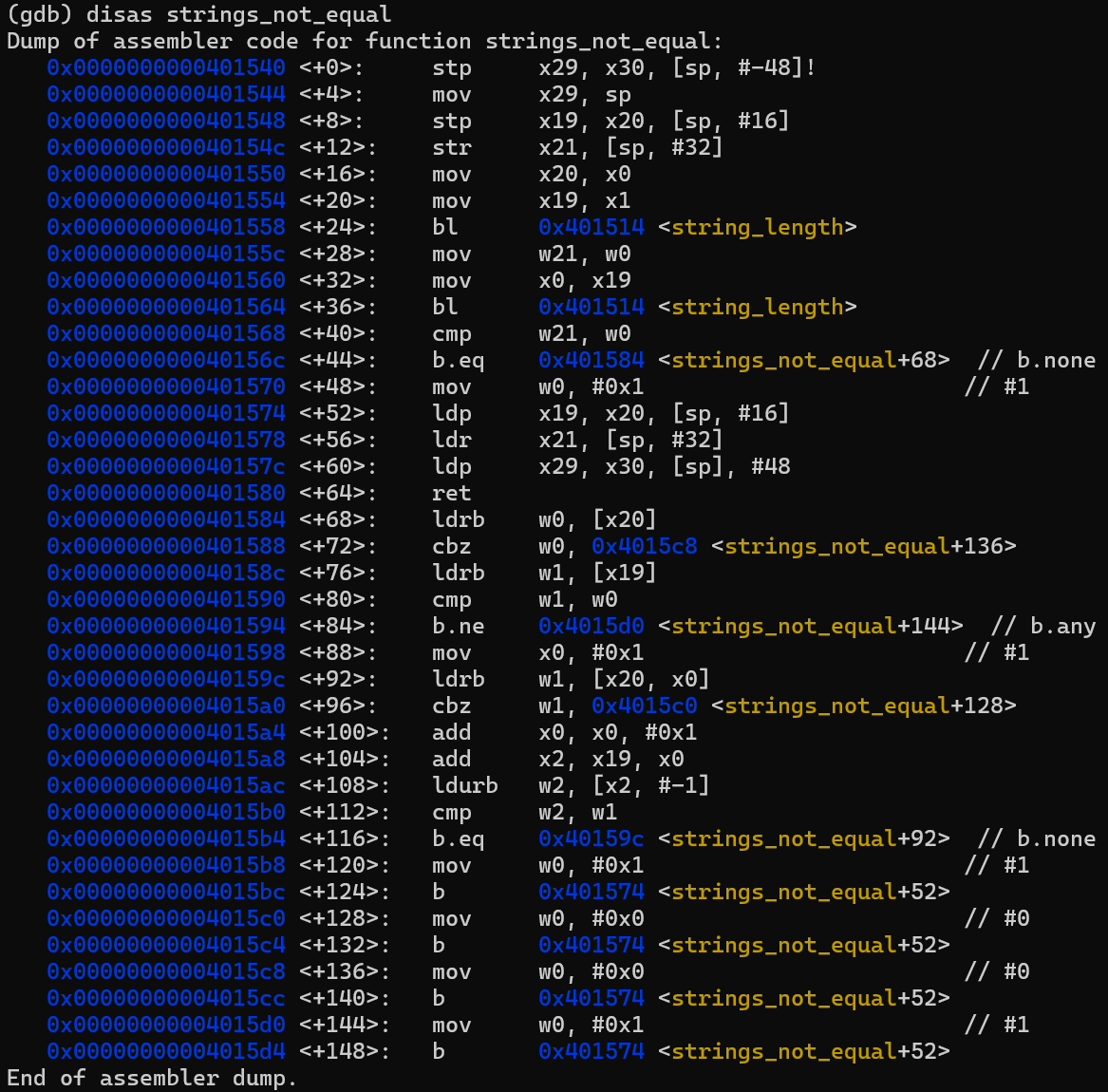
由截图可知，拆炸弹的主要流程同普通炸弹相似，都为为先发现函数先通过调用readline函数读取输入的内容，调用phase\_1等6个函数，若成功拆除炸弹则继续执行调用phase\_defused函数并输出若干内容，重复执行6次。故接下来需要依次查看phase\_1等6个函数的反汇编代码与phase\_defused()的反汇编代码。

1、第一阶段

使用disas phase\_1指令查看其反汇编代码



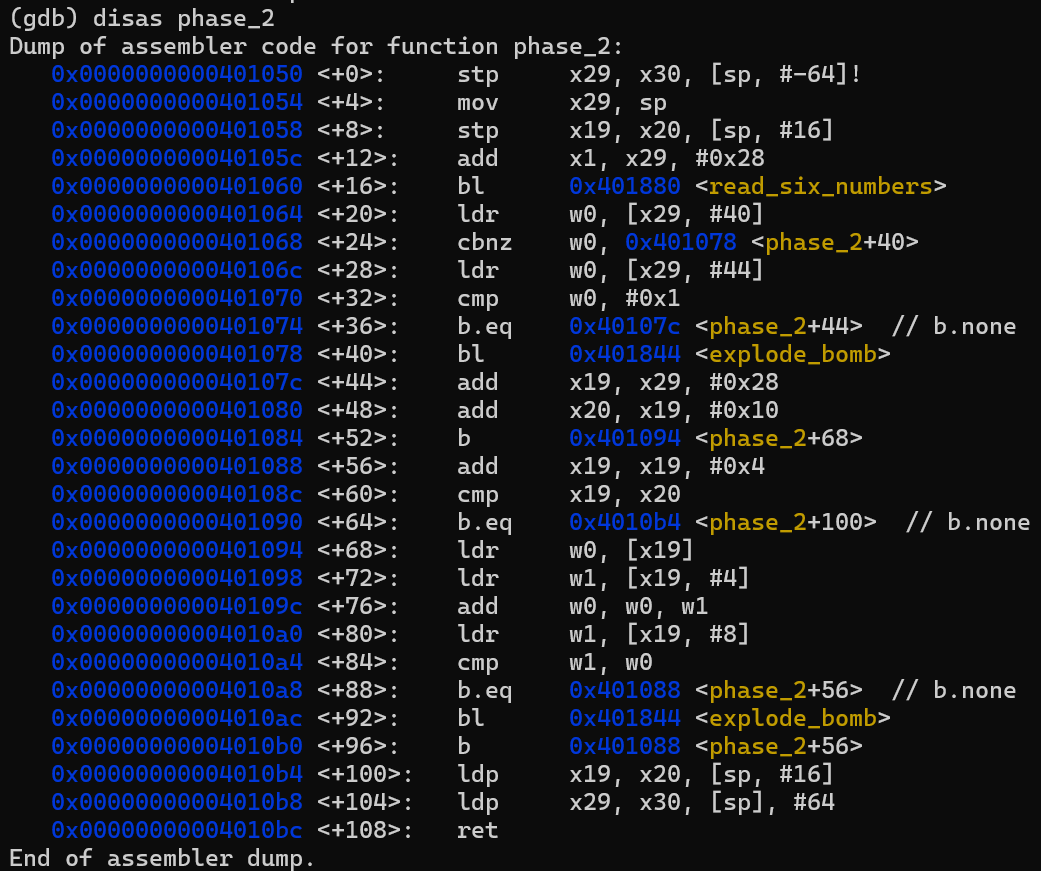
分析反汇编语言得phase\_1函数先初始化x1寄存器的值为地址0x402000储存的值，紧接着将其值再加上0x648，之后将x1作为参数调用了名为strings\_not\_equal函数，接着若该函数的返回值等于0则跳过炸弹，若不等于0则引爆炸弹。根据函数名称与调用的参数推测该函数的作用是比较x1寄存器储存的字符串与输入的字符串是否相等，若相等则返回0.

使用disas strings\_not\_equal指令查看其反汇编代码

分析汇编语言得知该函数首先比较了两个字符串的长度，后继续比较其他内容。观察到该函数调用了两次string\_length函数，猜测该函数为输入一个字符串，返回该字符串的长度，在<+31>行对w21与w0这两个函数的返回值的比较可验证猜想。两次调用的寄存器为输入的字符串和地址0x402648储存的值，故输入的字符串一定为地址0x402648储存的值。

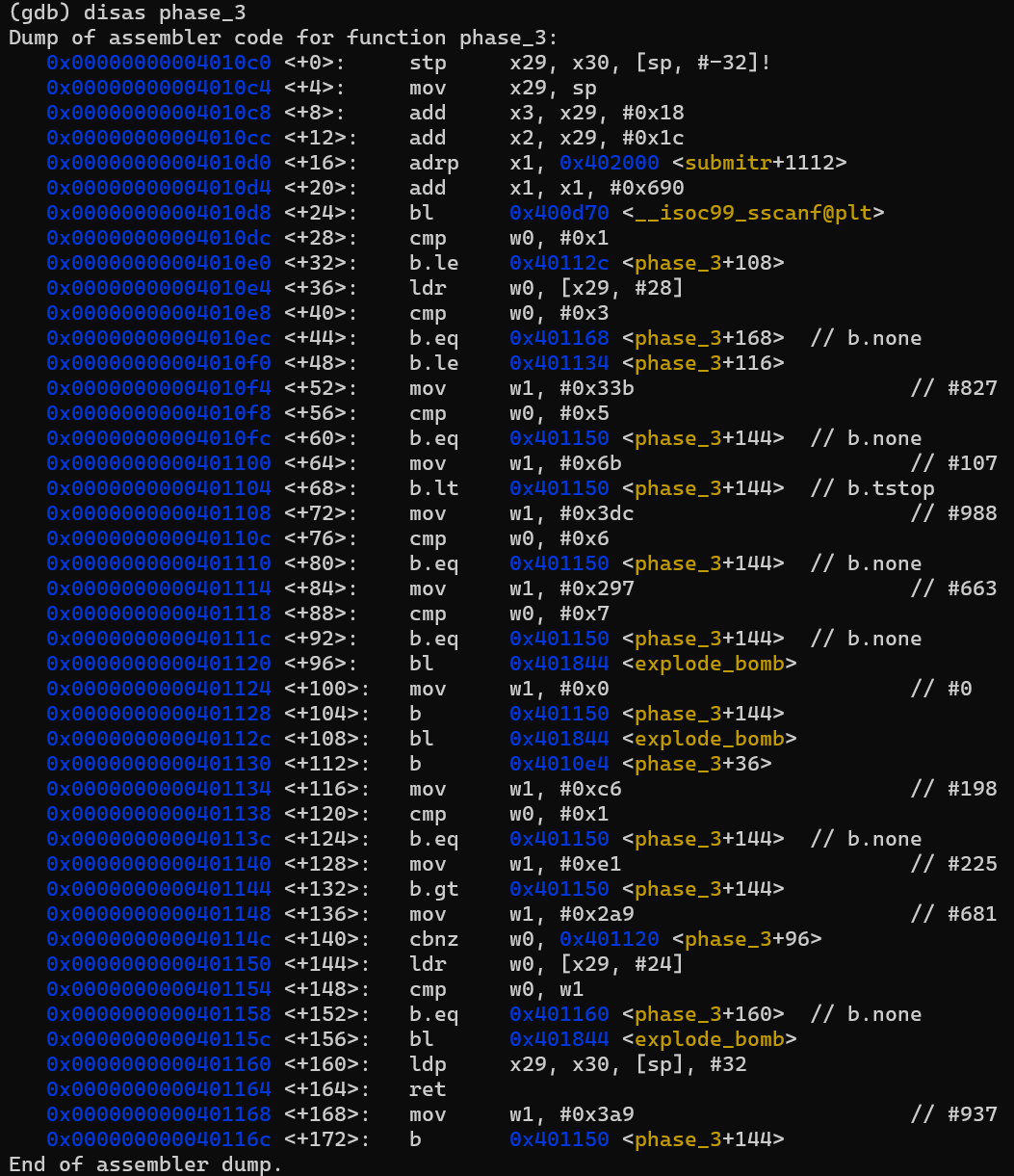
1732002809463使用x/s 0x402648指令查看该地址储存的值，该值为阶段一的答案，将其输入answer.txt文件

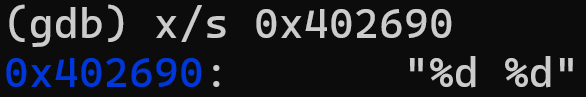
2、第二阶段

使用disas phase\_2指令查看其反汇编代码

分析汇编语言并结合普通炸弹的信息得知该函数输入6个整数并返回栈底指针w0储存第一个数的地址，即w0储存这个长度为6的一维数组的首地址。随后在<+24>行检验第一个数的值是否为0，若为0接着设置w0为下一个输入的数，在<+32>行检验第二个数的值是否为1，若为1则进入循环，w1依次储存第三、四、五、六个数字，要求第三个数的值为第一个和第二个数的值相加。即阶段二答案为0 1 1 2 3 5，将其输入answer.txt文件

3、第三阶段

使用disas phase\_3 指令查看其反汇编代码

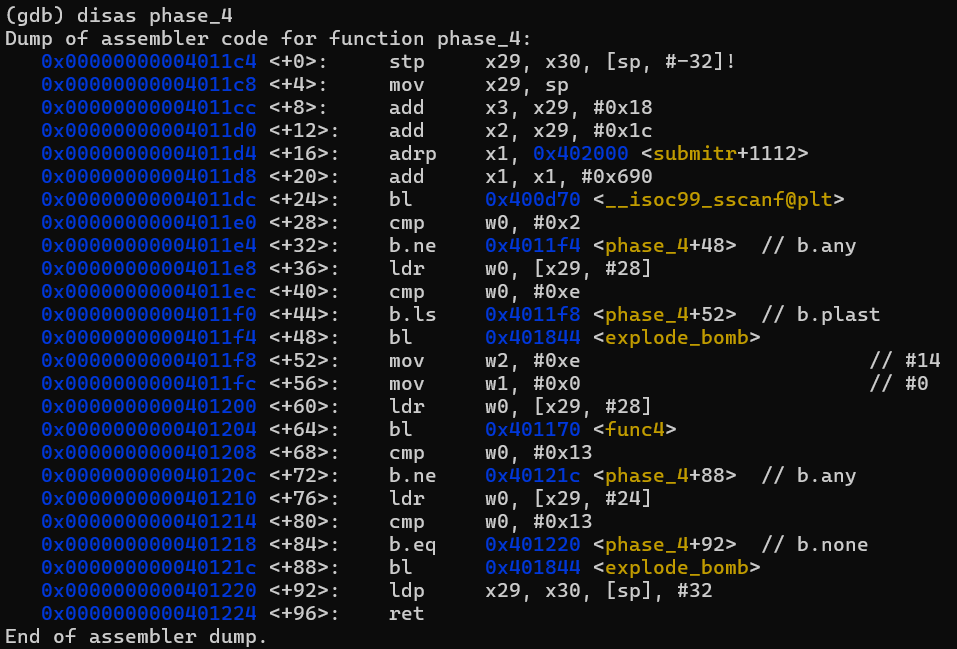
使用x/s 0x402690指令查看该地址储存的值

根据上文的分析得知该阶段首先输入两个数，w0储存这两个数。

接着从<+28>行开始分析，该行将w0即第一个数与1进行比较，若其小于等于1则炸弹爆炸。故第一个数必须大于1。之后进行了switch的操作，即多次判定w0的数值。为节省事件我们选择w0等于3的情况，<+44>行判断若w0为3跳转至<+168>行，随后将w1的值设置为0x3a9（即十进制937），跳转至<+144>行，将第二个数与w1比较，若不同则炸弹爆炸。故阶段三的答案的一种为3 937，将其输入answer.txt文件

4、第四阶段

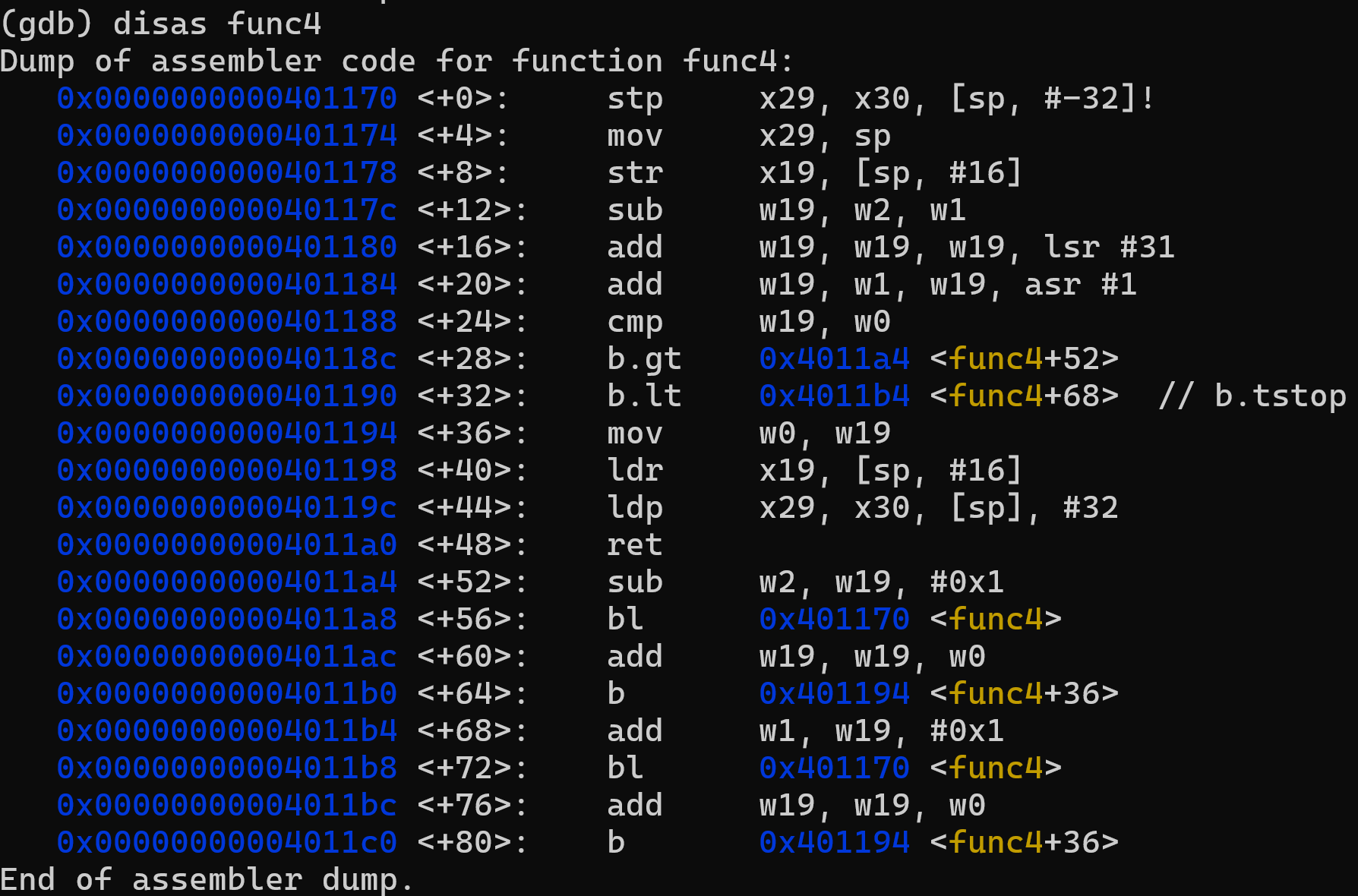
使用disas phase\_4 指令查看其反汇编代码



根据阶段3的分析得知该阶段首先输入两个数，接着在<+40>行判断第一个数是否小于14，若不同则炸弹爆炸；将w2和w1储存的数作为参数传入func4函数中。在其func4函数执行接收后判断其返回值是否为19，也判断第二个数是否为19，若不同则炸弹爆炸，若相同则继续执行。

经过如上的分析，我们可以确定第一个数一定小于等于14，第二个数为19

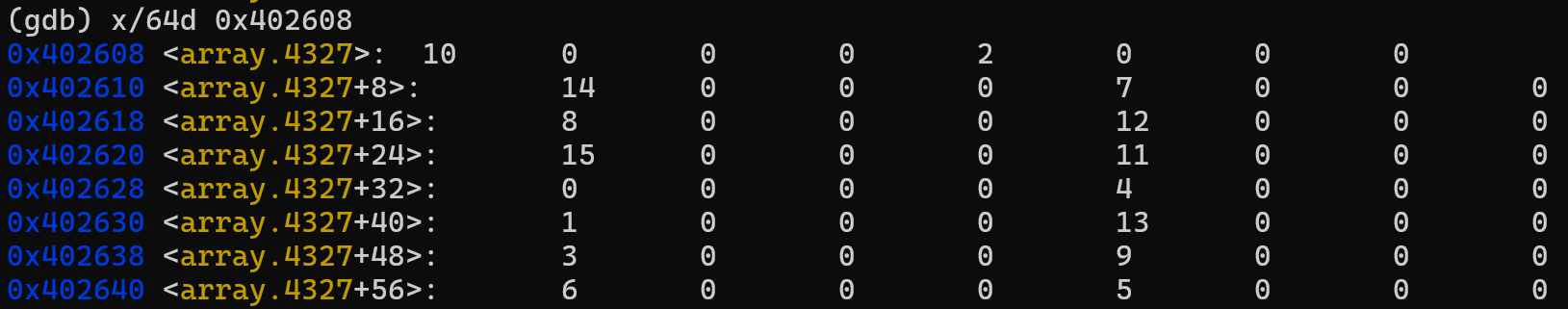
输入disas func4 指令查看func4函数的反汇编代码



不难看出该函数出现了递归操作，从0开始逐个输入函数可发现当输入4时返回值满足条件，故阶段四的答案的一种情况为4 19，将其输入answer.txt文件

5、第五阶段

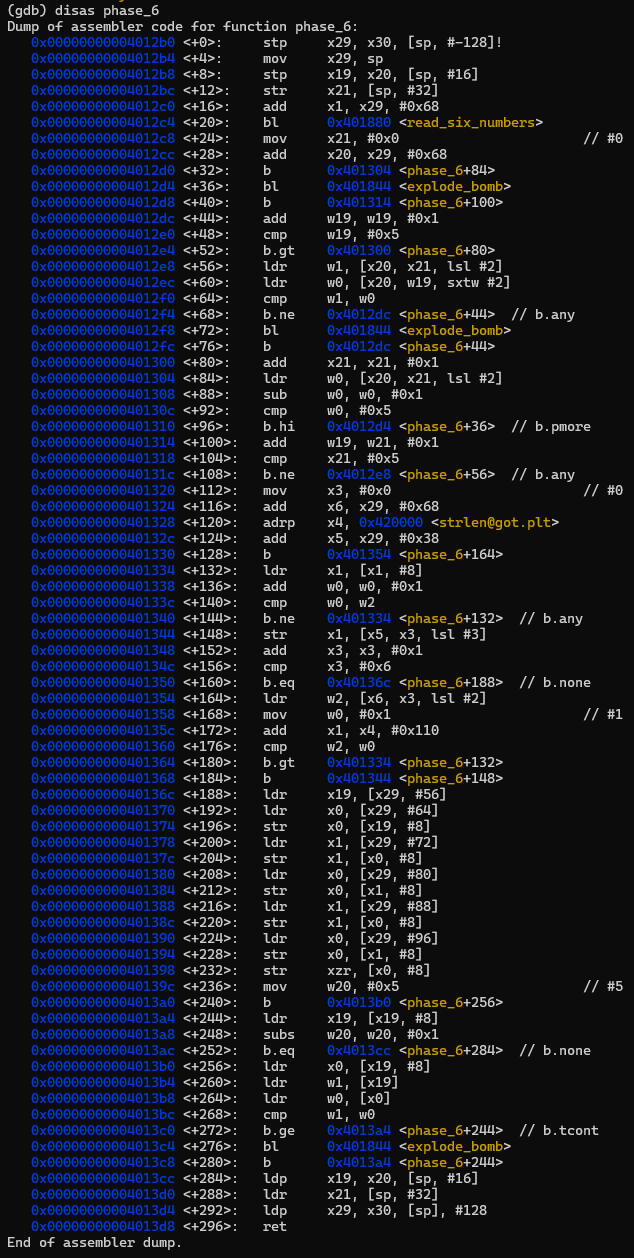
使用disas phase\_5 指令查看其反汇编代码

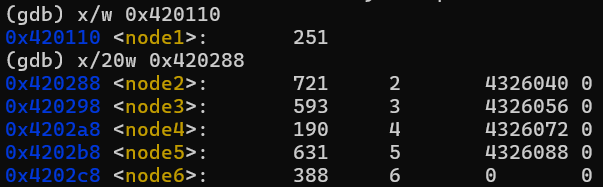
根据阶段3的分析得知该阶段首先输入两个数，使用x/64d 0x402608 指令查看该地址储存的值

分析汇编语言发现函数先将0x402608处的数组的前15个数相加后与输入的第二个数比较，再将数组的最后一个数和输入的第一个数比较。数组的前15个数的和为115，最后一个数的值为5，故阶段五的答案为5 115，将其输入answer.txt文件

6、第六阶段

使用disas phase\_6 指令查看其反汇编代码



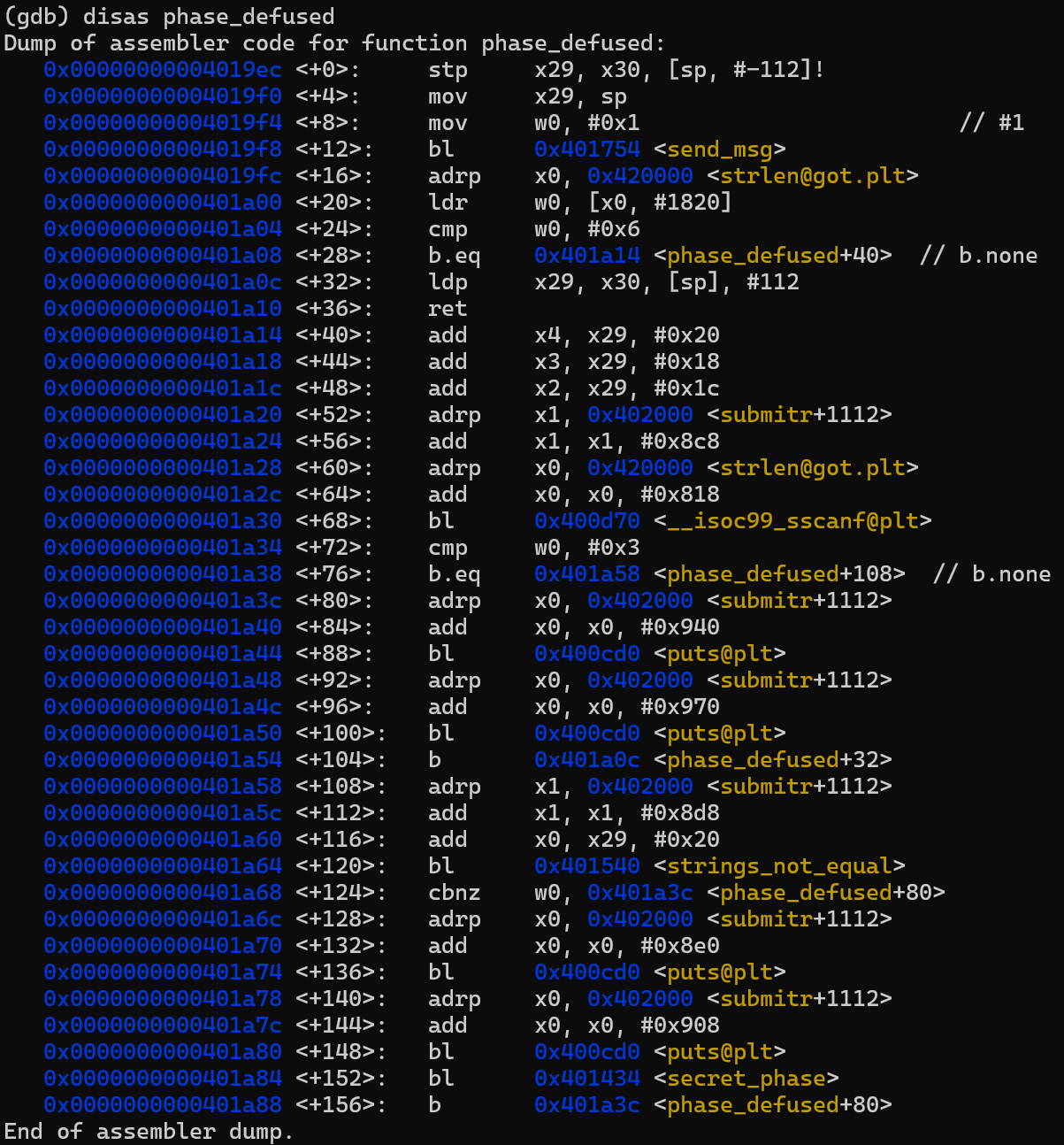
使用x/w 0x420110指令和 x/20w 0x420288指令查看该地址的内容

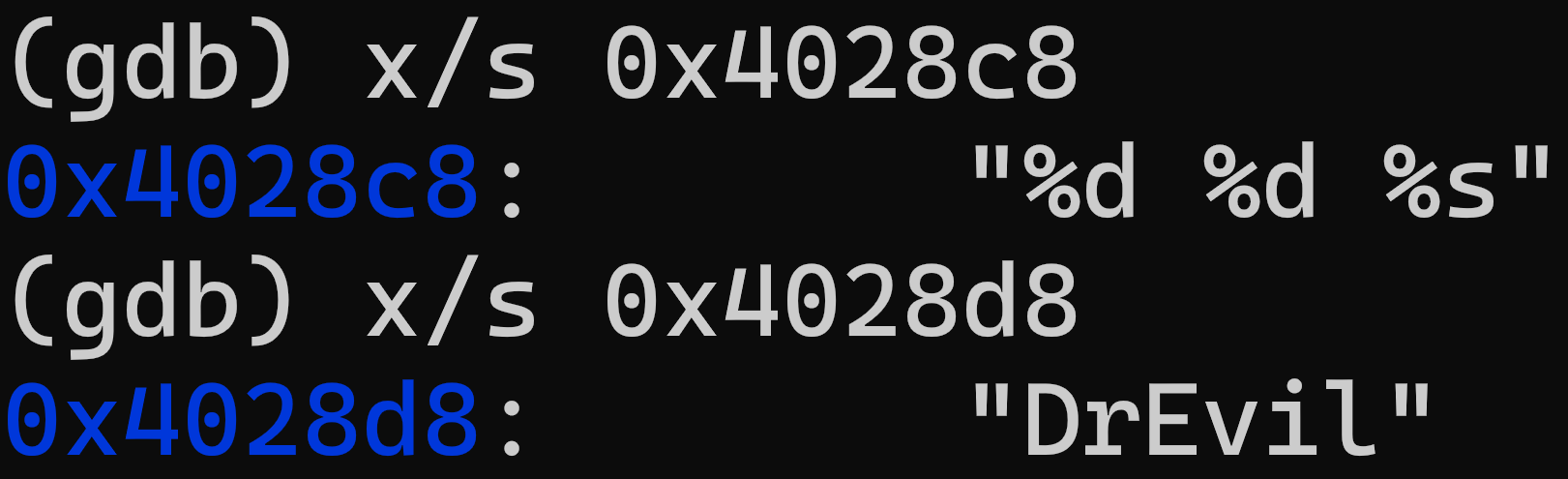
由普通炸弹阶段六的经验可知该阶段同样先输入6个数，随后在有6个节点的链表中按照输入的数字将节点重新排序，最后要求链表节点储存的值为从大到小的顺序。与普通炸弹的不同点在于没有用7减去输入的数字这一步。

分析完所有过程后回到链表具体储存的数进行分析。因为阶段六要求六个节点要为倒序，故六个链表应该从1-2-3-4-5-6的顺序变为2-5-3-6-1-4的顺序，其为阶段6的答案，将其输入answer.txt文件

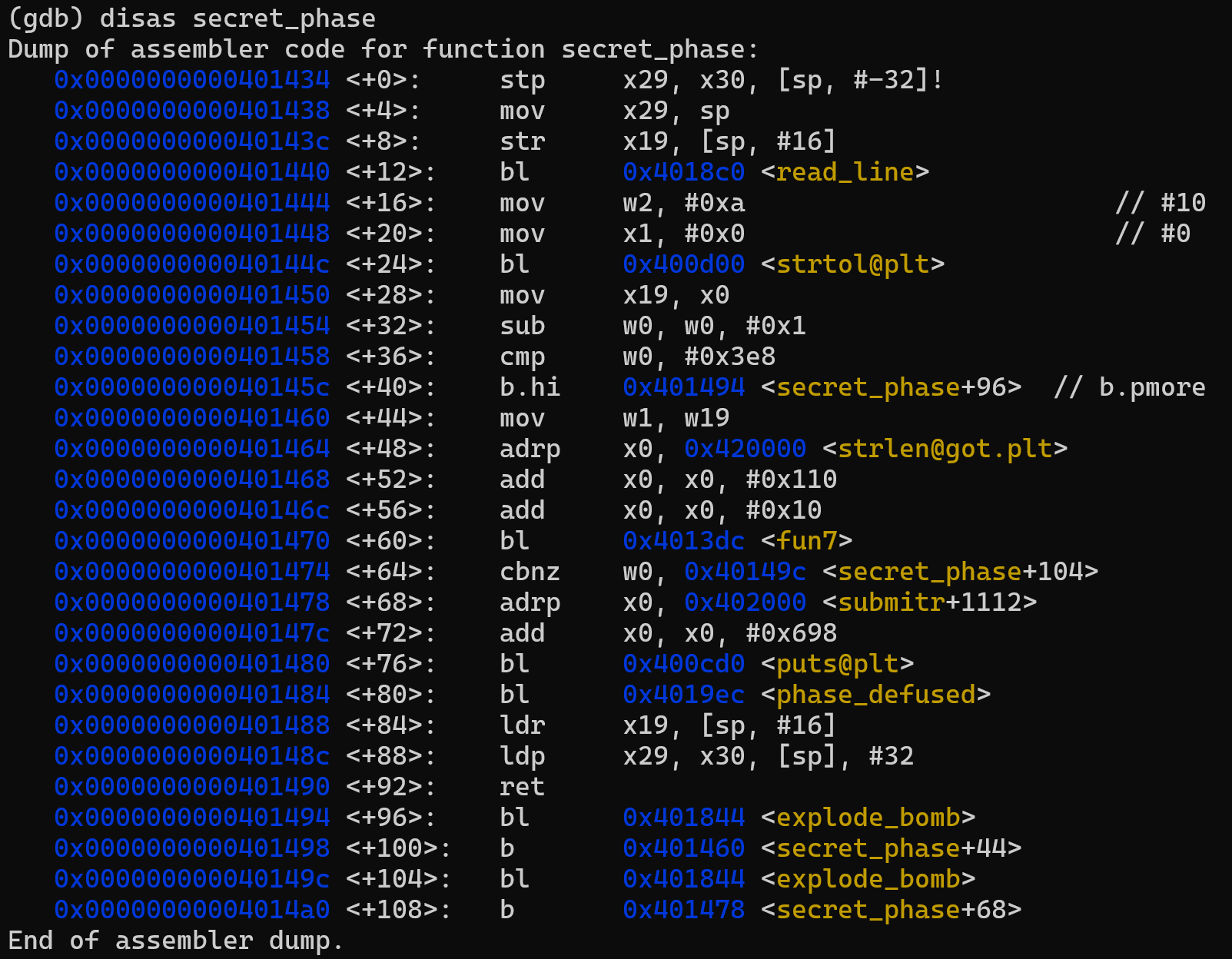
7、隐藏阶段

在分析完六个阶段的函数后，回到phase\_dufused函数进行分析。使用disas phase\_defused 指令查看其反汇编代码

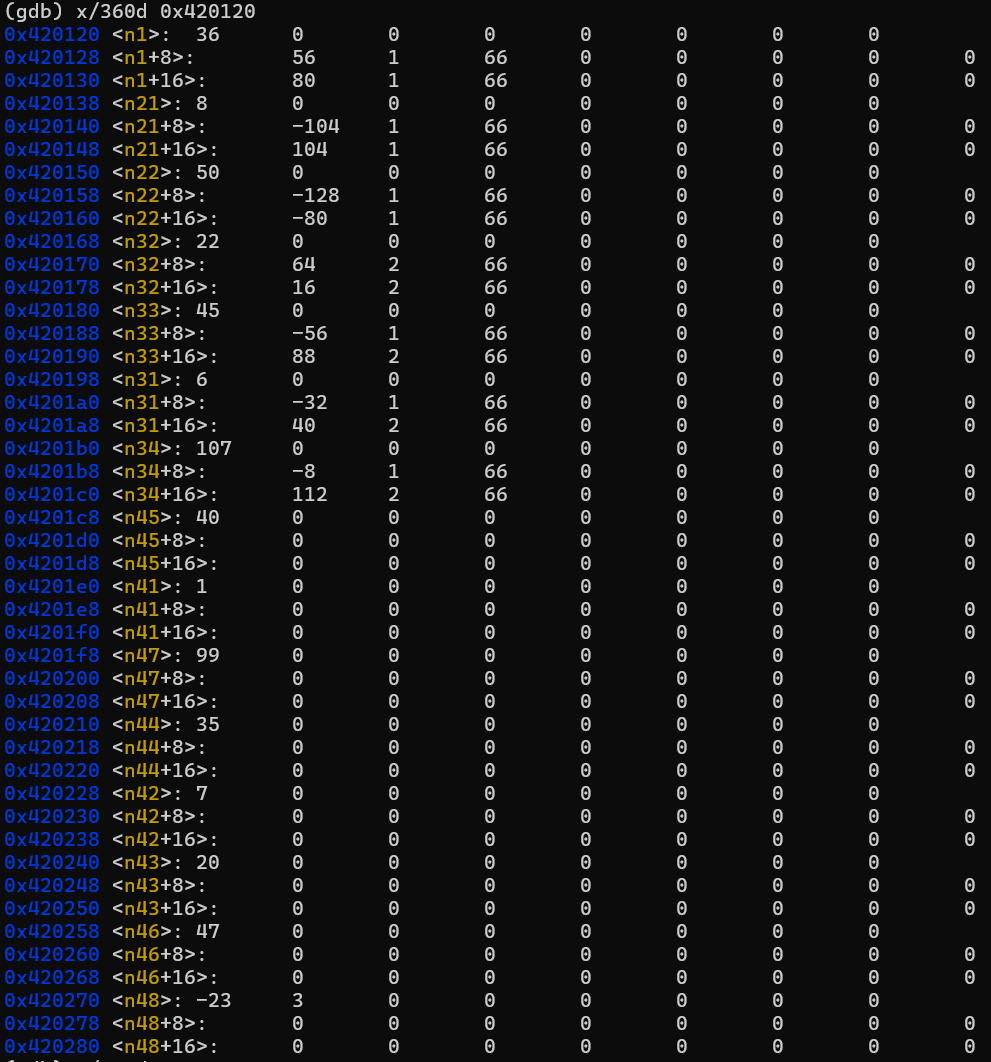


在<+123>行处发现了secret\_phase函数，使用x/s 0x4028c8和x/s 0x4028d8指令查看内容

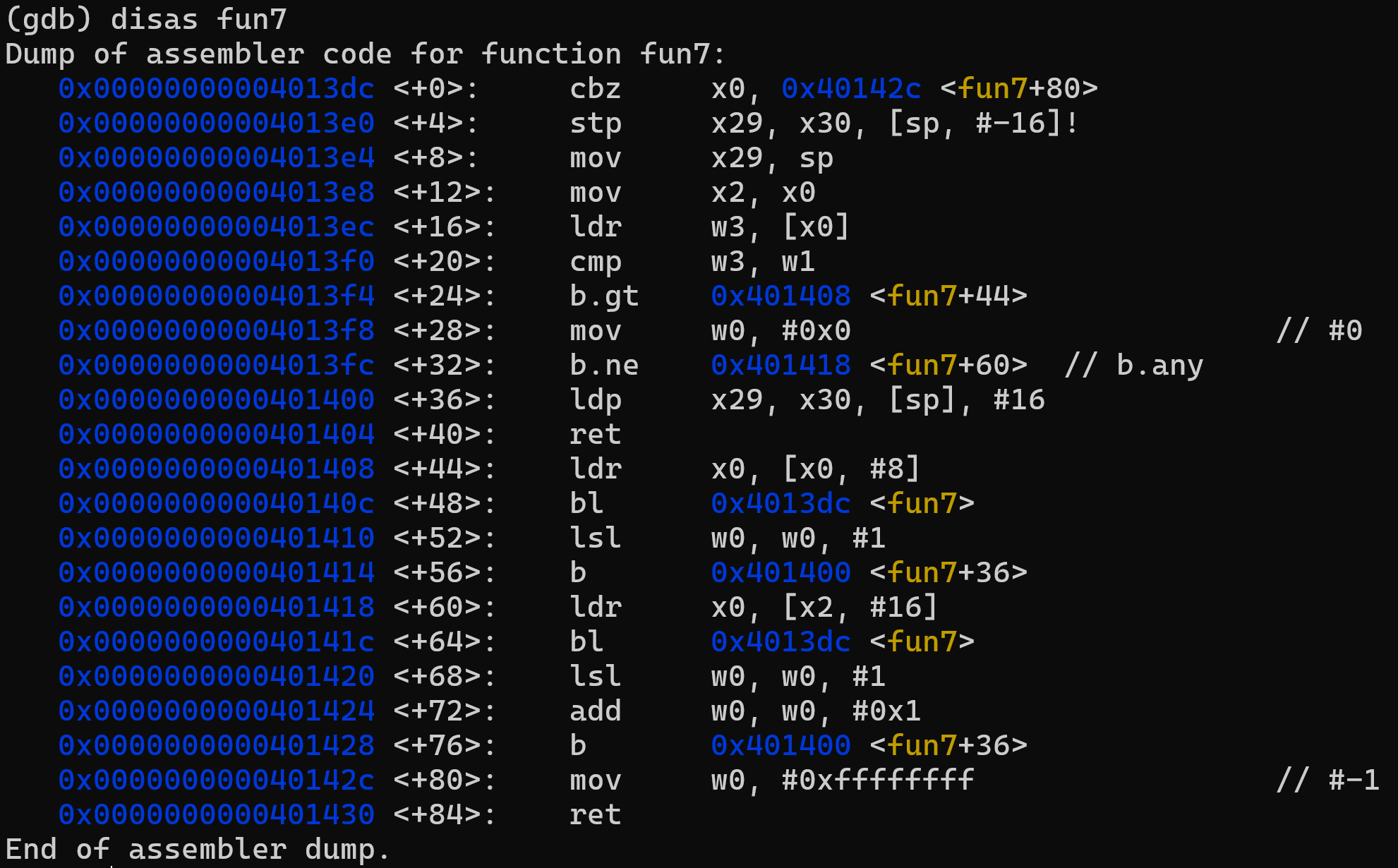
由普通炸弹的经验可知须在第四阶段的两个整数后加上DrEvil即可进入隐藏阶段。

输入disas secret\_phase指令查看其反汇编代码

分析得函数先设置w2为10，x1为0，后判断输入的数w0是否小于等于1000，若大于则炸弹爆炸，否则继续执行。使用x/360d 0x420120指令查看内容

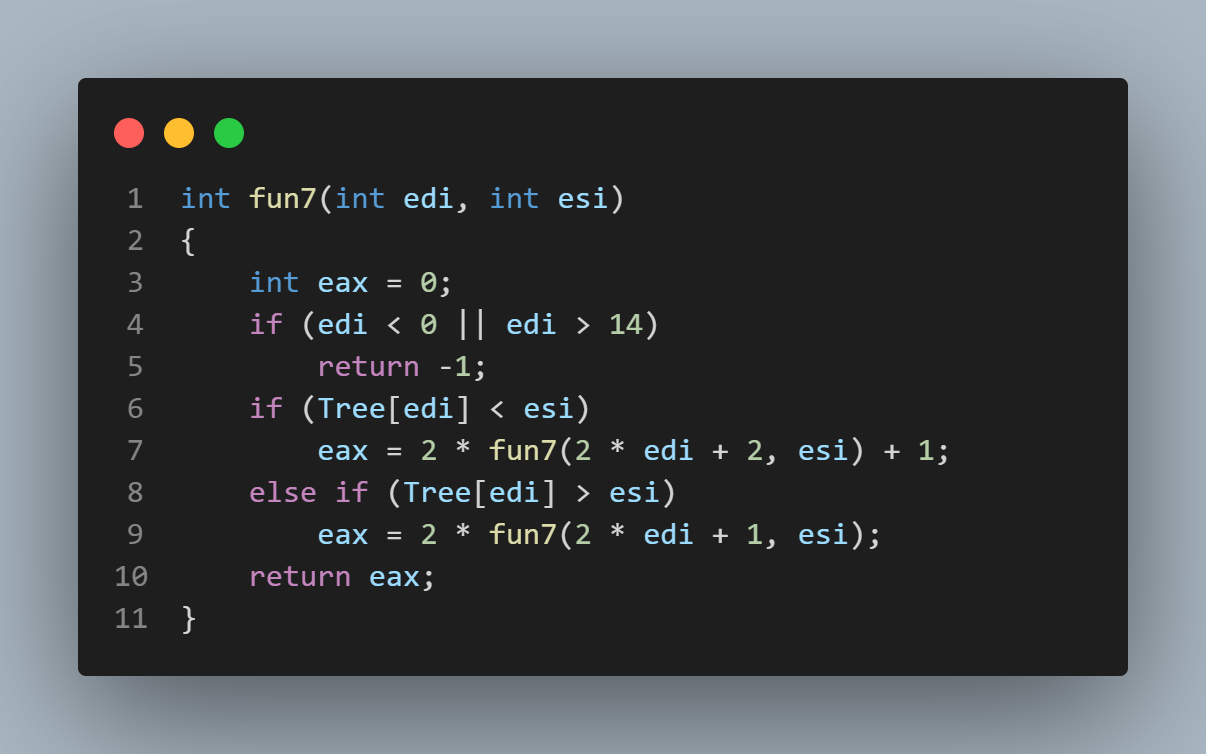
****

由图可知该地址处的内容为一个二叉树。随后将x0设置为该二叉树，后将x0和输入的数作为参数传入fun7函数。后判断fun7函数返回值是否为0，若不为0则炸弹爆炸，若为0则继续执行。输入disas fun7指令查看其反汇编代码



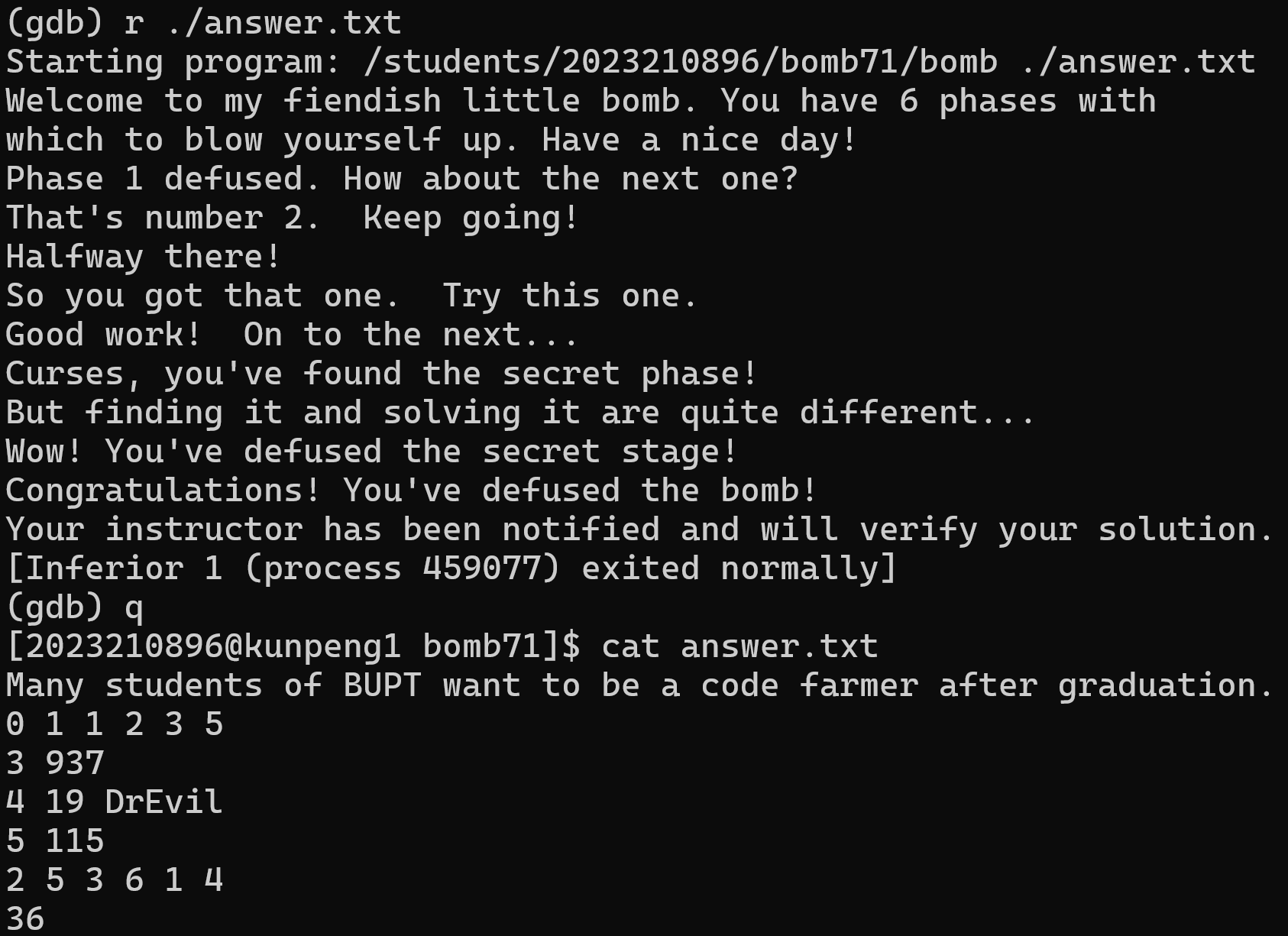
不难看出该函数出现了递归操作，故逐步分析汇编语言比较困难，不妨将其转化为C语言分析。

将汇编语言翻译成C语言为



将w0=0，1，……1000，1001代入该函数，并令函数的返回值为0时输出w0值，结果为1，6，8，36，答案从中任选一即可。取36为阶段6的答案，将其输入answer.txt文件

五、答案汇总与实验截图



六、总结体会

在这次高阶拆炸弹的实验中，出于相同的顾虑我依然全程使用静态调试的方式，即仅使用disas指令查看各个被调用的函数的反汇编代码，同时再结合arm64快速查看表一句一句将汇编语言翻译成C语言，虽然不如动态设置断点更加直观快捷，但是逐步分析反汇编代码让我的基本功更加扎实，再次加深了我对各种机器指令的理解。