二进制炸弹--拆弹实验

原创

阿布～ 于 2021-11-01 21:28:28 发布 1681 收藏 30

文章标签： 反汇编

本文链接：https://blog.csdn.net/weixin_46448719/article/details/108692985

二进制炸弹--拆弹实验

这是我们学校计算机系统基础课程的大作业实验，以下是我总结的每一关的通关方法。对于汇编语言的了解还不够深入，解释不好的地方还请大家批评指正。

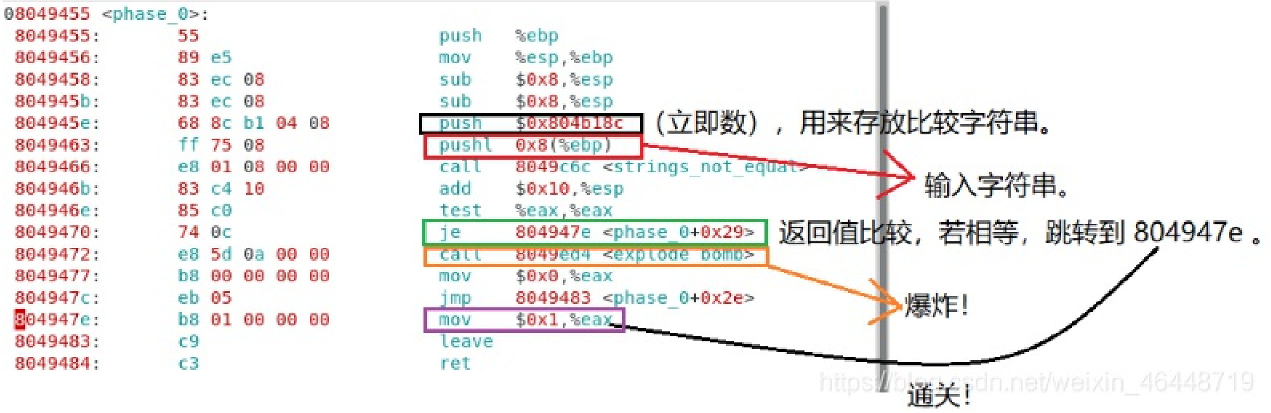
那，我们就出发吧~

大作业要求

该大作业为闯关实验，共需破解7关。下载以自己学号命名的<学号>.tar压缩文件，在Debian系统里进行解压缩（例：$tar -xvf 987654321.tar）后，通过对可执行文件bomb反汇编，分析汇编代码并获取相关线索，从而破解每一关的输入内容，完成闯关。

实验一.字符串比较

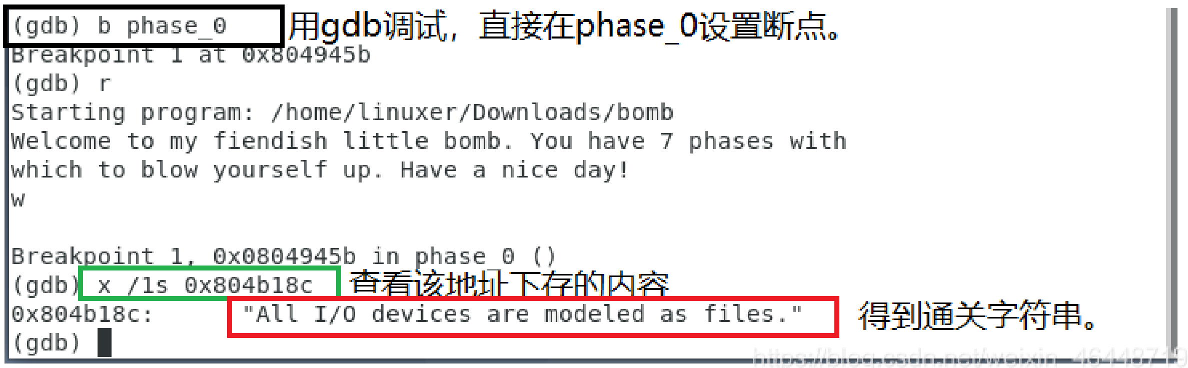
phase_0



```
08049455 <phase_0>:
 8049455:	55                   	push   %ebp
 8049456:	89 e5                	mov    %esp,%ebp
 8049458:	83 ec 08             	sub    $0x8,%esp
 804945b:	83 ec 08             	sub    $0x8,%esp
 804945e:	68 8c b1 04 08       	push   $0x804b18c
 8049463:	ff 75 08             	pushl  0x8(%ebp)
 8049466:	e8 01 08 00 00       	call   8049c6c <strings_not_equal>
 804946b:	83 c4 10             	add    $0x10,%esp
 804946e:	85 c0                	test   %eax,%eax
 8049470:	74 0c                	je     804947e <phase_0+0x29>
 8049472:	e8 5d 0a 00 00       	call   8049ed4 <explode_bomb>
 8049477:	b8 00 00 00 00       	mov    $0x0,%eax
 804947c:	eb 05                	jmp    8049483 <phase_0+0x2e>
 804947e:	b8 01 00 00 00       	mov    $0x1,%eax
 8049483:	c9                   	leave
 8049484:	c3                   	ret
```

（立即数），用来存放比较字符串。
输入字符串。
返回值比较，若相等，跳转到 804947e 。
爆炸!
通关!

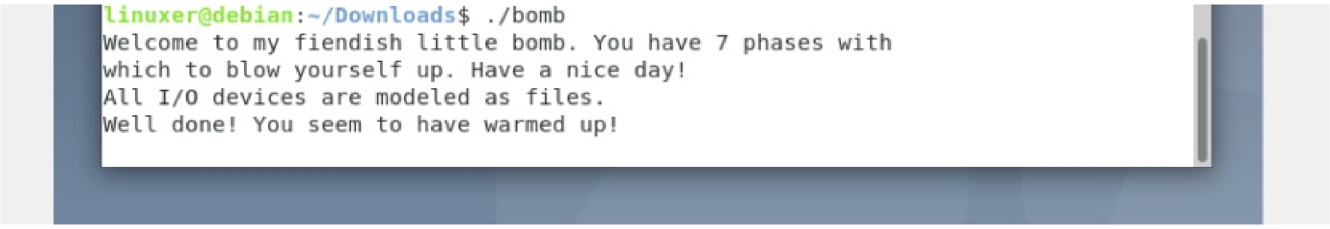
使用 objdump 生成 bomb 程序的汇编代码，下图将汇编代码定位在给出了第一关字符串 对比的地方<phase_0>。可以看出在调用<string_not_equal>函数前，现将要参与对比的两个 字符串压入栈中，push $0x804b18c 就是存放内置字符串的首地址，pushl 0x8(%ebp)是用户 输入字符串的首地址。



```
(gdb) b phase_0
Breakpoint 1 at 0x804945b
(gdb) r
Starting program: /home/linuxer/Downloads/bomb
Welcome to my fiendish little bomb. You have 7 phases with
which to blow yourself up. Have a nice day!
w

Breakpoint 1, 0x0804945b in phase_0 ()
(gdb) x /1s 0x804b18c
0x804b18c:	"All I/O devices are modeled as files."
(gdb) 
```

gdb 进行调试，断点设置在 b phase_0。这时根据汇编语句中发现的内置字符串首地址 0x804b18c，使用 examine 指令显示这个字符串为：All I/O devices are modeled as files.



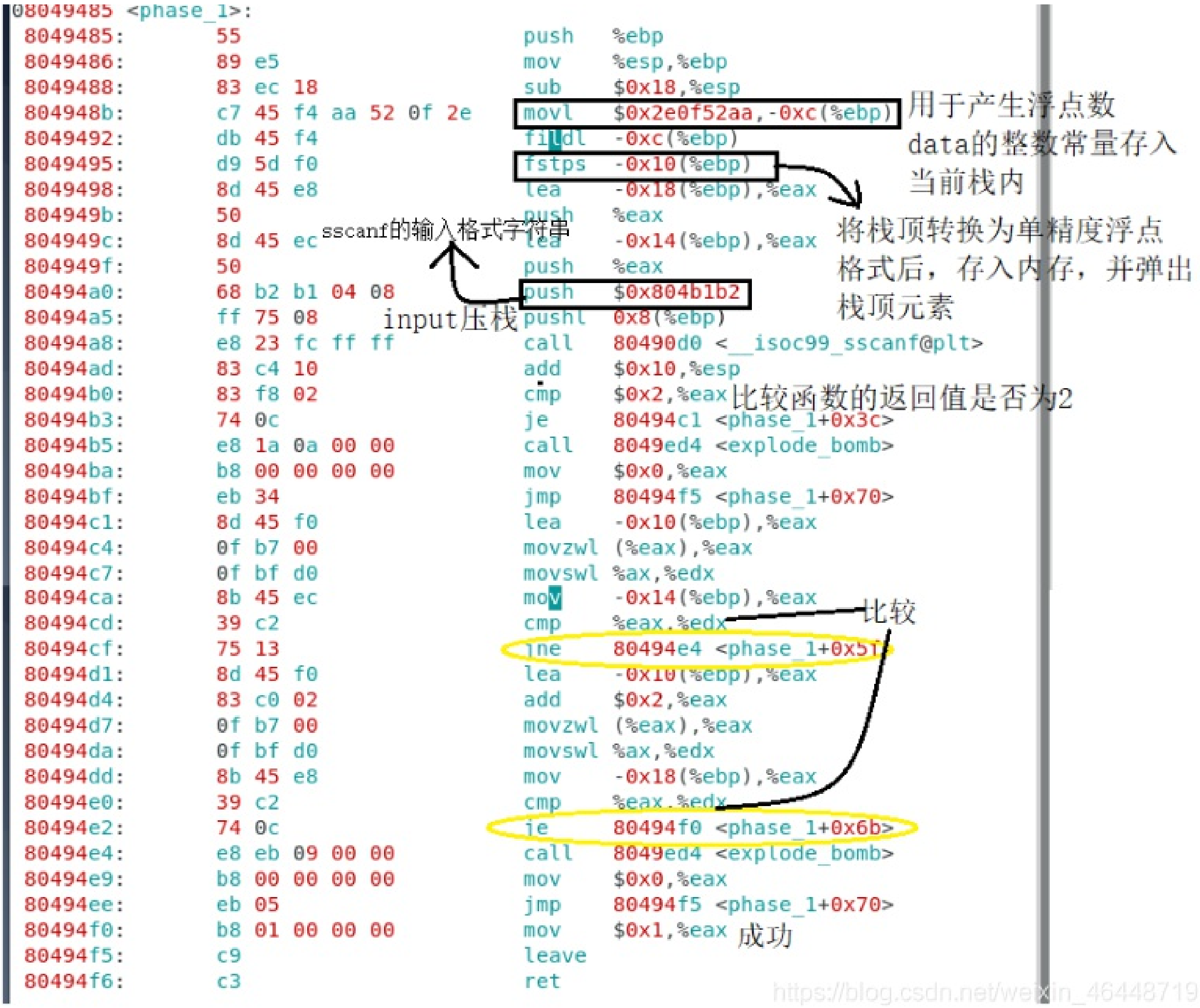
```
linuxer@debian:~/Downloads$ ./bomb
Welcome to my fiendish little bomb. You have 7 phases with
which to blow yourself up. Have a nice day!
All I/O devices are modeled as files.
Well done! You seem to have warmed up!
```

单步或继续执行后续的代码时输入这个字符串，结果显示成功过关。

第零关很easy，细心一点输入正确就好啦hhh

实验二.浮点表示

phase_1



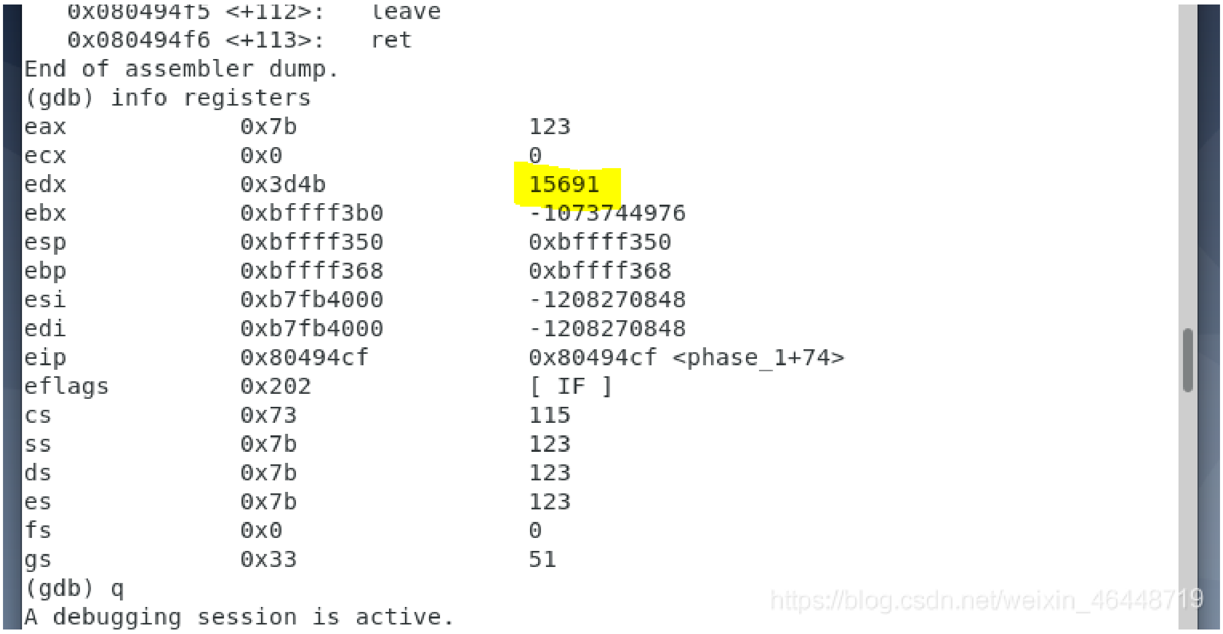
```
08049485 <phase_1>:
 8049485:	55                   	push   %ebp
 8049486:	89 e5                	mov    %esp,%ebp
 8049488:	83 ec 18             	sub    $0x18,%esp
 804948b:	c7 45 f4 aa 52 0f 2e 	movl   $0x2e0f52aa,-0xc(%ebp)
 8049492:	db 45 f4             	fildl  -0xc(%ebp)
 8049495:	d9 5d f0             	fstps  -0x10(%ebp)
 8049498:	8d 45 e8             	lea    -0x18(%ebp),%eax
 804949b:	50                   	push   %eax
 804949c:	8d 45 ec             	lea    -0x14(%ebp),%eax
 804949f:	50                   	push   %eax
 80494a0:	68 b2 b1 04 08       	push   $0x804b1b2
 80494a5:	ff 75 08             	pushl  0x8(%ebp)
 80494a8:	e8 23 fc ff ff       	call   80490d0 <__isoc99_sscanf@plt>
 80494ad:	83 c4 10             	add    $0x10,%esp
 80494b0:	83 f8 02             	cmp    $0x2,%eax
 80494b3:	74 0c                	je     80494c1 <phase_1+0x3c>
 80494b5:	e8 1a 0a 00 00       	call   8049ed4 <explode_bomb>
 80494ba:	b8 00 00 00 00       	mov    $0x0,%eax
 80494bf:	eb 34                	jmp    80494f5 <phase_1+0x70>
 80494c1:	8d 45 f0             	lea    -0x10(%ebp),%eax
 80494c4:	0f b7 00             	movzwl (%eax),%eax
 80494c7:	0f bf d0             	movswl %ax,%edx
 80494ca:	8b 45 ec             	mov    -0x14(%ebp),%eax
 80494cd:	39 c2                	cmp    %eax,%edx
 80494cf:	75 13                	jne    80494e4 <phase_1+0x5f>
 80494d1:	8d 45 f0             	lea    -0x10(%ebp),%eax
 80494d4:	83 c0 02             	add    $0x2,%eax
 80494d7:	0f b7 00             	movzwl (%eax),%eax
 80494da:	0f bf d0             	movswl %ax,%edx
 80494dd:	8b 45 e8             	mov    -0x18(%ebp),%eax
 80494e0:	39 c2                	cmp    %eax,%edx
 80494e2:	74 0c                	je     80494f0 <phase_1+0x6b>
 80494e4:	e8 eb 09 00 00       	call   8049ed4 <explode_bomb>
 80494e9:	b8 00 00 00 00       	mov    $0x0,%eax
 80494ee:	eb 05                	jmp    80494f5 <phase_1+0x70>
 80494f0:	b8 01 00 00 00       	mov    $0x1,%eax
 80494f5:	c9                   	leave
 80494f6:	c3                   	ret
```

用于产生浮点数data的整数常量存入当前栈内
将栈顶转换为单精度浮点格式后，存入内存，并弹出栈顶元素
sscanf的输入格式字符串
input压栈
比较函数的返回值是否为2
比较
成功

由 gdb 调试得知，在第二关需要输入的是两个整数。整数0x2e0f52aa使用单精度浮点数格式表达为0x4E383D4B，分别将高低位字节转换为十进制整数后为15691和20024。按照sscanf()指定的格式输入15691 20024后，回车。

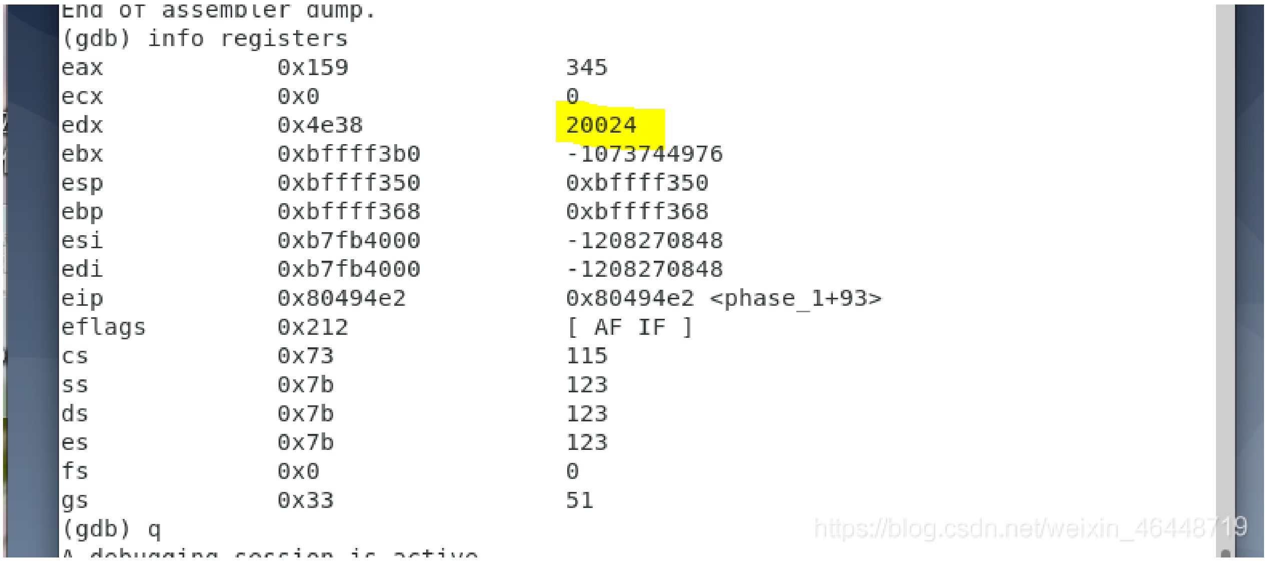
（其实，数值转换这个过程是应付老师的hh，我的做法是通过gdb调试直接查找寄存器地址内容~，由于我有一个小伙伴将这一部分解释的超级详细，下面我附上我的大概解题思路和张大大同学的博客，供大家参考 偷懒 ~~hhh）

通过在phase_1设置断点，运行到phase_1后先随意输入两个整数：123 345



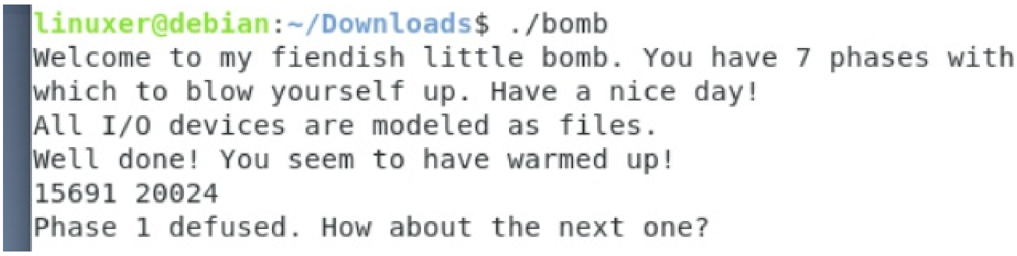
```
   0x080494f5 <+112>:    leave
   0x080494f6 <+113>:    ret
End of assembler dump.
(gdb) info registers
eax            0x7b                123
ecx            0x0                 0
edx            0x3d4b              15691
ebx            0xbffff3b0          -1073744976
esp            0xbffff350          0xbffff350
ebp            0xbffff368          0xbffff368
esi            0xb7fb4000          -1208270848
edi            0xb7fb4000          -1208270848
eip            0x80494cf           0x80494cf <phase_1+74>
eflags         0x202               [ IF ]
cs             0x73                115
ss             0x7b                123
ds             0x7b                123
es             0x7b                123
fs             0x0                 0
gs             0x33                51
(gdb) q
A debugging session is active.
```

跟踪到低字节整数为：15691；



```
End of assembler dump.
(gdb) info registers
eax            0x159               345
ecx            0x0                 0
edx            0x4e38              20024
ebx            0xbffff3b0          -1073744976
esp            0xbffff350          0xbffff350
ebp            0xbffff368          0xbffff368
esi            0xb7fb4000          -1208270848
edi            0xb7fb4000          -1208270848
eip            0x80494e2           0x80494e2 <phase_1+93>
eflags         0x212               [ AF IF ]
cs             0x73                115
ss             0x7b                123
ds             0x7b                123
es             0x7b                123
fs             0x0                 0
gs             0x33                51
(gdb) q
A debugging session is active.
```

跟踪到高字节整数为：20024；在找出答案过程中，所用指令有：break phase_1;run;stepi;disas;info registers;



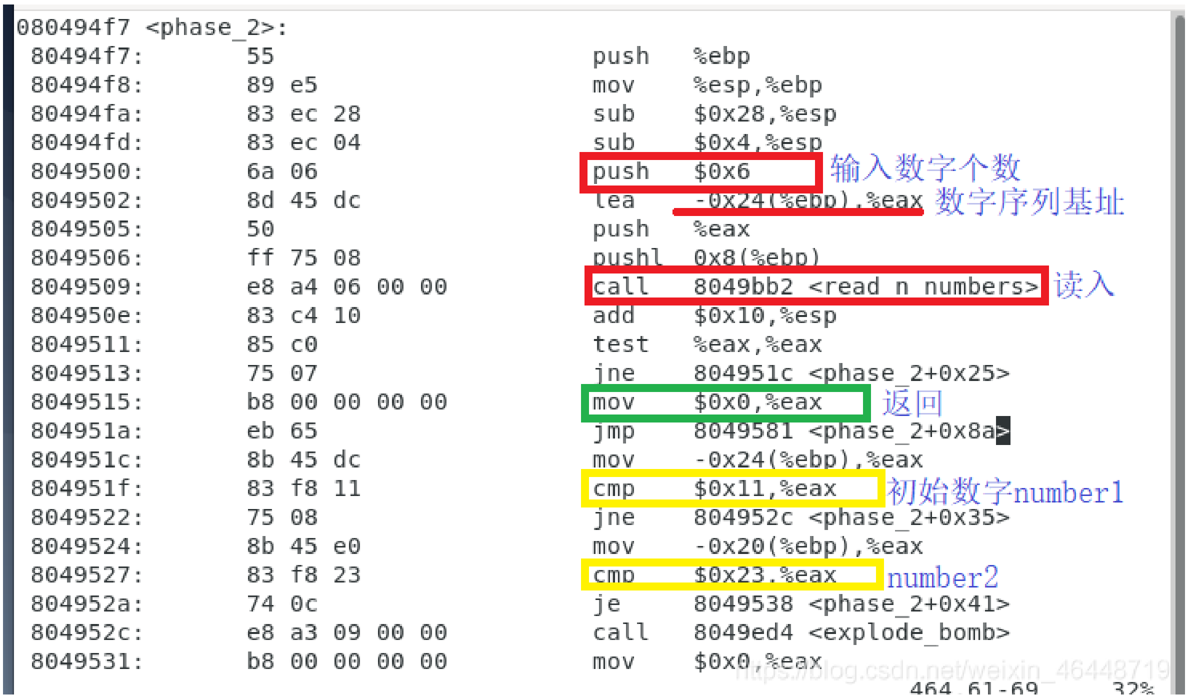
```
linuxer@debian:~/Downloads$ ./bomb
Welcome to my fiendish little bomb. You have 7 phases with
which to blow yourself up. Have a nice day!
All I/O devices are modeled as files.
Well done! You seem to have warmed up!
15691 20024
Phase 1 defused. How about the next one?
```

第一关就到这啦~我们继续

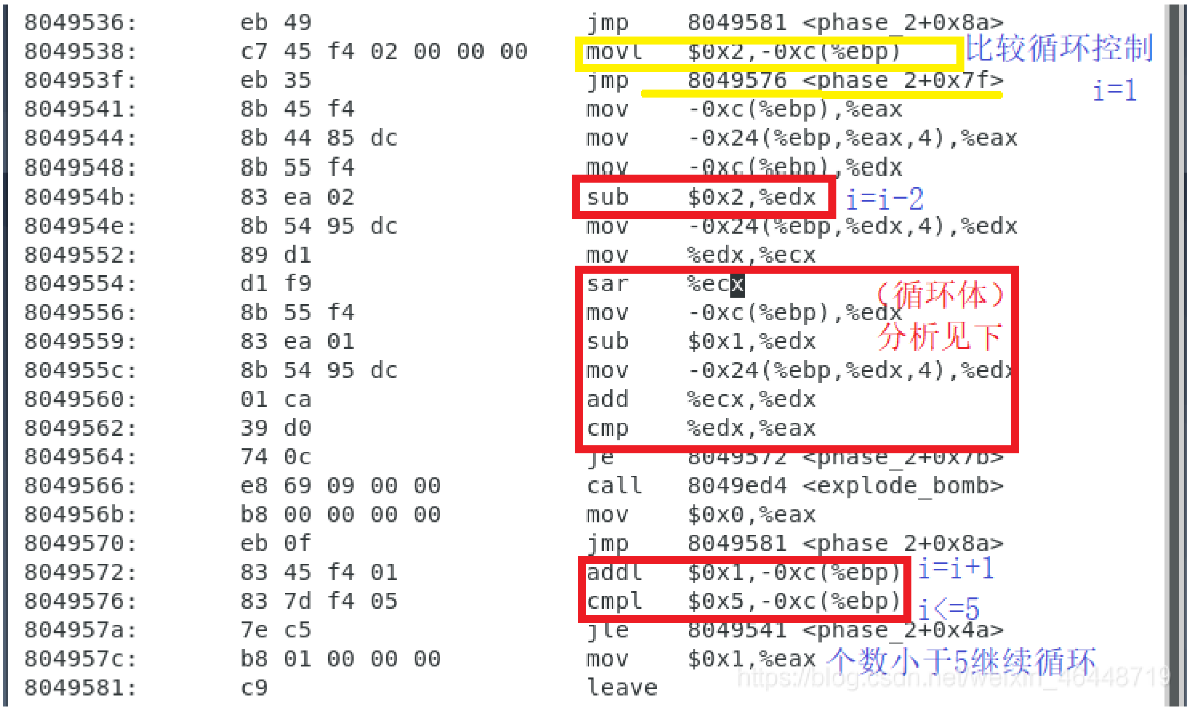
https://blog.csdn.net/weixin_45809643/article/details/106875783?utm_source=app (张大大的记录哦~）

实验三.循环

phase_2



```
080494f7 <phase_2>:
 80494f7:       55                      push   %ebp
 80494f8:       89 e5                   mov    %esp,%ebp
 80494fa:       83 ec 28                sub    $0x28,%esp
 80494fd:       83 ec 04                sub    $0x4,%esp
 8049500:       6a 06                   push   $0x6
 8049502:       8d 45 dc                lea    -0x24(%ebp),%eax
 8049505:       50                      push   %eax
 8049506:       ff 75 08                pushl  0x8(%ebp)
 8049509:       e8 a4 06 00 00          call   8049bb2 <read_n_numbers>
 804950e:       83 c4 10                add    $0x10,%esp
 8049511:       85 c0                   test   %eax,%eax
 8049513:       75 07                   jne    804951c <phase_2+0x25>
 8049515:       b8 00 00 00 00          mov    $0x0,%eax
 804951a:       eb 65                   jmp    8049581 <phase_2+0x8a>
 804951c:       8b 45 dc                mov    -0x24(%ebp),%eax
 804951f:       83 f8 11                cmp    $0x11,%eax
 8049522:       75 08                   jne    804952c <phase_2+0x35>
 8049524:       8b 45 e0                mov    -0x20(%ebp),%eax
 8049527:       83 f8 23                cmp    $0x23,%eax
 804952a:       74 0c                   je     8049538 <phase_2+0x41>
 804952c:       e8 a3 09 00 00          call   8049ed4 <explode_bomb>
 8049531:       b8 00 00 00 00          mov    $0x0,%eax
                                                         464,61-69     32%
```



```
 8049536:       eb 49                   jmp    8049581 <phase_2+0x8a>
 8049538:       c7 45 f4 02 00 00 00    movl   $0x2,-0xc(%ebp)
 804953f:       eb 35                   jmp    8049576 <phase_2+0x7f>
 8049541:       8b 45 f4                mov    -0xc(%ebp),%eax
 8049544:       8b 44 85 dc             mov    -0x24(%ebp,%eax,4),%eax
 8049548:       8b 55 f4                mov    -0xc(%ebp),%edx
 804954b:       83 ea 02                sub    $0x2,%edx
 804954e:       8b 54 95 dc             mov    -0x24(%ebp,%edx,4),%edx
 8049552:       89 d1                   mov    %edx,%ecx
 8049554:       d1 f9                   sar    %ecx
 8049556:       8b 55 f4                mov    -0xc(%ebp),%edx
 8049559:       83 ea 01                sub    $0x1,%edx
 804955c:       8b 54 95 dc             mov    -0x24(%ebp,%edx,4),%edx
 8049560:       01 ca                   add    %ecx,%edx
 8049562:       39 d0                   cmp    %edx,%eax
 8049564:       74 0c                   je     8049572 <phase_2+0x7b>
 8049566:       e8 69 09 00 00          call   8049ed4 <explode_bomb>
 804956b:       b8 00 00 00 00          mov    $0x0,%eax
 8049570:       eb 0f                   jmp    8049581 <phase_2+0x8a>
 8049572:       83 45 f4 01             addl   $0x1,-0xc(%ebp)
 8049576:       83 7d f4 05             cmpl   $0x5,-0xc(%ebp)
 804957a:       7e c5                   jle    8049541 <phase_2+0x4a>
 804957c:       b8 01 00 00 00          mov    $0x1,%eax
 8049581:       c9                      leave
```

通过对源代码的分析，找到了它的循环条件和循环体，对其进行运算： number[i]=number[i-2]/2+number[i-1] 又知，初始数字为 0x11=17和0x23=35 即

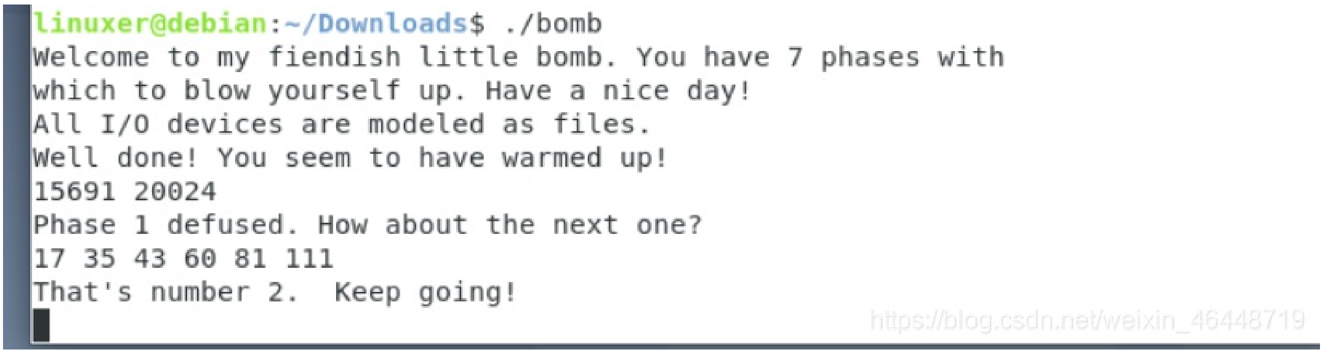
number[0]=17,number[1]=35。

number[2]=(17/2) + 35 = 43;

number[3]=(35/2) + 43 = 60;

number[4]=(43/2) + 60 = 81;

number[5]=(60/2) + 81 = 111;



```
linuxer@debian:~/Downloads$ ./bomb
Welcome to my fiendish little bomb. You have 7 phases with
which to blow yourself up. Have a nice day!
All I/O devices are modeled as files.
Well done! You seem to have warmed up!
15691 20024
Phase 1 defused. How about the next one?
17 35 43 60 81 111
That's number 2.  Keep going!
```

这一关我所遇到的循环体给大家总结了一下，需要注意的是一定先要找到输入数字的个数和第一个数值（有时候给的是前两个，比如我的这个）下面是我遇到的循环：

1.number[i+1]=number[i]/2+1

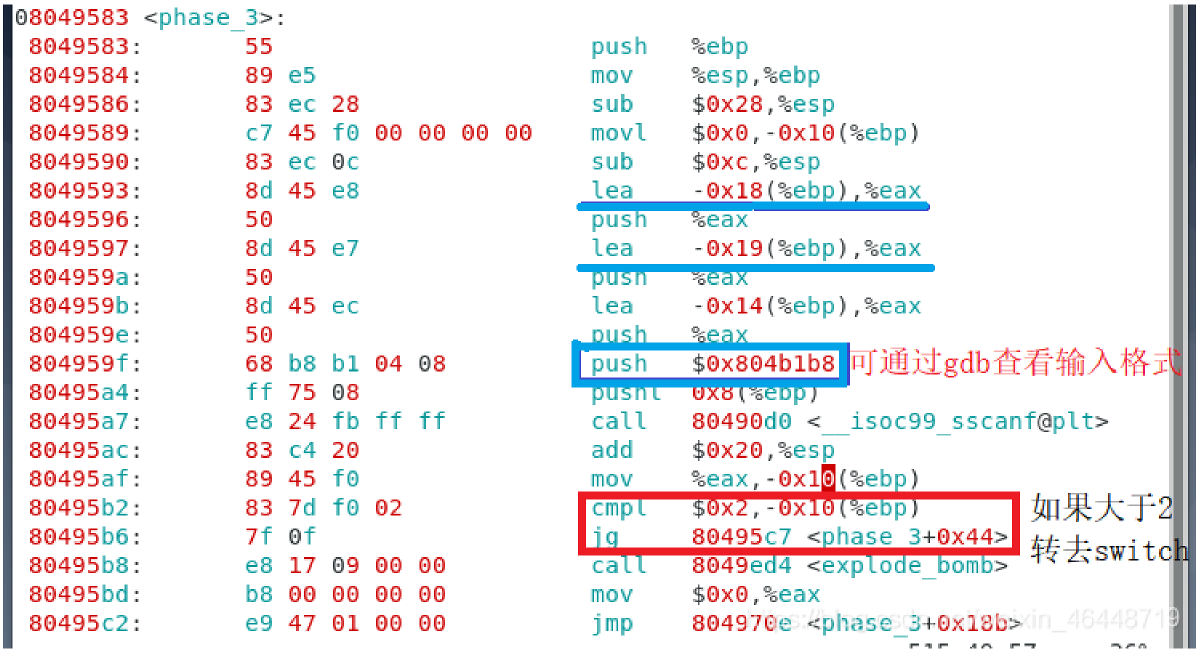
2.number[i]=number[i-1]-2*i+1

3.number[i]=number[i-2]/2+number[i-1]

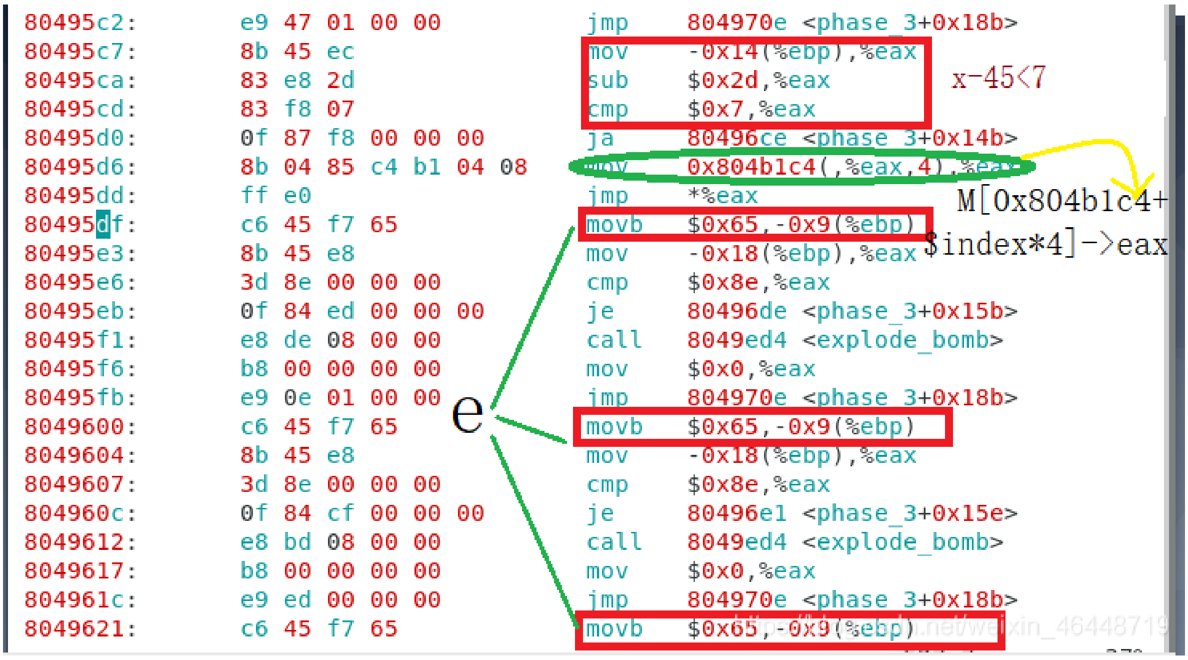
好像就这些嘤嘤嘤~ 过了过了…

实验四.条件/分支

phase_3

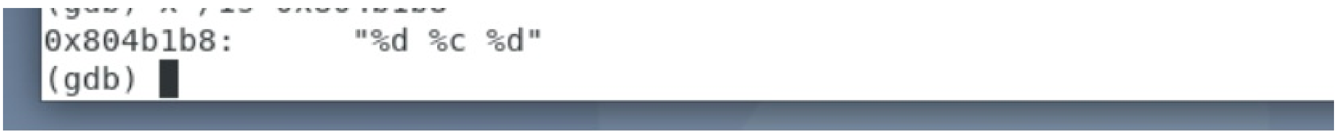


```
08049583 <phase_3>:
 8049583:	55                   	push   %ebp
 8049584:	89 e5                	mov    %esp,%ebp
 8049586:	83 ec 28             	sub    $0x28,%esp
 8049589:	c7 45 f0 00 00 00 00 	movl   $0x0,-0x10(%ebp)
 8049590:	83 ec 0c             	sub    $0xc,%esp
 8049593:	8d 45 e8             	lea    -0x18(%ebp),%eax
 8049596:	50                   	push   %eax
 8049597:	8d 45 e7             	lea    -0x19(%ebp),%eax
 804959a:	50                   	push   %eax
 804959b:	8d 45 ec             	lea    -0x14(%ebp),%eax
 804959e:	50                   	push   %eax
 804959f:	68 b8 b1 04 08       	push   $0x804b1b8
 80495a4:	ff 75 08             	pushl  0x8(%ebp)
 80495a7:	e8 24 fb ff ff       	call   80490d0 <__isoc99_sscanf@plt>
 80495ac:	83 c4 20             	add    $0x20,%esp
 80495af:	89 45 f0             	mov    %eax,-0x10(%ebp)
 80495b2:	83 7d f0 02          	cmpl   $0x2,-0x10(%ebp)
 80495b6:	7f 0f                	jg     80495c7 <phase_3+0x44>
 80495b8:	e8 17 09 00 00       	call   8049ed4 <explode_bomb>
 80495bd:	b8 00 00 00 00       	mov    $0x0,%eax
 80495c2:	e9 47 01 00 00       	jmp    804970e <phase_3+0x18b>
```



```
 80495c2:	e9 47 01 00 00       	jmp    804970e <phase_3+0x18b>
 80495c7:	8b 45 ec             	mov    -0x14(%ebp),%eax
 80495ca:	83 e8 2d             	sub    $0x2d,%eax
 80495cd:	83 f8 07             	cmp    $0x7,%eax
 80495d0:	0f 87 f8 00 00 00    	ja     80496ce <phase_3+0x14b>
 80495d6:	8b 04 85 c4 b1 04 08 	mov    0x804b1c4(,%eax,4),%eax
 80495dd:	ff e0                	jmp    *%eax
 80495df:	c6 45 f7 65          	movb   $0x65,-0x9(%ebp)
 80495e3:	8b 45 e8             	mov    -0x18(%ebp),%eax
 80495e6:	3d 8e 00 00 00       	cmp    $0x8e,%eax
 80495eb:	0f 84 ed 00 00 00    	je     80496de <phase_3+0x15b>
 80495f1:	e8 de 08 00 00       	call   8049ed4 <explode_bomb>
 80495f6:	b8 00 00 00 00       	mov    $0x0,%eax
 80495fb:	e9 0e 01 00 00       	jmp    804970e <phase_3+0x18b>
 8049600:	c6 45 f7 65          	movb   $0x65,-0x9(%ebp)
 8049604:	8b 45 e8             	mov    -0x18(%ebp),%eax
 8049607:	3d 8e 00 00 00       	cmp    $0x8e,%eax
 804960c:	0f 84 cf 00 00 00    	je     80496e1 <phase_3+0x15e>
 8049612:	e8 bd 08 00 00       	call   8049ed4 <explode_bomb>
 8049617:	b8 00 00 00 00       	mov    $0x0,%eax
 804961c:	e9 ed 00 00 00       	jmp    804970e <phase_3+0x18b>
 8049621:	c6 45 f7 65          	movb   $0x65,-0x9(%ebp)
```

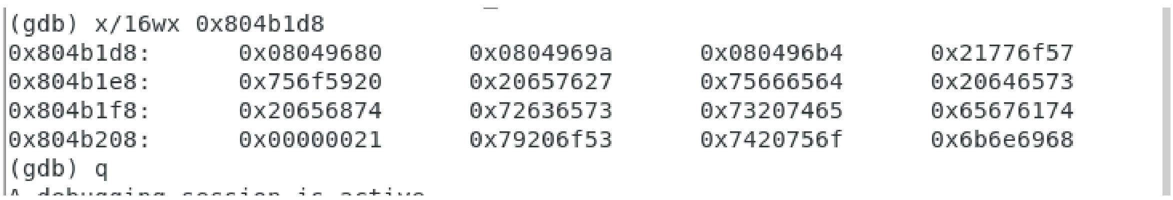
在无法获得源程序的情况下，只能通过对可执行程序进行反汇编来获得程序的汇编代 码。观察 bomb.s 对应的汇编代码，分析源程序的功能。



```
0x804b1b8:	"%d %c %d"
(gdb) 
```

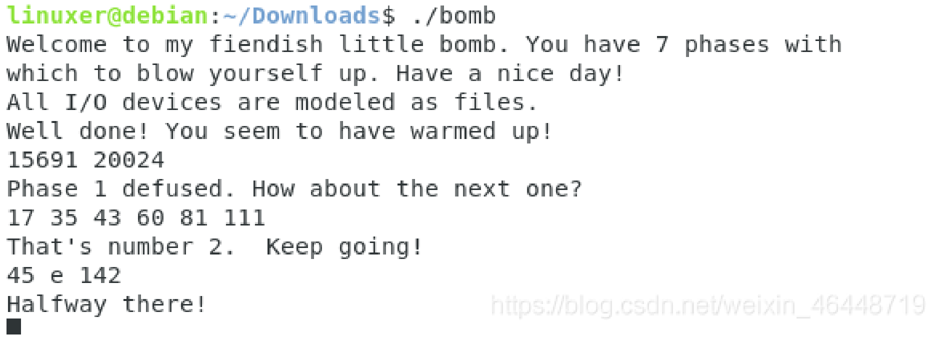
（有些只需要输入 %d %d 俩个整数）

结合源代码可以发现，程序内部对将要输入的第一个数字的要求是减去 45 后的值小于 7，第二个输入应该根据第一个输入的整数的值来判断程序转到哪个分支执行，从而确 定第二个数字的值应该输入哪个。 假设输入值为 45，然后通过计算在 gdb 中查看表基址。



```
(gdb) x/16wx 0x804b1d8
0x804b1d8:      0x08049680      0x0804969a      0x080496b4      0x21776f57
0x804b1e8:      0x756f5920      0x20657627      0x75666564      0x20646573
0x804b1f8:      0x20656874      0x72636573      0x73207465      0x65676174
0x804b208:      0x00000021      0x79206f53      0x7420756f      0x6b6e6968
(gdb) q
A debugging session is active.
```

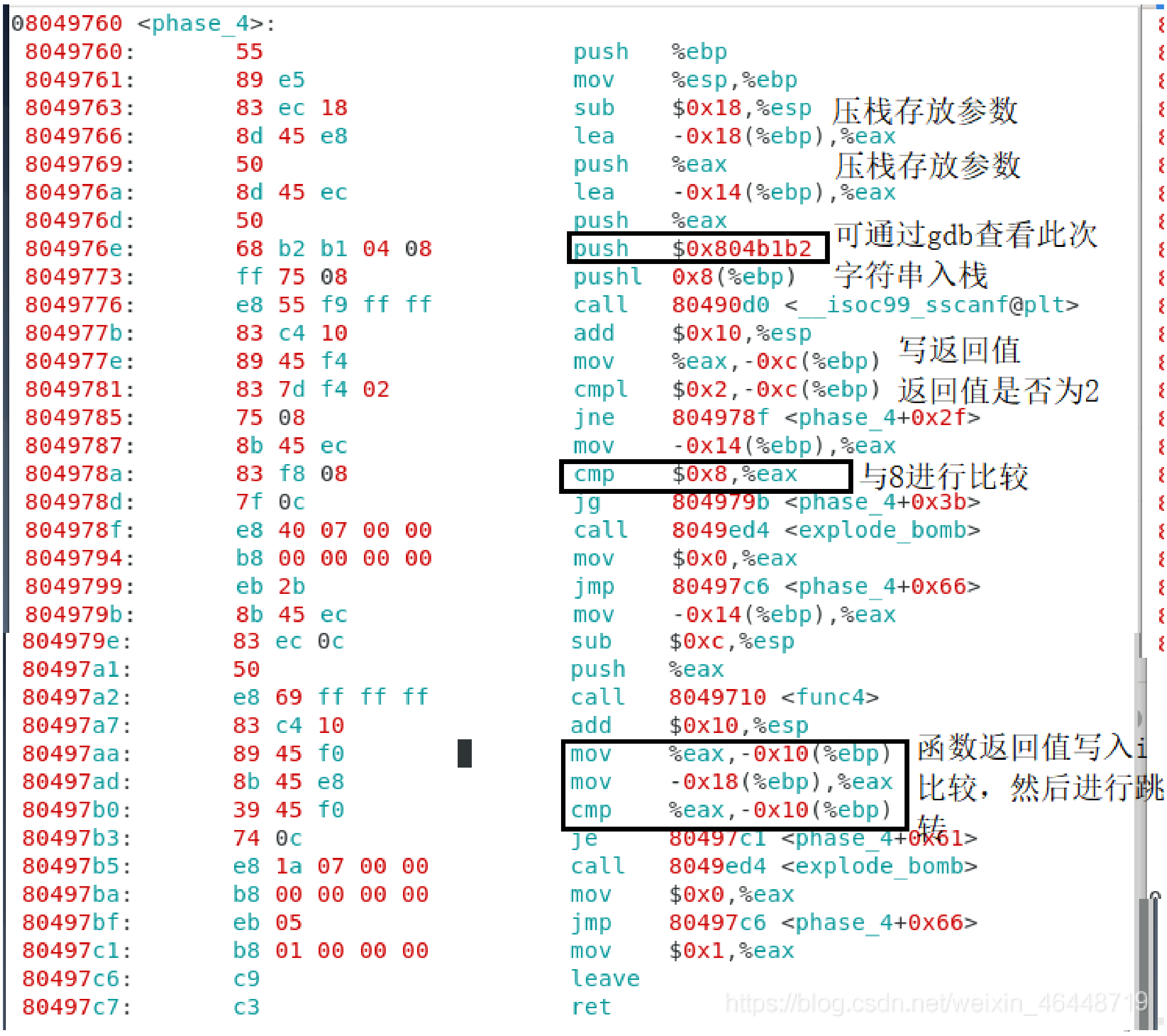
得到基址后返回在 bomb.s 中查看跳转通过计算转换得到后两个应输入e和142。
（注意看好自己的跳转！~）



```
linuxer@debian:~/Downloads$ ./bomb
Welcome to my fiendish little bomb. You have 7 phases with
which to blow yourself up. Have a nice day!
All I/O devices are modeled as files.
Well done! You seem to have warmed up!
15691 20024
Phase 1 defused. How about the next one?
17 35 43 60 81 111
That's number 2.  Keep going!
45 e 142
Halfway there!
```

实验五.递归调用和栈

phase_4



```
08049760 <phase_4>:
 8049760:       55                      push   %ebp
 8049761:       89 e5                   mov    %esp,%ebp
 8049763:       83 ec 18                sub    $0x18,%esp
 8049766:       8d 45 e8                lea    -0x18(%ebp),%eax
 8049769:       50                      push   %eax
 804976a:       8d 45 ec                lea    -0x14(%ebp),%eax
 804976d:       50                      push   %eax
 804976e:       68 b2 b1 04 08          push   $0x804b1b2
 8049773:       ff 75 08                pushl  0x8(%ebp)
 8049776:       e8 55 f9 ff ff          call   80490d0 <__isoc99_sscanf@plt>
 804977b:       83 c4 10                add    $0x10,%esp
 804977e:       89 45 f4                mov    %eax,-0xc(%ebp)
 8049781:       83 7d f4 02             cmpl   $0x2,-0xc(%ebp)
 8049785:       75 08                   jne    804978f <phase_4+0x2f>
 8049787:       8b 45 ec                mov    -0x14(%ebp),%eax
 804978a:       83 f8 08                cmp    $0x8,%eax
 804978d:       7f 0c                   jg     804979b <phase_4+0x3b>
 804978f:       e8 40 07 00 00          call   8049ed4 <explode_bomb>
 8049794:       b8 00 00 00 00          mov    $0x0,%eax
 8049799:       eb 2b                   jmp    80497c6 <phase_4+0x66>
 804979b:       8b 45 ec                mov    -0x14(%ebp),%eax
 804979e:       83 ec 0c                sub    $0xc,%esp
 80497a1:       50                      push   %eax
 80497a2:       e8 69 ff ff ff          call   8049710 <func4>
 80497a7:       83 c4 10                add    $0x10,%esp
 80497aa:       89 45 f0                mov    %eax,-0x10(%ebp)
 80497ad:       8b 45 e8                mov    -0x18(%ebp),%eax
 80497b0:       39 45 f0                cmp    %eax,-0x10(%ebp)
 80497b3:       74 0c                   je     80497c1 <phase_4+0x61>
 80497b5:       e8 1a 07 00 00          call   8049ed4 <explode_bomb>
 80497ba:       b8 00 00 00 00          mov    $0x0,%eax
 80497bf:       eb 05                   jmp    80497c6 <phase_4+0x66>
 80497c1:       b8 01 00 00 00          mov    $0x1,%eax
 80497c6:       c9                      leave
 80497c7:       c3                      ret
```

压栈存放参数

压栈存放参数

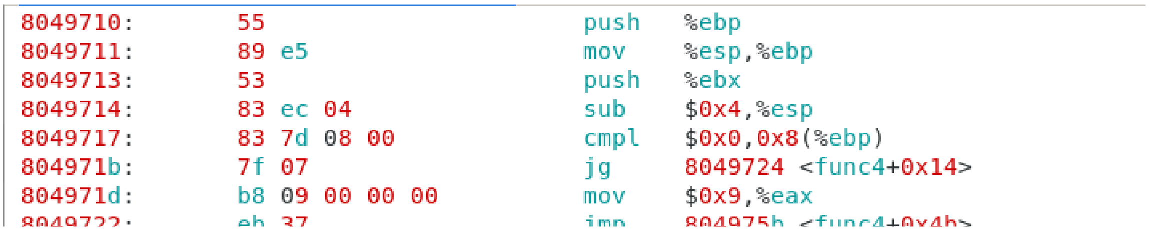
可通过gdb查看此次字符串入栈

写返回值

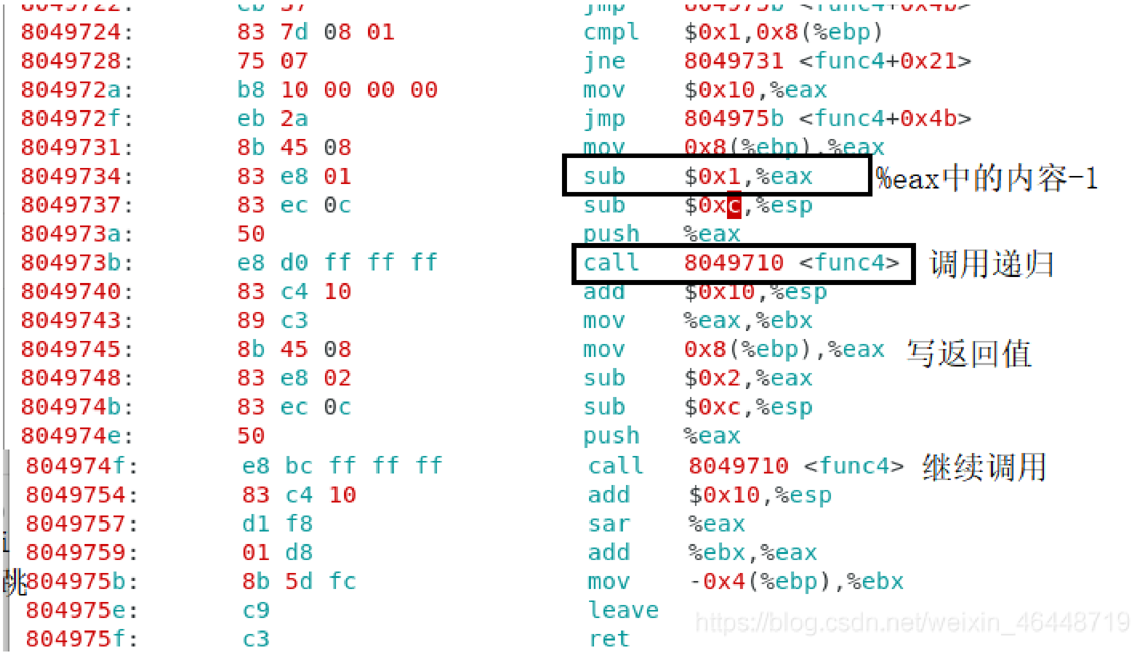
返回值是否为2

与8进行比较

函数返回值写入i比较，然后进行跳转



```
 8049710:       55                      push   %ebp
 8049711:       89 e5                   mov    %esp,%ebp
 8049713:       53                      push   %ebx
 8049714:       83 ec 04                sub    $0x4,%esp
 8049717:       83 7d 08 00             cmpl   $0x0,0x8(%ebp)
 804971b:       7f 07                   jg     8049724 <func4+0x14>
 804971d:       b8 09 00 00 00          mov    $0x9,%eax
 8049722:       eb 37                   jmp    804975b <func4+0x4b>
```



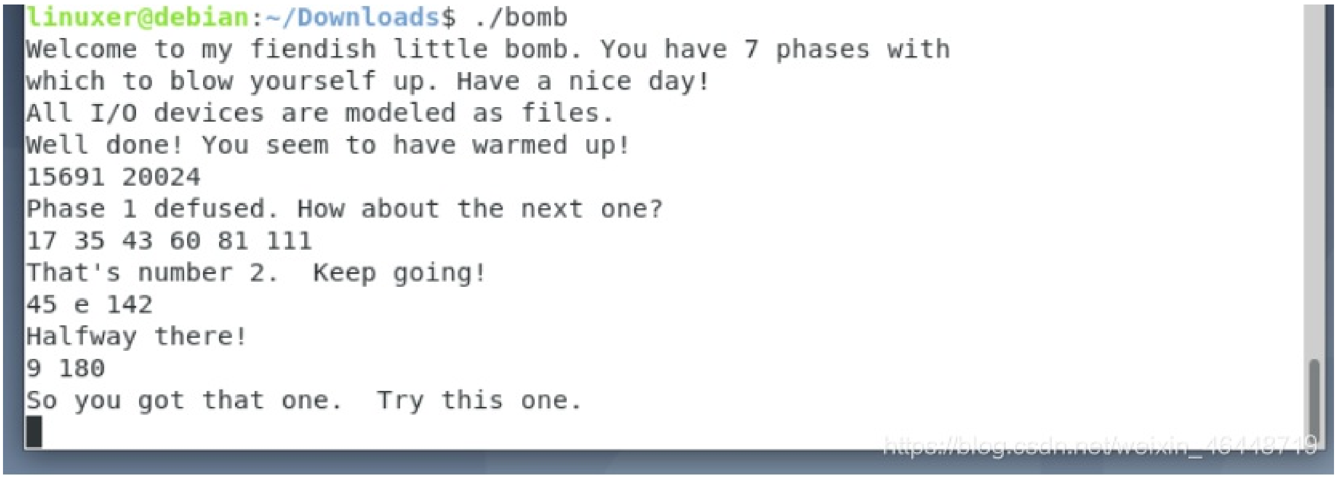
```
8049722:        eb 37                   jmp    804975b <func4+0x4b>
8049724:        83 7d 08 01             cmpl   $0x1,0x8(%ebp)
8049728:        75 07                   jne    8049731 <func4+0x21>
804972a:        b8 10 00 00 00          mov    $0x10,%eax
804972f:        eb 2a                   jmp    804975b <func4+0x4b>
8049731:        8b 45 08                mov    0x8(%ebp),%eax
8049734:        83 e8 01                sub    $0x1,%eax         %eax中的内容-1
8049737:        83 ec 0c                sub    $0xc,%esp
804973a:        50                      push   %eax
804973b:        e8 d0 ff ff ff          call   8049710 <func4>   调用递归
8049740:        83 c4 10                add    $0x10,%esp
8049743:        89 c3                   mov    %eax,%ebx
8049745:        8b 45 08                mov    0x8(%ebp),%eax    写返回值
8049748:        83 e8 02                sub    $0x2,%eax
804974b:        83 ec 0c                sub    $0xc,%esp
804974e:        50                      push   %eax
804974f:        e8 bc ff ff ff          call   8049710 <func4>   继续调用
8049754:        83 c4 10                add    $0x10,%esp
8049757:        d1 f8                   sar    %eax
8049759:        01 d8                   add    %ebx,%eax
804975b:        8b 5d fc                mov    -0x4(%ebp),%ebx
804975e:        c9                      leave
804975f:        c3                      ret
```

得到的递归条件为：

$$n_2 = f(n_1) = f(n_1-1)+(f(n_1-1)>>1)$$

$$f(x) = \begin{cases} 9, & x \leq 0 \\ 16, & x = 1 \\ f(x-1), & else \end{cases}$$

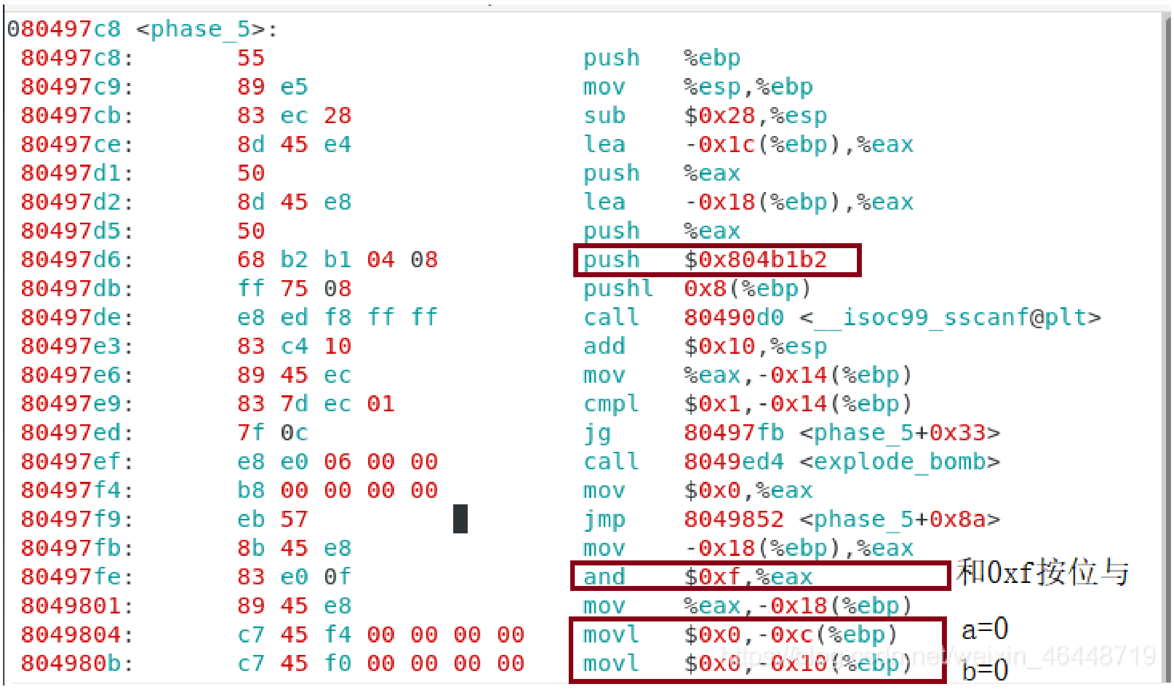
这一关，我做的是寂寞呀，感觉什么都没给大家解释清楚，其实~我是用 ida转换成伪代码从而找到递归条件的，实在是看不懂汇编代码了 ）



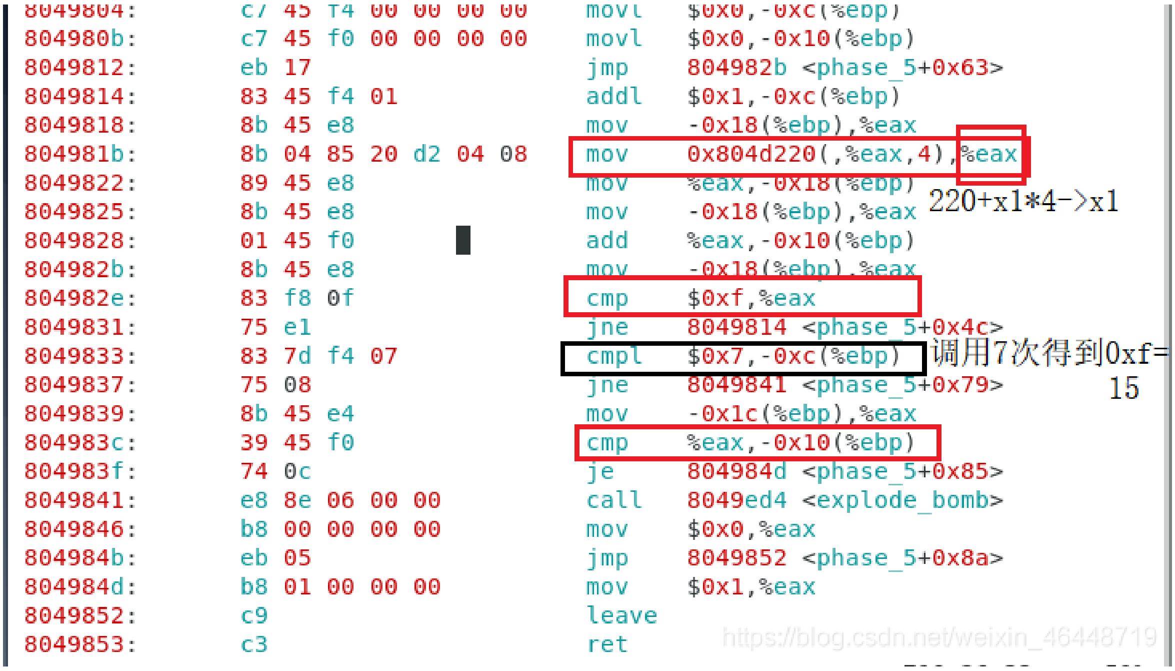
```
linuxer@debian:~/Downloads$ ./bomb
Welcome to my fiendish little bomb. You have 7 phases with
which to blow yourself up. Have a nice day!
All I/O devices are modeled as files.
Well done! You seem to have warmed up!
15691 20024
Phase 1 defused. How about the next one?
17 35 43 60 81 111
That's number 2.  Keep going!
45 e 142
Halfway there!
9 180
So you got that one.  Try this one.
```

继续吧，艰难的旅途...

实验六.指针



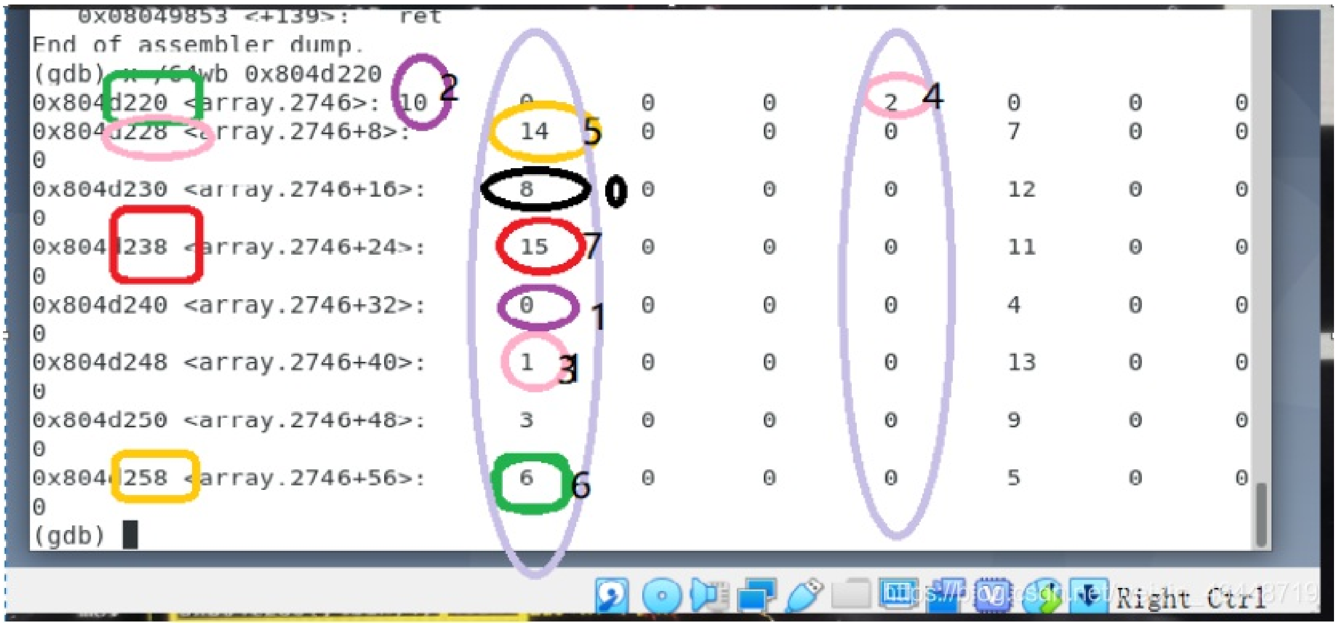
```
080497c8 <phase_5>:
 80497c8:	55                   	push   %ebp
 80497c9:	89 e5                	mov    %esp,%ebp
 80497cb:	83 ec 28             	sub    $0x28,%esp
 80497ce:	8d 45 e4             	lea    -0x1c(%ebp),%eax
 80497d1:	50                   	push   %eax
 80497d2:	8d 45 e8             	lea    -0x18(%ebp),%eax
 80497d5:	50                   	push   %eax
 80497d6:	68 b2 b1 04 08       	push   $0x804b1b2
 80497db:	ff 75 08             	pushl  0x8(%ebp)
 80497de:	e8 ed f8 ff ff       	call   80490d0 <__isoc99_sscanf@plt>
 80497e3:	83 c4 10             	add    $0x10,%esp
 80497e6:	89 45 ec             	mov    %eax,-0x14(%ebp)
 80497e9:	83 7d ec 01          	cmpl   $0x1,-0x14(%ebp)
 80497ed:	7f 0c                	jg     80497fb <phase_5+0x33>
 80497ef:	e8 e0 06 00 00       	call   8049ed4 <explode_bomb>
 80497f4:	b8 00 00 00 00       	mov    $0x0,%eax
 80497f9:	eb 57                	jmp    8049852 <phase_5+0x8a>
 80497fb:	8b 45 e8             	mov    -0x18(%ebp),%eax
 80497fe:	83 e0 0f             	and    $0xf,%eax
 8049801:	89 45 e8             	mov    %eax,-0x18(%ebp)
 8049804:	c7 45 f4 00 00 00 00 	movl   $0x0,-0xc(%ebp)
 804980b:	c7 45 f0 00 00 00 00 	movl   $0x0,-0x10(%ebp)
```



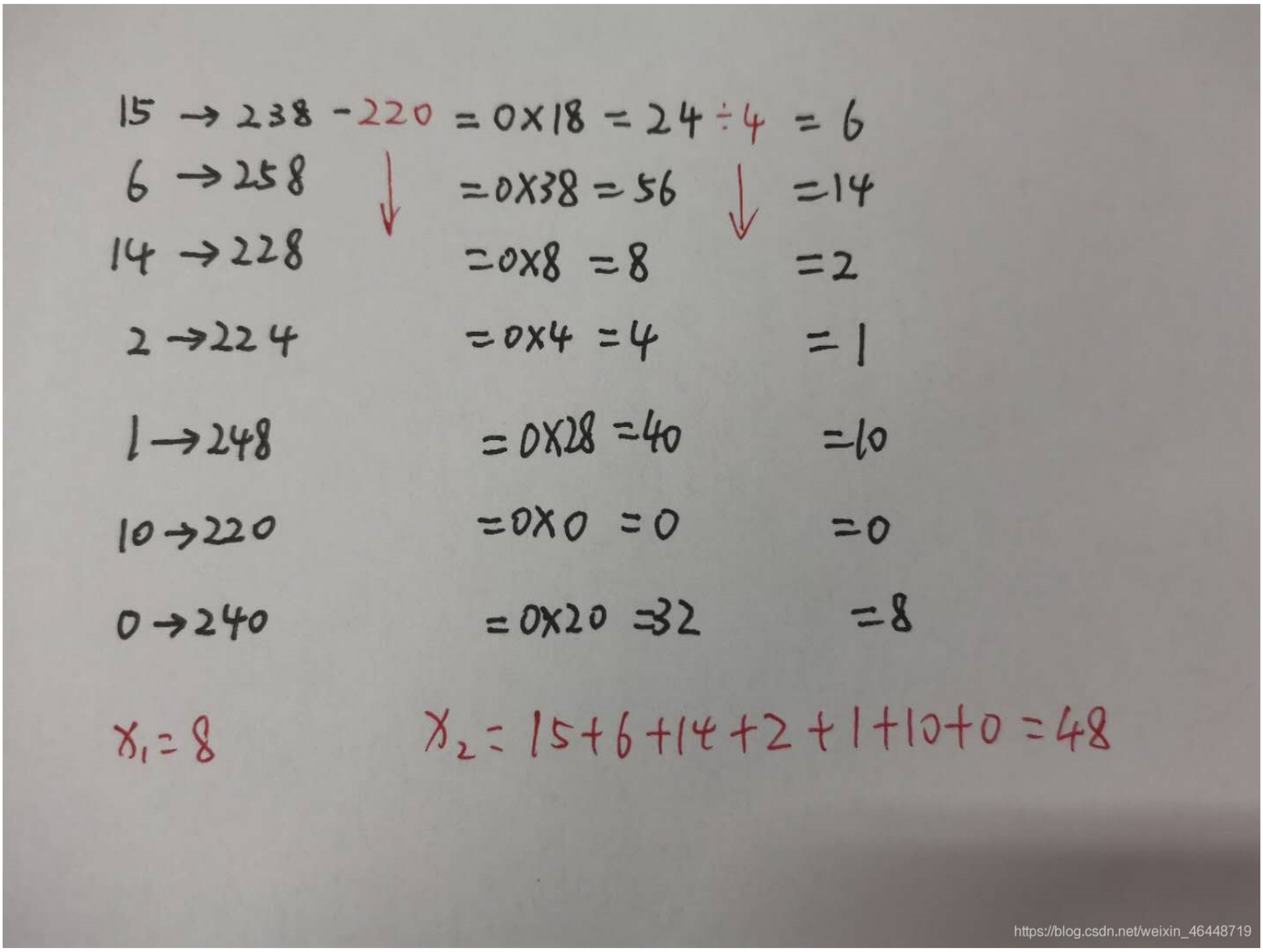
```
 8049804:	c7 45 f4 00 00 00 00 	movl   $0x0,-0xc(%ebp)
 804980b:	c7 45 f0 00 00 00 00 	movl   $0x0,-0x10(%ebp)
 8049812:	eb 17                	jmp    804982b <phase_5+0x63>
 8049814:	83 45 f4 01          	addl   $0x1,-0xc(%ebp)
 8049818:	8b 45 e8             	mov    -0x18(%ebp),%eax
 804981b:	8b 04 85 20 d2 04 08 	mov    0x804d220(,%eax,4),%eax
 8049822:	89 45 e8             	mov    %eax,-0x18(%ebp)
 8049825:	8b 45 e8             	mov    -0x18(%ebp),%eax
 8049828:	01 45 f0             	add    %eax,-0x10(%ebp)
 804982b:	8b 45 e8             	mov    -0x18(%ebp),%eax
 804982e:	83 f8 0f             	cmp    $0xf,%eax
 8049831:	75 e1                	jne    8049814 <phase_5+0x4c>
 8049833:	83 7d f4 07          	cmpl   $0x7,-0xc(%ebp)
 8049837:	75 08                	jne    8049841 <phase_5+0x79>
 8049839:	8b 45 e4             	mov    -0x1c(%ebp),%eax
 804983c:	39 45 f0             	cmp    %eax,-0x10(%ebp)
 804983f:	74 0c                	je     804984d <phase_5+0x85>
 8049841:	e8 8e 06 00 00       	call   8049ed4 <explode_bomb>
 8049846:	b8 00 00 00 00       	mov    $0x0,%eax
 804984b:	eb 05                	jmp    8049852 <phase_5+0x8a>
 804984d:	b8 01 00 00 00       	mov    $0x1,%eax
 8049852:	c9                   	leave
 8049853:	c3                   	ret
```

这个过程用了一个双循环，其主要条件就是与0xf按位与，通过找其他资料发现了明地址的作用，于是我找到了0x804d220这个

地址，用gdb调试



```
   0x08049853 <+139>:    ret
End of assembler dump.
(gdb) x /64wb 0x804d220
0x804d220 <array.2746>: 10      0       0       0       2       0       0       0
0x804d228 <array.2746+8>:       14      0       0       0       7       0       0
0
0x804d230 <array.2746+16>:      8       0       0       0       12      0       0
0
0x804d238 <array.2746+24>:      15      0       0       0       11      0       0
0
0x804d240 <array.2746+32>:      0       0       0       0       4       0       0
0
0x804d248 <array.2746+40>:      1       0       0       0       13      0       0
0
0x804d250 <array.2746+48>:      3       0       0       0       9       0       0
0
0x804d258 <array.2746+56>:      6       0       0       0       5       0       0
0
(gdb) 
```



15 → 238 −220 = 0x18 = 24 ÷4 = 6

6 → 258 ↓ = 0x38 = 56 ↓ = 14

14 → 228 = 0x8 = 8 = 2

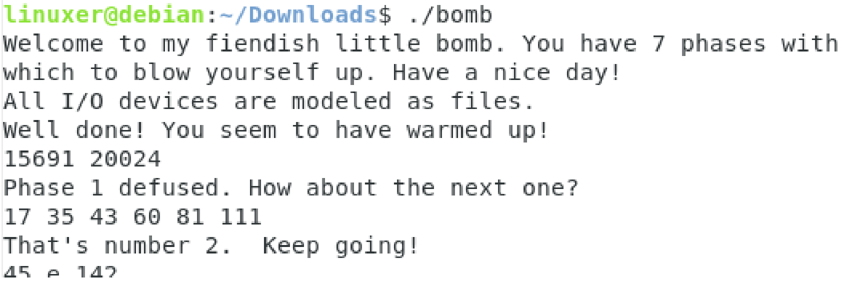
2 → 224 = 0x4 = 4 = 1

1 → 248 = 0x28 = 40 = 10

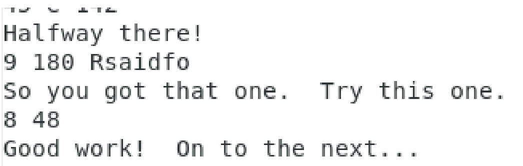
10 → 220 = 0x0 = 0 = 0

0 → 240 = 0x20 = 32 = 8

$x_1 = 8$ $x_2 = 15+6+14+2+1+10+0 = 48$



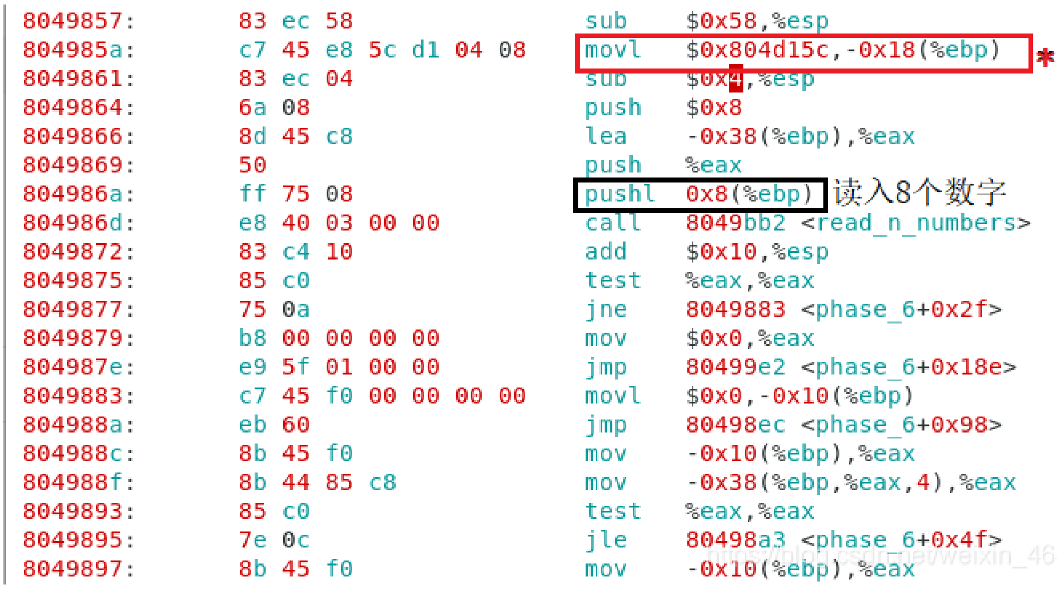
```
linuxer@debian:~/Downloads$ ./bomb
Welcome to my fiendish little bomb. You have 7 phases with
which to blow yourself up. Have a nice day!
All I/O devices are modeled as files.
Well done! You seem to have warmed up!
15691 20024
Phase 1 defused. How about the next one?
17 35 43 60 81 111
That's number 2.  Keep going!
45 e 142
```



```
Halfway there!
9 180 Rsaidfo
So you got that one.  Try this one.
8 48
Good work!  On to the next...
```

实验七.链表

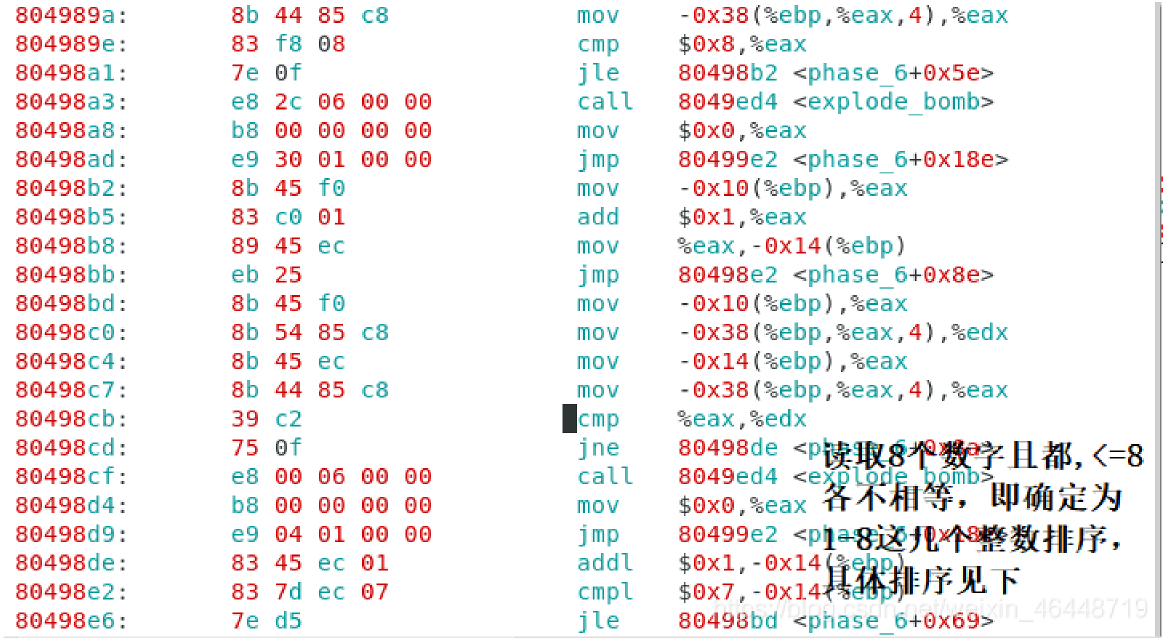
phase_6



```
8049857:       83 ec 58                sub    $0x58,%esp
804985a:       c7 45 e8 5c d1 04 08    movl   $0x804d15c,-0x18(%ebp)
8049861:       83 ec 04                sub    $0x4,%esp
8049864:       6a 08                   push   $0x8
8049866:       8d 45 c8                lea    -0x38(%ebp),%eax
8049869:       50                      push   %eax
804986a:       ff 75 08                pushl  0x8(%ebp)
804986d:       e8 40 03 00 00          call   8049bb2 <read_n_numbers>
8049872:       83 c4 10                add    $0x10,%esp
8049875:       85 c0                   test   %eax,%eax
8049877:       75 0a                   jne    8049883 <phase_6+0x2f>
8049879:       b8 00 00 00 00          mov    $0x0,%eax
804987e:       e9 5f 01 00 00          jmp    80499e2 <phase_6+0x18e>
8049883:       c7 45 f0 00 00 00 00    movl   $0x0,-0x10(%ebp)
804988a:       eb 60                   jmp    80498ec <phase_6+0x98>
804988c:       8b 45 f0                mov    -0x10(%ebp),%eax
804988f:       8b 44 85 c8             mov    -0x38(%ebp,%eax,4),%eax
8049893:       85 c0                   test   %eax,%eax
8049895:       7e 0c                   jle    80498a3 <phase_6+0x4f>
8049897:       8b 45 f0                mov    -0x10(%ebp),%eax
```

*

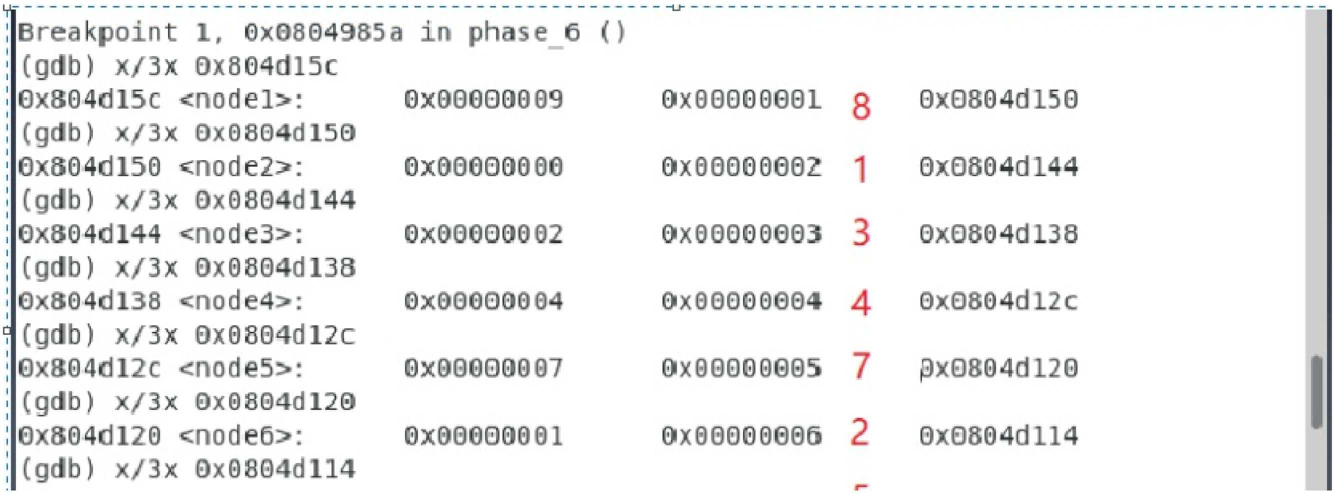
读入8个数字



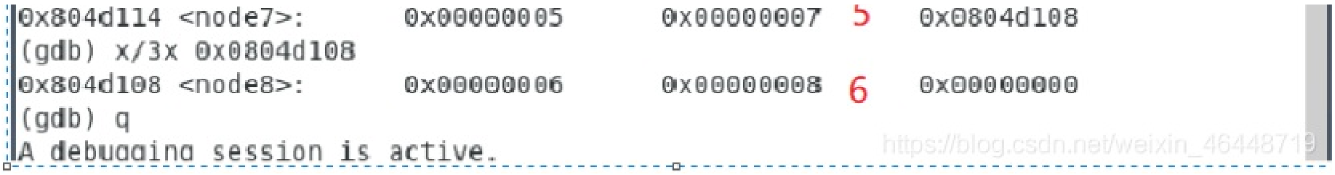
```
804989a:       8b 44 85 c8             mov    -0x38(%ebp,%eax,4),%eax
804989e:       83 f8 08                cmp    $0x8,%eax
80498a1:       7e 0f                   jle    80498b2 <phase_6+0x5e>
80498a3:       e8 2c 06 00 00          call   8049ed4 <explode_bomb>
80498a8:       b8 00 00 00 00          mov    $0x0,%eax
80498ad:       e9 30 01 00 00          jmp    80499e2 <phase_6+0x18e>
80498b2:       8b 45 f0                mov    -0x10(%ebp),%eax
80498b5:       83 c0 01                add    $0x1,%eax
80498b8:       89 45 ec                mov    %eax,-0x14(%ebp)
80498bb:       eb 25                   jmp    80498e2 <phase_6+0x8e>
80498bd:       8b 45 f0                mov    -0x10(%ebp),%eax
80498c0:       8b 54 85 c8             mov    -0x38(%ebp,%eax,4),%edx
80498c4:       8b 45 ec                mov    -0x14(%ebp),%eax
80498c7:       8b 44 85 c8             mov    -0x38(%ebp,%eax,4),%eax
80498cb:       39 c2                   cmp    %eax,%edx
80498cd:       75 0f                   jne    80498de <phase_6+0x8a>
80498cf:       e8 00 06 00 00          call   8049ed4 <explode_bomb>
80498d4:       b8 00 00 00 00          mov    $0x0,%eax
80498d9:       e9 04 01 00 00          jmp    80499e2 <phase_6+0x18e>
80498de:       83 45 ec 01             addl   $0x1,-0x14(%ebp)
80498e2:       83 7d ec 07             cmpl   $0x7,-0x14(%ebp)
80498e6:       7e d5                   jle    80498bd <phase_6+0x69>
```

读取8个数字且都,<=8各不相等，即确定为1-8这几个整数排序，具体排序见下

在此关中，我的代码发现最终输出的结果是按照降序排列数字然后输出：（有的是升序，有的需要用9减有的不需要）

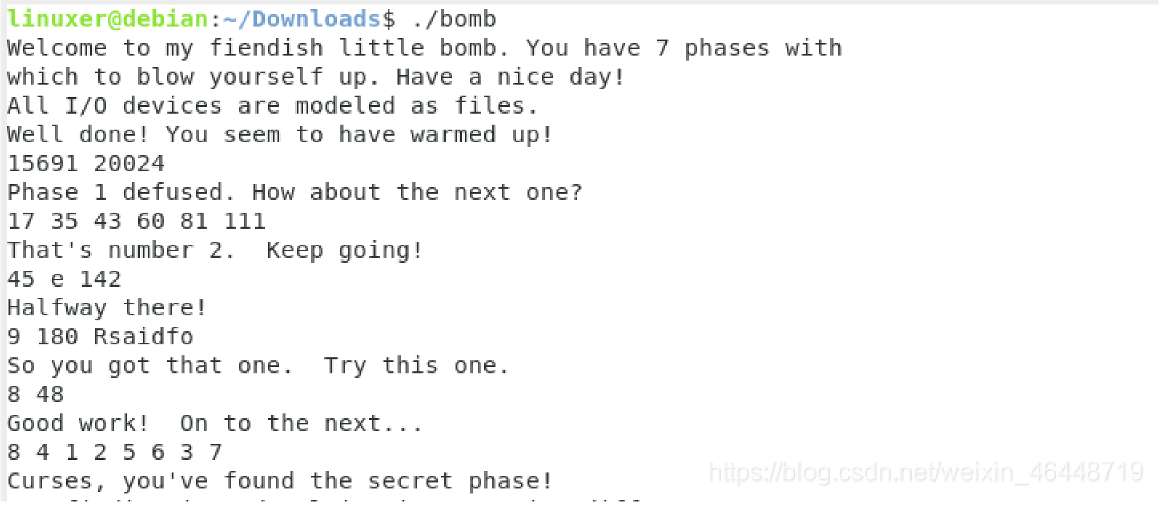


```
Breakpoint 1, 0x0804985a in phase_6 ()
(gdb) x/3x 0x804d15c
0x804d15c <node1>:      0x00000009      0x00000001  8    0x0804d150
(gdb) x/3x 0x0804d150
0x804d150 <node2>:      0x00000000      0x00000002  1    0x0804d144
(gdb) x/3x 0x0804d144
0x804d144 <node3>:      0x00000002      0x00000003  3    0x0804d138
(gdb) x/3x 0x0804d138
0x804d138 <node4>:      0x00000004      0x00000004  4    0x0804d12c
(gdb) x/3x 0x0804d12c
0x804d12c <node5>:      0x00000007      0x00000005  7    0x0804d120
(gdb) x/3x 0x0804d120
0x804d120 <node6>:      0x00000001      0x00000006  2    0x0804d114
(gdb) x/3x 0x0804d114
```



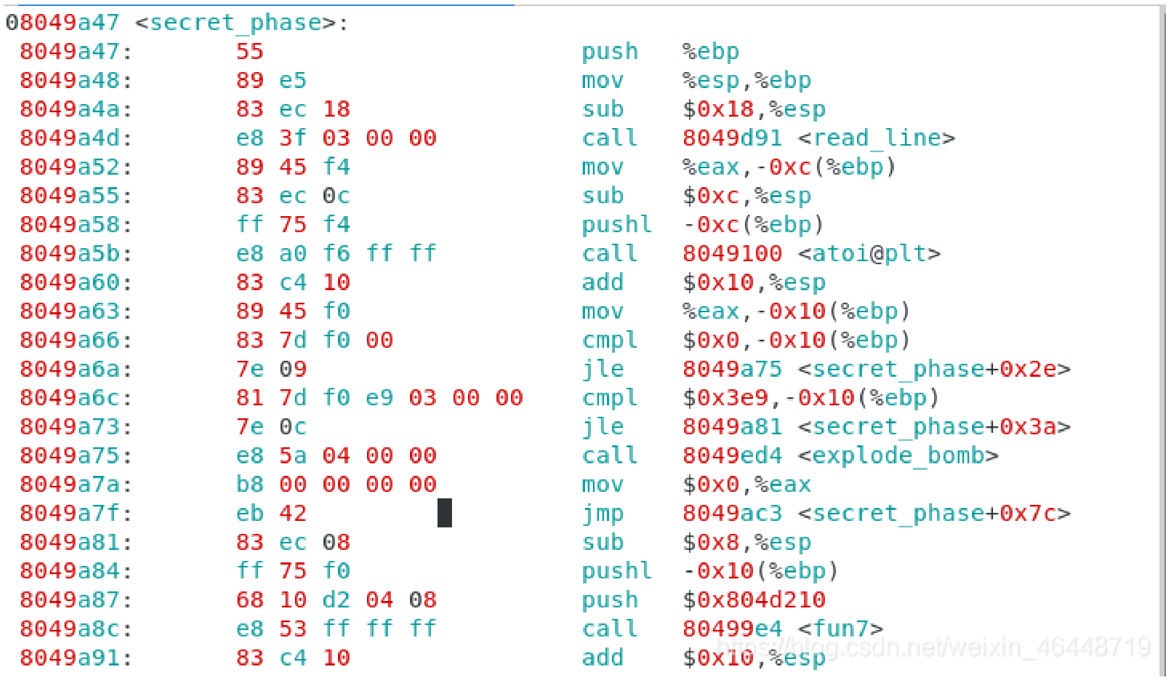
```
0x804d114 <node7>:      0x00000005      0x00000007  5   0x0804d108
(gdb) x/3x 0x0804d108
0x804d108 <node8>:      0x00000006      0x00000008  6   0x00000000
(gdb) q
A debugging session is active.
```

降序排列为：1 5 8 7 4 3 6 2，用9减去之后为：8 4 1 2 5 6 3 7

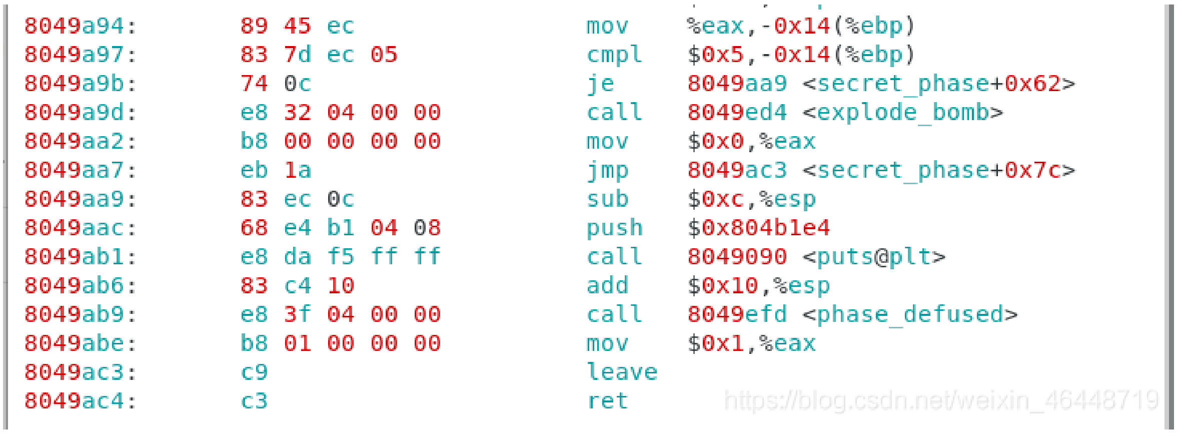


```
linuxer@debian:~/Downloads$ ./bomb
Welcome to my fiendish little bomb. You have 7 phases with
which to blow yourself up. Have a nice day!
All I/O devices are modeled as files.
Well done! You seem to have warmed up!
15691 20024
Phase 1 defused. How about the next one?
17 35 43 60 81 111
That's number 2.  Keep going!
45 e 142
Halfway there!
9 180 Rsaidfo
So you got that one.  Try this one.
8 48
Good work!  On to the next...
8 4 1 2 5 6 3 7
Curses, you've found the secret phase!
```

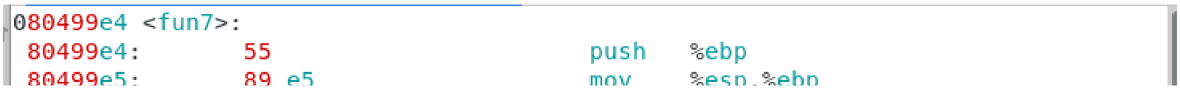
到这里，正式关卡已经结束，最后一关为隐藏关，主要就是先要找到通关密码：



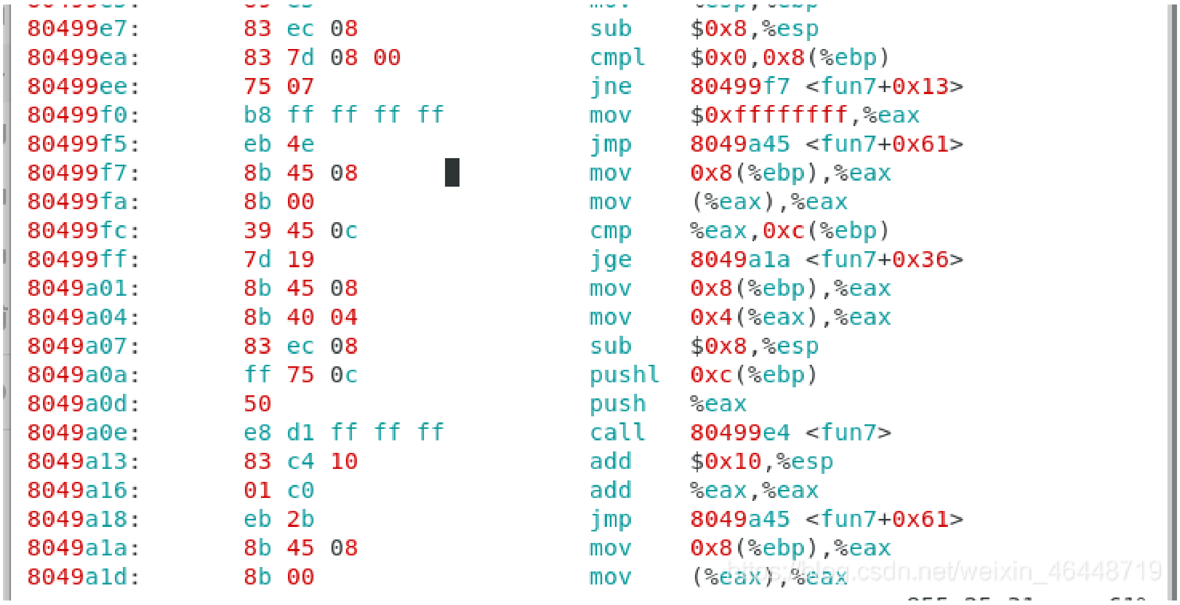
```
08049a47 <secret_phase>:
 8049a47:       55                      push   %ebp
 8049a48:       89 e5                   mov    %esp,%ebp
 8049a4a:       83 ec 18                sub    $0x18,%esp
 8049a4d:       e8 3f 03 00 00          call   8049d91 <read_line>
 8049a52:       89 45 f4                mov    %eax,-0xc(%ebp)
 8049a55:       83 ec 0c                sub    $0xc,%esp
 8049a58:       ff 75 f4                pushl  -0xc(%ebp)
 8049a5b:       e8 a0 f6 ff ff          call   8049100 <atoi@plt>
 8049a60:       83 c4 10                add    $0x10,%esp
 8049a63:       89 45 f0                mov    %eax,-0x10(%ebp)
 8049a66:       83 7d f0 00             cmpl   $0x0,-0x10(%ebp)
 8049a6a:       7e 09                   jle    8049a75 <secret_phase+0x2e>
 8049a6c:       81 7d f0 e9 03 00 00    cmpl   $0x3e9,-0x10(%ebp)
 8049a73:       7e 0c                   jle    8049a81 <secret_phase+0x3a>
 8049a75:       e8 5a 04 00 00          call   8049ed4 <explode_bomb>
 8049a7a:       b8 00 00 00 00          mov    $0x0,%eax
 8049a7f:       eb 42                   jmp    8049ac3 <secret_phase+0x7c>
 8049a81:       83 ec 08                sub    $0x8,%esp
 8049a84:       ff 75 f0                pushl  -0x10(%ebp)
 8049a87:       68 10 d2 04 08          push   $0x804d210
 8049a8c:       e8 53 ff ff ff          call   80499e4 <fun7>
 8049a91:       83 c4 10                add    $0x10,%esp
```



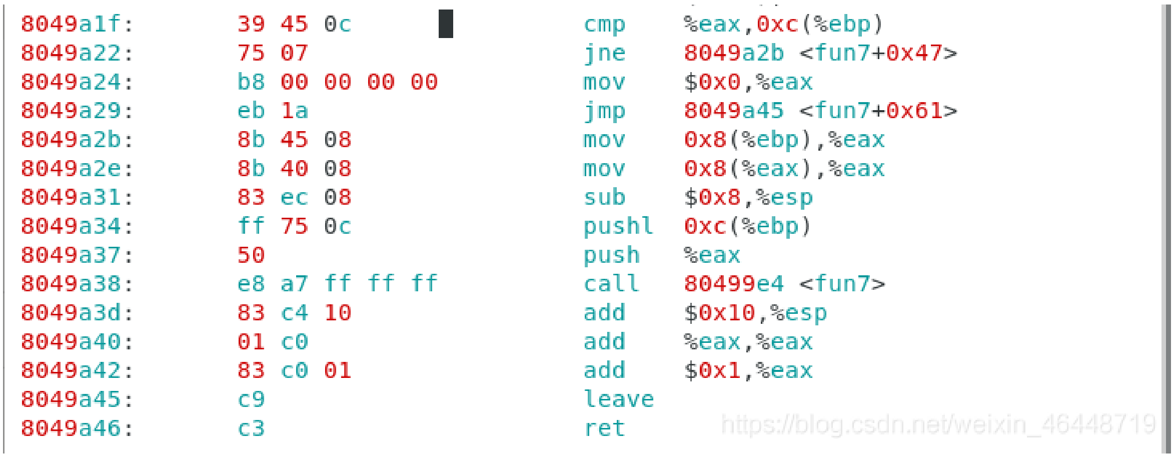
```
 8049a94:       89 45 ec                mov    %eax,-0x14(%ebp)
 8049a97:       83 7d ec 05             cmpl   $0x5,-0x14(%ebp)
 8049a9b:       74 0c                   je     8049aa9 <secret_phase+0x62>
 8049a9d:       e8 32 04 00 00          call   8049ed4 <explode_bomb>
 8049aa2:       b8 00 00 00 00          mov    $0x0,%eax
 8049aa7:       eb 1a                   jmp    8049ac3 <secret_phase+0x7c>
 8049aa9:       83 ec 0c                sub    $0xc,%esp
 8049aac:       68 e4 b1 04 08          push   $0x804b1e4
 8049ab1:       e8 da f5 ff ff          call   8049090 <puts@plt>
 8049ab6:       83 c4 10                add    $0x10,%esp
 8049ab9:       e8 3f 04 00 00          call   8049efd <phase_defused>
 8049abe:       b8 01 00 00 00          mov    $0x1,%eax
 8049ac3:       c9                      leave
 8049ac4:       c3                      ret
```



```
080499e4 <fun7>:
 80499e4:       55                      push   %ebp
 80499e5:       89 e5                   mov    %esp,%ebp
```

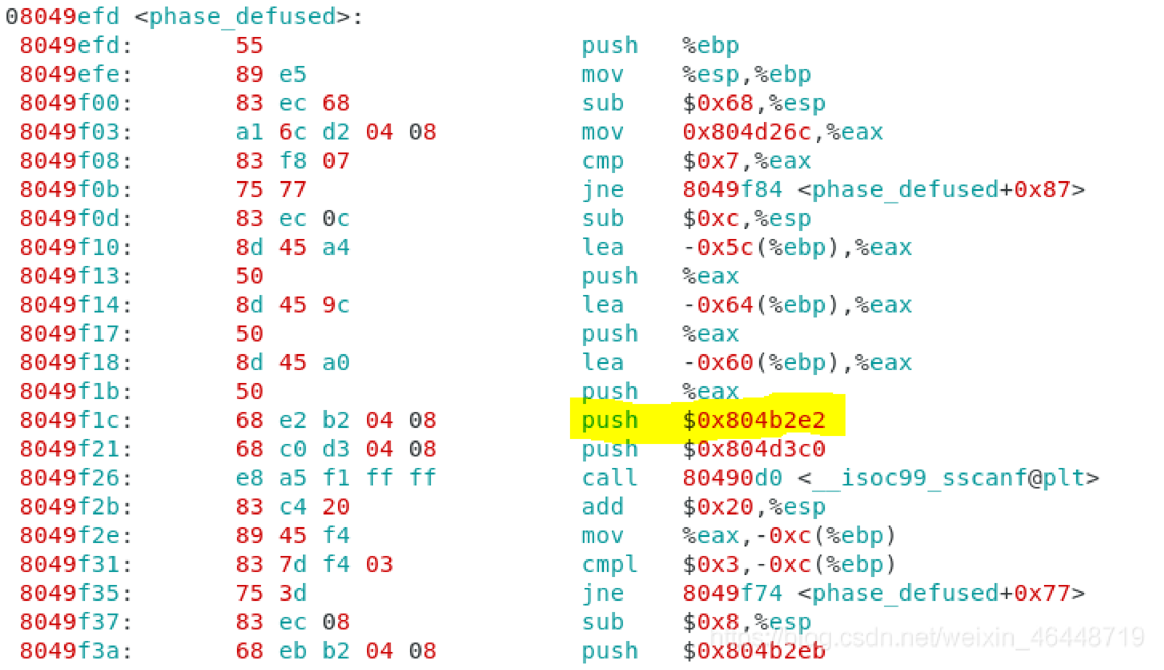


```
80499e7:	83 ec 08             	sub    $0x8,%esp
80499ea:	83 7d 08 00          	cmpl   $0x0,0x8(%ebp)
80499ee:	75 07                	jne    80499f7 <fun7+0x13>
80499f0:	b8 ff ff ff ff       	mov    $0xffffffff,%eax
80499f5:	eb 4e                	jmp    8049a45 <fun7+0x61>
80499f7:	8b 45 08             	mov    0x8(%ebp),%eax
80499fa:	8b 00                	mov    (%eax),%eax
80499fc:	39 45 0c             	cmp    %eax,0xc(%ebp)
80499ff:	7d 19                	jge    8049a1a <fun7+0x36>
8049a01:	8b 45 08             	mov    0x8(%ebp),%eax
8049a04:	8b 40 04             	mov    0x4(%eax),%eax
8049a07:	83 ec 08             	sub    $0x8,%esp
8049a0a:	ff 75 0c             	pushl  0xc(%ebp)
8049a0d:	50                   	push   %eax
8049a0e:	e8 d1 ff ff ff       	call   80499e4 <fun7>
8049a13:	83 c4 10             	add    $0x10,%esp
8049a16:	01 c0                	add    %eax,%eax
8049a18:	eb 2b                	jmp    8049a45 <fun7+0x61>
8049a1a:	8b 45 08             	mov    0x8(%ebp),%eax
8049a1d:	8b 00                	mov    (%eax),%eax
```

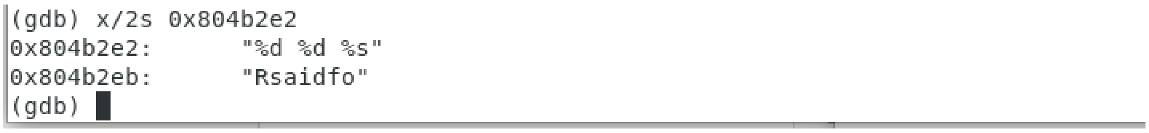


```
8049a1f:	39 45 0c             	cmp    %eax,0xc(%ebp)
8049a22:	75 07                	jne    8049a2b <fun7+0x47>
8049a24:	b8 00 00 00 00       	mov    $0x0,%eax
8049a29:	eb 1a                	jmp    8049a45 <fun7+0x61>
8049a2b:	8b 45 08             	mov    0x8(%ebp),%eax
8049a2e:	8b 40 08             	mov    0x8(%eax),%eax
8049a31:	83 ec 08             	sub    $0x8,%esp
8049a34:	ff 75 0c             	pushl  0xc(%ebp)
8049a37:	50                   	push   %eax
8049a38:	e8 a7 ff ff ff       	call   80499e4 <fun7>
8049a3d:	83 c4 10             	add    $0x10,%esp
8049a40:	01 c0                	add    %eax,%eax
8049a42:	83 c0 01             	add    $0x1,%eax
8049a45:	c9                   	leave
8049a46:	c3                   	ret
```

对隐藏关中的汇编进行分析，然后再通过 gdb 查看0804b2e2中得内容，最终可得知隐藏关 的入口在 phase_4,密码为"Rsaidfo"



```
08049efd <phase_defused>:
 8049efd:	55                   	push   %ebp
 8049efe:	89 e5                	mov    %esp,%ebp
 8049f00:	83 ec 68             	sub    $0x68,%esp
 8049f03:	a1 6c d2 04 08       	mov    0x804d26c,%eax
 8049f08:	83 f8 07             	cmp    $0x7,%eax
 8049f0b:	75 77                	jne    8049f84 <phase_defused+0x87>
 8049f0d:	83 ec 0c             	sub    $0xc,%esp
 8049f10:	8d 45 a4             	lea    -0x5c(%ebp),%eax
 8049f13:	50                   	push   %eax
 8049f14:	8d 45 9c             	lea    -0x64(%ebp),%eax
 8049f17:	50                   	push   %eax
 8049f18:	8d 45 a0             	lea    -0x60(%ebp),%eax
 8049f1b:	50                   	push   %eax
 8049f1c:	68 e2 b2 04 08       	push   $0x804b2e2
 8049f21:	68 c0 d3 04 08       	push   $0x804d3c0
 8049f26:	e8 a5 f1 ff ff       	call   80490d0 <__isoc99_sscanf@plt>
 8049f2b:	83 c4 20             	add    $0x20,%esp
 8049f2e:	89 45 f4             	mov    %eax,-0xc(%ebp)
 8049f31:	83 7d f4 03          	cmpl   $0x3,-0xc(%ebp)
 8049f35:	75 3d                	jne    8049f74 <phase_defused+0x77>
 8049f37:	83 ec 08             	sub    $0x8,%esp
 8049f3a:	68 eb b2 04 08       	push   $0x804b2eb
```



```
(gdb) x/2s 0x804b2e2
0x804b2e2:	"%d %d %s"
0x804b2eb:	"Rsaidfo"
(gdb) 
```

呼~密码找到了，此时又是想偷懒的刘渣渣，具体破解方法我的隐藏关和张大大完全相同，哈哈哈，不厚道的附上链接

链接: link.

真省事呦呼呼，
结束啦~~