Bomblab Discussion

刘沛雨

Lab解读

- 需要修改的文件：无
- 需要解决的问题：发现并拆除炸弹
- 如何开始?

```
tar  -xvf  bomb651.tar
cd ./bomb
objdump -d bomb > bomb.txt        ctrl/command+f
gdb bomb
```

gdb 调试

```
gdb bomb

# 获取帮助
help

# 设置断点
break explode_bomb
break phase_1

# 开始运行
run

# 检查汇编 会给出对应的代码的汇编
disas

# 查看寄存器内容
info registers

# 打印指定寄存器
print $rsp

# 每步执行
stepi
```

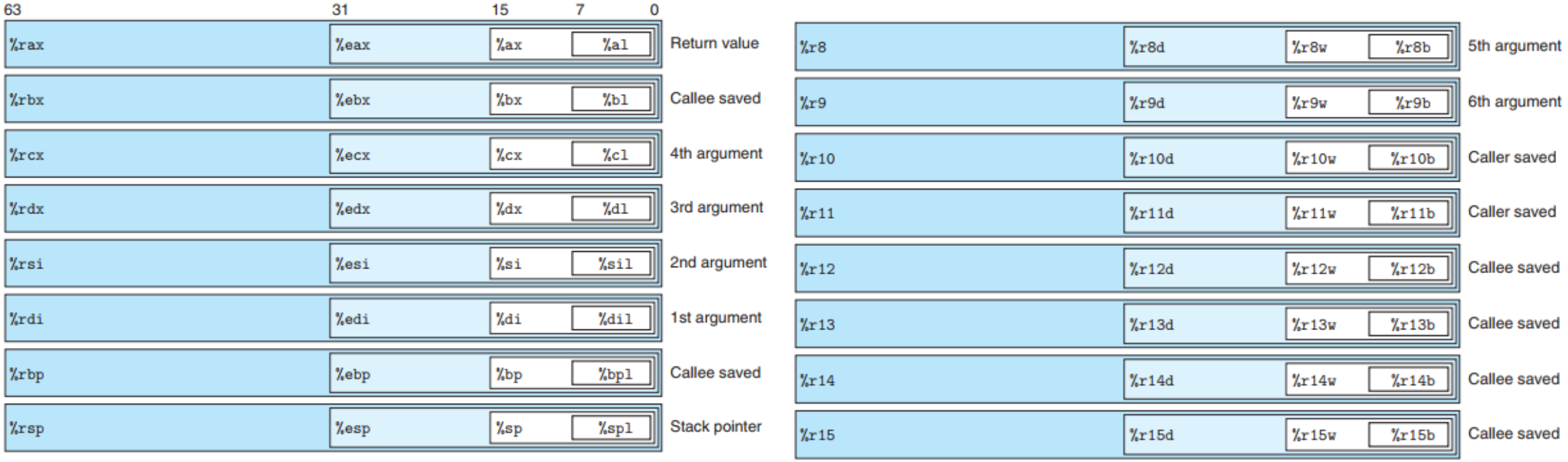
一条好用的指令：**display/i $pc**

显示下一步将要执行的指令

main 函数中的信息

```
/* Hmm...  Six phases must be more secure than one phase! */
input = read_line();             /* Get input                    */
phase_1(input);                  /* Run the phase                */
phase_defused(fp);                  /* Drat!  They figured it out!
                                 * Let me know how they did it. */
printf("Phase 1 defused. How about the next one?\n");
```

其他需要的信息


63
31
15
7
0
%rax
%eax
%ax
%al
Return value
%rbx
%ebx
%bx
%bl
Callee saved
%rcx
%ecx
%cx
%cl
4th argument
%rdx
%edx
%dx
%dl
3rd argument
%rsi
%esi
%si
%sil
2nd argument
%rdi
%edi
%di
%dil
1st argument
%rbp
%ebp
%bp
%bpl
Callee saved
%rsp
%esp
%sp
%spl
Stack pointer
%r8
%r8d
%r8w
%r8b
5th argument
%r9
%r9d
%r9w
%r9b
6th argument
%r10
%r10d
%r10w
%r10b
Caller saved
%r11
%r11d
%r11w
%r11b
Caller saved
%r12
%r12d
%r12w
%r12b
Callee saved
%r13
%r13d
%r13w
%r13b
Callee saved
%r14
%r14d
%r14w
%r14b
Callee saved
%r15
%r15d
%r15w
%r15b
Callee saved

phase_1

String Search

解题思路

- 从内存中找到目标字符串
- 与输入字符串比较
- 若二者相同则拆除炸弹，否则炸弹爆炸
- 因此输入字符串应当与内存中存放的目标字符串相同

将一个内存地址送入了%rsi

```
0000000000001798 <phase_1>:
    1798:   f3 0f 1e fa             endbr64
    179c:   48 83 ec 08             sub    $0x8,%rsp
    17a0:   48 8d 35 59 2a 00 00    lea    0x2a59(%rip),%rsi        # 4200 <_IO_stdin_used+0x200>
    17a7:   e8 34 05 00 00          call   1ce0 <strings_not_equal>
    17ac:   85 c0                   test   %eax,%eax
    17ae:   75 05                   jne    17b5 <phase_1+0x1d>
    17b0:   48 83 c4 08             add    $0x8,%rsp
    17b4:   c3                      ret
    17b5:   e8 89 08 00 00          call   2043 <explode_bomb>
    17ba:   eb f4                   jmp    17b0 <phase_1+0x18>
```

```
(gdb) stepi
0x0000563c91ec47a7 in phase_1 ()
1: x/i $pc
=> 0x563c91ec47a7 <phase_1+15>: call   0x563c91ec4ce0 <strings_not_equal>
(gdb) print (char*)($rsi)
$1 = 0x563c91ec7200 "Playing games without playing Genshin Impact is like reading Four Classics without reading Dream of Red Mansions."
```

%rsi=&answer_string

%rsi=&answer_string

将一个内存地址送入了%rsi

```
0000000000001798 <phase_1>:
    1798:   f3 0f 1e fa             endbr64
    179c:   48 83 ec 08             sub    $0x8,%rsp
    17a0:   48 8d 35 59 2a 00 00    lea    0x2a59(%rip),%rsi        # 4200 <_IO_stdin_used+0x200>
    17a7:   e8 34 05 00 00          call   1ce0 <strings_not_equal>
    17ac:   85 c0                   test   %eax,%eax
    17ae:   75 05                   jne    17b5 <phase_1+0x1d>
    17b0:   48 83 c4 08             add    $0x8,%rsp
    17b4:   c3                      ret
    17b5:   e8 89 08 00 00          call   2043 <explode_bomb>
    17ba:   eb f4                   jmp    17b0 <phase_1+0x18>
```

```
Welcome to Dr. Evil's little bomb. You have 6 phases with
which to blow yourself up. Have a nice day! Mua ha ha ha!
string for test.

Breakpoint 2, 0x000055e501fc7798 in phase_1 ()
1: x/i $pc
=> 0x55e501fc7798 <phase_1>:    endbr64
(gdb) print (char*)($rdi)
$4 = 0x55e501fce560 <input_strings> "string for test."
```

%rdi=&input_string

```
0000000000001ce0 <strings_not_equal>:
    1ce0:   f3 0f 1e fa             endbr64
    1ce4:   41 54                   push   %r12
    1ce6:   55                      push   %rbp
    1ce7:   53                      push   %rbx
    1ce8:   48 89 fb                mov    %rdi,%rbx
    1ceb:   48 89 f5                mov    %rsi,%rbp
    1cee:   e8 d5 ff ff ff          call   1cc8 <string_length>
    1cf3:   41 89 c4                mov    %eax,%r12d
    1cf6:   48 89 ef                mov    %rbp,%rdi
    1cf9:   e8 ca ff ff ff          call   1cc8 <string_length>
    1cfe:   41 39 c4                cmp    %eax,%r12d                #比较两个字符串长度
    1d01:   75 1d                   jne    1d20 <strings_not_equal+0x40>
    1d03:   0f b6 03                movzbl (%rbx),%eax
    1d06:   84 c0                   test   %al,%al
    1d08:   74 0f                   je     1d19 <strings_not_equal+0x39>   #第一个输入的串为空串，而两串长度相同，故两串相同（均为空串）
    1d0a:   38 45 00                cmp    %al,0x0(%rbp)             #比较两个串的第i个字节
    1d0d:   75 1b                   jne    1d2a <strings_not_equal+0x4a>
    1d0f:   48 83 c3 01             add    $0x1,%rbx
    1d13:   48 83 c5 01             add    $0x1,%rbp
    1d17:   eb ea                   jmp    1d03 <strings_not_equal+0x23>   #循环，逐字节比较
    1d19:   b8 00 00 00 00          mov    $0x0,%eax                 #两串相同，返回0
    1d1e:   eb 05                   jmp    1d25 <strings_not_equal+0x45>
    1d20:   b8 01 00 00 00          mov    $0x1,%eax                 #两串不同，返回1
    1d25:   5b                      pop    %rbx
    1d26:   5d                      pop    %rbp
    1d27:   41 5c                   pop    %r12
    1d29:   c3                      ret
    1d2a:   b8 01 00 00 00          mov    $0x1,%eax                 #两串不同，返回1
    1d2f:   eb f4                   jmp    1d25 <strings_not_equal+0x45>
```

写代码时对变量和函数进行清晰有意义地命名可以很好地提升代码可读性

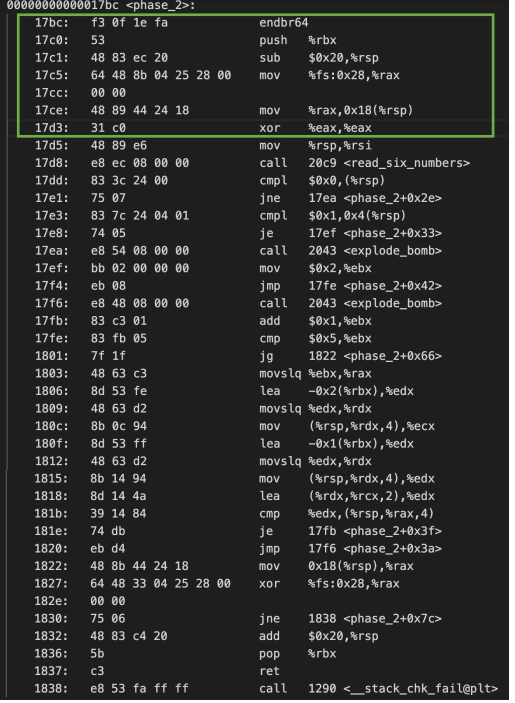
phase_2

Array and Stack

解题思路

- read_six_numbers 将输入的字符串格式化为6个整数，并将这6个整数**存放在在一个临时数组变量中**
- 通过汇编代码发现栈顶的前2个整数分别为0和1
- 汇编代码通过循环递推计算出后续4个整数值1 3 5 11
- 从而我们输入的字符串为0 1 1 3 5 11

```
00000000000017bc <phase_2>:
    17bc:   f3 0f 1e fa             endbr64
    17c0:   53                      push   %rbx
    17c1:   48 83 ec 20             sub    $0x20,%rsp
    17c5:   64 48 8b 04 25 28 00    mov    %fs:0x28,%rax
    17cc:   00 00
    17ce:   48 89 44 24 18          mov    %rax,0x18(%rsp)
    17d3:   31 c0                   xor    %eax,%eax
    17d5:   48 89 e6                mov    %rsp,%rsi
    17d8:   e8 ec 08 00 00          call   20c9 <read_six_numbers>
    17dd:   83 3c 24 00             cmpl   $0x0,(%rsp)
    17e1:   75 07                   jne    17ea <phase_2+0x2e>
    17e3:   83 7c 24 04 01          cmpl   $0x1,0x4(%rsp)
    17e8:   74 05                   je     17ef <phase_2+0x33>
    17ea:   e8 54 08 00 00          call   2043 <explode_bomb>
    17ef:   bb 02 00 00 00          mov    $0x2,%ebx
    17f4:   eb 08                   jmp    17fe <phase_2+0x42>
    17f6:   e8 48 08 00 00          call   2043 <explode_bomb>
    17fb:   83 c3 01                add    $0x1,%ebx
    17fe:   83 fb 05                cmp    $0x5,%ebx
    1801:   7f 1f                   jg     1822 <phase_2+0x66>
    1803:   48 63 c3                movslq %ebx,%rax
    1806:   8d 53 fe                lea    -0x2(%rbx),%edx
    1809:   48 63 d2                movslq %edx,%rdx
    180c:   8b 0c 94                mov    (%rsp,%rdx,4),%ecx
    180f:   8d 53 ff                lea    -0x1(%rbx),%edx
    1812:   48 63 d2                movslq %edx,%rdx
    1815:   8b 14 94                mov    (%rsp,%rdx,4),%edx
    1818:   8d 14 4a                lea    (%rdx,%rcx,2),%edx
    181b:   39 14 84                cmp    %edx,(%rsp,%rax,4)
    181e:   74 db                   je     17fb <phase_2+0x3f>
    1820:   eb d4                   jmp    17f6 <phase_2+0x3a>
    1822:   48 8b 44 24 18          mov    0x18(%rsp),%rax
    1827:   64 48 33 04 25 28 00    xor    %fs:0x28,%rax
    182e:   00 00
    1830:   75 06                   jne    1838 <phase_2+0x7c>
    1832:   48 83 c4 20             add    $0x20,%rsp
    1836:   5b                      pop    %rbx
    1837:   c3                      ret
    1838:   e8 53 fa ff ff          call   1290 <__stack_chk_fail@plt>
```



```
0000000000017bc <phase_2>:
   17bc:   f3 0f 1e fa             endbr64
   17c0:   53                      push   %rbx
   17c1:   48 83 ec 20             sub    $0x20,%rsp
   17c5:   64 48 8b 04 25 28 00    mov    %fs:0x28,%rax
   17cc:   00 00
   17ce:   48 89 44 24 18          mov    %rax,0x18(%rsp)
   17d3:   31 c0                   xor    %eax,%eax
   17d5:   48 89 e6                mov    %rsp,%rsi
   17d8:   e8 ec 08 00 00          call   20c9 <read_six_numbers>
   17dd:   83 3c 24 00             cmpl   $0x0,(%rsp)
   17e1:   75 07                   jne    17ea <phase_2+0x2e>
   17e3:   83 7c 24 04 01          cmpl   $0x1,0x4(%rsp)
   17e8:   74 05                   je     17ef <phase_2+0x33>
   17ea:   e8 54 08 00 00          call   2043 <explode_bomb>
   17ef:   bb 02 00 00 00          mov    $0x2,%ebx
   17f4:   eb 08                   jmp    17fe <phase_2+0x42>
   17f6:   e8 48 08 00 00          call   2043 <explode_bomb>
   17fb:   83 c3 01                add    $0x1,%ebx
   17fe:   83 fb 05                cmp    $0x5,%ebx
   1801:   7f 1f                   jg     1822 <phase_2+0x66>
   1803:   48 63 c3                movslq %ebx,%rax
   1806:   8d 53 fe                lea    -0x2(%rbx),%edx
   1809:   48 63 d2                movslq %edx,%rdx
   180c:   8b 0c 94                mov    (%rsp,%rdx,4),%ecx
   180f:   8d 53 ff                lea    -0x1(%rbx),%edx
   1812:   48 63 d2                movslq %edx,%rdx
   1815:   8b 14 94                mov    (%rsp,%rdx,4),%edx
   1818:   8d 14 4a                lea    (%rdx,%rcx,2),%edx
   181b:   39 14 84                cmp    %edx,(%rsp,%rax,4)
   181e:   74 db                   je     17fb <phase_2+0x3f>
   1820:   eb d4                   jmp    17f6 <phase_2+0x3a>
   1822:   48 8b 44 24 18          mov    0x18(%rsp),%rax
   1827:   64 48 33 04 25 28 00    xor    %fs:0x28,%rax
   182e:   00 00
   1830:   75 06                   jne    1838 <phase_2+0x7c>
   1832:   48 83 c4 20             add    $0x20,%rsp
   1836:   5b                      pop    %rbx
   1837:   c3                      ret
   1838:   e8 53 fa ff ff          call   1290 <__stack_chk_fail@plt>
```

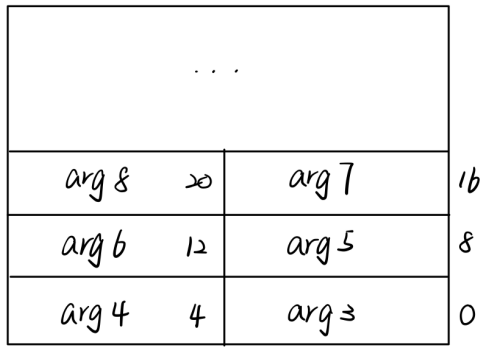
```
00000000000017bc <phase_2>:
    17bc:   f3 0f 1e fa             endbr64
    17c0:   53                      push   %rbx             #被调用者保存
    17c1:   48 83 ec 20             sub    $0x20,%rsp
    17c5:   64 48 8b 04 25 28 00    mov    %fs:0x28,%rax    #fs:0x28 特殊值，放在栈底，fs寄存器的值本身指向当前线程结构
    17cc:   00 00
    17ce:   48 89 44 24 18          mov    %rax,0x18(%rsp)  #金丝雀值，防止栈溢出
    17d3:   31 c0                   xor    %eax,%eax        #%rax置零
```

```
0000000000017bc <phase_2>:
   17bc:   f3 0f 1e fa             endbr64
   17c0:   53                      push   %rbx
   17c1:   48 83 ec 20             sub    $0x20,%rsp
   17c5:   64 48 8b 04 25 28 00    mov    %fs:0x28,%rax
   17cc:   00 00
   17ce:   48 89 44 24 18          mov    %rax,0x18(%rsp)
   17d3:   31 c0                   xor    %eax,%eax
   17d5:   48 89 e6                mov    %rsp,%rsi
   17d8:   e8 ec 08 00 00          call   20c9 <read_six_numbers>
   17dd:   83 3c 24 00             cmpl   $0x0,(%rsp)
   17e1:   75 07                   jne    17ea <phase_2+0x2e>
   17e3:   83 7c 24 04 01          cmpl   $0x1,0x4(%rsp)
   17e8:   74 05                   je     17ef <phase_2+0x33>
   17ea:   e8 54 08 00 00          call   2043 <explode_bomb>
   17ef:   bb 02 00 00 00          mov    $0x2,%ebx
   17f4:   eb 08                   jmp    17fe <phase_2+0x42>
   17f6:   e8 48 08 00 00          call   2043 <explode_bomb>
   17fb:   83 c3 01                add    $0x1,%ebx
   17fe:   83 fb 05                cmp    $0x5,%ebx
   1801:   7f 1f                   jg     1822 <phase_2+0x66>
   1803:   48 63 c3                movslq %ebx,%rax
   1806:   8d 53 fe                lea    -0x2(%rbx),%edx
   1809:   48 63 d2                movslq %edx,%rdx
   180c:   8b 0c 94                mov    (%rsp,%rdx,4),%ecx
   180f:   8d 53 ff                lea    -0x1(%rbx),%edx
   1812:   48 63 d2                movslq %edx,%rdx
   1815:   8b 14 94                mov    (%rsp,%rdx,4),%edx
   1818:   8d 14 4a                lea    (%rdx,%rcx,2),%edx
   181b:   39 14 84                cmp    %edx,(%rsp,%rax,4)
   181e:   74 db                   je     17fb <phase_2+0x3f>
   1820:   eb d4                   jmp    17f6 <phase_2+0x3a>
   1822:   48 8b 44 24 18          mov    0x18(%rsp),%rax
   1827:   64 48 33 04 25 28 00    xor    %fs:0x28,%rax
   182e:   00 00
   1830:   75 06                   jne    1838 <phase_2+0x7c>
   1832:   48 83 c4 20             add    $0x20,%rsp
   1836:   5b                      pop    %rbx
   1837:   c3                      ret
   1838:   e8 53 fa ff ff          call   1290 <__stack_chk_fail@plt>
```

```
17d5:   48 89 e6                mov    %rsp,%rsi                #第二个参数:%rsp
17d8:   e8 ec 08 00 00          call   20c9 <read_six_numbers>
```

```
00000000000020c9 <read_six_numbers>:
    20c9:   f3 0f 1e fa             endbr64
    20cd:   48 83 ec 08             sub    $0x8,%rsp
    20d1:   48 89 f2                mov    %rsi,%rdx                #第三个参数
    20d4:   48 8d 4e 04             lea    0x4(%rsi),%rcx           #第四个参数
    20d8:   48 8d 46 14             lea    0x14(%rsi),%rax
    20dc:   50                      push   %rax                     #第八个参数
    20dd:   48 8d 46 10             lea    0x10(%rsi),%rax
    20e1:   50                      push   %rax                     #第七个参数
    20e2:   4c 8d 4e 0c             lea    0xc(%rsi),%r9            #第六个参数
    20e6:   4c 8d 46 08             lea    0x8(%rsi),%r8            #第五个参数
    20ea:   48 8d 35 d5 26 00 00    lea    0x26d5(%rip),%rsi        # 47c6 <array.3497+0x386>   #第二个参数
    20f1:   b8 00 00 00 00          mov    $0x0,%eax
    20f6:   e8 55 f2 ff ff          call   1350 <__isoc99_sscanf@plt>
    20fb:   48 83 c4 10             add    $0x10,%rsp
    20ff:   83 f8 05                cmp    $0x5,%eax
    2102:   7e 05                   jle    2109 <read_six_numbers+0x40>
    2104:   48 83 c4 08             add    $0x8,%rsp
    2108:   c3                      ret
    2109:   e8 35 ff ff ff          call   2043 <explode_bomb>
```

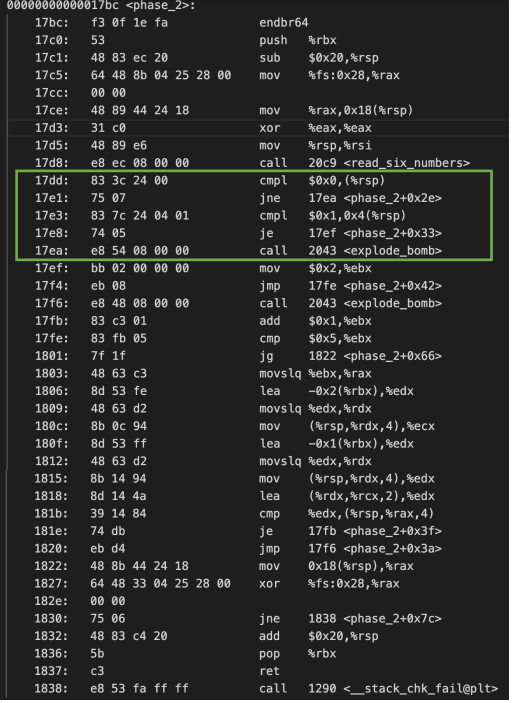
“%d %d %d %d %d %d”




```
int sscanf(const char* str, const char* format, ...);
```

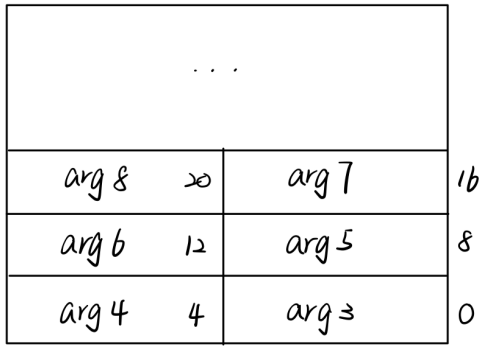
>man sccanf

第7，8个参数通过栈进行传递



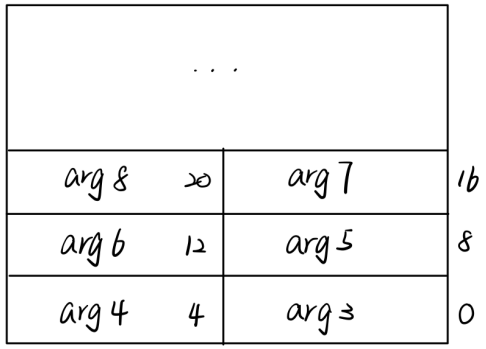
```
0000000000017bc <phase_2>:
   17bc:   f3 0f 1e fa             endbr64
   17c0:   53                      push   %rbx
   17c1:   48 83 ec 20             sub    $0x20,%rsp
   17c5:   64 48 8b 04 25 28 00    mov    %fs:0x28,%rax
   17cc:   00 00
   17ce:   48 89 44 24 18          mov    %rax,0x18(%rsp)
   17d3:   31 c0                   xor    %eax,%eax
   17d5:   48 89 e6                mov    %rsp,%rsi
   17d8:   e8 ec 08 00 00          call   20c9 <read_six_numbers>
   17dd:   83 3c 24 00             cmpl   $0x0,(%rsp)
   17e1:   75 07                   jne    17ea <phase_2+0x2e>
   17e3:   83 7c 24 04 01          cmpl   $0x1,0x4(%rsp)
   17e8:   74 05                   je     17ef <phase_2+0x33>
   17ea:   e8 54 08 00 00          call   2043 <explode_bomb>
   17ef:   bb 02 00 00 00          mov    $0x2,%ebx
   17f4:   eb 08                   jmp    17fe <phase_2+0x42>
   17f6:   e8 48 08 00 00          call   2043 <explode_bomb>
   17fb:   83 c3 01                add    $0x1,%ebx
   17fe:   83 fb 05                cmp    $0x5,%ebx
   1801:   7f 1f                   jg     1822 <phase_2+0x66>
   1803:   48 63 c3                movslq %ebx,%rax
   1806:   8d 53 fe                lea    -0x2(%rbx),%edx
   1809:   48 63 d2                movslq %edx,%rdx
   180c:   8b 0c 94                mov    (%rsp,%rdx,4),%ecx
   180f:   8d 53 ff                lea    -0x1(%rbx),%edx
   1812:   48 63 d2                movslq %edx,%rdx
   1815:   8b 14 94                mov    (%rsp,%rdx,4),%edx
   1818:   8d 14 4a                lea    (%rdx,%rcx,2),%edx
   181b:   39 14 84                cmp    %edx,(%rsp,%rax,4)
   181e:   74 db                   je     17fb <phase_2+0x3f>
   1820:   eb d4                   jmp    17f6 <phase_2+0x3a>
   1822:   48 8b 44 24 18          mov    0x18(%rsp),%rax
   1827:   64 48 33 04 25 28 00    xor    %fs:0x28,%rax
   182e:   00 00
   1830:   75 06                   jne    1838 <phase_2+0x7c>
   1832:   48 83 c4 20             add    $0x20,%rsp
   1836:   5b                      pop    %rbx
   1837:   c3                      ret
   1838:   e8 53 fa ff ff          call   1290 <__stack_chk_fail@plt>
```

```
17dd:   83 3c 24 00             cmpl   $0x0,(%rsp)              #输入的第一个数是0
17e1:   75 07                   jne    17ea <phase_2+0x2e>
17e3:   83 7c 24 04 01          cmpl   $0x1,0x4(%rsp)           #输入的第二个数是1
17e8:   74 05                   je     17ef <phase_2+0x33>
17ea:   e8 54 08 00 00          call   2043 <explode_bomb>
```



```
0000000000017bc <phase_2>:
   17bc:   f3 0f 1e fa             endbr64
   17c0:   53                      push   %rbx
   17c1:   48 83 ec 20             sub    $0x20,%rsp
   17c5:   64 48 8b 04 25 28 00    mov    %fs:0x28,%rax
   17cc:   00 00
   17ce:   48 89 44 24 18          mov    %rax,0x18(%rsp)
   17d3:   31 c0                   xor    %eax,%eax
   17d5:   48 89 e6                mov    %rsp,%rsi
   17d8:   e8 ec 08 00 00          call   20c9 <read_six_numbers>
   17dd:   83 3c 24 00             cmpl   $0x0,(%rsp)
   17e1:   75 07                   jne    17ea <phase_2+0x2e>
   17e3:   83 7c 24 04 01          cmpl   $0x1,0x4(%rsp)
   17e8:   74 05                   je     17ef <phase_2+0x33>
   17ea:   e8 54 08 00 00          call   2043 <explode_bomb>
   17ef:   bb 02 00 00 00          mov    $0x2,%ebx
   17f4:   eb 08                   jmp    17fe <phase_2+0x42>
   17f6:   e8 48 08 00 00          call   2043 <explode_bomb>
   17fb:   83 c3 01                add    $0x1,%ebx
   17fe:   83 fb 05                cmp    $0x5,%ebx
   1801:   7f 1f                   jg     1822 <phase_2+0x66>
   1803:   48 63 c3                movslq %ebx,%rax
   1806:   8d 53 fe                lea    -0x2(%rbx),%edx
   1809:   48 63 d2                movslq %edx,%rdx
   180c:   8b 0c 94                mov    (%rsp,%rdx,4),%ecx
   180f:   8d 53 ff                lea    -0x1(%rbx),%edx
   1812:   48 63 d2                movslq %edx,%rdx
   1815:   8b 14 94                mov    (%rsp,%rdx,4),%edx
   1818:   8d 14 4a                lea    (%rdx,%rcx,2),%edx
   181b:   39 14 84                cmp    %edx,(%rsp,%rax,4)
   181e:   74 db                   je     17fb <phase_2+0x3f>
   1820:   eb d4                   jmp    17f6 <phase_2+0x3a>
   1822:   48 8b 44 24 18          mov    0x18(%rsp),%rax
   1827:   64 48 33 04 25 28 00    xor    %fs:0x28,%rax
   182e:   00 00
   1830:   75 06                   jne    1838 <phase_2+0x7c>
   1832:   48 83 c4 20             add    $0x20,%rsp
   1836:   5b                      pop    %rbx
   1837:   c3                      ret
   1838:   e8 53 fa ff ff          call   1290 <__stack_chk_fail@plt>
```

```
17ef:   bb 02 00 00 00          mov    $0x2,%ebx              #                    ebx=2
17f4:   eb 08                   jmp    17fe <phase_2+0x42>
17f6:   e8 48 08 00 00          call   2043 <explode_bomb>
17fb:   83 c3 01                add    $0x1,%ebx              #                    rbx=3          rbx=4          rbx=5
17fe:   83 fb 05                cmp    $0x5,%ebx              #ebx:5
1801:   7f 1f                   jg     1822 <phase_2+0x66>    #ebx=2,3,4,5
1803:   48 63 c3                movslq %ebx,%rax              #rax=ebx             rax=2          rax=3          rax=4          rax=5
1806:   8d 53 fe                lea    -0x2(%rbx),%edx        #edx=rbx-2;          edx=0
1809:   48 63 d2                movslq %edx,%rdx              #rdx=edx             rdx=0          rdx=1          rdx=2          rdx=3
180c:   8b 0c 94                mov    (%rsp,%rdx,4),%ecx     #ecx=M[rsp+4*rdx]    ecx=0          ecx=1          ecx=1          ecx=3
180f:   8d 53 ff                lea    -0x1(%rbx),%edx        #edx=rbx-1           edx=1
1812:   48 63 d2                movslq %edx,%rdx              #rdx=edx             rdx=1;         rdx=2          rdx=3          rdx=4
1815:   8b 14 94                mov    (%rsp,%rdx,4),%edx     #edx=M[rsp+4*rdx]    edx=1;         edx=1          edx=3          edx=5
1818:   8d 14 4a                lea    (%rdx,%rcx,2),%edx     #edx=rdx+2*rcx       edx=1;         edx=3          edx=5          edx=11
181b:   39 14 84                cmp    %edx,(%rsp,%rax,4)     #M[rsp+4*rax]:edx    M[rsp+0x8]:1   M[%rsp+0xc]:3   M[%rsp+0x10]:5  M[%rsp+0x14]:11
181e:   74 db                   je     17fb <phase_2+0x3f>
1820:   eb d4                   jmp    17f6 <phase_2+0x3a>
```



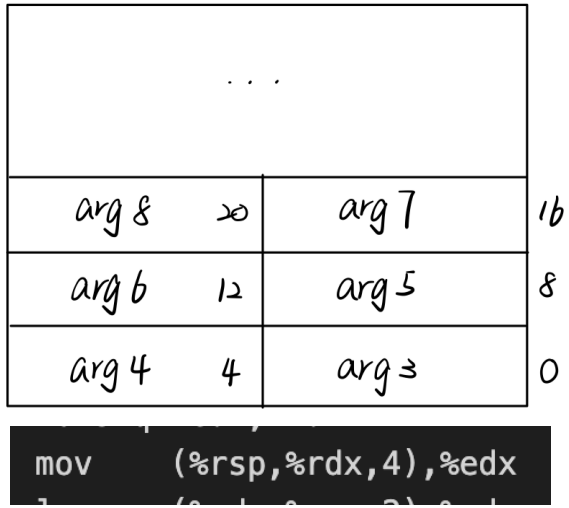
$$a[6] = [0,1,a_2,a_3,a_4,a_5]$$

$$a[i] = 2 * a[i-2] + a[i-1], i = 2,3,4,5$$

$$a[i] = [0,1,1,3,5,11]$$

$$arg_k = a[\mathrm{k}-3], \mathrm{k} = 3,4,5,6,7,8$$

局部变量的存储



- 在栈中
  - 寄存器不足够存放所有的本地数据
  - 对一个局部变量使用了地址运算‘&’
  - 局部变量是**数组**/结构，必须能够通过数组/结构引用被访问到（本题的情况）

- 在寄存器中
  - 寄存器是唯一被所有过程共享的资源
  - 被调用者保存（callee saved）寄存器：%rbx,%rbp,%r12~%r15
  - 调用者保存（caller saved）寄存器：除被调用者寄存器以及%rsp以外的其他寄存器

phase_3

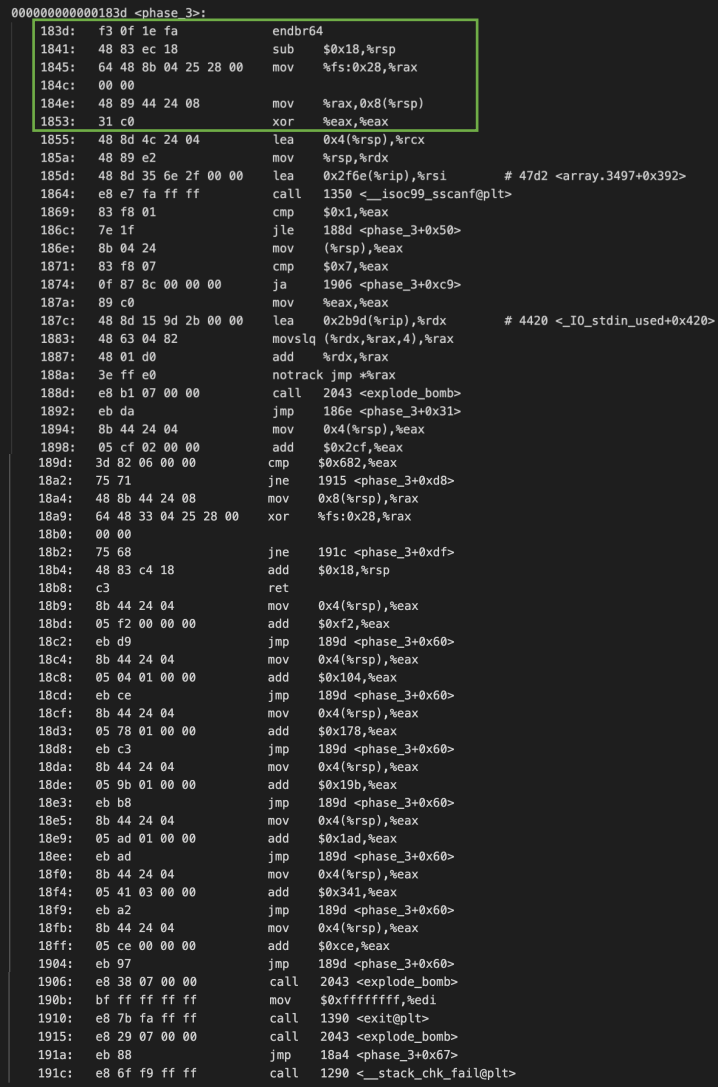
Switch and Jmp Table

解题思路

- 根据汇编代码易知输入的格式为%d %d，且第一个整数为0到7之间的数
- 第一个整数不同，跳转到的位置也不同，对应C中switch语句不同的case
- 根据不同的case计算得到与输入的第一个整数对应的第二个整数

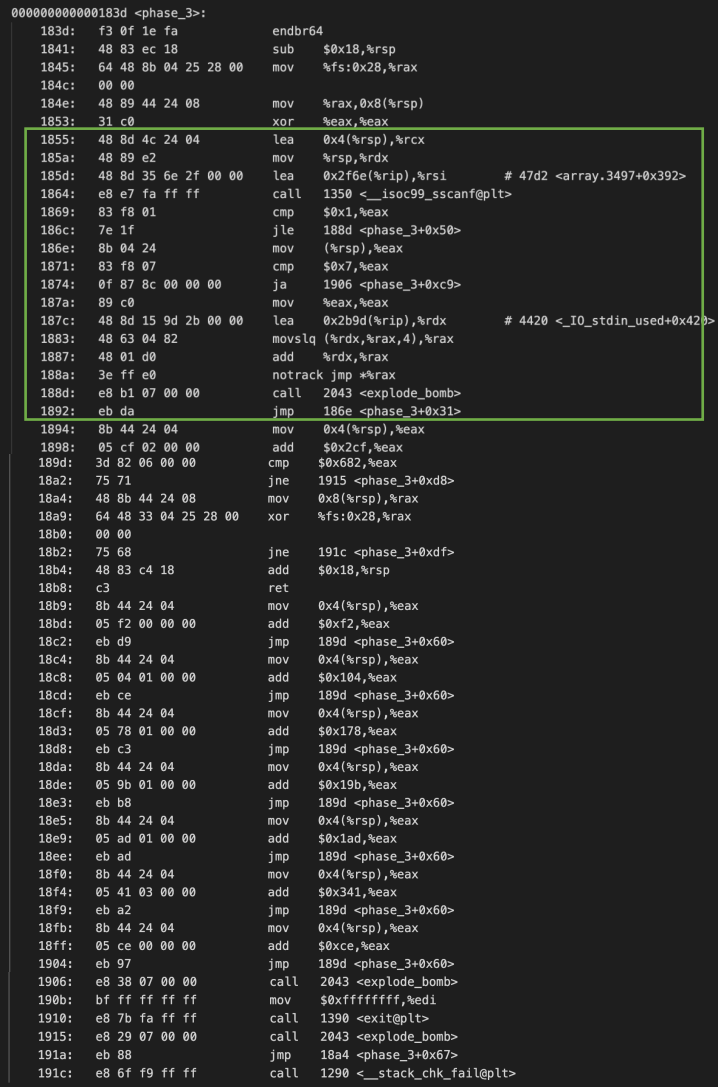
```
000000000000183d <phase_3>:
    183d:   f3 0f 1e fa             endbr64
    1841:   48 83 ec 18             sub    $0x18,%rsp
    1845:   64 48 8b 04 25 28 00    mov    %fs:0x28,%rax
    184c:   00 00
    184e:   48 89 44 24 08          mov    %rax,0x8(%rsp)
    1853:   31 c0                   xor    %eax,%eax
    1855:   48 8d 4c 24 04          lea    0x4(%rsp),%rcx
    185a:   48 89 e2                mov    %rsp,%rdx
    185d:   48 8d 35 6e 2f 00 00    lea    0x2f6e(%rip),%rsi        # 47d2 <array.3497+0x392>
    1864:   e8 e7 fa ff ff          call   1350 <__isoc99_sscanf@plt>
    1869:   83 f8 01                cmp    $0x1,%eax
    186c:   7e 1f                   jle    188d <phase_3+0x50>
    186e:   8b 04 24                mov    (%rsp),%eax
    1871:   83 f8 07                cmp    $0x7,%eax
    1874:   0f 87 8c 00 00 00       ja     1906 <phase_3+0xc9>
    187a:   89 c0                   mov    %eax,%eax
    187c:   48 8d 15 9d 2b 00 00    lea    0x2b9d(%rip),%rdx        # 4420 <_IO_stdin_used+0x420>
    1883:   48 63 04 82             movslq (%rdx,%rax,4),%rax
    1887:   48 01 d0                add    %rdx,%rax
    188a:   3e ff e0                notrack jmp *%rax
    188d:   e8 b1 07 00 00          call   2043 <explode_bomb>
    1892:   eb da                   jmp    186e <phase_3+0x31>
    1894:   8b 44 24 04             mov    0x4(%rsp),%eax
    1898:   05 cf 02 00 00          add    $0x2cf,%eax
```

```
    189d:   3d 82 06 00 00          cmp    $0x682,%eax
    18a2:   75 71                   jne    1915 <phase_3+0xd8>
    18a4:   48 8b 44 24 08          mov    0x8(%rsp),%rax
    18a9:   64 48 33 04 25 28 00    xor    %fs:0x28,%rax
    18b0:   00 00
    18b2:   75 68                   jne    191c <phase_3+0xdf>
    18b4:   48 83 c4 18             add    $0x18,%rsp
    18b8:   c3                      ret
    18b9:   8b 44 24 04             mov    0x4(%rsp),%eax
    18bd:   05 f2 00 00 00          add    $0xf2,%eax
    18c2:   eb d9                   jmp    189d <phase_3+0x60>
    18c4:   8b 44 24 04             mov    0x4(%rsp),%eax
    18c8:   05 04 01 00 00          add    $0x104,%eax
    18cd:   eb ce                   jmp    189d <phase_3+0x60>
    18cf:   8b 44 24 04             mov    0x4(%rsp),%eax
    18d3:   05 78 01 00 00          add    $0x178,%eax
    18d8:   eb c3                   jmp    189d <phase_3+0x60>
    18da:   8b 44 24 04             mov    0x4(%rsp),%eax
    18de:   05 9b 01 00 00          add    $0x19b,%eax
    18e3:   eb b8                   jmp    189d <phase_3+0x60>
    18e5:   8b 44 24 04             mov    0x4(%rsp),%eax
    18e9:   05 ad 01 00 00          add    $0x1ad,%eax
    18ee:   eb ad                   jmp    189d <phase_3+0x60>
    18f0:   8b 44 24 04             mov    0x4(%rsp),%eax
    18f4:   05 41 03 00 00          add    $0x341,%eax
    18f9:   eb a2                   jmp    189d <phase_3+0x60>
    18fb:   8b 44 24 04             mov    0x4(%rsp),%eax
    18ff:   05 ce 00 00 00          add    $0xce,%eax
    1904:   eb 97                   jmp    189d <phase_3+0x60>
    1906:   e8 38 07 00 00          call   2043 <explode_bomb>
    190b:   bf ff ff ff ff          mov    $0xffffffff,%edi
    1910:   e8 7b fa ff ff          call   1390 <exit@plt>
    1915:   e8 29 07 00 00          call   2043 <explode_bomb>
    191a:   eb 88                   jmp    18a4 <phase_3+0x67>
    191c:   e8 6f f9 ff ff          call   1290 <__stack_chk_fail@plt>
```



```
000000000000183d <phase_3>:
    183d:   f3 0f 1e fa             endbr64
    1841:   48 83 ec 18             sub    $0x18,%rsp
    1845:   64 48 8b 04 25 28 00    mov    %fs:0x28,%rax
    184c:   00 00
    184e:   48 89 44 24 08          mov    %rax,0x8(%rsp)
    1853:   31 c0                   xor    %eax,%eax
    1855:   48 8d 4c 24 04          lea    0x4(%rsp),%rcx
    185a:   48 89 e2                mov    %rsp,%rdx
    185d:   48 8d 35 6e 2f 00 00    lea    0x2f6e(%rip),%rsi        # 47d2 <array.3497+0x392>
    1864:   e8 e7 fa ff ff          call   1350 <__isoc99_sscanf@plt>
    1869:   83 f8 01                cmp    $0x1,%eax
    186c:   7e 1f                   jle    188d <phase_3+0x50>
    186e:   8b 04 24                mov    (%rsp),%eax
    1871:   83 f8 07                cmp    $0x7,%eax
    1874:   0f 87 8c 00 00 00       ja     1906 <phase_3+0xc9>
    187a:   89 c0                   mov    %eax,%eax
    187c:   48 8d 15 9d 2b 00 00    lea    0x2b9d(%rip),%rdx        # 4420 <_IO_stdin_used+0x420>
    1883:   48 63 04 82             movslq (%rdx,%rax,4),%rax
    1887:   48 01 d0                add    %rdx,%rax
    188a:   3e ff e0                notrack jmp *%rax
    188d:   e8 b1 07 00 00          call   2043 <explode_bomb>
    1892:   eb da                   jmp    186e <phase_3+0x31>
    1894:   8b 44 24 04             mov    0x4(%rsp),%eax
    1898:   05 cf 02 00 00          add    $0x2cf,%eax
    189d:   3d 82 06 00 00          cmp    $0x682,%eax
    18a2:   75 71                   jne    1915 <phase_3+0xd8>
    18a4:   48 8b 44 24 08          mov    0x8(%rsp),%rax
    18a9:   64 48 33 04 25 28 00    xor    %fs:0x28,%rax
    18b0:   00 00
    18b2:   75 68                   jne    191c <phase_3+0xdf>
    18b4:   48 83 c4 18             add    $0x18,%rsp
    18b8:   c3                      ret
    18b9:   8b 44 24 04             mov    0x4(%rsp),%eax
    18bd:   05 f2 00 00 00          add    $0xf2,%eax
    18c2:   eb d9                   jmp    189d <phase_3+0x60>
    18c4:   8b 44 24 04             mov    0x4(%rsp),%eax
    18c8:   05 04 01 00 00          add    $0x104,%eax
    18cd:   eb ce                   jmp    189d <phase_3+0x60>
    18cf:   8b 44 24 04             mov    0x4(%rsp),%eax
    18d3:   05 78 01 00 00          add    $0x178,%eax
    18d8:   eb c3                   jmp    189d <phase_3+0x60>
    18da:   8b 44 24 04             mov    0x4(%rsp),%eax
    18de:   05 9b 01 00 00          add    $0x19b,%eax
    18e3:   eb b8                   jmp    189d <phase_3+0x60>
    18e5:   8b 44 24 04             mov    0x4(%rsp),%eax
    18e9:   05 ad 01 00 00          add    $0x1ad,%eax
    18ee:   eb ad                   jmp    189d <phase_3+0x60>
    18f0:   8b 44 24 04             mov    0x4(%rsp),%eax
    18f4:   05 41 03 00 00          add    $0x341,%eax
    18f9:   eb a2                   jmp    189d <phase_3+0x60>
    18fb:   8b 44 24 04             mov    0x4(%rsp),%eax
    18ff:   05 ce 00 00 00          add    $0xce,%eax
    1904:   eb 97                   jmp    189d <phase_3+0x60>
    1906:   e8 38 07 00 00          call   2043 <explode_bomb>
    190b:   bf ff ff ff ff          mov    $0xffffffff,%edi
    1910:   e8 7b fa ff ff          call   1390 <exit@plt>
    1915:   e8 29 07 00 00          call   2043 <explode_bomb>
    191a:   eb 88                   jmp    18a4 <phase_3+0x67>
    191c:   e8 6f f9 ff ff          call   1290 <__stack_chk_fail@plt>
```

```
000000000000183d <phase_3>:
    183d:   f3 0f 1e fa             endbr64
    1841:   48 83 ec 18             sub    $0x18,%rsp
    1845:   64 48 8b 04 25 28 00    mov    %fs:0x28,%rax   #栈溢出检测
    184c:   00 00
    184e:   48 89 44 24 08          mov    %rax,0x8(%rsp)
    1853:   31 c0                   xor    %eax,%eax       #rax=0
```



```
000000000000183d <phase_3>:
    183d:   f3 0f 1e fa             endbr64
    1841:   48 83 ec 18             sub    $0x18,%rsp
    1845:   64 48 8b 04 25 28 00    mov    %fs:0x28,%rax
    184c:   00 00
    184e:   48 89 44 24 08          mov    %rax,0x8(%rsp)
    1853:   31 c0                   xor    %eax,%eax
    1855:   48 8d 4c 24 04          lea    0x4(%rsp),%rcx
    185a:   48 89 e2                mov    %rsp,%rdx
    185d:   48 8d 35 6e 2f 00 00    lea    0x2f6e(%rip),%rsi        # 47d2 <array.3497+0x392>
    1864:   e8 e7 fa ff ff          call   1350 <__isoc99_sscanf@plt>
    1869:   83 f8 01                cmp    $0x1,%eax
    186c:   7e 1f                   jle    188d <phase_3+0x50>
    186e:   8b 04 24                mov    (%rsp),%eax
    1871:   83 f8 07                cmp    $0x7,%eax
    1874:   0f 87 8c 00 00 00       ja     1906 <phase_3+0xc9>
    187a:   89 c0                   mov    %eax,%eax
    187c:   48 8d 15 9d 2b 00 00    lea    0x2b9d(%rip),%rdx        # 4420 <_IO_stdin_used+0x420>
    1883:   48 63 04 82             movslq (%rdx,%rax,4),%rax
    1887:   48 01 d0                add    %rdx,%rax
    188a:   3e ff e0                notrack jmp *%rax
    188d:   e8 b1 07 00 00          call   2043 <explode_bomb>
    1892:   eb da                   jmp    186e <phase_3+0x31>
    1894:   8b 44 24 04             mov    0x4(%rsp),%eax
    1898:   05 cf 02 00 00          add    $0x2cf,%eax
    189d:   3d 82 06 00 00          cmp    $0x682,%eax
    18a2:   75 71                   jne    1915 <phase_3+0xd8>
    18a4:   48 8b 44 24 08          mov    0x8(%rsp),%rax
    18a9:   64 48 33 04 25 28 00    xor    %fs:0x28,%rax
    18b0:   00 00
    18b2:   75 68                   jne    191c <phase_3+0xdf>
    18b4:   48 83 c4 18             add    $0x18,%rsp
    18b8:   c3                      ret
    18b9:   8b 44 24 04             mov    0x4(%rsp),%eax
    18bd:   05 f2 00 00 00          add    $0xf2,%eax
    18c2:   eb d9                   jmp    189d <phase_3+0x60>
    18c4:   8b 44 24 04             mov    0x4(%rsp),%eax
    18c8:   05 04 01 00 00          add    $0x104,%eax
    18cd:   eb ce                   jmp    189d <phase_3+0x60>
    18cf:   8b 44 24 04             mov    0x4(%rsp),%eax
    18d3:   05 78 01 00 00          add    $0x178,%eax
    18d8:   eb c3                   jmp    189d <phase_3+0x60>
    18da:   8b 44 24 04             mov    0x4(%rsp),%eax
    18de:   05 9b 01 00 00          add    $0x19b,%eax
    18e3:   eb b8                   jmp    189d <phase_3+0x60>
    18e5:   8b 44 24 04             mov    0x4(%rsp),%eax
    18e9:   05 ad 01 00 00          add    $0x1ad,%eax
    18ee:   eb ad                   jmp    189d <phase_3+0x60>
    18f0:   8b 44 24 04             mov    0x4(%rsp),%eax
    18f4:   05 41 03 00 00          add    $0x341,%eax
    18f9:   eb a2                   jmp    189d <phase_3+0x60>
    18fb:   8b 44 24 04             mov    0x4(%rsp),%eax
    18ff:   05 ce 00 00 00          add    $0xce,%eax
    1904:   eb 97                   jmp    189d <phase_3+0x60>
    1906:   e8 38 07 00 00          call   2043 <explode_bomb>
    190b:   bf ff ff ff ff          mov    $0xffffffff,%edi
    1910:   e8 7b fa ff ff          call   1390 <exit@plt>
    1915:   e8 29 07 00 00          call   2043 <explode_bomb>
    191a:   eb 88                   jmp    18a4 <phase_3+0x67>
    191c:   e8 6f f9 ff ff          call   1290 <__stack_chk_fail@plt>
```

```
1855:   48 8d 4c 24 04          lea    0x4(%rsp),%rcx                          #rcx=rsp+4   第四个参数
185a:   48 89 e2                mov    %rsp,%rdx                               #rdx=rsp     第三个参数
185d:   48 8d 35 6e 2f 00 00    lea    0x2f6e(%rip),%rsi        # 47d2 <array.3497+0x392>   #rsi=rip+0x2f6e    "%d %d"
1864:   e8 e7 fa ff ff          call   1350 <__isoc99_sscanf@plt>                  #read 2 numbers
1869:   83 f8 01                cmp    $0x1,%eax                               #eax:1 (eax=2)
186c:   7e 1f                   jle    188d <phase_3+0x50>
186e:   8b 04 24                mov    (%rsp),%eax                             #rax=M[rsp] (输入的第一个数)
1871:   83 f8 07                cmp    $0x7,%eax                               #rax:7
1874:   0f 87 8c 00 00 00       ja     1906 <phase_3+0xc9>                     #0<=rax<=7
187a:   89 c0                   mov    %eax,%eax                               #eax=eax (高位4字节清零)
187c:   48 8d 15 9d 2b 00 00    lea    0x2b9d(%rip),%rdx        # 4420 <_IO_stdin_used+0x420>   #rdx=rip+0x2b9d
1883:   48 63 04 82             movslq (%rdx,%rax,4),%rax                      #rax=M[rdx+4*rax]
1887:   48 01 d0                add    %rdx,%rax                               #rax=rax+rdx
188a:   3e ff e0                notrack jmp *%rax                              #跳转到rax
188d:   e8 b1 07 00 00          call   2043 <explode_bomb>
1892:   eb da                   jmp    186e <phase_3+0x31>
```

%rax=M[%rdx+4*%rax]+%rdx

基地址+存放在内存数组中的偏移量

```
000000000000183d <phase_3>:
    183d:   f3 0f 1e fa             endbr64
    1841:   48 83 ec 18             sub    $0x18,%rsp
    1845:   64 48 8b 04 25 28 00    mov    %fs:0x28,%rax
    184c:   00 00
    184e:   48 89 44 24 08          mov    %rax,0x8(%rsp)
    1853:   31 c0                   xor    %eax,%eax
    1855:   48 8d 4c 24 04          lea    0x4(%rsp),%rcx
    185a:   48 89 e2                mov    %rsp,%rdx
    185d:   48 8d 35 6e 2f 00 00    lea    0x2f6e(%rip),%rsi        # 47d2 <array.3497+0x392>
    1864:   e8 e7 fa ff ff          call   1350 <__isoc99_sscanf@plt>
    1869:   83 f8 01                cmp    $0x1,%eax
    186c:   7e 1f                   jle    188d <phase_3+0x50>
    186e:   8b 04 24                mov    (%rsp),%eax
    1871:   83 f8 07                cmp    $0x7,%eax
    1874:   0f 87 8c 00 00 00       ja     1906 <phase_3+0xc9>
    187a:   89 c0                   mov    %eax,%eax
    187c:   48 8d 15 9d 2b 00 00    lea    0x2b9d(%rip),%rdx        # 4420 <_IO_stdin_used+0x420>
    1883:   48 63 04 82             movslq (%rdx,%rax,4),%rax
    1887:   48 01 d0                add    %rdx,%rax
    188a:   3e ff e0                notrack jmp *%rax
    188d:   e8 b1 07 00 00          call   2043 <explode_bomb>
    1892:   eb da                   jmp    186e <phase_3+0x31>
    1894:   8b 44 24 04             mov    0x4(%rsp),%eax
    1898:   05 cf 02 00 00          add    $0x2cf,%eax
    189d:   3d 82 06 00 00          cmp    $0x682,%eax
    18a2:   75 71                   jne    1915 <phase_3+0xd8>
    18a4:   48 8b 44 24 08          mov    0x8(%rsp),%rax
    18a9:   64 48 33 04 25 28 00    xor    %fs:0x28,%rax
    18b0:   00 00
    18b2:   75 68                   jne    191c <phase_3+0xdf>
    18b4:   48 83 c4 18             add    $0x18,%rsp
    18b8:   c3                      ret
    18b9:   8b 44 24 04             mov    0x4(%rsp),%eax
    18bd:   05 f2 00 00 00          add    $0xf2,%eax
    18c2:   eb d9                   jmp    189d <phase_3+0x60>
    18c4:   8b 44 24 04             mov    0x4(%rsp),%eax
    18c8:   05 04 01 00 00          add    $0x104,%eax
    18cd:   eb ce                   jmp    189d <phase_3+0x60>
    18cf:   8b 44 24 04             mov    0x4(%rsp),%eax
    18d3:   05 78 01 00 00          add    $0x178,%eax
    18d8:   eb c3                   jmp    189d <phase_3+0x60>
    18da:   8b 44 24 04             mov    0x4(%rsp),%eax
    18de:   05 9b 01 00 00          add    $0x19b,%eax
    18e3:   eb b8                   jmp    189d <phase_3+0x60>
    18e5:   8b 44 24 04             mov    0x4(%rsp),%eax
    18e9:   05 ad 01 00 00          add    $0x1ad,%eax
    18ee:   eb ad                   jmp    189d <phase_3+0x60>
    18f0:   8b 44 24 04             mov    0x4(%rsp),%eax
    18f4:   05 41 03 00 00          add    $0x341,%eax
    18f9:   eb a2                   jmp    189d <phase_3+0x60>
    18fb:   8b 44 24 04             mov    0x4(%rsp),%eax
    18ff:   05 ce 00 00 00          add    $0xce,%eax
    1904:   eb 97                   jmp    189d <phase_3+0x60>
    1906:   e8 38 07 00 00          call   2043 <explode_bomb>
    190b:   bf ff ff ff ff          mov    $0xffffffff,%edi
    1910:   e8 7b fa ff ff          call   1390 <exit@plt>
    1915:   e8 29 07 00 00          call   2043 <explode_bomb>
    191a:   eb 88                   jmp    18a4 <phase_3+0x67>
    191c:   e8 6f f9 ff ff          call   1290 <__stack_chk_fail@plt>
```

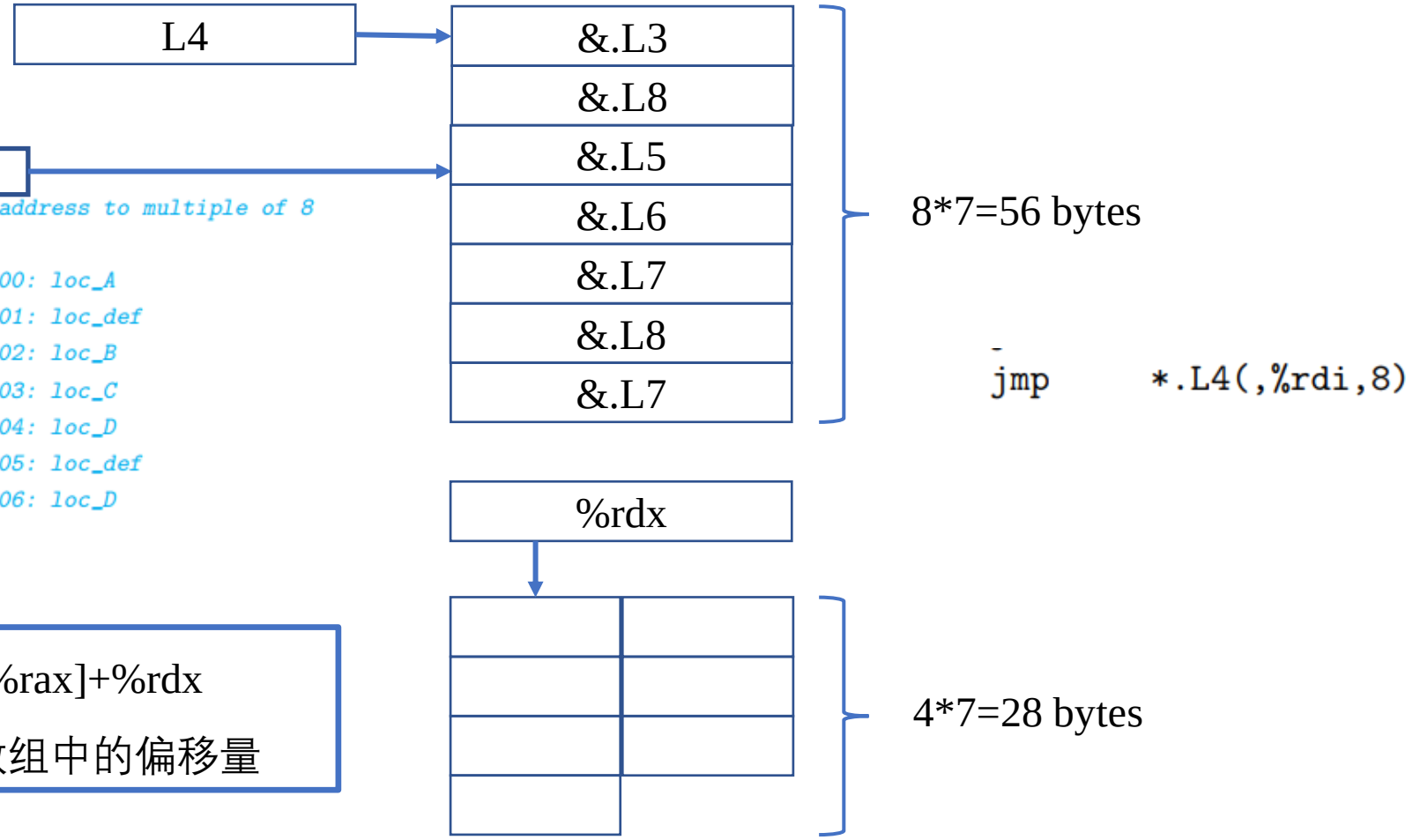
```
//输入的第一个数是0
1894:   8b 44 24 04             mov    0x4(%rsp),%eax                  #eax=M[rsp+0x4] (输入的第二个数)
1898:   05 cf 02 00 00          add    $0x2cf,%eax                     #eax+=0x2cf
//返回
189d:   3d 82 06 00 00          cmp    $0x682,%eax                     #eax:0x682
18a2:   75 71                   jne    1915 <phase_3+0xd8>             #eax==0x682
18a4:   48 8b 44 24 08          mov    0x8(%rsp),%rax
18a9:   64 48 33 04 25 28 00    xor    %fs:0x28,%rax
18b0:   00 00
18b2:   75 68                   jne    191c <phase_3+0xdf>
18b4:   48 83 c4 18             add    $0x18,%rsp
18b8:   c3                      ret
//输入的第一个数是1
18b9:   8b 44 24 04             mov    0x4(%rsp),%eax                  #eax=M[rsp+0x4] (输入的第二个数)
18bd:   05 f2 00 00 00          add    $0xf2,%eax                      #eax+=0xf2
18c2:   eb d9                   jmp    189d <phase_3+0x60>
//输入的第一个数是2
18c4:   8b 44 24 04             mov    0x4(%rsp),%eax
18c8:   05 04 01 00 00          add    $0x104,%eax
18cd:   eb ce                   jmp    189d <phase_3+0x60>
//输入的第一个数是3
18cf:   8b 44 24 04             mov    0x4(%rsp),%eax
18d3:   05 78 01 00 00          add    $0x178,%eax
18d8:   eb c3                   jmp    189d <phase_3+0x60>
//输入的第一个数是4
18da:   8b 44 24 04             mov    0x4(%rsp),%eax
18de:   05 9b 01 00 00          add    $0x19b,%eax
18e3:   eb b8                   jmp    189d <phase_3+0x60>
//输入的第一个数是5
18e5:   8b 44 24 04             mov    0x4(%rsp),%eax
18e9:   05 ad 01 00 00          add    $0x1ad,%eax
18ee:   eb ad                   jmp    189d <phase_3+0x60>
//输入的第一个数是6
18f0:   8b 44 24 04             mov    0x4(%rsp),%eax
18f4:   05 41 03 00 00          add    $0x341,%eax
18f9:   eb a2                   jmp    189d <phase_3+0x60>
//输入的第一个数是7
18fb:   8b 44 24 04             mov    0x4(%rsp),%eax
18ff:   05 ce 00 00 00          add    $0xce,%eax
```

```
000000000000183d <phase_3>:
    183d:   f3 0f 1e fa             endbr64
    1841:   48 83 ec 18             sub    $0x18,%rsp
    1845:   64 48 8b 04 25 28 00    mov    %fs:0x28,%rax
    184c:   00 00
    184e:   48 89 44 24 08          mov    %rax,0x8(%rsp)
    1853:   31 c0                   xor    %eax,%eax
    1855:   48 8d 4c 24 04          lea    0x4(%rsp),%rcx
    185a:   48 89 e2                mov    %rsp,%rdx
    185d:   48 8d 35 6e 2f 00 00    lea    0x2f6e(%rip),%rsi        # 47d2 <array.3497+0x392>
    1864:   e8 e7 fa ff ff          call   1350 <__isoc99_sscanf@plt>
    1869:   83 f8 01                cmp    $0x1,%eax
    186c:   7e 1f                   jle    188d <phase_3+0x50>
    186e:   8b 04 24                mov    (%rsp),%eax
    1871:   83 f8 07                cmp    $0x7,%eax
    1874:   0f 87 8c 00 00 00       ja     1906 <phase_3+0xc9>
    187a:   89 c0                   mov    %eax,%eax
    187c:   48 8d 15 9d 2b 00 00    lea    0x2b9d(%rip),%rdx        # 4420 <_IO_stdin_used+0x420>
    1883:   48 63 04 82             movslq (%rdx,%rax,4),%rax
    1887:   48 01 d0                add    %rdx,%rax
    188a:   3e ff e0                notrack jmp *%rax
    188d:   e8 b1 07 00 00          call   2043 <explode_bomb>
    1892:   eb da                   jmp    186e <phase_3+0x31>
    1894:   8b 44 24 04             mov    0x4(%rsp),%eax
    1898:   05 cf 02 00 00          add    $0x2cf,%eax
    189d:   3d 82 06 00 00          cmp    $0x682,%eax
    18a2:   75 71                   jne    1915 <phase_3+0xd8>
    18a4:   48 8b 44 24 08          mov    0x8(%rsp),%rax
    18a9:   64 48 33 04 25 28 00    xor    %fs:0x28,%rax
    18b0:   00 00
    18b2:   75 68                   jne    191c <phase_3+0xdf>
    18b4:   48 83 c4 18             add    $0x18,%rsp
    18b8:   c3                      ret
    18b9:   8b 44 24 04             mov    0x4(%rsp),%eax
    18bd:   05 f2 00 00 00          add    $0xf2,%eax
    18c2:   eb d9                   jmp    189d <phase_3+0x60>
    18c4:   8b 44 24 04             mov    0x4(%rsp),%eax
    18c8:   05 04 01 00 00          add    $0x104,%eax
    18cd:   eb ce                   jmp    189d <phase_3+0x60>
    18cf:   8b 44 24 04             mov    0x4(%rsp),%eax
    18d3:   05 78 01 00 00          add    $0x178,%eax
    18d8:   eb c3                   jmp    189d <phase_3+0x60>
    18da:   8b 44 24 04             mov    0x4(%rsp),%eax
    18de:   05 9b 01 00 00          add    $0x19b,%eax
    18e3:   eb b8                   jmp    189d <phase_3+0x60>
    18e5:   8b 44 24 04             mov    0x4(%rsp),%eax
    18e9:   05 ad 01 00 00          add    $0x1ad,%eax
    18ee:   eb ad                   jmp    189d <phase_3+0x60>
    18f0:   8b 44 24 04             mov    0x4(%rsp),%eax
    18f4:   05 41 03 00 00          add    $0x341,%eax
    18f9:   eb a2                   jmp    189d <phase_3+0x60>
    18fb:   8b 44 24 04             mov    0x4(%rsp),%eax
    18ff:   05 ce 00 00 00          add    $0xce,%eax
    1904:   eb 97                   jmp    189d <phase_3+0x60>
    1906:   e8 38 07 00 00          call   2043 <explode_bomb>
    190b:   bf ff ff ff ff          mov    $0xffffffff,%edi
    1910:   e8 7b fa ff ff          call   1390 <exit@plt>
    1915:   e8 29 07 00 00          call   2043 <explode_bomb>
    191a:   eb 88                   jmp    18a4 <phase_3+0x67>
    191c:   e8 6f f9 ff ff          call   1290 <__stack_chk_fail@plt>
```

输入的第一个数	输入的第二个数
0	$0x682 - 0x2cf = 947_{(10)}$
1	$0x682 - 0xf2 = 1424_{(10)}$
2	$0x682 - 0x104 = 1406_{(10)}$
3	$0x682 - 0x178 = 1290_{(10)}$
4	$0x682 - 0x19b = 1255_{(10)}$
5	$0x682 - 0x1ad = 1237_{(10)}$
6	$0x682 - 0x341 = 833_{(10)}$
7	$0x682 - 0xce = 1460_{(10)}$

跳转表的存储

```
1      .section        .rodata
2      .align 8            Align address to multiple of 8
3    .L4:
4      .quad    .L3        Case 100: loc_A
5      .quad    .L8        Case 101: loc_def
6      .quad    .L5        Case 102: loc_B
7      .quad    .L6        Case 103: loc_C
8      .quad    .L7        Case 104: loc_D
9      .quad    .L8        Case 105: loc_def
10     .quad    .L7        Case 106: loc_D
```



%rax=M[%rdx+4*%rax]+%rdx

基地址+存放在内存数组中的偏移量

节省了存储空间，但增加了计算量

phase_4

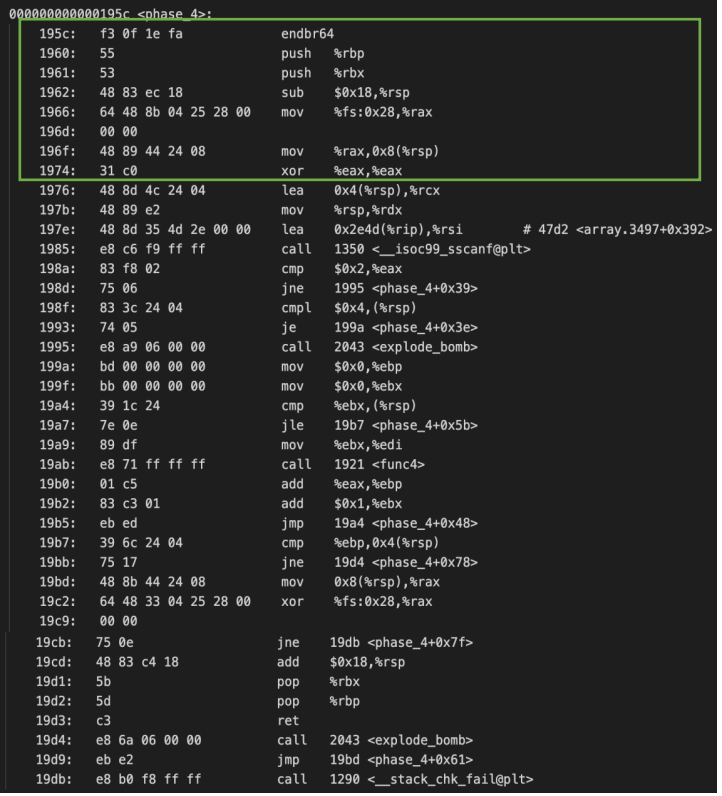
Recursion

解题思路

- 根据汇编代码易知输入字符串的格式为%d %d，且输入的第一个整数为4
- 观察func4的递归调用，将递归转化为递推可以更快地解决问题
- 根据递推公式可以手动计算需要的值
- 汇编代码通过循环计算$\sum_{i=0}^{3} func4(i)$
- 输入的第二个整数即为$\sum_{i=0}^{3} func4(i)$

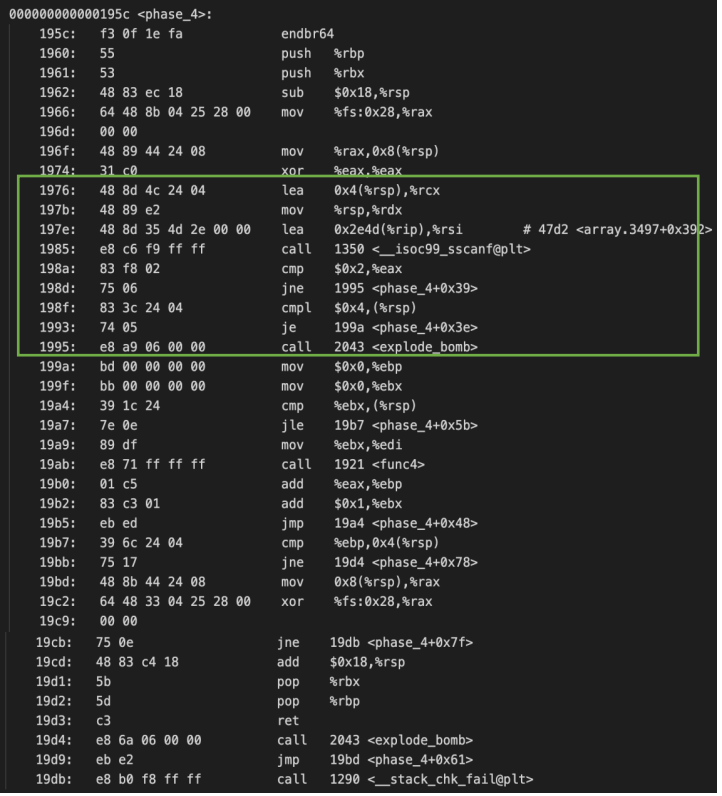
```
000000000000195c <phase_4>:
    195c:   f3 0f 1e fa             endbr64
    1960:   55                      push   %rbp
    1961:   53                      push   %rbx
    1962:   48 83 ec 18             sub    $0x18,%rsp
    1966:   64 48 8b 04 25 28 00    mov    %fs:0x28,%rax
    196d:   00 00
    196f:   48 89 44 24 08          mov    %rax,0x8(%rsp)
    1974:   31 c0                   xor    %eax,%eax
    1976:   48 8d 4c 24 04          lea    0x4(%rsp),%rcx
    197b:   48 89 e2                mov    %rsp,%rdx
    197e:   48 8d 35 4d 2e 00 00    lea    0x2e4d(%rip),%rsi        # 47d2 <array.3497+0x392>
    1985:   e8 c6 f9 ff ff          call   1350 <__isoc99_sscanf@plt>
    198a:   83 f8 02                cmp    $0x2,%eax
    198d:   75 06                   jne    1995 <phase_4+0x39>
    198f:   83 3c 24 04             cmpl   $0x4,(%rsp)
    1993:   74 05                   je     199a <phase_4+0x3e>
    1995:   e8 a9 06 00 00          call   2043 <explode_bomb>
    199a:   bd 00 00 00 00          mov    $0x0,%ebp
    199f:   bb 00 00 00 00          mov    $0x0,%ebx
    19a4:   39 1c 24                cmp    %ebx,(%rsp)
    19a7:   7e 0e                   jle    19b7 <phase_4+0x5b>
    19a9:   89 df                   mov    %ebx,%edi
    19ab:   e8 71 ff ff ff          call   1921 <func4>
    19b0:   01 c5                   add    %eax,%ebp
    19b2:   83 c3 01                add    $0x1,%ebx
    19b5:   eb ed                   jmp    19a4 <phase_4+0x48>
    19b7:   39 6c 24 04             cmp    %ebp,0x4(%rsp)
    19bb:   75 17                   jne    19d4 <phase_4+0x78>
    19bd:   48 8b 44 24 08          mov    0x8(%rsp),%rax
    19c2:   64 48 33 04 25 28 00    xor    %fs:0x28,%rax
    19c9:   00 00
    19cb:   75 0e                   jne    19db <phase_4+0x7f>
    19cd:   48 83 c4 18             add    $0x18,%rsp
    19d1:   5b                      pop    %rbx
    19d2:   5d                      pop    %rbp
    19d3:   c3                      ret
    19d4:   e8 6a 06 00 00          call   2043 <explode_bomb>
    19d9:   eb e2                   jmp    19bd <phase_4+0x61>
    19db:   e8 b0 f8 ff ff          call   1290 <__stack_chk_fail@plt>
```

```
0000000000001921 <func4>:
    1921:   f3 0f 1e fa             endbr64
    1925:   85 ff                   test   %edi,%edi
    1927:   7e 29                   jle    1952 <func4+0x31>
    1929:   55                      push   %rbp
    192a:   53                      push   %rbx
    192b:   48 83 ec 08             sub    $0x8,%rsp
    192f:   89 fb                   mov    %edi,%ebx
    1931:   83 ff 01                cmp    $0x1,%edi
    1934:   74 22                   je     1958 <func4+0x37>
    1936:   8d 7f ff                lea    -0x1(%rdi),%edi
    1939:   e8 e3 ff ff ff          call   1921 <func4>
    193e:   8d 2c 00                lea    (%rax,%rax,1),%ebp
    1941:   8d 7b fe                lea    -0x2(%rbx),%edi
    1944:   e8 d8 ff ff ff          call   1921 <func4>
    1949:   01 e8                   add    %ebp,%eax
    194b:   48 83 c4 08             add    $0x8,%rsp
    194f:   5b                      pop    %rbx
    1950:   5d                      pop    %rbp
    1951:   c3                      ret
    1952:   b8 00 00 00 00          mov    $0x0,%eax
    1957:   c3                      ret
    1958:   89 f8                   mov    %edi,%eax
    195a:   eb ef                   jmp    194b <func4+0x2a>
```



```
000000000000195c <phase_4>:
    195c:   f3 0f 1e fa             endbr64
    1960:   55                      push   %rbp
    1961:   53                      push   %rbx
    1962:   48 83 ec 18             sub    $0x18,%rsp
    1966:   64 48 8b 04 25 28 00    mov    %fs:0x28,%rax
    196d:   00 00
    196f:   48 89 44 24 08          mov    %rax,0x8(%rsp)
    1974:   31 c0                   xor    %eax,%eax
    1976:   48 8d 4c 24 04          lea    0x4(%rsp),%rcx
    197b:   48 89 e2                mov    %rsp,%rdx
    197e:   48 8d 35 4d 2e 00 00    lea    0x2e4d(%rip),%rsi        # 47d2 <array.3497+0x392>
    1985:   e8 c6 f9 ff ff          call   1350 <__isoc99_sscanf@plt>
    198a:   83 f8 02                cmp    $0x2,%eax
    198d:   75 06                   jne    1995 <phase_4+0x39>
    198f:   83 3c 24 04             cmpl   $0x4,(%rsp)
    1993:   74 05                   je     199a <phase_4+0x3e>
    1995:   e8 a9 06 00 00          call   2043 <explode_bomb>
    199a:   bd 00 00 00 00          mov    $0x0,%ebp
    199f:   bb 00 00 00 00          mov    $0x0,%ebx
    19a4:   39 1c 24                cmp    %ebx,(%rsp)
    19a7:   7e 0e                   jle    19b7 <phase_4+0x5b>
    19a9:   89 df                   mov    %ebx,%edi
    19ab:   e8 71 ff ff ff          call   1921 <func4>
    19b0:   01 c5                   add    %eax,%ebp
    19b2:   83 c3 01                add    $0x1,%ebx
    19b5:   eb ed                   jmp    19a4 <phase_4+0x48>
    19b7:   39 6c 24 04             cmp    %ebp,0x4(%rsp)
    19bb:   75 17                   jne    19d4 <phase_4+0x78>
    19bd:   48 8b 44 24 08          mov    0x8(%rsp),%rax
    19c2:   64 48 33 04 25 28 00    xor    %fs:0x28,%rax
    19c9:   00 00
    19cb:   75 0e                   jne    19db <phase_4+0x7f>
    19cd:   48 83 c4 18             add    $0x18,%rsp
    19d1:   5b                      pop    %rbx
    19d2:   5d                      pop    %rbp
    19d3:   c3                      ret
    19d4:   e8 6a 06 00 00          call   2043 <explode_bomb>
    19d9:   eb e2                   jmp    19bd <phase_4+0x61>
    19db:   e8 b0 f8 ff ff          call   1290 <__stack_chk_fail@plt>
```

```
000000000000195c <phase_4>:
    195c:   f3 0f 1e fa             endbr64
    1960:   55                      push   %rbp             #被调用者保存
    1961:   53                      push   %rbx             #被调用者保存
    1962:   48 83 ec 18             sub    $0x18,%rsp
    1966:   64 48 8b 04 25 28 00    mov    %fs:0x28,%rax    #栈溢出检测
    196d:   00 00
    196f:   48 89 44 24 08          mov    %rax,0x8(%rsp)   #M[rsp+8]=rax
    1974:   31 c0                   xor    %eax,%eax        #rax=0
```



```
000000000000195c <phase_4>:
    195c:   f3 0f 1e fa             endbr64
    1960:   55                      push   %rbp
    1961:   53                      push   %rbx
    1962:   48 83 ec 18             sub    $0x18,%rsp
    1966:   64 48 8b 04 25 28 00    mov    %fs:0x28,%rax
    196d:   00 00
    196f:   48 89 44 24 08          mov    %rax,0x8(%rsp)
    1974:   31 c0                   xor    %eax,%eax
    1976:   48 8d 4c 24 04          lea    0x4(%rsp),%rcx
    197b:   48 89 e2                mov    %rsp,%rdx
    197e:   48 8d 35 4d 2e 00 00    lea    0x2e4d(%rip),%rsi        # 47d2 <array.3497+0x392>
    1985:   e8 c6 f9 ff ff          call   1350 <__isoc99_sscanf@plt>
    198a:   83 f8 02                cmp    $0x2,%eax
    198d:   75 06                   jne    1995 <phase_4+0x39>
    198f:   83 3c 24 04             cmpl   $0x4,(%rsp)
    1993:   74 05                   je     199a <phase_4+0x3e>
    1995:   e8 a9 06 00 00          call   2043 <explode_bomb>
    199a:   bd 00 00 00 00          mov    $0x0,%ebp
    199f:   bb 00 00 00 00          mov    $0x0,%ebx
    19a4:   39 1c 24                cmp    %ebx,(%rsp)
    19a7:   7e 0e                   jle    19b7 <phase_4+0x5b>
    19a9:   89 df                   mov    %ebx,%edi
    19ab:   e8 71 ff ff ff          call   1921 <func4>
    19b0:   01 c5                   add    %eax,%ebp
    19b2:   83 c3 01                add    $0x1,%ebx
    19b5:   eb ed                   jmp    19a4 <phase_4+0x48>
    19b7:   39 6c 24 04             cmp    %ebp,0x4(%rsp)
    19bb:   75 17                   jne    19d4 <phase_4+0x78>
    19bd:   48 8b 44 24 08          mov    0x8(%rsp),%rax
    19c2:   64 48 33 04 25 28 00    xor    %fs:0x28,%rax
    19c9:   00 00
    19cb:   75 0e                   jne    19db <phase_4+0x7f>
    19cd:   48 83 c4 18             add    $0x18,%rsp
    19d1:   5b                      pop    %rbx
    19d2:   5d                      pop    %rbp
    19d3:   c3                      ret
    19d4:   e8 6a 06 00 00          call   2043 <explode_bomb>
    19d9:   eb e2                   jmp    19bd <phase_4+0x61>
    19db:   e8 b0 f8 ff ff          call   1290 <__stack_chk_fail@plt>
```

```
1976:   48 8d 4c 24 04          lea    0x4(%rsp),%rcx                          #rcx=rsp+4  第四个参数
197b:   48 89 e2                mov    %rsp,%rdx                               #rdx=rsp    第三个参数
197e:   48 8d 35 4d 2e 00 00    lea    0x2e4d(%rip),%rsi        # 47d2 <array.3497+0x392>  #rsi=rip+0x2e4d "%d %d"
1985:   e8 c6 f9 ff ff          call   1350 <__isoc99_sscanf@plt>              #读取两个整数
198a:   83 f8 02                cmp    $0x2,%eax                               #rax:2
198d:   75 06                   jne    1995 <phase_4+0x39>                     #rax==2
198f:   83 3c 24 04             cmpl   $0x4,(%rsp)                             #M[rsp]:4 输入的第一个数是4
1993:   74 05                   je     199a <phase_4+0x3e>
1995:   e8 a9 06 00 00          call   2043 <explode_bomb>
```

```
000000000000195c <phase_4>:
    195c:   f3 0f 1e fa             endbr64
    1960:   55                      push   %rbp
    1961:   53                      push   %rbx
    1962:   48 83 ec 18             sub    $0x18,%rsp
    1966:   64 48 8b 04 25 28 00    mov    %fs:0x28,%rax
    196d:   00 00
    196f:   48 89 44 24 08          mov    %rax,0x8(%rsp)
    1974:   31 c0                   xor    %eax,%eax
    1976:   48 8d 4c 24 04          lea    0x4(%rsp),%rcx
    197b:   48 89 e2                mov    %rsp,%rdx
    197e:   48 8d 35 4d 2e 00 00    lea    0x2e4d(%rip),%rsi        # 47d2 <array.3497+0x392>
    1985:   e8 c6 f9 ff ff          call   1350 <__isoc99_sscanf@plt>
    198a:   83 f8 02                cmp    $0x2,%eax
    198d:   75 06                   jne    1995 <phase_4+0x39>
    198f:   83 3c 24 04             cmpl   $0x4,(%rsp)
    1993:   74 05                   je     199a <phase_4+0x3e>
    1995:   e8 a9 06 00 00          call   2043 <explode_bomb>
    199a:   bd 00 00 00 00          mov    $0x0,%ebp
    199f:   bb 00 00 00 00          mov    $0x0,%ebx
    19a4:   39 1c 24                cmp    %ebx,(%rsp)
    19a7:   7e 0e                   jle    19b7 <phase_4+0x5b>
    19a9:   89 df                   mov    %ebx,%edi
    19ab:   e8 71 ff ff ff          call   1921 <func4>
    19b0:   01 c5                   add    %eax,%ebp
    19b2:   83 c3 01                add    $0x1,%ebx
    19b5:   eb ed                   jmp    19a4 <phase_4+0x48>
    19b7:   39 6c 24 04             cmp    %ebp,0x4(%rsp)
    19bb:   75 17                   jne    19d4 <phase_4+0x78>
    19bd:   48 8b 44 24 08          mov    0x8(%rsp),%rax
    19c2:   64 48 33 04 25 28 00    xor    %fs:0x28,%rax
    19c9:   00 00
    19cb:   75 0e                   jne    19db <phase_4+0x7f>
    19cd:   48 83 c4 18             add    $0x18,%rsp
    19d1:   5b                      pop    %rbx
    19d2:   5d                      pop    %rbp
    19d3:   c3                      ret
    19d4:   e8 6a 06 00 00          call   2043 <explode_bomb>
    19d9:   eb e2                   jmp    19bd <phase_4+0x61>
    19db:   e8 b0 f8 ff ff          call   1290 <__stack_chk_fail@plt>
```

```
199a:   bd 00 00 00 00          mov    $0x0,%ebp            #ebp=0
199f:   bb 00 00 00 00          mov    $0x0,%ebx            #ebx=0
19a4:   39 1c 24                cmp    %ebx,(%rsp)          #M[rsp]:ebx   M[rsp]:0        M[rsp]:1        M[rsp]:2        M[rsp]:3        M[rsp]:4
19a7:   7e 0e                   jle    19b7 <phase_4+0x5b>
19a9:   89 df                   mov    %ebx,%edi            #edi=ebx      rdi=ebx(0)      rdi=1           rdi=2           rdi=3
19ab:   e8 71 ff ff ff          call   1921 <func4>         #call         rax=0           rax=1           rax=2           rax=5
19b0:   01 c5                   add    %eax,%ebp            #ebp+=eax     ebp=0           ebp=1           ebp=3           ebp=8
19b2:   83 c3 01                add    $0x1,%ebx            #ebx+=1       ebx=1           ebx=2           ebx=3           ebx=4
19b5:   eb ed                   jmp    19a4 <phase_4+0x48>
19b7:   39 6c 24 04             cmp    %ebp,0x4(%rsp)       #M[rsp+0x4]:rbp  输入的第二个数是8=0+1+2+5
19bb:   75 17                   jne    19d4 <phase_4+0x78>
```

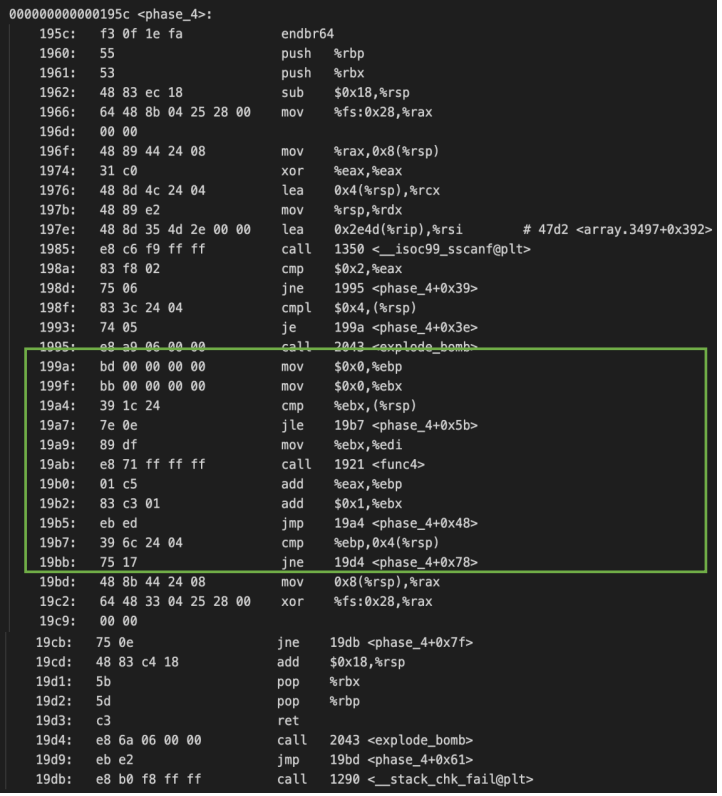
```
000000000000195c <phase_4>:
    195c:   f3 0f 1e fa             endbr64
    1960:   55                      push   %rbp
    1961:   53                      push   %rbx
    1962:   48 83 ec 18             sub    $0x18,%rsp
    1966:   64 48 8b 04 25 28 00    mov    %fs:0x28,%rax
    196d:   00 00
    196f:   48 89 44 24 08          mov    %rax,0x8(%rsp)
    1974:   31 c0                   xor    %eax,%eax
    1976:   48 8d 4c 24 04          lea    0x4(%rsp),%rcx
    197b:   48 89 e2                mov    %rsp,%rdx
    197e:   48 8d 35 4d 2e 00 00    lea    0x2e4d(%rip),%rsi        # 47d2 <array.3497+0x392>
    1985:   e8 c6 f9 ff ff          call   1350 <__isoc99_sscanf@plt>
    198a:   83 f8 02                cmp    $0x2,%eax
    198d:   75 06                   jne    1995 <phase_4+0x39>
    198f:   83 3c 24 04             cmpl   $0x4,(%rsp)
    1993:   74 05                   je     199a <phase_4+0x3e>
    1995:   e8 a9 06 00 00          call   2043 <explode_bomb>
    199a:   bd 00 00 00 00          mov    $0x0,%ebp
    199f:   bb 00 00 00 00          mov    $0x0,%ebx
    19a4:   39 1c 24                cmp    %ebx,(%rsp)
    19a7:   7e 0e                   jle    19b7 <phase_4+0x5b>
    19a9:   89 df                   mov    %ebx,%edi
    19ab:   e8 71 ff ff ff          call   1921 <func4>
    19b0:   01 c5                   add    %eax,%ebp
    19b2:   83 c3 01                add    $0x1,%ebx
    19b5:   eb ed                   jmp    19a4 <phase_4+0x48>
    19b7:   39 6c 24 04             cmp    %ebp,0x4(%rsp)
    19bb:   75 17                   jne    19d4 <phase_4+0x78>
    19bd:   48 8b 44 24 08          mov    0x8(%rsp),%rax
    19c2:   64 48 33 04 25 28 00    xor    %fs:0x28,%rax
    19c9:   00 00
    19cb:   75 0e                   jne    19db <phase_4+0x7f>
    19cd:   48 83 c4 18             add    $0x18,%rsp
    19d1:   5b                      pop    %rbx
    19d2:   5d                      pop    %rbp
    19d3:   c3                      ret
    19d4:   e8 6a 06 00 00          call   2043 <explode_bomb>
    19d9:   eb e2                   jmp    19bd <phase_4+0x61>
    19db:   e8 b0 f8 ff ff          call   1290 <__stack_chk_fail@plt>
```

```
0000000000001921 <func4>:
    1921:   f3 0f 1e fa             endbr64
    1925:   85 ff                   test   %edi,%edi
    1927:   7e 29                   jle    1952 <func4+0x31>        #edi<=0,return 0
    1929:   55                      push   %rbp                     #被调用者保存
    192a:   53                      push   %rbx                     #被调用者保存
    192b:   48 83 ec 08             sub    $0x8,%rsp
    192f:   89 fb                   mov    %edi,%ebx                #ebx=edi
    1931:   83 ff 01                cmp    $0x1,%edi                #edi:1
    1934:   74 22                   je     1958 <func4+0x37>        #edi==1,return 1
    1936:   8d 7f ff                lea    -0x1(%rdi),%edi          #edi=rdi-1
    1939:   e8 e3 ff ff ff          call   1921 <func4>             #rax
    193e:   8d 2c 00                lea    (%rax,%rax,1),%ebp       #ebp=2*rax=2*func4(rdi-1)
    1941:   8d 7b fe                lea    -0x2(%rbx),%edi          #edi=rbx-2
    1944:   e8 d8 ff ff ff          call   1921 <func4>             #rax
    1949:   01 e8                   add    %ebp,%eax                #eax+=ebp,eax=2*func4(rdi-1)+func4(rdi-2)
    194b:   48 83 c4 08             add    $0x8,%rsp
    194f:   5b                      pop    %rbx
    1950:   5d                      pop    %rbp
    1951:   c3                      ret
    1952:   b8 00 00 00 00          mov    $0x0,%eax
    1957:   c3                      ret
    1958:   89 f8                   mov    %edi,%eax
    195a:   eb ef                   jmp    194b <func4+0x2a>
```

$$func4(x) = func4(x-2) + 2 * func4(x-1), x \geq 2$$

$$func4(0) = 0, func4(1) = 1$$

$$func4(2) = 2, func4(3) = 5$$



```
000000000000195c <phase_4>:
   195c:   f3 0f 1e fa             endbr64
   1960:   55                      push   %rbp
   1961:   53                      push   %rbx
   1962:   48 83 ec 18             sub    $0x18,%rsp
   1966:   64 48 8b 04 25 28 00    mov    %fs:0x28,%rax
   196d:   00 00
   196f:   48 89 44 24 08          mov    %rax,0x8(%rsp)
   1974:   31 c0                   xor    %eax,%eax
   1976:   48 8d 4c 24 04          lea    0x4(%rsp),%rcx
   197b:   48 89 e2                mov    %rsp,%rdx
   197e:   48 8d 35 4d 2e 00 00    lea    0x2e4d(%rip),%rsi        # 47d2 <array.3497+0x392>
   1985:   e8 c6 f9 ff ff          call   1350 <__isoc99_sscanf@plt>
   198a:   83 f8 02                cmp    $0x2,%eax
   198d:   75 06                   jne    1995 <phase_4+0x39>
   198f:   83 3c 24 04             cmpl   $0x4,(%rsp)
   1993:   74 05                   je     199a <phase_4+0x3e>
   1995:   e8 a9 06 00 00          call   2043 <explode_bomb>
   199a:   bd 00 00 00 00          mov    $0x0,%ebp
   199f:   bb 00 00 00 00          mov    $0x0,%ebx
   19a4:   39 1c 24                cmp    %ebx,(%rsp)
   19a7:   7e 0e                   jle    19b7 <phase_4+0x5b>
   19a9:   89 df                   mov    %ebx,%edi
   19ab:   e8 71 ff ff ff          call   1921 <func4>
   19b0:   01 c5                   add    %eax,%ebp
   19b2:   83 c3 01                add    $0x1,%ebx
   19b5:   eb ed                   jmp    19a4 <phase_4+0x48>
   19b7:   39 6c 24 04             cmp    %ebp,0x4(%rsp)
   19bb:   75 17                   jne    19d4 <phase_4+0x78>
   19bd:   48 8b 44 24 08          mov    0x8(%rsp),%rax
   19c2:   64 48 33 04 25 28 00    xor    %fs:0x28,%rax
   19c9:   00 00
   19cb:   75 0e                   jne    19db <phase_4+0x7f>
   19cd:   48 83 c4 18             add    $0x18,%rsp
   19d1:   5b                      pop    %rbx
   19d2:   5d                      pop    %rbp
   19d3:   c3                      ret
   19d4:   e8 6a 06 00 00          call   2043 <explode_bomb>
   19d9:   eb e2                   jmp    19bd <phase_4+0x61>
   19db:   e8 b0 f8 ff ff          call   1290 <__stack_chk_fail@plt>
```

$$func4(x) = func4(x-2) + 2 * func4(x-1), x \geq 2$$

$$func4(0) = 0, func4(1) = 1$$

$$func4(2) = 2, func4(3) = 5$$

输入的第二个数=0+1+2+5=8

答案：4 8

```
199a:   bd 00 00 00 00          mov    $0x0,%ebp            #ebp=0
199f:   bb 00 00 00 00          mov    $0x0,%ebx            #ebx=0
19a4:   39 1c 24                cmp    %ebx,(%rsp)          #M[rsp]:ebx   M[rsp]:0        M[rsp]:1        M[rsp]:2        M[rsp]:3        M[rsp]:4
19a7:   7e 0e                   jle    19b7 <phase_4+0x5b>
19a9:   89 df                   mov    %ebx,%edi            #edi=ebx      rdi=ebx(0)      rdi=1           rdi=2           rdi=3
19ab:   e8 71 ff ff ff          call   1921 <func4>         #call         rax=0           rax=1           rax=2           rax=5
19b0:   01 c5                   add    %eax,%ebp            #ebp+=eax     ebp=0           ebp=1           ebp=3           ebp=8
19b2:   83 c3 01                add    $0x1,%ebx            #ebx+=1       ebx=1           ebx=2           ebx=3           ebx=4
19b5:   eb ed                   jmp    19a4 <phase_4+0x48>
19b7:   39 6c 24 04             cmp    %ebp,0x4(%rsp)       #M[rsp+0x4]:rbp  输入的第二个数是8=0+1+2+5
19bb:   75 17                   jne    19d4 <phase_4+0x78>
```

递归调用的运行时栈 以phase_4调用func4(3)为例

并不是每一个x86-64过程都会在栈上分配空间

```
1939:   e8 e3 ff ff ff          call   1921 <func4>              #rax
193e:   8d 2c 00                lea    (%rax,%rax,1),%ebp       #ebp=2*rax=2*func4(rdi-1)

0000000000001921 <func4>:
   1921:   f3 0f 1e fa          endbr64
   1925:   85 ff                test   %edi,%edi
   1927:   7e 29                jle    1952 <func4+0x31>        #edi<=0,return 0
   1929:   55                   push   %rbp                     #被调用者保存
   192a:   53                   push   %rbx                     #被调用者保存
   192b:   48 83 ec 08          sub    $0x8,%rsp
   192f:   89 fb                mov    %edi,%ebx                #ebx=edi
   1931:   83 ff 01             cmp    $0x1,%edi                #edi:1
   1934:   74 22                je     1958 <func4+0x37>        #edi==1,return 1
   1936:   8d 7f ff             lea    -0x1(%rdi),%edi          #edi=rdi-1
   1939:   e8 e3 ff ff ff       call   1921 <func4>             #rax
   193e:   8d 2c 00             lea    (%rax,%rax,1),%ebp       #ebp=2*rax=2*func4(rdi-1)
   1941:   8d 7b fe             lea    -0x2(%rbx),%edi          #edi=rbx-2
   1944:   e8 d8 ff ff ff       call   1921 <func4>             #rax
   1949:   01 e8                add    %ebp,%eax                #eax+=ebp,eax=2*func4(rdi-1)+func4(rdi-2)
   194b:   48 83 c4 08          add    $0x8,%rsp
   194f:   5b                   pop    %rbx
   1950:   5d                   pop    %rbp
   1951:   c3                   ret
   1952:   b8 00 00 00 00       mov    $0x0,%eax
   1957:   c3                   ret
   1958:   89 f8                mov    %edi,%eax
   195a:   eb ef                jmp    194b <func4+0x2a>
```

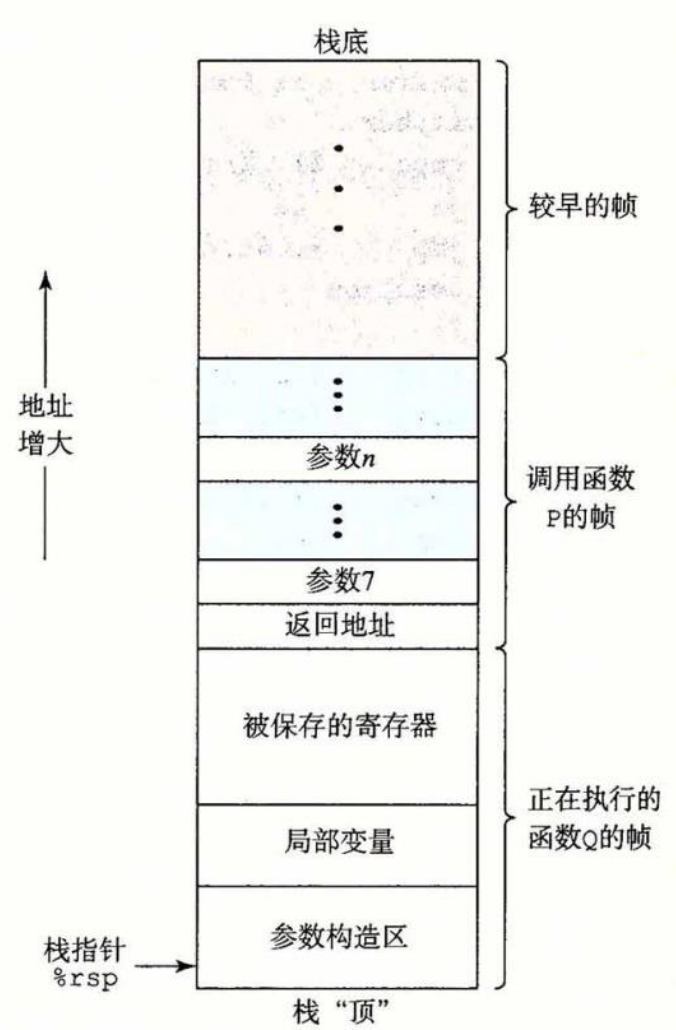
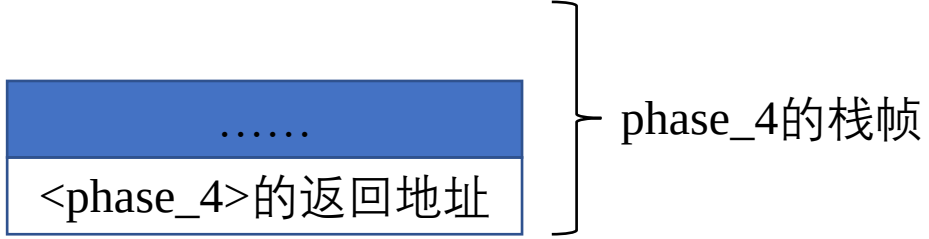


图 3-25 通用的栈帧结构(栈用来传递参数、存储返回信息、保存寄存器，以及局部存储。省略了不必要的部分)

递归调用的运行时栈

以phase_4调用func4(3)为例



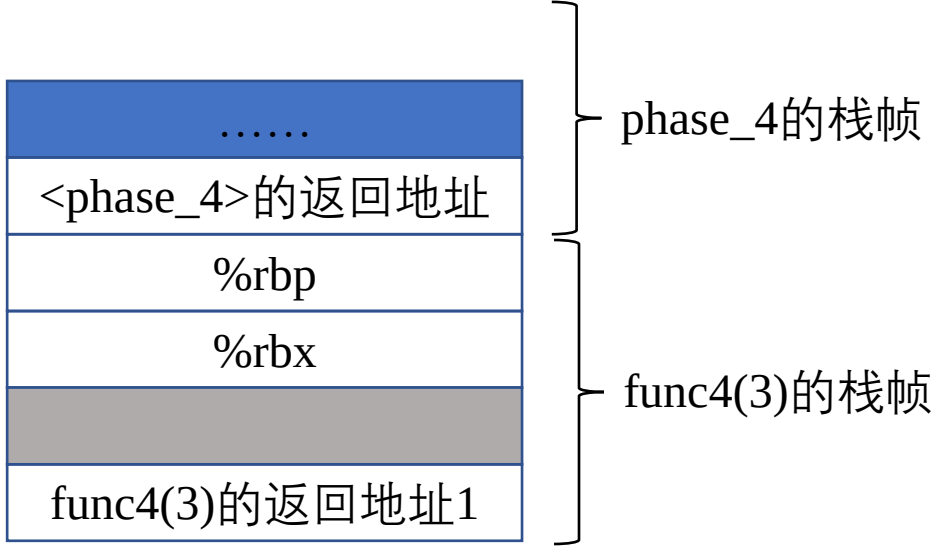
并不是每一个x86-64过程都会在栈上分配空间

```
1939:   e8 e3 ff ff ff          call   1921 <func4>            #rax
193e:   8d 2c 00                lea    (%rax,%rax,1),%ebp      #ebp=2*rax=2*func4(rdi-1)

0000000000001921 <func4>:
   1921:   f3 0f 1e fa          endbr64
   1925:   85 ff                test   %edi,%edi
   1927:   7e 29                jle    1952 <func4+0x31>       #edi<=0,return 0
   1929:   55                   push   %rbp                    #被调用者保存
   192a:   53                   push   %rbx                    #被调用者保存
   192b:   48 83 ec 08          sub    $0x8,%rsp
   192f:   89 fb                mov    %edi,%ebx               #ebx=edi
   1931:   83 ff 01             cmp    $0x1,%edi               #edi:1
   1934:   74 22                je     1958 <func4+0x37>       #edi==1,return 1
   1936:   8d 7f ff             lea    -0x1(%rdi),%edi         #edi=rdi-1
   1939:   e8 e3 ff ff ff       call   1921 <func4>            #rax
   193e:   8d 2c 00             lea    (%rax,%rax,1),%ebp      #ebp=2*rax=2*func4(rdi-1)
   1941:   8d 7b fe             lea    -0x2(%rbx),%edi         #edi=rbx-2
   1944:   e8 d8 ff ff ff       call   1921 <func4>            #rax
   1949:   01 e8                add    %ebp,%eax               #eax+=ebp,eax=2*func4(rdi-1)+func4(rdi-2)
   194b:   48 83 c4 08          add    $0x8,%rsp
   194f:   5b                   pop    %rbx
   1950:   5d                   pop    %rbp
   1951:   c3                   ret
   1952:   b8 00 00 00 00       mov    $0x0,%eax
   1957:   c3                   ret
   1958:   89 f8                mov    %edi,%eax
   195a:   eb ef                jmp    194b <func4+0x2a>
```

递归调用的运行时栈 以phase_4调用func4(3)为例

并不是每一个x86-64过程都会在栈上分配空间

```
1939:   e8 e3 ff ff ff          call   1921 <func4>            #rax
193e:   8d 2c 00                lea    (%rax,%rax,1),%ebp      #ebp=2*rax=2*func4(rdi-1)

0000000000001921 <func4>:
   1921:   f3 0f 1e fa          endbr64
   1925:   85 ff                test   %edi,%edi
   1927:   7e 29                jle    1952 <func4+0x31>       #edi<=0,return 0
   1929:   55                   push   %rbp                    #被调用者保存
   192a:   53                   push   %rbx                    #被调用者保存
   192b:   48 83 ec 08          sub    $0x8,%rsp
   192f:   89 fb                mov    %edi,%ebx               #ebx=edi
   1931:   83 ff 01             cmp    $0x1,%edi               #edi:1
   1934:   74 22                je     1958 <func4+0x37>       #edi==1,return 1
   1936:   8d 7f ff             lea    -0x1(%rdi),%edi         #edi=rdi-1
   1939:   e8 e3 ff ff ff       call   1921 <func4>            #rax
   193e:   8d 2c 00             lea    (%rax,%rax,1),%ebp      #ebp=2*rax=2*func4(rdi-1)
   1941:   8d 7b fe             lea    -0x2(%rbx),%edi         #edi=rbx-2
   1944:   e8 d8 ff ff ff       call   1921 <func4>            #rax
   1949:   01 e8                add    %ebp,%eax               #eax+=ebp,eax=2*func4(rdi-1)+func4(rdi-2)
   194b:   48 83 c4 08          add    $0x8,%rsp
   194f:   5b                   pop    %rbx
   1950:   5d                   pop    %rbp
   1951:   c3                   ret
   1952:   b8 00 00 00 00       mov    $0x0,%eax
   1957:   c3                   ret
   1958:   89 f8                mov    %edi,%eax
   195a:   eb ef                jmp    194b <func4+0x2a>
```

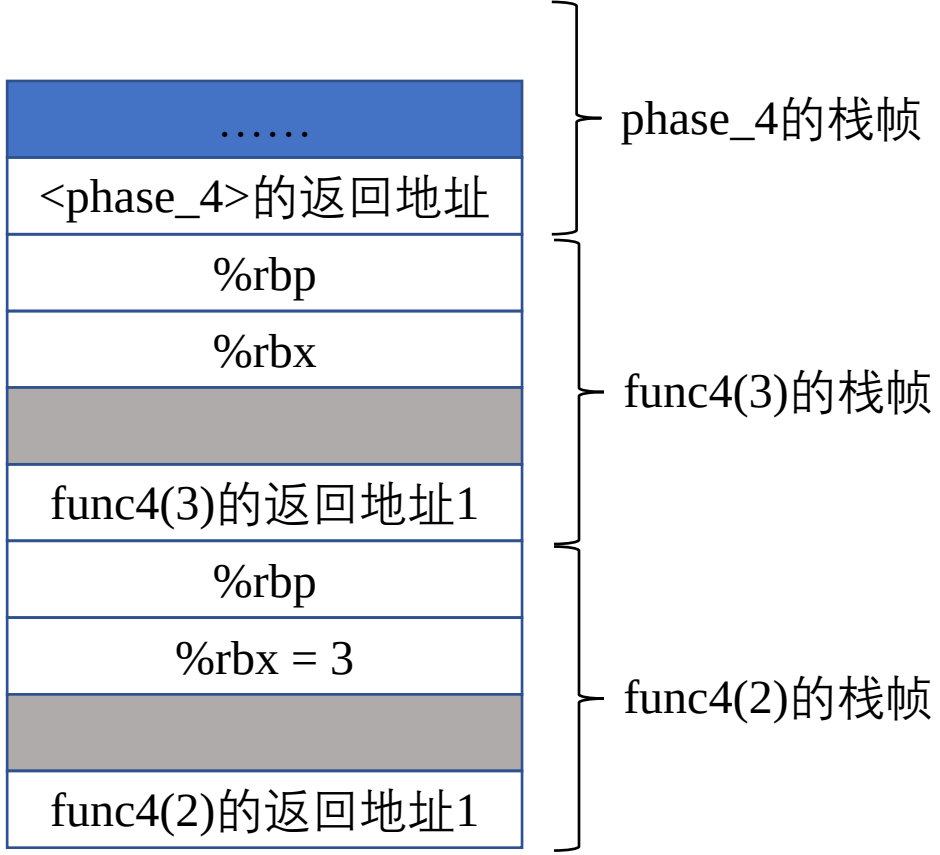


递归调用的运行时栈

以phase_4调用func4(3)为例

并不是每一个x86-64过程都会在栈上分配空间

```
1939:   e8 e3 ff ff ff          call   1921 <func4>              #rax
193e:   8d 2c 00                lea    (%rax,%rax,1),%ebp       #ebp=2*rax=2*func4(rdi-1)

0000000000001921 <func4>:
   1921:   f3 0f 1e fa          endbr64
   1925:   85 ff                test   %edi,%edi
   1927:   7e 29                jle    1952 <func4+0x31>        #edi<=0,return 0
   1929:   55                   push   %rbp                     #被调用者保存
   192a:   53                   push   %rbx                     #被调用者保存
   192b:   48 83 ec 08          sub    $0x8,%rsp
   192f:   89 fb                mov    %edi,%ebx                #ebx=edi
   1931:   83 ff 01             cmp    $0x1,%edi                #edi:1
   1934:   74 22                je     1958 <func4+0x37>        #edi==1,return 1
   1936:   8d 7f ff             lea    -0x1(%rdi),%edi          #edi=rdi-1
   1939:   e8 e3 ff ff ff       call   1921 <func4>             #rax
   193e:   8d 2c 00             lea    (%rax,%rax,1),%ebp       #ebp=2*rax=2*func4(rdi-1)
   1941:   8d 7b fe             lea    -0x2(%rbx),%edi          #edi=rbx-2
   1944:   e8 d8 ff ff ff       call   1921 <func4>             #rax
   1949:   01 e8                add    %ebp,%eax                #eax+=ebp,eax=2*func4(rdi-1)+func4(rdi-2)
   194b:   48 83 c4 08          add    $0x8,%rsp
   194f:   5b                   pop    %rbx
   1950:   5d                   pop    %rbp
   1951:   c3                   ret
   1952:   b8 00 00 00 00       mov    $0x0,%eax
   1957:   c3                   ret
   1958:   89 f8                mov    %edi,%eax
   195a:   eb ef                jmp    194b <func4+0x2a>
```

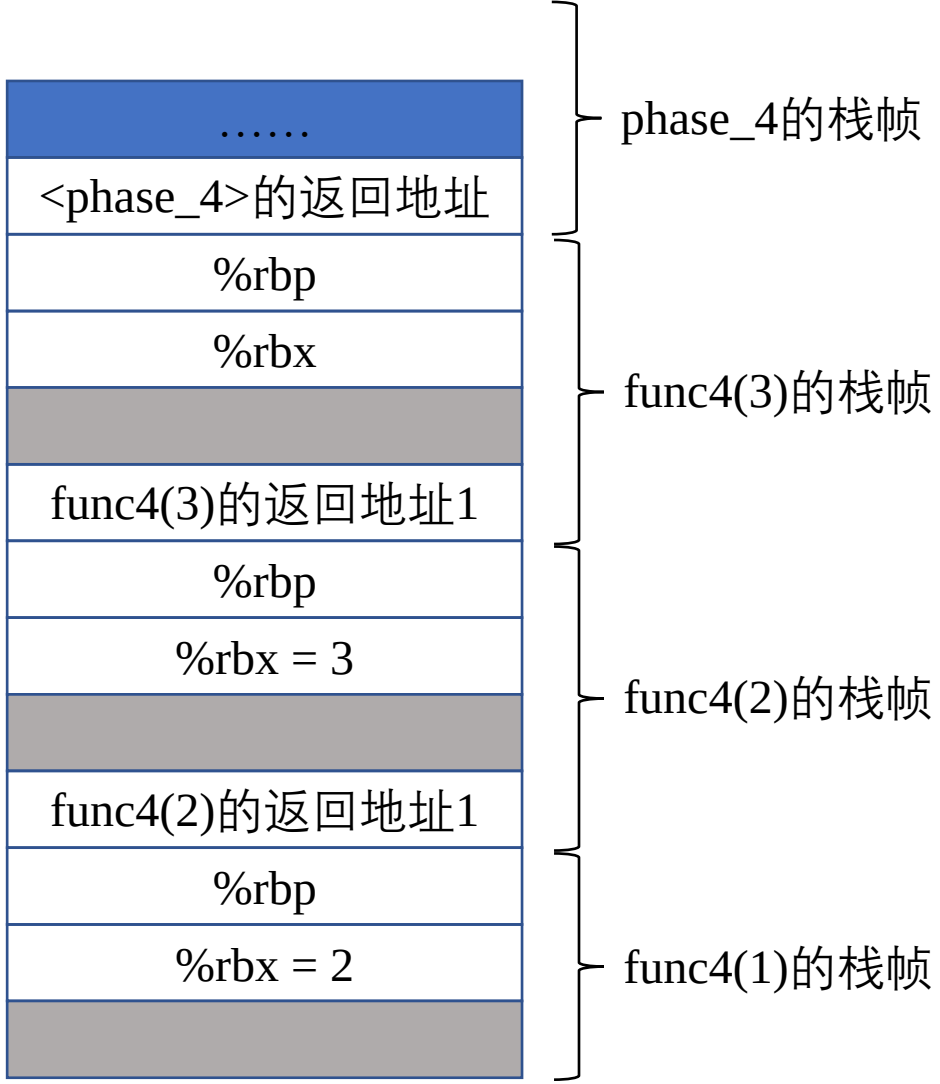


递归调用的运行时栈

以phase_4调用func4(3)为例

并不是每一个x86-64过程都会在栈上分配空间

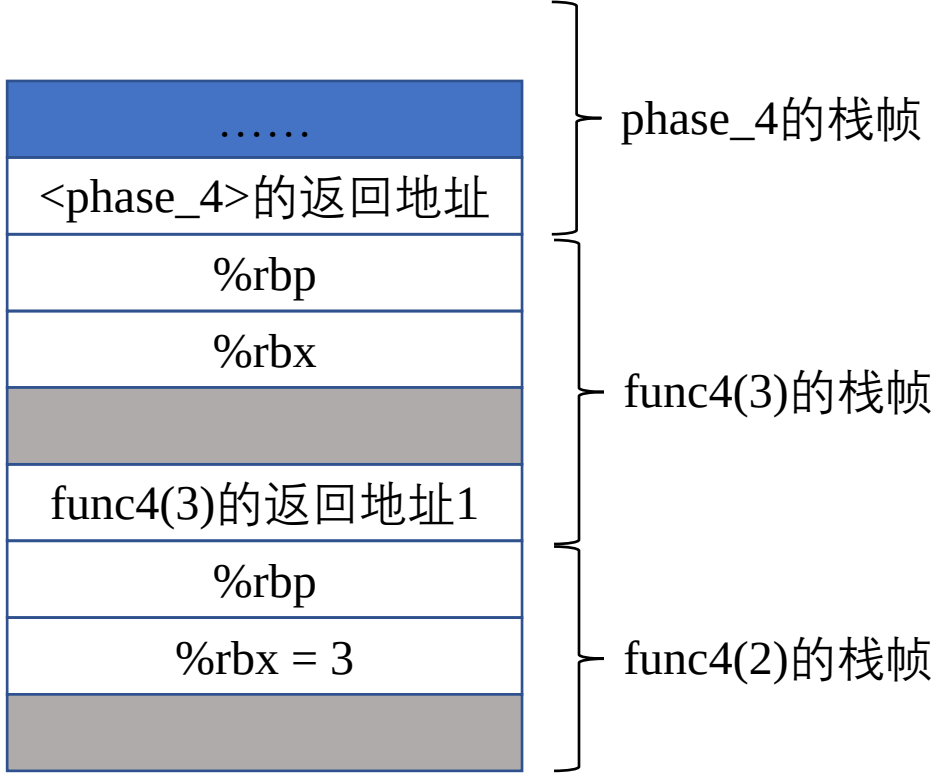
```
1939:   e8 e3 ff ff ff          call   1921 <func4>              #rax
193e:   8d 2c 00                lea    (%rax,%rax,1),%ebp       #ebp=2*rax=2*func4(rdi-1)

0000000000001921 <func4>:
   1921:   f3 0f 1e fa          endbr64
   1925:   85 ff                test   %edi,%edi
   1927:   7e 29                jle    1952 <func4+0x31>        #edi<=0,return 0
   1929:   55                   push   %rbp                     #被调用者保存
   192a:   53                   push   %rbx                     #被调用者保存
   192b:   48 83 ec 08          sub    $0x8,%rsp
   192f:   89 fb                mov    %edi,%ebx                #ebx=edi
   1931:   83 ff 01             cmp    $0x1,%edi                #edi:1
   1934:   74 22                je     1958 <func4+0x37>        #edi==1,return 1
   1936:   8d 7f ff             lea    -0x1(%rdi),%edi          #edi=rdi-1
   1939:   e8 e3 ff ff ff       call   1921 <func4>             #rax
   193e:   8d 2c 00             lea    (%rax,%rax,1),%ebp       #ebp=2*rax=2*func4(rdi-1)
   1941:   8d 7b fe             lea    -0x2(%rbx),%edi          #edi=rbx-2
   1944:   e8 d8 ff ff ff       call   1921 <func4>             #rax
   1949:   01 e8                add    %ebp,%eax                #eax+=ebp,eax=2*func4(rdi-1)+func4(rdi-2)
   194b:   48 83 c4 08          add    $0x8,%rsp
   194f:   5b                   pop    %rbx
   1950:   5d                   pop    %rbp
   1951:   c3                   ret
   1952:   b8 00 00 00 00       mov    $0x0,%eax
   1957:   c3                   ret
   1958:   89 f8                mov    %edi,%eax
   195a:   eb ef                jmp    194b <func4+0x2a>
```



递归调用的运行时栈 以phase_4调用func4(3)为例

并不是每一个x86-64过程都会在栈上分配空间

```
1939:   e8 e3 ff ff ff          call   1921 <func4>              #rax
193e:   8d 2c 00                lea    (%rax,%rax,1),%ebp       #ebp=2*rax=2*func4(rdi-1)

0000000000001921 <func4>:
   1921:   f3 0f 1e fa             endbr64
   1925:   85 ff                   test   %edi,%edi
   1927:   7e 29                   jle    1952 <func4+0x31>        #edi<=0,return 0
   1929:   55                      push   %rbp                     #被调用者保存
   192a:   53                      push   %rbx                     #被调用者保存
   192b:   48 83 ec 08             sub    $0x8,%rsp
   192f:   89 fb                   mov    %edi,%ebx                #ebx=edi
   1931:   83 ff 01                cmp    $0x1,%edi                #edi:1
   1934:   74 22                   je     1958 <func4+0x37>        #edi==1,return 1
   1936:   8d 7f ff                lea    -0x1(%rdi),%edi          #edi=rdi-1
   1939:   e8 e3 ff ff ff          call   1921 <func4>             #rax
   193e:   8d 2c 00                lea    (%rax,%rax,1),%ebp       #ebp=2*rax=2*func4(rdi-1)
   1941:   8d 7b fe                lea    -0x2(%rbx),%edi          #edi=rbx-2
   1944:   e8 d8 ff ff ff          call   1921 <func4>             #rax
   1949:   01 e8                   add    %ebp,%eax                #eax+=ebp,eax=2*func4(rdi-1)+func4(rdi-2)
   194b:   48 83 c4 08             add    $0x8,%rsp
   194f:   5b                      pop    %rbx
   1950:   5d                      pop    %rbp
   1951:   c3                      ret
   1952:   b8 00 00 00 00          mov    $0x0,%eax
   1957:   c3                      ret
   1958:   89 f8                   mov    %edi,%eax
   195a:   eb ef                   jmp    194b <func4+0x2a>
```

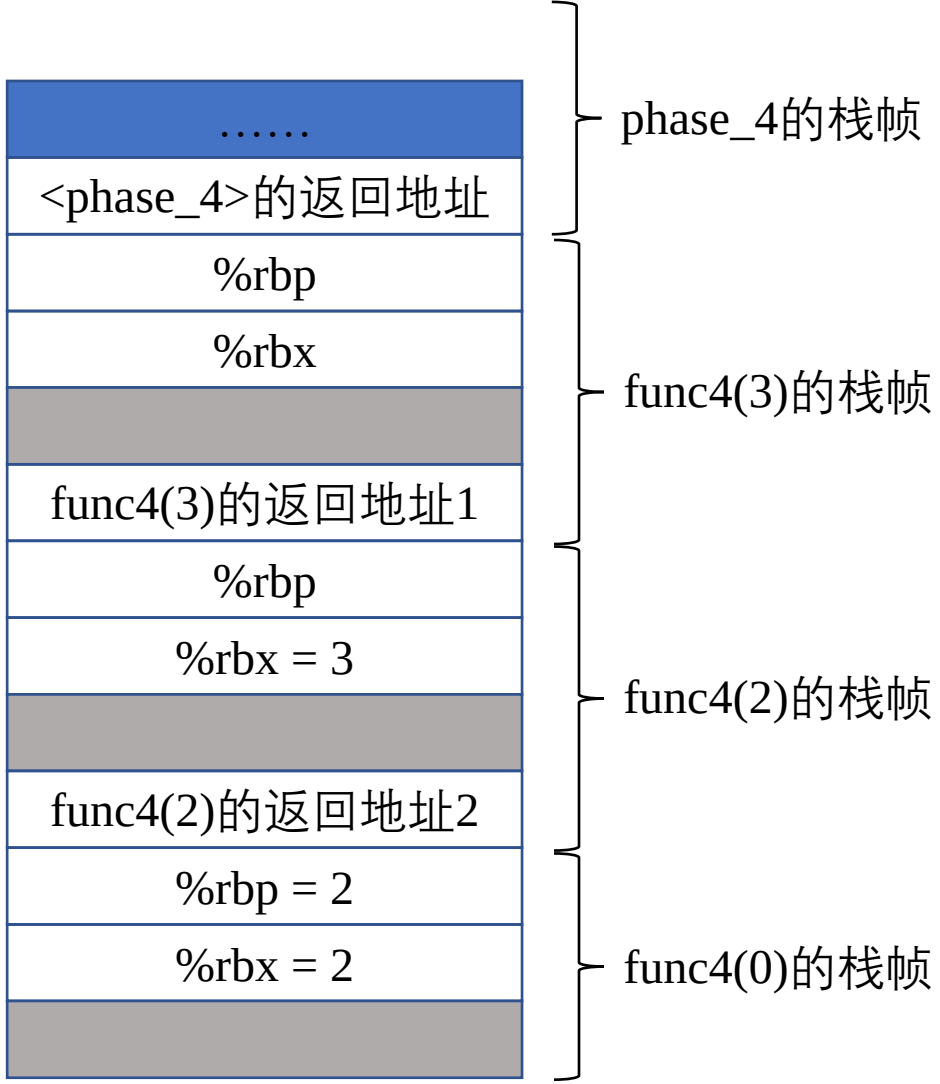


递归调用的运行时栈

以phase_4调用func4(3)为例

并不是每一个x86-64过程都会在栈上分配空间

```
1939:   e8 e3 ff ff ff          call   1921 <func4>              #rax
193e:   8d 2c 00                lea    (%rax,%rax,1),%ebp       #ebp=2*rax=2*func4(rdi-1)

0000000000001921 <func4>:
   1921:   f3 0f 1e fa          endbr64
   1925:   85 ff                test   %edi,%edi
   1927:   7e 29                jle    1952 <func4+0x31>        #edi<=0,return 0
   1929:   55                   push   %rbp                     #被调用者保存
   192a:   53                   push   %rbx                     #被调用者保存
   192b:   48 83 ec 08          sub    $0x8,%rsp
   192f:   89 fb                mov    %edi,%ebx                #ebx=edi
   1931:   83 ff 01             cmp    $0x1,%edi                #edi:1
   1934:   74 22                je     1958 <func4+0x37>        #edi==1,return 1
   1936:   8d 7f ff             lea    -0x1(%rdi),%edi          #edi=rdi-1
   1939:   e8 e3 ff ff ff       call   1921 <func4>             #rax
   193e:   8d 2c 00             lea    (%rax,%rax,1),%ebp       #ebp=2*rax=2*func4(rdi-1)
   1941:   8d 7b fe             lea    -0x2(%rbx),%edi          #edi=rbx-2
   1944:   e8 d8 ff ff ff       call   1921 <func4>             #rax
   1949:   01 e8                add    %ebp,%eax                #eax+=ebp,eax=2*func4(rdi-1)+func4(rdi-2)
   194b:   48 83 c4 08          add    $0x8,%rsp
   194f:   5b                   pop    %rbx
   1950:   5d                   pop    %rbp
   1951:   c3                   ret
   1952:   b8 00 00 00 00       mov    $0x0,%eax
   1957:   c3                   ret
   1958:   89 f8                mov    %edi,%eax
   195a:   eb ef                jmp    194b <func4+0x2a>
```

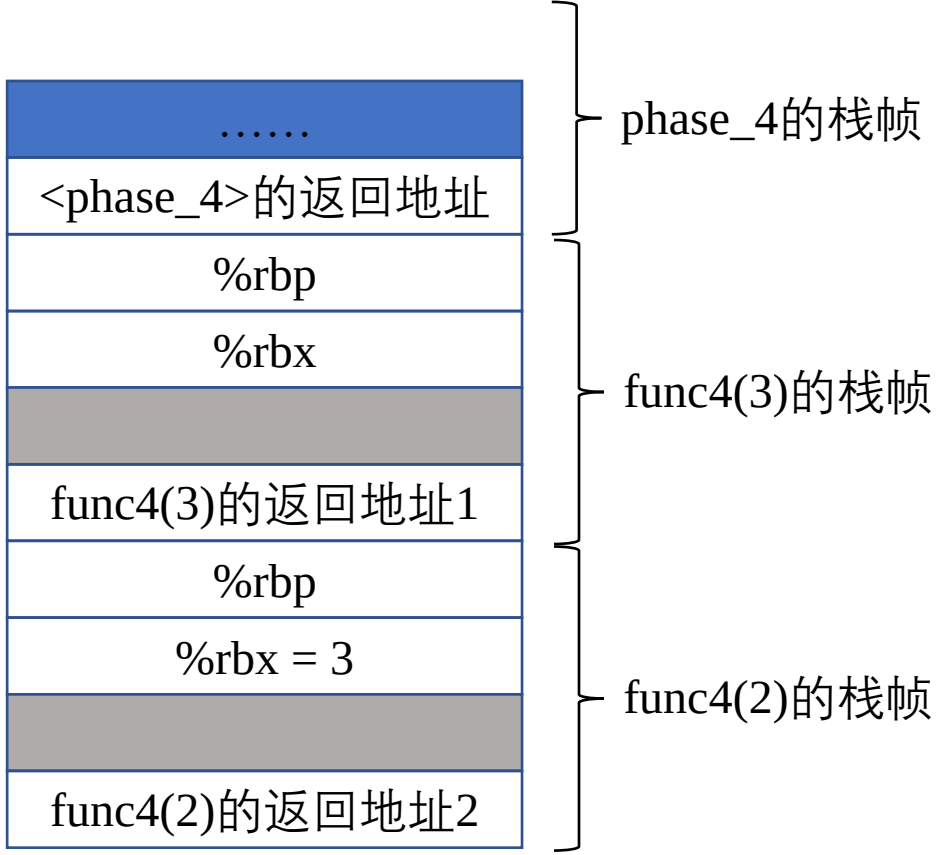


递归调用的运行时栈

以phase_4调用func4(3)为例

并不是每一个x86-64过程都会在栈上分配空间

```
1939:   e8 e3 ff ff ff          call   1921 <func4>              #rax
193e:   8d 2c 00                lea    (%rax,%rax,1),%ebp       #ebp=2*rax=2*func4(rdi-1)

0000000000001921 <func4>:
   1921:   f3 0f 1e fa          endbr64
   1925:   85 ff                test   %edi,%edi
   1927:   7e 29                jle    1952 <func4+0x31>        #edi<=0,return 0
   1929:   55                   push   %rbp                     #被调用者保存
   192a:   53                   push   %rbx                     #被调用者保存
   192b:   48 83 ec 08          sub    $0x8,%rsp
   192f:   89 fb                mov    %edi,%ebx                #ebx=edi
   1931:   83 ff 01             cmp    $0x1,%edi                #edi:1
   1934:   74 22                je     1958 <func4+0x37>        #edi==1,return 1
   1936:   8d 7f ff             lea    -0x1(%rdi),%edi          #edi=rdi-1
   1939:   e8 e3 ff ff ff       call   1921 <func4>             #rax
   193e:   8d 2c 00             lea    (%rax,%rax,1),%ebp       #ebp=2*rax=2*func4(rdi-1)
   1941:   8d 7b fe             lea    -0x2(%rbx),%edi          #edi=rbx-2
   1944:   e8 d8 ff ff ff       call   1921 <func4>             #rax
   1949:   01 e8                add    %ebp,%eax                #eax+=ebp,eax=2*func4(rdi-1)+func4(rdi-2)
   194b:   48 83 c4 08          add    $0x8,%rsp
   194f:   5b                   pop    %rbx
   1950:   5d                   pop    %rbp
   1951:   c3                   ret
   1952:   b8 00 00 00 00       mov    $0x0,%eax
   1957:   c3                   ret
   1958:   89 f8                mov    %edi,%eax
   195a:   eb ef                jmp    194b <func4+0x2a>
```



phase_5

Array

解题思路

- 根据汇编代码易知输入的字符串长度为6
- 可以观察到汇编代码访问了储存在内存中的数组（**全局变量**）
- 并且通过循环对该数组的某些元素求和，其中每个元素的数组下标与我们输入的字符串有关
- 查看该数组，可发现数组的前6个元素和第3～8个元素的和均为0x2f，满足要求
- 我们可以根据以上条件构造出符合要求的字符串
- 知道字符‘***x***’**的ascii码为0x3*x*（x=0,1,2,3,4,5,6,7,8,9）**对解题有帮助

```
00000000000019e0 <phase_5>:
    19e0:   f3 0f 1e fa             endbr64
    19e4:   53                      push   %rbx
    19e5:   48 89 fb                mov    %rdi,%rbx
    19e8:   e8 db 02 00 00          call   1cc8 <string_length>
    19ed:   83 f8 06                cmp    $0x6,%eax
    19f0:   75 28                   jne    1a1a <phase_5+0x3a>
    19f2:   b9 00 00 00 00          mov    $0x0,%ecx
    19f7:   b8 00 00 00 00          mov    $0x0,%eax
    19fc:   83 f8 05                cmp    $0x5,%eax
    19ff:   7f 20                   jg     1a21 <phase_5+0x41>
    1a01:   48 63 d0                movslq %eax,%rdx
    1a04:   0f b6 14 13             movzbl (%rbx,%rdx,1),%edx
    1a08:   83 e2 0f                and    $0xf,%edx
    1a0b:   48 8d 35 2e 2a 00 00    lea    0x2a2e(%rip),%rsi        # 4440 <array.3497>
    1a12:   03 0c 96                add    (%rsi,%rdx,4),%ecx
    1a15:   83 c0 01                add    $0x1,%eax
    1a18:   eb e2                   jmp    19fc <phase_5+0x1c>
    1a1a:   e8 24 06 00 00          call   2043 <explode_bomb>
    1a1f:   eb d1                   jmp    19f2 <phase_5+0x12>
    1a21:   83 f9 2f                cmp    $0x2f,%ecx
    1a24:   75 02                   jne    1a28 <phase_5+0x48>
    1a26:   5b                      pop    %rbx
    1a27:   c3                      ret
    1a28:   e8 16 06 00 00          call   2043 <explode_bomb>
    1a2d:   eb f7                   jmp    1a26 <phase_5+0x46>
```

```
00000000000019e0 <phase_5>:
    19e0:   f3 0f 1e fa             endbr64
    19e4:   53                      push   %rbx
    19e5:   48 89 fb                mov    %rdi,%rbx
    19e8:   e8 db 02 00 00          call   1cc8 <string_length>
    19ed:   83 f8 06                cmp    $0x6,%eax
    19f0:   75 28                   jne    1a1a <phase_5+0x3a>
    19f2:   b9 00 00 00 00          mov    $0x0,%ecx
    19f7:   b8 00 00 00 00          mov    $0x0,%eax
    19fc:   83 f8 05                cmp    $0x5,%eax
    19ff:   7f 20                   jg     1a21 <phase_5+0x41>
    1a01:   48 63 d0                movslq %eax,%rdx
    1a04:   0f b6 14 13             movzbl (%rbx,%rdx,1),%edx
    1a08:   83 e2 0f                and    $0xf,%edx
    1a0b:   48 8d 35 2e 2a 00 00    lea    0x2a2e(%rip),%rsi        # 4440 <array.3497>
    1a12:   03 0c 96                add    (%rsi,%rdx,4),%ecx
    1a15:   83 c0 01                add    $0x1,%eax
    1a18:   eb e2                   jmp    19fc <phase_5+0x1c>
    1a1a:   e8 24 06 00 00          call   2043 <explode_bomb>
    1a1f:   eb d1                   jmp    19f2 <phase_5+0x12>
    1a21:   83 f9 2f                cmp    $0x2f,%ecx
    1a24:   75 02                   jne    1a28 <phase_5+0x48>
    1a26:   5b                      pop    %rbx
    1a27:   c3                      ret
    1a28:   e8 16 06 00 00          call   2043 <explode_bomb>
    1a2d:   eb f7                   jmp    1a26 <phase_5+0x46>
```

```
00000000000019e0 <phase_5>:
    19e0:   f3 0f 1e fa             endbr64
    19e4:   53                      push   %rbx
    19e5:   48 89 fb                mov    %rdi,%rbx              #rbx->"xxxxxx"
    19e8:   e8 db 02 00 00          call   1cc8 <string_length>
    19ed:   83 f8 06                cmp    $0x6,%eax              #eax:6 输入字符串长度为6
    19f0:   75 28                   jne    1a1a <phase_5+0x3a>
```

```
00000000000019e0 <phase_5>:
    19e0:   f3 0f 1e fa             endbr64
    19e4:   53                      push   %rbx
    19e5:   48 89 fb                mov    %rdi,%rbx
    19e8:   e8 db 02 00 00          call   1cc8 <string_length>
    19ed:   83 f8 06                cmp    $0x6,%eax
    19f0:   75 28                   jne    1a1a <phase_5+0x3a>
    19f2:   b9 00 00 00 00          mov    $0x0,%ecx
    19f7:   b8 00 00 00 00          mov    $0x0,%eax
    19fc:   83 f8 05                cmp    $0x5,%eax
    19ff:   7f 20                   jg     1a21 <phase_5+0x41>
    1a01:   48 63 d0                movslq %eax,%rdx
    1a04:   0f b6 14 13             movzbl (%rbx,%rdx,1),%edx
    1a08:   83 e2 0f                and    $0xf,%edx
    1a0b:   48 8d 35 2e 2a 00 00    lea    0x2a2e(%rip),%rsi        # 4440 <array.3497>
    1a12:   03 0c 96                add    (%rsi,%rdx,4),%ecx
    1a15:   83 c0 01                add    $0x1,%eax
    1a18:   eb e2                   jmp    19fc <phase_5+0x1c>
    1a1a:   e8 24 06 00 00          call   2043 <explode_bomb>
    1a1f:   eb d1                   jmp    19f2 <phase_5+0x12>
    1a21:   83 f9 2f                cmp    $0x2f,%ecx
    1a24:   75 02                   jne    1a28 <phase_5+0x48>
    1a26:   5b                      pop    %rbx
    1a27:   c3                      ret
    1a28:   e8 16 06 00 00          call   2043 <explode_bomb>
    1a2d:   eb f7                   jmp    1a26 <phase_5+0x46>
```

```
(gdb) x/72b $rsi
0x55a6f2ffe440 <array.3497>:    0x02    0x00    0x00    0x00    0x0a    0x00    0x00    0x00
0x55a6f2ffe448 <array.3497+8>:  0x06    0x00    0x00    0x00    0x01    0x00    0x00    0x00
0x55a6f2ffe450 <array.3497+16>: 0x0c    0x00    0x00    0x00    0x10    0x00    0x00    0x00
0x55a6f2ffe458 <array.3497+24>: 0x09    0x00    0x00    0x00    0x03    0x00    0x00    0x00
0x55a6f2ffe460 <array.3497+32>: 0x04    0x00    0x00    0x00    0x07    0x00    0x00    0x00
0x55a6f2ffe468 <array.3497+40>: 0x0e    0x00    0x00    0x00    0x05    0x00    0x00    0x00
0x55a6f2ffe470 <array.3497+48>: 0x0b    0x00    0x00    0x00    0x08    0x00    0x00    0x00
0x55a6f2ffe478 <array.3497+56>: 0x0f    0x00    0x00    0x00    0x0d    0x00    0x00    0x00
0x55a6f2ffe480: 0x53    0x6f    0x20    0x79    0x6f    0x75    0x20    0x74
```

x86-64 小端法存储整型数据

本步骤为一个**全局数组变量**创建了地址，为接下来的数组引用做准备

```
19f2:   b9 00 00 00 00          mov    $0x0,%ecx                        #ecx=0
19f7:   b8 00 00 00 00          mov    $0x0,%eax                        #eax=0
19fc:   83 f8 05                cmp    $0x5,%eax                        #eax:0x5
19ff:   7f 20                   jg     1a21 <phase_5+0x41>
1a01:   48 63 d0                movslq %eax,%rdx                        #rdx=eax(0)          rdx=0          rdx=1          ...    rdx=5
1a04:   0f b6 14 13             movzbl (%rbx,%rdx,1),%edx               #edx=M[rbx+rdx]      edx=M[rbx]     edx=M[rbx+1]   ...    edx=M[rbx+5]
1a08:   83 e2 0f                and    $0xf,%edx                        #edx&=0xf  保留低4位
1a0b:   48 8d 35 2e 2a 00 00    lea    0x2a2e(%rip),%rsi        # 4440 <array.3497>  #rsi=rip+0x2a2e->2
1a12:   03 0c 96                add    (%rsi,%rdx,4),%ecx               #ecx+=M[rsi+4*rdx]
1a15:   83 c0 01                add    $0x1,%eax                        #eax+=1              eax=1          eax=2          ...    eax=6
1a18:   eb e2                   jmp    19fc <phase_5+0x1c>
1a1a:   e8 24 06 00 00          call   2043 <explode_bomb>
1a1f:   eb d1                   jmp    19f2 <phase_5+0x12>
1a21:   83 f9 2f                cmp    $0x2f,%ecx                       #ecx:0x2f
1a24:   75 02                   jne    1a28 <phase_5+0x48>
```

phase_6

Single Linked List

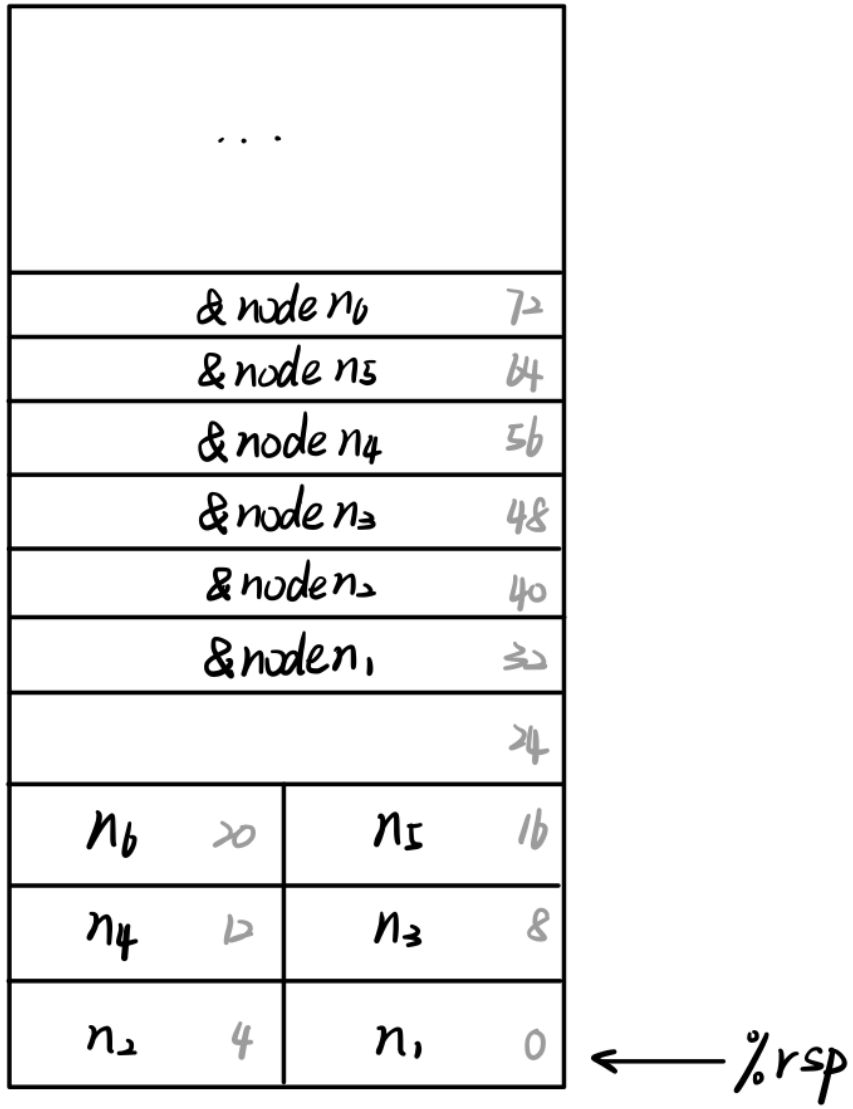
```
//读取6个整数到临时数组变量
1a4b:   48 89 e6                mov    %rsp,%rsi
1a4e:   e8 76 06 00 00          call   20c9 <read_six_numbers>
1a53:   bd 00 00 00 00          mov    $0x0,%ebp
1a58:   eb 27                   jmp    1a81 <phase_6+0x52>
```

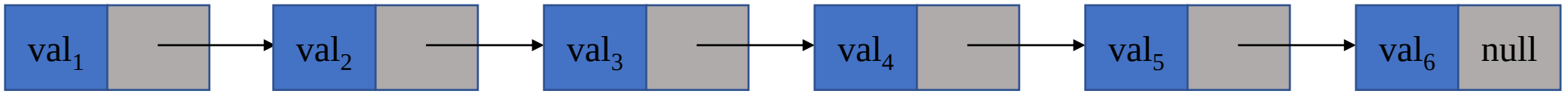
```
//两层循环，要求6个整数在1-6之间且互不相等
1a5a:   e8 e4 05 00 00          call   2043 <explode_bomb>
1a5f:   eb 33                   jmp    1a94 <phase_6+0x65>
1a61:   83 c3 01                add    $0x1,%ebx
1a64:   83 fb 05                cmp    $0x5,%ebx                #rbx:5
1a67:   7f 15                   jg     1a7e <phase_6+0x4f>
1a69:   48 63 c5                movslq %ebp,%rax                #rax=ebp
1a6c:   48 63 d3                movslq %ebx,%rdx                #rdx=ebx
1a6f:   8b 3c 94                mov    (%rsp,%rdx,4),%edi       #rdi=M[rsp+4*rdx] 数组引用
1a72:   39 3c 84                cmp    %edi,(%rsp,%rax,4)       #M[rsp+4*rax]:rdi  第i个数和第i+j个数比较
1a75:   75 ea                   jne    1a61 <phase_6+0x32>      #互不相等
1a77:   e8 c7 05 00 00          call   2043 <explode_bomb>
1a7c:   eb e3                   jmp    1a61 <phase_6+0x32>
1a7e:   44 89 e5                mov    %r12d,%ebp
1a81:   83 fd 05                cmp    $0x5,%ebp                #rbp:5
1a84:   7f 17                   jg     1a9d <phase_6+0x6e>
1a86:   48 63 c5                movslq %ebp,%rax                #rax=rbp
1a89:   8b 04 84                mov    (%rsp,%rax,4),%eax       #eax=M[rsp+4*rax] 数组引用
1a8c:   83 e8 01                sub    $0x1,%eax                #eax-=1
1a8f:   83 f8 05                cmp    $0x5,%eax                #eax:5
1a92:   77 c6                   ja     1a5a <phase_6+0x2b>      #eax<=6
1a94:   44 8d 65 01             lea    0x1(%rbp),%r12d          #r12d=rbp+1
1a98:   44 89 e3                mov    %r12d,%ebx               #rbx=r12d
1a9b:   eb c7                   jmp    1a64 <phase_6+0x35>
```

```
for(int i = 0; i < 6; i++){
    if(x[i] < 1 || x[i] > 6)
        explode_bomb();
    for(int j = i + 1; j < 6; j++){
        if(x[i] == x[j])
            explode_bomb();
    }
}
```

```
//按照临时数组中数字的大小存储结点指针数组
1a9d:   be 00 00 00 00          mov    $0x0,%esi                        #esi=0
1aa2:   eb 08                   jmp    1aac <phase_6+0x7d>
1aa4:   48 89 54 cc 20          mov    %rdx,0x20(%rsp,%rcx,8)           #M[rsp+8*rcx+0x20]=rdx
1aa9:   83 c6 01                add    $0x1,%esi
1aac:   83 fe 05                cmp    $0x5,%esi                        #esi:5
1aaf:   7f 1d                   jg     1ace <phase_6+0x9f>
1ab1:   b8 01 00 00 00          mov    $0x1,%eax                        #eax=1
1ab6:   48 8d 15 53 66 00 00    lea    0x6653(%rip),%rdx        # 8110 <node1>  #rdx=rip+0x6653
1abd:   48 63 ce                movslq %esi,%rcx                        #rcx=esi
1ac0:   39 04 8c                cmp    %eax,(%rsp,%rcx,4)               #M[rsp+4*rcx]:eax
1ac3:   7e df                   jle    1aa4 <phase_6+0x75>
1ac5:   48 8b 52 08             mov    0x8(%rdx),%rdx                   #rdx=M[rdx+8]
1ac9:   83 c0 01                add    $0x1,%eax                        #eax+=1
1acc:   eb ef                   jmp    1abd <phase_6+0x8e>
```

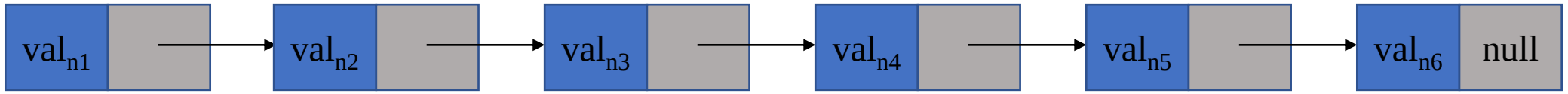
```
0x561c303e1110 <node1>: 0x00000001000000e2      0x0000561c303e1120
(gdb) x/12g $rdx
0x561c303e1110 <node1>: 0x00000001000000e2      0x0000561c303e1120
0x561c303e1120 <node2>: 0x000000020000012f      0x0000561c303e1130
0x561c303e1130 <node3>: 0x00000003000003ae      0x0000561c303e1140
0x561c303e1140 <node4>: 0x000000040000022c      0x0000561c303e1150
0x561c303e1150 <node5>: 0x000000050000015b      0x0000561c303e0080
0x561c303e1160 <host_table>:    0x0000561c303dd81c      0x0000561c303dd825
(gdb) x/12g 0x0000561c303e0080
0x561c303e0080 <node6>: 0x0000000600000131      0x0000000000000000
```





```
//重排链表
1ace:   48 8b 5c 24 20          mov    0x20(%rsp),%rbx          #rbx=&node n_i
1ad3:   48 89 d9                mov    %rbx,%rcx                #rcx=rbx
1ad6:   b8 01 00 00 00          mov    $0x1,%eax                #eax=1
1adb:   eb 12                   jmp    1aef <phase_6+0xc0>
1add:   48 63 d0                movslq %eax,%rdx                #rdx=eax
1ae0:   48 8b 54 d4 20          mov    0x20(%rsp,%rdx,8),%rdx   #rdx=M[rsp+8*rdx+20]=&node n_j
1ae5:   48 89 51 08             mov    %rdx,0x8(%rcx)           #M[rcx+8]=rdx;
1ae9:   83 c0 01                add    $0x1,%eax                #eax+=1
1aec:   48 89 d1                mov    %rdx,%rcx                #rcx=rdx
1aef:   83 f8 05                cmp    $0x5,%eax                #eax:5
1af2:   7e e9                   jle    1add <phase_6+0xae>
1af4:   48 c7 41 08 00 00 00    movq   $0x0,0x8(%rcx)           #尾结点空指针
1afb:   00
```

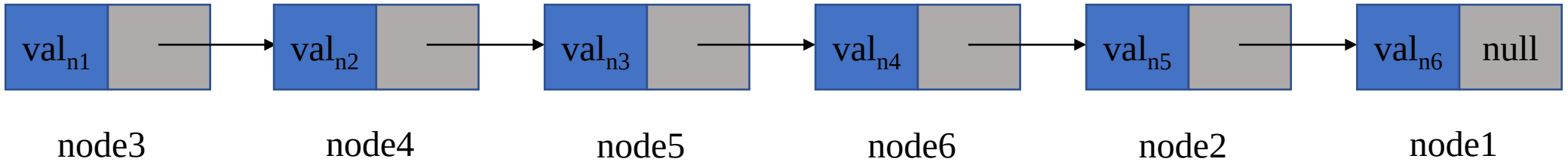
```
typedef struct node{
  long val;
  struct node *next;
}
```



```
//重排后链表低位4字节值单调递减
1afc:   bd 00 00 00 00          mov    $0x0,%ebp                #rbp=0
1b01:   eb 07                   jmp    1b0a <phase_6+0xdb>
1b03:   48 8b 5b 08             mov    0x8(%rbx),%rbx           #rbx=&node n_(i+1)
1b07:   83 c5 01                add    $0x1,%ebp
1b0a:   83 fd 04                cmp    $0x4,%ebp                #rbp:4
1b0d:   7f 11                   jg     1b20 <phase_6+0xf1>
1b0f:   48 8b 43 08             mov    0x8(%rbx),%rax           #rax=M[rbx+8]=&node n2
1b13:   8b 00                   mov    (%rax),%eax              #eax=node n2.val  高位4字节清零
1b15:   39 03                   cmp    %eax,(%rbx)              #node n_i.val:node n_(i+1).val
1b17:   7d ea                   jge    1b03 <phase_6+0xd4>      #node n_i.val:node n_(i+1).val
1b19:   e8 25 05 00 00          call   2043 <explode_bomb>
1b1e:   eb e3                   jmp    1b03 <phase_6+0xd4>
```

```
0x561c303e1110 <node1>: 0x00000001000000e2      0x0000561c303e1120
(gdb) x/12g $rdx
0x561c303e1110 <node1>: 0x00000001000000e2      0x0000561c303e1120
0x561c303e1120 <node2>: 0x000000020000012f      0x0000561c303e1130
0x561c303e1130 <node3>: 0x00000003000003ae      0x0000561c303e1140
0x561c303e1140 <node4>: 0x000000040000022c      0x0000561c303e1150
0x561c303e1150 <node5>: 0x000000050000015b      0x0000561c303e0080
0x561c303e1160 <host_table>:    0x0000561c303dd81c      0x0000561c303dd825
(gdb) x/12g 0x0000561c303e0080
0x561c303e0080 <node6>: 0x0000000600000131      0x0000000000000000
```

答案：3 4 5 6 2 1



secret_phase

如何进入?

```
Welcome to Dr. Evil's little bomb. You have 6 phases with
which to blow yourself up. Have a nice day! Mua ha ha ha!
Playing games without playing Genshin Impact is like reading Four Classics without reading Dream of Red Mansions.
Phase 1 defused. How about the next one?
0 1 1 3 5 11
That's number 2. Keep going!
1 1424
Halfway there!
4 8
So you got that one.  Try this one.
543210
Good work!  On to the next...
3 4 5 6 2 1
Huh……? No explosion occurred? (release covered ears)
Your instructor has been notified and will verify your solution.
But, I can't give you a clear answer yet. You need to measure it yourself.
[Inferior 1 (process 156242) exited normally]
(gdb)
```

```
/* Round and 'round in memory we go, where we stop, the bomb blows! */
input = read_line();
phase_5(input);
phase_defused(fp);
printf("Good work!  On to the next...\n");

/* This phase will never be used, since no one will get past the
 * earlier ones.  But just in case, make this one extra hard. */
input = read_line();
phase_6(input);
phase_defused(fp);

/* Wow, they got it!  But isn't something... missing?  Perhaps
 * something they overlooked?  Mua ha ha ha ha! */
```

注意到main函数以外的字符串输出

如何进入?

```
0000000000002259 <phase_defused>:
    2259:   f3 0f 1e fa             endbr64
    225d:   53                      push   %rbx
    225e:   48 89 fb                mov    %rdi,%rbx
    2261:   c7 07 00 00 00 00       movl   $0x0,(%rdi)
    2267:   48 89 fe                mov    %rdi,%rsi
    226a:   bf 01 00 00 00          mov    $0x1,%edi
    226f:   e8 65 fc ff ff          call   1ed9 <send_msg>
    2274:   83 3b 01                cmpl   $0x1,(%rbx)
    2277:   75 0b                   jne    2284 <phase_defused+0x2b>
    2279:   83 3d cc 62 00 00 06    cmpl   $0x6,0x62cc(%rip)        # 854c <num_input_strings>
    2280:   74 22                   je     22a4 <phase_defused+0x4b>
    2282:   5b                      pop    %rbx
    2283:   c3                      ret
    2284:   48 8d 35 c5 22 00 00    lea    0x22c5(%rip),%rsi        # 4550 <array.3497+0x110>
    228b:   bf 01 00 00 00          mov    $0x1,%edi
    2290:   b8 00 00 00 00          mov    $0x0,%eax
    2295:   e8 c6 f0 ff ff          call   1360 <__printf_chk@plt>
    229a:   bf 08 00 00 00          mov    $0x8,%edi
    229f:   e8 ec f0 ff ff          call   1390 <exit@plt>
    22a4:   e8 cf f3 ff ff          call   1678 <genshin>
```

当前phase为phase_6时会调用函数genshin

如何进入?

进入函数genshin

```
0000000000001678 <genshin>:
    1678:   f3 0f 1e fa             endbr64
    167c:   48 81 ec 98 00 00 00    sub    $0x98,%rsp
    1683:   64 48 8b 04 25 28 00    mov    %fs:0x28,%rax
    168a:   00 00
    168c:   48 89 84 24 88 00 00    mov    %rax,0x88(%rsp)
    1693:   00
    1694:   31 c0                   xor    %eax,%eax
    1696:   48 8d 4c 24 0c          lea    0xc(%rsp),%rcx                                #第四个参数
    169b:   48 8d 54 24 08          lea    0x8(%rsp),%rdx                                #第三个参数
    16a0:   4c 8d 44 24 10          lea    0x10(%rsp),%r8                                #第五个参数
    16a5:   48 8d 35 f0 2a 00 00    lea    0x2af0(%rip),%rsi        # 419c <_IO_stdin_used+0x19c>   #第二个参数
    16ac:   48 8d 3d 15 70 00 00    lea    0x7015(%rip),%rdi        # 86c8 <input_strings+0x168>    #第一个参数
    16b3:   e8 98 fc ff ff          call   1350 <__isoc99_sscanf@plt>
    16b8:   83 f8 03                cmp    $0x3,%eax                                     #eax:3
    16bb:   74 20                   je     16dd <genshin+0x65>
(gdb) print (char*)($rsi)
$2 = 0x55c07e43319c "%d %d %s"
(gdb) print (char*)($rdi)
$3 = 0x55c07e4376c8 <input_strings+360> "4 8"
```

phase_4还需要多输入一个字符串

如何进入?

```
16dd:   48 8d 7c 24 10          lea    0x10(%rsp),%rdi                                  #第一个参数
16e2:   48 8d 35 bf 2a 00 00    lea    0x2abf(%rip),%rsi        # 41a8 <_IO_stdin_used+0x1a8>   #第二个参数
16e9:   e8 f2 05 00 00          call   1ce0 <strings_not_equal>
16ee:   85 c0                   test   %eax,%eax
16f0:   74 07                   je     16f9 <genshin+0x81>
16f2:   b8 00 00 00 00          mov    $0x0,%eax
16f7:   eb c9                   jmp    16c2 <genshin+0x4a>
16f9:   b8 01 00 00 00          mov    $0x1,%eax                                        #phase_4输入的字符串和内存中的字符串相同则返回1，否则返回0
16fe:   eb c2                   jmp    16c2 <genshin+0x4a>
1700:   e8 8b fb ff ff          call   1290 <__stack_chk_fail@plt>
```

```
(gdb) print (char*)($rsi)
$4 = 0x5579dcd471a8 "DoUKnowThatTheHarderThing2DoAnd..."
```

phase_4多输入的字符串是 DoUKnowThatTheHarderThing2DoAnd...

如何进入?

```
Welcome to Dr. Evil's little bomb. You have 6 phases with
which to blow yourself up. Have a nice day! Mua ha ha ha!
Playing games without playing Genshin Impact is like reading Four Classics without reading Dream of Red Mansions.
Phase 1 defused. How about the next one?
0 1 1 3 5 11
That's number 2. Keep going!
1 1424
Halfway there!
4 8 DoUKnowThatTheHarderThing2DoAnd...
So you got that one.  Try this one.
543210
Good work!  On to the next...
3 4 5 6 2 1
Do you truly believe you've opened the gates to Tivat? Stop daydreaming!
Huh……? No explosion occurred? (release covered ears)
Your instructor has been notified and will verify your solution.
But, I can't give you a clear answer yet. You need to measure it yourself.
[Inferior 1 (process 158559) exited normally]
```

多了一行输出，仍然无法进入secret_phase

如何进入?

回到函数phase_dufused

```
22a9:   85 c0                   test   %eax,%eax                #函数genshin返回1（即phase_4输入了正确字符串）则调用函数qidong
22ab:   75 26                   jne    22d3 <phase_defused+0x7a>
22ad:   48 8d 3d c4 23 00 00    lea    0x23c4(%rip),%rdi        # 4678 <array.3497+0x238>
22b4:   e8 b7 ef ff ff          call   1270 <puts@plt>
22b9:   48 8d 3d f8 23 00 00    lea    0x23f8(%rip),%rdi        # 46b8 <array.3497+0x278>
22c0:   e8 ab ef ff ff          call   1270 <puts@plt>
22c5:   48 8d 3d 34 24 00 00    lea    0x2434(%rip),%rdi        # 4700 <array.3497+0x2c0>
22cc:   e8 9f ef ff ff          call   1270 <puts@plt>
22d1:   eb af                   jmp    2282 <phase_defused+0x29>
22d3:   e8 2d f4 ff ff          call   1705 <qidong>
```

如何进入?

进入函数qidong

```
//找到特定字符串的最后一个字符
1720:   48 8d 05 b1 6e 00 00    lea    0x6eb1(%rip),%rax        # 85d8 <input_strings+0x78>
1727:   80 38 00                cmpb   $0x0,(%rax)
172a:   74 06                   je     1732 <qidong+0x2d>
172c:   48 83 c0 01             add    $0x1,%rax
1730:   eb f5                   jmp    1727 <qidong+0x22>
1732:   48 83 e8 01             sub    $0x1,%rax
```

```
(gdb) print (char*)($rax)
$5 = 0x55c7bba725d8 <input_strings+120> "0 1 1 3 5 11"
```

phase_2可能还需要输入一个字符串

如何进入?

```
//将该字符串的最后一部分倒置存储在临时字符串变量中
1736:   48 89 e2                mov    %rsp,%rdx                #rdx=rsp
1739:   eb 0a                   jmp    1745 <qidong+0x40>
173b:   88 0a                   mov    %cl,(%rdx)               #M[rdx]=%cl
173d:   48 83 c2 01             add    $0x1,%rdx
1741:   48 83 e8 01             sub    $0x1,%rax
1745:   0f b6 08                movzbl (%rax),%ecx              #ecx=M[rax]
1748:   80 f9 20                cmp    $0x20,%cl                #%cl:0x20 (' ')
174b:   74 0c                   je     1759 <qidong+0x54>
174d:   48 8d 35 84 6e 00 00    lea    0x6e84(%rip),%rsi        # 85d8 <input_strings+0x78>
1754:   48 39 f0                cmp    %rsi,%rax
1757:   75 e2                   jne    173b <qidong+0x36>
1759:   c6 02 00                movb   $0x0,(%rdx)              #添加'\0'
```

如何进入?

```
//将临时字符串变量与特定字符串比较，若相同则返回1，否则返回0
175c:   48 89 e7                mov    %rsp,%rdi
175f:   48 8d 35 6a 2a 00 00    lea    0x2a6a(%rip),%rsi        # 41d0 <_IO_stdin_used+0x1d0>
1766:   e8 75 05 00 00          call   1ce0 <strings_not_equal>
176b:   85 c0                   test   %eax,%eax
176d:   74 1d                   je     178c <qidong+0x87>
176f:   b8 00 00 00 00          mov    $0x0,%eax
1774:   48 8b 7c 24 78          mov    0x78(%rsp),%rdi
1779:   64 48 33 3c 25 28 00    xor    %fs:0x28,%rdi
1780:   00 00
1782:   75 0f                   jne    1793 <qidong+0x8e>
1784:   48 81 c4 88 00 00 00    add    $0x88,%rsp
178b:   c3                      ret
178c:   b8 01 00 00 00          mov    $0x1,%eax
1791:   eb e1                   jmp    1774 <qidong+0x6f>
1793:   e8 f8 fa ff ff          call   1290 <__stack_chk_fail@plt>
```

```
(gdb) print (char*)($rsi)
$6 = 0x55c7bba6e1d0 "?gnihTema3ehTyllausUerAoD2gnihTthgiRehT..."
```

phase_2需要多输入的字符串是...TheRightThing2DoAreUsuallyThe3ameThing?

如何进入?

```
//将临时字符串变量与特定字符串比较，若相同则返回1，否则返回0
175c:   48 89 e7                mov    %rsp,%rdi
175f:   48 8d 35 6a 2a 00 00    lea    0x2a6a(%rip),%rsi        # 41d0 <_IO_stdin_used+0x1d0>
1766:   e8 75 05 00 00          call   1ce0 <strings_not_equal>
176b:   85 c0                   test   %eax,%eax
176d:   74 1d                   je     178c <qidong+0x87>
176f:   b8 00 00 00 00          mov    $0x0,%eax
1774:   48 8b 7c 24 78          mov    0x78(%rsp),%rdi
1779:   64 48 33 3c 25 28 00    xor    %fs:0x28,%rdi
1780:   00 00
1782:   75 0f                   jne    1793 <qidong+0x8e>
1784:   48 81 c4 88 00 00 00    add    $0x88,%rsp
178b:   c3                      ret
178c:   b8 01 00 00 00          mov    $0x1,%eax
1791:   eb e1                   jmp    1774 <qidong+0x6f>
1793:   e8 f8 fa ff ff          call   1290 <__stack_chk_fail@plt>
```

```
(gdb) print (char*)($rsi)
$6 = 0x55c7bba6e1d0 "?gnihTema3ehTyllausUerAoD2gnihTthgiRehT..."
```

phase_2需要多输入的字符串是...TheRightThing2DoAreUsuallyThe3ameThing?

如何进入?

回到函数phase_dufused

```
22d8:   85 c0                   test   %eax,%eax                #qidong返回1则调用函数secret_phase
22da:   74 24                   je     2300 <phase_defused+0xa7>
22dc:   48 8d 3d dd 22 00 00    lea    0x22dd(%rip),%rdi        # 45c0 <array.3497+0x180>
22e3:   e8 88 ef ff ff          call   1270 <puts@plt>
22e8:   48 8d 3d f9 22 00 00    lea    0x22f9(%rip),%rdi        # 45e8 <array.3497+0x1a8>
22ef:   e8 7c ef ff ff          call   1270 <puts@plt>
22f4:   b8 00 00 00 00          mov    $0x0,%eax
22f9:   e8 90 f8 ff ff          call   1b8e <secret_phase>
22fe:   eb ad                   jmp    22ad <phase_defused+0x54>
2300:   48 8d 3d 21 23 00 00    lea    0x2321(%rip),%rdi        # 4628 <array.3497+0x1e8>
2307:   e8 64 ef ff ff          call   1270 <puts@plt>
230c:   eb 9f                   jmp    22ad <phase_defused+0x54>
```

如何进入?

```
Welcome to Dr. Evil's little bomb. You have 6 phases with
which to blow yourself up. Have a nice day! Mua ha ha ha!
Playing games without playing Genshin Impact is like reading Four Classics without reading Dream of Red Mansions.
Phase 1 defused. How about the next one?
0 1 1 3 5 11 ...TheRightThing2DoAreUsuallyThe3ameThing?
That's number 2. Keep going!
1 1424
Halfway there!
4 8 DoUKnowThatTheHarderThing2DoAnd...
So you got that one.  Try this one.
543210
Good work!  On to the next...
3 4 5 6 2 1
Curses, you've found the secret phase!
But finding phase and unlocking the door are quite different...
```

解题思路

```
1ba7:   e8 62 05 00 00          call   210e <read_line>
1bac:   48 89 c7                mov    %rax,%rdi                #第一个参数 &"63"
1baf:   ba 0a 00 00 00          mov    $0xa,%edx                #第三个参数
1bb4:   be 00 00 00 00          mov    $0x0,%esi                #第二个参数
1bb9:   e8 52 f7 ff ff          call   1310 <strtol@plt>        #将字符串转化为long
1bbe:   48 89 c3                mov    %rax,%rbx                #rbx=rax
1bc1:   48 8d 3d e8 64 00 00    lea    0x64e8(%rip),%rdi        #80b0 <n1>    #第一个参数
1bc8:   e8 71 ff ff ff          call   1b3e <get_sum>           #求和
1bcd:   39 d8                   cmp    %ebx,%eax                #ebx:eax
```

```
strtol(s, &end, b)
```

参数：该函数接受三个强制性参数，如下所述：

- **s**：指定具有整数表示形式的字符串。
- **end**：引用类型为char *的已分配对象。 end的值由函数设置为s中最后一个有效字符后的下一个字符。它也可以是空指针，在这种情况下不使用。
- **b**：指定整数值的底数。

答案：63

总结

- 局部变量与全局变量的存储
- 过程运行时栈结构与寄存器状态的动态变化
- 递归调用与栈
- 一些基本的数据结构在汇编代码层面的实现
- 写代码时函数与变量的命名

谢谢！