实验4

班级：
姓名：
学号：

实验内容：
1 二进制炸弹实验的内容、方法和基本步骤；
2 程序机器级表示、分析与调试基本知识和方法的应用。

实验目标：
1 加深对程序的机器级表示、汇编与反汇编、二进制程序分析与调试、逆向工程等方面知识的理解和掌握；
2 从程序员角度认识计算机系统，分析高级语言对应的机器行为及其对程序执行结果和性能的影响，解决计算机系统设计、程序开发过程中的关键问题。
3 掌握各种开源的编译调试工具。

实验任务：
1 学习 MOOC 内容
https://www.icourse163.org/learn/NJU-1449521162
第五周 二进制程序逆向工程
第 1 讲 二进制炸弹实验：概述
第 2 讲 二进制炸弹实验：字符串比较
第 3 讲 二进制炸弹实验：浮点数表示
第 4 讲 二进制炸弹实验：课后实验

2 完成实验
详见二进制逆向工程实验文档
2.1 字符串 phase0
2.1.1 程序代码和注释说明
将bomb文件反汇编，得到bomb.s文件：

```
1.  objdump -d bomb > bomb.s
```

查看phase_0函数的汇编代码：

```
08049454 <phase_0>:
 8049454:       55                      push   %ebp
 8049455:       89 e5                   mov    %esp,%ebp
 8049457:       83 ec 08                sub    $0x8,%esp
 804945a:       83 ec 08                sub    $0x8,%esp
 804945d:       68 e0 a1 04 08          push   $0x804a1e0
 8049462:       ff 75 08                pushl  0x8(%ebp)
 8049465:       e8 f7 07 00 00          call   8049c61 <strings_not_equal>
 804946a:       83 c4 10                add    $0x10,%esp
 804946d:       85 c0                   test   %eax,%eax
 804946f:       74 0c                   je     804947d <phase_0+0x29>
 8049471:       e8 53 0a 00 00          call   8049ec9 <explode_bomb>
 8049476:       b8 00 00 00 00          mov    $0x0,%eax
 804947b:       eb 05                   jmp    8049482 <phase_0+0x2e>
 804947d:       b8 01 00 00 00          mov    $0x1,%eax
 8049482:       c9                      leave
 8049483:       c3                      ret
```

分析可知，前四句，为局部变量分配栈空间，接下来两句为call调用的函数准备参数，test用来测试strings_not_equal的返回值，检查是否为0。如果相等（返回值为0），则跳转到标签804947d，否则继续执行下一条指令。如果不相等，调用函数explode_bomb，表示输入不符合预期，程序爆炸。
综合起来，这个函数的目的是比较两个字符串是否相等。如果相等，返回1；否则，调用explode_bomb函数，表示输入不符合预期，程序爆炸。
所以拆除炸弹的关键就在于在0x804a1e0处存放的字符串。

2.2.2 结果分析与讨论

打开gdb调试程序，断点设置在phase_0里面调用strings_not_equal函数的call指令的位置，即0x8049465。

```
(gdb) b phase_0
Breakpoint 1 at 0x804945a
(gdb) r
Starting program: /home/sanfenbai/Desktop/计算机系统/课程设计/拆炸弹/bomb
Welcome to my fiendish little bomb. You have 7 phases with
which to blow yourself up. Have a nice day!
asdasd

Breakpoint 1, 0x0804945a in phase_0 ()
(gdb) ni
0x0804945d in phase_0 ()
(gdb)
0x08049462 in phase_0 ()
(gdb)
0x08049465 in phase_0 ()
(gdb) x/1s 0x804a1e0
0x804a1e0:      "A text line is a sequence of ASCII characters."
```

查看位于0x804a1e0处的字符串，结果为“A text line is a sequence of ASCII characters.”，运行bomb，输入字符串，拆除成功。

```
sanfenbai@ubuntu:~/Desktop/计算机系统/课程设计/拆炸弹$ ./bomb
Welcome to my fiendish little bomb. You have 7 phases with
which to blow yourself up. Have a nice day!
A text line is a sequence of ASCII characters.
Well done! You seem to have warmed up!
```

2.2 浮点数表示 phase1

2.2.1 程序代码和注释说明

找到phase_1的汇编代码：

```
08049484 <phase_1>:
 8049484:       55                      push   %ebp
 8049485:       89 e5                   mov    %esp,%ebp
 8049487:       83 ec 28                sub    $0x28,%esp
 804948a:       c7 45 f4 a1 84 76 09    movl   $0x97684a1,-0xc(%ebp)
 8049491:       db 45 f4                fildl  -0xc(%ebp)
 8049494:       dd 5d e8                fstpl  -0x18(%ebp)
 8049497:       8d 45 e0                lea    -0x20(%ebp),%eax
 804949a:       50                      push   %eax
 804949b:       8d 45 e4                lea    -0x1c(%ebp),%eax
 804949e:       50                      push   %eax
 804949f:       68 0f a2 04 08          push   $0x804a20f
 80494a4:       ff 75 08                pushl  0x8(%ebp)
 80494a7:       e8 24 fc ff ff          call   80490d0 <__isoc99_sscanf@plt>
 80494ac:       83 c4 10                add    $0x10,%esp
 80494af:       83 f8 02                cmp    $0x2,%eax
 80494b2:       74 0c                   je     80494c0 <phase_1+0x3c>
 80494b4:       e8 10 0a 00 00          call   8049ec9 <explode_bomb>
 80494b9:       b8 00 00 00 00          mov    $0x0,%eax
 80494be:       eb 2c                   jmp    80494ec <phase_1+0x68>
 80494c0:       8d 45 e8                lea    -0x18(%ebp),%eax
 80494c3:       8b 10                   mov    (%eax),%edx
 80494c5:       8b 45 e4                mov    -0x1c(%ebp),%eax
 80494c8:       39 c2                   cmp    %eax,%edx
 80494ca:       75 0f                   jne    80494db <phase_1+0x57>
 80494cc:       8d 45 e8                lea    -0x18(%ebp),%eax
 80494cf:       83 c0 04                add    $0x4,%eax
 80494d2:       8b 10                   mov    (%eax),%edx
 80494d4:       8b 45 e0                mov    -0x20(%ebp),%eax
 80494d7:       39 c2                   cmp    %eax,%edx
 80494d9:       74 0c                   je     80494e7 <phase_1+0x63>
 80494db:       e8 e9 09 00 00          call   8049ec9 <explode_bomb>
 80494e0:       b8 00 00 00 00          mov    $0x0,%eax
 80494e5:       eb 05                   jmp    80494ec <phase_1+0x68>
 80494e7:       b8 01 00 00 00          mov    $0x1,%eax
 80494ec:       c9                      leave
 80494ed:       c3                      ret
```

前几句用一个整形常量0x97684a1对-0xc(%ebp)中的浮点数初始化，然后将整型值0x97684a1通过浮点栈转为双精度浮点表示，并传送到本阶段函数栈帧中地址为-0x18(%ebp)处开始连续存放。然后用几个push指令为call指令调用的sscanf函数准备参数，此函数按照格式字符串参数的指示，从其第一个参数所对应的输入字符串中读出数据。使用gdb查看sscanf读入的格式字符串：

```
(gdb) x/1s 0x804a20f
0x804a20f:      "%d %d"
```

发现是读入两个整型，下面2条指令也检查是否输入两个整数，若是，则跳转后续检查继续执行，否则引爆炸弹。

```
 80494ac:       83 c4 10                add    $0x10,%esp
 80494af:       83 f8 02                cmp    $0x2,%eax
 80494b2:       74 0c                   je     80494c0 <phase_1+0x3c>
 80494b4:       e8 10 0a 00 00          call   8049ec9 <explode_bomb>
```

```
80494c0:       8d 45 e8                lea    -0x18(%ebp),%eax
80494c3:       8b 10                   mov    (%eax),%edx
80494c5:       8b 45 e4                mov    -0x1c(%ebp),%eax
80494c8:       39 c2                   cmp    %eax,%edx
80494ca:       75 0f                   jne    80494db <phase_1+0x57>
80494cc:       8d 45 e8                lea    -0x18(%ebp),%eax
80494cf:       83 c0 04                add    $0x4,%eax
80494d2:       8b 10                   mov    (%eax),%edx
80494d4:       8b 45 e0                mov    -0x20(%ebp),%eax
80494d7:       39 c2                   cmp    %eax,%edx
80494d9:       74 0c                   je     80494e7 <phase_1+0x63>
80494db:       e8 e9 09 00 00          call   8049ec9 <explode_bomb>
```

前五条指令用于比较存放于-0x18(%ebp)处的浮点数的低四个字节和第一个输入整数的值是否相等，若不相等则引爆炸弹。

后几条指令则比较存放于-0x18(%ebp)处的浮点数的高四个字节和第二个输入整数的值是否相等，若不相等则引爆炸弹。

通过以上分析可知，输入的两个整数分别对应于存放的浮点数的高32位和低32位。

2.2.2 结果分析与讨论

整型值0x97684a1，其对应的双精度浮点数的IEEE 754表示为(十六进制字节序列，从高位到低位）:41a2ed0942000000。

因此输入的第一个整数应为：$42000000_{16}=1107296256_{10}$

第二个整数应为：$41a2ed094_{16}=1101196553_{10}$

运行bomb程序，输入两个整数，拆除成功。

```
sanfenbai@ubuntu:~/Desktop/计算机系统/课程设计/拆炸弹$ ./bomb
Welcome to my fiendish little bomb. You have 7 phases with
which to blow yourself up. Have a nice day!
A text line is a sequence of ASCII characters.
Well done! You seem to have warmed up!
1107296256 1101196553
Phase 1 defused. How about the next one?
```