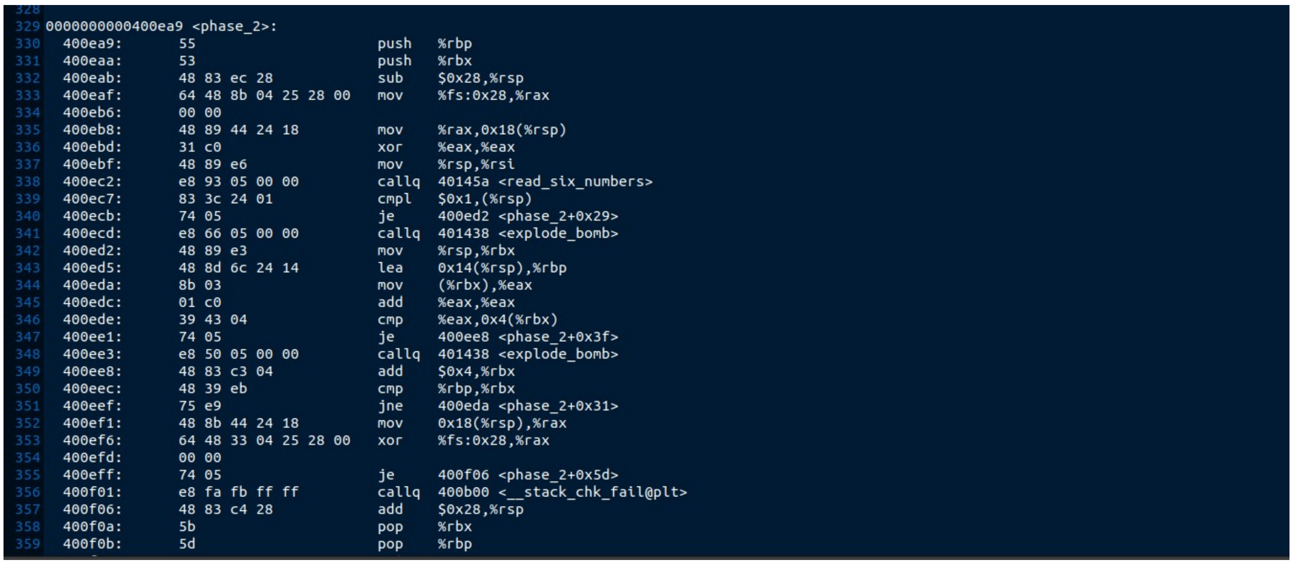
Phase_2

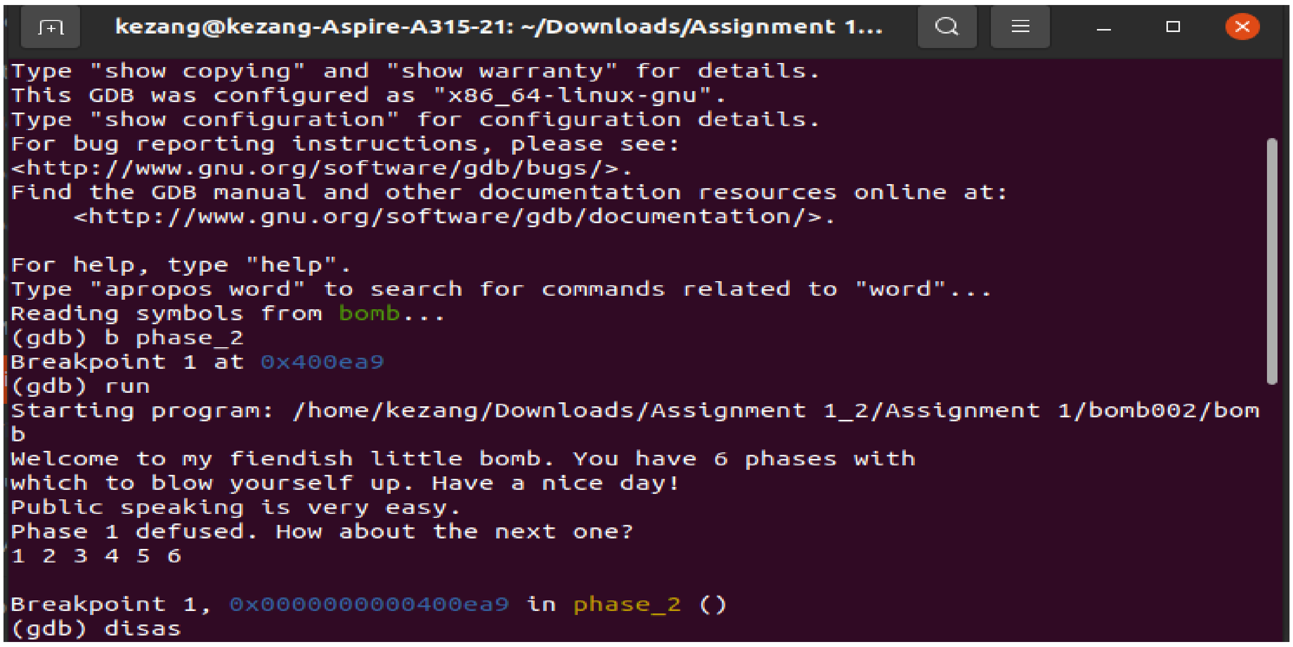
In second phase we have to enter six interger in order to proceed to next phase. How we know it should be six integer is because o this comment vi bomb.s where we can see the assembly code.



```
0000000000400ea9 <phase_2>:
  400ea9:	55                   	push   %rbp
  400eaa:	53                   	push   %rbx
  400eab:	48 83 ec 28          	sub    $0x28,%rsp
  400eaf:	64 48 8b 04 25 28 00 	mov    %fs:0x28,%rax
  400eb6:	00 00 
  400eb8:	48 89 44 24 18       	mov    %rax,0x18(%rsp)
  400ebd:	31 c0                	xor    %eax,%eax
  400ebf:	48 89 e6             	mov    %rsp,%rsi
  400ec2:	e8 93 05 00 00       	callq  40145a <read_six_numbers>
  400ec7:	83 3c 24 01          	cmpl   $0x1,(%rsp)
  400ecb:	74 05                	je     400ed2 <phase_2+0x29>
  400ecd:	e8 66 05 00 00       	callq  401438 <explode_bomb>
  400ed2:	48 89 e3             	mov    %rsp,%rbx
  400ed5:	48 8d 6c 24 14       	lea    0x14(%rsp),%rbp
  400eda:	8b 03                	mov    (%rbx),%eax
  400edc:	01 c0                	add    %eax,%eax
  400ede:	39 43 04             	cmp    %eax,0x4(%rbx)
  400ee1:	74 05                	je     400ee8 <phase_2+0x3f>
  400ee3:	e8 50 05 00 00       	callq  401438 <explode_bomb>
  400ee8:	48 83 c3 04          	add    $0x4,%rbx
  400eec:	48 39 eb             	cmp    %rbp,%rbx
  400eef:	75 e9                	jne    400eda <phase_2+0x31>
  400ef1:	48 8b 44 24 18       	mov    0x18(%rsp),%rax
  400ef6:	64 48 33 04 25 28 00 	xor    %fs:0x28,%rax
  400efd:	00 00 
  400eff:	74 05                	je     400f06 <phase_2+0x5d>
  400f01:	e8 fa fb ff ff       	callq  400b00 <__stack_chk_fail@plt>
  400f06:	48 83 c4 28          	add    $0x28,%rsp
  400f0a:	5b                   	pop    %rbx
  400f0b:	5d                   	pop    %rbp
```

Step 1:

First we have to enter **gdb bomb** command in order to do inside the assembly file. After that we have to set break point for phase_2 in order to stop our bomb from exploding. We have to **run** the program give six interger of our choice in our second phase. So I have given 1 2 3 4 5 6 as a input.



kezang@kezang-Aspire-A315-21: ~/Downloads/Assignment 1...

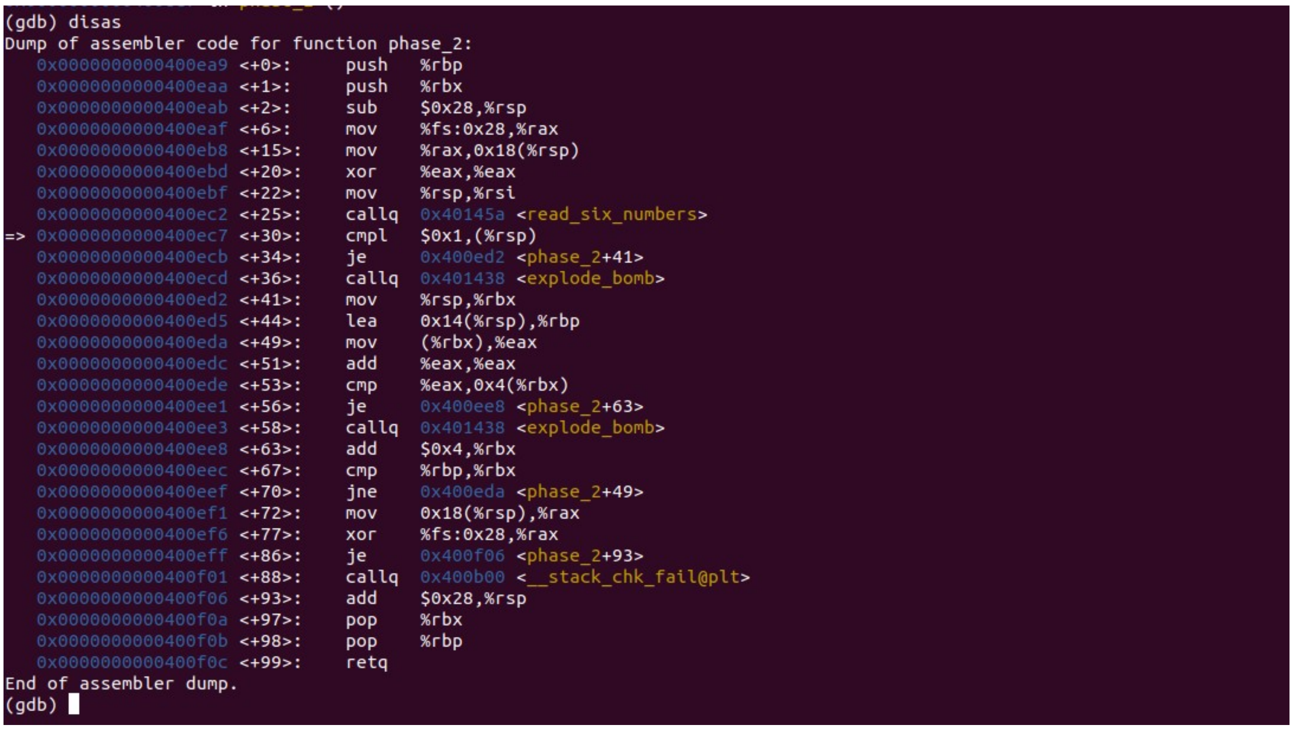
```
Type "show copying" and "show warranty" for details.
This GDB was configured as "x86_64-linux-gnu".
Type "show configuration" for configuration details.
For bug reporting instructions, please see:
<http://www.gnu.org/software/gdb/bugs/>.
Find the GDB manual and other documentation resources online at:
    <http://www.gnu.org/software/gdb/documentation/>.

For help, type "help".
Type "apropos word" to search for commands related to "word"...
Reading symbols from bomb...
(gdb) b phase_2
Breakpoint 1 at 0x400ea9
(gdb) run
Starting program: /home/kezang/Downloads/Assignment 1_2/Assignment 1/bomb002/bom
b
Welcome to my fiendish little bomb. You have 6 phases with
which to blow yourself up. Have a nice day!
Public speaking is very easy.
Phase 1 defused. How about the next one?
1 2 3 4 5 6

Breakpoint 1, 0x0000000000400ea9 in phase_2 ()
(gdb) disas
```

Step 2:

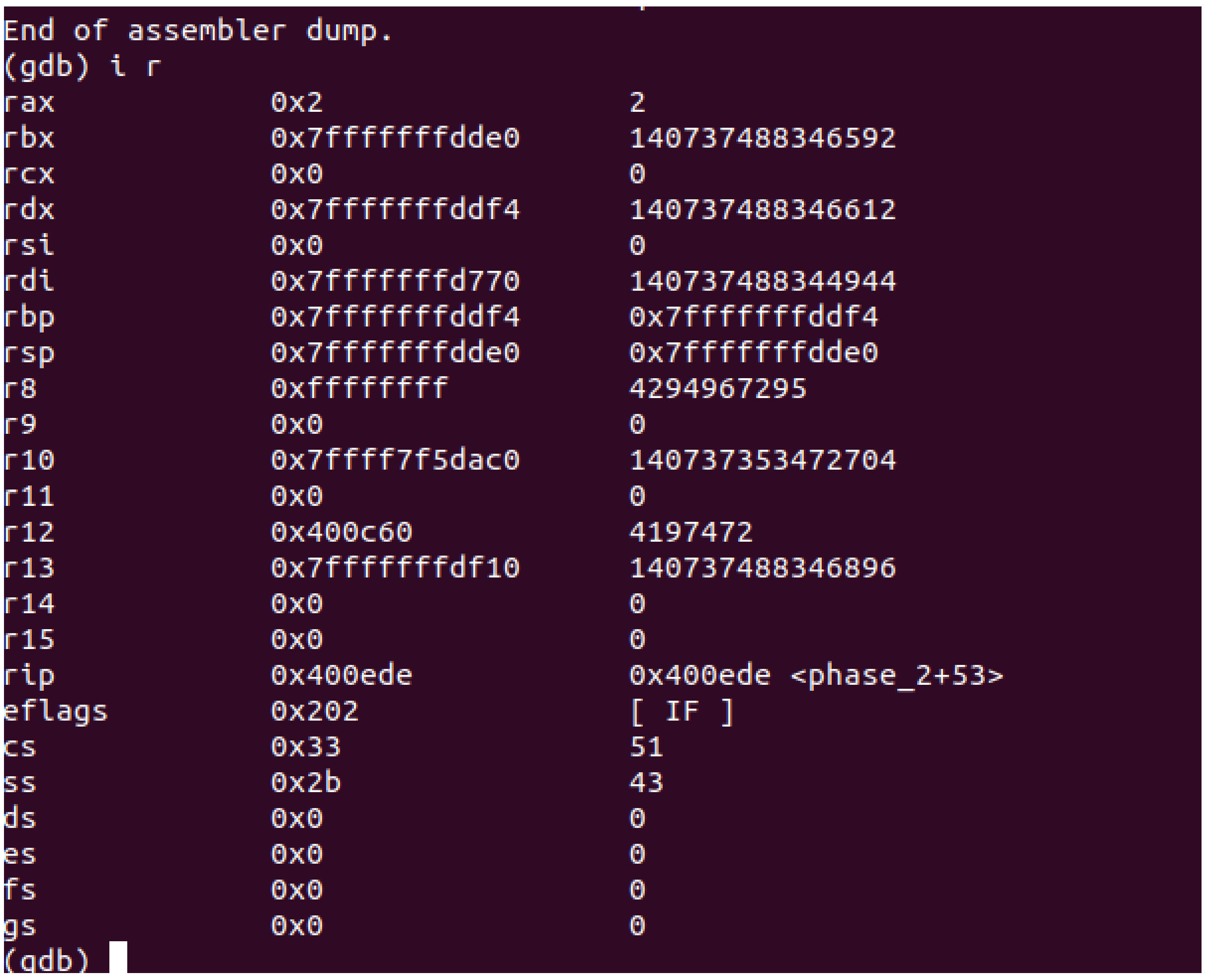
After the completion of step 1 we have to enter **disas** command in order to see assembly code and we have to focus on cmp command and we can shift our execution by using u* address of that particular line.



```
(gdb) disas
Dump of assembler code for function phase_2:
   0x0000000000400ea9 <+0>:     push   %rbp
   0x0000000000400eaa <+1>:     push   %rbx
   0x0000000000400eab <+2>:     sub    $0x28,%rsp
   0x0000000000400eaf <+6>:     mov    %fs:0x28,%rax
   0x0000000000400eb8 <+15>:    mov    %rax,0x18(%rsp)
   0x0000000000400ebd <+20>:    xor    %eax,%eax
   0x0000000000400ebf <+22>:    mov    %rsp,%rsi
   0x0000000000400ec2 <+25>:    callq  0x40145a <read_six_numbers>
=> 0x0000000000400ec7 <+30>:    cmpl   $0x1,(%rsp)
   0x0000000000400ecb <+34>:    je     0x400ed2 <phase_2+41>
   0x0000000000400ecd <+36>:    callq  0x401438 <explode_bomb>
   0x0000000000400ed2 <+41>:    mov    %rsp,%rbx
   0x0000000000400ed5 <+44>:    lea    0x14(%rsp),%rbp
   0x0000000000400eda <+49>:    mov    (%rbx),%eax
   0x0000000000400edc <+51>:    add    %eax,%eax
   0x0000000000400ede <+53>:    cmp    %eax,0x4(%rbx)
   0x0000000000400ee1 <+56>:    je     0x400ee8 <phase_2+63>
   0x0000000000400ee3 <+58>:    callq  0x401438 <explode_bomb>
   0x0000000000400ee8 <+63>:    add    $0x4,%rbx
   0x0000000000400eec <+67>:    cmp    %rbp,%rbx
   0x0000000000400eef <+70>:    jne    0x400eda <phase_2+49>
   0x0000000000400ef1 <+72>:    mov    0x18(%rsp),%rax
   0x0000000000400ef6 <+77>:    xor    %fs:0x28,%rax
   0x0000000000400eff <+86>:    je     0x400f06 <phase_2+93>
   0x0000000000400f01 <+88>:    callq  0x400b00 <__stack_chk_fail@plt>
   0x0000000000400f06 <+93>:    add    $0x28,%rsp
   0x0000000000400f0a <+97>:    pop    %rbx
   0x0000000000400f0b <+98>:    pop    %rbp
   0x0000000000400f0c <+99>:    retq
End of assembler dump.
(gdb) 
```

Step 3:

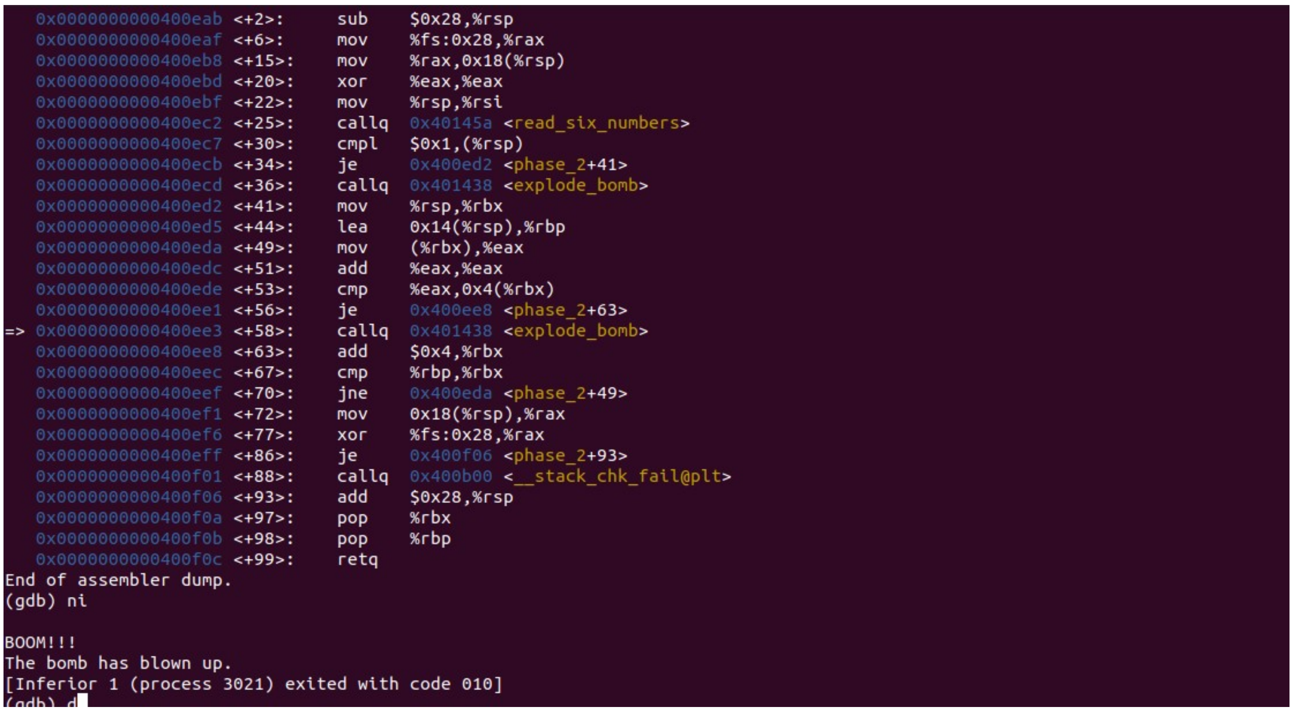
We have to compare value of rsp register with our input. We can find the value of register using **i r** command and in order to get value of rsp register we have to enter **x/d address** of the rsp register so rsp is equal to user input so it will directly jump to phase_2+41 line.



```
End of assembler dump.
(gdb) i r
rax            0x2                 2
rbx            0x7fffffffdde0      140737488346592
rcx            0x0                 0
rdx            0x7fffffffddf4      140737488346612
rsi            0x0                 0
rdi            0x7fffffffd770      140737488344944
rbp            0x7fffffffddf4      0x7fffffffddf4
rsp            0x7fffffffdde0      0x7fffffffdde0
r8             0xffffffff          4294967295
r9             0x0                 0
r10            0x7ffff7f5dac0      140737353472704
r11            0x0                 0
r12            0x400c60            4197472
r13            0x7fffffffdf10      140737488346896
r14            0x0                 0
r15            0x0                 0
rip            0x400ede            0x400ede <phase_2+53>
eflags         0x202               [ IF ]
cs             0x33                51
ss             0x2b                43
ds             0x0                 0
es             0x0                 0
fs             0x0                 0
gs             0x0                 0
(gdb)
```

Step 4:

After completion of step 3 we have to go again using disas command to assembly code and again see each line using **ni** and **disas** command. Again we have to compare eax and rbx value and if it is equal then it will jump to phase_2+63 line. It will again compare with value of rbx and rbp register and if it is not equal it will jump to phase_2+49 line and if it is equal then bomb will explode.

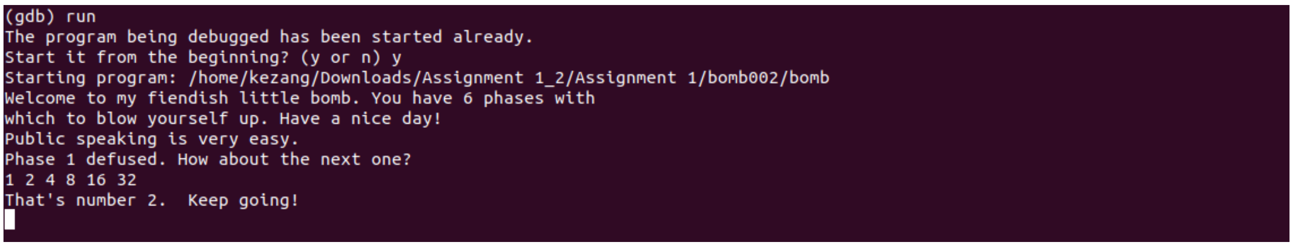


```
   0x0000000000400eab <+2>:     sub    $0x28,%rsp
   0x0000000000400eaf <+6>:     mov    %fs:0x28,%rax
   0x0000000000400eb8 <+15>:    mov    %rax,0x18(%rsp)
   0x0000000000400ebd <+20>:    xor    %eax,%eax
   0x0000000000400ebf <+22>:    mov    %rsp,%rsi
   0x0000000000400ec2 <+25>:    callq  0x40145a <read_six_numbers>
   0x0000000000400ec7 <+30>:    cmpl   $0x1,(%rsp)
   0x0000000000400ecb <+34>:    je     0x400ed2 <phase_2+41>
   0x0000000000400ecd <+36>:    callq  0x401438 <explode_bomb>
   0x0000000000400ed2 <+41>:    mov    %rsp,%rbx
   0x0000000000400ed5 <+44>:    lea    0x14(%rsp),%rbp
   0x0000000000400eda <+49>:    mov    (%rbx),%eax
   0x0000000000400edc <+51>:    add    %eax,%eax
   0x0000000000400ede <+53>:    cmp    %eax,0x4(%rbx)
   0x0000000000400ee1 <+56>:    je     0x400ee8 <phase_2+63>
=> 0x0000000000400ee3 <+58>:    callq  0x401438 <explode_bomb>
   0x0000000000400ee8 <+63>:    add    $0x4,%rbx
   0x0000000000400eec <+67>:    cmp    %rbp,%rbx
   0x0000000000400eef <+70>:    jne    0x400eda <phase_2+49>
   0x0000000000400ef1 <+72>:    mov    0x18(%rsp),%rax
   0x0000000000400ef6 <+77>:    xor    %fs:0x28,%rax
   0x0000000000400eff <+86>:    je     0x400f06 <phase_2+93>
   0x0000000000400f01 <+88>:    callq  0x400b00 <__stack_chk_fail@plt>
   0x0000000000400f06 <+93>:    add    $0x28,%rsp
   0x0000000000400f0a <+97>:    pop    %rbx
   0x0000000000400f0b <+98>:    pop    %rbp
   0x0000000000400f0c <+99>:    retq
End of assembler dump.
(gdb) ni

BOOM!!!
The bomb has blown up.
[Inferior 1 (process 3021) exited with code 010]
(gdb) d
```

Step 5:

From above step we are able to get hint of answer that could be our key for the phase 2 because we have got 1 2 4 so we can guess that it will be addition of same number till six interger. So we get the key for the phase 2 is **1 2 4 8 16 32**. so when we put key for phase 2 we are able to defuse the bomb.



```
(gdb) run
The program being debugged has been started already.
Start it from the beginning? (y or n) y
Starting program: /home/kezang/Downloads/Assignment 1_2/Assignment 1/bomb002/bomb
Welcome to my fiendish little bomb. You have 6 phases with
which to blow yourself up. Have a nice day!
Public speaking is very easy.
Phase 1 defused. How about the next one?
1 2 4 8 16 32
That's number 2.  Keep going!
```