Assignment 1

ITS304

Submitted by: Kinley Gyeltshen

Enrollment no: 12190060

Class: BSc IT 'A'

Necessary key words to understand

disas – to disassemble the assembly code.

b – To set breakpoints.

r – to run.

i r – To read information about registers.

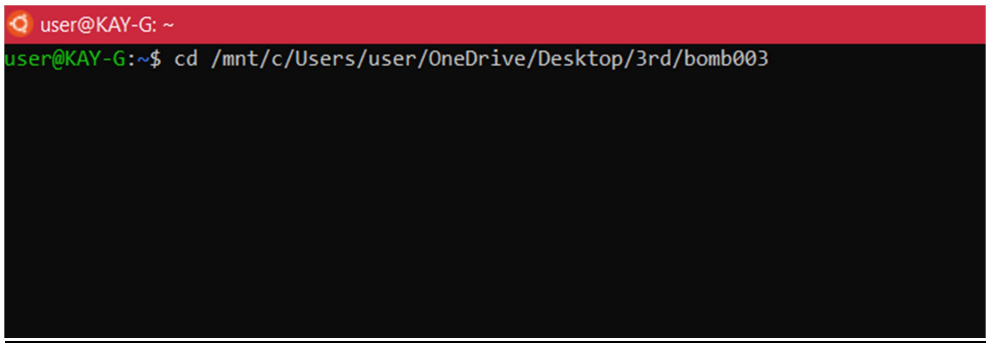
q – To quit from the debugger.

Jne – Jump if not equal.

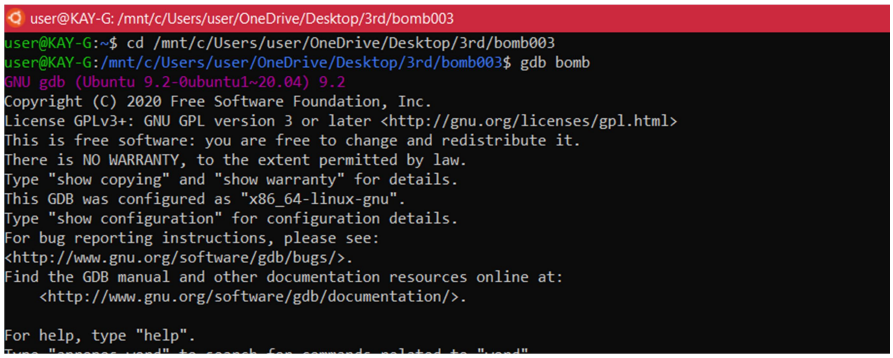
Cmp – compare
x/s – To know the number of inputs required.

Phase 1

- To first execute the code, we have to get inside the directory in which our bomb is in.



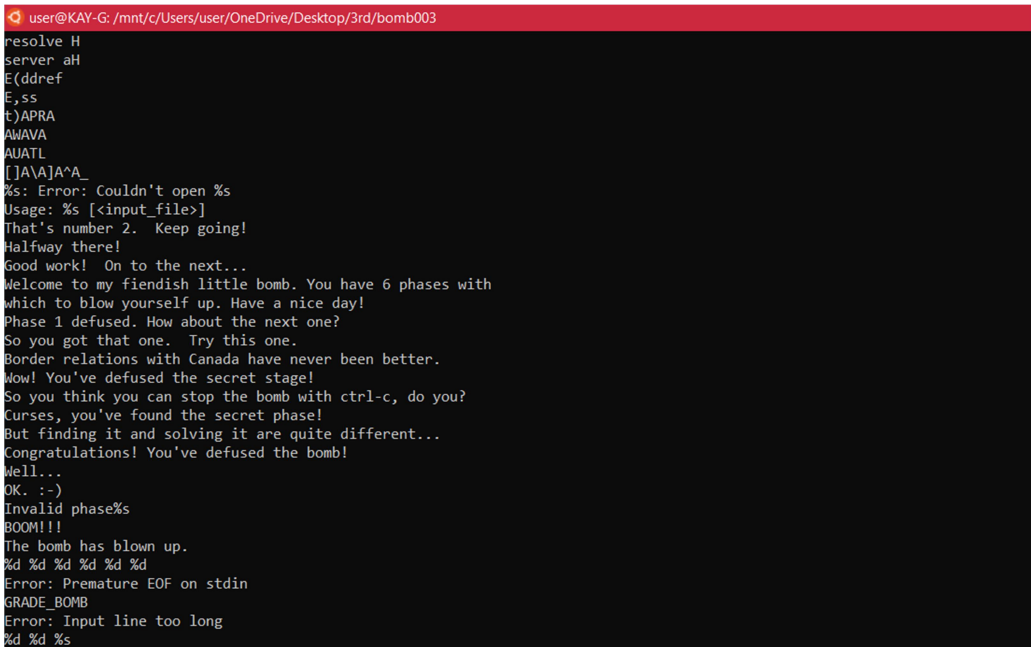
- Then we have to use the gdb debugger to debug our code / to understand what is actually happening in our code.



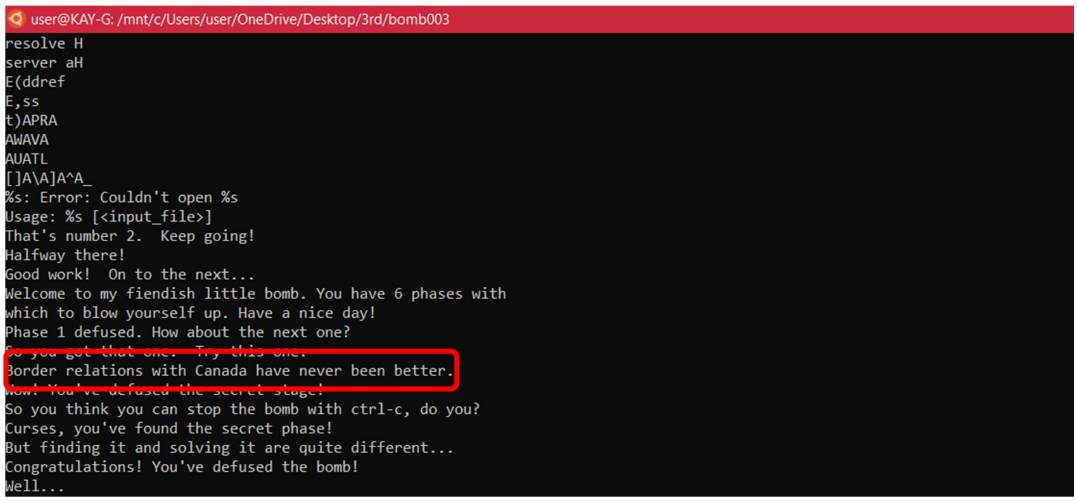
- After we enter inside our debugger we then set a breakpoint at phase 1, in order to prevent our bomb from exploding incase we guess the wrong answer. Then we run the program, and we are welcomed by Dr.Evil with a message. We then give our first input, and dissemble the assembly code to see the operations happening within the code.

- From the <strings_not_equal> function we know that out input must be a string.

❖ We “strings bomb” to display all the strings in the code.

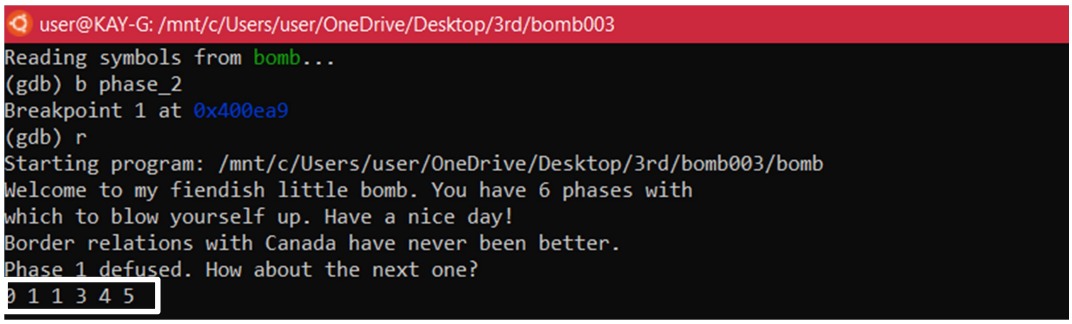


❖ From the strings displayed we find a string that is completely unrelated to the Dr.evil’s speech. Hence we find the our input for the first phase.

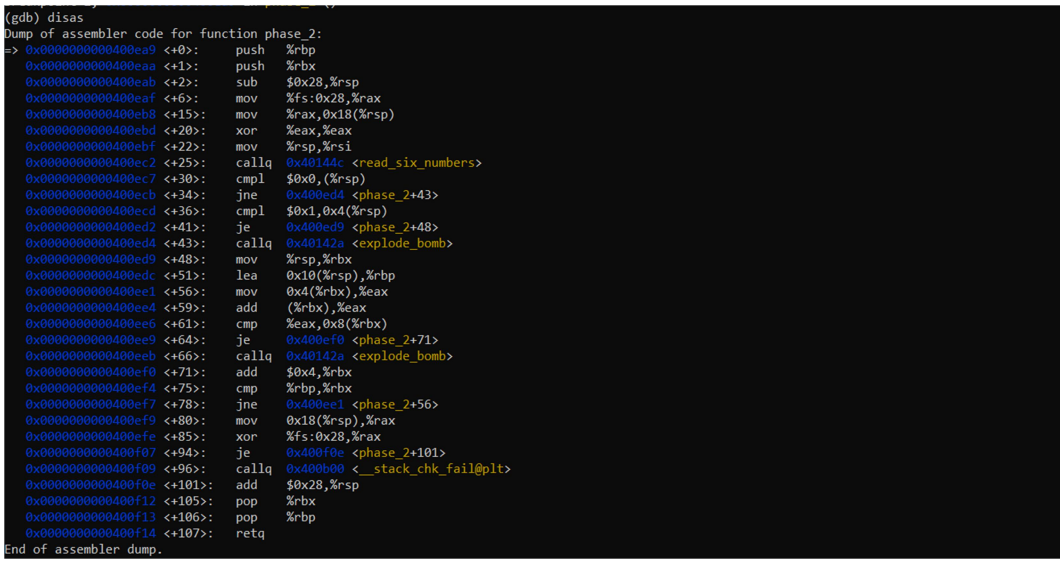


Phase 2

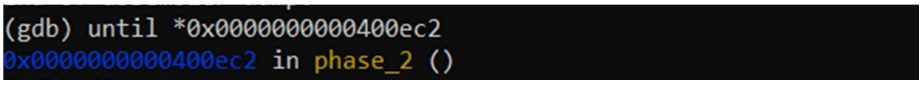
- After diffusing the first bomb, we then go on to the next phase, there we enter a random input.



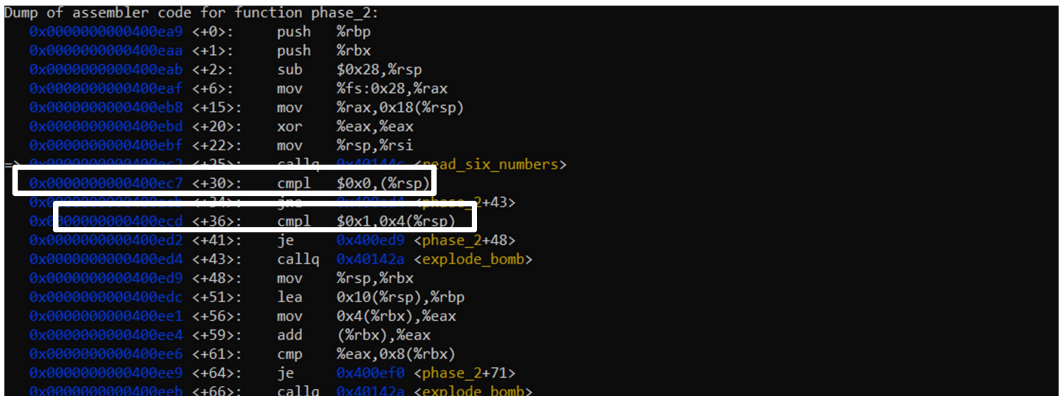
- Then we disassemble the code using the “disas” code. From there we understand that the input is six numbers from the <read_six_numbers> function.



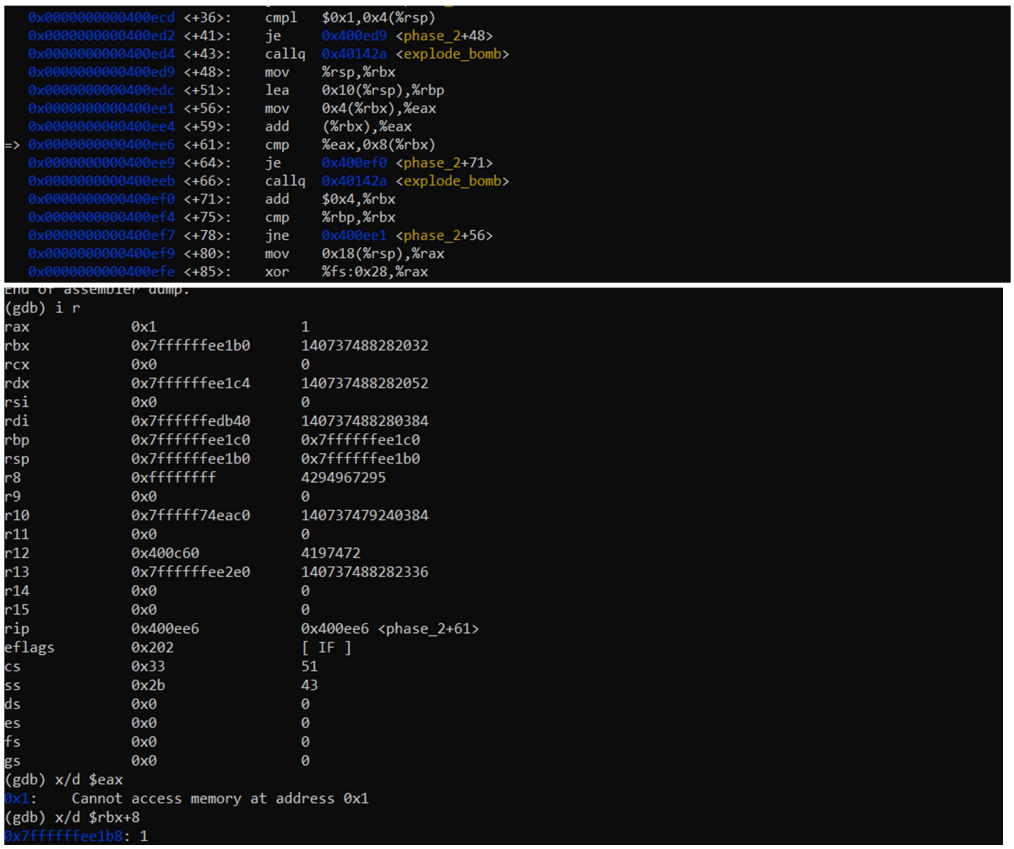
- We mostly focus on the callq functions in the code and compare functions.



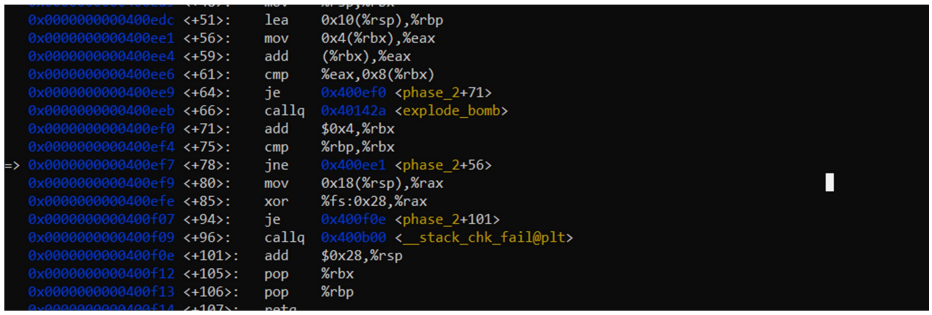
- We move on to the compare function where it compares our first number with 0,if its not equal then the bomb explodes, if it does not then it proves that 0 is the first input required. For the second input as well 1 is compared to the second number if its correct then it proceeds if not it explodes.

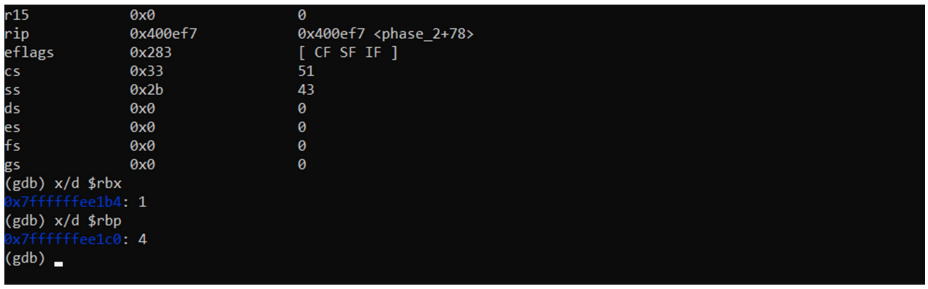


- For the third value, we again go to compare function. Where it compares if value at register %eax is equal to value at address 0x8*%rbx. If it is equal it proceeds if not the bomb explodes. We check the values by typing " i r" and compare the values of %eax and 0x8%rbx which are both 1 so third value is 1.

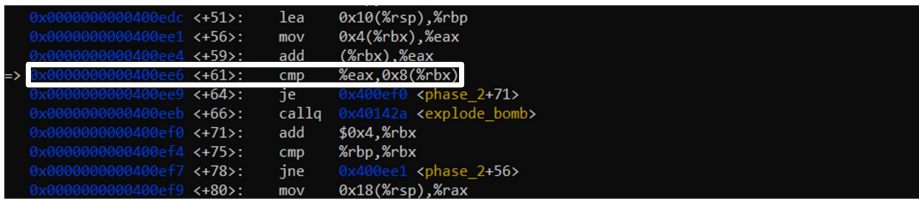


❖ We again move to the next compare function. Here it compares the value at %rbp and %rbx which are not equal so we jump to line 56.

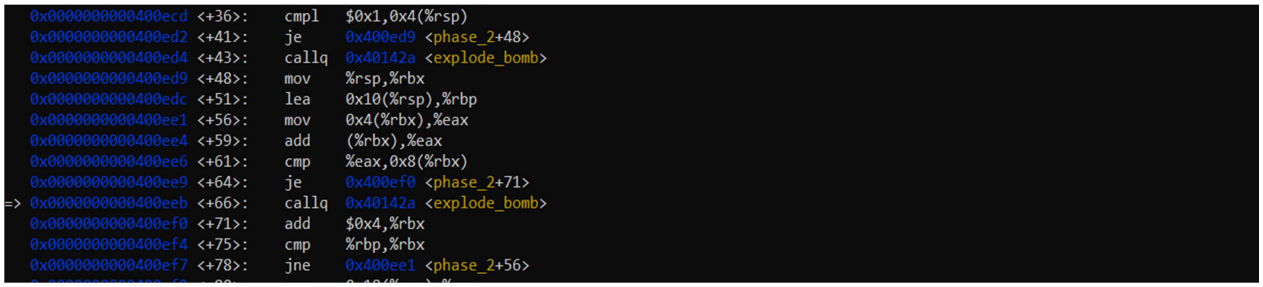




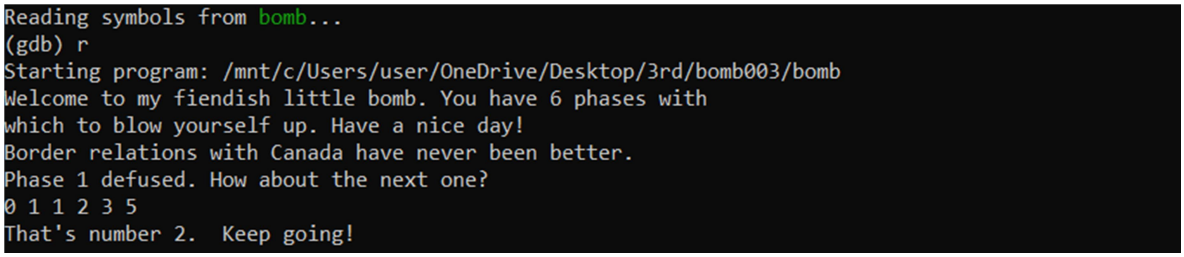
❖ We move to the next compare function. It compares the value at %eax and 0x8(%rbx),if they are equal it will proceed if not it will explode.



The value at 0x8(%rbx) is the value we entered that is 3 so it is being compared to %eax which has value 2, which is not equal so hence the bomb explodes. Because the bomb exploded we now know that the fourth value is 2. So currently we have 0 1 1 2 as our input, if we closely look at the inputs it follows a certain pattern that is the Fibonacci series so hence we can predict the next two values.

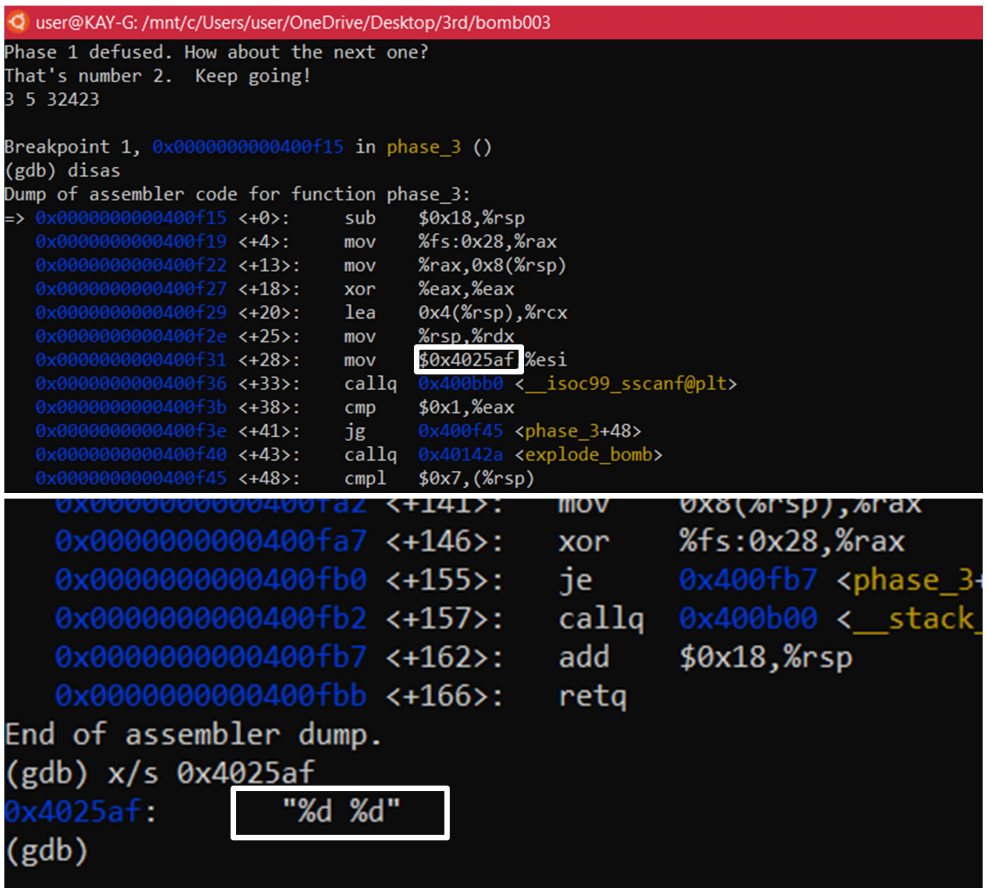


- We can predict that the inputs are 0 1 1 2 3 5. We run the program again and give the correct answer, and bomb gets difused.

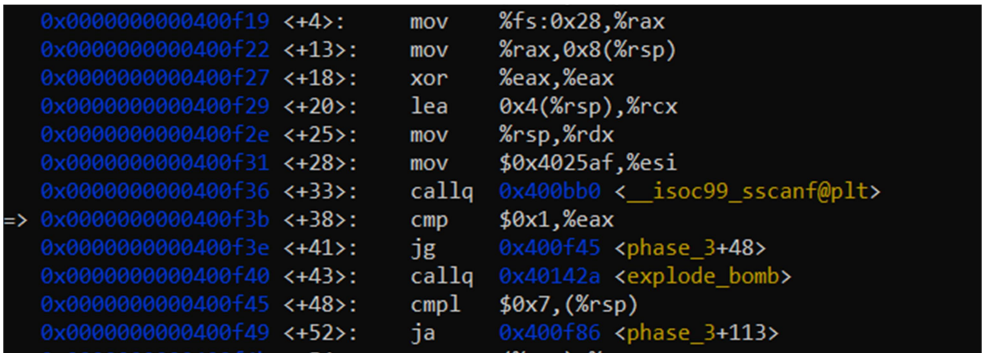


Phase 3

- For phase 3 we enter a random number and enter. Then we disassemble the code and from the address above the callq function we can figure out the type of input and number of input required. So we know that we need two integers.



- We jump to the compare statement where it compares if our first input is greater than 1. If it is greater than it does not explode.



- The next compare function compares our value with 7 and checks it is less than 7 , in other words our first digit must be between 1 -7.

```
   0x0000000000400f3b <+38>:    cmp    $0x1,%eax
   0x0000000000400f3e <+41>:    jg     0x400f45 <phase_3+48>
   0x0000000000400f40 <+43>:    callq  0x40142a <explode_bomb>
=> 0x0000000000400f45 <+48>:    cmpl   $0x7,(%rsp)
   0x0000000000400f49 <+52>:    ja     0x400f86 <phase_3+113>
   0x0000000000400f4b <+54>:    mov    (%rsp),%eax
   0x0000000000400f4e <+57>:    jmpq   *0x402420(,%rax,8)
   0x0000000000400f55 <+64>:    mov    $0xc6,%eax
   0x0000000000400f5a <+69>:    jmp    0x400f97 <phase_3+130>
   0x0000000000400f5c <+71>:    mov    $0x31e,%eax
```

- It compares the value at address 0x4(%rsp) and %eax, and if its equal then it's the correct value if not the bomb explodes. The bomb explodes as our value is 5 and the value at %eax is 665. So we know that the second input is 665.

```
   0x0000000000400f92 <+125>:   mov    $0x3f,%eax
=> 0x0000000000400f97 <+130>:   cmp    0x4(%rsp),%eax
   0x0000000000400f9b <+134>:   je     0x400fa2 <phase_3+141>
   0x0000000000400f9d <+136>:   callq  0x40142a <explode_bomb>
   0x0000000000400fa2 <+141>:   mov    0x8(%rsp),%rax
   0x0000000000400fa7 <+146>:   xor    %fs:0x28,%rax
(gdb) i r
rax            0x299               665
rbx            0x7fffffffee2d8     140737488282328
rcx            0x0                 0
rdx            0x7fffffffee1c4     140737488282052
rsi            0x0                 0
rdi            0x7ffffffedb70      140737488280432
rbp            0x0                 0x0
rsp            0x7fffffffee1c0     0x7fffffffee1c0
r8             0x20                32
r9             0x0                 0
r10            0x7fffff74eac0      140737479240384
r11            0x0                 0
r12            0x400c60            4197472
r13            0x7fffffffee2d0     140737488282320
r14            0x0                 0
r15            0x0                 0
rip            0x400f9d            0x400f9d <phase_3+136>
eflags         0x202               [ IF ]
cs             0x33                51
ss             0x2b                43
ds             0x0                 0
es             0x0                 0
fs             0x0                 0
gs             0x0                 0
(gdb) x/d $rsp+4
0x7fffffffee1c4: 5
(gdb)
```

- So the two inputs are 3 and 665.

```
(gdb) r
Starting program: /mnt/c/Users/user/OneDrive/Desktop/3rd/bomb003/bomb
Welcome to my fiendish little bomb. You have 6 phases with
which to blow yourself up. Have a nice day!
Border relations with Canada have never been better.
Phase 1 defused. How about the next one?
0 1 1 2 3 5
That's number 2.  Keep going!
3 665
Halfway there!
```