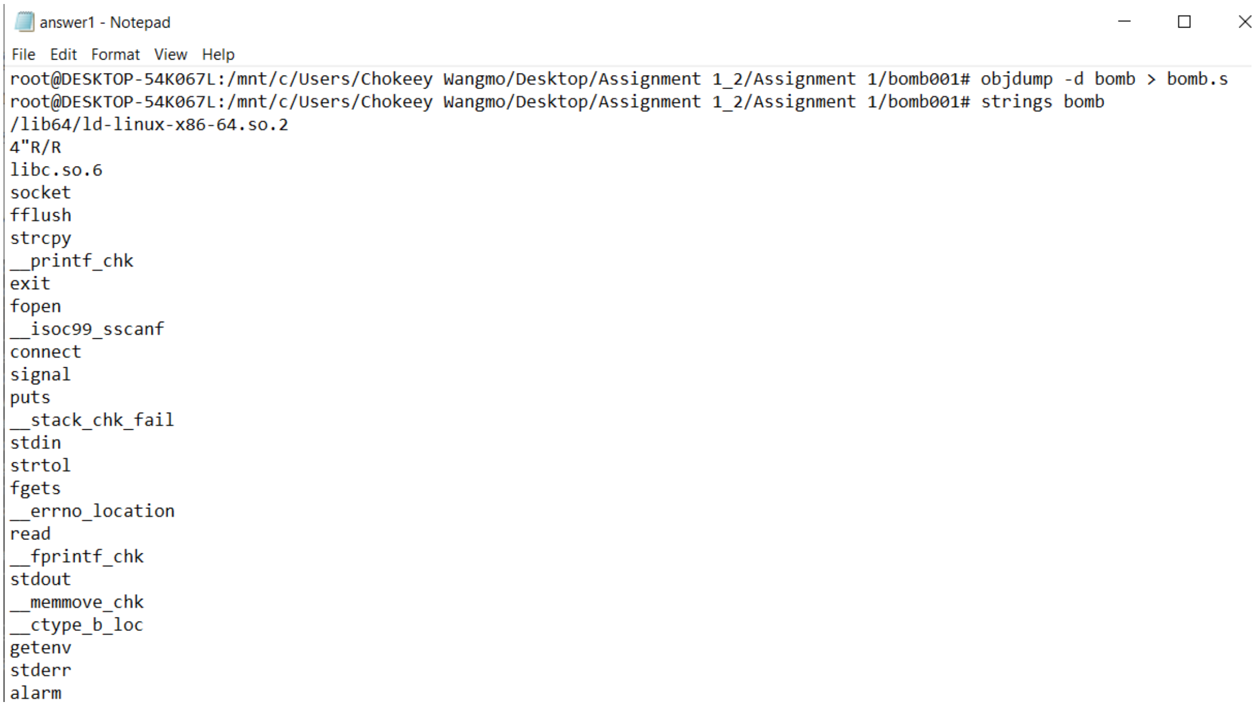
Phase_1

Firstly, to access the bomb file we have to give permission to the system. To do this “chmod 777 bomb” command was executed.

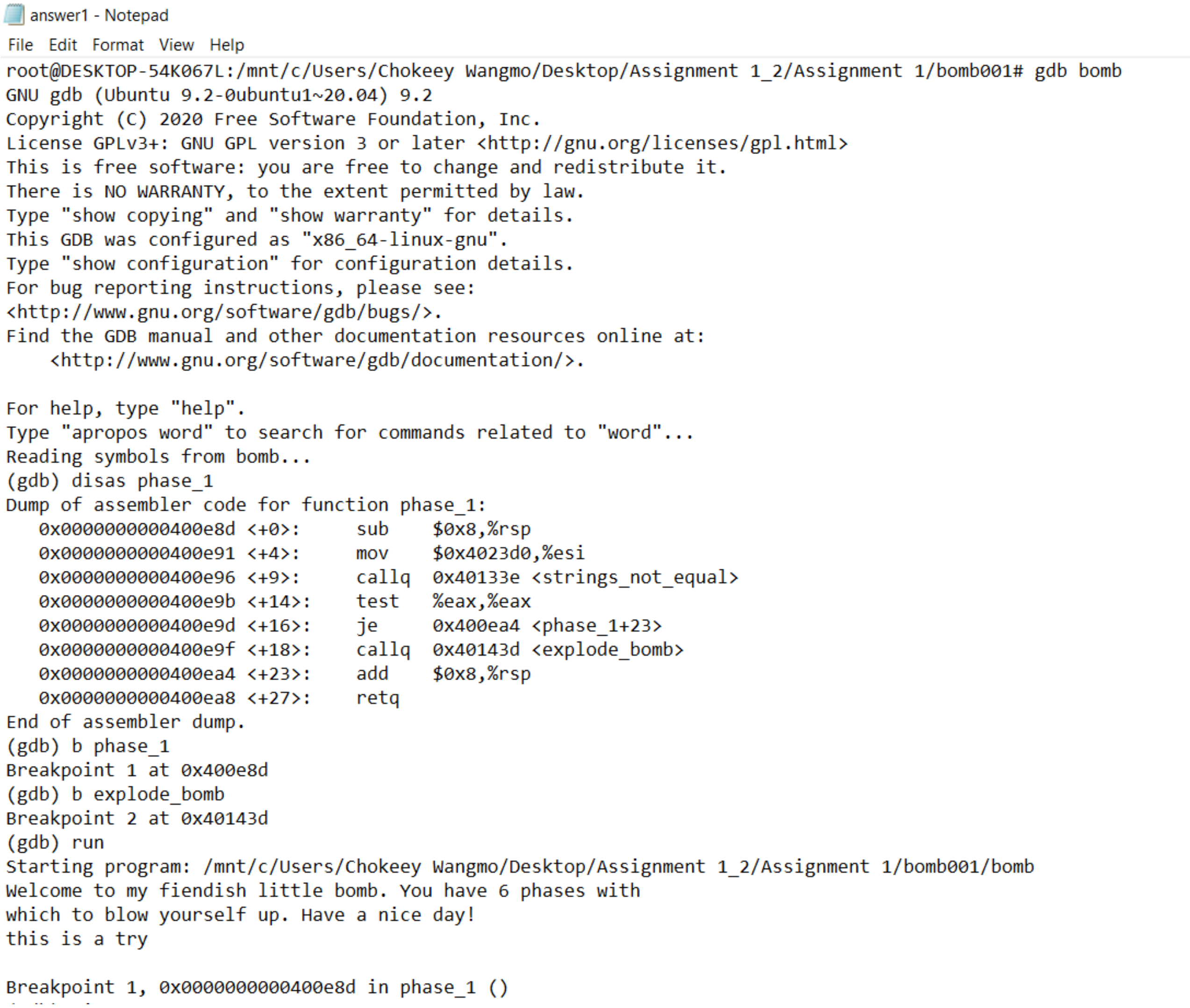
After that bomb file was converted assembly file using “objdump” command as shown below.

To print all the information about this program we can start with “strings bomb” command. This gives us more information about the bomb.



```
root@DESKTOP-54K067L:/mnt/c/Users/Chokeey Wangmo/Desktop/Assignment 1_2/Assignment 1/bomb001# objdump -d bomb > bomb.s
root@DESKTOP-54K067L:/mnt/c/Users/Chokeey Wangmo/Desktop/Assignment 1_2/Assignment 1/bomb001# strings bomb
/lib64/ld-linux-x86-64.so.2
4"R/R
libc.so.6
socket
fflush
strcpy
__printf_chk
exit
fopen
__isoc99_sscanf
connect
signal
puts
__stack_chk_fail
stdin
strtol
fgets
__errno_location
read
__fprintf_chk
stdout
__memmove_chk
__ctype_b_loc
getenv
stderr
alarm
```

To start debugging, debugger gdb was used to executed the assembly programs. Then breakpoint was set using “b phase_1” command to ensure that the bomb doesn’t blow up when the program is run. Breakpoint is set before the program is run. To view the assembler code of phase_1 type “disas phase_1” command. When we run the program, it will ask for the input. We can give any input for the try. Since we set the breakpoint, it will save us from bomb detonation. As soon as we enter our string and hit enter, the breakpoint stops execution of the program.



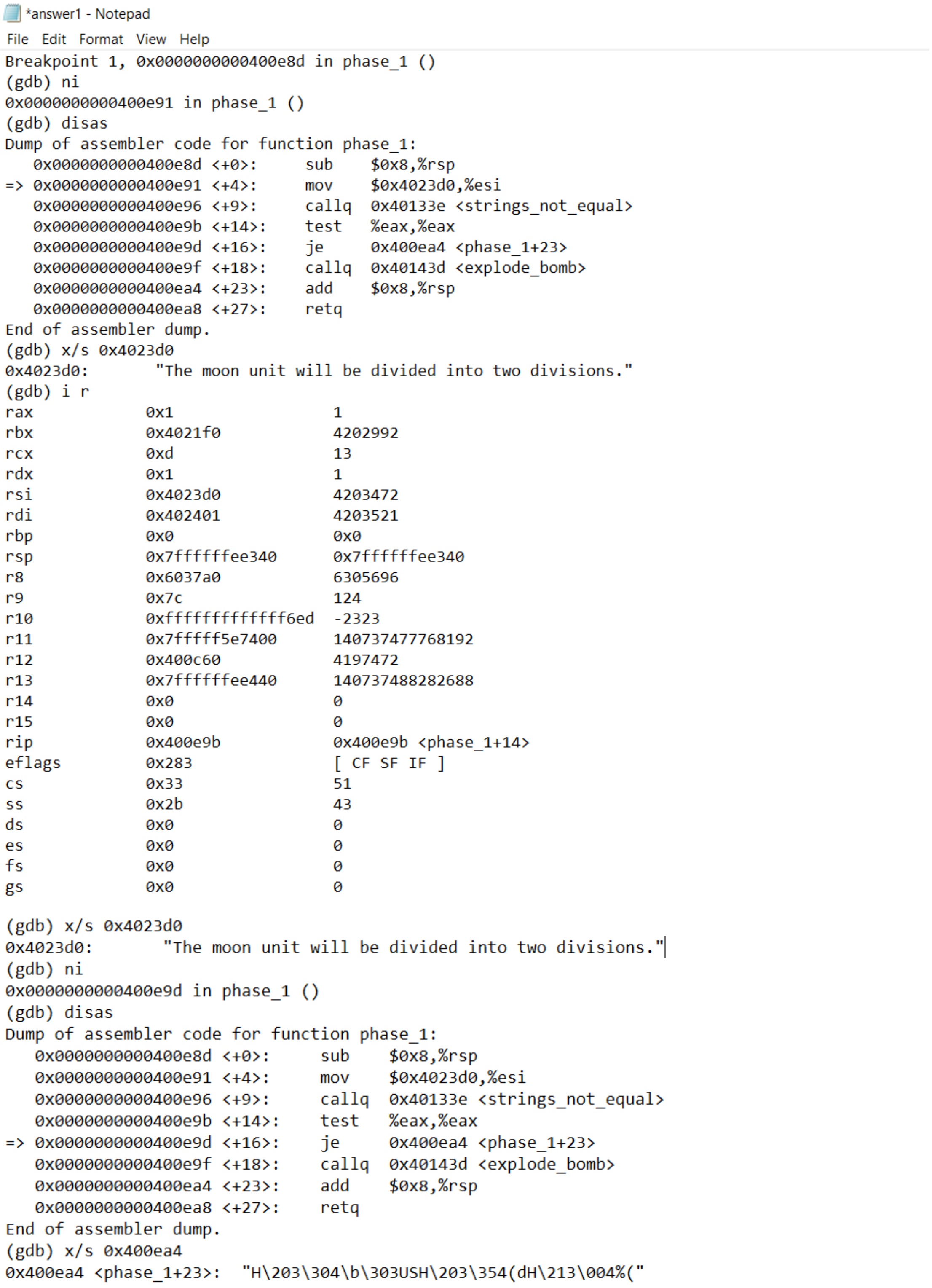
```
root@DESKTOP-54K067L:/mnt/c/Users/Chokeey Wangmo/Desktop/Assignment 1_2/Assignment 1/bomb001# gdb bomb
GNU gdb (Ubuntu 9.2-0ubuntu1~20.04) 9.2
Copyright (C) 2020 Free Software Foundation, Inc.
License GPLv3+: GNU GPL version 3 or later <http://gnu.org/licenses/gpl.html>
This is free software: you are free to change and redistribute it.
There is NO WARRANTY, to the extent permitted by law.
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) disas phase_1
Dump of assembler code for function phase_1:
   0x0000000000400e8d <+0>:     sub    $0x8,%rsp
   0x0000000000400e91 <+4>:     mov    $0x4023d0,%esi
   0x0000000000400e96 <+9>:     callq  0x40133e <strings_not_equal>
   0x0000000000400e9b <+14>:    test   %eax,%eax
   0x0000000000400e9d <+16>:    je     0x400ea4 <phase_1+23>
   0x0000000000400e9f <+18>:    callq  0x40143d <explode_bomb>
   0x0000000000400ea4 <+23>:    add    $0x8,%rsp
   0x0000000000400ea8 <+27>:    retq
End of assembler dump.
(gdb) b phase_1
Breakpoint 1 at 0x400e8d
(gdb) b explode_bomb
Breakpoint 2 at 0x40143d
(gdb) run
Starting program: /mnt/c/Users/Chokeey Wangmo/Desktop/Assignment 1_2/Assignment 1/bomb001/bomb
Welcome to my fiendish little bomb. You have 6 phases with
which to blow yourself up. Have a nice day!
this is a try

Breakpoint 1, 0x0000000000400e8d in phase_1 ()
```

By looking at the code there is a call to function “strings_not_equal” which hints that the input type of this phase is string. Now look for the value of the register esi, which is stored at $0x4023d0. To view the value of $0x4023d0 use “x/s 0x4023d0” to display the strings. The strings stored was "The moon unit will be divided into two divisions."

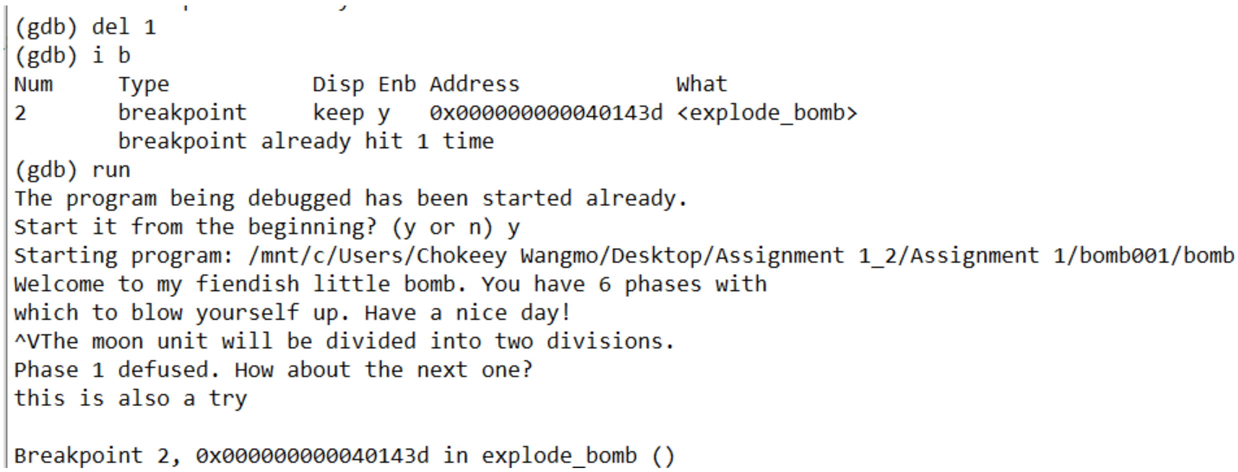
Moving to the 4th instruction, check whether the test is equal or not. If it is equal then move to phase_1+23 which move to line 23 otherwise bomb will blow up.



```
Breakpoint 1, 0x0000000000400e8d in phase_1 ()
(gdb) ni
0x0000000000400e91 in phase_1 ()
(gdb) disas
Dump of assembler code for function phase_1:
   0x0000000000400e8d <+0>:     sub    $0x8,%rsp
=> 0x0000000000400e91 <+4>:     mov    $0x4023d0,%esi
   0x0000000000400e96 <+9>:     callq  0x40133e <strings_not_equal>
   0x0000000000400e9b <+14>:    test   %eax,%eax
   0x0000000000400e9d <+16>:    je     0x400ea4 <phase_1+23>
   0x0000000000400e9f <+18>:    callq  0x40143d <explode_bomb>
   0x0000000000400ea4 <+23>:    add    $0x8,%rsp
   0x0000000000400ea8 <+27>:    retq
End of assembler dump.
(gdb) x/s 0x4023d0
0x4023d0:       "The moon unit will be divided into two divisions."
(gdb) i r
rax            0x1                 1
rbx            0x4021f0            4202992
rcx            0xd                 13
rdx            0x1                 1
rsi            0x4023d0            4203472
rdi            0x402401            4203521
rbp            0x0                 0x0
rsp            0x7fffffffe340      0x7fffffffe340
r8             0x6037a0            6305696
r9             0x7c                124
r10            0xfffffffffffff6ed  -2323
r11            0x7ffff5e7400       140737477768192
r12            0x400c60            4197472
r13            0x7fffffffe440      140737488282688
r14            0x0                 0
r15            0x0                 0
rip            0x400e9b            0x400e9b <phase_1+14>
eflags         0x283               [ CF SF IF ]
cs             0x33                51
ss             0x2b                43
ds             0x0                 0
es             0x0                 0
fs             0x0                 0
gs             0x0                 0

(gdb) x/s 0x4023d0
0x4023d0:       "The moon unit will be divided into two divisions."
(gdb) ni
0x0000000000400e9d in phase_1 ()
(gdb) disas
Dump of assembler code for function phase_1:
   0x0000000000400e8d <+0>:     sub    $0x8,%rsp
   0x0000000000400e91 <+4>:     mov    $0x4023d0,%esi
   0x0000000000400e96 <+9>:     callq  0x40133e <strings_not_equal>
   0x0000000000400e9b <+14>:    test   %eax,%eax
=> 0x0000000000400e9d <+16>:    je     0x400ea4 <phase_1+23>
   0x0000000000400e9f <+18>:    callq  0x40143d <explode_bomb>
   0x0000000000400ea4 <+23>:    add    $0x8,%rsp
   0x0000000000400ea8 <+27>:    retq
End of assembler dump.
(gdb) x/s 0x400ea4
0x400ea4 <phase_1+23>:  "H\203\304\b\303USH\203\354(dH\213\004%("
```

Before running the program delete the breakpoint and run the program using the “run” or “r” command. Provide the string value that we got. The input and the string stored in register matches and the bomb is successfully diffused.



```
(gdb) del 1
(gdb) i b
Num     Type           Disp Enb Address            What
2       breakpoint     keep y   0x000000000040143d <explode_bomb>
        breakpoint already hit 1 time
(gdb) run
The program being debugged has been started already.
Start it from the beginning? (y or n) y
Starting program: /mnt/c/Users/Chokeey Wangmo/Desktop/Assignment 1_2/Assignment 1/bomb001/bomb
Welcome to my fiendish little bomb. You have 6 phases with
which to blow yourself up. Have a nice day!
^VThe moon unit will be divided into two divisions.
Phase 1 defused. How about the next one?
this is also a try

Breakpoint 2, 0x000000000040143d in explode_bomb ()
```