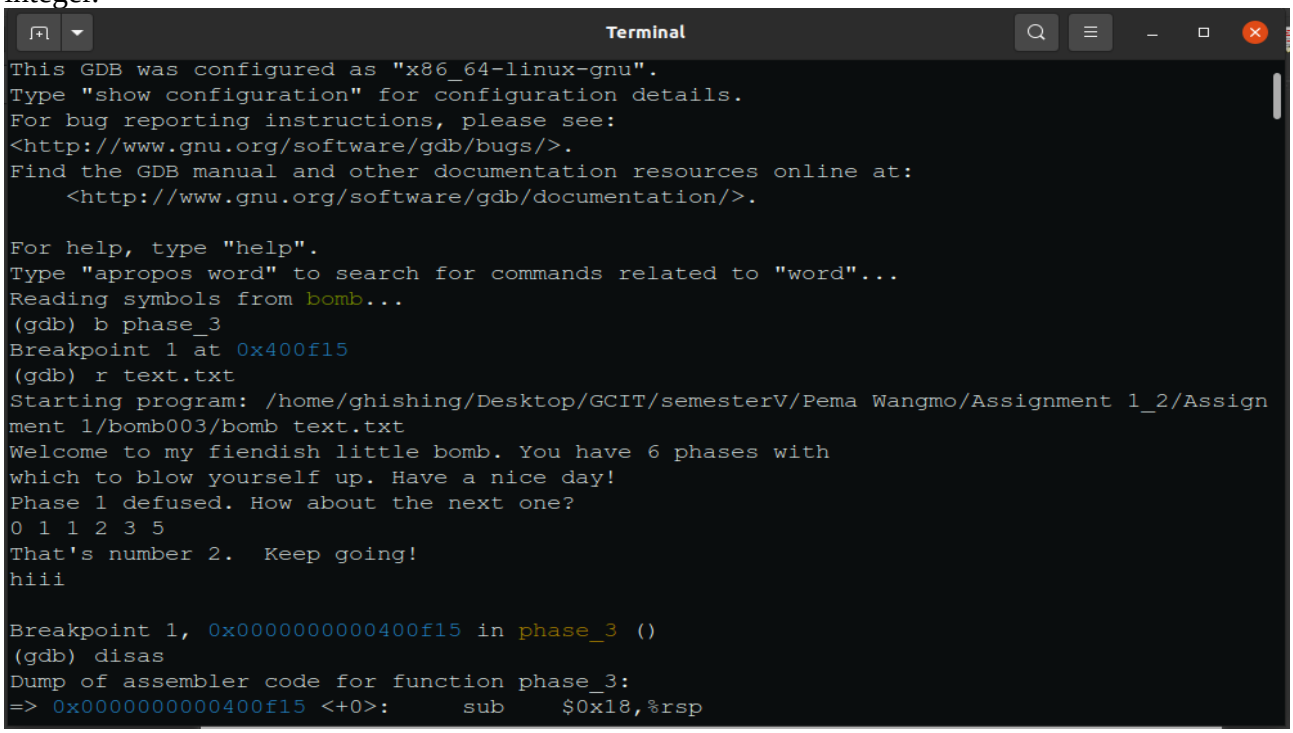
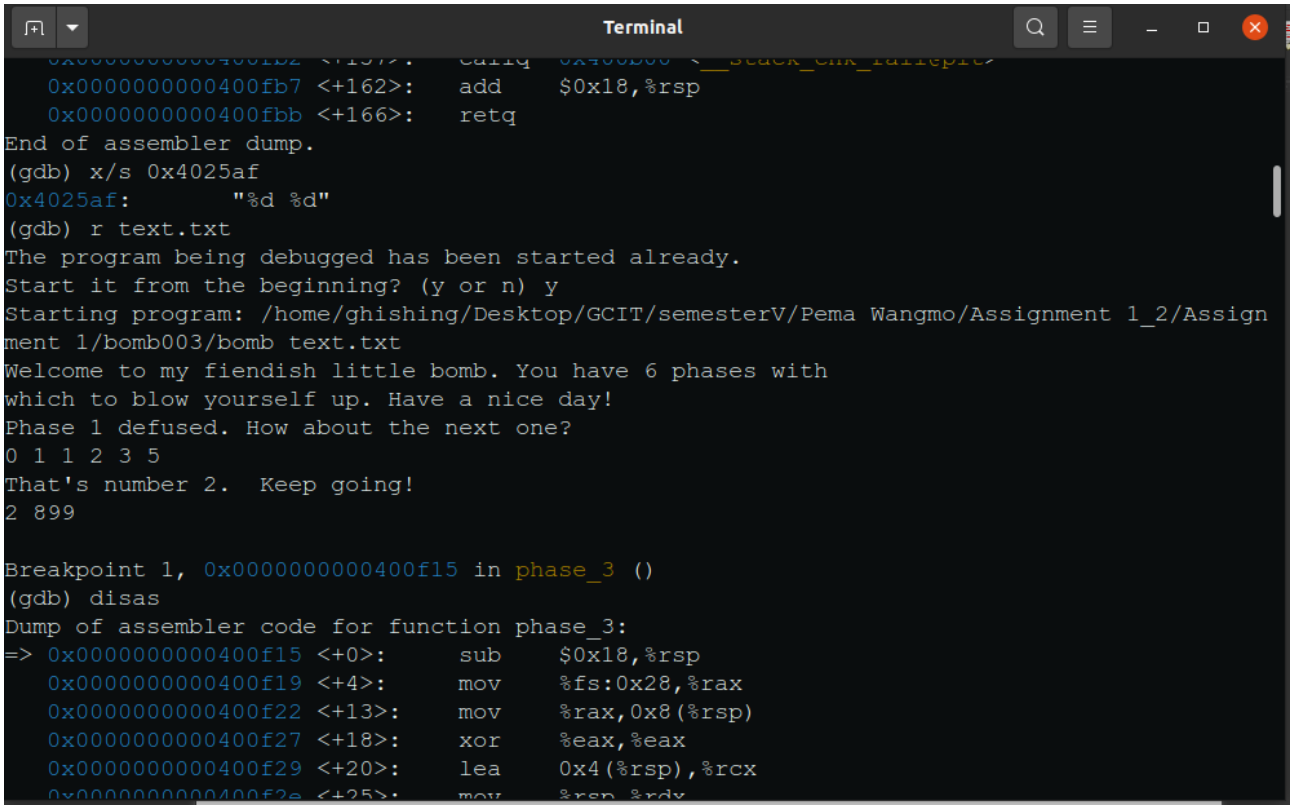
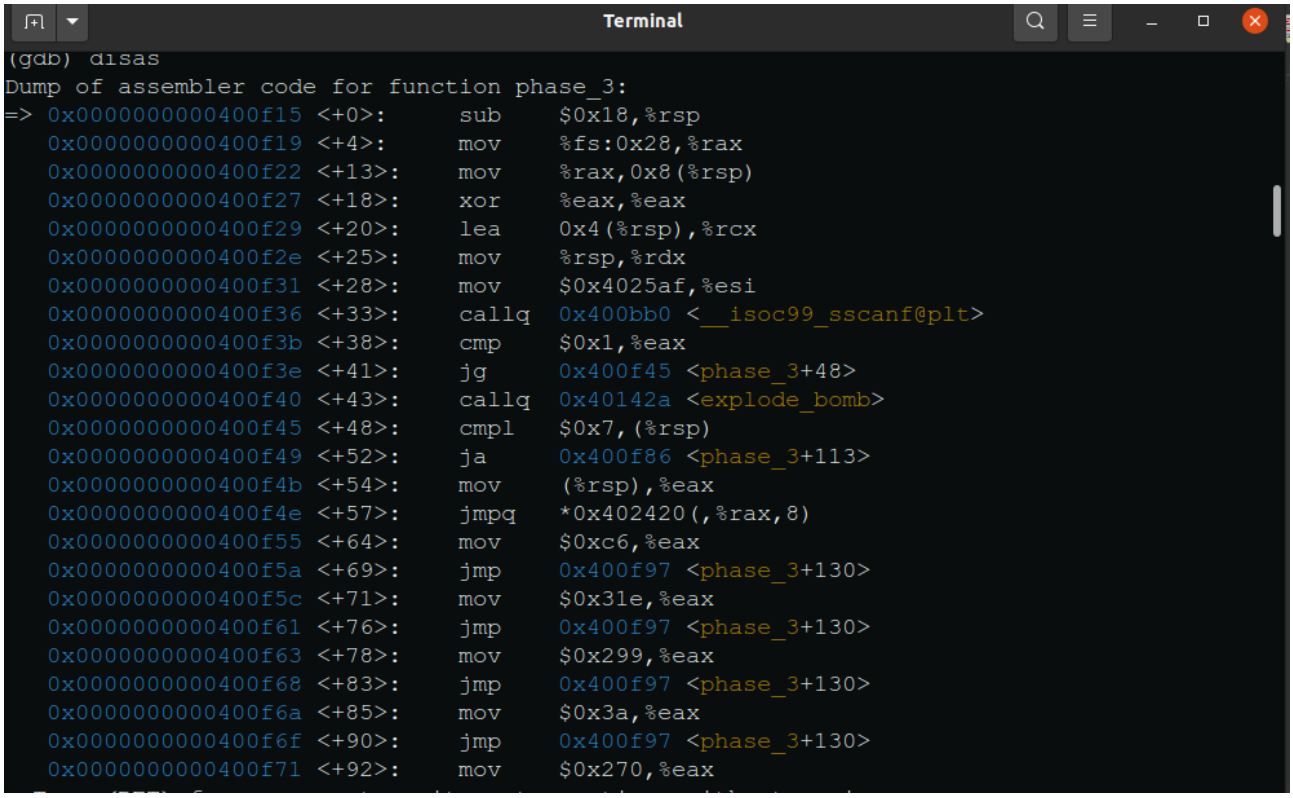
Phase 03

To begin with 3rd phase, first of all, break point was set to avoid exploding of bomb in phase 3. The program was run and random value was inputted. Then the program code was disassemble and and first function was <__isoc99_sscanf@plt> and basacilly through this function we can understand that we have to provide user input. Before executing the function, there is a value stored in hexadecimal form i.e., $0x4025af . To view this value “x/s 0x4025af” command was executed and it displayed "%d %d". From this we can understand that this program asks user to input two integer.

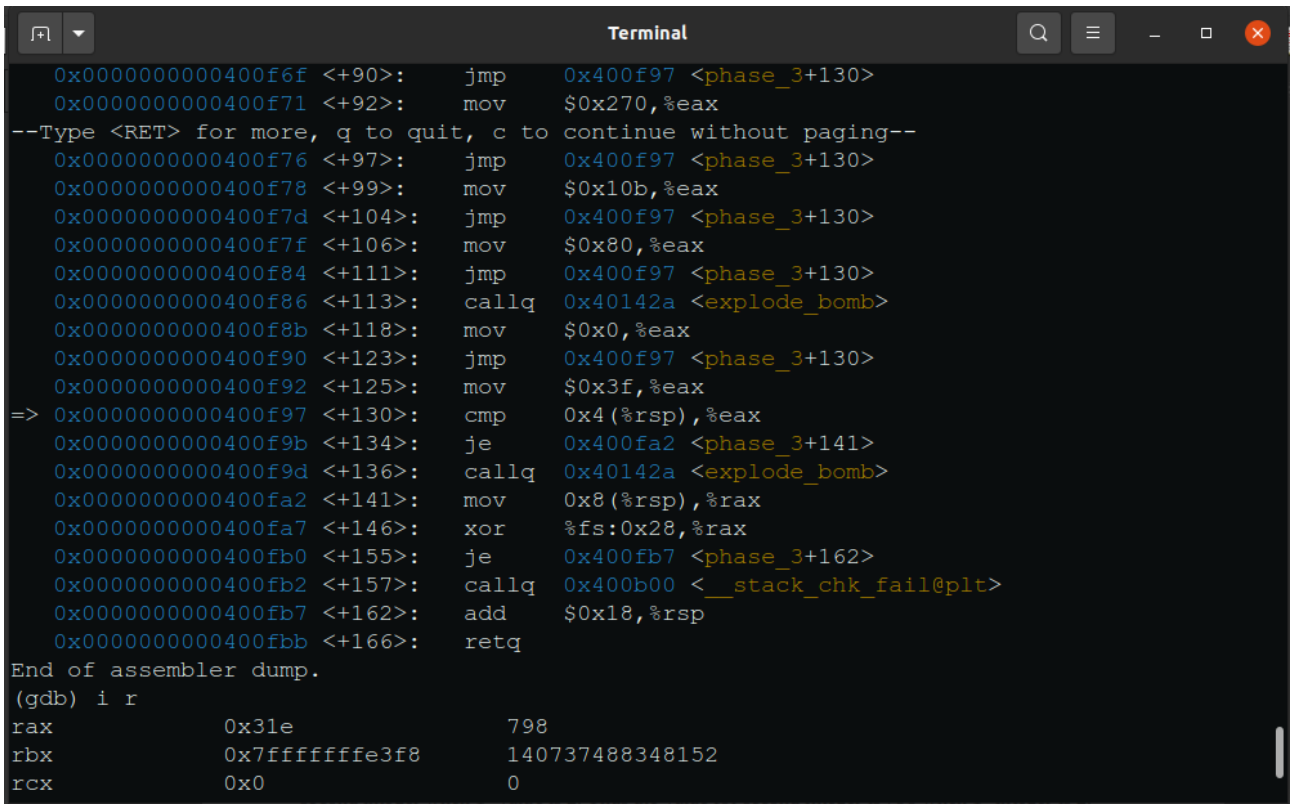


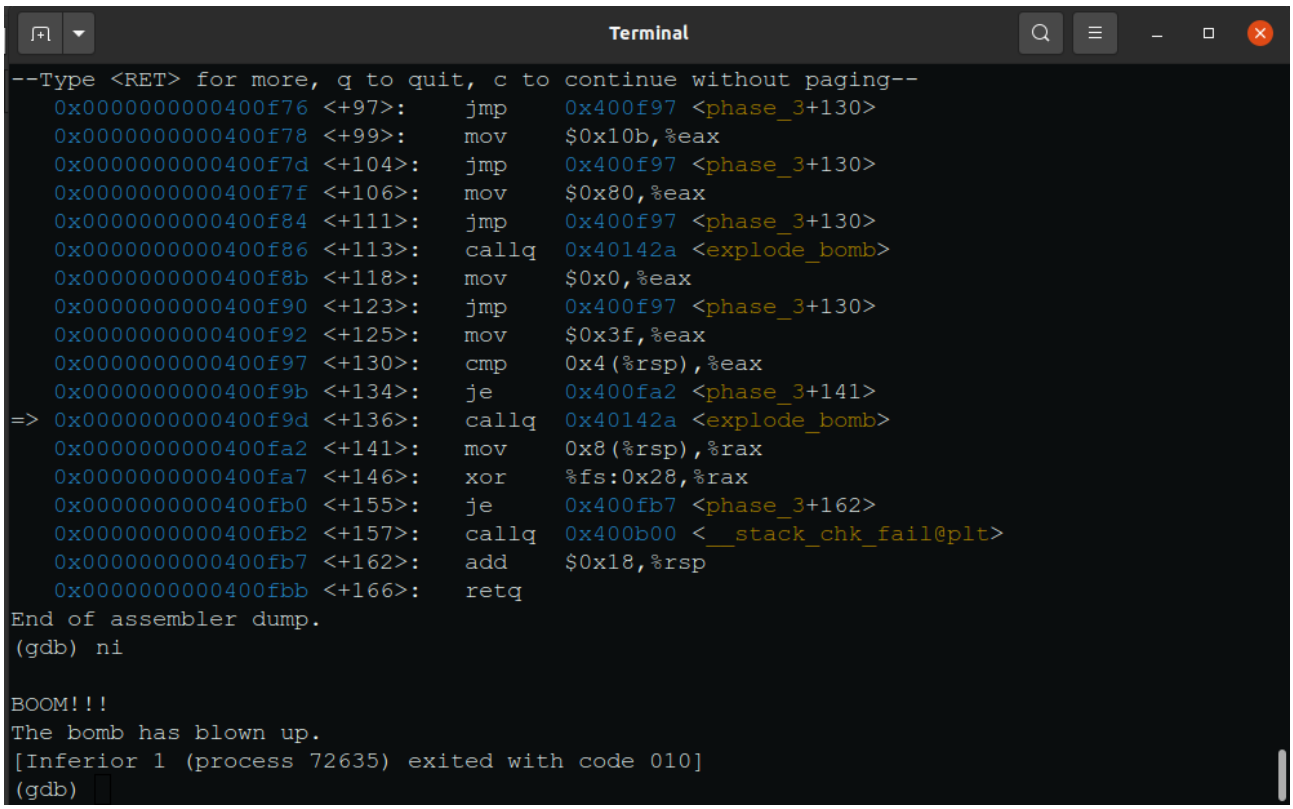


Now lets discuss about function in the program code. In phase 3 , predefined value i.e., 1 is being compare with the value of %eax i.e., 2. Followed by compare we have one condition that value of %eax should be greater than 1 and equal to 1. If this condition is not fulfilled then the bomb will get exploded. As condition happen to be true, we have move to next line i.e., 48. Again same value is compared with 7 but this the value is stored in (%rsp). As soon as it is compared, it executes next line and we have condition stating that user input value should be eqaul 7 and less than 7.

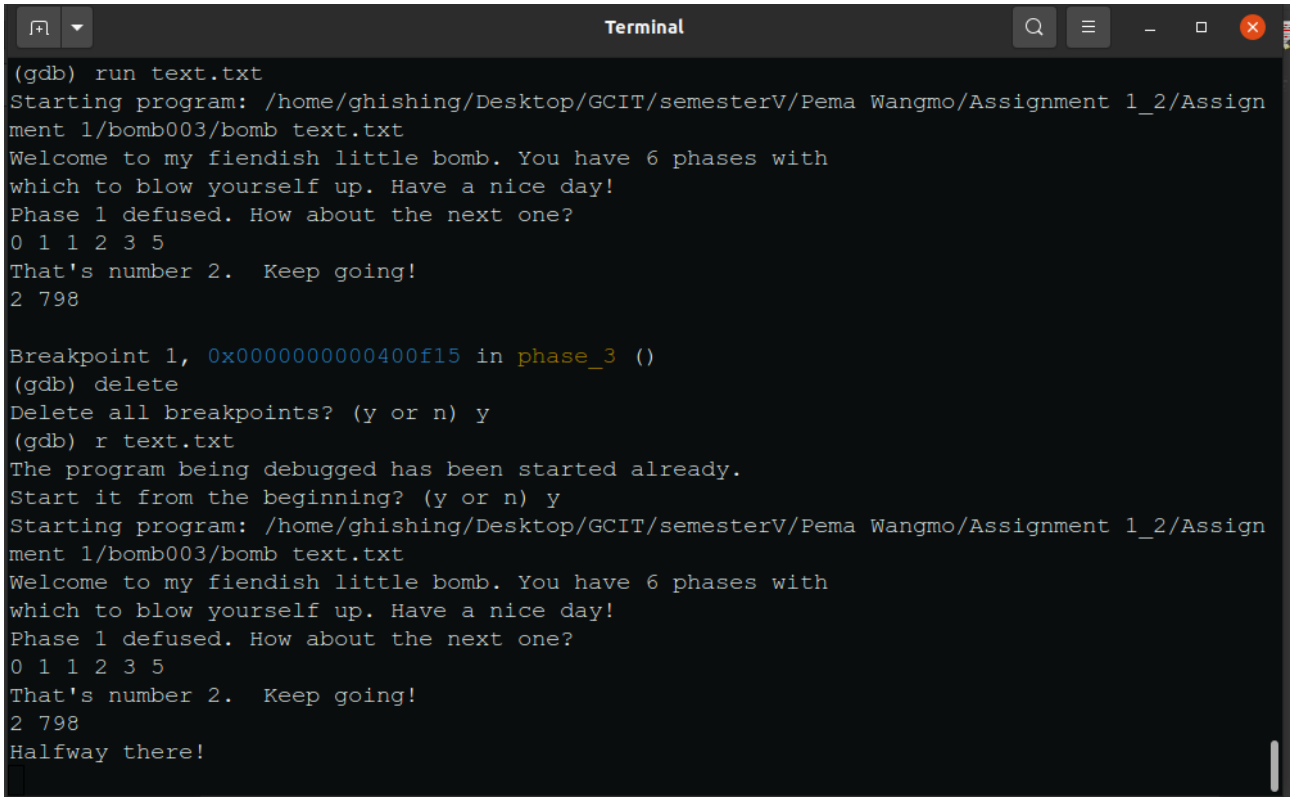


From this program code we can understand the first integer should be between 1 and 7. if it does not meet this condition then the bomb will exploded. As our condition is fulfilled, now it will execute next instruction. In next instruction program code have several operation like sub, mov, lea and then compare. After executing all the operation in program code, now second digit or integer is being compared with value of %eax, if it matches then bomb defuses else the bomb will explode. As it doesn't matches the bomb has exploded as shown below. But by this time we knew the value of second integer when 1st value is 2. Thus the value for second integer is 798 when 1st integer is 2.

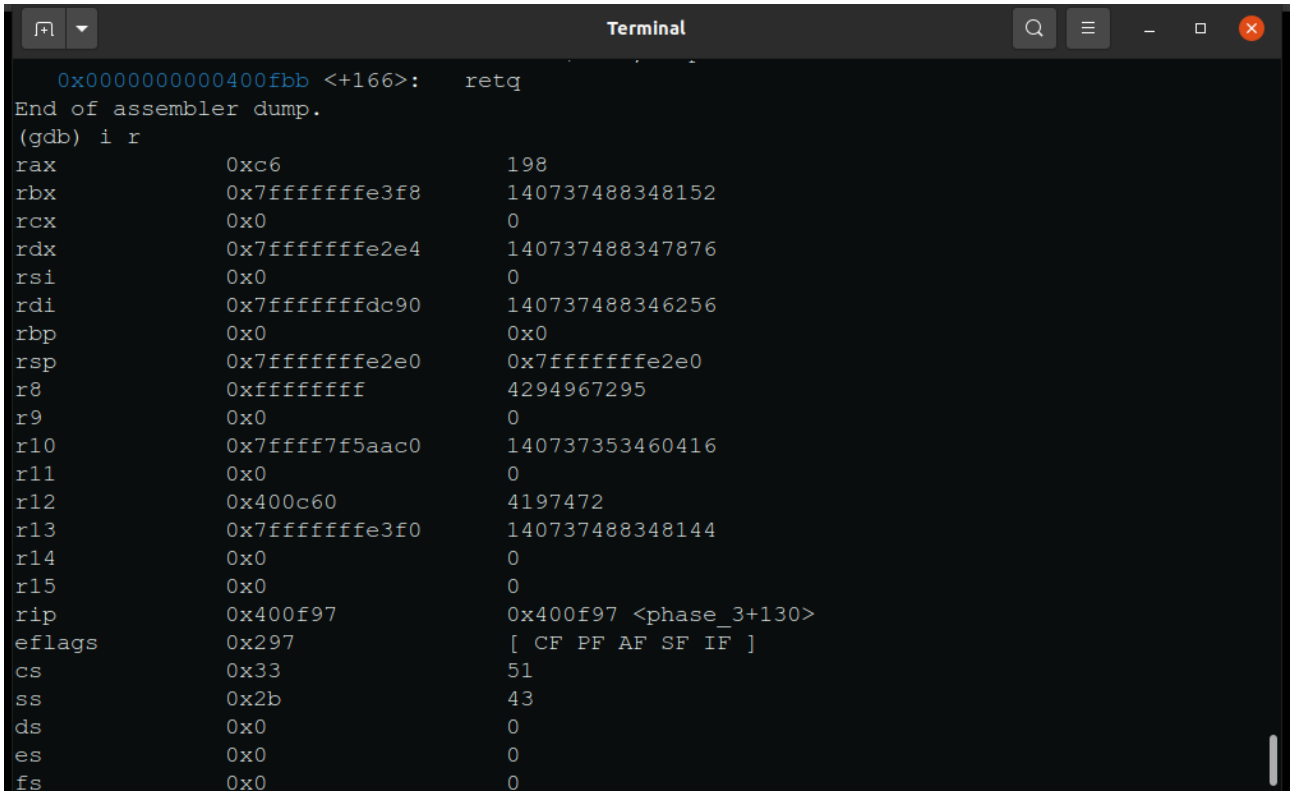

Terminal
0x0000000000400f6f <+90>: jmp 0x400f97 <phase_3+130>
0x0000000000400f71 <+92>: mov $0x270,%eax
--Type <RET> for more, q to quit, c to continue without paging--
0x0000000000400f76 <+97>: jmp 0x400f97 <phase_3+130>
0x0000000000400f78 <+99>: mov $0x10b,%eax
0x0000000000400f7d <+104>: jmp 0x400f97 <phase_3+130>
0x0000000000400f7f <+106>: mov $0x80,%eax
0x0000000000400f84 <+111>: jmp 0x400f97 <phase_3+130>
0x0000000000400f86 <+113>: callq 0x40142a <explode_bomb>
0x0000000000400f8b <+118>: mov $0x0,%eax
0x0000000000400f90 <+123>: jmp 0x400f97 <phase_3+130>
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
0x0000000000400fb0 <+155>: je 0x400fb7 <phase_3+162>
0x0000000000400fb2 <+157>: callq 0x400b00 <__stack_chk_fail@plt>
0x0000000000400fb7 <+162>: add $0x18,%rsp
0x0000000000400fbb <+166>: retq
End of assembler dump.
(gdb) i r
rax 0x31e 798
rbx 0x7fffffffe3f8 140737488348152
rcx 0x0 0


Terminal
--Type <RET> for more, q to quit, c to continue without paging--
0x0000000000400f76 <+97>: jmp 0x400f97 <phase_3+130>
0x0000000000400f78 <+99>: mov $0x10b,%eax
0x0000000000400f7d <+104>: jmp 0x400f97 <phase_3+130>
0x0000000000400f7f <+106>: mov $0x80,%eax
0x0000000000400f84 <+111>: jmp 0x400f97 <phase_3+130>
0x0000000000400f86 <+113>: callq 0x40142a <explode_bomb>
0x0000000000400f8b <+118>: mov $0x0,%eax
0x0000000000400f90 <+123>: jmp 0x400f97 <phase_3+130>
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
0x0000000000400fb0 <+155>: je 0x400fb7 <phase_3+162>
0x0000000000400fb2 <+157>: callq 0x400b00 <__stack_chk_fail@plt>
0x0000000000400fb7 <+162>: add $0x18,%rsp
0x0000000000400fbb <+166>: retq
End of assembler dump.
(gdb) ni
BOOM!!!
The bomb has blown up.
[Inferior 1 (process 72635) exited with code 010]
(gdb)

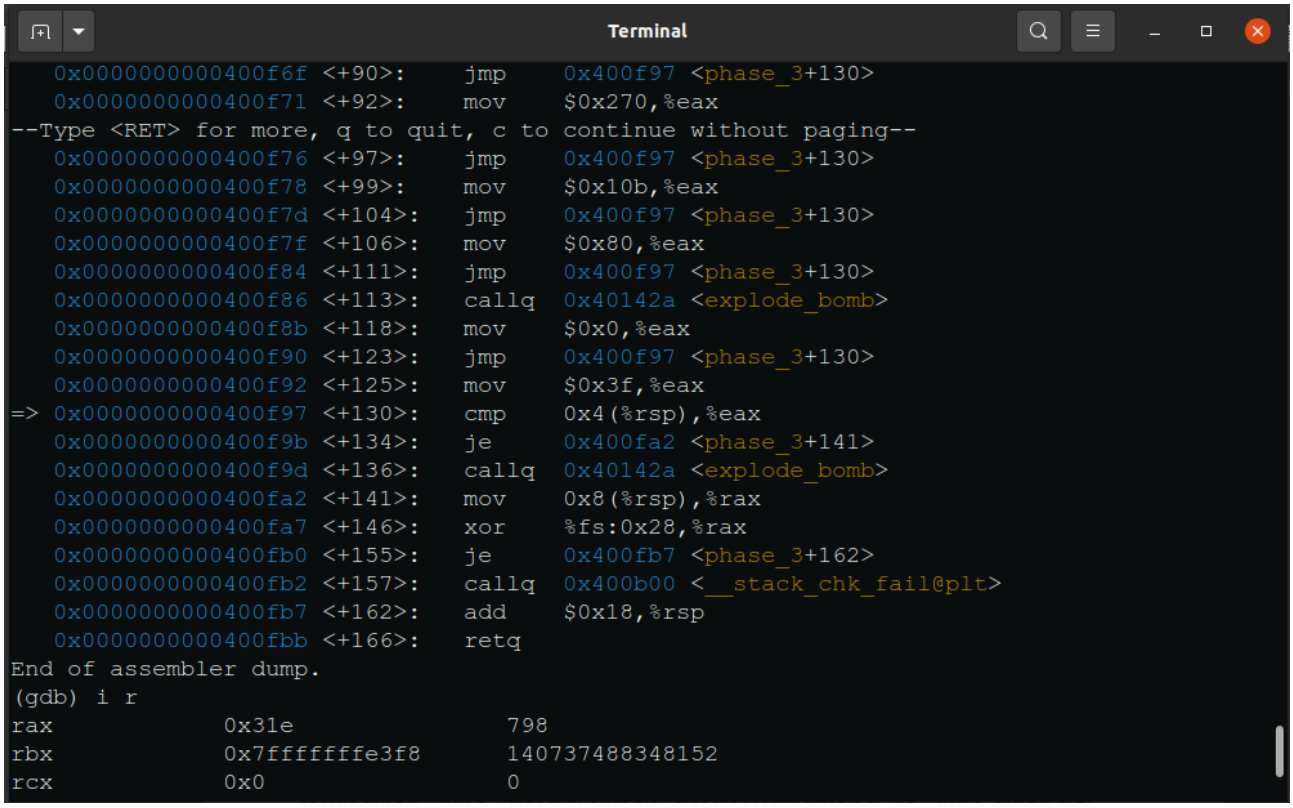
Finally the program code was restarted and the bomb of phase 3 was defused as shown below.



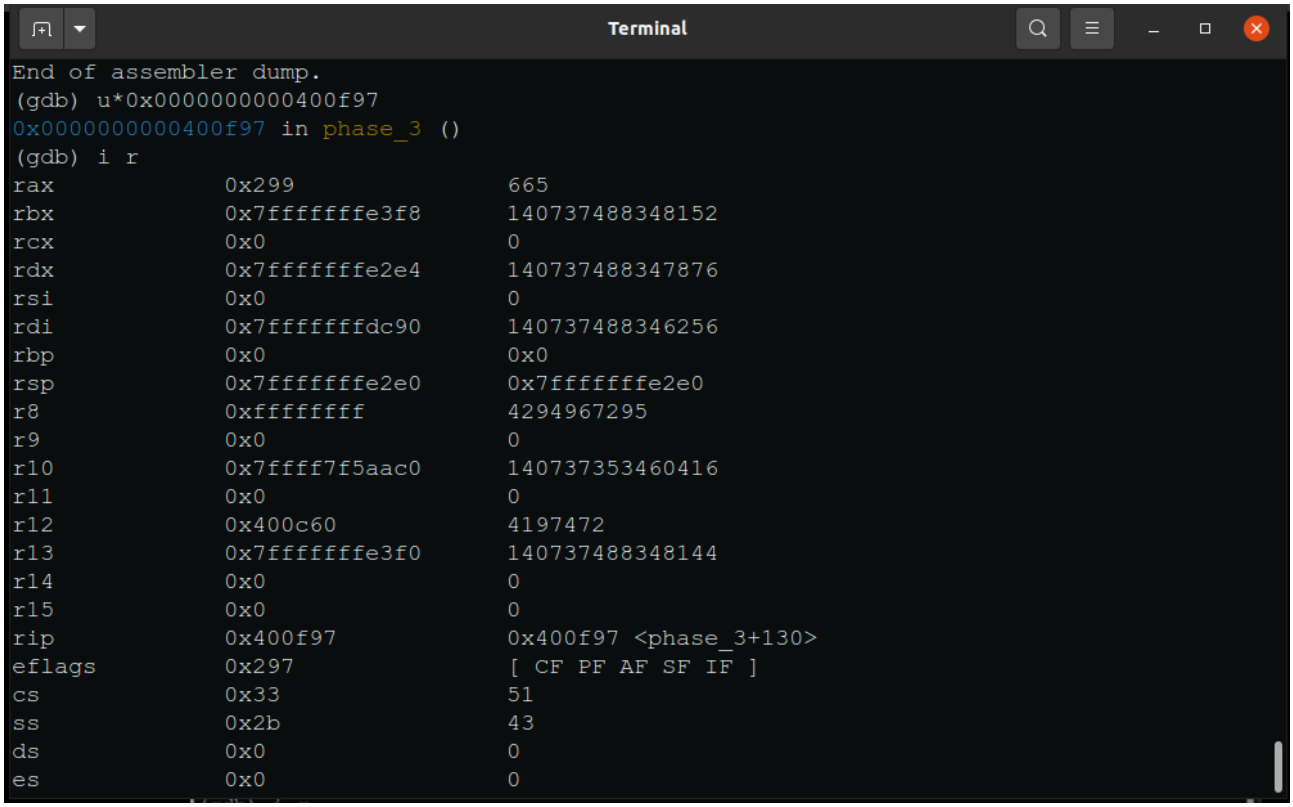
After understanding the condition of program code, we have learned that first integer ranges from 1 to 7. Thus it should have its own respective integer as a second value. After executing the program code and providing input ranging from 1 to 7, respective second value is extracted as follows 1 & 198, 2&798, 3&665, 4&58, 5 &624, 6&267 and 7 &128.
For 1st integer 1:



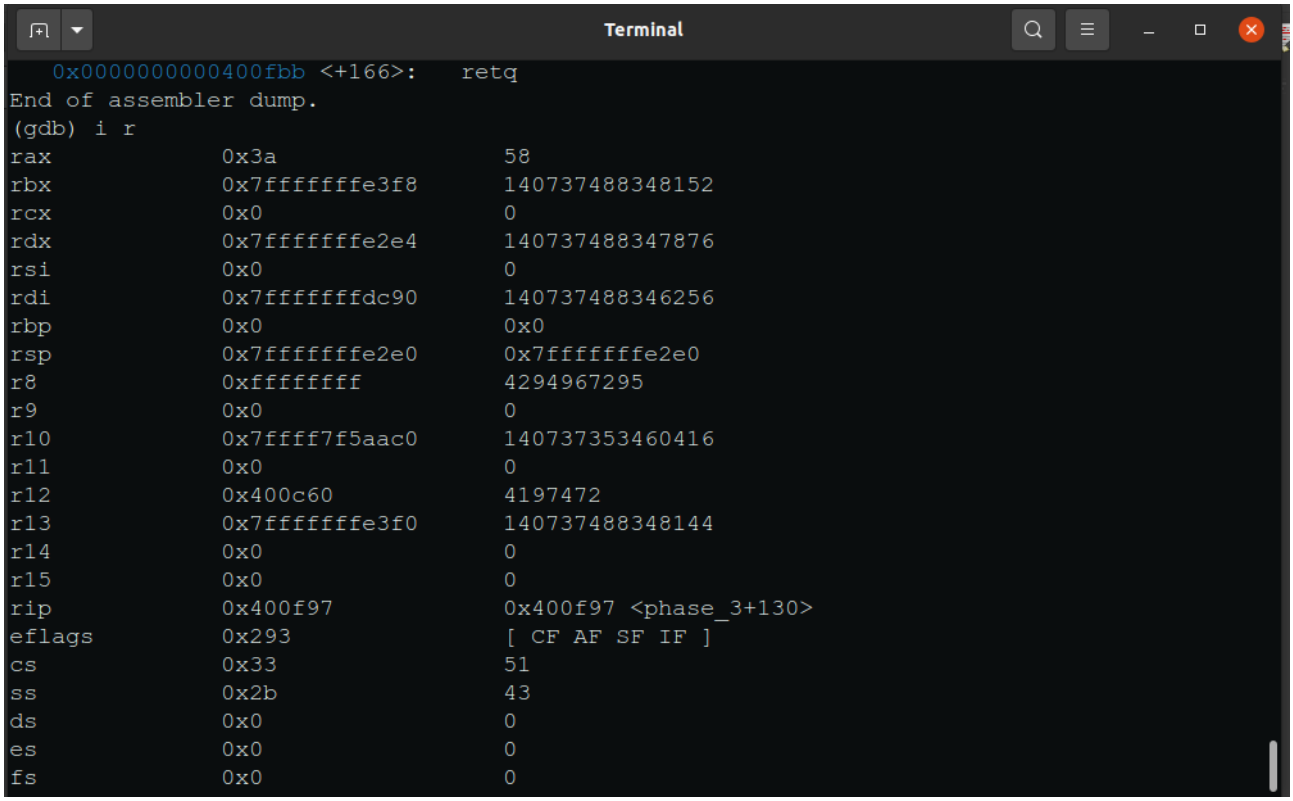
For 2nd integer 2 :



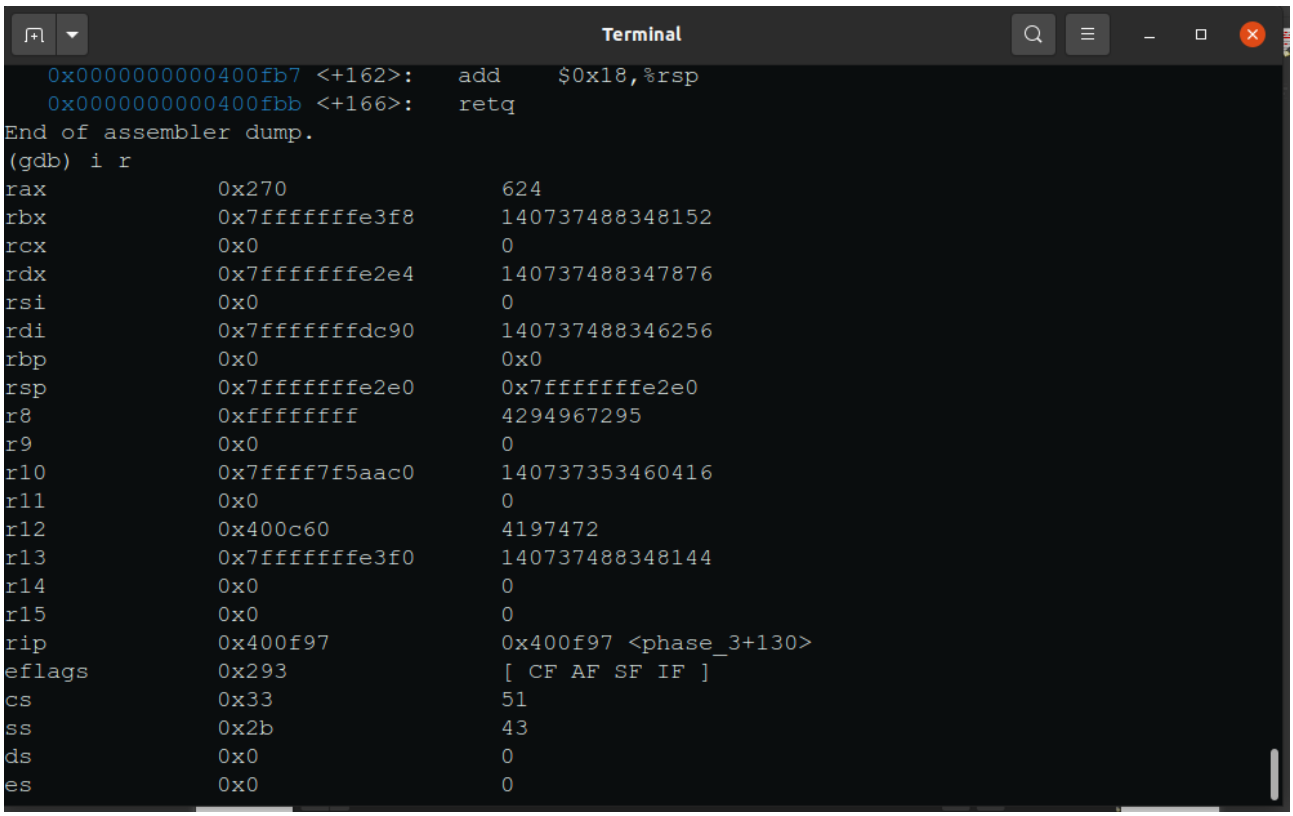
For 3rd integer 3:



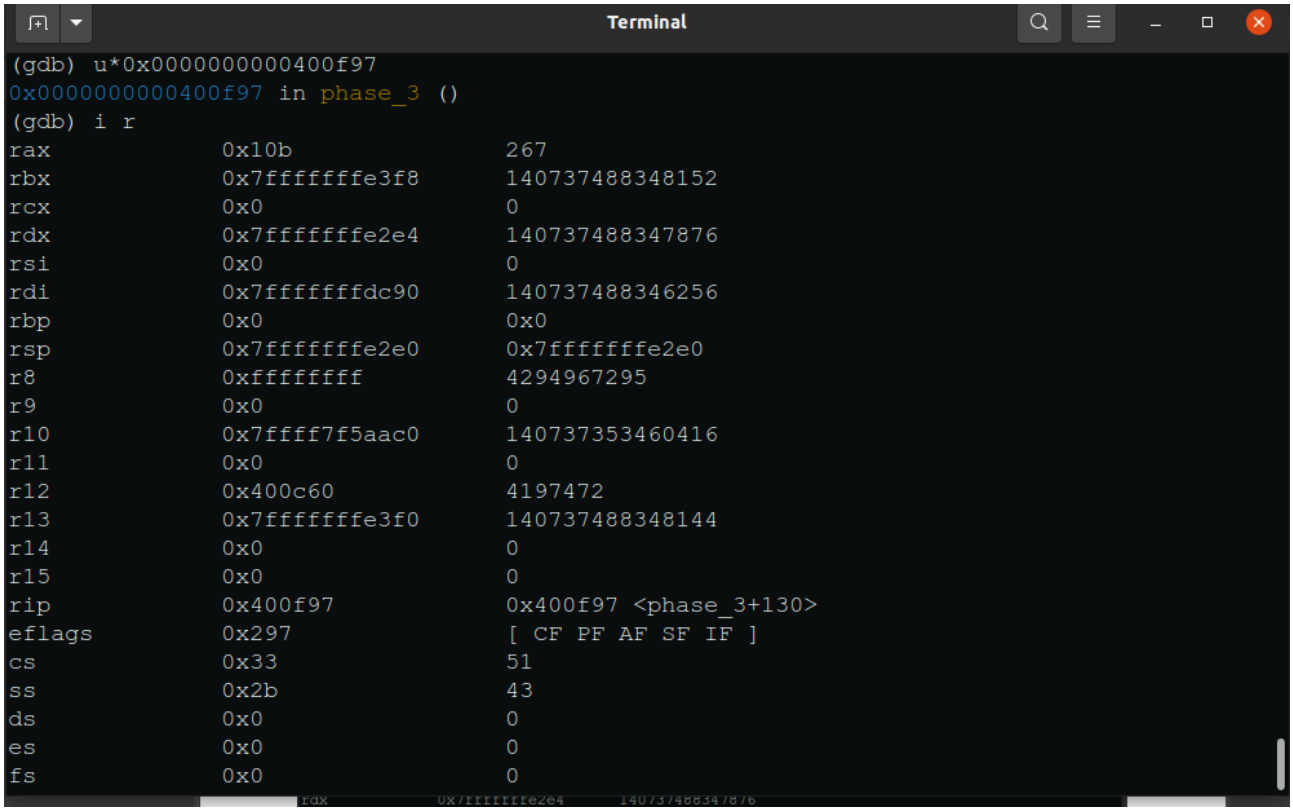
For 4th integer 4:



For 5th integer 5:



For 6th integer 6:



For 7th integer 7: