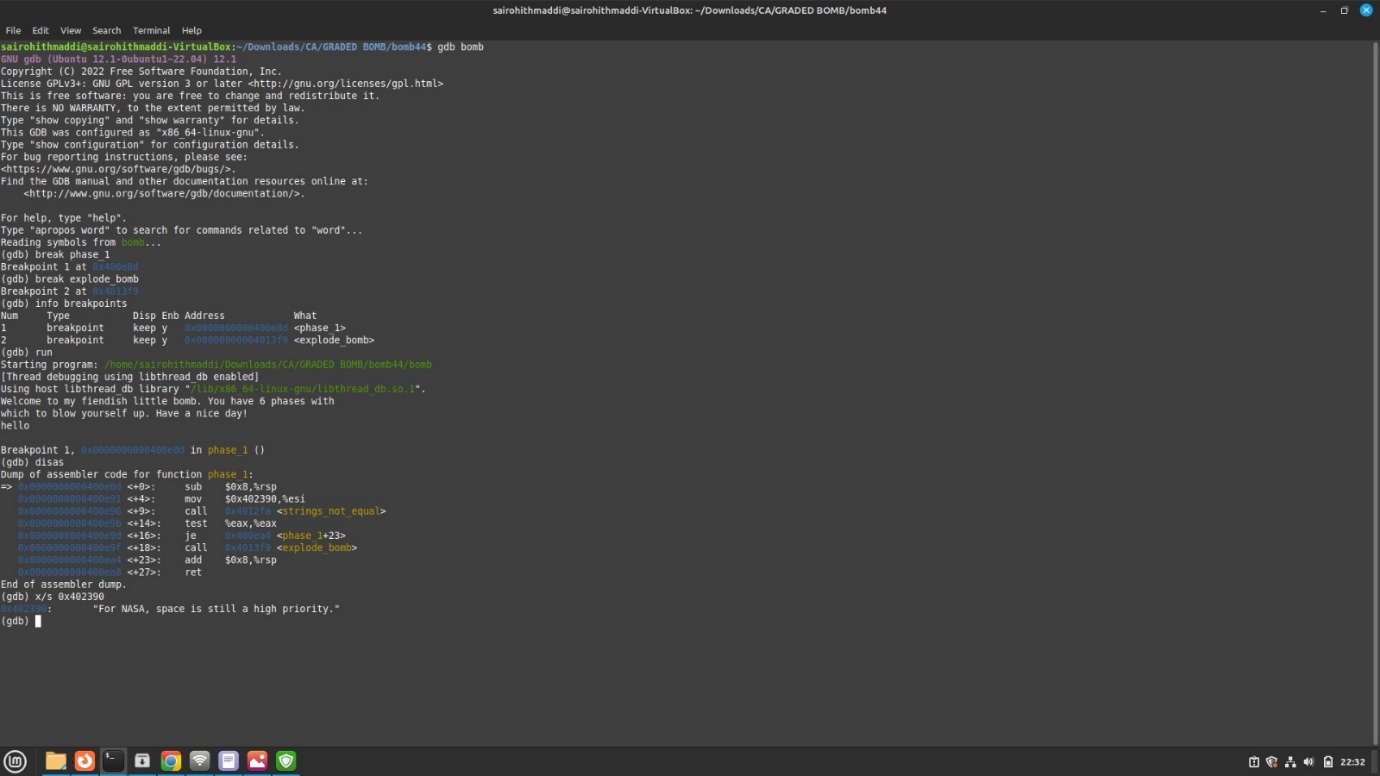
|  |
| --- |
| CA LAB REPORT -1 |
| BOMB LAB-1 |
| -S20220010130 |

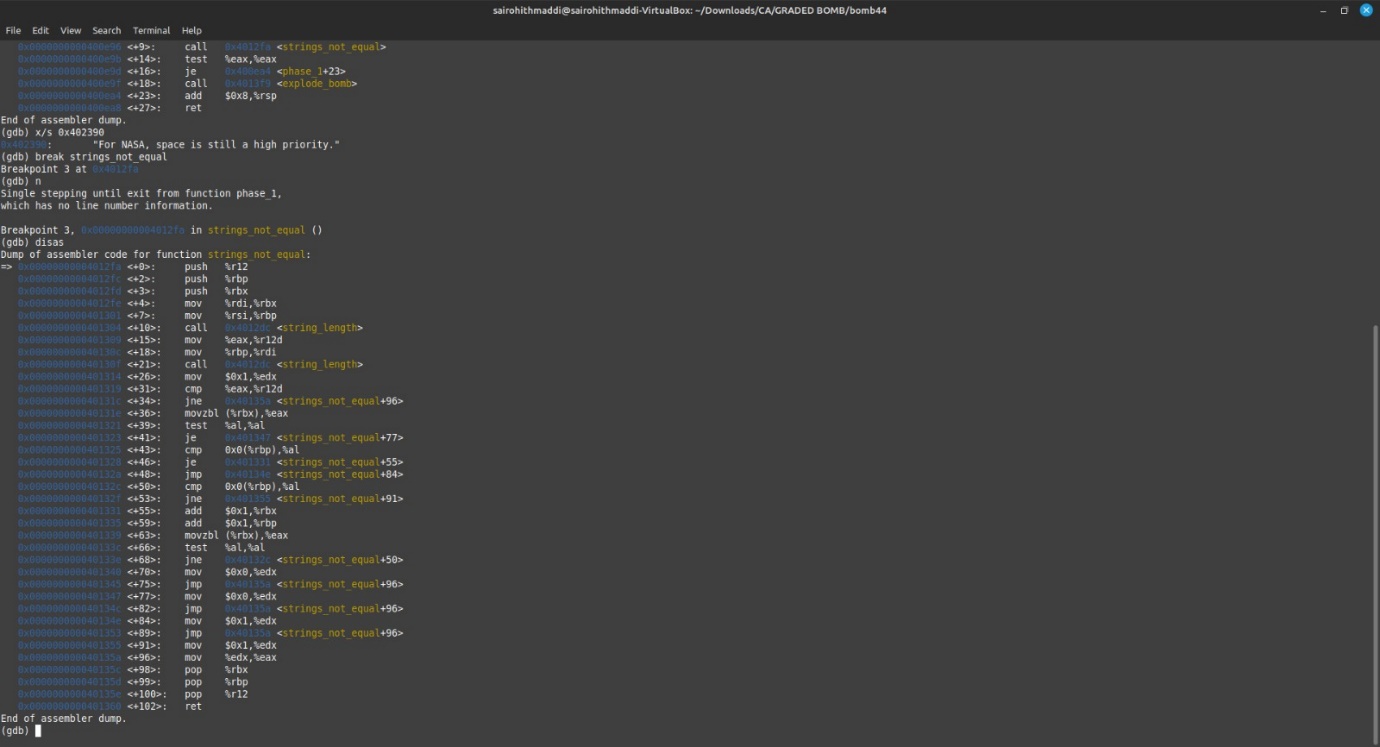
phase 1:-

we can see that we storing something in esi



which is what we are going to be used to compare the answer string

that is in the strings not equal function so our answer must be the value that we have assigned to esi



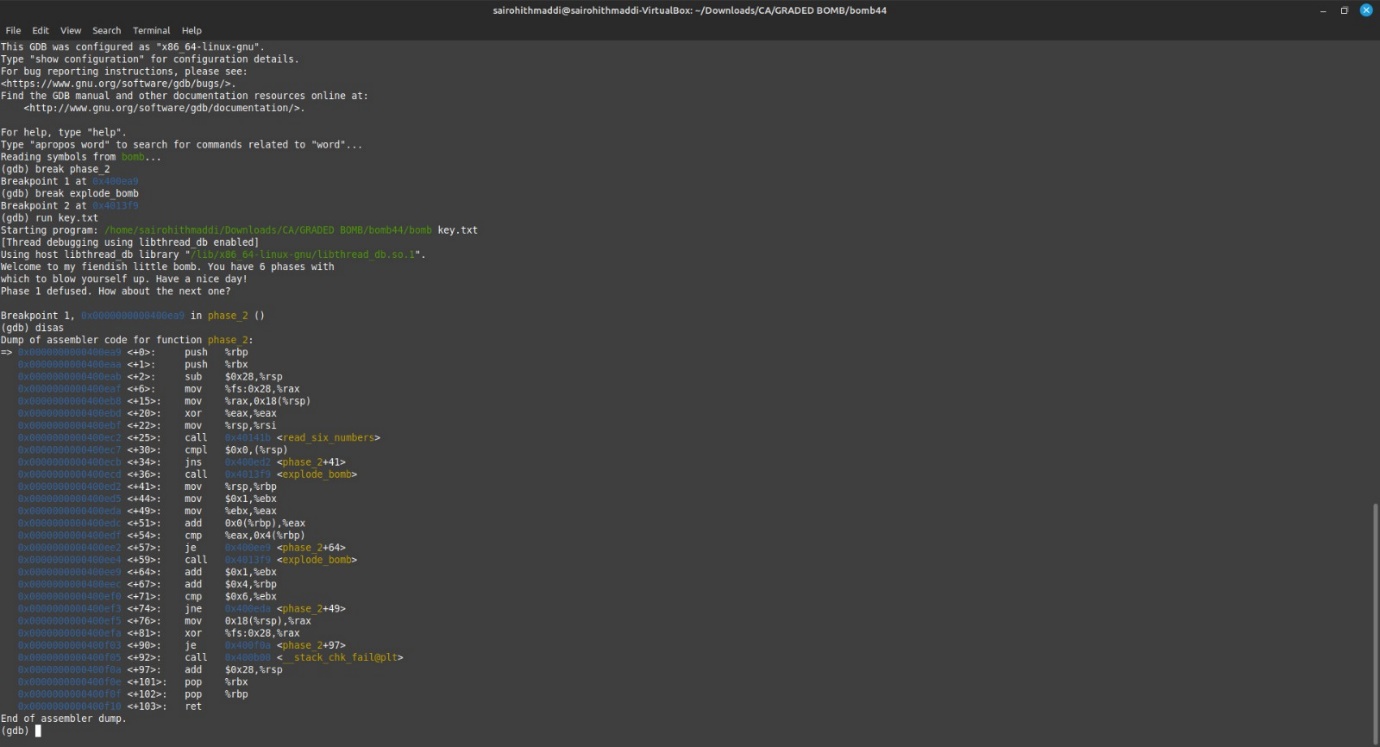
so identifying the value in the 0x402390 gives a string and it will also be the answer for our question

x/s command give the string present in address

x/s 0x402390 gives us the answer or the required string

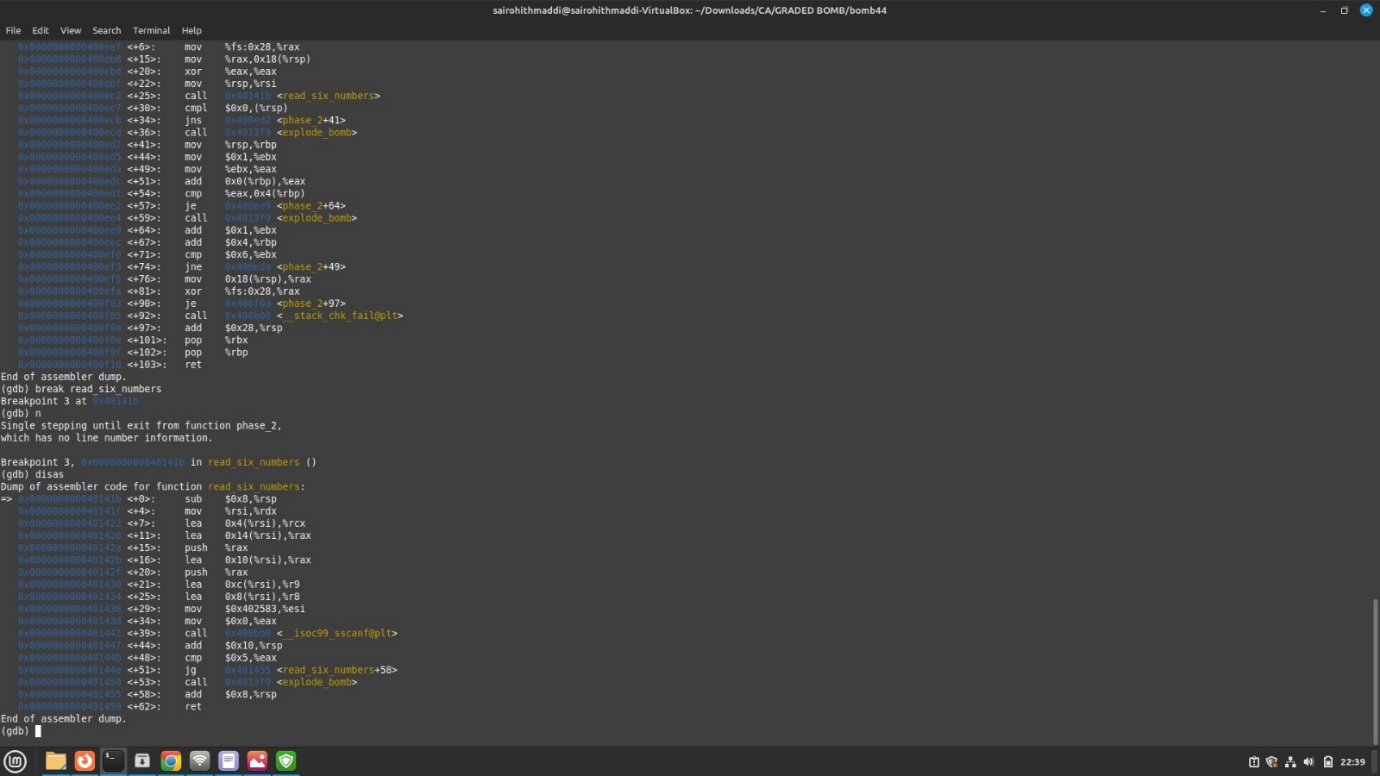
Phase :-2

first comparing 0 with rsp and then we are moving to 0x400ed2.



storing 1 in the register ebx and moving ebx to eax then after comparsion we are adding 1 to ebx and increasing the rbp

finally comparing the no of inputs to be entered and jumping to 0x400eda again just like a, loop where ebx is the index being incremented by 1 everytime for a total of six times .



so for first time in eax it stored 1(intialy 1 and in the loop we are adding 1) + 1 =2

and next time it would be 2 + 1 ( second time in the loop)=3

next time 3+2 (third time in loop)

but it runs until 5 not 6

in this way 1 2 3 4 5 are being added to to the find the next adfacentn number so the ordr would be

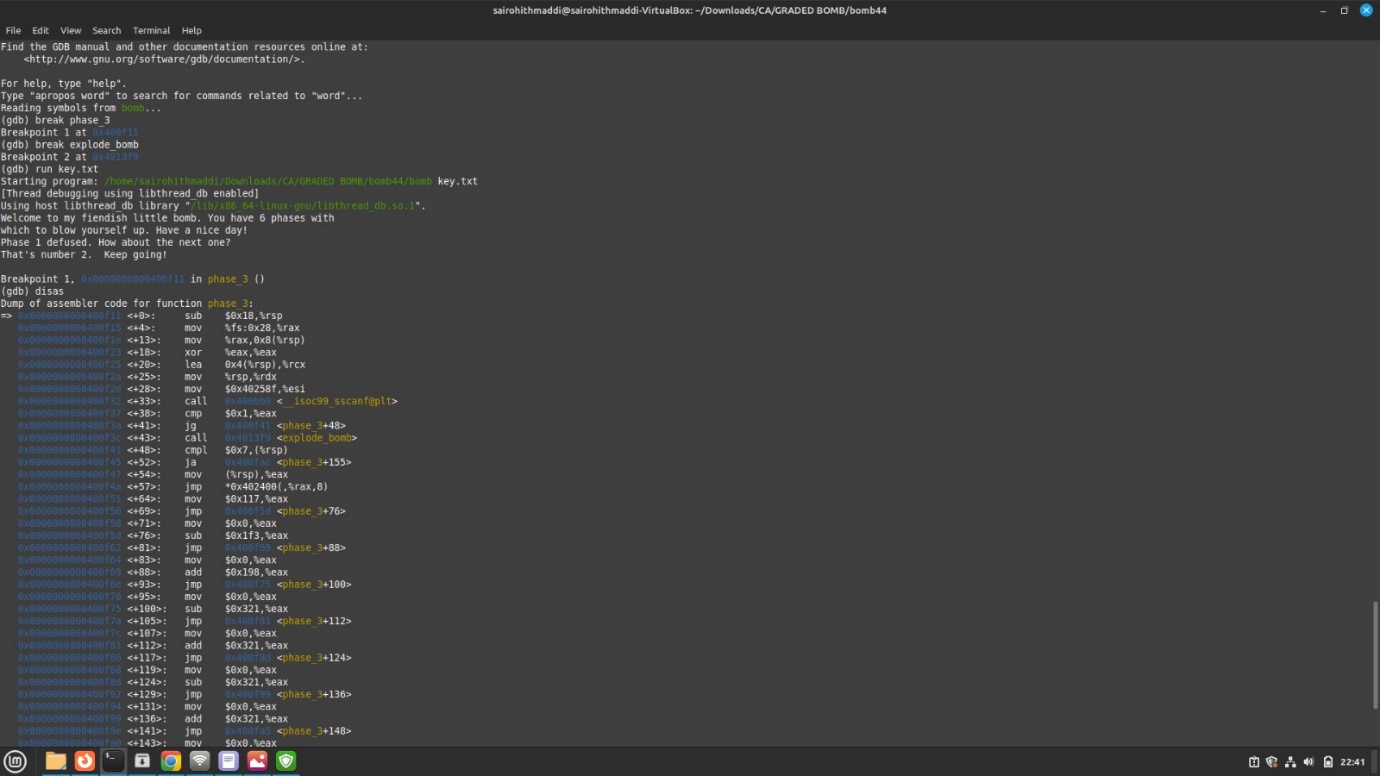
1 2 4 7 11 16.

Phase:- 3

we can see that we are checking the value of rsp with 7 if it exceeds it then it directs control to bomb , hence our value should be less than 7

on assuming the value to be 0 .

so our rsp value will become 0



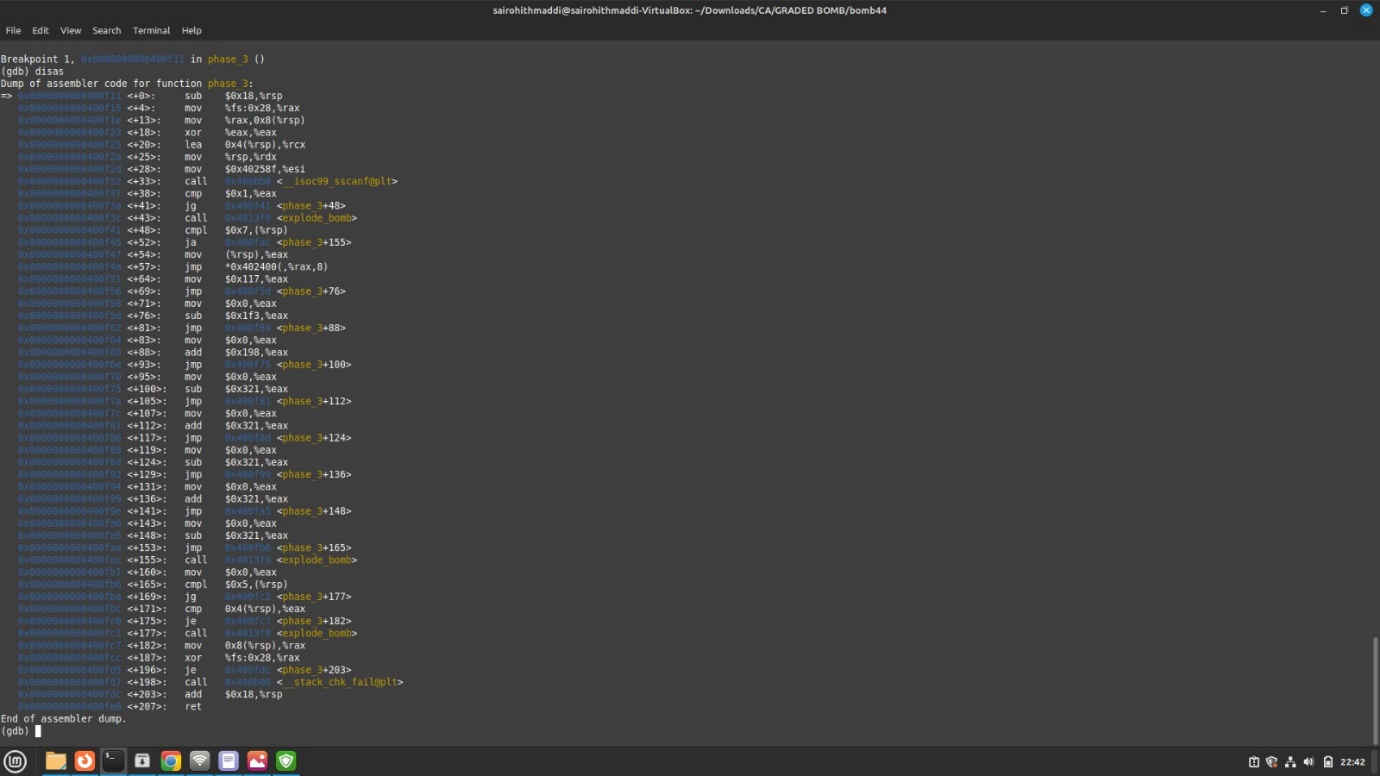
now in the line 0x400f4a we can see there is jump

eax has the value of rsp now

jump is to 0x402400+ 8\* rax here rax=0 for us so jump is to the line 0x400f51

eax value is 117 (279 in decimal)is stored

and then there are series of jump statements so the eax values keep on updating



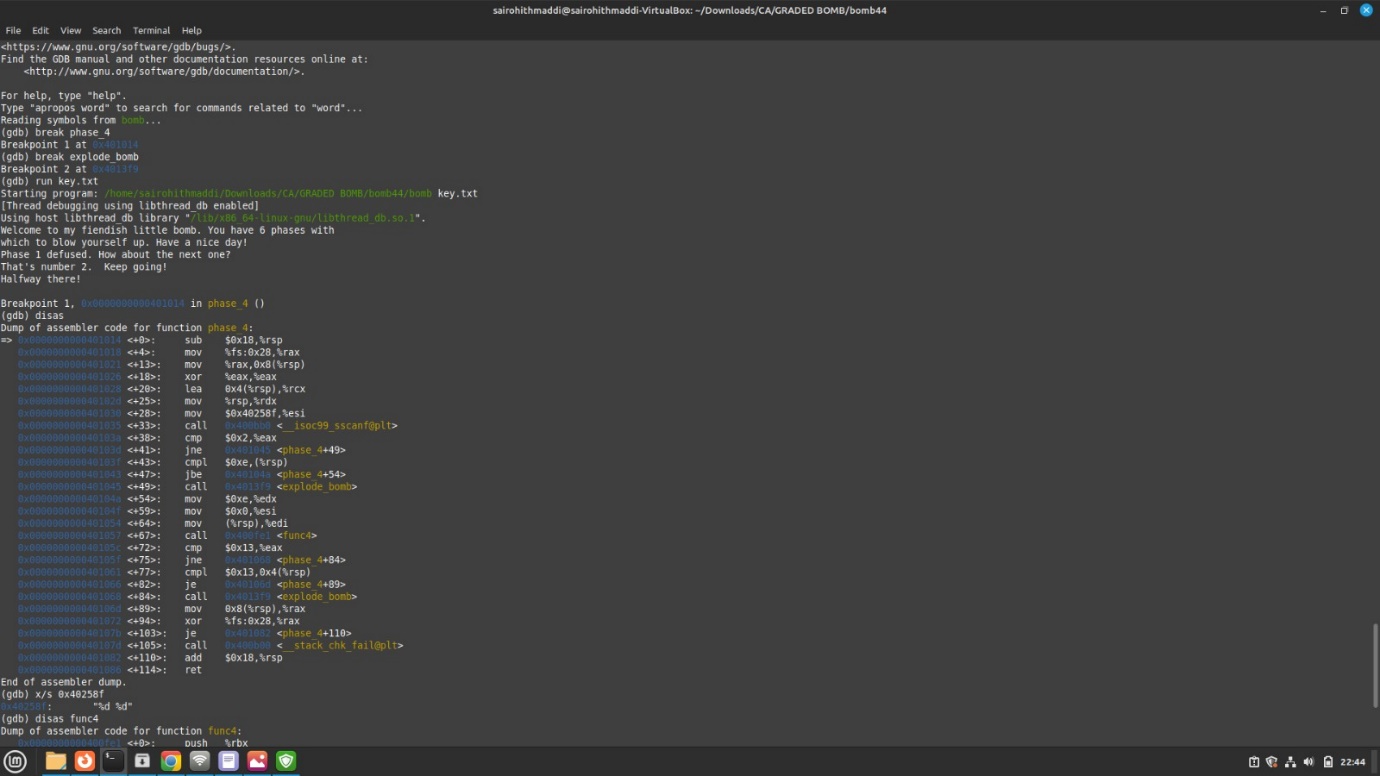
eax = 117-1f3+198-321+321-321+321-321 = -277 in hexa decimal and in decimal it is equla to -631

so finally eax will be -631 in decimal so

our answer should be 0 -631

phase:-4

The assembly code for the phase\_4 function performs the following steps:

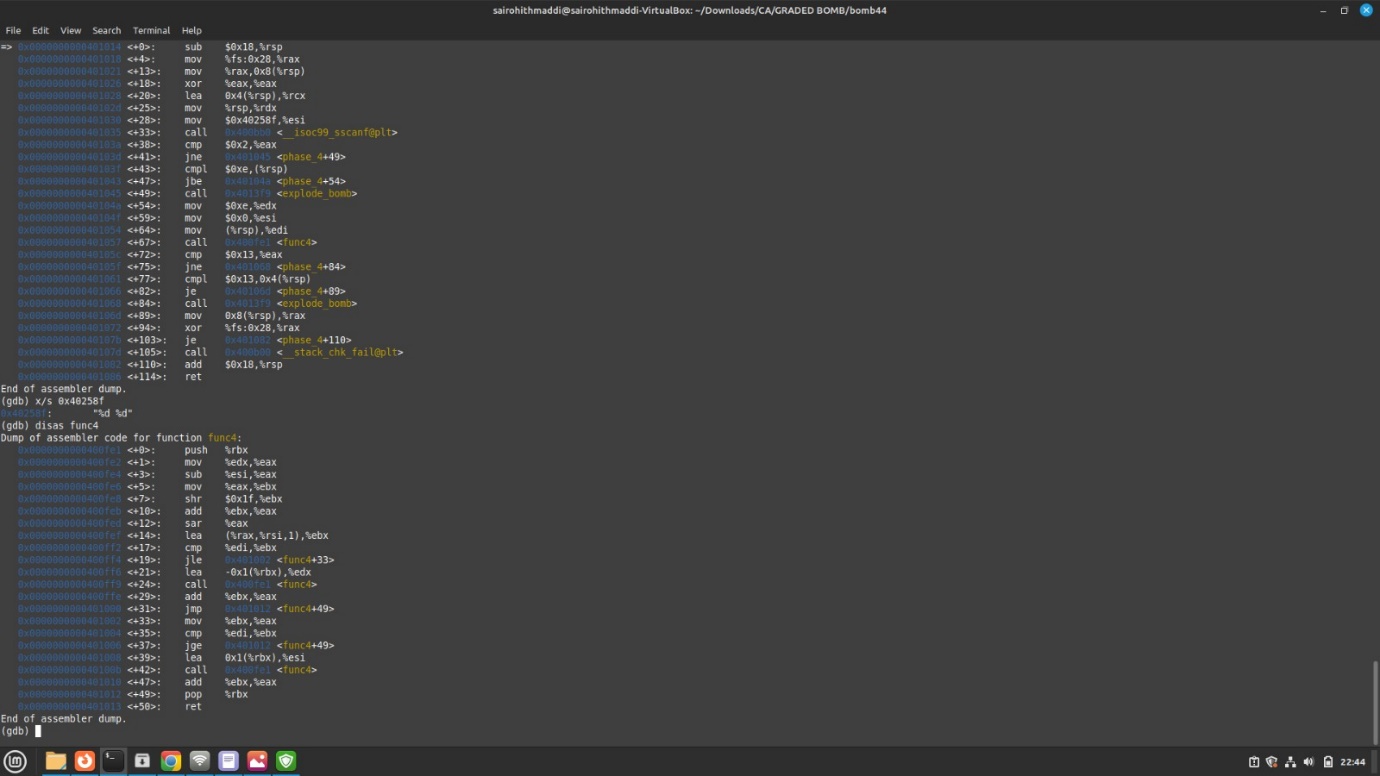


Reads two integers from a formatted string using sscanf.

Checks if the sscanf call successfully read two values.

Compares the first value with 14 and explodes the bomb if it's greater.

Calls func4 with arguments and checks the result.



Compares the result of func4 with 19 and the second value with 19.

Explodes the bomb if the comparisons fail.

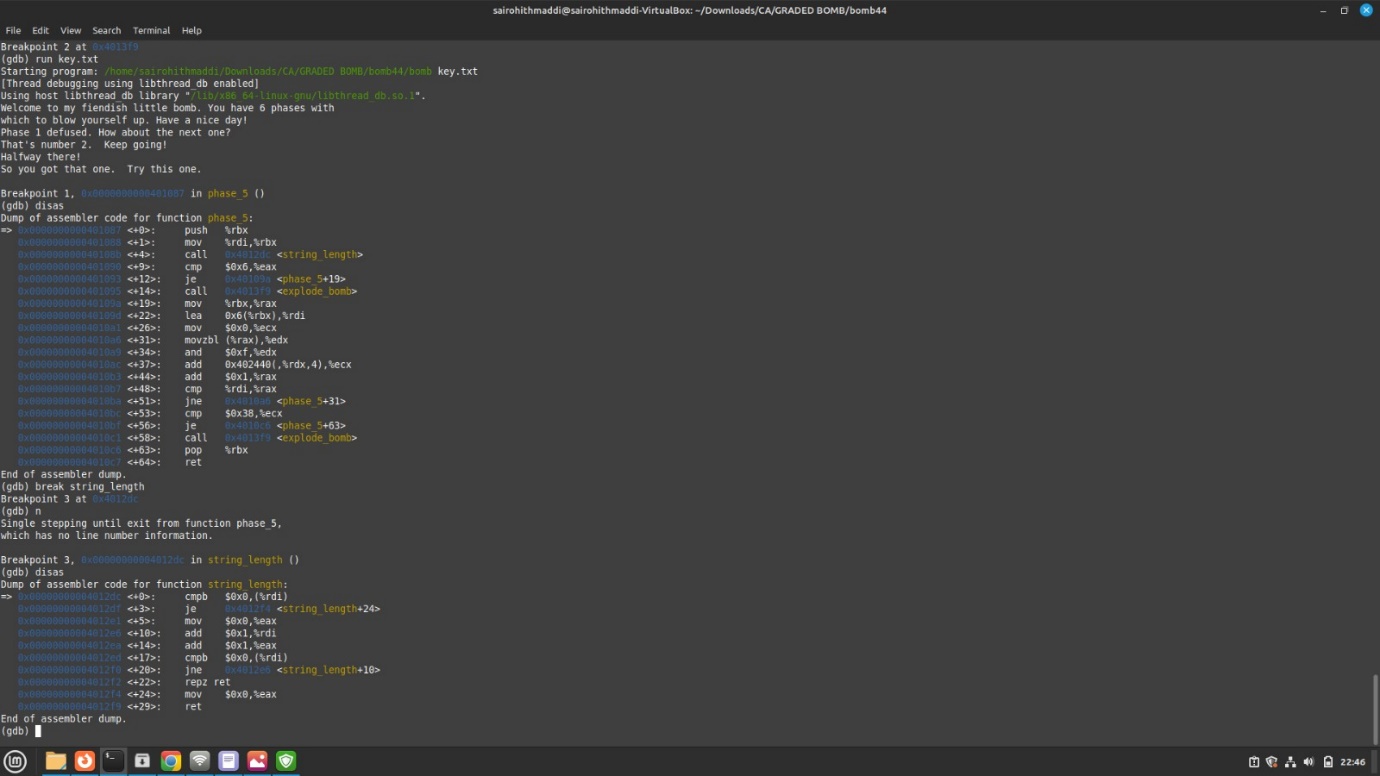
Handles stack overflow protection.

Returns from the function.

In summary, the function reads two integers, performs checks and comparisons on the values, and calls func4 before making additional comparisons and handling stack protection.

Phase:-5

Certainly! The code you provided is a loop that goes through each character in a word or sentence.



For each character, the code looks at the last four bits of that character .It takes those four bits and uses them as a number to look up another value in a table.

Then, it adds that value to a running total.

The code then moves on to the next character in the word or sentence and repeats the process.

Once it has gone through all the characters, it stops and does something with the final total.

Phase:-6

1.It sets up the stack by pushing registers (%r13, %r12, %rbp, %rbx) and subtracting 0x68 from %rsp to allocate space.

2.It stores the value of %fs:0x28 into %rax to set up stack overflow protection.

3.It calls a function called read\_six\_numbers to read six integers from input into memory starting from %rsp.

4.It initializes some registers and begins a loop.

5.Inside the loop, it compares the value at the current memory location with the value in %eax.

6.If the comparison fails, it calls explode\_bomb to explode the bomb.

7.It increments the loop counter and checks if it has reached 6. If not, it continues the loop.

8.If the loop is finished, it performs some memory operations and additional checks.

9.It handles stack overflow protection by comparing %fs:0x28 with the previously saved %rax.

10.It restores the stack by adjusting %rsp and pops the registers from the stack.

11.Finally, it returns from the function.

In summary, the function reads six numbers, compares them with specific values, performs memory operations, handles stack overflow protection, and returns.

