Sheldon Phase 3- Writeup

Objective of this report is to introduce the used approach to find the passphrase for the phase_3 of the Sheldon_1 binary file.

Since we already know how the program works, let's start with disassembling the phase_3 to analyze its functionality. Also, notice that the 'b phase_3' command was used to set a breakpoint at phase_3 from the beginning.

This phase can take multiple combination of values as inputs. Disassembled code implied that this might include a case-based scenario to check the input values.

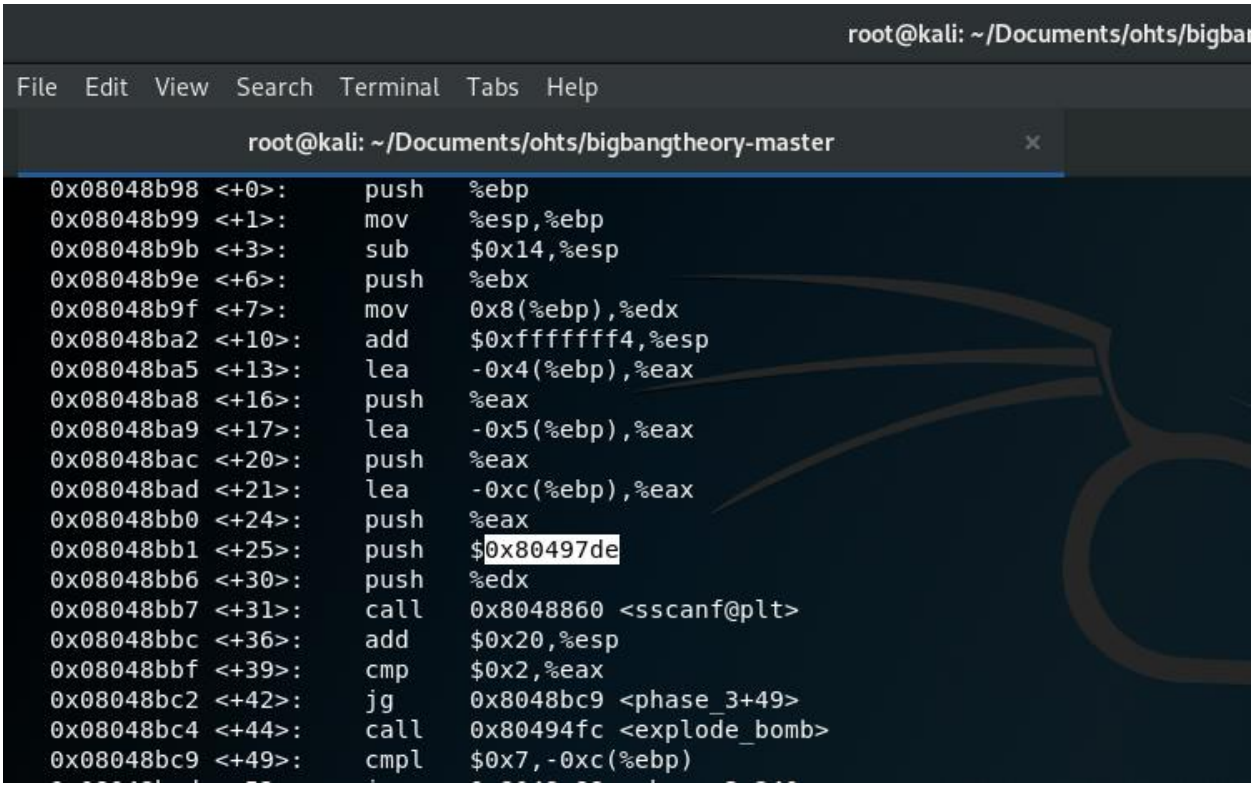

Figure 1: Disassembled phase_3 function

Prior to the call of scanf function, there's a value getting pushed to the stack at the line 25. That looks like an address we must check before executing anything [Figure 1].

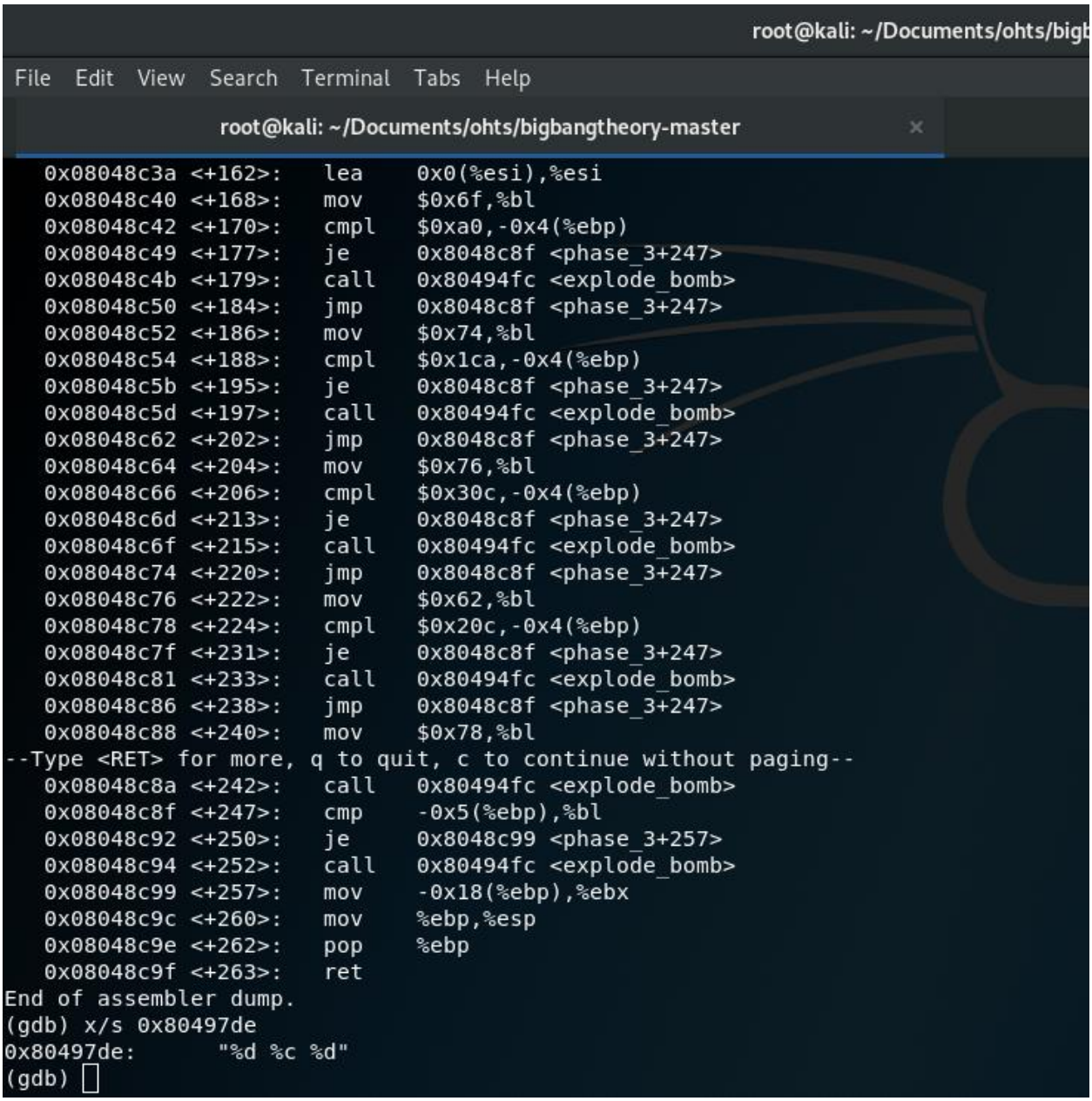

Figure 2: Identification of the input pattern

By checking the address (0x80497de), we can see the input pattern of the phase_3. Input is limited to two integers and for a one character [Figure 2]. Also, it is possible to see the values getting pushed to their respective locations based on their data type sizes at the lines 13,17 and 21 [Figure 3].

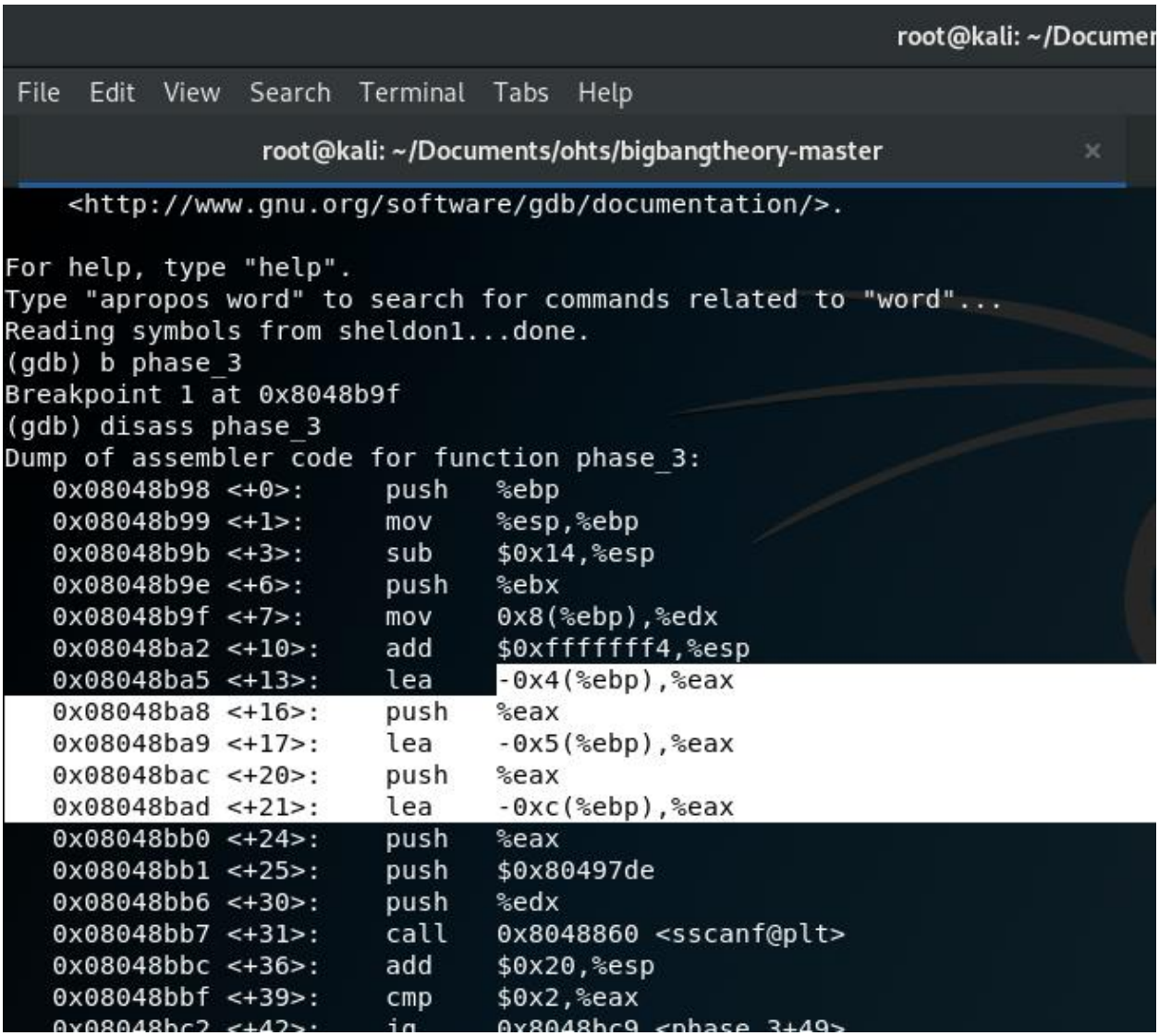

Figure 3 : Values and their positions

At the line 39 we can see a condition related to eax register value and for the value 2 [Figure 4].

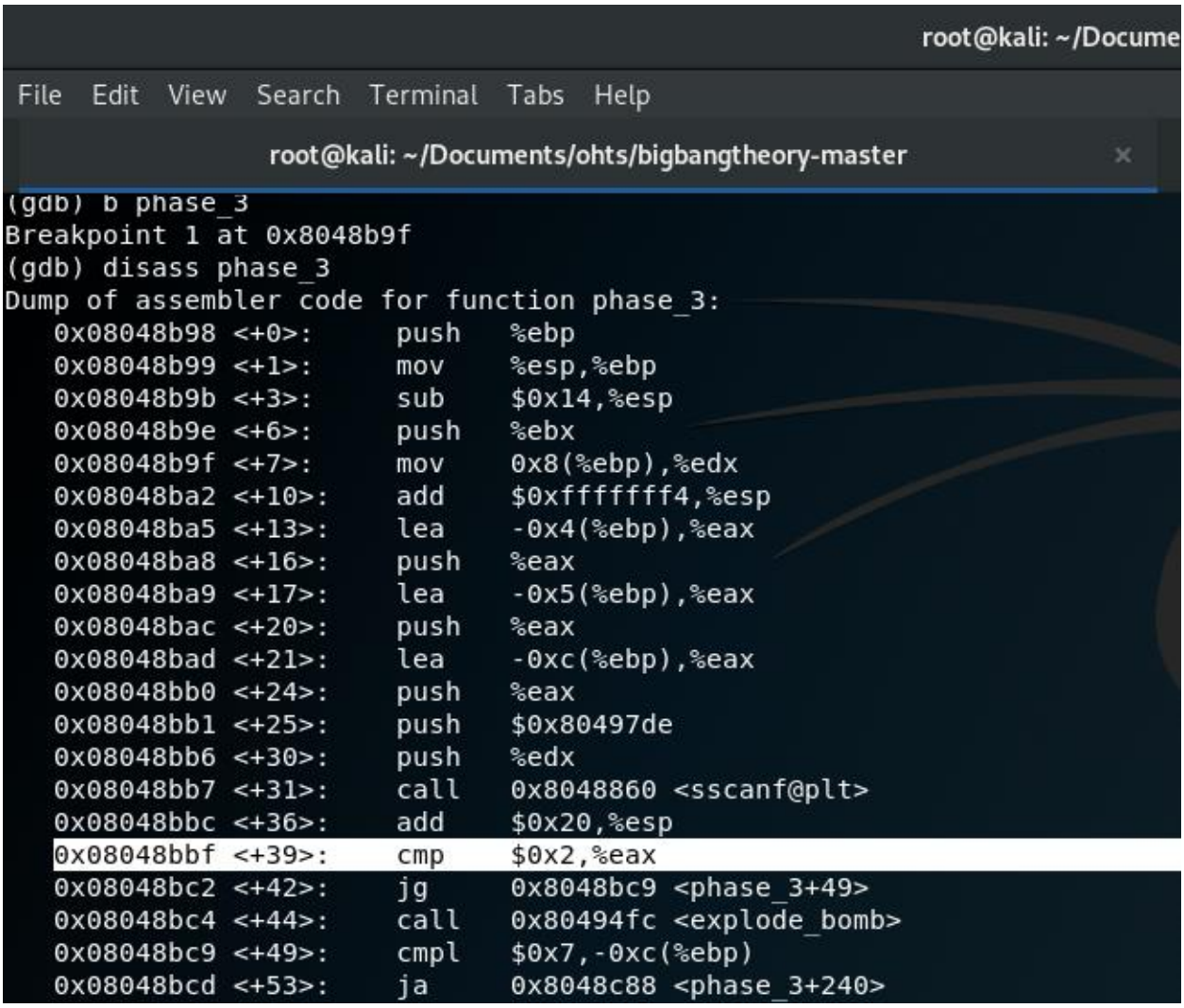

Figure 4 : Clue to start with

Also, we can see there's another condition with the first position of the input and if the input value is greater than 7 the bomb will explode (insert the first integer value as 8 and use ni, i r and disas phase_3 commands to follow the bomb execution behavior). These conditions guaranteed that the first integer value should be a value under number 7 [Figure 5].

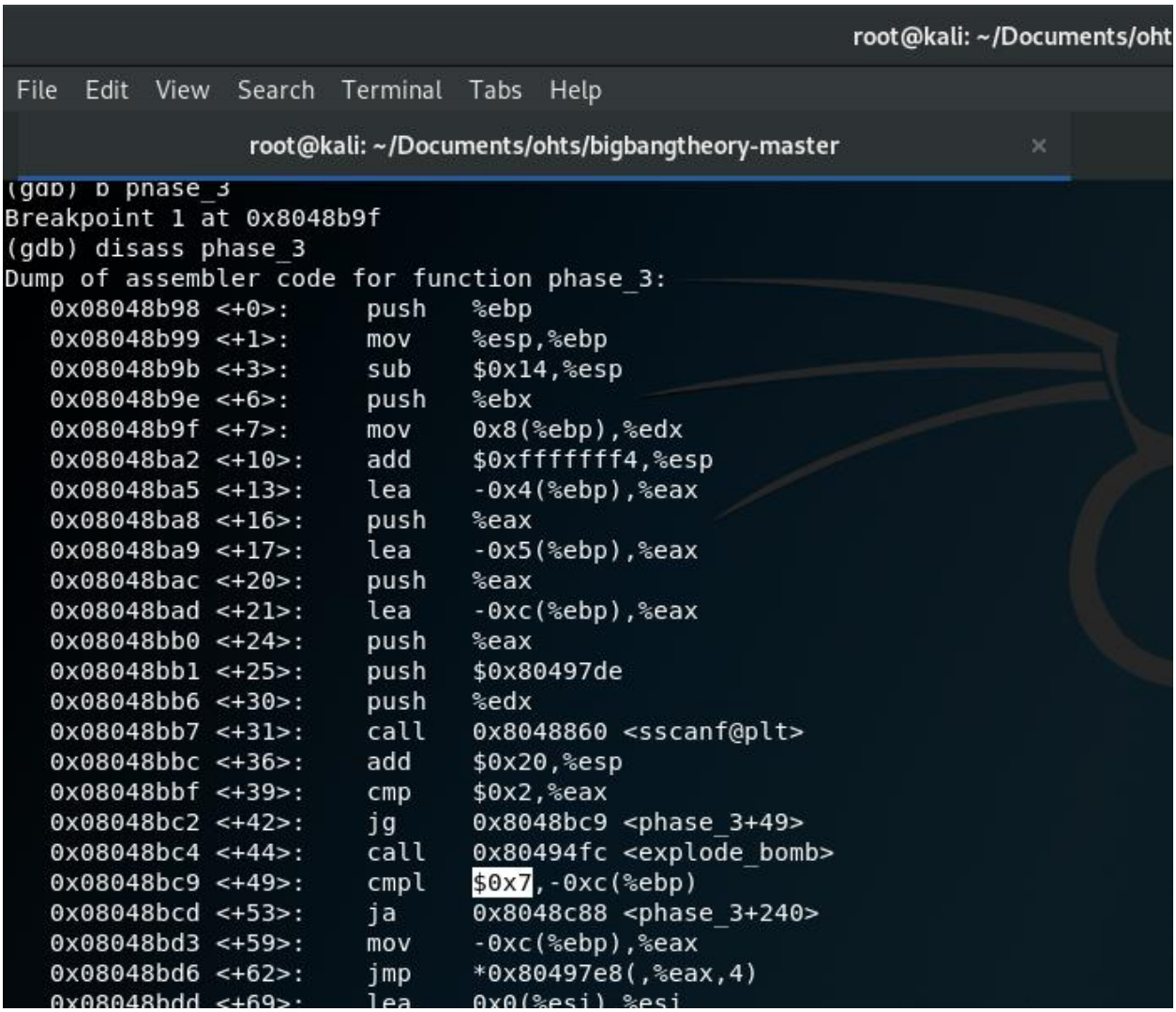

Figure 5 : Identification of the highest number for the first integer position

After inserting an input which starts with the integer value 2, we can follow through the execution by using the command next instruction(ni). First integer (value 2) will eventually direct the execution to line number 126 [Figure 6].

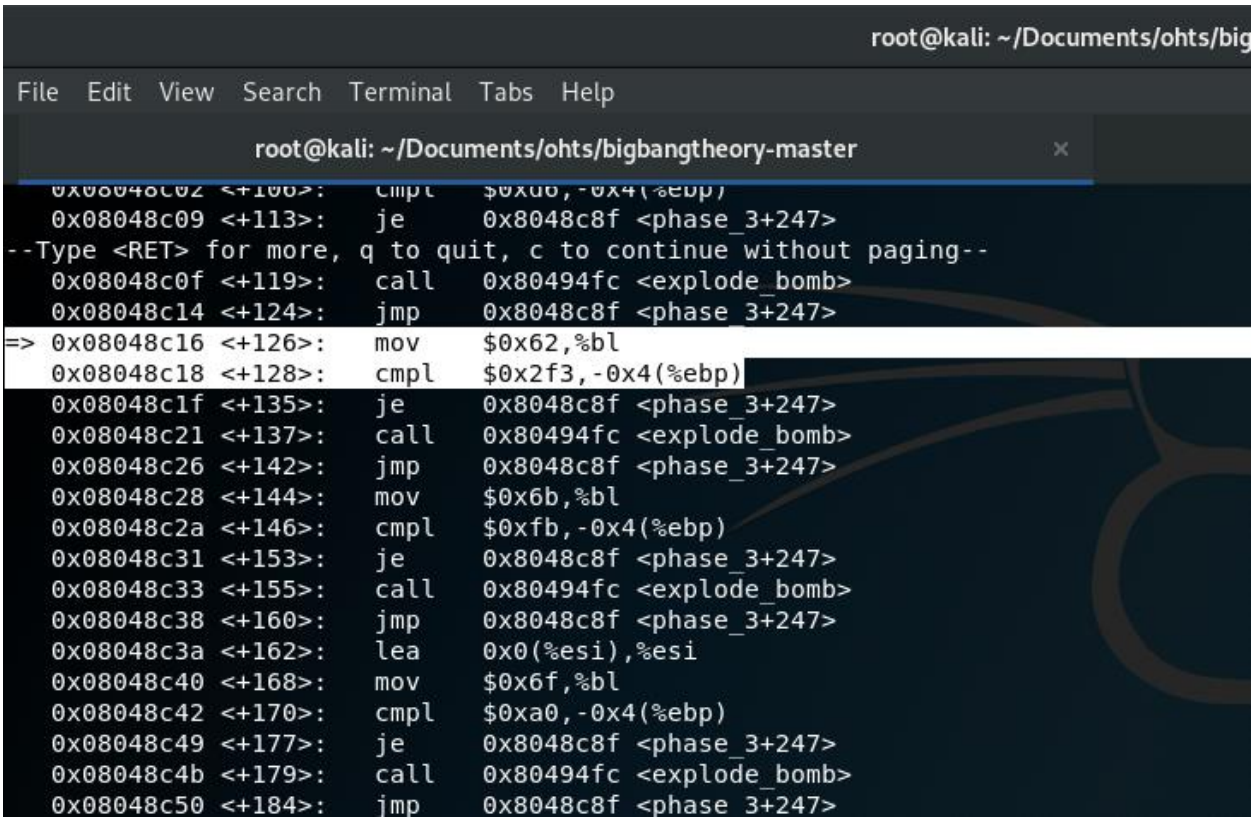

Figure 6 : Identification of the second and third values

At line number 126 we can see that a value getting moved to the %bl [Figure 6]. A value is being compared with the second integer (third input value) at the line 128 [Figure 6]. Now, let's see what's inside those variables.

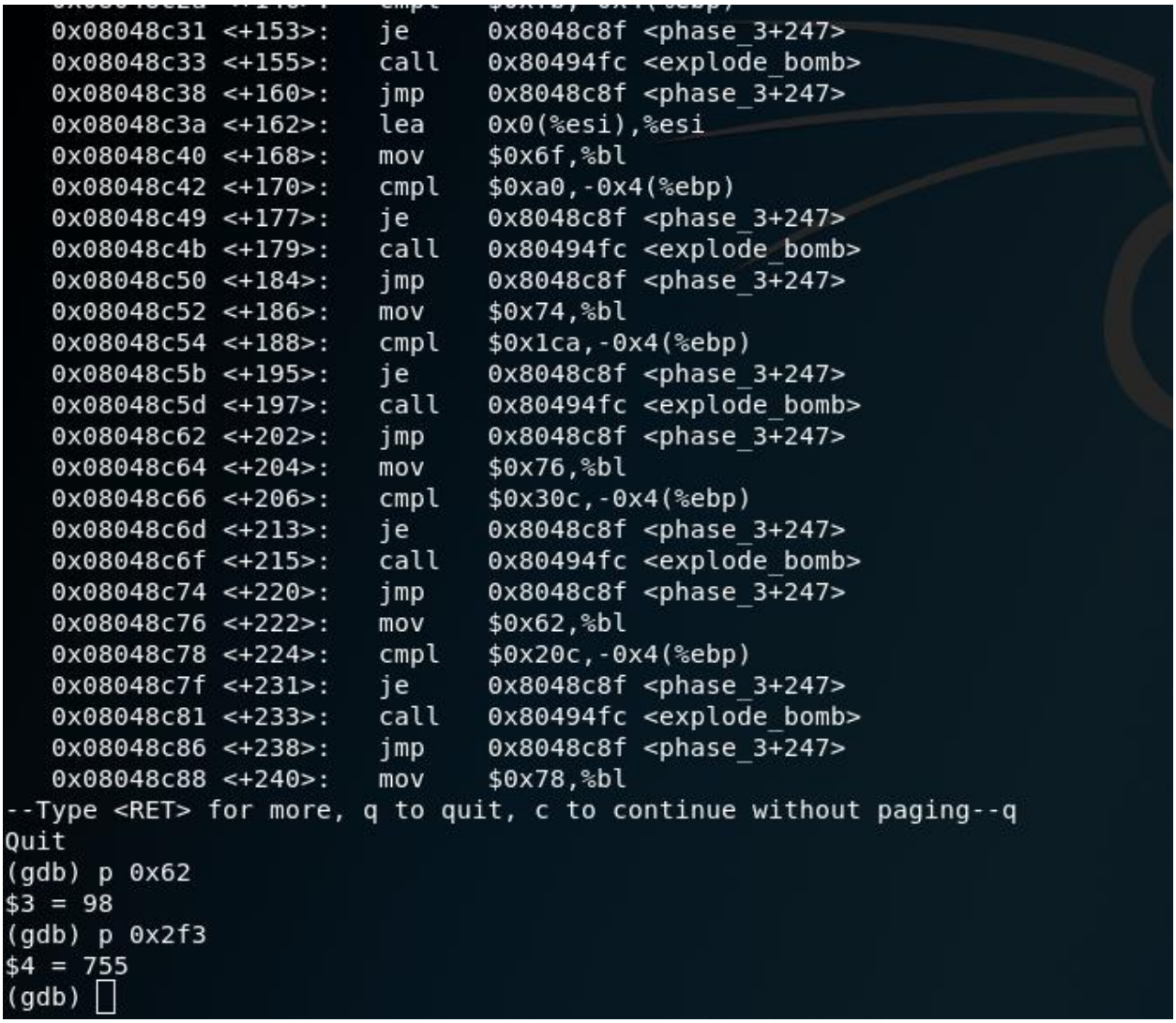

Figure 7 : Printing the values in decimal

We found two values by following the code which was initiated with number two as the first integer. Now we know that the first integer value is 2 while the second integer value being 755 [Figure 7].Though, we are sure about 98 belongs to the second input value, let's see how we can confirm it. By looking at the line number 247 [Figure 8], we can see the comparison happens with ox5(position) and %bl value. That confirms the second value is 98. But we found it was a character from the previous findings. Now we must find the ASCII character of the decimal value 98 by referring an ASCII table. The character turned out as 'b'.

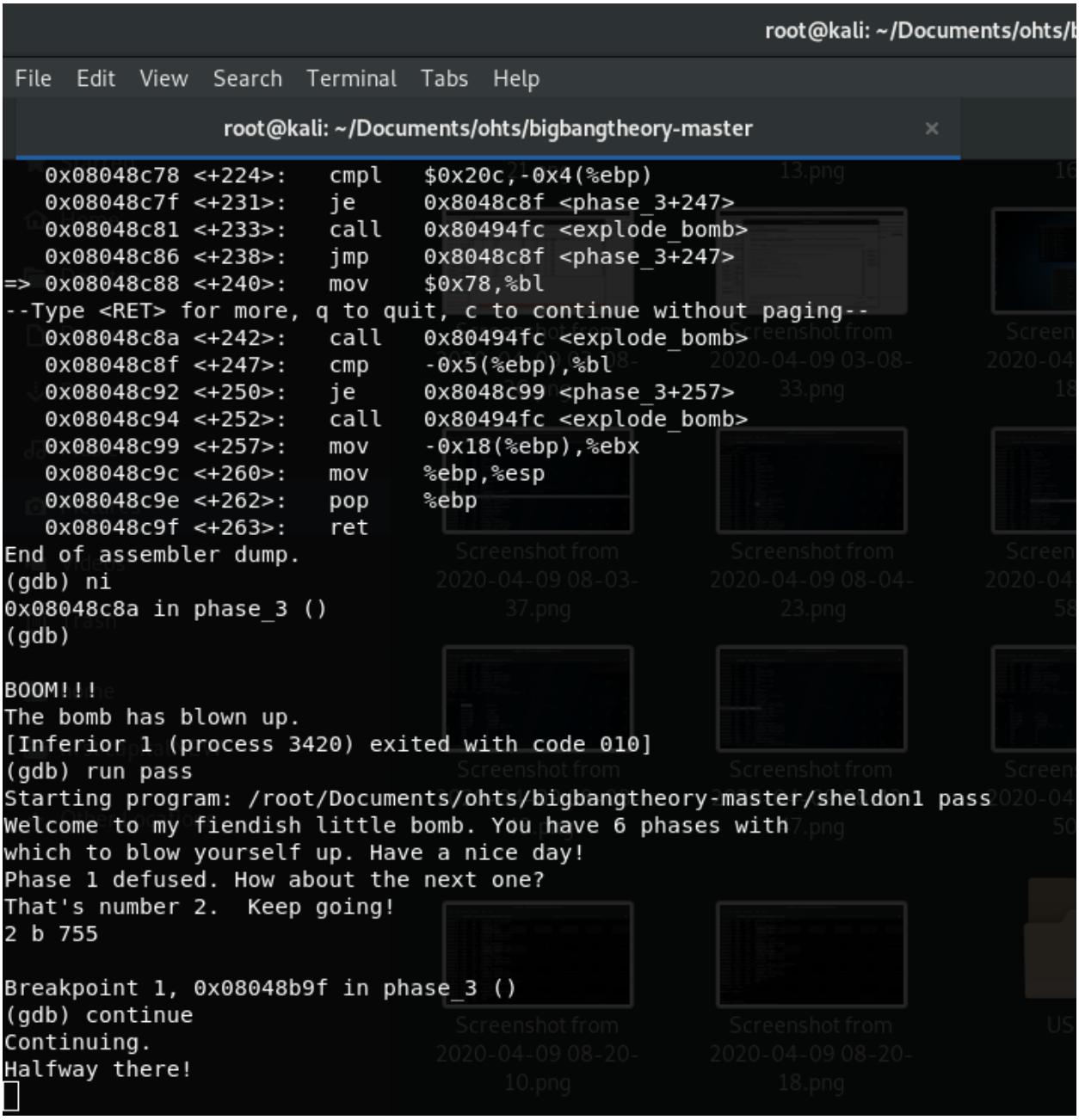

Figure 8: Testing the functionality with the found input

After providing the '2 b 755' as the input the phase_3 was diffused [Figure 8].

At the beginning I've mentioned that there are several combinations which can be taken as inputs. Below are some other found valid inputs.

' 1 b 214 '

' 3 k 251 '