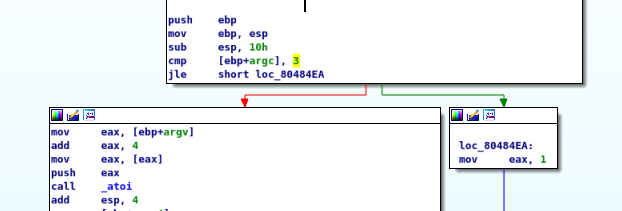
**1. Magic: use GDB (along with any tools covered in class) to reverse “magic”. (10 points)**

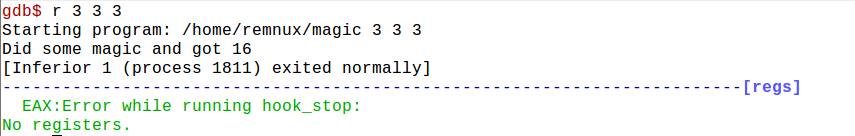
**a. How did you get this program to execute its functionality? Provide program inputs and evidence in the assembly code to support your findings.**

1.First run “chmod u=wrx magic” to gain privilege.

2. Open with IDA pro, it seems like if the input greater or equal than 3, than do the multiplication



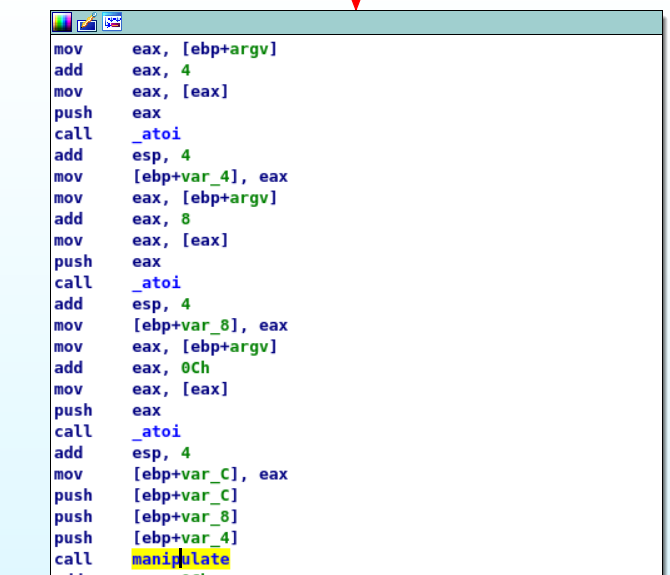
3. run with 3 parameters, it works.



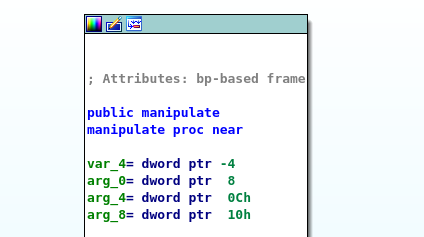
**b. What is the “magic” being done in this program? Provide evidence in the assembly code to support your findings.**

1. Take three input

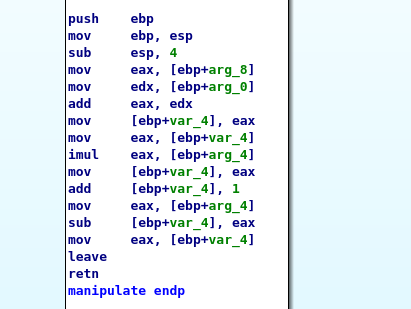
2. Take three inputs and turn into int.



3. pass the three input into function manipulate



4.Manipulate



Var\_4=input1+input3

Var\_4\*=input2

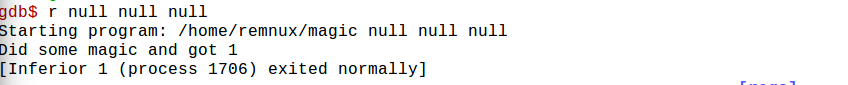
Var\_4+1

Var\_4-=input2

**c. Does this program have vulnerabilities? If so, list all vulnerabilities you can find? Provide evidence to support each of your claims?**

The function atoi will return 0 if the input is an invalid string, similarly will return 0 if the input is zero. Thus, the function can't distinguish a valid 0 from an invalid string

d. If vulnerabilities exist, perform **an exploit** and document your steps.

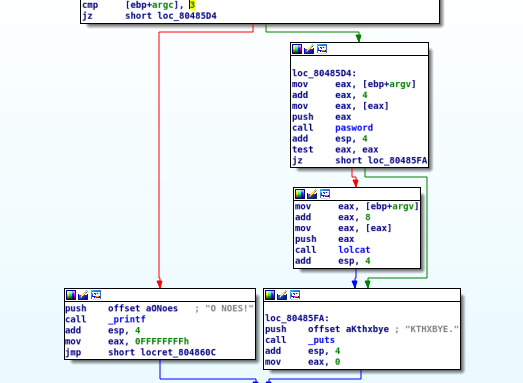


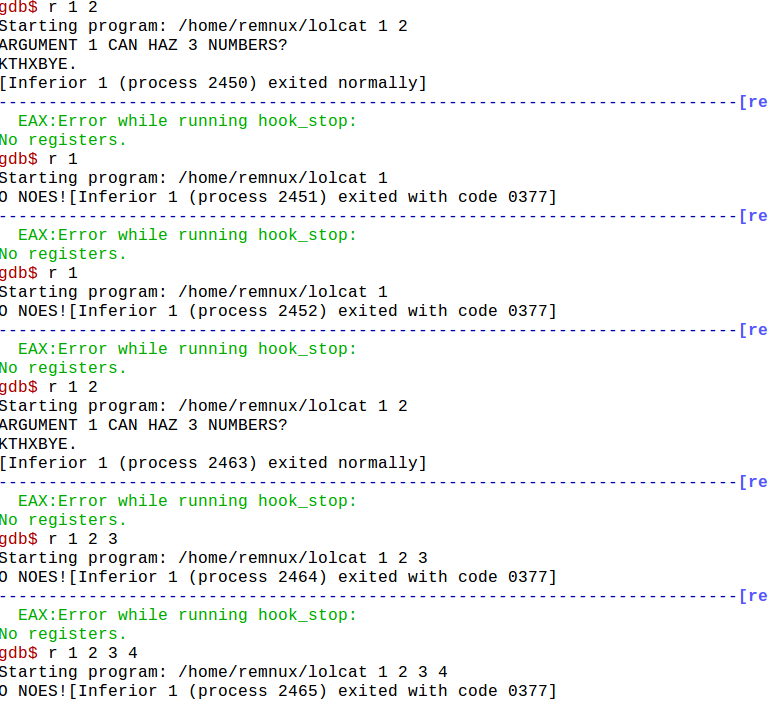
**2. LoLCat: use any reversing tool of your choice to reverse “lolcat”, be sure to specify the tool(s) used. (5 points)**

**a. How did you get this program to execute its functionality? Provide program inputs and evidence in the assembly code to support your findings.**

1.Run “chmod u=wrx lolcat” to gain privilege .

2. Based on IDA pro, I found out the input number will impact the behavior of the program.

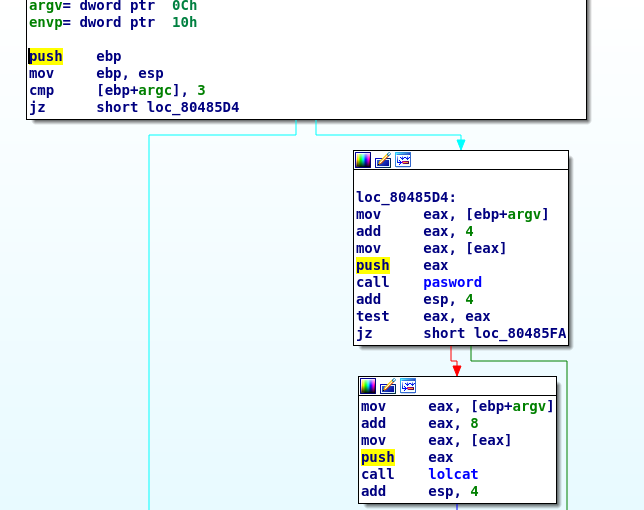




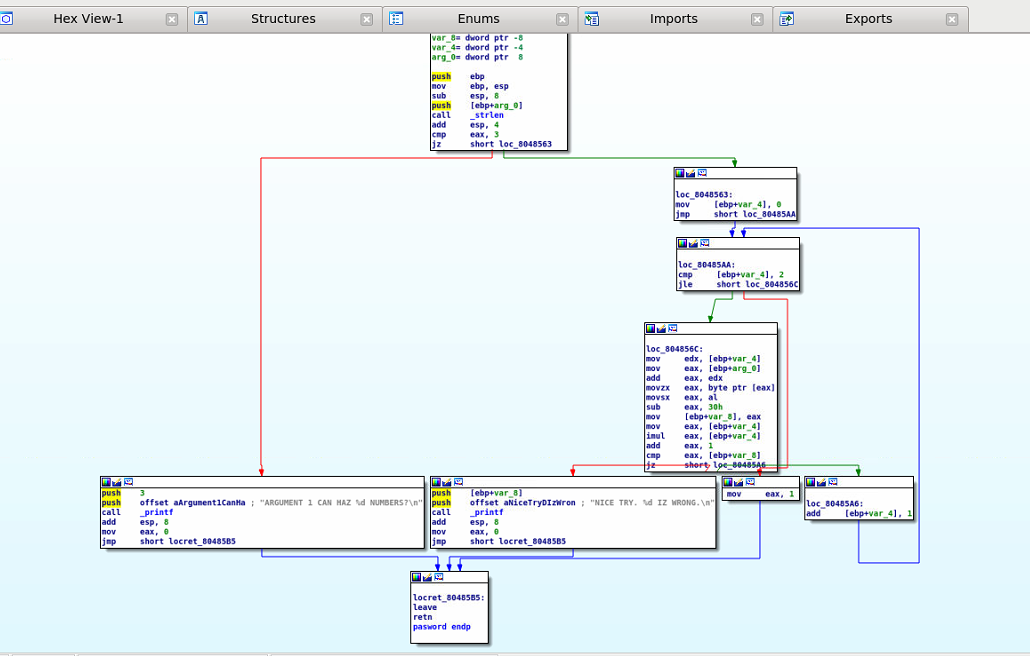
3. In order to reach lolcat function, the first input should be 125.

b. How did you execute the functionality in the function lolcat()? Provide **program inputs**and evidence in the **assembly code** to support your findings.

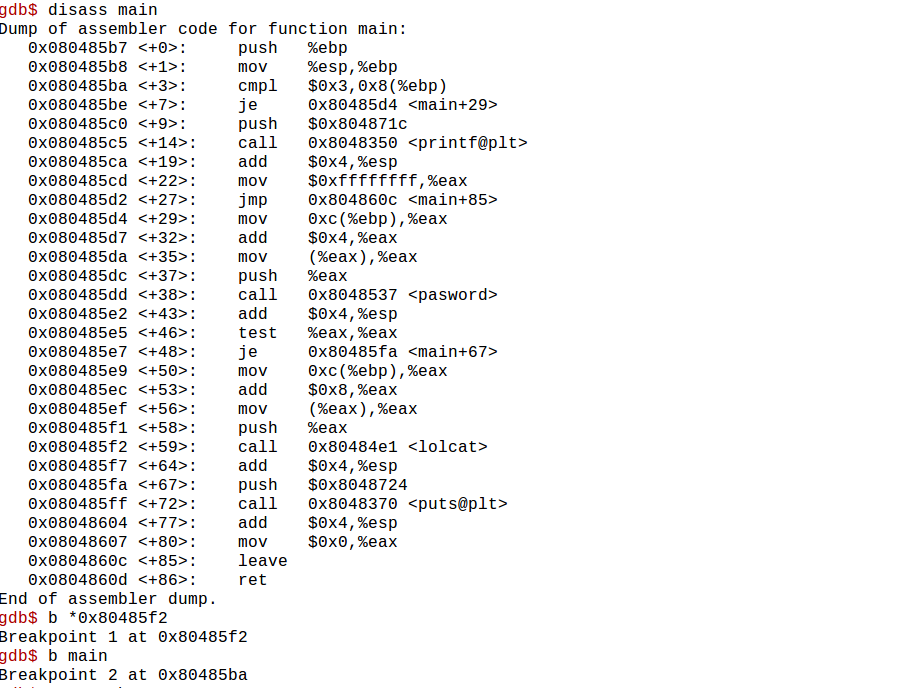
1. When examining the main function, I found that in order to hit the code block that contains lolcat, we have to pass the test eax,eax syntax.(check the return )



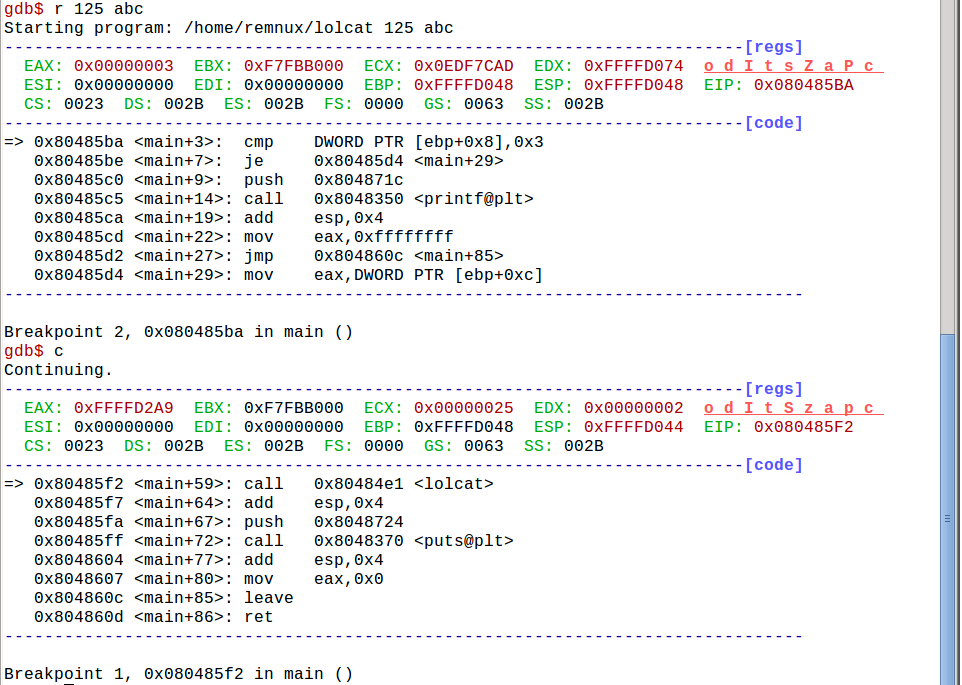
2. In the pasword function, I found out the parameters should be 125, then it will return 1.



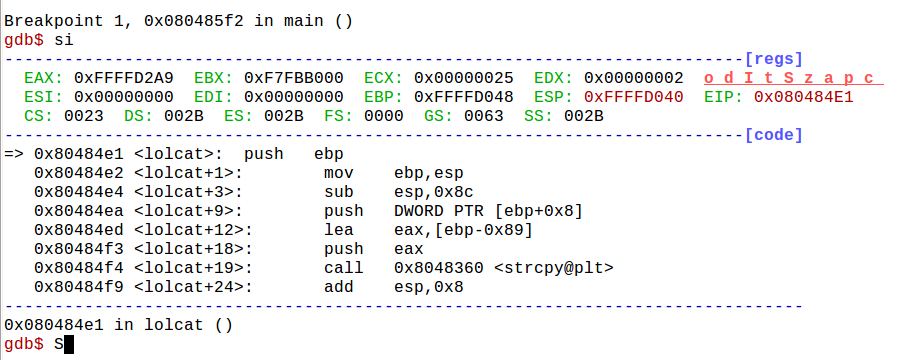


3. Set breakpoints 

4. Then run with parameters

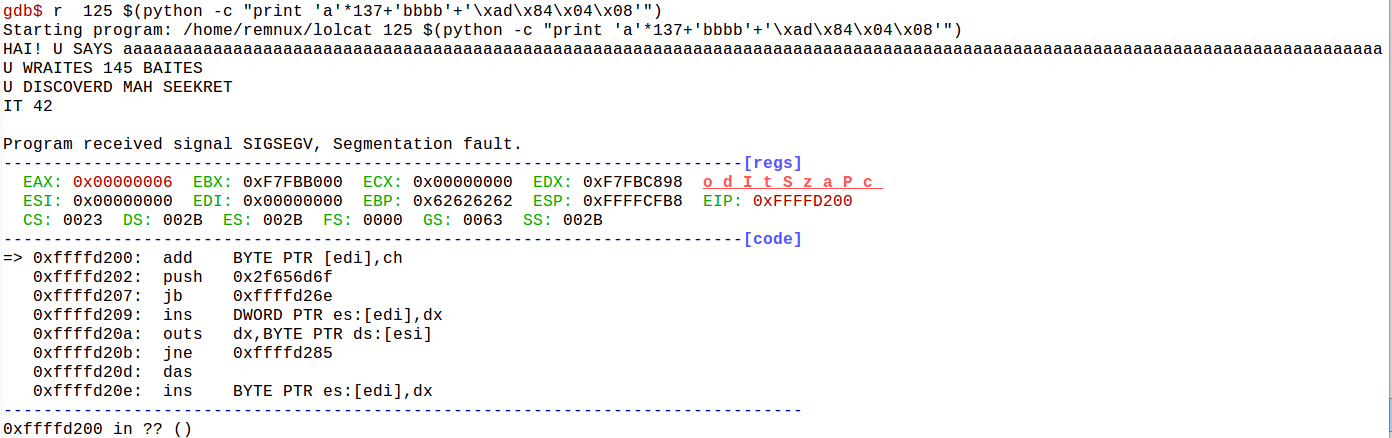


5. Assure it went into lolcat

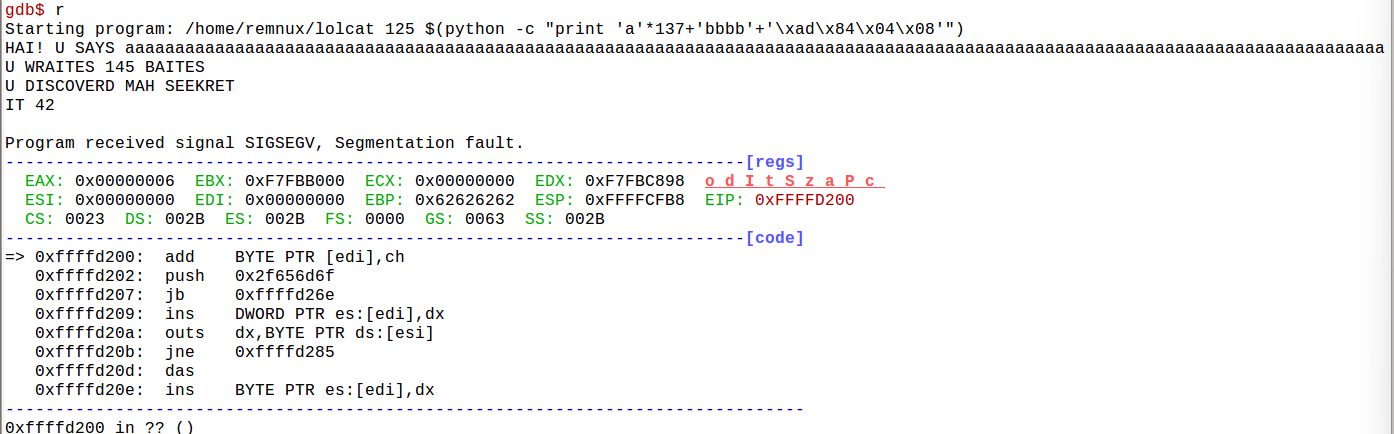


**c. What is the “secret” in this program? Provide evidence in the assembly code to support your findings.**

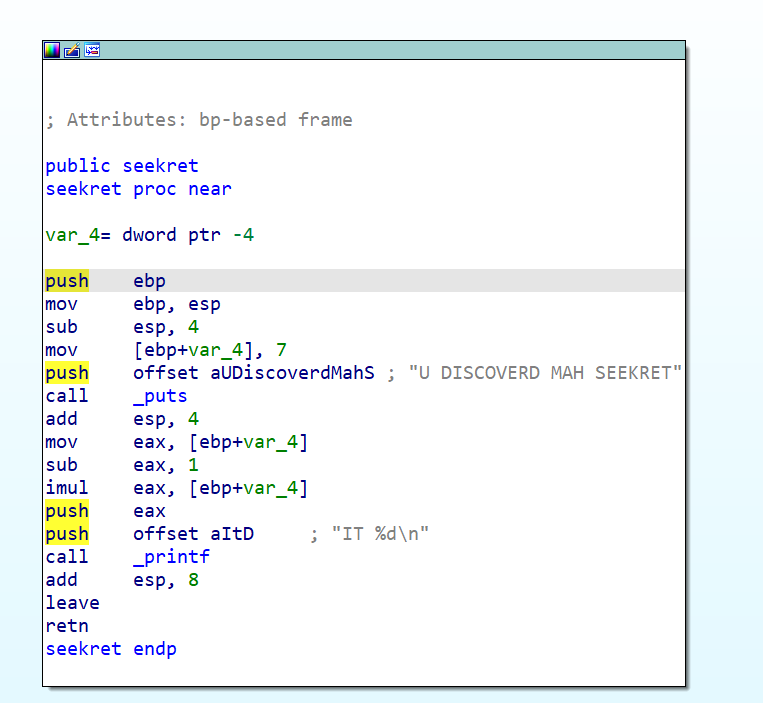
1.If overwrite the return address in “lolcat” function, we can force it to call the “seekret” function while returning.



2. It will crash the system after once it reached.



3. I viewed it on IDA PRO.





It will return 42 every time.

**d. Does this program have vulnerabilities? If so, list all vulnerabilities you can find? Provide evidence to support each of your claims?**

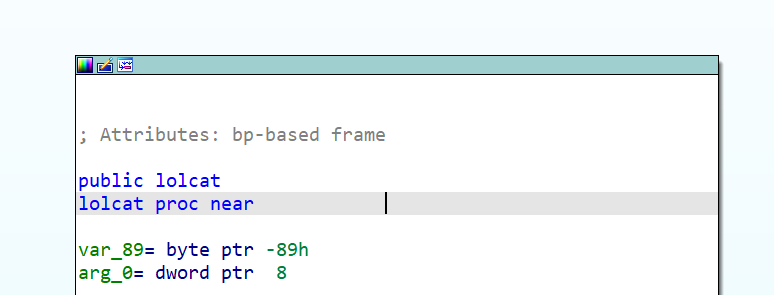
1. Buffer overflow

2. System carsh

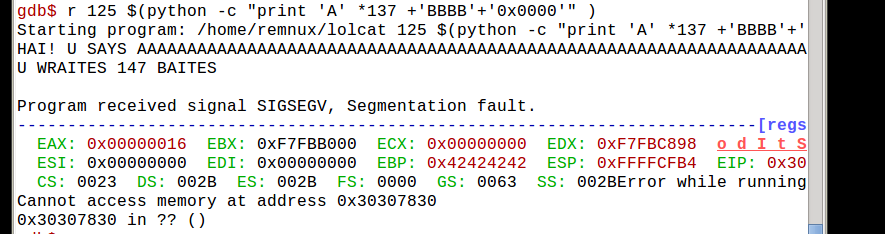
e. If vulnerabilities exist, perform **an exploit** and document your steps.

1. Buffer overflow

1.1 I found out there’s a buffer in lolcat function

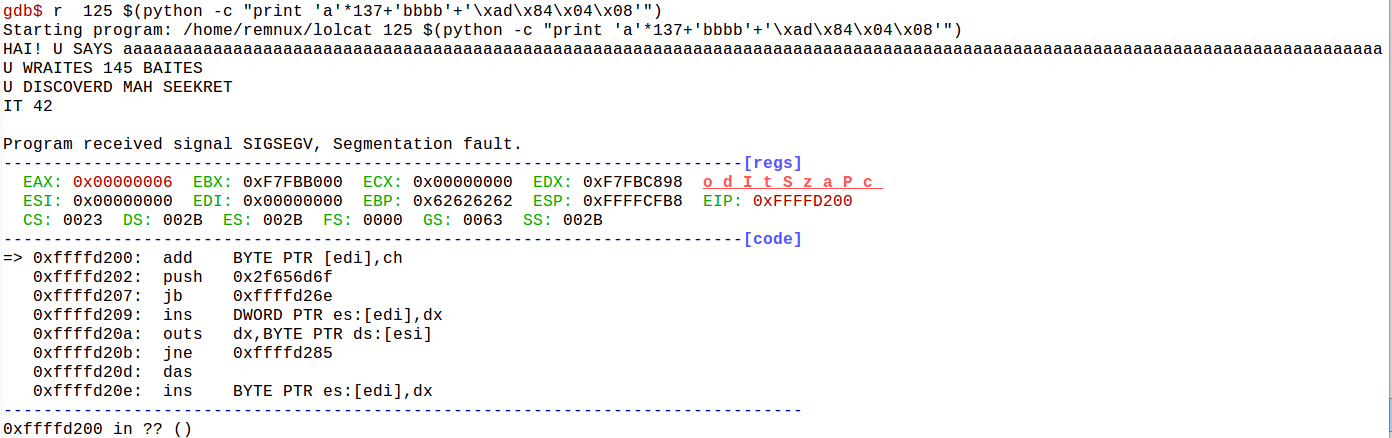


1.2 Try to overwrite it’s return address and it caused segmentation fault.



2. System Crash

2.1.If overwrite the return address in “lolcat” function, we can force it to call the “seekret” function while returning.



2.2. It will crash the system once it is reached.

