PSP0201
Week 5
Writeup

Group Name: Undecided

Members

ID	Name	Role
1211101390	Aslamia Najwa Binti Ahmad Khadri	Leader
1211100431	Mohammad Omar Torofder	Member
1211103388	Vishnu Karmegam	Member
1211103092	Farryn Aisha binti Muhd Firdaus	Member

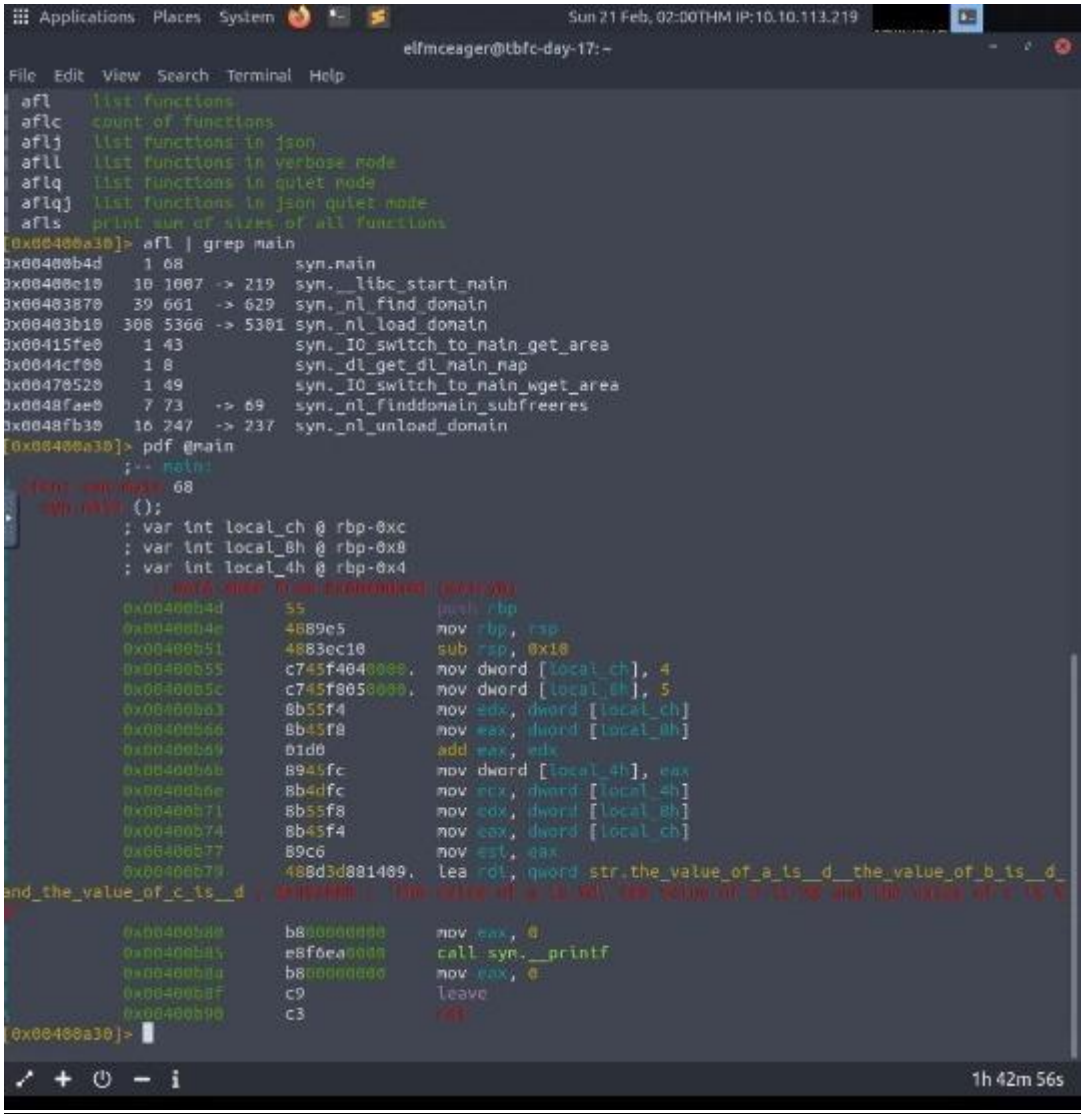
Day 17: Reverse Engineering – ReverseELFneering

Tools used: Attackbox, Firefox, Virtualbox

Solution/walkthrough:

Question 1

View the pdf@main and print the code that is in the sim.main



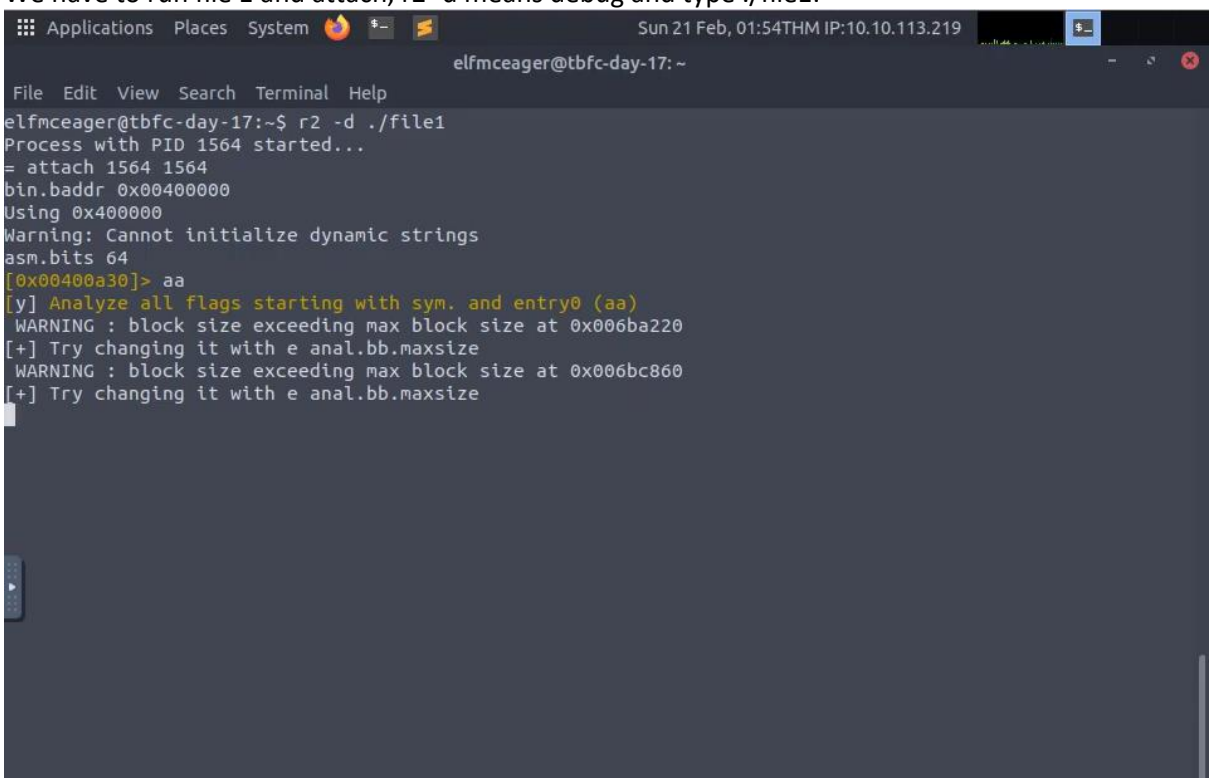
A register is a workbench where we can work on task one at a time where we can restore its value. We will have a register that will hold the value of local ch and another folder that hold the local 8h.

We have to perform mathematical functions on the two registers and it will overwrite the source register with the result.

Therefore, the data type that were matched with the size in bytes is byte = 1, Word = 2, Double Word = 4, Quad = 8, Single Precision = 4, Double Precision = 8

Question 2

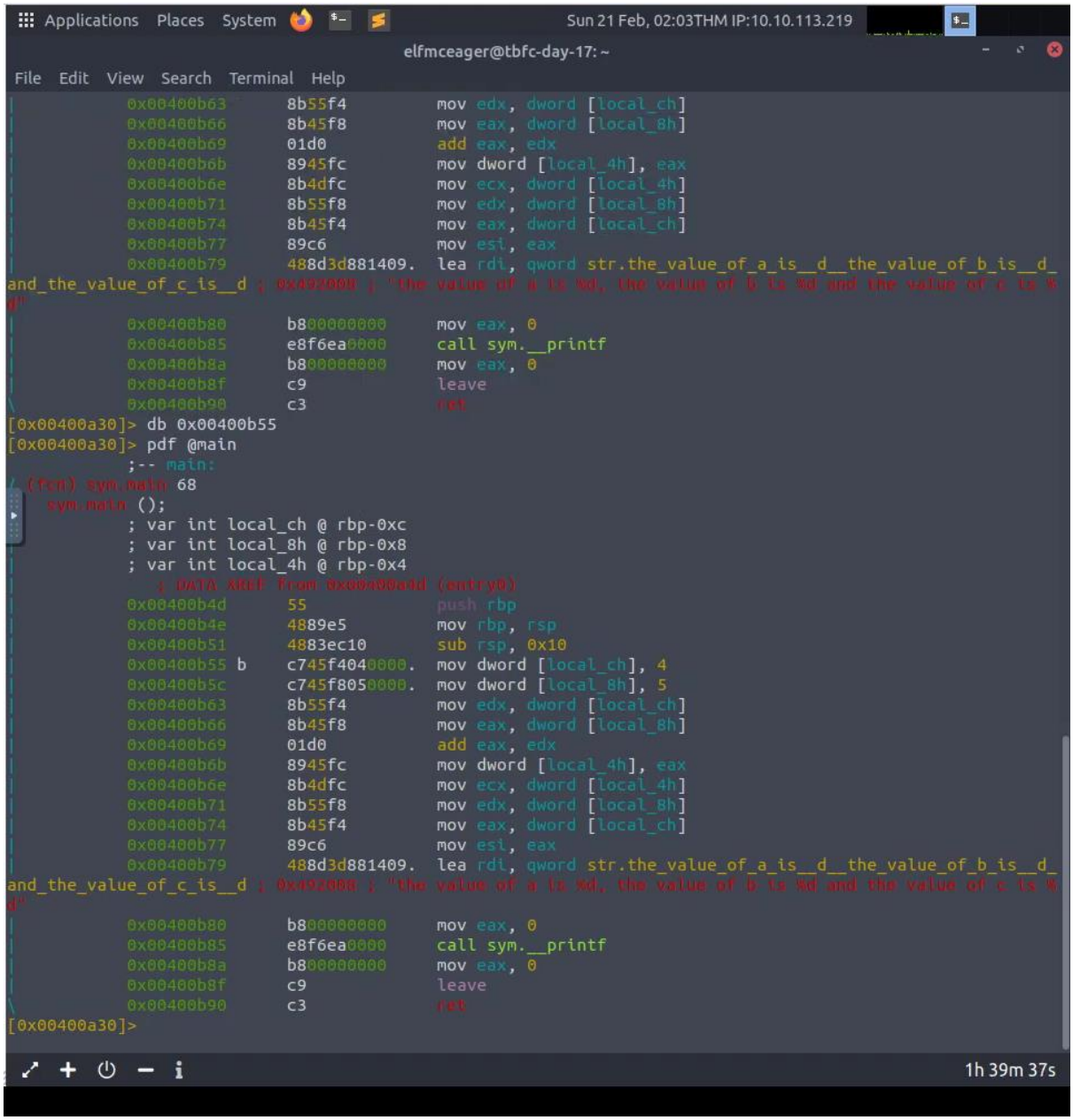
We have to run file 1 and attach, r2 -d means debug and type ./file1.



Thus, aa is the command that we entered to start analysing.

Question 3

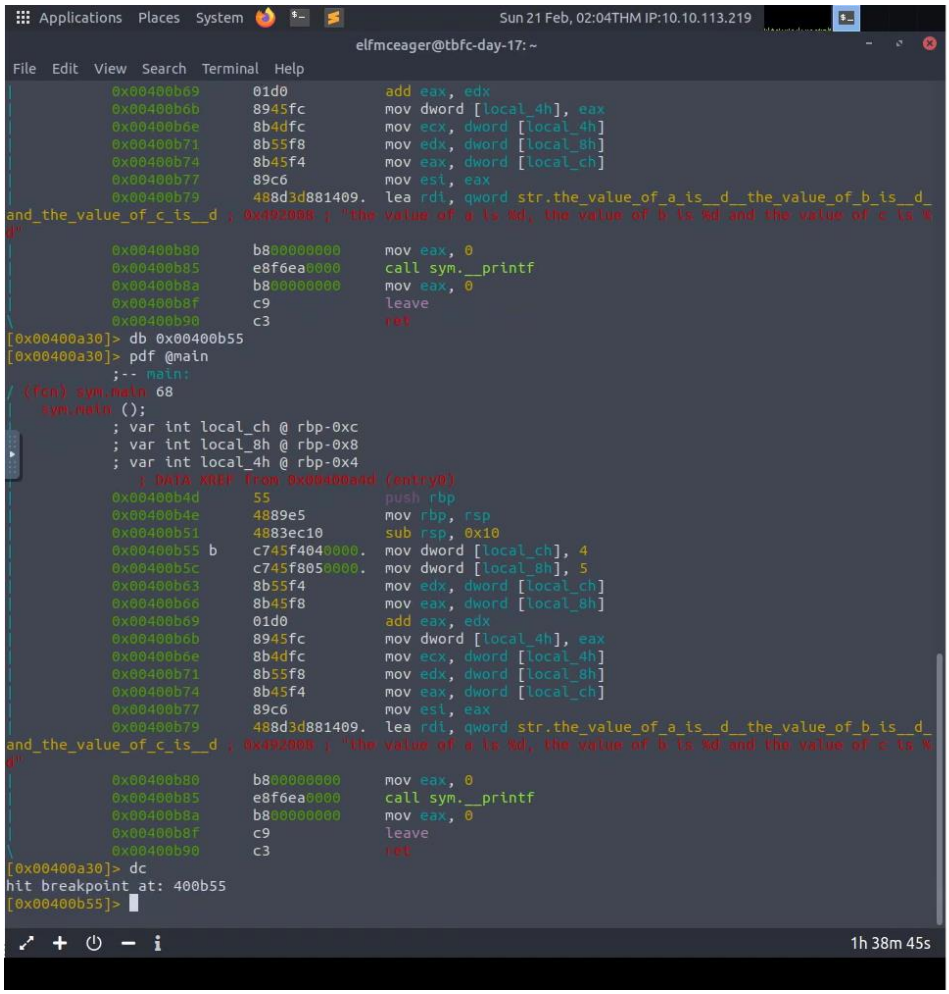
We can set the breakpoint by using the db command. Type db0x00400b55 and pdf@main.



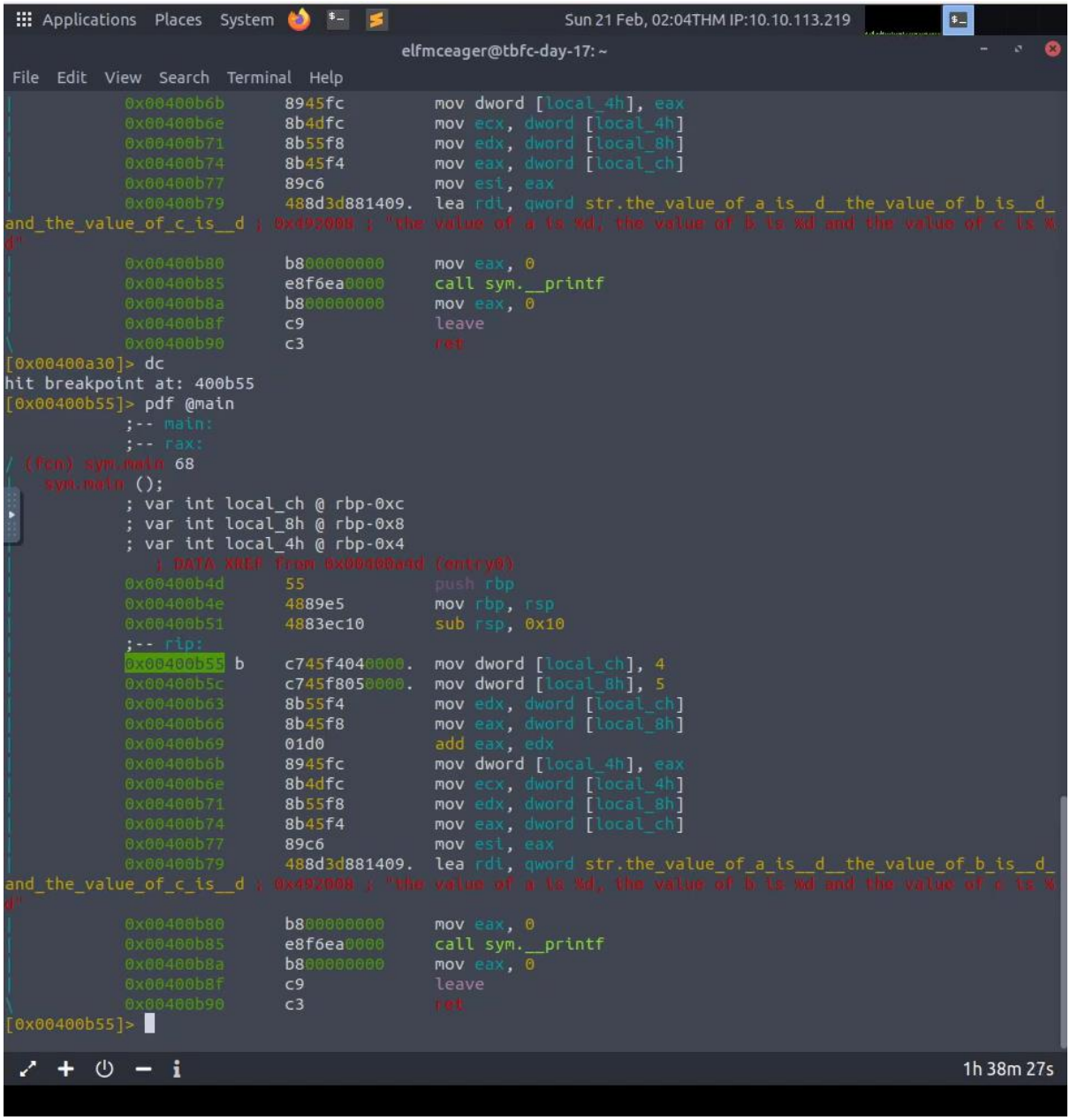
We will see a lowercase b next to it which is telling us that we have a breakpoint.

Question 4

We can type in dc which starts sunning the code and it will run up the first breakpoint.

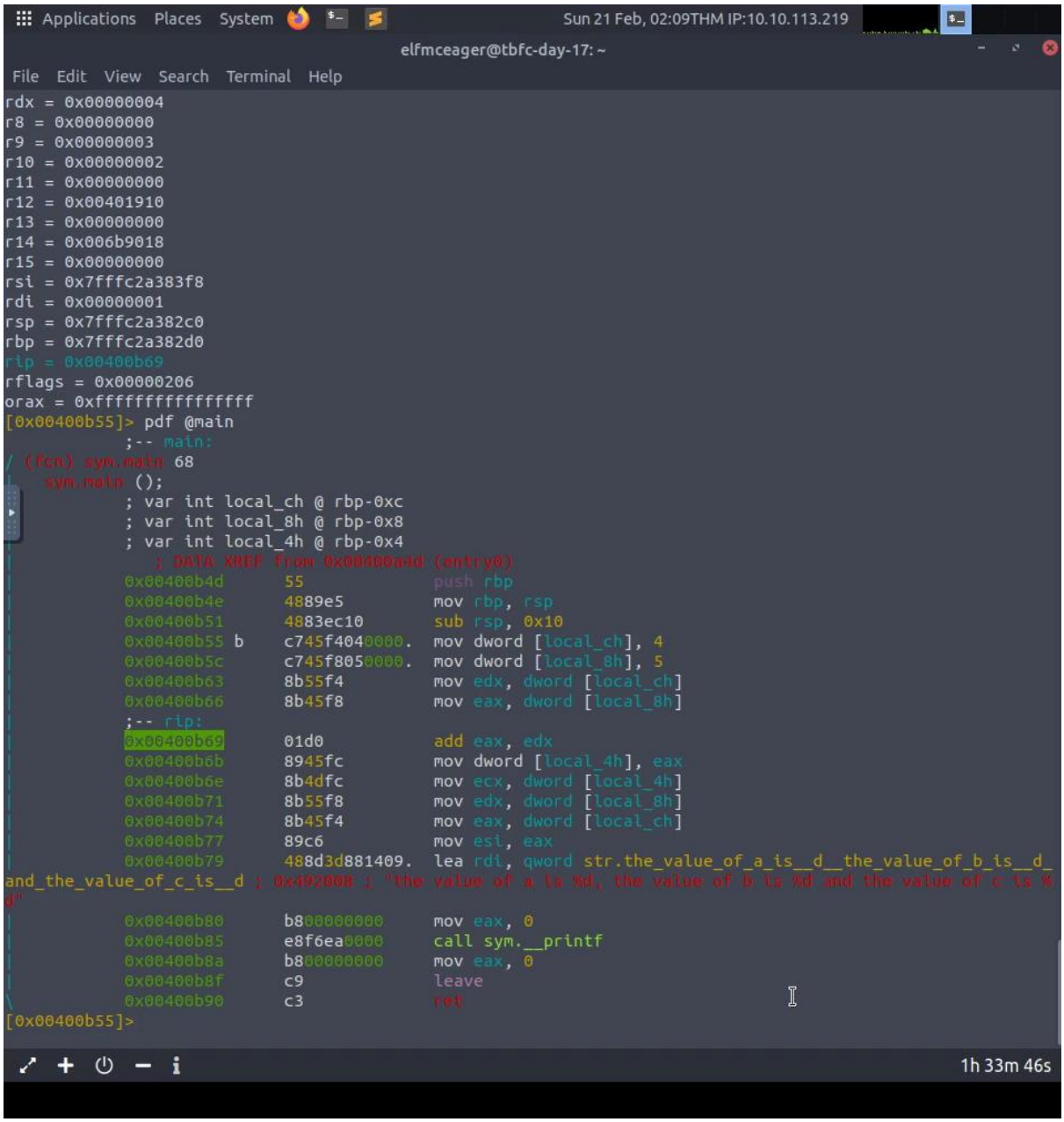


We have set a memory location 400b55 and that is where it stopped. When we do a pdf@main again, we will see it is highlighted and there is a rip comment where we are currently sitting.

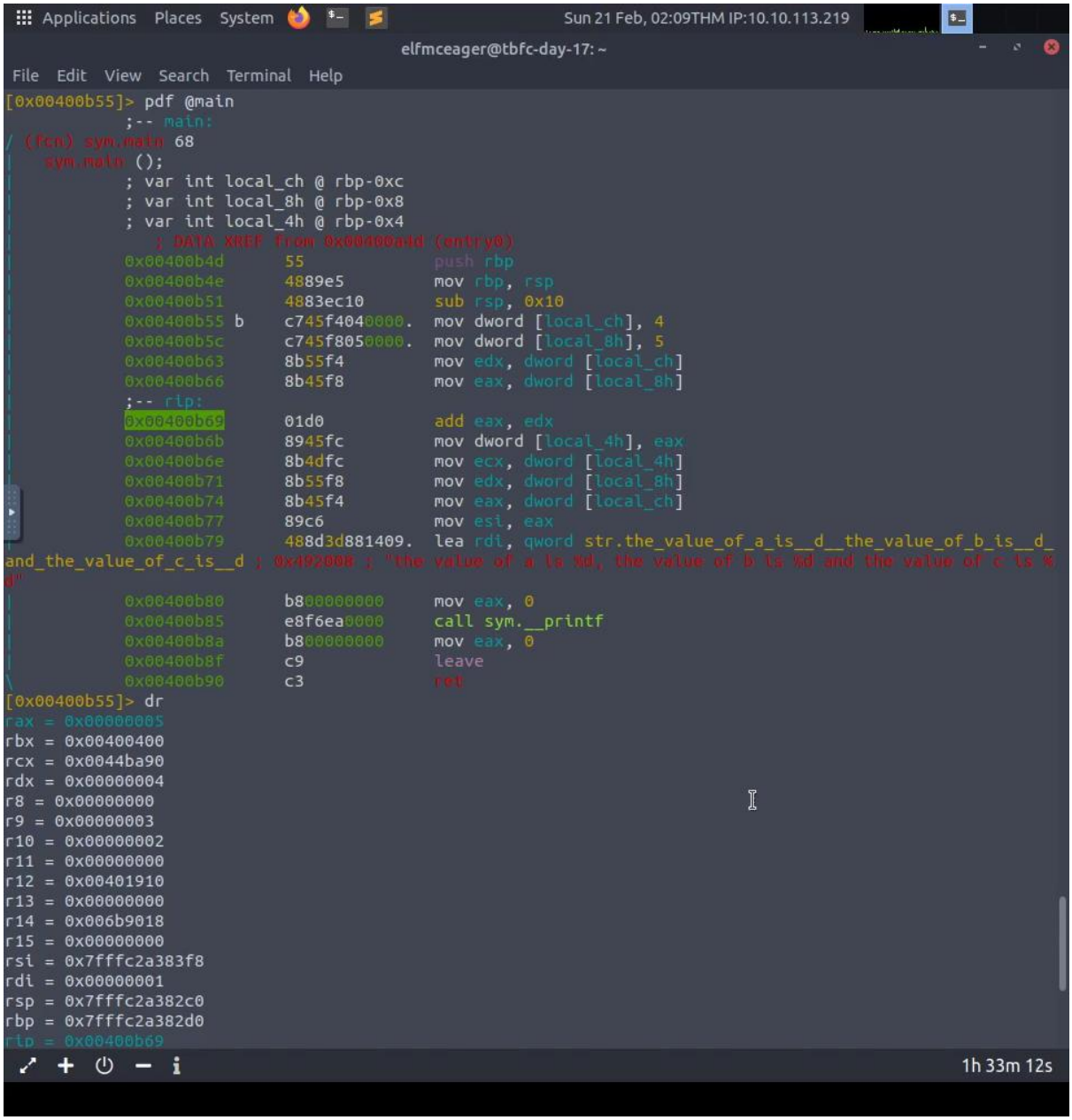


Question 5,6 & 7

When we look at the code again, we will see that we are right at the line where we do the addition of eax + edx



We have to look at the registers, rax = 5, rdx = 4, when we step, we will perform the addition and one of our registers will be modified.



Thus, 1 is the value of local_ch when its corresponding movl instruction is called, 6 is the value of eax when the imull instruction is called, and 6 is the value of local_4h before eax is set to 0.

Thought Process/Methodology:

Print the code found in sim.main and view the pdf@main. A register serves as a work surface where we can do tasks one at a time in order to recover their value. The local ch value will be kept in a register, while the local 8h value will be kept in another folder. The two registers must be subjected to mathematical operations, and the results will be written over the source register. As a result, byte = 1, word = 2, double word = 4, quad = 8, single precision = 4, double precision = 8, and the data type were matched with the size in bytes. To execute file 1 and attach, type./file1. Type r2 -d to debug. Thus, we entered the command aa to begin our analysis. Using the db command, we can set the breakpoint. Enter pdf@main and db0x00400b55. The fact that there is a lowercase b next to it indicates that we have reached a breakpoint. We may start the code by typing dc, and it will run up

the first breakpoint. It halted at a memory location that we placed at 400b55. It will be underlined and have a rip comment where we are sitting if we run a pdf@main once more. Rereading the code will reveal that we are currently on the line where the addition of eax and edx is performed. When we step, we will conduct the addition, and one of our registers will be changed. The registers are: rax = 5, rdx = 4, etc. Thus, 1 represents the value of local ch when the movl instruction corresponding to it is called, 6 represents the value of eax when the imull instruction is executed, and 6 represents the value of local 4h prior to eax being set to 0.