MINISTRY OF EDUCATION AND SCIENCE OF UKRAINE

NATIONAL TECHNICAL UNIVERSITY

"KHARKIV POLYTECHNICAL INSTITUTE"

Department of Computer Engineering and Programming

«Software Means of Information Protection »

*Laboratory work report No 2*

*Topic: «* **Code injection in an executable file at the end of a section without API-function calls** *»*

Student:

Group. KH 919 I.E

Ayoub EL-HADDADI

Verified by:

Lecturer Viktor CHELAK

Kharkiv – 2022

***Purpose of work***:

To acquire practical skills for correcting errors in software that is in operation; creation of new functionality using implicit in use software for the x64 platform in the masm64 environment.

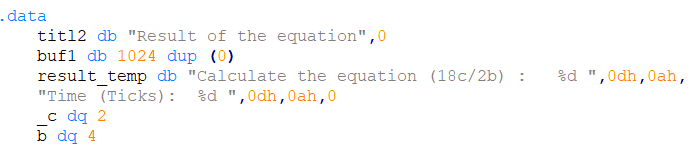
***Individual task:***

Variant 8:

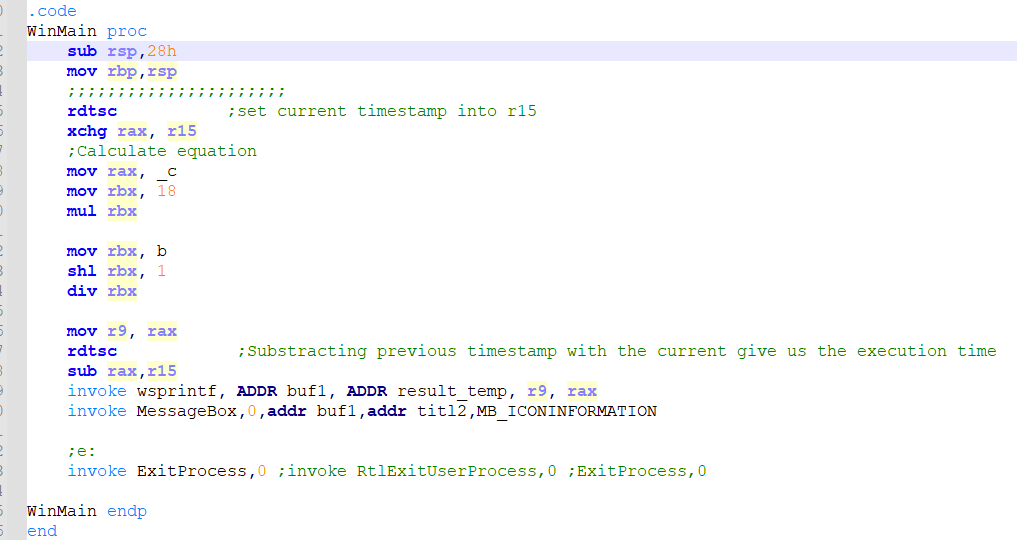
8. 18c/2b;   —>     32ab/2a – 4с;

**Algorithm of the program**

1. *data section of the program*



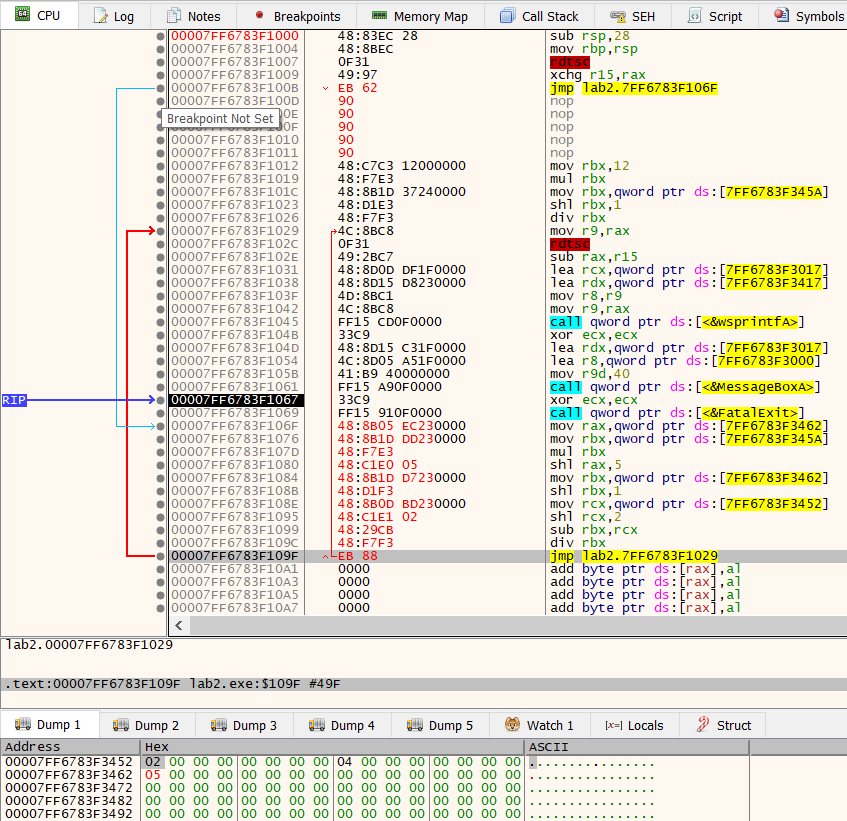
In this section, we initialize variables “**c**”, “**d**” and template that we are going to use in printing results of the .code section.

1. *code section of the program*

In this section we are saving current timestamp into **“r15**” **(**using **rdtsc** that save current timestamp in register “**rax**”), and proceed to calculating the equation: **(18c / 2b)**, next we get timestamp after calculation of equation and subtract it with the first one, the result of this subtraction represent the execution time of the equation.

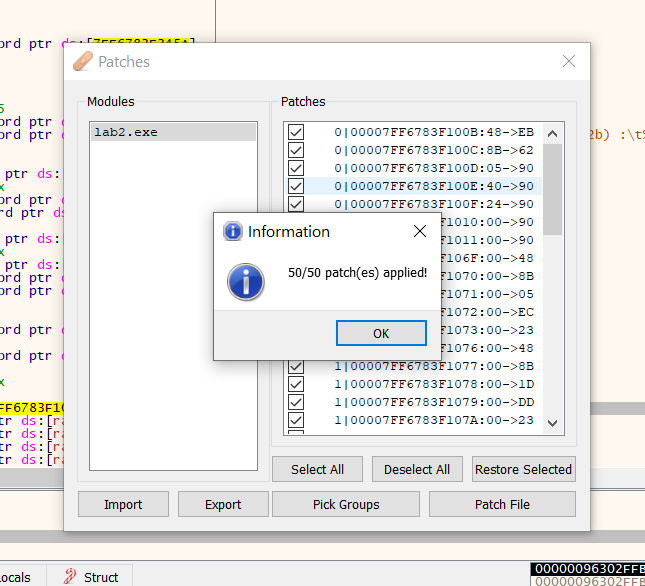
Final step is to show results through a message box, and invoke an exit process.

1. Injection in an executable file



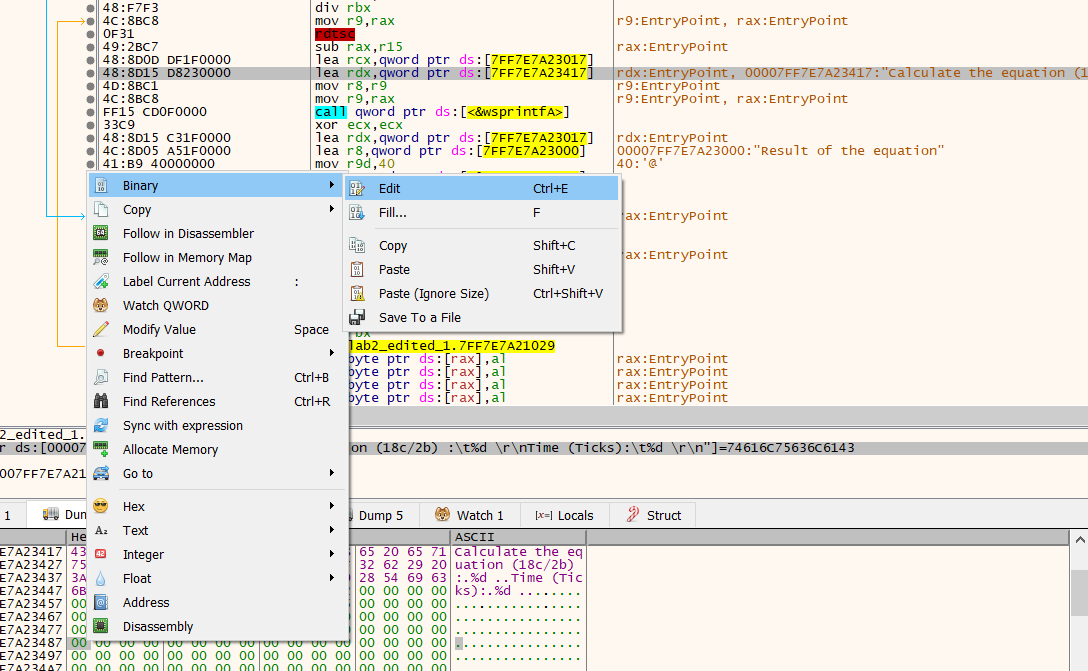
To modify executable file for injecting the new equation, we set a jump to the end of the code exactly after the first **rdtsc** command (After saving timestamp before the calculation begins), then, we just need to specify the new commands with assemble (space). After writing desired instructions, we jump to the second **rdtsc** command (which will get timestamp after calculation and subtract it with the first one), and calling the message box for showing results.

*N.B: Since we need a new variable (****a****), we can add it to memory and use its address (you can notice it in bottom of previous picture with red highlighting).*

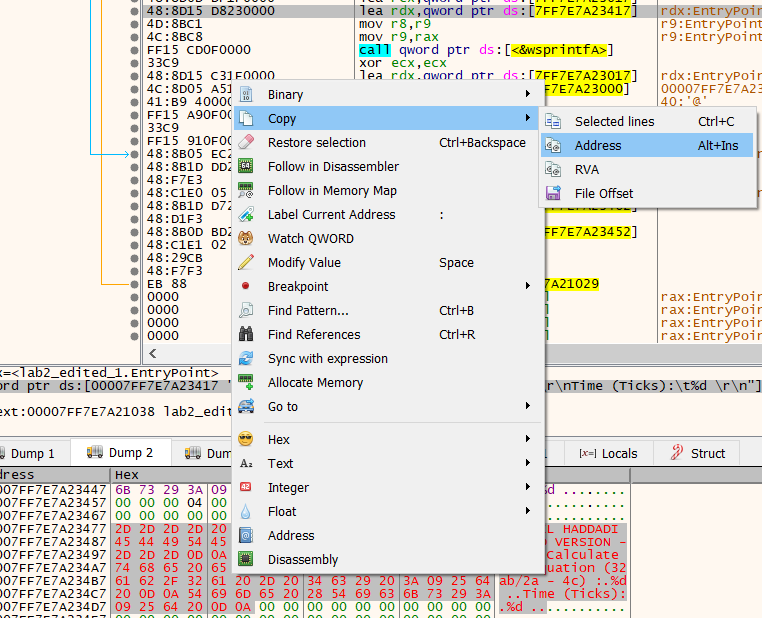


The following step is to patch the file, and we can see how many bytes we have changed or added.

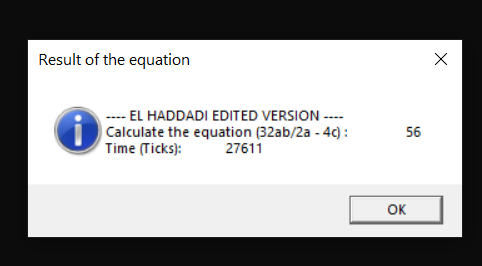
After patching and saving the modified executable file, we open it once again in the x64dbg for more editing in string references.



We can create a new string in memory by clicking in empty space with right click, and select binary edit



After creating new string we can copy its address and replace the string used in message box. After this changes we can patch a new executable file.



**Source Code**

Full source code of this lab you can find it in:

[**https://github.com/Elh-Ayoub/RP\_Labs/tree/main/lab2**](https://github.com/Elh-Ayoub/RP_Labs/tree/main/lab2)

**Results of the program:**

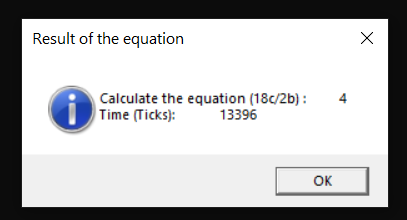


Figure 1 – lab2.exe

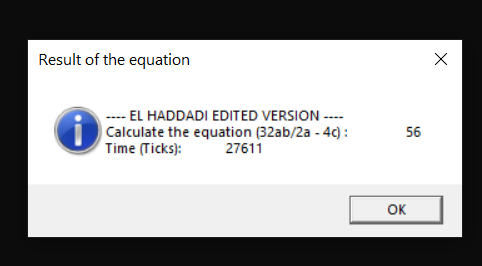


Figure 2 – lab2\_edited\_2.exe

**Conclusions:**

As a result of laboratory work we gained a practical skills in injecting and creation of new functionality using implicit in use software for the x64 platform in the masm64 environment.

**You can also find this report in:**

<https://github.com/Elh-Ayoub/RP_Labs/tree/main/Docs>