|  |  |
| --- | --- |
| Gerb-BMSTU_01 | **Министерство науки и высшего образования Российской Федерации**  **Федеральное государственное бюджетное образовательное учреждение**  **высшего образования**  **«Московский государственный технический университет**  **имени Н.Э. Баумана**  **(национальный исследовательский университет)»**  **(МГТУ им. Н.Э. Баумана)** |

ФАКУЛЬТЕТ «Информатика и системы управления»\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

КАФЕДРА «Программное обеспечение ЭВМ и информационные технологии»\_\_\_\_\_\_\_\_\_\_\_\_\_\_

**Лабораторная работа № \_\_9\_\_**

|  |  |
| --- | --- |
| **Тема Низкоуровневое программирование под Windows/Linux. Дизассемблирование. Реверс-инжиниринг.**  **Студент \_Брянская Е.В.\_\_\_\_\_\_\_\_**  **Группа \_ИУ7-42Б\_\_\_\_\_\_\_\_\_\_\_\_\_**  **Оценка (баллы) \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_**  **Преподаватель \_Кузнецов Д.А.** |  |

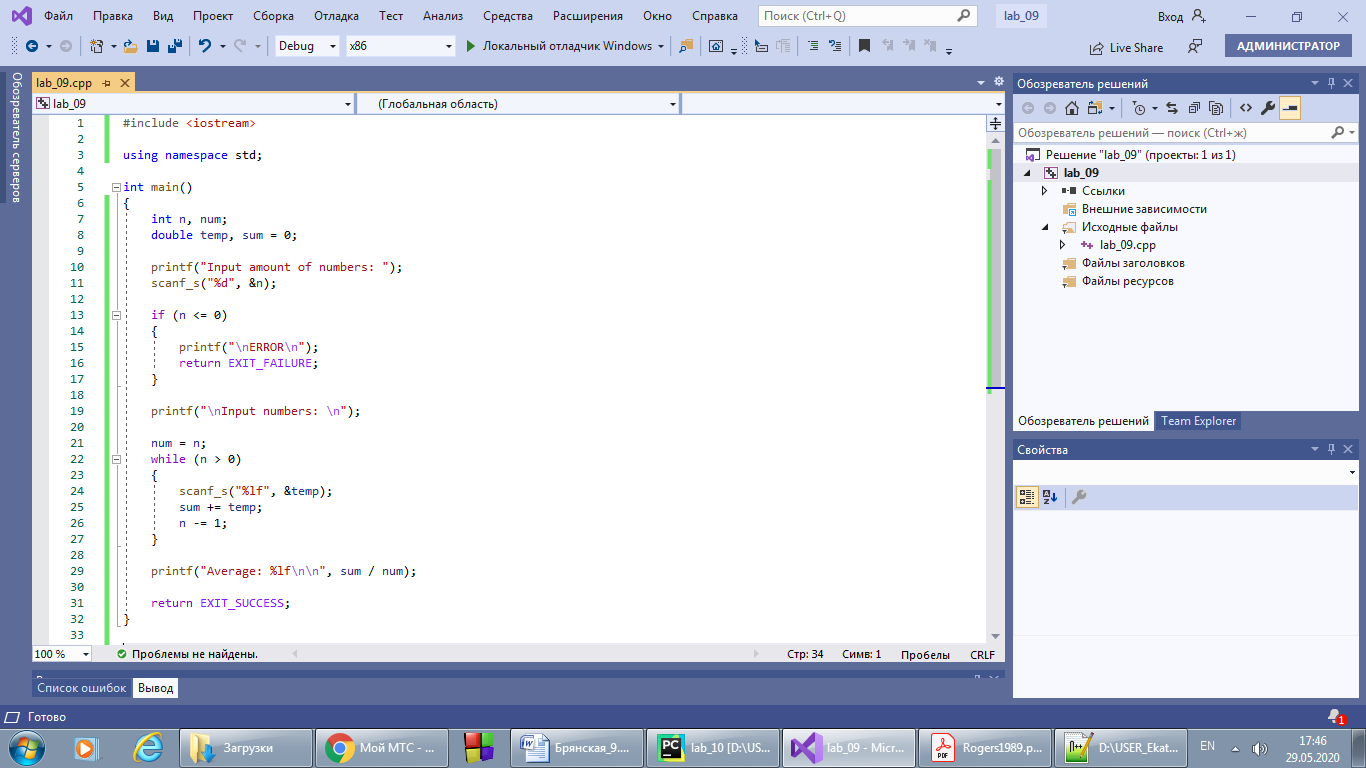
Москва.

2020 г.

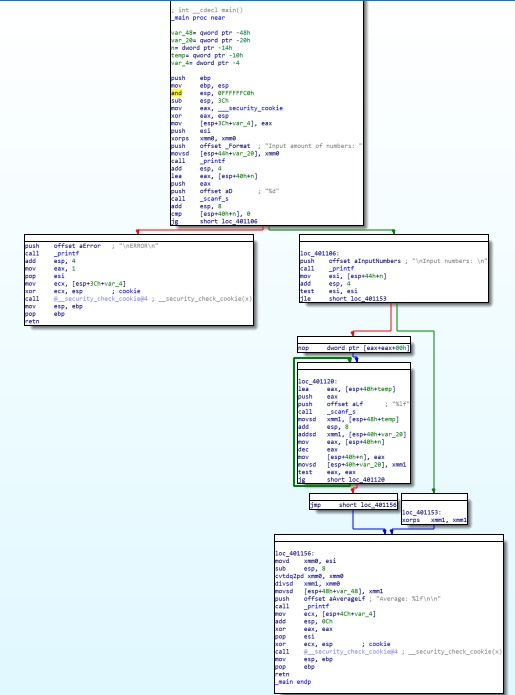
**Задание:**

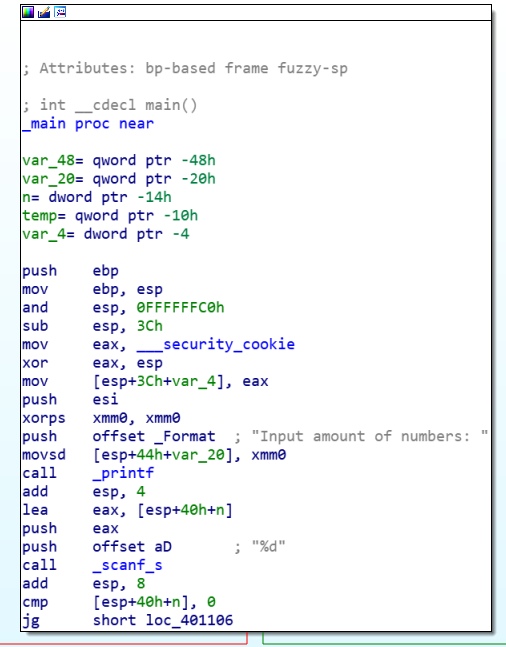
Требуется провести дизассемблирование какого-либо исполняемого файла, например, самостоятельно созданного .exe, и изучить особенности его устройства/получить какую-либо информацию, доступную только при реверс-инжиниринге.

**Исходный код исследуемой программы:**

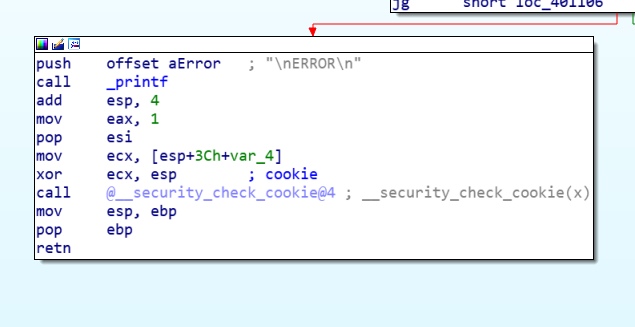


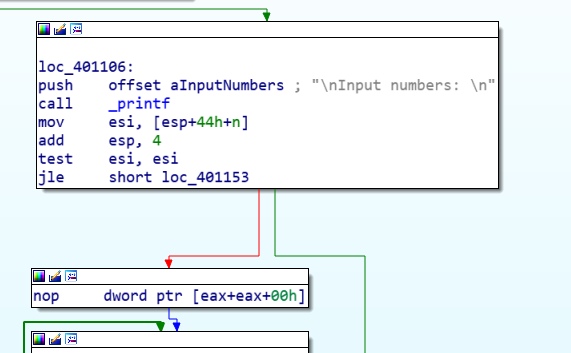
**Схема IDA Freeware**

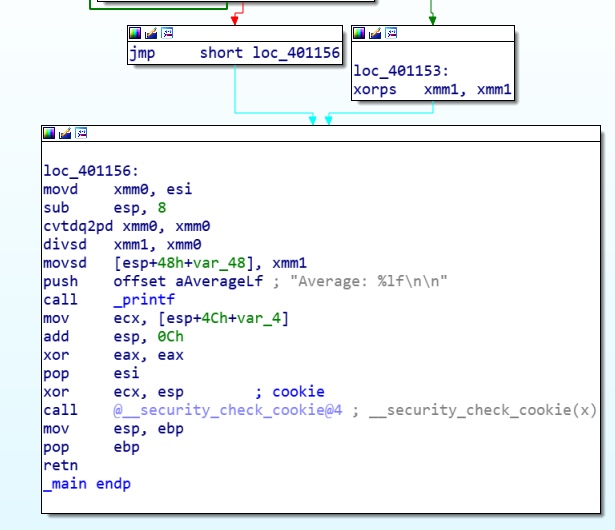


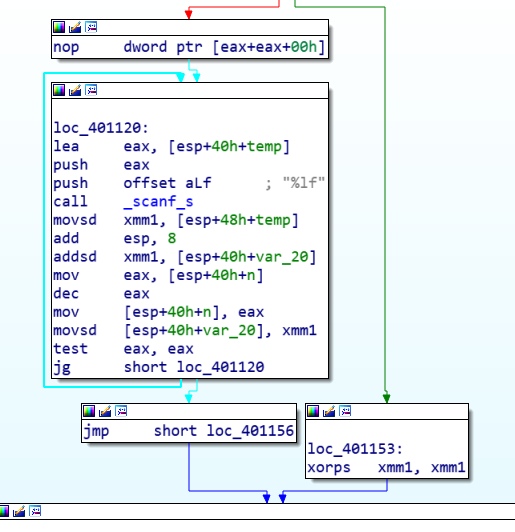
В верхнем блоке находится инициализация переменных (первые строчки). Далее идут команды вывода на экран (\_printf) приглашения (“Input amount of numbers: ”) и считывания с клавиатуры целого числа (команда \_scanf\_s). Всё передаётся через стек и после выполнения команды call стек очищается, путём увеличения вершины стека esp.

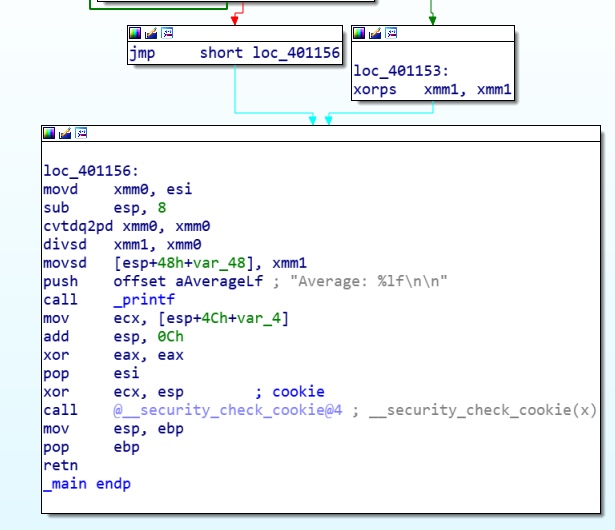
В последних двух строчках этого блока происходит сравнение значения, которое лежит по адресу [esp + 40h + n] с нулём. И если это значение больше нуля, то осуществляется короткий переход по метке loc\_401106.

Если же условие не выполнено, то начинается обработка ошибки. Выводится соответствующее сообщение с помощью функции \_printf, и выполняется выход из \_main (аналогично описанному далее).

Если же условие было выполнено, то осуществляется переход к следующему блоку. Выводится приглашение для ввода (\_printf). И значение n записывается в esi, который ниже: если значение регистра меньше или равно нулю, то делается короткий переход по метке loc\_401153. Если же нет, то программа переходит к следующему блоку.

В блоке с меткой loc\_401153 происходит обнуление регистра xmm1 с помощью команды xorps. И далее работа переходит в последующий блок (с меткой loc\_401156).

Если же регистр esi был больше нуля, то программа переходит к блоку с меткой loc\_401120. В регистр eax записывается адрес temp. C помощью функции \_scanf\_s считывается вещественное число в эту переменную, после чего она переносится в регистр xmm1. Далее к значению, хранящемуся в этом регистре, **добавляется** переменная var\_20 (в самом начале ее значение 0). Последующие три строки описывают процесс уменьшения на 1 переменной n. Дальнейшая команда movsd записывает значение регистра xmm1 в переменную var\_20. Затем проверяется регистр eax (если его значение больше, чем ноль, то переход по метке loc\_401120, то есть возврат на начало этого блока, таким образом, организован цикл)

Если же не выполнено условие для eax, то происходит короткий переход по метке loc\_401156. Команда movd заносит в xmm0 значение esi (начальное значение переменной n), которое дальше преобразуется в число с плавающей запятой. Содержимое xmm1 **делится** на xmm0 (команда divsd) и записывается в стек. Далее следует вывод на экран сообщения и вычисленного значения. В ecx записывается результат операции xor \_\_security\_cookie и значения esp (которое было в начале выполнения программы). Поэтому, если esp имеет такое же значение как вначале, то при xor с ecx результат должен быть равен \_\_security\_cookie, что и проверяется в вызываемой подпрограмме. Программа на этом завершается.

Анализируя данный код можно сделать вывод о том, что программа находит среднее арифметическое введенных чисел, причем сначала вводится их количество n. И дальше вводятся сами числа (ввод организован с помощью цикла по n). Также отлавливается ситуация, когда введенное значение n <= 0, в таком случае, программа выдает сообщение об ошибке. Что и соответствует выполняемой задаче исходной программы, приведенной в самом начале.

**Вывод**

Таким образом, реверс-инжиниринг дает возможность понять принцип работы программы, получить некоторые закрытые сведения о ее внутреннем устройстве. Также по полученной схеме можно сделать вывод о структуре, взаимосвязи различных блоков.