Министерство науки и высшего образования РФ

Федеральное государственное бюджетное образовательное учреждение

высшего образования

«Новгородский государственный университет имени Ярослава Мудрого»

Кафедра информационных технологий и систем

**Декомпиляция**

Лабораторная работа по дисциплине:

«Специальные методы информационной безопасности»

Выполнил

Студент группы №1094

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_Е.А.Менделеев

\_\_\_\_\_ \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_2021

Проверил

Преподаватель

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_П.М.Довгалюк

\_\_\_\_\_ \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_2021

**Великий Новгород**

**2021**

**Оглавление**

[**1.** **Постановка задачи** 3](#_Toc86856912)

[**2.** **Дизассемблирование бинарного файла** 3](#_Toc86856913)

[**3.** **Исходный код программы** 4](#_Toc86856914)

[**4.** **Тестирование декомпилированной и исходной программ** 5](#_Toc86856915)

[**5.** **Ответы на вопросы** 6](#_Toc86856916)

## **Постановка задачи**

Дизассемблируйте бинарный файл.

Восстановите исходный текст алгоритма на языке высокого уровня, реализуемого бинарным файлом.

Протестируйте исходную и декомпилированную программу на различных входных данных.

Ответьте на вопросы:

* Какие переменные можно выделить в программе?
* Какие функции можно выделить в программе?
* Что является результатом работы программы?

## **Дизассемблирование бинарного файла**

В качестве программы, которую необходимо дизассемблировать и восстановить её исходный код, был выбран бинарный файл под номером 13. При использовании команды **objdump -d 13** в терминале Ubuntu отображается следующий ассемблерный код файла 13:

*Листинг 1. Участок ассемблерного кода бинарного файла 13.*

…………………………….

00000000000006fa <\_Z1fii>:

6fa: 55 push %rbp

6fb: 48 89 e5 mov %rsp,%rbp

6fe: 89 7d ec mov %edi,-0x14(%rbp)

701: 89 75 e8 mov %esi,-0x18(%rbp)

704: c7 45 f8 01 00 00 00 movl $0x1,-0x8(%rbp)

70b: c7 45 fc 01 00 00 00 movl $0x1,-0x4(%rbp)

712: 8b 45 fc mov -0x4(%rbp),%eax

715: 3b 45 e8 cmp -0x18(%rbp),%eax

718: 7f 1f jg 739 <\_Z1fii+0x3f>

71a: 8b 45 f8 mov -0x8(%rbp),%eax

71d: 0f af 45 ec imul -0x14(%rbp),%eax

721: 89 45 f8 mov %eax,-0x8(%rbp)

724: 8b 0d e6 08 20 00 mov 0x2008e6(%rip),%ecx # 201010 <x>

72a: 8b 45 f8 mov -0x8(%rbp),%eax

72d: 99 cltd

72e: f7 f9 idiv %ecx

730: 89 55 f8 mov %edx,-0x8(%rbp)

733: 83 45 fc 01 addl $0x1,-0x4(%rbp)

737: eb d9 jmp 712 <\_Z1fii+0x18>

739: 8b 45 f8 mov -0x8(%rbp),%eax

73c: 5d pop %rbp

73d: c3 retq

000000000000073e <main>:

73e: 55 push %rbp

73f: 48 89 e5 mov %rsp,%rbp

742: 48 83 ec 20 sub $0x20,%rsp

746: 64 48 8b 04 25 28 00 mov %fs:0x28,%rax

74d: 00 00

74f: 48 89 45 f8 mov %rax,-0x8(%rbp)

753: 31 c0 xor %eax,%eax

755: 48 8d 3d 08 01 00 00 lea 0x108(%rip),%rdi # 864 <\_IO\_stdin\_used+0x4>

75c: b8 00 00 00 00 mov $0x0,%eax

761: e8 5a fe ff ff callq 5c0 <printf@plt>

766: 48 8d 55 ec lea -0x14(%rbp),%rdx

76a: 48 8d 45 e8 lea -0x18(%rbp),%rax

76e: 48 89 c6 mov %rax,%rsi

771: 48 8d 3d 00 01 00 00 lea 0x100(%rip),%rdi # 878 <\_IO\_stdin\_used+0x18>

778: b8 00 00 00 00 mov $0x0,%eax

77d: e8 4e fe ff ff callq 5d0 <scanf@plt>

782: 8b 55 ec mov -0x14(%rbp),%edx

785: 8b 45 e8 mov -0x18(%rbp),%eax

788: 89 d6 mov %edx,%esi

78a: 89 c7 mov %eax,%edi

78c: e8 69 ff ff ff callq 6fa <\_Z1fii>

791: 89 45 f0 mov %eax,-0x10(%rbp)

794: 8b 55 e8 mov -0x18(%rbp),%edx

797: 8b 45 ec mov -0x14(%rbp),%eax

79a: 89 d6 mov %edx,%esi

79c: 89 c7 mov %eax,%edi

79e: e8 57 ff ff ff callq 6fa <\_Z1fii>

7a3: 89 45 f4 mov %eax,-0xc(%rbp)

7a6: 8b 55 f4 mov -0xc(%rbp),%edx

7a9: 8b 45 f0 mov -0x10(%rbp),%eax

7ac: 89 c6 mov %eax,%esi

7ae: 48 8d 3d c9 00 00 00 lea 0xc9(%rip),%rdi # 87e <\_IO\_stdin\_used+0x1e>

7b5: b8 00 00 00 00 mov $0x0,%eax

7ba: e8 01 fe ff ff callq 5c0 <printf@plt>

7bf: b8 00 00 00 00 mov $0x0,%eax

7c4: 48 8b 4d f8 mov -0x8(%rbp),%rcx

7c8: 64 48 33 0c 25 28 00 xor %fs:0x28,%rcx

7cf: 00 00

7d1: 74 05 je 7d8 <main+0x9a>

7d3: e8 d8 fd ff ff callq 5b0 <\_\_stack\_chk\_fail@plt>

7d8: c9 leaveq

7d9: c3 retq

…………………………….

Листинг 1 представляет собой ассемблерный код функций (функции **main()** и **\_Z1fii()**), используемые в программе для получения выходных данных.

## **Исходный код программы**

Исходя из ассемблерного кода программы в листинге 1, исходный текст алгоритма на языке высокого уровня выглядел следующим образом:

*Листинг 2. Исходный код программы на языке высокого уровня.*

[#include](https://vk.com/im?sel=147708033&st=%23include) <stdio.h>  
[#include](https://vk.com/im?sel=147708033&st=%23include) <stdlib.h>  
[#include](https://vk.com/im?sel=147708033&st=%23include) <string.h>  
  
int \_Z1fii(int firstNumber, int secondNumber)  
{  
 int result = 1;  
 for(int i = 1; i <= firstNumber; i++)  
 {  
 result \*= secondNumber;  
 result %= 15;  
 }  
 return result;  
}  
  
int main(int argc, char \*argv[])  
{  
 int firstNumber, secondNumber;  
  
 printf("Enter two numbers: ");  
 scanf("%i %i", &firstNumber, &secondNumber);  
  
 int a = \_Z1fii(firstNumber, secondNumber);  
 int b = \_Z1fii(secondNumber, firstNumber);  
  
 printf("Result: %i, %i\n", b, a);  
  
 return 0;  
}

## **Тестирование декомпилированной и исходной программ**

В таблице 1 представлены тестовые наборы данных для декомпилированной и исходной программ.

Таблица 1. Тестовый набор данных для декомпилированной и исходной программ.

|  |  |  |  |
| --- | --- | --- | --- |
| **Исходная** | | **Декомпилированная** | |
| **Входные данные** | **Выходные данные** | **Входные данные** | **Выходные данные** |
| Enter two numbers: 2 5 | Results: 2, 10 | Enter two numbers: 2 5 | Results: 2, 10 |
| Enter two numbers: 7 4 | Results 1, 4 | Enter two numbers: 7 4 | Results 1, 4 |
| Enter two numbers: -2 5 | Results: -2, 1 | Enter two numbers: -2 5 | Results: -2, 1 |
| Enter two numbers: 123 57 | Results: 3, 3 | Enter two numbers: 123 57 | Results: 3, 3 |
| Enter two numbers: -20 -10 | Results: 1, 1 | Enter two numbers: -20 -10 | Results: 1, 1 |

## **Ответы на вопросы**

* 1. *Какие переменные можно выделить в программе?*

В функции **main()**: **int a**, находящаяся по адресу **-0x10(%rbp)** (791 адрес ассемблерного кода), **int b**, находящаяся по адресу **-0xc(%rbp)** (адрес 7а3 ассемблерного кода), **int firstNumber**, находящаяся по адресу **-0x14(%rbp)** (адрес 766 ассемблерного кода), **int secondNumber**, находящаяся по адресу -**0x18(%rbp)** (адрес 76а ассемблерного кода).

В функции **\_Z1fii()**: переменная цикла **int i**, находящаяся по адресу -**0x4(%rbp)** (адрес 70b ассемблерного кода), **int result**, находящаяся по адресу **-0x8(%rbp)** (адрес 704 ассемблерного кода), параметр функции **int firstNumber**, находящаяся по адресу **-0x18(%rbp)** (адрес 701 ассемблерного кода), параметр функции **int secondNumber**, находящаяся по адресу -**0x14(%rbp)** (адрес 6fe ассемблерного кода).

* 1. *Какие функции можно выделить в программе?*

В программе можно выделить функции **main()** и **\_Z1fii()**. В функции **main()** происходит ввод входных данных для работы программы, а также в ней вызывается два раза функция **\_Z1fii()**, в которой происходит следующий алгоритм.

В функцию принимаются два параметра **firstNumber**, **secondNumber**. Число **secondNumber** возводится в степень **firstNumber**, а потом результат возведения в степень делится на 15 и берётся остаток от деления.

* 1. *Что является результатом работы программы?*

Результатом работы программы являются два числа, первое из которых **secondNumber** возведено в степень **firstNumber** и поделено с остатком на 15, а второе из которых **firstNumber** возведено в степень **secondNumber** и поделено с остатком на 15.