Отчёт по лабораторной работе №8

Дисциплина: Архитектура компьютера

Гасанова Шакира Чингизовна

Содержание

1 Цель работы .............................................................................................………………………………............. 3

2 Задание .....................................................................................................…………………………………............ 4

3 Теоретическое введение ..............................................................…………………………...................... 5

4 Выполнение лабораторной работы

4.1 Реализация циклов в NASM...................................................................………………………..............6

4.2 Обработка аргументов командной строки.................................……………......................12

4.3 Выполнение заданий для самостоятельной работы.................……….......................18

5 Выводы ........................................................................................................…………………………………........22

6 Источники ....................................................................................................………………………………........23

# 1 Цель работы

Целью данной работы является приобретение навыков написания программ с использованием циклов и обработкой аргументов командной строки.

# 2 Задание

1. Реализация циклов в NASM.
2. Обработка аргументов командной строки.
3. Самостоятельное написание программы по материалам лабораторной работы.

# 3 Теоретическое введение

Стек — это структура данных, организованная по принципу LIFO («Last In — First Out» или «последним пришёл — первым ушёл»). Стек является частью архитектуры процессора и реализован на аппаратном уровне. Для работы со стеком в процессоре есть специальные регистры (ss, bp, sp) и команды. Основной функцией стека является функция сохранения адресов возврата и передачи аргументов при вызове процедур. Кроме того, в нём выделяется память для локальных переменных и могут временно храниться значения регистров.

# 4 Выполнение лабораторной работы

**4.1 Реализация циклов в NASM** Создаю каталог для программ лабораторной работы №8 и файл для написания программы(рис. 1).

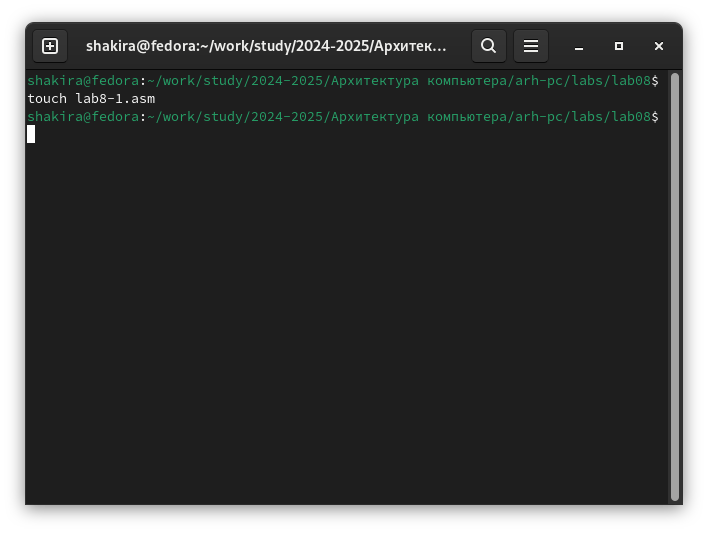


Рис. 1: 1

Копирую в созданный файл программу из листинга (рис. 2).

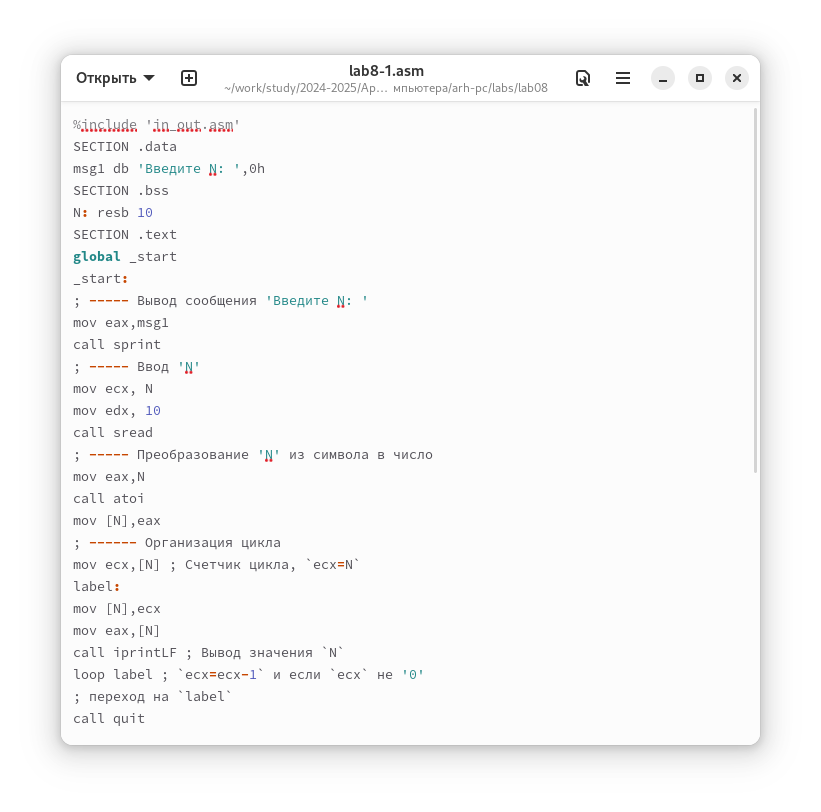


Рис. 2: 2

Запускаю программу, она показывает работу циклов в NASM (рис. 3).

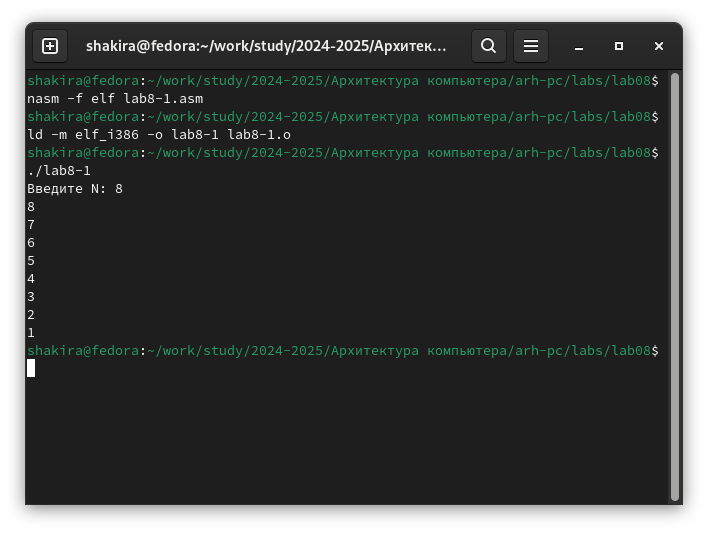


Рис. 3: 3

Заменяю программу изначальную так, что в теле цикла я изменяю значение регистра ecx (рис. 4).

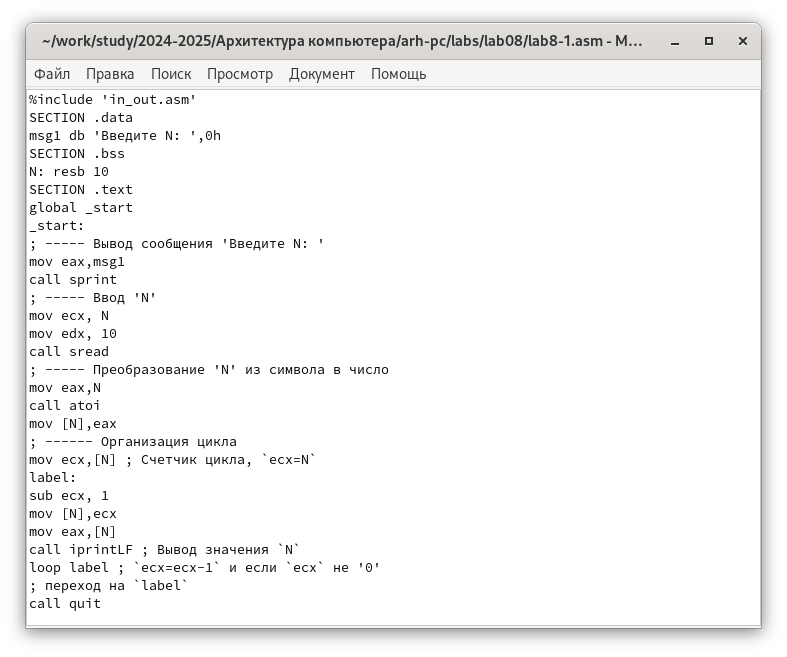


Рис. 4: 4

Из-за того, что теперь регистр ecx на каждой итерации уменьшается на 2 значения, количество итераций уменьшается вдвое (рис. 5).

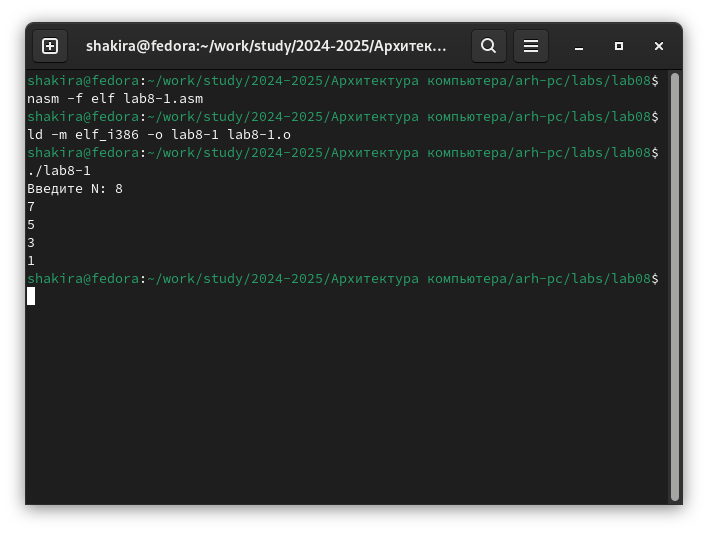


Рис. 5: 5

Добавляю команды push и pop в программу (рис. 6).

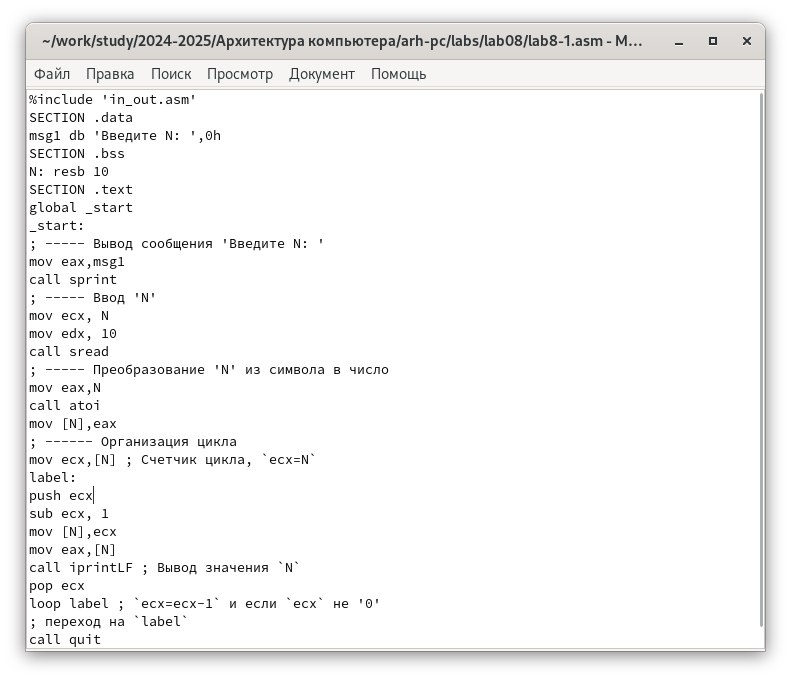


Рис. 6: 6

Теперь количество итераций совпадает введенному N, но произошло смещение выводимых чисел на -1 (рис. 7).

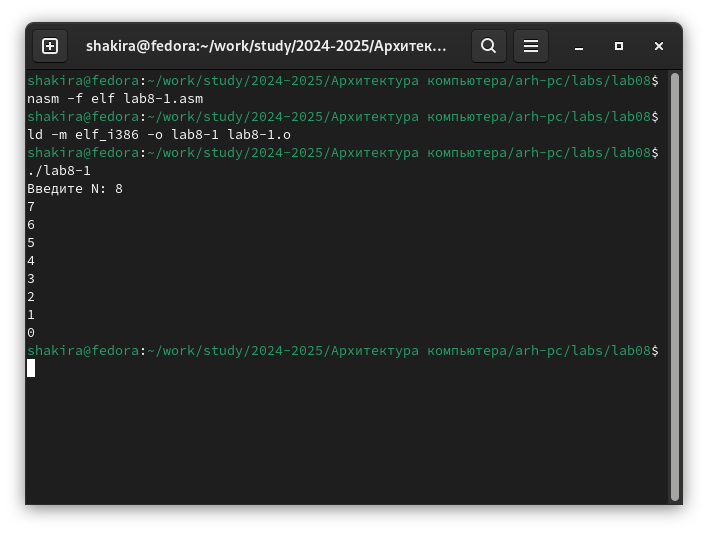


Рис. 7: 7

**4.2 Обработка аргументов командной строки** Создаю новый файл для программы и копирую в него код из следующего листинга (рис. 8).

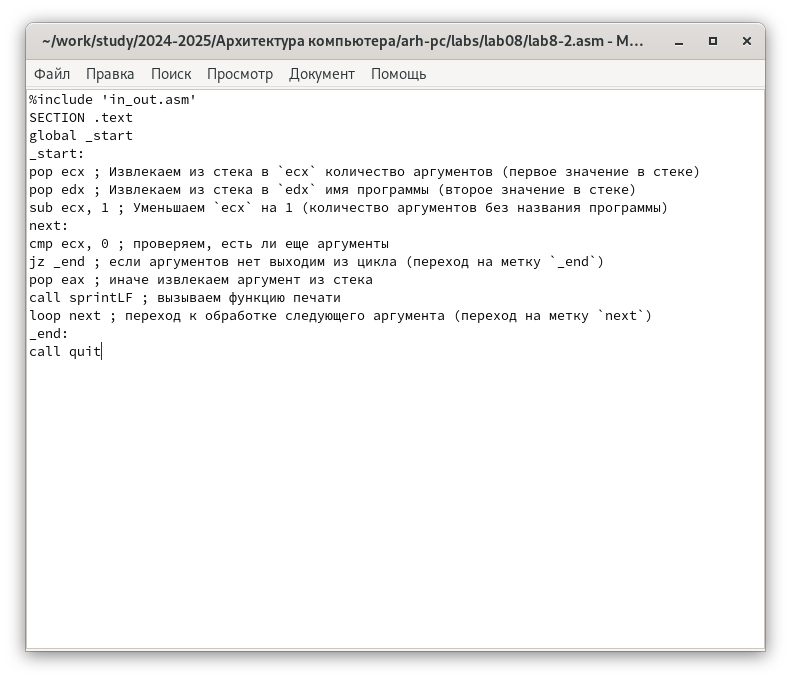


Рис. 8: 8

Компилирую программу и запускаю, указав аргументы. Программой было обратоно то же количество аргументов, что и было введено (рис. 9).

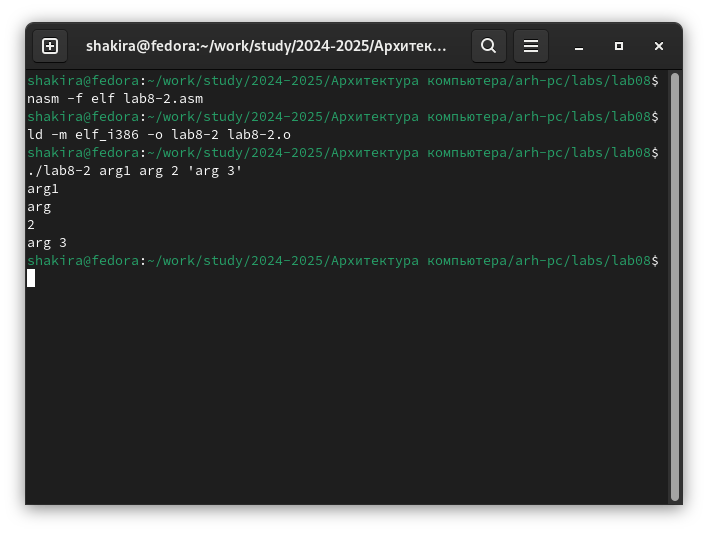


Рис. 9: 9

Создаю новый файл для программы и копирую в него код из третьего листинга (рис. 10).



Рис. 10: 10

Компилирую программу и запускаю, указав в качестве аргументов некоторые числа, программа их складывает (рис. 11).

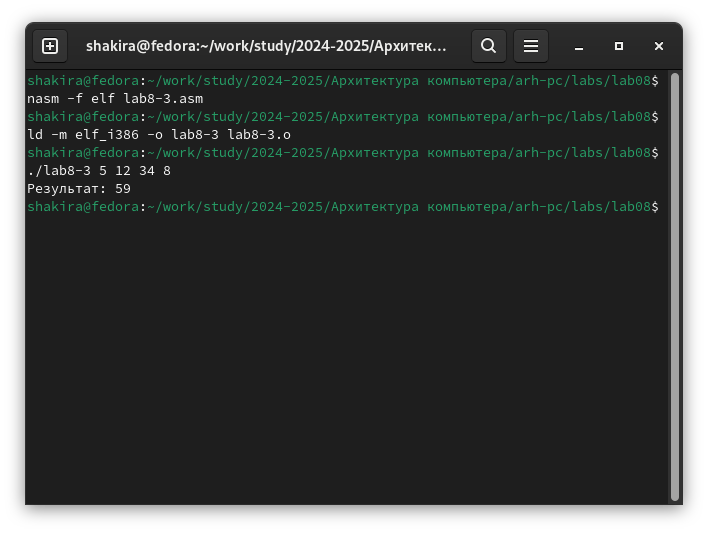


Рис. 11: 11

Изменяю поведение программы так, чтобы указанные аргументы она умножала, а не складывала (рис. 12).

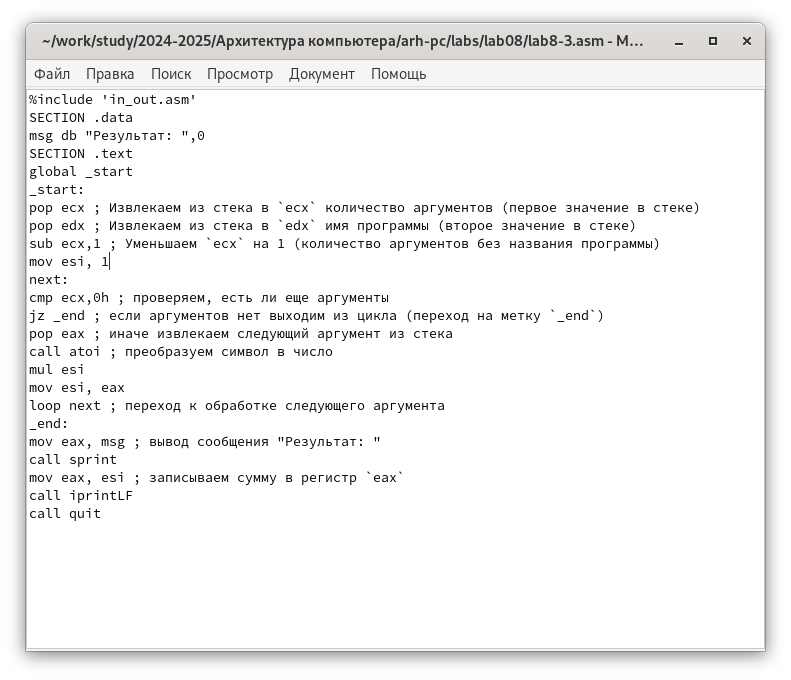


Рис. 12: 12

Программа действительно теперь умножает данные на вход числа (рис. 13).

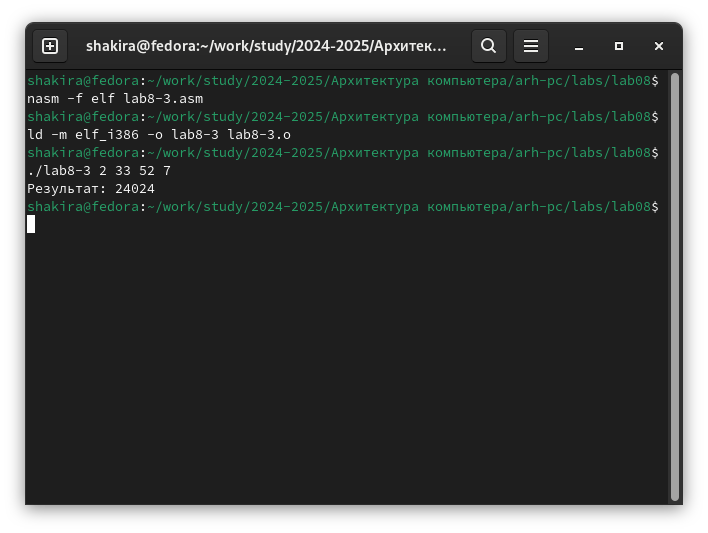


Рис. 13: 13

**4.3 Выполнение заданий для самостоятельной работы** Пишу программму, которая будет находить сумму значений для функции f(x) = 2(x – 1), которая соответсвует 4 варианту (рис. 14).

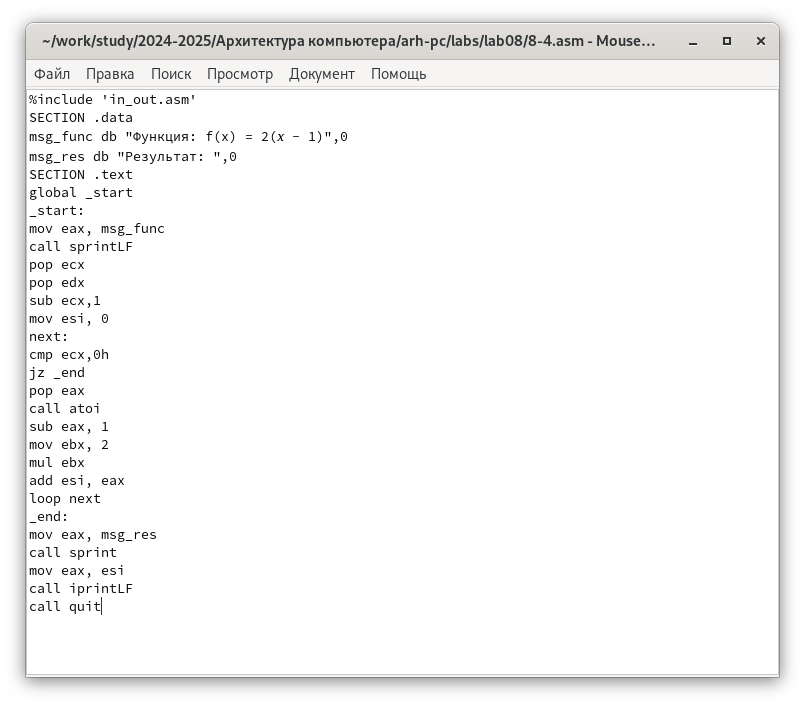


Рис. 14: 14

Код программы:

%include 'in\_out.asm'

SECTION .data

msg\_func db "Функция: f(x) = 2( − 1)",0

msg\_res db "Результат: ",0

SECTION .text

global \_start

\_start:

mov eax, msg\_func

call sprintLF

pop ecx

pop edx

sub ecx,1

mov esi, 0

next:

cmp ecx,0h

jz \_end

pop eax

call atoi

sub eax, 1

mov ebx, 2

mul ebx

add esi, eax

loop next

\_end:

mov eax, msg\_res

call sprint

mov eax, esi

call iprintLF

call quit

Проверяю работу программы, указав в качестве аргумента несколько чисел (рис. 15).

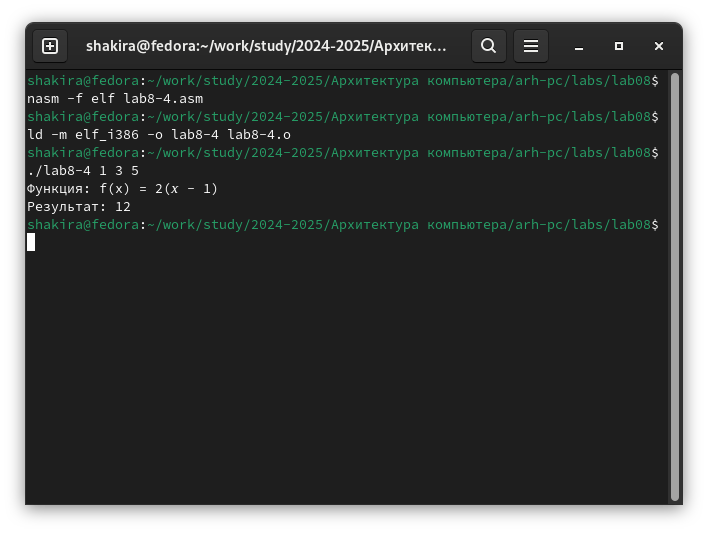


Рис. 15: 15

# 5 Выводы

В ходе выполнения этой лабораторной работы я приобрела навыки написания программ с использованием циклов а также научилась обрабатывать аргументы командной строки.

# Список литературы

1. [Архитектура ЭВМ (rudn.ru)](https://esystem.rudn.ru/)