Отчёт по лабораторной работе №8

Дисциплина: архитектура компьютеров

Симонова Виктория Игоревна

Содержание

# 1 Цель работы

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. Задание для самостоятельной работы.

# 3 Теоретическое введение

## 3.1 Организация стека

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. На рис. 8.1 показана схема организации стека в процессоре. Стек имеет вершину, адрес последнего добавленного элемента, который хранится в регистре esp (указатель стека). Противоположный конец стека называется дном. Значение, помещённое в стек последним, извлекается первым. При помещении значения в стек указатель стека уменьшается, а при извлечении — увеличивается. Для стека существует две основные операции: • добавление элемента в вершину стека (push); • извлечение элемента из вершины стека (pop).

# 4 Выполнение лабораторной работы

## 4.1 Реализация циклов в NASM

Создаю каталог lab08 для файлов лабораторной работы и файл lab8-1.asm (рис. [[1](#fig:001)]).

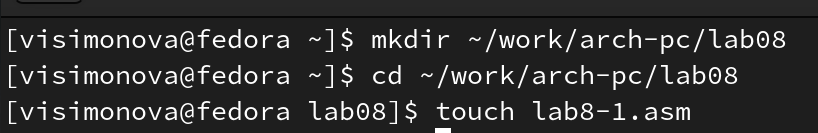


Figure 1: Создание каталога и файла

Ввожу текст прграммы из листинга в файл (рис. [[2](#fig:002)]).

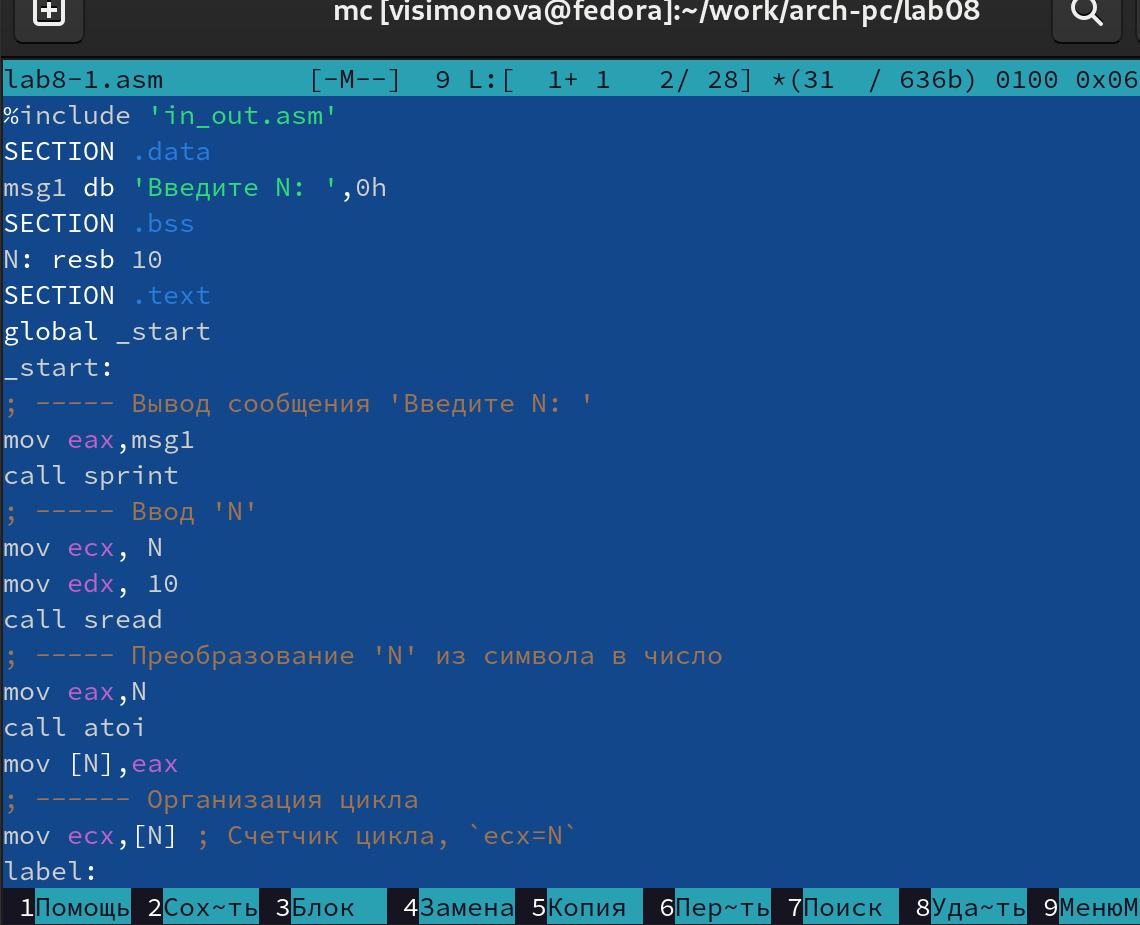


Figure 2: Программа для вывода значения регистра

Создаю исполняемый файл и проверяю его работу. (рис. [[3](#fig:003)]).

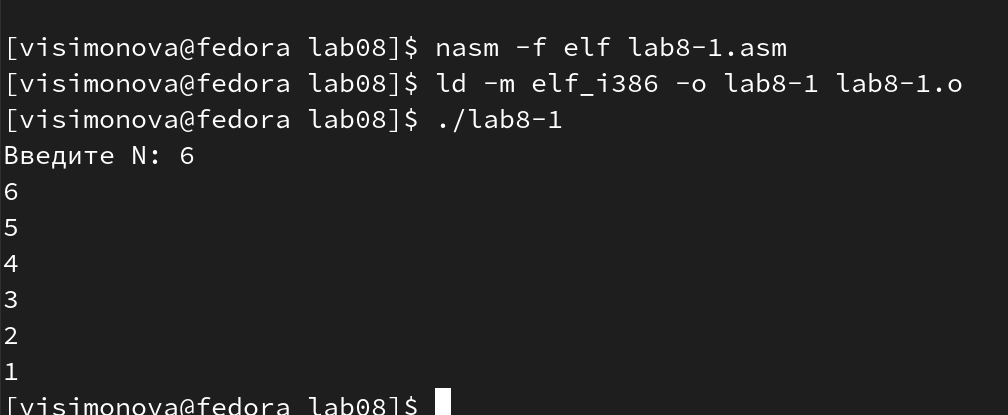


Figure 3: Запускаю исполняемый файл

Программа по порядку выводит введённое с клавиатуры значение N, которое каждый раз уменьшается на 1(инструкция loop), число проходов цикла соответсятвует значению N.

Корректирую текст программы (рис. [[4](#fig:004)]).

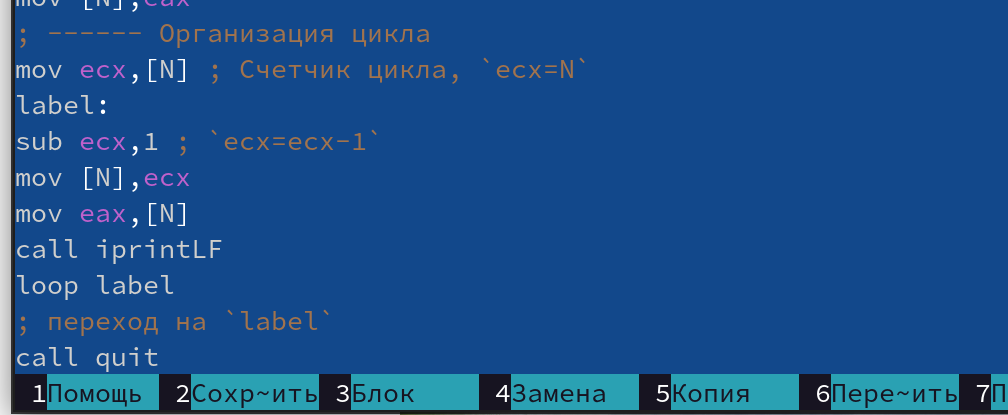


Figure 4: Корректирую текст программы

Создаю исполняемый файл и проверяю его работу. (рис. [[5](#fig:005)]).

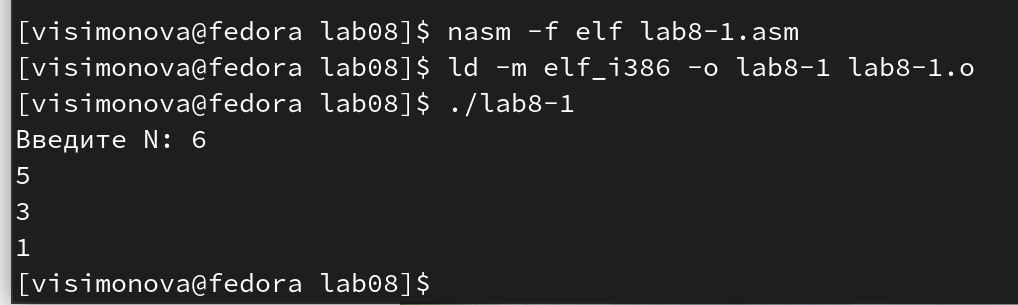


Figure 5: Запускаю исполняемый файл

Число проходов цикла не соответствует введённому значению N (N принимает значения через 1, тк в теле цикла добавлено изменение знаяения регистра ecx).

Изсменяю программу, добавляя команду push(испозуем стек) (рис. [[6](#fig:006)]).

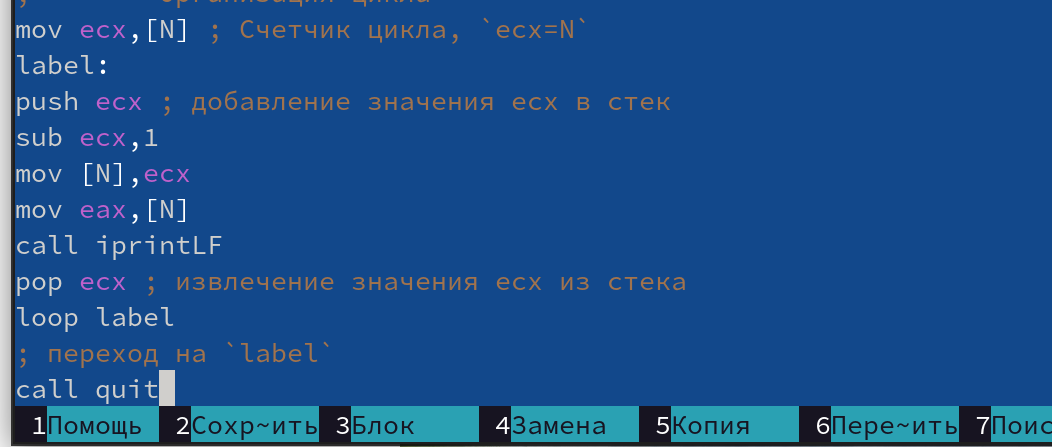


Figure 6: Изменяю файл второй раз

Создаю исполняемый файл и проверяю его работу. (рис. [[7](#fig:007)]).

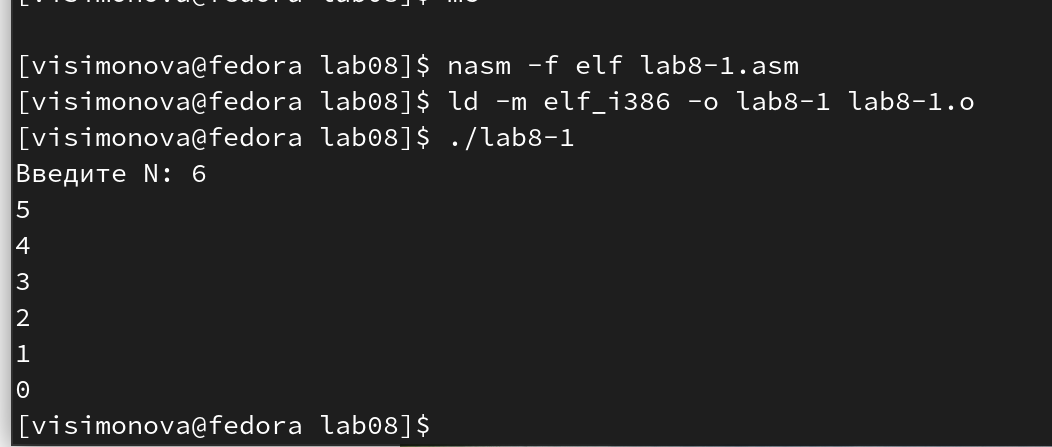


Figure 7: Запускаю исполняемый файл

Число проходов цикла соответствует введённому значению N и выводит числа от N-1 до 0 включительно.

## 4.2 Обработка аргументов командной строки

Создаю файл lab8-2.asm (рис. [[8](#fig:008)]).

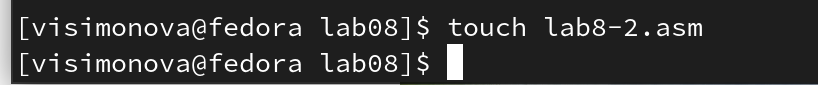


Figure 8: Создаю файл

Ввожу в файл текст программы, которая выводит на экран аргументы командной строки (рис. [[9](#fig:009)]).

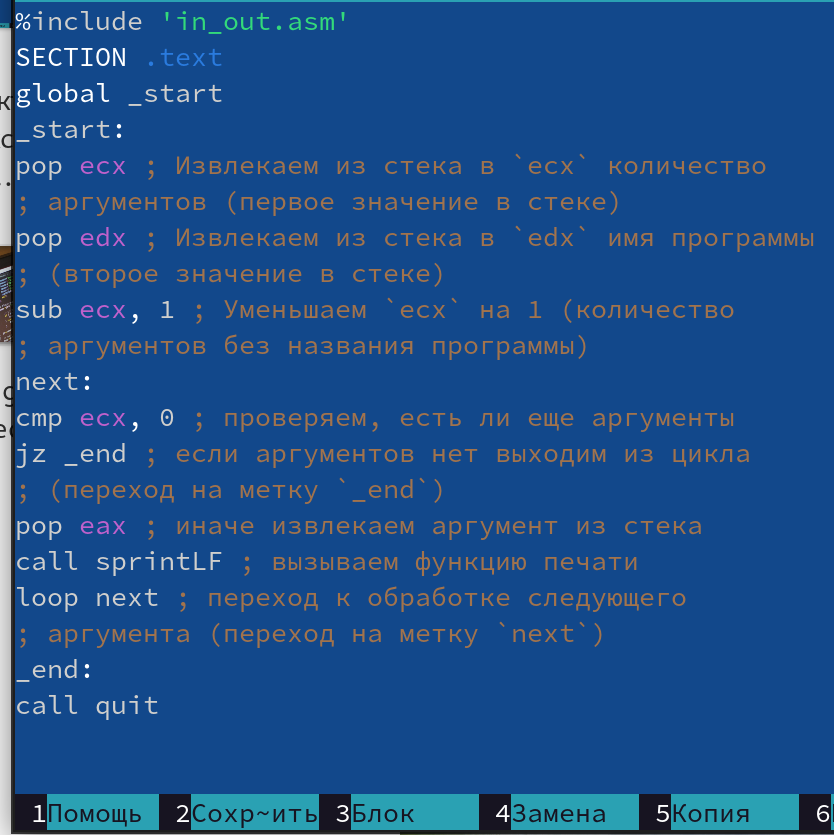


Figure 9: Ввожу код

Создаю исполняемый файл и проверяю его работу. (рис. [[10](#fig:010)]).

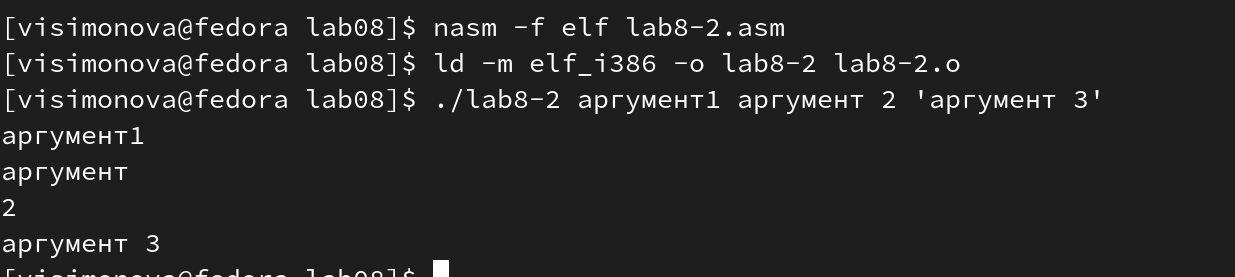


Figure 10: Запускаю исполняемый файл

Программа обработала 4 аргумента, т.к. цифра “2” не была взята в кавычки вместе со словом “аргумент”, в отличие от ‘аргумент 3’, который программа восприняла как один аргумент.

Создаю файл lab8-3.asm (рис. [[11](#fig:011)]).

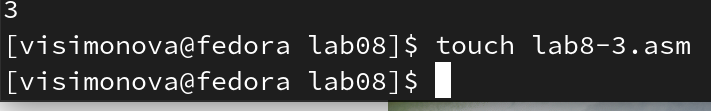


Figure 11: Создаю файл

Ввожу в файл текст программы, которая выводит на экран сумму чисел, которые введены как аргументы командной строки (рис. [[12](#fig:012)]).

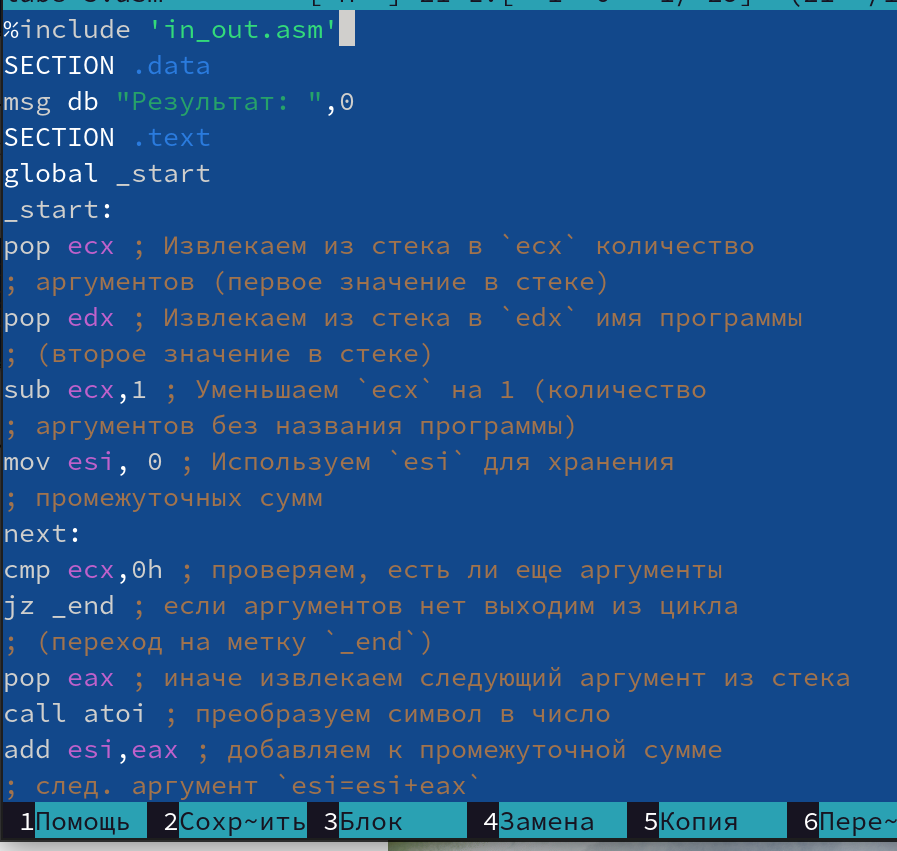


Figure 12: Ввожу код

Создаю исполняемый файл и проверяю его работу. (рис. [[13](#fig:013)]).

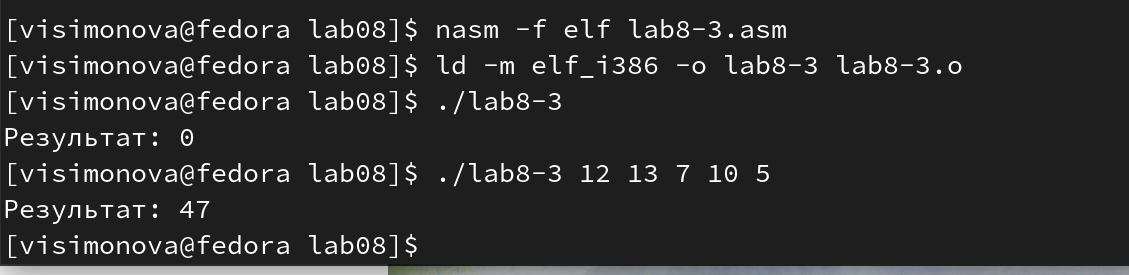


Figure 13: Запускаю исполняемый файл

Изменяю программу так, чтобы она вычислла произведеник аргументов,введённых из командной строки (рис. [[14](#fig:014)]).

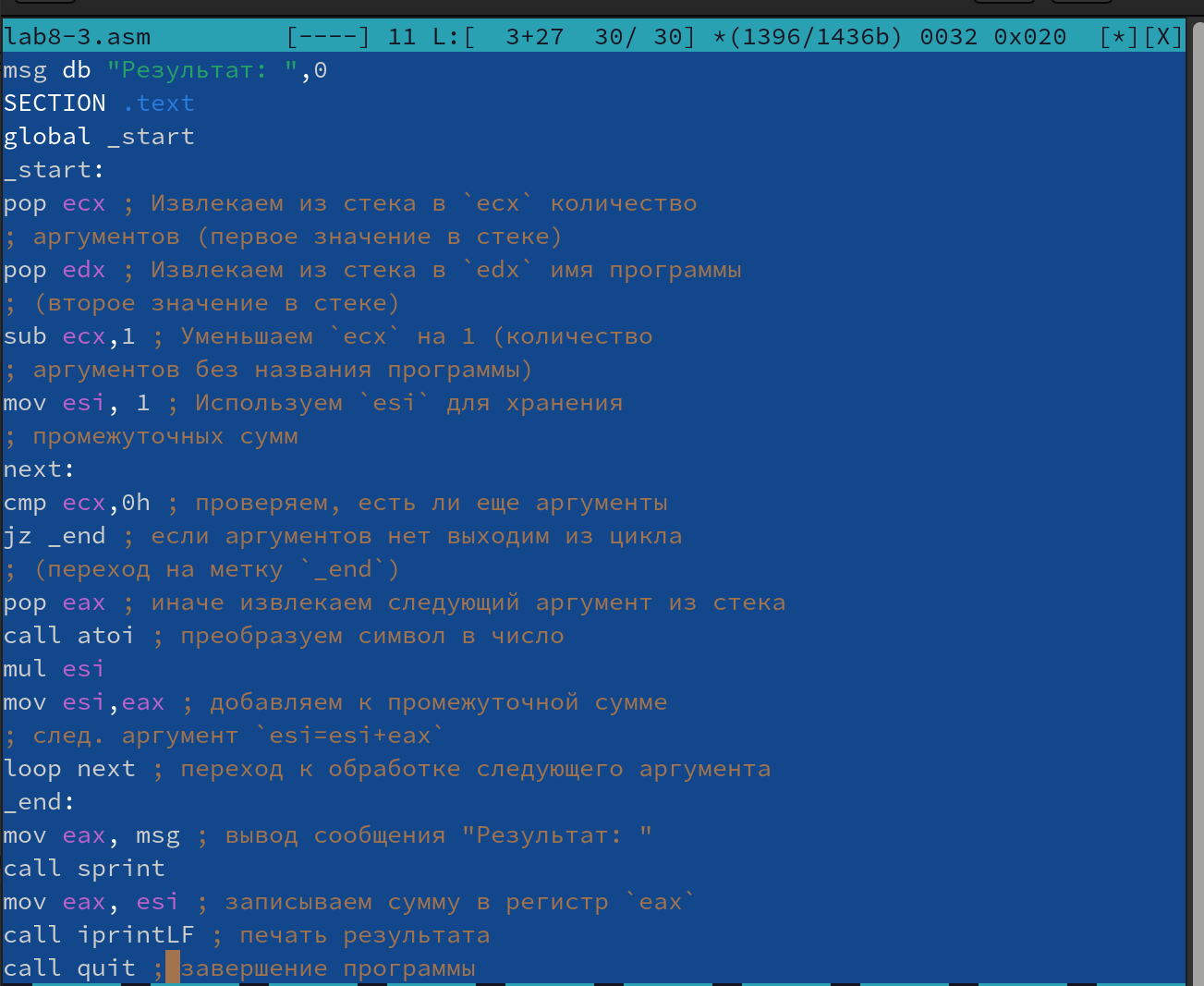


Figure 14: Изменяю код

Создаю исполняемый файл и проверяю его работу. (рис. [[15](#fig:015)]).

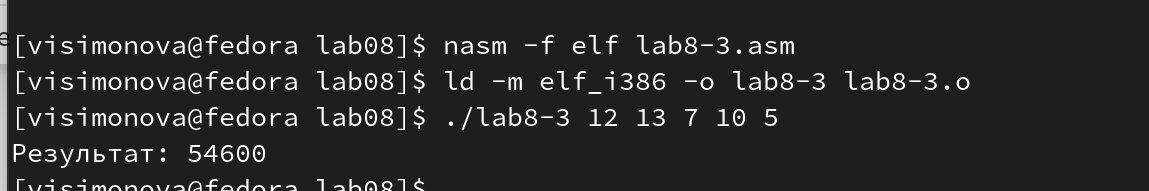


Figure 15: Запускаю исполняемый файл

## 4.3 Задание для самостоятельной работы

Пишу текст программы для вычисления суммы значений функции f(x)=12\*x-7 в соответствии с моим номером варианта (13) для x = x1, x2, …, xn. Значения xi передаются как аргументы (рис. [[16](#fig:016)]).

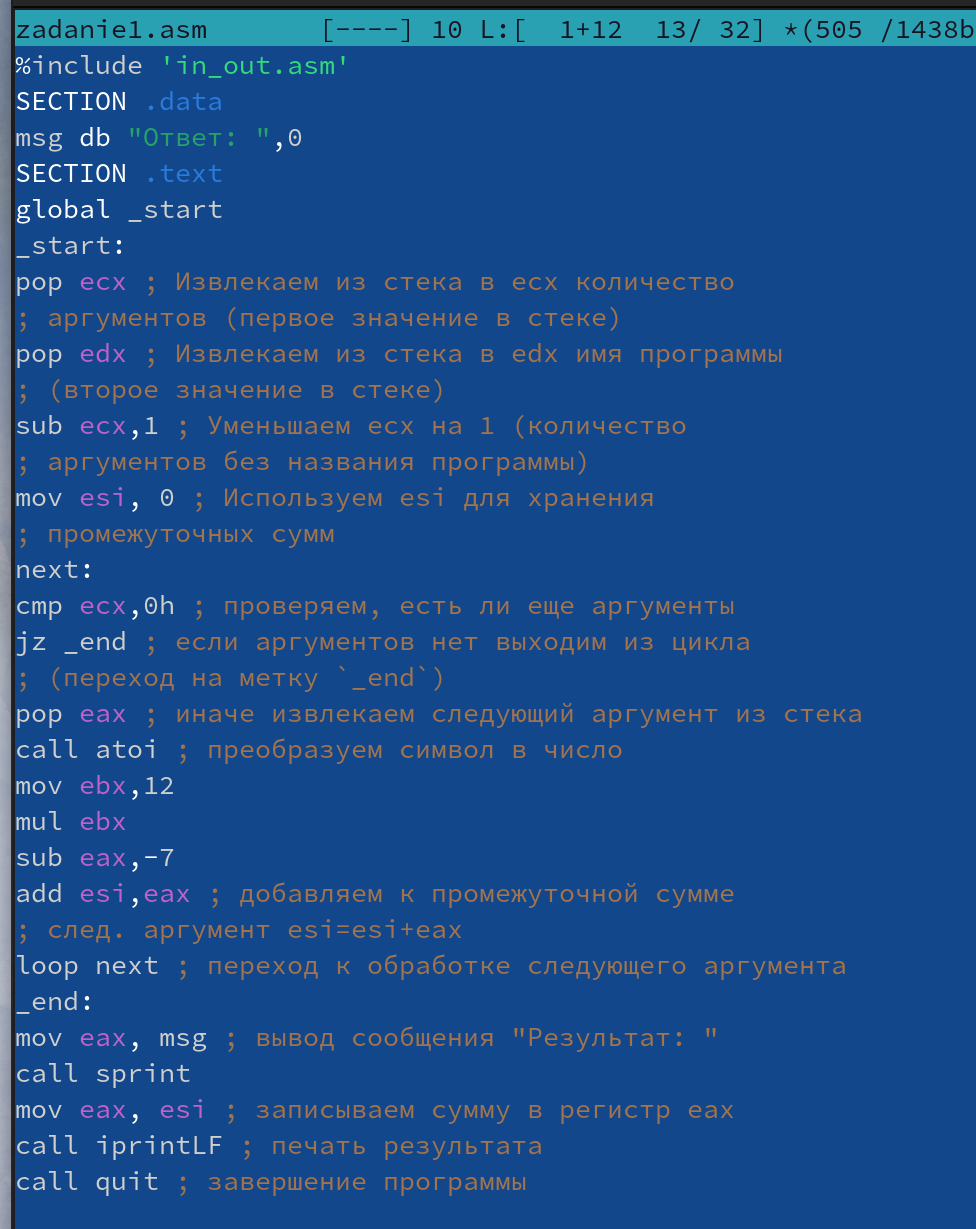


Figure 16: Текст программы

Текст программы:

%include 'in\_out.asm'  
SECTION .data  
msg db "Ответ: ",0  
SECTION .text  
global \_start  
\_start:  
pop ecx ; Извлекаем из стека в ecx количество  
; аргументов (первое значение в стеке)  
pop edx ; Извлекаем из стека в edx имя программы  
; (второе значение в стеке)  
sub ecx,1 ; Уменьшаем ecx на 1 (количество  
; аргументов без названия программы)  
mov esi, 0 ; Используем esi для хранения  
; промежуточных сумм  
next:  
cmp ecx,0h ; проверяем, есть ли еще аргументы  
jz \_end ; если аргументов нет выходим из цикла  
; (переход на метку `\_end`)  
pop eax ; иначе извлекаем следующий аргумент из стека  
call atoi ; преобразуем символ в число  
mov ebx,12  
mul ebx  
sub eax,-7  
add esi,eax ; добавляем к промежуточной сумме  
; след. аргумент esi=esi+eax  
loop next ; переход к обработке следующего аргумента  
\_end:  
mov eax, msg ; вывод сообщения "Результат: "  
call sprint  
mov eax, esi ; записываем сумму в регистр eax  
call iprintLF ; печать результата  
call quit ; завершение программы

Создаю исполняемый файл и проверяю его работу для несколькиз наборов xi. (рис. [[17](#fig:017)]).

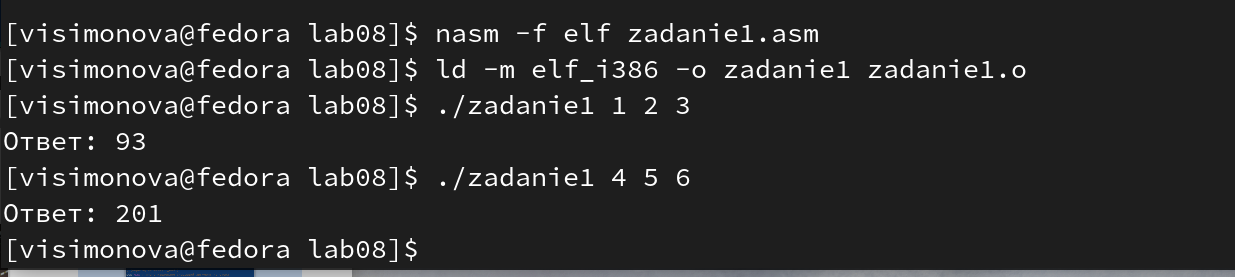


Figure 17: Запускаю исполняемый файл

# 5 Выводы

Приобрела навыки написания программ с использованием циклов и обработкой аргументов командной строки.

# Список литературы

[Лабораторная работа №8](https://esystem.rudn.ru/pluginfile.php/2089095/mod_resource/content/0/%D0%9B%D0%B0%D0%B1%D0%BE%D1%80%D0%B0%D1%82%D0%BE%D1%80%D0%BD%D0%B0%D1%8F%20%D1%80%D0%B0%D0%B1%D0%BE%D1%82%D0%B0%20%E2%84%968.%20%D0%9F%D1%80%D0%BE%D0%B3%D1%80%D0%B0%D0%BC%D0%BC%D0%B8%D1%80%D0%BE%D0%B2%D0%B0%D0%BD%D0%B8%D0%B5%20%D1%86%D0%B8%D0%BA%D0%BB%D0%B0.%20%D0%9E%D0%B1%D1%80%D0%B0%D0%B1%D0%BE%D1%82%D0%BA%D0%B0%20%D0%B0%D1%80%D0%B3%D1%83%D0%BC%D0%B5%D0%BD%D1%82%D0%BE%D0%B2%20%D0%BA%D0%BE%D0%BC%D0%B0%D0%BD%D0%B4%D0%BD%D0%BE%D0%B9%20%D1%81%D1%82%D1%80%D0%BE%D0%BA%D0%B8..pdf)