Отчёт по лабораторной работе №8

Дисциплина: архитектура компьютеров и операционные системы

Глущенко Евгений Игоревич

# 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 Выполнение лабораторной работы

## 3.1 Реализация циклов в NASM

Создаю каталог для лабораторной работы № 8, перехожу в него и создаю файл lab8-1.asm. (рис. [15](#fig:001)).

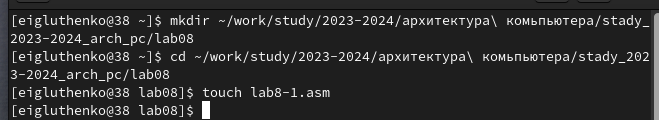


Рисунок 1: Создание файлов для лабораторной работы

Ввожу в файл lab8-1.asm текст программы. (рис. [15](#fig:001)).

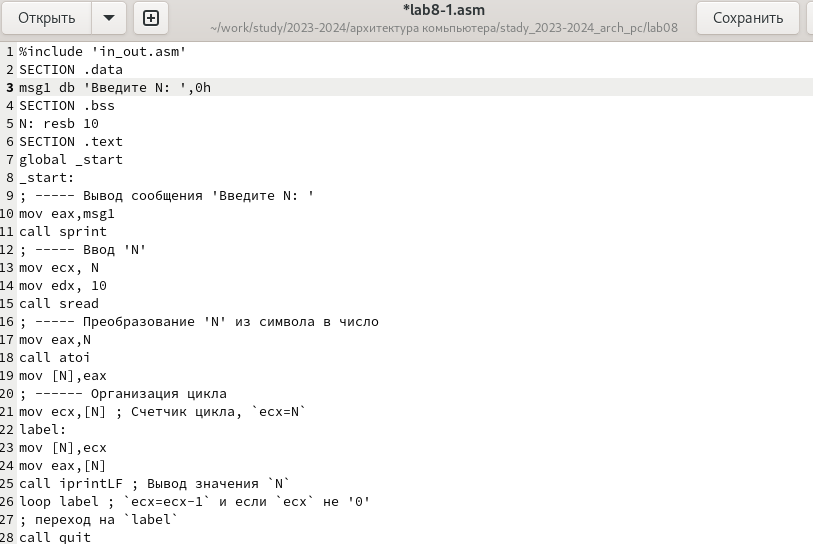


Рисунок 2: Ввод текста из листинга 8.1

Создаю исполняемый файл и проверяю его работу. (рис. [15](#fig:001)).

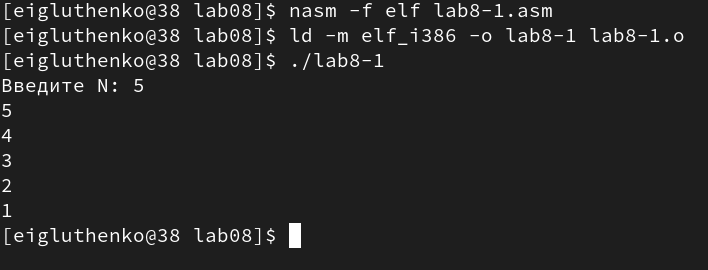


Рисунок 3: Запуск исполняемого файла

Данная программа выводит числа от N до 1 включительно.

Изменяю текст программы, добавив изменение значения регистра ecx в цикле. (рис. [15](#fig:001)).

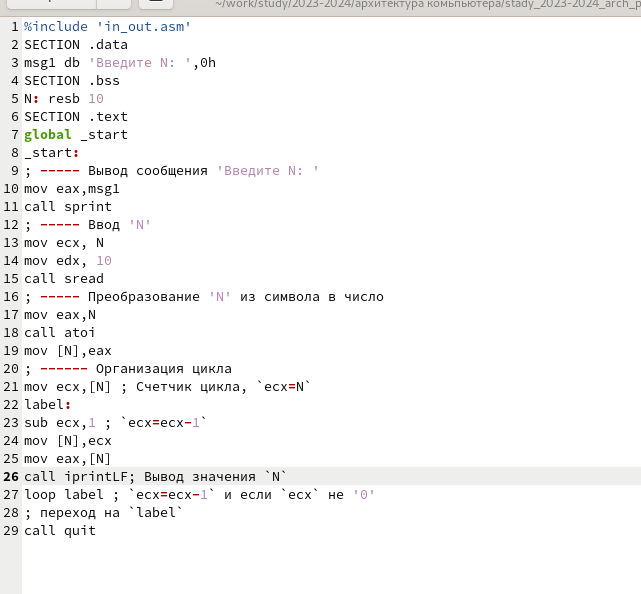


Рисунок 4: Изменение текста программы

Создаю исполняемый файл и проверяю его работу. (рис. [15](#fig:001)).



Рисунок 5: Запуск обновленной программы

В данном случае число проходов цикла не соответствует введенному с клавиатуры значению.

Вношу изменения в текст программы, добавив команды push и pop для сохранения значения счетчика цикла loop. (рис. [15](#fig:001)).

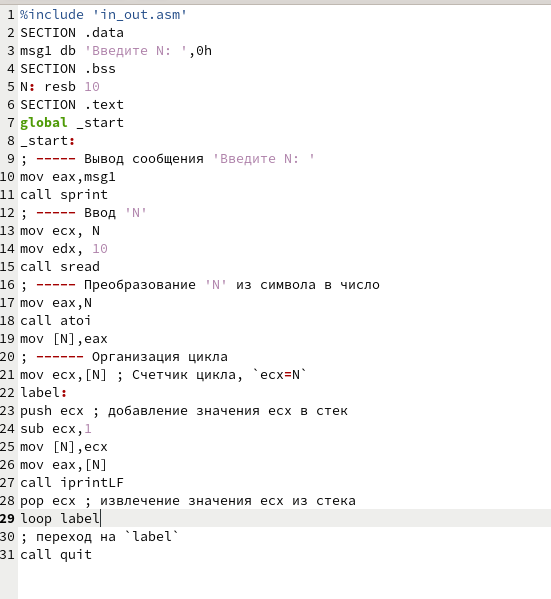


Рисунок 6: Изменение текста программы

Создаю исполняемый файл и проверяю его работу.(рис. [15](#fig:001)).

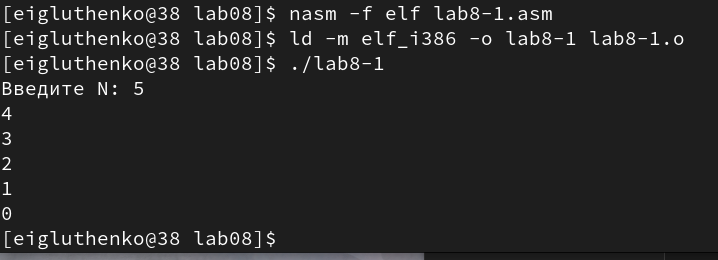


Рисунок 7: Запуск исполняемого файла

В данном случае число проходов цикла соответствует введенному с клавиатуры значению и выводит числа от N-1 до 0 включительно.

## 3.2 Обработка аргументов командной строки

Создаю файл lab8-2.asm в каталоге и ввожу в него текст программы. (рис. [15](#fig:001)).

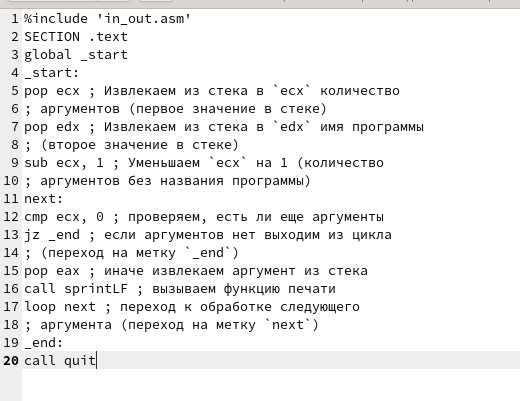


Рисунок 8: Ввод текста программы из листинга 8.2

Создаю исполняемый файл и запускаю его, указав нужные аргументы. (рис. [15](#fig:001)).

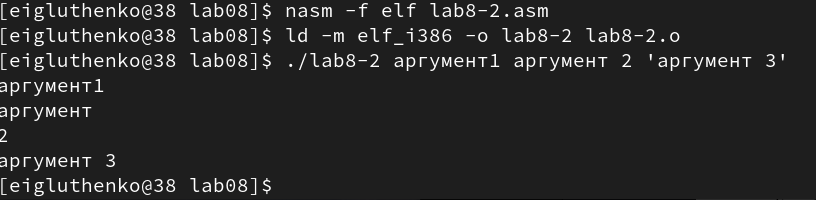


Рисунок 9: Запуск исполняемого файла

Программа вывела 4 аргумента, так как аргумент 2 не взят в кавычки, в отличии от аргумента 3, поэтому из-за пробела программа считывает “2” как отдельный аргумент.

Рассмотрим пример программы, которая выводит сумму чисел, которые передаются в программу как аргументы. Создаю файл lab8-3.asm в каталоге и ввожу в него текст программы. (рис. [15](#fig:001)).

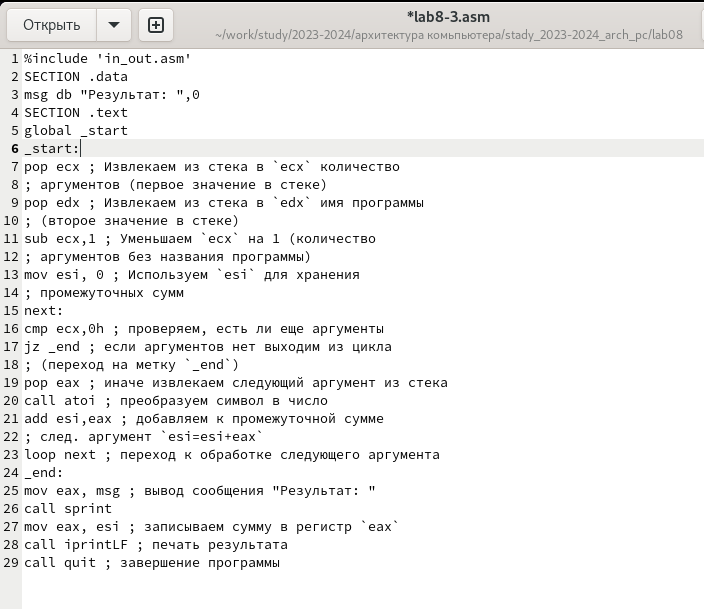


Рисунок 10: Ввод текста программы из листинга 8.3

Создаю исполняемый файл и запускаю его, указав аргументы. (рис. [15](#fig:001)).

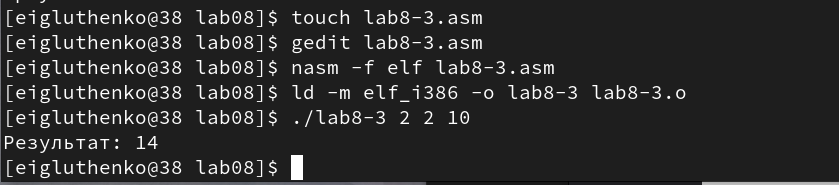


Рисунок 11: Запуск исполняемого файла

Изменяю текст программы из листинга 8.3 для вычисления произведения аргументов командной строки. (рис. [15](#fig:001)).

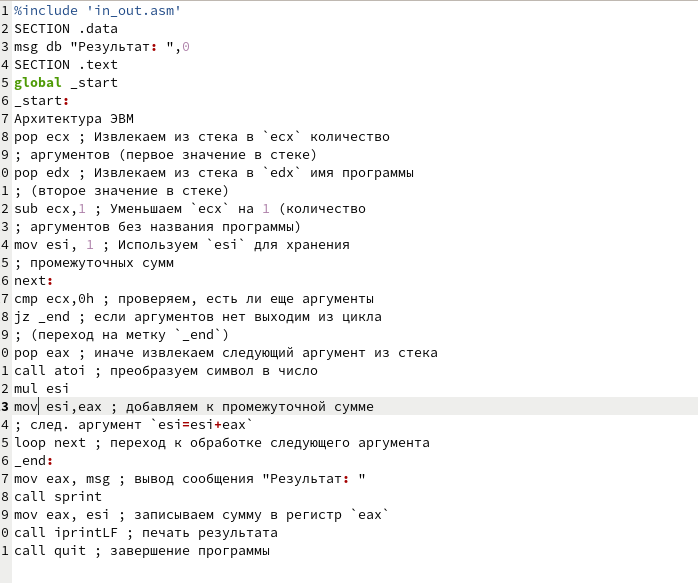


Рисунок 12: Изменение текста программы

Создаю исполняемый файл и запускаю его, указав аргументы. (рис. [15](#fig:001)).



Рисунок 13: Запуск исполняемого файла

## 3.3 Задание для самостоятельной работы

Пишу текст программы, которая находит сумму значений функции f(x) = 5\*(2 + x) в соответствии с моим номером варианта (10). (рис. [15](#fig:001)).

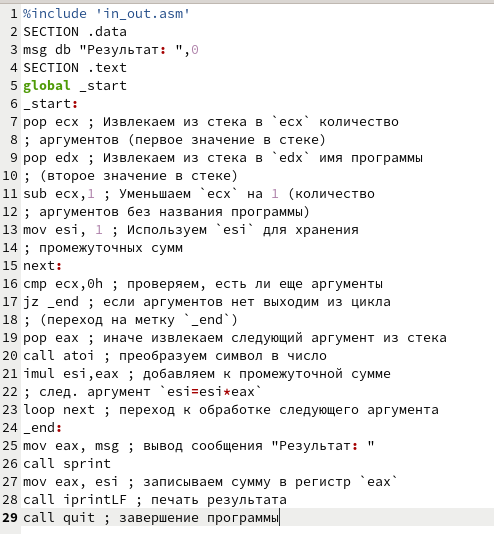


Рисунок 14: Текст программы

Создаю исполняемый файл и проверьте его работу на нескольких наборах x = x1, x2, …, xn. (рис. [15](#fig:001)).

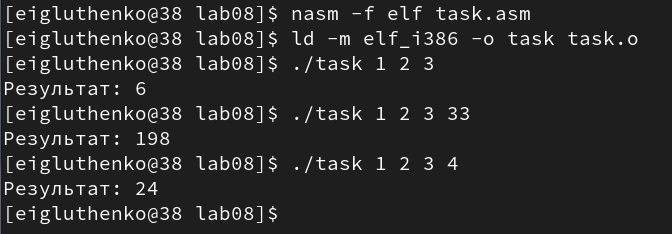


Рисунок 15: Запуск исполняемого файла и проверка его работы

Программа работает корректно.

Текст программы:

%include 'in\_out.asm'

SECTION .data

msg db "Результат: ",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

add eax,2

imul eax,3

add esi,eax ; добавляем к промежуточной сумме

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

# 4 Выводы

Благодаря данной лабораторной работе я получил навыки написания программ с использованием циклов и обработкой аргументов командной строки.