Отчёт по лабораторной работе 8

Программирование цикла. Обработка аргументов командной строки.

Ахмед Кусей

Содержание

# 1 Цель работы

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

# 2 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.

Для стека существует две основные операции:

* добавление элемента в вершину стека (push);
* извлечение элемента из вершины стека (pop).

# 3 Выполнение лабораторной работы

## 3.1 Hеализация циклов в NASM

Создал каталог для программам лабораторной работы № 8 и файл lab8-1.asm

При реализации циклов в NASM с использованием инструкции loop необходимо помнить о том, что эта инструкция использует регистр ecx в качестве счетчика и на каждом шаге уменьшает его значение на единицу. В качестве примера рассмотрим программу, которая выводит значение регистра ecx.

Написал в файл lab8-1.asm текст программы из листинга 8.1. Создал исполняемый файл и проверил его работу.

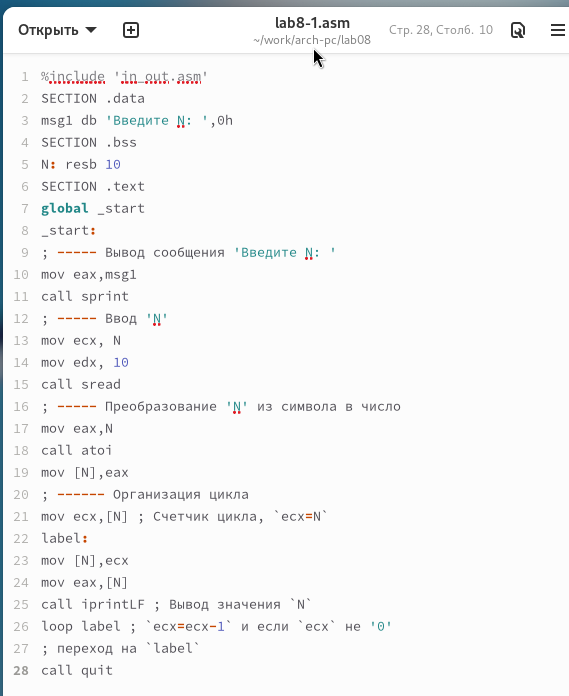


Рис. 1: Программа в файле lab8-1.asm

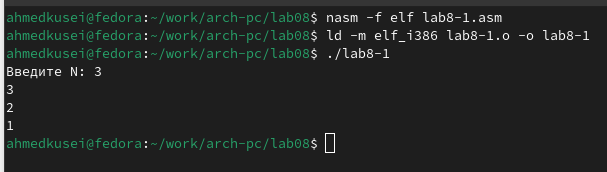


Рис. 2: Запуск программы lab8-1.asm

Данный пример показывает, что использование регистра ecx в теле цилка loop может привести к некорректной работе программы. Изменил текст программы добавив изменение значение регистра ecx в цикле.

Программа запускает бесконечный цикл при нечетном N и выводит только нечетные числа при четном N.

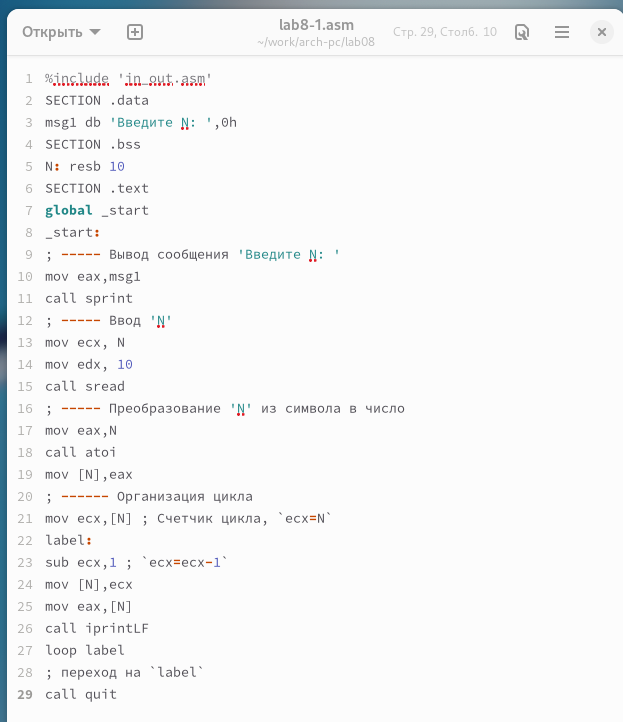


Рис. 3: Программа в файле lab8-1.asm



Рис. 4: Запуск программы lab8-1.asm

Для использования регистра ecx в цикле и сохранения корректности работы программы можно использовать стек. Внес изменения в текст программы добавив команды push и pop (добавления в стек и извлечения из стека) для сохранения значения счетчика цикла loop. Создал исполняемый файл и проверьте его работу.

Программа выводит числа от N-1 до 0, число проходов цикла соответсвует N.

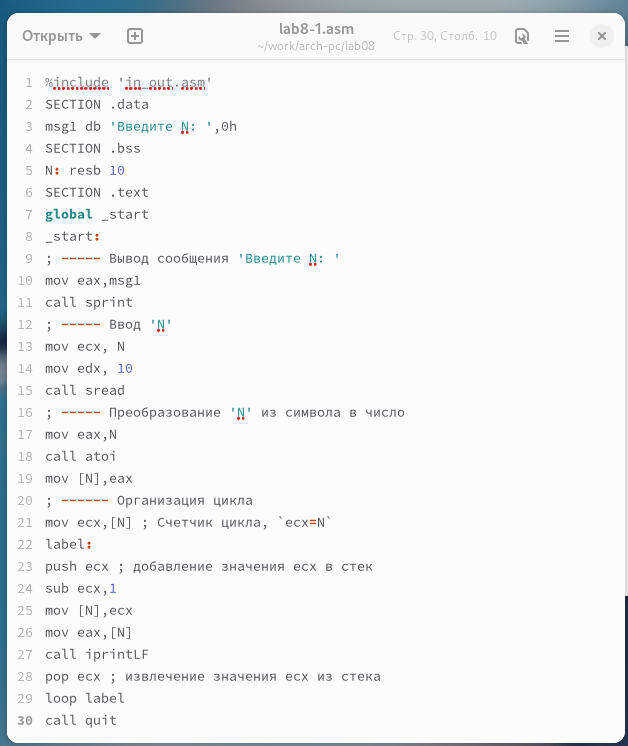


Рис. 5: Программа в файле lab8-1.asm

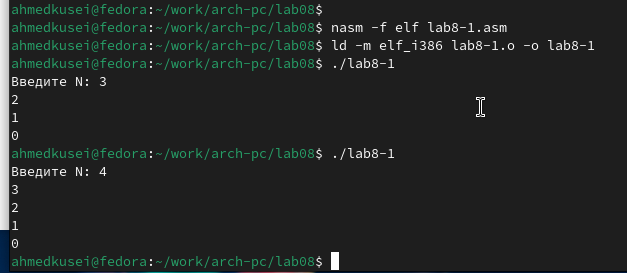


Рис. 6: Запуск программы lab8-1.asm

## 3.2 Обработка аргументов командной строки

Создал файл lab8-2.asm в каталоге ~/work/arch-pc/lab08 и ввел в него текст программы из листинга 8.2.

Создал исполняемый файл и запустил его, указав аргументы. Программа обработала 5 аргументов. Аргументами считаются слова/числа, разделенные пробелом.

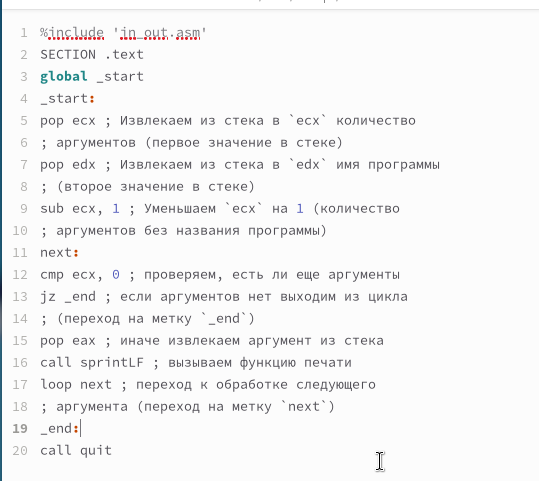


Рис. 7: Программа в файле lab8-2.asm

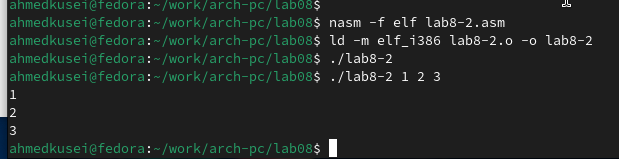


Рис. 8: Запуск программы lab8-2.asm

Рассмотрим еще один пример программы которая выводит сумму чисел, которые передаются в программу как аргументы.

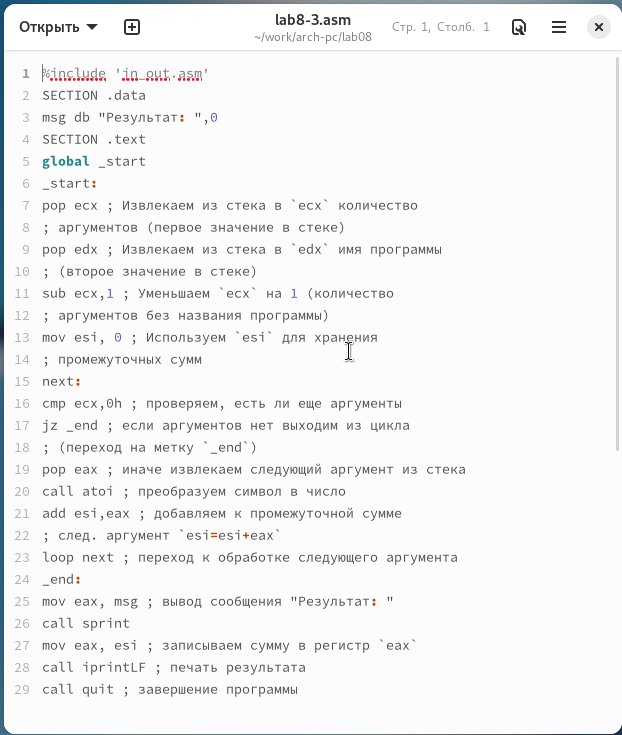


Рис. 9: Программа в файле lab8-3.asm

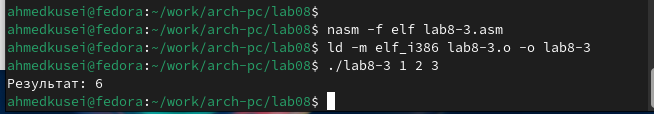


Рис. 10: Запуск программы lab8-3.asm

Изменл текст программы из листинга 8.3 для вычисления произведения аргументов командной строки.

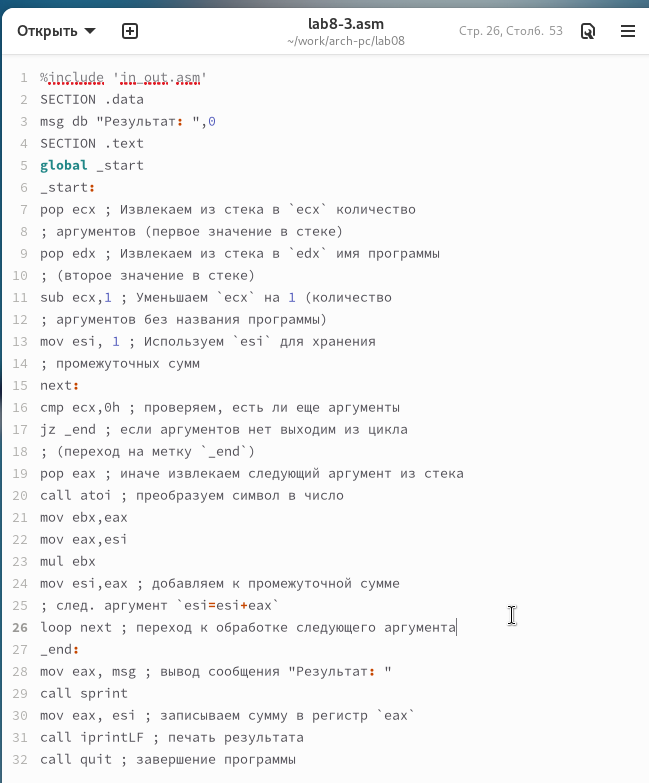


Рис. 11: Программа в файле lab8-3.asm

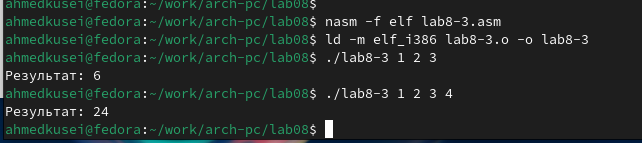


Рис. 12: Запуск программы lab8-3.asm

## 3.3 Задание для самостоятельной работы

Напишите программу, которая находит сумму значений функции для , т.е. программа должна выводить значение . Значения передаются как аргументы. Вид функции выбрать из таблицы 8.1 вариантов заданий в соответствии с вариантом, полученным при выполнении лабораторной работы № 7. Создайте исполняемый файл и проверьте его работу на нескольких наборах .

для варианта 10

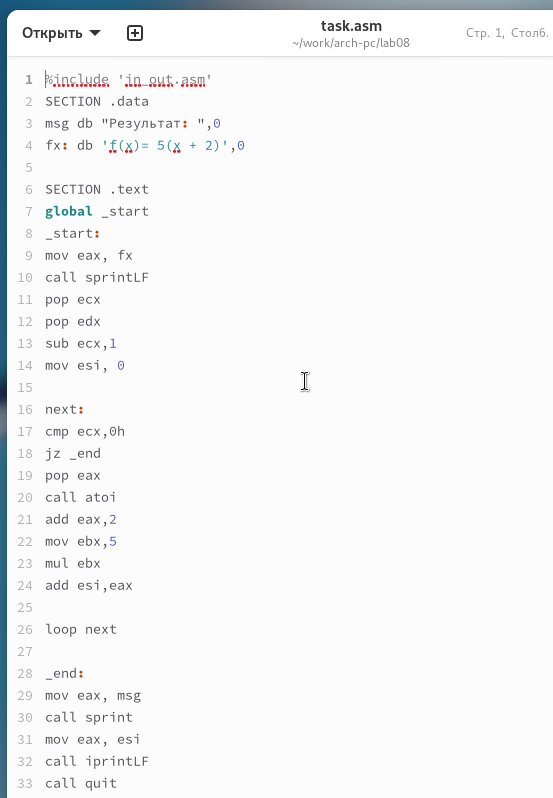


Рис. 13: Программа в файле task.asm

Для проверки я запустил сначала с одним аргументом. Так, при подстановке

Затем подал несколько аргументов и получил сумму значений функции.

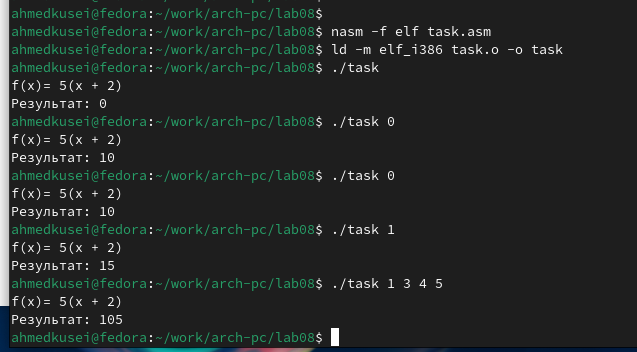


Рис. 14: Запуск программы task.asm

# 4 Выводы

Освоили работы со стеком, циклом и аргументами на ассемблере nasm.