Лабораторная работа №8

Дисциплина: Архитектура компьютера

Серебрякова Дарья Ильинична

Содержание

# 1 Цель работы

Приобретение навыков написания программ с использованием циклов и обработкой аргументов командной строки

# 2 Задание

1. Реализовать циклы в nasm
2. Познакомиться с обработкой аргументов командной строки

# 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. 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).

Команда push размещает значение в стеке, т.е. помещает значение в ячейку памяти, на которую указывает регистр esp, после этого значение регистра esp увеличивается на 4. Данная команда имеет один операнд — значение, которое необходимо поместить в стек.

Команда pop извлекает значение из стека, т.е. извлекает значение из ячейки памяти, на которую указывает регистр esp, после этого уменьшает значение регистра esp на 4. У этой команды также один операнд, который может быть регистром или переменной в памяти. Нужно помнить, что извлечённый из стека элемент не стирается из памяти и остаётся как “мусор”, который будет перезаписан при записи нового значения в стек.

# 4 Выполнение лабораторной работы

## 4.1 Реализация циклов в NASM

Создала каталог для программ лабораторной работы 8, перешла в него и создала файл lab8-1.asm (рис. 1).

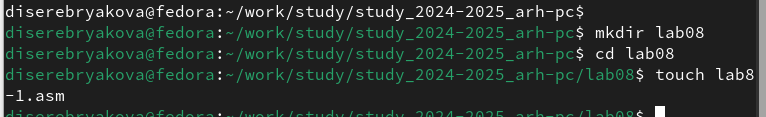


Рис. 1: Создание каталога для работы

В только что созданный файл вписываю программу из предложенного листинга (рис. 2).

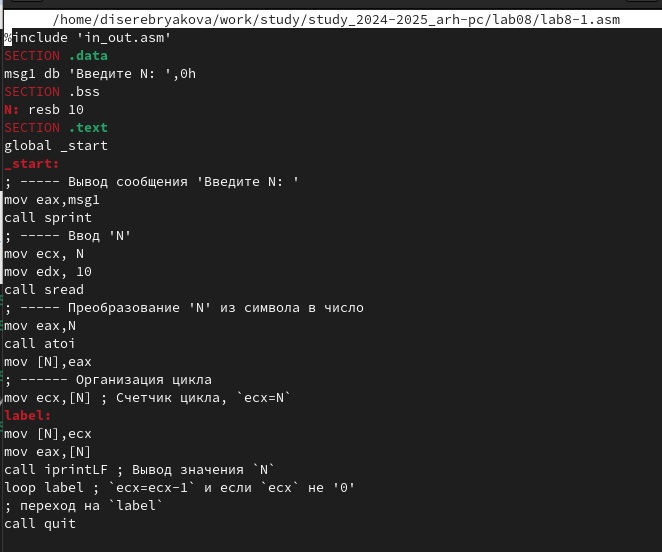


Рис. 2: Программа из листинга

Создаю исполняемый файл и запускаю его (рис. 3).

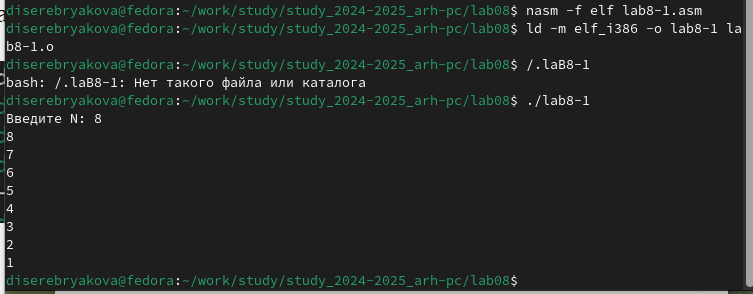


Рис. 3: Запуск программы

Поменяла программу, добавив изменение значение регистра ecx в цикле. Создала исполняемый файл и запустила его (рис. 4).

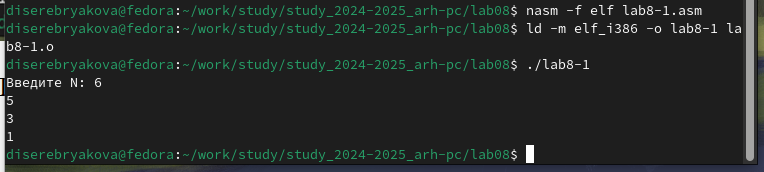


Рис. 4: Запуск измененной программы

Регистр ecx в цикле принимает значения «через одно», то есть каждое следующее значение меньше предыдущего на 2. За счет этого число проходов цикла в два раза меньше введенного с клавиатуры числа N

Для использования регистра ecx в цикле и сохранения корректности работы программы можно использовать стек. Вношу изменения в текст программы добавив команды push и pop (добавления в стек и извлечения из стека) для сохранения значения счетчика цикла loop (рис. 5).

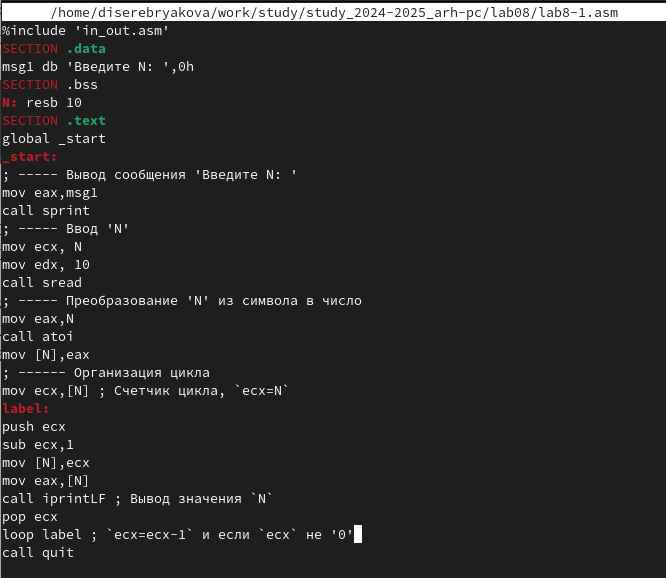


Рис. 5: Измененный текст программы

Создаю исполняемый файл и запускаю его. Теперь число проходов цикла соответствует введенному с клавиатуры значению N, но выводимые числа теперь начинаются от N-1 до 0 (рис. 6).

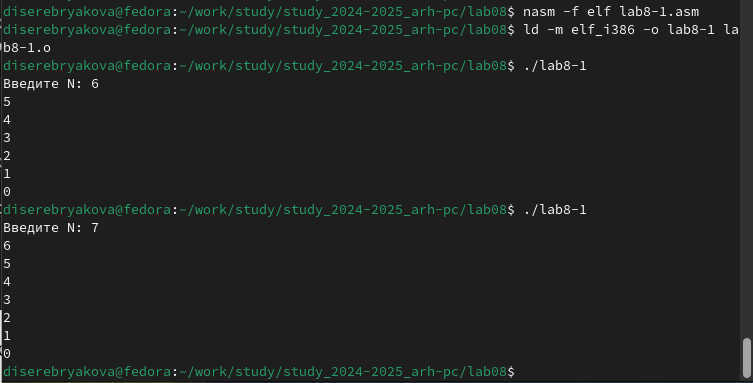


Рис. 6: Проверка работы программы

## 4.2 Обработка аргументов командной строки

Создаю файл lab8-2.asm и ввожу в него текст программы из предложенного листинга (рис. 7).

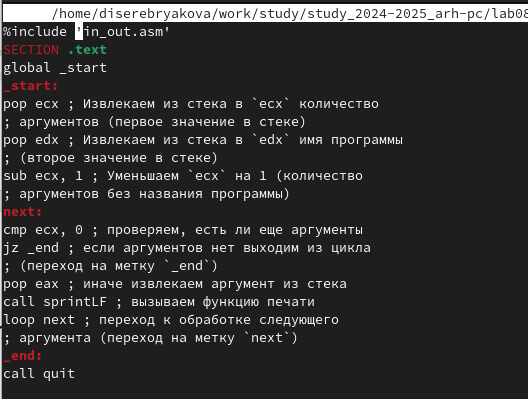


Рис. 7: Написание программы в новом файле

Создаю исполняемый файл и запускаю его, указав предложенные аргументы. Командой было обработано 4 аргумента – столько же, сколько и введено (рис. 8).

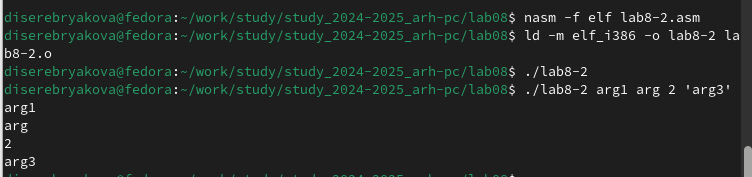


Рис. 8: Обработка заданных аргументов

Командой touch создаю файл lab8-3.asm и ввожу в него текст программы из предложенного листинга (рис. 9).

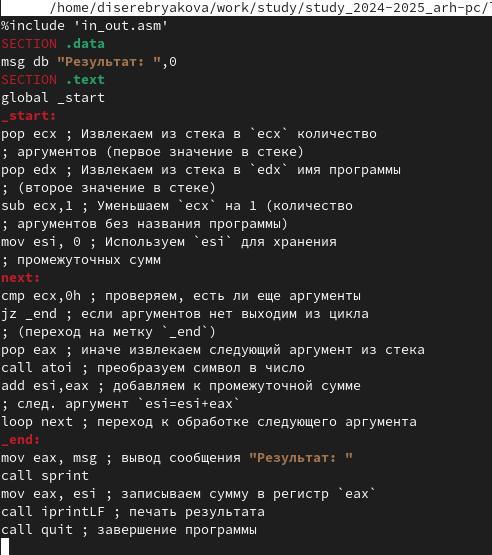


Рис. 9: Программа по предложенному листингу

Создаю исполняемый файл и запускаю его, указав несколько чисел в качестве аргументов. Программа считает их сумму и выводит результат на экран (рис. 10).

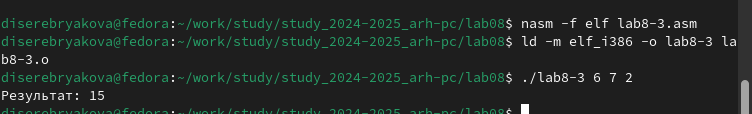


Рис. 10: Подсчет суммы заданных аргументов

Изменяю текст программы так, чтобы она считала произведение введенных аргументов и выводила на экран (рис. 11).

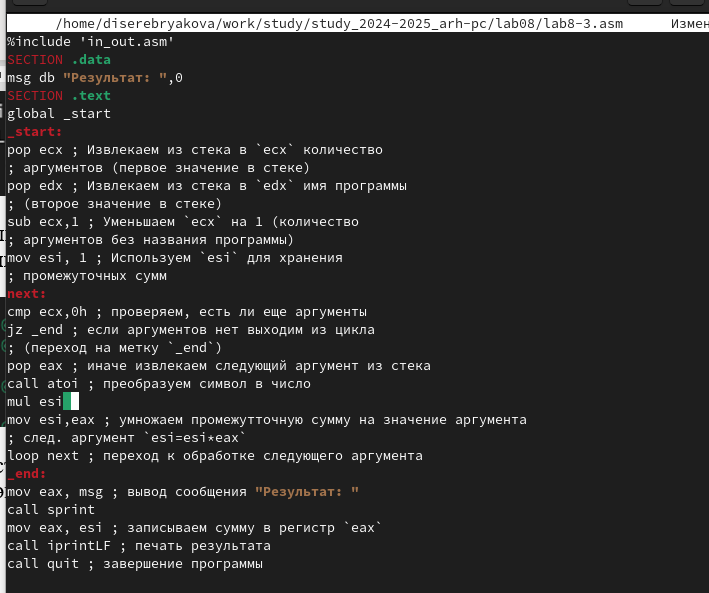


Рис. 11: Измененный текст программы

Создаю исполняемый файл и запускаю его. Теперь программа перемножает заданные аргументы (рис. 12).

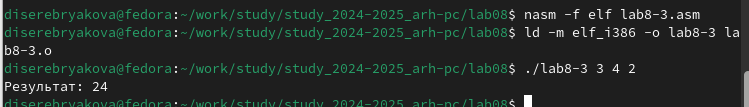


Рис. 12: Подсчет произведения заданных аргументов

# 5 Выполнение заданий для самостоятельной работы

Командой touch создаю файл lab8-4.asm и прописываю в нем текст программы, которая будет подсчитывать сумму значений функции при заданных аргументах (рис. 13).

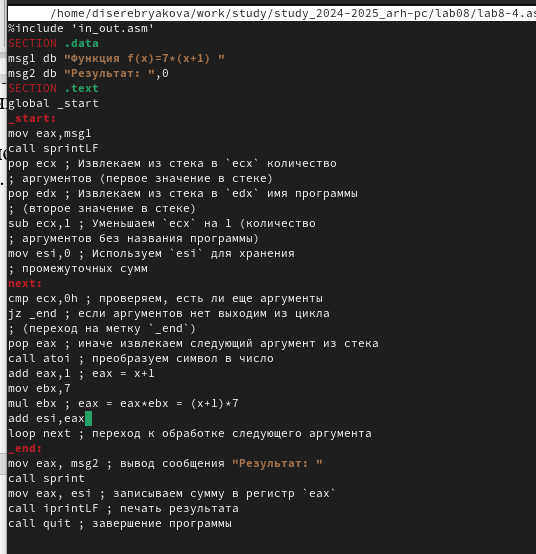


Рис. 13: Текст программы по заданию для самостоятельной работы

Создаю исполняемый файл и запускаю его несколько раз на разных значениях аргументов. Результат получается верный, значит программа написана корректно (рис. 14).

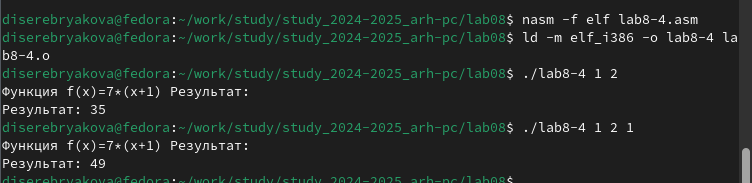


Рис. 14: Проверка работы написанной программы

# 6 Выводы

Приобретены навыки написания программ с использованием циклов и обработкой аргументов командной строки

# Список литературы

1. Лабораторная работа 8