Отчёт по лабораторной работе № 8

Дисциплина: архитектура компьютера

Кайнова Алина Андреевна

Содержание

# 1 Цель работы

Научиться писать программы с использованием циклов и обработкой аргументов командной строки.

# 2 Задание

1. Реализация циклов в NASM
2. Обработка аргументов командной строки
3. Выполнение заданий для самостоятельной 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. 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. 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 реги- стра 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. Для организации циклов существуют специальные инструкции. Для всех ин- 7струкций максимальное количество проходов задаётся в регистре ecx. Наиболее простой является инструкция loop. Она позволяет организовать безусловный цикл.

# 4 Выполнение лабораторной работы

## 4.1 Реализация циклов в NASM

Создаю каталог для данной лабораторной работы и файл lab8-1.asm

Figure 1: Создание каталога и файла

Figure 1: Создание каталога и файла

Ввожу в созданный файл текст программы из листинга 8.1

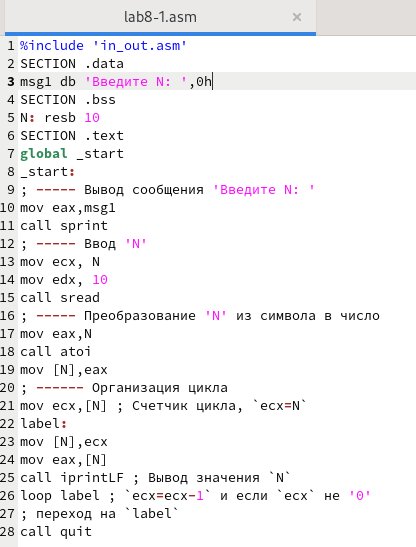


Figure 2: Ввод текста программы

Создаю исполняемый файл и проверяю его работу

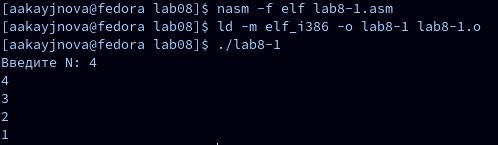


Figure 3: Создание и запуск исполняемого файла

Программа выводит числа от N до 1 включительно.

Изменяю текст программы, изменив значение регистра ecx в цикле

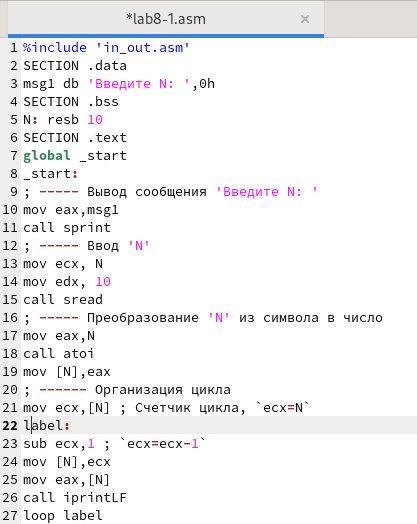


Figure 4: Изменение текста программы

Создаю исполняемый файл и проверяю его работу

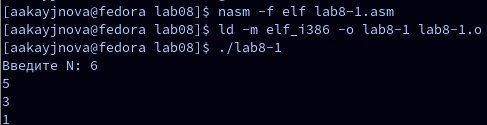


Figure 5: Создание и запуск исполняемого файла

Регистр ecx принимает нечётные значения в цикле. Число проходов не соответствует введённому значению N.

Вношу изменения в текст программы, добавив команды push и pop для сохранения значения счётчика цикла loop

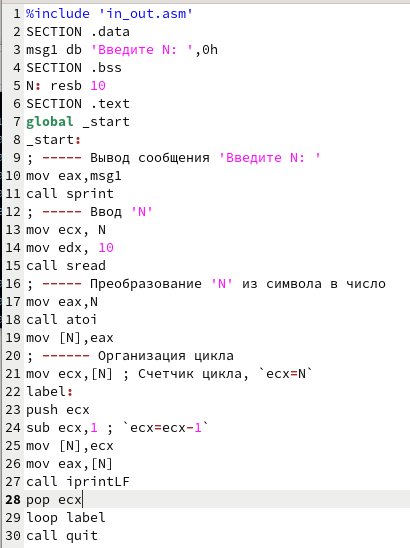


Figure 6: Изменение текста программы

Создаю исполняемый файл и проверяю его работу

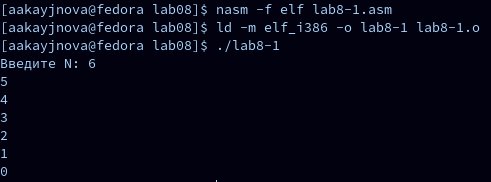


Figure 7: Создание и запуск исполняемого файла

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

## 4.2 Обработка агрмуентов командной строки

Создаю файл lab8-2.asm в данном каталоге и ввожу в него текст из программы листинга 8.2

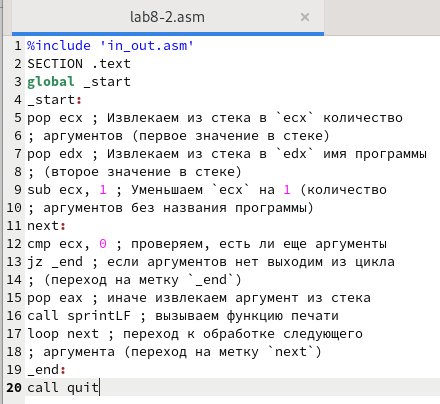


Figure 8: Ввод текста программы

Создаю исполняемый файл и проверяю его работу

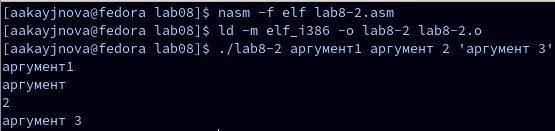


Figure 9: Создание и запуск исполняемого файла

Программа выводит 4 аргумента, так как аргумент 2 не взят в кавычки, поэтому из-за пробела программа считывает 2 как отдельный аргумент.

Создаю файл lab8-3.asm в данном каталоге и ввожу в него текст программы из листинга 8.3

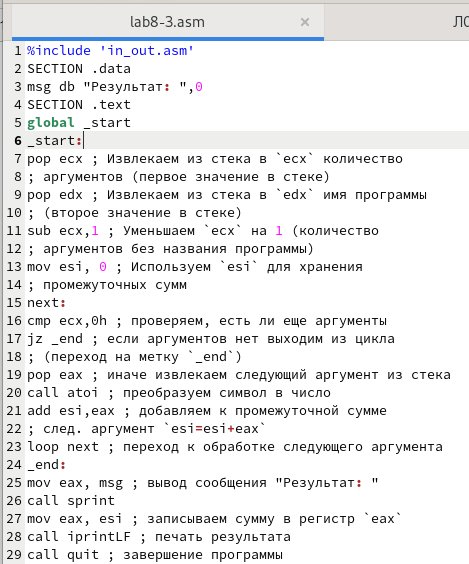


Figure 10: Ввод текста программы

Создаю исполняемый файл и запускаю его, указав аргументы

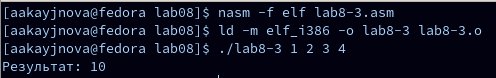


Figure 11: Создание и запуск исполняемого файла

Изменяю текст программы для вычисления произведения аргументов командной строки

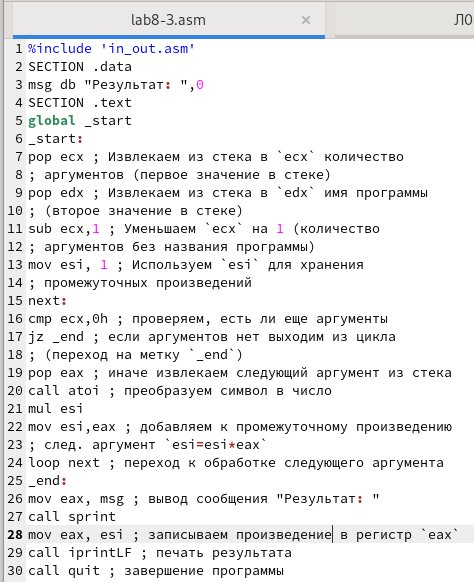


Figure 12: Изменение текста программы

Создаю исполняемый файл и запускаю его, указав аргументы

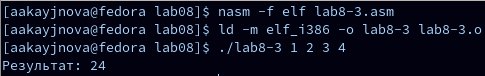


Figure 13: Создание и запуск исполняемого файла

## 4.3 Выполнение заданий для самостоятельной работы

Пишу текст программы, которая находит сумму значений функции f(x)=4\*x+3 в соответствии с моим вариантом № 5 для значений x=x1,x2,…,xn. Значения xi передаются как аргументы.

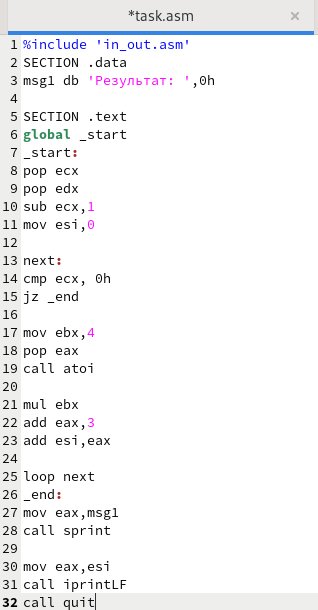


Figure 14: Текст программы

Создаю исполняемый файл и проверяю его работу на нескольких наборах x=x1,x2,…,xn

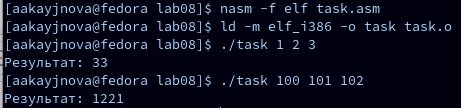


Figure 15: Запуск исполняемого файла и проверка его работы

Программа работает корректно.

# 5 Выводы

В ходе данной лабораторной работы мы приобрели навыки написания программ с использованием циклов и обработкой аргументов командной строки.

# Список литературы

1. 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 :::