Шаблон отчёта по лабораторной работе

8

Бембо Жозе Лумингу

Содержание

[1 Цель работы 1](#__RefHeading___1)

[2 Задание 1](#__RefHeading___2)

[3 Теоретическое введение 1](#__RefHeading___3)

[4 Выполнение лабораторной работы 2](#__RefHeading___4)

[4.1 Реализация циклов в NASM. 2](#__RefHeading___5)

[4.2 Обработка аргументов командной строки. 5](#__RefHeading___6)

[4.3 Задание для самостоятельной работы. 8](#__RefHeading___7)

[5 Выводы 9](#__RefHeading___8)

[Список литературы 9](#__RefHeading___9)

# 1 Цель работы

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

# 2 Задание

Реализация циклов в NASM.

Обработка аргументов командной строки.

Задание 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.

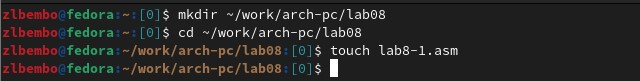
Команда pop извлекает значение из стека, т.е. извлекает значение из ячейки памяти, на которую указывает регистр esp, после этого уменьшает значение регистра esp на 4. У этой команды также один операнд, который может быть регистром или переменной в памяти. Нужно помнить, что извлечённый из стека элемент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

Для организации циклов существуют специальные инструкции. Для всех инструкций максимальное количество проходов задаётся в регистре ecx. Наиболее простой является инструкция loop. Она позволяет организовать безусловный цикл.

# 4 Выполнение лабораторной работы

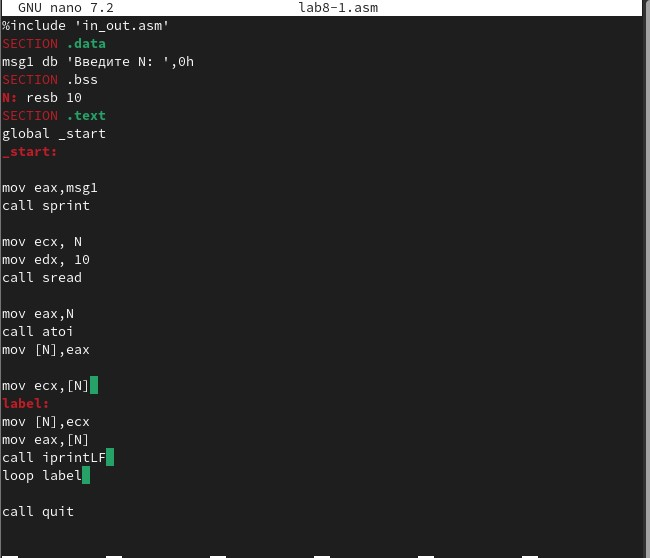
## 4.1 Реализация циклов в NASM.

Создаю каталог для программ лабораторной работы № 8, перехожу в него и создаю файл lab8-1.asm.(рис.[??]).



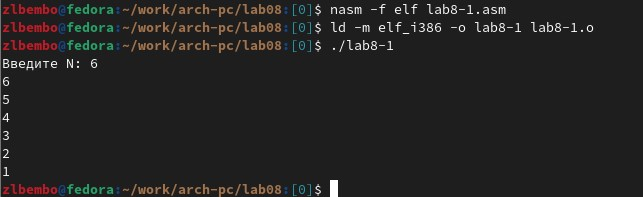
создание файлов

Ввожу в файл lab8-1.asm текст программы из листинга 8.1. (рис.[??]).



ввод текста

Создаю исполняемый файл и проверяю его работу. (рис.[??]).



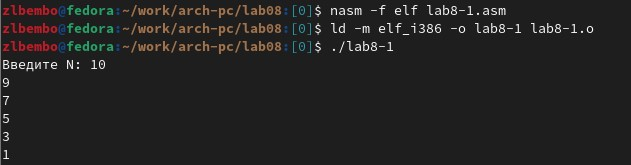
запуск исполняемого файла

Изменяю текст программы, добавив изменение значения регистра ecx в цикле. (рис.[??]).



изменение текста программы

Создаю исполняемый файл и проверяю его работу. (рис.[??]).



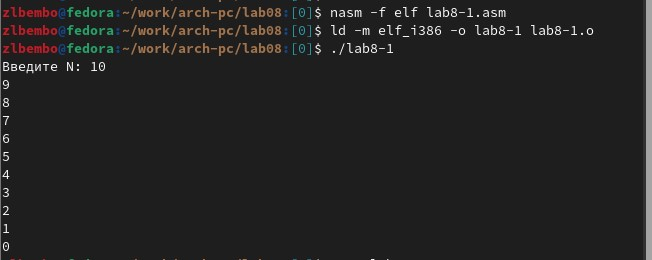
запуск обновленной файла

Вношу изменения в текст программы, добавив команды push и pop для сохранения значения счетчика цикла loop. (рис. [??]).



изменение текста программы

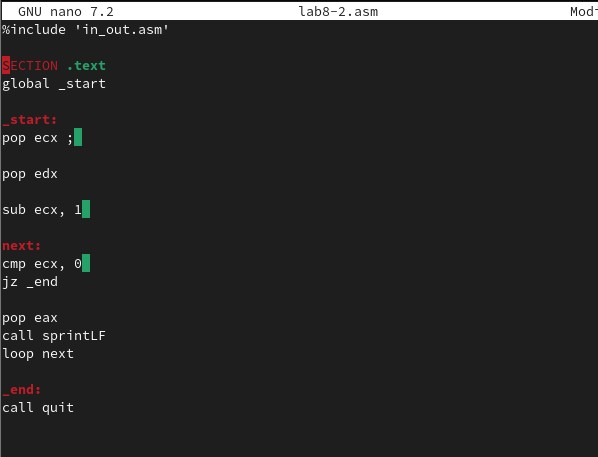
Создаю исполняемый файл и проверяю его работу. (рис.[??]).



запуск исполняемого файла

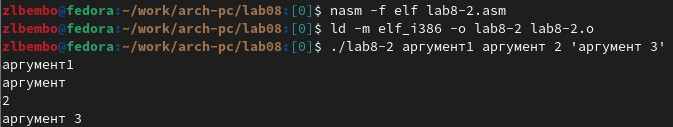
## 4.2 Обработка аргументов командной строки.

На этом шаге мы создали файл lab8-2.asm, затем заполнили в нем наш код. (рис.[??]).



ввод текста

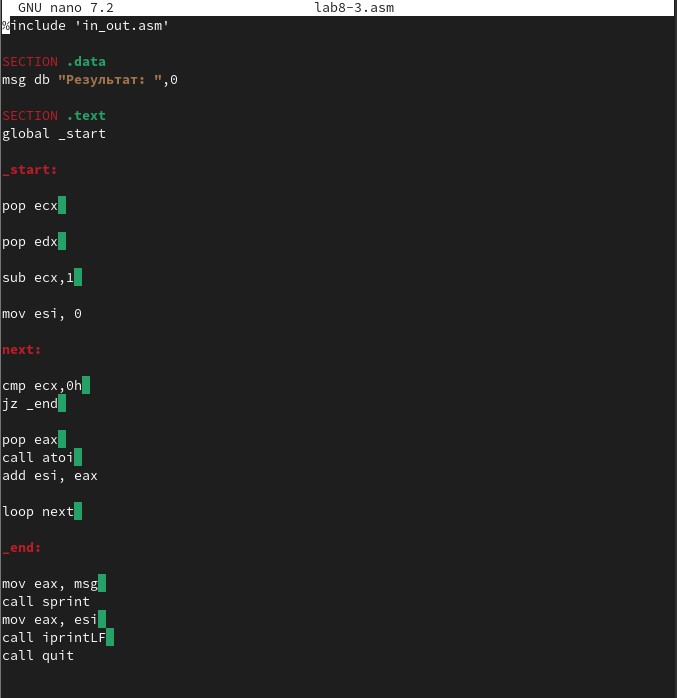
Создаю исполняемый файл и запускаю его, указав нужные аргументы. (рис.[??]).



запуск исполняемого файла

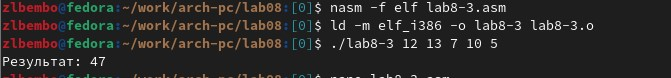
И, как вы можете видеть, на этот раз при запуске программы мы добавили в команду три аргумента, и в этом случае были обработаны три аргумента

Первым делом мы создали файл lab8-3.asm, затем заполнили кодом программы. (рис.[??]).



ввод текста

Создаю исполняемый файл и запускаю его, указав аргументы.(рис.[??]).



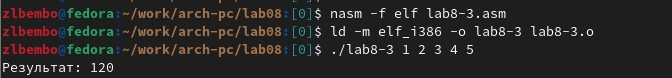
запуск исполняемого файла

Изменяю текст программы из листинга 8.3 для вычисления произведения аргументов командной строки. (рис.[??]).



изменение текста программы

Создаю исполняемый файл и запускаю его, указав аргументы. (рис.[??]).

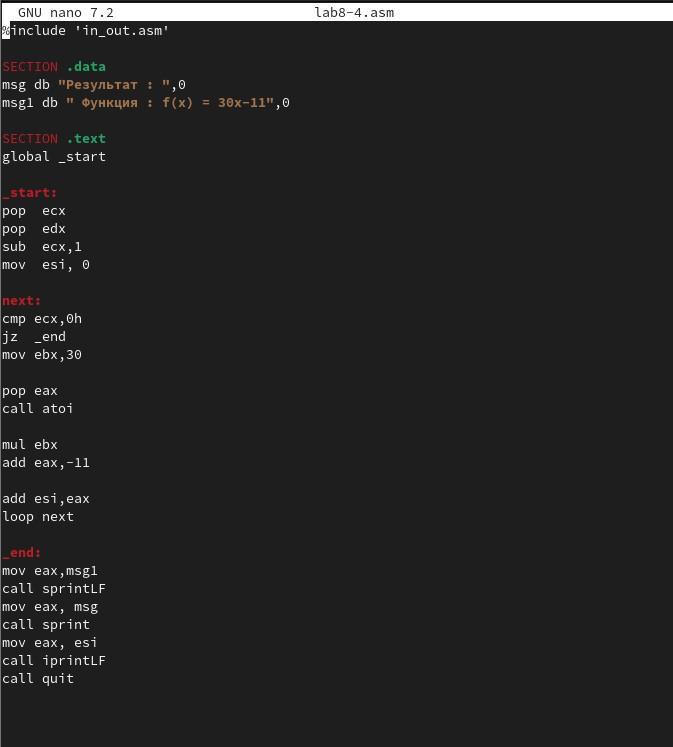


запуск исполняемого файла

## 4.3 Задание для самостоятельной работы.

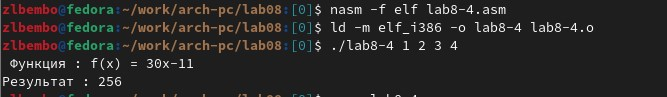
В этой части мы должны были написать программу,которая находит сумму значений функции f(x) для x = x1, x2, …, xn

сначала мы создали наш файл lab8-4.asm, где будет находиться наш код, затем мы написали программу. (рис.[??]).



текст программы

Создаю исполняемый файл и проверьте его работу на нескольких наборах x = x1, x2, …, xn. (рис.[??]).



запуск исполняемого файла

# 5 Выводы

Благодаря этой лабораторной работе мы научились писать программы с использованием циклов и обработки аргументов командной строки, что поможет нам в дальнейшей лабораторной работе.

# Список литературы

::: {#refs} :::