Отчет по лабораторной работе №8

Программирование цикла. Обработка аргументов командной строки.

Ашуров Захид Фамил оглы

Содержание

# 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

# 3 Теоретическое введение

* Организация стека 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 ре- гистре esp (указатель стека). Противоположный конец стека называется дном. Значение, помещённое в стек последним, извлекается первым. При помещении значения в стек указа- тель стека уменьшается, а при извлечении — увеличивается. Для стека существует две основные операции:
* • добавление элемента в вершину стека (push); • извлечение элемента из вершины стека (pop). 8.2.1.1. Добавление элемента в стек.
* 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.

push -10 ; Поместить -10 в стек push ebx ; Поместить значение регистра ebx в стек push [buf] ; Поместить значение переменной buf в стек push word [ax] ; Поместить в стек слово по адресу в ax

Существует ещё две команды для добавления значений в стек. Это команда pusha, которая

помещает в стек содержимое всех регистров общего назначения в следующем порядке: ах, сх, dx, bх, sp, bp, si, di. А также команда pushf, которая служит для перемещения в стек содержимого регистра флагов. Обе эти команды не имеют операндов.

8.2.1.2. Извлечение элемента из стека.

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.

Примеры:

pop eax ; Поместить значение из стека в регистр eax pop [buf] ; Поместить значение из стека в buf pop word[si] ; Поместить значение из стека в слово по адресу в si

Аналогично команде записи в стек существует команда popa, которая восстанавливает из стека все регистры общего назначения, и команда popf для перемещения значений из вершины стека в регистр флагов.

Для организации циклов существуют специальные инструкции. Для всех инструкций

максимальное количество проходов задаётся в регистре ecx. Наиболее простой является ин- струкция loop. Она позволяет организовать безусловный цикл, типичная структура которого имеет следующий вид:

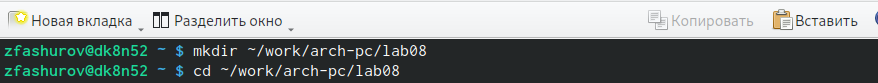
mov ecx, 100 ; Количество проходов NextStep: … … ; тело цикла … loop NextStep ; Повторить ecx раз от метки NextStep

Иструкция loop выполняется в два этапа. Сначала из регистра ecx вычитается единица и

его значение сравнивается с нулём. Если регистр не равен нулю, то выполняется переход к указанной метке. Иначе переход не выполняется и управление передаётся команде, которая следует сразу после команды loop.

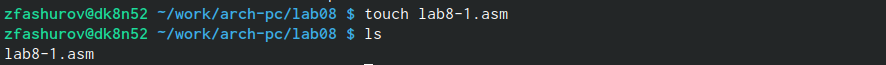
# 4 Выполнение лабораторной работы

Создаем папку и переходим в него (Рис. ??)



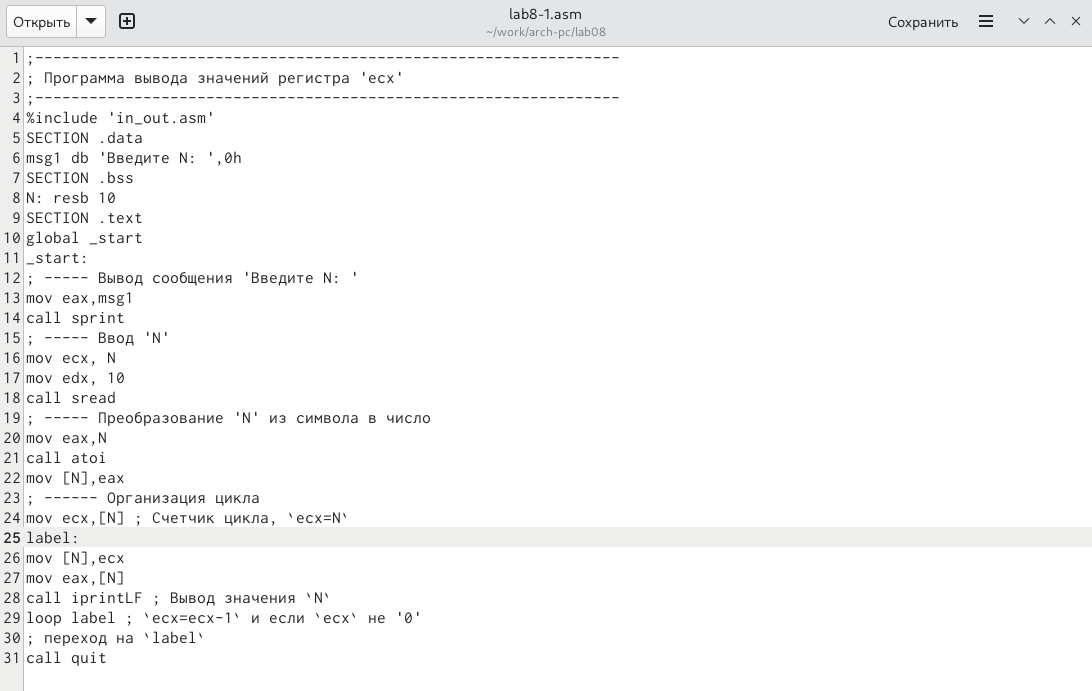
Создание папки и переход в него

Создаем файл lab8-1.asm (Рис. ??).



Создание файла lab8-1.asm

Вписываем текст из листинга в lab8-1.asm (Рис. ??).



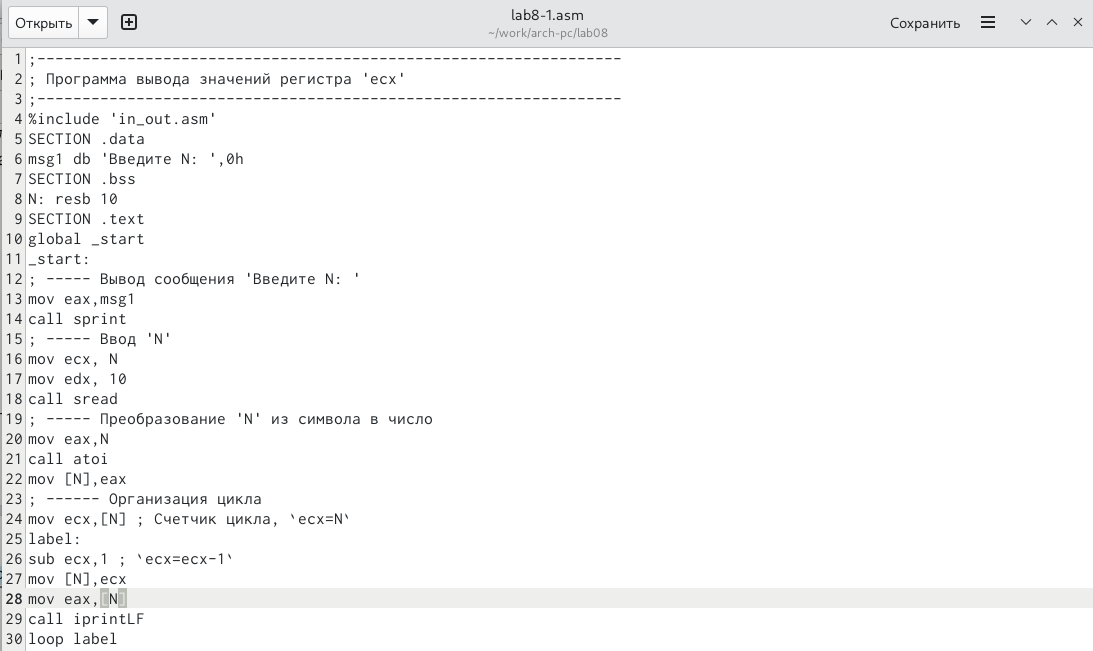
Вписывание текста из листинга в lab8-1.asm

Некоректный запуск (Рис. ??).

Некоректный запуск

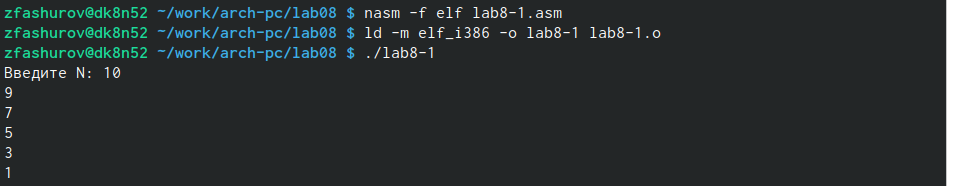
Некоректный запуск

Редактируем файл lab8-1.asm (Рис. ??).



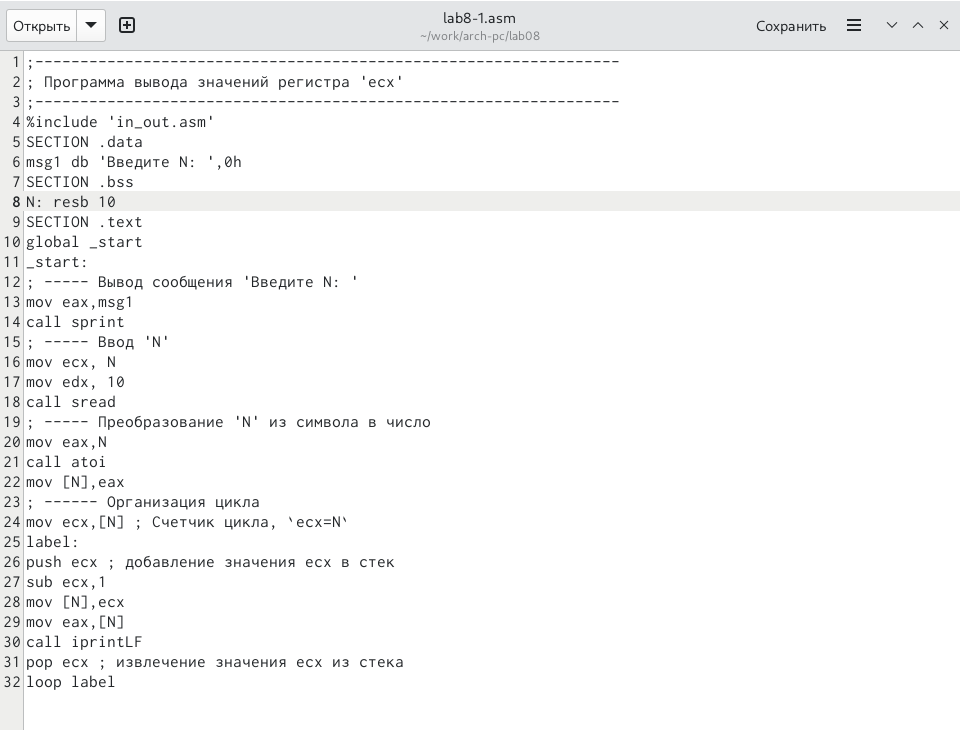
Редактирование файла lab8-1.asm

Создаем и запускаем исполняемый файл (Рис. ??).



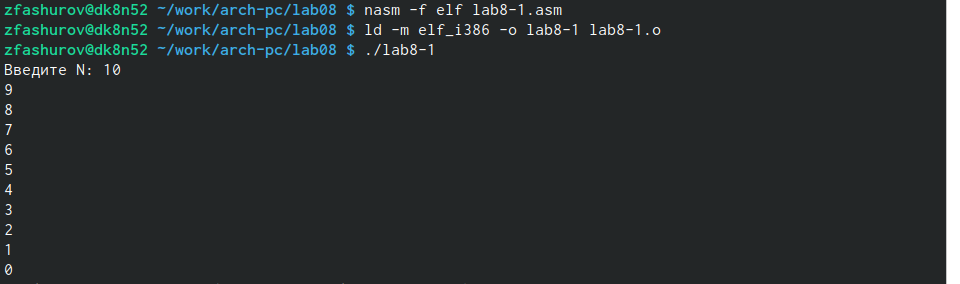
Создание и запускание исполняемого файла

Вносим изменения в файл lab8-1.asm(Рис. ??).



Внесения изменения в lab8-1.asm

Создаем и запускаем исполняемый файл (Рис. ??).



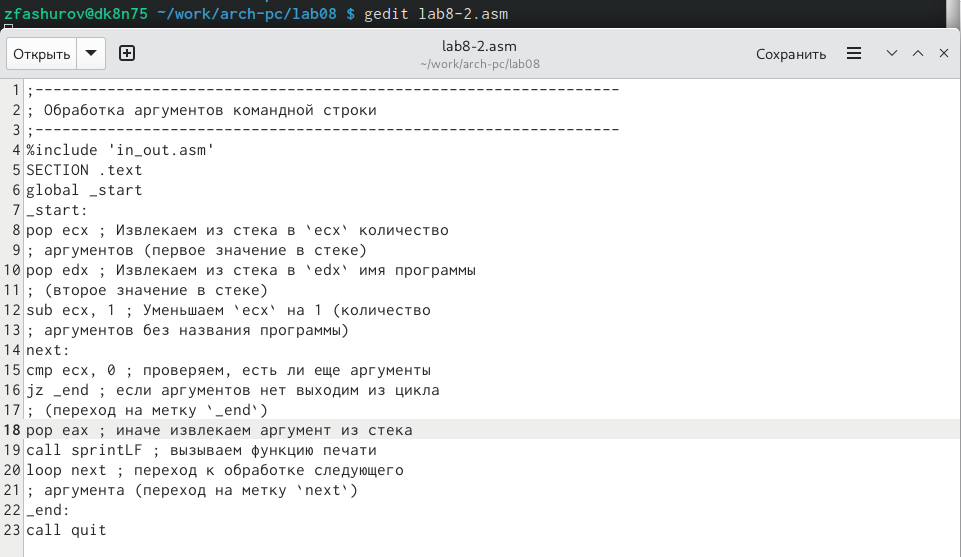
Создание и запускание исполняемого файл

Создаем файл lab8-2.asm (Рис. ??).

Создание файла lab8-2.asm

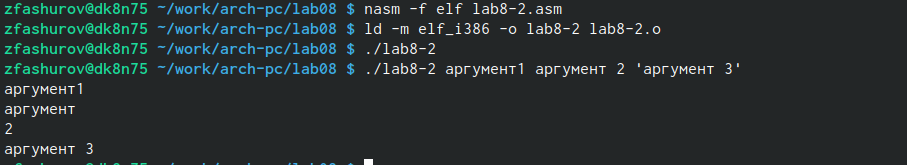
Создание файла lab8-2.asm

Вписываем текст из листинга в файл lab8-2.asm(Рис. ??).



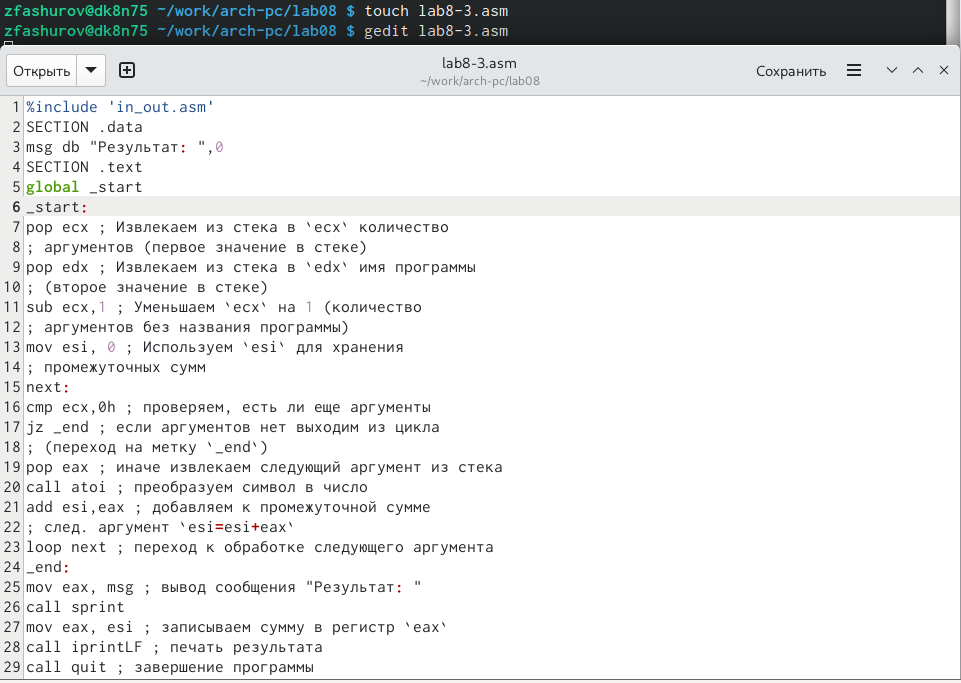
Вписывание текста из листинга в файл lab8-2.asm

Создаем и запускаем исполняемый файл (Рис. ??).



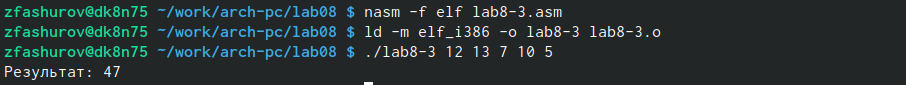
Создание и запускание исполняемого файла

Создаем и редактируем файл lab8-3.asm (Рис. ??).



Создание и редактирование файла lab8-3.asm

Создаем и запускаем исполняемый файл (Рис. ??).



Создание и запускание исполняемого файла

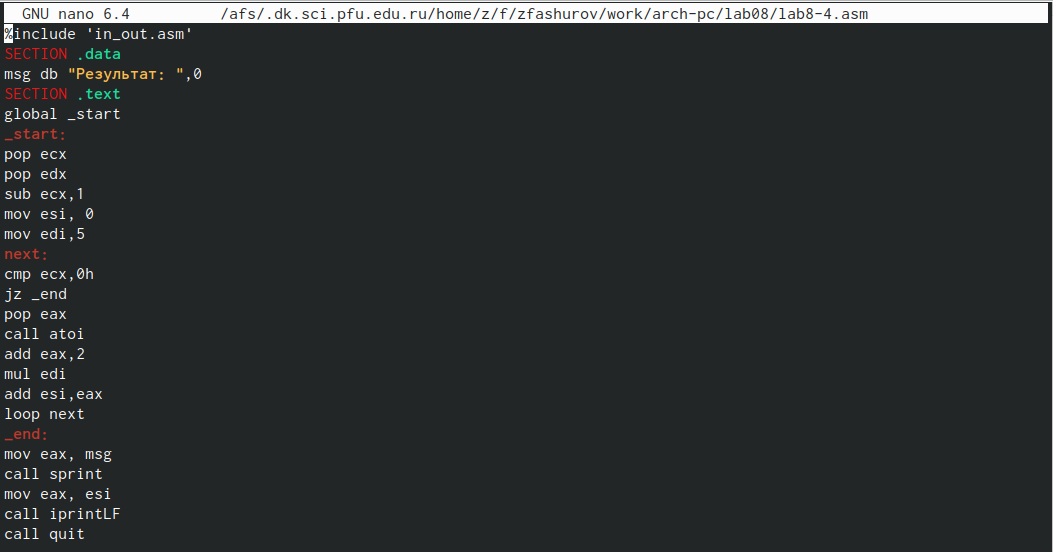
# 5 Самостаятельная работы

Создаем файл lab8-4.asm (Рис. ??).

Создание файла lab8-4.asm

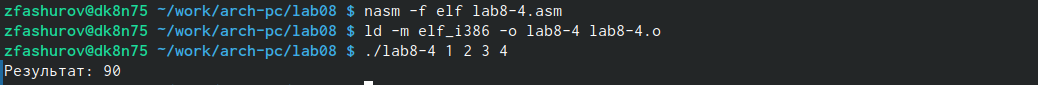
Создание файла lab8-4.asm

Записываем код (Рис. ??).



Записывание кода

Создаем и запускаем исполняемый файл (Рис. ??).



Создание и запускание исполняемого файла

# 6 Выводы

Я приобрел навыки написания программ с ипользованием циклов и обработкой аргументов командной строки.

# Список литературы