Отчёт по лабораторной работе №8

Дисциплина: Архитектура компьютера

Абрамова Ульяна Михайловна

Содержание

# 1 Цель работы

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

# 2 Задание

1. Реализация циклов в NASM
2. Обработка аргументов командной строки
3. Выполнение задания для самостоятельной работы

# 3 Теоретическое введение

## 3.1 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).

### 3.1.1 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.

### 3.1.2 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. 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.

## 3.2 Инструкции организации циклов

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

## 4.1 Реализация циклов в NASM

Создаю каталог для программ лабораторной работы №8, перехожу в него и создаю файл lab8-1.asm, в который записываю программу вывода значений регистра ecx (рис. 1).

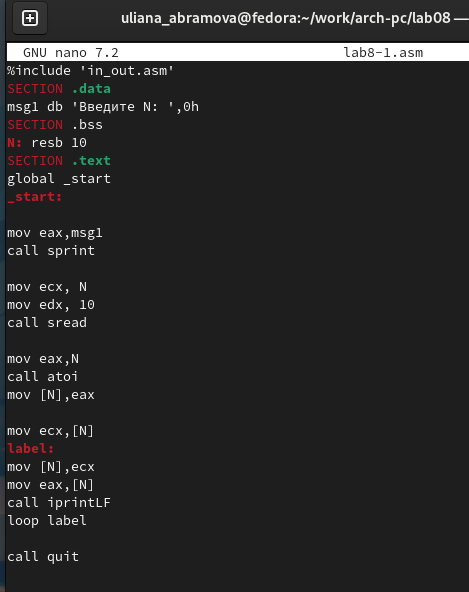


Рис. 1: Написание программы

Создаю исполняемый файл и проверяю его работу (рис. 2).

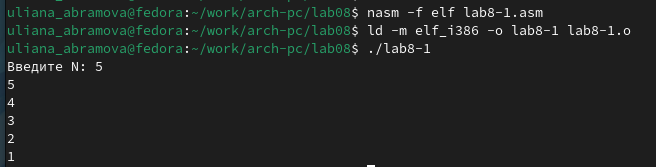


Рис. 2: Запуск программы

Далее изменяю текст программы, добавив изменение значения регистра ecx в цикле (рис. 3).

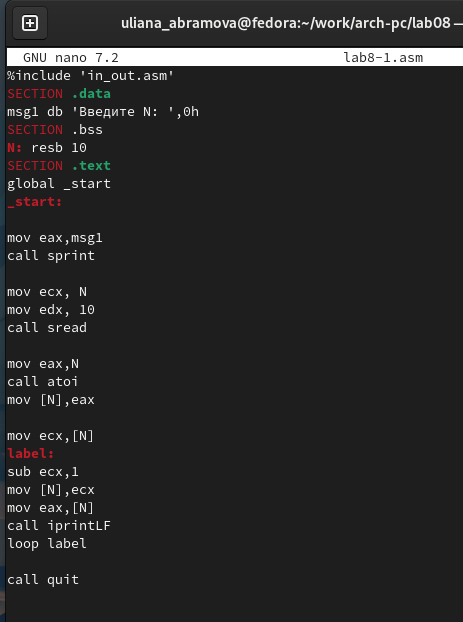


Рис. 3: Изменение программы

Cоздаю исполняемый файл и проверяю его работу. Из-за изменения цикл закольцевался и стал бесконечным (рис. 4).

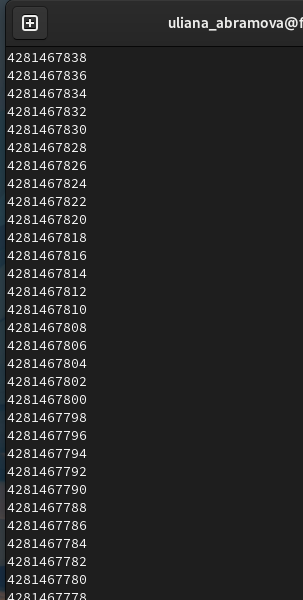


Рис. 4: Запуск программы

Снова ввожу изменения в текст программы для корректкной работы цикла и счетчика (рис. 5).

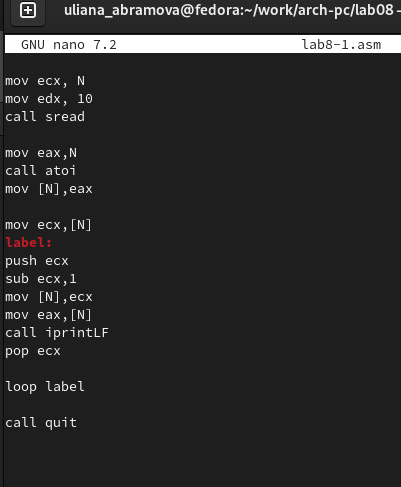


Рис. 5: Изменение программы

Создаю исполняемый файл и проверяю его работу. По итогу изменения программы, число проходки циклов стало соответсвовать числу введенному с клавиатуры (рис. 6).

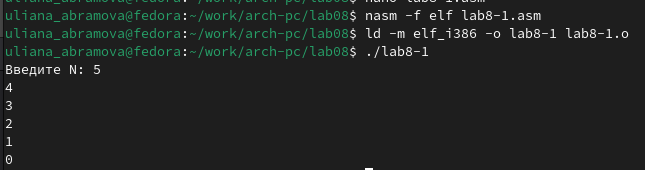


Рис. 6: Запуск программы

## 4.2 Обработка агрументов командной строки

Создаю в том же каталоге файл lab8-2.asm и пишу в нем программу, выводящую на экран аргументы командной строки (рис. 7).

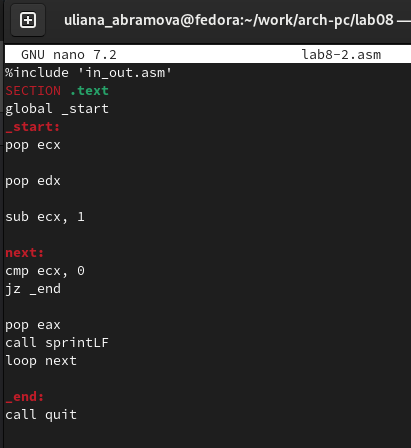


Рис. 7: Написание программы

Создаю исполняемый файл и прверяю его работу. Данная программа выводит все 3 введенных аргумента, но в разной вариации. (рис. 8)

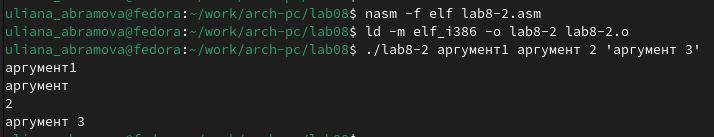


Рис. 8: Запуск программы

СОздаю новый файл lab8-3.asm в прежнем каталоге и пишу в него программу вычисления суммы аргументов командной строки (рис. 9).

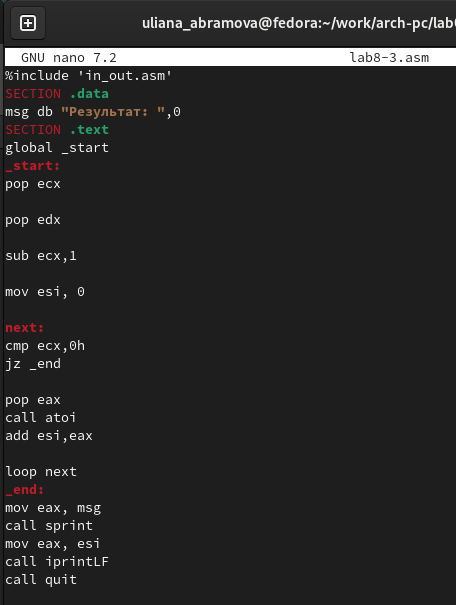


Рис. 9: Написание программы

Создаю исполняемый файл и проверяю его работу с указанными аргументами (рис. 10).

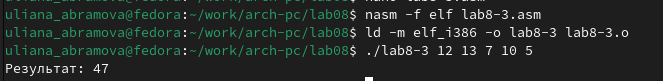


Рис. 10: Запуск программы

Изменяю текст программы для вычисления произведения аргументов командной строки (рис. 11).

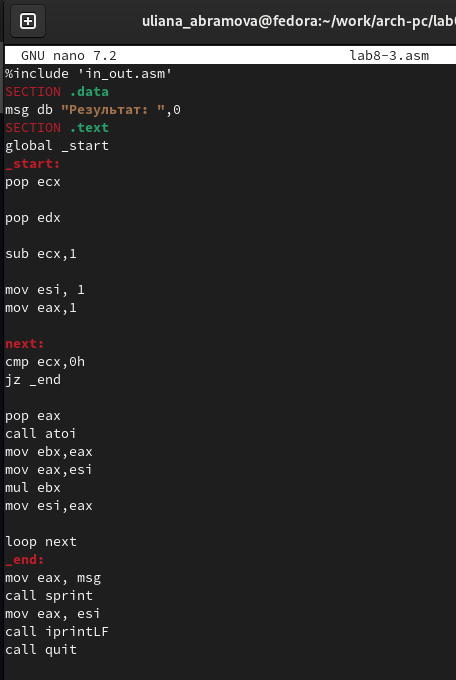


Рис. 11: Изменение программы

Создаю исполняемый файл и проверяю его работу. Программа отработана верно (рис. 12).

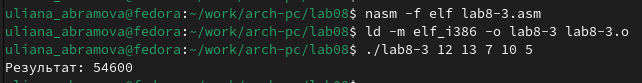


Рис. 12: Запуск программы

## 4.3 Выполнение задания для самостоятельной работы

Нужно написать программу, которая находит сумму значений функции f(x) для x=x1, x2,…, xn,т.е. программа должна выводить значение f(x1)+f(x2)+…+f(xn). Вид функции моего, 3 варианта: f(x)=10x-5, соответственно программа написана для этого вида (рис. 13).

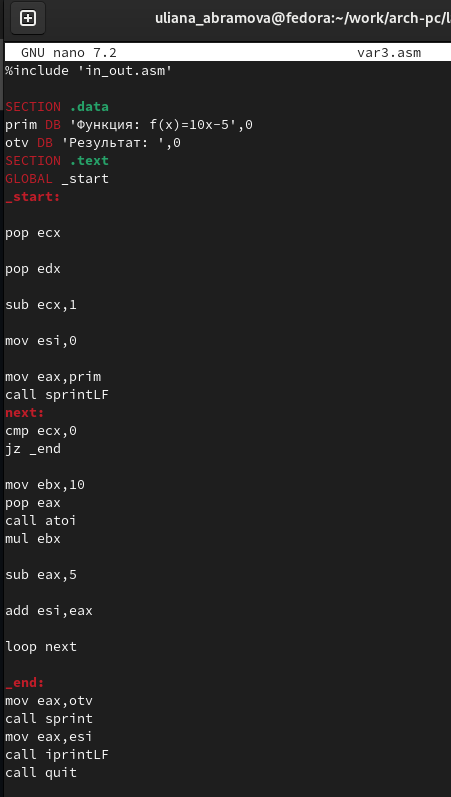


Рис. 13: Написание программы

Создаю исполняемый файл и проверяю его работу при разных значениях аргументов. Программа отработана верно (рис. 14).

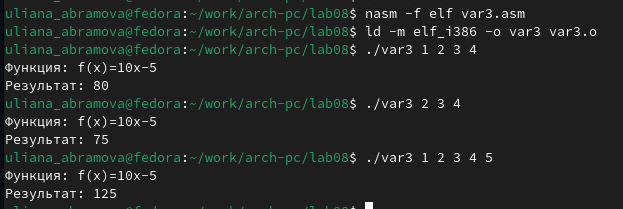


Рис. 14: Запукс программы

# 5 Выводы

Я приобрела навыки написания программы с использованием цикла.

# 6 Список литературы

1. [Архитектура компьютера](https://esystem.rudn.ru/pluginfile.php/2089548/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)