**РОССИЙСКИЙ УНИВЕРСИТЕТ ДРУЖБЫ НАРОДОВ**

**Факультет физико-математических и естественных наук**

**Кафедра прикладной информатики и теории вероятностей**

**ОТЧЕТ**

**по лабораторной работе № 9**

*дисциплина: Архитектура компьютера*

Студент: Волгин А.А.

Группа: НПИбд-01-22

**МОСКВА**

2022 г.

**Цель работы:**

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

**Порядок выполнения лабораторной работы:**

**Реализация циклов в NASM.**

Создадим каталог для программ лабораторной работы №9, перейдем в него и создадим нужный файл (рис. 1).

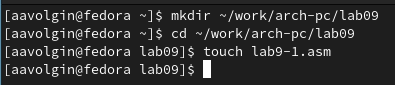


рис. 1. Создание файла lab9-1.asm

При реализации циклов в NASM с использованием инструкции loop необходимо помнить о том, что эта инструкция использует регистр ecx в качестве счетчика и на каждом шаге уменьшает его значение на единицу. В качестве примера рассмотрим программу, которая выводит значение регистра ecx (рис. 2).

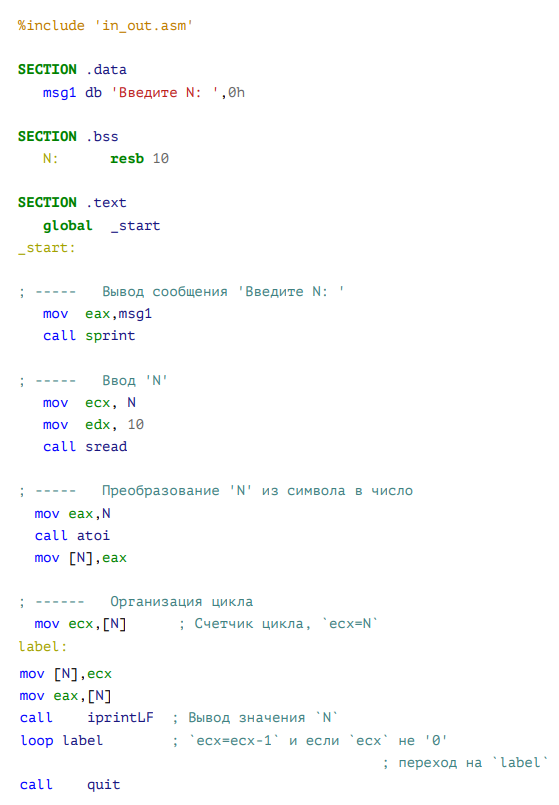


рис. 2. Текст программы lab9-1

Создадим исполняемый файл и проверим его работу (рис. 3).

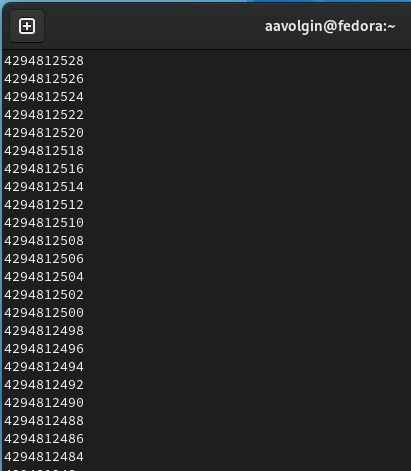


рис. 3. Результат работы программы lab9-1

Данный пример показывает, что использование регистра ecx в теле цилка loop может привести к некорректной работе программы. Изменим текст программы добавив изменение значение регистра ecx в цикле по следующему примеру (рис. 4).

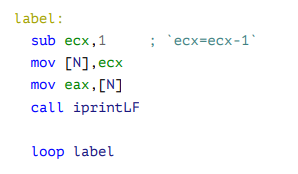


рис. 4. Пример изменения части программы lab9-1

Создадим исполняемый файл и проверим его работу (рис. 5).

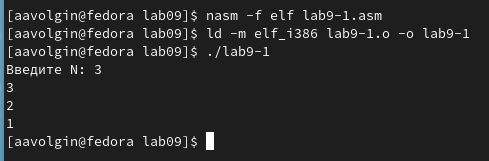


рис. 5. Результат работы измененной программы lab9-1

Как видим, все работает. Регистр ecx принимает все значения от N до 1 включительно, что соответствует числу проходов цикла, введенному с клавиатуры.

Для использования регистра ecx в цикле и сохранения корректности работы программы можно использовать стек. Внесем изменения в текст программы по примеру, добавив команды push и pop (добавления в стек и извлечения из стека) для сохранения значения счетчика цикла loop (рис. 6).

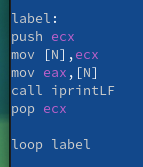


рис. 6. Внесение команд push и pop в текст программы lab9-1

Создадим исполняемый файл и проверим его работу (рис. 7).

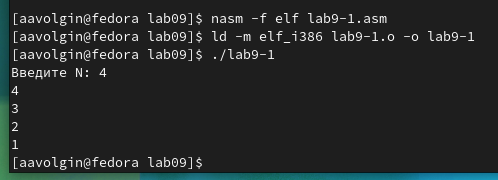


рис. 7. Результат работы измененной программы lab9-1

Как видим, программа работает корректно, число проходов цикла соответствует значению N, введенному с клавиатуры.

**Обработка аргументов командной строки.**

При разработке программ иногда встает необходимость указывать аргументы, которые будут использоваться в программе, непосредственно из командной строки при запуске программы. При запуске программы в NASM аргументы командной строки загружаются в стек в обратном порядке, кроме того в стек записывается имя программы и общее количество аргументов. Последние два элемента стека для программы, скомпилированной NASM, – это всегда имя программы и количество переданных аргументов. Таким образом, для того чтобы использовать аргументы в программе, их просто нужно извлечь из стека. Обработку аргументов нужно проводить в цикле. Т.е. сначала нужно извлечь из стека количество аргументов, а затем циклично для каждого аргумента выполнить логику программы. В качестве примера рассмотрим программу, которая выводит на экран аргументы командной строки (рис. 8). Создадим в каталоге лабораторной работы №9 файл lab9-2 и введем текст из рис. 8.

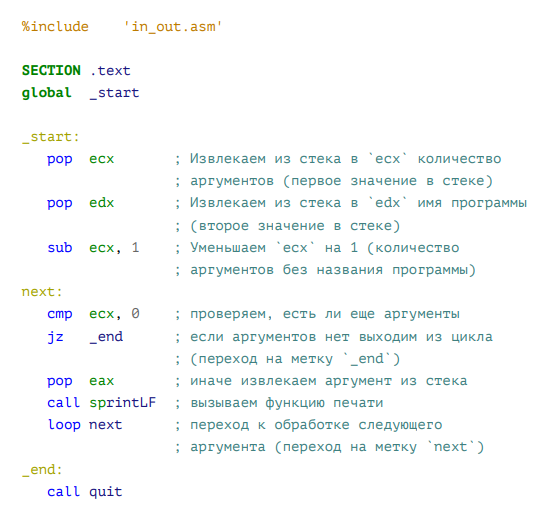


рис. 8. Текст программы lab9-2

Затем создадим исполняемый файл и запустим программу, указав следующие аргументы (рис. 9).

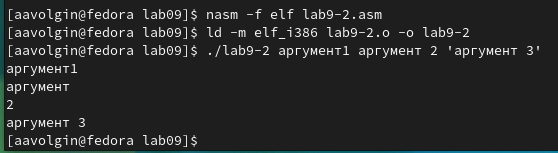


рис. 9. Результат работы программы lab9-2

Как видим, программа восприняла “аргумент” и “2” как отдельные аргументы, в то время как ‘аргумент 3’ как один. Соответственно программой было обработано 4 аргумента.

Рассмотрим еще один пример программы которая выводит сумму чисел, которые передаются в программу как аргументы. Создадим файл lab9-3.asm в том же каталоге и введем в него следующий текст программы (рис. 10).

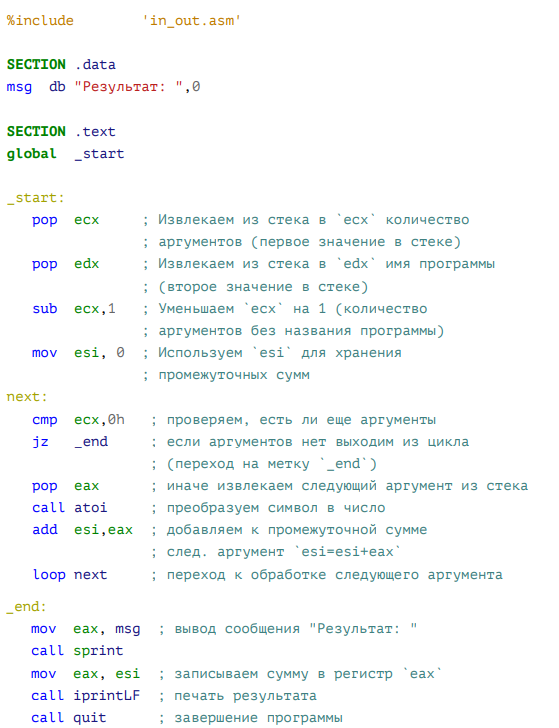


рис. 10. Текст программы lab9-3

Затем создадим исполняемый файл и запустим его, указав аргументы (рис. 11).

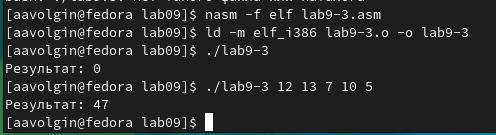


рис. 11. Результат работы программы lab9-3

Как видим, все работает корректно.

Изменим строку

*add esi,ecx*

на

*move ebx, eax*

*mov eax, esi*

*mul ecx*

*mov esi, eax*

а также присвоим esi значение 1, чтобы программа выводила произведение аргументов командной строки и запустим ее (рис. 12).

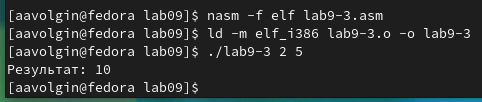


рис. 12. Результат работы программы

**Порядок выполнения самостоятельной работы:**

Напишем 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). Значения 𝑥𝑖 передаются как аргументы. Вид функции 𝑓(𝑥) выберем в соответствии с вариантом, полученным при выполнении лабораторной работы № 7 (вариант 15). Создадим исполняемый файл и проверим его работу на нескольких наборах 𝑥 = 𝑥1 , 𝑥2 , ..., 𝑥𝑛.

f (x)= 6𝑥 + 13

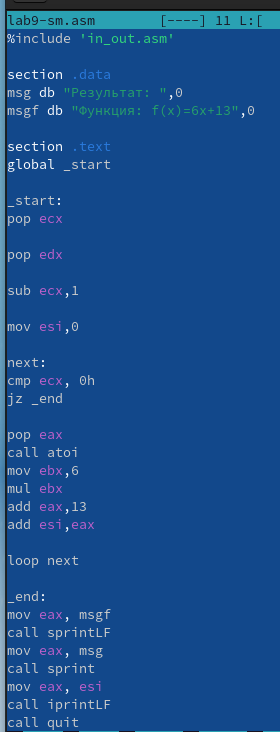


рис. 13. Текст программы lab9-sm

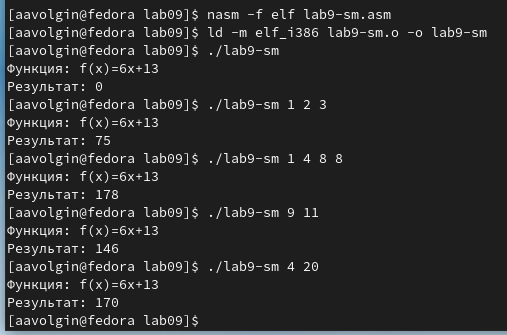


рис. 14. Результат работы программы lab9-sm

Все работает корректно.

**Вывод:**

Во время выполнения лабораторной работы были приобретены навыки написания программ с использованием циклов и обработкой аргументов командной строки.