Отчёта по лабораторной работе №9

Уткина Алина Дмитриевна

Содержание

# 1 Цель работы

Целью данной работы является приобретение навыков написания программ с использованием циклов и обработкой аргументов командной строки.

# 2 Выполнение лабораторной работы

## 2.1 Реализация циклов в NASM

Создадим каталог для программам лабораторной работы № 9, перейдем в него и создадим файл 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. Внимательно изучим текст программы из листинга 9.1 и введем его в файл lab9-1.asm (рис. 1). Создадим исполняемый файл и проверим его работу (рис. 2).

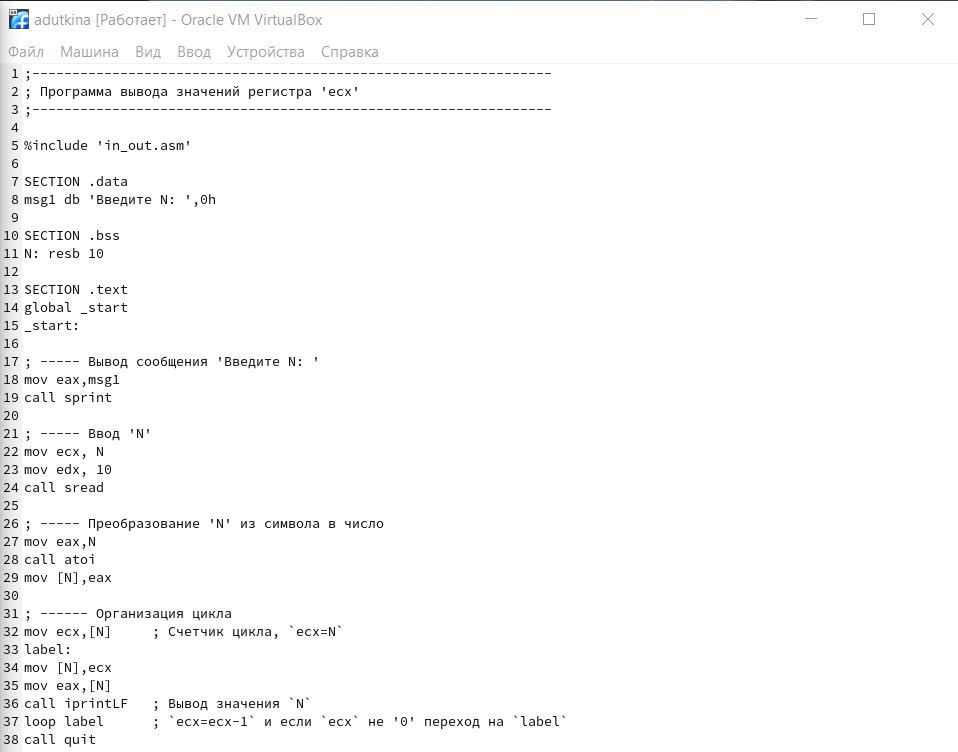


Рис. 1: Программа вывода значений регистра ECX

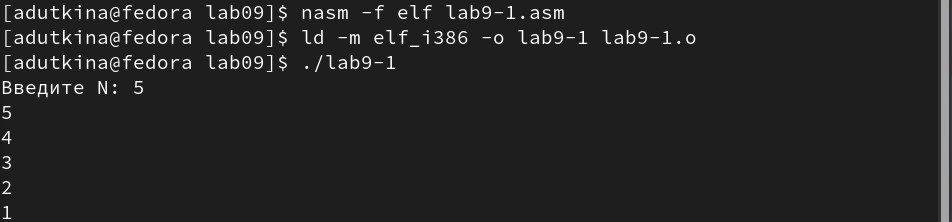


Рис. 2: Вывод значений регистра ECX

Данный пример показывает, что использование регистра ecx в теле цикла loop может привести к некорректной работе программы. Изменим текст программы, добавив изменение значения регистра ecx в цикле (рис. 3).

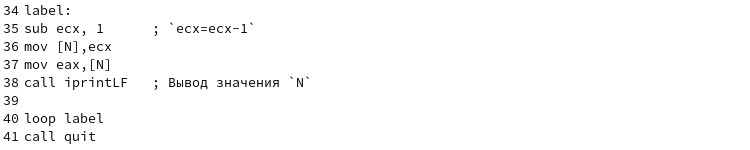


Рис. 3: Изменение текста программы

Создадим исполняемый файл и проверим его работу (рис. 4). Регистр ecx принимает значения на 2 меньше предыдущих. Также, из-за того, что мы ввели нечетное число, 0 не попадает в проверку условия, то есть при одной проверке у нас значение регистра равно 1, а в следующей -1, значит происходит зацикливание. Если мы введем четное число, цикл остановится на 0, сделав в два раза меньше проходов, чем нужно (рис. 5).

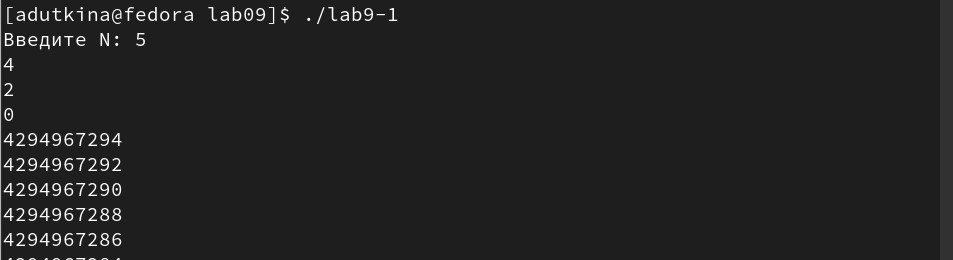


Рис. 4: Результат работы программы (1)



Рис. 5: Результат работы программы (2)

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

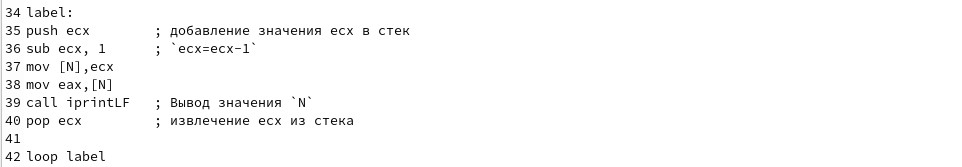


Рис. 6: Добавление команд push и pop в программу

Создадим исполняемый файл и проверим его работу (рис. 7). Количество проходов цикла соответствует введенному значению.

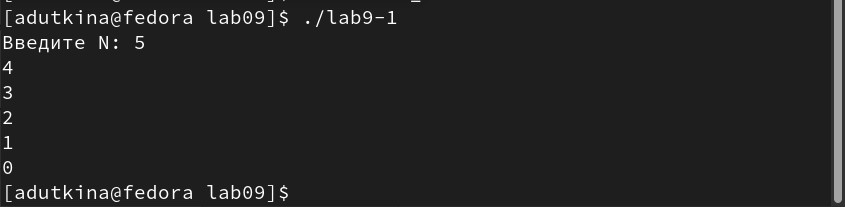


Рис. 7: Результат работы программы с командами push и pop

## 2.2 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. Внимательно изучим текст программы из листинга 9.2 и введем его в файл lab9-2.asm (рис. 8). Создадим исполняемый файл и запустим его, указав аргументы (рис. 9). Все аргументы были обработаны программой.

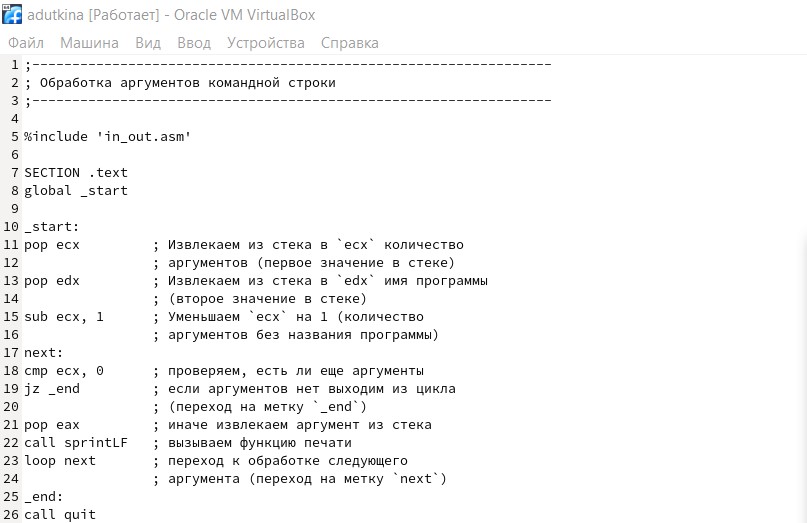


Рис. 8: Программа вывода аргументов командной строки

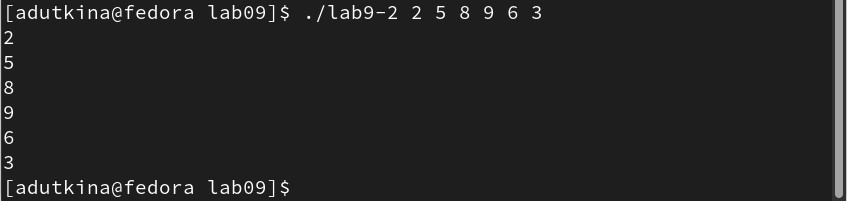


Рис. 9: Результат работы программы с аргументами

Рассмотрим еще один пример программы которая выводит сумму чисел, которые передаются в программу как аргументы. Создадим файл lab9-3.asm и введем в него текст программы из листинга 9.3 (рис. 10). Создадим исполняемый файл и запустим его, указав аргументы (рис. 11).

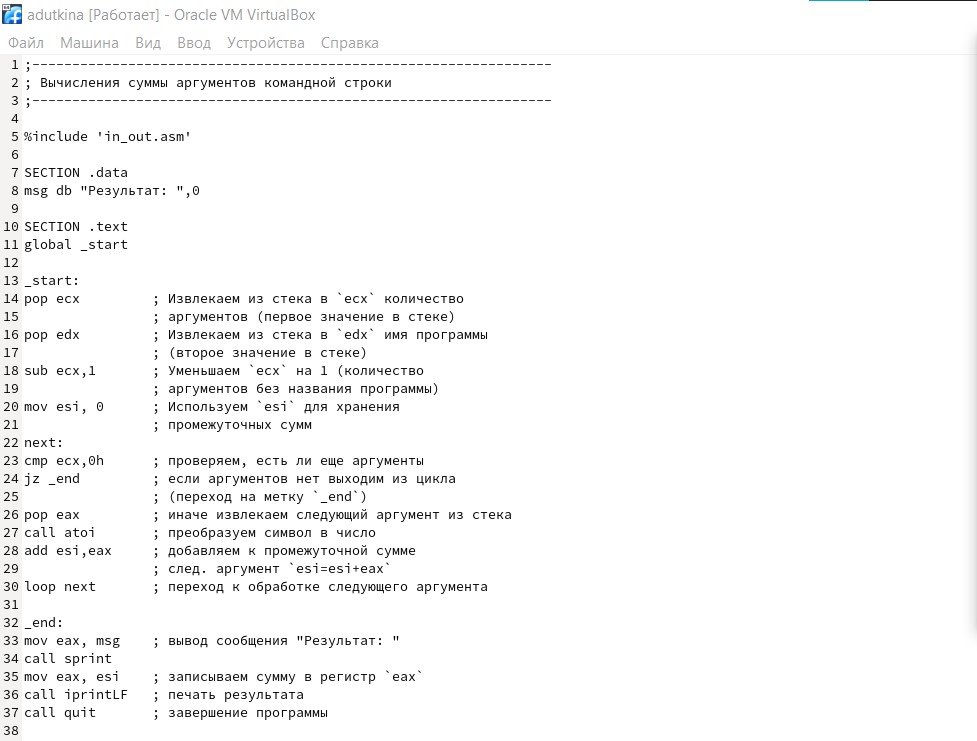


Рис. 10: Программа вычисления суммы аргументов командной сторки

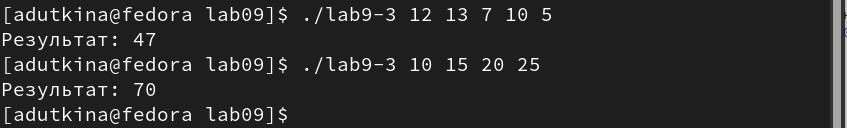


Рис. 11: Результат работы программы вычисления суммы аргументов

## 2.3 Самостоятельная работа

Напишем программу, которая находит сумму значений функции f(x) = 6x + 13 для x = x1, x2, …, xn, т.е. программа должна выводить значение f(x1) + f(x2) +…+ f(xn), где значения xi передаются как аргументы (рис. 12). Создадим исполняемый файл и проверим его работу на нескольких наборах x = x1, x2, …, xn (рис. 13). Программа работает при различном количестве аргументов верно.

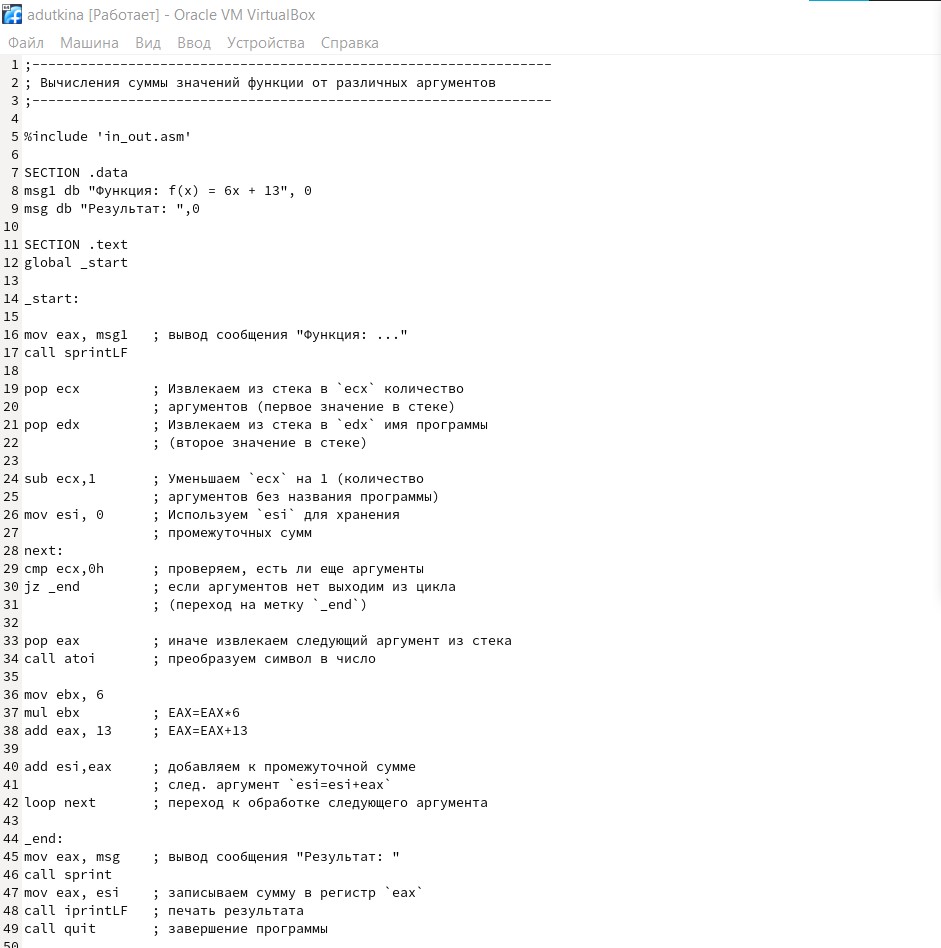


Рис. 12: Программа вычисления суммы значений функции от аргументов

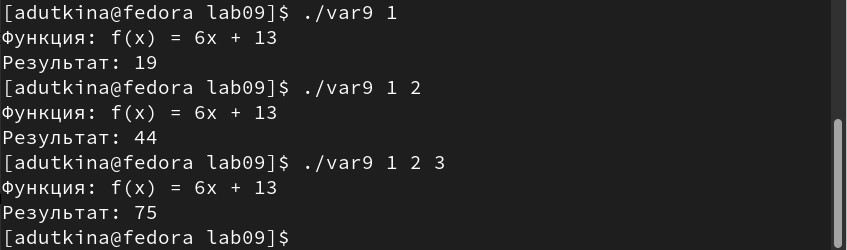


Рис. 13: Результат работы программы

# 3 Выводы

В ходе данной работы были приобретены навыки написания программ с использованием циклов и обработкой аргументов командной строки.