Отчет по лабораторной работе №8

Дисциплина: Архитектура компьютера

Мутаев Муртазаали Магомедович

Содержание

# 1 Цель работы

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

# 2 Задание

1. Реализация циклов NASM
2. Обработка аргументов командной строки
3. Задание для самостоятельной работы

# 3 Выполнение лабораторной работы

## 3.1 Реализация циклов NASM

Я создал каталог для программам лабораторной работы № 8, перешел в него и создал файл lab8-1.asm. Далее ввел текст программы из Листинга 8.1, создал исполняемый файл и запустил его:

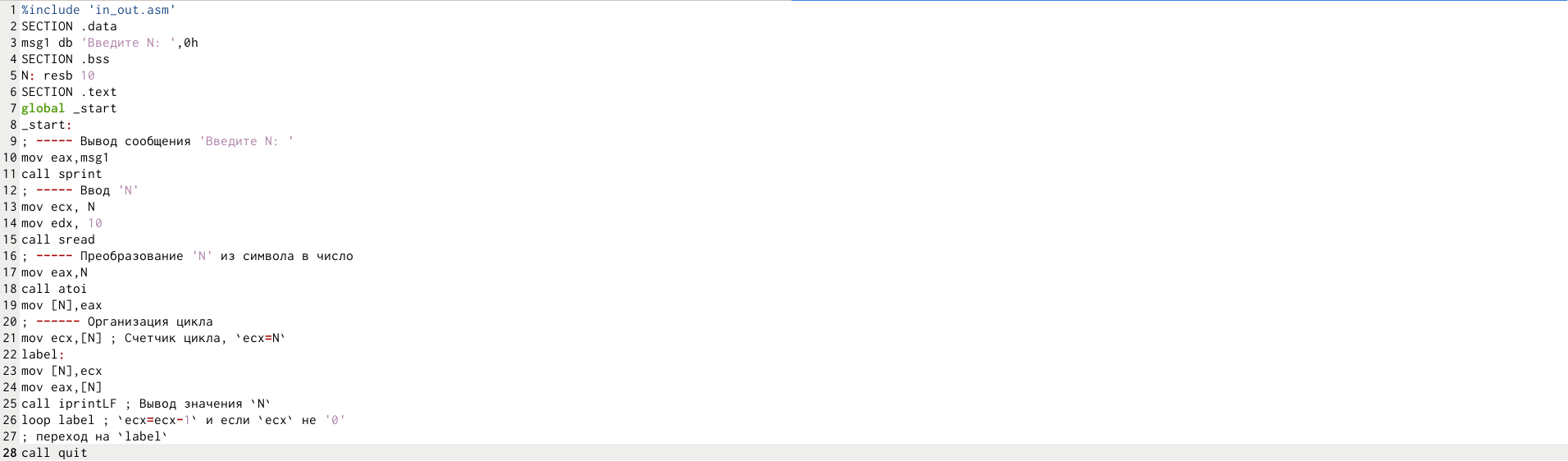


Рис. 1: Листинг 8.1

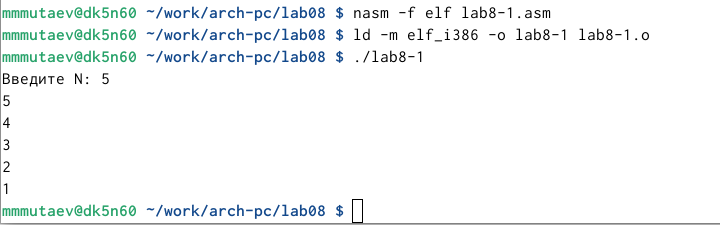


Рис. 2: Результат листинга 8.1

Программа выводит значения регистра ecx. Его значение уменьшается на 1 с каждым проходом цикла.

Добавим в тело цикла уменьшение ecx и проверим работу программы:

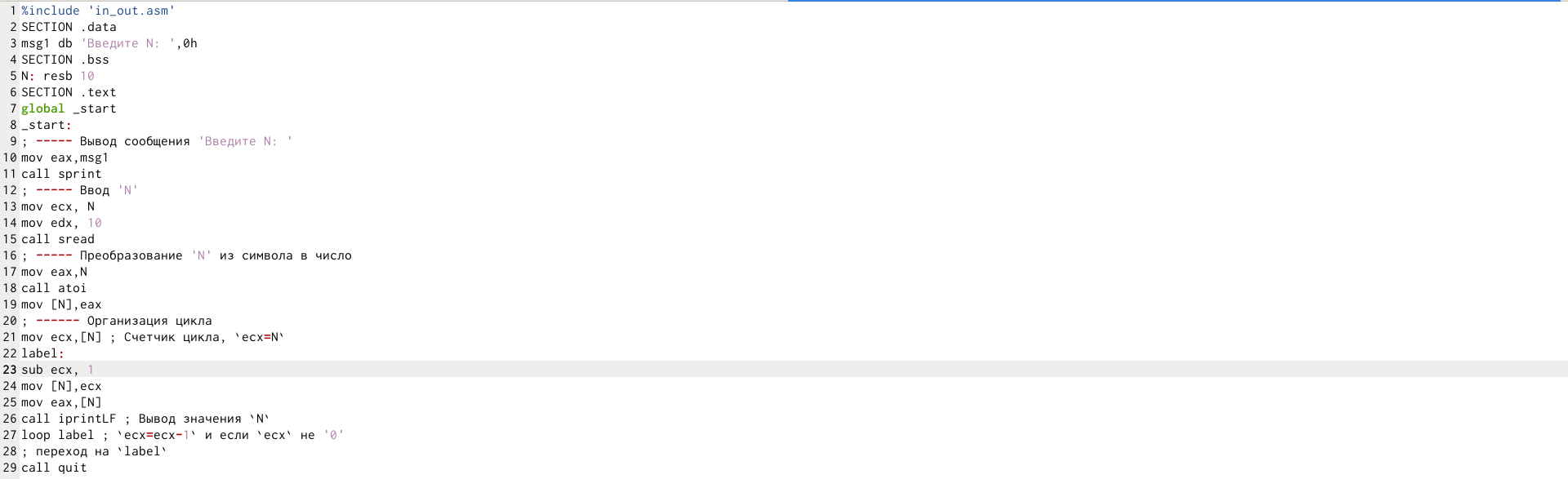


Рис. 3: Попытка обмануть Листинг 8.1



Рис. 4: Результат попытки обмануть Листинг 8.1

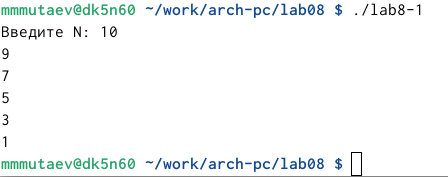


Рис. 5: Удачная попытка обмануть Листинг 8.1

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:

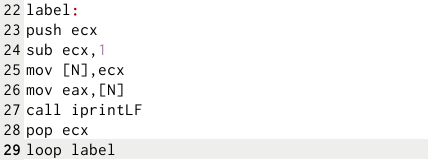


Рис. 6: Использование push и pop

В результате программа выдала мне значение (ecx-1), но в данном случае количество проходов цикла соответствует введенному числу N.

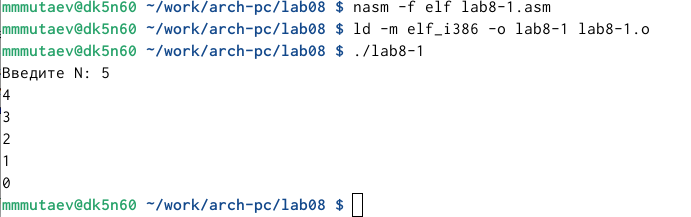


Рис. 7: Pезультат использования push и pop

## 3.2 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 из Листинга 8.2, которая выводит на экран аргументы командной строки:

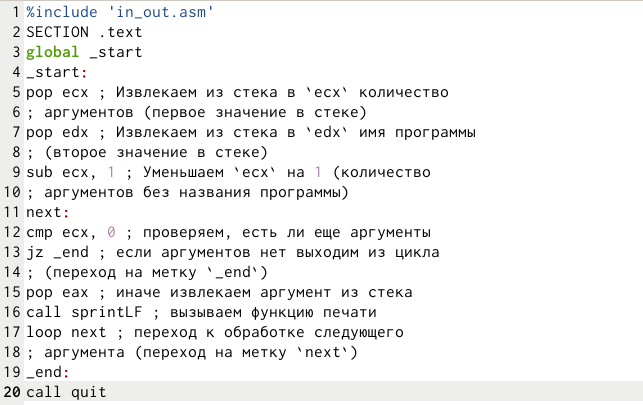


Рис. 8: Листинг 8.2

После запустил программу, введя *./lab8-2 аргумент1 аргумент 2 ‘аргумент 3’*. Мне выдало следующий результат:

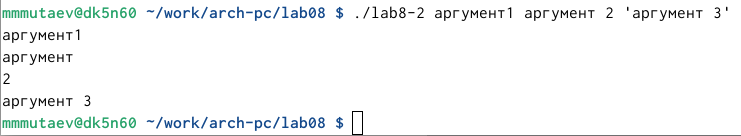


Рис. 9: Результат листинга 8.2

Таким образом программа обработала 4 аргумента.

Рассмотрим еще один пример программы которая выводит сумму чисел, которые передаются в программу как аргументы. Для этого создадим файл lab8-3.asm и введем туда текст из Листинга 8.3:

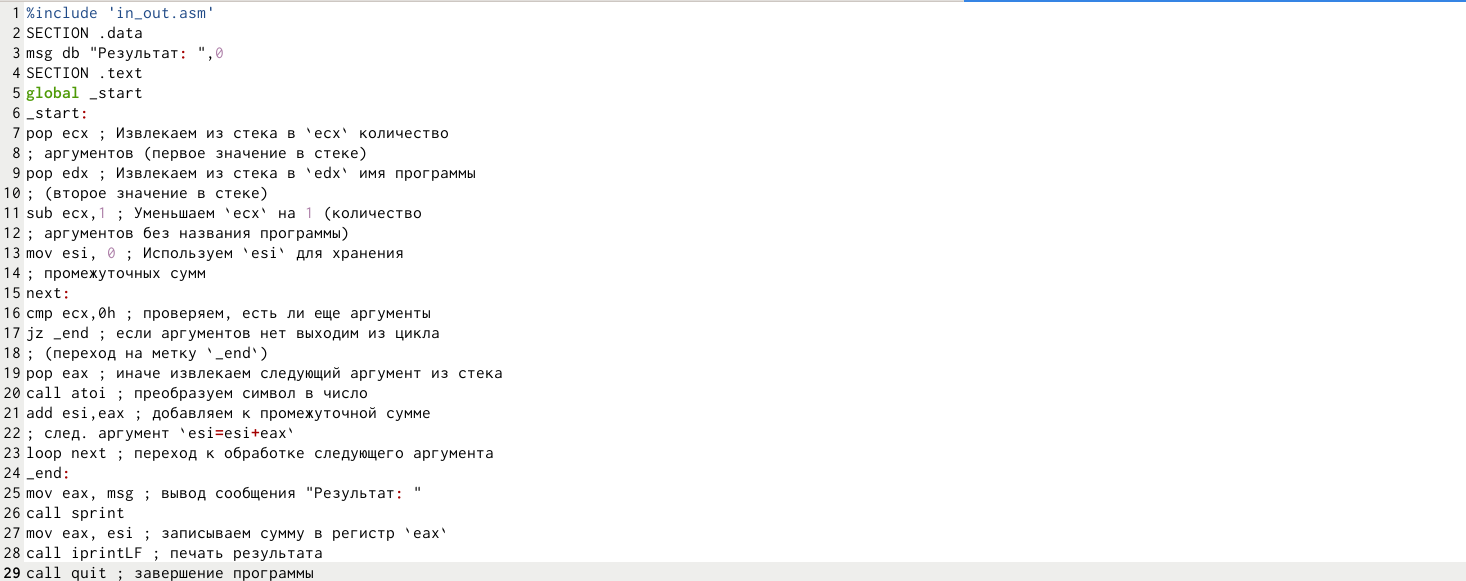


Рис. 10: Листинг 8.3

В результате программа выводит нам сумму введенных чисел:

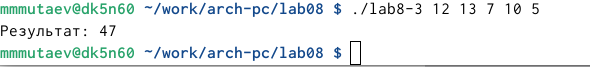


Рис. 11: Результат листинга 8.3

Теперь попробуем самостоятельно изменить текст программы так, чтобы в результате у нас выводилось произведение введенных чисел. Для этого я первоначально присвоил esi значение 1, чтобы при умножении числа результат не обнулялся. После этого заменил строку *add esi, eax* на 2 строки:

*mul esi*

*mov esi, eax*

Т.е. я сначала умножил еах на esi, а потом присвоил esi значение еах.

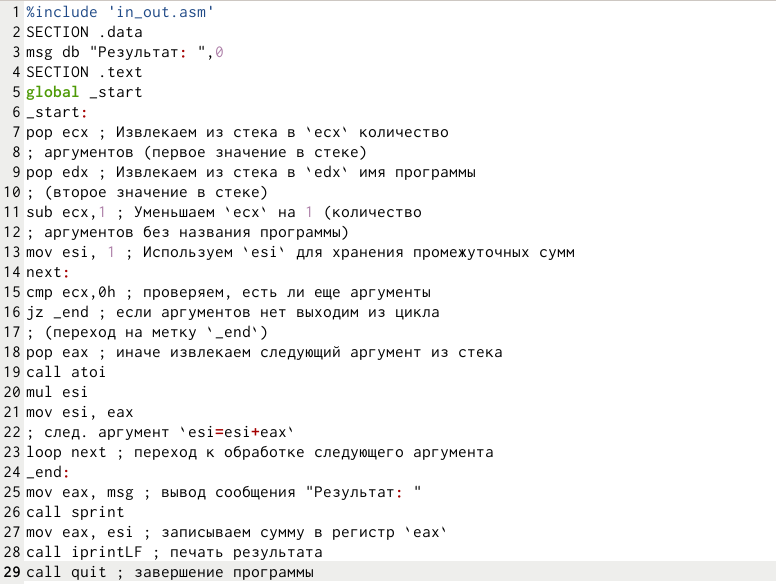


Рис. 12: Листинг 8.3 (Умножение)



Рис. 13: Результат Листинга 8.3 (Умножение)

## 3.3 Задание для самостоятельной работы

Напишите 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)* выбрать из таблицы 8.1 вариантов заданий в соответствии с вариантом, полученным при выполнении лабораторной работы № 7. Создайте исполняемый файл и проверьте его работу на нескольких наборах *x* = *x1, x2, …, xn*.

Возьмем за основу программу для нахождения суммы аргументов. Вытаскивая каждый аргумент из стека, отредактируем их в соответствии с функцией, т.е. подставим их вместо *х*, и уже измененные значения будем добавлять в переменную *esi*. Вот программа, которая у меня получилась:

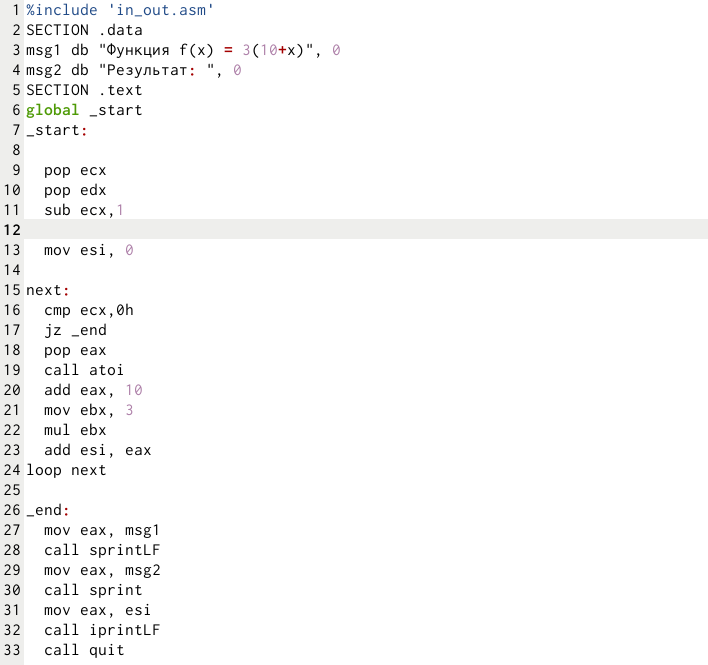


Рис. 14: Задание для самостоятельной работы

И вот наш результат:

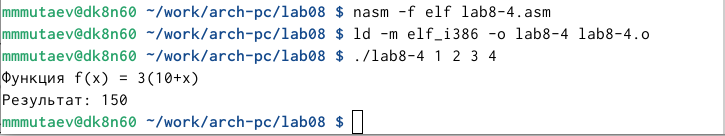


Рис. 15: Задание для самостоятельной работы. Результат

# 4 Выводы

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