Лабораторная работа №8

Программирование цикла. Обработка аргументов командной строки.

Перфилов Александр Константинович | группа: НПИбд 02-23

Содержание

# 1 Цель работы

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

# 2 Выполнение лабораторной работы

**Реализация циклов в NASM**

Создадим каталог для программам лабораторной работы № 8, перейдем в него и создадим файл lab8-1.asm

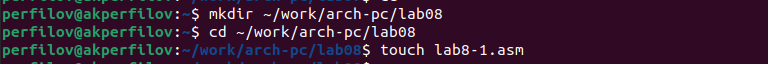


Рис 2.1.1: Создание каталога и файла .asm

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 (Листинг 8.1).

Введем в файл lab8-1.asm текст программы из листинга 8.1.

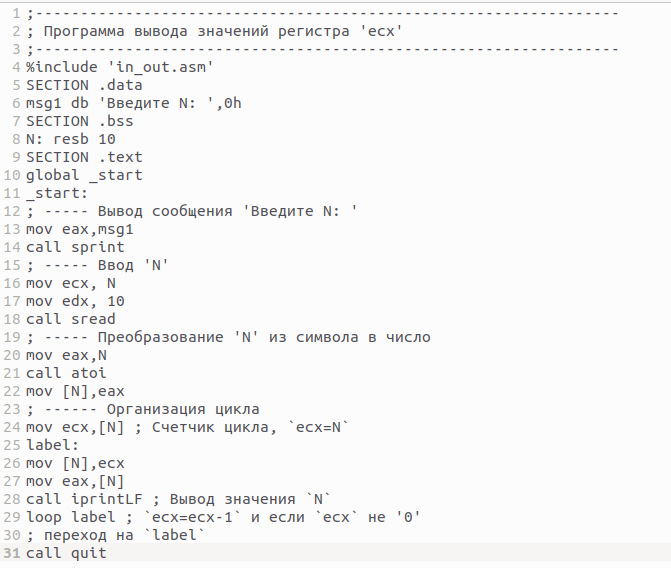


Рис 2.1.2: Программа

Создадим исполняемый файл и запустим его. Результат работы данной программы будет следующим:

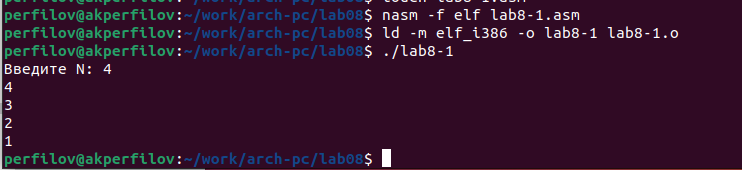


Рис 2.1.3: Создание исполняемого файла и проверка работы

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:

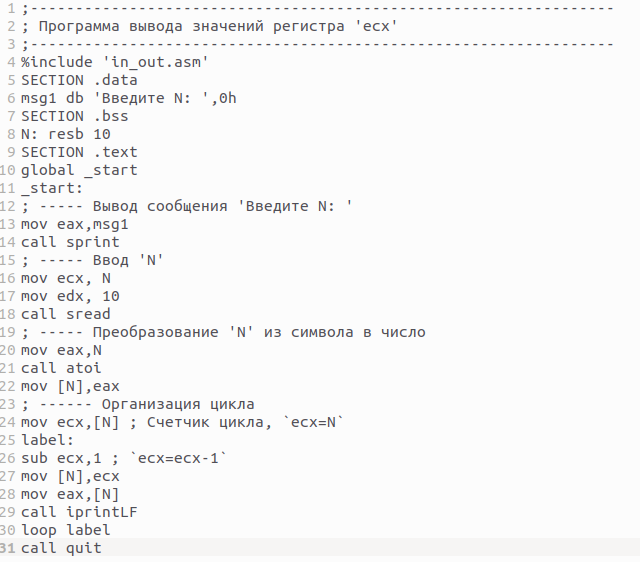


Рис 2.1.4: Измененная программа

Создадим исполняемый файл и проверим его работу.

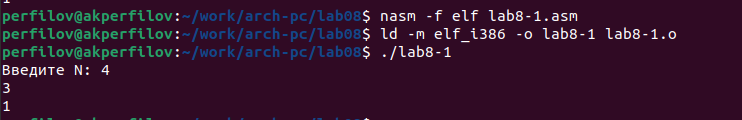


Рис 2.1.5: Создание исполняемого файла и проверка работы

Если ввести 4, то число проходов цикла не будет соответствовать введенному с клавиатуры значению

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:

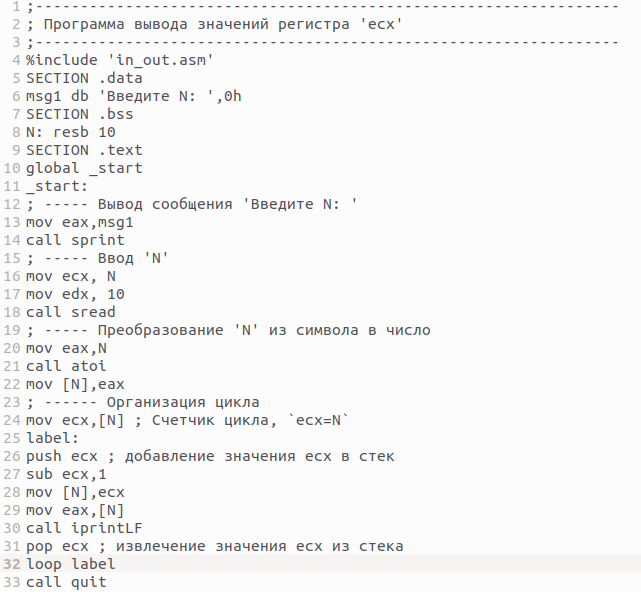


Рис 2.1.6: Измененная программа

Создадим исполняемый файл и проверим его работу.

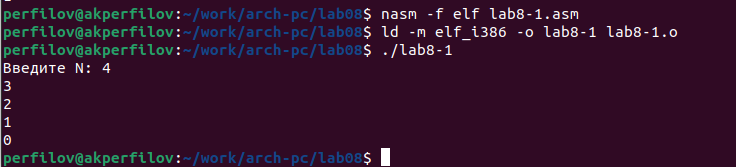


Рис 2.1.7: Создание исполняемого файла и проверка работы

Число проходов цикла будет соответствовать введенному значению 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. Внимательно изучим текст программы (Листинг 8.2)

Создадим файл lab8-2.asm в каталоге ~/work/arch-pc/lab08 и введем в него текст программы из листинга 8.2.

Рис 2.2.1: Создание файла .asm

Рис 2.2.1: Создание файла .asm

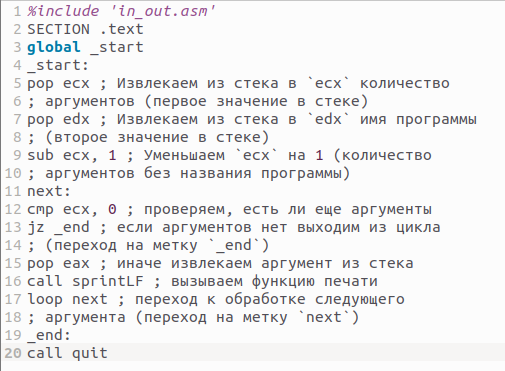


Рис 2.2.2: Программа

Создадим исполняемый файл и запустим его, указав аргументы:

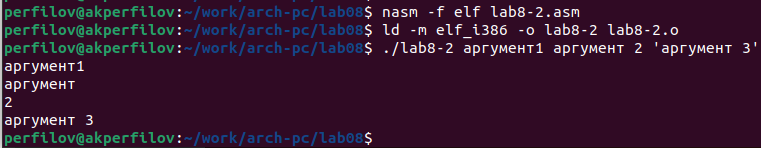


Рис 2.2.3: Создание исполняемого файла и проверка работы с указанием аргументов

Всего было обработано 4 аргумента, так как второй и третий аргумент не были взяты в кавычки, в отличии от 4-го аргумента

Рассмотрим еще один пример программы которая выводит сумму чисел, которые передаются в программу как аргументы. Создадим файл lab8-3.asm в каталоге ~/work/arch-pc/lab08 и введем в него текст программы из листинга 8.3.

Рис 2.2.4: Создание файла .asm

Рис 2.2.4: Создание файла .asm

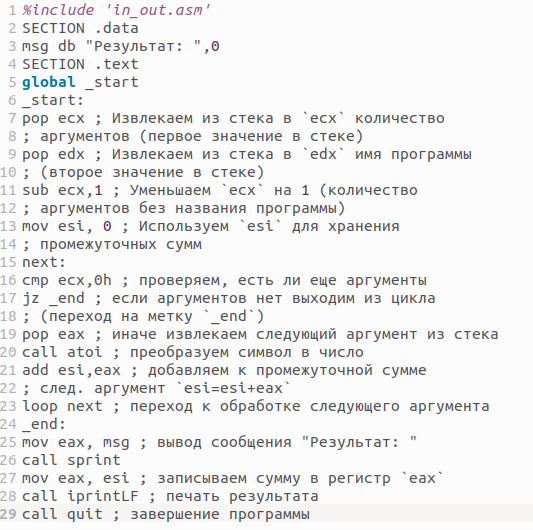


Рис 2.2.5: Программа

Создадим исполняемый файл и запустим его, указав аргументы.

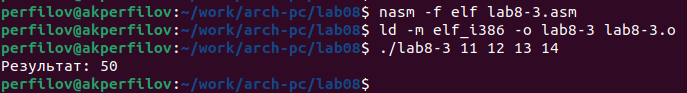


Рис 2.2.6: Создание исполняемого файла и проверка работы с указанием аргументов

Изменим текст программы из листинга 8.3 для вычисления произведения аргументов командной строки.

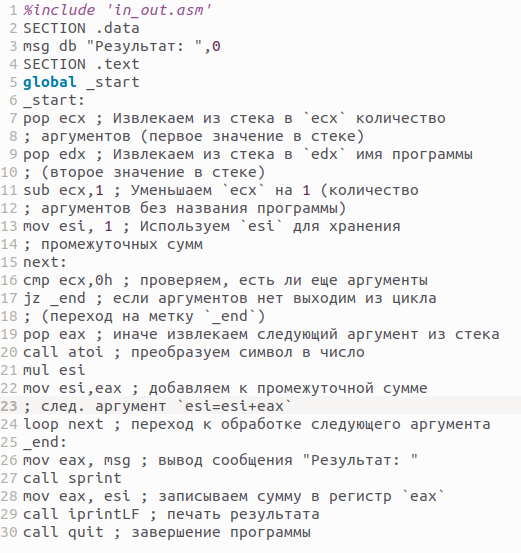


Рис 2.2.7: Измененная программа

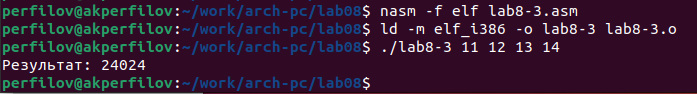


Рис 2.2.8: Создание исполняемого файла и проверка работы с указанием аргументов

# 3 Самостоятельная работа

Задание№1 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.

Создадим новый файл ex1.asm и напишем программу нахождения суммы значения функции f(x) для варианта 19 ().

Рис 3.1.1: Создание файла

Рис 3.1.1: Создание файла

Возьмем за основу код из lab8-3.asm и переделаем его под 1-ое задание

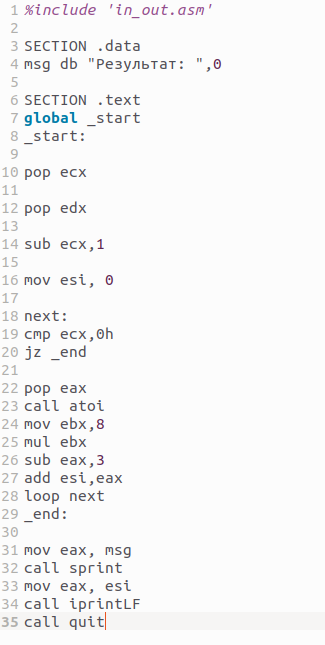


Рис 3.1.2: Программа для задания

Проверим программу с несколькими значениями x

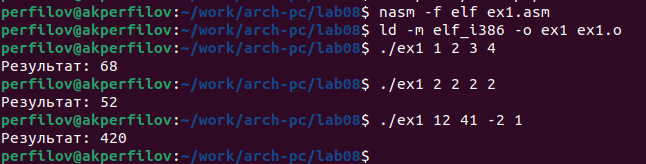


Рис 3.1.3: Проверка программы

Как видно по (Рис 3.1.3) программа работает верно

Загрузим все файлы на github

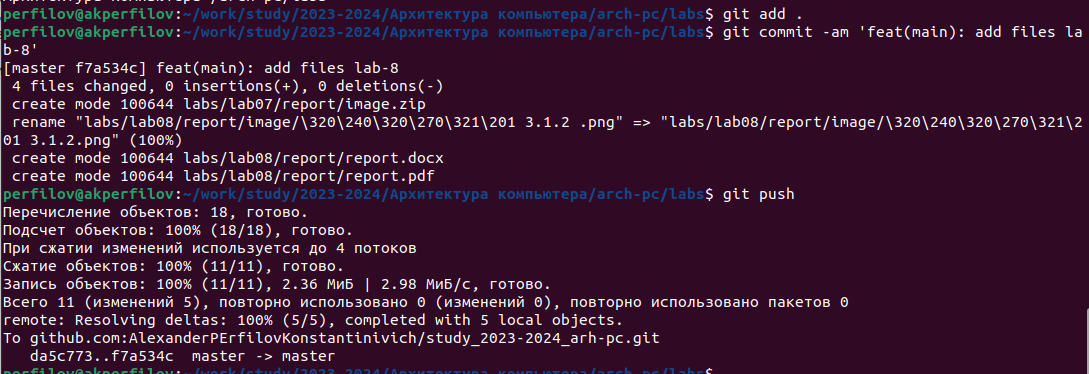


Рис 3.1.4: Загрузка файлов на github

# 4 Выводы

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