Лабораторная работа №9

Архитектура компьютера

Голованова Мария Константиновна

Содержание

# 1 Цель работы

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

# 2 Задание

Написать программы с использованием циклов и обработкой аргументов командной строки. Написать программу, которая находит сумму значений функции f(x) для x = x1, x2, …, xn (т. е. находит f(x1) + f(x2) +…+ f(xn)), где значения xi передаются как аргументы.

# 3 Теоретическое введение

Стек — это структура данных, организованная по принципу LIFO («Last In — First Out» или «последним пришёл — первым ушёл»). Основной функцией стека является функция сохранения адресов возврата и передачи аргументов при вызове процедур. Также в нём выделяется память для локальных переменных и могут временно храниться значения регистров. Стек имеет вершину, адрес последнего добавленного элемента, который хранится в регистре esp. Противоположный конец стека называется дном. Последнее добавленное в стек значение извлекается первым. При помещении значения в стек указатель стека уменьшается, а при извлечении — увеличивается. Для работы со стеком в процессоре есть специальные регистры (ss, bp, sp) и команды. Также для стека существует две основные операции: добавление элемента в вершину стека (push) и извлечение элемента из вершины стека (pop). Для организации циклов существуют специальные инструкции, для каждой максимальное количество проходов задаётся в регистре ecx. Наиболее простой является инструкция loop, которая позволяет организовать безусловный цикл. Она 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, если равен, то переход не выполняется и управление передаётся команде, следующей сразу после команды loop.

# 4 Выполнение лабораторной работы

## 4.1 Реализация циклов в NASM

Я создала каталог для программам лабораторной работы №9, перешла в него и создала файл lab9-1.asm (рис. 1).

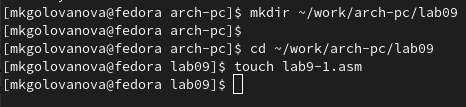


Рис. 1: Создание каталога для программам лабораторной работы №9 и файла lab9-1.asm

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 я рассмотрела программу, выводящую значение регистра ecx. Я внимательно изучила текст программы листинга 9.1 и ввела его в файл lab9-1.asm (рис. 2).

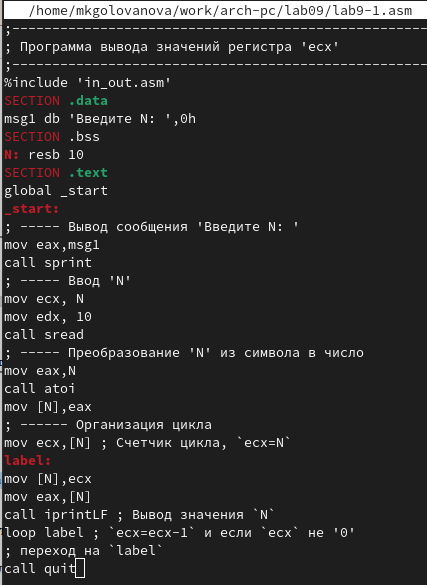


Рис. 2: Введение текста программы из листинга 9.1

Я создала исполняемый файл и проверила его работу (рис. 3).

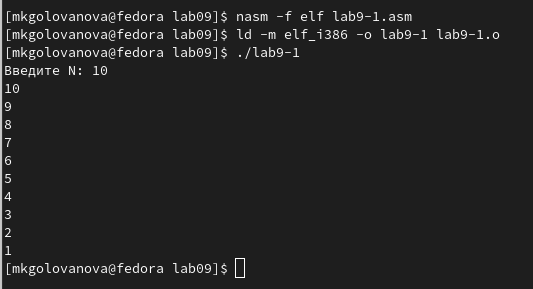


Рис. 3: Создание и запуск исполняемого файла lab9-1

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

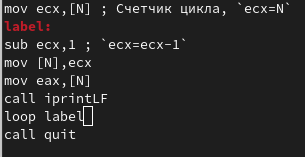


Рис. 4: Изменение текста файла lab9-1.asm

Я создала исполняемый файл и проверила его работу (рис. 5). Регистр ecx принимает в цикле значения, которые не могут входить в N (слишком большие для введённого N). Число проходов цикла не соответствует значению N, введенному с клавиатуры.

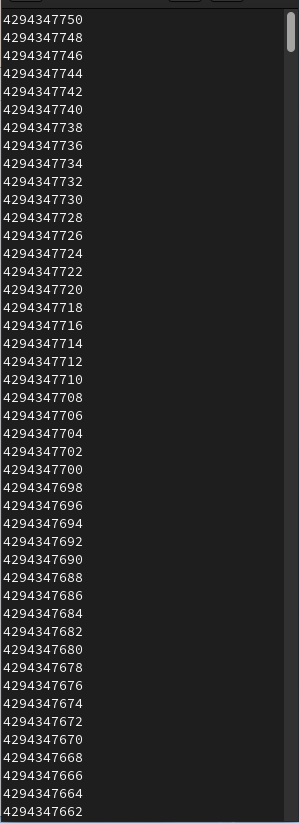


Рис. 5: Результат работы изменённого файла lab9-1.asm

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

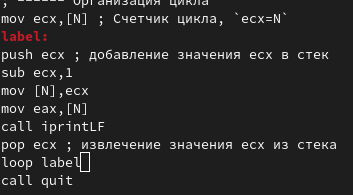


Рис. 6: Изменение текста файла lab9-1.asm

Я создала исполняемый файл и проверила его работу (рис. 7). В данном случае число проходов цикла соответствует значению N, введенному с клавиатуры.

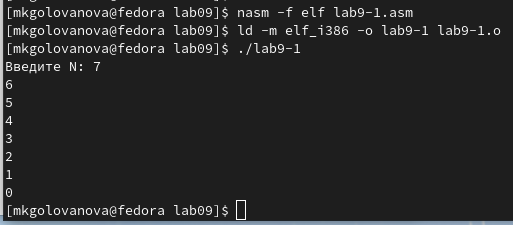


Рис. 7: Создание и запуск изменённого исполняемого файла lab9-1

## 4.2 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 я рассмотрела программу, которая выводит на экран аргументы командной строки. Я создала файл lab9-2.asm в каталоге ~/work/arch-pc/lab09 (рис. 8).

Рис. 8: Создание файла lab9-2.asm в каталоге ~/work/arch-pc/lab09

Рис. 8: Создание файла lab9-2.asm в каталоге ~/work/arch-pc/lab09

Я внимательно изучила текст программы листинга 9.2 и ввела его в файл lab9-2.asm (рис. 9)

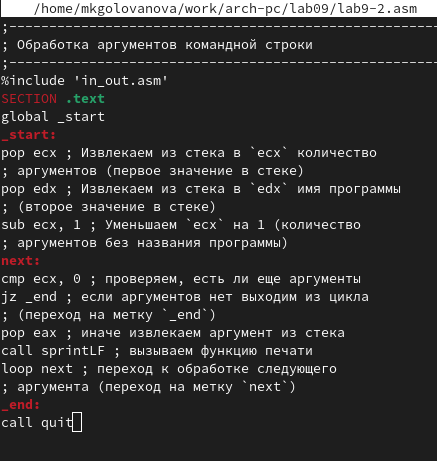


Рис. 9: Введение текста программы из листинга 9.2

Я создала исполняемый файл и проверила его работу (рис. 10). Программа обработала 4 аргумента.

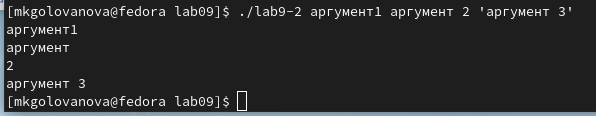


Рис. 10: Создание и запуск исполняемого файла lab9-2

Я рассмотрела еще один пример программы, которая выводит сумму чисел, которые передаются в программу как аргументы. Я создала файл lab9-3.asm в каталоге ~/work/arch-pc/lab09 (рис. 11) и ввела в него текст программы из листинга 9.3(рис. 12)

Рис. 11: Создание файла lab9-3.asm в каталоге ~/work/arch-pc/lab09

Рис. 11: Создание файла lab9-3.asm в каталоге ~/work/arch-pc/lab09

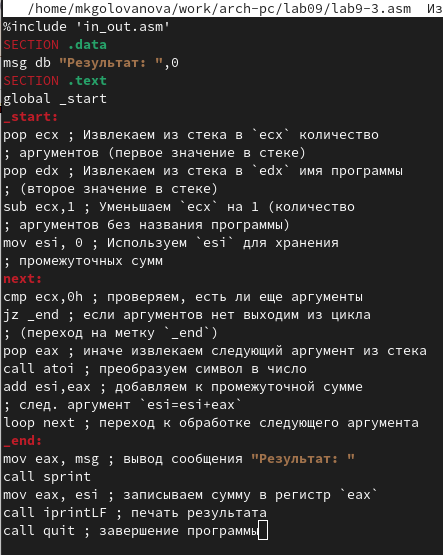


Рис. 12: Введение текста программы из листинга 9.3

Я создала исполняемый файл и запустила его, указав аргументы (рис. 13)

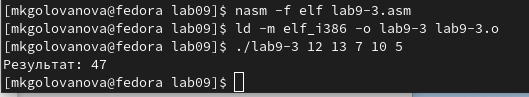


Рис. 13: Создание исполняемого файла lab9-3 и его запуск с указанием аргументов

Я изменила текст программы из листинга 9.3 для вычисления произведения аргументов командной строки (рис. 14).

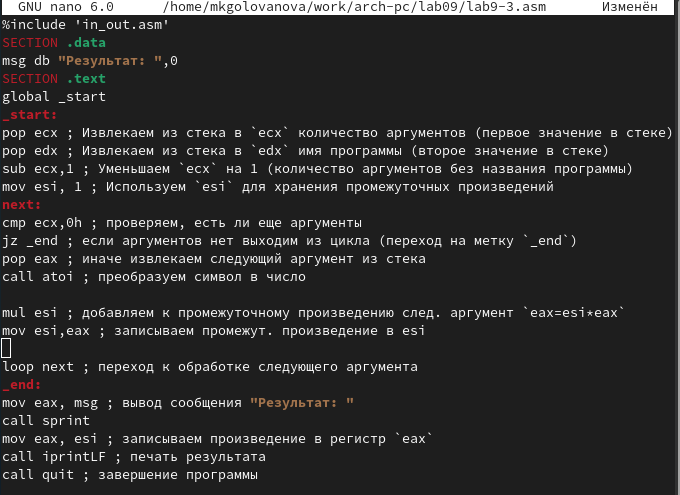


Рис. 14: Изменение текста файла lab9-3.asm для вычисления произведения аргументов командной строки

Я создала новый исполняемый файл и запустила его, указав аргументы (рис. 15)

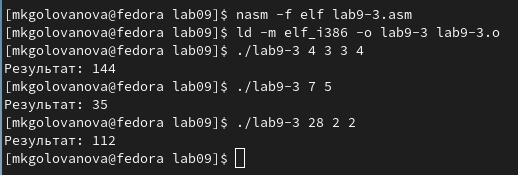


Рис. 15: Создание нового исполняемого файла lab9-3 и его запуск с указанием аргументов

# 5 Задание для самостоятельной работы

Я написала программу, которая находит сумму значений функции f(x) для x = x1, x2, …, xn, т.е. программа выводит значение f(x1) + f(x2) +…+ f(xn). Значения xi передаются как аргументы. Вид функции f(x) я выбрала из таблицы 9.1 вариантов заданий в соответствии с вариантом, полученным при выполнении лабораторной работы №7 (вариант 19). Я создала файл lab9-4-var19.asm в каталоге ~/work/arch-pc/lab09 (рис. 16) и ввела в него текст составленной программы (рис. 17).

Рис. 16: Создание файла lab9-4-var19.asm в каталоге ~/work/arch-pc/lab09

Рис. 16: Создание файла lab9-4-var19.asm в каталоге ~/work/arch-pc/lab09

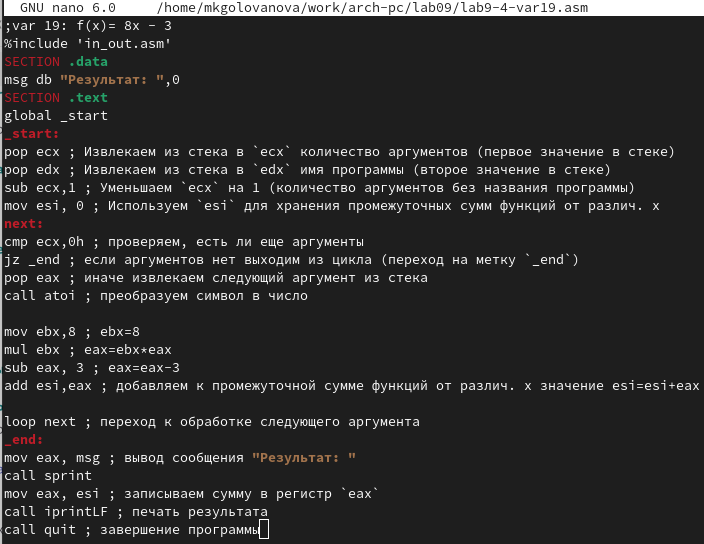


Рис. 17: Текст составленной программы в файле lab9-4-var19.asm

Я создала исполняемый файл и проверила его работу на нескольких наборах x = x1, x2, …, xn (рис. 18).

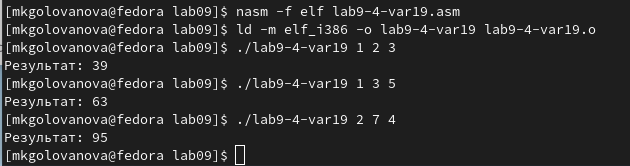


Рис. 18: Создание исполняемого файла lab9-3 и его проверка на нескольких наборах x = x1, x2, …, xn

# 6 Выводы

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

# Список литературы