Лабораторная работа №8

Архитектура компьютера

Казначеева Кристина Никитична

Содержание

# 1 Цель работы

Лабораторная работа направлена на практическое освоение программирования с использованием циклов и обработки аргументов командной строки.

# 2 Задание

В данной лабораторной работе мы изучим реализацию циклов в NASM, программы вывода значений регистра ecx и вычисления суммы аргументов командной строки, а также программу, выводящую на экран аргументы командной строки. Научимся обработке аргументов командной строки.

# 3 Выполнение лабораторной работы

Создадим каталог lab08 (рис. 1).

Рис. 1: Создание каталога

Рис. 1: Создание каталога

Перейдём в этот каталог и создадим файл lab8-1.asm (рис. 2).

Рис. 2: Создание файла

Рис. 2: Создание файла

Введём в файл lab8-1.asm текст программу вывода значений регистра ecx (рис. 3).

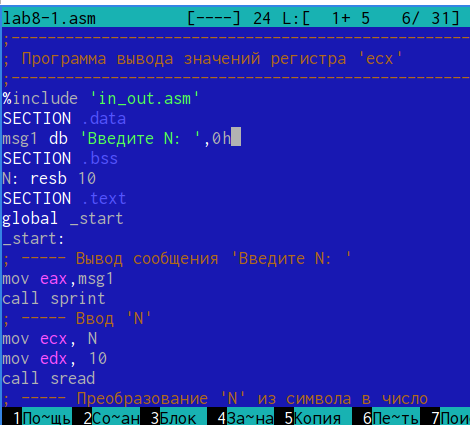


Рис. 3: Ввод текста программы

Создадим исполняемый файл и запустим его (рис. 4).

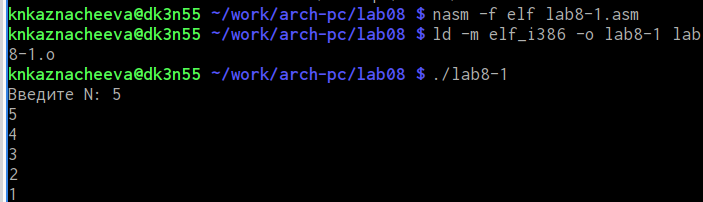


Рис. 4: Проверка работы исходного файла

Затем изменим текст программы, добавив изменение значение регистра ecx в цикле (рис. 5).

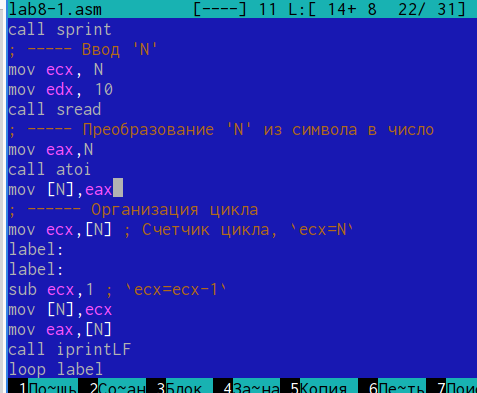


Рис. 5: Изменение текста программы

Запустим исполняемый файл и проверим его. Регистр ecx принимает в цикле значения, уменьшающиеся на 1 на каждой итерации цикла, начиная с начального значения, которое задано до начала цикла. Число проходов цикла точно соответствует значению N, которое было загружено в ecx до начала цикла. Цикл выполняется N раз( (рис. 6).

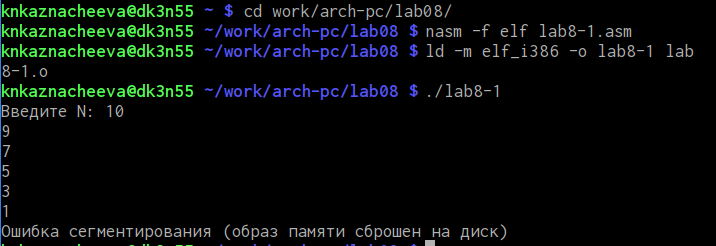


Рис. 6: Проверка работы исходного файла

Внесём изменения в текст программы, добавив команды push и pop (добавления в стек и извлечения из стека) для сохранения значения счетчика цикла loop (рис. 7).

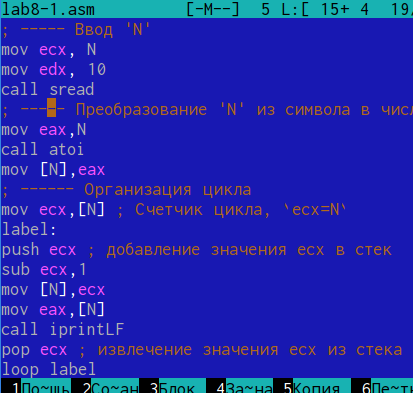


Рис. 7: Изменение текста программы

Создадим исполняемый файл и проверим его работу. При добавлении команды push и pop число проходов цикла в измененном коде не будет точно соответствовать значению N, введенному с клавиатуры. (рис. 8):

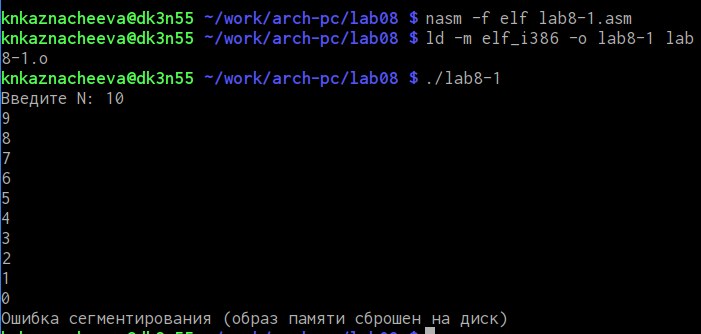


Рис. 8: Проверка работы исходного файла

Создадим файл lab8-2.asm в каталоге ~/work/arch-pc/lab08 (рис. 9).

Рис. 9: Создание файла

Рис. 9: Создание файла

Введём в него текст программы, выводящей на экран аргументы командной строки (рис. 10).

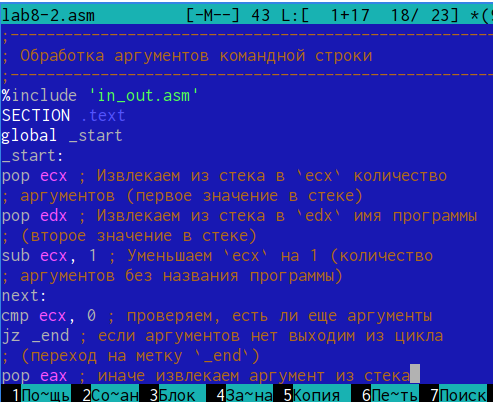


Рис. 10: Ввод текста программы

Создадим исполняемый файл и запустим его, указав аргументы: аргумент1 аргумент 2 ‘аргумент 3’. В результате было обработано 3 аргумента (рис. 11).

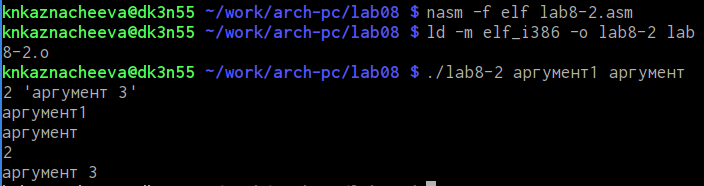


Рис. 11: Проверка работы исходного файла

Создадим файл lab8-3.asm в каталоге ~/work/arch-pc/lab08 (рис. 12).

Рис. 12: оздание файла

Рис. 12: оздание файла

Ведём в него текст программы вычисления суммы аргументов командной строки (рис. 13).

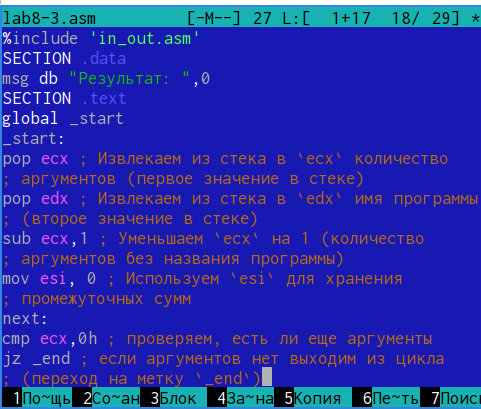


Рис. 13: Ввод текста программы

Создадим исполняемый файл и запустим его, указав аргументы (рис. 14).

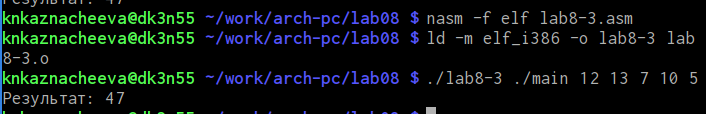


Рис. 14: Проверка работы исходного файла

Изменим текст программы для вычисления произведения аргументов командной строки (рис. 15).

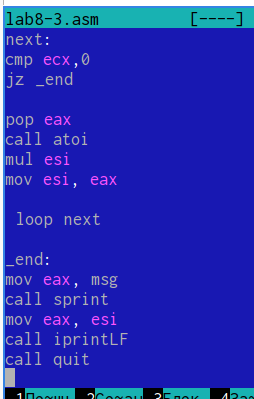


Рис. 15: Изменение текста программы

Cоздадим исполняемый файл и проверим его работу (рис. 16).

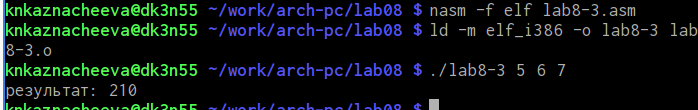


Рис. 16: Проверка работы исходного файла

Создадим файл lab8-4.asm (рис. 17).

Рис. 17: Создание файла

Рис. 17: Создание файла

Напишем программу, которая находит сумму значений функции f(x)=7(x+1) (вариант 14) для x = x1, x2, …, xn, т.е. программа должна выводить значение f(x1) + f(x2) + … + f(x3), где значения x передаются как аргументы (рис. 18).

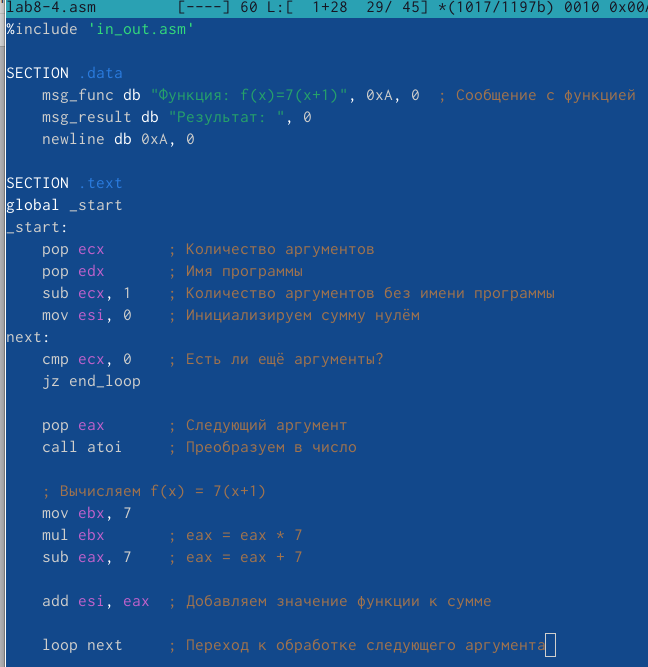


Рис. 18: Ввод текста программы

Создадим исполняемый файл и проверим его работу для значений x: 1, 2, 3, 4 (рис. 19).

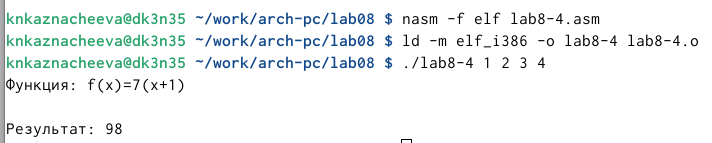


Рис. 19: Проверка работы исходного файла

1. Команда loop реализует цикл, который повторяется ecx раз. Она выполняет три действия: • Уменьшает значение регистра ecx на 1. Регистр ecx обычно используется как счетчик цикла. • Проверяет значение регистра ecx. Если ecx не равно нулю, то происходит переход к указанной метке. • Переход к метке. Если ecx не равно нулю, выполнение переходит к метке, указанной после команды loop. Если ecx равно нулю, выполнение продолжается с инструкции, следующей за командой loop.
2. Для организации цикла без использования специальных команд управления циклами, таких как loop, используются условные переходы, например, jnz (jump if not zero) или jz (jump if zero).
3. Стек — это структура данных типа LIFO (Last-In, First-Out — последний вошел, первый вышел). В программировании стек используется для временного хранения данных, например, адресов возврата при вызовах функций, локальных переменных и промежуточных результатов вычислений.
4. Данные извлекаются из стека в порядке, обратном порядку их занесения (LIFO). Последний элемент, добавленный в стек (то есть, находящийся на вершине стека), является первым элементом, который будет извлечен. Операции добавления и извлечения данных в стеке часто называются push (добавление) и pop (извлечение).

# 4 Вывод

В рамках лабораторной работы были изучены команды условного и безусловного перехода, структура файлов листинга и особенности реализации переходов в среде NASM. Были получены практические навыки написания программ с использованием переходов.