Отчет по лабораторной работе №8

Дисциплина: Архитектура компьютера

Гашимова Эсма Эльшан кызы

Содержание

# 1 Цель работы

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

# 2 Задание

1. Реализация циклом в NASM
2. Обработка аргументов командной строки
3. Самостоятельное написание программы по материалам лабораторной работы

# 3 Теоретическое введение

Стек — это структура данных, организованная по принципу LIFO («Last In — First Out» или «последним пришёл — первым ушёл»). Стек является частью архитектуры процессора и реализован на аппаратном уровне. Для работы со стеком в процессоре есть специальные регистры (ss, bp, sp) и команды. Основной функцией стека является функция сохранения адресов возврата и передачи аргументов при вызове процедур. Кроме того, в нём выделяется память для локальных переменных и могут временно храниться значения регистров.

# 4 Выполнение лабораторной работы

## 4.1 Реализация циклов в NASM

Создаю каталог для программ лабораторной работы №8 (рис. 1).

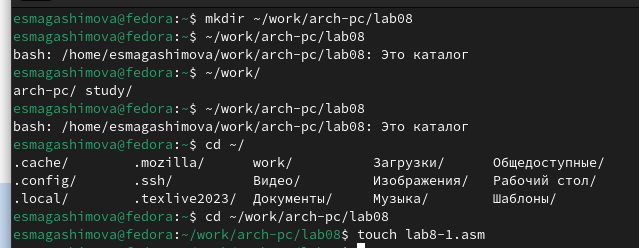


Рис. 1: Создание каталога

Копирую в созданный файл программу из листинга и запускаю программу, она показывает работу циклов в NASM (рис. 2).

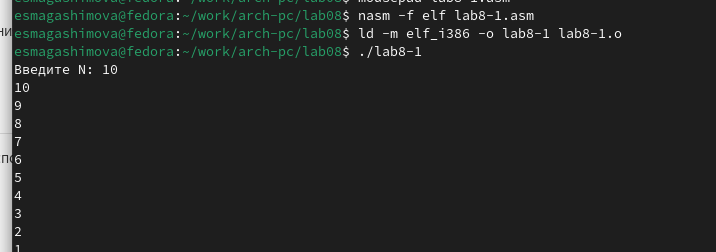


Рис. 2: Запуск программы

Заменяю программу изначальную так, что в теле цикла изменяю значение регистра ecx (рис. 3).



Рис. 3: Изменение программы

Из-за того, что теперь регистр ecx на каждой итерации уменьшается на 2 значения, количество итераций уменьшается вдвое (рис. 4).

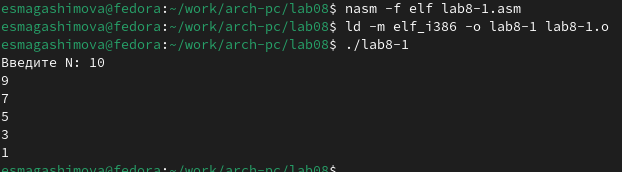


Рис. 4: Запуск измененной программы

Добавляю команды push и pop в программу (рис. 5).

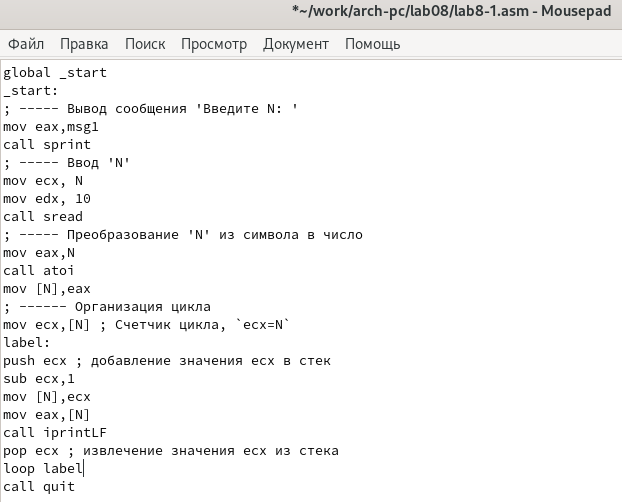


Рис. 5: Добавление push и pop в цикл программы

Теперь количество итераций совпадает введенному N, но произошло смещение выводимых чисел на -1 (рис. 6).

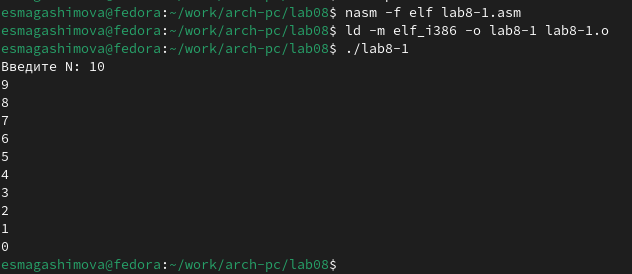


Рис. 6: Запуск измененной программы

## 4.2 Обработка аргументов командной строки

Создаю новый файл для программы и копирую в него код из следующего листинга (рис. 7).

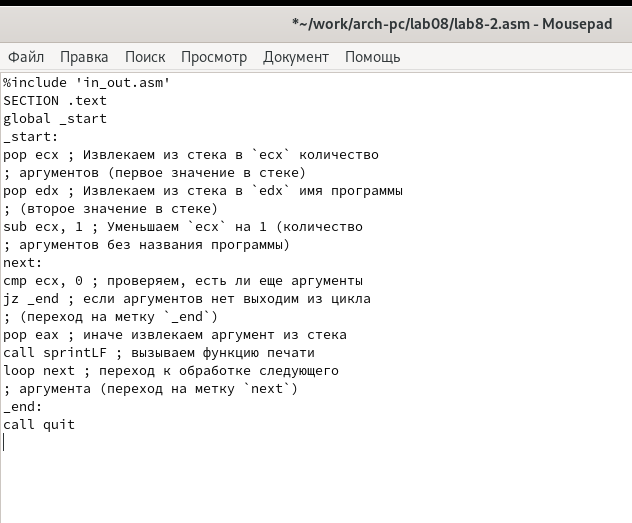


Рис. 7: Копирование программы из листинга

Компилирую программу и запускаю, указав аргументы. Программой было обработано то же количество аргументов, что и было введено (рис. 8).

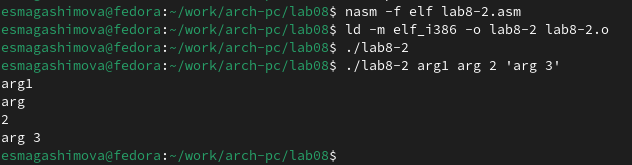


Рис. 8: Запуск второй программы

Создаю новый файл для программы и копирую в него код из третьего листинга (рис. 9).

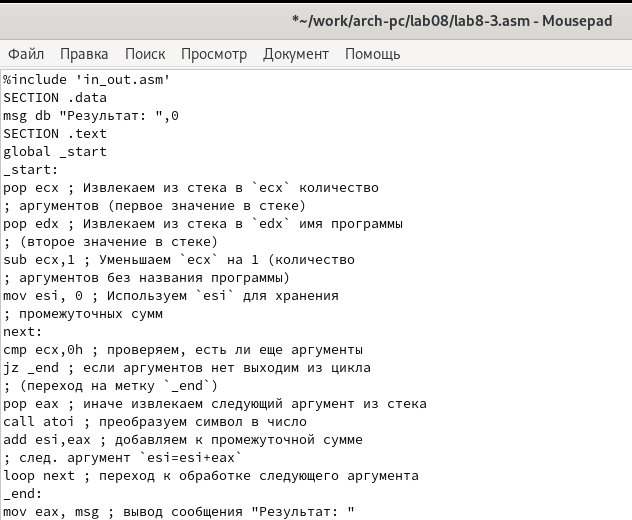


Рис. 9: Копирование программы из третьего листинга

Компилирую программу и запускаю, указав в качестве аргументов некоторые числа, программа их складывает (рис. 10).

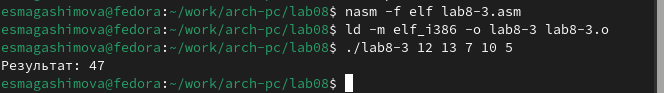


Рис. 10: Запуск третьей программы

Изменяю поведение программы так, чтобы указанные аргументы она умножала, а не складывала (рис. 11).

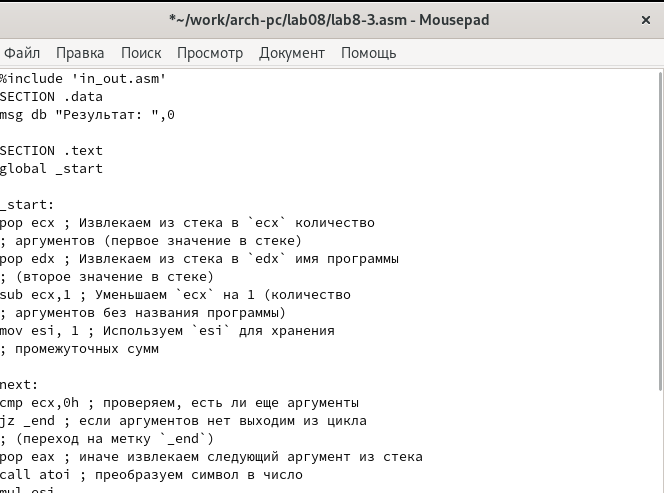


Рис. 11: Изменение третьей программы

Программа действительно теперь умножает данные на вход числа (рис. 12).

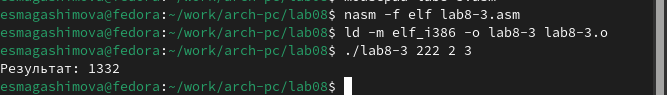


Рис. 12: Запуск измененной третьей программы

## 4.3 Задание для самостоятельной работы

Пишу программму, которая будет находить сумма значений для функции f(x) = 2x+15, которая совпадает с моим первым варинтом (рис. 13).

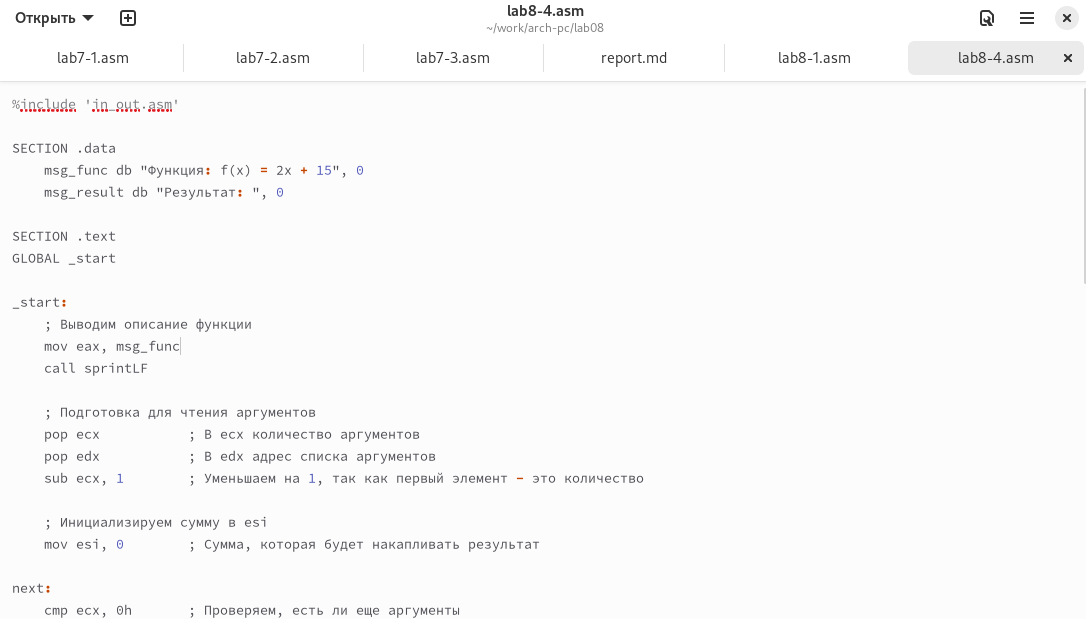


Рис. 13: Написание программы для самостоятельной работы

Код программы:

%include 'in\_out.asm'  
  
SECTION .data  
 msg\_func db "Функция: f(x) = 2x + 15", 0  
 msg\_result db "Результат: ", 0  
  
SECTION .text  
GLOBAL \_start  
  
\_start:  
 ; Выводим описание функции  
 mov eax, msg\_func  
 call sprintLF  
  
 ; Подготовка для чтения аргументов  
 pop ecx ; В ecx количество аргументов  
 pop edx ; В edx адрес списка аргументов  
 sub ecx, 1 ; Уменьшаем на 1, так как первый элемент - это количество  
  
 ; Инициализируем сумму в esi  
 mov esi, 0 ; Сумма, которая будет накапливать результат  
  
next:  
 cmp ecx, 0h ; Проверяем, есть ли еще аргументы  
 jz \_end ; Если нет, переходим к выводу результата  
  
 pop eax ; Получаем следующий аргумент (x\_i)  
 call atoi ; Преобразуем строку в число  
  
 ; Вычисляем f(x) = 2x + 15  
 mov ebx, 2 ; Загружаем 2 в ebx  
 mul ebx ; eax = eax \* ebx (2 \* x)  
 add eax, 15 ; eax = eax + 15 (2x + 15)  
  
 add esi, eax ; Добавляем результат в сумму  
  
 loop next ; Переходим к следующему аргументу  
  
\_end:  
 ; Выводим результат  
 mov eax, msg\_result  
 call sprint  
  
 mov eax, esi ; Результат суммы  
 call iprintLF ; Выводим результат  
  
 ; Завершаем программу  
 call quit

Проверяю работу программы, указав в качестве аргумента несколько чисел (рис. 14).

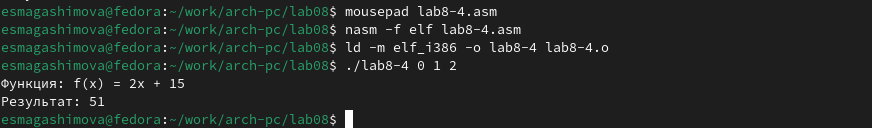


Рис. 14: Запуск программы для самостоятельной работы

# 5 Выводы

В результате выполнения данной лабораторной работы я приобрела навыки написания программ с использованием циклов а также научилась обрабатывать аргументы командной строки.

# Список литературы

1. [Курс на ТУИС](https://esystem.rudn.ru/course/view.php?id=112)
2. [Лабораторная работа №8](https://esystem.rudn.ru/pluginfile.php/2089095/mod_resource/content/0/%D0%9B%D0%B0%D0%B1%D0%BE%D1%80%D0%B0%D1%82%D0%BE%D1%80%D0%BD%D0%B0%D1%8F%20%D1%80%D0%B0%D0%B1%D0%BE%D1%82%D0%B0%20%E2%84%968.%20%D0%9F%D1%80%D0%BE%D0%B3%D1%80%D0%B0%D0%BC%D0%BC%D0%B8%D1%80%D0%BE%D0%B2%D0%B0%D0%BD%D0%B8%D0%B5%20%D1%86%D0%B8%D0%BA%D0%BB%D0%B0.%20%D0%9E%D0%B1%D1%80%D0%B0%D0%B1%D0%BE%D1%82%D0%BA%D0%B0%20%D0%B0%D1%80%D0%B3%D1%83%D0%BC%D0%B5%D0%BD%D1%82%D0%BE%D0%B2%20%D0%BA%D0%BE%D0%BC%D0%B0%D0%BD%D0%B4%D0%BD%D0%BE%D0%B9%20%D1%81%D1%82%D1%80%D0%BE%D0%BA%D0%B8.pdf)
3. [Программирование на языке ассемблера NASM Столяров А. В.](https://esystem.rudn.ru/pluginfile.php/2088953/mod_resource/content/2/%D0%A1%D1%82%D0%BE%D0%BB%D1%8F%D1%80%D0%BE%D0%B2%20%D0%90.%20%D0%92.%20-%20%D0%9F%D1%80%D0%BE%D0%B3%D1%80%D0%B0%D0%BC%D0%BC%D0%B8%D1%80%D0%BE%D0%B2%D0%B0%D0%BD%D0%B8%D0%B5%20%D0%BD%D0%B0%20%D1%8F%D0%B7%D1%8B%D0%BA%D0%B5%20%D0%B0%D1%81%D1%81%D0%B5%D0%BC%D0%B1%D0%BB%D0%B5%D1%80%D0%B0%20NASM%20%D0%B4%D0%BB%D1%8F%20%D0%9E%D0%A1%20Unix.pdf)