Шаблон отчёта по лабораторной работе

Простейший вариант

Сокирка Анна Константиновна

Содержание

# 1 Цель работы

Целью данной лабораторной работы является приобретение практических навыков работы в Midnight Commander, освоение инструкций языка ассемблера mov и int.

# 2 Задание

1. Основы работы с mc
2. Структура программы на языке ассемблера NASM
3. Подключение внешнего файла
4. Выполнение заданий для самостоятельной работы

# 3 Теоретическое введение

Midnight Commander (или просто mc) — это программа, которая позволяет просматривать структуру каталогов и выполнять основные операции по управлению файловой системой, т.е. mc является файловым менеджером. Midnight Commander позволяет сделать работу с файлами более удобной и наглядной. Программа на языке ассемблера NASM, как правило, состоит из трёх секций: секция кода программы (SECTION .text), секция инициированных (известных во время компиляции) данных (SECTION .data) и секция неинициализированных данных (тех, под которые во время компиляции только отводится память, а значение присваивается в ходе выполнения программы) (SECTION .bss). Для объявления инициированных данных в секции .data используются директивы DB, DW, DD, DQ и DT, которые резервируют память и указывают, какие значения должны храниться в этой памяти: - DB (define byte) — определяет переменную размером в 1 байт; - DW (define word) — определяет переменную размеров в 2 байта (слово); - DD (define double word) — определяет переменную размером в 4 байта (двойное слово); - DQ (define quad word) — определяет переменную размером в 8 байт (учетве- рённое слово); - DT (define ten bytes) — определяет переменную размером в 10 байт. Директивы используются для объявления простых переменных и для объявления массивов. Для определения строк принято использовать директиву DB в связи с особенностями хранения данных в оперативной памяти. Инструкция языка ассемблера mov предназначена для дублирования данных источника в приёмнике. mov dst,src Здесь операнд dst — приёмник, а src — источник. В качестве операнда могут выступать регистры (register), ячейки памяти (memory) и непосредственные значения (const). Инструкция языка ассемблера intпредназначена для вызова прерывания с указанным номером. int n Здесь n — номер прерывания, принадлежащий диапазону 0–255. При программировании в Linux с использованием вызовов ядра sys\_calls n=80h (принято задавать в шестнадцатеричной системе счисления).

# 4 Выполнение лабораторной работы

## 4.1 Основы работы с mc

Открою Midnight Commander (рис. 1).

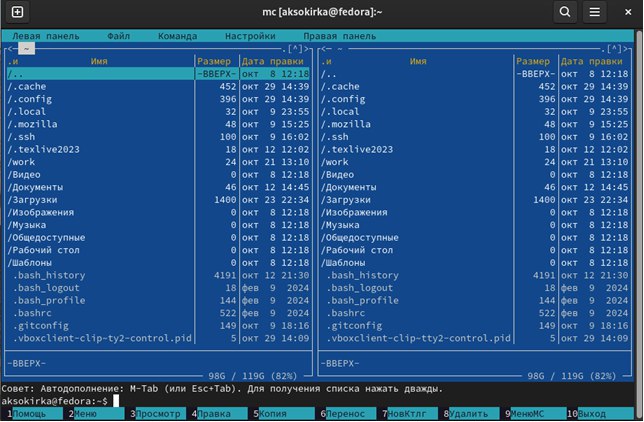


Рис. 1: Рисунок 1

Перейду в каталог ~/work/arch-pc созданный при выполнении лабораторной работы №4 (рис. 2).

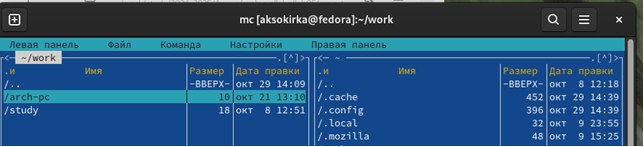


Рис. 2: Рисунок 2

С помощью функциональной клавиши F7 создам папку lab05 и перейду в созданный каталог (рис. 3).

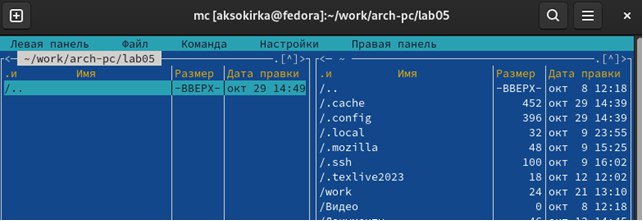


Рис. 3: Рисунок 3

Пользуясь строкой ввода и командой touch создам файл lab5-1.asm (рис. 4).

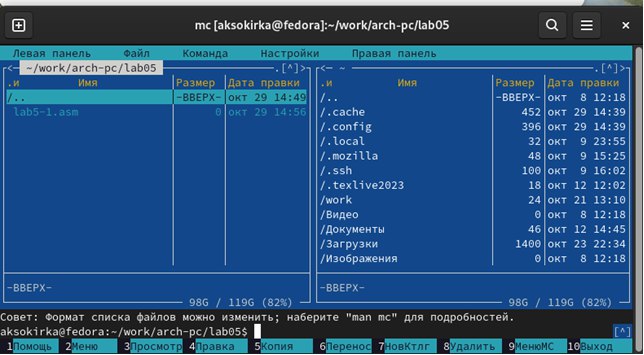


Рис. 4: Рисунок 4

## 4.2 Структура программы на языке ассемблера NASM

С помощью функциональной клавиши F4 откройте файл lab5-1.asm для редактирования во встроенном редакторе (рис. 5).

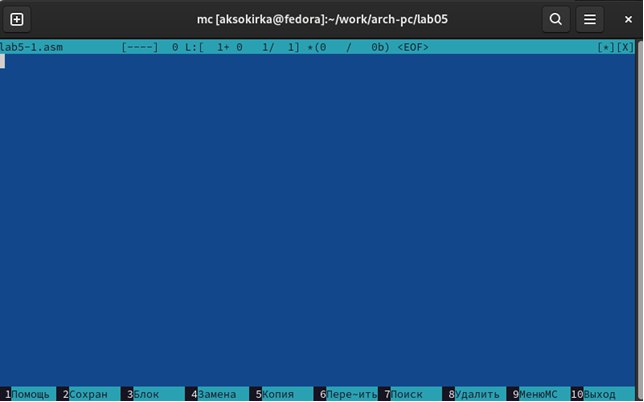


Рис. 5: Рисунок 5

Ввожу текст программы из листинга, сохраняю изменения и закрываю файл (рис. 6).

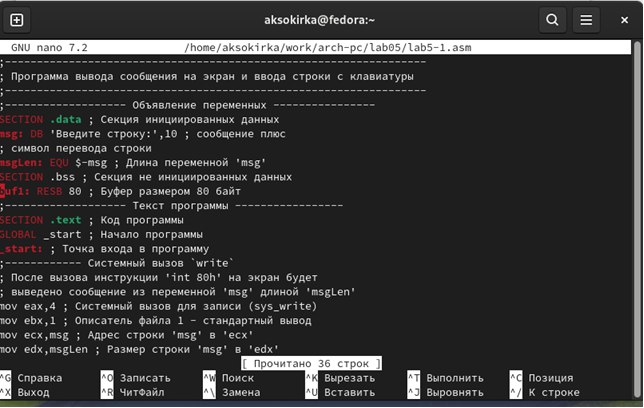


Рис. 6: Рисунок 6

С помощью функциональной клавиши F3 открою файл lab5-1.asm для просмотра.Убеждаюсь, что файл содержит текст программы (рис. 7).

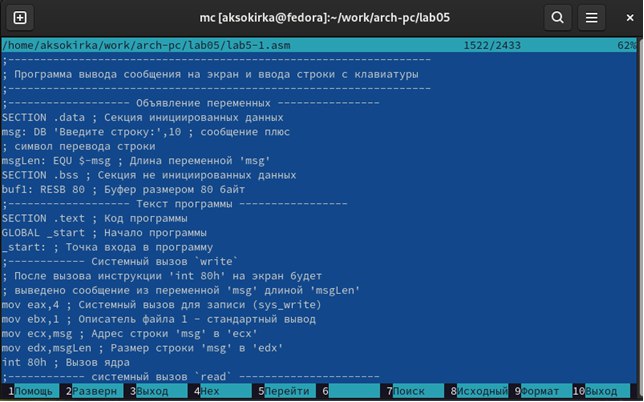


Рис. 7: Рисунок 7

Оттранслирую текст программы lab5-1.asm в объектный файл. Выполню компоновку объектного файла и запущу получившийся исполняемый файл. Программа выводит строку ‘Введите строку:’ и ожидает ввода с клавиатуры. На запрос введу мои ФИО (рис. 8).

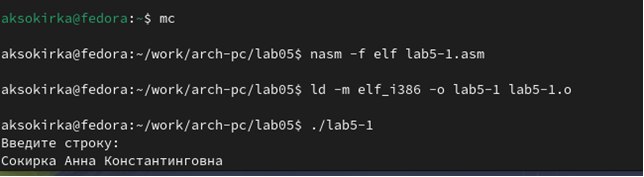


Рис. 8: Рисунок 8

## 4.3 Подключение внешнего файла

Скачиваю файл in\_out.asm со страницы курса в ТУИС (рис. 9).

Рис. 9: Рисунок 9

Рис. 9: Рисунок 9

С помощью функциональной клавиши F5 копирую файл in\_out.asm из каталога Загрузки в созданный каталог lab05-1.asm (рис. 10).

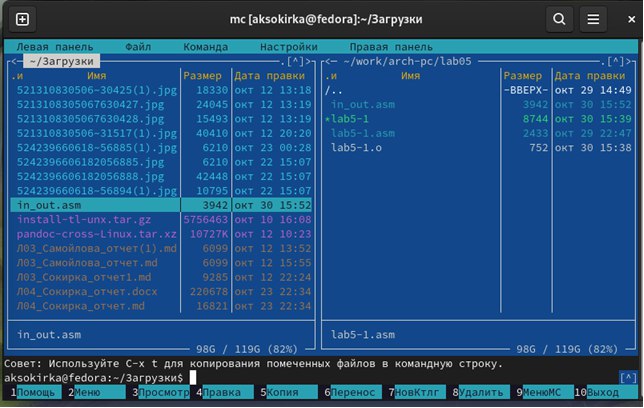


Рис. 10: Рисунок 10

Cоздам копию файла lab5-1.asm с именем lab5-2.asm. Выделю файл lab5-1.asm, введу имя файла lab5-2.asm (рис. 11).

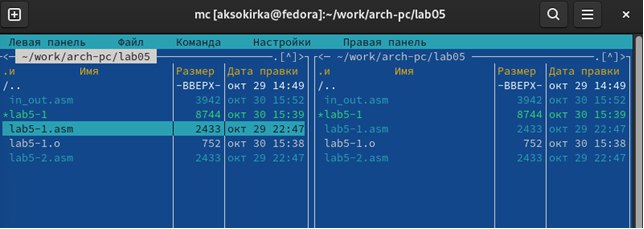


Рис. 11: Рисунок 11

Изменяю содержимое файла lab5-2.asm во встроенном редакторе nano, чтобы в программе использовались подпрограммы из внешнего файла in\_out.asm (рис. 12).

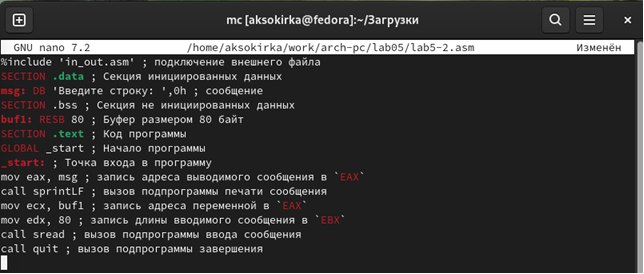


Рис. 12: Рисунок 12

Транслирую текст программы файла в объектный файл командой nasm -f elf lab5-2.asm. Создался объектный файл lab5-2.o. Выполняю компоновку объектного файла с помощью команды ld -m elf\_i386 -o lab5-2 lab5-2.o Создался исполняемый файл lab5-2. Запускаю исполняемый файл (рис. 13).

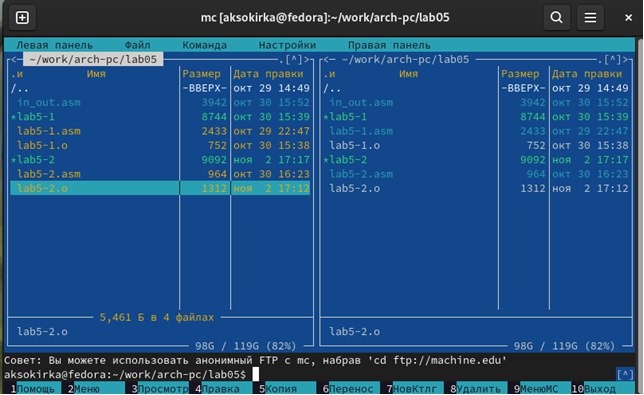


Рис. 13: Рисунок 13

Разница между первым исполняемым файлом lab5-2 и вторым lab5-2-2 в том, что запуск первого запрашивает ввод с новой строки, а программа, которая исполняется при запуске второго, запрашивает ввод без переноса на новую строку, потому что в этом заключается различие между подпрограммами sprintLF и sprint (рис. 14).

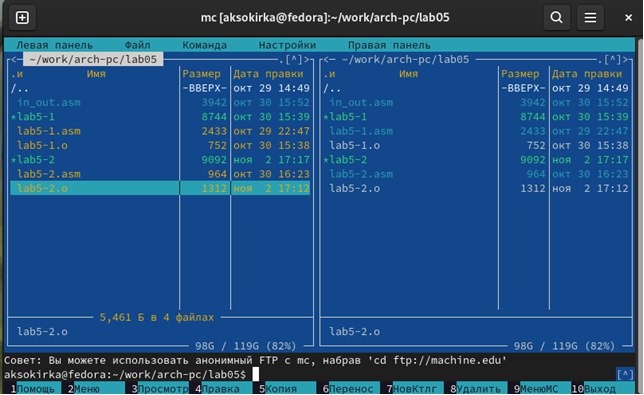


Рис. 14: Рисунок 14

Открываю файл lab5-2.asm для редактирования в nano функциональной клавишей F4. Изменяю в нем подпрограмму sprintLF на sprint. Сохраняю изменения и открываю файл для просмотра, чтобы проверить сохранение действий (рис. 15).

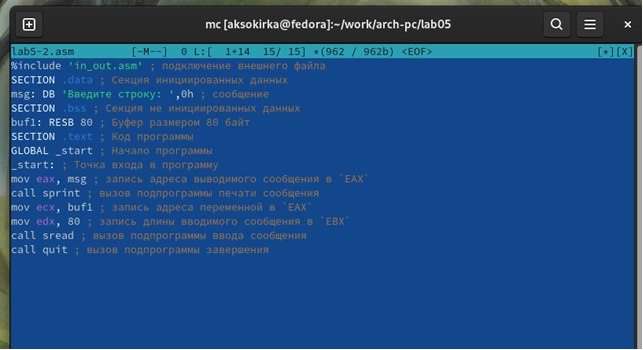


Рис. 15: Рисунок 15

Снова транслирую файл, выполняю компоновку созданного объектного файла, запускаю новый исполняемый файл (рис. 16).

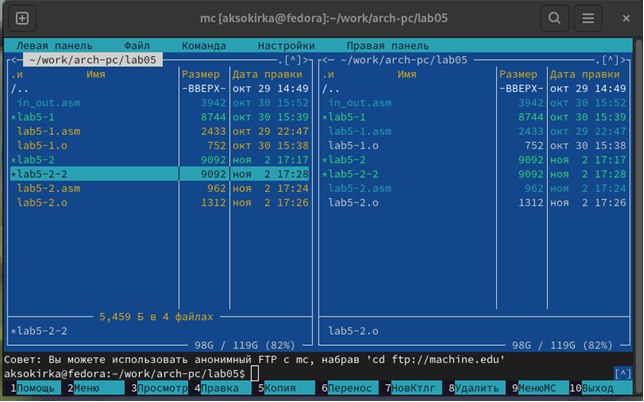


Рис. 16: Рисунок 16

Разница между первым исполняемым файлом lab5-2 и вторым lab5-2-2 в том, что запуск первого запрашивает ввод с новой строки, а программа, которая исполняется при запуске второго, запрашивает ввод без переноса на новую строку, потому что в этом заключается различие между подпрограммами sprintLF и sprint.

## 4.4 Выполнение заданий для самостоятельной работы

Создаю копию файла lab5-1.asm с именем lab5-1-1.asm с помощью функциональной клавиши F5 (рис. 17).

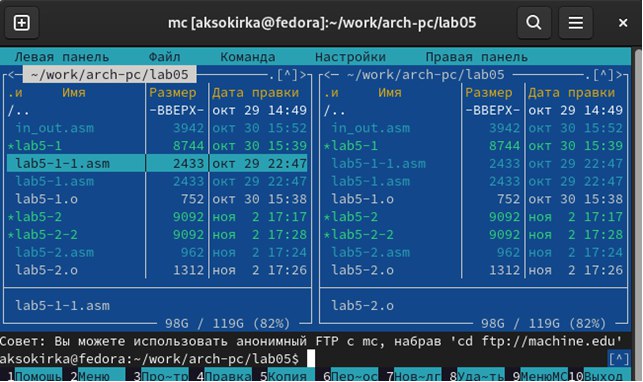


Рис. 17: Рисунок 17

С помощью функциональной клавиши F4 открываю созданный файл для редактирования. Изменяю программу так, чтобы кроме вывода приглашения и запроса ввода, она выводила вводимую пользователем строку (рис. 18).

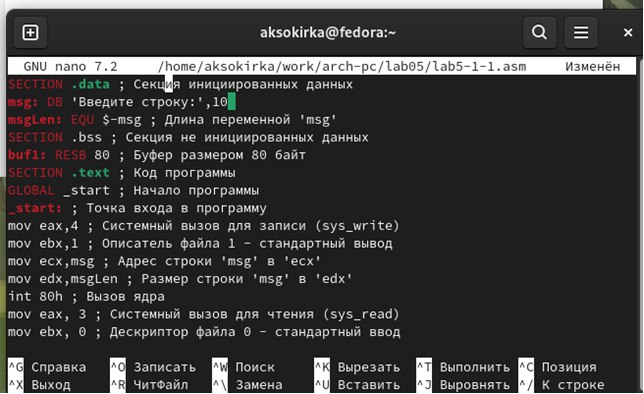


Рис. 18: Рисунок 18

Создаю объектный файл lab5-1-1.o, отдаю его на обработку компоновщику, получаю исполняемый файл lab5-1-1, запускаю полученный исполняемый файл. Программа запрашивает ввод, ввожу свои ФИО, далее программа выводит введенные мною данные (рис. 19).

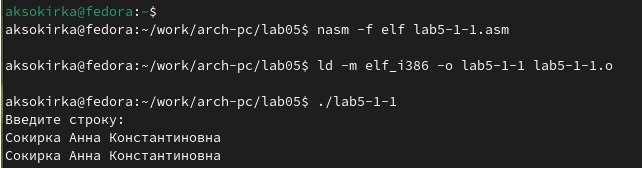


Рис. 19: Рисунок 19

Создаю копию файла lab5-2.asm с именем lab5-2-1.asm с помощью функциональной клавиши F5 (рис. 20).

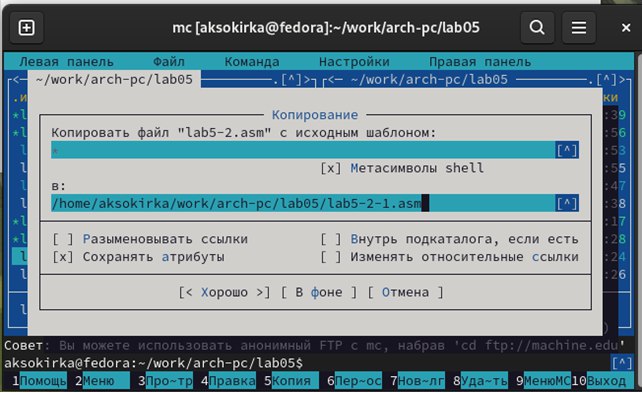


Рис. 20: Рисунок 20

С помощью функциональной клавиши F4 открываю созданный файл для редактирования. Изменяю программу так, чтобы кроме вывода приглашения и запроса ввода, она выводила вводимую пользователем строк (рис. 21).

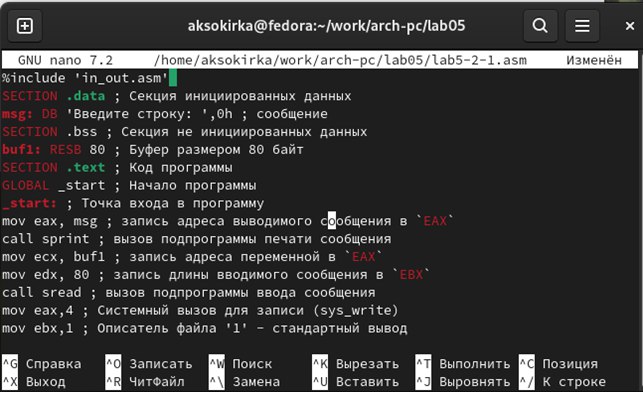


Рис. 21: Рисунок 21

Создаю объектный файл lab5-2-1.o, отдаю его на обработку компоновщику, получаю исполняемый файл lab5-2-1, запускаю полученный исполняемый файл. Программа запрашивает ввод без переноса на новую строку, ввожу свои ФИО, далее программа выводит введенные мною данные (рис. 22).

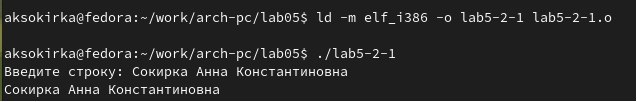


Рис. 22: Рисунок 22

# 5 Выводы

При выполнении данной лабораторной работы я приобрела практические навыки работы в Midnight Commander, а также освоила инструкции языка ассемблера mov и int.

# 6 Список литературы

https://esystem.rudn.ru/pluginfile.php/2089085/mod\_resource/content/0/Лабораторная%20работа%20№5.%20Основы%20работы%20с%20Midnight%20Commander%20%28%29.%20Структура%20программы%20на%20языке%20ассемблера%20NASM.%20Системные%20вызовы%20в%20ОС%20GNU%20Linux.pdf