Лабораторная работа №5

Дисциплина: Архитектура компьютера

Серебрякова Дарья Ильинична

Содержание

# 1 Цель работы

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

# 2 Задания

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

Открываю Midnight Commander командой mc (рис. 1).

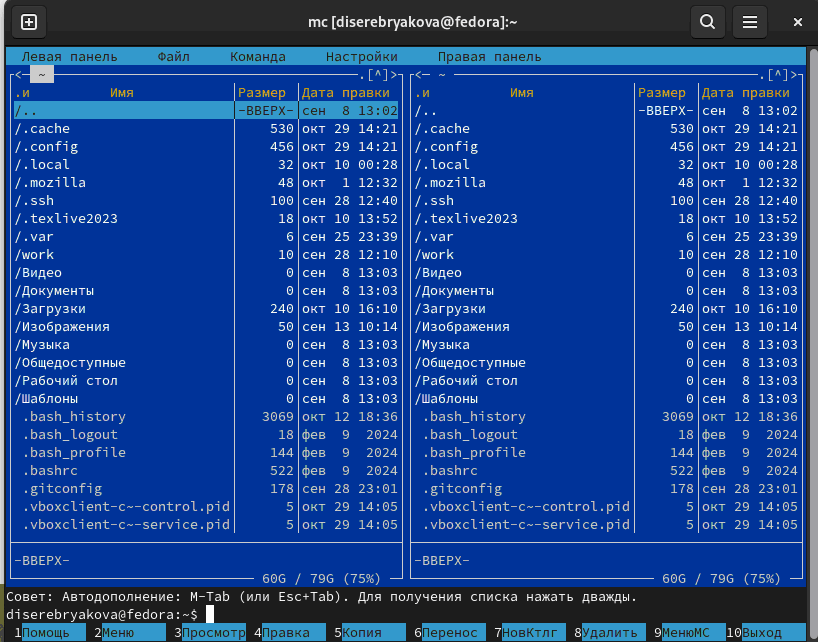


Рис. 1: Открыт Midnight Commander

Перехожу в каталог ~/work/study/study\_2024-2025\_arch-pc, используя файловый менеджер mc и клавиши (рис. 2).

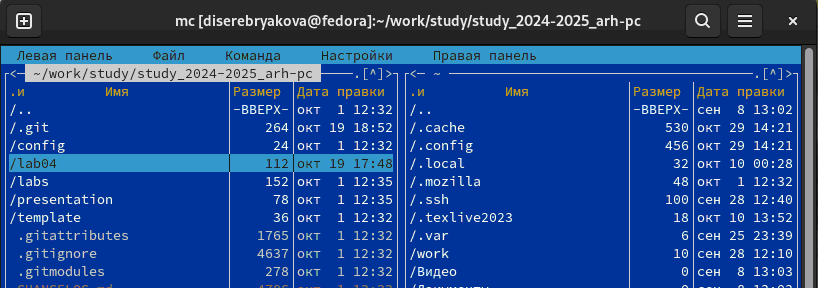


Рис. 2: Перемещение по каталогам

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

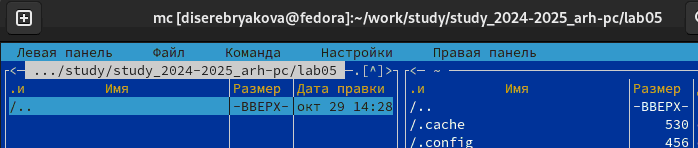


Рис. 3: Создание папки

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

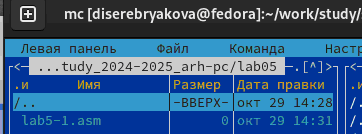


Рис. 4: Создание файла

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

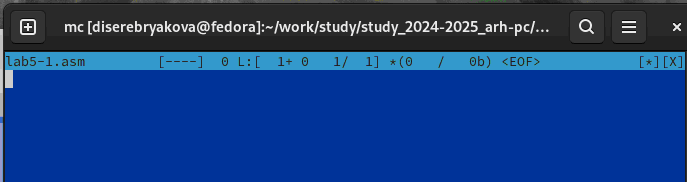


Рис. 5: Открытие файла для редактирования в mcedit

Ввожу текст программы из предложенного листинга 5.1, сохраняю изменения и закрываю файл (рис. 6).

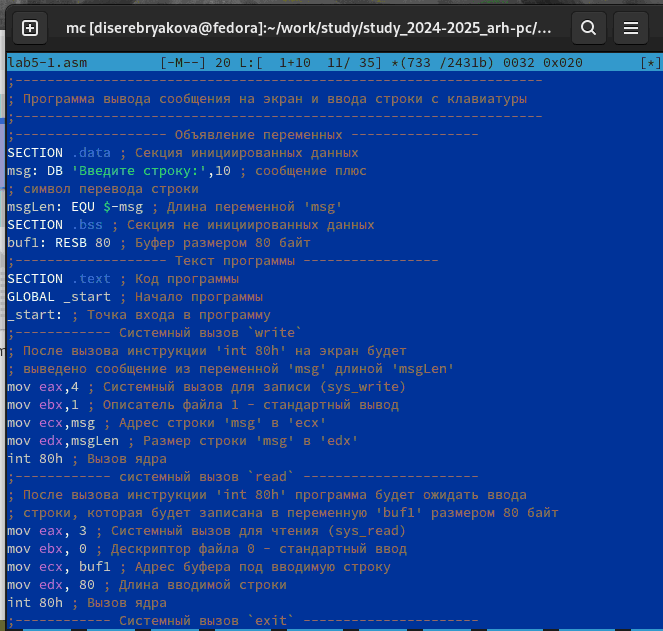


Рис. 6: Изменение текста файла

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

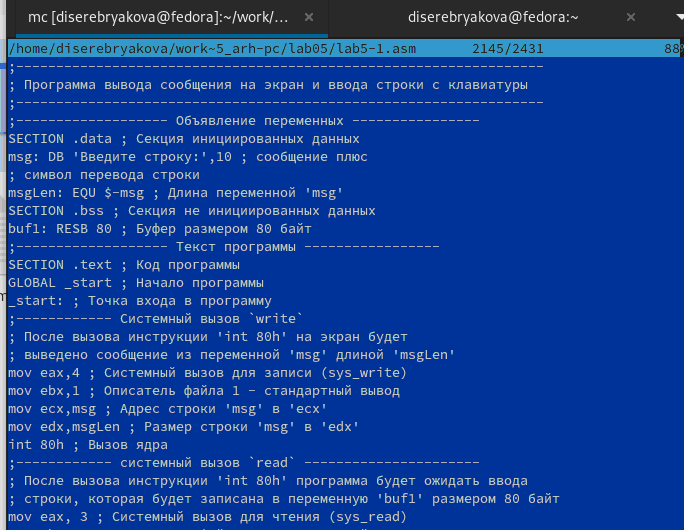


Рис. 7: Проверка сохраненных изменений

Транслирую текст программы файла в объектный файл командой nasm -f elf lab5-1.asm. Создался объектный файл lab5-1.o. (рис. 8).

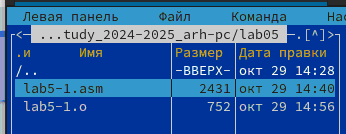


Рис. 8: Трансляция текста объектный файл

Выполняю компоновку объектного файла с помощью команды ld -m elf\_i386 -o lab5-1 lab5-1.o. Создался исполняемый файл lab5-1.(рис. 9).

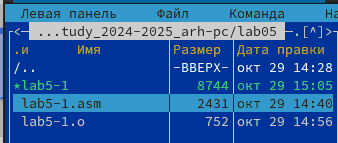


Рис. 9: Компоновка объектного файла

Запускаю исполняемый файл командой ./lab5-1. Программа выводит строку “Введите строку:” и ждет ввода с клавиатуры, я ввожу свои ФИО, на этом программа заканчивает свою работу (рис. 10).

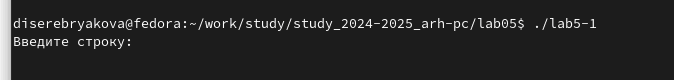


Рис. 10: Запуск исполняемого файла

Введенные выше команды (рис. 11).

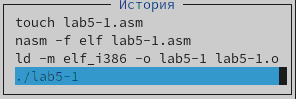


Рис. 11: История команд

## 4.2 Подключение внешнего файла in\_out.asm

Скачиваю файл in\_out.asm со страницы курса в ТУИС. Он сохранился в каталог “Загрузки” (рис. 12).

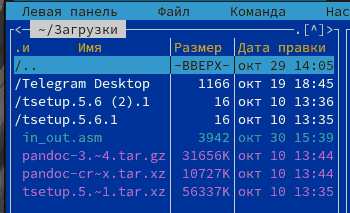


Рис. 12: Скачивание файла in\_out.asm

В одной из панелей mc открываю каталог с файлом lab5-1.asm. В другой панели каталог со скаченным файлом in\_out.asm. Копирую файл in\_out.asm в каталог с файлом lab5-1.asm с помощью функциональной клавиши F5 (рис. 13).

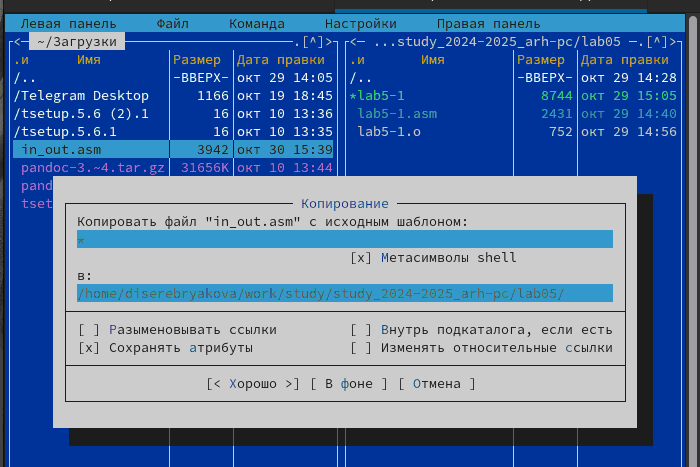


Рис. 13: Создание копии файла с помощью функциональных клавиш

С помощью функциональной клавиши F5 создаю копию файла lab5-1.asm с именем lab5-2.asm (рис. 14).

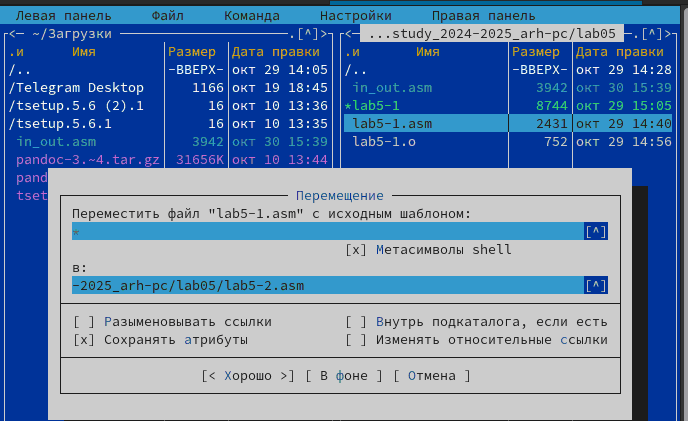


Рис. 14: Создание копии файла

Исправляю текст программы в файле lab5-2.asm с использование подпрограмм из внешнего файла in\_out.asm (использую подпрограммы sprintLF, sread и quit) (рис. 15).

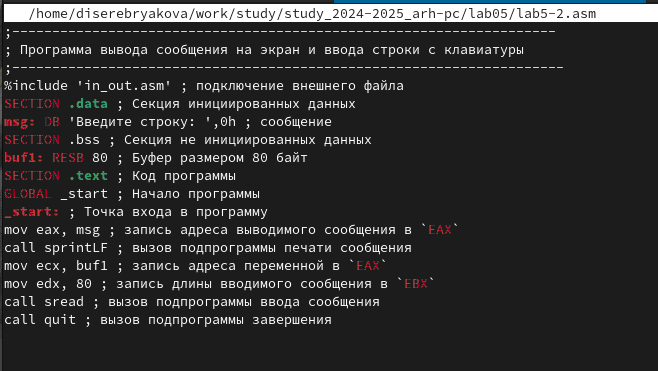


Рис. 15: Использование подпрограмм из внешнего файла

Создаю объектный файл lab5-2.o, отдаю его на обработку компоновщику, получаю исполняемый файл lab5-2, запускаю полученный исполняемый файл. Программа запрашивает ввод, ввожу свои ФИО, далее программа выводит введенные мною данные (image/рис. 16).

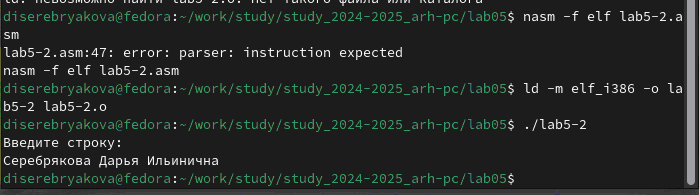


Рис. 16: Запуск испольняемого файла

В файле lab5-2.asm заменила подпрограмму sprintLF на sprint. Создала исполняемый файл и запустила его (рис. 17).

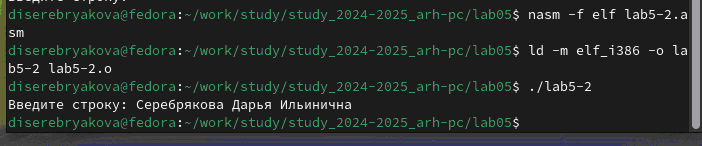


Рис. 17: Смена подпрограмм

Разница между исполняемым файлом lab5-2 и только что измененным этим файлом 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

# 5 Выполнение заданий для самостоятельной работы

Создаю копию файла lab5-1.asm с именем lab5-1-1.asm с помощью функциональной клавиши F5 (рис. 18).

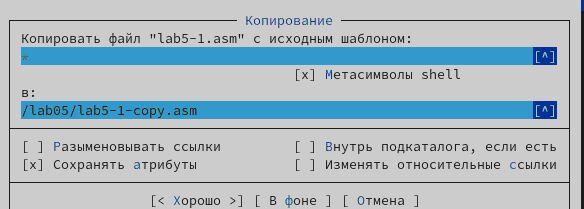


Рис. 18: Создание копии файла функциональной клавишей

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. 19).

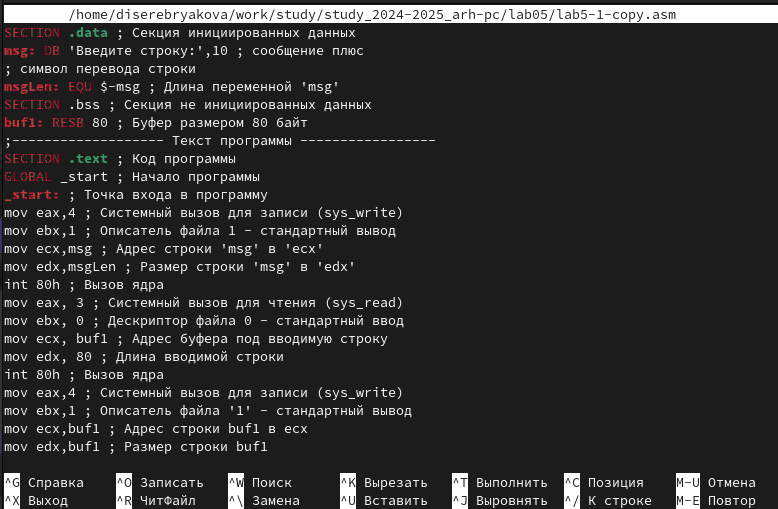


Рис. 19: Редактирование программы

Теперь проверю правильность работы отредактированного файла. Создаю объектный файл lab5-1-copy.o, отдаю его на обработку компоновщику, получаю исполняемый файл lab5-1-copy, запускаю полученный исполняемый файл. Программа запрашивает ввод, ввожу свою фамилию, далее программа выводит введенные мною данные (рис. 20).

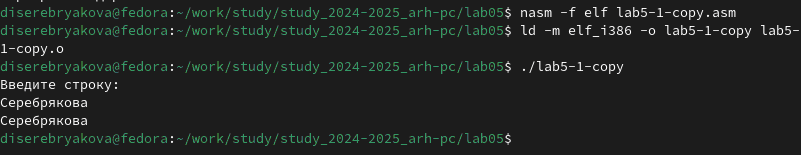


Рис. 20: Запуск исполняемого файла, проверка правильности выполнения задания

Создаю копию файла lab5-2.asm с именем lab5-2-copy.asm с помощью функциональной клавиши F5 (рис. 21).

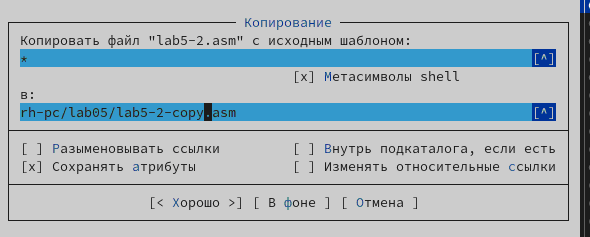


Рис. 21: Создание копии файла

Исправляю текст программы с использование подпрограмм из внешнего файла in\_out.asm, так чтобы она работала по следующему алгоритму: • вывести приглашение типа “Введите строку:” • ввести строку с клавиатуры • вывести введённую строку на экран (рис. 22).

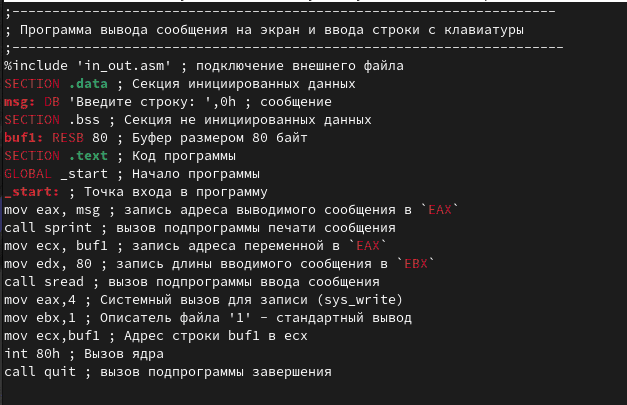


Рис. 22: Редактирование текста программы

Создаю объектный файл lab5-2-copy.o, отдаю его на обработку компоновщику, получаю исполняемый файл lab5-2-copy, запускаю полученный исполняемый файл. Программа запрашивает ввод без переноса на новую строку, ввожу свою Фамилию, далее программа выводит введенные мною данные (рис. 23).

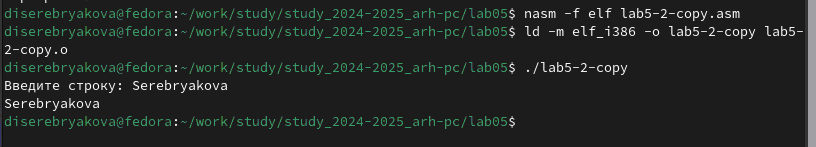


Рис. 23: Запуск исполняемого файла

# 6 Выводы

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

# Список литературы

Лабораторная работа №5