Лабораторная работа №5

Дисциплина: Архитектура компьютера

Серебрякова Дарья Ильинична

Содержание

Список иллюстраций

Список таблиц

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. 4.1).

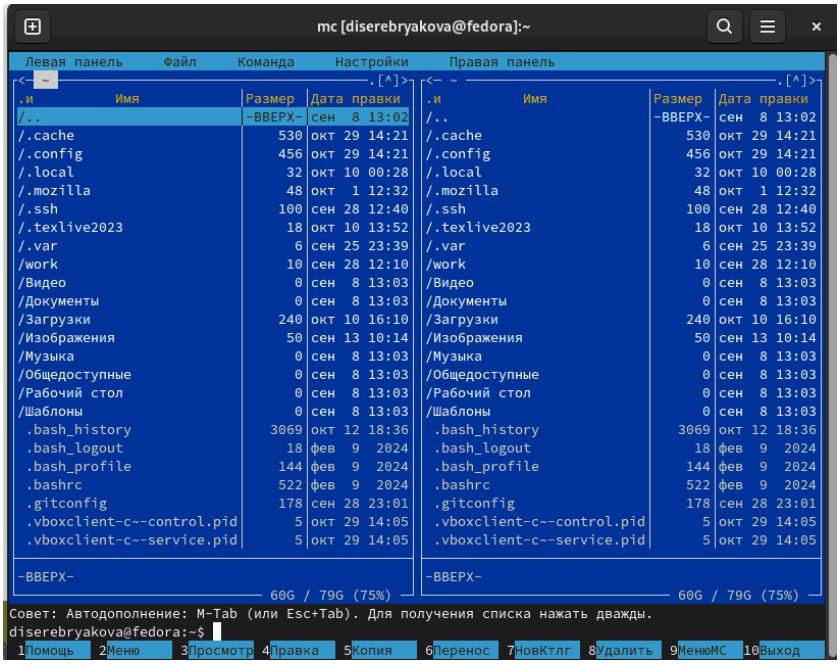

Рис. 4.1: Открыт Midnight Commander

Перехожу в каталог ~/work/study/study_2024-2025_arch-pc, используя файловый менеджер mc и клавиши (рис. 4.2).

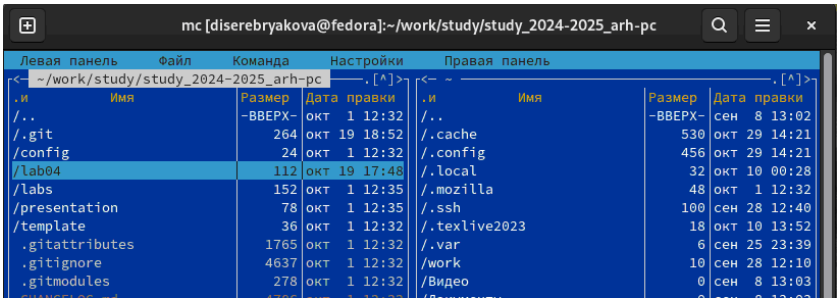

Рис. 4.2: Перемещение по каталогам

С помощью функциональной клавиши F7 создаю папку lab05 и перехожу в созданный каталог (рис. 4.3).

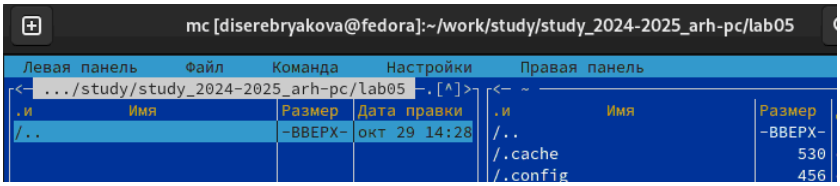

Рис. 4.3: Создание папки

Пользуясь строкой ввода и командой touch создаю файл lab5-1.asm (рис. 4.4).

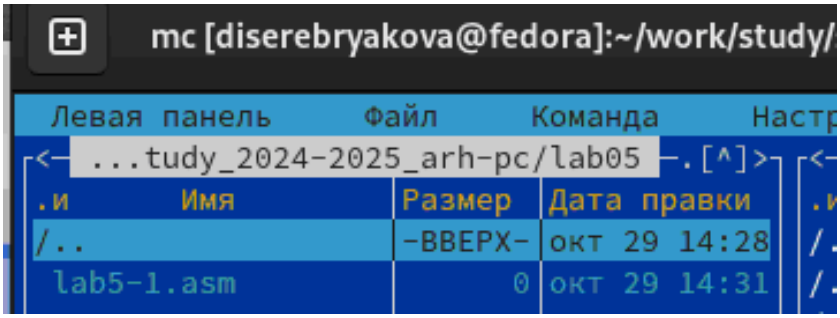

Рис. 4.4: Создание файла

С помощью функциональной клавиши F4 открываю файл lab5-1.asm для редактирования во встроенном редакторе mcedit (рис. 4.5).

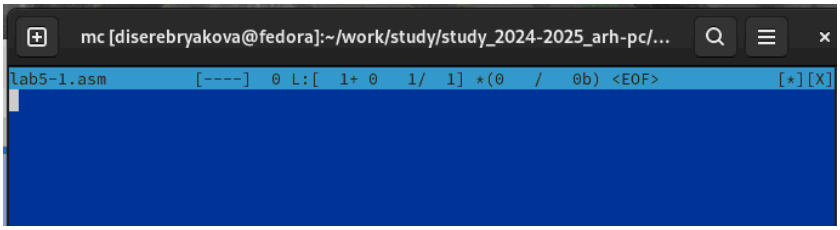

Рис. 4.5: Открытие файла для редактирования в mcedit

Ввожу текст программы из предложенного листинга 5.1, сохраняю изменения и закрываю файл (рис. 4.6).

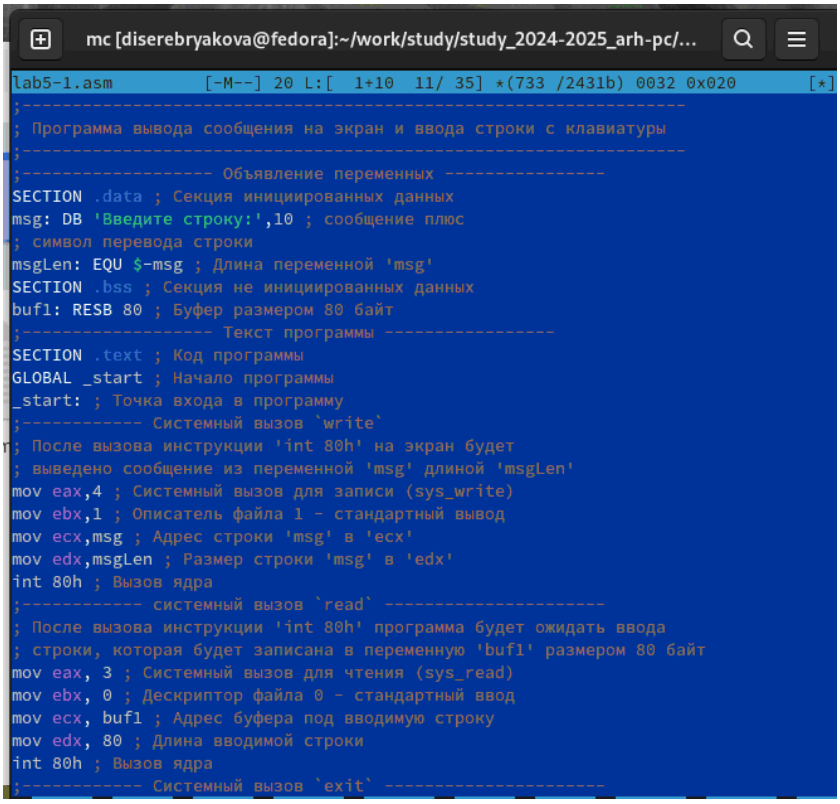

Рис. 4.6: Изменение текста файла

С помощью функциональной клавиши F3 открываю файл lab5-1.asm для просмотра. Убеждаюсь, что файл содержит текст программы (рис. 4.7).

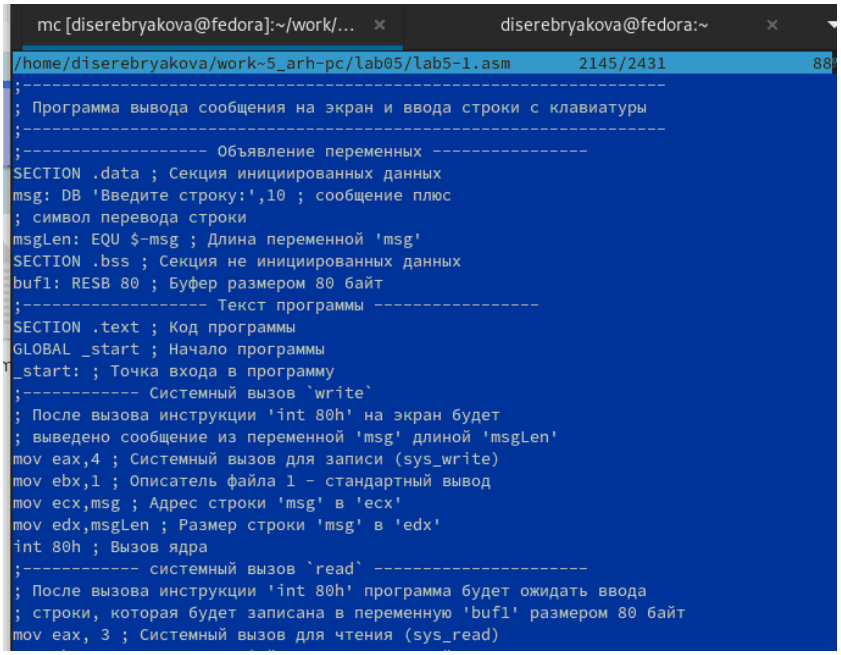

Рис. 4.7: Проверка сохраненных изменений

Транслирую текст программы файла в объектный файл командой nasm -f elf lab5-1.asm. Создался объектный файл lab5-1.o. (рис. 4.8).

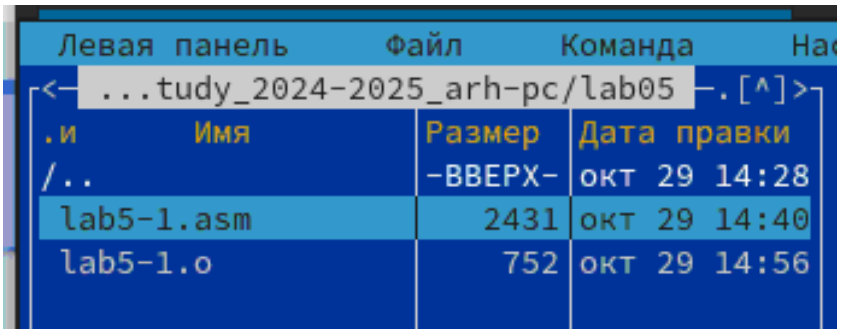

Рис. 4.8: Трансляция текста объектный файл

Выполняю компоновку объектного файла с помощью команды ld -m elf_i386 -o lab5-1 lab5-1.o. Создался исполняемый файл lab5-1.(рис. 4.9).

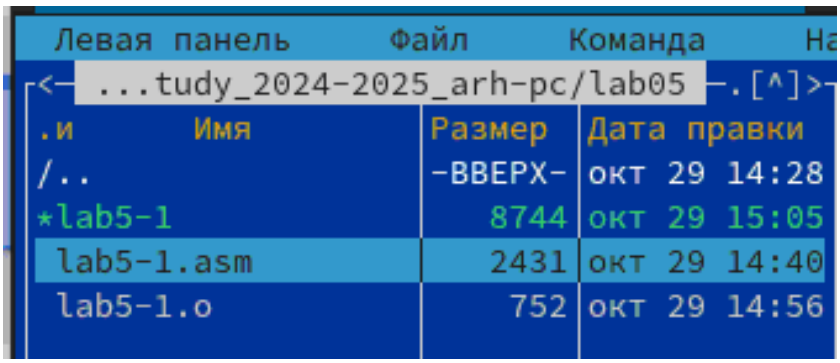

Рис. 4.9: Компоновка объектного файла

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. 4.10).

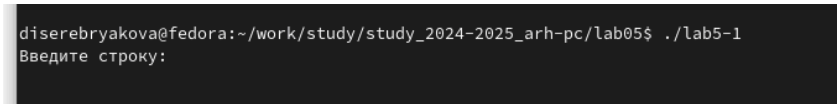

Рис. 4.10: Запуск исполняемого файла

Введенные выше команды (рис. 4.11).

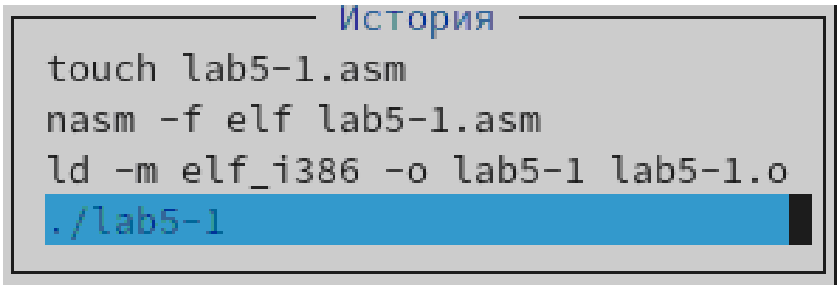

Рис. 4.11: История команд

4.2 Подключение внешнего файла in_out.asm

Скачиваю файл in_out.asm со страницы курса в ТУИС. Он сохранился в каталог “Загрузки” (рис. 4.12).

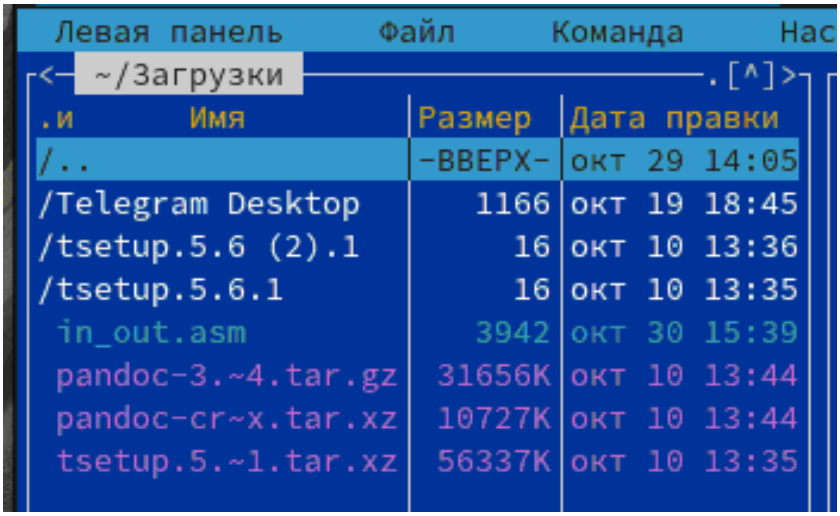

Рис. 4.12: Скачивание файла in_out.asm

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. 4.13).

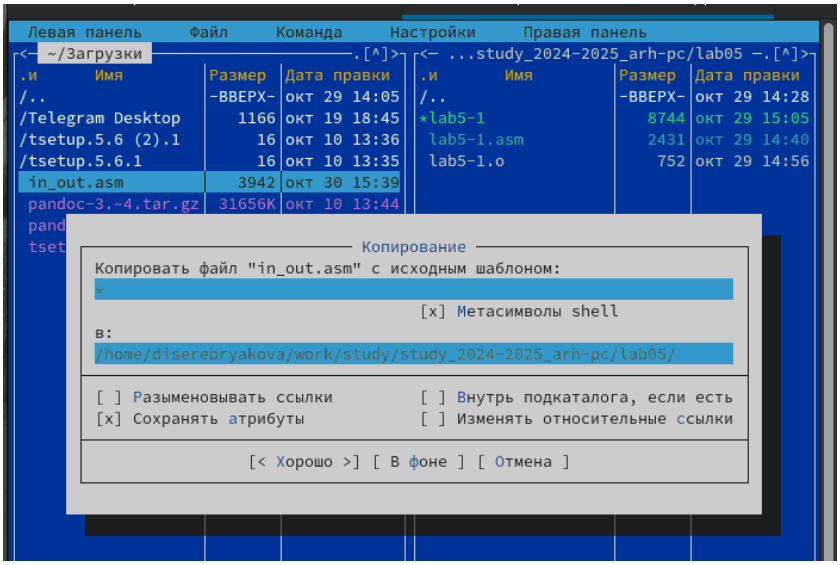

Рис. 4.13: Создание копии файла с помощью функциональных клавиш

С помощью функциональной клавиши F5 создаю копию файла lab5-1.asm с именем lab5-2.asm (рис. 4.14).

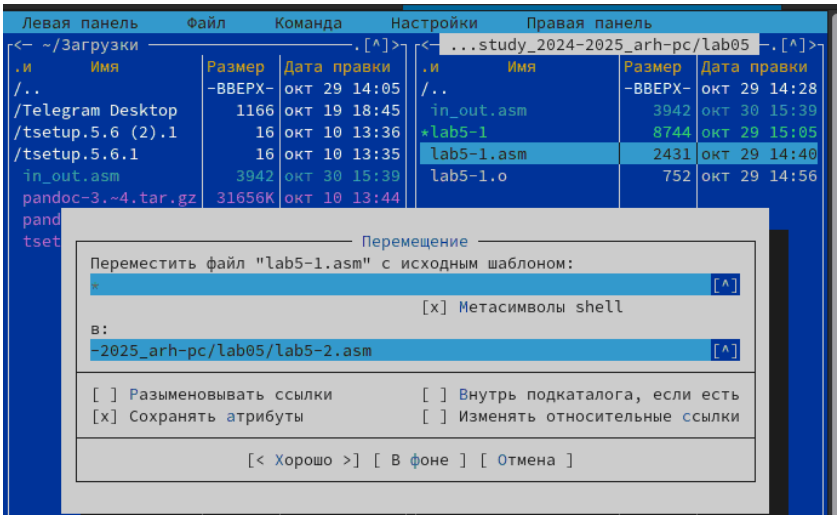

Рис. 4.14: Создание копии файла

Исправляю текст программы в файле lab5-2.asm с использование подпрограмм из внешнего файла in_out.asm (использую подпрограммы sprintLF, sread и quit) (рис. 4.15).

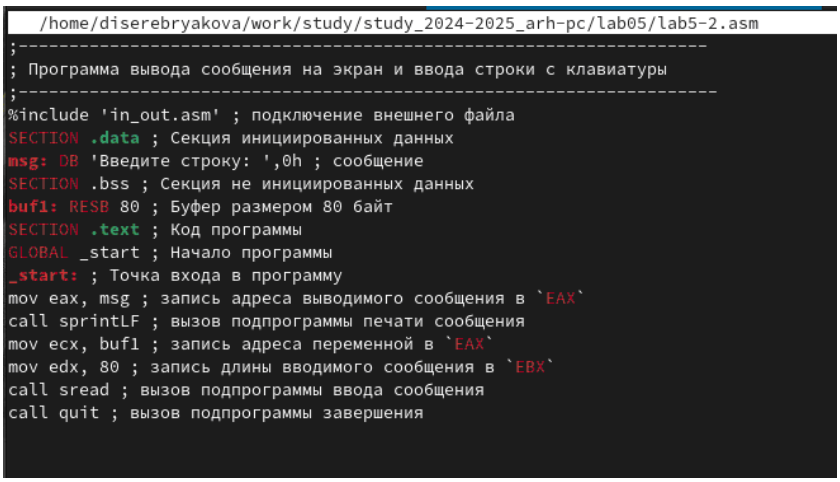

Рис. 4.15: Использование подпрограмм из внешнего файла

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. 4.16).

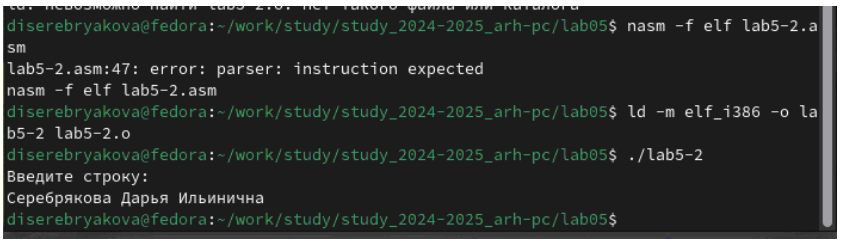

Рис. 4.16: Запуск испольняемого файла

В файле lab5-2.asm заменила подпрограмму sprintLF на sprint. Создала исполняемый файл и запустила его (рис. 4.17).

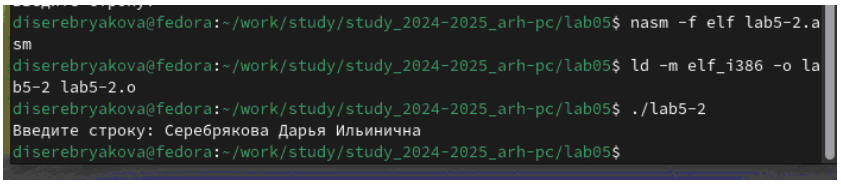

Рис. 4.17: Смена подпрограмм

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. 5.1).

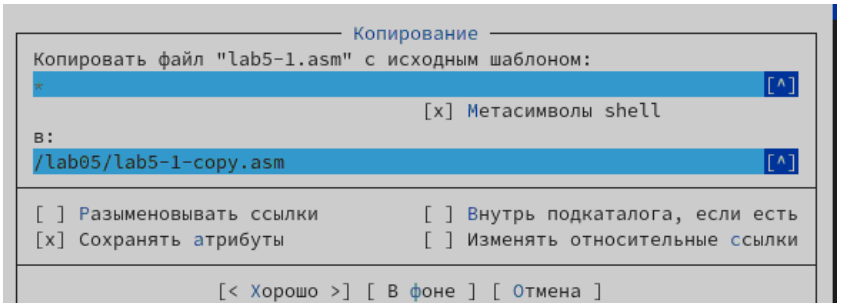

Рис. 5.1: Создание копии файла функциональной клавишей

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. 5.2).

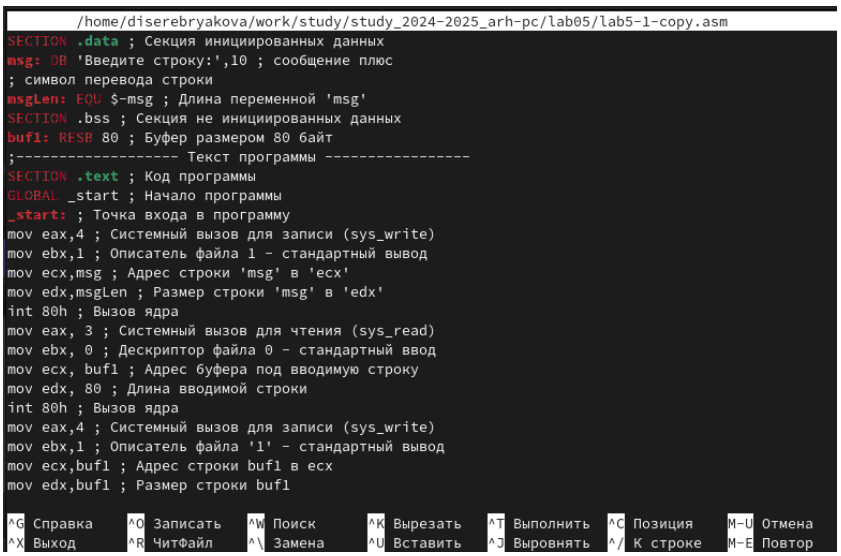

Рис. 5.2: Редактирование программы

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. 5.3).

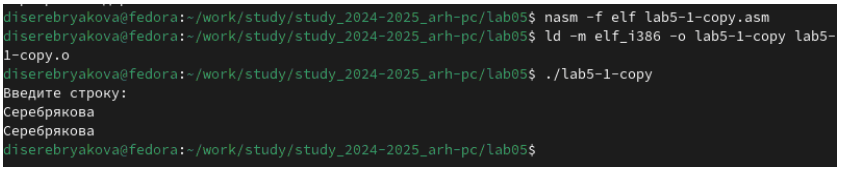

Рис. 5.3: Запуск исполняемого файла, проверка правильности выполнения задания

Создаю копию файла lab5-2.asm с именем lab5-2-copy.asm с помощью функциональной клавиши F5 (рис. 5.4).

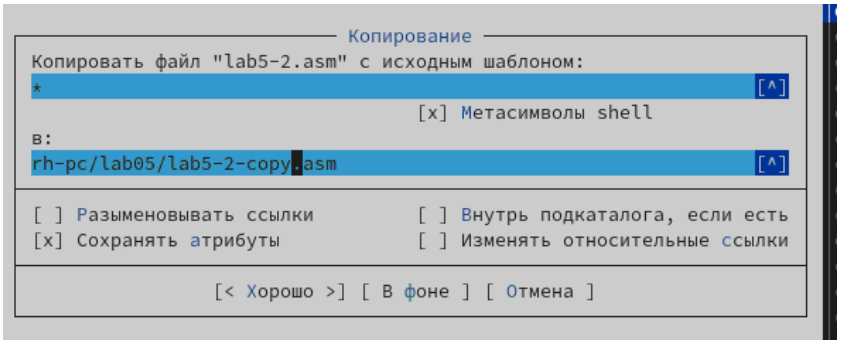

Рис. 5.4: Создание копии файла

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. 5.5).

```
;-------------------------------------------------------------------
; Программа вывода сообщения на экран и ввода строки с клавиатуры
;-------------------------------------------------------------------
%include 'in_out.asm' ; подключение внешнего файла
SECTION .data ; Секция инициированных данных
msg: DB 'Введите строку: ',0h ; сообщение
SECTION .bss ; Секция не инициированных данных
buf1: RESB 80 ; Буфер размером 80 байт
SECTION .text ; Код программы
GLOBAL _start ; Начало программы
_start: ; Точка входа в программу
mov eax, msg ; запись адреса выводимого сообщения в `EAX`
call sprint ; вызов подпрограммы печати сообщения
mov ecx, buf1 ; запись адреса переменной в `EAX`
mov edx, 80 ; запись длины вводимого сообщения в `EBX`
call sread ; вызов подпрограммы ввода сообщения
mov eax,4 ; Системный вызов для записи (sys_write)
mov ebx,1 ; Описатель файла '1' - стандартный вывод
mov ecx,buf1 ; Адрес строки buf1 в ecx
int 80h ; Вызов ядра
call quit ; вызов подпрограммы завершения
```

Рис. 5.5: 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. 5.6).

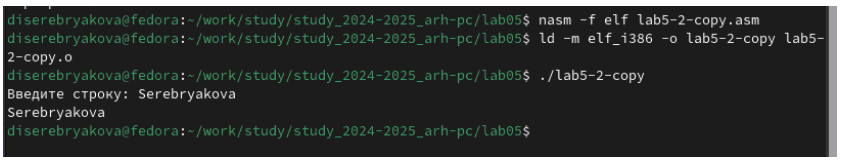

Рис. 5.6: Запуск исполняемого файла

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