Отчёт по лабораторной работе №5

Дисциплина: Архитектура компьтеров

Карпова Анастасия Александровна

Содержание

[1 Цель работы 1](#_Toc150619253)

[2 Задание 1](#_Toc150619254)

[3 Теоретическое введение 1](#_Toc150619255)

[4 Выполнение лабораторной работы 2](#_Toc150619256)

[5 Выводы 8](#_Toc150619257)

[Список литературы 8](#_Toc150619258)

# 1 Цель работы

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

Основы работы с mc

Открываю Midhight Commander, введя mc в терминал (рис. 4.1).

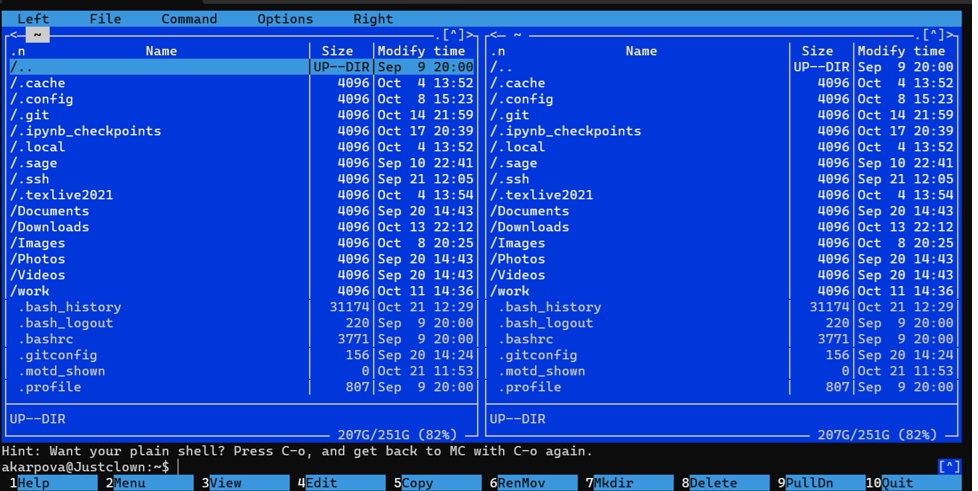


Рис. 4.1. Открытие mc

Перехожу в каталог ~/work/arch-pc и создаю в нем каталог lab05, перехожу в этот каталог(рис. 4.2).

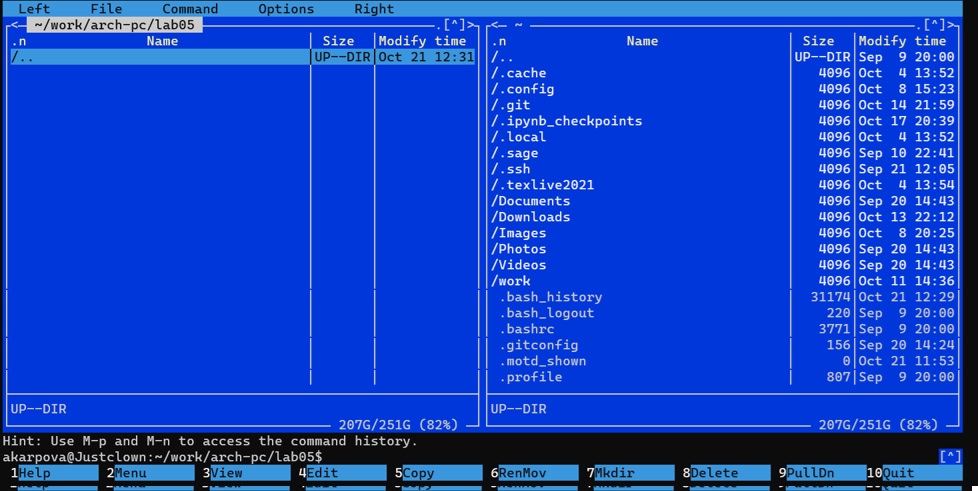


Рис. 4.2. Каталог lab05

Создаю файл lab5-1.asm пр (и помощи команды touch (рис. 4.3)

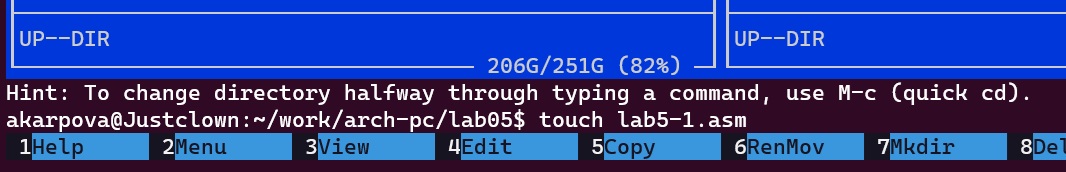


Рис. 4.3. Создание файла

Структура программы на языке ассемблера NASM

При помощи клавиши f4 открываю файл lab5-1.asm для редактирования в nano (рис. 4.4)



Рис. 4.4. Открытие файла

Ввожу в файл код программы для запроса строки у пользователя. Потом выхожу из файла при помощи Ctrl+X, сохранив изменения - Y, Enter. (рис. 4.5)

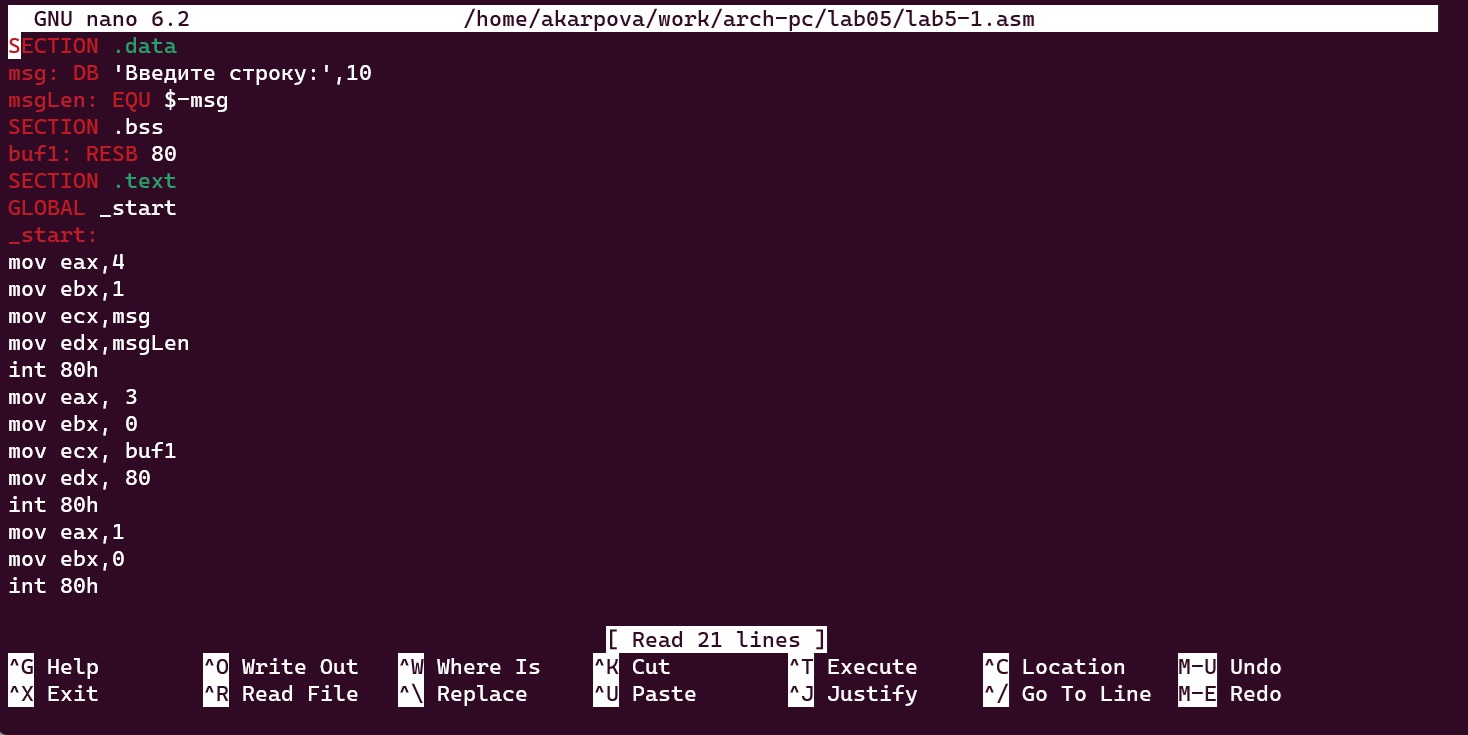


Рис. 4.5. Ввод кода

Проверяю файл на содержание кода при помощи f3(для просмотра файла) (рис. 4.6)

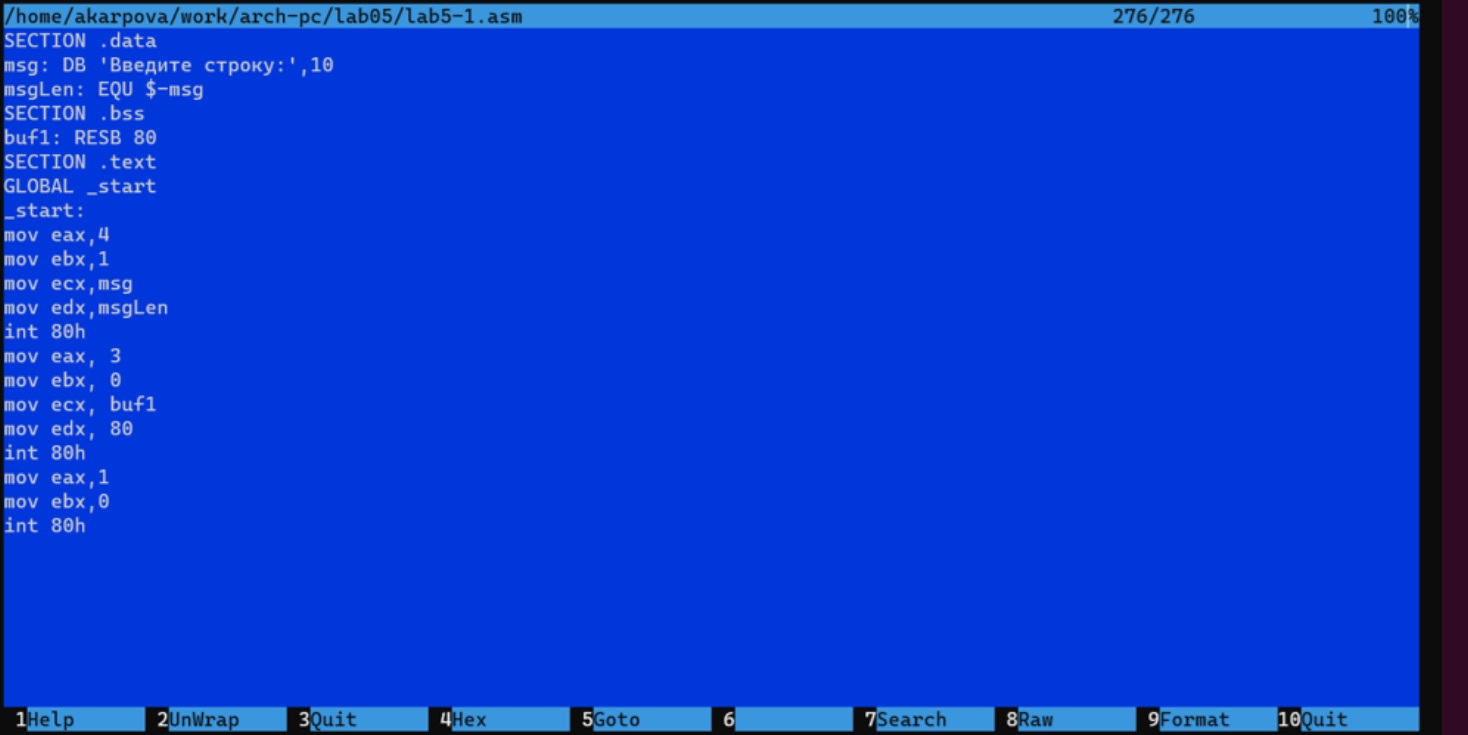


Рис. 4.6. Проверка

Транслирую текст программы в объектный файл командой nasm -f elf lab5-1.asm (создался объектный файл lab5-1.o). Затем выполняю компоновку объектного файла при помощи команды ld -m elf\_i386 -o lab5-1 lab5-1.o. (рис. 4.7)

Транслятор nasm и компоновка объектного файла

Рис. 4.7. Транслятор nasm и компоновка объектного файла

Запускаю исполняемый файл. (Программа выводит строку “Введите строку” - соответственно мы вводим своё ФИО и программа заканчивает свою работу, выведя наше ФИО (рис. 4.8)

Запуск исполняемого файла

Рис. 4.8. Запуск исполняемого файла

Запуск исполняемого файла

Скачиваю файл in\_out.asm из ТУИСа. (рис. 4.9)

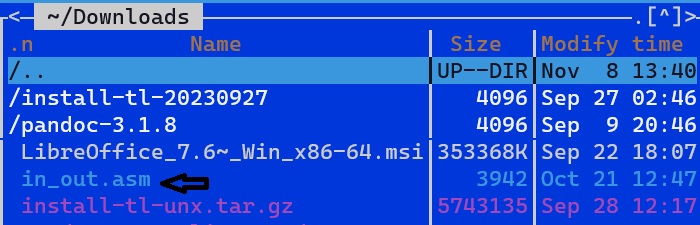


Рис. 4.9. Скачанный файл

С помощью функциональной клавиши f5 копирую файл in\_out.asm из каталога Downloads в созданный каталог lab05. (рис. 4.10)

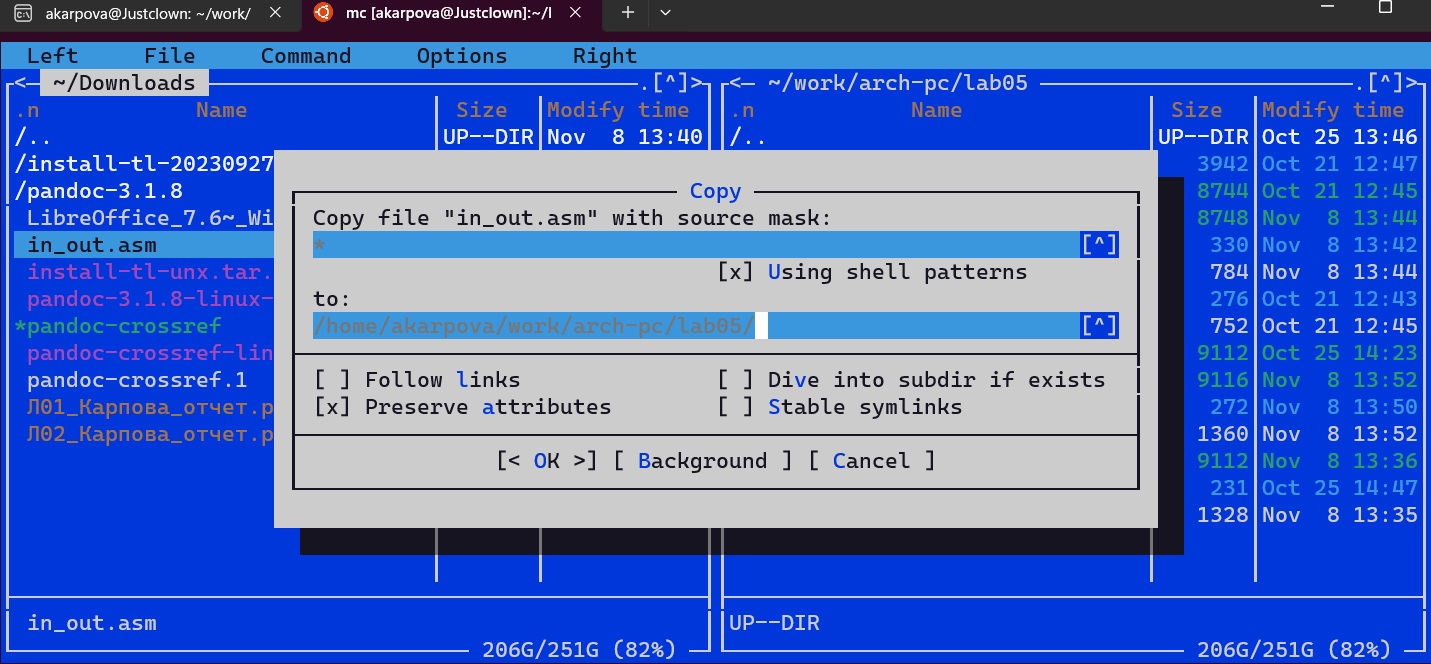
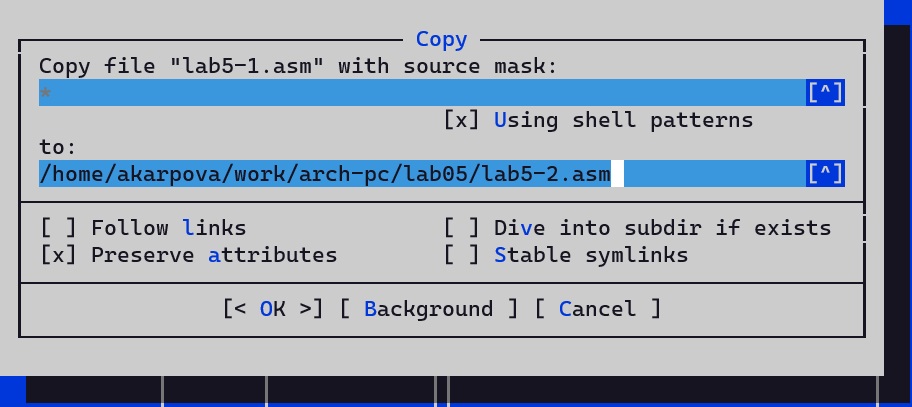


Рис. 4.10. Копирование файла

С помощью функциональной клавиши f5 копирую файл lab5-1 в тот же каталог, но с другим именем, для этого в появившемся окне mc прописываю имя для копии файла. (рис. 4.11)



4.11. Копирование файла

Изменяю содержимое файла lab5-2.asm редакторе nano, чтобы в программе использовались подпрограммы из внешнего файла in\_out.asm. (рис. 4.12)



Рис. 4.12. Изменение содержимого файла

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. 4.13)

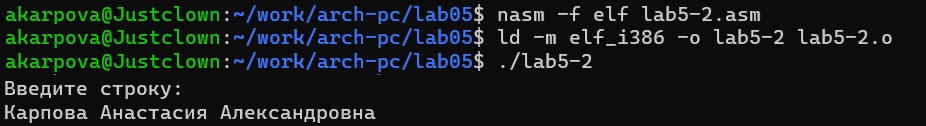


Рис. 4.13. Компоновка объектного файла и запуск исполняемого файла

Открываю файл lab5-2.asm в редакторе в nano функциональной клавишей f4. Изменяю в нем подпрограмму sprintLF на sprint. Сохраняю изменения и открываю файл для просмотра, чтобы проверить сохранение действий. (рис. 4.1)



Рис. 4.14. Изменение spintLF на sprint

Снова транслирую файл, выполняю компоновку созданного объектного файла, запускаю новый исполняемый файл. (рис. 4.15)

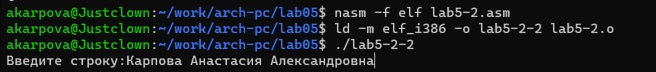


Рис. 4.15. Компоновка объектного файла и запуск исполняемого файла

Выполнение заданий для самостоятельной работы

1. Создаю копию файла lab5-1.asm с именем lab5-1-1.asm с помощью клавиши f5. (рис. 4.16)

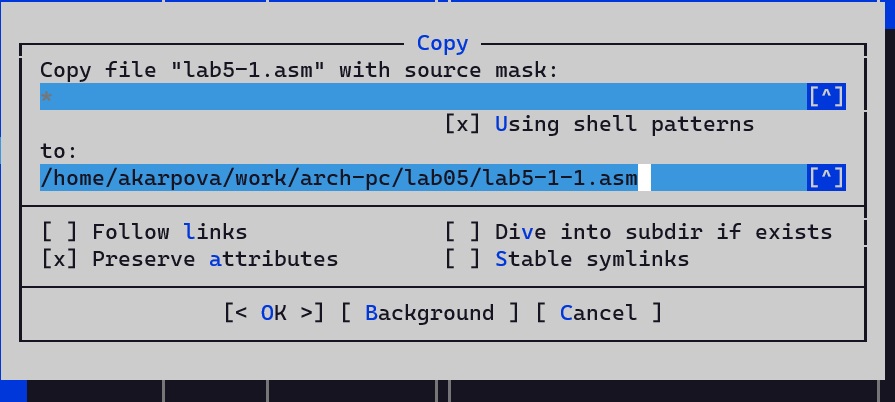


Рис. 4.16. Копия файла с другим именем

С помощью клавиши f4 открываю созданный файл для редактирования. Изменяю программу так, чтобы кроме вывода приглашения и запроса ввода, она выводила вводимую пользователем строку. (рис. 4.17)



Рис. 4.17. Редактирование

1. 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, и программа соответственно выводит введенные мною данные (рис. 4.18)

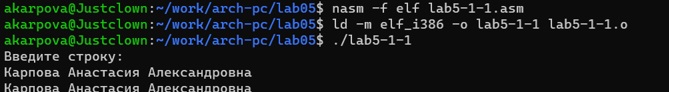


Рис. 4.18. Создание объектного файла, его компоновка и запуск исполняемого файла

1. Создаю копию файла lab5-2.asm с именем lab5-2-1.asm с помощью клавиши f5. (рис. 4.19)

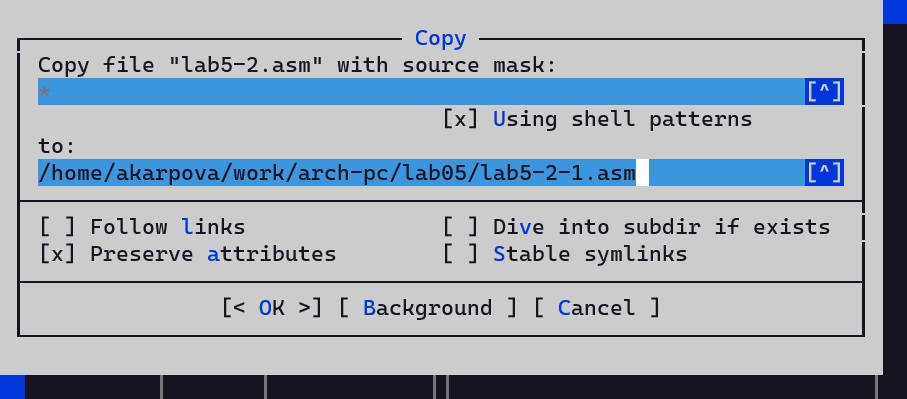


Рис. 4.19. Создание копии файла с новым именем

С помощью клавиши f4 открываю созданный файл для редактирования. Изменяю программу так, чтобы кроме вывода приглашения и запроса ввода, она выводила вводимую пользователем строку. (рис. 4.20)



Рис. 4.20. Редактирование файла

1. 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. 4.21)

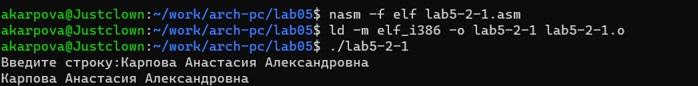


Рис. 4.21. Создание объектного файла и его компоновка, запуск имполняемого файла

# 5 Выводы

В ходе выполнения лабораторной работы я приобрела практические навыки работы в Midnight Commander, а также освоила инструкции языка ассемблера mov и int.

# Список литературы

1. Архитектура ЭВМ