Отчёт по лабораторной работе №5

Дисциплина: Архитектура компьютера

София Андреевна Кудякова

Содержание

# 1 Цель работы

Цель данной лабораторной работы - научиться работать с Midnight Commander, а также освоить инструкции языка ассамблера mov и int.

# 2 Задания

1. Основы работы с mc
2. Структура программы на языке ассамблера 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 (учетверённое 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. Синтаксис директив определения данных следующий: DB [, ] [, ]. Для объявления неинициированных данных в секции .bss используются директивы resb, resw, resd и другие, которые сообщают ассемблеру, что необходимо зарезервировать заданное количество ячеек памяти. Инструкция языка ассемблера mov предназначена для дублирования данных источника в приёмнике. В общем виде эта инструкция записывается в виде

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. В общем виде она записывается в виде

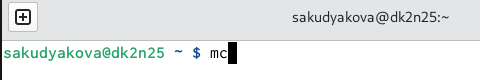
int n

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

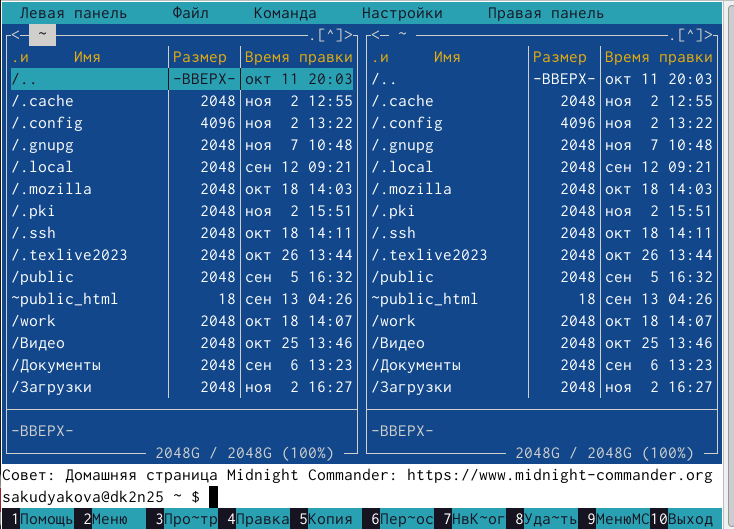
## 4.1 Основы работы с mc

Ввожу команду mc, с помощью которой открывается Midnight Commander. (рис. ??).



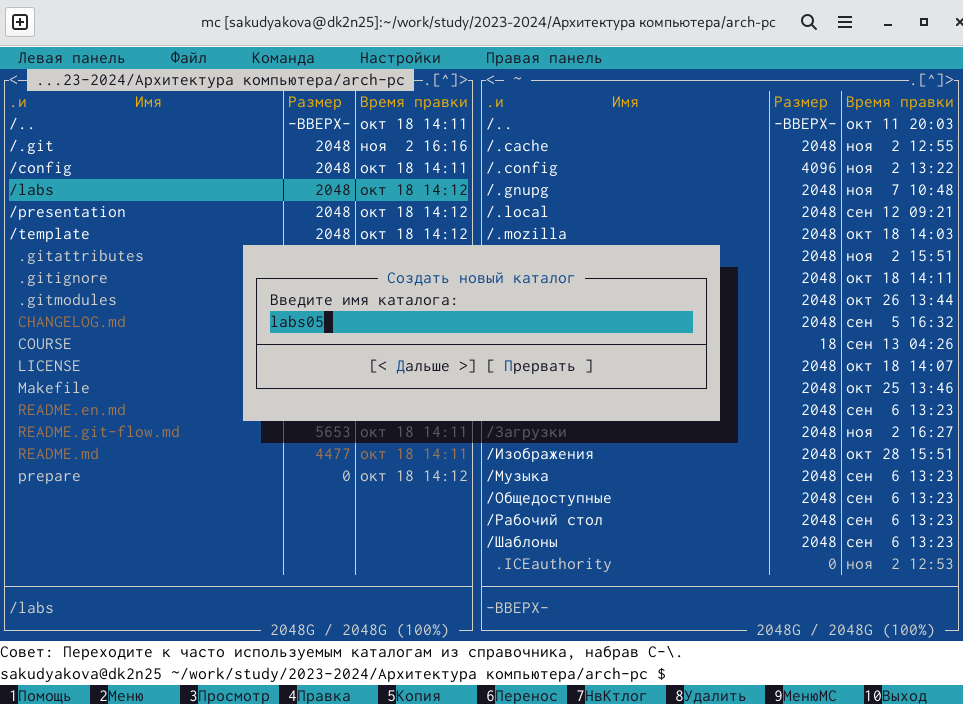
Команда mc

Убеждаюсь, что команда сработала, так как открылся mc. (рис. ??).



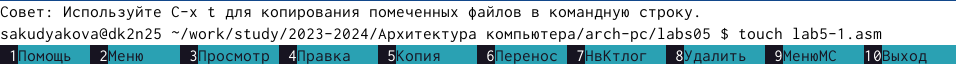
Открытый mc

Перехожу в каталог ~/work/arch-pc и с помощью функциональной клавиши F7 создаю папку lab05 и перехожу в созданный каталог. (рис. ??).



Создание каталога

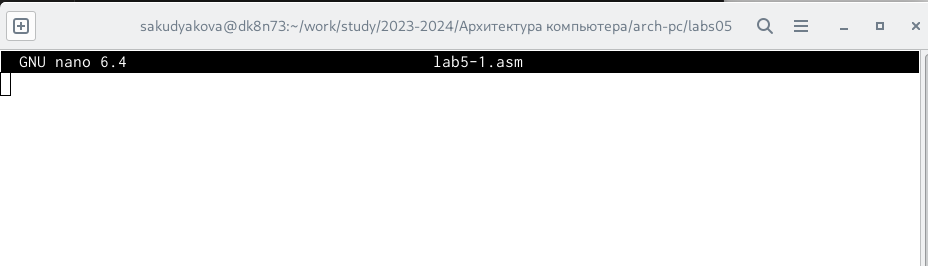
Пользуясь строкой ввода и командой touch создаю файл lab5-1.asm. (рис. ??).



Создание файла

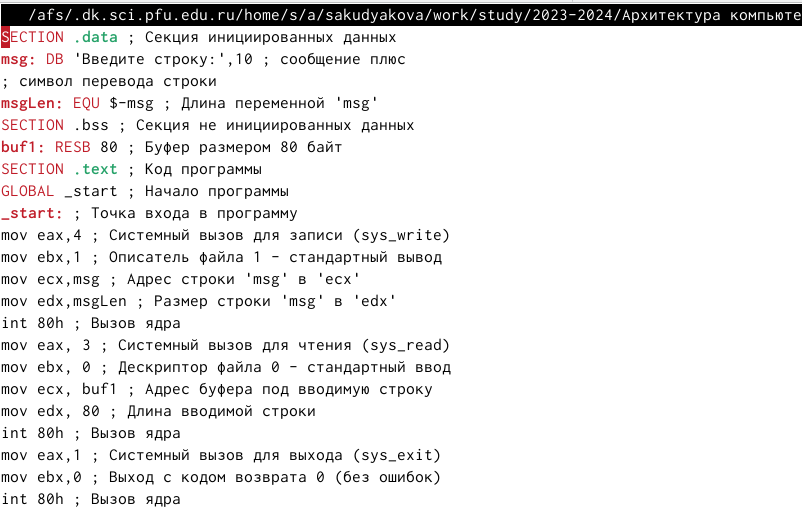
## 4.2 Структура программы на языке ассамблера NASM

С помощью клавиши F4 открываю созданный файл в редакторе nano. (рис. ??).



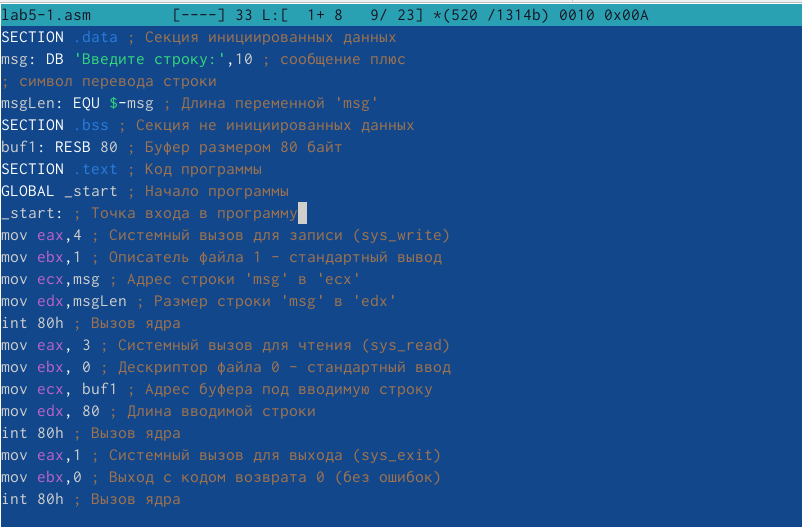
Открытие файла в редакторе

Ввожу код программы для запроса строки у пользователя в файл. Затем выхожу из файла, сохранив все изменения. (рис. ??).



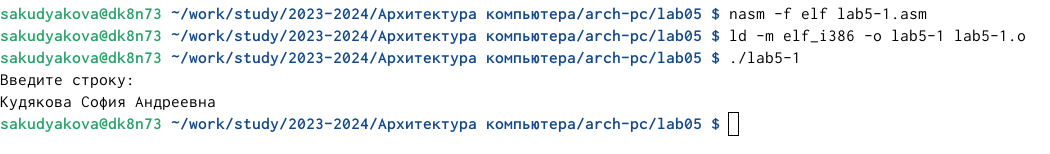
Ввод кода программы

С помощью клавиши F3 открываю файл для просмотра и убеждаюсь, что внесенный код программы был сохранен. (рис. ??).



Просмотр файла

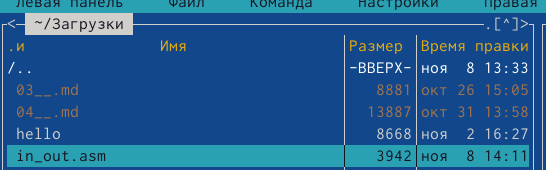
Транслируйте текст программы lab5-1.asm в объектный файл. Выполняю компоновку объектного файла и запускаю получившийся исполняемый файл. Программа выводит строку ‘Введите строку:’ и ожидает ввода с клавиатуры. На запрос введите свои ФИО.(рис. ??).



Исполнение файла

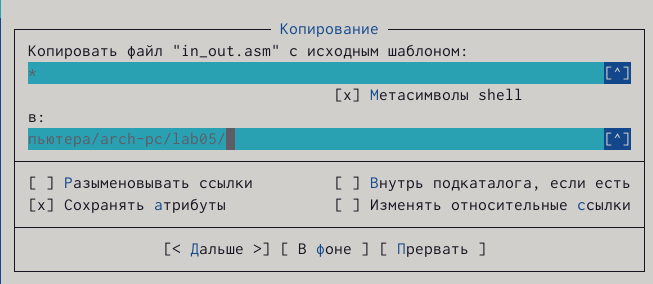
## 4.3 Подключение внешнего файла

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



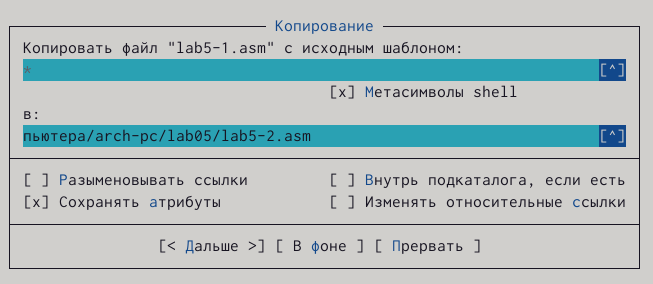
Скачивание файла

Копирую файл in\_out.asm из каталога “Загрузки” в каталог lab05 с помощью функциональной клавиши F5. (рис. ??).



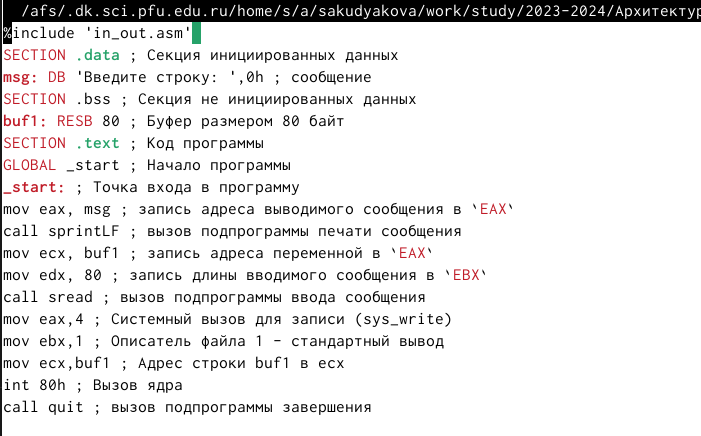
Копирования файла

С помощью функциональной клавиши F6 создаю копию файла lab5-1.asm с именем lab5-2.asm. Выделяю файл lab5-1.asm, нажимаю клавишу F6, ввожу имя файла lab5-2.asm и нажимю Enter. (рис. ??).



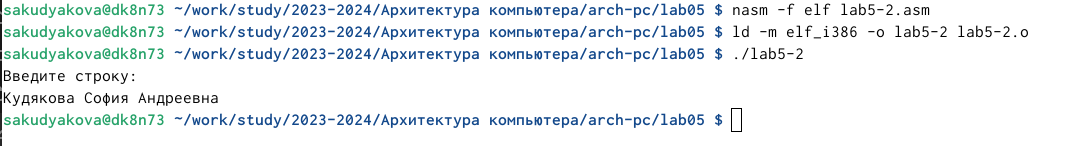
Копирование файла

Исправляю текст программы в файле lab5-2.asm с использованием подпрограмм из внешнего файла in\_out.asm. (рис. ??).



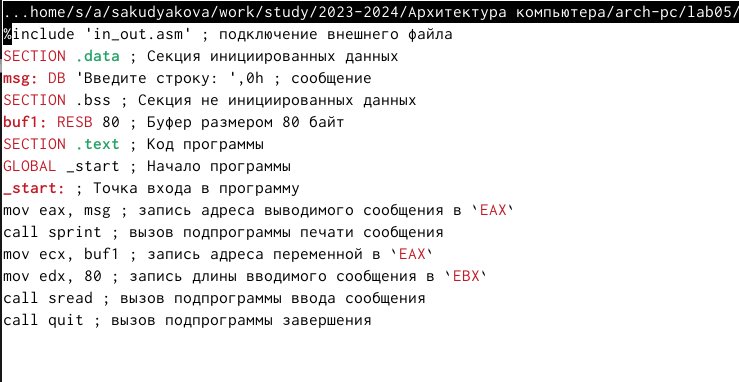
Редактирование файла

Транслируйте текст программы lab5-2.asm в объектный файл. Выполняю компоновку объектного файла и запускаю получившийся исполняемый файл. Программа выводит строку ‘Введите строку:’ и ожидает ввода с клавиатуры. (рис. ??).



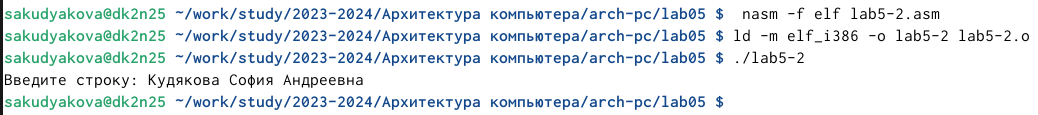
Исполнение файла

В файле lab5-2.asm меняю подпрограмму sprintLF на sprin. (рис. ??).



Изменение подпрограммы

Затем транслирую файл, выполняю компоновку объектного файла и заупскаю новый исполняемый файл. (рис. ??).

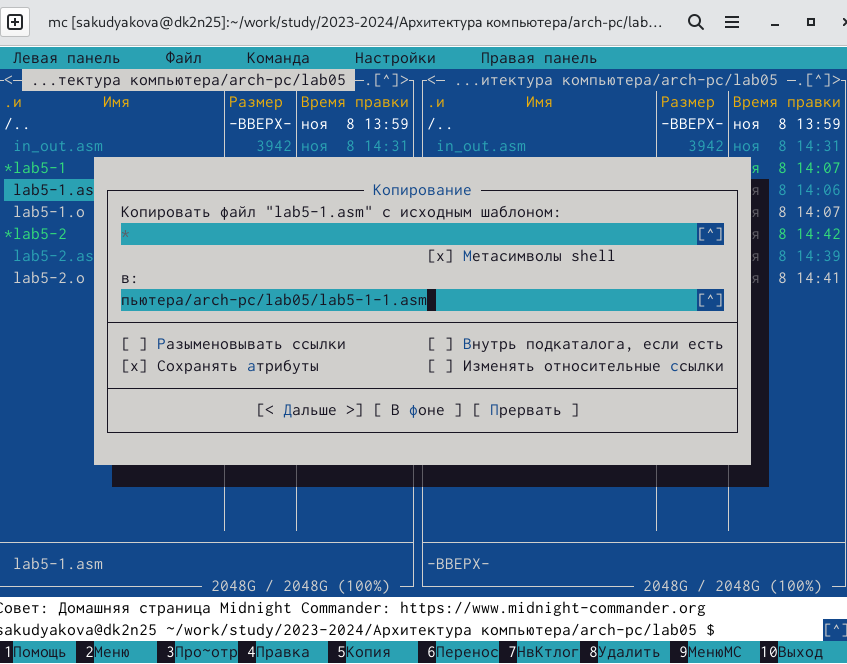


Исполнение файла

Разница между подпрограммами sprintLF и sprint заключается в том, что в первом случае(sprintLF) ввод запрашивается с новой строки, а во втором случае(sprint) - с той же.

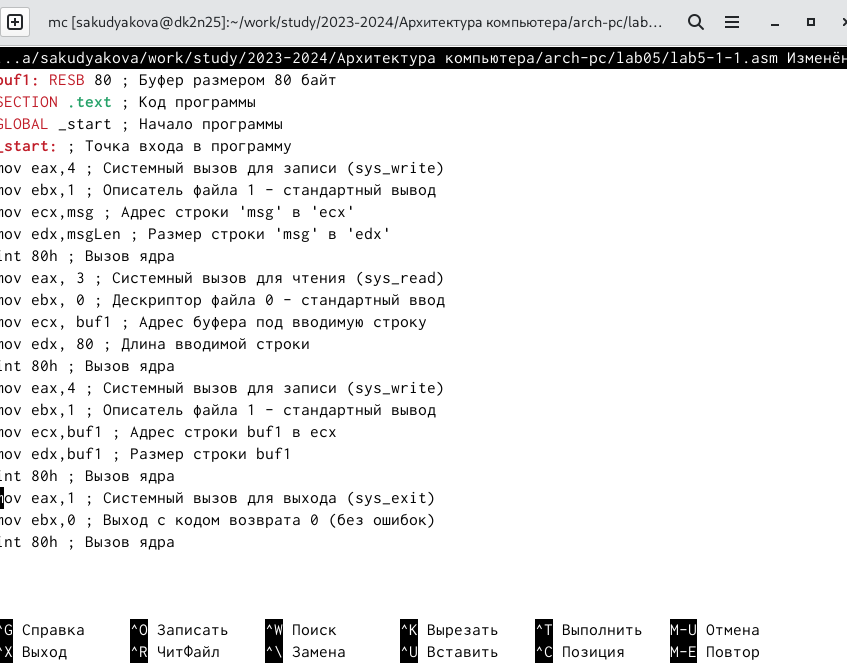
## 4.4 Выполнение заданий для самостоятельной работы

1. Копирую файл lab5-1.asm и создаю нвый с именем lab5-1-1.asm. (рис. ??).



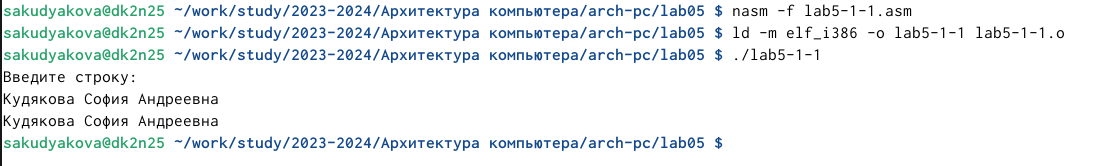
Копирование файла

Открываю файл с помощью F4 для редактирования. Изменяю программу так, чтобы она выводила вывод приглашения, запрос ввода и вводимую пользователем строку. (рис. ??).



Редактирование файла

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, далее программа выводит введенные мною данные. (рис. ??).

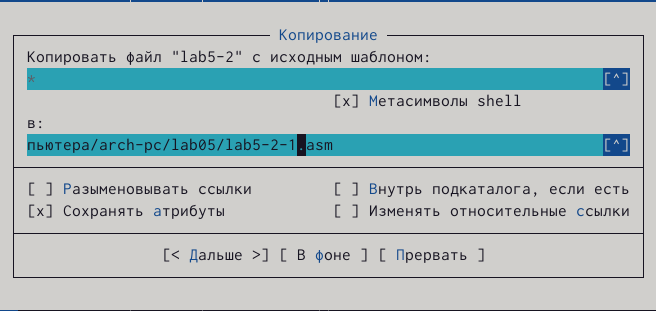


Исполнение файла

Код программы из п.1:

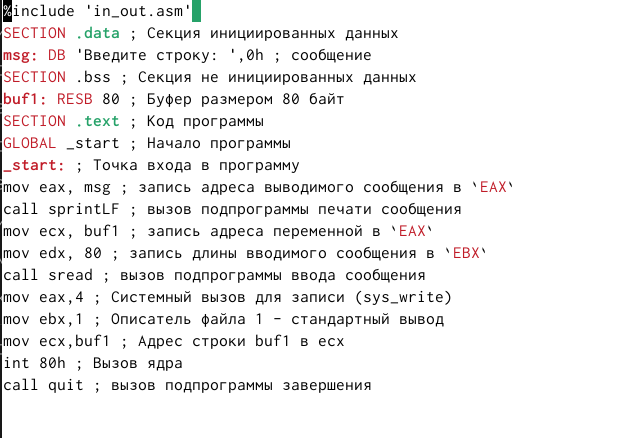
SECTION .text ; Код программы  
GLOBAL \_start ; Начало программы  
\_start: ; Точка входа в программу  
mov eax,4 ; Системный вызов для записи (sys\_write)  
mov ebx,1 ; Описатель файла 1 - стандартный вывод  
mov ecx,msg ; Адрес строки 'msg' в 'ecx'  
mov edx,msgLen ; Размер строки 'msg' в 'edx'  
int 80h ; Вызов ядра  
mov eax, 3 ; Системный вызов для чтения (sys\_read)  
mov ebx, 0 ; Дескриптор файла 0 - стандартный ввод  
mov ecx, buf1 ; Адрес буфера под вводимую строку  
mov edx, 80 ; Длина вводимой строки  
int 80h ; Вызов ядра  
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  
mov edx,buf1 ; Размер строки buf1  
int 80h ; Вызов ядра  
mov eax,1 ; Системный вызов для выхода (sys\_exit)  
mov ebx,0 ; Выход с кодом возврата 0 (без ошибок)  
int 80h ; Вызов ядра

1. Копирую файл lab5-2.asm и создаю нвый с именем lab5-2-1.asm. (рис. ??).



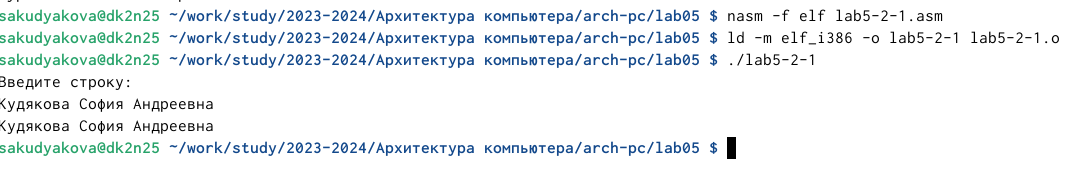
Копирование файла

Открываю файл с помощью F4 для редактирования. Изменяю программу так, чтобы она выводила вывод приглашения, запрос ввода и вводимую пользователем строку. (рис. ??).



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, ввожу свои ФИО, далее программа выводит введенные мною данные. (рис. ??).



Редактирование файла

Код программы из п.3:

%include 'in\_out.asm'   
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 sprintLF ; 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

# 5 Выводы

В ходе данной лабораторной работы я научилась работать с Midnight Commander, а также освоила инструкции языка ассамблера mov и int.

# Список литературы

[Архитектура ЭВМ](https://esystem.rudn.ru/pluginfile.php/2089085/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%965.%20%D0%9E%D1%81%D0%BD%D0%BE%D0%B2%D1%8B%20%D1%80%D0%B0%D0%B1%D0%BE%D1%82%D1%8B%20%D1%81%20Midnight%20Commander%20%28%29.%20%D0%A1%D1%82%D1%80%D1%83%D0%BA%D1%82%D1%83%D1%80%D0%B0%20%D0%BF%D1%80%D0%BE%D0%B3%D1%80%D0%B0%D0%BC%D0%BC%D1%8B%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%A1%D0%B8%D1%81%D1%82%D0%B5%D0%BC%D0%BD%D1%8B%D0%B5%20%D0%B2%D1%8B%D0%B7%D0%BE%D0%B2%D1%8B%20%D0%B2%20%D0%9E%D0%A1%20GNU%20Linux.pdf)