

Отчет по лабораторной работе №5

Дисциплина архитектура компьютера

Ахатов Эмиль Эрнстович

Содержание

1	Цель работы	4
2	Задание	5
3	Теоретическое введение	6
4	Выполнение лабораторной работы	8
4.1	Основы работы с mc	8
4.2	Структура программы на языке ассемблера NASM	10
4.3	Работа с расширенным синтаксисом командной строки NASM . . .	10
5	Выводы	18

Список иллюстраций

4.1	Открытый тс	8
4.2	Перемещение между директорями	9
4.3	Создание каталога	9
4.4	Создание файла	10
4.5	Открытие файла	11
4.6	Ввод программы	11
4.7	Компиляция файла и передача на обработку	12
4.8	Исполнение файла	12
4.9	Скачанный файл	13
4.10	Копирование файла	13
4.11	Копирование файла	14
4.12	Редактирование файла	14
4.13	Исполнение файла	14
4.14	Изменение файла	15
4.15	Исполнение файла	15
4.16	Редактирование файла	16
4.17	Выполнение программы	16
4.18	Копирование файла	16
4.19	Редактирование файла	17
4.20	Выполнение программы	17

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 тс
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 размером 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 (четырёх- 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 `int` предназначена 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, введя в терминал mc

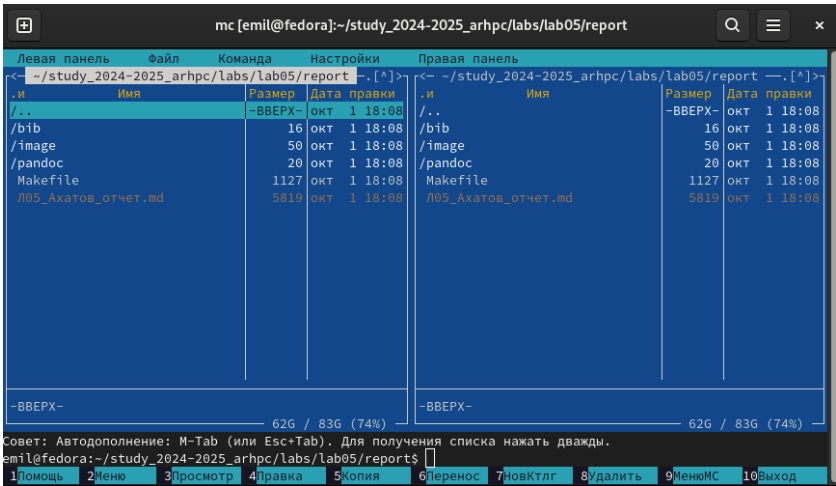


Рис. 4.1: Открытый mc

Перехожу в каталог ~/work/study/2023-2024/Архитектура Компьютера/arch-рс

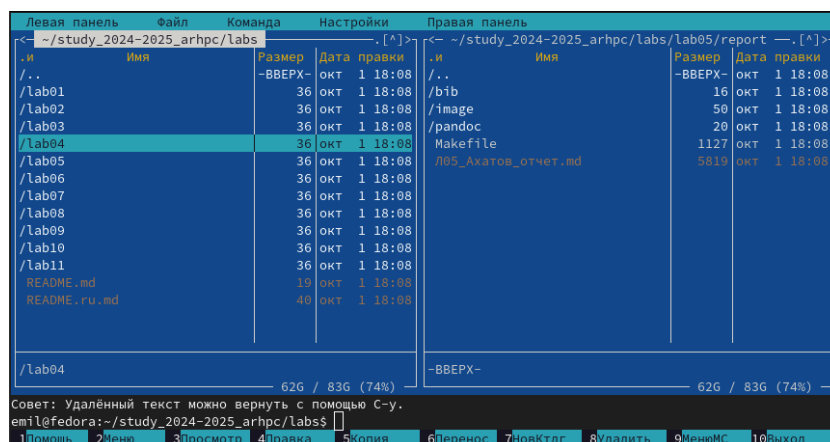


Рис. 4.2: Перемещение между директориями

С помощью клавиши F7 создаю каталог lab05

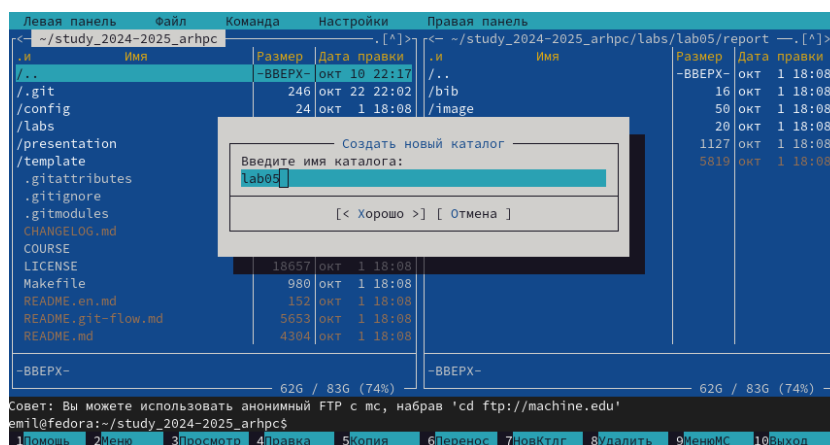


Рис. 4.3: Создание каталога

Перехожу в созданный каталог, в строке ввода прописываю команду touch lab05-1.asm, чтобы создать файл

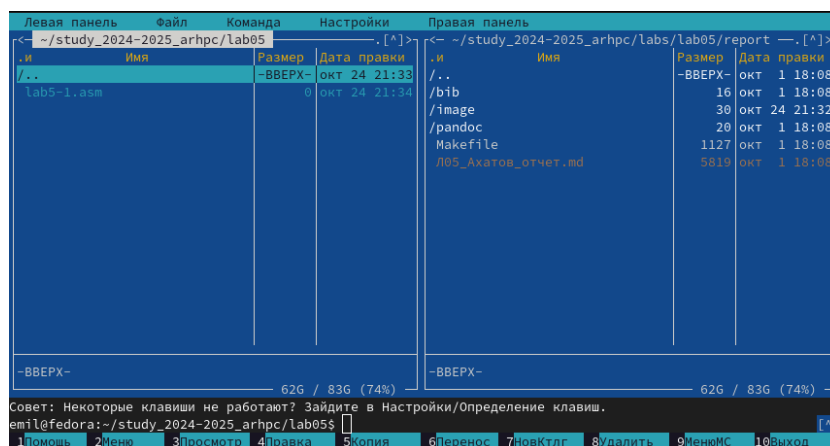


Рис. 4.4: Создание файла

4.2 Структура программы на языке ассемблера NASM

Превращаю текст программы для вывода “Hello world!” в объектный код с помощью транслятора NASM, используя команду `nasm -f elf hello.asm`, ключ `-f` указывает транслятору `nasm`, что требуется создать бинарный файл в формате ELF (рис. 4.5). Далее проверяю правильность выполнения команды с помощью утилиты `ls`: действительно, создан файл “hello.o”

4.3 Работа с расширенным синтаксисом командной строки NASM

С помощью функциональной клавиши F4 открываю файл `lab5-1.asm` для редактирования во встроенном редакторе.

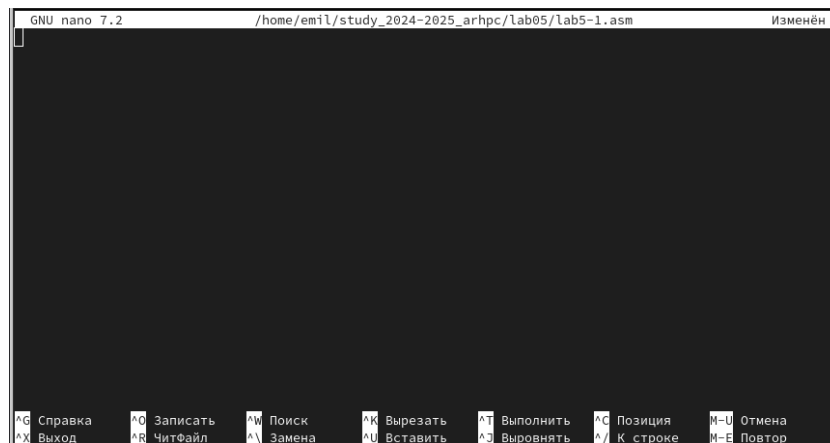


Рис. 4.5: Открытие файла

Ввожу в файл код программы для запроса строки у пользователя

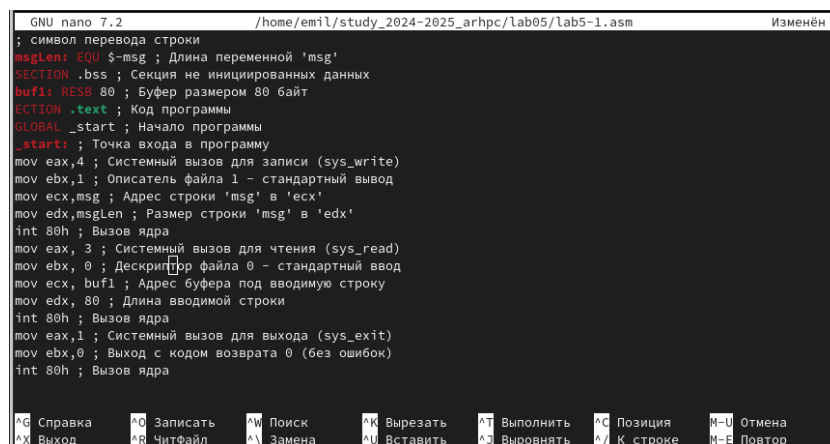


Рис. 4.6: Ввод программы

выхожу из файла (Ctrl+X), сохраняя изменения (Y, Enter). Открыл файл и убедился, что файл содержит текст программы. Транслирую текст программы файла в объектный файл командой `nasm -f elf lab5-1.asm`. Создался объектный файл `lab5-1.o`. Выполняю компоновку объектного файла с помощью команды `ld -m elf_i386 -o lab5-1 lab5-1.o`. Создался исполняемый файл `lab5-1`.

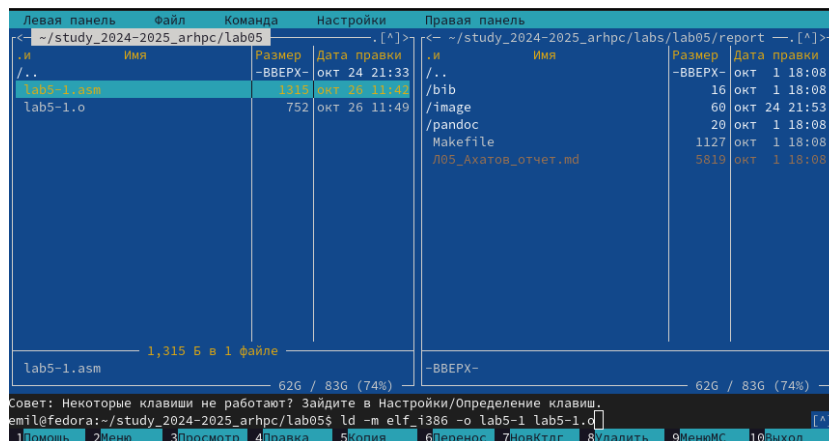


Рис. 4.7: Компиляция файла и передача на обработку

Запускаю исполняемый файл. Программа выводит строку “Введите строку:” и ждет ввода с клавиатуры, я ввожу свои ФИО, на этом программа заканчивает свою работу

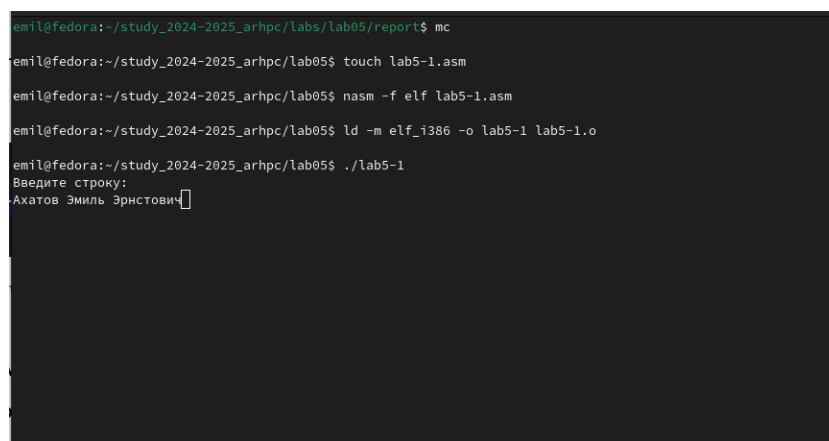


Рис. 4.8: Исполнение файла

##Подключение внешнего файла

Скачиваю файл in_out.asm со страницы курса в ТУИС.

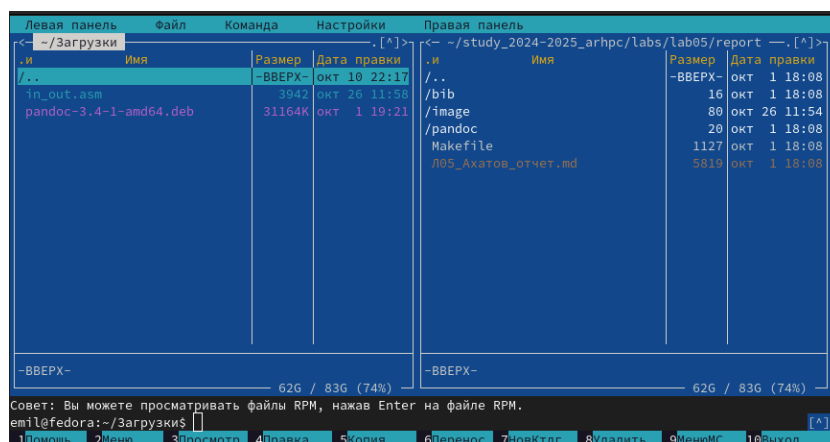


Рис. 4.9: Скачанный файл

Копирую файл in_out.asm из каталога Загрузки в созданный каталог lab05

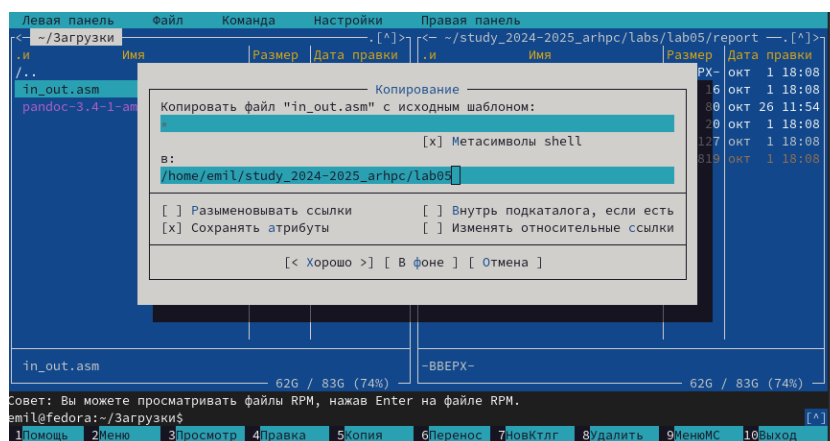


Рис. 4.10: Копирование файла

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

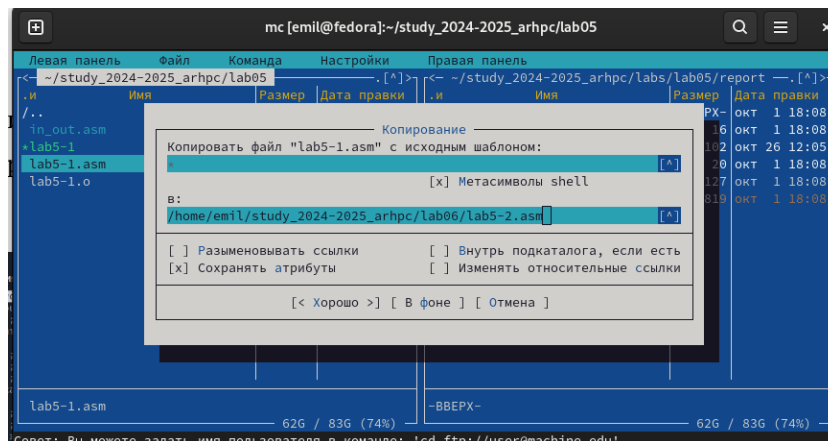


Рис. 4.11: Копирование файла

Изменяю содержимое файла lab5-2.asm во встроенном редакторе nano, чтобы в программе использовались подпрограммы из внешнего файла in_out.asm.

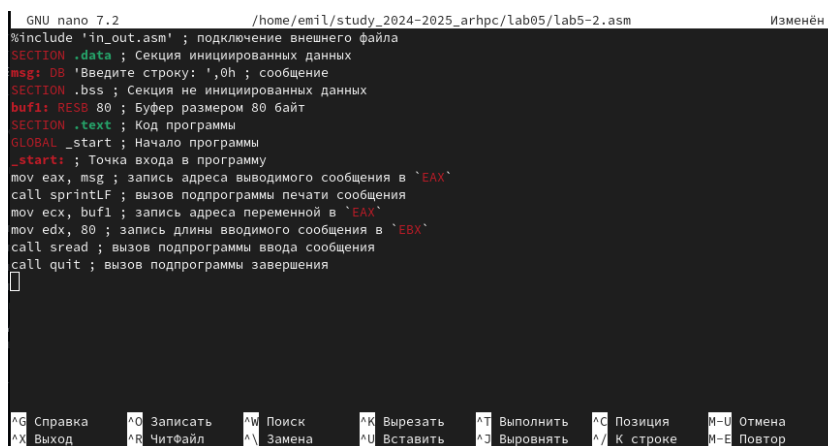


Рис. 4.12: Редактирование файла

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.

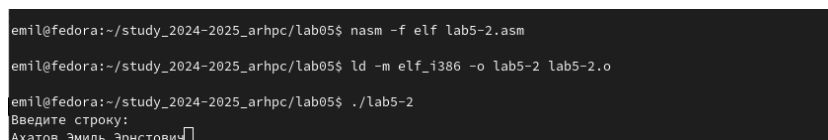
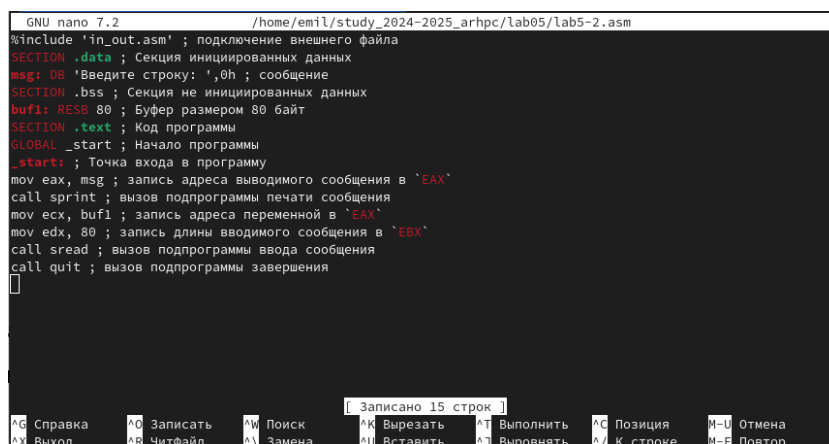


Рис. 4.13: Исполнение файла

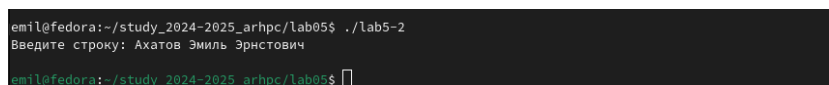
Открываю файл lab5-2.asm для редактирования в nano. Изменяю в нем подпрограмму sprintLF на sprint. Сохраняю изменения и открываю файл для просмотра, чтобы проверить сохранение действий



```
GNU nano 7.2 /home/emil/study_2024-2025_arhpc/lab05/lab5-2.asm
#include "in_out.asm" ; подключение внешнего файла
SECTION .data ; Секция инициализированных данных
msg: DB 'Введите строку: ',0h ; сообщение
SECTION .bss ; Секция не инициализированных 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 выводимого сообщения в `EBX`
call sread ; вызов подпрограммы ввода сообщения
call quit ; вызов подпрограммы завершения
[]
```

Рис. 4.14: Изменение файла

Снова транслирую файл, выполняю компоновку созданного объектного файла, запускаю новый исполняемый файл



```
emil@fedora:~/study_2024-2025_arhpc/lab05$ ./lab5-2
Введите строку: Ахатов Эмиль Эрнстович
emil@fedora:~/study_2024-2025_arhpc/lab05$
```

Рис. 4.15: Исполнение файла

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

#Выполнение заданий для самостоятельной работы

Создаю копию файла lab5-1.asm с именем lab5-1-1.asm. открываю созданный файл для редактирования. Изменяю программу так, чтобы кроме вывода приглашения и запроса ввода, она выводила вводимую пользователем строку.

```

msgLen: EQU $-msg ; Длина переменной 'msg'
SECTION .bss ; Секция не иницированных данных
buf1: RESB 80 ; Буфер размером 80 байт
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)

```

Рис. 4.16: Редактирование файла

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.

```

emil@fedora:~/study_2024-2025_arhpc/lab05$ nasm -f elf lab5-1-1.asm
emil@fedora:~/study_2024-2025_arhpc/lab05$ ld -m elf_i386 -o lab5-1-1 lab5-1-1.o
emil@fedora:~/study_2024-2025_arhpc/lab05$ ./lab5-1-1
Введите строку:
Ахатов Эмиль Эрнстович
Ахатов Эмиль Эрнстович

```

Рис. 4.17: Выполнение программы

Создаю копию файла lab5-2.asm с именем lab5-2-1.asm

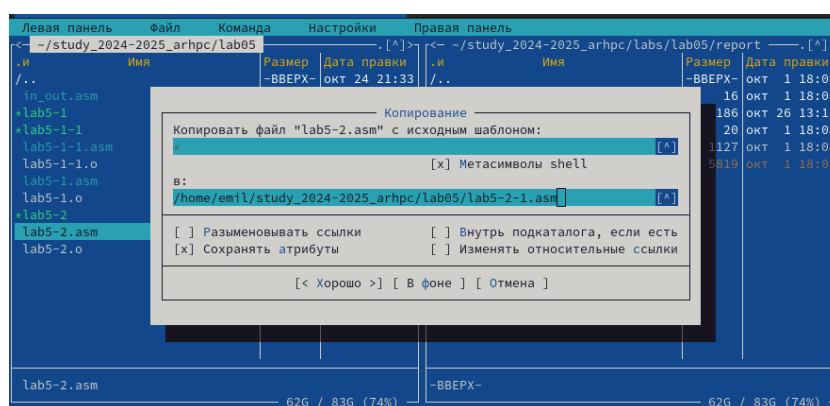


Рис. 4.18: Копирование файла

Открываю созданный файл для редактирования. Изменяю программу так, чтобы кроме вывода приглашения и запроса ввода, она выводила вводимую пользователем строку.

```
%include 'in_out.asm'
SECTION .data ; Секция иницированных данных
msg: DB 'Введите строку: ',0h ; сообщение
SECTION .bss ; Секция не иницированных 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. 4.19: Редактирование файла

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

```
emil@fedora:~/study_2024-2025_arhpc/lab05$ nasm -f elf lab5-2-1.asm
emil@fedora:~/study_2024-2025_arhpc/lab05$ ld -m elf_i386 -o lab5-2-1 lab5-2-1.o
emil@fedora:~/study_2024-2025_arhpc/lab05$ ./lab5-2-1
Введите строку: Ахатов Эмиль Эрнстович
Ахатов Эмиль Эрнстович
```

Рис. 4.20: Выполнение программы

5 Выводы

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