РОССИЙСКИЙ УНИВЕРСИТЕТ ДРУЖБЫ НАРОДОВ

Факультет физико-математических и естественных наук

Кафедра прикладной информатики и теории вероятностей

ОТЧЕТ
ПО ЛАБОРАТОРНОЙ РАБОТЕ № 5

дисциплина: Архитектура компьютера

Студент: Бондаренко София Николаевна

Группа: НБИбд-01-25

МОСКВА

2025 г.

Оглавление

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 работы

1. Основы работы с mc

Открываю Midnight Commander, введя в терминал mc (рис. 1).

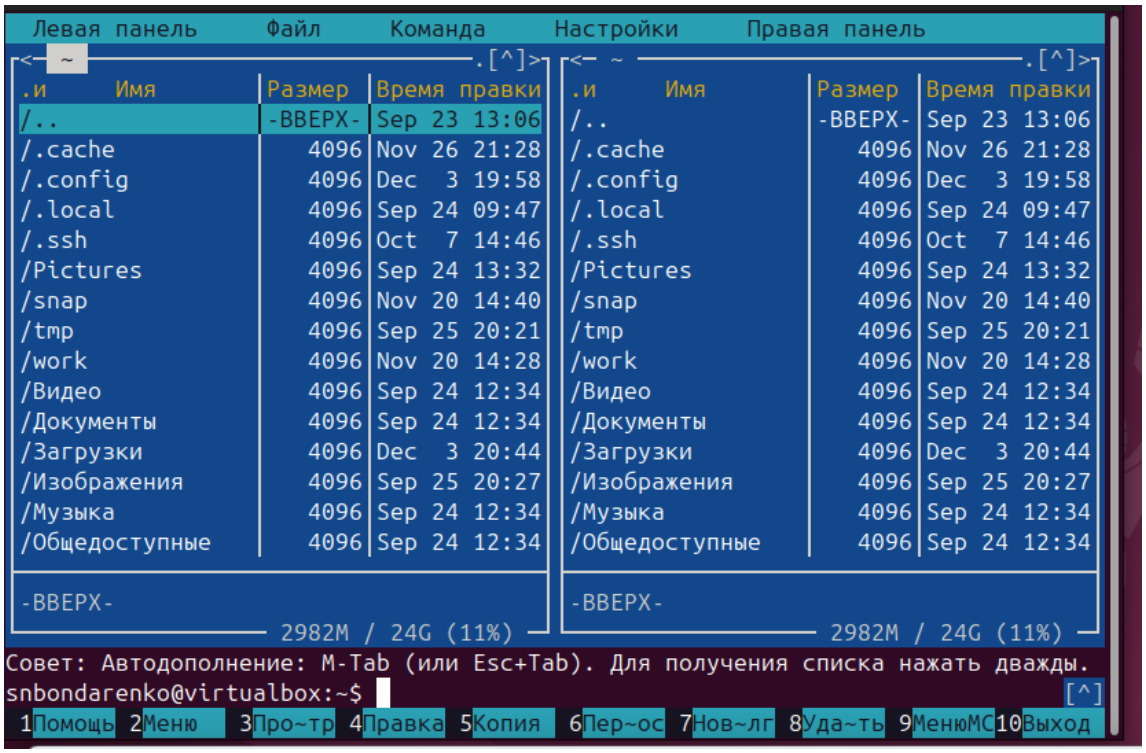

рис.1 Открытый mc

Перехожу в каталог ~/work/study/2022-2023/Архитектура Компьютера/arch-pc, используя файловый менеджер mc (рис. 2)

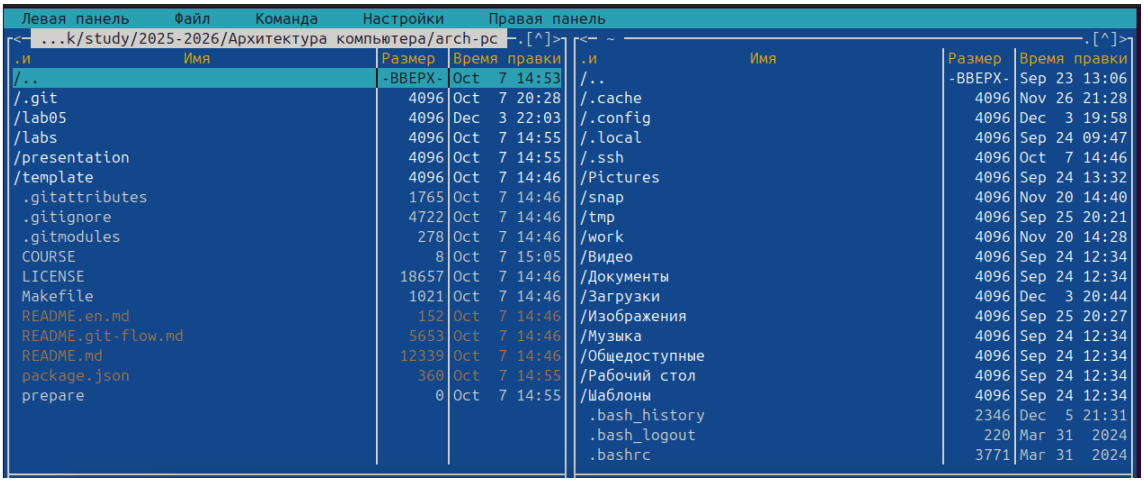

рис.2 Перемещение между директориями

С помощью функциональной клавиши F7 создаю каталог lab06 (рис. 3).

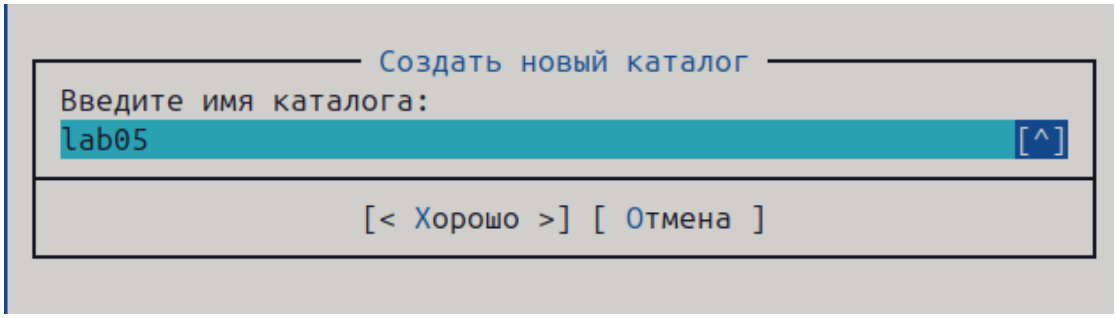

рис.3 Создание каталога

Перехожу в созданный каталог (рис. 4).

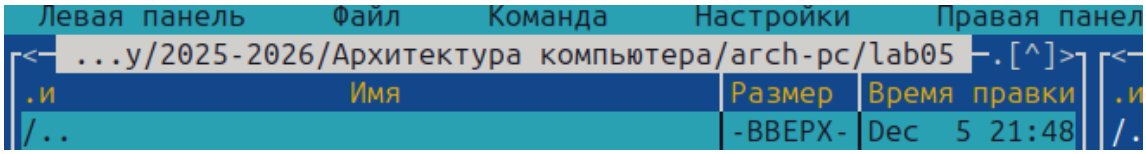

рис.4 Перемещение между директориями

В строке ввода прописываю команду touch lab05-1.asm, чтобы создать файл, в котором буду работать (рис. 5).

рис.5 Создание файла

2. *Структура программы на языке ассемблера NASM*

С помощью функциональной клавиши F4 открываю созданный файл для редактирования в редакторе nano (рис. 6).

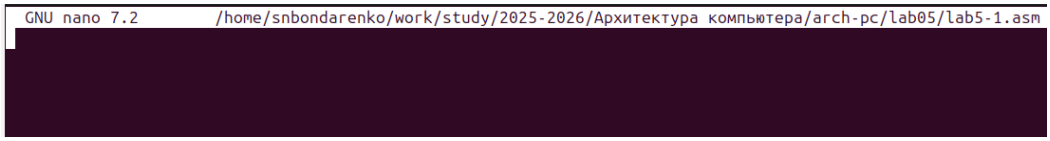

рис.6 Открытие файла для редактирования

Ввожу в файл код программы для запроса строки у пользователя (рис. 7). Ввожу в файл код программы для запроса строки у пользователя. Далее выхожу из файла (Ctrl+X), сохраняя изменения (Y, Enter) (рис.7).

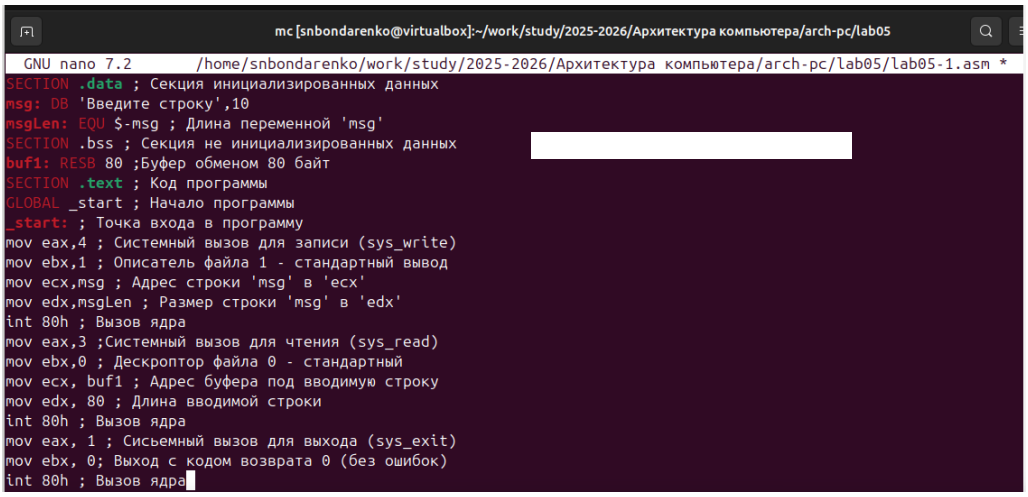

рис. 7 Редактирование файла

С помощью функциональной клавиши F3 открываю файл для просмотра, чтобы проверить, содержит ли файл текст программы (рис. 8).

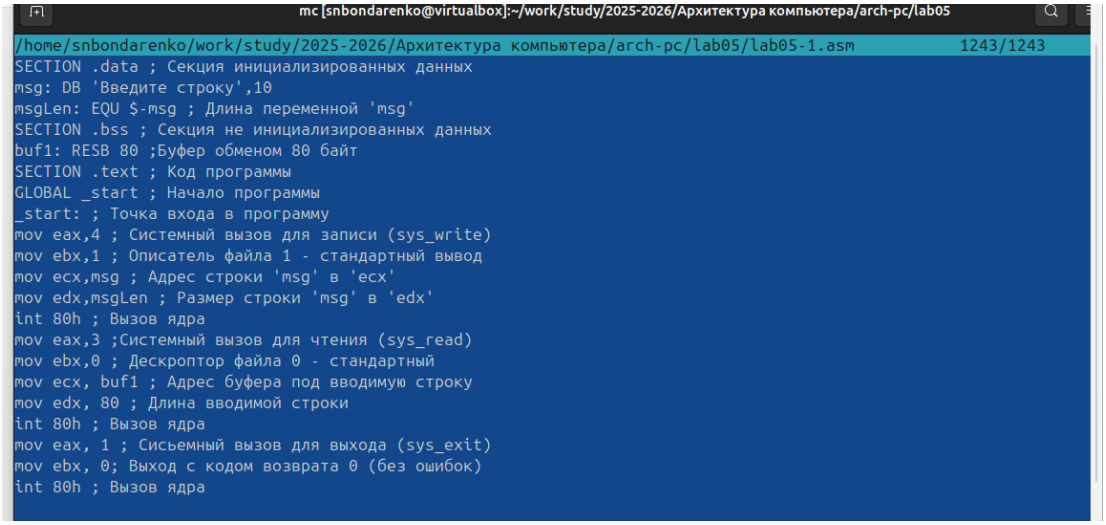

рис. 8 Открытие файла для просмотра

Транслирую текст программы файла в объектный файл командой nasm -f elf lab6-1.asm. Создался объектный файл lab05-1.o. Выполняю компоновку объектного файла с помощью команды ld -m elf_i386 -o lab05-1 lab05-1.o (рис. 9). Создался исполняемый файл lab05-1.

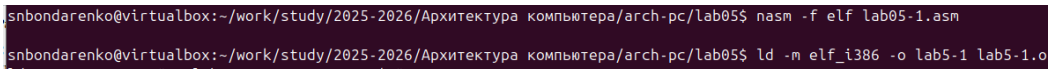

рис. 9 Компиляция файла и передача на обработку компоновщику

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

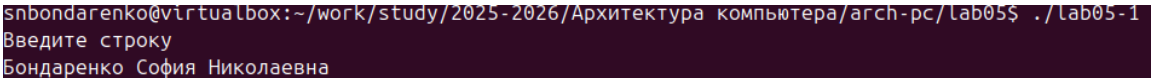

рис. 10 Исполнение файла

3. Подключение внешнего файла

Скачиваю файл in_out.asm со страницы курса в ТУИС. Он сохранился в каталог “Загрузки” (рис. 11).

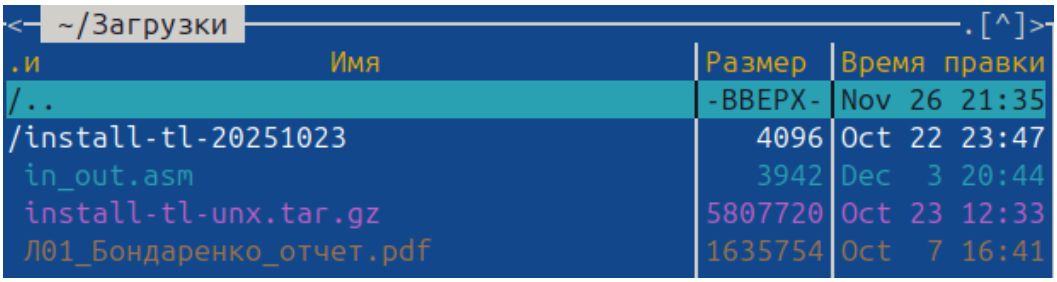

рис. 11 Скачанный файл

С помощью функциональной клавиши F5 копирую файл in_out.asm из каталога Загрузки в созданный каталог lab05 (рис. 12).

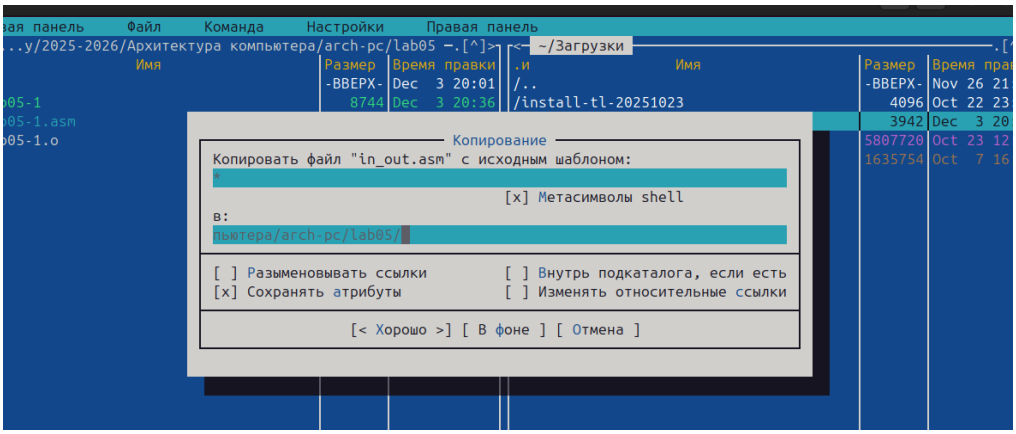

рис. 12 Копирование файла

С помощью функциональной клавиши F5 копирую файл lab05-1 в тот же каталог, но с другим именем, для этого в появившемся окне mc прописываю имя для копии файла (рис. 13).

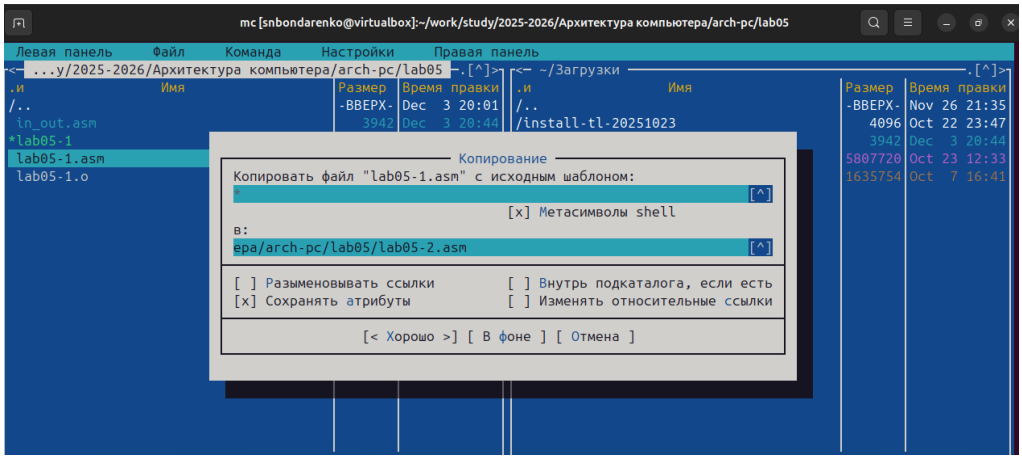

рис.13 Копирование файла

Изменяю содержимое файла lab05-2.asm во встроенном редакторе nano (рис. 14), чтобы в программе использовались подпрограммы из внешнего файла in_out.asm.

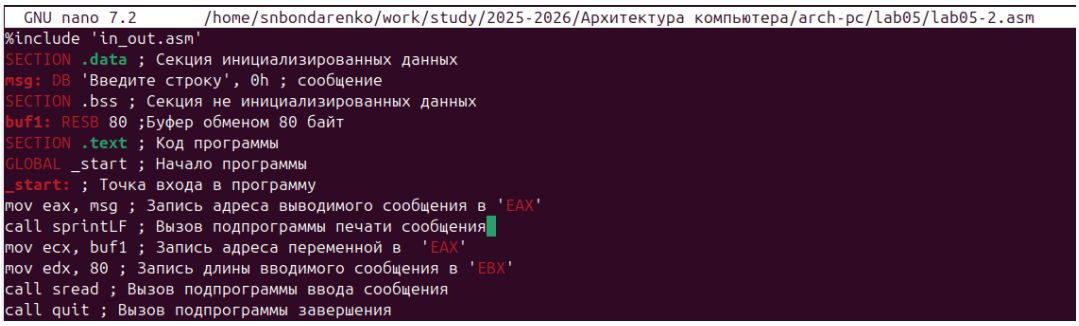

рис.14 Редактирование файла

Транслирую текст программы файла в объектный файл командой nasm -f elf lab6-2.asm. Создался объектный файл lab05-2.o. Выполняю компоновку объектного файла с помощью команды ld -m elf_i386 -o lab05-2 lab05-2.o Создался исполняемый файл lab05-2. Запускаю исполняемый файл (рис. 15).

```
snbondarenko@virtualbox:~/work/study/2025-2026/Архитектура компьютера/arch-pc/lab05$ ld  -m elf_i386 -o lab05-2 lab05-2.o

snbondarenko@virtualbox:~/work/study/2025-2026/Архитектура компьютера/arch-pc/lab05$ ld  -m elf_i386 -o lab05-2 lab05-2.o
```

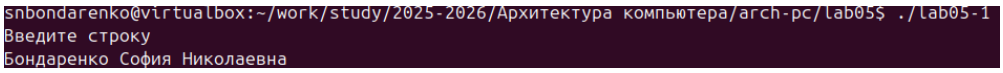

рис.15 Исполнение файла

Открываю файл lab05-2.asm для редактирования в nano функциональной клавишей F4. Изменяю в нем подпрограмму sprintLF на sprint (рис.16). Сохраняю изменения и проверяю, что изменения сохранились.

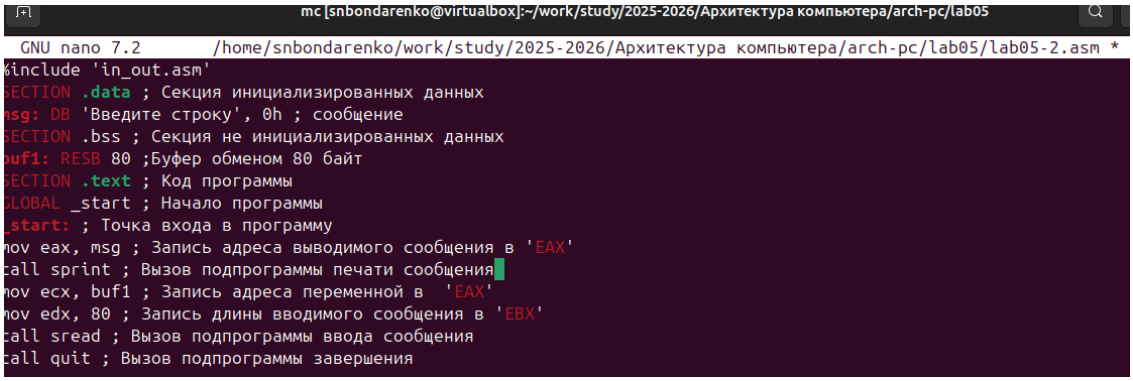

рис.16 Редактирование файла

Снова транслирую файл, выполняю компоновку созданного объектного файла, запускаю новый исполняемый файл (рис. 17).

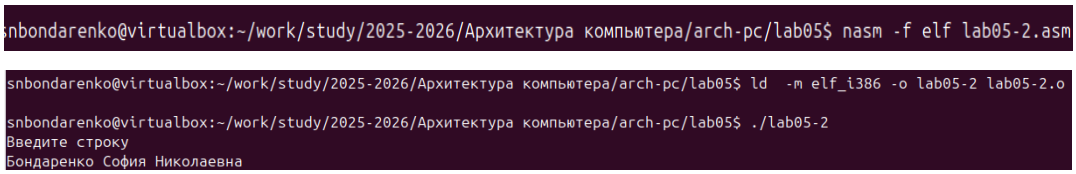

рис.17 Исполнение файла

Разница между первым исполняемым файлом lab05-2 и вторым lab05-2-2 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.

4. Выполнение заданий для самостоятельной работы

1. Создаю копию файла lab05-1.asm с именем lab05-1-1.asm с помощью функциональной клавиши F5 (рис. 18).

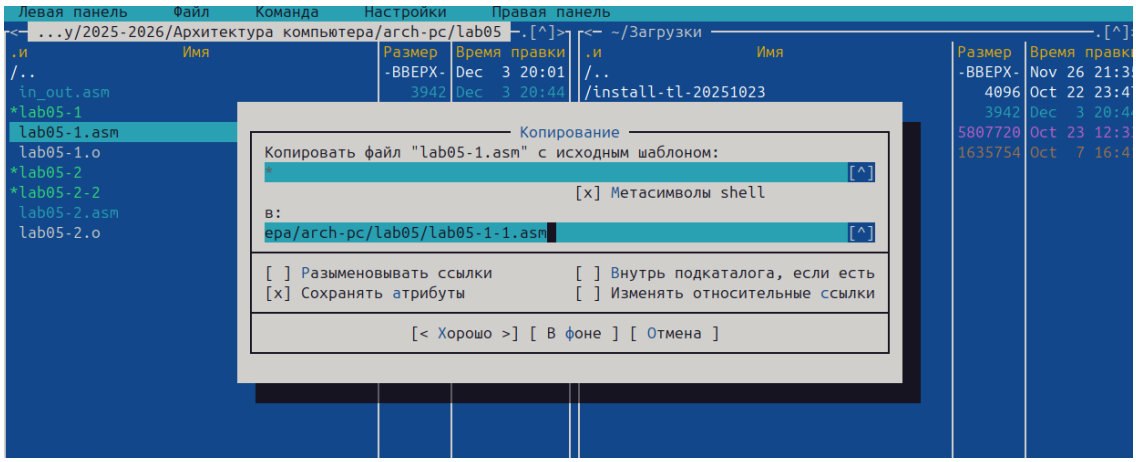

рис.18 Копирование файла

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).

```
SECTION .data ; Секция инициализированных данных
msg: DB 'Введите строку',10
msgLen: EQU $-msg ; Длина переменной 'msg'
SECTION .bss ; Секция не инициализированных данных
buf1: RESB 80 ;Буфер обменом 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 ; Дескроптор 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' в 'edx'
int 80h ; Вызов ядра
```

рис.19 Редактирование файла

2. Создаю объектный файл lab05-1-1.o, отдаю его на обработку

компоновщику, получаю исполняемый файл lab05-1-1, запускаю полученный исполняемый файл. Программа запрашивает ввод, ввожу свои ФИО, далее программа выводит введенные мною данные (рис. 20).

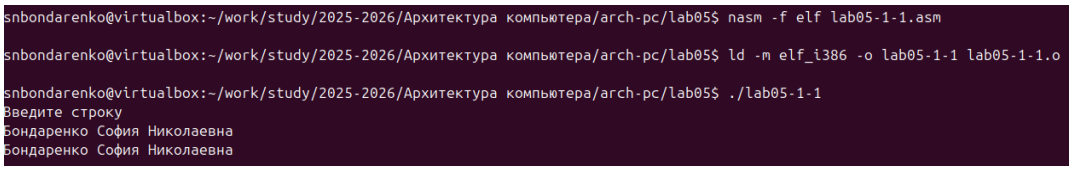

рис.20 Исполнение файла

3. Создаю копию файла lab05-2.asm с именем lab05-2-1.asm с помощью функциональной клавиши F5 (рис. 21).

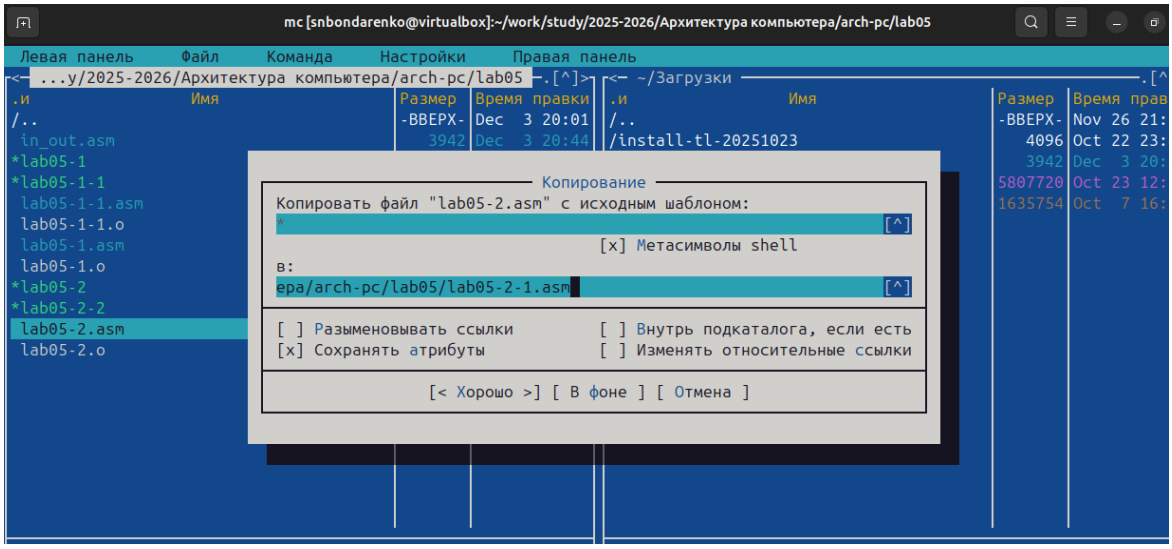

рис.21 Копирование файла

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. 22).

```
 GNU nano 7.2      /home/snbondarenko/work/study/2025-2026/Архитектура компьютера/arch-pc/lab05/lab05-2-1.asm
%include 'in_out.asm'
SECTION .data ; Секция инициализированных данных
msg: DB 'Введите строку', 0h ; сообщение
SECTION .bss ; Секция не инициализированных данных
buf1: RESB 80 ;Буфер обменом 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_writer)
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.22 Редактирование файла

4. Создаю объектный файл lab05-2-1.o, отдаю его на обработку компоновщику, получаю исполняемый файл lab05-2-1, запускаю полученный исполняемый файл. Программа запрашивает ввод без переноса на новую строку, ввожу свои ФИО, далее программа выводит введенные мною данные (рис. 23).

```
snbondarenko@virtualbox:~/work/study/2025-2026/Архитектура компьютера/arch-pc/lab05$ nasm -f elf lab05-2-1.asm

snbondarenko@virtualbox:~/work/study/2025-2026/Архитектура компьютера/arch-pc/lab05$ ld -m elf_i386 -o lab05-2-1 lab05-2-1.o

snbondarenko@virtualbox:~/work/study/2025-2026/Архитектура компьютера/arch-pc/lab05$ ./lab05-2-1
Введите строку Бондаренко София Николаевна
 Бондаренко София Николаевна
```

рис.23 Исполнение файла

5. 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

https://esystem.rudn.ru/pluginfile.php/2091239/mod_resource/content/0/Лабораторная%20работа%20№5.%20Основы%20работы%20с%20Midnight%20Commander%20%28%29.%20Структура%20программы%20на%20языке%20ассемблера%20NASM.%20Системные%20вызовы%20в%20ОС%20GNU%20Linux.pdf