Отчёт по лабораторной работе №5

Дисциплина: Архитектура компьютера

Гасанова Шакира Чингизовна

Содержание

1 Цель работы ................................................................................................................................................…...3

2 Задание .............................................................................................................................................................….4

3 Теоретическое введение ..........................................................................................................................5

4 Выполнение лабораторной работы

4.1 Основы работы с Midnight Commander....................................................……………………..........6

4.2 Структура программы на языке ассемблера NASM.....................…………......................9

4.3 Подключение внешнего файла...........................................................…………………................12

4.4 Выполнение заданий для самостоятельной работы...........……….............................15

5 Выводы ................................................................................................................................................................20

6 Источники ............................................................................................................................................…........21

# 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 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 файловый менеджер, который облегчает навигацию по структуре каталогов и позволяет выполнять базовые операции с файловой системой, делая работу с файлами более удобной и визуально понятной. Программа, написанная на языке ассемблера NASM, обычно включает три основных секции: секцию кода (SECTION .text), секцию инициализированных данных (SECTION .data), где хранятся данные, известные на этапе компиляции, и секцию неинициализированных данных (SECTION .bss), в которой резервируется память для данных, значения которых присваиваются во время выполнения программы. Для объявления инициализированных данных в секции .data применяются директивы DB, DW, DD, DQ и DT. Они резервируют память и указывают размер данных: - DB (define byte) — выделяет 1 байт для хранения значения, - DW (define word) — выделяет 2 байта (слово), - DD (define double word) — выделяет 4 байта (двойное слово), - DQ (define quad word) — выделяет 8 байт (четверное слово), - DT (define ten bytes) — выделяет 10 байт. Эти директивы позволяют задавать простые переменные и массивы. Для строковых данных обычно используют директиву DB, поскольку она оптимально подходит для их хранения в оперативной памяти. Инструкция mov используется для копирования данных из источника в приёмник: mov dst, src Здесь dst — приёмник, а src — источник. Операндами могут быть регистры, ячейки памяти или непосредственные значения. Инструкция intслужит для вызова прерывания: int n Здесь n — номер прерывания в диапазоне от 0 до 255. В Linux для вызовов ядра (sys\_calls) обычно используется n = 80h (в шестнадцатеричном формате).

# 4 Выполнение лабораторной работы

**4.1 Основы работы с Midnight Commander** Открываю Midnight Commander, введя в терминал mc (рис. 1).

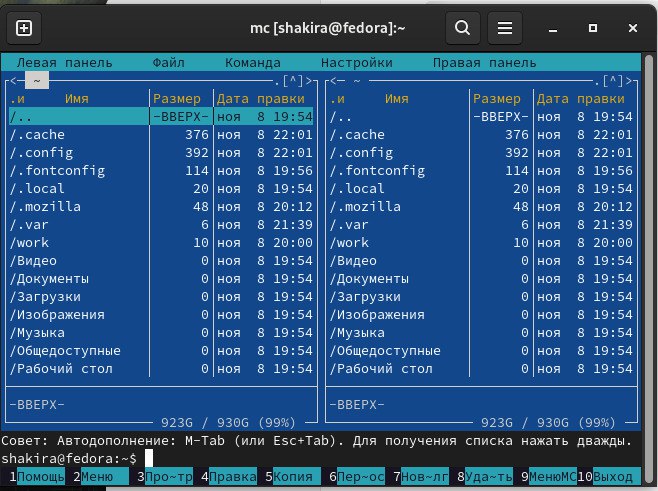
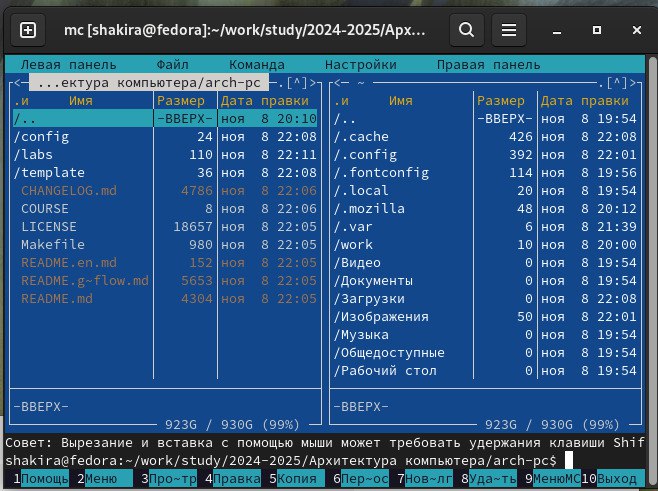
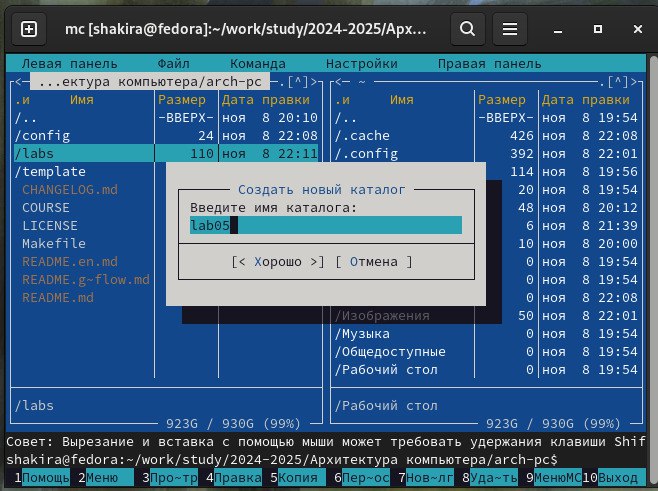


Рис. 1: 1

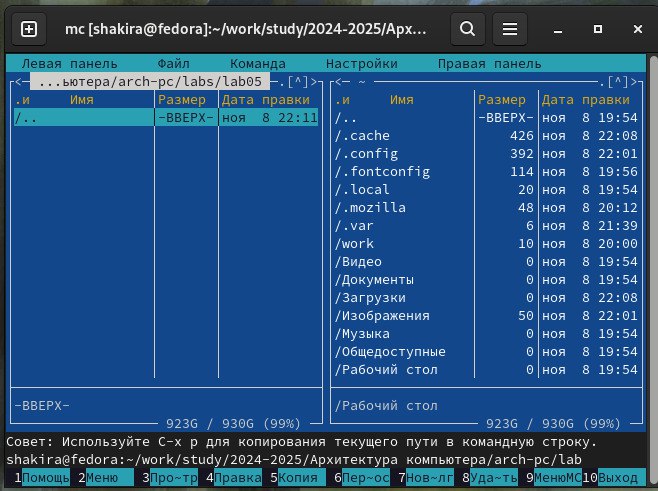
Перехожу в каталог ~/work/study/2024-2025/Архитектура Компьютера/ arch-pc, используя файловый менеджер mc (рис. 2).

Рис. 2: 2

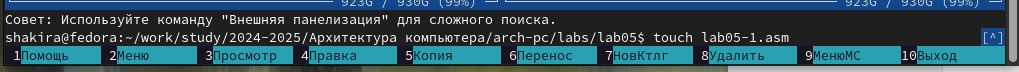
С помощью функциональной клавиши F7 создаю каталог lab05 (рис. 3).

Рис. 3: 3

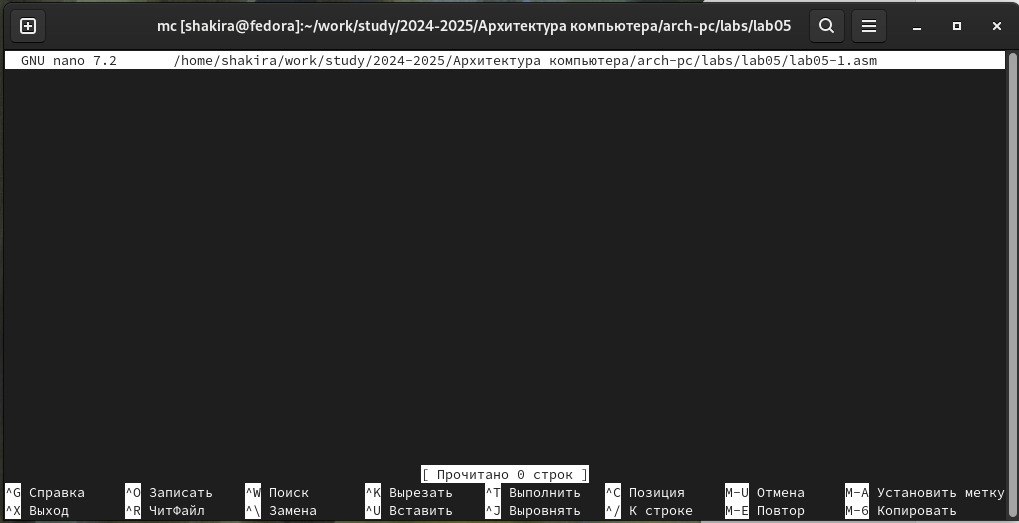
Перехожу в созданный каталог (рис. 4).

Рис. 4: 4

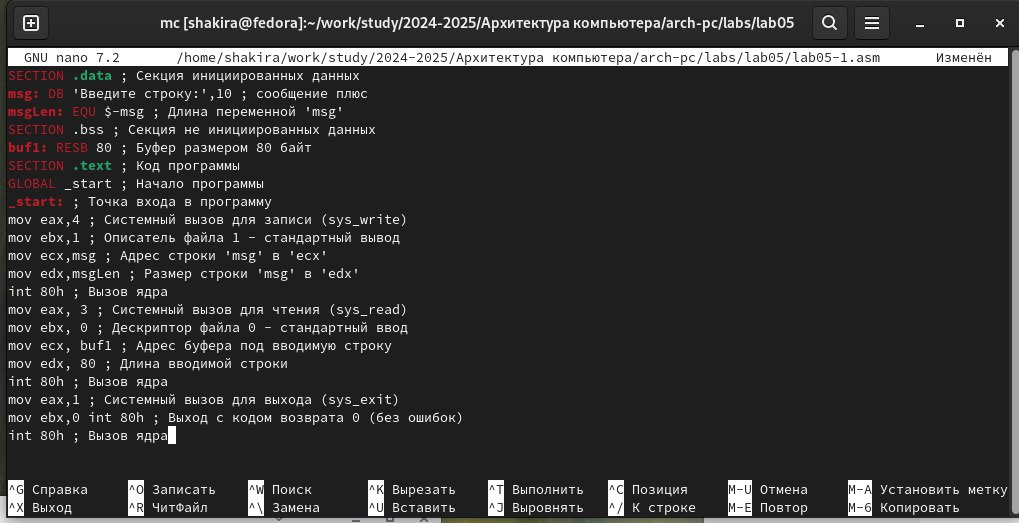
В строке ввода прописываю команду touch lab05-1.asm, чтобы создать файл, в котором буду работать (рис. 5).

Рис. 5: 5

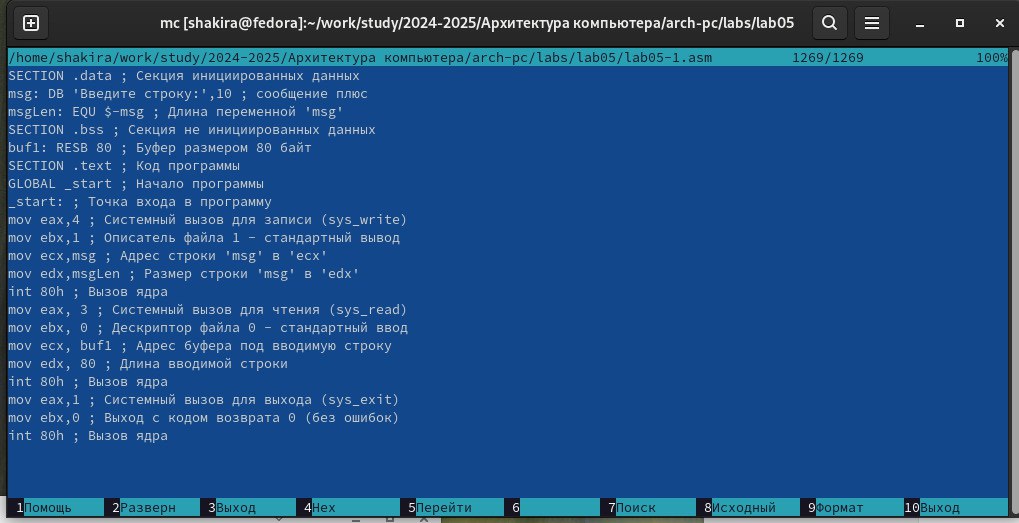
**4.2 Структура программы на языке ассебмлера NASM** С помощью функциональной клавиши F4 открываю созданный файл для редактирования в редакторе nano (рис. 6).

Рис. 6: 6

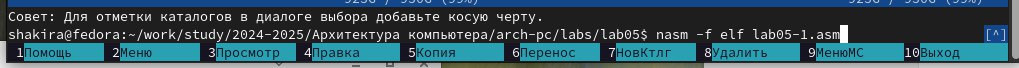
Ввожу в файл код программы для запроса строки у пользователя. Далее выхожу из файла (Ctrl+X), сохраняя изменения (Y, Enter) (рис. 7).

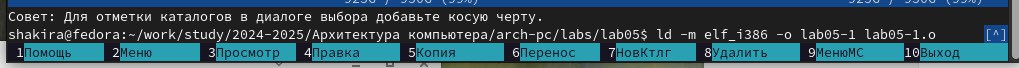
Рис. 7: 7

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

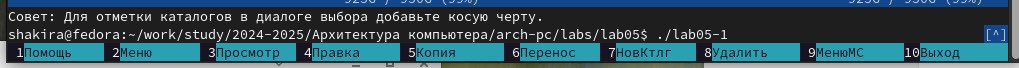
Рис. 8: 8

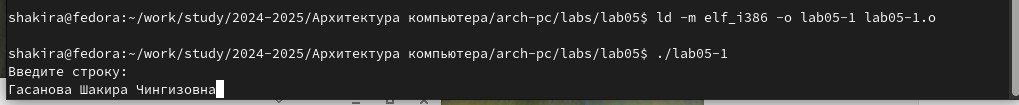
Транслирую текст программы файла в объектный файл командой nasm -f elf lab05-1.asm (рис. **¿fig:009?**).. Создался объектный файл lab05-1.o. Выполняю компоновку объектного файла с помощью команды ld -m elf\_i386 -o lab05-1 lab05-1.o. Создался исполняемый файл lab05-1 (рис. **¿fig:010?**).

Рис. 9: 9

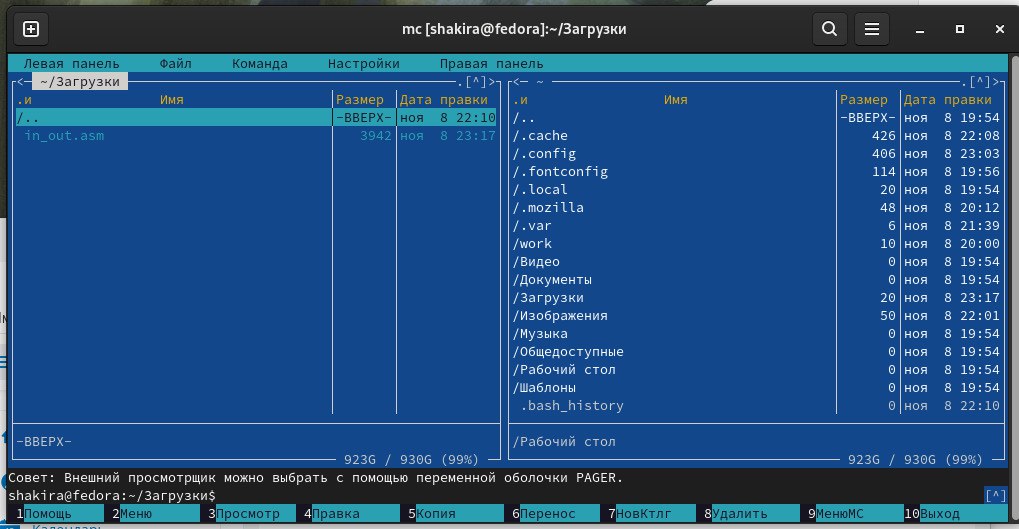
Рис. 10: 10

Запускаю исполняемый файл. Программа выводит строку “Введите строку:” и ждет ввода с клавиатуры, я ввожу свои ФИО, на этом программа заканчивает свою работу (рис. 11, 12).

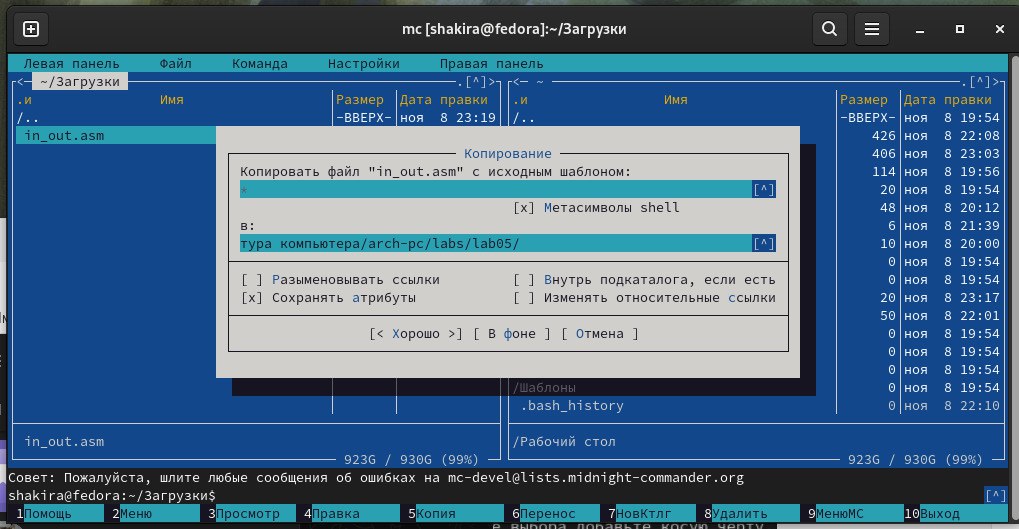
Рис. 11: 11

Рис. 12: 12

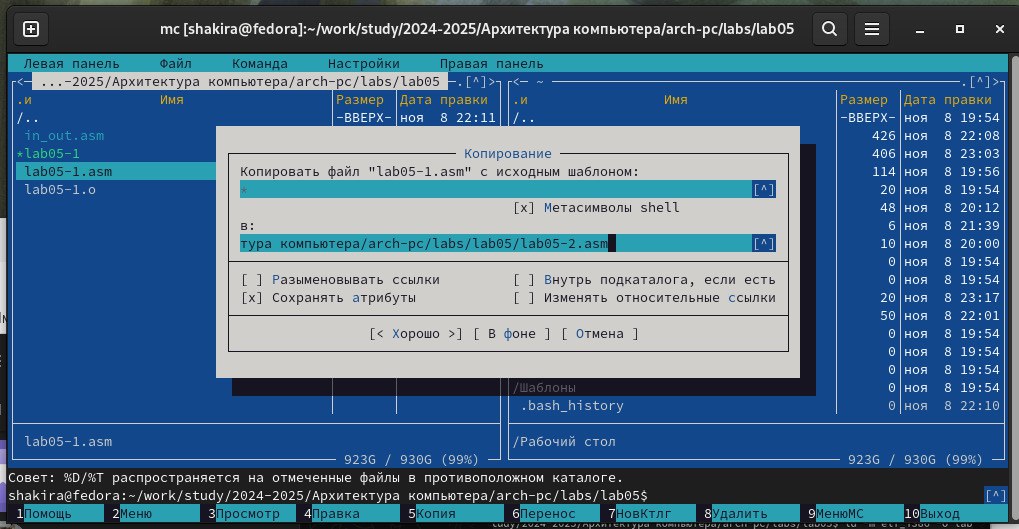
**4.3 Подключение внешнего файла** Скачиваю файл in\_out.asm со страницы курса в ТУИС. Он сохранился в каталог “Загрузки” (рис. 13).

Рис. 13: 13

С помощью функциональной клавиши F5 копирую файл in\_out.asm из каталога Загрузки в созданный каталог lab05 (рис. 14).

Рис. 14: 14

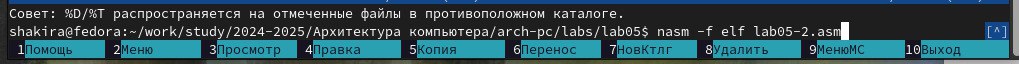
С помощью функциональной клавиши F5 копирую файл lab05-1.asm в тот же каталог, но с другим именем - lab05-2.asm, для этого в появившемся окне mc прописываю имя для копии файла (рис. 15).

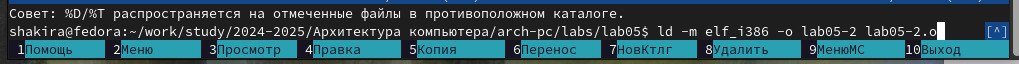
Рис. 15: 15

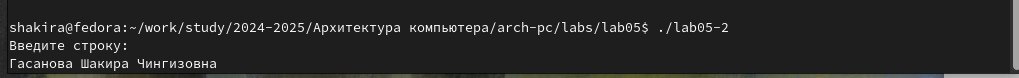
Изменяю содержимое файла lab05-2.asm во встроенном редакторе nano, чтобы в программе использовались подпрограммы из внешнего файла in\_out.asm (рис. 16).

Рис. 16: 16

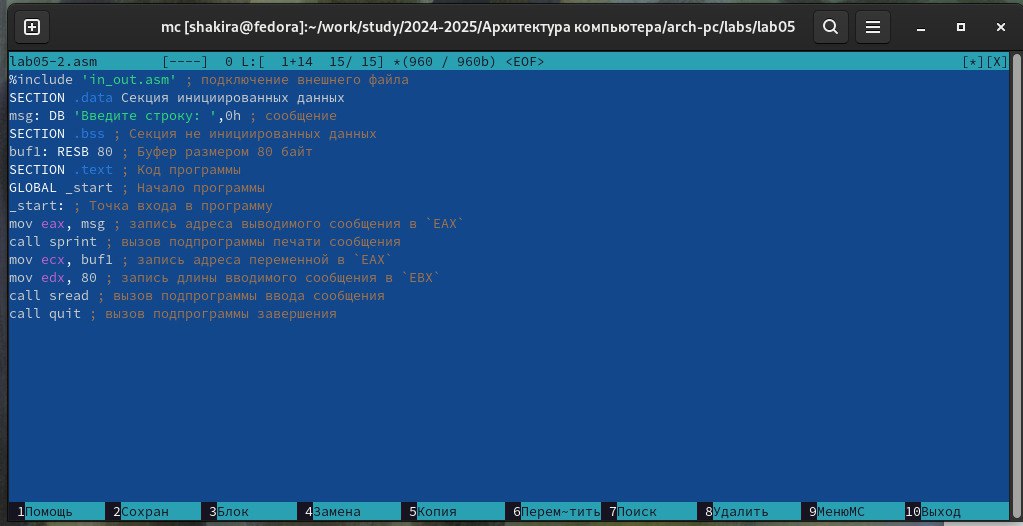
Транслирую текст программы файла в объектный файл командой nasm -f elf lab05-2.asm (рис. 17). Создался объектный файл lab05-2.o. Выполняю компоновку объектного файла с помощью команды ld -m elf\_i386 -o lab05-2 lab05-2.o Создался исполняемый файл lab05-2 (рис. 18). Запускаю исполняемый файл (рис.19).

Рис. 17: 17

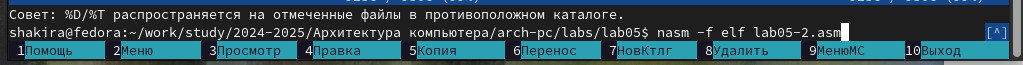
Рис. 18: 18

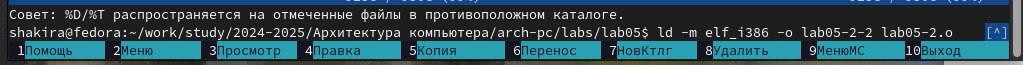
Рис. 19: 19

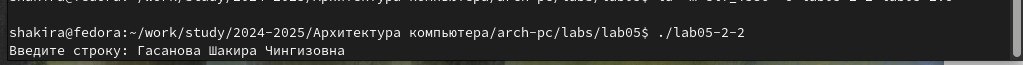
Открываю файл lab05-2.asm для редактирования в nano функциональной клавишей F4. Изменяю в нем подпрограмму sprintLF на sprint. Сохраняю изменения и открываю файл для просмотра, чтобы проверить сохранение действий (рис. 20).

Рис. 20: 20

Снова транслирую файл, выполняю компоновку созданного объектного файла, запускаю новый исполняемый файл (рис. 21, 22, 23).

Рис. 21: 21

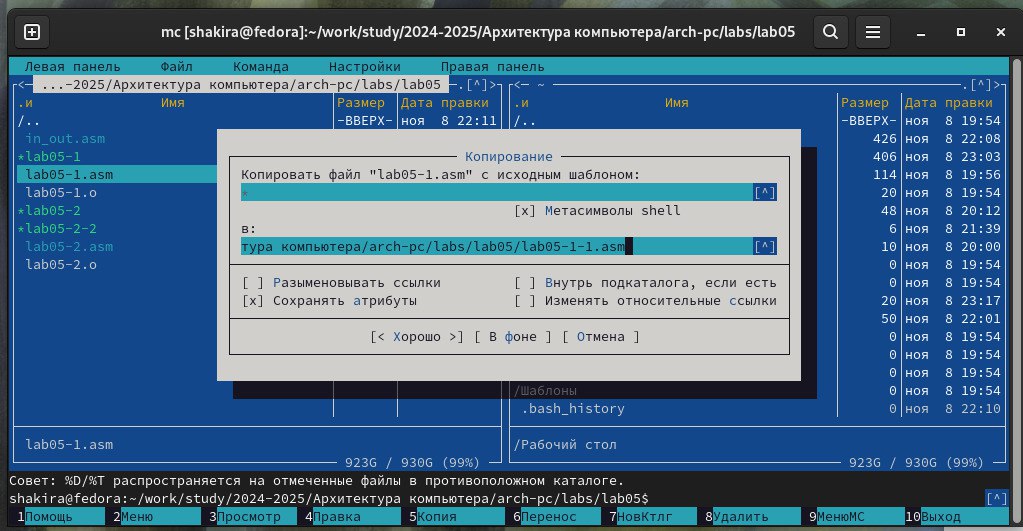
Рис. 22: 22

Рис. 23: 23

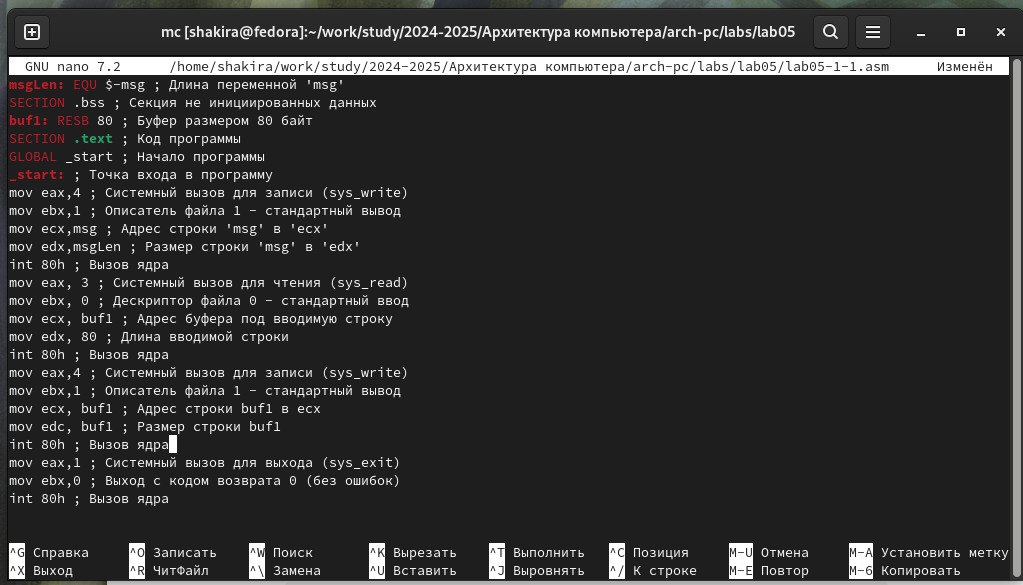
Разница между первым исполняемым файлом lab05-2 и вторым lab05-2-2 заключается в том, что при запуске первого программа запрашивает ввод с новой строки, тогда как при запуске второго - без переноса на новую строку. Это различие обусловлено использованием подпрограмм: в первом случае применяется sprintLF, а во втором - sprint.

**4.4 Выполнение заданий для самостоятельной работы**

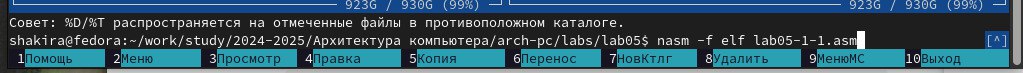
1. Создаю копию файла lab05-1.asm с именем lab05-1-1.asm с помощью функциональной клавиши F5 (рис. 24).

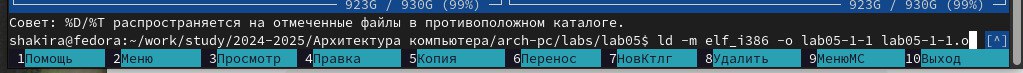
Рис. 24: 24

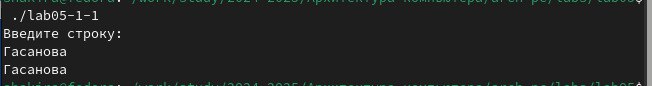
С помощью функциональной клавиши F4 открываю созданный файл для редактирования. Изменяю программу так, чтобы кроме вывода приглашения и запроса ввода, она выводила вводимую пользователем строку (рис. 25).

Рис. 25: 25

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 свою фамилию, далее программа выводит введенные мною данные (рис. *26, 27, 28*).

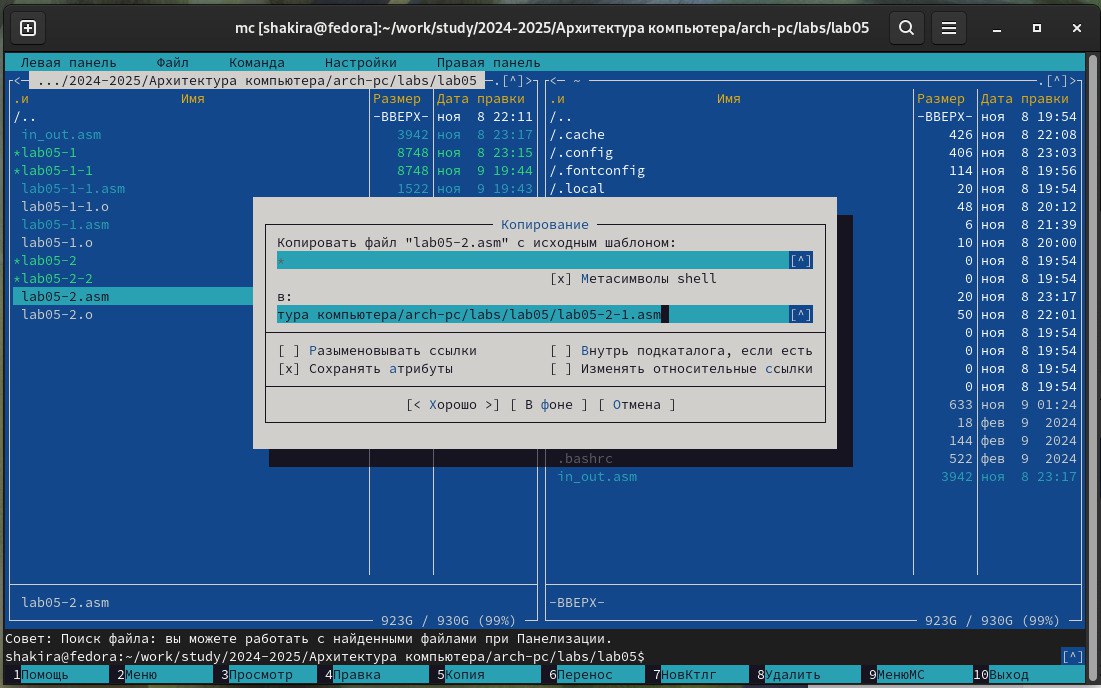
Рис. 26: 26

Рис. 27: 27

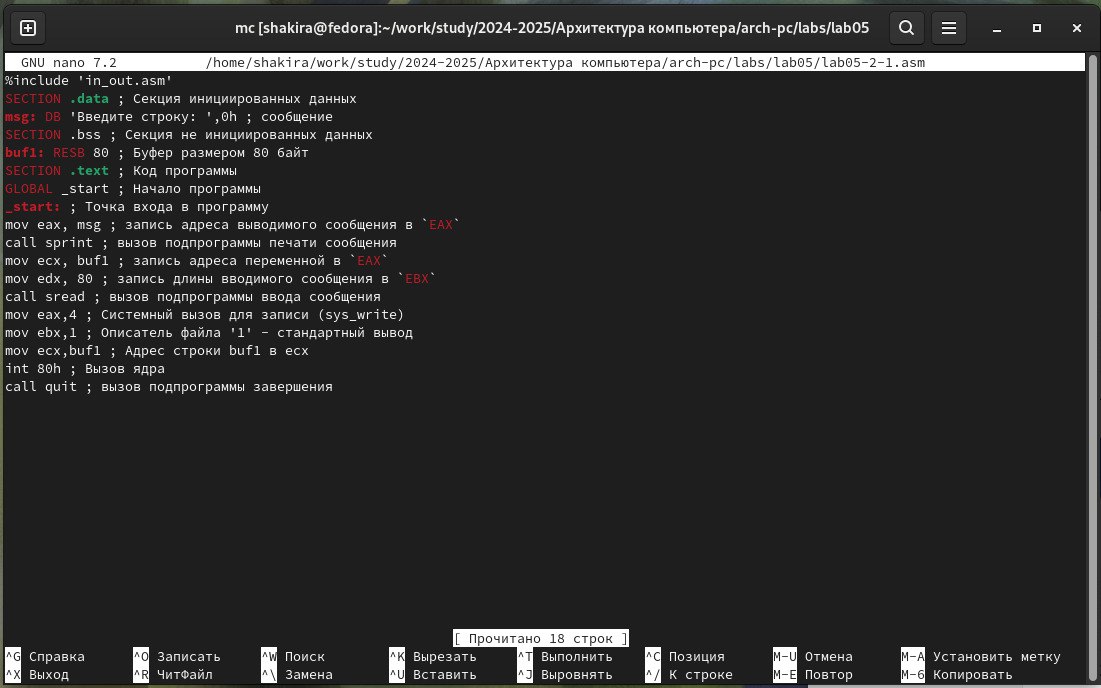
Рис. 28: 28

Код программы: SECTION .data ; Секция инициированных данных msg: DB ‘Введите строку:’,10 ; сообщение плюс msgLen: EQU $-msg ; Длина переменной ‘msg’ SECTION .bss ; Секция не инициированных данных 16buf1: 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) mov ebx,0 ; Выход с кодом возврата 0 (без ошибок) int 80h ; Вызов ядра

3. Создаю копию файла lab05-2.asm с именем lab05-2-1.asm с помощью функциональной клавиши F5 (рис. 29).

Рис. 29: 29

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. 30).

Рис. 30: 30

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 свою фамилию, далее программа выводит введенные мною данные (рис. 31, 32, 33).

Рис. 31: 31

Рис. 32: 32

Рис. 33: 33

Код программы: %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 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

# 5 Выводы

При выполнении данной лабораторной работы я приобрела практические навыки работы в Midnight Commander, а также освоила инструкции языка ассемблера mov и int

# 6 Список литературы

1. [Архитектура ЭВМ (rudn.ru)](https://esystem.rudn.ru/)