

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: Жарикова Таисия Александровна

Группа: НКАбд-05-24

МОСКВА

2024 г.

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.....	10
4.3.Подключение внешнего файла	15
4.4.Выполнение заданий для самостоятельной работы.....	20
5.Выводы.....	29
6.Источники	30

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 intslужит 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).

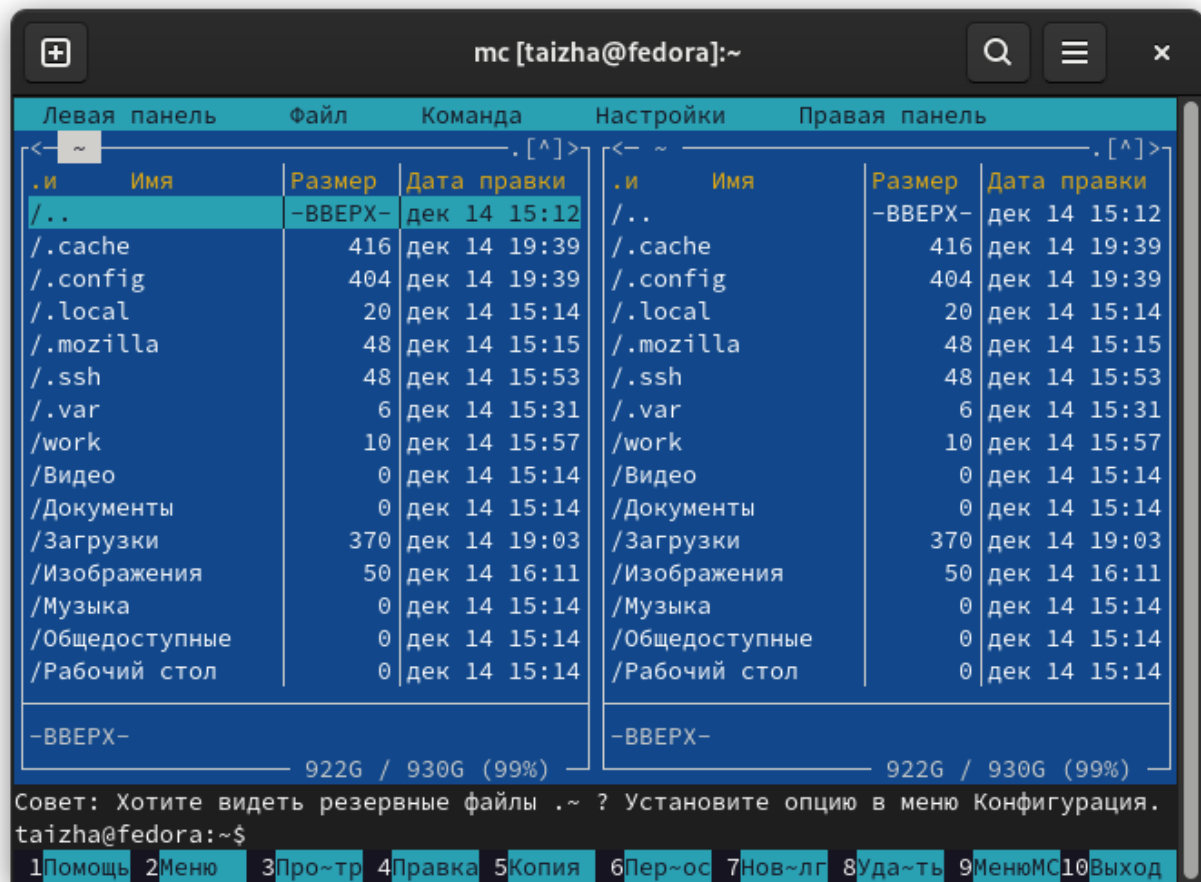


Рис.1 Открытый mc

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

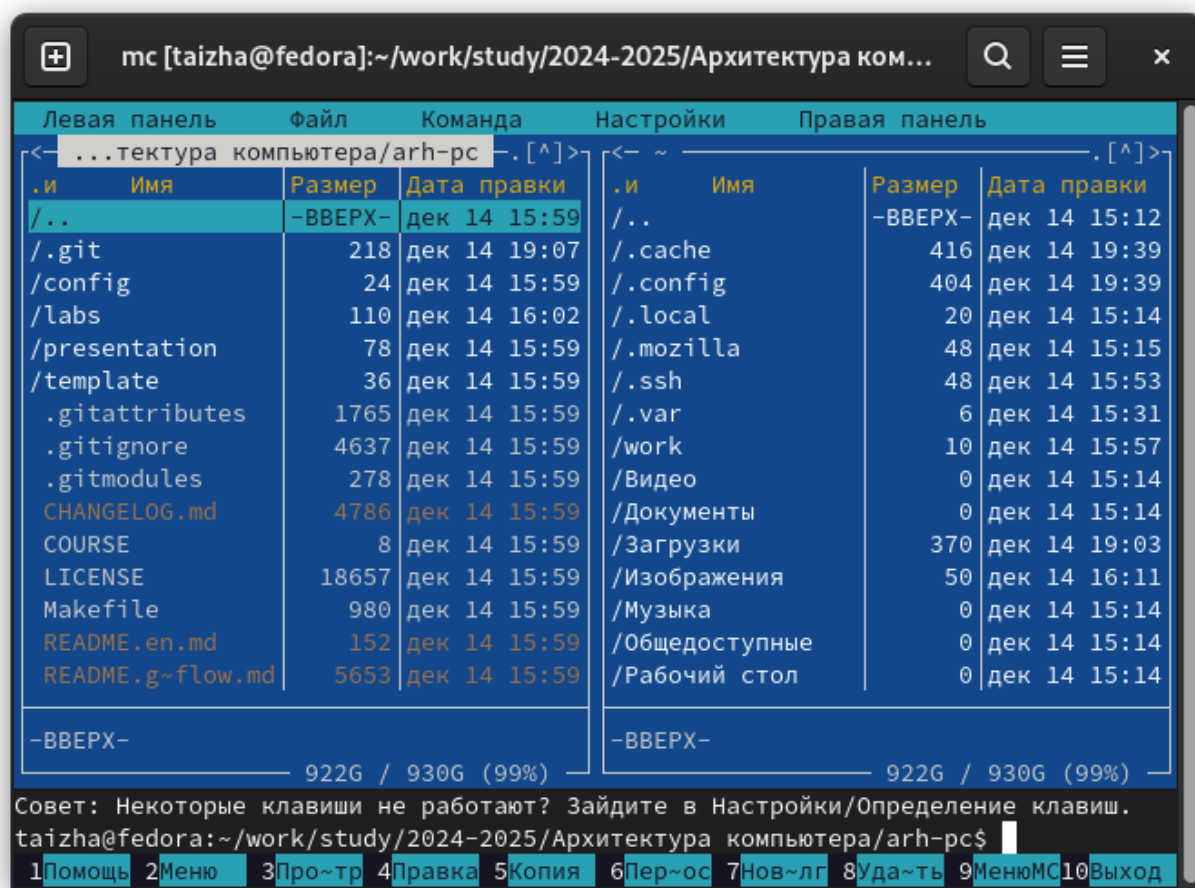


Рис.2 Перемещение между директориями

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

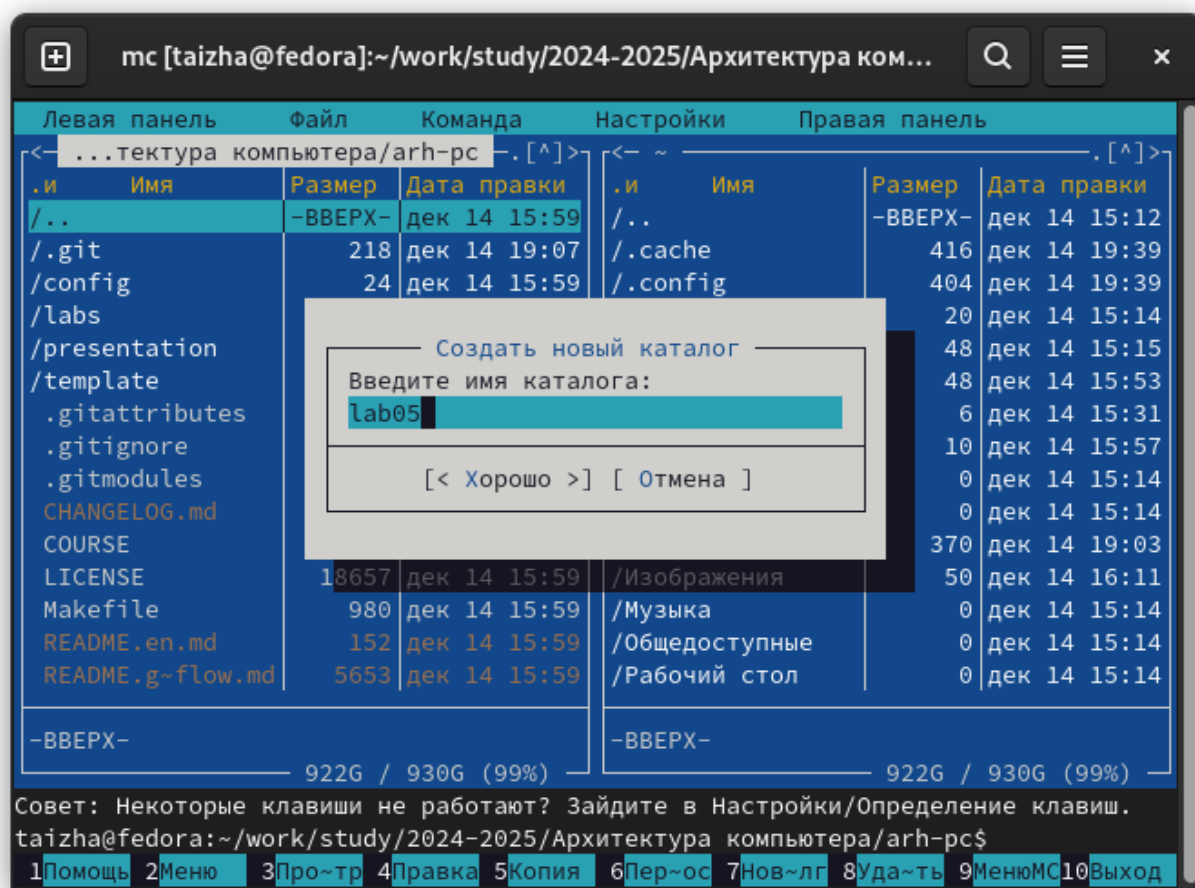


Рис.3 Создание каталога

Перехожу в созданный каталог (рис. 4).

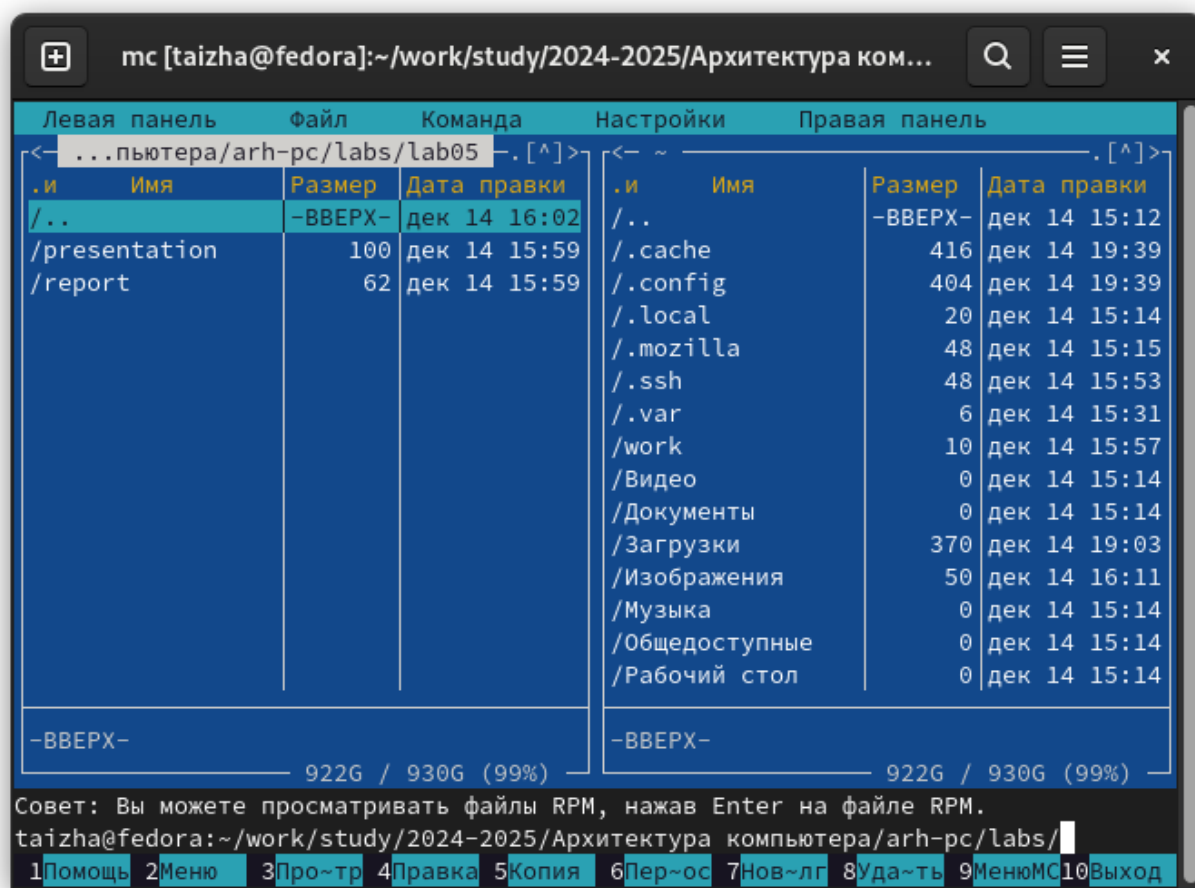


Рис.4 Перемещение между директориями

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

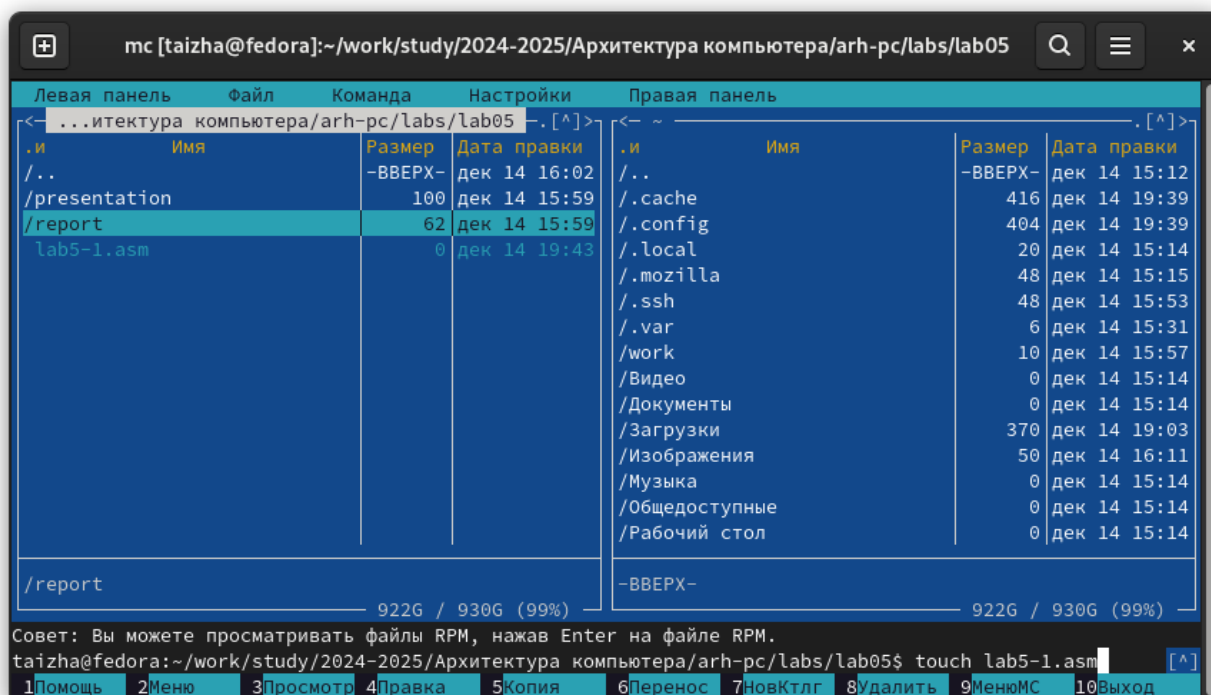


Рис.5 Создание файла

4.2. Структура программы на языке ассемблера NASM

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

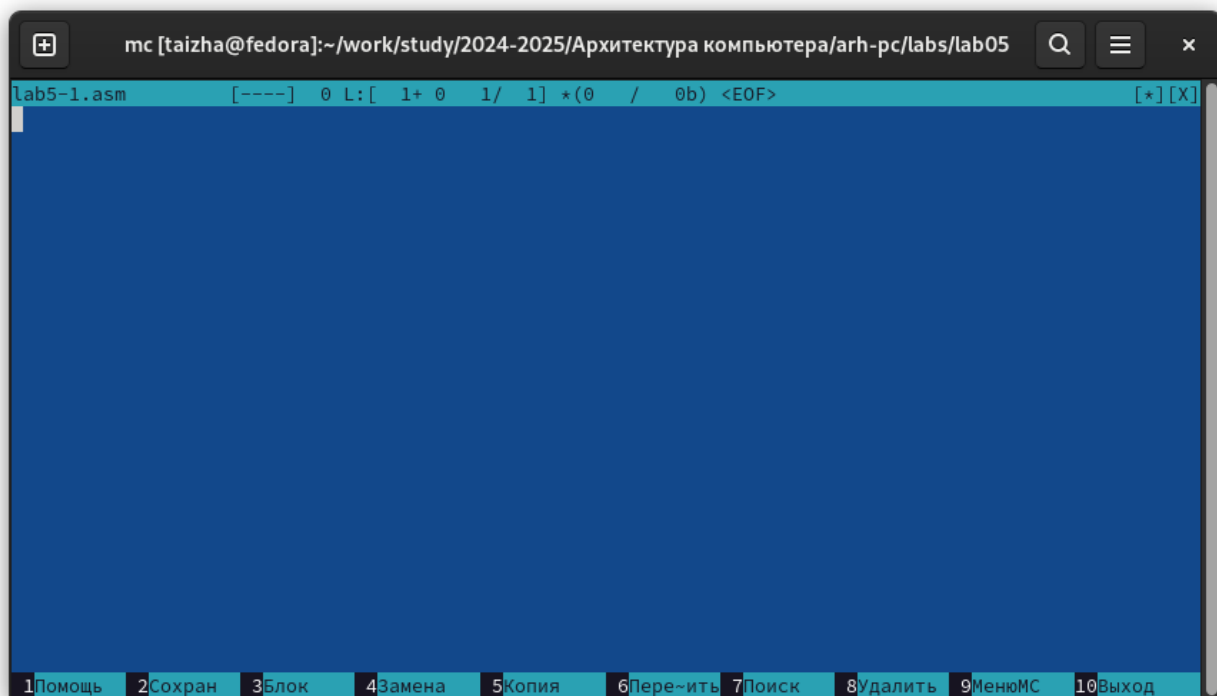


Рис.6 Открытие файла для редактирования

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

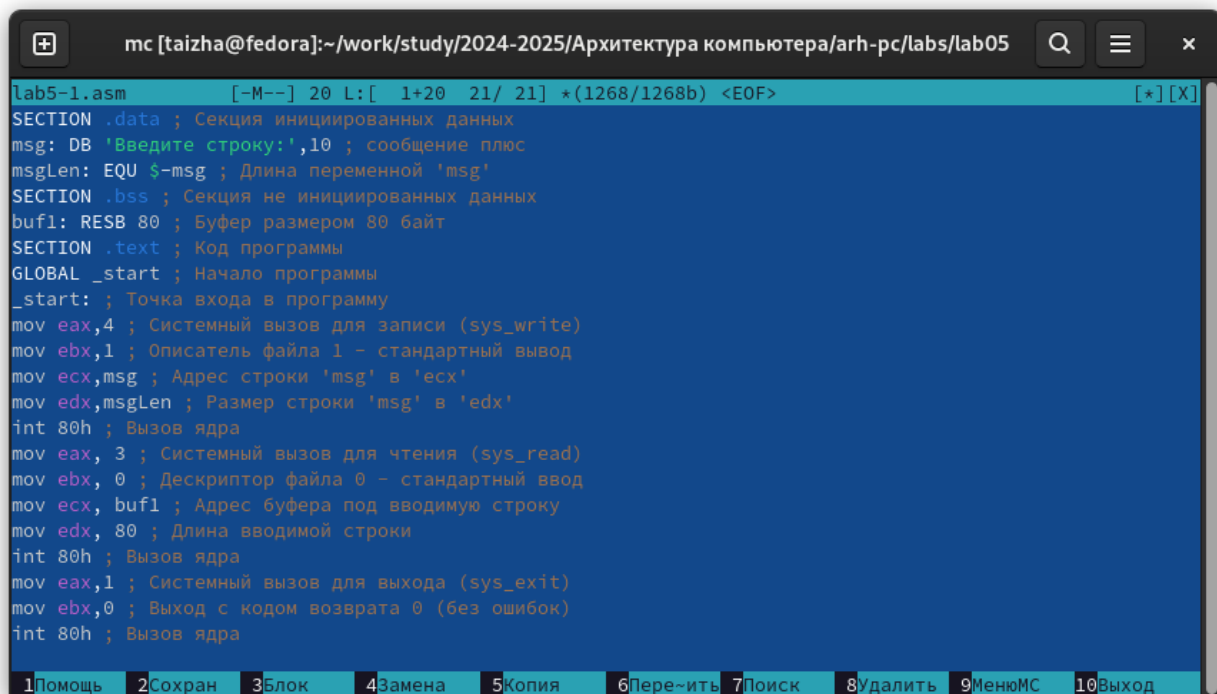


Рис.7 Редактирование файла

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

```

/home/taizha/work/study/2024-2025/Архитектура компьютера/arh-pc/labs/lab05/lab5-1.asm 1268/1268 100%
SECTION .data ; Секция инициализированных данных
msg: DB 'Введите строку:',10 ; сообщение плюс
msgLen: EQU $-msg ; Длина переменной 'msg'
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,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 ;Descriptor файла 0 - стандартный ввод
mov ecx, buf1 ; Адрес буфера под вводимую строку
mov edx, 80 ; Длина вводимой строки
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
1Помощь 2Разверн 3Выход 4Нех 5Перейти 6 7Поиск 8Исходный 9Формат 10Выход

```

Рис.8 Открытие файла для просмотра

Транслирую текст программы файла в объектный файл командой `nasm -f elf lab05-1.asm` (рис. 9). 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. 10).

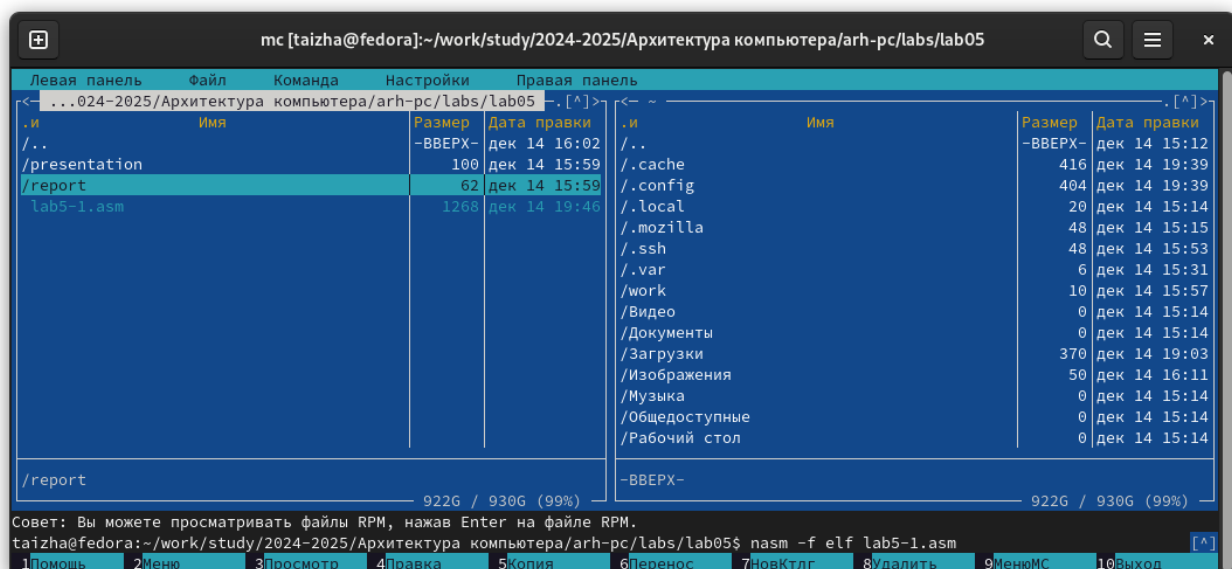


Рис.9 Компиляция файла

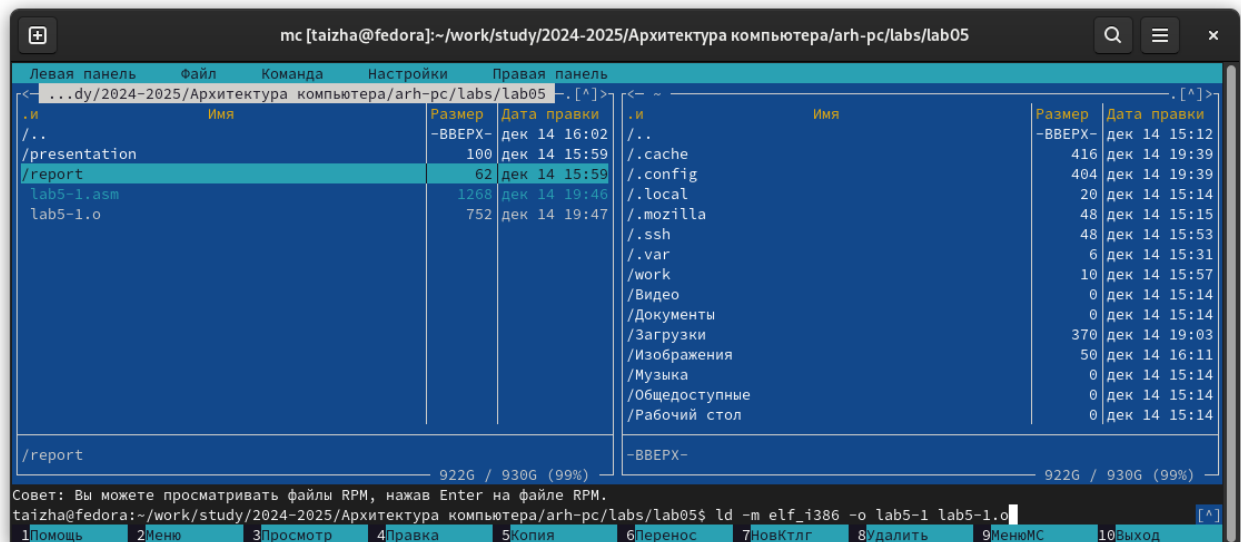


Рис.10 Компиляция файла

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).

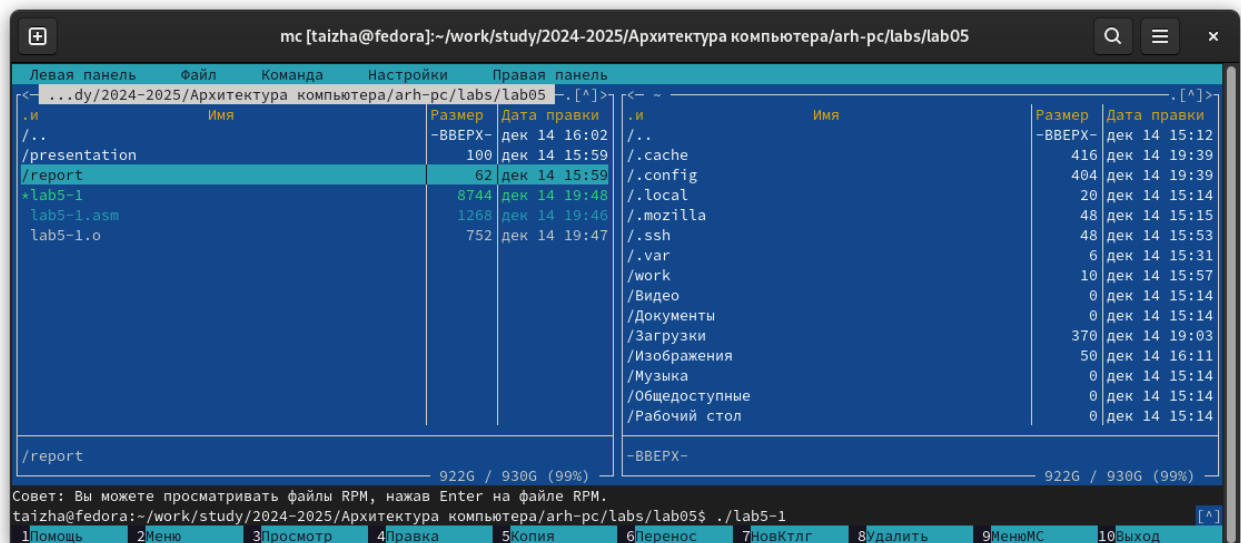
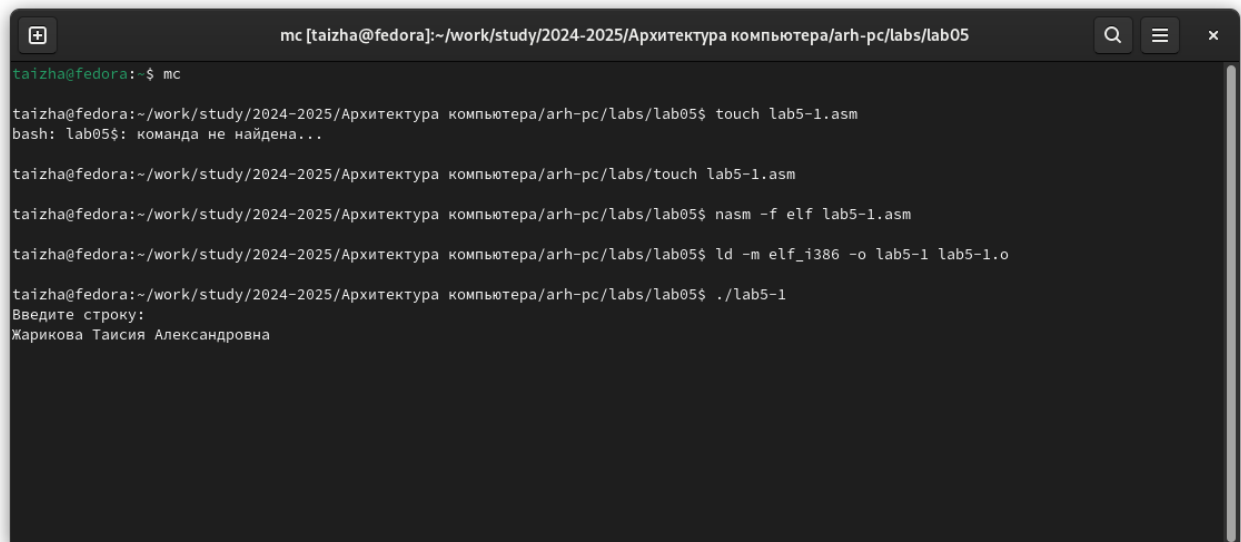


Рис.11 Исполнение файла



A terminal window titled "mc [taizha@fedora]: ~/work/study/2024-2025/Архитектура компьютера/arh-pc/labs/lab05". The window shows a series of commands and their outputs in a dark theme. The commands include creating a file with 'touch', assembling with 'nasm', linking with 'ld', and running the resulting binary with './lab5-1'. The output of the last command is a prompt to enter a line of text, followed by the name "Жарикова Таисия Александровна".

```
mc [taizha@fedora]: ~/work/study/2024-2025/Архитектура компьютера/arh-pc/labs/lab05
taizha@fedora:~$ mc
taizha@fedora:~/work/study/2024-2025/Архитектура компьютера/arh-pc/labs/lab05$ touch lab5-1.asm
bash: lab05$: команда не найдена...
taizha@fedora:~/work/study/2024-2025/Архитектура компьютера/arh-pc/labs/lab05$ touch lab5-1.asm
taizha@fedora:~/work/study/2024-2025/Архитектура компьютера/arh-pc/labs/lab05$ nasm -f elf lab5-1.asm
taizha@fedora:~/work/study/2024-2025/Архитектура компьютера/arh-pc/labs/lab05$ ld -m elf_i386 -o lab5-1 lab5-1.o
taizha@fedora:~/work/study/2024-2025/Архитектура компьютера/arh-pc/labs/lab05$ ./lab5-1
Введите строку:
Жарикова Таисия Александровна
```

Рис.12 Исполнение файла

4.3. Подключение внешнего файла

Скачиваю файл `in_out.asm` со страницы курса в ТУИС. Он сохранился в каталог “Загрузки” (рис. 13).

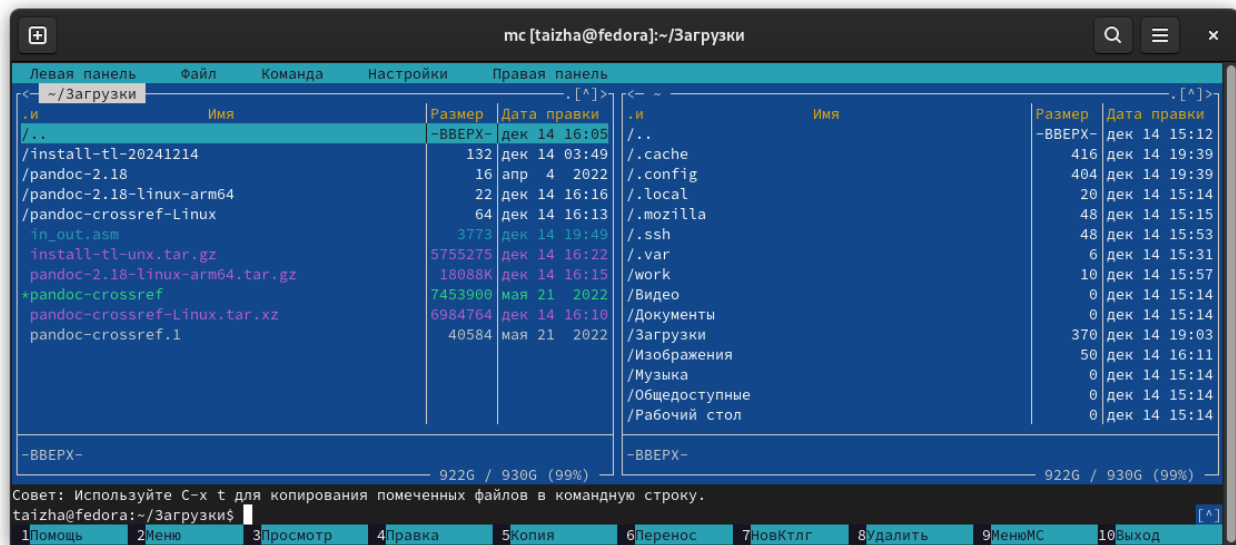


Рис.13 Скачанный файл

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

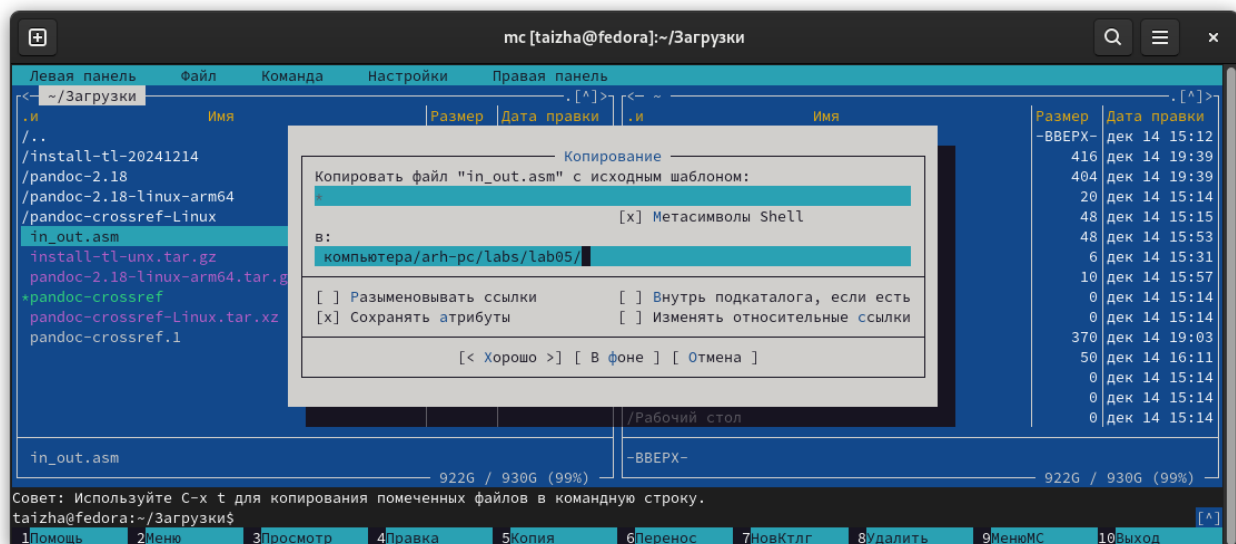


Рис.14 Копирование файла

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).

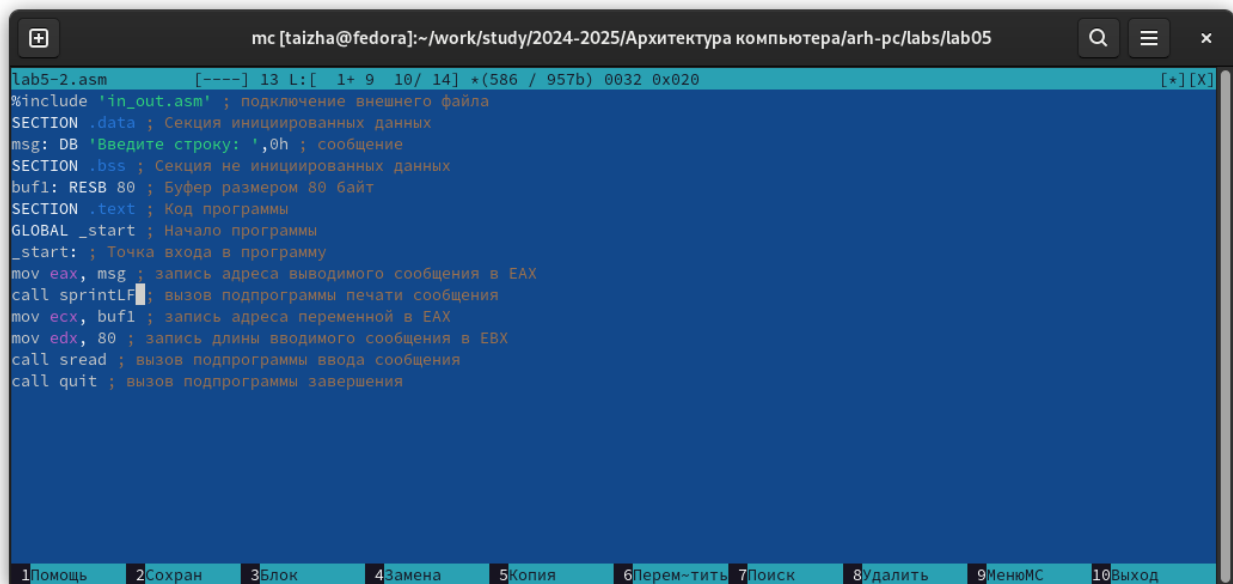


Рис.15 Копирование файла

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

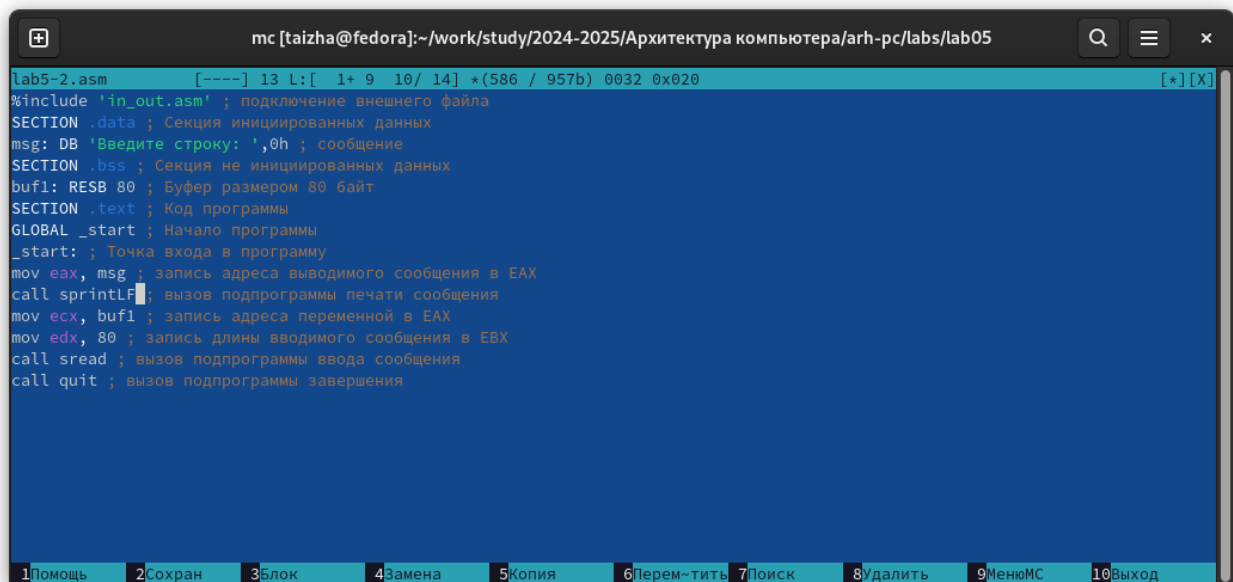


Рис.16 Редактирование файла

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).

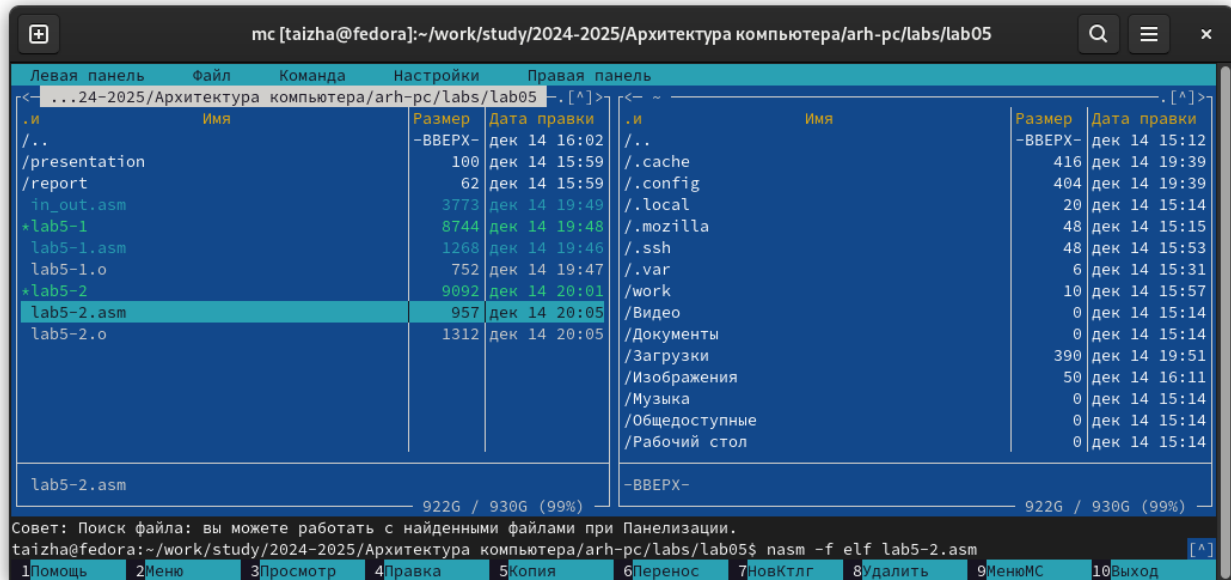


Рис.17 Трансляция файла

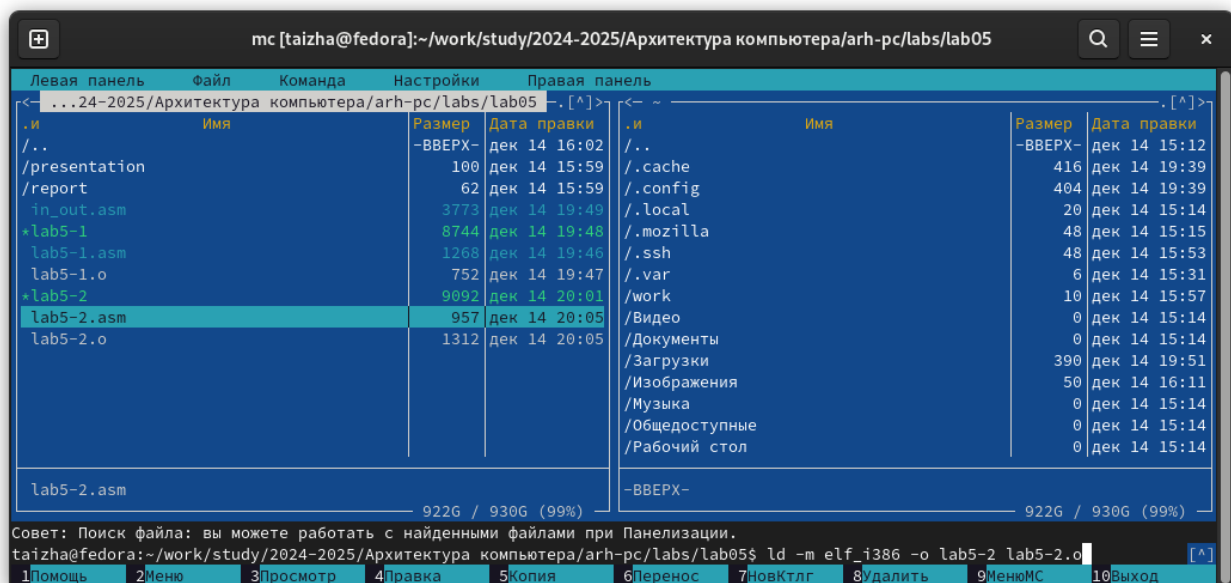
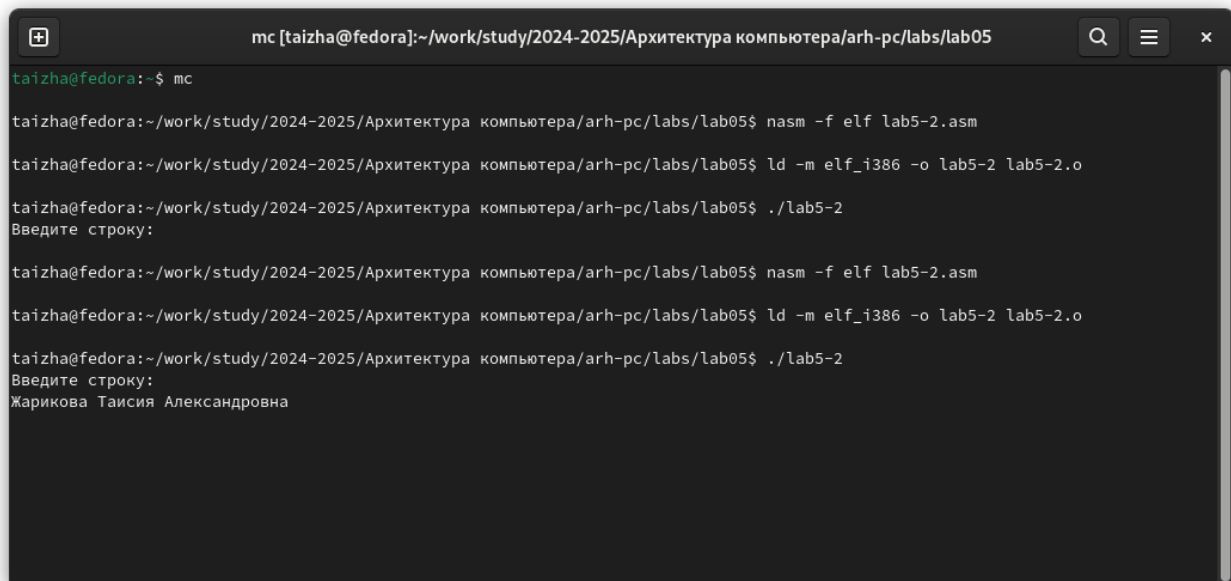


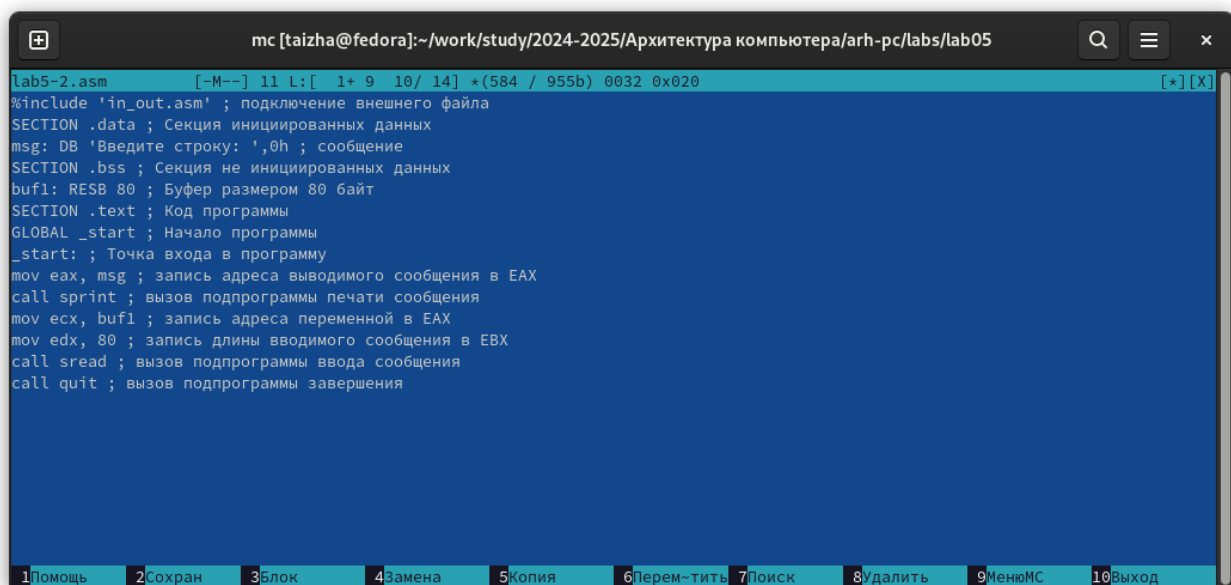
Рис.18 Компоновка файла



```
mc [taizha@fedora]:~/work/study/2024-2025/Архитектура компьютера/arh-pc/labs/lab05
taizha@fedora:~$ mc
taizha@fedora:~/work/study/2024-2025/Архитектура компьютера/arh-pc/labs/lab05$ nasm -f elf lab5-2.asm
taizha@fedora:~/work/study/2024-2025/Архитектура компьютера/arh-pc/labs/lab05$ ld -m elf_i386 -o lab5-2 lab5-2.o
taizha@fedora:~/work/study/2024-2025/Архитектура компьютера/arh-pc/labs/lab05$ ./lab5-2
Введите строку:
taizha@fedora:~/work/study/2024-2025/Архитектура компьютера/arh-pc/labs/lab05$ nasm -f elf lab5-2.asm
taizha@fedora:~/work/study/2024-2025/Архитектура компьютера/arh-pc/labs/lab05$ ld -m elf_i386 -o lab5-2 lab5-2.o
taizha@fedora:~/work/study/2024-2025/Архитектура компьютера/arh-pc/labs/lab05$ ./lab5-2
Введите строку:
Жарикова Таисия Александровна
```

Рис.19 Исполнение файла

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).



```
lab5-2.asm [-M--] 11 L:[ 1+ 9 10/ 14] *(584 / 955b) 0032 0x020 [*] [X]
#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 вводимого сообщения в EBX
call sread ; вызов подпрограммы ввода сообщения
call quit ; вызов подпрограммы завершения

1Помощь 2Сохран 3Блок 4Замена 5Копия 6Перем-тить 7Поиск 8Удалить 9МенюМС 10Выход
```

Рис.20 Отредактированный файл

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

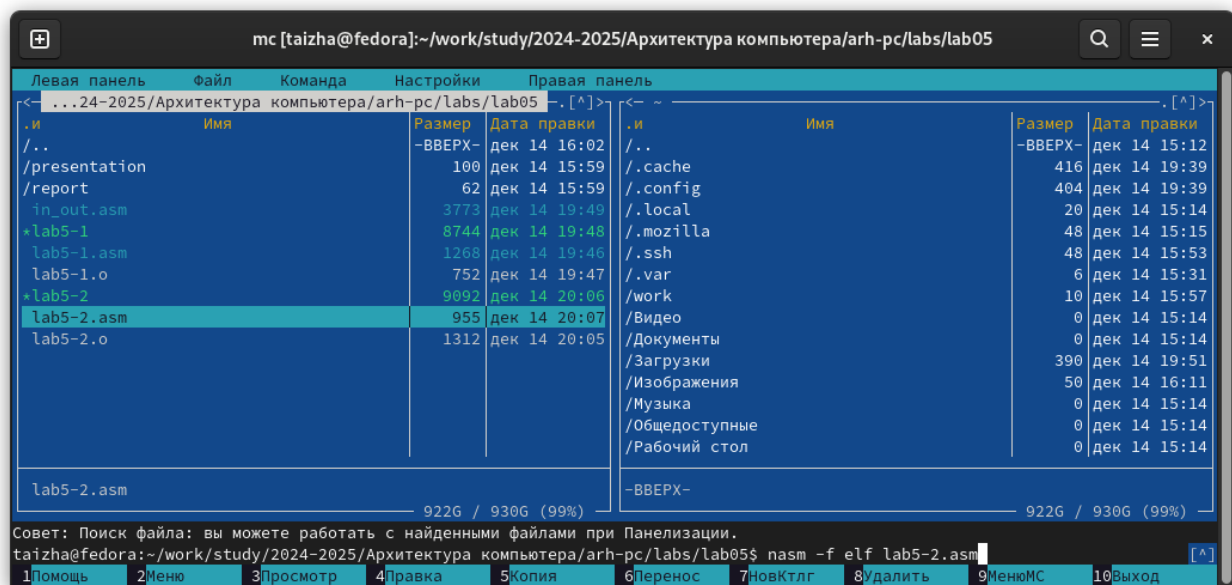


Рис.21 Трансляция файла

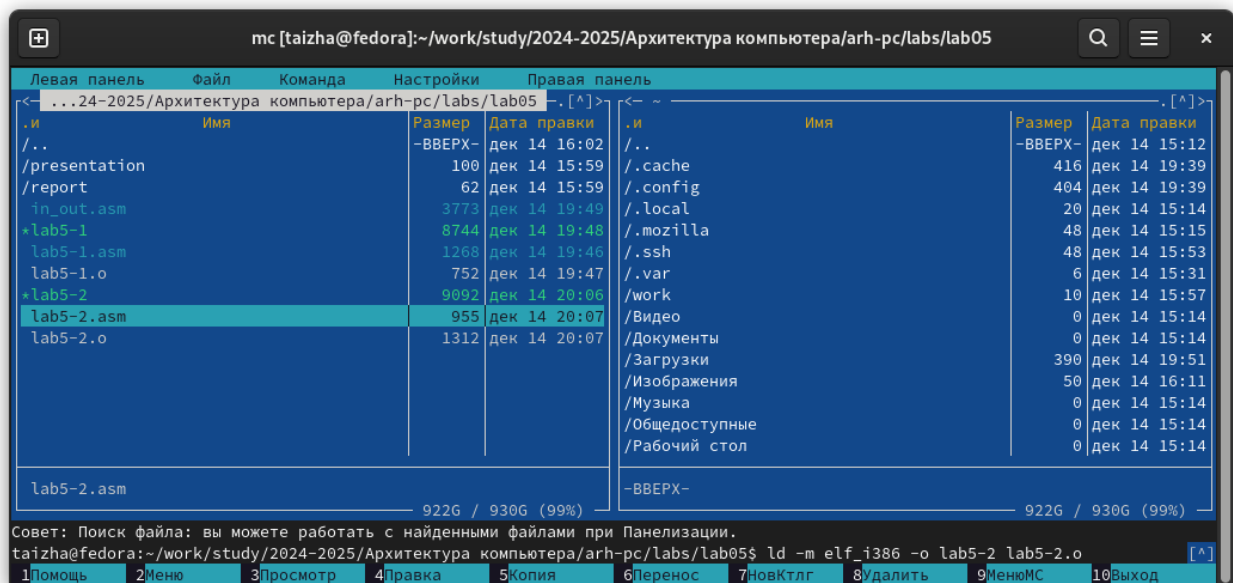


Рис.22 Компоновка файла

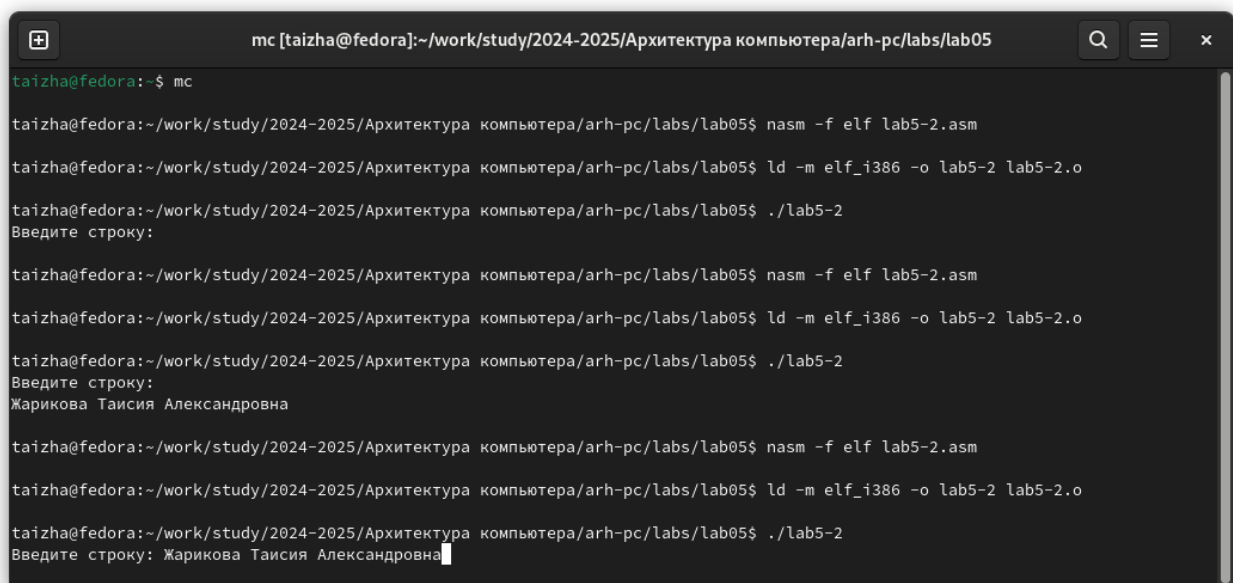


Рис.23 Исполнение файла

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 `sprintf`, а во втором - `sprintf`.

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).

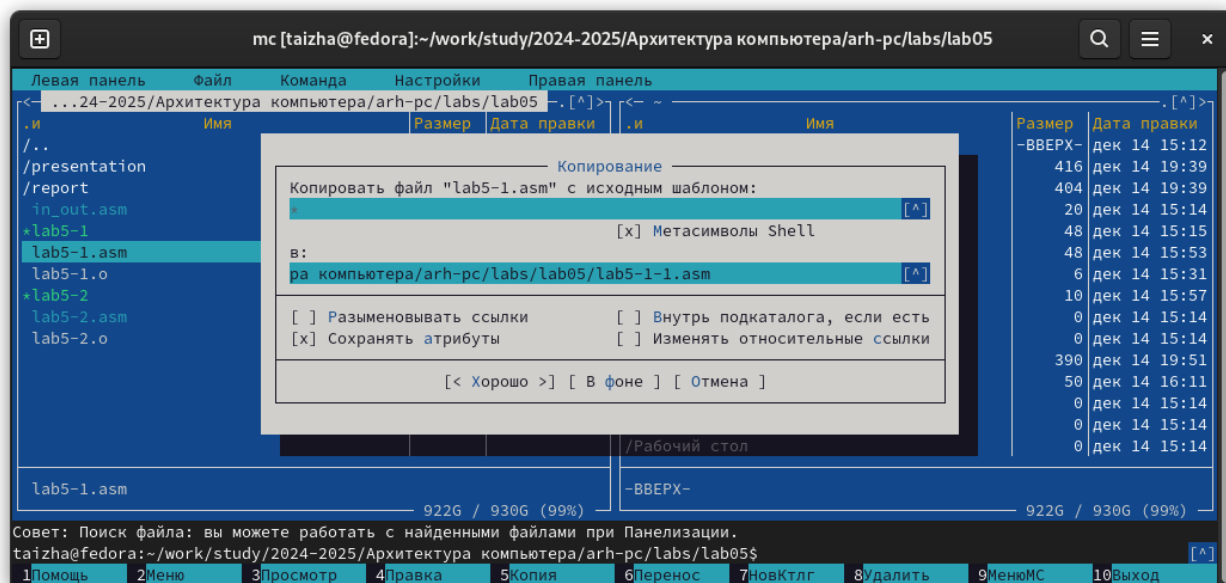


Рис.24 Копирование файла

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. 25).

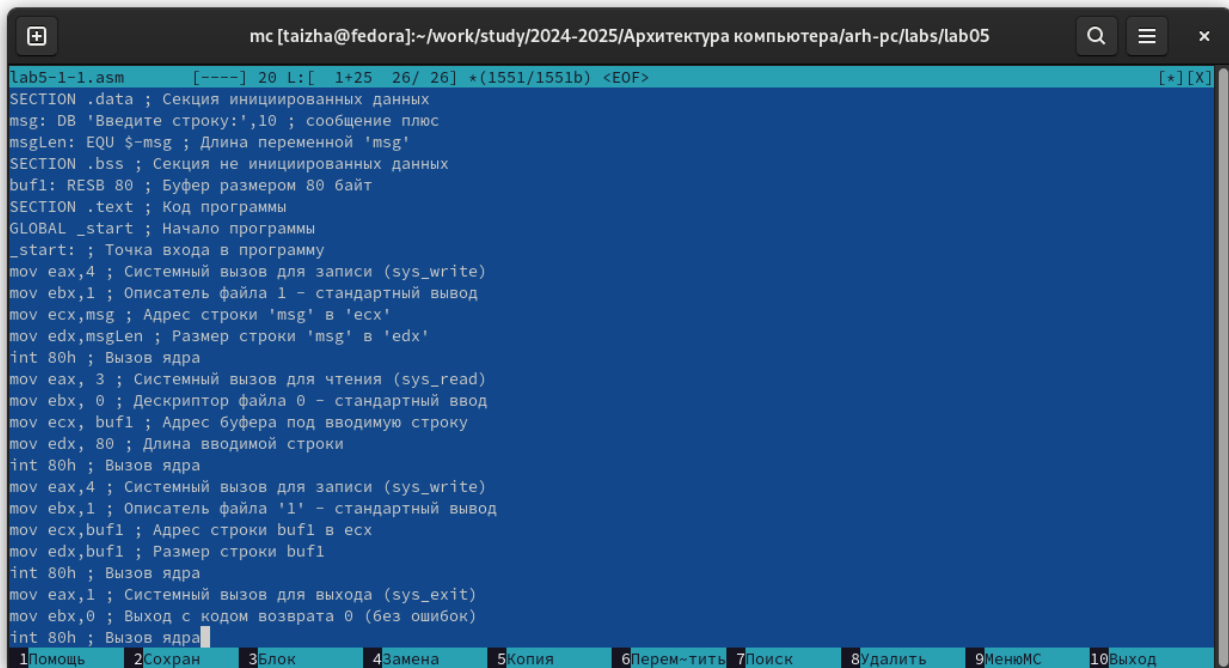


Рис.25 Редактирование файла

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

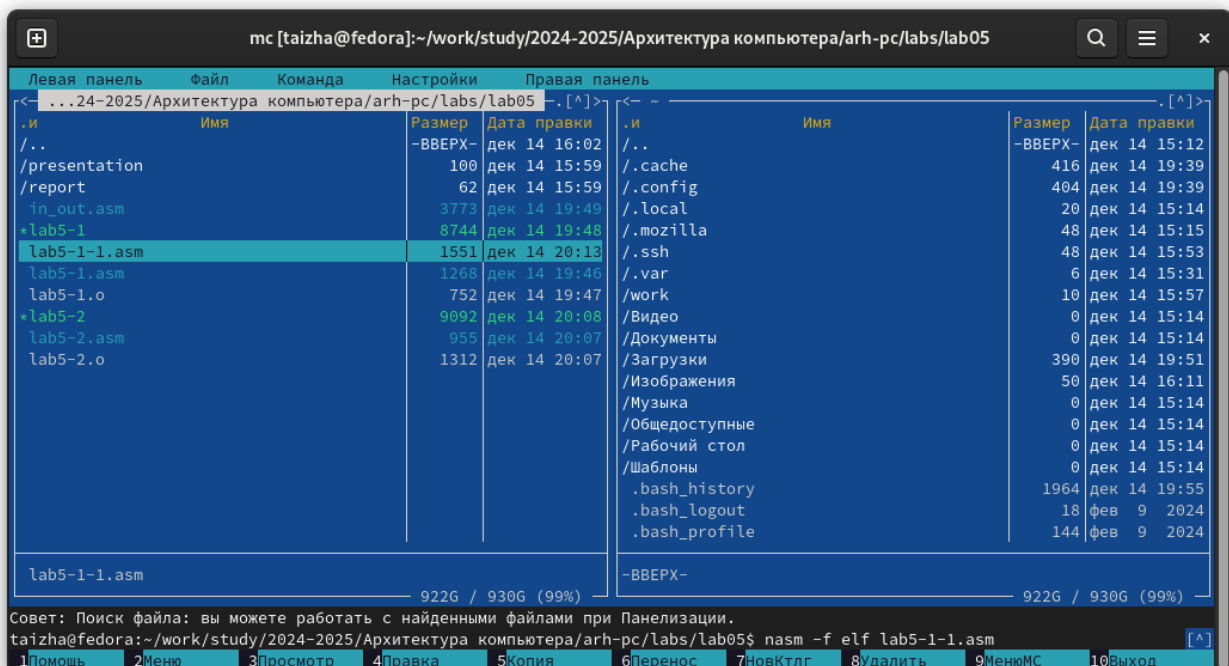


Рис.26 Трансляция файла

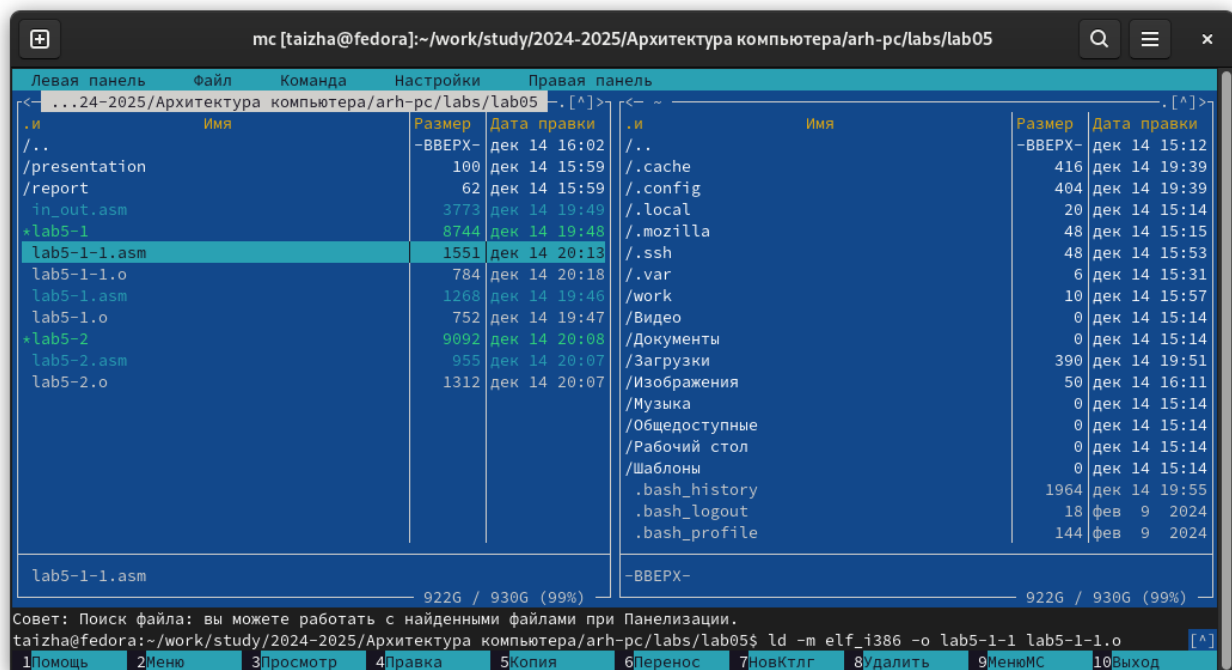


Рис.27 Компоновка файла



Рис.28 Исполнение файла

Код программы:

SECTION .data ; Секция иницированных данных
 msg: DB 'Введите строку:',10 ; сообщение плюс
 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)
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).

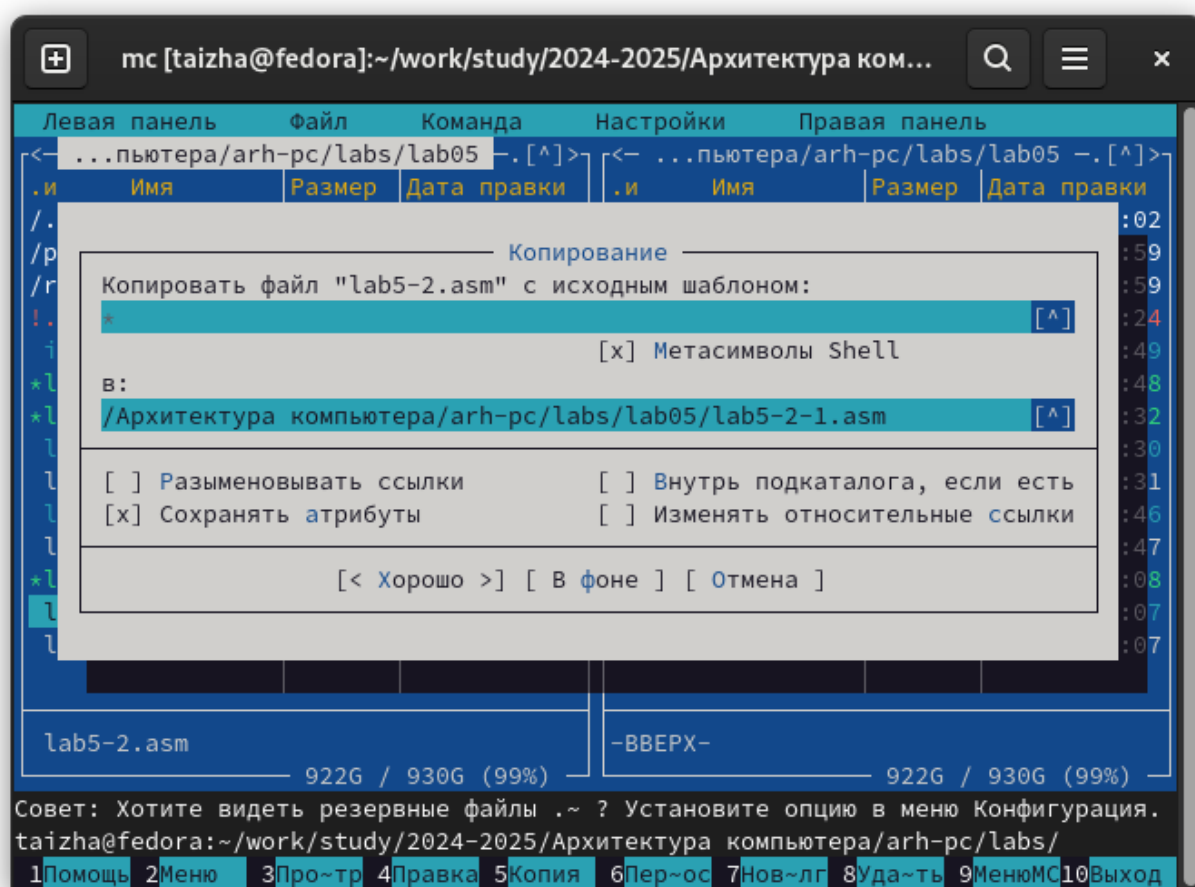


Рис.29 Копирование файла

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. 30).

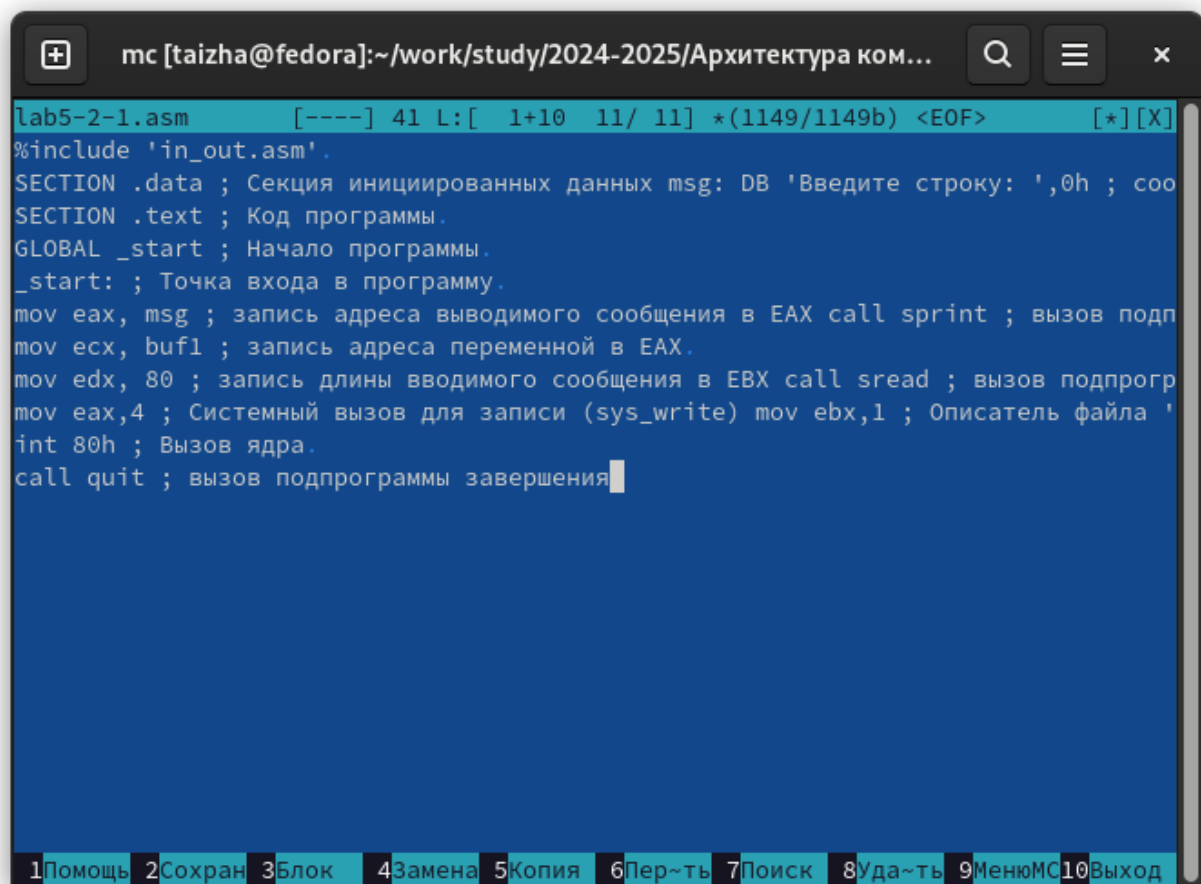


Рис.30 Редактирование файла

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

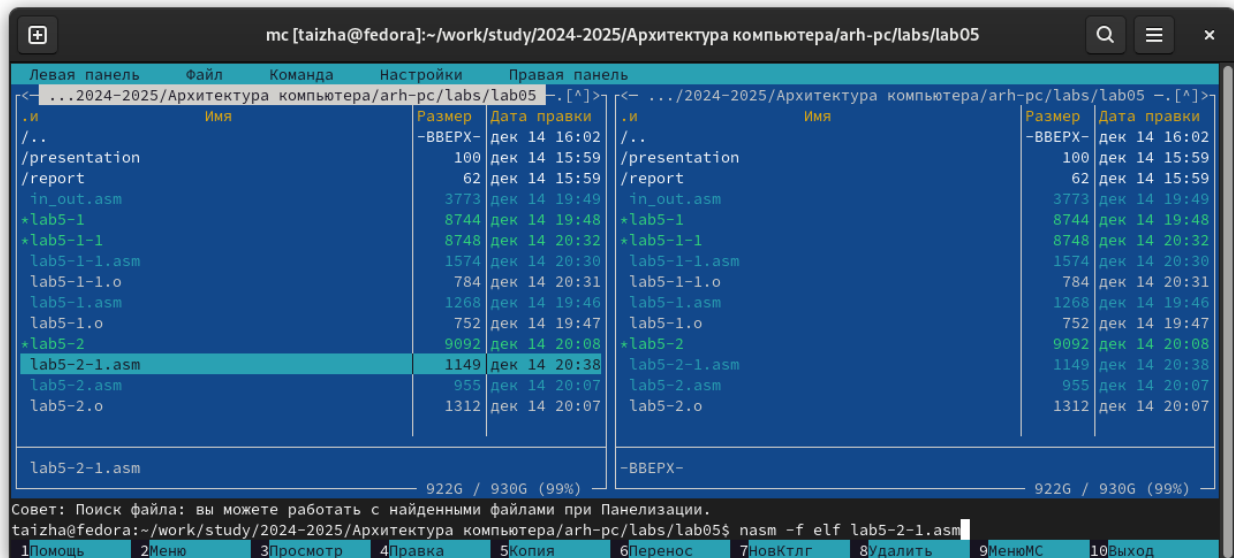


Рис.31 Трансляция файла

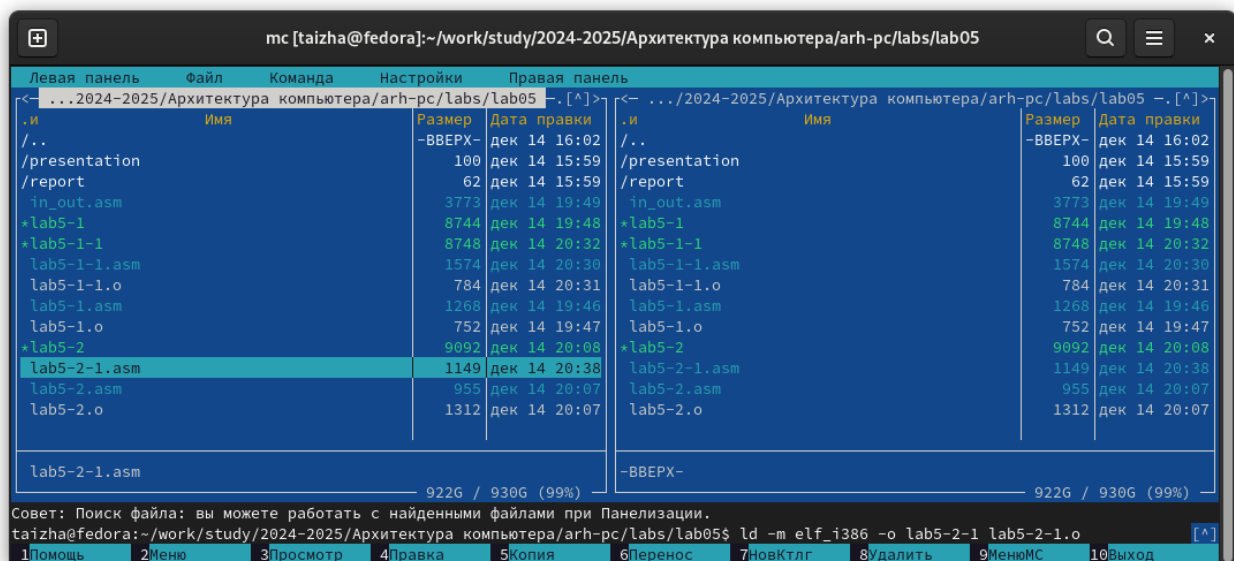


Рис.32 Компоновка файла

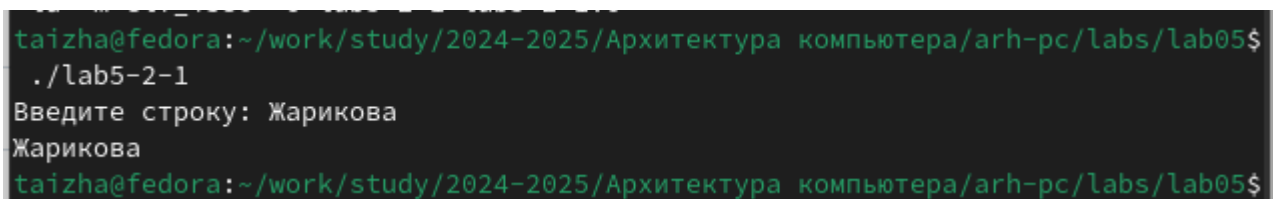


Рис.33 Исполнение файла

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.Источники

1. [Архитектура ЭВМ \(rudn.ru\)](http://rudn.ru)