

Отчёт по лабораторной работе №5

дисциплина: Архитектура компьютеров

Терещенкова Маргарита Владимировна

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	11
4.3	Подключение внешнего файла	12
4.4	Выполнение самостоятельной работы	16
5	Выводы	20
	Список литературы	21

Список иллюстраций

4.1	Открытый тс	8
4.2	Перемещение между директориями	9
4.3	Создание каталога	9
4.4	Перемещение между директориями	10
4.5	Созданный файл	10
4.6	Открытие файла для редактирования	11
4.7	Редактирование файла	11
4.8	Открытие файла для просмотра	12
4.9	Компиляция файла, передача на обработку компоновщику и исполнение файла	12
4.10	Копирование файла	13
4.11	Копирование файла	13
4.12	Редактирование файла	14
4.13	Исполнение файла	14
4.14	Редактирование файла	15
4.15	Проверка редактирования файла	15
4.16	Исполнение файла	16
4.17	Редактирование файла	17
4.18	Исполнение файла	17
4.19	Копирование файла	18
4.20	Редактирование файла	18
4.21	Проверка редактирования файла	19
4.22	Исполнение файла	19

1 Цель работы

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**.

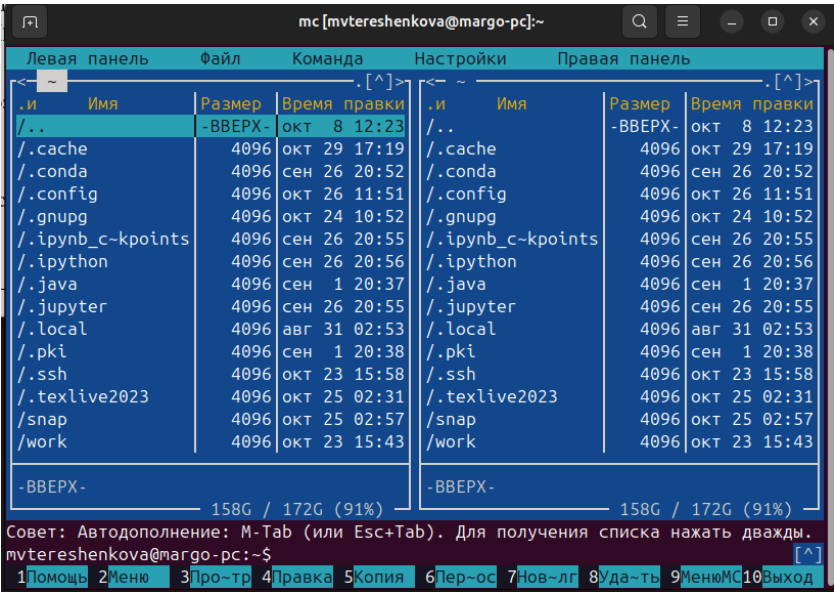


Рис. 4.1: Открытый mc

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

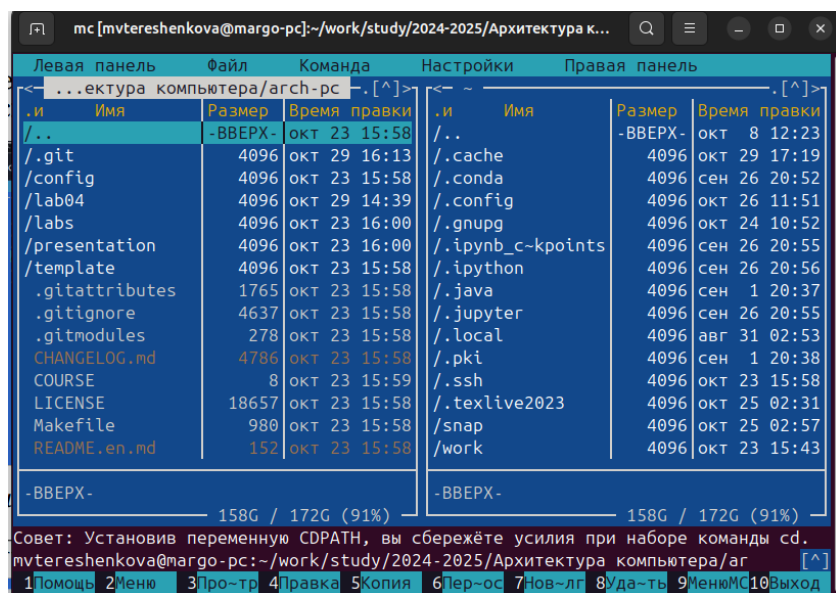


Рис. 4.2: Перемещение между директориями

С помощью функциональной клавиши **F7** создаю каталог lab05.

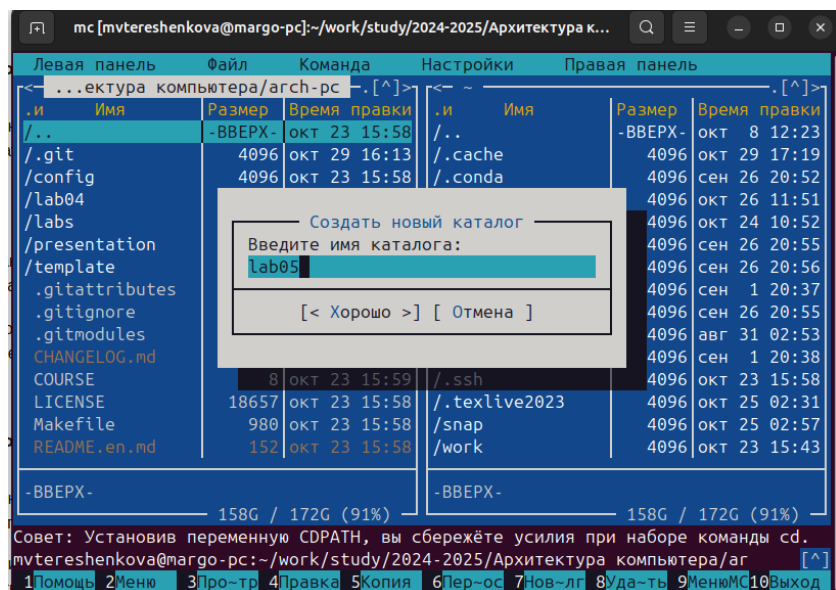


Рис. 4.3: Создание каталога

Перехожу в созданный каталог.

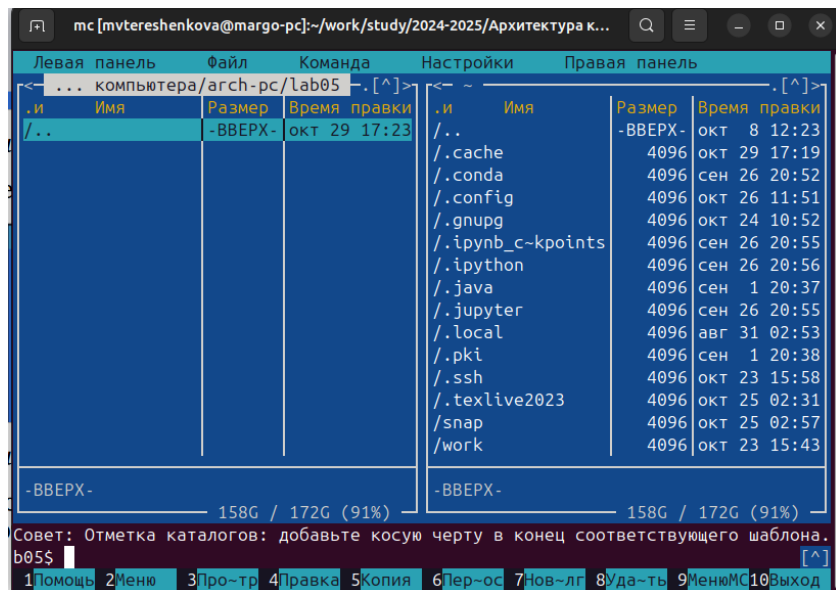


Рис. 4.4: Перемещение между директориями

В строке ввода прописываю команду **touch lab5-1.asm**, чтобы создать файл, в котором буду работать.

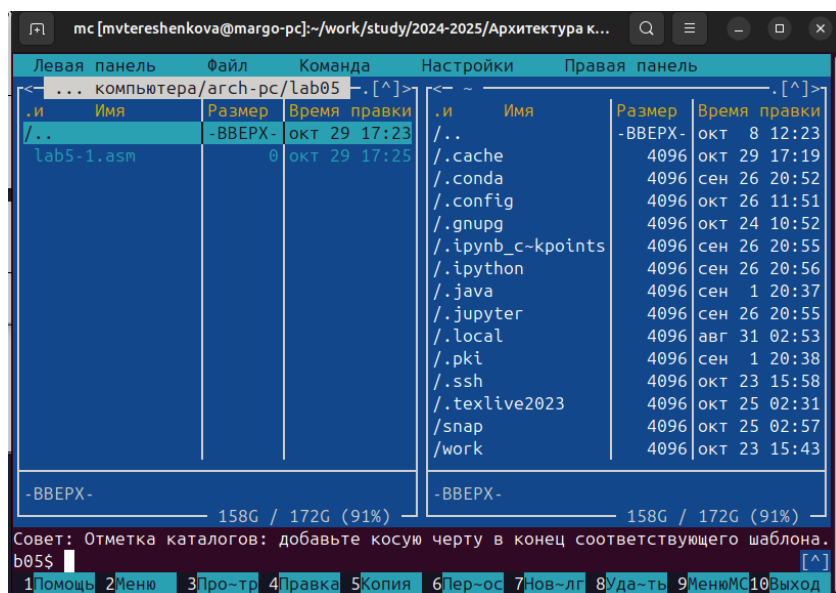


Рис. 4.5: Созданный файл

4.2 Структура программы на языке ассемблера NASM

С помощью функциональной клавиши **F4** открываю созданный файл для редактирования в редакторе nano.

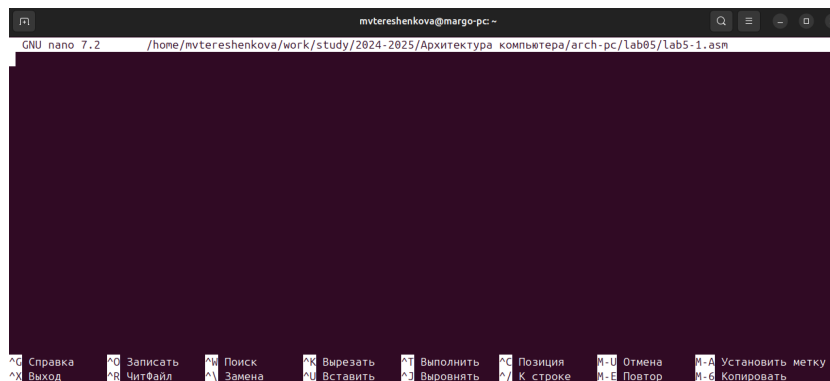


Рис. 4.6: Открытие файла для редактирования

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

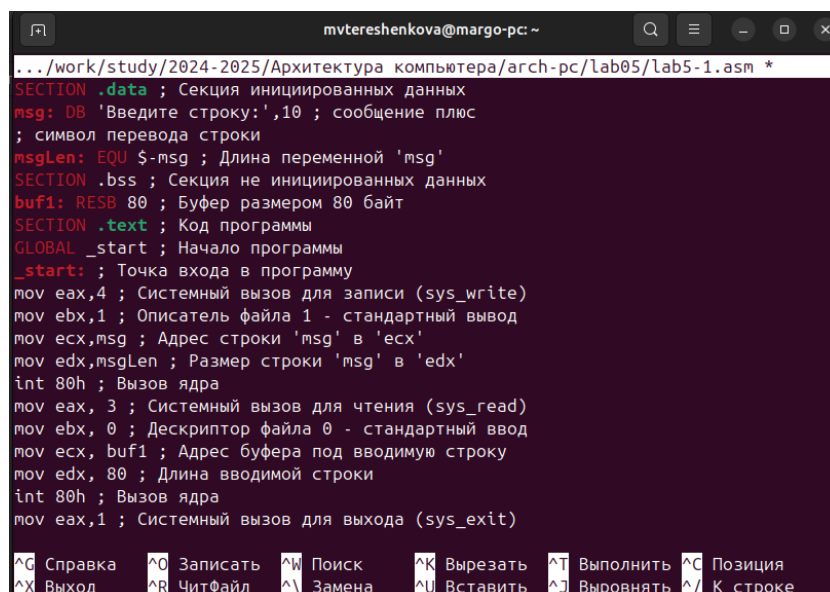
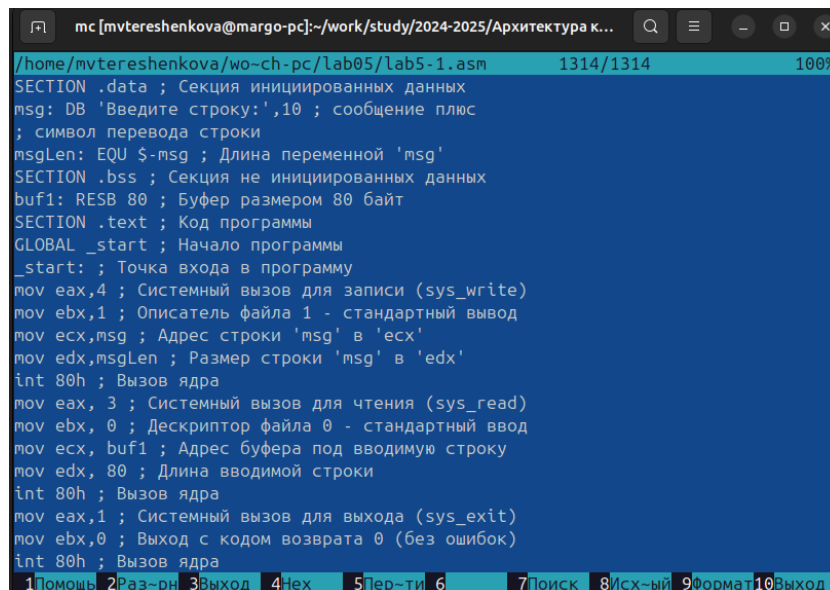


Рис. 4.7: Редактирование файла

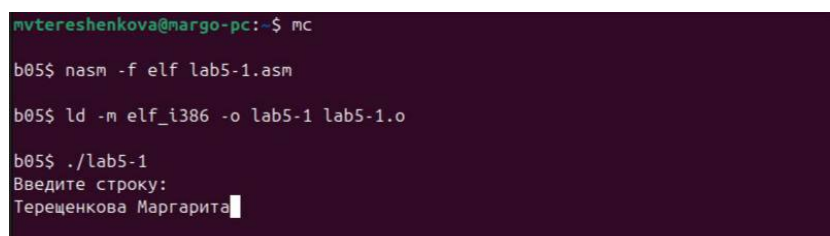
С помощью функциональной клавиши **F3** открываю файл для просмотра для проверки.



```
mc [mvtereshenkova@margo-pc]:~/work/study/2024-2025/Архитектура к...
/home/mvtereshenkova/wo-ch-pc/lab05/lab5-1.asm 1314/1314 100%
SECTION .data ; Секция иницированных данных
msg: DB 'Введите строку:',10 ; сообщение плюс
; символ перевода строки
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,1 ; Системный вызов для выхода (sys_exit)
mov ebx,0 ; Выход с кодом возврата 0 (без ошибок)
int 80h ; Вызов ядра
1Помощь 2Раз-рн 3Выход 4Нех 5Пер-ти 6 7Поиск 8Исх-ый 9Формат 10Выход
```

Рис. 4.8: Открытие файла для просмотра

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



```
mvtereshenkova@margo-pc:~$ mc
b05$ nasm -f elf lab5-1.asm
b05$ ld -m elf_i386 -o lab5-1 lab5-1.o
b05$ ./lab5-1
Введите строку:
Терещенкова Маргарита
```

Рис. 4.9: Компиляция файла, передача на обработку компоновщику и исполнение файла

4.3 Подключение внешнего файла

Скачиваю файл in_out.asm со страницы курса в ТУИС. Он сохранился в каталог “Загрузки”. С помощью функциональной клавиши **F5** копирую файл in_out.asm

из каталога Загрузки в созданный каталог lab05.

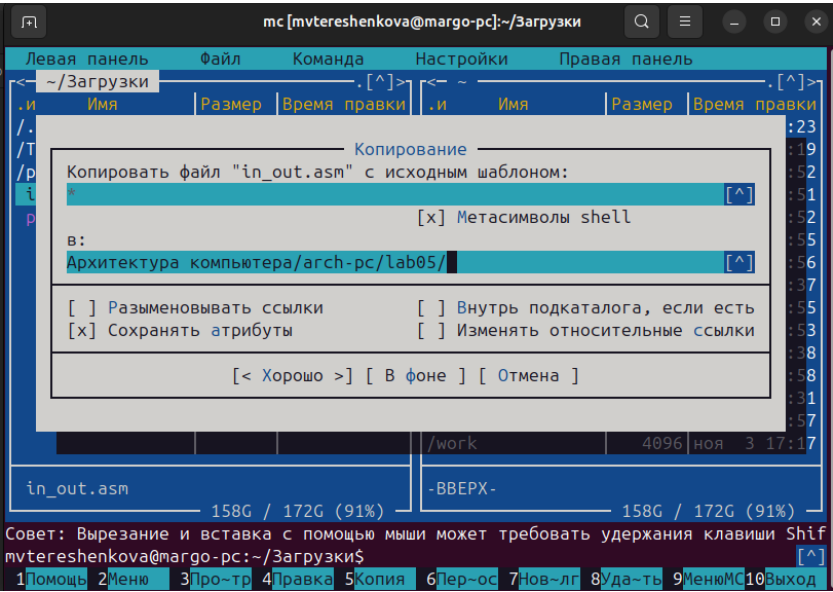


Рис. 4.10: Копирование файла

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.

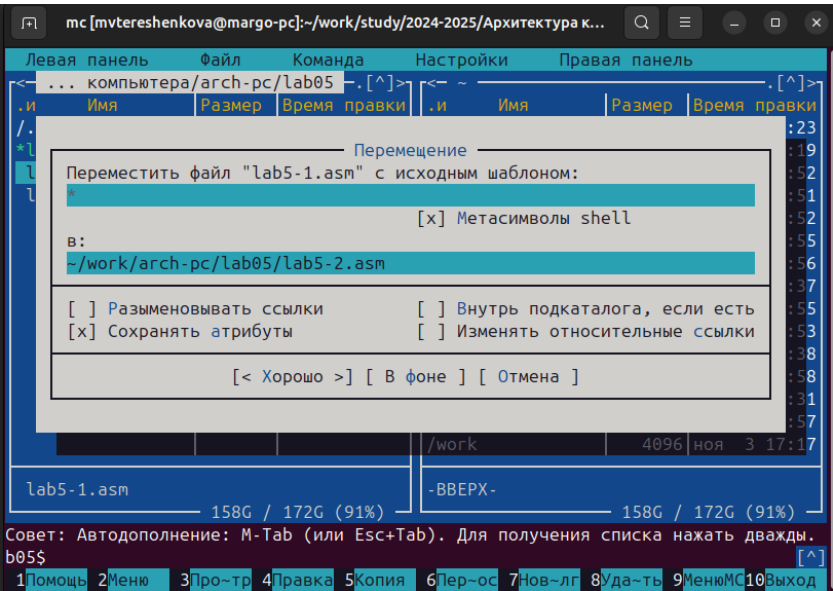
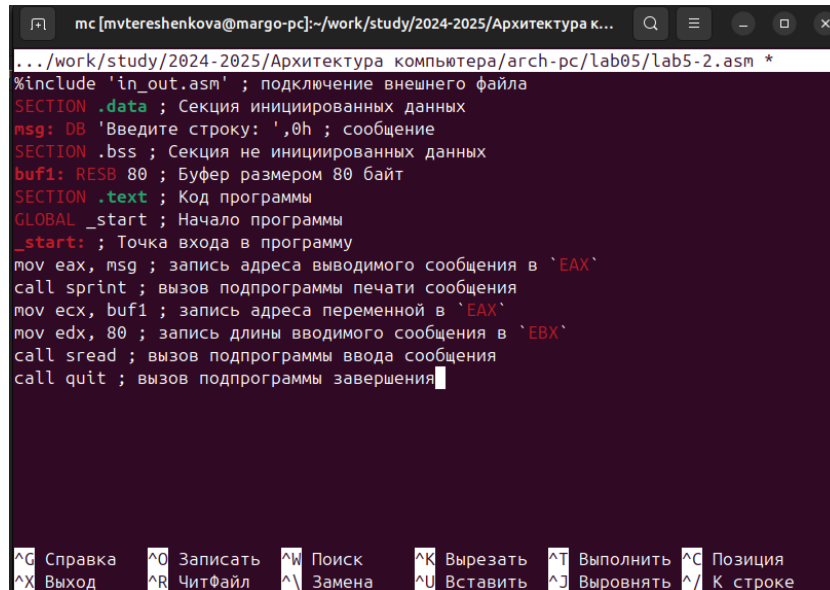


Рис. 4.11: Копирование файла

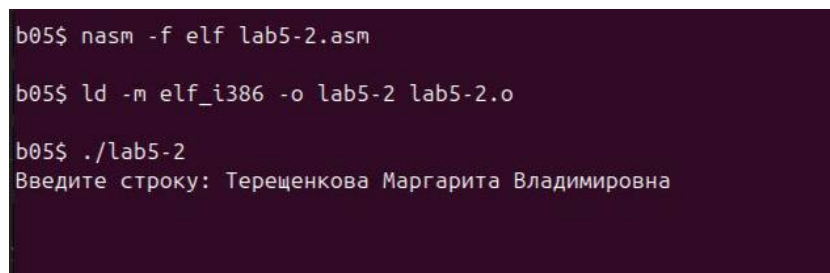
Изменяю содержимое файла lab5-2.asm во встроенном редакторе nano, чтобы в программе использовались подпрограммы из внешнего файла in_out.asm.



```
mc [mvtereshenkova@margo-pc]:~/work/study/2024-2025/Архитектура к...
.../work/study/2024-2025/Архитектура компьютера/arch-pc/lab05/lab5-2.asm *
#include 'in_out.asm' ; подключение внешнего файла
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
call quit ; вызов подпрограммы завершения
```

Рис. 4.12: Редактирование файла

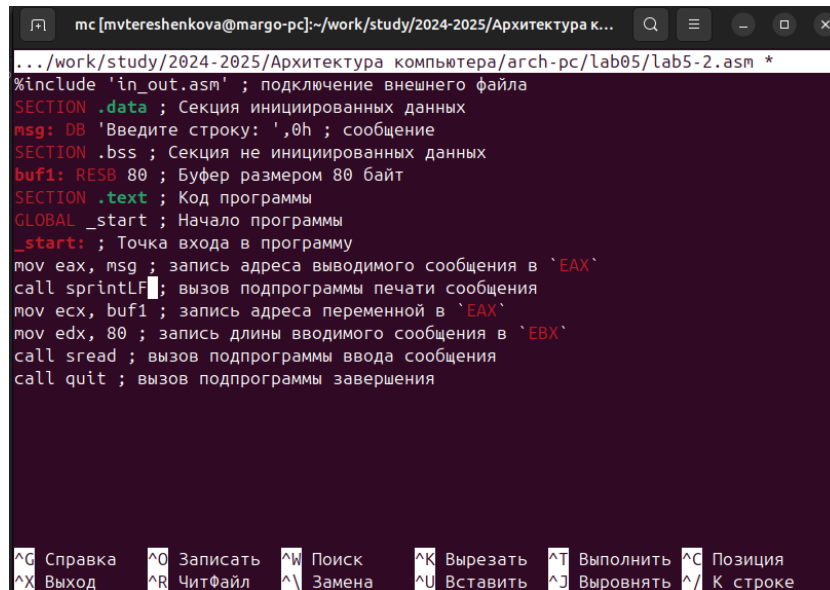
Транслирую текст программы файла в объектный файл командой **nasm -f elf lab5-2.asm**. Создался объектный файл lab5-2.o. Выполняю компоновку объектного файла с помощью команды **ld -m elf_i386 -o lab5-2 lab5-2.o**. Создался исполняемый файл lab5-2. Запускаю исполняемый файл.



```
b05$ nasm -f elf lab5-2.asm
b05$ ld -m elf_i386 -o lab5-2 lab5-2.o
b05$ ./lab5-2
Введите строку: Тереценкова Маргарита Владимировна
```

Рис. 4.13: Исполнение файла

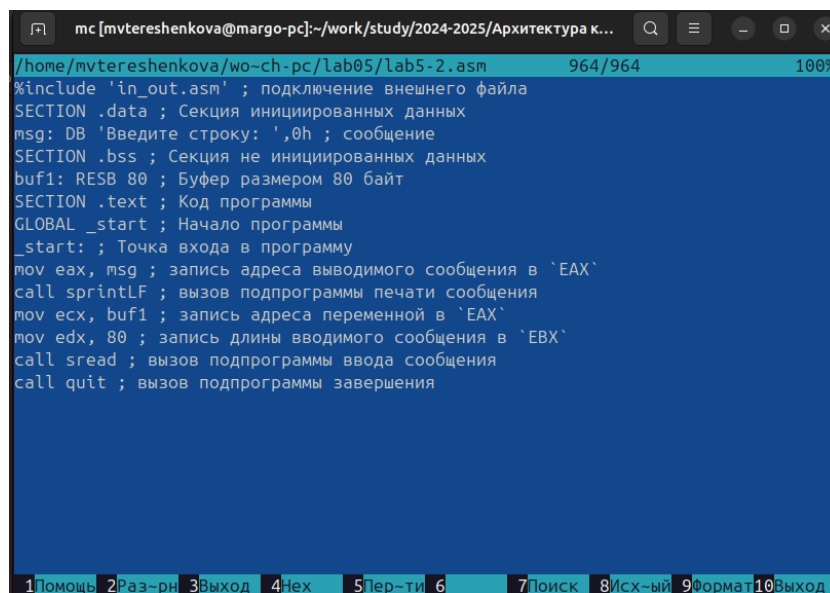
Открываю файл lab5-2.asm для редактирования в nano функциональной клавишей **F4**. Изменяю в нем подпрограмму sprint на sprintLF. Сохраняю изменения и открываю файл для просмотра клавишей **F3**, чтобы проверить сохранение действий.



```
mc [mvtereshenkova@margo-pc]:~/work/study/2024-2025/Архитектура к...
.../work/study/2024-2025/Архитектура компьютера/arch-pc/lab05/lab5-2.asm *
%include 'in_out.asm' ; подключение внешнего файла
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 sprintf ; 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
```

^G Справка ^O Записать ^W Поиск ^K Вырезать ^T Выполнить ^C Позиция
^X Выход ^R ЧитФайл ^\ Замена ^U Вставить ^J Выводить ^_ К строке

Рис. 4.14: Редактирование файла



```
/home/mvtereshenkova/wo-ch-pc/lab05/lab5-2.asm 964/964 100%
%include 'in_out.asm' ; подключение внешнего файла
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 sprintf ; 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 2Раз-ри 3Выход 4Hex 5Пер-ти 6 7Поиск 8Исх-ый 9Формат10Выход

Рис. 4.15: Проверка редактирования файла

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

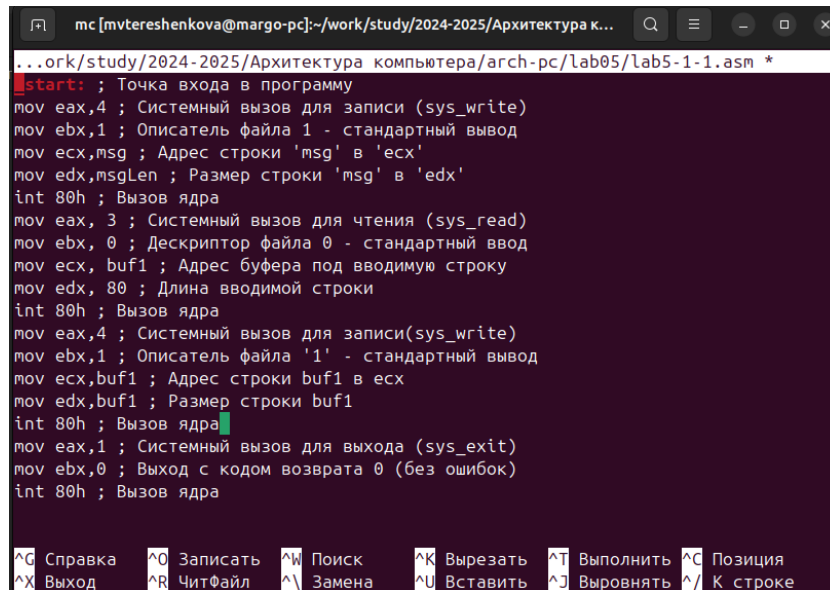
```
b05$ nasm -f elf lab5-2.asm  
  
mvtereshenkova@margo-pc:~$ mc  
  
b05$ ld -m elf_i386 -o lab5-2-2 lab5-2.o  
  
b05$ ./lab5-2-2  
Введите строку:  
Терещенкова Маргарита Владимировна
```

Рис. 4.16: *Исполнение файла*

Разница между первым исполняемым файлом lab5-2 и вторым lab5-2-2 в том, что запуск первого запрашивает ввод без переноса на новую строку, а программа, которая исполняется при запуске второго, запрашивает ввод с новой строки, потому что в этом заключается различие между подпрограммами `sprint` и `sprintLF`.

4.4 Выполнение самостоятельной работы

1. Создаю копию файла lab5-1.asm с именем lab5-1-1.asm с помощью функциональной клавиши **F5**. 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.

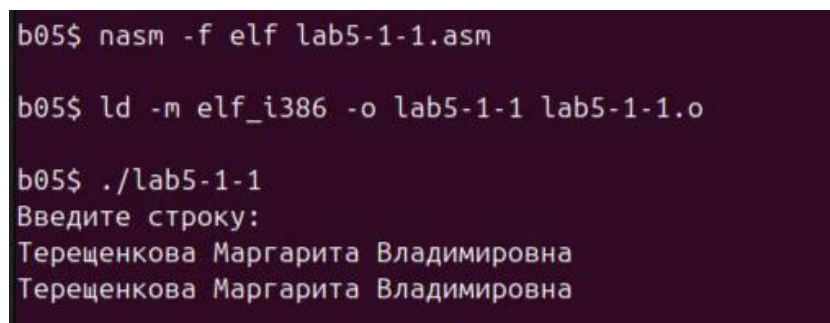


```
mc [mvtereshenkova@margo-pc]:~/work/study/2024-2025/Архитектура к...
...ork/study/2024-2025/Архитектура компьютера/arch-pc/lab05/lab5-1-1.asm *
start: ; Точка входа в программу
mov eax,4 ; Системный вызов для записи (sys_write)
mov ebx,1 ; Описатель файла 1 - стандартный вывод
mov ecx,msg ; Адрес 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
```

^G Справка ^O Записать ^W Поиск ^K Вырезать ^T Выполнить ^C Позиция
^X Выход ^R ЧитФайл ^\ Замена ^U Вставить ^J Выводить ^_ К строке

Рис. 4.17: Редактирование файла

2. 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.



```
b05$ nasm -f elf lab5-1-1.asm
b05$ ld -m elf_i386 -o lab5-1-1 lab5-1-1.o
b05$ ./lab5-1-1
Введите строку:
Терещенкова Маргарита Владимировна
Терещенкова Маргарита Владимировна
```

Рис. 4.18: Исполнение файла

3. Создаю копию файла lab5-2.asm с именем lab5-2-1.asm с помощью функциональной клавиши **F5**.

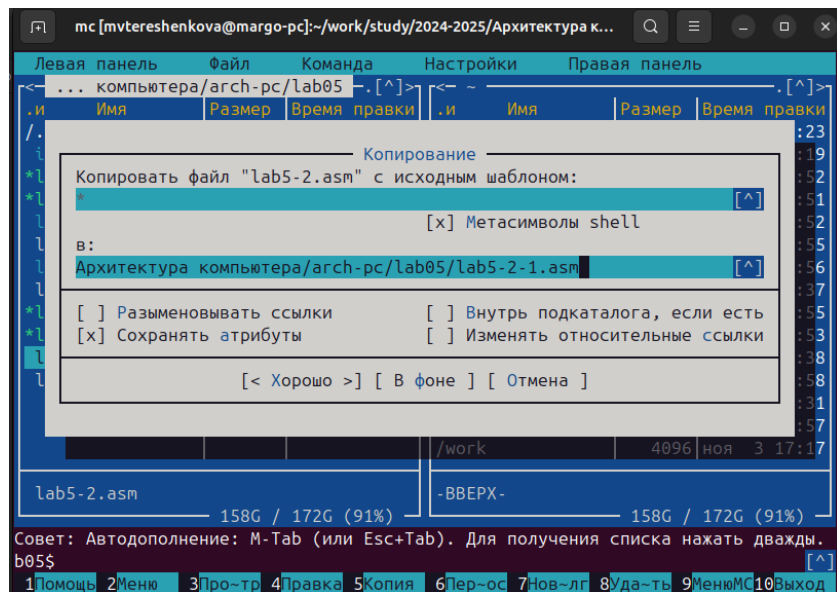


Рис. 4.19: Копирование файла

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.

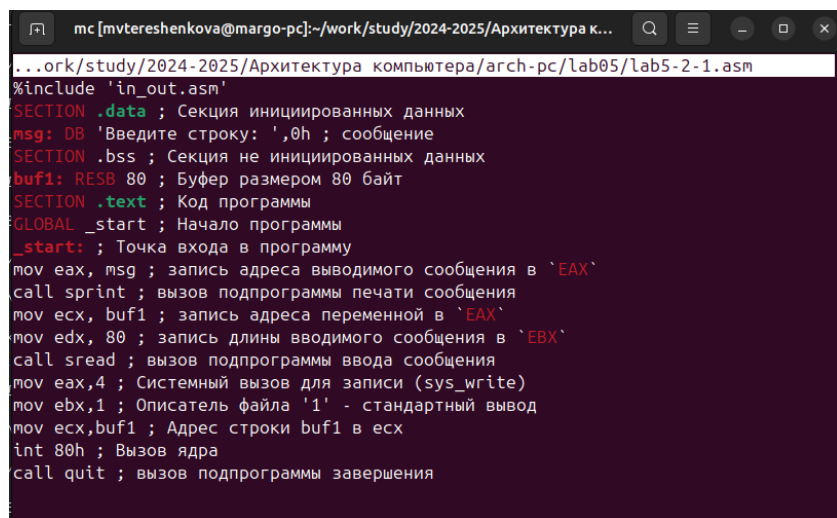


Рис. 4.20: Редактирование файла

С помощью функциональной клавиши **F3** открываю файл для просмотра для проверки.

```
/home/mvtereshenkova/wo--pc/lab05/lab5-2-1.asm 1146/1146 100%
%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. 4.21: Проверка редактирования файла

4. 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.

```
b05$ nasm -f elf lab5-2-1.asm
b05$ ld -m elf_i386 -o lab5-2-1 lab5-2-1.o
b05$ ./lab5-2-1
Введите строку: Терещенкова Маргарита Владимировна
Терещенкова Маргарита Владимировна
```

Рис. 4.22: Исполнение файла

5 Выводы

Благодаря данной лабораторной работе приобрела практические навыки работы в Midnight Commander. Освоение инструкций языка ассемблера mov и int.

Список литературы

Архитектура ЭВМ