

Отчёт по лабораторной работе №5

Дисциплина: архитектура компьютера

Барбакова Алиса Саяновна

Содержание

1	Цель работы	4
2	Задание	5
3	Теоретическое введение	6
4	Выполнение лабораторной работы	8
4.1	Основы работы с тс	8
4.2	Структура программы на языке ассемблера NASM	11
4.3	Подключение внешнего файла	15
4.4	Выполнение заданий для самостоятельной работы	19
5	Выводы	25
	Список литературы	26

Список иллюстраций

4.1	мс	8
4.2	Перемещение между директориями	9
4.3	Создание каталога	9
4.4	Переход в каталог	10
4.5	Создание файла	11
4.6	Открытие файла для редактирования	12
4.7	Редактирование файла	13
4.8	Открытие файла для просмотра	14
4.9	Исполнение файла	14
4.10	Скачанный файл	15
4.11	Копирование файла	15
4.12	Копирование файла	16
4.13	Редактирование файла	17
4.14	Исполнение файла	17
4.15	Отредактированный файл	18
4.16	Исполнение файла	18
4.17	Копирование файла	19
4.18	Редактирование файла	20
4.19	Исполнение файла	22
4.20	Копирование файла	22
4.21	Редактирование файла	23
4.22	Исполнение файла	24

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 Midnight Commander
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. Для активации оболочки Midnight Commander достаточно ввести в командной строке mc и нажать клавишу Enter (рис. 5.1). В Midnight Commander используются функциональные клавиши F1 — F10, к которым привязаны часто выполняемые операции. 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 (учетверённое 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).

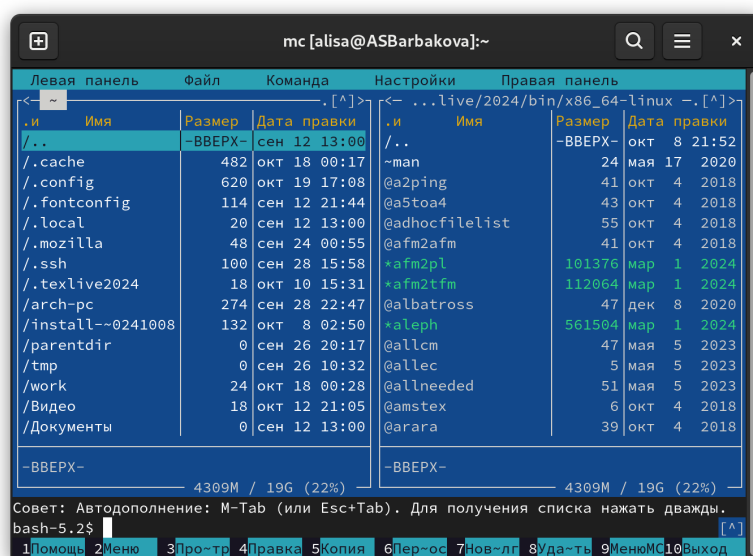


Рис. 4.1: mc

Пользуясь клавишами, перехожу в каталог ~/work/arch-rc. (рис. 4.2)

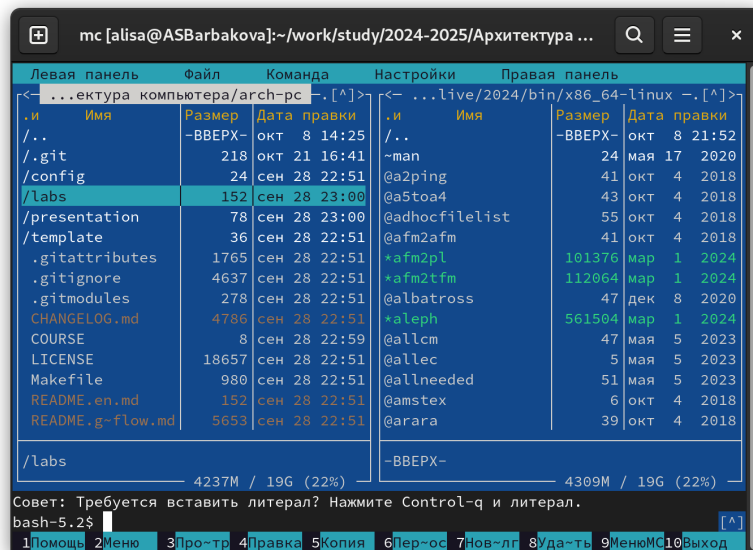


Рис. 4.2: Перемещение между директориями

С помощью функциональной клавиши F7 создаю папку lab05 (рис. 4.3).

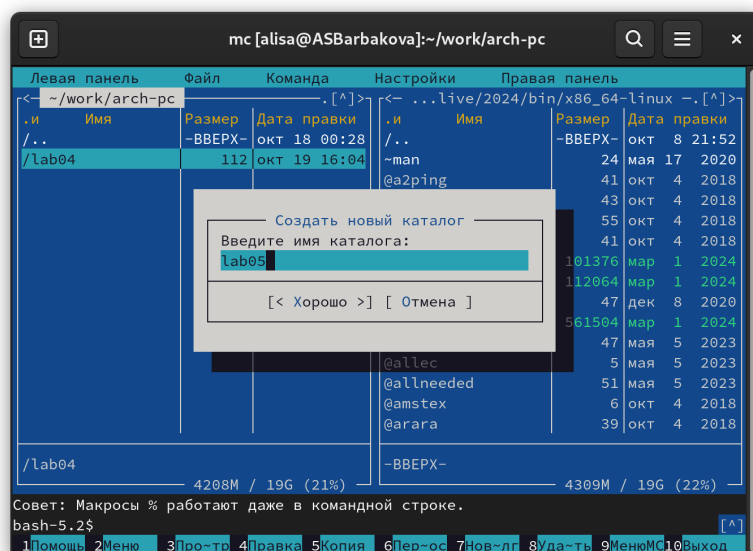


Рис. 4.3: Создание каталога

Перехожу в созданный каталог (рис. 4.4).

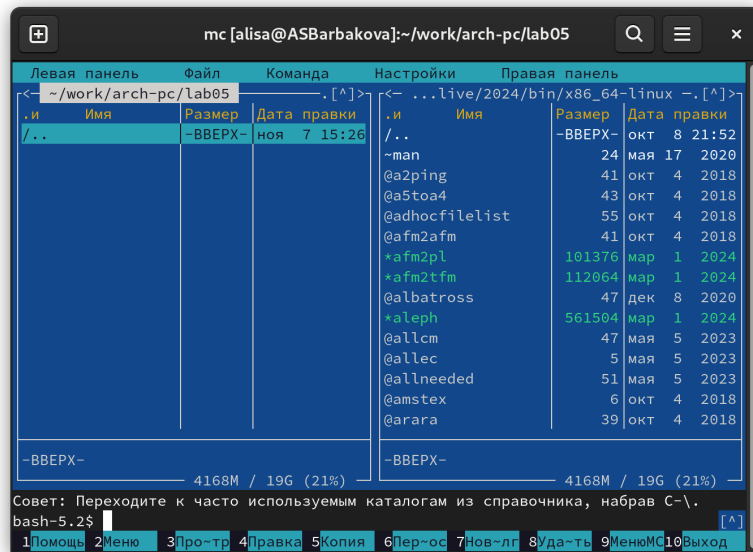


Рис. 4.4: Переход в каталог

В строке ввода пользуюсь командой `touch` и создаю файл `lab5-1.asm`, в котором буду работать (рис. 4.5).

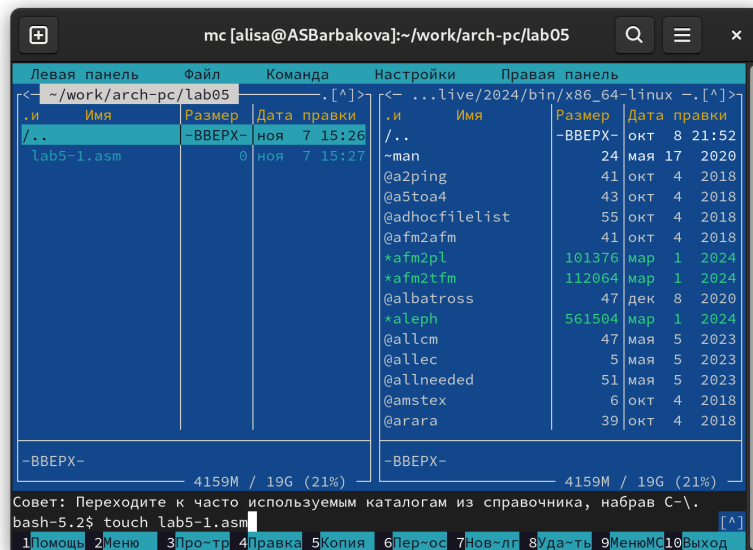


Рис. 4.5: Создание файла

4.2 Структура программы на языке ассемблера NASM

С помощью функциональной клавиши F4 открываю созданный файл для редактирования в редакторе mcedit (рис. 4.6).

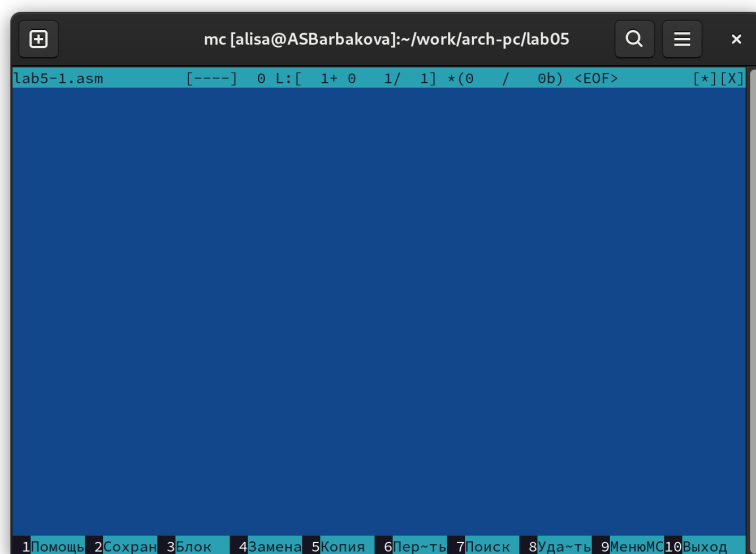
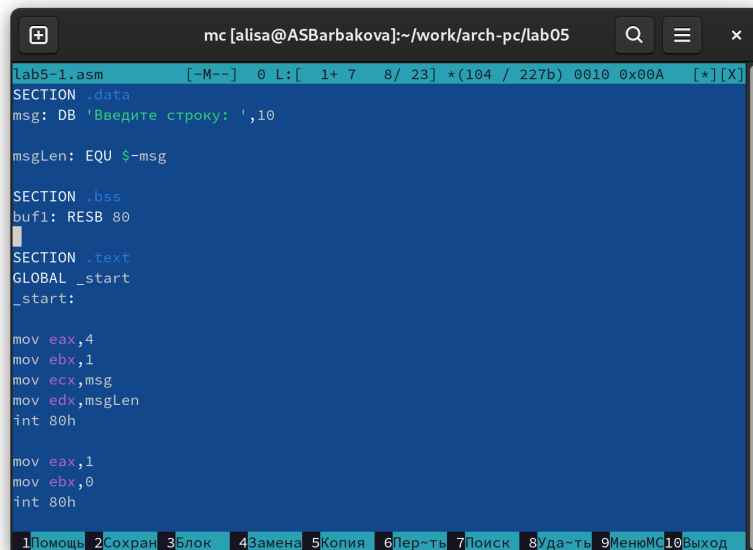


Рис. 4.6: Открытие файла для редактирования

Ввожу в файл код программы для запроса строки у пользователя (рис. 4.7). Далее выхожу из файла, сохраняя изменения.



The screenshot shows a text editor window titled 'mc [alisa@ASBarbakova]:~/work/arch-pc/lab05'. The editor displays assembly code for a file named 'lab5-1.asm'. The code includes sections for data, bss, and text. The data section contains a message 'Введите строку: ',10. The bss section reserves 80 bytes for a buffer. The text section contains assembly instructions for moving values into registers, setting up a loop, and interrupting the system. The bottom of the window features a menu bar with options: 1Помощь, 2Сохран, 3Блок, 4Замена, 5Копия, 6Пер-ть, 7Поиск, 8Уда-ть, 9МенюMC10, and 10Выход.

```
lab5-1.asm [-M--] 0 L:[ 1+ 7 8/ 23] *(104 / 227b) 0010 0x00A [*][X]
SECTION .data
msg: DB 'Введите строку: ',10

msgLen: EQU $-msg

SECTION .bss
buf1: RESB 80

SECTION .text
GLOBAL _start
_start:

mov eax,4
mov ebx,1
mov ecx,msg
mov edx,msgLen
int 80h

mov eax,1
mov ebx,0
int 80h
```

Рис. 4.7: Редактирование файла

С помощью функциональной клавиши F3 открываю файл для просмотра, чтобы проверить, содержит ли файл текст программы (рис. 4.8).

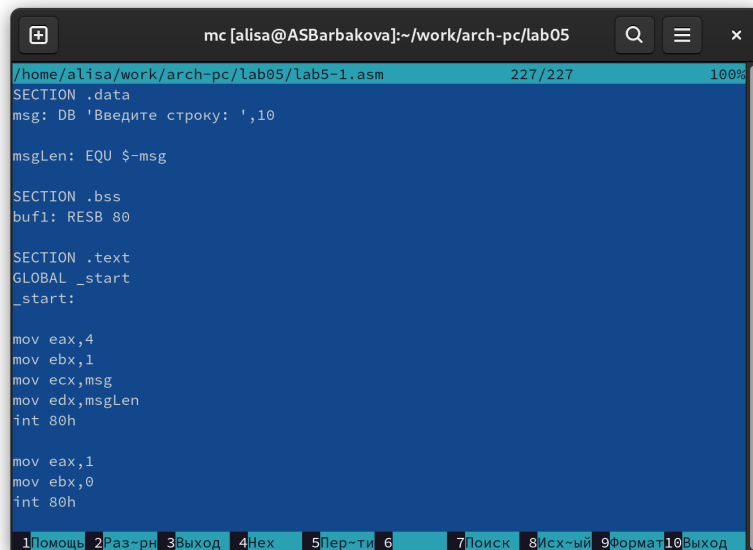


Рис. 4.8: Открытие файла для просмотра

Транслирую текст программы файла в объектный файл командой `nasm -f elf lab5-1.asm`. Выполняю компоновку объектного файла. Создался исполняемый файл `lab5-1`. Запускаю исполняемый файл. Программа выводит строку “Введите строку:”, я ввожу свои ФИО с клавиатуры (рис. 4.9).

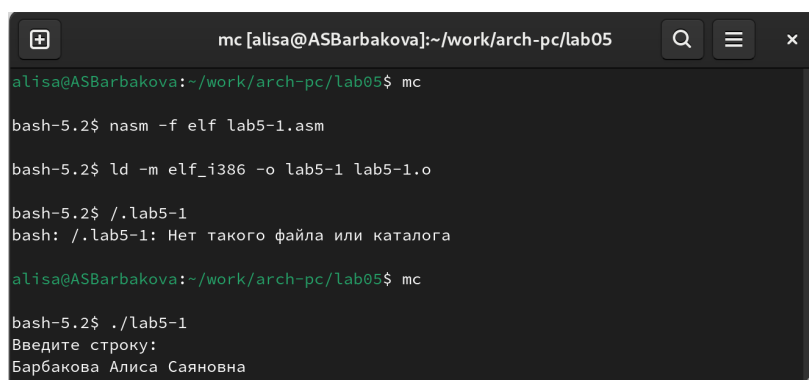


Рис. 4.9: Исполнение файла

4.3 Подключение внешнего файла

Скачиваю файл in_out.asm со страницы курса в ТУИС (рис. 4.10).

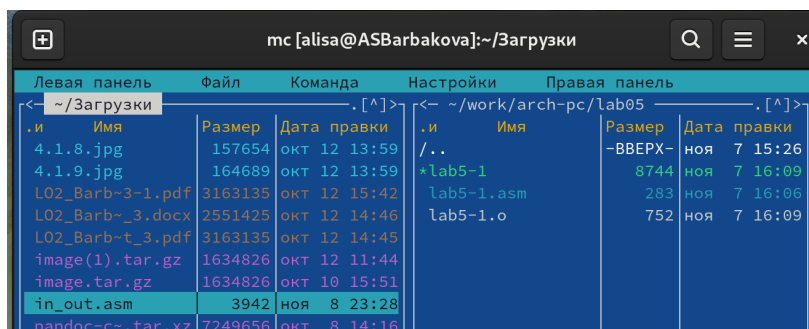


Рис. 4.10: Скачанный файл

С помощью F5 копирую файл in_out.asm из каталога Загрузки в созданный каталог lab05 (рис. 4.11).

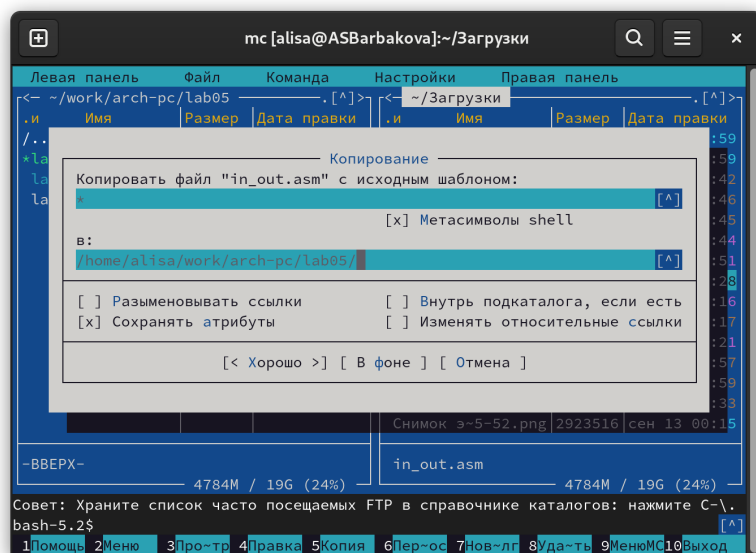


Рис. 4.11: Копирование файла

С помощью функциональной клавиши F5 создаю копию файла lab5-1 в тот же каталог, но с другим именем. В появившемся окне mc прописываю имя для копии файла (рис. 4.12).

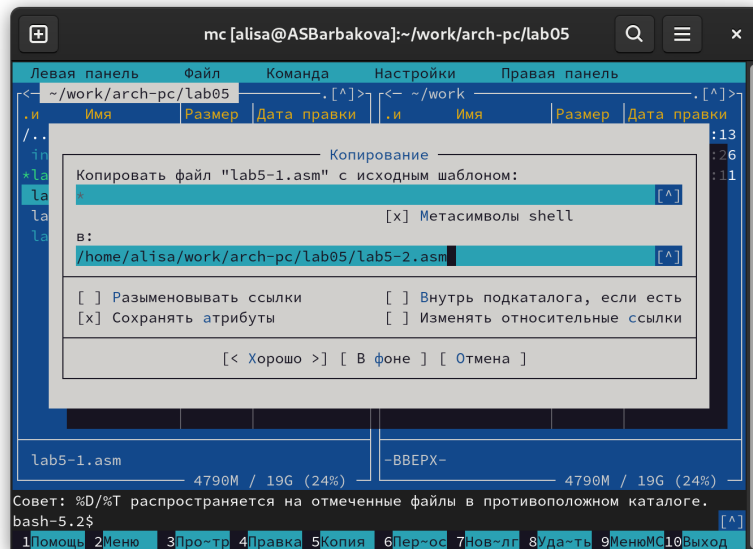
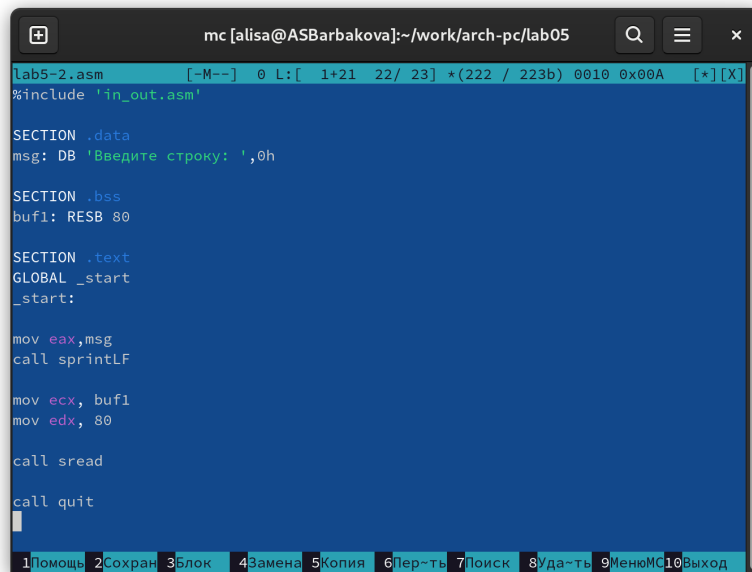


Рис. 4.12: Копирование файла

Исправляю текст программы в файле lab5-2.asm с использованием подпрограмм из внешнего файла in_out.asm - sprintLF, sread и quit. (рис. 4.13).



```
lab5-2.asm [-M--] 0 L: [ 1+21 22/ 23] *(222 / 223b) 0010 0x00A [*] [X]
%include 'in_out.asm'

SECTION .data
msg: DB 'Введите строку: ',0h

SECTION .bss
buf1: RESB 80

SECTION .text
GLOBAL _start
_start:

mov eax,msg
call sprintLF

mov ecx, buf1
mov edx, 80

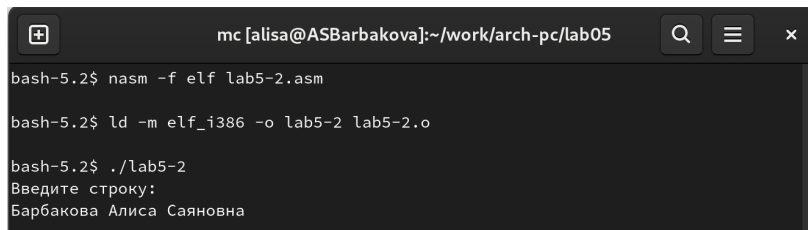
call sread

call quit
```

1Помощь 2Сохран 3Блок 4Замена 5Копия 6Пер-ть 7Поиск 8И-да-ть 9МенюMC10Выход

Рис. 4.13: Редактирование файла

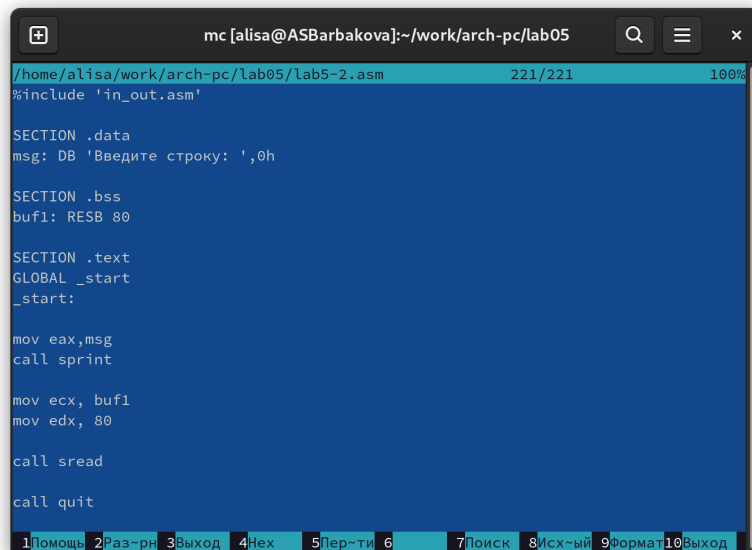
Создаю исполняемый файл и проверяю его работу. (рис. 4.14).



```
bash-5.2$ nasm -f elf lab5-2.asm
bash-5.2$ ld -m elf_i386 -o lab5-2 lab5-2.o
bash-5.2$ ./lab5-2
Введите строку:
Барбакова Алиса Саяновна
```

Рис. 4.14: Исполнение файла

Открываю файл lab5-2.asm для редактирования в mscedit. Изменяю в нем подпрограмму sprintLF на sprint. (рис. 4.15).



The screenshot shows a text editor window titled 'mc [alisa@ASBarbakova]:~/work/arch-pc/lab05'. The file being edited is '/home/alisa/work/arch-pc/lab05/lab5-2.asm' at line 221 of 221. The code includes a directive to include 'in_out.asm', followed by section declarations for .data, .bss, and .text. The .data section contains a string 'Введите строку: ',0h. The .bss section reserves 80 bytes for 'buf1'. The .text section starts at _start, moves the address of msg into eax, calls sprint, moves the address of buf1 into ecx, sets edx to 80, calls sread, and finally calls quit. At the bottom, there is a menu bar with options: 1Помощь, 2Раз-рн, 3Выход, 4Hex, 5Пер-ти, 6, 7Поиск, 8/сх-ый, 9Формат, 10Выход.

```
/home/alisa/work/arch-pc/lab05/lab5-2.asm 221/221 100%
#include 'in_out.asm'

SECTION .data
msg: DB 'Введите строку: ',0h

SECTION .bss
buf1: RESB 80

SECTION .text
GLOBAL _start
_start:

mov eax,msg
call sprint

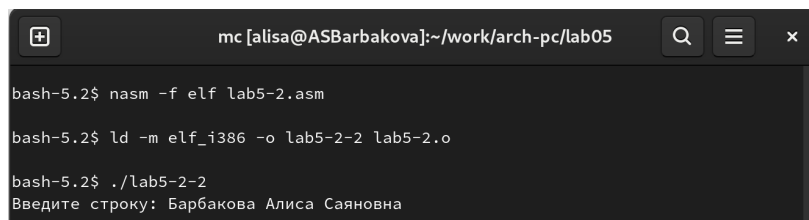
mov ecx, buf1
mov edx, 80

call sread

call quit
```

Рис. 4.15: Отредактированный файл

Снова создаю исполняемый файл и проверяю его работу (рис. 4.16).



The screenshot shows a terminal window with the following commands and output: 'nasm -f elf lab5-2.asm' is executed successfully. Then 'ld -m elf_i386 -o lab5-2-2 lab5-2.o' is executed successfully. Finally, './lab5-2-2' is executed, and the prompt changes to 'Введите строку: Барбакова Алиса Саяновна', indicating the program is running and waiting for input.

```
bash-5.2$ nasm -f elf lab5-2.asm
bash-5.2$ ld -m elf_i386 -o lab5-2-2 lab5-2.o
bash-5.2$ ./lab5-2-2
Введите строку: Барбакова Алиса Саяновна
```

Рис. 4.16: Исполнение файла

Разница подпрограммы sprint в том, что исполняемый файл запрашивает ввод без переноса на новую строку.

4.4 Выполнение заданий для самостоятельной работы

1. Создаю копию файла lab5-1.asm с именем lab5-1-1.asm с помощью функциональной клавиши F5 (рис. 4.17).

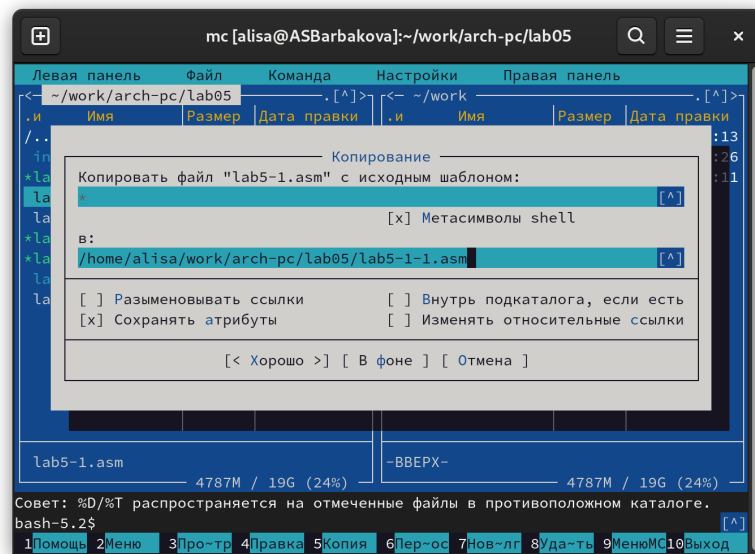


Рис. 4.17: Копирование файла

Редактирую файл, изменяю программу так, чтобы она выводила вводимую пользователем строку (рис. 4.18).

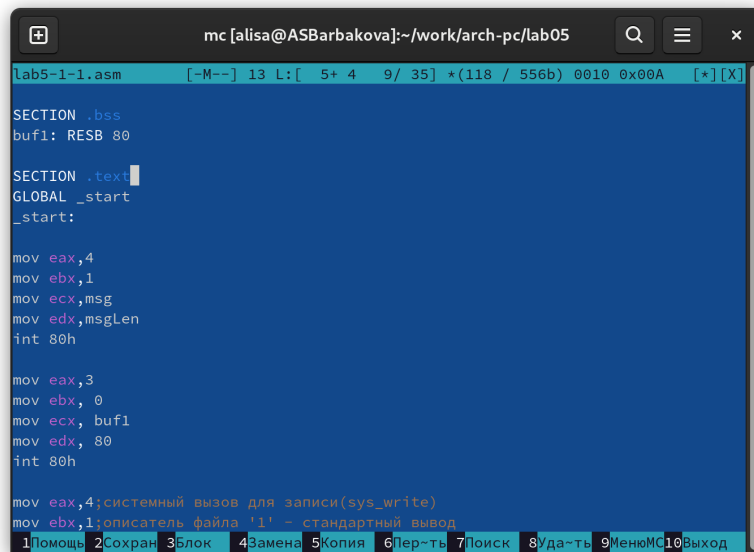


Рис. 4.18: Редактирование файла

Код программы:

SECTION .data

msg: DB 'Введите строку: ',10

msgLen: EQU \$-msg

SECTION .bss

buf1: RESB 80

SECTION .text

GLOBAL _start

_start:

mov eax,4

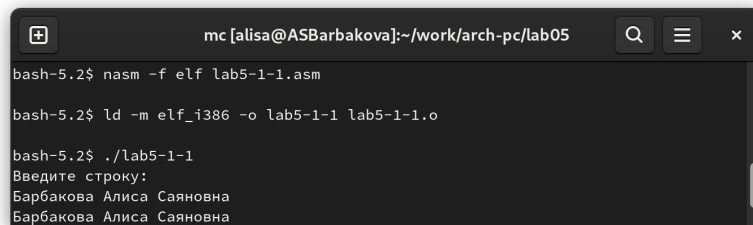
```
mov ebx,1
mov ecx,msg
mov edx,msgLen
int 80h
```

```
mov eax,3
mov ebx, 0
mov ecx, buf1
mov edx, 80
int 80h
```

```
mov eax,4;системный вызов для записи(sys_write)
mov ebx,1;описатель файла '1' - стандартный вывод
mov ecx,buf1;адрес строки buf1 в есх
mov edx,buf1;размер строки buf1
int 80h;вызов ядра
```

```
mov eax,1
mov ebx,0
int 80h
```

2. Создаю объектный файл lab5-1-1.o, отдаю его на обработку компоновщику, получаю исполняемый файл lab5-1-1, запускаю полученный исполняемый файл. Ввожу свои ФИО, программа выводит введенные данные (рис. 4.19).



```
mc [alisa@ASBarbakova]:~/work/arch-pc/lab05
bash-5.2$ nasm -f elf lab5-1-1.asm
bash-5.2$ ld -m elf_i386 -o lab5-1-1 lab5-1-1.o
bash-5.2$ ./lab5-1-1
Введите строку:
Барбакова Алиса Саяновна
Барбакова Алиса Саяновна
```

Рис. 4.19: Исполнение файла

3. Создаю копию файла lab5-2.asm с именем lab5-2-1.asm с помощью F5 (рис. 4.20).

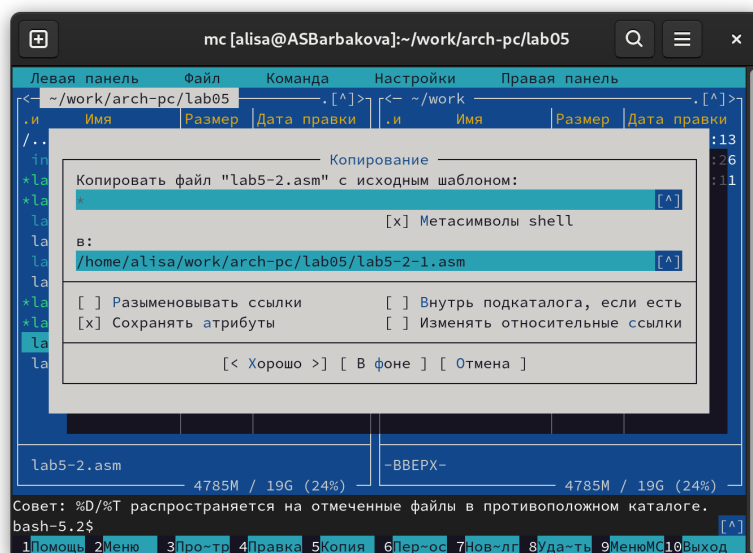


Рис. 4.20: Копирование файла

С помощью F4 открываю созданный файл для редактирования. Изменяю программу так, чтобы, она выводила вводимую пользователем строку (рис. 4.21).

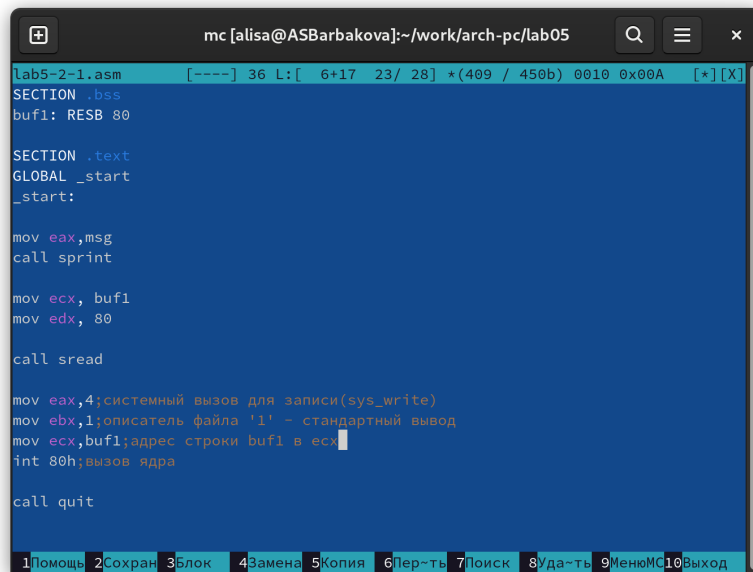


Рис. 4.21: Редактирование файла

Код программы:

```
%include 'in_out.asm'
```

```
SECTION .data
```

```
msg: DB 'Введите строку: ',0h
```

```
SECTION .bss
```

```
buf1: RESB 80
```

```
SECTION .text
```

```
GLOBAL _start
```

```
_start:
```

```
mov eax,msg
```

```
call sprint
```

```
mov ecx, buf1
```

```
mov edx, 80
```

```
call sread
```

```
mov eax,4;системный вызов для записи(sys_write)
```

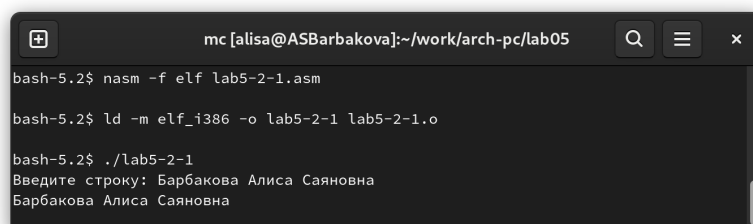
```
mov ebx,1;описатель файла '1' - стандартный вывод
```

```
mov ecx,buf1;адрес строки buf1 в ecx
```

```
int 80h;вызов ядра
```

```
call quit
```

4. Создаю 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 (рис. 4.22).



```
mc [alisa@ASBarbakova]:~/work/arch-pc/lab05
bash-5.2$ nasm -f elf lab5-2-1.asm
bash-5.2$ ld -m elf_i386 -o lab5-2-1 lab5-2-1.o
bash-5.2$ ./lab5-2-1
Введите строку: Барбакова Алиса Саяновна
Барбакова Алиса Саяновна
```

Рис. 4.22: 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

1. Лабораторная работа №5