

# **Отчёт по лабораторной работе №5**

**Дисциплина: Архитектура компьютера**

Зарицкая Марина Петровна

# Содержание

<b>1</b>	<b>Цель работы</b>	<b>5</b>
<b>2</b>	<b>Задание</b>	<b>6</b>
<b>3</b>	<b>Теоретическое введение</b>	<b>7</b>
<b>4</b>	<b>Выполнение лабораторной работы</b>	<b>9</b>
4.1	Основы работы с Midnight Commander . . . . .	9
4.2	Структура программы на языке ассемблера NASM . . . . .	11
4.3	Подключение внешнего файла in_out.asm . . . . .	14
4.4	Задание для самостоятельной работы . . . . .	18
<b>5</b>	<b>Выводы</b>	<b>24</b>

## Список иллюстраций

4.1	Окно Midnight Commander . . . . .	9
4.2	Создание каталога . . . . .	10
4.3	Создание файла lab5-1.asm . . . . .	10
4.4	Редактирование файла . . . . .	11
4.5	Открытие файла для просмотра . . . . .	12
4.6	Создание объектного файла lab5-1.o . . . . .	13
4.7	Компоновка объектного файла . . . . .	14
4.8	Запуск программы . . . . .	14
4.9	Копирование файла . . . . .	15
4.10	Создание копии файла . . . . .	15
4.11	Изменение файла . . . . .	16
4.12	Исполнение файла . . . . .	16
4.13	Отредактированный файл . . . . .	17
4.14	Исполнение файла . . . . .	17
4.15	Копирование файла . . . . .	18
4.16	Редактирование файла . . . . .	19
4.17	Исполнение файла . . . . .	19
4.18	Копирование файла . . . . .	21
4.19	Редактирование файла . . . . .	22
4.20	Исполнение файла . . . . .	22

## **Список таблиц**

# 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 in\_out.asm
4. Задание 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 Midnight Commander

Открываю Midnight Commander с помощью команды `mc` (рис. 4.1).

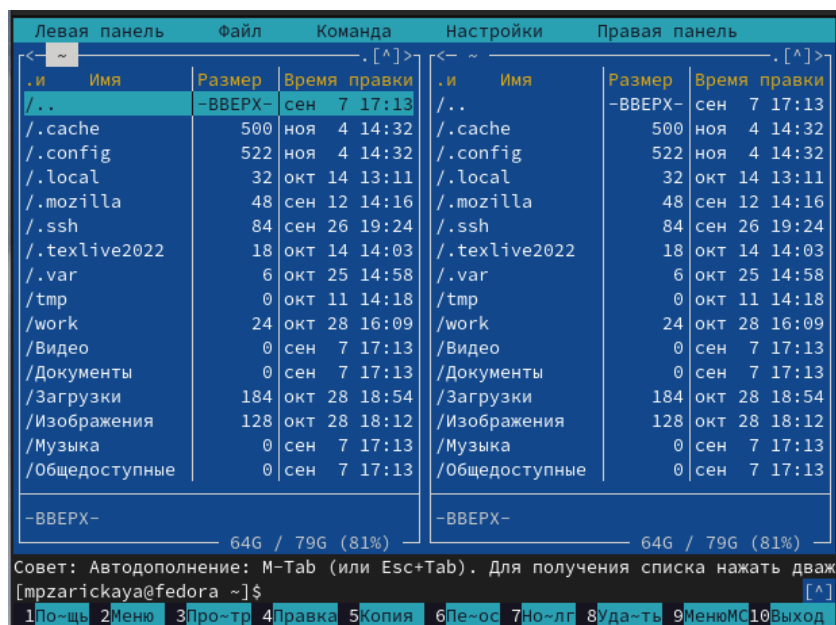


Рис. 4.1: Окно Midnight Commander

Перехожу в каталог `~/work/arch-рс` созданный при выполнении лабораторной работы №4. С помощью функциональной клавиши F7 создаю каталог `lab05` (рис. 4.2).

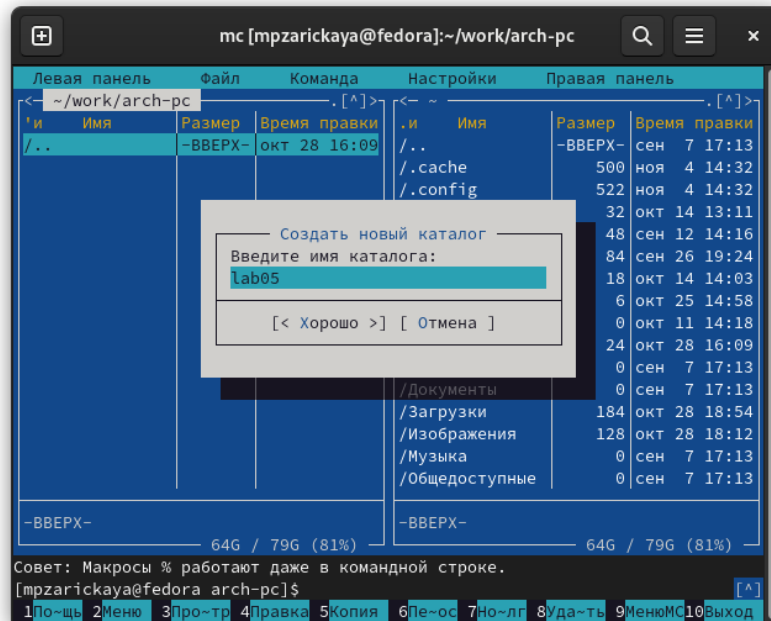


Рис. 4.2: Создание каталога

Пользуясь строкой ввода и командой touch создаю файл lab5-1.asm (рис. 4.3).

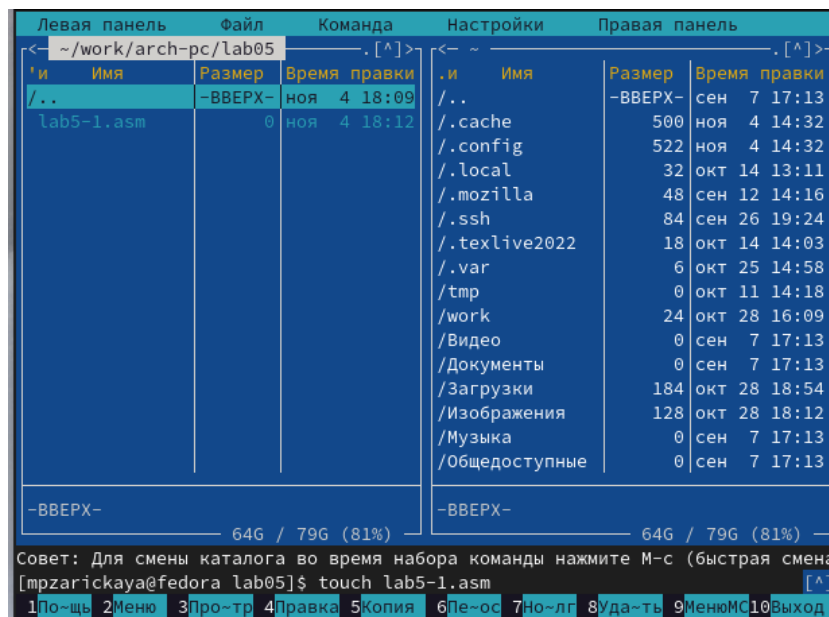
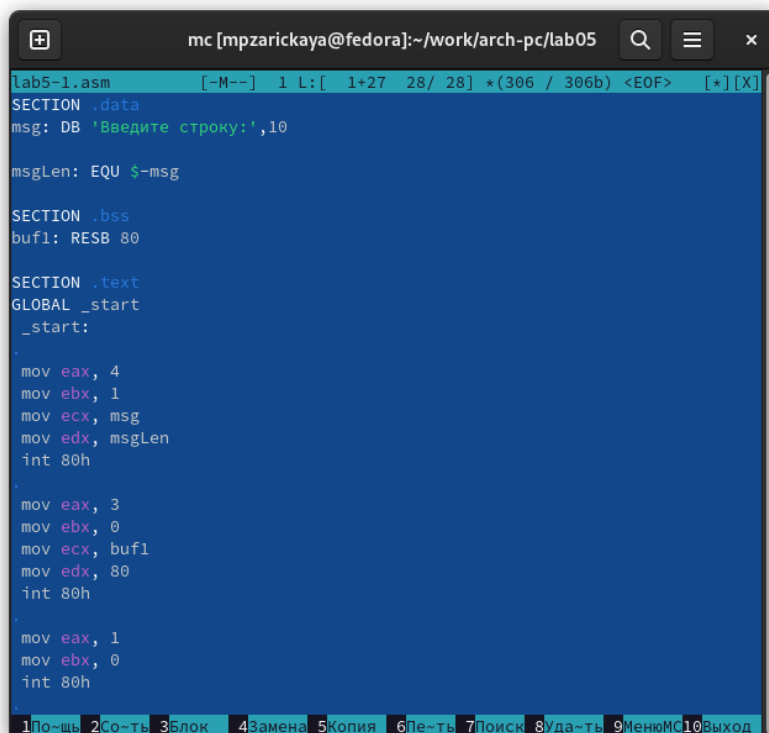


Рис. 4.3: Создание файла lab5-1.asm

## 4.2 Структура программы на языке ассемблера NASM

С помощью функциональной клавиши F4 открываю созданный файл для редактирования в редакторе mcedit. Ввожу в файл код программы вывода сообщения на экран и ввода строки с клавиатуры, сохраняю изменения и закрываю файл (рис. 4.4).



```
lab5-1.asm [-M--] 1 L: [ 1+27 28/ 28] *(306 / 306b) <EOF> [*][X]
SECTION .data
msg: DB 'Введите строку:',10

msgLen: EQU $-msg

SECTION .bss
buf1: RESB 80

SECTION .text
GLOBAL _start
_start:

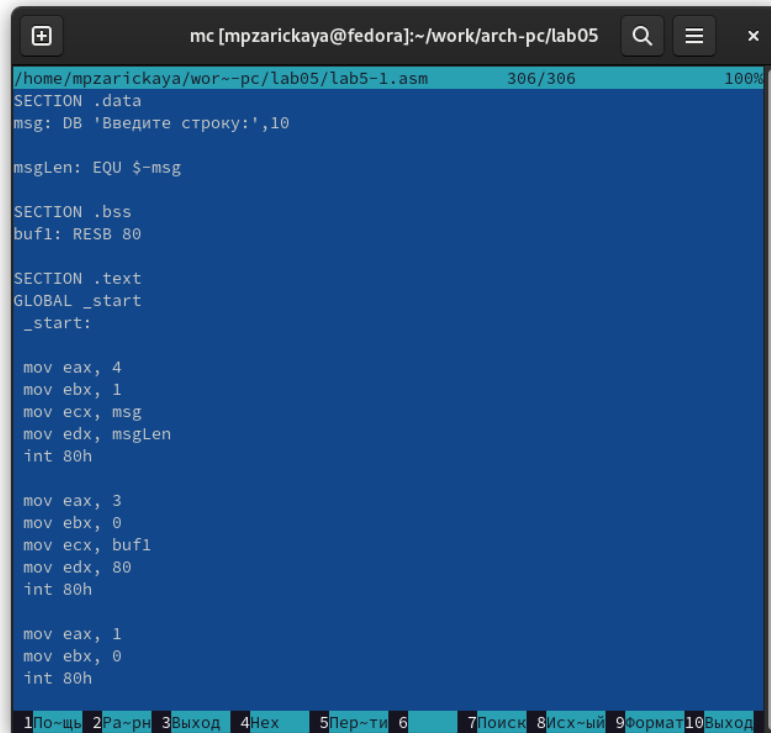
    mov eax, 4
    mov ebx, 1
    mov ecx, msg
    mov edx, msgLen
    int 80h

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

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

Рис. 4.4: Редактирование файла

С помощью функциональной клавиши F3 открываю файл для просмотра, чтобы проверить, содержит ли файл текст программы (рис. 4.5).



The screenshot shows a text editor window titled 'mc [mpzarickaya@fedora]:~/work/arch-pc/lab05'. The editor displays the contents of the file 'lab5-1.asm'. The code is as follows:

```
/home/mpzarickaya/work/arch-pc/lab05/lab5-1.asm 306/306 100%
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, 1
    mov ebx, 0
    int 80h
```

At the bottom of the window, there is a menu bar with the following items: 1По-щ, 2Па-рн, 3Выход, 4Нех, 5Пер-ти, 6, 7Поиск, 8Исх-ый, 9Формат, 10Выход.

Рис. 4.5: Открытие файла для просмотра

Транслирую текст программы файла в объектный файл командой `nasm -f elf lab5-1.asm` (рис. 4.6).

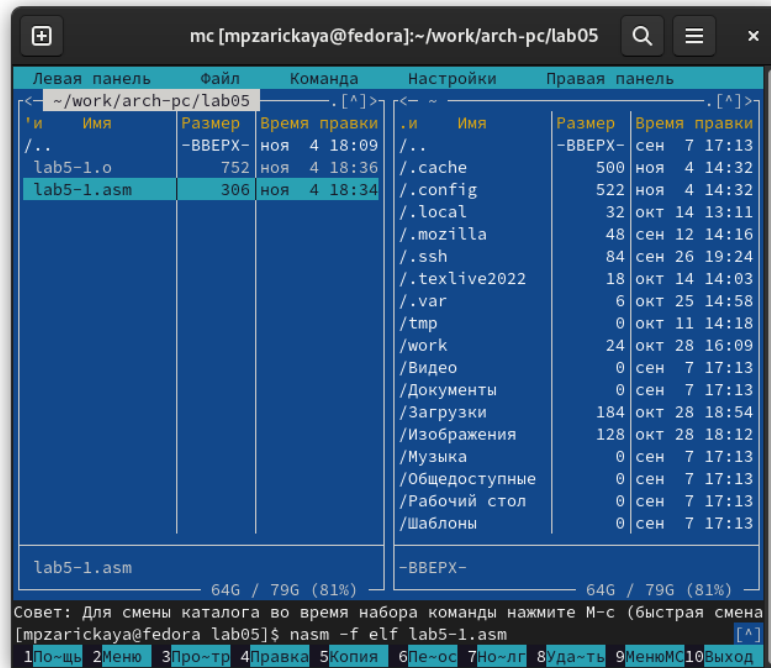


Рис. 4.6: Создание объектного файла lab5-1.o

Выполняю компоновку объектного файла (рис. 4.7).

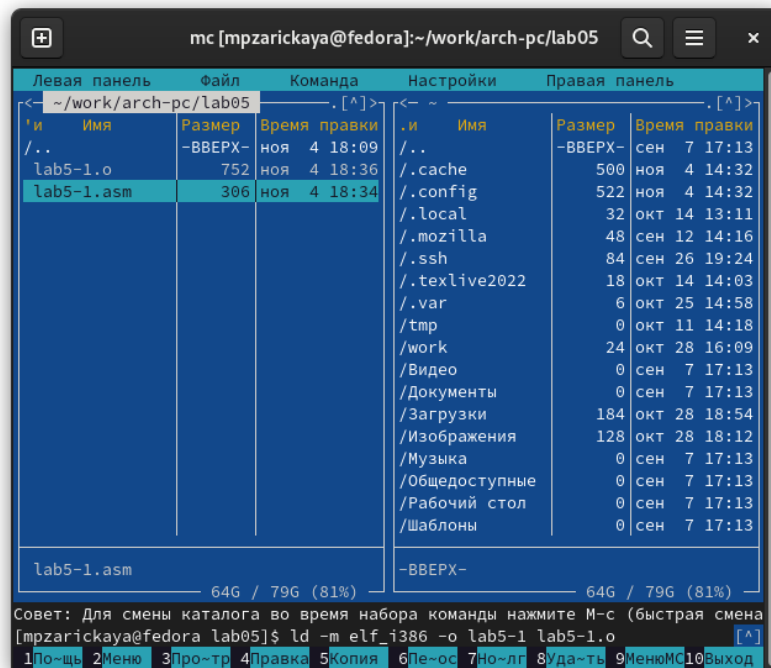


Рис. 4.7: Компоновка объектного файла

Запускаю исполняемый файл. Программа выводит строку “Введите строку:” и ждет ввода с клавиатуры, я ввожу свои ФИО и программа заканчивает свою работу (рис. 4.8).

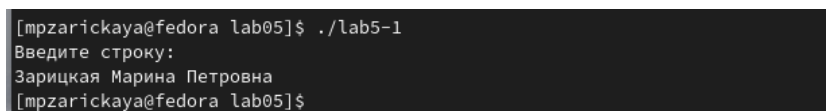


Рис. 4.8: Запуск программы

### 4.3 Подключение внешнего файла in\_out.asm

Скачиваю файл in\_out.asm со страницы курса в ТУИС. С помощью функциональной клавиши F5 копирую файл из каталога Загрузки в созданный каталог lab05 (рис. 4.9).

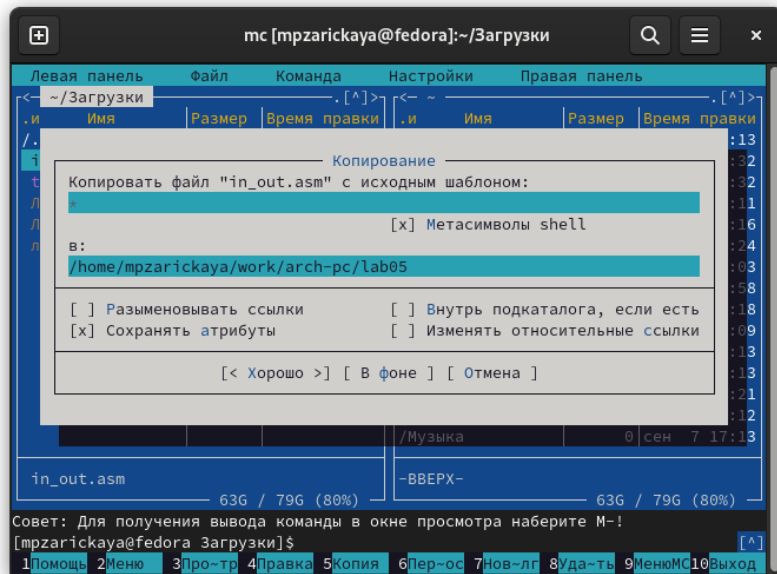


Рис. 4.9: Копирование файла

С помощью функциональной клавиши F5 создаю копию файла lab5-1.asm с именем lab5-2.asm (рис. 4.10).

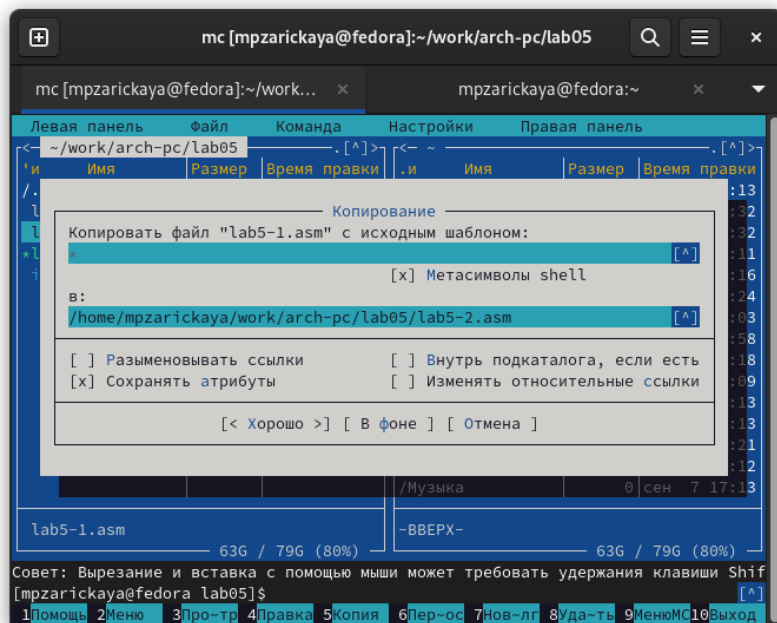
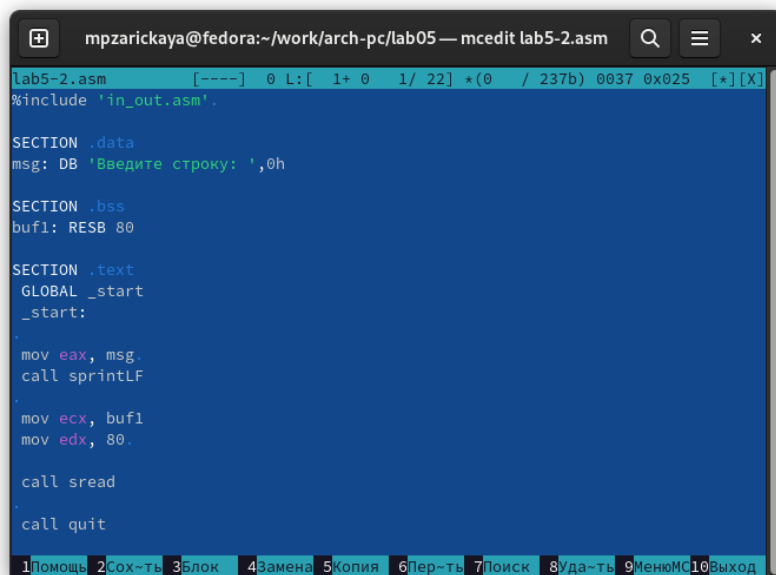


Рис. 4.10: Создание копии файла

Изменяю содержимое файла lab5-2.asm во встроенном редакторе mcedit так, чтобы в программе использовались подпрограммы из внешнего файла in\_out.asm (рис. 4.11).



```
lab5-2.asm [----] 0 L: [ 1+ 0 1/ 22] *(0 / 237b) 0037 0x025 [*][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 sprintf

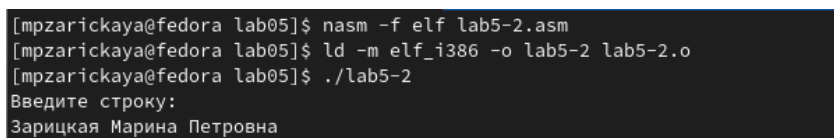
mov ecx, buf1
mov edx, 80

call sread

call quit
```

Рис. 4.11: Изменение 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 (рис. 4.12).

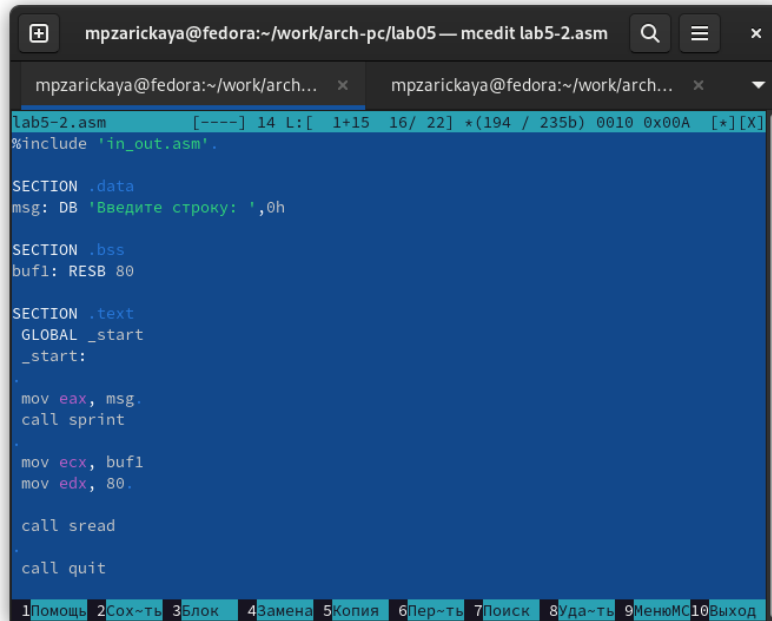


```
[mpzarickaya@fedora lab05]$ nasm -f elf lab5-2.asm
[mpzarickaya@fedora lab05]$ ld -m elf_i386 -o lab5-2 lab5-2.o
[mpzarickaya@fedora lab05]$ ./lab5-2
Введите строку:
Зарицкая Марина Петровна
```

Рис. 4.12: Исполнение файла

Изменяю в файле lab5-2.asm подпрограмму `sprintf` на `sprint`. Сохраняю изменения и открываю файл для просмотра, чтобы проверить сохранение действий (рис. 4.13).





```
mpzarickaya@fedora:~/work/arch-pc/lab05 — mcedit lab5-2.asm
lab5-2.asm [----] 14 L: [ 1+15 16/ 22] *(194 / 235b) 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 sprint

mov ecx, buf1
mov edx, 80

call sread

call quit

1Помощь 2Сох-ть 3Блок 4Замена 5Копия 6Пер-ть 7Поиск 8Уда-ть 9Меню 10Выход
```

Рис. 4.13: Отредактированный файл

Транслирую файл, выполняю компоновку созданного объектного файла, запускаю программу (рис. 4.14).

```
[mpzarickaya@fedora lab05]$ nasm -f elf lab5-2.asm
[mpzarickaya@fedora lab05]$ ld -m elf_i386 -o lab5-2 lab5-2.o
[mpzarickaya@fedora lab05]$ ./lab5-2
Введите строку: Зарицкая Марина Петровна
```

Рис. 4.14: Исполнение файла

Различие между первым и вторым исполняемыми файлами в том, что запуск первого запрашивает ввод с новой строки, а второго запрашивает ввод без переноса на новую строку. В этом и заключается разница между подпрограммами `sprintLF` и `sprint`.

## 4.4 Задание для самостоятельной работы

1. Создаю копию файла lab5-1.asm с именем lab5-1-1.asm с помощью функциональной клавиши F5 (рис. 4.15).

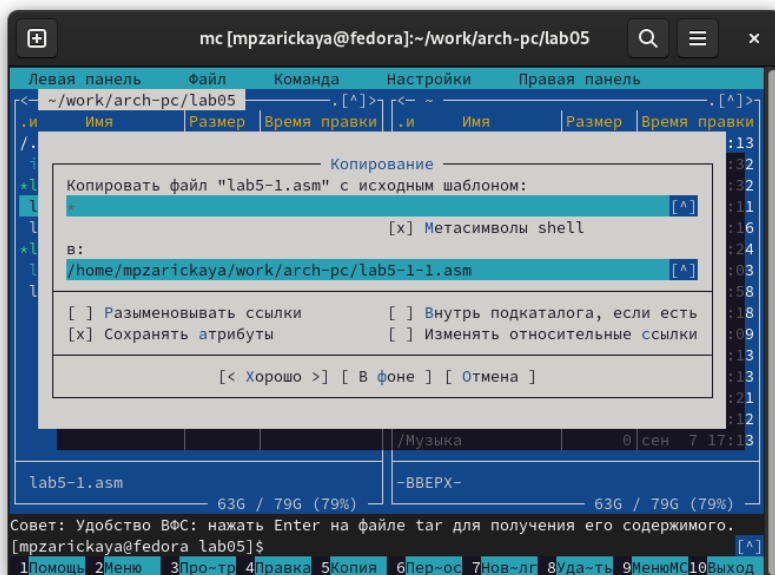
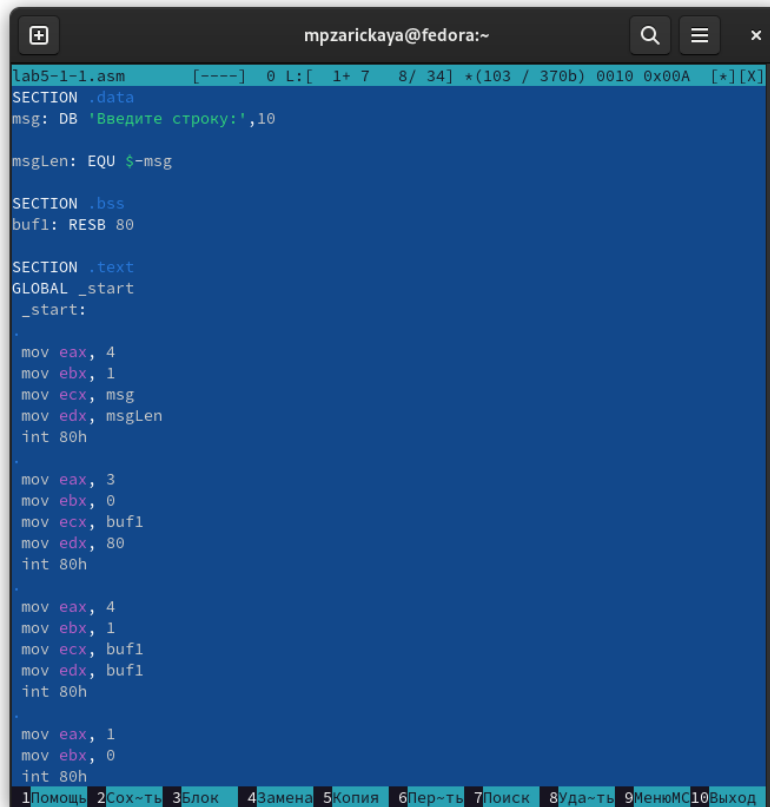


Рис. 4.15: Копирование файла

Открываю созданный файл в mcedit для редактирования. Изменяю программу так, чтобы кроме вывода приглашения и запроса ввода, она выводила вводимую пользователем строку (рис. 4.16).



```
lab5-1-1.asm [----] 0 L: [ 1+ 7 8/ 34] *(103 / 370b) 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, 3
mov ebx, 0
mov ecx, buf1
mov edx, 80
int 80h

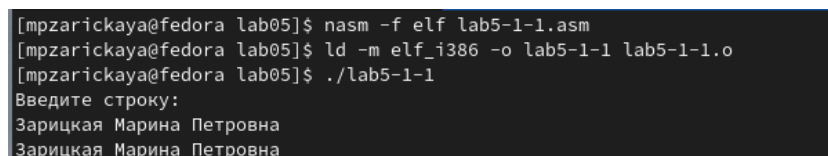
mov eax, 4
mov ebx, 1
mov ecx, buf1
mov edx, buf1
int 80h

mov eax, 1
mov ebx, 0
int 80h

1Помощь 2Сох-ть 3Блок 4Замена 5Копия 6Пер-ть 7Поиск 8Уда-ть 9МенюMC10Выход
```

Рис. 4.16: 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 (рис. 4.17).



```
[mpzarickaya@fedora lab05]$ nasm -f elf lab5-1-1.asm
[mpzarickaya@fedora lab05]$ ld -m elf_i386 -o lab5-1-1 lab5-1-1.o
[mpzarickaya@fedora lab05]$ ./lab5-1-1
Введите строку:
Зарицкая Марина Петровна
Зарицкая Марина Петровна
```

Рис. 4.17: Исполнение файла

Листинг первой 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

mov ebx, 1

mov ecx, buf1

mov edx, buf1

int 80h

```

mov eax, 1
mov ebx, 0
int 80h

```

3. Создаю копию файла lab5-2.asm с именем lab5-2-1.asm с помощью функциональной клавиши F5 (рис. 4.18).

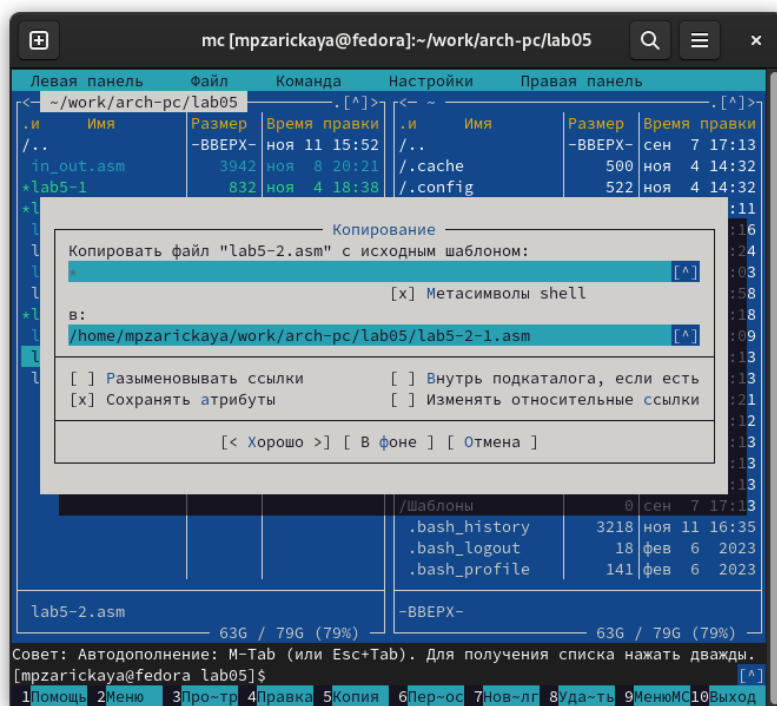
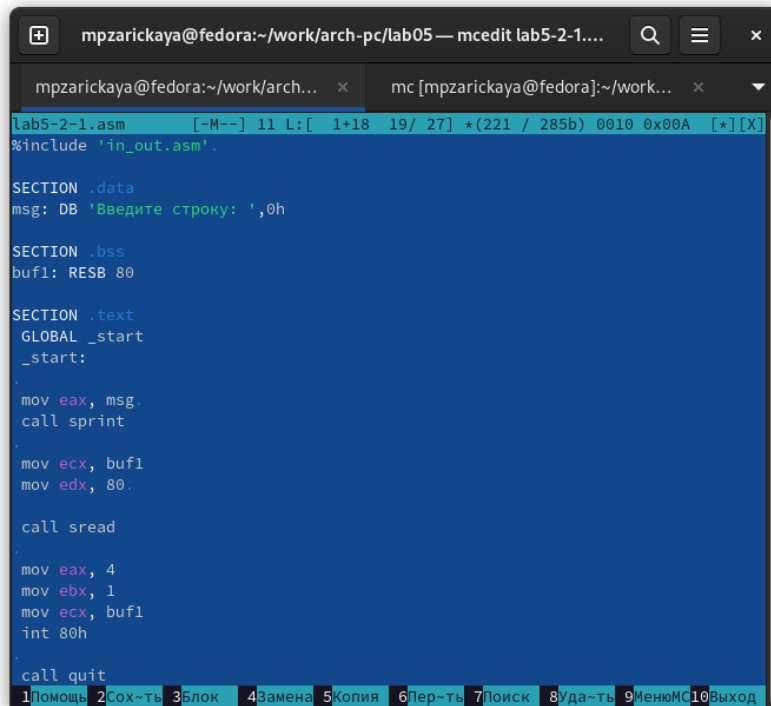


Рис. 4.18: Копирование файла

Открываю созданный файл в mcedit для редактирования. Изменяю программу так, чтобы кроме вывода приглашения и запроса ввода, она выводила вводимую пользователем строку (рис. 4.19).



```
lab5-2-1.asm [-M--] 11 L: [ 1+18 19/ 27] *(221 / 285b) 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 sprint

mov ecx, buf1
mov edx, 80

call sread

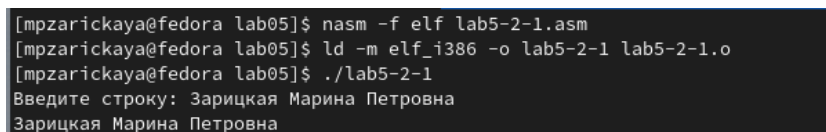
mov eax, 4
mov ebx, 1
mov ecx, buf1
int 80h

call quit

1Помощь 2Сох-ть 3Блок 4Замена 5Копия 6Пер-ть 7Поиск 8Уда-ть 9Меню 10Выход
```

Рис. 4.19: Редактирование 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 (рис. 4.20).



```
[mpzarickaya@fedora lab05]$ nasm -f elf lab5-2-1.asm
[mpzarickaya@fedora lab05]$ ld -m elf_i386 -o lab5-2-1 lab5-2-1.o
[mpzarickaya@fedora lab05]$ ./lab5-2-1
Введите строку: Зарицкая Марина Петровна
Зарицкая Марина Петровна
```

Рис. 4.20: Исполнение файла

Листинг второй 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
```

```
mov ebx, 1
```

```
mov ecx, buf1
```

```
int 80h
```

```
call quit
```

## 5 Выводы

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