

# **Отчёт по лабораторной работе №5**

**Дисциплина: Архитектура компьютера**

Филатов Илья Гурамович

# Содержание

<b>1</b>	<b>Цель работы</b>	<b>4</b>
<b>2</b>	<b>Задание</b>	<b>5</b>
<b>3</b>	<b>Теоретическое введение</b>	<b>6</b>
<b>4</b>	<b>Выполнение лабораторной работы</b>	<b>7</b>
4.1	Работа с Midnight Commander . . . . .	7
4.2	Подключение внешнего файла in_out.asm . . . . .	11
4.3	Задание для самостоятельной работы . . . . .	14
<b>5</b>	<b>Выводы</b>	<b>21</b>
<b>6</b>	<b>Список литературы</b>	<b>22</b>

# Список иллюстраций

4.1	Переход в Midnight Commander . . . . .	7
4.2	Переход в каталог . . . . .	8
4.3	Создание каталога . . . . .	8
4.4	Создание файла . . . . .	9
4.5	Редактирование файла . . . . .	9
4.6	Создание объектного кода . . . . .	10
4.7	Проверка файла . . . . .	10
4.8	Создание и запуск исполняемого файла . . . . .	11
4.9	Скачивание файла . . . . .	11
4.10	Копирование файла . . . . .	12
4.11	Копирование файла с новым именем . . . . .	12
4.12	Изменение текста программы . . . . .	13
4.13	Создание и запуск исполняемого файла . . . . .	13
4.14	Замена подпрограммы . . . . .	14
4.15	Создание и запуск исполняемого файла . . . . .	14
4.16	Создание копии файла . . . . .	15
4.17	Преобразование текста программы . . . . .	15
4.18	Создание и проверка работы файла . . . . .	17
4.19	Запуск файла . . . . .	18
4.20	Преобразование текста программы . . . . .	18
4.21	Создание и проверка работы файла . . . . .	20

# 1 Цель работы

Цель работы — приобрести практические навыки работы в Midnight Commander и освоить инструкции языка ассемблера `mov` и `int`.

## 2 Задание

1. Работа с Midnight Commander
2. Подключение внешнего файла in\_out.asm
3. 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).

Инструкция языка ассемблера mov предназначена для дублирования данных источника в приёмнике. В общем виде эта инструкция записывается в виде

```
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. В общем виде она записывается в виде

```
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 Работа с Midnight Commander

Открываю терминал. Ввожу команду `mc` для открытия Midnight Commander (рис. 4.1).

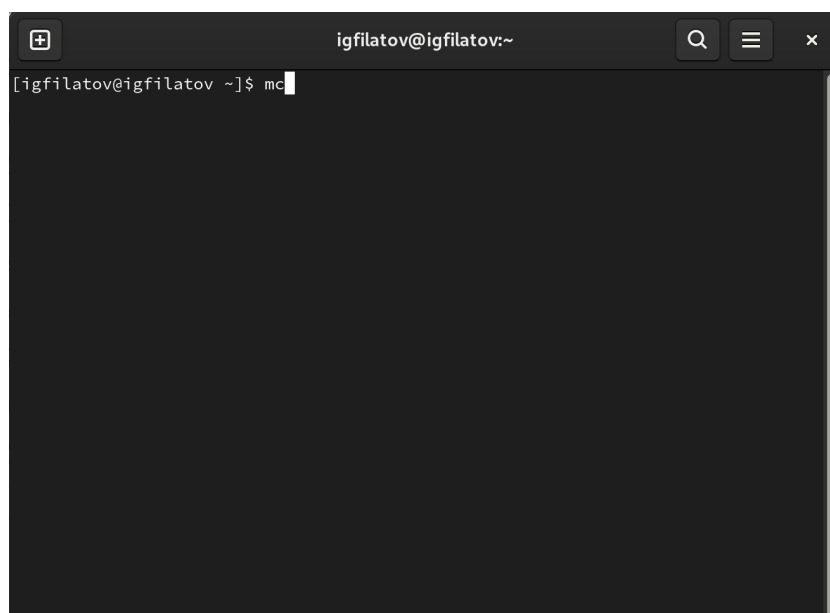


Рис. 4.1: Переход в Midnight Commander

Пользуясь клавишами `↑`, `↓` и `Enter` перехожу в каталог `~/work/arch-рс` (рис. 4.2).

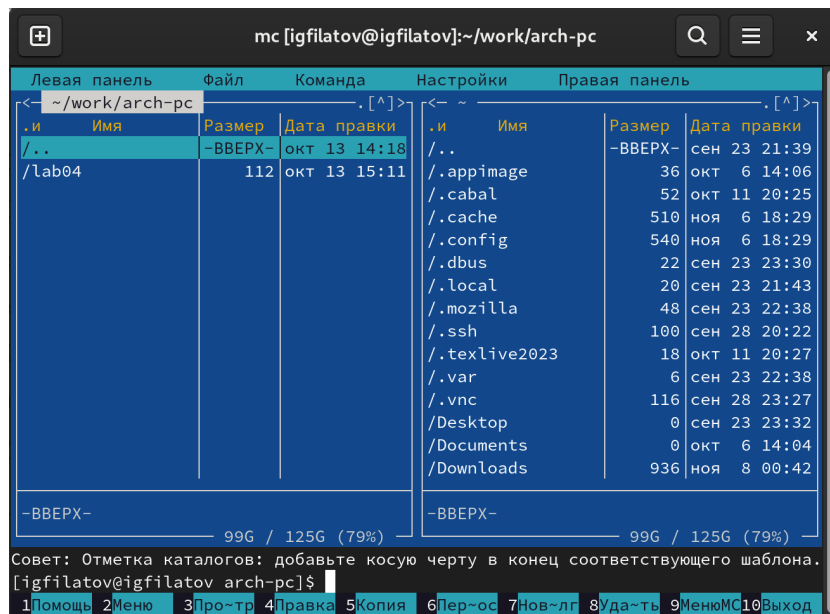


Рис. 4.2: Переход в каталог

С помощью клавиши F7 создаю папку lab05 и перехожу в созданный каталог (рис. 4.3).

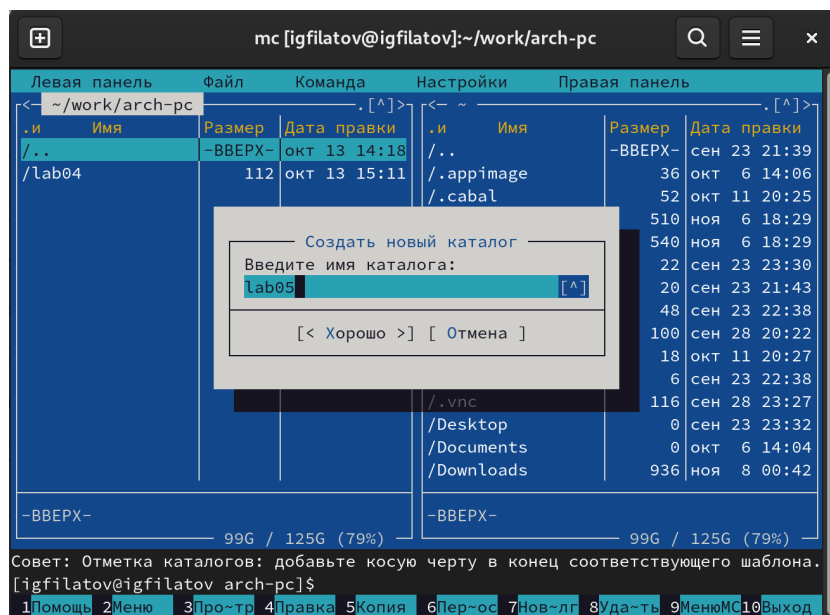


Рис. 4.3: Создание каталога

Пользуясь строкой ввода и командой touch создаю файл lab5-1.asm (рис. 4.4).





```

lab5-1.asm      [-M--] 20 L:[ 14+21  35/ 35] *(2431/2431b) <E0F>      [*][X]
_start: ; Точка входа в программу
;----- Системный вызов 'write'
; После вызова инструкции 'int 80h' на экран будет
; выведено сообщение из переменной 'msg' длиной 'msgLen'
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
;----- системный вызов 'read' -----
; После вызова инструкции 'int 80h' программа будет ожидать ввода
; строки, которая будет записана в переменную 'buf1' размером 80 байт
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
;----- Системный вызов 'exit' -----
; После вызова инструкции 'int 80h' программа завершит работу
mov eax,1 ; Системный вызов для выхода (sys_exit)
mov ebx,0 ; Выход с кодом возврата 0 (без ошибок)
int 80h ; Вызов ядра
1Помощь 2Сохран 3Блок 4Замена 5Копия 6Пер-ть 7Поиск 8Уда-ть 9МенюМС10Выход

```

Рис. 4.6: Создание объектного кода

С помощью клавиши F3 открываю файл lab5-1.asm для просмотра и убеждаюсь, что он содержит текст программы (рис. 4.7).

```

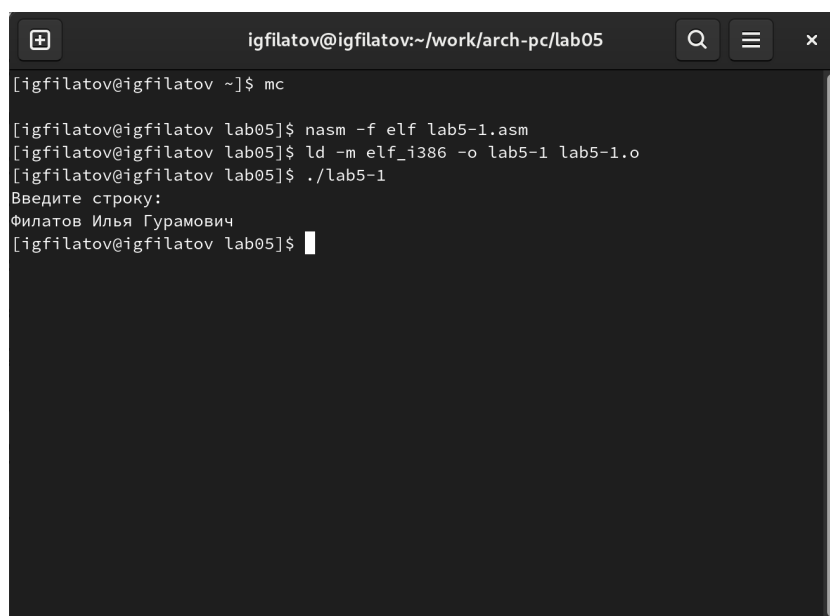
/home/igfilatov/work/arch-pc/lab05/lab5-1.asm      1448/2431      59%
;-----
; Программа вывода сообщения на экран и ввода строки с клавиатуры
;-----
;----- Объявление переменных -----
SECTION .data ; Секция иницированных данных
msg: DB 'Введите строку:',10 ; сообщение плюс
; символ перевода строки
msgLen: EQU $-msg ; Длина переменной 'msg'
SECTION .bss ; Секция не иницированных данных
buf1: RESB 80 ; Буфер размером 80 байт
;----- Текст программы -----
SECTION .text ; Код программы
GLOBAL _start ; Начало программы
_start: ; Точка входа в программу
;----- Системный вызов 'write'
; После вызова инструкции 'int 80h' на экран будет
; выведено сообщение из переменной 'msg' длиной 'msgLen'
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
1Помощь 2Раз-рн 3Выход 4Hex 5Пер-ти 6 7Поиск 8Исх-ый 9Формат10Выход

```

Рис. 4.7: Проверка файла

Сочетанием клавиш Ctrl+o скрываю панели Midnight Commander. Создаю объектный файл из программы lab5-1.asm. Выполняю компоновку и запускаю полу-

чившийся исполняемый файл. На запрос ввожу своё ФИО (рис. 4.8).



```
igfilatov@igfilatov:~/work/arch-pc/lab05
[igfilatov@igfilatov ~]$ mc
[igfilatov@igfilatov lab05]$ nasm -f elf lab5-1.asm
[igfilatov@igfilatov lab05]$ ld -m elf_i386 -o lab5-1 lab5-1.o
[igfilatov@igfilatov lab05]$ ./lab5-1
Введите строку:
Филатов Илья Гурамович
[igfilatov@igfilatov lab05]$
```

Рис. 4.8: Создание и запуск исполняемого файла

## 4.2 Подключение внешнего файла in\_out.asm

Скачиваю файл in\_out.asm со страницы курса в ТУИС (рис. 4.9).

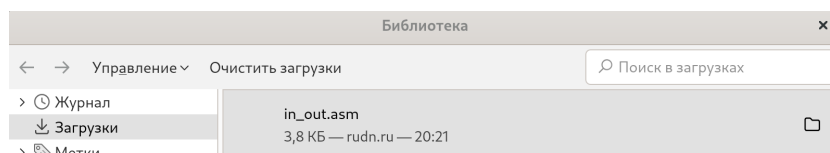


Рис. 4.9: Скачивание файла

В одной из панелей mc открываю каталог с файлом lab5-1.asm, а в другой — каталог со скаченным файлом in\_out.asm. Копирую файл in\_out.asm клавишей F5 (рис. 4.10).

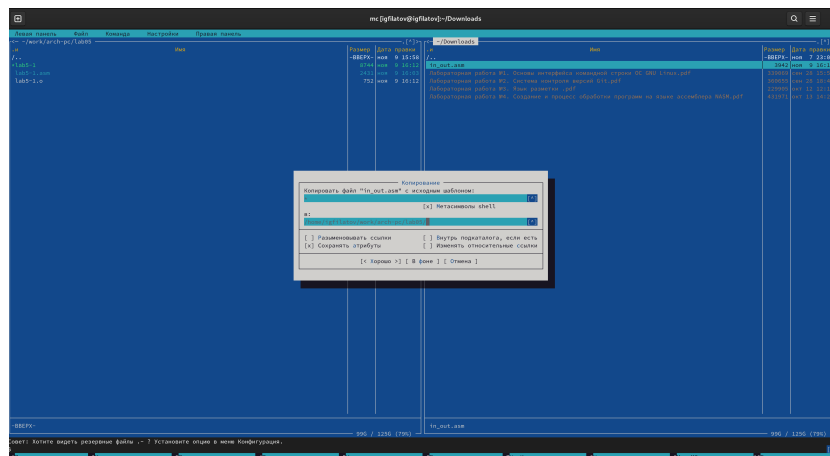


Рис. 4.10: Копирование файла

С помощью клавиши F5 создаю копию файла lab5-1.asm с именем lab5-2.asm. (рис. 4.11).

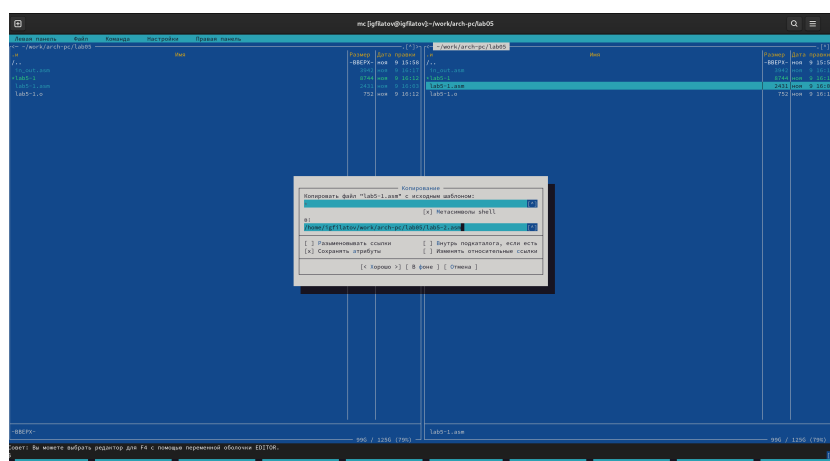


Рис. 4.11: Копирование файла с новым именем

Исправляю текст программы в файле lab5-2.asm и сохраняю изменения клавишей F2 (рис. 4.12).

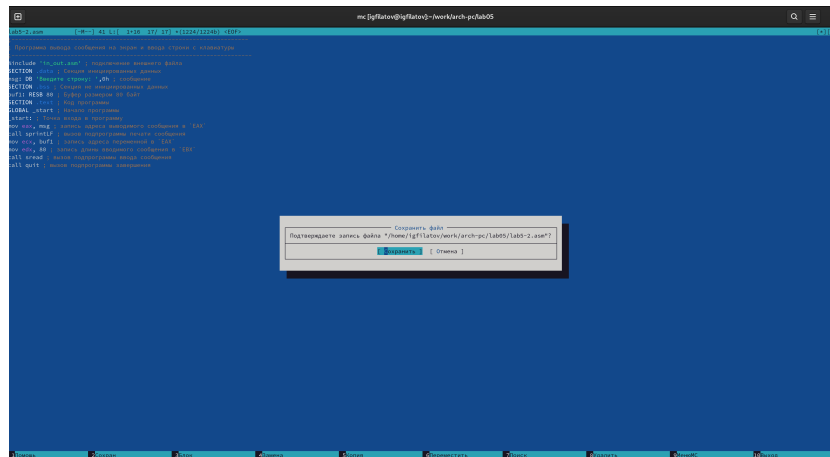


Рис. 4.12: Изменение текста программы

Создаю исполняемый файл и проверяю его работу, вводя на запрос своё ФИО (рис. 4.13).

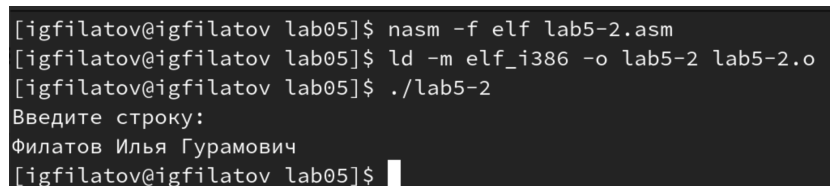
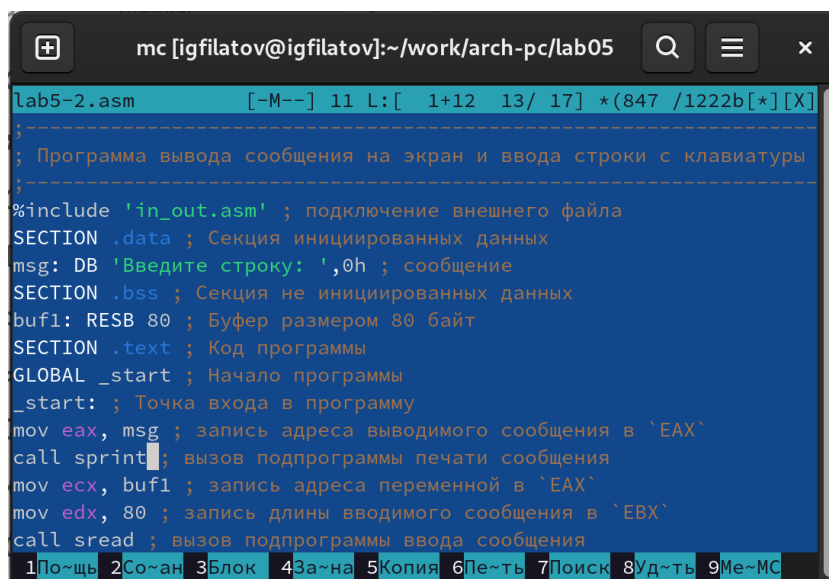


Рис. 4.13: Создание и запуск исполняемого файла

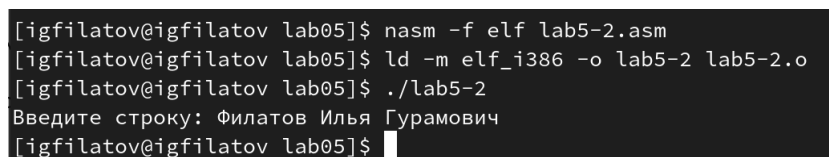
В файле lab5-2.asm заменяю подпрограмму sprintLF на sprint и сохраняю изменения (рис. 4.14).



```
lab5-2.asm [-M--] 11 L: [ 1+12 13/ 17] *(847 /1222b[*] [X]
;
; Программа вывода сообщения на экран и ввода строки с клавиатуры
;
-----
%include 'in_out.asm' ; подключение внешнего файла
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
```

Рис. 4.14: Замена подпрограммы

Создаю и запускаю исполняемый файл. В отличие от предыдущей версии он предлагает ввести сообщение на той же строке, на которой выводит текст, а не на следующей, из-за замены команды (рис. 4.15).



```
[igfilatov@igfilatov lab05]$ nasm -f elf lab5-2.asm
[igfilatov@igfilatov lab05]$ ld -m elf_i386 -o lab5-2 lab5-2.o
[igfilatov@igfilatov lab05]$ ./lab5-2
Введите строку: Филатов Илья Гурамович
[igfilatov@igfilatov lab05]$
```

Рис. 4.15: Создание и запуск исполняемого файла

## 4.3 Задание для самостоятельной работы

Создаю копию файла lab5-1.asm с именем lab5-3.asm (рис. 4.16).

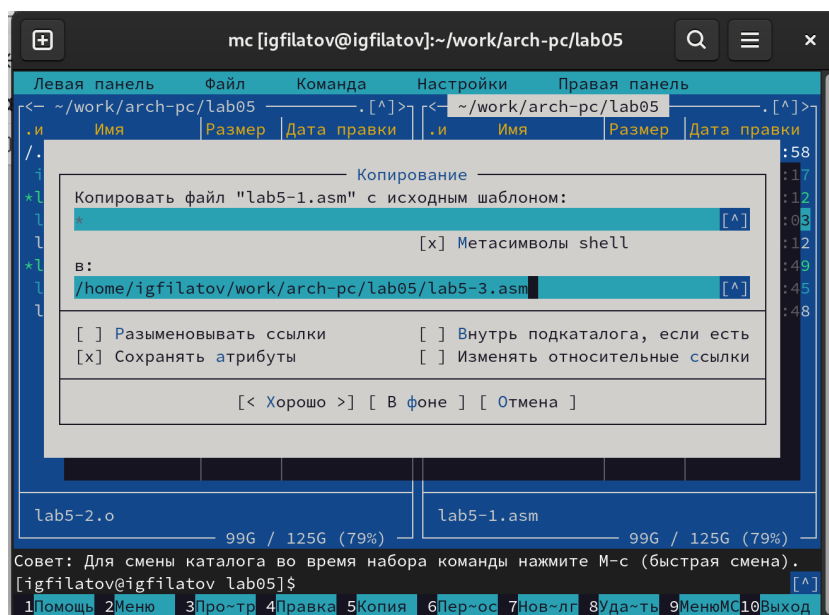


Рис. 4.16: Создание копии файла

Преобразовываю текст программы так, чтобы на экран выводилась введенная строка. Для этого после блока “Системный вызов read” добавляю команды из блока “Системный вызов write”, указывая буфер для его вывода. Сохраняю изменения (рис. 4.17).

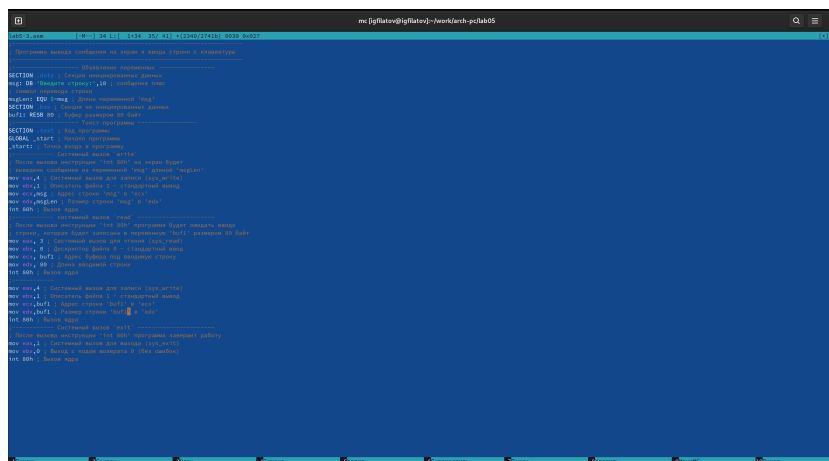


Рис. 4.17: Преобразование текста программы

Преобразованный текст программы:

;-----

```

; Программа вывода сообщения на экран и ввода строки с клавиатуры
;-----
;----- Объявление переменных -----
SECTION .data ; Секция инициированных данных
msg: DB 'Введите строку:',10 ; сообщение плюс
; символ перевода строки
msgLen: EQU $-msg ; Длина переменной 'msg'
SECTION .bss ; Секция не инициированных данных
buf1: RESB 80 ; Буфер размером 80 байт
;----- Текст программы -----
SECTION .text ; Код программы
GLOBAL _start ; Начало программы
_start: ; Точка входа в программу
;----- Системный вызов write
; После вызова инструкции 'int 80h' на экран будет
; выведено сообщение из переменной 'msg' длиной 'msgLen'
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
;----- системный вызов read -----
; После вызова инструкции 'int 80h' программа будет ожидать ввода
; строки, которая будет записана в переменную 'buf1' размером 80 байт
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

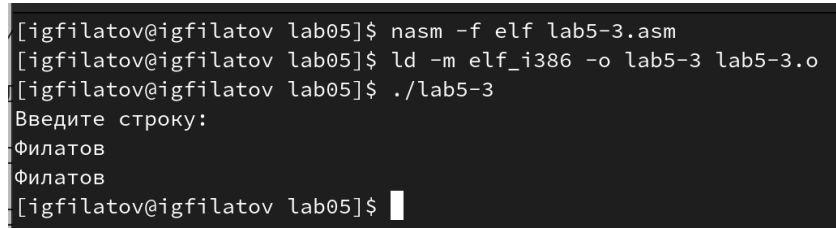
```



```

;-----
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' в 'edx'
int 80h ; Вызов ядра
;----- Системный вызов exit -----
; После вызова инструкции 'int 80h' программа завершит работу
mov eax,1 ; Системный вызов для выхода (sys_exit)
mov ebx,0 ; Выход с кодом возврата 0 (без ошибок)
int 80h ; Вызов ядра
Создаю исполняемый файл и проверяю его работу (рис. 4.18).

```



```

[igfilatov@igfilatov lab05]$ nasm -f elf lab5-3.asm
[igfilatov@igfilatov lab05]$ ld -m elf_i386 -o lab5-3 lab5-3.o
[igfilatov@igfilatov lab05]$ ./lab5-3
Введите строку:
Филатов
Филатов
[igfilatov@igfilatov lab05]$

```

Рис. 4.18: Создание и проверка работы файла

Создаю копию файла lab5-2.asm с именем lab5-4.asm (рис. 4.19).

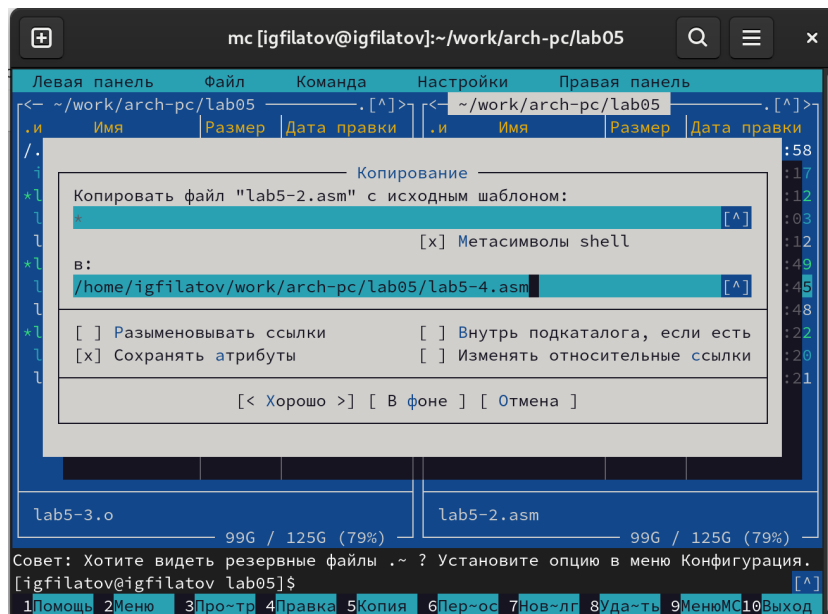


Рис. 4.19: Запуск файла

Преобразовываю текст программы так, чтобы на экран выводилась введенная строка. Для этого использую подпрограмму `sprintLF` из `in_out.asm` для печати сообщения. Указываю буфер для его вывода и сохраняю файл (рис. 4.20).

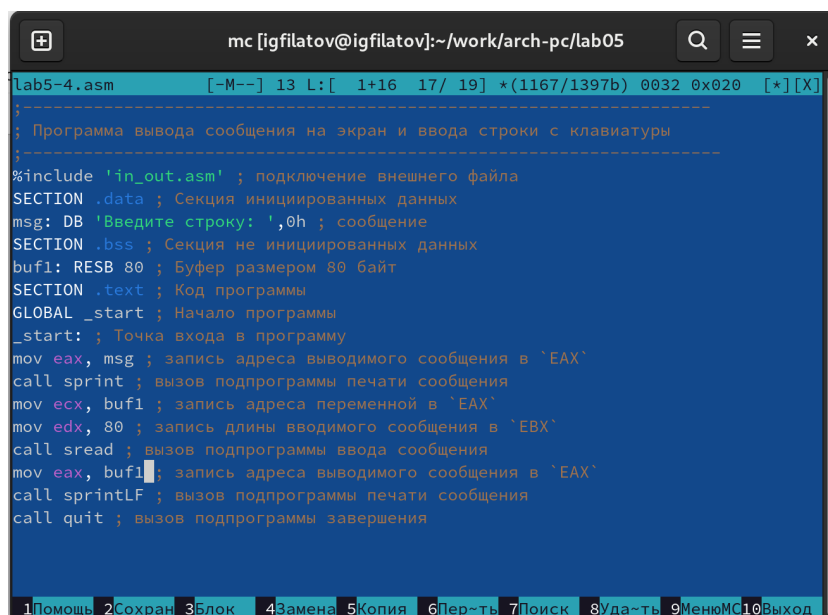


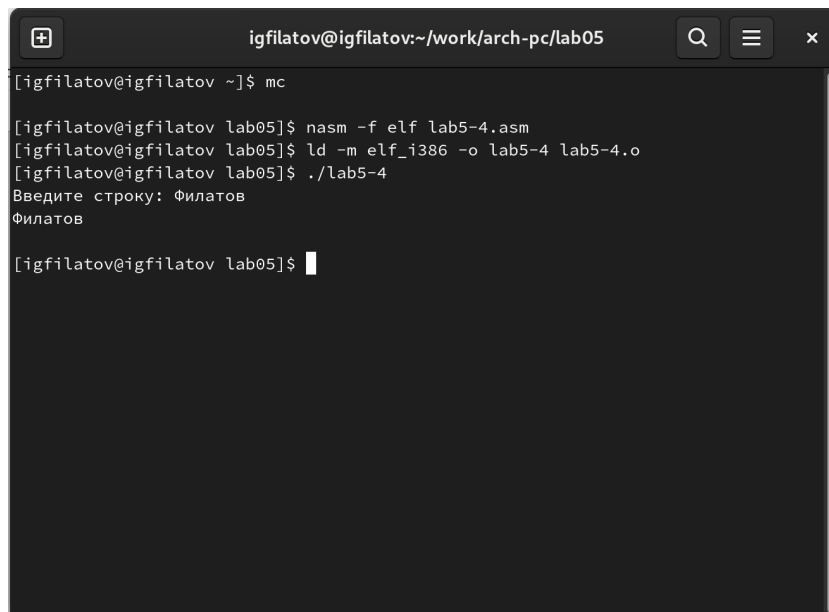
Рис. 4.20: Преобразование текста программы

Преобразованный текст программы:

```

;-----
; Программа вывода сообщения на экран и ввода строки с клавиатуры
;-----
%include 'in_out.asm' ; подключение внешнего файла
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, buf1 ; запись адреса выводимого сообщения в EAX
call sprintLF ; вызов подпрограммы печати сообщения
call quit ; вызов подпрограммы завершения
Создаю исполняемый файл и проверяю его работу (рис. 4.21).

```

A terminal window with a dark background and light gray text. The window title bar shows the user 'igfilatov' at host 'igfilatov' in the directory '~/work/arch-pc/lab05'. The terminal content shows a sequence of commands: 'mc' is run in the home directory; then in the 'lab05' directory, 'nasm -f elf lab5-4.asm' is executed, followed by 'ld -m elf\_i386 -o lab5-4 lab5-4.o'. Then './lab5-4' is run, which prompts for a password 'Филатов'. Finally, the prompt returns to the 'lab05' directory.

```
igfilatov@igfilatov:~/work/arch-pc/lab05
[igfilatov@igfilatov ~]$ mc
[igfilatov@igfilatov lab05]$ nasm -f elf lab5-4.asm
[igfilatov@igfilatov lab05]$ ld -m elf_i386 -o lab5-4 lab5-4.o
[igfilatov@igfilatov lab05]$ ./lab5-4
Введите строку: Филатов
Филатов
[igfilatov@igfilatov lab05]$
```

Рис. 4.21: Создание и проверка работы файла

## 5 Выводы

Я получил практические навыки работы в Midnight Commander и освоил инструкции языка ассемблера `mov` и `int`.

## **6 Список литературы**

1. Архитектура ЭВМ