

Отчет по лабораторной работе №5

Дисциплина: архитектура компьютера

Шабакowa Карина Баировна

Содержание

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 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 инициализированных (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 инициализированных 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 (четырёхбайтное 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. 1).

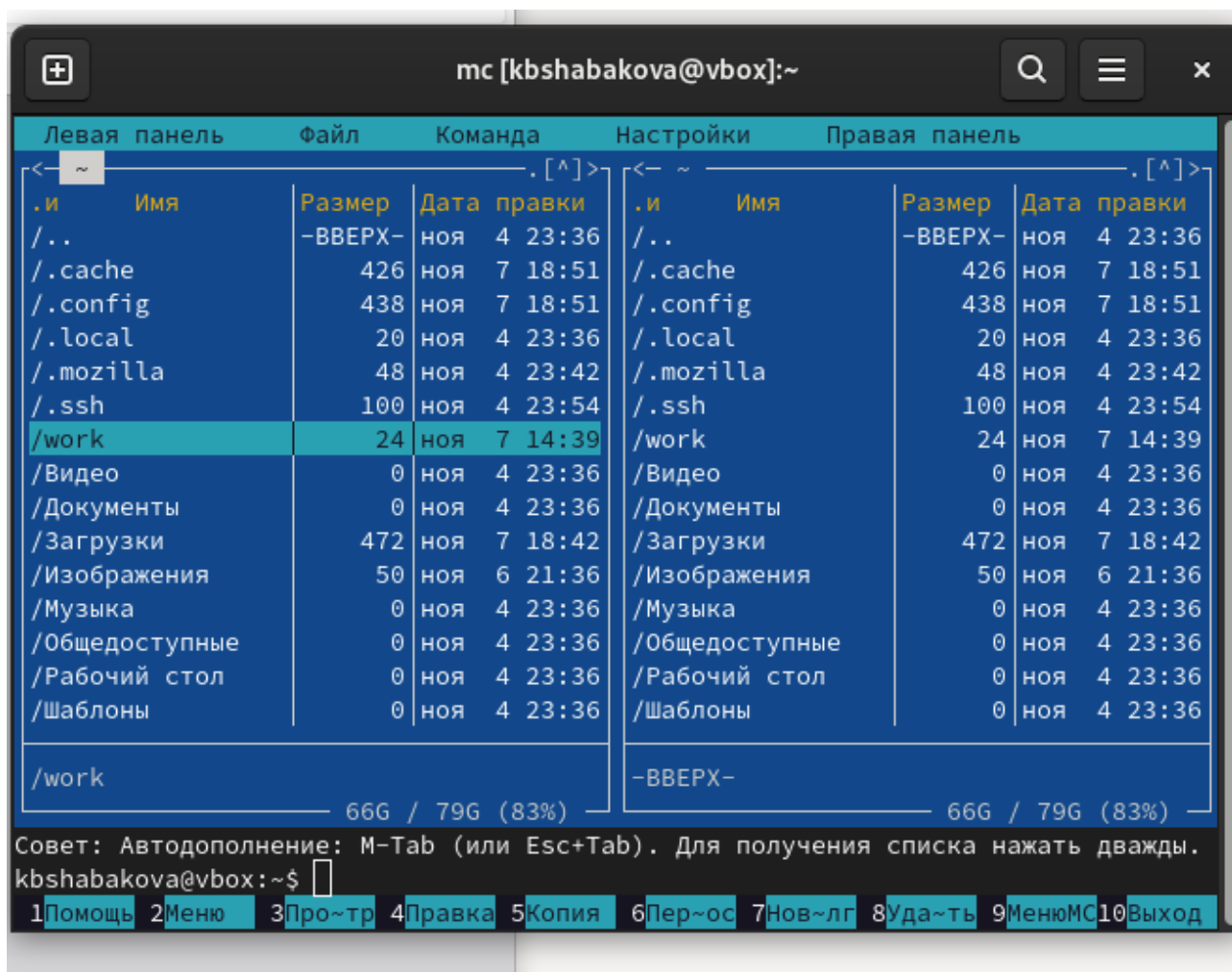


Рис. 1: Открытый mc

С помощью функциональной клавиши F7 создаю каталог lab05 (рис. 2).

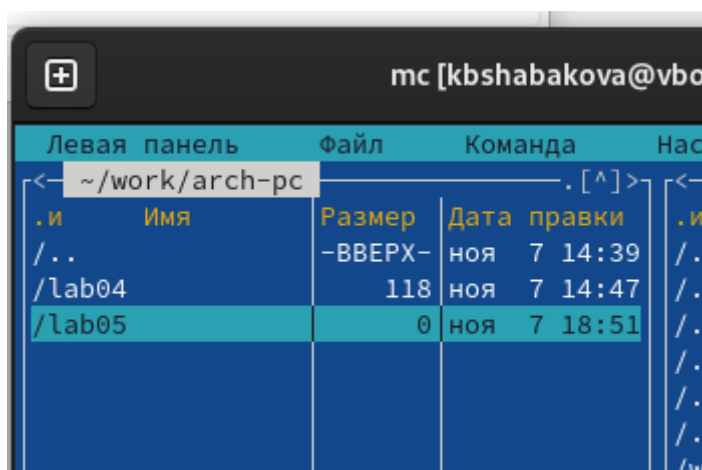


Рис. 2: Создание каталога

В строке ввода прописываю команду `touch lab5-1.asm`, чтобы создать файл, в котором буду работать (рис. 3).

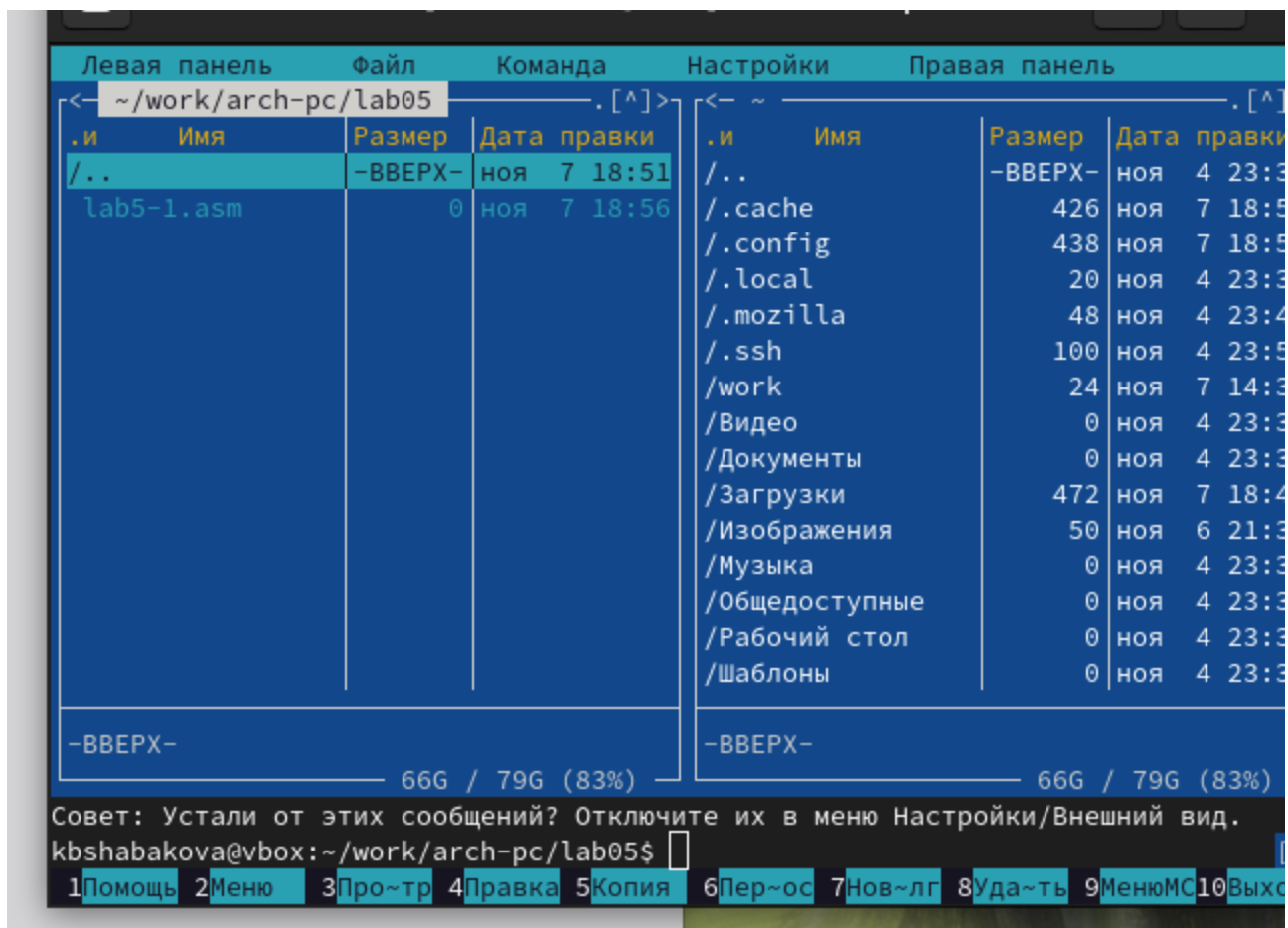
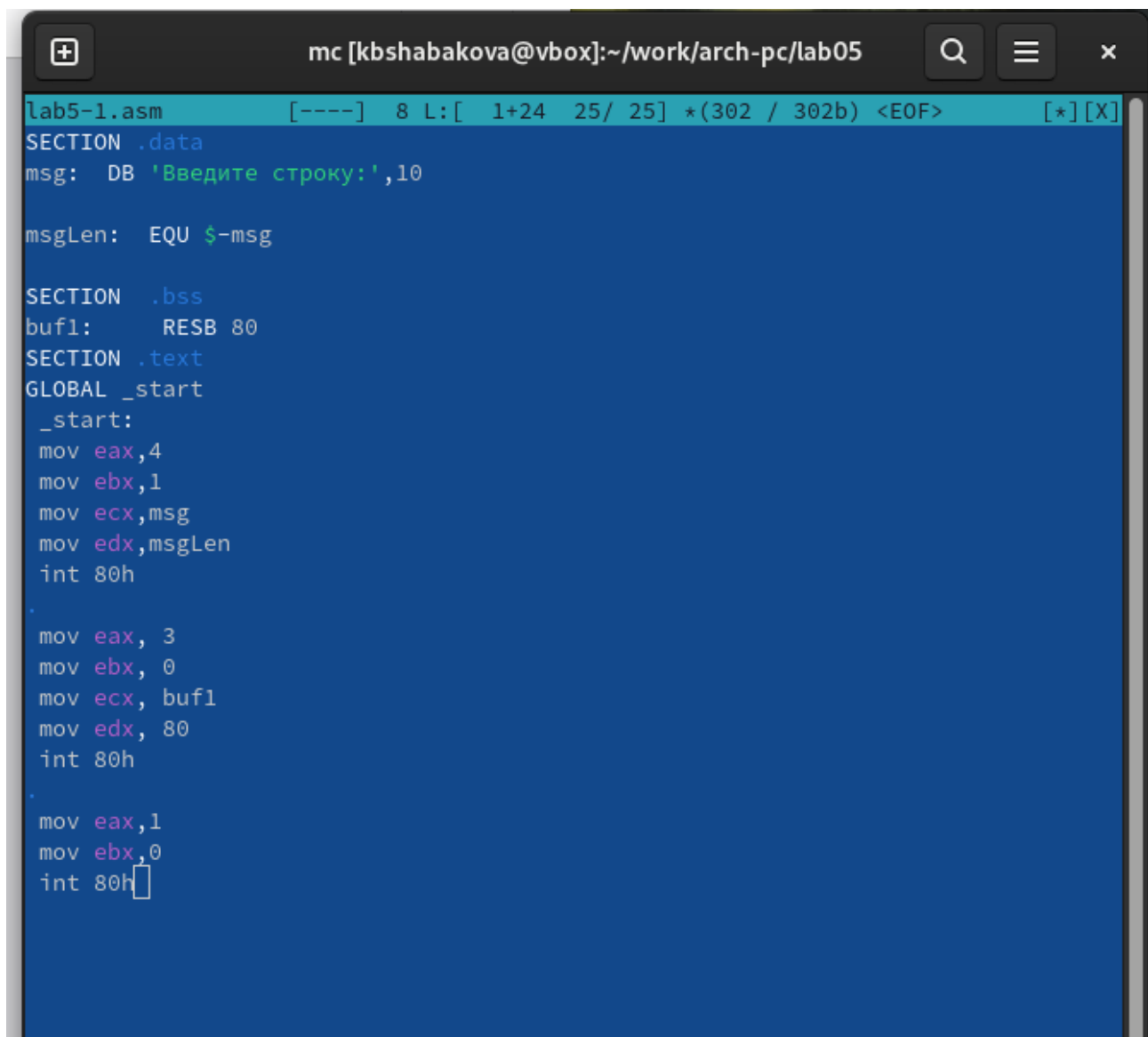


Рис. 3: Создание файла

4.2 Структура программы на языке ассемблера NASM

С помощью функциональной клавиши F4 открываю созданный файл для редактирования в редакторе mcedit и ввожу в файл код программы для запроса строки у пользователя (рис. 4).

Далее выхожу из файла (F10), сохраняя изменения (F2).



```
mc [kbshabakova@vbox]:~/work/arch-pc/lab05
lab5-1.asm [----] 8 L: [ 1+24 25/ 25] *(302 / 302b) <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

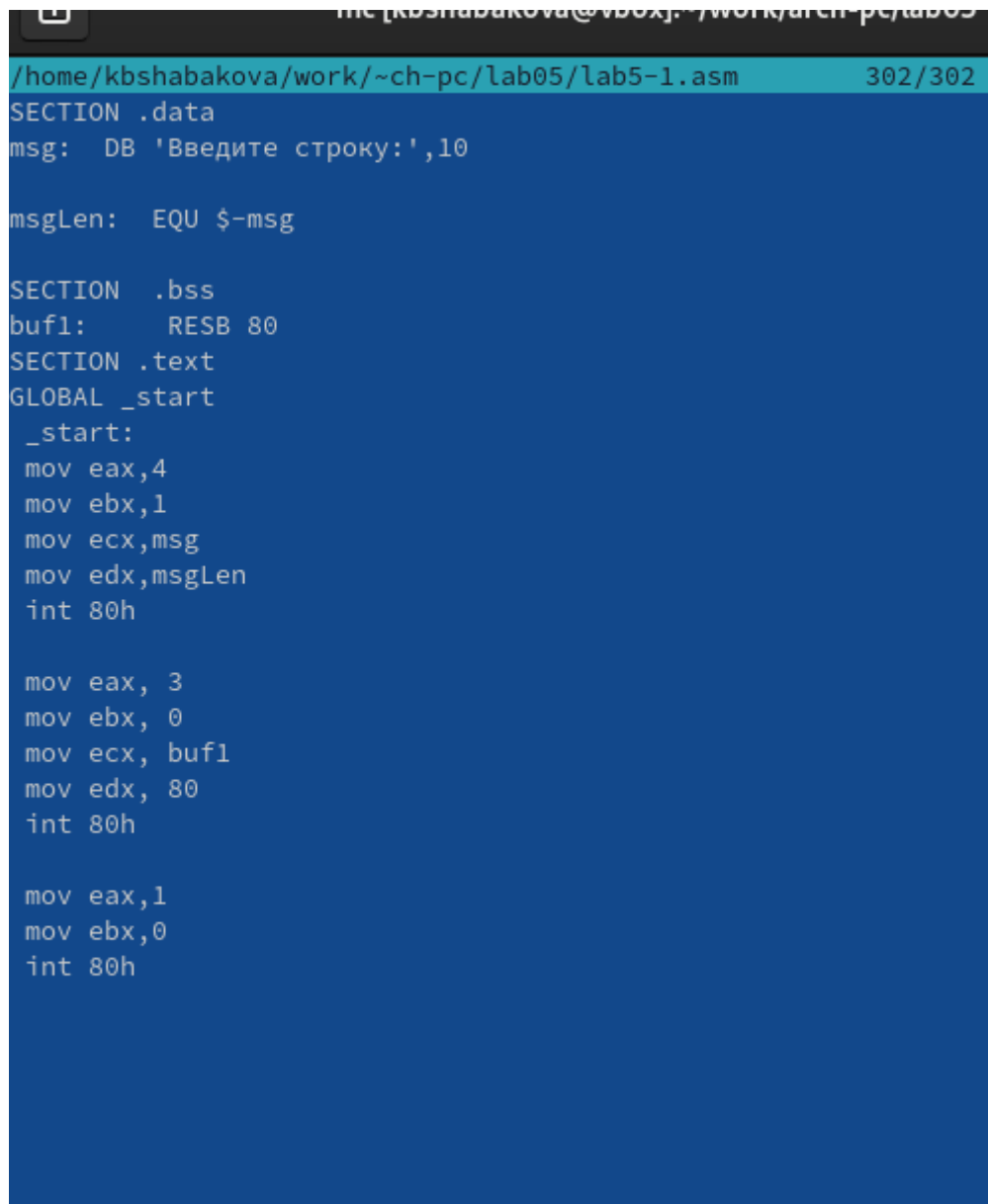
mov eax, 3
mov ebx, 0
mov ecx, buf1
mov edx, 80
int 80h

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

Рис. 4: Редактирование файла

С помощью функциональной клавиши F3 открываю файл для просмотра, чтобы проверить, содержит ли файл текст программы (рис. 5).

Рис. 5:



```
mc [kbshabakova@vbox]: /work/ch-pc/lab05
/home/kbshabakova/work/~ch-pc/lab05/lab5-1.asm 302/302
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
```

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` (рис. 6). Создался исполняемый файл `lab5-1`.

Запускаю исполняемый файл. Программа выводит строку “Введите строку:” и ждет ввода с клавиатуры, я ввожу свои ФИО, на этом программа заканчивает свою работу (рис. 6).

```

kbshabakova@vbox:~/work/arch-pc/lab05$ nasm -f elf lab5-1.asm

kbshabakova@vbox:~/work/arch-pc/lab05$ ld -m elf_i386 -o lab5-1 lab5-1.o

kbshabakova@vbox:~/work/arch-pc/lab05$ ./lab5-1
Введите строку:
Шабакова Карина Баировна

```

Рис. 6: Компиляция файла, передача на обработку компоновщику и исполнение файла

4.3 Подключение внешнего файла

Скачиваю файл `in_out.asm` со страницы курса в ТУИС (рис. 7).

Левая панель	Файл	Команда	Настрой
<-	~/work/arch-pc/lab05	~	<- ~
.и	Имя	Размер	Дата правки
/..	-ВВЕРХ-	ноя 7 18:51	/..
in_out.asm	3942	ноя 7 19:25	/.cach
*lab5-1	8744	ноя 7 19:23	/.cont
lab5-1.asm	302	ноя 7 19:07	/.loc
lab5-1.o	752	ноя 7 19:22	/.moz
			/.ssh
			/work

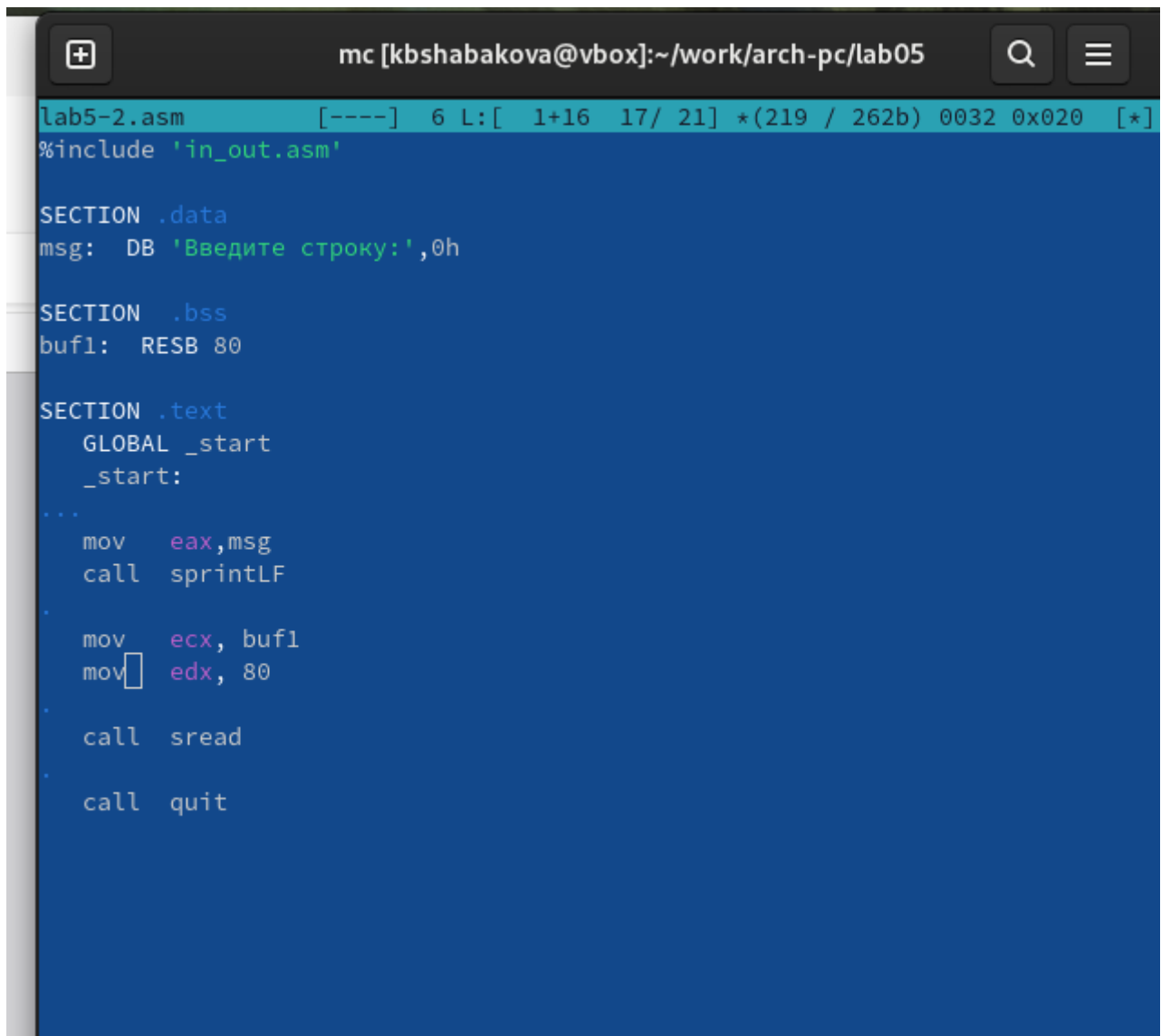
Рис. 7: Скачанный файл

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. 8).

Левая панель	Файл	Команда	На
<-	~/work/arch-pc/lab05	~	<- ~
.и	Имя	Размер	Дата правки
/..	-ВВЕРХ-	ноя 7 18:51	/..
in_out.asm	3942	ноя 7 19:25	/.cach
in_out.asm.save	3944	ноя 7 19:40	/.cont
*lab5-1	8744	ноя 7 19:23	/.loc
lab5-1.asm	302	ноя 7 19:07	/.moz
lab5-1.o	752	ноя 7 19:22	/.ssh
lab5-2.asm	302	ноя 7 19:07	/work

Рис. 8: Копирование файла

Изменяю содержимое файла `lab5-2.asm` во встроенном редакторе `nano` (рис. 9), чтобы в программе использовались подпрограммы из внешнего файла `in_out.asm`.

A screenshot of a text editor window titled 'mc [kbshabakova@vbox]:~/work/arch-pc/lab05'. The editor shows the contents of 'lab5-2.asm'. The code includes an include directive for 'in_out.asm', followed by three sections: .data with a message string, .bss with a buffer reservation, and .text with assembly instructions for printing the message and reading input. The instructions include 'mov', 'call', and 'quit' with various registers and constants.

```
lab5-2.asm [----] 6 L:[ 1+16 17/ 21] *(219 / 262b) 0032 0x020 [*]
#include 'in_out.asm'

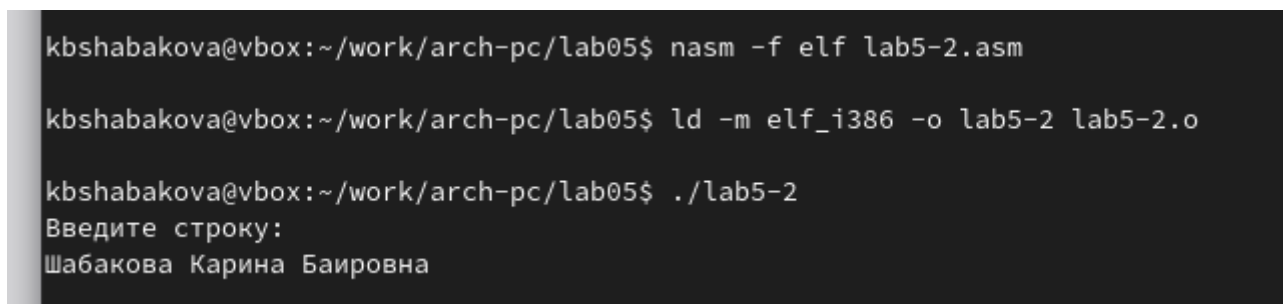
SECTION .data
msg: DB 'Введите строку:',0h

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
```

Рис. 9: Редактирование файла

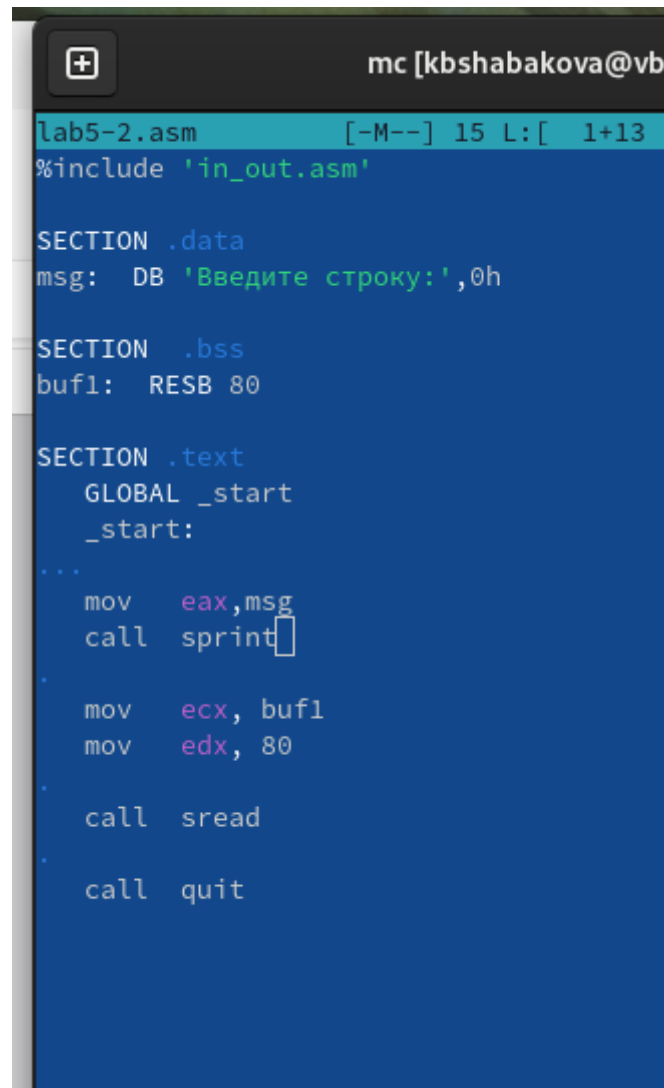
Транслирую текст программы файла в объектный файл командой `nasm -f elf lab5-2.asm`. Создался объектный файл `lab5-2.o`. Выполняю компоновку объектного файла с помощью команды `ld -m elf_i386 -o lab5-2 lab5-2.o`. Создался исполняемый файл `lab5-2`. Запускаю исполняемый файл (рис. 10).

A screenshot of a terminal window showing the compilation and execution of the assembly program. The user runs 'nasm -f elf lab5-2.asm', then 'ld -m elf_i386 -o lab5-2 lab5-2.o', and finally './lab5-2'. The program prompts 'Введите строку:' and the user enters 'Шабакowa Кaрина Бaирoвнa'.

```
kbshabakova@vbox:~/work/arch-pc/lab05$ nasm -f elf lab5-2.asm
kbshabakova@vbox:~/work/arch-pc/lab05$ ld -m elf_i386 -o lab5-2 lab5-2.o
kbshabakova@vbox:~/work/arch-pc/lab05$ ./lab5-2
Введите строку:
Шабакowa Кaрина Бaирoвнa
```

Рис. 10: Исполнение файла

Открываю файл lab5-2.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. 11).



```
mc [kbshabakova@vb
lab5-2.asm [-M--] 15 L: [ 1+13
#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

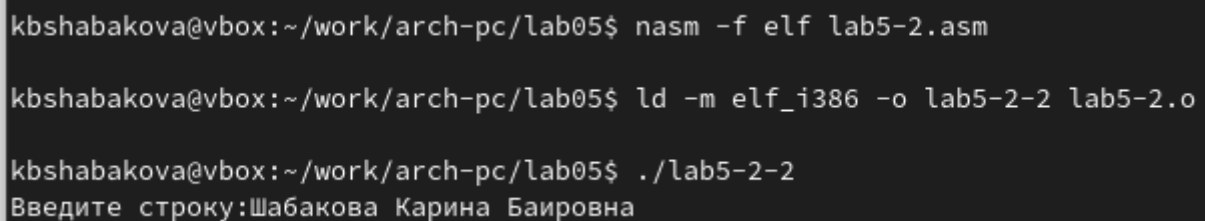
mov    ecx, buf1
mov    edx, 80

call   sread

call   quit
```

Рис. 11: Отредактированный файл

Снова транслирую файл, выполняю компоновку созданного объектного файла, запускаю новый исполняемый файл (рис. 12).



```
kbshabakova@vbox:~/work/arch-pc/lab05$ nasm -f elf lab5-2.asm
kbshabakova@vbox:~/work/arch-pc/lab05$ ld -m elf_i386 -o lab5-2-2 lab5-2.o
kbshabakova@vbox:~/work/arch-pc/lab05$ ./lab5-2-2
Введите строку:Шабакова Карина Баировна
```

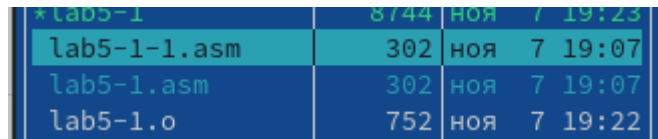
Рис. 12: Исполнение файла

Разница между первым исполняемым файлом lab5-2 и вторым lab5-2-2 в том, что запуск первого запрашивает ввод с новой строки, а программа, которая выполняется при запуске

второго, запрашивает ввод без переноса на новую строку, потому что в этом заключается различие между подпрограммами `sprintLF` и `sprint`.

4.4 Выполнение заданий для самостоятельной работы

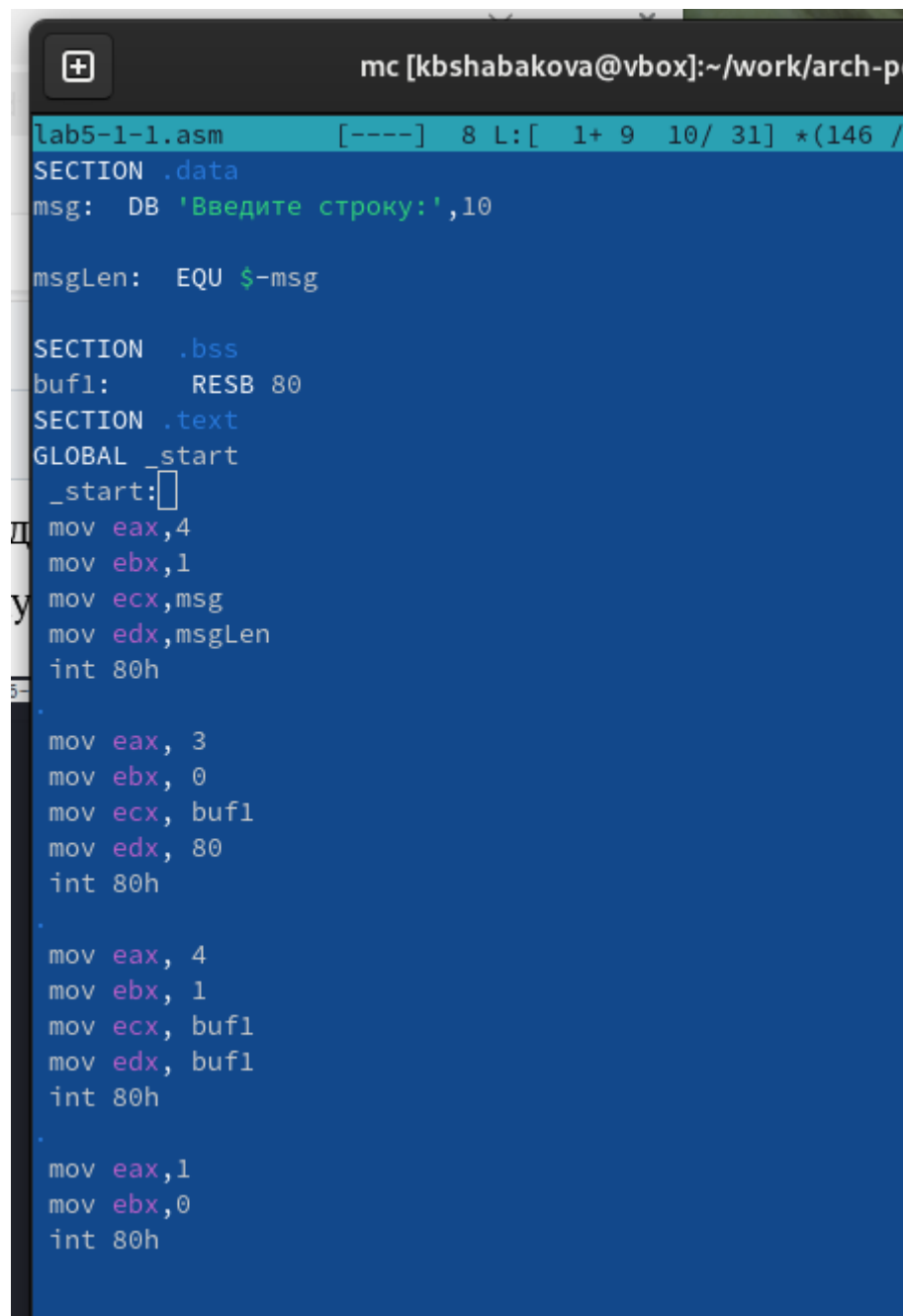
1. Создаю копию файла `lab5-1.asm` с именем `lab51-1.asm` с помощью функциональной клавиши F5 (рис. 13).



*lab5-1	8744	ноя 7 19:23
lab5-1-1.asm	302	ноя 7 19:07
lab5-1.asm	302	ноя 7 19:07
lab5-1.o	752	ноя 7 19:22

Рис. 13: 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. 14).



```
mc [kbshabakova@vbox]:~/work/arch-p
lab5-1-1.asm [----] 8 L: [ 1+ 9 10/ 31] *(146 /
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
```

Рис. 14:

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. 15).

```

kbshabakova@vbox:~/work/arch-pc/lab05$ nasm -f elf lab5-1-1.asm

kbshabakova@vbox:~/work/arch-pc/lab05$ ld -m elf_i386 -o lab5-1-1 lab5-1-1.o

kbshabakova@vbox:~/work/arch-pc/lab05$ ./lab5-1-1
Введите строку:
Шабакова Карина Баировна
Шабакова Карина Баировна

```

Рис. 15: Исполнение файла

Код программы из пункта 1:

```

SECTION .data ; Секция инициализированных данных
msg: DB 'Введите строку:',10
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 ; Описание 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 ; Описание 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 lab5-2.asm с именем lab5-2-1.asm с помощью функциональной клавиши F5 (рис. 16).

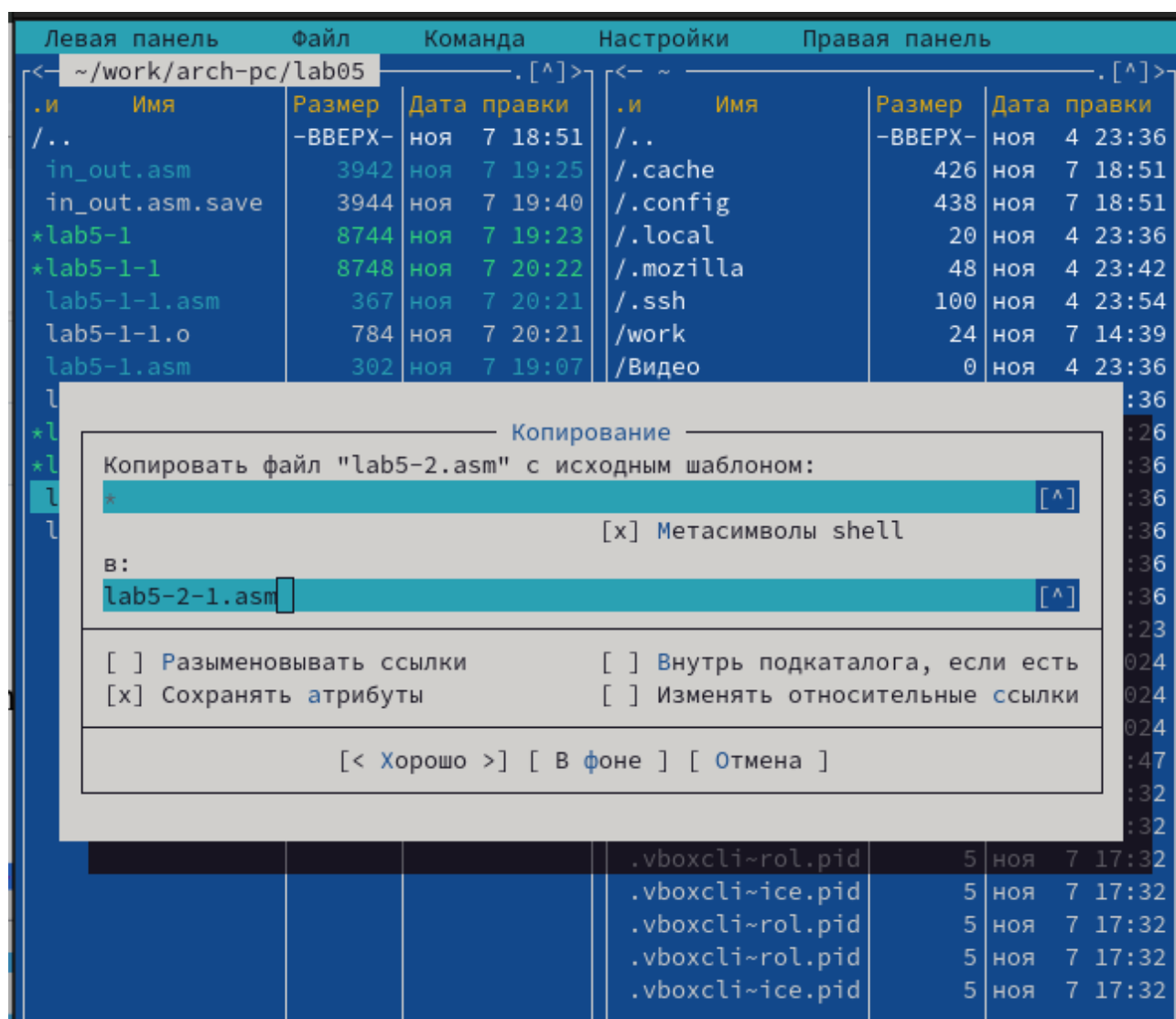
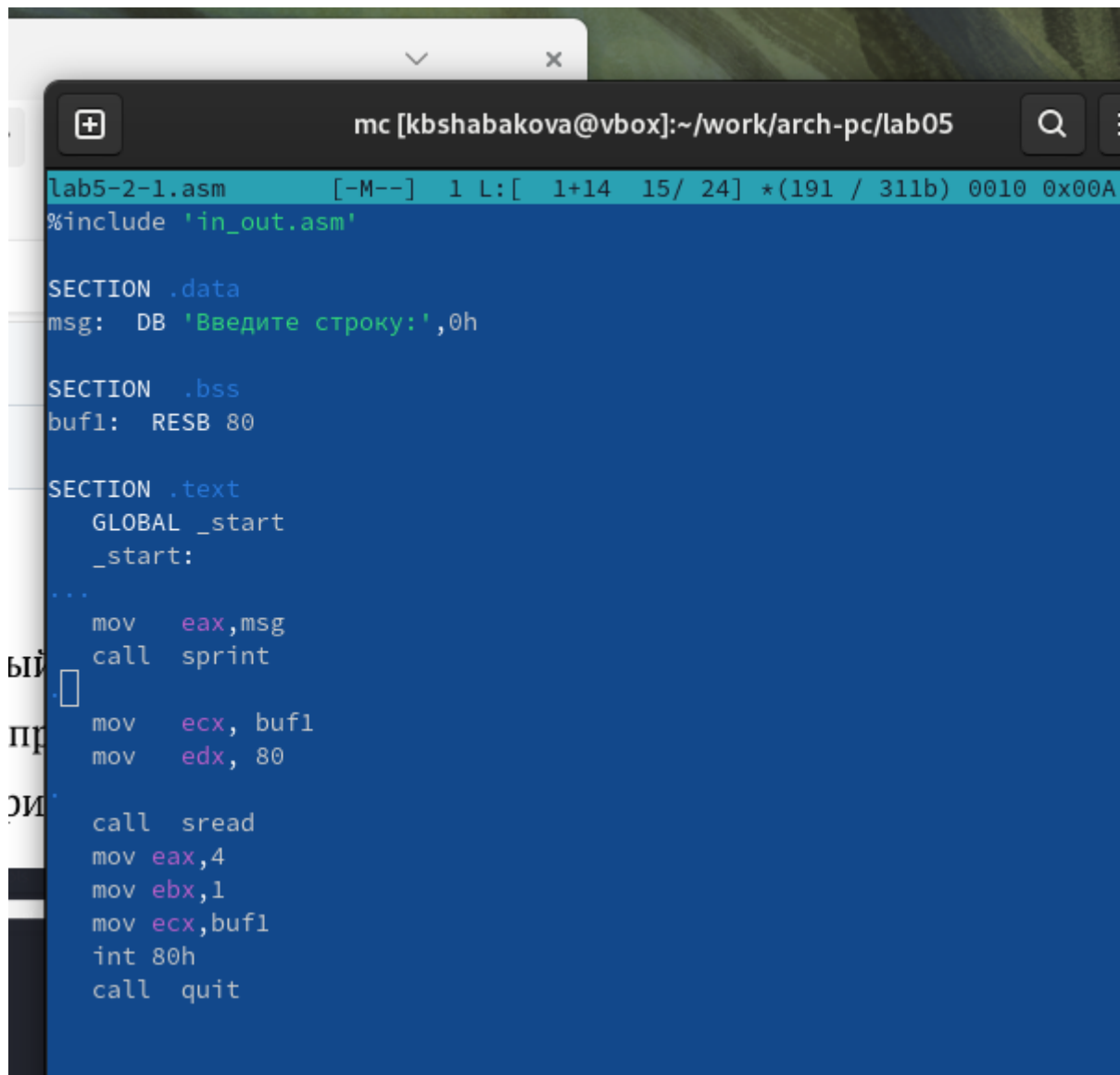


Рис. 16: Копирование файла

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. 17).



```
lab5-2-1.asm [-M--] 1 L: [ 1+14 15/ 24] *(191 / 311b) 0010 0x00A
#include 'in_out.asm'

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

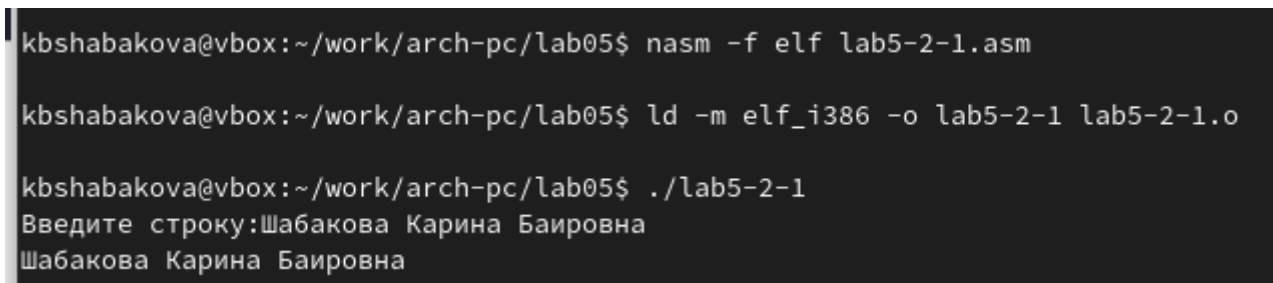
SECTION .bss
buf1: RESB 80

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
```

Рис. 17: 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. 18).



```
kbshabakova@vbox:~/work/arch-pc/lab05$ nasm -f elf lab5-2-1.asm
kbshabakova@vbox:~/work/arch-pc/lab05$ ld -m elf_i386 -o lab5-2-1 lab5-2-1.o
kbshabakova@vbox:~/work/arch-pc/lab05$ ./lab5-2-1
Введите строку:Шабакова Карина Баировна
Шабакова Карина Баировна
```

Рис. 18: Исполнение файла

Код программы из пункта 3:

```
%include 'in_out.asm'
SECTION .data ; Секция инициированных 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
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 Список литературы

1.

https://esystem.rudn.ru/pluginfile.php/2089085/mod_resource/content/0/%D0%9B%D0%B0%D0%B1%D0%BE%D1%80%D0%B0%D1%82%D0%BE%D1%80%D0%BD%D0%B0%D1%8F%20%D1%80%D0%B0%D0%B1%D0%BE%D1%82%D0%B0%20%E2%84%965.%20%D0%9E%D1%81%D0%BD%D0%BE%D0%B2%D1%8B%20%D1%80%D0%B0%D0%B1%D0%BE%D1%82%D1%8B%20%D1%81%20Midnight%20Commander%20%28%29.%20%D0%A1%D1%82%D1%80%D1%83%D0%BA%D1%82%D1%83%D1%80%D0%B0%20%D0%BF%D1%80%D0%BE%D0%B3%D1%80%D0%B0%D0%BC%D0%BC%D1%8B%20%D0%BD%D0%B0%20%D1%8F%D0%B7%D1%8B%D0%BA%D0%B5%20%D0%B0%D1%81%D1%81%D0%B5%D0%BC%D0%B1%D0%BB%D0%B5%D1%80%D0%B0%20NASM.%20%D0%A1%D0%B8%D1%81%D1%82%D0%B5%D0%BC%D0%BD%D1%8B%D0%B5%20%D0%B2%D1%8B%D0%B7%D0%BE%D0%B2%D1%8B%20%D0%B2%20%D0%9E%D0%A1%20GNU%20Linux.pdf