

**РОССИЙСКИЙ УНИВЕРСИТЕТ ДРУЖБЫ НАРОДОВ**

**Факультет физико-математических и естественных наук**

**Кафедра прикладной информатики и теории вероятностей**

**ОТЧЕТ**

**ПО ЛАБОРАТОРНОЙ РАБОТЕ № 5**

*disciplina: Архитектура компьютера*

Студент: Валиева Карина Ринатовна

Группа: НБИбд-01-25

**МОСКВА**

2025 г.

## **Оглавление**

1 Цель работы.....	3
2 Задание.....	4
3 Теоретическое введение .....	5
4 Выполнение лабораторной работы .....	6
<i>4.1 Основы работы с тс .....</i>	<i>6</i>
<i>4.2 Структура программы на языке ассемблера NASM .....</i>	<i>7</i>
<i>4.3 Подключение внешнего файла .....</i>	<i>9</i>
<i>4.4 Выполнение заданий для самостоятельной работы .....</i>	<i>11</i>
5 Выводы .....	16

## **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 (четвертьё 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)

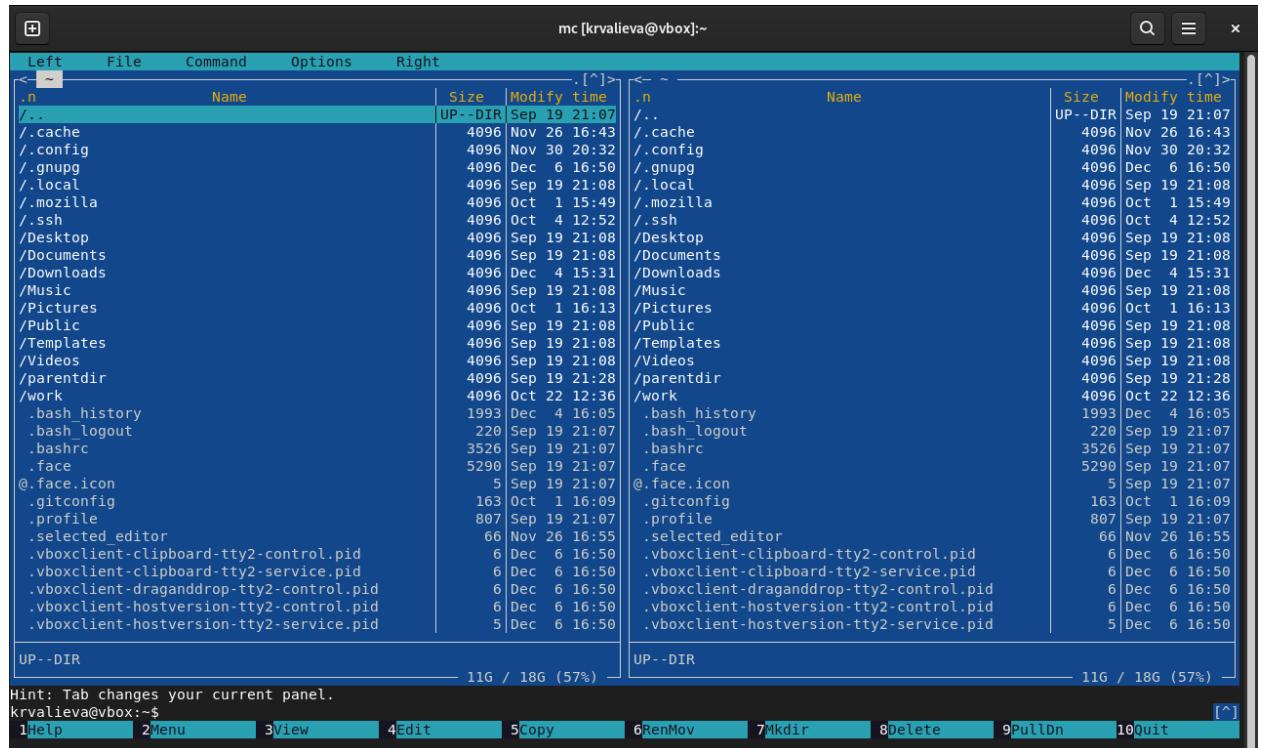


Рис.4. 1. Открытый mc

Перехожу в каталог ~/work/arch-pc, используя файловый менеджер mc и с помощью функциональной клавиши F7 создаю каталог lab05 (Рис.4.2)

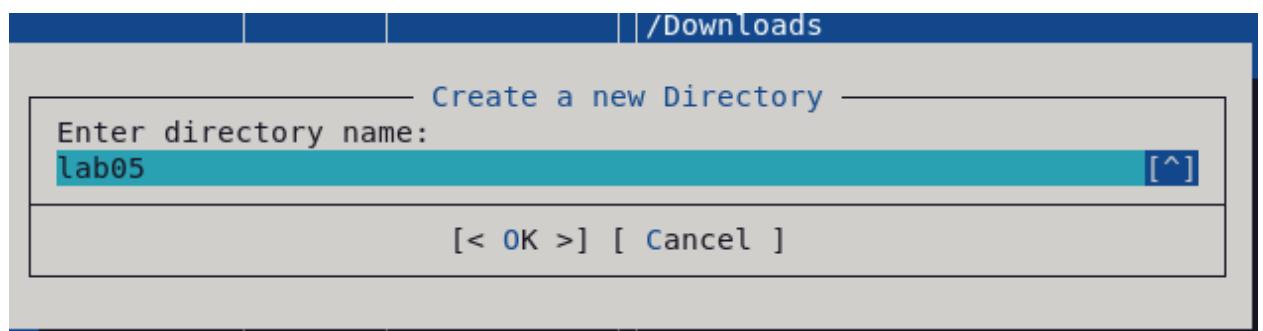


Рис.4. 2. Создание каталога

После того как я перешла в созданный каталог в строке ввода прописываю команду touch lab5-1.asm, чтобы создать файл, в котором буду работать (Рис.4.3).

```

UP--DIR
11G / 18G (57%)
Hint: On slow terminals the -s flag may help.
krvalieva@vbox:~/work/arch-pc/lab05$ touch lab5-1.asm
1Help 2Menu 3View 4Edit 5Copy

```

Рис.4. 3. Создание файла

#### 4.2 Структура программы на языке ассемблера NASM

С помощью функциональной клавиши F4 открываю созданный файл для редактирования в редакторе nano и ввожу в файл код программы для запроса строки у пользователя. Далее выхожу из файла (Ctrl+X), сохраняя изменения (Y, Enter). (Рис.4.4)

```

GNU nano 8.4
/home/krvalieva/work/arch-pc/lab05/lab5-1.asm

; Программа вывода сообщения на экран и ввода строки с клавиатуры
;
;----- Оявление переменных -----
SECTION .data ; Секция инициализированных данных
msg: DB 'Введите строку:',10 ; сообщение плюс
; символ перевода строки
msgLen: EQU $-msg ; Длина переменной 'msg'
SECTION .bss ; Секция не инициализированных 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
;
;----- 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

```

[ Read 35 lines ]

FG Help ^O Write Out ^F Where Is ^K Cut ^T Execute ^C Location M-U Undo M-A Set Mark M-] To Bracket  
 ^X Exit ^R Read File ^L Replace ^U Paste ^J Justify ^A Go To Line M-E Redo M-6 Copy ^B Where Was

Рис.4. 4. Открытие и редактирование файла

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

The screenshot shows the Mars Debugger (MC) window. The assembly code in the main pane is as follows:

```

/home/krvalieva/work/arch-pc/lab05/lab5-1.asm
; Программа вывода сообщения на экран и ввода строки с клавиатуры
;----- Оявление переменных -----
SECTION .data ; Секция инициализированных данных
msg: DB 'Введите строку:',10 ; сообщение плюс
; символ перевода строки
msgLen: EQU $-msg ; Длина переменной 'msg'
SECTION .bss ; Секция не инициализированных 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
;----- 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

```

The status bar at the bottom shows: 1Help 2UnWrap 3Quit 4Hex 5Goto 6 7Search 8Raw 9Format 10Quit.

*Рис.4. 5. 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. (Рис.4.6)

```

nasm -f elf lab5-1.asm
ld -m elf_i386 -o lab5-1 lab5-1.o
mc

```

*Рис.4. 6. Компиляция файла и передача на обработку компоновщику*

Запускаю исполняемый файл. Программа выводит строку “Введите строку:” и ждет ввода с клавиатуры, я ввожу свои ФИО, на этом программа заканчивает свою работу (Рис. 4.7).

```

krvalieva@vbox:~/work/arch-pc/lab05$ ./lab5-1
Введите строку:
Валиева Карина

```

*Рис.4. 7. Исполнение файла*

#### 4.3 Подключение внешнего файла

Скачиваю файл `in_out.asm` со страницы курса в ТУИС. Он сохранился в каталог “Загрузки” (Рис. 4.8)

< ~/Downloads		Name		Size	Modify time
.	n	UP--DIR	Dec 4 13:50	4096	Oct 4 20:31
/..				4096	Oct 4 20:31
/tsetup.6.1.3				338	Oct 22 12:33
hello.asm				3942	Nov 30 20:56
in_out.asm				782	Nov 21 21:23
lab4.asm					

Рис.4. 8. Скачанный файл

С помощью функциональной клавиши F5 копирую файл `in_out.asm` из каталога Загрузки в созданный каталог `lab05` (Рис.4.9).

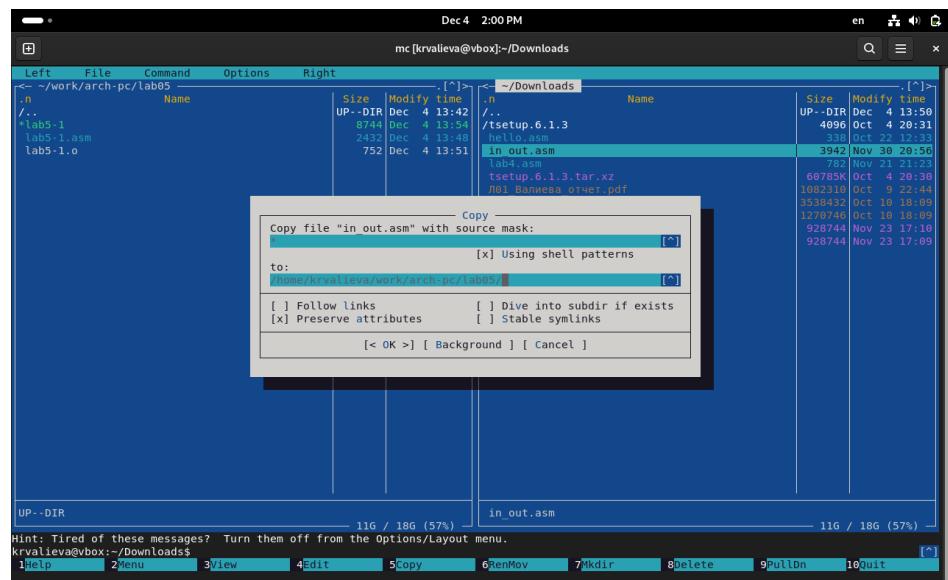


Рис.4. 9. Копирование файла

С помощью функциональной клавиши F5 копирую файл `lab5-1` в тот же каталог, но с другим именем. (Рис.4.10)

< ~/work/arch-pc/lab05		Name		Size	Modify time
.	n	UP--DIR	Dec 4 13:42	3942	Nov 30 20:56
/..				8744	Dec 4 13:54
*lab5-1				2432	Dec 4 13:48
lab5-1.asm				752	Dec 4 13:51
lab5-1.o					
lab5-2.asm				2432	Dec 4 13:48

Рис.4. 10. Копирование файла

Изменяю содержимое файла lab5-2.asm во встроенным редакторе nano, чтобы в программе использовались подпрограммы из внешнего файла in\_out.asm. (Рис. 4.11)

```
GNU nano 8.4                               /home/krvalieva/work/arch-pc/lab05/lab5-2.asm *
;-----;
; Программа вывода сообщения на экран и ввода строки с клавиатуры ;
;-----;
%include 'in_out.asm' ; подключение внешнего файла
SECTION .data ; Секция инициализированных 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 sprintLF ; вызов подпрограммы печати сообщения
    mov ecx, buf1 ; запись адреса переменной в `EAX`
    mov edx, 80 ; запись длины вводимого сообщения в `EBX`
    56 Демидова А. В.
    Архитектура ЭВМ
    call sread ; вызов подпрограммы ввода сообщения
    call quit ; вызов подпрограммы завершения
```

*Рис.4. 11. 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. 4.12).

```
krvalieva@vbox:~/work/arch-pc/lab05$ nasm -f elf lab5-2.asm
krvalieva@vbox:~/work/arch-pc/lab05$ ld -m elf_i386 -o lab5-2 lab5-2.o
krvalieva@vbox:~/work/arch-pc/lab05$ ./lab5-2
Введите строку: Valieva Karina
```

*Рис.4. 12. 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. 4.13).

```
/home/krvalieva/work/arch-pc/lab05/lab5-2.asm
;-----;
; Программа вывода сообщения на экран и ввода строки с клавиатуры ;
;-----;
%include 'in_out.asm' ; подключение внешнего файла
SECTION .data ; Секция инициализированных 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
    call quit ; вызов подпрограммы завершения
```

*Рис.4. 13. Отредактированный файл*

Снова транслирую файл, выполняю компоновку созданного объектного файла, запускаю новый исполняемый файл (Рис. 4.14).

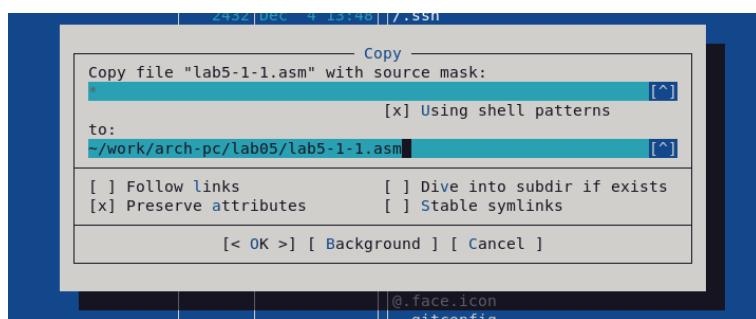
```
krvalieva@vbox:~/work/arch-pc/lab05$ nasm -f elf lab5-2.asm  
krvalieva@vbox:~/work/arch-pc/lab05$ ld -m elf_i386 -o lab5-2-2 lab5-2.o  
krvalieva@vbox:~/work/arch-pc/lab05$ ./lab5-2-2  
Введите строку:  
Валиева Карина
```

*Рис.4. 14. 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 исполняется 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 lab5-1-1.asm с помощью функциональной клавиши F5 (рис. 4.15).



*Рис.4. 15. 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. 4.16).

```

GNU nano 8.4                               /home/krvalieva/work/arch-pc/lab05/lab5-1-1.asm
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 ; 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

```

*Рис.4. 16. Редактирование файла*

Код программы :

```

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 ; 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

```

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).

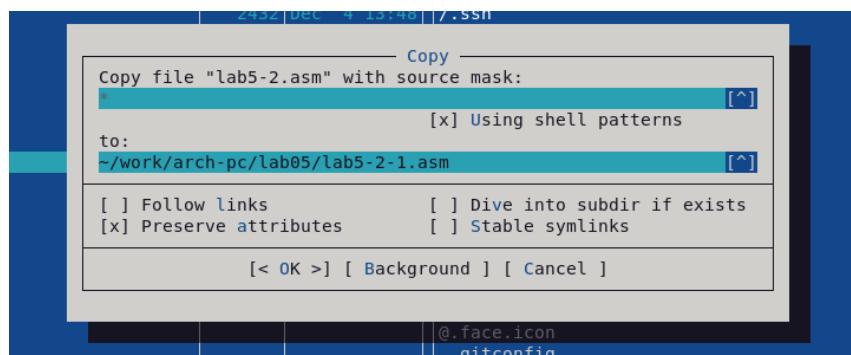
```

krvalieva@vbox:~/work/arch-pc/lab05$ nasm -f elf lab5-1-1.asm
krvalieva@vbox:~$ mc
krvalieva@vbox:~/work/arch-pc/lab05$ ld -m elf_i386 -o lab5-1-1 lab5-1-1.o
krvalieva@vbox:~/work/arch-pc/lab05$ ./lab5-1-1
Введите строку:
Валиева Карина

```

*Рис.4. 17. Исполнение файла*

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*

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. 4.19).

```

GNU nano 8.4                               /home/krvalieva/work/arch-pc/lab05/lab5-2-1.asm
%include 'in_out.asm'
SECTION .data ; Секция инициализированных 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 выводимого 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

```

*Rис.4. 19. Редактирование файла*

Код программы:

```
%include 'in_out.asm'

SECTION .data ; Секция инициализированных 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 выводимого 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
```

call quit ; вызов подпрограммы завершения

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).

```
krvalieva@vbox:~/work/arch-pc/lab05$ nasm -f elf lab5-2-1.asm  
krvalieva@vbox:~/work/arch-pc/lab05$ ld -m elf_i386 -o lab5-2-1 lab5-2-1.o  
krvalieva@vbox:~/work/arch-pc/lab05$ ./lab5-2-1  
Ведите строку: Валиева Карина
```

*Рис.4. 20. 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.