

# Отчет по лабораторной работе №5

## Дисциплина: архитектура компьютера

Королев Павел

### Содержание

1	Цель работы .....	1
2	Задание.....	1
3	Теоретическое введение .....	1
4	Выполнение лабораторной работы .....	2
4.1	Основы работы с mc .....	2
4.2	Структура программы на языке ассемблера NASM.....	4
4.3	Подключение внешнего файла .....	8
4.4	Выполнение заданий для самостоятельной работы .....	12
5	Выводы.....	16
6	Список литературы.....	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 иницированных (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 иницированных 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.

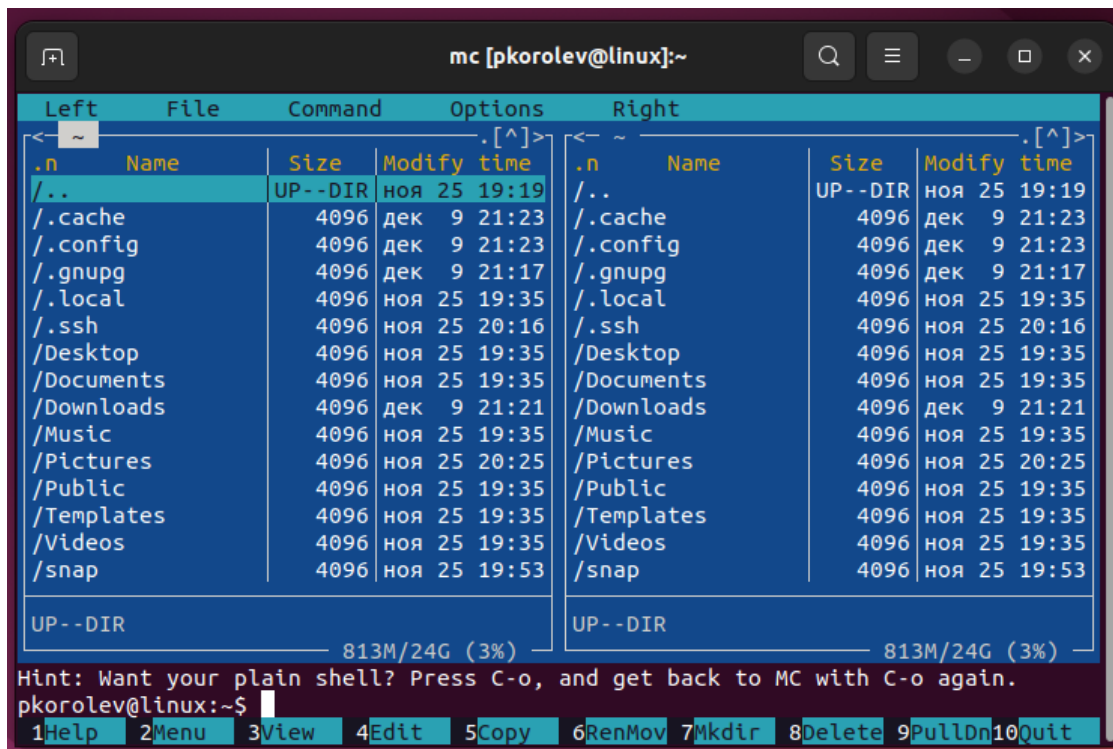
```
int n
```

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

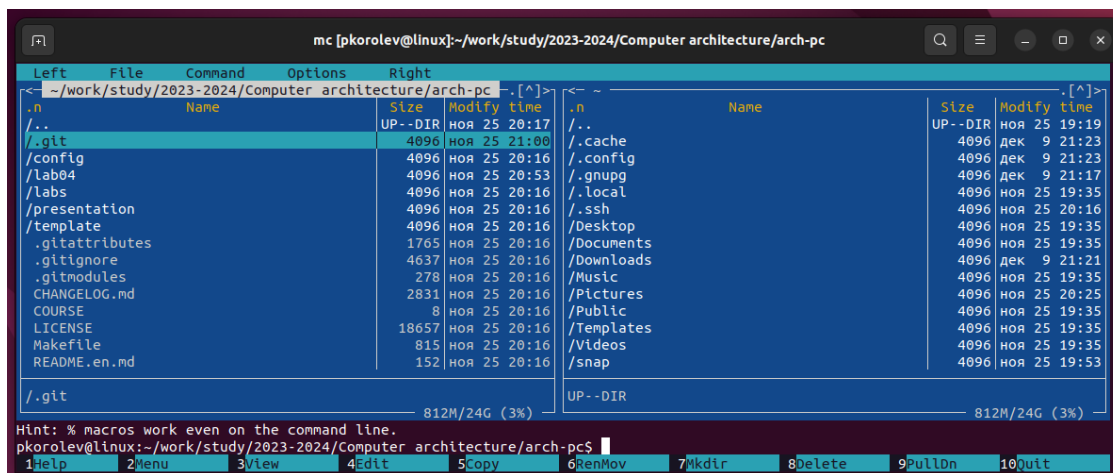
### 4.1 Основы работы с mc

Открываю Midnight Commander, введя в терминал mc (рис. 01).



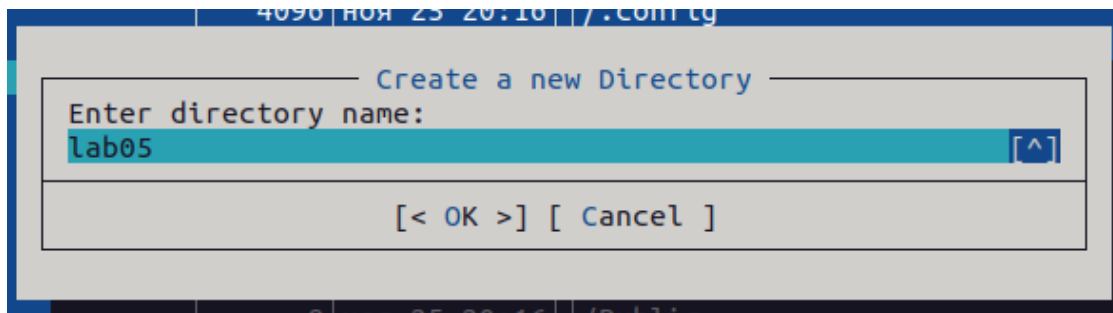
Открытый mc

Перехожу в каталог ~/work/study/2022-2023/Архитектура Компьютера/arch-pc, используя файловый менеджер mc (рис. 02)



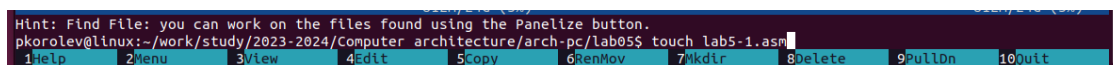
Перемещение между директориями

С помощью функциональной клавиши F7 создаю каталог lab05 (рис. 03).



### *Создание каталога*

Переходу в созданный каталог (рис. 05).



### *Перемещение между директориями*

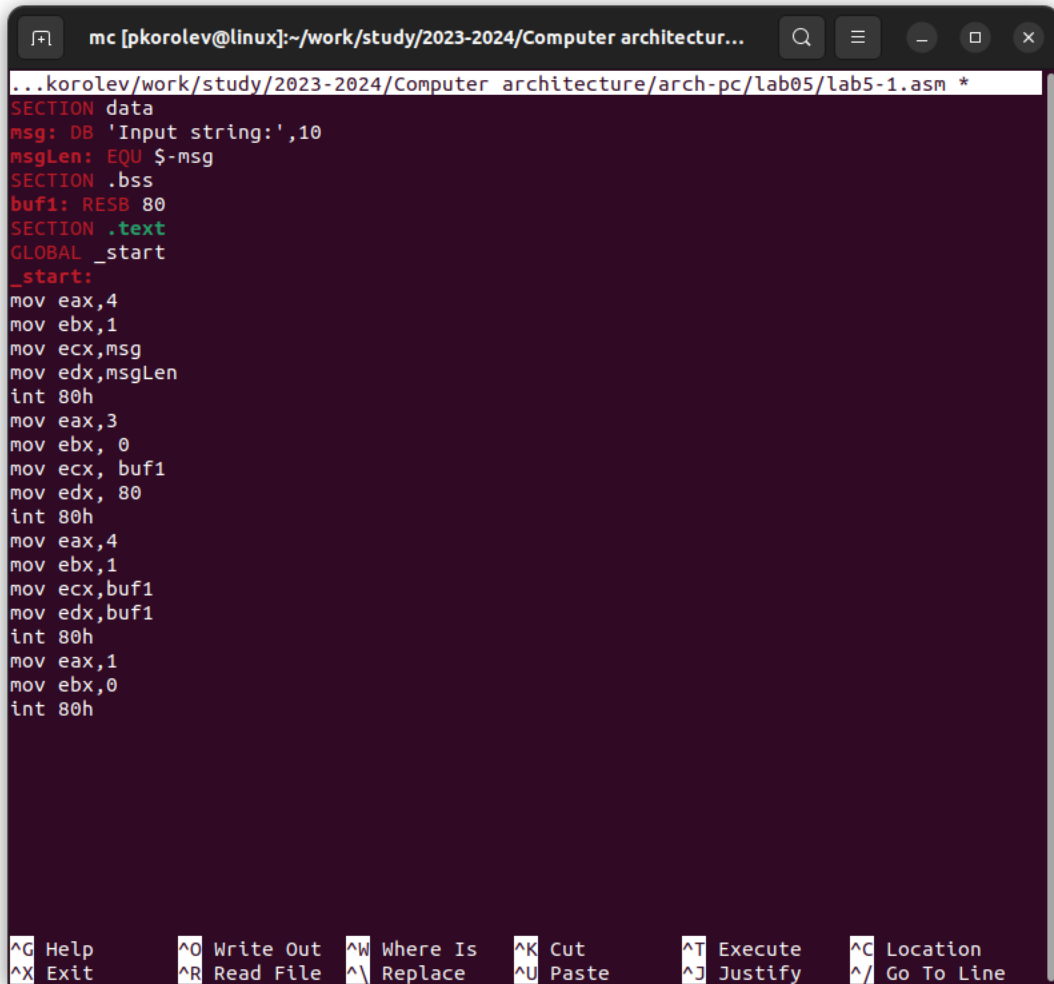
В строке ввода прописываю команду `touch lab5-1.asm`, чтобы создать файл, в котором буду работать (рис. 06).



### *Создание файла*

## **4.2 Структура программы на языке ассемблера NASM**

С помощью функциональной клавиши F4 открываю созданный файл для редактирования в редакторе nano (рис. 07).



The screenshot shows a text editor window with a dark theme. The title bar at the top reads "mc [pkorolev@linux]:~/work/study/2023-2024/Computer architecture...". The editor's address bar shows the file path "...korolev/work/study/2023-2024/Computer architecture/arch-pc/lab05/lab5-1.asm \*". The main text area contains the following assembly code:

```
SECTION data
msg: DB 'Input string:',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
```

At the bottom of the window, there is a menu bar with the following items:

^G Help	^O Write Out	^W Where Is	^K Cut	^T Execute	^C Location
^X Exit	^R Read File	^_ Replace	^U Paste	^J Justify	^_ Go To Line

*Открытие файла для редактирования*

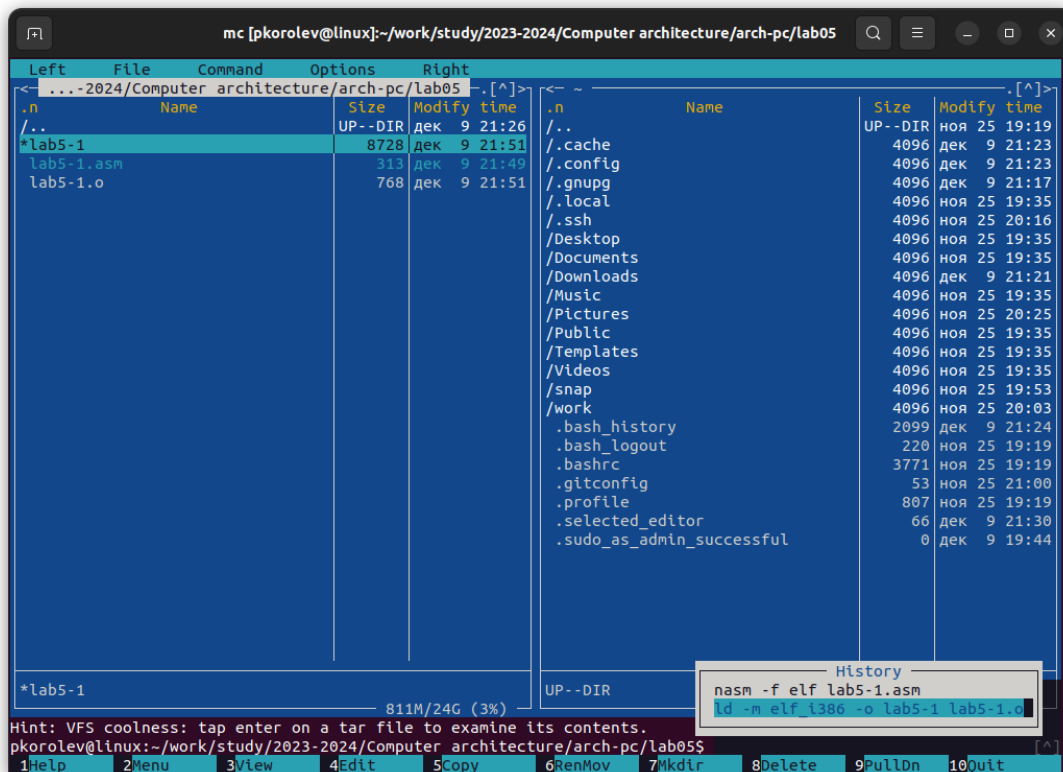
Ввожу в файл код программы для запроса строки у пользователя (рис. 08). Далее выхожу из файла (Ctrl+X), сохраняя изменения (Y, Enter).

```
mc [pkorolev@linux]:~/work/study/2023-2024/Computer architectur...
/home/pkorolev/work/study/~/arch-pc/lab05/lab5-1.asm 313/313 100%
SECTION data
msg: DB 'Input string:',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

1Help 2Unwrap 3Quit 4Hex 5Goto 6 7Search 8Raw 9Format 10Quit
```

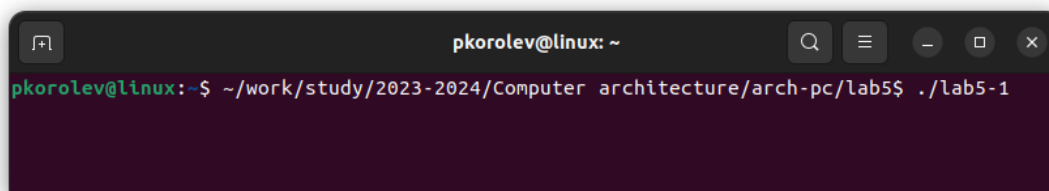
### *Редактирование файла*

С помощью функциональной клавиши F3 открываю файл для просмотра, чтобы проверить, содержит ли файл текст программы (рис. 09).



### 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. 10). Создался исполняемый файл `lab5-1`.



### Компиляция файла и передача на обработку компоновщику

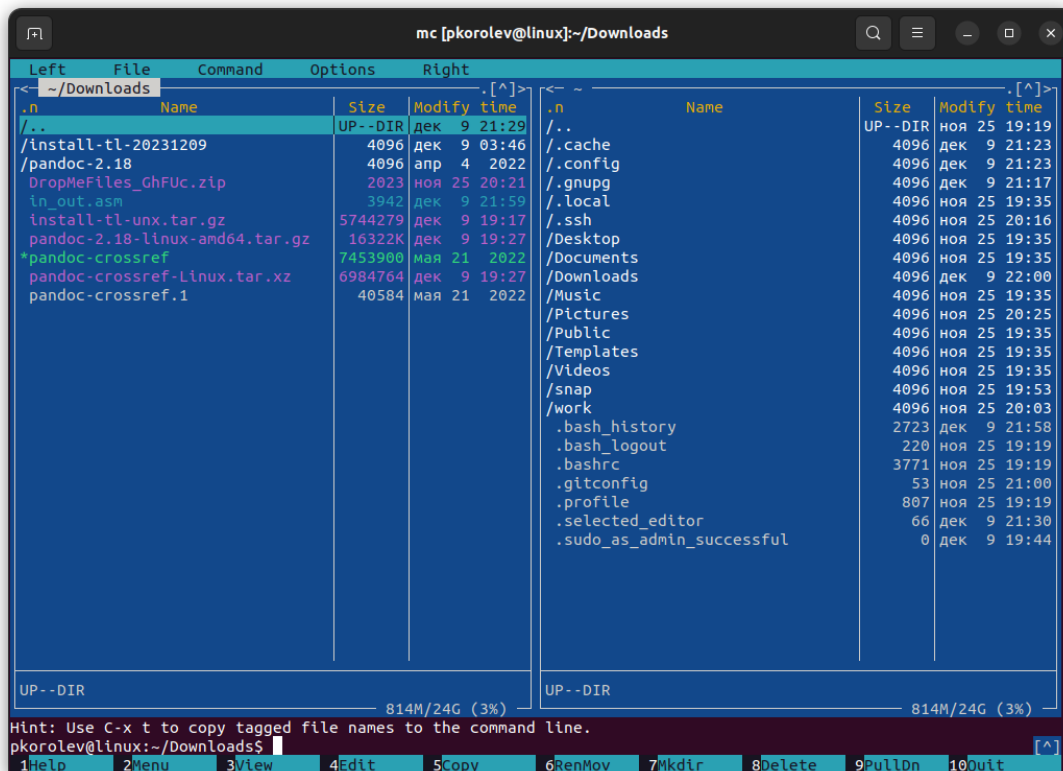
Запускаю исполняемый файл. Программа выводит строку "Введите строку:" и ждет ввода с клавиатуры, я ввожу свои ФИО, на этом программа заканчивает свою работу (рис. 11).

```
pkorolev@linux: ~  
pkorolev@linux:~$ ~/work/study/2023-2024/"Computer architecture"/arch-pc/lab05/lab5-1  
Input string:  
Korolev Pavel
```

Исполнение файла

## 4.3 Подключение внешнего файла

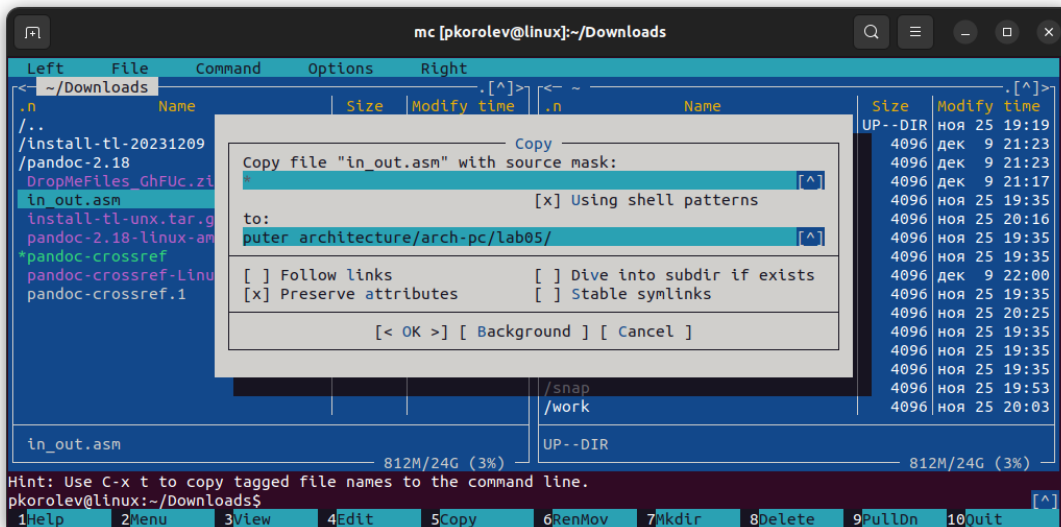
Скачиваю файл in\_out.asm со страницы курса в ТУИС. Он сохранился в каталог “Загрузки” (рис. 12).



Скачанный файл

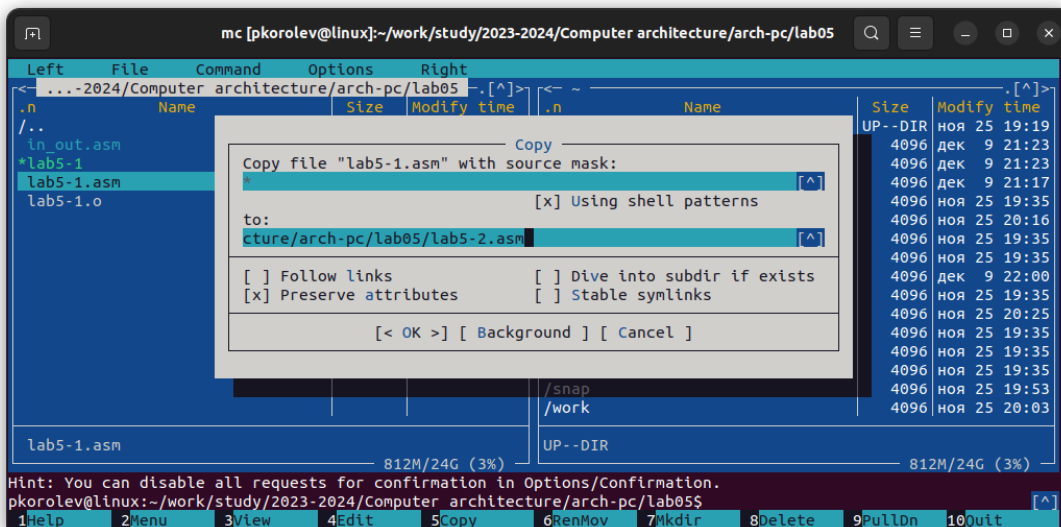
С помощью функциональной клавиши F5 копирую файл in\_out.asm из каталога Загрузки в созданный каталог lab05 (рис. 13).





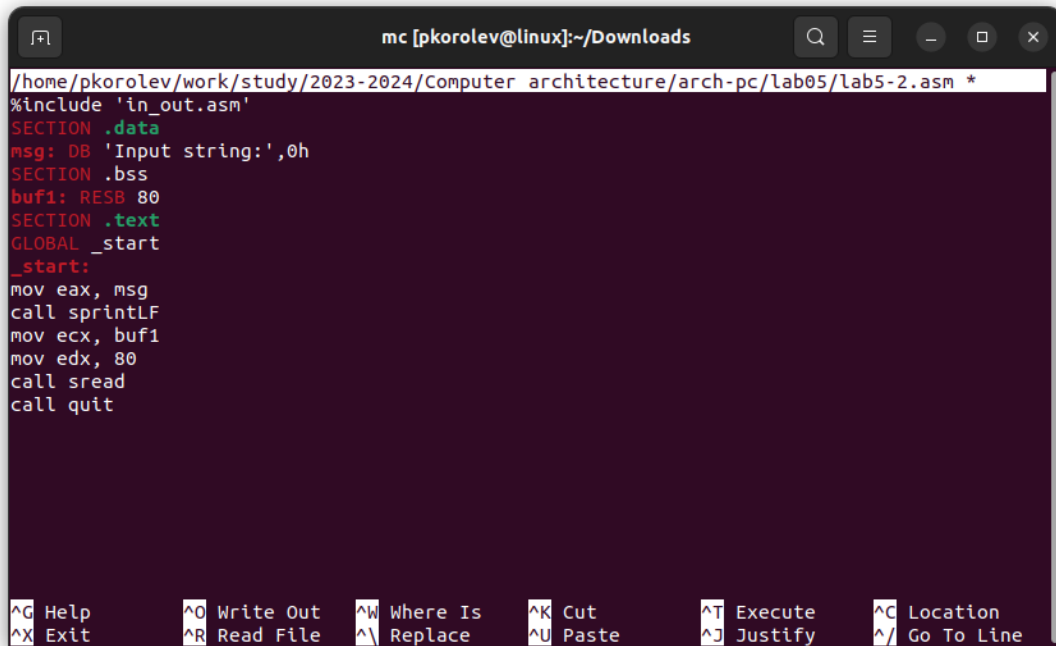
### Копирование файла

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. 14).



### Копирование файла

Изменяю содержимое файла lab5-2.asm во встроенном редакторе nano (рис. 15), чтобы в программе использовались подпрограммы из внешнего файла in\_out.asm.

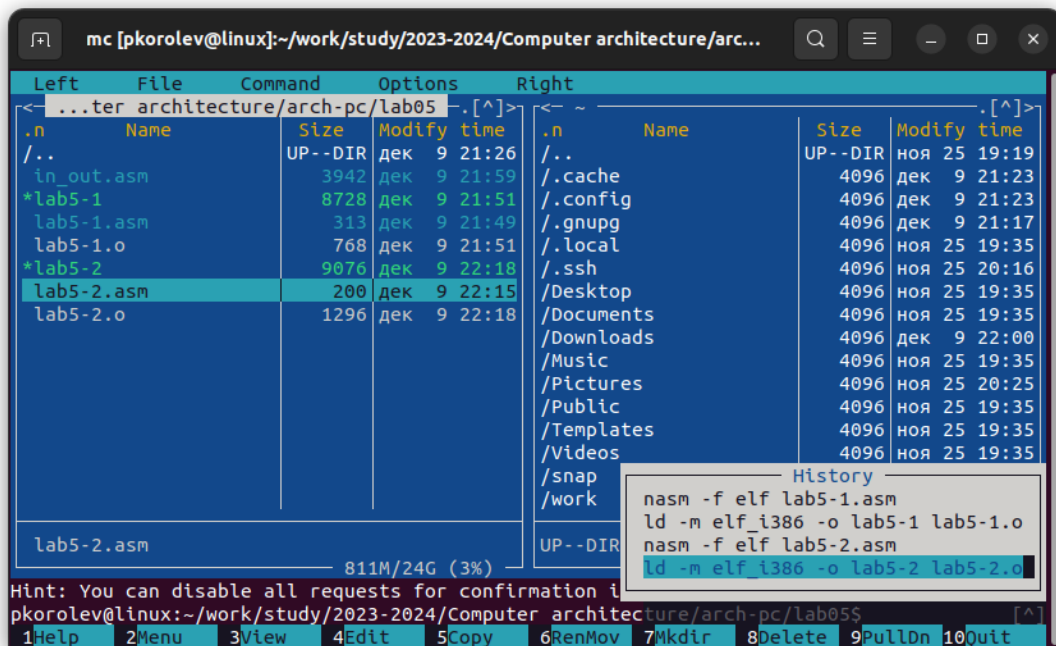


```
mc [pkorolev@linux]:~/Downloads
/home/pkorolev/work/study/2023-2024/Computer architecture/arch-pc/lab05/lab5-2.asm *
#include 'in_out.asm'
SECTION .data
msg: DB 'Input string:',0h
SECTION .bss
buf1: RESB 80
SECTION .text
GLOBAL _start
_start:
mov eax, msg
call sprintf
mov ecx, buf1
mov edx, 80
call sread
call quit
```

Help Write Out Where Is Cut Execute Location  
Exit Read File Replace Paste Justify Go To Line

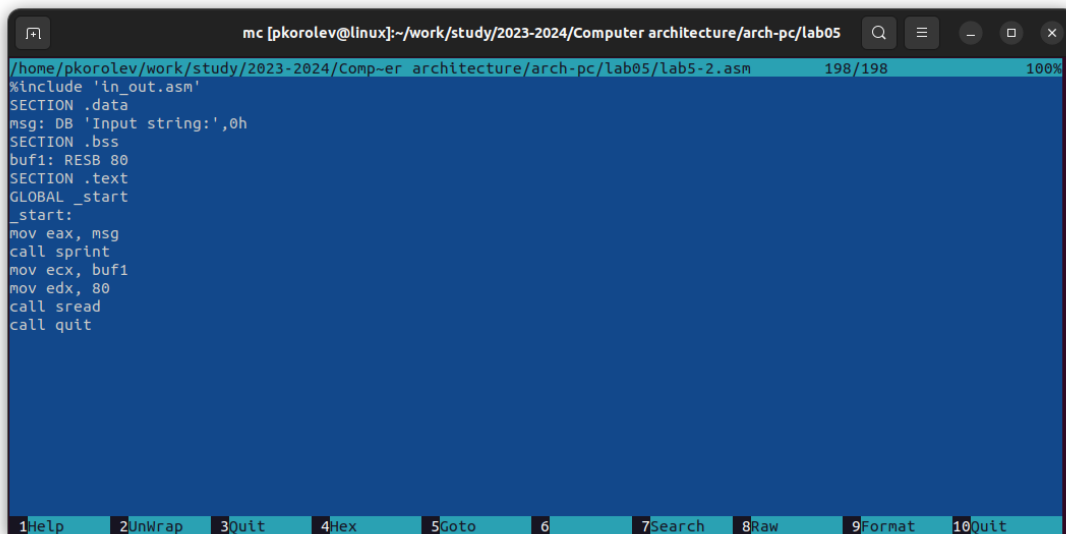
### *Редактирование файла*

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. 16).



### 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. 17).



### Отредактированный файл

Снова транслирую файл, выполняю компоновку созданного объектного файла, запускаю новый исполняемый файл (рис. 18).

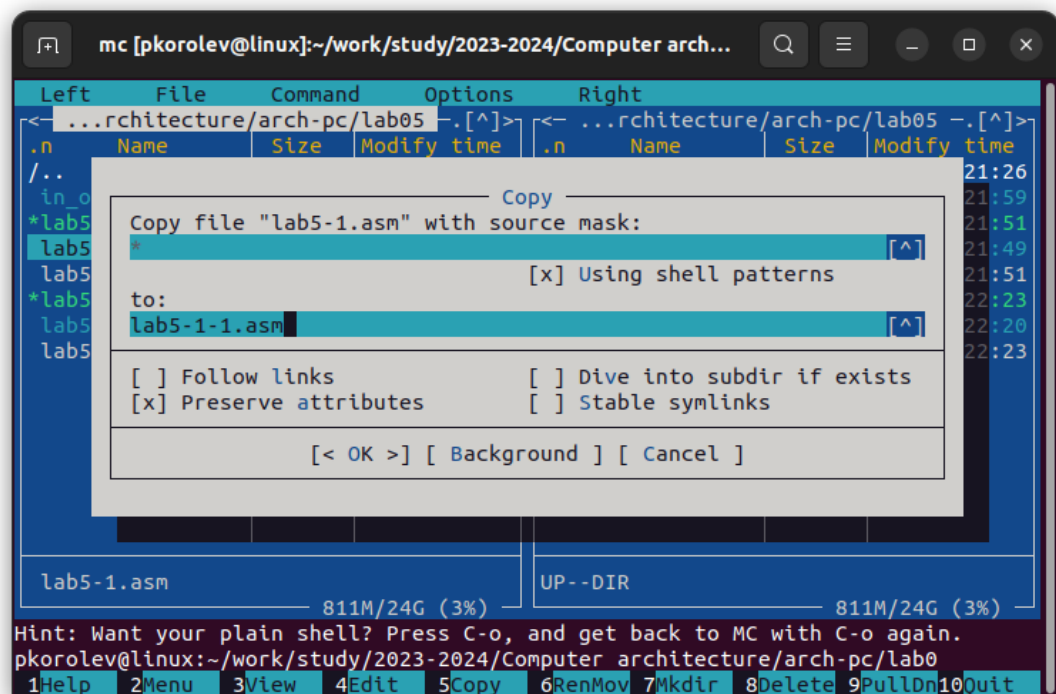
```
pkorolev@linux: ~  
pkorolev@linux:~$ ~/work/study/2023-2024/"Computer architecture"/arch-pc/lab05/lab5-2  
Input string:Korolev Pavel  
pkorolev@linux:~$
```

### 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 `sprintf` и `sprint`.

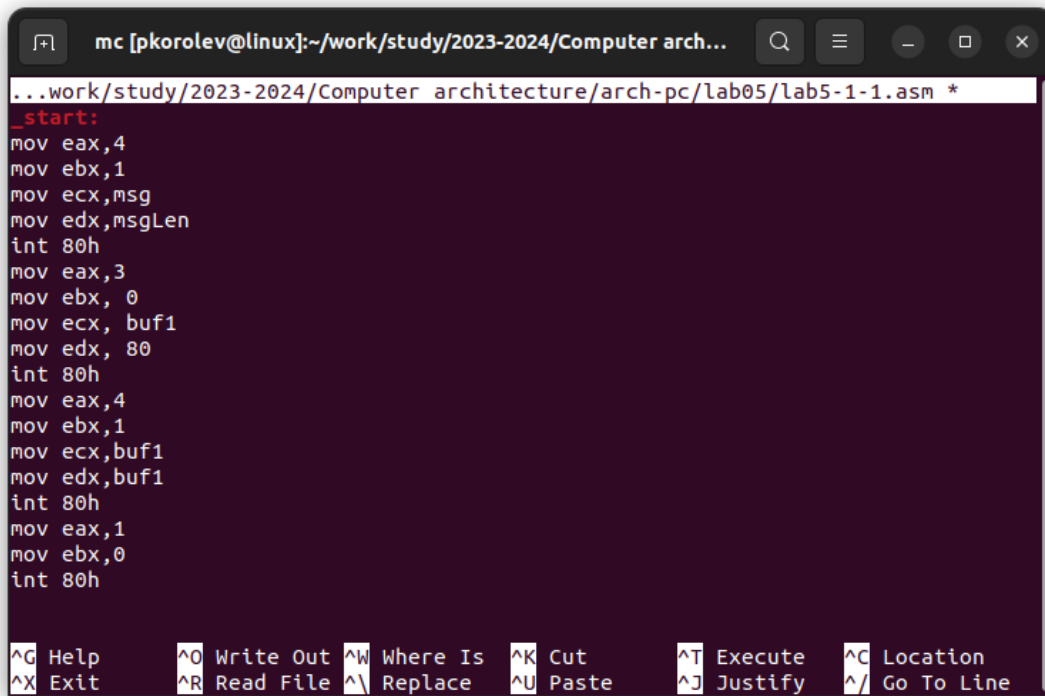
## 4.4 Выполнение заданий для самостоятельной работы

1. Создаю копию файла lab5-1.asm с именем lab5-1-1.asm с помощью функциональной клавиши F5 (рис. 19).



### 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. 20).

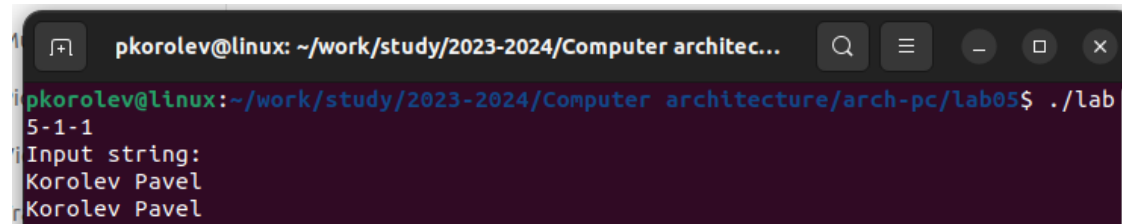


```
mc [pkorolev@linux]:~/work/study/2023-2024/Computer arch...
...work/study/2023-2024/Computer architecture/arch-pc/lab05/lab5-1-1.asm *
_start:
mov eax,4
mov ebx,1
mov ecx,msg
mov edx,msgLen
int 80h
mov eax,3
mov ebx,0
mov ecx,buf1
mov edx,80
int 80h
mov eax,4
mov ebx,1
mov ecx,buf1
mov edx,buf1
int 80h
mov eax,1
mov ebx,0
int 80h

^G Help      ^O Write Out ^W Where Is  ^K Cut       ^T Execute   ^C Location
^X Exit      ^R Read File ^_ Replace   ^U Paste     ^J Justify   ^_ Go To Line
```

### 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. 21).



```
pkorolev@linux: ~/work/study/2023-2024/Computer architec...
pkorolev@linux:~/work/study/2023-2024/Computer architecture/arch-pc/lab05$ ./lab5-1-1
Input string:
Korolev Pavel
Korolev Pavel
```

### Исполнение файла

Код программы из пункта 1:

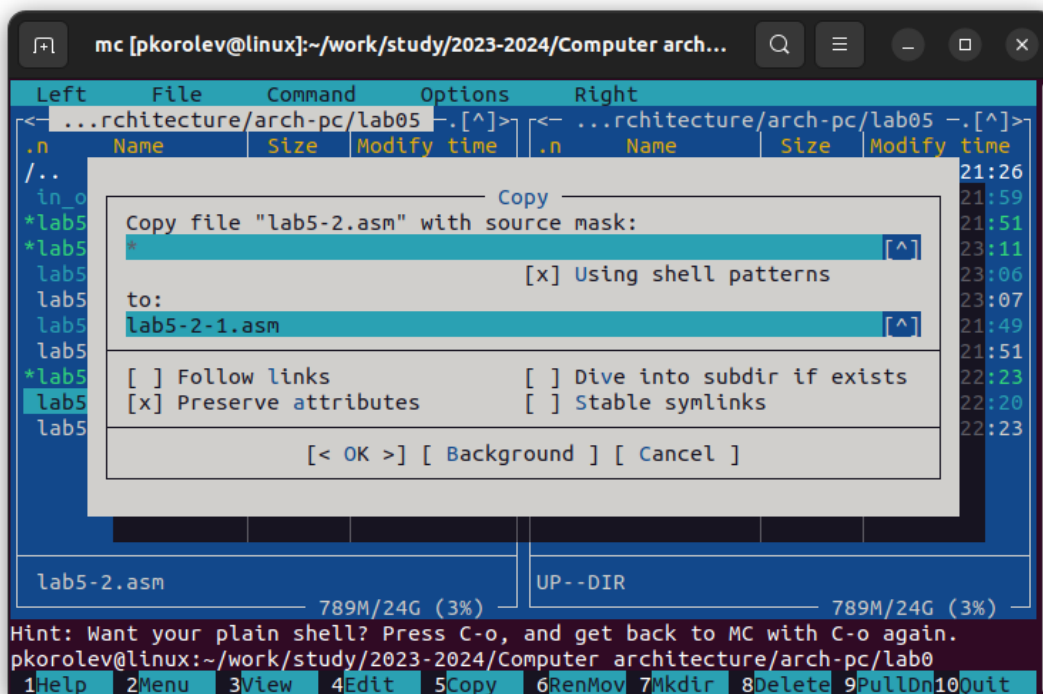
```
SECTION .data ; Секция иницированных данных
msg: DB 'Введите строку:',10
msgLen: EQU $-msg ; Длина переменной 'msg'
SECTION .bss ; Секция не иницированных данных
buf1: RESB 80 ; Буфер размером 80 байт
SECTION .text ; Код программы
GLOBAL _start ; Начало программы
_start: ; Точка входа в программу
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

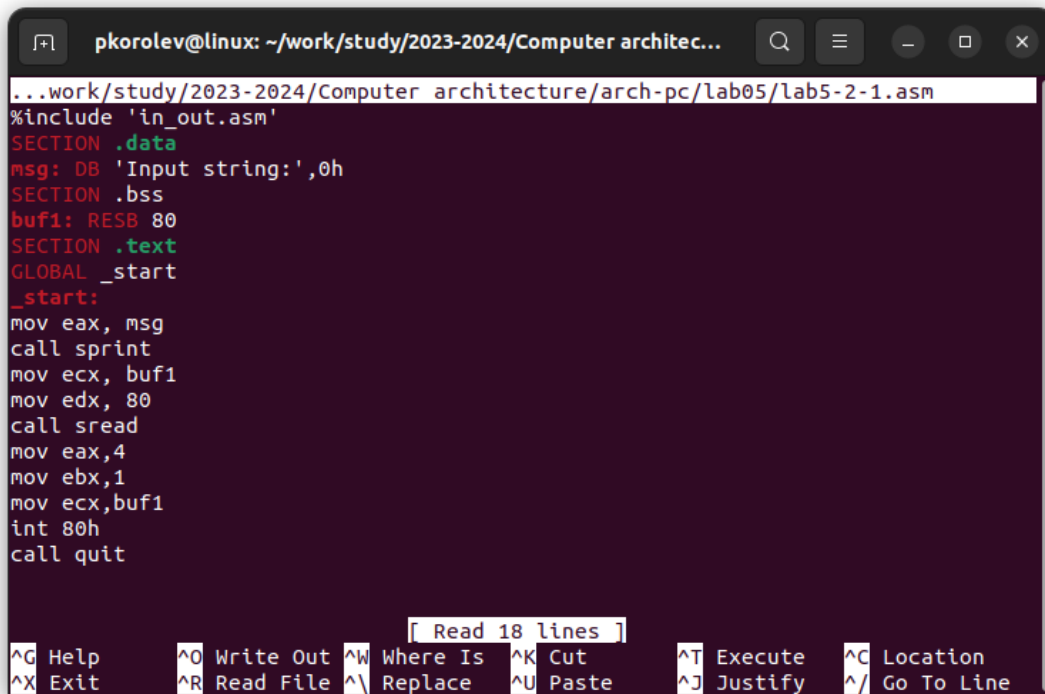
```

3. Создаю копию файла lab5-2.asm с именем lab5-2-1.asm с помощью функциональной клавиши F5 (рис. 22).



### 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. 23).



```
..work/study/2023-2024/Computer architecture/arch-pc/lab05/lab5-2-1.asm
%include 'in_out.asm'
SECTION .data
msg: DB 'Input string:',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
```

### 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. 24).

```
pkorolev@linux:~/work/study/2023-2024/Computer architecture/arch-pc/lab05$ ./lab5-2-1
Input string:Korolev Pavel
Korolev Pavel

pkorolev@linux:~/work/study/2023-2024/Computer architecture/arch-pc/lab05$ ./lab5-2-1
Input string:
```

### Исполнение файла

Код программы из пункта 3:

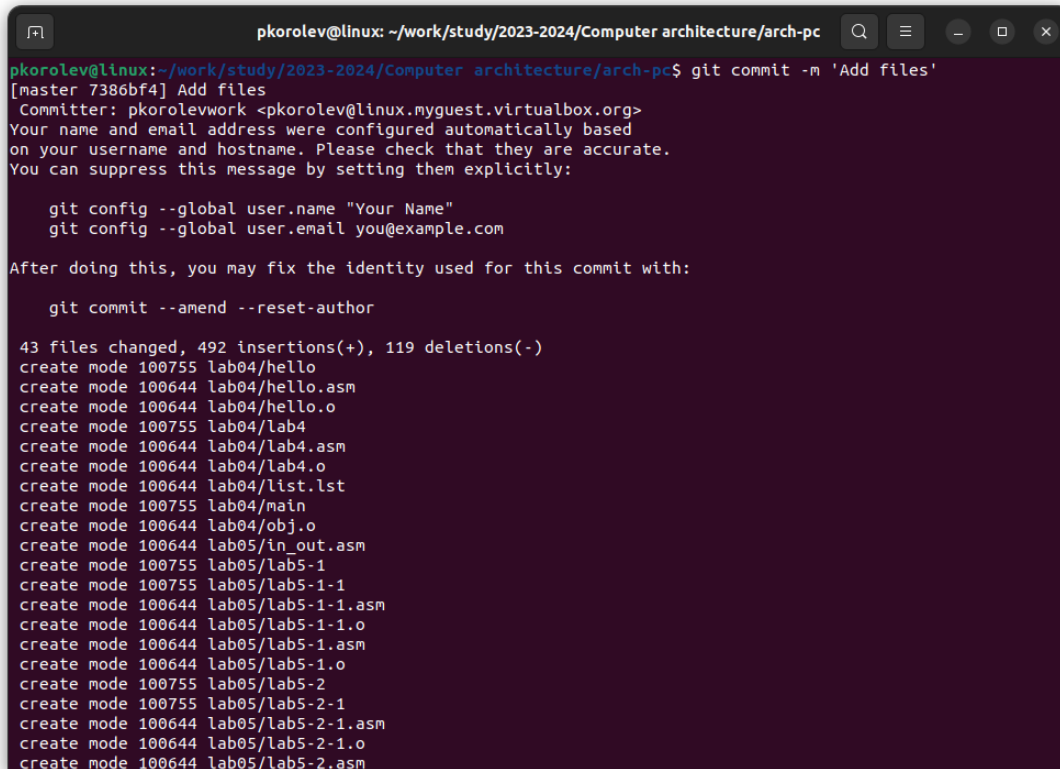
```
%include 'in_out.asm'
SECTION .data ; Секция инициированных данных
msg: DB 'Введите строку: ',0h ; сообщение
SECTION .bss ; Секция не инициированных данных
buf1: RESB 80 ; Буфер размером 80 байт
SECTION .text ; Код программы
GLOBAL _start ; Начало программы
_start: ; Точка входа в программу
mov eax, 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, 4 ; Системный вызов для записи (sys_write)
mov ebx, 1 ; Описание файла '1' - стандартный вывод
mov ecx, buf1 ; Адрес строки buf1 в ecx
int 80h ; Вызов ядра
call quit ; вызов подпрограммы завершения

```

5. С помощью команд `git add .`, `git commit -m 'Add files'`, `git push` добавляю файлы лабораторной в репозиторий GitHub



```

pkorolev@linux: ~/work/study/2023-2024/Computer architecture/arch-pc
pkorolev@linux:~/work/study/2023-2024/Computer architecture/arch-pc$ git commit -m 'Add files'
[master 7386bf4] Add files
Committer: pkorolevwork <pkorolev@linux.myguest.virtualbox.org>
Your name and email address were configured automatically based
on your username and hostname. Please check that they are accurate.
You can suppress this message by setting them explicitly:

    git config --global user.name "Your Name"
    git config --global user.email you@example.com

After doing this, you may fix the identity used for this commit with:

    git commit --amend --reset-author

43 files changed, 492 insertions(+), 119 deletions(-)
create mode 100755 lab04/hello
create mode 100644 lab04/hello.asm
create mode 100644 lab04/hello.o
create mode 100755 lab04/lab4
create mode 100644 lab04/lab4.asm
create mode 100644 lab04/lab4.o
create mode 100644 lab04/list.lst
create mode 100755 lab04/main
create mode 100644 lab04/obj.o
create mode 100644 lab05/in_out.asm
create mode 100755 lab05/lab5-1
create mode 100755 lab05/lab5-1-1
create mode 100644 lab05/lab5-1-1.asm
create mode 100644 lab05/lab5-1-1.o
create mode 100644 lab05/lab5-1.asm
create mode 100644 lab05/lab5-1.o
create mode 100755 lab05/lab5-2
create mode 100755 lab05/lab5-2-1
create mode 100644 lab05/lab5-2-1.asm
create mode 100644 lab05/lab5-2-1.o
create mode 100644 lab05/lab5-2.asm

```

Загрузка на сервер

## 5 Выводы

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

## 6 Список литературы

1. Лабораторная работа №5