

Отчет по лабораторной работе №5

Костеренко Полина

Содержание

1	1.Цель работы:	5
2	2.Задание:	6
3	3.Выполнение лабораторной работы.	7
4	4. Задание для самостоятельной работы.	17
5	Выводы	21

Список иллюстраций

3.1	Вводим в консоль команду ms.	8
3.2	Переходим в каталог.	9
3.3	Воспользуемся командой touch.	9
3.4	Открываем файл функциональной клавишей, заполняем и сохраняем.	10
3.5	Текст программы.	11
3.6	Проверяем, как работает данная программа.	11
3.7	Скачиваем файл.	12
3.8	Копируем скачанный файл.	12
3.9	Создаем копию файла клавишей F6.	13
3.10	Проверяем скопировался ли файл.	14
3.11	Открываем и заполняем файл.	15
3.12	Смотрим, как сработала программа.	15
3.13	Редактируем файл.	16
3.14	Сравниваем с прошлой.	16
4.1	Создаем копию файла lab5-1.asm.	17
4.2	Редактируем файл.	18
4.3	Проверяем правильность написания программы.	18
4.4	Создаем копию файла lab5-2.asm.	19
4.5	Редактируем файл.	19
4.6	Проверяем правильность написания программы.	20

Список таблиц

1 1.Цель работы:

Приобретение практических навыков работы в Midnight Commander. Освоение инструкций языка ассемблера mov и int.

2 2.Задание:

Написать 2 программы по примеру и впоследствии изменить их по условию.

3 3.Выполнение лабораторной работы.

3.1 Открываем Midnight Commander. рис. 3.1

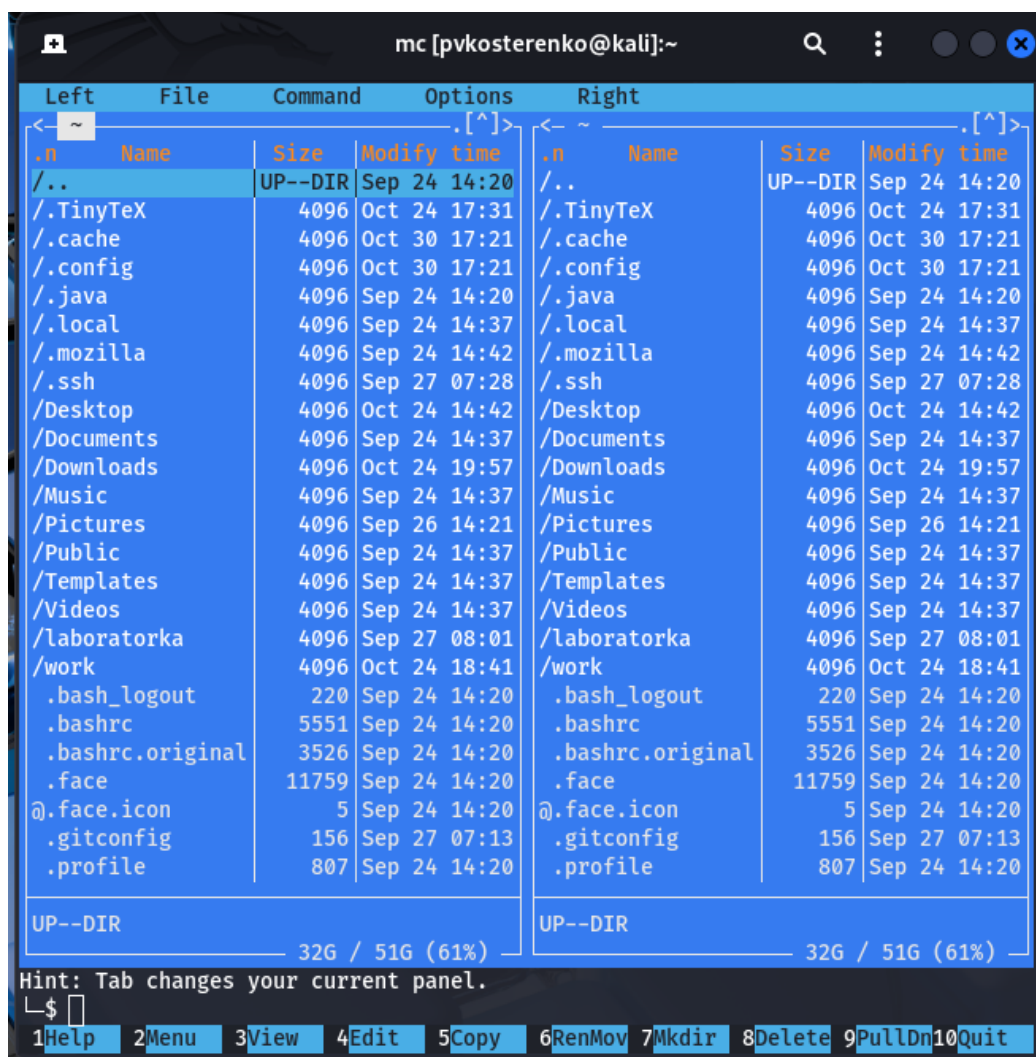


Рисунок 3.1: Вводим в консоль команду mc.

3.2 Переходим в каталог, созданный при выполнении 4 лабораторной работы.

рис. 3.2

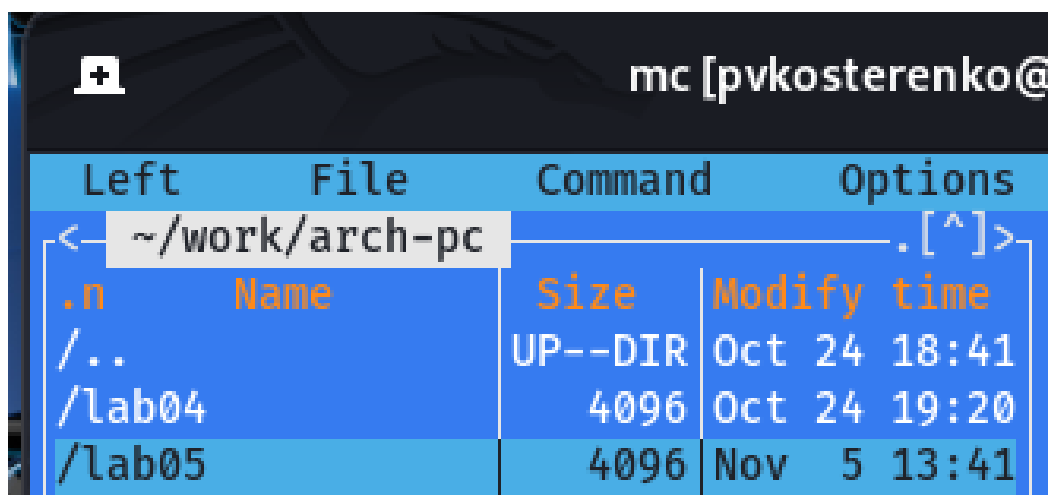


Рисунок 3.2: Переходим в каталог.

3.3 Создаем каталог lab05 функциональной клавишей F7, создаем файл lab5-1.asm.

рис. 3.3

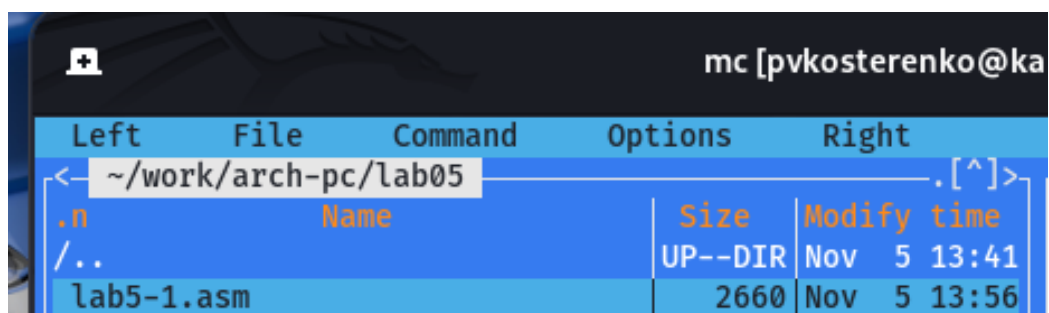
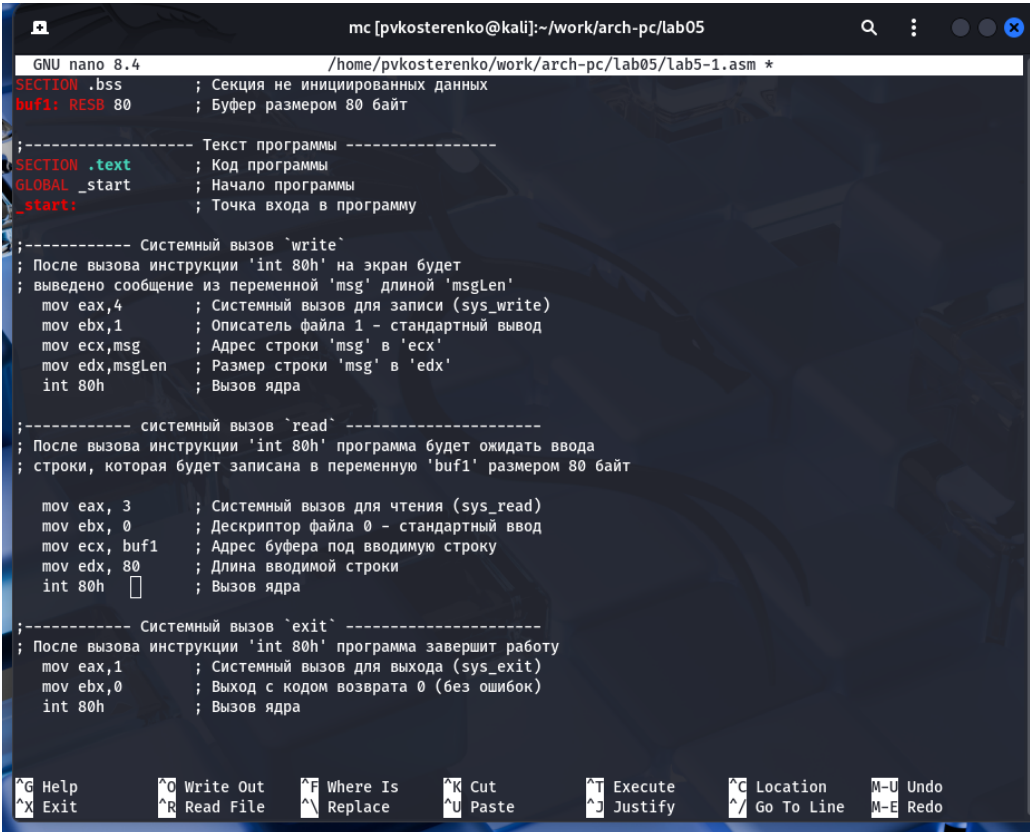


Рисунок 3.3: Воспользуемся командой touch.

3.4 Открываем файл для редактирования и заполняем его по листингу. рис. 3.4



```
mc [pvkosterenko@kali]:~/work/arch-pc/lab05
GNU nano 8.4 /home/pvkosterenko/work/arch-pc/lab05/lab5-1.asm *
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

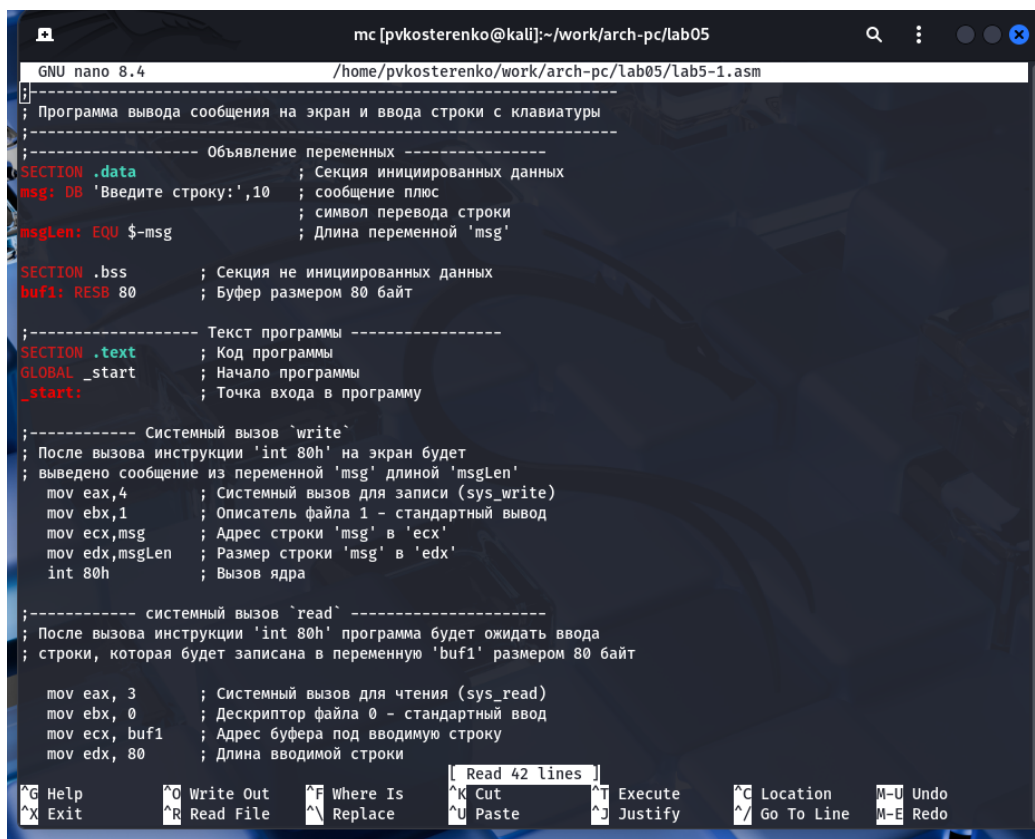
mov eax, 3 ; Системный вызов для чтения (sys_read)
mov ebx, 0 ; Дескриптор файла 0 - стандартный ввод
mov ecx, buf1 ; Адрес буфера под вводимую строку
mov edx, 80 ; Длина вводимой строки
int 80h ; Вызов ядра

;----- 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

^C Help ^O Write Out ^F Where Is ^K Cut ^T Execute ^C Location M-U Undo
^X Exit ^R Read File ^\ Replace ^U Paste ^_ Justify ^_ Go To Line M-E Redo
```

Рисунок 3.4: Открываем файл функциональной клавишей, заполняем и сохраняем.

3.5 Открываем файл для просмотра и убеждаемся, что файл содержит текст программы. рис. 3.5



```
mc [pvkosterenko@kali]:~/work/arch-pc/lab05
GNU nano 8.4 /home/pvkosterenko/work/arch-pc/lab05/lab5-1.asm
; Программа вывода сообщения на экран и ввода строки с клавиатуры
;
;----- Объявление 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

^C Help      ^O Write Out  ^F Where Is   Read 42 lines
^X Exit      ^R Read File ^V Replace    ^K Cut      ^T Execute
            ^U Paste    ^J Justify    ^C Location  ^M-U Undo
            ^_ Go To Line ^M-E Redo
```

Рисунок 3.5: Текст программы.

3.6 Транслируем текст программы и запускаем исполняемый файл. рис. 3.6



```
(pvkosterenko@kali)-[~/work/arch-pc/lab05]
$ nasm -f elf lab5-1.asm

(pvkosterenko@kali)-[~/work/arch-pc/lab05]
$ ld -m elf_i386 -o lab5-1 lab5-1.o

(pvkosterenko@kali)-[~/work/arch-pc/lab05]
$ ./lab5-1
Введите строку:
Костеренко Полина
```

Рисунок 3.6: Проверяем, как работает данная программа.

3.7 Скачиваем файл со страницы ТУИС. рис. 3.7

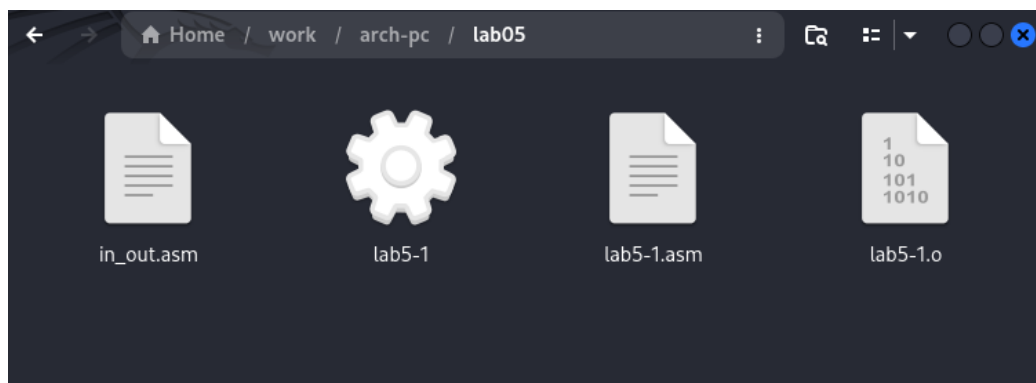


Рисунок 3.7: Скачиваем файл.

3.8 Копируем файл в нужную директорию. рис. 3.8

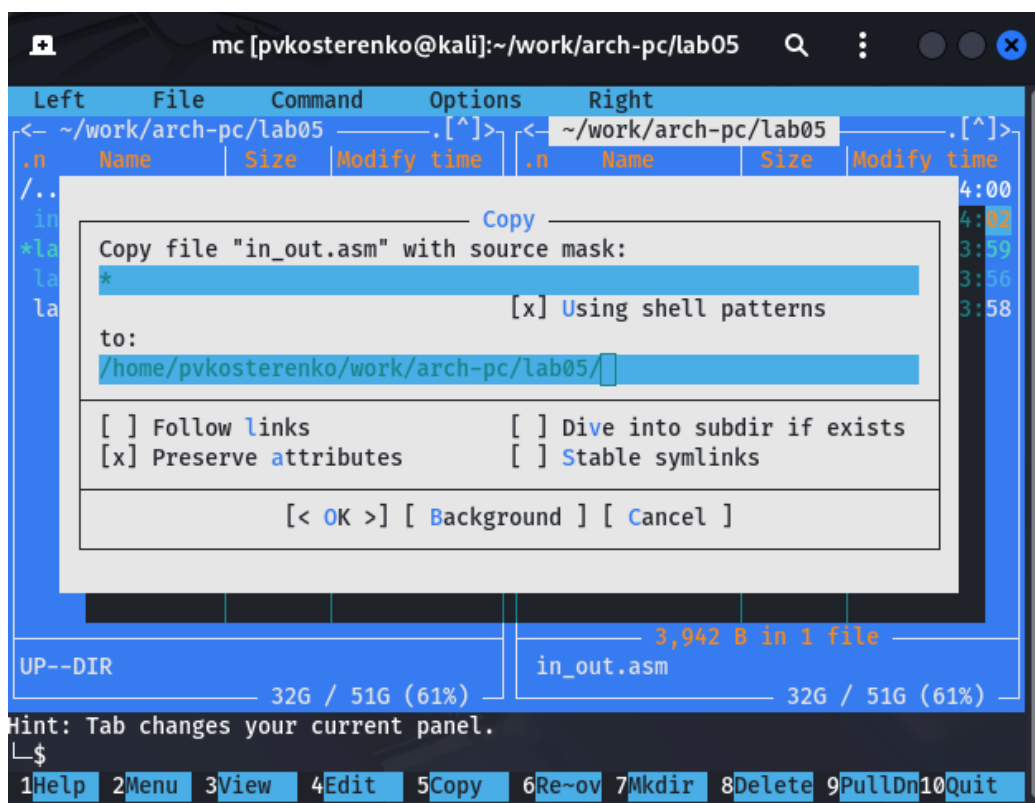


Рисунок 3.8: Копируем скачанный файл.

3.9 Создаем копию файла lab5-1.asm рис. 3.9

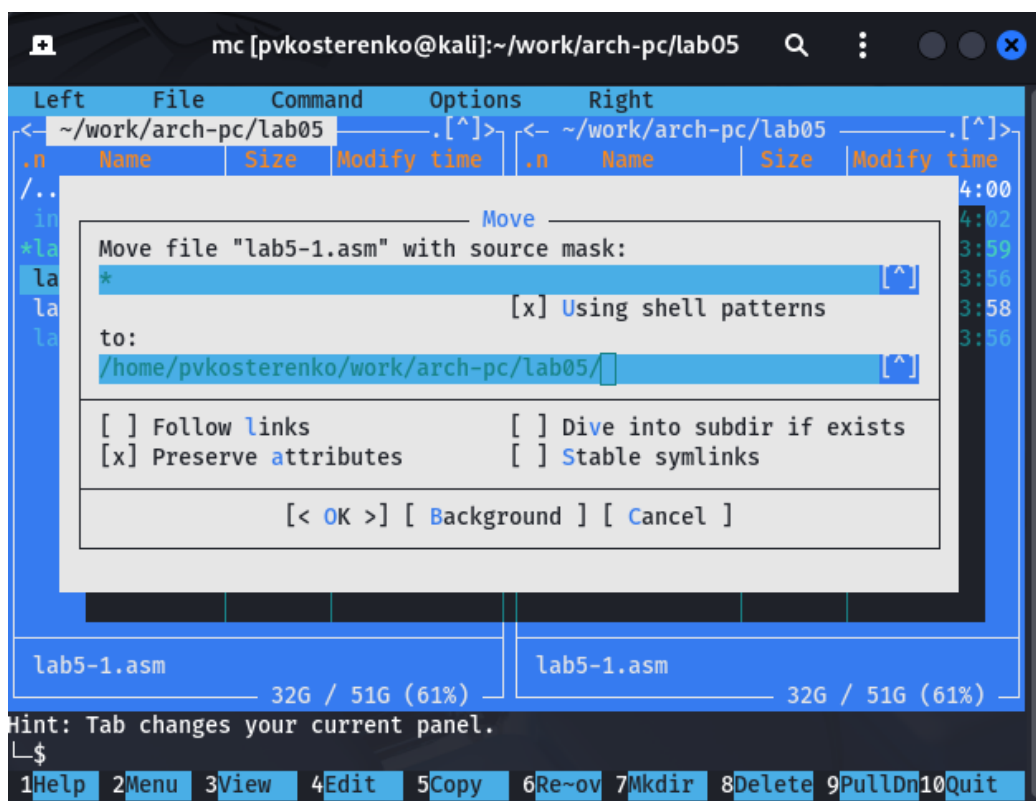


Рисунок 3.9: Создаем копию файла клавишей F6.

3.10 Проверяем созданный файл. рис. 3.10

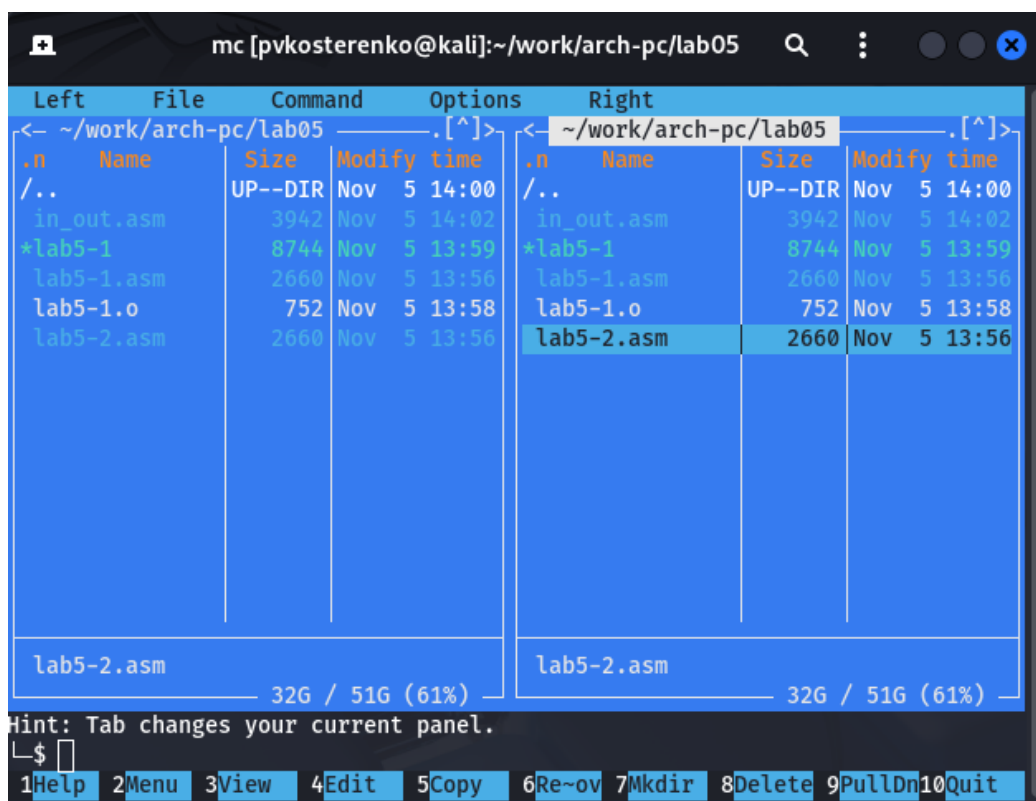
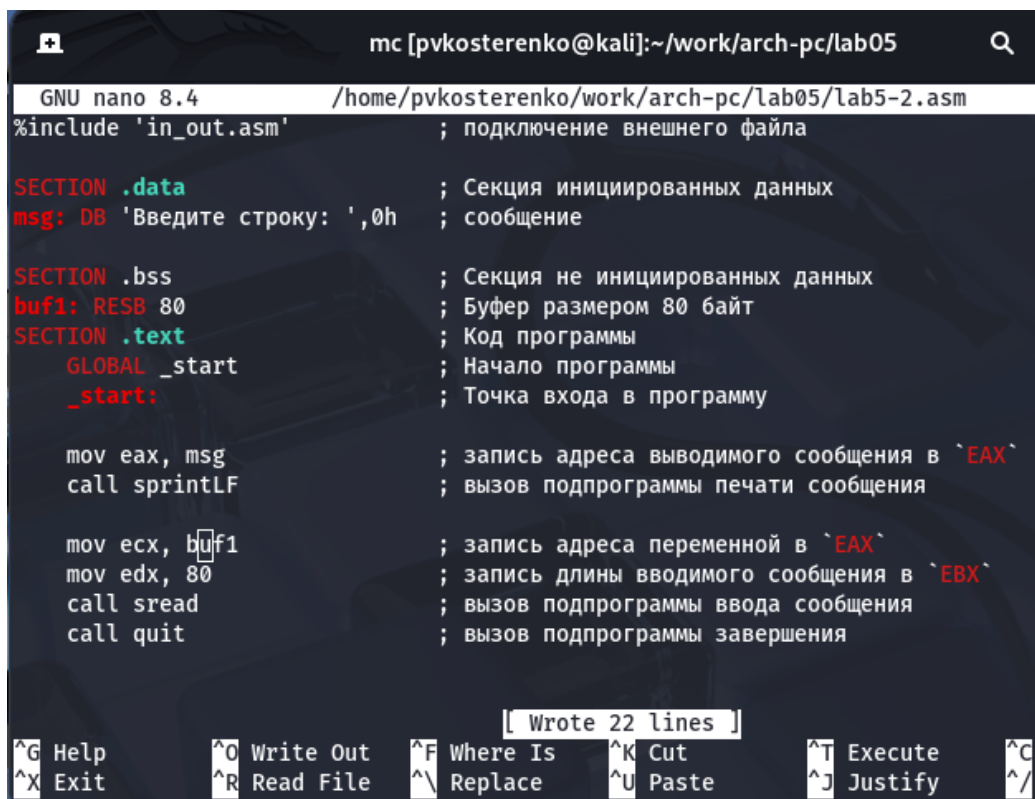


Рисунок 3.10: Проверяем скопировался ли файл.

3.11 Открываем новый файл и заполняем его в соответствии с листингом.

рис. 3.11



```
mc [pvkosterenko@kali]:~/work/arch-pc/lab05
GNU nano 8.4 /home/pvkosterenko/work/arch-pc/lab05/lab5-2.asm
#include 'in_out.asm' ; подключение внешнего файла

SECTION .data ; Секция иницированных данных
msg: DB 'Введите строку: ',0h ; сообщение

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

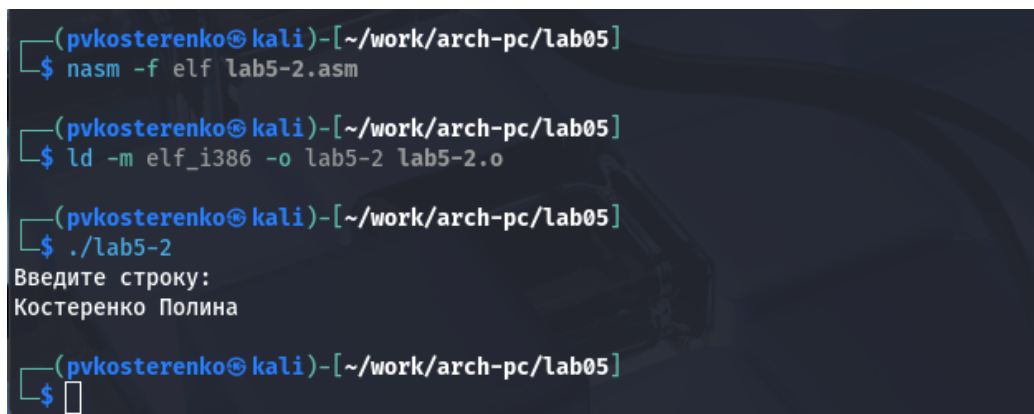
mov eax, msg ; запись адреса выводимого сообщения в `EAX`
call printf ; вызов подпрограммы печати сообщения

mov ecx, buf1 ; запись адреса переменной в `EAX`
mov edx, 80 ; запись длины вводимого сообщения в `EBX`
call read ; вызов подпрограммы ввода сообщения
call quit ; вызов подпрограммы завершения

[Wrote 22 lines]
^G Help ^O Write Out ^F Where Is ^K Cut ^T Execute ^C
^X Exit ^R Read File ^\ Replace ^U Paste ^J Justify ^_
```

Рисунок 3.11: Открываем и заполняем файл.

3.12 Транслируем и запускаем новый файл. рис. 3.12



```
(pvkosterenko@kali)-[~/work/arch-pc/lab05]
$ nasm -f elf lab5-2.asm

(pvkosterenko@kali)-[~/work/arch-pc/lab05]
$ ld -m elf_i386 -o lab5-2 lab5-2.o

(pvkosterenko@kali)-[~/work/arch-pc/lab05]
$ ./lab5-2
Введите строку:
Костеренко Полина

(pvkosterenko@kali)-[~/work/arch-pc/lab05]
$
```

Рисунок 3.12: Смотрим, как сработала программа.

3.13 Снова открываем файл для редактирования и меняем printf на print.

рис. 3.13

```
GNU nano 8.4 /home/pvkosterenko/work/arch-pc/lab05/lab5-2.asm

%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

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

Рисунок 3.13: Редактируем файл.

3.14 Транслируем и запускаем файл. рис. 3.14

```
(pvkosterenko@kali)-[~/work/arch-pc/lab05]
$ nasm -f elf lab5-2.asm

(pvkosterenko@kali)-[~/work/arch-pc/lab05]
$ ld -m elf_i386 -o lab5-2 lab5-2.o

(pvkosterenko@kali)-[~/work/arch-pc/lab05]
$ ./lab5-2
Введите строку: Костеренко Полина

(pvkosterenko@kali)-[~/work/arch-pc/lab05]
$
```

Рисунок 3.14: Сравниваем с прошлой.

Таким образом можем понять, что команда `sprint` выводит текст в той же строке, а `sprintLF` переносит на новую строку.

4 4. Задание для самостоятельной работы.

4.1 Создаем копию файла lab5-1.asm и называем lab5-3.asm. рис. 4.1

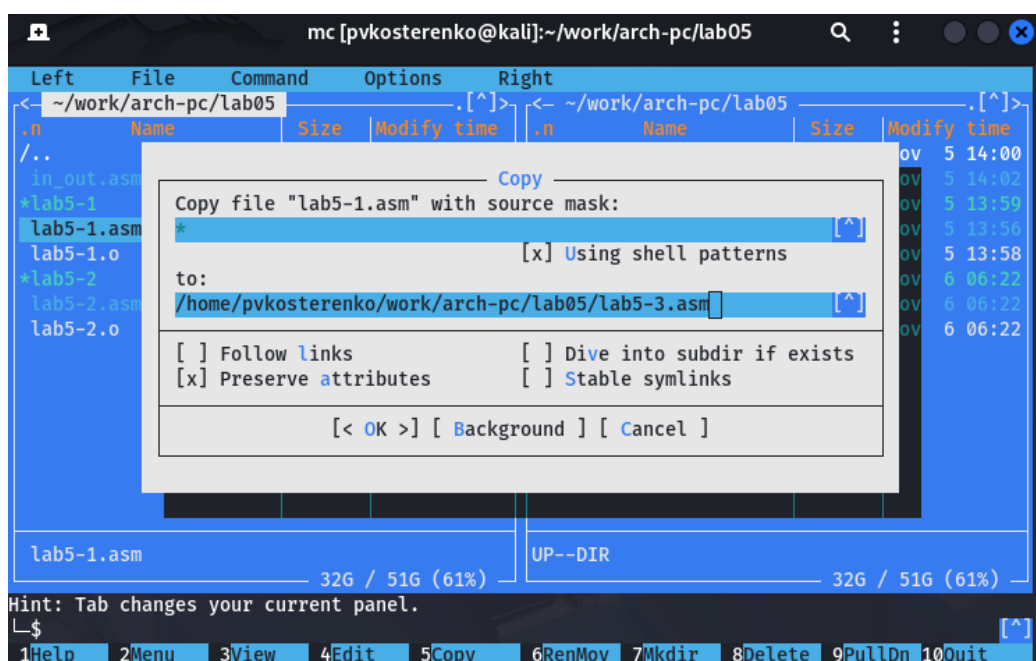


Рисунок 4.1: Создаем копию файла lab5-1.asm.

4.2 Редактируем файл, чтобы введенный текст с клавиатуры выводился в консоль. рис. 4.2

```
mc [pvkosterenko@kali]:~/work/arch-pc/lab05
GNU nano 8.4 /home/pvkosterenko/work/arch-pc/lab05/lab5-3.asm
;-----
; Программа вывода сообщения на экран и ввода строки с клавиатуры
;-----
;----- Объявление 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

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

;----- Системный вызов 'exit' -----
```

Рисунок 4.2: Редактируем файл.

4.3 Транслируем файл и запускаем программу. рис. 4.3

```
(pvkosterenko@kali)-[~/work/arch-pc/lab05]
$ nasm -f elf lab5-3.asm

(pvkosterenko@kali)-[~/work/arch-pc/lab05]
$ ld -m elf_i386 -o lab5-3 lab5-3.o

(pvkosterenko@kali)-[~/work/arch-pc/lab05]
$ ./lab5-3
Введите строку:
Костеренко Полина
Костеренко Полина

(pvkosterenko@kali)-[~/work/arch-pc/lab05]
```

Рисунок 4.3: Проверяем правильность написания программы.

4.4 Создаем копию файла lab5-2.asm и называем его lab5-4.asm. рис. 4.4

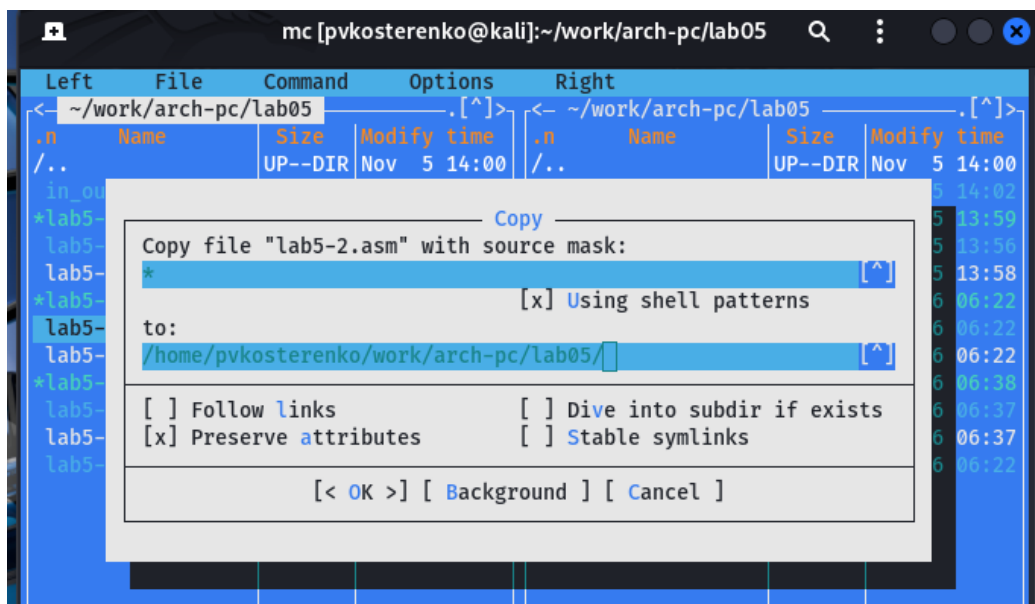


Рисунок 4.4: Создаем копию файла lab5-2.asm.

4.5 Редактируем файл, чтобы введенный текст выводился в консоль. рис. 4.5

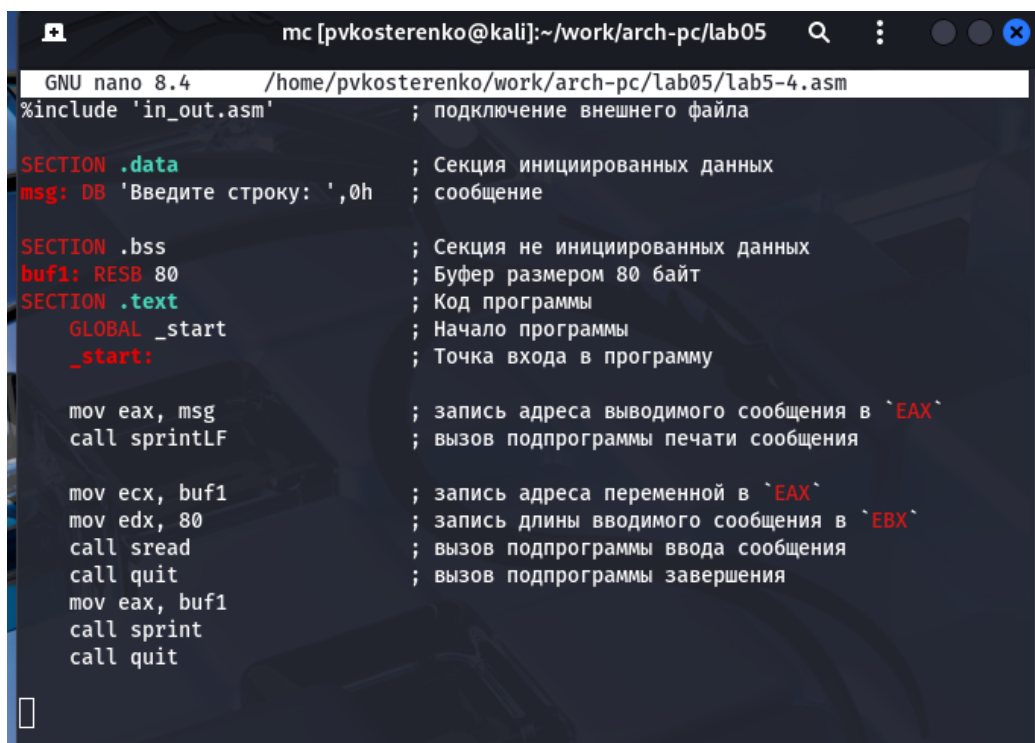
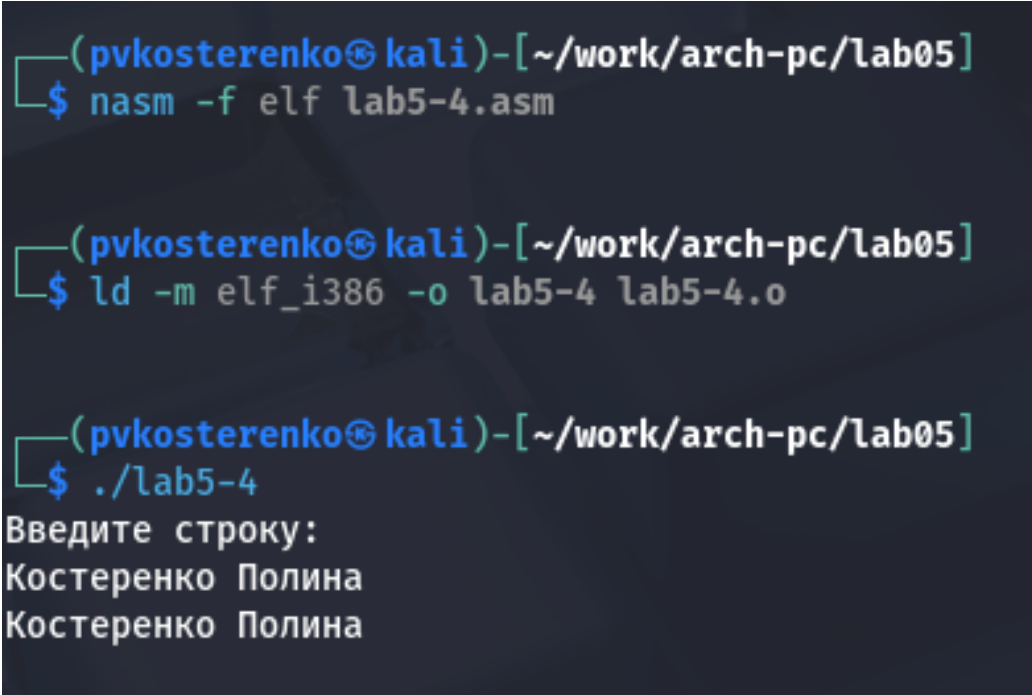


Рисунок 4.5: Редактируем файл.

4.6 Транслируем файл и запускаем программу. рис. 4.6



```
(pvkosterenko@kali)-[~/work/arch-pc/lab05]
$ nasm -f elf lab5-4.asm

(pvkosterenko@kali)-[~/work/arch-pc/lab05]
$ ld -m elf_i386 -o lab5-4 lab5-4.o

(pvkosterenko@kali)-[~/work/arch-pc/lab05]
$ ./lab5-4
Введите строку:
Костеренко Полина
Костеренко Полина
```

Рисунок 4.6: Проверяем правильность написания программы.

5 Выводы

Мы приобрели навыки работы с Midnight Commander и освоили инструкцию mov.