

# **Лабораторная работа №5**

**Основы работы с Midnight Commander (mc). Структура программы  
на языке ассемблера NASM. Системные вызовы в ОС GNU Linux**

Борисенкова София Павловна

# Содержание

<b>1</b>	<b>Цель работы</b>	<b>5</b>
<b>2</b>	<b>Выполнение лабораторной работы</b>	<b>6</b>
<b>3</b>	<b>Выполнение задания для самостоятельной работы</b>	<b>16</b>
<b>4</b>	<b>Выводы</b>	<b>23</b>

# Список иллюстраций

2.1	Интерфейс midnight commander . . . . .	6
2.2	Переход в нужный каталог . . . . .	7
2.3	Создание папки . . . . .	8
2.4	Создание файла lab5-1.asm с помощью команды touch . . . . .	9
2.5	Выбор текстового редактора . . . . .	10
2.6	Редактирование файла lab5-1.asm . . . . .	11
2.7	Сборка исполняемого файла . . . . .	11
2.8	Взаимодействие с программой . . . . .	12
2.9	Копирование файла . . . . .	12
2.10	Копирование файла с помощью F5 . . . . .	13
2.11	Редактирование файла lab5-2.asm . . . . .	14
2.12	Запуск исполняемого файла . . . . .	14
2.13	Изменение файла lab5-2.asm . . . . .	15
2.14	Запуск изменённого файла . . . . .	15
3.1	Создание файла lab5-3.asm . . . . .	16
3.2	Изменение файла lab5-3.asm . . . . .	17
3.3	Создание исполняемого файла . . . . .	18
3.4	Проверка работы программы . . . . .	18
3.5	Создание файла lab5-4.asm . . . . .	19
3.6	Изменение файла lab5-4.asm . . . . .	20
3.7	Создание исполняемого файла . . . . .	21
3.8	Проверка работы программы . . . . .	22

## **Список таблиц**

# 1 Цель работы

Ознакомиться с программой Midnight commander и освоить написание программ на языке ассемблера с помощью инструкций `mov` и `int`



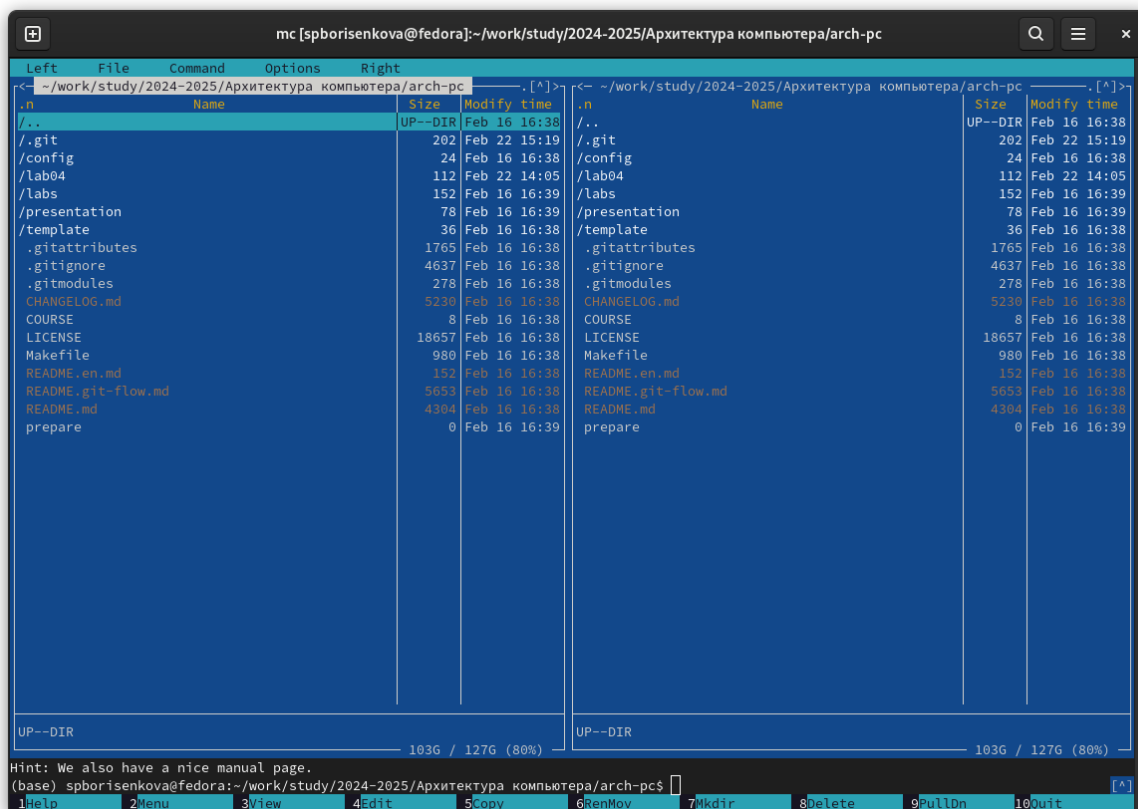


Рис. 2.2: Переход в нужный каталог

Создадим папку lab05 с помощью клавиши F7 (Рис. 2.3):

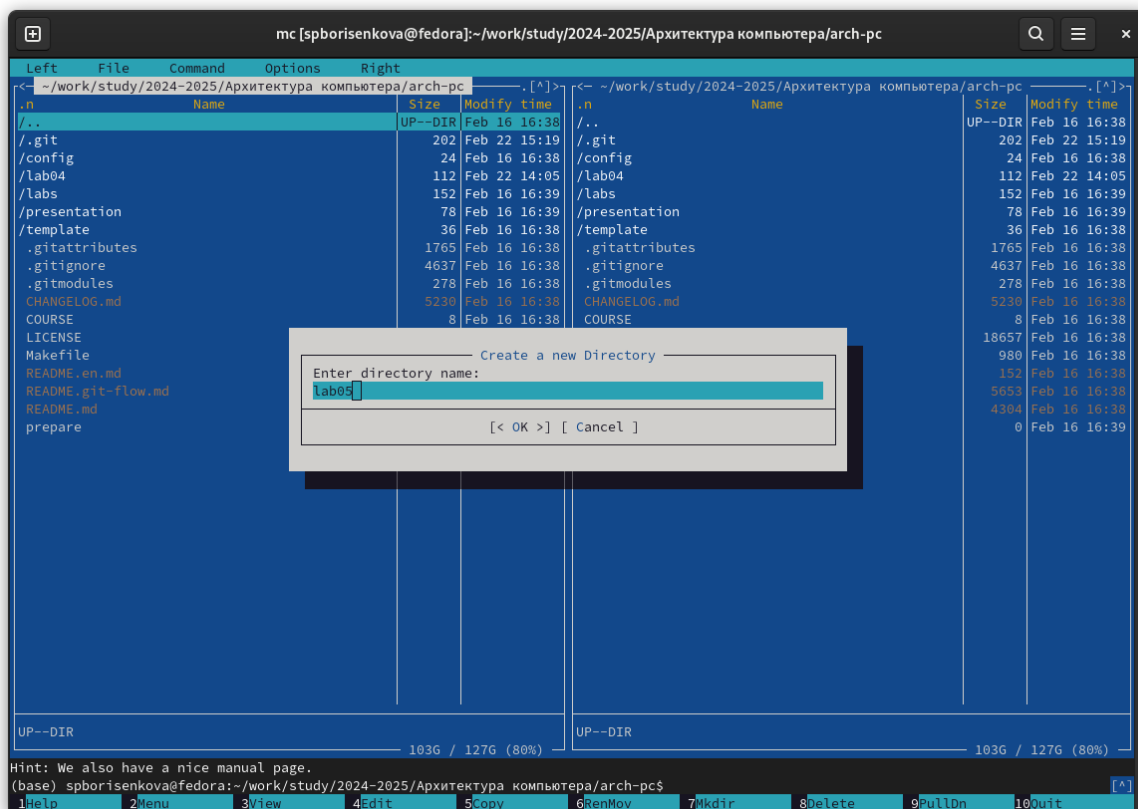


Рис. 2.3: Создание папки

Теперь с помощью команды `touch` создадим файл `lab5-1.asm` (Рис. 2.4):



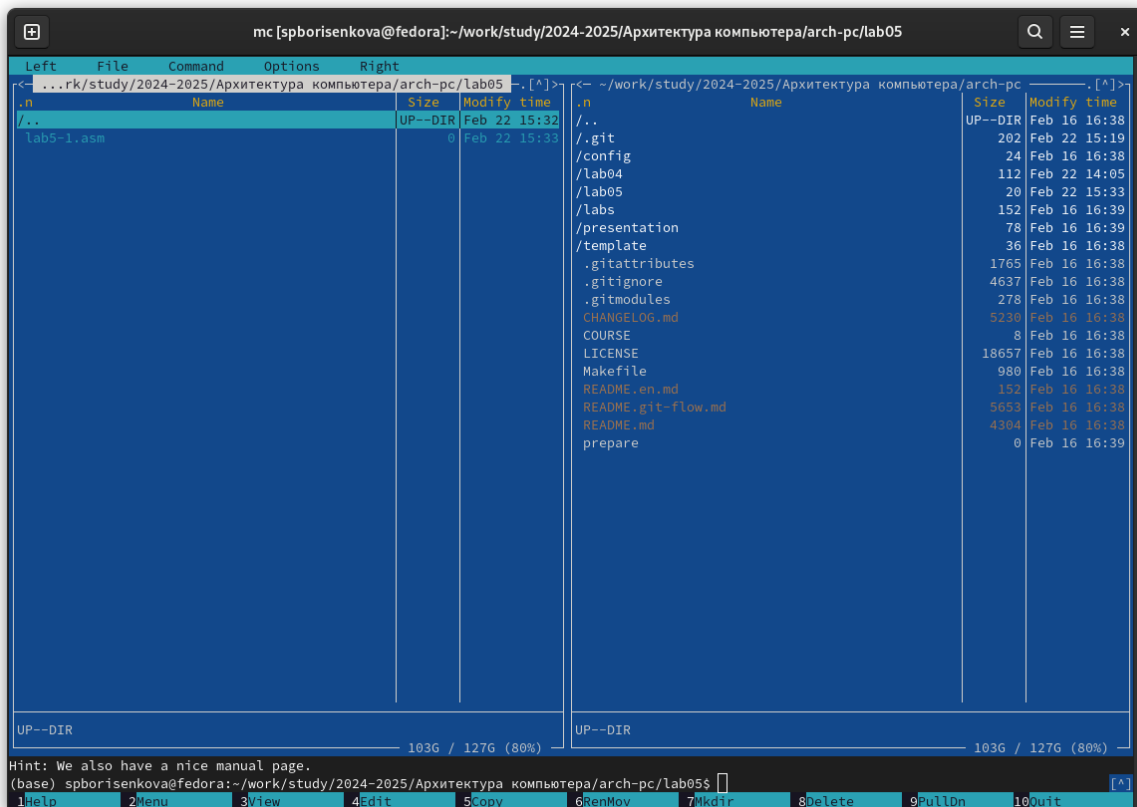


Рис. 2.4: Создание файла lab5-1.asm с помощью команды touch

Теперь откроем только что созданный файл с помощью редактора nano (Рис. 2.5):



Рис. 2.5: Выбор текстового редактора

Теперь отредактируем файл и поместим в него следующий код (Рис. 2.6):

```
mc [spborisenkova@fedora]:~/work/study/2024-2025/Архитектура компьютера/arch-pc/lab05
GNU nano 2.2.2 lab5-1.asm Modified
;
; 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
;----- 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
^G Help      ^O Write Out ^W Where Is  ^K Cut       ^T Execute   ^C Location  ^U Undo      ^A Set Mark  ^J To Bracket
^X Exit      ^R Read File ^\ Replace   ^U Paste     ^D Justify   ^/ Go To Line ^E Redo      ^M Copy      ^Q Where Was
```

Рис. 2.6: Редактирование файла lab5-1.asm

Теперь сохраним его,скомпилируем и соберём (Рис. 2.7):

```
spborisenkova@fedora:~/work/study/2024-2025/Архитектура компьютер... x spborisenkova@fedora:~/work/study/2024-2025/Архитектура компют... x
(base) spborisenkova@fedora:~/work/study/2024-2025/Архитектура компьютера/arch-pc/lab05$ nasm -f elf lab5-1.asm
(base) spborisenkova@fedora:~/work/study/2024-2025/Архитектура компьютера/arch-pc/lab05$ ld -m elf_i386 lab5-1.o -o lab5-1
(base) spborisenkova@fedora:~/work/study/2024-2025/Архитектура компьютера/arch-pc/lab05$ ls
lab5-1 lab5-1.asm lab5-1.o
(base) spborisenkova@fedora:~/work/study/2024-2025/Архитектура компьютера/arch-pc/lab05$
```

Рис. 2.7: Сборка исполняемого файла

После этого запустим получившийся исполняемый файл и введём ФИО (Рис. 2.8):

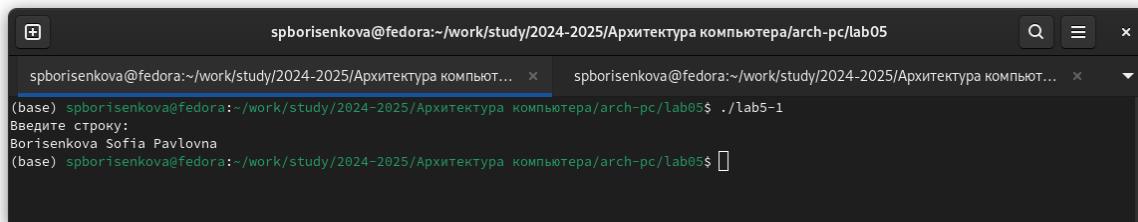


Рис. 2.8: Взаимодействие с программой

После нажатия Enter программа завершится и ничего не произойдёт. Теперь скачаем файл in\_out.asm, откроем папку с ним в правой панели и скопируем его в нашу рабочую папку с помощью F6 (Рис. 2.9):

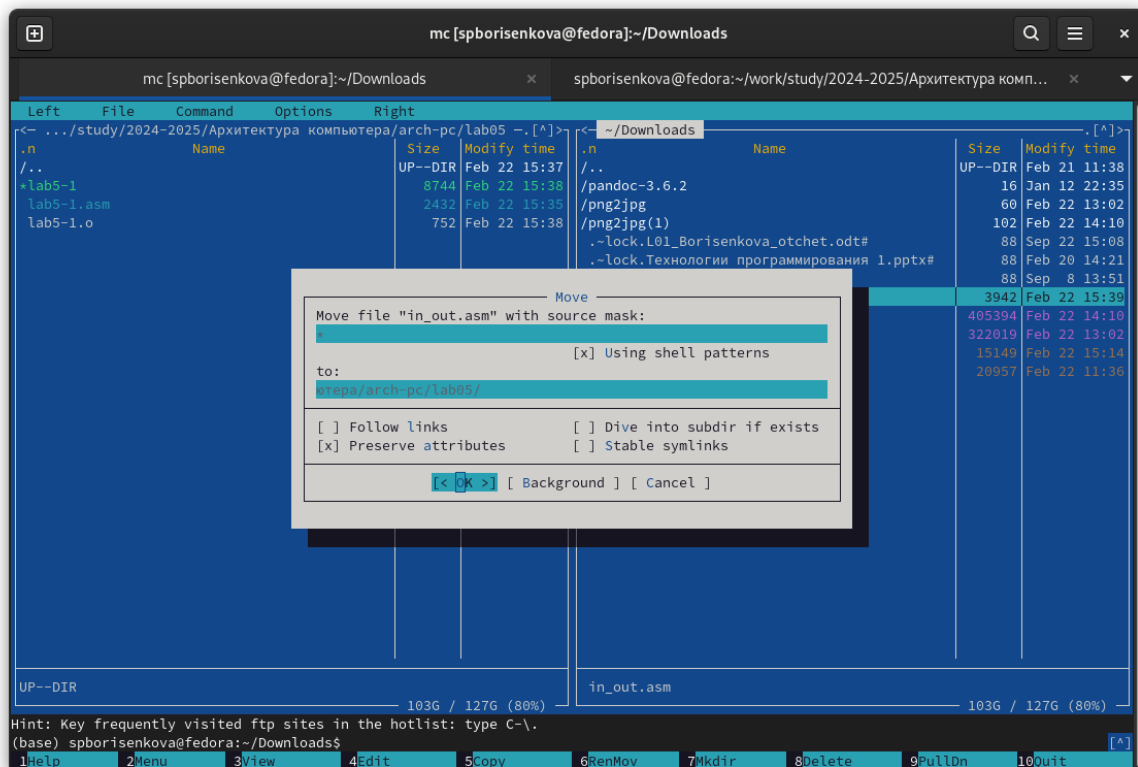


Рис. 2.9: Копирование файла

Теперь сделаем копию файла lab5-1.asm с помощью команды F5. Назовём копию lab5-2.asm (Рис. 2.10):

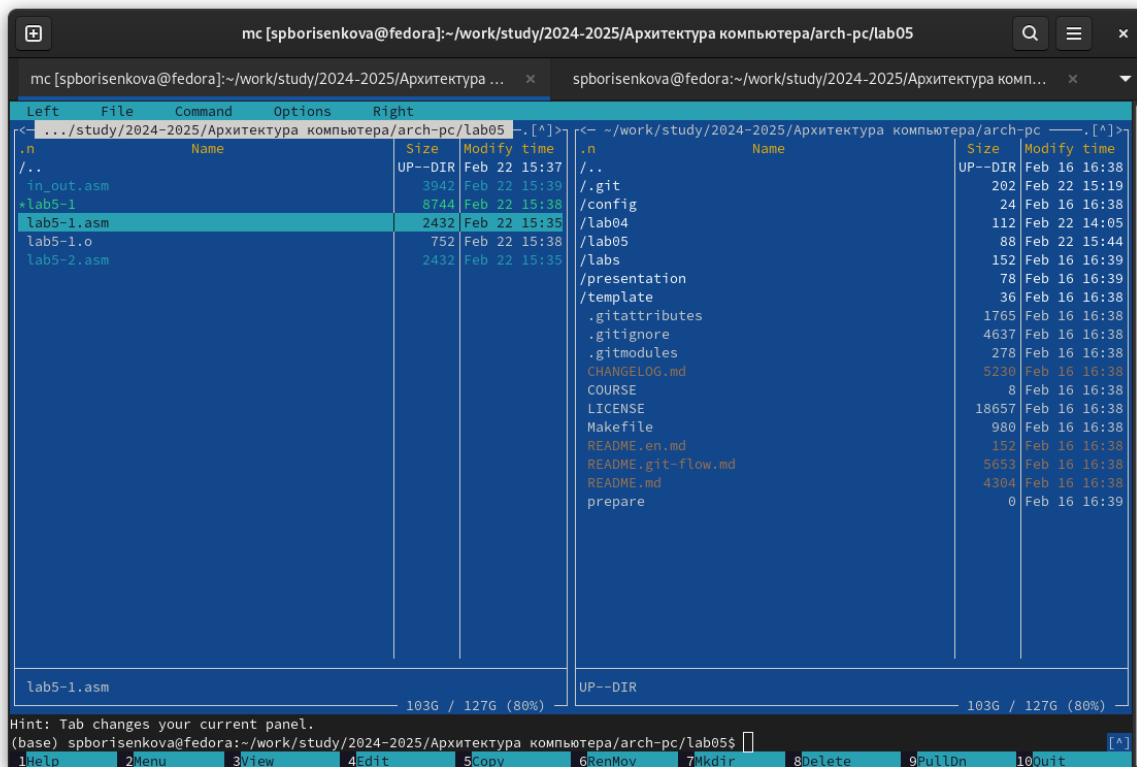


Рис. 2.10: Копирование файла с помощью F5

Откроем в текстовом редакторе файл lab5-2.asm и напишем туда следующий код (Рис. 2.11):

```
mc [spborisenkova@fedora]:~/work/study/2024-2025/Архитектура компьютера/arch-pc/lab05
GNU nano 7.2 lab5-2.asm Modified
;-----
; 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 sprintf ; 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
```

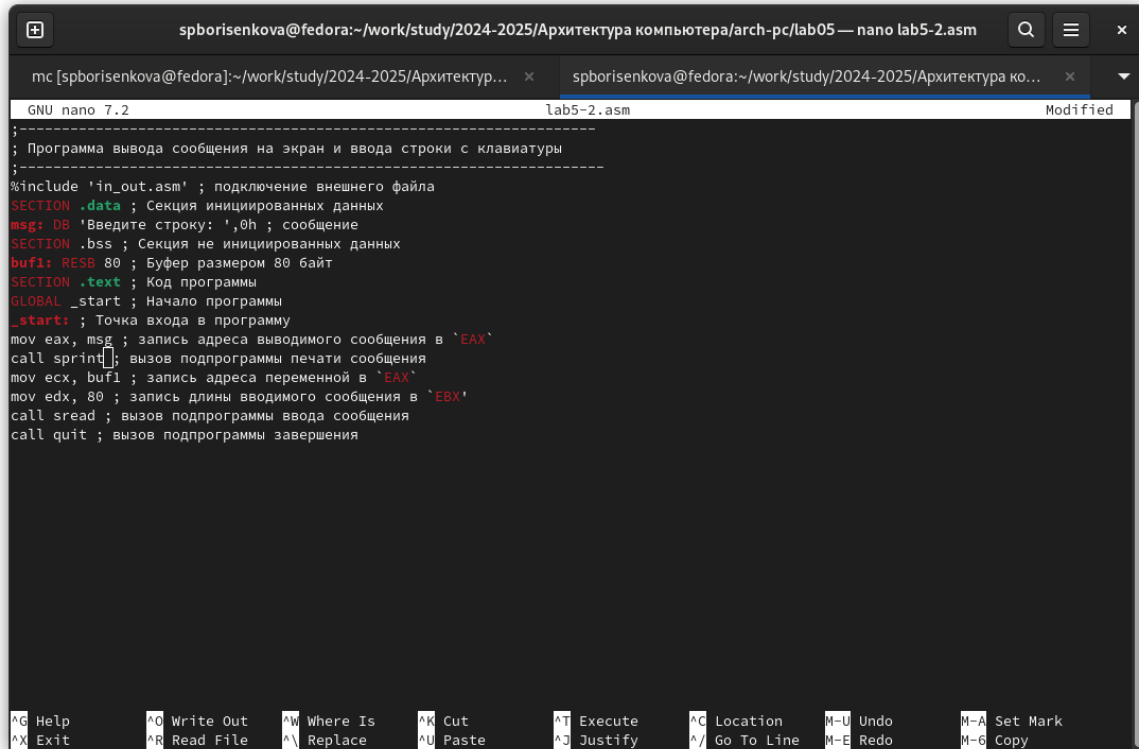
Рис. 2.11: Редактирование файла lab5-2.asm

После чего создадим исполняемый файл с помощью `nasm` и `ld` и запустим созданный файл (Рис. 2.12):

```
spborisenkova@fedora:~/work/study/2024-2025/Архитектура компьютера/arch-pc/lab05
(base) spborisenkova@fedora:~/work/study/2024-2025/Архитектура компьютера/arch-pc/lab05$ nasm -f elf lab5-2.asm
(base) spborisenkova@fedora:~/work/study/2024-2025/Архитектура компьютера/arch-pc/lab05$ ld -m elf_i386 lab5-2.o -o lab5-2
(base) spborisenkova@fedora:~/work/study/2024-2025/Архитектура компьютера/arch-pc/lab05$ ls
in_out.asm lab5-1 lab5-1.asm lab5-1.o lab5-2 lab5-2.asm lab5-2.o
(base) spborisenkova@fedora:~/work/study/2024-2025/Архитектура компьютера/arch-pc/lab05$ ./lab5-2
Введите строку:
Borisenkova Sofia Pavlovna
(base) spborisenkova@fedora:~/work/study/2024-2025/Архитектура компьютера/arch-pc/lab05$
```

Рис. 2.12: Запуск исполняемого файла

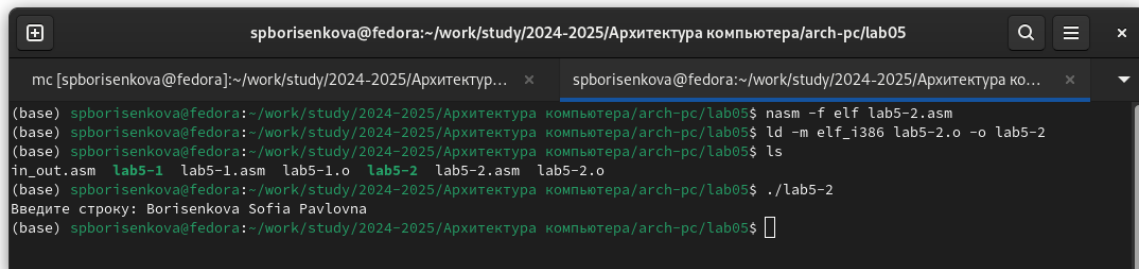
Попробуем теперь вместо команды `sprintf` использовать просто команду `sprint` (Рис. 2.13):



```
GNU nano 7.2 lab5-2.asm Modified
;-----
; 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. 2.13: Изменение файла lab5-2.asm

Точно также соберём исполняемый файл и запустим его (Рис. 2.14):



```
(base) spborisenkova@fedora:~/work/study/2024-2025/Архитектура компьютера/arch-pc/lab05$ nasm -f elf lab5-2.asm
(base) spborisenkova@fedora:~/work/study/2024-2025/Архитектура компьютера/arch-pc/lab05$ ld -m elf_i386 lab5-2.o -o lab5-2
(base) spborisenkova@fedora:~/work/study/2024-2025/Архитектура компьютера/arch-pc/lab05$ ls
in_out.asm lab5-1 lab5-1.asm lab5-1.o lab5-2 lab5-2.asm lab5-2.o
(base) spborisenkova@fedora:~/work/study/2024-2025/Архитектура компьютера/arch-pc/lab05$ ./lab5-2
Введите строку: Borisenkova Sofia Pavlovna
(base) spborisenkova@fedora:~/work/study/2024-2025/Архитектура компьютера/arch-pc/lab05$
```

Рис. 2.14: Запуск изменённого файла

Теперь нет переноса на следующую строку. Этим и отличаются команды `sprintf` от `sprint`. Первая добавляет перенос после текста, а вторая нет

### 3 Выполнение задания для самостоятельной работы

Теперь создадим с помощью копию файла lab5-1.asm (Рис. 3.1):

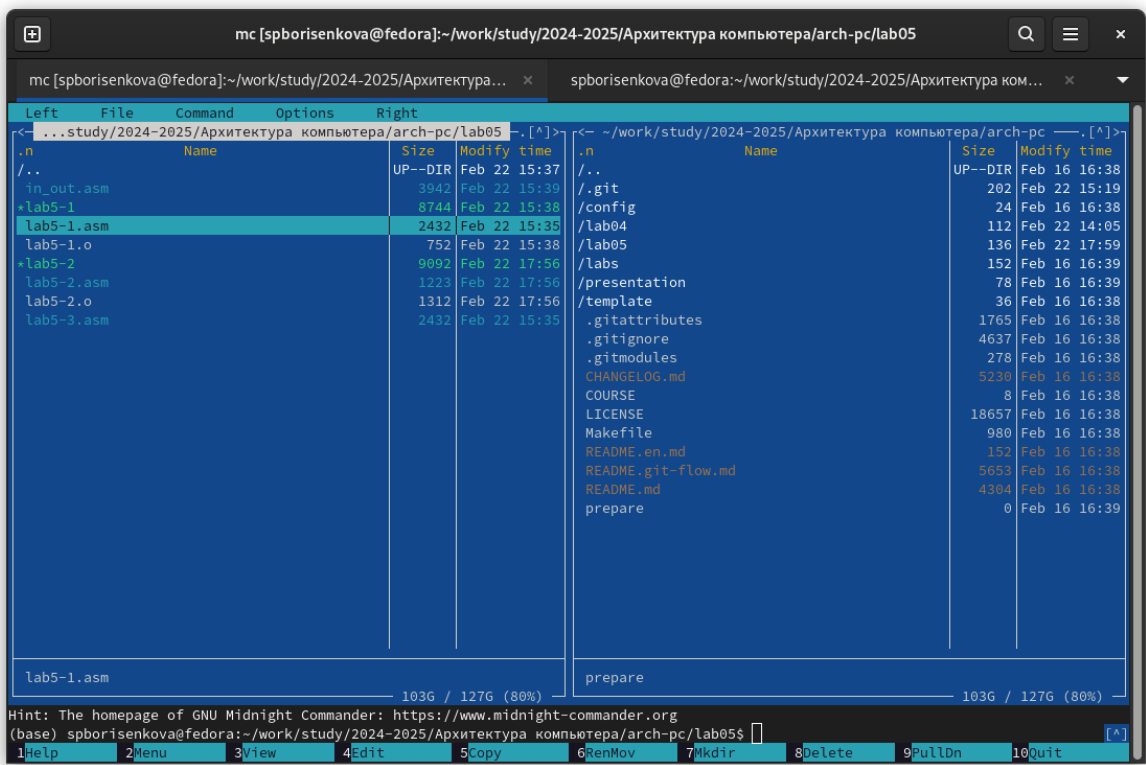


Рис. 3.1: Создание файла lab5-3.asm

Изменим копию так, чтобы она выводила тот текст, который получила на ввод. Для этого перед системным вызовом `exit` вставим текст с системным вызовом `write`. Он очень похож на системный вызов `write`, который уже был в коде, но есть



несколько отличий. Так, мы перемещаем адрес строки buf1 в ecx и размер строки buf1 (80) в edx (Рис. 3.2):

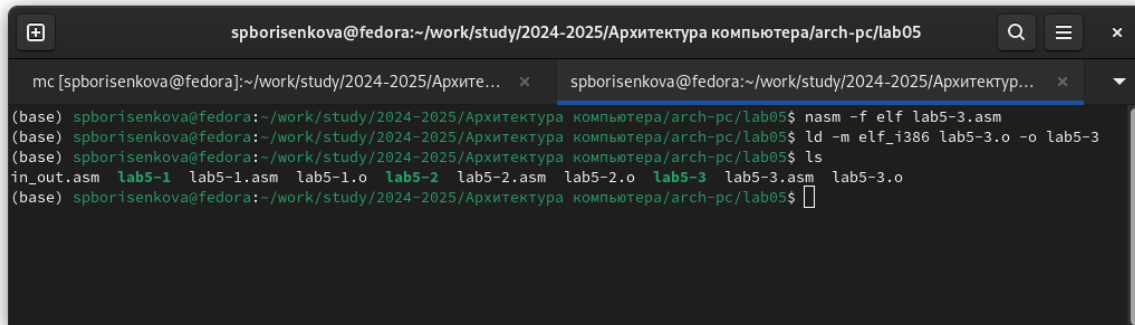
```

GNU nano 7.2 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
;----- Вывод полученной строки -----
mov eax,4 ;
mov ebx,1 ;
mov ecx,buf1 ;
mov edx,80 ;
int 80h ;
;----- 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

^G Help      ^O Write Out  ^W Where Is   ^K Cut        ^T Execute    ^C Location   M-U Undo      M-A Set Mark
^X Exit      ^R Read File  ^\ Replace    ^U Paste      ^J Justify    ^_/ Go To Line M-E Redo      M-6 Copy
  
```

Рис. 3.2: Изменение файла lab5-3.asm

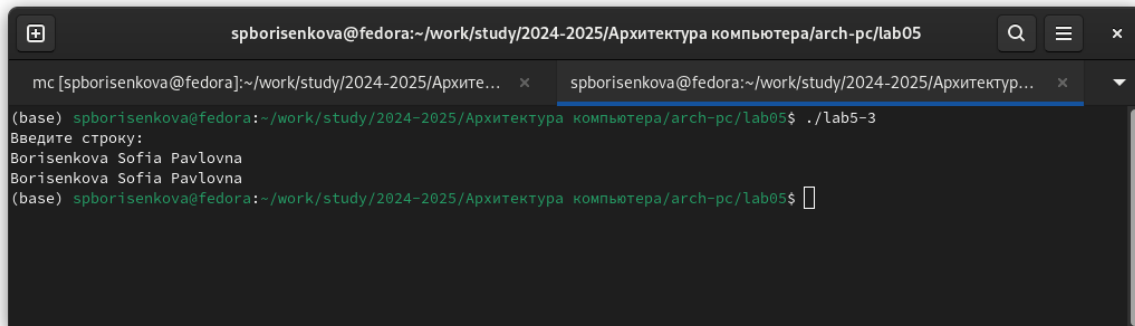
Сохраним изменения и создадим исполняемый файл (Рис. 3.3):



```
spborisenkova@fedora:~/work/study/2024-2025/Архитектура компьютера/arch-pc/lab05
mc [spborisenkova@fedora:~/work/study/2024-2025/Архите... x spborisenkova@fedora:~/work/study/2024-2025/Архитектур... x
(base) spborisenkova@fedora:~/work/study/2024-2025/Архитектура компьютера/arch-pc/lab05$ nasm -f elf lab5-3.asm
(base) spborisenkova@fedora:~/work/study/2024-2025/Архитектура компьютера/arch-pc/lab05$ ld -m elf_i386 lab5-3.o -o lab5-3
(base) spborisenkova@fedora:~/work/study/2024-2025/Архитектура компьютера/arch-pc/lab05$ ls
in_out.asm lab5-1 lab5-1.asm lab5-1.o lab5-2 lab5-2.asm lab5-2.o lab5-3 lab5-3.asm lab5-3.o
(base) spborisenkova@fedora:~/work/study/2024-2025/Архитектура компьютера/arch-pc/lab05$
```

Рис. 3.3: Создание исполняемого файла

Запустим его и проверим, что всё работает (Рис. 3.4):



```
spborisenkova@fedora:~/work/study/2024-2025/Архитектура компьютера/arch-pc/lab05
mc [spborisenkova@fedora:~/work/study/2024-2025/Архите... x spborisenkova@fedora:~/work/study/2024-2025/Архитектур... x
(base) spborisenkova@fedora:~/work/study/2024-2025/Архитектура компьютера/arch-pc/lab05$ ./lab5-3
Введите строку:
Borisenkova Sofia Pavlovna
Borisenkova Sofia Pavlovna
(base) spborisenkova@fedora:~/work/study/2024-2025/Архитектура компьютера/arch-pc/lab05$
```

Рис. 3.4: Проверка работы программы

Теперь создадим с помощью F5 копию файла lab5-2.asm (Рис. 3.5):

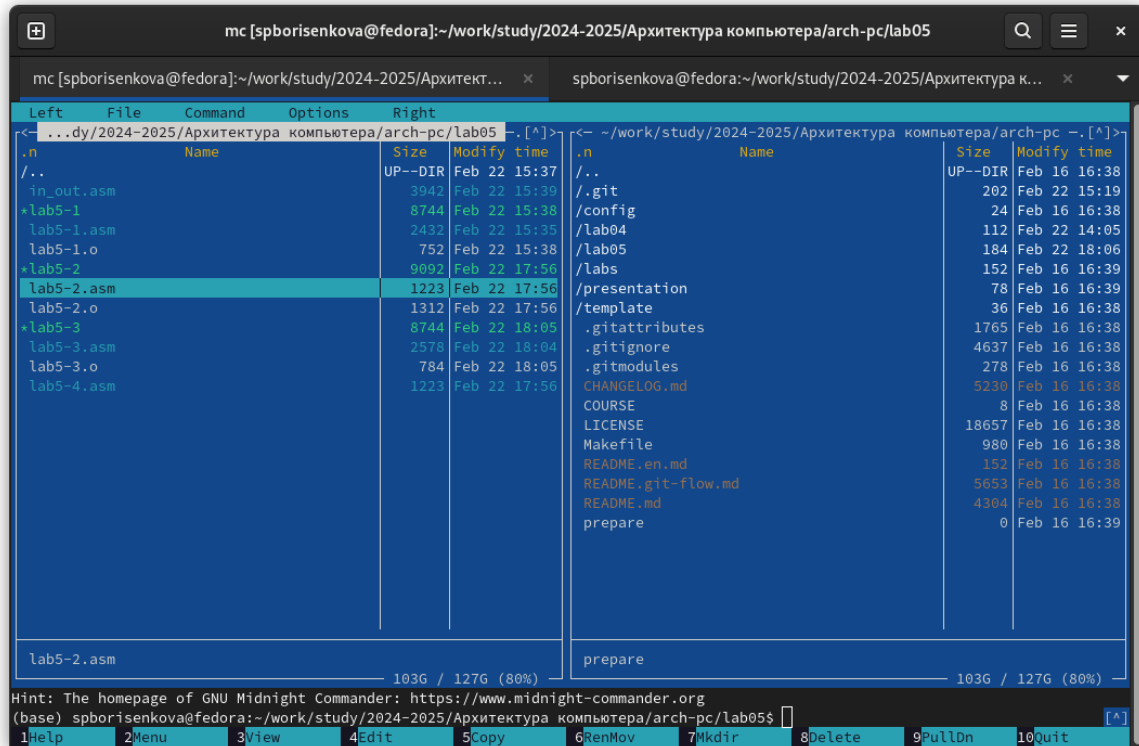
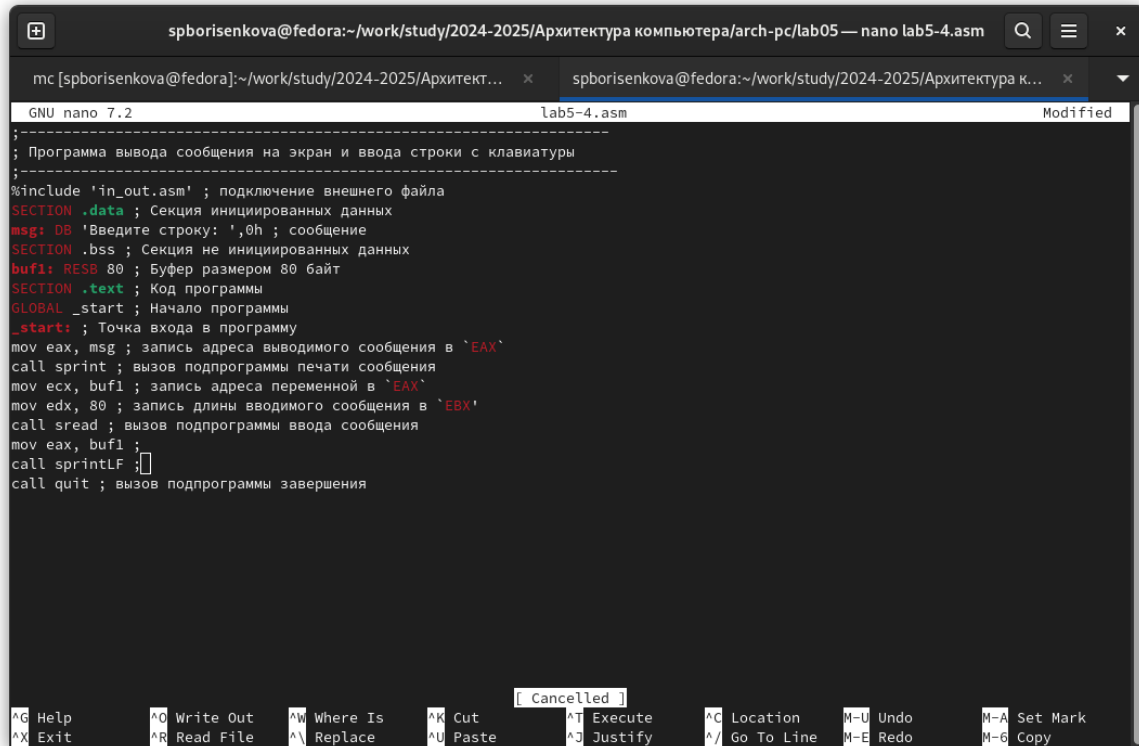


Рис. 3.5: Создание файла lab5-4.asm

Сделаем так, чтобы этот код также выводил тот текст, что получит на ввод. Для этого перед последней строкой добавим строку, которая записывает в еах адрес buf1, а также строку, которая вызывает подпрограмму sprintLF (Рис. 3.6):

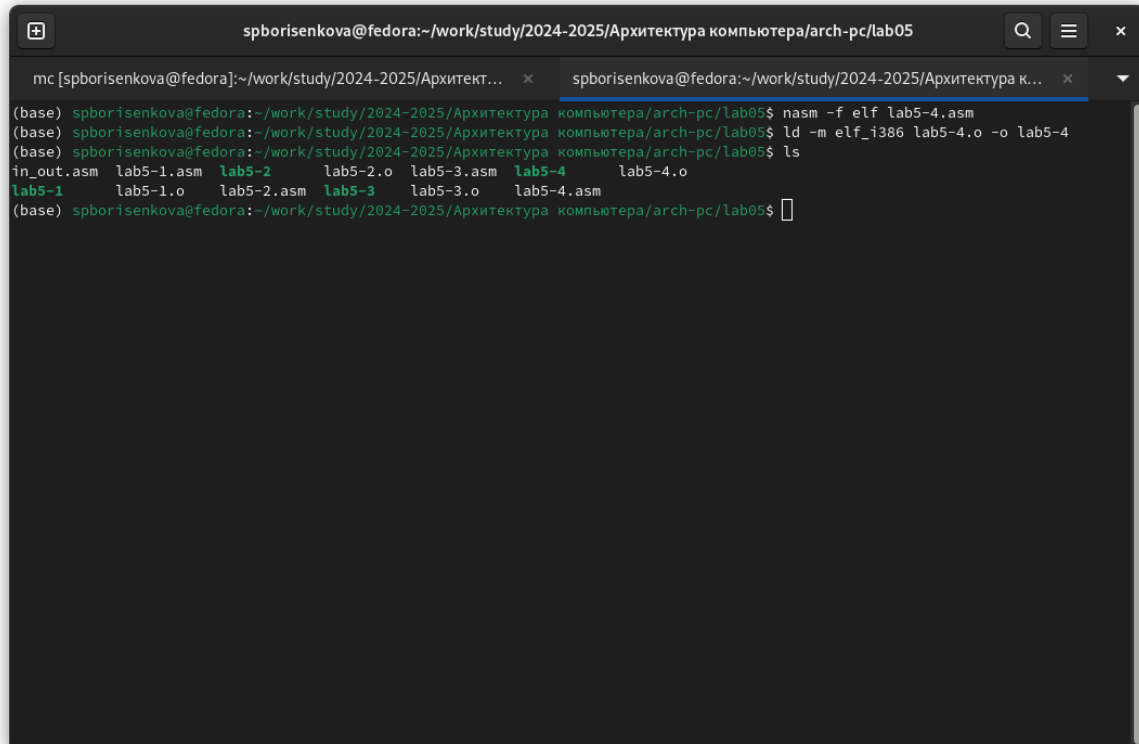


```
spborisenkova@fedora:~/work/study/2024-2025/Архитектура компьютера/arch-pc/lab05 — nano lab5-4.asm
mc [spborisenkova@fedora]:~/work/study/2024-2025/Архитект... x spborisenkova@fedora:~/work/study/2024-2025/Архитектура к... x
GNU nano 7.2 lab5-4.asm Modified
;-----
; Программа вывода сообщения на экран и ввода строки с клавиатуры
;-----
#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 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 ;
call sprintLF ;
call quit ; вызов подпрограммы завершения

[ Cancelled ]
^G Help      ^O Write Out  ^W Where Is   ^K Cut        ^T Execute    ^C Location   M-U Undo      M-A Set Mark
^X Exit      ^R Read File  ^\ Replace    ^U Paste      ^J Justify    ^_ Go To Line  M-E Redo      M-6 Copy
```

Рис. 3.6: Изменение файла lab5-4.asm

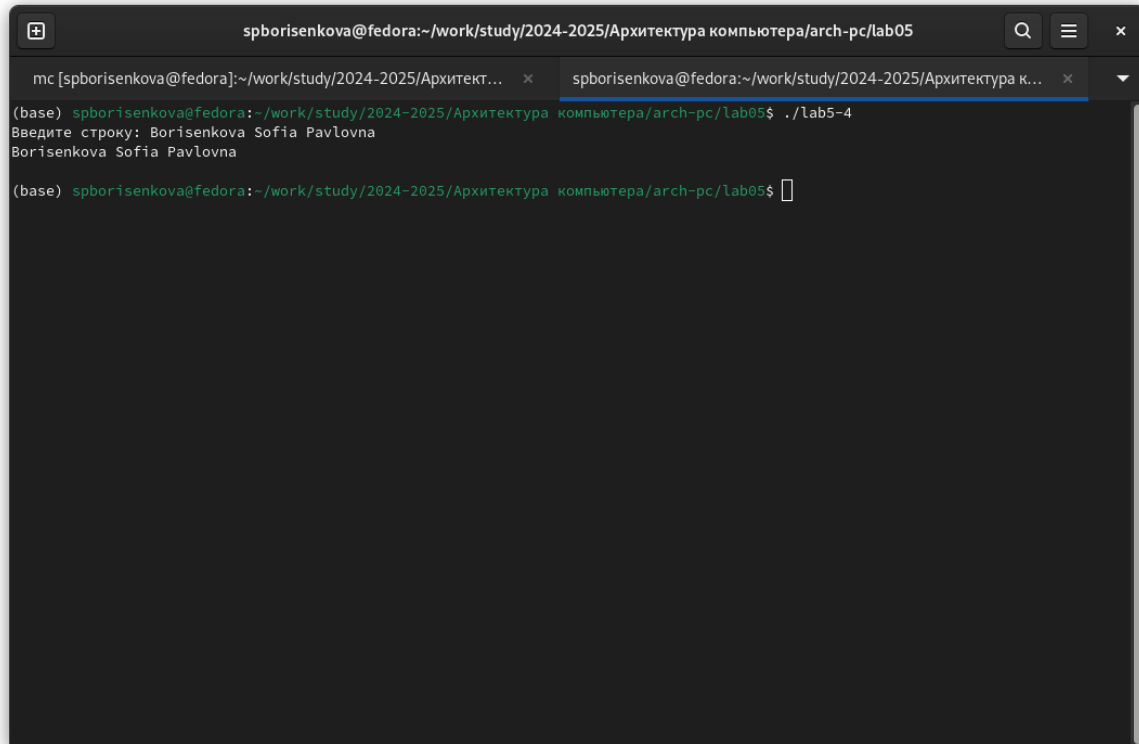
Теперь создадим исполняемый файл (Рис. 3.7):



```
spborisenkova@fedora:~/work/study/2024-2025/Архитектура компьютера/arch-pc/lab05
mc [spborisenkova@fedora]:~/work/study/2024-2025/Архитект... x spborisenkova@fedora:~/work/study/2024-2025/Архитектура к... x
(base) spborisenkova@fedora:~/work/study/2024-2025/Архитектура компьютера/arch-pc/lab05$ nasm -f elf lab5-4.asm
(base) spborisenkova@fedora:~/work/study/2024-2025/Архитектура компьютера/arch-pc/lab05$ ld -m elf_i386 lab5-4.o -o lab5-4
(base) spborisenkova@fedora:~/work/study/2024-2025/Архитектура компьютера/arch-pc/lab05$ ls
in_out.asm  lab5-1.asm  lab5-2      lab5-2.o    lab5-3.asm  lab5-4      lab5-4.o
lab5-1      lab5-1.o    lab5-2.asm  lab5-3      lab5-3.o    lab5-4.asm
(base) spborisenkova@fedora:~/work/study/2024-2025/Архитектура компьютера/arch-pc/lab05$
```

Рис. 3.7: Создание исполняемого файла

Теперь запустим программу и убедимся, что она работает (Рис. 3.8):



The image shows a terminal window with a dark background. The title bar at the top reads "spborisenkova@fedora:~/work/study/2024-2025/Архитектура компьютера/arch-pc/lab05". The terminal content shows a command prompt "(base) spborisenkova@fedora:~/work/study/2024-2025/Архитектура компьютера/arch-pc/lab05\$ ./lab5-4". The output of the command is "Введите строку: Borisenkova Sofia Pavlovna" followed by "Borisenkova Sofia Pavlovna" on the next line. The prompt then returns to "(base) spborisenkova@fedora:~/work/study/2024-2025/Архитектура компьютера/arch-pc/lab05\$".

```
mc [spborisenkova@fedora]:~/work/study/2024-2025/Архитект... x spborisenkova@fedora:~/work/study/2024-2025/Архитектура к... x
(base) spborisenkova@fedora:~/work/study/2024-2025/Архитектура компьютера/arch-pc/lab05$ ./lab5-4
Введите строку: Borisenkova Sofia Pavlovna
Borisenkova Sofia Pavlovna
(base) spborisenkova@fedora:~/work/study/2024-2025/Архитектура компьютера/arch-pc/lab05$
```

Рис. 3.8: Проверка работы программы

## **4 Выводы**

В результате выполнения работы были получены навыки работы с Midnight commander, а также навыки написания простых программ ввода-вывода на языке ассемблера