

# **Лабораторная работа №5**

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

Хасанов Тимур

# Содержание

1	Цель работы	5
2	Выполнение лабораторной работы	6
3	Выполнение задания для самостоятельной работы	19
4	Выводы	23

# Список иллюстраций

2.1	Запуск Midnight commander . . . . .	6
2.2	Интерфейс midnight commander . . . . .	6
2.3	Переход в нужный каталог . . . . .	7
2.4	Создание папки . . . . .	8
2.5	Создание файла lab5-1.asm с помощью команды touch прямо в mc	9
2.6	Выбор текстового редактора . . . . .	10
2.7	Редактирование файла lab5-1.asm . . . . .	11
2.8	Проверка успешного редактирования . . . . .	12
2.9	Компиляция файла с помощью nasm . . . . .	12
2.10	Сборка исполняемого файла с помощью ld . . . . .	13
2.11	Запуск исполняемого файла . . . . .	13
2.12	Взаимодействие с программой . . . . .	13
2.13	Открытие папки с файлом in_out.asm в правой панели . . . . .	14
2.14	Перемещение файла с помощью F6 . . . . .	15
2.15	Копирование файла с помощью F5 . . . . .	16
2.16	Текущий вид рабочей папки . . . . .	16
2.17	Редактирование файла lab5-2.asm . . . . .	17
2.18	Создание исполняемого файла . . . . .	17
2.19	Запуск исполняемого файла . . . . .	17
2.20	Изменение файла lab5-2.asm . . . . .	18
2.21	Запуск изменённого файла . . . . .	18
3.1	Создание копии файла lab5-1.asm . . . . .	19
3.2	Изменение файла lab5-1-1.asm . . . . .	20
3.3	Проверка работы программы . . . . .	20
3.4	Создание копии файла lab5-2.asm . . . . .	21
3.5	Изменение файла lab5-2-1.asm . . . . .	21
3.6	Создание исполняемого файла . . . . .	22

## **Список таблиц**

# 1 Цель работы

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

## 2 Выполнение лабораторной работы

Для начала выполнения лабораторной работы нам необходимо открыть Midnight commander с помощью команды mc (Рис. 2.1):

```
[tihasanov@tihasanov ~]$ mc
```

Рис. 2.1: Запуск Midnight commander

После ввода команды мы увидим такой интерфейс (Рис. 2.2):

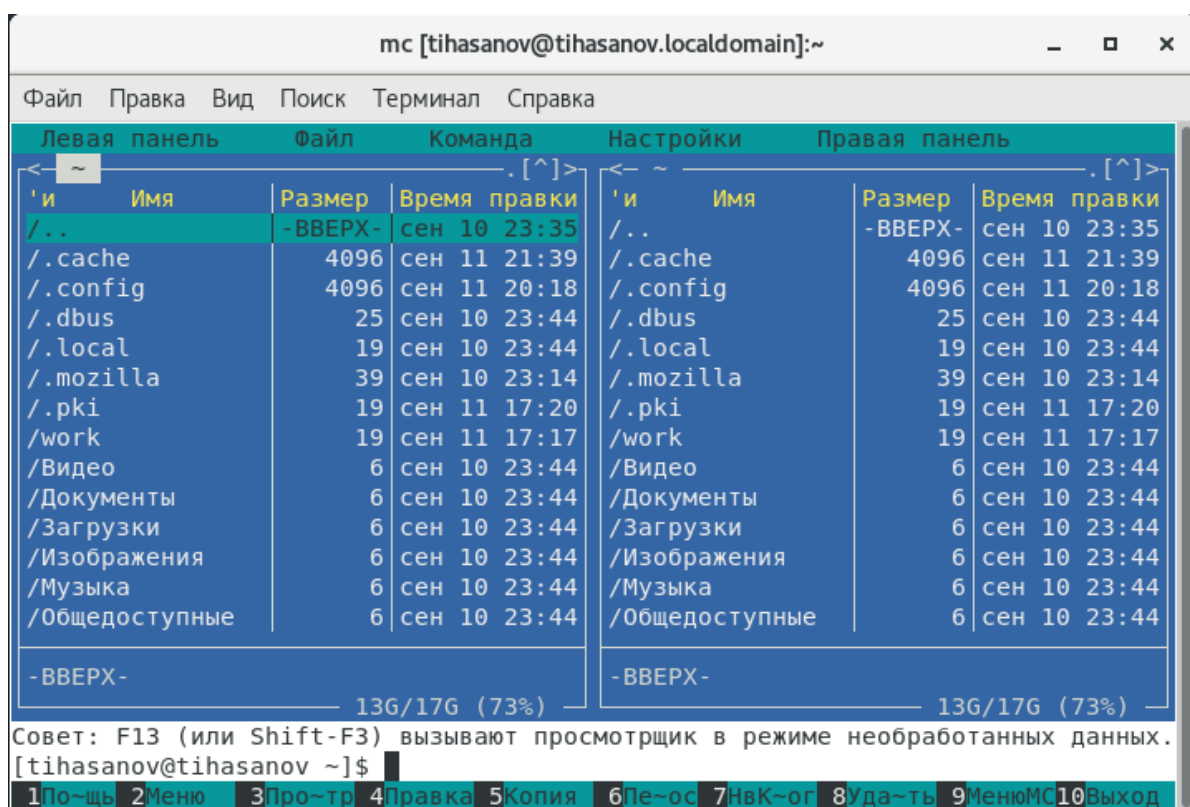


Рис. 2.2: Интерфейс midnight commander

С помощью стрелок и клавиши Enter перейдём в каталог с репозиторием `~/work/study/2023-2024/Архитектура компьютера/-study_2023-2024_arh-pc` (Рис. 2.3):

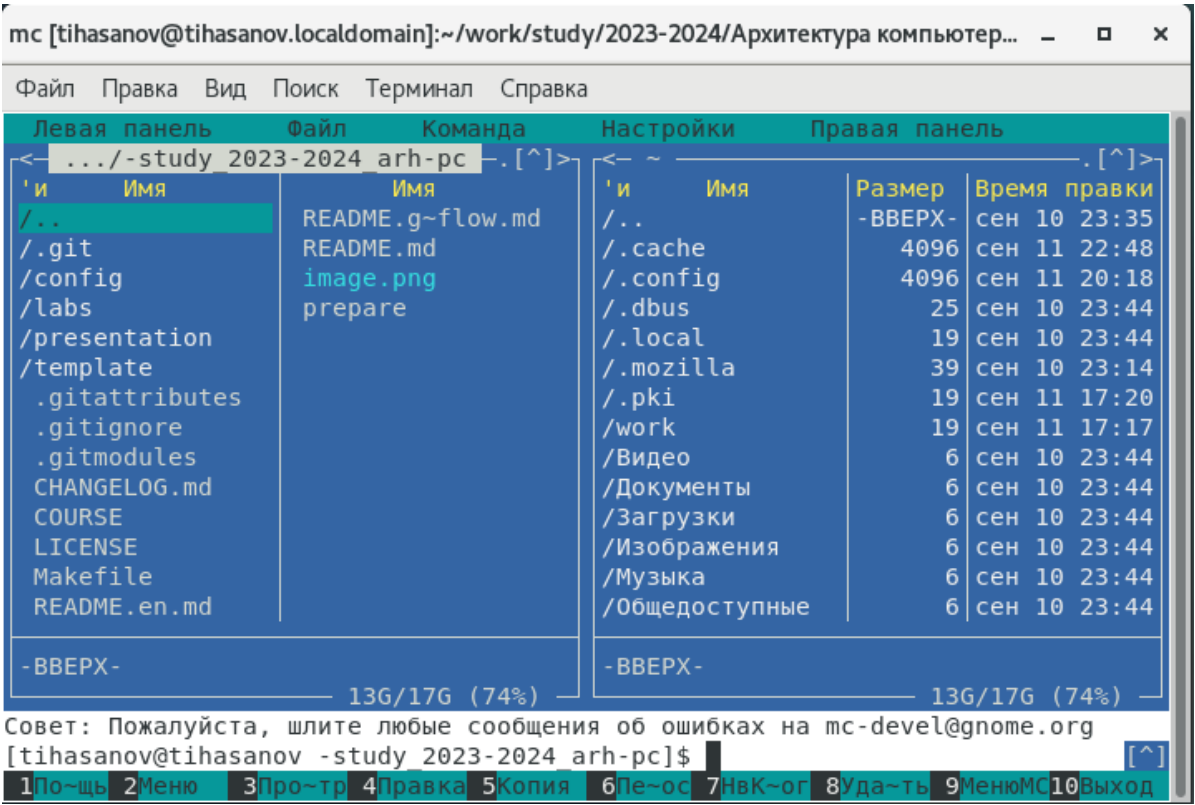


Рис. 2.3: Переход в нужный каталог

Далее с помощью утилиты `cd` перейдём в каталог в котором будем работать (Рис. 2.4):

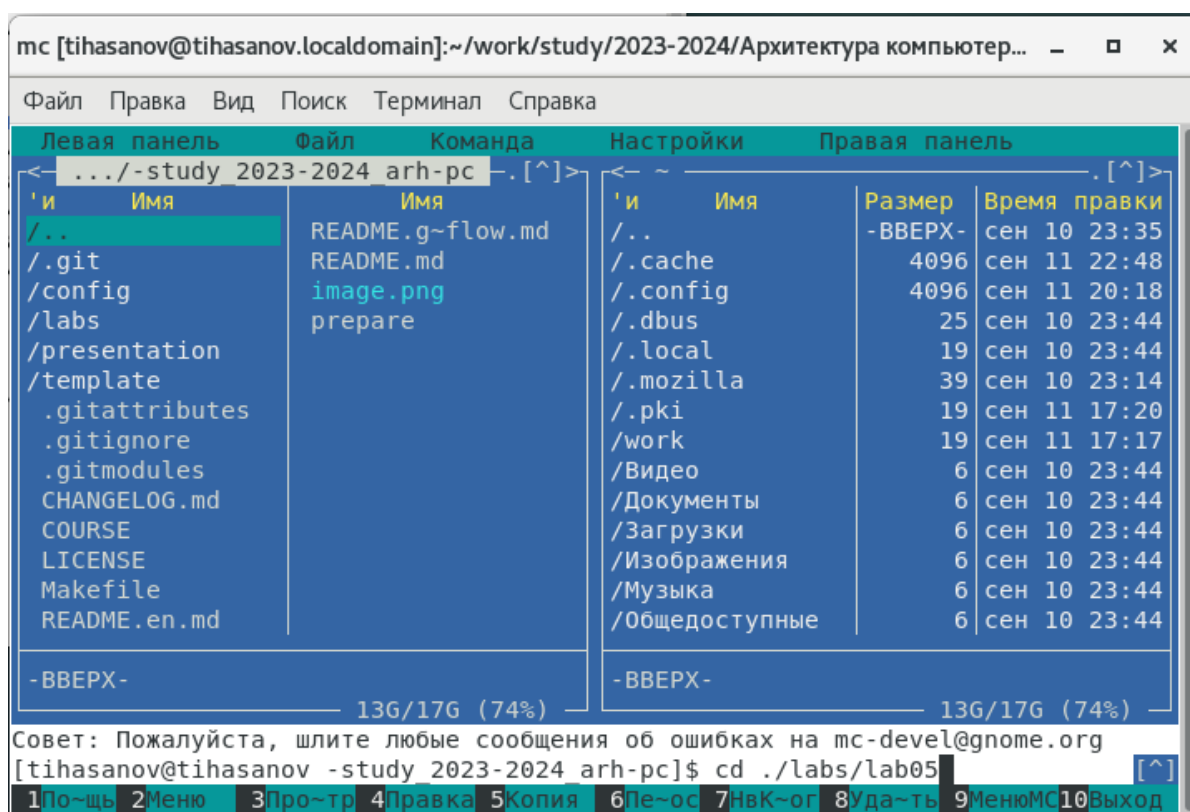


Рис. 2.4: Создание папки

Теперь с помощью команды touch создадим файл lab5-1.asm (Рис. 2.5):



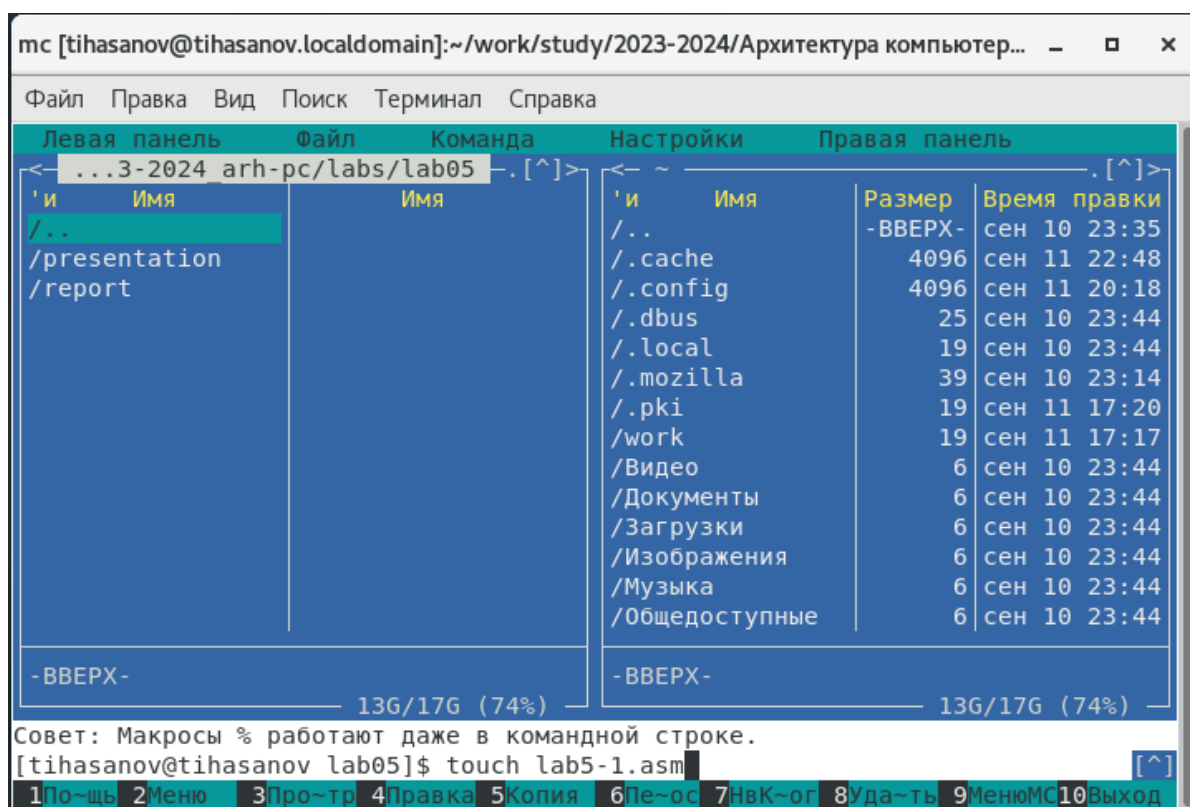


Рис. 2.5: Создание файла lab5-1.asm с помощью команды touch прямо в mc

Теперь с помощью клавиши F4 откроем только что созданный файл. Файл по умолчанию будет открыт в редакторе mcedit (Рис. 2.6):

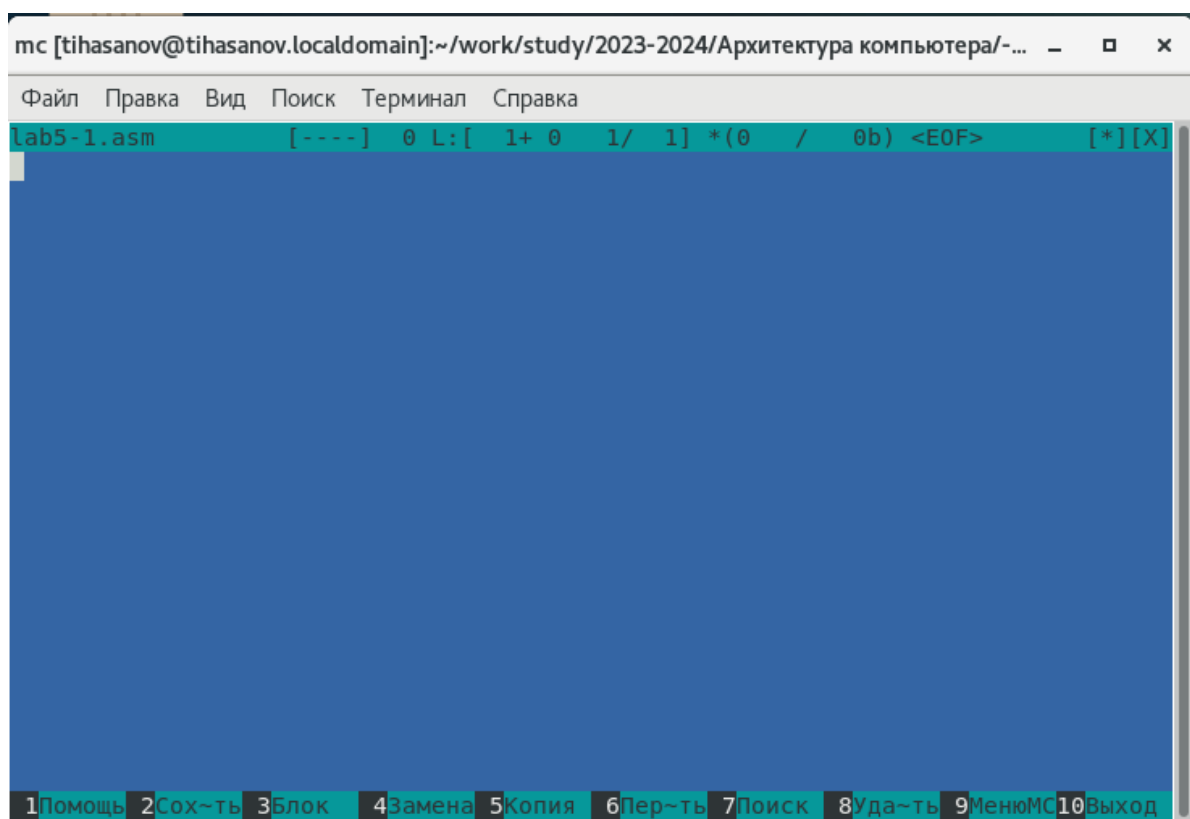
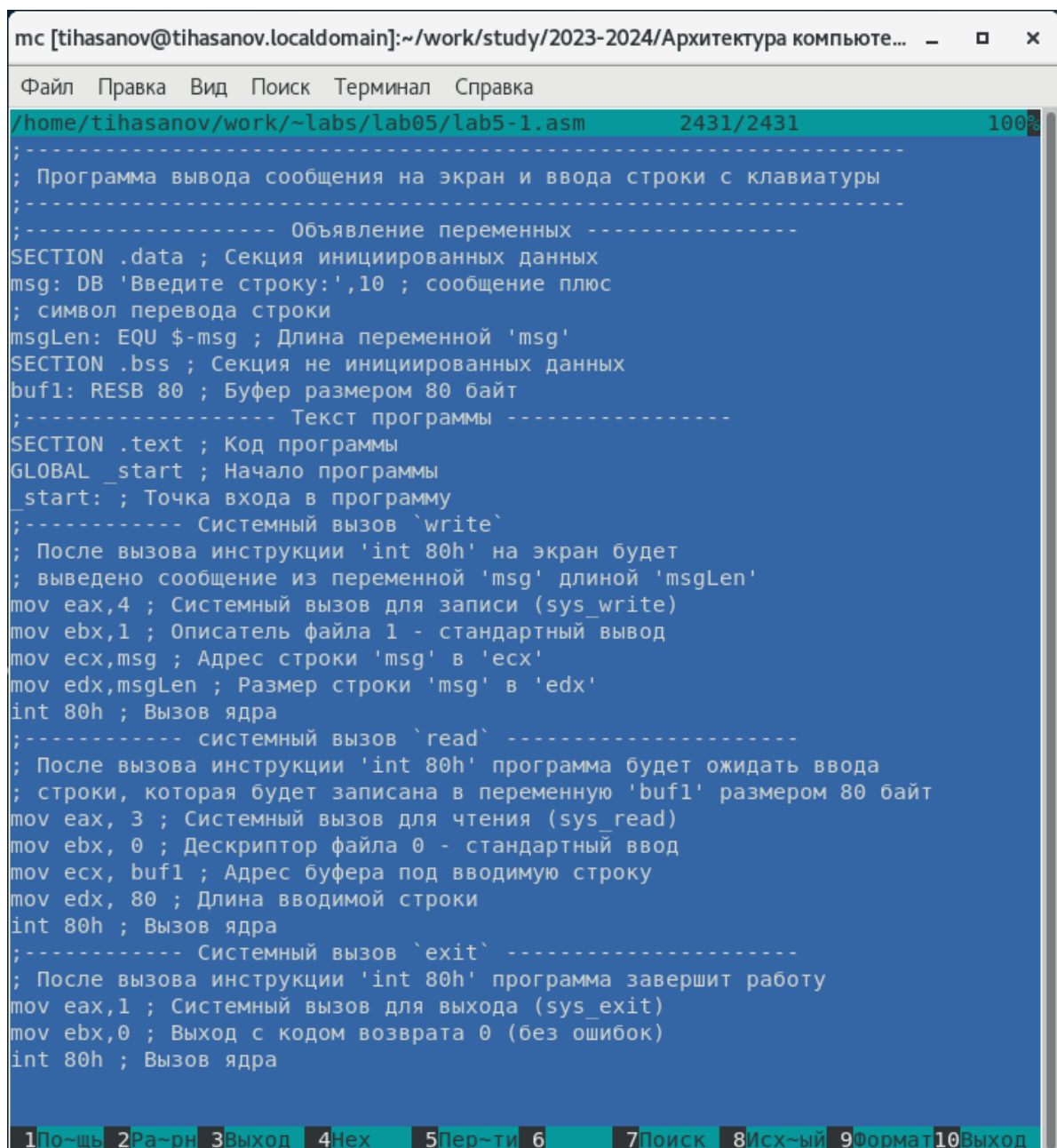


Рис. 2.6: Выбор текстового редактора

Теперь отредактируем файл и поместим в него следующий код (Рис. 2.7):

Рис. 2.7: Редактирование файла lab5-1.asm

Теперь сохраним его (клавишей F2, согласившись с сохранением, и F10 чтобы выйти из редактора) и с помощью F3 откроем для просмотра, чтобы убедиться, что он сохранился корректно (Рис. 2.8):



mc [tihasanov@tihasanov.localdomain]:~/work/study/2023-2024/Архитектура компьюте... — □ ×

Файл Правка Вид Поиск Терминал Справка

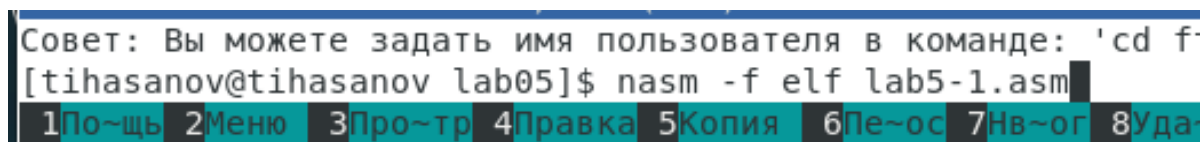
/home/tihasanov/work/~labs/lab05/lab5-1.asm 2431/2431 100%

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

1По-щ 2Ра-рн 3Выход 4Нех 5Пер-ти 6 7Поиск 8Исх-ый 9Формат10Выход

Рис. 2.8: Проверка успешного редактирования

Теперь скомпилируем его (Рис. 2.9):



```
Совет: Вы можете задать имя пользователя в команде: 'cd f-  
[tihasanov@tihasanov lab05]$ nasm -f elf lab5-1.asm
```

1По-щ 2Меню 3Про-тр 4Правка 5Копия 6Пе-ос 7Нв-ог 8Уда-

Рис. 2.9: Компиляция файла с помощью nasm

И соберём (Рис. 2.10):

```
Совет: Поиск файла: вы можете работать с найденными файлами при Па
[tihasanov@tihasanov lab05]$ ld -m elf_i386 -o lab5-1 lab5-1.o
1По~щъ 2Меню 3Про~тр 4Правка 5Копия 6Пе~ос 7Нв~ог 8Уда~ть 9Меню
```

Рис. 2.10: Сборка исполняемого файла с помощью ld

После этого запустим получившийся исполняемый файл (Рис. 2.11):

```
Совет: Автодополнение работает во всех ст
[tihasanov@tihasanov lab05]$ ./lab5-1
1По~щъ 2Меню 3Про~тр 4Правка 5Копия 6П
```

Рис. 2.11: Запуск исполняемого файла

Теперь введём ФИ (Рис. 2.12):

```
[tihasanov@tihasanov lab05]$ ./lab5-1
Введите строку:
Хасанов Тимур
```

Рис. 2.12: Взаимодействие с программой

После нажатия Enter программа завершится и ничего не произойдёт. Теперь скачаем файл in\_out.asm и откроем папку с ним в правой панели (Рис. 2.13):

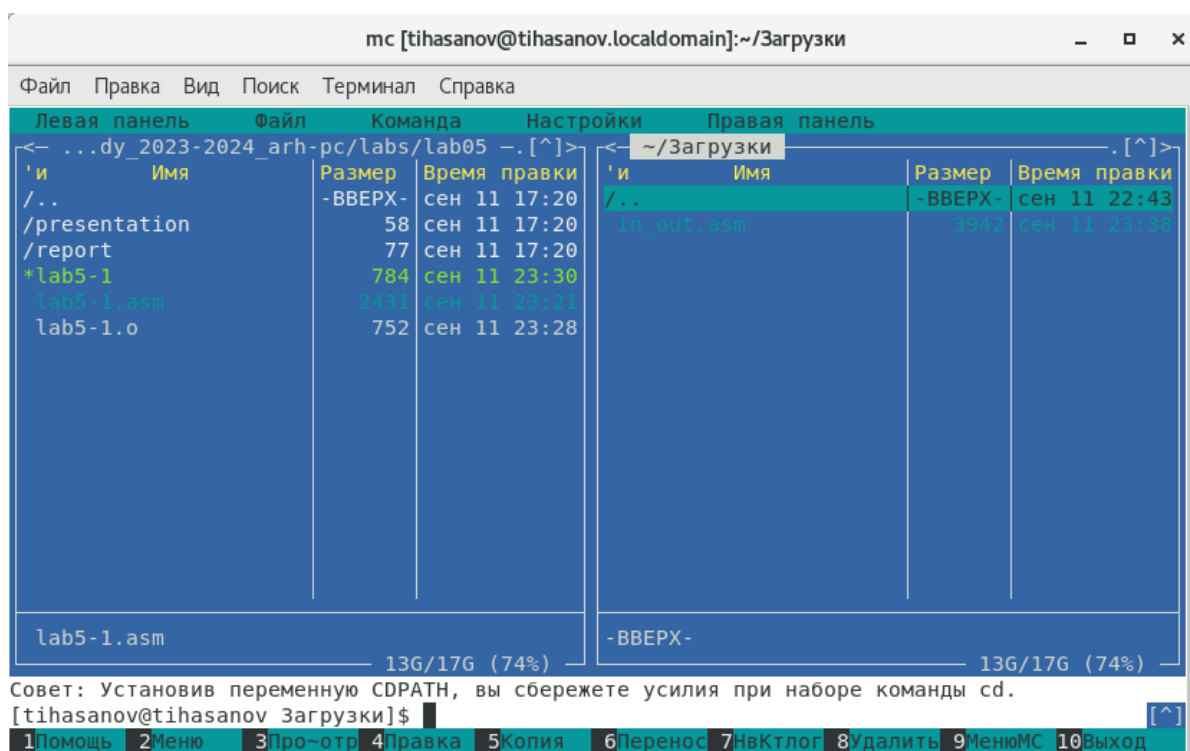


Рис. 2.13: Открытие папки с файлом in\_out.asm в правой панели

Переместим его в нашу рабочую папку с помощью F6 (Рис. 2.14):

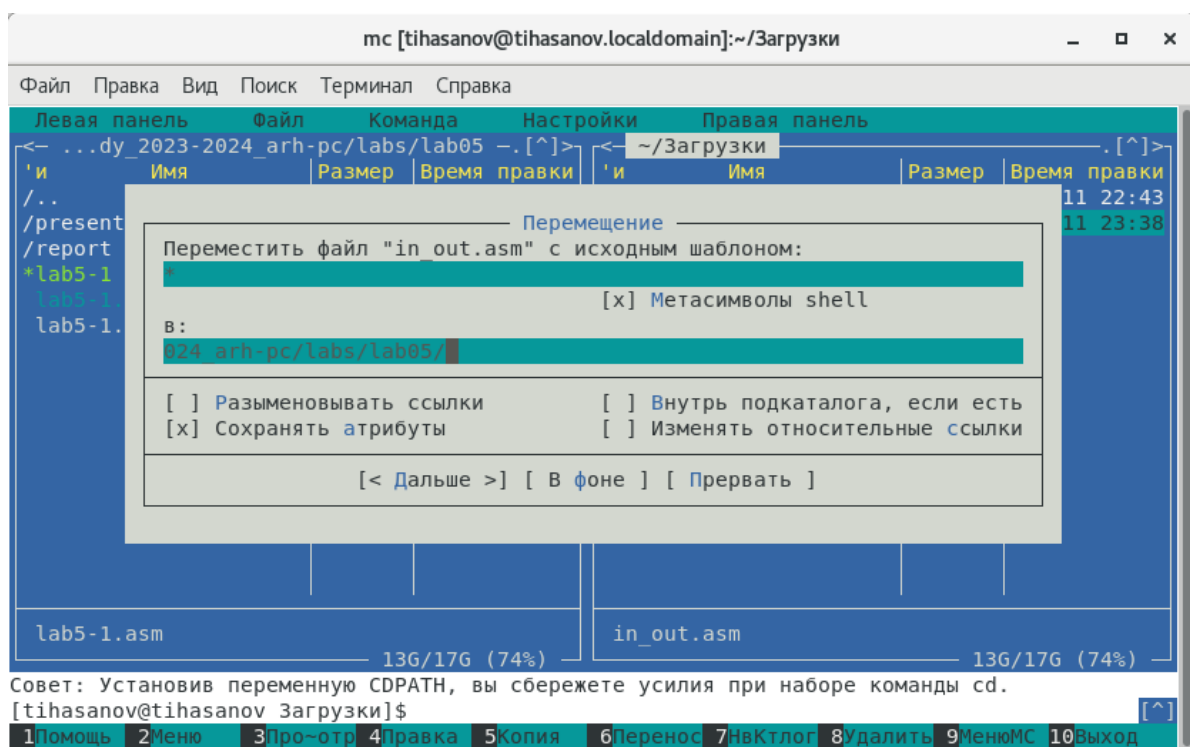


Рис. 2.14: Перемещение файла с помощью F6

Теперь сделаем копию файла lab5-1.asm с помощью команды F5. Назовём копию lab5-2.asm (Рис. 2.15):

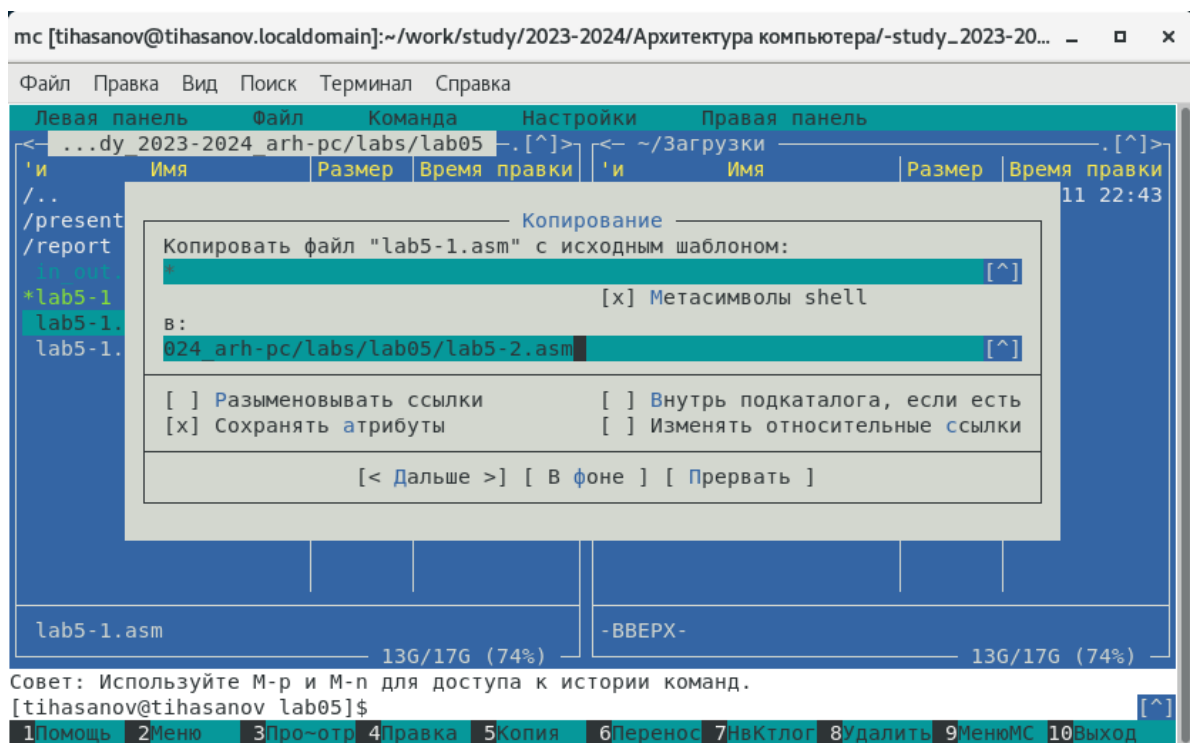


Рис. 2.15: Копирование файла с помощью F5

Теперь наша папка выглядит следующим образом (Рис. 2.16):

in_out.asm	3942	сен 11 23:38
*lab5-1	784	сен 11 23:30
lab5-1.asm	2431	сен 11 23:21
lab5-1.o	752	сен 11 23:28
lab5-2.asm	2431	сен 11 23:21

Рис. 2.16: Текущий вид рабочей папки

Откроем в текстовом редакторе файл lab5-2.asm и напишем туда следующий код (Рис. 2.17):



```
mc [tihasanov@tihasanov.localdomain]:~/work/study/2023-2024/Архитектура компьютера/-study_2
Файл Правка Вид Поиск Терминал Справка
lab5-2.asm [-M- -] 41 L:[ 1+16 17/ 17] *(1224/1224b) <EOF>
;
; Программа вывода сообщения на экран и ввода строки с клавиатуры
;
-----
%include 'in_out.asm' ; подключение внешнего файла
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 sprintf ; 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. 2.17: Редактирование файла lab5-2.asm

После чего создадим исполняемый файл с помощью nasm и ld (Рис. 2.18):

```
[tihasanov@tihasanov lab05]$ nasm -f elf lab5-2.asm
[tihasanov@tihasanov lab05]$ ld -m elf_i386 -o lab5-2 lab5-2.o
[tihasanov@tihasanov lab05]$ ./lab5-2
```

Рис. 2.18: Создание исполняемого файла

Запустим созданный файл (Рис. 2.19):

```
[tihasanov@tihasanov lab05]$ ./lab5-2
Введите строку:
Хасанов Тимур
```

Рис. 2.19: Запуск исполняемого файла

Он работает также, как и файл lab5-1, но использует для работы сторонний

файл. Попробуем теперь вместо команды `sprintf` использовать просто команду `sprint` (Рис. 2.20):

```
;-----  
; 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. 2.20: Изменение файла `lab5-2.asm`

Точно также соберём исполняемый файл и запустим его (Рис. 2.21):

```
[tihasanov@tihasanov lab05]$ ./lab5-2  
Введите строку: Хасанов Тимур
```

Рис. 2.21: Запуск изменённого файла

Как мы видим, 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 F5 копию файла lab5-1.asm (Рис. 3.1):

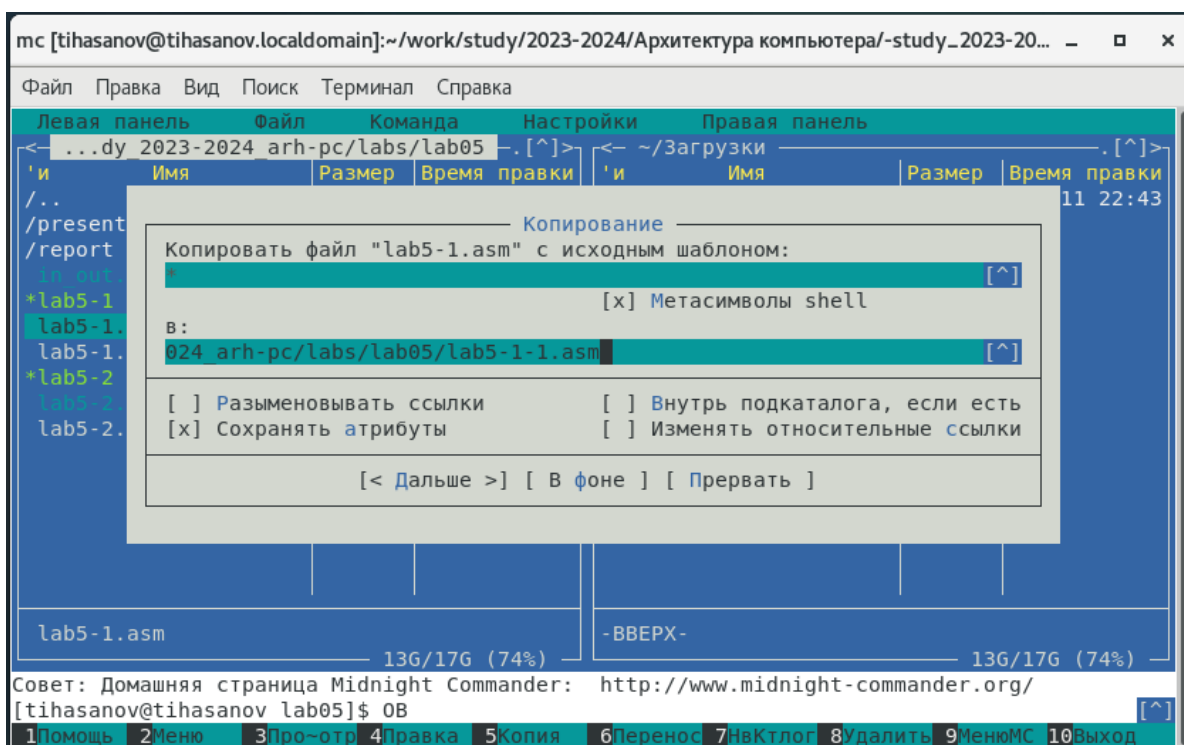


Рис. 3.1: Создание копии файла lab5-1.asm

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 `esx` и размер

строки buf1 (80) в edx (Рис. 3.2):

```
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 ;Descriptor файла 0 - стандартный ввод
mov ecx, buf1 ; Адрес буфера под вводимую строку
mov edx, 80 ; Длина вводимой строки
int 80h ; Вызов ядра
;----- Системный вызов `write` -----
; После вызова инструкции 'int 80h' на экран будет
; выведено сообщение из переменной 'buf1' длиной 80
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,80 ; Размер строки 'buf1' в 'edx'
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
```

Рис. 3.2: Изменение файла lab5-1-1.asm

Сохраним изменения, создадим исполняемый файл, запустим программу и проверим, что всё работает (Рис. 3.3):

```
[tihasanov@tihasanov lab05]$ nasm -f elf lab5-1-1.asm
[tihasanov@tihasanov lab05]$ ld -m elf_i386 -o lab5-1-1 lab5-1-1.o
[tihasanov@tihasanov lab05]$ ./lab5-1-1
Введите строку:
Хасанов Тимур
Хасанов Тимур
```

Рис. 3.3: Проверка работы программы

Теперь создадим с помощью F5 копию файла lab5-2.asm (Рис. 3.4):

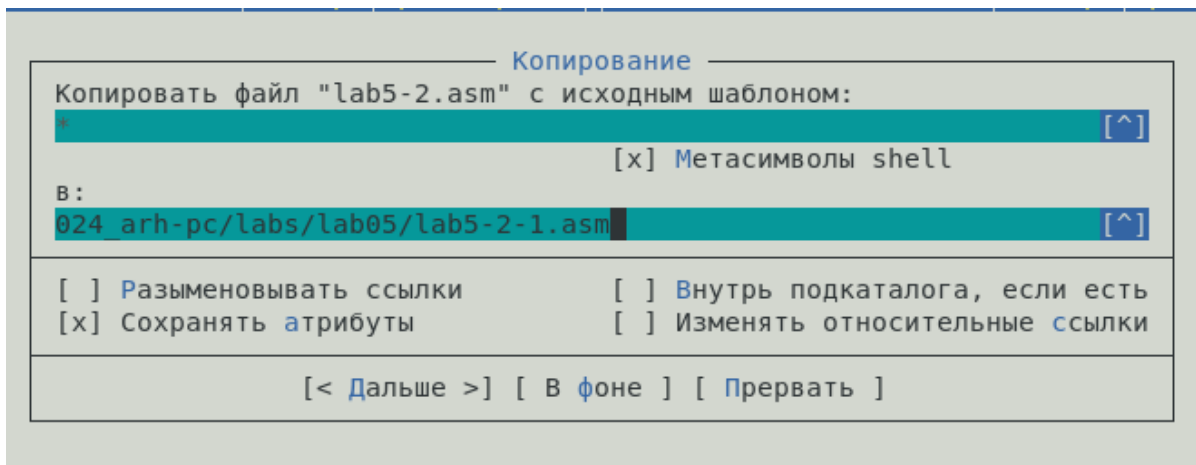


Рис. 3.4: Создание копии файла lab5-2.asm

Теперь сделаем так, чтобы этот код также выводил тот текст, что получит на ввод. Для этого перед последней строкой добавим строчку, которая записывает в еах адрес buf1, а также строчку, которая вызывает подпрограмму sprintLF (Рис. 3.5):

```
lab5-2-1.asm      [-M--] 41 L:[ 1+18 19/ 19] *(1399/1399b) <EOF>
;-----
; Программа вывода сообщения на экран и ввода строки с клавиатуры
;-----
%include 'in_out.asm' ; подключение внешнего файла
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 sprintLF ; 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 ; запись адреса выводимого сообщения в `EAX`
call sprintLF ; вызов подпрограммы печати сообщения
call quit ; вызов подпрограммы завершения
```

Рис. 3.5: Изменение файла lab5-2-1.asm

Теперь создадим исполняемый файл, запустим программу и убедимся, что

она работает (Рис. 3.6):

```
[tihasanov@tihasanov lab05]$ nasm -f elf lab5-2-1.asm
[tihasanov@tihasanov lab05]$ ld -m elf_i386 -o lab5-2-1 lab5-2-1.o
[tihasanov@tihasanov lab05]$ ./lab5-2-1
Введите строку:
Хасанов Тимур
Хасанов Тимур
```

Рис. 3.6: Создание исполняемого файла

## 4 Выводы

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