

# **отчёта по лабораторной работе №5**

**Дисциплина: архитектура компьютера**

Аджабханян Овик

# Содержание

<b>1</b>	<b>Цель работы</b>	<b>5</b>
<b>2</b>	<b>Задание</b>	<b>6</b>
<b>3</b>	<b>Теоретическое введение</b>	<b>7</b>
<b>4</b>	<b>Выполнение лабораторной работы</b>	<b>9</b>
4.1	Основы работы с mc . . . . .	9
4.2	Структура программы на языке ассемблера NASM . . . . .	12
4.3	Подключение внешнего файла . . . . .	14
4.4	Выполнение заданий для самостоятельной работы . . . . .	17
<b>5</b>	<b>Выводы</b>	<b>20</b>
	<b>Список литературы</b>	<b>21</b>

# Список иллюстраций

4.1	Открытый тс . . . . .	9
4.2	Перемещение между директориями . . . . .	10
4.3	Создание каталога . . . . .	10
4.4	Перемещение между директориями . . . . .	11
4.5	Создание файла . . . . .	11
4.6	Открытие файла для редактирования . . . . .	12
4.7	Открытие файла для просмотра . . . . .	13
4.8	Компиляция файла . . . . .	13
4.9	Передача на обработку компоновщику . . . . .	13
4.10	Проверка файлов . . . . .	13
4.11	Исполнение файла . . . . .	14
4.12	Копирование файла . . . . .	14
4.13	Копирование файла . . . . .	14
4.14	Редактирование файла . . . . .	15
4.15	Исполнение файла . . . . .	15
4.16	Отредактированный файл . . . . .	16
4.17	Запуск файла . . . . .	16
4.18	Запуск файла . . . . .	16
4.19	Запуск файла . . . . .	17
4.20	Копирование файла . . . . .	17
4.21	Редактирование файла . . . . .	17
4.22	Исполнение файла . . . . .	18
4.23	Копирование файла . . . . .	18
4.24	Редактирование файла . . . . .	19
4.25	Исполнение файла . . . . .	19

## **Список таблиц**

# 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 инициированных (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 инициированных 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 (четырёх- рённое 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).

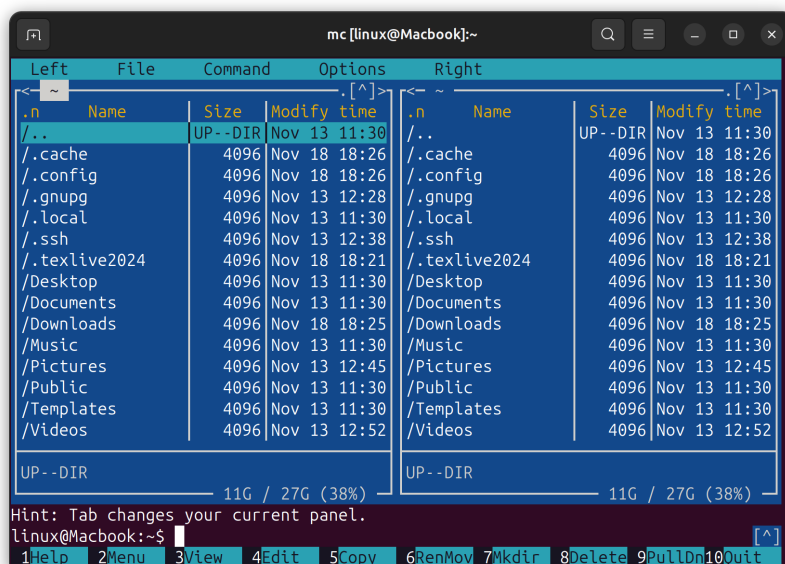


Рис. 4.1: Открытый mc

Перехожу в каталог ~/work/arch-рс, используя файловый менеджер mc (рис. 4.2)

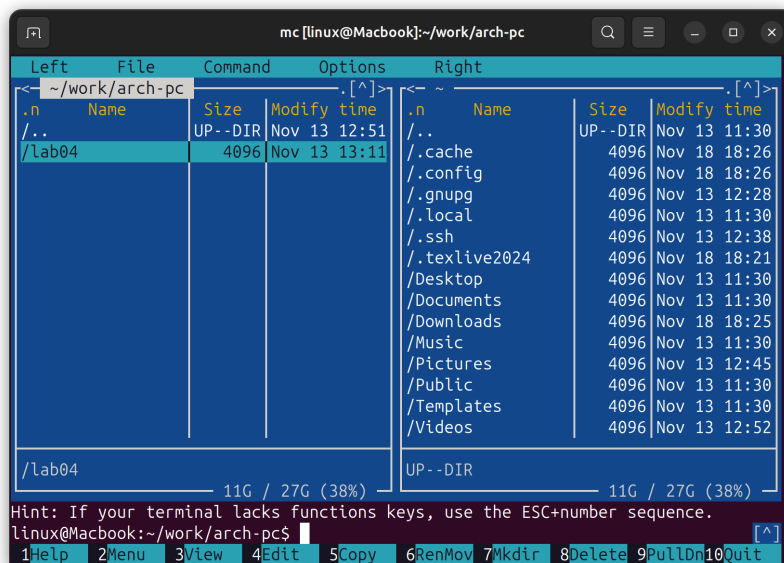


Рис. 4.2: Перемещение между директориями

С помощью функциональной клавиши F7 создаю каталог lab05 (рис. 4.3).

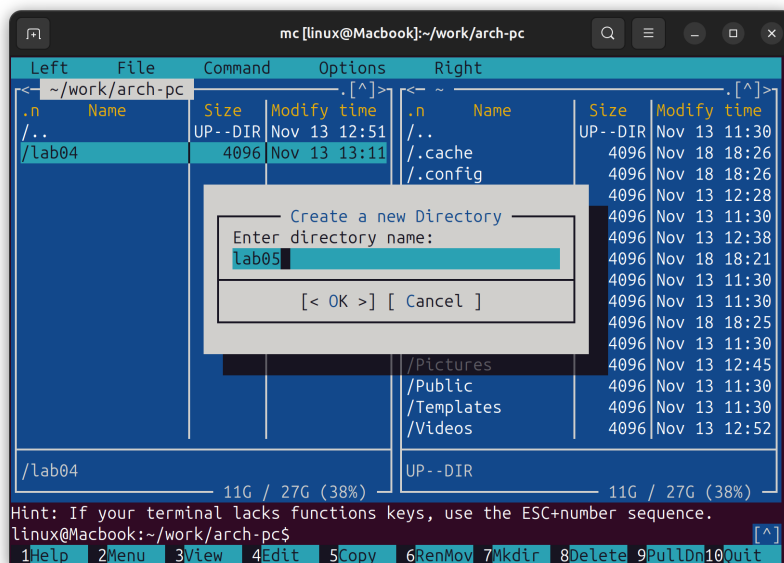


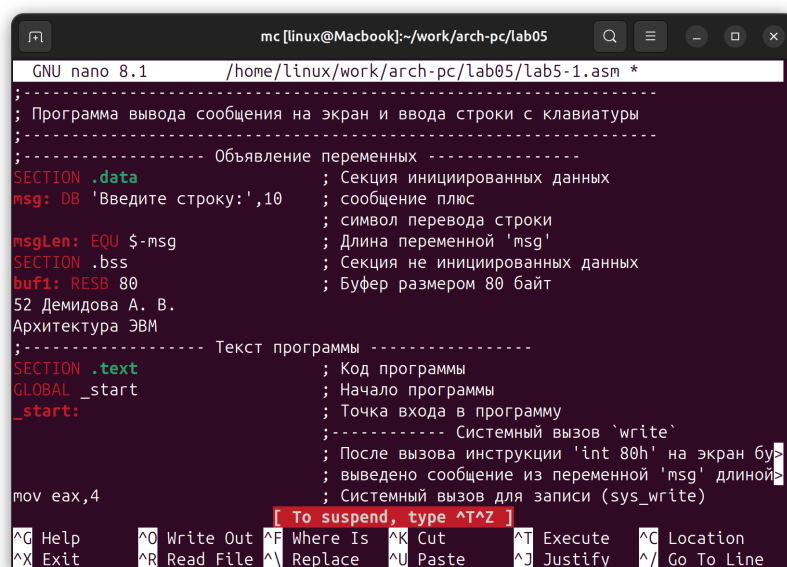
Рис. 4.3: Создание каталога

Переходу в созданный каталог (рис. 4.4).



## 4.2 Структура программы на языке ассемблера NASM

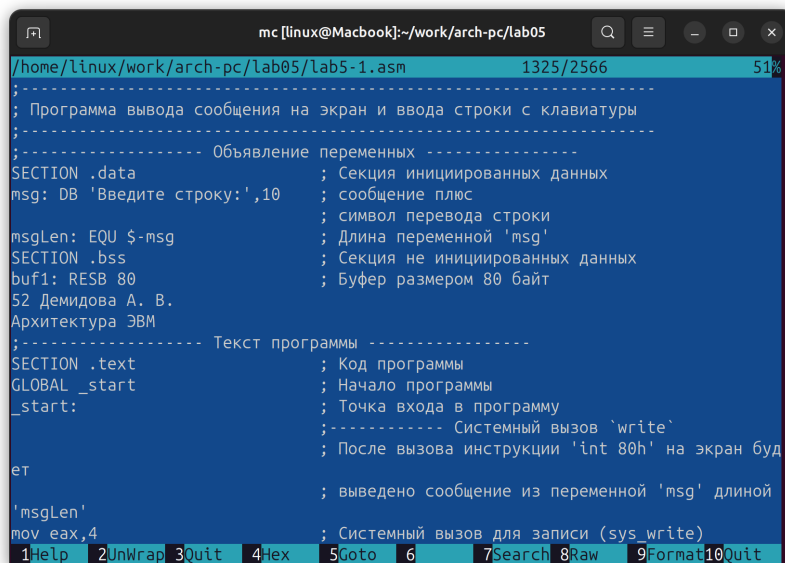
С помощью функциональной клавиши F4 открываю созданный файл для редактирования в редакторе nano, Ввожу в файл код программы для запроса строки у пользователя (рис. 4.6). Далее выхожу из файла сохраняя изменения



```
GNU nano 8.1 /home/linux/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
52 Демидова А. В.
Архитектура ЭВМ
;----- 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 бу>
; выведено сообщение из переменной 'msg' длиной>
; Системный вызов для записи (sys_write)
mov eax,4
[ To suspend, type ^T^Z ]
^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
```

Рис. 4.6: Открытие файла для редактирования

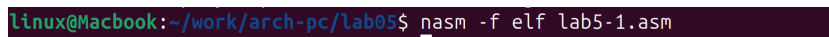
С помощью функциональной клавиши F3 открываю файл для просмотра, чтобы проверить, содержит ли файл текст программы (рис. 4.7).



```
mc [linux@Macbook]:~/work/arch-pc/lab05
/home/linux/work/arch-pc/lab05/lab5-1.asm 1325/2566 51%
; 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
52 Демидова А. В.
Архитектура ЭВМ
;----- 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 буд
ет ; выведено сообщение из переменной 'msg' длиной
'msgLen'
mov eax,4 ; Системный вызов для записи (sys_write)
1Help 2UnWrap 3Quit 4Hex 5Goto 6 7Search 8Raw 9Format10Quit
```

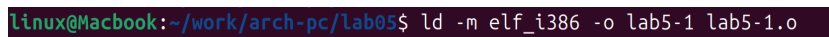
Рис. 4.7: Открытие файла для просмотра

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. 4.8). Создался исполняемый файл `lab5-1`.



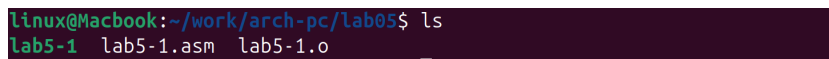
```
linux@Macbook:~/work/arch-pc/lab05$ nasm -f elf lab5-1.asm
```

Рис. 4.8: Компиляция файла



```
linux@Macbook:~/work/arch-pc/lab05$ ld -m elf_i386 -o lab5-1 lab5-1.o
```

Рис. 4.9: Передача на обработку компоновщику



```
linux@Macbook:~/work/arch-pc/lab05$ ls
lab5-1 lab5-1.asm lab5-1.o
```

Рис. 4.10: Проверка файлов

Запускаю исполняемый файл. Программа выводит строку “Введите строку:” и ждет ввода с клавиатуры, я ввожу свои ФИО, на этом программа заканчивает свою работу (рис. 4.11).

```
linux@Macbook:~/work/arch-pc/lab05$ ./lab5-1
Введите строку:
Adjabkhanian Novik
```

Рис. 4.11: Исполнение файла

## 4.3 Подключение внешнего файла

С помощью функциональной клавиши F5 копирую файл `in_out.asm` из каталога Загрузки в созданный каталог `lab05` (рис. 4.12).

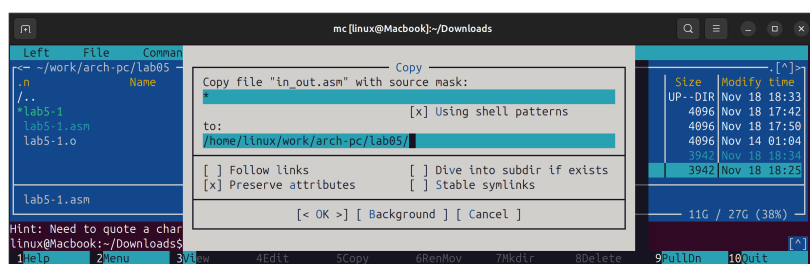


Рис. 4.12: Копирование файла

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. 4.13).

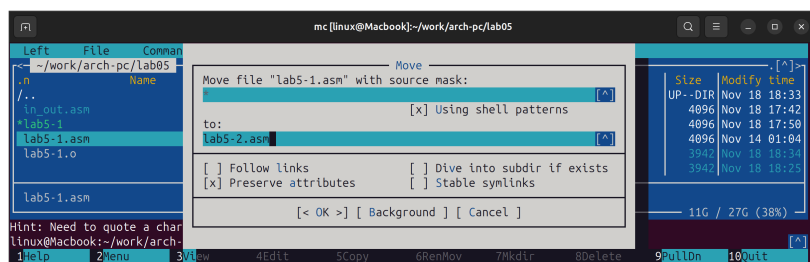


Рис. 4.13: Копирование файла

Изменяю содержимое файла `lab5-2.asm` во встроенном редакторе `nano` (рис. 4.14), чтобы в программе использовались подпрограммы из внешнего файла `in_out.asm`.

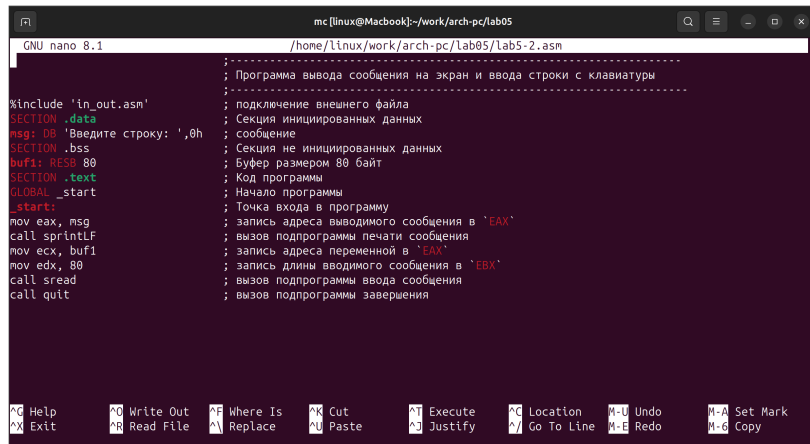


Рис. 4.14: Редактирование файла

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. 4.15).

```
linux@Macbook:~/work/arch-pc/lab05$ nasm -f elf lab5-2.asm
linux@Macbook:~/work/arch-pc/lab05$ ld -m elf_i386 -o lab5-2 lab5-2.o
linux@Macbook:~/work/arch-pc/lab05$ ./lab5-2
Введите строку:
Adjabkhanian Hovik
```

Рис. 4.15: Исполнение файла

Открываю файл `lab5-2.asm` для редактирования в nano функциональной клавишей F4. Изменяю в нем подпрограмму `sprintf` на `sprint`. Сохраняю изменения и выполняю компоновку созданного объектного файла, (рис. 4.16).

```

GNU nano 8.1 /home/linux/work/arch-pc/lab05/lab5-2.asm
;----->
; Программа вывода сообщения на экран и ввода с>
;----->
; подключение внешнего файла
; Секция инициализированных данных
%include 'in_out.asm'
; сообщение
SECTION .data
msg: DB 'Введите строку: ',0h
; Секция не инициализированных данных
SECTION .bss
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. 4.16: Отредактированный файл

Запускаю новый исполняемый файл (рис. 4.17).

```

linux@Macbook:~/work/arch-pc/lab05$ nasm -f elf lab5-2.asm
linux@Macbook:~/work/arch-pc/lab05$ ld -m elf_i386 -o lab5-2 lab5-2.o
linux@Macbook:~/work/arch-pc/lab05$ ./lab5-2
Введите строку:
Novik Adjabkhanian

```

Рис. 4.17: Запуск 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`. (рис. 4.18).

```

linux@Macbook:~/work/arch-pc/lab05$ nasm -f elf lab5-2.asm
linux@Macbook:~/work/arch-pc/lab05$ ld -m elf_i386 -o lab5-2 lab5-2.o
linux@Macbook:~/work/arch-pc/lab05$ ./lab5-2
Введите строку:
Novik Adjabkhanian

```

Рис. 4.18: Запуск файла



```
linux@Macbook:~/work/arch-pc/lab05$ ./lab5-2-2
Введите строку: Novik Adjabkhanian
```

Рис. 4.19: Запуск файла

## 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. 4.20).

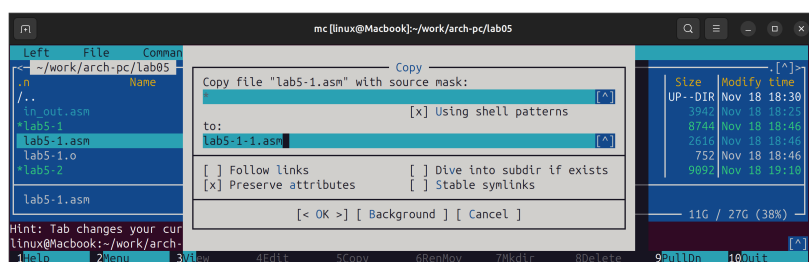


Рис. 4.20: Копирование файла

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. 4.21).

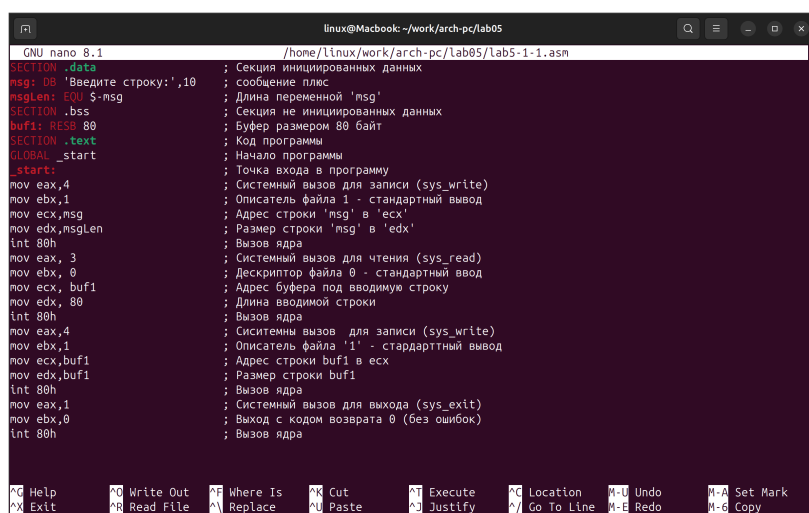


Рис. 4.21: Редактирование файла

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. 4.22).

```
linux@Macbook:~/work/arch-pc/lab05$ nasm -f elf lab5-2.asm
linux@Macbook:~/work/arch-pc/lab05$ ld -m elf_i386 -o lab5-2 lab5-2.o
linux@Macbook:~/work/arch-pc/lab05$ ./lab5-2
Введите строку: Adjabkhanian Novik
```

Рис. 4.22: Исполнение файла

3. Создаю копию файла lab5-2.asm с именем lab5-2-1.asm с помощью функциональной клавиши F5 (рис. 4.23).

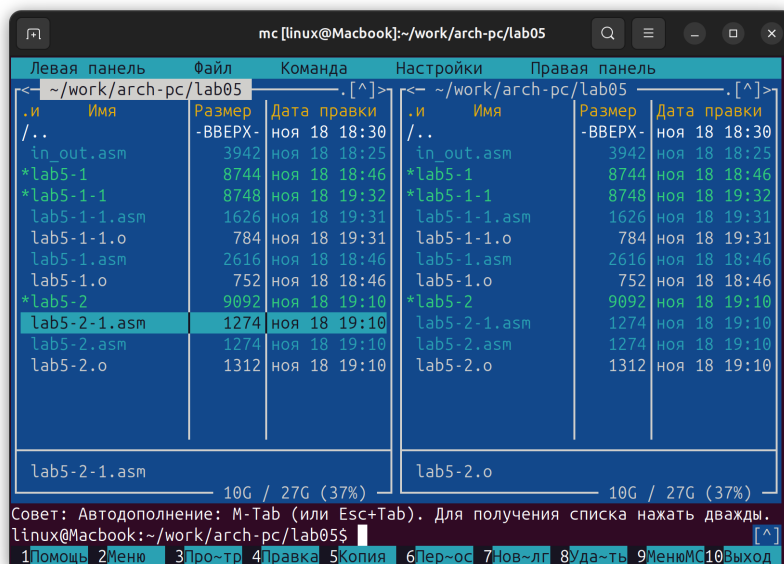
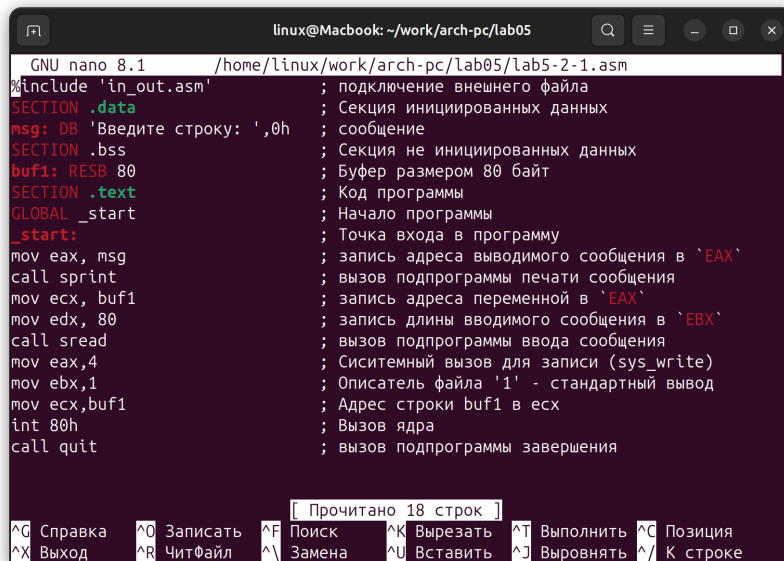


Рис. 4.23: Копирование файла

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. 4.24).

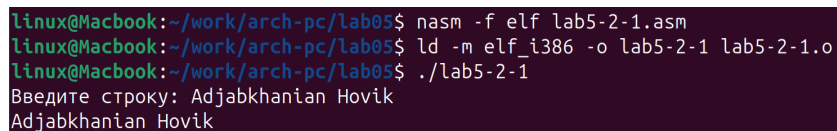


```
GNU nano 8.1 /home/linux/work/arch-pc/lab05/lab5-2-1.asm
%include 'in_out.asm'      ; подключение внешнего файла
SECTION .data              ; Секция инициализированных данных
msg: DB 'Введите строку: ',0h ; сообщение
SECTION .bss               ; Секция не инициализированных данных
buf1: RESB 80              ; 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
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

[ Прочитано 18 строк ]
^G Справка  ^O Записать  ^F Поиск    ^K Вырезать ^T Выполнить ^C Позиция
^X Выход    ^R ЧитФайл  ^\ Замена  ^U Вставить ^J Выровнять ^_ К строке
```

Рис. 4.24: 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. 4.25).



```
linux@Macbook:~/work/arch-pc/lab05$ nasm -f elf lab5-2-1.asm
linux@Macbook:~/work/arch-pc/lab05$ ld -m elf_i386 -o lab5-2-1 lab5-2-1.o
linux@Macbook:~/work/arch-pc/lab05$ ./lab5-2-1
Введите строку: Adjabkhanian Novik
Adjabkhanian Novik
```

Рис. 4.25: 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`.

# **Список литературы**

1. Лабораторная работа №6