

Отчет по лабораторной работе №5

Дисциплина: архитектура компьютера

Бызова Мария Олеговна

Содержание

1	Цель работы	5
2	Теоретическое введение	6
3	Выполнение лабораторной работы	8
3.1	Подключение внешнего файла	13
3.2	Выполнение заданий для самостоятельной работы	17
4	Выводы	22

Список иллюстраций

3.1	Введение команды	8
3.2	Открытый mc	8
3.3	Перемещение между директориями	9
3.4	Создание каталога	9
3.5	Перемещение между директориями	10
3.6	Создание файла	10
3.7	Открытие файла для редактирования	11
3.8	Редактирование файла	11
3.9	Открытие файла для просмотра	12
3.10	Компиляция файла и передача на обработку компоновщику . . .	12
3.11	Исполнение файла	12
3.12	Скачанный файл	13
3.13	Копирование файла	13
3.14	Копирование файла	14
3.15	Редактирование файла	15
3.16	Компиляция файла и передача на обработку компоновщику . . .	15
3.17	Исполнение файла	15
3.18	Отредактированный файл	16
3.19	Компиляция файла и передача на обработку компоновщику . . .	16
3.20	Исполнение файла	17
3.21	Копирование файла	17
3.22	Редактирование файла	18
3.23	Компиляция файла и передача на обработку компоновщику . . .	18
3.24	Исполнение файла	18
3.25	Копирование файла	20
3.26	Редактирование файла	20
3.27	Компиляция файла и передача на обработку компоновщику . . .	21
3.28	Исполнение файла	21

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 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).

3 Выполнение лабораторной работы

1. Открываем Midnight Commander, введя в терминал `mc` (рис. [3.1], [3.2]).

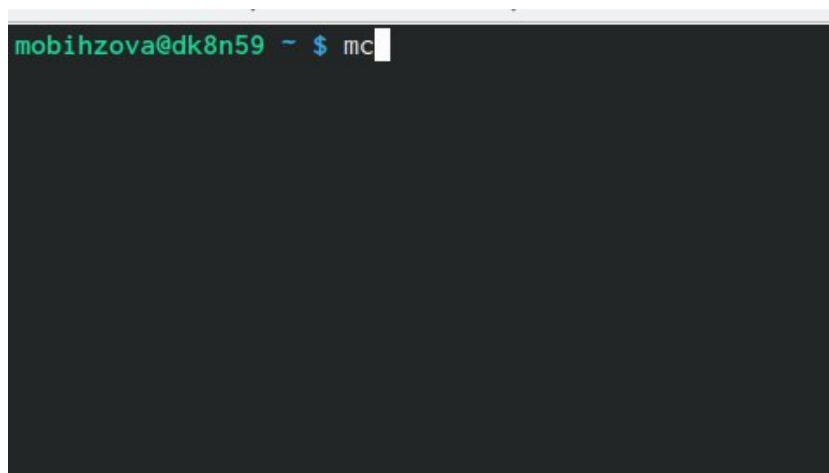


Рис. 3.1: Введение команды

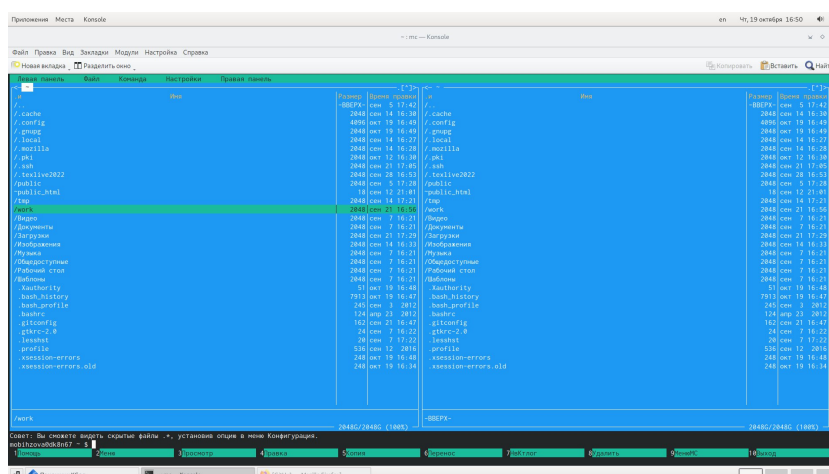


Рис. 3.2: Открытый mc

2. Пользуясь клавишами клавиатуры, переходим в каталог ~/work/arch-рс, созданный при выполнении предыдущей лабораторной работы рис. [3.3])

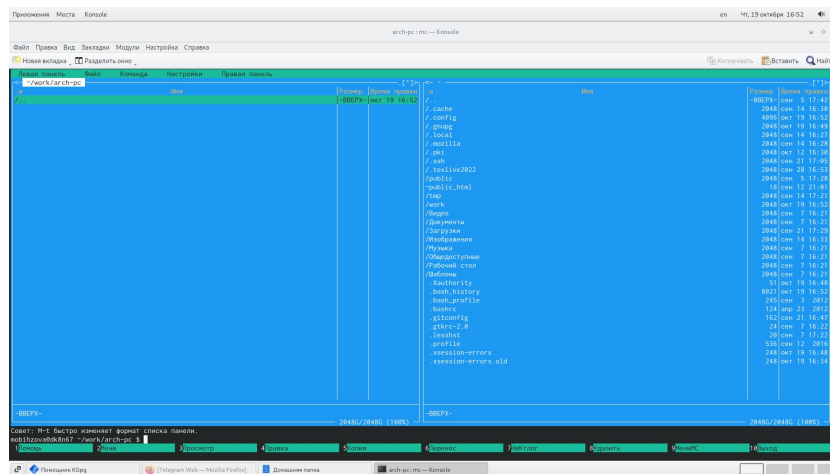


Рис. 3.3: Перемещение между директориями

3. С помощью функциональной клавиши F7 создаем каталог lab05 (рис. [3.4]).

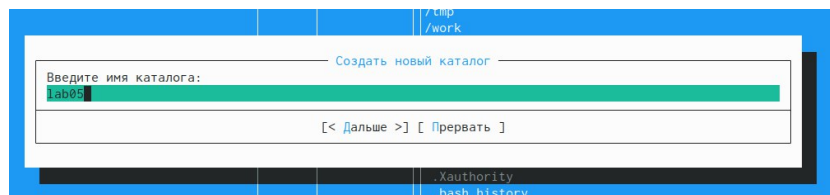


Рис. 3.4: Создание каталога

Переходим в созданный каталог (рис. [3.5]).

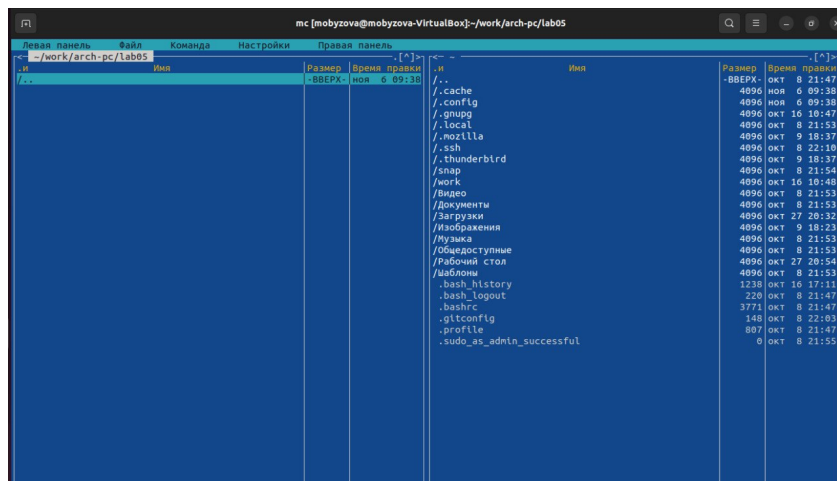


Рис. 3.5: Перемещение между директориями

4. В строке ввода прописываем команду `touch lab5-1.asm`, чтобы создать файл, в котором будем работать (рис. [3.6]).

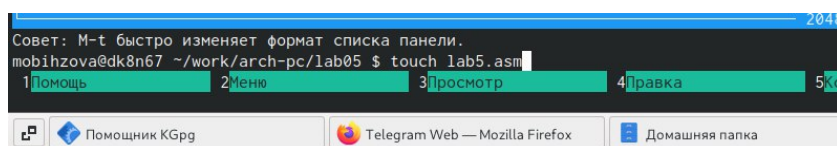


Рис. 3.6: Создание файла

5. С помощью функциональной клавиши F4 открываем созданный файл для редактирования во встроенном редакторе (рис. [3.7]). Как правило в качестве встроенного редактора Midnight Commander используется редакторы nano или mcedit.

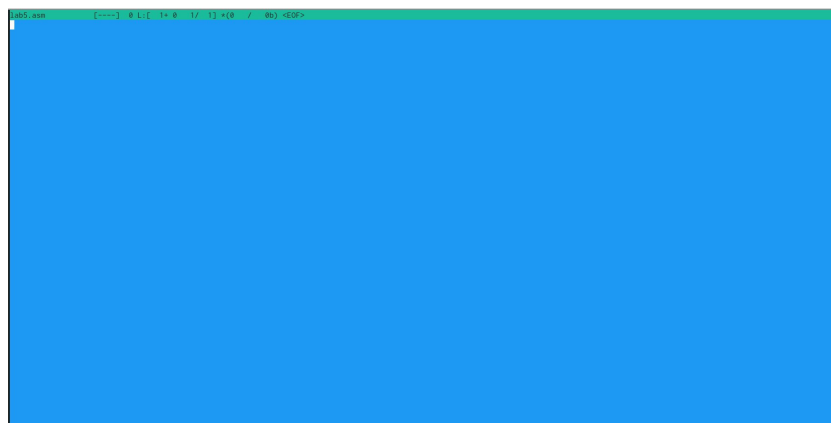


Рис. 3.7: Открытие файла для редактирования

6. Вводим в файл код программы для запроса строки у пользователя (рис. [3.8]). Далее выходим из файла (Ctrl+X), сохраняя изменения (Y, Enter).

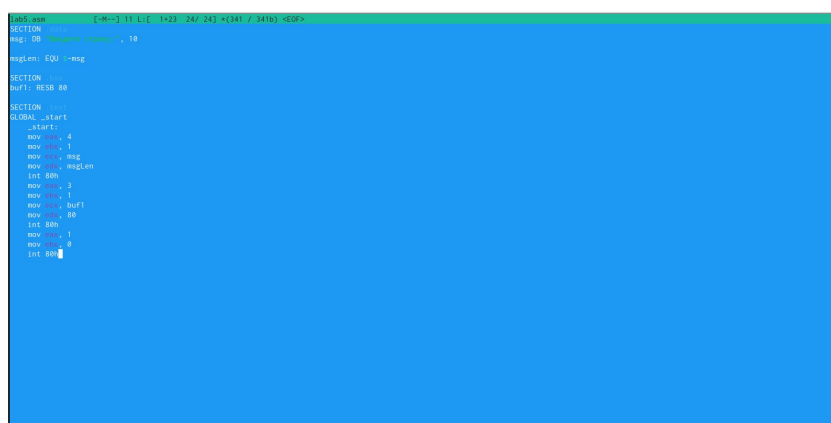


Рис. 3.8: Редактирование файла

7. С помощью функциональной клавиши F3 открываем файл для просмотра, чтобы проверить, содержит ли файл текст программы (рис. [3.9]).

```
mobihzova@dk8n67: ~/work/arch-pc/lab05 $ cat lab5-1.asm
SECTION .data
msg: DB "Введите строку:", 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 0x80

    mov eax, 3
    mov ebx, 1
    mov ecx, buf1
    mov edx, 80
    int 0x80

    mov eax, 1
    mov ebx, 0
    int 0x80
```

Рис. 3.9: Открытие файла для просмотра

8. Транслируем текст программы файла в объектный файл командой `nasm -f elf lab5-1.asm`. Создается объектный файл `lab5-1.o`. Выполняем компоновку объектного файла с помощью команды `ld -m elf_i386 -o lab5-1 lab5-1.o` (рис. [3.10]). Создается исполняемый файл `lab5-1`.

```
mobihzova@dk8n67 ~/work/arch-pc/lab05 $ nasm -f elf lab5-1.asm
mobihzova@dk8n67 ~/work/arch-pc/lab05 $ ld -m elf_i386 -o lab5-1 lab5-1.o
```

Рис. 3.10: Компиляция файла и передача на обработку компоновщику

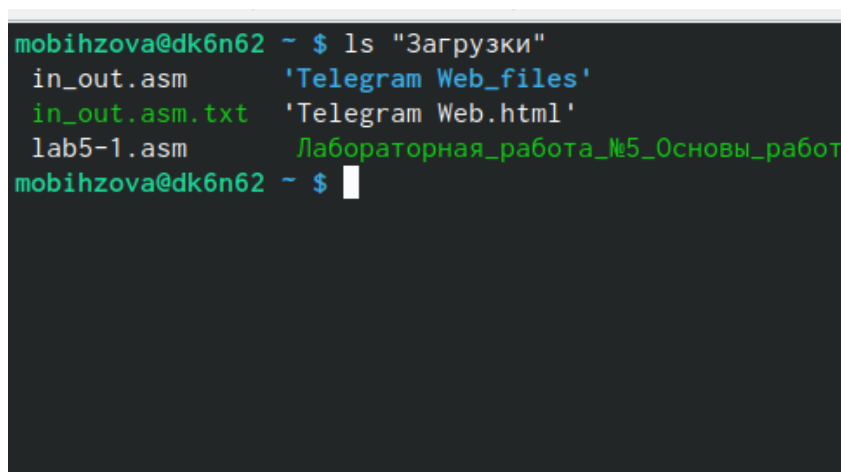
Запускаем исполняемый файл. Программа выводит строку “Введите строку:” и ждет ввода с клавиатуры, вводим свои ФИО, на этом программа заканчивает свою работу (рис. [3.11]).

```
mobihzova@dk8n67 ~/work/arch-pc/lab05 $ ./lab5-1
Введите строку:
Мария Бызова
```

Рис. 3.11: Исполнение файла

3.1 Подключение внешнего файла

9. Скачиваем файл `in_out.asm` со страницы курса в ТУИС. Он сохраняется в каталог “Загрузки” (рис. [3.12]).



```
mobihzova@dk6n62 ~ $ ls "Загрузки"
in_out.asm      'Telegram Web_files'
in_out.asm.txt  'Telegram Web.html'
lab5-1.asm      Лабораторная_работа_№5_Основы_работ
mobihzova@dk6n62 ~ $
```

Рис. 3.12: Скачанный файл

10. С помощью функциональной клавиши F5 копируем файл `in_out.asm` из каталога Загрузки в созданный каталог `lab05` (рис. [3.13]).

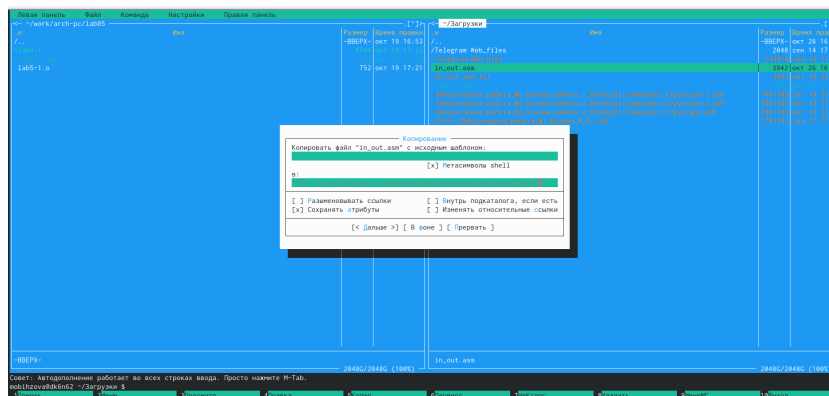


Рис. 3.13: Копирование файла

11. С помощью функциональной клавиши F5 копируем файл `lab5-1` в тот же каталог, но с другим именем, для этого в появившемся окне `mc` прописываем

имя для копии файла (рис. [3.14]).

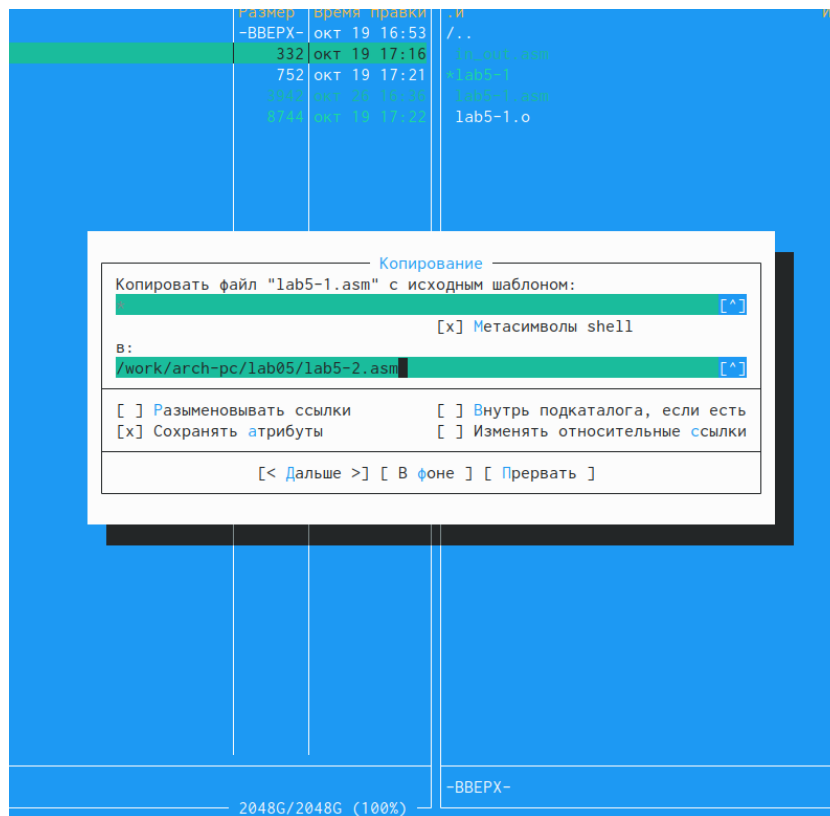


Рис. 3.14: Копирование файла

12. Изменяем содержимое файла lab5-2.asm во встроенном редакторе (рис. [3.15]), чтобы в программе использовались подпрограммы sprintLF, sread и quit из внешнего файла in_out.asm.

```
lab5-2.asm [-M--] 13 L:[ 1+11 12/ 14] *(216 / 247b) 0056 0x038
%include "incbin.asm"
SECTION .data
msg: DB "Введите строку" ,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
```

Рис. 3.15: Редактирование файла

Транслируем текст программы файла в объектный файл командой `nasm -f elf lab5-2.asm`. Создается объектный файл `lab5-2.o`. Выполняем компоновку объектного файла с помощью команды `ld -m elf_i386 -o lab5-2 lab5-2.o`. Создается исполняемый файл `lab5-2`. Запускаем исполняемый файл (рис. [3.16], [3.17],).

```
mobihzova@dk6n62 ~/work/arch-pc/lab05 $ nasm -f elf lab5-2.asm
mobihzova@dk6n62 ~/work/arch-pc/lab05 $ ld -m elf_i386 -o lab5-2 lab5-2.o
```

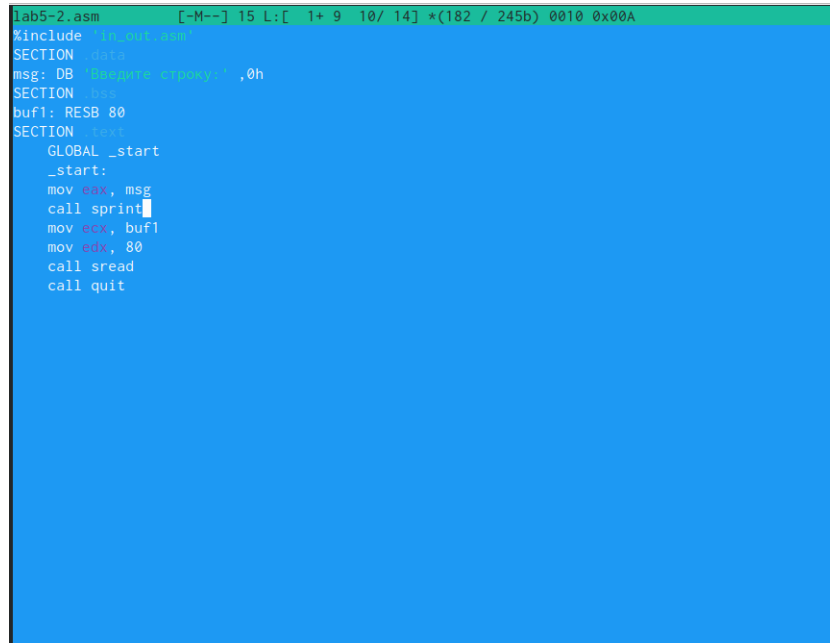
Рис. 3.16: Компиляция файла и передача на обработку компоновщику

```
mobihzova@dk6n62 ~/work/arch-pc/lab05 $ ./lab5-2
Введите строку:

```

Рис. 3.17: Исполнение файла

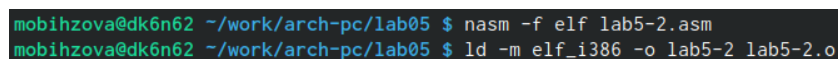
13. Открываем файл lab5-2.asm для редактирования во встроенном редакторе функциональной клавишей F4. Изменяем в нем подпрограмму sprintLF на sprint. Сохраняем изменения и открываем файл для просмотра, чтобы проверить сохранение действий (рис. [3.18]).



```
lab5-2.asm [-M--] 15 L: [ 1+ 9 10/ 14] *(182 / 245b) 0010 0x00A
%include "lab5-2.asm"
SECTION .data
msg: DB "Welcome to NASM!",0h
SECTION .bss
buf: RESB 80
SECTION .text
GLOBAL _start
_start:
mov eax, msg
call sprint
mov ecx, buf
mov edx, 80
call sread
call quit
```

Рис. 3.18: Отредактированный файл

Снова транслируем файл, выполняем компоновку созданного объектного файла. Транслируем текст программы файла в объектный файл командой `nasm -f elf lab5-2.asm`. Создается объектный файл lab5-2.o. Выполняем компоновку объектного файла с помощью команды `ld -m elf_i386 -o lab5-2 lab5-2.o` Создается исполняемый файл lab5-2. Запускаем новый исполняемый файл (рис. [3.19], [3.20]).



```
mobihzova@dk6n62 ~/work/arch-pc/lab05 $ nasm -f elf lab5-2.asm
mobihzova@dk6n62 ~/work/arch-pc/lab05 $ ld -m elf_i386 -o lab5-2 lab5-2.o
```

Рис. 3.19: Компиляция файла и передача на обработку компоновщику


```
mobihzova@dk6n62 ~/work/arch-pc/lab05 $ ./lab5-2
Введите строку: 
```

Рис. 3.20: Исполнение файла

Разница между первым исполняемым файлом и вторым 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 `sprintLF` и `sprint`.

3.2 Выполнение заданий для самостоятельной работы

1. Создаем копию файла `lab5-1.asm` с именем `lab5-1-1.asm` с помощью функциональной клавиши F5 (рис. [3.21]).

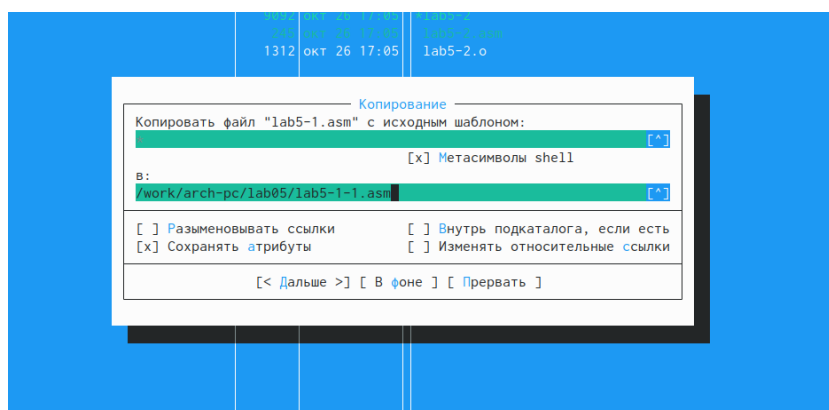


Рис. 3.21: Копирование файла

С помощью функциональной клавиши F4 открываем созданный файл для редактирования. Изменяем программу так, чтобы кроме вывода приглашения и запроса ввода, она выводила вводимую пользователем строку (рис. [3.22]).

```

lab5-1-1.asm  [-M--] 11 L: [ 1+25 26/ 26] *(406 / 406b) <EOF>
SECTION      .data
msg: DB "Введите строку: ",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

```

Рис. 3.22: Редактирование файла

2. Создаем объектный файл lab5-1-1.o, отдаем его на обработку компоновщику, получаем исполняемый файл lab5-1-1, запускаем полученный исполняемый файл. Программа запрашивает ввод, вводим свои ФИО, далее программа выводит введенные нами данные (рис. [3.23], [3.24]).

```

mobihzova@dk6n62 ~/work/arch-pc/lab05 $ nasm -f elf lab5-1-1.asm
mobihzova@dk6n62 ~/work/arch-pc/lab05 $ ld -m elf_i386 -o lab5-1-1 lab5-1-1.o
mobihzova@dk6n62 ~/work/arch-pc/lab05 $ ls
in_out.asm  lab5-1  lab5-1-1  lab5-1-1.asm  lab5-1-1.o  lab5-1.asm  lab5-1.o  lab5-2  lab5-2.asm  lab5-2.o
mobihzova@dk6n62 ~/work/arch-pc/lab05 $

```

Рис. 3.23: Компиляция файла и передача на обработку компоновщику

```

mobihzova@dk6n62 ~/work/arch-pc/lab05 $ ./lab5-1-1
Введите строку:
Мария Бызова
Мария Бызова
mobihzova@dk6n62 ~/work/arch-pc/lab05 $

```

Рис. 3.24: Исполнение файла

Код программы из пункта 1:

```

SECTION .data ; Секция инициированных данных
msg: DB "Введите строку: ",10

```

```

msgLen: EQU $-msg ; Длина переменной 'msg'
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,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. Создаем копию файла lab5-2.asm с именем lab5-2-2.asm с помощью функциональной клавиши F5 (рис. [3.25]).

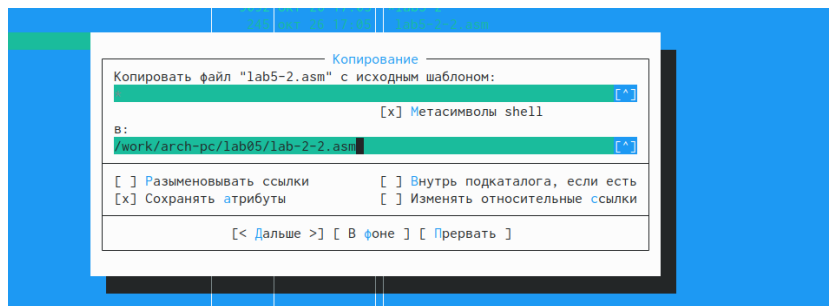


Рис. 3.25: Копирование файла

С помощью функциональной клавиши F4 открываем созданный файл для редактирования. Изменяем программу так, чтобы кроме вывода приглашения и запроса ввода, она выводила вводимую пользователем строку (рис. [3.26]).

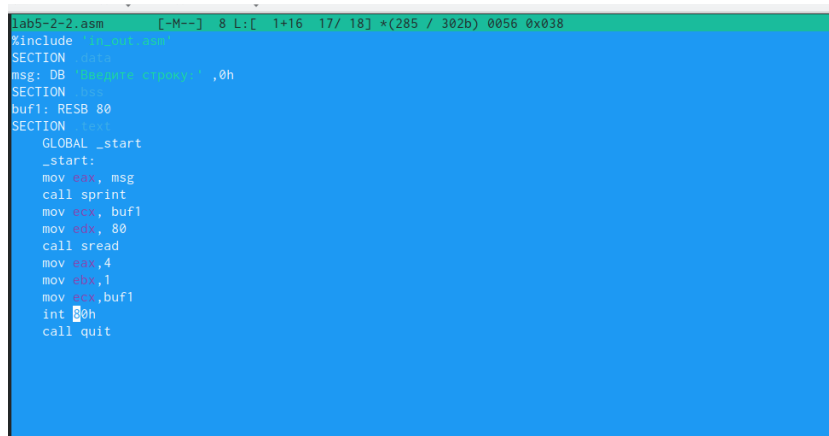


Рис. 3.26: Редактирование файла

4. Создаем объектный файл lab5-2-2.o, отдаем его на обработку компоновщику, получаем исполняемый файл lab5-2-2, запускаем полученный исполняемый файл. Программа запрашивает ввод без переноса на новую строку, вводим свои ФИО, далее программа выводит введенные нами данные (рис. [3.27], [3.28]).

```

mobihzova@dk6n62 ~/work/arch-pc/lab05 $ nasm -f elf lab5-2-2.asm
mobihzova@dk6n62 ~/work/arch-pc/lab05 $ ld -m elf_i386 -o lab5-2-2 lab5-2-2.o
mobihzova@dk6n62 ~/work/arch-pc/lab05 $ ls
in_out.asm lab5-1 lab5-1-1 lab5-1-1.o lab5-1.asm lab5-1.o lab5-2 lab5-2-2 lab5-2-2.asm lab5-2-2.o lab5-2.asm lab5-2.o
mobihzova@dk6n62 ~/work/arch-pc/lab05 $

```

Рис. 3.27: Компиляция файла и передача на обработку компоновщику

```

mobihzova@dk6n62 ~/work/arch-pc/lab05 $ ./lab5-2-2
Введите строку: Бызова
mobihzova@dk6n62 ~/work/arch-pc/lab05 $

```

Рис. 3.28: 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 ; Описатель файла '1' - стандартный вывод
mov ecx,buf1 ; Адрес строки buf1 в ecx
int 80h ; Вызов ядра
call quit ; вызов подпрограммы завершения

```

4 Выводы

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