

Отчет по лабораторной работе №5

Дисциплина: архитектура компьютера

Выполняла: Романова Елизавета Романовна

НКАбд-04-24

2024

Содержание

1	Цель работы	2
2	Задание	4
3	Теоретическое введение.....	4
4	Выполнение лабораторной работы.....	5
4.1	Основы работы с тс	5
4.2	Структура программы на языке ассемблера NASM	8
4.3	Подключение внешнего файла.....	10
4.4	Выполнение заданий для самостоятельной работы	12
5	Выводы	16

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 инициализированных (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 инициализированных 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-байтное 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. 1).

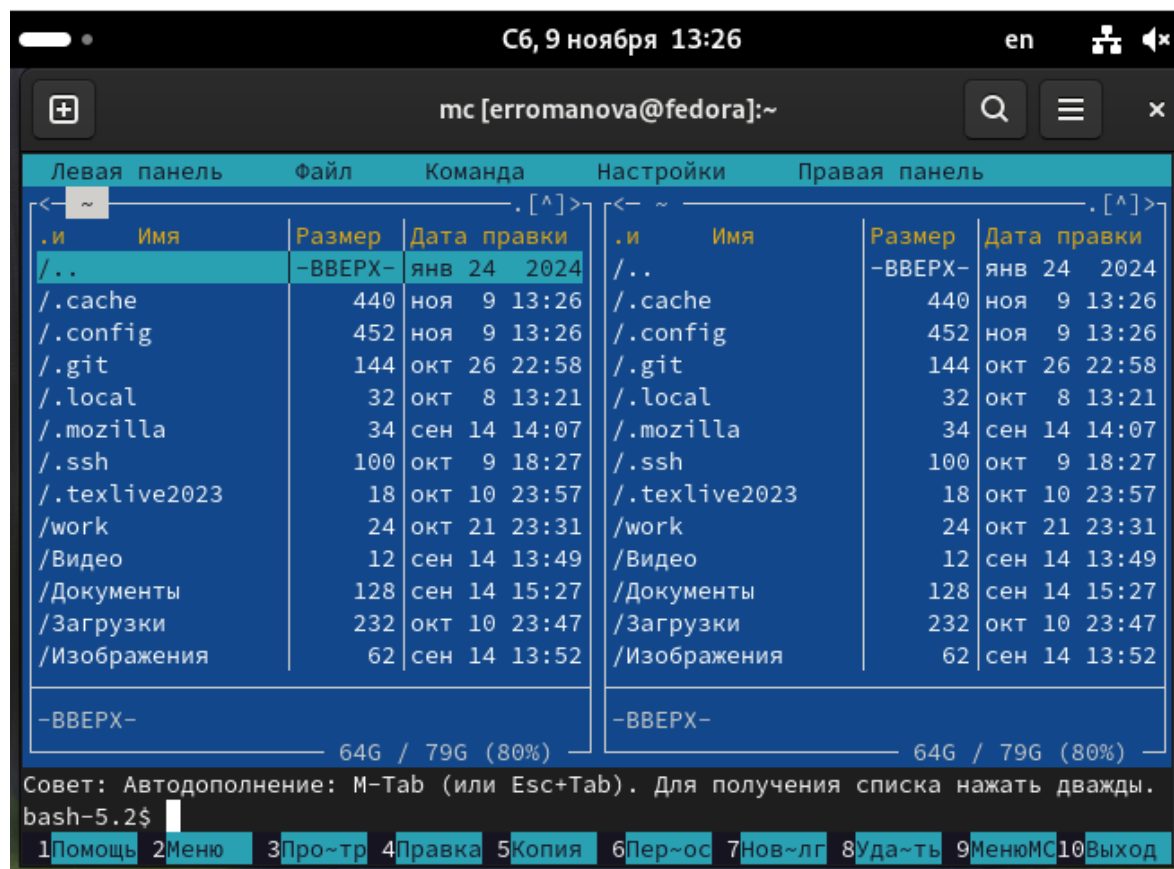


Рис. 1: Открытый mc

Перехожу в каталог ~/work/study/2024-2025/Архитектура Компьютера/arch-рс, используя файловый менеджер mc (рис. 2)

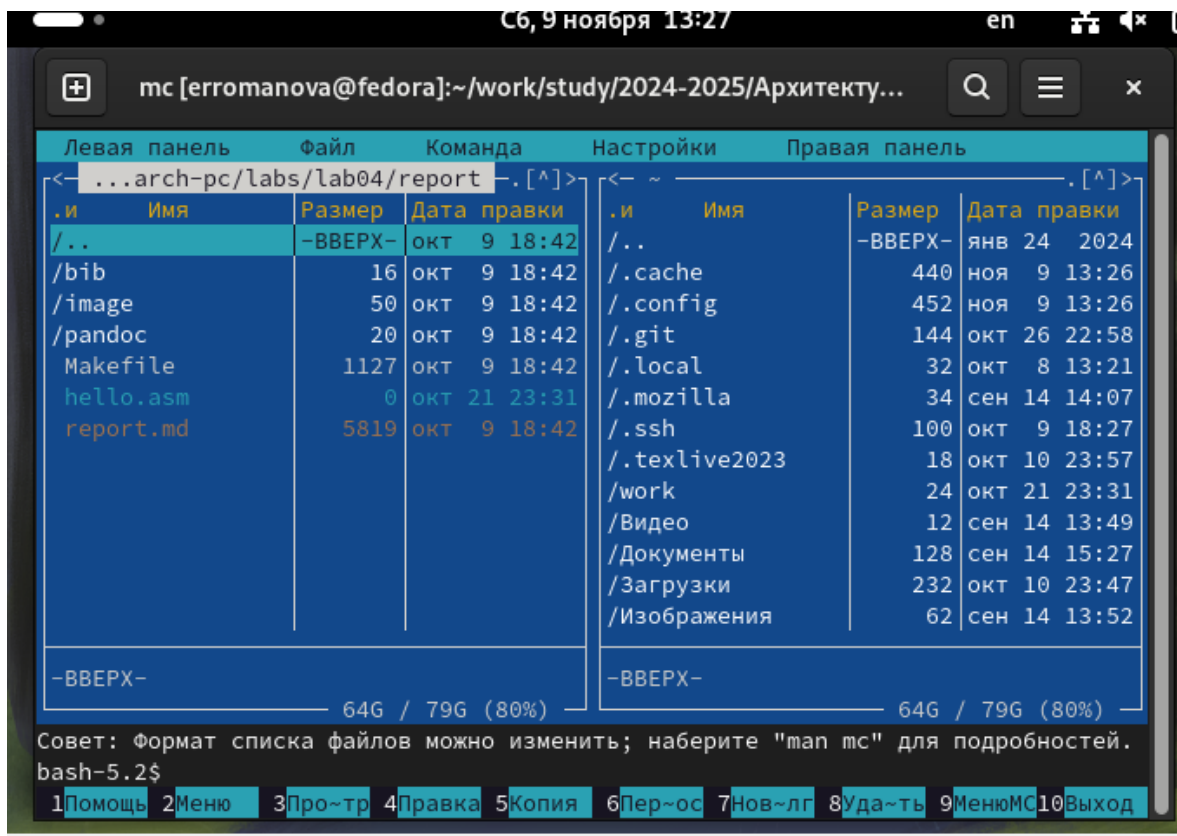


Рис. 2: Перемещение между директориями

С помощью функциональной клавиши F7 создаю каталог lab06 (рис. 3).

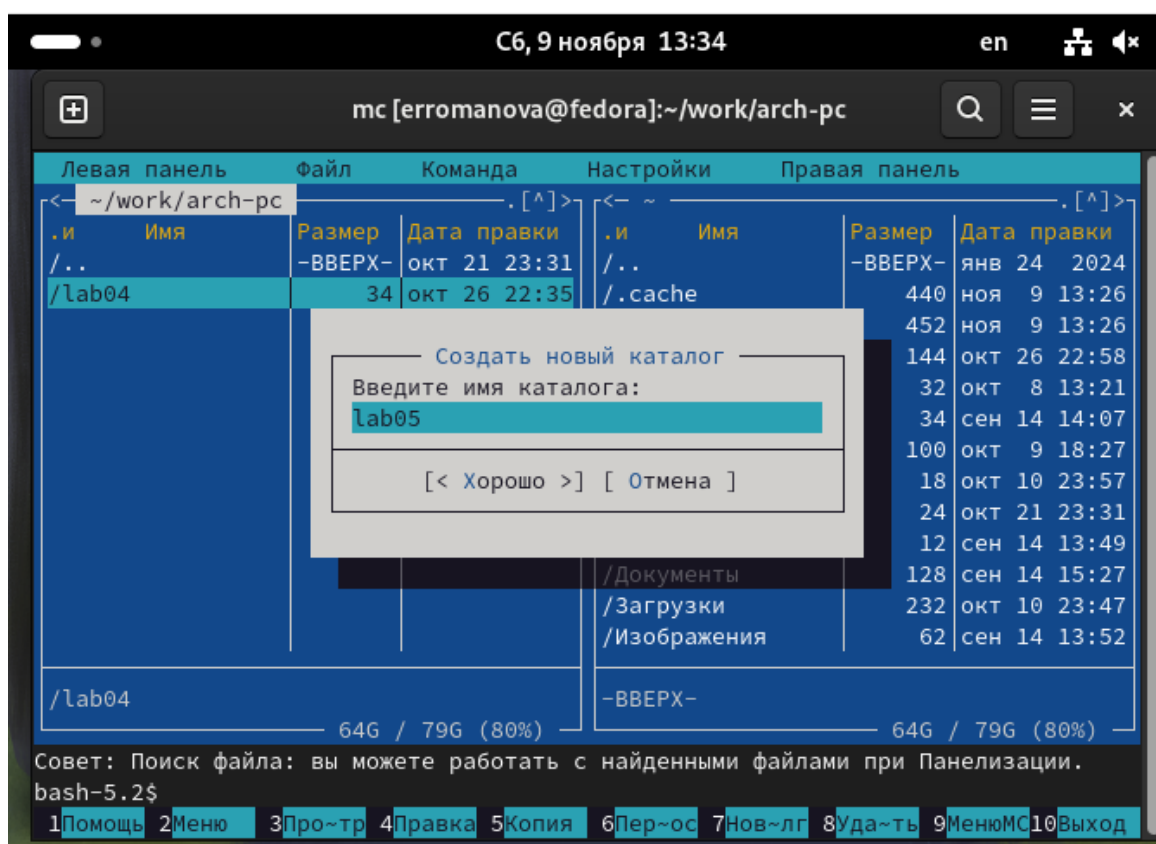


Рис. 3: Создание каталога

Переходу в созданный каталог (рис. 4).

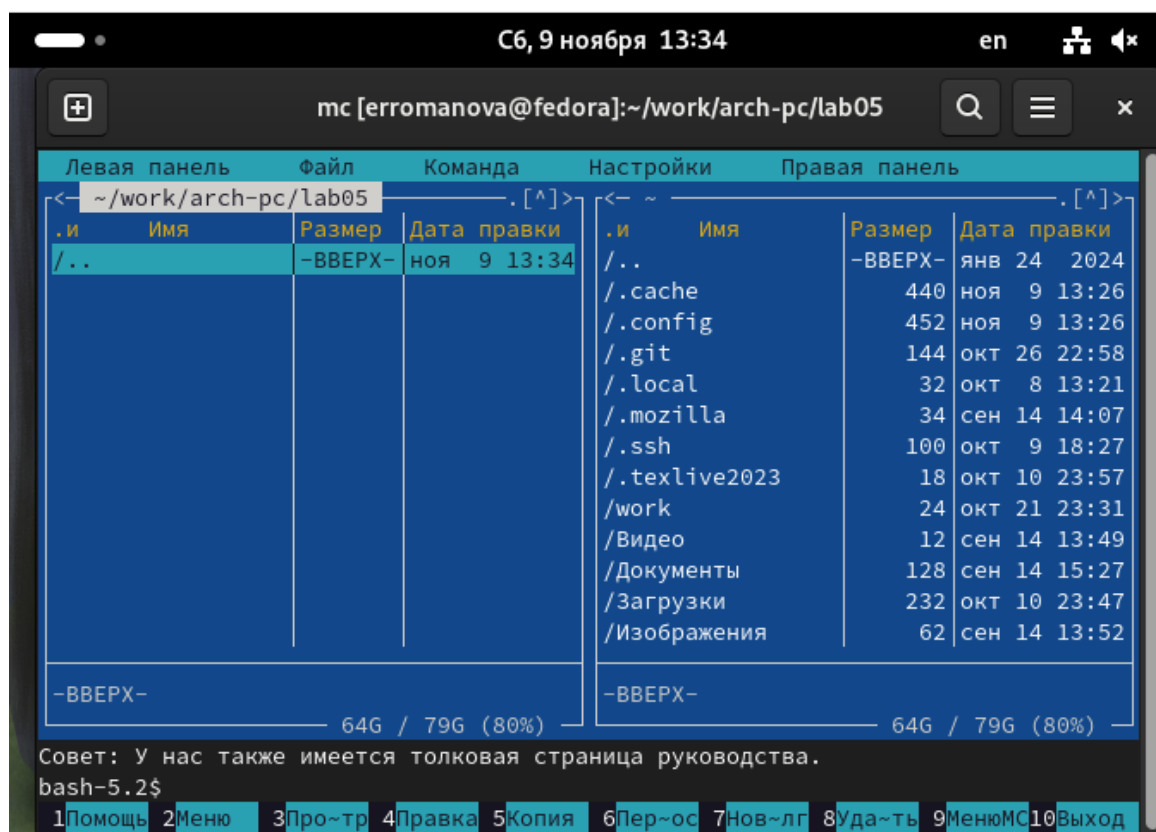


Рис. 4: Перемещение между директориями

В строке ввода прописываю команду `touch lab6-1.asm`, чтобы создать файл, в котором буду работать (рис. 5).

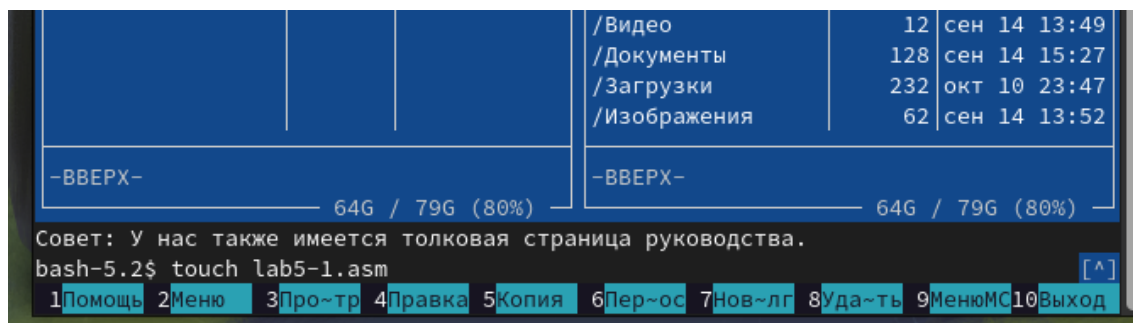
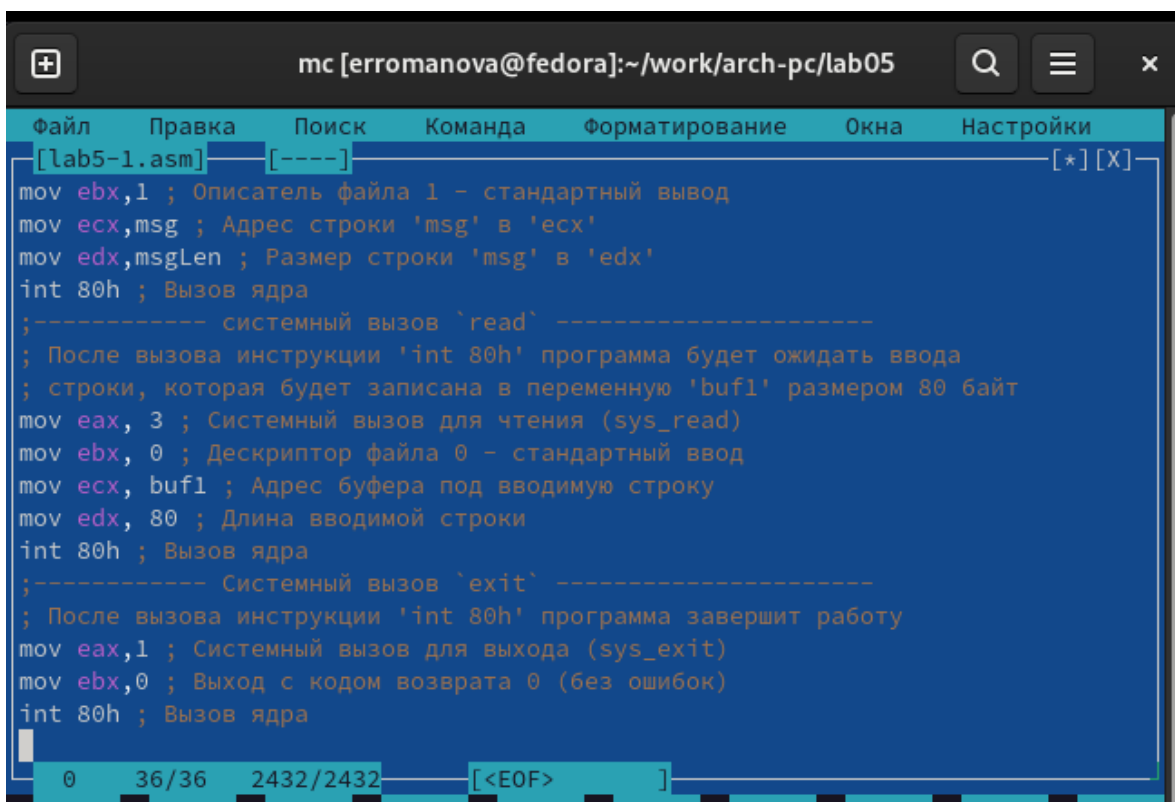


Рис. 5: Создание файла

4.2 Структура программы на языке ассемблера NASM

Ввожу в файл код программы для запроса строки у пользователя (рис. 6). Далее выхожу из файла (Ctrl+X), сохраняя изменения (Y, Enter).



The screenshot shows a text editor window titled "mc [erromanova@fedora]:~/work/arch-pc/lab05". The menu bar includes "Файл", "Правка", "Поиск", "Команда", "Форматирование", "Окна", and "Настройки". The editor displays assembly code for a program that reads a message and exits. The code is as follows:

```
[lab5-1.asm] [----] [*] [X]
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 ;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 `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
```

The status bar at the bottom shows "0 36/36 2432/2432" and a "[<EOF>]" indicator.

Рис. 6: Редактирование файла

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

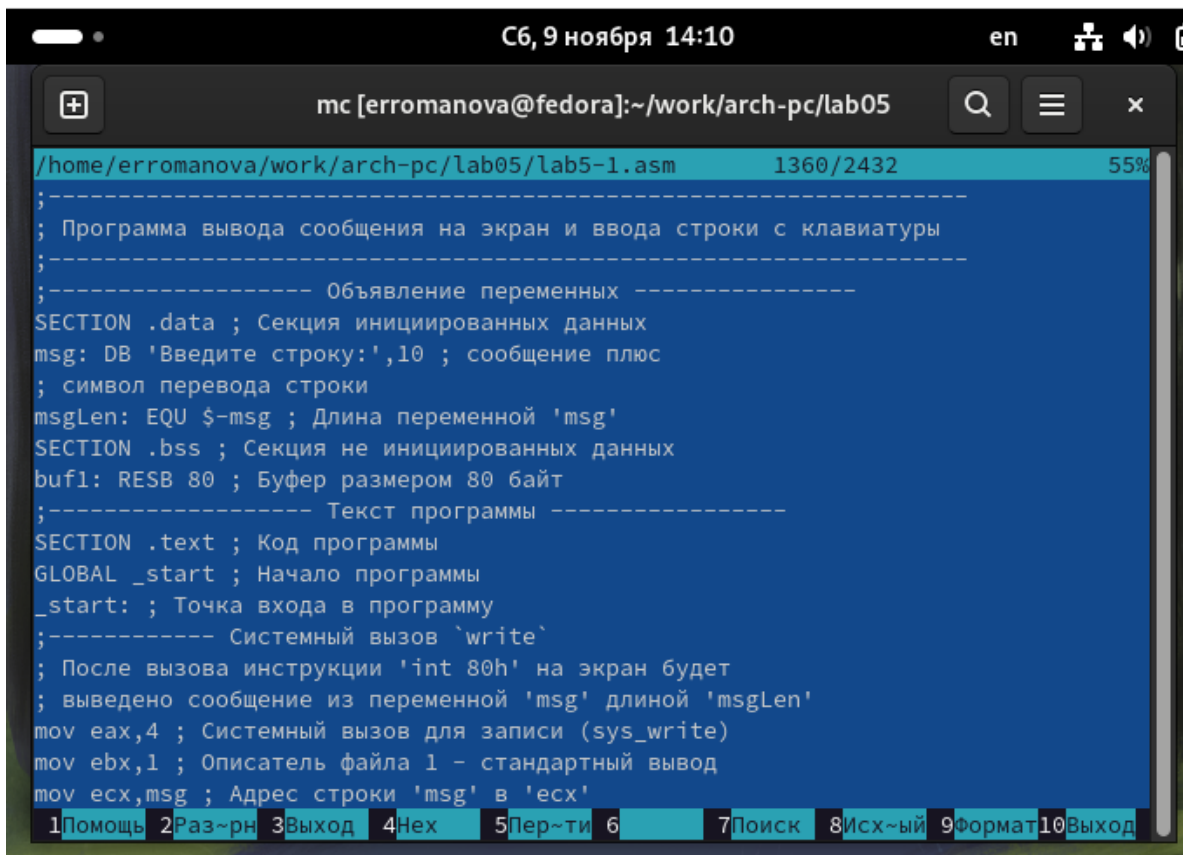


Рис. 7: Открытие файла для просмотра

Транслирую текст программы файла в объектный файл командой `nasm -f elf lab6-1.asm`. Создался объектный файл `lab6-1.o`. Выполняю компоновку объектного файла с помощью команды `ld -m elf_i386 -o lab6-1 lab6-1.o` (рис. 8). Создался исполняемый файл `lab6-1`.

```
bash-5.2$ nasm -f elf lab5-1.asm
bash-5.2$ ld -m elf_i386 -o lab5-1 lab5-1.o
```

Рис. 8: Компиляция файла и передача на обработку компоновщику

Запускаю исполняемый файл. Программа выводит строку “Введите строку:” и ждет ввода с клавиатуры, я ввожу свои ФИО, на этом программа заканчивает свою работу (рис. 9).

```
bash-5.2$ ./lab5-1
Введите строку:
Романова Елизавета Романовна
```

Рис. 9: Исполнение файла

4.3 Подключение внешнего файла

Скачиваю файл `in_out.asm` со страницы курса в ТУИС. Он сохранился в каталог “Загрузки” (рис. 10).

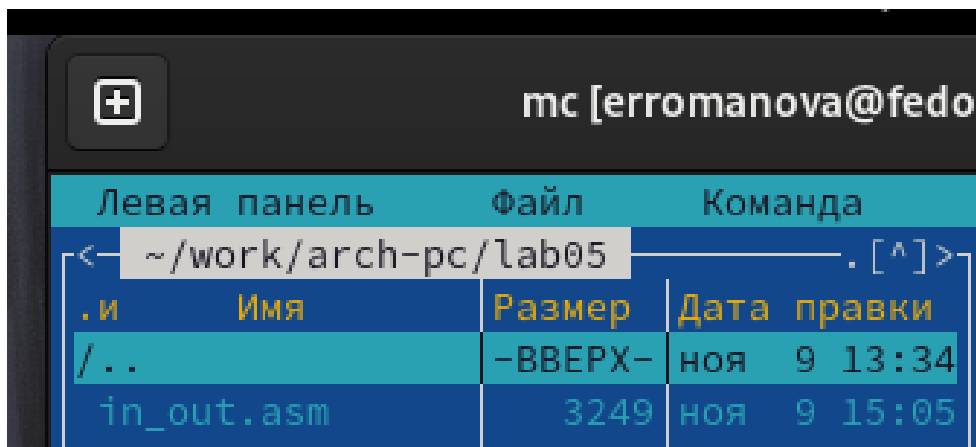


Рис. 10: Скачанный файл

Изменяю содержимое файла lab6-2.asm во встроенном редакторе nano (рис. 11), чтобы в программе использовались подпрограммы из внешнего файла in_out.asm.

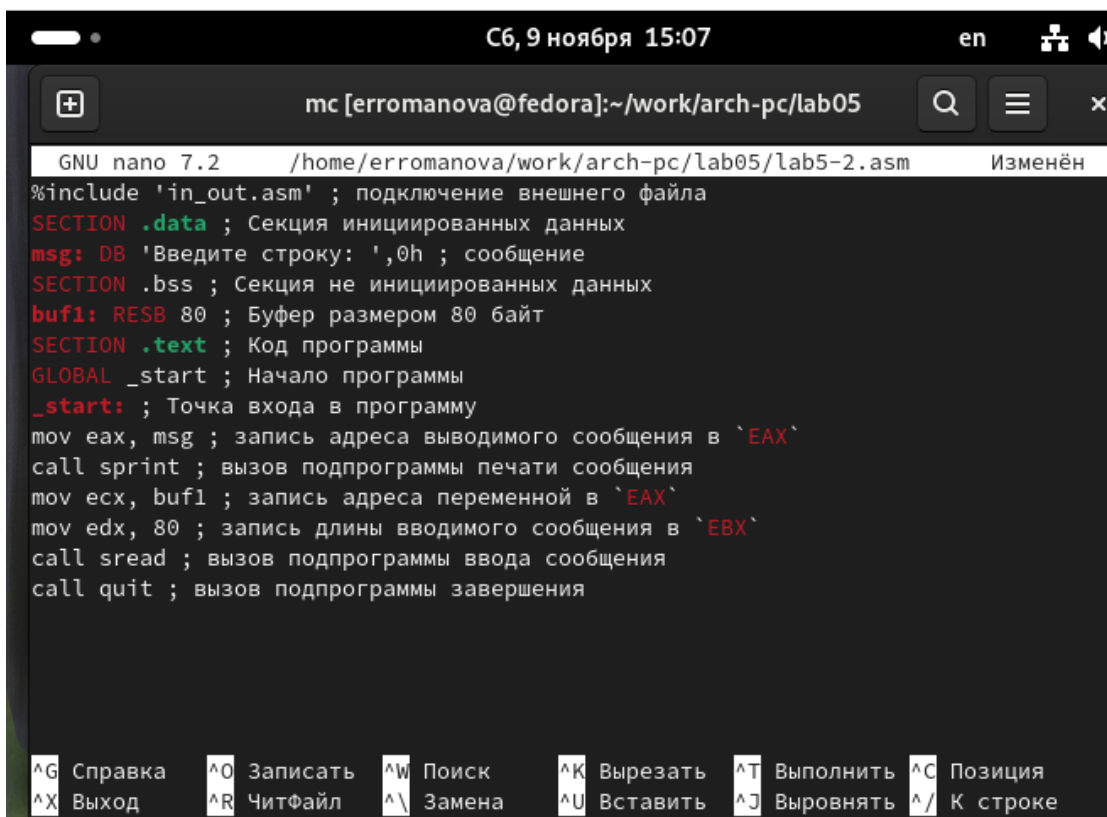


Рис. 11: Редактирование файла

Транслирую текст программы файла в объектный файл командой `nasm -f elf lab6-2.asm`. Создался объектный файл lab6-2.o. Выполняю компоновку объектного файла с помощью команды `ld -m elf_i386 -o lab6-2 lab6-2.o` Создался исполняемый файл lab6-2. Запускаю исполняемый файл (рис. 12).

```

bash-5.2$ nasm -f elf lab5-2.asm
bash-5.2$ ld -m elf_i386 -o lab5-2 lab5-2.o
bash-5.2$ ./lab5-2
Введите строку:
Романова Елизавета Романовна

```

Рис. 12: Исполнение файла

Открываю файл lab6-2.asm для редактирования в nano функциональной клавишей F4. Изменяю в нем подпрограмму `sprintLF` на `sprint`. Сохраняю изменения и открываю файл для просмотра, чтобы проверить сохранение действий (рис. 13).

```

#include 'in_out.asm'
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
call quit ; вызов подпрограммы завершения

```

Рис. 13: Отредактированный файл

Снова транслирую файл, выполняю компоновку созданного объектного файла, запускаю новый исполняемый файл (рис. 14).

```

bash-5.2$ nasm -f elf lab5-2.asm
bash-5.2$ ld -m elf_i386 -o lab5-2-2 lab5-2.o
bash-5.2$ ./lab5-2-2
Введите строку: Романова Елизавета Романовна

```

Рис. 14: Исполнение файла

Разница между первым исполняемым файлом lab6-2 и вторым lab6-2-2 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`.

4.4 Выполнение заданий для самостоятельной работы

1. Создаю копию файла lab6-1.asm с именем lab6-1-1.asm с помощью функциональной клавиши F5 (рис. 15).

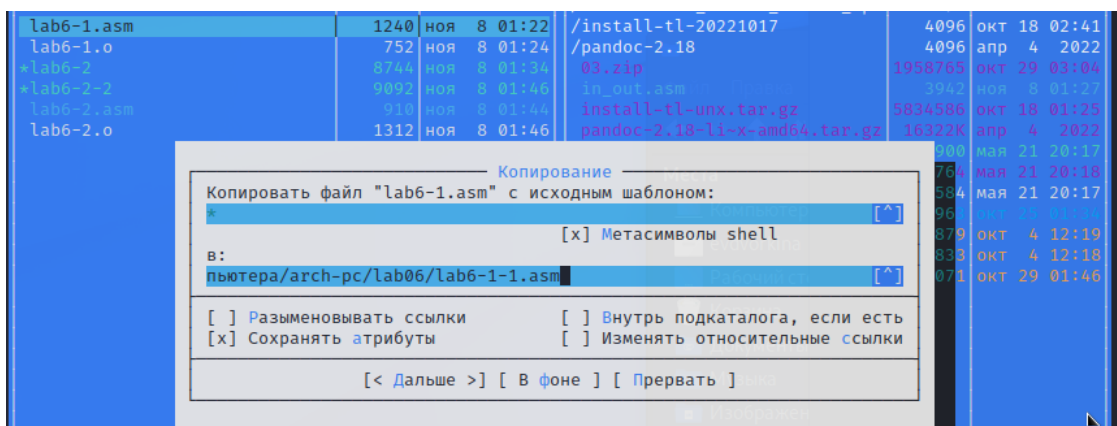


Рис. 15: Копирование файла

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. 16).

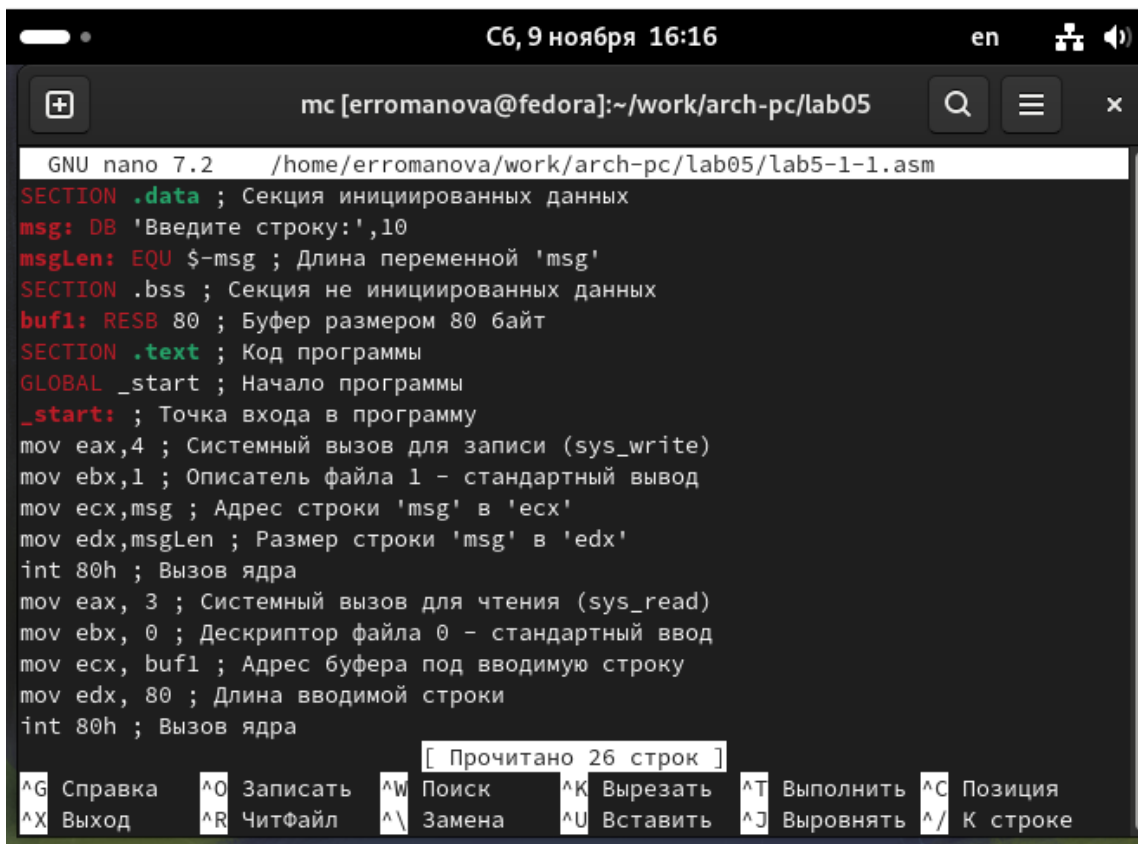


Рис. 16: Редактирование файла

2. Создаю объектный файл lab6-1-1.o, отдаю его на обработку компоновщику, получаю исполняемый файл lab6-1-1, запускаю полученный исполняемый файл. Программа запрашивает ввод, ввожу свои ФИО, далее программа выводит введенные мною данные (рис. 17).

```

bash-5.2$ nasm -f elf lab5-1-1.asm
bash-5.2$ ld -m elf_i386 -o lab5-1-1 lab5-1-1.o
bash-5.2$ ./lab5-1-1
Введите строку:
Романова Елизавета Романовна
Романова Елизавета Романовна

```

Рис. 17: Исполнение файла

Код программы из пункта 1:

```

SECTION .data ; Секция инициализированных данных
msg: DB 'Введите строку:',10
msgLen: EQU $-msg ; Длина переменной 'msg'
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,4 ; Системный вызов для записи (sys_write)
mov ebx,1 ; Описание 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 ; Описание 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. Создаю копию файла lab6-2.asm с именем lab6-2-1.asm с помощью функциональной клавиши F5 (рис. 18).

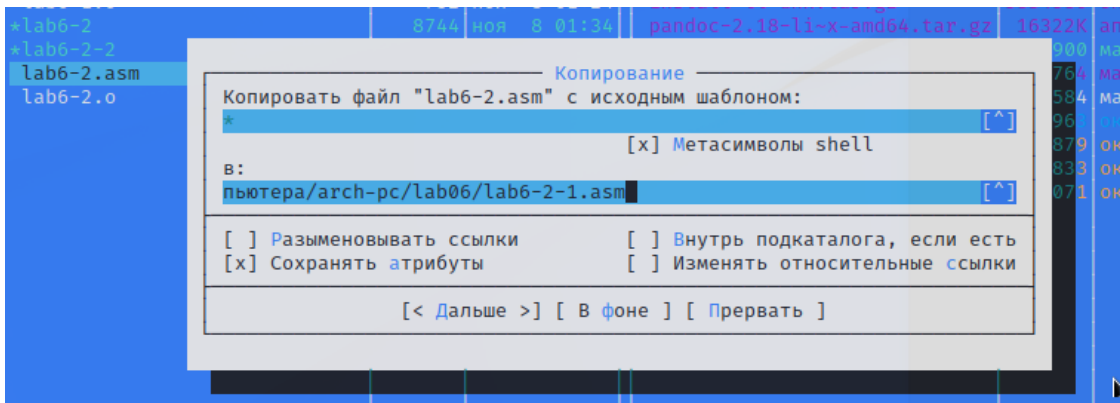


Рис. 18: Копирование файла

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. 19)

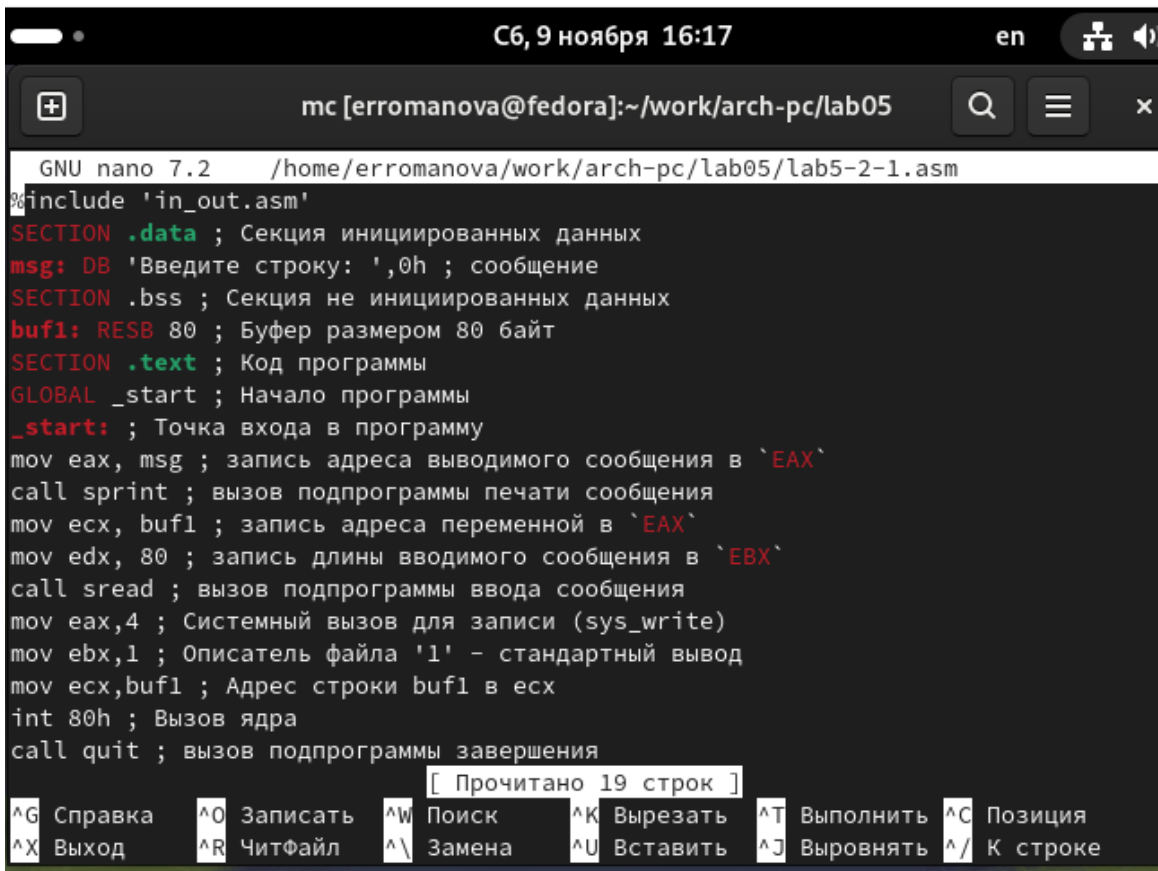


Рис. 19: Редактирование файла

4. Создаю объектный файл lab6-2-1.o, отдаю его на обработку компоновщику, получаю исполняемый файл lab6-2-1, запускаю полученный исполняемый файл. Программа запрашивает ввод без переноса на новую строку, ввожу свои ФИО, далее программа выводит введенные мною данные (рис. 20).

```
bash-5.2$ nasm -f elf lab5-2-1.asm
bash-5.2$ ld -m elf_i386 -o lab5-2-1 lab5-2-1.o
bash-5.2$ ./lab5-2-1
Введите строку: Романова Елизвета Романовна
Романова Елизвета Романовна
```

Рис. 20: Исполнение файла

Код программы из пункта 3:

```
%include 'in_out.asm'
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 ; Описание файла '1' - стандартный вывод
mov ecx,buf1 ; Адрес строки buf1 в ecx
int 80h ; Вызов ядра
call quit ; вызов подпрограммы завершения
```


5 Вывод

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