

Отчёт по лабораторной работе №5

Дисциплина: Архитектура компьютера

Кириянова Екатерина Андреевна

Содержание

1	Цель работы	4
2	Задание	5
3	Теоретическое введение	6
4	Выполнение лабораторной работы	8
4.1	Основы работы с Midnight Commander	8
4.2	Структура программы на языке ассемблера NASM	9
4.3	Подключение внешнего файла	11
4.4	Задание для самостоятельной работы	16
5	Выводы	22
6	Список литературы	23

Список иллюстраций

4.1	Midnight Commander	8
4.2	Новый каталог	9
4.3	Новый файл	9
4.4	Nano	10
4.5	Программа	10
4.6	Сохранение	11
4.7	Запуск программы	11
4.8	Обе панели	12
4.9	Перемещение файла	12
4.10	Результат копирования	13
4.11	Создание копии	13
4.12	Копия файла	14
4.13	Текст программы	14
4.14	Запуск файла	15
4.15	Изменение программы	15
4.16	Запуск	15
4.17	Копирование файла	16
4.18	Редактирование файла	17
4.19	Запуск программы	17
4.20	Копирование файла	19
4.21	Редактирование файла	19
4.22	Запуск программы	20

1 Цель работы

Приобрести практические навыки работы в Midnight Commander и освоить инструкции 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 Midnight Commander

Открываю Midnight Commander и перехожу в нужный каталог (рис. 4.1).

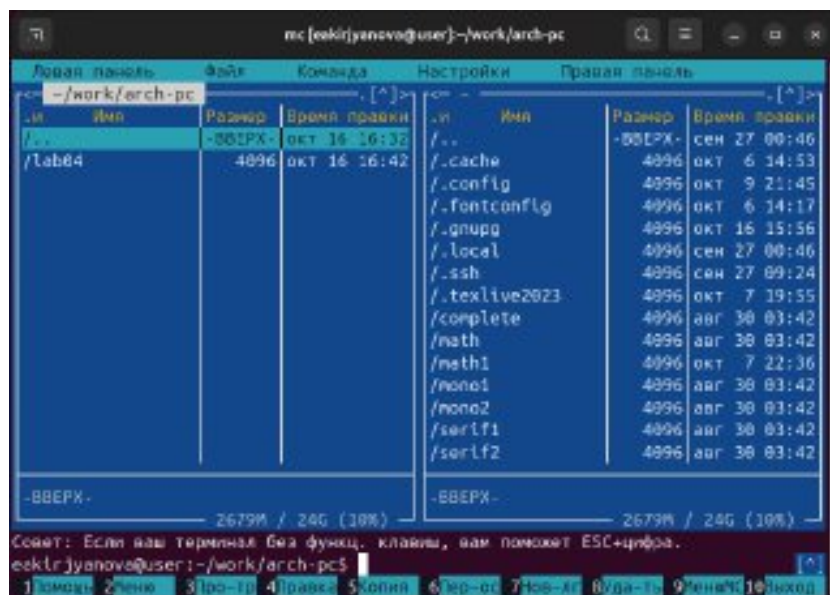


Рис. 4.1: Midnight Commander

Создаю новый каталог с помощью клавиши F7 и перехожу в него (рис. 4.2)

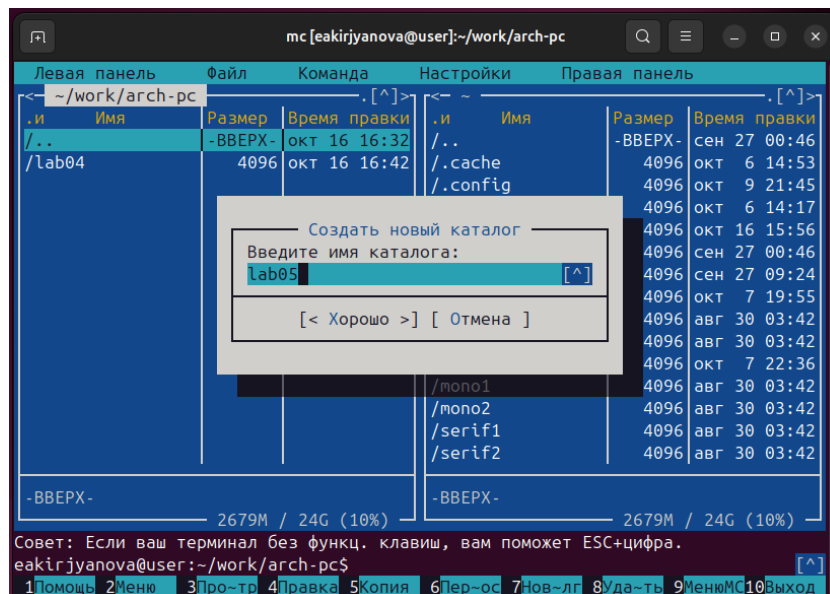


Рис. 4.2: Новый каталог

Создаю новый файл lab5-1.asm (рис. 4.3).

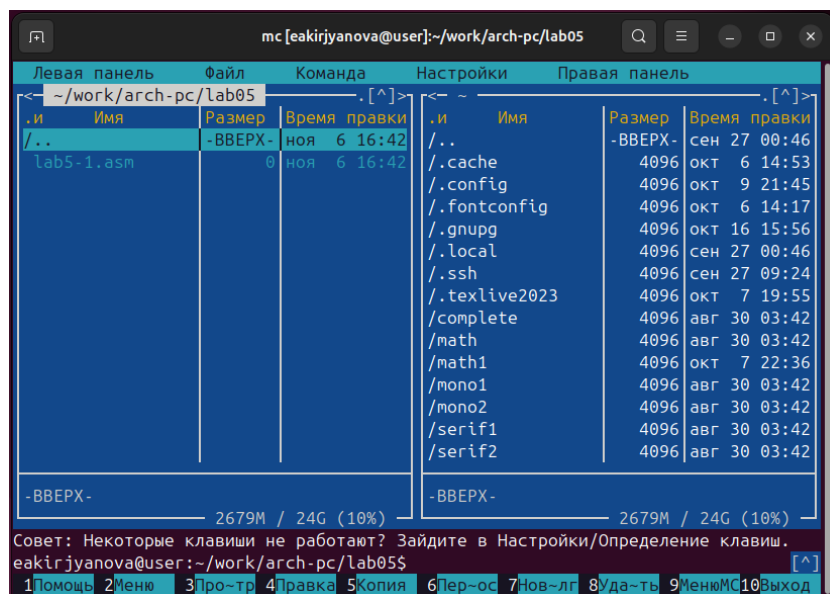


Рис. 4.3: Новый файл

4.2 Структура программы на языке ассемблера NASM

Открываю файл в текстовом редакторе nano (рис. 4.4).

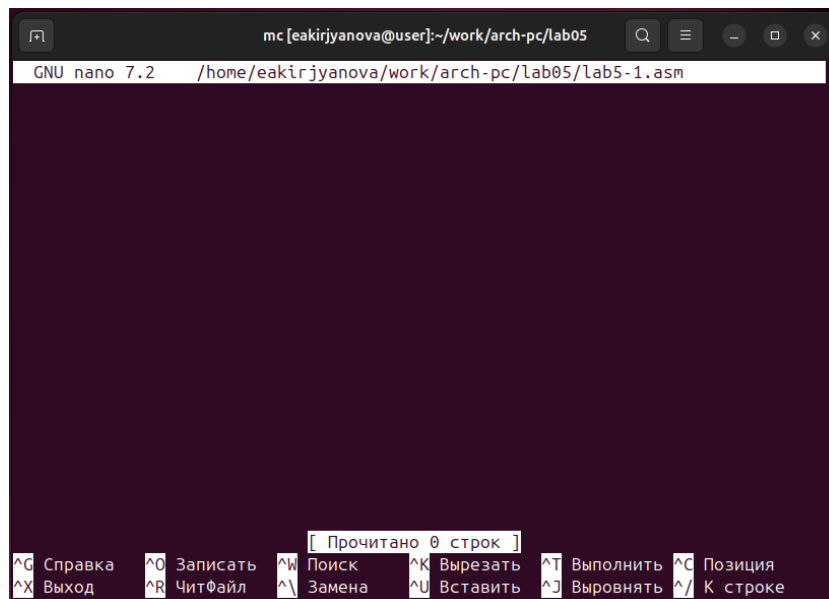


Рис. 4.4: Nano

Ввожу текст программы из листинга 5.1 (рис. 4.5).

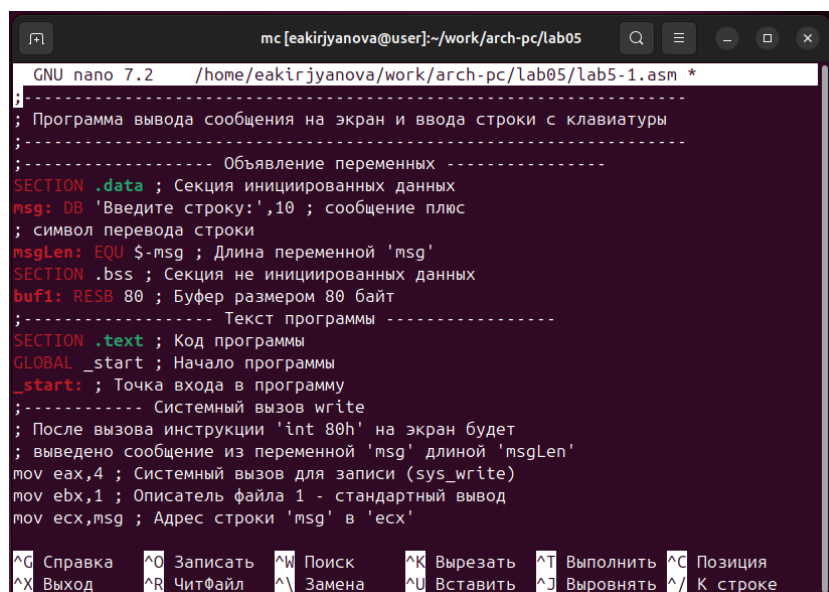
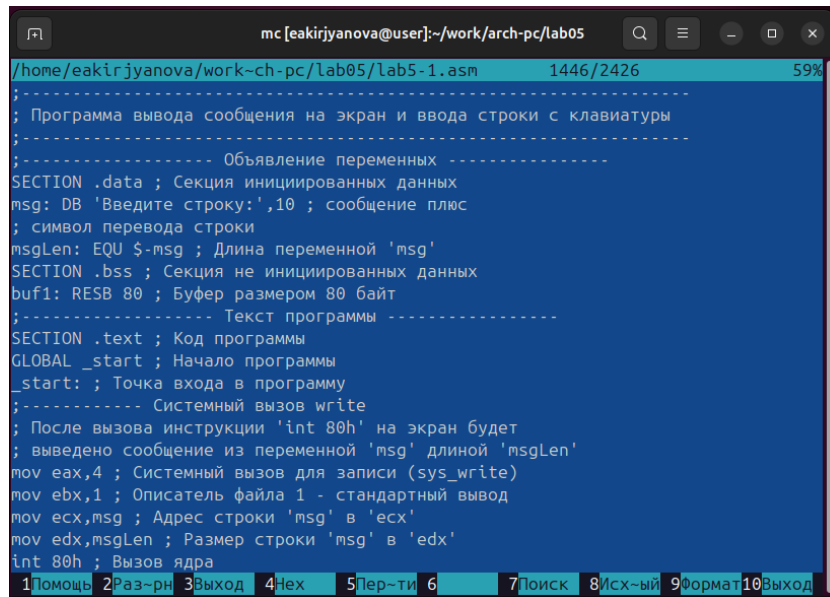


Рис. 4.5: Программа

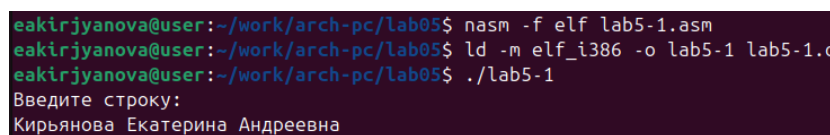
Выхожу из файла с помощью клавиш Ctrl+X и сохраняю с помощью клавиш Y, Enter. Далее открываю файл клавишей F3 и проверяю успешность выполнения (рис. 4.6).



```
mc [eakirjyanova@user]:~/work/arch-pc/lab05
/home/eakirjyanova/work-ch-pc/lab05/lab5-1.asm 1446/2426 59%
;-----
; Программа вывода сообщения на экран и ввода строки с клавиатуры
;-----
;----- Объявление переменных -----
SECTION .data ; Секция инициированных данных
msg: DB 'Введите строку:',10 ; сообщение плюс
; символ перевода строки
msgLen: EQU $-msg ; Длина переменной 'msg'
SECTION .bss ; Секция не инициированных 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
1Помощь 2Раз-рн 3Выход 4Тех 5Пер-ти 6 7Поиск 8Исх-ый 9Формат10Выход
```

Рис. 4.6: Сохранение

Транслирую текст программы в объектный файл, выполняю компоновку объектного файла и запускаю исполняемый файл. После строки “Введите строку:” ввожу ФИО (рис. 4.7).



```
eakirjyanova@user:~/work/arch-pc/lab05$ nasm -f elf lab5-1.asm
eakirjyanova@user:~/work/arch-pc/lab05$ ld -m elf_i386 -o lab5-1 lab5-1.o
eakirjyanova@user:~/work/arch-pc/lab05$ ./lab5-1
Введите строку:
Кириянова Екатерина Андреевна
```

Рис. 4.7: Запуск программы

4.3 Подключение внешнего файла

Открываю две панели (рис. 4.8).

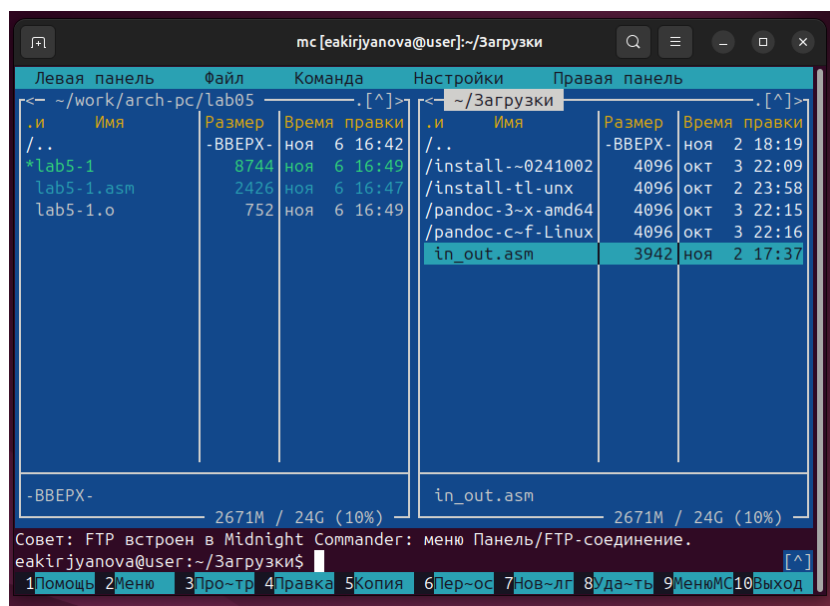


Рис. 4.8: Обе панели

Копирую внешний файл в нужный каталог с помощью клавиши F5 (рис. 4.9).

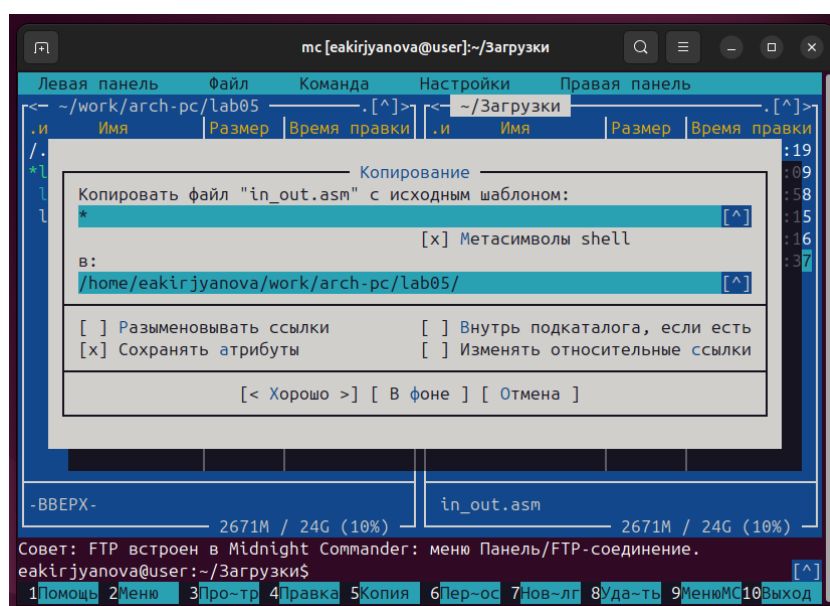


Рис. 4.9: Перемещение файла

Копирование прошло успешно (рис. 4.10).

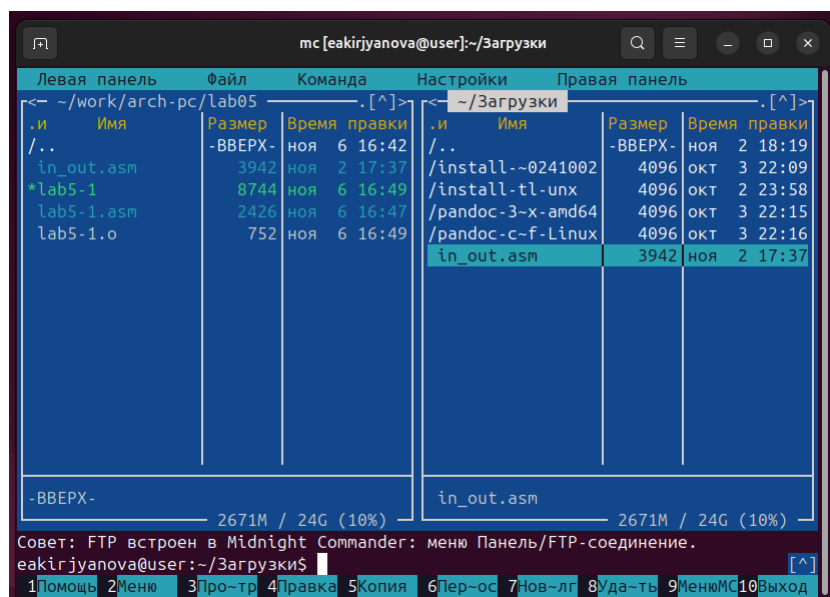


Рис. 4.10: Результат копирования

Создаю копию файла lab5-1.asm с именем lab5-2.asm с помощью клавиши F5 (рис. 4.11).

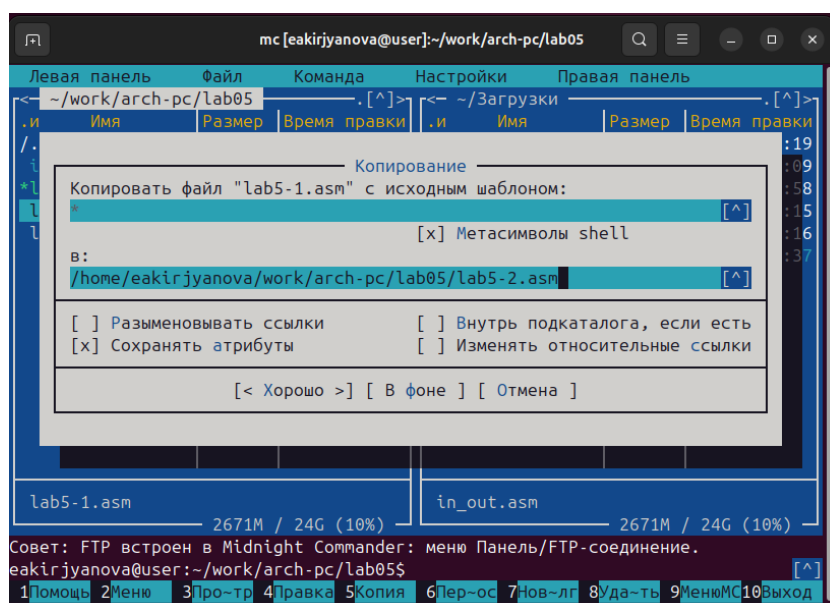


Рис. 4.11: Создание копии

Демонстрирую успешность выполнения (рис. 4.12).

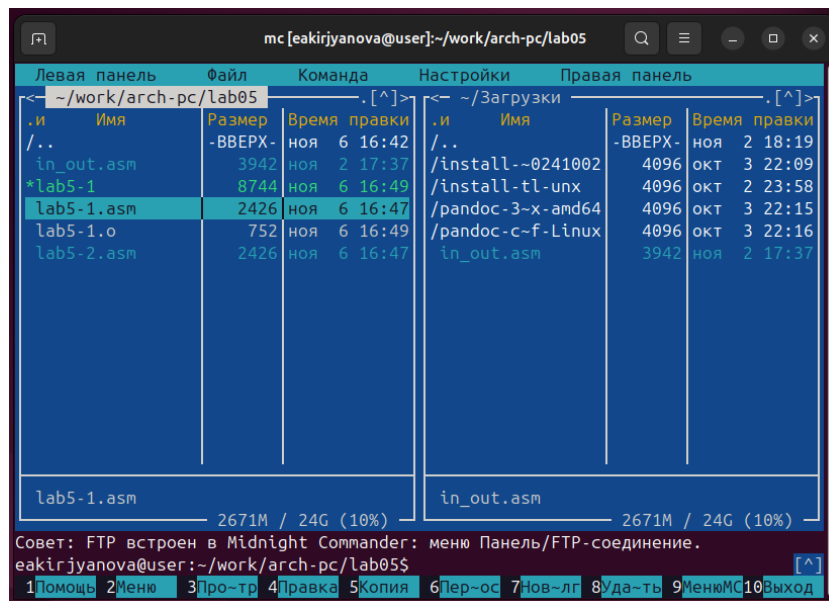


Рис. 4.12: Копия файла

Редактирую файл согласно листингу 5.2 (рис. 4.13).

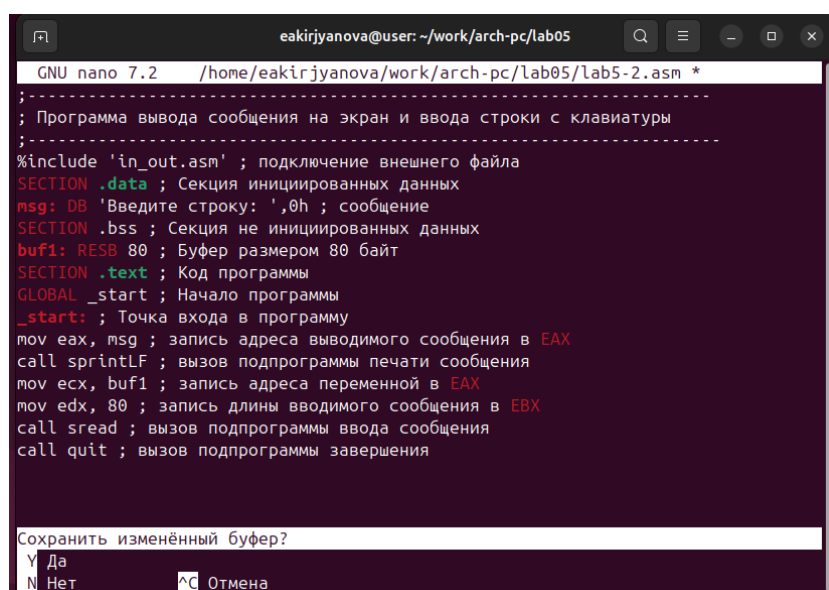


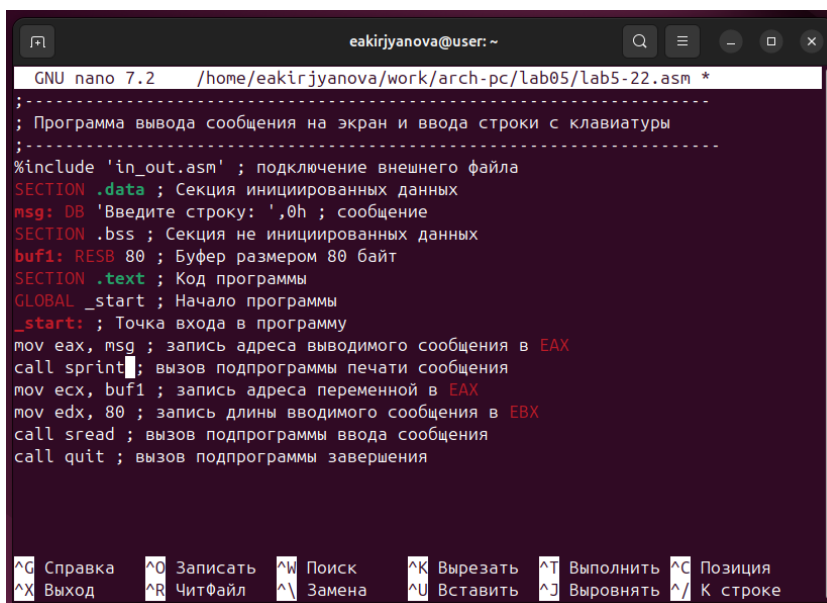
Рис. 4.13: Текст программы

Транслирую текст программы в объектный файл, выполняю компоновку объектного файла и запускаю исполняемый файл (рис. 4.14).

```
eakirjyanova@user:~/work/arch-pc/lab05$ nasm -f elf lab5-2.asm
eakirjyanova@user:~/work/arch-pc/lab05$ ld -m elf_i386 -o lab5-2 lab5-2.o
eakirjyanova@user:~/work/arch-pc/lab05$ ./lab5-2
Введите строку:
Кириянова Екатерина Андреевна
```

Рис. 4.14: Запуск файла

Создаю файл lab5-22.asm на основе файла lab5-2.asm и меняю sprintLF на sprint (рис. 4.15).



```
GNU nano 7.2 /home/eakirjyanova/work/arch-pc/lab05/lab5-22.asm *
;-----
; 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

^G Справка ^O Записать ^W Поиск ^K Вырезать ^T Выполнить ^C Позиция
^X Выход ^R ЧитФайл ^\ Замена ^U Вставить ^J Выровнять ^/_ К строке
```

Рис. 4.15: Изменение программы

Снова транслирую в объектный файл, выполняю его компоновку и запускаю (рис. 4.16).

```
eakirjyanova@user:~/work/arch-pc/lab05$ nasm -f elf lab5-22.asm
eakirjyanova@user:~/work/arch-pc/lab05$ ld -m elf_i386 -o lab5-22 lab5-22.o
eakirjyanova@user:~/work/arch-pc/lab05$ ./lab5-22
Введите строку: Кириянова Екатерина Андреевна
```

Рис. 4.16: Запуск

Разница между первым исполняемым файлом lab5-2 и вторым lab5-22 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 Задание для самостоятельной работы

Создаю копию файла lab5-1.asm с именем lab5-11.asm с помощью клавиши F5 (рис. 4.17).

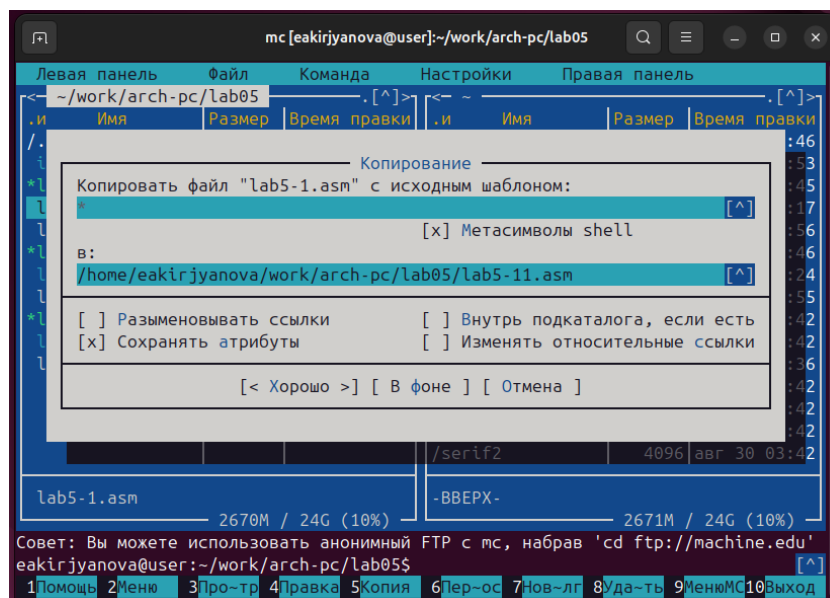


Рис. 4.17: Копирование файла

Изменяю программу, чтобы кроме запроса ввода она выводила заданную строку (рис. 4.18).


```

GNU nano 7.2 /home/eakiryanova/work/arch-pc/lab05/lab5-11.asm *
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 ; 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

```

Рис. 4.18: Редактирование файла

Транслирую текст программы в объектный файл, выполняю компоновку объектного файла и запускаю исполняемый файл. Программа выводит заданную мною строку (рис. 4.19).

```

eakiryanova@user:~/work/arch-pc/lab05$ nasm -f elf lab5-11.asm
eakiryanova@user:~/work/arch-pc/lab05$ ld -m elf_i386 -o lab5-11 lab5-11.o
eakiryanova@user:~/work/arch-pc/lab05$ ./lab5-11
Введите строку:
Кириянова Екатерина Андреевна

```

Рис. 4.19: Запуск программы

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 ; 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

```

Создаю копию файла lab5-2.asm с именем lab5-222.asm с помощью клавиши F5 (рис. 4.20).

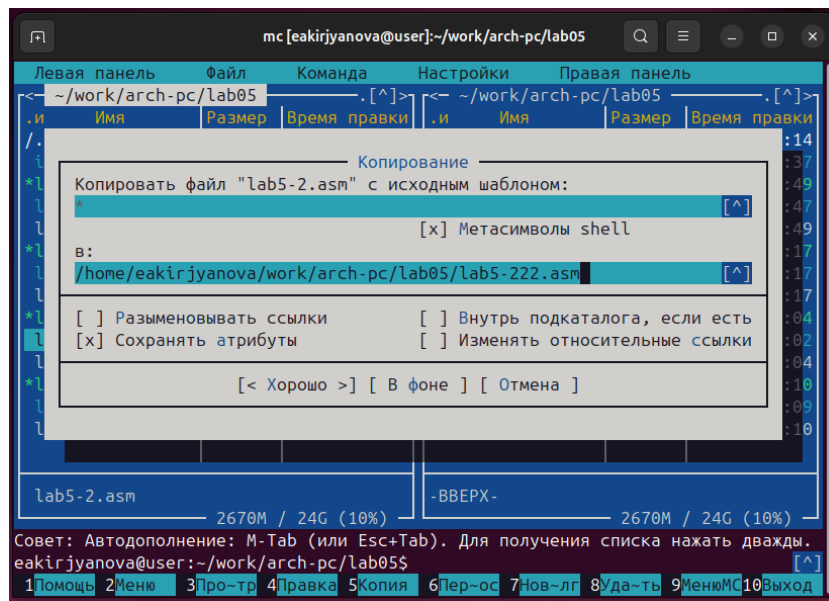


Рис. 4.20: Копирование файла

Изменяю программу, чтобы кроме запроса ввода она выводила заданную строку (рис. 4.21).

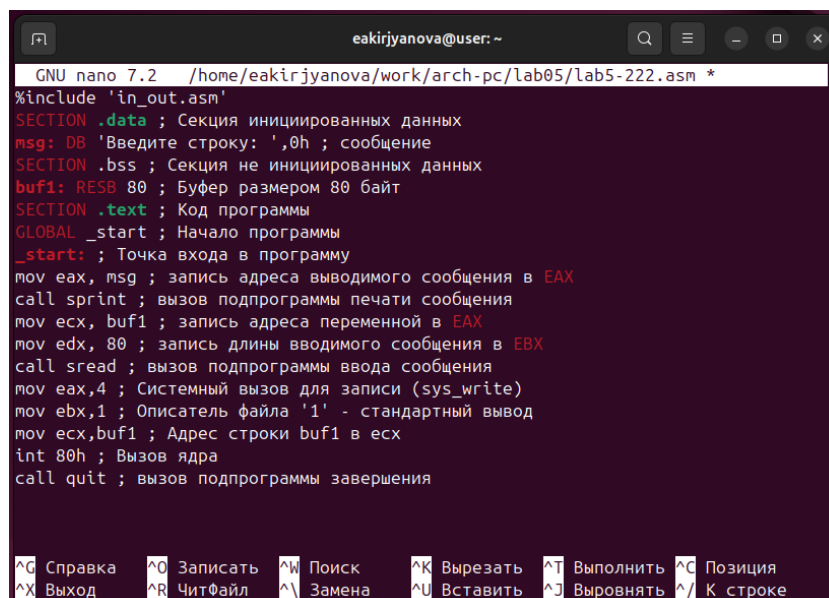
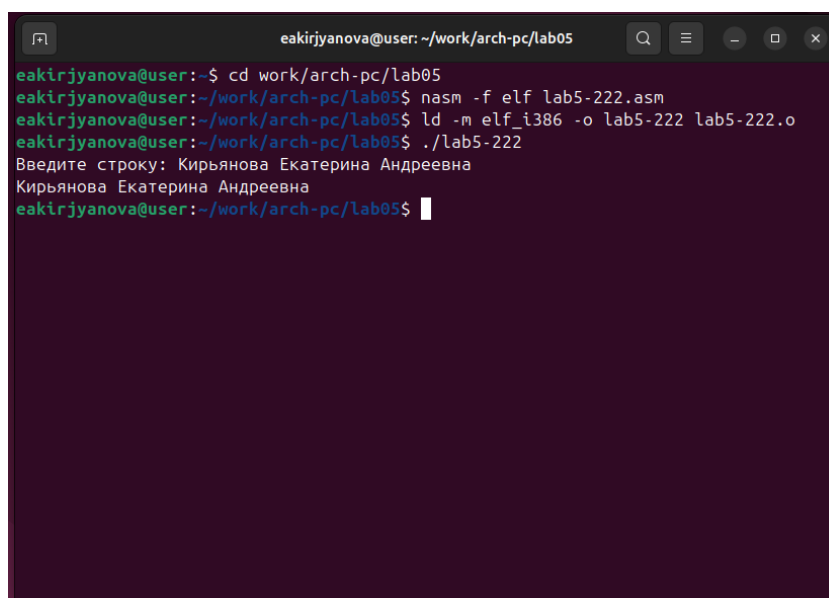


Рис. 4.21: Редактирование файла

Транслирую текст программы в объектный файл, выполняю компоновку объектного файла и запускаю исполняемый файл. Программа выводит заданную

мною строку (рис. 4.22).



```
eakirjyanova@user: ~/work/arch-pc/lab05
eakirjyanova@user:~$ cd work/arch-pc/lab05
eakirjyanova@user:~/work/arch-pc/lab05$ nasm -f elf lab5-222.asm
eakirjyanova@user:~/work/arch-pc/lab05$ ld -m elf_i386 -o lab5-222 lab5-222.o
eakirjyanova@user:~/work/arch-pc/lab05$ ./lab5-222
Введите строку: Кирьянова Екатерина Андреевна
Кирьянова Екатерина Андреевна
eakirjyanova@user:~/work/arch-pc/lab05$
```

Рис. 4.22: Запуск программы

Код программы из пункта 2:

```
%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 ; *Описатель файла '1' - стандартный вывод*

mov ecx,buf1 ; *Адрес строки buf1 в есх*

int 80h ; *Вызов ядра*

call quit ; *вызов подпрограммы завершения*

5 Выводы

В ходе выполнения данной лабораторной работы я приобрела практические навыки работы в Midnight Commander и освоила инструкции языка ассемблера `mov` и `int`.

6 Список литературы

1. Лабораторная работа №5