

Отчет по лабораторной работе №5

Дисциплина: архитектура компьютера

Закиров Нурислам Дамирович

Содержание

1	Цель работы	4
2	Задание	5
3	Теоретическое введение	6
4	Выполнение лабораторной работы	8
4.1	Основы работы с тс	8
4.2	Структура программы на языке ассемблера NASM	10
4.3	Подключение внешнего файла	13
4.4	Выполнение заданий для самостоятельной работы	16
5	Выводы	21
6	Список литературы	22

Список иллюстраций

4.1	Открытый тс	8
4.2	Перемещение между директориями	9
4.3	Создание каталога	9
4.4	Перемещение между директориями	9
4.5	Создание файла	10
4.6	Открытие файла для редактирования	10
4.7	Редактирование файла	11
4.8	Открытие файла для просмотра	12
4.9	Компиляция файла и передача на обработку компоновщику	12
4.10	Исполнение файла	12
4.11	Скачанный файл	13
4.12	Копирование файла	13
4.13	Копирование файла	14
4.14	Редактирование файла	14
4.15	Исполнение файла	15
4.16	Отредактированный файл	15
4.17	Исполнение файла	16
4.18	Копирование файла	16
4.19	Редактирование файла	17
4.20	Исполнение файла	17
4.21	Копирование файла	19
4.22	Редактирование файла	19
4.23	Исполнение файла	20

1 Цель работы

Цель данной лабораторной работы состоит в том, чтобы приобрести практические навыки работы в Midnight Commander и освоить использование инструкции языка ассемблера `mov` и `int`.

2 Задание

1. Основы работы с тс
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).

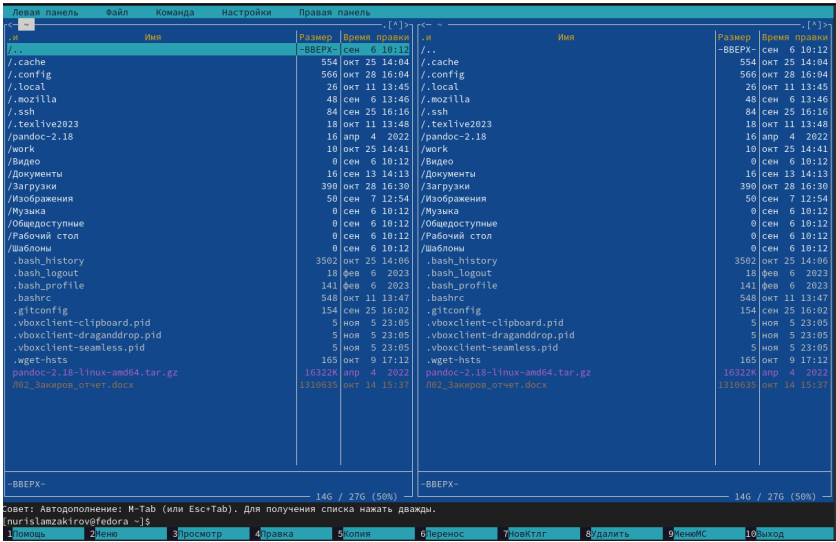


Рис. 4.1: Открытый mc

Используя файловый менеджер mc, перехожу в каталог ~/work/study/2023-2024/Архитектура Компьютера/arch-ps. (рис. -4.2)

Левая панель	Файл	Команда	Настройки	Правая панель	Левая панель	Файл	Команда	Настройки	Правая панель
~	work/study/2023-2024/Архитектура компьютера/arch-pc				~	work/study/2023-2024/Архитектура компьютера/arch-pc			
..					..				
.git					.git				
/config					/config				
/lab04					/lab04				
/labs					/labs				
/presentation					/presentation				
/template					/template				
.gitattributes					.gitattributes				
.gitignore					.gitignore				
.gitmodules					.gitmodules				
CHANGELOG.md					CHANGELOG.md				
COURSE					COURSE				
LICENSE					LICENSE				
Makefile					Makefile				
README_en.md					README_en.md				
README_git-flow.md					README_git-flow.md				
README.md					README.md				
prepare					prepare				

Рис. 4.2: Перемещение между директориями

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

Создать новый каталог

Введите имя каталога:

lab05

[^]

[< Хорошо >] [Отмена]

Рис. 4.3: Создание каталога

Перехожу в созданный каталог (рис. -4.4).

Левая панель	Файл	Команда	Настройки	Правая панель
~	work/study/2023-2024/Архитектура компьютера/arch-pc/lab05			
..				
..				

Рис. 4.4: Перемещение между директориями

Я прописываю команду touch lab5-1.asm в строке ввода, чтобы создать файл, в котором буду работать. (рис. -4.5).

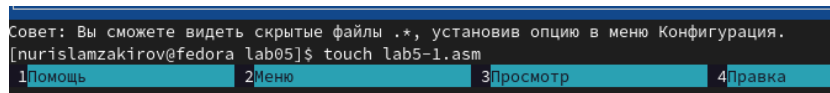


Рис. 4.5: Создание файла

4.2 Структура программы на языке ассемблера NASM

Открываю созданный файл для редактирования в редакторе с помощью функциональной клавиши F4. (рис. -4.6).



Рис. 4.6: Открытие файла для редактирования

Ввожу в файл код программы для запроса строки у пользователя (рис. -4.7). Далее выхожу из файла, используя комбинацию Ctrl+X, сохраняя изменения, используя комбинацию Y, Enter.

```

Файл  Машина  Вид  Ввод  Устройства  Справка
lab5-1.asm      [----]  0 L: [ 1+ 0  1/ 30] *(0  /2024b) 0083 0x053
SECTION .data ; Секция инициализированных данных.
msg: DB 'Введите строку:',10 ; сообщение плюс ; символ перевода строки
msgLen: EQU $-msg ; Длина переменной.
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
;----- 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 ;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

```

Рис. 4.7: Редактирование файла

Открываю файл с помощью функциональной клавиши F3, чтобы увидеть, содержит ли он текст программы. (рис. -4.8).

```

/home/nurislamzakirov/work/study/2023-2024/Архитектура компьютера/arch-pc/lab05/lab5-1.asm
SECTION .data ; Секция инициализированных данных
msg: DB 'Введите строку:',10 ; сообщение плюс ; символ перевода строки
msgLen: EQU $-msg ; Длина переменной
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
;----- 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

```

Рис. 4.8: Открытие файла для просмотра

Командой `nasm -f elf lab5-1.asm` транслирую текст программы файла в объектный файл. Я использую команду `ld -m elf_i386 -o lab5-1 lab5-1.o`, чтобы выполнить компоновку объектного файла. (рис. -4.9). Создался исполняемый файл `lab5-1`.

```

[nurislamzakirov@fedora lab05]$ nasm -f elf lab5-1.asm
[nurislamzakirov@fedora lab05]$ ld -m elf_i386 -o lab5-1 lab5-1.o

```

Рис. 4.9: Компиляция файла и передача на обработку компоновщику

Программа выдает строку «Введите строку» и ожидает ввода с клавиатуры; я ввожу свои ФИО, и программа завершает свою работу. (рис. -4.10).

```

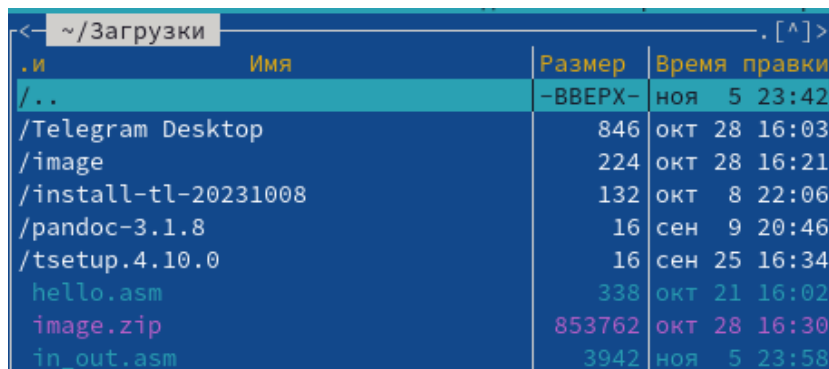
[nurislamzakirov@fedora lab05]$ ./lab5-1
Введите строку:
Закиров Нурислам Дамирович

```

Рис. 4.10: Исполнение файла

4.3 Подключение внешнего файла

Скачиваю файл `in_out.asm` со страницы курса в ТУИС. Он сохранился в каталог “Загрузки” (рис. -4.11).



.и	Имя	Размер	Время правки
./..	-ВВЕРХ-		ноя 5 23:42
/Telegram Desktop		846	окт 28 16:03
/image		224	окт 28 16:21
/install-tl-20231008		132	окт 8 22:06
/pandoc-3.1.8		16	сен 9 20:46
/tsetup.4.10.0		16	сен 25 16:34
hello.asm		338	окт 21 16:02
image.zip		853762	окт 28 16:30
in_out.asm		3942	ноя 5 23:58

Рис. 4.11: Скачанный файл

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

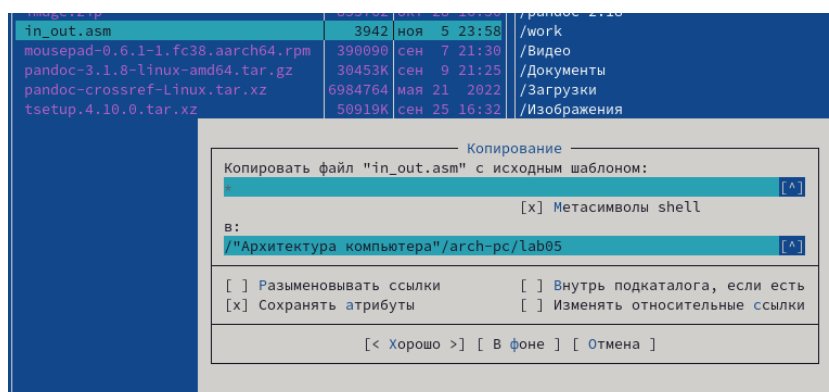


Рис. 4.12: Копирование файла

Копирую файл `lab5-1` в тот же каталог под другим именем с помощью функции F5. Это проделываю в окне `mc`. (рис. -4.13).

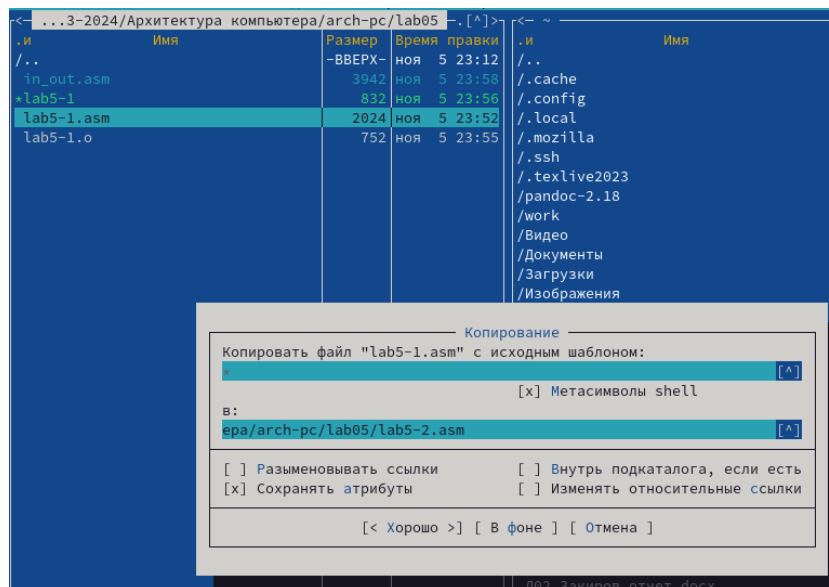


Рис. 4.13: Копирование файла

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

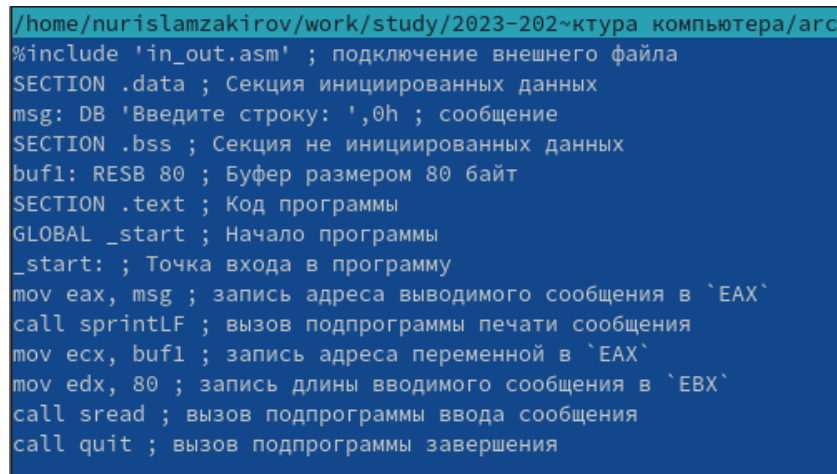


Рис. 4.14: Редактирование файла

Транслирую текст программы файла в объектный файл командой `nasm -f elf lab5-2.asm`. Я использую команду `ld -m elf_i386 -o lab5-2 lab5-2`, чтобы выполнить

компоновку объектного файла. Файл lab5-2 для выполнения. Запускаю исполняемый файл. (рис. -4.15).

```
[nurislamzakirov@fedora lab05]$ nasm -f elf lab5-2.asm
[nurislamzakirov@fedora lab05]$ ld -m elf_i386 -o lab5-2 lab5-2.o
[nurislamzakirov@fedora lab05]$ ./lab6-2
bash: ./lab6-2: Нет такого файла или каталога
[nurislamzakirov@fedora lab05]$ ./lab5-2
Введите строку:
Закиров Нурислам Дамирович
```

Рис. 4.15: Исполнение файла

Открываю файл lab5-2.asm, чтобы изменить его на nano функциональной клавише F4. В нем изменил подпрограмму sprint LF на sprint. Я сохраняю изменения и открываю файл для проверки сохранения. (рис. -4.16).

```
lab5-2.asm      [-M--] 11 L:[ 1+ 9 10/ 15] *(586 / 962b
%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 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

Я сначала транслирую файл, затем выполняю компоновку объектного файла, а затем запускаю новый исполняемый файл. (рис. -4.17).

```
[nurislamzakirov@fedora lab05]$ nasm -f elf lab5-2.asm
[nurislamzakirov@fedora lab05]$ ld -m elf_i386 -o lab5-2 lab5-2.o
[nurislamzakirov@fedora lab05]$ ./lab5-2
Введите строку: Закиров Нурислам Дамирович
```

Рис. 4.17: Исполнение 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 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 lab5-1.asm с именем lab5-1-1.asm с помощью функциональной клавиши F5 (рис. -4.18).

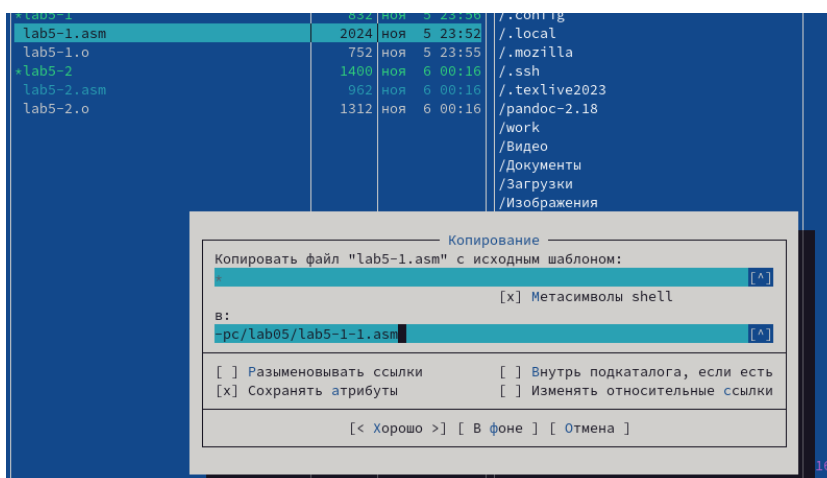


Рис. 4.18: Копирование файла

Открываю созданный файл для редактирования с помощью функциональной клавиши F4. Я изменил программу, чтобы она выдавала строку, вводимую пользователем, помимо приглашения и запроса ввода. (рис. -4.19).


```

lab5-1-1.asm      [-M--]  0 L:[ 1+34 35/ 35] *(2078/2078b) <EOF>
SECTION .data ; Секция иницированных данных.
msg: DB 'Введите строку:',10 ; сообщение плюс ; символ перевода строки
msgLen: EQU $-msg ; Длина переменной.
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
;----- 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
mov eax,4
mov ebx,1
mov ecx,buf1
mov edx,buf1
int 80h
;----- 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. 4.19: Редактирование файла

2. Я создаю проектный файл lab5-1-1.o, передаю его компоновщику для обработки, получаю исполняемый файл lab5-1-1 и запускаю полученный исполняемый файл. Программа запрашивает ввод данных, а затем выводит мои данные. (рис. -4.20).

```

[nurislamzakirov@fedora lab05]$ nasm -f elf lab5-1-1.asm
[nurislamzakirov@fedora lab05]$ ld -m elf_i386 -o lab5-1-1 lab5-1-1.o
[nurislamzakirov@fedora lab05]$ ./lab5-1-1
Введите строку:
Закиров Нурислам Дамирович
Закиров Нурислам Дамирович
[nurislamzakirov@fedora lab05]$

```

Рис. 4.20: 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 ;
mov ebx,1 ;
mov ecx,buf1 ;
mov edx,buf1 ;
int 80h ;
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 lab5-2.asm с именем lab5-2-1.asm с помощью функциональной клавиши F5 (рис. -4.21).

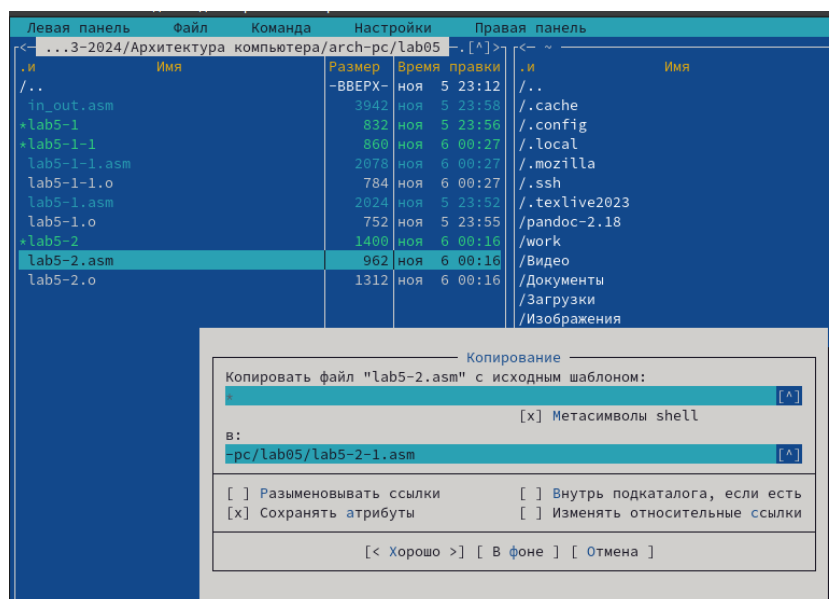


Рис. 4.21: Копирование файла

Открываю созданный файл для редактирования с помощью функциональной клавиши F4. Я изменил программу, чтобы она выдавала строку, вводимую пользователем, помимо приглашения и запроса ввода. (рис. -4.22).

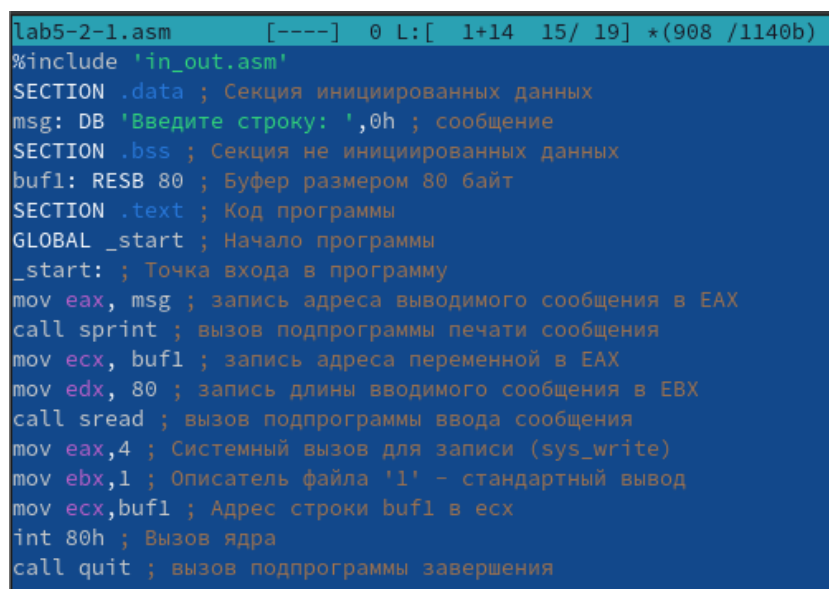


Рис. 4.22: Редактирование файла

4. Я создаю проектный файл lab5-2-1, отдаю его компоновщику, получаю исполняемый файл lab5-2-1, а затем запускаю полученный файл. Программа запрашивает ввод без переноса данных на новую строку. Затем она вводит мои ФИО и выводит введенные мною данные. (рис. -4.23).

```
[nurislamzakirov@fedora lab05]$ nasm -f elf lab5-2-1.asm
[nurislamzakirov@fedora lab05]$ ld -m elf_i386 -o lab5-2-1 lab5-2-1.o
[nurislamzakirov@fedora lab05]$ ./lab5-2-1
Введите строку: Закиров Нурислам Дамирович
Закиров Нурислам Дамирович
[nurislamzakirov@fedora lab05]$
```

Рис. 4.23: 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
```

5 Выводы

При выполнении данной лабораторной работы я приобрел практические навыки работы в Midnight Commander и освоил инструкции языка ассемблера `mov` и `int`.

6 Список литературы

1. Лабораторная работа №5