

Отчет по лабораторной работе №5

Дисциплина: архитектура компьютера

Карачевцева Елизавета Васильевна

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	12
4.4	Выполнение заданий для самостоятельной работы	14
5	Выводы	18
6	Список литературы	19

Список иллюстраций

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

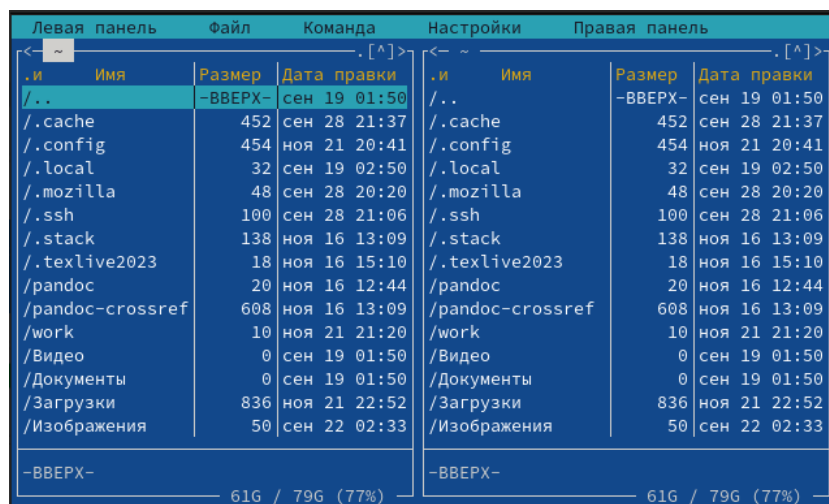


Рис. 4.1: Открытый mc

Перехожу в каталог ~/work/study/2022-2023/Архитектура Компьютера/arch-рс, используя файловый менеджер mc.

Левая панель		Файл	Команда	Настройки	Правая панель		
< ~ ...ктура компьютера/arch-pc -.[^]>				< ~ -.[^]>			
.и	Имя	Размер	Дата правки	.и	Имя	Размер	Дата правки
/..		-ВВЕРХ-	ноя 16 11:47	/..		-ВВЕРХ-	сен 19 01:50
/.git		218	ноя 21 22:57	/.cache	452	сен 28 21:37	
/config		24	ноя 16 11:47	/.config	454	ноя 21 20:41	
/lab04		18	ноя 22 00:22	/.local	32	сен 19 02:50	
/labs		152	ноя 16 12:44	/.mozilla	48	сен 28 20:20	
/presentation		78	ноя 16 12:44	/.ssh	100	сен 28 21:06	
/template		36	ноя 16 11:47	/.stack	138	ноя 16 13:09	
.gitattributes		1765	ноя 16 11:47	/.texlive2023	18	ноя 16 15:10	
.gitignore		4637	ноя 16 11:47	/pandoc	20	ноя 16 12:44	
.gitmodules		278	ноя 16 11:47	/pandoc-crossref	608	ноя 16 13:09	
CHANGELOG.md		4786	ноя 16 11:47	/work	10	ноя 21 21:20	
COURSE		8	ноя 16 11:49	/Видео	0	сен 19 01:50	
LICENSE		18657	ноя 16 11:47	/Документы	0	сен 19 01:50	
Makefile		980	ноя 16 11:47	/Загрузки	836	ноя 21 22:52	
README.en.md		152	ноя 16 11:47	/Изображения	50	сен 22 02:33	
-ВВЕРХ-				-ВВЕРХ-			
61G / 79G (77%)				61G / 79G (77%)			

Рис. 4.2: Перемещение между директориями

С помощью функциональной клавиши F7 создаю каталог lab05. Перехожу в созданный каталог.

Левая панель				Настройки		Правая панель	
Файл		Команда		-.[^]>			
<- ...компьютера/arch-pc/lab05				<- ~			
.и	Имя	Размер	Дата правки	.и	Имя	Размер	Дата правки
/..		-ВВЕРХ-	ноя 22 00:23	/..		-ВВЕРХ-	сен 19 01:50
				/.cache		452	сен 28 21:37
				/.config		454	ноя 21 20:41
				/.local		32	сен 19 02:50
				/.mozilla		48	сен 28 20:20
				/.ssh		100	сен 28 21:06
				/.stack		138	ноя 16 13:09
				/.texlive2023		18	ноя 16 15:10
				/pandoc		20	ноя 16 12:44
				/pandoc-crossref		608	ноя 16 13:09
				/work		10	ноя 21 21:20
				/Видео		0	сен 19 01:50
				/Документы		0	сен 19 01:50
				/Загрузки		836	ноя 21 22:52
				/Изображения		50	сен 22 02:33
-ВВЕРХ-				-ВВЕРХ-			
61G / 79G (77%)				61G / 79G (77%)			

Рис. 4.3: Перемещение между директориями

В строке ввода прописываю команду touch lab5-1.asm, чтобы создать файл, в котором буду работать.

```
touch lab5-1.asm
```

Рис. 4.4: Создание файла

4.2 Структура программы на языке ассемблера NASM

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

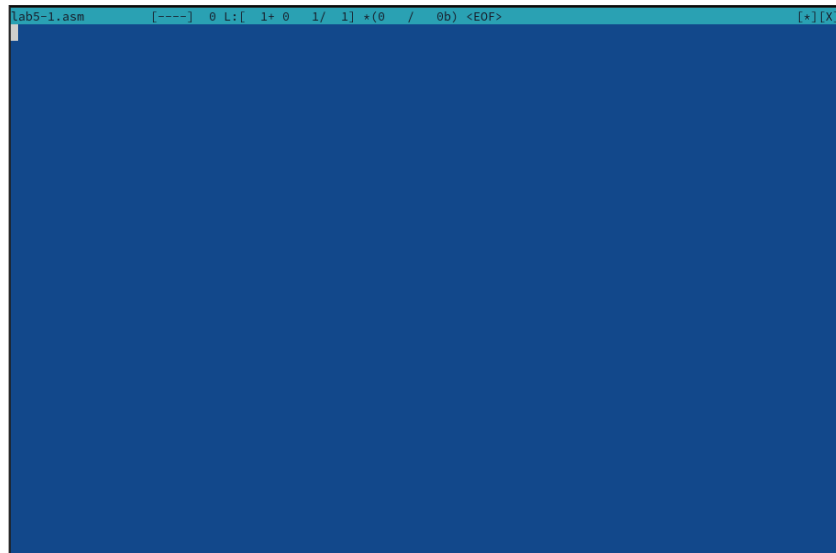


Рис. 4.5: Редактирование файла

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

```

lab5-1.asm      [-M--] 20 L:[ 1+20 21/ 21] *(1236/1236b) <EOF>
SECTION .data ; Секция инициорванных 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,1 ; Системный вызов для выхода (sys_exit)
mov ebx,0 ; Выход с кодом возврата 0 (без ошибок)
int 80h ; Вызов ядра

```

Рис. 4.6: 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`. Создался исполняемый файл `lab5-1`.

```

/home/evkarachevtseva/work/study/2024-2-ypa_компьютера/arch-pc/lab05/lab5-1.asm      1236/1236      100%
SECTION .data ; Секция инициорванных 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,1 ; Системный вызов для выхода (sys_exit)
mov ebx,0 ; Выход с кодом возврата 0 (без ошибок)
int 80h ; Вызов ядра

```

Рис. 4.7: 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.

```
evkarachevtseva@fedora: ~/work/study/2024-2025/Архитектура компьютера/arch-pc/lab05$ nasm -f elf lab5-1.asm
evkarachevtseva@fedora: ~/work/study/2024-2025/Архитектура компьютера/arch-pc/lab05$ ld -m elf_1386 -o lab5-1 lab5-1.o
```

Рис. 4.8: Исполнение файла

4.3 Подключение внешнего файла

Скачиваю файл in_out.asm со страницы курса в ТУИС. Он сохранился в каталог “Загрузки”.

```
evkarachevtseva@fedora: ~/work/study/2024-2025/Архитектура компьютера/arch-pc/lab05$ ./lab5-1
Введите строку:
Карачевцева Елизавета Васильевна
```

Рис. 4.9: Скачанный файл

С помощью функциональной клавиши F5 копирую файл in_out.asm из каталога Загрузки в созданный каталог lab05.

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.

Изменяю содержимое файла lab5-2.asm во встроенном редакторе, чтобы в программе использовались подпрограммы из внешнего файла in_out.asm.

```
lab5-2.asm [-M--] 41 L: [ 1+17 18/ 18] *(1145/1145b) <EOF> [*][X]
#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 ecx
int 80h ; Вызов ядра
call quit ; вызов подпрограммы завершения
```

Рис. 4.10: 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.

```
/home/evkarachevtseva/work/study/2024-2025/Архитектура компьютера/arch-pc/lab05/lab5-2.asm
#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 sprintf ; вызов подпрограммы печати сообщения
mov ecx, buf1 ; запись адреса переменной в 'ECX'
mov edx, 80 ; запись длины вводимого сообщения в 'EDX'
call sread ; вызов подпрограммы ввода сообщения
call quit ; вызов подпрограммы завершения
```

Рис. 4.11: Исполнение файла

Открываю файл lab5-2.asm для редактирования функциональной клавишей F4. Изменяю в нем подпрограмму `sprintf` на `sprint`. Сохраняю изменения и открываю файл для просмотра, чтобы проверить сохранение действий).

```
evkarachevtseva@fedora: ~/work/study/2024-2025/Архитектура компьютера/arch-pc/lab05$ nasm -f elf lab5-2.asm
evkarachevtseva@fedora: ~/work/study/2024-2025/Архитектура компьютера/arch-pc/lab05$ ld -m elf_i386 -o lab5-2 lab5-2.o
evkarachevtseva@fedora: ~/work/study/2024-2025/Архитектура компьютера/arch-pc/lab05$ ./lab5-2
Введите строку:
Карачевцева Елизавета Васильевна
evkarachevtseva@fedora: ~/work/study/2024-2025/Архитектура компьютера/arch-pc/lab05$
```

Рис. 4.12: Отредактированный файл

Снова транслирую файл, выполняю компоновку созданного объектного файла, запускаю новый исполняемый файл.

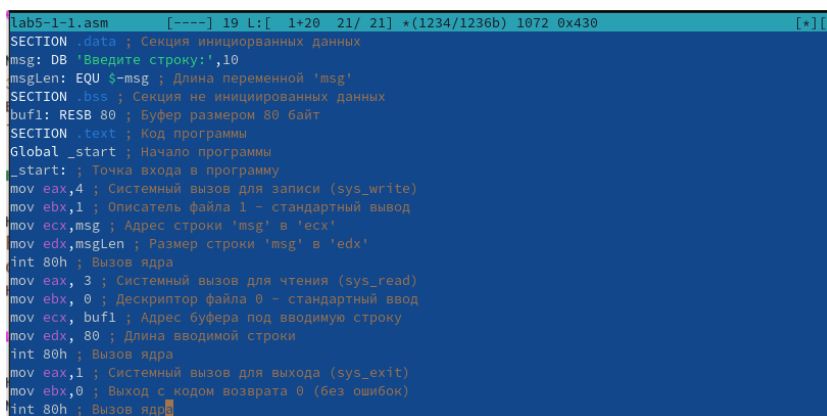
```
evkarachevtseva@fedora: ~/work/study/2024-2025/Архитектура компьютера/arch-pc/lab05$ nasm -f elf lab5-2.asm
evkarachevtseva@fedora: ~/work/study/2024-2025/Архитектура компьютера/arch-pc/lab05$ ld -m elf_i386 -o lab5-2-2 lab5-2.o
evkarachevtseva@fedora: ~/work/study/2024-2025/Архитектура компьютера/arch-pc/lab05$ ./lab5-2-2
Введите строку: Карачевцева Елизавета Васильевна
evkarachevtseva@fedora: ~/work/study/2024-2025/Архитектура компьютера/arch-pc/lab05$
```

Рис. 4.13: 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 `sprintf` и `sprint`.

4.4 Выполнение заданий для самостоятельной работы

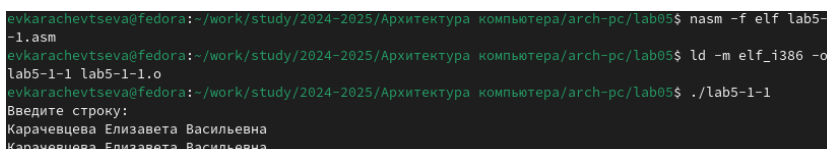
1. Создаю копию файла lab5-1.asm с именем lab5-1-1.asm с помощью функциональной клавиши F5. С помощью функциональной клавиши F4 открываю созданный файл для редактирования. Изменяю программу так, чтобы кроме вывода приглашения и запроса ввода, она выводила вводимую пользователем строку.



```
lab5-1-1.asm [----] 19 L: [ 1+20 21/ 21] *(1234/1236b) 1072 0x430 [*][X]
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 ; Описание файла 0 - стандартный ввод
mov ecx,buf1 ; Адрес буфера под вводимую строку
mov edx,80 ; Длина вводимой строки
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
```

Рис. 4.14: 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.



```
evkarachevtseva@fedora:~/work/study/2024-2025/Архитектура компьютера/arch-pc/lab05$ nasm -f elf lab5-1-1.asm
evkarachevtseva@fedora:~/work/study/2024-2025/Архитектура компьютера/arch-pc/lab05$ ld -m elf_i386 -o lab5-1-1 lab5-1-1.o
evkarachevtseva@fedora:~/work/study/2024-2025/Архитектура компьютера/arch-pc/lab05$ ./lab5-1-1
Введите строку:
Карачевцева Елизавета Васильевна
```

Рис. 4.15: 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 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. Создаю копию файла lab5-2.asm с именем lab5-2-1.asm с помощью функциональной клавиши F5.

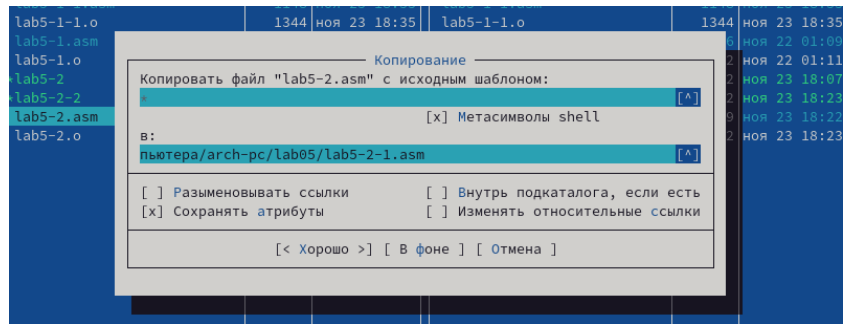


Рис. 4.16: Копирование файла

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.

```
/home/evkarachevtseva/work/study/2024-2025/Архитектура компьютера/arch-pc/lab05/lab5-2-1.asm
#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 ecx
int 80h ; Вызов ядра
call quit ; вызов подпрограммы завершения
```

Рис. 4.17: 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.

```
evkarachevtseva@fedora:~/work/study/2024-2025/Архитектура компьютера/arch-pc/lab05$ nasm -f elf lab5-2-1.asm
evkarachevtseva@fedora:~/work/study/2024-2025/Архитектура компьютера/arch-pc/lab05$ ld -m elf_i386 -o lab5-2-1 lab5-2-1.o
evkarachevtseva@fedora:~/work/study/2024-2025/Архитектура компьютера/arch-pc/lab05$ ./lab5-2-1
Введите строку: Карачевцева Елизавета Васильевна
Карачевцева Елизавета Васильевна
```

Рис. 4.18: 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 приобрела практические навыки работы в Midnight Commander, а также освоила инструкции языка ассемблера `mov` и `int`.

6 Список литературы

1. Лабораторная работа №5