

Отчет по лабораторной работе №5

Дисциплина: архитектура компьютера

Цоппа Ева Эдуардовна

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	22
6	Список литературы	23

Список иллюстраций

4.1 Открытый тс.....	Error! Bookmark not defined.
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	13
4.10 Исполнение файла	13
4.11 Скачанный файл	13
4.12 Копирование файла	14
4.13 Копирование файла	14
4.14 Редактирование файла.....	15
4.15 Исполнение файла	15
4.16 Отредактированный файл	16
4.17 Исполнение файла	16
4.18 Копирование файла	17
4.19 Редактирование файла.....	17
4.20 Исполнение файла	18
4.21 Копирование файла	19
4.22 Редактирование файла.....	20
4.23 Исполнение файла	20

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 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. 4.1).

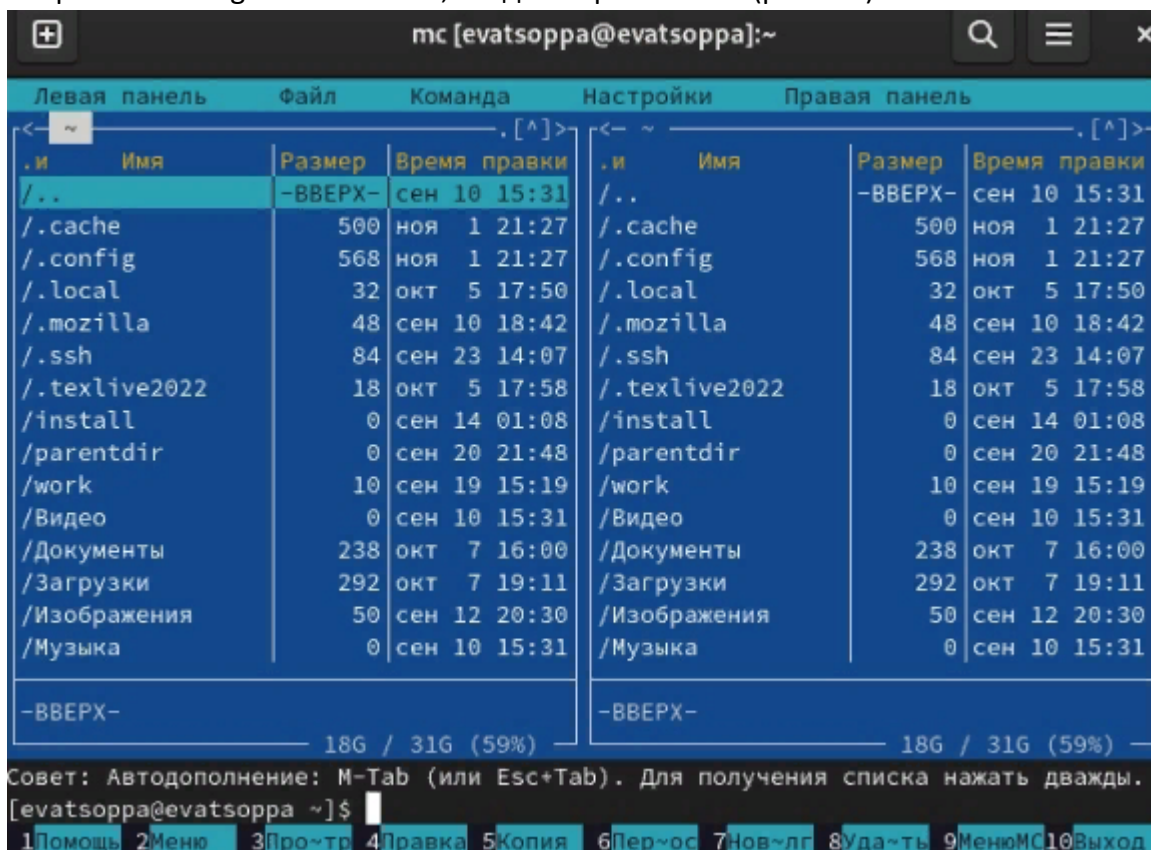


Рис. 4.1: Открытый mc

Перехожу в каталог `~/work/study/2023-2024/Архитектура Компьютера/arch-рс`, используя файловый менеджер mc (рис. 4.2)

/Архитектура компьютера/arch-pc			.[^]>			.[^]>		
Размер	Время	Правка	Имя	Размер	Время	Правка	Имя	Размер
-BBERX-	окт 4 03:03		/..	-BBERX-	сен 12 19:12			
4096	ноя 7 19:08		/.cache	4096	ноя 7 19:11			
4096	окт 4 03:03		/.config	4096	сен 12 20:41			
4096	ноя 7 15:57		/.gnupg	4096	сен 12 19:14			
4096	окт 4 03:06		/.java	4096	сен 12 19:12			
4096	окт 4 03:03		/.local	4096	сен 12 19:14			
1765	окт 4 03:03		/.mozilla	4096	сен 12 19:21			
4637	окт 4 03:03		/.rpmdb	4096	сен 12 20:58			
278	окт 4 03:03		/.ssh	4096	окт 4 03:03			
2126	окт 4 03:03		/.texlive2022	4096	окт 18 02:51			
13	окт 4 03:07		/parentdir	4096	сен 27 12:27			
18657	окт 4 03:03		/work	4096	окт 4 02:50			
815	окт 4 03:03		/Видео	4096	сен 12 19:14			
152	окт 4 03:03		/Документы	4096	сен 12 19:14			
5653	окт 4 03:03		/Загрузки	4096	окт 29 03:36			
4477	окт 4 03:03		/Изображения	4096	окт 29 01:48			
0	окт 4 03:06		/Музыка	4096	сен 12 19:14			
			/Общедоступные	4096	сен 12 19:14			
			/Рабочий стол	4096	сен 12 19:14			
			/Шаблоны	4096	сен 12 19:14			
			./ICEauthority	0	сен 12 19:14			

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

Переходу в созданный каталог (рис. 4.4).

Файл			Команда		
/lab05			.[^]>		
Имя	Размер	Время	Правка		
-BBERX-		ноя 1 21:30			

Рис. 4.4: Перемещение между директориями

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

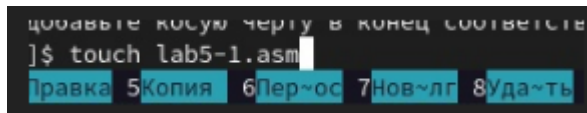


Рис. 4.5: Создание файла

4.2 Структура программы на языке ассемблера NASM

С помощью функциональной клавиши F4 открываю созданный файл для редактирования в редакторе (рис. 4.6).

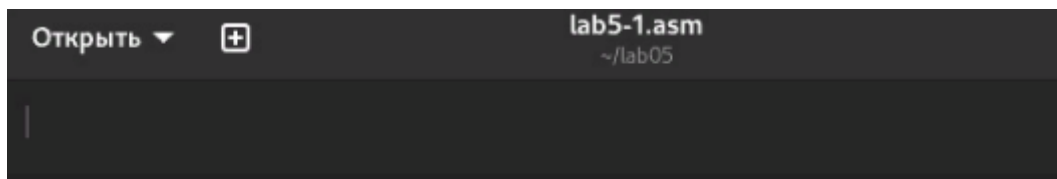
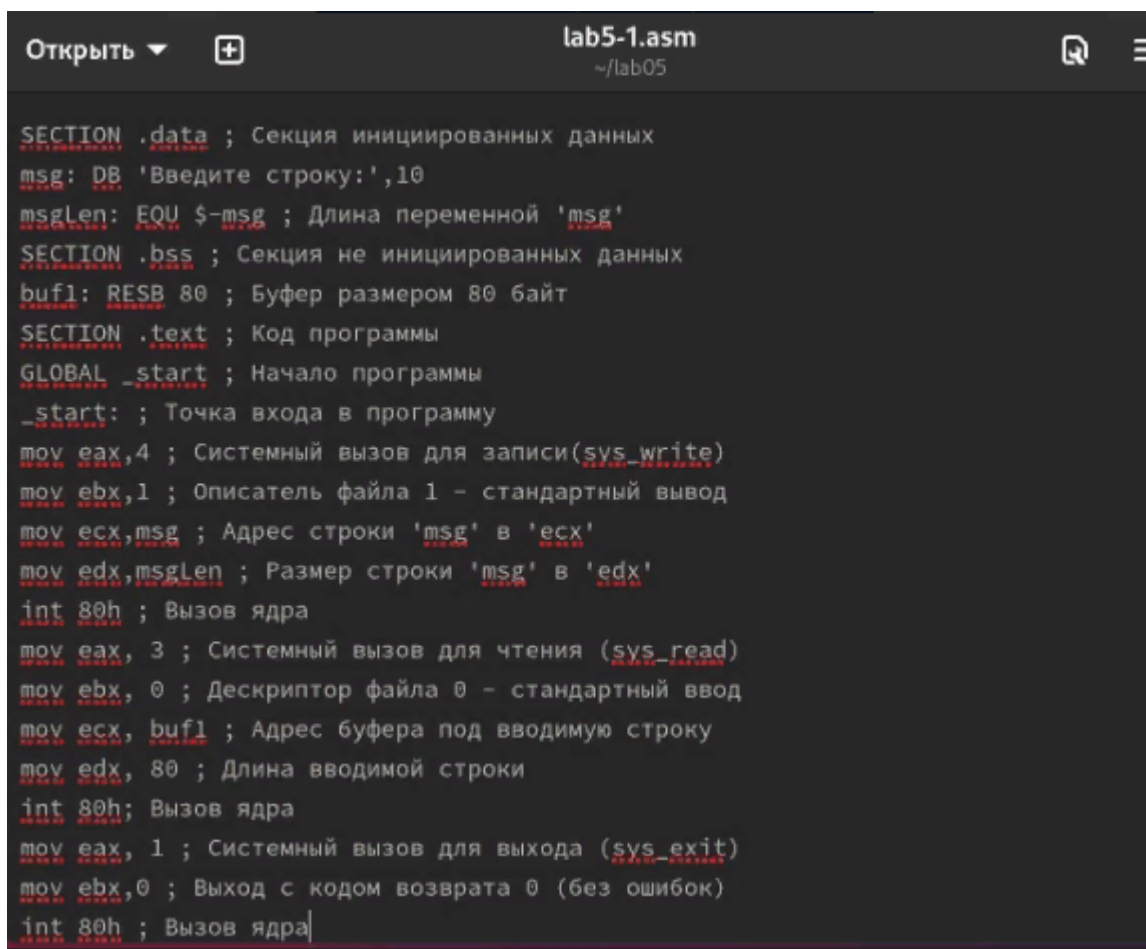


Рис. 4.6: Открытие файла для редактирования

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



```
Открыть ▾ + lab5-1.asm ~/lab05

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, 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

С помощью функциональной клавиши F3 открываю файл для просмотра, чтобы проверить, содержит ли файл текст программы (рис. 4.8).

```
/home/evatsoppa/work/st-ch-pc/lab05/lab5-1.asm 12
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, 1 ; Системный вызов для выхода (sys_exit)
mov ebx,0 ; Выход с кодом возврата 0 (без ошибок)
int 80h ; Вызов ядра
1Помощь 2Раз-ри 3Выход 4Нех 5Пер-ти 6 7Поис
```

Рис. 4.8: 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` (рис. 4.9). Создался исполняемый файл `lab5-1`.

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

Рис. 4.9: 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. 4.10).

```
[evatsoppa@evatsoppa lab05]$ ./lab5-1
Введите строку:
Цоппа Ева Эдуардовна
```

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).

L02_Tsop~het.pdf	1464939	окт 6 20:26	/
in_out.asm	3942	ноя 1 22:26	/
L01_Цопп~чет.pdf	1532696	сен 25 14:40	/

Рис. 4.11: Скачанный файл

С помощью функциональной клавиши F5 копирую файл in_out.asm из каталога Загрузки в созданный каталог lab05 (рис. 4.12).

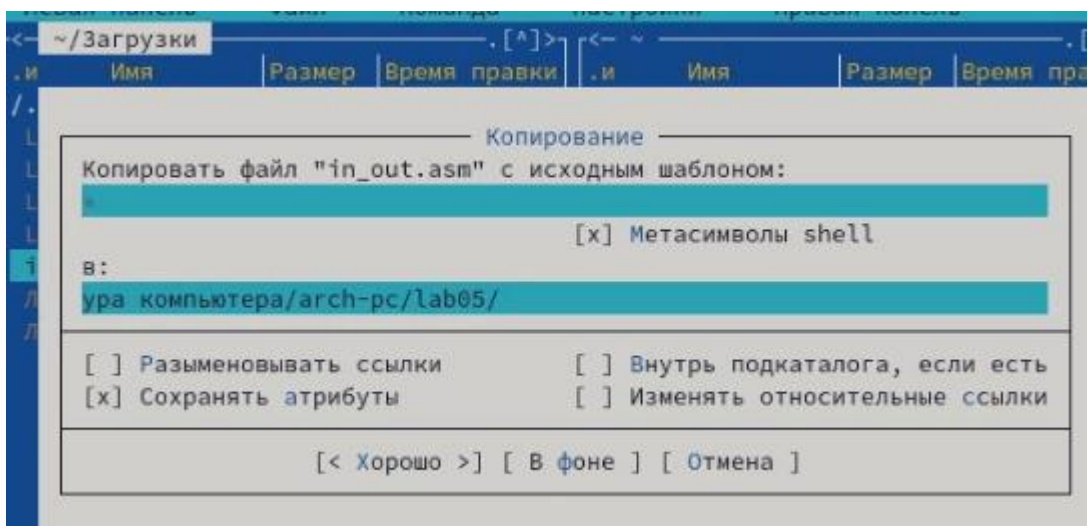


Рис. 4.12: Копирование файла

С помощью функциональной клавиши F5 копирую файл lab5-1 в тот же каталог, но с другим именем, для этого в появившемся окне тс прописываю имя для копии файла (рис. 4.13).

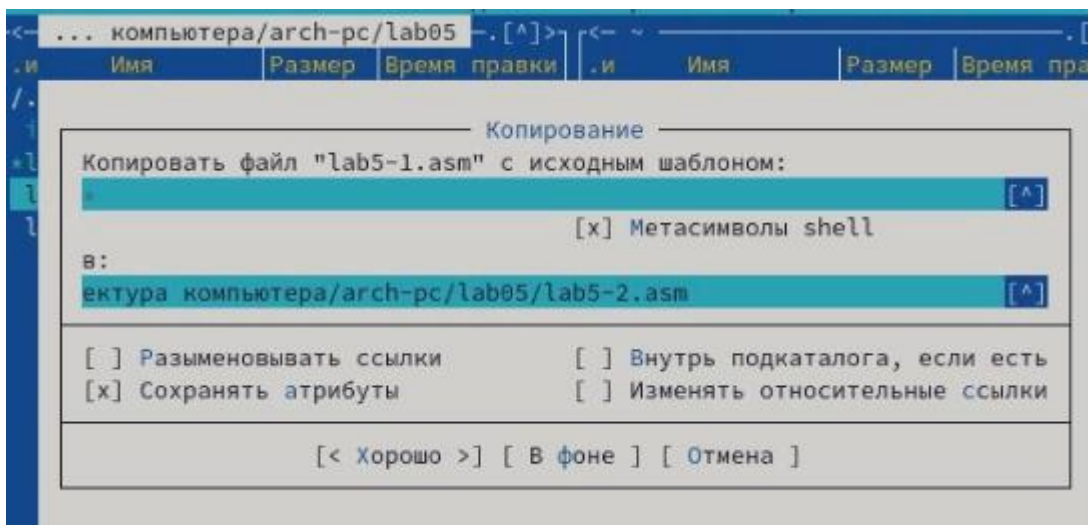
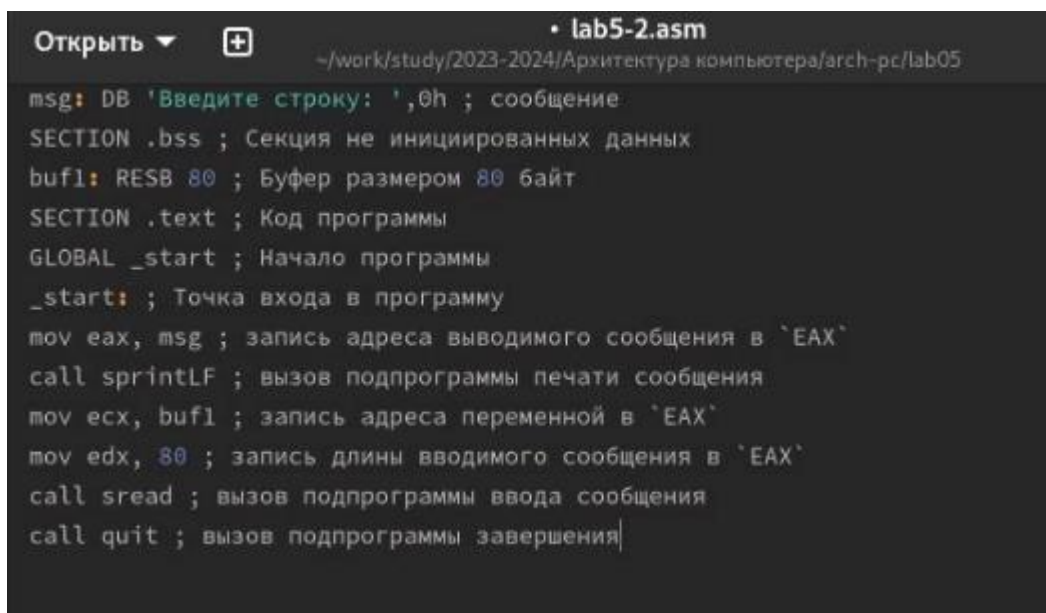


Рис. 4.13: Копирование файла

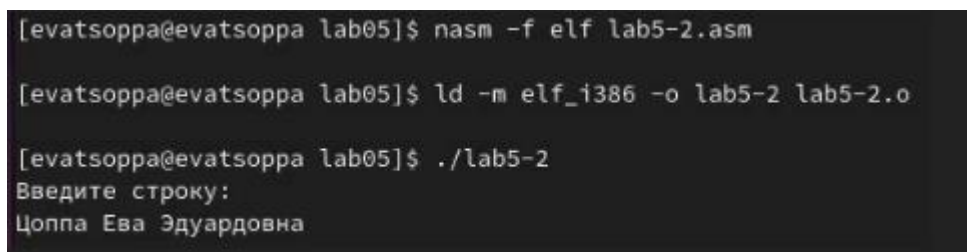
Изменяю содержимое файла lab5-2.asm в редакторе (рис. 4.14), чтобы в программе использовались подпрограммы из внешнего файла in_out.asm.



```
Открыть ▾ + • lab5-2.asm
~/work/study/2023-2024/Архитектура компьютера/arch-pc/lab05
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 `EAX`
mov edx, 80 ; запись длины вводимого сообщения в `EAX`
call sread ; вызов подпрограммы ввода сообщения
call quit ; вызов подпрограммы завершения
```

Рис. 4.14: Редактирование файла

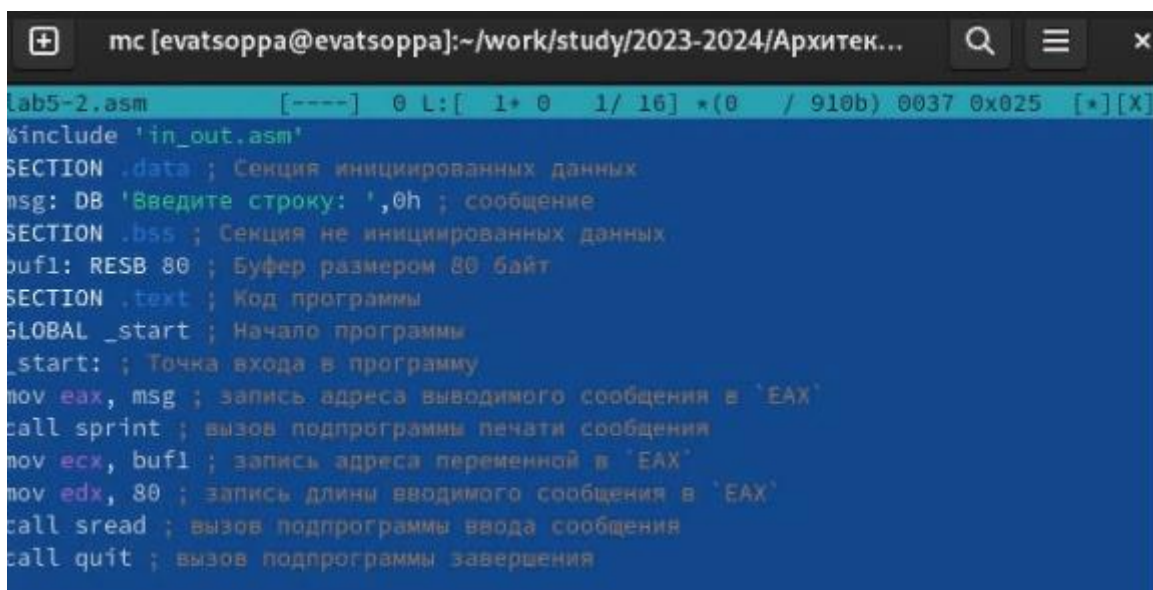
Транслирую текст программы файла в объектный файл командой `nasm -f elf lab5-2.asm`. Создался объектный файл `lab5-2.o`. Выполняю компоновку объектного файла с помощью команды `ld -m elf_i386 -o lab5-2 lab5-2.o`. Создался исполняемый файл `lab5-2`. Запускаю исполняемый файл (рис. 4.15).



```
[evatsoppa@evatsoppa lab05]$ nasm -f elf lab5-2.asm
[evatsoppa@evatsoppa lab05]$ ld -m elf_i386 -o lab5-2 lab5-2.o
[evatsoppa@evatsoppa lab05]$ ./lab5-2
Введите строку:
Цоппа Ева Эдуардовна
```

Рис. 4.15: Исполнение файла

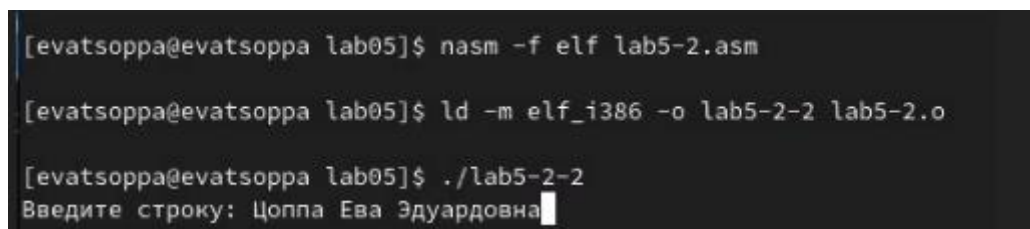
Открываю файл `lab5-2.asm` для редактирования в nano функциональной клавишей F4. Изменяю в нем подпрограмму `sprintf` на `sprint`. Сохраняю изменения и открываю файл для просмотра, чтобы проверить сохранение действий (рис. 4.16).



```
lab5-2.asm [----] 0 L: [ 1+ 0 1/ 16] *(0 / 910b) 0037 0x025 [*][X]
#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 'EAX'
call sread ; вызов подпрограммы ввода сообщения
call quit ; вызов подпрограммы завершения
```

Рис. 4.16: Отредактированный файл

Снова транслирую файл, выполняю компоновку созданного объектного файла, запускаю новый исполняемый файл (рис. 4.17).



```
[evatsoppa@evatsoppa lab05]$ nasm -f elf lab5-2.asm
[evatsoppa@evatsoppa lab05]$ ld -m elf_i386 -o lab5-2-2 lab5-2.o
[evatsoppa@evatsoppa lab05]$ ./lab5-2-2
Введите строку: Цоппа Ева Эдуардовна
```

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).

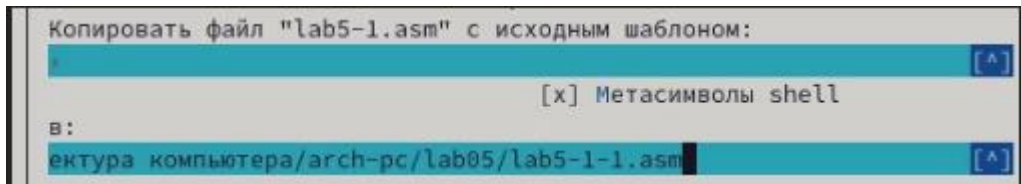


Рис. 4.18: Копирование файла

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. 4.19).

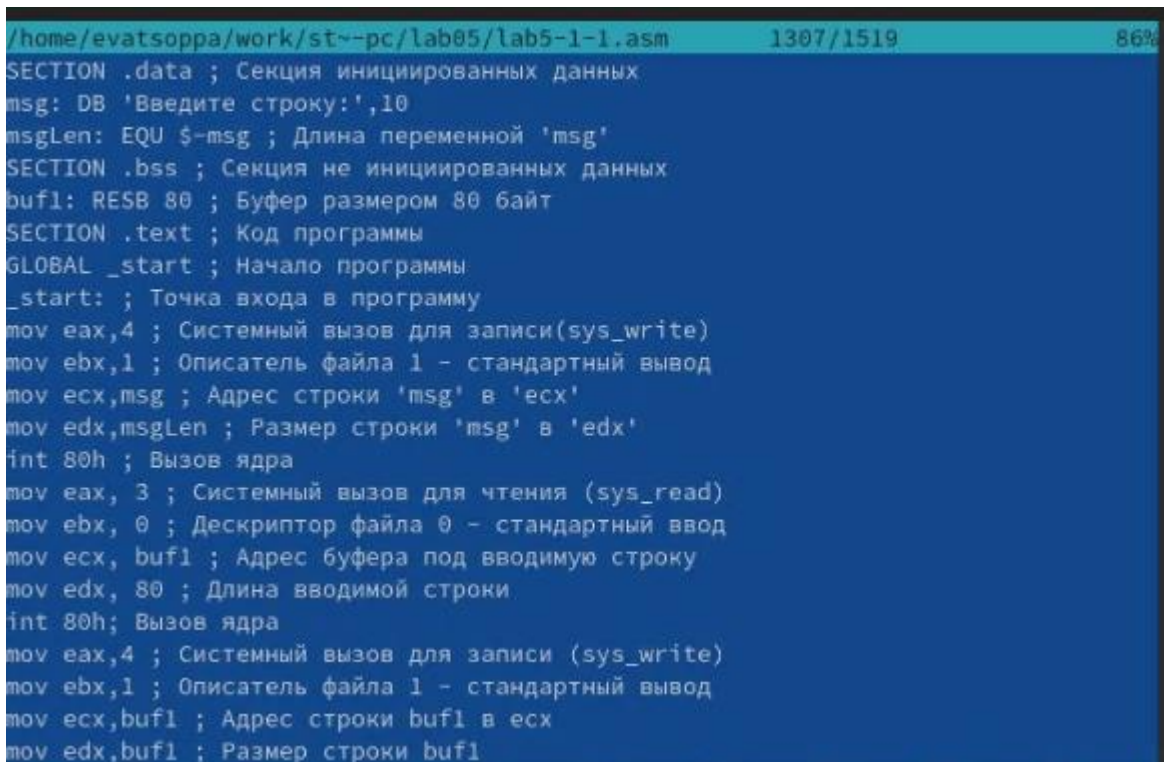


Рис. 4.19: Редактирование файла

2. Создаю объектный файл lab5-1-1.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. 4.20).

```
[evatsoppa@evatsoppa lab05]$ nasm -f elf lab5-1-1.asm
[evatsoppa@evatsoppa lab05]$ ld -m elf_i386 -o lab5-1-1 lab5-1-1.o
[evatsoppa@evatsoppa lab05]$ ./lab5-1-1
Введите строку:
Цоппа Ева Эдуардовна
Цоппа Ева Эдуардовна
```

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 ; Системный вызов для записи (sys_write)

mov ebx,1 ; Описатель файла '1' - стандартный вывод

mov ecx,buf1 ; Адрес строки buf1 в есх

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. 4.21).

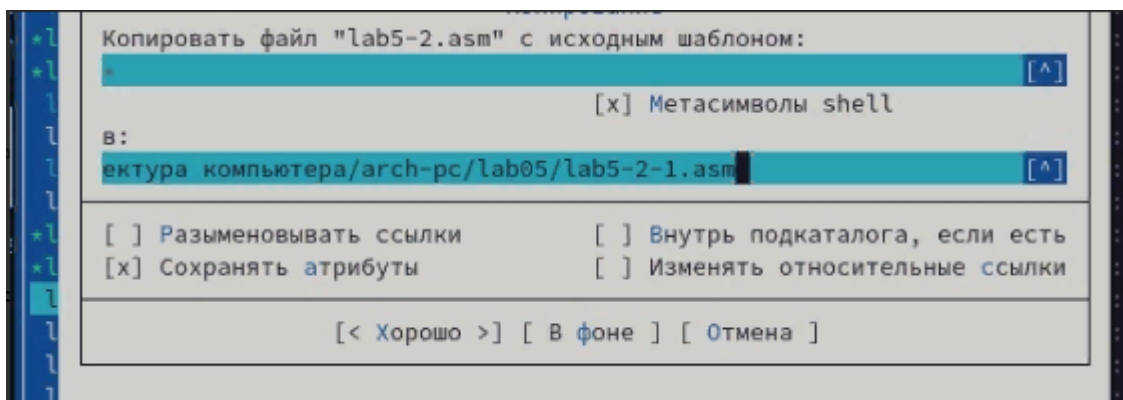


Рис. 4.21: Копирование файла

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. 4.22).

```

/home/evatsoppa/work/st~pc/lab05/lab5-2-1.asm 1146/1146 100%
%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 `EAX`
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. 4.22: 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 (рис. 4.23).

```

[evatsoppa@evatsoppa lab05]$ nasm -f elf lab5-2-1.asm
[evatsoppa@evatsoppa lab05]$ ld -m elf_i386 -o lab5-2-1 lab5-2-1.o
[evatsoppa@evatsoppa lab05]$ ./lab5-2-1
Введите строку: Цоппа Ева Эдуардовна
Цоппа Ева Эдуардовна

```

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

6 Список литературы

1. Лабораторная работа №5