

# **Лабораторная работа №5**

**Дисциплина: архитектура компьютера**

Ларина Наталья Денисовна

# Содержание

|          |   |           |
|----------|---|-----------|
| <b>1</b> | <b>Цель работы</b>                                      | <b>5</b>  |
| <b>2</b> | <b>Задание</b>  | <b>6</b>  |
| <b>3</b> | <b>Теоретическое введение</b>                           | <b>7</b>  |
| <b>4</b> | <b>Выполнение лабораторной работы</b>                   | <b>9</b>  |
| 4.1      | Основы работы в тс . . . . .                            | 9         |
| 4.2      | Структура программы на языке ассемблера NASM . . . . .  | 11        |
| 4.3      | Подключение внешнего файла . . . . .                    | 13        |
| 4.4      | Выполнение заданий для самостоятельной работы . . . . . | 16        |
| <b>5</b> | <b>Выводы</b>   | <b>21</b> |
| <b>6</b> | <b>Список литературы</b>                                | <b>22</b> |

## **Список таблиц**

## Список иллюстраций

|      |  |    |
|------|--|----|
| 4.1  | Открытый тс . . . . .                        | 9  |
| 4.2  | Перемещение между директориями . . . . .     | 10 |
| 4.3  | Создание каталога . . . . .                  | 10 |
| 4.4  | Перемещение между директориями . . . . .     | 11 |
| 4.5  | Создание файла . . . . .                     | 11 |
| 4.6  | Открытый отредактированный файл . . . . .    | 12 |
| 4.7  | Открытие файла для просмотра . . . . .       | 12 |
| 4.8  | Компиляция файла . . . . .                   | 12 |
| 4.9  | 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 . . . . .               | 19 |
| 4.23 | Исполнение файла . . . . .                   | 20 |

# 1 Цель работы

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 в mc

Открываю Midnight Commander и ввожу в терминал mc (рис. 4.1).

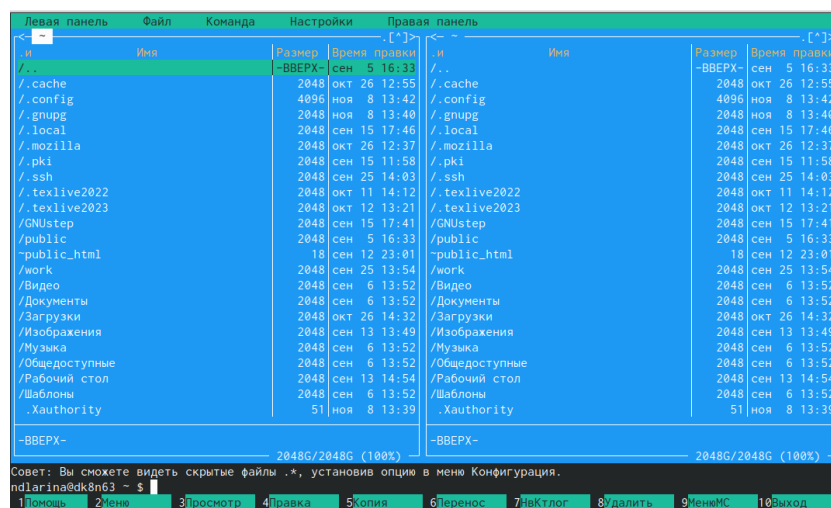


Рис. 4.1: Открытый mc

Далее перехожу в каталог ~/work/study/2022-2023/Архитектура Компьютера/arch-рс, используя файловый менеджер mc (рис. 4.2)

Left pane (File Manager):

| Имя                  | Размер | Время        | Права    |
|----------------------|--------|--------------|----------|
| ./                   | 2048   | сен 27 14:00 | -BBERPX- |
| ./git                | 2048   | окт 26 14:35 | 2048     |
| ./template           | 2048   | сен 27 14:00 | 2048     |
| ./presentation       | 2048   | сен 27 14:01 | 4896     |
| ./labs               | 2048   | сен 27 14:01 | 2048     |
| ./config             | 2048   | сен 27 14:00 | 2048     |
| ./gitmodules         | 278    | сен 27 14:00 | 2048     |
| ./gitignore          | 4637   | сен 27 14:00 | 2048     |
| ./gitattributes      | 1765   | сен 27 14:00 | 2048     |
| ./prepare            | 0      | сен 27 14:01 | 2048     |
| ./README.md          | 4277   | сен 27 14:00 | 2048     |
| ./README_git-flow.md | 5553   | сен 27 14:00 | 2048     |
| ./README_en.md       | 152    | сен 27 14:00 | 2048     |
| ./Makefile           | 815    | сен 27 14:00 | 2048     |
| ./LICENSE            | 18657  | сен 27 14:00 | 2048     |
| ./COURSE             | 8      | сен 27 14:01 | 2048     |
| ./CHANGELOG.md       | 2031   | сен 27 14:00 | 2048     |

Right pane (File Manager):

| Имя             | Размер | Время        | Права    |
|-----------------|--------|--------------|----------|
| ./              | 2048   | сен 5 16:33  | -BBERPX- |
| ./cache         | 2048   | окт 26 12:55 | 2048     |
| ./config        | 4896   | ноя 8 13:42  | 2048     |
| ./gnupg         | 2048   | ноя 8 13:40  | 2048     |
| ./local         | 2048   | сен 15 17:46 | 2048     |
| ./mozilla       | 2048   | окт 26 12:37 | 2048     |
| ./pki           | 2048   | сен 15 11:58 | 2048     |
| ./ssh           | 2048   | сен 25 14:03 | 2048     |
| ./texlive2022   | 2048   | ноя 11 14:12 | 2048     |
| ./texlive2023   | 2048   | окт 12 21:21 | 2048     |
| ./GNUpstap      | 2048   | сен 15 17:41 | 2048     |
| ./public        | 2048   | сен 5 16:33  | 2048     |
| ./public_html   | 18     | сен 12 23:01 | 2048     |
| ./work          | 2048   | сен 25 13:54 | 2048     |
| ./Видео         | 2048   | сен 6 13:52  | 2048     |
| ./Документы     | 2048   | сен 6 13:52  | 2048     |
| ./Загрузки      | 2048   | окт 26 14:32 | 2048     |
| ./Изображения   | 2048   | сен 13 13:49 | 2048     |
| ./Музыка        | 2048   | сен 6 13:52  | 2048     |
| ./Общедоступные | 2048   | сен 6 13:52  | 2048     |
| ./Рабочий стол  | 2048   | сен 13 14:54 | 2048     |
| ./Шаблоны       | 2048   | сен 6 13:52  | 2048     |
| ./Xauthority    | 51     | ноя 8 13:39  | 2048     |

Bottom status bar:

Left: -BBERPX- 2048G/2048G (100%)

Right: -BBERPX- 2048G/2048G (100%)

Bottom: Совет: Для получения вывода команды в окне просмотра наберите M-!   
 ndlarina@dk8n63 ~ /work/study/2023-2024/Архитектура компьютера/arch-ps \$

Рис. 4.2: Перемещение между директориями

Затем создаю каталог lab05 с помощью функциональной клавиши F7 (рис. 4.3).

| Левая панель  | Файл | Команда | Настройки    | Права |
|---|------|---------|--------------|-------|
| <pre>&lt; ...udy/2023-2024/Архитектура компьютера/arch-pc -.[^]&gt;</pre> |      |         |              |       |
| 'и  | Имя  | Размер  | Время правки |       |
| /..   |      | -BBEPX- | сен 27 14:00 |       |
| /.git   |      | 2048    | окт 26 14:35 |       |
| /template   |      | 2048    | сен 27 14:00 |       |
| /presentation   |      | 2048    | сен 27 14:01 |       |
| /labs   |      | 2048    | сен 27 14:01 |       |
| /lab05  |      | 2048    | ноя 8 13:49  |       |
| /config   |      | 2048    | сен 27 14:00 |       |
| .gitmodules   |      | 278     | сен 27 14:00 |       |
| .gitignore  |      | 4637    | сен 27 14:00 |       |

Рис. 4.3: Создание каталога

Перехожу в созданный каталог (рис. 4.4).

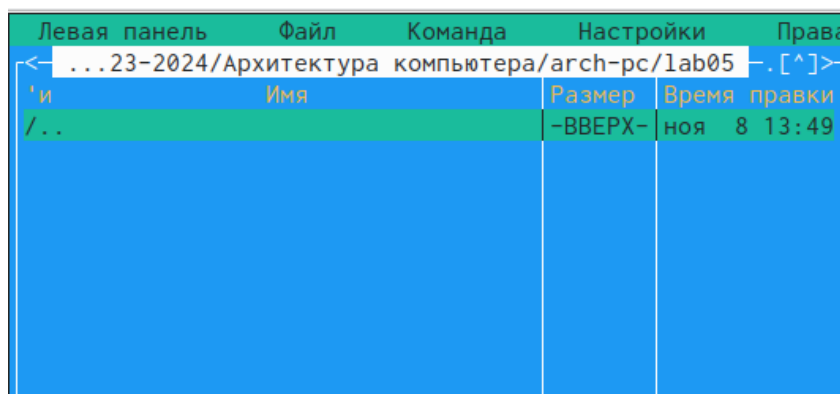


Рис. 4.4: Перемещение между директориями

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

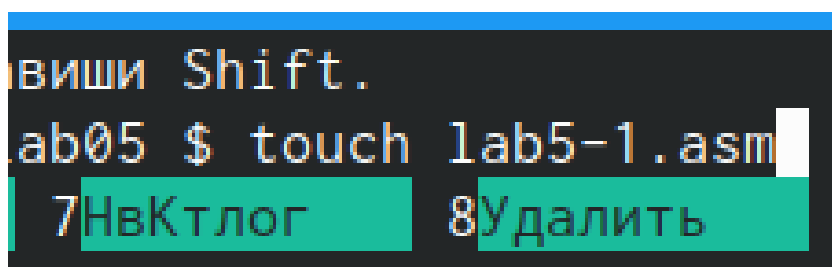
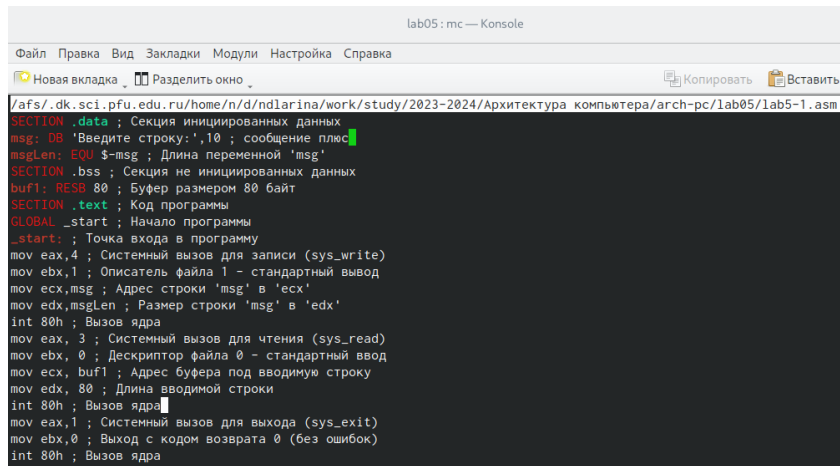


Рис. 4.5: Создание файла

## 4.2 Структура программы на языке ассемблера NASM

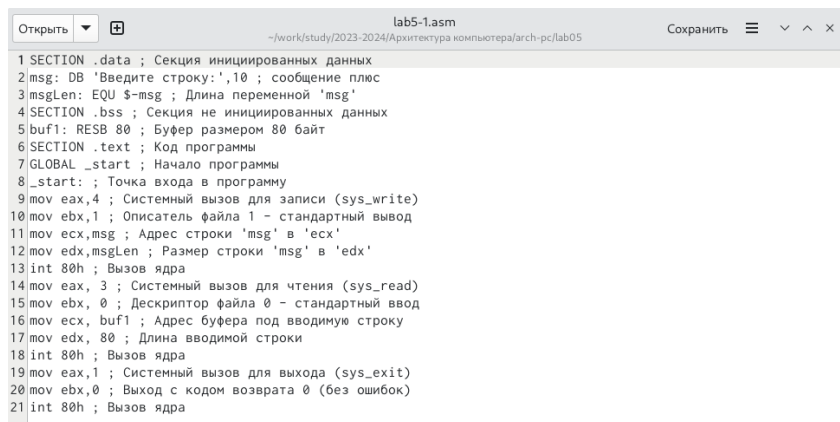
Открываю созданный файл для редактирования в редакторе nano с помощью функциональной клавиши F4. Ввожу в файл код программы для запроса строки у пользователя. Далее выхожу из файла (Ctrl+X), сохраняя изменения (Y, Enter) (рис. 4.6).



```
lab05: mc — Konsole
Файл Правка Вид Закладки Модули Настройка Справка
Новая вкладка Разделить окно Копировать Вставить
/afs/.dk.sci.pfu.edu.ru/home/n/d/ndlarina/work/study/2023-2024/Архитектура компьютера/arch-pc/lab05/lab5-1.asm
SECTION .data ; Секция иницированных данных
msg: DB 'Введите строку:',10 ; сообщение плюс
msgLen: EQU $-msg ; Длина переменной 'msg'
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,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: Открытый отредактированный файл

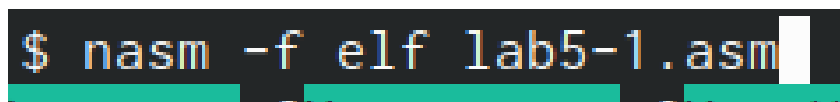
Затем с помощью функциональной клавиши F3 открываю файл для просмотра, чтобы проверить, содержит ли файл текст программы (рис. 4.7).



```
Открыть lab5-1.asm Сохранить
~/work/study/2023-2024/Архитектура компьютера/arch-pc/lab05
1 SECTION .data ; Секция иницированных данных
2 msg: DB 'Введите строку:',10 ; сообщение плюс
3 msgLen: EQU $-msg ; Длина переменной 'msg'
4 SECTION .bss ; Секция не иницированных данных
5 buf1: RESB 80 ; Буфер размером 80 байт
6 SECTION .text ; Код программы
7 GLOBAL _start ; Начало программы
8 _start: ; Точка входа в программу
9 mov eax,4 ; Системный вызов для записи (sys_write)
10 mov ebx,1 ; Описатель файла 1 - стандартный вывод
11 mov ecx,msg ; Адрес строки 'msg' в 'ecx'
12 mov edx,msgLen ; Размер строки 'msg' в 'edx'
13 int 80h ; Вызов ядра
14 mov eax, 3 ; Системный вызов для чтения (sys_read)
15 mov ebx, 0 ; Дескриптор файла 0 - стандартный ввод
16 mov ecx, buf1 ; Адрес буфера под вводимую строку
17 mov edx, 80 ; Длина вводимой строки
18 int 80h ; Вызов ядра
19 mov eax,1 ; Системный вызов для выхода (sys_exit)
20 mov ebx,0 ; Выход с кодом возврата 0 (без ошибок)
21 int 80h ; Вызов ядра
```

Рис. 4.7: Открытие файла для просмотра

Транслирую текст программы файла в объектный файл командой `nasm -f elf lab5-1.asm`. Создался объектный файл `lab5-1.o` (рис. 4.8).



```
$ nasm -f elf lab5-1.asm
```

Рис. 4.8: Компиляция файла

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`.

```
$ ld -m elf_i386 -o lab5-1 lab5-1.o
```

Рис. 4.9: Передача на обработку компоновщику

Далее 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).

```
ndlarina@dk8n63 ~/work/study/2023-2024/Архитектура компьютера/arch-pc/lab05 $ ./lab5-1
Введите строку:
Ларина Наталья Денисовна
```

Рис. 4.10: Исполнение файла

## 4.3 Подключение внешнего файла

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

| Левая панель                        | Файл | Команда | Настройки    | Права |
|-------------------------------------|------|---------|--------------|-------|
| ~/Загрузки                          |      |         |              |       |
| Имя                                 |      | Размер  | Время правки |       |
| /..                                 |      | -ВВЕРХ- | ноя 8 13:39  |       |
| Снимок экрана от ~9-13 13-49-52.png |      | 203405  | сен 13 13:52 |       |
| Л02_Ларина_отчет.doc                |      | 0       | сен 27 14:07 |       |
| Л01_Ларина_отчет.pdf                |      | 0       | сен 27 14:07 |       |
| Л04_Ларина_отчёт.pdf                |      | 1002421 | окт 26 14:31 |       |
| Л04_Ларина_отчёт~1.pdf              |      | 1002421 | окт 26 14:32 |       |
| in_out.asm                          |      | 3942    | ноя 8 14:18  |       |
| Annot_AКиОС_НФИ6д.docx              |      | 16342   | сен 13 13:59 |       |

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).

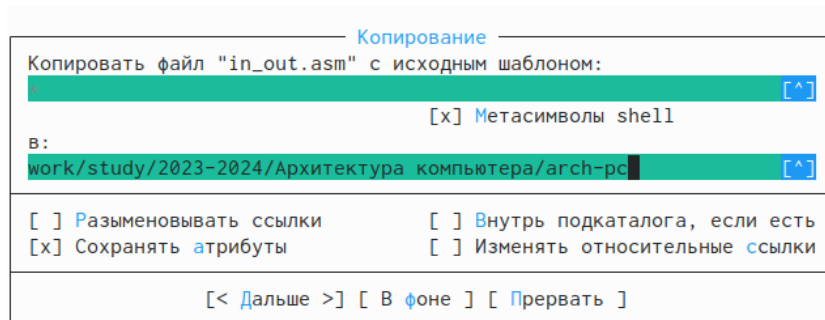


Рис. 4.12: Копирование файла

Затем 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 (рис. 4.13).

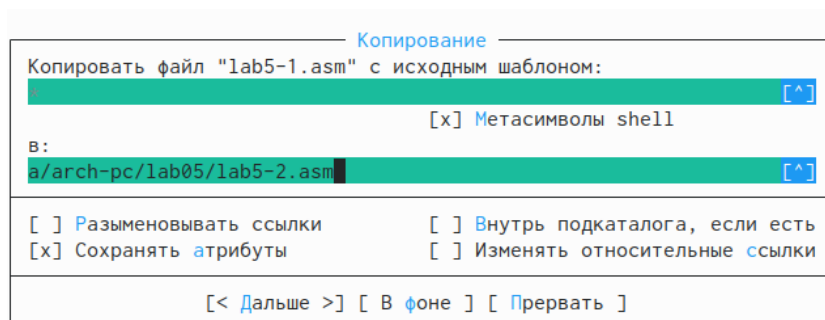
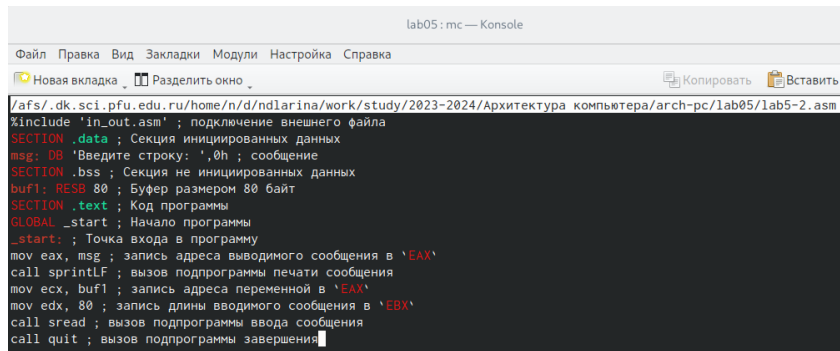


Рис. 4.13: Копирование файла

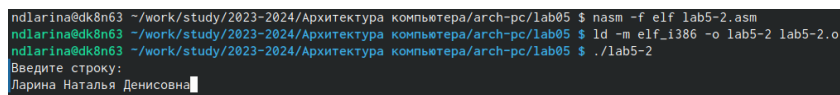
Изменяю содержимое файла lab5-2.asm во встроенном редакторе nano (рис. 4.14), чтобы в программе использовались подпрограммы из внешнего файла in\_out.asm.



```
lab05: mc — Konsole
Файл  Правка  Вид  Закладки  Модули  Настройка  Справка
Новая вкладка  Разделить окно  Копировать  Вставить
/afs/dk.sci.pfu.edu.ru/home/n/d/ndlarina/work/study/2023-2024/Архитектура компьютера/arch-pc/lab05/lab5-2.asm
#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 sprintLF; 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. 4.14: 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).



```
ndlarina@dk8n63 ~/work/study/2023-2024/Архитектура компьютера/arch-pc/lab05 $ nasm -f elf lab5-2.asm
ndlarina@dk8n63 ~/work/study/2023-2024/Архитектура компьютера/arch-pc/lab05 $ ld -m elf_i386 -o lab5-2 lab5-2.o
ndlarina@dk8n63 ~/work/study/2023-2024/Архитектура компьютера/arch-pc/lab05 $ ./lab5-2
Введите строку:
Ларина Наталья Денисовна
```

Рис. 4.15: Исполнение файла

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

```

lab5-2.asm      [-M--] 11 L: [ 1+13 14/ 16] *(904 / 963b) 0032 0x020
#include "lab5-1.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

```

Рис. 4.16: Отредактированный файл

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

```

ndlarina@dk8n63 ~/work/study/2023-2024/Архитектура компьютера/arch-pc/lab05 $ nasm -f elf lab5-2.asm
ndlarina@dk8n63 ~/work/study/2023-2024/Архитектура компьютера/arch-pc/lab05 $ ld -m elf_i386 -o lab5-2-2 lab5-2.o
ndlarina@dk8n63 ~/work/study/2023-2024/Архитектура компьютера/arch-pc/lab05 $ ./lab5-2-2
Введите строку: Ларина Наталья Денисовна

```

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).



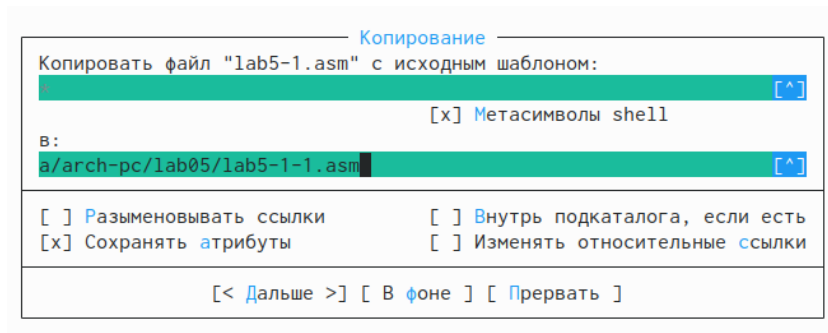


Рис. 4.18: Копирование файла

Открываю созданный файл для редактирования с помощью функциональной клавиши F4. Изменяю программу так, чтобы кроме вывода приглашения и запроса ввода, она выводила вводимую пользователем строку (рис. 4.19).

```

/afs/.dk.sci.pfu.edu.ru/home/n/d/ndlarina/work/study/2023-2024/Архитектура компьютера/arch-pc/lab05/lab5-1-1.asm
SECTION .data ; Секция иницированных данных
msg: DB 'Введите строку:',10 ; сообщение плюс
msgLen: EQU $-msg ; Длина переменной 'msg'
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,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 в edx
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. 4.19: 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 (рис. 4.20).

```
ndlarina@dk8n63 ~/work/study/2023-2024/Архитектура компьютера/arch-pc/lab05 $ nasm -f elf lab5-1-1.asm
ndlarina@dk8n63 ~/work/study/2023-2024/Архитектура компьютера/arch-pc/lab05 $ ld -m elf_i386 -o lab5-1-1 lab5-1-1.o
ndlarina@dk8n63 ~/work/study/2023-2024/Архитектура компьютера/arch-pc/lab05 $ ./lab5-1-1
Введите строку:
Лarina Наталья Денисовна
Лarina Наталья Денисовна
ndlarina@dk8n63 ~/work/study/2023-2024/Архитектура компьютера/arch-pc/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 ; 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. 4.21).

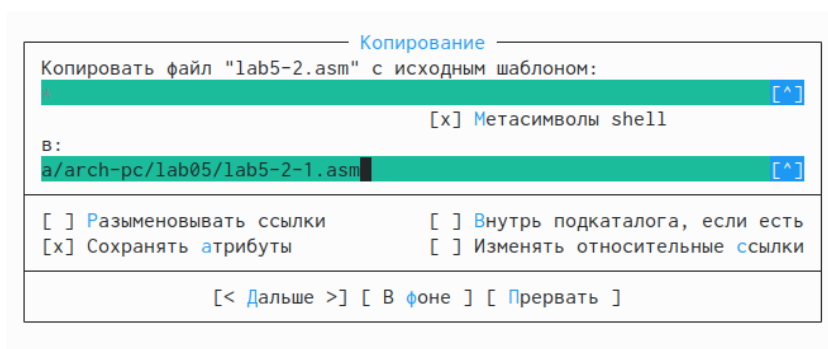


Рис. 4.21: Копирование файла

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).

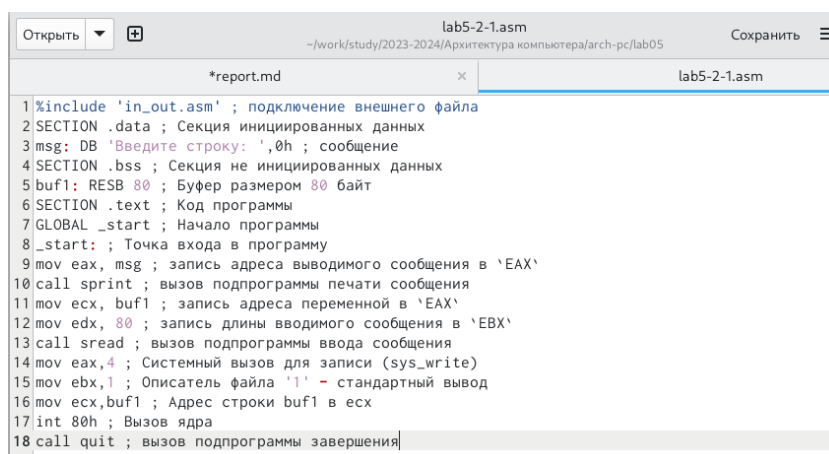


Рис. 4.22: Редактирование файла

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).

```
ndlarina@dk8n63 ~/work/study/2023-2024/Архитектура компьютера/arch-pc/lab05 $ nasm -f elf lab5-2-1.asm
ndlarina@dk8n63 ~/work/study/2023-2024/Архитектура компьютера/arch-pc/lab05 $ ld -m elf_i386 -o lab5-2-1 lab5-2-1.o
ndlarina@dk8n63 ~/work/study/2023-2024/Архитектура компьютера/arch-pc/lab05 $ ./lab5-2-1
Введите строку: Ларина Наталья
Ларина Наталья
```

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

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

## **6 Список литературы**

1. Лабораторная работа №5