

Лабораторная работа №5

Дисциплина: Архитектура компьютера

карпачев Ярослав Олегович

Содержание

| | | |
|----------|--|-----------|
| 1 | Цель работы | 4 |
| 2 | Задание | 5 |
| 3 | Теоретическое введение | 6 |
| 3.1 | Основы работы с Midnight Commander | 7 |
| 3.2 | Структура программы на языке ассемблера NASM | 8 |
| 3.3 | Описание инструкции mov | 9 |
| 3.4 | Описание инструкций int | 9 |
| 4 | Выполнение лабораторной работы | 10 |
| 4.1 | Основы работы с Midnight Commander | 10 |
| 4.2 | Подключение внешнего файла in_out.asm | 12 |
| 4.3 | Задание для самостоятельной работы | 15 |
| 5 | Выводы | 18 |

Список иллюстраций

| | | |
|------|--|----|
| 4.1 | Создание каталога lab05 | 10 |
| 4.2 | Создание lab5-1.asm | 10 |
| 4.3 | Ввод текста программы из листинга | 11 |
| 4.4 | Проверка содержимого программы | 11 |
| 4.5 | Трансляция, компоновка, запуск программы | 12 |
| 4.6 | Копирование файла | 12 |
| 4.7 | Копирование файла | 13 |
| 4.8 | Редактирование файла | 13 |
| 4.9 | Исполнение файла | 14 |
| 4.10 | Отредактированный файл | 14 |
| 4.11 | Вывод программ | 14 |
| 4.12 | Копирование файла | 15 |
| 4.13 | Редактирование файла | 15 |
| 4.14 | Исполнение файла | 16 |
| 4.15 | Копирование файла | 16 |
| 4.16 | Редактирование файла | 17 |
| 4.17 | Исполнение файла | 17 |

1 Цель работы

Приобретение практических навыков 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 (четырёхбайтное 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).

3.1 Основы работы с Midnight Commander

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. В Midnight Commander используются функциональные клавиши `F1` — `F10`, к которым привязаны часто выполняемые операции (табл. 3.1).

Таблица 3.1: Функциональные клавиши Midnight Commander

| Функциональные | |
|----------------|---|
| клавиши | Выполняемое действие |
| F1 | вызов контекстно-зависимой подсказки |
| F2 | вызов меню, созданного пользователем |
| F3 | просмотр файла, на который указывает подсветка в активной панели |
| F4 | вызов встроенного редактора для файла, на который указывает подсветка в активной панели |

| Функциональные | |
|----------------|--|
| клавиши | Выполняемое действие |
| F5 | копирование файла (группы файлов из каталога), отображаемого в активной панели, в каталог, отображаемый на второй панели |
| F6 | перенос файла (группы файлов из каталога), отображаемого в активной панели, в каталог, отображаемый на второй панели |
| F7 | создание подкаталога в каталоге, отображаемом в активной панели |
| F8 | удаление файла (подкаталога) или группы отмеченных файлов |
| F9 | вызов основного меню программы |
| F10 | выход из программы |

3.2 Структура программы на языке ассемблера NASM

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 (учетверённое 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. Для определения строк принято использовать директиву DB в связи с особенностями хранения данных в оперативной памяти.

3.3 Описание инструкции mov

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).

3.4 Описание инструкций int

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

Открыл Midnight Commander с mc. Перешел в каталог ~/work/arch-pc. С помощью F7 создал папку lab05 (рис. [4.1]). Перешел в каталог и, пользуясь командой touch, создал файл lab5-1.asm (рис. [4.2])

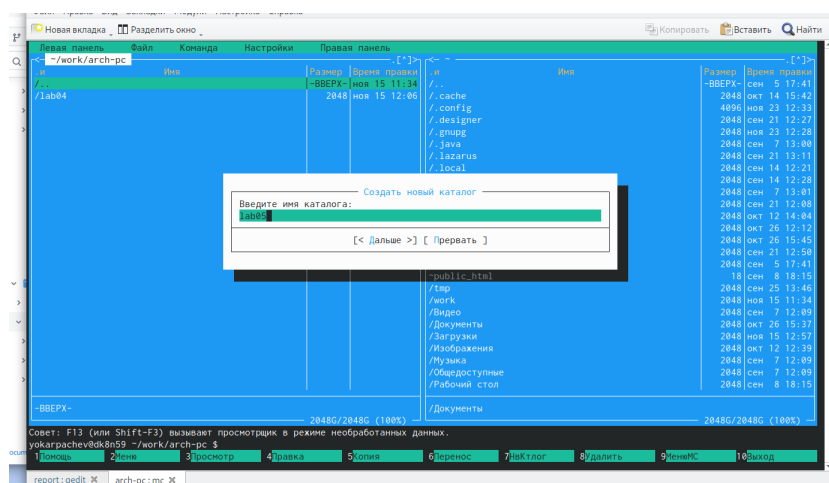


Рис. 4.1: Создание каталога lab05

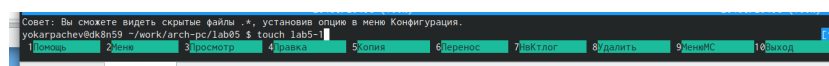
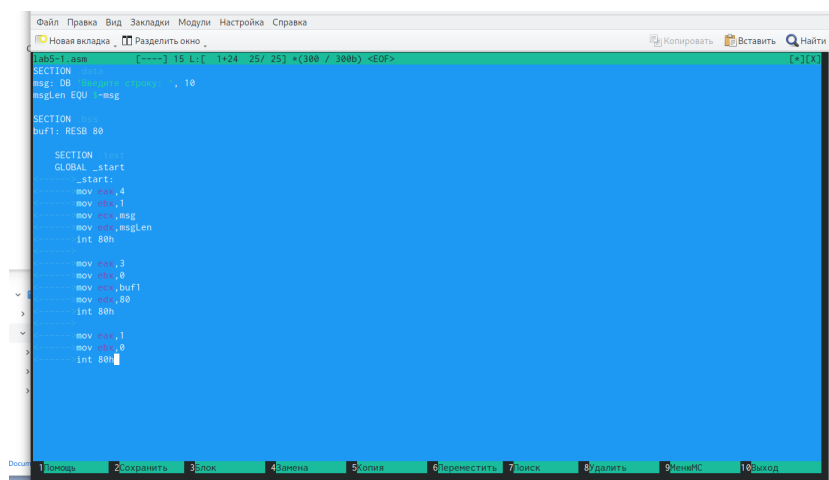


Рис. 4.2: Создание lab5-1.asm

С помощью функциональной клавиши F4 открыл файл lab5-1.asm для редактирования в редакторе. Скопировал текст программы из листинга 5.1, сохранил

файл (рис. [4.3]). С помощью F3 открыл файл lab5-1.asm для просмотра. Убедился, что файл содержит текст программы (рис. [4.4]).



```
lab5-1.asm [-----] 15 L: [ 1+24 25/ 25] *(388 / 388b) <EOF> [-----] [+1][X]
msg: DB "Введите строку: ", 10
msgLen EQU $-msg

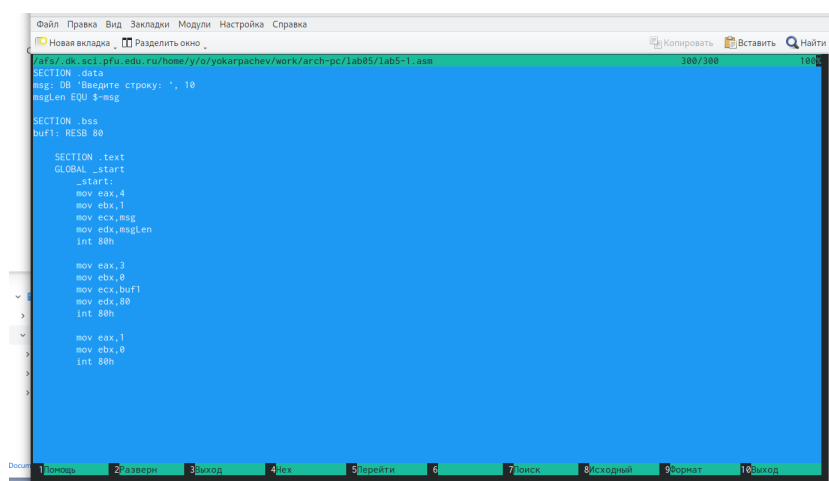
SECTION
buf1: RESB 80

SECTION
GLOBAL _start
_start:
    mov eax, 4
    mov ebx, 1
    mov ecx, msg
    mov edx, msgLen
    int 80h

    mov eax, 3
    mov ebx, 0
    mov ecx, buf1
    mov edx, 80
    int 80h

    mov eax, 1
    mov ebx, 0
    int 80h
```

Рис. 4.3: Ввод текста программы из листинга



```
lab5-1.asm [-----] 15 L: [ 1+24 25/ 25] *(388 / 388b) <EOF> [-----] [+1][X]
msg: DB "Введите строку: ", 10
msgLen EQU $-msg

SECTION
buf1: RESB 80

SECTION
GLOBAL _start
_start:
    mov eax, 4
    mov ebx, 1
    mov ecx, msg
    mov edx, msgLen
    int 80h

    mov eax, 3
    mov ebx, 0
    mov ecx, buf1
    mov edx, 80
    int 80h

    mov eax, 1
    mov ebx, 0
    int 80h
```

Рис. 4.4: Проверка содержимого программы

Оттранслировал программу lab5-1.asm в объектный файл. Выполнил компоновку объектного файла и запустил исполняемый файл. Программа выводит строку 'Введите строку:' и ожидает ввода с клавиатуры. На запрос ввел свои ФИО (рис. [4.5]).

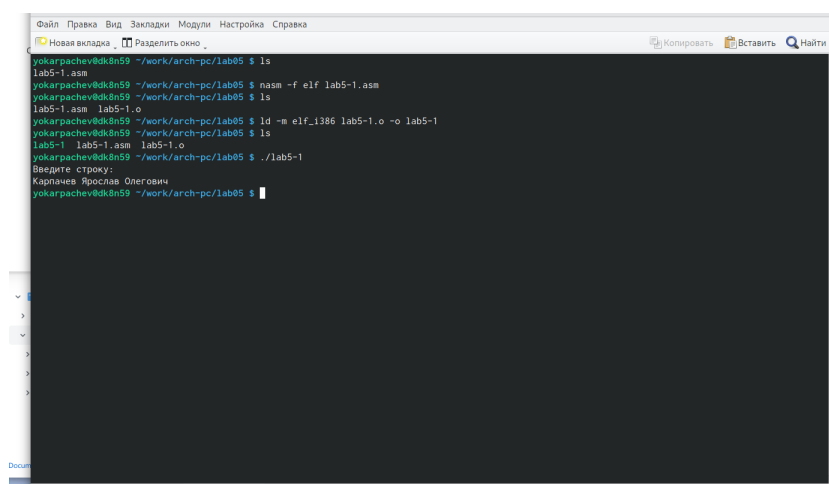


Рис. 4.5: Трансляция, компоновка, запуск программы

4.2 Подключение внешнего файла in_out.asm

Скачал файл in_out.asm со страницы курса в ТУИСе. С помощью функциональной клавиши F5 скопировал in_out.asm из каталога Загрузки в созданный каталог lab05 (рис. [4.6]).

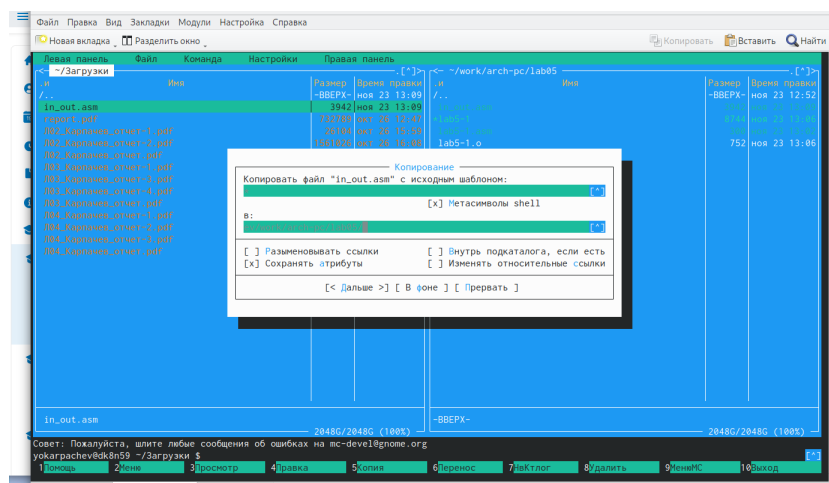


Рис. 4.6: Копирование файла

С помощью функциональной клавиши F5 скопировал lab5-1 в тот же каталог, с другим именем. (рис. [4.7]).

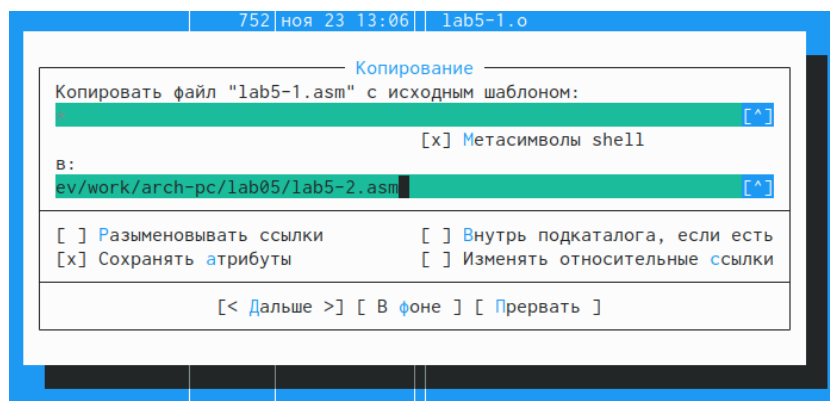


Рис. 4.7: Копирование файла

Изменил содержимое файла lab5-2.asm в редакторе кода (рис. [4.8]), чтобы в программе использовались подпрограммы из файла in_out.asm.

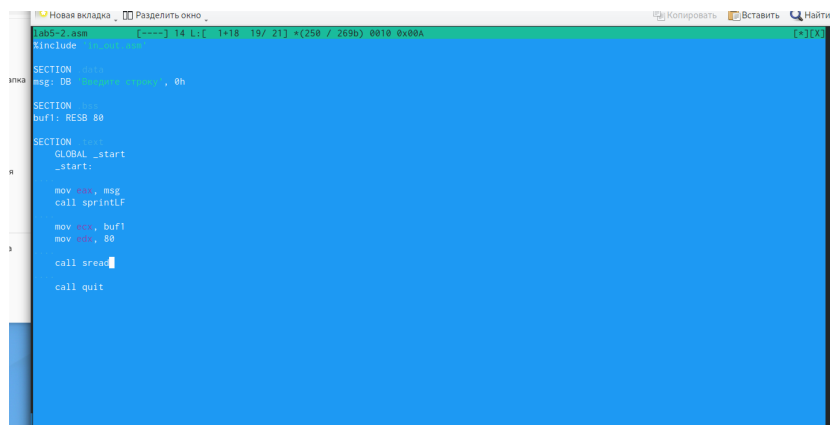
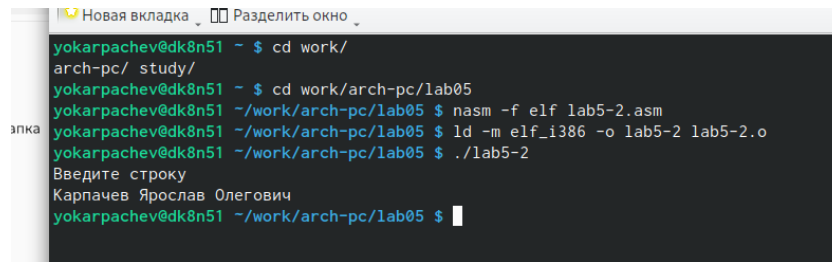


Рис. 4.8: Редактирование файла

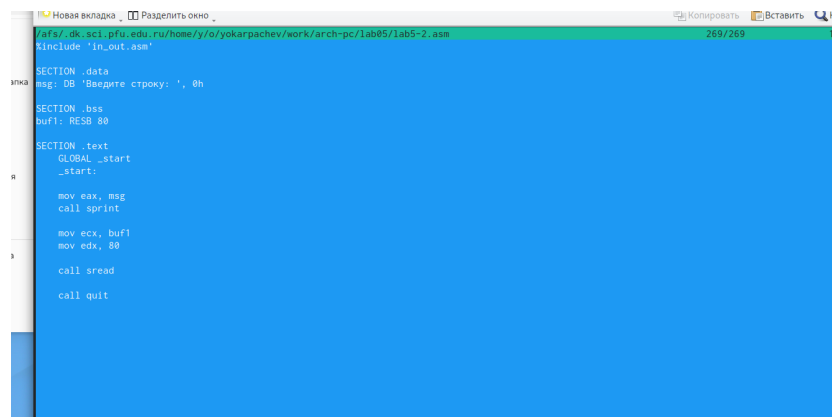
Оттранслировал файл в объектный код командой `nasm -f elf lab5-2.asm`. Создался объектный файл lab5-2.o. Скомпоновал объектный файла с помощью команды `ld -m elf_i386 -o lab5-2 lab5-2.o` Создался исполняемый файл lab5-2. Запустил файл (рис. [4.11]).



```
Новая вкладка Разделить окно
yokarpachev@dk8n51 ~ $ cd work/
arch-pc/ study/
yokarpachev@dk8n51 ~ $ cd work/arch-pc/lab05
yokarpachev@dk8n51 ~/work/arch-pc/lab05 $ nasm -f elf lab5-2.asm
yokarpachev@dk8n51 ~/work/arch-pc/lab05 $ ld -m elf_i386 -o lab5-2 lab5-2.o
yokarpachev@dk8n51 ~/work/arch-pc/lab05 $ ./lab5-2
Введите строку
Карпачев Ярослав Олегович
yokarpachev@dk8n51 ~/work/arch-pc/lab05 $
```

Рис. 4.9: Исполнение файла

Открыл файл lab5-2.asm с помощью F4. Изменил в нем подпрограмму sprintLF на sprint. Сохранил изменения и открыл файл для просмотра, чтобы проверить, что программа сохранилась. (рис. [??]).



```
Новая вкладка Разделить окно
/afs/dk.sci.pfu.edu.ru/home/y/o/yokarpachev/work/arch-pc/lab05/lab5-2.asm 269/269 100
#include 'in.out.asm'

SECTION .data
msg: DB 'Введите строку: ', 0h

SECTION .bss
buf1: RESB 80

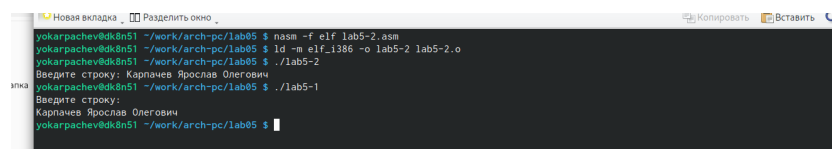
SECTION .text
GLOBAL _start
_start:
    mov eax, msg
    call sprint

    mov ecx, buf1
    mov edx, 80

    call sread
    call quit
```

Рис. 4.10: Отредактированный файл

Разница между первым файлом lab5-1 и вторым lab5-2 в том, Первый запрашивает ввод с новой строки, а lab5-2 запрашивает ввод без переноса на новую строку, потому что была замена sprintLF на sprint (рис. [4.11]).



```
Новая вкладка Разделить окно
yokarpachev@dk8n51 ~/work/arch-pc/lab05 $ nasm -f elf lab5-2.asm
yokarpachev@dk8n51 ~/work/arch-pc/lab05 $ ld -m elf_i386 -o lab5-2 lab5-2.o
yokarpachev@dk8n51 ~/work/arch-pc/lab05 $ ./lab5-2
Введите строку: Карпачев Ярослав Олегович
yokarpachev@dk8n51 ~/work/arch-pc/lab05 $ ./lab5-1
Введите строку:
Карпачев Ярослав Олегович
yokarpachev@dk8n51 ~/work/arch-pc/lab05 $
```

Рис. 4.11: Вывод программ

4.3 Задание для самостоятельной работы

1. Создал копию файла lab5-1.asm с именем lab5-1-1.asm с помощью F5 (рис. [4.12]).

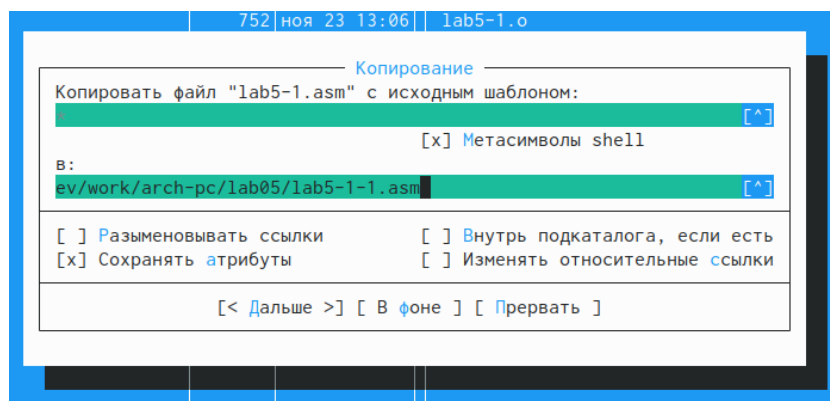


Рис. 4.12: Копирование файла

С F4 открываю созданный файл для редактирования. Изменил программу так, чтобы она дополнительно выводит ФИО. (рис. [4.13]).

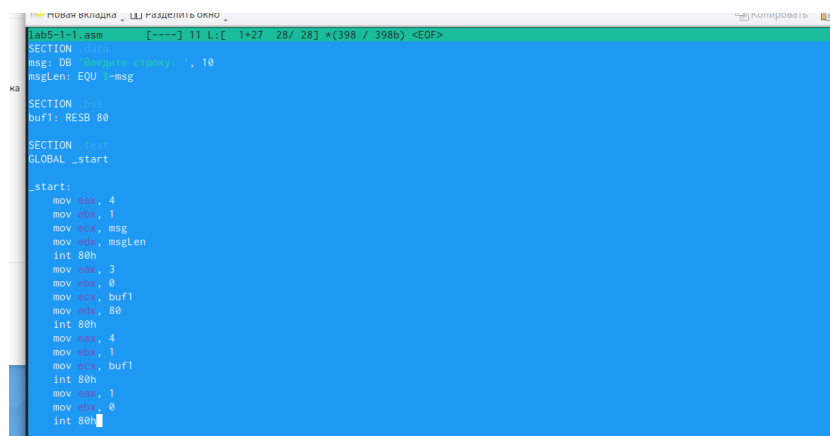
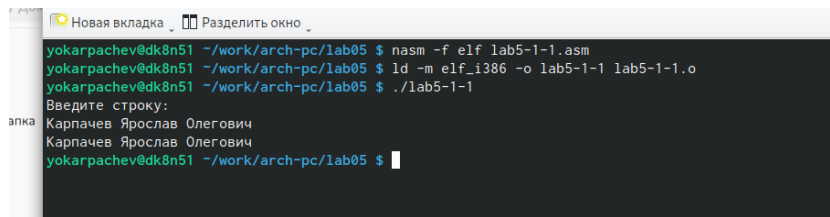


Рис. 4.13: Редактирование файла

Создал объектный файл, отдал его на обработку компоновщику, получил исполняемый файл lab5-1-1. Запустил полученный исполняемый файл. Все работает правильно, программа запрашивает ФИО и выводит его.(рис. [4.14]).



```
Новая вкладка Разделить окно
yokarpachev@dk8n51 ~/work/arch-pc/lab05 $ nasm -f elf lab5-1-1.asm
yokarpachev@dk8n51 ~/work/arch-pc/lab05 $ ld -m elf_i386 -o lab5-1-1 lab5-1-1.o
yokarpachev@dk8n51 ~/work/arch-pc/lab05 $ ./lab5-1-1
Введите строку:
Карпачев Ярослав Олегович
Карпачев Ярослав Олегович
yokarpachev@dk8n51 ~/work/arch-pc/lab05 $
```

Рис. 4.14: Исполнение файла

2. Создал копию файла lab5-2.asm с именем lab5-2-1.asm с помощью F5 (рис. [4.15]).

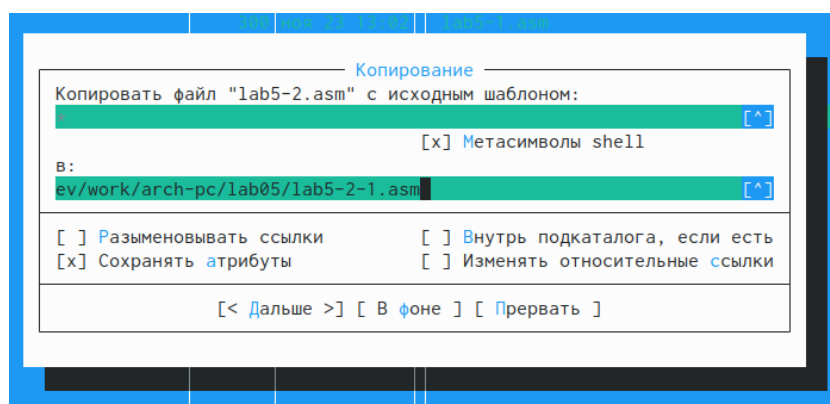
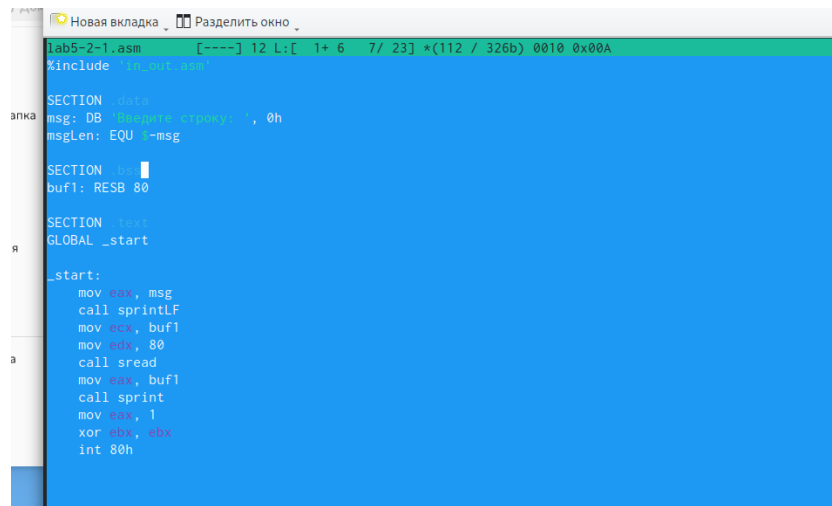


Рис. 4.15: Копирование файла

С помощью функциональной клавиши F4 открывал созданный файл для редактирования. Изменил программу чтобы она дополнительно выводила ФИО. (рис. [4.16]).



```
lab5-2-1.asm [---] 12 L: [1+ 6 7/ 23] *(112 / 326b) 0010 0x00A
%include "io.inc.asm"

SECTION .data
msg: DB "Введите строку: ", 0h
msgLen: EQU $-msg

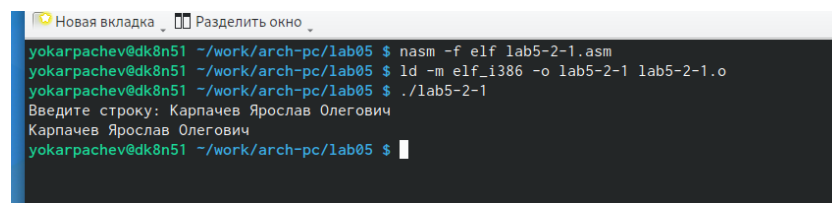
SECTION .bss
buf1: RESB 80

SECTION .text
GLOBAL _start

_start:
    mov eax, msg
    call sprintf
    mov ecx, buf1
    mov edx, 80
    call sread
    mov eax, buf1
    call sprint
    mov ecx, 1
    xor ebx, ebx
    int 80h
```

Рис. 4.16: Редактирование файла

Создал объектный файл, отдал его на обработку компоновщику, получил исполняемый файл lab5-2-1. Запустил полученный исполняемый файл. Все работает правильно, программа запрашивает ФИО и выводит его. (рис. [4.17]).



```
yokarpachev@dk8n51 ~/work/arch-pc/lab05 $ nasm -f elf lab5-2-1.asm
yokarpachev@dk8n51 ~/work/arch-pc/lab05 $ ld -m elf_i386 -o lab5-2-1 lab5-2-1.o
yokarpachev@dk8n51 ~/work/arch-pc/lab05 $ ./lab5-2-1
Введите строку: Карпачев Ярослав Олегович
Карпачев Ярослав Олегович
yokarpachev@dk8n51 ~/work/arch-pc/lab05 $
```

Рис. 4.17: Исполнение файла

5 Выводы

Я приобрел навыки работы в Midnight Commander, а также освоил инструкции языка ассемблера `mov` и `int`.