

Отчет по лабораторной работе №5

Дисциплина: архитектура компьютера

Рахматова Жылдыз Талантбековна

Содержание

1	Цель работы	5
2	Задание	6
3	Теоретическое введение	7
4	Выполнение лабораторной работы	9
4.1	Основы работы с mc	9
4.2	Структура программы на языке ассемблера NASM	11
4.3	Подключение внешнего файла	13
4.4	Выполнение заданий для самостоятельной работы	16
5	Выводы	23

Список иллюстраций

4.1	Открытый тс	9
4.2	Перемещение между директориями и создание каталога	10
4.3	создание файла	10
4.4	Открытие файла для редактирования	11
4.5	Редактирование и открытие файла	12
4.6	Компиляция файла и передача на обработку компоновщику	12
4.7	Исполнение файла	13
4.8	Скачанный файл	13
4.9	Копирование файла	14
4.10	Копирование файла	14
4.11	Редактирование файла	15
4.12	Исполнение файла	15
4.13	Отредактированный файл	16
4.14	Исполнение файла	16
4.15	Копирование файла	17
4.16	Редактирование файла	18
4.17	Исполнение файла	18
4.18	Копирование файла	20
4.19	Редактирование файла	21
4.20	Исполнение файла	21

Список таблиц

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 mc
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 (учетверённое слово);

DT (define ten bytes) — определяет переменную размером в 10 байт. Директивы используются д

```
mov dst,src
```

Здесь операнд dst — приёмник, а src — источник. В качестве операнда могут выступать регистры (register), ячейки памяти (memory) и непосредственные значе-

ния (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).

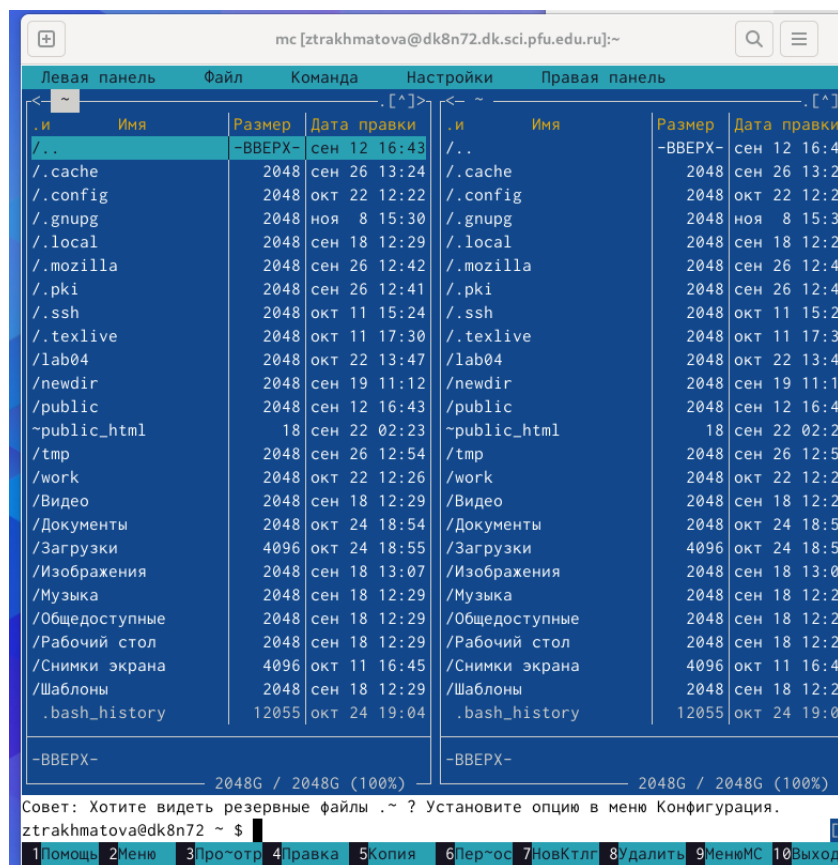


Рис. 4.1: Открытый mc

Перехожу в каталог ~/work/arch-рс, используя файловый менеджер mc и с помощью функциональной клавиши F7 создаю каталог lab05 (рис. 4.2)

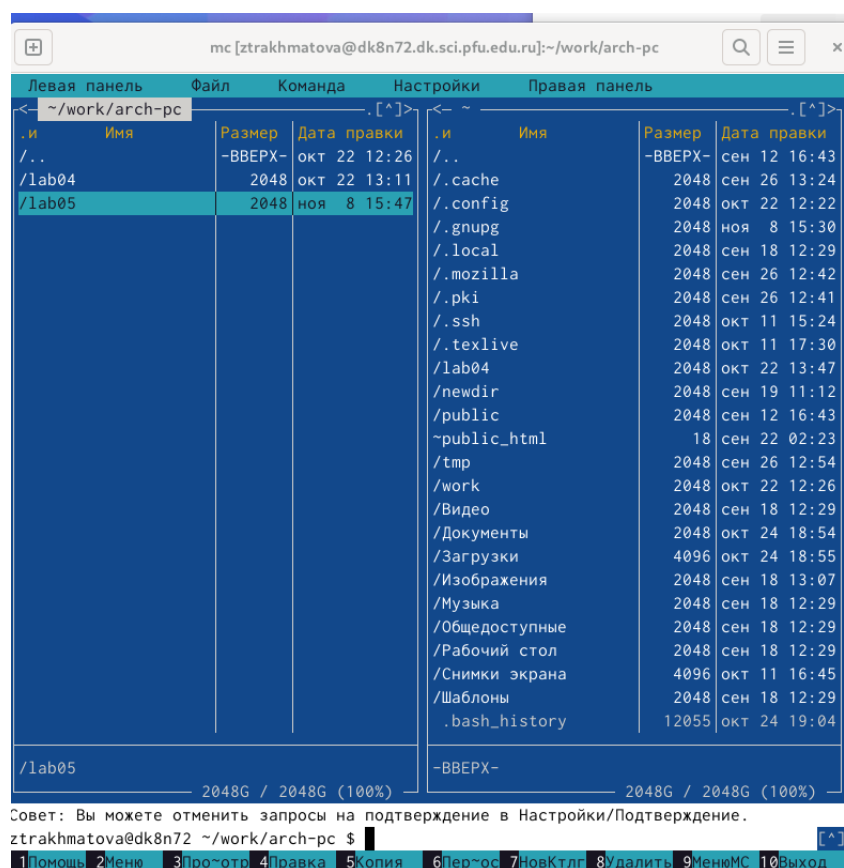


Рис. 4.2: Перемещение между директориями и создание каталога

Перехожу в созданный каталог. В строке ввода прописываю команду touch lab5-1.asm, чтобы создать файл, в котором буду работать (рис. 4.3).

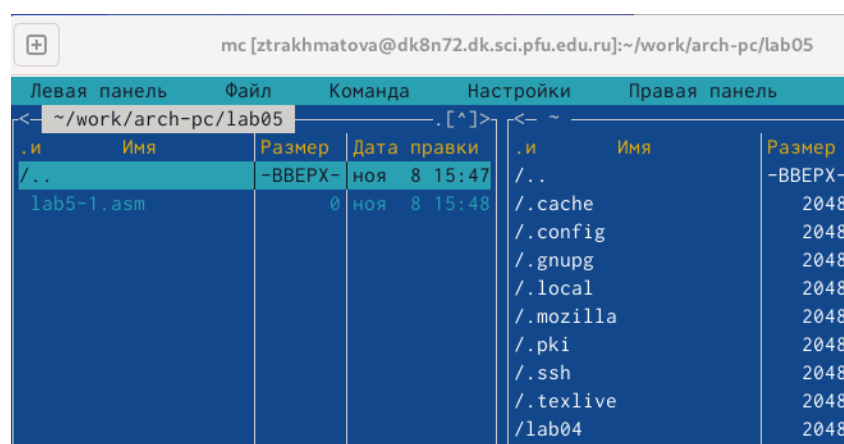
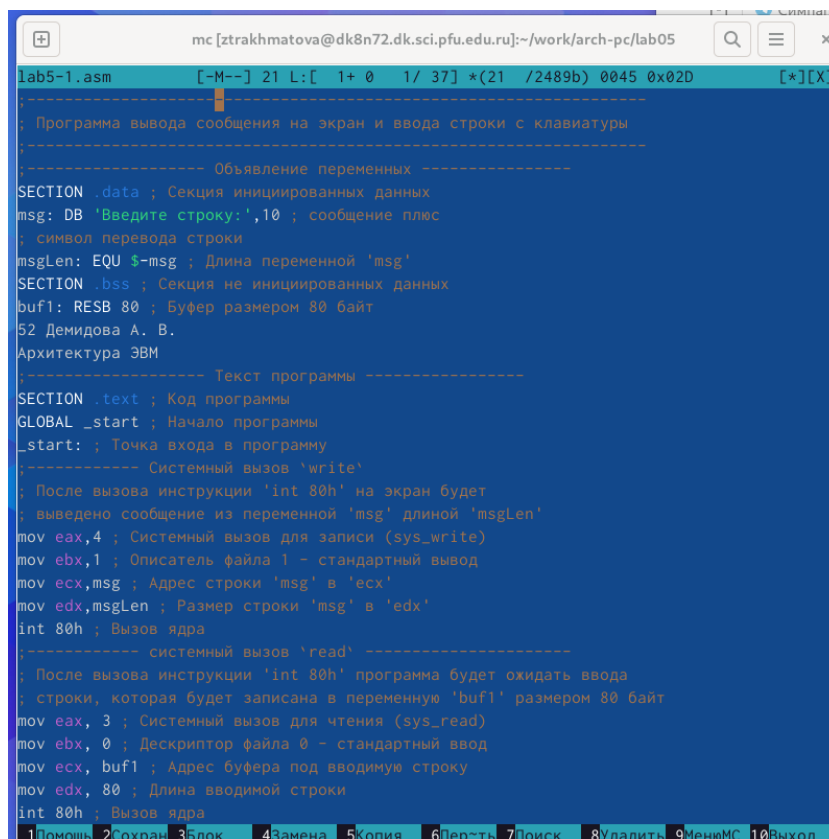


Рис. 4.3: создание файла

4.2 Структура программы на языке ассемблера NASM

С помощью функциональной клавиши F4 открываю созданный файл для редактирования в редакторе mcedit. Ввожу в файл код программы для запроса строки у пользователя (рис. 4.4).



```
lab5-1.asm [-M--] 21 L:[ 1+ 0 1/ 37] *(21 /2489b) 0045 0x02D [*][X]
; Программа вывода сообщения на экран и ввода строки с клавиатуры
;----- Объявление переменных -----
SECTION .data ; Секция инициализированных данных
msg: DB 'Введите строку:',10 ; сообщение плюс
; символ перевода строки
msgLen: EQU $-msg ; Длина переменной 'msg'
SECTION .bss ; Секция не инициализированных данных
buf1: RESB 80 ; Буфер размером 80 байт
52 Демидова А. В.
Архитектура ЭВМ
;----- Текст программы -----
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
1Помощь 2Сохран 3Блок 4Замена 5Копия 6Пер-ть 7Поиск 8Удалить 9МенюМС 10Выход
```

Рис. 4.4: Открытие файла для редактирования

Далее выхожу из файла (Ctrl+X), сохраняя изменения (Y, Enter). С помощью функциональной клавиши F3 открываю файл для просмотра, чтобы проверить, содержит ли файл текст программы (рис. 4.5).

```

/afs/.dk.sci.pfu.edu.ru/ho-k/arch-pc/lab05/lab5-1.asm 2129/2489 85%
;
; Программа вывода сообщения на экран и ввода строки с клавиатуры
;
;----- Объявление переменных -----
SECTION .data ; Секция иницированных данных
msg: DB 'Введите строку:',10 ; сообщение плюс
; символ перевода строки
msgLen: EQU $-msg ; Длина переменной 'msg'
SECTION .bss ; Секция не иницированных данных
buf1: RESB 80 ; Буфер размером 80 байт
52 Демидова А. В.
Архитектура ЭВМ
;----- Текст программы -----
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
1Помощь 2Разгн 3Выход 4Нех 5Перейти 6 7Поиск 8Исхтнй 9Формат 10Выход

```

Рис. 4.5: Редактирование и открытие файла

Транслирую текст программы файла в объектный файл командой `nasm -f elf lab5-1.asm`. Создался объектный файл `lab5-1.o`. Выполняю компоновку объектно-го файла с помощью команды `ld -m elf_i386 -o lab5-1 lab5-1.o` (рис. 4.6). Создался исполняемый файл `lab5-1`.

```

ztrakhmatova@dk8n72 ~/work/arch-pc/lab05 $ nasm -f elf lab5-1.asm
ztrakhmatova@dk8n72 ~/work/arch-pc/lab05 $ ld -m elf_i386 -o lab5-1 lab5-1.o
ztrakhmatova@dk8n72 ~/work/arch-pc/lab05 $ 

```

Рис. 4.6: 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. 4.7).

```
ztrakhmatova@dk8n72 ~/work/arch-pc/lab05 $ ./lab5-1
Введите строку:
Рахматова Жылдыз Талантбековна
ztrakhmatova@dk8n72 ~/work/arch-pc/lab05 $ █
```

Рис. 4.7: Исполнение файла

4.3 Подключение внешнего файла

Скачиваю файл `in_out.asm` со страницы курса в ТУИС. Он сохранился в каталог “Загрузки” (рис. 4.8).

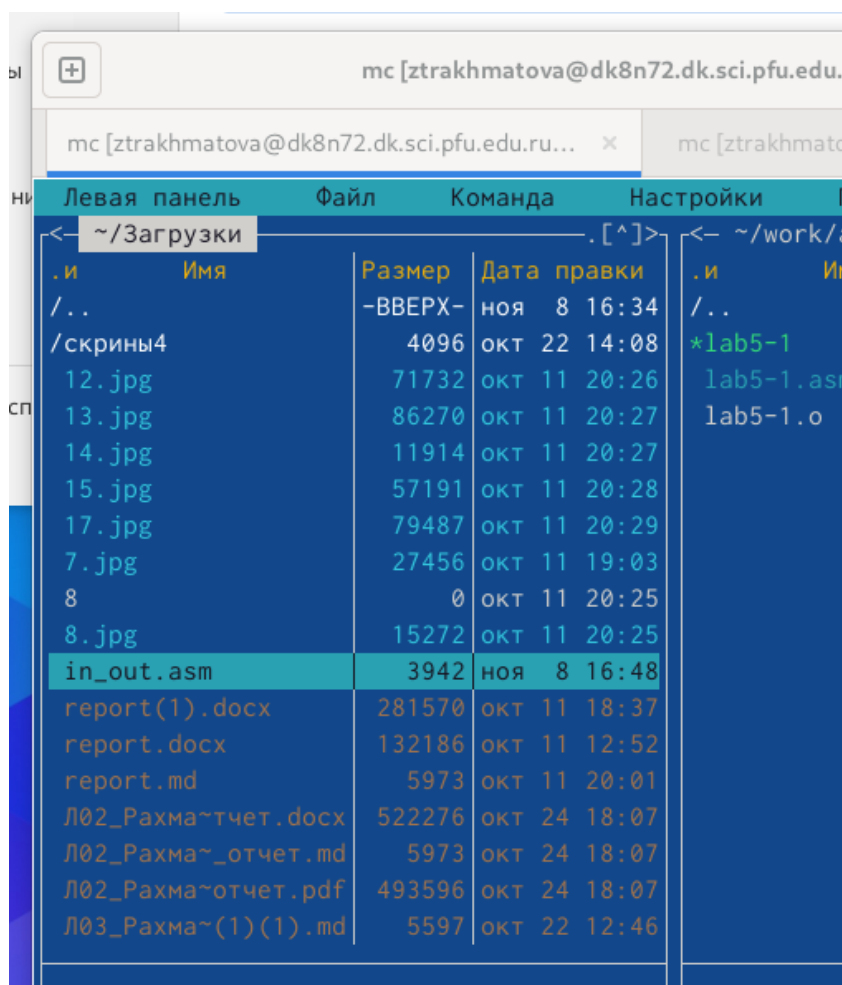


Рис. 4.8: Скачанный файл

С помощью функциональной клавиши F5 копирую файл `in_out.asm` из каталога Загрузки в созданный каталог `lab05` (рис. 4.9).

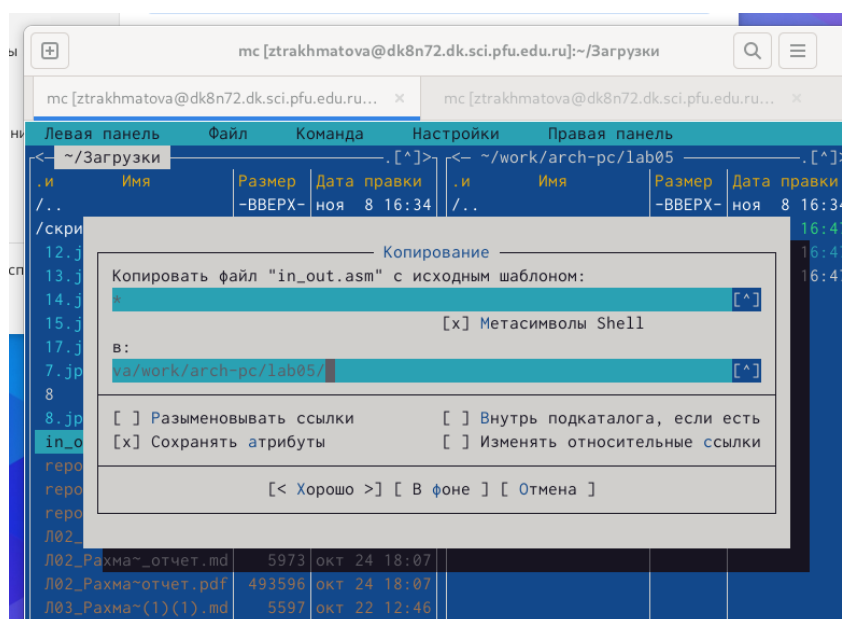


Рис. 4.9: Копирование файла

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. 4.10).

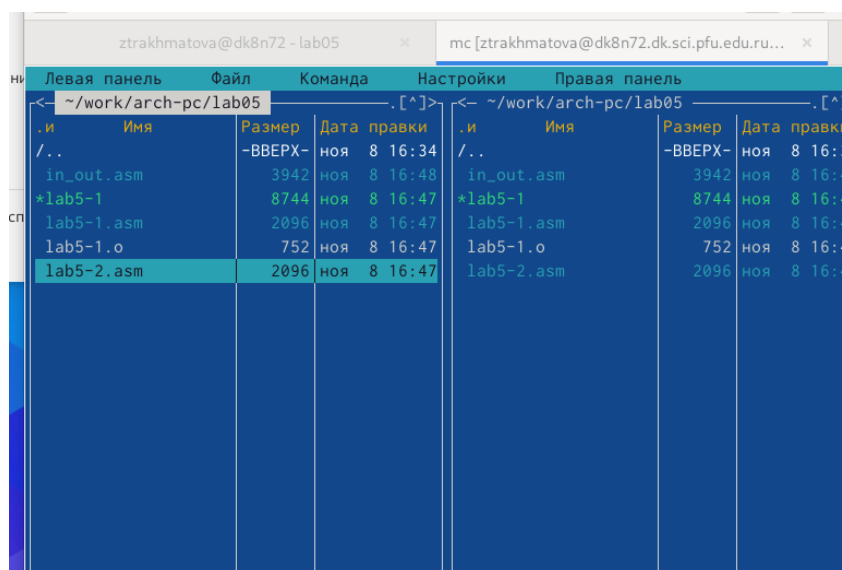
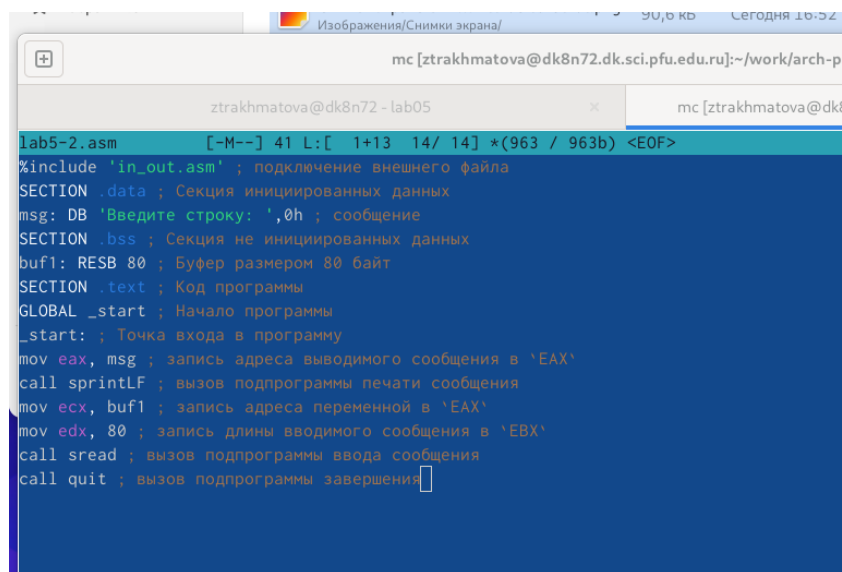


Рис. 4.10: Копирование файла

Изменяю содержимое файла lab5-2.asm во встроенном редакторе nano (рис.

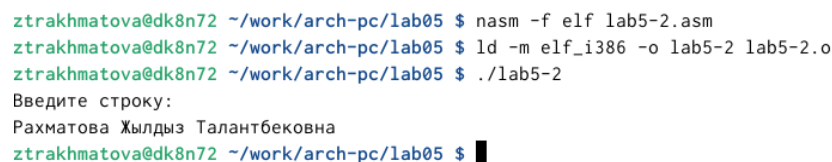
4.11), чтобы в программе использовались подпрограммы из внешнего файла in_out.asm.



```
lab5-2.asm [-M--] 41 L:[ 1+13 14/ 14] *(963 / 963b) <EOF>
#include 'in_out.asm' ; подключение внешнего файла
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 'EAX'
mov edx, 80 ; запись длины вводимого сообщения в 'EBX'
call read ; вызов подпрограммы ввода сообщения
call quit ; вызов подпрограммы завершения
```

Рис. 4.11: Редактирование файла

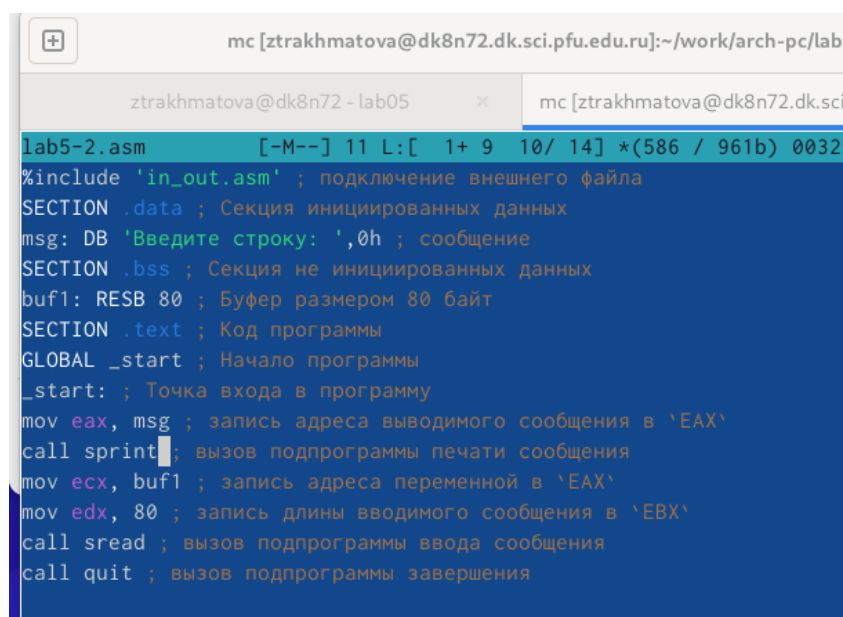
Транслирую текст программы файла в объектный файл командой `nasm -f elf lab5-2.asm`. Создался объектный файл `lab5-2.o`. Выполняю компоновку объектного файла с помощью команды `ld -m elf_i386 -o lab5-2 lab5-2.o`. Создался исполняемый файл `lab5-2`. Запускаю исполняемый файл (рис. 4.12).



```
ztrakhmatova@dk8n72 ~/work/arch-pc/lab05 $ nasm -f elf lab5-2.asm
ztrakhmatova@dk8n72 ~/work/arch-pc/lab05 $ ld -m elf_i386 -o lab5-2 lab5-2.o
ztrakhmatova@dk8n72 ~/work/arch-pc/lab05 $ ./lab5-2
Введите строку:
Рахматова Жылдыз Талантбековна
ztrakhmatova@dk8n72 ~/work/arch-pc/lab05 $
```

Рис. 4.12: Исполнение файла

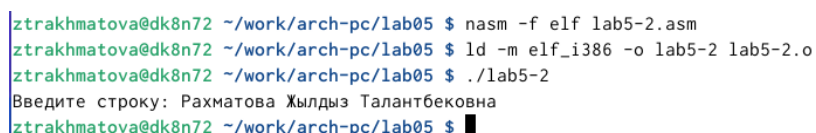
Открываю файл `lab5-2.asm` для редактирования в nano функциональной клавишей F4. Изменяю в нем подпрограмму `sprintf` на `sprint`. Сохраняю изменения и открываю файл для просмотра, чтобы проверить сохранение действий (рис. 4.13).



```
mc [ztrakhmatova@dk8n72.dk.sci.pfu.edu.ru]:~/work/arch-pc/lab
ztrakhmatova@dk8n72 - lab05
lab5-2.asm [-M--] 11 L:[ 1+ 9 10/ 14] *(586 / 961b) 0032
#include 'in_out.asm' ; подключение внешнего файла
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 'EBX'
call sread ; вызов подпрограммы ввода сообщения
call quit ; вызов подпрограммы завершения
```

Рис. 4.13: Отредактированный файл

Снова транслирую файл, выполняю компоновку созданного объектного файла, запускаю новый исполняемый файл (рис. 4.14).



```
ztrakhmatova@dk8n72 ~/work/arch-pc/lab05 $ nasm -f elf lab5-2.asm
ztrakhmatova@dk8n72 ~/work/arch-pc/lab05 $ ld -m elf_i386 -o lab5-2 lab5-2.o
ztrakhmatova@dk8n72 ~/work/arch-pc/lab05 $ ./lab5-2
Введите строку: Рахматова Жылдыз Талантбековна
ztrakhmatova@dk8n72 ~/work/arch-pc/lab05 $
```

Рис. 4.14: 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 исполняется 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. 4.15).

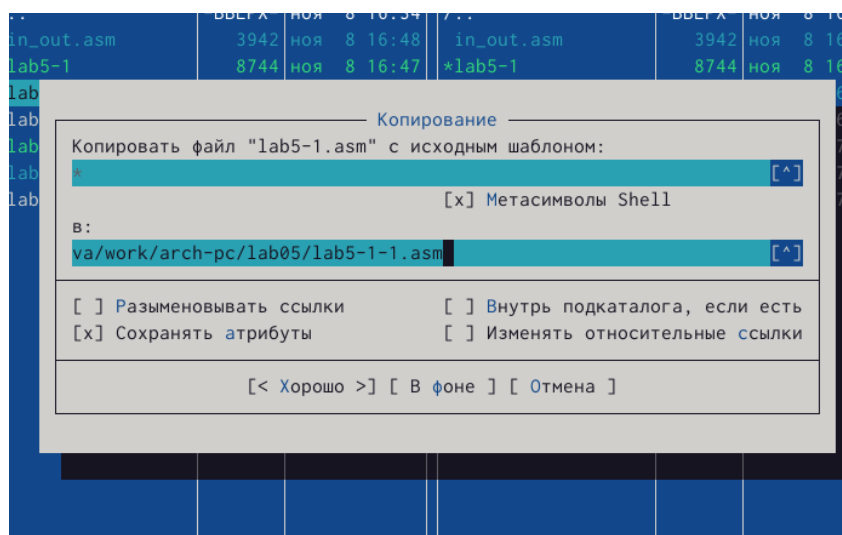


Рис. 4.15: Копирование файла

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. 4.16).

```

lab5-1-1.asm  [-M--] 20 L:[ 9+28 37/ 37] *(2376/2376b) <EOF>
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,4 ; Системный вызов для выхода (sys_exit)
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
1Помощь 2Сохран 3Блок 4Замена 5Копия 6Пер-ть 7Поиск 8Уда-ть 9МенюМС 10

```

Рис. 4.16: Редактирование файла

2. Создаю объектный файл lab5-1-1.o, отдаю его на обработку компоновщи-
ку, получаю исполняемый файл lab5-1-1, запускаю полученный исполняе-
мый файл. Программа запрашивает ввод, ввожу свои ФИО, далее програм-
ма выводит введенные мною данные (рис. 4.17).

```

ztrakhmatova@dk8n72 ~/work/arch-pc/lab05 $ nasm -f elf lab5-1-1.asm
ztrakhmatova@dk8n72 ~/work/arch-pc/lab05 $ ld -m elf_i386 -o lab5-1-1 lab5-1-1.o
ztrakhmatova@dk8n72 ~/work/arch-pc/lab05 $ ./lab5-1-1
Введите строку:
Рхматова Жылдыз Талантбековна
Рхматова Жылдыз Талантбековна
ztrakhmatova@dk8n72 ~/work/arch-pc/lab05 $

```

Рис. 4.17: 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 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 ; 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 ; Описание 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. 4.18).

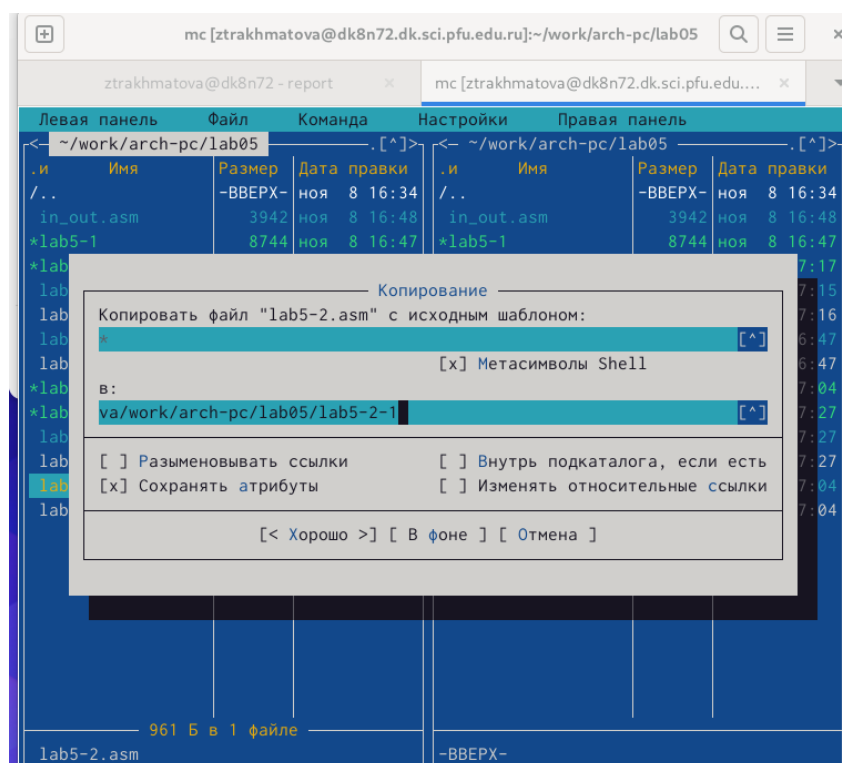


Рис. 4.18: Копирование файла

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).

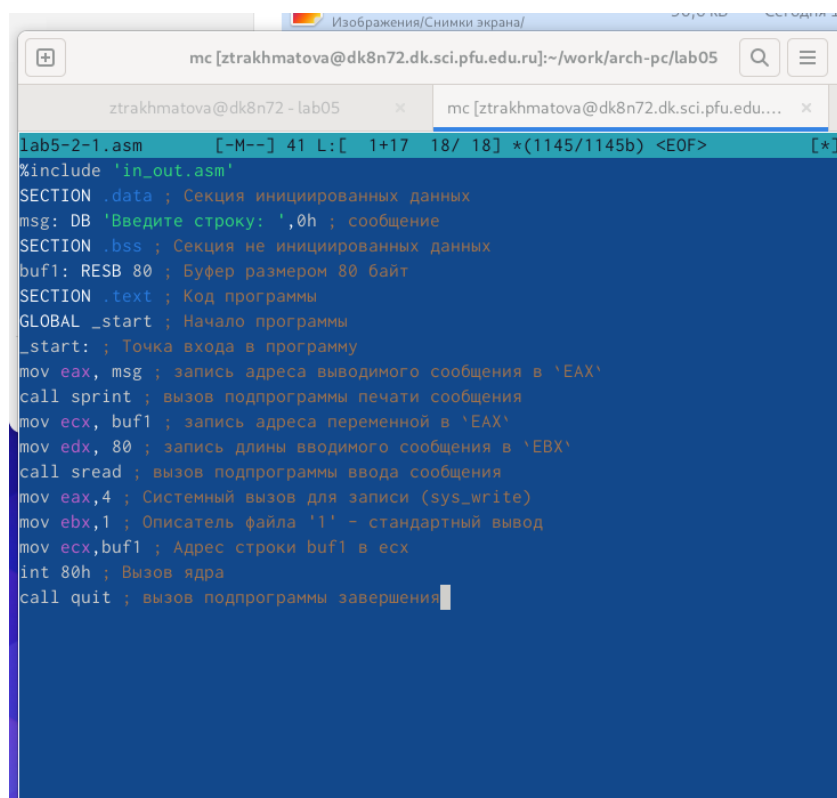


Рис. 4.19: Редактирование файла

4. Создаю объектный файл lab5-2-1.o, отдаю его на обработку компоновщи-
ку, получаю исполняемый файл lab5-2-1, запускаю полученный исполня-
емый файл. Программа запрашивает ввод без переноса на новую строку,
ввожу свои ФИО, далее программа выводит введенные мною данные (рис.
4.20).

```

ztrakhmatova@dk8n72 ~/work/arch-pc/lab05 $ nasm -f elf lab5-2-1.asm
ztrakhmatova@dk8n72 ~/work/arch-pc/lab05 $ ld -m elf_i386 -o lab5-2-1 lab5-2-1.o
ztrakhmatova@dk8n72 ~/work/arch-pc/lab05 $ ./lab5-2-1
Введите строку: Рахматова Жылдыз Талантбековна
Рахматова Жылдыз Талантбековна
ztrakhmatova@dk8n72 ~/work/arch-pc/lab05 $

```

Рис. 4.20: 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 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 ; Описание 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`.