

Лабораторная работа №5

Архитектура компьютера

Тойчубекова Асель Нурлановна

Содержание

1	Цель работы	5
2	Задание	6
3	Теоретическое введение	8
4	Выполнение лабораторной работы	9
4.1	Подключение внешнего файла in_out.asm	13
4.2	Задание для самостоятельной работы	17
5	Выводы	24
	Список литературы	25

Список иллюстраций

4.1	РИС.1 Открытый Midnight Commander	9
4.2	РИС.2 Перемещение между директориями	9
4.3	РИС.3 Создание папки lab5	10
4.4	РИС.3.1 Созданная папка lab5	10
4.5	РИС.4 Создание файла lab5-1.asm	10
4.6	РИС.4.1 Созданный файла lab5-1.asm	11
4.7	РИС.5 Редактирование файла	11
4.8	РИС.6 Проверка содержания файла	12
4.9	РИС.7 Компиляция и компоновка файла	12
4.10	РИС.8 Запуск программы	12
4.11	РИС.9 Скаченный файл	13
4.12	РИС.10 Копирование файла	14
4.13	РИС.10 Скопированный файл	14
4.14	РИС.11 Копирование файла lab5-1.asm с другим именем	15
4.15	РИС.12 Редактирование файла lab5-2.asm	15
4.16	РИС.13 Исполнение файла и запуск программы	16
4.17	РИС.14 Измененный файл	16
4.18	РИС.15 Исполнение нового файла и запуск программы	17
4.19	РИС.16 Копирование файла lab5-1.asm с именем lab5-1-1.asm	17
4.20	РИС.17 Редактирование программы	19
4.21	РИС.18 Исполнение файла и запуск программы	19
4.22	РИС.19 Копирование файла lab5-2.asm с именем lab5-2-1.asm	20
4.23	РИС.19 Скопированный файл lab5-2-1.asm	21
4.24	РИС.20 Редактирование программы	23
4.25	РИС.21 Исполнение файла и запуск программы	23

Список таблиц

1 Цель работы

Целью данной лабораторной работы является приобретение практических навыков работы в Midnight Commander. Также освоение инструкций языка ассемблера `mov` и `int`.

2 Задание

- Понять основы работы с Midnight Commander.
 - Познакомиться со структурой программы на языке ассемблера NASM:
 - Секции ассемблера NASM;
 - Директивы.
 - Изучить структуру инструкции языка ассемблера mov.
 - Изучить структуру инструкции языка ассемблера int.
 - Познакомиться с системными вызовами для обеспечения диалога с пользователями.
 - Научиться писать программу для вывода сообщения на экран и ввода строки с клавиатуры, используя пример.
 - Научиться подключать внешний файл in_out.asm и написать программу вывода сообщения на экран и ввода строки с клавиатуры с использованием файла in_out.asm.
 - Задание для самостоятельной работы
1. Создайте копию файла lab5-1.asm. Внесите изменения в программу (без использования внешнего файла in_out.asm), так чтобы она работала по следующему алгоритму:
 - вывести приглашение типа “Введите строку:”;
 - ввести строку с клавиатуры;
 - вывести введённую строку на экран
 2. Получите исполняемый файл и проверьте его работу. На приглашение вве-

сти строку введите свою фамилию.

3. Создайте копию файла lab5-2.asm. Исправьте текст программы с использование под- программ из внешнего файла in_out.asm, так чтобы она работала по следующему алгоритму:
 - вывести приглашение типа “Введите строку:”;
 - ввести строку с клавиатуры;
 - вывести введённую строку на экран.
4. Создайте исполняемый файл и проверьте его работу.

3 Теоретическое введение

Midnight Commander (mc, MC) – текстовая полнофункциональная программа, которая позволяет пользователю копировать, перемещать и удалять файлы и директории, производить поиск файлов и запускать на выполнение команды оболочки. Также в него встроен редактор и программа для просмотра файлов. В Midnight Commander используются функциональные клавиши, к которым привязаны часто выполняемые операции.

Программа на языке ассемблера NASM состоит из трех секций: секция кода программы-SECTION .text, секция инициализированных данных-SECTION .data, секция неинициализированных данных-SECTION .bss. Для объявления инициализированных и неинициализированных данных используют директивы (DB, DW, DD, DQ и DT; и resb, resw, resd).

Инструкция mov

Инструкция языка ассемблера mov предназначена для дублирования данных источника в приёмнике. В качестве операнда могут выступать регистры (register), ячейки памяти (memory) и непосредственно значение const.

Инструкция int

Инструкция языка ассемблера int предназначена для вызова прерывания с указанным номером. В общем виде она записывается в виде 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

Открывем Midnight Commander с помощью команды `mc`. Затем переходим в каталог `~/work/arch-рс`, созданный нами при выполнении лабораторной работы №4.(РИС.1) и (РИС.2)

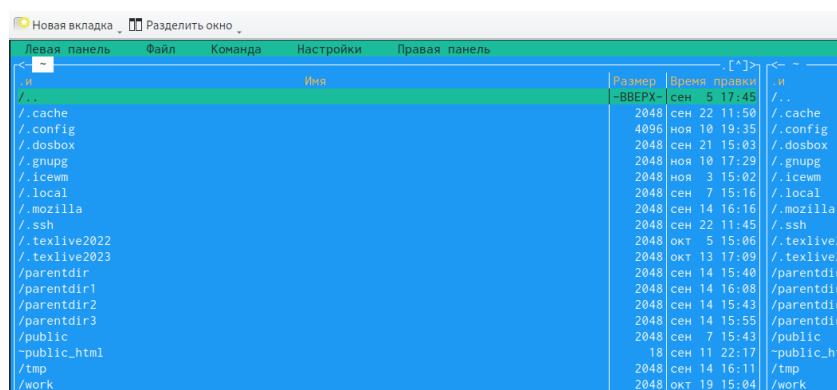


Рис. 4.1: РИС.1 Открытый Midnight Commander

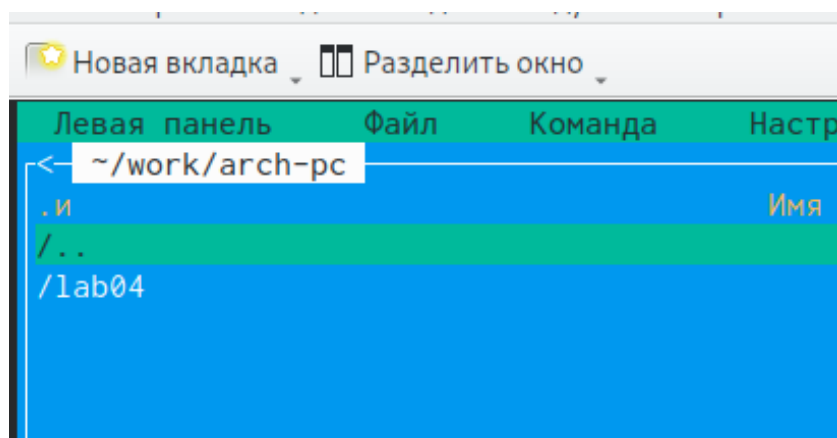


Рис. 4.2: РИС.2 Перемещение между директориями

С помощью функциональной клавиши F7 создаем каталог lab5 и перейдем в него. Мы видим, что каталог был удачно создан (РИС.3) и (РИС.3.1)

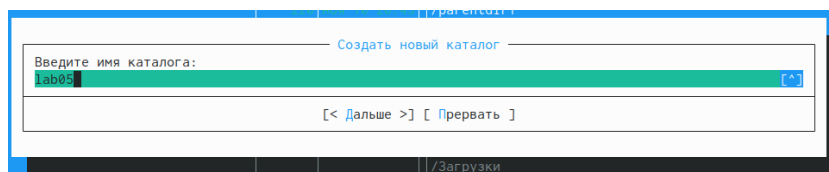


Рис. 4.3: РИС.3 Создание папки lab5

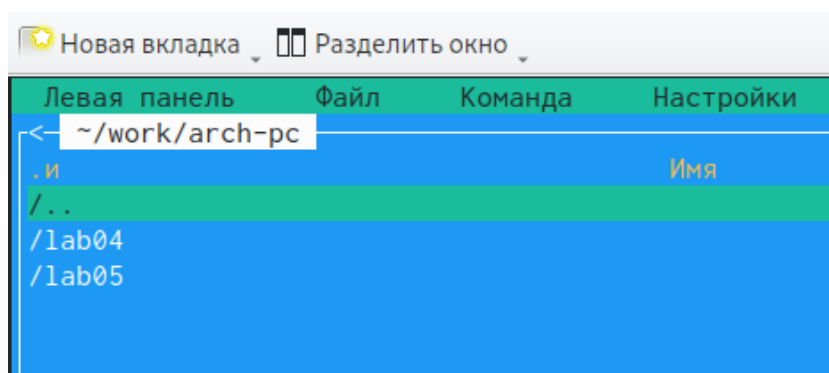


Рис. 4.4: РИС.3.1 Созданная папка lab5

Пользуясь строкой ввода и командой touch создам файл lab5-1.asm (РИС.4) и (РИС4.1)

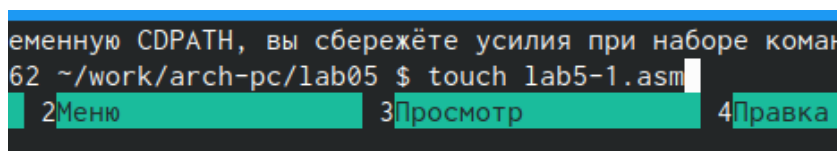


Рис. 4.5: РИС.4 Создание файла lab5-1.asm

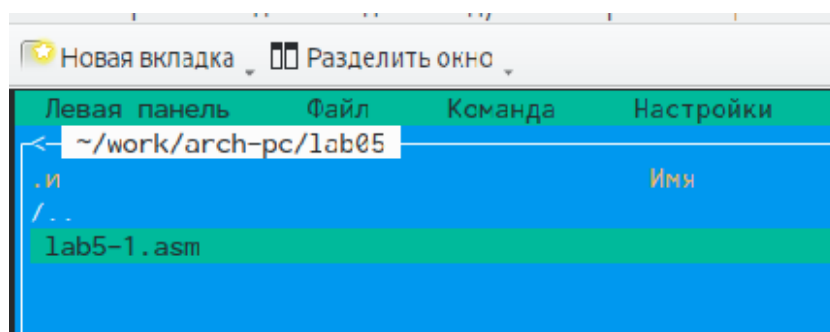


Рис. 4.6: РИС.4.1 Созданный файла lab5-1.asm

С помощью функциональной клавиши F4 открываем созданный файл для редактирования в редакторе nano. Затем введем в нее программу для вывода сообщения на экран и ввода строки с клавиатуры. Используя комбинацию клавиш Y,ENTER сохраняем изменения, и выходим из редактора с X,Ctrl.(РИС.5)

```
lab5-1.asm [----] 10 L: [ 1+25 26/ 29] *(290 / 301b) 0010 0x00A
SECTION .data
msg: DB "Введите число ",10

msgLen: EQU $-msg

SECTION .bss
buf1: RESB 80

SECTION .text
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. 4.7: РИС.5 Редактирование файла

Далее с клавишей F3 откроем файл для просмотра и видим, что файл содержит

текст программы.(РИС.6)

```
/afs/.dk.sci.pfu.edu.ru/home/a/n/antoyjchubekova/work/arch-pc/lab05/lab5-1.asm
SECTION .data
msg: DB 'Введите число:',10

msgLen: EQU $-msg

SECTION .bss
buf1: RESB 80

SECTION .text
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. 4.8: РИС.6 Проверка содержания файла

Оттранслируем текст программы lab5-1.asm в объектный файл используя команду “nasm -f elf lab5-1.asm” Далее выполним компоновку объектного файла с помощью компоновщика ld .(РИС.7)

```
antoyjchubekova@dk5n53 ~/work/arch-pc/lab05 $ nasm -f elf lab5-1.asm
antoyjchubekova@dk5n53 ~/work/arch-pc/lab05 $ ld -m elf_i386 -o lab5-1 lab5-1.o
```

Рис. 4.9: РИС.7 Компиляция и компоновка файла

Запускаем полученный исполняемый файл, после того как программа выведет строку “Введите строку:” и будет ждать ввода с клавиатуры введем наше ФИО.(РИС.8)

```
antoyjchubekova@dk5n53 ~/work/arch-pc/lab05 $ ./lab5-1
Введите строку:
Тойчубекова Асель Нурлановна
```

Рис. 4.10: РИС.8 Запуск программы

4.1 Подключение внешнего файла in_out.asm

Скачиваем файл in_out.asm со страницы курса в ТУИС. Зайдя в загрузки мы видим, что файл был сохранен в каталоге “Загрузки”(РИС.9)

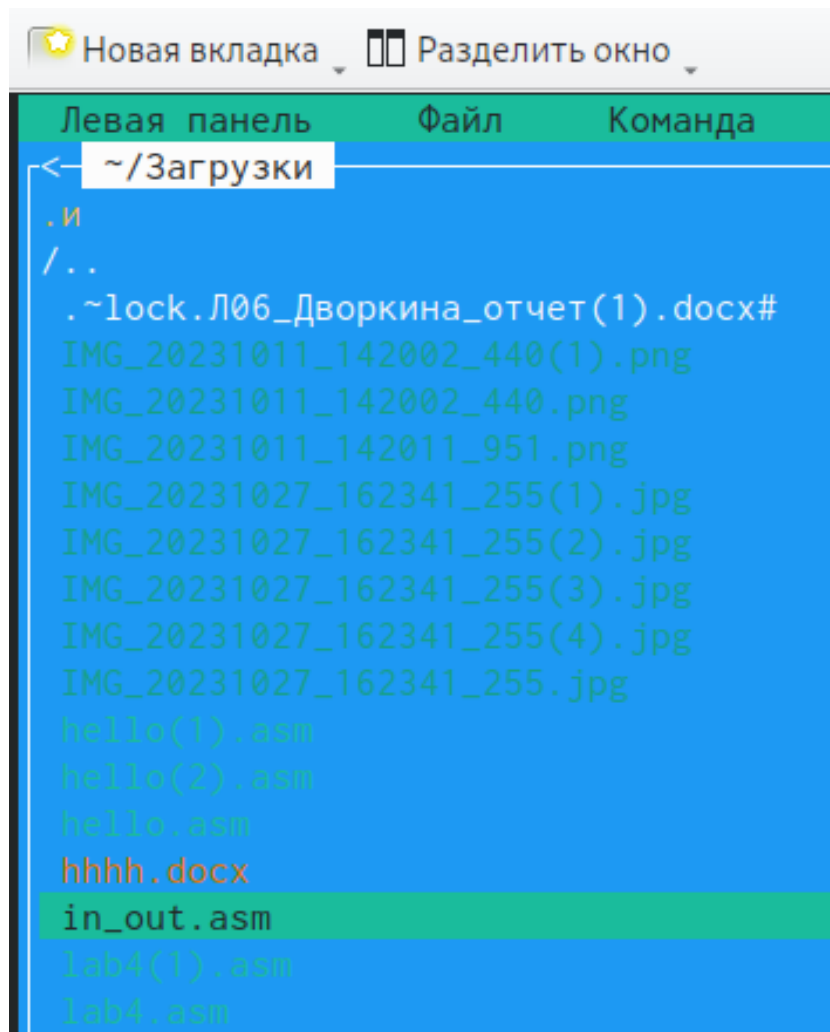


Рис. 4.11: РИС.9 Скаченный файл

С помощью функциональной клавиши F5 копируем файл in_out.asm в каталог с файлом lab5-1.asm.(РИС.10) и (РИС.10.1)

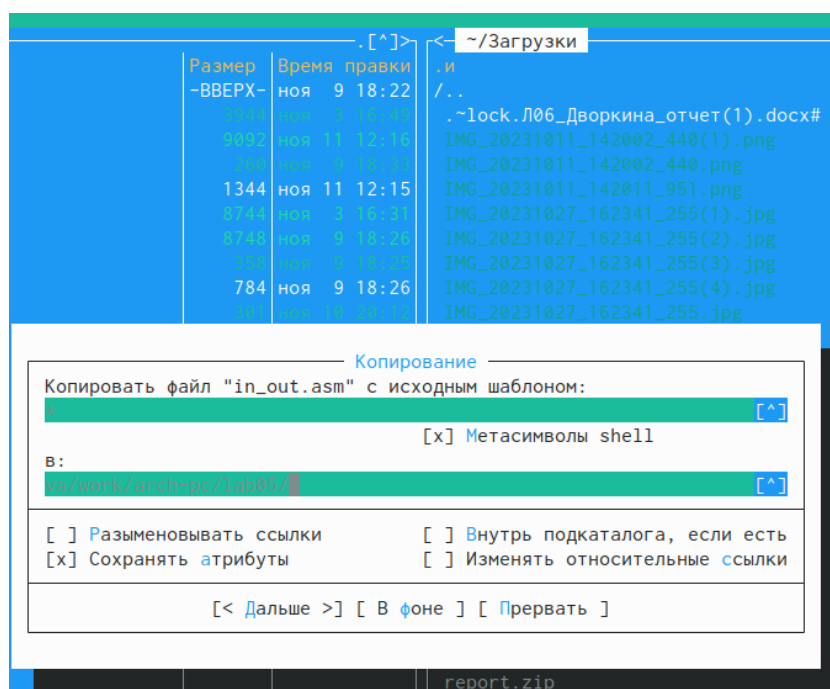


Рис. 4.12: РИС.10 Копирование файла

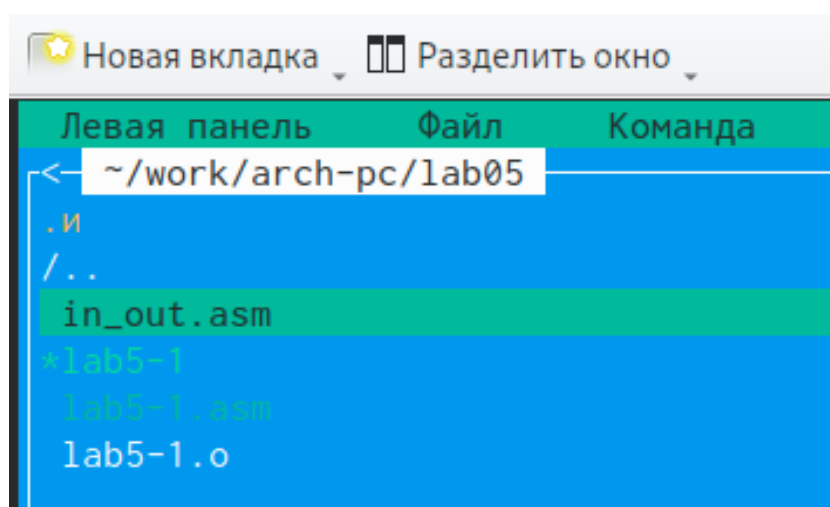


Рис. 4.13: РИС.10 Скопированный файл

Используя функциональную клавишу F6 создадим копию файла lab5-1.asm в этот же каталог, только с именем lab5-2.asm (РИС.11)

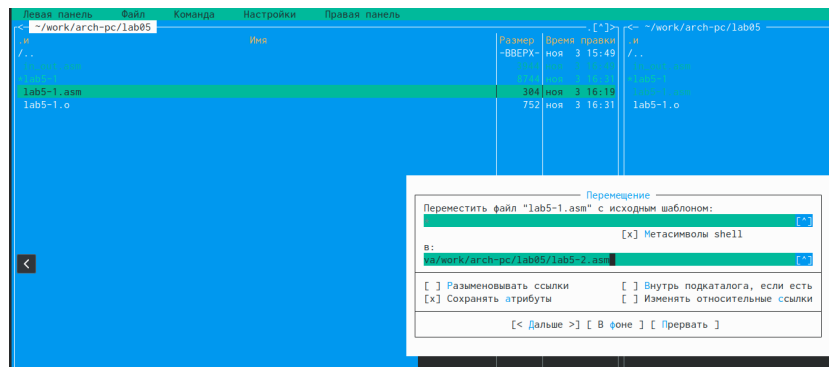


Рис. 4.14: РИС.11 Копирование файла lab5-1.asm с другим именем

Зайдем в файл lab5-2.asm и редактируем программу, которая в ней записана так, чтобы программа работала с помощью подпрограмм из внешнего файла in_out.asm.(РИС.12)

```
lab5-2.asm [-M--] 9 L: [ 1+20 21/ 21] *(222 / 222b) <EOF>
#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 sprintf

mov ecx, buf1
mov edx, 80

call sread

call quit
```

Рис. 4.15: РИС.12 Редактирование файла lab5-2.asm

Далее транслируем текст программы в объективный файл с помощью команды `nasm -f elf lab5-2.asm`. Созданный объективный файл передаю компоновщику `ld` и с помощью команды `ld -m elf_i386 -o lab5-2 lab5-2.o` создается исполняемый файл, который мы далее запускаем.(РИС.13)

```

antoychubekova@dk5n53 ~/work/arch-pc/lab05 $ nasm -f elf lab5-2.asm
antoychubekova@dk5n53 ~/work/arch-pc/lab05 $ ld -m elf_i386 -o lab5-2 lab5-2.o
antoychubekova@dk5n53 ~/work/arch-pc/lab05 $ ./lab5-2
Введите строку:
Тойчубекова Асель Нурлановна

```

Рис. 4.16: РИС.13 Исполнение файла и запуск программы

Открываем файл lab5-2 и редактируем программу заменяя подпрограмму `sprintLF` и `sprint`. Сохраняем изменения и открываем файл и видим, что изменения были сохранены(РИС.14)

```

lab5-2.asm [-M--] 9 L: [ 1+20 21/ 21] *(220 / 220b) <EOF>
#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. 4.17: РИС.14 Измененный файл

Транслируем файл и выполняем компоновку, затем запускаем исполняемый файл. На РИС.15 мы видим, что разница между двумя исполняемыми файлами заключается в том, что запуск первого запрашивает ввод с новой строки, а во втором случае запрашивает ввод без переноса на новую строку, так как в этом и заключается различие между подпрограммами `sprintLF` и `sprint`.


```

antoychubekova@dk5n53 ~/work/arch-pc/lab05 $ nasm -f elf lab5-2.asm
antoychubekova@dk5n53 ~/work/arch-pc/lab05 $ ld -m elf_i386 -o lab5-2 lab5-2.o
antoychubekova@dk5n53 ~/work/arch-pc/lab05 $ ./lab5-2
Введите строку: Тойчубекова Асель Нурлановна
antoychubekova@dk5n53 ~/work/arch-pc/lab05 $

```

Рис. 4.18: РИС.15 Исполнение нового файла и запуск программы

4.2 Задание для самостоятельной работы

1. Применяя функциональную клавишу F5 создаем копию файла lab5-1.asm с именем lab5-1-1.asm.(РИС.16)

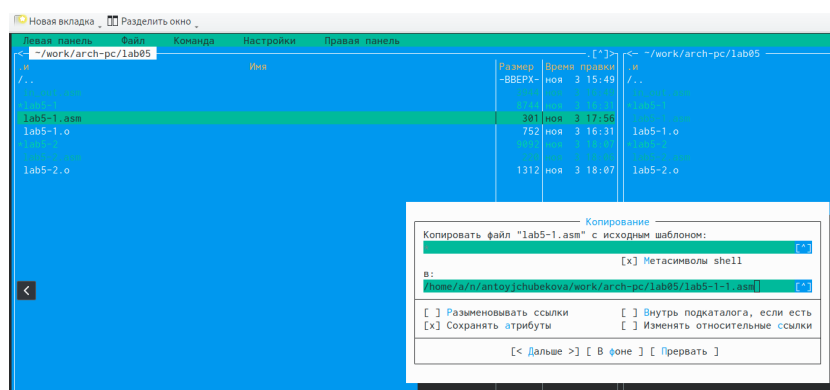


Рис. 4.19: РИС.16 Копирование файла lab5-1.asm с именем lab5-1-1.asm

Открываем этот файл с помощью функциональной клавиши F4 и редактируем программу так, чтобы кроме вывода приглашение типа “Введите строку:” программа выводила на экран введенную нами строку.(РИС.17) Измененная программа выглядит следующим образом:

```

SECTION .data
msg: DB 'Введите строку:',10
msgLen: EQU $-msg

```

```

SECTION .bss
buf1: RESB 80

```

```

SECTION .text
GLOBAL _start
_start:

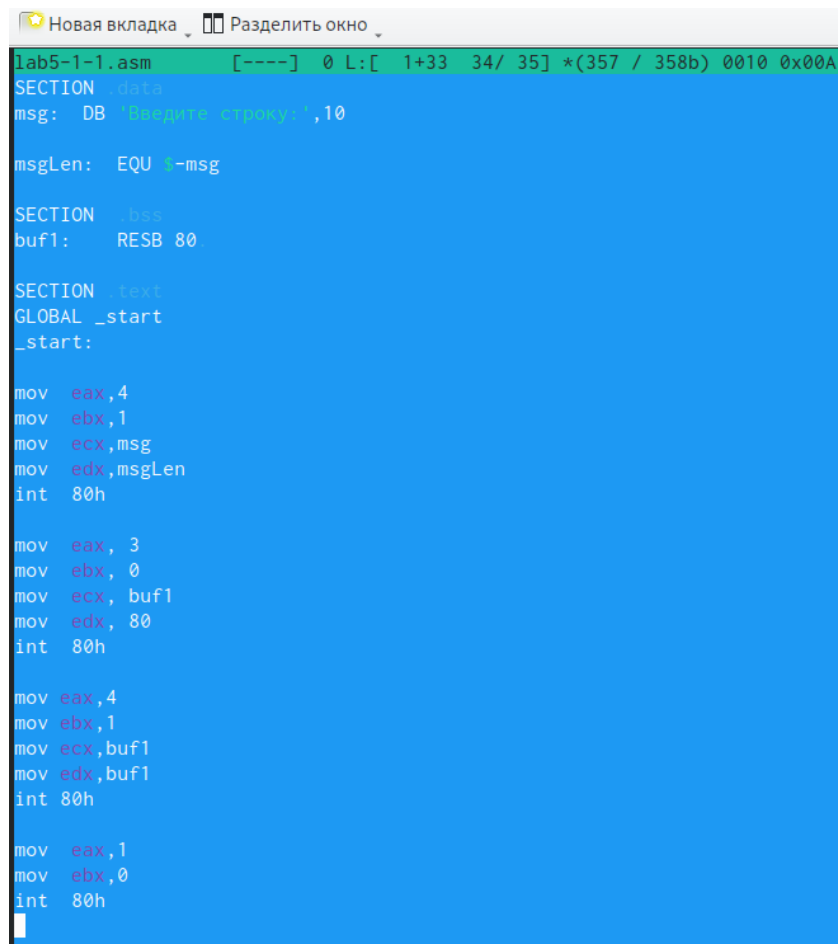
    mov eax,4
    mov ebx,1
    mov ecx,msg
    mov edx,msgLen
    int 80h

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

    mov eax,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
    mov ebx,0
    int 80h

```



```
lab5-1-1.asm [----] 0 L: [ 1+33 34/ 35] *(357 / 358b) 0010 0x00A
SECTION .data
msg: DB 'Введите строку:',10

msgLen: EQU $-msg

SECTION .bss
buf1: RESB 80

SECTION .text
GLOBAL _start
_start:

mov eax,4
mov ebx,1
mov ecx,msg
mov edx,msgLen
int 80h

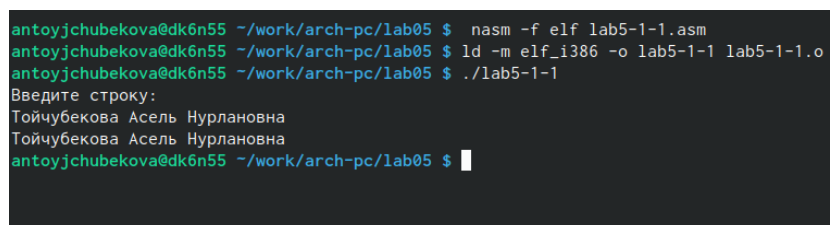
mov eax, 3
mov ebx, 0
mov ecx, buf1
mov edx, 80
int 80h

mov eax,4
mov ebx,1
mov ecx,buf1
mov edx,buf1
int 80h

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

Рис. 4.20: РИС.17 Редактирование программы

2. Транслируем файл и выполняем компоновку. Запустив исполняемый файл и введя свое ФИО на приглашение мы видим, что все выполняется правильно и выводит введенную нами строку.(РИС.18)



```
antoyjchubekova@dk6n55 ~/work/arch-pc/lab05 $ nasm -f elf lab5-1-1.asm
antoyjchubekova@dk6n55 ~/work/arch-pc/lab05 $ ld -m elf_i386 -o lab5-1-1 lab5-1-1.o
antoyjchubekova@dk6n55 ~/work/arch-pc/lab05 $ ./lab5-1-1
Введите строку:
Тойчубекова Асель Нурлановна
Тойчубекова Асель Нурлановна
antoyjchubekova@dk6n55 ~/work/arch-pc/lab05 $
```

Рис. 4.21: РИС.18 Исполнение файла и запуск программы

3. Также используя клавишу F5 создадим копию файла lab5-2.asm с именем lab5-2-1.asm. Можно заметить, что копия файла с успехом была создана (РИС.19)

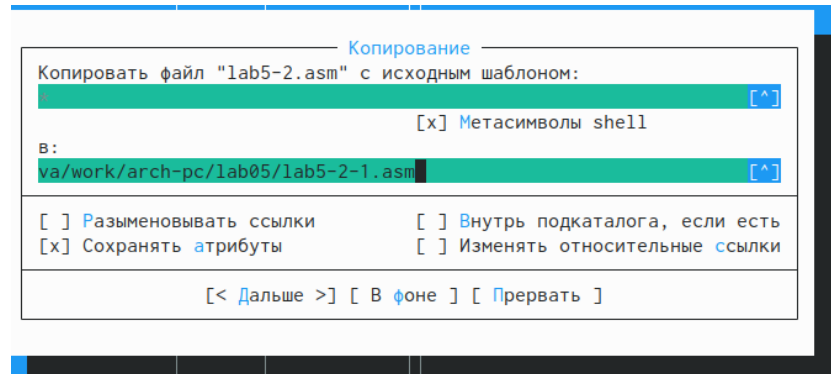


Рис. 4.22: РИС.19 Копирование файла lab5-2.asm с именем lab5-2-1.asm

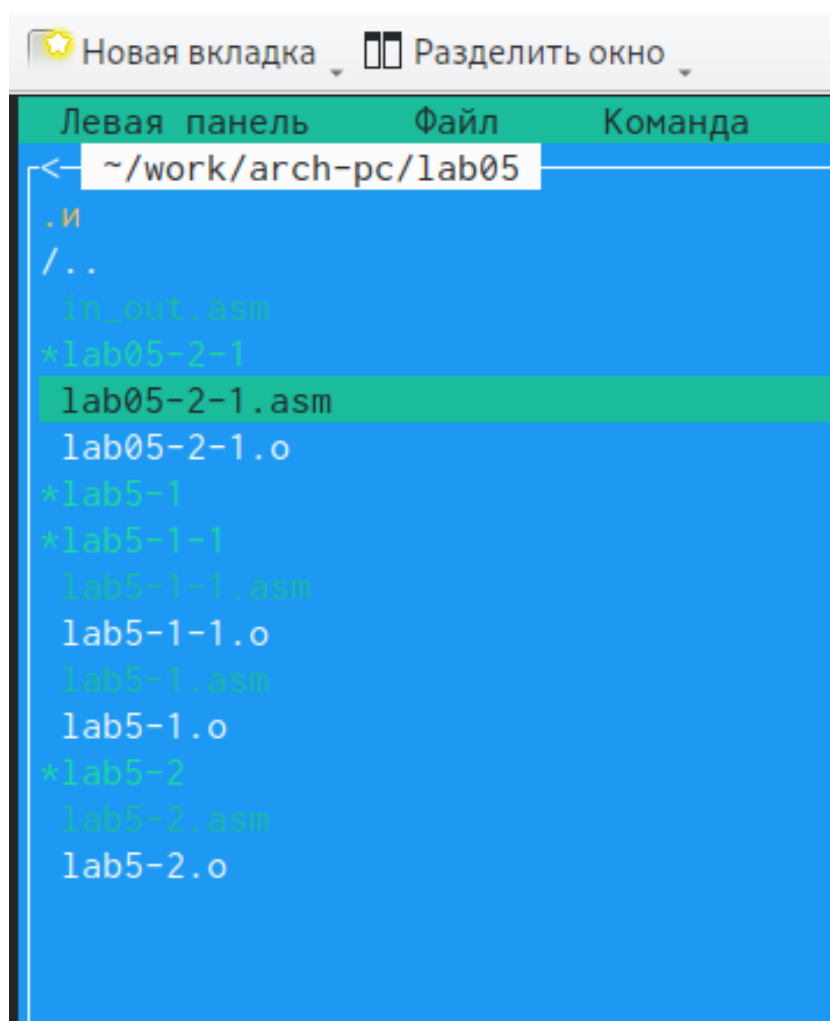


Рис. 4.23: РИС.19 Скопированный файл lab5-2-1.asm

С F4 открываю созданный файл и редактирую программу так, чтобы кроме вывода приглашение типа “Введите строку:” программа выводила на экран введенную нами строку.(РИС.20) Измененная программа выглядин слкдующим образом:

```
%include 'in_out.asm'
```

```
SECTION .data
```

```
msg: DB 'Введите строку:',0h
```

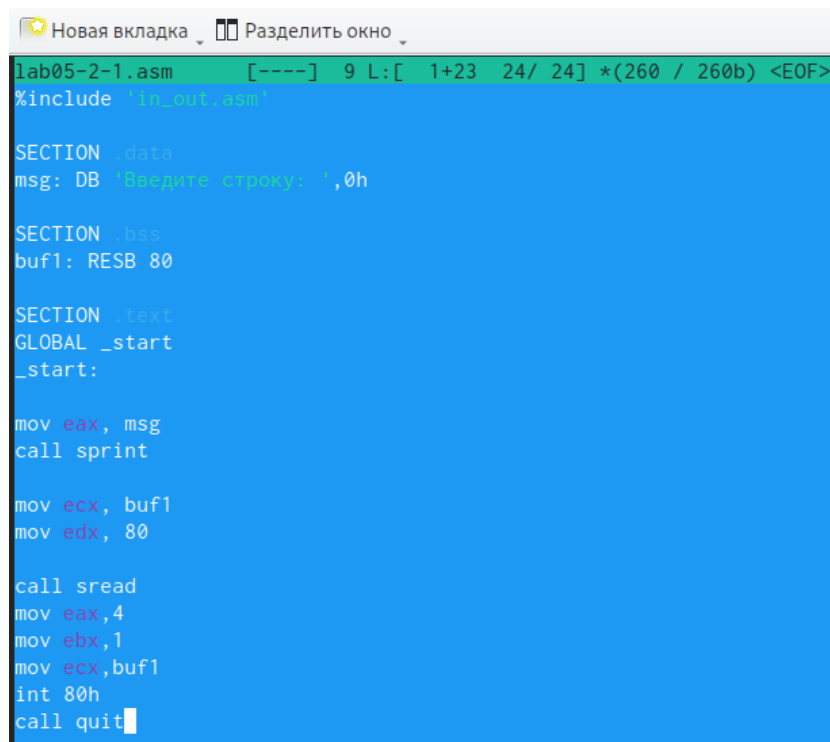
```
SECTION .bss
buf1: RESB 80
```

```
SECTION .text
GLOBAL _start
_start:
```

```
    mov eax, msg
    call sprint
```

```
    mov ecx, buf1
    mov edx, 80
```

```
    call sread
    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
```



```
lab05-2-1.asm [----] 9 L: [ 1+23 24/ 24] *(260 / 260b) <EOF>
#include "in_out.asm"

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

SECTION .bss
buf1: RESB 80

SECTION .text
GLOBAL _start
_start:

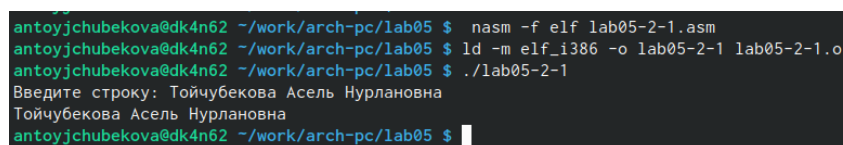
mov eax, msg
call sprint

mov ecx, buf1
mov edx, 80

call sread
mov eax, 4
mov ebx, 1
mov ecx, buf1
int 80h
call quit
```

Рис. 4.24: РИС.20 Редактирование программы

4. Транслируем отредактированный файл и выполняем компоновку. Запустив исполняемый файл мы видим, что программа запрашивает ввод без переноса на новую строку, введя свое ФИО, удостоверяемся, что программа выводит на экран введенные данные.(РИС.21)



```
antoyjchubekova@dk4n62 ~/work/arch-pc/lab05 $ nasm -f elf lab05-2-1.asm
antoyjchubekova@dk4n62 ~/work/arch-pc/lab05 $ ld -m elf_i386 -o lab05-2-1 lab05-2-1.o
antoyjchubekova@dk4n62 ~/work/arch-pc/lab05 $ ./lab05-2-1
Введите строку: Тойчубекова Асель Нурлановна
Тойчубекова Асель Нурлановна
antoyjchubekova@dk4n62 ~/work/arch-pc/lab05 $
```

Рис. 4.25: РИС.21 Исполнение файла и запуск программы

5 Выводы

В ходе лабораторной работы № 5 я приобрела практические навыки в работе с Midnight Commander. Вместе с тем освоила инструкции языка ассемблера `mov`, `int`. Используя полученные навыки написала программу вывода сообщения на экран и ввода строки с клавиатуры, также написала ту же программу, только с использованием файла `in_out.asm`. Также написала программу которая на второй раз выводит введенный мной текст с использованием файла `in_out.asm` и без нее.

Список литературы

- <https://esystem.rudn.ru/course/view.php?id=4975>
- <https://redos.red-soft.ru/base/manual/utilites/mc-filemanager>
- <https://dzen.ru/list/gadgets/linux-mc-commander>.