

РОССИЙСКИЙ УНИВЕРСИТЕТ ДРУЖБЫ НАРОДОВ

Факультет физико-математических и естественных наук

Кафедра прикладной информатики и теории вероятностей

ОТЧЕТ

ПО ЛАБОРАТОРНОЙ РАБОТЕ № 5

дисциплина: *Архитектура компьютера*

Студент: Лисенков Е.Р.

Группа: НКАбд-03-23

МОСКВА

2023 г.

О г л а в л е н и е

Цель работы	4
Задание	5
Теоретическое введение	6
Выполнение лабораторной работы	8
1 Основы работы с тс	8
2 Структура программы на языке ассемблера NASM	9
3 Подключение внешнего файла	11
4 Выполнение заданий	13
Выводы	16
Список литературы	17

Список иллюстраций

Изображение 1	Открытый Midnight Commander.....	8
Изображение 2	Выполню переход.....	8
Изображение 3	Создание каталога.....	9
Изображение 4	Открытие файла для редактирования.....	9
Изображение 5	Редактирование файла.....	10
Изображение 6	Просмотр текста программы.....	10
Изображение 7	Трансляция и запуск.....	11
Изображение 8	Скачанный файл.....	11
Изображение 9	Копирование файла.....	11
Изображение 10	Компоновка и запуск программы.....	12
Изображение 11	Создаю копию файла.....	13
Изображение 12	Изменения в коде.....	13
Изображение 13	Компоновка и запуск.....	14
Изображение 14	Создадим копию.....	14
Изображение 15	Изменение кода.....	15
Изображение 16	Компоновка и запуск программы.....	15

Цель работы

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

Задание

1. Основы работы с mc
2. Структура программы на языке ассемблера NASM
3. Подключение внешнего файла
4. Выполнение заданий для самостоятельной работы

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 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

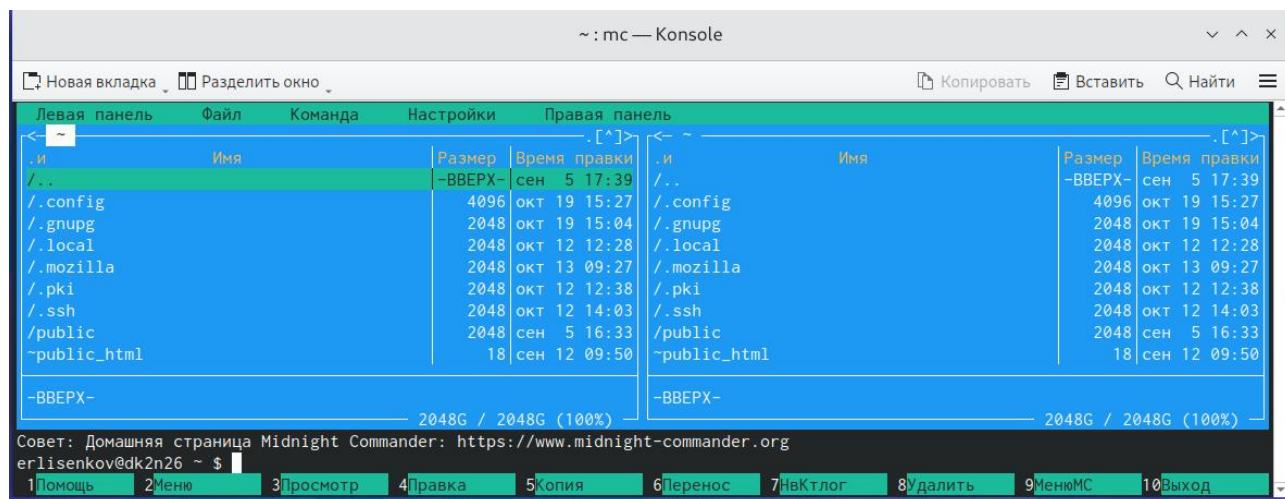
Здесь n — номер прерывания, принадлежащий диапазону 0–255. При

программировании в Linux с использованием вызовов ядра `sys_calls` `n=80h` (принято задавать в шестнадцатеричной системе счисления).

Выполнение лабораторной работы

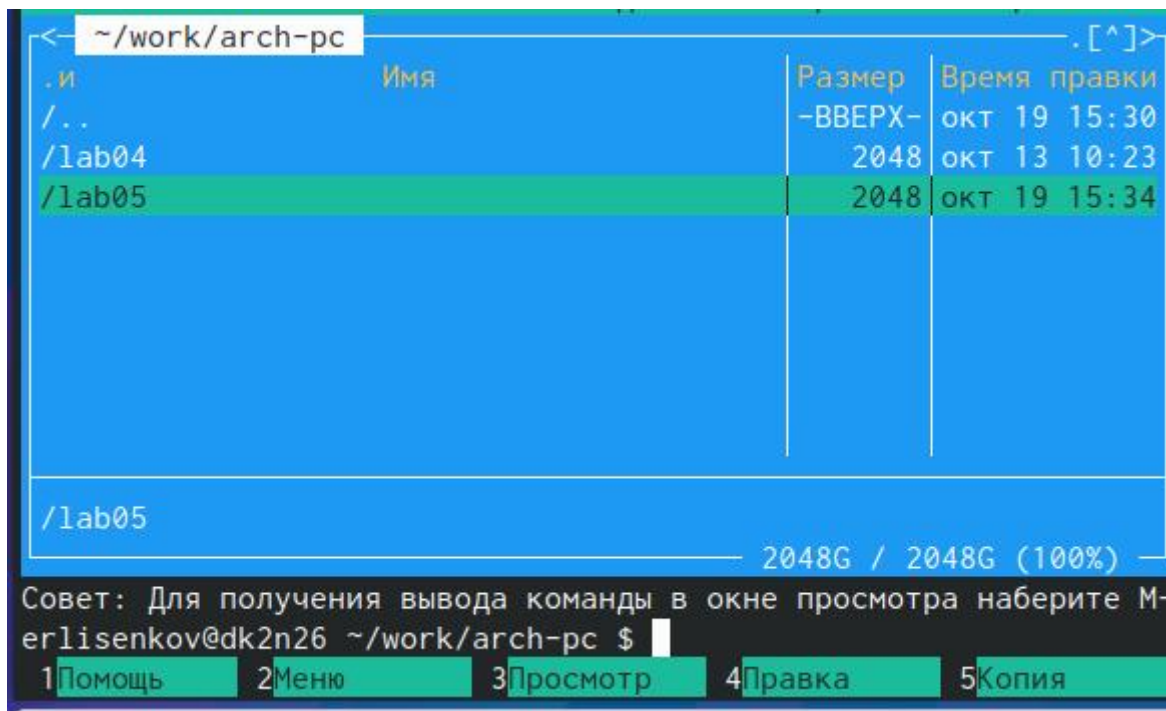
1. Основы работы с mc

Открываю Midnight Commander с помощью команды *mc* (Рис 1.).



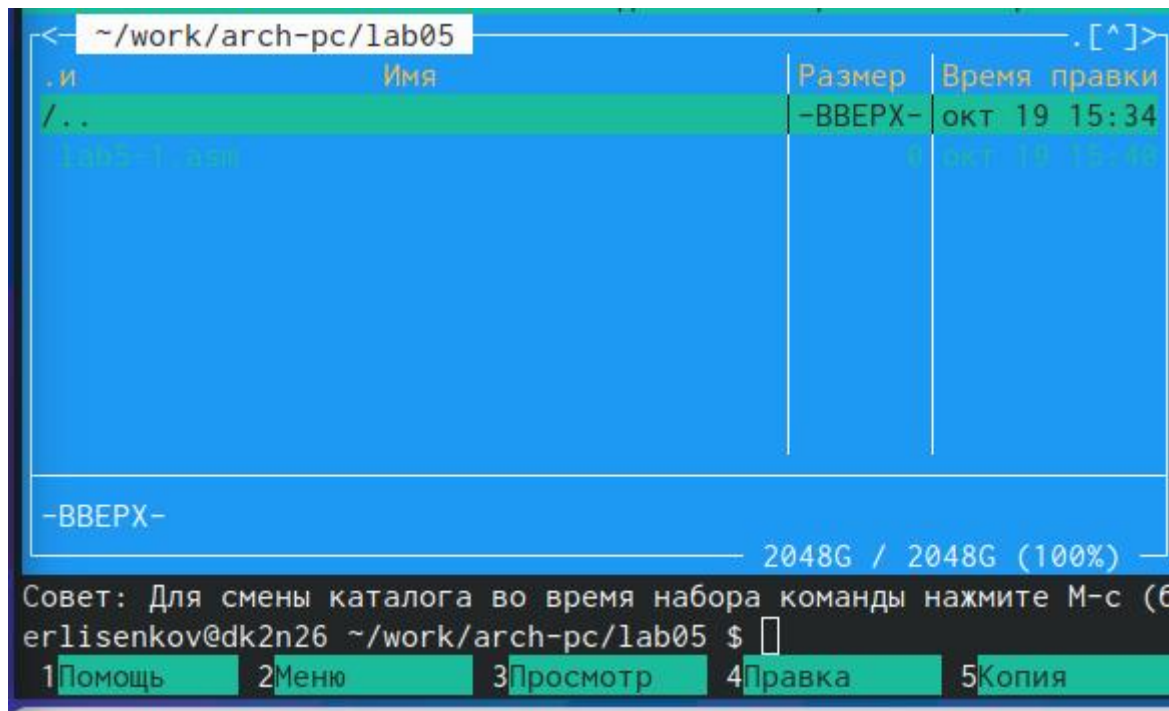
Изображение 1 | Открытый Midnight Commander

Перехожу в каталог `~/work/arch-pc` (Рис. 2).



Изображение 2 | Выполню переход

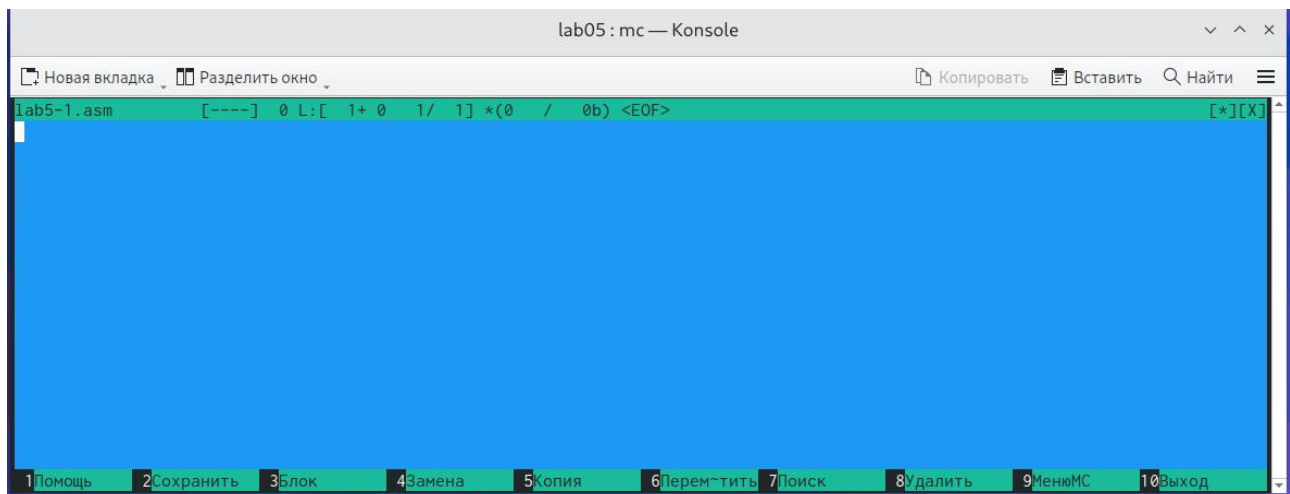
С помощью функциональной клавиши F7 создаю папку `lab05`, прописываю команду `touch lab5-1.asm` (Рис. 3).



Изображение 3 | Создание каталога

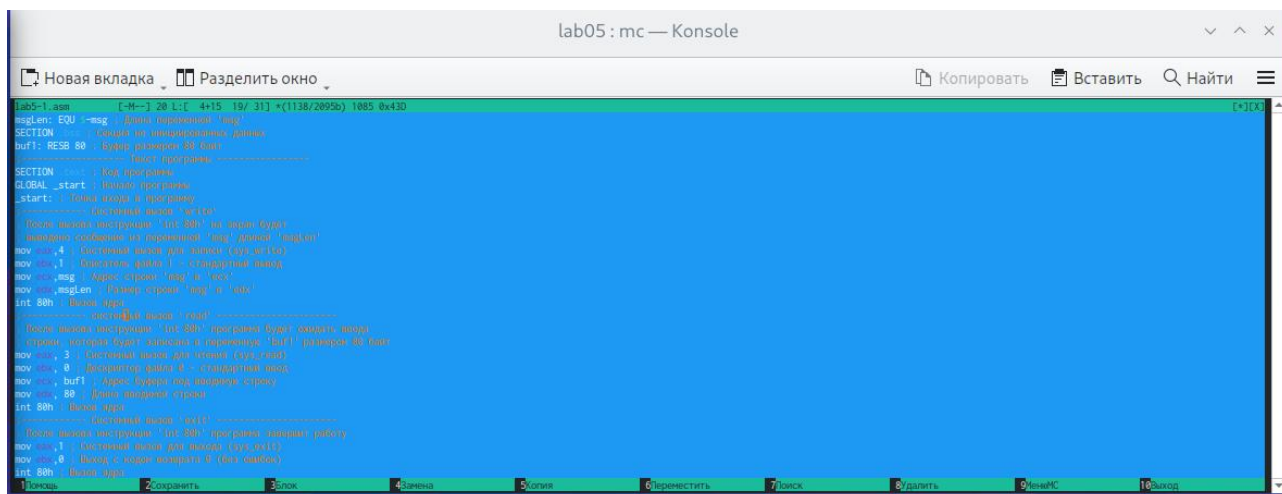
2. Структура программы на языке ассемблера NASM

С помощью функциональной клавиши F4 открою файл для редактирования в редакторе nano (Рис. 4).



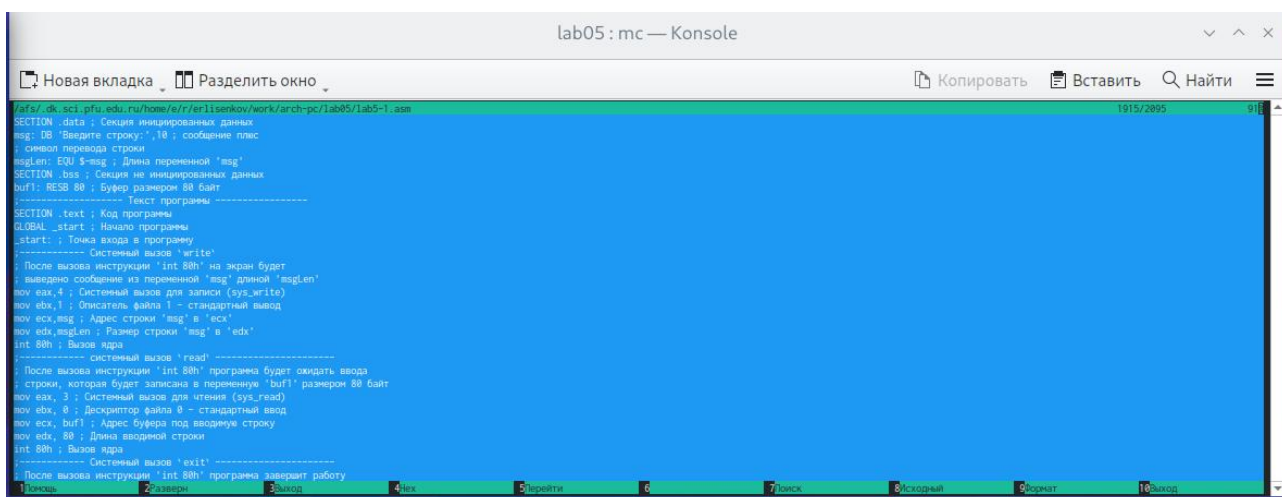
Изображение 4 | Открытие файла для редактирования

Введу в файл код программы для запроса строки у пользователя (Рис. 5).



Изображение 5 | Редактирование файла

Функциональная клавиша **F3** поможет нам открыть файл для просмотра, чтобы проверить содержание на наличие текста программы (Рис. 6).



Изображение 6 | Просмотр текста программы

Выполню трансляцию своей программы с помощью команд и запущу её:

nasm -f elf lab5-1.asm (транслировал текст файла в объектный файл)

ld -m elf_i386 -o lab5-1 lab5-1.o (выполню компоновку файла).

./lab5-1 (Запускаю программу этой командой)

```

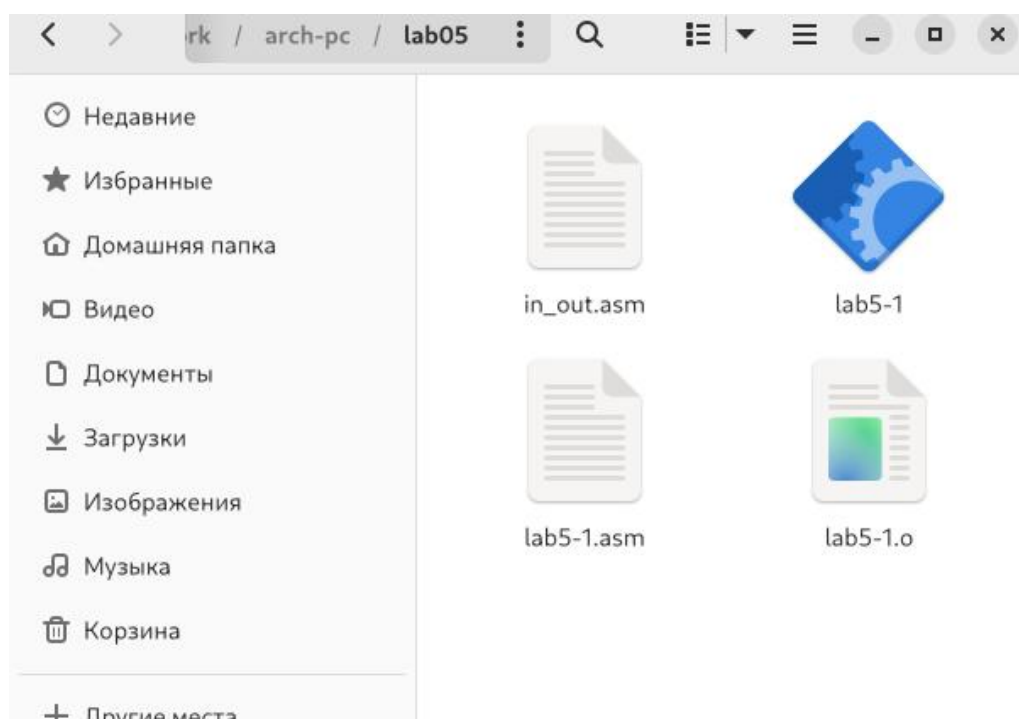
erlisenkov@dk1n22 ~/work/arch-pc/lab05 $ nasm -f elf lab5-1.asm
erlisenkov@dk1n22 ~/work/arch-pc/lab05 $ ld -m elf_i386 -o lab5-1 lab5-1.o
erlisenkov@dk1n22 ~/work/arch-pc/lab05 $ ./lab5-1
Введите строку:
Лисенков Егор Романович
erlisenkov@dk1n22 ~/work/arch-pc/lab05 $ 

```

Изображение 7 | Трансляция и запуск

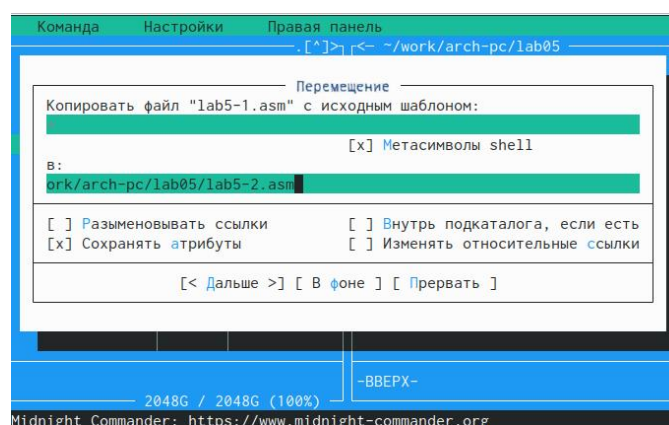
3. Подключение внешнего файла

На ТУИС загружаю файл и проверяю, что он был загружен в папку загрузки (Рис. 8).



Изображение 8 | Скачанный файл

Выполню копирование файла с помощью клавиши F5 in_out.asm из каталога Загрузки в lab06 (Рис. 9).



Изображение 9 | Копирование файла

Исправляю текст программы и включаю в работу подпрограммы из внешнего файла `in_out.asm`. Для изменений используем редактор `naпo`. Я извиняюсь, но я, к сожалению забыл сделать ТОЛЬКО этот скриншот. Прикрепляю код, который должен быть тут:

```
%include 'in_out.asm'      ; подключение внешнего файла

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

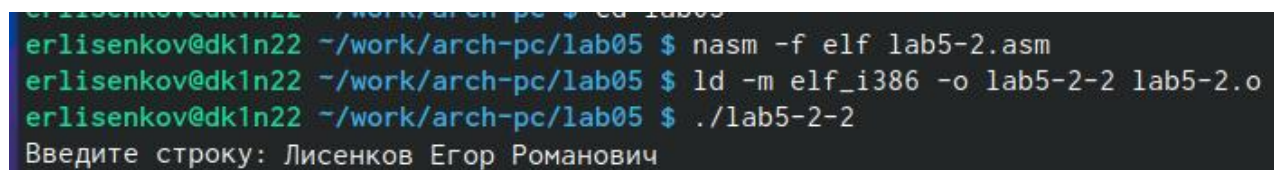
mov ecx, buf1             ; запись адреса переменной в `EAX`

mov edx, 80               ; запись длины вводимого сообщения в `EBX`

call sread                ; вызов подпрограммы ввода сообщения

call quit                 ; вызов подпрограммы завер
```

Выполню компоновку кода и запущу программу (Рис. 10).



```
erlisenkov@dk1n22 ~/work/arch-pc/lab05 $ nasm -f elf lab5-2.asm
erlisenkov@dk1n22 ~/work/arch-pc/lab05 $ ld -m elf_i386 -o lab5-2-2 lab5-2.o
erlisenkov@dk1n22 ~/work/arch-pc/lab05 $ ./lab5-2-2
Введите строку: Лисенков Егор Романович
```

Изображение 10 | Компоновка и запуск программы

Разница заключается в том, что программа запрашивает текст без переноса на новую строку (как было в прошлой программе).

4. Выполнение заданий

№1 Создам копию файла `lab5-1.asm` (использую клавишу F5). (Рис. 11)

Копирование

Копировать файл "lab5-1.asm" с исходным шаблоном:

[^]

[x] Метасимволы shell

В:

v/work/arch-pc/lab05/lab5-1-1.asm [^]

<input type="checkbox"/> Разыменовывать ссылки <input checked="" type="checkbox"/> Сохранять атрибуты	<input type="checkbox"/> Внутрь подкаталога, если есть <input type="checkbox"/> Изменять относительные ссылки
--	--

[< Дальше >]
[В фоне]
[Прервать]

Изображение 11 | Создаю копию файла

Выполню изменения в коде (с помощью клавиши F4) (Рис. 12).

```

lab5-1-1.asm      [-M--] 33 L:[ 1+ 0  1/ 26] *(49 /1550b) 1085 0x43D
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 вывод 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

```

1Помощь
2Сохранить
3Блок
4Замена
5Копия
6Переместить
7Поиск

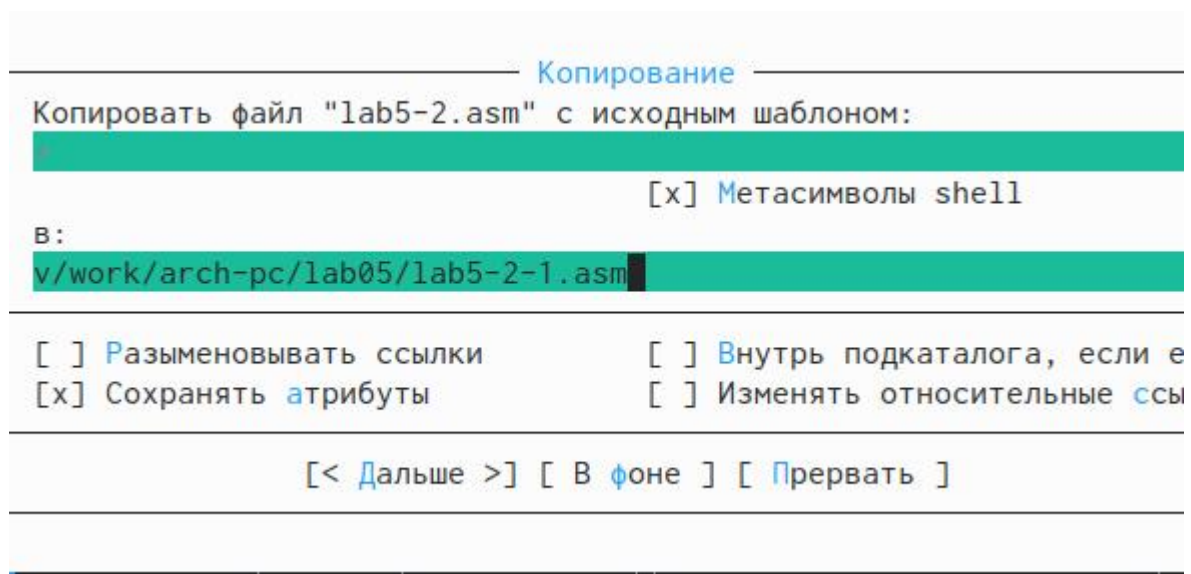
Изображение 12 | Изменения в коде

№2 Выполняем компоновку и запускаем программу. (Рис. 13)

```
erlisenkov@dk1n22 ~ $ cd work
erlisenkov@dk1n22 ~/work $ cd arch-pc
erlisenkov@dk1n22 ~/work/arch-pc $ cd lab05
erlisenkov@dk1n22 ~/work/arch-pc/lab05 $ nasm -f elf lab5-1-1.asm
erlisenkov@dk1n22 ~/work/arch-pc/lab05 $ ld -m elf_i386 -o lab5-1-1 lab5-1-1.o
erlisenkov@dk1n22 ~/work/arch-pc/lab05 $ ./lab-1-1
bash: ./lab-1-1: Нет такого файла или каталога
erlisenkov@dk1n22 ~/work/arch-pc/lab05 $ ./lab5-1-1
Введите строку:
Лисенков Егор Романович
Лисенков Егор Романович
erlisenkov@dk1n22 ~/work/arch-pc/lab05 $
```

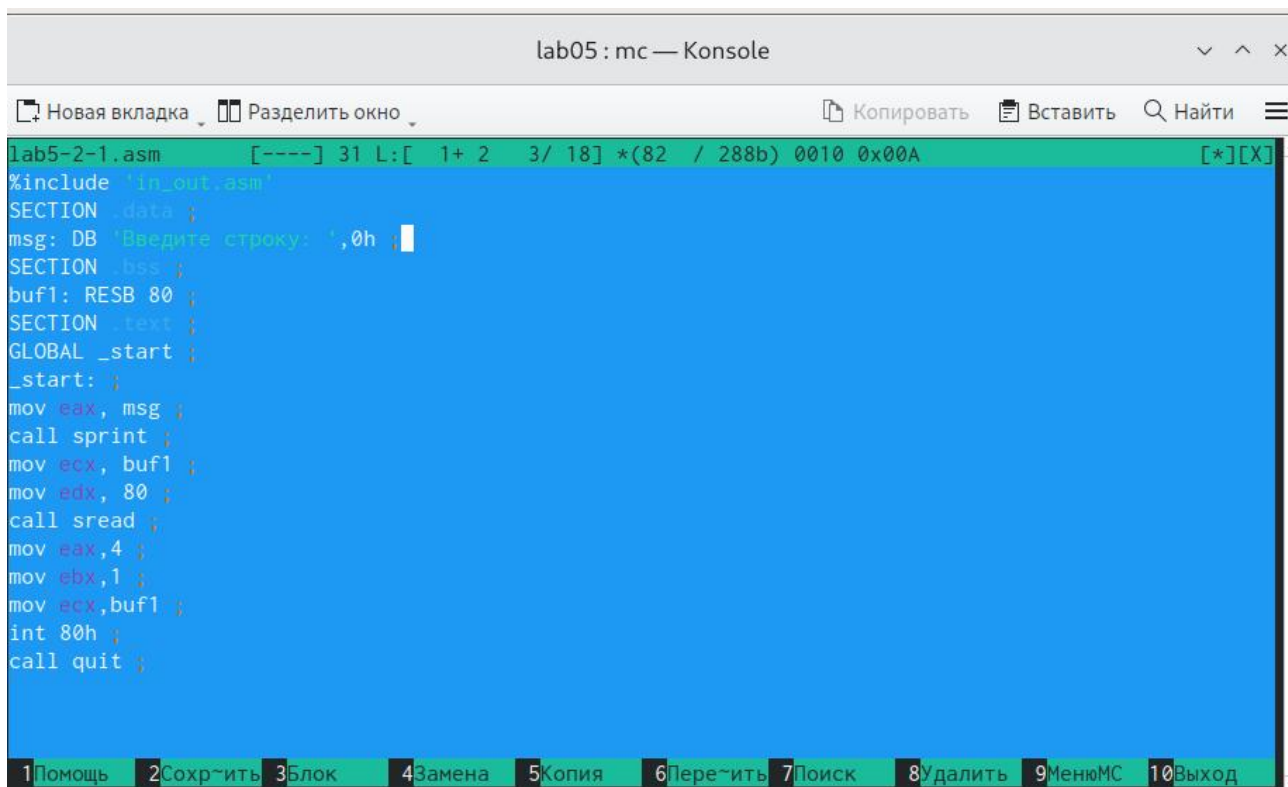
Изображение 13 | Компоновка и запуск

№3 Создам копию файла lab5-2.asm (используя клавишу F5). (Рис. 14)



Изображение 14 | Создадим копию

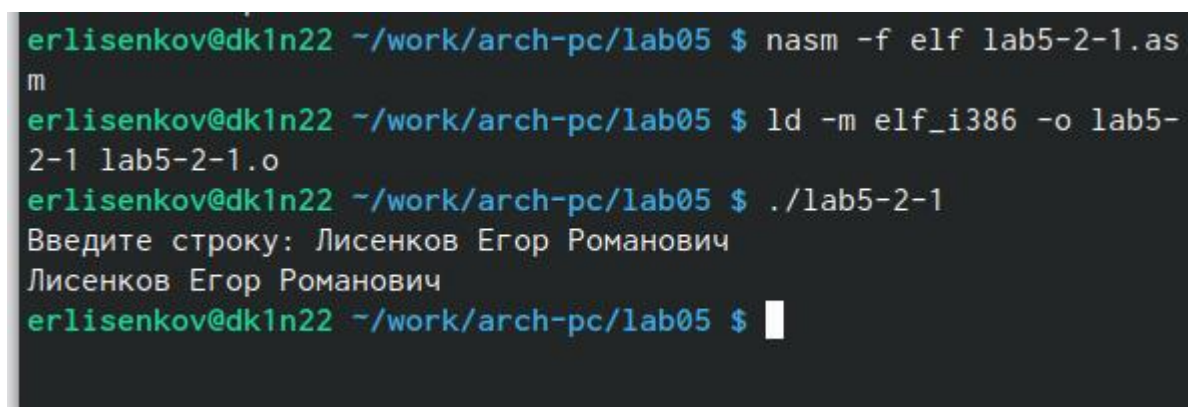
Исправляю код программы и включаю в работу внешний файл in_out.asm.
(Рис. 15).



```
lab5-2-1.asm [----] 31 L: [ 1+ 2 3/ 18] *(82 / 288b) 0010 0x00A [*][X]  
%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
```

Изображение 15 | Изменение кода

№4 Выполню компоновку и запускаю программу (Рис. 16).



```
erlisenkov@dk1n22 ~/work/arch-pc/lab05 $ nasm -f elf lab5-2-1.asm  
erlisenkov@dk1n22 ~/work/arch-pc/lab05 $ ld -m elf_i386 -o lab5-2-1 lab5-2-1.o  
erlisenkov@dk1n22 ~/work/arch-pc/lab05 $ ./lab5-2-1  
Введите строку: Лисенков Егор Романович  
Лисенков Егор Романович  
erlisenkov@dk1n22 ~/work/arch-pc/lab05 $
```

Изображение 16 | Компоновка и запуск программы

Выводы

По итогу выполнения этой лабораторной работы я смог приобрести практические знания, которые однозначно пригодятся мне в работе с языками программирования.

Список литературы

Лабораторная работа №5. Основы работы с Midnight Commander (mc). Структура программы на языке ассемблера NASM. Системные вызовы в ОС GNU Linux