ОТЧЕТ по лабораторной работе № 5

дисциплина: Архитектура компьютера

Студент: Лисенков Е.Р.

Содержание

# 1 Цель работы

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

# 2 Задания

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 размеров 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 (учетве- 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 intпредназначена для вызова прерывания с указанным номером.

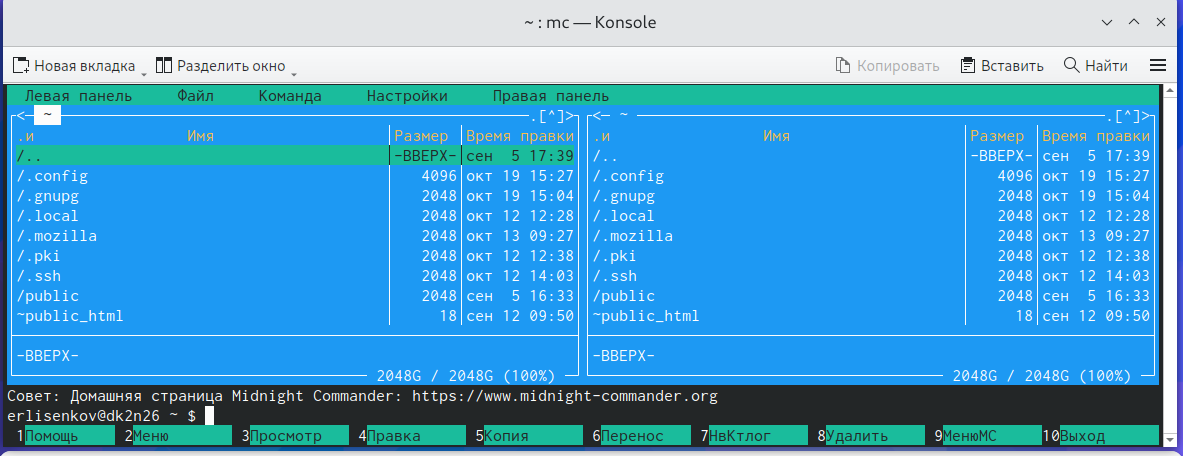
int n

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

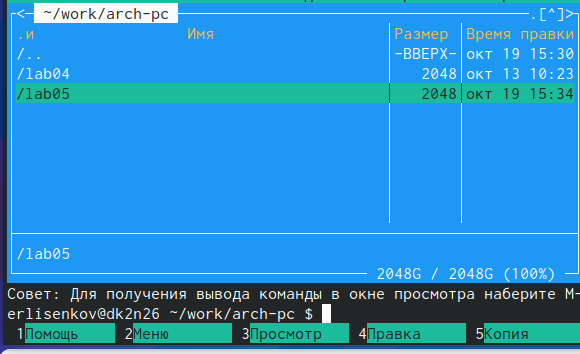
## 4.1 1 Основы работы с mc.

Открываю Midnight Commander с помощью команды mc (рис. ??).



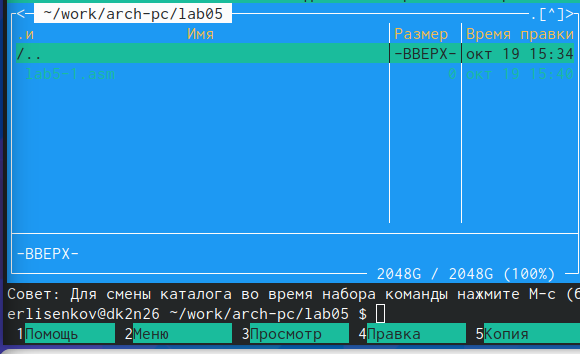
Открытый Midnight Commander

Перехожу в каталог ~/work/arch-pc (рис. ??).



Выполню переход

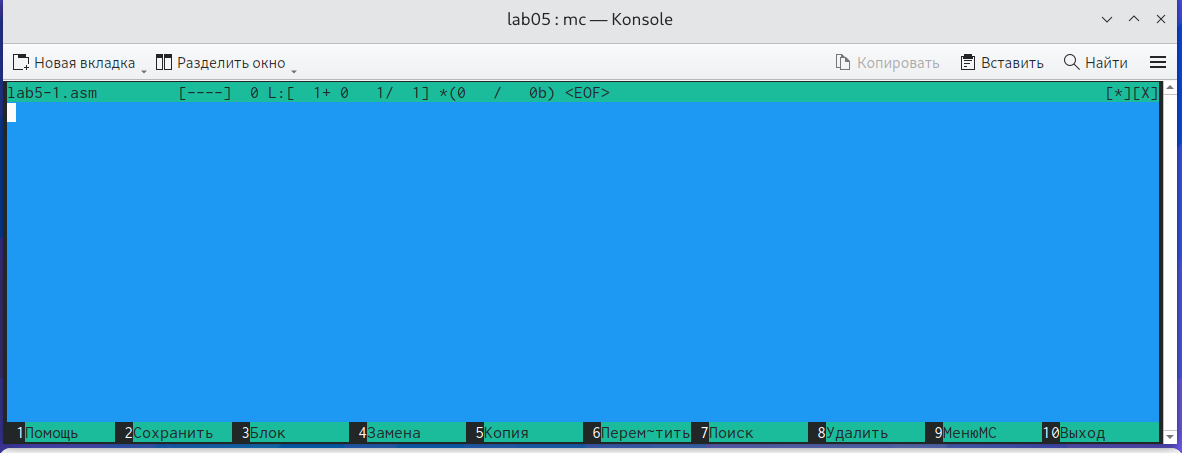
С помощью функциональной клавиши F7 создаю папку lab05, прописываю команду touch lab5-1.asm (рис. ??).



Создание каталога

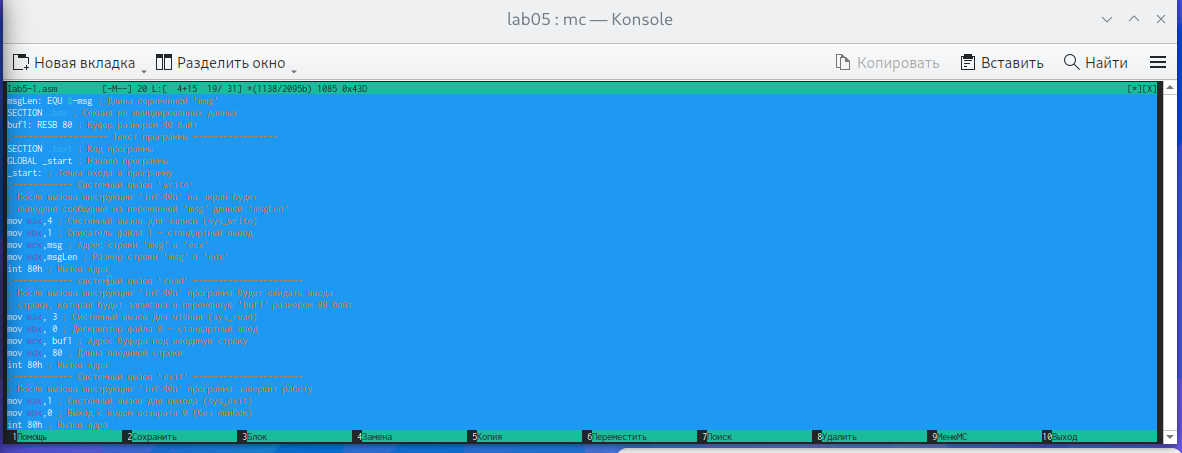
## 4.2 2 Структура программы на языке ассемблера NASM.

С помошью функциональной клавишы F4 открою файл для редактирования в редакторе nano (рис. ??).



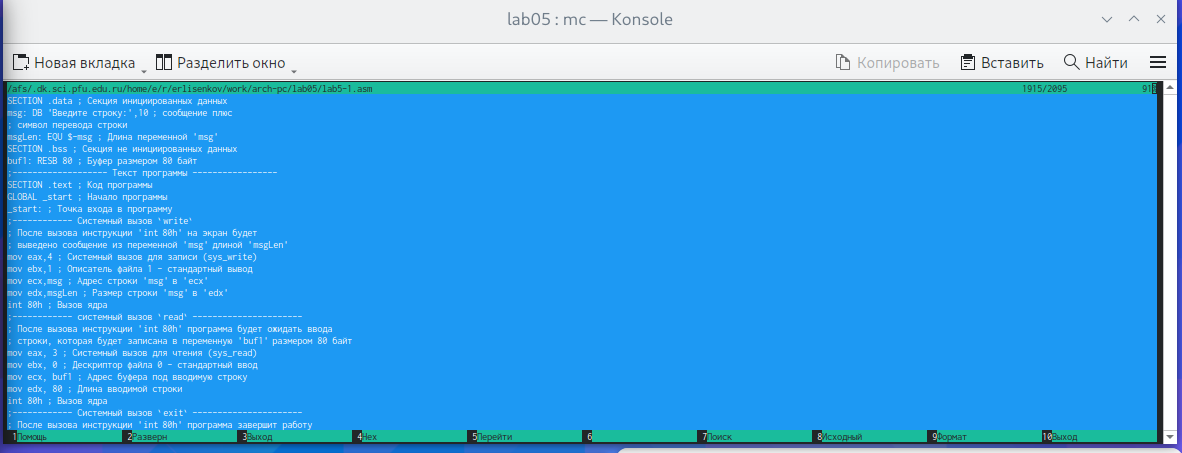
Открытие файла для редактирования

Введу в файл код программы для запроса строки у пользователя (рис. ??).



Редактирование файла

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



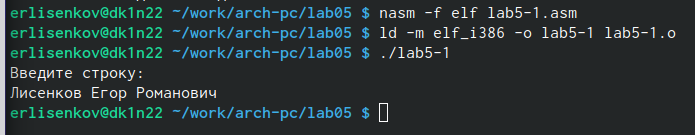
Просмотр текста программы

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 компановку файла).

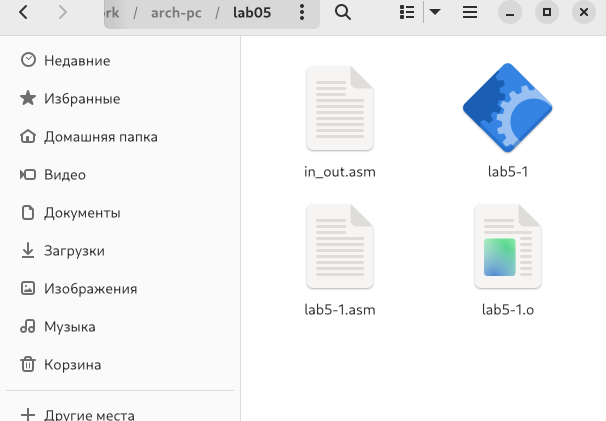
./lab5-1 (Запускаю программу этой командой) (рис. ??).



Трансляция и запуск

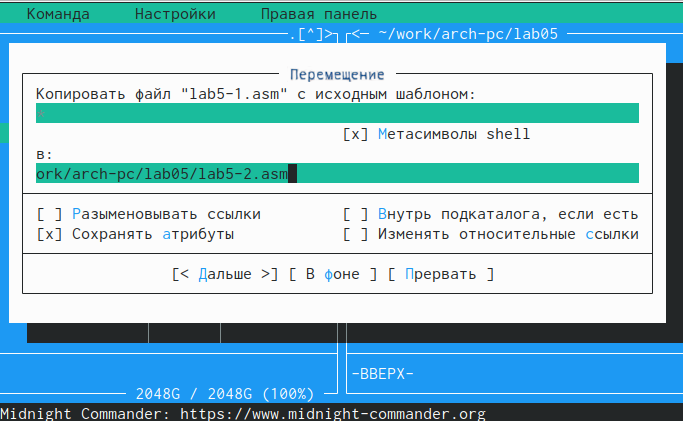
## 4.3 3.Подключение внешнего файла

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



Скачанный файл

Выполню копирование файла c помощью клавиши F5 in\_out.asm из каталога Загрузки в lab06 (рис. ??).



Копирование файла

Исправляю текст программы и включаю в работу подпрограммы из внешнего файла in\_out.asm. Для изменений используем редактор nano. 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

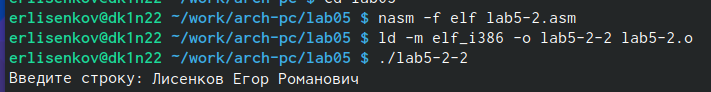
mov ecx, buf1 ; запись адреса переменной в EAX

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. ??).

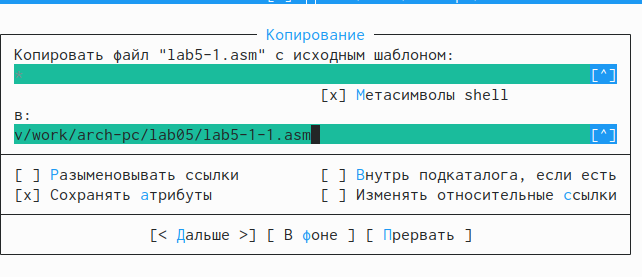


Компоновка и запуск программы

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

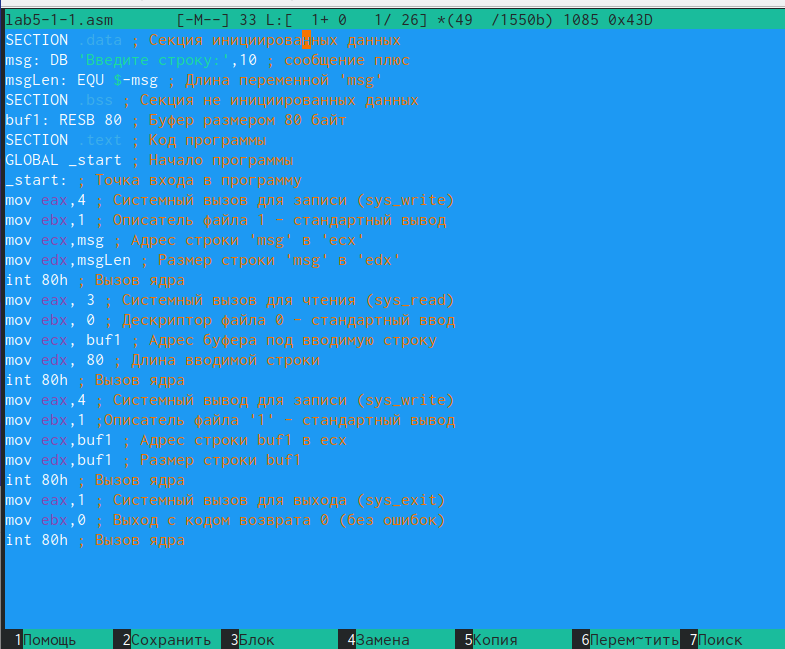
# 5 Выполнение заданий

№1 Создам копию файла lab5-1.asm (использую клавишу F5). (рис. ??).



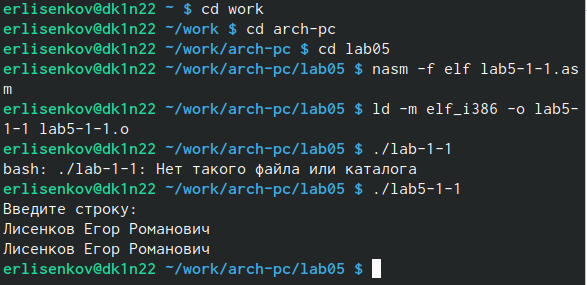
Создаю копию файла

Выполню изменения в коде (с помощью клавиши F4) (рис. ??).



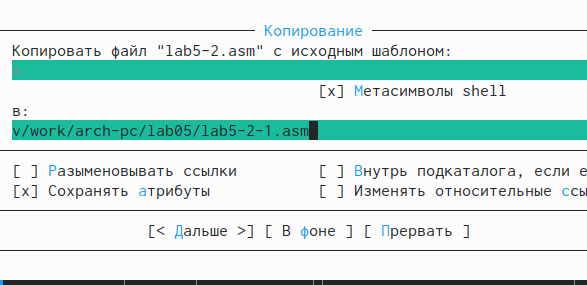
Изменения в коде

№2 Выполняем компоновку и запускаем программу.(рис. ??).



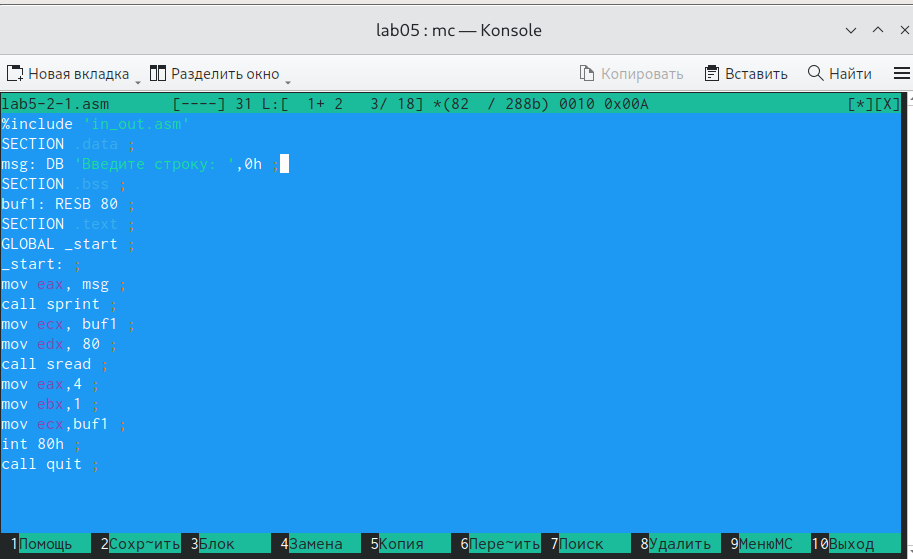
Компоновка и запуск

№3 Создам копию файла lab5-2.asm (использую клавишу F5). (рис. ??).



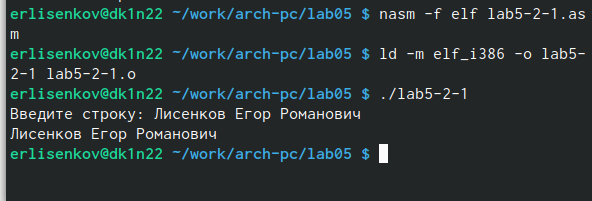
Создадим копию

Исправляю код программы и включаю в работу внешний файл in\_out.asm. (рис. ??).



Изменение кода

№4 Выполню компоновку и запускаю программу (рис. ??).



Компоновка и запуск программы

# 6 Выводы

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

# 7 Список литературы

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