Отчет по лабораторной работе №5

Дисциплина: архитектура компьютера

Осипов Никита Александрович

Содержание

[1 Цель работы 1](#_Toc185035863)

[2 Задание 1](#_Toc185035864)

[3 Теоретическое введение 1](#_Toc185035865)

[4 Выполнение лабораторной работы 2](#_Toc185035866)

[4.1 Основы работы c Midnight Commander 2](#_Toc185035867)

[4.2 Работа в NASM 5](#_Toc185035868)

[4.3 Подключение внешнего файла 6](#_Toc185035869)

[4.4 Задание для самостоятельной работы 9](#_Toc185035870)

[5 Выводы 12](#_Toc185035871)

[Список литературы 12](#_Toc185035872)

# 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 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 Основы работы c Midnight Commander

Введя соответствующ комманду в терминале (рис. 1), я открываю Midnight Commander (рис. 2).

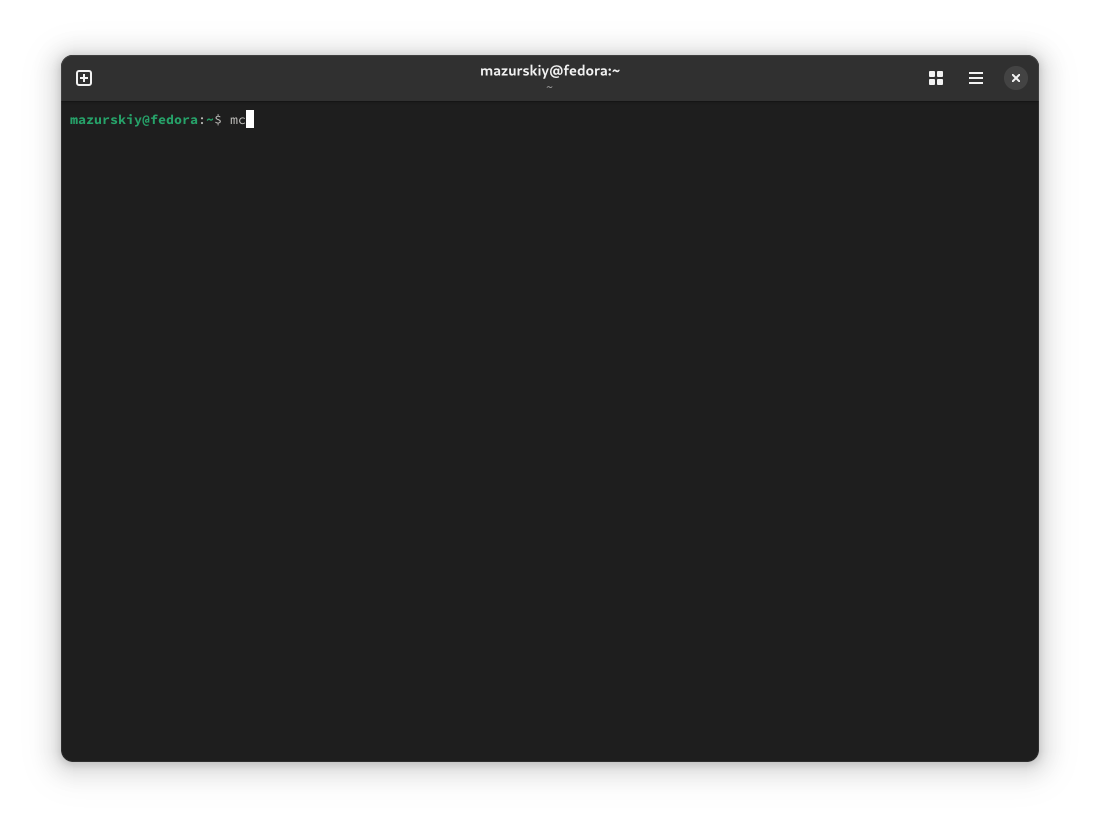


Рис. 1: Отркрытие Midnight Commander

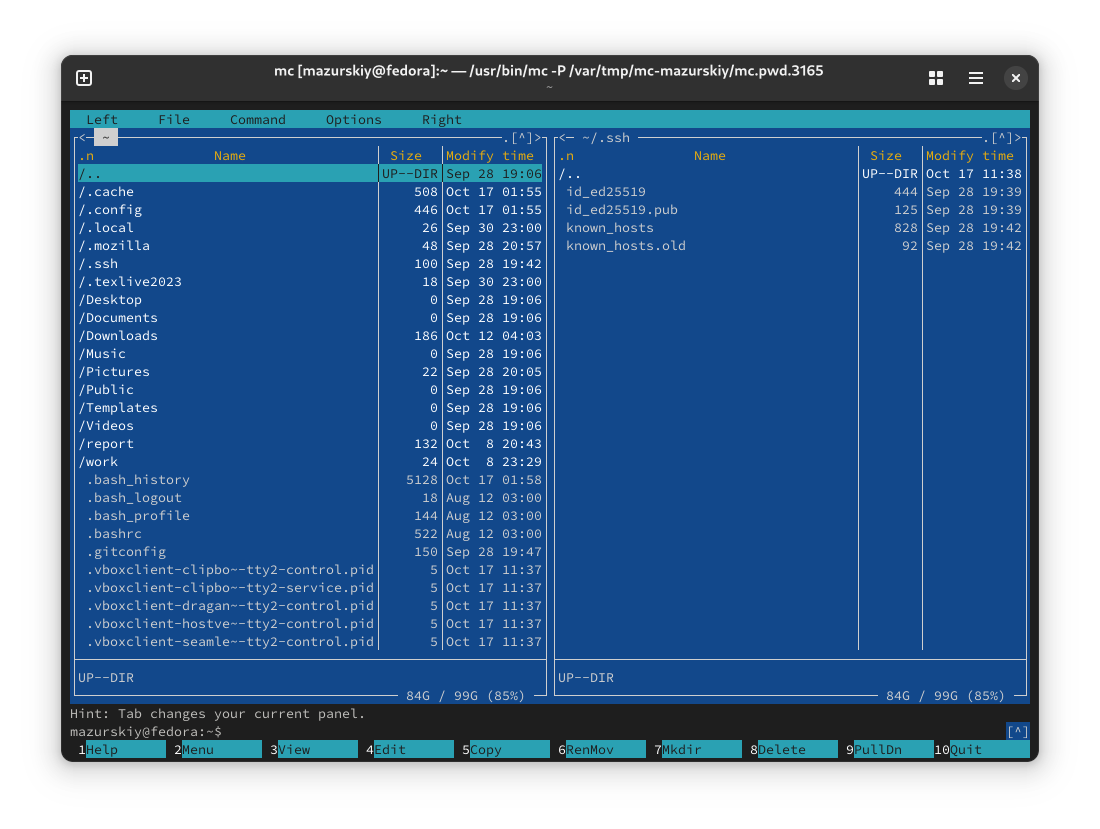


Рис. 2: Интерфейс Midnight Commander

Перехожу в созданный каталог в предыдущей лабораторной работе (рис. 3).

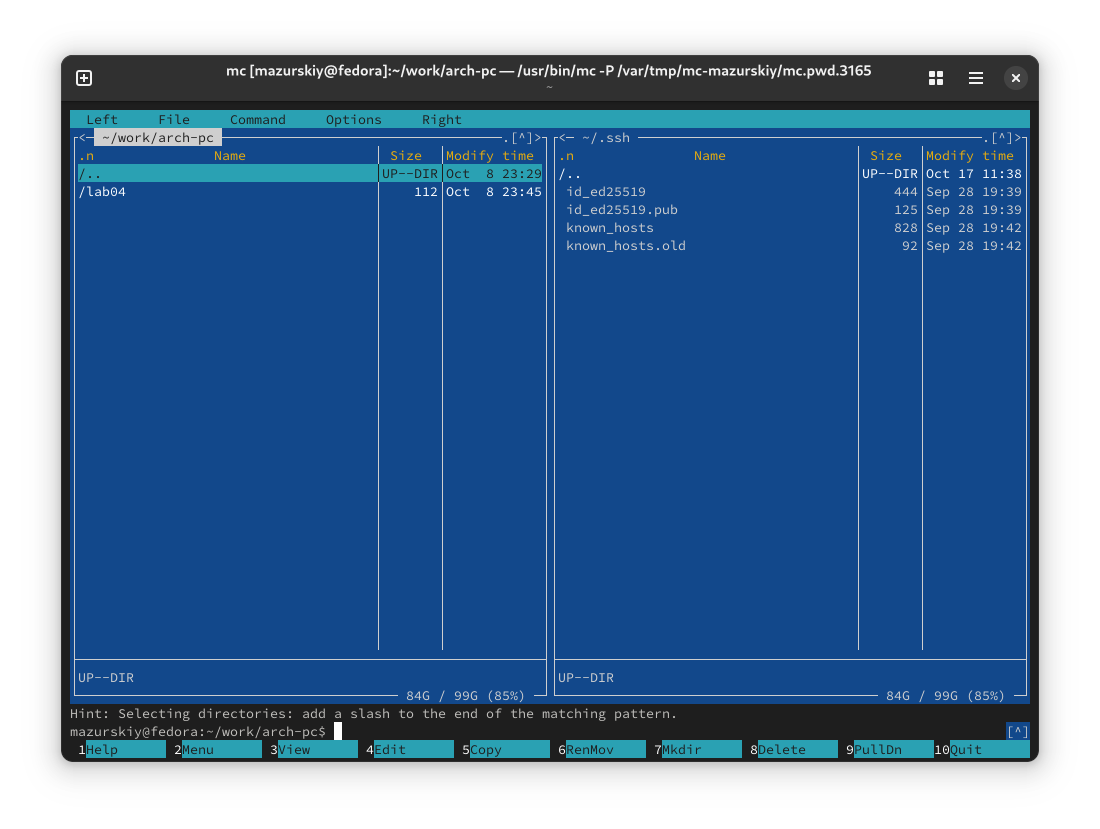


Рис. 3: Открытый каталог arch-pc

С помощью функциональной клавиши, я создаю подкаталог lab05, в котором буду работать (рис. 4).

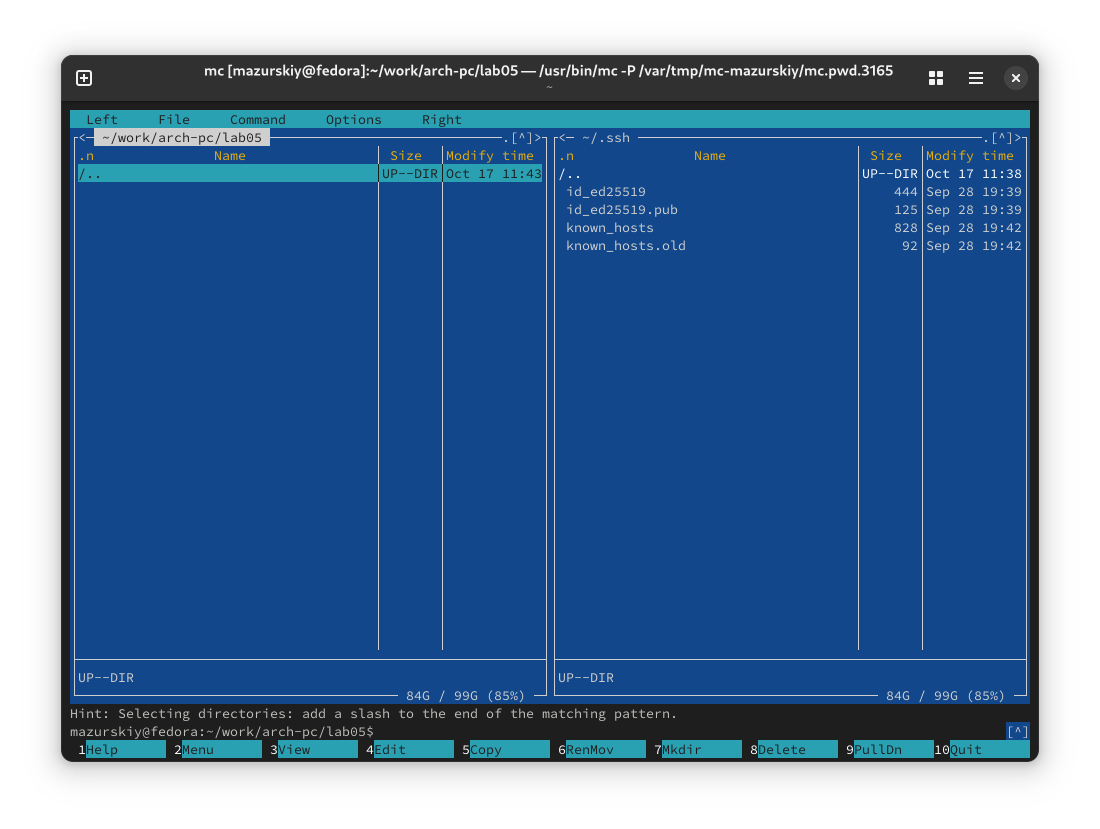


Рис. 4: Создание рабочего подкаталога

В строке ввода вводжу команду touch и создаю файл (рис. 5).

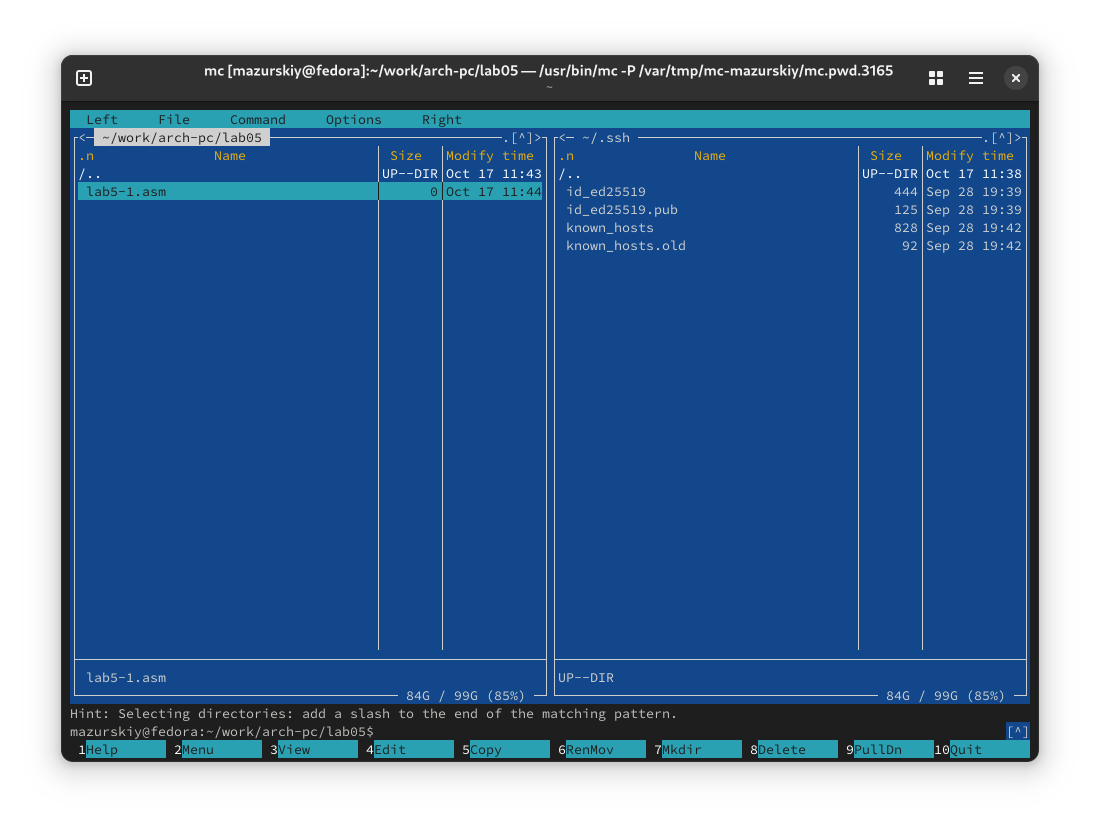


Рис. 5: Создание файла в Midnight Commander

## 4.2 Работа в NASM

С помощью F4 открываю только что созданный файл и вношу код с листинга (рис. 6).

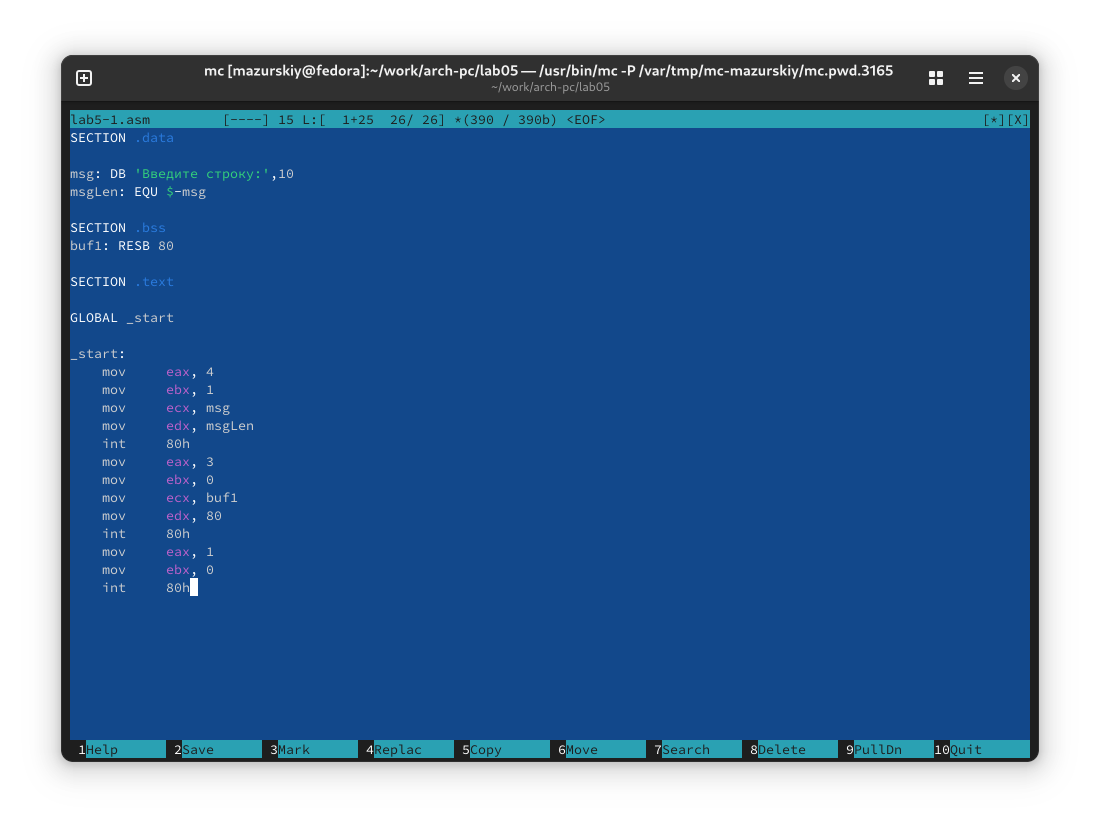


Рис. 6: Редактирование файла в Midnight Commander

Проверяю сохраненные изменения с помощью клавиши F3 (рис. 7).

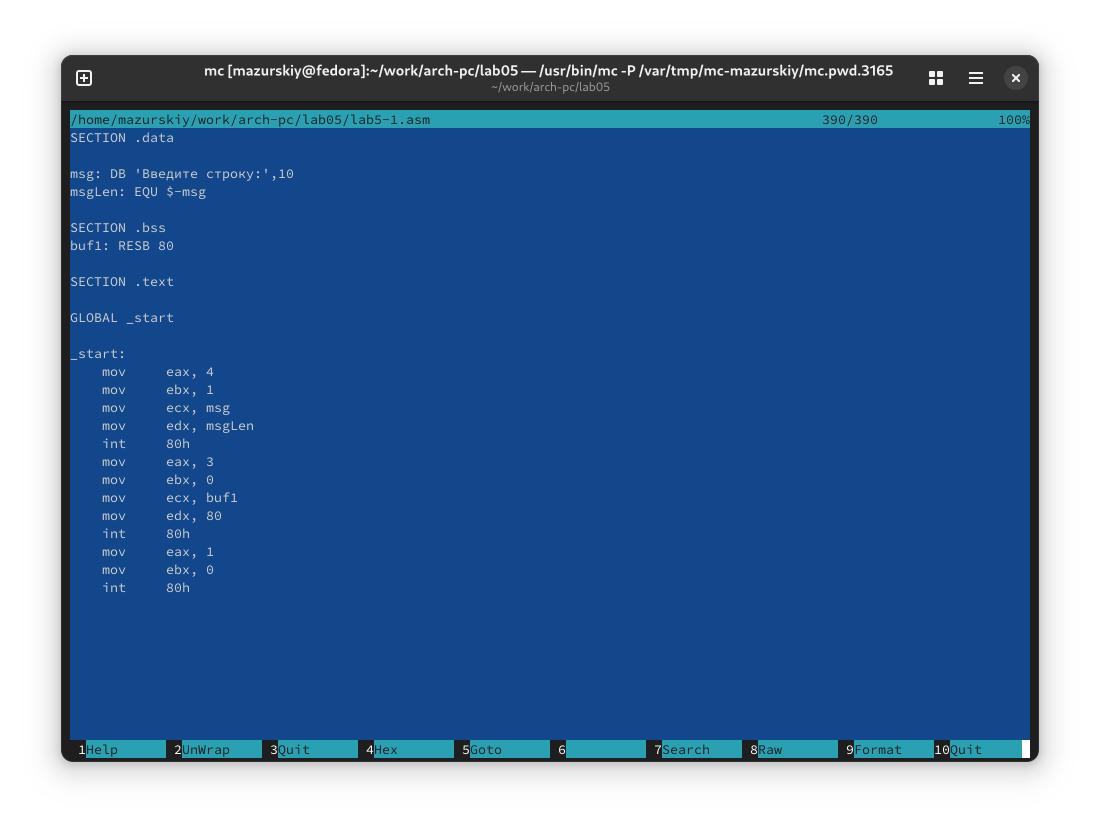


Рис. 7: Проверка сохранения сделанных изменений

Транслирую и компоную измененный файл, запускаю (рис. 8).

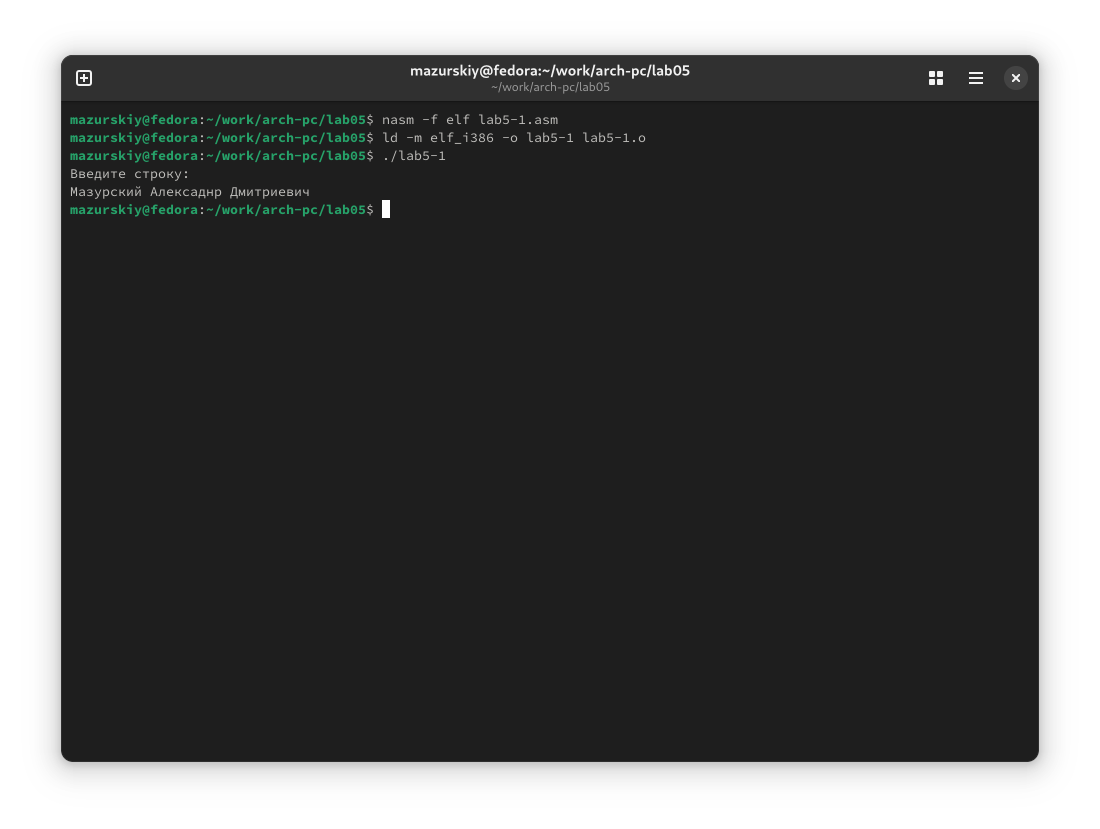


Рис. 8: Трансляция, компоновка и последующий запуск программы

## 4.3 Подключение внешнего файла

Скачанный с ТУИС файл сохраняю в общую папку на своем компьютере, на виртуальной машине в интерфейсе Midnight Commander перехожу в директорию общей папки, копирую файл в рабочий подкаталог. (рис. 9).

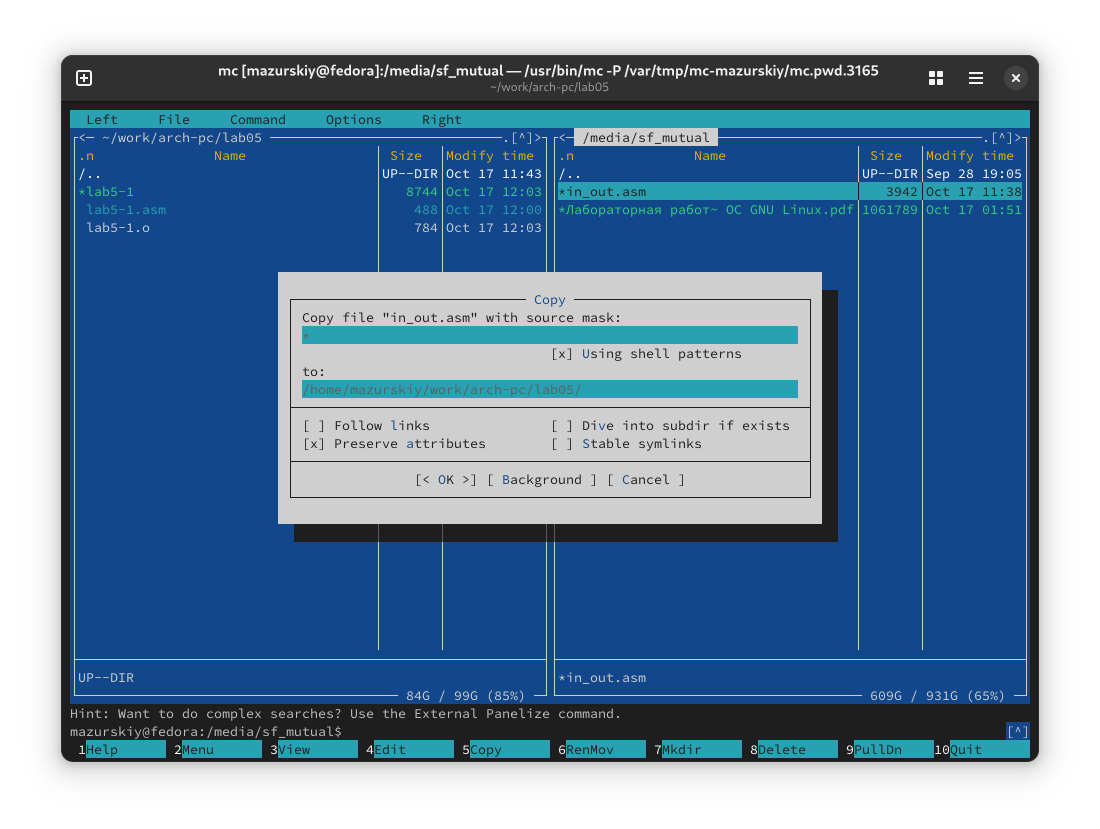


Рис. 9: Копирование файла в рабочий каталог

Создаю копию файла для последующей работы с ним (рис. 10).

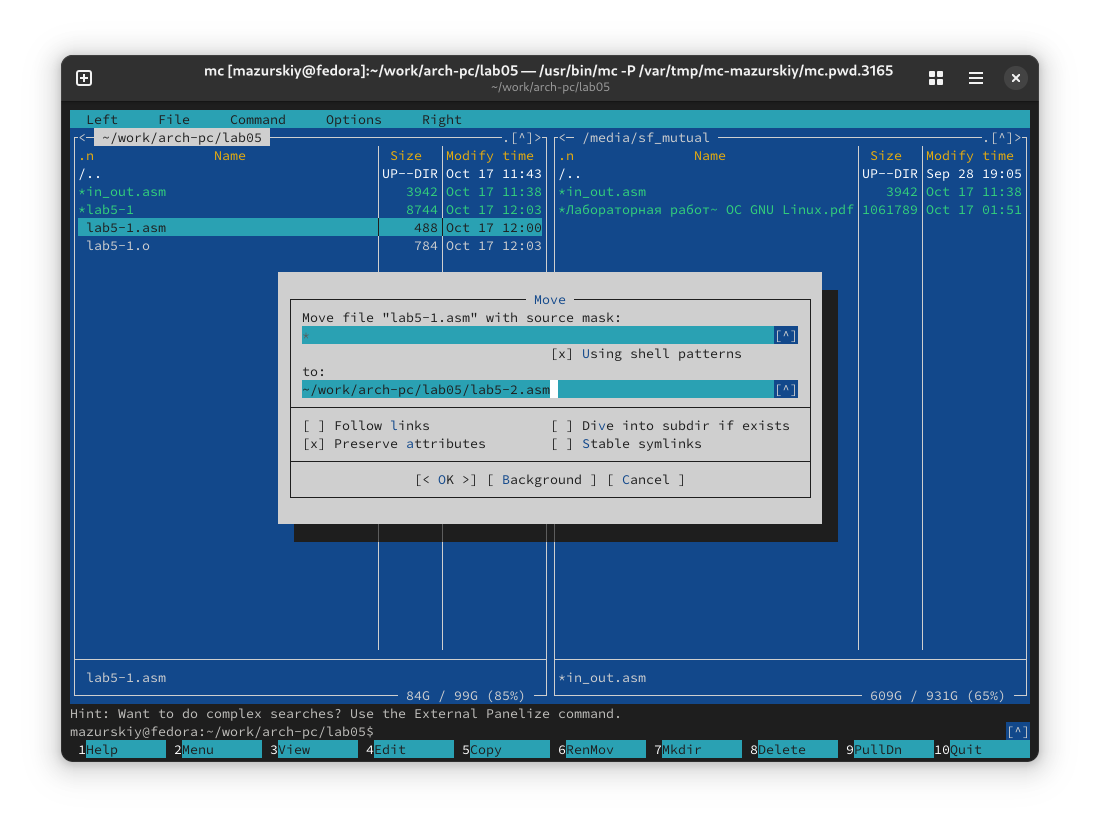


Рис. 10: Создание копии файла в Midnight Commander

В копии файла подключаю подпограмм из подключенного файла (рис. 11).



Рис. 11: Изменение программы

Транслирую, компоную и запускаю программу с подключенным файлом (рис. 12).

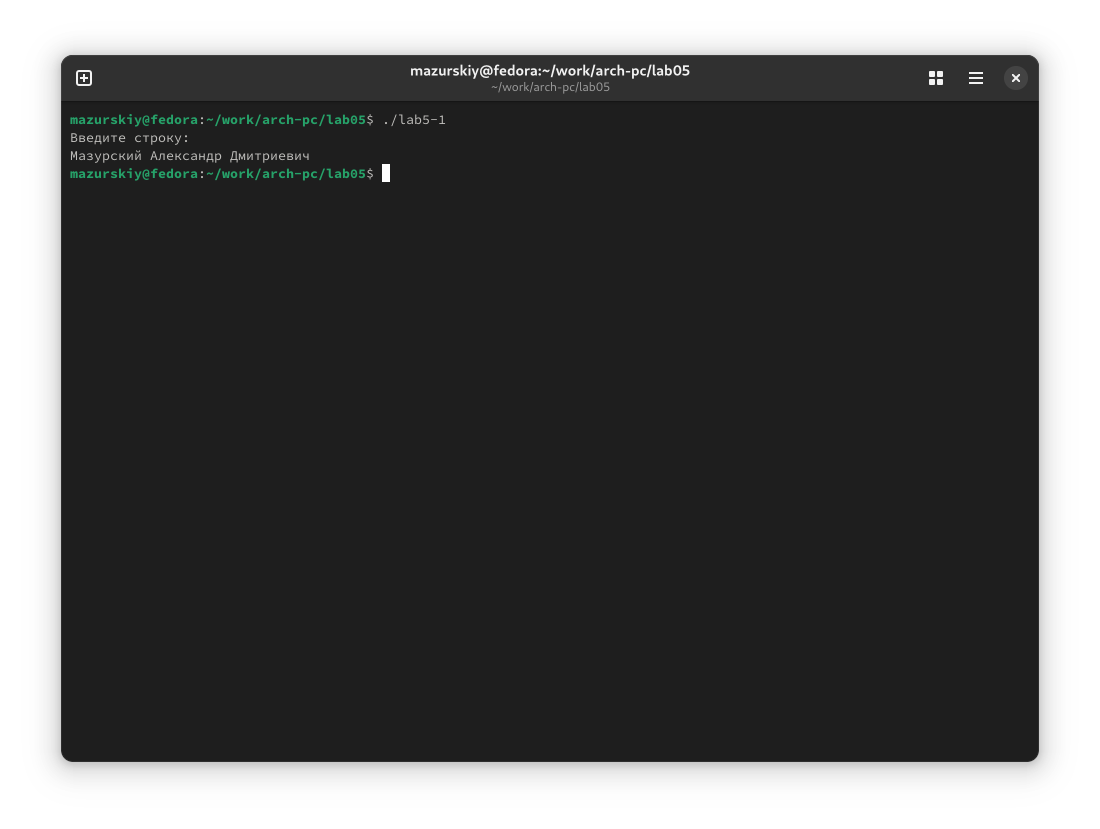


Рис. 12: Запуск измененной программы

Редактирую файл и заменяю в нем подпрограмму sprintLF на sprint. Разница подпрограмм в том, что вторая вызывает ввод на той же строке (рис. 13).

## 4.4 Задание для самостоятельной работы

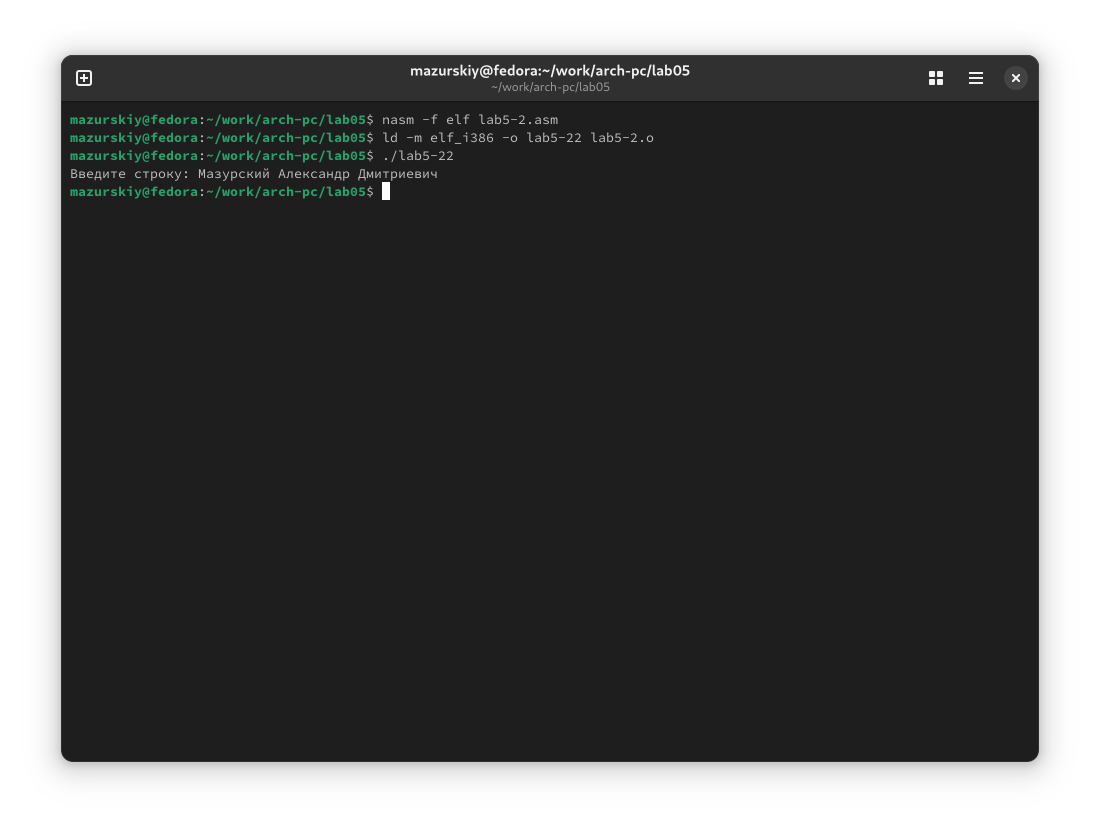


Рис. 13: Запуск изменной программы с другой подпрограммой

Создаю копию lab5-1.asm, редактирую так, чтобы в конце выводилась введеная мною строка с клавиатуры (рис. 14).

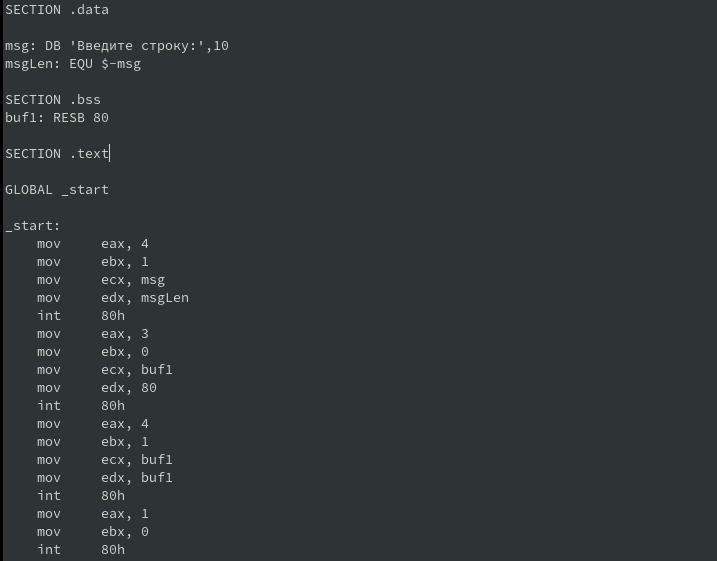


Рис. 14: Редактирование копии

Транслирую, компоную и запускаю свою программу (рис. 15).

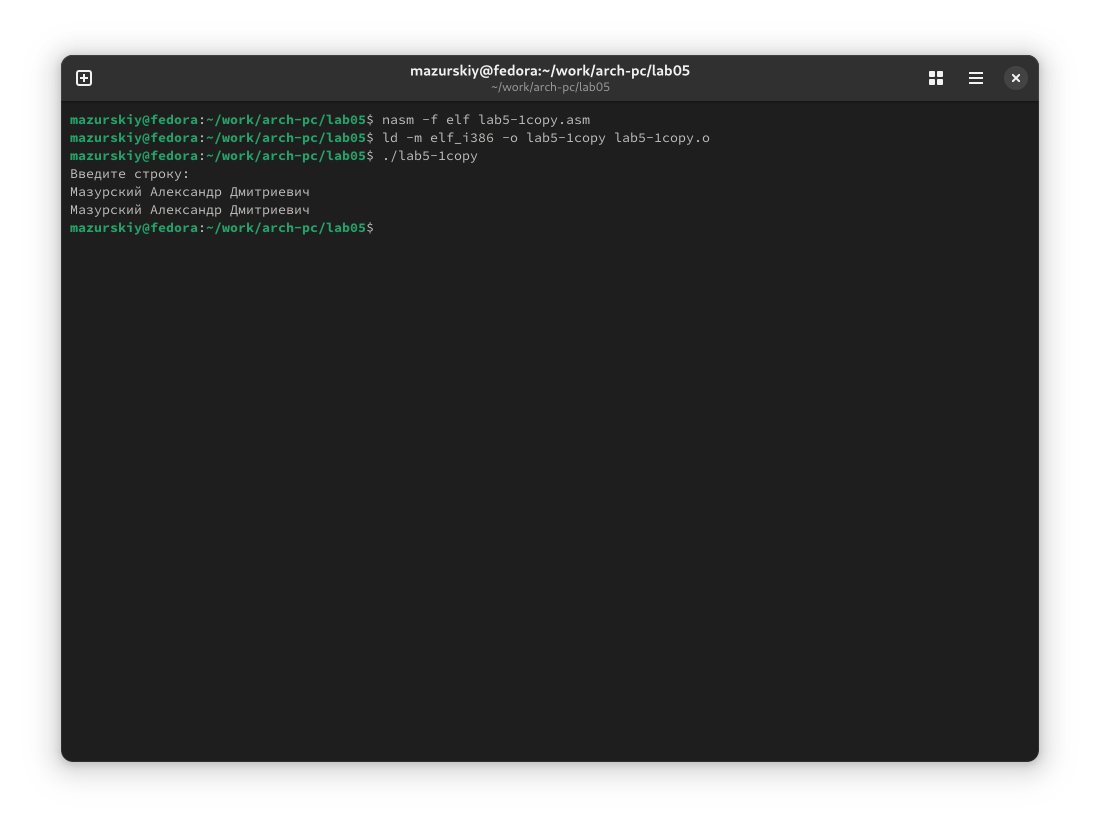


Рис. 15: Запуск своей программы

Код прикладываю

ECTION .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

Создаю копию lab5-2.asm, редактирую так, чтобы в конце выводилась введеная мною строка с клавиатуры (рис. 16).



Рис. 16: Редактирование копии

Транслирую, компоную и запускаю свою программу (рис. 17).

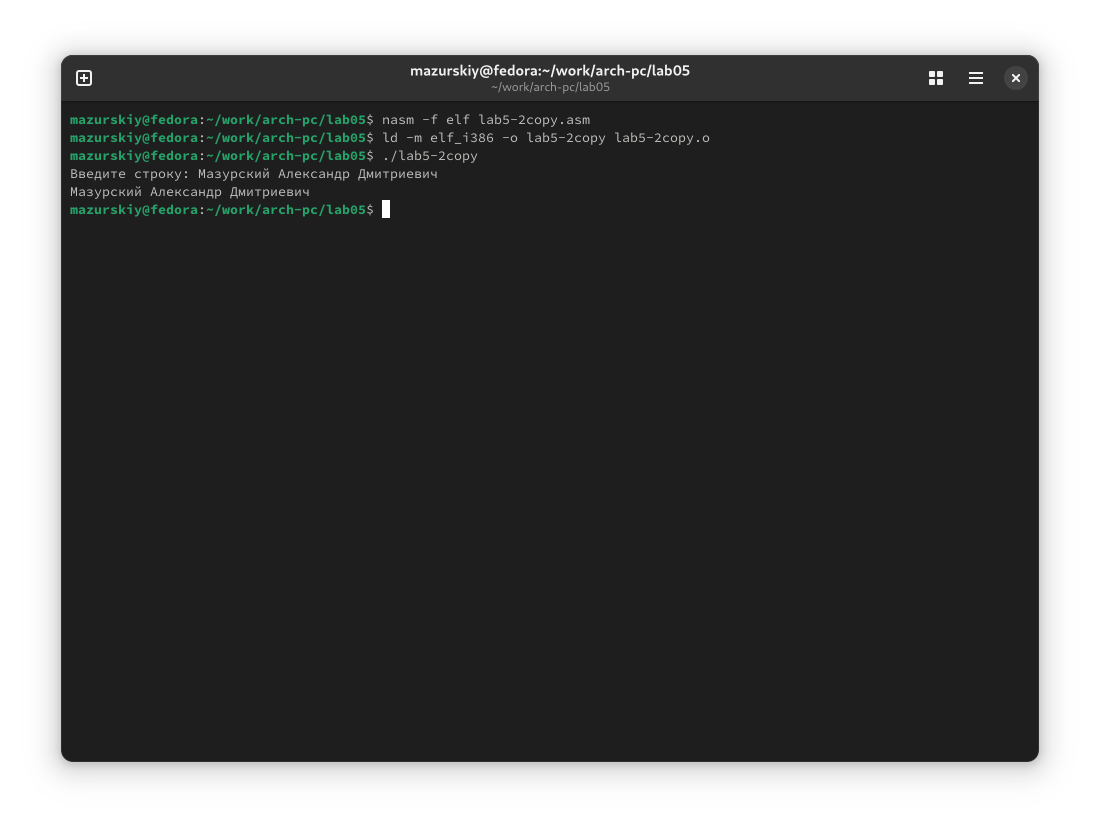


Рис. 17: Запуск своей программы

Код прикладываю:

%include 'in\_out.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 sprint  
  
 mov ecx, buf1  
 mov edx, 80  
  
 call sread  
  
 mov eax, 4  
 mov ebx, 1  
 mov ecx, buf1  
 int 80h  
  
 call quit

# 5 Выводы

При выполнении данной лабораторной работы я приобрёл практические навыки работы в Midnight Commander, а также освоил инструкции языка ассемблера mov и int.

# Список литературы

1. [Пример выполнения лабораторной работы](https://github.com/evdvorkina/study_2022-2023_arh-pc/blob/master/labs/lab06/report/%D0%9B06_%D0%94%D0%B2%D0%BE%D1%80%D0%BA%D0%B8%D0%BD%D0%B0_%D0%BE%D1%82%D1%87%D0%B5%D1%82.md?plain=1)
2. [Курс на ТУИС](https://esystem.rudn.ru/course/view.php?id=112)
3. [Лабораторная работа №5](https://esystem.rudn.ru/pluginfile.php/2089085/mod_resource/content/0/%D0%9B%D0%B0%D0%B1%D0%BE%D1%80%D0%B0%D1%82%D0%BE%D1%80%D0%BD%D0%B0%D1%8F%20%D1%80%D0%B0%D0%B1%D0%BE%D1%82%D0%B0%20%E2%84%965.%20%D0%9E%D1%81%D0%BD%D0%BE%D0%B2%D1%8B%20%D1%80%D0%B0%D0%B1%D0%BE%D1%82%D1%8B%20%D1%81%20Midnight%20Commander%20%28%29.%20%D0%A1%D1%82%D1%80%D1%83%D0%BA%D1%82%D1%83%D1%80%D0%B0%20%D0%BF%D1%80%D0%BE%D0%B3%D1%80%D0%B0%D0%BC%D0%BC%D1%8B%20%D0%BD%D0%B0%20%D1%8F%D0%B7%D1%8B%D0%BA%D0%B5%20%D0%B0%D1%81%D1%81%D0%B5%D0%BC%D0%B1%D0%BB%D0%B5%D1%80%D0%B0%20NASM.%20%D0%A1%D0%B8%D1%81%D1%82%D0%B5%D0%BC%D0%BD%D1%8B%D0%B5%20%D0%B2%D1%8B%D0%B7%D0%BE%D0%B2%D1%8B%20%D0%B2%20%D0%9E%D0%A1%20GNU%20Linux.pdf)
4. [Программирование на языке ассемблера NASM Столяров А. В.](https://esystem.rudn.ru/pluginfile.php/2088953/mod_resource/content/2/%D0%A1%D1%82%D0%BE%D0%BB%D1%8F%D1%80%D0%BE%D0%B2%20%D0%90.%20%D0%92.%20-%20%D0%9F%D1%80%D0%BE%D0%B3%D1%80%D0%B0%D0%BC%D0%BC%D0%B8%D1%80%D0%BE%D0%B2%D0%B0%D0%BD%D0%B8%D0%B5%20%D0%BD%D0%B0%20%D1%8F%D0%B7%D1%8B%D0%BA%D0%B5%20%D0%B0%D1%81%D1%81%D0%B5%D0%BC%D0%B1%D0%BB%D0%B5%D1%80%D0%B0%20NASM%20%D0%B4%D0%BB%D1%8F%20%D0%9E%D0%A1%20Unix.pdf)