РУДН. Архитектура компьютеров

Отчёт по лабораторной работе №5

Косинов Никита Андреевич, НПМбв-02-20

Содержание

# 1 Цель работы

Цель данной работы заключается в приобретении навыков работы с фаловым менеджером *Midnight Commander*: навигация, копирование, перемещение и редактирование объектов. Также, мы продолжаем изучать команды языка программирования Ассемблер такие, как **mov** и **int**.

# 2 Ход работы

Лабораторная работа выполнена с использованием консоли **OC Linux**, программы **Midnight Commander** и языка программирования ассемблера **NASM**.

1. Навигация в *Midnight Commander*;
2. Редактирование файла в *mcedit*;
3. Подключение “библиотеки”.

В конце выполнена самостоятельная работа.

# 3 Навигация в *Midnight Commander*

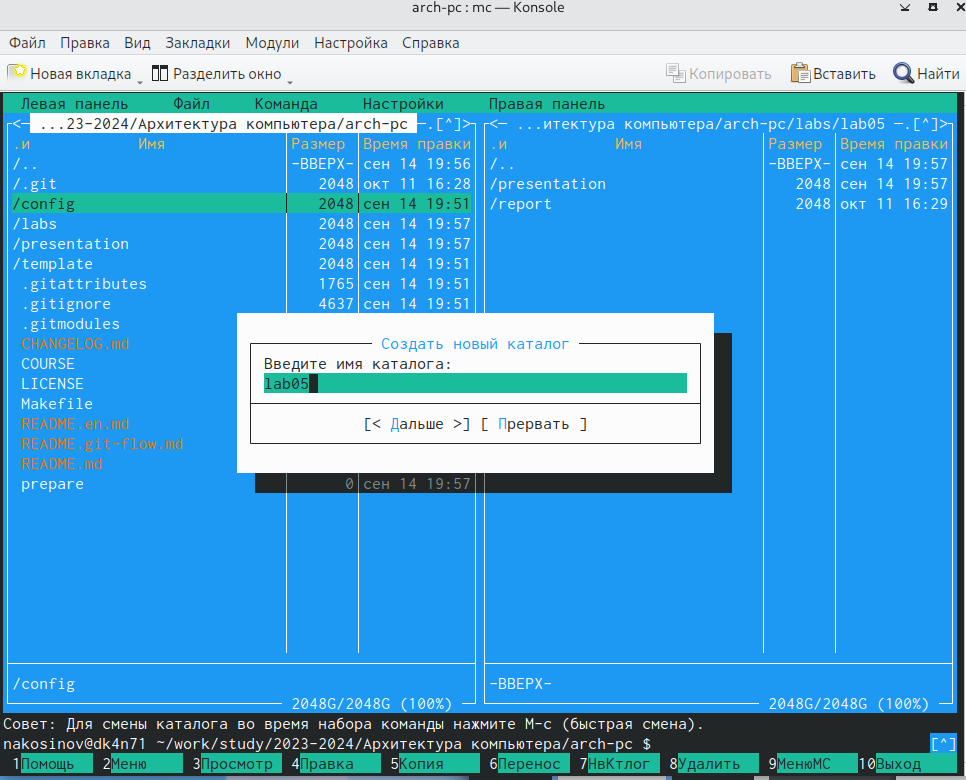
С помощью файлового навигатора *Midnight Commander* научимся переходить между каталогами, создавать новые и изменять файлы с помощью встроенного редактора

1. Командой **mc** открываем *Midnight Commander*.

Запуск менеджера

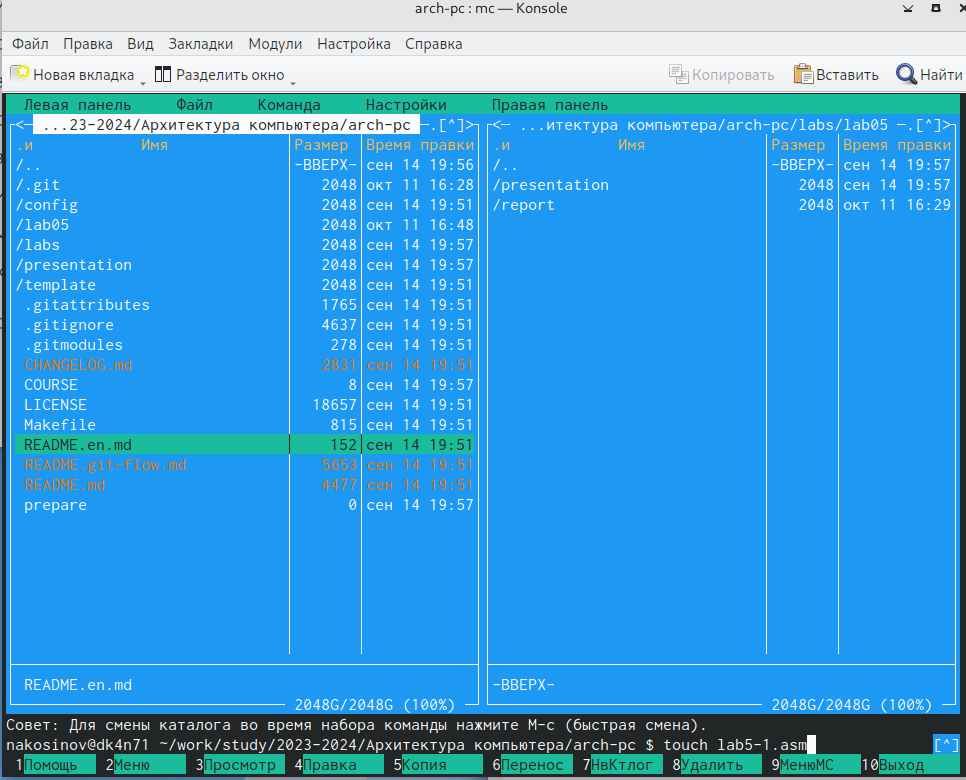
Запуск менеджера

1. Переходим выше по файловой системе в папку *~arch-pc* и создаём в ней каталог *lab05* с помощью клавиши **F7**.



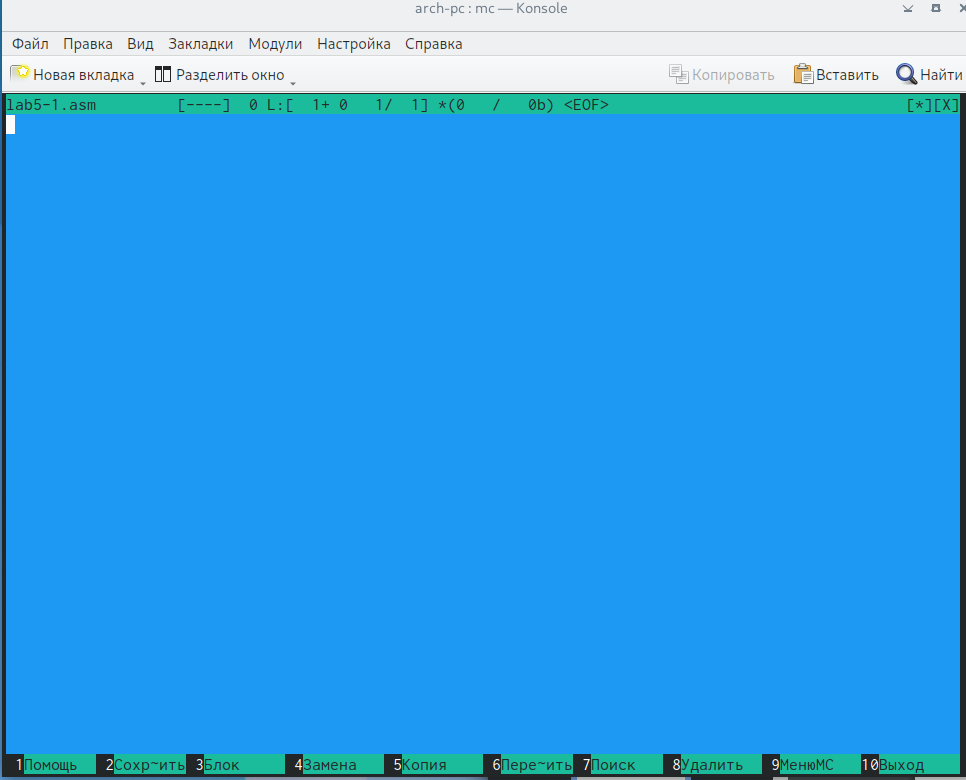
Создание каталога в МС

1. С помощью строки ввода, оставшейся в нижней части консоли, командой **touch** создаём файл программы *lab5-1.asm*.



Создание файла в MC

1. Выделив стрелками созданный файл и нажав клавишу **F4**, открываем файл во встроенном текстовом редакторе *mcedit*.



Изменение файла в MC

# 4 Редактирование файла в *mcedit*

Открыв файл программы, напишем сам набор команд.

1. В блоке *.data* вводим переменные **msg** и **msgLen**, т.к. их значение нам известно.
2. В блоке *.bss* вводим переменную **buf1**, т.к. в ней будет храниться значение, заданное пользователем.
3. В блоке *.text* записываем последовательность групп команд. Каждая группа заканчивается вызовом ядра: **int 80h**.
   1. Первая группа отвечает за вывод на экран значения переменной **msg**.
   2. Вторая группа отвечает за сохранение значения, введённого пользователем, в переменную **buf1**.
   3. Третья группа отвечает за выход из программы без ошибок.

’’’ SECTION .data

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

SECTION .bss

buf1: RESB 80

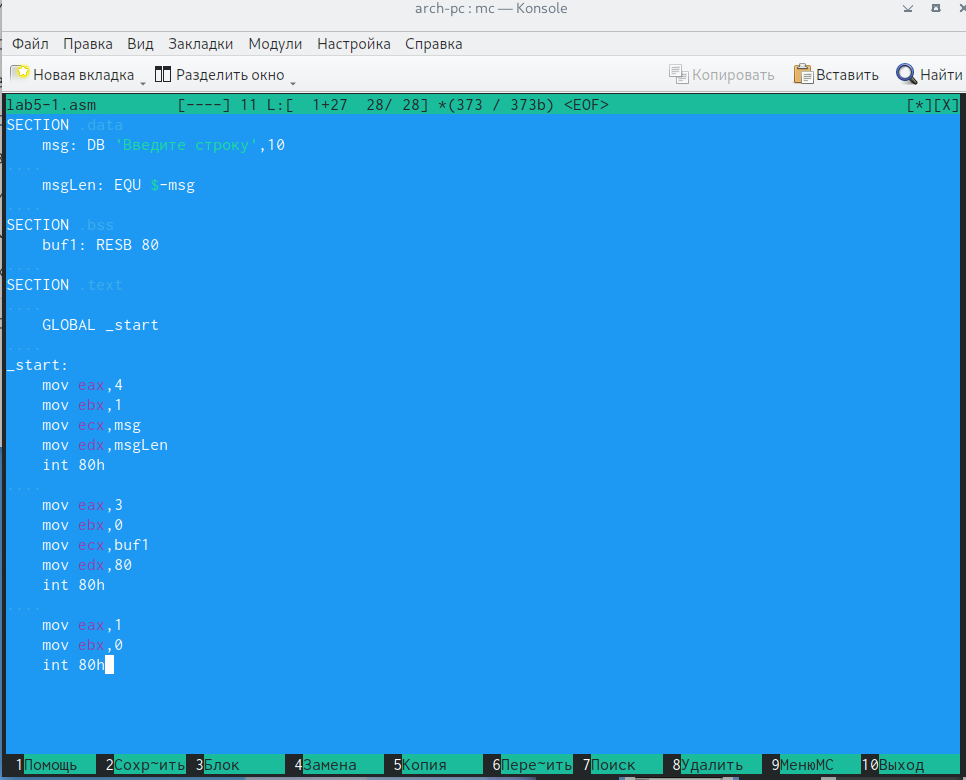
SECTION .text

GLOBAL \_start

\_start:

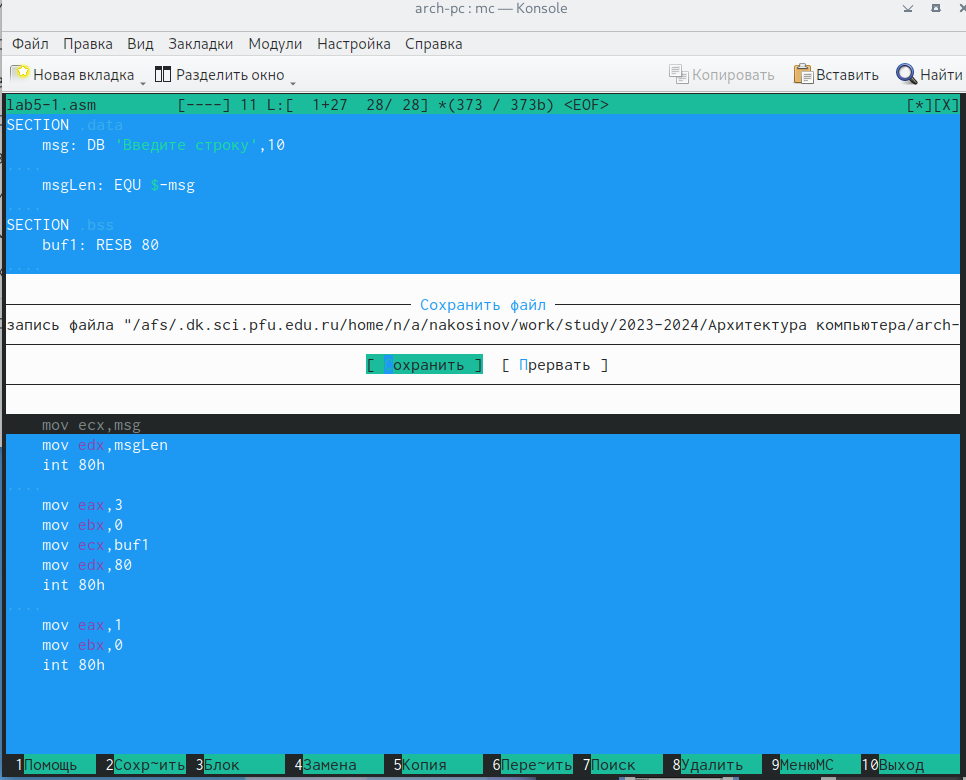
mov eax,4  
mov ebx,1  
mov ecx,msg  
mov edx,msgLen  
int 80h  
  
mov eax,3  
mov ebx,0  
mov ecx,buf1  
mov edx,80  
int 80h  
  
mov eax,1  
mov ebx,0  
int 80h

’’’



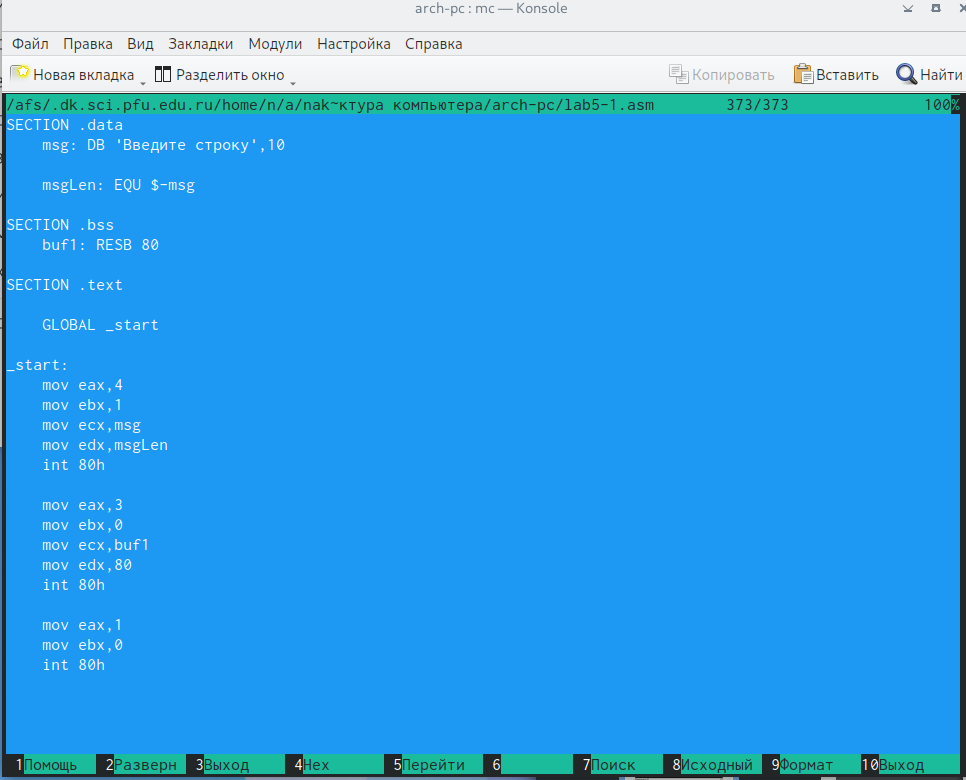
Код lab5-1

1. Клавишей **F2** сохраняем данные изменения.



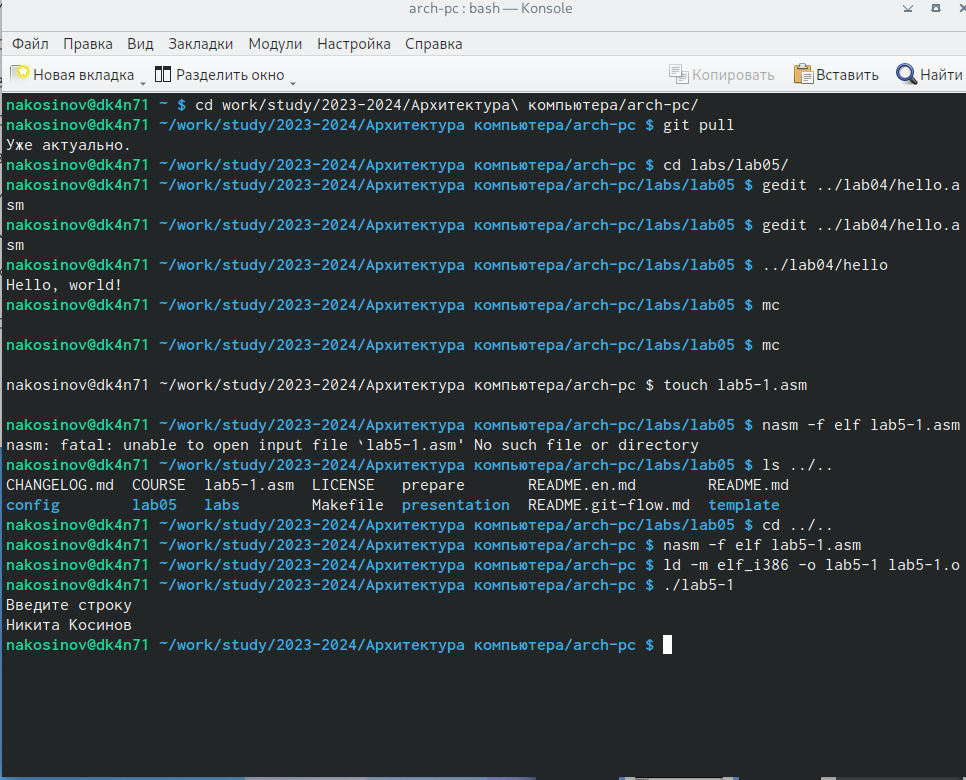
Сохранение файла

1. Клавишей **F3** откроем написанный код только для простмотра.



Просмотр файла

1. Транслируем написанный код в объектный файл и компонуем последний. Запустив исполняемый файл, убеждаемся, что он работает исправно!

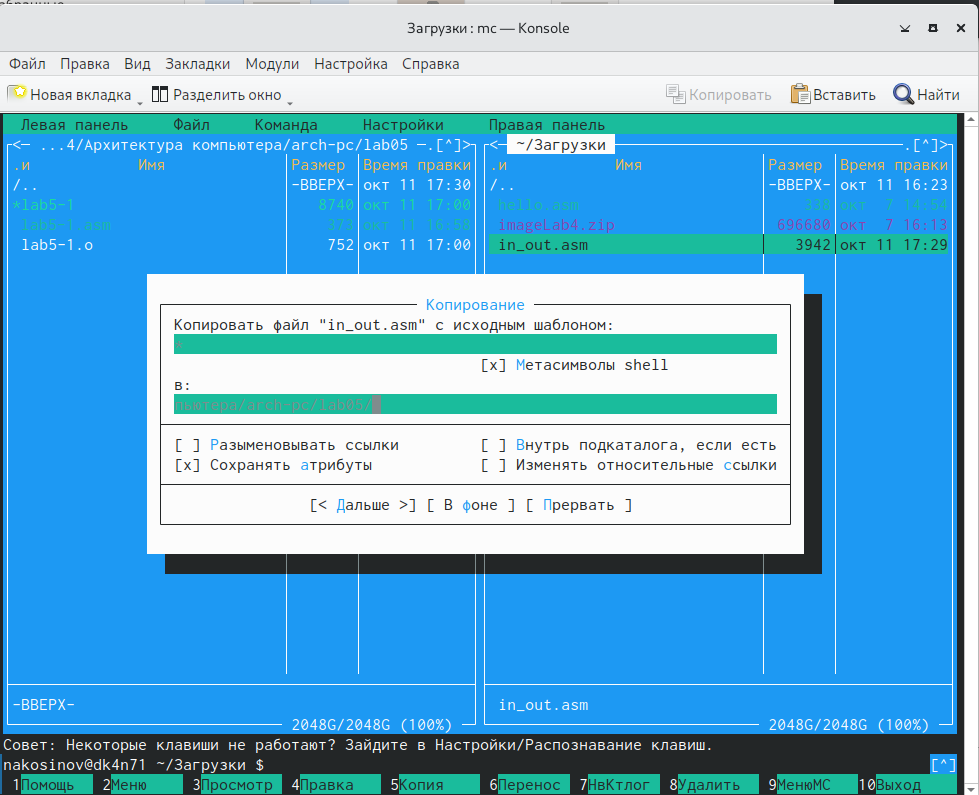


Запуск программы

# 5 Подключение “библиотеки”

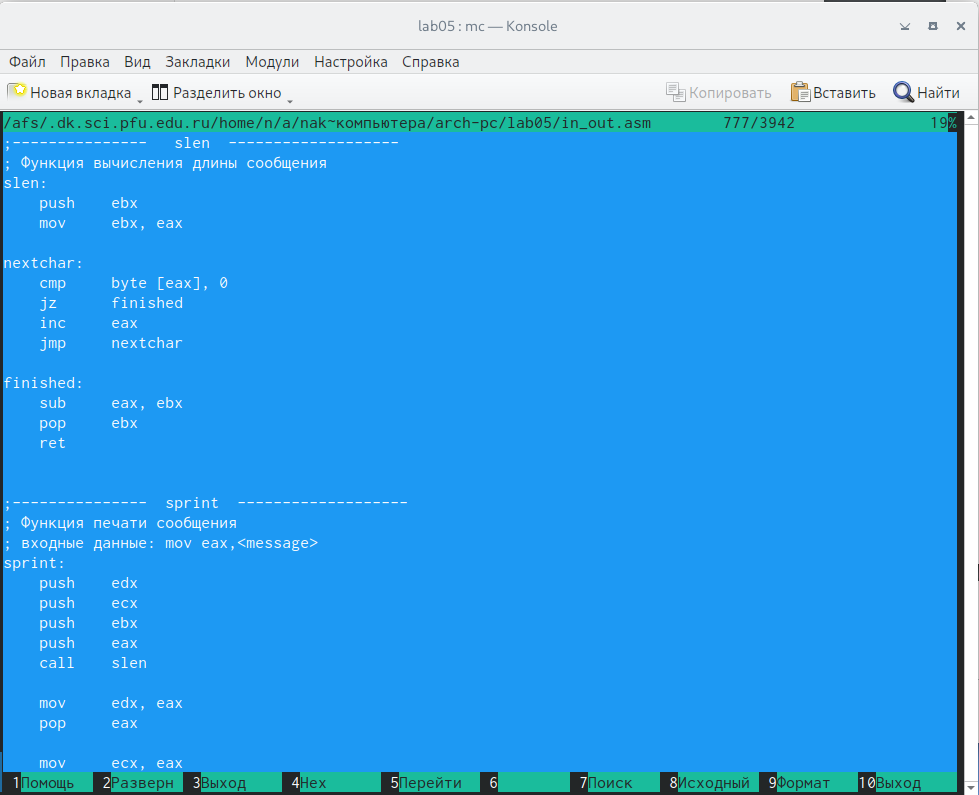
Для удобства работы и ускорения процесса, для возможности командной работы в ассемблере предусмотрена работа с подключаемыми файлами, в которых находятся прописанные заранее команды. При подключении такого файла ассемблер, видя незнакомую ему команду, ищет её в подключенном внешнем файле и подставляет написанный там набор команд.

1. Скачиваем файл *in\_out.asm*, открываем папку *Загрузки* во второй вкладке *МС* и копируем файл в наш рабочий каталог.



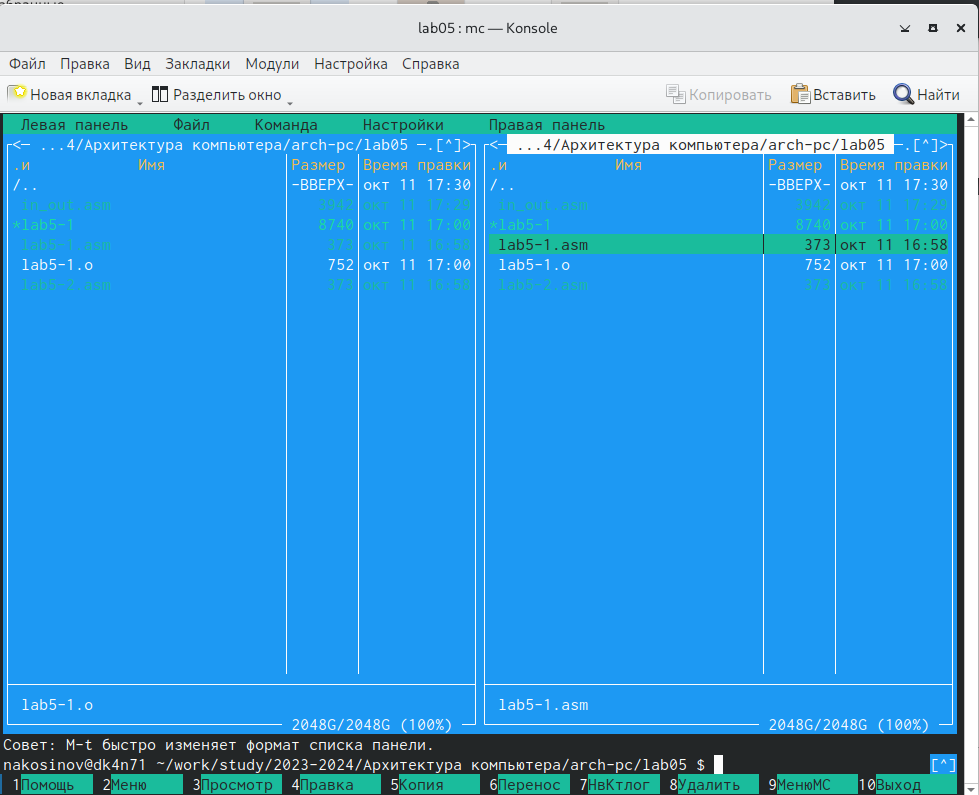
Копирование файла

1. Клавишей **F3** откроем файл *in\_out.asm*, чтобы ознакомиться с его структурой.



Код подключаемого файла

1. Также скопируем написанную нами ранее программу, дав ей новое имя и оставив в той же папке.



Новая структура рабочего каталога

1. Изменяем код, подключая внешний файл и меняя стандартные команды на команды из “библиотеки”.

’’’ %include ‘in\_out.asm’

SECTION .data

msg: DB 'Введите строку',0h

SECTION .bss

buf1: RESB 80

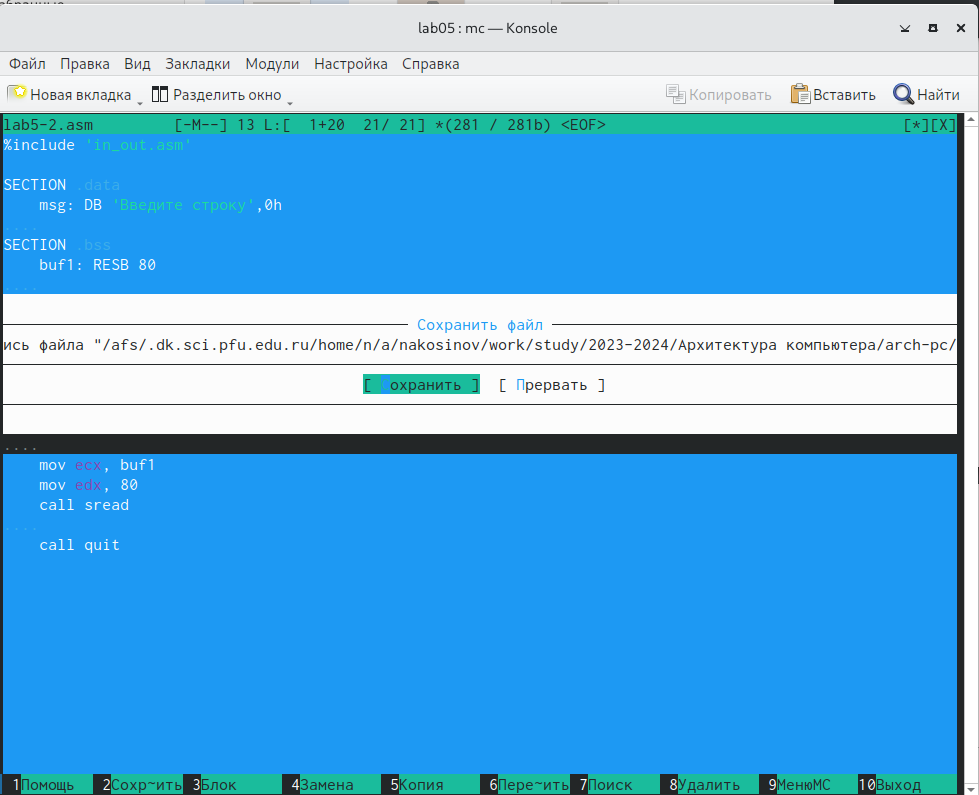
SECTION .text

GLOBAL \_start

\_start:

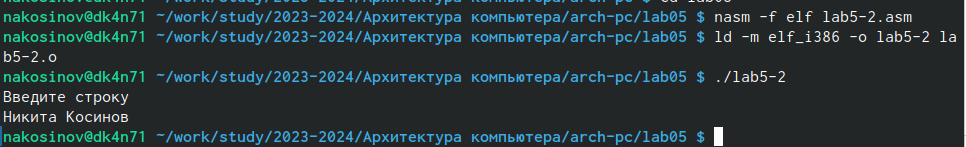
mov eax, msg   
call sprintLF  
  
mov ecx, buf1  
mov edx, 80  
call sread  
  
call quit

’’’



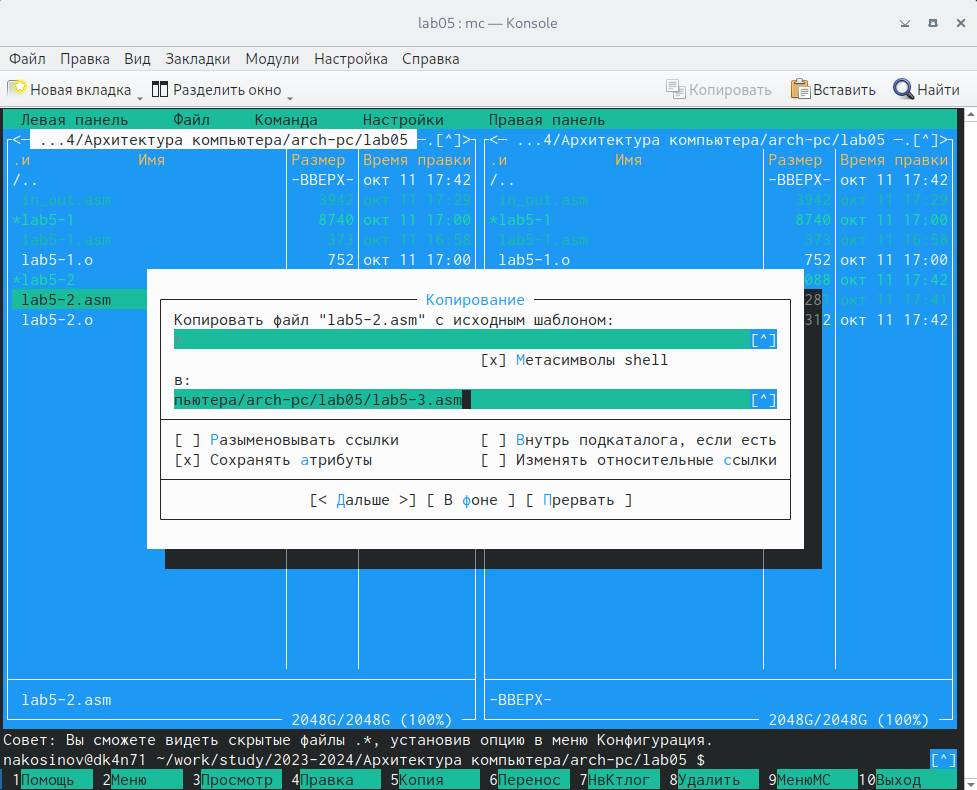
Код lab5-2

1. Компилируем и запускаем исполняемый файл. Проверяем его работу.



Результат исполнения lab5-2

1. Дублируем *lab5-2.asm*, меняя команду **sprintLF** на **sprint**.



Копирование файла

’’’ %include ‘in\_out.asm’

SECTION .data

msg: DB 'Введите строку',0h

SECTION .bss

buf1: RESB 80

SECTION .text

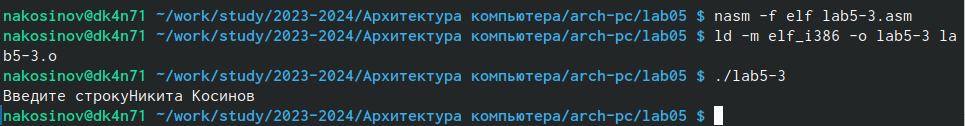
GLOBAL \_start

\_start:

mov eax, msg   
call sprint  
  
mov ecx, buf1  
mov edx, 80  
call sread  
  
call quit

’’’

1. Компилируем её и видим изменения: как и ожидалось, после вывода переменной **msg** не происходит переноса каретки на новую строку.



Результат исполнения lab5-3

#Самостоятельная работа

1. Копируем файл *lab5-1.asm* (без подключаемого файла).

[Копирование исходного файла](./image/19.png)

1. Изменяем код, добавив двоеточие после слов *Введите строку*, а также убрав символ переноса строки, чтобы потренироваться и чтобы результат выглядел логичнее. Также, добавляем группу команд по выводу значения, записанного в **buf1**.

’’’ SECTION .data

msg: DB 'Введите строку: '  
  
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,80  
int 80h  
  
mov eax,1  
mov ebx,0  
int 80h

’’’

[Сохранение кода lab5-4.asm](./image/27.png)

1. Компилируем и запускаем соответствующий исполняемый файл

[Создание исполняемого файла lab5-4 и её запуск](./image/20.png)

[Результат работы программы](./image/21.png)

1. Копируем файл *lab5-2.asm* (с подключенным файлом).

[Копирование исходного файла](./image/22.png)

1. Изменяем код, заменив команду **sprintLF** на **sprint** первый раз и добавив вывод значения, введённого пользователем.

’’’ %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, buf1  
mov edx,80  
call sprintLF  
  
call quit

’’’

[Сохранение кода lab5-5.asm](./image/23.png)

1. Проверяем полученный результат.

[Создание исполняемого файла lab5-5 и её запуск](./image/24.png)

[Результат работы программы](./image/25.png)

1. В менеджере файлов *Midnight Commander* переносим папку lab05 в папку рабочей директории согласно файловой структуре наших лабораторных работ

[Перемещение папки](./image/26.png)

1. После создания данного отчёта все изменения перемещаются в глобальный репозиторий на *Github*.

# 6 Выводы

В ходе данной лабораторной работы мы научились перемещаться по файловой системе с помощью программы *Midnight Commander*, создавать папки и файлы, изменять и сохранять документы встроенным редактором *mcedit*, а также изучили работу инструкций **mov** и **int** языка Ассемблер.