Отчёт по лабораторной работе №5

Дисциплина: Архитектура компьютера

Зарицкая Марина Петровна

Содержание

# 1 Цель работы

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 in\_out.asm
4. Задание 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 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).

# 4 Выполнение лабораторной работы

## 4.1 Основы работы с Midnight Commander

Открываю Midnight Commander с помощью команды mc (рис. [1](#fig%3A001)).

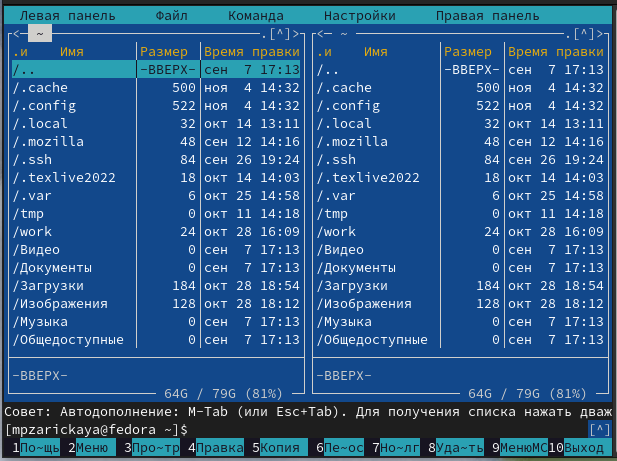


Рис. 1: Окно Midnight Commander

Перехожу в каталог ~/work/arch-pc созданный при выполнении лабораторной работы №4. С помощью функциональной клавиши F7 создаю каталог lab05 (рис. [2](#fig%3A002)).

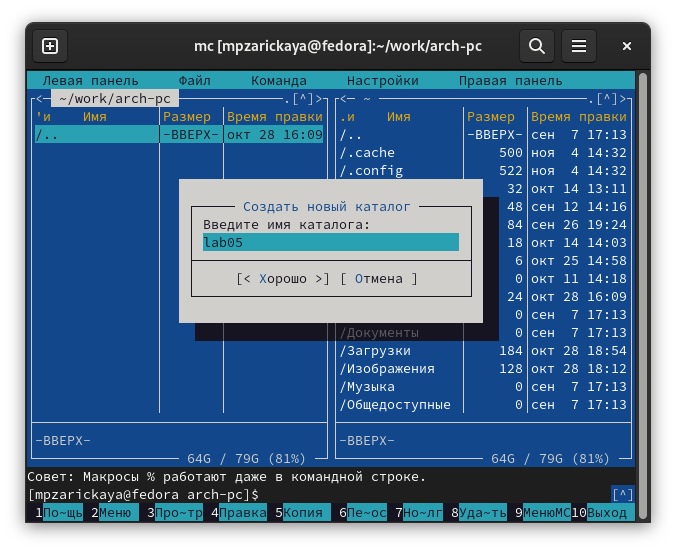


Рис. 2: Создание каталога

Пользуясь строкой ввода и командой touch создаю файл lab5-1.asm (рис. [3](#fig%3A003)).

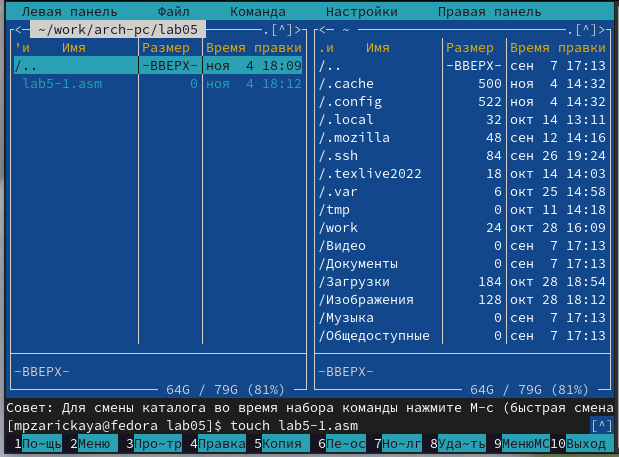


Рис. 3: Создание файла lab5-1.asm

## 4.2 Структура программы на языке ассемблера NASM

С помощью функциональной клавиши F4 открываю созданный файл для редактирования в редакторе mcedit. Ввожу в файл код программы вывода сообщения на экран и ввода строки с клавиатуры, сохраняю изменения и закрываю файл (рис. [4](#fig%3A004)).

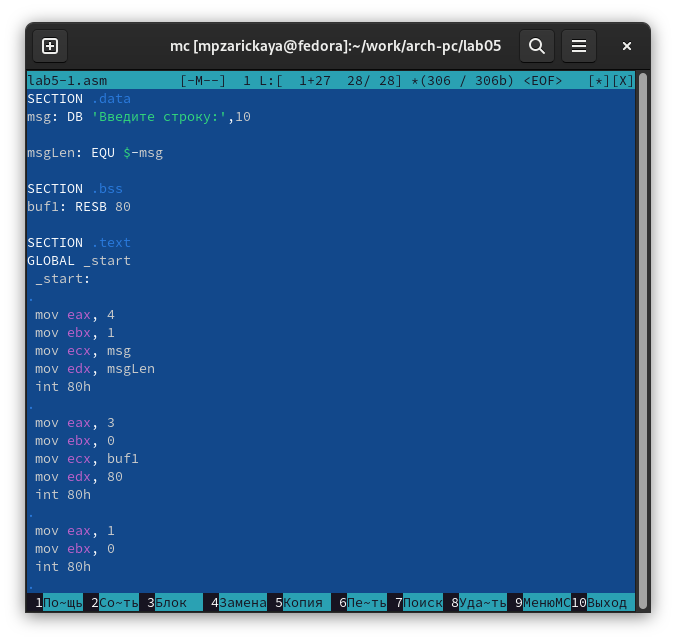


Рис. 4: Редактирование файла

С помощью функциональной клавиши F3 открываю файл для просмотра, чтобы проверить, содержит ли файл текст программы (рис. [5](#fig%3A005)).

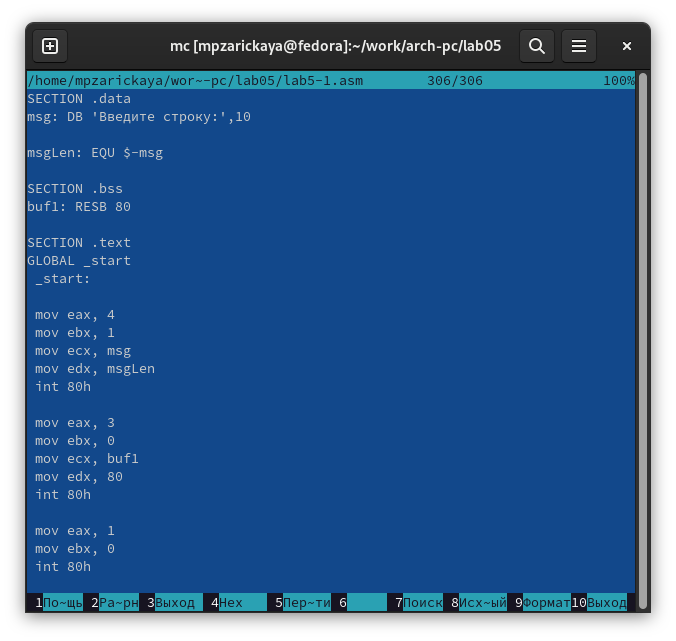


Рис. 5: Открытие файла для просмотра

Транслирую текст программы файла в объектный файл командой nasm -f elf lab5-1.asm (рис. [6](#fig%3A006)).

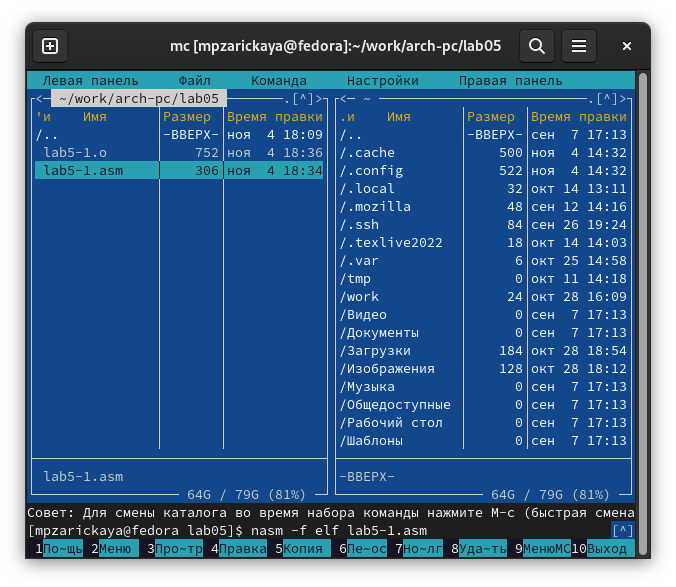


Рис. 6: Создание объектного файла lab5-1.o

Выполняю компоновку объектного файла (рис. [7](#fig%3A007)).

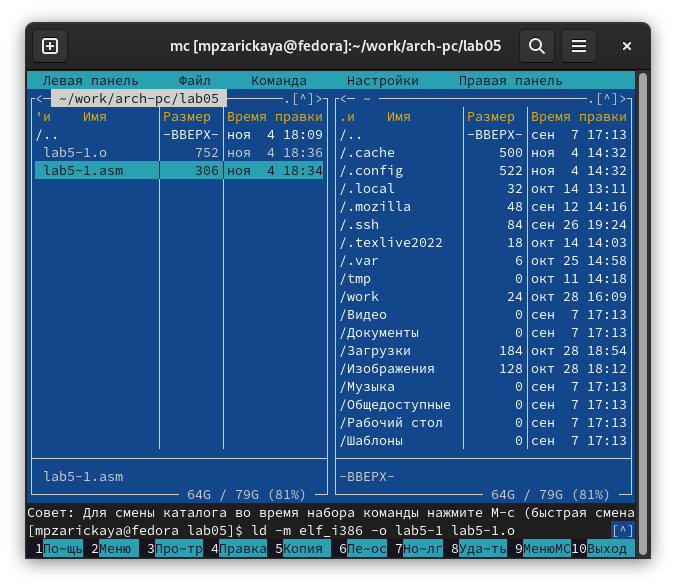


Рис. 7: Компоновка объектного файла

Запускаю исполняемый файл. Программа выводит строку “Введите строку:” и ждет ввода с клавиатуры, я ввожу свои ФИО и программа заканчивает свою работу (рис. [8](#fig%3A008)).

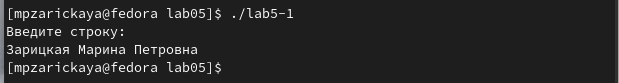


Рис. 8: Запуск программы

## 4.3 Подключение внешнего файла in\_out.asm

Скачиваю файл in\_out.asm со страницы курса в ТУИС. С помощью функциональной клавиши F5 копирую файл из каталога Загрузки в созданный каталог lab05 (рис. [9](#fig%3A009)).

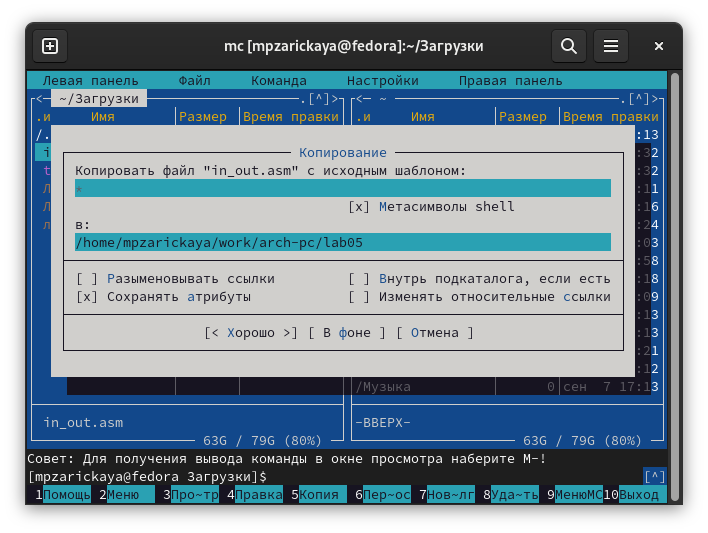


Рис. 9: Копирование файла

С помощью функциональной клавиши F5 создаю копию файла lab5-1.asm с именем lab5-2.asm (рис. [10](#fig%3A010)).

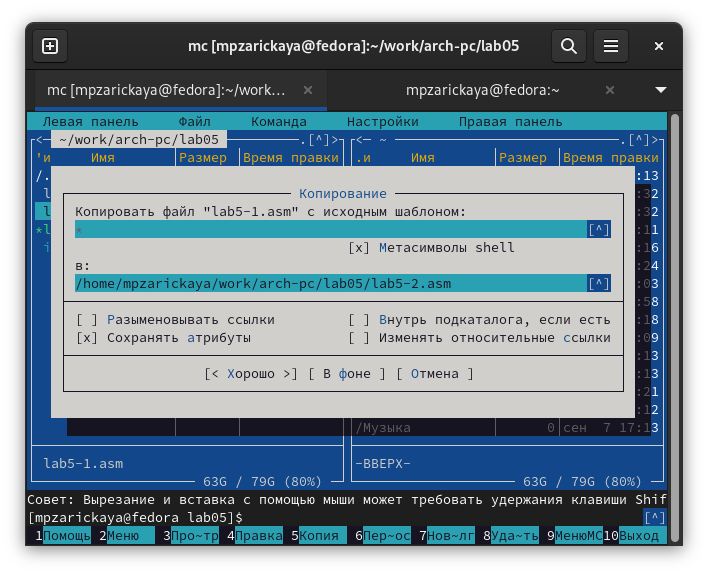


Рис. 10: Создание копии файла

Изменяю содержимое файла lab5-2.asm во встроенном редакторе mcedit так, чтобы в программе использовались подпрограммы из внешнего файла in\_out.asm (рис. [11](#fig%3A011)).

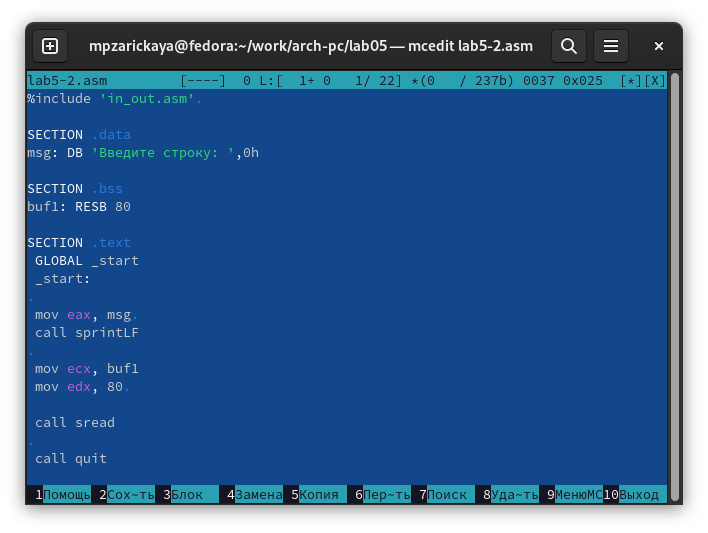


Рис. 11: Изменение файла

Транслирую текст программы файла в объектный файл командой nasm -f elf lab5-2.asm. Создался объектный файл lab5-2.o. Выполняю компоновку объектного файла с помощью команды ld -m elf\_i386 -o lab5-2 lab5-2.o Создался исполняемый файл lab5-2. Запускаю исполняемый файл (рис. [12](#fig%3A012)).

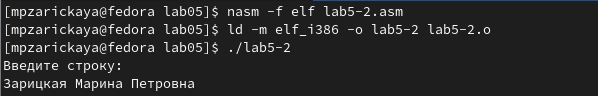


Рис. 12: Исполнение файла

Изменяю в файле lab5-2.asm подпрограмму sprintLF на sprint. Сохраняю изменения и открываю файл для просмотра, чтобы проверить сохранение действий (рис. [13](#fig%3A013)).

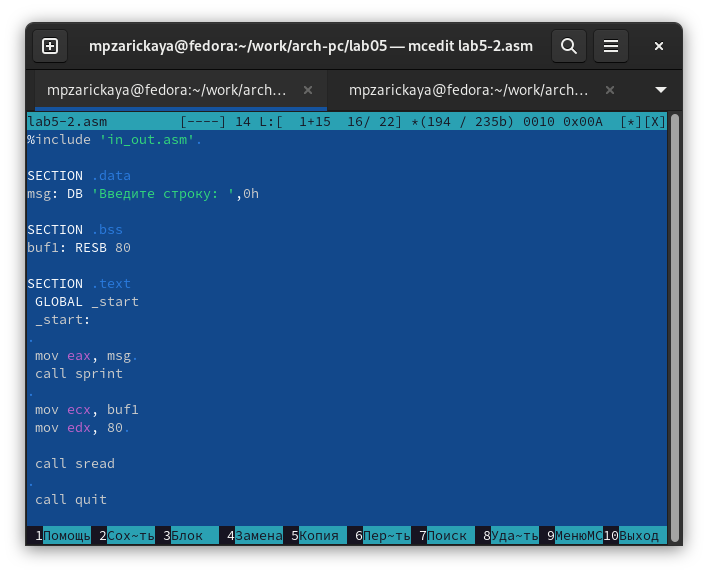


Рис. 13: Отредактированный файл

Транслирую файл, выполняю компоновку созданного объектного файла, запускаю программу (рис. [14](#fig%3A014)).

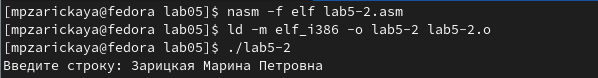


Рис. 14: Исполнение файла

Различие между первым и вторым исполняемыми файлами в том, что запуск первого запрашивает ввод с новой строки, а второго запрашивает ввод без переноса на новую строку. В этом и заключается разница между подпрограммами sprintLF и sprint.

## 4.4 Задание для самостоятельной работы

1. Создаю копию файла lab5-1.asm с именем lab5-1-1.asm с помощью функциональной клавиши F5 (рис. [15](#fig%3A015)).

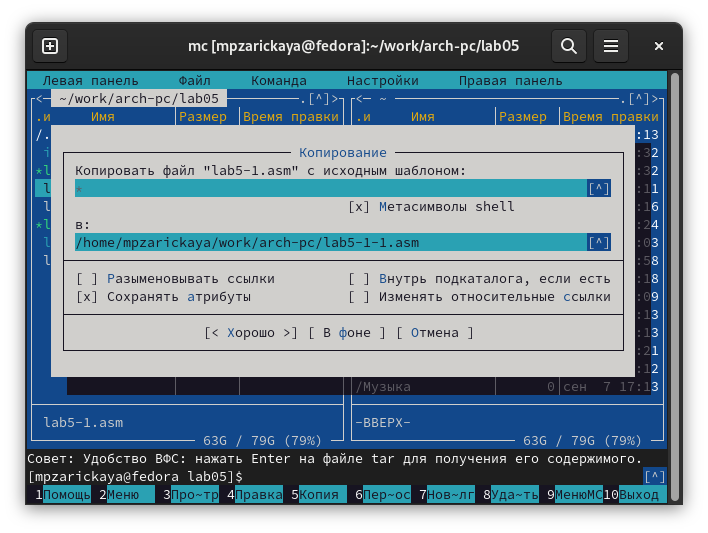


Рис. 15: Копирование файла

Открываю созданный файл в mcedit для редактирования. Изменяю программу так, чтобы кроме вывода приглашения и запроса ввода, она выводила вводимую пользователем строку (рис. [16](#fig%3A016)).

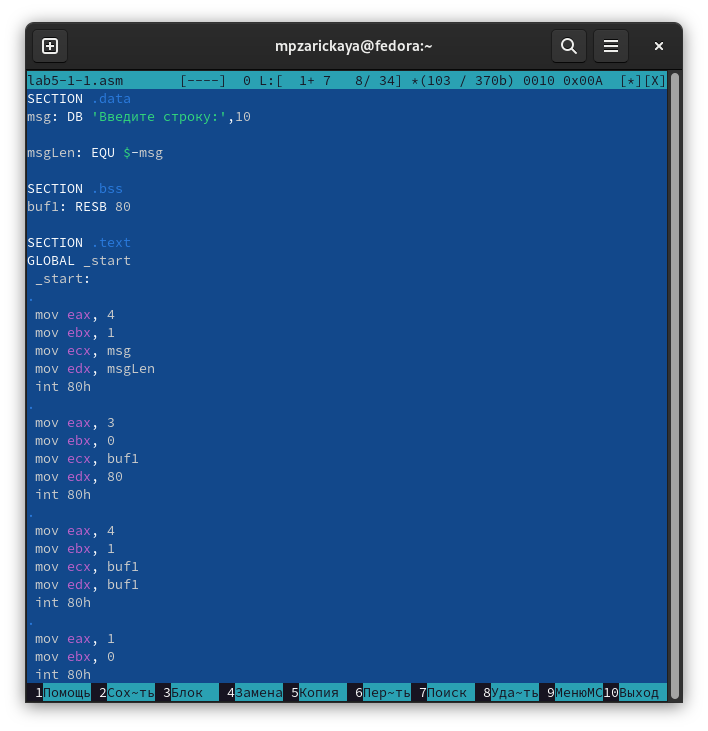


Рис. 16: Редактирование файла

1. Создаю объектный файл lab5-1-1.o, отдаю его на обработку компоновщику, получаю исполняемый файл lab5-1-1, запускаю полученный исполняемый файл. Программа запрашивает ввод, ввожу свои ФИО, далее программа выводит введенные мною данные (рис. [17](#fig%3A017)).

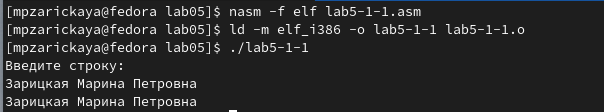


Рис. 17: Исполнение файла

Листинг первой программы:

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

1. Создаю копию файла lab5-2.asm с именем lab5-2-1.asm с помощью функциональной клавиши F5 (рис. [18](#fig%3A018)).

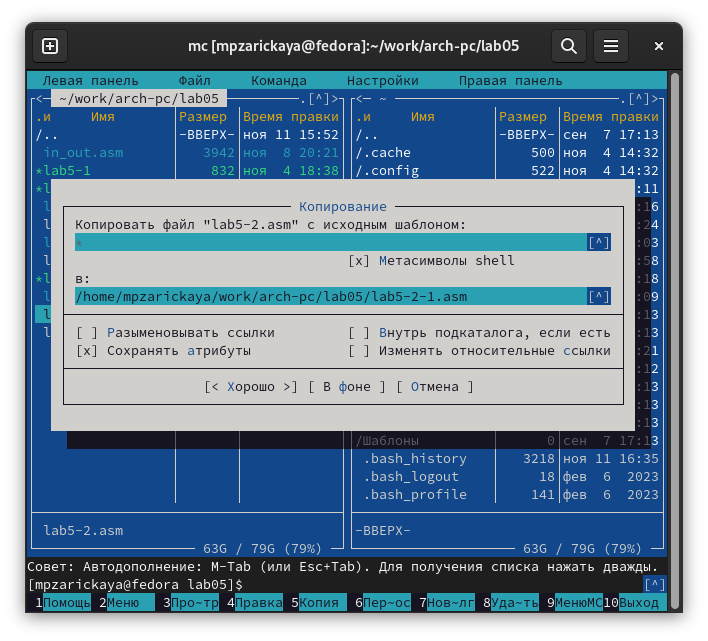


Рис. 18: Копирование файла

Открываю созданный файл в mcedit для редактирования. Изменяю программу так, чтобы кроме вывода приглашения и запроса ввода, она выводила вводимую пользователем строку (рис. [19](#fig%3A019)).

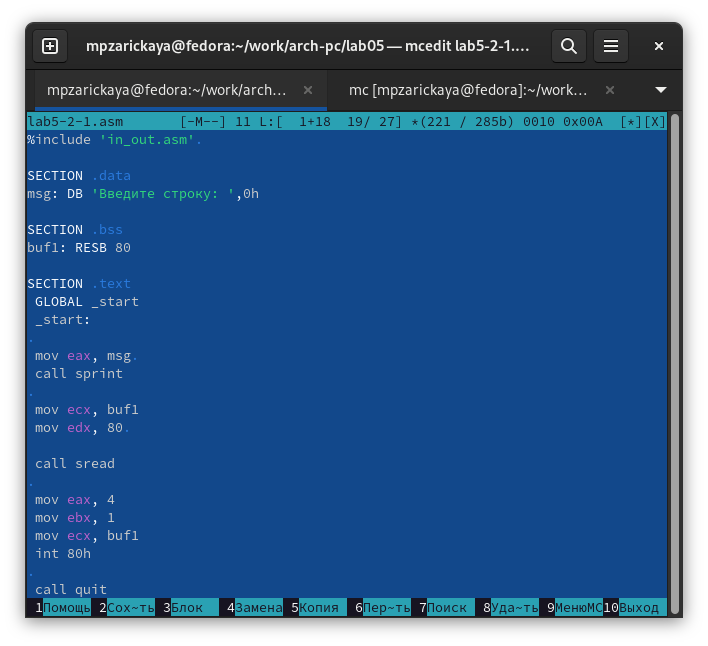


Рис. 19: Редактирование файла

1. Создаю объектный файл lab5-2-1.o, отдаю его на обработку компоновщику, получаю исполняемый файл lab5-2-1, запускаю полученный исполняемый файл. Программа запрашивает ввод без переноса на новую строку, ввожу свои ФИО, далее программа выводит введенные мною данные (рис. [20](#fig%3A020)).

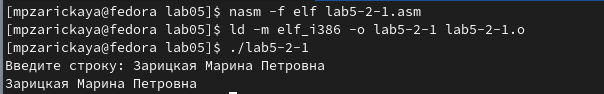


Рис. 20: Исполнение файла

Листинг второй программы:

%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

# 5 Выводы

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