Отчёт по лабораторной работе № 5

дисциплина: Архитектура компьютера. Основы работы с *Midnight Commander* (mc). Структура программы на языке ассемблера NASM. Системные вызовы в OC GNU Linux

Студент: Святашова Ксения Евгеньевна

Содержание

# 1 Цель работы

Целью работы является приобретение практических навыков работы в *Midnight Commander* и освоение языка ассемблера mov и int.

# 2 Теоритическое 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).

# 3 Выполнение лабораторной работы

## 3.1 Midnight Commander

1. Откроем *Midnight Commander*(рис. 1):

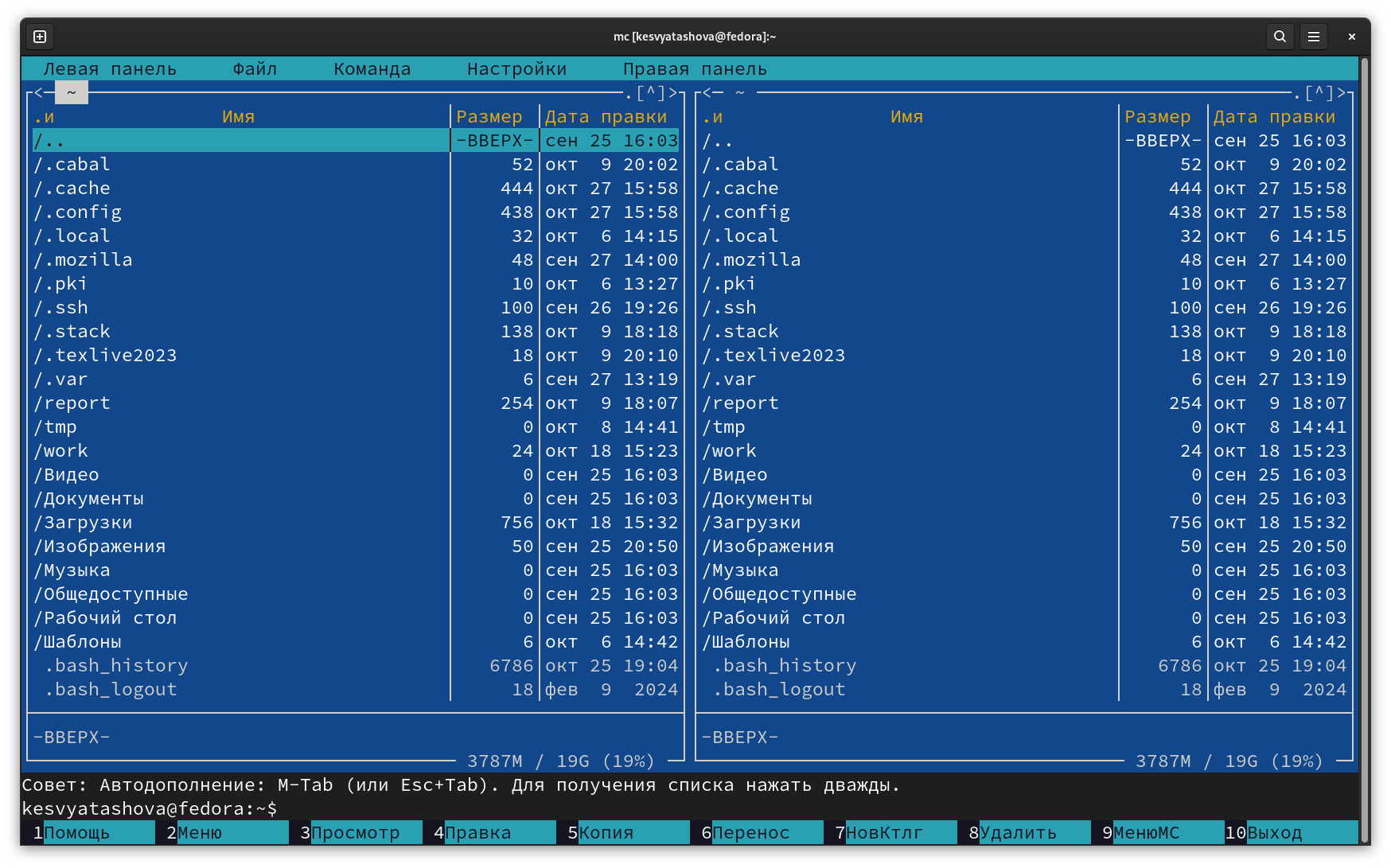


Рис. 1: Команда mc

1. Пользуясь клавишами на клавиатуре перейдем в каталог ~/work/arch-pc созданный при выполнении лабораторной работы №4(рис. 2):

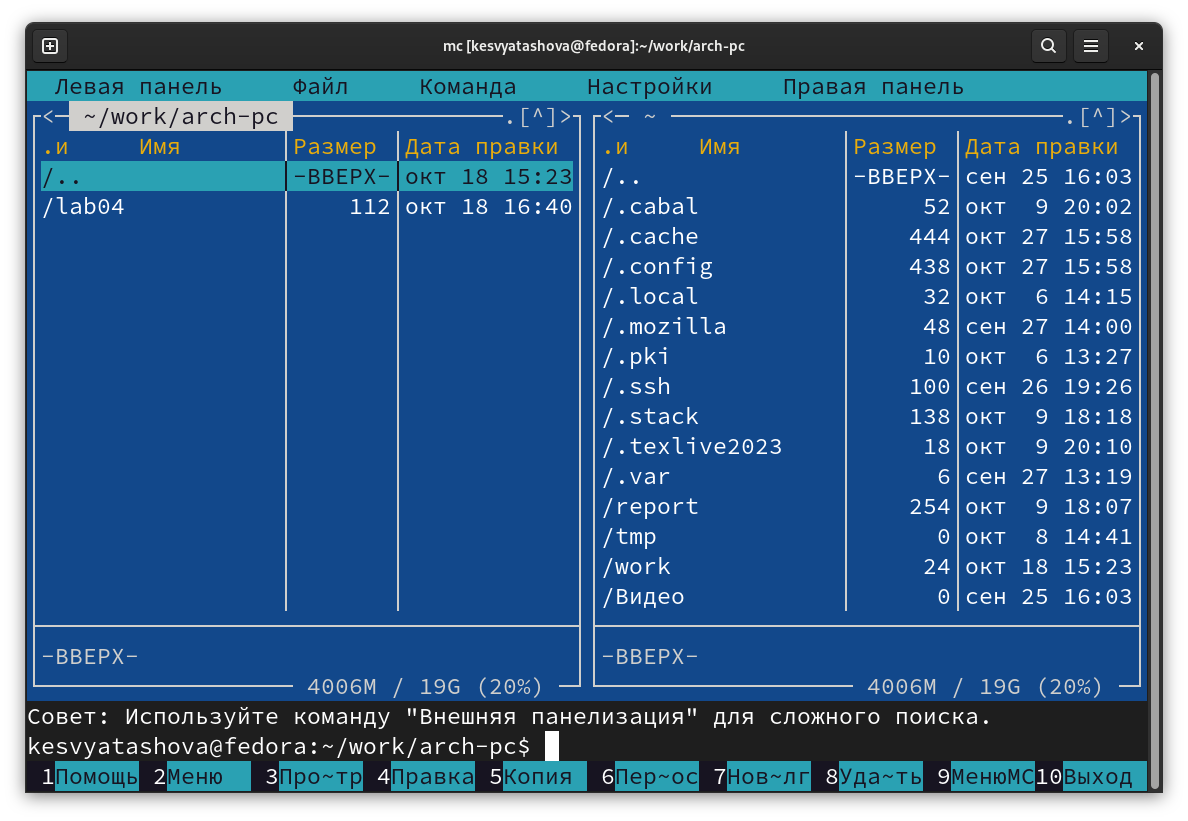


Рис. 2: Каталог arch-pc

1. С помощью функциональной клавиши F7 создадим папку lab05(рис. 3) и перейдем в созданный каталог(рис. 4):

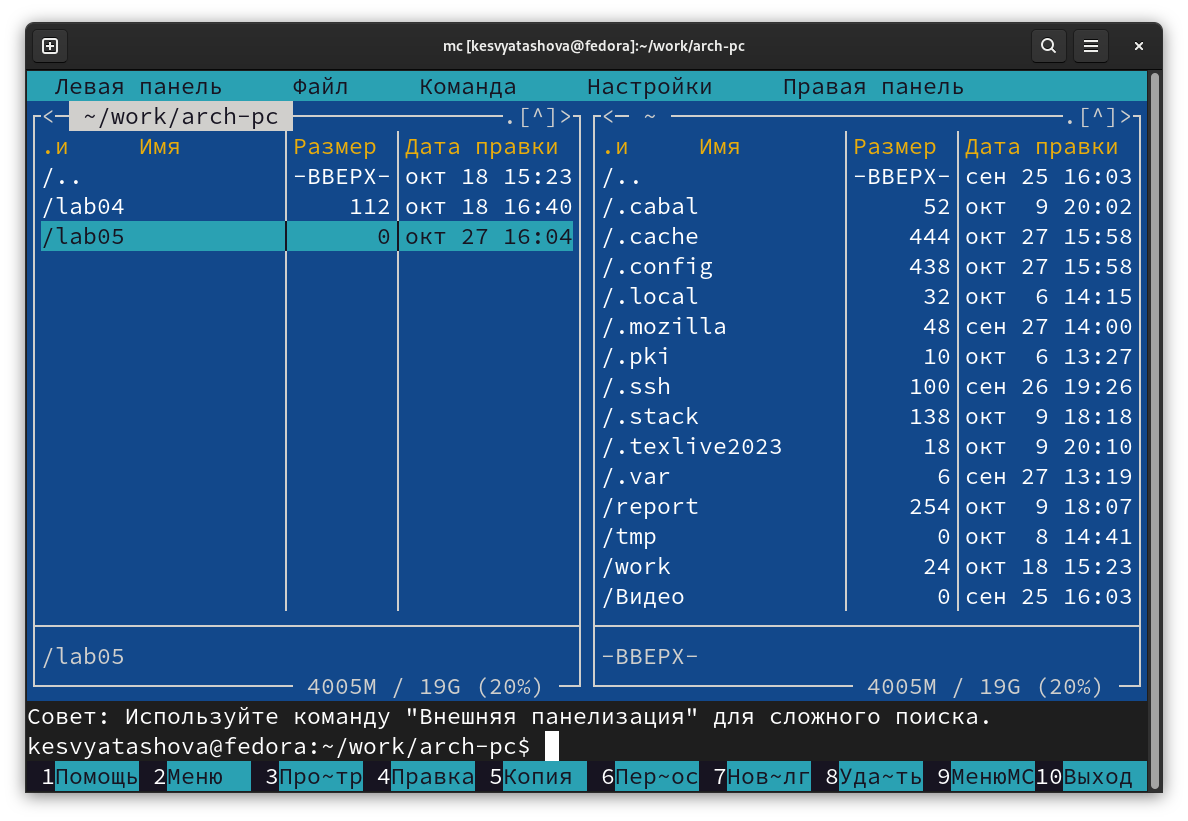


Рис. 3: Создание папки

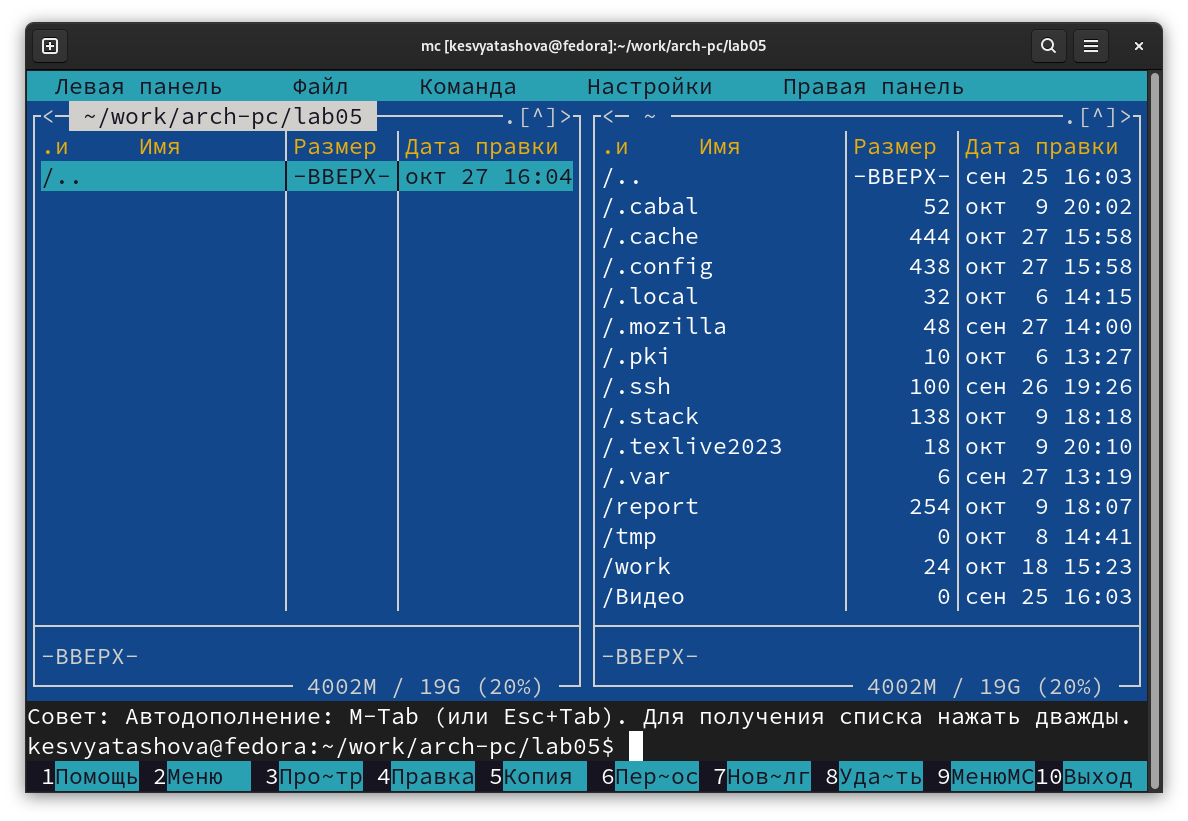


Рис. 4: Папка

1. Пользуясь строкой ввода и коммандой touch создадим файл lab5-1.asm(рис. 5):

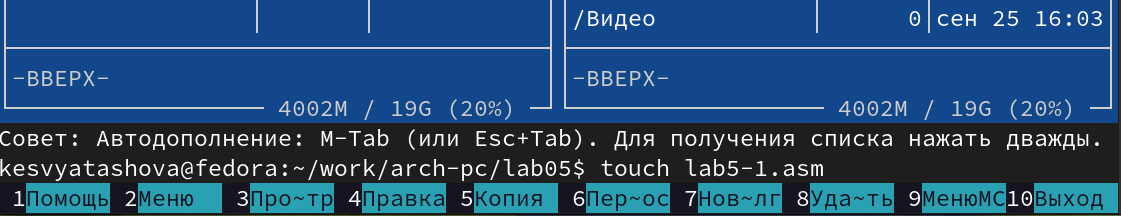


Рис. 5: Создание файла lab5-1.asm

1. С помощью функциональной клавиши F4 откроем файл lab5-1.asm для редактирования во встроенном редакторе mcedit(рис. 6):

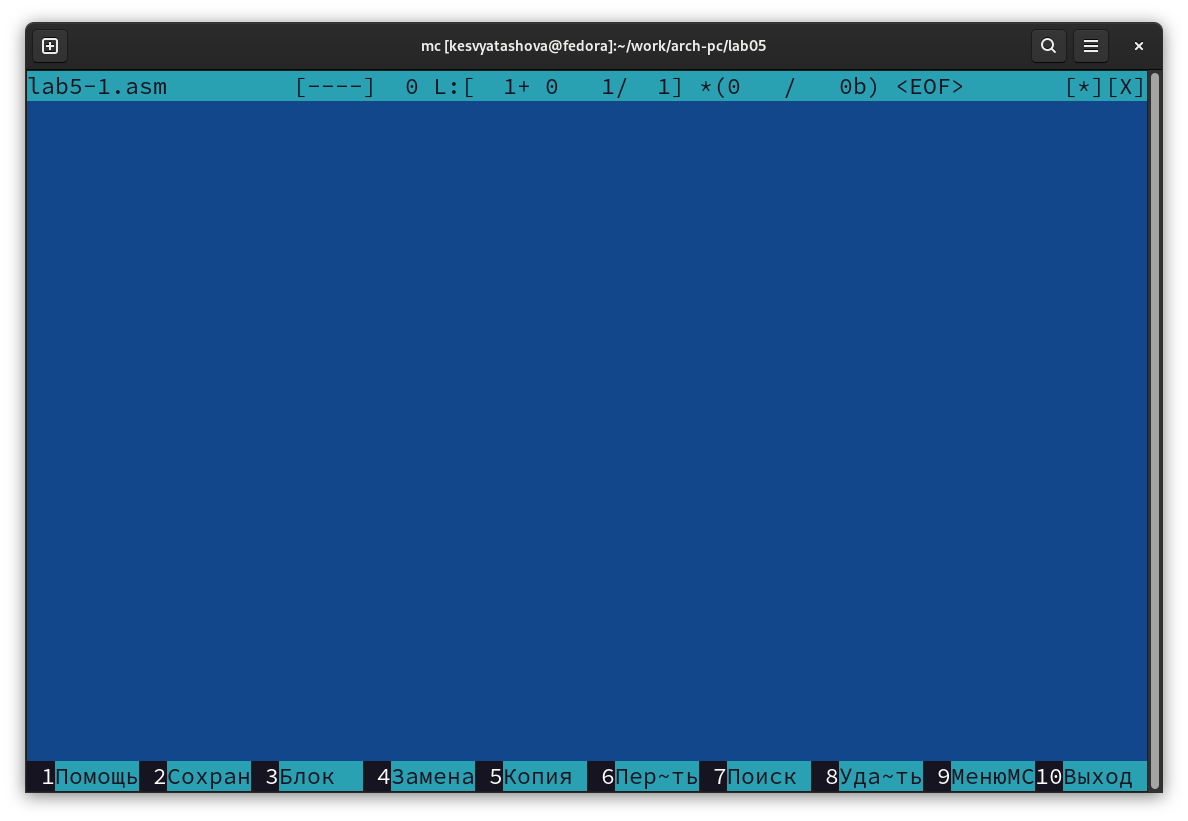


Рис. 6: Файл lab5-1.asm

1. Введем текст программы из листинга 1.1(рис. 7), сохраним изменения(рис. 8) и закроем файл:

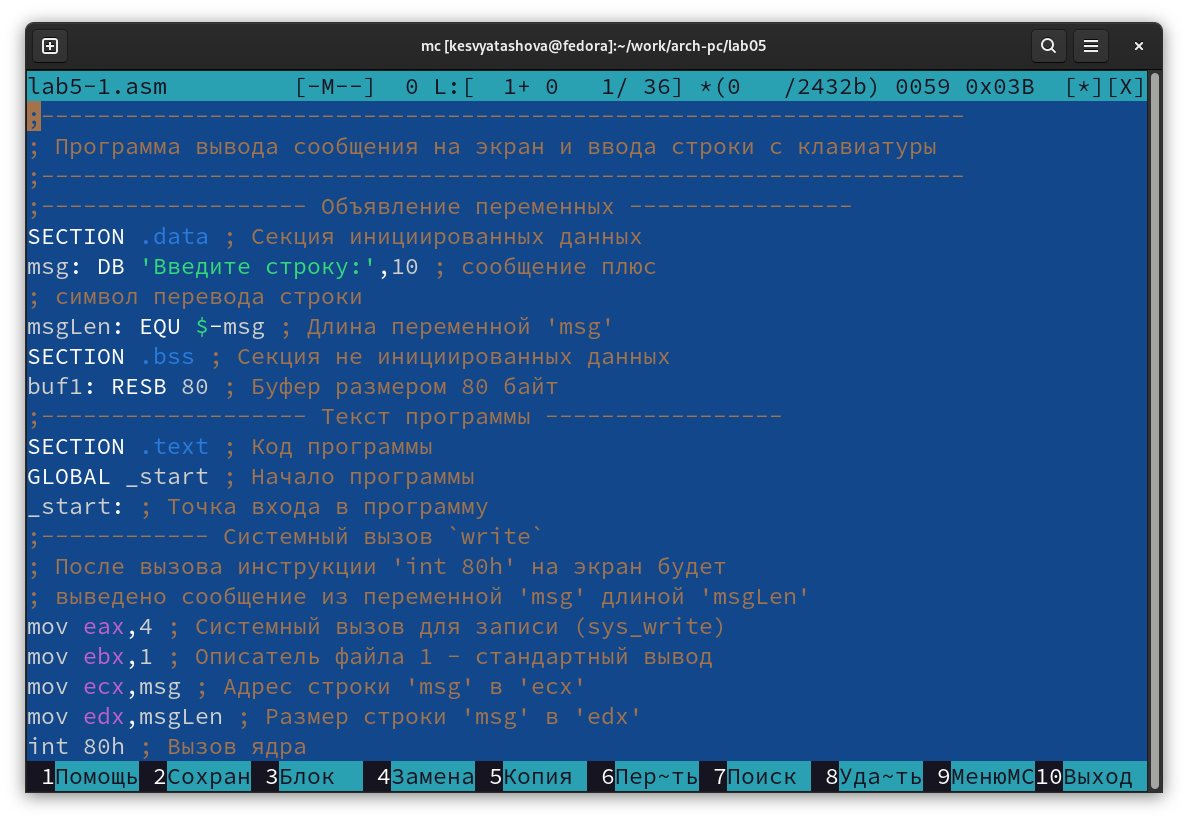


Рис. 7: Ввод текста из листинга

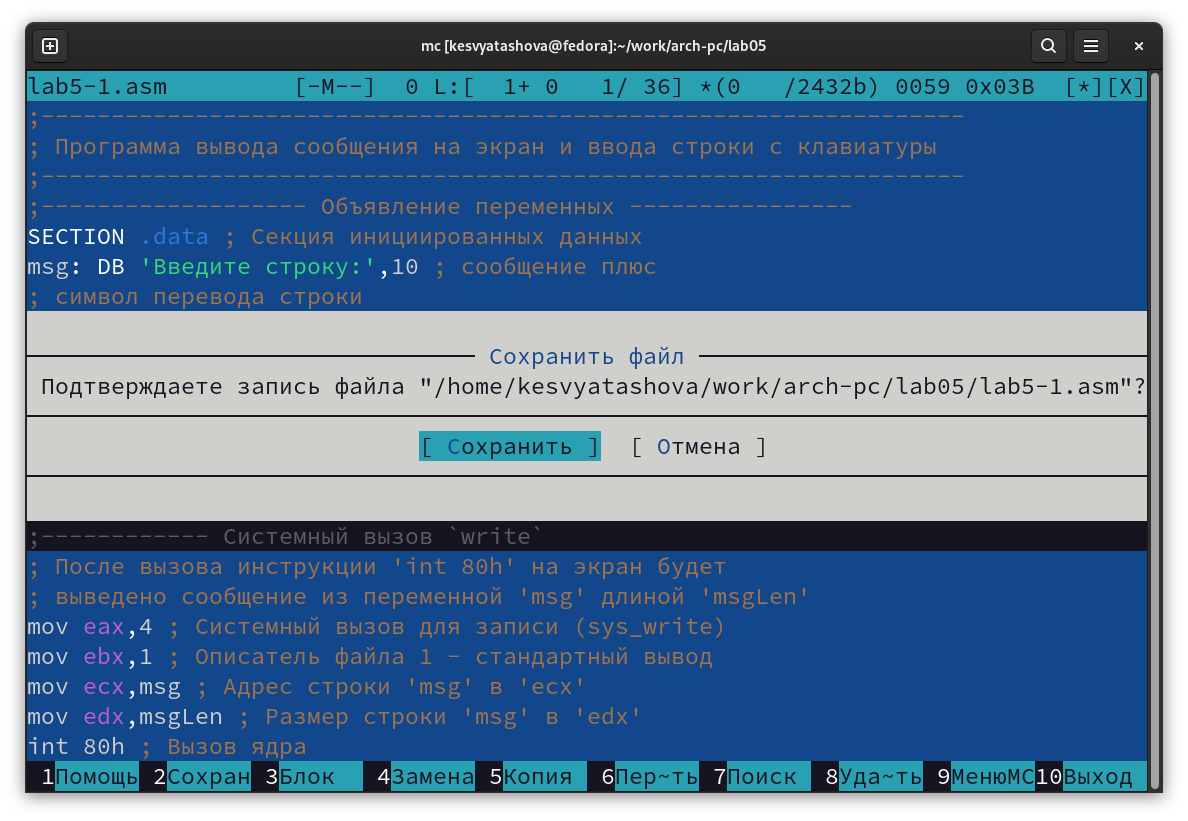


Рис. 8: Сохранения файла с текстом из листинга

**Листинг 1.1. Программа вывода сообщения на экран и ввода строки с клавиатуры**

;—————————————————————— ; Программа вывода сообщения на экран и ввода строки с клавиатуры ;—————————————————————— ;——————- Объявление переменных —————- SECTION .data ; Секция инициированных данных msg: DB ‘Введите строку:’,10 ; сообщение плюс ; символ перевода строки msgLen: EQU $-msg ; Длина переменной ‘msg’ SECTION .bss ; Секция не инициированных данных buf1: RESB 80 ; Буфер размером 80 байт ;——————- Текст программы —————– SECTION .text ; Код программы GLOBAL \_start ; Начало программы \_start: ; Точка входа в программу ;———— Cистемный вызов write ; После вызова инструкции ‘int 80h’ на экран будет ; выведено сообщение из переменной ‘msg’ длиной ‘msgLen’ 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 ;———— системный вызов read ———————- ; После вызова инструкции ‘int 80h’ программа будет ожидать ввода ; строки, которая будет записана в переменную ‘buf1’ размером 80 байт 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 ;———— Системный вызов exit ———————- ; После вызова инструкции ‘int 80h’ программа завершит работу mov eax,1 ; Системный вызов для выхода (sys\_exit) mov ebx,0 ; Выход с кодом возврата 0 (без ошибок) int 80h ; Вызов ядра

1. С помощью функциональной клавиши F3 откройте файл lab5-1.asm для просмотра. Убедимся, что файл содержит текст программы(рис. 9):

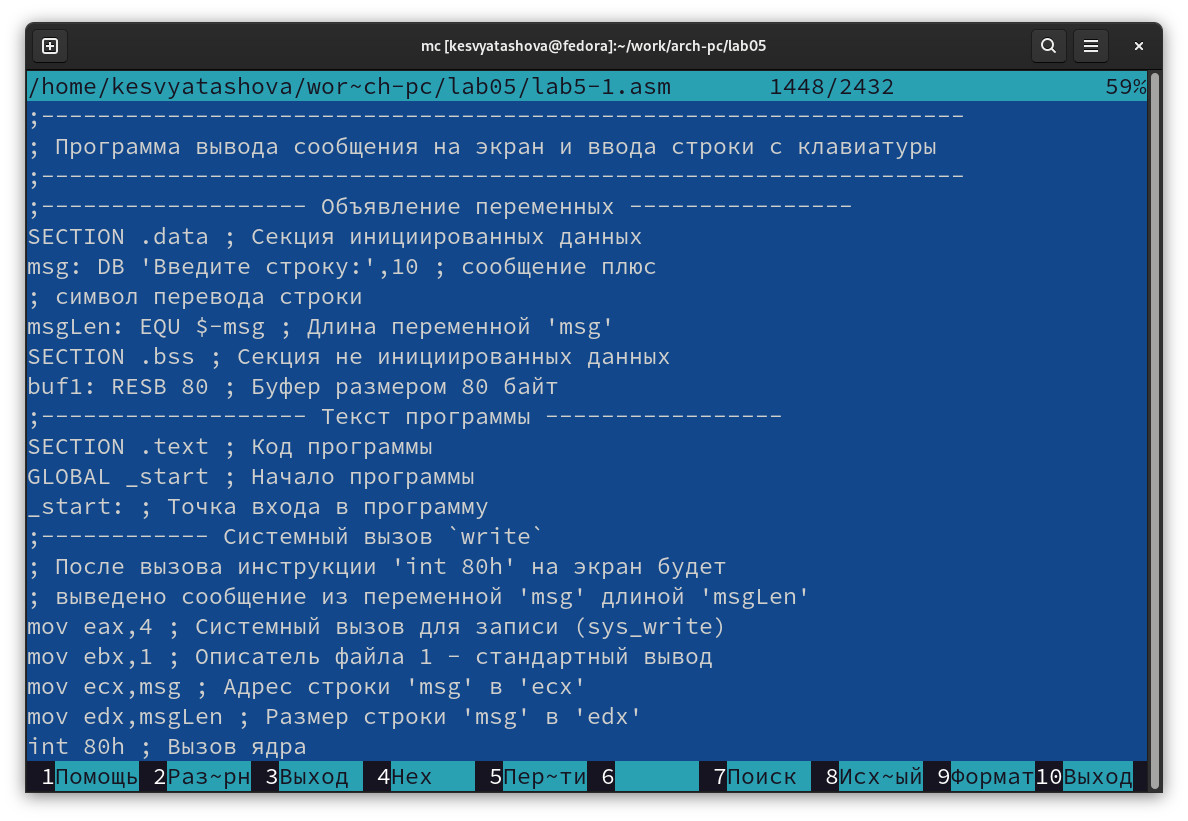


Рис. 9: Проверка файла

1. Оттранслируем текст программы lab5-1.asm в объектный файл. Выполним компоновку объектного файла и запустим получившийся исполняемый файл. Программа выводит строку ‘Введите строку:’ и ожидает ввода с клавиатуры. На запрос введу мое ФИО(рис. 10):

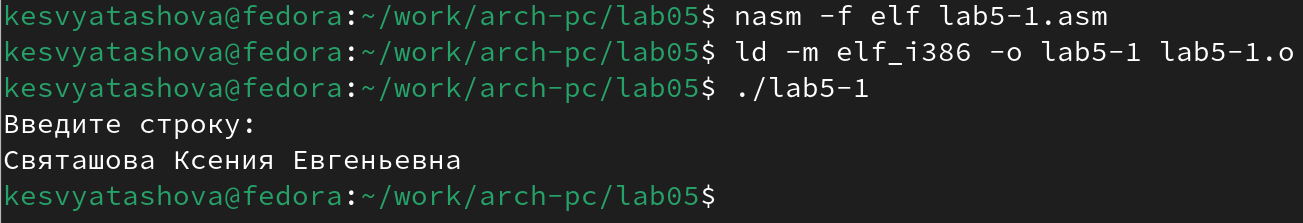


Рис. 10: Программа

## 3.2 Подключение внешнего файла in\_out.asm

Для упрощения написания программ часто встречающиеся одинаковые участки кода (такие как, например, вывод строки на экран или выход их программы) можно оформить в виде подпрограмм и сохранить в отдельные файлы, а во всех нужных местах поставить вызов нужной подпрограммы. Это позволяет сделать основную программу более удобной для написания и чтения. NASM позволяет подключать внешние файлы с помощью директивы %include, которая предписывает ассемблеру заменить эту директиву содержимым файла. Подключаемые файлы также написаны на языке ассемблера. Важно отметить, что директива %include в тексте программы должна стоять раньше, чем встречаются вызовы подпрограмм из подключаемого файла. Для вызова подпрограммы из внешнего файла используется инструкция call, которая имеет следующий вид

call где function имя подпрограммы

1. Скачаем файл in\_out.asm со страницы курса в ТУИС.
2. Подключаемый файл in\_out.asm должен лежать в том же каталоге, что и файл с программой, в которой он используется.

В одной из панелей mc откроем каталог с файлом lab5-1.asm. В другой панели каталог со скаченным файлом in\_out.asm(рис. 11). Скопируем этот файл в каталог с файлом lab5-1.asm с помощью функциональной клавиши F5(рис. 12):

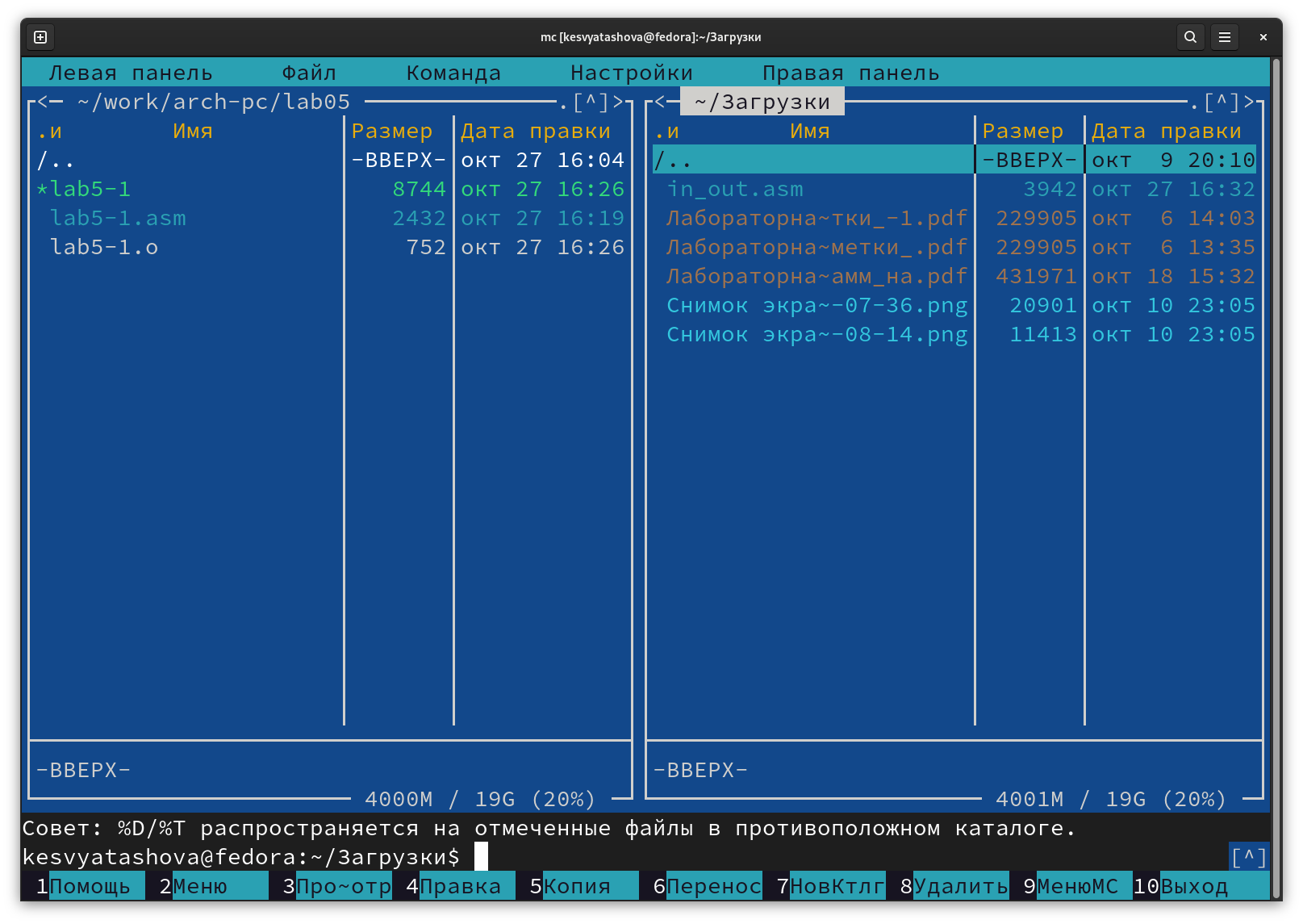


Рис. 11: Каталог с файлом in\_out.asm

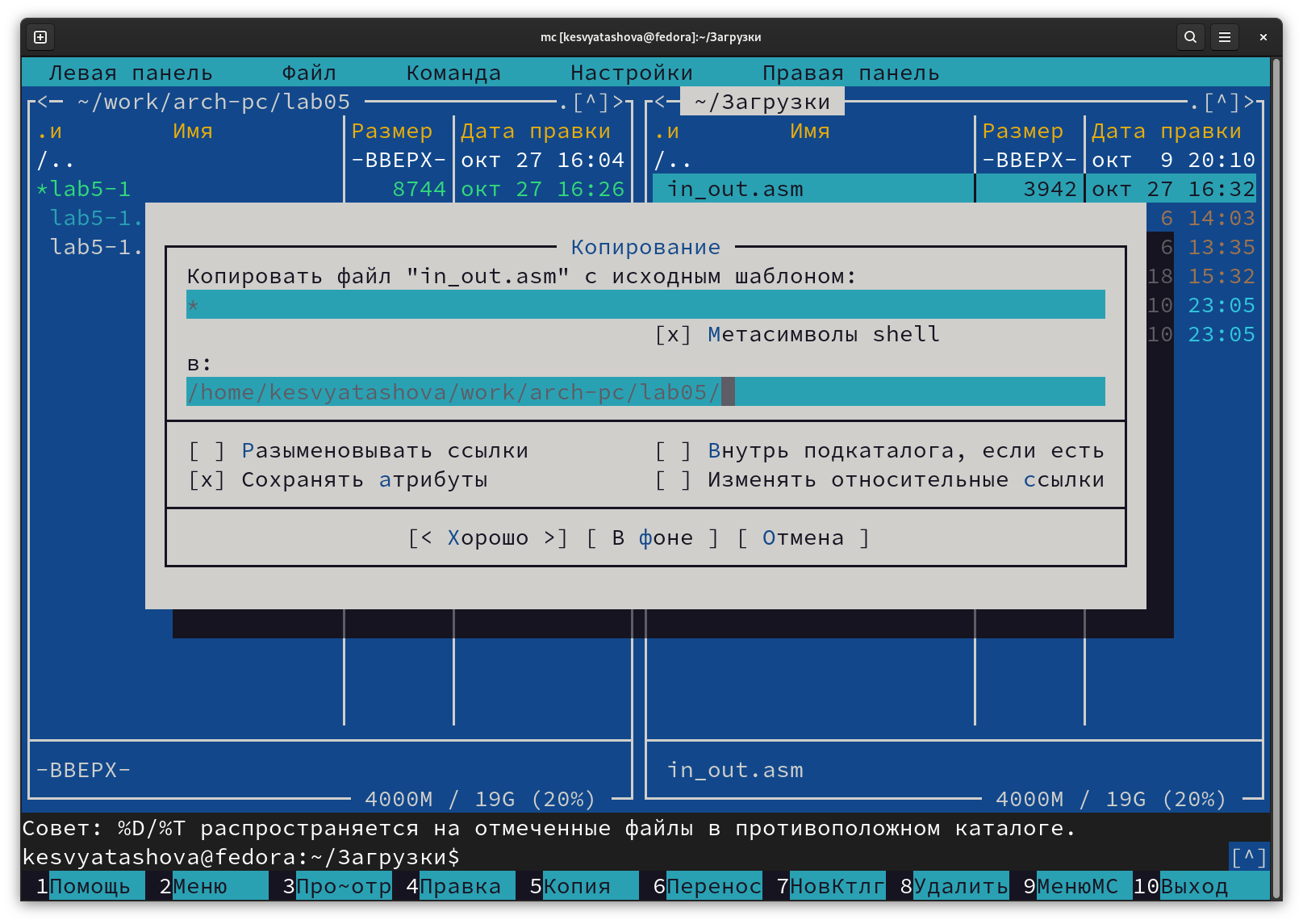


Рис. 12: Копирование файла

1. С помощью функциональной клавиши F6 создадим копию файла lab5-1.asm с именем lab5-2.asm. Выделим файл lab5-1.asm, нажмем клавишу F6 , введем имя файла lab5-2.asm и нажмем клавишу Enter(рис. 13):

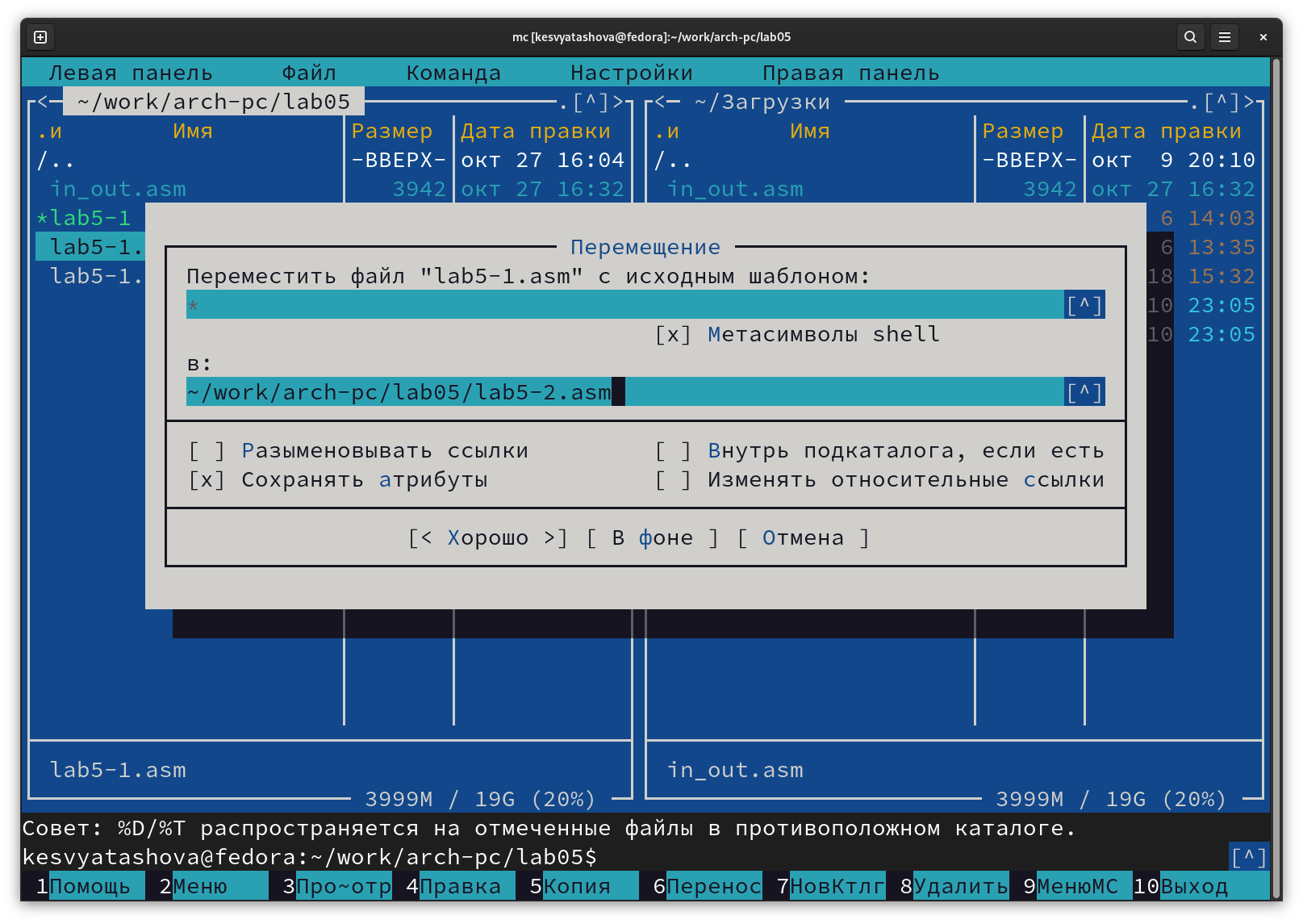


Рис. 13: Копия файла

1. Исправим текст программы в файле lab5-2.asm с использование подпрограмм из внешнего файла in\_out.asm (используем подпрограммы sprintLF, sread и quit) в соответствии с листингом 1.2(рис. 14). Создим исполняемый файл и проверим его работу(рис. 15):

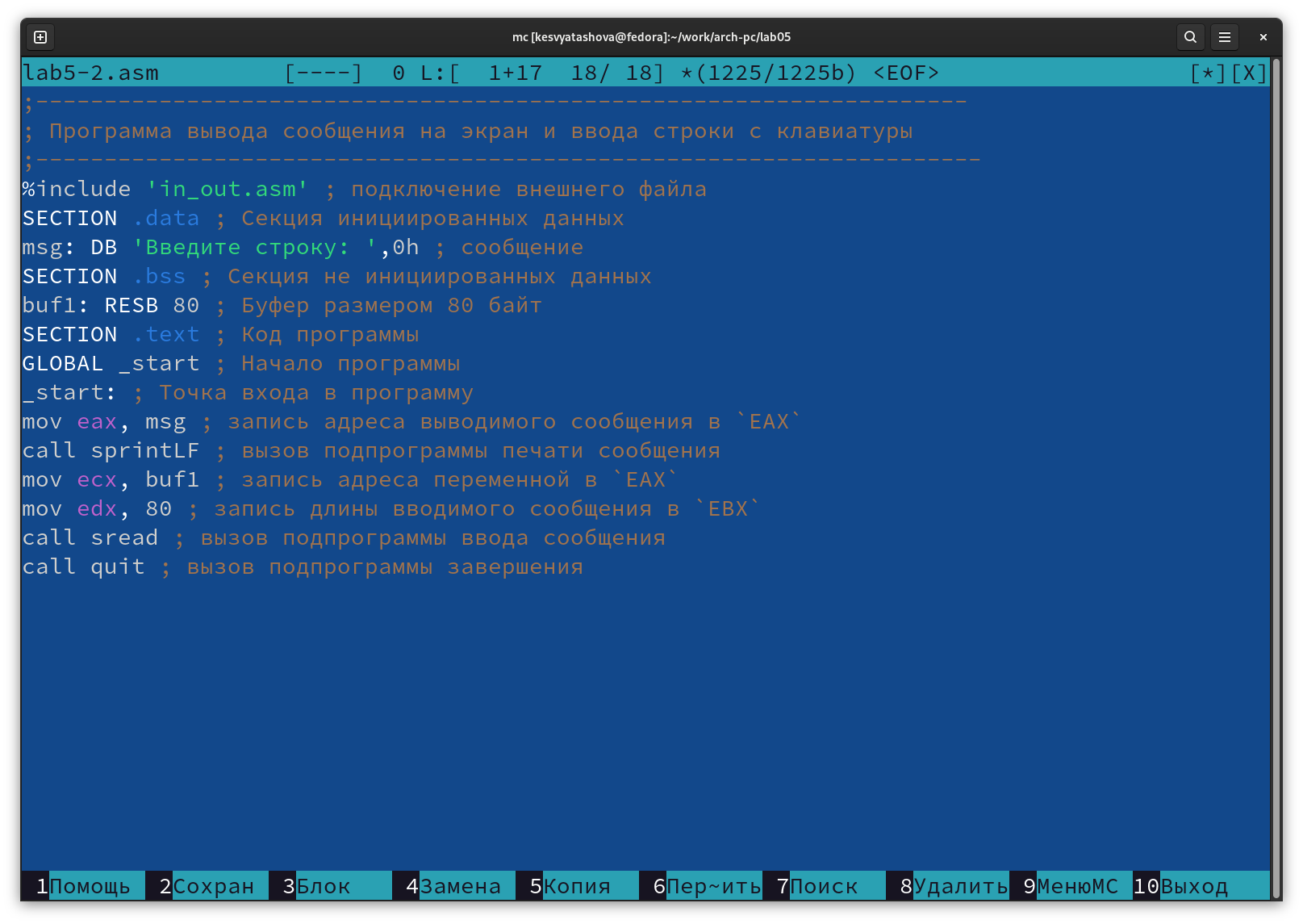


Рис. 14: Исправление текста

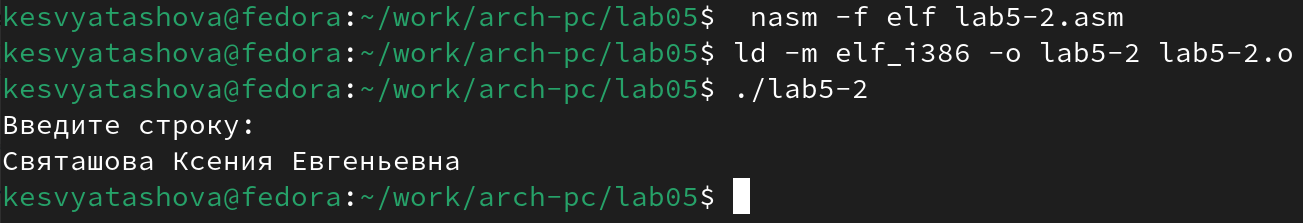


Рис. 15: Проверка работы программы

**Листинг 1.2. Программа вывода сообщения на экран и ввода строки с клавиатуры c использованием файла in\_out.asm**

;——————————————————————– ; Программа вывода сообщения на экран и ввода строки с клавиатуры ;——————————————————————— %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 sprintLF ; 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 завершения

1. В файле lab5-2.asm заменим подпрограмму sprintLF на sprint(рис. 16). Создадим исполняемый файл и проверим его работу(рис. 17).

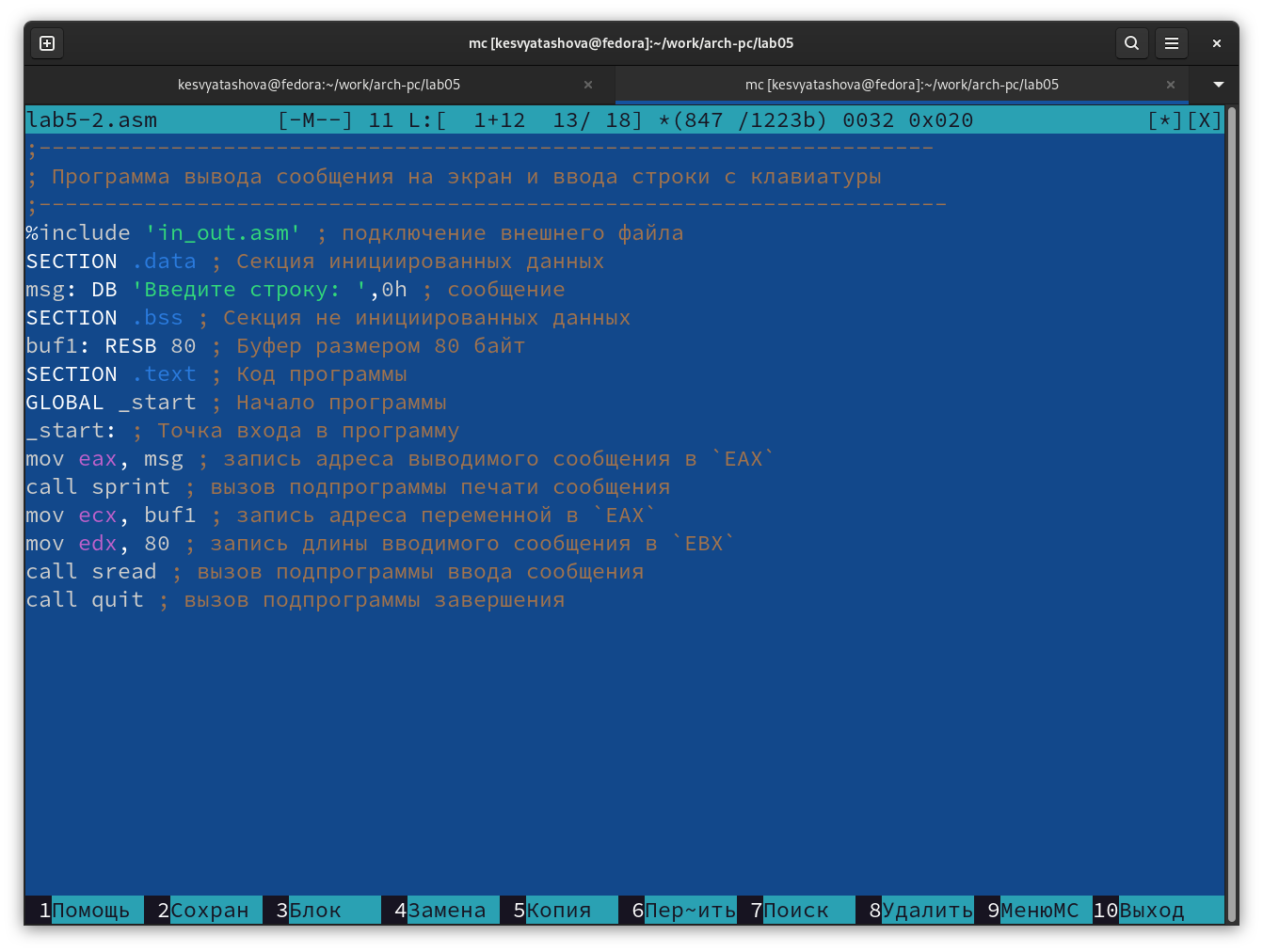


Рис. 16: Замена sprintLF на sprint

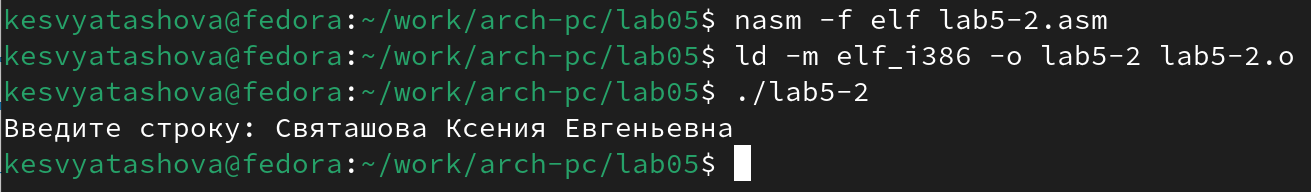


Рис. 17: Проверка программы

Разница между первым исполняемым файлом lab5-2 и вторым lab5-2-2 в том, что запуск первого запрашивает ввод с новой строки, а программа, которая исполняется при запуске второго, запрашивает ввод без переноса на новую строку.

# 4 Задания для самостоятельной работы

1. Создадим копию файла lab5-1.asm(рис. 18). Внесем изменения в программу(рис. 19)(без использования внешнего файла in\_out.asm), так чтобы она работала по следующему алгоритму: • вывести приглашение типа “Введите строку:”; • ввести строку с клавиатуры; • вывести введённую строку на экран.

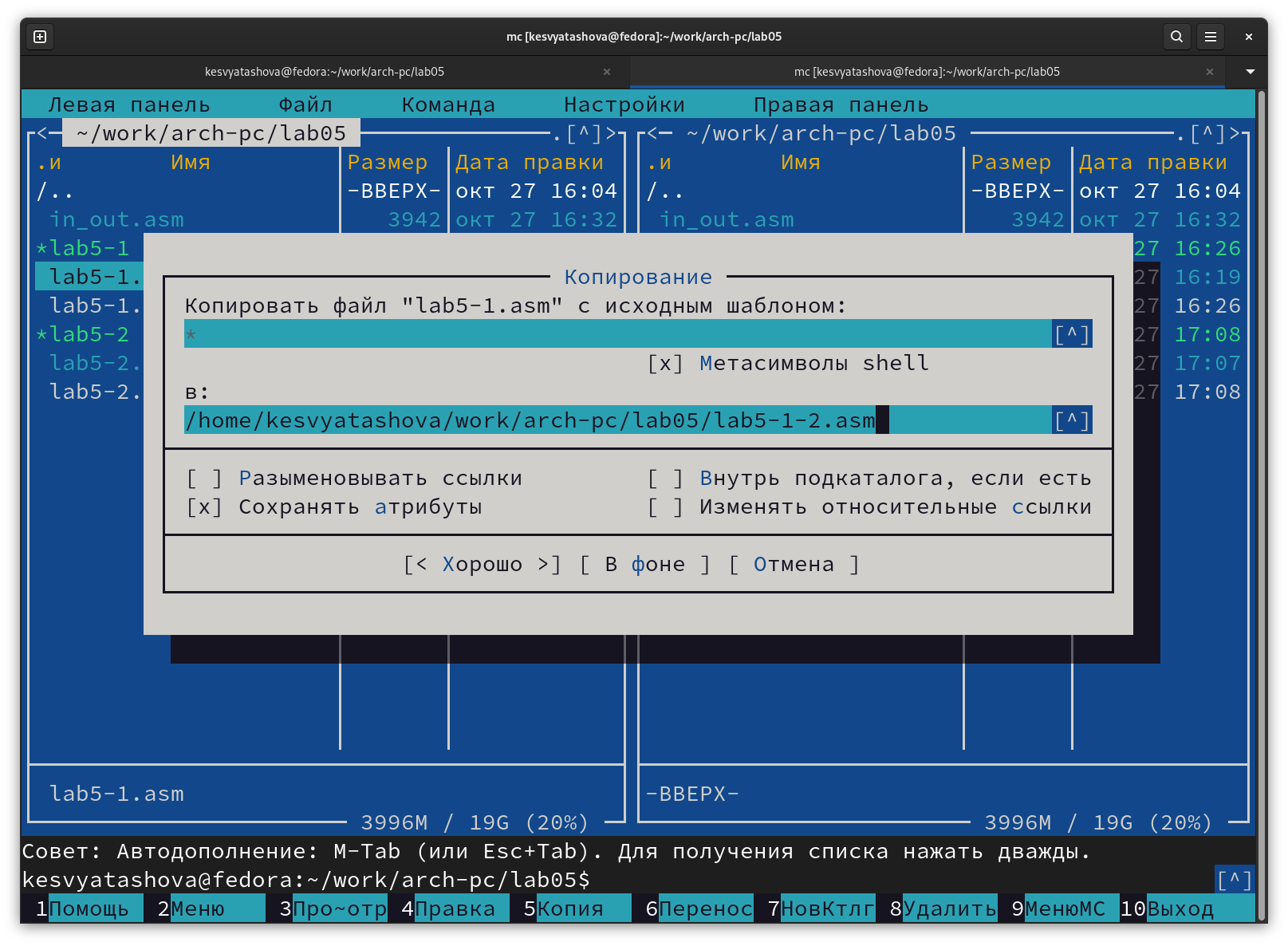


Рис. 18: Копирование файла

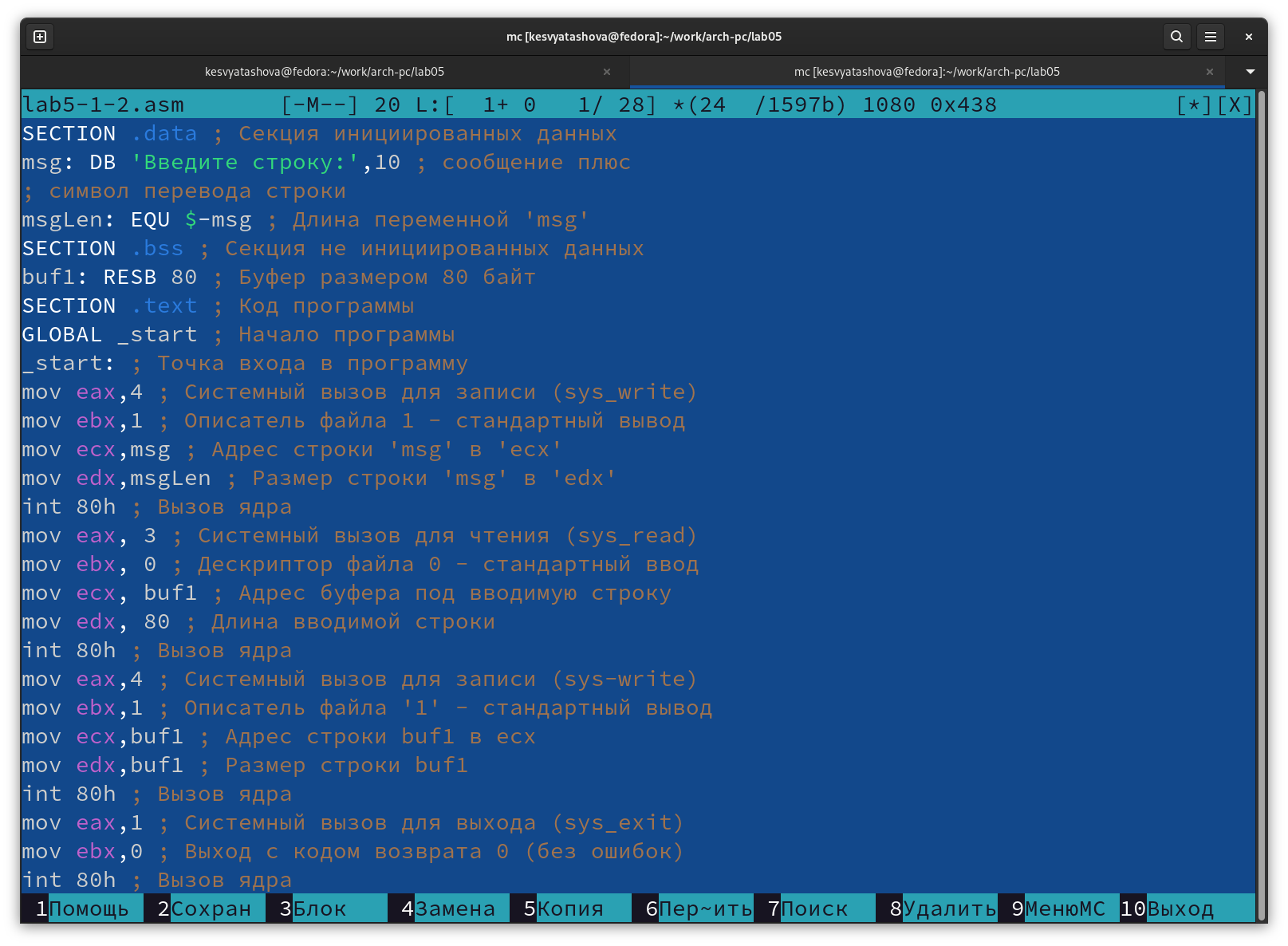


Рис. 19: Исправленная программа

1. Получим исполняемый файл и проверьте его работу. На приглашение ввести строку введу свою фамилию(рис. 20):

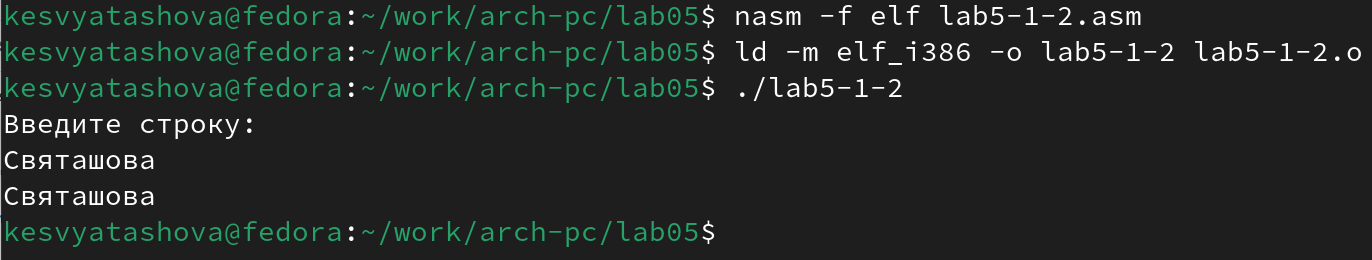


Рис. 20: Программа

1. Создим копию файла lab5-2.asm(рис. 21). Исправим текст программы(рис. 22) с использование подпрограмм из внешнего файла in\_out.asm, так чтобы она работала по следующему алгоритму: • вывести приглашение типа “Введите строку:”; • ввести строку с клавиатуры; • вывести введённую строку на экран.:

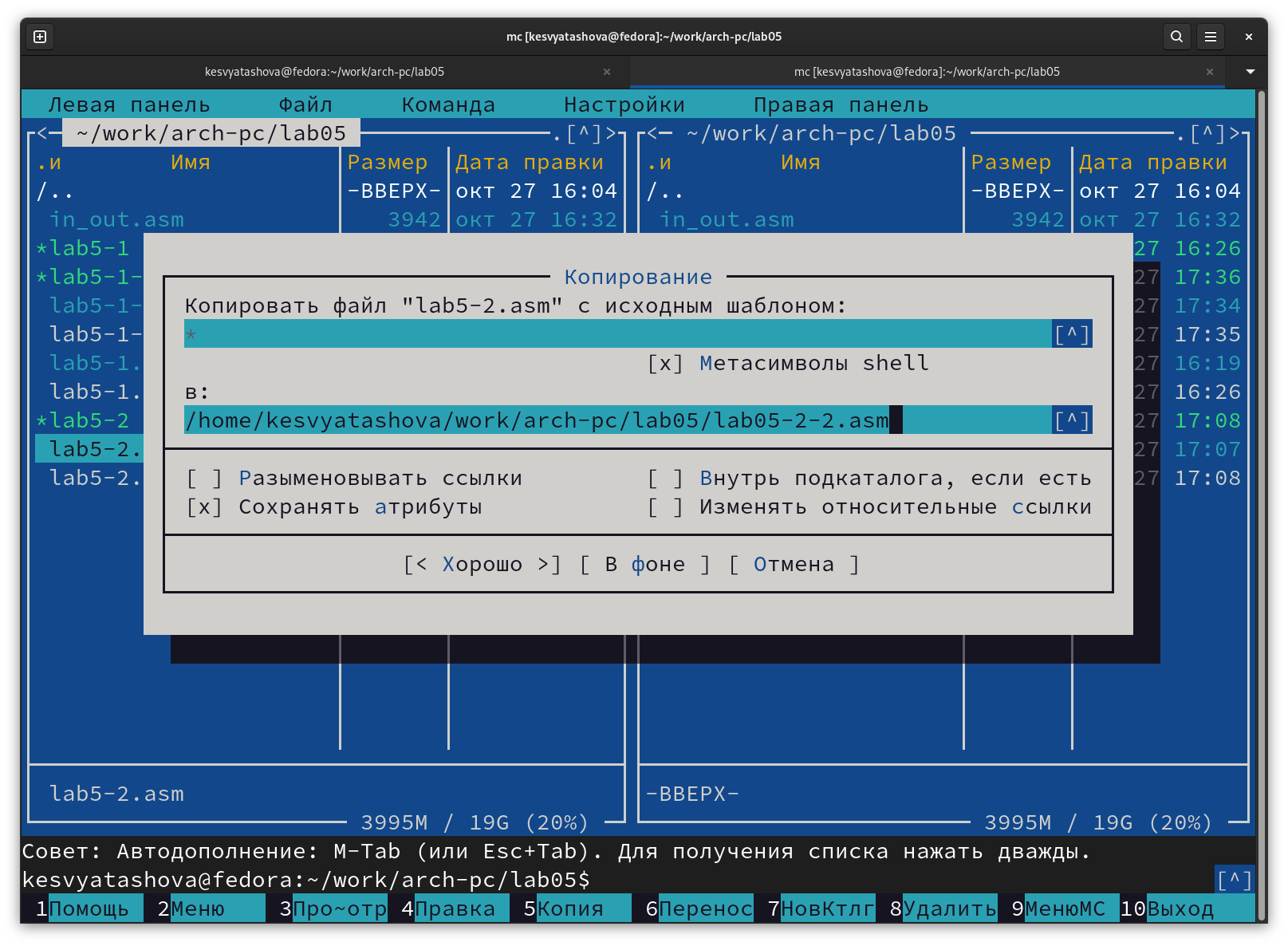


Рис. 21: Копирование файла

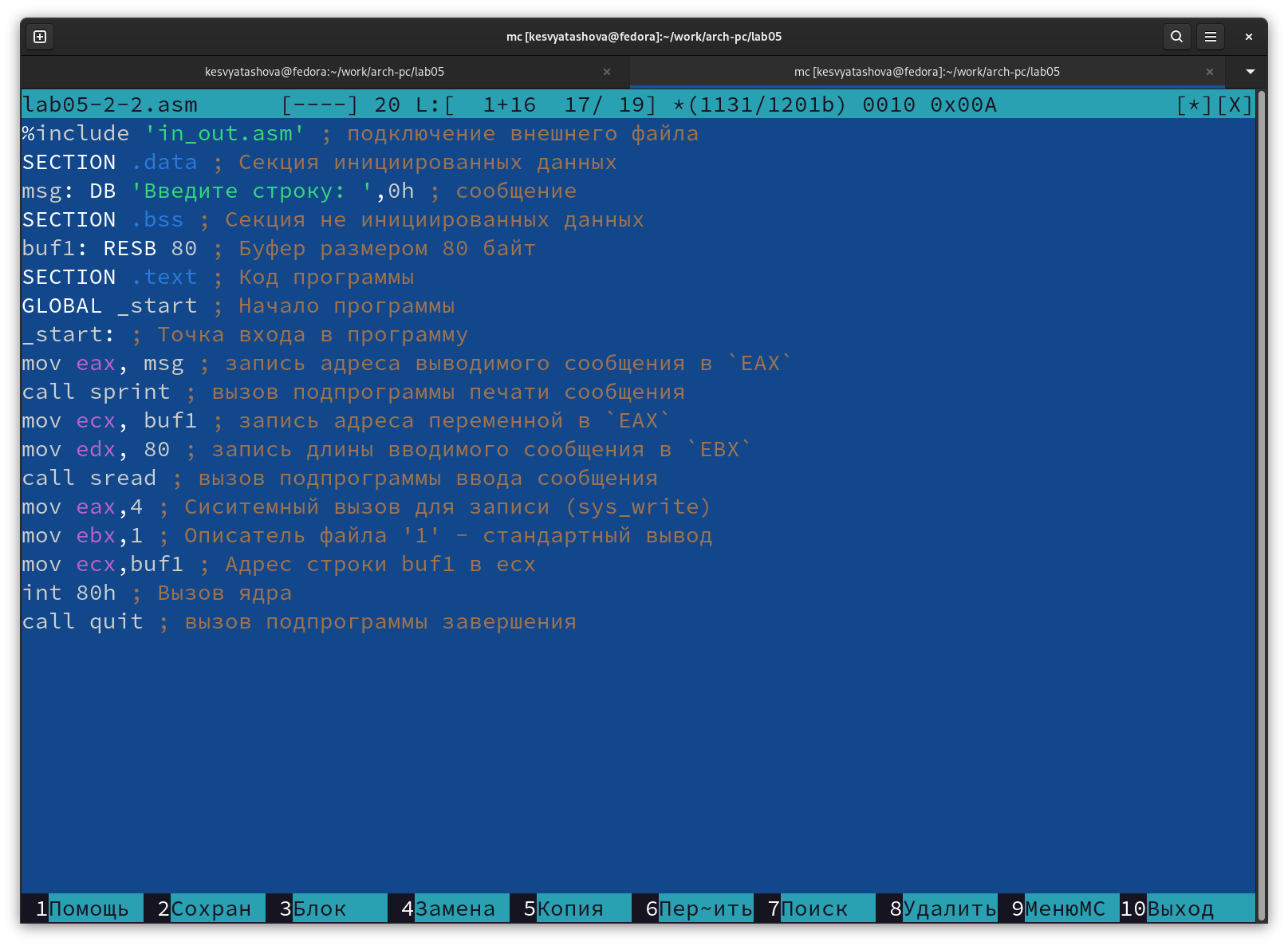


Рис. 22: Исправленная программа

1. Создайте исполняемый файл и проверьте его работу(рис. 23):

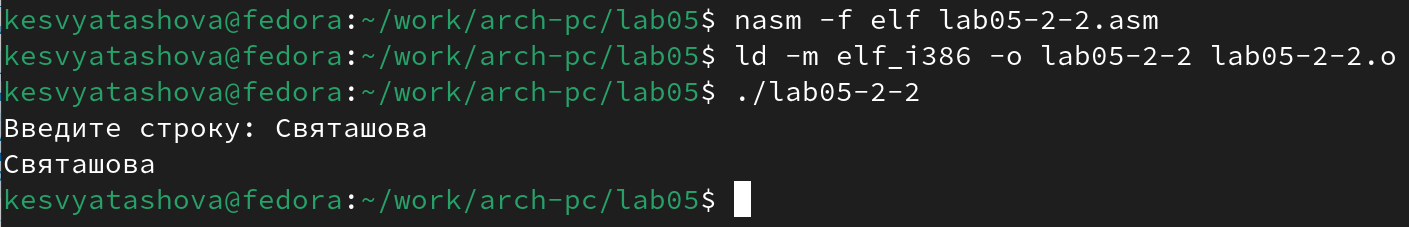


Рис. 23: Программа

# 5 Вывод

В результате выполнения работы я приобрела практические навыки работы в *Midnight Commander* и освоила язык ассемблера mov и int.