Отчёт по лабораторной работе №5

Дисциплина: Архитектура компьютера

Осина Виктория Александровна

Содержание

# 1 Цель работы

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

# 2 Задания

1. Создание каталога и файла, с которыми будет вестись дальнейшая работа
2. Освоение программы вывода сообщения на экран и ввода строки с клавиатуры
3. Подключение внешнего файла in\_out.asm
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. Для активации оболочки Midnight Commander достаточно ввести в командной строке mc и нажать клавишу Enter. В Midnight Commander используются функциональные клавиши F1 — F10, к которым привязаны часто выполняемые операции. 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. 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. Для объявления неинициированных данных в секции .bss используются директивы resb, resw, resd и другие, которые сообщают ассемблеру, что необходимо зарезервировать заданное количество ячеек памяти. Инструкция языка ассемблера mov предназначена для дублирования данных источника в приёмнике. В общем виде эта инструкция записывается в виде > mov dst,src

Здесь операнд dst — приёмник, а src — источник. В качестве операнда могут выступать регистры (register), ячейки памяти (memory) и непосредственные значения (const). Переслать значение из одной ячейки памяти в другую нельзя, для этого необходимо использовать две инструкции mov: >mov eax, x >mov y, eax

Также необходимо учитывать то, что размер операндов приемника и источника должны совпадать. Инструкция языка ассемблера int предназначена для вызова прерывания с указанным номером. В общем виде она записывается в виде > int n

Здесь n — номер прерывания, принадлежащий диапазону 0–255. После вызова инструкции int 80h выполняется системный вызов какой-либо функции ядра Linux. При этом происходит передача управления ядру операционной системы. Чтобы узнать, какую именно системную функцию нужно выполнить, ядро извлекает номер системного вызова из регистра eax. Поэтому перед вызовом прерывания необходимо поместить в этот регистр нужный номер. Кроме того, многим системным функциям требуется передавать какие-либо параметры. По принятым в ОС Linux правилам эти параметры помещаются в порядке следования в остальные регистры процессора: ebx, ecx, edx. Если системная функция должна вернуть значение, то она помещает его в регистр eax.

# 4 Выполнение лабораторной работы

## 4.1 Создание каталога и файла, с которыми будет вестись дальнейшая работа

Открываю Midnight Commander (рис. [1](#fig:001)) и (рис. [2](#fig:002)).

Figure 1: Открытие Midnight Commander

Figure 1: Открытие Midnight Commander

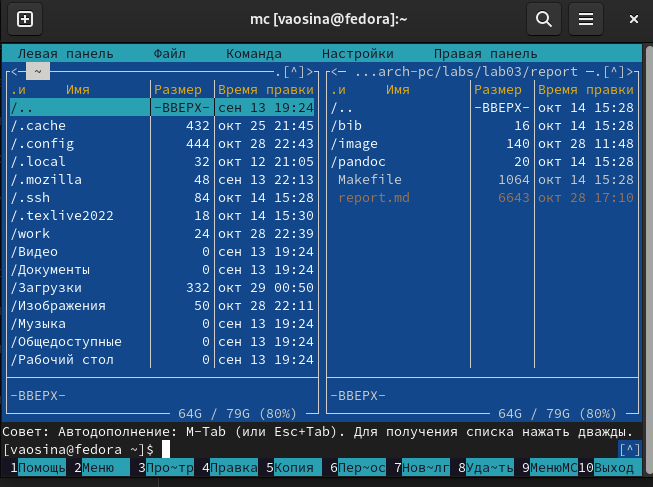


Figure 2: Открытый Midnight Commander

Пользуясь клавишами ↑, ↓ и Enter перехожу в каталог ~/work/arch-pc, созданный при выполнении лабораторной работы №4 (рис. [3](#fig:003)).

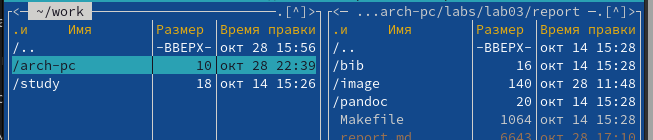


Figure 3: Переход в каталог

С помощью функциональной клавиши F7 создаю каталог lab05 и перехожу в него (рис. [4](#fig:004)).

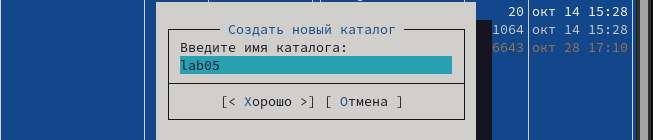


Figure 4: Создание каталога и переход в него

Пользуясь строкой ввода и командой touch создаю файл lab5-1.asm (рис. [5](#fig:005)).

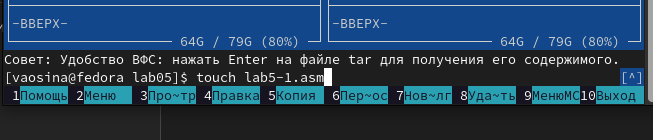


Figure 5: Создание файла lab5-1.asm

## 4.2 Освоение программы вывода сообщения на экран и ввода строки с клавиатуры

С помощью функциональной клавиши F4 открываю файл lab5-1.asm для редактирования во встроенном редакторе (рис. [6](#fig:006)).

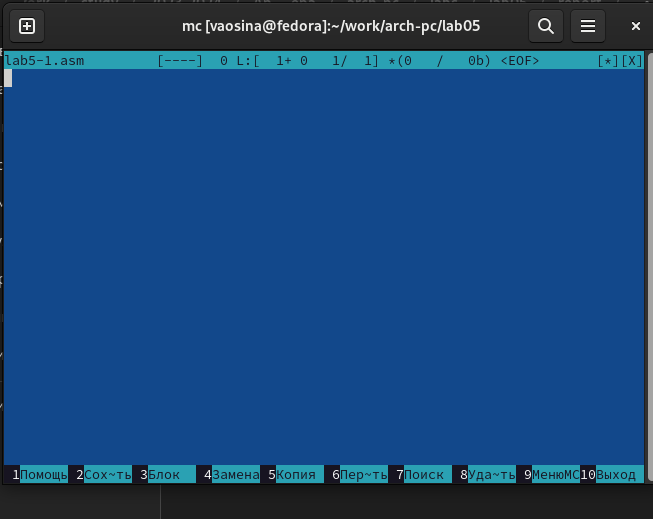


Figure 6: Открытие файла для редактирования

Ввожу текст программы, сохраняю изменения и закрываю файл. (рис. [7](#fig:007)) и (рис. [8](#fig:008)).

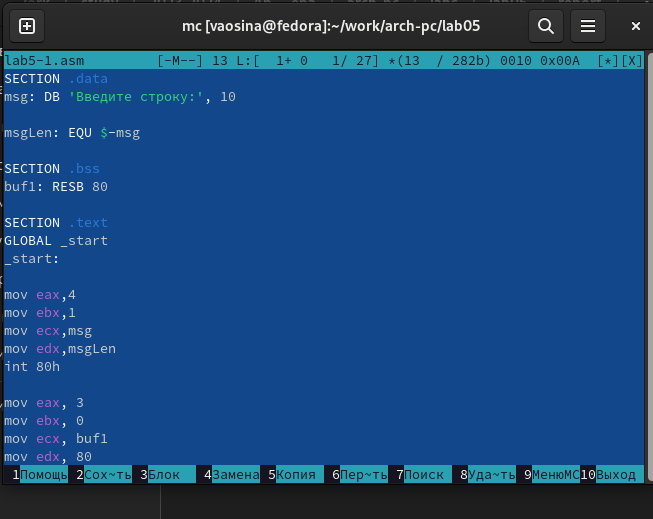


Figure 7: Ввод текста программы

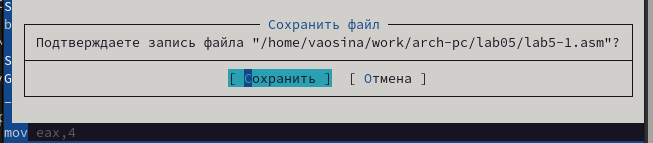


Figure 8: Сохранение изменений

С помощью функциональной клавиши F3 открываю файл lab5-1.asm для просмотра. Убеждаюсь, что файл содержит текст программы. (рис. [9](#fig:009)).

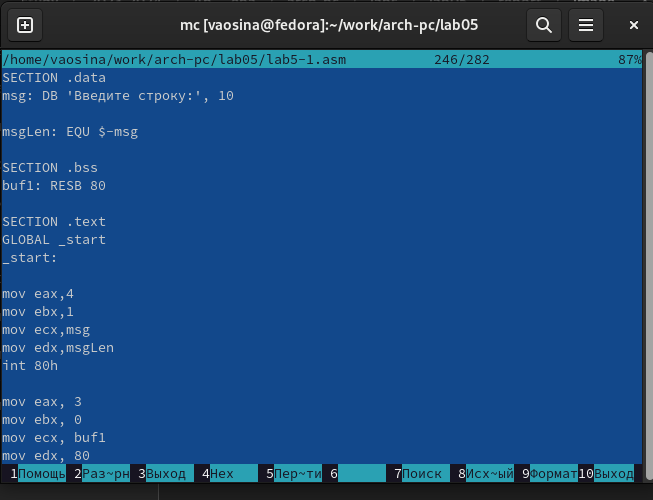


Figure 9: Открытие файла для просмотра

Транслирую текст программы lab5-1.asm в объектный файл. Выполняю компоновку объектного файла и запускаю получившийся исполняемый файл. Программа выводит строку ‘Введите строку:’ и ожидает ввода с клавиатуры. На запрос ввожу свои ФИО (рис. [10](#fig:010)) и (рис. [11](#fig:011)).

Figure 10: Трансляция текста

Figure 10: Трансляция текста

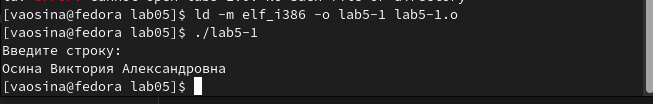


Figure 11: Компановка объектного файла и запуск исполняемого файла

## 4.3 Подключение внешнего файла in\_out.asm

Скачиваю файл in\_out.asm. В одной из панелей mc открываю каталог с файлом lab5-1.asm. В другой панели каталог со скаченным файлом in\_out.asm (для перемещения между панелями использую Tab ) (рис. [12](#fig:012)).

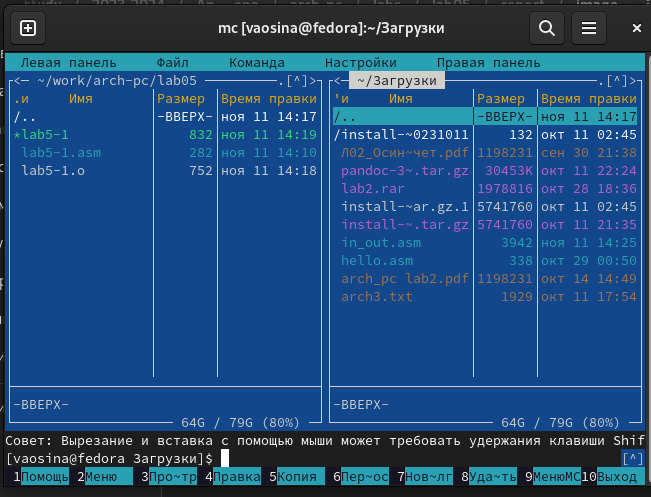


Figure 12: Открытие двух каталогов в панелях mc

Копирую файл in\_out.asm в каталог с файлом lab5-1.asm с помощью функциональной клавиши F5 (рис. [13](#fig:013)).

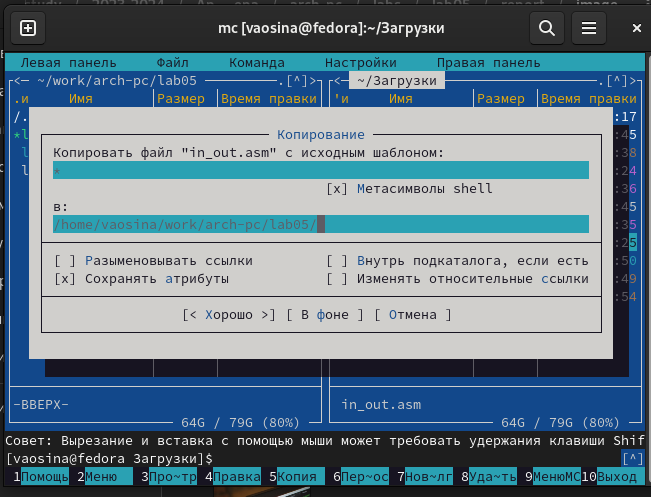


Figure 13: Копирование in\_out.asm

Убеждаюсь, что файл in\_out.asm теперь находится в нужном каталоге (рис. [14](#fig:014)).

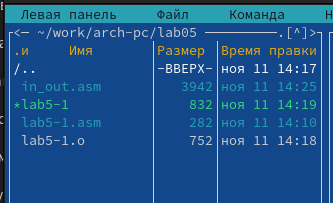


Figure 14: Файл теперь находится в нужном каталоге

С помощью функциональной клавиши F5 создаю копию файла lab5-1.asm с именем lab5-2.asm и убеждаюсь в том, что скопированный файл есть в каталоге (рис. [15](#fig:015)) (рис. [16](#fig:016)).

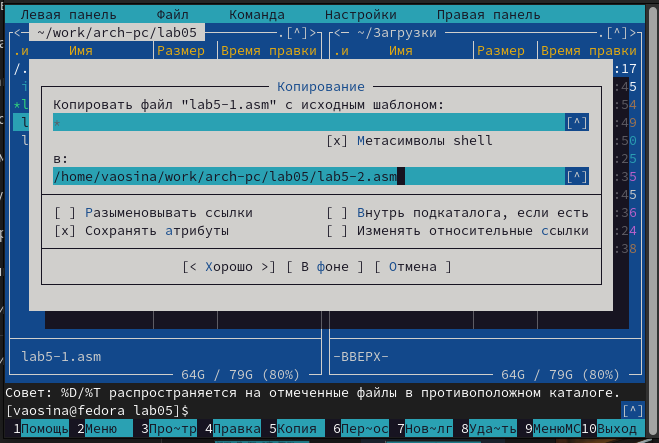


Figure 15: Создание копии файла

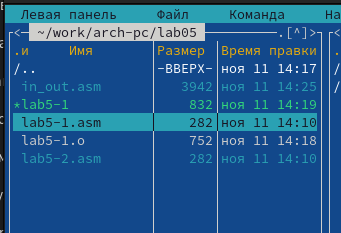


Figure 16: Файл находится в нужном каталоге

Исправляю текст программы в файле lab5-2.asm с использованием подпрограмм из внешнего файла in\_out.asm (использую подпрограммы sprintLF, sread и quit) (рис. [17](#fig:017)).

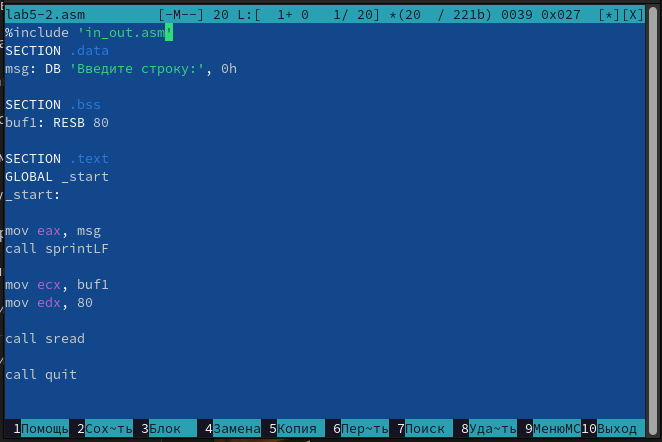


Figure 17: Исправление текста программы

Создаю исполняемый файл и проверяю его работу. (рис. [18](#fig:018)).

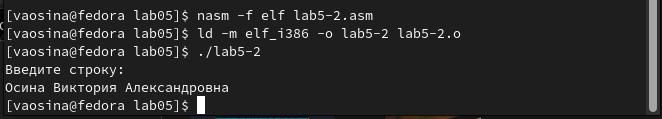


Figure 18: Создание исполняемого файла и проверка его работы

В файле lab5-2.asm заменяю подпрограмму sprintLF на sprint. Создаю исполняемый файл и проверяю его работу (рис. [19](#fig:019)) и (рис. [20](#fig:020)).

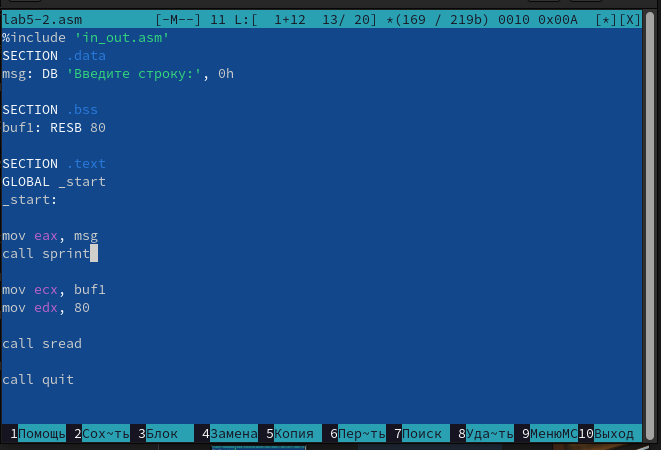


Figure 19: Замена программы sprintLF на sprint

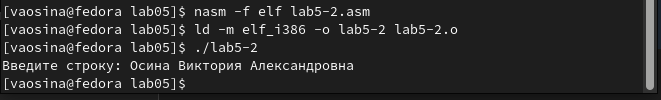


Figure 20: Создание исполняемого файла и проверка работы файла

Разница в том, что подкомманда sprintLF при выводе на экран добавляет к сообщению символ перевода строки, а sprint не добавляет, поэтому во втором случае ввод осуществляется на той же строке, где было выведено сообщение.

## 4.4 Выполнение заданий для самостоятельной работы

Создаю копию файла lab5-1.asm и называю её lab5-1-1.asm (рис. [21](#fig:021)).

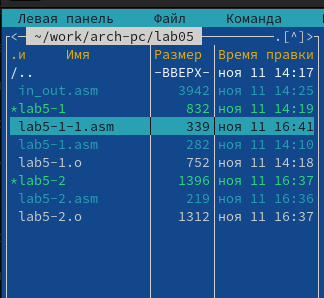


Figure 21: Создание копии файла

Вношу изменения в программу (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. [22](#fig:022)).

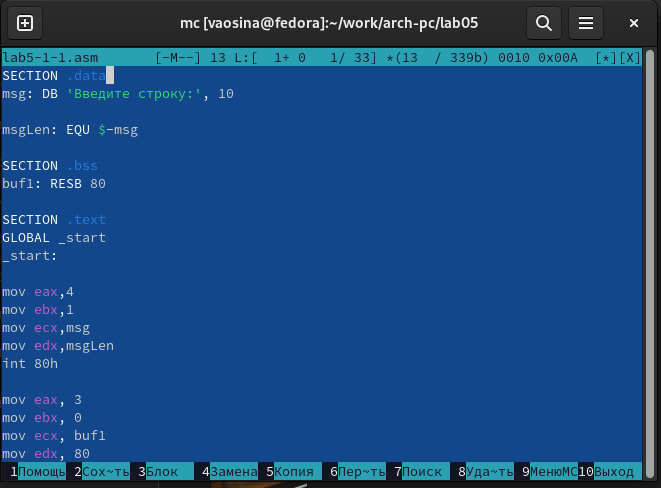


Figure 22: Изменение текста программы

Получаю исполняемый файл и проверяю его работу. На приглашение ввести строку ввожу свои ФИО (рис. [23](#fig:023)).

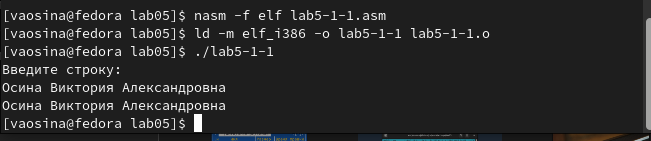


Figure 23: Получение исполняемого файла и проверка его работы

Создаю копию файла lab5-2.asm и называю её lab5-2-2.asm (рис. [24](#fig:024)) и (рис. [25](#fig:025)).

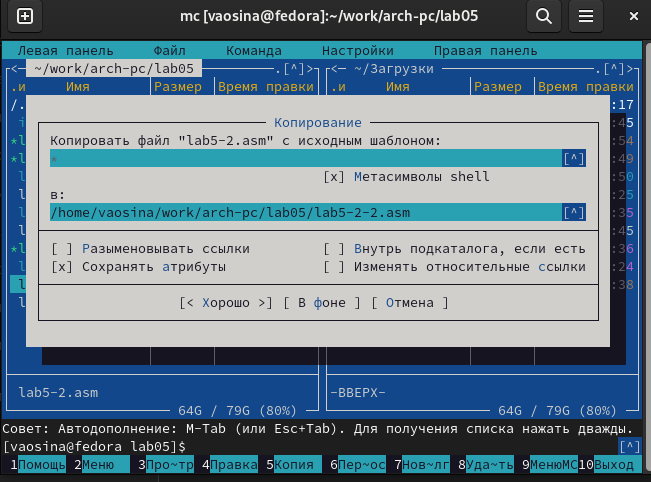


Figure 24: Создание копии файла

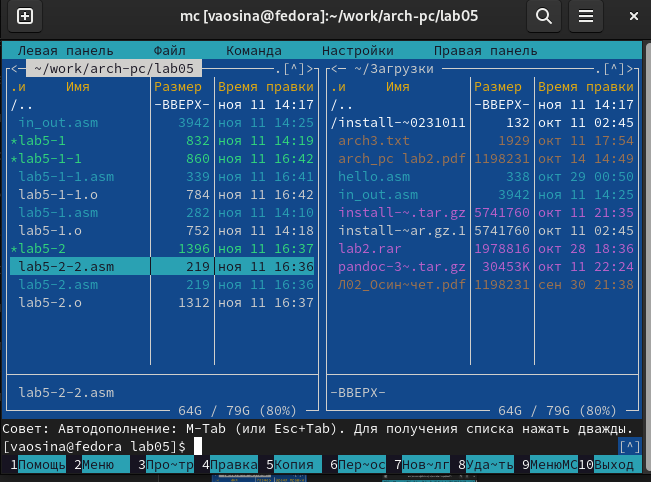


Figure 25: Файл находится в нужном каталоге

Исправляю текст программы 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. [26](#fig:028)).

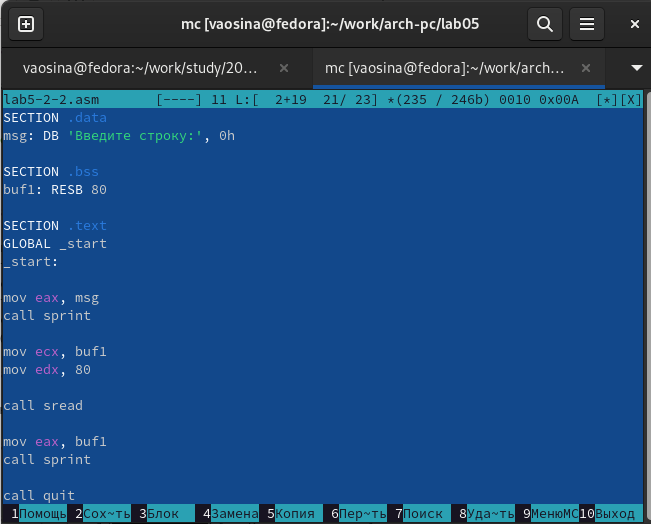


Figure 26: Изменение текста программы

Создаю исполняемый файл и проверяю его работу.(рис. [27](#fig:026)) и (рис. [28](#fig:027)).

Figure 27: Создание исполняемого файла

Figure 27: Создание исполняемого файла

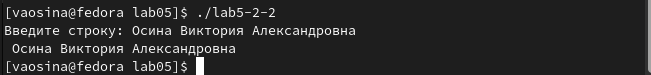


Figure 28: Проверка работы исполняемого файла

# 5 Выводы

Я приобрела практические навыки работы в Midnight Commander и освоила инструкции языка ассемблера mov и int.

# 6 Список литературы

1. [Архитектура ЭВМ](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)