Отчёт по лабораторной работе 5

Архитектура компьютера

Нилова Кристина Артуровна

Содержание

# 1 Цель работы

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

# 2 Задания

1. Изучение Midnight Commander
2. Примеры программ с использованием внешнего файла in\_out.asm
3. 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).

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

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

# 4 Выполнение лабораторной работы

## 4.1 Основы работы в Midnight Commander

Я открыла Midnight Commander и перешла в каталог ~/work/arch-pc. Затем я создала новый каталог под названием lab05. рис. [[1](#fig:001)]

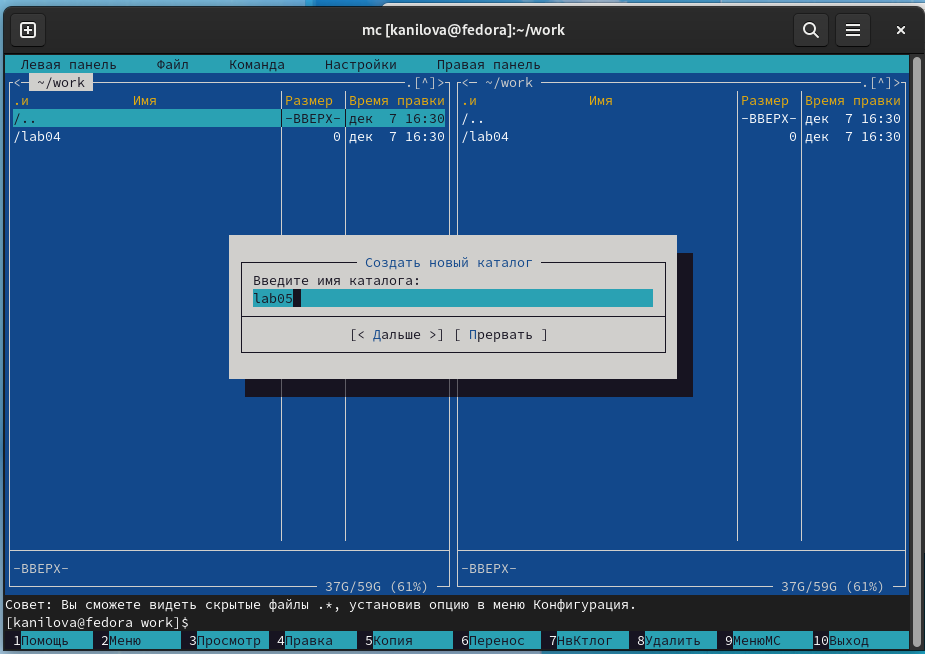


Figure 1: Создание каталога

Внутри каталога lab05 я создала файл с именем lab05-1.asm. рис. [[2](#fig:002)]

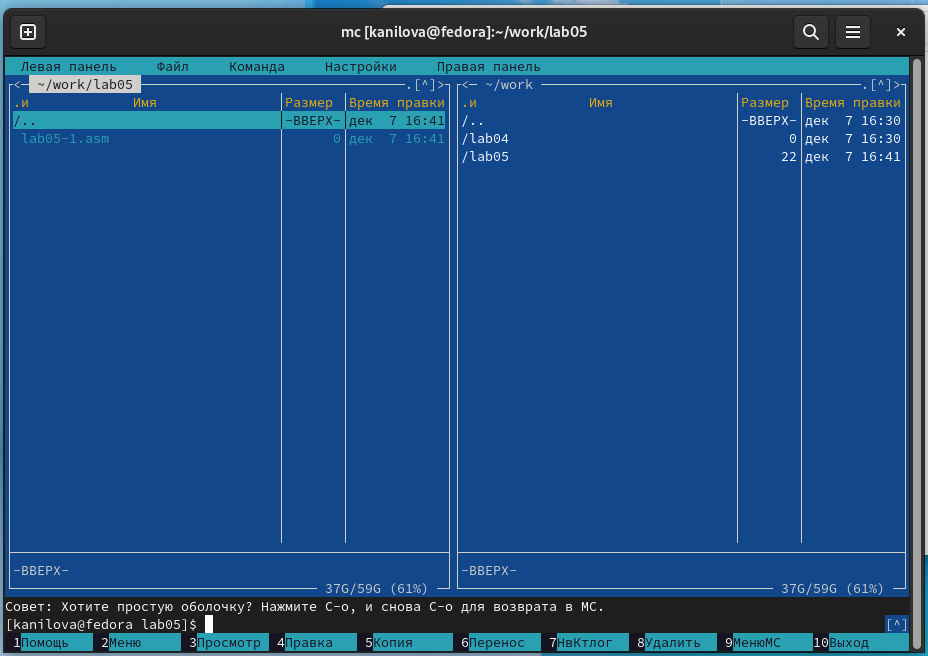


Figure 2: Создание файла lab05-1.asm

Я открыла файл lab05-1.asm для редактирования и написала в нем код программы. рис. [[3](#fig:003)]

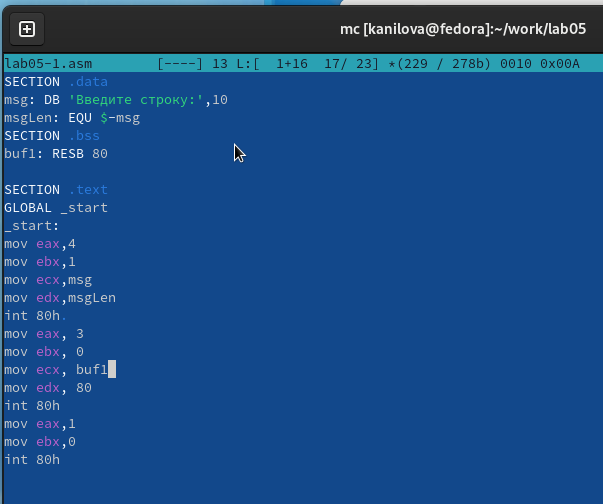


Figure 3: Заполнение файла lab05-1.asm

После этого я открыла файл для просмотра и убедилась, что он содержит написанный мной код. рис. [[4](#fig:004)]

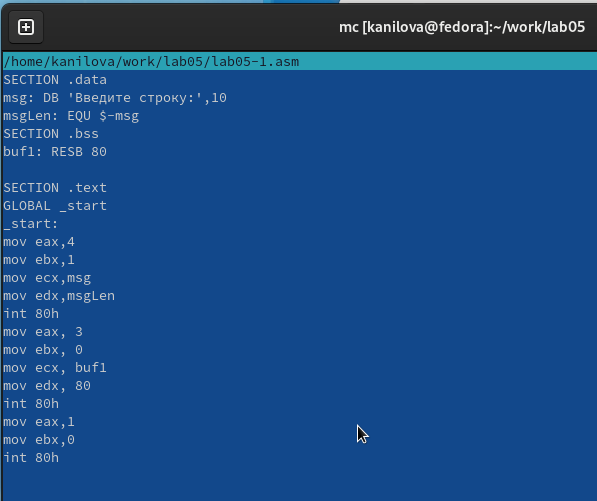


Figure 4: Просмотр файла lab05-1.asm

Затем я скомпилировала программу и проверила ее работу, получив исполняемый файл. рис. [[5](#fig:005)]

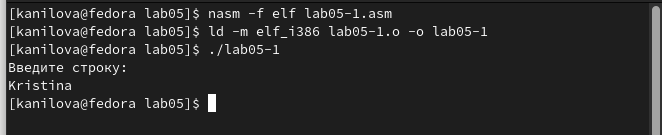


Figure 5: Компиляция текста программы lab05-1.asm

## 4.2 Подключение внешнего файла in\_out.asm

Далее, я скачала файл с именем in\_out.asm и добавила его в рабочий каталог. Затем я скопировала файл lab05-1.asm и создала копию с именем lab05-2.asm. рис. [[6](#fig:006)]

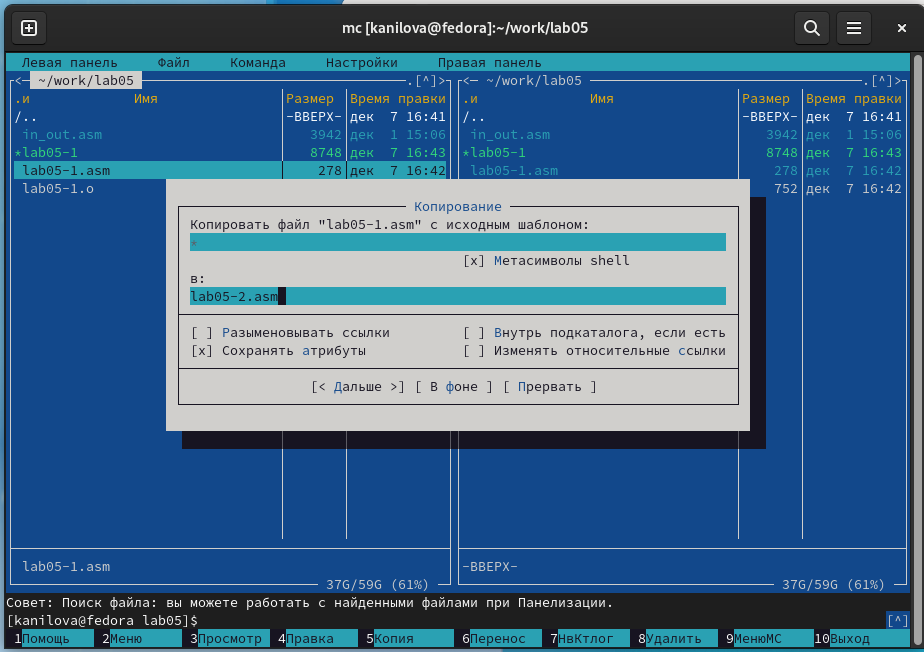


Figure 6: Копирование файла

Я написала код программы в файле lab05-2.asm. рис. [[7](#fig:007)]

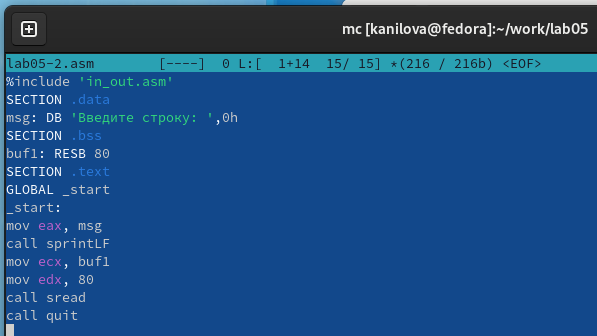


Figure 7: Заполнение файла lab05-2.asm

После этого я скомпилировала программу и проверила ее запуск. рис. [[8](#fig:008)]

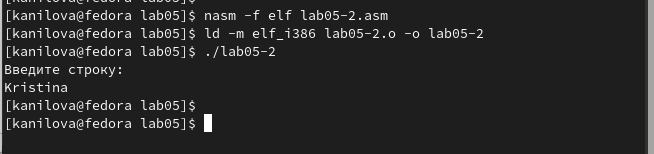


Figure 8: Компиляция текста программы lab05-2.asm

В файле lab05-2.asm я внесла изменения, заменив подпрограмму sprintLF на sprint. Это позволило строке вывода не завершаться символом перехода на новую строку. рис. [[9](#fig:009)].

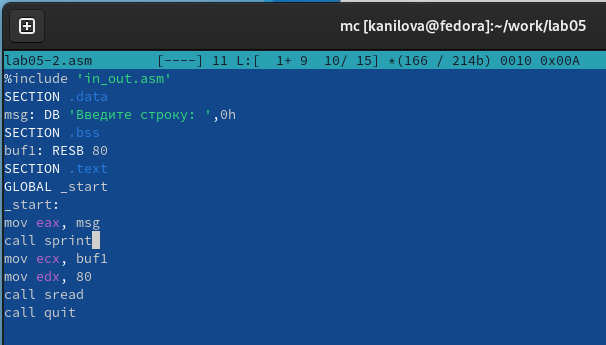


Figure 9: Заполнение файла lab05-2.asm

Затем я снова собрала исполняемый файл. Теперь после вывода строки она не будет завершаться символом перехода на новую строку. рис. [[10](#fig:010)].

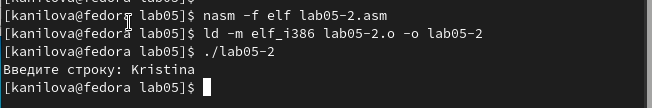


Figure 10: Компиляция текста программы lab05-2.asm

## 4.3 Задание для самостоятельной работы

Я решила скопировать программу lab05-1.asm и внести изменения в код, чтобы программа выводила приглашение вроде “Введите строку:”, затем считывала строку с клавиатуры и выводила ее на экран. рис. [-[11](#fig:011)][[12](#fig:012)]

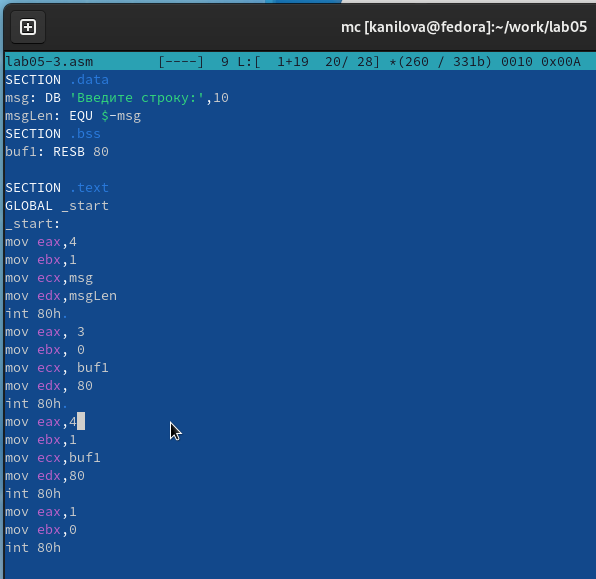


Figure 11: Заполнение файла lab05-3.asm

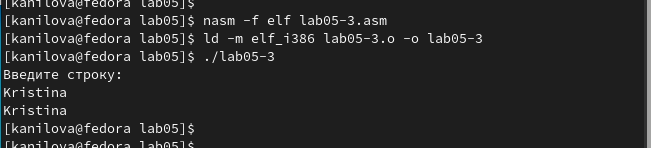


Figure 12: Компиляция текста программы lab05-3.asm

Также я скопировала программу lab05-2.asm и внесла соответствующие изменения в код, чтобы программа тоже выводила приглашение вроде “Введите строку:”, считывала строку с клавиатуры и выводила ее на экран. рис. [-[13](#fig:013)][[14](#fig:014)]

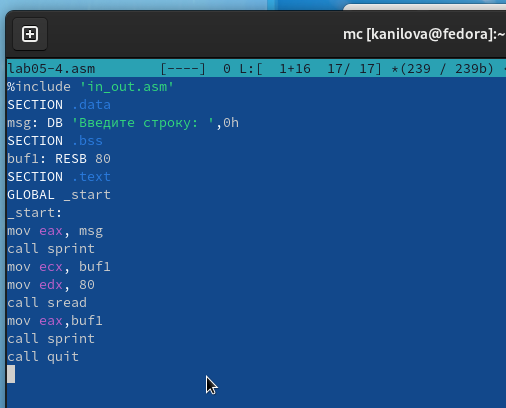


Figure 13: Заполнение файла lab05-4.asm

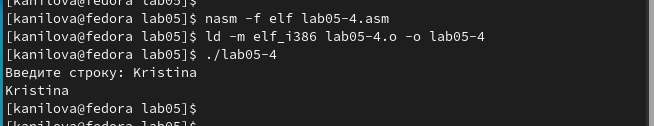


Figure 14: Компиляция текста программы lab05-4.asm

Отличие между этими двумя реализациями заключается в том, что файл in\_out.asm уже содержит готовые подпрограммы для обеспечения ввода/вывода. Таким образом, нам остается только разместить данные в нужных регистрах и вызвать нужную подпрограмму с помощью инструкции call. Это упрощает кодирование и обеспечивает более гибкую работу с вводом и выводом данных.

# 5 Выводы

Научились писать базовые ассемблерные программы. Освоили ассемблерные инструкции mov и int.