Отчёт по лабораторной работе №6

Архитектура компьютера

Рогожина Надежда Александровна

Содержание

# 1 Цель работы

Приобретение практических навыков работы в Midmight Commander. Освоение инструкций языка ассемблера mov и int.

# 2 Задание

1. Создайте копию файла lab6-1.asm. Внесите 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;

1. Получите исполняемый файл и проверьте его работу. На приглашение ввести строку введите свою фамилию.
2. Создайте копию файла lab6-2.asm. Исправьте 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.

1. Создайте исполняемый файл и проверьте его работу.

# 3 Теоретическое введение

## 3.1 Основы работы с Midnight Commander

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, позволяет сделать работу с файлами более удобной и наглядной.

Таблица 1: Функциональные клавиши Midnight Commander

| Функциональные клавиши | Выполняемое действие |
| --- | --- |
| F1 | Вызов контекстно-зависимой подсказки |
| F2 | Вызов меню, созданного пользователем |
| F3 | Просмотр файла, на который указывает подсветка в активной панели |
| F4 | Вызов встроенного редактора для файла, на который указывает подсветка в активной панели |
| F5 | Копирование файла или группы отмеченных файлов из каталога, отображаемого в активной панели, в каталог, отображаемый на второй панели |
| F6 | Перенос файла или группы отмеченных файлов из каталога, отображаемого в активной панели, в каталог отображаемый на второй панели |
| F7 | Создание подкаталога в каталоге, отображаемом в активной панели |
| F8 | Удаление файла (подкаталога) или группы отмеченных файлов |
| F9 | Вызов основного меню программы |
| F10 | Выход из программы |

## 3.2 Структура программы на языке ассемблера NASM

Программа на языке ассемблера NASM, как правило, состоит из трёх секций:

1. секция кода программы (SECTION .text),
2. секция инициированных (известных во время компиляции) данных (SECTION .data)
3. секция неинициализированных данных (тех, под которые во время компиляции только отводится память, а значение присваивается в ходе выполнения программы) (SECTION .bss)

## 3.3 Описание инструкции mov

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).

## 3.4 Описание инструкции int

Инструкция языка ассемблера intпредназначена для вызова прерывания с указанным номером. В общем виде она записывается в виде

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).

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

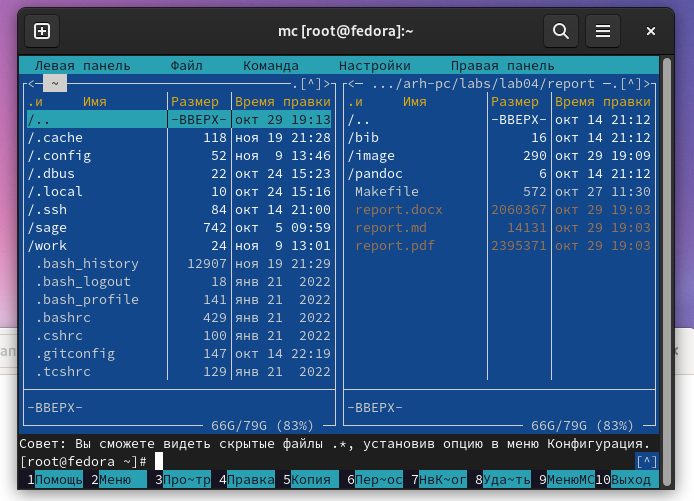


Рис. 1: Прежде всего откроем Midnight Commander

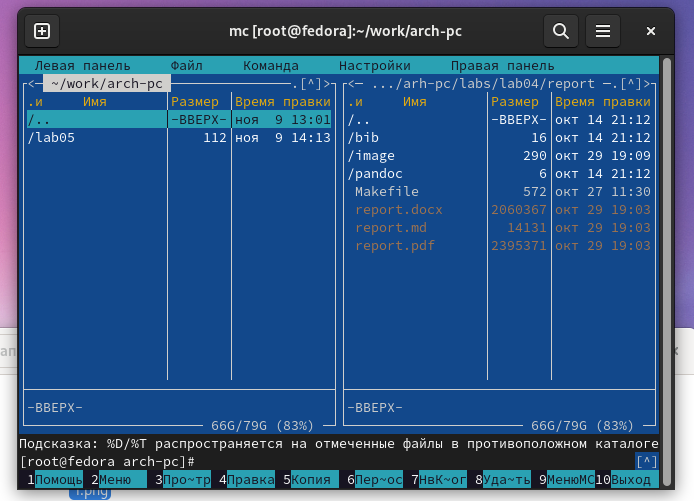


Рис. 2: Перейдем в каталог, созданный при выполнении лабораторной работы №5

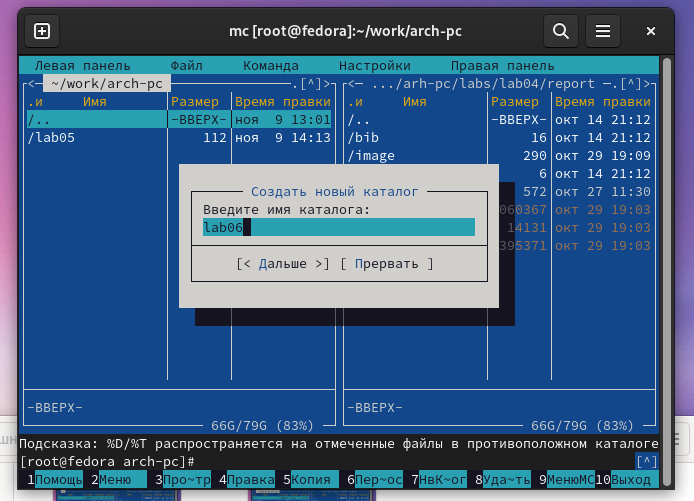


Рис. 3: Создадим новый каталог lab06

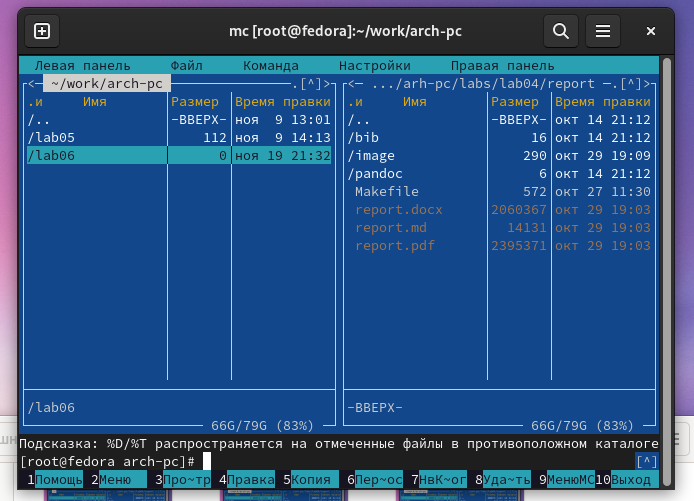


Рис. 4: Проверим создание каталога

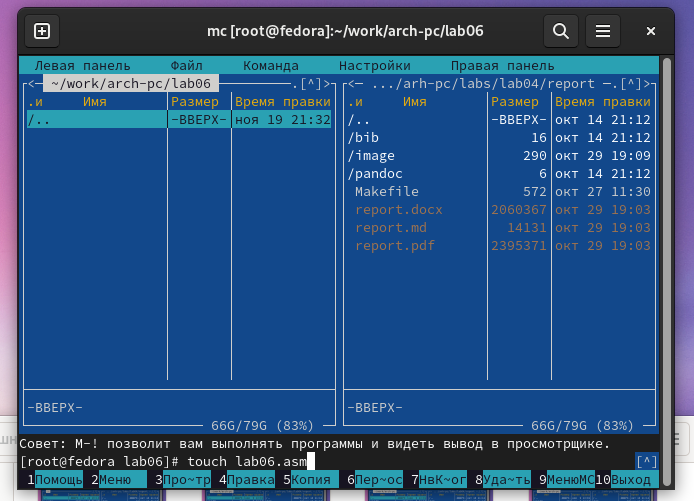


Рис. 5: Перейдем в этот каталог и создадим файл lab06.asm

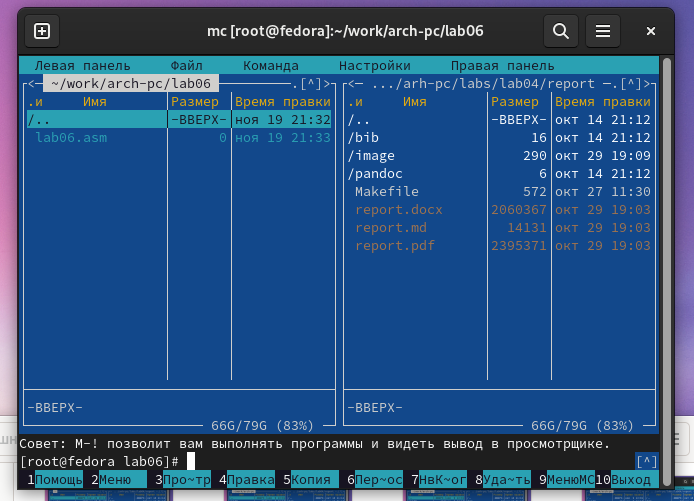


Рис. 6: Проверим создание файла

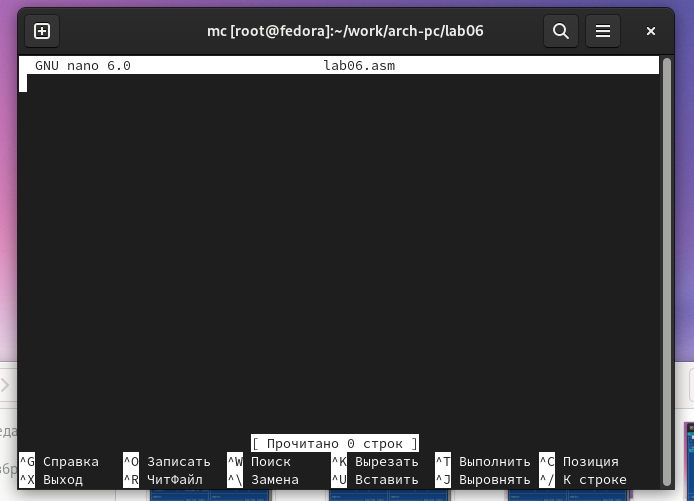


Рис. 7: С помощью функциональной клавиши F4 откроем файл lab06-1.asm в редакторе nano

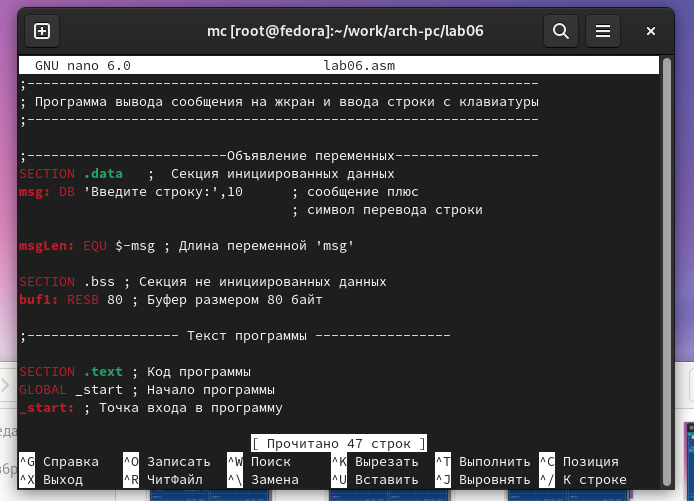


Рис. 8: Введем текст программы из листинга 6.1

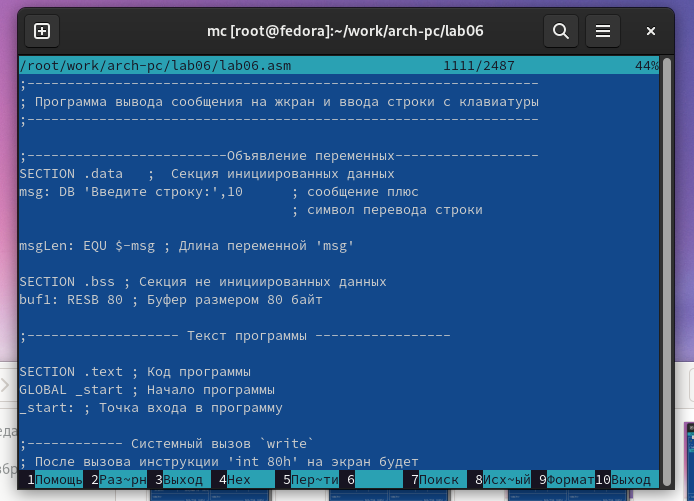


Рис. 9: С помощью функиональной клавиши F3 проверим текст программы

Оттранслируем текст программы в объектный файл, а затем выполним компоновку и запустим получившийся исполняемый файл

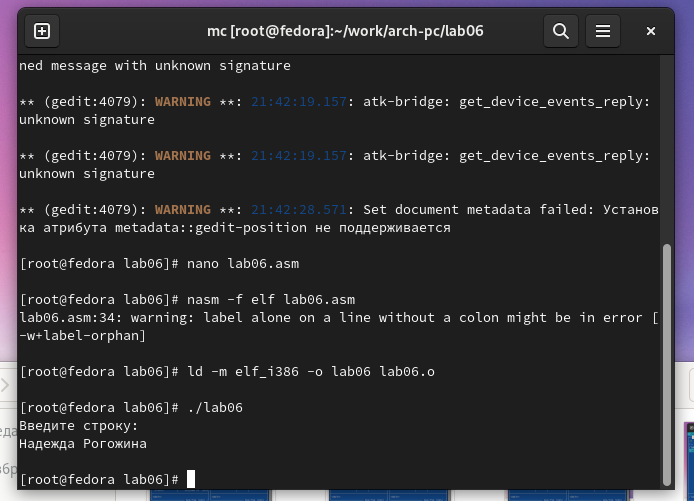


Рис. 10: Трансляция и компоновка

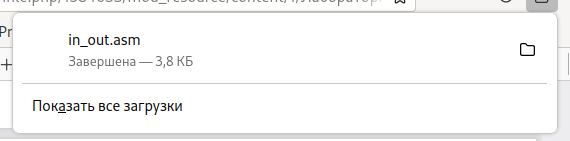


Рис. 11: Скачаем файл

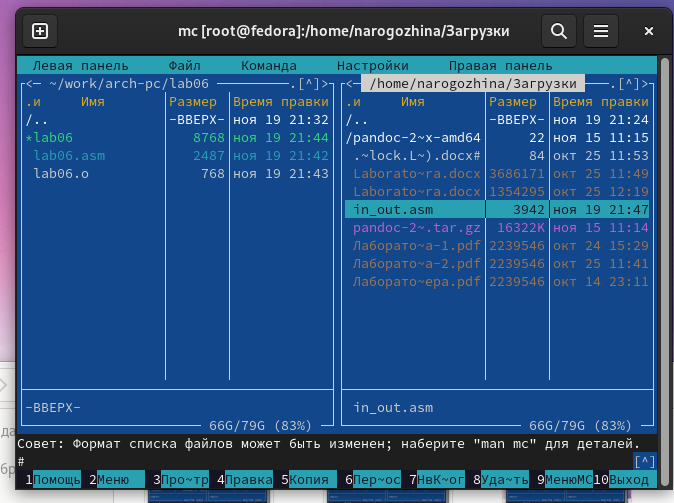


Рис. 12: Зайдем в папку, куда файл был скачан через mc

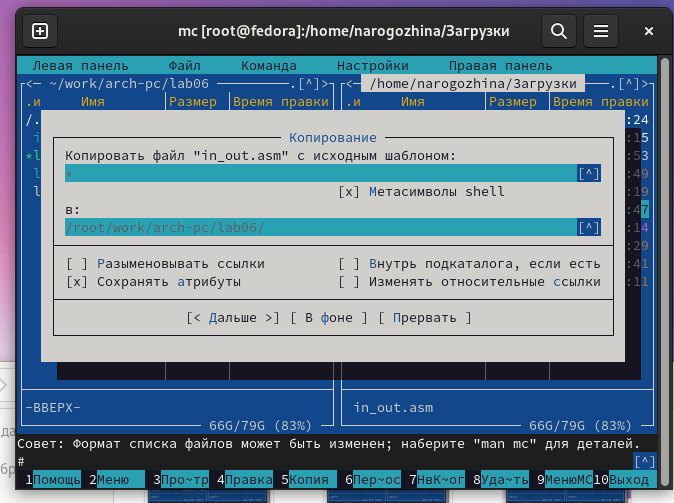


Рис. 13: Скопируем файл в каталог с файлов lab06-1.asm с помощью F5

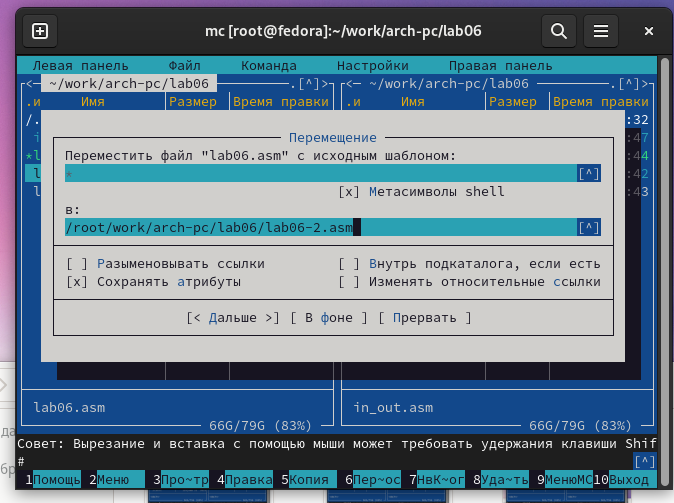


Рис. 14: С помощью F6 создадим копию файла lab6-01.asm c именем lab06-2.asm

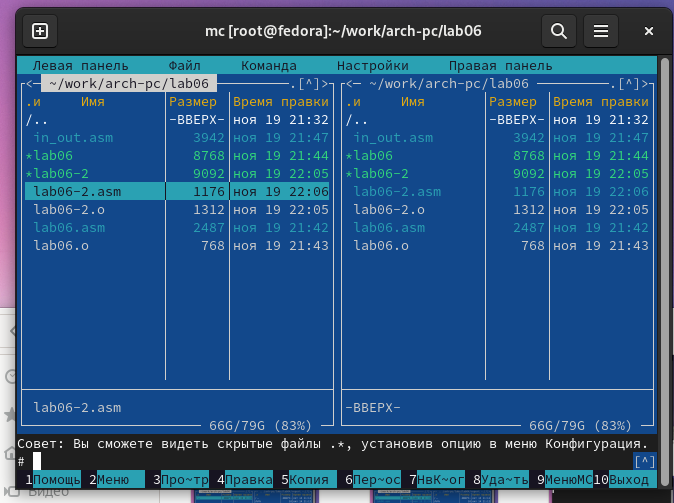


Рис. 15: Проверим создание файла

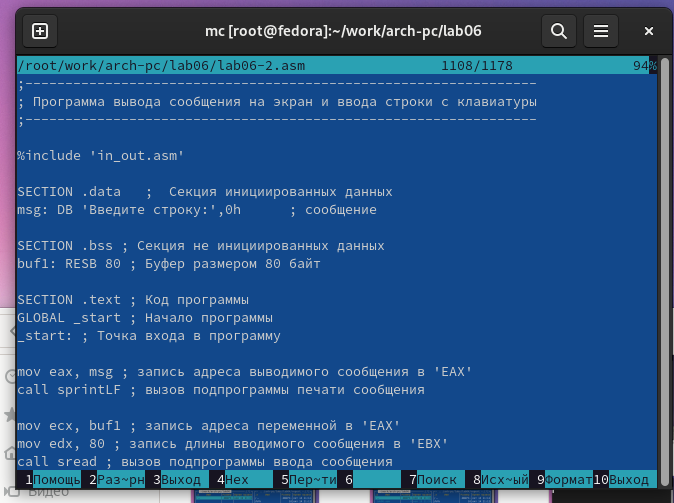


Рис. 16: Исправим текст программы в файле lab06-2.asm

Оттранслируем текст программы в объектный файл, а затем выполним компоновку и запустим получившийся исполняемый файл

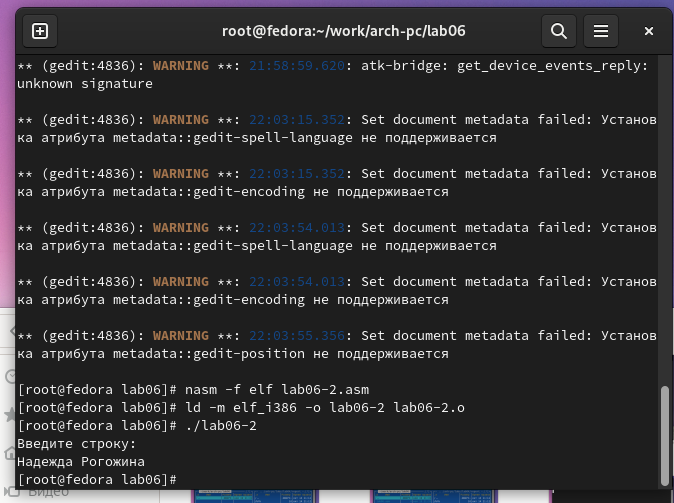


Рис. 17: Трансляция и компоновка

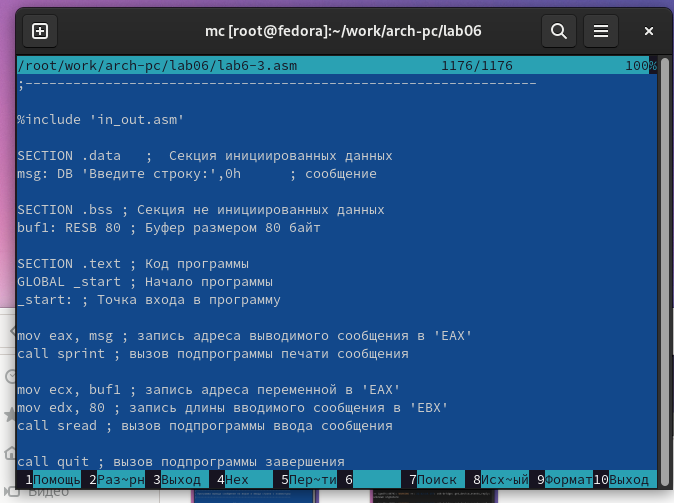


Рис. 18: Внесем изменения в текст программы

Оттранслируем текст программы в объектный файл, а затем выполним компоновку и запустим получившийся исполняемый файл

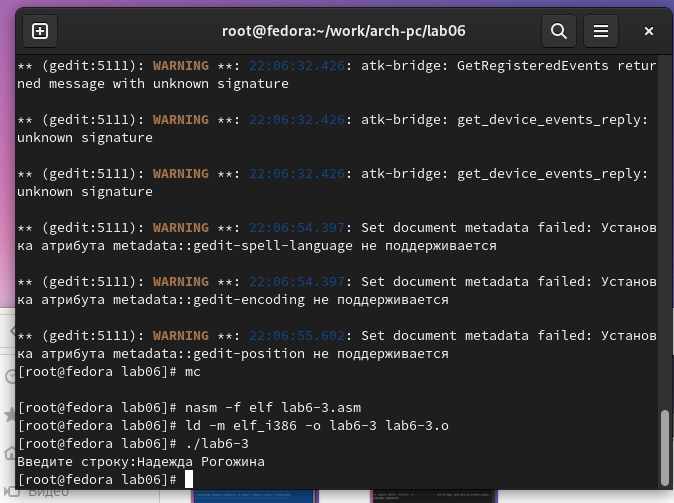


Рис. 19: Трансляция и компоновка

После изменения текста программы, при выполнении программы больше не происходит переноса строки.

# 5 Выполнение заданий для самостоятельной работы

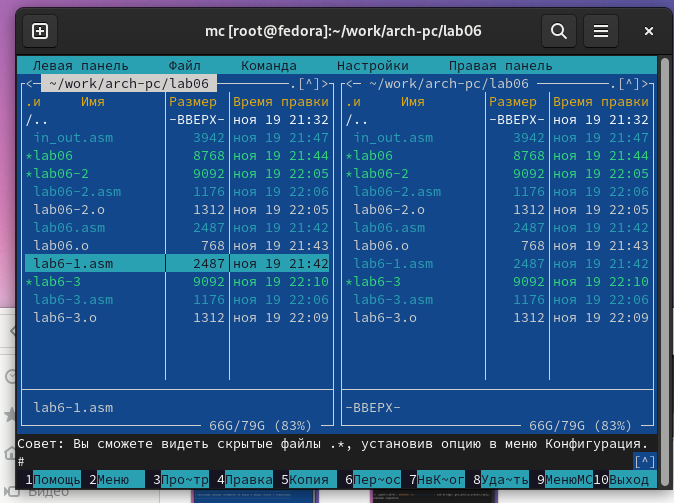


Рис. 20: Создадим копию исходного файла

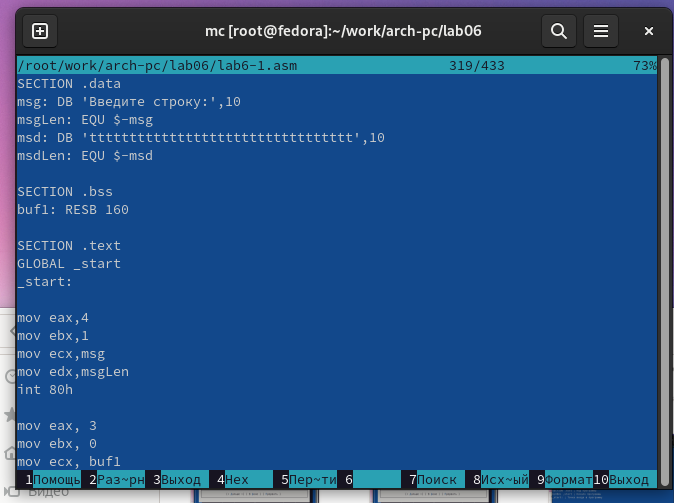


Рис. 21: Проверим изменение текста программы

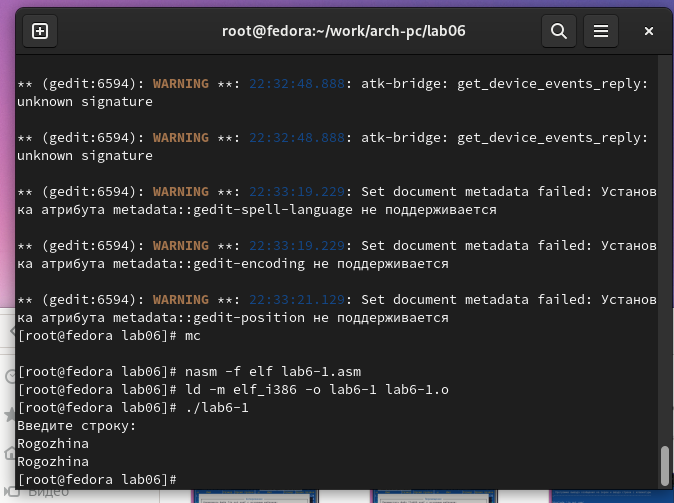


Рис. 22: Трансляция и компоновка

Таким образом, после запуска программы, мы удостоверились, что все работает как нужно

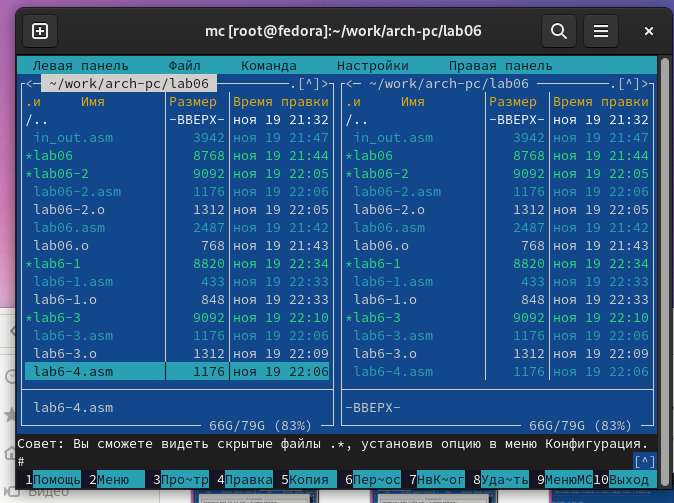


Рис. 23: Создадим копию файла lab06-2.asm

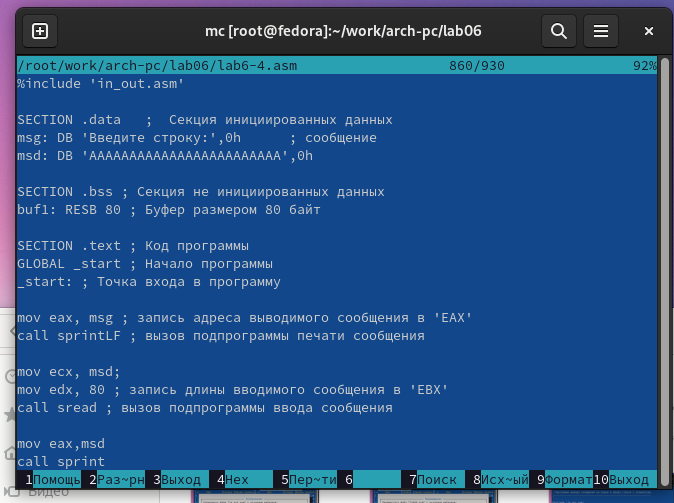


Рис. 24: Проверим изменение текста программы

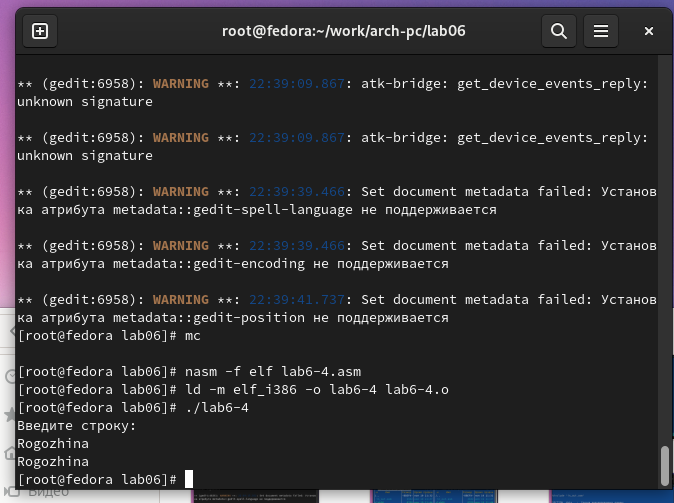


Рис. 25: Трансляция и компоновка

После запуска программы мы убедились, что все работает

# 6 Выводы

Таким образом, по выполнению лабораторной работы мы научились работать с Midnight Commander.

1. MC - это программа, которая позволяет просматривать структуру каталогов и выполнять основные операции по управлению файловой системы. Она упрощает работу с файлами, делает ее более удобной и наглядной.
2. Такие операции как (например) копирование, перемещение, удаление файлов можно выполнить и с помощью bash, и с помощью mc.
3. Типичный формат записи команд NASM имеет вид:

[метка:] мнемокод [операнд {, операнд}] [; комментарий]

1. Секция bss содержит переменные, для которых **не задано** начальное значение. Секция data содержит переменные, для которых **задано** начальное значение.
2. Для объявления инициированных данных в секции .data используются следующие директивы:

* DB - определяет переменную размером в 1 байт
* DW - определяет переменную размером в 2 байта (слово)
* DD - определяет переменную размером в 4 байта (двойное слово)
* DQ - определяет переменную размером в 8 байт (учетверённое слово)
* DT - определяет переменную размером в 10 байт

1. Инструкция **mov eax, esi** пересылает значение регистра esi в регистр eax.
2. int 80 - вызов ядра.