Отчёт по лабораторной работе №5

Дисциплина: Архитектура Компьютера

Ислам Карданов Вячеславович

Содержание

Список иллюстраций

1 Цель работы

Целью данной работы является приобретение практического опыта работы с Midnight Commander, освоение инструкций языка ассемблера mov и int.

2 Задание

1. Основы работы с Midnight Commander
2. Подключение внешнего файла
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). 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: 1) DB (define byte) — определяет переменную размером в 1 байт; 2) DW (define word) — определяет переменную размеров в 2 байта (слово); 3) DD (define double word) — определяет переменную размером в 4 байта (двойное слово); 4) DQ (define quad word)— определяет переменную размером в 8 байт (учетверённое слово); 5) 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. 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. 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. Также необходимо учитывать то, что размер операндов приемника и источника должны совпадать. Инструкция языка ассемблера int предназначена для вызова прерывания с указанным номером. 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 тот 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) Основы Midnight Commander

Открываю Midnight Commander, введя в терминале команду mc.

Перехожу в каталог ~/work/arch-pc, используя файловый менеджер mc. (рис. 4.1).

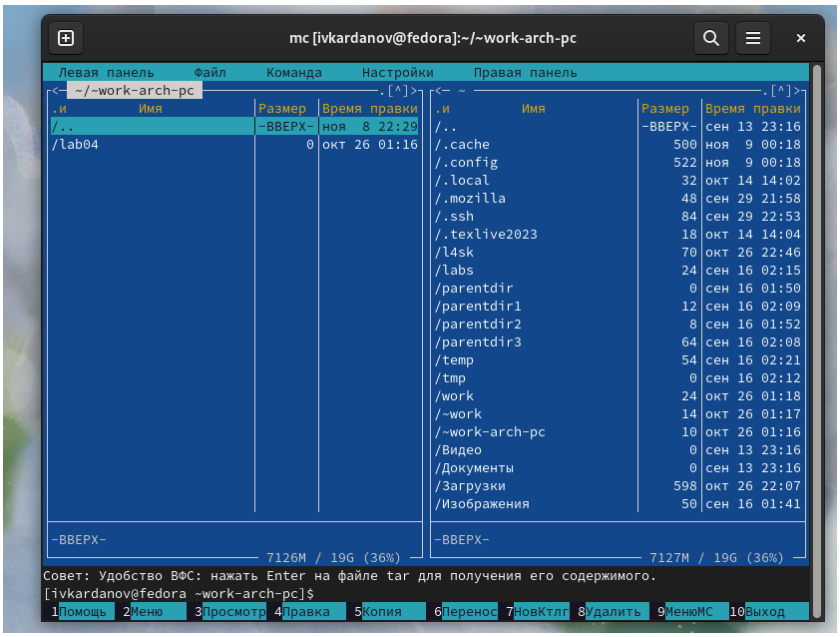

Рис. 4.1: Перемещение в Midnight Commander

С помощью функциональной клавиши F7 создаю каталог lab05. (рис. 4.2).

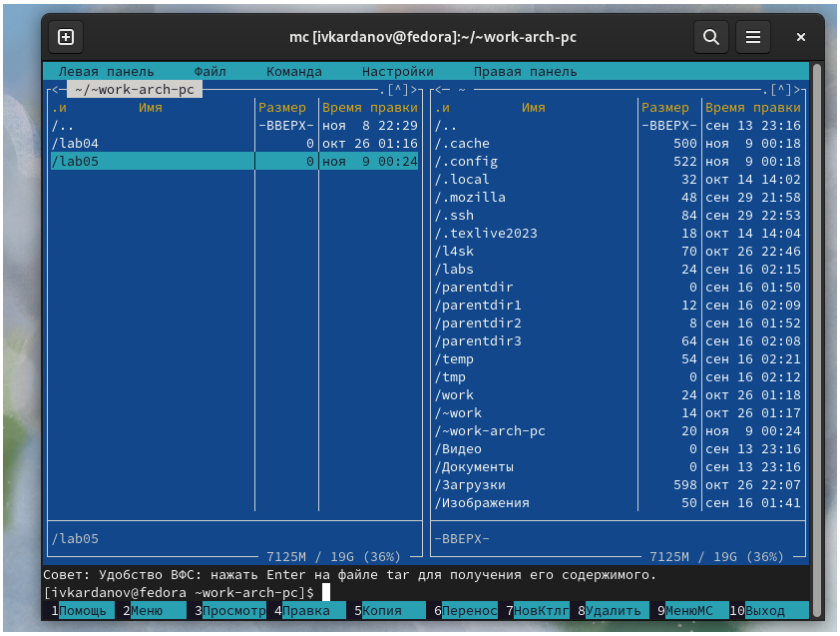

Рис. 4.2: Создание каталога

В строке ввода прописываю команду touch lab5-1.asm, чтобы создать соответствующий файл. (рис. 4.3).

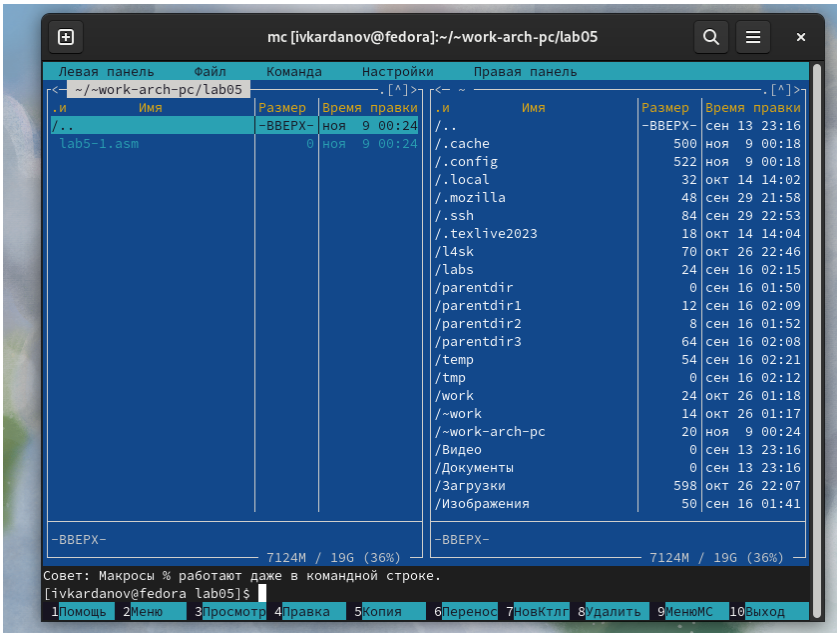

Рис. 4.3: Создание файла

С помощью функциональной клавиши F4 открываю созданный файл для его изменения. Ввожу в файл нужный код программы. Затем сохраняю и выхожу . (рис. 4.4).

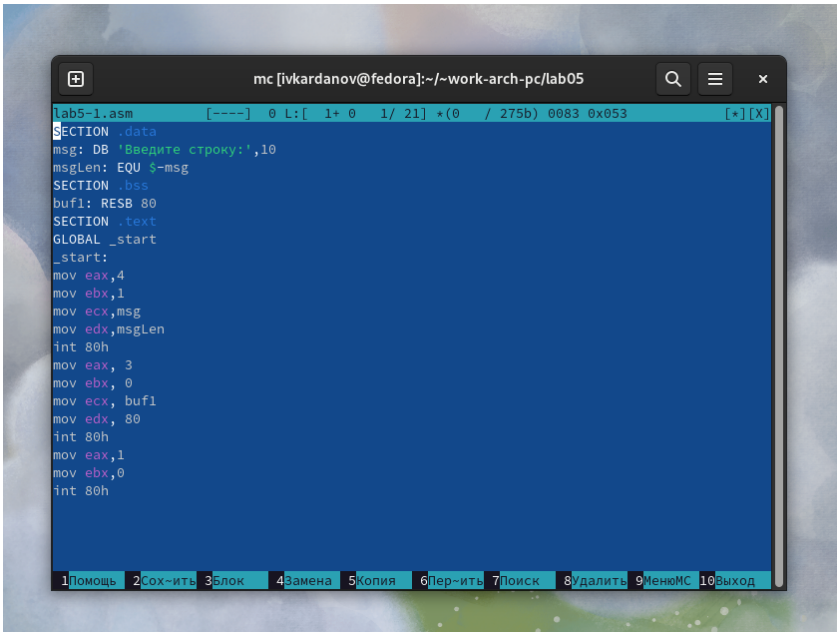

Рис. 4.4: Изменения файла

Для проверки открываю файл в режиме просмотра (рис. 4.5).

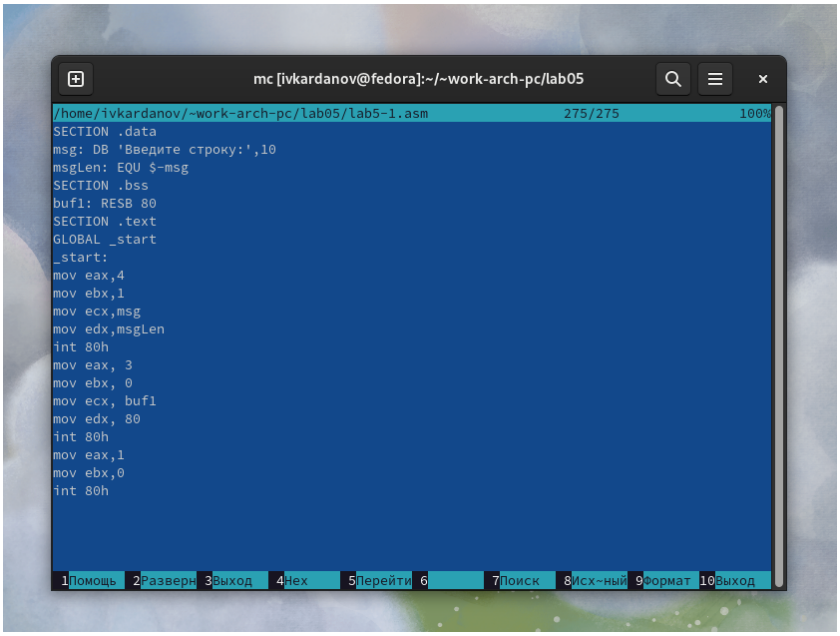

Рис. 4.5: Открытие файла для просмотра

Ввожу команду nasm -f elf lab5-1.asm. Создается объектный файл lab5-1.o. Ввожу команду ld -m elf_i386 -o lab5-1 lab5-1.o. Создается файл lab5-1. Запускаю исполняемый файл. Программа выводит “Введите строку:” ввожу свои ФИО, программа останавливается. (рис. 4.6).

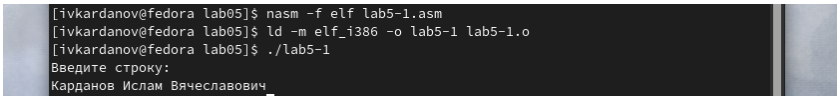

Рис. 4.6: Трансляция, обработка , запуск исполненяемого файла

4.2) Подключение внешнего файла

Скачиваю файл in_out.asm. Копирую файл in_out.asm из каталога “Загрузки”, в созданный каталог lab05. (рис. 4.7).

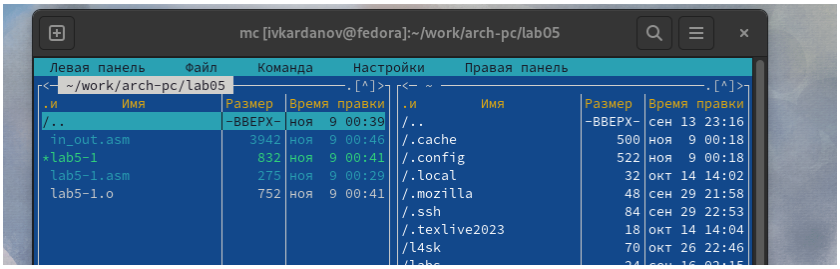

Рис. 4.7: Копирование файла

Далее копирую файл lab5-1 в тот же каталог, но с другим именем, для этого в окне mc прописываю путь к каталогу и новое имя файла. (рис. 4.8).

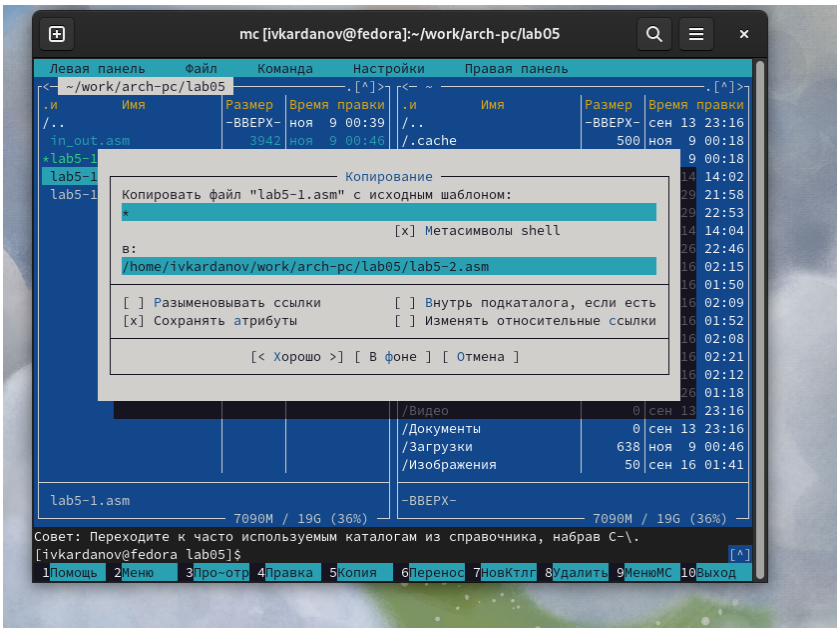

Рис. 4.8: Копирование файла

Изменяю содержимое файла lab05-2.asm, чтобы в программе использовались подпрограммы из внешнего файла inout.asm (рис. 4.9).

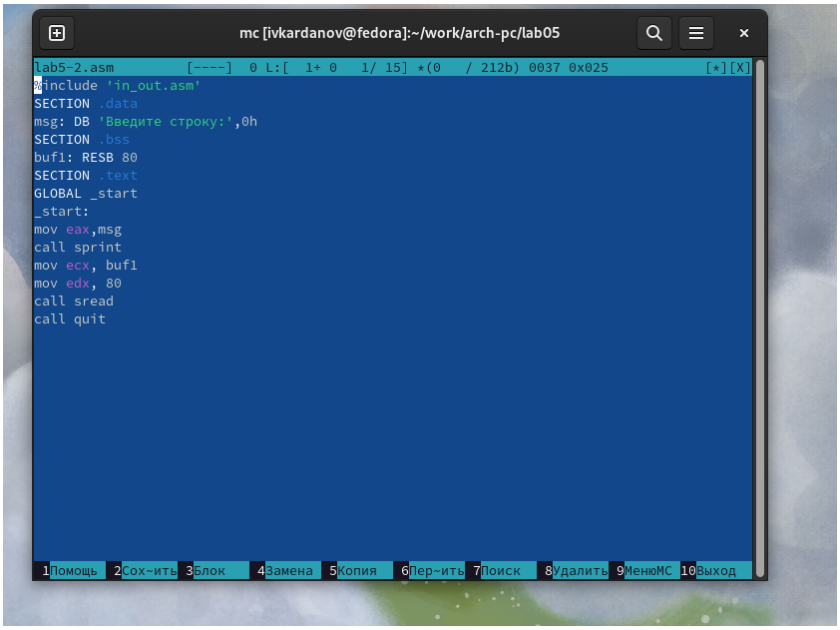

Рис. 4.9: Редактирование и подключение файла

Транслирую текст программы файла в объектный файл командой nasm -f elf lab5-2.asm. Создался объектный файл la5-2.o.Выполняю компоновку объектного файла с помощью команды ld -m elf_i386 -o lab5-2 lab5-2.o Создался исполняемый файл lab5-2. Запускаю его. (рис. 4.10).

Рис. 4.10: Запуск исполняемого файла

Открываю файл lab5-2.asm в режиме правки, меняю sprintLF на sprint. Далее проделываю те же действия что и раннее для выполнения файла. (рис. 4.11).

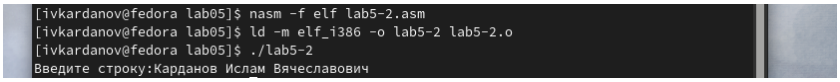

Рис. 4.11: Трансляция, обработка и запуск файла

Разница между первым исполняемым файлом и вторым в том, что запуск первого запрашивает ввод с новой строки, а программа, которая исполняется

при запуске второго, запрашивает ввод без переноса на новую строку, потому что в этом заключается различие между программами sprintLF и sprint.

4.3) Выполнение заданий для самостоятельной работы

Создаю копию файла lab5-1.asm с именем lab5-1-1.asm. (рис. 4.12).

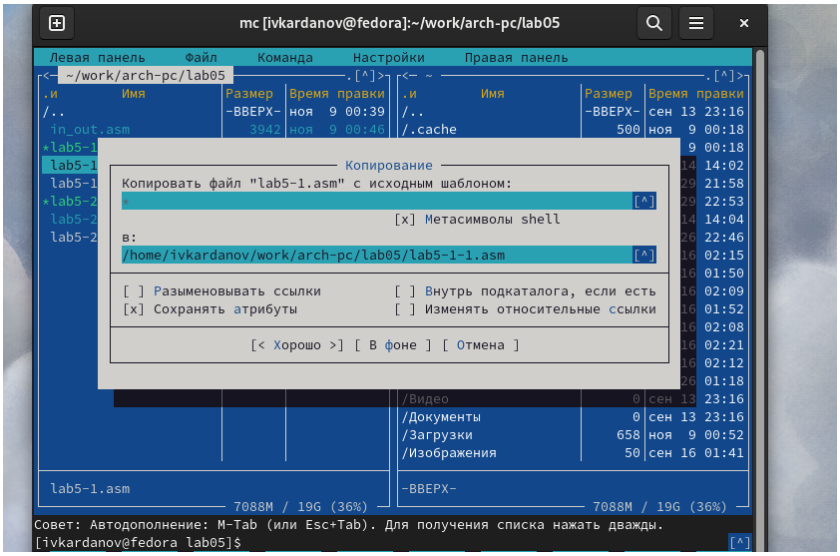

Рис. 4.12: Копирование файла

Изменяю программу так, чтобы кроме вывода приглашения и запроса ввода, она выводила вводимую пользователем строку. (рис. 4.13)

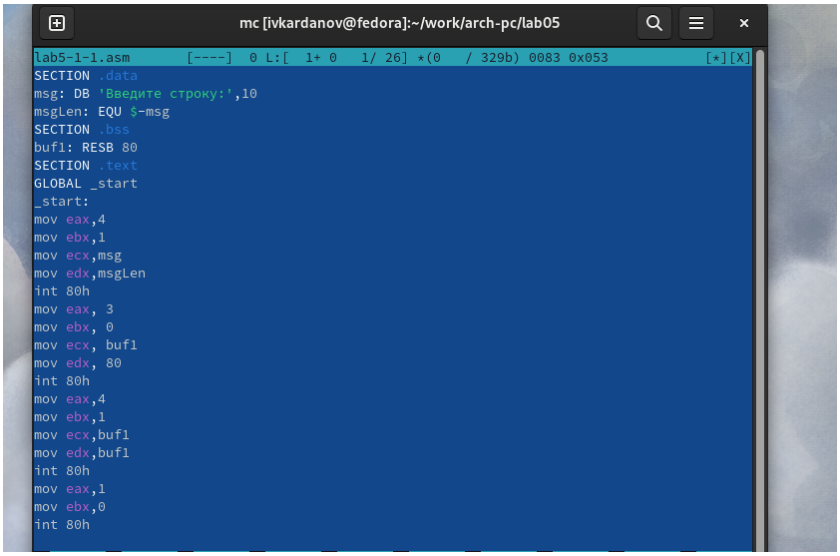

Рис. 4.13: Редактирование файла

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 ФИО, далее программа выводит введенные данные.(рис. 4.14)

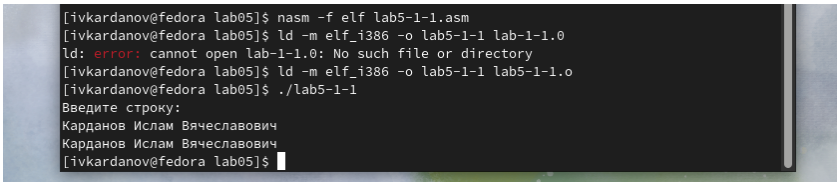

Рис. 4.14: Трансляция, обработка и запуск исполняемого файла

Создаю копию файла lab5-2.asm с именем lab5-2-1.asm. Изменяю программу так, чтобы кроме вывода приглашения и запроса ввода, она выводила вводимую пользователем строку. (рис. 4.15)

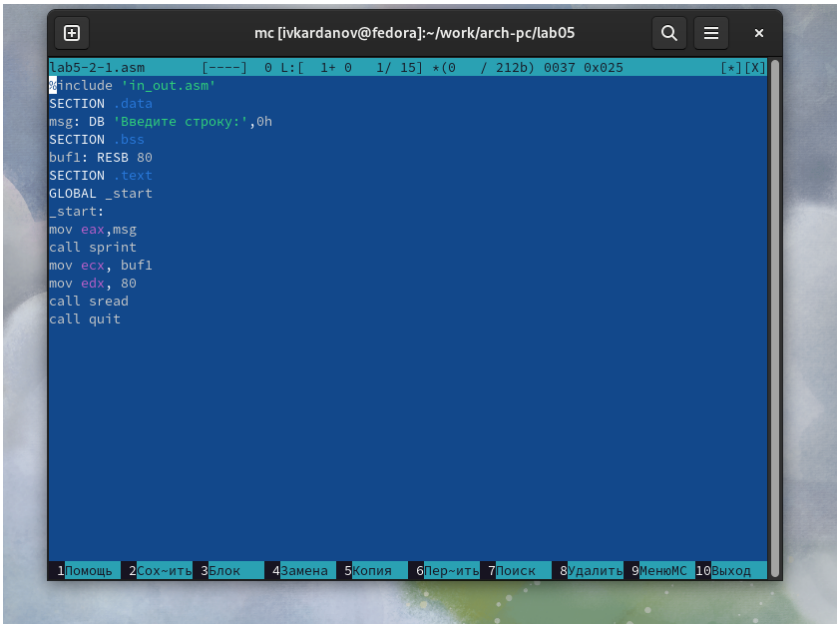

Рис. 4.15: Редактирование файлов

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, ввожу ФИО, далее программа выводит введенные данные.(рис. 4.16)

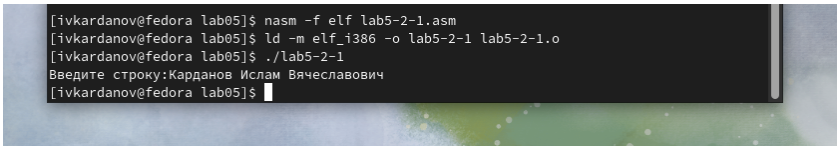

Рис. 4.16: Трансляция, обработка и запуск исполняемого файла

5 Выводы

При выполнении заданий я обрёл практический опыт работы с Midnight Commander и освоил инструкции языка ассемблера.

Список литературы

Архитектура ЭВМ