Отчёт по лабораторной работе №5

Дисциплина: архитектура компьютера

Кайнова Алина Андреевна

Содержание

# 1 Цель работы

Приобрести навыки по работе в программе Midnight Commander и освоить инструкции языка ассемблера NASM.

# 2 Задание

1. Основы работы с mc
2. Структура программы на языке ассемблера NASM
3. Подключение внешнего файла
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. 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: - DB (define byte) — определяет переменную размером в 1 байт; - DW (define word) — определяет переменную размеров в 2 байта (слово); - DD (define double word) — определяет переменную размером в 4 байта (двойное слово); - DQ (define quad word) — определяет переменную размером в 8 байт (учетве- рённое слово); - 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. Инструкция языка ассемблера mov предназначена для дублирования данных источника в приёмнике.

mov dst,src Здесь операнд dst — приёмник, а src — источник. В качестве операнда могут выступать регистры (register), ячейки памяти (memory) и непосредственные значения (const). Инструкция языка ассемблера intпредназначена для вызова прерывания с указанным номером. 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).

# 4 Выполнение лабораторной работы

##Основы работы с mc

Ввожу в терминал mc, чтобы открыть Midnight Commander

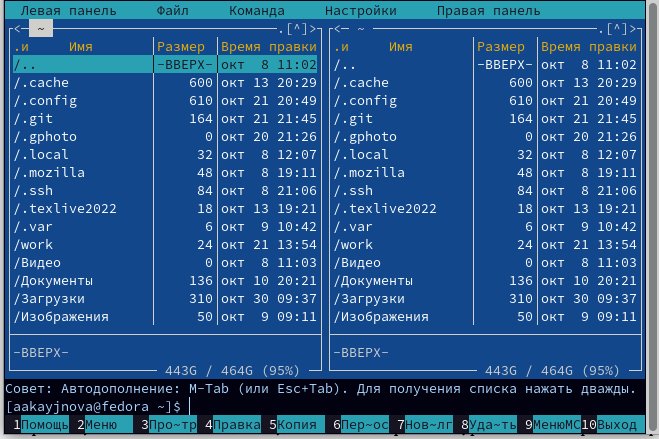


Figure 1: Открытый mc

Перехожу в каталог arch-pc, используя mc

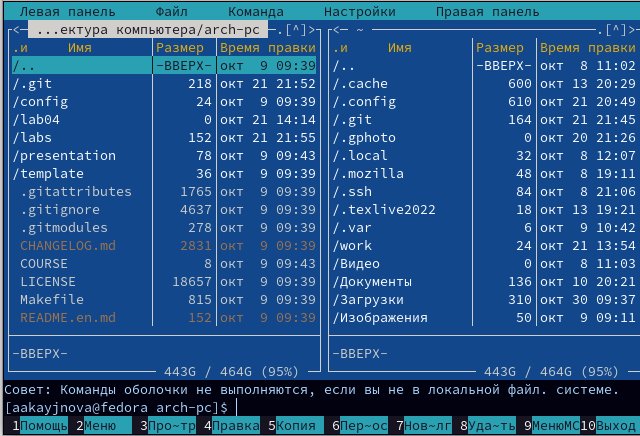


Figure 2: Перемещение между директориями

Создаю каталог lab05

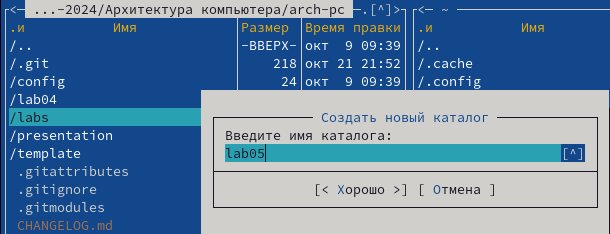


Figure 3: Создание каталога

Перехожу в созданный каталог

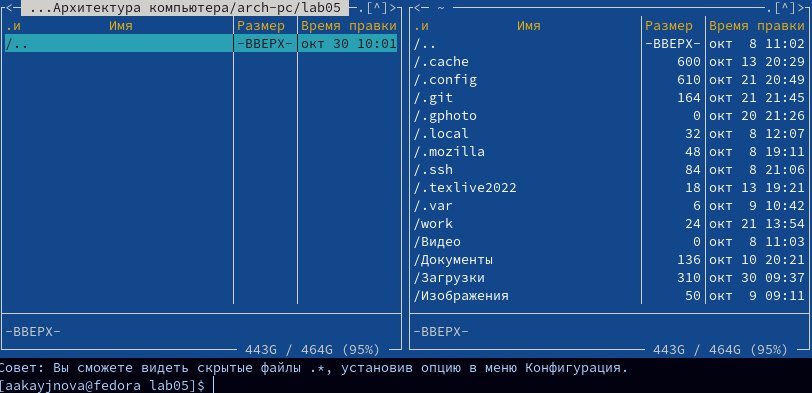


Figure 4: Перемещение между директориями

Создаю файл lab5-1.asm в котором дальше буду работать

Figure 5: Создание файла

Figure 5: Создание файла

##Структура программы на языке ассемблера NASM

Открываю созданный файл в редакторе nano

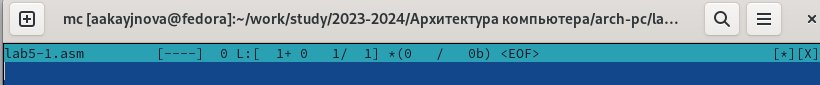


Figure 6: Открытие файла для редактирования

Копирую в файл код программы для запроса строки у пользователя

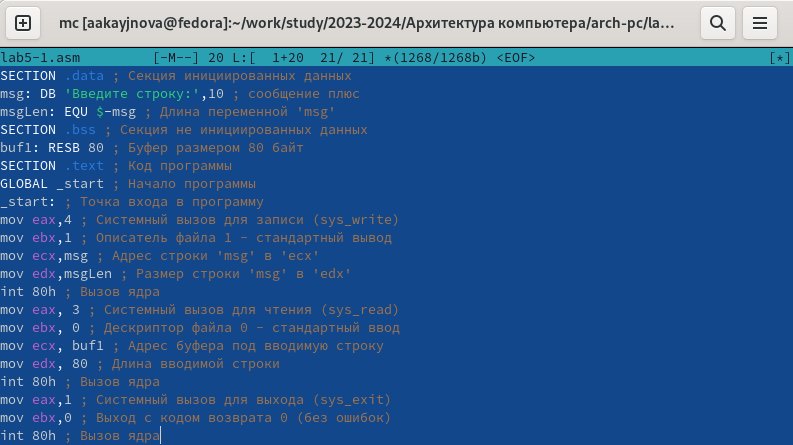


Figure 7: Редактирование файла

Открываю файл для проверки содержания нужного текста

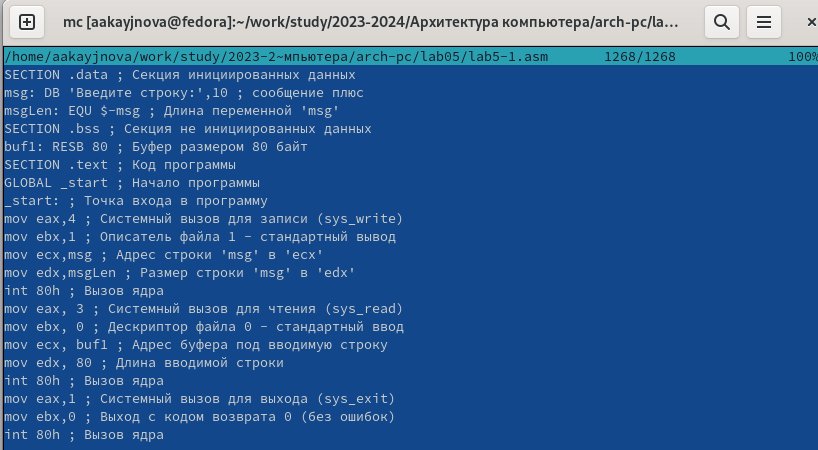


Figure 8: Открытие файла для просмотра

Транслирую текст программы файла в объектный файл: создаётся lab5-1.o

Figure 9: Компиляция файла

Figure 9: Компиляция файла

Выполняю компоновку объектного файла: создаётся исполняемый файл lab5-1

Figure 10: Передача на обработку копоновщику

Figure 10: Передача на обработку копоновщику

Запускаю исполняемый файл и ввожу свои ФИО

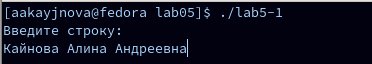


Figure 11: Исполнение файла

##Подключение внешнего файла

Нахожу в каталоге “Загрузки” скачанный файл in\_out.asm

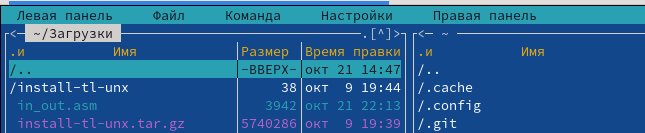


Figure 12: Скачанный файл

Копирую этот файл в каталог lab05

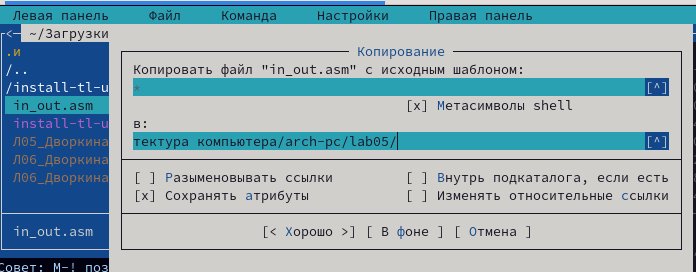


Figure 13: Копирование файла

Копирую файл lab5-1.asm в тот же каталог на с другим именем: lab5-2.asm

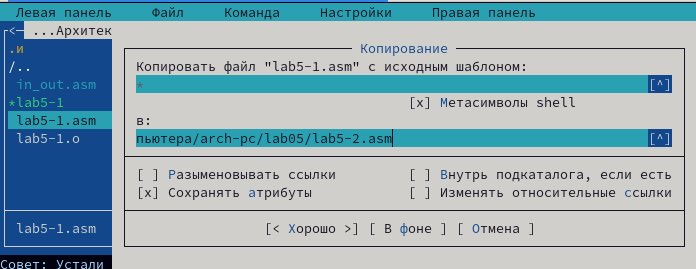


Figure 14: Копирование файла

Изменяю содержимое файла lab5-2.asm в редакторе nano, чтобы использовать подпрограммы из файла in\_out.asm

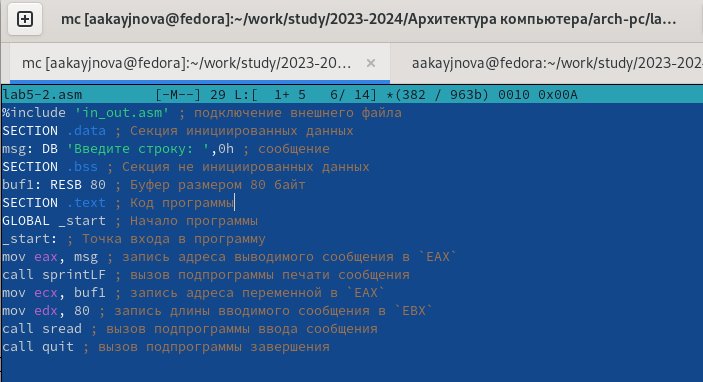


Figure 15: Редактирование файла

Транслирую текст программы файла в объектный файл: создаётся файл lab5-2.o

Figure 16: Компиляция файла

Figure 16: Компиляция файла

Выполняю компоновку объектного файла: создаётся исполняемый файл lab5-2

Figure 17: Передача на обработку компоновщику

Figure 17: Передача на обработку компоновщику

Запускаю исполняемый файл и ввожу свои ФИО

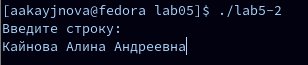


Figure 18: Исполнение файла

Открываю файл lab5-2.asm для редактирования в nano, изменяю подпрограмму sprintLF на sprint и открываю файл для просмотра сохранённых изменений

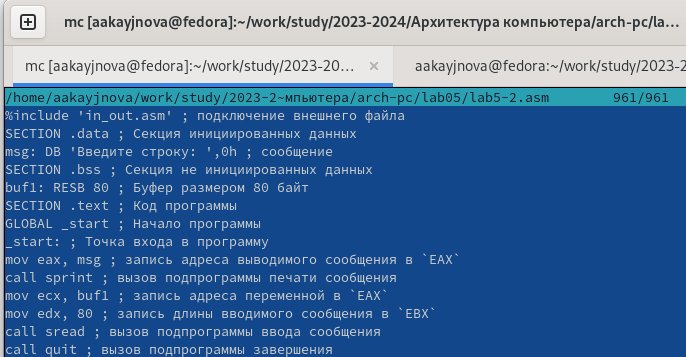


Figure 19: Отредактированный файл

Транслирую текст программы файла в объектный файл lab5-2.o, выполняю компоновку созданного файла(создаётся lab5-2-2) и проверяю правильность всех выполненных действий

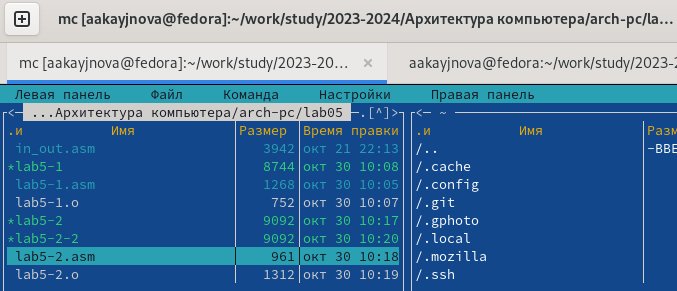


Figure 20: Проверка правильной компиляции и передачи на обработку файла

Запускаю исполняемый файл и ввожу свои ФИО

Figure 21: Исполнение файла

Figure 21: Исполнение файла

Разница между первым исполняемым файлом lab6-2 и вторым lab6-2-2 состоит 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 подпрограммами sprintLF и sprint.

##Выполнение заданий для самостоятельной работы

Создаю копию файла lab5-1.asm с именем lab5-1-1.asm

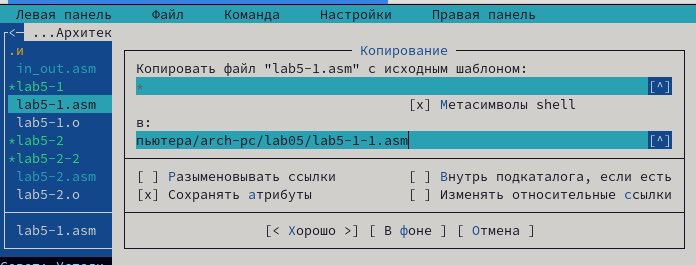


Figure 22: Копирование файла

Открываю созданный файл для редактирования и изменяю программу так, чтобы она ещё выводила введённую пользователем строку

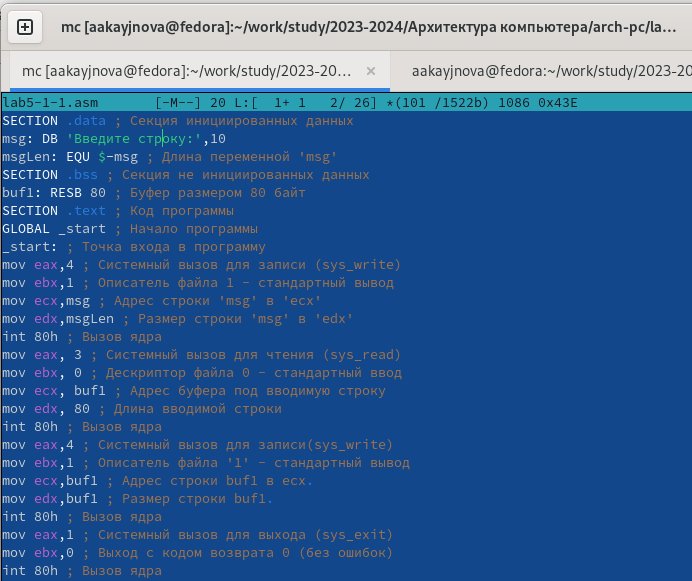


Figure 23: Редактирование файла

Транслирую текст программы файла в объектный файл: создаётся файл lab5-1-1.o

Figure 24: Компиляция файла

Figure 24: Компиляция файла

Выполняю компоновку объектного файла: создаётся исполняемый файл lab5-1-1

Figure 25: Передача на обработку компоновщику

Figure 25: Передача на обработку компоновщику

Запускаю исполняемый файл

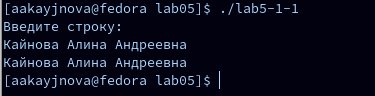


Figure 26: Исполнение файла

Создаю копию файла lab5-2.asm с именем lab5-2-1.asm

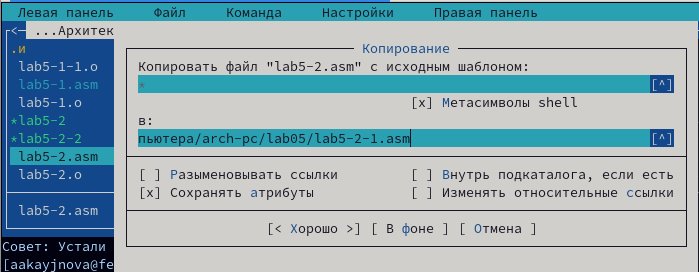


Figure 27: Копирование файла

Открываю созданный файл и изменяю программу так, чтобы она выводила ещё и введённую пользователем строку

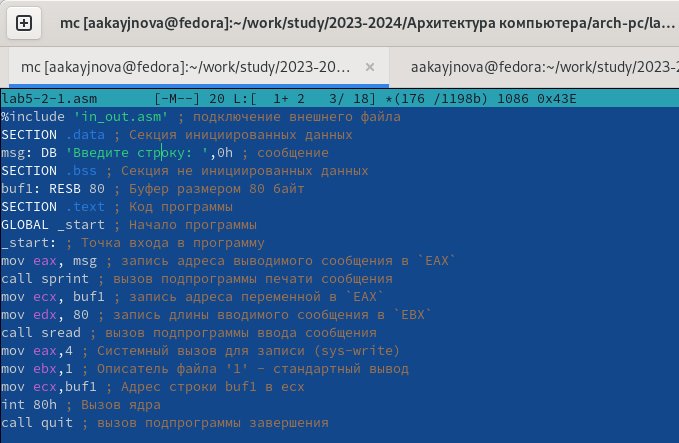


Figure 28: Редактирование файла

Транслирую текст программы файла в объектный файл: создаётся файл lab5-2-1.o (на скриншоте ошибка:должен быть lab-2-1.asm; потом создалосб всё правильно: на следующем снимке экрана это будет видно, там будет нужный файл)

Figure 29: Компиляция файла

Figure 29: Компиляция файла

Выполняю компоновку объектного файла: создаётся исполняемый файл lab5-1-1

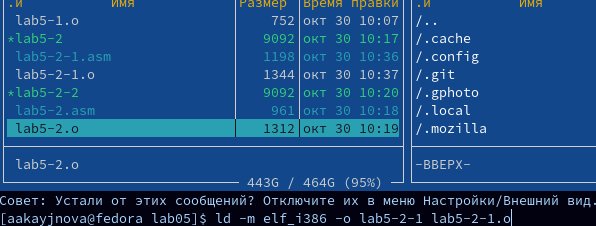


Figure 30: Передача на обработку компоновщику

Запускаю исполняемый файл

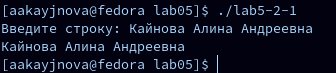


Figure 31: Исполнение файла

# 5 Выводы

В ходе данной лабораторной работы мы научились работать с Midnight Commander и освоили язык ассемблера NASM.

# Список литературы

1. 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