РОССИЙСКИЙ УНИВЕРСИТЕТ ДРУЖБЫ НАРОДОВ

Факультет физико-математических и естественных наук

направление: *Компьютерные и информационные науки*

Лабораторная работа №5

дисциплина: *Архитектура компьютеров и операционные системы*

студент: Гробман Александр Евгеньевич

Группа: НКАбд-02-23

Цель работы

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

Задание

1. Начало работы с Midnight Commander.
2. Подключение внешнего файла in_out.asm.
3. Создайте копию файла lab5-1.asm. Внесите изменения в программу (без использования внешнего файла in_out.asm), так чтобы она работала по следующему алгоритму:
4. Получите исполняемый файл и проверьте его работу. На приглашение ввести строку
   введите свою фамилию.
5. Создайте копию файла lab5-2.asm. Исправьте текст программы с использование подпрограмм из внешнего файла in_out.asm, так чтобы она работала по следующему
   алгоритму:
6. Создайте исполняемый файл и проверьте его работу

Теория

Midnight Commander — это программа, которая позволяет просматривать структуру каталогов и выполнять основные операции по управлению
файловой системой, т.е. MC является файловым менеджером. Для активации оболочки Midnight Commander достаточно ввести в командной строке mc и нажать
клавишу Enter

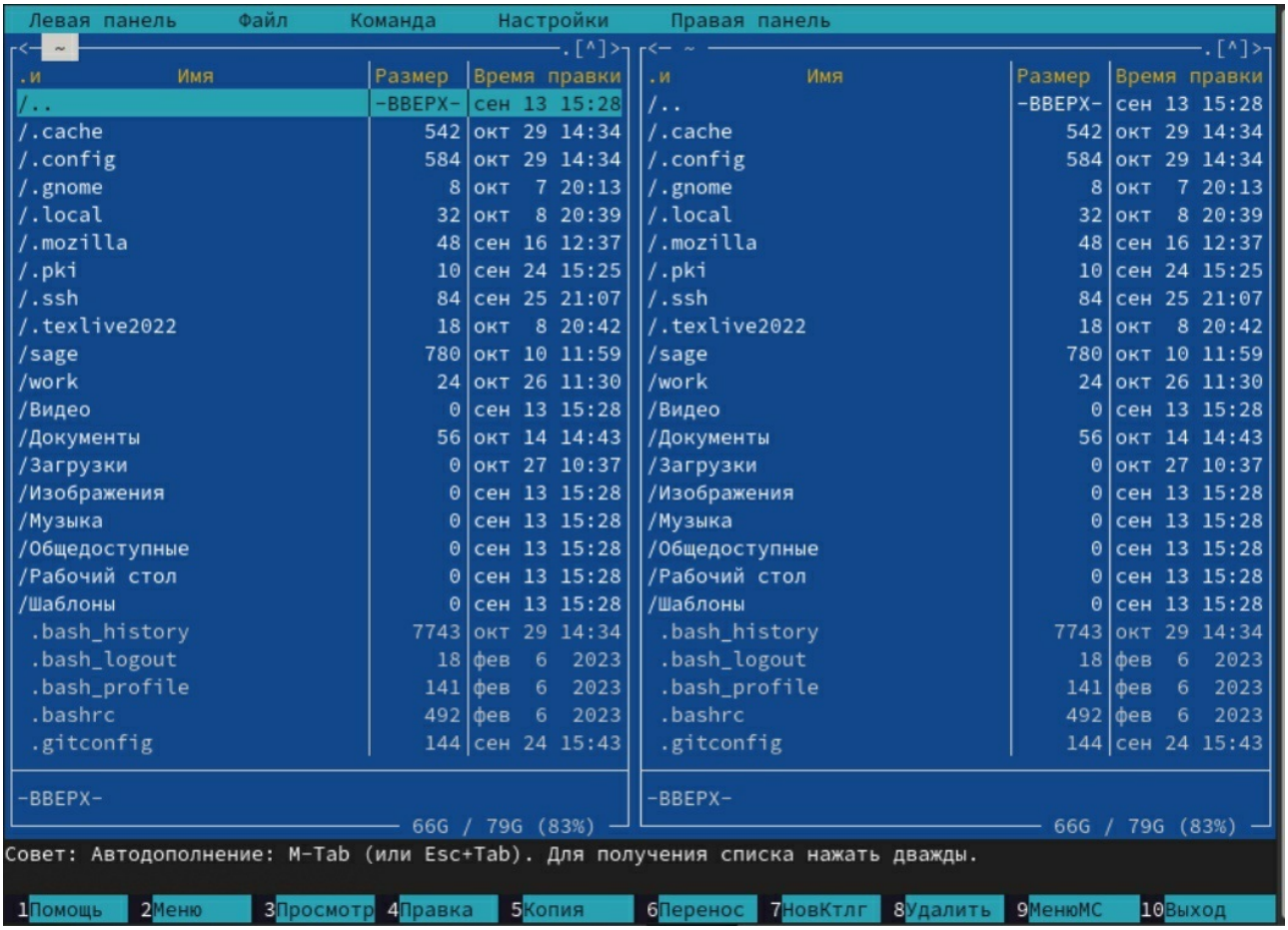
Программа на языке ассемблера NASM, как правило, состоит из трёх секций: секция кода программы (SECTION .text), секция инициированных (известных во время компиляции) данных (SECTION .data) и секция неинициализированных данных (тех, под которые во время компиляции только отводится память, а значение присваивается 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. Инструкция языка ассемблера mov предназначена для дублирования данных источника в приёмнике. 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 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. Простейший диалог с пользователем требует наличия двух функций — вывода текста на экран и ввода текста с клавиатуры. Простейший способ вывести строку на экран— использовать системный вызов write. Этот системный вызов имеет номер 4, поэтому перед вызовом инструкции int необходимо поместить значение 4 в регистр eax. Первым аргументом write, помещаемым в регистр ebx, задаётся дескриптор файла. Для вывода на экран в качестве дескриптора файла нужно указать 1 (это означает «стандартный вывод», т. е. вывод на экран). Вторым аргументом задаётся адрес выводимой строки (помещаем его в регистр ecx, например, инструкцией mov ecx, msg). Строка может иметь любую длину. Последним аргументом (т.е. в регистре edx) должна задаваться максимальная длина выводимой строки. Для ввода строки с клавиатуры можно использовать аналогичный системный вызов read. Его аргументы –такие же, как у вызова write,только для «чтения» с клавиатуры 9 используется файловый дескриптор 0 (стандартный ввод). Системный вызов exit является обязательным в конце любой программы на языке ассемблер. Для обозначения конца программы перед вызовом инструкции int 80h необходимо поместить в регистр eax значение 1, а в регистр ebx код завершения 0.

Выполнение лабораторной работы

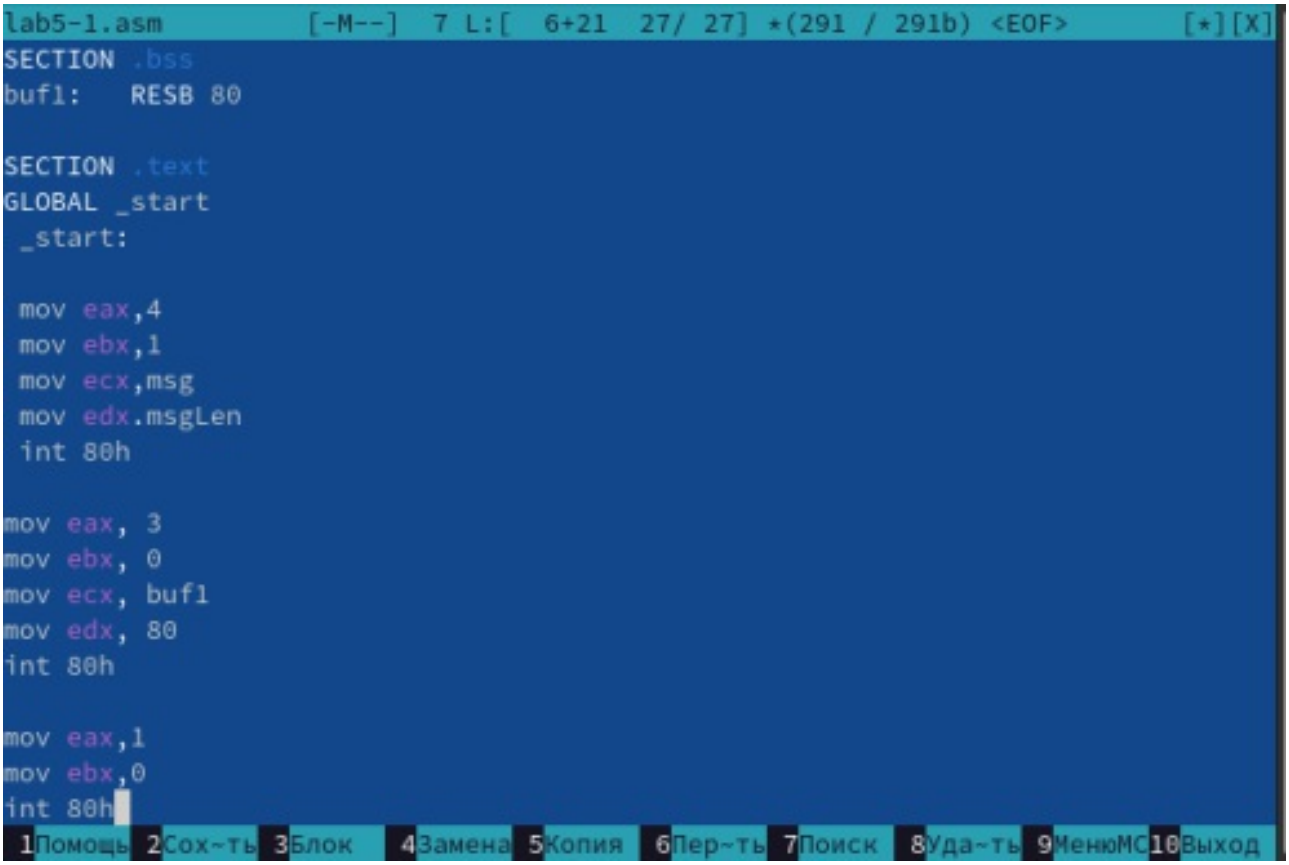
1. **Начало работы с Midnight Commander**

Открываем Midnight Commander



Пользуясь клавишами вверх, вниз и Enter перехожу в каталог ~/work/arch-pc,
с помощью клавиши F7 создаю папку lab05 и перехожу в созданный каталог.
открываем mousepad, заполняем программу

С помощью клавиши F4 открываю файл lab5-1.asm для редактирования во встроенном редакторе, ввожу текст программы из листинга 5.1,
сохраняю изменения и закрываю файл.

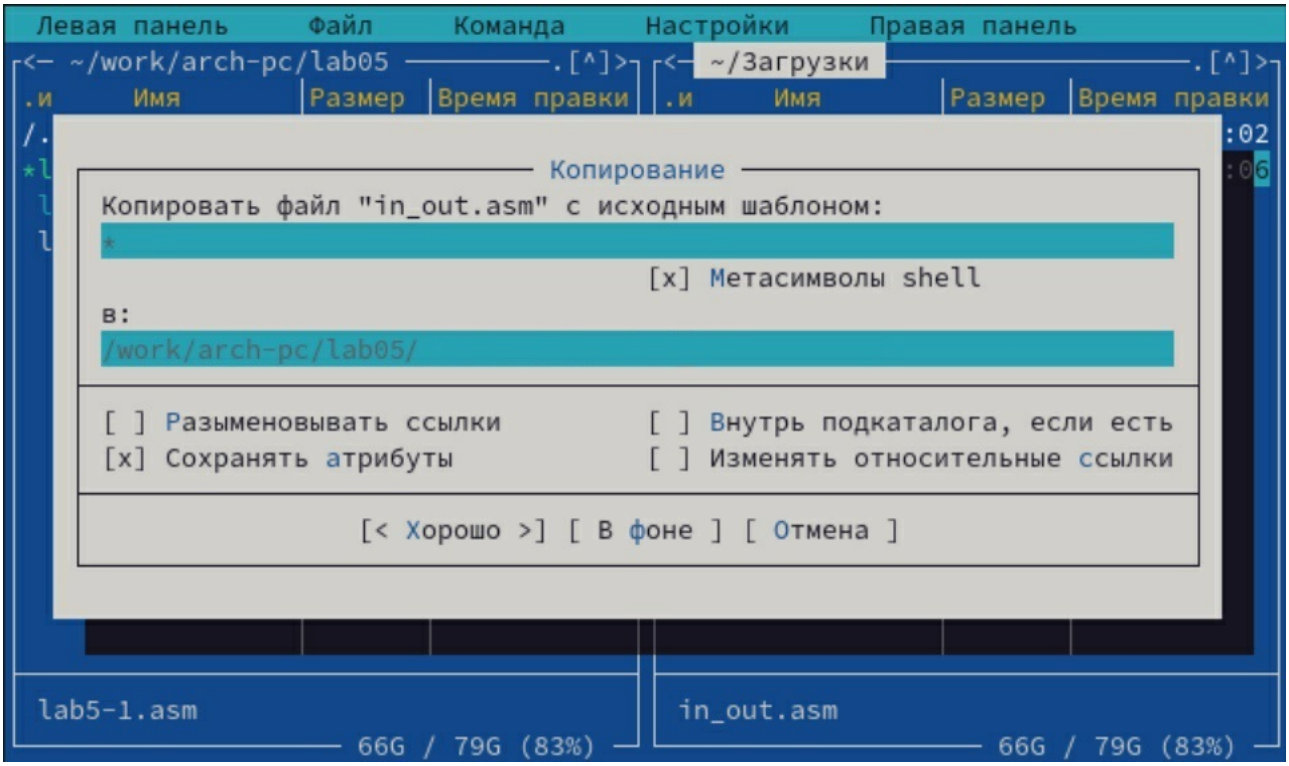


Открываю файл lab5-1.asm для просмотра и убеждаюсь, что файл содержит текст программы

Компилирую текст программы lab5-1.asm в объектный файл, выполняю компоновку и запускаю получившийся файл
в строку ввожу свои ФИО.

2. **Подключение внешнего файла in_out.asm**

Скачиваю файл in_out.asm с ТУИС. В первой панели открываю каталог с файлом lab5-1.asm, а во второй панели каталог со скачанным файлом in_out.asm.
Копирую файл in_out.asm в каталог с файлом lab5-1.asm с помощью F5.



С помощью клавиши F6 создаю копию файла lab5-1.asm с именем lab5-2.asm.

Исправляю текст программы в файле lab5-2.asm в соответствии с листингом 5.2.
Создаю исполняемый файл и проверяю его работу

```
Введите строку: Гробман Александр Евгеньевич
```

3. **Создайте копию файла lab5-1.asm**

Создаю копию файла lab5-1.asm.
Вношу изменения в программу так, чтобы она выводила введённую строку на экран.

```
mov edx,msgLen
int 80h



mov eax, 3
mov ebx, 0
mov ecx, buf1
mov edx, 80
int 80h


mov eax,4
mov ebx,1
mov ecx,buf1
mov edx,buf1
int 80h
mov eax,1
mov ebx,0
int 80h
```

4. **Создайте копию файла lab5-2.asm**

Получаю исполняемый файл и проверяю его работу.

```
Введите строку:
Гробман А.Е
Гробман А.Е
```

5. **проверяем работу программы**

```
[aegrobman lab05]$ ./lab5-2-2
Введите строку: Гробман А.Е
Гробман А.Е
```

4. **Отправляем файлы на гитхаб.**

Ссылка на отчёт https://github.com/DaOneme/AEGrobman_study_2023-2024_arhpc/tree/main/Labs/Lab05