Отчёт по лабораторной работе No5

Основы работы с Midnight Commander (mc)

Четвергова Мария

Содержание

# 1 Цель работы

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

# 2 Теоретическое введение

##Структура программы на языке ассемблера NASM##

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.

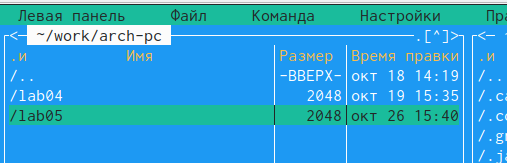
##Системные вызовы для обеспечения диалога с пользователем##

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 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 еах значение 1, а в регистр ebx код завершения 0.

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 систем- ного 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 по- рядке следования в остальные регистры процессора: ebx, ecx, edx. Если системная функция должна вернуть значение, то она помещает его в регистр eax.

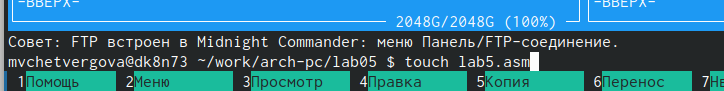
# 3 Выполнение лабораторной работы

1. Откройте Midnight Commander
2. Пользуясь клавишами ↑ , ↓ и Enter перейдите в каталог ~/work/arch-pc созданный при выполнении лабораторной работы No4
3. С помощью функциональной клавиши F7 создайте папку lab05 и перейдите в созданный каталог.



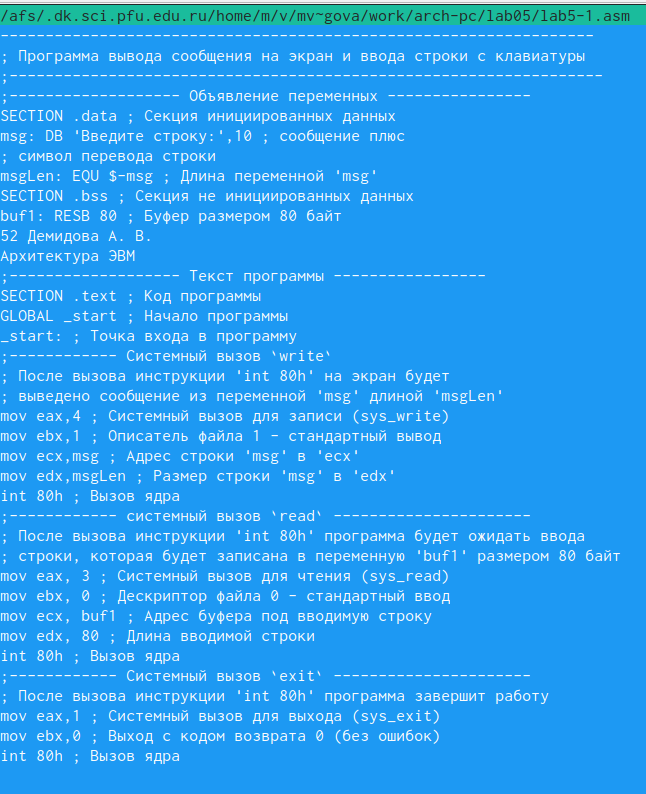
Создание папку lab05

1. Пользуясь строкой ввода и командой touch создайте файл lab5-1.asm



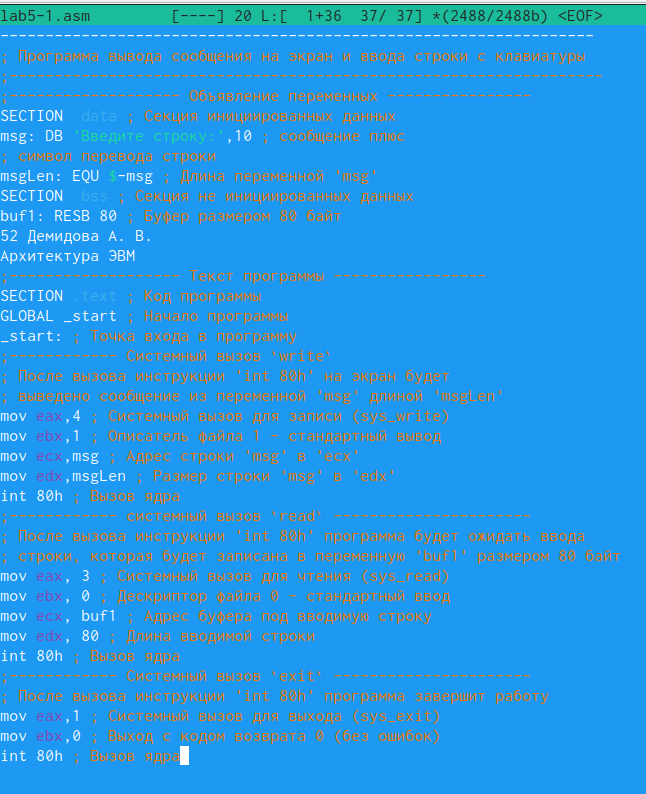
Создание файла lab5-1.asm

1. С помощью функциональной клавиши F4 откройте файл lab5-1.asm для редактирования во встроенном редакторе. Как правило в качестве встроенного редактора Midnight Commander используется редакторы nano или mcedit
2. Введите текст программы из листинга 5.1 (можно без комментариев), сохраните изменения и закройте файл.



Текст программы из листинга 5.1 в файле

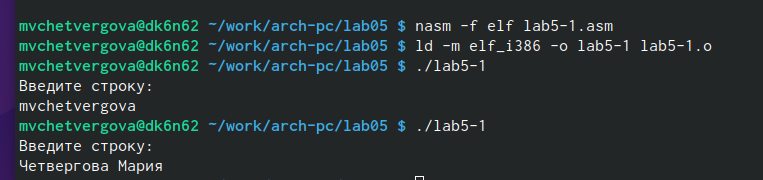
1. С помощью функциональной клавиши F3 откройте файл lab5-1.asm для просмотра. Убедитесь, что файл содержит текст программы.
2. Оттранслируйте текст программы lab5-1.asm в объектный файл. Выполните компоновку объектного файла и запустите получившийся исполняемый файл.



Оттранслируем текст программы lab5-1.asm в объектный файл

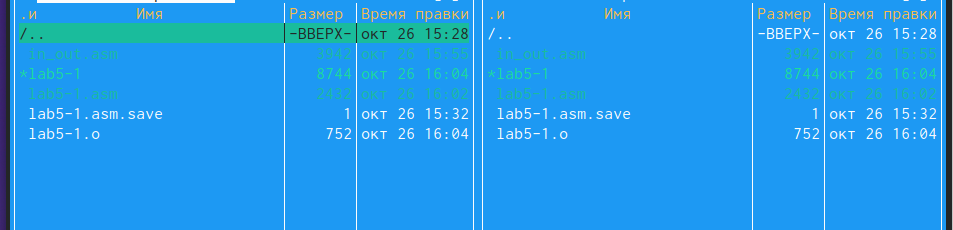
##5.3.1. Подключение внешнего файла in\_out.asm## Для упрощения написания программ часто встречающиеся одинаковые участки кода (такие как, например, вывод строки на экран или выход их программы) можно оформить в виде подпрограмм и сохранить в отдельные файлы, а во всех нужных местах поставить вызов нужной подпрограммы. Это позволяет сделать основную программу более удобной для написания и чтения.

1. Скачайте файл in\_out.asm со страницы курса в ТУИС.
2. Подключаемый файл in\_out.asm должен лежать в том же каталоге, что и файл с программой, в которой он используется. В одной из панелей mc откройте каталог с файлом lab5-1.asm. В другой панели каталог со скаченным файлом in\_out.asm (для перемещения между панелями используйте Tab ). Скопируйте файл in\_out.asm в каталог с файлом lab5-1.asm с помощью функциональной клавиши



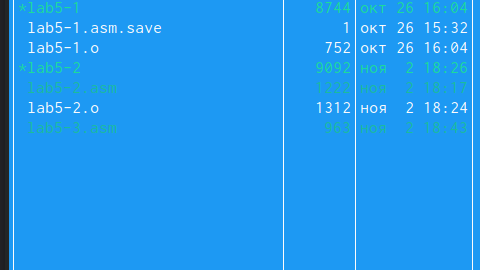
Скопируйте файл в каталог с помощью функциональной клавиши

1. С помощью функциональной клавиши F6 создайте копию файла lab5-1.asm с именем lab5-2.asm. Выделите файл lab5-1.asm, нажмите клавишу F6 , введите имя файла lab5-2.asm и нажмите клавишу Enter.



Cоздание копии файла lab5-1.asm с именем lab5-2.asm.

1. Исправьте текст программы в файле lab5-2.asm с использование подпрограмм из внешнего файла in\_out.asm (используйте подпрограммы sprintLF, sread и quit) в соответствии с листингом 5.2. Создайте исполняемый файл и проверьте его работу.

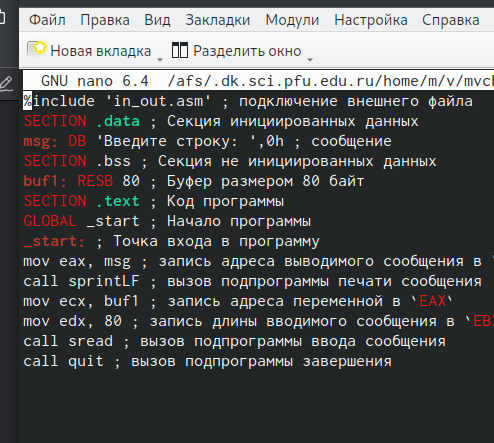


Создание исполняемого файла

В файле lab5-2.asm замените подпрограмму sprintLF на sprint. Создайте исполняемый файл и проверьте его работу.

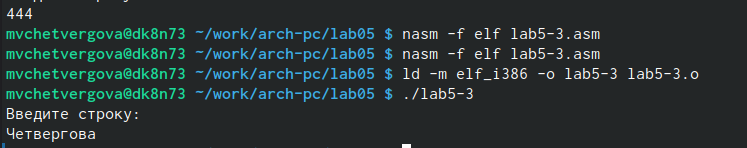
## 3.1 Задание для самостоятельной работы

1. Создайте копию файла lab5-1.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;



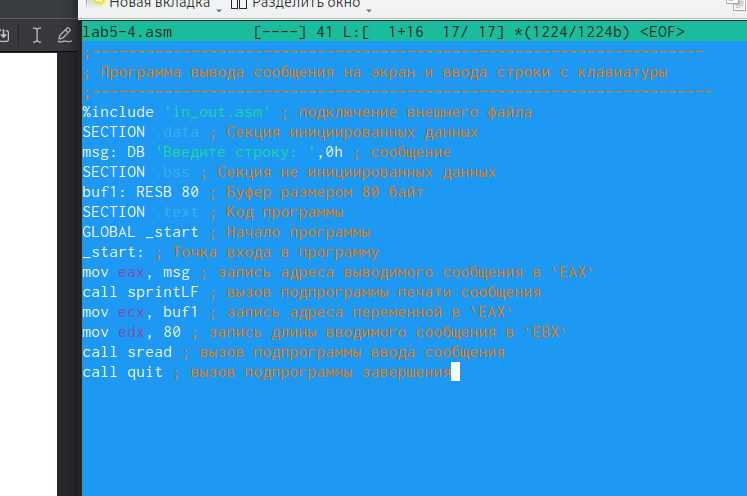
Создание копии файла lab5-1.asm

1. Получите исполняемый файл и проверьте его работу. На приглашение ввести строку введите свою фамилию.



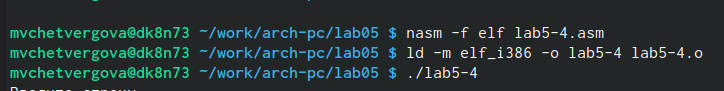
Получение исполняемого файла и проверка его работы

1. 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:



Исправьте текст программы с использование подпрограмм из внешнего файла in\_out.asm

1. Создайте исполняемый файл и проверьте его работу



Создайте исполняемый файл и проверьте его работу

# 4 Выводы

Мы приобрели практические навыки работы в Midnight Commander, а также освоили инструкции языка ассемблера mov и int.