Лабораторная работа №5

Основы работы с Midnight Commander (mc). Структура программы на языке ассемблера NASM. Системные вызовы в ОС GNU Linux

Богданюк Анна Васильевна

Содержание

# Цель работы

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

# Задание

1. Выполнение лабораторной работы
2. Задания для самостоятельной работы

# 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).

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.

# Выполнение лабораторной работы

1. Выполнение лабораторной работы

Окрываю Midnight Commander с помощью mc (рис. [1](#fig:001)).

Figure 1: Открываю Midnight Commander

Figure 1: Открываю Midnight Commander

Midnight Commander (рис. [2](#fig:002)).

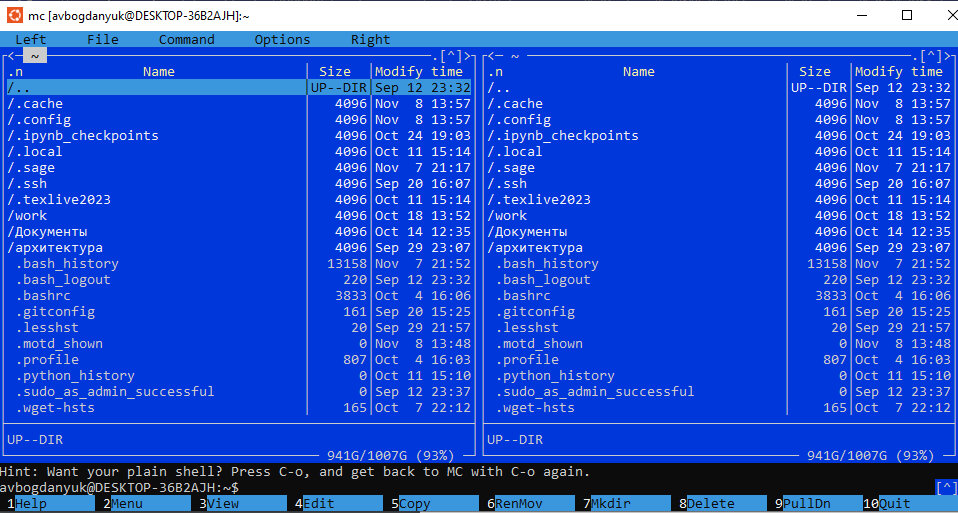


Figure 2: Midnight Commander

Перехожу в каталог, созданный в ходе лабораторной работы №4 (рис. [3](#fig:003)).

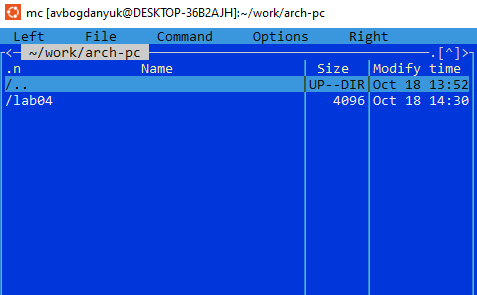


Figure 3: Перекожу в каталог

Создаю новый каталог lab05 (рис. [4](#fig:004)).

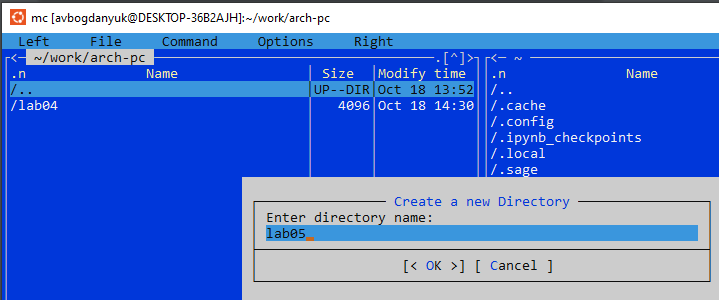


Figure 4: Создаю lab05

Перехожу в новый каталог(рис. [5](#fig:005)).

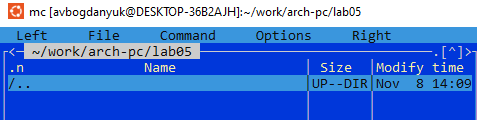


Figure 5: Перехожу в lab05

С помощью touch создаю новый файл lab5-1.asm (рис. [6](#fig:006)).

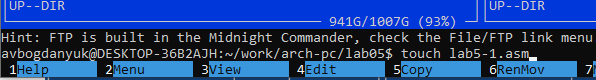


Figure 6: Создание файла

Редактирую созданный файл (рис. [7](#fig:007)).

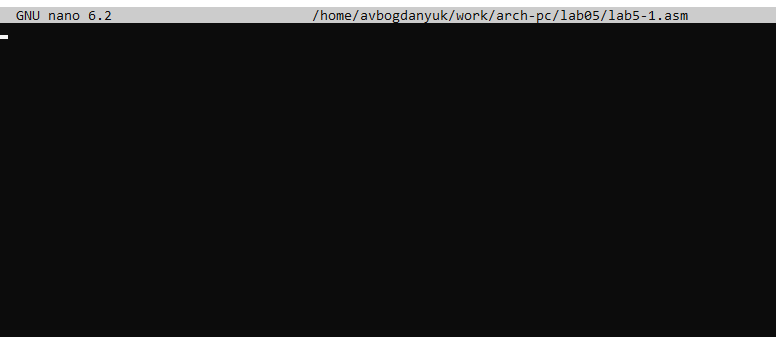


Figure 7: Редактирую файл

Ввожу текст из листинга, сохраняю файл и закрываю его (рис. [8](#fig:008)).

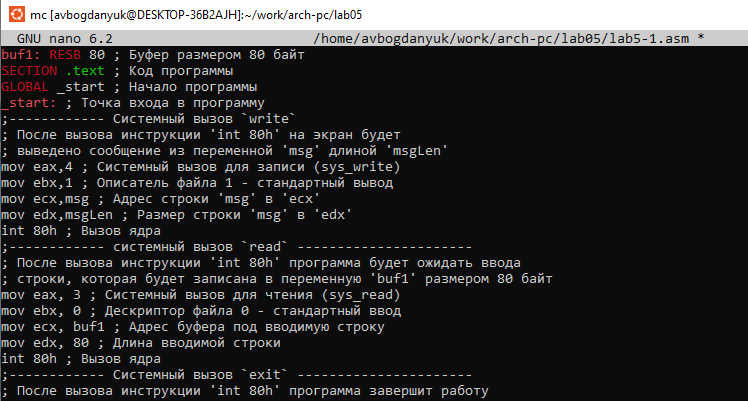


Figure 8: Ввод текста

Листинг программы:

SECTION .data ; Секция инициированных данных  
msg: DB 'Введите строку:',10 ; сообщение плюс  
; символ перевода строки  
msgLen: EQU $-msg ; Длина переменной 'msg'  
SECTION .bss ; Секция не инициированных данных  
buf1: RESB 80 ; Буфер размером 80 байт  
SECTION .text ; Код программы  
GLOBAL \_start ; Начало программы  
\_start: ; Точка входа в программу  
;------------ Cистемный вызов `write`  
; После вызова инструкции 'int 80h' на экран будет  
; выведено сообщение из переменной 'msg' длиной 'msgLen'  
mov eax,4 ; Системный вызов для записи (sys\_write)  
mov ebx,1 ; Описатель файла 1 - стандартный вывод  
mov ecx,msg ; Адрес строки 'msg' в 'ecx'  
mov edx,msgLen ; Размер строки 'msg' в 'edx'  
int 80h ; Вызов ядра  
;------------ системный вызов `read` ----------------------  
; После вызова инструкции 'int 80h' программа будет ожидать ввода  
; строки, которая будет записана в переменную 'buf1' размером 80 байт  
mov eax, 3 ; Системный вызов для чтения (sys\_read)  
mov ebx, 0 ; Дескриптор файла 0 - стандартный ввод  
mov ecx, buf1 ; Адрес буфера под вводимую строку  
mov edx, 80 ; Длина вводимой строки  
int 80h ; Вызов ядра  
;------------ Системный вызов `exit` ----------------------  
; После вызова инструкции 'int 80h' программа завершит работу  
mov eax,1 ; Системный вызов для выхода (sys\_exit)  
mov ebx,0 ; Выход с кодом возврата 0 (без ошибок)  
int 80h ; Вызов ядра"

С помощью функциональной клавиши F3 открываю файл для просмотра. Убеждаюсь, что файл содержит текст программы (рис. [9](#fig:009)).

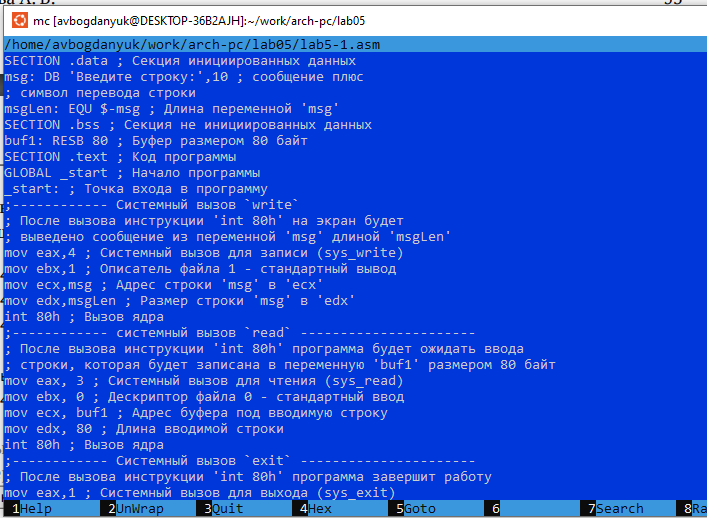


Figure 9: Открываю файл для просмотра

Оттранслирую текст программы lab5-1.asm в объектный файл (рис. [10](#fig:010)).

Figure 10: Оттранслрую текст программы в объектный файл

Figure 10: Оттранслрую текст программы в объектный файл

Выполняю компановку объектного файла (рис. [11](#fig:011)).

Figure 11: Выполнение компановки

Figure 11: Выполнение компановки

Запускаю получившийся файл и ввожу своё ФИО (рис. [12](#fig:012)).

Figure 12: Запуск файла

Figure 12: Запуск файла

Скачиваю файл in\_out.asm (рис. [13](#fig:013)).

Figure 13: Файл in_out.asm

Figure 13: Файл in\_out.asm

Так как файл in\_out.asm должен находиться в одном каталоге с файлом с программой, в которой он используется. Открываю в одной панели mc каталог с файлом lab5-1.asm. В другой панели каталог со скаченным файлом in\_out.asm. (рис. [14](#fig:014)).

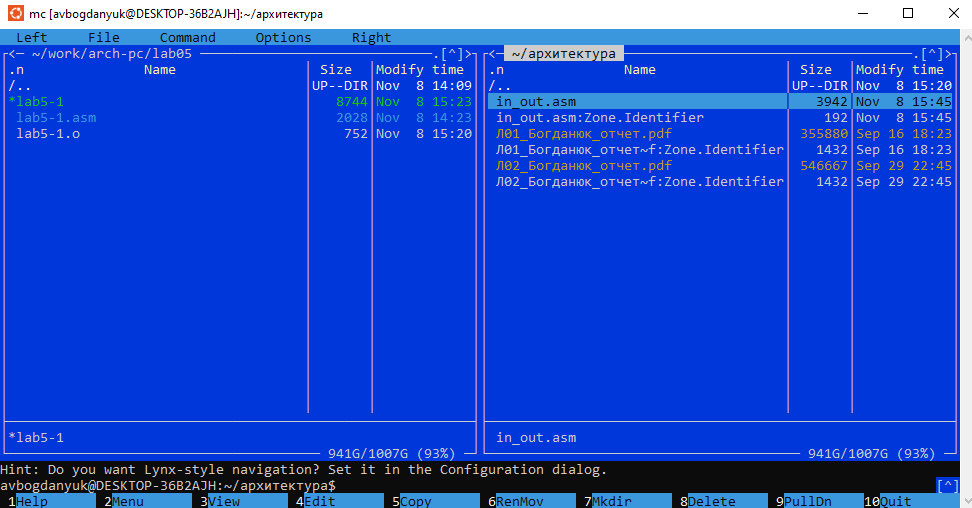


Figure 14: Открываю каталоги в двух панелях

Копирую файл in\_out.asm в каталог с файлом lab5-1.asm (рис. [15](#fig:015)).

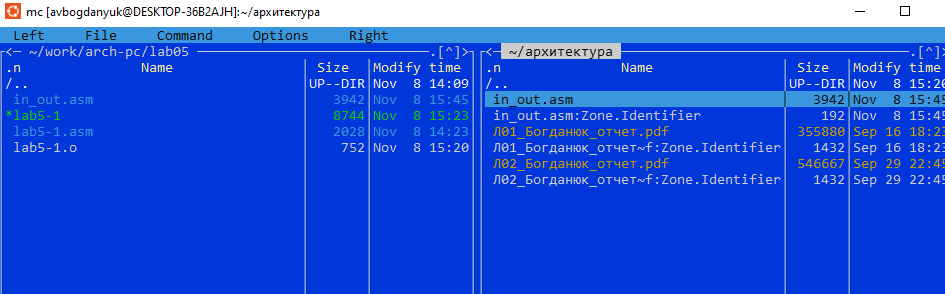


Figure 15: Копирую файл

Копирую файл lab5-1.asm с именем lab5-2.asm (рис. [16](#fig:016)).

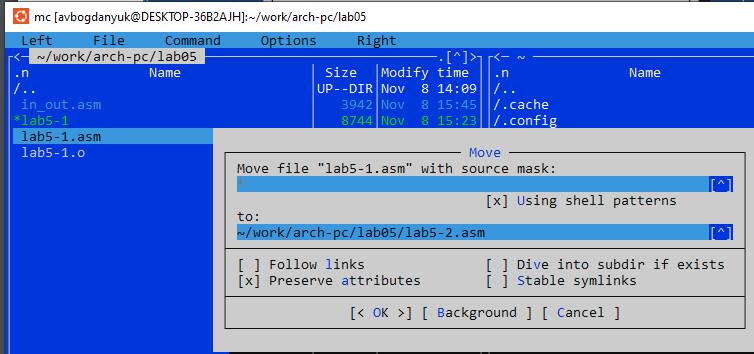


Figure 16: Копирую файл

Исправляю текст программы lab5-2.asm, где меняю sprintLF на sprint (рис. [17](#fig:017)).

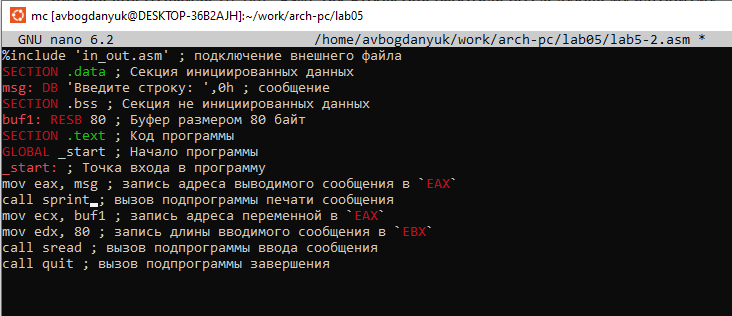


Figure 17: Исправляю файл

Листинг программы:

"%include 'in\_out.asm' ; подключение внешнего файла  
SECTION .data ; Секция инициированных данных  
msg: DB 'Введите строку: ',0h ; сообщение  
SECTION .bss ; Секция не инициированных данных  
buf1: RESB 80 ; Буфер размером 80 байт  
SECTION .text ; Код программы  
GLOBAL \_start ; Начало программы  
\_start: ; Точка входа в программу  
mov eax, msg ; запись адреса выводимого сообщения в `EAX`  
call sprint ; вызов подпрограммы печати сообщения  
mov ecx, buf1 ; запись адреса переменной в `EAX`  
mov edx, 80 ; запись длины вводимого сообщения в `EBX`  
call sread ; вызов подпрограммы ввода сообщения  
call quit ; вызов подпрограммы завершения"

Создаю исполнительный файл для lab5-2.asm и проверью его работу. Разница: sprintLF – работает аналогично sprint, но при выводе на экран добавляет к сообщению символ перевода строки (рис. [18](#fig:018)).

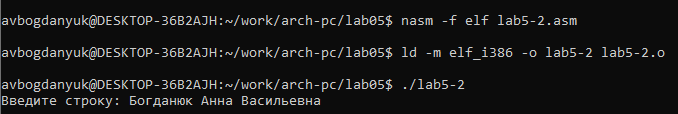


Figure 18: Создание и вывод

1. Задания для самостоятельной работы

Создаю копию файла lab5-1.asm с названием lab5-3.asm (рис. [19](#fig:019)).

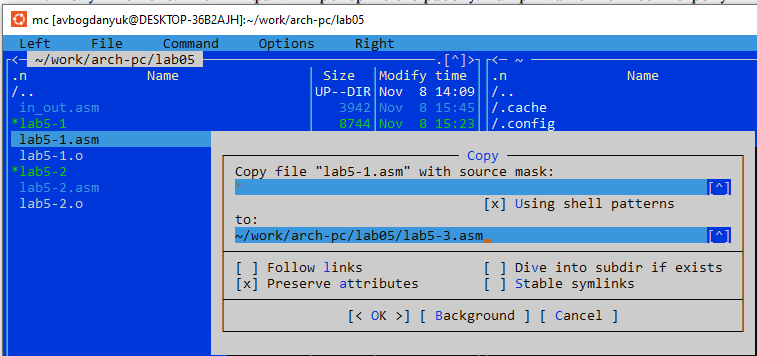


Figure 19: Копирую файл

Изменяю файл, чтобы был ввывод строки, ввод пользователем и вывод строки (рис. [20](#fig:020)).

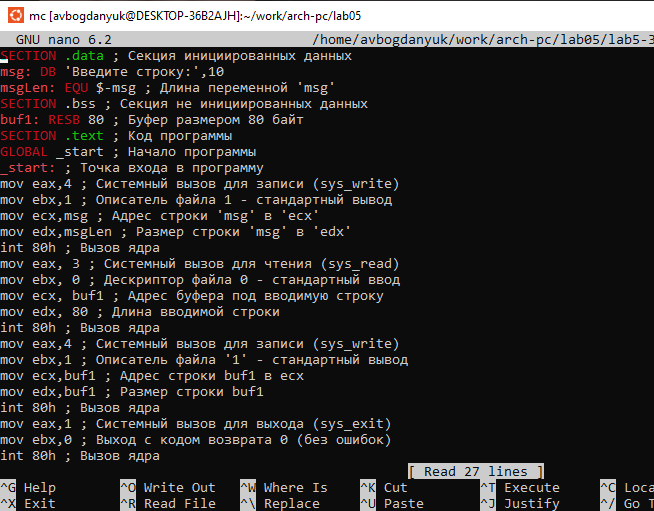


Figure 20: Изменение файла

Листинг программы:

"SECTION .data ; Секция инициированных данных  
msg: DB 'Введите строку:',10  
msgLen: EQU $-msg ; Длина переменной 'msg'  
SECTION .bss ; Секция не инициированных данных  
buf1: RESB 80 ; Буфер размером 80 байт  
SECTION .text ; Код программы  
GLOBAL \_start ; Начало программы  
\_start: ; Точка входа в программу  
mov eax,4 ; Системный вызов для записи (sys\_write)  
mov ebx,1 ; Описатель файла 1 - стандартный вывод  
mov ecx,msg ; Адрес строки 'msg' в 'ecx'  
mov edx,msgLen ; Размер строки 'msg' в 'edx'  
int 80h ; Вызов ядра  
mov eax, 3 ; Системный вызов для чтения (sys\_read)  
mov ebx, 0 ; Дескриптор файла 0 - стандартный ввод  
mov ecx, buf1 ; Адрес буфера под вводимую строку  
mov edx, 80 ; Длина вводимой строки  
int 80h ; Вызов ядра  
mov eax,4 ; Системный вызов для записи (sys\_write)  
mov ebx,1 ; Описатель файла '1' - стандартный вывод  
mov ecx,buf1 ; Адрес строки buf1 в ecx  
mov edx,buf1 ; Размер строки buf1  
int 80h ; Вызов ядра  
mov eax,1 ; Системный вызов для выхода (sys\_exit)  
mov ebx,0 ; Выход с кодом возврата 0 (без ошибок)  
int 80h ; Вызов ядра"

Запускаю исполнительный файл (рис. [21](#fig:021)).

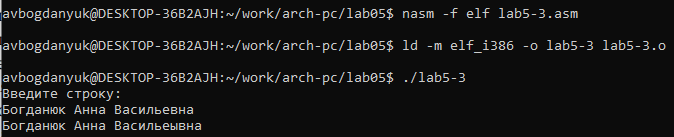


Figure 21: Вывод

Копирую файл lab5-2.asm с именем lab5-4.asm (рис. [22](#fig:022)).

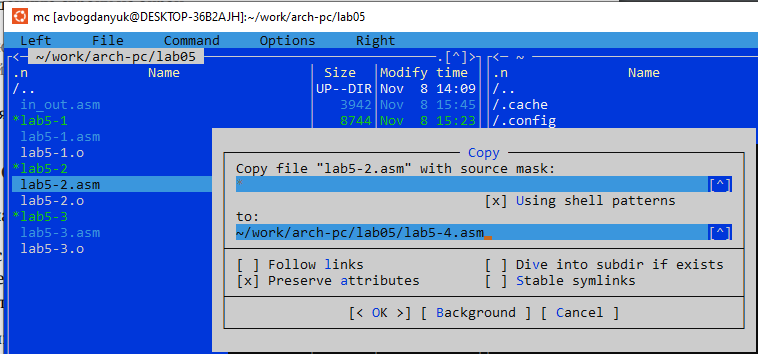


Figure 22: Копирование файла

Изменяю файл, чтобы был ввывод строки, ввод пользователем и вывод строки (рис. [23](#fig:023)).

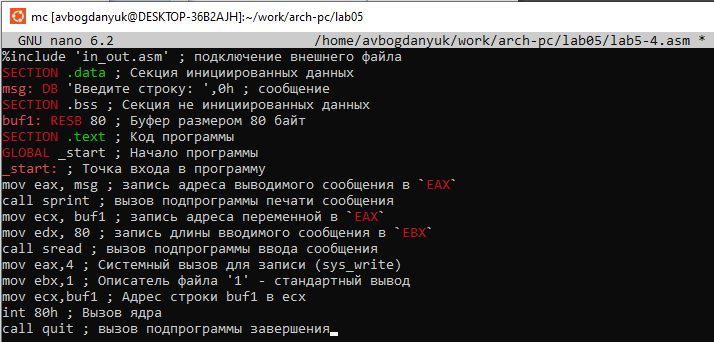


Figure 23: Изменение файла

Листинг программы:

"%include 'in\_out.asm' ; подключение внешнего файла  
SECTION .data ; Секция инициированных данных  
msg: DB 'Введите строку: ',0h ; сообщение  
SECTION .bss ; Секция не инициированных данных  
buf1: RESB 80 ; Буфер размером 80 байт  
SECTION .text ; Код программы  
GLOBAL \_start ; Начало программы  
\_start: ; Точка входа в программу  
mov eax, msg ; запись адреса выводимого сообщения в `EAX`  
call sprint ; вызов подпрограммы печати сообщения  
mov ecx, buf1 ; запись адреса переменной в `EAX`  
mov edx, 80 ; запись длины вводимого сообщения в `EBX`  
call sread ; вызов подпрограммы ввода сообщения  
mov eax,4 ; Системный вызов для записи (sys\_write)  
mov ebx,1 ; Описатель файла '1' - стандартный вывод  
mov ecx,buf1 ; Адрес строки buf1 в ecx  
int 80h ; Вызов ядра  
call quit ; вызов подпрограммы завершения"

Запускаю исполнительный файл (рис. [24](#fig:024)).

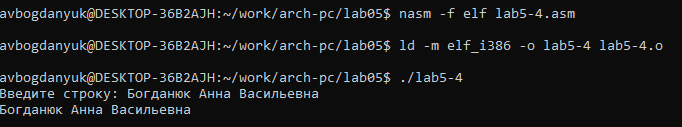


Figure 24: Вывод

# Выводы

Во время выполнения лабораторной работы приобрела практические навыки работы в Midnight Commander. Основание инструкций языка ассемблера mov и int.

# Список литературы