Отчет по лабораторной работе №5

Дисциплина: архитектура компьютера

Адмиральская Александра Андреевна

Содержание

# 1 Цель работы

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

# 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

Здесь описываются теоретические аспекты, связанные с выполнением работы.

Например, в табл. 1 приведено краткое описание стандартных каталогов Unix.

Таблица 1: Описание некоторых каталогов файловой системы GNU Linux

| Имя каталога | Описание каталога |
| --- | --- |
| / | Корневая директория, содержащая всю файловую |
| /bin | Основные системные утилиты, необходимые как в однопользовательском режиме, так и при обычной работе всем пользователям |
| /etc | Общесистемные конфигурационные файлы и файлы конфигурации установленных программ |
| /home | Содержит домашние директории пользователей, которые, в свою очередь, содержат персональные настройки и данные пользователя |
| /media | Точки монтирования для сменных носителей |
| /root | Домашняя директория пользователя root |
| /tmp | Временные файлы |
| /usr | Вторичная иерархия для данных пользователя |

Более подробно про Unix см. в [1–4].

# 4 Выполнение лабораторной работы

Для начала открывваем Midnight Commander с помощью команды mc (рис. 1).

Рис. 1: Midnight Commander

Рис. 1: Midnight Commander

Затем переходим в каталаог ~/work/arch-pc созданный при выполнении лабораторной работы №4 (рис. 2).

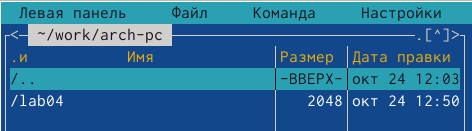


Рис. 2: Переход в каталаог ~/work/arch-pc

С помощью клавиши F7 создаем папку lab05 и переходим в этот каталог (рис. 3).

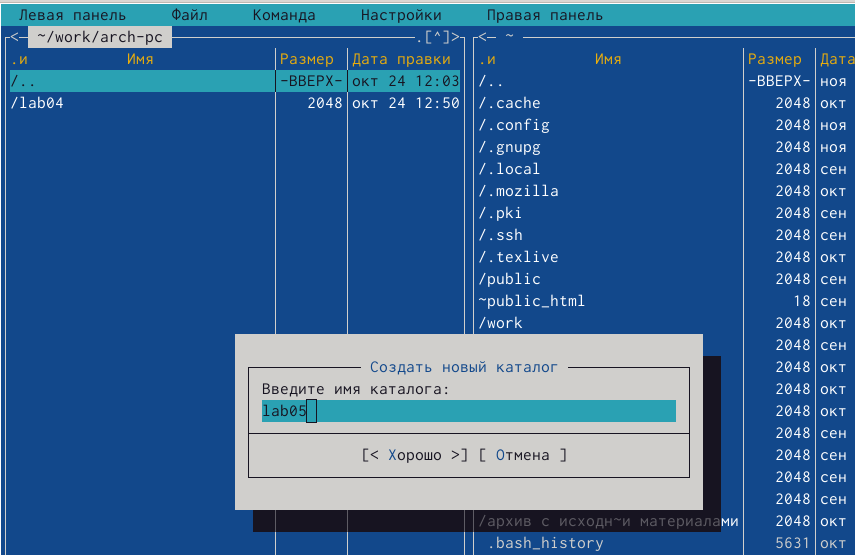


Рис. 3: Создание папки lab05

Пользуясь строкой ввода и командой touch создаем файл lab5-1.asm (рис. 4).

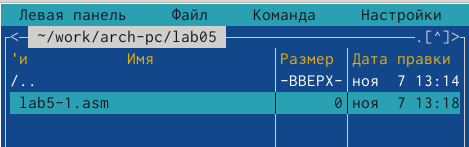


Рис. 4: Создание файла lab5-1.asm

С помощью функциональной клавиши F4 открываем созданный файл и вводим текст программы для запроса строки у пользователя (рис. 5).

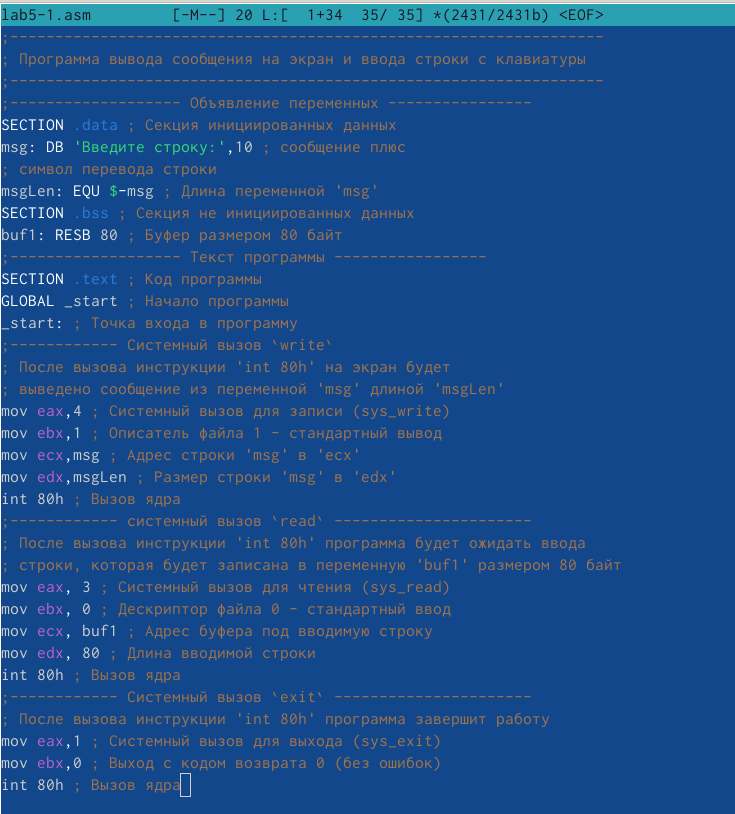


Рис. 5: Ввод текста программы

При помощи клавиши F3 открываем файл lab5-1.asm для просмотра и убеждаемся, что файл содержит текст программы (рис. 6).

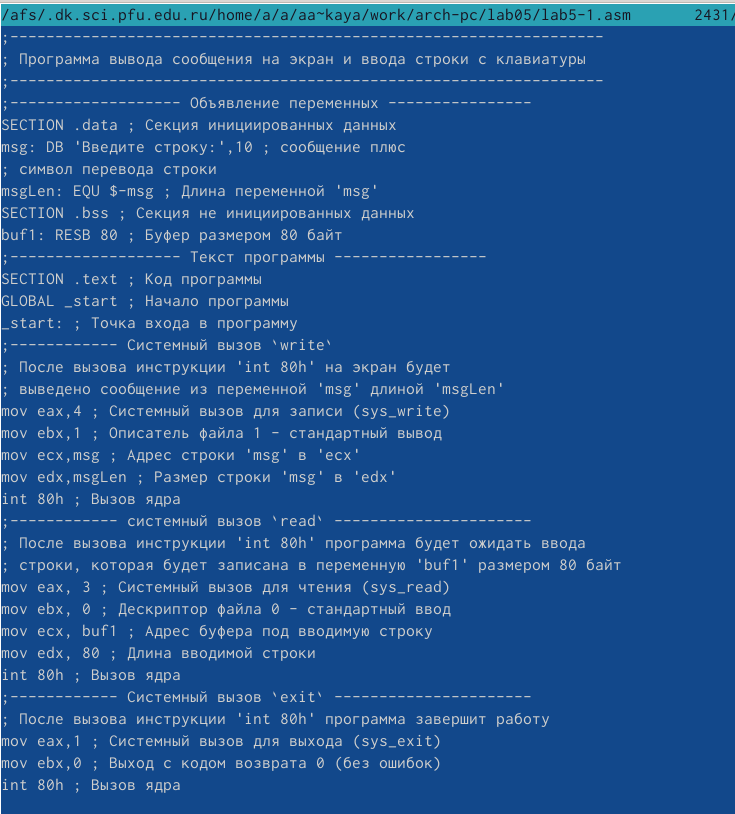


Рис. 6: Проверка

Транслируем текст программы файла в объектный файл командой nasm -f elf lab5-1.asm. Создался объектный файл lab5-1.o. Выполняем компоновку объектного файла с помощью команды ld -m elf\_i386 -o lab5-1 lab5-1.o. Создался исполняемый файл lab5-1 (рис. 7).

Рис. 7: Транслируем текст программы в объектный файл и выполняем его компоновку

Рис. 7: Транслируем текст программы в объектный файл и выполняем его компоновку

Запускаем получившийся исполняемый файл. Программа выводит строку ‘Введите строку:’ и мы вводим свое ФИО (рис. 8).

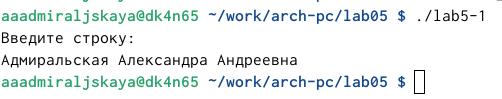


Рис. 8: Запуск получившегося исполняемого файла

Скачиваем файл in\_out.asm со страницы курса в ТУИС. Он сохранился в каталог “Загрузки” (рис. 9).

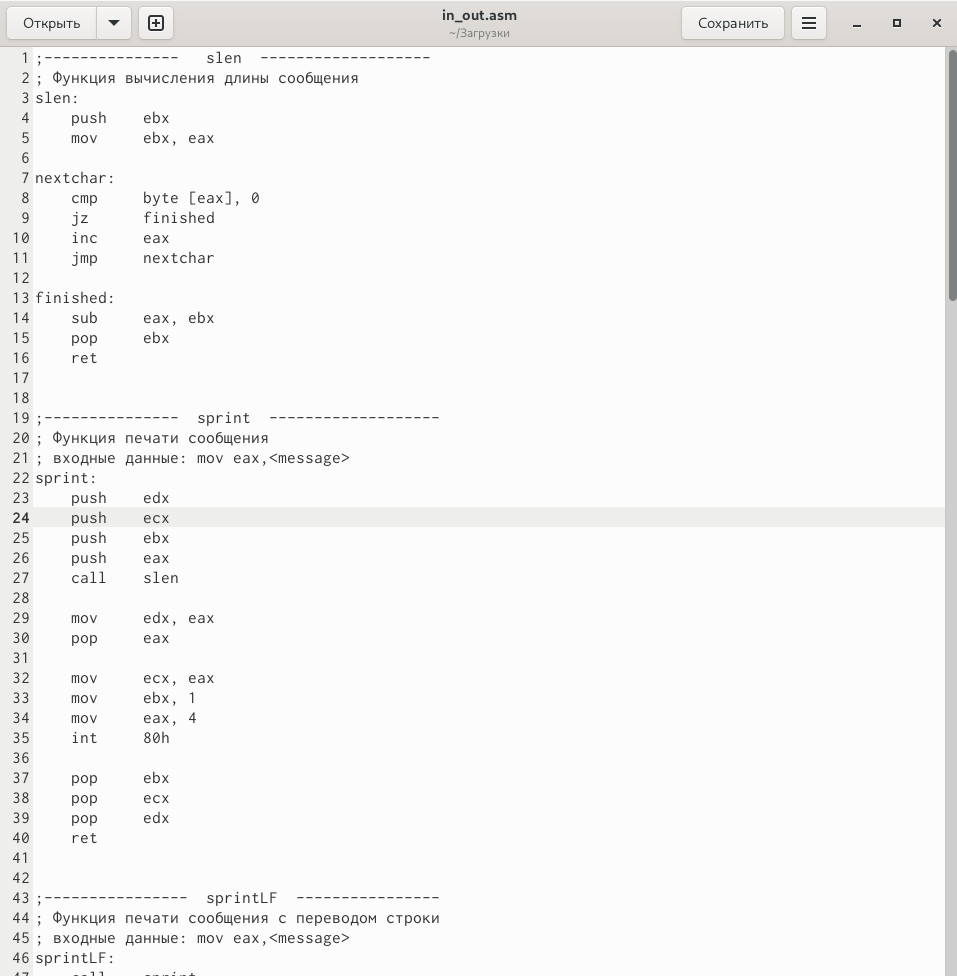


Рис. 9: Файл in\_out.asm

С помощью функциональной клавиши F5 копируем файл in\_out.asm из каталога Загрузки в созданный каталог lab05 (рис. 10).

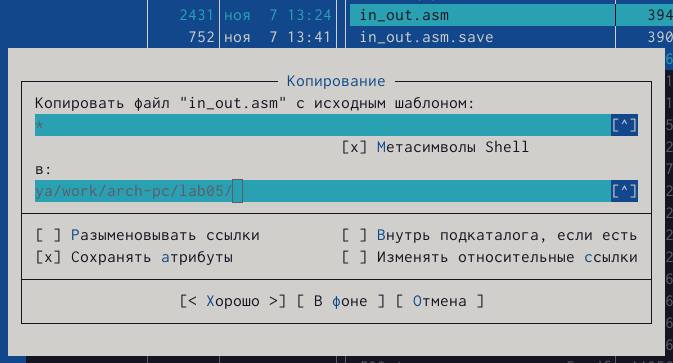


Рис. 10: Копирование файла в созданный каталог

С помощью функциональной клавиши F5 копируем файл lab5-1 в тот же каталог, но с другим именем, для этого в появившемся окне mc прописываем имя для копии файла (рис. 11).

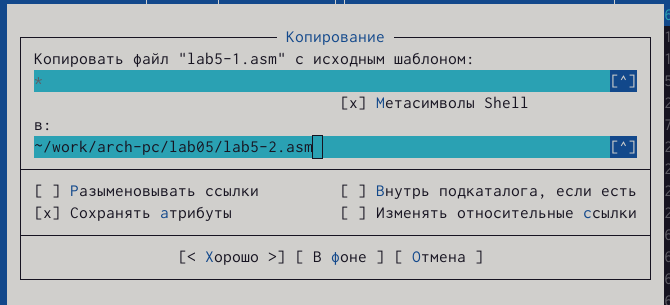


Рис. 11: Копирование файла с другим именем

Изменяем содержимое файла lab5-2.asm во встроенном редакторе nano, чтобы в программе использовались подпрограммы из внешнего файла in\_out.asm (рис. 12).

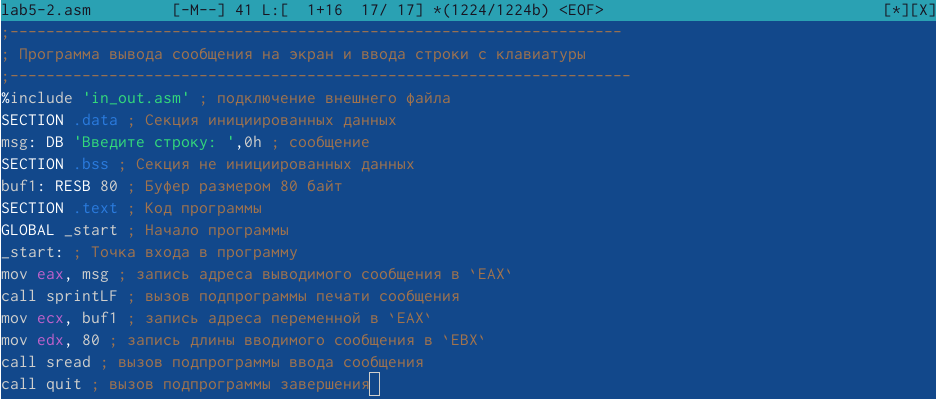


Рис. 12: Изменение содержимого файла

Транслируем текст программы файла в объектный файл командой nasm -f elf lab5-2.asm. Создался объектный файл lab5-2.o. Выполняем компоновку объектного файла с помощью команды ld -m elf\_i386 -o lab5-2 lab5-2.o Создался исполняемый файл lab5-2. Запускаем исполняемый файл (рис. 13).

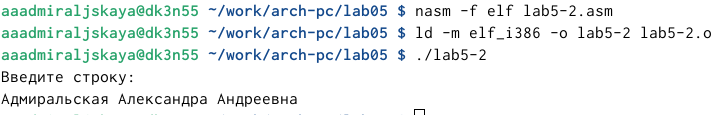


Рис. 13: Транслируем текст в объектный файл, выполняем компоновку объектного файла и запускаем исполняемый файл

Открываем файл lab5-2.asm для редактирования в nano функциональной клавишей F4. Изменяем в нем подпрограмму sprintLF на sprint (рис. 14).

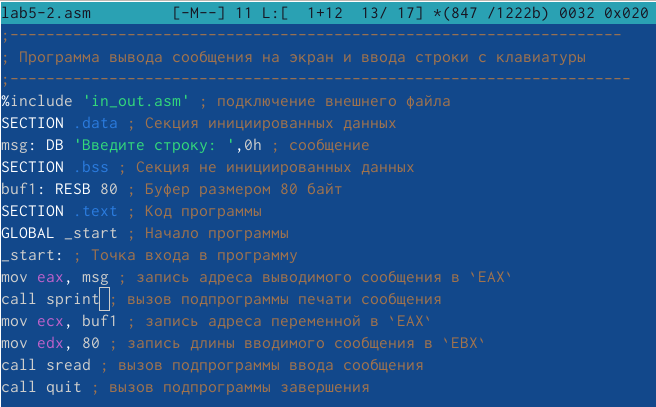


Рис. 14: Изменение подпрограммы sprintLF на sprint

Снова транслируем файл, выполняем компоновку созданного объектного файла, запускаем новый исполняемый файл (рис. 15). Разница между файлами lab5-2 и lab5-2-2 в том, что запуск первого исполняемого файла запрашивает ввод с новой строки, а программа, используемая при запуске второго файла, запрашивает ввод без переноса на новую строку. В этом и заключается различие между подпрограммами sprintLF и sprint.

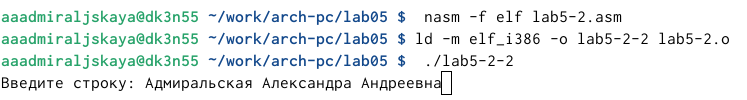


Рис. 15: Трансляция файла, его компоновка и запуск

Приступим к выполнению заданий для самостоятельной работы. 1) Создаем копию файла lab5-1.asm с именем lab5-1-1.asm (рис. 16).

Рис. 16: Копирование файла с новым именем

Рис. 16: Копирование файла с новым именем

С помощью функциональной клавиши F4 открываем созданный файл для редактирования. Изменяем программу так, чтобы кроме вывода приглашения и запроса ввода, она выводила вводимую пользователем строку (рис. 17).

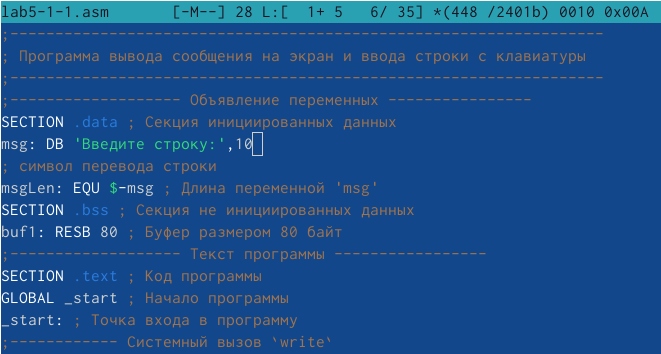


Рис. 17: Изменение программы

Код измененной программы из пункта 1):

;—————————————————————— ; Программа вывода сообщения на экран и ввода строки с клавиатуры ;—————————————————————— ;——————- Объявление переменных —————- SECTION .data ; Секция инициированных данных msg: DB ‘Введите строку:’,10 ; символ перевода строки msgLen: EQU $-msg ; Длина переменной ‘msg’ SECTION .bss ; Секция не инициированных данных buf1: RESB 80 ; Буфер размером 80 байт ;——————- Текст программы —————– 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

1. Создаем объектный файл lab5-1-1.o, отдаем его на обработку компоновщику, получаем исполняемый файл lab5-1-1, запускаем полученный исполняемый файл. Программа запрашивает ввод, вводим свои ФИО, далее программа выводит введенные мною данные (рис. 18).

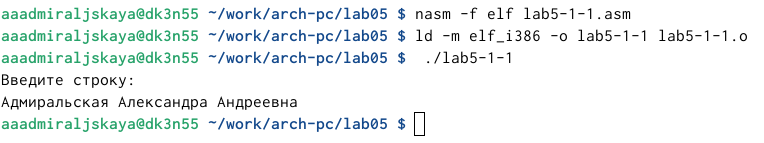


Рис. 18: Запуск полученного исполняемого файла

1. Создаем копию файла lab5-2.asm с именем lab5-2-1.asm с помощью функциональной клавиши F5 (рис. 19).

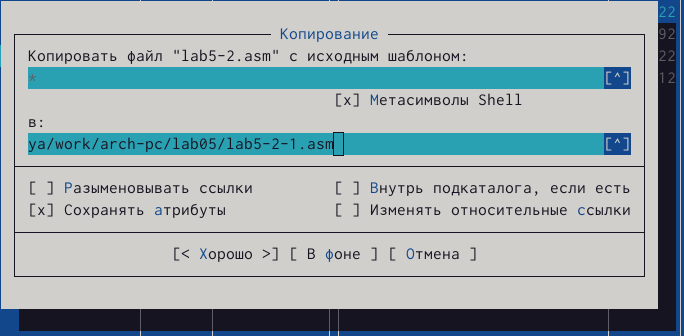


Рис. 19: Создание копии файла с новым именем

С помощью функциональной клавиши F4 открываем созданный файл для редактирования. Изменяем программу так, чтобы кроме вывода приглашения и запроса ввода, она выводила вводимую пользователем строку (рис. 20).

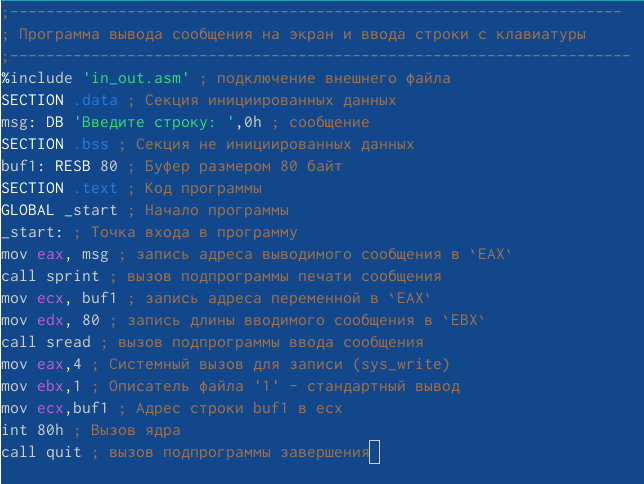


Рис. 20: изменение программы

Код программы из пункта 3):

;——————————————————————– ; Программа вывода сообщения на экран и ввода строки с клавиатуры ;——————————————————————— %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

1. Создаем объектный файл lab5-2-1.o, отдаем его на обработку компоновщику, получаем исполняемый файл lab5-2-1, запускаем полученный исполняемый файл. Программа запрашивает ввод без переноса на новую строку, ввожу свои ФИО, далее программа выводит введенные мною данные (рис. 21).

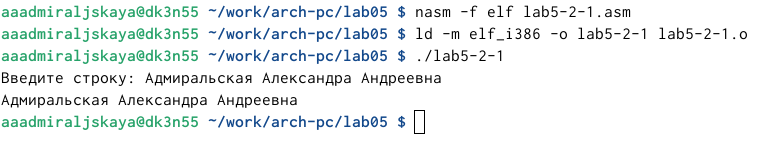


Рис. 21: Запуск полученного исполняемого файла

# 5 Выводы

При выполнении данной лабораторной работы я приобрела практические навыки работы в Midnight Commander, а также освоила инструкции языка ассемблера mov и int.

# Список литературы

1. Таненбаум Э., Бос Х. Современные операционные системы. 4-е изд. СПб.: Питер, 2015. 1120 с.

2. Robbins A. Bash Pocket Reference. O’Reilly Media, 2016. 156 с.

3. Zarrelli G. Mastering Bash. Packt Publishing, 2017. 502 с.

4. Newham C. [Learning the bash Shell: Unix Shell Programming](http://www.amazon.com/Learning-bash-Shell-Programming-Nutshell/dp/0596009658). O’Reilly Media, 2005. 354 с.