

Отчет по лабораторной работе №5

Дисциплина: архитектура компьютера

Комаров Владимир Артемович

Содержание

1	Цель работы	4
2	Задание	5
3	Теоретическое введение	6
4	Выполнение лабораторной работы	8
4.1	Основы работы с тс	8
4.2	Структура программы на языке ассемблера NASM	10
4.3	Подключение внешнего файла	12
4.4	Выполнение заданий для самостоятельной работы	15
5	Выводы	19
6	Список литературы	20

Список иллюстраций

4.1	Открытый mc	8
4.2	Перемещение между директориями	9
4.3	Создание каталога	9
4.4	Перемещение между директориями	9
4.5	Создание файла	9
4.6	Открытие файла для редактирования	10
4.7	Редактирование файла	11
4.8	Компиляция файла и передача на обработку компоновщику . . .	11
4.9	Исполнение файла	12
4.10	Скачанный файл	12
4.11	Копирование файла	12
4.12	Копирование файла	13
4.13	Редактирование файла	13
4.14	Исполнение файла	14
4.15	Отредактированный файл	14
4.16	Исполнение файла	14
4.17	Редактирование файла	15
4.18	Исполнение файла	15
4.19	Копирование файла	17
4.20	Редактирование файла	17
4.21	Исполнение файла	17

1 Цель работы

Целью данной лабораторной работы является приобретение практических навыков работы в Midnight Commander, освоение инструкций языка ассемблера `mov` и `int`.

2 Задание

1. Основы работы с тс
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 размером 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 (четырёх- 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 `int` предназначена 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

4.1 Основы работы с mc

Открываю Midnight Commander, введя в терминал mc (рис. 4.1).

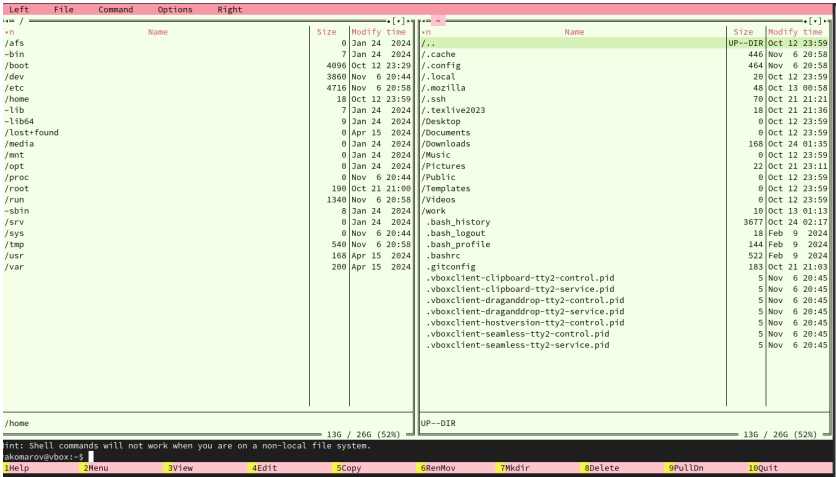


Рис. 4.1: Открытый mc

Перехожу в каталог ~/work/study/2023-2024/Архитектура Компьютера/arch-рс, используя файловый менеджер mc (рис. 4.2)

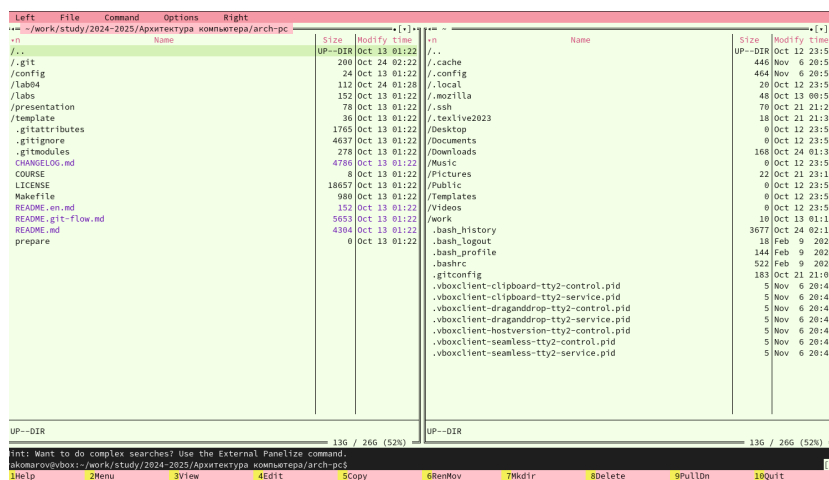


Рис. 4.2: Перемещение между директориями

С помощью функциональной клавиши F7 создаю каталог lab05 (рис. 4.3).

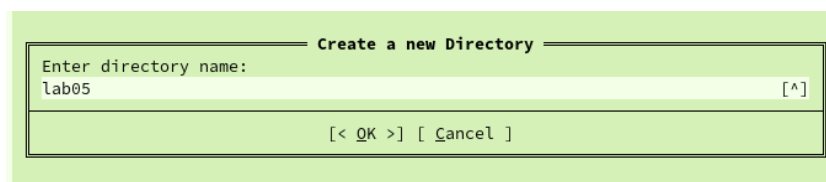


Рис. 4.3: Создание каталога

Перехожу в созданный каталог (рис. 4.4).

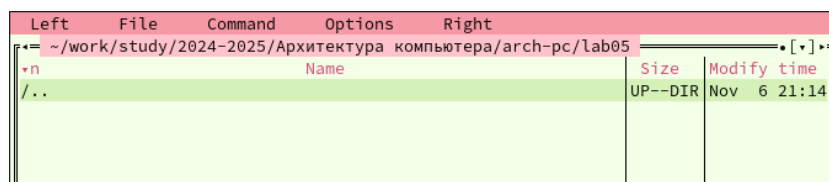


Рис. 4.4: Перемещение между директориями

В строке ввода прописываю команду touch lab05-1.asm, чтобы создать файл, в котором буду работать (рис. 4.5).

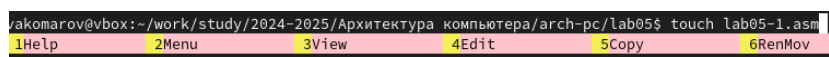


Рис. 4.5: Создание файла

4.2 Структура программы на языке ассемблера NASM

С помощью функциональной клавиши F4 открываю созданный файл для редактирования в редакторе (рис. 4.6).



Рис. 4.6: Открытие файла для редактирования

Ввожу в файл код программы для запроса строки у пользователя (рис. 4.7). Далее выхожу из файла (Ctrl+X), сохраняя изменения (Y, Enter).

```

lab05-1.asm      [-M--] 20 L:[ 1+ 2 3/ 33] *(179 /2153b) 1088 0x4
SECTION .data ; Секция иницированных данных
msg: DB 'Введите строку:',10 ; сообщение плюс
; символ перевода строки
msgLen: EQU $-msg ; Длина переменной 'msg'
SECTION .bss ; Секция не иницированных данных
buf1: RESB 80 ; Буфер размером 80 байт
62 Демидова А. В.
Архитектура ЭВМ
----- Текст программы -----
SECTION .text ; Код программы
GLOBAL _start ; Начало программы
_start: ; Точка входа в программу
; ----- Системный 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 ;Descriptor 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

```

Рис. 4.7: Редактирование файла

Транслирую текст программы файла в объектный файл командой `nasm -f elf lab05-1.asm`. Создался объектный файл `lab05-1.o`. Выполняю компоновку объектного файла с помощью команды `ld -m elf_i386 -o lab05-1 lab05-1.o` (рис. 4.8). Создался исполняемый файл `lab6-1`.

```

05$ nasm -f elf lab05-1.asm
05$ ld -m elf_i386 -o lab05-1 lab05-1.o

```

Рис. 4.8: Компиляция файла и передача на обработку компоновщику

Запускаю исполняемый файл. Программа выводит строку “Введите строку:” и ждет ввода с клавиатуры, я ввожу свои ФИО, на этом программа заканчивает свою работу (рис. 4.9).

```

vakomarov@vbox:~/work/study/2024-2025/Архитектура компьютера/arch-pc/lab05$ ./lab05-1
Введите строку:
Комаров Владимир Артемович

```

Рис. 4.9: Исполнение файла

4.3 Подключение внешнего файла

Скачиваю файл `in_out.asm` со страницы курса в ТУИС. Он сохранился в каталог “Загрузки” (рис. 4.10).



Name	Size	Modify time
UP--DIR		Nov 6 20:45
/pandoc-crossref-Linux	64	Oct 14 20:35
in_out.asm	3942	Nov 6 21:42
pandoc-crossref-Linux.tar.xz	7249656	Oct 14 20:34

Рис. 4.10: Скачанный файл

С помощью функциональной клавиши F5 копирую файл `in_out.asm` из каталога Загрузки в созданный каталог `lab05` (рис. 4.11).

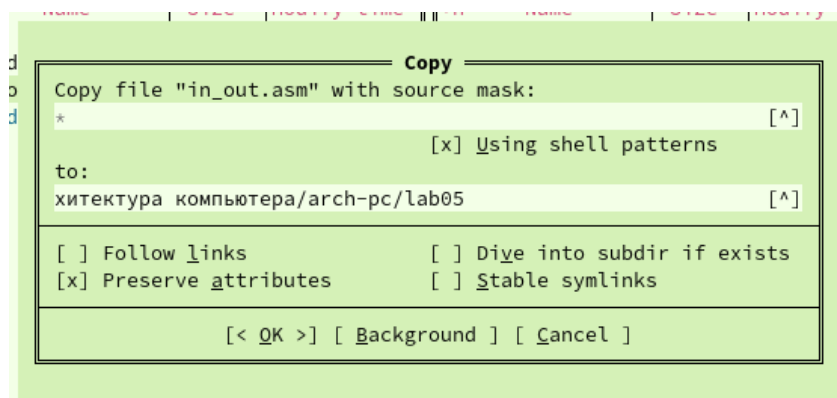


Рис. 4.11: Копирование файла

С помощью функциональной клавиши F5 копирую файл `lab05-1` в тот же каталог, но с другим именем, для этого в появившемся окне `mc` прописываю имя для копии файла (рис. 4.12).

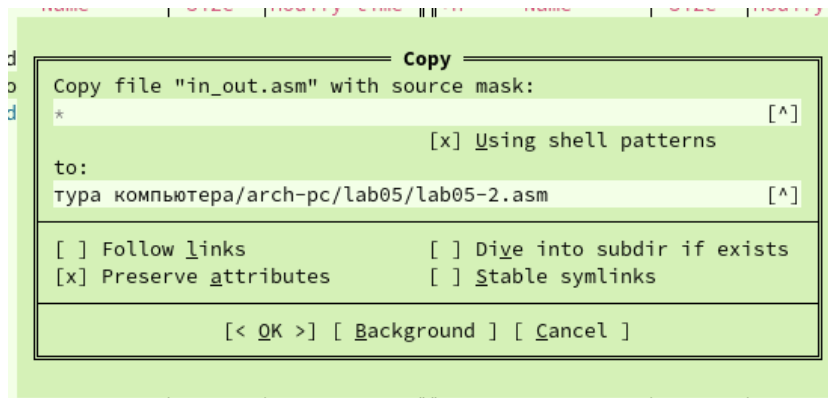


Рис. 4.12: Копирование файла

Изменяю содержимое файла lab05-2.asm во встроенном редакторе nano (рис. 4.13), чтобы в программе использовались подпрограммы из внешнего файла in_out.asm.

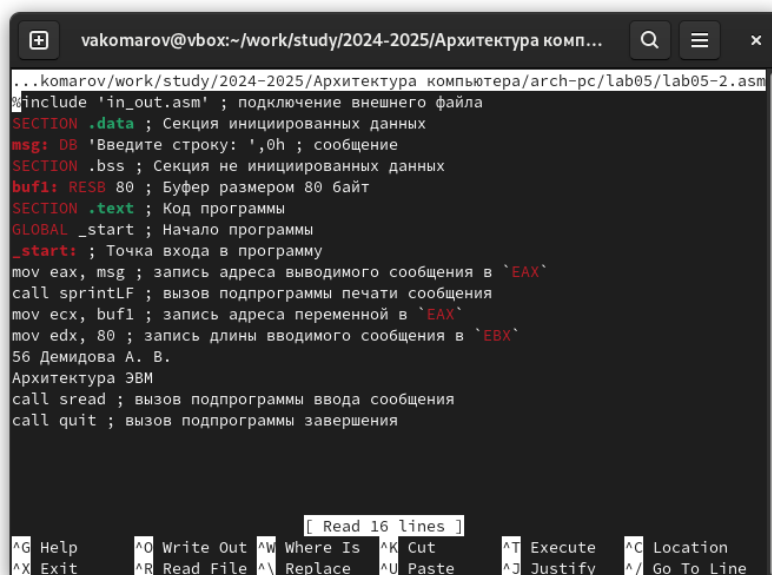


Рис. 4.13: Редактирование файла

Транслирую текст программы файла в объектный файл командой `nasm -f elf lab05-2.asm`. Создался объектный файл lab05-2.o. Выполняю компоновку объектного файла с помощью команды `ld -m elf_i386 -o lab05-2 lab05-2.o`. Создался исполняемый файл lab05-2. Запускаю исполняемый файл (рис. 4.14).

```

vakomarov@vbox:~/work/study/2024-2025/Архитектура компьютера/arch-pc/lab05$ nasm -f elf lab05-2.asm
vakomarov@vbox:~/work/study/2024-2025/Архитектура компьютера/arch-pc/lab05$ ld -m elf_i386 -o lab05-2 lab05-2.o
vakomarov@vbox:~/work/study/2024-2025/Архитектура компьютера/arch-pc/lab05$ ./lab05-2
Введите строку:
Комаров Владимир Артемович

```

Рис. 4.14: Исполнение файла

Открываю файл lab05-2.asm для редактирования в nano функциональной клавишей F4. Изменяю в нем подпрограмму sprintLF на sprint. Сохраняю изменения и открываю файл для просмотра, чтобы проверить сохранение действий (рис. 4.15).

```

#include 'in_out.asm' ; подключение внешнего файла
SECTION .data ; Секция иницированных данных
msg: DB 'Введите строку: ',0h ; сообщение
SECTION .bss ; Секция не иницированных 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

```

Рис. 4.15: Отредактированный файл

Снова транслирую файл, выполняю компоновку созданного объектного файла, запускаю новый исполняемый файл (рис. 4.16).

```

vakomarov@vbox:~/work/study/2024-2025/Архитектура компьютера/arch-pc/lab05$ nasm -f elf lab05-2.asm
vakomarov@vbox:~/work/study/2024-2025/Архитектура компьютера/arch-pc/lab05$ ld -m elf_i386 -o lab05-2 lab05-2.o
vakomarov@vbox:~/work/study/2024-2025/Архитектура компьютера/arch-pc/lab05$ ./lab05-2
Введите строку: Комаров Владимир Артемович

```

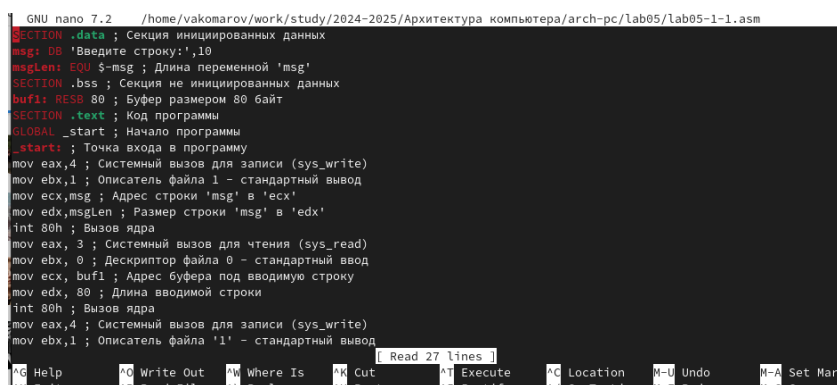
Рис. 4.16: Исполнение файла

Разница между первым исполняемым файлом lab05-2 и изменённым lab05-2 в том, что запуск первого запрашивает ввод с новой строки, а программа, которая выполняется при запуске второго, запрашивает ввод без переноса на новую строку, потому что в этом заключается различие между подпрограммами sprintLF и sprint.

4.4 Выполнение заданий для самостоятельной работы

1. Создаю копию файла lab6-1.asm с именем lab6-1-1.asm с помощью функциональной клавиши F5

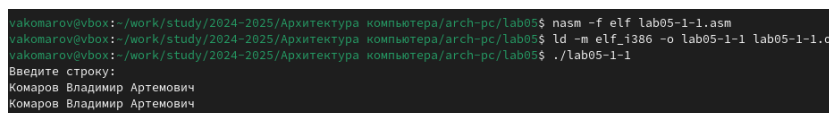
С помощью функциональной клавиши F4 открываю созданный файл для редактирования. Изменяю программу так, чтобы кроме вывода приглашения и запроса ввода, она выводила вводимую пользователем строку (рис. 4.17).



```
GNU nano 7.2 /home/vakomarov/work/study/2024-2025/Архитектура компьютера/arch-pc/lab05/lab05-1-1.asm
SECTION .data ; Секция иницированных данных
msg: DB 'Введите строку:',10
msgLen: EQU $-msg ; Длина переменной 'msg'
SECTION .bss ; Секция не иницированных 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
```

Рис. 4.17: Редактирование файла

2. Создаю объектный файл lab05-1-1.o, отдаю его на обработку компоновщику, получаю исполняемый файл lab05-1-1, запускаю полученный исполняемый файл. Программа запрашивает ввод, ввожу свои ФИО, далее программа выводит введенные мною данные (рис. 4.18).



```
vakomarov@vbox: /work/study/2024-2025/Архитектура компьютера/arch-pc/lab05$ nasm -f elf lab05-1-1.asm
vakomarov@vbox: /work/study/2024-2025/Архитектура компьютера/arch-pc/lab05$ ld -m elf_i386 -o lab05-1-1 lab05-1-1.o
vakomarov@vbox: /work/study/2024-2025/Архитектура компьютера/arch-pc/lab05$ ./lab05-1-1
Введите строку:
Комаров Владимир Артемович
Комаров Владимир Артемович
```

Рис. 4.18: Исполнение файла

Код программы из пункта 1:

```
SECTION .data ; Секция иницированных 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

```

3. Создаю копию файла lab05-2.asm с именем lab05-2-1.asm с помощью функциональной клавиши F5 (рис. 4.19).

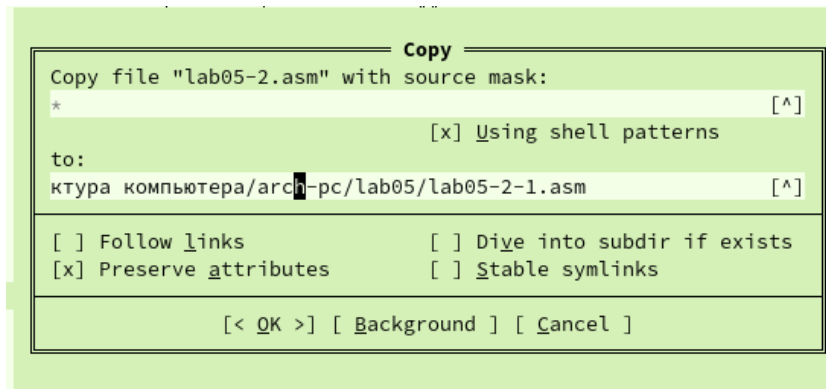


Рис. 4.19: Копирование файла

С помощью функциональной клавиши F4 открываю созданный файл для редактирования. Изменяю программу так, чтобы кроме вывода приглашения и запроса ввода, она выводила вводимую пользователем строку (рис. 4.20).

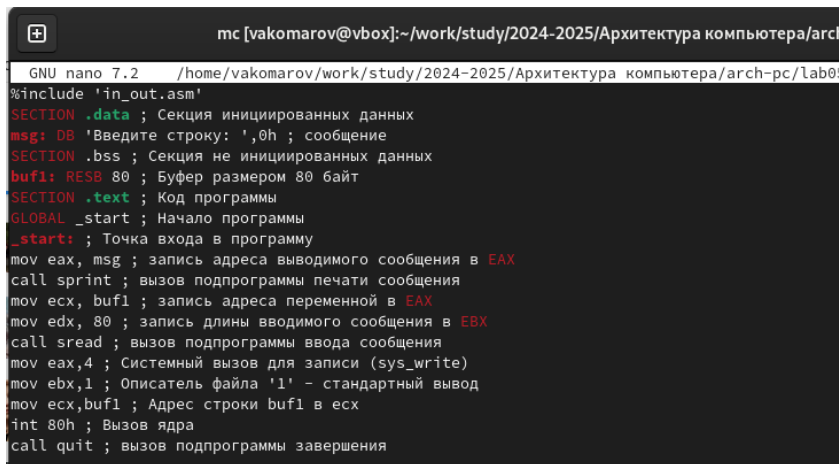


Рис. 4.20: Редактирование файла

4. Создаю объектный файл lab05-2-1.o, отдаю его на обработку компоновщику, получаю исполняемый файл lab05-2-1, запускаю полученный исполняемый файл. Программа запрашивает ввод без переноса на новую строку, ввожу свои ФИО, далее программа выводит введенные мною данные (рис. 4.21).

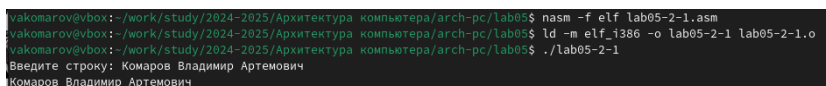


Рис. 4.21: Исполнение файла

Код программы из пункта 3:

```
%include 'in_out.asm'

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
```

5 Выводы

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

6 Список литературы

1. Лабораторная работа №5