

Отчёта по лабораторной работе №5

Дисциплина: Архитектура компьютера

Гафоров Нурмухаммад Вомикович

Содержание

1	Цель работы	5
2	Задание	6
3	Теоретическое введение	7
4	Выполнение лабораторной работы	9
4.1	Структура программы на языке ассемблера NASM	11
4.2	Подключение внешнего файла	14
5	Выводы	19
	Список литературы	20

Список иллюстраций

4.1	Открытый mc	9
4.2	Перемещение между директориями	10
4.3	Создание каталога	10
4.4	Перемещение между директориями	11
4.5	Создание файла	11
4.6	Открытие файла для редактирования	12
4.7	Редактирование файла	12
4.8	Открытие файла для просмотра	13
4.9	Компиляция файла и передача на обработку компоновщику	13
4.10	Исполнение файла	13
4.11	Скачанный файл	14
4.12	Копирование файла	15
4.13	Редактирование файла	15
4.14	Исполнение файла	16
4.15	Отредактированный файл	17
4.16	Исполнение файла	17
4.17	Исполнение файла	18

Список таблиц

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

##Основы работы с mc

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

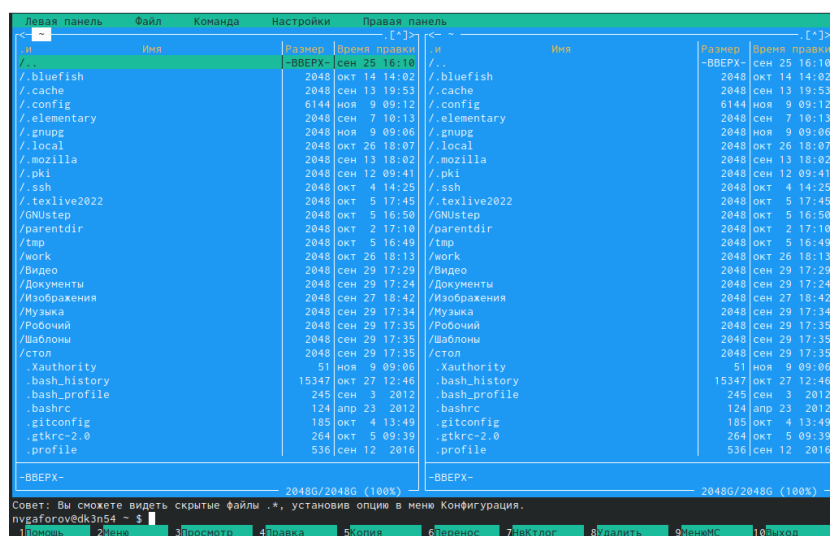


Рис. 4.1: Открытый mc

Перехожу в каталог ~/work/study/2022-2023/Архитектура Компьютера/arch-рс, используя файловый менеджер mc (рис. [4.2])

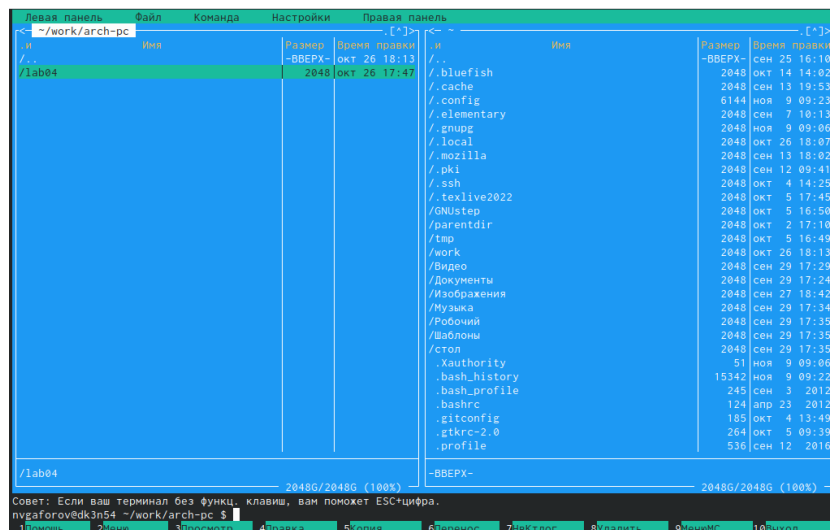


Рис. 4.2: Перемещение между директориями

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

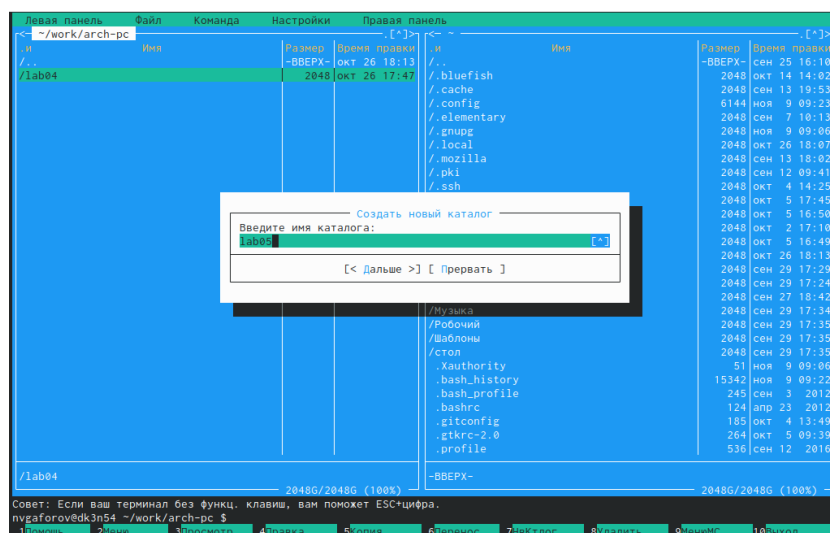


Рис. 4.3: Создание каталога

Переходу в созданный каталог (рис. [4.4]).

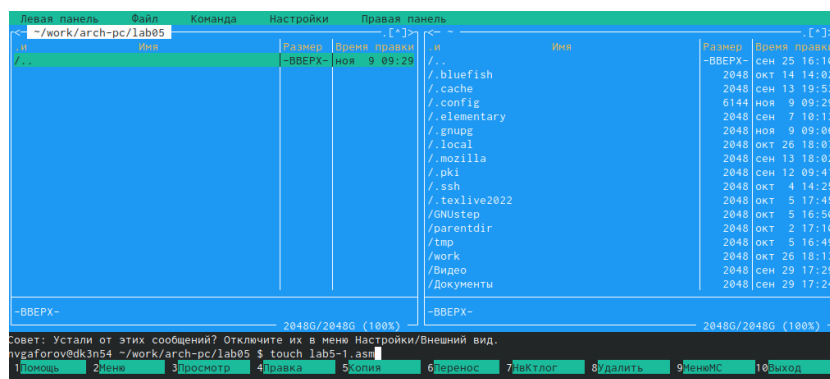


Рис. 4.4: Перемещение между директориями

В строке ввода прописываю команду `touch lab5-1.asm`, чтобы создать файл, в котором буду работать (рис. [4.5]).

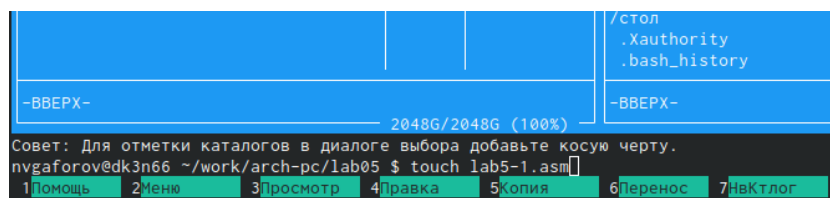


Рис. 4.5: Создание файла

4.1 Структура программы на языке ассемблера NASM

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

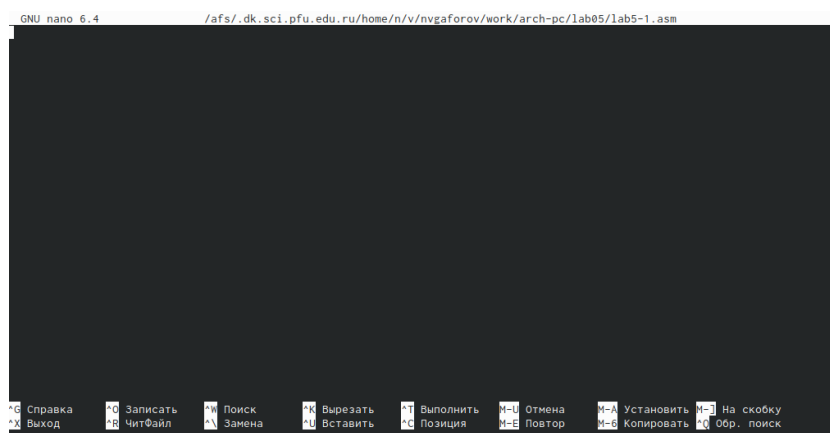


Рис. 4.6: Открытие файла для редактирования

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

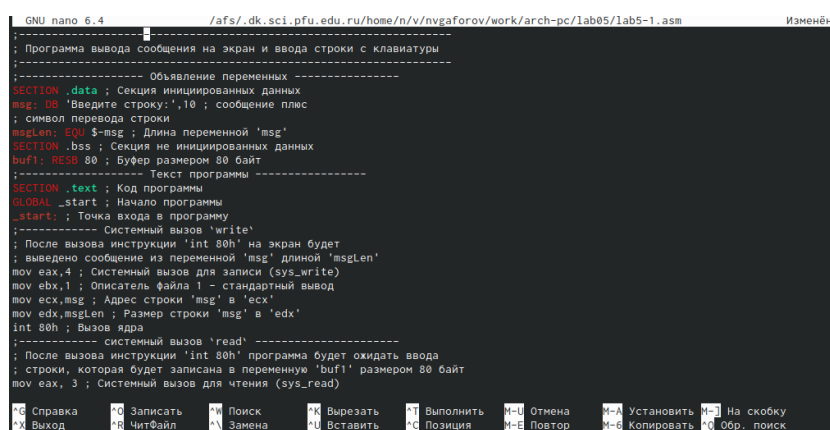


Рис. 4.7: Редактирование файла

С помощью функциональной клавиши F3 открываю файл для просмотра, чтобы проверить, содержит ли файл текст программы (рис. [4.8])

```

/afs/dk.sci.pfu.edu.ru/home/n/v/vgaforov/work/arch-pc/lab05/lab5-1.asm 1986/2433 81
;-----
; Программа вывода сообщения на экран и ввода строки с клавиатуры
;-----
;----- Объявление переменных -----
SECTION .data ; Секция инициализированных данных
msg: DB "Введите строку:",10 ; сообщение плюс
; символ перевода строки
msgLen: EQU $-msg ; Длина переменной 'msg'
SECTION .bss ; Секция не инициализированных данных
buf1: RESB 80 ; Буфер размером 80 байт
;----- 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 ; Описание 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

```

Рис. 4.8: Открытие файла для просмотра

Транслирую текст программы файла в объектный файл командой `nasm -f elf lab5-1.asm`. Создался объектный файл `lab5-1.o`. Выполняю компоновку объектного файла с помощью команды `ld -m elf_i386 -o lab5-1 lab5-1.o` (рис. [4.9]). Создался исполняемый файл `lab5-1`

Компиляция файла и передача на обработку компоновщику

Рис. 4.9: 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. [4.10]).

```

lab05: mc — Konsole
Файл  Правка  Вид  Закладки  Модули  Настройка  Справка
[Новая вкладка] [Разделить окно] [Копировать] [Вставить] [Найти]
nvgaforov@dk3n54 ~ $ mc
nvgaforov@dk3n54 ~/work/arch-pc/lab05 $ touch lab5-1.asm
nvgaforov@dk3n54 ~ $ mc
nvgaforov@dk3n54 ~/work/arch-pc/lab05 $ nasm -f elf lab5-1.asm
nvgaforov@dk3n54 ~/work/arch-pc/lab05 $ ld -m elf_i386 -o lab5-1 lab5-1.o
nvgaforov@dk3n54 ~/work/arch-pc/lab05 $ ./lab5-1
Введите строку:
Гафоров Нурмухаммад Вомирович
nvgaforov@dk3n54 ~/work/arch-pc/lab05 $

```

Рис. 4.10: Исполнение файла

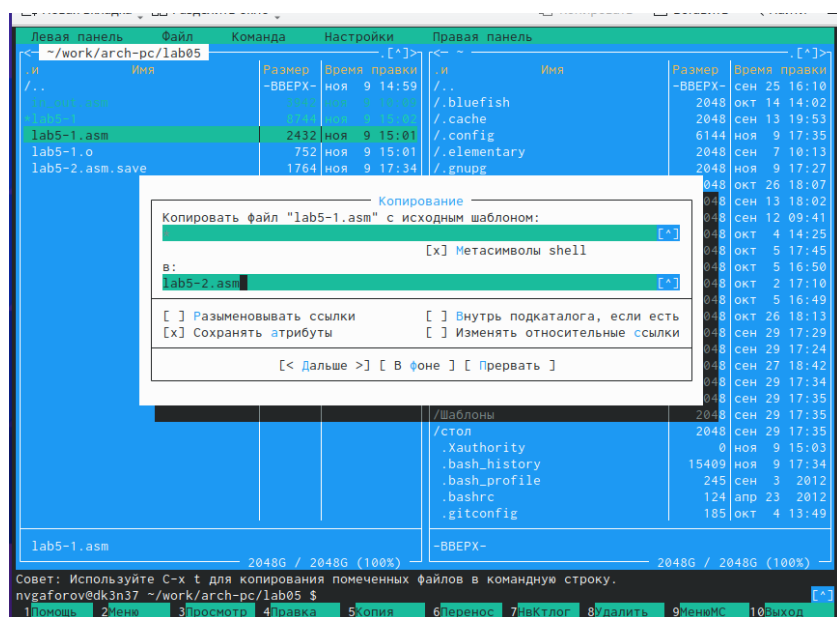


Рис. 4.12: Копирование файла

Изменяю содержимое файла lab5-2.asm во встроенном редакторе nano (рис. [4.13]), чтобы в программе использовались подпрограммы из внешнего файла in_out.asm.

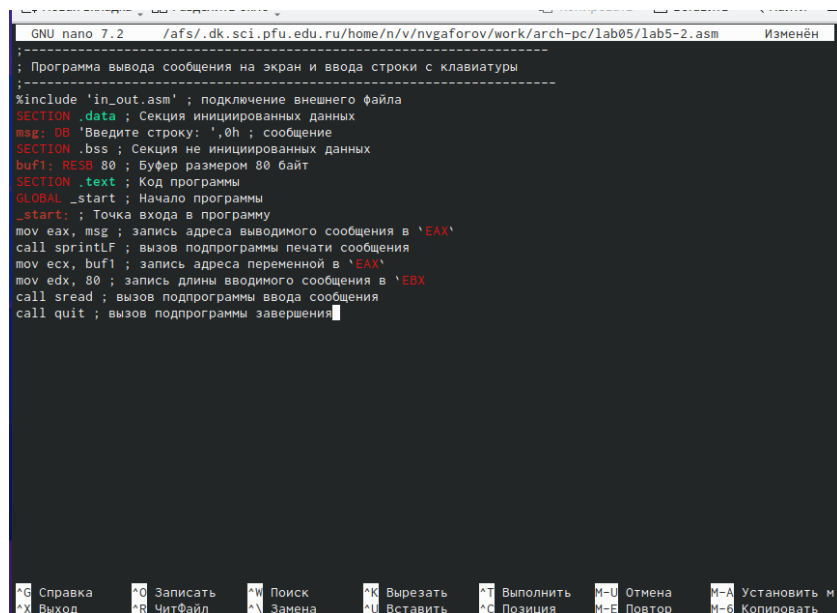
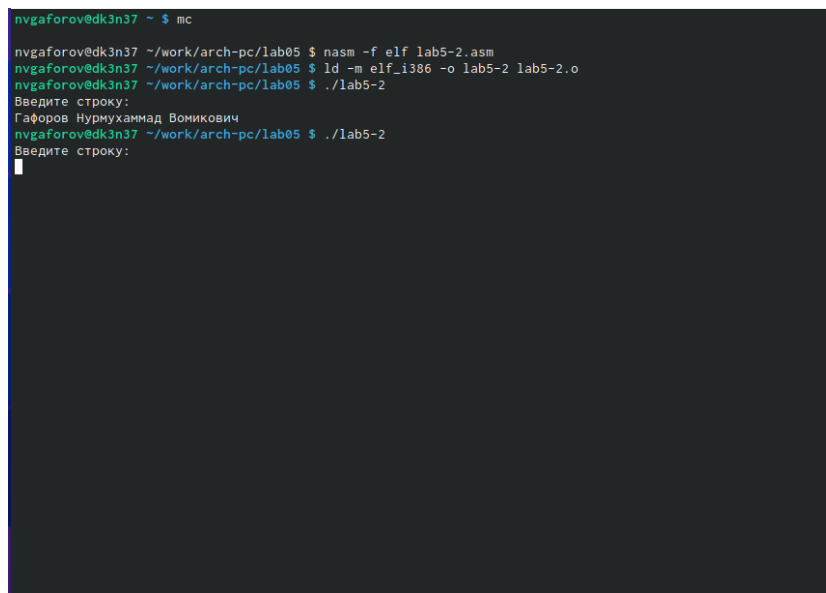


Рис. 4.13: Редактирование файла

Транслирую текст программы файла в объектный файл командой `nasm -f elf lab5-2.asm`. Создался объектный файл `lab5-2.o`. Выполняю компоновку объектного файла с помощью команды `ld -m elf_i386 -o lab5-2 lab5-2.o`. Создался исполняемый файл `lab5-2`. Запускаю исполняемый файл (рис. [4.14]).



```
nvgafarov@dk3n37 ~ $ mc
nvgafarov@dk3n37 ~/work/arch-pc/lab05 $ nasm -f elf lab5-2.asm
nvgafarov@dk3n37 ~/work/arch-pc/lab05 $ ld -m elf_i386 -o lab5-2 lab5-2.o
nvgafarov@dk3n37 ~/work/arch-pc/lab05 $ ./lab5-2
Введите строку:
Гафоров Нурмухаммад Воинович
nvgafarov@dk3n37 ~/work/arch-pc/lab05 $ ./lab5-2
Введите строку:
█
```

Рис. 4.14: Исполнение файла

Открываю файл `lab5-2.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]).


```

lab5-2.asm      [-M--]  0 L: [ 1+17  18/ 18] *(1222/1222b) <EOF>      [*][X]
;~~~~~
; Программа вывода сообщения на экран и ввода строки с клавиатуры
;~~~~~
include "io.inc" ; подключение внешнего файла
SECTION .data ; Секция инициализированных данных
msg: DB "Введите строку: ", 0h ; сообщение
SECTION .bss ; Секция не инициализированных 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 'ECX'
mov edx, 80 ; запись длины вводимого сообщения в 'EDX'
call sread ; вызов подпрограммы ввода сообщения
call quit ; вызов подпрограммы завершения
;~~~~~

```

Рис. 4.15: Отредактированный файл

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

```

nvgaforov@dk3n37 ~/work/arch-pc/lab05 $ nasm -f elf lab5-2-2.asm
nvgaforov@dk3n37 ~/work/arch-pc/lab05 $ ld -m elf_i386 -o lab5-2-2 lab5-2-2.o
nvgaforov@dk3n37 ~/work/arch-pc/lab05 $ ./lab5-2-2
Введите строку: Гафоров Нурмухаммад Вомирович

```

Рис. 4.16: Исполнение файла

Разница между первым исполняемым файлом lab5-2 и вторым lab5-2-2 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`.

Разница между `sprintLF` и `sprint` (рис. [4.17]).

```
nvgaforov@dk3n37 ~ $ mc
nvgaforov@dk3n37 ~/work/arch-pc/lab05 $ nasm -f elf lab5-2.asm
nvgaforov@dk3n37 ~/work/arch-pc/lab05 $ ld -m elf_i386 -o lab5-2 lab5-2.o
nvgaforov@dk3n37 ~/work/arch-pc/lab05 $ ./lab5-2
Введите строку:
Гафоров Нурмухаммад Вомикович
nvgaforov@dk3n37 ~/work/arch-pc/lab05 $ nasm -f elf lab5-2-2.asm
nvgaforov@dk3n37 ~/work/arch-pc/lab05 $ ld -m elf_i386 -o lab5-2-2 lab5-2-2.o
nvgaforov@dk3n37 ~/work/arch-pc/lab05 $ ./lab5-2-2
Введите строку: Гафоров Нурмухаммад Вомикович
```

Рис. 4.17: Исполнение 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

https://esystem.rudn.ru/pluginfile.php/1584633/mod_resource/content/1/%D0%9B%D0%B0%D0%