

РОССИЙСКИЙ УНИВЕРСИТЕТ ДРУЖБЫ НАРОДОВ

**Факультет физико-математических и естественных наук Кафедра
прикладной информатики и теории вероятностей**

ОТЧЕТ

по лабораторной работе № 6

дисциплина: Архитектура компьютера

Студент: Клименко Кирилл Русланович

Группа: НММбд-02-24

МОСКВА

2025г.

Содержание

1 Цель работы	4
2 Задание	5
3 Теоретическое введение	6
4 Выполнение лабораторной работы	8
4.1 Основы работы с mc	8
4.2 Структура программы на языке ассемблера NASM	10
4.3 Подключение внешнего файла.....	12
4.4 Выполнение заданий для самостоятельной работы.....	14
5 Выводы	16
6 Список литературы	17

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	10
4.8 Открытие файла для просмотра.....	11
4.9 Компиляция файла и передача на обработку компоновщику.....	11
4.10 Исполнение файла.....	11
4.11 Скачанный файл	12
4.12 Копирование файла.....	12
4.13 Редактирование файла	13
4.14 Исполнение файла.....	13
4.15 Отредактированный файл	14
4.16 Исполнение файла.....	14
4.17 Копирование файла.....	15
4.18 Редактирование файла	15
4.19 Исполнение файла.....	16

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 mc
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 инициализированных (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 инициализированных 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 (четвертое 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).

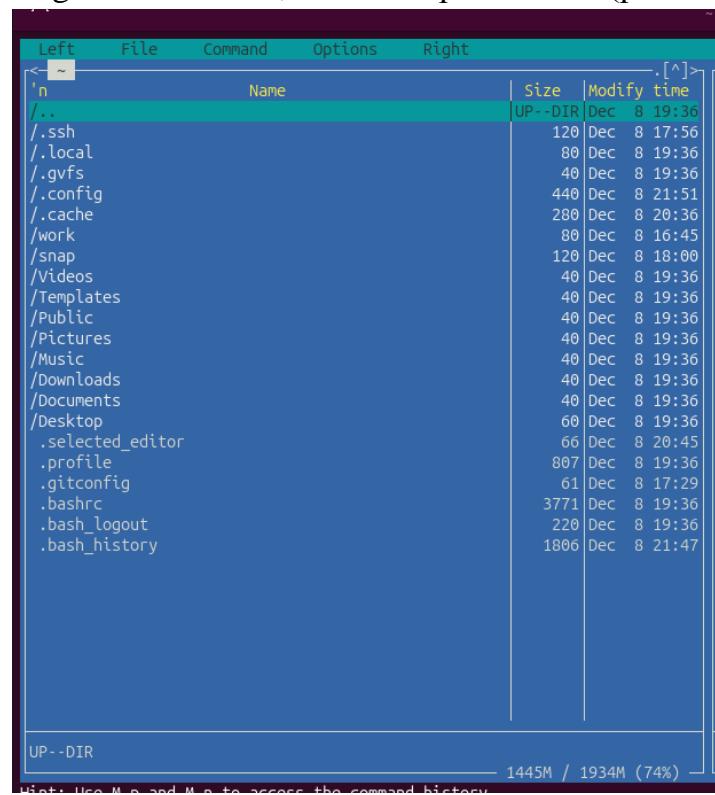


Рис. 4.1: Открытый mc

Перехожу в каталог ~/work/study/arhPC/labs/lab06, используя файловый менеджер mc (рис. 4.2)

Left	File	Command	Options	Right
<- ~/work/study/arhPC/labs				[^]>
'n	Name		Size	Modify time
/..		UP--DIR	Dec 8 17:42	/
/lab11		80	Dec 8 16:40	/
/lab10		80	Dec 8 16:40	/
/lab09		80	Dec 8 16:40	/
/lab08		80	Dec 8 16:40	/
/lab07		80	Dec 8 16:40	/
/lab06		100	Dec 8 22:38	/
/lab05		240	Dec 8 22:08	/
/lab04		120	Dec 8 21:26	/
/lab03		80	Dec 8 16:40	/
/lab02		80	Dec 8 16:40	/
/lab01		40	Dec 8 16:40	/
README.ru.md		19	Dec 8 16:40	
README.md				

Рис. 4.2: Перемещение между директориями

С помощью mkdir -p создаю каталог lab06

```
ubuntu@ubuntu:~/work/study/arhPC$ mkdir -p ~/work/study/arhPC/labs/lab06
```

Рис. 4.3: Создание каталога

Переходу в созданный каталог (рис. 4.4).

Left	File	Command	Options	Right
<- ~/work/study/arhPC/labs/lab06				[^]>
.n	Name		Size	Modify time
/..		UP--DIR	Dec 8 16:40	/
/presentation		220	Dec 8 16:40	/
/report		240	Dec 8 16:40	/
in_out.asm		3942	Dec 8 22:34	

Рис. 4.4: Перемещение между директориями

В строке ввода прописываю команду touch lab6-1.asm, чтобы создать файл, в котором буду работать (рис. 4.5).

```
Hint: Tab changes your current panel.  
ubuntu@ubuntu:~/work/study/arhPC/labs/lab06$
```

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 nano (рис. 4.6).

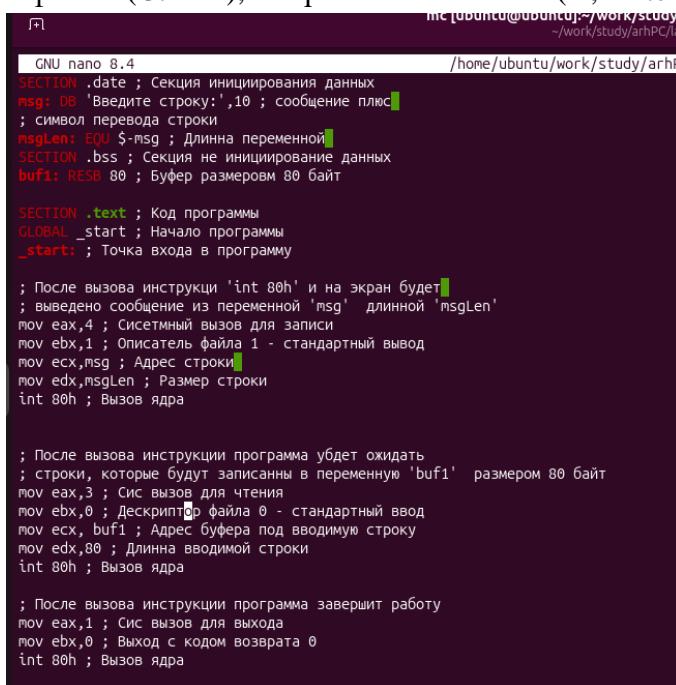


The screenshot shows a terminal window with the command 'mc [ubuntu@ubuntu:~/work/study/arhPC/labs/lab06 — mc ~/work/study/arhPC/labs/lab06' at the top. Below it, the nano editor interface is visible, showing the path '/home/ubuntu/work/study/arhPC/labs/lab06/lab6-1.as' and the text 'GNU nano 8.4'.

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).



The screenshot shows the nano editor with the following assembly code:

```
SECTION .data ; Секция инициализации данных
msg: DB 'Введите строку: ',10 ; сообщение плюс
; символ перевода строки
msgLen: EQU $-msg ; Длина переменной
SECTION .bss ; Секция не инициализированы данные
buf1: RESB 80 ; Буфер размером 80 байт

SECTION .text ; Код программы
GLOBAL _start ; Начало программы
_start: ; Точка входа в программу

; После вызова инструкции 'int 80h' и на экран будет
; выведено сообщение из переменной 'msg' длиной 'msgLen'
mov eax,4 ; Системный вызов для записи
mov ebx,1 ; Описатель файла 1 - стандартный вывод
mov ecx,msg ; Адрес строки
mov edx,buf1 ; Адрес буфера под вводимую строку
int 80h ; Вызов ядра

; После вызова инструкции программа будет ожидать
; строки, которые будут записаны в переменную 'buf1' размером 80 байт
mov eax,3 ; Сис вызов для чтения
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

; После вызова инструкции программа завершит работу
mov eax,1 ; Сис вызов для выхода
mov ebx,0 ; Выход с кодом возврата 0
int 80h ; Вызов ядра
```

Рис. 4.7: Редактирование файла

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

```

/home/ubuntu/work/study/arhPC/labs/lab06/lab0-1.asm
SECTION .data ; Секция инициализации данных
msg: DB 'Введите строку:',10 ; сообщение плюс
; символ перевода строки
msgLen: EQU $-msg ; Длина переменной
SECTION .bss ; Секция не инициализироване данных
buf1: RESB 80 ; Буфер размером 80 байт

SECTION .text ; Код программы
GLOBAL _start ; Начало программы
_start: ; Точка входа в программу

; После вызова инструкции 'int 80h' и на экран будет
; выведено сообщение из переменной 'msg' длиной 'msgLen'
mov eax,4 ; Системный вызов для записи
mov ebx,1 ; Описатель файла 1 - стандартный вывод
mov ecx,msg ; Адрес строки
mov edx,msgLen ; Размер строки
int 80h ; Вызов ядра

; После вызова инструкции программа убдет ожидать
; строки, которые будут записаны в переменную 'buf1' размером 80 байт
mov eax,3 ; СИС вызов для чтения
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

; После вызова инструкции программа завершит работу
mov eax,1 ; СИС вызов для выхода
mov ebx,0 ; Выход с кодом возврата 0
int 80h ; Вызов ядра

```

Рис. 4.8: Открытие файла для просмотра

Транслирую текст программы файла в объектный файл командой nasm -f elf lab6-1.asm. Создался объектный файл lab6-1.o. Выполняю компоновку объектного файла с помощью команды ld -m elf_i386 -o lab6-1 lab6-1.o (рис. 4.9). Создался исполняемый файл lab6-1.

```

ubuntu@ubuntu:~/work/study/arhPC/labs/lab06$ nasm -f elf lab6-1.asm
ubuntu@ubuntu:~/work/study/arhPC/labs/lab06$ ld -m elf_i386 -o lab6-1 lab6-1.o

```

Рис. 4.9: Компиляция файла и передача на обработку компоновщику

Запускаю исполняемый файл. Программа выводит строку “Ведите строку:” и ждет ввода с клавиатуры, я ввожу свои ФИО, на этом программа заканчивает свою работу (рис. 4.10).

```

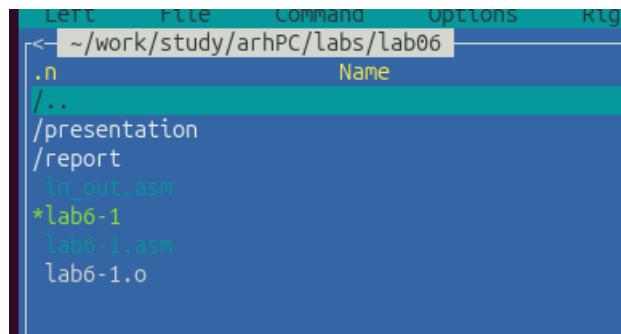
ubuntu@ubuntu:~/work/study/arhPC/labs/lab06$ ./lab6-1
Введите строку:
Klimenko Kirill Ruslanovich

```

Рис. 4.10: Исполнение файла

4.3 Подключение внешнего файла

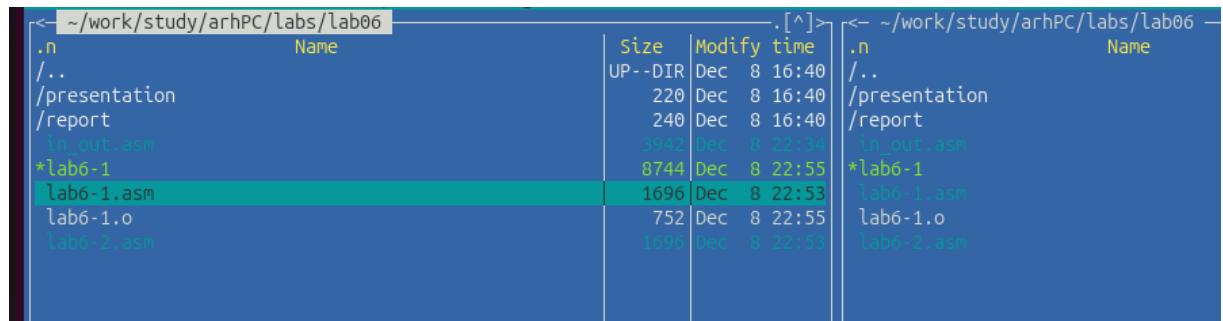
Я добавил файл in_out.asm, сохранив его на github. Он сохранился в каталог lab06



```
lett   file      command    options      rig
<- ~/work/study/arhPC/labs/lab06
.n
Name
/..
/presentation
/report
in_out.asm
*lab6-1
lab6-1.asm
lab6-1.o
```

Рис. 4.11: Скачанный файл

С помощью функциональной клавиши F5 копирую файл lab6-1 в тот же каталог, но с другим именем. (рис. 4.12).



Name	Size	Modify time
UP--DIR	220	Dec 8 16:40
	240	Dec 8 16:40
	3942	Dec 8 22:34
	8744	Dec 8 22:55
	1696	Dec 8 22:53
	752	Dec 8 22:55
	1696	Dec 8 22:53

Рис. 4.12: Копирование файла

Изменяю содержимое файла lab6-2.asm во встроенным редакторе nano (рис. 4.13), чтобы в программе использовались подпрограммы из внешнего файла in_out.asm.

```
ubuntu@ubuntu:~/work/study/arthPC/labs/lab06$ nano lab6-2.asm
GNU nano 8.4
%include 'in_out.asm' ; Подключение внешнего файла
SECTION .data ; Секция инициализации данных
msg: DB 'Введите строку:',0h ; сообщение
SECTION .bss ; Секция не инициализироване данных
buf1: RESB 80 ; Буфер размером 80 байт
SECTION .text ; Код программы
GLOBAL _start ; Начало программы
_start: ; Точка входа в программу
    mov eax, msg ;
    call sprintLF ;
    mov ecx,buf1 ;
    mov edx,80 ;
    call sread
    call quit
```

Рис. 4.13: Редактирование файла

Транслирую текст программы файла в объектный файл командой nasm -f elf lab6-2.asm. Создался объектный файл lab6-2.o. Выполняю компоновку объектного файла с помощью команды ld -m elf_i386 -o lab6-2 lab6-2.o Создался исполняемый файл lab6-2. Запускаю исполняемый файл (рис. 4.14).

```
ubuntu@ubuntu:~/work/study/arthPC/labs/lab06$ nasm -f elf lab6-2.asm
ubuntu@ubuntu:~/work/study/arthPC/labs/lab06$ ld -m elf_i386 -o lab6-2 lab6-2.o
ubuntu@ubuntu:~/work/study/arthPC/labs/lab06$ ./lab6-2
Введите строку:
Klimenko Kirill Ruslanovich
```

Рис. 4.14: Исполнение файла

Открываю файл lab6-2.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).

```
ubuntu@ubuntu:~/work/study/arthPC/labs/lab06$ nano lab6-2.asm
/home/ubuntu/work/study/arthPC/labs/lab06/lab6-2.asm
%include 'in_out.asm' ; Подключение внешнего файла
SECTION .data ; Секция инициализации данных
msg: DB 'Введите строку:',0h ; сообщение
SECTION .bss ; Секция не инициализироване данных
buf1: RESB 80 ; Буфер размером 80 байт
SECTION .text ; Код программы
GLOBAL _start ; Начало программы
_start: ; Точка входа в программу
    mov eax, msg ;
    call sprint ; changed from sprintLF
    mov ecx,buf1 ;
    mov edx,80 ;
    call sread
    call quit
```

Рис. 4.15: Отредактированный файл

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

```
ubuntu@ubuntu:~/work/study/арhPC/labs/lab06$ nasm -f elf lab6-2.asm
ubuntu@ubuntu:~/work/study/арhPC/labs/lab06$ ld -m elf_i386 -o lab6-2 lab6-2.o
ubuntu@ubuntu:~/work/study/арhPC/labs/lab06$ ./lab6-2
Введите строку:Klimenko Kirill Ruslanovich
Klimenko Kirill Ruslanovich
```

Рис. 4.16: Исполнение файла

Разница между первым исполняемым файлом lab6-2 и вторым lab6-2-2 в том, что запуск первого запрашивает ввод с новой строки, а программа, которая исполняется при запуске второго, запрашивает ввод без переноса на новую строку, потому что в этом заключается различие между подпрограммами sprintLF и sprint.

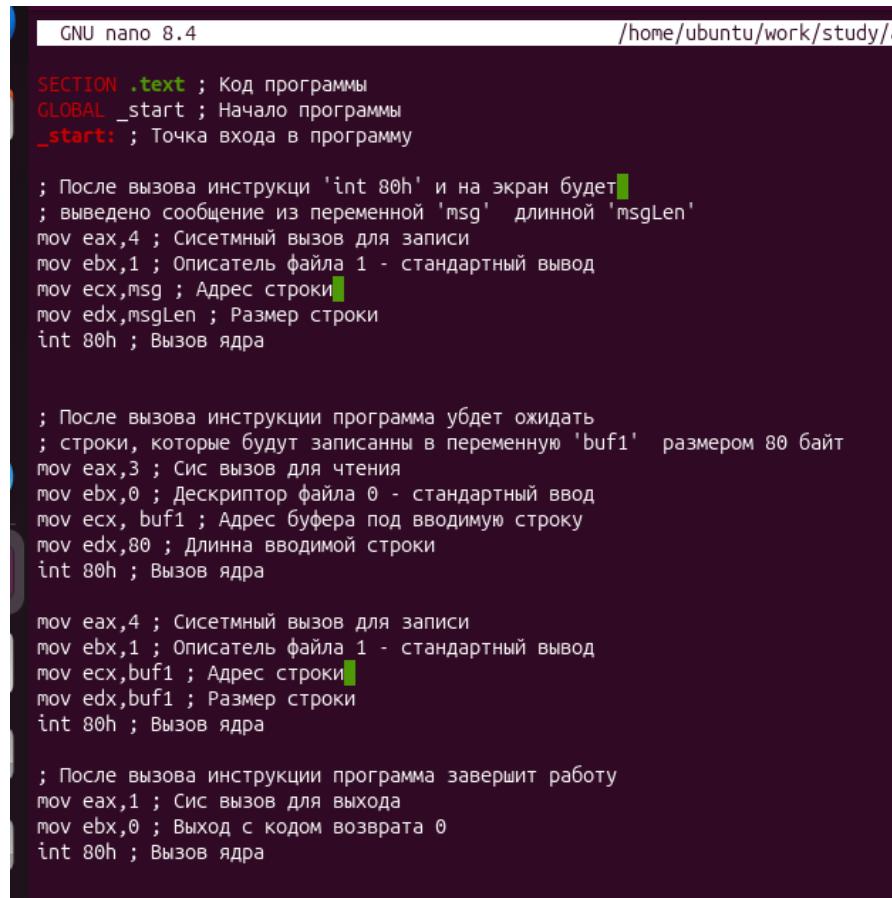
4.4 Выполнение заданий для самостоятельной работы

Создаю копию файла lab6-1.asm с именем lab6-1-1.asm с помощью функциональной клавиши F5 (рис. 4.17).

.n	Name	Size	Modify time	.n
/..		UP--DIR	Dec 8 16:40	/..
/presentation		220	Dec 8 16:40	/presentation
/report		240	Dec 8 16:40	/report
in_out.asm		3942	Dec 8 22:34	in_out.asm
*lab6-1		8744	Dec 8 22:55	*lab6-1
lab6-1-1.asm		1696	Dec 8 22:53	lab6-1-1.asm
lab6-1.asm		1696	Dec 8 22:53	lab6-1.asm
lab6-1.o		752	Dec 8 22:55	lab6-1.o
*lab6-2		9092	Dec 8 23:12	*lab6-2
lab6-2.asm		562	Dec 8 23:08	lab6-2.asm
lab6-2.o		1312	Dec 8 23:12	lab6-2.o

Рис. 4.17: 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. 4.18).



The screenshot shows the GNU nano 8.4 text editor interface. The title bar says "GNU nano 8.4" and the path is "/home/ubuntu/work/study/". The main area contains assembly code:

```
SECTION .text ; Код программы
GLOBAL _start ; Начало программы
_start: ; Точка входа в программу

; После вызова инструкции 'int 80h' и на экран будет
; выведено сообщение из переменной 'msg' длинной 'msgLen'
mov eax,4 ; Системный вызов для записи
mov ebx,1 ; Описатель файла 1 - стандартный вывод
mov ecx,msg ; Адрес строки
mov edx,msgLen ; Размер строки
int 80h ; Вызов ядра

; После вызова инструкции программы убдет ожидать
; строки, которые будут записаны в переменную 'buf1' размером 80 байт
mov eax,3 ; Сис вызов для чтения
mov ebx,0 ; Дескриптор файла 0 - стандартный ввод
mov ecx,buf1 ; Адрес буфера под вводимую строку
mov edx,80 ; Длинна вводимой строки
int 80h ; Вызов ядра

mov eax,4 ; Системный вызов для записи
mov ebx,1 ; Описатель файла 1 - стандартный вывод
mov ecx,buf1 ; Адрес строки
mov edx,buf1 ; Размер строки
int 80h ; Вызов ядра

; После вызова инструкции программы завершит работу
mov eax,1 ; Сис вызов для выхода
mov ebx,0 ; Выход с кодом возврата 0
int 80h ; Вызов ядра
```

Рис. 4.18: Редактирование файла

Создаю объектный файл lab6-1-1.o, отдаю его на обработку компоновщику, получаю исполняемый файл lab6-1-1, запускаю полученный исполняемый файл. Программа запрашивает ввод, ввожу свои ФИО, далее программа выводит введенные мною данные (рис. 4.19).

```
ubuntu@ubuntu:~/work/study/arhPC/labs/lab06$ nasm -f elf lab6-1-1.asm
ubuntu@ubuntu:~/work/study/arhPC/labs/lab06$ ld -m elf_i386 -o lab6-2 lab6-1-1.o
ubuntu@ubuntu:~/work/study/arhPC/labs/lab06$ ./lab6-1-1
bash: ./lab6-1-1: No such file or directory
ubuntu@ubuntu:~/work/study/arhPC/labs/lab06$ ls
in_out.asm  lab6-1-1.asm  lab6-1.o    lab6-2.o      report
lab6-1      lab6-1-1.o   lab6-1.o   lab6-2.asm  presentation
ubuntu@ubuntu:~/work/study/arhPC/labs/lab06$ ld -m elf_i386 -o lab6-1-1 lab6-1-1.o
ubuntu@ubuntu:~/work/study/arhPC/labs/lab06$ ./lab6-1-1
Введите строку:
Klimenko Kirill Ruslanovich
Klimenko Kirill Ruslanovich
```

Рис. 4.19: Исполнение файла

5 Выводы

При выполнении данной лабораторной работы я приобрел практические навыки работы в Midnight Commander, а также освоил инструкции языка ассемблера mov и int, научился копировать, создавать и запускать файлы с помощью Midnight Commander

6 Список литературы

1. Лабораторная работа №6