

Отчет по лабораторной работе №5

Дисциплина: архитектура компьютера

Тищенко Диана Борисовна

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	12
4.2	Подключение внешнего файла	16
4.3	Выполнение заданий для самостоятельной работы	18
5	Выводы	22

Список иллюстраций

4.1	Открытый тс	9
4.2	Перемещение между директориями	10
4.3	Создание каталога	10
4.4	Перемещение между директориями	11
4.5	Создание файла	12
4.6	Открытие файла для редактирования	13
4.7	Редактирование файла	14
4.8	Открытие файла для просмотра	15
4.9	Компиляция файла и передача на обработку компоновщику	15
4.10	Исполнение файла	15
4.11	Скачанный файл	16
4.12	Копирование файла	16
4.13	Копирование файла	16
4.14	Редактирование файла	17
4.15	Исполнение файла	17
4.16	Отредактированный файл	18
4.17	Исполнение файла	18
4.18	Копирование файла	19
4.19	Редактирование файла	19
4.20	Исполнение файла	20
4.21	Копирование файла	20
4.22	Редактирование файла	20
4.23	Исполнение файла	21

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 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 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

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

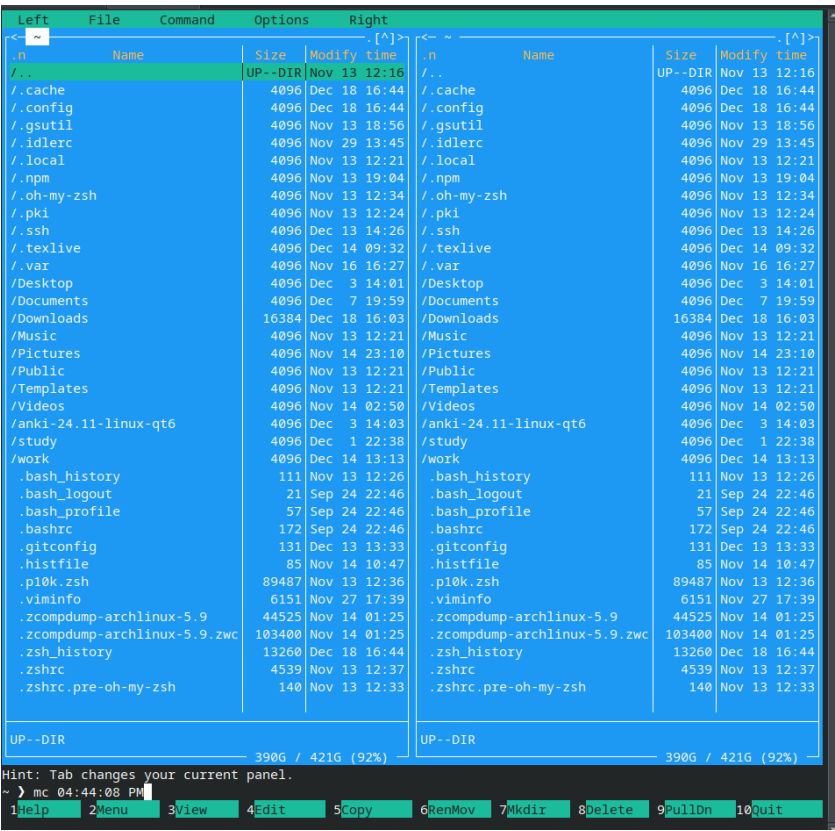


Рис. 4.1: Открытый mc

Перехожу в каталог ~/work/study/2024-2025/Архитектура Компьютера/arch-рс, используя файловый менеджер mc (рис. 4.2)

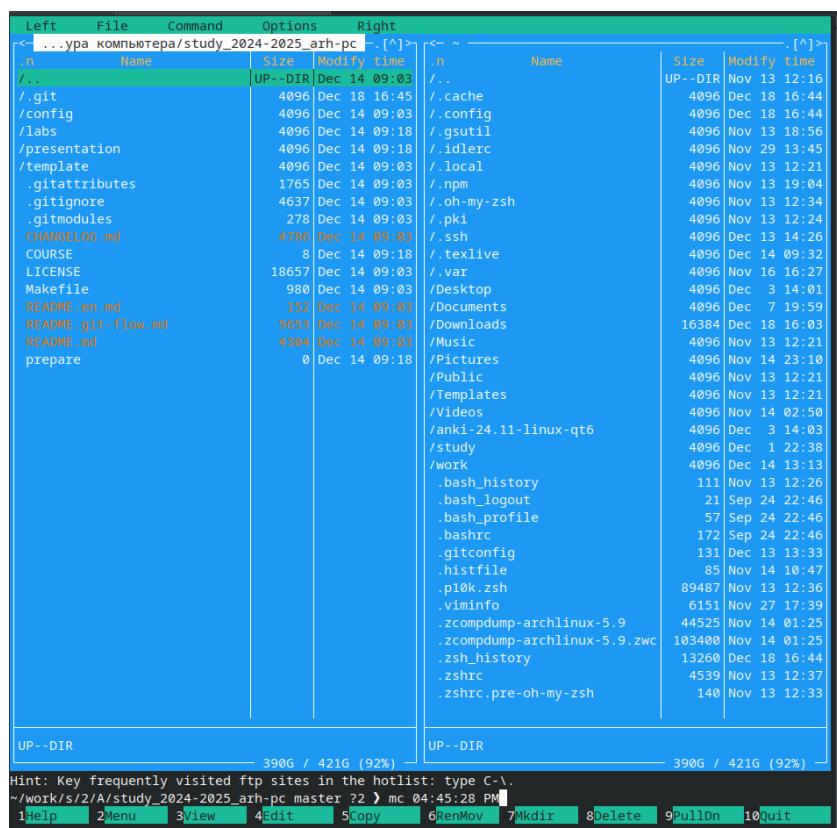


Рис. 4.2: Перемещение между директориями

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

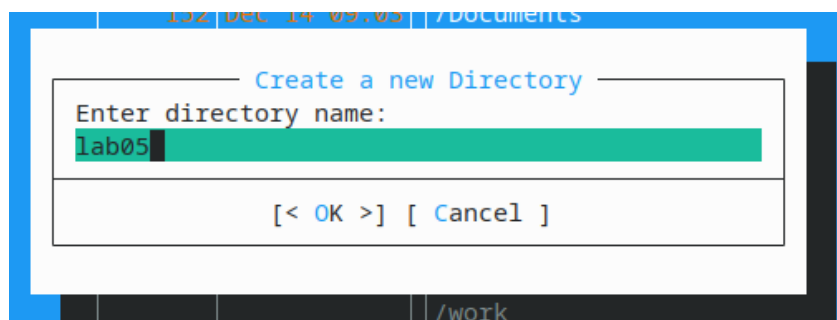


Рис. 4.3: Создание каталога

Переходу в созданный каталог (рис. 4.4).

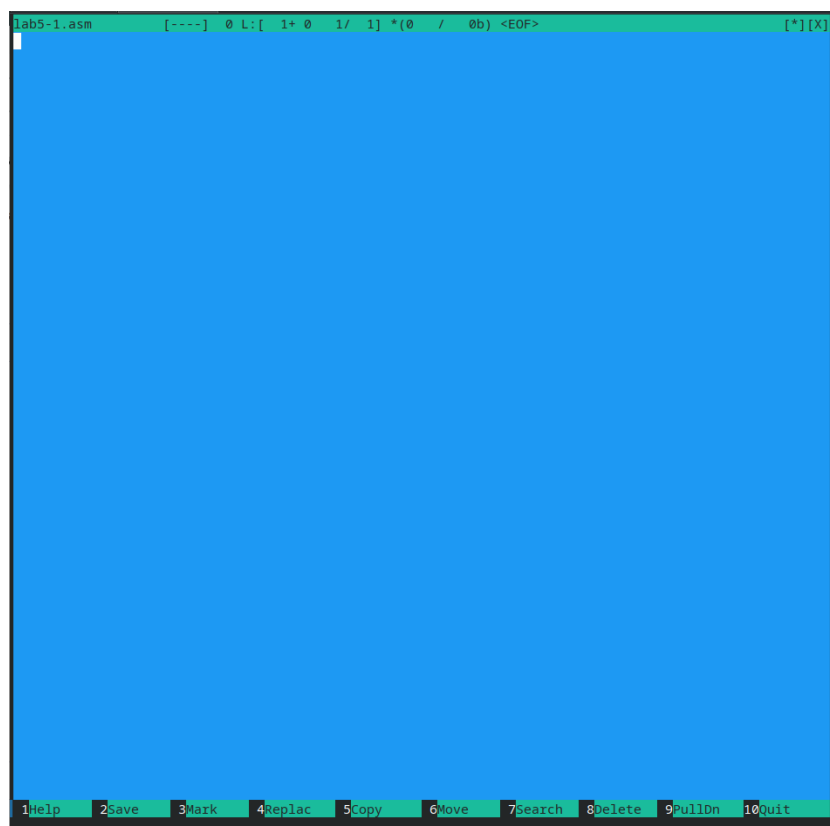


Рис. 4.6: Открытие файла для редактирования

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

```
lab5-1.asm [-M--] 7 L: [ 1+25 26/ 26] *(280 / 280b) <EOF> [*][X]
SECTION .data
msg: DB "Message: ",10

msglen: EQU $-msg
SECTION .text
Global _start
_start:

mov eax,4
mov ebx,1
mov ecx,msg
mov edx,msglen
int 80h

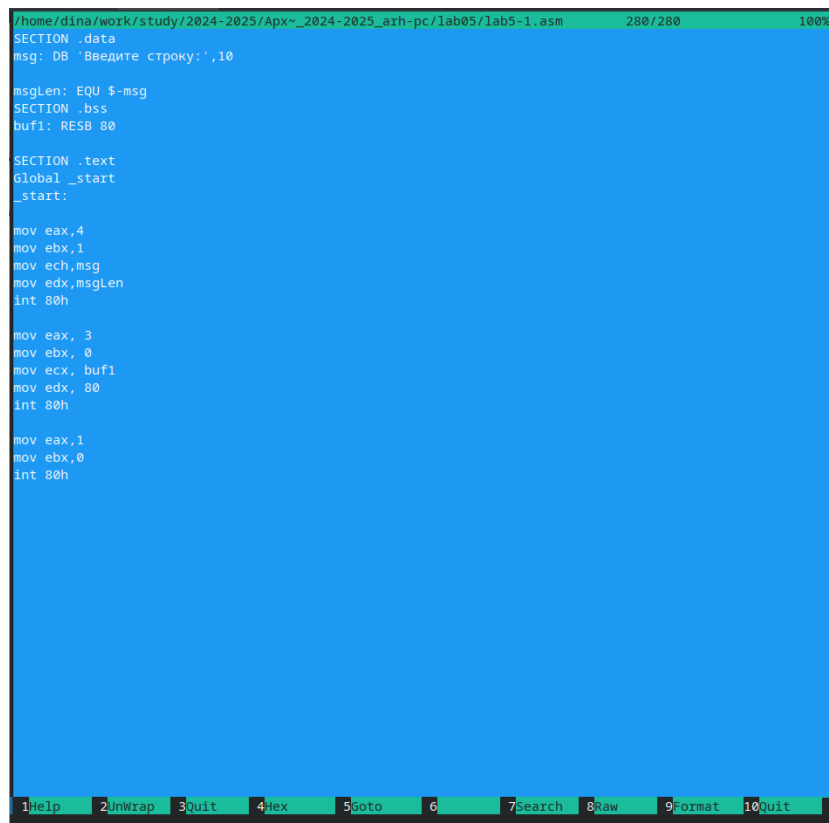
mov eax,3
mov ebx,0
mov ecx,buf1
mov edx,80
int 80h

mov eax,1
mov ebx,0
int 80h

1Help 2Save 3Mark 4Replac 5Copy 6Move 7Search 8Delete 9PullDn 10Quit
```

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/dina/work/study/2024-2025/Apx~_2024-2025_arh-pc/lab05/lab5-1.asm 280/280 100%
SECTION .data
msg: DB "Введите строку:",10

msgLen: EQU $-msg
SECTION .bss
buf1: RESB 80

SECTION .text
Global _start
_start:

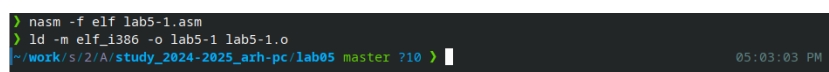
mov eax,4
mov ebx,1
mov ecx,msg
mov edx,msgLen
int 80h

mov eax,3
mov ebx,0
mov ecx,buf1
mov edx,80
int 80h

mov eax,1
mov ebx,0
int 80h
```

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`.



```
> nasm -f elf lab5-1.asm
> ld -m elf_i386 -o lab5-1 lab5-1.o
~/work/s/2/A/study_2024-2025_arh-pc/lab05 master ?10 >
```

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).



```
> ./lab5-1
Введите строку:
Тищенко Диана Борисовна
```

Рис. 4.10: Исполнение файла

4.2 Подключение внешнего файла

Скачиваю файл in_out.asm со страницы курса в ТУИС. Он сохранился в каталог “Загрузки” (рис. 4.11).

in_out.asm	3942	Dec 18 17:06
------------	------	--------------

Рис. 4.11: Скачанный файл

С помощью функциональной клавиши F5 копирую файл in_out.asm из каталога Загрузки в созданный каталог lab05 (рис. 4.12).

-<- ...мпыютера/study_2024-2025_arh-pc/lab05 -.[^]>-			
'm	Name	Size	Modify time
/..	UP--DIR		Dec 18 16:46
	in_out.asm	3942	Dec 18 17:06
*lab5-1		8744	Dec 18 17:03
	lab5-1.o	752	Dec 18 17:02
	lab5-1.asm	281	Dec 18 17:02

Рис. 4.12: Копирование файла

С помощью функциональной клавиши F5 копирую файл lab5-1 в тот же каталог, но с другим именем, для этого в появившемся окне mc прописываю имя для копии файла (рис. 4.13).

-<- ...мпыютера/study_2024-2025_arh-pc/lab05 -.[^]>-			
'm	Name	Size	Modify time
/..	UP--DIR		Dec 18 16:46
	in_out.asm	3942	Dec 18 17:06
*lab5-1		8744	Dec 18 17:03
	lab5-1.o	752	Dec 18 17:02
	lab5-2.asm	281	Dec 18 17:02
	lab5-1.asm	281	Dec 18 17:02

Рис. 4.13: Копирование файла

Изменяю содержимое файла lab5-2.asm во встроенном редакторе mcedit (рис. 4.14), чтобы в программе использовались подпрограммы из внешнего файла in_out.asm.


```
lab5-2.asm [----] 0 L: [ 1+ 0 1/ 15] *(0
%include 'in_out.asm'
SECTION .data
msg: DB 'Введите строку: ',0h
SECTION .bss
buf1: RESB 80
SECTION .text
GLOBAL _start
_start:
mov eax, msg
call sprint
mov ecx, buf1
mov edx, 80
call sread
call quit
```

Рис. 4.14: 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. 4.15).

```
> nasm -f elf lab5-2.asm
> ld -m elf_i386 -o lab5-2 lab5-2.o
> ./lab5-2
Введите строку: Тищенко Диана Борисовна
~/work/s/2/A/study_2024-2025_arh-pc/lab05 master ?16 >
```

Рис. 4.15: Исполнение файла

Открываю файл `lab5-2.asm` для редактирования в `mcedit` функциональной клавишей `F4`. Изменяю в нем подпрограмму `sprintLF` на `sprint`. Сохраняю изменения и открываю файл для просмотра, чтобы проверить сохранение действий (рис. 4.16).

```

%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

```

Рис. 4.16: Отредактированный файл

Снова транслирую файл, выполняю компоновку созданного объектного файла, запускаю новый исполняемый файл (рис. 4.17).

```

> nasm -f elf lab5-2.asm
> ld -m elf_i386 -o lab5-2 lab5-2.o
> ./lab5-2
Введите строку: Тищенко Диана Борисовна
~/work/s/2/A/study_2024-2025_arh-pc/lab05 master ?16 > 

```

Рис. 4.17: 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`.

4.3 Выполнение заданий для самостоятельной работы

1. Создаю копию файла lab5-1.asm с именем lab5-1-1.asm с помощью функциональной клавиши F5 (рис. 4.18).

in_out.asm	3942	Dec 18 17:00	/.idlerc
*lab5-1	8744	Dec 18 17:00	/.local
lab5-1.o	752	Dec 18 17:02	/.npm
lab5-1.asm	281	Dec 18 17:02	/.oh-my-zsh
lab5-1-1.asm	281	Dec 18 17:02	/.pki
			/.ssh

Рис. 4.18: 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. 4.19).

```
lab5-1-1.asm [----] 0 L:[ 1+ 0 1/ 27] *(0 /1523b) 00
SECTION .data ; Секция инициализированных данных
msg: DB 'Введите строку:',10
msgLen: EQU $-msg ; Длина переменной 'msg'
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,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 ;Descriptor 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
```

Рис. 4.19: Редактирование файла

2. Создаю объектный файл lab5-1-1.o, отдаю его на обработку компоновщику, получаю исполняемый файл lab5-1-1, запускаю полученный исполняемый файл. Программа запрашивает ввод, ввожу свои ФИО, далее программа выводит введенные мною данные (рис. 4.20).

```

> nasm -f elf lab5-1-1.asm
> ld -m elf_i386 -o lab5-1-1 lab5-1-1.o
ld: cannot find lab5-1-1.o: No such file or directory
> ld -m elf_i386 -o lab5-1-1 lab5-1-1.o
> ./lab5-1-1
Введите строку:
Тищенко Диана Борисовна
Тищенко Диана Борисовна
~/work/s/2/A/study_2024-2025_arh-pc/lab05 master ?21 >

```

Рис. 4.20: Исполнение файла

3. Создаю копию файла lab5-2.asm с именем lab5-2-1.asm с помощью функциональной клавиши F5 (рис. 4.21).

*lab5-2	9092	Dec 18 17:23	/.idlerc
lab5-2.o	1312	Dec 18 17:23	/.local
lab5-2.asm	214	Dec 18 17:20	/.npm
lab5-2-1.asm	214	Dec 18 17:20	/.oh-my-zsh
in_out.asm	3943	Dec 18 17:06	/.pki
*lab5-1	8744	Dec 18 17:03	/.ssh
lab5-1.o	752	Dec 18 17:02	/.texlive
lab5-1.asm	201	Dec 18 17:02	/.var
			/Desktop
			/Documents

Рис. 4.21: 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. 4.22).

```

lab5-2-1.asm [----] 0 L: [ 1+ 0 1/ 15] *(0 / 214b) 003
#include "in_out.asm"
SECTION .data
msg: DB "Введите строку: ",0h
SECTION .bss
buf1: RESB 80
SECTION .text
GLOBAL _start
_start:
mov eax, msg
call sprint
mov ecx, buf1
mov edx, 80
call sread
call quit

```

Рис. 4.22: Редактирование файла

4. Создаю объектный файл lab5-2-1.o, отдаю его на обработку компоновщику, получаю исполняемый файл lab5-2-1, запускаю полученный исполняемый файл. Программа запрашивает ввод без переноса на новую строку, ввожу свои ФИО, далее программа выводит введенные мною данные (рис. 4.23).

```
> nasm -f elf lab5-2-1.asm
> ld -m elf_i386 -o lab5-2-1 lab5-2-1.o
> ./lab5-2-1
Введите строку: Тищенко Диана Борисовна
Тищенко Диана Борисовна
~/work/s/2/A/study_2024-2025_arh-pc/lab05 master ?24 > |
```

Рис. 4.23: Исполнение файла

5 Выводы

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