

Отчет по лабораторной работе №5

Дисциплина: архитектура компьютера

Учаева Алёна Сергеевна

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	17
	Список литературы	18

Список иллюстраций

4.1	Открытый тс	8
4.2	Перемещение между директориями	9
4.3	Создание каталога	9
4.4	Создание файла	10
4.5	Открытие файла для редактирования	10
4.6	Редактирование файла	11
4.7	Открытие файла для проверки	11
4.8	Исполнение файла	12
4.9	Копирование файла	12
4.10	Копирование файла	13
4.11	Редактирование файла	13
4.12	Исполнение файла	14
4.13	Исполнение файла	14
4.14	Редактирование файла	15
4.15	Исполнение файла	15
4.16	Редактирование файла	16
4.17	Исполнение файла	16

1 Цель работы

Целью 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 (учетверённое слово);
- DT (define ten bytes) — определяет переменную размером в 10 байт.

Инструкция языка ассемблера mov предназначена для дублирования данных источника в приёмнике. В общем виде эта инструкция записывается в виде `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. В общем виде она записывается в

виде `int n` Здесь `n` — номер прерывания, принадлежащий диапазону 0–255

4 Выполнение лабораторной работы

4.1 Основы работы с mc

Открываю Midnight Commander, с помощью команды mc (рис. 4.1).

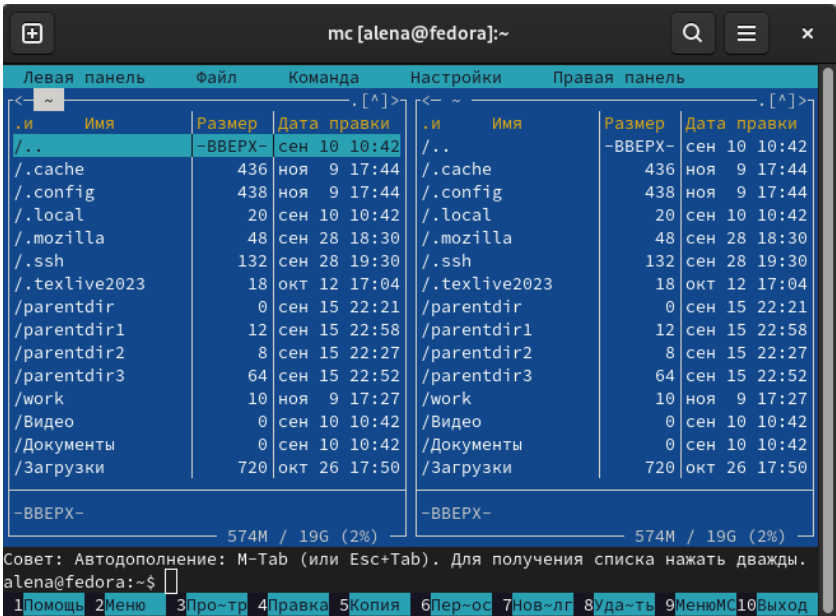


Рис. 4.1: Открытый mc

Перехожу в каталог ~/work/study/2024-2025/Архитектура компьютера/arch-pc (рис. 4.2).

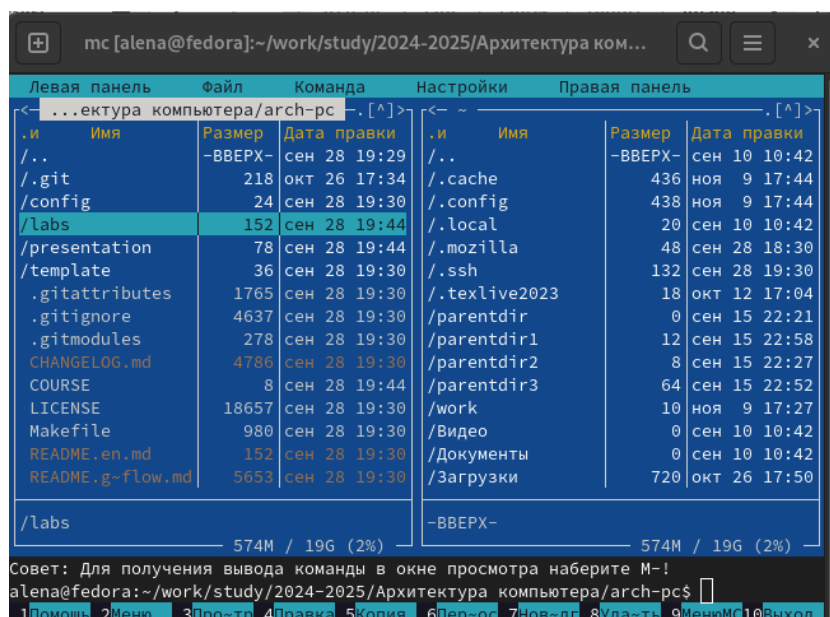


Рис. 4.2: Перемещение между директориями

Создаю каталог lab05 с помощью функциональной клавиши F7 (рис. 4.3).

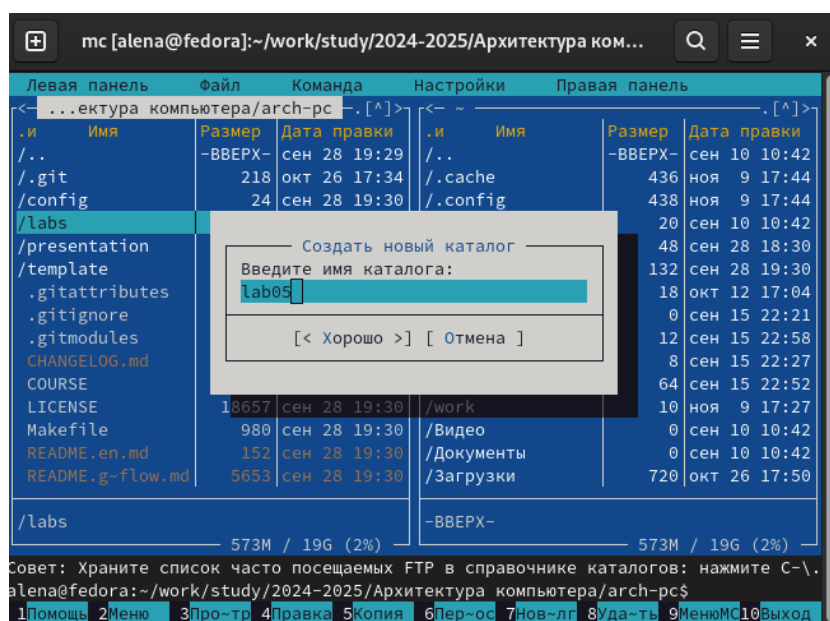


Рис. 4.3: Создание каталога

Перехожу в созданный каталог и в строке ввода прописываю команду touch lab5-1.asm, чтобы создать файл (рис. 4.4).

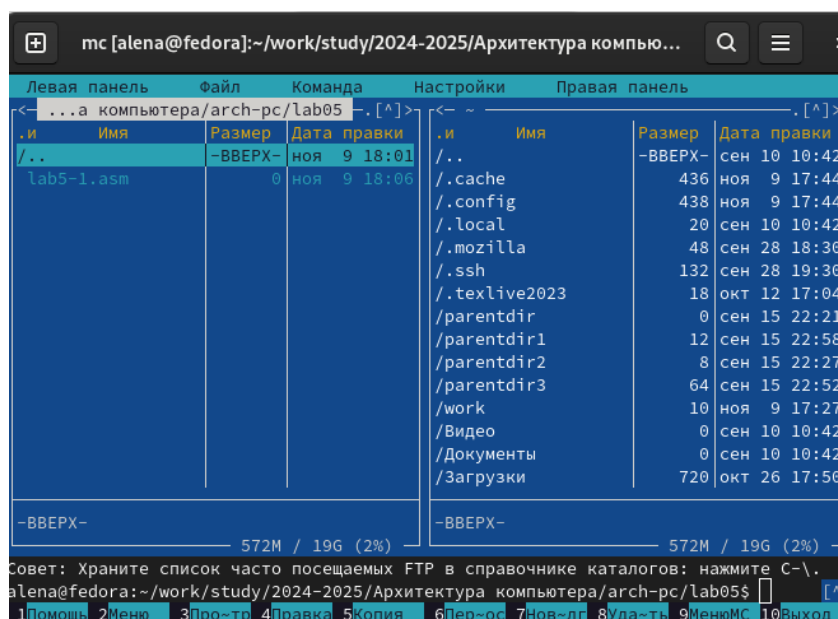


Рис. 4.4: Создание файла

4.2 Структура программы на языке ассемблера NASM

С помощью функциональной клавиши открываю созданный файл для редактирования во встроенном редакторе (рис. 4.5).

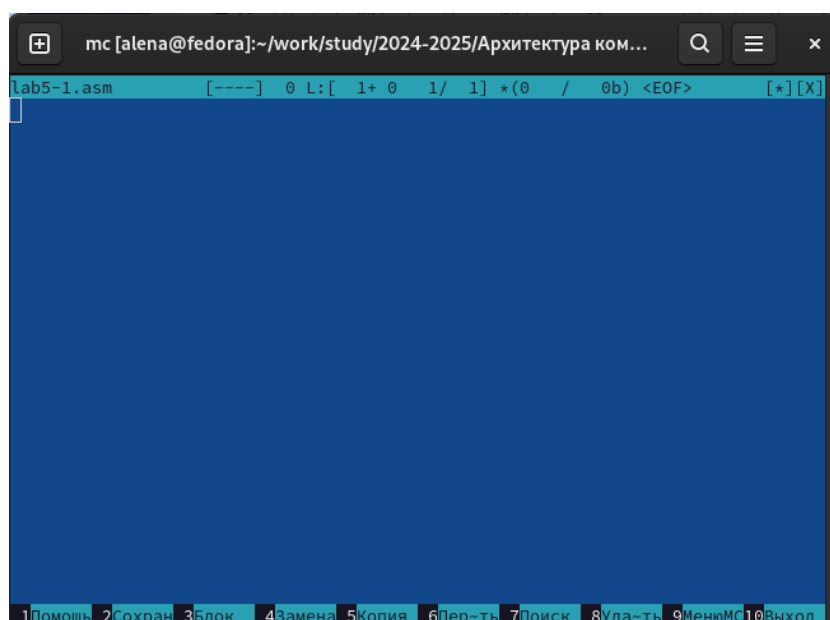
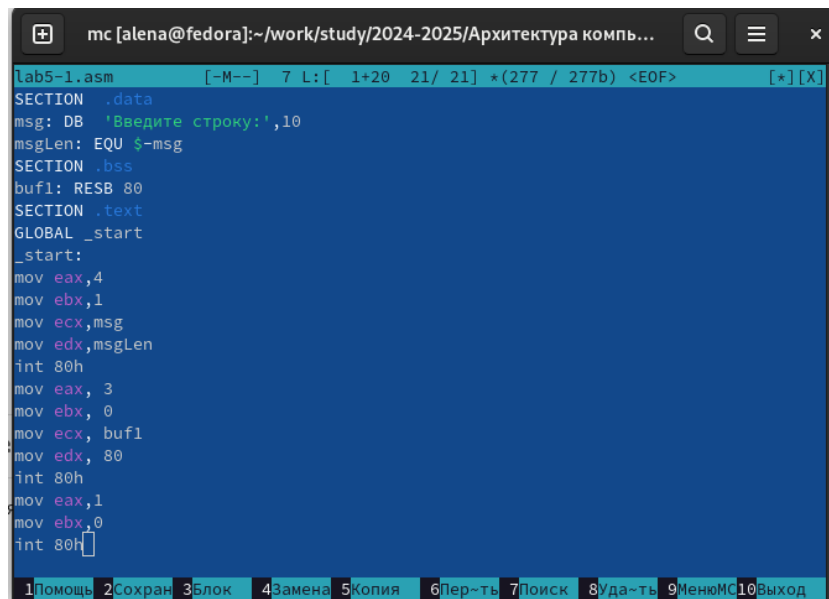


Рис. 4.5: Открытие файла для редактирования

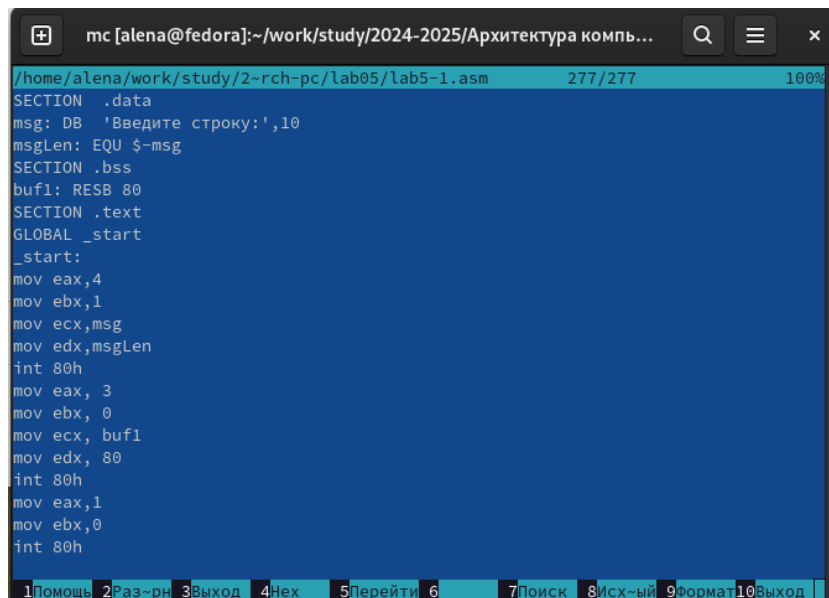
Далее ввожу код программы, сохраняю изменения и сохраняю файл (рис. 4.6).



```
lab5-1.asm [-M--] 7 L: [ 1+20 21/ 21] *(277 / 277b) <EOF> [*] [X]
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. 4.6: Редактирование файла

С помощью функциональной клавиши F3 открываю файл и проверяю, что файл содержит текст программы (рис. 4.7).



```
/home/alena/work/study/2~rch-pc/lab05/lab5-1.asm 277/277 100%
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. 4.7: Открытие файла для проверки

Транслирую текст программы в объектный файл. Выполняю компоновку объ-

ектного файла и запускаю получившийся исполняемый файл. На запрос ввожу свое ФИО (рис. 4.8).

```
alena@fedora:~/work/study/2024-2025/Архитектура компьютера/arch-pc/lab05$ nasm -f elf lab5-1.asm
alena@fedora:~/work/study/2024-2025/Архитектура компьютера/arch-pc/lab05$ ./lab5-1
Введите строку:
Учаева Алёна Сергеевна
alena@fedora:~/work/study/2024-2025/Архитектура компьютера/arch-pc/lab05$
```

Рис. 4.8: Исполнение файла

4.3 Подключение внешнего файла

Скачиваю файл in_out.asm со страницы курса в ТУИС и перемещаю в созданный каталог lab05 (рис. 4.9).

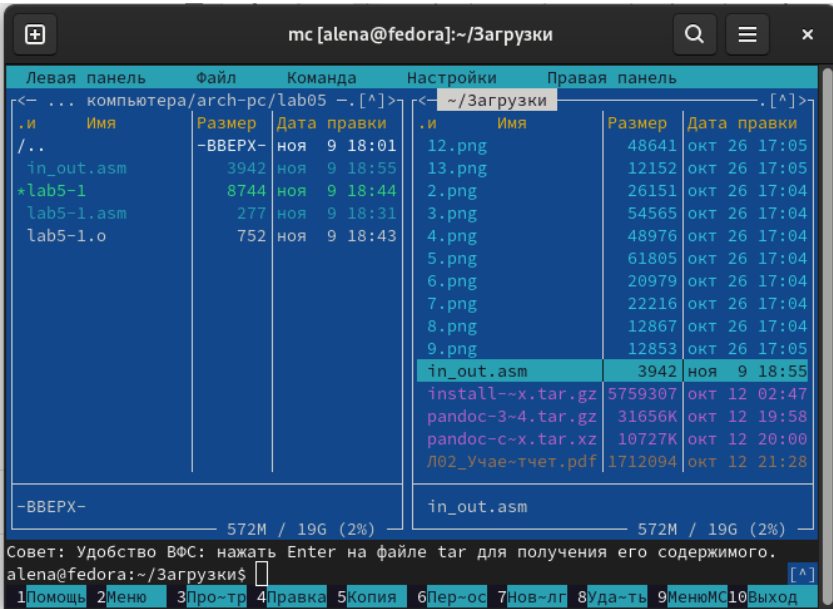


Рис. 4.9: Копирование файла

С помощью функциональной клавиши F5 создаю копию файла с другим именем (рис. 4.10).

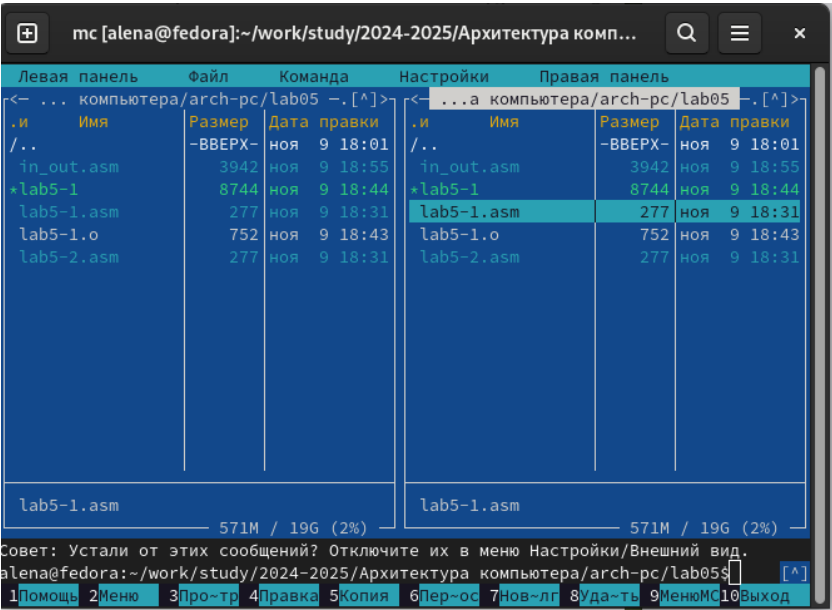


Рис. 4.10: Копирование файла

Исправляю текст программы в файле lab5-2.asm (рис. 4.11).

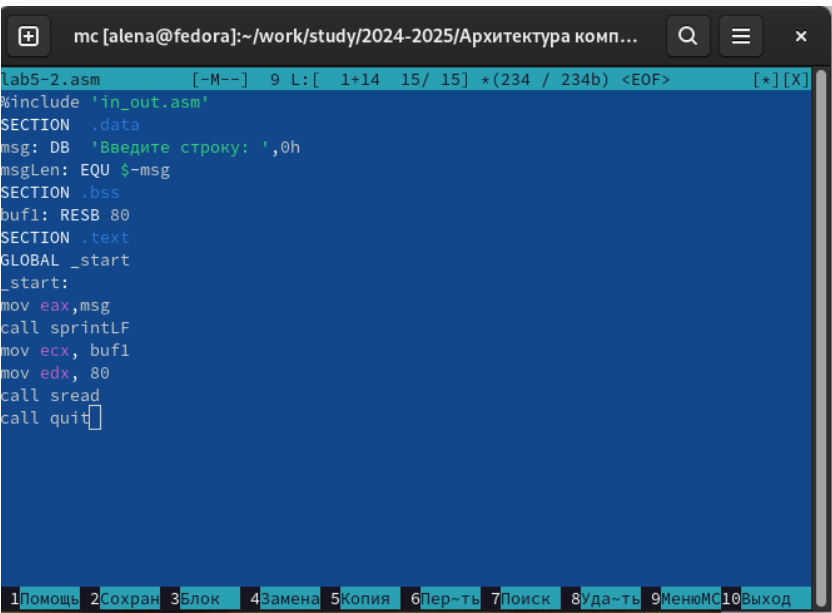


Рис. 4.11: Редактирование файла

Транслирую текст программы в объектный файл. Выполняю компоновку объ-

ектного файла и запускаю получившийся исполняемый файл. На запрос ввожу свое ФИО (рис. 4.12).

```
alena@fedora: ~/work/study/2024-2025/Архитектура компьютера/arch-pc/lab05$ ld -m elf_i386  
-o lab5-2 lab5-2.o  
alena@fedora: ~/work/study/2024-2025/Архитектура компьютера/arch-pc/lab05$ ./lab5-2  
Введите строку:  
Учаева Алёна Сергеевна
```

Рис. 4.12: Исполнение файла

В файле lab5-2.asm заменяю подпрограмму sprintLF на sprint, создаю исполняемый файл и проверяю его работу (рис. 4.13).

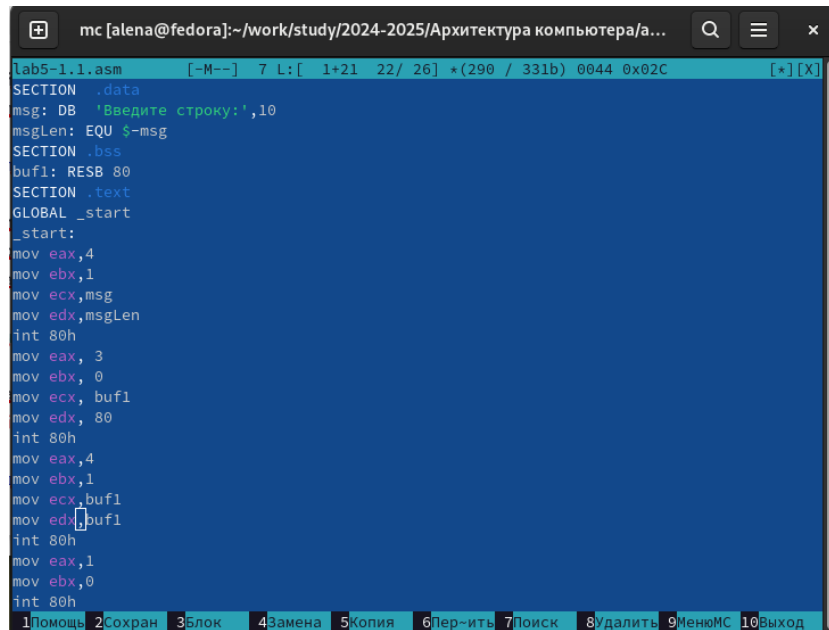
```
alena@fedora: ~/work/study/2024-2025/Архитектура компьютера/arch-pc/lab05$ nasm -f elf lab5-2.asm  
alena@fedora: ~/work/study/2024-2025/Архитектура компьютера/arch-pc/lab05$ ld -m elf_i386  
-o lab5-2-2 lab5-2.o  
alena@fedora: ~/work/study/2024-2025/Архитектура компьютера/arch-pc/lab05$ ./lab5-2-2  
Введите строку: Учаева Алёна Сергеевна  
alena@fedora: ~/work/study/2024-2025/Архитектура компьютера/arch-pc/lab05$
```

Рис. 4.13: Исполнение файла

Разница между первым исполняемым файлом и вторым в том, что запуск первого файла запрашивает ввод с новой строки, а второго запрашивает ввод без перехода на новую строку

4.4 Выполнение заданий для самостоятельной работы

Создаю копию файла lab5-1.asm и вношу изменения в программу (рис. 4.14).

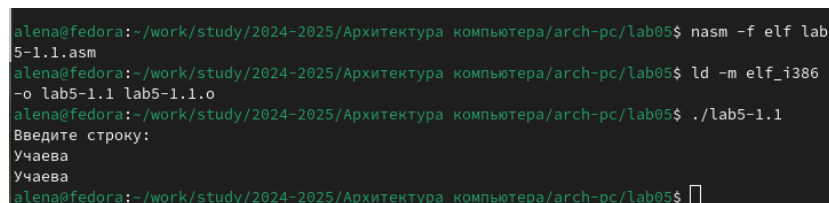


```
lab5-1.1.asm [-M--] 7 L: [ 1+21 22/ 26] *(290 / 331b) 0044 0x02C [*] [X]
SECTION .data
msg: DB 'Введите строку:',10
msgLen: EQU $-msg
SECTION .bss
buf1: RESB 80
SECTION .text
GLOBAL _start
_start:
mov eax,4
mov ebx,1
mov ecx,msg
mov edx,msgLen
int 80h
mov eax,3
mov ebx,0
mov ecx,buf1
mov edx,80
int 80h
mov eax,4
mov ebx,1
mov ecx,buf1
mov edx,buf1
int 80h
mov eax,1
mov ebx,0
int 80h
```

1Помощь 2Сохран 3Блок 4Замена 5Копия 6Пер-ить 7Поиск 8Удалить 9МенюМС 10Выход

Рис. 4.14: Редактирование файла

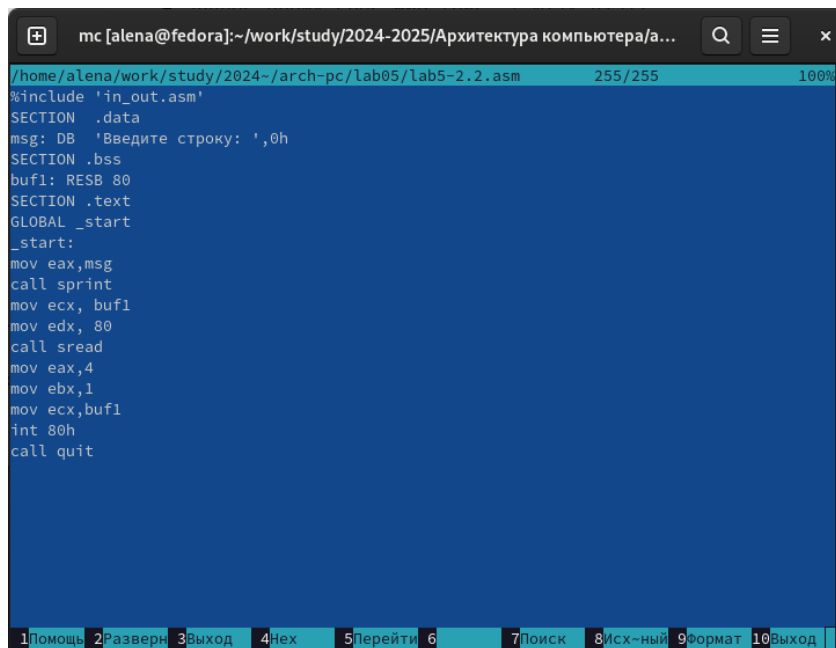
Транслирую текст программы в объектный файл. Выполняю компоновку объектного файла и запускаю получившийся исполняемый файл. На запрос ввожу свое ФИО (рис. 4.15).



```
alena@fedora: ~/work/study/2024-2025/Архитектура компьютера/arch-pc/lab05$ nasm -f elf lab5-1.1.asm
alena@fedora: ~/work/study/2024-2025/Архитектура компьютера/arch-pc/lab05$ ld -m elf_i386 -o lab5-1.1 lab5-1.1.o
alena@fedora: ~/work/study/2024-2025/Архитектура компьютера/arch-pc/lab05$ ./lab5-1.1
Введите строку:
Учаева
Учаева
alena@fedora: ~/work/study/2024-2025/Архитектура компьютера/arch-pc/lab05$
```

Рис. 4.15: Исполнение файла

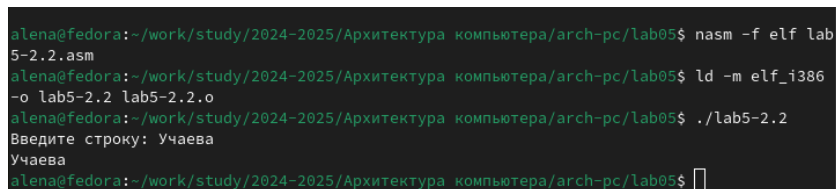
Создаю копию файла lab5-2.asm, исправляю текст программы с использованием подпрограмм из внешнего файла n_out.asm (рис. 4.16).



```
mc [alena@fedora]:~/work/study/2024-2025/Архитектура компьютера/a...
/home/alena/work/study/2024-2025/arch-pc/lab05/lab5-2.2.asm 255/255 100%
#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
mov eax,4
mov ebx,1
mov ecx,buf1
int 80h
call quit
```

Рис. 4.16: Редактирование файла

Транслирую текст программы в объектный файл. Выполняю компоновку объектного файла и запускаю получившийся исполняемый файл. На запрос ввожу свое ФИО (рис. 4.17).



```
alena@fedora: ~/work/study/2024-2025/Архитектура компьютера/arch-pc/lab05$ nasm -f elf lab5-2.2.asm
alena@fedora: ~/work/study/2024-2025/Архитектура компьютера/arch-pc/lab05$ ld -m elf_i386 -o lab5-2.2 lab5-2.2.o
alena@fedora: ~/work/study/2024-2025/Архитектура компьютера/arch-pc/lab05$ ./lab5-2.2
Введите строку: Учаева
Учаева
alena@fedora: ~/work/study/2024-2025/Архитектура компьютера/arch-pc/lab05$
```

Рис. 4.17: Исполнение файла

5 Выводы

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

Список литературы

1. Архитектура ЭВМ