

Отчёт по лабораторной работе 5

**Основы работы с Midnight Commander. Структура программы на языке
ассемблера NASM. Системные вызовы в ОС GNU Linux**

Зиборова Вероника Николаевна НММбд-02-24

Содержание

1	Цель работы	5
2	Выполнение лабораторной работы	6
2.1	Знакомство с Midnight Commander	6
2.2	Подключение внешнего файла in_out.asm	10
2.3	Задание для самостоятельной работы	14
3	Выводы	18
4	Вопросы для самопроверки	19

Список иллюстраций

2.1	Запуск Midnight Commander	6
2.2	Создание каталога lab05	7
2.3	Создание файла lab05-1.asm	7
2.4	Редактирование программы в файле lab05-1.asm	8
2.5	Просмотр содержимого файла lab05-1.asm	9
2.6	Запуск программы lab05-1.asm	10
2.7	Копирование файла in_out.asm в рабочий каталог	10
2.8	Копирование файла lab05-1.asm в lab05-2.asm	11
2.9	Программа в файле lab05-2.asm	12
2.10	Запуск программы lab05-2.asm	12
2.11	Обновленная программа в lab05-2.asm	13
2.12	Запуск обновленной программы lab05-2.asm	14
2.13	Копирование файла lab05-1.asm для нового задания	14
2.14	Код программы в файле lab05-3.asm	15
2.15	Запуск программы lab05-3.asm	15
2.16	Копирование файла lab05-2.asm для модификации	16
2.17	Код программы в файле lab05-4.asm	17
2.18	Запуск программы lab05-4.asm	17

Список таблиц

1 Цель работы

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

2 Выполнение лабораторной работы

2.1 Знакомство с Midnight Commander

Я запустила Midnight Commander и, используя стрелочные клавиши и клавишу Enter, перешла в каталог ~/work/arch-рс. Затем нажала F7 для создания нового каталога lab05.

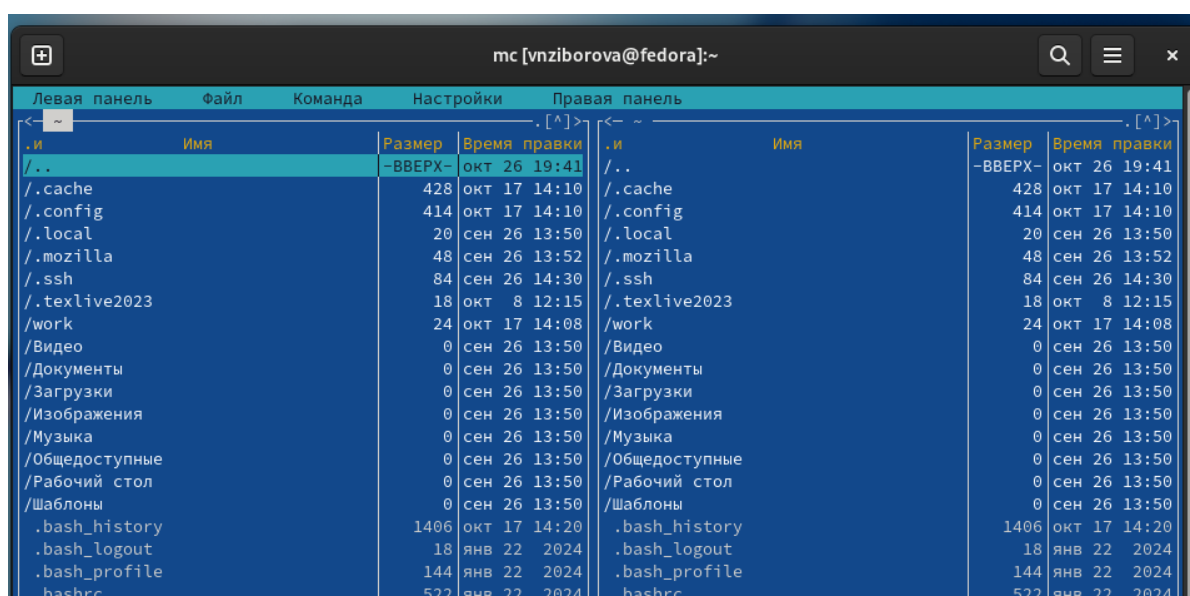


Рис. 2.1: Запуск Midnight Commander

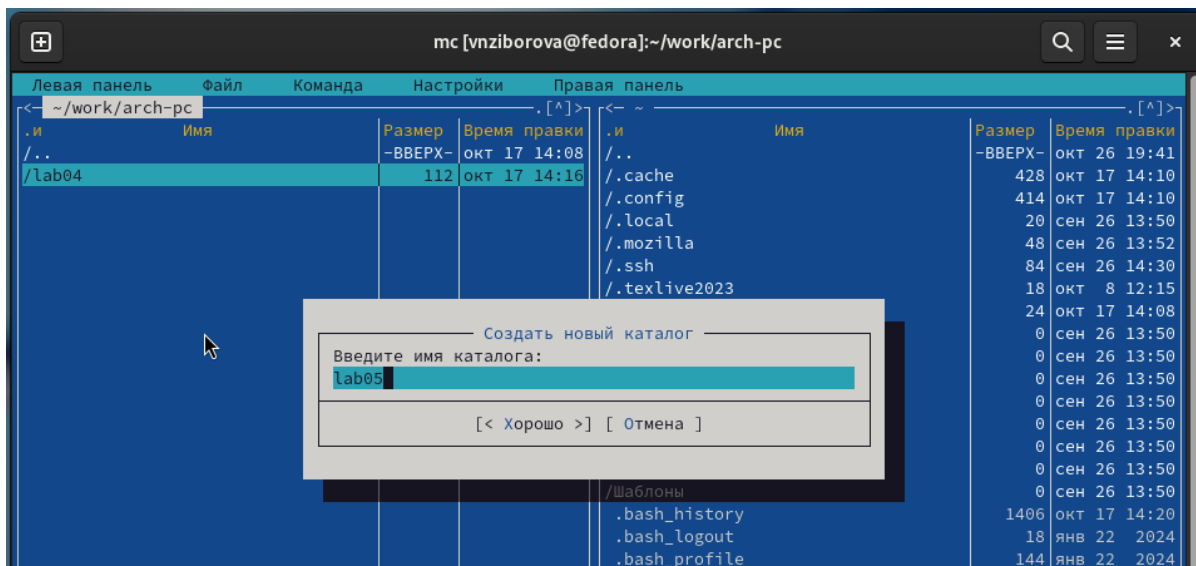


Рис. 2.2: Создание каталога lab05

С помощью команды `touch` я создала файл `lab05-1.asm`.

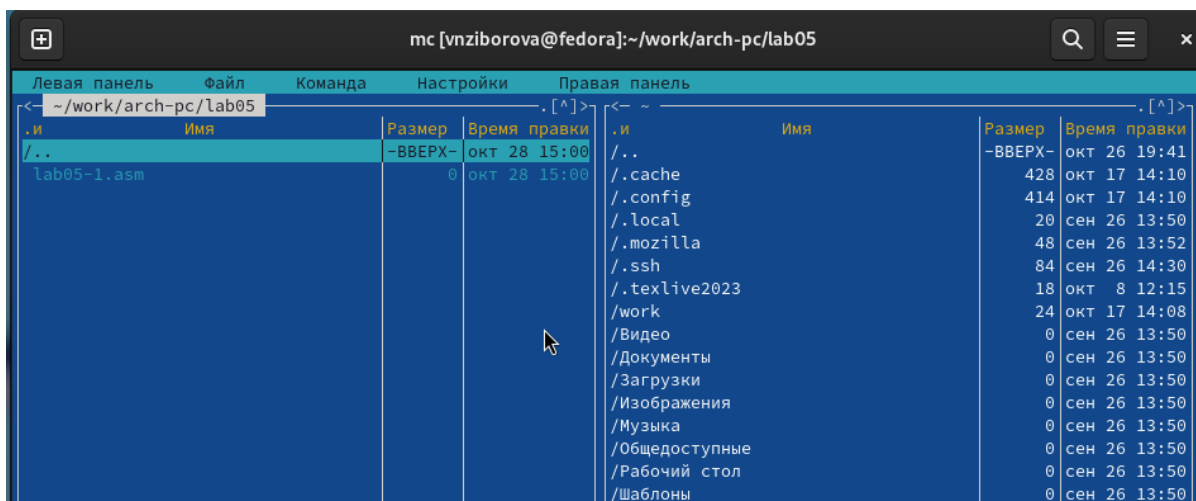


Рис. 2.3: Создание файла lab05-1.asm

Я открыла файл для редактирования, нажав F4, выбрала редактор `mceditor` и написала код программы в соответствии с заданием.

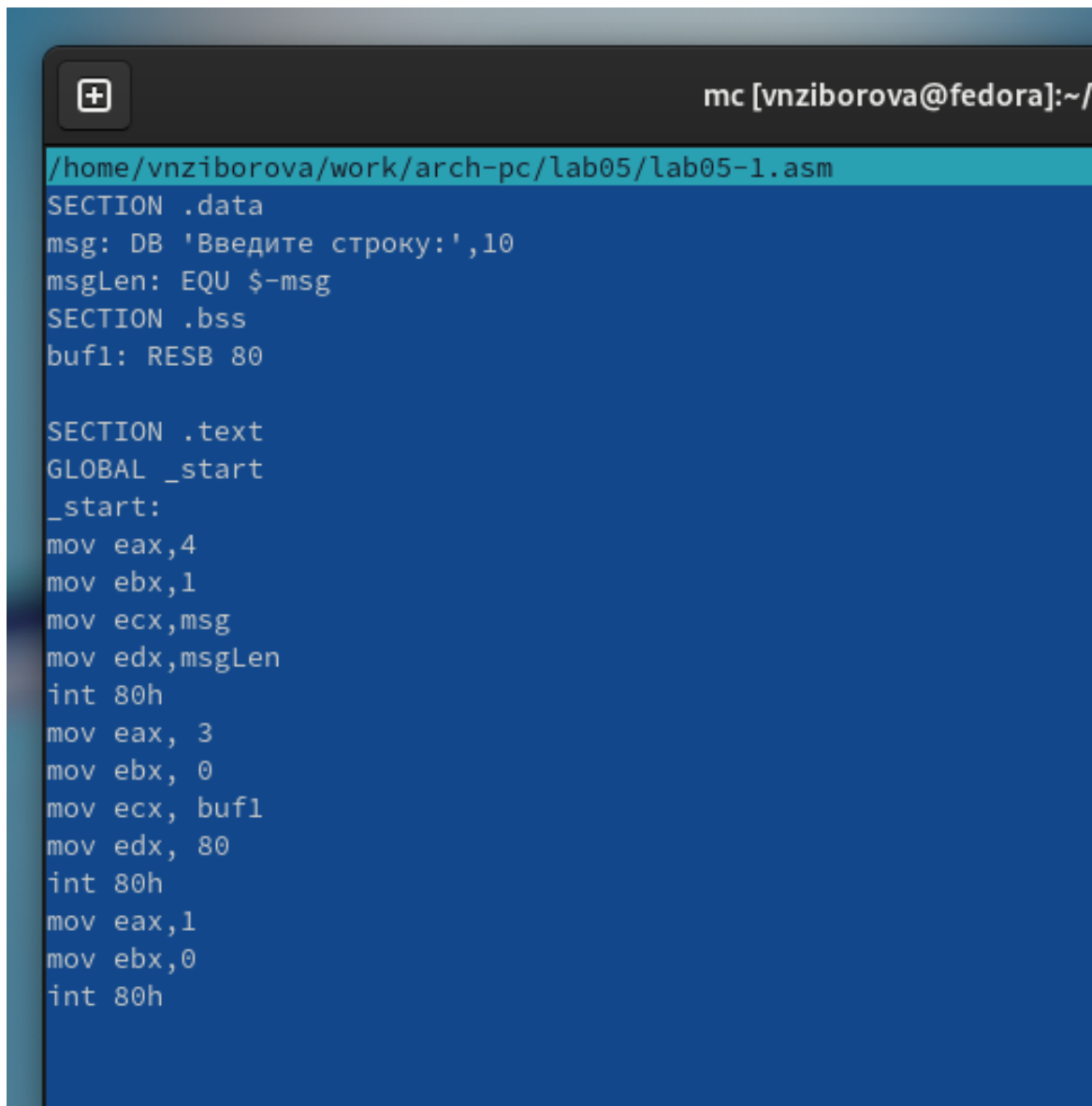


```
lab05-1.asm [----] 9 L: [ 1+10 11/ 24] *(
SECTION .data
msg: DB 'Введите строку:',10
msgLen: EQU $-msg
SECTION .bss
buf1: RESB 80

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. 2.4: Редактирование программы в файле lab05-1.asm

Для проверки я открыла файл на просмотр, нажав F3, и убедилась, что он содержит правильный код.



```
mc [vnziborova@fedora]:~/  
/home/vnziborova/work/arch-pc/lab05/lab05-1.asm  
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. 2.5: Просмотр содержимого файла lab05-1.asm

Я скомпилировала программу, сгенерировала объектный файл, произвела компоновку и запустила исполняемый файл, чтобы убедиться в корректности работы.

```

vnziborova@fedora:~/work/arch-pc/lab05$ nasm -f elf lab05-1.asm
vnziborova@fedora:~/work/arch-pc/lab05$ ld -m elf_i386 lab05-1.o -o lab05-1
vnziborova@fedora:~/work/arch-pc/lab05$ ./lab05-1
Введите строку:
Veronika
vnziborova@fedora:~/work/arch-pc/lab05$

```

Рис. 2.6: Запуск программы lab05-1.asm

2.2 Подключение внешнего файла in_out.asm

Я скачала файл in_out.asm и разместила его в рабочем каталоге. Для копирования я использовала клавишу F5, а для перемещения – клавишу F6.

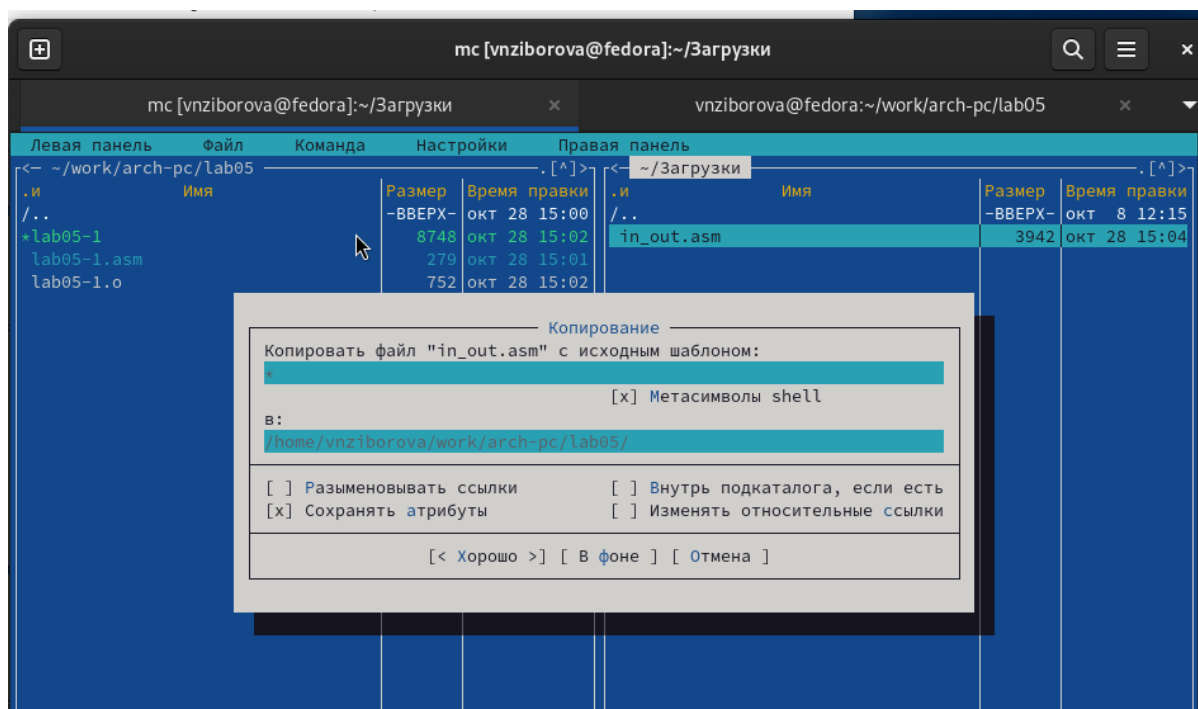


Рис. 2.7: Копирование файла in_out.asm в рабочий каталог

Я также скопировала файл lab05-1.asm в новый файл lab05-2.asm.

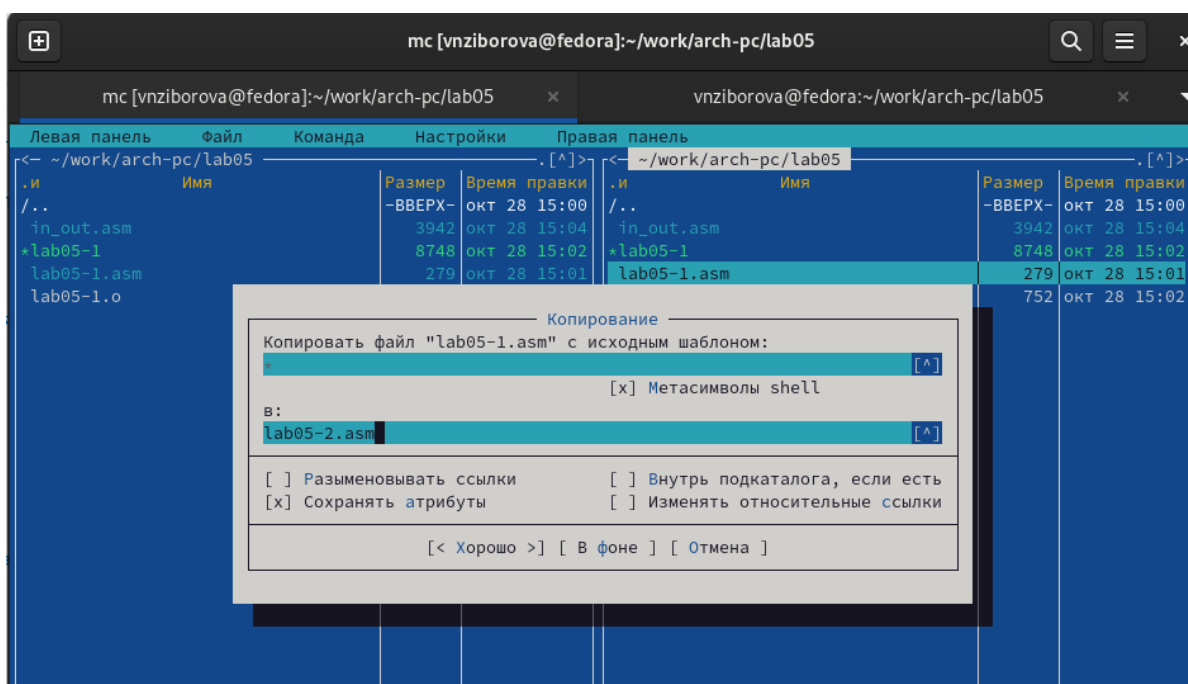
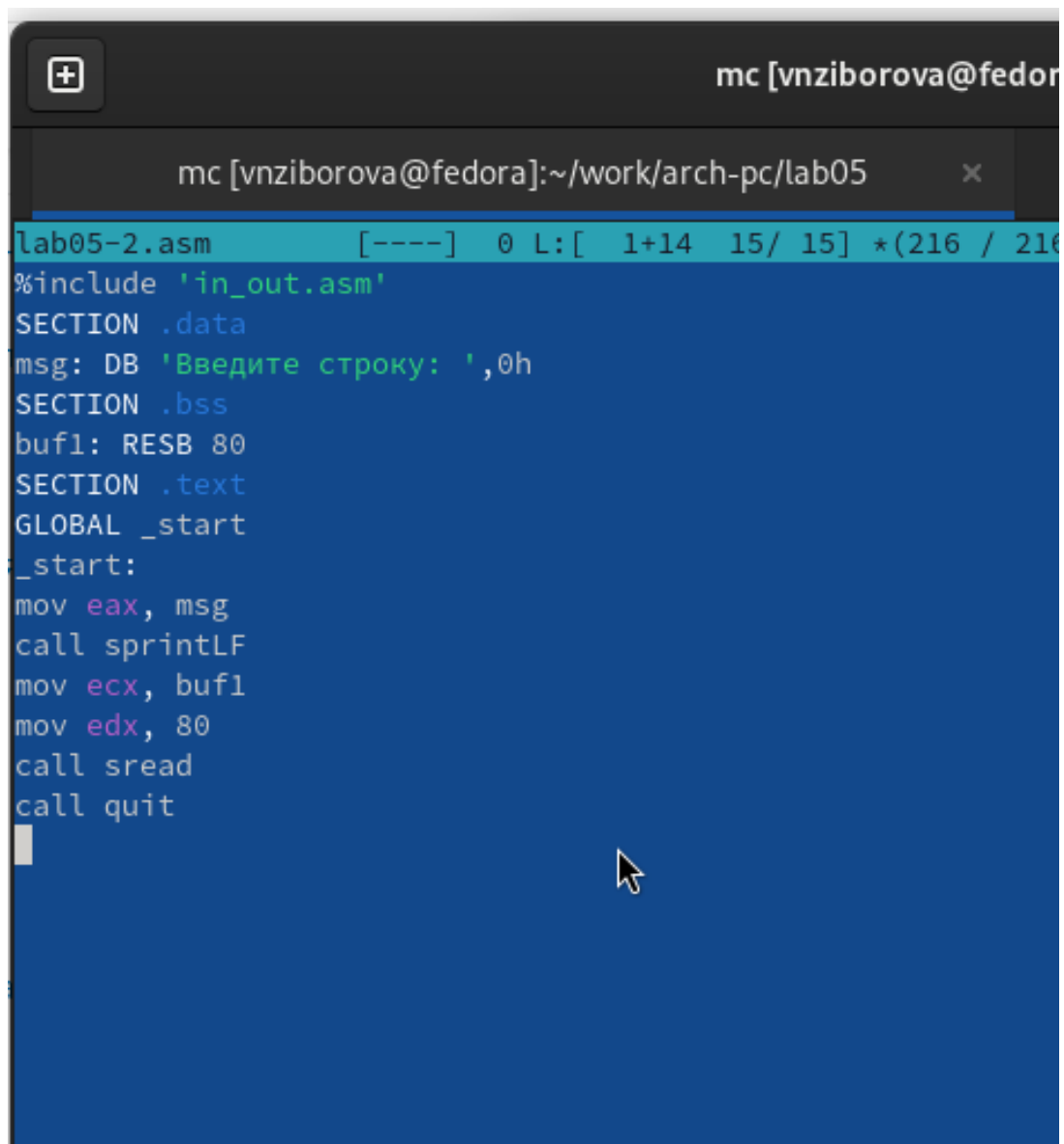


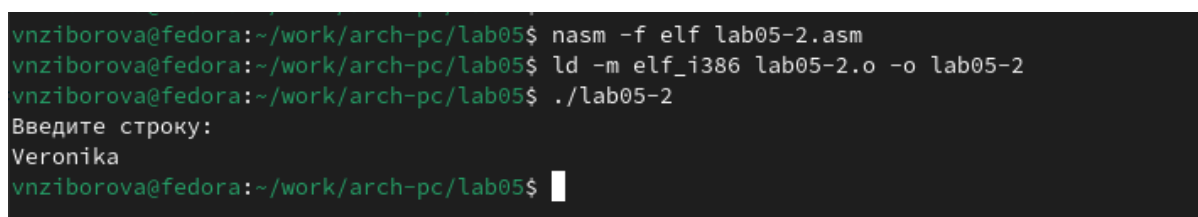
Рис. 2.8: Копирование файла lab05-1.asm в lab05-2.asm

В файле lab05-2.asm я написала код программы, используя подпрограммы из внешнего файла in_out.asm. Программа была успешно скомпилирована и протестирована.



```
lab05-2.asm [----] 0 L: [ 1+14 15/ 15] *(216 / 216)
#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 sprintLF
mov ecx, buf1
mov edx, 80
call sread
call quit
```

Рис. 2.9: Программа в файле lab05-2.asm

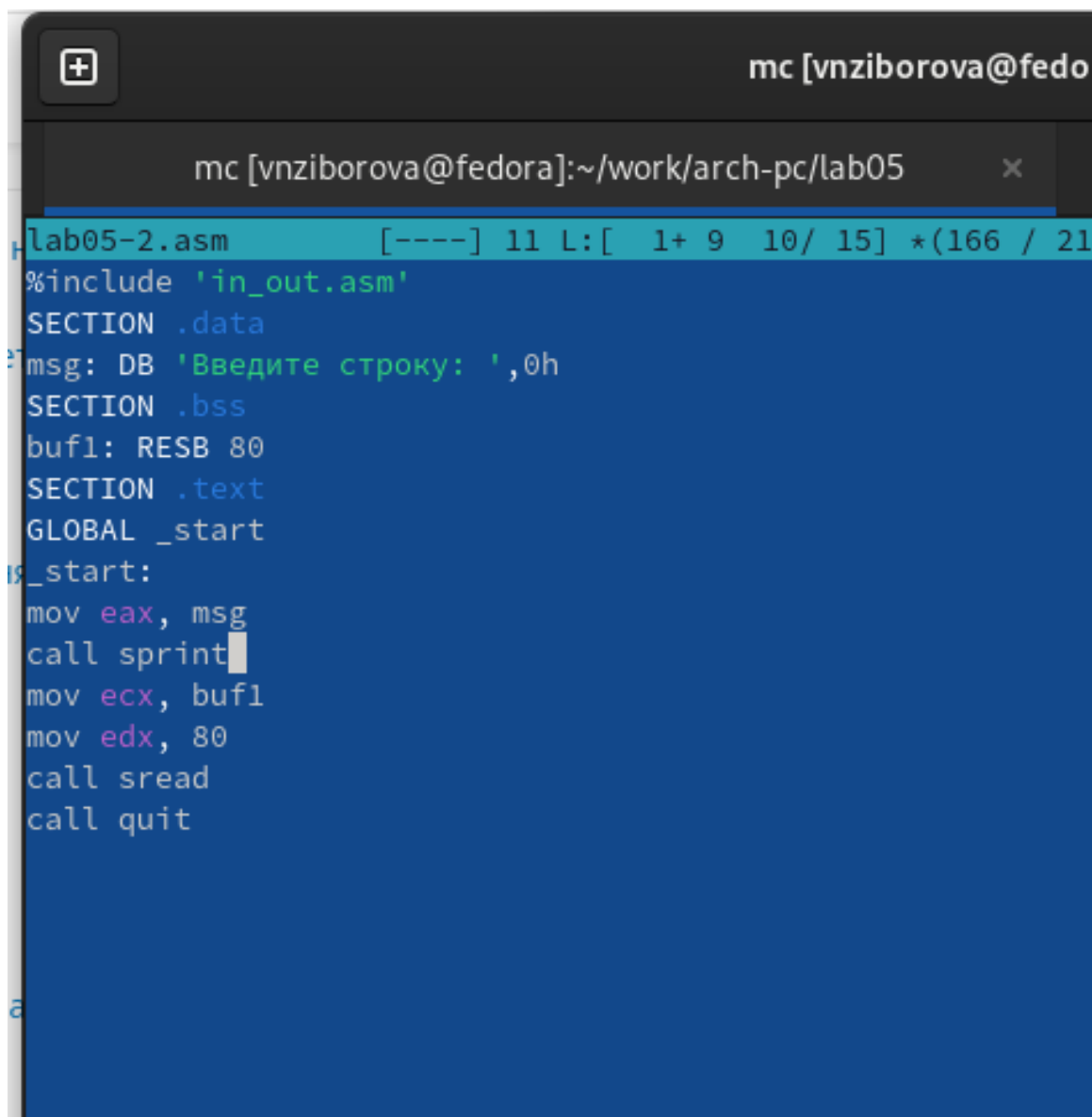


```
vnziborova@fedora:~/work/arch-pc/lab05$ nasm -f elf lab05-2.asm
vnziborova@fedora:~/work/arch-pc/lab05$ ld -m elf_i386 lab05-2.o -o lab05-2
vnziborova@fedora:~/work/arch-pc/lab05$ ./lab05-2
Введите строку:
Veronika
vnziborova@fedora:~/work/arch-pc/lab05$
```

Рис. 2.10: Запуск программы lab05-2.asm

В файле lab05-2.asm я заменила подпрограмму sprintLF на sprint. После пе-

ресборки исполняемого файла, теперь вывод строки не завершается символом новой строки.



```
mc [vnziborova@fedora]
mc [vnziborova@fedora]:~/work/arch-pc/lab05
lab05-2.asm [----] 11 L:[ 1+ 9 10/ 15] *(166 / 21
%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. 2.11: Обновленная программа в lab05-2.asm

```
vnziborova@fedora:~/work/arch-pc/lab05$  
vnziborova@fedora:~/work/arch-pc/lab05$ nasm -f elf lab05-2.asm  
vnziborova@fedora:~/work/arch-pc/lab05$ ld -m elf_i386 lab05-2.o -o lab05-2  
vnziborova@fedora:~/work/arch-pc/lab05$ ./lab05-2  
Введите строку: Veronika  
vnziborova@fedora:~/work/arch-pc/lab05$
```

Рис. 2.12: Запуск обновленной программы lab05-2.asm

2.3 Задание для самостоятельной работы

Я скопировала программу lab05-1.asm и изменила код так, чтобы он работал по следующему алгоритму:

- вывести приглашение с текстом “Введите строку:”;
- считать строку с клавиатуры;
- вывести введенную строку на экран.

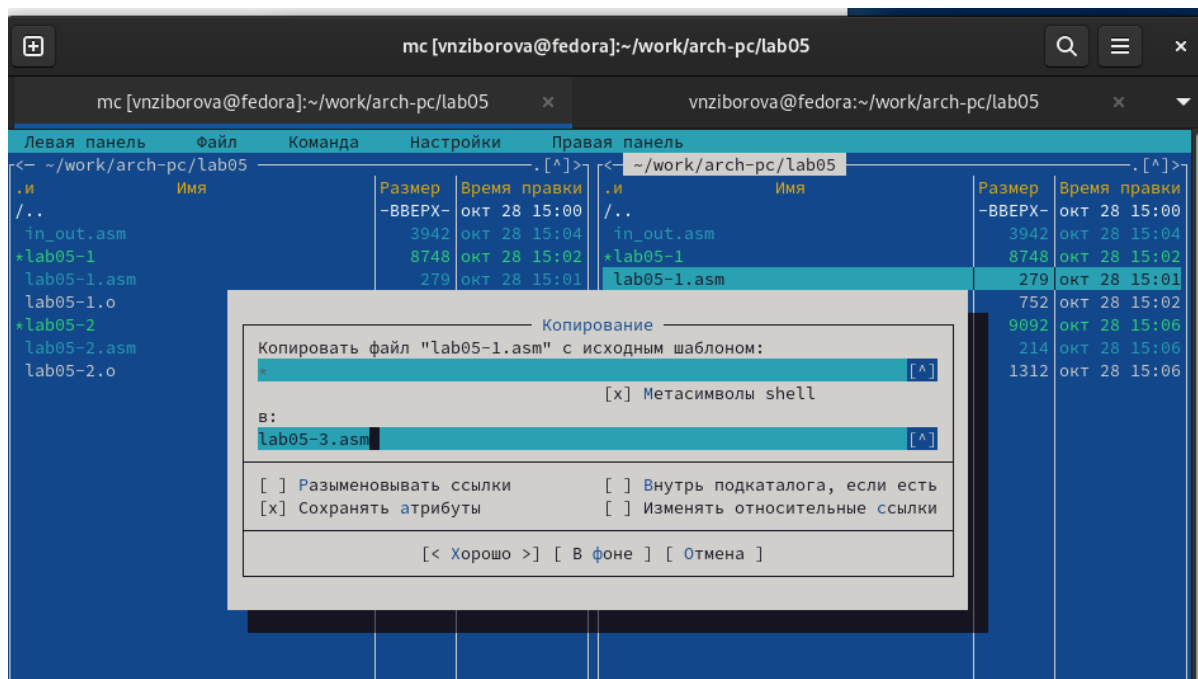
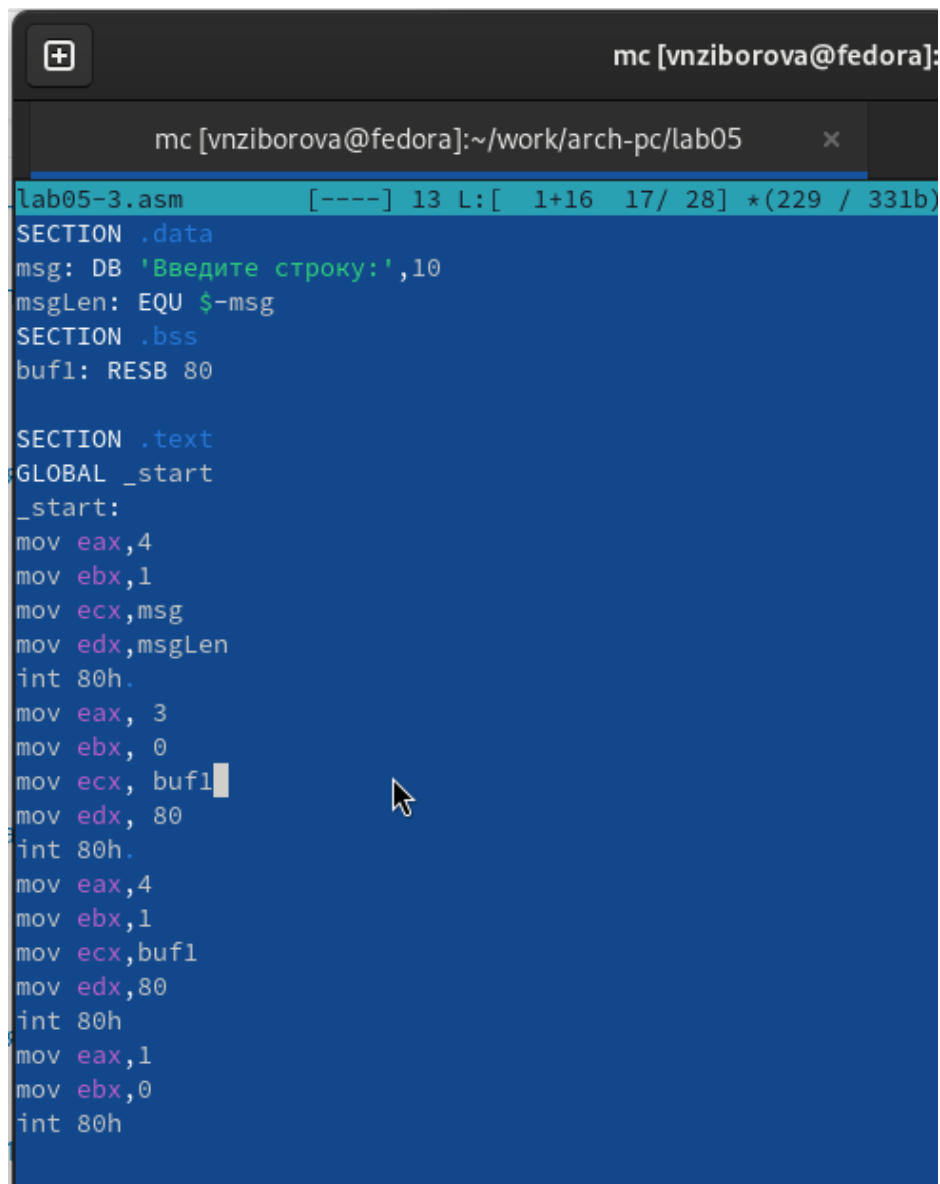
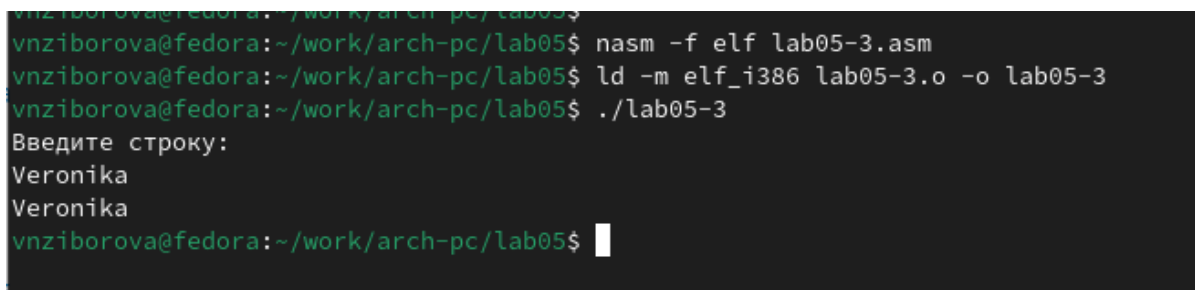


Рис. 2.13: Копирование файла lab05-1.asm для нового задания



```
mc [vnziborova@fedora]:  
mc [vnziborova@fedora]:~/work/arch-pc/lab05  
lab05-3.asm [----] 13 L: [ 1+16 17/ 28] *(229 / 331b)  
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,80  
int 80h  
mov eax,1  
mov ebx,0  
int 80h
```

Рис. 2.14: Код программы в файле lab05-3.asm



```
vnziborova@fedora:~/work/arch-pc/lab05$  
vnziborova@fedora:~/work/arch-pc/lab05$ nasm -f elf lab05-3.asm  
vnziborova@fedora:~/work/arch-pc/lab05$ ld -m elf_i386 lab05-3.o -o lab05-3  
vnziborova@fedora:~/work/arch-pc/lab05$ ./lab05-3  
Введите строку:  
Veronika  
Veronika  
vnziborova@fedora:~/work/arch-pc/lab05$
```

Рис. 2.15: Запуск программы lab05-3.asm

Аналогично, я скопировала программу lab05-2.asm и внесла изменения в код, теперь используя подпрограммы из файла in_out.asm.

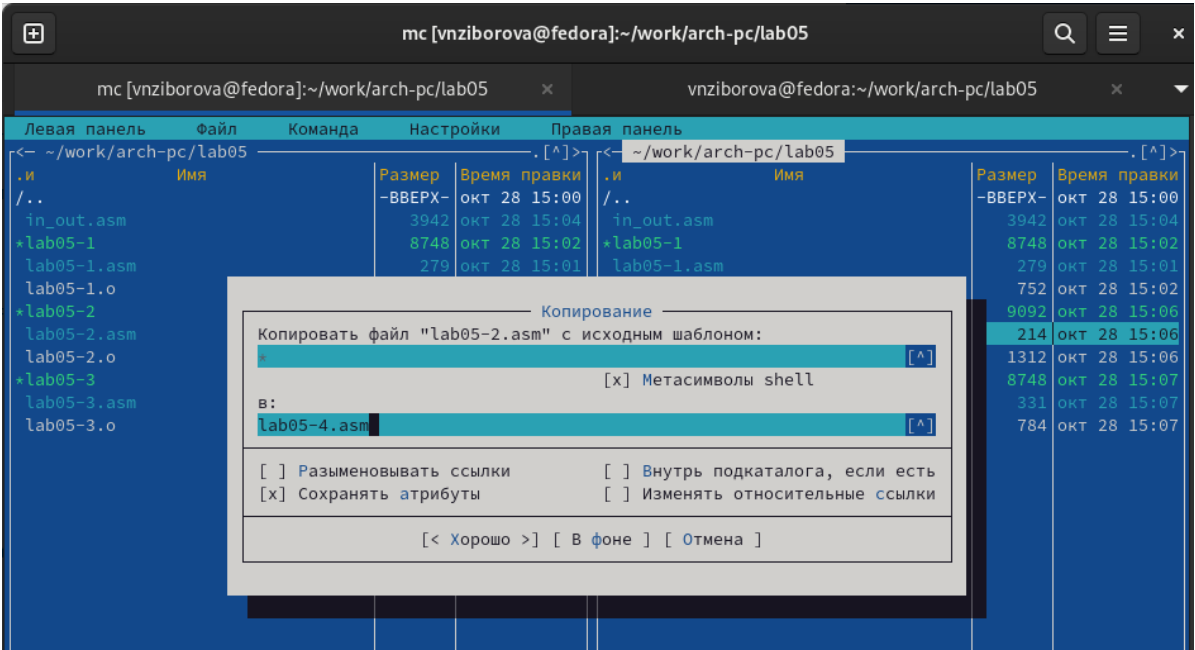
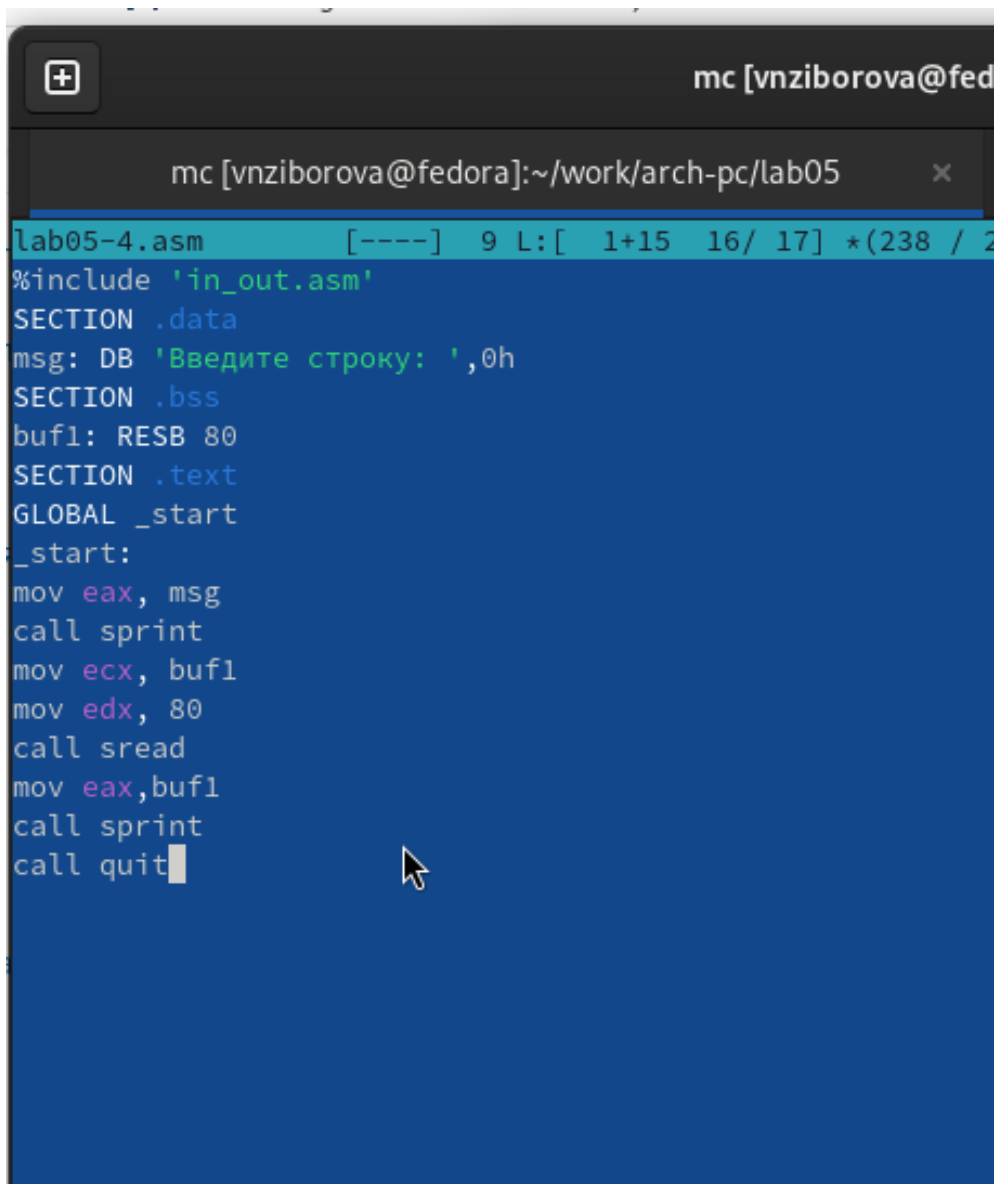


Рис. 2.16: Копирование файла lab05-2.asm для модификации



```
mc [vnziborova@fedora]:~/work/arch-pc/lab05
lab05-4.asm [----] 9 L:[ 1+15 16/ 17] *(238 / 2
#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, buf1
call sprint
call quit
```

Рис. 2.17: Код программы в файле lab05-4.asm



```
vnziborova@fedora:~/work/arch-pc/lab05$
vnziborova@fedora:~/work/arch-pc/lab05$ nasm -f elf lab05-4.asm
vnziborova@fedora:~/work/arch-pc/lab05$ ld -m elf_i386 lab05-4.o -o lab05-4
vnziborova@fedora:~/work/arch-pc/lab05$ ./lab05-4
Введите строку: Veronika
vnziborova@fedora:~/work/arch-pc/lab05$
```

Рис. 2.18: Запуск программы lab05-4.asm

3 Выводы

Я научилась писать базовые ассемблерные программы и освоила ассемблерные инструкции `mov` и `int`.

4 Вопросы для самопроверки

1. Каково назначение mc?

- Midnight Commander (mc) — это текстовый файловый менеджер, который позволяет пользователям управлять файлами и каталогами в командной строке. Он предоставляет удобный интерфейс для выполнения операций с файлами, таких как копирование, перемещение, удаление и просмотр.

2. Какие операции с файлами можно выполнить как с помощью команд bash, так и с помощью меню (комбинаций клавиш) mc? Приведите несколько примеров.

- В mc можно выполнять операции с файлами как с помощью команд bash, так и с помощью меню. Примеры:

– **Копирование файла:**

- * Команда bash: `cp файл1.txt файл2.txt`
- * В mc: выделить файл файл1.txt, нажать F5 и указать имя файл2.txt.

– **Перемещение файла:**

- * Команда bash: `mv файл1.txt папка/`
- * В mc: выделить файл файл1.txt, нажать F6 и указать путь папка/.

– **Удаление файла:**

- * Команда bash: `rm файл1.txt`
- * В mc: выделить файл файл1.txt, нажать F8.

3. Какова структура программы на языке ассемблера NASM?

- Программа на NASM состоит из трех основных секций:
 - **section .data** — секция для инициализации данных.
 - **section .bss** — секция для объявления неинициализированных данных.
 - **section .text** — секция, содержащая исполняемый код программы.

4. Для описания каких данных используются секции bss и data в языке ассемблера NASM?

- **Секция .data** используется для хранения инициализированных данных, которые имеют фиксированные значения.
- **Секция .bss** используется для объявления неинициализированных данных, которые будут выделены в памяти, но не содержат начальных значений.

5. Для чего используются компоненты db, dw, dd, dq и dt языка ассемблера NASM?

- Эти директивы используются для определения переменных различных типов:
 - **db** (define byte) — определяет байт.
 - **dw** (define word) — определяет слово (2 байта).
 - **dd** (define double word) — определяет двойное слово (4 байта).
 - **dq** (define quad word) — определяет квадратичное слово (8 байт).
 - **dt** (define ten bytes) — определяет десятибайтовое значение.

6. Какое произойдёт действие при выполнении инструкции mov eax, esi?

- Инструкция mov eax, esi копирует значение из регистра esi в регистр eax. Это означает, что после выполнения этой инструкции регистр eax будет содержать то же значение, что и esi.

7. Для чего используется инструкция int 80h?

- Инструкция `int 80h` используется для вызова системных функций в Linux. Она передает управление ядру операционной системы для выполнения системного вызова, например, для работы с файлами, сетевыми операциями и управления процессами.