

# **Отчёт по лабораторной работе 5**

**Архитектура компьютера**

Кази ар Рафи НКАбд-03-24

# Содержание

<b>1</b>	<b>Цель работы</b>	<b>5</b>
<b>2</b>	<b>Выполнение лабораторной работы</b>	<b>6</b>
2.1	Знакомство с Midnight Commander . . . . .	6
2.2	Подключение внешнего файла in_out.asm . . . . .	11
2.3	Задание для самостоятельной работы . . . . .	14
<b>3</b>	<b>Выводы</b>	<b>18</b>

## Список иллюстраций

2.1	Запуск Midnight Commander . . . . .	6
2.2	Создание каталога . . . . .	7
2.3	Создание файла lab05-1.asm . . . . .	7
2.4	Создание файла lab05-1.asm . . . . .	8
2.5	Программа lab05-1.asm . . . . .	9
2.6	Просмотр файла lab05-1.asm . . . . .	10
2.7	Запуск программы lab05-1.asm . . . . .	10
2.8	Копирование файла in_out.asm . . . . .	11
2.9	Копирование файла lab05-1.asm . . . . .	11
2.10	Программа lab05-2.asm . . . . .	12
2.11	Запуск программы lab05-2.asm . . . . .	12
2.12	Программа в файле lab05-2.asm . . . . .	13
2.13	Запуск программы lab05-2.asm . . . . .	13
2.14	Программа lab05-3.asm . . . . .	15
2.15	Запуск программы lab05-3.asm . . . . .	15
2.16	Программа lab05-4.asm . . . . .	16
2.17	Запуск программы lab05-4.asm . . . . .	17

## **Список таблиц**

# 1 Цель работы

Целью работы является приобретение практических навыков работы в Midnight Commander. Освоение инструкций языка ассемблера `mov` и `int`.

## 2 Выполнение лабораторной работы

### 2.1 Знакомство с Midnight Commander

Я открываю Midnight Commander (рис. 2.1) и с помощью клавиш со стрелками и Enter перехожу в каталог ~/work/arch-rc. Затем нажимаю F7, чтобы создать новый каталог lab05 (рис. 2.2).

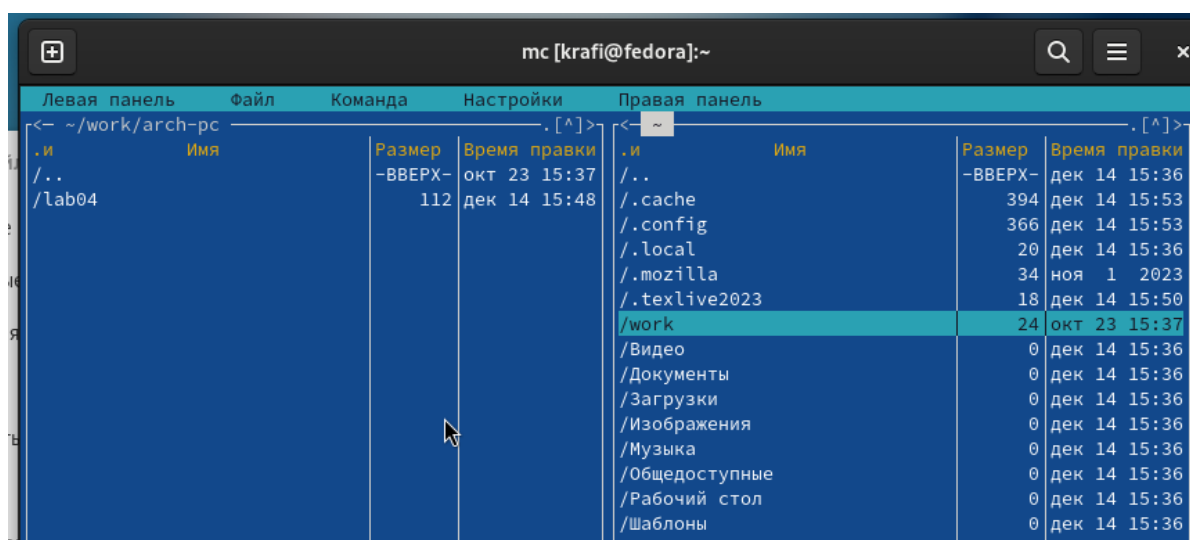


Рис. 2.1: Запуск Midnight Commander

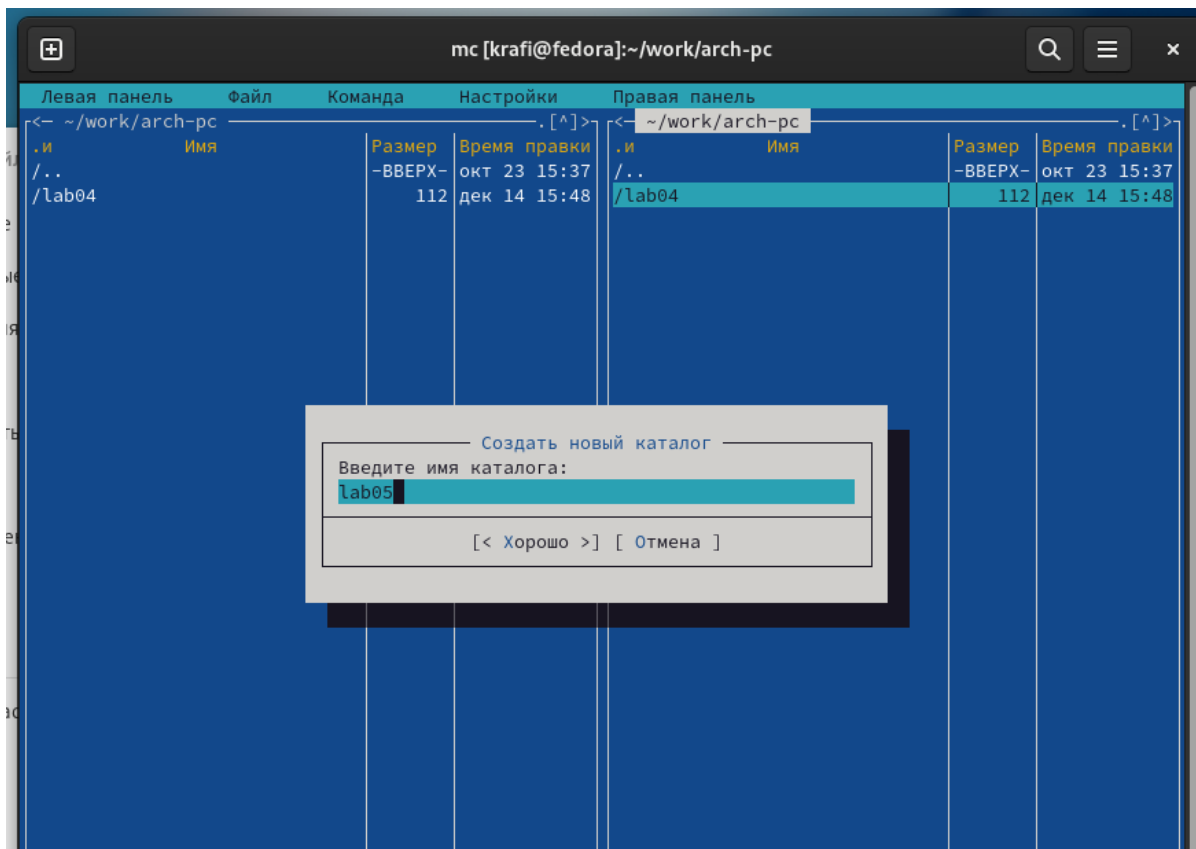


Рис. 2.2: Создание каталога

Используя команду touch, создаю файл lab05-1.asm (рис. 2.3).

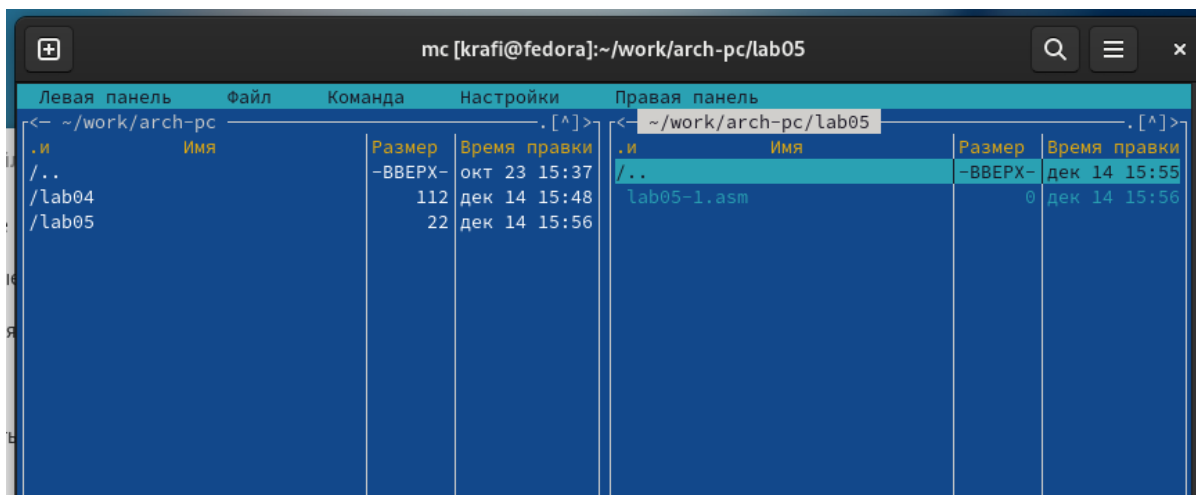


Рис. 2.3: Создание файла lab05-1.asm

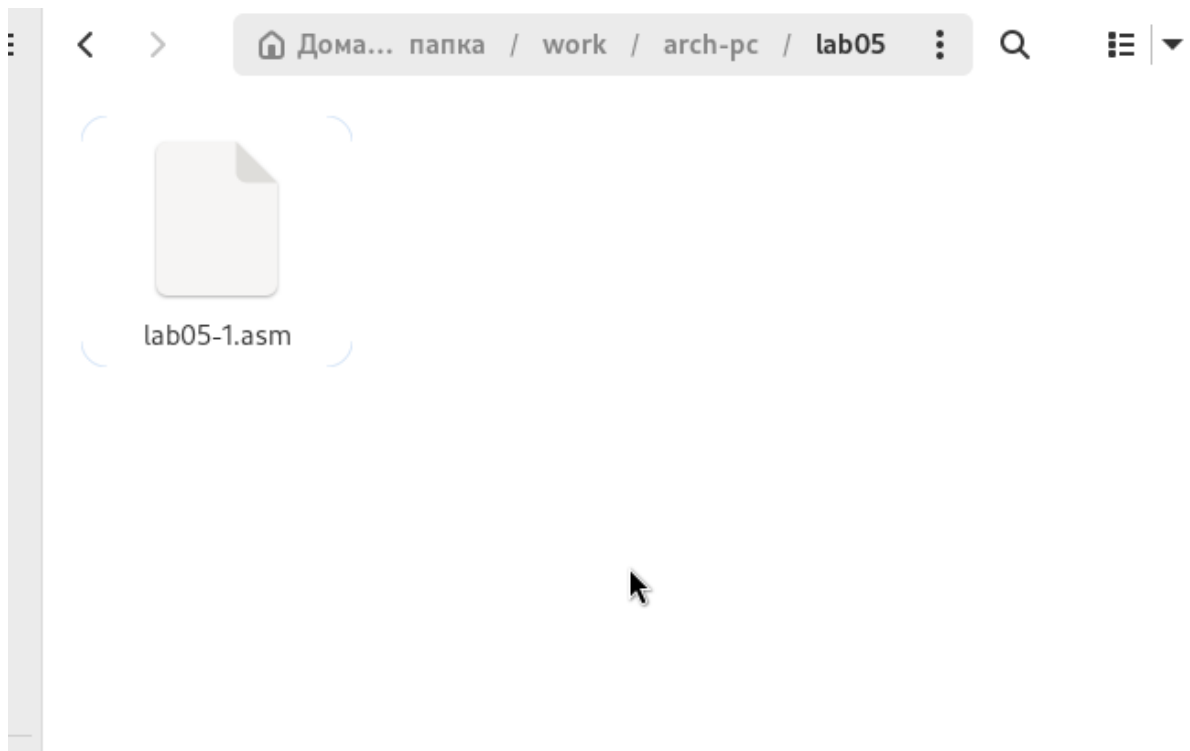
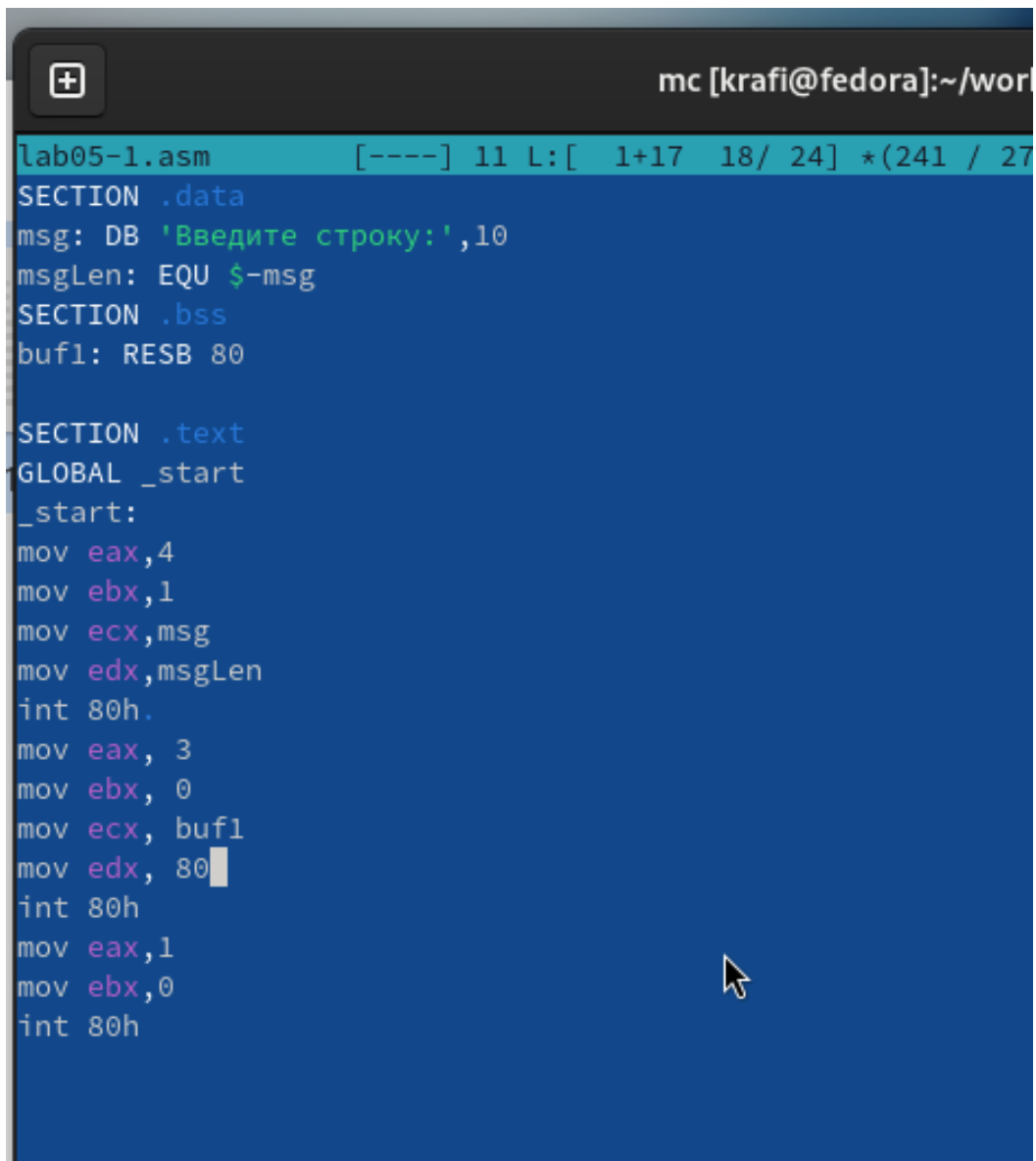


Рис. 2.4: Создание файла lab05-1.asm

Открываю файл на редактирование, нажав клавишу F4. Выбираю редактор mscedit и пишу код программы согласно заданию (рис. 2.5).



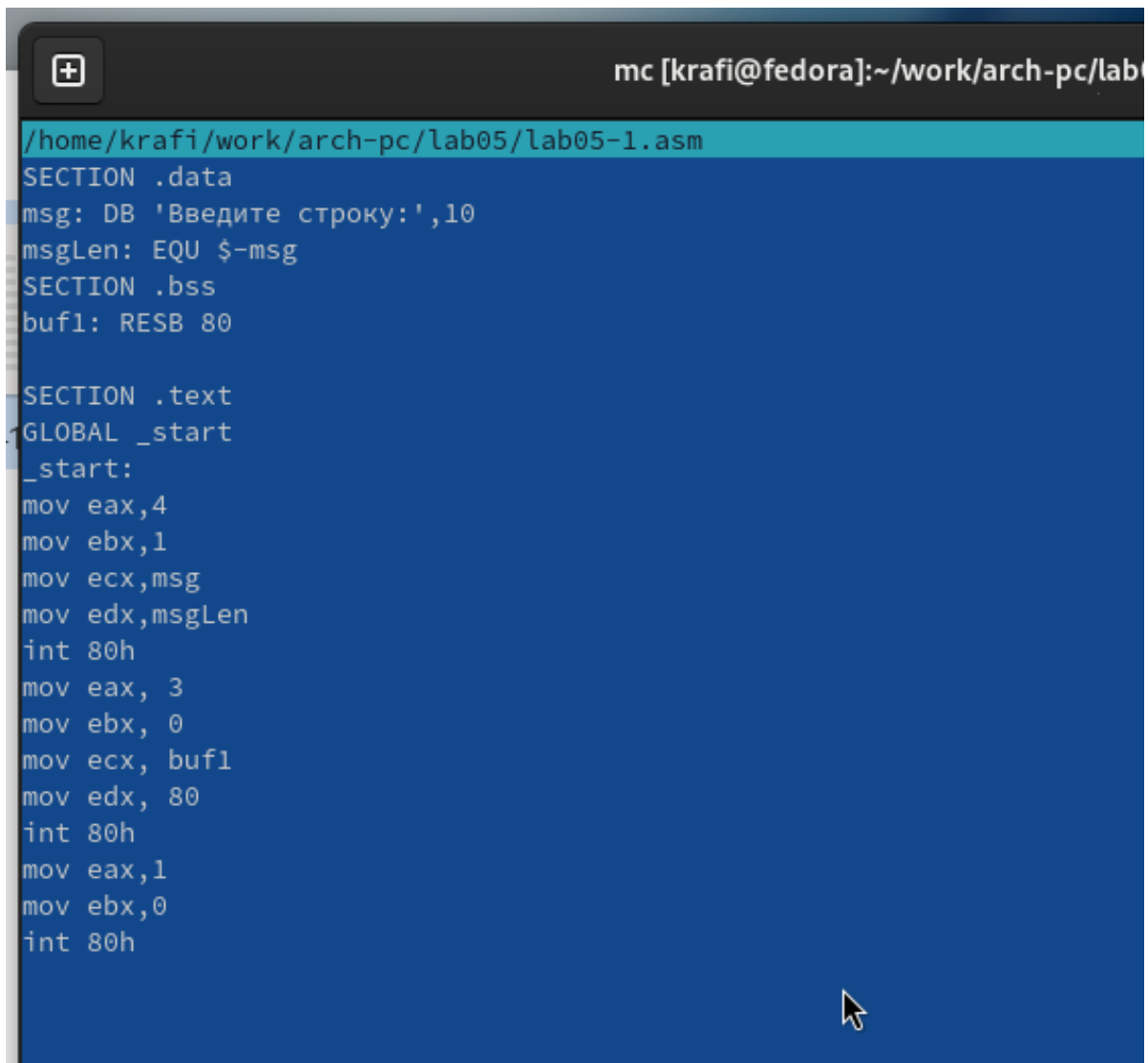


```
mc [krafi@fedora]:~/wor
lab05-1.asm [----] 11 L: [ 1+17 18/ 24] *(241 / 27
SECTION .data
msg: DB 'Введите строку:',10
msgLen: EQU $-msg
SECTION .bss
buf1: RESB 80

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: Программа lab05-1.asm

Для проверки кода открываю файл на просмотр, нажав клавишу F3, и убеждаюсь, что он содержит необходимый текст (рис. 2.6).

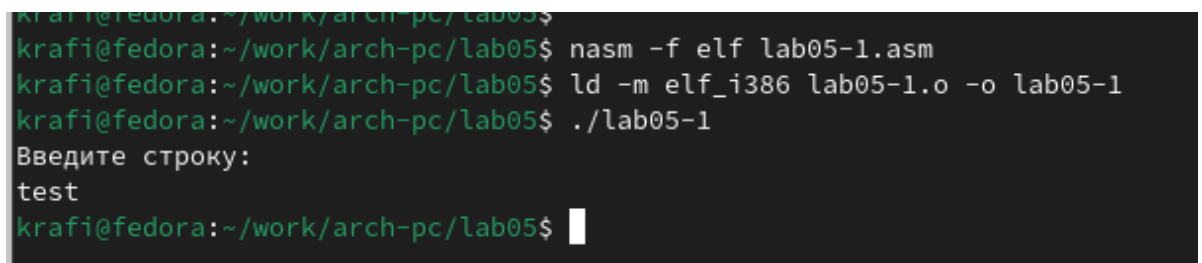


```
mc [krafi@fedora]:~/work/arch-pc/lab05
/home/krafi/work/arch-pc/lab05/lab05-1.asm
SECTION .data
msg: DB 'Введите строку:',10
msgLen: EQU $-msg
SECTION .bss
buf1: RESB 80

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. 2.6: Просмотр файла lab05-1.asm

Затем я компилирую файл программы в объектный файл, выполняю компоновку объектного файла и получаю исполняемый файл программы (рис. 2.7).



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

Рис. 2.7: Запуск программы lab05-1.asm

## 2.2 Подключение внешнего файла in\_out.asm

Скачиваю файл in\_out.asm и помещаю его в рабочий каталог (рис. 2.8). Для копирования файла использую клавишу F5, а для перемещения — клавишу F6.

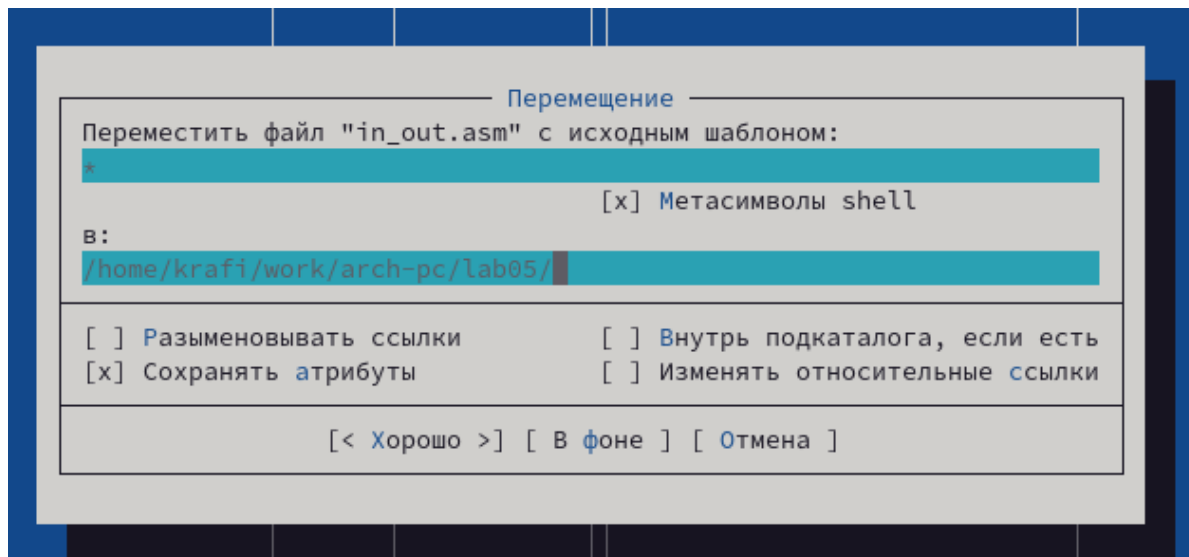


Рис. 2.8: Копирование файла in\_out.asm

Затем я копирую lab05-1.asm в lab05-2.asm (рис. 2.9).

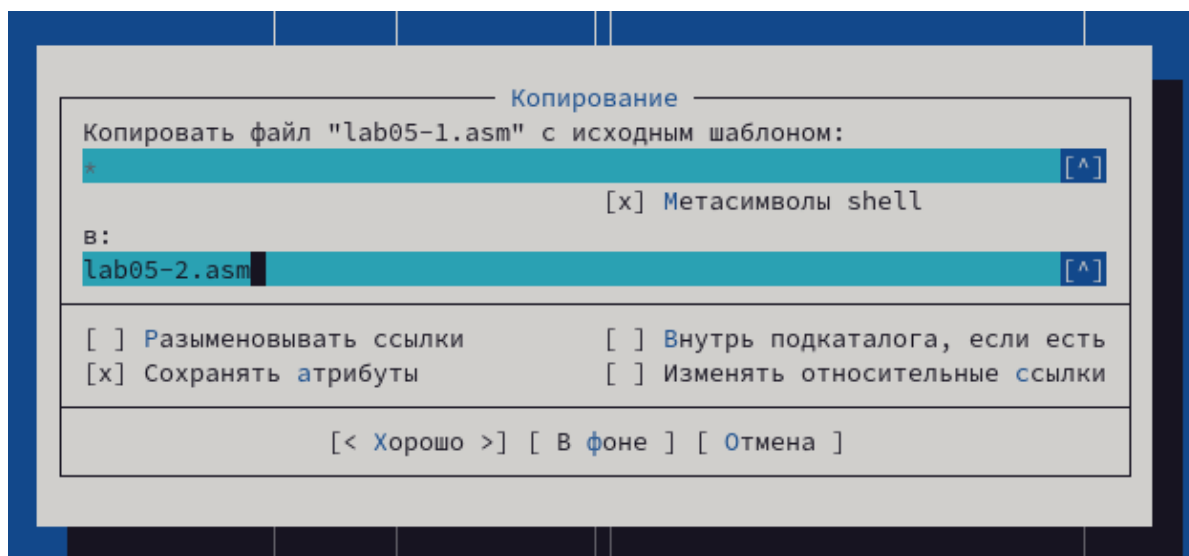
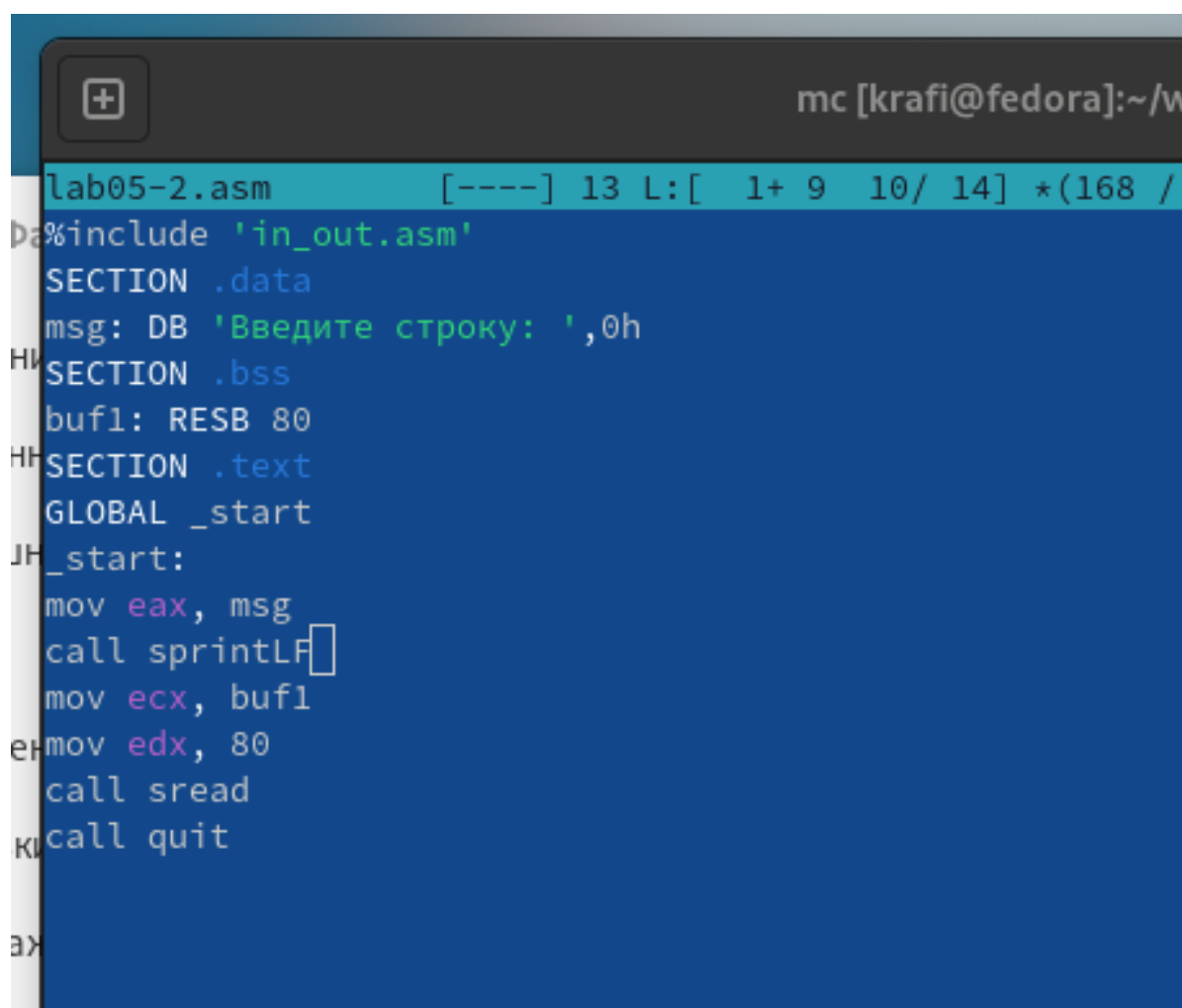


Рис. 2.9: Копирование файла lab05-1.asm

В новом файле lab05-2.asm пишу код программы с использованием подпро-


грамм из внешнего файла in\_out.asm (рис. 2.10).



```
lab05-2.asm [----] 13 L: [ 1+ 9 10/ 14] *(168 /
%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. 2.10: Программа lab05-2.asm

После компиляции программы я проверяю её запуск (рис. 2.11).

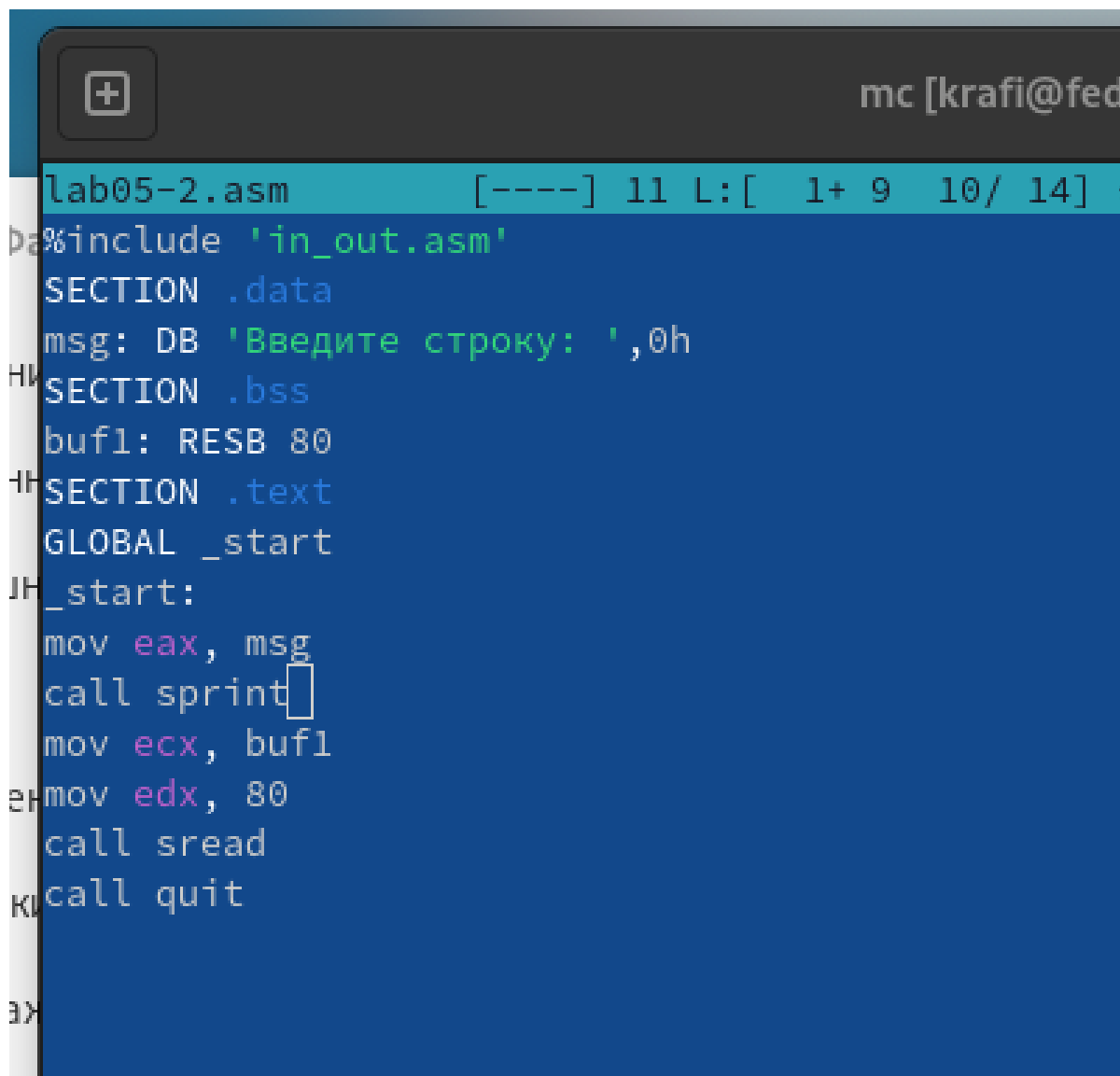


```
krafi@fedora:~/work/arch-pc/lab05$ nasm -f elf lab05-2.asm
krafi@fedora:~/work/arch-pc/lab05$ ld -m elf_i386 lab05-2.o -o lab05-2
krafi@fedora:~/work/arch-pc/lab05$ ./lab05-2
Введите строку:
test
krafi@fedora:~/work/arch-pc/lab05$
```

Рис. 2.11: Запуск программы lab05-2.asm

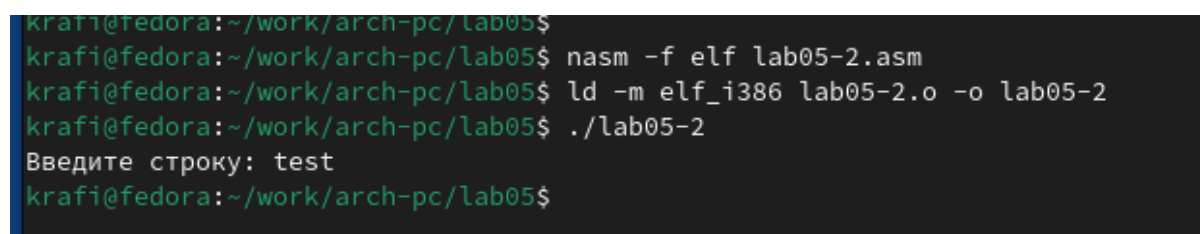
В файле lab05-2.asm я заменяю подпрограмму sprintLF на sprint, после чего

заново собираю исполняемый файл (рис. 2.12) (рис. 2.13).



```
lab05-2.asm [-----] 11 L:[ 1+ 9 10/ 14]
%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. 2.12: Программа в файле lab05-2.asm



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

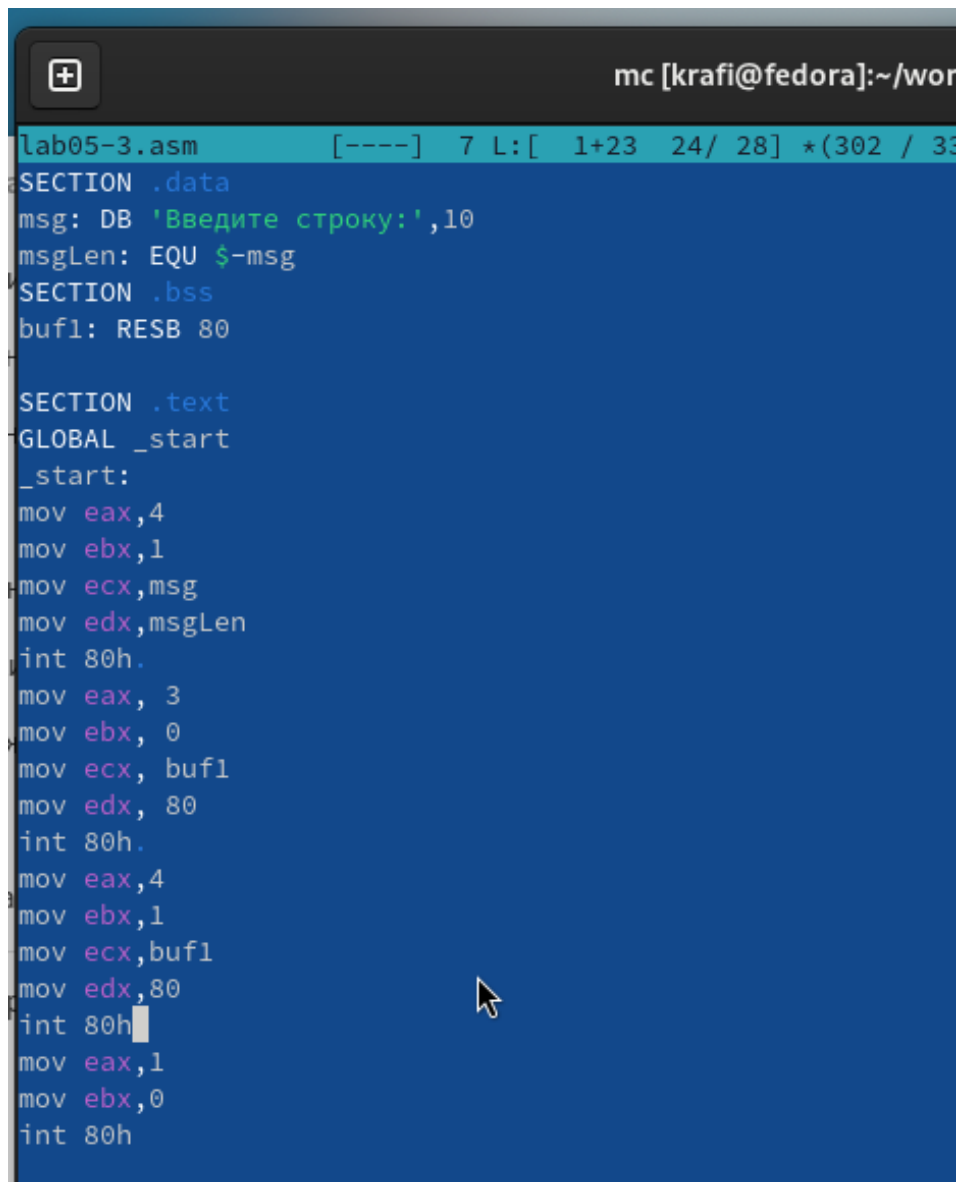
Рис. 2.13: Запуск программы lab05-2.asm

Теперь после вывода строки программа не завершается символом перехода на новую строку.

## **2.3 Задание для самостоятельной работы**

Я скопировал программу lab05-1.asm и изменил код так, чтобы она работала по следующему алгоритму: (рис. 2.14) (рис. 2.15)

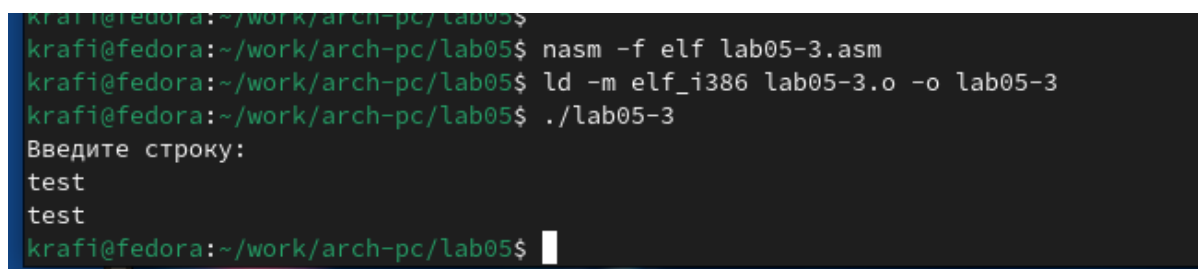
- вывести приглашение типа “Введите строку:”;
- ввести строку с клавиатуры;
- вывести введённую строку на экран.



```
mc [krafi@fedora]:~/work
lab05-3.asm [----] 7 L: [ 1+23 24/ 28] *(302 / 33
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
```

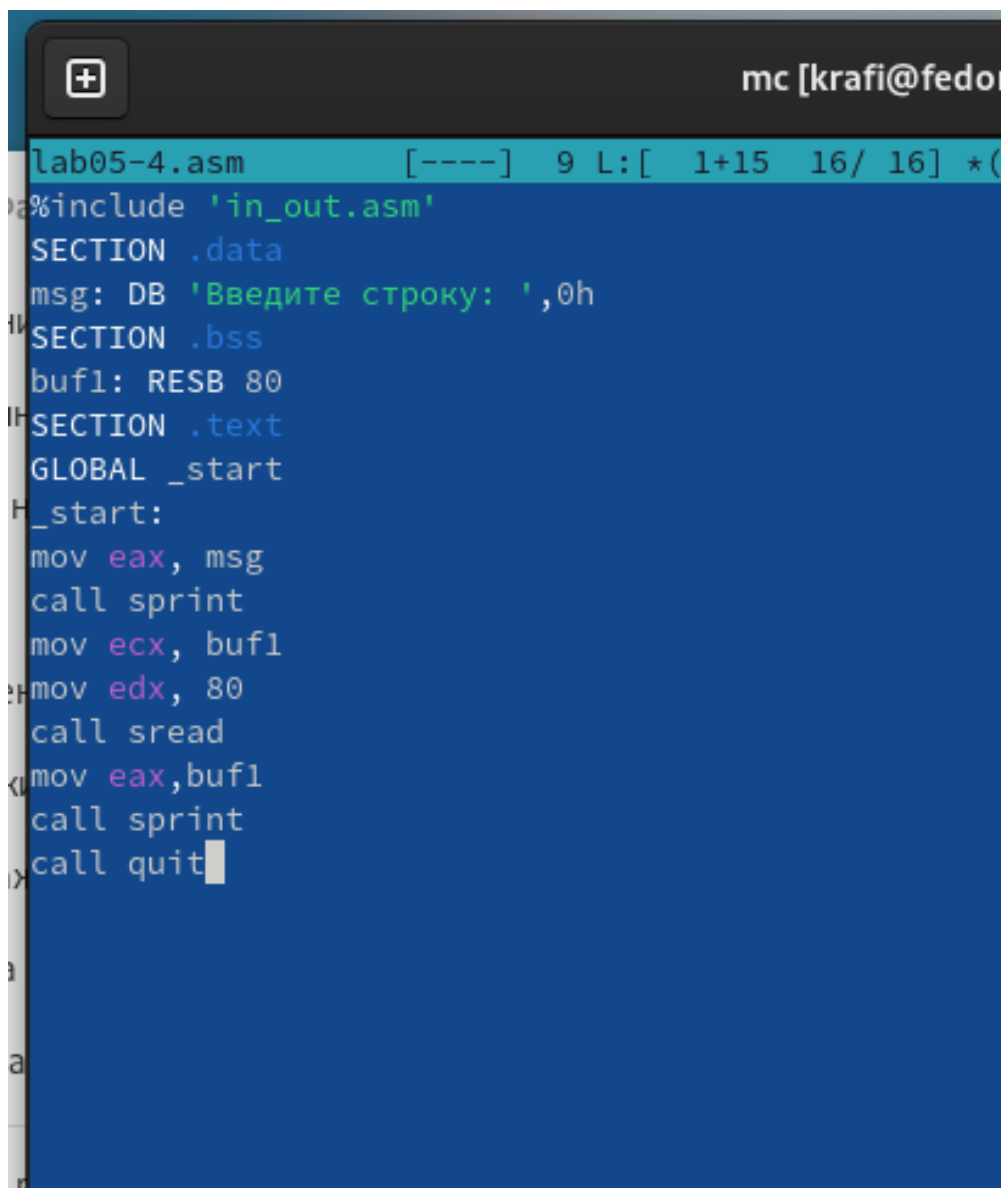
Рис. 2.14: Программа lab05-3.asm



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

Рис. 2.15: Запуск программы lab05-3.asm

Аналогичным образом я скопировал программу lab05-2.asm и изменил код, но теперь использовал подпрограммы из файла in\_out.asm (рис. 2.16) (рис. 2.17).



```
lab05-4.asm [----] 9 L:[ 1+15 16/ 16] *(
%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 sprintf
mov ecx, buf1
mov edx, 80
call sread
mov eax, buf1
call sprintf
call quit
```

Рис. 2.16: Программа lab05-4.asm



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

Рис. 2.17: Запуск программы lab05-4.asm

## 3 Выводы

Научились писать базовые ассемблерные программы. Освоили ассемблерные инструкции `mov` и `int`.