

Отчёт по лабораторной работе 5

Основы работы с Midnight Commander (mc). Структура программы на языке ассемблера NASM

Газибагандов Шейхахмед Арсенович

Содержание

1	Цель работы	5
2	Теоретическое введение	6
3	Выполнение лабораторной работы	7
3.1	Знакомство с Midnight Commander	7
3.2	Подключение внешнего файла in_out.asm	11
3.3	Задание для самостоятельной работы	16
4	Выводы	20

Список иллюстраций

3.1	Запуск Midnight Commander	7
3.2	Создание каталога	8
3.3	Создание файла lab05-1.asm	8
3.4	Программа в файле lab05-1.asm	9
3.5	Просмотр файла lab05-1.asm	10
3.6	Запуск программы lab05-1.asm	11
3.7	Копирование файла in_out.asm	12
3.8	Копирование файла lab05-1.asm	13
3.9	Программа в файле lab05-2.asm	14
3.10	Запуск программы lab05-2.asm	14
3.11	Программа в файле lab05-2.asm	15
3.12	Запуск программы lab05-2.asm	15
3.13	Копирование файла lab05-1.asm	16
3.14	Программа в файле lab05-3.asm	17
3.15	Запуск программы lab05-3.asm	17
3.16	Копирование файла lab05-2.asm	18
3.17	Программа в файле lab05-4.asm	19
3.18	Запуск программы lab05-4.asm	19

Список таблиц

1 Цель работы

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

2 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).

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

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

3 Выполнение лабораторной работы

3.1 Знакомство с Midnight Commander

Открыл Midnight Commander, с помощью клавишь со стрелками и Enter перешел в каталог ~/work/arch-pc. Далее нажал F7 и создал каталог lab05

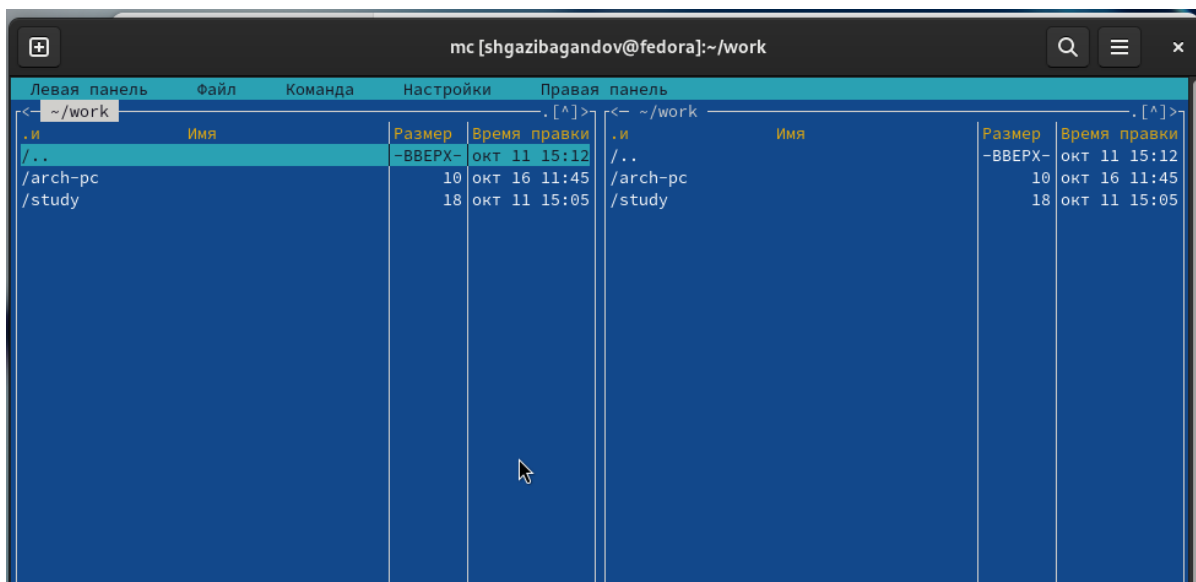


Рис. 3.1: Запуск Midnight Commander

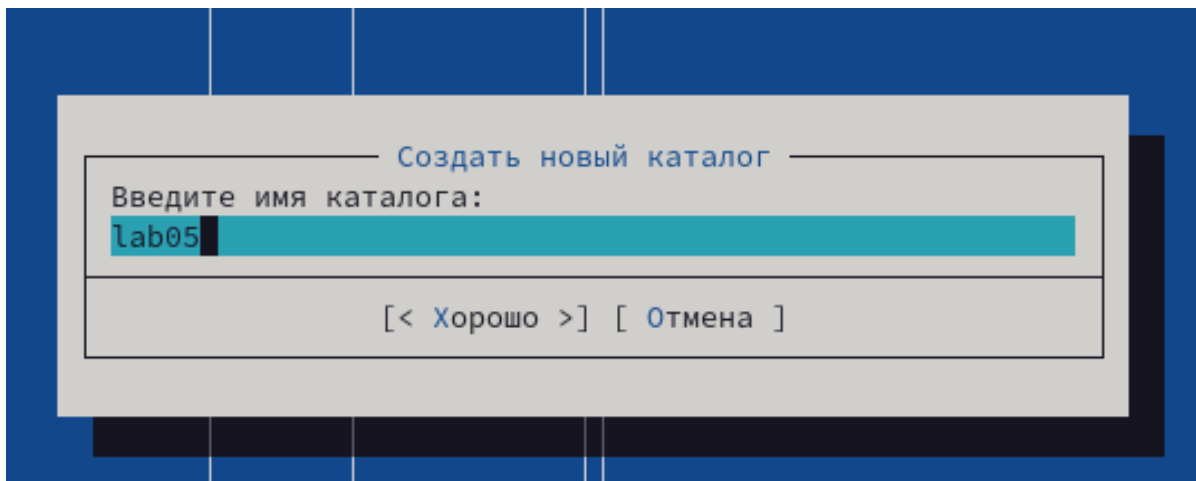


Рис. 3.2: Создание каталога

При помощи touch создал файл lab05-1.asm

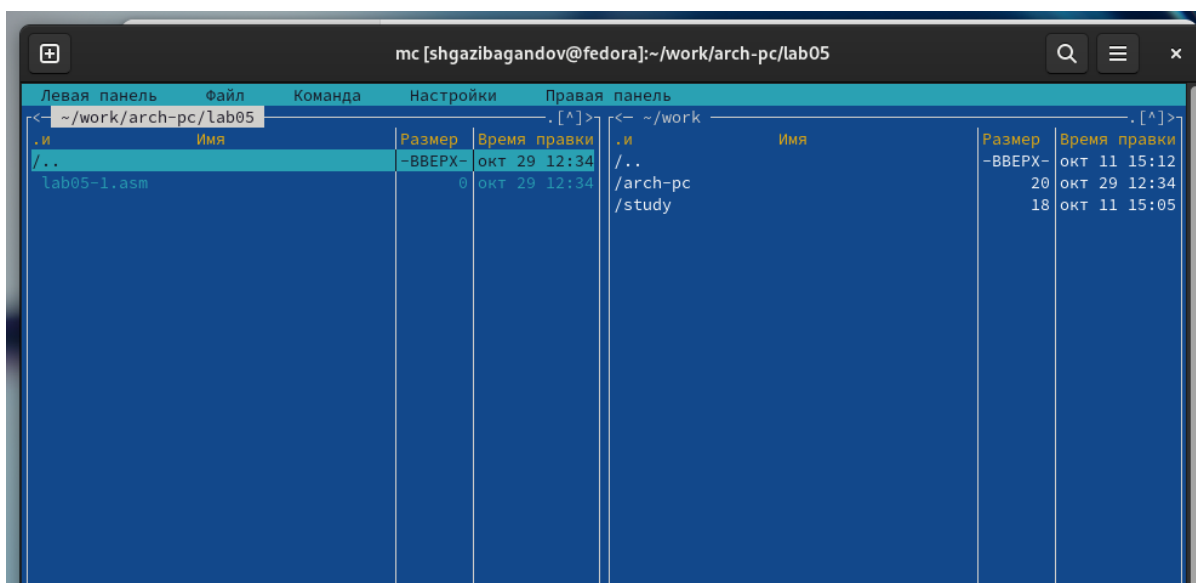
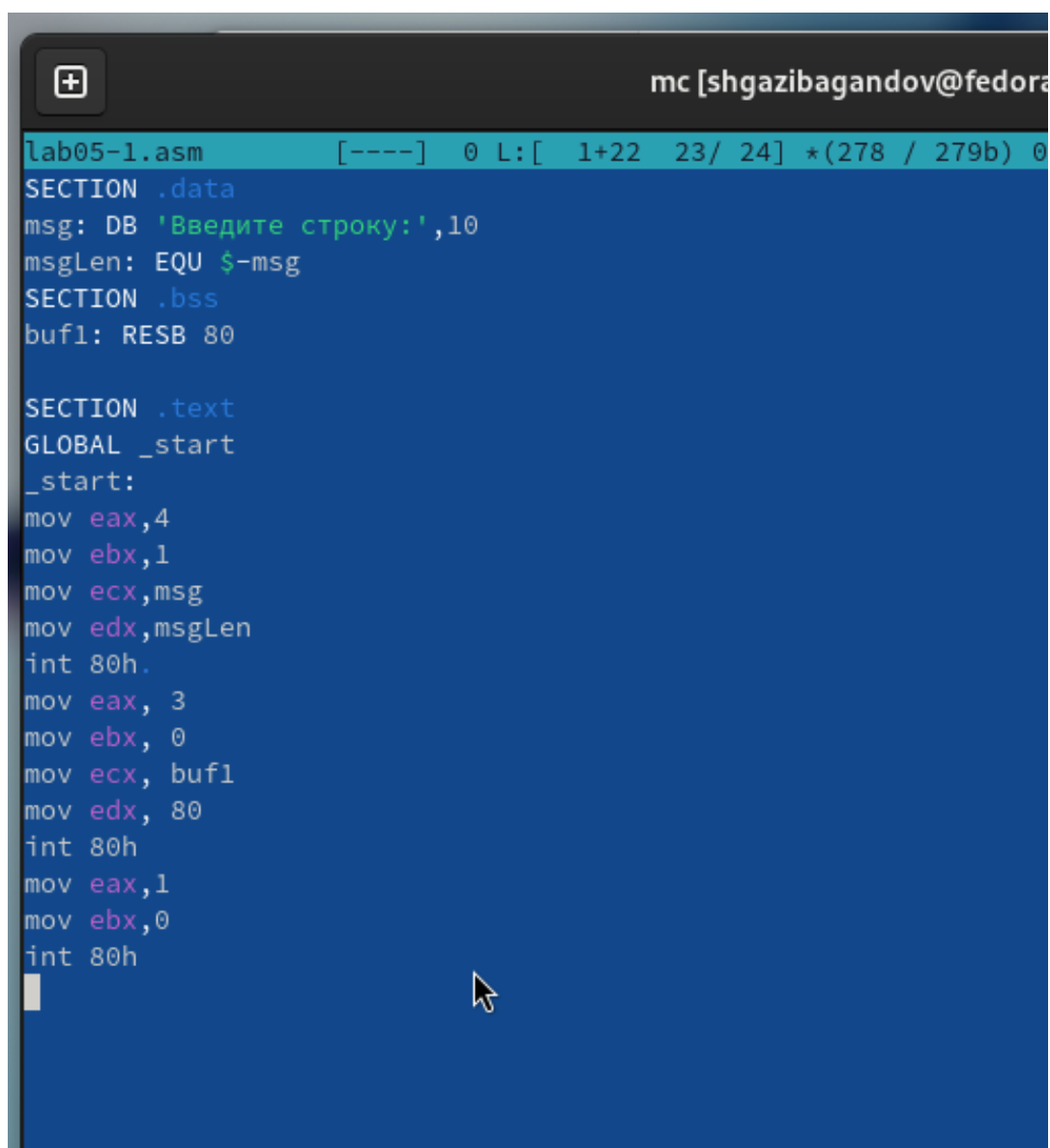


Рис. 3.3: Создание файла lab05-1.asm

Открыл файл на редактирование клавишей F4, выбрал редактор mceditor, написал код программы из задания.

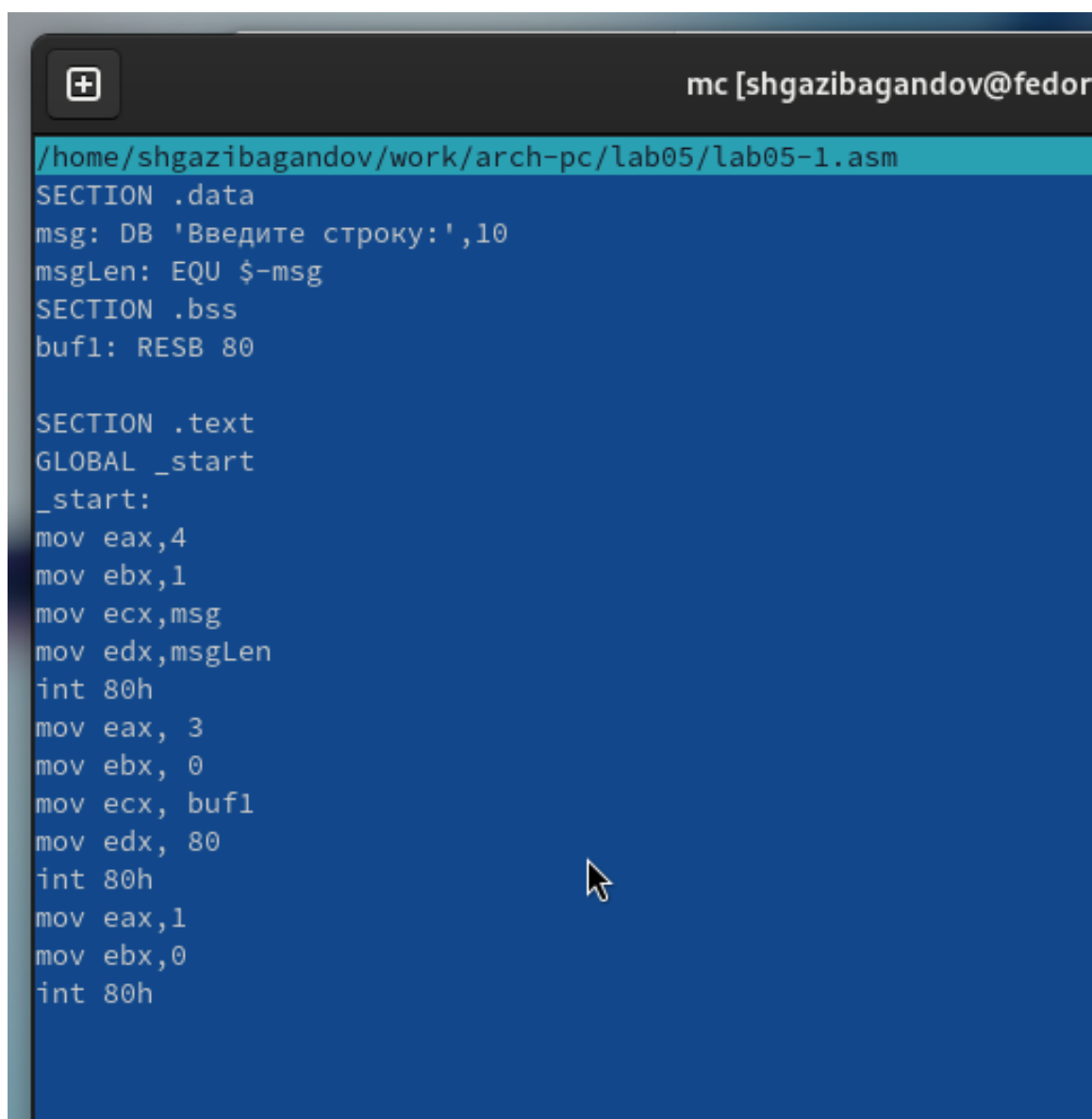


```
lab05-1.asm [----] 0 L: [ 1+22 23/ 24] *(278 / 279b) 0
SECTION .data
msg: DB 'Введите строку:',10
msgLen: EQU $-msg
SECTION .bss
buf1: RESB 80

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. 3.4: Программа в файле lab05-1.asm

Открыл файл на просмотр клавишей F3 и убелился, что он содержит набранный код.



```
mc [shgazibagandov@fedor...]  
/home/shgazibagandov/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. 3.5: Просмотр файла lab05-1.asm

Транслировал файл программы в объектный файл, выполнил компоновку объектного файла, получил исполняемый файл программы и проверил ее работу.

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

Рис. 3.6: Запуск программы lab05-1.asm

3.2 Подключение внешнего файла in_out.asm

Для упрощения написания программ часто встречающиеся одинаковые участки кода (такие как, например, вывод строки на экран или выход из программы) можно оформить в виде подпрограмм и сохранить в отдельные файлы, а во всех нужных местах поставить вызов нужной подпрограммы. Это позволяет сделать основную программу более удобной для написания и чтения.

Для выполнения лабораторных работ используется файл in_out.asm, который содержит следующие подпрограммы:

- `slen` – вычисление длины строки (используется в подпрограммах печати сообщения для определения количества выводимых байтов);
- `sprint` – вывод сообщения на экран, перед вызовом `sprint` в регистр `eax` необходимо записать выводимое сообщение (`mov eax,;`);
- `sprintLF` – работает аналогично `sprint`, но при выводе на экран добавляет к сообщению символ перевода строки;
- `sread` – ввод сообщения с клавиатуры, перед вызовом `sread` в регистр `eax` необходимо записать адрес переменной в которую введенное сообщение буд записано (`mov eax,;`), в регистр `ebx` – длину вводимой строки (`mov ebx,;`);
- `iprint` – вывод на экран чисел в формате ASCII, перед вызовом `iprint` в регистр `eax` необходимо записать выводимое число (`mov eax,;`);

- `iprintLF` – работает аналогично `iprint`, но при выводе на экран после числа добавляет к символ перевода строки;
- `atoi` – функция преобразует `ascii`-код символа в целое число и записывает результат в регистр `eax`, перед вызовом `atoi` в регистр `eax` необходимо записать число (`mov eax,;`);
- `quit` – завершение программы.

Скачал файл `in_out.asm` и разместил его в рабочем каталоге. Для копирования используется клавиша F5. Для перемещения используется клавиша F6.

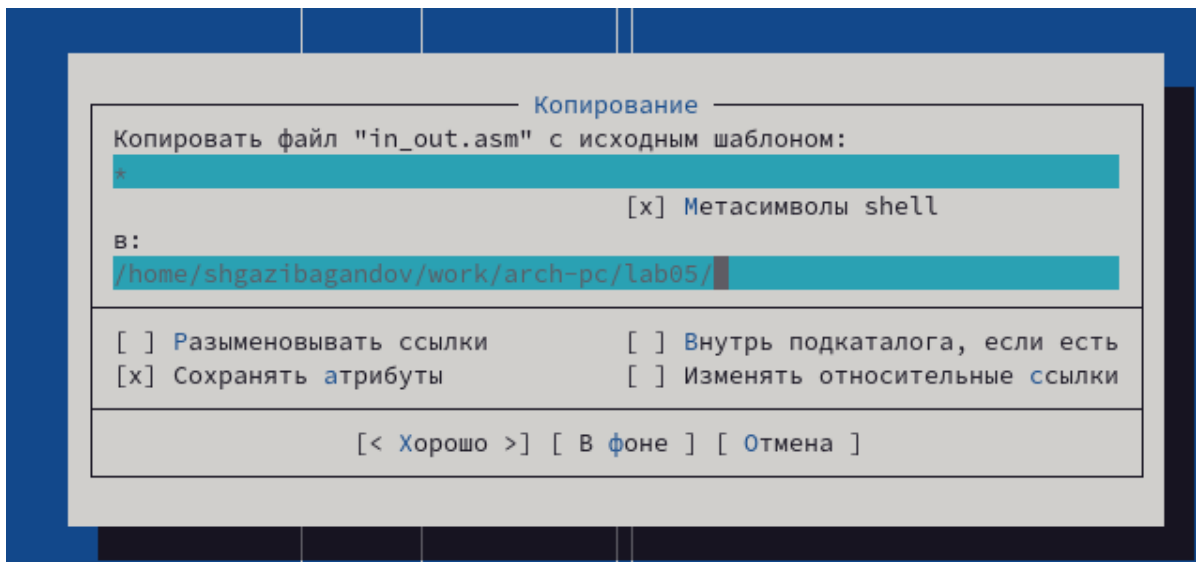


Рис. 3.7: Копирование файла `in_out.asm`

Скопировал `lab05-1.asm` в `lab05-2.asm`.

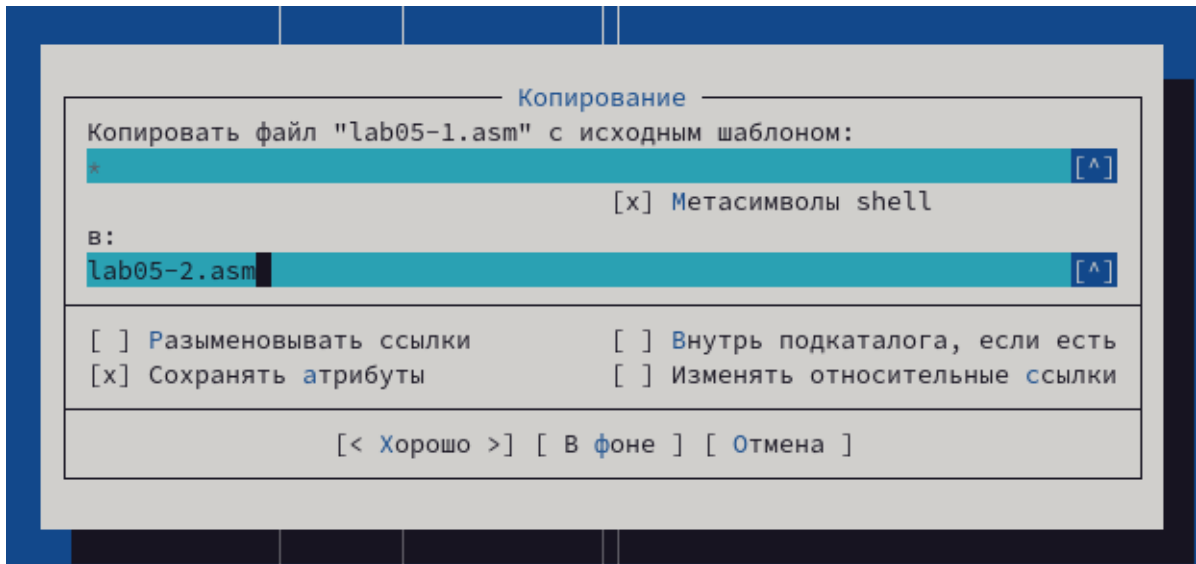
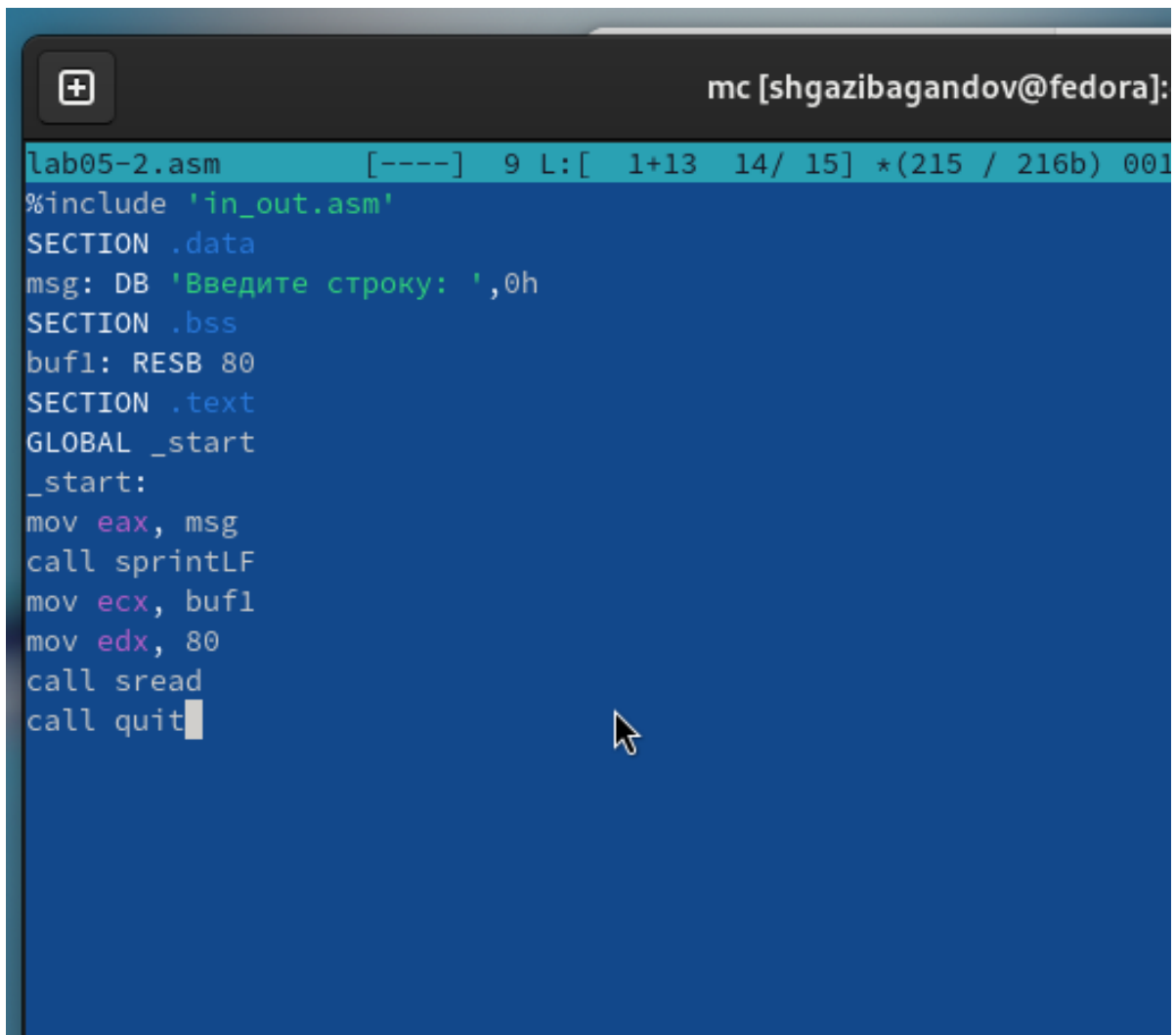


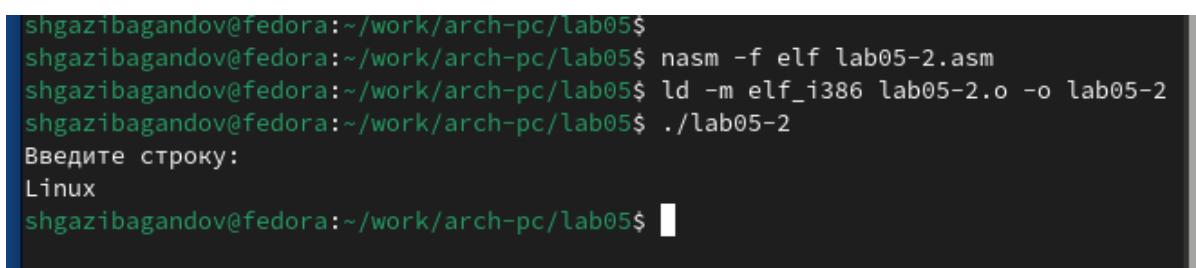
Рис. 3.8: Копирование файла lab05-1.asm

Написал код программы lab05-2.asm с использованием подпрограмм из внешнего файла in_out.asm . Скомпилировал программу и проверил запуск.



```
lab05-2.asm [----] 9 L:[ 1+13 14/ 15] *(215 / 216b) 001
#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. 3.9: Программа в файле lab05-2.asm

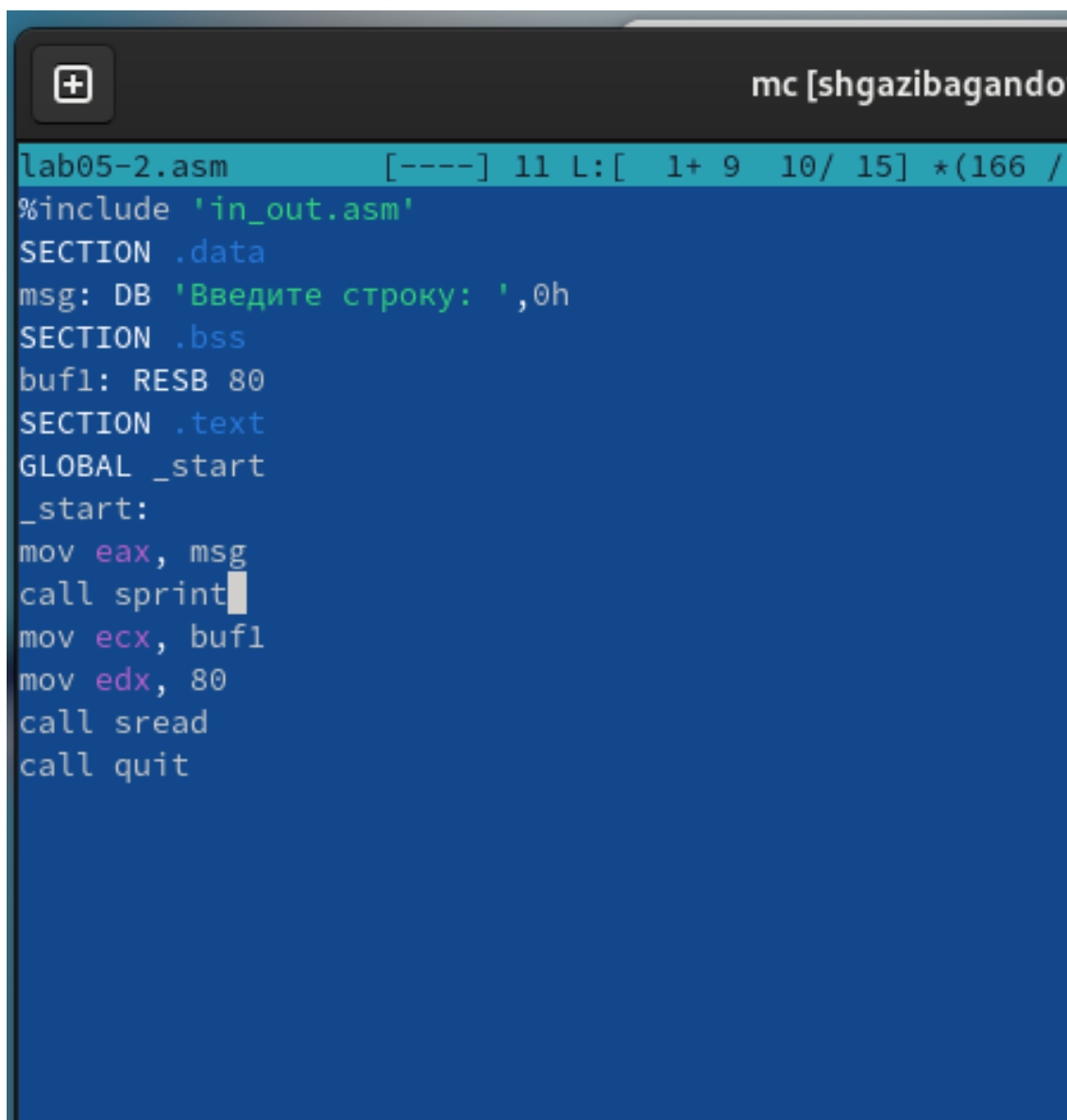


```
shgazibagandov@fedora:~/work/arch-pc/lab05$
shgazibagandov@fedora:~/work/arch-pc/lab05$ nasm -f elf lab05-2.asm
shgazibagandov@fedora:~/work/arch-pc/lab05$ ld -m elf_i386 lab05-2.o -o lab05-2
shgazibagandov@fedora:~/work/arch-pc/lab05$ ./lab05-2
Введите строку:
Linux
shgazibagandov@fedora:~/work/arch-pc/lab05$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

Рис. 3.10: Запуск программы lab05-2.asm

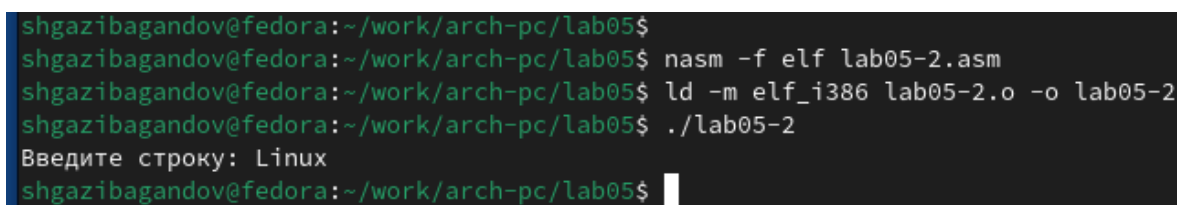
В файле lab5-2.asm заменил подпрограмму sprintLF на sprint. Заново собрал исполняемый файл. Теперь после вывода строки она не завершается символом

перехода на новую строку.



```
lab05-2.asm [----] 11 L: [ 1+ 9 10/ 15] *(166 /
#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. 3.11: Программа в файле lab05-2.asm



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

Рис. 3.12: Запуск программы lab05-2.asm

3.3 Задание для самостоятельной работы

Скопировал программу lab05-1.asm и изменил код, так чтобы она работала по следующему алгоритму:

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

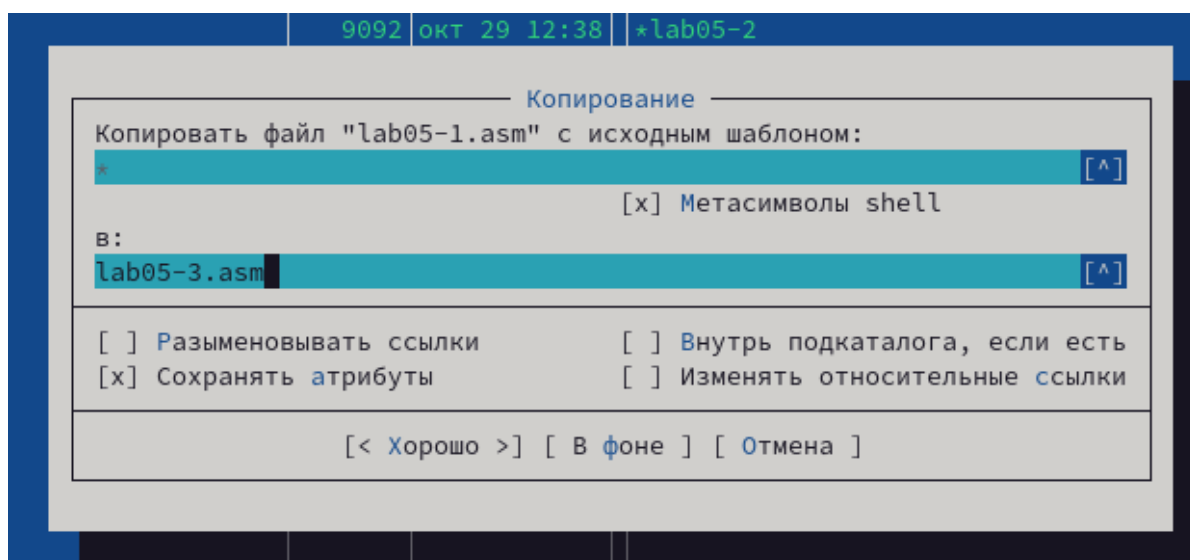
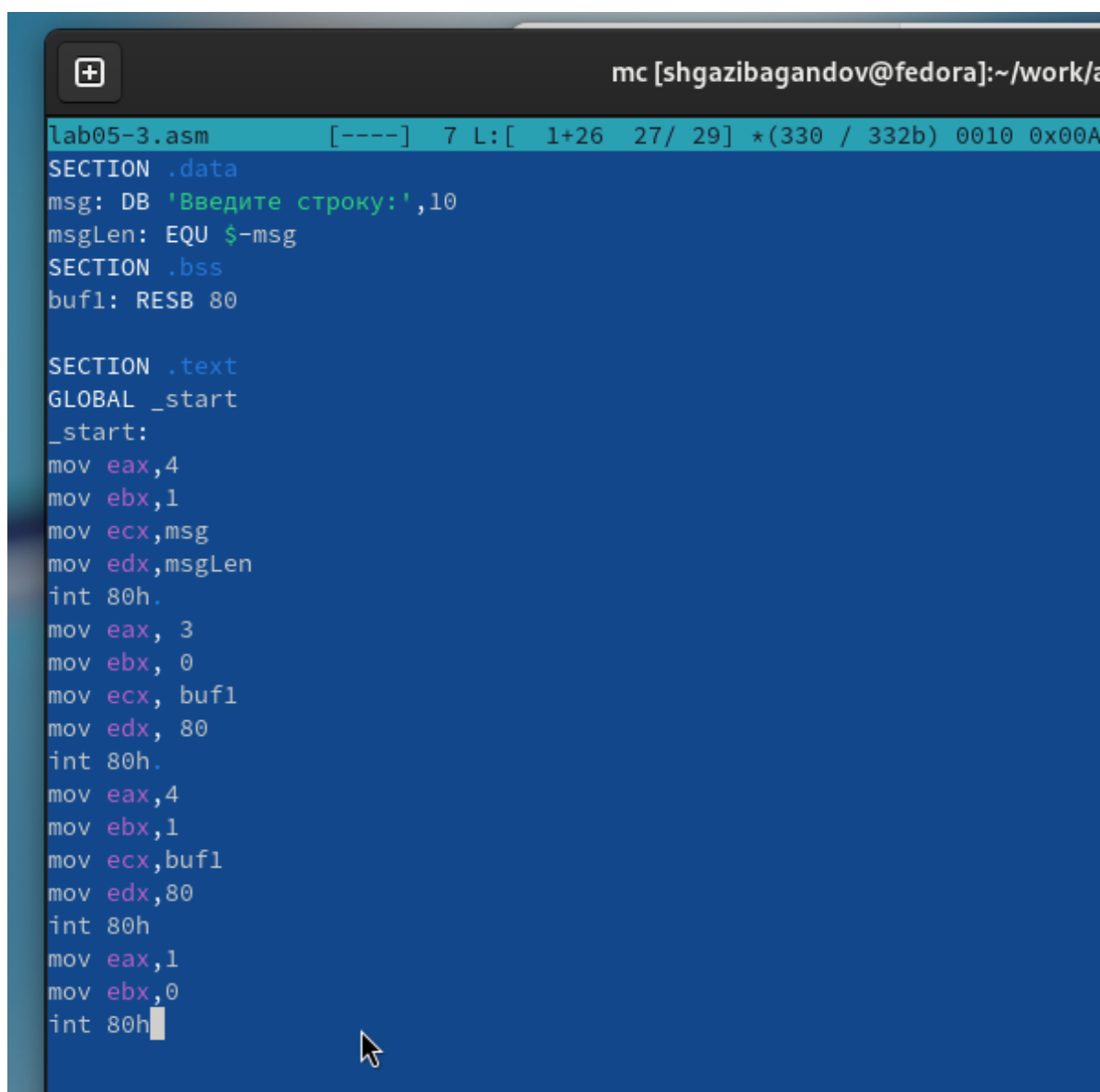


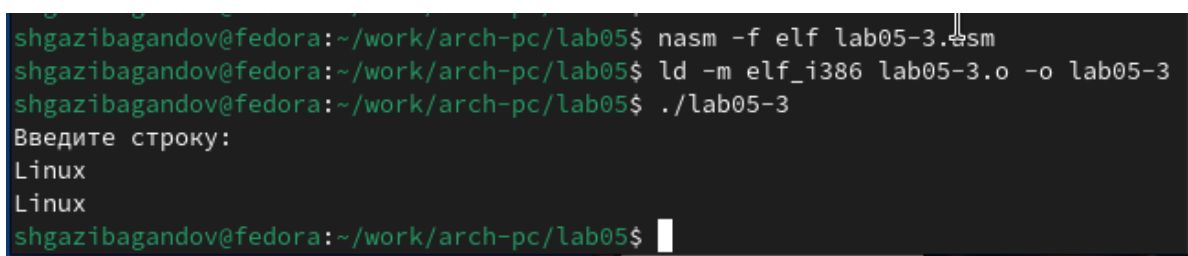
Рис. 3.13: Копирование файла lab05-1.asm



```
lab05-3.asm [----] 7 L: [ 1+26 27/ 29] *(330 / 332b) 0010 0x00A
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. 3.14: Программа в файле lab05-3.asm



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

Рис. 3.15: Запуск программы lab05-3.asm

Аналогично скопировал программу lab05-2.asm и изменил код, но теперь использовал подпрограммы из файла in_out.asm.

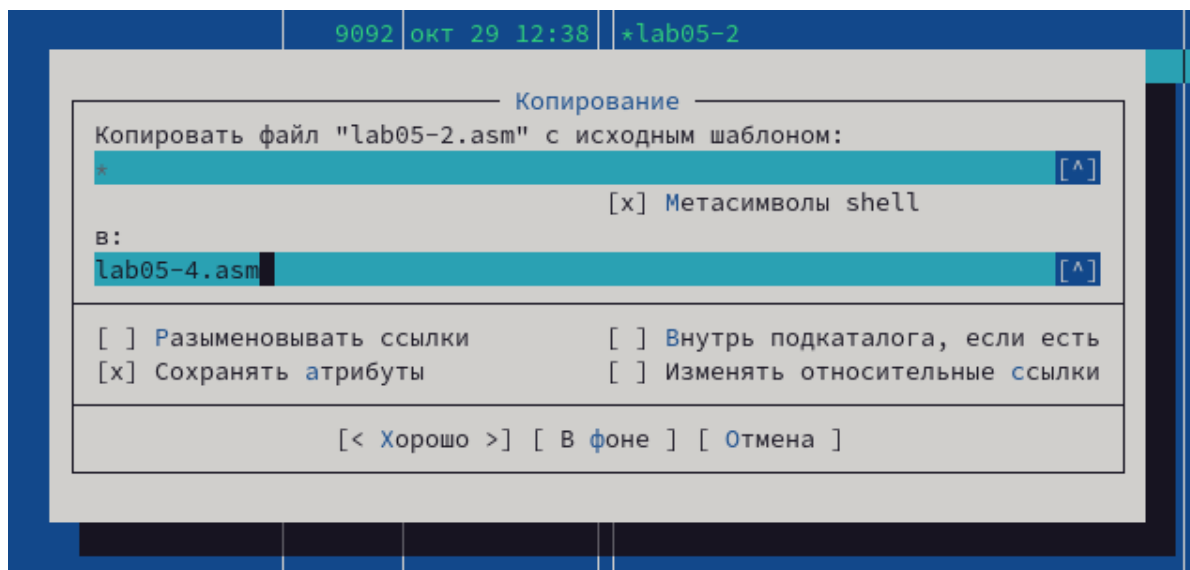
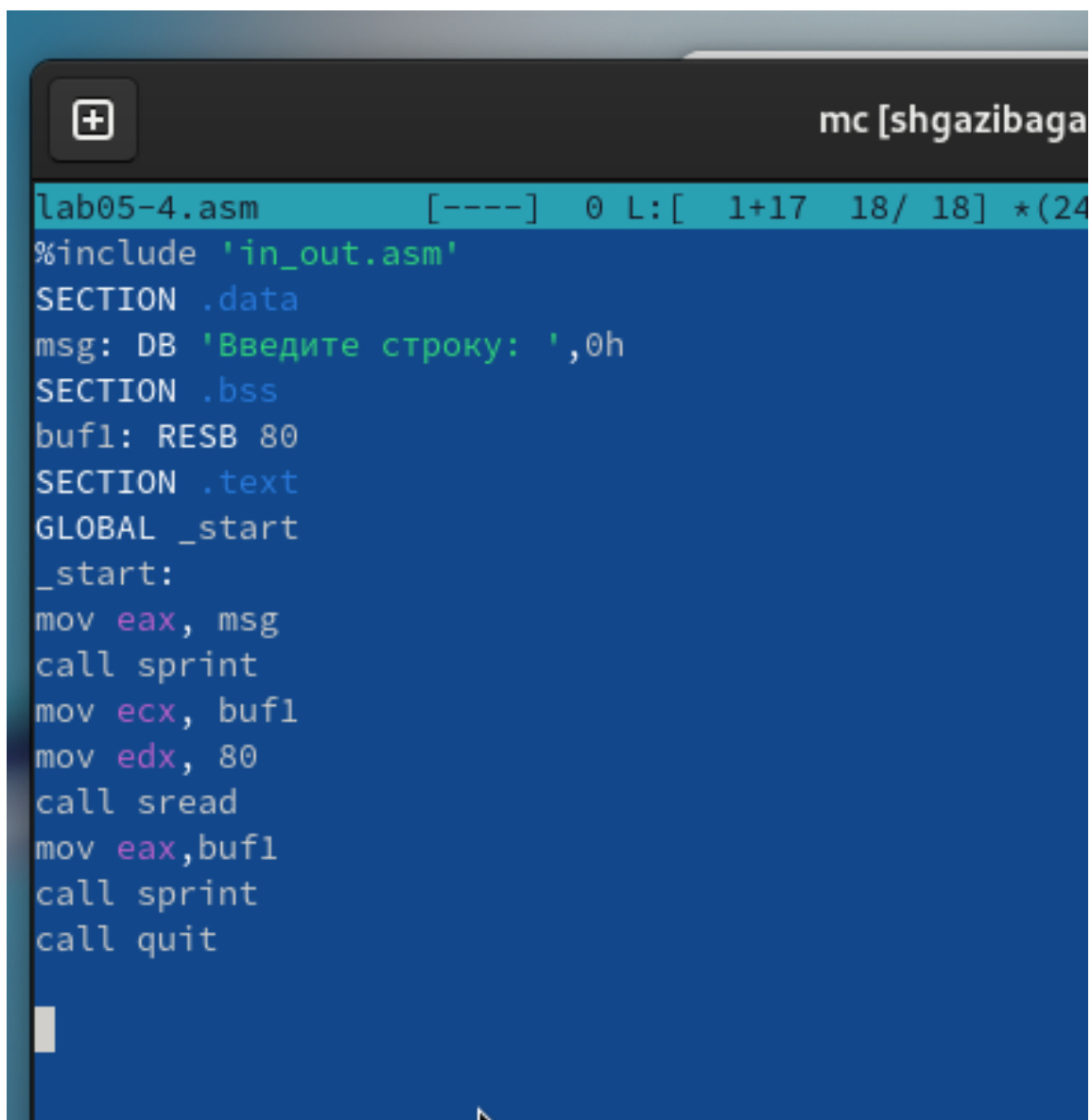
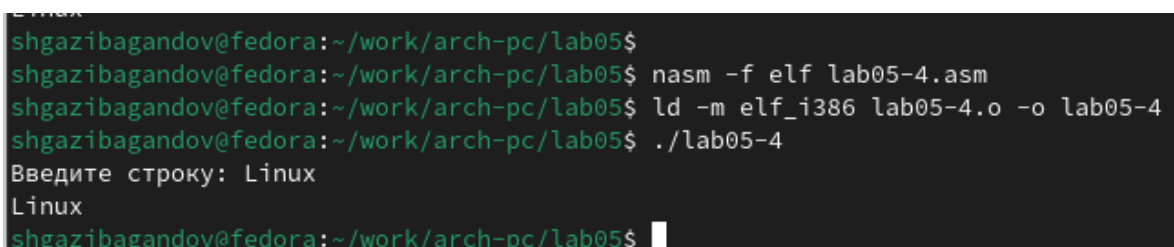


Рис. 3.16: Копирование файла lab05-2.asm



```
lab05-4.asm [----] 0 L: [ 1+17 18/ 18] *(24
#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. 3.17: Программа в файле lab05-4.asm



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

Рис. 3.18: Запуск программы lab05-4.asm

4 Выводы

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