

Отчёт по лабораторной работе 5

Архитектура компьютера

Сувд Адисурэн

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	11
2.3	Задание для самостоятельной работы	15
3	Выводы	19

Список иллюстраций

2.1	Запуск Midnight Commander	6
2.2	Создание каталога	7
2.3	Создание файла lab05-1.asm	8
2.4	Программа в файле lab05-1.asm	9
2.5	Просмотр файла lab05-1.asm	10
2.6	Запуск программы lab05-1.asm	11
2.7	Копирование файла in_out.asm	12
2.8	Копирование файла lab05-1.asm	12
2.9	Программа в файле lab05-2.asm	13
2.10	Запуск программы lab05-2.asm	13
2.11	Программа в файле lab05-2.asm	14
2.12	Запуск программы lab05-2.asm	14
2.13	Копирование файла lab05-1.asm	15
2.14	Программа в файле lab05-3.asm	16
2.15	Запуск программы lab05-3.asm	16
2.16	Копирование файла lab05-2.asm	17
2.17	Программа в файле lab05-4.asm	18
2.18	Запуск программы lab05-4.asm	18

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 открыла Midnight Commander и с помощью клавиш со стрелками и Enter перешла в каталог ~/work/arch-рс. Затем нажала F7 и создала каталог lab05.

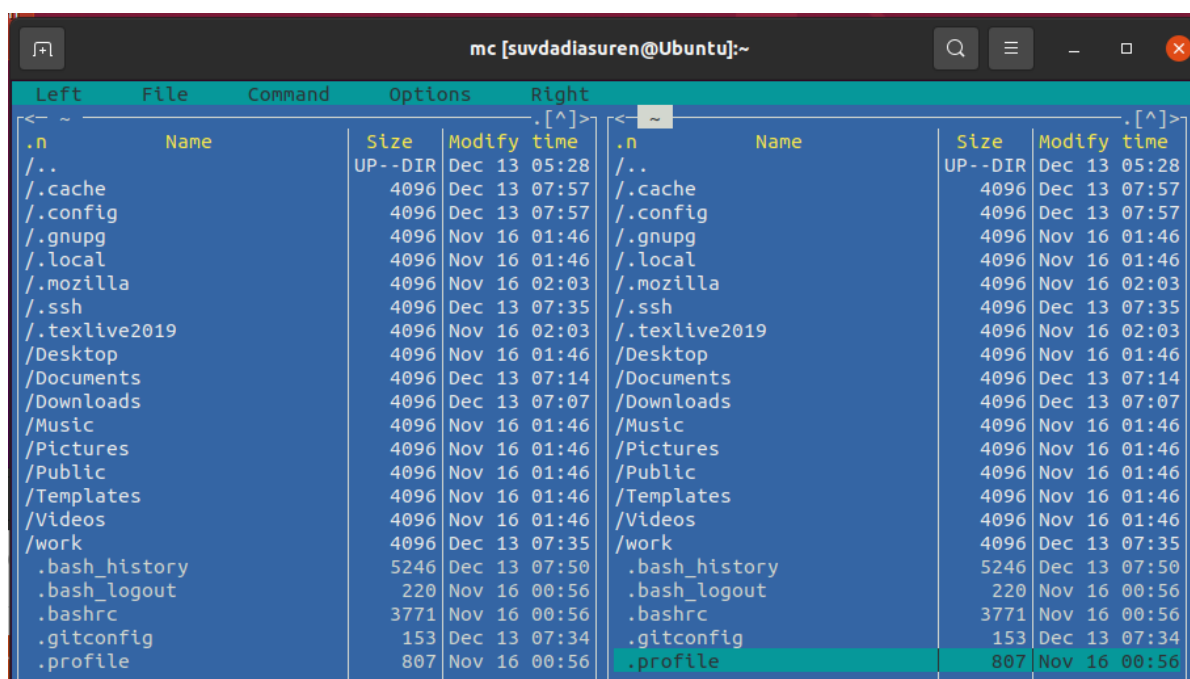


Рис. 2.1: Запуск Midnight Commander

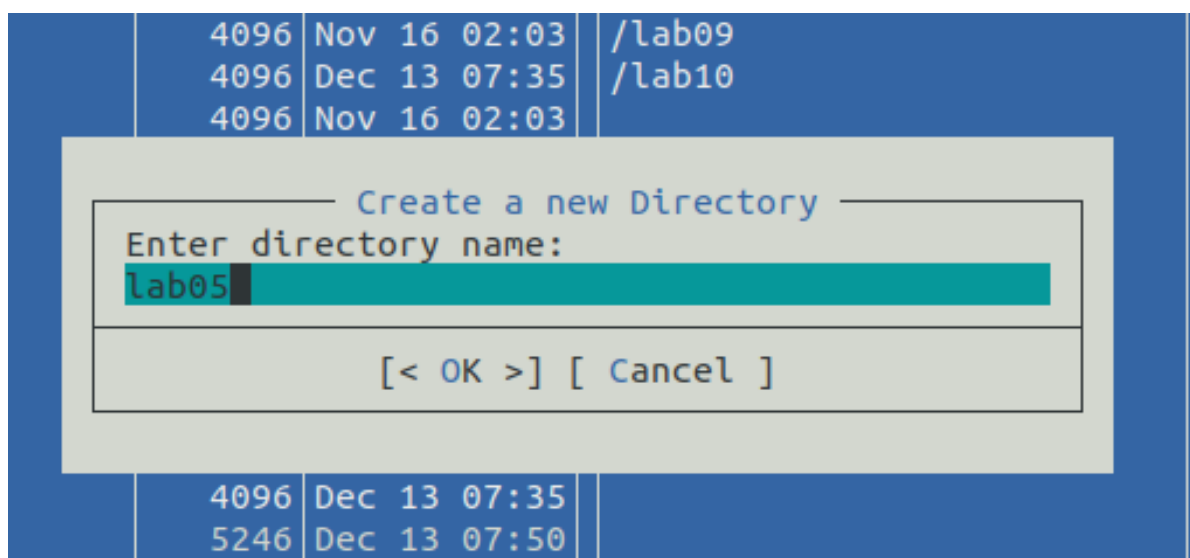


Рис. 2.2: Создание каталога

При помощи команды `touch` я создала файл `lab05-1.asm`.

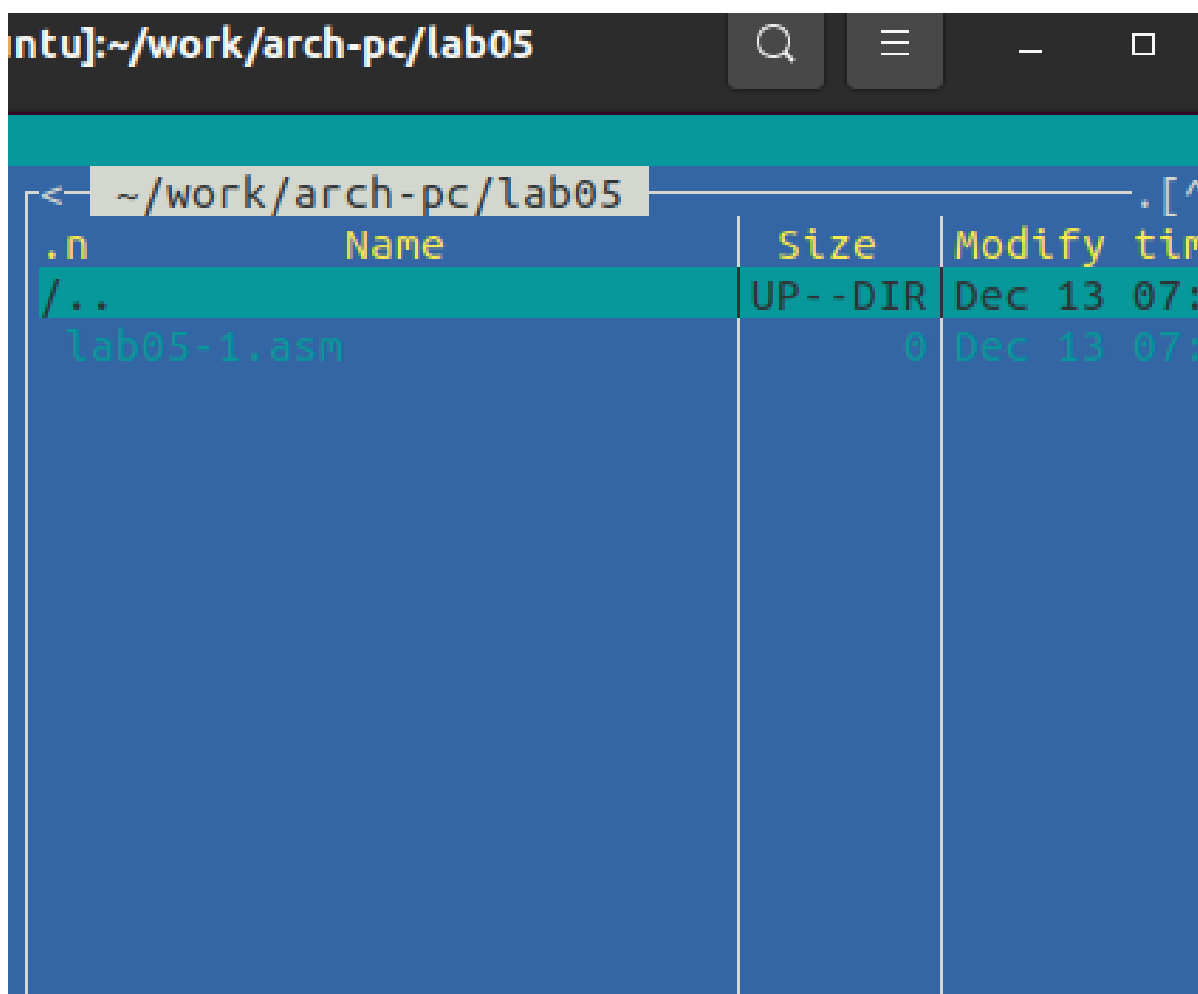
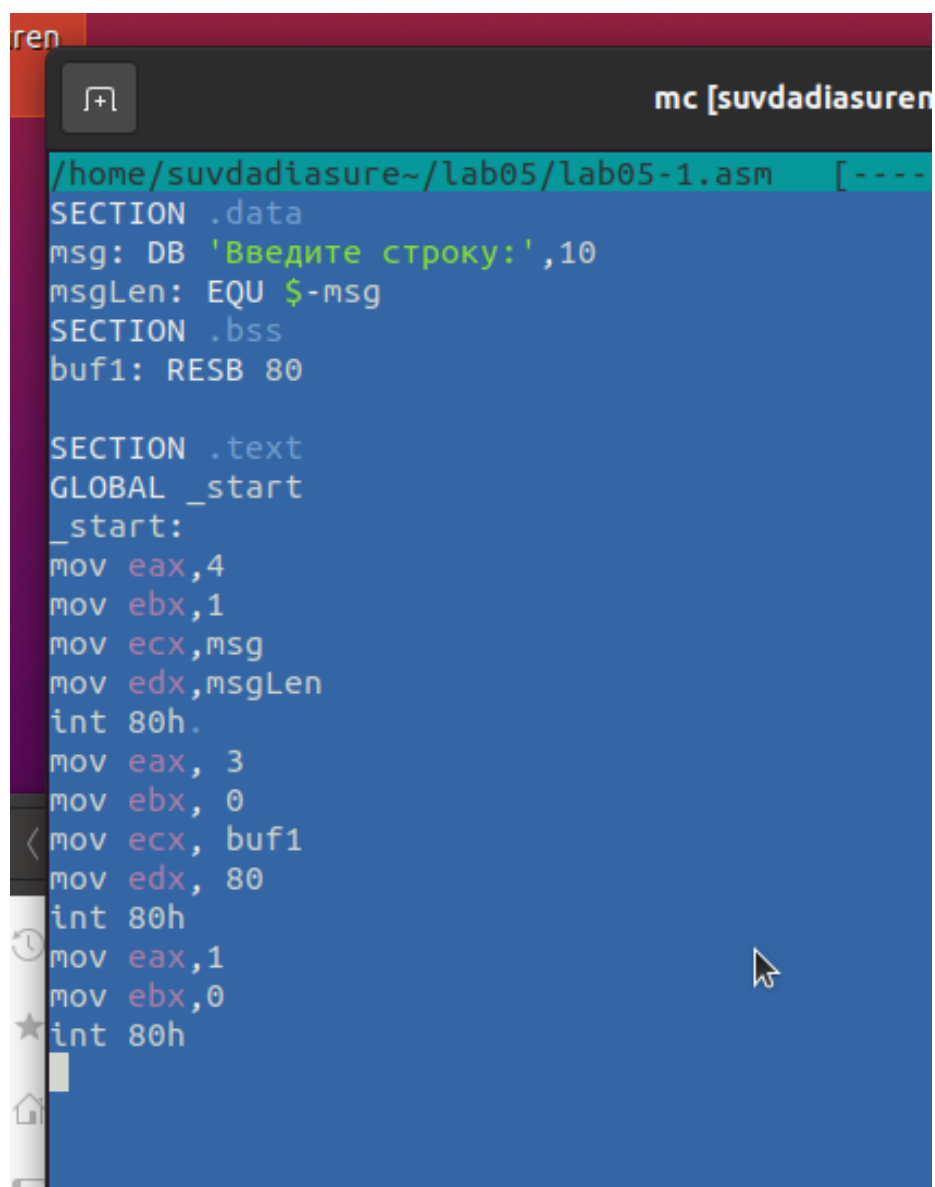


Рис. 2.3: Создание файла lab05-1.asm

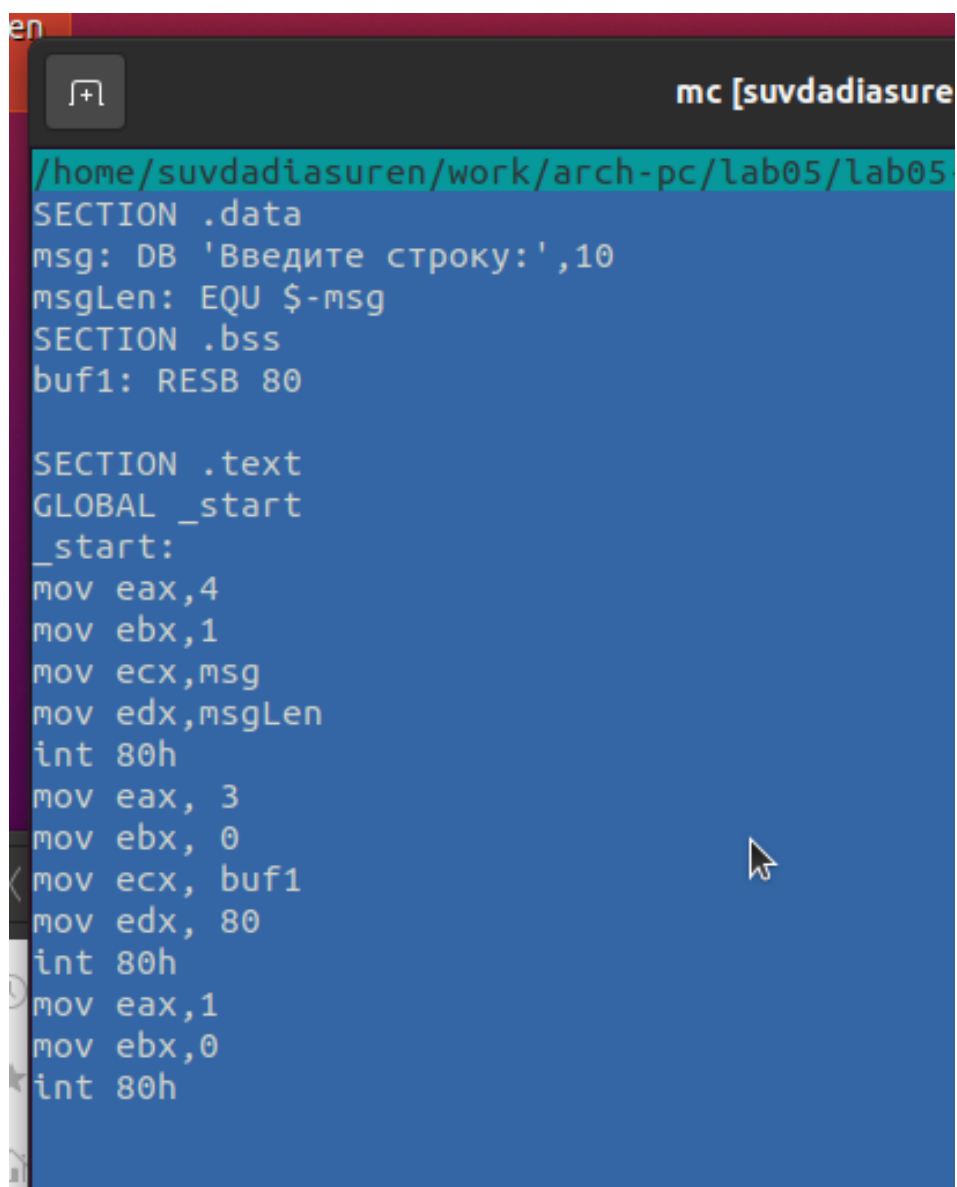
Открыла файл на редактирование, нажав F4, выбрала редактор mceditor и написала код программы из задания.



```
ren  
mc [suvdadiasuren  
/home/suvdadiasure~/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. 2.4: Программа в файле lab05-1.asm

Открыла файл для просмотра, нажав F3, и убедилась, что он содержит написанный код.

A screenshot of a text editor window titled 'mc [suvdadiasuren]'. The editor displays assembly code for a file named 'lab05-1.asm'. The code is as follows:

```
/home/suvdadiasuren/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 файла lab05-1.asm

Я транслировала файл программы в объектный файл, выполнила компоновку объектного файла и получила исполняемый файл программы, проверив ее работу.

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

Рис. 2.6: Запуск программы lab05-1.asm

2.2 Подключение внешнего файла in_out.asm

Для упрощения написания программ часто встречающиеся одинаковые участки кода, такие как вывод строки на экран или выход из программы, можно оформить в виде подпрограмм и сохранить в отдельные файлы. 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, <message>`);
- `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 будет записано (`mov eax, <buffer>`), в регистр `ebx` – длину вводимой строки (`mov ebx, <N>`);
- `iprint` – вывод на экран чисел в формате ASCII. Перед вызовом `iprint` в регистр `eax` необходимо записать выводимое число (`mov eax, <int>`);
- `iprintLF` – работает аналогично `iprint`, но при выводе на экран после числа добавляет 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, <int>`);
- `quit` – завершение программы.

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

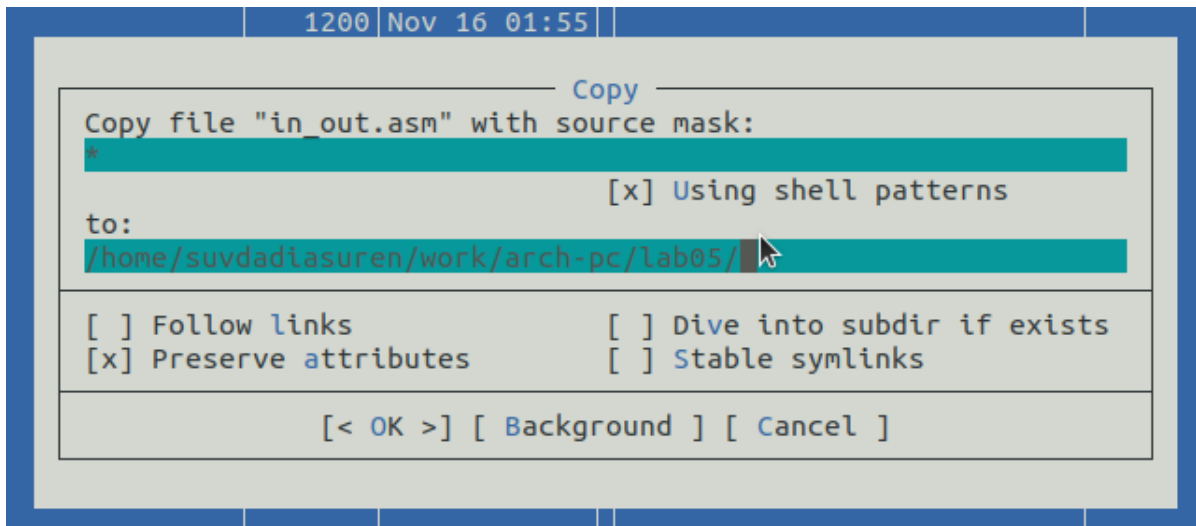


Рис. 2.7: Копирование файла `in_out.asm`

Я скопировала `lab05-1.asm` в `lab05-2.asm`.

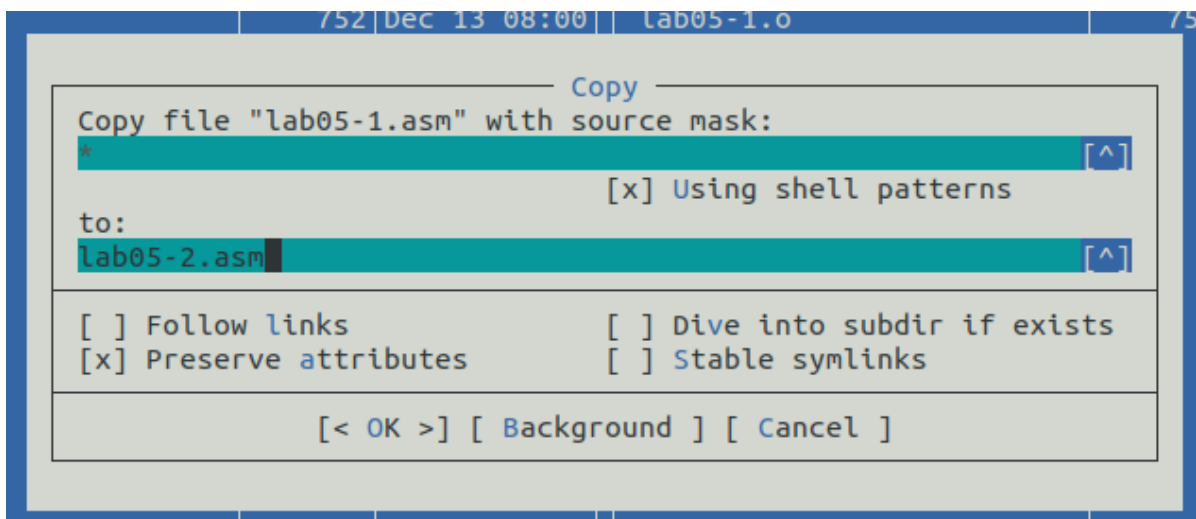
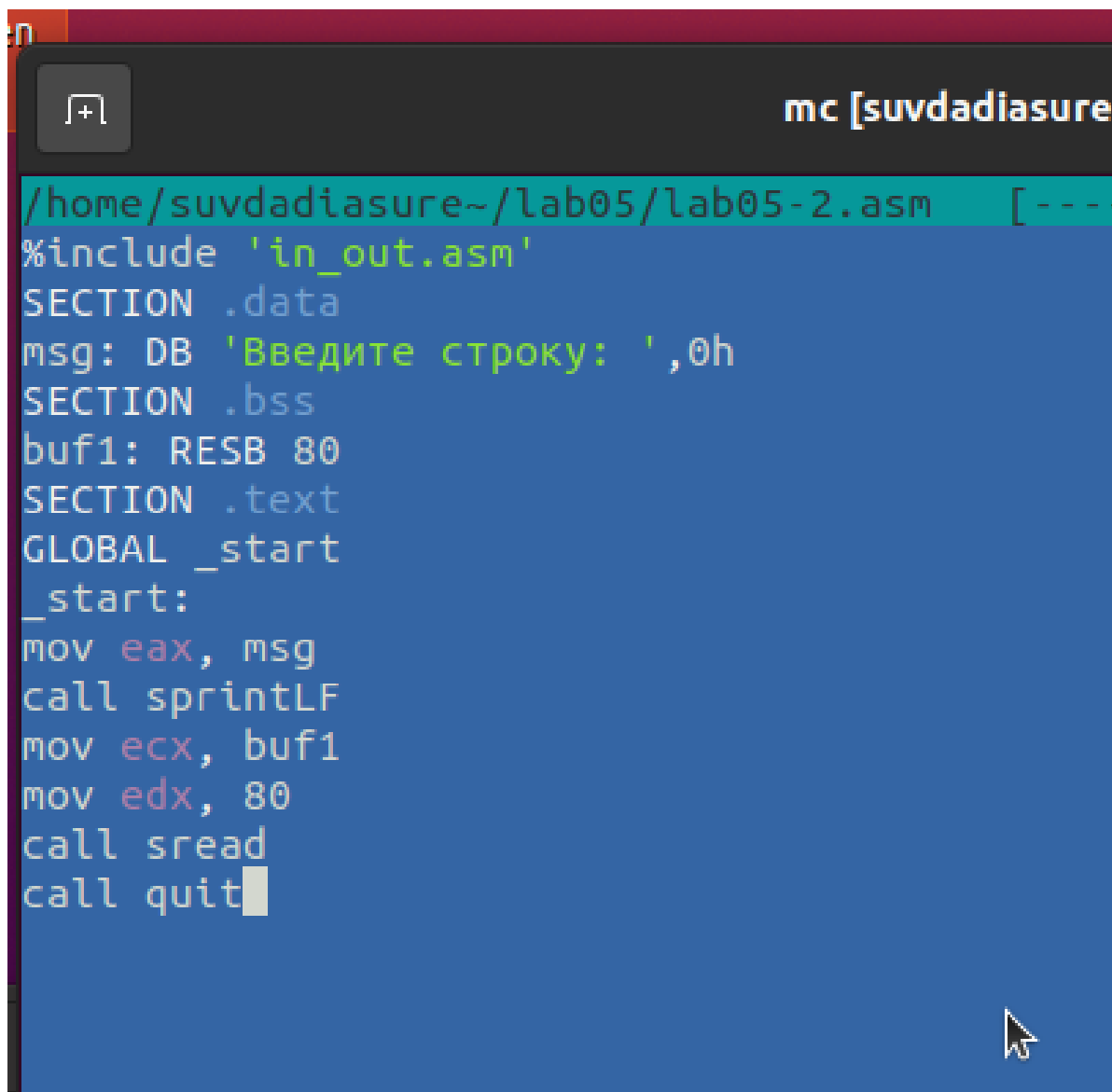


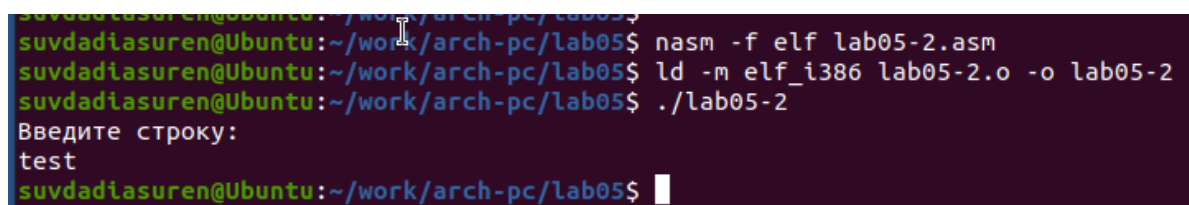
Рис. 2.8: Копирование файла `lab05-1.asm`

Написала код программы lab05-2.asm, используя подпрограммы из внешнего файла in_out.asm. Скомпилировала программу и проверила запуск.

A screenshot of a text editor window titled 'mc [suvdadiasuren@ubuntu:~/work/arch-pc/lab05\$]'. The editor displays the assembly code for 'lab05-2.asm'. The code includes a directive to include 'in_out.asm', defines a data section with a message string, a bss section with a buffer, and a text section with the main logic. The logic involves printing the message and reading user input into the buffer.

```
/home/suvdadiasuren~/lab05/lab05-2.asm [ ---
%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
```

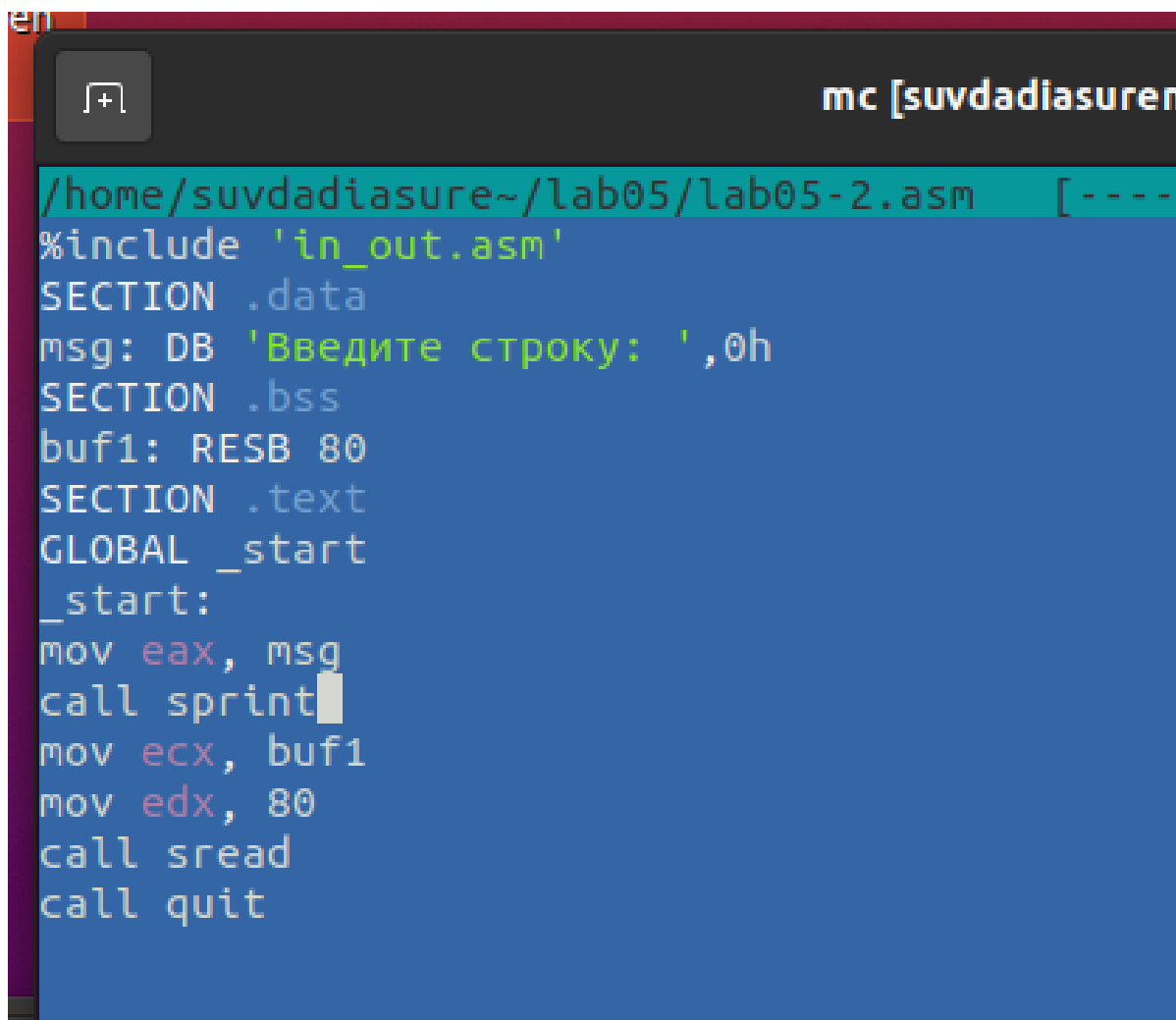
Рис. 2.9: Программа в файле lab05-2.asm

A screenshot of a terminal window showing the compilation and execution of the assembly program. The user runs 'nasm' to compile the assembly file into an object file, then 'ld' to link it into an executable, and finally runs the executable. The program prompts for input, and the user enters 'test'.

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

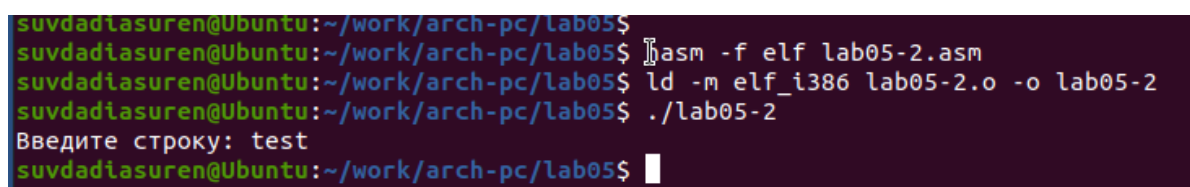
Рис. 2.10: Запуск программы lab05-2.asm

В файле lab05-2.asm я заменила подпрограмму `sprintLF` на `sprint`. Заново собрала исполняемый файл. Теперь после вывода строки она не завершается символом перехода на новую строку.



```
mc [suvdadiasuren
/home/suvdadiasure~/lab05/lab05-2.asm [----
%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: Программа в файле lab05-2.asm



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

Рис. 2.12: Запуск программы lab05-2.asm

2.3 Задание для самостоятельной работы

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

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

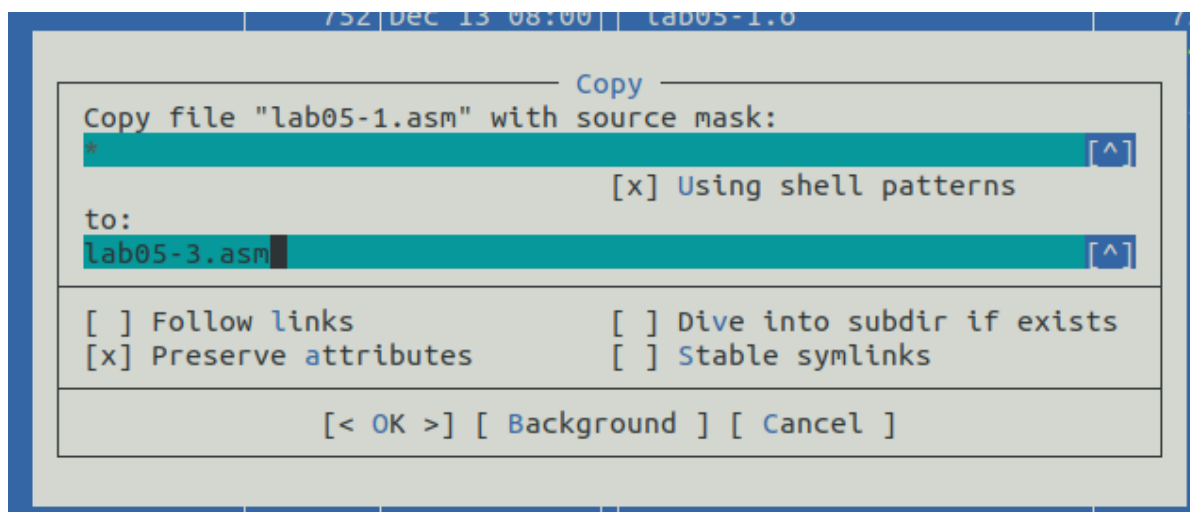
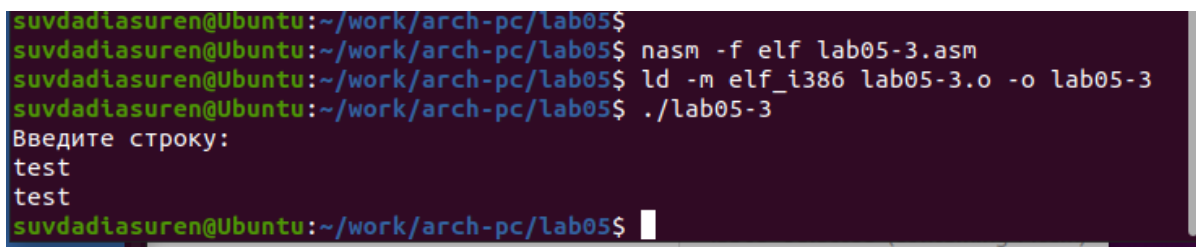


Рис. 2.13: Копирование файла lab05-1.asm



```
mc [suvdadiasuren@  
/home/suvdadiasure~/lab05/lab05-3.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,4  
mov ebx,1  
mov ecx,buf1  
mov edx,80  
int 80h  
mov eax,1  
mov ebx,0  
int 80h
```

Рис. 2.14: Программа в файле lab05-3.asm



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

Рис. 2.15: Запуск программы lab05-3.asm

Аналогично я скопировала программу lab05-2.asm и изменила код, но теперь использовала подпрограммы из файла in_out.asm.

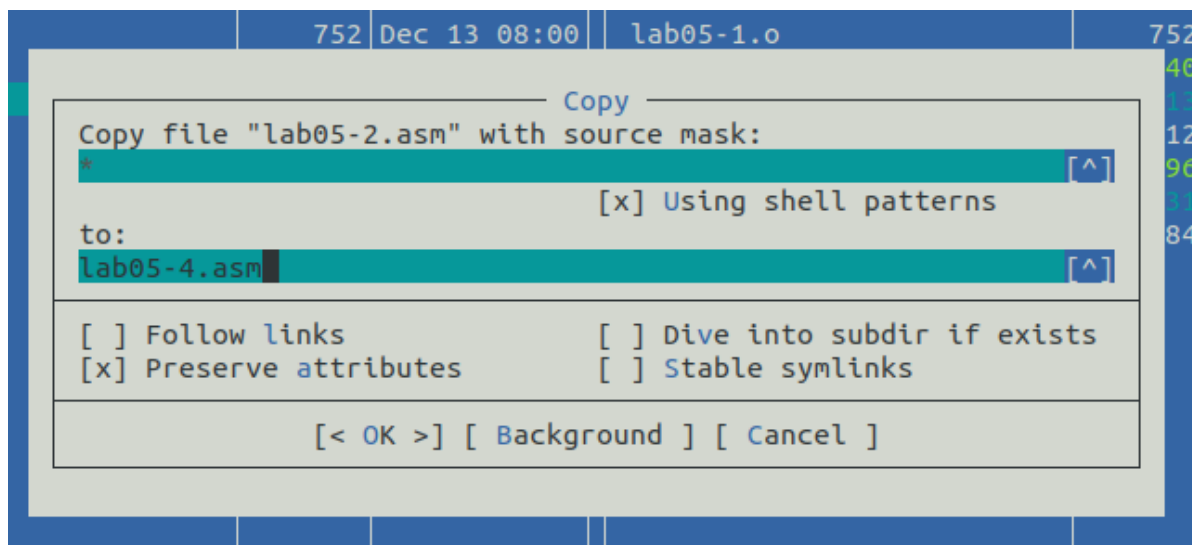
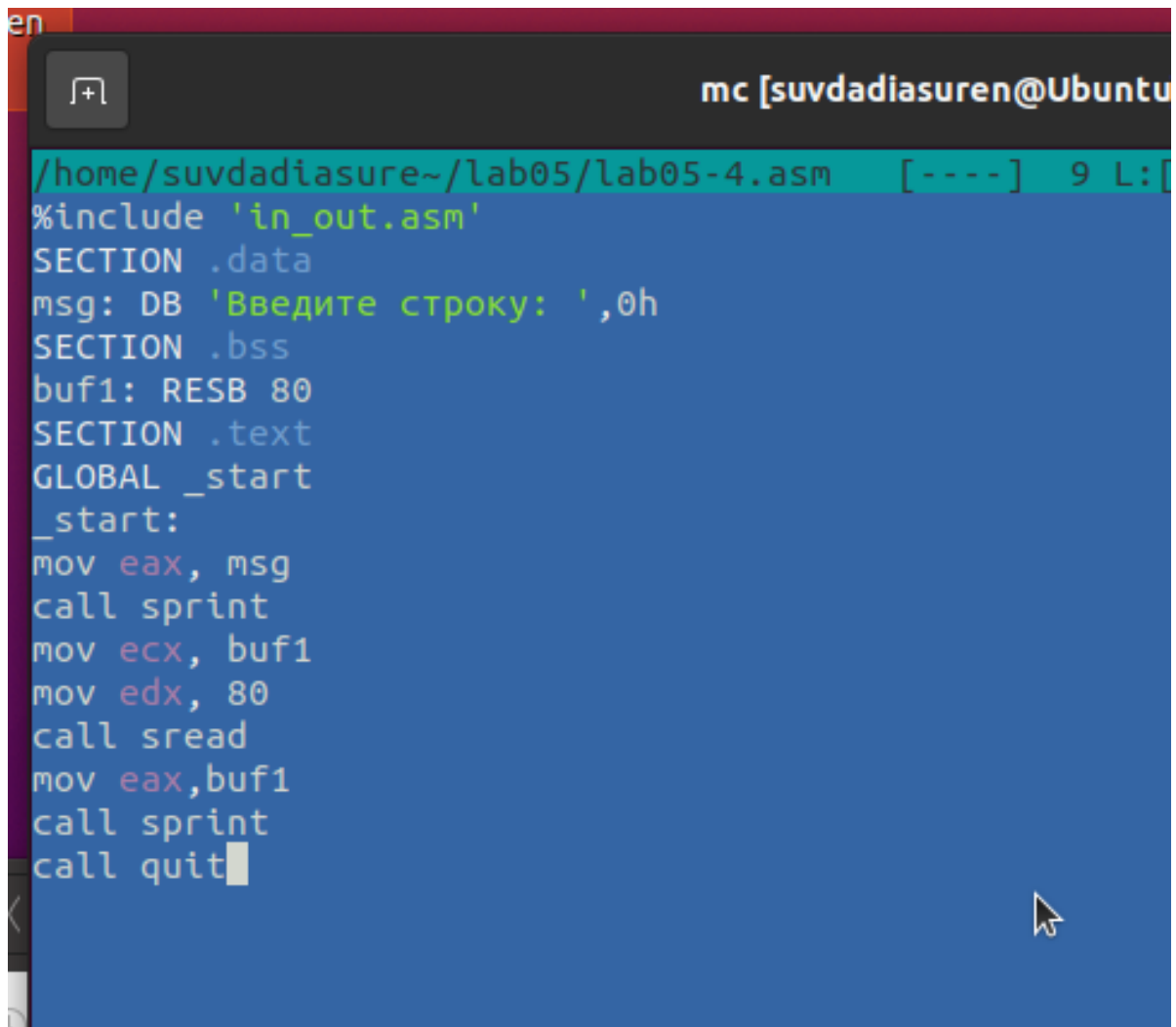
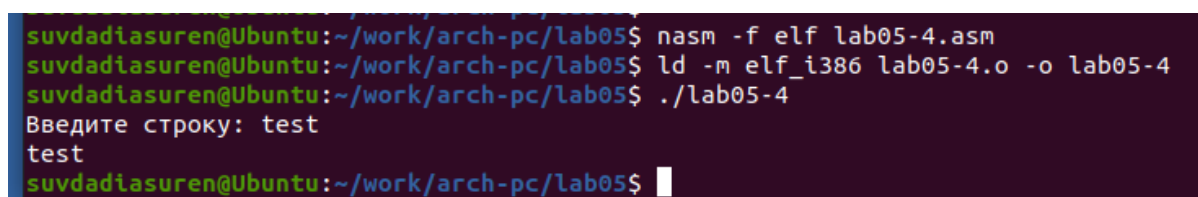


Рис. 2.16: Копирование файла lab05-2.asm



```
mc [suvdadiasuren@Ubuntu
/home/suvdadiasure~/lab05/lab05-4.asm [----] 9 L:
#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: Программа в файле lab05-4.asm



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