

Отчёт по лабораторной работе 5

Архитектура компьютеров

Когенгар Ришард

Содержание

1	Цель работы	5
2	Выполнение лабораторной работы	6
3	Выводы	15

Список иллюстраций

2.1	Создание каталога	6
2.2	Создание файла lab05-1.asm	7
2.3	Программа в файле lab05-1.asm	8
2.4	Просмотр файла lab05-1.asm	9
2.5	Запуск программы lab05-1.asm	9
2.6	Копирование файла	10
2.7	Программа в файле lab05-2.asm	11
2.8	Запуск программы lab05-2.asm	11
2.9	Программа в файле lab05-2.asm	12
2.10	Запуск программы lab05-2.asm	12
2.11	Программа в файле lab05-3.asm	13
2.12	Запуск программы lab05-3.asm	13
2.13	Программа в файле lab05-4.asm	14
2.14	Запуск программы lab05-4.asm	14

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

1. Открыл Midnight Commander
2. Перешел в каталог ~/work/arch-pc
3. Создал каталог lab05

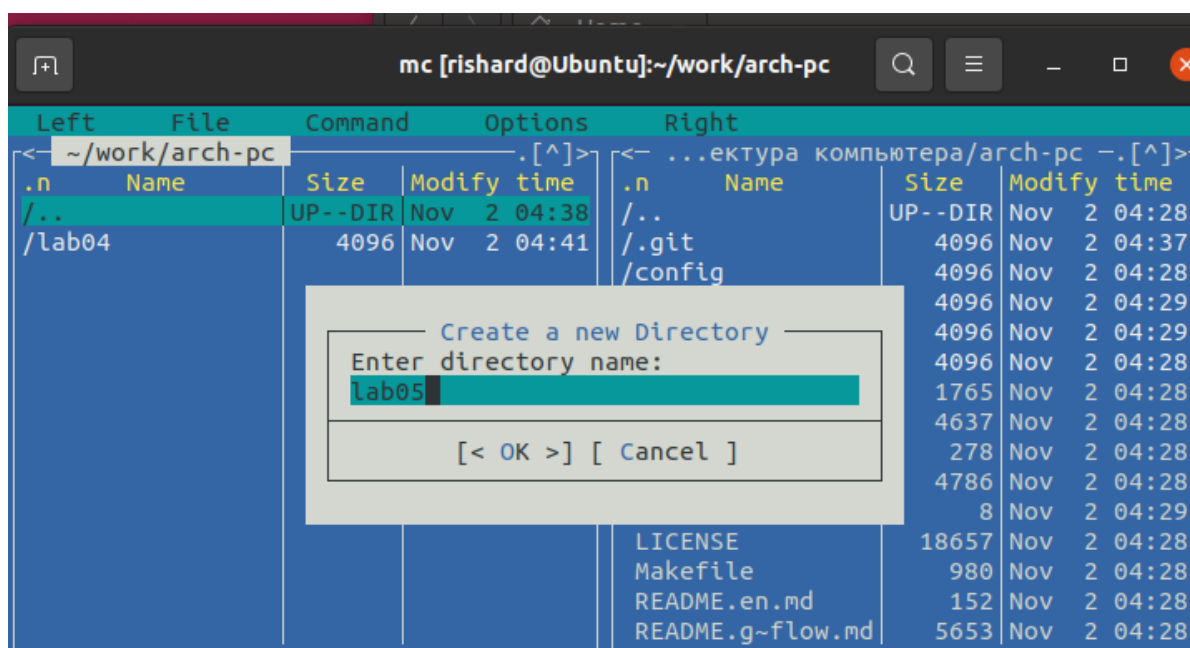


Рис. 2.1: Создание каталога

4. Создал файл lab05-1.asm

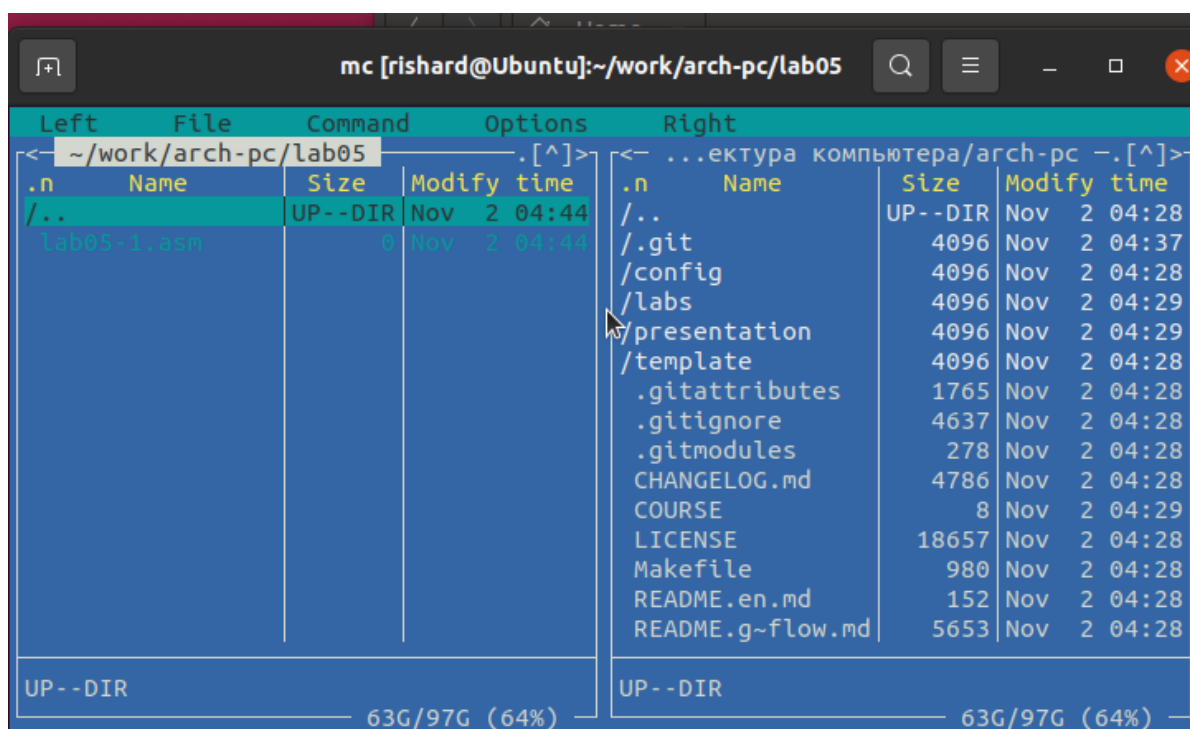
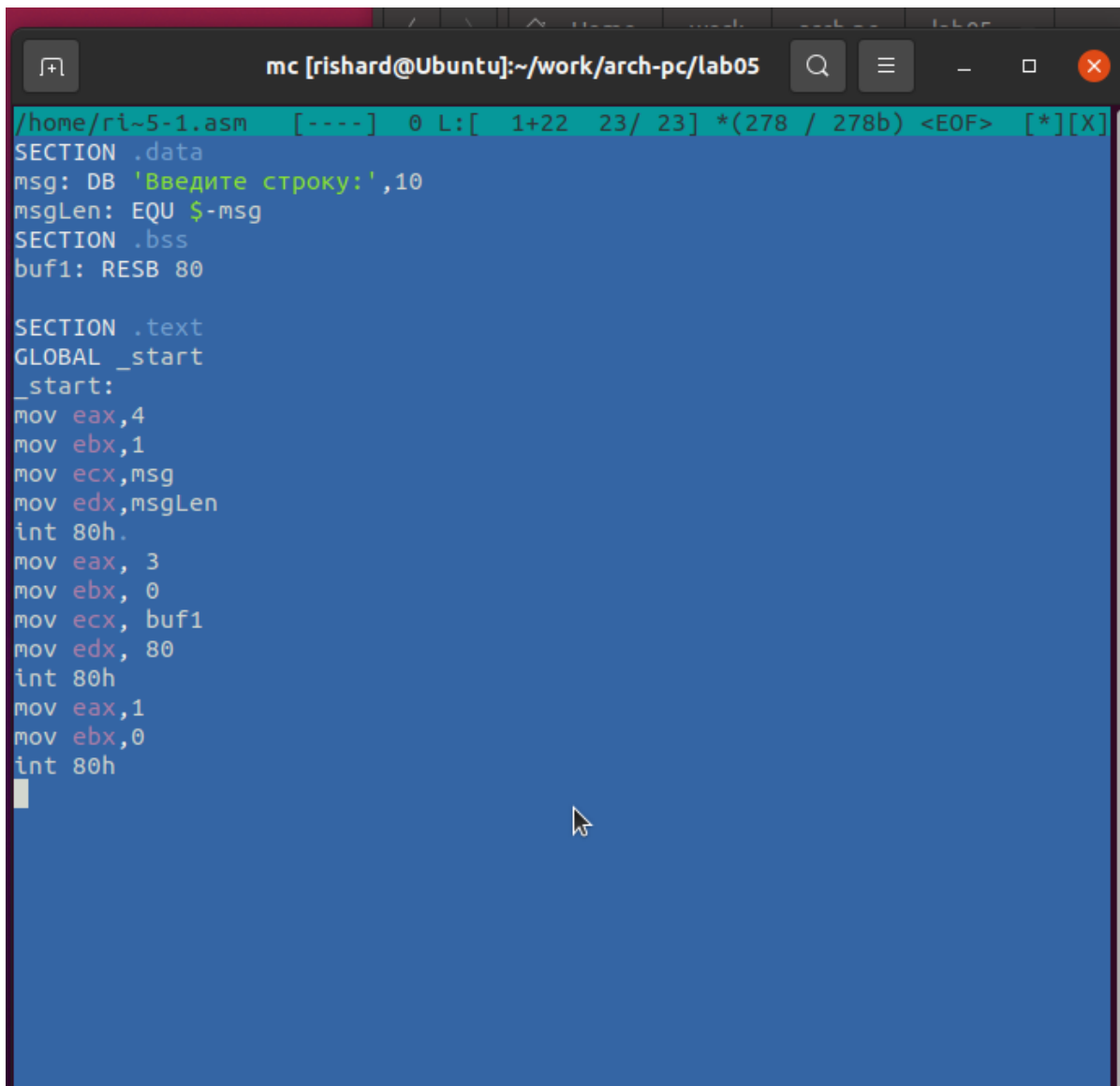


Рис. 2.2: Создание файла lab05-1.asm

5. Открыл файл на редактирование

6. Написал код

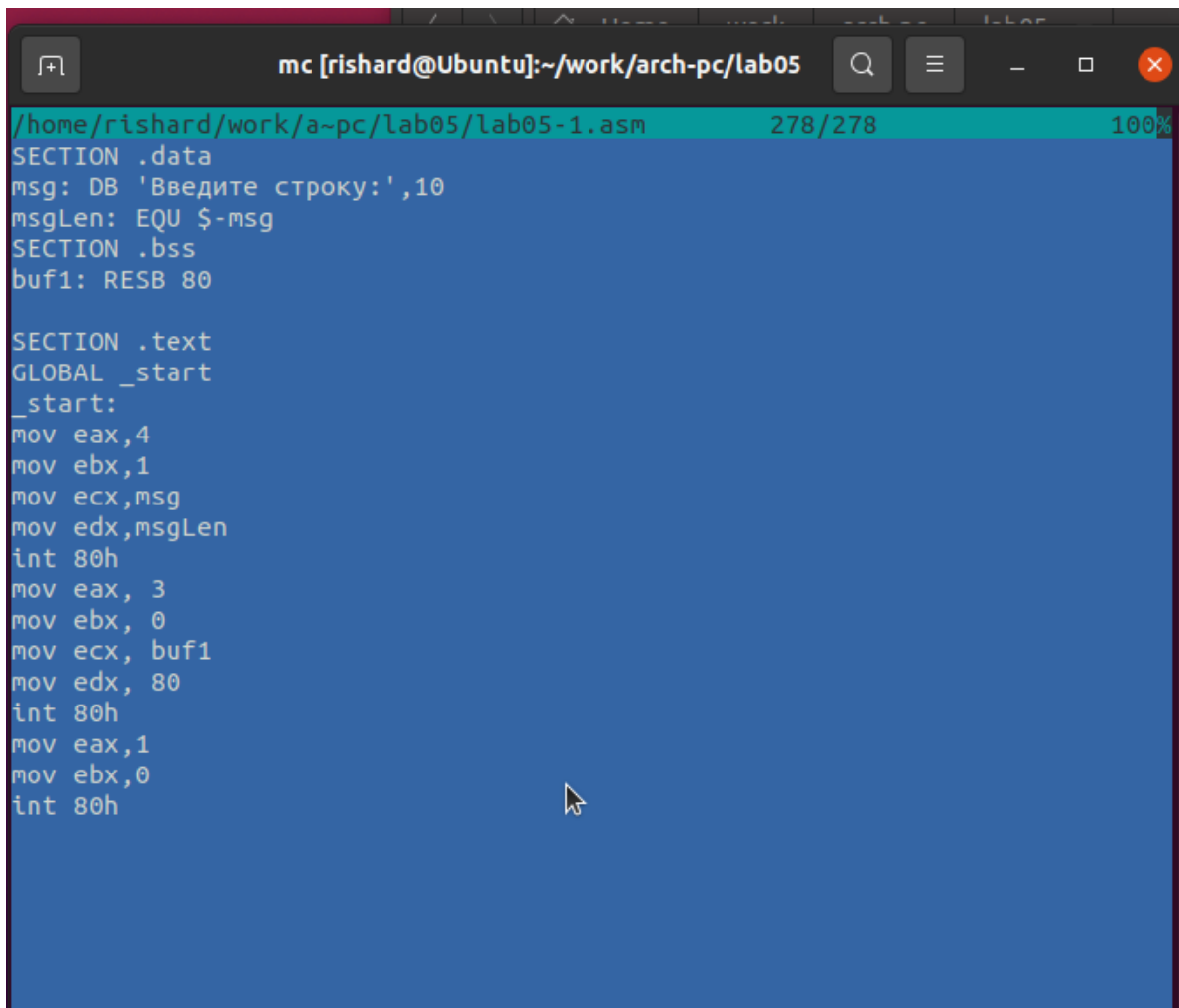
A screenshot of a code editor window titled 'mc [rishard@Ubuntu]:~/work/arch-pc/lab05'. The editor displays assembly code for a file named '/home/rishard-5-1.asm'. The code is organized into sections: .data, .bss, and .text. The .data section contains a message 'Введите строку:' and its length. The .bss section reserves space for a buffer. The .text section contains the main logic, starting with a global _start label. The code uses MOV instructions to set up registers (eax, ebx, ecx, edx) and INT 80h calls to interact with the operating system. The code is as follows:

```
/home/rishard-5-1.asm  [----]  0 L:[ 1+22 23/ 23] *(278 / 278b) <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. 2.3: Программа в файле lab05-1.asm

7. Открыл файл на просмотр и убедился, что он содержит набранный код.

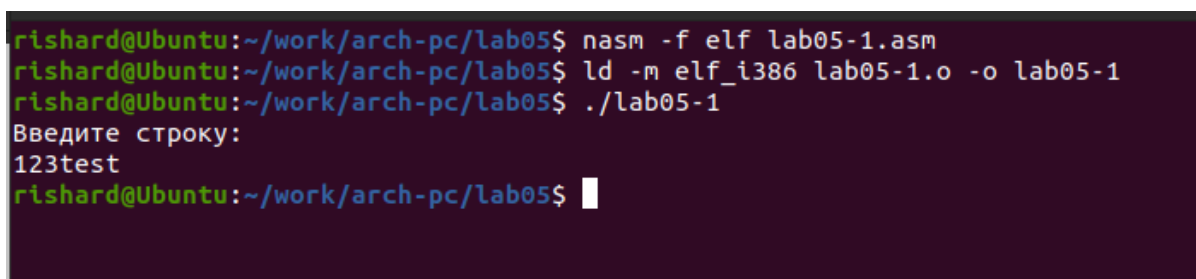


```
mc [rishard@Ubuntu]:~/work/arch-pc/lab05 278/278 100%
/home/rishard/work/a~pc/lab05/lab05-1.asm
SECTION .data
msg: DB 'Введите строку:',10
msgLen: EQU $-msg
SECTION .bss
buf1: RESB 80

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. 2.4: Просмотр файла lab05-1.asm

8. Получил исполняемый файл программы и проверил ее работу.



```
rishard@Ubuntu:~/work/arch-pc/lab05$ nasm -f elf lab05-1.asm
rishard@Ubuntu:~/work/arch-pc/lab05$ ld -m elf_i386 lab05-1.o -o lab05-1
rishard@Ubuntu:~/work/arch-pc/lab05$ ./lab05-1
Введите строку:
123test
rishard@Ubuntu:~/work/arch-pc/lab05$
```

Рис. 2.5: Запуск программы lab05-1.asm

9. Скачал файл in_out.asm.

10. Добавил файл in_out.asm в рабочий каталог.

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

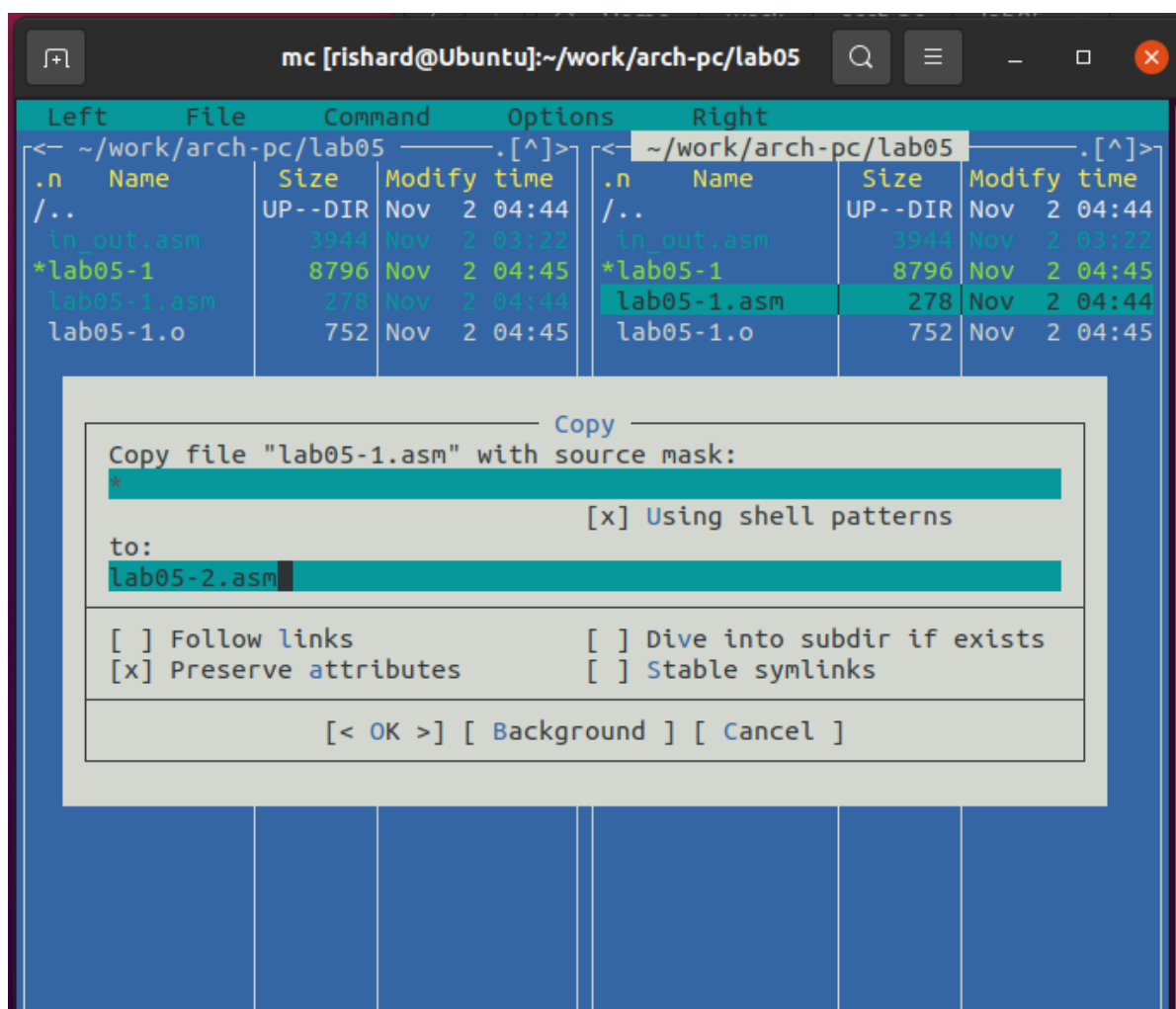
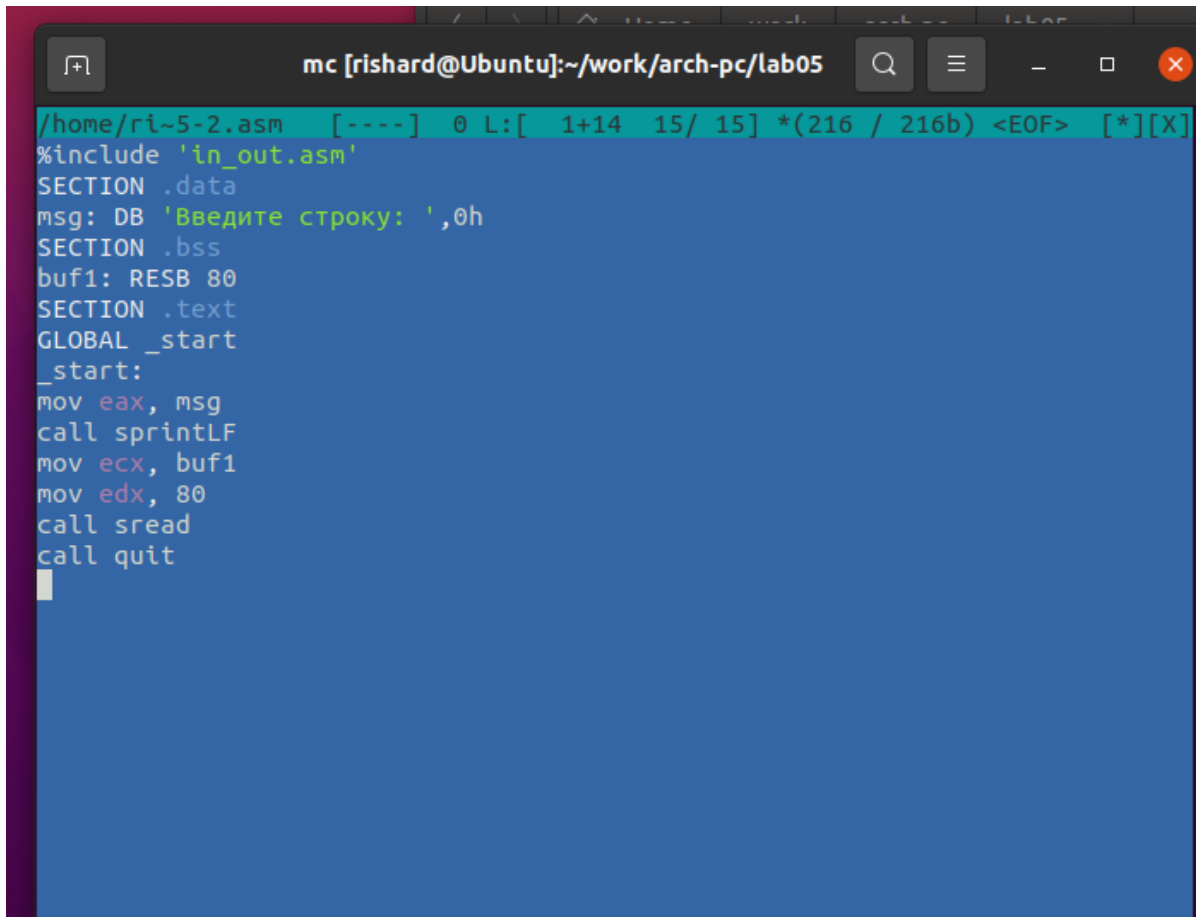


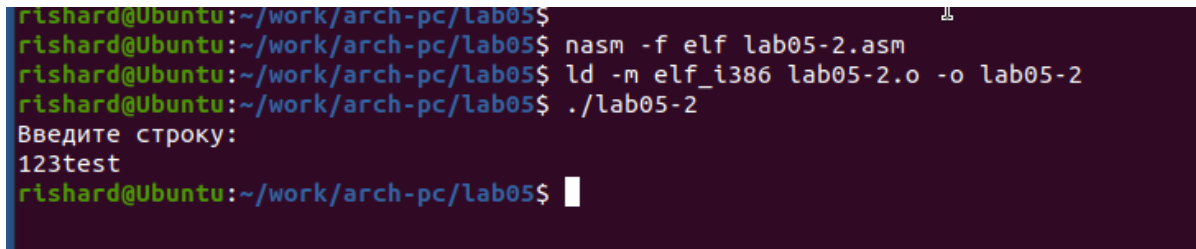
Рис. 2.6: Копирование файла

12. Написал код программы lab05-2.asm. Скомпилировал программу и проверили запуск.



```
mc [rishard@Ubuntu]:~/work/arch-pc/lab05
/home/ri~5-2.asm [----] 0 L:[ 1+14 15/ 15] *(216 / 216b) <EOF> [*][X]
#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
```

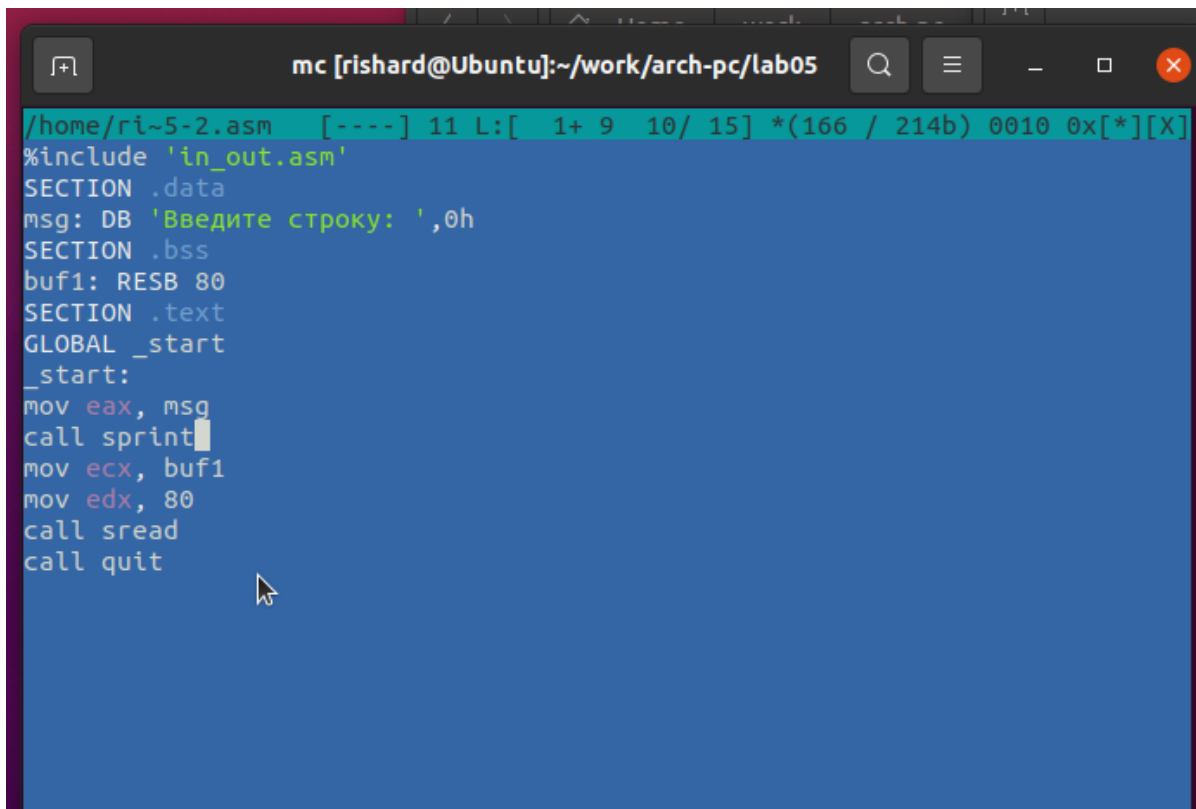
Рис. 2.7: Программа в файле lab05-2.asm



```
rishard@Ubuntu:~/work/arch-pc/lab05$
rishard@Ubuntu:~/work/arch-pc/lab05$ nasm -f elf lab05-2.asm
rishard@Ubuntu:~/work/arch-pc/lab05$ ld -m elf_i386 lab05-2.o -o lab05-2
rishard@Ubuntu:~/work/arch-pc/lab05$ ./lab05-2
Введите строку:
123test
rishard@Ubuntu:~/work/arch-pc/lab05$
```

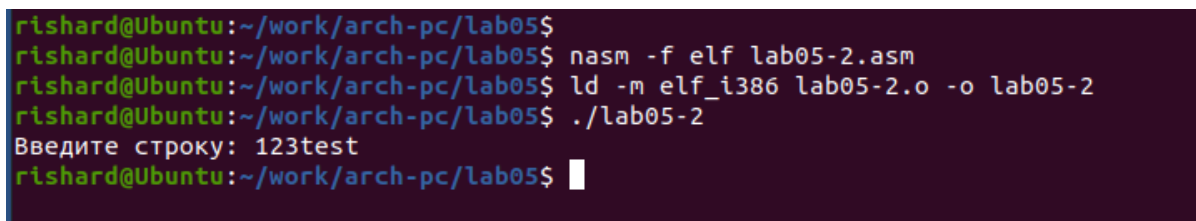
Рис. 2.8: Запуск программы lab05-2.asm

13. 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.



```
mc [rishard@Ubuntu]:~/work/arch-pc/lab05
/home/ri~5-2.asm [---] 11 L:[ 1+ 9 10/ 15] *(166 / 214b) 0010 0x[*][X]
#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
```

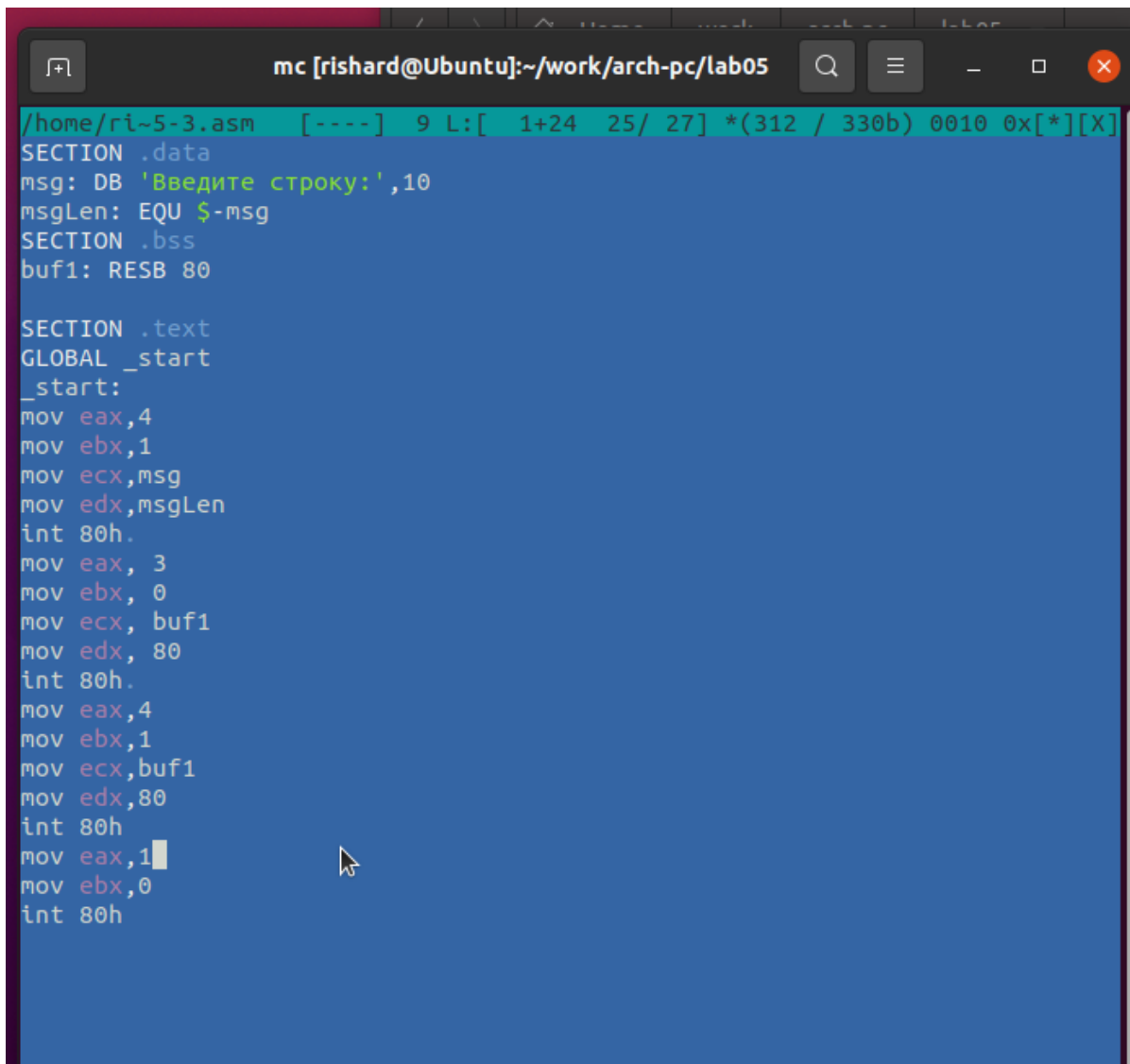
Рис. 2.9: Программа в файле lab05-2.asm



```
rishard@Ubuntu:~/work/arch-pc/lab05$
rishard@Ubuntu:~/work/arch-pc/lab05$ nasm -f elf lab05-2.asm
rishard@Ubuntu:~/work/arch-pc/lab05$ ld -m elf_i386 lab05-2.o -o lab05-2
rishard@Ubuntu:~/work/arch-pc/lab05$ ./lab05-2
Введите строку: 123test
rishard@Ubuntu:~/work/arch-pc/lab05$
```

Рис. 2.10: Запуск программы lab05-2.asm

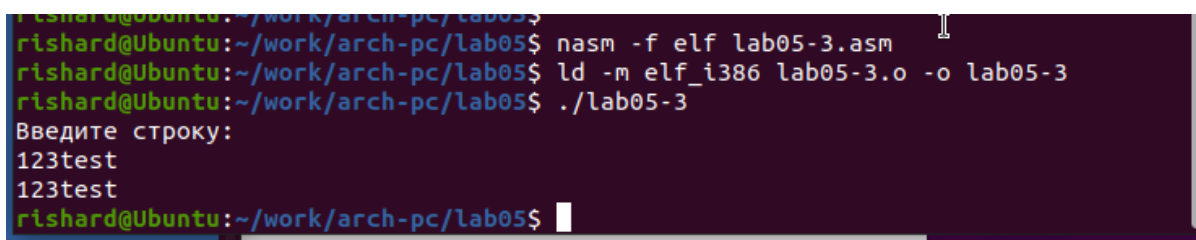
14. Скопировал программу lab05-1.asm и изменил код, чтобы вывести приглашение типа “Введите строку:”, ввести строку с клавиатуры, вывести введённую строку на экран.



```
mc [rishard@Ubuntu]:~/work/arch-pc/lab05
/home/ri~5-3.asm [----] 9 L:[ 1+24 25/ 27] *(312 / 330b) 0010 0x[*][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,80
int 80h
mov eax,1
mov ebx,0
int 80h
```

Рис. 2.11: Программа в файле lab05-3.asm

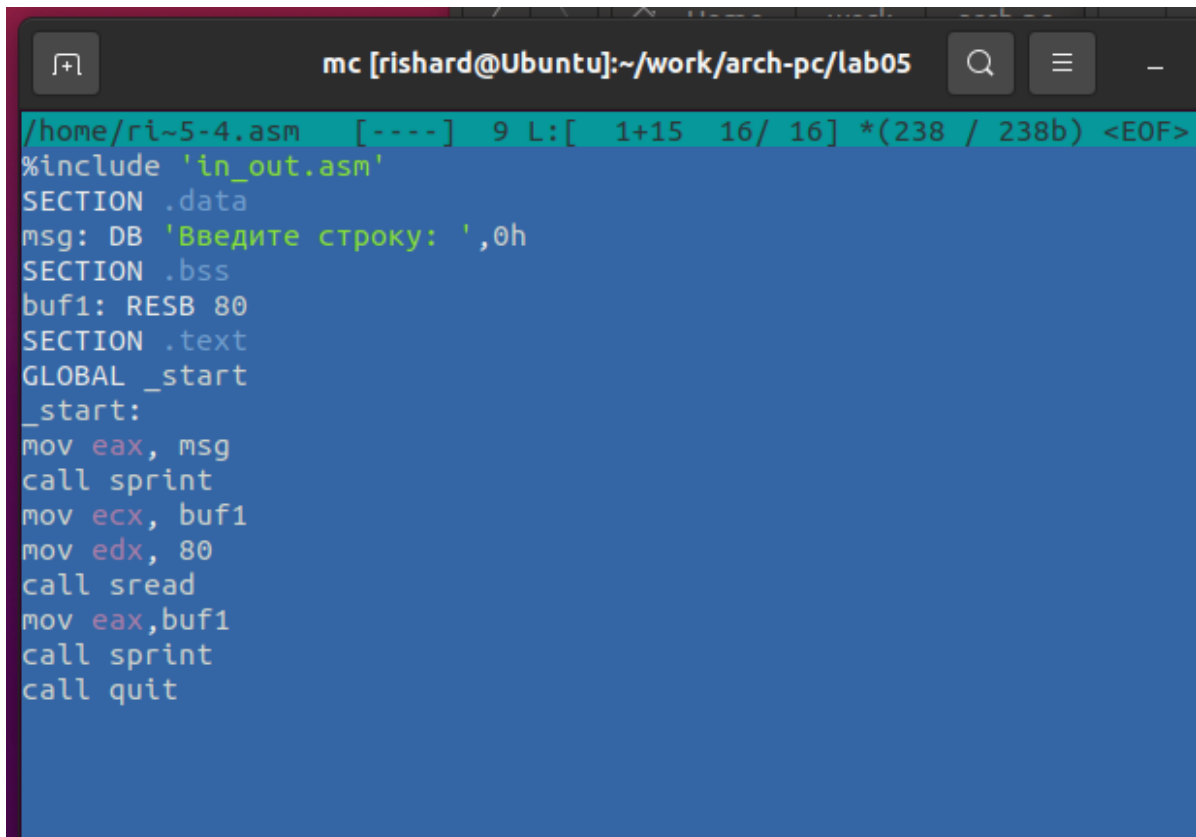


```
rishard@ubuntu:~/work/arch-pc/lab05$
rishard@Ubuntu:~/work/arch-pc/lab05$ nasm -f elf lab05-3.asm
rishard@Ubuntu:~/work/arch-pc/lab05$ ld -m elf_i386 lab05-3.o -o lab05-3
rishard@Ubuntu:~/work/arch-pc/lab05$ ./lab05-3
Введите строку:
123test
123test
rishard@Ubuntu:~/work/arch-pc/lab05$
```

Рис. 2.12: Запуск программы lab05-3.asm

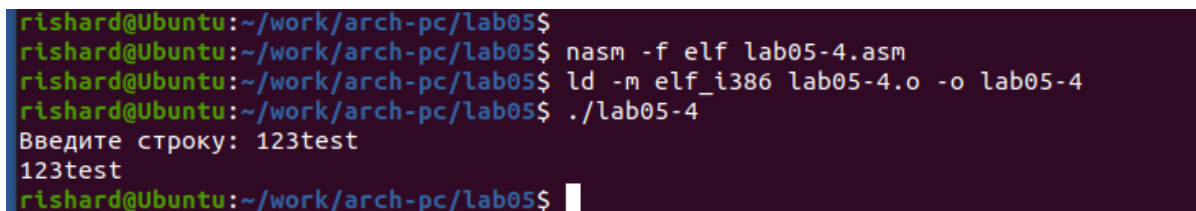
15. Скопировал программу lab05-2.asm и изменил код, чтобы вывести приглашение типа “Введите строку:”, ввести строку с клавиатуры, вывести

введённую строку на экран.



```
mc [rishard@Ubuntu]:~/work/arch-pc/lab05
/home/rishard~5-4.asm [----] 9 L:[ 1+15 16/ 16] *(238 / 238b) <EOF>
#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. 2.13: Программа в файле lab05-4.asm



```
rishard@Ubuntu:~/work/arch-pc/lab05$
rishard@Ubuntu:~/work/arch-pc/lab05$ nasm -f elf lab05-4.asm
rishard@Ubuntu:~/work/arch-pc/lab05$ ld -m elf_i386 lab05-4.o -o lab05-4
rishard@Ubuntu:~/work/arch-pc/lab05$ ./lab05-4
Введите строку: 123test
123test
rishard@Ubuntu:~/work/arch-pc/lab05$
```

Рис. 2.14: Запуск программы lab05-4.asm

Отличие этих двух реализаций в том, что файл `in_out.asm` содержит уже готовые подпрограммы для обеспечения ввода/вывода. Таким образом, нам остается только разместить данные в нужных регистрах и вызвать желаемую подпрограмму с помощью `call`.

3 Выводы

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