

Отчёт по лабораторной работе 5

Архитектура компьютеров

Сухоруков Сергей Андреевич

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	18

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	11
2.8	Копирование файла in_out.asm	11
2.9	Копирование файла lab05-1.asm	12
2.10	Программа lab05-2.asm	13
2.11	Запуск программы lab05-2.asm	13
2.12	Программа в файле lab05-2.asm	14
2.13	Запуск программы lab05-2.asm	14
2.14	Программа lab05-3.asm	15
2.15	Запуск программы lab05-3.asm	16
2.16	Программа lab05-4.asm	17
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-рс. Затем нажимаю F7, чтобы создать новый каталог lab05 (рис. 2.2).

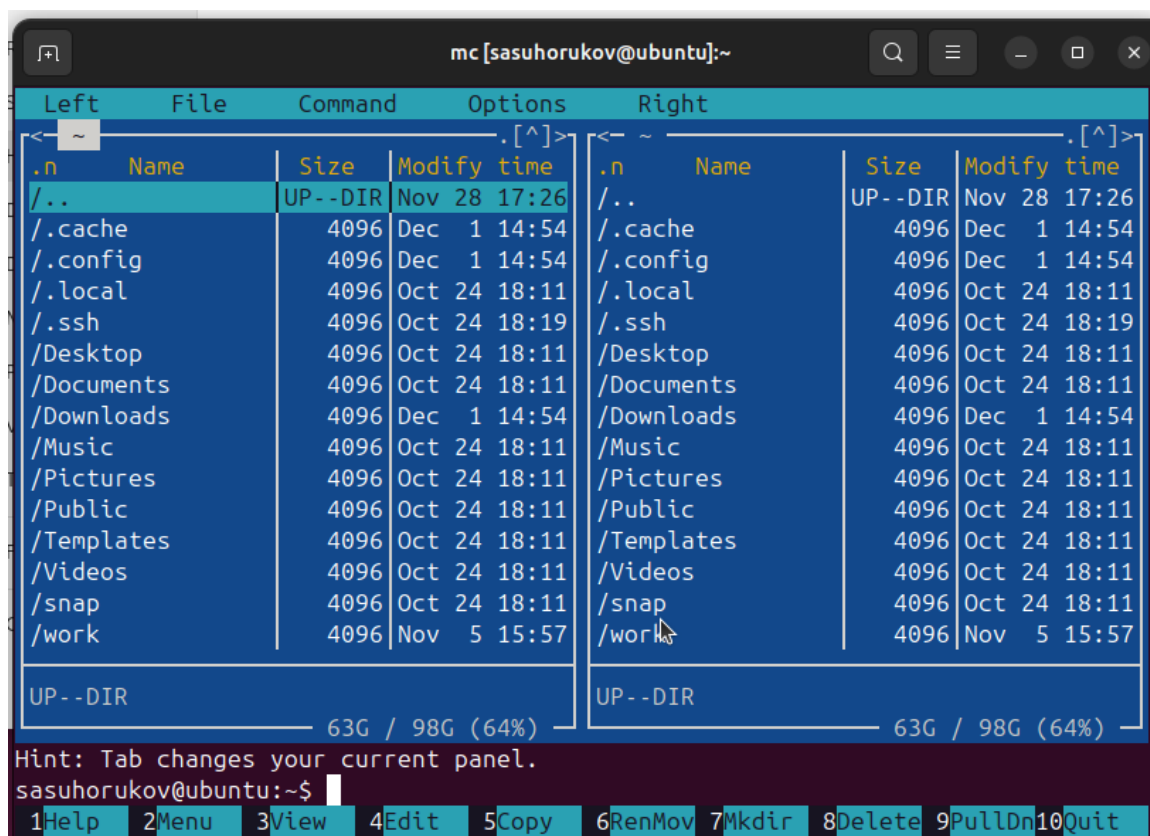


Рисунок 2.1: Запуск Midnight Commander

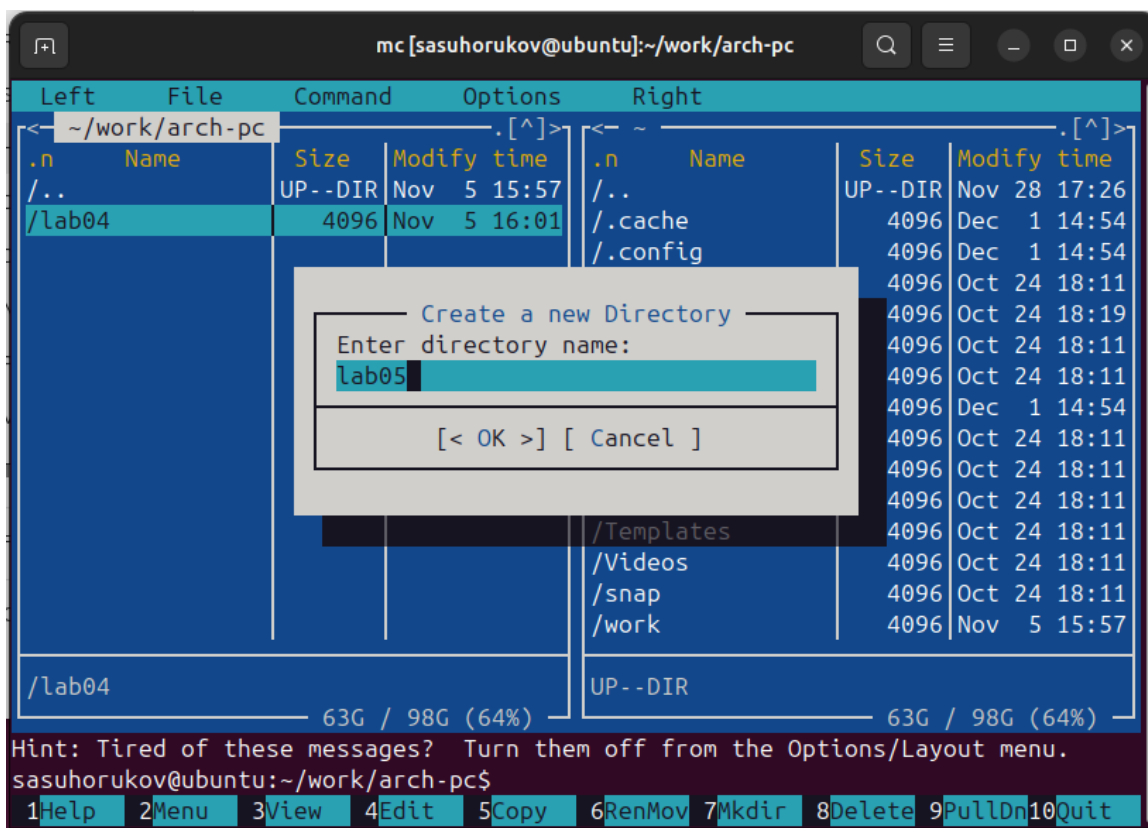


Рисунок 2.2: Создание каталога

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

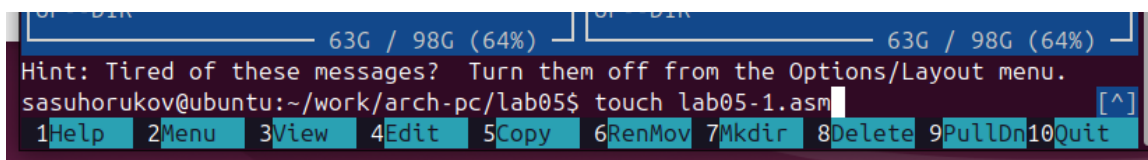


Рисунок 2.3: Создание файла `lab05-1.asm`

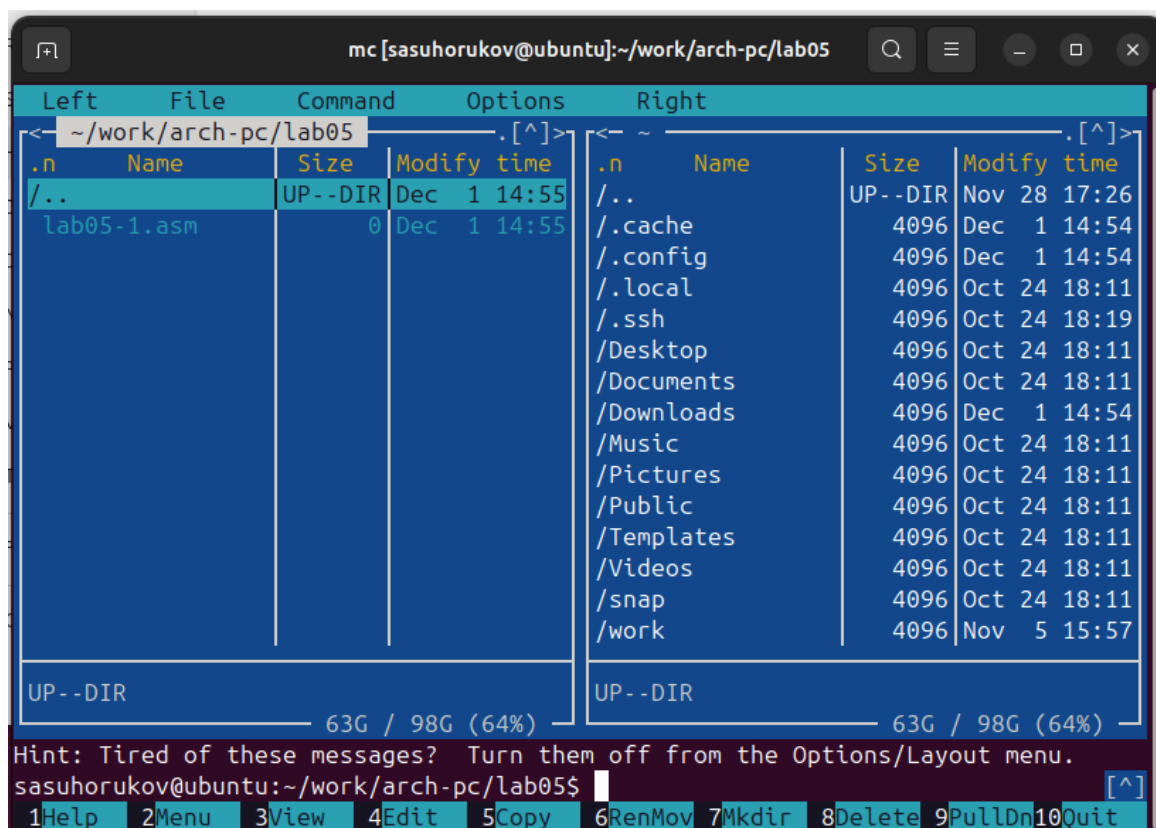
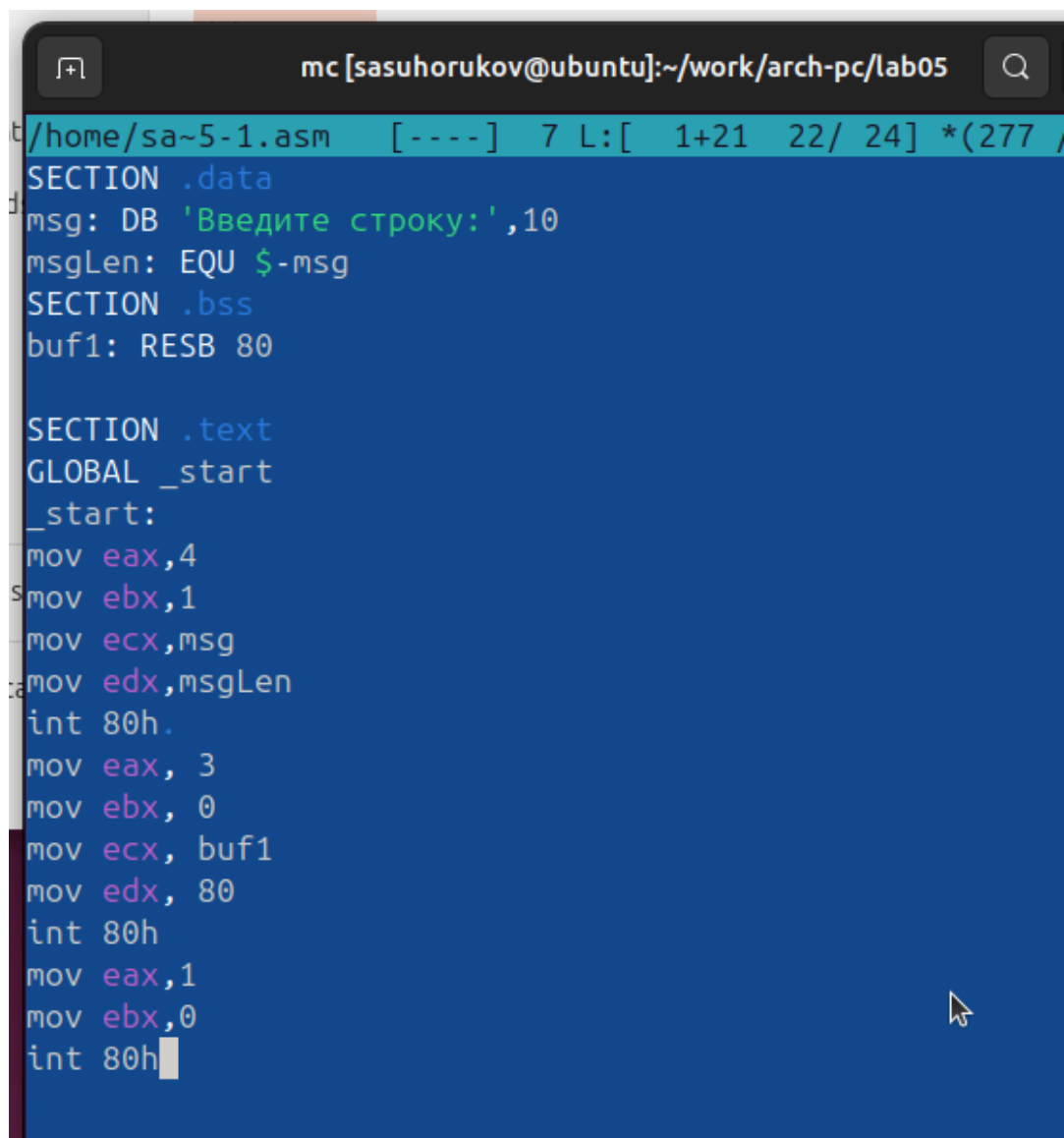


Рисунок 2.4: Создание файла lab05-1.asm

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

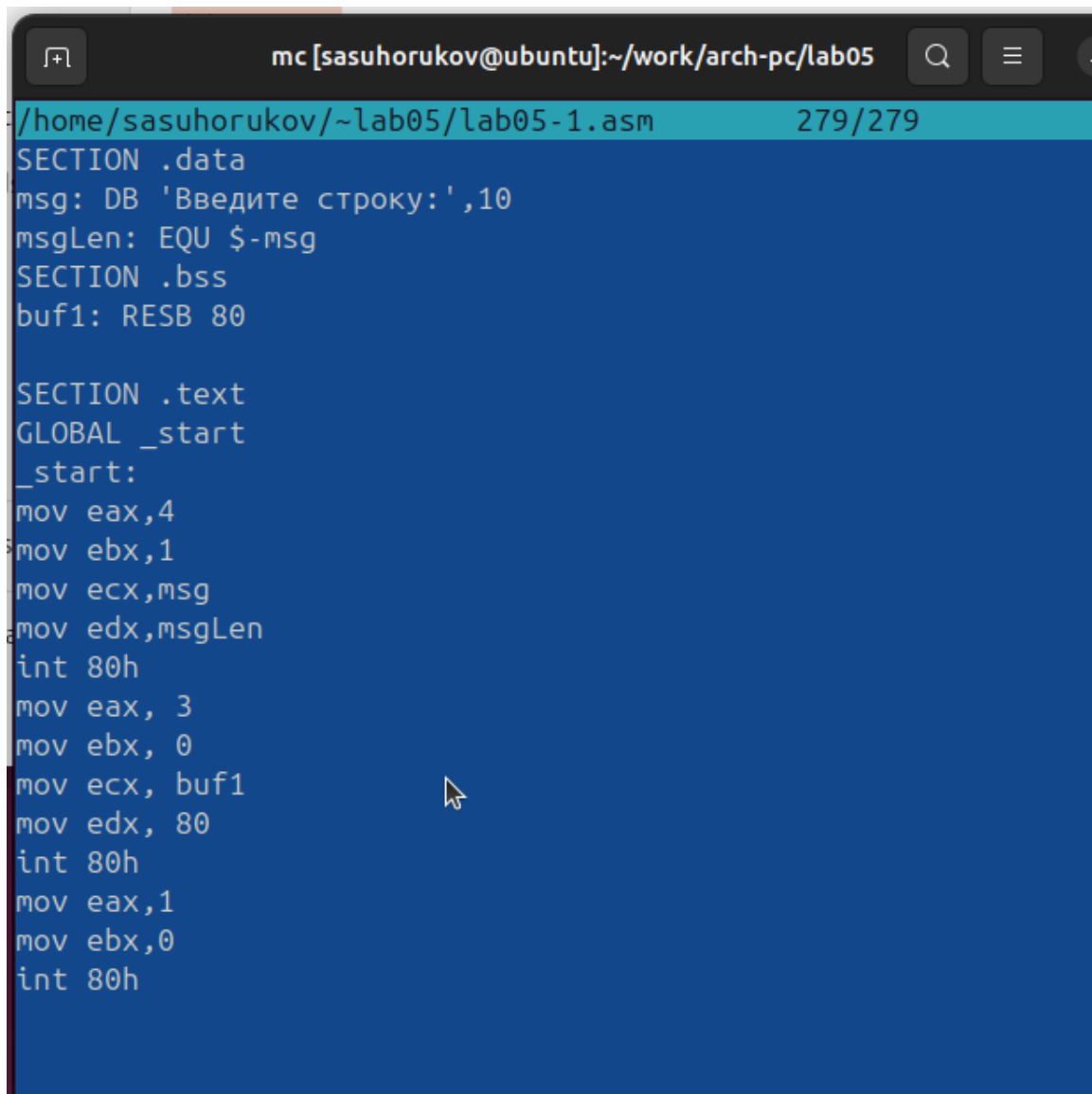


```
mc [sasuhorukov@ubuntu]:~/work/arch-pc/lab05
/home/sa~5-1.asm [ - - - ] 7 L: [ 1+21 22/ 24 ] *(277 /
SECTION .data
msg: DB 'Введите строку:',10
msgLen: EQU $-msg
SECTION .bss
buf1: RESB 80

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
```

Рисунок 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).

A screenshot of a code editor window. The title bar shows the path `mc [sasuhorukov@ubuntu]:~/work/arch-pc/lab05` and a search icon. The editor displays the file `/home/sasuhorukov/~lab05/lab05-1.asm` at line 279 of 279. The code is written in assembly and includes sections for data, bss, and text. The text section contains instructions for setting up a buffer and performing system calls.

```
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
```

Рисунок 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).

```
sasuhorukov@ubuntu:~/work/arch-pc/lab05$ nasm -f elf lab05-1.asm
sasuhorukov@ubuntu:~/work/arch-pc/lab05$ ld -m elf_i386 lab05-1.o -o lab05-1
sasuhorukov@ubuntu:~/work/arch-pc/lab05$ ./lab05-1
Введите строку:
linux
sasuhorukov@ubuntu:~/work/arch-pc/lab05$
```

Рисунок 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.

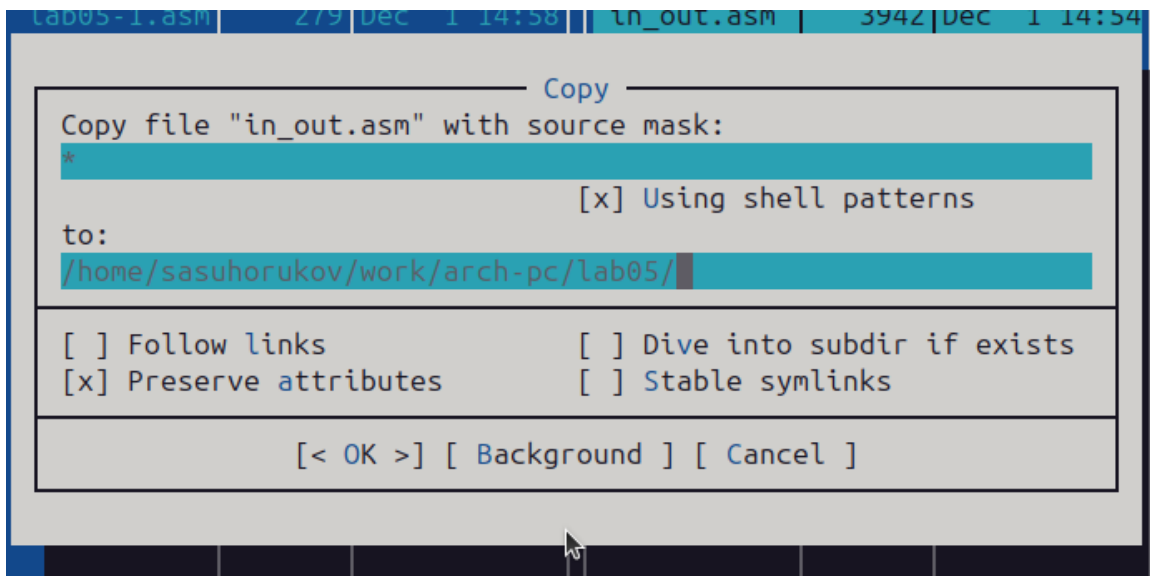


Рисунок 2.8: Копирование файла in_out.asm

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

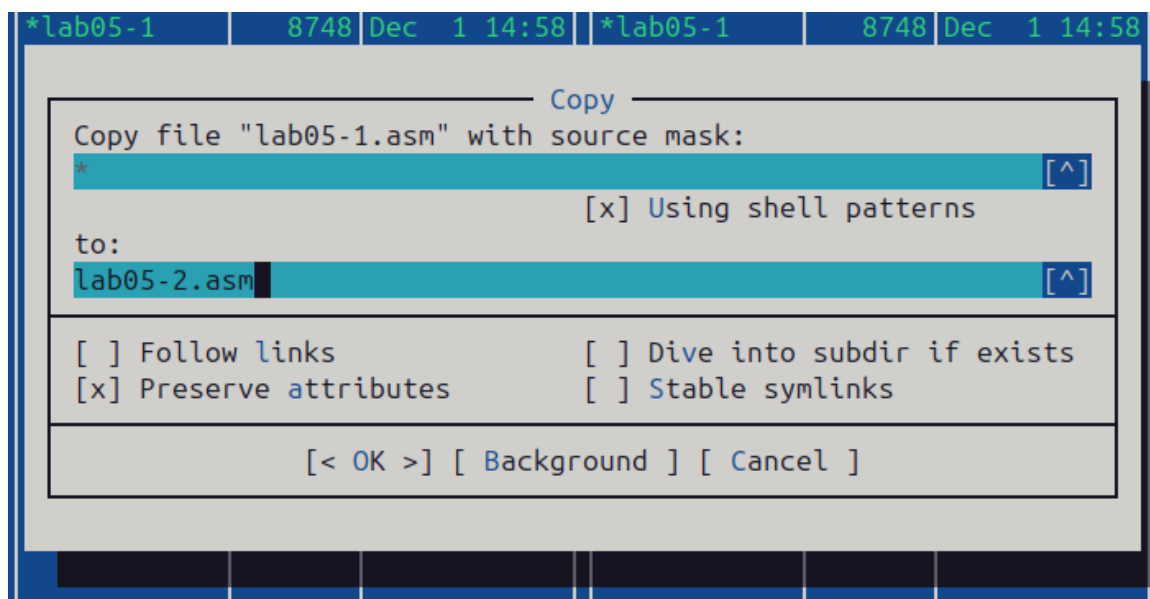
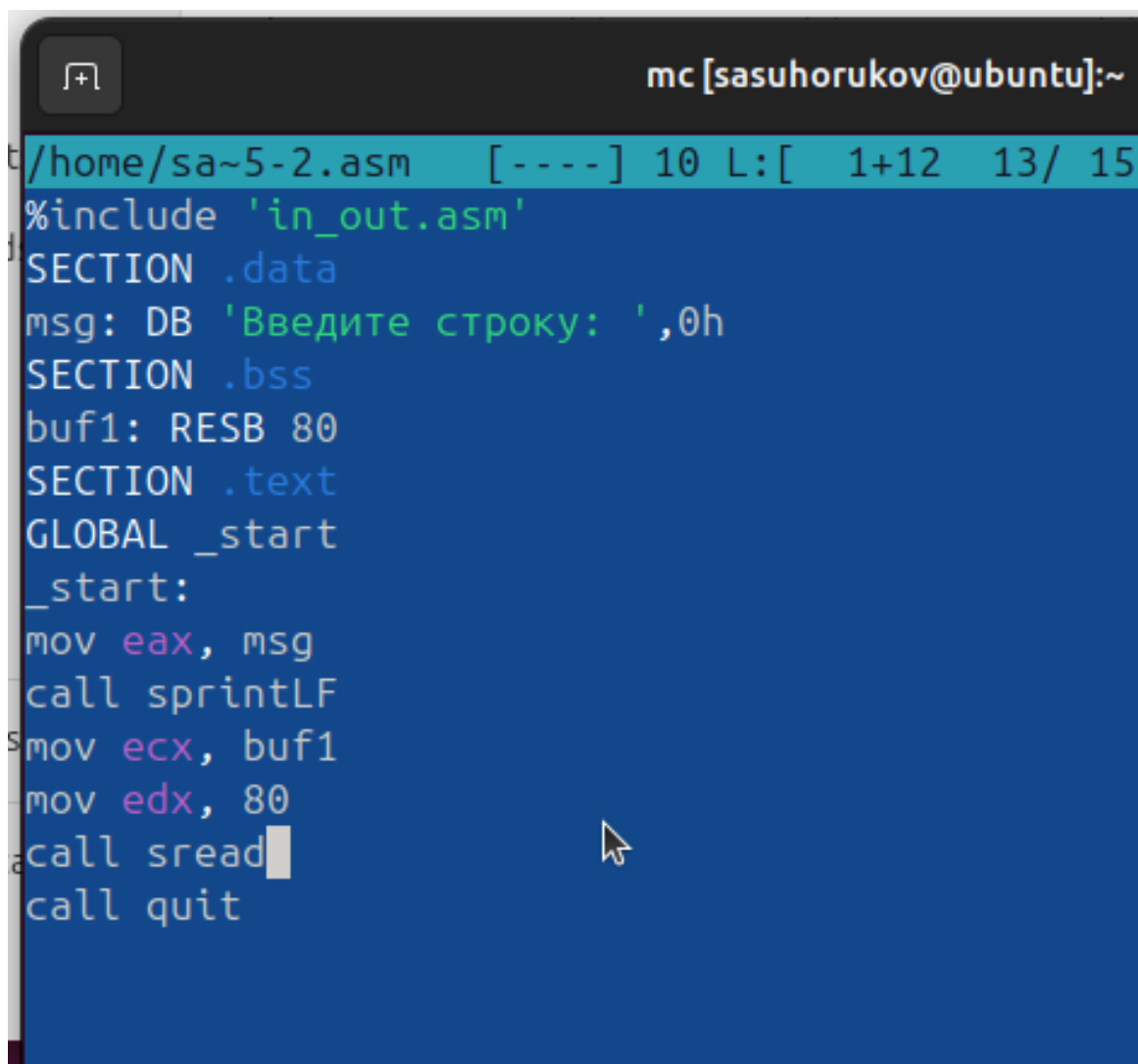


Рисунок 2.9: Копирование файла lab05-1.asm

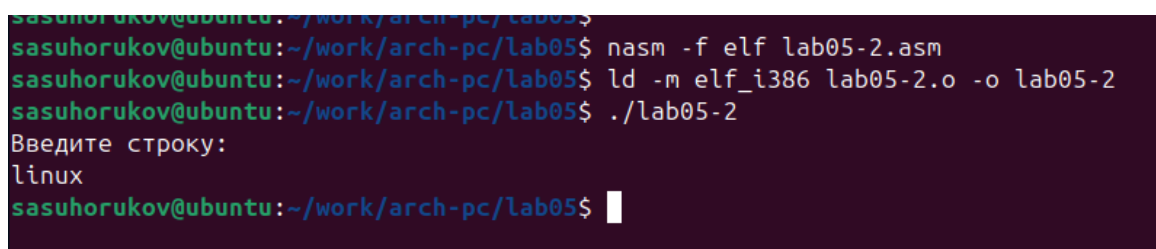
В новом файле lab05-2.asm пишу код программы с использованием подпрограмм из внешнего файла in_out.asm (рис. 2.10).



```
mc [sasuhorukov@ubuntu]:~  
t /home/sa~5-2.asm [----] 10 L:[ 1+12 13/ 15  
%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  
S mov ecx, buf1  
a mov edx, 80  
call sread  
call quit
```

Рисунок 2.10: Программа lab05-2.asm

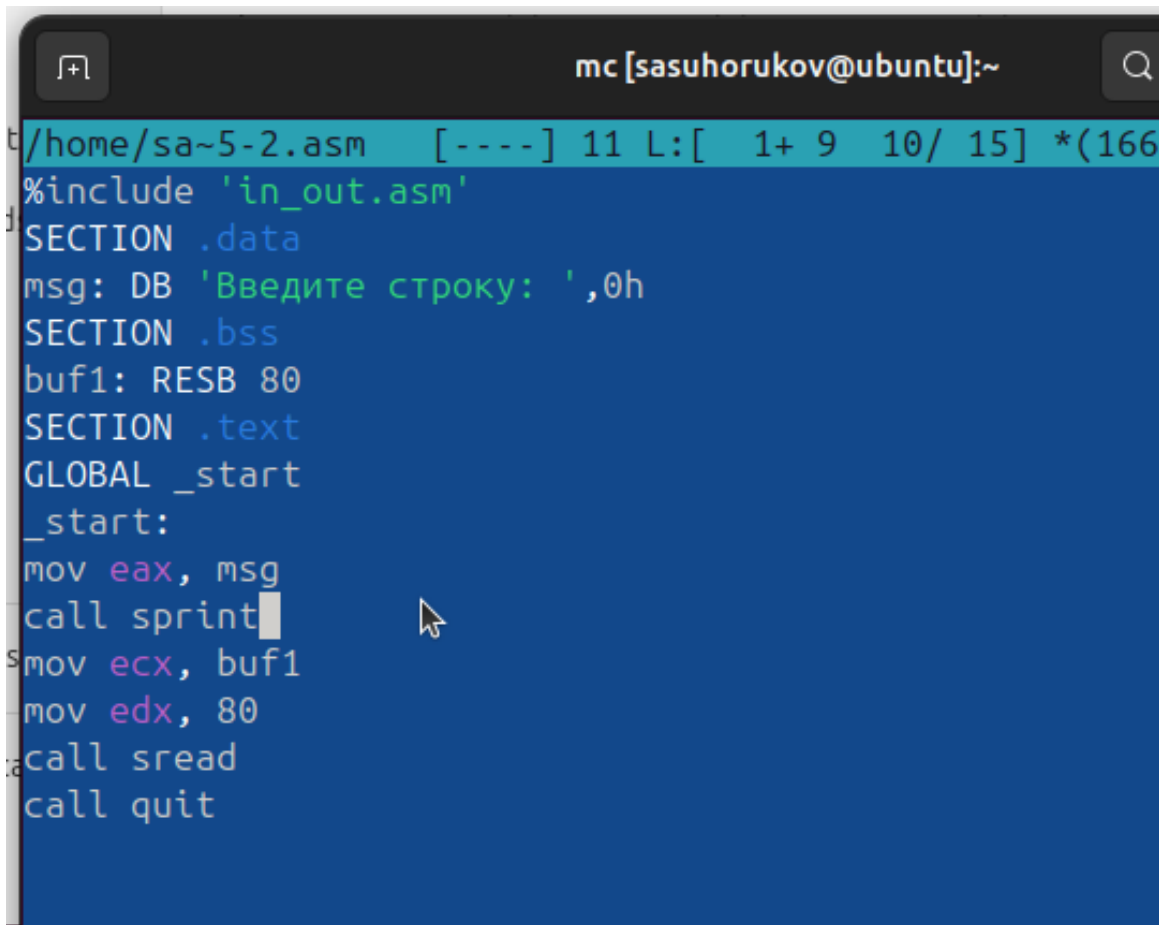
После компиляции программы я проверяю её запуск (рис. 2.11).



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

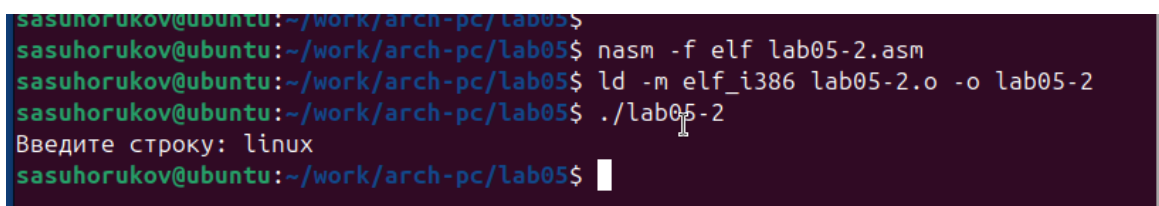
Рисунок 2.11: Запуск программы lab05-2.asm

В файле lab05-2.asm я заменяю подпрограмму sprintLF на sprint, после чего заново собираю исполняемый файл (рис. 2.12) (рис. 2.13).



```
mc [sasuhorukov@ubuntu]:~  
t/home/sa~5-2.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
```

Рисунок 2.12: Программа в файле lab05-2.asm



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

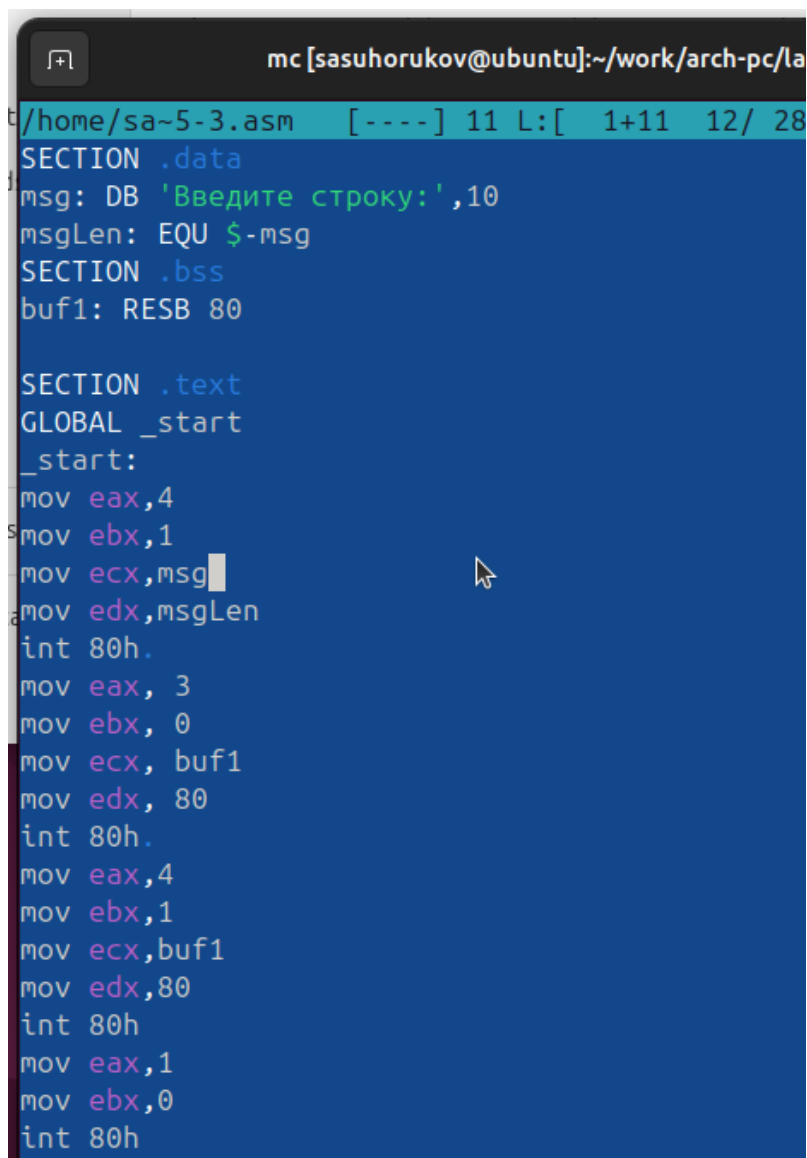
Рисунок 2.13: Запуск программы lab05-2.asm

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 введенную строку на экран.



```
mc [sasuhorukov@ubuntu]:~/work/arch-pc/la
/home/sa~5-3.asm [----] 11 L:[ 1+11 12/ 28
SECTION .data
msg: DB 'Введите строку:',10
msgLen: EQU $-msg
SECTION .bss
buf1: RESB 80

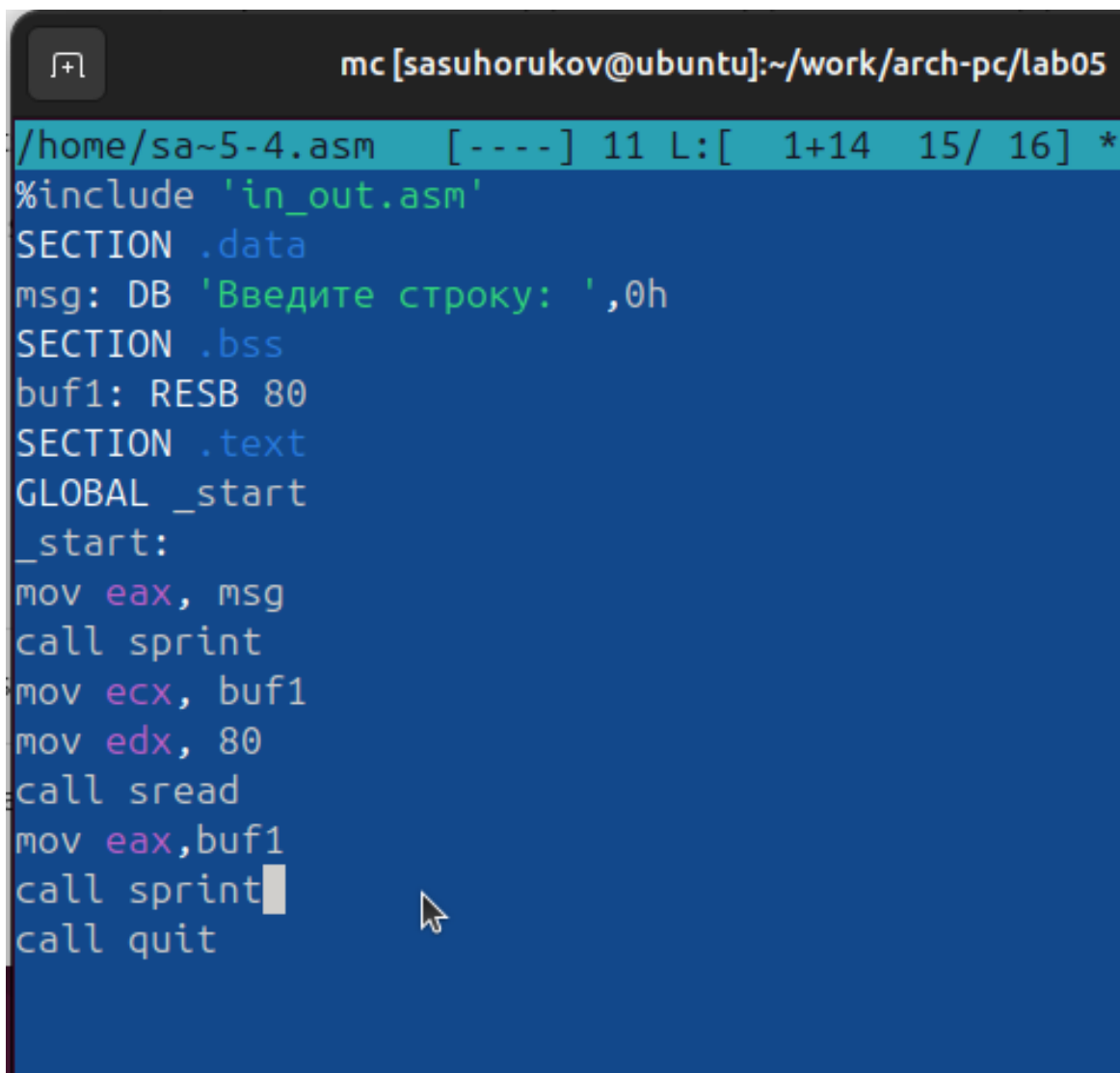
SECTION .text
GLOBAL _start
_start:
mov eax,4
mov ebx,1
mov ecx,msg
mov edx,msgLen
int 80h.
mov eax, 3
mov ebx, 0
mov ecx, buf1
mov edx, 80
int 80h.
mov eax,4
mov ebx,1
mov ecx,buf1
mov edx,80
int 80h
mov eax,1
mov ebx,0
int 80h
```

Рисунок 2.14: Программа lab05-3.asm

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

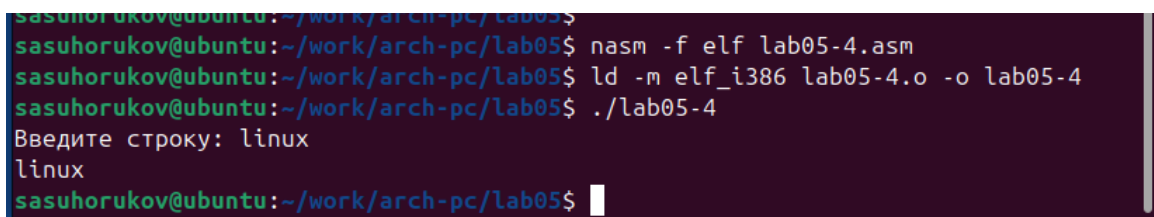
Рисунок 2.15: Запуск программы lab05-3.asm

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).



```
mc [sasuhorukov@ubuntu]:~/work/arch-pc/lab05
/home/sa~5-4.asm [----] 11 L:[ 1+14 15/ 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 sprint
mov ecx, buf1
mov edx, 80
call sread
mov eax, buf1
call sprint
call quit
```

Рисунок 2.16: Программа lab05-4.asm



```
sasuhorukov@ubuntu:~/work/arch-pc/lab05$
sasuhorukov@ubuntu:~/work/arch-pc/lab05$ nasm -f elf lab05-4.asm
sasuhorukov@ubuntu:~/work/arch-pc/lab05$ ld -m elf_i386 lab05-4.o -o lab05-4
sasuhorukov@ubuntu:~/work/arch-pc/lab05$ ./lab05-4
Введите строку: linux
linux
sasuhorukov@ubuntu:~/work/arch-pc/lab05$
```

Рисунок 2.17: Запуск программы lab05-4.asm

3 Выводы

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