

Отчёт по лабораторной работе 5

Архитектура компьютеров

Амир Расули

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	16
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	выбираю редактор	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	12
2.9	Копирование файла lab05-1.asm	13
2.10	Программа lab05-2.asm	14
2.11	Запуск программы lab05-2.asm	14
2.12	Программа в файле lab05-2.asm	15
2.13	Запуск программы lab05-2.asm	15
2.14	Программа lab05-3.asm	16
2.15	Запуск программы lab05-3.asm	17
2.16	Программа lab05-4.asm	17
2.17	Запуск программы lab05-4.asm	18

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

Запускаю Midnight Commander (см. рис. 2.1), используя клавиши со стрелками и Enter, перехожу в каталог ~/work/arch-pc. Затем нажимаю F7 для создания нового каталога под названием lab05 (см. рис. 2.2).

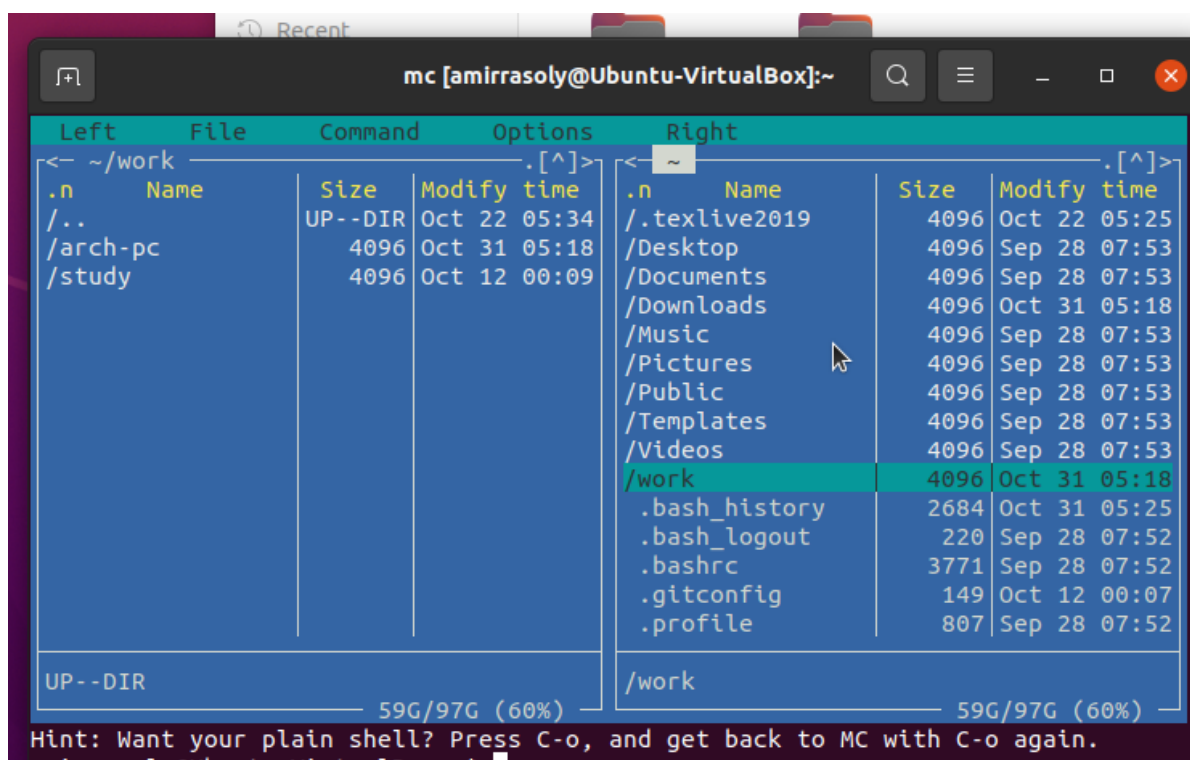


Рис. 2.1: Запуск Midnight Commander

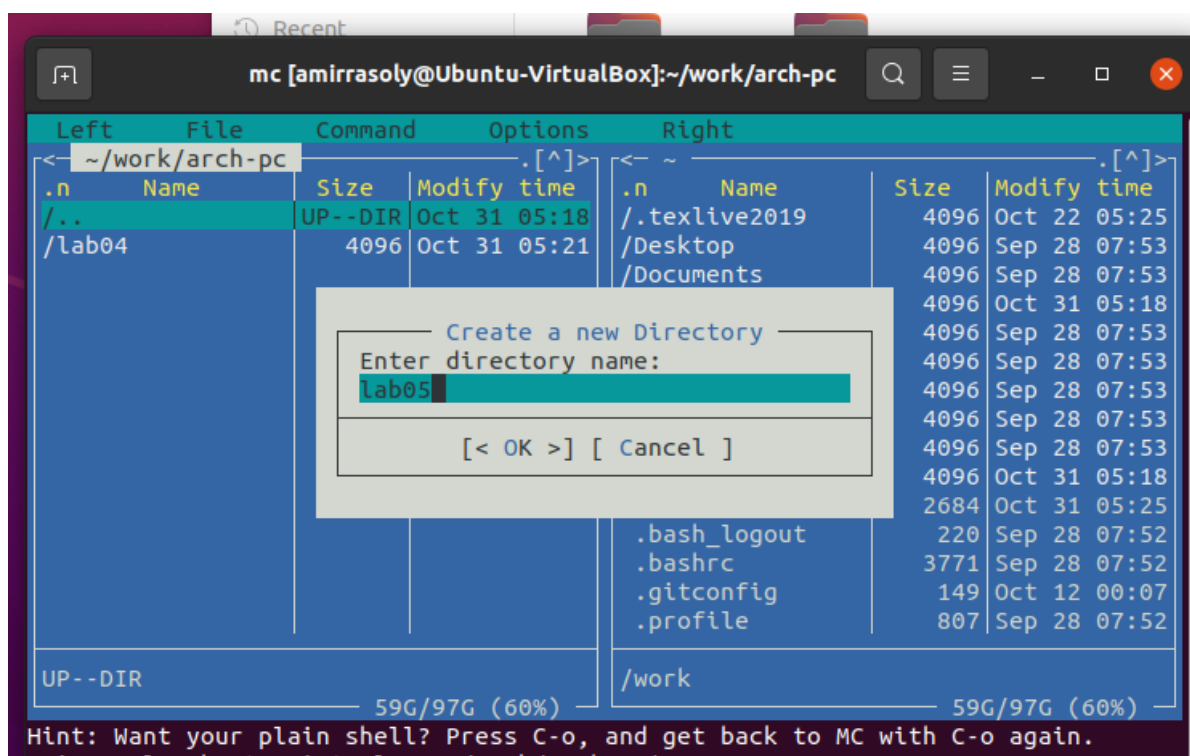
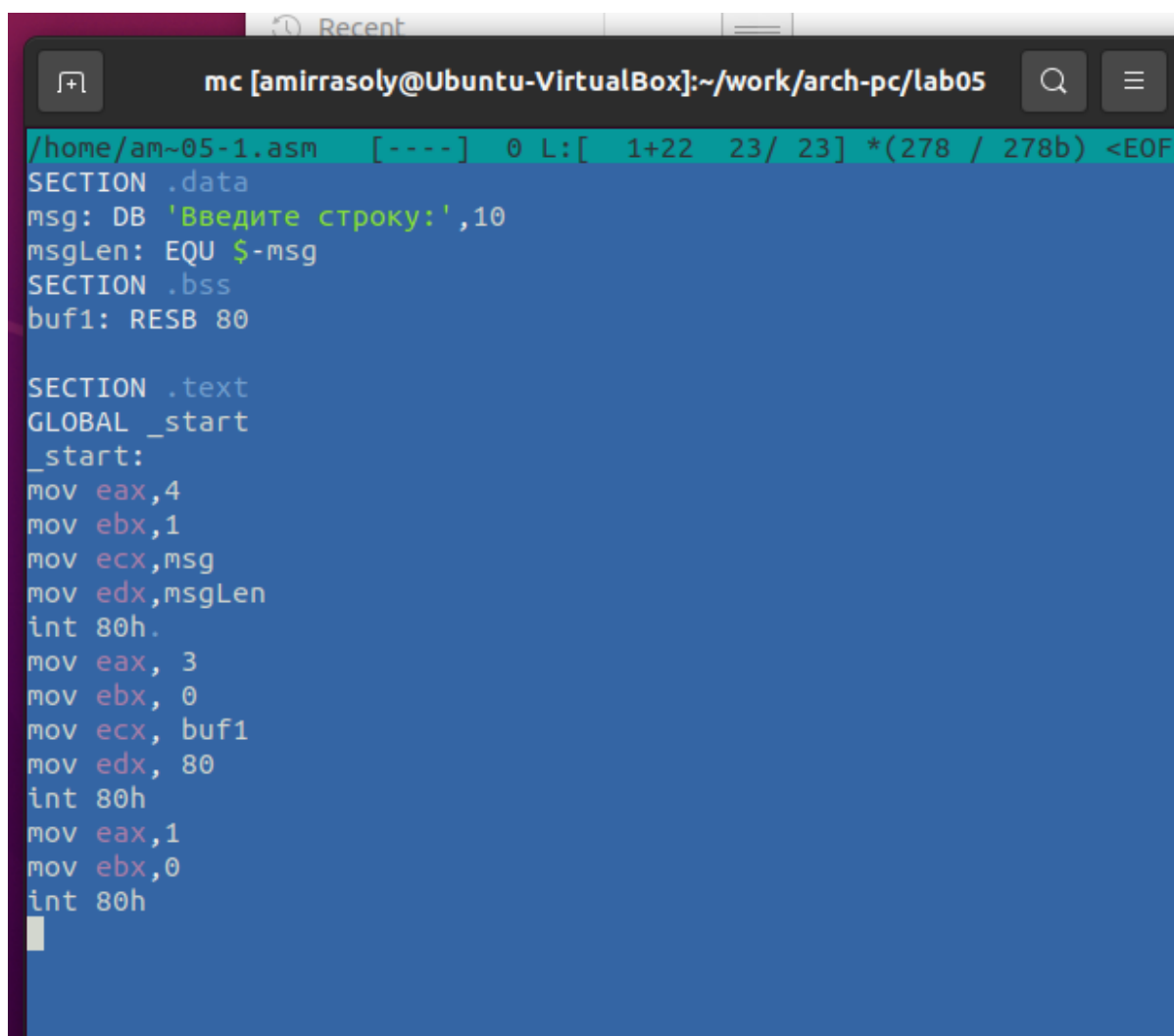


Рис. 2.2: Создание каталога

С помощью команды `touch` создаю файл `lab05-1.asm` (см. рис. 2.3).

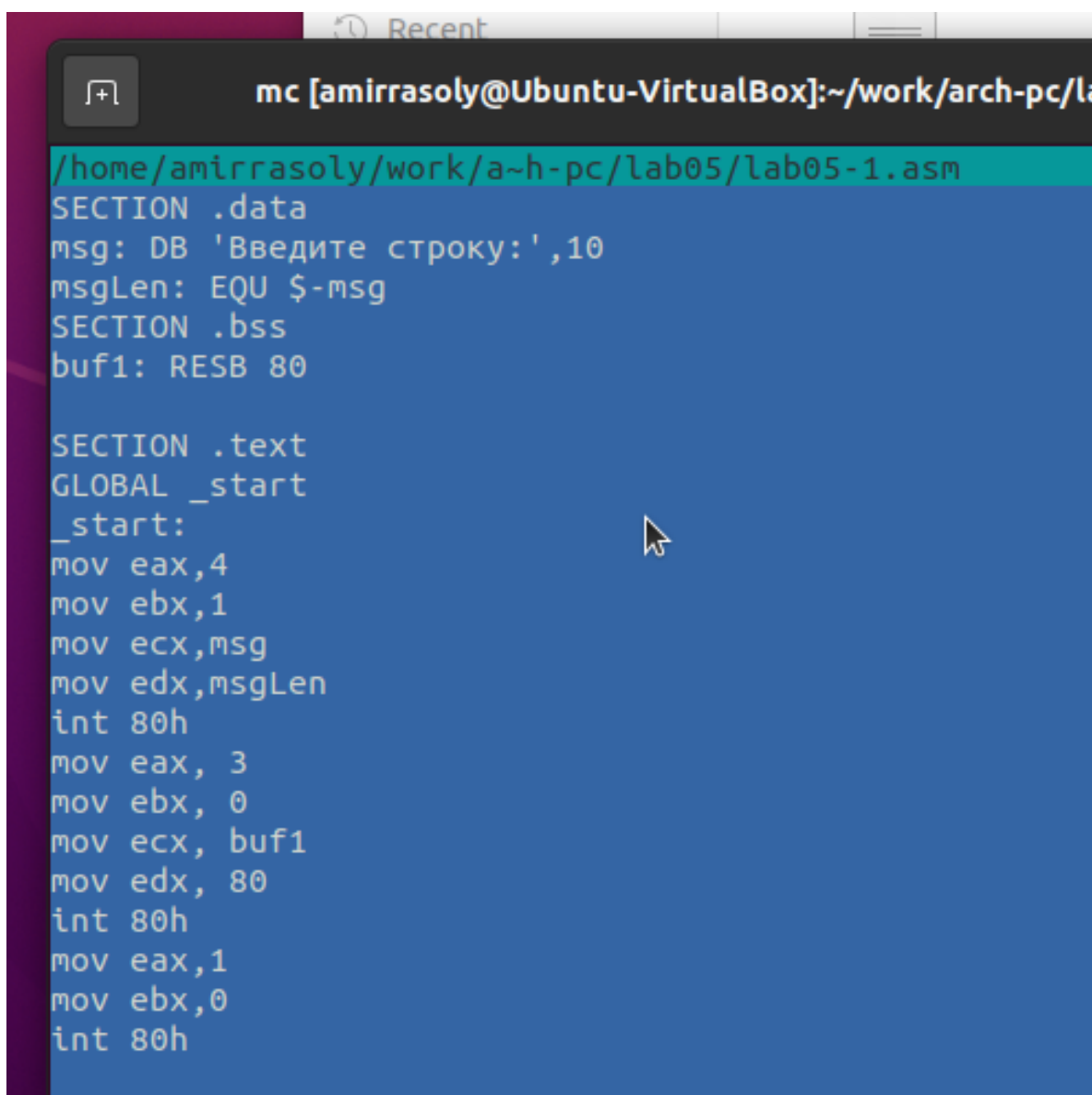


```
mc [amirrasoly@Ubuntu-VirtualBox]:~/work/arch-pc/lab05
/home/am~05-1.asm  [ - - - - ]  0 L: [ 1+22 23/ 23 ] *(278 / 278b) <EOF
SECTION .data
msg: DB 'Введите строку:',10
msgLen: EQU $-msg
SECTION .bss
buf1: RESB 80

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 содержимого файла открываю его на просмотр, нажав F3, и убеждаюсь, что код написан верно (см. рис. 2.6).



```
mc [amirrasoly@Ubuntu-VirtualBox]:~/work/arch-pc/la
/home/amirrasoly/work/a~h-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

Транслирую файл программы в объектный файл, а затем выполняю компоновку, в результате чего получаю исполняемый файл программы (см. рис. 2.7).

```
amirrasoly@Ubuntu-VirtualBox:~/work/arch-pc/lab05$ nasm -f elf lab05-1.asm
amirrasoly@Ubuntu-VirtualBox:~/work/arch-pc/lab05$ ld -m elf_i386 lab05-1.o -o lab05-1
amirrasoly@Ubuntu-VirtualBox:~/work/arch-pc/lab05$ ./lab05-1
Введите строку:
Amir
amirrasoly@Ubuntu-VirtualBox:~/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 размещаю его в рабочем каталоге (см. рис. 2.8). Для копирования файла использую клавишу F5, а для перемещения — клавишу F6.

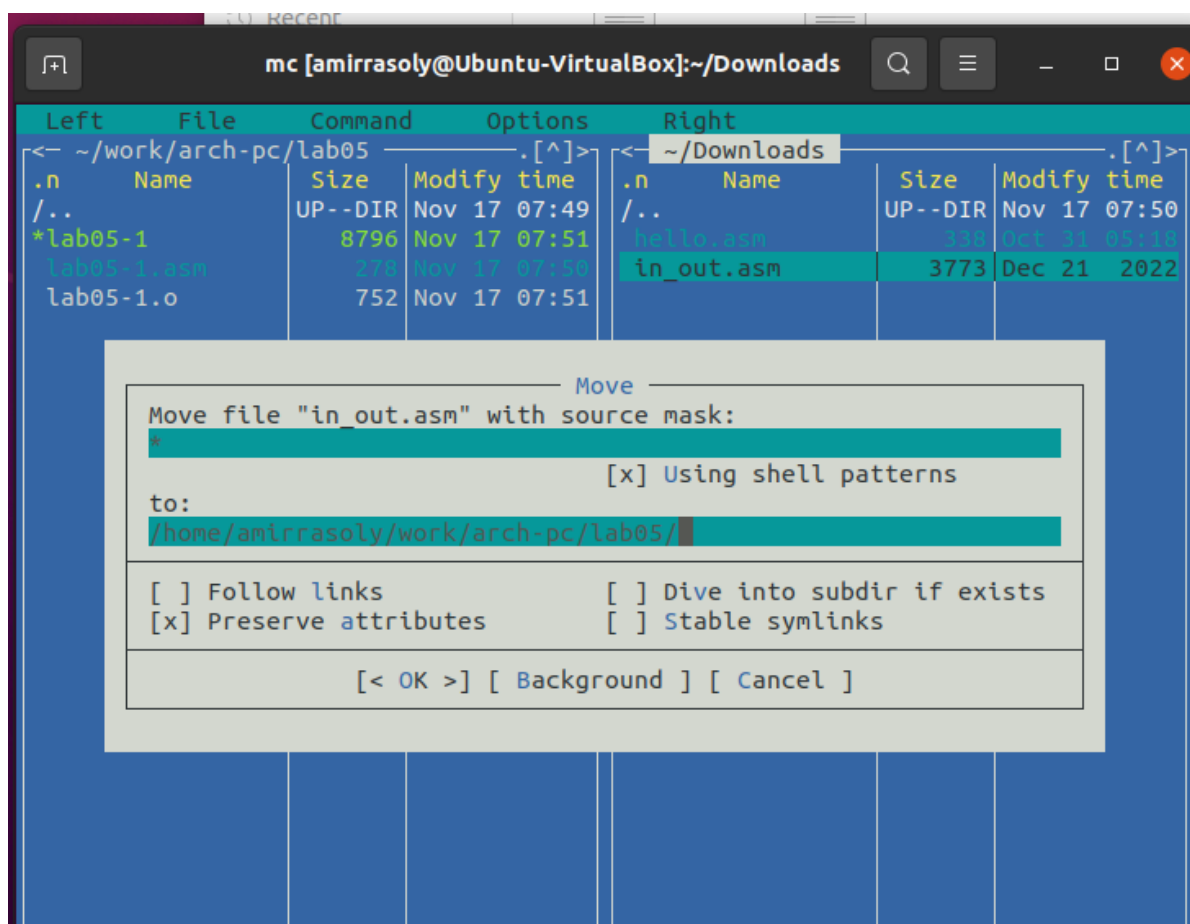


Рис. 2.8: Копирование файла in_out.asm

Копирую файл lab05-1.asm, создавая его копию под именем lab05-2.asm (см. рис. 2.9).

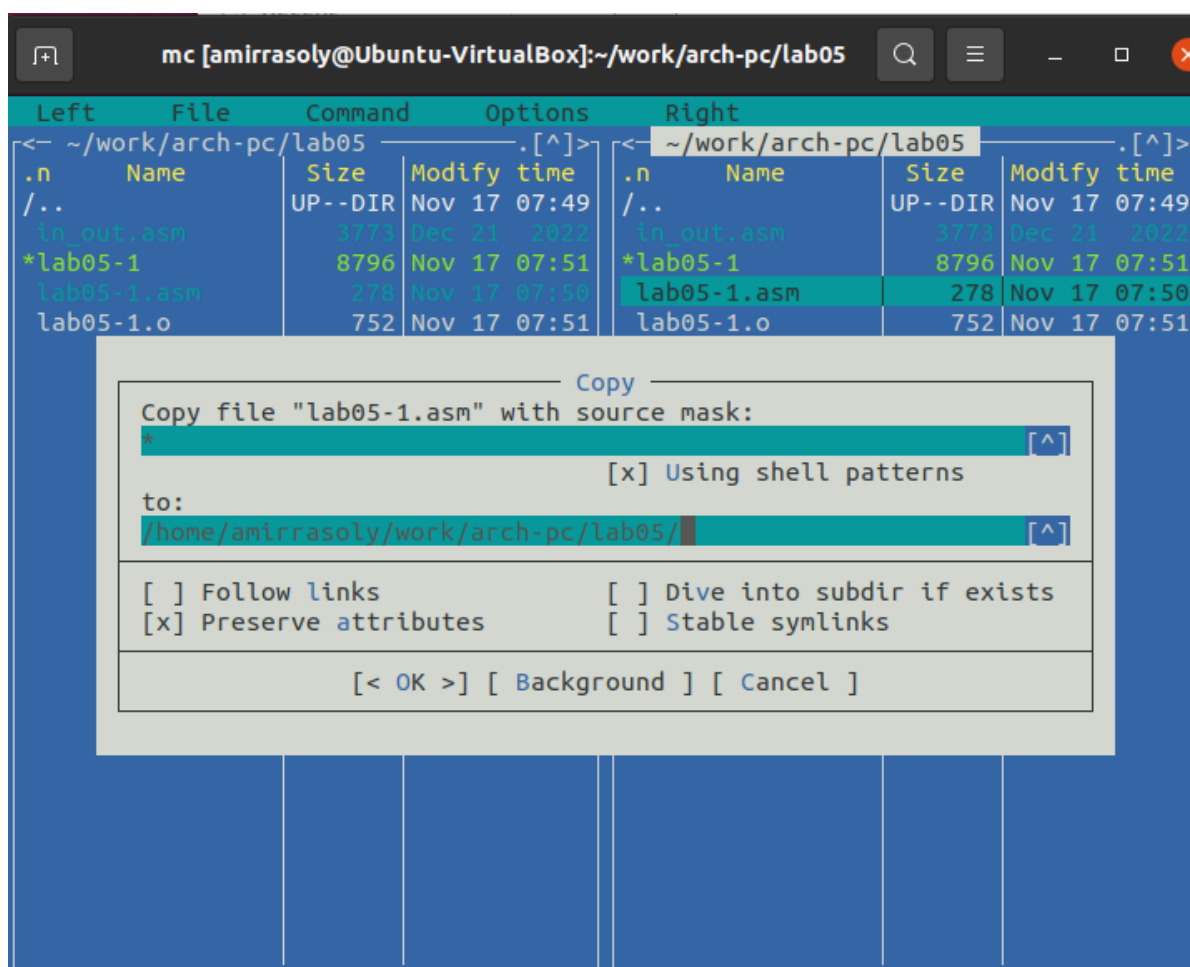
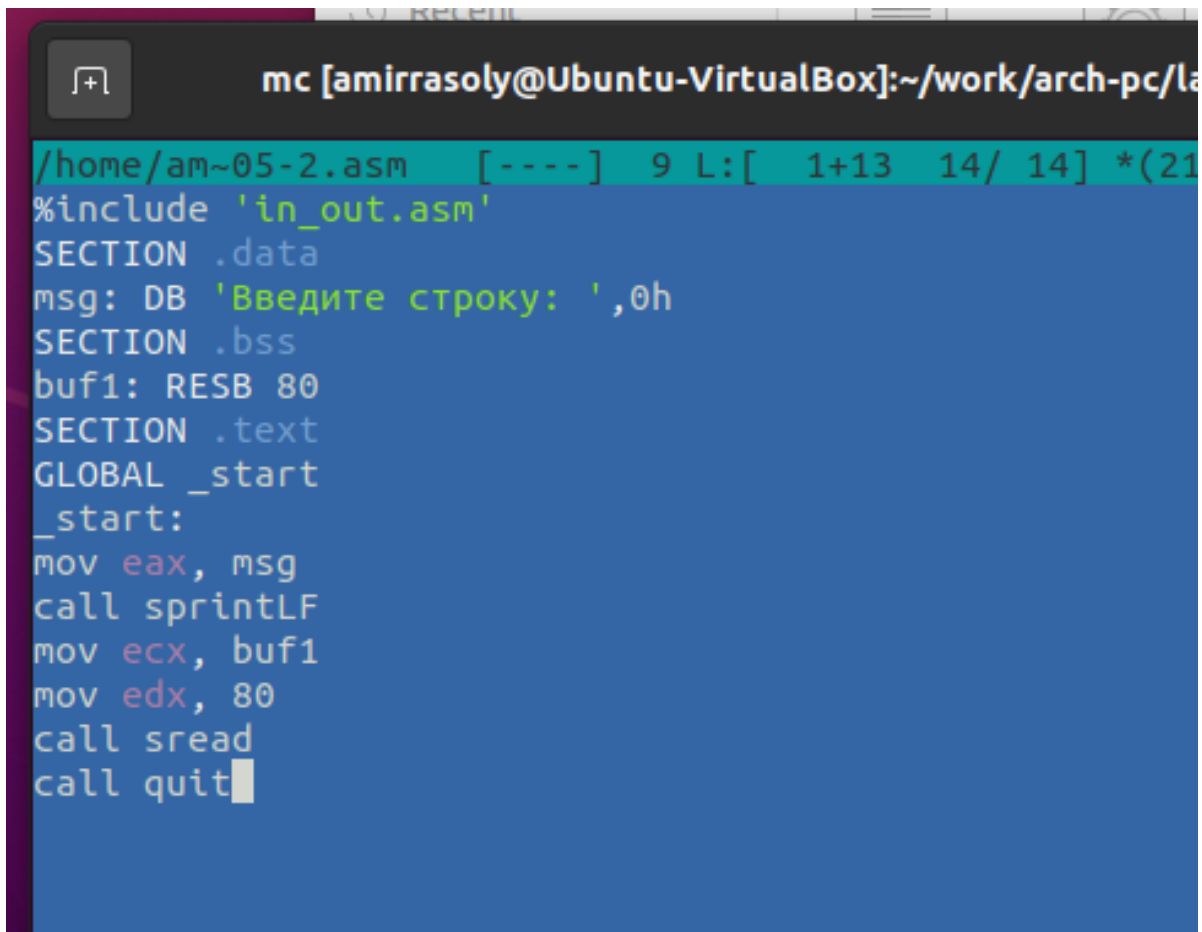


Рис. 2.9: Копирование файла lab05-1.asm

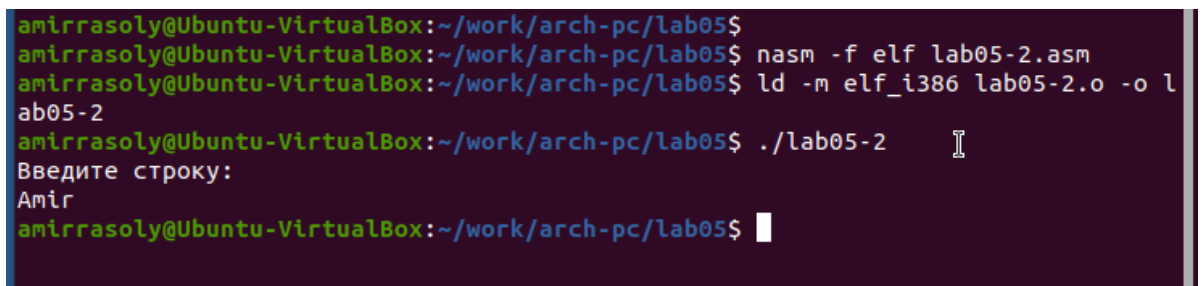
Пишу код для программы lab05-2.asm, используя подпрограммы из внешнего файла in_out.asm (см. рис. 2.10).



```
mc [amirrasoly@Ubuntu-VirtualBox]:~/work/arch-pc/la
/home/am~05-2.asm [----] 9 L:[ 1+13 14/ 14] *(21
#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

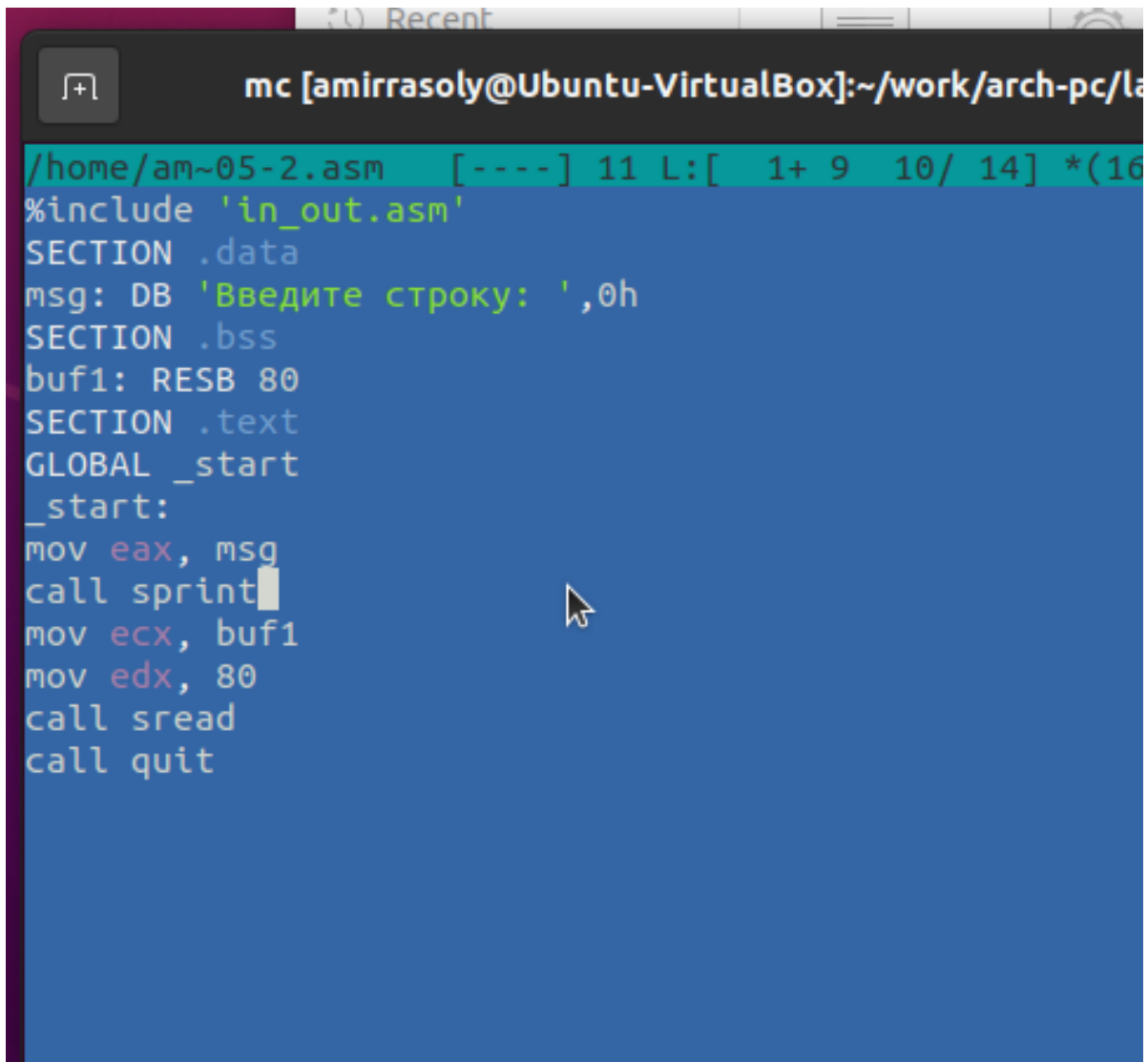
Компилирую программу и проверяю её запуск (см. рис. 2.11).



```
amirrasoly@Ubuntu-VirtualBox:~/work/arch-pc/lab05$
amirrasoly@Ubuntu-VirtualBox:~/work/arch-pc/lab05$ nasm -f elf lab05-2.asm
amirrasoly@Ubuntu-VirtualBox:~/work/arch-pc/lab05$ ld -m elf_i386 lab05-2.o -o l
ab05-2
amirrasoly@Ubuntu-VirtualBox:~/work/arch-pc/lab05$ ./lab05-2
Введите строку:
Amir
amirrasoly@Ubuntu-VirtualBox:~/work/arch-pc/lab05$
```

Рис. 2.11: Запуск программы lab05-2.asm

В файле lab05-2.asm заменяю подпрограмму sprintLF на sprint. После этого заново собираю исполняемый файл (см. рис. 2.12 и 2.13).

A screenshot of a code editor window titled 'mc [amirrasoly@Ubuntu-VirtualBox]:~/work/arch-pc/lab05-2.asm'. The editor shows assembly code for a program. The code includes a header file 'in_out.asm', defines a data section with a message 'Введите строку: ',0h, and a bss section with a buffer 'buf1' of size 80. The text section starts at '_start' and contains instructions to move the message to 'eax', call 'sprint', move the buffer to 'ecx', set 'edx' to 80, call 'sread', and finally call 'quit'.

```
mc [amirrasoly@Ubuntu-VirtualBox]:~/work/arch-pc/lab05-2.asm [----] 11 L:[ 1+ 9 10/ 14] *(16
/home/am~05-2.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. 2.12: Программа в файле lab05-2.asm

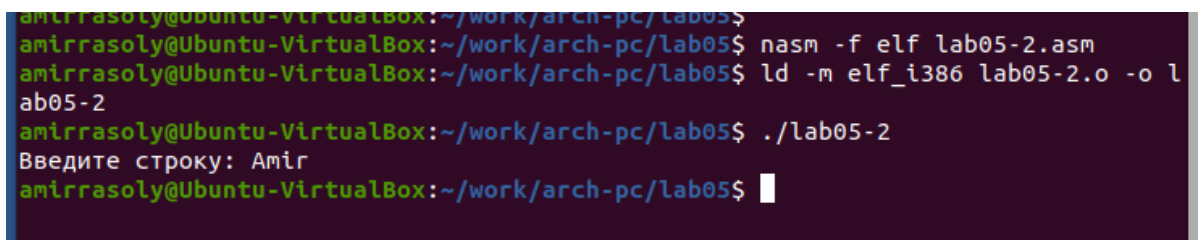
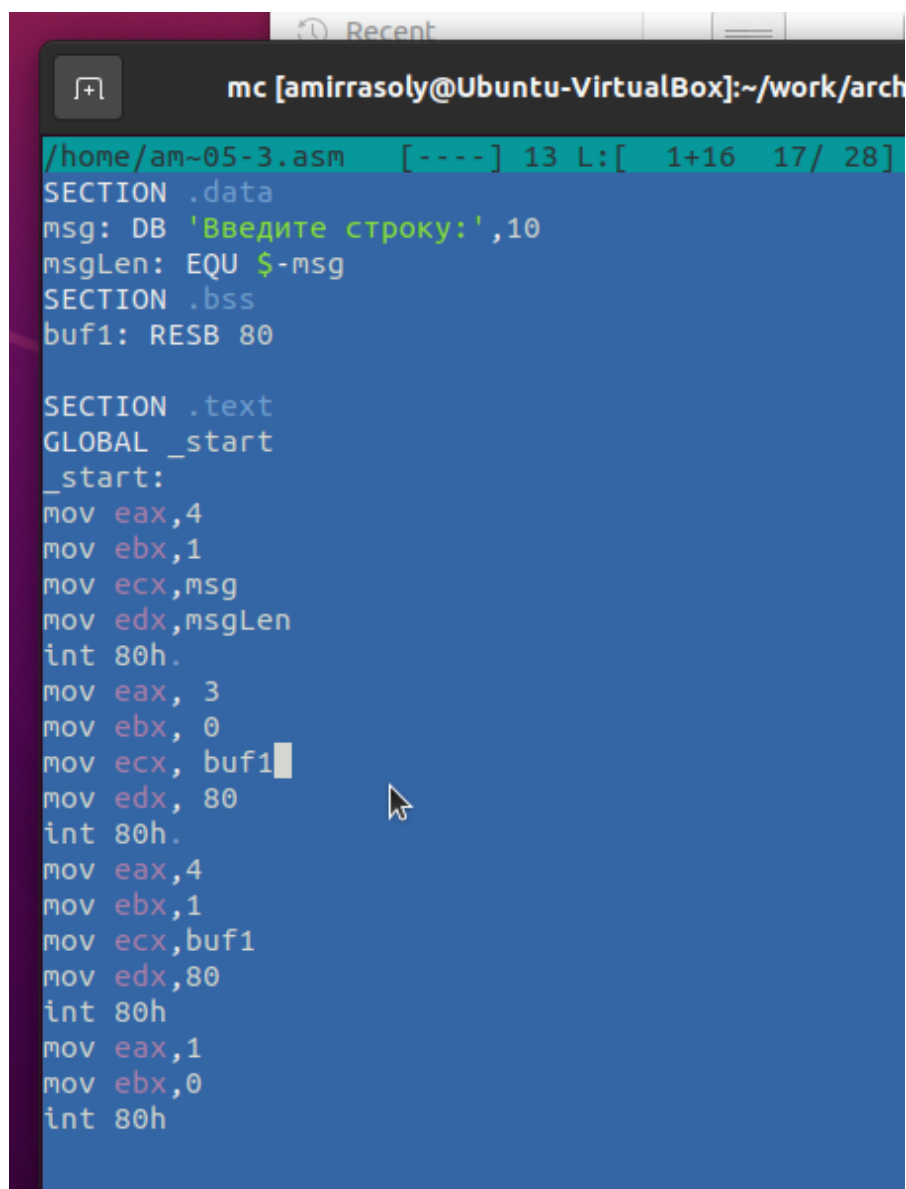
A screenshot of a terminal window showing the compilation and execution of the assembly program. The user runs 'nasm -f elf lab05-2.asm' to compile the assembly file into an object file 'lab05-2.o'. Then, they run 'ld -m elf_i386 lab05-2.o -o lab05-2' to link the object file into an executable 'lab05-2'. Finally, they run './lab05-2' to execute the program. The program prompts 'Введите строку: ' and the user enters 'Amir'.

Рис. 2.13: Запуск программы lab05-2.asm

Теперь программа выводит строку без перехода на новую строку в конце.

2.3 Задание для самостоятельной работы

Копирую программу lab05-1.asm и модифицирую код, чтобы она работала по следующему алгоритму (см. рис. 2.14 и 2.15): - выводит приглашение “Введите строку:”; - принимает строку с клавиатуры; - отображает введенную строку на экране.



```
mc [amirrasoly@Ubuntu-VirtualBox]:~/work/arch
/home/am~05-3.asm [----] 13 L:[ 1+16 17/ 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
```

Рис. 2.14: Программа lab05-3.asm

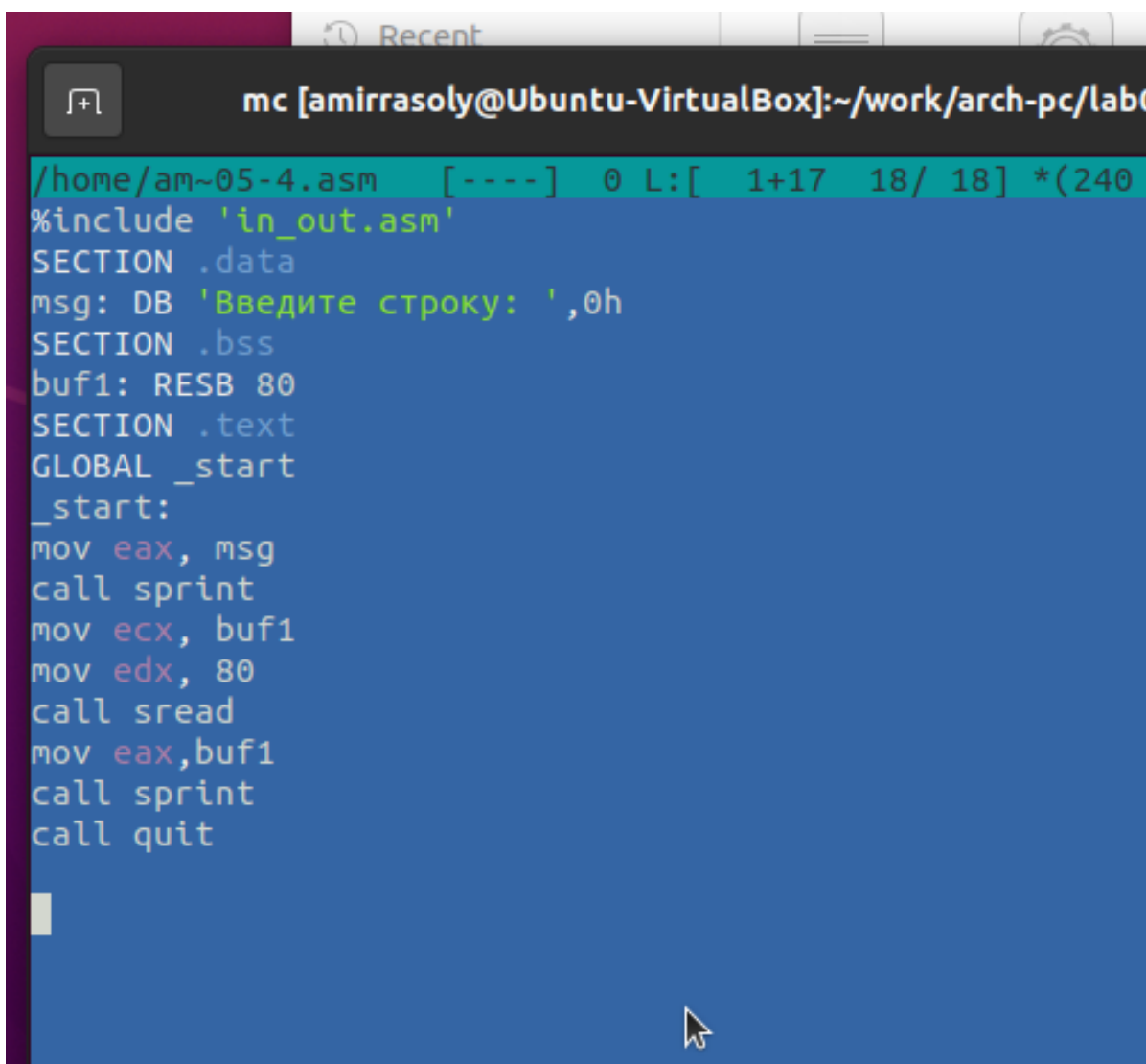

```

amirrasoly@Ubuntu-VirtualBox:~/work/arch-pc/lab05$
amirrasoly@Ubuntu-VirtualBox:~/work/arch-pc/lab05$ nasm -f elf lab05-3.asm
amirrasoly@Ubuntu-VirtualBox:~/work/arch-pc/lab05$ ld -m elf_i386 lab05-3.o -o lab05-3
amirrasoly@Ubuntu-VirtualBox:~/work/arch-pc/lab05$ ./lab05-3
Введите строку:
Amir
Amir
amirrasoly@Ubuntu-VirtualBox:~/work/arch-pc/lab05$
amirrasoly@Ubuntu-VirtualBox:~/work/arch-pc/lab05$

```

Рис. 2.15: Запуск программы lab05-3.asm

Аналогично, копирую программу lab05-2.asm и изменяю код, теперь используя подпрограммы из файла in_out.asm (см. рис. 2.16 и 2.17).



```

mc [amirrasoly@Ubuntu-VirtualBox]:~/work/arch-pc/lab0
/home/am~05-4.asm [----] 0 L:[ 1+17 18/ 18] *(240
%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. 2.16: Программа lab05-4.asm

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