

Отчёт по лабораторной работе 5

Архитектура компьютеров

Исмаил М. А. НКАбд-03-24

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	14
4	Выводы	17

Список иллюстраций

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

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 мощный файловый менеджер, который предоставляет пользователям удобный способ навигации по структуре каталогов и выполнения основных операций с файлами в файловой системе. Эта программа значительно упрощает работу с файлами, делая её более интуитивной и визуально понятной.

Программа, написанная на языке ассемблера NASM, обычно состоит из трёх основных секций: - **Секция кода программы (SECTION .text)**: здесь располагается исполняемый код. - **Секция инициализированных данных (SECTION .data)**: в этой секции содержатся данные, известные на момент компиляции. - **Секция неинициализированных данных (SECTION .bss)**: в этой секции резервируется память для данных, значения которых присваиваются во время выполнения программы.

3 Выполнение лабораторной работы

3.1 Знакомство с Midnight Commander

Запускаю Midnight Commander (см. рис. 3.1), используя клавиши со стрелками и Enter, перехожу в каталог `~/work/arch-rc`. Затем нажимаю F7 для создания нового каталога под названием `lab05` (см. рис. 3.2).

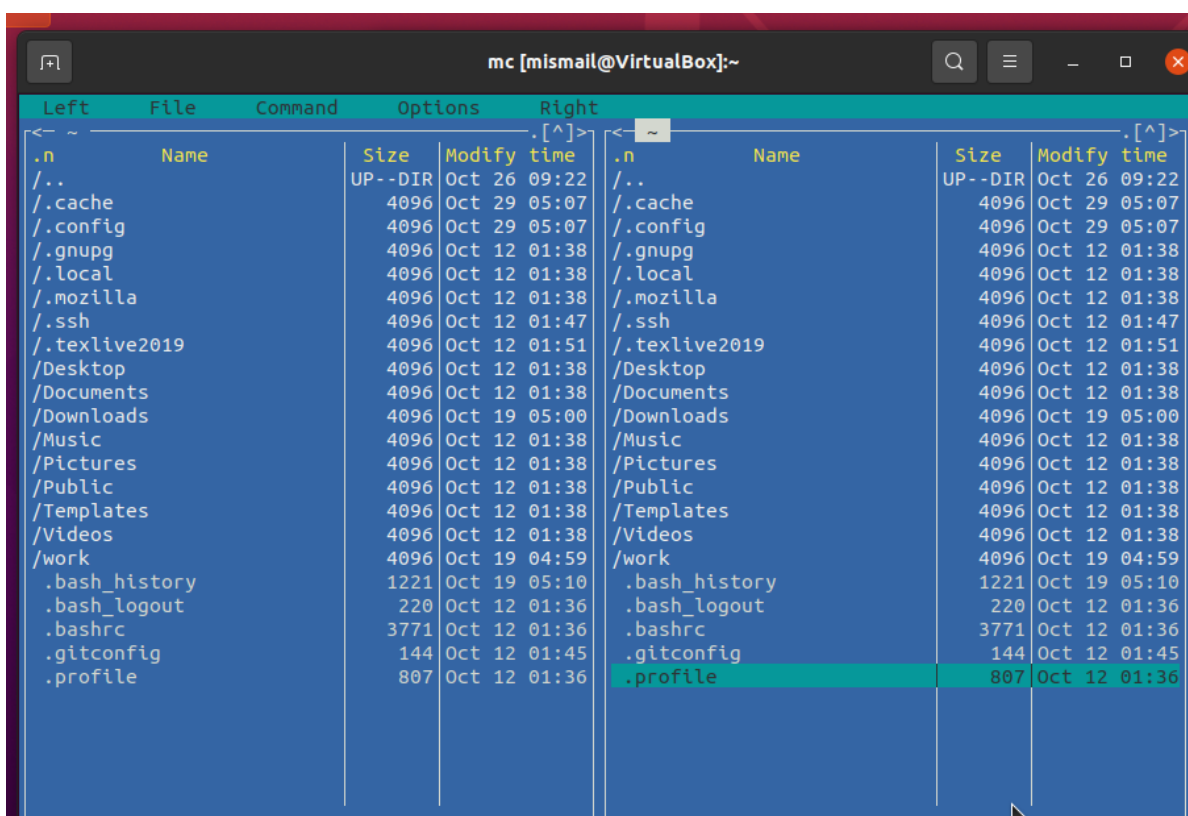


Рис. 3.1: Запуск Midnight Commander

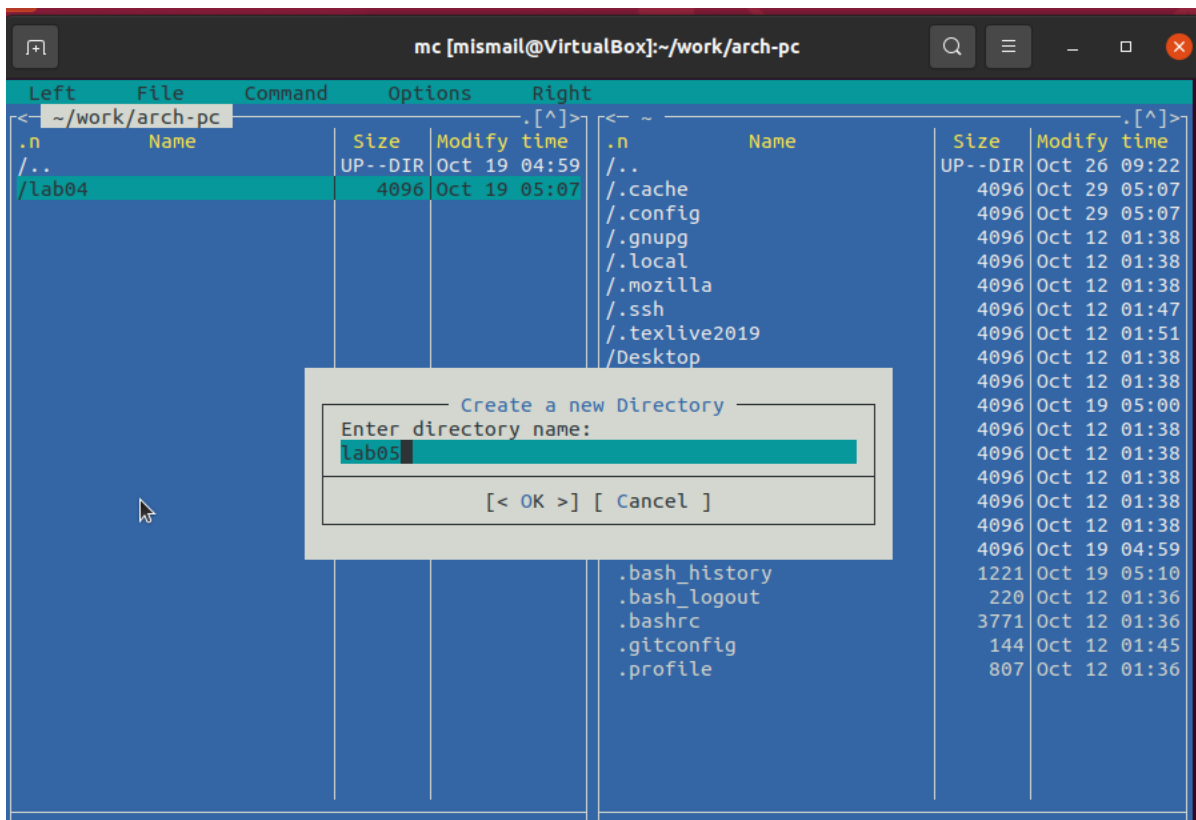
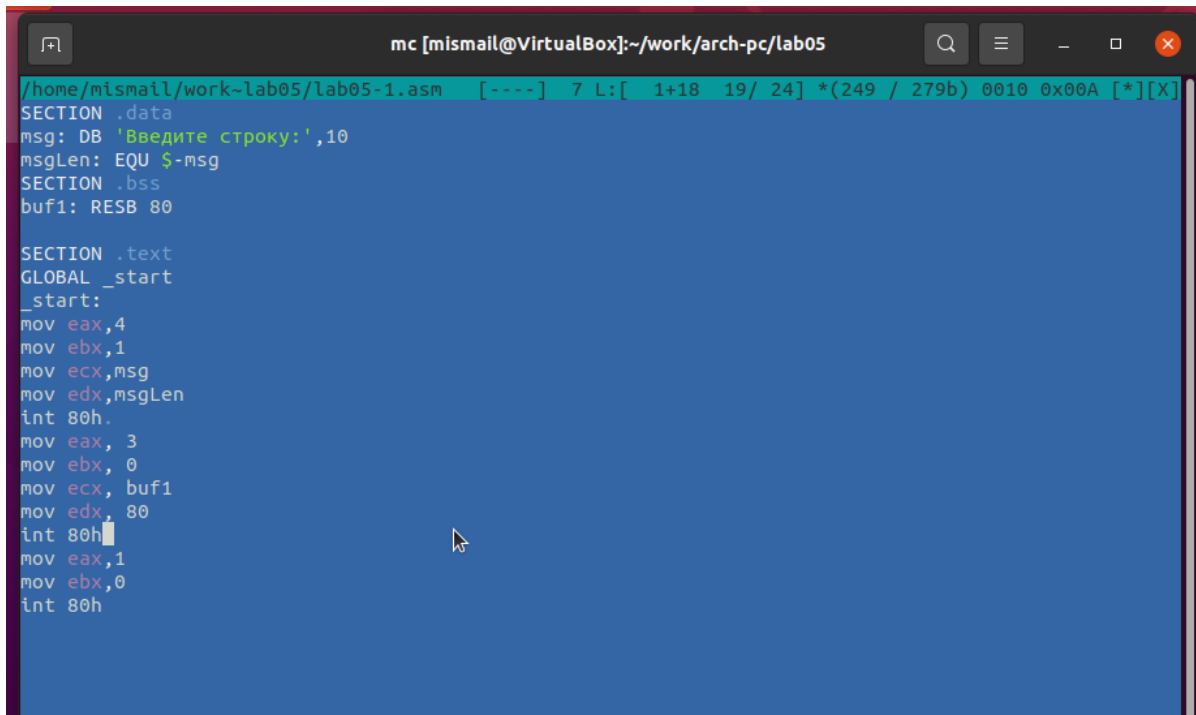


Рис. 3.2: Создание каталога

С помощью команды touch создаю файл lab05-1.asm (см. рис. 3.3).

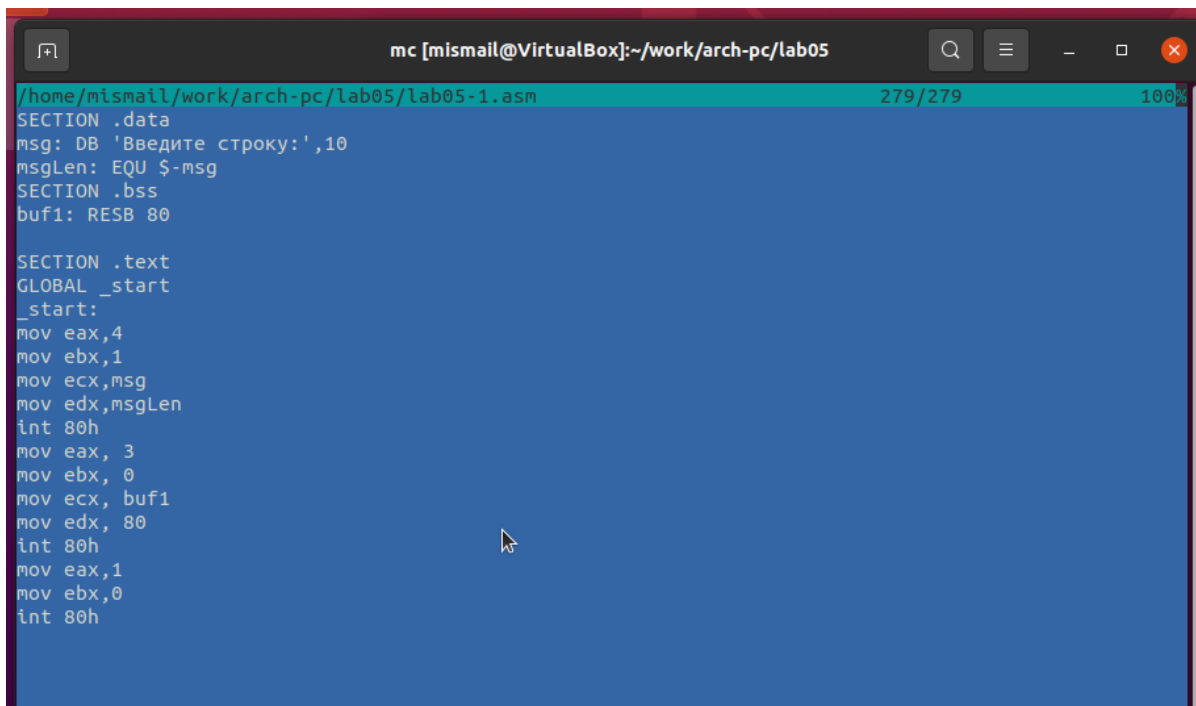


```
mc [mismail@VirtualBox]:~/work/arch-pc/lab05
/home/mismail/work-lab05/lab05-1.asm [----] 7 L: [ 1+18 19/ 24] *(249 / 279b) 0010 0x00A [*][X]
SECTION .data
msg: DB 'Введите строку:',10
msgLen: EQU $-msg
SECTION .bss
buf1: RESB 80

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. 3.5: Программа lab05-1.asm

Для проверки содержимого файла открываю его на просмотр, нажав F3, и убеждаюсь, что код написан верно (см. рис. 3.6).

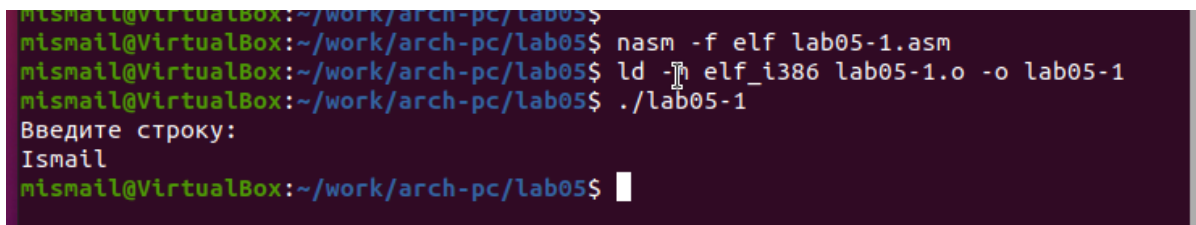


```
mc [mismail@VirtualBox]:~/work/arch-pc/lab05
/home/mismail/work/arch-pc/lab05/lab05-1.asm 279/279 100%
SECTION .data
msg: DB 'Введите строку:',10
msgLen: EQU $-msg
SECTION .bss
buf1: RESB 80

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. 3.6: Просмотр файла lab05-1.asm

Транслирую файл программы в объектный файл, а затем выполняю компоновку, в результате чего получаю исполняемый файл программы (см. рис. 3.7).



```
mismail@VirtualBox:~/work/arch-pc/lab05$
mismail@VirtualBox:~/work/arch-pc/lab05$ nasm -f elf lab05-1.asm
mismail@VirtualBox:~/work/arch-pc/lab05$ ld -T elf_i386 lab05-1.o -o lab05-1
mismail@VirtualBox:~/work/arch-pc/lab05$ ./lab05-1
Введите строку:
Ismail
mismail@VirtualBox:~/work/arch-pc/lab05$
```

Рис. 3.7: Запуск программы lab05-1.asm

3.2 Подключение внешнего файла in_out.asm

Скачиваю файл in_out.asm и размещаю его в рабочем каталоге (см. рис. 3.8). Для копирования файла использую клавишу F5, а для перемещения — клавишу F6.

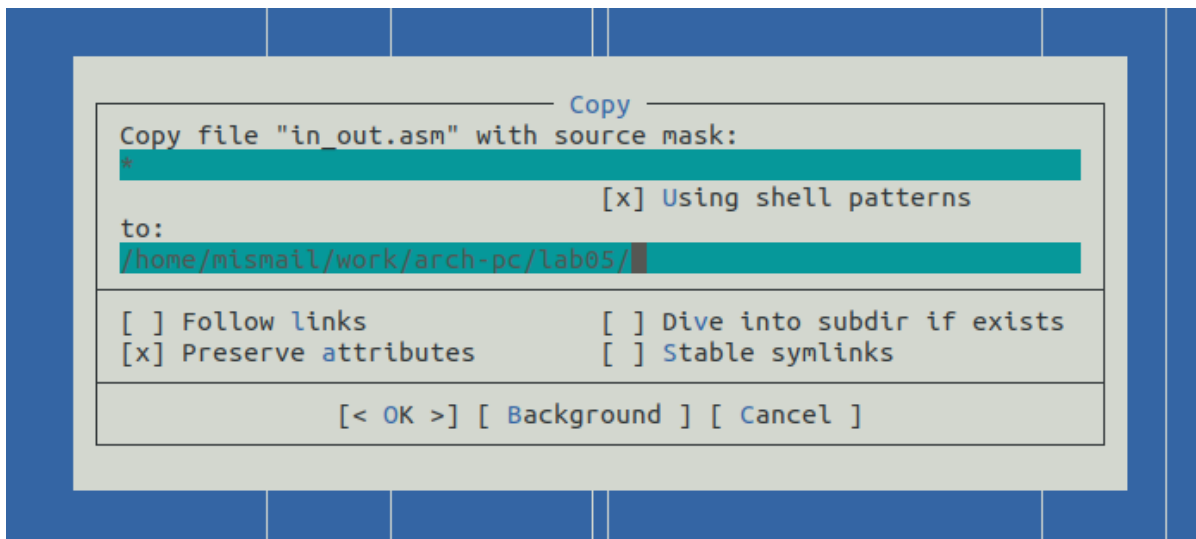


Рис. 3.8: Копирование файла in_out.asm

Копирую файл lab05-1.asm, создавая его копию под именем lab05-2.asm (см. рис. 3.9).

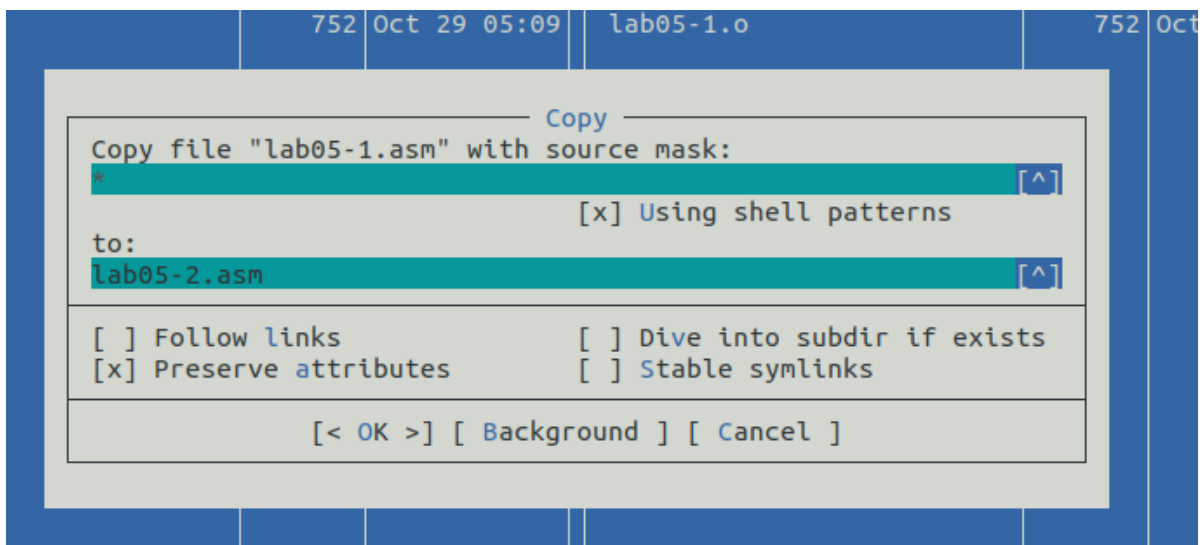
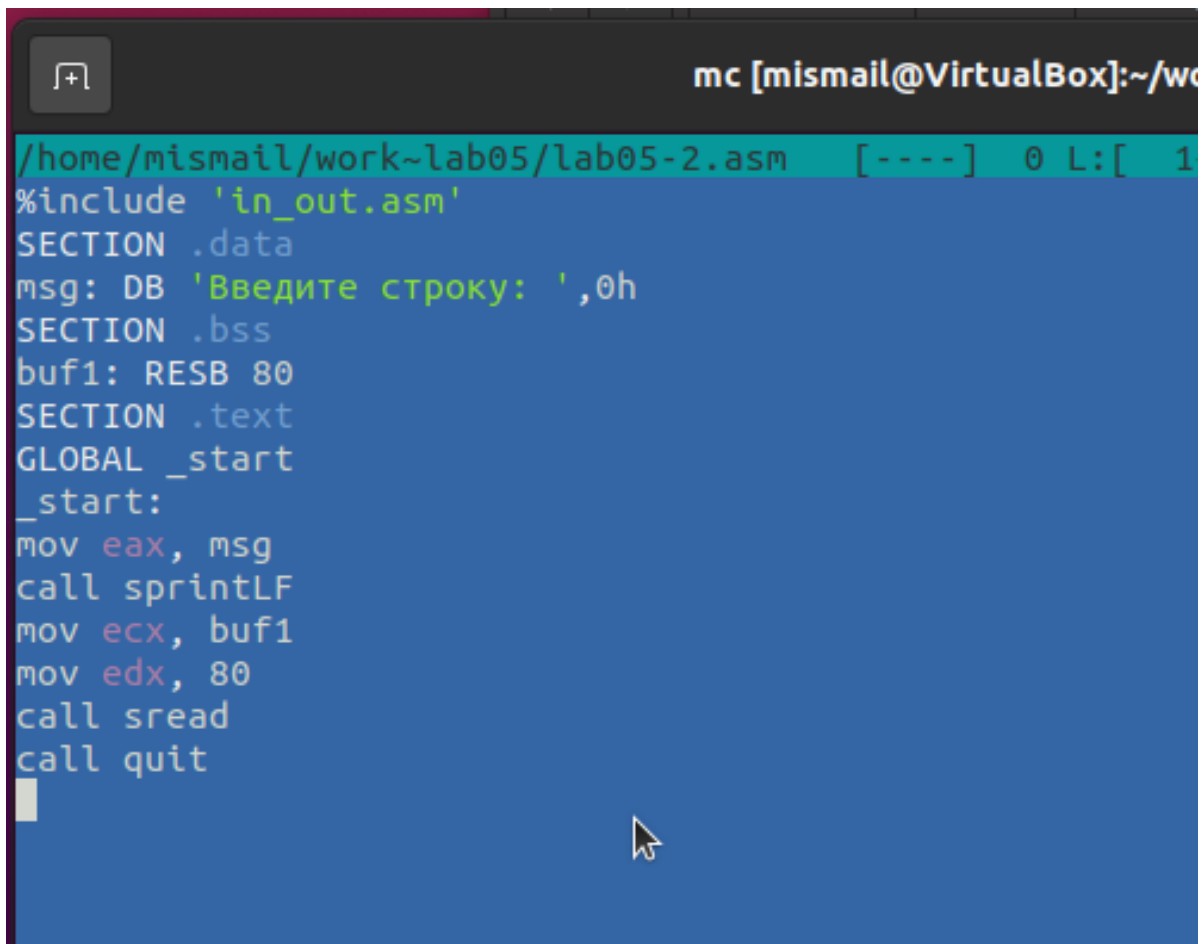


Рис. 3.9: Копирование файла lab05-1.asm

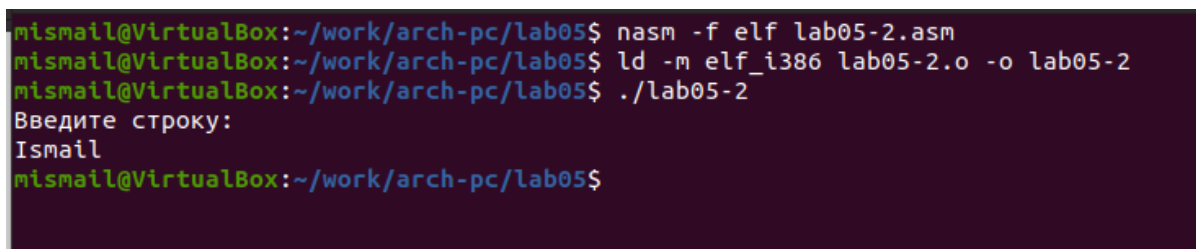
Пишу код для программы lab05-2.asm, используя подпрограммы из внешнего файла in_out.asm (см. рис. 3.10).



```
mc [mismail@VirtualBox]:~/wo
/home/mismail/work~lab05/lab05-2.asm [----] 0 L:[ 1
#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 sprintf
mov ecx, buf1
mov edx, 80
call sread
call quit
```

Рис. 3.10: Программа lab05-2.asm

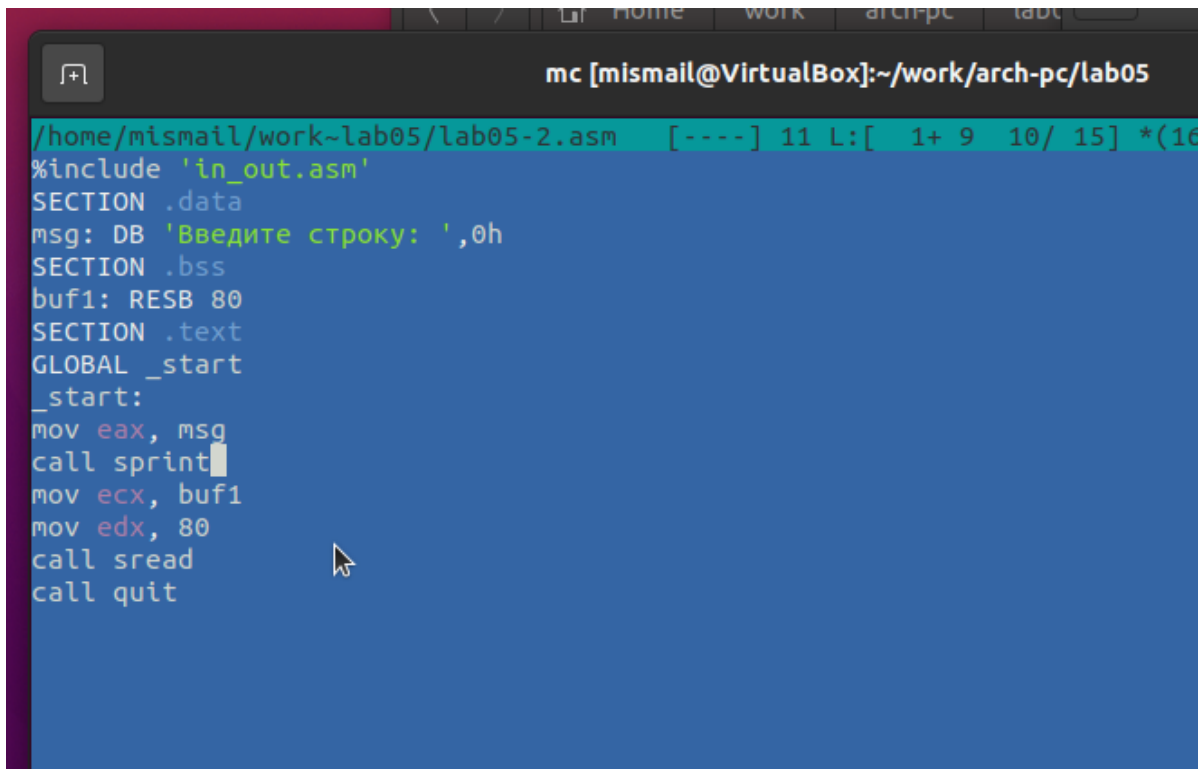
Компилирую программу и проверяю её запуск (см. рис. 3.11).



```
mismail@VirtualBox:~/work/arch-pc/lab05$ nasm -f elf lab05-2.asm
mismail@VirtualBox:~/work/arch-pc/lab05$ ld -m elf_i386 lab05-2.o -o lab05-2
mismail@VirtualBox:~/work/arch-pc/lab05$ ./lab05-2
Введите строку:
Ismail
mismail@VirtualBox:~/work/arch-pc/lab05$
```

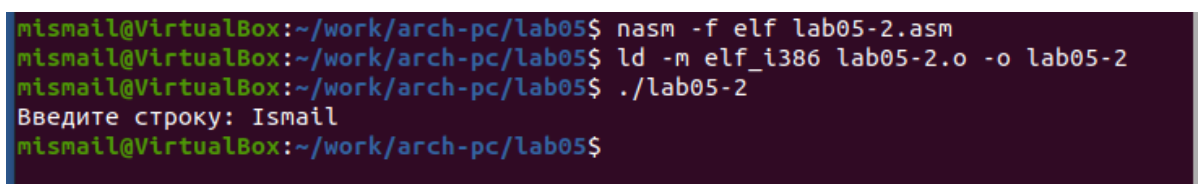
Рис. 3.11: Запуск программы lab05-2.asm

В файле lab05-2.asm заменяю подпрограмму sprintf на sprintf. После этого заново собираю исполняемый файл (см. рис. 3.12 и 3.13).



```
mc [mismail@VirtualBox]:~/work/arch-pc/lab05
/home/mismail/work~lab05/lab05-2.asm [----] 11 L: [ 1+ 9 10/ 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
call quit
```

Рис. 3.12: Программа в файле lab05-2.asm



```
mismail@VirtualBox:~/work/arch-pc/lab05$ nasm -f elf lab05-2.asm
mismail@VirtualBox:~/work/arch-pc/lab05$ ld -m elf_i386 lab05-2.o -o lab05-2
mismail@VirtualBox:~/work/arch-pc/lab05$ ./lab05-2
Введите строку: Ismail
mismail@VirtualBox:~/work/arch-pc/lab05$
```

Рис. 3.13: Запуск программы lab05-2.asm

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

3.3 Задание для самостоятельной работы

Копирую программу lab05-1.asm и модифицирую код, чтобы она работала по следующему алгоритму (см. рис. 3.14 и 3.15): - выводит приглашение “Введите строку:”; - принимает строку с клавиатуры; - отображает введенную строку на экране.

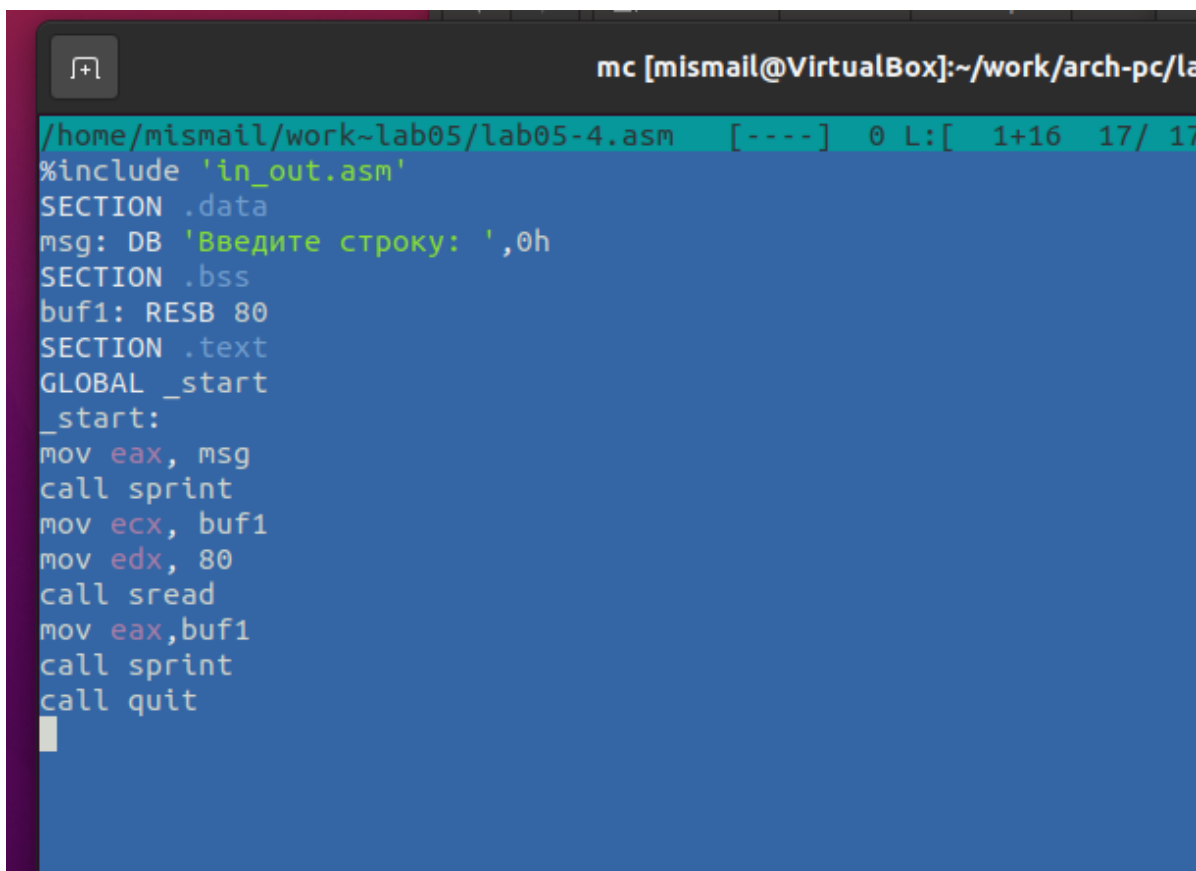
```
mc [mismail@VirtualBox]:~/work/  
/home/mismail/work~lab05/lab05-3.asm [----] 12 L:[ 1+21  
SECTION .data  
msg: DB 'Введите строку:',10  
msgLen: EQU $-msg  
SECTION .bss  
buf1: RESB 80  
  
SECTION .text  
GLOBAL _start  
_start:  
mov eax,4  
mov ebx,1  
mov ecx,msg  
mov edx,msgLen  
int 80h.  
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 lab05-3.asm

```
mismail@VirtualBox:~/work/arch-pc/lab05$  
mismail@VirtualBox:~/work/arch-pc/lab05$ nasm -f elf lab05-3.asm  
mismail@VirtualBox:~/work/arch-pc/lab05$ ld -m elf_i386 lab05-3.o -o lab05-3  
mismail@VirtualBox:~/work/arch-pc/lab05$ ./lab05-3  
Введите строку:  
Ismail  
Ismail  
mismail@VirtualBox:~/work/arch-pc/lab05$
```

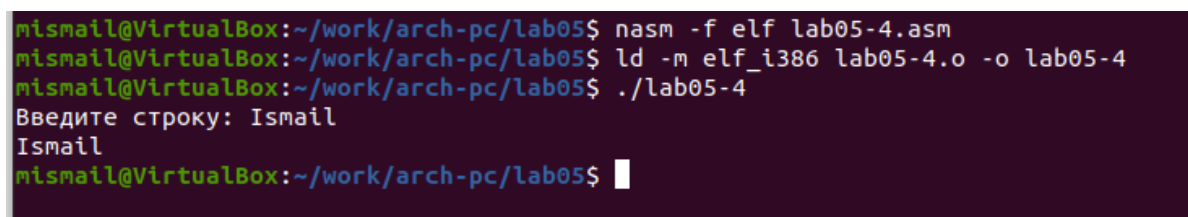
Рис. 3.15: Запуск программы lab05-3.asm

Аналогично, копирую программу lab05-2.asm и изменяю код, теперь используя подпрограммы из файла in_out.asm (см. рис. 3.16 и 3.17).



```
mc [mismail@VirtualBox]:~/work/arch-pc/lab05-4.asm
/home/mismail/work~lab05/lab05-4.asm [---] 0 L:[ 1+16 17/ 17
#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. 3.16: Программа lab05-4.asm



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

Рис. 3.17: Запуск программы lab05-4.asm

4 Выводы

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