

Отчёт по лабораторной работе 5

Архитектура компьютеров

Агджабекова Эся Рустамовна НПИбд-01-24

Содержание

1	Цель работы	5
2	Выполнение лабораторной работы	6
3	Выводы	17

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	10
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	13
2.11	Программа в файле lab05-3.asm	14
2.12	Запуск программы lab05-3.asm	14
2.13	Программа в файле lab05-4.asm	15
2.14	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 Выполнение лабораторной работы

Я открыла Midnight Commander.
Перешла в каталог ~/work/arch-pc.
Создала каталог lab05 (рис. 2.1).

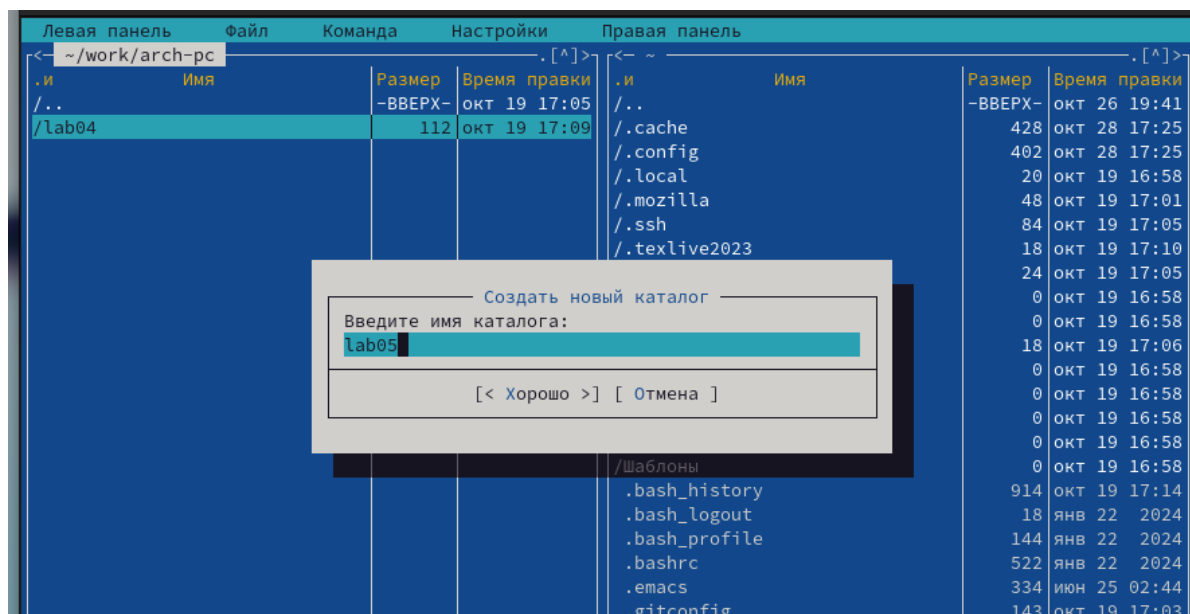


Рис. 2.1: Создание каталога

Создала файл lab05-1.asm (рис. 2.2).

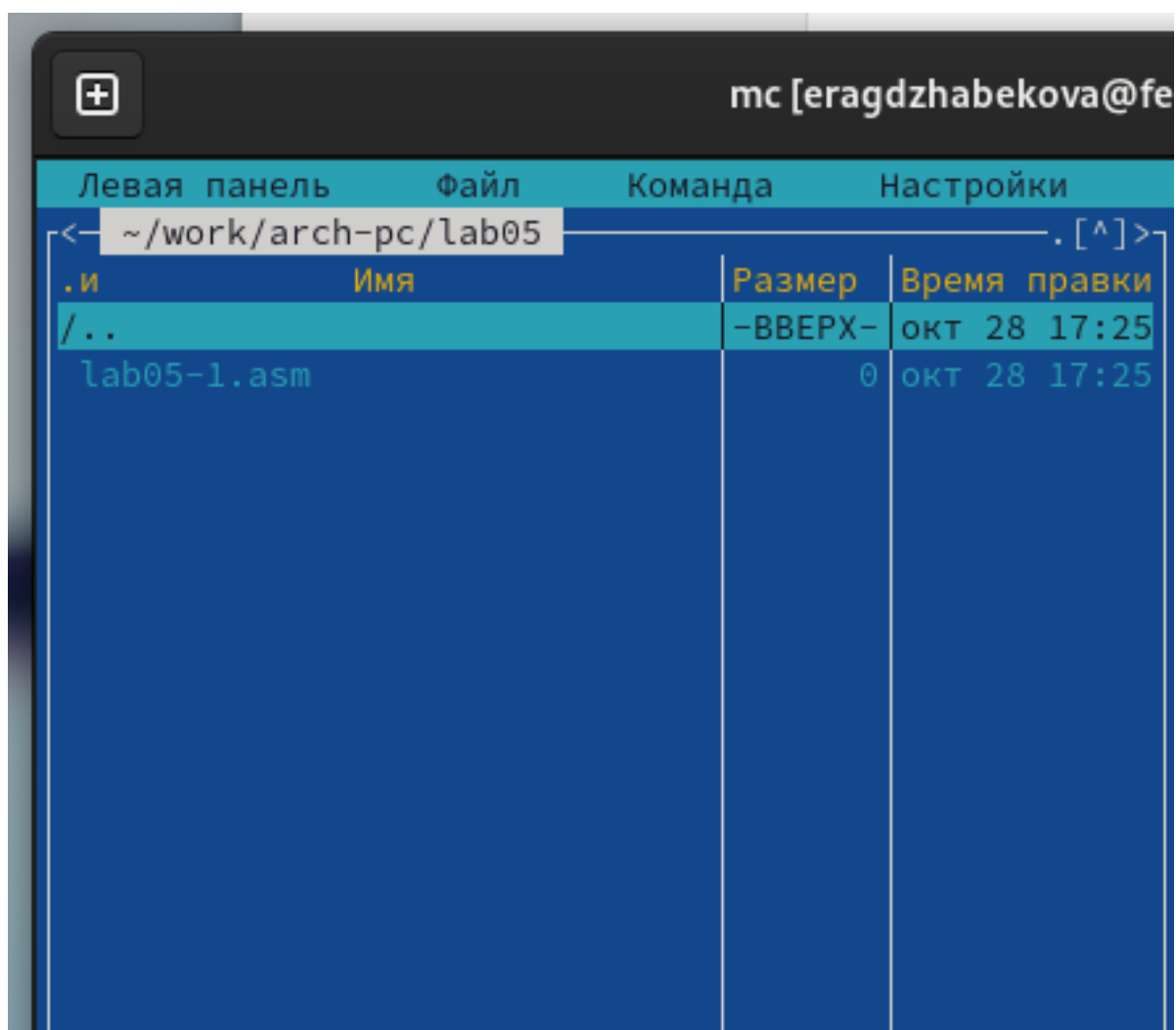
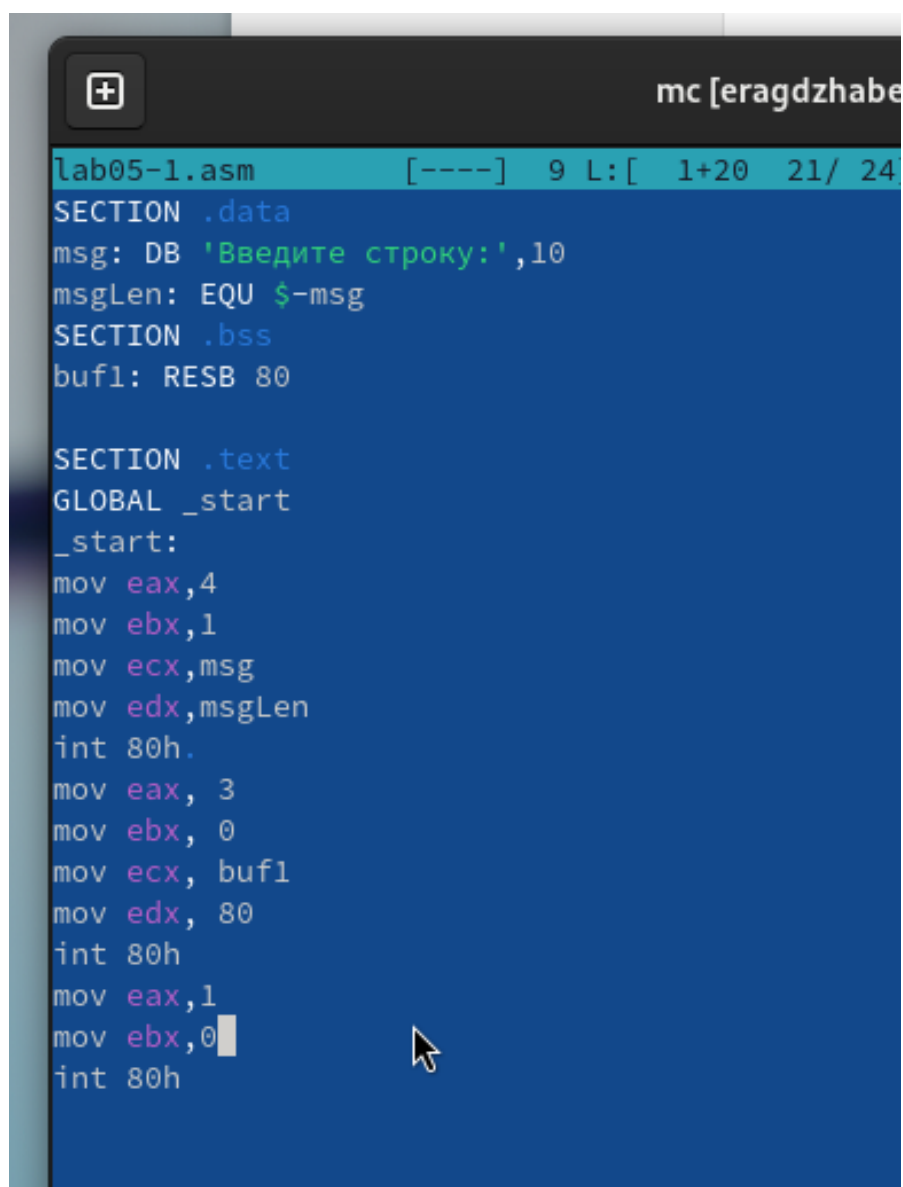


Рис. 2.2: Создание файла lab05-1.asm

Открыла файл на редактирование и написала код (рис. 2.3).

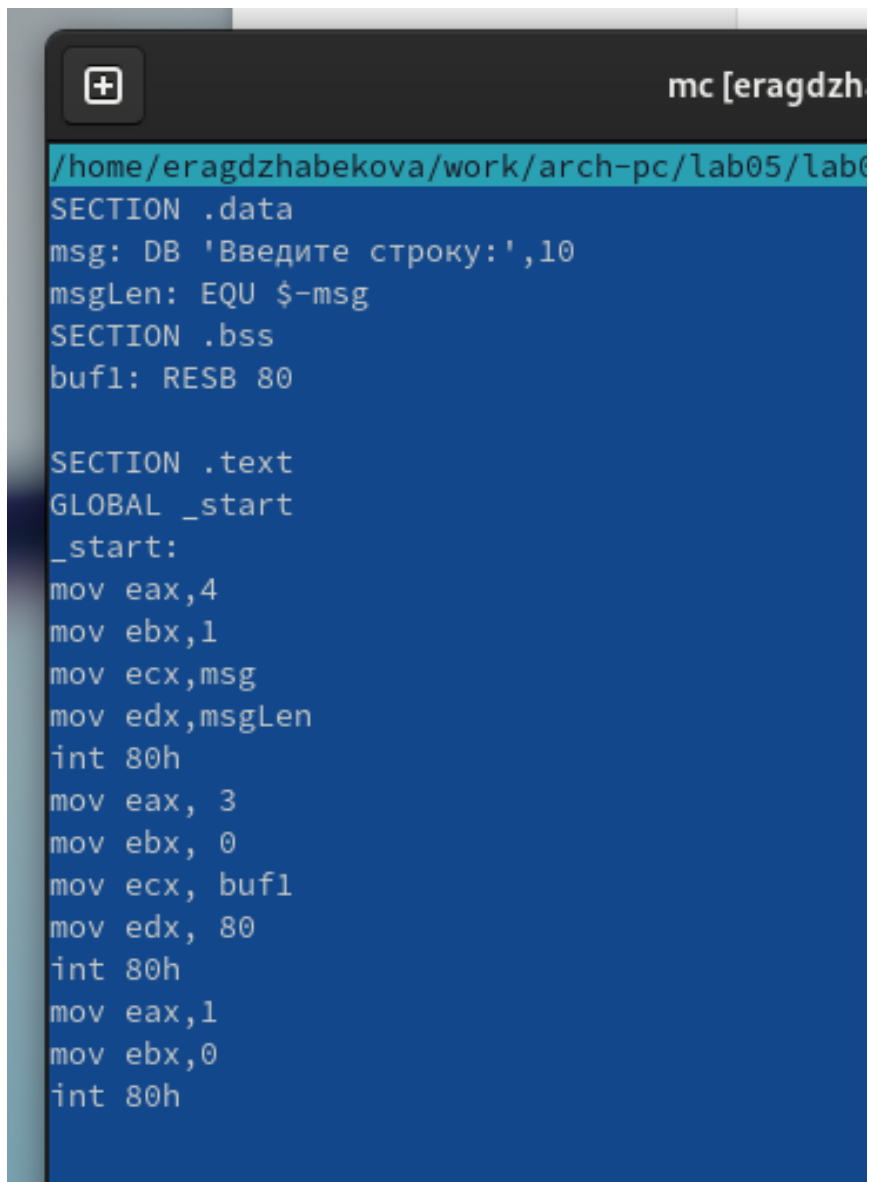


```
lab05-1.asm [----] 9 L: [ 1+20 21/ 24]
SECTION .data
msg: DB 'Введите строку:',10
msgLen: EQU $-msg
SECTION .bss
buf1: RESB 80

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

Открыла файл для просмотра и убедилась, что он содержит написанный код (рис. 2.4).



```
mc [eragdzh...
/home/eragdzhbekova/work/arch-pc/lab05/lab0
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

Получила исполняемый файл программы и проверила его работу (рис. 2.5).

```
eragdzhbekova@fedora:~/work/arch-pc/lab05$ nasm -f elf lab05-1.asm
eragdzhbekova@fedora:~/work/arch-pc/lab05$ ld -m elf_i386 lab05-1.o -o lab05-1
eragdzhbekova@fedora:~/work/arch-pc/lab05$ ./lab05-1
Введите строку:
Fedora
eragdzhbekova@fedora:~/work/arch-pc/lab05$
```

Рис. 2.5: Запуск программы lab05-1.asm

Скачала файл in_out.asm. Добавила его в рабочий каталог.
Скопировала lab05-1.asm в lab05-2.asm (рис. 2.6).

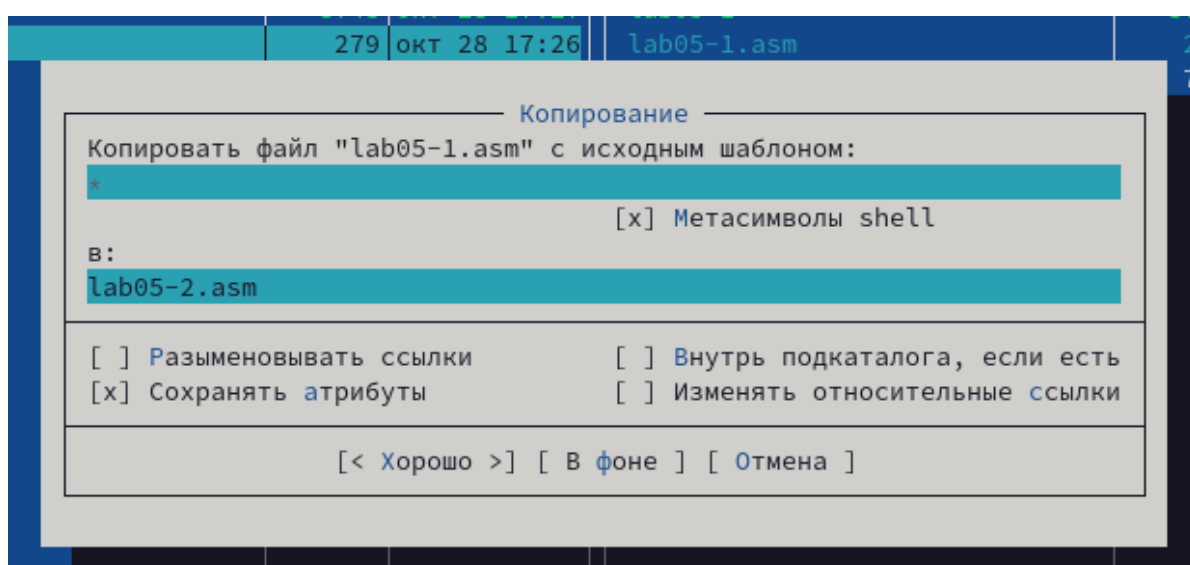
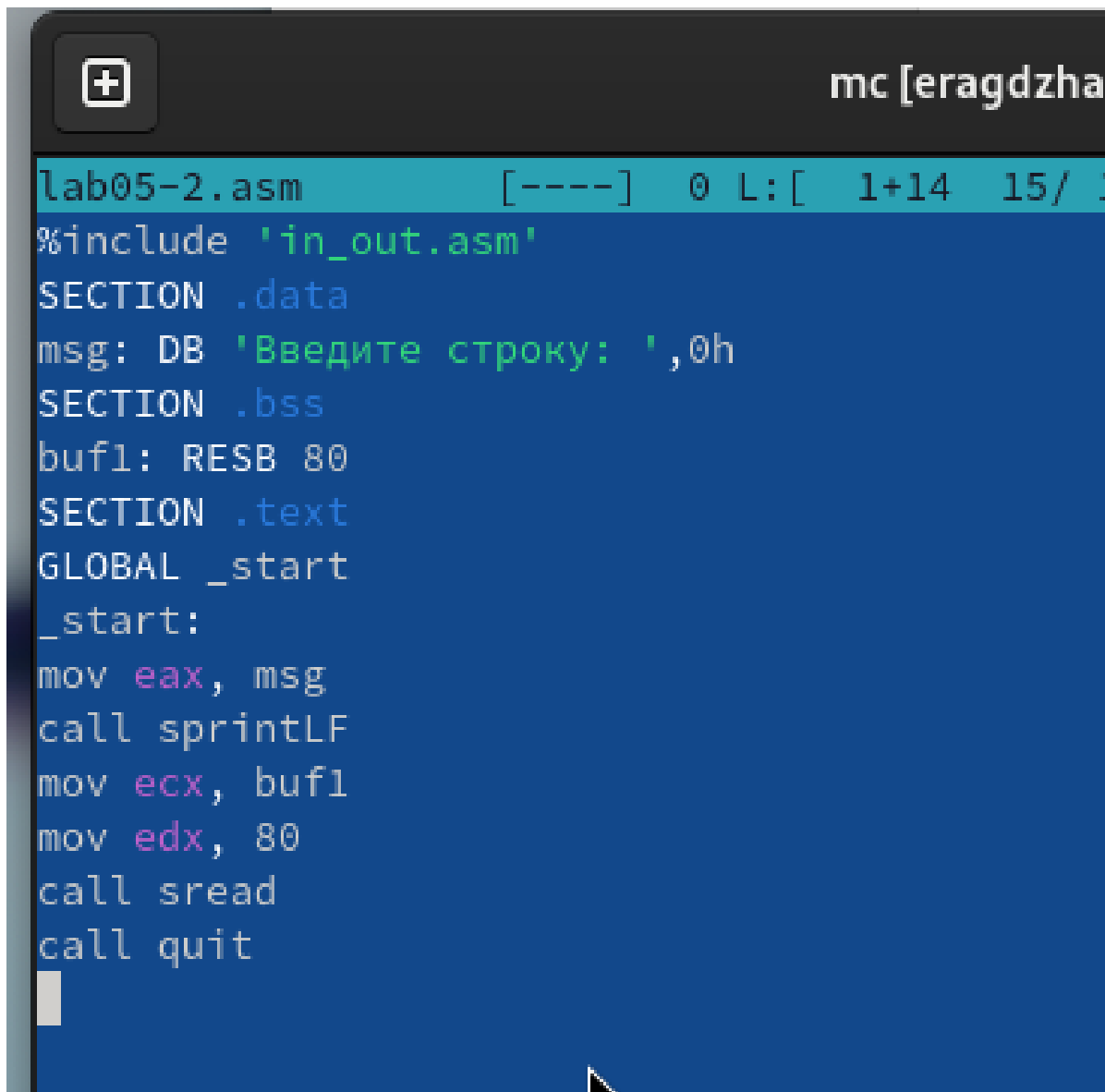


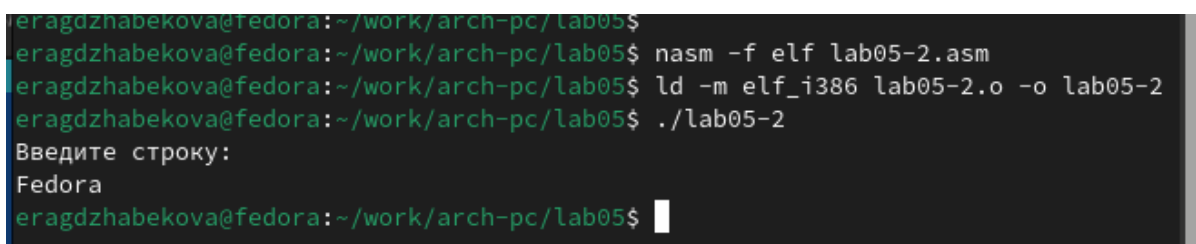
Рис. 2.6: Копирование файла

Написала код программы lab05-2.asm (рис. 2.7).
Скомпилировала программу и проверила запуск (рис. 2.8).



```
lab05-2.asm [----] 0 L: [ 1+14 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
mov ecx, buf1
mov edx, 80
call sread
call quit
```

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



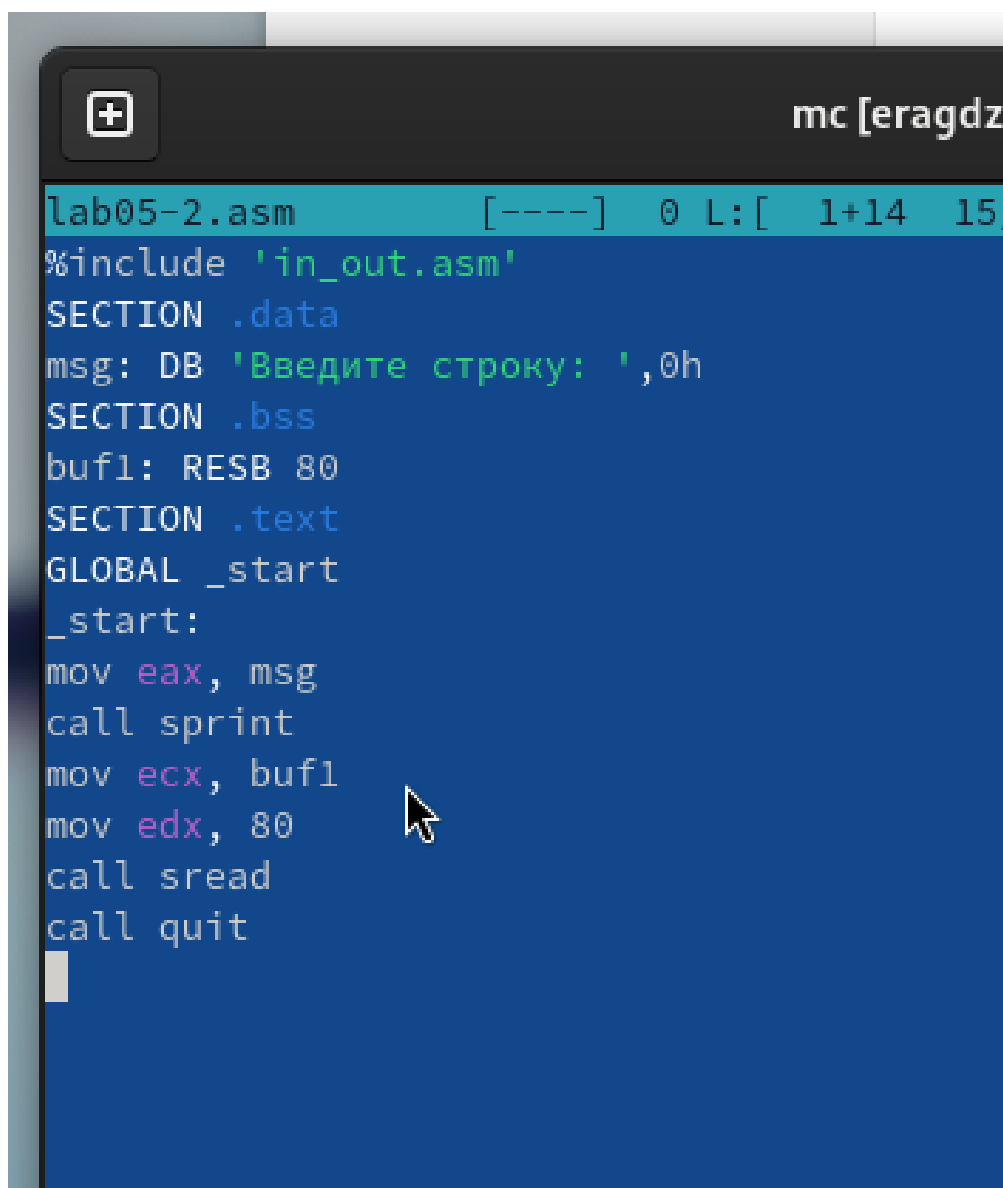
```
eragdzhabekova@fedora:~/work/arch-pc/lab05$
eragdzhabekova@fedora:~/work/arch-pc/lab05$ nasm -f elf lab05-2.asm
eragdzhabekova@fedora:~/work/arch-pc/lab05$ ld -m elf_i386 lab05-2.o -o lab05-2
eragdzhabekova@fedora:~/work/arch-pc/lab05$ ./lab05-2
Введите строку:
Fedora
eragdzhabekova@fedora:~/work/arch-pc/lab05$
```

Рис. 2.8: Запуск программы lab05-2.asm

В файле lab05-2.asm я заменила подпрограмму sprintLF на sprint (рис. 2.9).

Затем я снова собрала исполняемый файл (рис. 2.10).

Теперь после вывода строки она не завершается символом перехода на новую строку.



```
lab05-2.asm [----] 0 L: [ 1+14 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 sprint
mov ecx, buf1
mov edx, 80
call sread
call quit
```

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

```
eragdzhbekova@fedora:~/work/arch-pc/lab05$  
eragdzhbekova@fedora:~/work/arch-pc/lab05$ nasm -f elf lab05-2.asm  
eragdzhbekova@fedora:~/work/arch-pc/lab05$ ld -m elf_i386 lab05-2.o -o lab05-2  
eragdzhbekova@fedora:~/work/arch-pc/lab05$ ./lab05-2  
Введите строку: Fedora  
eragdzhbekova@fedora:~/work/arch-pc/lab05$
```

Рис. 2.10: Запуск программы lab05-2.asm

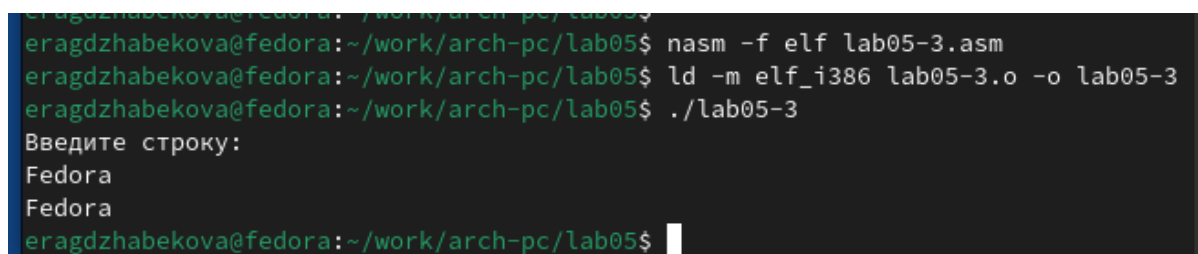
Скопировала программу lab05-1.asm и изменила код, чтобы программа выводила приглашение типа “Введите строку:”, затем считывала строку с клавиатуры и выводила введенную строку на экран (рис. 2.11, рис. 2.12).



```
lab05-3.asm [----] 9 L: [ 1+25
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
```

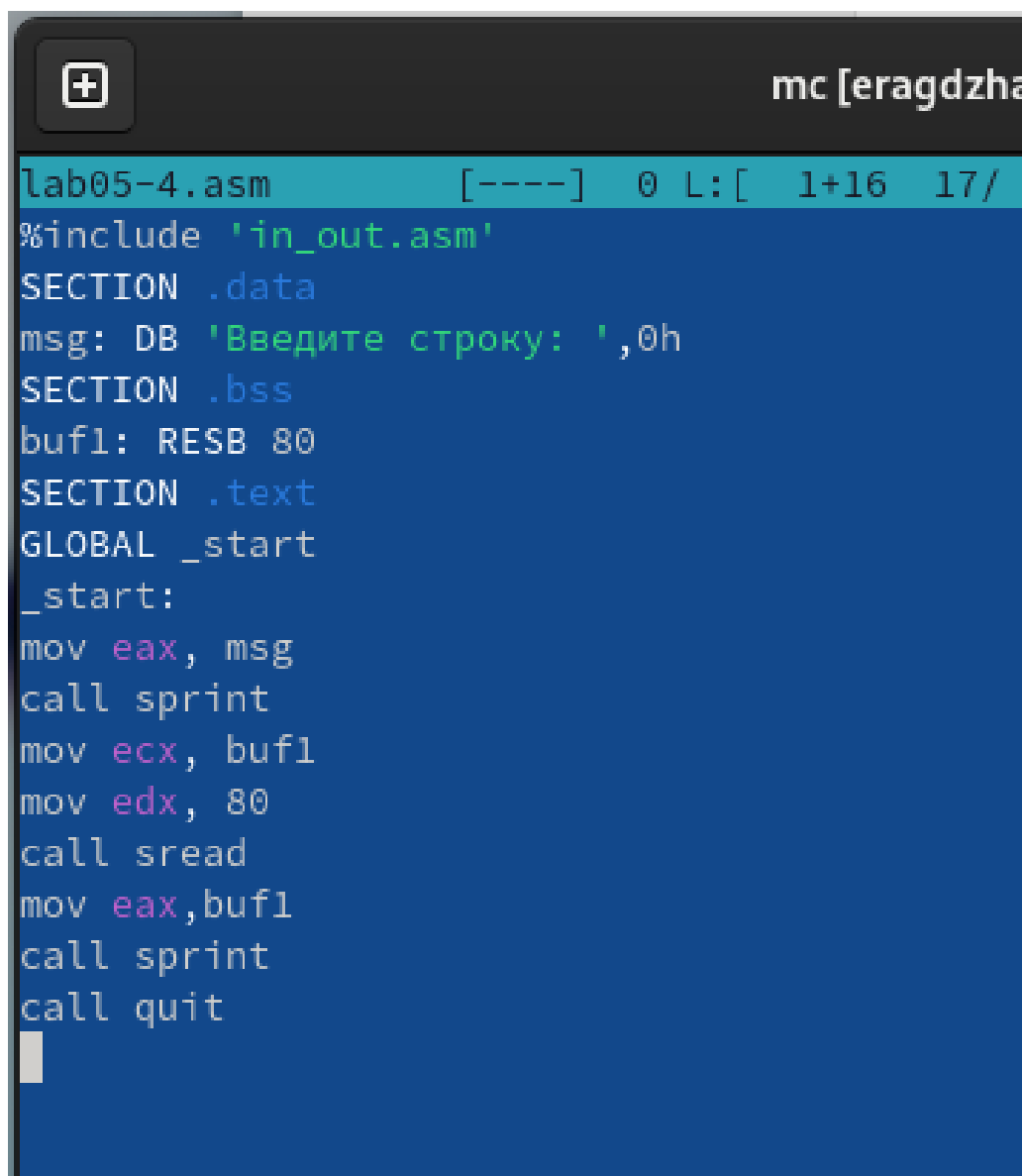
Рис. 2.11: Программа в файле lab05-3.asm



```
eragdzhbekova@fedora: ~/work/arch-pc/lab05$ nasm -f elf lab05-3.asm
eragdzhbekova@fedora: ~/work/arch-pc/lab05$ ld -m elf_i386 lab05-3.o -o lab05-3
eragdzhbekova@fedora: ~/work/arch-pc/lab05$ ./lab05-3
Введите строку:
Fedora
Fedora
eragdzhbekova@fedora: ~/work/arch-pc/lab05$
```

Рис. 2.12: Запуск программы lab05-3.asm

Также я скопировала программу lab05-2.asm и внесла соответствующие изменения в код, чтобы программа выводила приглашение типа “Введите строку:”, затем считывала строку с клавиатуры и выводила введенную строку на экран (рис. 2.13, рис. 2.14).



```
lab05-4.asm [-----] 0 L:[ 1+16 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. 2.13: Программа в файле lab05-4.asm

```
eragdzhbekova@fedora:~/work/arch-pc/lab05$ nasm -f elf lab05-3.asm
eragdzhbekova@fedora:~/work/arch-pc/lab05$ ld -m elf_i386 lab05-3.o -o lab05-3
eragdzhbekova@fedora:~/work/arch-pc/lab05$ ./lab05-3
Введите строку:
Fedora
Fedora
eragdzhbekova@fedora:~/work/arch-pc/lab05$
```

Рис. 2.14: Запуск программы lab05-4.asm

Отличие этих двух реализаций заключается 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 инструкции `call`.

3 Выводы

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