

Отчёт по лабораторной работе 5

Архитектура компьютера

Тумуреева Галина Аркадьевна

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	15
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

1. Я запустила Midnight Commander.
2. Перешла в папку ~/work/arch-pc.
3. Создала папку lab05.

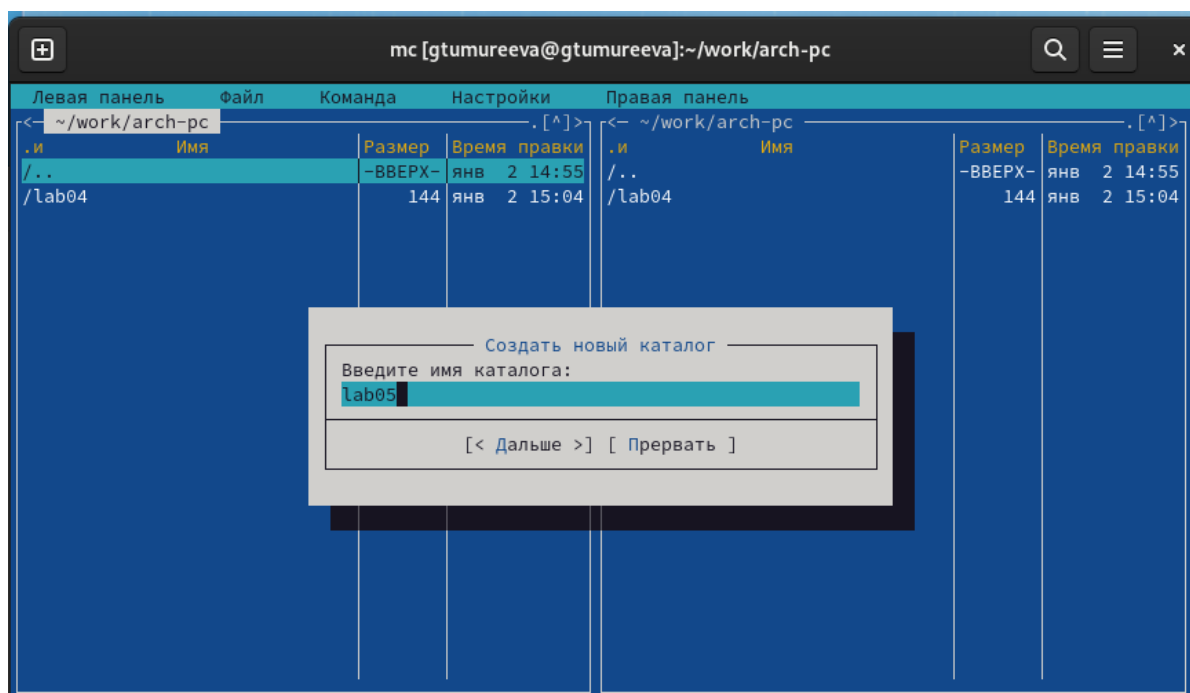


Рис. 2.1: Создание каталога

4. Сделала файл lab05-1.asm.

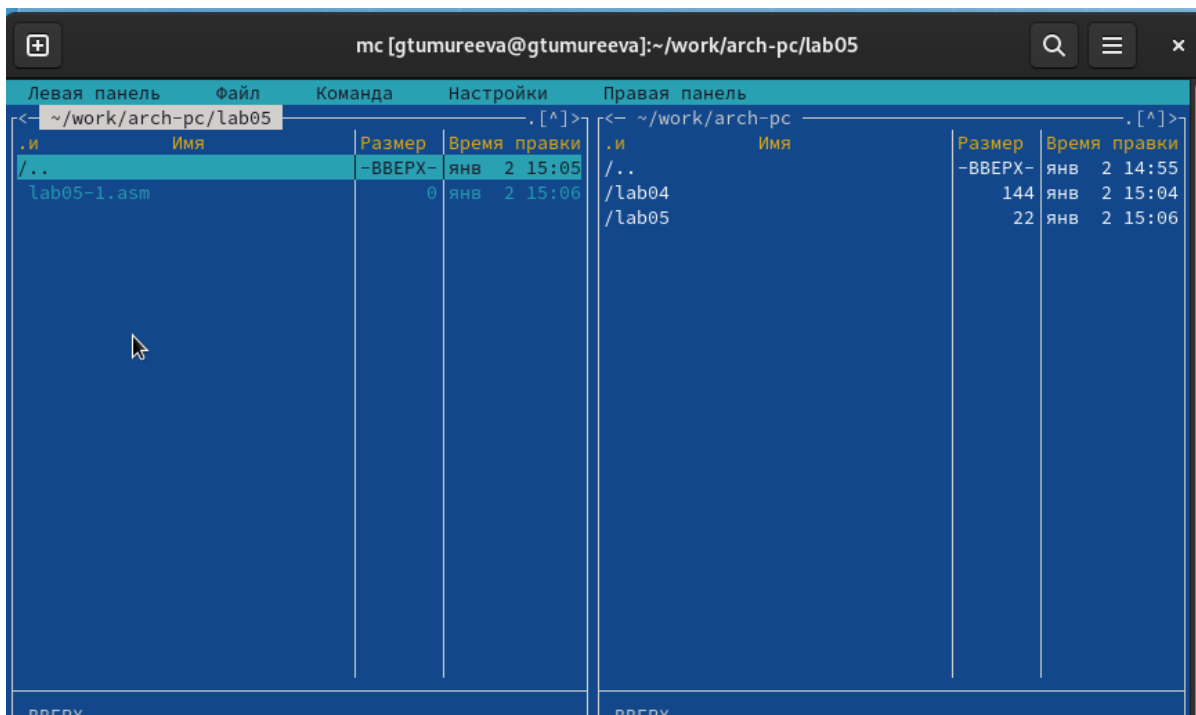
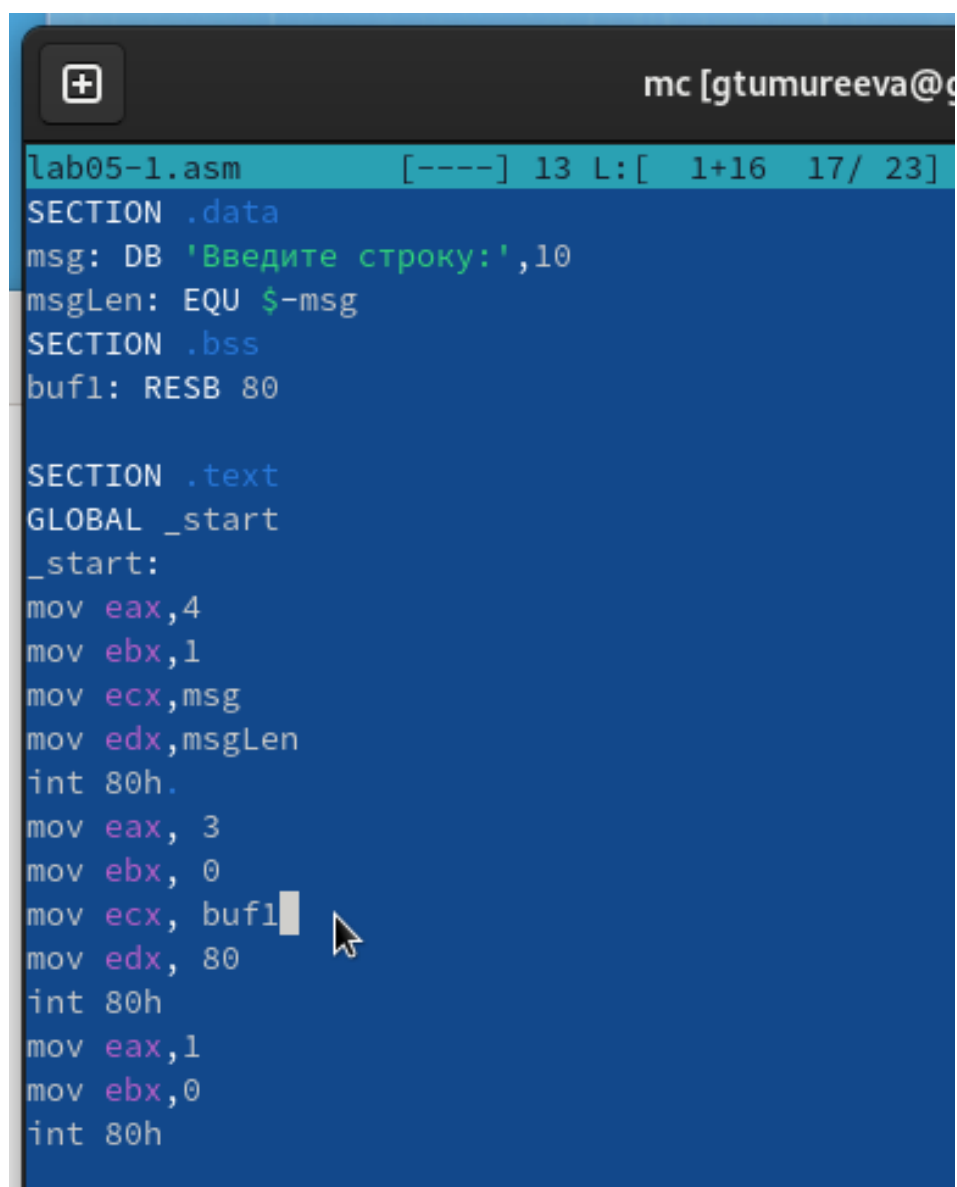


Рис. 2.2: Создание файла lab05-1.asm

5. Открыла этот файл для редактирования.
6. Написала код программы по заданию.

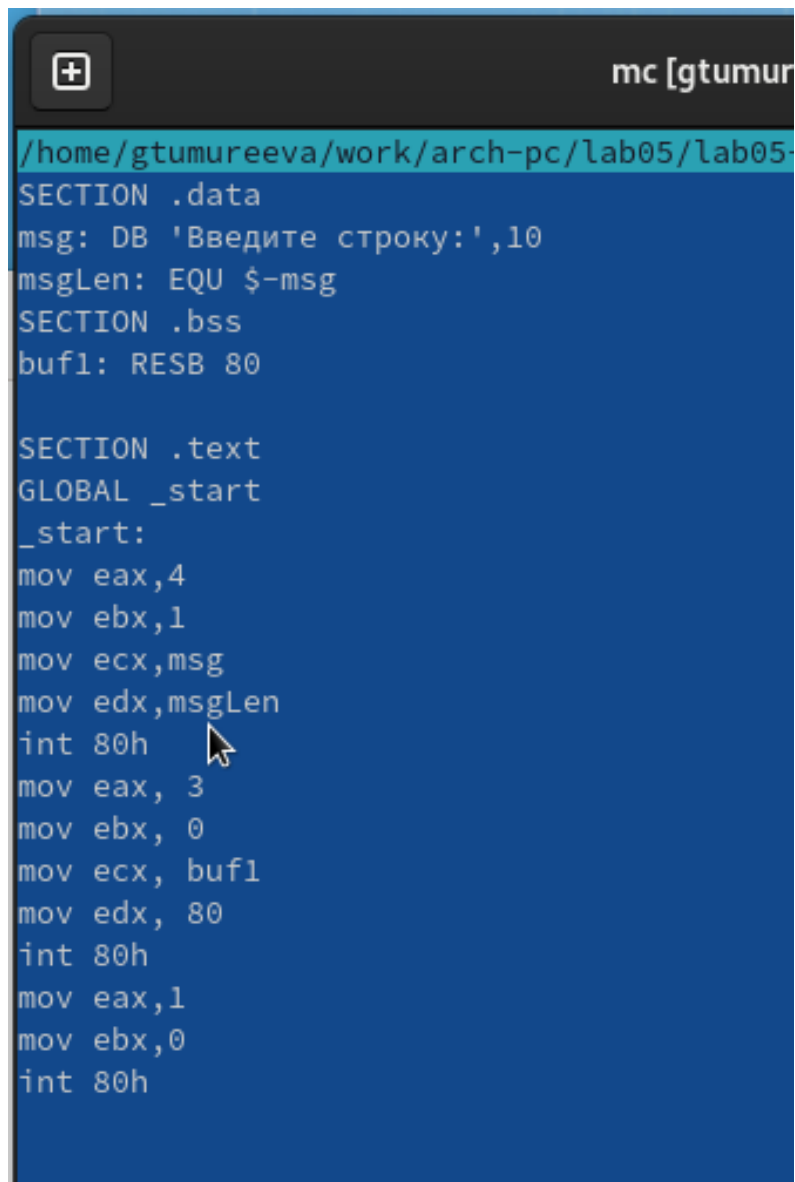


```
lab05-1.asm [----] 13 L:[ 1+16 17/ 23]
SECTION .data
msg: DB 'Введите строку:',10
msgLen: EQU $-msg
SECTION .bss
buf1: RESB 80

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. Открыла файл, чтобы проверить, и увидела, что в нем есть мой код.



```
mc [gtumur  
/home/gtumureeva/work/arch-pc/lab05/lab05-  
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. Скомпилировала программу, получила исполняемый файл и проверила его работу.

```
[gtumureeva@gtumureeva lab05]$ nasm -f elf lab05-1.asm
[gtumureeva@gtumureeva lab05]$ ld -m elf_i386 lab05-1.o -o lab05-1
[gtumureeva@gtumureeva lab05]$ ./lab05-1
Введите строку:
Galina
[gtumureeva@gtumureeva lab05]$
```

Рис. 2.5: Запуск программы lab05-1.asm

9. Загрузила файл in_out.asm.
10. Добавила файл in_out.asm в мою рабочую папку.
11. Сделала копию файла lab05-1.asm и назвала её lab05-2.asm.

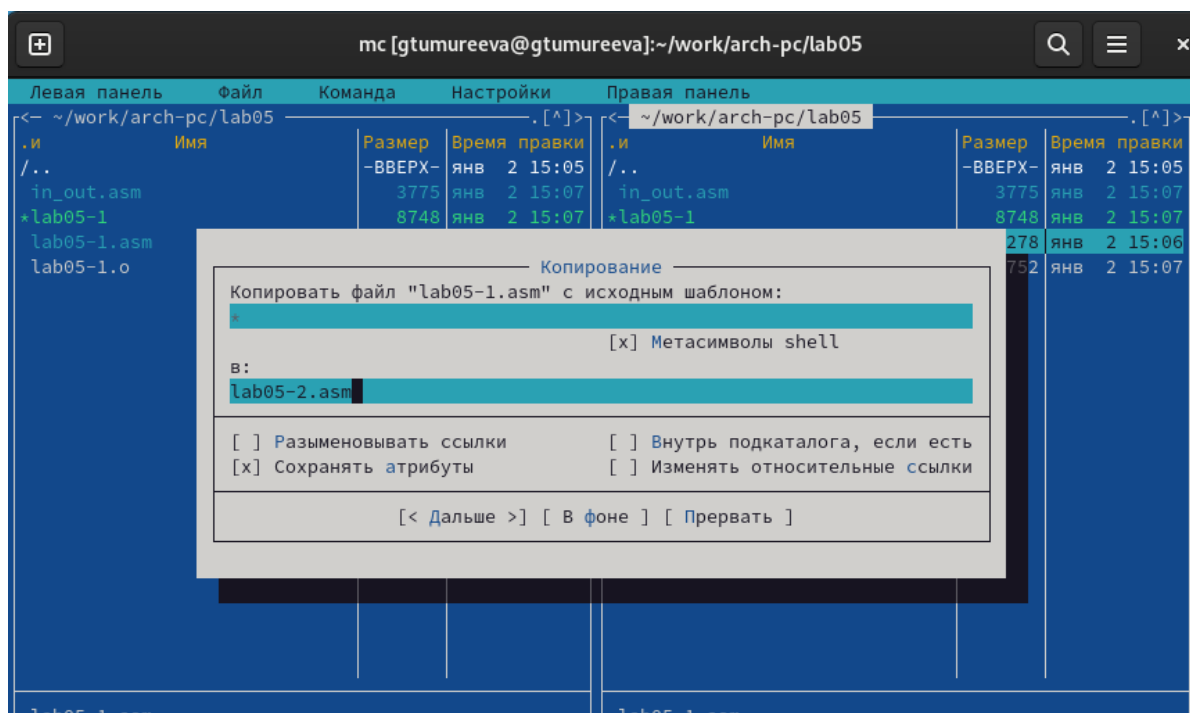
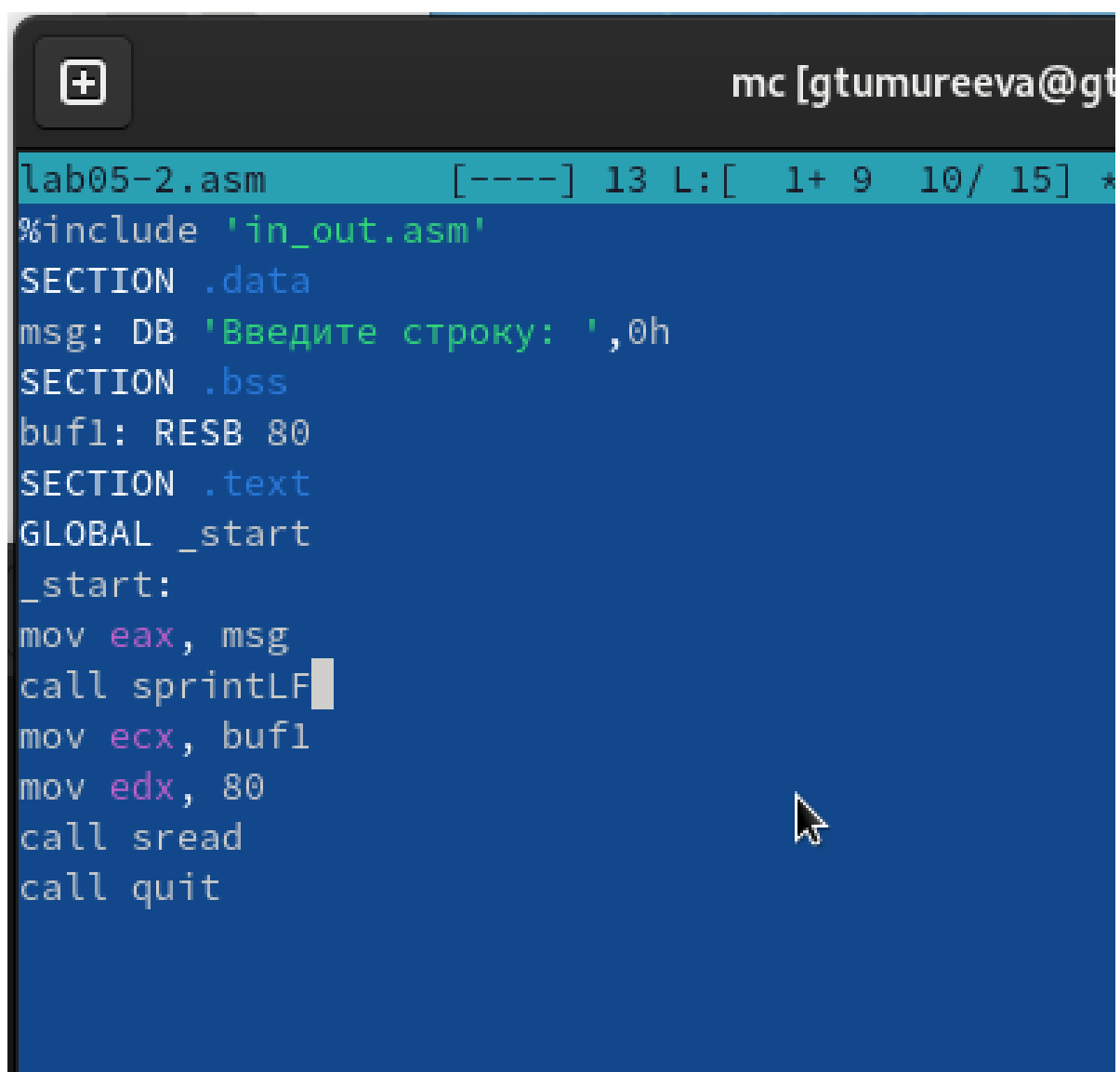


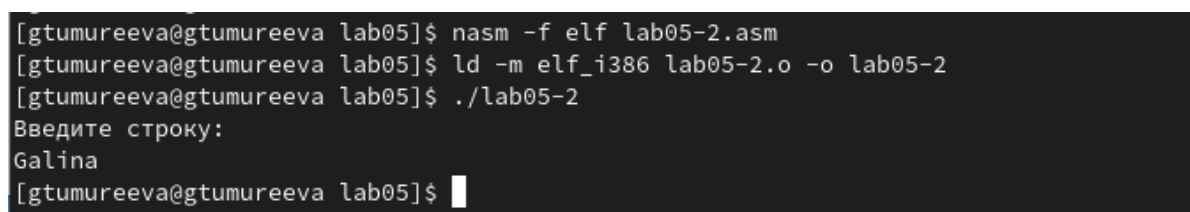
Рис. 2.6: Копирование файла

12. Написала программу в файле lab05-2.asm, скомпилировала его и убедилась, что программа работает.



```
lab05-2.asm [----] 13 L:[ 1+ 9 10/ 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 sprintf
mov ecx, buf1
mov edx, 80
call sread
call quit
```

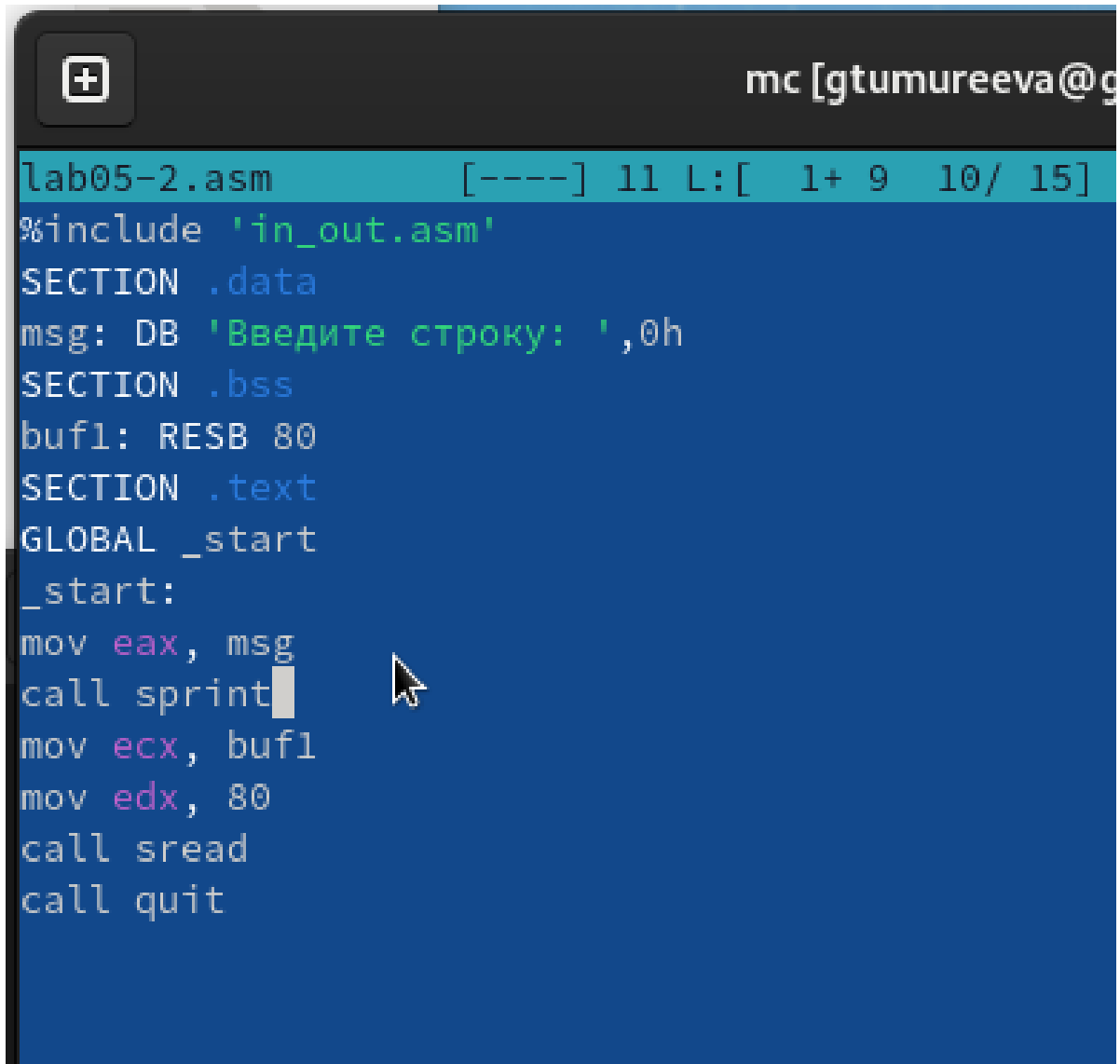
Рис. 2.7: Программа в файле lab05-2.asm



```
[gtumureeva@gtumureeva lab05]$ nasm -f elf lab05-2.asm
[gtumureeva@gtumureeva lab05]$ ld -m elf_i386 lab05-2.o -o lab05-2
[gtumureeva@gtumureeva lab05]$ ./lab05-2
Введите строку:
Galina
[gtumureeva@gtumureeva lab05]$
```

Рис. 2.8: Запуск программы lab05-2.asm

13. В файле lab05-2.asm поменяла подпрограмму `sprintLF` на `sprint`, пересобрала исполняемый файл, и теперь после вывода строки не происходит переход на новую строку.



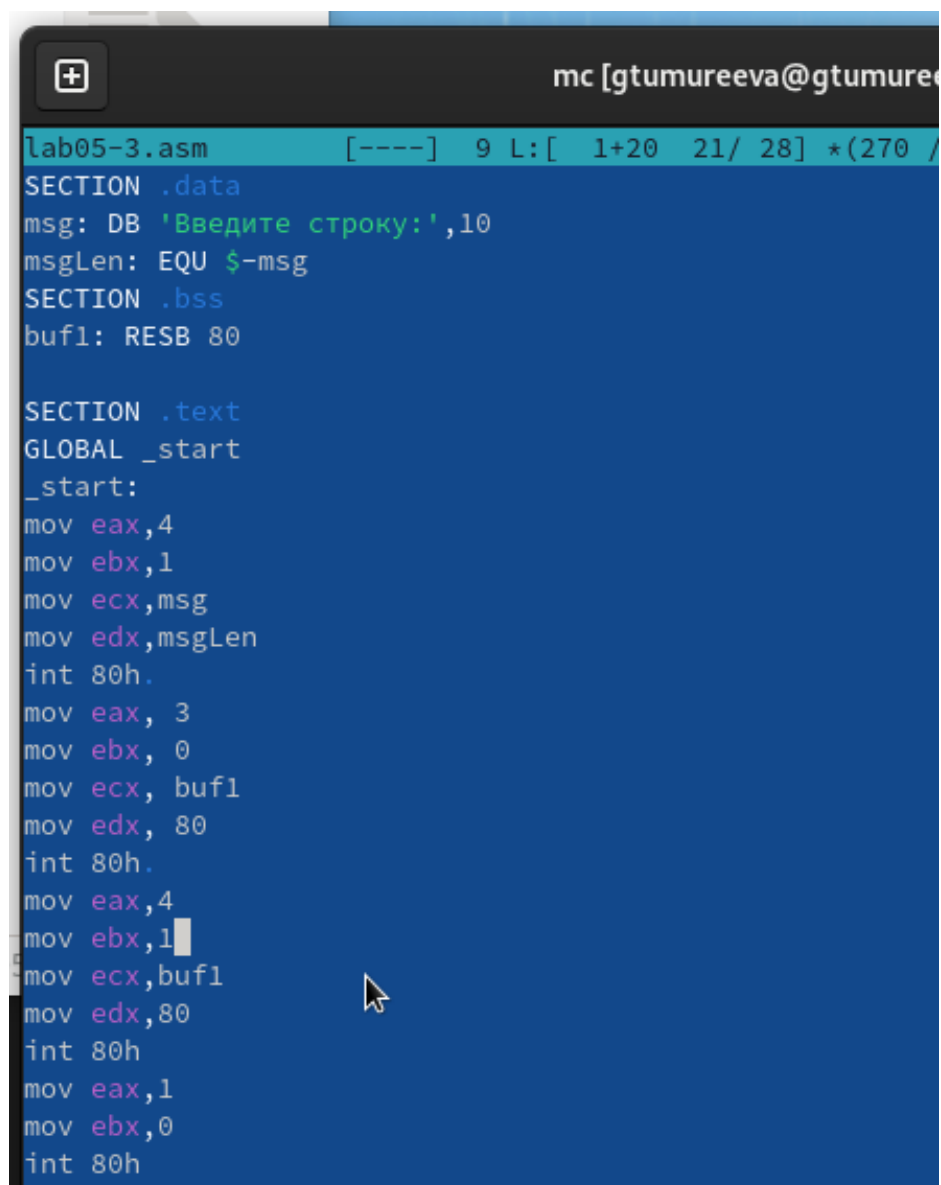
```
lab05-2.asm [----] 11 L:[ 1+ 9 10/ 15]
#include 'in_out.asm'
SECTION .data
msg: DB 'Введите строку: ',0h
SECTION .bss
buf1: RESB 80
SECTION .text
GLOBAL _start
_start:
mov eax, msg
call sprint
mov ecx, buf1
mov edx, 80
call sread
call quit
```

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

```
[gtumureeva@gtumureeva lab05]$ nasm -f elf lab05-2.asm
[gtumureeva@gtumureeva lab05]$ ld -m elf_i386 lab05-2.o -o lab05-2
[gtumureeva@gtumureeva lab05]$ ./lab05-2
Введите строку: Galina
[gtumureeva@gtumureeva lab05]$
```

Рис. 2.10: Запуск программы lab05-2.asm

14. Скопировала файл lab05-1.asm и изменила код так, чтобы программа запрашивала ввести строку с помощью сообщения “Введите строку:”, затем читала строку с клавиатуры и выводила её обратно на экран.



```
mc [gtumureeva@gtumuree
lab05-3.asm [----] 9 L: [ 1+20 21/ 28] *(270 /
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

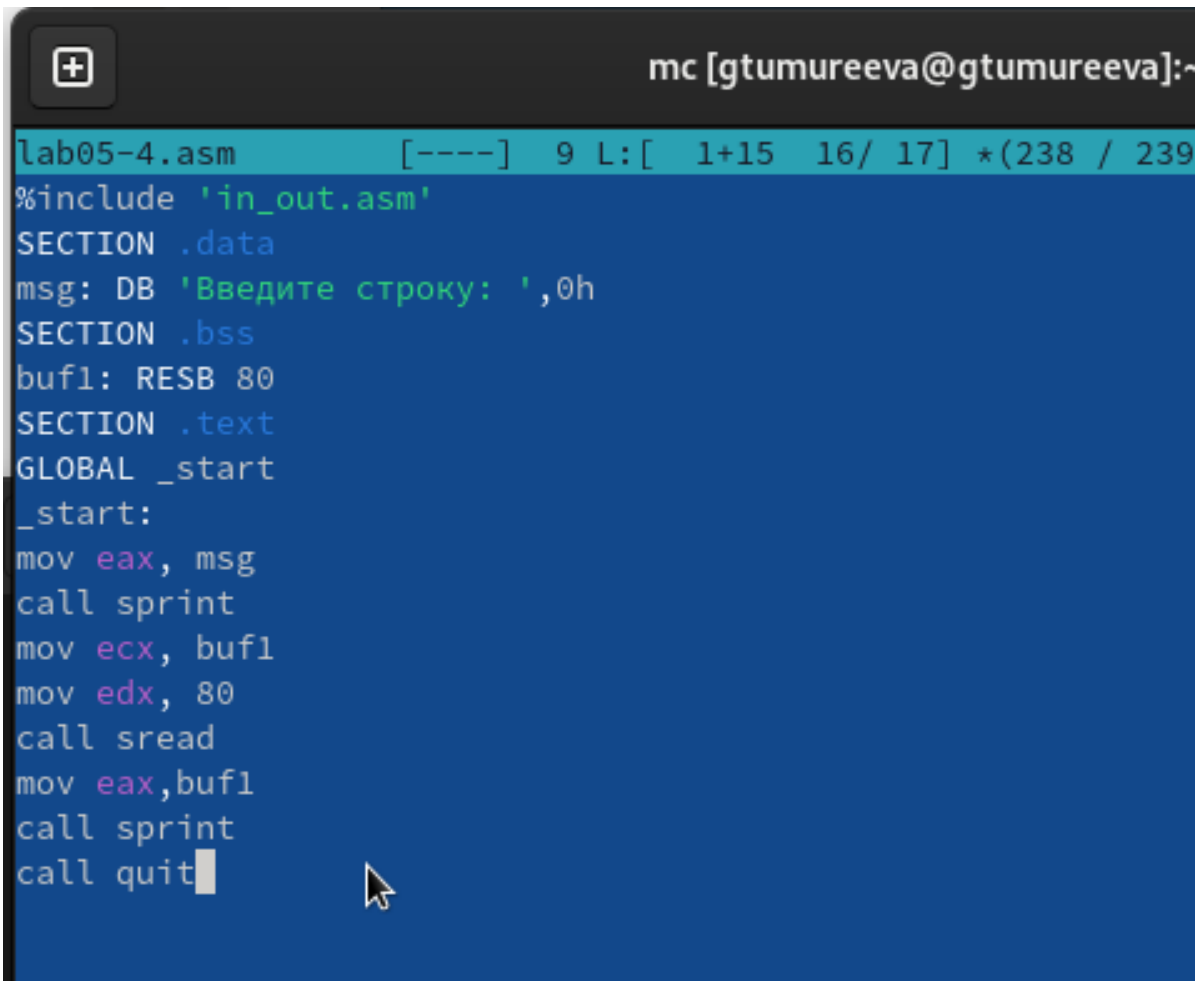
```

[gtumureeva@gtumureeva lab05]$ nasm -f elf lab05-3.asm
[gtumureeva@gtumureeva lab05]$ ld -m elf_i386 lab05-3.o -o lab05-3
[gtumureeva@gtumureeva lab05]$ ./lab05-3
Введите строку:
Galina
Galina
[gtumureeva@gtumureeva lab05]$
[gtumureeva@gtumureeva lab05]$

```

Рис. 2.12: Запуск программы lab05-3.asm

15. Скопировала файл lab05-2.asm и адаптировала код таким же образом, чтобы он запрашивал ввод строки и выводил её на экран.



```

mc [gtumureeva@gtumureeva]:~
lab05-4.asm  [----]  9  L:[  1+15  16/ 17]  *(238 / 239
%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

```
[gtumureeva@gtumureeva lab05]$ nasm -f elf lab05-4.asm
[gtumureeva@gtumureeva lab05]$ ld -m elf_i386 lab05-4.o -o lab05-4
[gtumureeva@gtumureeva lab05]$ ./lab05-4
Введите строку: Galina
Galina
[gtumureeva@gtumureeva lab05]$
```

Рис. 2.14: Запуск программы lab05-4.asm

Разница между этими двумя способами заключается в том, что файл `in_out.asm` уже содержит готовые подпрограммы для ввода и вывода данных, поэтому мне просто нужно было разместить данные в соответствующих регистрах и вызвать нужную подпрограмму с помощью инструкции `call`.

3 Выводы

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