

# **Лабораторная работа №5**

**Дисциплина: Архитектура компьютера**

Швед Карина Дмитриевна

# Содержание

<b>1</b>	<b>Цель работы</b>	<b>5</b>
<b>2</b>	<b>Ход работы</b>	<b>6</b>
<b>3</b>	<b>Задание для самостоятельной работы</b>	<b>11</b>
<b>4</b>	<b>Выводы</b>	<b>13</b>
	<b>Список литературы</b>	<b>14</b>

## **Список таблиц**

## Список иллюстраций

2.1	Использования горячих клавиш в МС . . . . .	6
2.2	открытие asm файла в nano . . . . .	7

# 1 Цель работы

Приобретение практических навыков работы в Midnight Commander. Освоение инструкций языка ассемблера `mov` и `int`.

## 2 Ход работы

Сначала я скачала Midnight Commander, открыла его с помощью команды `mc` и, пользуясь клавишами `↑`, `↓` и `Enter` перешла в каталог `~/work/arch-pc` созданный мною при выполнении лабораторной работы №4 (рис. 2.1).

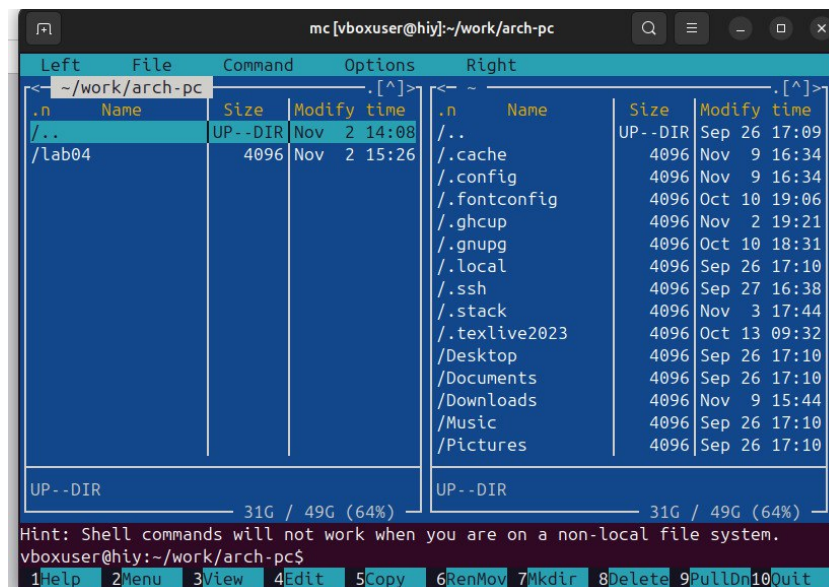
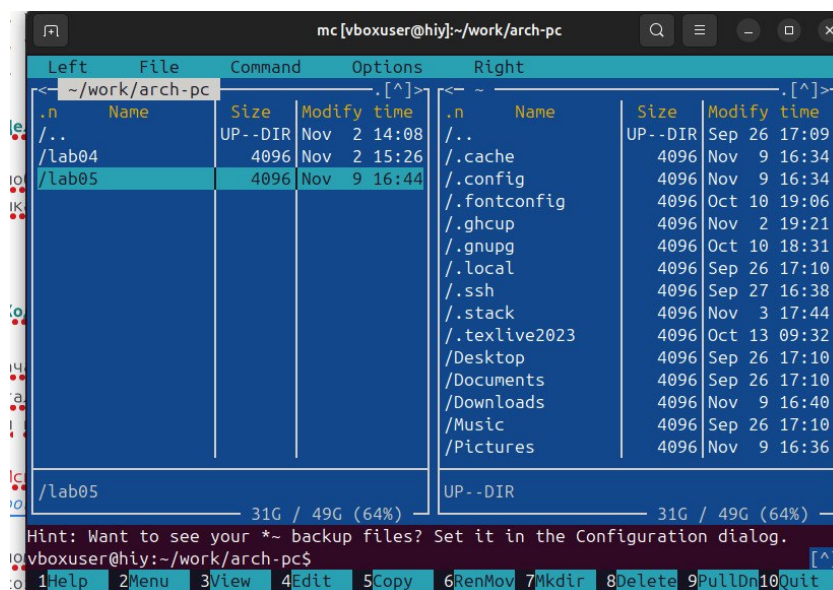


Рис. 2.1: Использование горячих клавиш в MC

С помощью функциональной клавиши `F7` я создала папку `lab05` и перешла в созданный каталог. (рис. ??).



Пользуясь строкой

ввода и командой touch я создала файл lab5-1.asm

С помощью функциональной клавиши F4 я открыла файл lab5-1.asm для редактирования во встроенном редакторе. (рис. 2.2).

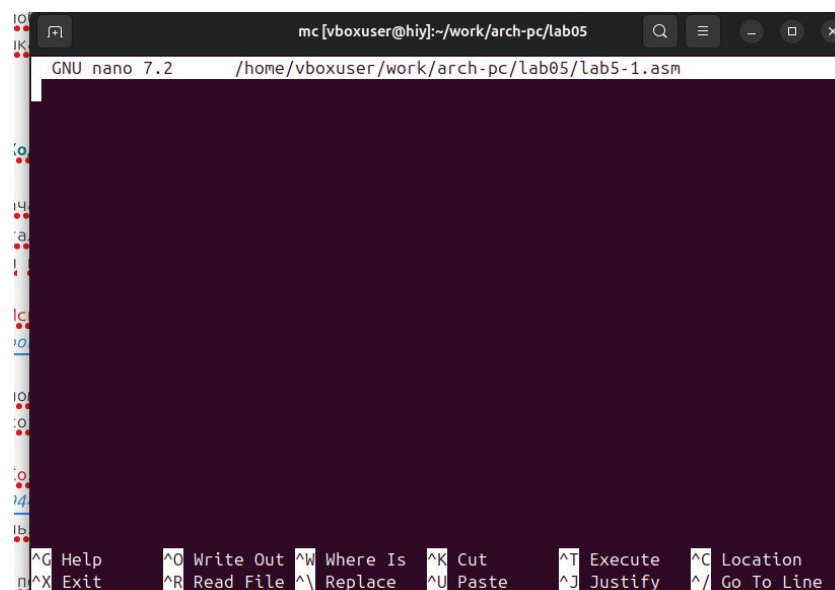
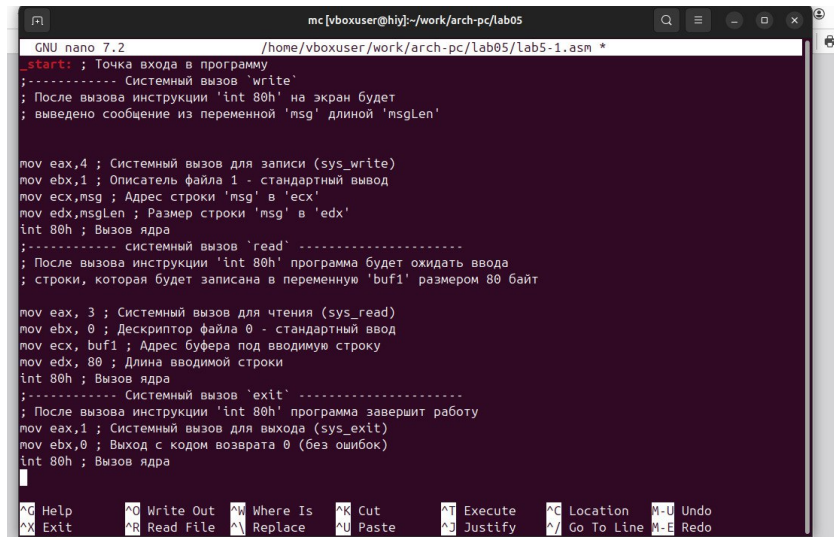


Рис. 2.2: открытие asm файла в nano

Далее ввела текст программы из листинга 5.1, сохранила изменения и закры-



```
mc [vboxuser@hty]:~/work/arch-pc/lab05
GNU nano 7.2 /home/vboxuser/work/arch-pc/lab05/lab5-1.asm
_start: ; Точка входа в программу
;----- Системный вызов 'write'
; После вызова инструкции 'int 80h' на экран будет
; выведено сообщение из переменной 'msg' длиной 'msgLen'

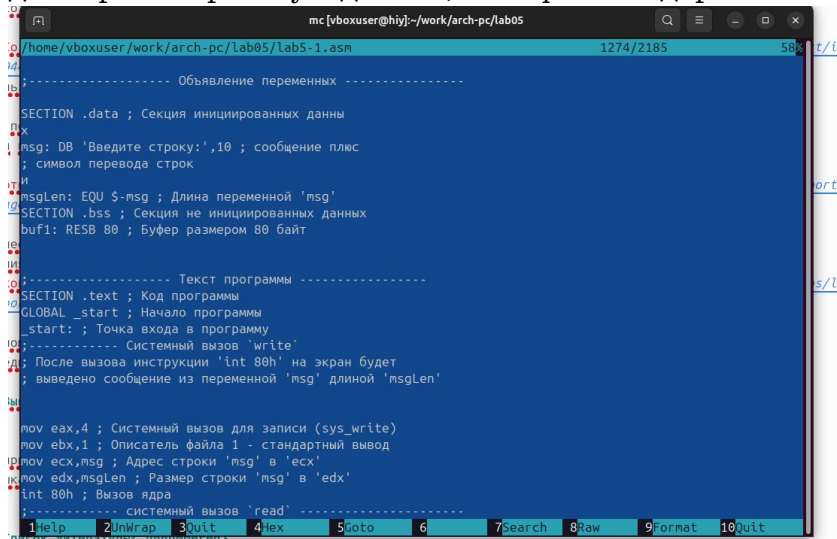
mov eax,4 ; Системный вызов для записи (sys_write)
mov ebx,1 ; Описатель файла 1 - стандартный вывод
mov ecx,msg ; Адрес строки 'msg' в 'ecx'
mov edx,msgLen ; Размер строки 'msg' в 'edx'
int 80h ; Вызов ядра
;----- системный вызов 'read' -----
; После вызова инструкции 'int 80h' программа будет ожидать ввода
; строки, которая будет записана в переменную 'buf1' размером 80 байт

mov eax, 3 ; Системный вызов для чтения (sys_read)
mov ebx, 0 ; Descriptor файла 0 - стандартный ввод
mov ecx, buf1 ; Адрес буфера под вводимую строку
mov edx, 80 ; Длина вводимой строки
int 80h ; Вызов ядра
;----- Системный вызов 'exit' -----
; После вызова инструкции 'int 80h' программа завершит работу
mov eax,1 ; Системный вызов для выхода (sys_exit)
mov ebx,0 ; Выход с кодом возврата 0 (без ошибок)
int 80h ; Вызов ядра

^C Help      ^O Write Out ^W Where Is  ^K Cut       ^T Execute  ^C Location  ^U Undo
^X Exit      ^R Read File ^L Replace   ^U Paste     ^J Justify  ^_ Go To Line ^E Redo
```

ла файл. (рис. ??).

С помощью функциональной клавиши F3 я открыла файл lab5-1.asm для просмотра и убедилась, что файл содержит текст программы.(рис. ??).



```
mc [vboxuser@hty]:~/work/arch-pc/lab05
/home/vboxuser/work/arch-pc/lab05/lab5-1.asm 1274/2185 58% t/in
;----- Объявление переменных -----
SECTION .data ; Секция инициализированных данных
msg: DB 'Введите строку:',10 ; сообщение плюс
; символ перевода строк
msgLen: EQU $-msg ; Длина переменной 'msg'
SECTION .bss ; Секция не инициализированных данных
buf1: RESB 80 ; Буфер размером 80 байт

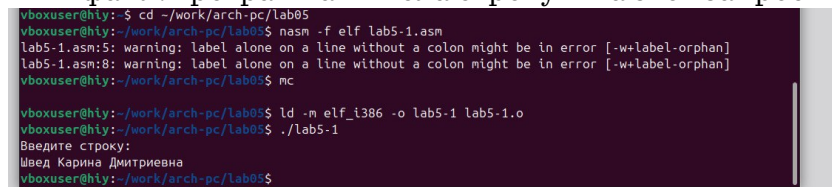
;----- Текст программы -----
SECTION .text ; Код программы
GLOBAL _start ; Начало программы
_start: ; Точка входа в программу
;----- Системный вызов 'write'
; После вызова инструкции 'int 80h' на экран будет
; выведено сообщение из переменной 'msg' длиной 'msgLen'

mov eax,4 ; Системный вызов для записи (sys_write)
mov ebx,1 ; Описатель файла 1 - стандартный вывод
mov ecx,msg ; Адрес строки 'msg' в 'ecx'
mov edx,msgLen ; Размер строки 'msg' в 'edx'
int 80h ; Вызов ядра
;----- системный вызов 'read' -----
; После вызова инструкции 'int 80h' программа будет ожидать ввода
; строки, которая будет записана в переменную 'buf1' размером 80 байт

mov eax, 3 ; Системный вызов для чтения (sys_read)
mov ebx, 0 ; Descriptor файла 0 - стандартный ввод
mov ecx, buf1 ; Адрес буфера под вводимую строку
mov edx, 80 ; Длина вводимой строки
int 80h ; Вызов ядра
;----- Системный вызов 'exit' -----
; После вызова инструкции 'int 80h' программа завершит работу
mov eax,1 ; Системный вызов для выхода (sys_exit)
mov ebx,0 ; Выход с кодом возврата 0 (без ошибок)
int 80h ; Вызов ядра

1 Help      2 UnWrap  3 Quit      4 lex       5 goto      6
inбек  литературы  оптимизатор
```

Далее я оттранслировала текст программы lab5-1.asm в объектный файл, выполнила компоновку объектного файла и запустила получившийся исполняемый файл. Программа вывела строку и на этот запрос я ввела свои ФИО (рис. ??).



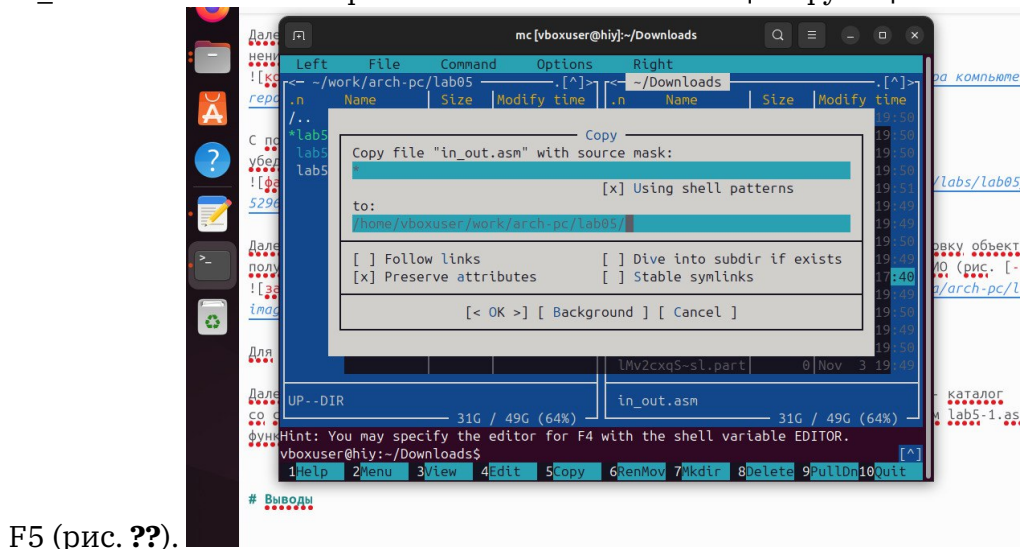
```
vboxuser@hty:~/work/arch-pc/lab05$ cd ~/work/arch-pc/lab05
vboxuser@hty:~/work/arch-pc/lab05$ nasm -f elf lab5-1.asm
lab5-1.asm:5: warning: label alone on a line without a colon might be in error [-w+label-orphan]
lab5-1.asm:8: warning: label alone on a line without a colon might be in error [-w+label-orphan]
vboxuser@hty:~/work/arch-pc/lab05$ mc

vboxuser@hty:~/work/arch-pc/lab05$ ld -m elf_i386 -o lab5-1 lab5-1.o
vboxuser@hty:~/work/arch-pc/lab05$ ./lab5-1
Введите строку:
Введ Карина Дмитриевна
vboxuser@hty:~/work/arch-pc/lab05$
```

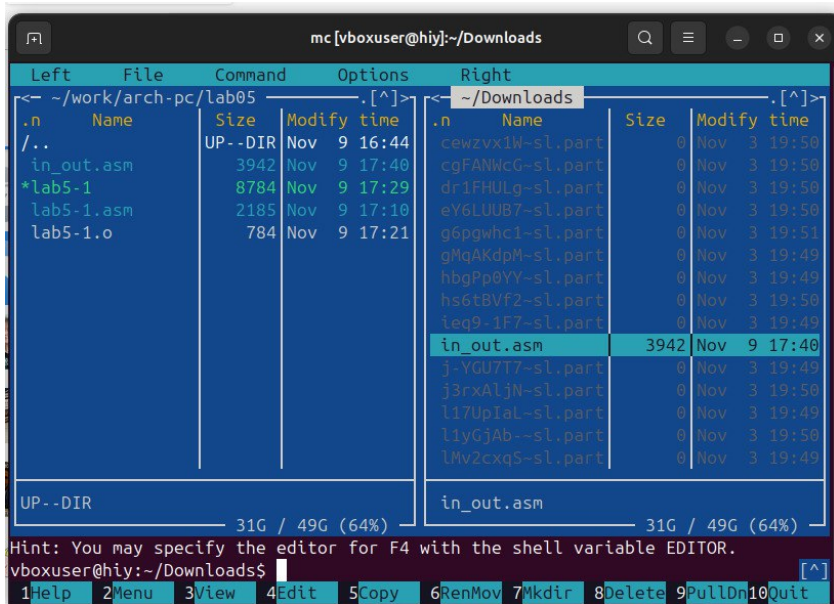
Для подключения внешних файлов я скачала файл in\_out.asm со страницы курса в ТУИС



Далее в одной из панелей mc я открыла каталог с файлом lab5-1.asm, а в другой панели - каталог со скаченным файлом in\_out.asm. Затем я скопировала файл in\_out.asm в каталог с файлом lab5-1.asm с помощью функциональной клавиши



F5 (рис. ??).



(рис. ??).

Далее с помощью функциональной клавиши F6 я создала копию файла lab5-1.asm с именем lab5-2.asm

Далее я исправила текст программы в файле lab5-2.asm с использованием подпрограмм из внешнего файла in\_out.asm в соответствии с листингом 5.2. Создала исполняемый файл и проверила его работу.

```
vboxuser@hiy: ~/work/arch-pc/lab05
vboxuser@hiy:~$ nasm -f elf lab5-2.asm -o lab5-2.o
nasm: fatal: unable to open input file `lab5-2.asm' No such file or directory
vboxuser@hiy:~$ cd work
vboxuser@hiy:~/work$ cd arch-pc
vboxuser@hiy:~/work/arch-pc$ ls
lab04  lab05
vboxuser@hiy:~/work/arch-pc$ cd lab05
vboxuser@hiy:~/work/arch-pc/lab05$ nasm -f elf lab5-2.asm -o lab5-2.o
vboxuser@hiy:~/work/arch-pc/lab05$ ld -m elf_i386 -o lab5-2 lab5-2.o
vboxuser@hiy:~/work/arch-pc/lab05$ ./lab5-2
Введите строку:
```

(рис. ??).

Далее в файле lab5-2.asm заменила подпрограмму sprintLF на sprin и создала исполняемый файл, проверила его работу. Теперь после вывода строки она не завершается символом перехода на новую строку.

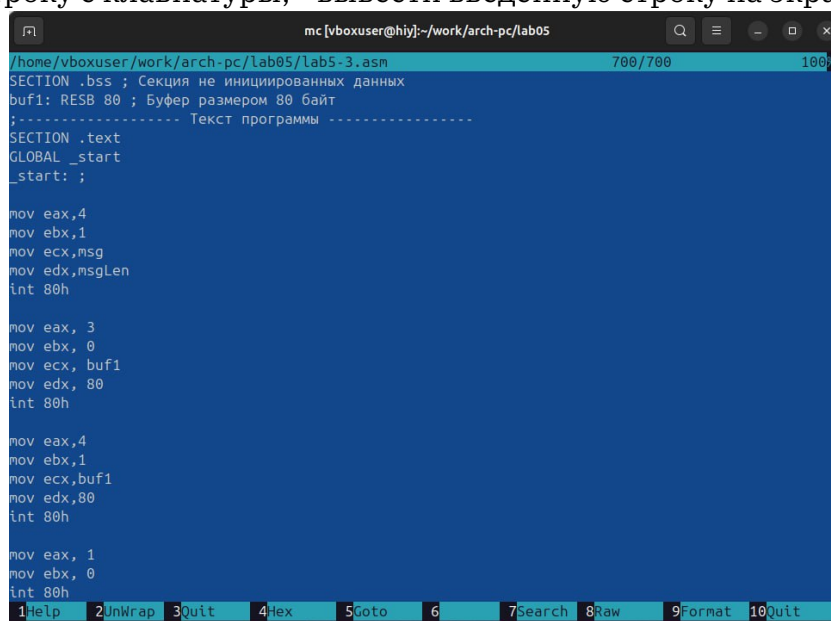
```
vboxuser@hiy: ~/work/arch-pc/lab05
nasm: fatal: unable to open input file `lab5-2.asm' No such file or directory
vboxuser@hiy:~$ cd work
vboxuser@hiy:~/work$ cd arch-pc
vboxuser@hiy:~/work/arch-pc$ ls
lab04  lab05
vboxuser@hiy:~/work/arch-pc$ cd lab05
vboxuser@hiy:~/work/arch-pc/lab05$ nasm -f elf lab5-2.asm -o lab5-2.o
vboxuser@hiy:~/work/arch-pc/lab05$ ld -m elf_i386 -o lab5-2 lab5-2.o
vboxuser@hiy:~/work/arch-pc/lab05$ ./lab5-2
Введите строку:
Швед Карина Дмитриевна
vboxuser@hiy:~/work/arch-pc/lab05$ mc
vboxuser@hiy:~/work/arch-pc/lab05$ 10
10: command not found
vboxuser@hiy:~/work/arch-pc/lab05$ 10
10: command not found
vboxuser@hiy:~/work/arch-pc/lab05$ ld -m elf_i386 -o lab5-2 lab5-2.o
vboxuser@hiy:~/work/arch-pc/lab05$ ./lab5-2
Введите строку:
Карина Швед
vboxuser@hiy:~/work/arch-pc/lab05$
```

(рис. ??).

### 3 Задание для самостоятельной работы

Я создала копию файла lab5-1.asm с именем lab5-3.asm и внесла изменения в программу (без использования внешнего файла in\_out.asm), так чтобы она работала по следующему алгоритму:

- вывести приглашение типа “Введите строку:”;
- ввести строку с клавиатуры;
- вывести введенную строку на экран.



```
mc [vboxuser@hiy]:~/work/arch-pc/lab05
/home/vboxuser/work/arch-pc/lab05/lab5-3.asm 700/700 100%
SECTION .bss ; Секция не инициализированных данных
buf1: RESB 80 ; Буфер размером 80 байт
;----- Текст программы -----
SECTION .text
GLOBAL _start
_start: ;

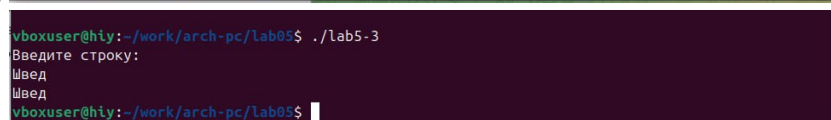
mov eax,4
mov ebx,1
mov ecx,msg
mov edx,msgLen
int 80h

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. ??).



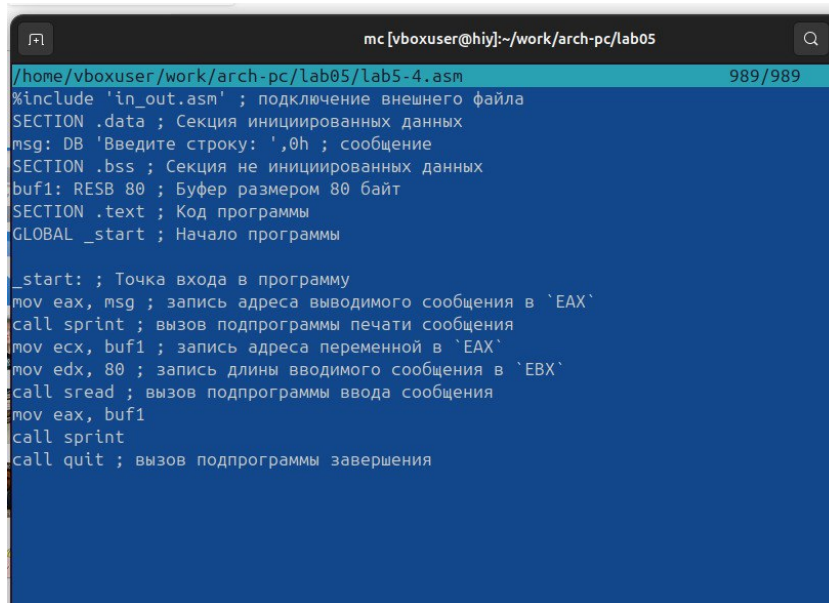
```
vboxuser@hiy:~/work/arch-pc/lab05$ ./lab5-3
Введите строку:
Швед
vboxuser@hiy:~/work/arch-pc/lab05$
```

(рис. ??).

Далее я создала копию файла lab5-2.asm с именем lab5-4 и исправила текст программы с использованием подпрограмм из внешнего файла in\_out.asm, так чтобы она работала по следующему алгоритму:

- вывести приглашение типа

“Введите строку:”; • ввести строку с клавиатуры; • вывести введенную строку на экран.

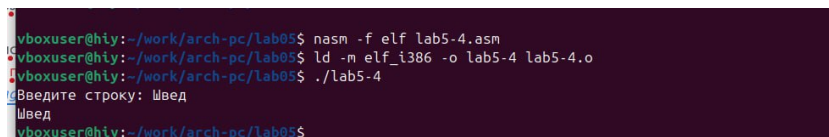


```
mc[vboxuser@hiy]:~/work/arch-pc/lab05
/home/vboxuser/work/arch-pc/lab05/lab5-4.asm 989/989
%include 'in_out.asm' ; подключение внешнего файла
SECTION .data ; Секция иницированных данных
msg: DB 'Введите строку: ',0h ; сообщение
SECTION .bss ; Секция не иницированных данных
buf1: RESB 80 ; Буфер размером 80 байт
SECTION .text ; Код программы
GLOBAL _start ; Начало программы

_start: ; Точка входа в программу
mov eax, msg ; запись адреса выводимого сообщения в `EAX`
call sprint ; вызов подпрограммы печати сообщения
mov ecx, buf1 ; запись адреса переменной в `EAX`
mov edx, 80 ; запись длины вводимого сообщения в `EBX`
call sread ; вызов подпрограммы ввода сообщения
mov eax, buf1
call sprint
call quit ; вызов подпрограммы завершения
```

(рис. ??).

Я создала исполняемый файл и проверила работу программы. Отличие этих двух программ в том, что файл in\_out.asm содержит уже готовые подпрограммы для обеспечения ввода/вывода. Таким образом, остается только разместить данные в нужных регистрах и вызвать желаемую подпрограмму с помощью инструкции call.



```
vboxuser@hiy:~/work/arch-pc/lab05$ nasm -f elf lab5-4.asm
vboxuser@hiy:~/work/arch-pc/lab05$ ld -m elf_i386 -o lab5-4 lab5-4.o
vboxuser@hiy:~/work/arch-pc/lab05$ ./lab5-4
Введите строку: Швед
vboxuser@hiy:~/work/arch-pc/lab05$
```

(рис. ??).

## 4 Выводы

Я приобрела практические навыки работы в Midnight Commander и освоила инструкции языка ассемблера `mov` и `int`.

## Список литературы

1. GDB: The GNU Project Debugger. — URL: <https://www.gnu.org/software/gdb/>.
2. GNU Bash Manual. — 2016. — URL: <https://www.gnu.org/software/bash/manual/>.
3. Midnight Commander Development Center. — 2021. — URL: <https://midnight-commander.org/>.
4. NASM Assembly Language Tutorials. — 2021. — URL: <https://asmtutor.com/>.
5. Newham C. Learning the bash Shell: Unix Shell Programming. — O'Reilly Media, 2005. — 354 с. — (In a Nutshell). — ISBN 0596009658. — URL: <http://www.amazon.com/Learning-bash-Shell-Programming-Nutshell/dp/0596009658>.
6. Robbins A. Bash Pocket Reference. — O'Reilly Media, 2016. — 156 с. — ISBN 978-1491941591.
7. The NASM documentation. — 2021. — URL: <https://www.nasm.us/docs.php>.
8. Zarrelli G. Mastering Bash. — Packt Publishing, 2017. — 502 с. — ISBN 9781784396879.
9. Колдаев В. Д., Лупин С. А. Архитектура ЭВМ. — М. : Форум, 2018.
10. Куляс О. Л., Никитин К. А. Курс программирования на ASSEMBLER. — М. : Солон-Пресс,
- 11.
12. Новожилов О. П. Архитектура ЭВМ и систем. — М. : Юрайт, 2016.
13. Расширенный ассемблер: NASM. — 2021. — URL: <https://www.opennet.ru/docs/RUS/nasm/>.
14. Робачевский А., Немнюгин С., Стесик О. Операционная система UNIX. — 2-е изд. — БХВ- Петербург, 2010. — 656 с. — ISBN 978-5-94157-538-1.
15. Столяров А. Программирование на языке ассемблера NASM для ОС Unix. — 2-

- е изд. — М. : МАКС Пресс, 2011. — URL: [http://www.stolyarov.info/books/asm\\_unix](http://www.stolyarov.info/books/asm_unix).
16. Таненбаум Э. Архитектура компьютера. — 6-е изд. — СПб. : Питер, 2013. — 874 с. — (Классика Computer Science).
17. Таненбаум Э., Бос Х. Современные операционные системы. — 4-е изд. — СПб. : Питер,
18. — 1120 с. — (Классика Computer Science).