

Отчет по лабораторной работе №5

дисциплина: Архитектура компьютера

Дельгадильо Валерия

Содержание

1 Цель работы	5
2 Теоретическое введение	6
2.1 Основы работы с Midnight Commander	6
2.2 Структура программы на языке ассемблера NASM	6
3 Лабораторной работы	7
4 Задание для самостоятельной работы	22
4.1 Создайте копию файла lab5-1.asm.	23
4.2 Создайте копию файла lab5-2.asm.	26
5 Выводы	31
6 Список литературы	32

Список иллюстраций

3.1	8
3.2	9
3.3	10
3.4	11
3.5	12
3.6	13
3.7	14
3.8	15
3.9	16
3.10	17
3.11	18
3.12	19
3.13	20
3.14	21
4.1	23
4.2	24
4.3	25
4.4	27
4.5	28
4.6	29

Список таблиц

1 Цель работы

Приобретение практических навыков работы в Midnight Commander.

Освоение инструкций языка ассемблера mov и int.

2 Теоретическое введение

2.1 Основы работы с Midnight Commander

Midnight Commander (или просто mc) — это программа, которая позволяет просматривать структуру каталогов и выполнять основные операции по управлению файловой системой, т.е. mc является файловым менеджером. Midnight Commander позволяет сделать работу с файлами более удобной и наглядной.

2.2 Структура программы на языке ассемблера NASM

Программа на языке ассемблера NASM, как правило, состоит из трёх секций: секция кода программы (SECTION .text), секция инициализированных (известных во время компиляции) данных (SECTION .data) и секция неинициализированных данных (тех, под которые во время компиляции только отводится память, а значение присваивается в ходе выполнения программы) (SECTION .bss).

3 Лабораторной работы

Откройте Midnight Commander user@dk4n31:~\$mc

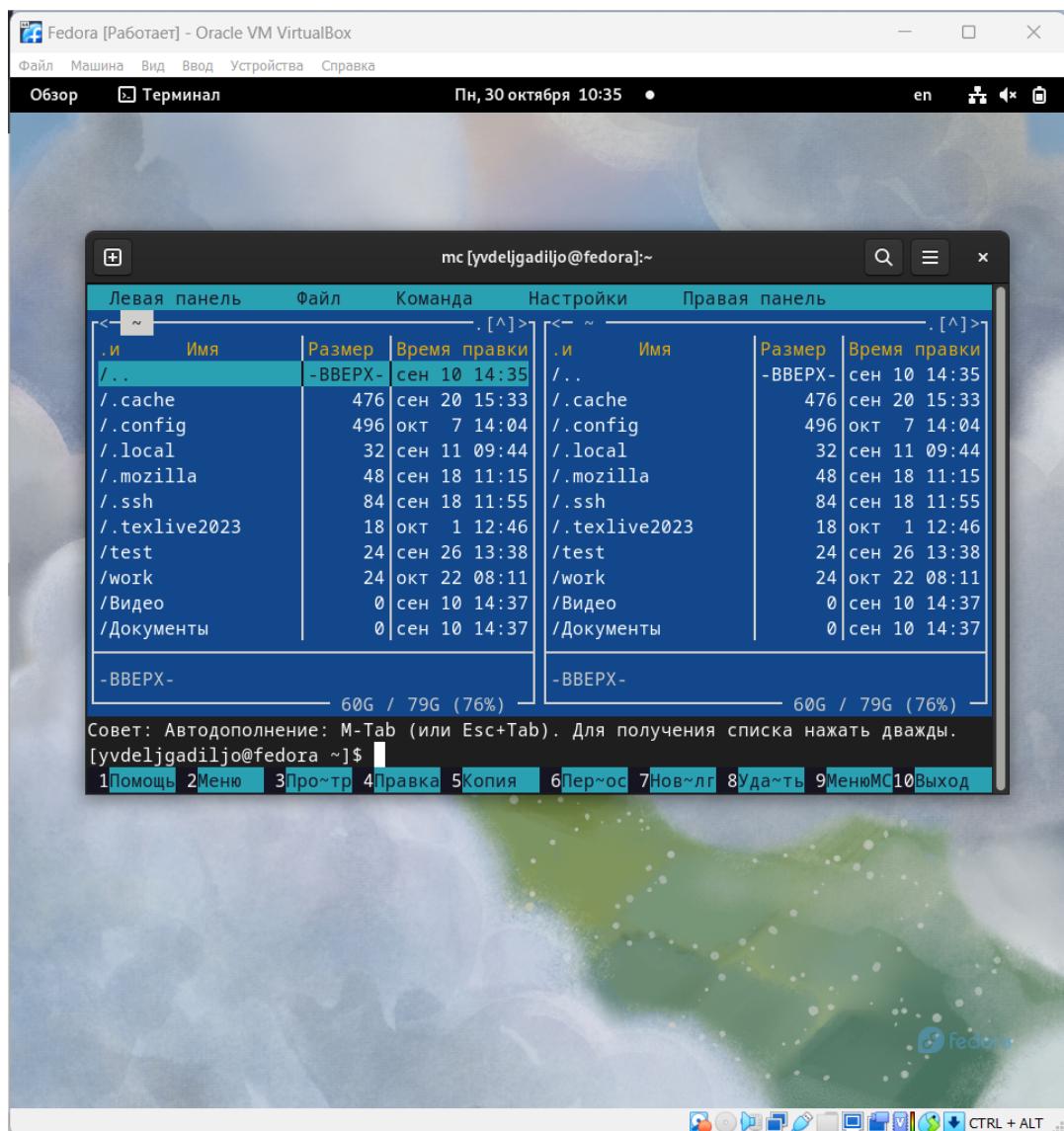


Рис. 3.1:

Пользуясь клавишами **☒**, **☒** и Enter перейдите в каталог `~/work/arch-рс` созданный при выполнении лабораторной работы №4.

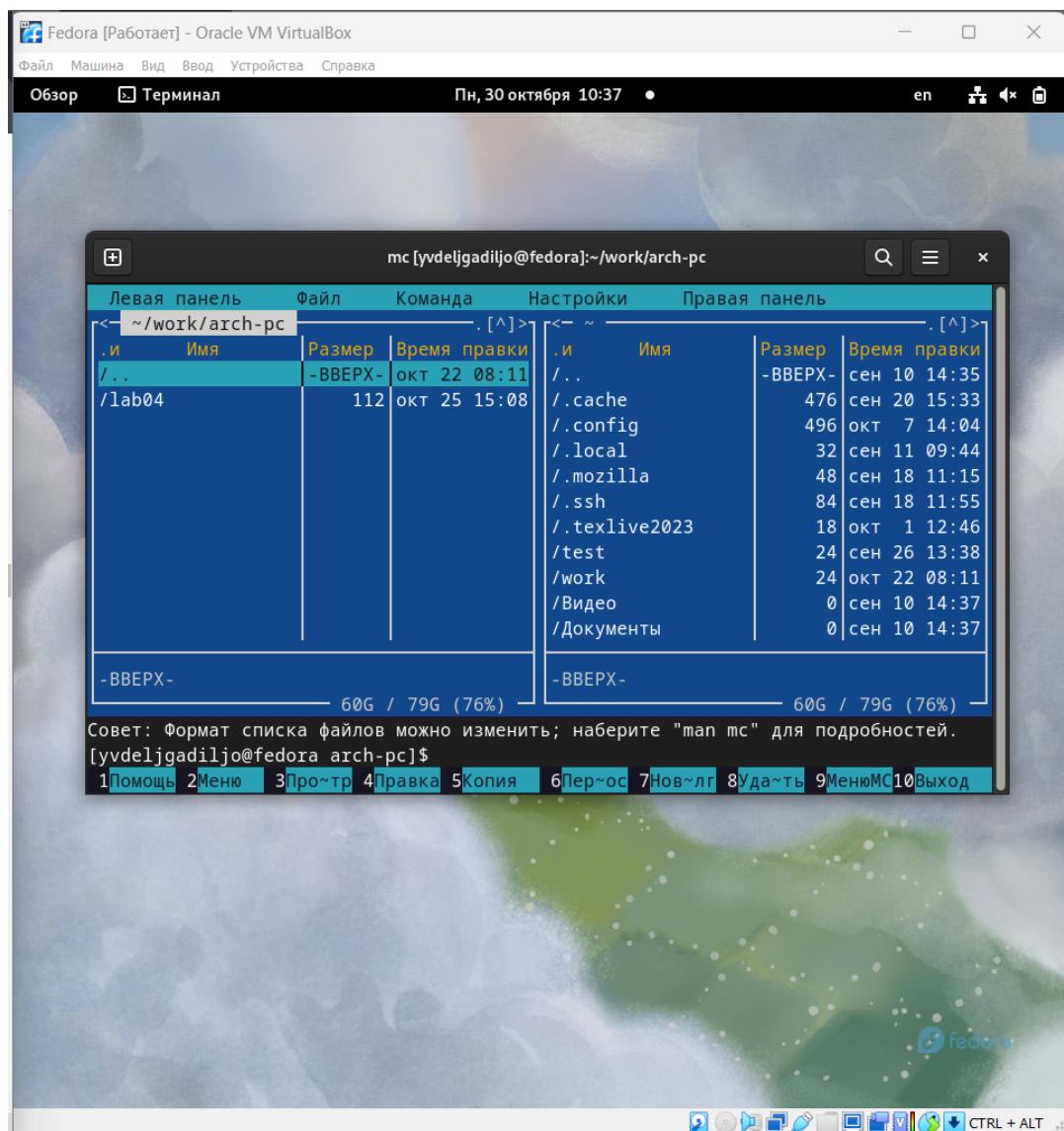


Рис. 3.2:

С помощью функциональной клавиши F7 создайте папку lab05 и перейдите в созданный каталог.

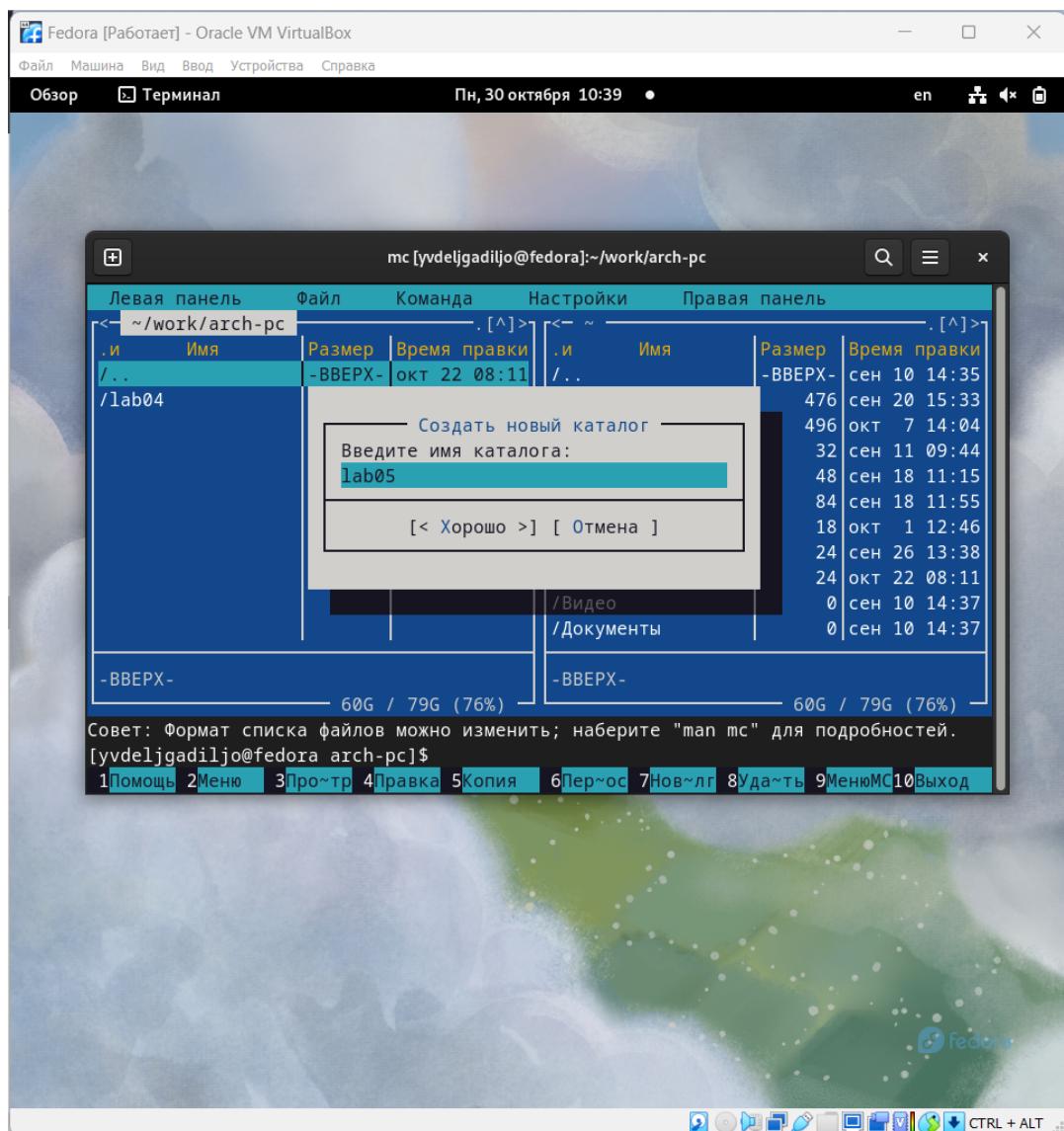


Рис. 3.3:

Пользуясь строкой ввода и командой touch создайте файл lab5-1.asm

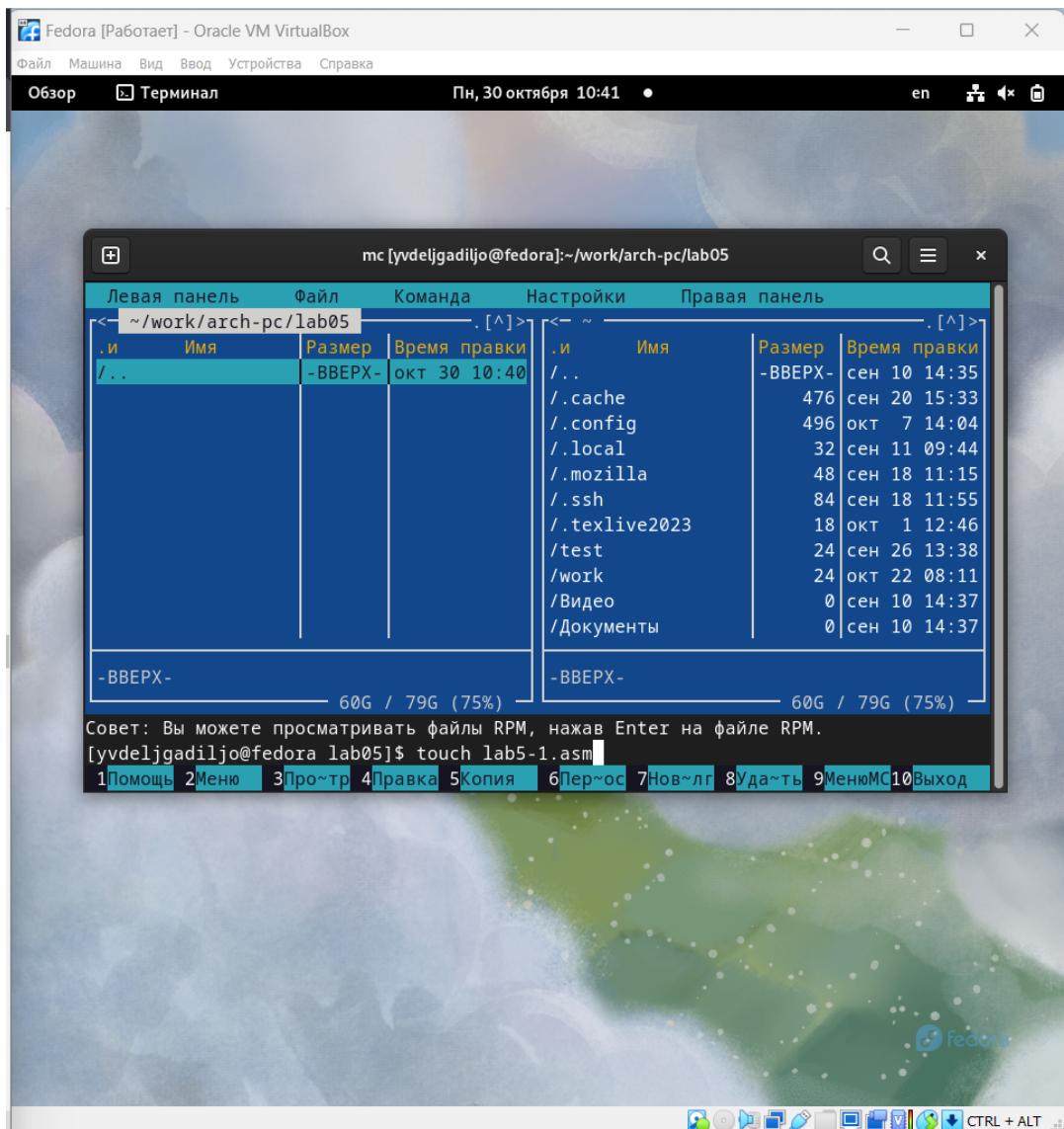


Рис. 3.4:

С помощью функциональной клавиши F4 откройте файл lab5-1.asm для редактирования во встроенным редакторе. Как правило в качестве встроенного редактора Midnight Commander используется редакторы nano или mcedit.

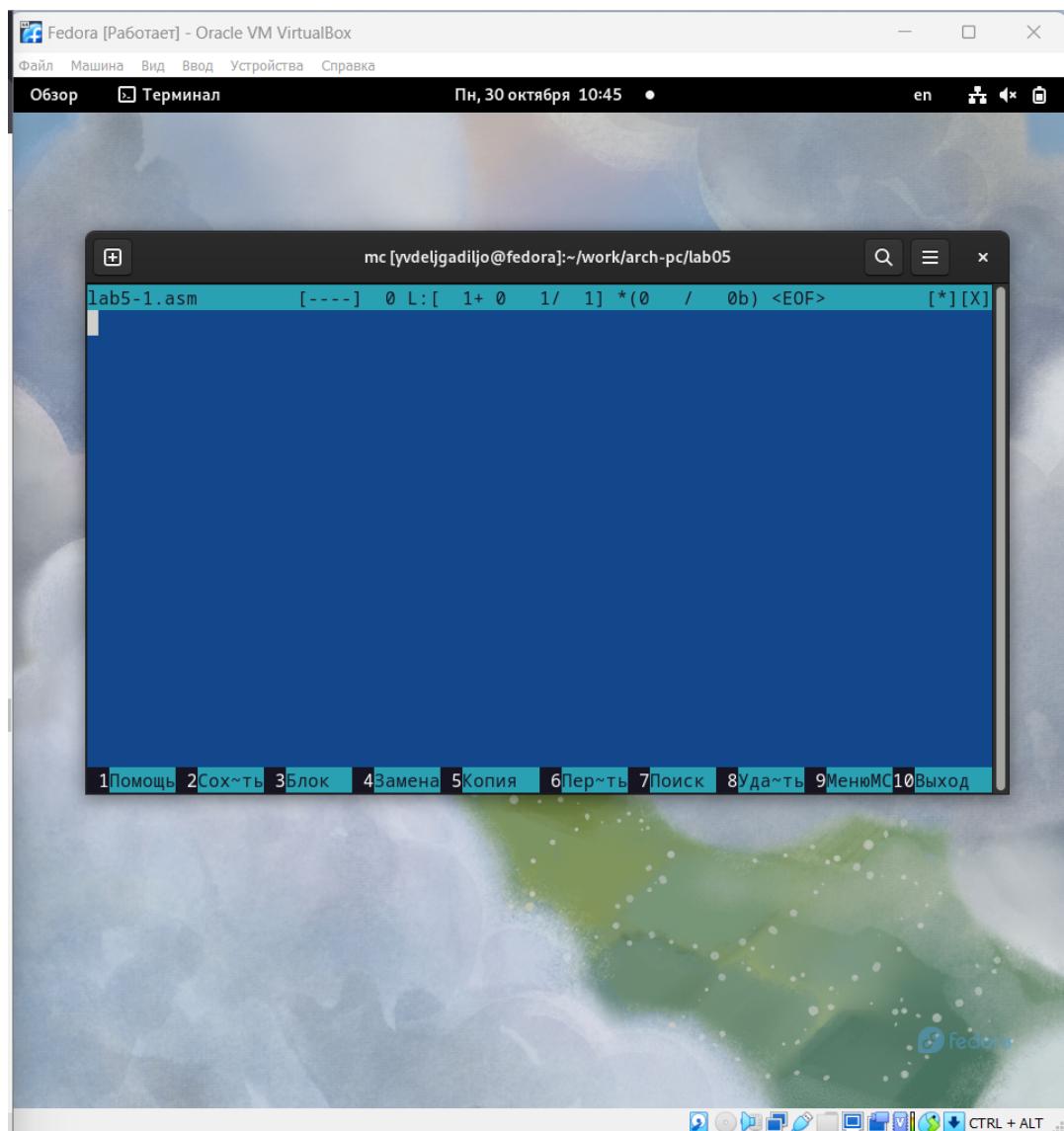


Рис. 3.5:

Введите текст программы из листинга (можно без комментариев), сохраните изменения и закройте файл.

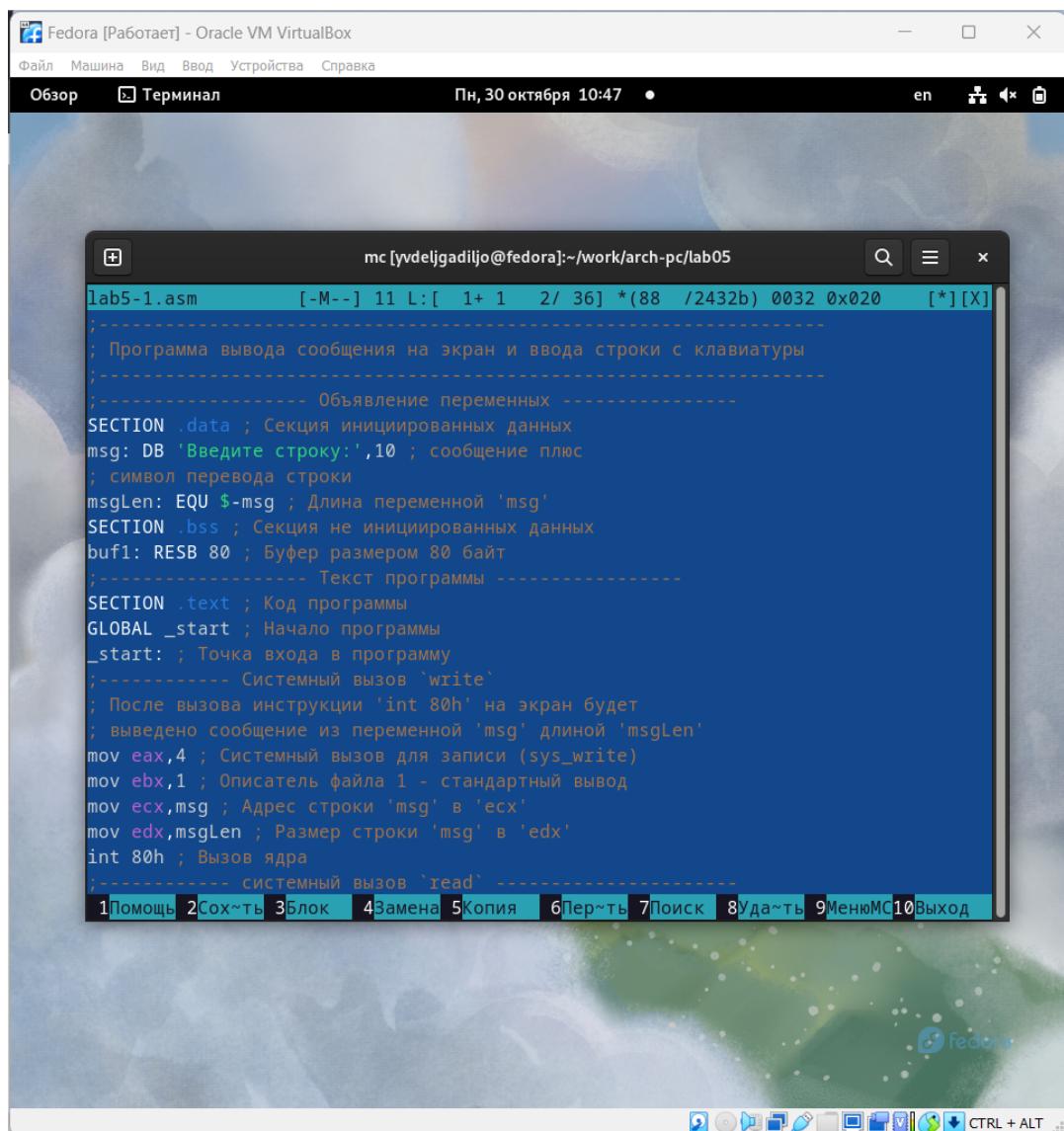


Рис. 3.6:

Оттранслируйте текст программы lab5-1.asm в объектный файл. Выполните компоновку объектного файла и запустите получившийся исполняемый файл. Программа выводит строку 'Введите строку:' и ожидает ввода с клавиатуры. На запрос введите Ваши ФИО.

```
user@dk4n31:~$ nasm -f elf lab5-1.asm
```

```
user@dk4n31:~$ ld -m elf_i386 -o lab5-1 lab5-1.o
```

```
user@dk4n31:~$ ./lab5-1
```

Введите строку:

Имя пользователя

```
user@dk4n31:~$
```

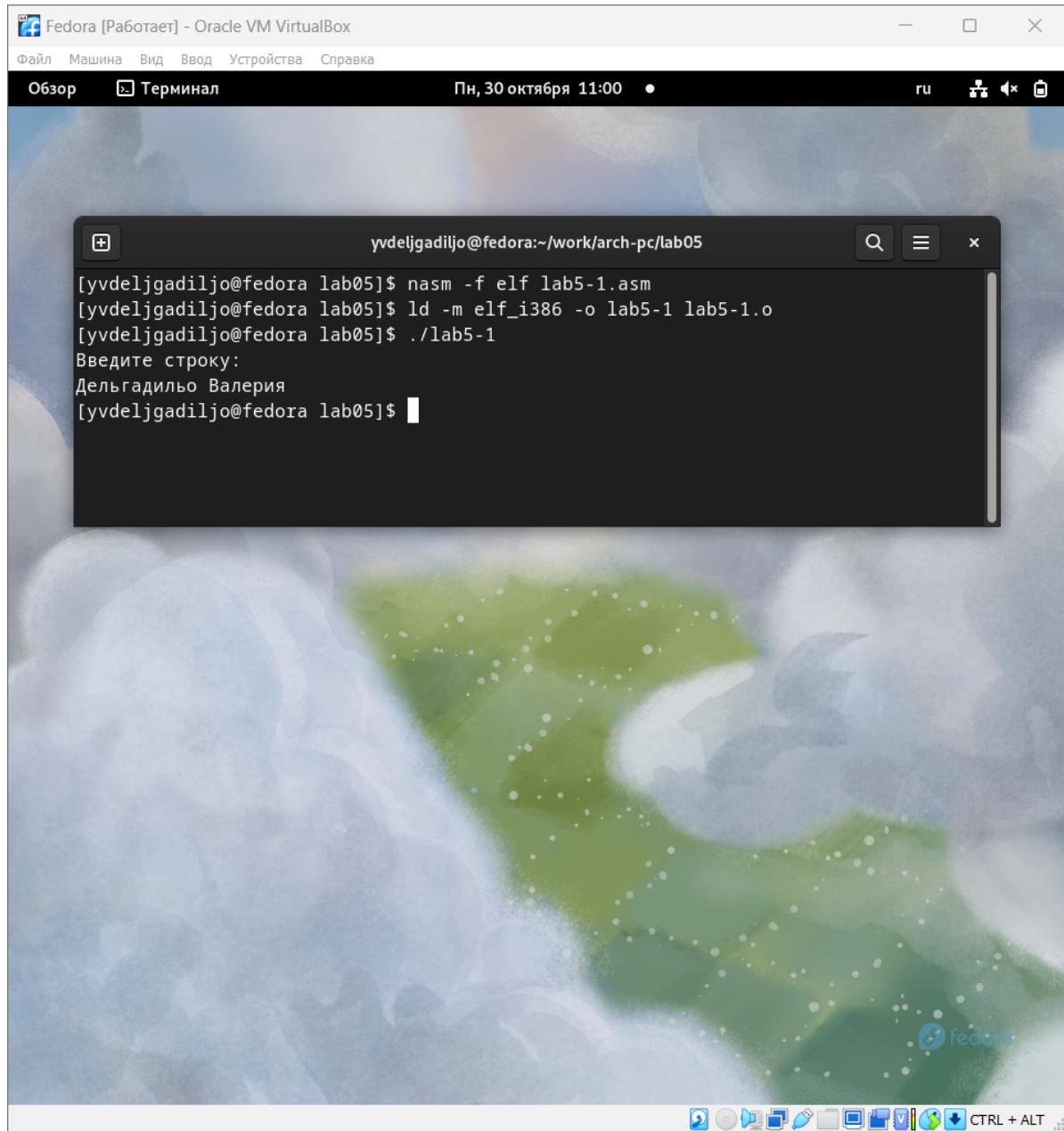


Рис. 3.7:

Подключение внешнего файла in_out.asm

Скачайте файл in_out.asm со страницы курса в ТУИС

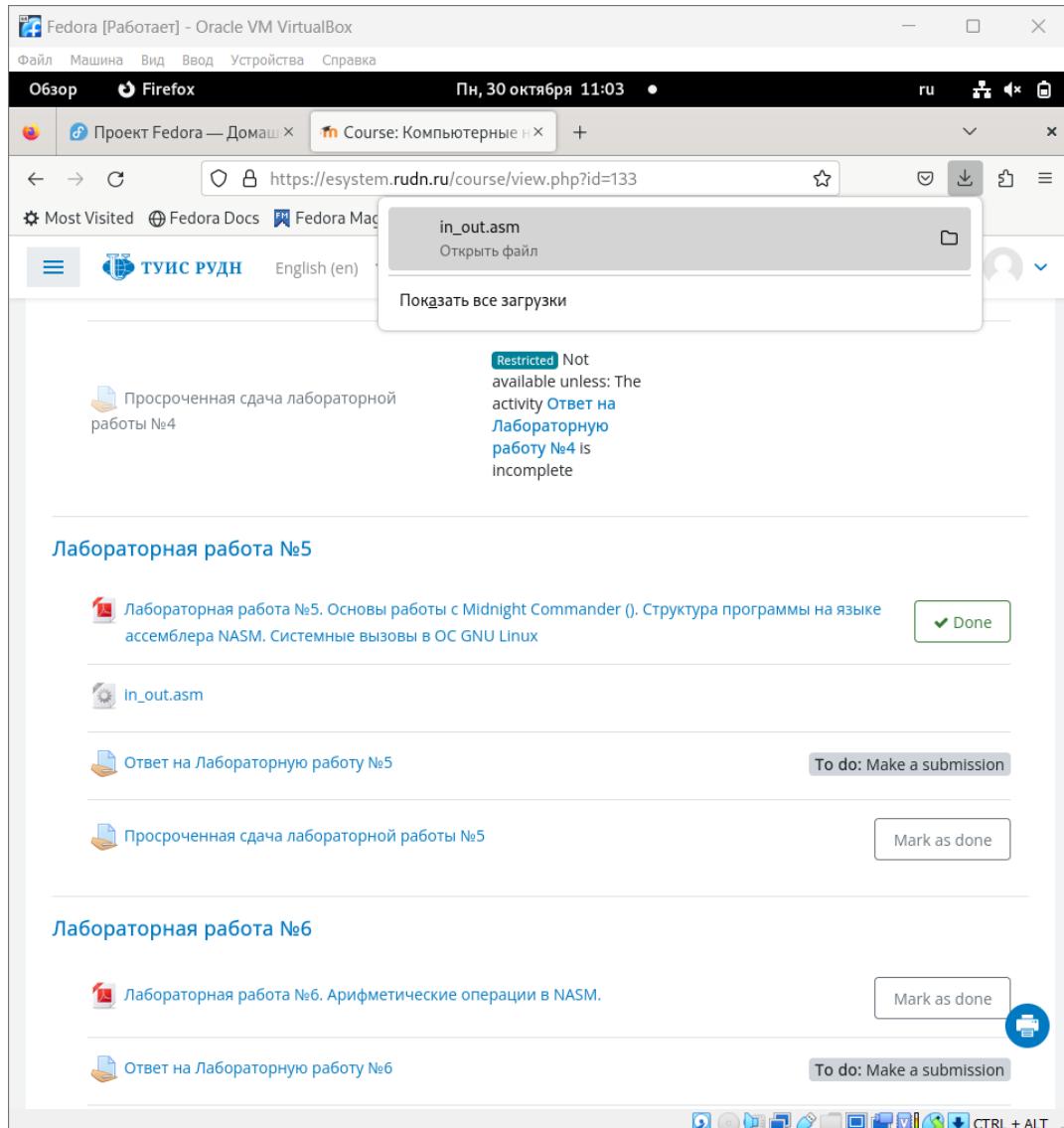


Рис. 3.8:

Подключаемый файл in_out.asm должен лежать в том же каталоге, что и файл с программой, в которой он используется.

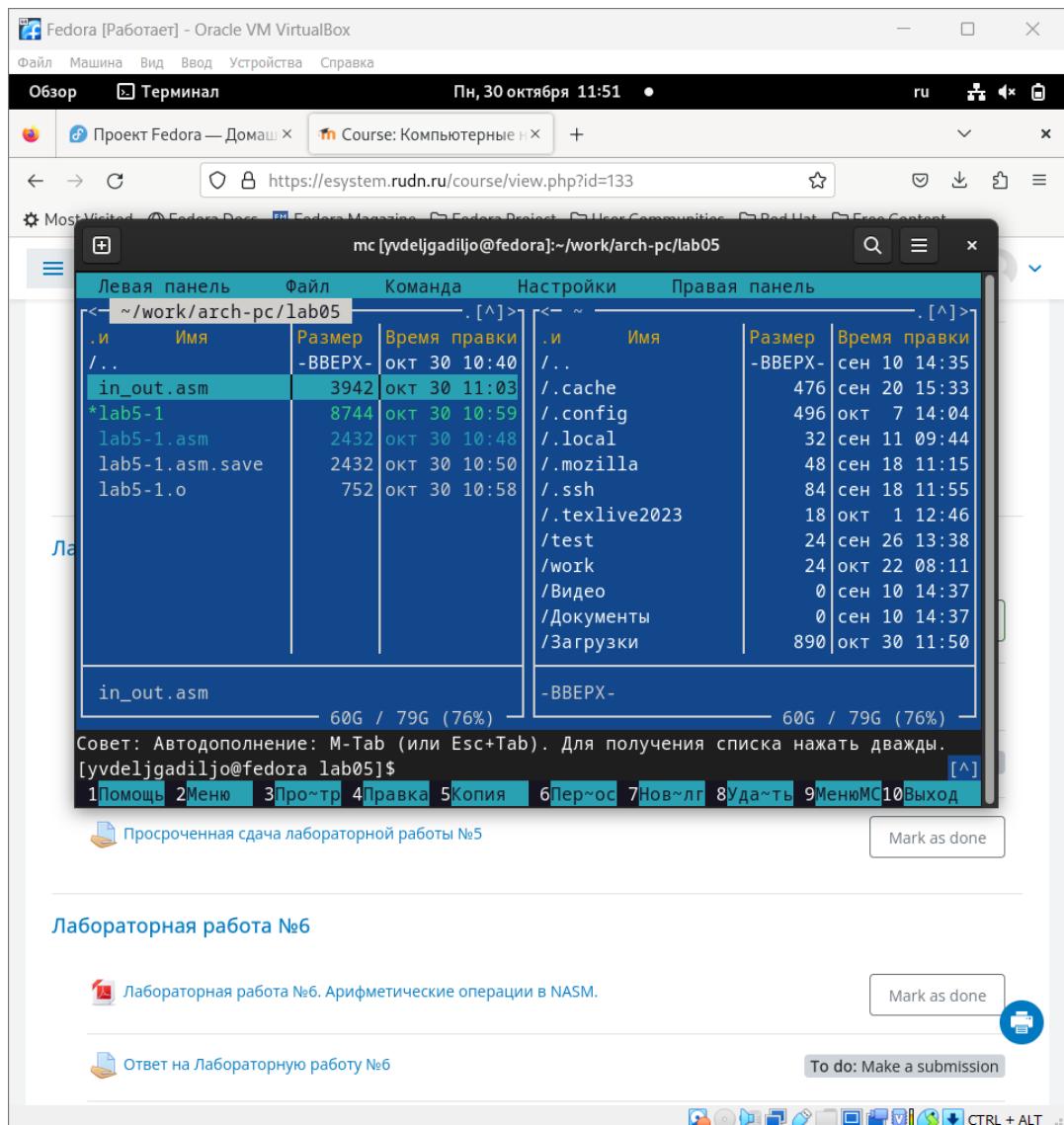


Рис. 3.9:

С помощью функциональной клавиши F6 создайте копию файла lab5-1.asm с именем lab5-2.asm. Выделите файл lab5-1.asm, нажмите клавишу F6, введите имя файла lab5-2.asm и нажмите клавишу Enter.

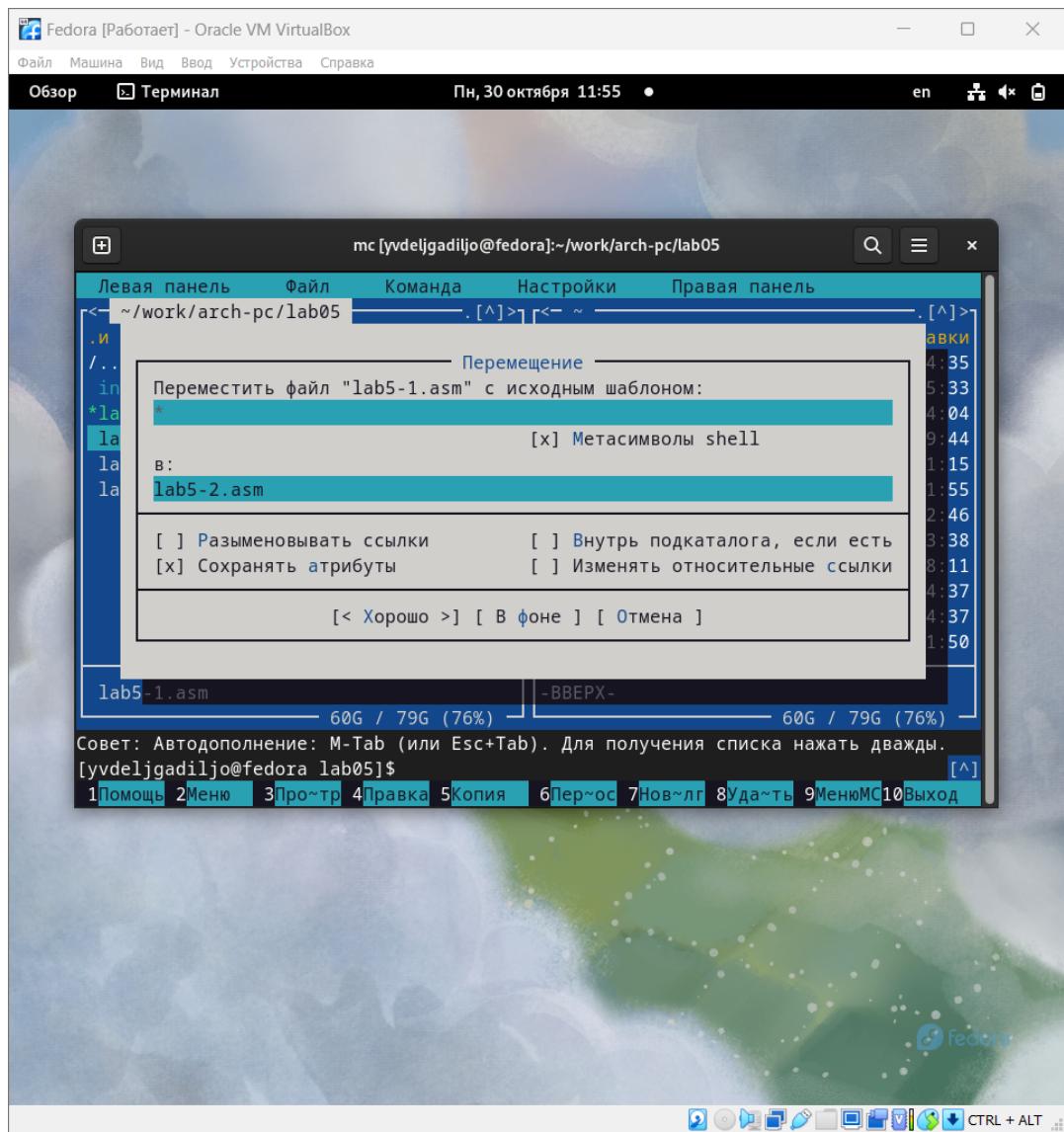


Рис. 3.10:

Исправьте текст программы в файле lab5-2.asm с использованием подпрограмм из внешнего файла in_out.asm (используйте подпрограммы sprintLF, sread и quit) в соответствии с листингом 5.2. Создайте исполняемый файл и проверьте его работу

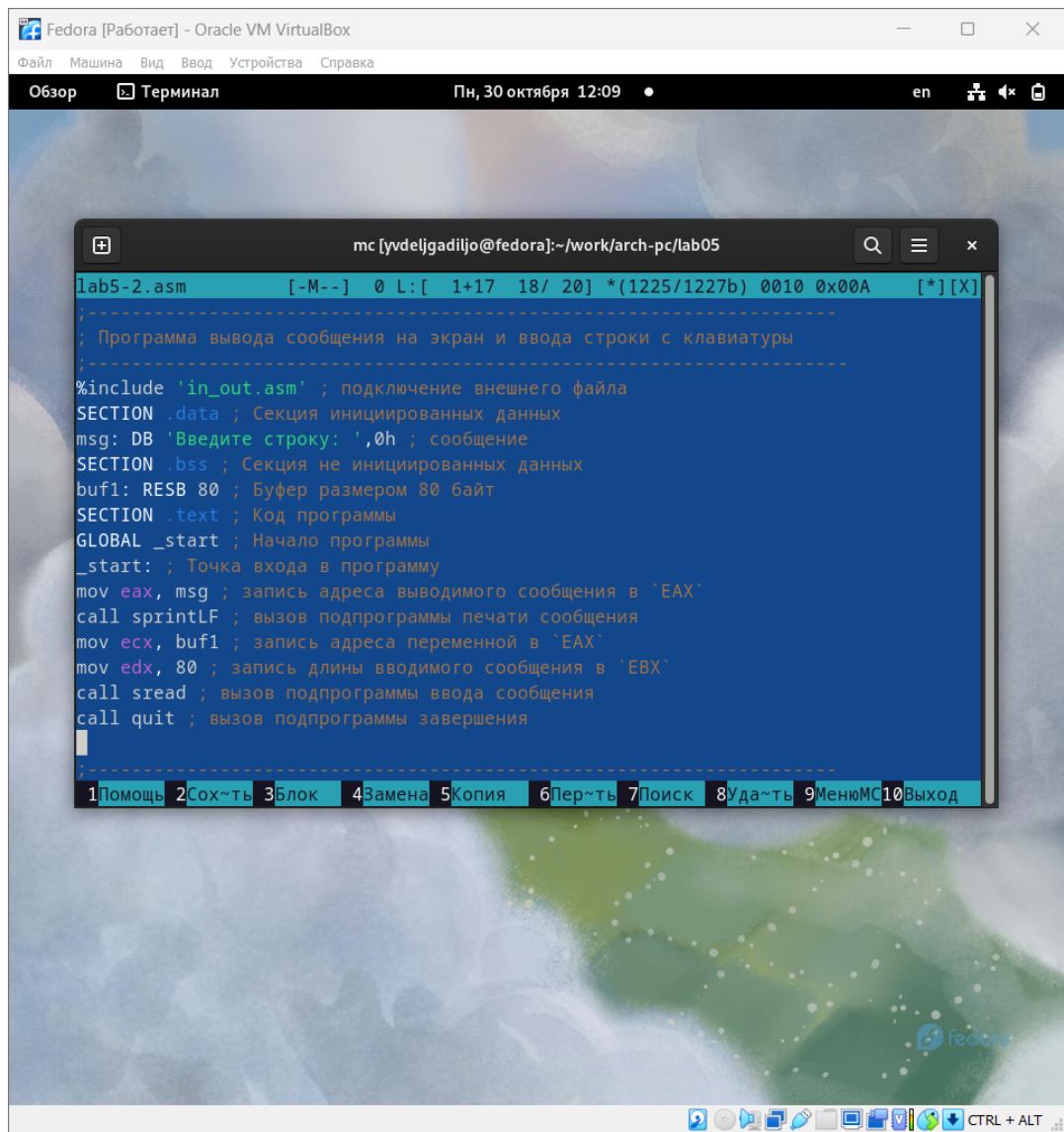


Рис. 3.11:

Создаю объектный файл lab5-2.o, выполняю компоновку объектного файла и запускаю исполняемый файл.

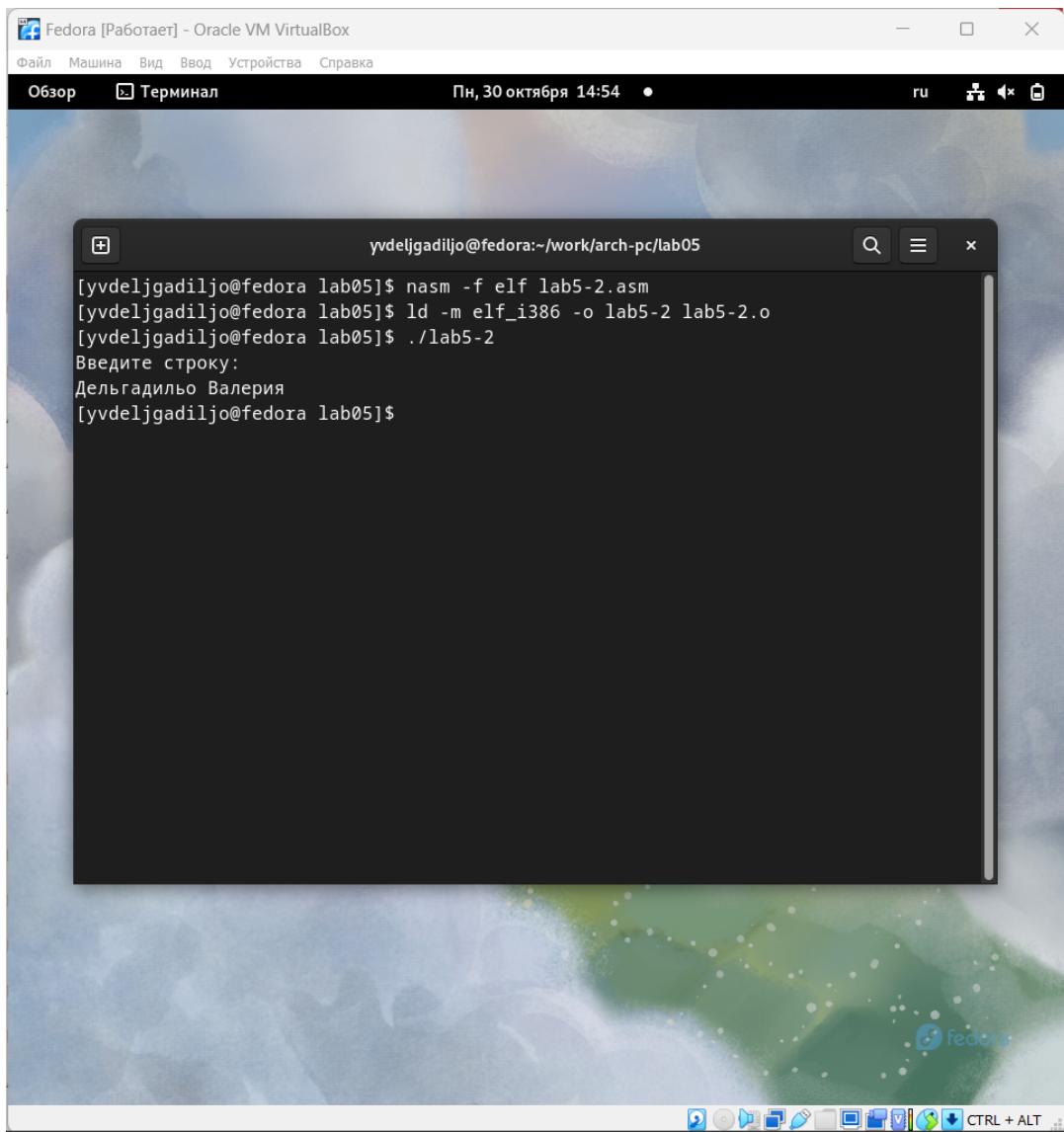


Рис. 3.12:

В файле lab5-2.asm замените подпрограмму sprintLF на sprint. Создайте исполняемый файл и проверьте его работу.

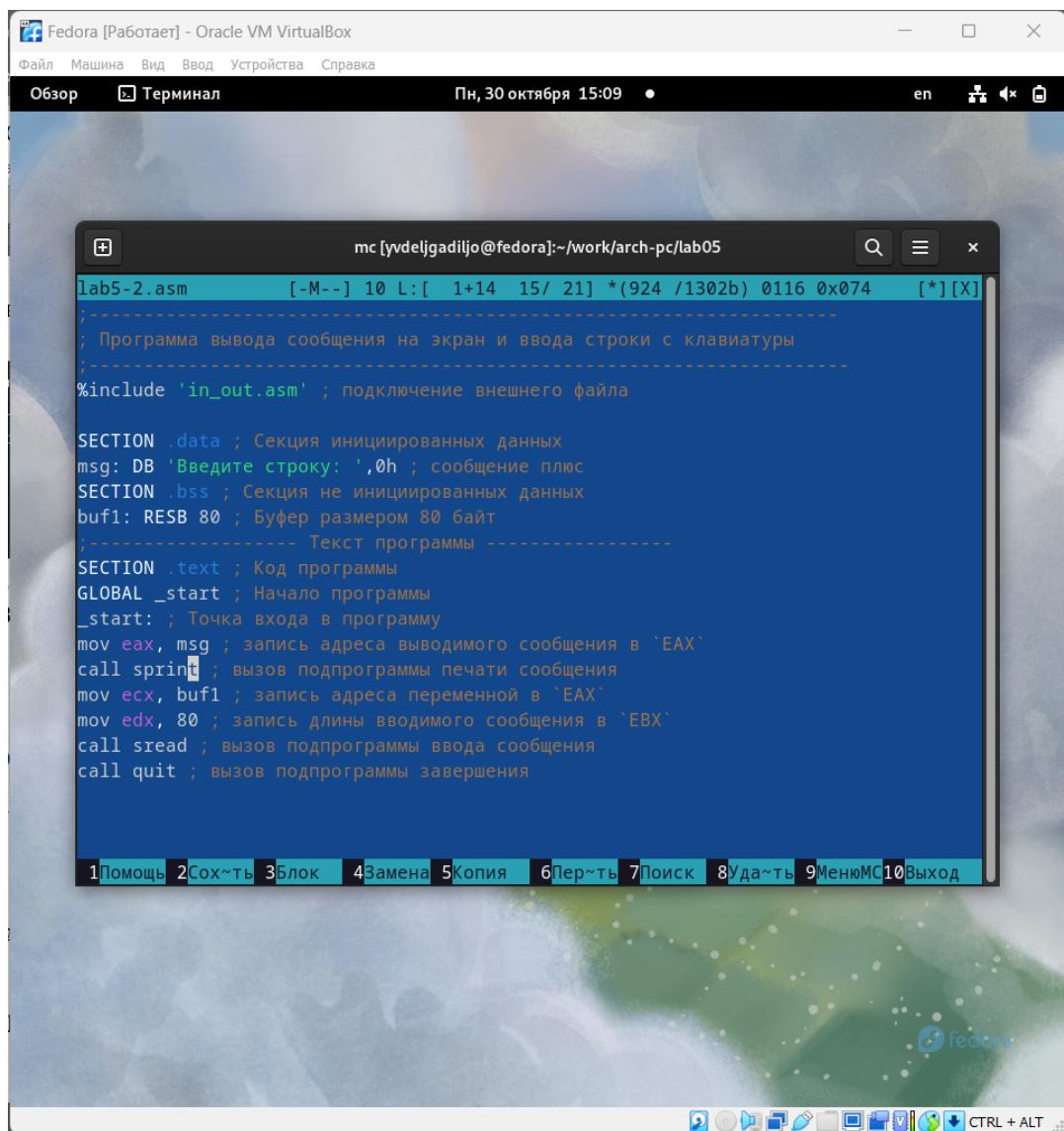


Рис. 3.13:

Создаю объектный файл lab5-2.o, выполняю компоновку объектного файла и запускаю исполняемый файл. Теперь ввод производится на той же строке, что и вывод, убран символ перевода строки после вывода.

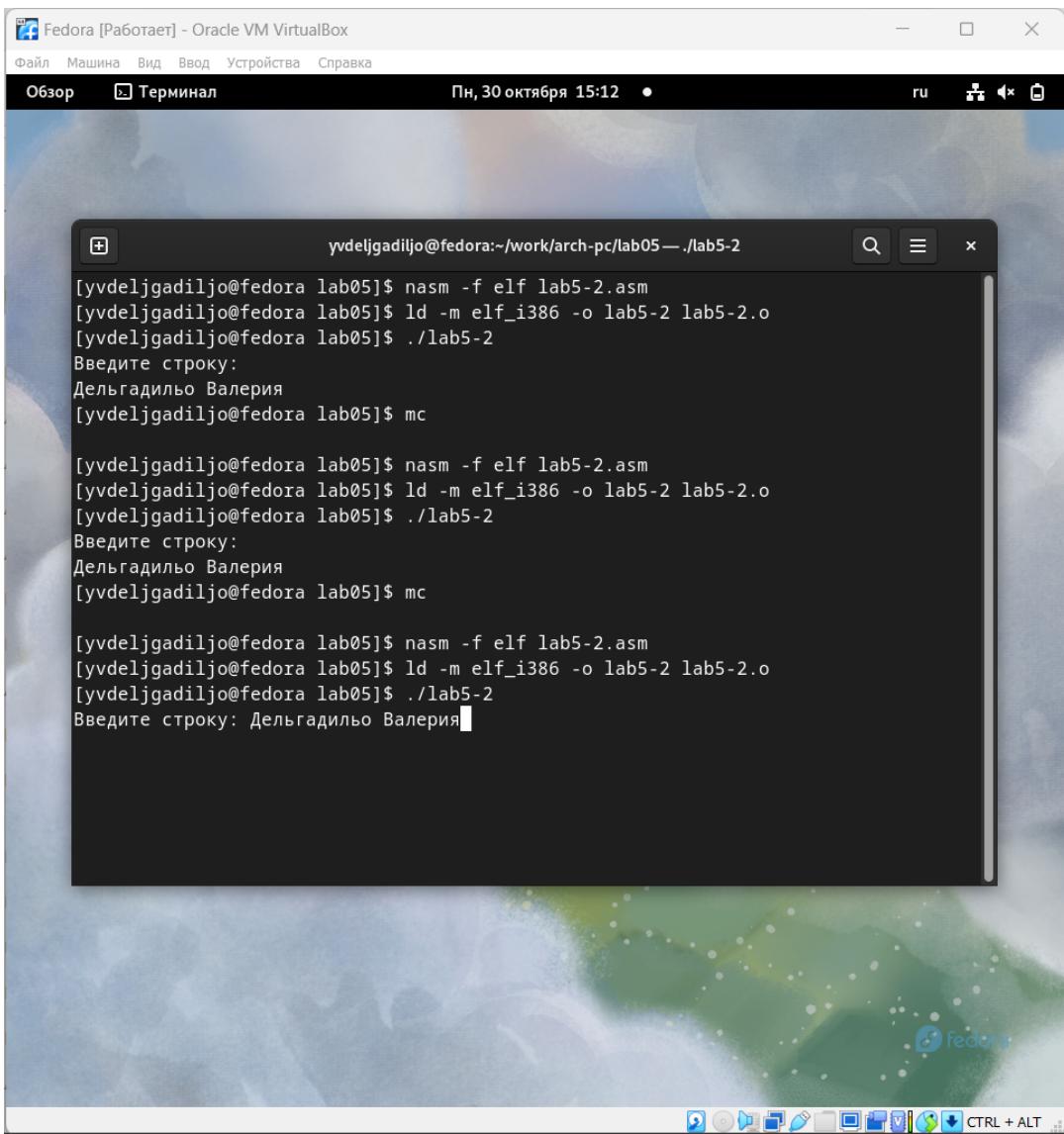


Рис. 3.14:

4 Задание для самостоятельной работы

4.1 Создайте копию файла lab5-1.asm.

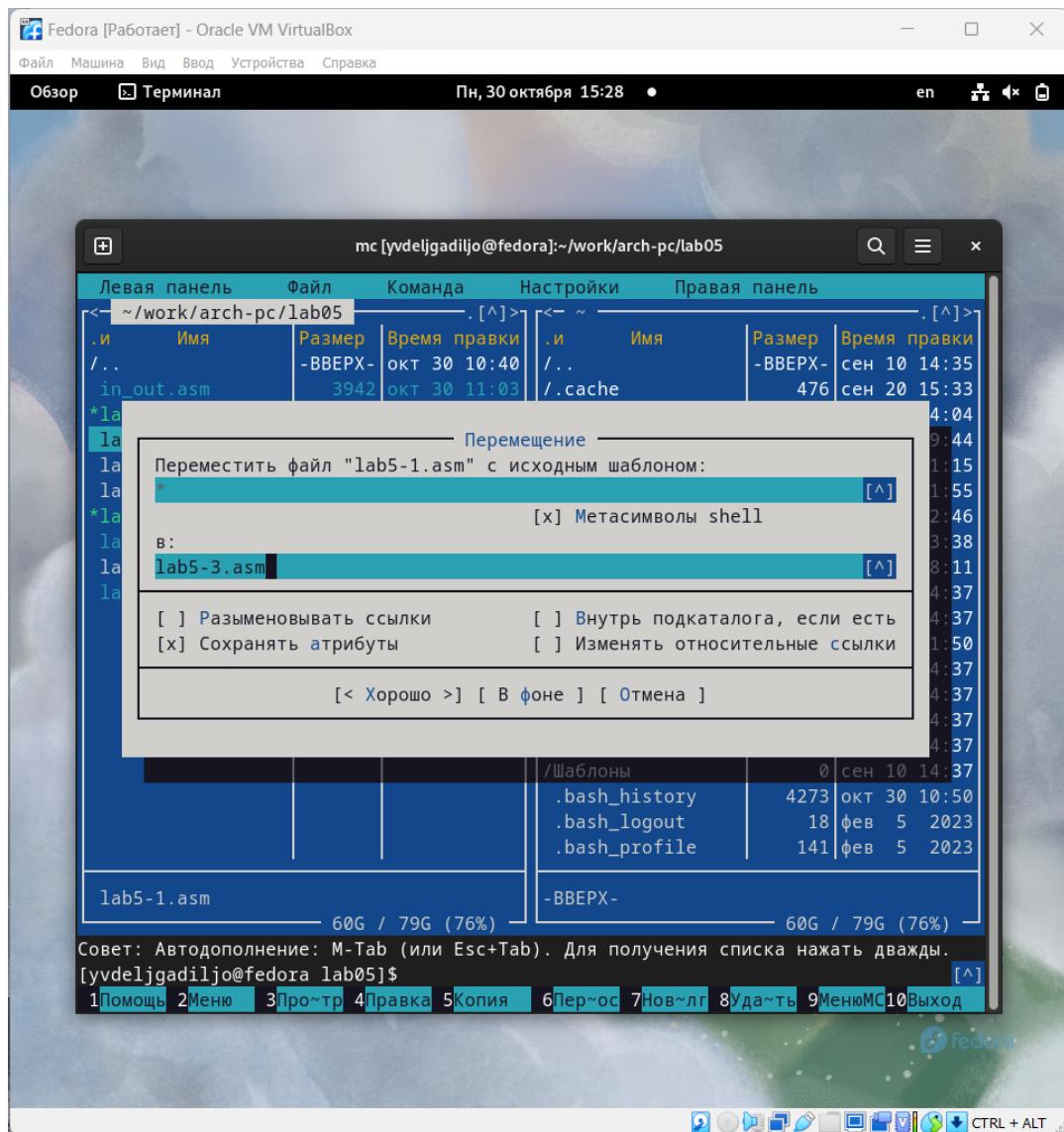


Рис. 4.1:

Изменяю код программы, добавляя вывод введенной строки.

The screenshot shows a terminal window titled 'mc [yvdeljgadiljo@fedora]:~/work/arch-pc/lab05'. The window contains assembly code for a program named 'lab5-3.asm'. The code includes comments explaining the purpose of various instructions, such as system calls for 'write', 'read', and 'exit'. The assembly instructions involve registers like eax, ebx, ecx, edx, and msgLen. The terminal window has a standard Linux-style interface with a menu bar at the top and a toolbar at the bottom.

```
Lab5-3.asm      [-M--] 12 L:[ 14+ 0 14/ 44] *(886 /2953b) 1095 0x447  [*][X]
_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 ; Дескриптор файла 0 - стандартный ввод
mov ecx, buf1 ; Адрес буфера под вводимую строку
mov edx, 80 ; Длина вводимой строки
int 80h ; Вызов ядра
----- 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,buf1 ; Адрес строки 'msg' в 'ecx'
mov edx,buf1 ; Размер строки 'msg' в 'edx'
int 80h ; Вызов ядра
----- Системный вызов 'exit' -----
; После вызова инструкции 'int 80h' программа завершит работу
```

Рис. 4.2:

Создаю объектный файл lab5-3.o, компоную его в исполняемый файл, запускаю исполняемый файл.

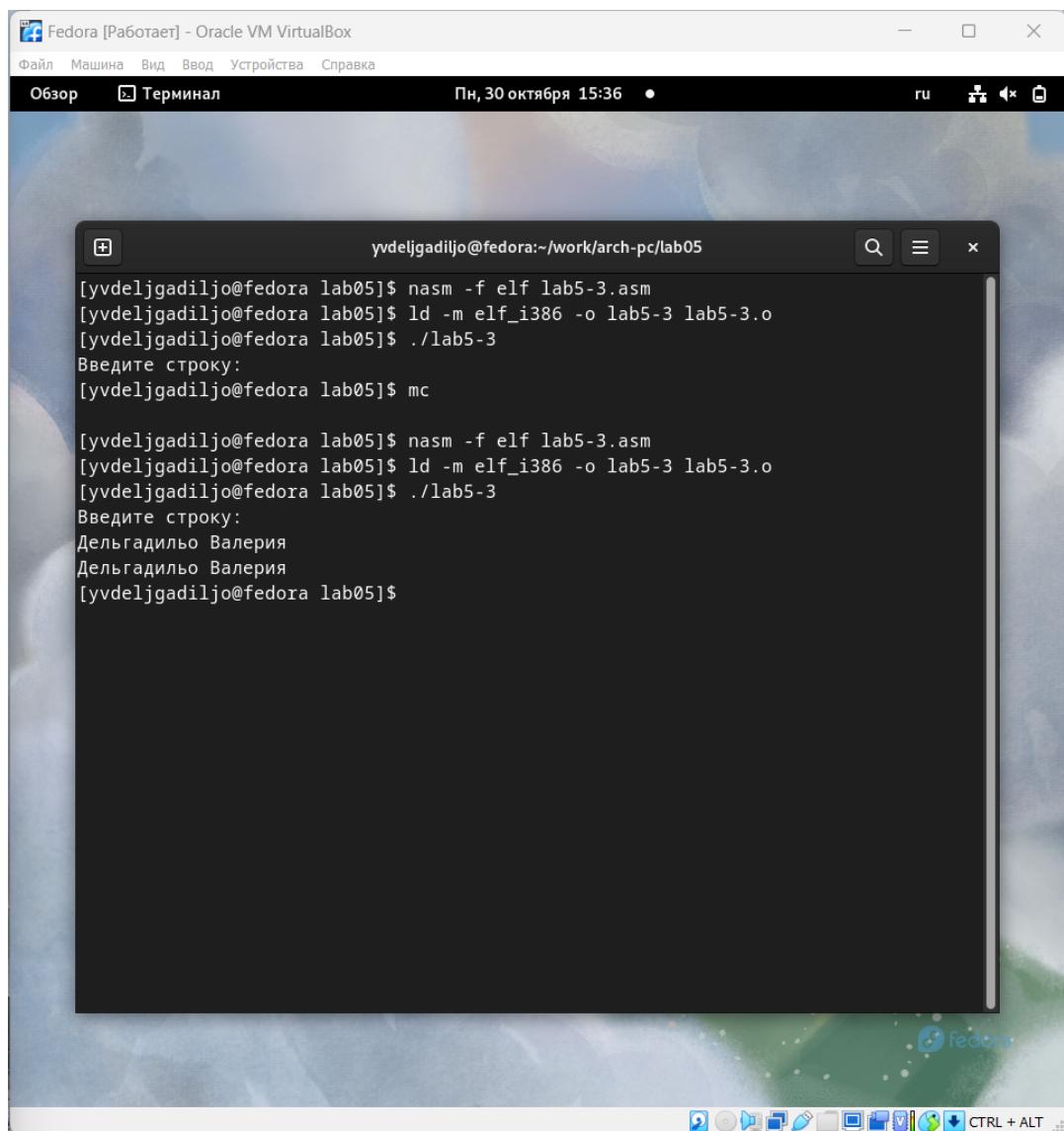


Рис. 4.3:

Программа из пункта 1:

```
;----- Объявление переменных -----
SECTION .data ; Секция инициализированных данных
msg: DB 'Введите строку:',10 ; сообщение плюс
; символ перевода строки
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
    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
    mov eax, 3 ; Системный вызов для чтения (sys_read)
    mov ebx, 0 ; Дескриптор файла 0 - стандартный ввод
    mov ecx, buf1 ; Адрес буфера под вводимую строку
    mov edx, 80 ; Длина вводимой строки
    int 80h ; Вызов ядра
    mov eax,4 ; Системный вызов для записи (sys_write)
    mov ebx,1 ; Описатель файла '1' - стандартный вывод
    mov ecx,buf1 ; Адрес строки buf1 в ecx
    mov edx,buf1 ; Размер строки buf1
    int 80h ; Вызов ядра
    mov eax,1 ; Системный вызов для выхода (sys_exit)
    mov ebx,0 ; Выход с кодом возврата 0 (без ошибок)
    int 80h ; Вызов ядра

```

4.2 Создайте копию файла lab5-2.asm.

Исправьте текст программы с использование подпрограмм из внешнего файла `in_out.asm`, так чтобы она работала по следующему алго-

ритму:

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

Копирую файл lab5-2.asm с именем lab5-4.asm

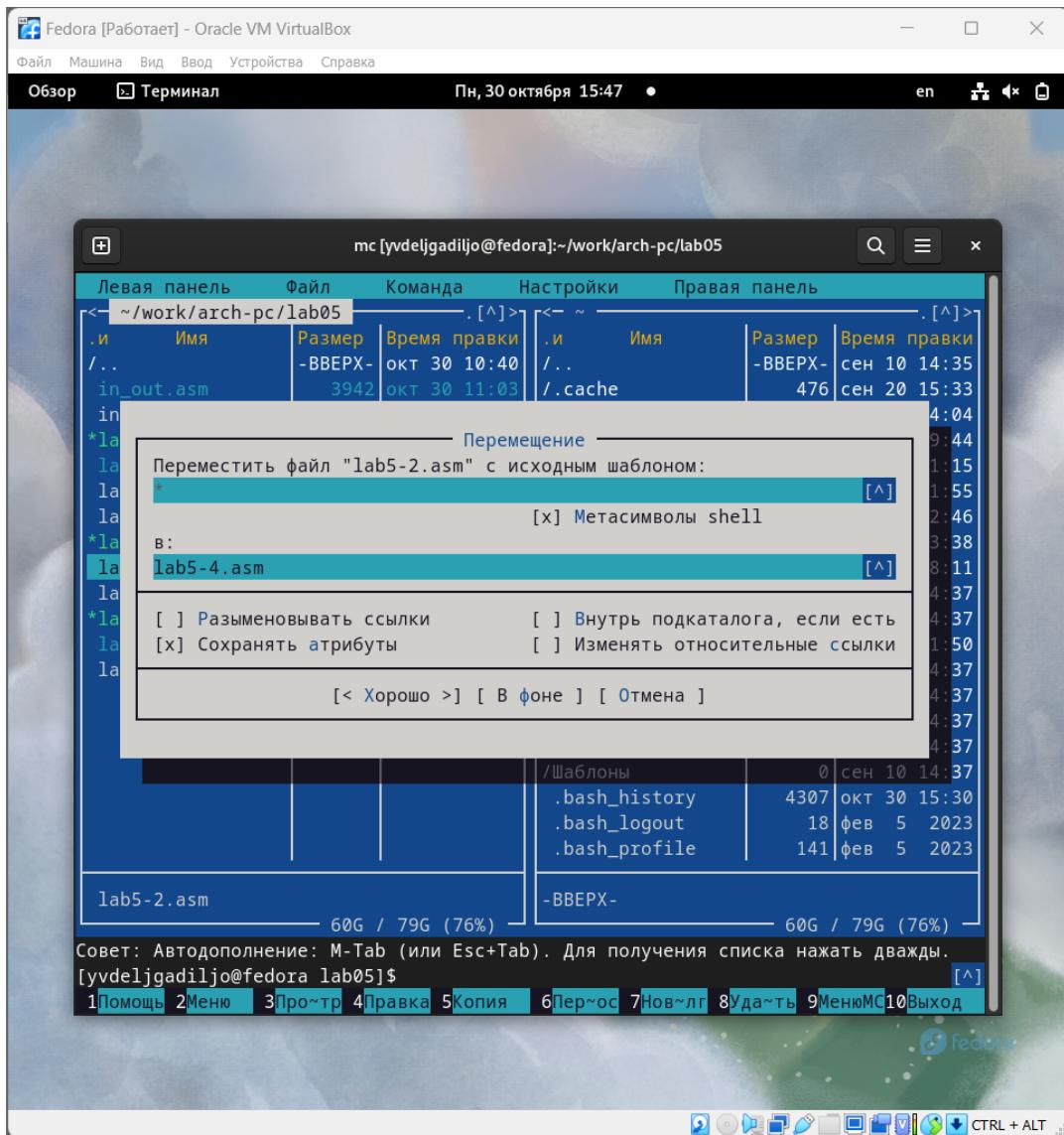
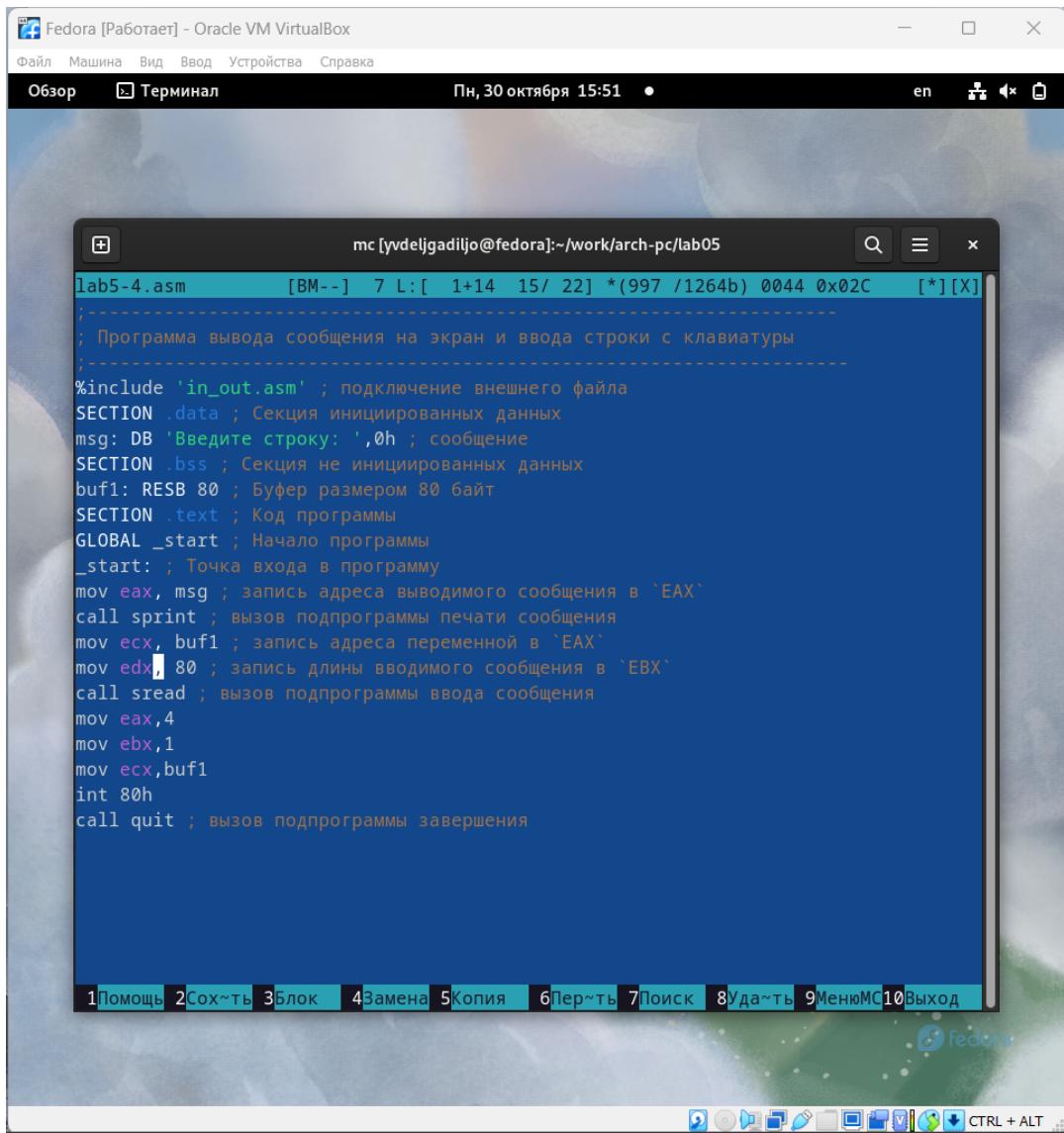


Рис. 4.4:

Изменяю код программы, добавляя вывод введенной строки.



The screenshot shows a Fedora desktop environment running in Oracle VM VirtualBox. The terminal window title is "mc [yvdel]gadiljo@fedora]:~/work/arch-pc/lab05". The code in the terminal is:

```
lab5-4.asm      [BM--] 7 L:[ 1+14 15/ 22] *(997 /1264b) 0044 0x02C  [*][X]
;-
; Программа вывода сообщения на экран и ввода строки с клавиатуры
;-
%include 'in_out.asm' ; подключение внешнего файла
SECTION .data ; Секция инициализированных данных
msg: DB 'Введите строку: ',0h ; сообщение
SECTION .bss ; Секция не инициализированных 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 edx, buf1 ; запись адреса переменной в `EBX`
    mov edx, 80 ; запись длины вводимого сообщения в `EBX`
    call sread ; вызов подпрограммы ввода сообщения
    mov eax,4
    mov ebx,1
    mov ecx,buf1
    int 80h
    call quit ; вызов подпрограммы завершения
```

Рис. 4.5:

Создаю объектный файл lab6-3.o, компоную его в исполняемый файл, запускаю исполняемый файл.

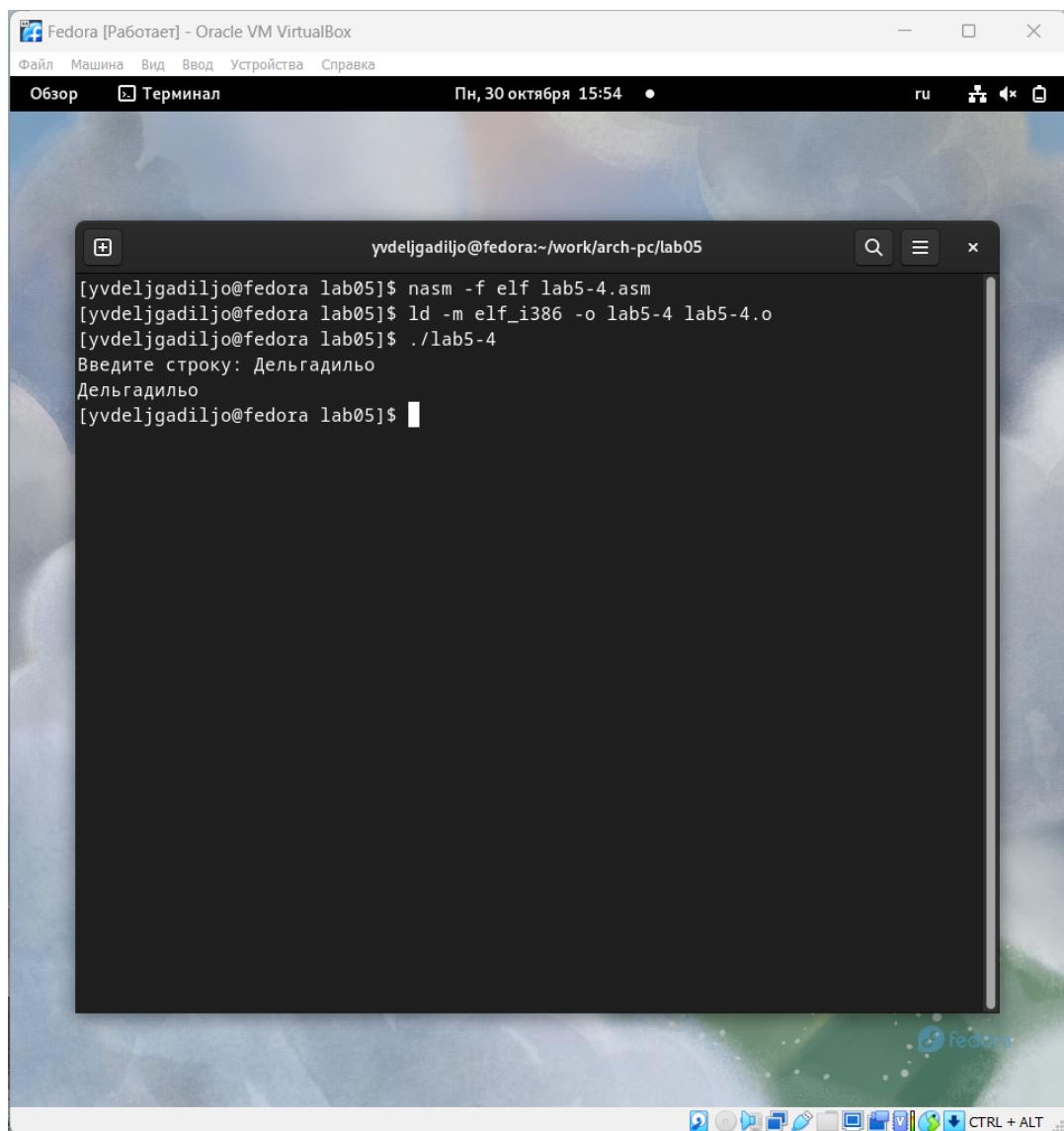


Рис. 4.6:

Программа из пункта 2:

```
;-----
; Программа вывода сообщения на экран и ввода строки с клавиатуры
;-----

%include 'in_out.asm' ; подключение внешнего файла
SECTION .data ; Секция инициализированных данных
```

```
msg: DB 'Введите строку: ',0h ; сообщение
SECTION .bss ; Секция не инициализированных 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,4 ; Системный вызов для записи (sys_write)
    mov ebx,1 ; Описатель файла '1' - стандартный вывод
    mov ecx,buf1 ; Адрес строки buf1 в ecx
    int 80h ; Вызов ядра
    call quit ; вызов подпрограммы завершения
```

5 Выводы

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

6 Список литературы

- GDB: The GNU Project Debugger. — URL: <https://www.gnu.org/software/gdb/>.
- GNU Bash Manual. — 2016. — URL: <https://www.gnu.org/software/bash/manual/>.
- Midnight Commander Development Center. — 2021. — URL: <https://midnight-commander.org/>.
- NASM Assembly Language Tutorials. — 2021. — URL: <https://asmtutor.com/>.
- Newham C. Learning the bash Shell: Unix Shell Programming. — O'Reilly Media, 2005. — 354 c. — (In a Nutshell). — ISBN 0596009658. — URL: <http://www.amazon.com/Learningbash-Shell-Programming-Nutshell/dp/0596009658>.
- Robbins A. Bash Pocket Reference. — O'Reilly Media, 2016. — 156 c. — ISBN 978-1491941591.
- The NASM documentation. — 2021. — URL: <https://www.nasm.us/docs.php>.
- Zarrelli G. Mastering Bash. — Packt Publishing, 2017. — 502 c. — ISBN 9781784396879.
- Колдаев В. Д., Лупин С. А. Архитектура ЭВМ. — М. : Форум, 2018.
- Куляс О. Л., Никитин К. А. Курс программирования на ASSEMBLER. — М. : Солон-Пресс, 2017.
- Новожилов О. П. Архитектура ЭВМ и систем. — М. : Юрайт, 2016.
- Расширенный ассемблер: NASM. — 2021. — URL: <https://www.opennet.ru/docs/RUS/nasm/>.

- Робачевский А., Немнюгин С., Стесик О. Операционная система UNIX. — 2-е изд. — БХВПетербург, 2010. — 656 с. — ISBN 978-5-94157-538-1.
- 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.
- Таненбаум Э. Архитектура компьютера. — 6-е изд. — СПб. : Питер, 2013. — 874 с. — (Классика Computer Science).
- Таненбаум Э., Бос Х. Современные операционные системы. — 4-е изд. — СПб. : Питер, 2015. — 1120 с. — (Классика Computer Science).