

Лабораторная работа №5

**Основы работы с Midnight Commander (mc). Структура программы
на языке ассемблера NASM. Системные вызовы в ОС GNU Linux**

Яковлуква Дарья Сергеевна

Содержание

1	Цель работы	5
2	Выполнение лабораторной работы	6
3	Выполнение задания для самостоятельной работы	20
4	Выводы	24

Список иллюстраций

2.1	Запуск Midnight commander	6
2.2	Интерфейс midnight commander	7
2.3	Переход в нужный каталог (~/.work/arch-pc)	8
2.4	Создание папки	9
2.5	Создание файла lab5-1.asm с помощью команды touch прямо в mc	10
2.6	Выбор текстового редактора	11
2.7	Редактирование файла lab5-1.asm	12
2.8	Проверка успешного редактирования	13
2.9	Компиляция файла с помощью nasm	14
2.10	Сборка исполняемого файла с помощью ld	15
2.11	Взаимодействие с программой	15
2.12	Открытие папки с файлом in_out.asm в правой панели	16
2.13	Копирование файла с помощью F6	16
2.14	Копирование файла с помощью F5	17
2.15	Текущий вид рабочей папки	17
2.16	Редактирование файла lab5-2.asm	18
2.17	Создание исполняемого файла	18
2.18	Запуск исполняемого файла	18
2.19	Изменение файла lab5-2.asm	19
2.20	Запуск изменённого файла	19
3.1	Создание копии файла lab5-1.asm	20
3.2	Изменение файла lab5-1-1.asm	21
3.3	Создание исполняемого файла	21
3.4	Проверка работы программы	21
3.5	Создание копии файла lab5-2.asm	22
3.6	Изменение файла lab5-2-1.asm	23
3.7	Создание исполняемого файла	23
3.8	Проверка работы программы	23

Список таблиц

1 Цель работы

Ознакомиться с программой Midnight commander и освоить написание программ на языке ассемблера с помощью инструкций `mov` и `int`

2 Выполнение лабораторной работы

Для начала выполнения лабораторной работы нам необходимо открыть Midnight commander с помощью команды mc (Рис. 2.1):

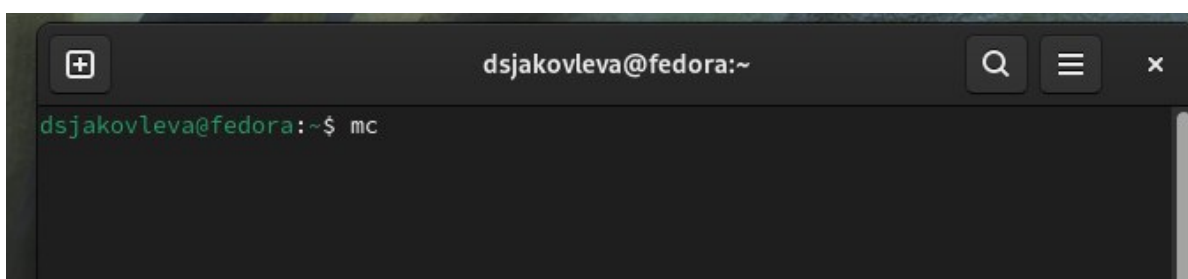


Рис. 2.1: Запуск Midnight commander

После ввода команды мы увидим такой интерфейс (Рис. 2.2):

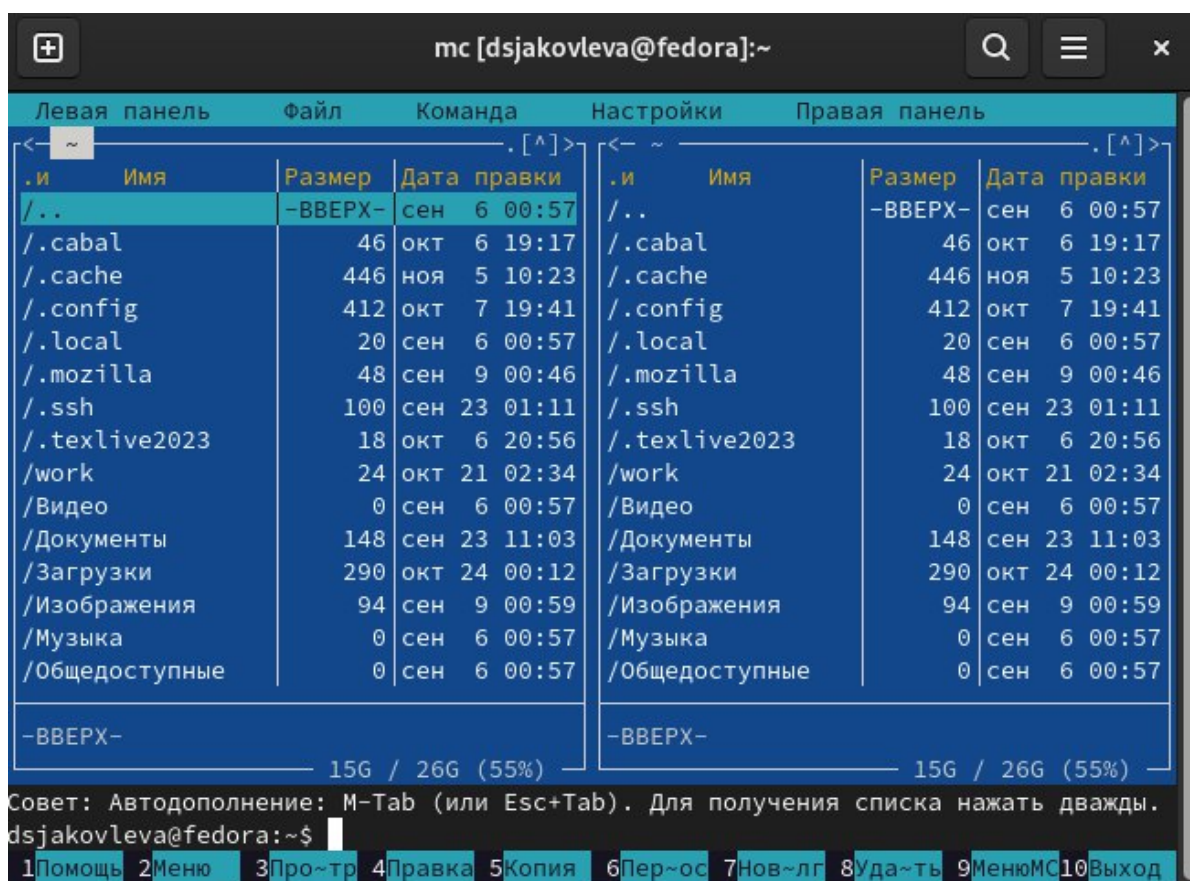


Рис. 2.2: Интерфейс midnight commander

С помощью стрелок и клавиши Enter перейдём в каталог ~/work/arch-pc (Рис. 2.3):

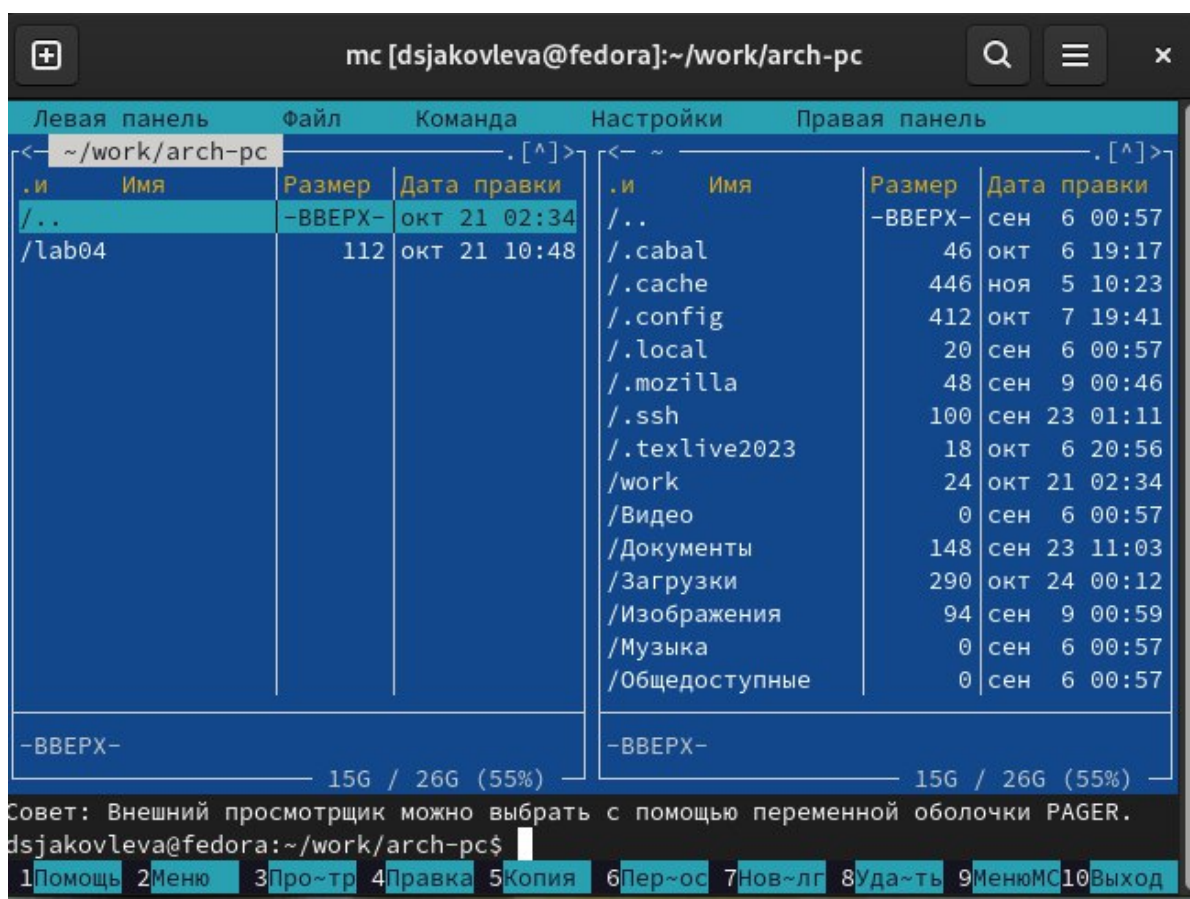


Рис. 2.3: Переход в нужный каталог (`~/work/arch-pc`)

Создадим папку `lab05` с помощью клавиши `F7` (Рис. 2.4):

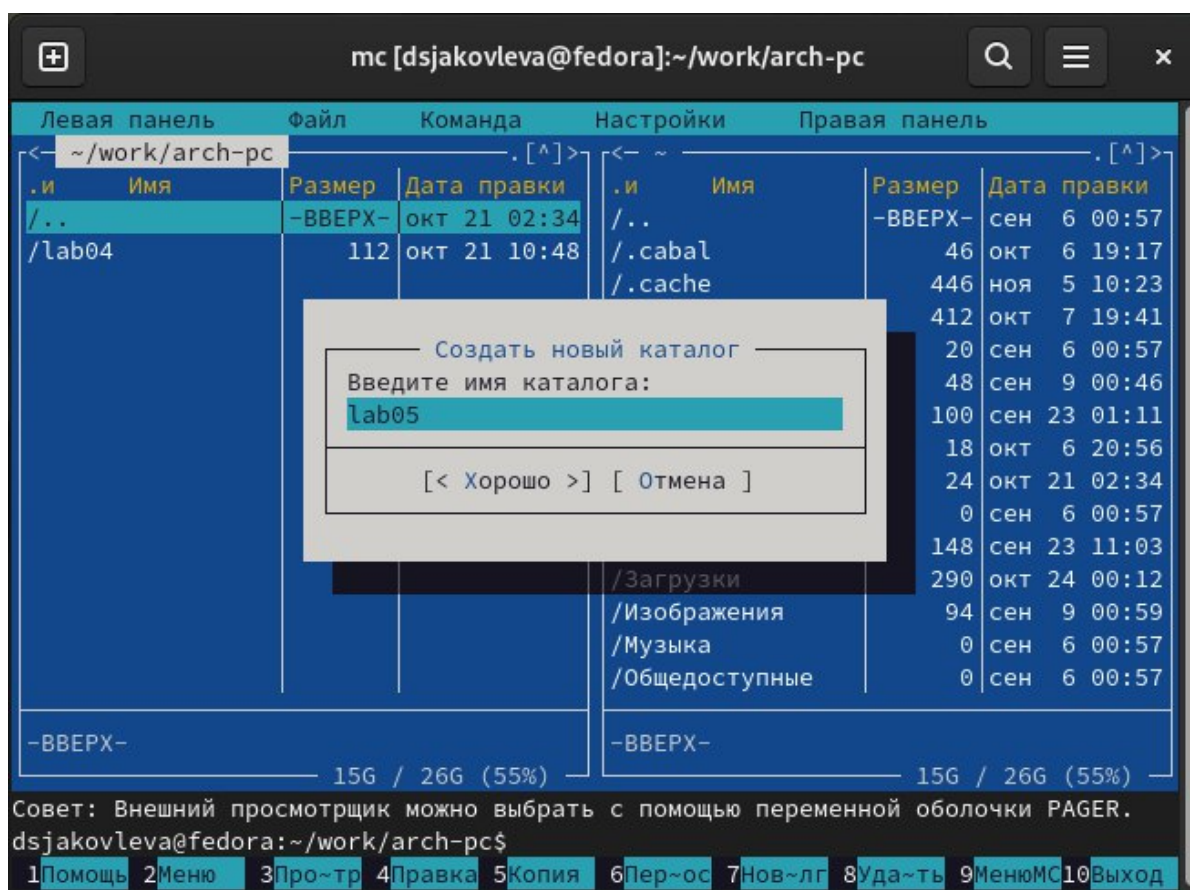


Рис. 2.4: Создание папки

Теперь с помощью команды `touch` создадим файл `lab5-1.asm` (Рис. 2.5):

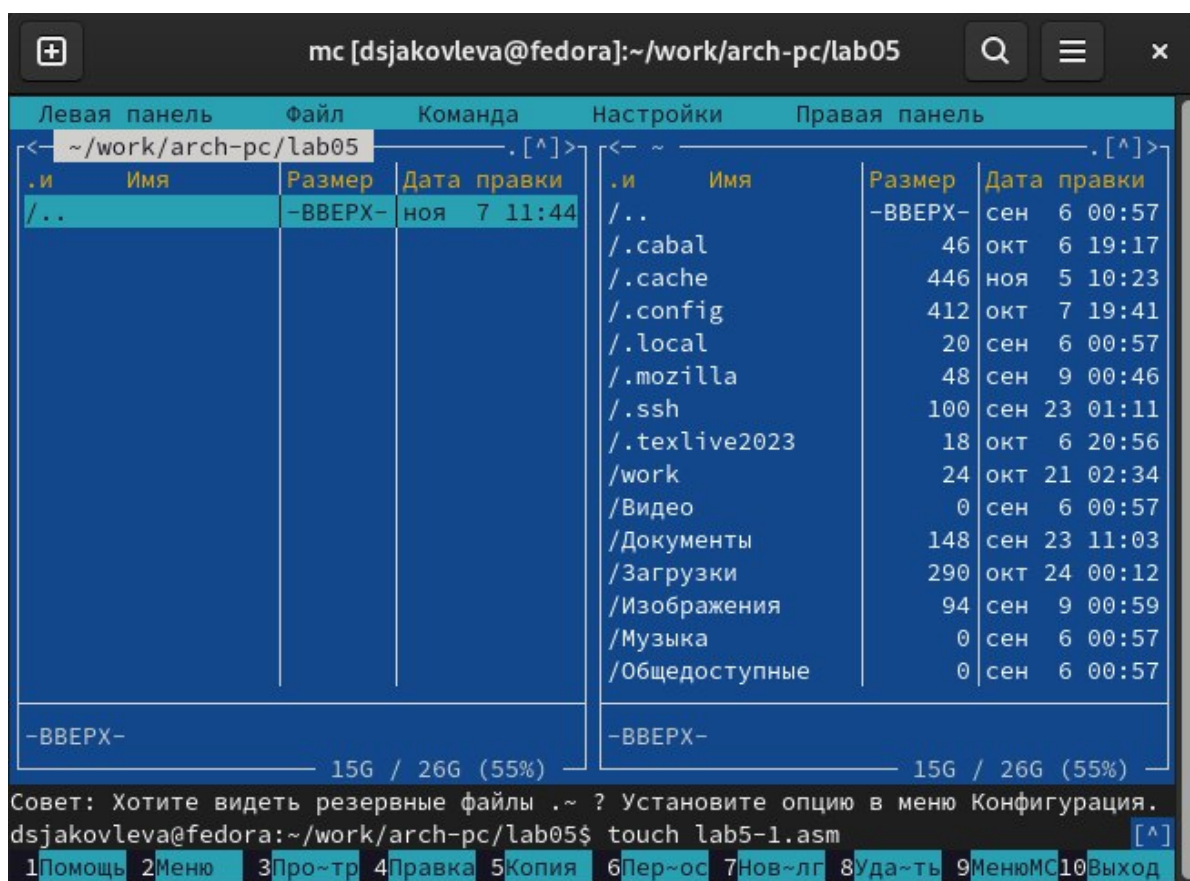


Рис. 2.5: Создание файла lab5-1.asm с помощью команды touch прямо в mc

Теперь откроем только что созданный файл с помощью редактор nano (Рис. 2.6):

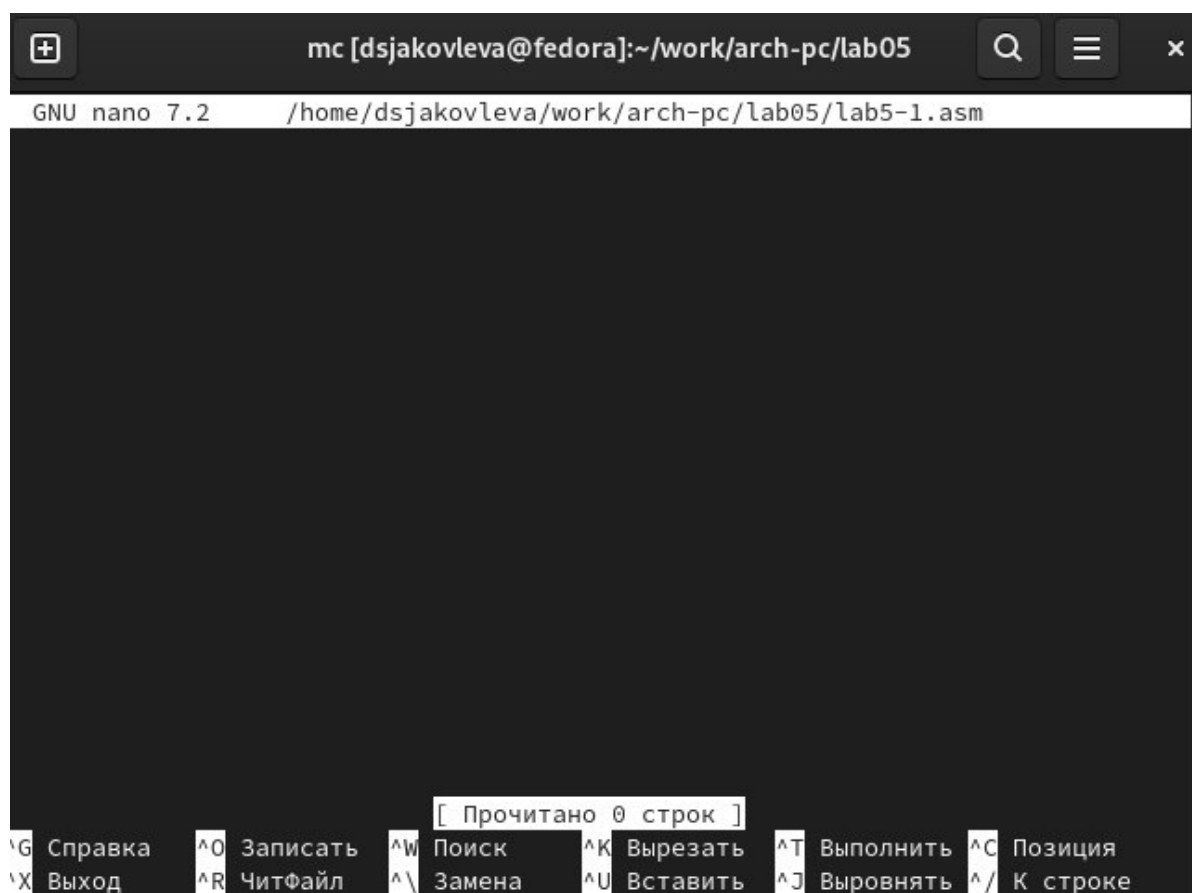


Рис. 2.6: Выбор текстового редактора

Теперь отредактируем файл и поместим в него следующий код (Рис. 2.7):

```
mc [dsjakovleva@fedora]:~/work/arch-pc/lab05
GNU nano 7.2 /home/dsjakovleva/work/arch-pc/lab05/lab5-1.asm  Изменён
;-----
; Программа вывода сообщения на экран и ввода строки с клавиатуры
;-----
;----- Объявление переменных -----
SECTION .data ; Секция иницированных данных
msg: DB 'Введите строку:',10 ; сообщение плюс
; символ перевода строки
msgLen: EQU $-msg ; Длина переменной 'msg'
SECTION .bss ; Секция не иницированных данных
buf1: RESB 80 ; Буфер размером 80 байт
52 Демидова А. В.
Архитектура ЭВМ
;----- 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)
```

^G Справка ^O Записать ^W Поиск ^K Вырезать ^T Выполнить ^C Позиция
^X Выход ^R ЧитФайл ^\ Замена ^U Вставить ^J Выровнять ^/ К строке

Рис. 2.7: Редактирование файла lab5-1.asm

Теперь сохраним его (сочетанием клавиш `ctrl+x` и согласившись с сохранением) и с помощью `F3` откроем для просмотра, чтобы убедиться, что он сохранился корректно (Рис. 2.8):

```
mc [dsjakovleva@fedora]:~/work/arch-pc/lab05
/home/dsjakovleva/work/~ch-pc/lab05/lab5-1.asm 1419/2491 56%
; Программа вывода сообщения на экран и ввода строки с клавиатуры
; ----- Объявление переменных -----
SECTION .data ; Секция инициированных данных
msg: DB 'Введите строку:',10 ; сообщение плюс
; символ перевода строки
msgLen: EQU $-msg ; Длина переменной 'msg'
SECTION .bss ; Секция не инициированных данных
buf1: RESB 80 ; Буфер размером 80 байт
52 Демидова А. В.
Архитектура ЭВМ
; ----- 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'
1Помощь 2Раз-рн 3Выход 4Нех 5Пер-ти 6 7Поиск 8Исх-ый 9Формат10Выход
```

Рис. 2.8: Проверка успешного редактирования

Теперь скомпилируем его (Рис. 2.9):

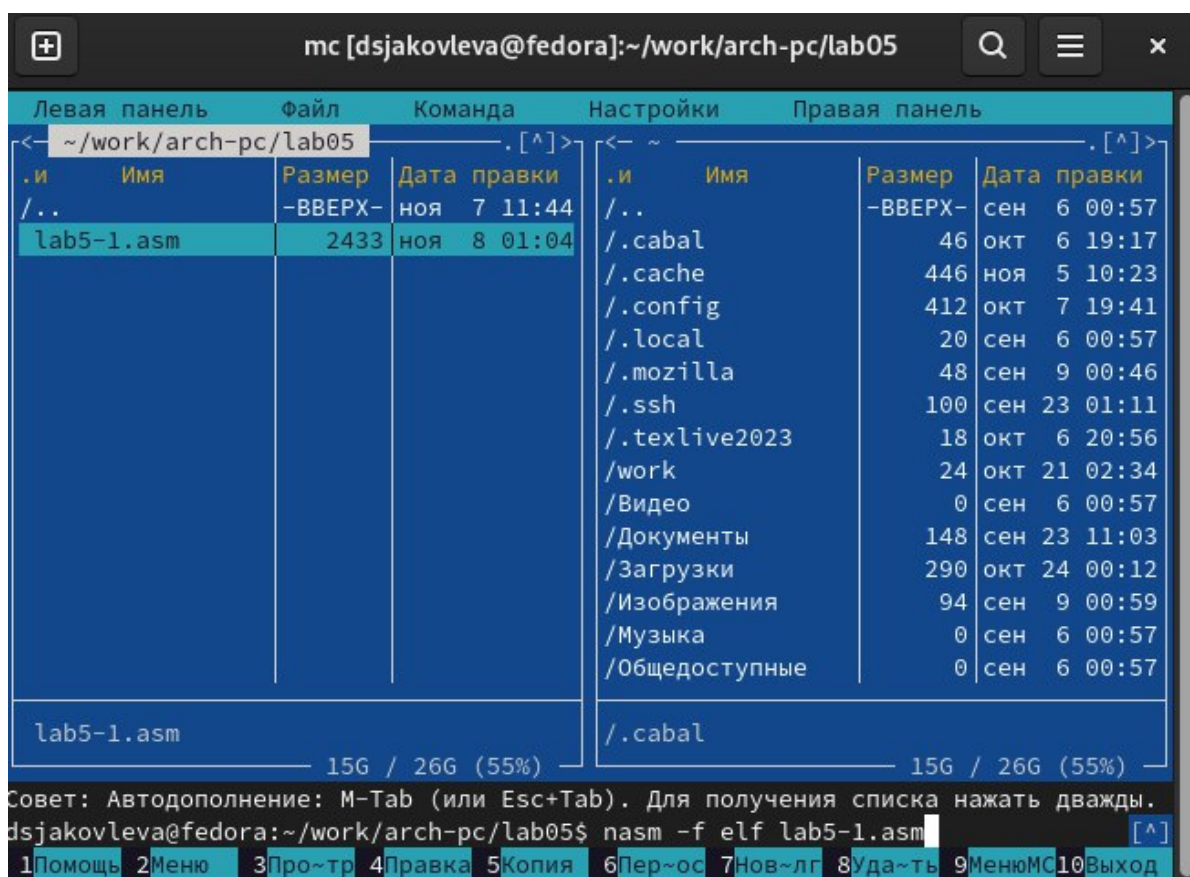


Рис. 2.9: Компиляция файла с помощью nasm

И соберём (Рис. 2.10):

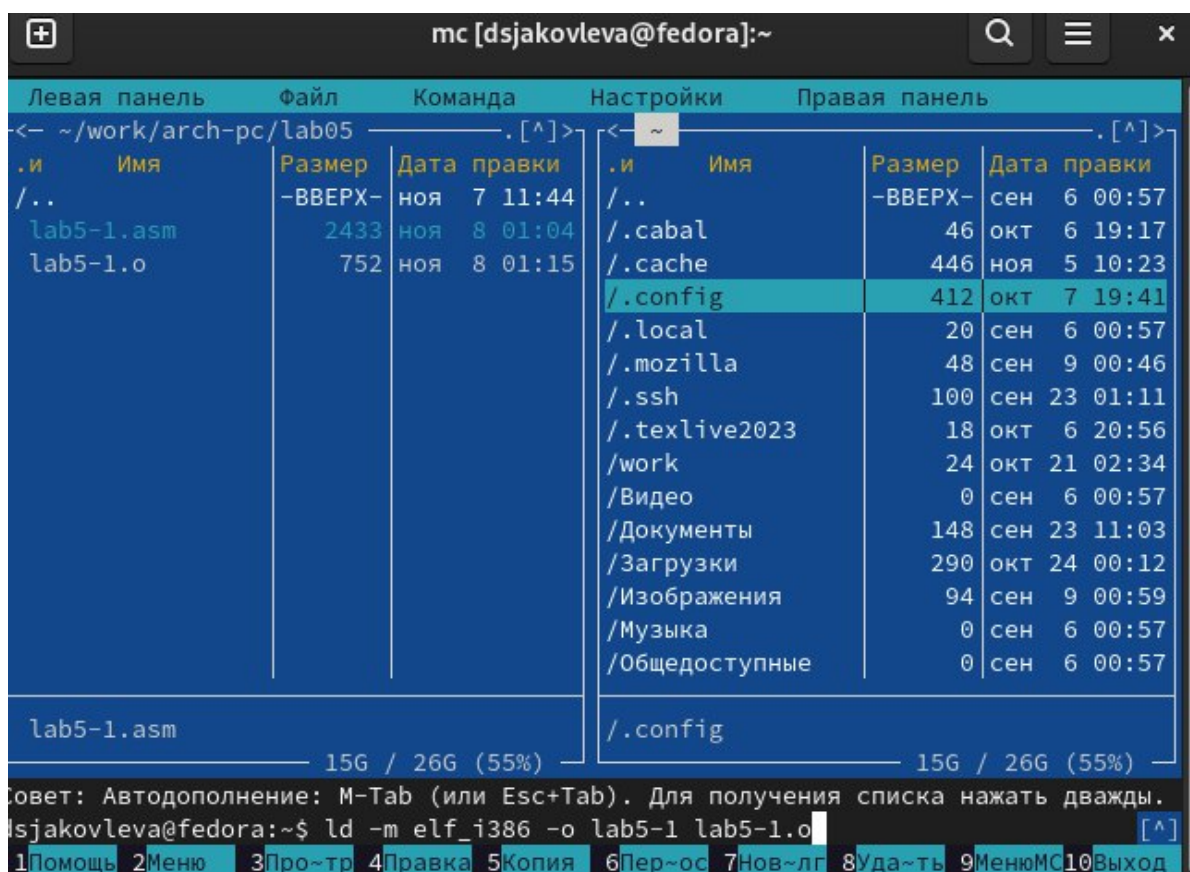


Рис. 2.10: Сборка исполняемого файла с помощью ld

После этого запустим получившийся исполняемый файл и введём ФИО (Рис. 2.11):

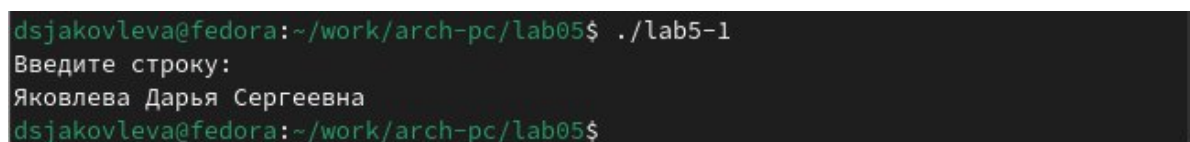


Рис. 2.11: Взаимодействие с программой

После нажатия Enter программа завершится и ничего не произойдёт. Теперь скачаем файл in_out.asm и откроем папку с ним в правой панели (Рис. 2.12):

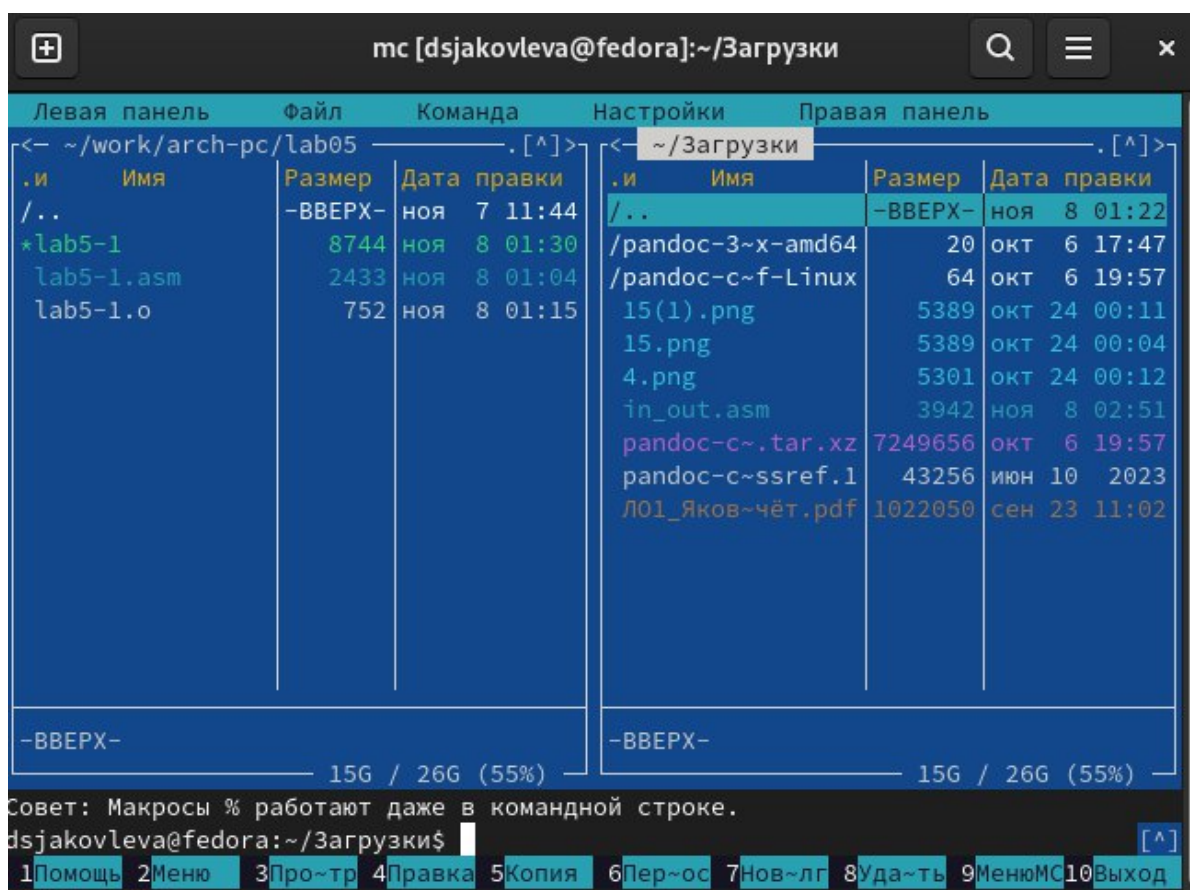


Рис. 2.12: Открытие папки с файлом in_out.asm в правой панели

Скопируем его в нашу рабочую папку с помощью F6 (Рис. 2.13):

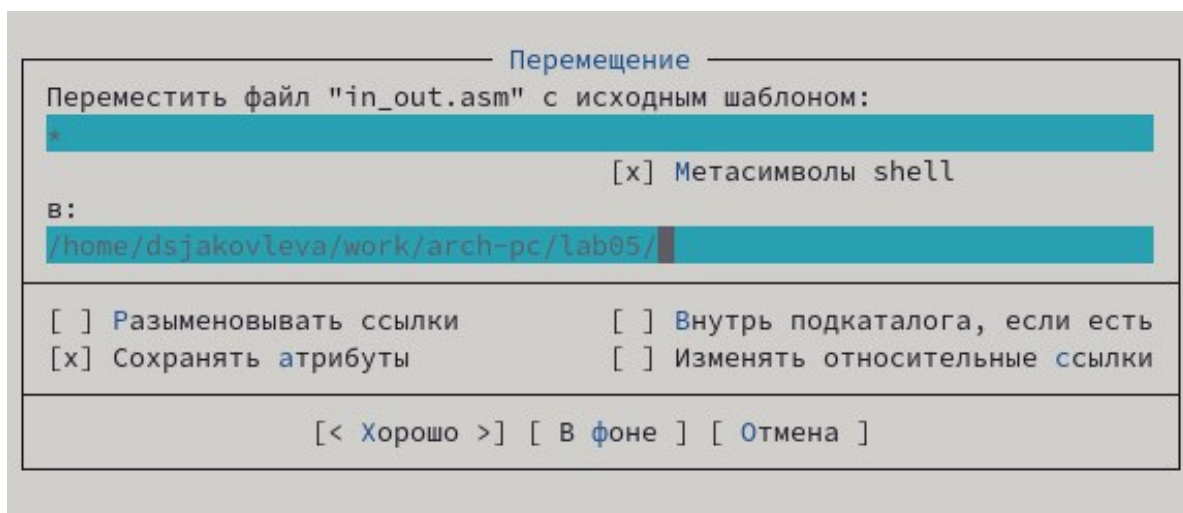


Рис. 2.13: Копирование файла с помощью F6

Теперь сделаем копию файла lab5-1.asm с помощью команды F5. Назовём копию lab5-2.asm (Рис. 2.14):

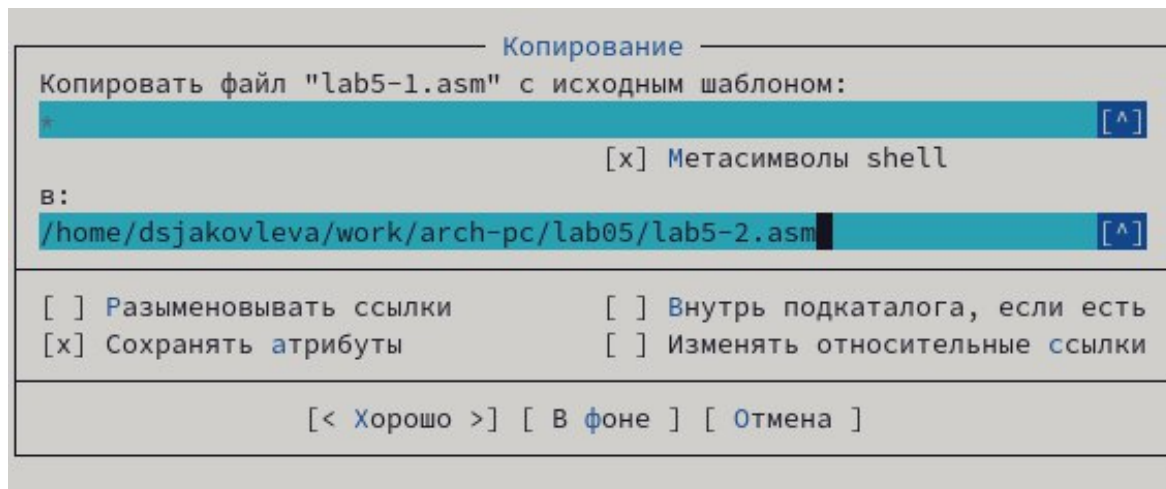


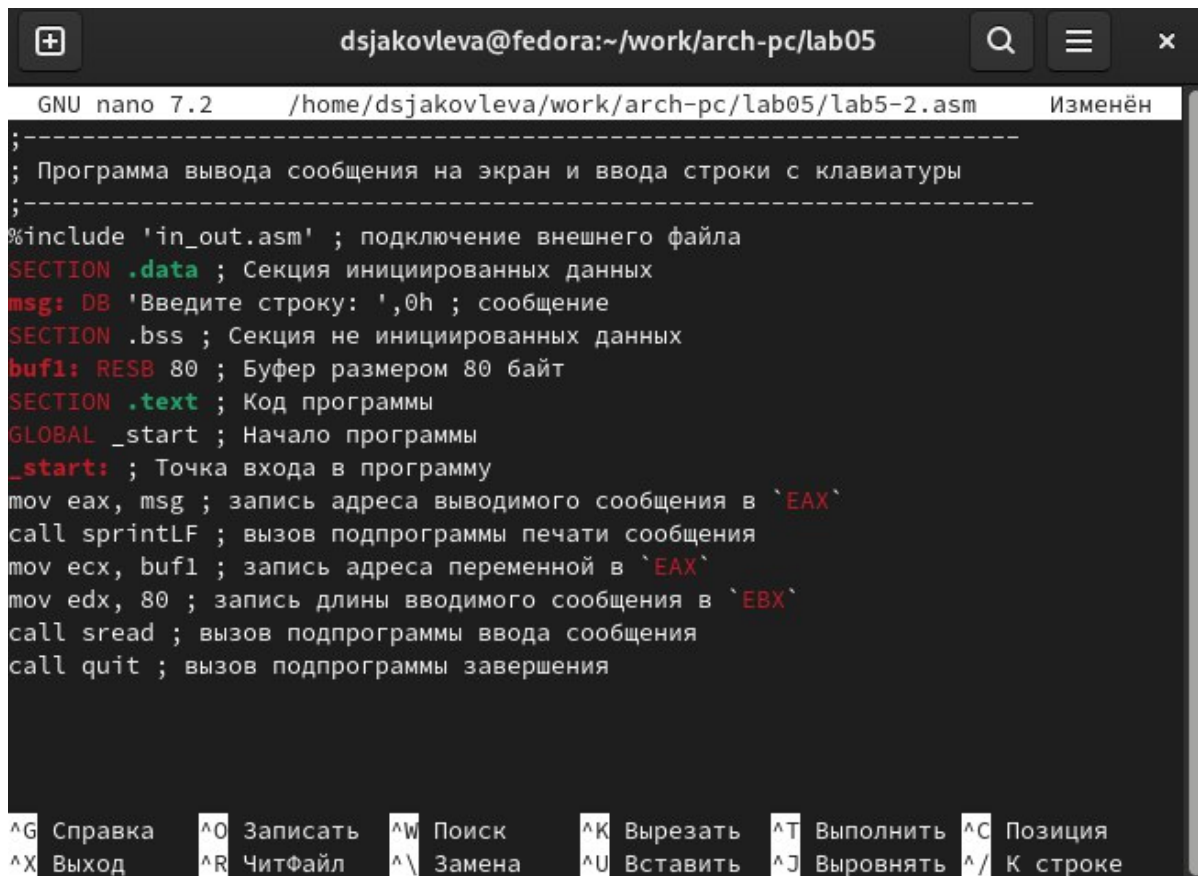
Рис. 2.14: Копирование файла с помощью F5

Теперь наша папка выглядит следующим образом (Рис. 2.15):

Левая панель		Файл	Команда
< ~/work/arch-pc/lab05			. [^]>
.и	Имя	Размер	Дата правки
/..		-ВВЕРХ-	ноя 7 11:44
in_out.asm		3942	ноя 8 02:51
*lab5-1		8744	ноя 8 01:30
lab5-1.asm		2433	ноя 8 01:04
lab5-1.o		752	ноя 8 01:15
lab5-2.asm		2433	ноя 8 01:04

Рис. 2.15: Текущий вид рабочей папки

Откроем в текстовом редакторе файл lab5-2.asm и напишем туда следующий код (Рис. 2.16):

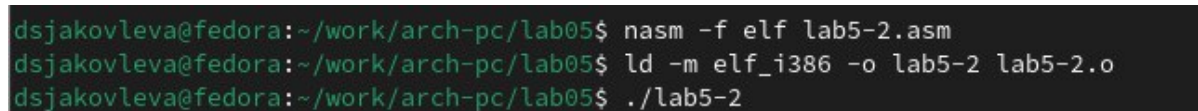


```
GNU nano 7.2 /home/dsjakovleva/work/arch-pc/lab05/lab5-2.asm Изменён
;-----
; Программа вывода сообщения на экран и ввода строки с клавиатуры
;-----
%include 'in_out.asm' ; подключение внешнего файла
SECTION .data ; Секция инициализированных данных
msg: DB 'Введите строку: ',0h ; сообщение
SECTION .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 sprintf ; 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
call quit ; вызов подпрограммы завершения

^G Справка  ^O Записать  ^W Поиск    ^K Вырезать  ^T Выполнить ^C Позиция
^X Выход    ^R ЧитФайл  ^\ Замена   ^U Вставить  ^J Выводить  ^_ К строке
```

Рис. 2.16: Редактирование файла lab5-2.asm

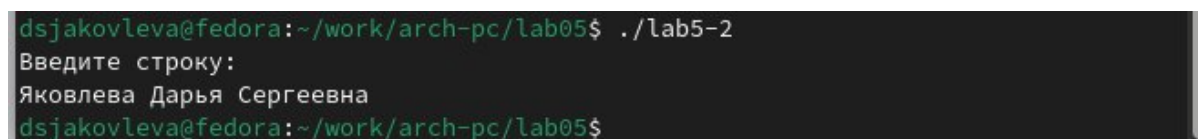
После чего создадим исполняемый файл с помощью nasm и ld (Рис. 2.17):



```
dsjakovleva@fedora:~/work/arch-pc/lab05$ nasm -f elf lab5-2.asm
dsjakovleva@fedora:~/work/arch-pc/lab05$ ld -m elf_i386 -o lab5-2 lab5-2.o
dsjakovleva@fedora:~/work/arch-pc/lab05$ ./lab5-2
```

Рис. 2.17: Создание исполняемого файла

Запустим созданный файл (Рис. 2.18):

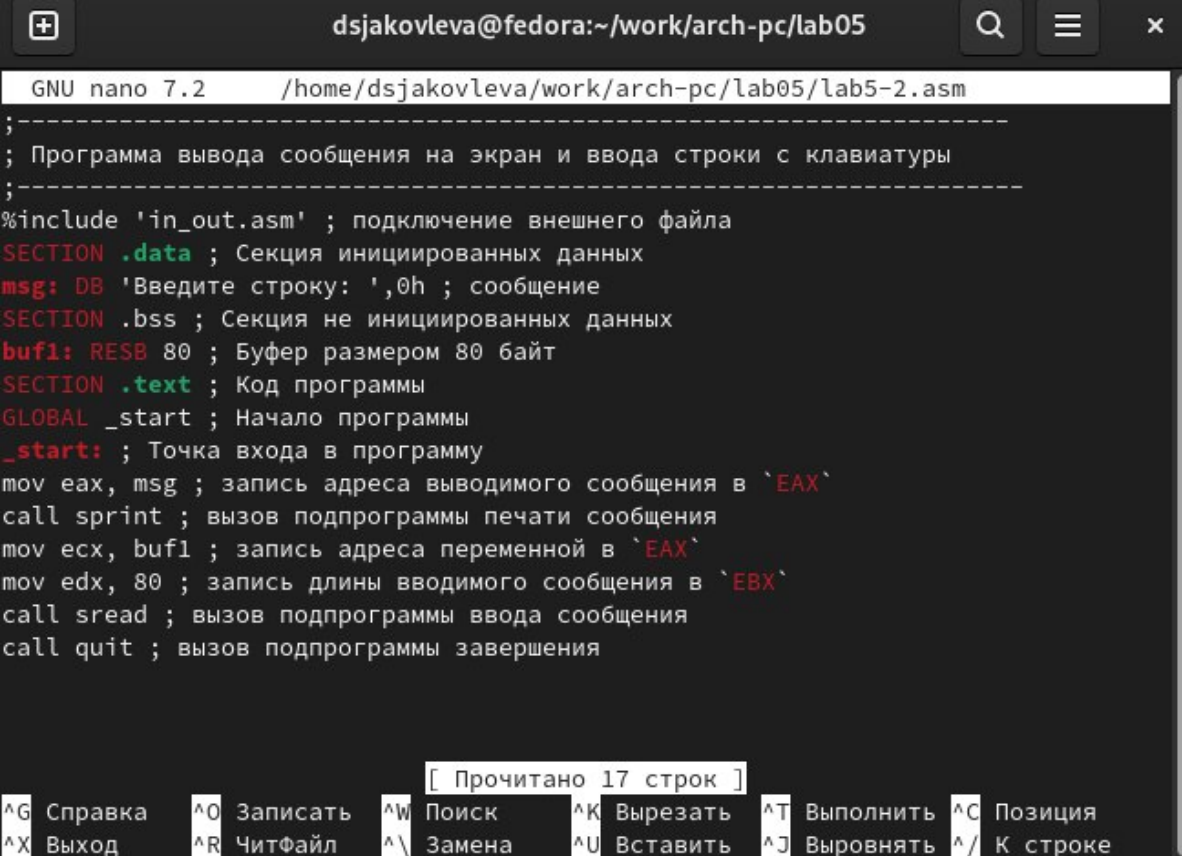


```
dsjakovleva@fedora:~/work/arch-pc/lab05$ ./lab5-2
Введите строку:
Яковлева Дарья Сергеевна
dsjakovleva@fedora:~/work/arch-pc/lab05$
```

Рис. 2.18: Запуск исполняемого файла

Он работает также, как и файл lab5-1, но использует для работы сторонний

файл. Попробуем теперь вместо команды `sprintLF` использовать просто команду `sprint` (Рис. 2.19):

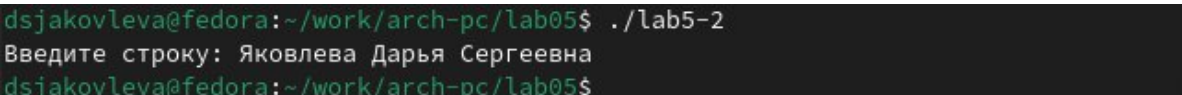


```
GNU nano 7.2 /home/dsjakovleva/work/arch-pc/lab05/lab5-2.asm
;-----
; Программа вывода сообщения на экран и ввода строки с клавиатуры
;-----
#include 'in_out.asm' ; подключение внешнего файла
SECTION .data ; Секция инициированных данных
msg: DB 'Введите строку: ',0h ; сообщение
SECTION .bss ; Секция не инициированных 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
call quit ; вызов подпрограммы завершения

[ Прочитано 17 строк ]
^G Справка      ^O Записать     ^W Поиск        ^K Вырезать     ^T Выполнить    ^C Позиция
^X Выход        ^R ЧитФайл     ^\ Замена       ^U Вставить     ^J Вывернуть    ^/_ К строке
```

Рис. 2.19: Изменение файла lab5-2.asm

Точно также соберём исполняемый файл и запустим его (Рис. 2.20):



```
dsjakovleva@fedora:~/work/arch-pc/lab05$ ./lab5-2
Введите строку: Яковлева Дарья Сергеевна
dsjakovleva@fedora:~/work/arch-pc/lab05$
```

Рис. 2.20: Запуск изменённого файла

Как мы видим, теперь нет переноса на следующую строку. Этим и отличаются команды `sprintLF` от `sprint`. Первая добавляет перенос после текста, а вторая нет

3 Выполнение задания для самостоятельной работы

Теперь создадим с помощью F5 копию файла lab5-1.asm (Рис. 3.1):

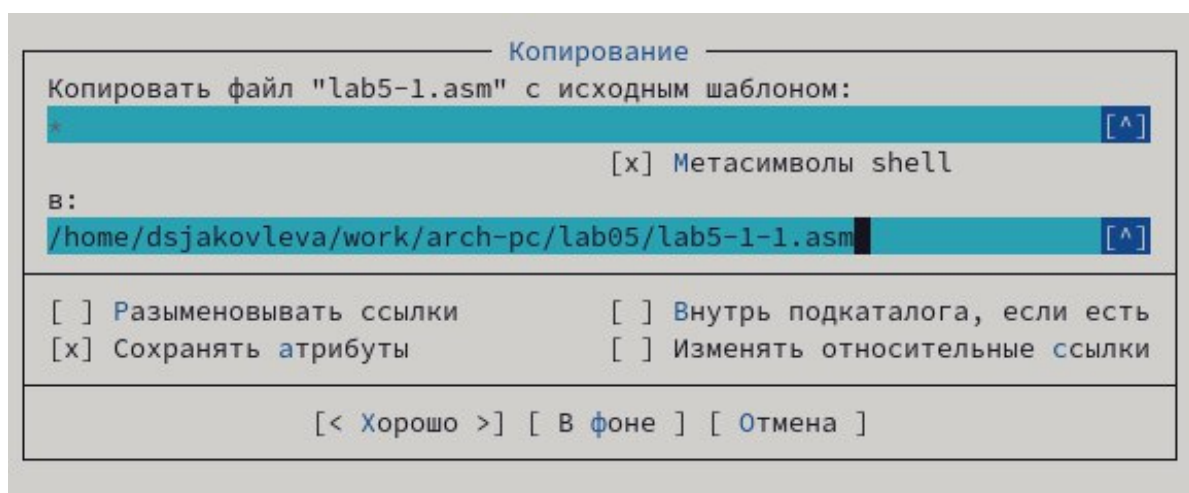
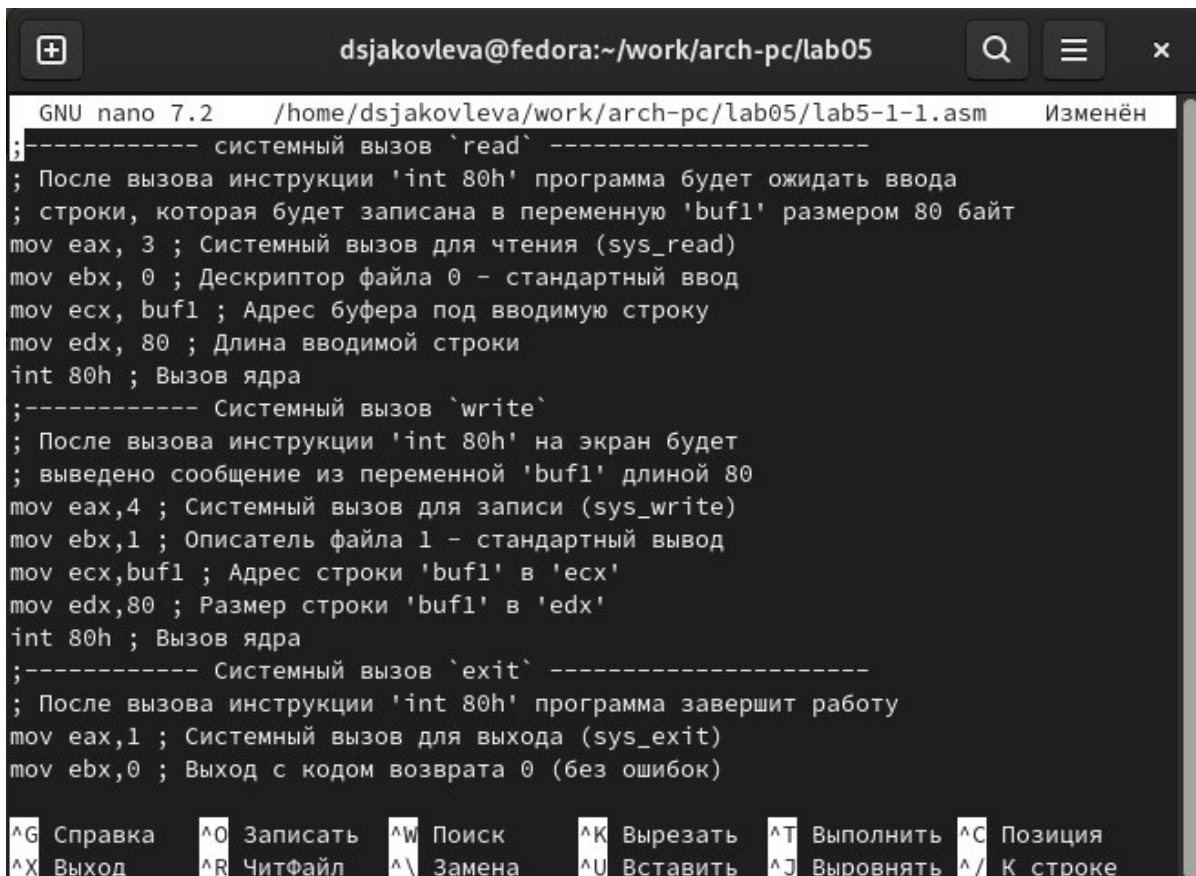


Рис. 3.1: Создание копии файла lab5-1.asm

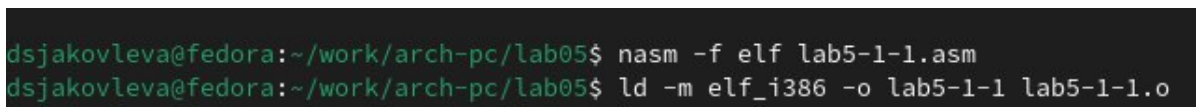
Изменим копию так, чтобы она выводила тот текст, который получила на ввод. Для этого перед системным вызовом `exit` вставим текст с системным вызовом `write`. Он очень похож на системный вызов `write`, который уже был в коде, но есть несколько отличий. Так, мы перемещаем адрес строки `buf1` в `ecx` и размер строки `buf1` (80) в `edx` (Рис. 3.2):



```
dsjakovleva@fedora:~/work/arch-pc/lab05
GNU nano 7.2 /home/dsjakovleva/work/arch-pc/lab05/lab5-1-1.asm  Изменён
;----- 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 'buf1' длиной 80
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, 80 ; Размер строки 'buf1' в 'edx'
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)
```

Рис. 3.2: Изменение файла lab5-1-1.asm

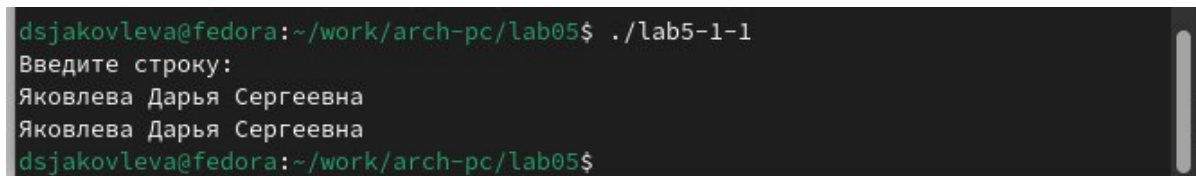
Сохраним изменения и создадим исполняемый файл (Рис. 3.3):



```
dsjakovleva@fedora:~/work/arch-pc/lab05$ nasm -f elf lab5-1-1.asm
dsjakovleva@fedora:~/work/arch-pc/lab05$ ld -m elf_i386 -o lab5-1-1 lab5-1-1.o
```

Рис. 3.3: Создание исполняемого файла

Запустим его и проверим, что всё работает (Рис. 3.4):



```
dsjakovleva@fedora:~/work/arch-pc/lab05$ ./lab5-1-1
Введите строку:
Яковлева Дарья Сергеевна
Яковлева Дарья Сергеевна
dsjakovleva@fedora:~/work/arch-pc/lab05$
```

Рис. 3.4: Проверка работы программы

Теперь создадим с помощью F5 копию файла lab5-2.asm (Рис. 3.5):

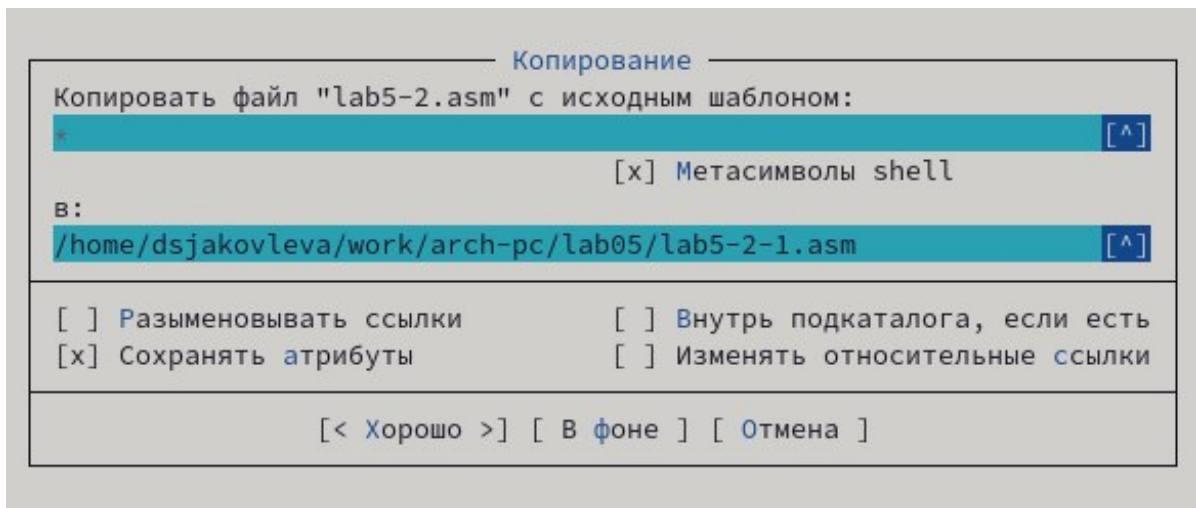
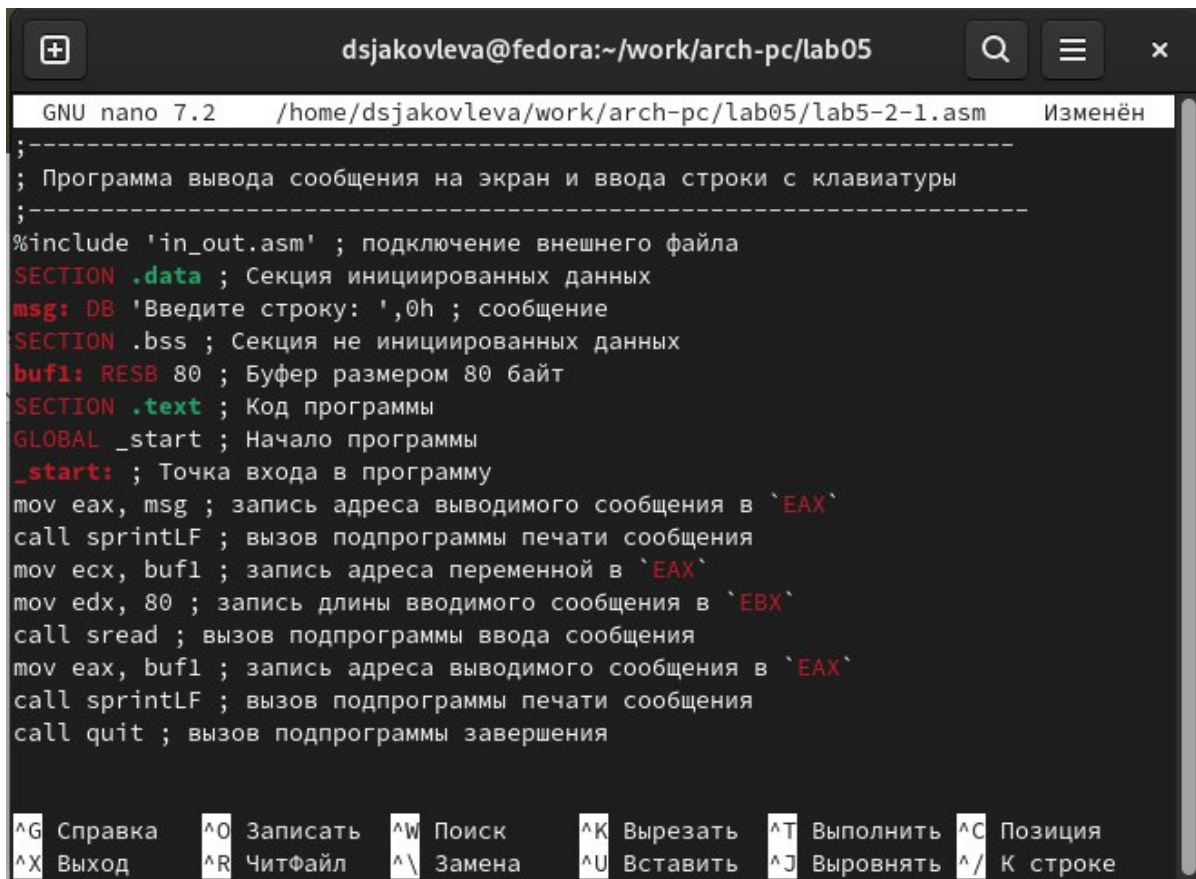


Рис. 3.5: Создание копии файла lab5-2.asm

теперь сделаем так, чтобы этот код также выводил тот текст, что получит на ввод. Для этого перед последней строкой добавим строчку, которая записывает в еах адрес buf1, а также строчку, которая вызывает подпрограмму sprintLF (Рис. 3.6):

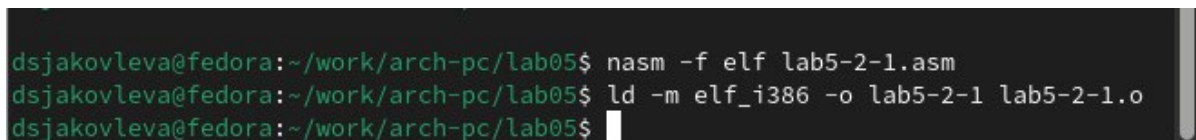


```
GNU nano 7.2 /home/dsjakovleva/work/arch-pc/lab05/lab5-2-1.asm Изменён
;-----
; Программа вывода сообщения на экран и ввода строки с клавиатуры
;-----
%include 'in_out.asm' ; подключение внешнего файла
SECTION .data ; Секция инициированных данных
msg: DB 'Введите строку: ',0h ; сообщение
SECTION .bss ; Секция не инициированных 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 sprintLF ; 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 ; запись адреса выводимого сообщения в `EAX`
call sprintLF ; вызов подпрограммы печати сообщения
call quit ; вызов подпрограммы завершения

^G Справка      ^O Записать     ^W Поиск        ^K Вырезать     ^T Выполнить    ^C Позиция
^X Выход         ^R ЧитФайл     ^\ Замена       ^U Вставить     ^J Выводить     ^/_ К строке
```

Рис. 3.6: Изменение файла lab5-2-1.asm

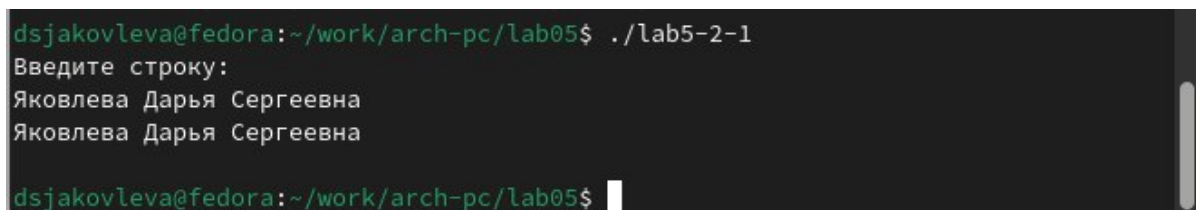
Теперь создадим исполняемый файл (Рис. 3.7):



```
dsjakovleva@fedora:~/work/arch-pc/lab05$ nasm -f elf lab5-2-1.asm
dsjakovleva@fedora:~/work/arch-pc/lab05$ ld -m elf_i386 -o lab5-2-1 lab5-2-1.o
dsjakovleva@fedora:~/work/arch-pc/lab05$
```

Рис. 3.7: Создание исполняемого файла

Теперь запустим программу и убедимся, что она работает (Рис. 3.8):



```
dsjakovleva@fedora:~/work/arch-pc/lab05$ ./lab5-2-1
Введите строку:
Яковлева Дарья Сергеевна
Яковлева Дарья Сергеевна
dsjakovleva@fedora:~/work/arch-pc/lab05$
```

Рис. 3.8: Проверка работы программы

4 Выводы

В результате выполнения работы были получены навыки работы с Midnight commander, а также навыки написания простых программ ввода-вывода на языке ассемблера