

Отчёт по лабораторной работе № 5

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

Студент: Святасова Ксения Евгеньевна

Содержание

1	Цель работы	4
2	Теоритическое введение	5
3	Выполнение лабораторной работы	6
3.1	Midnight Commander	6
3.2	Подключение внешнего файла in_out.asm	15
4	Задания для самостоятельной работы	22
5	Вывод	27

Список иллюстраций

3.1	Команда <code>mc</code>	6
3.2	Каталог <code>arch-ps</code>	7
3.3	Создание папки	8
3.4	Папка	9
3.5	Создание файла <code>lab5-1.asm</code>	9
3.6	Файл <code>lab5-1.asm</code>	10
3.7	Ввод текста из листинга	11
3.8	Сохранения файла с текстом из листинга	12
3.9	Проверка файла	14
3.10	Программа	14
3.11	Каталог с файлом <code>in_out.asm</code>	16
3.12	Копирование файла	17
3.13	Копия файла	18
3.14	Исправление текста	19
3.15	Проверка работы программы	19
3.16	Замена <code>sprintLF</code> на <code>sprint</code>	21
3.17	Проверка программы	21
4.1	Копирование файла	23
4.2	Исправленная программа	24
4.3	Программа	24
4.4	Копирование файла	25
4.5	Исправленная программа	26
4.6	Программа	26

1 Цель работы

Целью работы является приобретение практических навыков работы в *Midnight Commander* и освоение языка ассемблера `mov` и `int`.

2 Теоритическое введение

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.

Программа на языке ассемблера NASM, как правило, состоит из трёх секций: секция кода программы (SECTION .text), секция инициированных (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 Выполнение лабораторной работы

3.1 Midnight Commander

1. Откроем *Midnight Commander* (рис. 3.1):

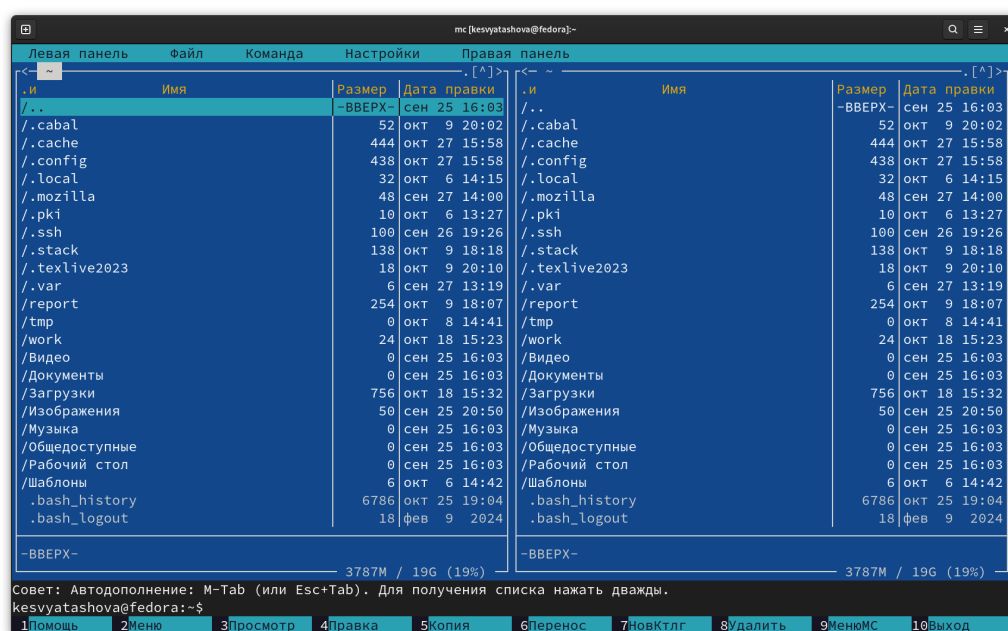


Рис. 3.1: Команда mc

2. Пользуясь клавишами на клавиатуре перейдем в каталог `~/work/arch-pc` созданный при выполнении лабораторной работы №4(рис. 3.2):

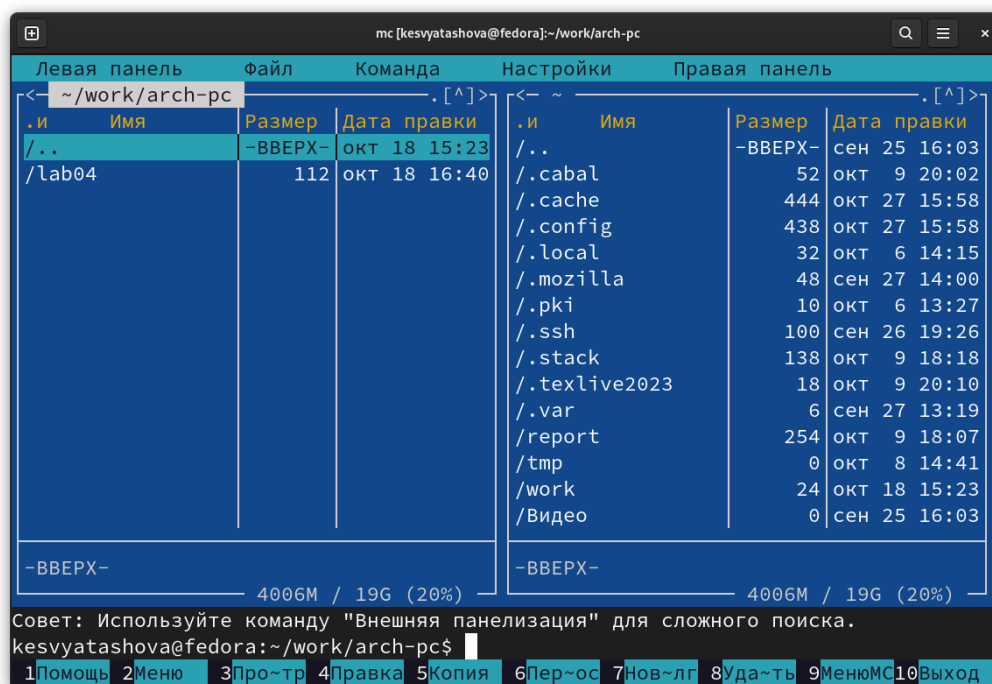


Рис. 3.2: Каталог arch-pc

3. С помощью функциональной клавиши F7 создадим папку lab05(рис. 3.3) и перейдем в созданный каталог(рис. 3.4):

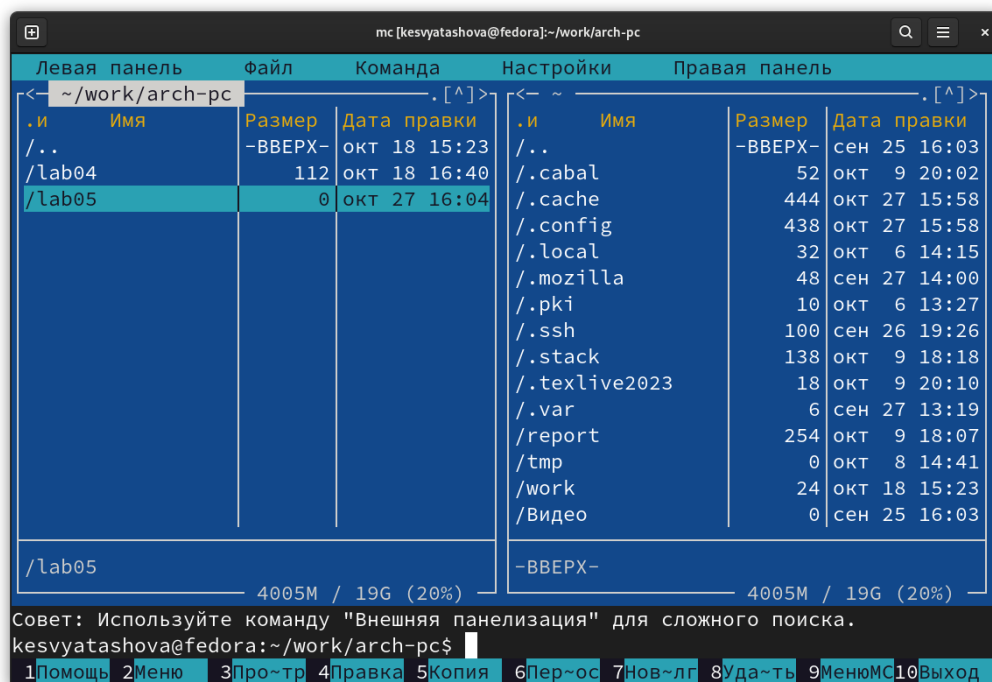


Рис. 3.3: Создание папки

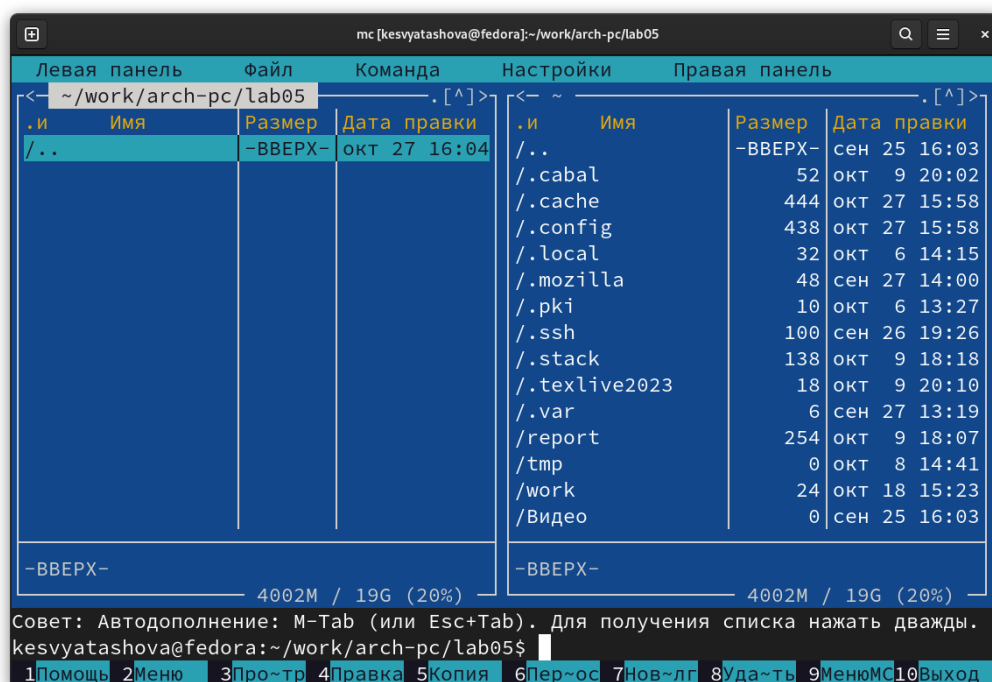


Рис. 3.4: Папка

4. Пользуясь строкой ввода и командой touch создадим файл lab5-1.asm(рис. 3.5):

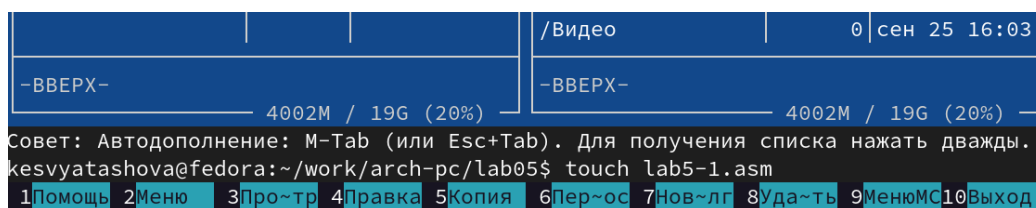


Рис. 3.5: Создание файла lab5-1.asm

5. С помощью функциональной клавиши F4 откроем файл lab5-1.asm для редактирования во встроенном редакторе mcedit(рис. 3.6):

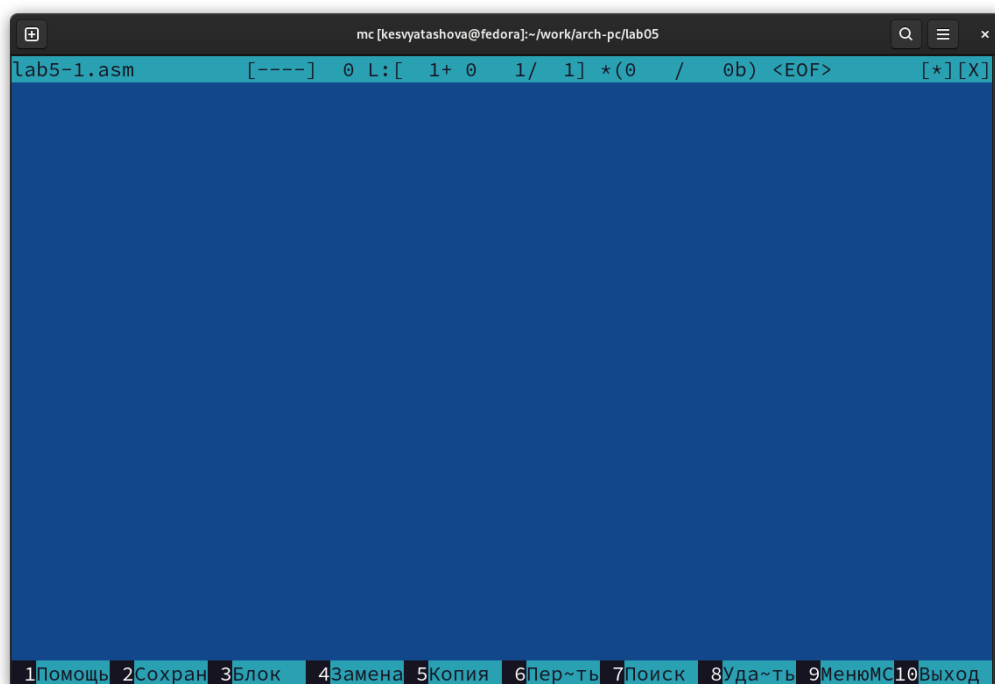


Рис. 3.6: Файл lab5-1.asm

6. Введем текст программы из листинга 1.1(рис. 3.7), сохраним изменения(рис. 3.8) и закроем файл:

```
mc [kesvyatashova@fedora]:~/work/arch-pc/lab05
lab5-1.asm [-M--] 0 L: [ 1+ 0 1/ 36] *(0 /2432b) 0059 0x03B [*] [X]
;-----
; Программа вывода сообщения на экран и ввода строки с клавиатуры
;-----
;----- 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
;----- 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
1Помощь 2Сохран 3Блок 4Замена 5Копия 6Пер~ть 7Поиск 8Уда~ть 9МенюМС10Выход
```

Рис. 3.7: Ввод текста из листинга

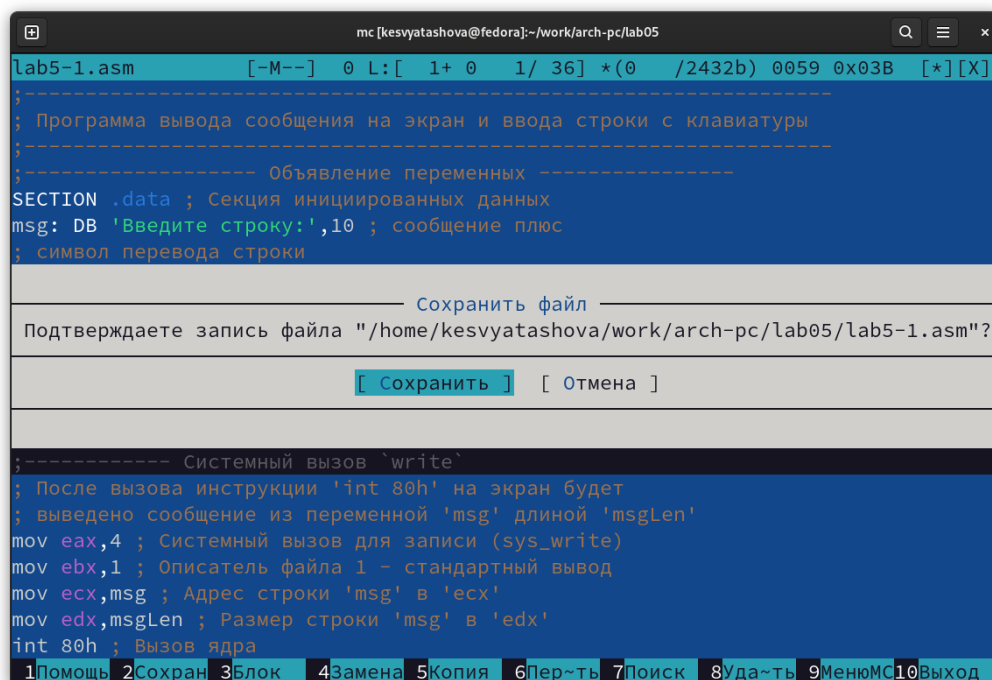


Рис. 3.8: Сохранения файла с текстом из листинга

Листинг 1.1. Программа вывода сообщения на экран и ввода строки с клавиатуры

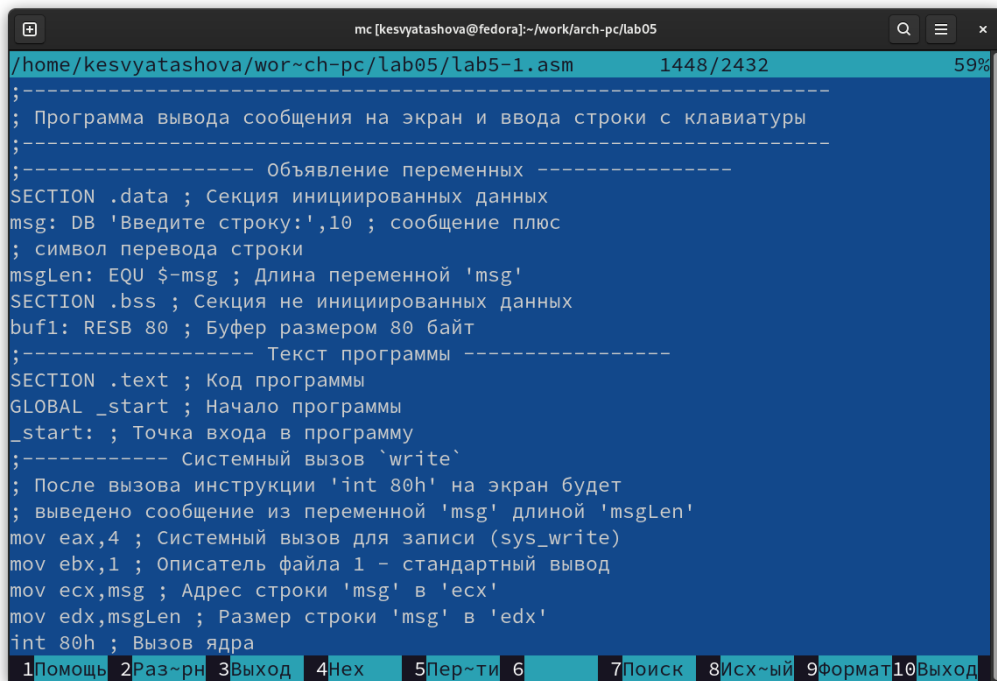
```

;----- ; Программа вывода сообщения на
экран и ввода строки с клавиатуры ;-----
;----- 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 ; Бу-
фер размером 80 байт ;----- Текст программы ----- SECTION
.text ; Код программы GLOBAL _start ; Начало программы _start: ; Точка
входа в программу ;----- Системный вызов write ; После вызова ин-
струкции '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 ; Си-
 стемный 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 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

7. С помощью функциональной клавиши F3 откройте файл lab5-1.asm для просмотра. Убедимся, что файл содержит текст программы(рис. 3.9):



```
mc [kesvyatashova@fedora]:~/work/arch-pc/lab05
/home/kesvyatashova/work/arch-pc/lab05/lab5-1.asm 1448/2432 59%
;-----
; Программа вывода сообщения на экран и ввода строки с клавиатуры
;-----
;----- 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
;----- 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
1Помощь 2Раз-рн 3Выход 4Нех 5Пер-ти 6 7Поиск 8Исх-ый 9Формат10Выход
```

Рис. 3.9: Проверка файла

8. Оттранслируем текст программы lab5-1.asm в объектный файл. Выполним компоновку объектного файла и запустим получившийся исполняемый файл. Программа выводит строку 'Введите строку:' и ожидает ввода с клавиатуры. На запрос введу мое ФИО(рис. 3.10):

```
kesvyatashova@fedora:~/work/arch-pc/lab05$ nasm -f elf lab5-1.asm
kesvyatashova@fedora:~/work/arch-pc/lab05$ ld -m elf_i386 -o lab5-1 lab5-1.o
kesvyatashova@fedora:~/work/arch-pc/lab05$ ./lab5-1
Введите строку:
Святасова Ксения Евгеньевна
kesvyatashova@fedora:~/work/arch-pc/lab05$
```

Рис. 3.10: Программа

3.2 Подключение внешнего файла `in_out.asm`

Для упрощения написания программ часто встречающиеся одинаковые участки кода (такие как, например, вывод строки на экран или выход из программы) можно оформить в виде подпрограмм и сохранить в отдельные файлы, а во всех нужных местах поставить вызов нужной подпрограммы. Это позволяет сделать основную программу более удобной для написания и чтения. NASM позволяет подключать внешние файлы с помощью директивы `%include`, которая предписывает ассемблеру заменить эту директиву содержимым файла. Подключаемые файлы также написаны на языке ассемблера. Важно отметить, что директива `%include` в тексте программы должна стоять раньше, чем встречаются вызовы подпрограмм из подключаемого файла. Для вызова подпрограммы из внешнего файла используется инструкция `call`, которая имеет следующий вид

`call` где `function` имя подпрограммы

9. Скачаем файл `in_out.asm` со страницы курса в ТУИС.
10. Подключаемый файл `in_out.asm` должен лежать в том же каталоге, что и файл с программой, в которой он используется.

В одной из панелей `mc` откроем каталог с файлом `lab5-1.asm`. В другой панели каталог со скачанным файлом `in_out.asm` (рис. 3.11). Скопируем этот файл в каталог с файлом `lab5-1.asm` с помощью функциональной клавиши `F5` (рис. 3.12):

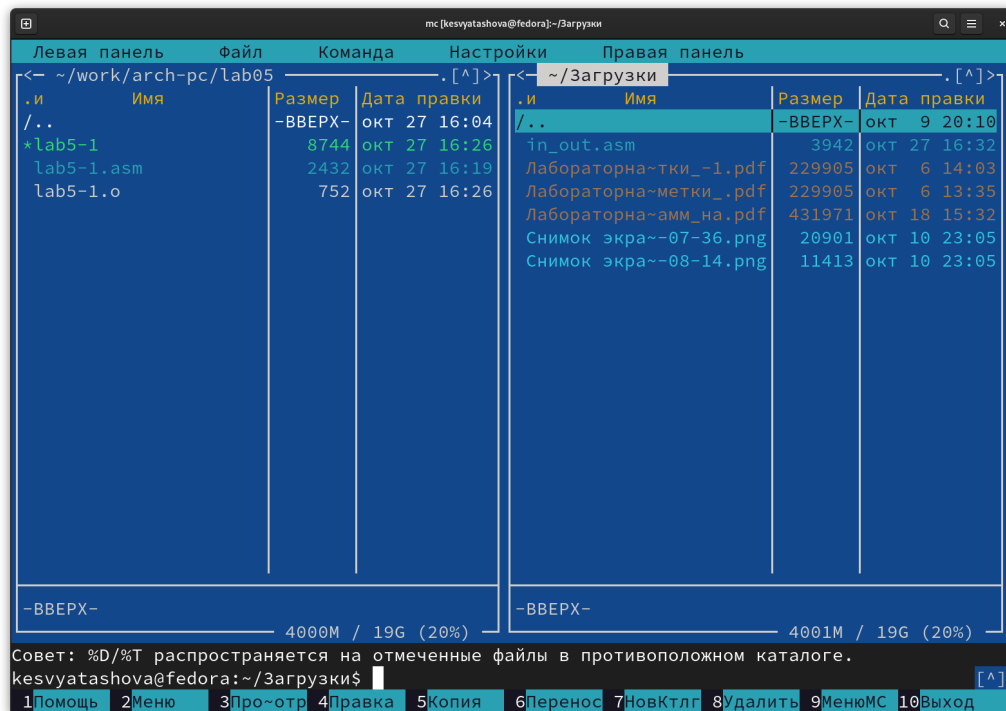


Рис. 3.11: Каталог с файлом in_out.asm

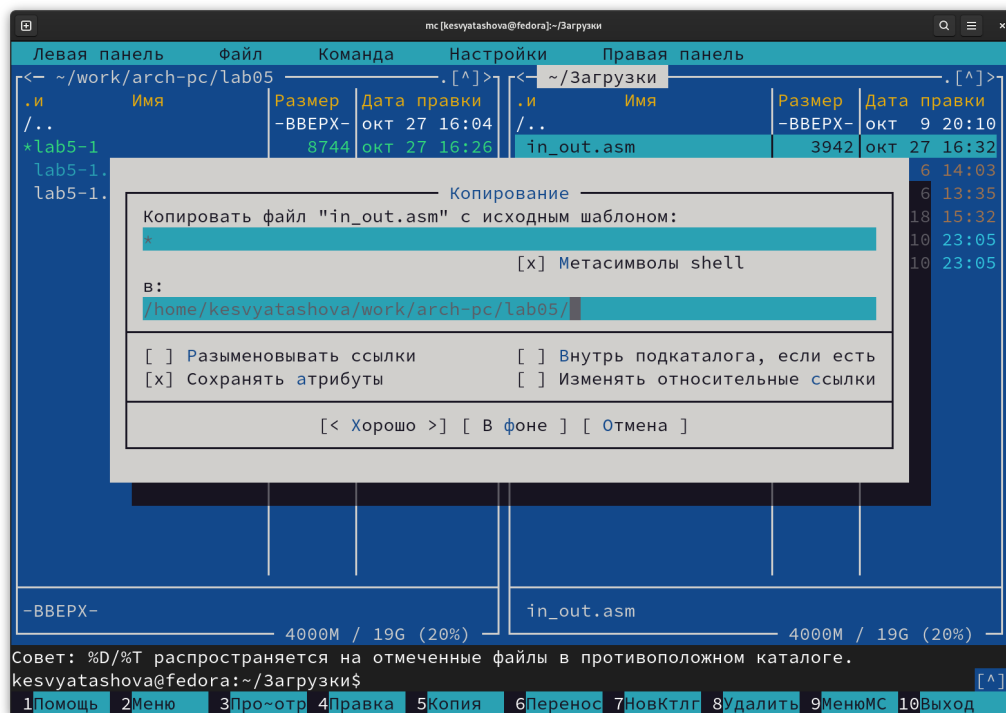


Рис. 3.12: Копирование файла

11. С помощью функциональной клавиши F6 создадим копию файла lab5-1.asm с именем lab5-2.asm. Выделим файл lab5-1.asm, нажмем клавишу F6, введем имя файла lab5-2.asm и нажмем клавишу Enter(рис. 3.13):

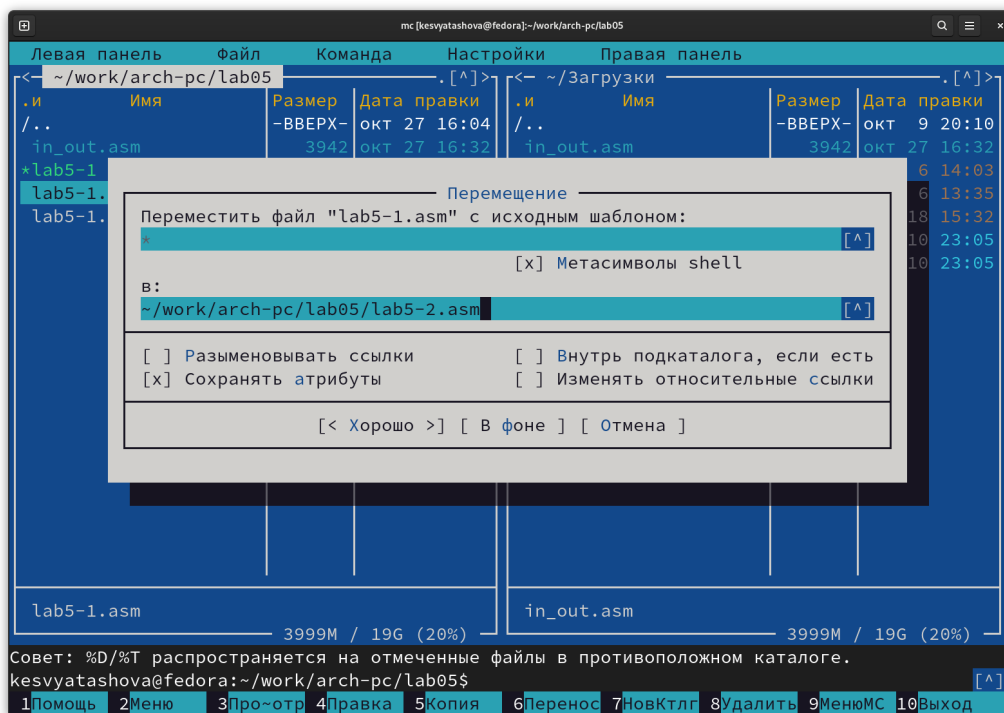
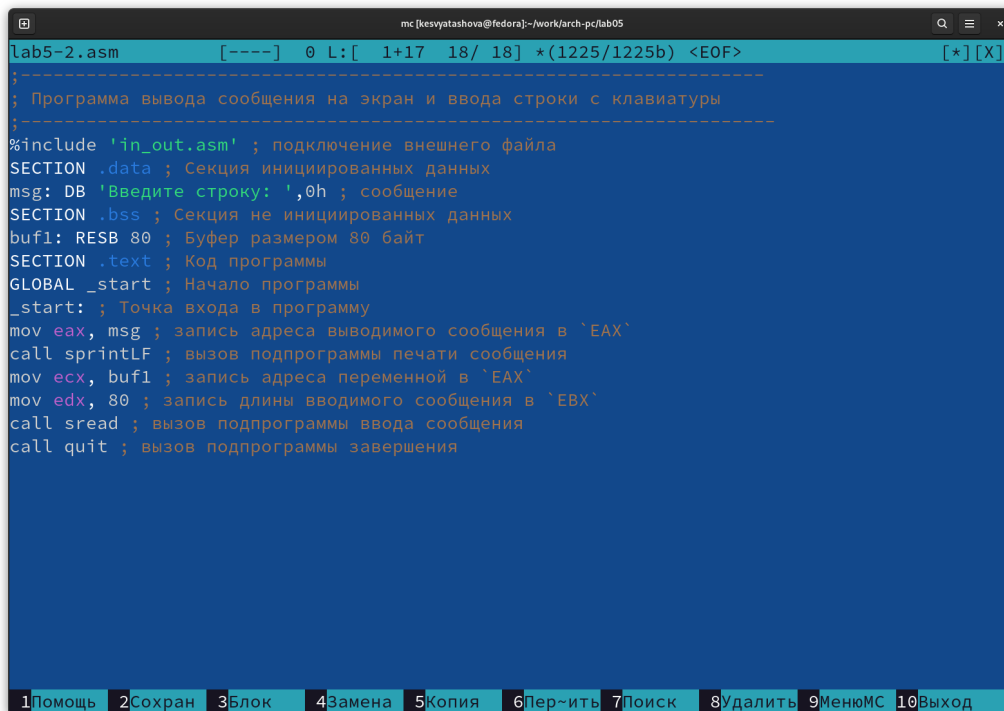


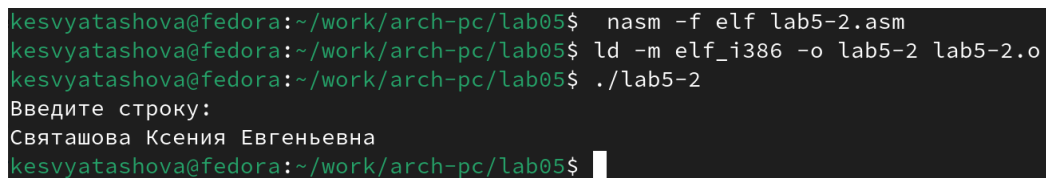
Рис. 3.13: Копия файла

12. Исправим текст программы в файле lab5-2.asm с использованием подпрограмм из внешнего файла in_out.asm (используем подпрограммы sprintLF, sread и quit) в соответствии с листингом 1.2 (рис. 3.14). Создадим исполняемый файл и проверим его работу (рис. 3.15):



```
lab5-2.asm      [----]  0 L: [ 1+17 18/ 18] *(1225/1225b) <EOF> [*] [X]
;
; 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
call quit ; вызов подпрограммы завершения
```

Рис. 3.14: Исправление текста



```
kesvyatashova@fedora:~/work/arch-pc/lab05$ nasm -f elf lab5-2.asm
kesvyatashova@fedora:~/work/arch-pc/lab05$ ld -m elf_i386 -o lab5-2 lab5-2.o
kesvyatashova@fedora:~/work/arch-pc/lab05$ ./lab5-2
Введите строку:
Святасова Ксения Евгеньевна
kesvyatashova@fedora:~/work/arch-pc/lab05$
```

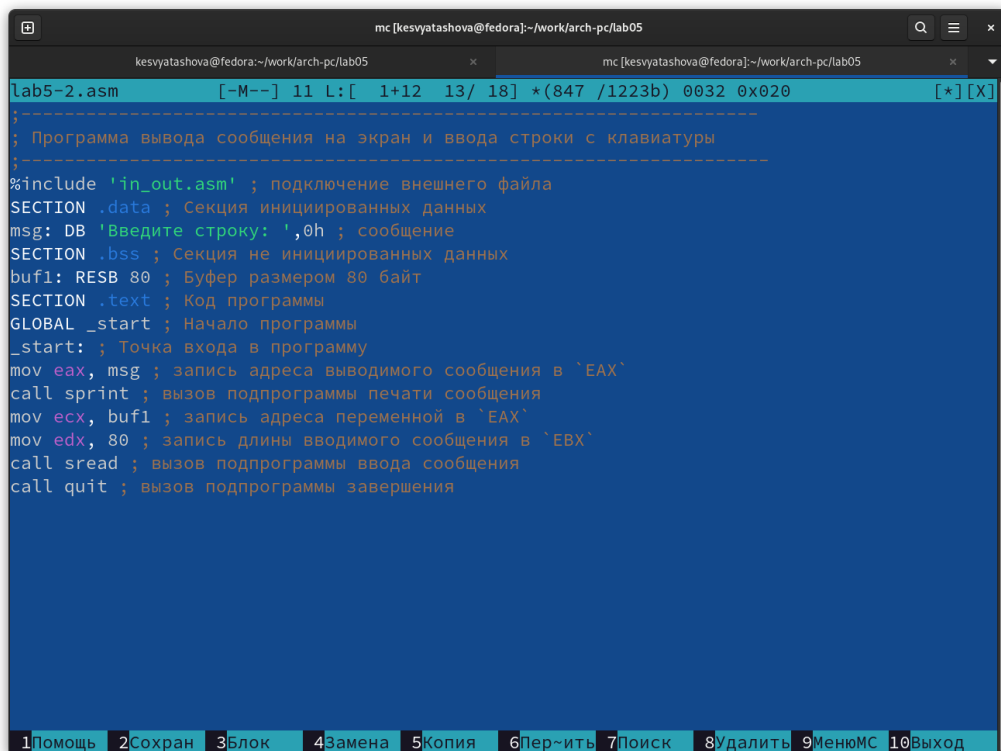
Рис. 3.15: Проверка работы программы

Листинг 1.2. Программа вывода сообщения на экран и ввода строки с клавиатуры с использованием файла in_out.asm

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

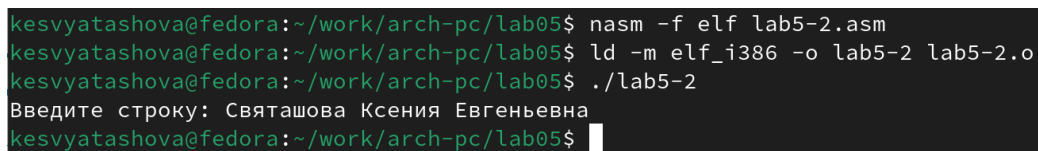
%include 'in_out.asm' ; подключение внешнего файла SECTION .data ; Секция инициированных данных msg: DB 'Введите строку:',0h ; сообщение SECTION .bss ; Секция не инициированных данных buf1: RESB 80 ; Буфер размером 80 байт SECTION .text ; Код программы GLOBAL _start ; Начало программы _start: ; Точка входа в программу mov eax, msg ; запись адреса выводимого сообщения в EAX call 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 call quit ; вызов подпрограммы завершения

13. В файле lab5-2.asm заменим подпрограмму sprintLF на sprint(рис. 3.16). Создадим исполняемый файл и проверим его работу(рис. 3.17).



```
lab5-2.asm [-M--] 11 L: [ 1+12 13/ 18] *(847 /1223b) 0032 0x020 [*] [X]
;
; Программа вывода сообщения на экран и ввода строки с клавиатуры
;
-----
%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
call quit ; вызов подпрограммы завершения
```

Рис. 3.16: Замена sprintf на sprint



```
kesvyatashova@fedora:~/work/arch-pc/lab05$ nasm -f elf lab5-2.asm
kesvyatashova@fedora:~/work/arch-pc/lab05$ ld -m elf_i386 -o lab5-2 lab5-2.o
kesvyatashova@fedora:~/work/arch-pc/lab05$ ./lab5-2
Введите строку: Святасова Ксения Евгеньевна
kesvyatashova@fedora:~/work/arch-pc/lab05$
```

Рис. 3.17: Проверка программы

Разница между первым исполняемым файлом lab5-2 и вторым lab5-2-2 в том, что запуск первого запрашивает ввод с новой строки, а программа, которая исполняется при запуске второго, запрашивает ввод без переноса на новую строку.

4 Задания для самостоятельной работы

1. Создадим копию файла lab5-1.asm(рис. 4.1). Внесем изменения в программу(рис. 4.2)(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.

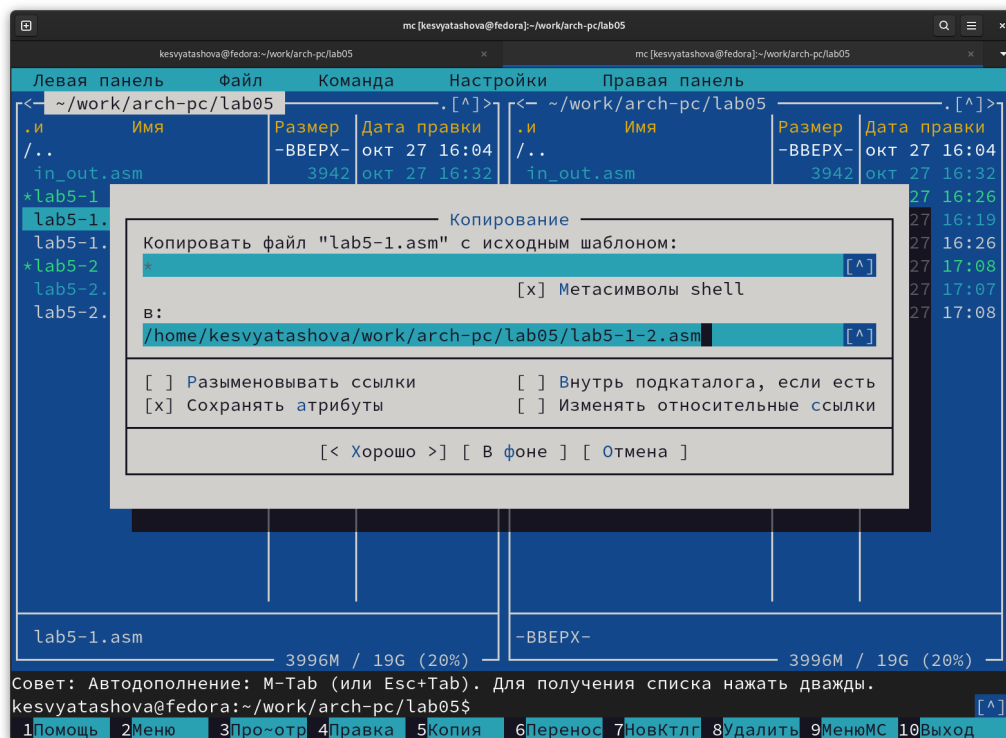
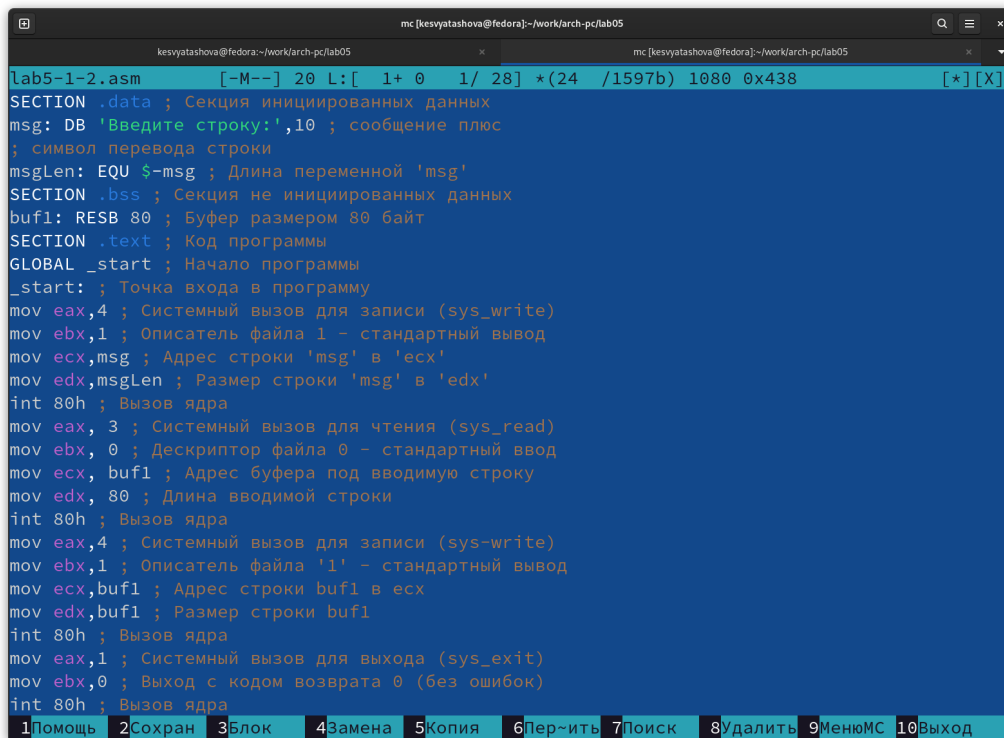


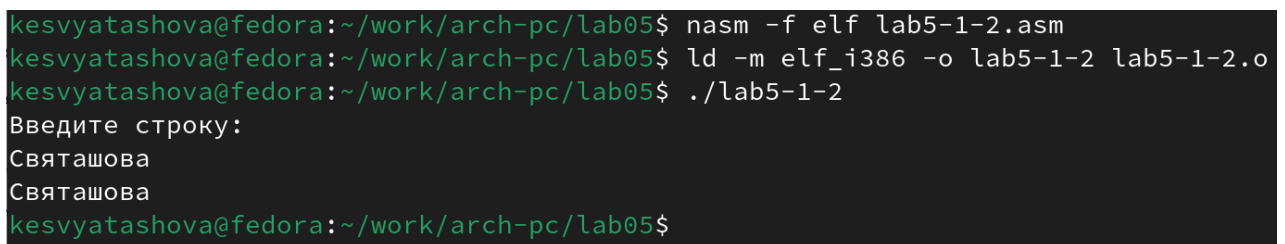
Рис. 4.1: Копирование файла



```
lab5-1-2.asm [-M--] 20 L: [ 1+ 0 1/ 28] *(24 /1597b) 1080 0x438 [*][X]
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
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
1Помощь 2Сохран 3Блок 4Замена 5Копия 6Пер-ить 7Поиск 8Удалить 9МенюМС 10Выход
```

Рис. 4.2: Исправленная программа

2. Получим исполняемый файл и проверьте его работу. На приглашение ввести строку введу свою фамилию(рис. 4.3):



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

Рис. 4.3: Программа

3. Создадим копию файла lab5-2.asm(рис. 4.4). Исправим текст програм-

мы(рис. 4.5) с использование 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.:

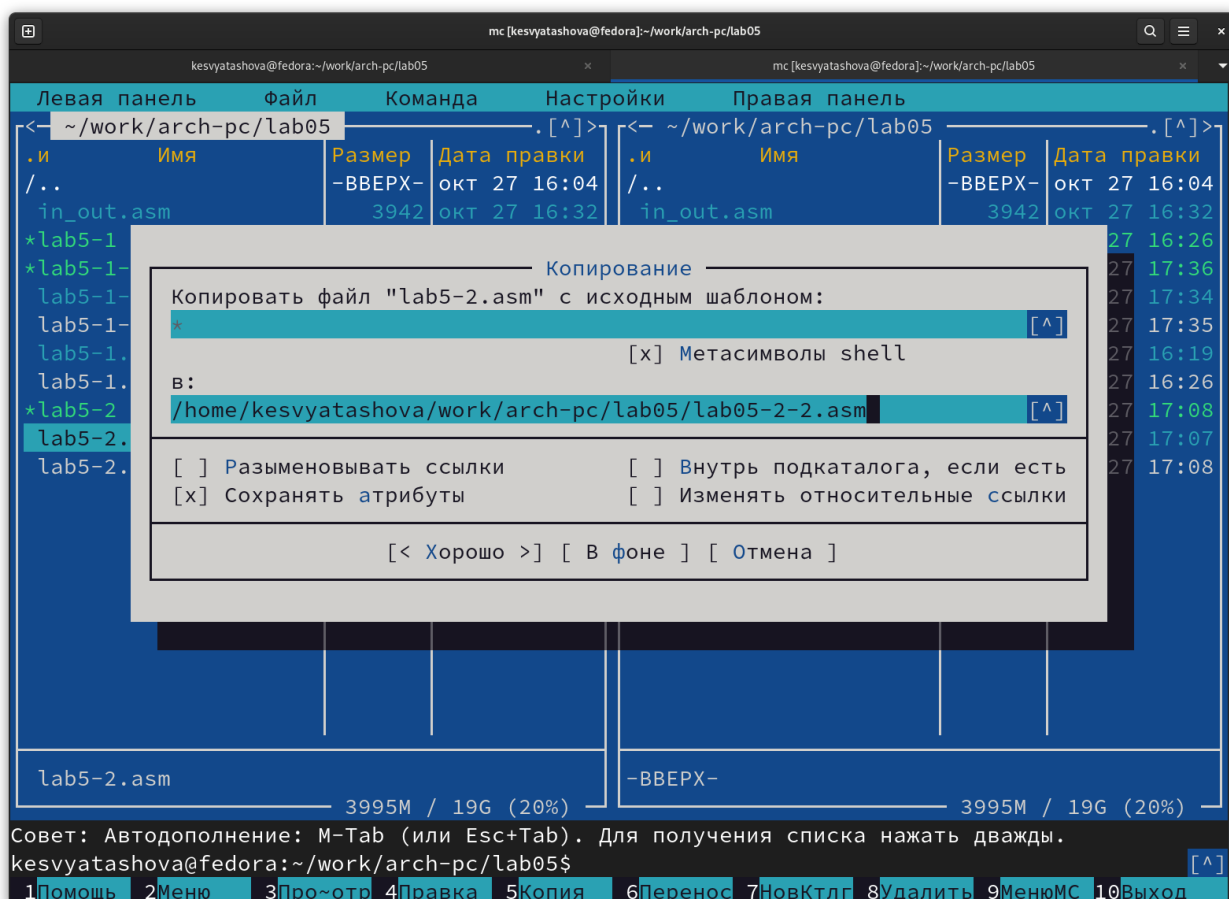
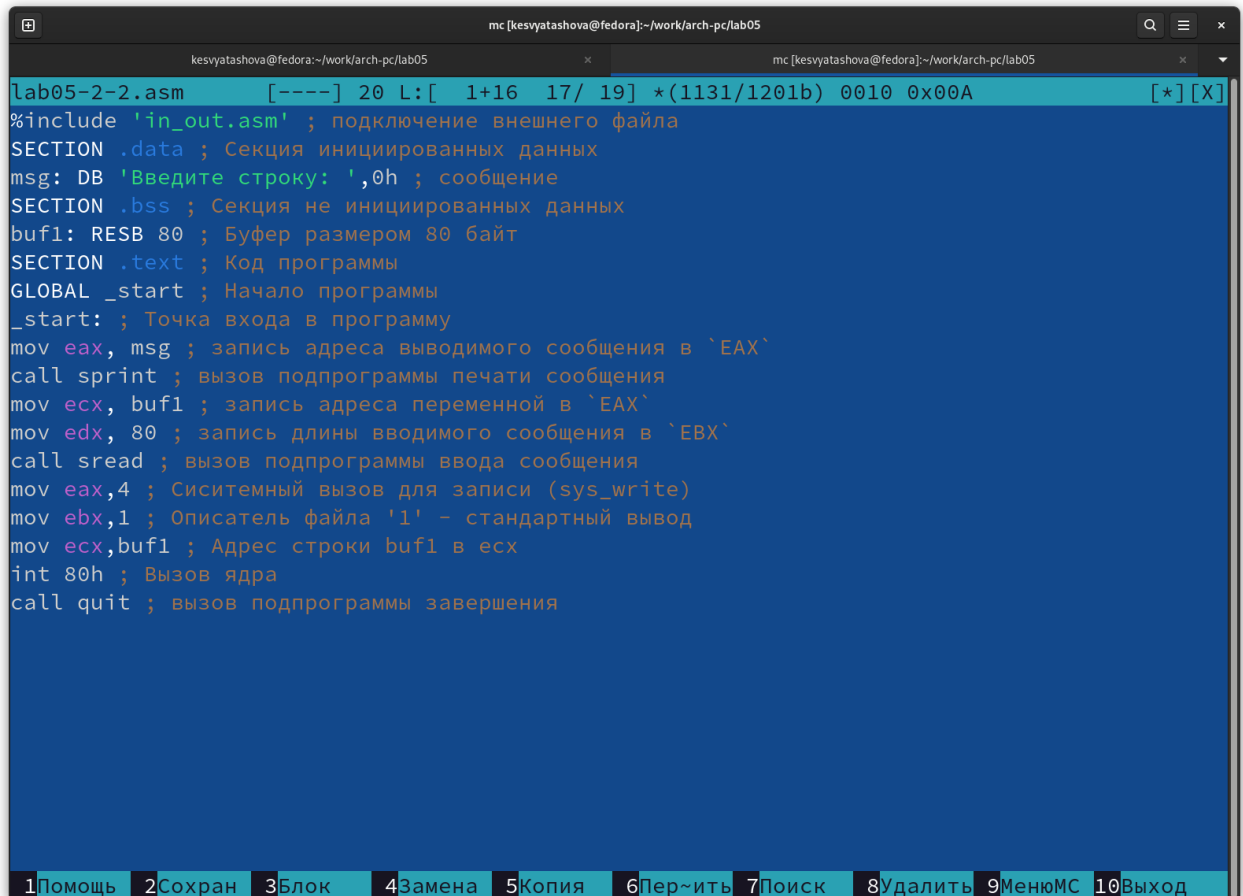


Рис. 4.4: Копирование файла



```
lab05-2-2.asm [----] 20 L:[ 1+16 17/ 19] *(1131/1201b) 0010 0x00A [*][X]
#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,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
```

1Помощь 2Сохран 3Блок 4Замена 5Копия 6Пер~ить 7Поиск 8Удалить 9МенюМС 10Выход

Рис. 4.5: Исправленная программа

4. Создайте исполняемый файл и проверьте его работу(рис. 4.6):

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

Рис. 4.6: Программа

5 Вывод

В результате выполнения работы я приобрела практические навыки работы в *Midnight Commander* и освоила язык ассемблера `mov` и `int`.