

Лабораторная работа №5

Midnight Commander

Архитектура компьютера и Операционные системы

Ван Сихэм Франклин О Нил Джон

Содержание

1	Цель работы	5
2	Введение	6
3	Порядок выполнения лабораторной работы	7
3.1	Задание для самостоятельной работы	21
3.2	Листинг 5.1	25
3.3	Листинг 5.2	26
4	Заключение	28

Список иллюстраций

2.1	Midnight Commander	6
3.1	команда чтобы открыть Midnight Commander	7
3.2	Редактор Midnight Commander	8
3.3	Перешёл в каталог ~/work/arch-pc	8
3.4	Создаем lab05	9
3.5	Создан lab05	9
3.6	Создаем lab5-1.asm с помощью touch	10
3.7	Создан lab05-1.asm	10
3.8	lab05-1.asm используя mcedit	11
3.9	Текст программы из листинга 5.1 в nano	12
3.10	Текст программы из листинга 5.1 в mcedit	13
3.11	lab5-1.asm	14
3.12	Команда nasm -f elf lab5-1.asm	15
3.13	lab5-1.o был создан	15
3.14	ld -m elf_i386 -o lab5-1 lab5-1.o	16
3.15	*lab5-1 был создан	16
3.16	Команда ./lab5-1	16
3.17	ФИО ввел	17
3.18	in_out.asm файл скачанный	17
3.19	in_out.asm файл скопирован в каталог ~/work/arch-pc/lab05	17
3.20	in_out.asm файл в каталог ~/Документы	18
3.21	in_out.asm файл копируется в каталог ~/work/arch-pc/lab05	18
3.22	in_out.asm файл скопирован в каталог ~/work/arch-pc/lab05	19
3.23	lab5-2.asm был создан	19
3.24	lab5-2.asm с использование подпрограмм из внешнего файла in_out.asm	20
3.25	заменял подпрограмму sprintLF на sprint	20
3.26	копия файла lab5-1.asm	21
3.27	редактировал файл lab5-1.asm	22
3.28	вывод lab5-lab5-1.asm	23
3.29	вывод lab5-lab5-1.asm	23
3.30	вывод lab5-lab5-1.asm	24
3.31	вывод lab5-lab5-1.asm	24

Список таблиц

1 Цель работы

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

2 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. Для активации оболочки Midnight Commander достаточно ввести в командной строке mc и нажать клавишу Enter. В Midnight Commander используются функциональные клавиши F1 — F10, к которым привязаны часто выполняемые операции.

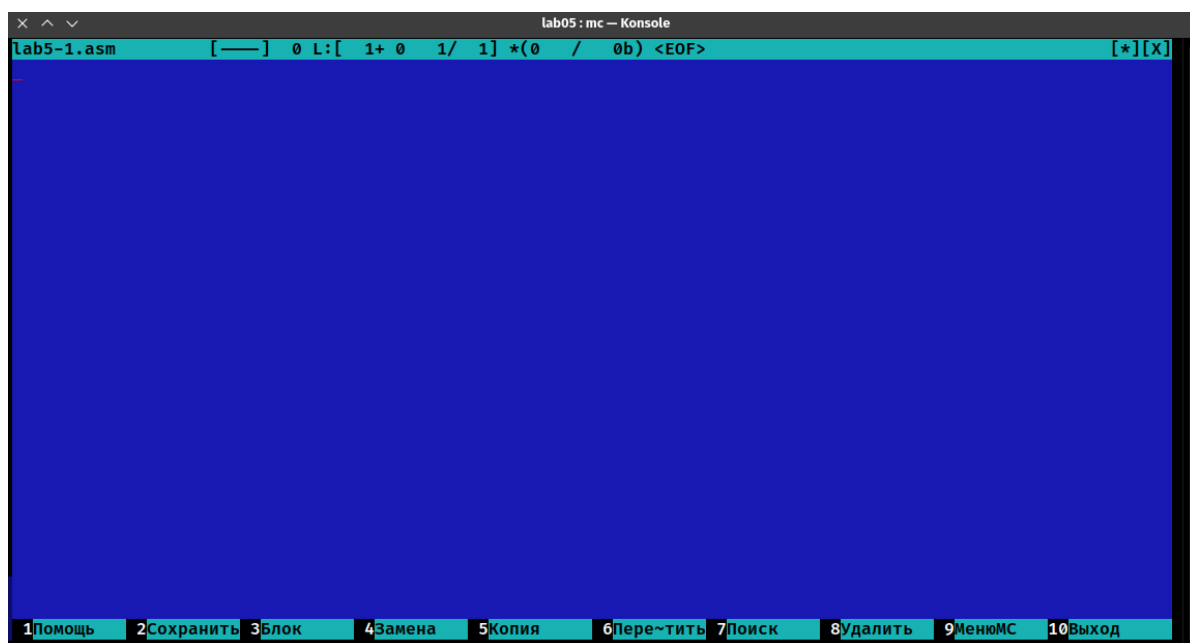


Рис. 2.1: Midnight Commander

3 Порядок выполнения лабораторной работы

1. Откройте Midnight Commander

```
print(mishanya4u@Legenda in ~ via C v13.2.1-gcc)  
>> mc
```

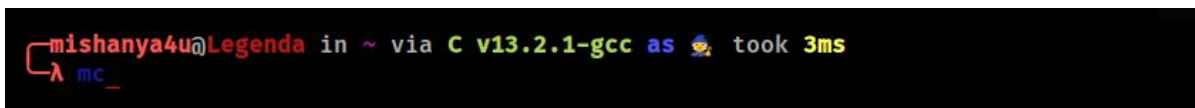


Рис. 3.1: команда чтобы открыть Midnight Commander

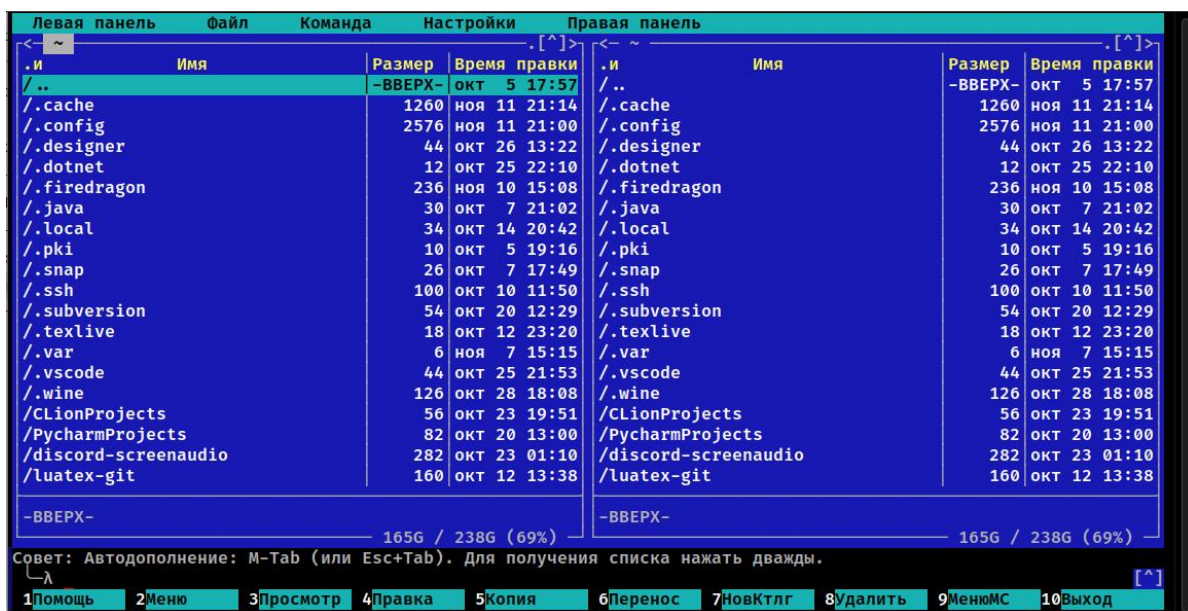


Рис. 3.2: Редактор Midnight Commander

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

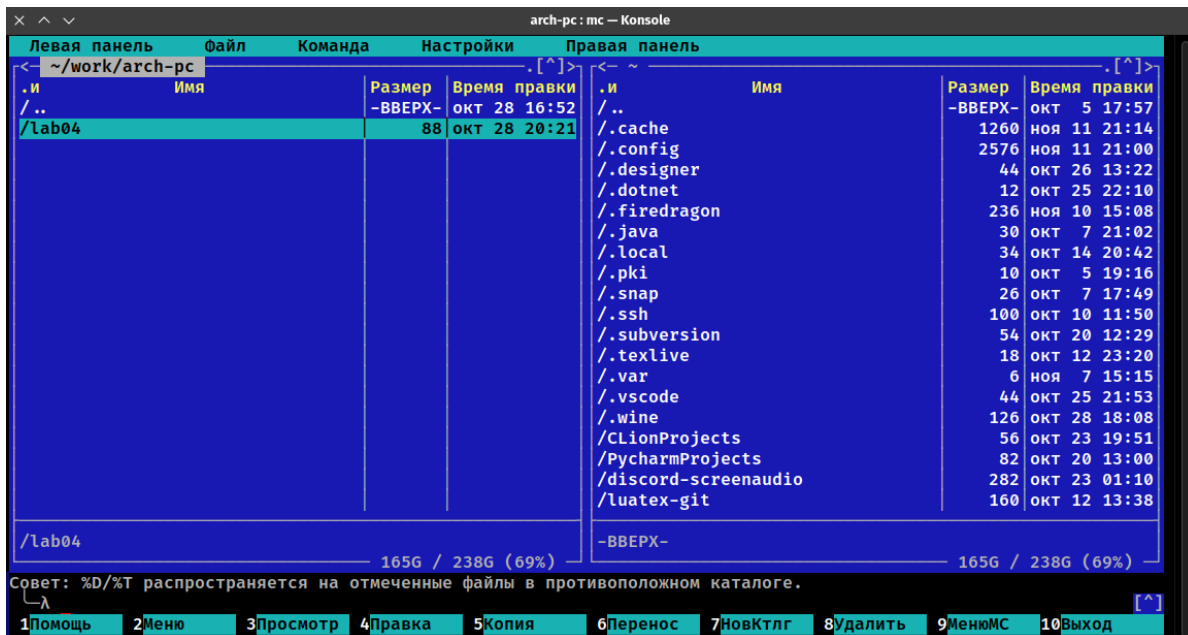


Рис. 3.3: Перешёл в каталог `~/work/arch-pc`

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

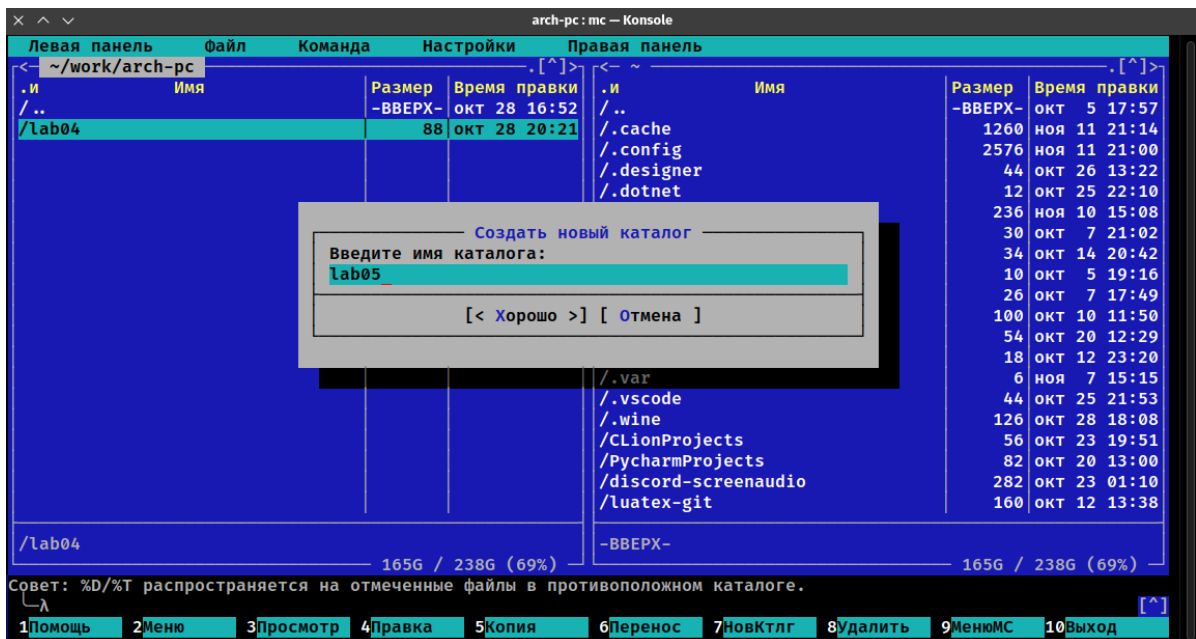


Рис. 3.4: Создаем lab05

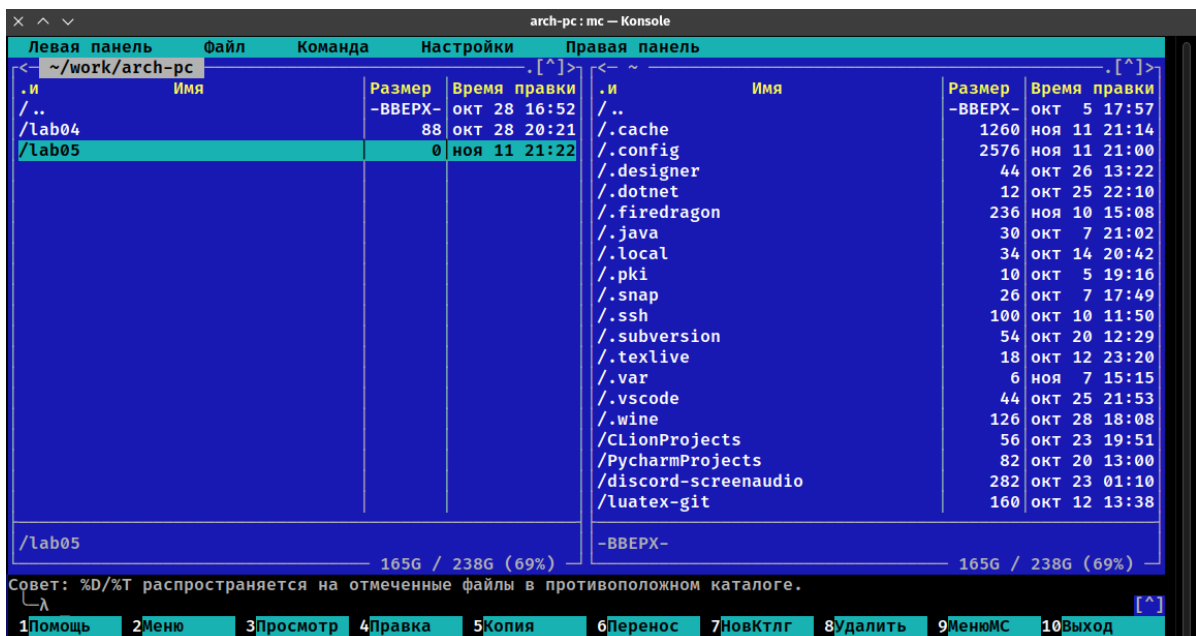


Рис. 3.5: Создан lab05

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

```
mishanya4u@Legenda in ~ via C v13.2.1-gcc took 5m26s
λ mc

λ touch lab5-1.asm

mishanya4u@Legenda in ~/work/arch-pc/lab05 took 1m30s
λ mc
```

Рис. 3.6: Создаем lab5-1.asm с помощью touch

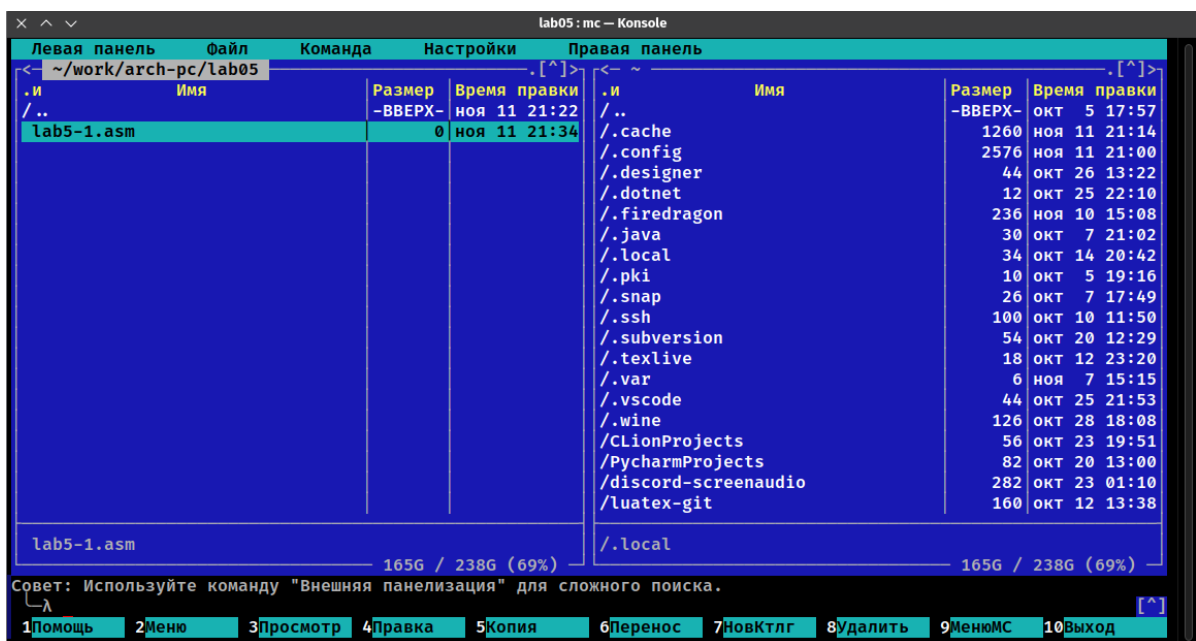


Рис. 3.7: Создан lab05-1.asm

5. С помощью функциональной клавиши F4 откройте файл lab5-1.asm для редактирования во встроенном редакторе. Как правило в качестве встроенного редактора Midnight Commander используется редактор mcedit.

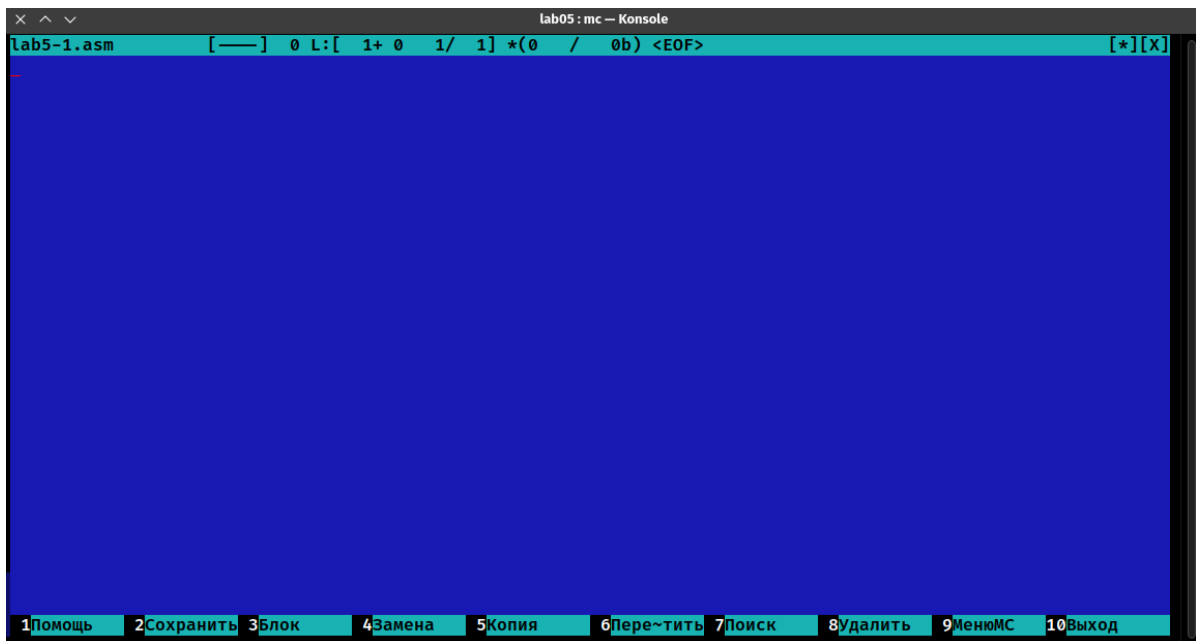


Рис. 3.8: lab05-1.asm используя mcedit

6. Введите текст программы из листинга 5.1, сохраните изменения и закройте файл.

```

1 ;
2 ; Программа вывода сообщения на экран и ввода строки с клавиатуры
3 ;
4
5 ; ----- Объявление переменных -----
6 SECTION .data ;Секция иницированных данных
7 msg: DB 'Введите строку:',10 ;сообщение плюс
8 ;символ перевода строки
9 msgLen: EQU $-msg ;Длина переменной 'msg'
10
11 SECTION .bss ;Секция не иницированных данных
12 buf1: RESB 80 ;Буфер размером 80 байт
13
14 ; ----- Текст программы -----
15
16 SECTION .text ; Код программы
17 GLOBAL _start ; Начало программы
18 _start: ; Точка входа в программу
19
20
21 ; ----- Системный вызов `write` -----
22 ; После вызова инструкции 'int 80h' на экран будет
23 ; выведено сообщение из переменной 'msg' длиной 'msgLen'
24
25 mov eax,4 ; Системный вызов для записи (sys_write)
26 mov ebx,1 ; Описатель файла 1 - стандартный вывод
27 mov ecx,msg ; Адрес строки 'msg' в 'ecx'
28 mov edx,msgLen ; Размер строки 'msg' в 'edx'
29 int 80h ; Вызов ядра
30
31
32 ; ----- системный вызов `read` -----
33 ; После вызова инструкции 'int 80h' программа будет ожидать ввода
34 ; строки, которая будет записана в переменную 'buf1' размером 80 байт
35
36 mov eax, 3 ; Системный вызов для чтения (sys_read)
37 mov ebx, 0 ;Descriptor файла 0 - стандартный ввод
38 mov ecx, buf1 ; Адрес буфера под вводимую строку
39 mov edx, 80 ; Длина вводимой строки
40 int 80h ; Вызов ядра
41
42
43 ; ----- Системный вызов `exit` -----
44 ; После вызова инструкции 'int 80h' программа завершит работу
45
46 mov eax,1 ; Системный вызов для выхода (sys_exit)
47 mov ebx,0 ; Выход с кодом возврата 0 (без ошибок)
48 int 80h ; Вызов ядра

```

Рис. 3.9: Текст программы из листинга 5.1 в nano

```

lab5-1.asm  [-----]  0 L:  1+ 0  1/ 48]  *(0  /2623b) 0059 0x03B
;
; 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'
;
;----- Секция не иницированных данных -----
SECTION .bss <-----> <-----> ;Секция не иницированных 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 <-----> <-----> ; 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

```

Рис. 3.10: Текст программы из листинга 5.1 в mcedit

7. С помощью функциональной клавиши F3 откройте файл lab5-1.asm для просмотра. Убедитесь, что файл содержит текст программы.

```

/home/mishanya4u/work/arch-pc/lab05/lab5-1.asm
;
; 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

;
; ----- 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                    ; 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

```

Рис. 3.11: lab5-1.asm

- 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.

mishanya4u@Legenda in ~ via C v13.2.1-gcc

```
print(nasm -f elf lab5-1.asm)
```

mishanya4u@Legenda in ~ via C v13.2.1-gcc

```
print(ld -m elf_i386 -o lab5-1 lab5-1.o)
```

mishanya4u@Legenda in ~ via C v13.2.1-gcc

```
print(./lab5-1)
```

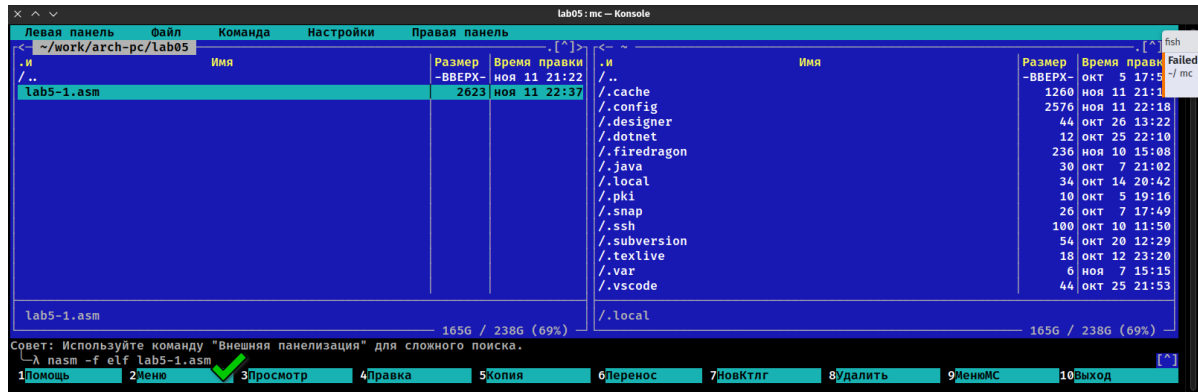


Рис. 3.12: Команда `nasm -f elf lab5-1.asm`

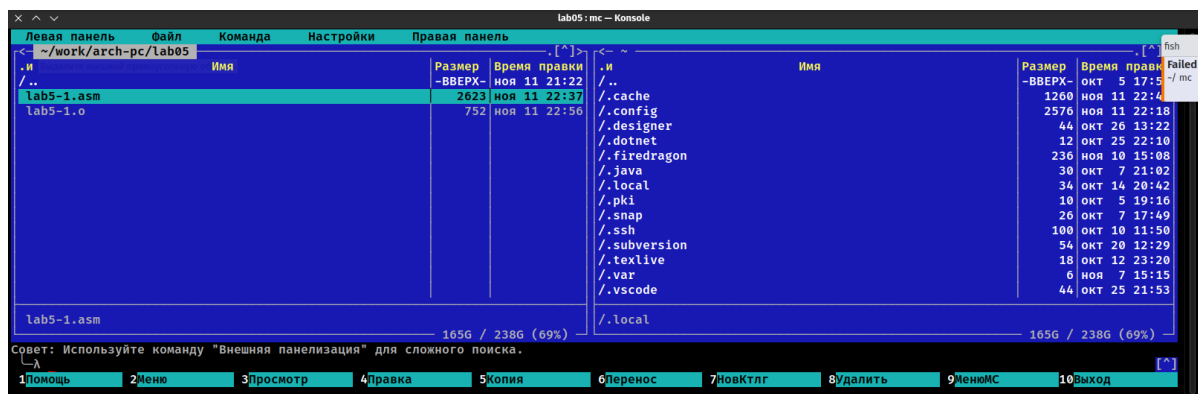


Рис. 3.13: lab5-1.o был создан

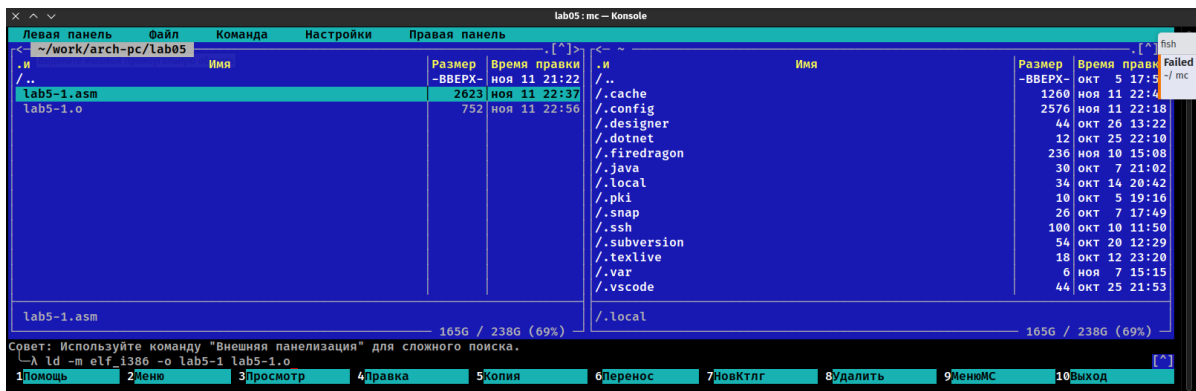


Рис. 3.14: `ld -m elf_i386 -o lab5-1 lab5-1.o`

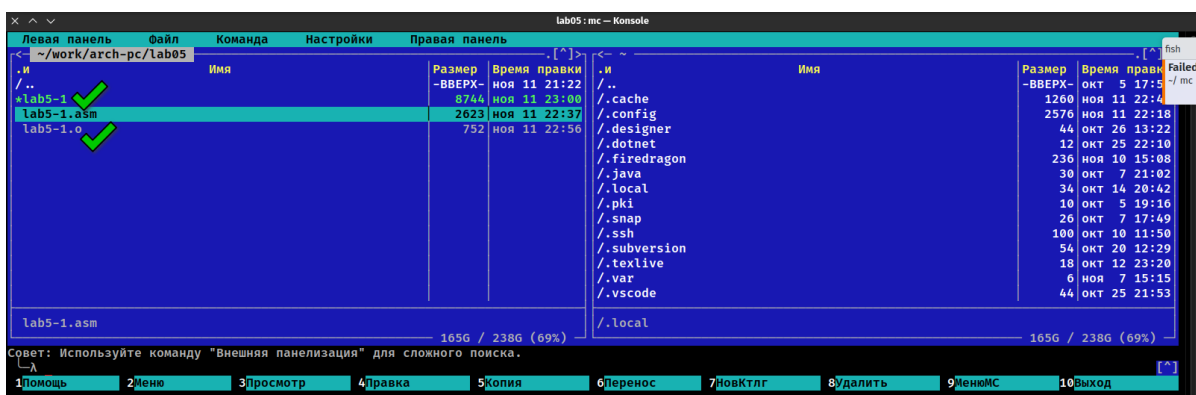


Рис. 3.15: *lab5-1 был создан

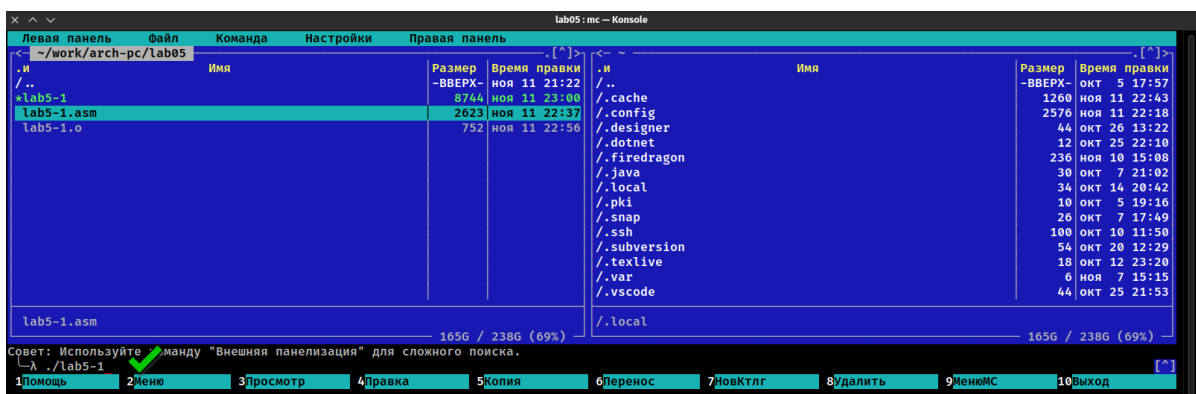


Рис. 3.16: Команда `./lab5-1`


```
mishanya4u@Legenda in ~/work/arch-pc/lab05 took 1m41s
└─λ nasm -f elf lab5-1.asm

└─λ ld -m elf_i386 -o lab5-1 lab5-1.o

└─λ ./lab5-1
Введите строку:
Ван Сихэм Франклин О Нил Джон (Миша)_
```




Рис. 3.17: ФИО ввел

9. Скачайте файл in_out.asm со страницы курса в ТУИС

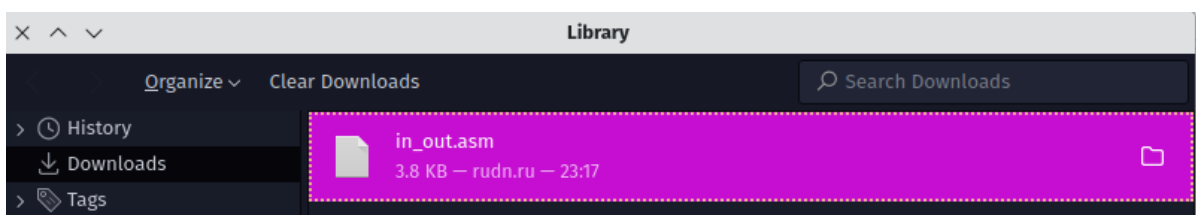


Рис. 3.18: in_out.asm файл скачанный

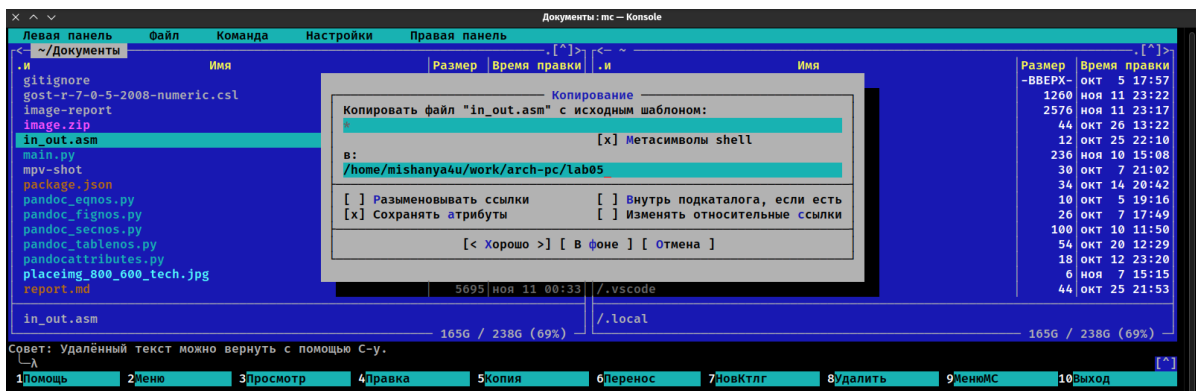


Рис. 3.19: in_out.asm файл скопирован в каталог ~/work/arch-pc/lab05

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 откройте каталог с файлом lab5-1.asm. В другой панели каталог со скачанным файлом in_out.asm (для перемещения между панелями используйте

Tab. Скопируйте файл in_out.asm в каталог с файлом lab5-1.asm с помощью функциональной клавиши F5.

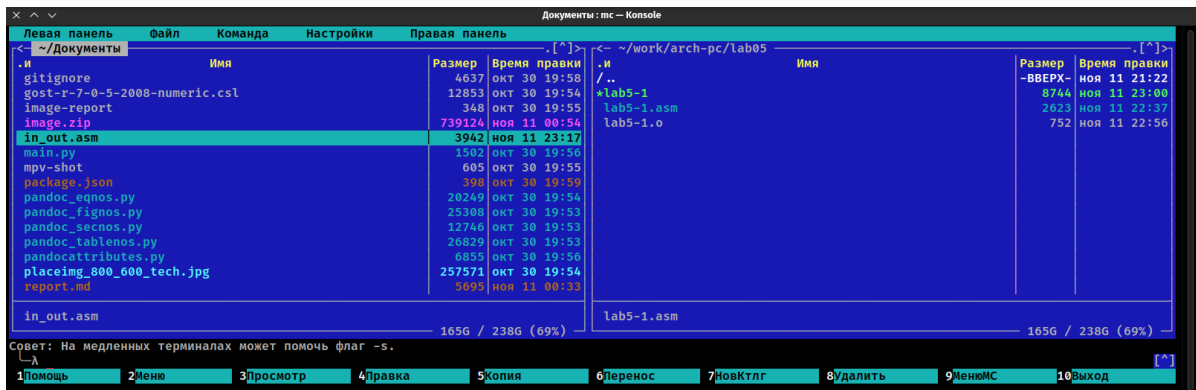


Рис. 3.20: in_out.asm файл в каталог ~/Документы

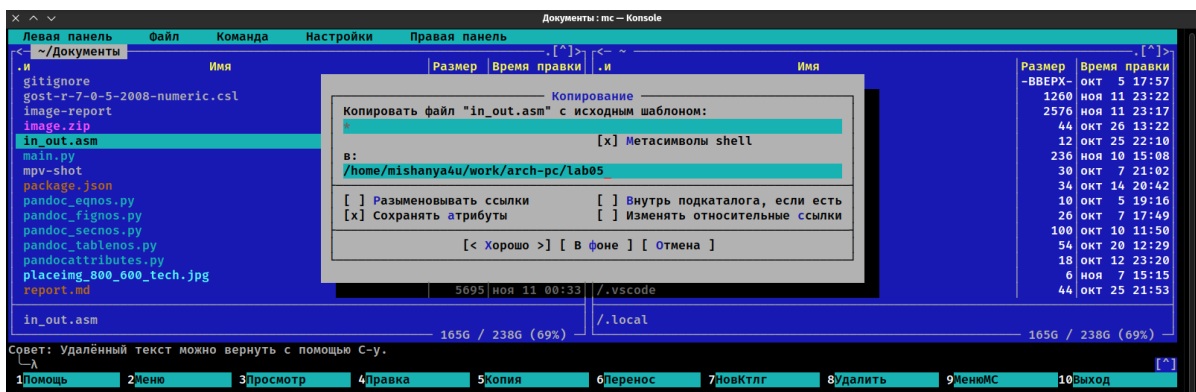


Рис. 3.21: in_out.asm файл копируется в каталог ~/work/arch-pc/lab05

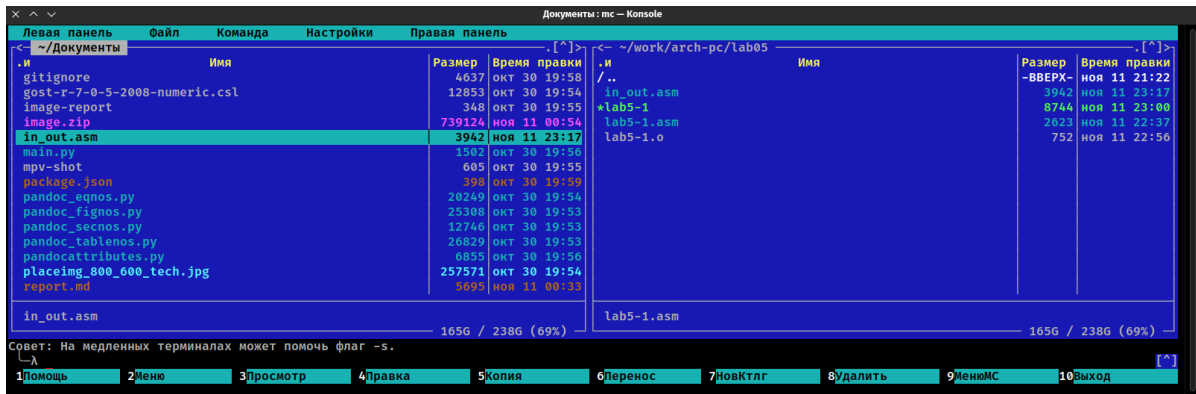


Рис. 3.22: in_out.asm файл скопирован в каталог ~/work/arch-pc/lab05

11. С помощью функциональной клавиши F6 создайте копию файла lab5-1.asm с именем lab5-2.asm.

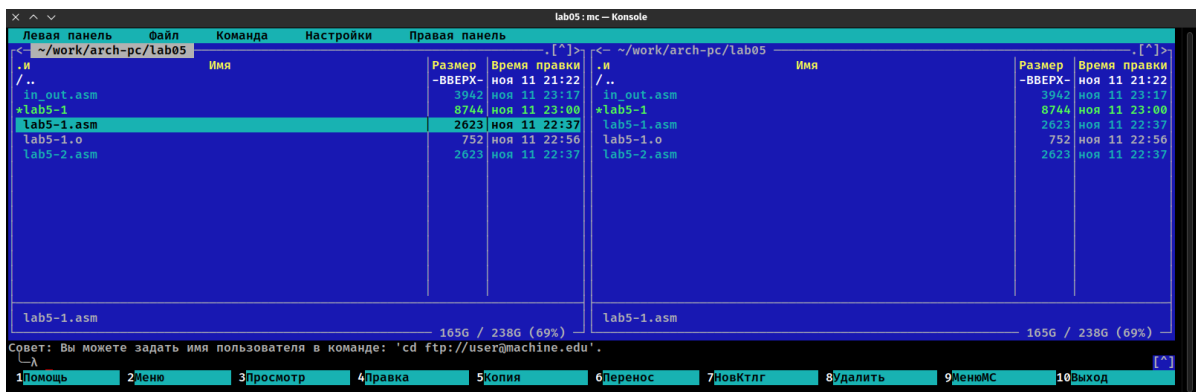


Рис. 3.23: lab5-2.asm был создан

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

3.1 Задание для самостоятельной работы

1. Создайте копию файла lab5-1.asm. Внесите 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.

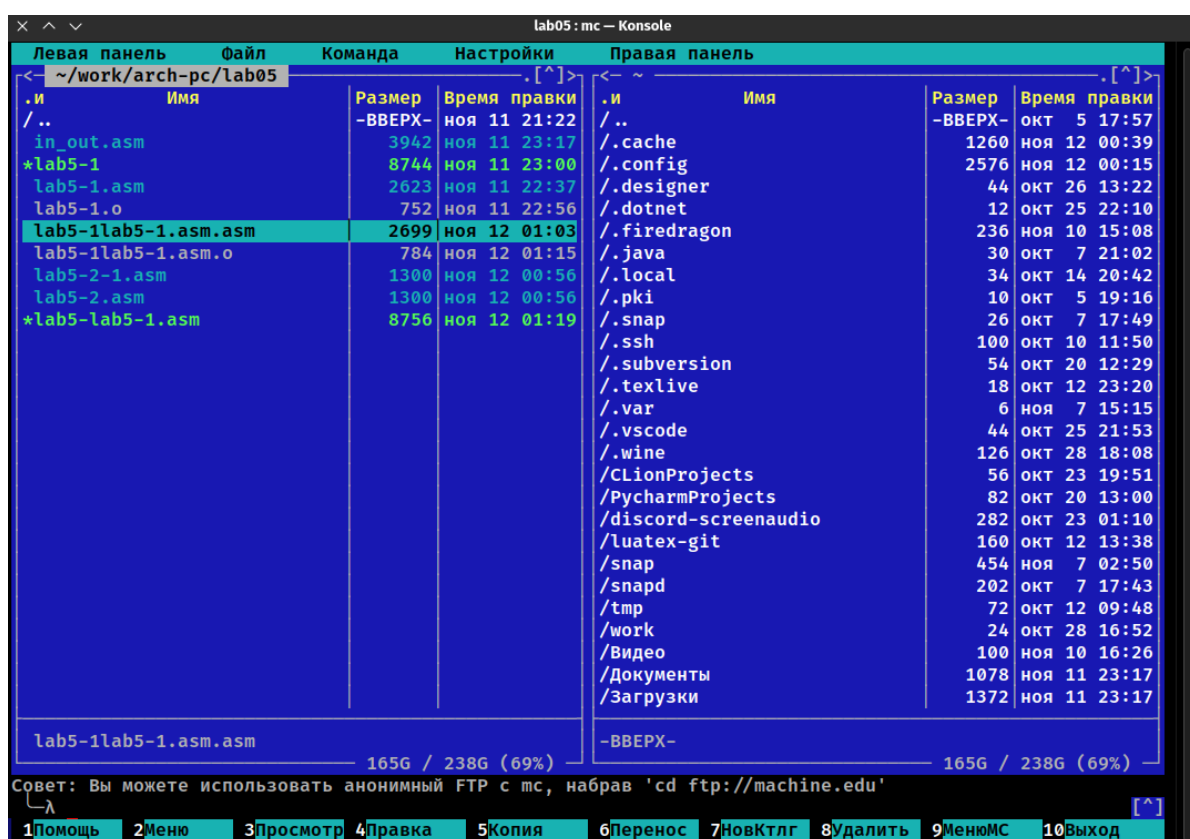


Рис. 3.26: копия файла lab5-1.asm

сти строку введите свою фамилию.

```
[?] * ./lab5-lab5-1.asm
Введите строку:
Миша Архангельский
Миша Архангельский
```

Рис. 3.28: вывод lab5-lab5-1.asm

3. Создайте копию файла lab5-2.asm. Исправьте 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.

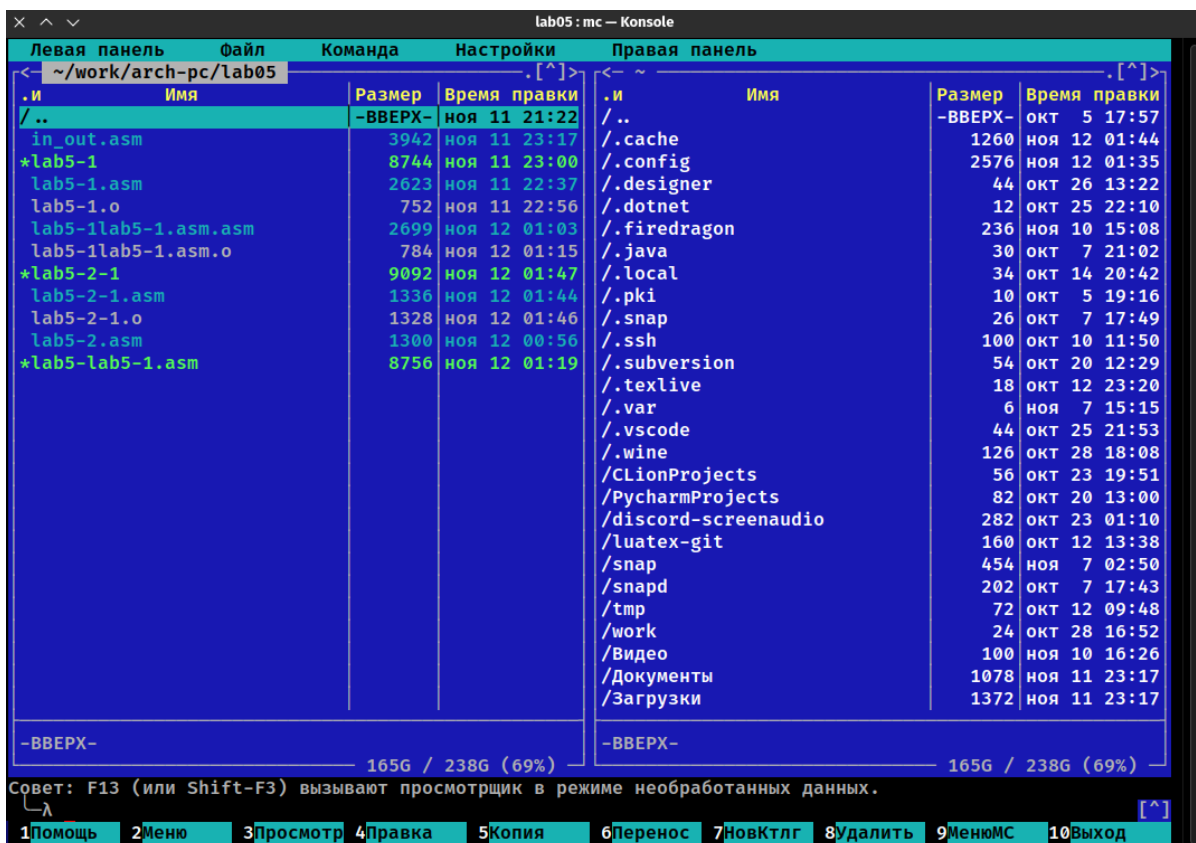


Рис. 3.29: вывод lab5-lab5-1.asm

```
lab05: mc — Konsole
1 ;
2 ; Программа вывода сообщения на экран и ввода строки с клавиатуры
3 ;
4
5 %include 'in_out.asm'          ; подключение внешнего файла
6
7 SECTION .data                 ; Секция инициированных данных
8 msg: DB 'Введите строку: ',0h ; сообщение
9
10 SECTION .bss                  ; Секция не инициированных данных
11 buf1: RESB 80                  ; Буфер размером 80 байт
12
13 SECTION .text                  ; Код программы
14 GLOBAL _start                  ; Начало программы
15 _start:                        ; Точка входа в программу
16
17 mov eax, msg                    ; запись адреса выводимого сообщения в `EAX`
18 call sprintf                    ; вызов подпрограммы печати сообщения
19 mov ecx, buf1                   ; запись адреса переменной в `EAX`
20 mov edx, 80                     ; запись длины вводимого сообщения в `EBX`
21
22 call sread                      ; вызов подпрограммы ввода сообщения
23 mov eax, buf1
24 call sprint
25
26 call quit                      ; вызов подпрограммы завершения

/home/mishanya4u/work/arch-pc/lab05/lab5-2-1.asm (1,1) | ft:asm | unix | utf-8Alt-g: bindings, Ctrl-g: h
```

Рис. 3.30: вывод lab5-lab5-1.asm

4. Создайте исполняемый файл и проверьте его работу.

```
mishanya4u@Legenda in ~ via C v13.2.1-gcc took 4h8m
└─ mc
└─ nasm -f elf lab5-2-1.asm
└─ ld -m elf_i386 -o lab5-2-1 lab5-2-1.o
└─ ./lab5-2-1
Введите строку:
Ван Сихэм Франклин О Нил Джон (Миша)
Ван Сихэм Франклин О Нил Джон (Миша)
```

Рис. 3.31: вывод lab5-lab5-1.asm

3.2 Листинг 5.1

```
;-----  
; 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  
  
;----- 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
```

3.3 Листинг 5.2

```
;-----  
; 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 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

```

4 Заключение

После работы с Midnight Commander, могу сказать что MC очень простой и интуитивно понятный интерфейс, который позволяет легко навигировать по файловой системе и выполнять нужные операции. Он также поддерживает работу с архивами, просмотр и редактирование текстовых файлов, а также поддерживает множество других функций, которые делают его очень полезным инструментом для системного администратора или разработчика.