

# **Отчёта по лабораторной работе №6**

**Лабораторная работа №6. Основы работы с Midnight Commander (mc).  
Структура программы на языке ассемблера NASM. Системные вызовы в  
ОС GNU Linux**

Акопян Сатеник Манвеловна

# Содержание

1	Цель работы	5
2	Теоретическое введение	6
3	Выполнение лабораторной работы	7
4	Задание для самостоятельной работы	13
	Список литературы	16

## Список иллюстраций

3.1	рисунок 1 . . . . .	7
3.2	рисунок 2 . . . . .	7
3.3	рисунок 3 . . . . .	8
3.4	рисунок 4 . . . . .	8
3.5	рисунок 5 . . . . .	9
3.6	рисунок 6 . . . . .	9
3.7	рисунок 7 . . . . .	9
3.8	рисунок 8 . . . . .	9
3.9	рисунок 9 . . . . .	10
3.10	рисунок 10 . . . . .	11
3.11	рисунок 11 . . . . .	11
3.12	рисунок 12 . . . . .	12
3.13	рисунок 13 . . . . .	12
3.14	рисунок 14 . . . . .	12
4.1	рисунок 15 . . . . .	14
4.2	рисунок 16 . . . . .	14
4.3	рисунок 17 . . . . .	15

## Список таблиц

# 1 Цель работы

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

## 2 Теоретическое 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

## 3 Выполнение лабораторной работы

1. Открываем Midnight Commander (рис. 3.1)

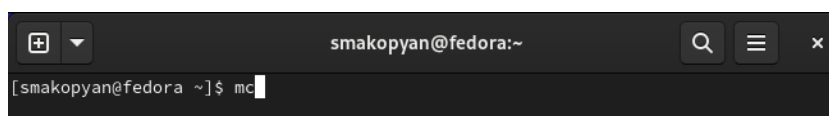


Рис. 3.1: рисунок 1

2. Переходим в каталог ~/work/arch-pc с помощью функциональной клавиши F7 создаём папку lab06 и переходим в созданный каталог (рис. 3.2)

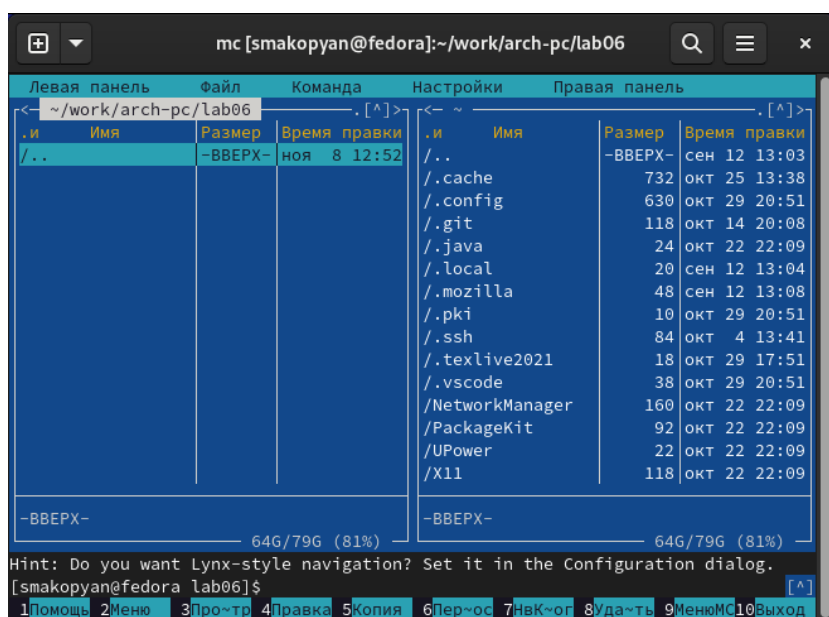


Рис. 3.2: рисунок 2

3. Пользуясь строкой ввода и командой touch создаём файл lab6-1.asm (рис. 3.3)

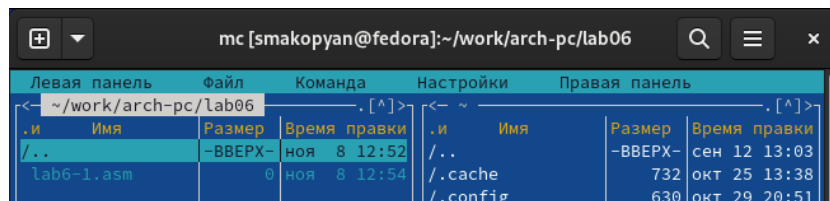


Рис. 3.3: рисунок 3

- С помощью функциональной клавиши F4 открываем файл lab6-1.asm, вводим текст программы сохраняем изменения и закрываем файл. (рис. 3.4)

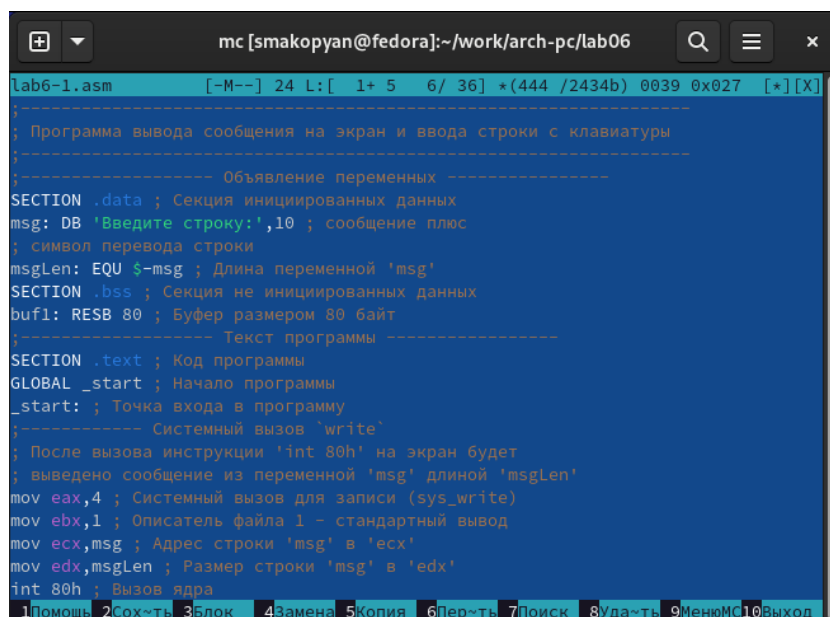
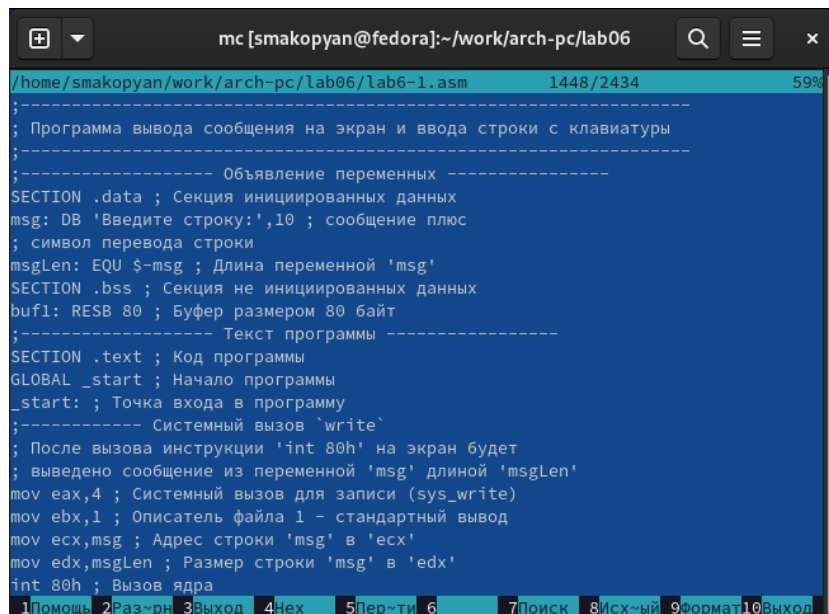


Рис. 3.4: рисунок 4

- С помощью функциональной клавиши F3 открываем файл lab6-1.asm для просмотра. Убеждаемся, что файл содержит текст программы. (рис. 3.5).

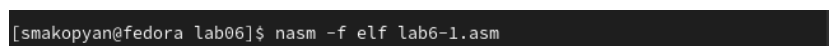




```
mc [smakopyan@fedora]:~/work/arch-pc/lab06
/home/smakopyan/work/arch-pc/lab06/lab6-1.asm 1448/2434 59%
;-----
; Программа вывода сообщения на экран и ввода строки с клавиатуры
;-----
;----- Объявление переменных -----
SECTION .data ; Секция инициализированных данных
msg: DB 'Введите строку:',10 ; 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 4Hex 5Пер-ти 6 7Поиск 8Исх-ый 9Формат10Выход
```

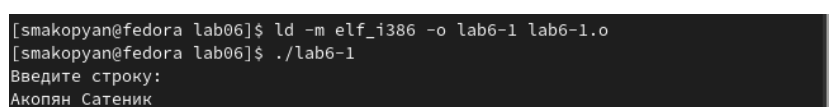
Рис. 3.5: рисунок 5

- Оттранслируем текст программы lab6-1.asm в объектный файл. Выполняем компоновку объектного файла и запускаем получившийся исполняемый файл. Программа выводит строку 'Введите строку:' и ожидает ввода с клавиатуры. 3.6) (рис. 3.7)



```
[smakopyan@fedora lab06]$ nasm -f elf lab6-1.asm
```

Рис. 3.6: рисунок 6



```
[smakopyan@fedora lab06]$ ld -m elf_i386 -o lab6-1 lab6-1.o
[smakopyan@fedora lab06]$ ./lab6-1
Введите строку:
Акопян Сатеник
```

Рис. 3.7: рисунок 7

- Скачиваем файл in\_out.asm со страницы курса в ТУИС. (рис. 3.8)



Рис. 3.8: рисунок 8

8. В одной из панелей mc откройте каталог с файлом lab6-1.asm. В другой панели каталог со скаченным файлом in\_out.asm (рис. 3.9)

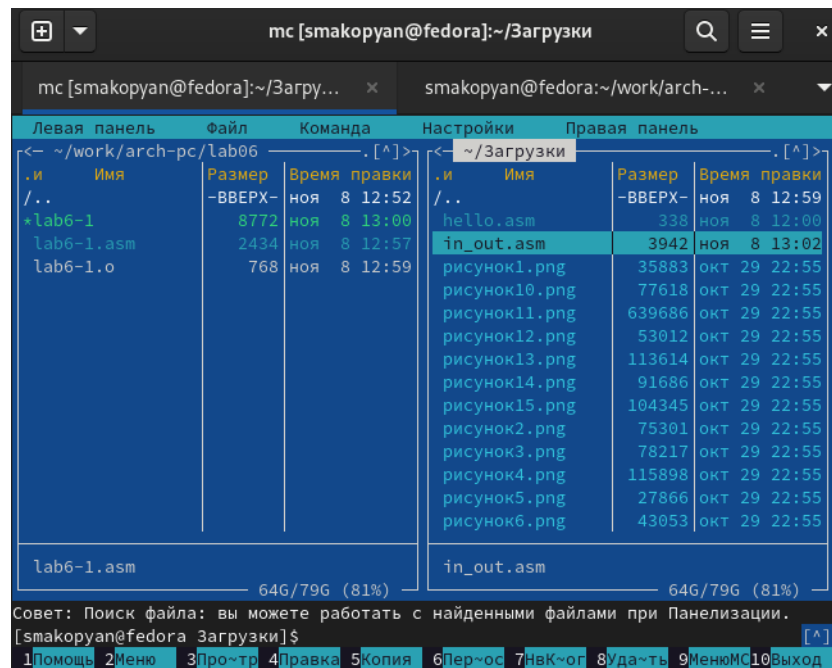


Рис. 3.9: рисунок 9

9. С помощью функциональной клавиши F6 создаём копию файла lab6-1.asm с именем lab6-2.asm.(рис 3.10)

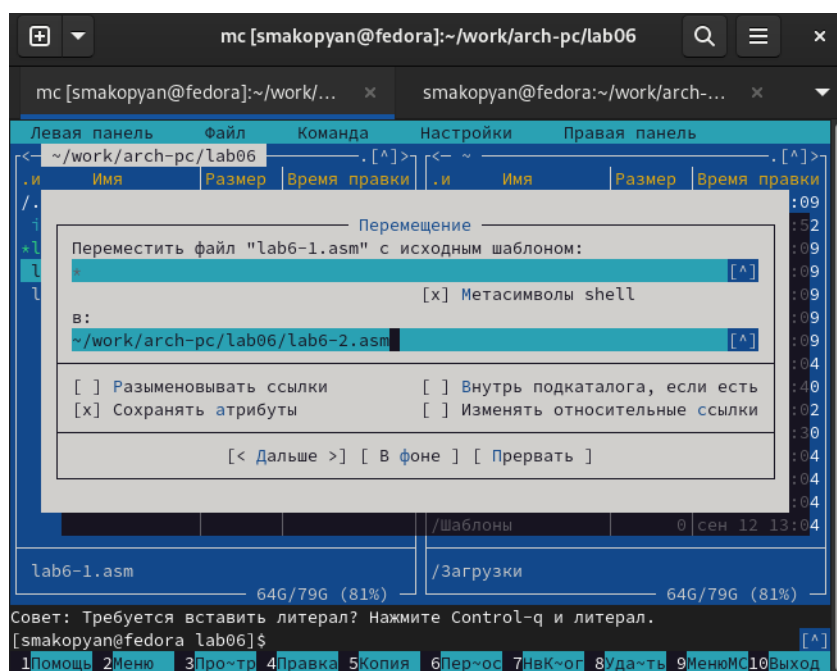


Рис. 3.10: рисунок 10

10. Исправим текст программы в файле lab6-2.asm с использованием подпрограмм из внешнего файла in\_out.asm (рис. 3.11)

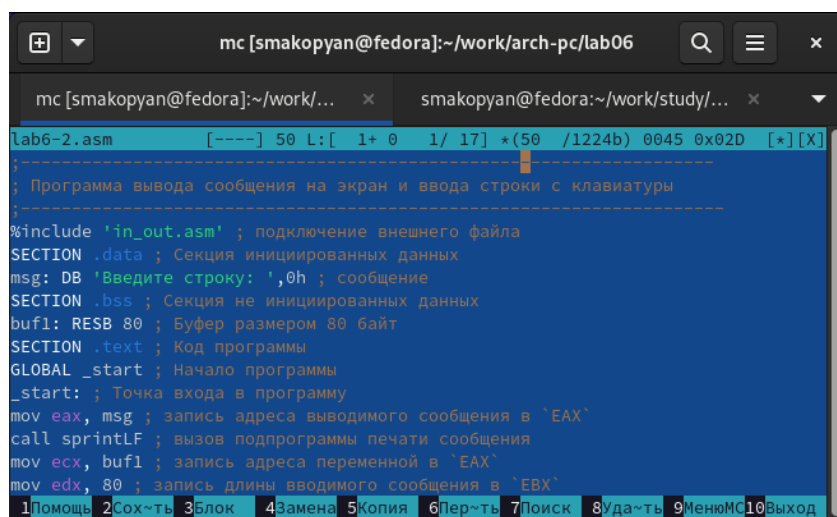


Рис. 3.11: рисунок 11

11. Создаем исполняемый файл и проверяем его работу (рис. 3.12)

```

[smakopyan@fedora lab06]$ nasm -f elf lab6-2.asm
[smakopyan@fedora lab06]$ ld -m elf_i386 -o lab6-2 lab6-2.o
[smakopyan@fedora lab06]$ ./lab6-2
Введите строку:
привет
[smakopyan@fedora lab06]$

```

Рис. 3.12: рисунок 12

12. Меняем подпрограмму `sprintLF` на `sprint`. Создаём исполняемый файл и проверяем его работу (рис. 3.13) (рис. 3.14). Отличие лишь в том, что с помощью подпрограммы `sprintLF` сообщение печатается с новой строки.

```

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
1Помощь 2Сох-ть 3Блок 4Замена 5Копия 6Пер-ть 7Поиск 8Уда-ть 9МенюМС10Выход

```

Рис. 3.13: рисунок 13

```

[smakopyan@fedora lab06]$ nasm -f elf lab6-2.asm
[smakopyan@fedora lab06]$ ld -m elf_i386 -o lab6-2 lab6-2.o
[smakopyan@fedora lab06]$ ./lab6-2
Введите строку: привет
[smakopyan@fedora lab06]$

```

Рис. 3.14: рисунок 14

## 4 Задание для самостоятельной работы

1. Создаем копию файла lab6-1.asm под названием lab6-1c.asm и вносим изменения так, как чтобы программа работала по алгоритму, описанному в лабораторной работе (рис. 4.1)

```

/home/smakopyan/work/arch-pc/lab06/lab6-1c.asm
;-----
; 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'
red: DB '-----' ,10
redLen: EQU $-red
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 ; 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
mov ebx,1
mov ecx,buf1
mov edx,redLen
int 80h
;----- 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. 4.1: рисунок 15

## 2. Создаем исполняемый файл и проверяем его работу (рис. 4.2)

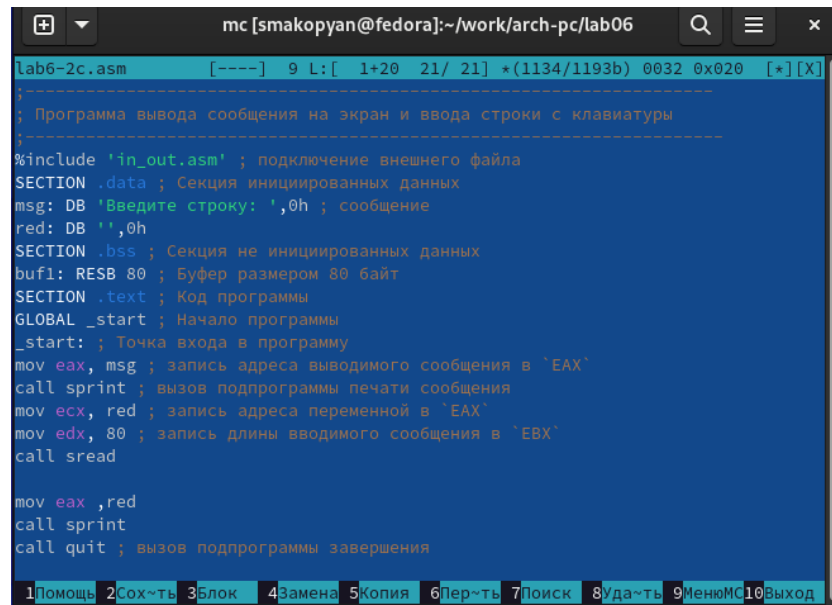
```

[smakopyan@fedora lab06]$ nasm -f elf lab6-1c.asm
lab6-1c.asm:28: warning: label alone on a line without a colon might be in error
[-w+label-orphan]
[smakopyan@fedora lab06]$ ld -m elf_i386 -o lab6-1c lab6-1c.o
[smakopyan@fedora lab06]$ ./lab6-1c
Введите строку:
Акопян
Акопян
[smakopyan@fedora lab06]$

```

Рис. 4.2: рисунок 16

3. Создаем копию файла lab6-2.asm под названием lab6-2c.asm и вносим изменения с использованием подпрограмм из внешнего файла in\_out.asm так, чтобы программа работала по алгоритму, описанному в лабораторной работе (рис. 4.3)



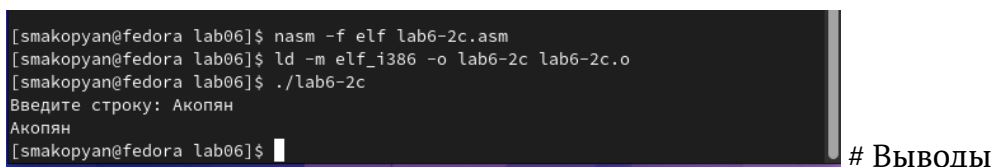
```
lab6-2c.asm  [----]  9  L: [ 1+20  21/ 21]  *(1134/1193b) 0032 0x020 [*][X]
;
; Программа вывода сообщения на экран и ввода строки с клавиатуры
;-----
#include 'in_out.asm' ; подключение внешнего файла
SECTION .data ; Секция инициализированных данных
msg: DB 'Введите строку: ',0h ; сообщение
red: DB '\n',0h
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, red ; запись адреса переменной в 'EAX'
mov edx, 80 ; запись длины вводимого сообщения в 'EBX'
call sread

mov eax, red
call sprint
call quit ; вызов подпрограммы завершения

1Помощь 2Сох-ть 3Блок 4Замена 5Копия 6Пер-ть 7Поиск 8Уда-ть 9МенюMC10Выход
```

Рис. 4.3: рисунок 17

4. Создаем исполняемый файл и проверяем его работу (рис. ??)



```
[smakopyan@fedora lab06]$ nasm -f elf lab6-2c.asm
[smakopyan@fedora lab06]$ ld -m elf_i386 -o lab6-2c lab6-2c.o
[smakopyan@fedora lab06]$ ./lab6-2c
Введите строку: Акопян
Акопян
[smakopyan@fedora lab06]$ # Выводы
```

В результате данной лабораторной работы я приобрела практические навыки работы в Midnight Commander, освоила инструкции языка ассемблера mov и int.

## **Список литературы**