

Отчет по лабораторной работе №5

Дисциплина: архитектура компьютера

Никуленков Степан Сергеевич

Содержание

1	Цель работы	5
2	Задание	6
3	Теоретическое введение	7
4	Выполнение лабараторной работы	9
4.1	Основы работы с msg	9
4.2	Структура программы на языке ассемблера NASM	11
4.3	Подключение внешнего файла	15
5	Выполнение заданий для самостоятельной работы	18
6	Выводы	19

Список иллюстраций

4.1	Создан новый каталог lab05	9
4.2	Создаю lab5-1.asm	10
4.3	Файл создан	11
4.4	Файл открыт	12
4.5	Код	13
4.6	Проверка	14
4.7	Исполнение программы	14
4.8	in_out.asm в папке downloads	15
4.9	Копирование в lab05	15
4.10	Переименование файла	16
4.11	Исполнение файла	17
4.12	Исполнение файла	17
5.1	Исполнени файлов	18

Список таблиц

1 Цель работы

Целью данной лабораторной работы является приобретение практических навыков работы в Midnight Commander, освоение инструкций языка ассемблера `mov` и `int`

2 Задание

1. Основы работы с mc
2. Структура программы на языке ассемблера NASM
3. Подключение внешнего файла
4. Выполнение заданий для самостоятельной работы

3 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. 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). Для объявления инициированных данных в секции .data используются директивы DB, DW, DD, DQ и DT, которые резервируют память и указывают, какие значения должны храниться в этой памяти: - DB (define byte) — определяет переменную размером в 1 байт; - DW (define word) — определяет переменную размером в 2 байта (слово); - DD (define double word) — определяет переменную размером в 4 байта (двойное слово); - DQ (define quad word) — определяет переменную размером в 8 байт (четырёх- рённое слово); - DT (define ten bytes) — определяет переменную размером в 10 байт. Директивы используются для объявления простых переменных и для объявления массивов. Для определения строк принято использовать директиву DB в связи с особенностями хранения данных в оперативной памяти. Инструкция языка ассемблера mov предназначена для дублирования данных источника в приёмнике. mov dst,src Здесь операнд dst — приёмник, а src — источник. В качестве операнда могут выступать регистры (register), ячейки памяти

(memory) и непосредственные значения (const). Инструкция языка ассемблера `int` предназначена для вызова прерывания с указанным номером. `int n` Здесь `n` — номер прерывания, принадлежащий диапазону 0–255. При программировании в Linux с использованием вызовов ядра `sys_calls` `n=80h` (принято задавать в шестнадцатеричной системе счисления).

4 Выполнение лабораторной работы

4.1 Основы работы с mc

Открываю Midnight Commander, введя в терминал `mc`. Перехожу в каталог `~/work/arch-pc` и создаю (F7) там новый каталог `lab05`.

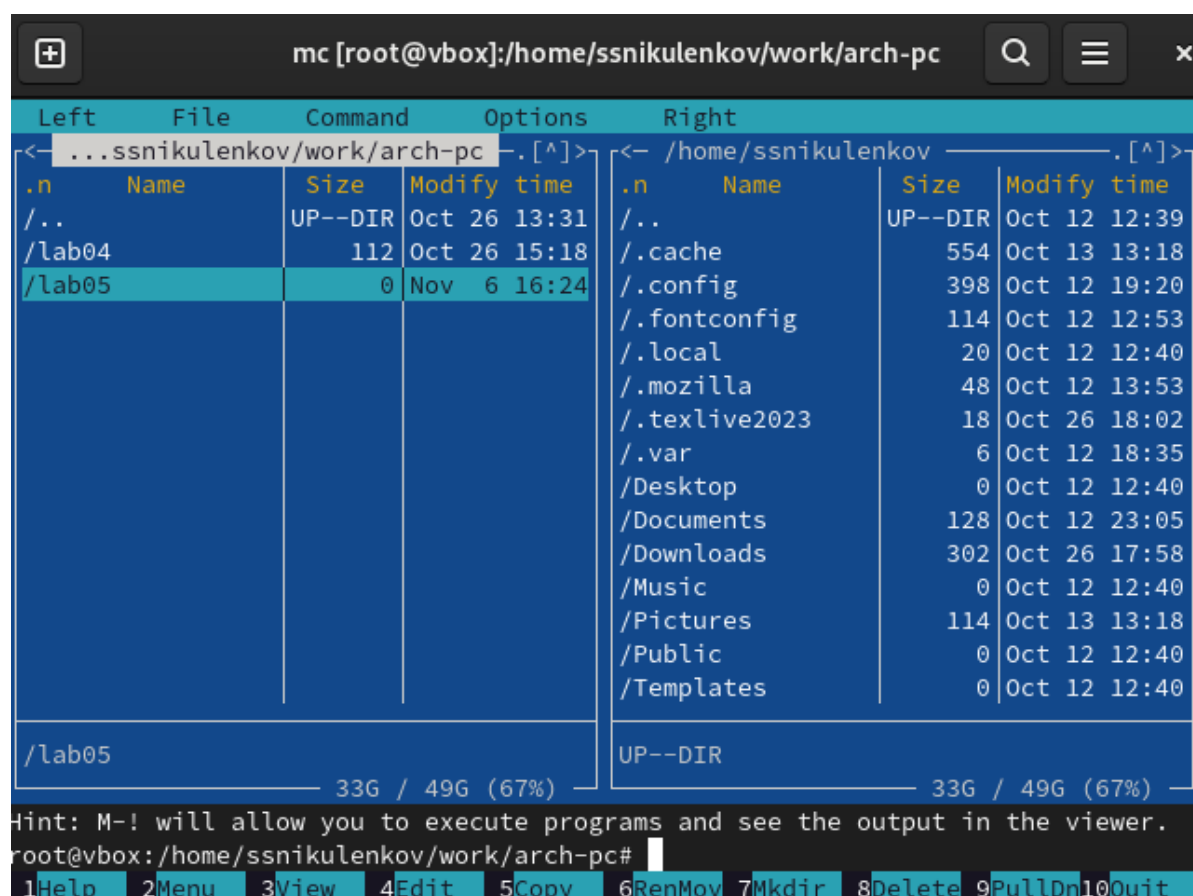


Рис. 4.1: Создан новый каталог lab05

Перехожу в каталог `lab05` и с помощью команды `touch` создаю файл `lab5-1.asm`.

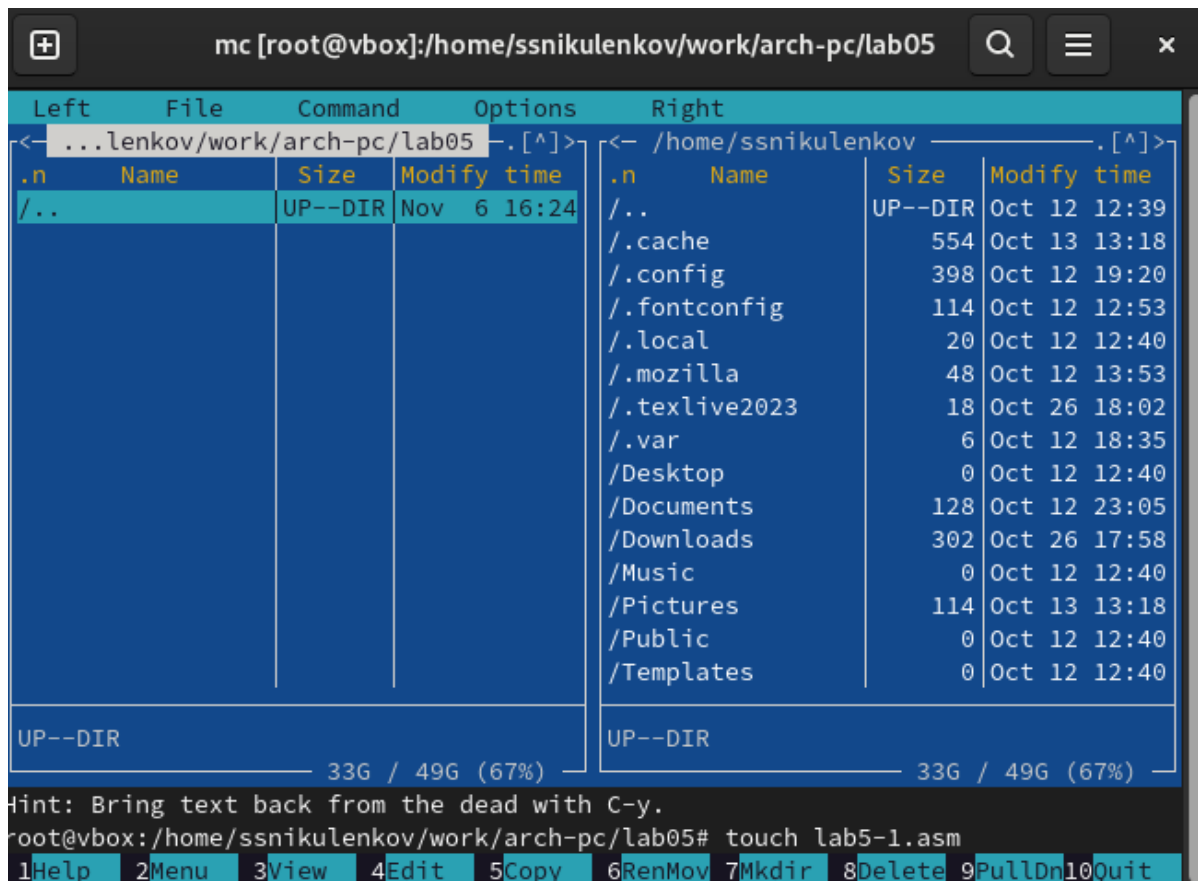


Рис. 4.2: Создаю lab5-1.asm

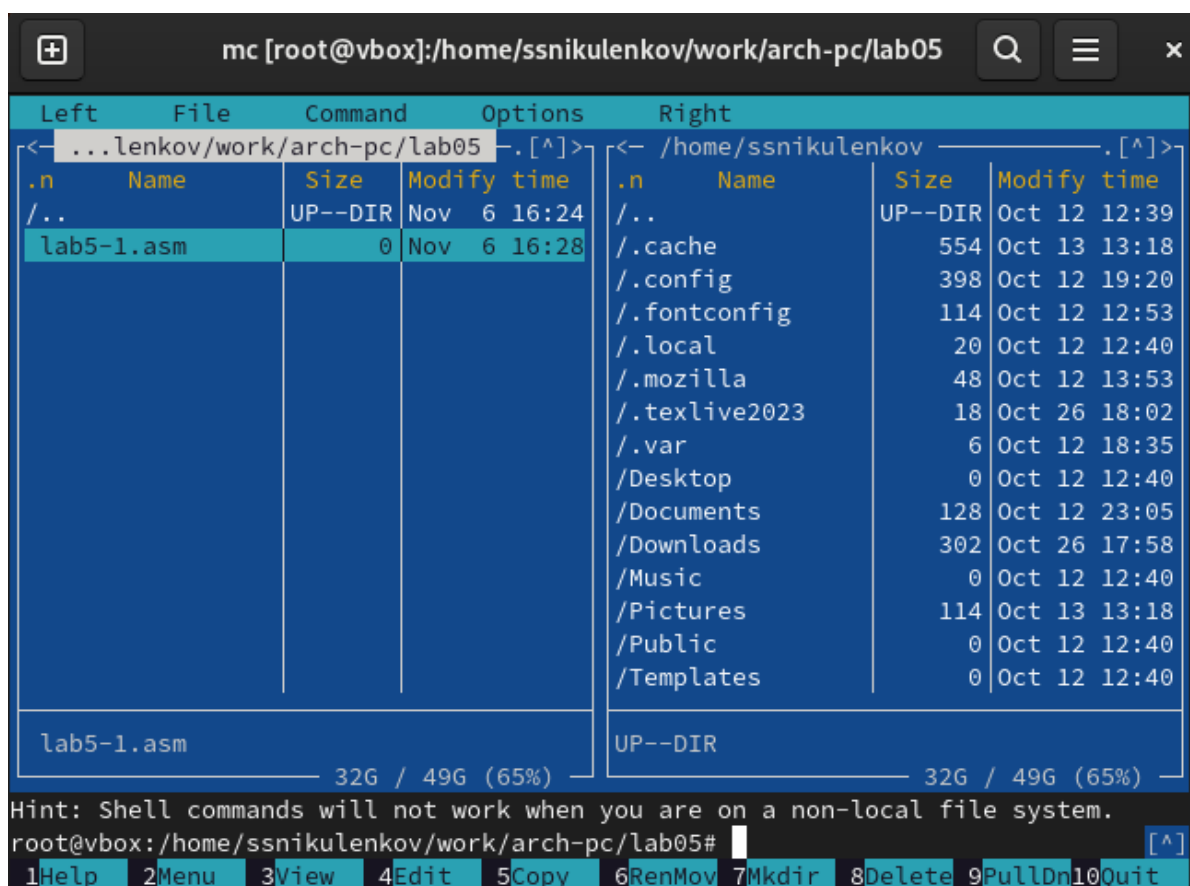


Рис. 4.3: Файл создан

4.2 Структура программы на языке ассемблера NASM

С помощью функциональной клавиши F4 открываю созданный файл для редактирования.

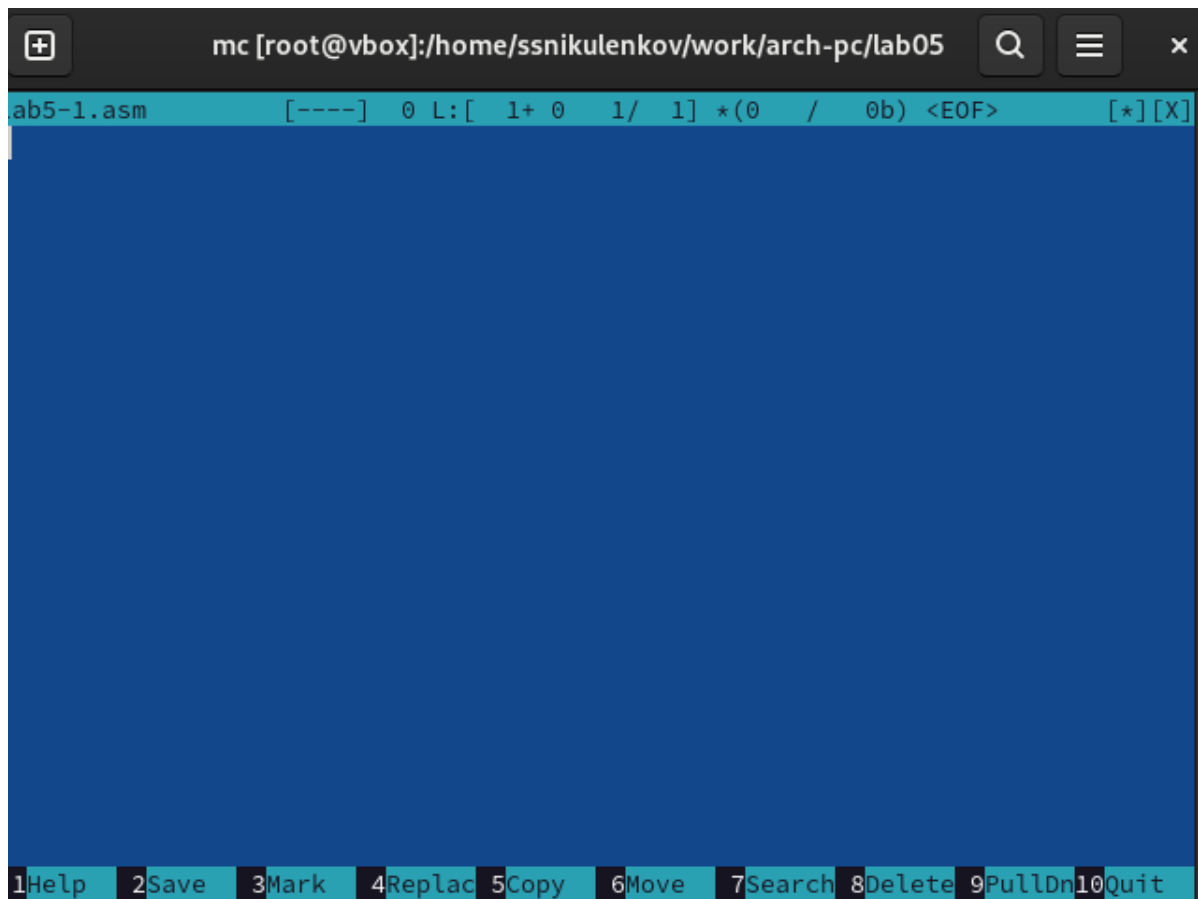
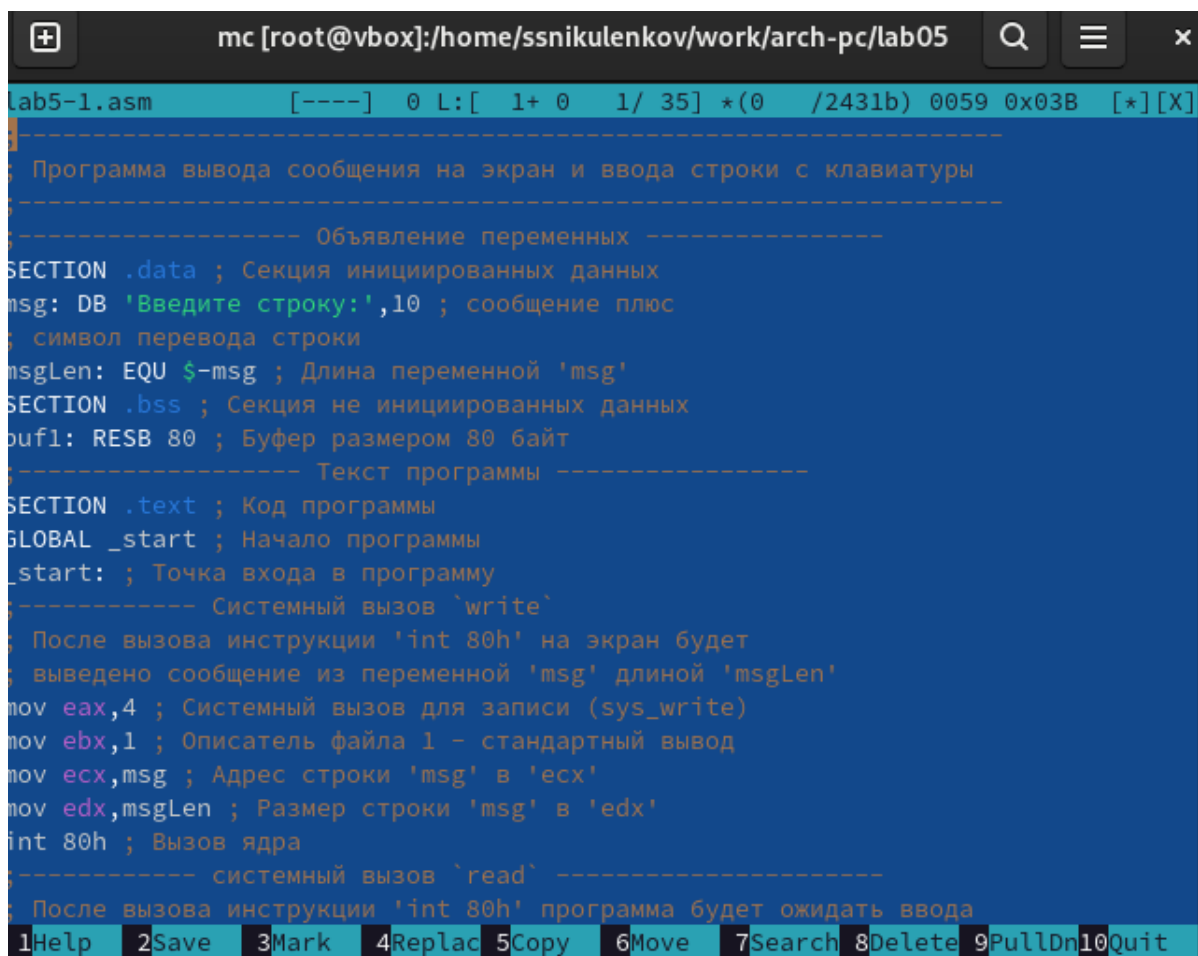


Рис. 4.4: Файл открыт

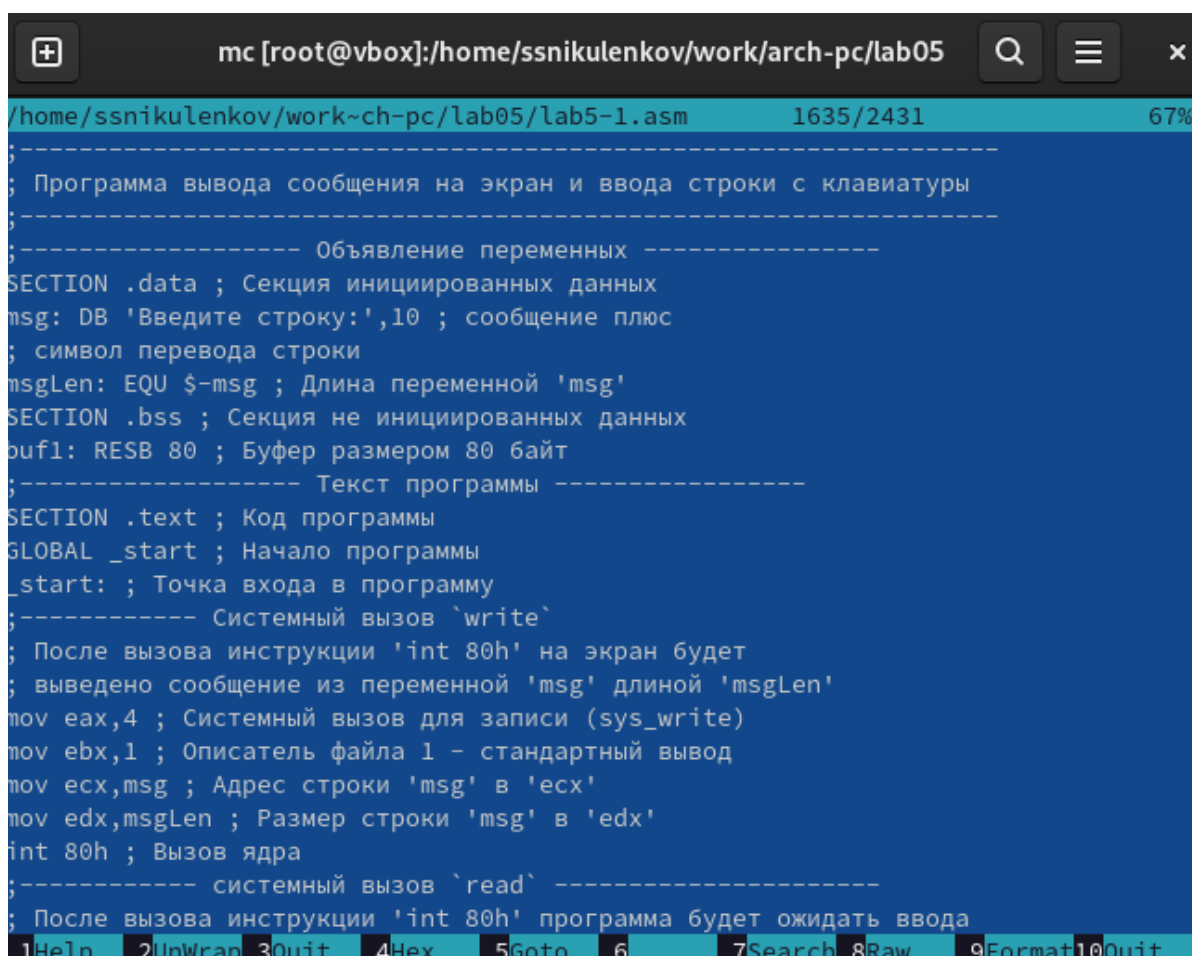
Пишу код программы.



```
mc [root@vbox]:/home/ssnikulenkov/work/arch-pc/lab05  Q  ≡  ×
lab5-1.asm  [----]  0 L:[ 1+ 0 1/ 35] *(0 /2431b) 0059 0x03B [*][X]
;
; 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
1Help 2Save 3Mark 4Replac 5Copy 6Move 7Search 8Delete 9PullDn10Quit
```

Рис. 4.5: Код

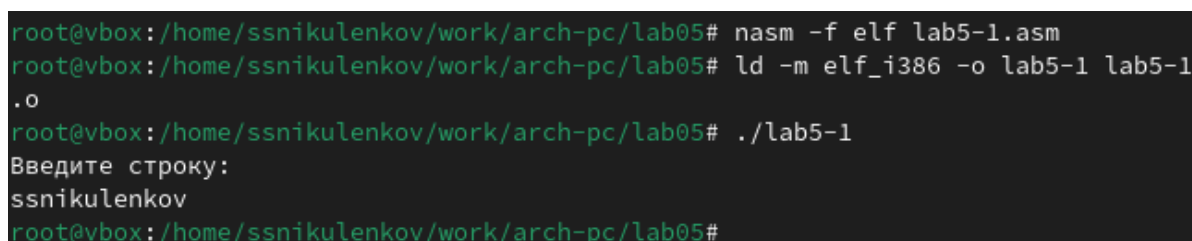
Открываю файл для просмотра.



```
mc [root@vbox]:/home/ssnikulenkov/work/arch-pc/lab05 1635/2431 67%
/home/ssnikulenkov/work~ch-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
1Help 2UnWrap 3Quit 4Hex 5Goto 6 7Search 8Raw 9Format10Quit
```

Рис. 4.6: Проверка

Транслирую текст программы файла в объектный файл командой `nasm -f elf lab5-1.asm`. Создался объектный файл `lab5-1.o`. Выполняю компоновку объектного файла с помощью команды `ld -m elf_i386 -o lab5-1 lab5-1.o`. Создался исполняемый файл `lab5-1`. Запускаю программу.



```
root@vbox:/home/ssnikulenkov/work/arch-pc/lab05# nasm -f elf lab5-1.asm
root@vbox:/home/ssnikulenkov/work/arch-pc/lab05# ld -m elf_i386 -o lab5-1 lab5-1.o
root@vbox:/home/ssnikulenkov/work/arch-pc/lab05# ./lab5-1
Введите строку:
ssnikulenkov
root@vbox:/home/ssnikulenkov/work/arch-pc/lab05#
```

Рис. 4.7: Исполнение программы

4.3 Подключение внешнего файла

Скачиваю файл in_out.asm со страницы курса в ТУИС. Он сохранился в каталог “Downloads”

```
root@vbox:/home/ssnikulenkov/Downloads# ls
16.jpg      pandoc-crossref-Linux      photo_2024-10-12_17-54-57.jpg
1.jpg       pandoc-crossref-Linux.tar.xz photo_2024-10-26_16-58-27.jpg
in_out.asm  photo_2024-10-12_17-54-31.jpg
root@vbox:/home/ssnikulenkov/Downloads#
```

Рис. 4.8: in_out.asm в папке downloads

С помощью функциональной клавиши F5 копирую файл in_out.asm из каталога Downloads в созданный каталог lab05.

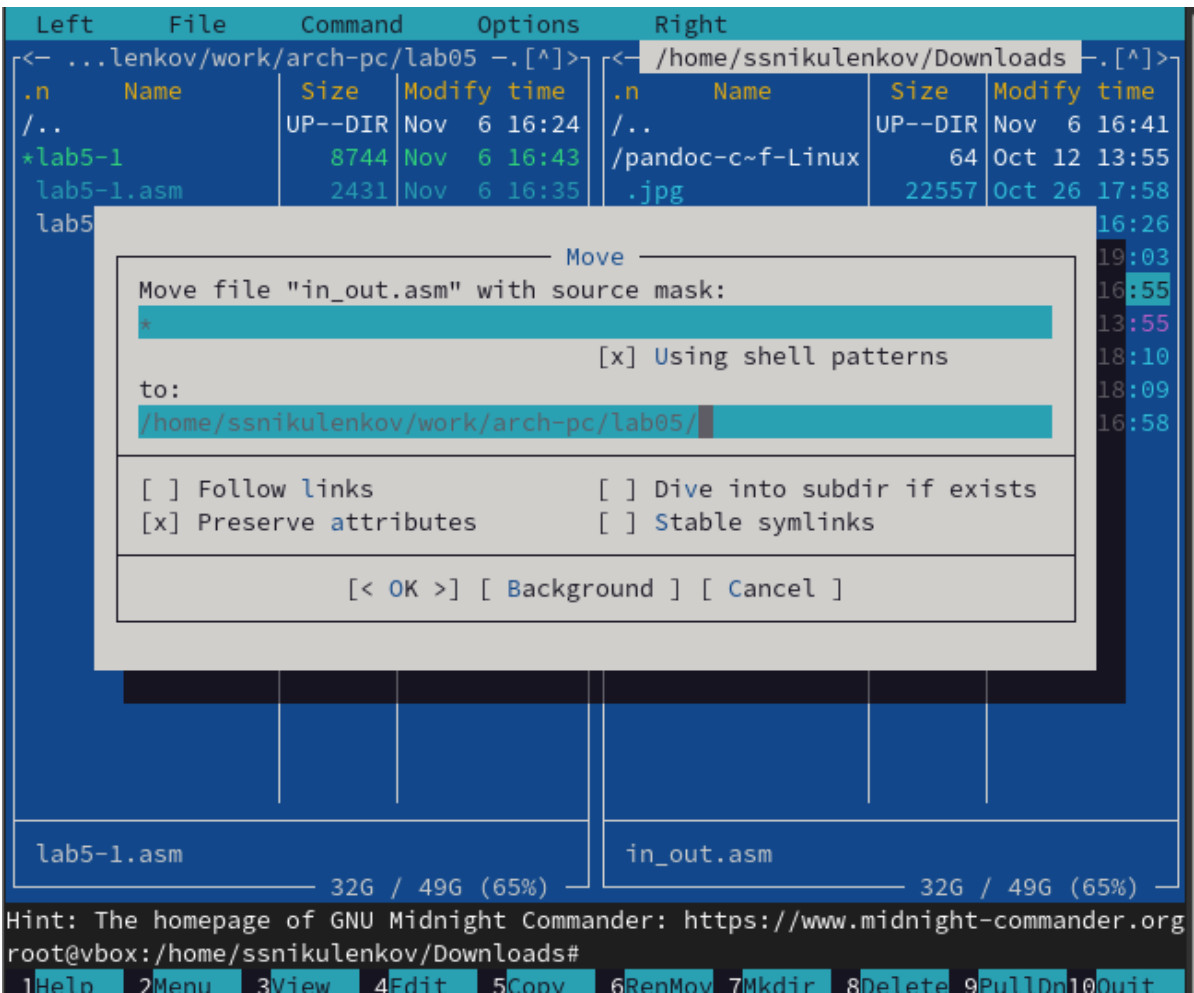


Рис. 4.9: Копирование в lab05

С помощью функциональной клавиши F5 копирую файл lab6-1 в тот же каталог, но с другим именем, для этого в появившемся окне mc прописываю имя для копии файла

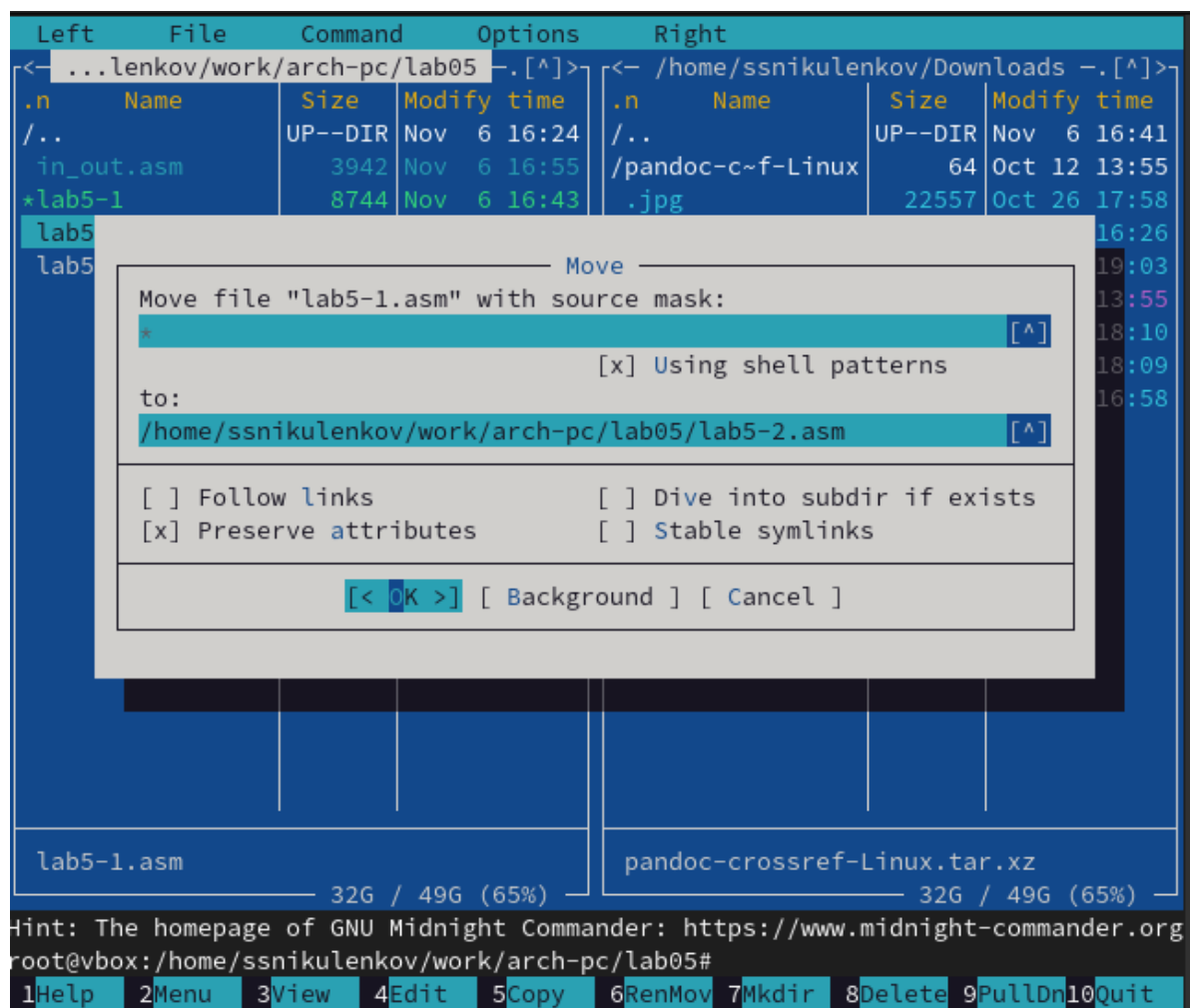


Рис. 4.10: Переименование файла

Изменяю содержимое файла lab5-2.asm.Создаю объектный файл и исполняю его.


```

root@vbox:/home/ssnikulenkov/work/arch-pc/lab05# nasm -f elf lab5-2.asm
root@vbox:/home/ssnikulenkov/work/arch-pc/lab05# ld -m elf_i386 -o lab5-2 lab5-2
.o
root@vbox:/home/ssnikulenkov/work/arch-pc/lab05# ./lab5-2.asm
bash: ./lab5-2.asm: Permission denied
root@vbox:/home/ssnikulenkov/work/arch-pc/lab05# ls
in_out.asm  lab5-1  lab5-1.o  lab5-2  lab5-2.asm  lab5-2.o
root@vbox:/home/ssnikulenkov/work/arch-pc/lab05# ./lab5-2
Введите строку:
ssnikulenkov
root@vbox:/home/ssnikulenkov/work/arch-pc/lab05#

```

Рис. 4.11: Исполнение файла

Снова изменяю файл lab5-2.asm. Изменяю в нем подпрограмму sprintLF на sprint. Создаю объектный файл и исполняю его.

```

root@vbox:/home/ssnikulenkov/work/arch-pc/lab05# nasm -f elf lab5-2.asm
root@vbox:/home/ssnikulenkov/work/arch-pc/lab05# ld -m elf_i386 -o lab5-2 lab5-2
.o
root@vbox:/home/ssnikulenkov/work/arch-pc/lab05# ./lab5-2
Введите строку: ssnikulenkov
root@vbox:/home/ssnikulenkov/work/arch-pc/lab05#

```

Рис. 4.12: Исполнение файла

Разница между первым исполняемым файлом lab6-2 и вторым lab6-2-2 в том, что запуск первого запрашивает ввод с новой строки, а программа, которая исполняется при запуске второго, запрашивает ввод без переноса на новую строку, потому что в этом заключается различие между подпрограммами sprintLF и sprint.

5 Выполнение заданий для самостоятельной работы

Создаю копии файлов lab5-1.asm и lab5-2.asm редактирую их.Создаю объектные файлы для каждого из них и исполняю их.

```
root@vbox:/home/ssnikulenkov/work/arch-pc/lab05# nasm -f elf lab5-1.asm
root@vbox:/home/ssnikulenkov/work/arch-pc/lab05# ld -m elf_i386 -o lab5-1 lab5-1
.o
root@vbox:/home/ssnikulenkov/work/arch-pc/lab05# ./lab5-1
Введите строку:
Никуленков
Никуленков
root@vbox:/home/ssnikulenkov/work/arch-pc/lab05# ^C
root@vbox:/home/ssnikulenkov/work/arch-pc/lab05# nasm -f elf lab5-2.asm
root@vbox:/home/ssnikulenkov/work/arch-pc/lab05# ld -m elf_i386 -o lab5-2 lab5-2
.o
root@vbox:/home/ssnikulenkov/work/arch-pc/lab05# ./lab5-2
Введите строку: Никуленков
Никуленков
root@vbox:/home/ssnikulenkov/work/arch-pc/lab05#
```

Рис. 5.1: Исполнени файлов

6 Выводы

При выполнении данной лабораторной работы я приобрел практические навыки работы в Midnight Commander, а также освоил инструкции языка ассемблера `mov` и `int`.