

РОССИЙСКИЙ УНИВЕРСИТЕТ ДРУЖБЫ НАРОДОВ

Факультет физико-математических и естественных наук

Кафедра прикладной информатики и теории вероятностей

ДОКЛАД

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

дисциплина: Архитектура компьютера

Студент: Ким И.Е.

Группа: НКАбд-03-25

№ ст. билета: 1032253496

МОСКВА

2025 г.

Содержание

1. Цель работы.....	4
2. Задание.....	5
3. Теоретическое введение.....	6
4. Выполнение лабораторной работы.....	7
4.1. Основы работы с Midnight Commander	7
4.2. Работа в NASM	10
4.3. Подключение внешнего файла	12
4.4. Задание для самостоятельной работы.	14
5. Выводы.....	18
6. Список литературы.....	19

Список иллюстраций

4.1. Открытие Midnight Commander.....	7
4.2. Интерфейс Midnight Commander.....	7
4.3. Открытый каталог arch-rc.....	8
4.4. Создание рабочего подкаталога	8
4.5. Создание файла в Midnight Commander	9
4.6. Редактирование файла в Midnight Commander.	10
4.7. Проверка сохранения сделанных изменений	11
4.8. Трансляция, компоновка и последующий запуск программы	11
4.9. Копирование файла в рабочий каталог	12
4.10. Создание копии файла в Midnight Commander	13
4.11. Изменение программы	13
4.12. Запуск измененной программы	13
4.13. Запуск низменной программы с другой подпрограммой	14
4.14. Редактирование копии.	14
4.15. Запуск своей программы	14
4.16. Редактирование копии.	16
4.17. Запуск своей программы	16

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 иницированных (известных во время компиляции) данных (SECTION .data) и секция неинициализированных данных (тех, под которые во время компиляции только отводится память, а значение присваивается входе выполнения программы) (SECTION .bss). Для объявления иницированных 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 (definebyte) определяет переменную размером в 1байт; -DW (defineword) определяет переменную размером в 2 байта (слово); -DD (definedoubleword) определяет переменную размером в 4 байта (двойное слово); -DQ (definequadword) определяет переменную размером в 8 байт (учетверенное слово); -DT (definetenbytes) 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 7
```

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.

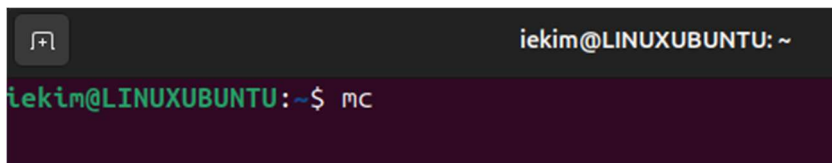
```
int n
```

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 Midnight Commander

Введя соответствующую команду в терминале (рис.4.1), я открываю Midnight Commander(рис.4.2).



```
iekim@LINUXUBUNTU: ~  
iekim@LINUXUBUNTU:~$ mc
```

Рис. 4.1: Открытие Midnight Commander

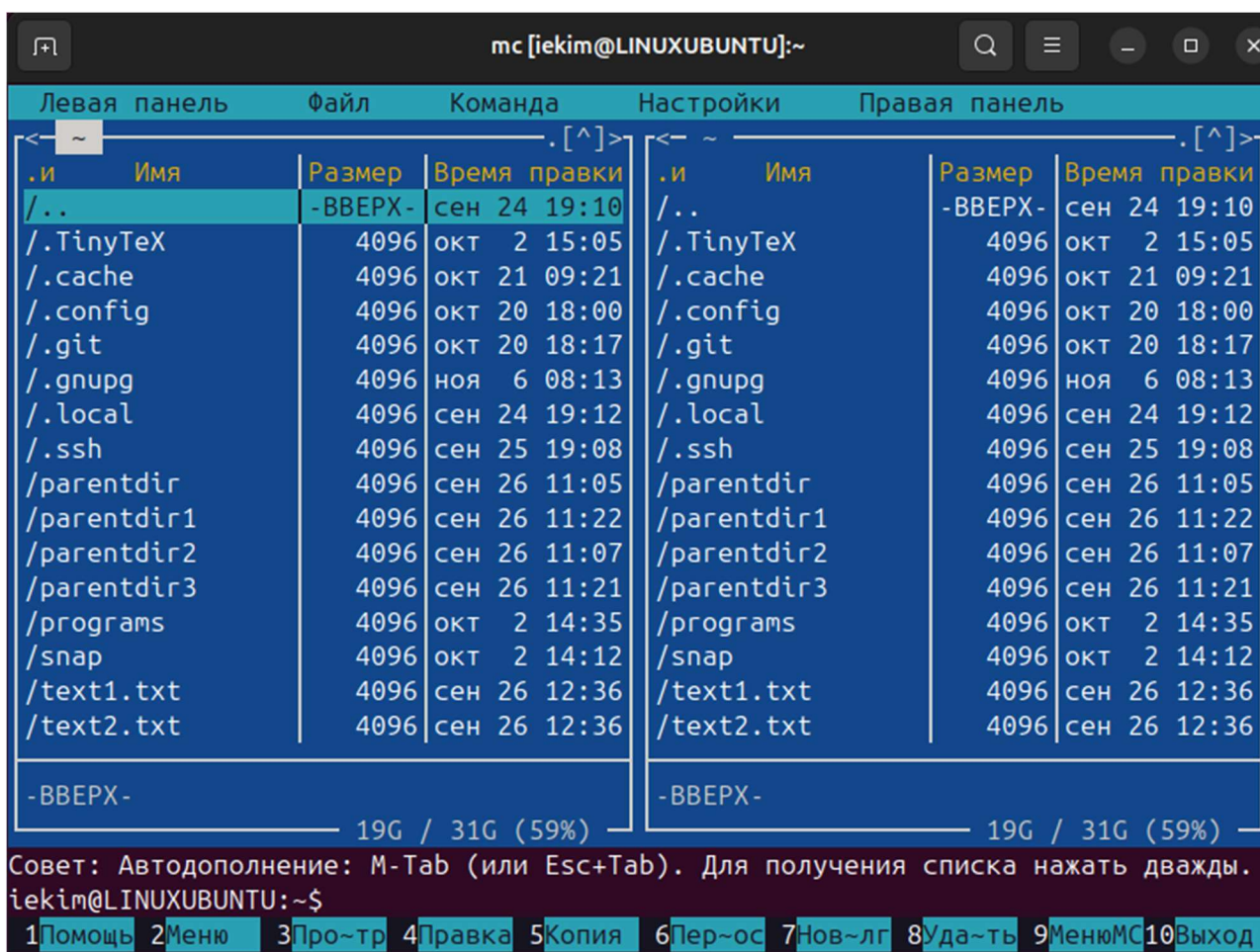


Рис. 4.2: Интерфейс Midnight Commander

Перехожу в созданный каталог в предыдущей лабораторной работе (рис.4.3).

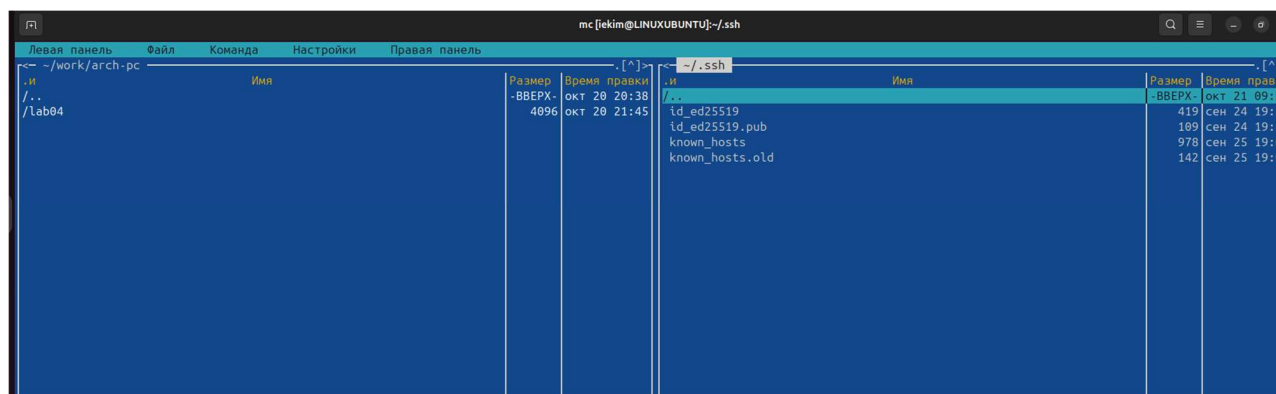


Рис. 4.3: Открытый каталог arch-pc

С помощью функциональной клавиши, я создаю подкаталог lab05, в котором буду работать (рис. 4.4).

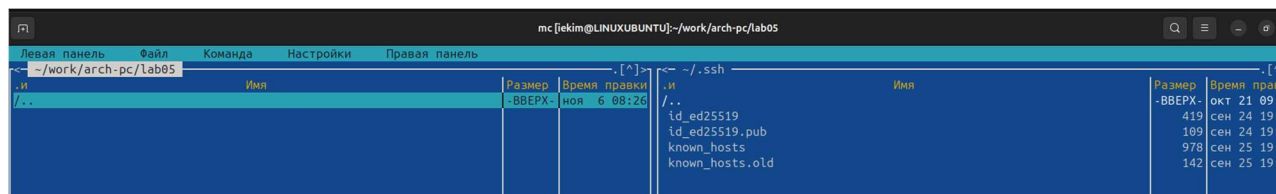


Рис. 4.4: Создание рабочего подкаталога

В строке ввода ввожу команду touch и создаю файл (рис.4.5).

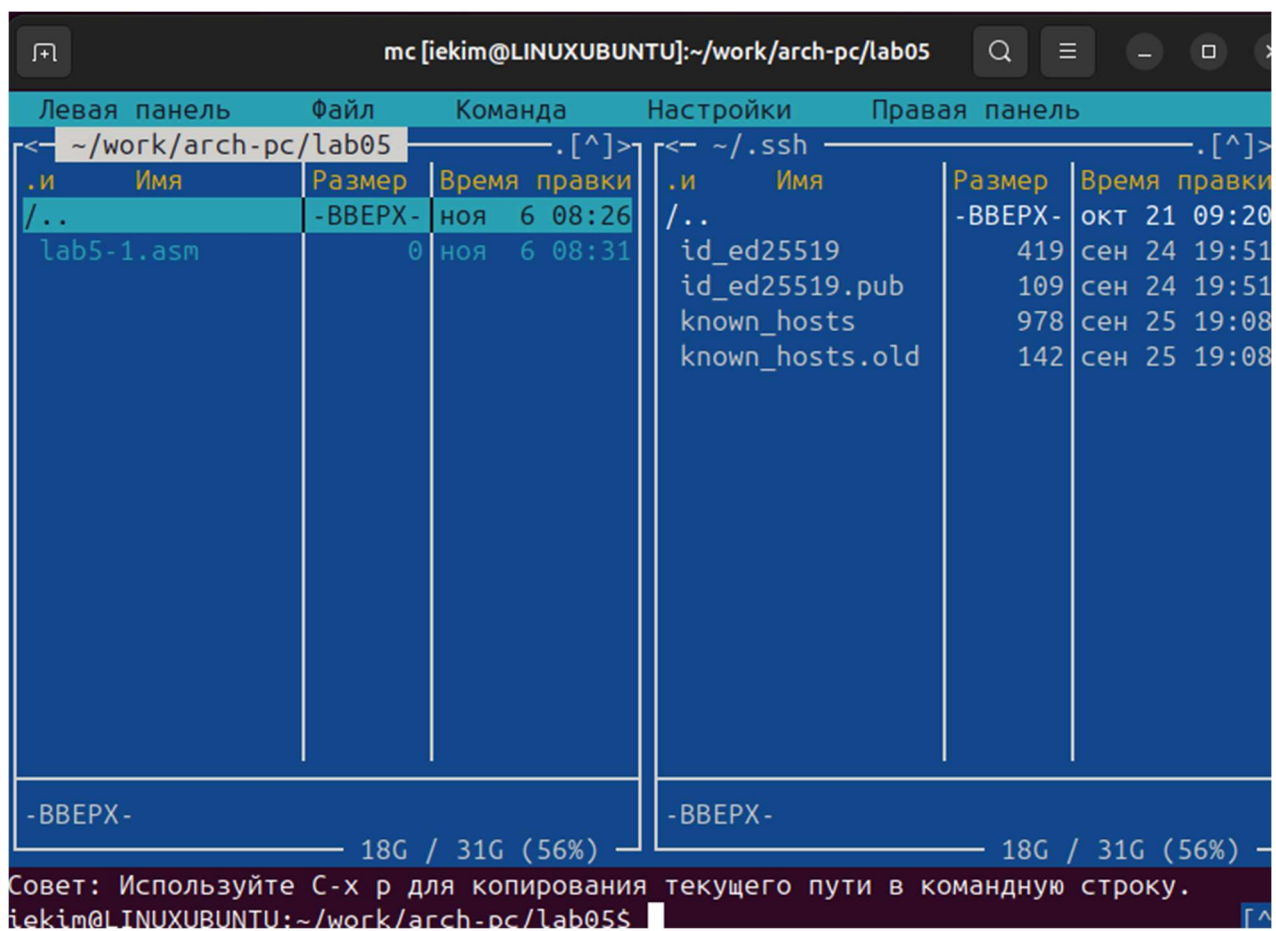


Рис. 4.5: Создание файла

4.2. Работа в NASM

С помощью F4 открываю только что созданный файл и вношу код с листинга (рис. 4.6).

```
SECTION .data
msg      db 'Введите строку:', 0Ah      ; сообщение + перевод строки
msgLen    equ $ - msg                  ; длина сообщения

SECTION .bss
buf1      resb 80                      ; буфер для ввода 80 байт

SECTION .text
GLOBAL _start

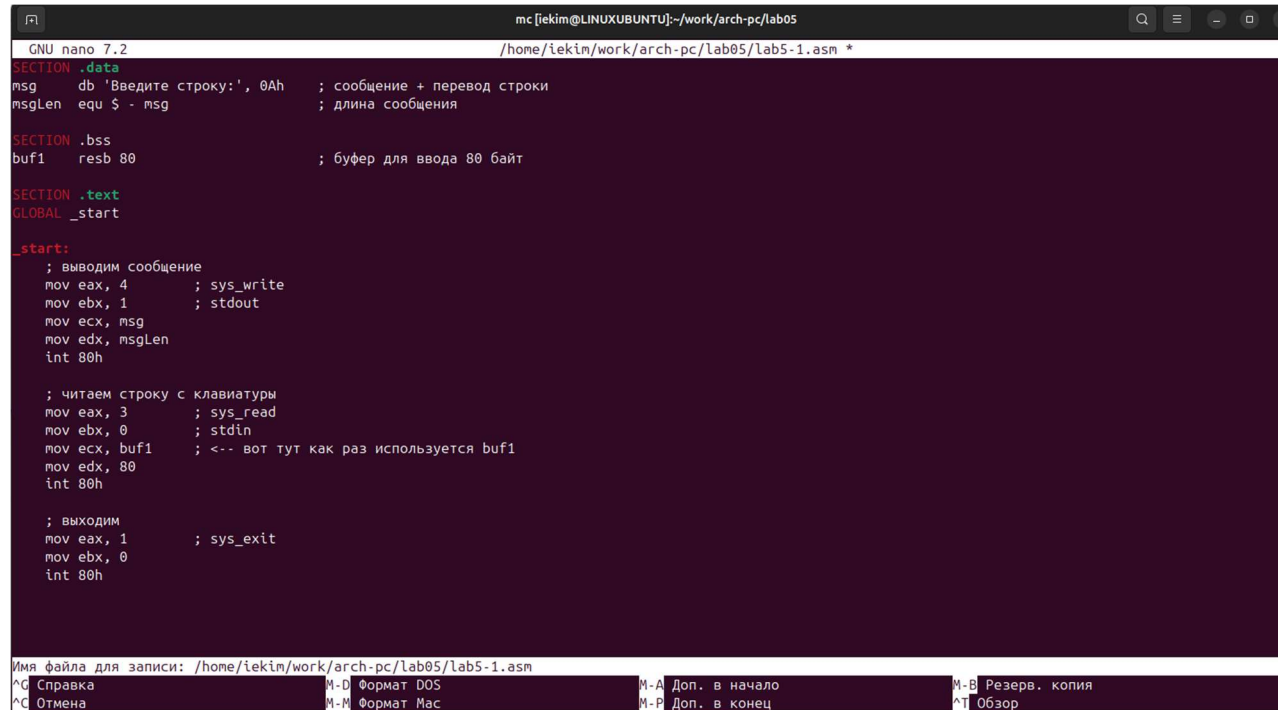
_start:
    ; выводим сообщение
    mov eax, 4                        ; sys_write
    mov ebx, 1                        ; stdout
    mov ecx, msg
    mov edx, msgLen
    int 80h

    ; читаем строку с клавиатуры
    mov eax, 3                        ; sys_read
    mov ebx, 0                        ; stdin
    mov ecx, buf1                    ; <-- вот тут как раз используется buf1
    mov edx, 80
    int 80h

    ; выходим
    mov eax, 1                        ; sys_exit
    mov ebx, 0
    int 80h
```

Рис. 4.6: Редактирование файла в Midnight Commander

Проверяю сохраненные изменения с помощью клавиши F3 (рис.4.7).



```
GNU nano 7.2 /home/iekim/work/arch-pc/lab05/lab5-1.asm *
SECTION .data
msg db 'Введите строку:', 0Ah ; сообщение + перевод строки
msgLen equ $ - msg ; длина сообщения

SECTION .bss
buf1 resb 80 ; буфер для ввода 80 байт

SECTION .text
GLOBAL _start
_start:
; выводим сообщение
mov eax, 4 ; sys_write
mov ebx, 1 ; stdout
mov ecx, msg
mov edx, msgLen
int 80h

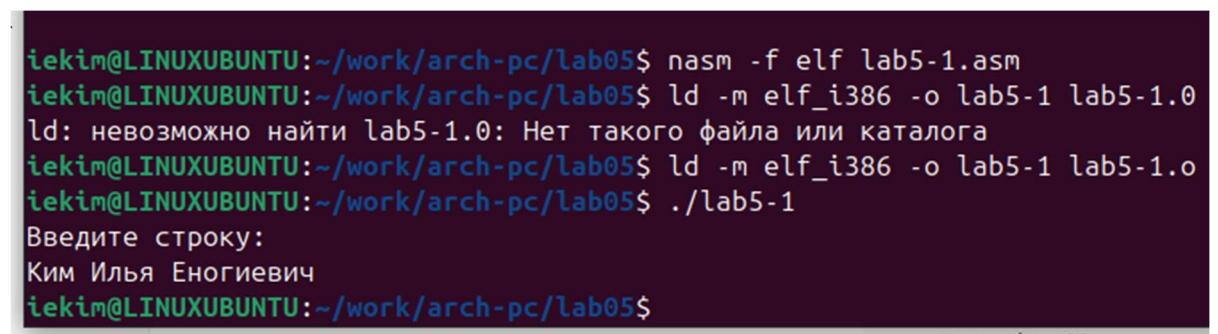
; читаем строку с клавиатуры
mov eax, 3 ; sys_read
mov ebx, 0 ; stdin
mov ecx, buf1 ; <-- вот тут как раз используется buf1
mov edx, 80
int 80h

; выходим
mov eax, 1 ; sys_exit
mov ebx, 0
int 80h

Имя файла для записи: /home/iekim/work/arch-pc/lab05/lab5-1.asm
^G Справка М-D Формат DOS М-A Доп. в начало М-B Резерв. копия
^C Отмена М-M Формат Mac М-R Доп. в конец ^T Обзор
```

Рис. 4.7: Проверка сохранения сделанных изменений

Транслирую и компоную измененный файл, запускаю (рис.4.8).



```
iekim@LINUXUBUNTU:~/work/arch-pc/lab05$ nasm -f elf lab5-1.asm
iekim@LINUXUBUNTU:~/work/arch-pc/lab05$ ld -m elf_i386 -o lab5-1 lab5-1.o
ld: невозможно найти lab5-1.0: Нет такого файла или каталога
iekim@LINUXUBUNTU:~/work/arch-pc/lab05$ ld -m elf_i386 -o lab5-1 lab5-1.o
iekim@LINUXUBUNTU:~/work/arch-pc/lab05$ ./lab5-1
Введите строку:
Ким Илья Еногиевич
iekim@LINUXUBUNTU:~/work/arch-pc/lab05$
```

Рис. 4.8: Трансляция, компоновка и последующий запуск

4.3. Подключение внешнего файла

Скачанный с ТУИС файл сохраняю в общую папку на своем компьютере, на виртуальной машине в интерфейсе Midnight Commander перехожу в директорию общей папки, копирую файл в рабочий подкаталог. (рис.4.9).

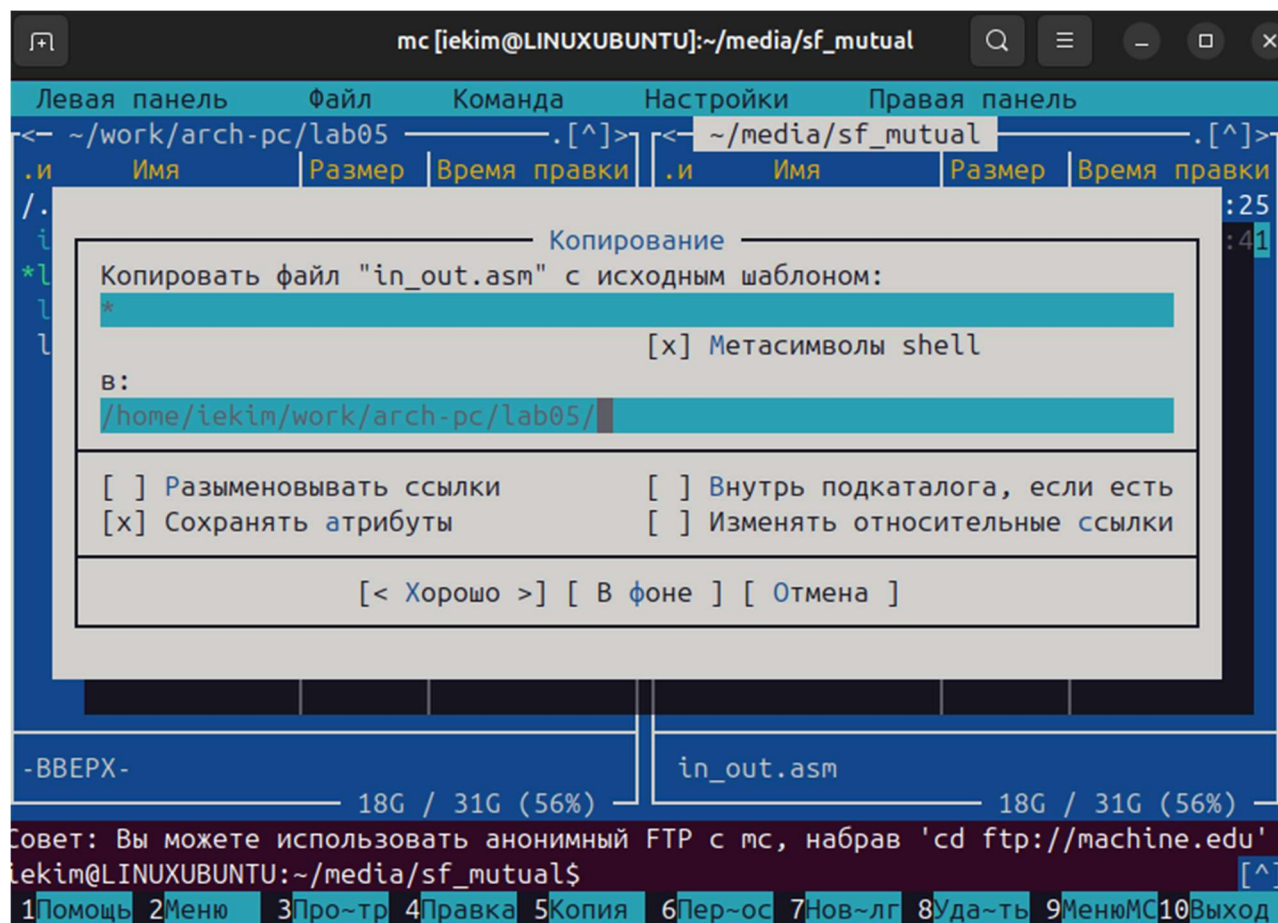


Рис. 4.9: Копирование файла в рабочий каталог

Создаю копию файла для последующей работы с ним (рис. 4.10).

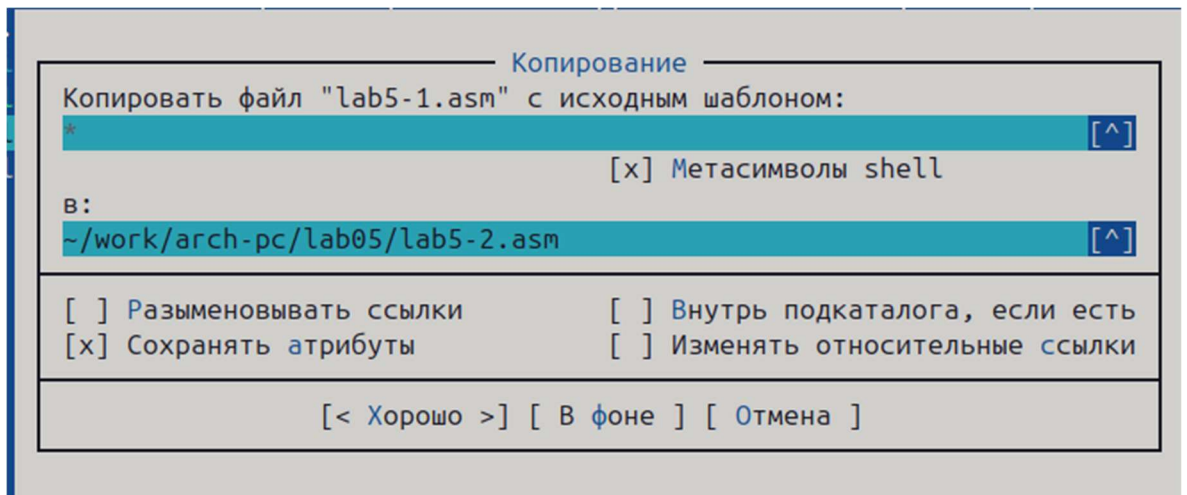


Рис. 4.10: Создание копии файла в Midnight Commander

В копии файла подключаю подпрограмм из подключенного файла (рис. 4.11).

```
GNU nano 7.2 /home/iekim/work/arch-pc/lab05/lab5-2.asm

SECTION .data
msg     DB 'Введите строку: ', 0h
msgLen  EQU $-msg

SECTION .bss
buf1    RESB 80

SECTION .text
GLOBAL _start
_start:
    mov eax, msg
    call sprintf

    mov ecx, buf1
    mov edx, 80

    call sread
    call quit
```

Рис. 4.11: Изменение программы

Транслирую, компоную и запускаю программу с подключенным файлом (рис. 4.12).

```
iekim@LINUXUBUNTU:~/work/arch-pc/lab05$ ./lab5-1
Введите строку:
Ким Илья Еногиевич
iekim@LINUXUBUNTU:~/work/arch-pc/lab05$
```

Рис. 4.12: Запуск измененной программы

Редактирую файл и заменяю в нем подпрограмму `sprintLF` на `sprint`. Разница подпрограмм в том, что вторая вызывает ввод на той же строке (рис. 4.13).

4.4. Задание для самостоятельной работы

```
iekim@LINUXUBUNTU:~/work/arch-pc/lab05$ nasm -f elf lab5-2.asm
iekim@LINUXUBUNTU:~/work/arch-pc/lab05$ ld -m elf_i386 -o lab5-22 lab5-2.o
iekim@LINUXUBUNTU:~/work/arch-pc/lab05$ ./lab5-22
Введите строку:
Ким Илья Еногиевич
```

Рис. 4.13: Запуск измененной программы с другой подпрограммой

Создаю копию `lab5-1.asm`, редактирую так, чтобы в конце выводилась введенная

мною строка с клавиатуры (рис. 4.14).

```
GNU nano 7.2 /home/iekim/work/arch-pc/lab05/lab5-3.asm
SECTION .data
msg DB 'Введите строку:', 10
msgLen EQU $-msg

SECTION .bss
buf1 RESB 80

SECTION .text
GLOBAL _start

_start:
    mov eax, 4        ; sys_write
    mov ebx, 1        ; stdout
    mov ecx, msg      ; адрес строки
    mov edx, msgLen   ; длина строки
    int 80h          ; вывод приглашения

    mov eax, 3        ; sys_read
    mov ebx, 0        ; stdin
    mov ecx, buf1     ; куда сохранять
    mov edx, 80       ; максимум символов
    int 80h          ; чтение строки

    mov eax, 4        ; sys_write
    mov ebx, 1        ; вывести введенную строку
    mov ecx, buf1
    mov edx, 80
    int 80h

    mov eax, 1        ; sys_exit
    mov ebx, 0
    int 80h
```

Рис. 4.14: Редактирование копии

Транслирую, компоную и запускаю свою программу (рис. 4.15).

```
iekim@LINUXUBUNTU:~/work/arch-pc/lab05$ nasm -f elf lab5-1copy.asm
iekim@LINUXUBUNTU:~/work/arch-pc/lab05$ ld -m elf_i386 -o lab5-1copy lab5-1copy.o
iekim@LINUXUBUNTU:~/work/arch-pc/lab05$ ./lab5-1copy
Введите строку:
Ким Илья Еногиевич
Ким Илья Еногиевич
iekim@LINUXUBUNTU:~/work/arch-pc/lab05$
```

Код прикладываю

SECTION .data

msg: **DB** 'Введите строку:',10

msgLen: **EQU** \$-msg

SECTION .bss

buf1: **RESB** 80

SECTION .text

GLOBAL _start

_start:

mov **eax**, 4

mov **ebx**, 1

mov **ecx**, msg

mov **edx**, msgLen

int 80h

mov **eax**, 3

mov **ebx**, 0

mov **ecx**, buf1

mov **edx**, 80

int 80h

mov **eax**, 4

mov **ebx**, 1

mov **ecx**, buf1

mov **edx**, buf1

int 80h

mov **eax**, 1

19

mov **ebx**, 0

int 80h

Создаю копию lab5-2.asm, редактирую так, чтобы в конце выводилась введенная

мною строка с клавиатуры (рис. 4.16).

```
SECTION .data
    msg      DB 'Введите строку: ', 0Ah      ; сообщение + перевод строки
    msgLen   EQU $ - msg                    ; длина сообщения

SECTION .bss
    buf      RESB 80                        ; буфер для ввода

SECTION .text
    GLOBAL _start

_start:
    ; выводим приглашение
    mov eax, 4          ; sys_write
    mov ebx, 1          ; stdout
    mov ecx, msg         ; адрес сообщения
    mov edx, msgLen      ; длина сообщения
    int 80h

    ; читаем строку с клавиатуры
    mov eax, 3          ; sys_read
```

Рис. 4.16: Редактирование копии

Транслирую, компоную и запускаю свою программу (рис. 4.17).

```
iekim@LINUXUBUNTU:~/work/arch-pc/lab05$ nasm -f elf lab5-2copy.asm
iekim@LINUXUBUNTU:~/work/arch-pc/lab05$ ld -m elf_i386 -o lab5-2copy lab5-2copy.o
iekim@LINUXUBUNTU:~/work/arch-pc/lab05$ ./lab5-2copy
Введите строку:
Ким Илья Еногиевич
Ким Илья Еногиевич
iekim@LINUXUBUNTU:~/work/arch-pc/lab05$
```

Рис. 4.17: Запуск своей программы

Код прикладываю:

```
%include 'in_out.asm'

SECTION .data
msg: DB 'Введите строку: ', 0h
msgLen: EQU $-msg

SECTION .bss
buf1: RESB 80

SECTION .text
GLOBAL _start
_start:
mov eax, msg
call sprint
mov ecx, buf1
mov edx, 80
call sread
mov eax, 4
mov ebx, 1
mov ecx, buf1
int 80h
call quit
```

5. Выводы

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

Список литературы

1. Пример выполнения лабораторной работы
2. Курс на ТУИС
3. Лабораторная работа №5
4. Программирование на языке ассемблера NASM Столяров А. В.