

# РОССИЙСКИЙ УНИВЕРСИТЕТ ДРУЖБЫ НАРОДОВ

Факультет физико-математических и естественных наук

Кафедра прикладной информатики и теории вероятностей

## ОТЧЕТ

### ПО ЛАБОРАТОРНОЙ РАБОТЕ № 5

дисциплина: Архитектура компьютера

Студент: Безлепкина Татьяна Игоревна

Группа: НКАБД-01-25

МОСКВА

2025 г.

## Оглавление

1	Цель
работы.....	
2	Теоретическое
введение.....	
2.1 Основы работы с Midnight Commander.....	
2.2 Структура программы на языке ассемблера NASM.....	
2.3	Элементы
программирования.....	
2.3.1 Описание инструкции mov.....	
2.3.2 Описание инструкции int.....	
2.3.3 Системные вызовы для обеспечения диалога с пользователем.....	
3	Выполнение лабораторной
работы.....	
3.1 Работа с ms.....	
3.2 Подключение внешнего файла in_out.asm .....	
4 Задания для самостоятельной работы.....	
5 Вывод .....	
6 Список литературы.....	

## **1 Цель работы**

Освоение практических навыков работы в среде Midnight Commander и изучение ассемблерных инструкций `mov` и `int`.

## 2 Теоретическое введение

### 2.1 Основы работы с Midnight Commander

Midnight Commander (mc) — это файловый менеджер, предоставляющий удобный интерфейс для навигации по файловой системе и выполнения основных операций с файлами. Запуск программы осуществляется вводом команды mc в командной строке с последующим нажатием Enter. Для удобства пользователя в интерфейсе используются функциональные клавиши F1-F10, связанные с наиболее распространёнными операциями.

Функцио- нальные клавиши	Выполняемое действие
F1	вызов контекстно-зависимой подсказки
F2	вызов меню, созданного пользователем
F3	просмотр файла, на который указывает подсветка в активной панели
F4	вызов встроенного редактора для файла, на который указывает подсветка в активной панели

Функцио- нальные клавиши	Выполняемое действие
F5	копирование файла или группы отмеченных файлов из каталога, отображаемого в активной панели, в каталог, отображаемый на второй панели
F6	перенос файла или группы отмеченных файлов из каталога, отображаемого в активной панели, в каталог, отображаемый на второй панели
F7	создание подкаталога в каталоге, отображаемом в активной панели
F8	удаление файла (подкаталога) или группы отмеченных файлов
F9	вызов основного меню программы
F10	выход из программы

Рис 2.1.1 - Функциональные клавиши Midnight Commander

## 2.2 Структура программы на языке ассемблера NASM

Программа на NASM состоит из трёх секций:

SECTION .text - исполняемый код

SECTION .data - инициализированные данные (директивы DB, DW, DD и др.)

SECTION .bss - неинициализированные данные (директивы resb, resw и др.)

Директивы DB, DW, DD резервируют память и задают значения в секции .data.

Для строк обычно используется DB из-за особенностей хранения в памяти.

Директивы resb, resw, resd в секции .bss только резервируют память без инициализации значений.

## **2.3 Элементы программирования**

### **2.3.1 Описание инструкции mov**

MOV - инструкция перемещения данных, которая копирует значение из источника в приёмнику.

В качестве операндов могут использоваться:

- регистры (register).
- ячейки памяти (memory).
- непосредственные значения (const).

Ограничения:

- Запрещена прямая пересылка между двумя ячейками памяти
  - Для такой операции требуется две команды MOV через промежуточный регистр.
  - Размеры операндов источника и приёмника должны быть одинаковыми.
- Некорректные примеры, которые вызовут ошибку:
- Прямое копирование из памяти в память.
  - Использование операндов разного размера.
  - Несогласованные типы данных между источником и приёмником.

### **2.3.2 Описание инструкции `int`**

Инструкция языка ассемблера `int` предназначена для вызова прерывания с указанным номером. Инструкция `INT 80h` в Linux используется для выполнения системных вызовов ядра операционной системы. После выполнения этой инструкции управление передается ядру ОС, которое определяет запрашиваемую системную функцию по номеру, заранее помещенному в регистр `EAX`. Параметры системного вызова передаются через регистры в строго определенном порядке: первый параметр помещается в `EBX`, второй - в `ECX`, третий - в `EDX`. Если системной функции требуется вернуть результат, она размещает его в регистре `EAX` после выполнения операции. Например, для вызова функции `exit` нужно поместить в `EAX` номер 1, в `EBX` - код возврата, после чего выполнить `INT 80h`. Этот механизм обеспечивает стандартизированный интерфейс взаимодействия прикладных программ с ядром Linux через программные прерывания.

### **2.3.3 Системные вызовы для обеспечения диалога с пользователем**

Для организации диалога с пользователем в Linux используются системные вызовы `write` и `read`. Вызов `write` (номер 4 в EAX) выводит текст на экран, принимая в EBX дескриптор 1 (стандартный вывод), в ECX - адрес строки, в EDX - её длину. Для ввода текста с клавиатуры используется вызов `read` (номер 3 в EAX), где EBX=0 (стандартный ввод), ECX - адрес буфера ввода, EDX - максимальная длина. Завершать программу необходимо вызовом `exit` (номер 1 в EAX) с кодом возврата 0 в EBX, обеспечивающим корректное завершение работы.

### 3 Выполнение лабораторной работы

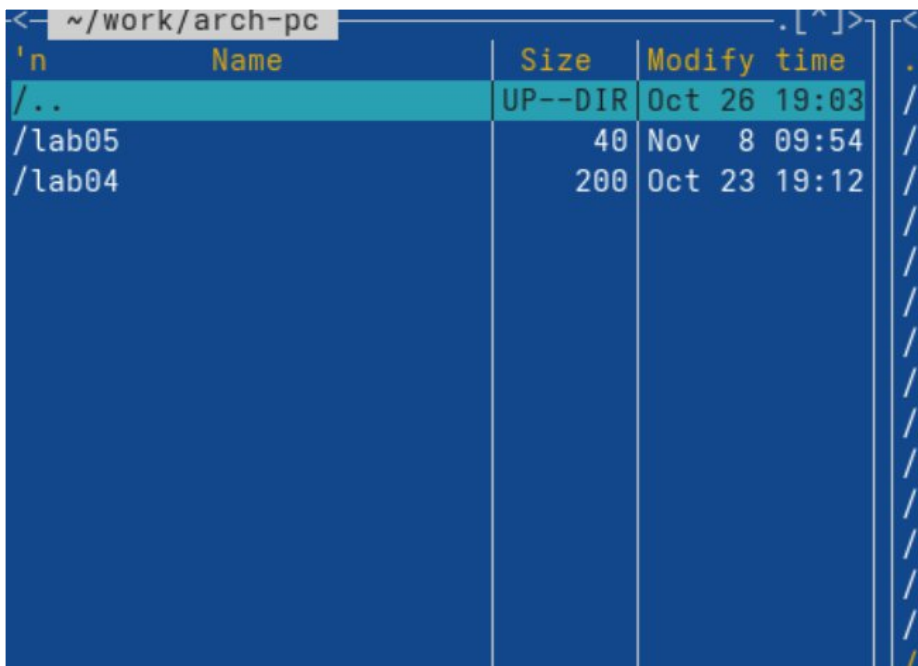
#### 3.1 Работа с mc

Так как mc не скачан,скачаю его.

```
tibezlepkina1@localhost-live:~$ sudo dnf install mc
[sudo] password for tibezlepkina1:
Updating and loading repositories:
Fedora 42 openh264 (From Cisco) - 100% | 986.0 B | 00m02s
Fedora 42 - x86_64 - Updates      100% | 7.5 KiB | 00m00s
Fedora 42 - x86_64                100% | 24.1 KiB | 00m02s
Fedora 42 - x86_64 - Updates      100% | 6.2 MiB | 00m02s
Repositories loaded.
Package      Arch  Version      Reposito     Size
```

Рис 3.1.1 - скачивание mc

Открою Midnight Commander,затем перейду в каталог ~/work/arch-pc,созданный при выполнении лабораторной работы №4.



Name	Size	Modify time
../	UP--DIR	Oct 26 19:03
/lab05	40	Nov 8 09:54
/lab04	200	Oct 23 19:12

Рис 3.1.2 переход в каталог ~/work/arch-pc

Создам папку с помощью функциональной клавиши F7 папку lab05. Но, так как данная клавиша не работает, воспользуюсь альтернативным способом. Нажму клавиши CTRL+O для доступа к командной строке. Введу:

```
tibezlepkina1@localhost-live:~/work/arch-pc$ mkdir lab05
```

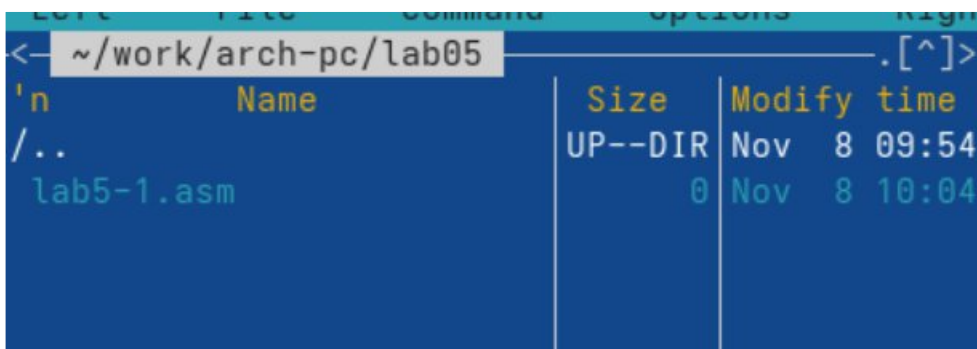
Рис 3.1.3 - создание папки lab05

Снова нажму CTRL+O, чтобы вернуться в mc.

Далее, перейду в созданный каталог аналогичным способом, но уже с помощью команды touch создам папку lab-1.asm.

```
tibezlepkina1@localhost-live:~/work/arch-pc/lab05$ touch lab5-1.asm
```

Рис 3.1.4 - создание папки с помощью команды touch



Name	Size	Modify time
../	UP--DIR	Nov 8 09:54
lab5-1.asm	0	Nov 8 10:04

Name	Size	Modify time
UP--DIR		Nov 8 09:54
lab5-1.asm	0	Nov 8 10:04

Рис 3.1.5 - просмотр папки в mc

Нажимаю Ctrl+O для доступа к командной строке.Ввожу команду для редактора:

```
tibezlepkina1@localhost-live:~/work/arch-pc/lab05$ nano lab5-1.asm
```

Рис 3.1.6 - переход в редактор nano

В качестве встроенного редактора,использую nano.Ввожу текст программы из листинга.

```
GNU nano 8.3 lab5-1.asm
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
----- 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'
```

Рис 3.1.7 - ввожу текст программы.

Сохраняю содержимое с помощью CTRL+X -----> Y. Проверяю изменения.

```
tibezlepkina1@localhost-live:~/work/arch-pc/lab05$ nasm -f elf lab5-1.asm
```

Рис 3.1.8 - трансляция в объектный файл

```
tibezlepkina1@localhost-live:~/work/arch-pc/lab05$ ld -m elf_i386 -o lab5-1 lab5-1.o
```

Рис 3.1.9 - компоновка в исполняемый файл

```
tibezlepkina1@localhost-live:~/work/arch-pc/lab05$ ./lab5-1
Введите строку:
Bezlepkina Tatyana Igorevna
```

Рис 3.1.10 - запуск панели

### 3.2 Подключение внешнего файла in\_out.asm

Используя `find` найдем где находится файл `in_out.asm` и переместим его с помощью команды `mv` в папку `lab05`, проверим с помощью команды `ls` его нахождение в директории `lab05`.

```
tibezlepkin1@localhost-live:~/work/arch-pc/lab05$ find / -name "in_out.asm" 2>/dev/null
/home/tibezlepkin1/Downloads/in_out.asm
/run/overlayfs/home/tibezlepkin1/Downloads/in_out.asm
tibezlepkin1@localhost-live:~/work/arch-pc/lab05$ mv /home/tibezlepkin1/Downloads/in_out.asm
mv: missing destination file operand after '/home/tibezlepkin1/Downloads/in_out.asm'
Try 'mv --help' for more information.
tibezlepkin1@localhost-live:~/work/arch-pc/lab05$ mv /home/tibezlepkin1/Downloads/in_out.asm ~/work/arch-pc/lab05
tibezlepkin1@localhost-live:~/work/arch-pc/lab05$ ls
in_out.asm lab5-1 lab5-1.asm lab5-1.o
```

Рис 3.2.1 - перемещение файла `in_out.asm` в директорию `lab05`, проверка с помощью `ls`

Создаю копию файла `lab5-1.asm` с именем `lab5-2.asm`

```
tibezlepkin1@localhost-live:~/work/arch-pc/lab05$ cp lab5-1.asm lab5-2.asm
```

Рис 3.2.2 - создание копии файла

Меняю содержимое файла `lab5-2.asm`, для начала перехожу в `nano`.

```
GNU nano 8.3 lab5-2.asm
%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
```

Рисунок 3.2.3 - изменение содержимого файла

```
tibezlepkina1@localhost-live:~/work/arch-pc/lab05$ nasm -f elf32 lab5-2.asm -o lab5-2.o
tibezlepkina1@localhost-live:~/work/arch-pc/lab05$ ld -m elf_i386 lab5-2.o -o lab5-2
tibezlepkina1@localhost-live:~/work/arch-pc/lab05$ ./lab5-2
Введите строку:
Bezlepkina Tatyana Igorevna
```

Рис 3.2.4 - трансляция в объектный файл, компоновка в исполняемый файл, запуск панели

Сравним вывод программы при изменении подпрограммы sprintf на printf.

Если мы заменим sprintf на printf, то вывод не будет переходить на новую строку после сообщения.

#### 4 Задания для самостоятельной работы

```
tibezlepkina1@localhost-live:~/work/arch-pc/lab05$ cp lab5-1.asm lab5-1-modified.asm
```

Рис 4.1 - копирование файла

```
GNU nano 8.3 lab5-1-modified.asm
SECTION .data

    msg: DB 'Введите строку: ',0h      ; сообщение

SECTION .bss
    buf1: RESB 80                      ; Буфер размером 80 байт

SECTION .text
    GLOBAL _start

_start:
    ; Вывод приглашения "Введите строку: "
    mov eax, 4                        ; системный вызов sys_write
    mov ebx, 1                        ; файловый дескриптор stdout
    mov ecx, msg                      ; указатель на строку
    mov edx, 16                      ; длина строки (16 символов)
    int 0x80                          ; вызов ядра
```

Рис 4.2 - изменение содержимого файла

```
tibezlepkina1@localhost-live:~/work/arch-pc/lab05$ nasm -f elf32 lab5-1-modified.asm -o lab5-1-modified.o
tibezlepkina1@localhost-live:~/work/arch-pc/lab05$ ld -m elf_i386 lab5-1-modified.o -o lab5-1-modified
tibezlepkina1@localhost-live:~/work/arch-pc/lab05$ ./lab5-1-modified
Введите 0Bezlepkina
Bezlepkina
```

Рисунок 4.3 - трансляция в объектный файл, компоновка в исполняемый файл, запуск панели

```
tibezlepkina1@localhost-live:~/work/arch-pc/lab05$ cp lab5-2.asm lab5-2-modified.asm
```

Рисунок 4.4 - копирование файла

```
GNU nano 8.3 lab5-2-modified.asm
%include 'in_out.asm'      ; подключение внешнего файла

SECTION .data              ; Секция инициализированных данных
    msg: DB 'Введите строку: ', 0h ; сообщение

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
    ; Вывод приглашения
    mov eax, msg           ; запись адреса выводимого сообщения в EAX
    call sprint            ; вызов подпрограммы печати сообщения

    ; Ввод строки с клавиатуры
    mov ecx, buf1          ; запись адреса буфера в ECX
    mov edx, 80            ; запись длины буфера в EDX
```

Рис 4.5 - изменение содержимого файла

```
tibezlepkina1@localhost-live:~/work/arch-pc/lab05$ nasm -f elf lab5-2-modified.asm -o lab5-2-modified.o
tibezlepkina1@localhost-live:~/work/arch-pc/lab05$ ld -m elf_i386 lab5-2-modified.o -o lab5-2-modified
tibezlepkina1@localhost-live:~/work/arch-pc/lab05$ ./lab5-2-modified
Введите строку: Tanya
Tanya
```

Рис 4.6 - трансляция в объектный файл, компоновка в исполняемый файл, запуск панели

## **5 Вывод**

Проделанная работа научила меня основам низкоуровневого программирования на ассемблере, включая работу с системными вызовами Linux для ввода-вывода данных, организацию памяти в секциях `.data`, `.bss` и `.text`, а также использование регистров процессора и прерывания `int 0x80`. Я освоил инструменты разработки - компилятор NASM и линковщик LD, научился работать с командной строкой для компиляции и запуска программ. Практика показала важность модульности кода через использование внешних файлов и процесс рефакторинга программ. Эти навыки дают фундаментальное понимание работы компьютера на уровне операционной системы и являются основой для дальнейшего изучения программирования и оптимизации кода.

## **6 Список литературы**

<https://esystem.rudn.ru/>