

Содержание

| | |
|--|----------|
| 1 ОТЧЕТ | 4 |
| 1.1 по лабораторной работе №5 | 4 |
| 1.2 «Основы работы с Midnight Commander (mc). Структура программы на языке ассемблера NASM. Системные вызовы в ОС GNU Linux» | 4 |
| 1.3 1. Цель работы | 4 |
| 1.4 2. Выполнение лабораторной работы | 5 |
| 1.5 3. Задание для самостоятельной работы | 27 |
| 1.6 4. Ответы на контрольные вопросы | 39 |
| 1.7 5. Выводы | 48 |
| 1.8 6. Список файлов работы | 49 |

Список иллюстраций

Список таблиц

1 ОТЧЕТ

1.1 по лабораторной работе №5

1.2 «Основы работы с Midnight Commander (mc).

Структура программы на языке ассемблера

NASM. Системные вызовы в ОС GNU Linux»

Выполнил: Нхари Хатим

Группа: НБИбд-03-25

Дата: 18.01.2026

1.3 1. Цель работы

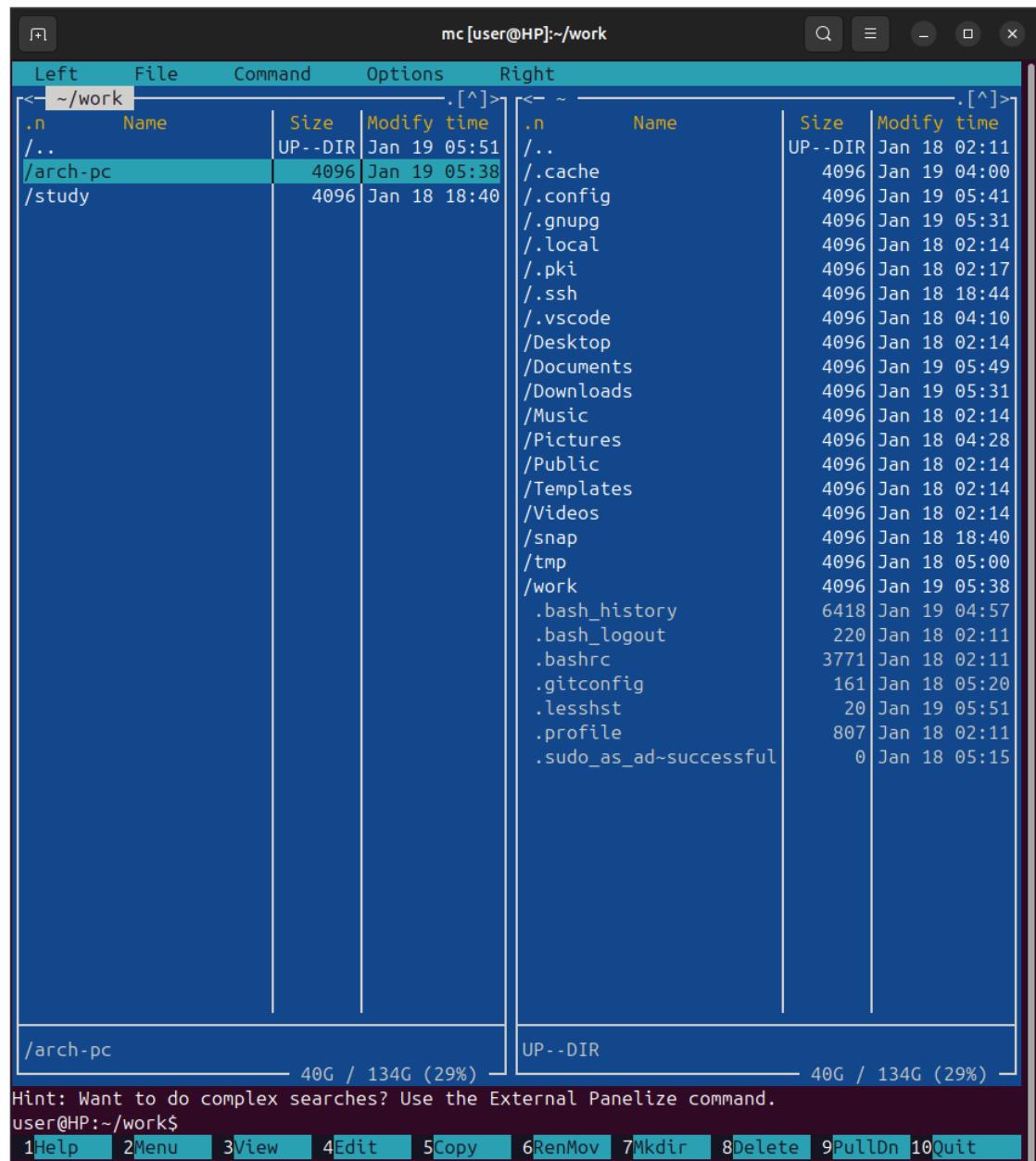
Приобретение практических навыков работы в Midnight Commander. Освоение инструкций языка ассемблера mov и int.

1.4 2. Выполнение лабораторной работы

1.4.1 2.1. Запуск Midnight Commander

Команда:

mc

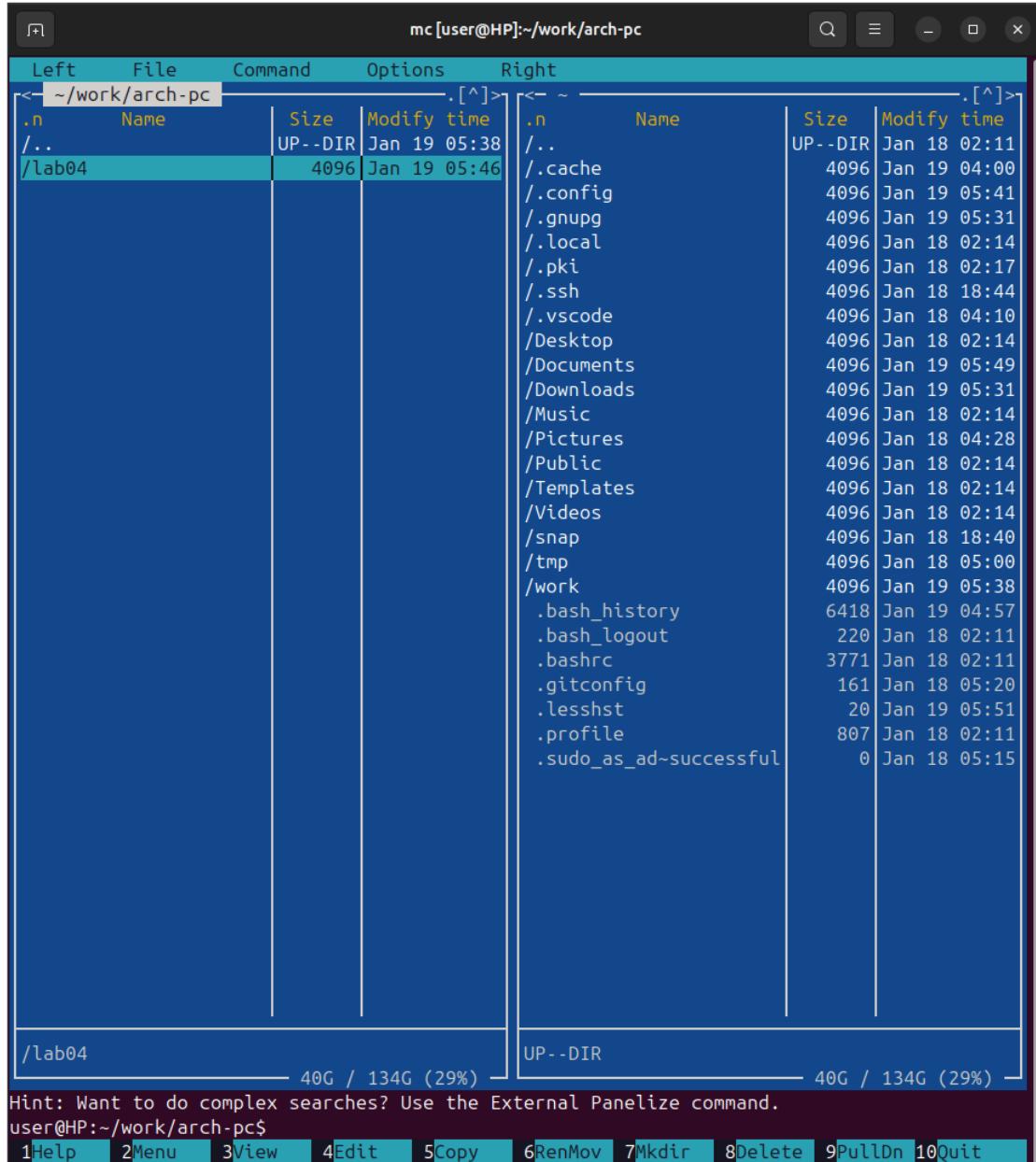


Скриншот 1:

Комментарий: Midnight Commander (mc) - это файловый менеджер с текстовым интерфейсом, который позволяет удобно работать с файловой системой. После запуска открывается двухпанельный интерфейс для навигации по каталогам.

1.4.2 2.2. Переход в рабочий каталог

Действия: - Используя клавиши ↑ , ↓ и Enter, перешел в каталог ~/work/arch-
pc



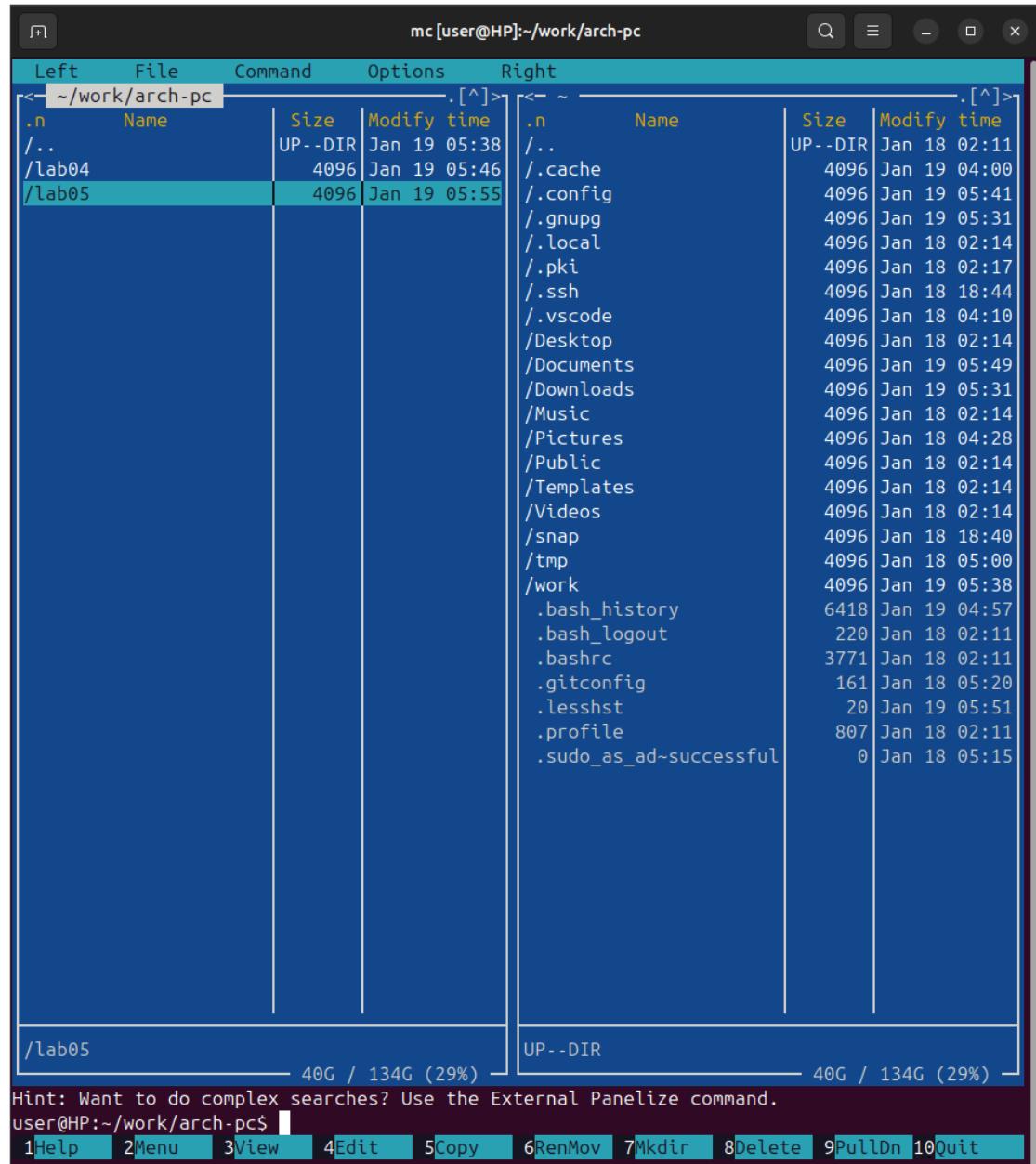
Скриншот 2:

Комментарий: Для навигации в Midnight Commander используются стрелки. Клавиша Enter позволяет войти в выделенный каталог. Клавиша Tab переключает между левой и правой панелями.

1.4.3 2.3. Создание каталога lab05

Действия: - Нажал функциональную клавишу F7 - Ввел имя каталога: lab05

- Нажал Enter



Скриншот 3:

Комментарий: Функциональная клавиша F7 в Midnight Commander используется для создания нового каталога (директории). После нажатия F7 появляется диалоговое окно для ввода имени каталога.

1.4.4 2.4. Переход в каталог lab05

Действия: - Выделил созданный каталог lab05 - Нажал Enter для входа в каталог

| Left | | File | | Command | | Options | | Right | |
|------|------|---------|--------------|---------|---|---------|------------------------|---------|--------------|
| .n | Name | Size | Modify time | .[^]> | | .n | Name | Size | Modify time |
| | .. | UP--DIR | Jan 19 05:55 | <- | ~ | <- | .. | UP--DIR | Jan 18 02:11 |
| | | | | | | | /.. | 4096 | Jan 19 04:00 |
| | | | | | | | /.cache | 4096 | Jan 19 05:41 |
| | | | | | | | /.config | 4096 | Jan 19 05:31 |
| | | | | | | | /.gnupg | 4096 | Jan 18 02:14 |
| | | | | | | | /.local | 4096 | Jan 18 02:17 |
| | | | | | | | /.pki | 4096 | Jan 18 18:44 |
| | | | | | | | /.ssh | 4096 | Jan 18 04:10 |
| | | | | | | | /.vscode | 4096 | Jan 18 02:14 |
| | | | | | | | /Desktop | 4096 | Jan 19 05:49 |
| | | | | | | | /Documents | 4096 | Jan 19 05:31 |
| | | | | | | | /Downloads | 4096 | Jan 18 02:14 |
| | | | | | | | /Music | 4096 | Jan 18 02:14 |
| | | | | | | | /Pictures | 4096 | Jan 18 04:28 |
| | | | | | | | /Public | 4096 | Jan 18 02:14 |
| | | | | | | | /Templates | 4096 | Jan 18 02:14 |
| | | | | | | | /Videos | 4096 | Jan 18 02:14 |
| | | | | | | | /snap | 4096 | Jan 18 18:40 |
| | | | | | | | /tmp | 4096 | Jan 18 05:00 |
| | | | | | | | /work | 4096 | Jan 19 05:38 |
| | | | | | | | .bash_history | 6418 | Jan 19 04:57 |
| | | | | | | | .bash_logout | 220 | Jan 18 02:11 |
| | | | | | | | .bashrc | 3771 | Jan 18 02:11 |
| | | | | | | | .gitconfig | 161 | Jan 18 05:20 |
| | | | | | | | .lessht | 20 | Jan 19 05:51 |
| | | | | | | | .profile | 807 | Jan 18 02:11 |
| | | | | | | | .sudo_as_ad-successful | 0 | Jan 18 05:15 |

Скриншот 4:

Комментарий: После создания каталога перешел в него для дальнейшей работы.

1.4.5 2.5. Создание файла lab5-1.asm

Команда (в строке ввода mc):

```
touch lab5-1.asm
```

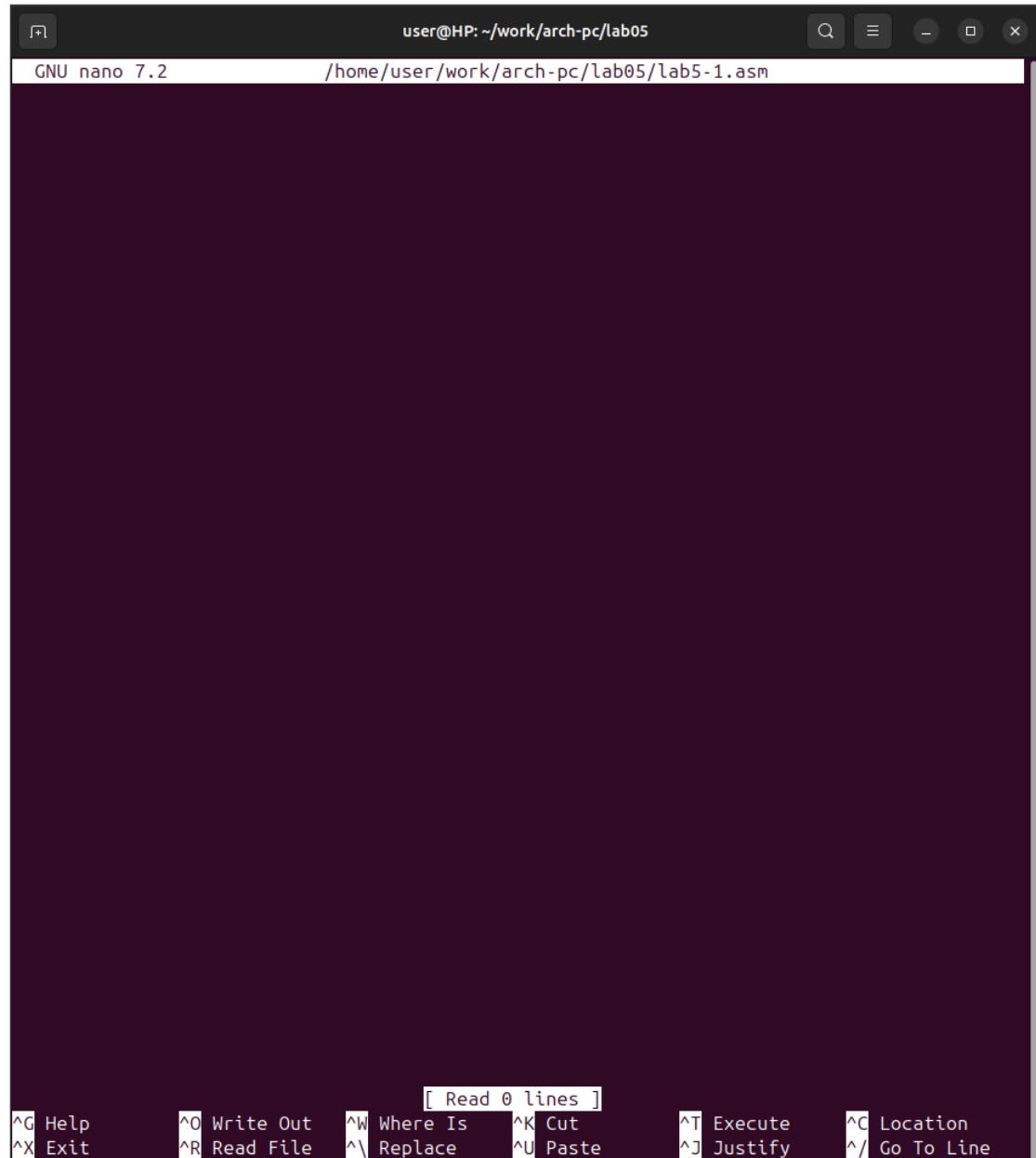
```
user@HP:~/work/arch-pc/lab05$ touch lab5-1.asm
user@HP:~/work/arch-pc/lab05$ █
```

Скриншот 5:

Комментарий: Команда touch создает пустой файл. В Midnight Commander можно использовать встроенную строку ввода для выполнения команд bash без выхода из файлового менеджера.

1.4.6 2.6. Открытие файла для редактирования

Действия: - Выделил файл lab5-1.asm - Нажал функциональную клавишу F4



Скриншот 6:

Комментарий: Функциональная клавиша F4 открывает файл во встроенным текстовом редакторе. В зависимости от настроек системы это может быть nano или mcedit.

1.4.7 2.7. Ввод текста программы lab5-1.asm

Действия: - Ввел текст программы из листинга 5.1

Листинг 5.1. Программа вывода сообщения на экран и ввода строки с клавиатуры:

```
; -----  
; 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

;----- 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

```

Действия по сохранению: - Для mcedit: нажал F2 (сохранить), затем F10 (выход) - Для nano: нажал Ctrl+X, затем Y, затем Enter

```
user@HP:~/work/arch-pc/lab05$ 10
10: command not found

user@HP:~/work/arch-pc/lab05$ cat lab5-1.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'

SECTION .bss            ; Секция не инициализированных данных
    buf1: RESB 80              ; Буфер размером 80 байт

;----- Текст программы -----
SECTION .text           ; Код программы
    GLOBAL _start             ; Начало программы
_start:
    ;----- 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

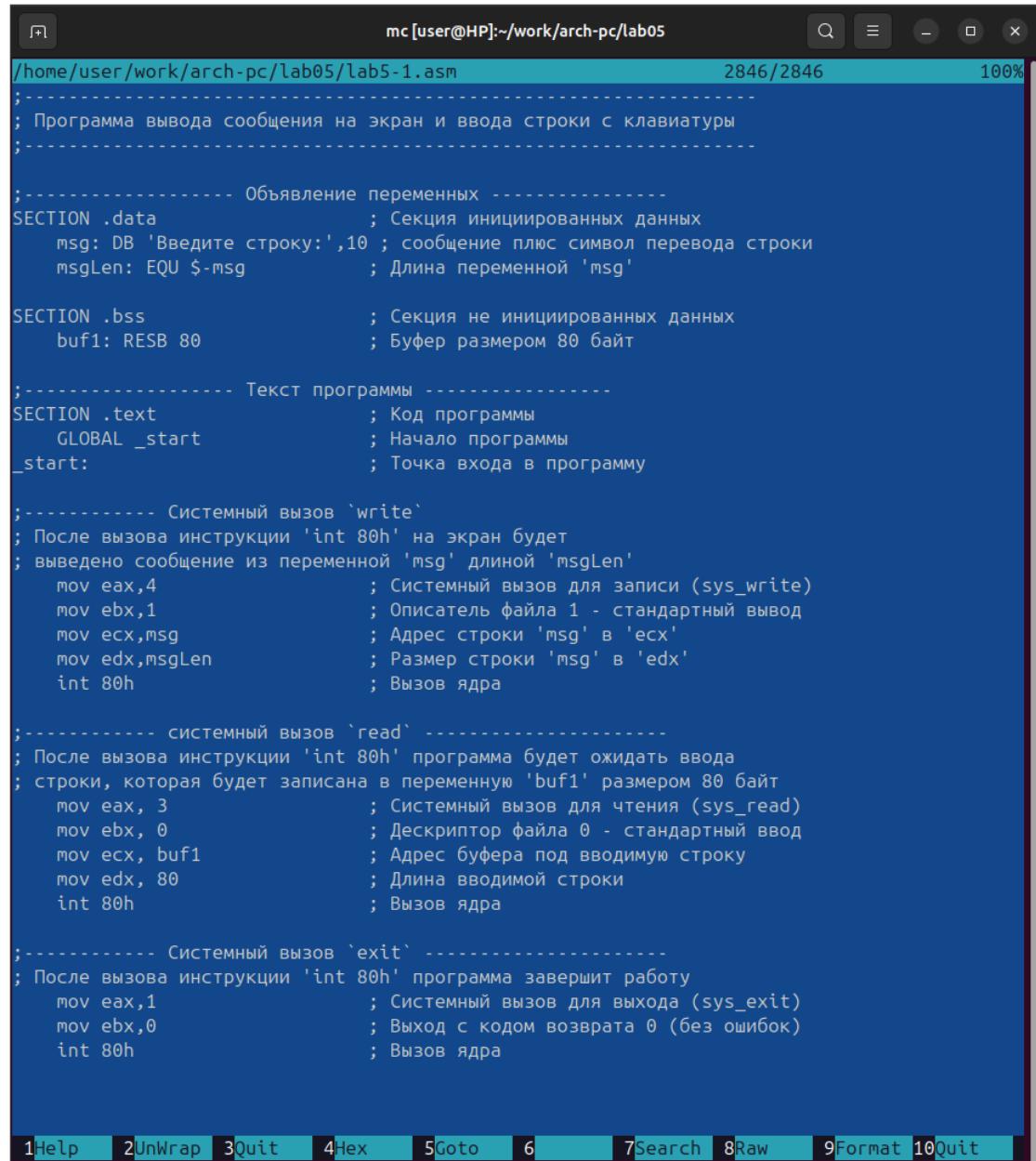
    ;----- Системный вызов `exit` -----
    ; После вызова инструкции 'int 80h' программа завершит работу
        mov eax,1                ; Системный вызов для выхода (sys_exit)
        mov ebx,0                ; Выход с кодом возврата 0 (без ошибок)
        int 80h                  ; Вызов ядра
user@HP:~/work/arch-pc/lab05$
```

Скриншот 7:

Комментарий: Программа состоит из трех секций: .data (инициализированные данные), .bss (неинициализированные данные) и .text (код программы). Программа выводит приглашение, ожидает ввод строки и завершается.

1.4.8 2.8. Просмотр файла

Действия: - Выделил файл lab5-1.asm - Нажал функциональную клавишу F3



The screenshot shows the mc [user@HP]:~/work/arch-pc/lab05 terminal window displaying the assembly code for lab5-1.asm. The code is annotated with Russian comments explaining the purpose of each section and instruction.

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

At the bottom of the window, there is a menu bar with the following options: 1Help, 2UnWrap, 3Quit, 4Hex, 5Goto, 6, 7Search, 8Raw, 9Format, 10Quit. The '1Help' option is currently selected.

Скриншот 8:

Комментарий: Функциональная клавиша F3 открывает файл в режиме просмотра (только для чтения). Убедился, что файл содержит введенный текст программы.

1.4.9 2.9. Трансляция, компоновка и запуск программы

Действия: - Нажал Ctrl+O для доступа к командной строке

Команды:

```
nasm -f elf lab5-1.asm  
ld -m elf_i386 -o lab5-1 lab5-1.o  
./lab5-1
```

Результат:

Введите строку:

Нхари Хатим

```
user@HP: ~/work/arch-pc/lab05

;----- 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

;----- 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
user@HP:~/work/arch-pc/lab05$ nc

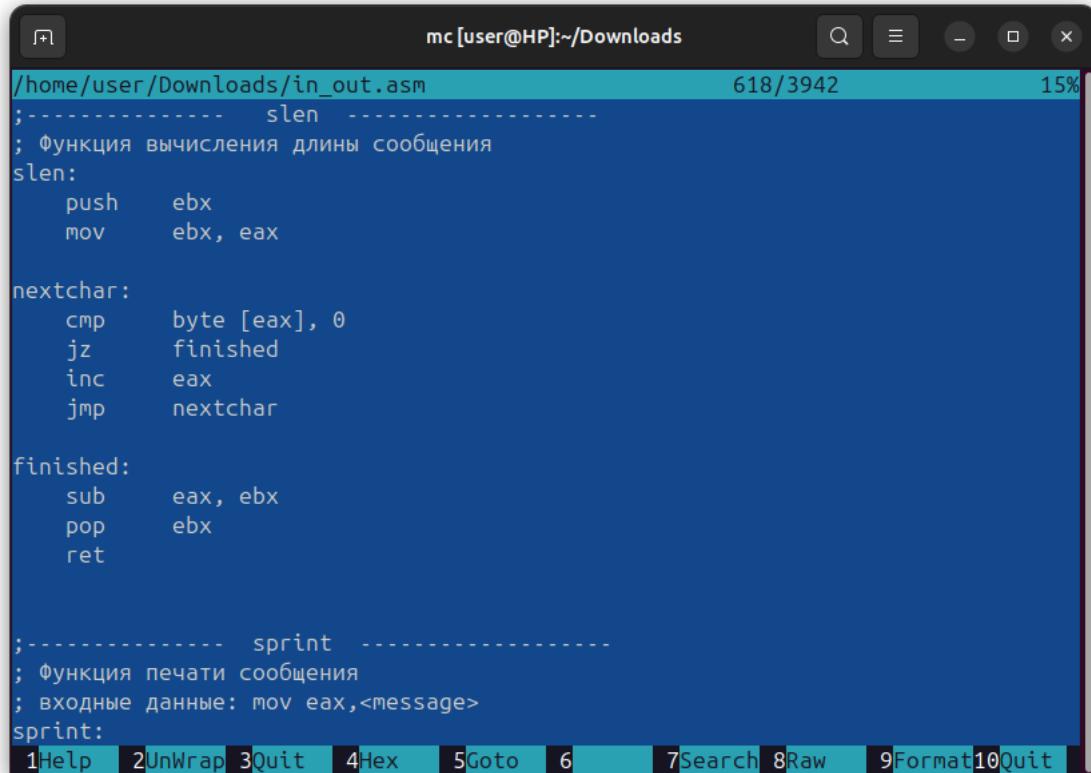
user@HP:~/work/arch-pc/lab05$ nasm -f elf lab5-1.asm
user@HP:~/work/arch-pc/lab05$ ld -m elf_i386 -o lab5-1 lab5-1.o
user@HP:~/work/arch-pc/lab05$ ./lab5-1
Введите строку:
Нхари Хатим
user@HP:~/work/arch-pc/lab05$
```

Скриншот 9:

Комментарий: Команда nasm транслирует исходный код в объектный файл, ld выполняет компоновку, создавая исполняемый файл. При запуске программы вывела приглашение, я ввел свое ФИО.

1.4.10 2.10. Загрузка файла in_out.asm

Действия: - Скачал файл in_out.asm со страницы курса в ТУИС - Открыл Midnight Commander - in_out.asm



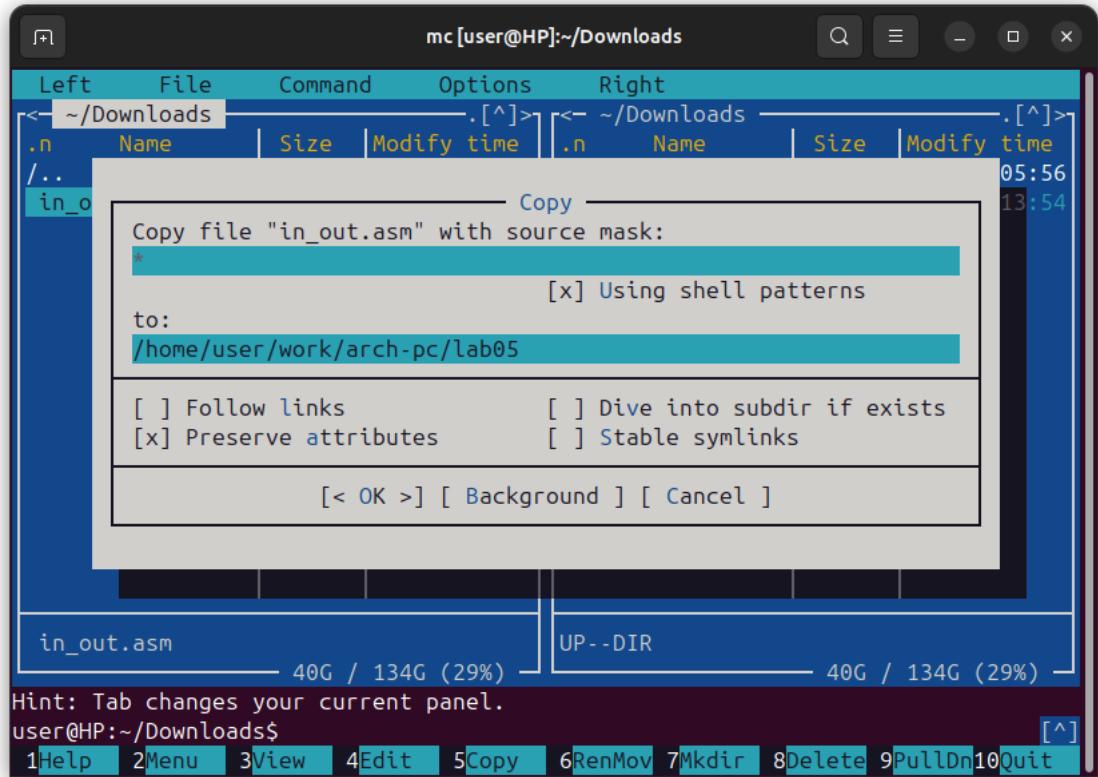
The screenshot shows a terminal window titled 'mc [user@HP:~/Downloads]' displaying assembly code. The file path is '/home/user/Downloads/in_out.asm'. The assembly code includes functions for calculating message length ('slen') and printing messages ('sprint'). The code uses registers eax and ebx. The terminal window has a dark blue background and a light blue header bar. The bottom of the window shows a menu bar with items 1Help, 2UnWrap, 3Quit, 4Hex, 5Goto, 6, 7Search, 8Raw, 9Format, and 10Quit.

```
;----- slen -----  
; Функция вычисления длины сообщения  
slen:  
    push    ebx  
    mov     ebx, eax  
  
nextchar:  
    cmp     byte [eax], 0  
    jz      finished  
    inc     eax  
    jmp     nextchar  
  
finished:  
    sub     eax, ebx  
    pop     ebx  
    ret  
  
;----- sprint -----  
; Функция печати сообщения  
; входные данные: mov eax,<message>  
sprint:  
    1Help  2UnWrap 3Quit  4Hex  5Goto  6  7Search 8Raw  9Format10Quit
```

Скриншот 10:

1.4.11 2.11. Копирование файла in_out.asm

Действия: - Выделил файл in_out.asm - Нажал функциональную клавишу F5 - Подтвердил копирование

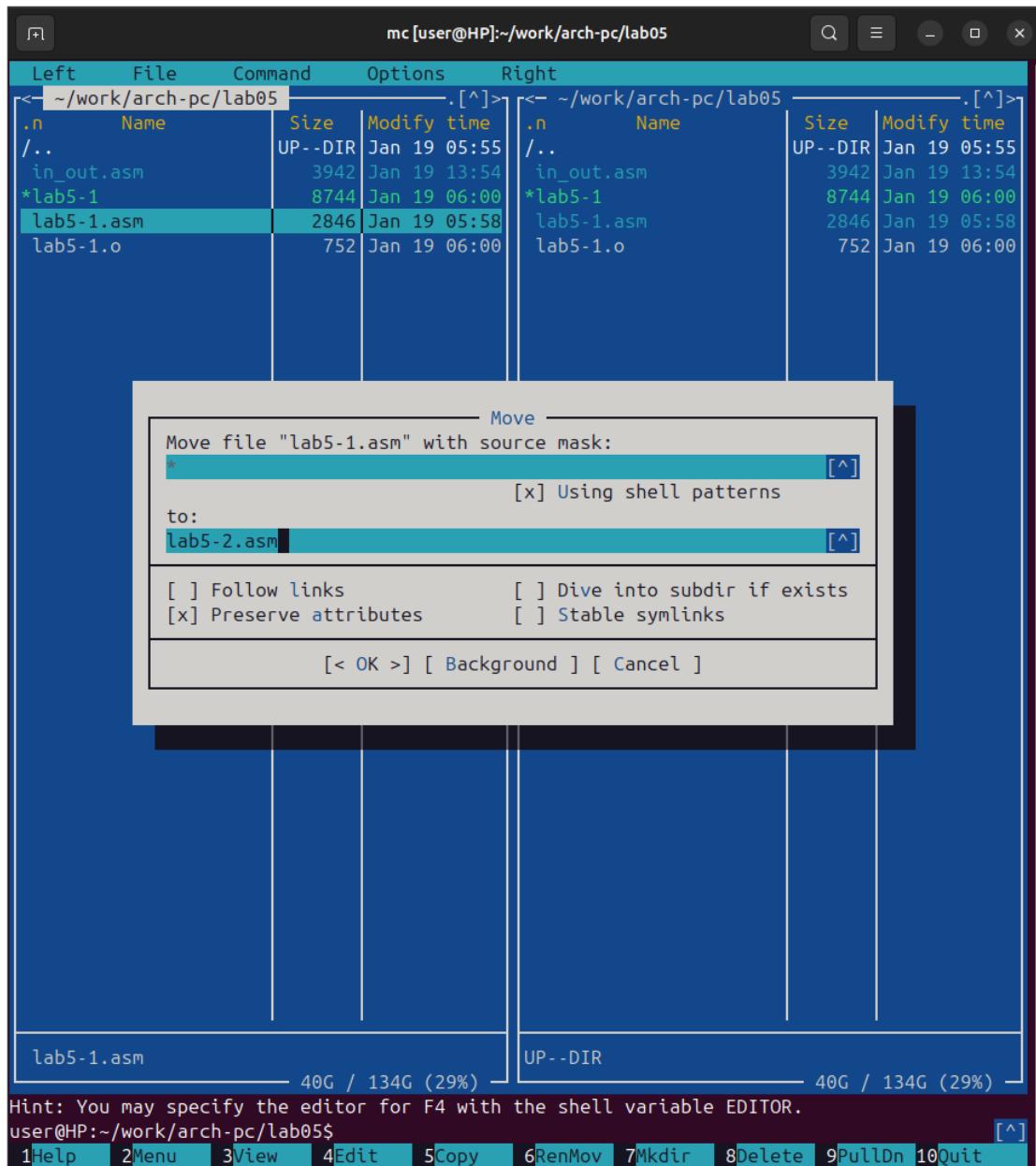


Скриншот 11:

Комментарий: Функциональная клавиша F5 используется для копирования файлов между панелями. Файл `in_out.asm` содержит подпрограммы для упрощения ввода-вывода в программах на ассемблере.

1.4.12 2.12. Создание копии файла `lab5-1.asm`

Действия: - Выделил файл `lab5-1.asm` - Нажал функциональную клавишу F6 - Ввел новое имя: `lab5-2.asm` - Нажал Enter



Скриншот 12:

Комментарий: Функциональная клавиша F6 используется для перемещения или переименования файлов. Если целевой файл находится в том же каталоге, происходит создание копии.

1.4.13 2.13. Редактирование файла lab5-2.asm

Действия: - Выделил файл lab5-2.asm - Нажал F4 для редактирования - Изменил содержимое файла в соответствии с листингом 5.2

Листинг 5.2. Программа с использованием внешнего файла in_out.asm:

```
; -----  
; 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  
  
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 sprintLF ; вызов подпрограммы печати сообщения  
  
    mov ecx, buf1 ; запись адреса переменной в `ECX'  
    mov edx, 80 ; запись длины вводимого сообщения в `EDX'  
    call sread ; вызов подпрограммы ввода сообщения  
  
    call quit ; вызов подпрограммы завершения
```

Действия по сохранению: - Сохранил изменения (F2 для mcedit или Ctrl+X, Y, Enter для nano)

The screenshot shows the mc (midnight commander) editor displaying assembly code for a program named lab5-2.asm. The code is written in AT&T syntax. It includes a comment about printing a string from the keyboard, an include directive for in_out.asm, and definitions for .data and .bss sections. The .text section contains the _start label, which initializes ECX with the address of msg, calls sprintLF to print the message, initializes EDX with the length of the message, calls sread to read input into buf1, and finally calls quit to exit the program. The editor interface includes tabs for Help, UnWrap, Quit, Hex, Goto, Search, Raw, Format, and Quit.

```
mc [user@HP:~/work/arch-pc/lab05] 1487/1487 100%
/home/user/work/arch-pc/lab05/lab5-2.asm

;
; Программа вывода сообщения на экран и ввода строки с клавиатуры
;

%include 'in_out.asm'          ; подключение внешнего файла

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

SECTION .bss                  ; Секция не иницированных данных
buf1: RESB 80                ; Буфер размером 80 байт

SECTION .text                 ; Код программы
GLOBAL _start                 ; Начало программы
_start:
    mov eax, msg             ; запись адреса выводимого сообщения в `EAX`
    call sprintLF            ; вызов подпрограммы печати сообщения

    mov ecx, buf1            ; запись адреса переменной в `ECX`
    mov edx, 80               ; запись длины вводимого сообщения в `EDX`
    call sread                ; вызов подпрограммы ввода сообщения

    call quit                 ; вызов подпрограммы завершения

1Help 2UnWrap 3Quit 4Hex 5Goto 6 7Search 8Raw 9Format 10Quit
```

Скриншот 13:

Комментарий: В этой версии программы используются подпрограммы из внешнего файла in_out.asm: sprintLF для вывода строки с переводом строки, sread для ввода и quit для завершения программы.

1.4.14 2.14. Компиляция и запуск программы lab5-2.asm

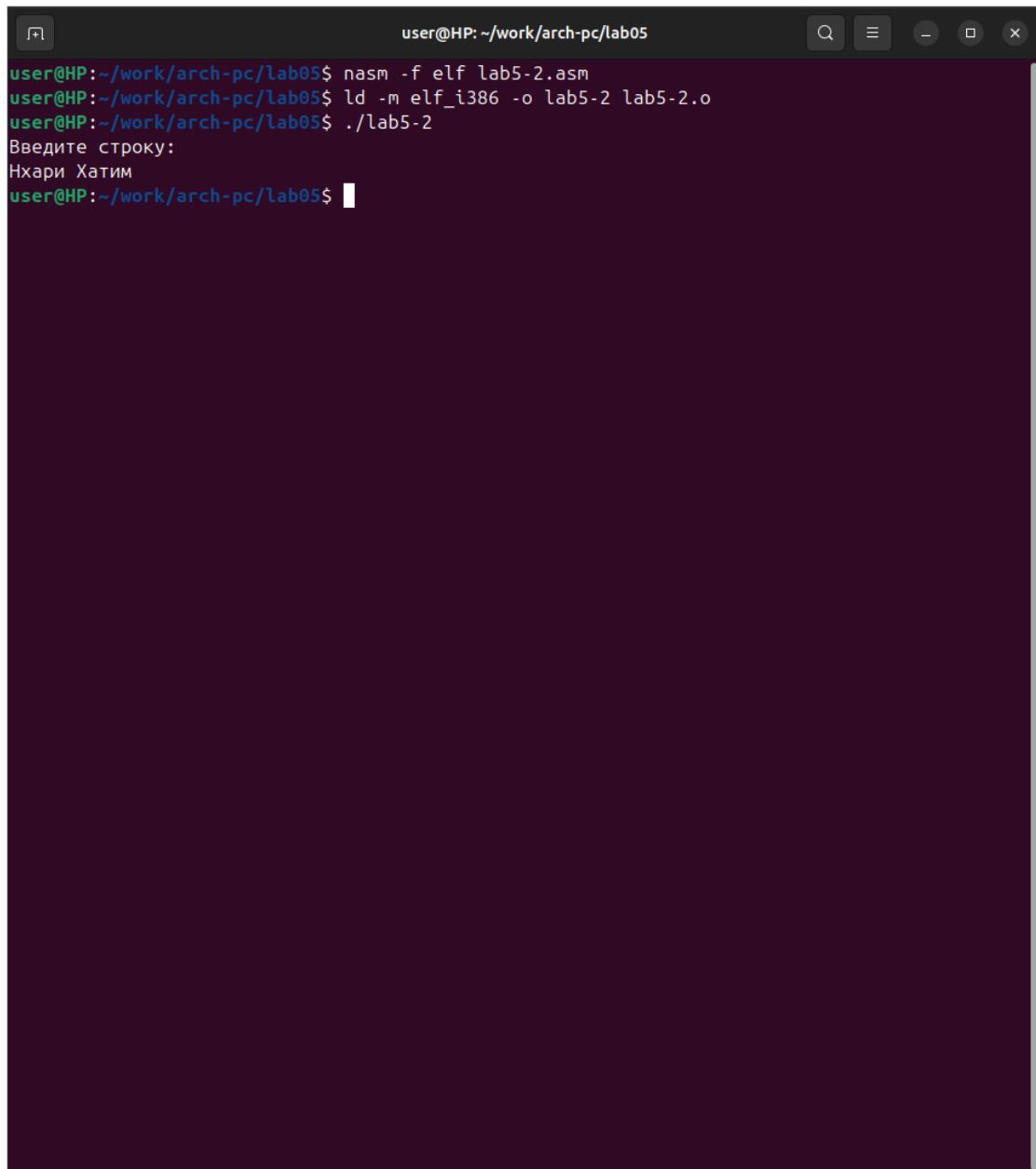
Команды:

```
nasm -f elf lab5-2.asm  
ld -m elf_i386 -o lab5-2 lab5-2.o  
./lab5-2
```

Результат:

Введите строку:

Нхари Хатим



The screenshot shows a terminal window with the following session:

```
+ user@HP:~/work/arch-pc/lab05$ nasm -f elf lab5-2.asm
user@HP:~/work/arch-pc/lab05$ ld -m elf_i386 -o lab5-2 lab5-2.o
user@HP:~/work/arch-pc/lab05$ ./lab5-2
Введите строку:
Нхари Хатим
user@HP:~/work/arch-pc/lab05$
```

Скриншот 14:

Комментарий: Программа работает корректно. Использование внешнего файла `in_out.asm` упрощает код и делает его более читаемым.

1.4.15 2.15. Замена подпрограммы sprintLF на sprint

Действия: - Открыл файл lab5-2.asm для редактирования (F4) - Заменил строку call sprintLF на call sprint - Сохранил изменения

Измененная строка:

```
call sprint ; вызов подпрограммы печати сообщения (без перевода с
```

Команды:

```
nasm -f elf lab5-2.asm  
ld -m elf_i386 -o lab5-2 lab5-2.o  
./lab5-2
```

Результат:

Введите строку: Нхари Хатим

The screenshot shows a terminal window with the following session:

```
user@HP:~/work/arch-pc/lab05$ nasm -f elf lab5-2.asm
user@HP:~/work/arch-pc/lab05$ ld -m elf_i386 -o lab5-2 lab5-2.o
user@HP:~/work/arch-pc/lab05$ ./lab5-2
Введите строку:
Нхари Хатим
user@HP:~/work/arch-pc/lab05$ mc

user@HP:~/work/arch-pc/lab05$ 1
1: command not found

user@HP:~/work/arch-pc/lab05$ nasm -f elf lab5-2.asm
user@HP:~/work/arch-pc/lab05$ ld -m elf_i386 -o lab5-2 lab5-2.o
user@HP:~/work/arch-pc/lab05$ ./lab5-2
Введите строку: Нхари Хатим
user@HP:~/work/arch-pc/lab05$
```

Скриншот 15:

Комментарий:

Разница между `sprintLF` и `sprint`: - `sprintLF` выводит сообщение и добавляет символ перевода строки (переход на новую строку) - `sprint` выводит только сообщение без перевода строки

При использовании `sprint` курсор остается на той же строке, что и приглашение «Ведите строку:», поэтому ввод происходит сразу после двоеточия.

При использовании `sprintLF` ввод осуществляется на новой строке.

1.5 3. Задание для самостоятельной работы

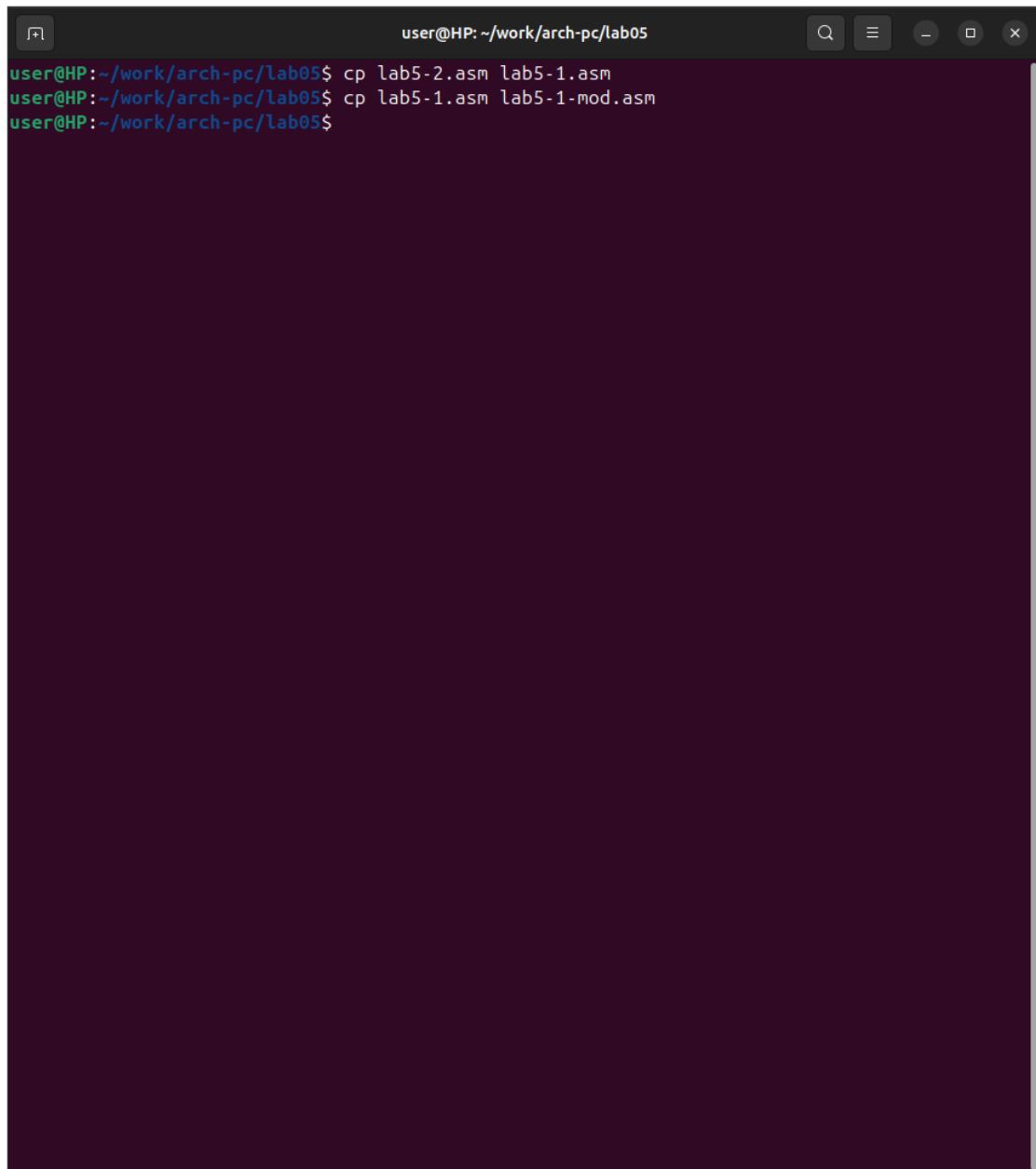
1.5.1 3.1. Модификация программы lab5-1.asm (без использования in_out.asm)

Задание: Создать копию файла `lab5-1.asm` и модифицировать программу так, чтобы она: 1. Выводила приглашение «Ведите строку:» 2. Считывала строку с клавиатуры 3. Выводила введенную строку на экран

Действия: - Создал копию файла `lab5-1.asm` с именем `lab5-1-mod.asm`

Команда:

```
cp lab5-1.asm lab5-1-mod.asm
```



The screenshot shows a terminal window with a dark background and light-colored text. At the top, it displays the user's name and the current directory: `user@HP: ~/work/arch-pc/lab05`. Below this, three commands are listed in green text:
`user@HP:~/work/arch-pc/lab05$ cp lab5-2.asm lab5-1.asm`
`user@HP:~/work/arch-pc/lab05$ cp lab5-1.asm lab5-1-mod.asm`
`user@HP:~/work/arch-pc/lab05$`

Скриншот 16:

1.5.2 3.2. Редактирование файла lab5-1-mod.asm

Листинг программы lab5-1-mod.asm:

```

; -----
; Программа вывода сообщения на экран, ввода строки и вывода строки
; -----



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

; Вывод приглашения
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

; Ввод строки с клавиатуры
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 esi, eax ; Сохраняем количество введенных символов в esi
```

; Вывод введенной строки на экран

```
mov eax, 4 ; Системный вызов для записи (sys_write)  
mov ebx, 1 ; Описатель файла 1 - стандартный вывод  
mov ecx, buf1 ; Адрес буфера с введенной строкой  
mov edx, esi ; Количество символов для вывода  
int 80h ; Вызов ядра
```

; Завершение программы

```
mov eax, 1 ; Системный вызов для выхода (sys_exit)  
mov ebx, 0 ; Выход с кодом возврата 0 (без ошибок)  
int 80h ; Вызов ядра
```

The screenshot shows the Mars Debugger (MC) interface. The title bar reads "mc [user@HP]:~/work/arch-pc/lab05". The status bar at the bottom shows various menu options: 1Help, 2UnWrap, 3Quit, 4Hex, 5Goto, 6, 7Search, 8Raw, 9Format, 10Quit. The main window displays assembly code for a program named "lab5-1-mod.asm". The code includes sections for data, bss, and text, with comments explaining the purpose of each instruction. The assembly code uses mnemonics like mov, eax, ebx, ecx, edx, and int, along with comments in Russian. The code handles user input, system calls for reading and writing, and exits the program.

```
/home/user/work/arch-pc/lab05/lab5-1-mod.asm
; -----
; Программа вывода сообщения на экран, ввода строки и вывода строки
;

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

SECTION .text ; Код программы
    GLOBAL _start ; Начало программы
_start: ; Точка входа в программу

; Вывод приглашения
    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

; Ввод строки с клавиатуры
    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 esi, eax ; Сохраняем количество введенных символов в esi

; Вывод введенной строки на экран
    mov eax, 4 ; Системный вызов для записи (sys_write)
    mov ebx, 1 ; Описатель файла 1 - стандартный вывод
    mov ecx, buf1 ; Адрес буфера с введенной строкой
    mov edx, esi ; Количество символов для вывода
    int 80h ; Вызов ядра

; Завершение программы
    mov eax,1 ; Системный вызов для выхода (sys_exit)
    mov ebx,0 ; Выход с кодом возврата 0 (без ошибок)
    int 80h ; Вызов ядра
```

Скриншот 17:

1.5.3 3.3. Компиляция и запуск lab5-1-mod.asm

Команды:

```
nasasm -f elf lab5-1-mod.asm  
ld -m elf_i386 -o lab5-1-mod lab5-1-mod.o  
./lab5-1-mod
```

Результат:

Введите строку:

Нхари

Нхари

```
user@HP:~/work/arch-pc/lab05$ mc
user@HP:~/work/arch-pc/lab05$ nasm -f elf lab5-1-mod.asm
user@HP:~/work/arch-pc/lab05$ ld -m elf_i386 -o lab5-1-mod lab5-1-mod.o
user@HP:~/work/arch-pc/lab05$ ./lab5-1-mod
Введите строку:
Нхари
Нхари
user@HP:~/work/arch-pc/lab05$
```

Скриншот 18:

Комментарий: Программа корректно выводит приглашение, принимает ввод с клавиатуры (моя фамилия «Нхари») и выводит введенную строку на экран. Ключевое изменение - добавлен третий блок системного вызова write для вывода содержимого буфера buf1.

1.5.4 3.4. Модификация программы lab5-2.asm (с использованием in_out.asm)

Задание: Создать копию файла lab5-2.asm и модифицировать программу с использованием подпрограмм из in_out.asm так, чтобы она: 1. Выводила приветствие «Введите строку:» 2. Считывала строку с клавиатуры 3. Выводила введенную строку на экран

Действия: - Создал копию файла lab5-2.asm с именем lab5-2-mod.asm

Команда:

```
cp lab5-2.asm lab5-2-mod.asm
```

```
user@HP:~/work/arch-pc/lab05$ cp lab5-2.asm lab5-2-mod.asm
user@HP:~/work/arch-pc/lab05$
```

Скриншот 19:

1.5.5 3.5. Редактирование файла lab5-2-mod.asm

Листинг программы lab5-2-mod.asm:

```
; -----
; Программа вывода сообщения, ввода и вывода строки с использованием in_out.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

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

mov ecx, buf1           ; запись адреса переменной в `ECX`
mov edx, 80              ; запись длины вводимого сообщения в `EDX`
call sread               ; вызов подпрограммы ввода сообщения

mov eax, buf1           ; запись адреса буфера с введенной строкой в `EAX`
call sprint             ; вызов подпрограммы печати введенной строки

call quit                ; вызов подпрограммы завершения

```

The screenshot shows a terminal window with the title "user@HP: ~/work/arch-pc/lab05". The file being edited is "/home/user/work/arch-pc/lab05/lab5-2-mod.asm". The code is as follows:

```
GNU nano 7.2 /home/user/work/arch-pc/lab05/lab5-2-mod.asm
;-----;
; Программа вывода сообщения, ввода и вывода строки с использованием in_out.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:
    mov eax, msg                ; запись адреса выводимого сообщения в `EAX`
    call sprint                 ; вызов подпрограммы печати сообщения

    mov ecx, buf1              ; запись адреса переменной в `ECX`
    mov edx, 80                  ; запись длины вводимого сообщения в `EDX`
    call sread                  ; вызов подпрограммы ввода сообщения

    mov eax, buf1              ; запись адреса буфера с введенной строкой в `EAX`
    call sprint                 ; вызов подпрограммы печати введенной строки

    call quit                   ; вызов подпрограммы завершения

[ Read 27 lines ]
^G Help      ^O Write Out   ^W Where Is   ^K Cut        ^T Execute   ^C Location
^X Exit      ^R Read File   ^V Replace    ^U Paste      ^J Justify   ^/ Go To Line
```

Скриншот 20:

Комментарий: В программу добавлен второй вызов подпрограммы `sprint` для вывода содержимого буфера `buf1` на экран после ввода.

1.5.6 3.6. Компиляция и запуск lab5-2-mod.asm

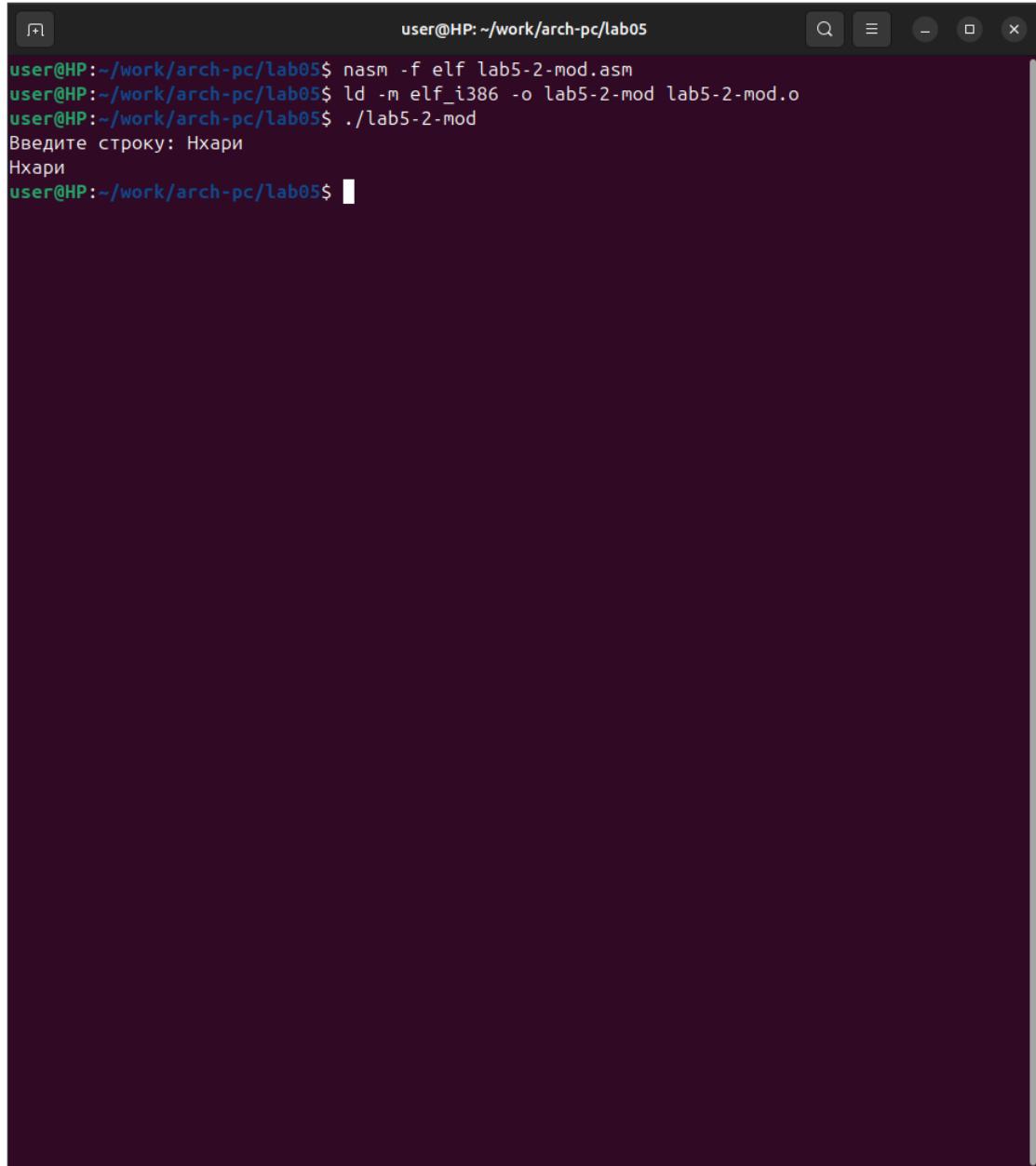
Команды:

```
nasm -f elf lab5-2-mod.asm  
ld -m elf_i386 -o lab5-2-mod lab5-2-mod.o  
./lab5-2-mod
```

Результат:

Введите строку: Нхари

Нхари



The screenshot shows a terminal window with a dark background and light-colored text. At the top, it displays the command line interface with the user's name and the path: `user@HP: ~/work/arch-pc/lab05`. Below this, several commands are run sequentially:

```
user@HP:~/work/arch-pc/lab05$ nasm -f elf lab5-2-mod.asm
user@HP:~/work/arch-pc/lab05$ ld -m elf_i386 -o lab5-2-mod lab5-2-mod.o
user@HP:~/work/arch-pc/lab05$ ./lab5-2-mod
Введите строку: Нхари
Нхари
user@HP:~/work/arch-pc/lab05$
```

Скриншот 21:

Комментарий: Программа работает корректно. После ввода строки она выводится на экран. Использование подпрограмм из `in_out.asm` делает код более компактным и понятным по сравнению с версией без использования внешнего файла.

1.5.7 3.7. Загрузка файлов на GitHub

Команды:

```
cp -r ~/work/arch-pc/lab05/ ~/work/study/2025-2026/Архитектура\ компьютера/arch-pc/lab05
cd ~/work/study/2025-2026/Архитектура\ компьютера/arch-pc/labs/lab05
git add .
git commit -m "feat(lab05): add lab5 programs and report"
git push
```

Комментарий: Все файлы лабораторной работы успешно загружены в репозиторий на GitHub.

1.6 4. Ответы на контрольные вопросы

1. Каково назначение mc?

Midnight Commander (mc) - это файловый менеджер с текстовым интерфейсом для операционных систем семейства Unix/Linux. Его назначение:

- Просмотр структуры каталогов файловой системы
- Выполнение основных операций по управлению файлами и каталогами (копирование, перемещение, удаление, создание)
- Просмотр и редактирование текстовых файлов
- Навигация по файловой системе в удобном двухпанельном интерфейсе
- Выполнение команд оболочки без выхода из программы

Midnight Commander делает работу с файлами более удобной и наглядной, особенно при работе через терминал без графического интерфейса.

2. Какие операции с файлами можно выполнить как с помощью команд bash, так и с помощью меню (комбинаций клавиш) mc? Приведите несколько примеров.

Многие операции доступны в обоих интерфейсах:

| Операция | Команда bash | Midnight Commander |
|----------------------------------|--------------------------------|--------------------------------------|
| Просмотр содержимого каталога | ls | Автоматически отображается в панелях |
| Создание каталога | mkdir dirname | F7 |
| Копирование файла | cp file1 file2 | F5 |
| Перемещение/переименование файла | mv file1 file2 | F6 |
| Удаление файла | rm filename | F8 |
| Просмотр файла | cat filename или less filename | F3 |
| Редактирование файла | nano filename или vi filename | F4 |
| Создание файла | touch filename | Shift+F4 или через командную строку |
| Смена каталога | cd /path/to/dir | Навигация стрелками и Enter |
| Поиск файлов | find или grep | Alt+? или Команда > Поиск файла |

Преимущество mc заключается в визуальном представлении файловой системы и более интуитивном интерфейсе для некоторых операций.

3. Какова структура программы на языке ассемблера NASM?

Программа на языке ассемблера NASM обычно состоит из трех основных секций:

```
SECTION .data          ; Секция инициализированных данных
    ; Переменные с заданными начальными значениями
    msg db 'Hello', 0
    num dd 42

SECTION .bss          ; Секция неинициализированных данных
    ; Переменные, для которых только резервируется память
    buffer resb 100
    count resd 1

SECTION .text          ; Секция кода программы
    GLOBAL _start        ; Объявление точки входа
    _start:                ; Метка точки входа
    ; Инструкции программы
    mov eax, 4
    mov ebx, 1
    int 80h

    ; Обязательное завершение программы
    mov eax, 1
    mov ebx, 0
    int 80h
```

Основные элементы: - **SECTION .data** - содержит инициализированные данные (константы, строки со значениями) - **SECTION .bss** - содержит неинициализированные данные (буферы, переменные без начальных значений) - **SECTION**

.text - содержит исполняемый код программы - `**GLOBAL _start**` - объявляет точку входа в программу видимой для компоновщика - `**_start:**` - метка, указывающая начало выполнения программы - Программа должна завершаться системным вызовом `exit (int 80h с eax=1)`

4. Для описания каких данных используются секции `bss` и `data` в языке ассемблера NASM?

SECTION .data используется для описания инициализированных данных - данных, для которых известно начальное значение во время компиляции:

- Строковые константы: `msg db 'Hello, World!', 10, 0`
- Числовые константы: `pi dd 3.14159`
- Массивы с известными значениями: `array db 1, 2, 3, 4, 5`
- Любые данные, значения которых известны заранее

SECTION .bss используется для описания неинициализированных данных - данных, для которых во время компиляции только резервируется память, а значение присваивается в ходе выполнения программы:

- Буферы для ввода: `buffer resb 80`
- Переменные для хранения результатов вычислений: `result resd 1`
- Массивы без начальных значений: `array resw 100`
- Любые данные, значения которых будут определены во время выполнения программы

Основные различия:

| Характеристика | SECTION .data | SECTION .bss |
|---------------------------|---------------------------------|--|
| Инициализация | Данные имеют начальное значение | Данные не имеют начального значения |
| Размер исполняемого файла | Увеличивает размер файла | Не увеличивает (только указывается размер) |
| Использование | Константы, известные строки | Буферы, переменные для вычислений |
| Директивы | DB, DW, DD, DQ, DT | RESB, RESW, RESD, RESQ, REST |

5. Для чего используются компоненты db, dw, dd, dq и dt языка ассемблера NASM?

Эти директивы используются для объявления инициированных данных в секции .data и определяют размер резервируемой памяти:

DB (Define Byte) - определяет переменную размером 1 байт (8 бит):

```
char db 'A'           ; один символ
flag db 1             ; число 0-255
binary db 10011001b   ; двоичное число
```

DW (Define Word) - определяет переменную размером 2 байта (16 бит / слово):

```
number dw 1234         ; число 0-65535
word_val dw 0FFFFh     ; шестнадцатеричное значение
```

DD (Define Double Word) - определяет переменную размером 4 байта (32 бита / двойное слово):

```
big_num dd 123456789      ; большое целое число
float_num dd 3.14          ; число с плавающей точкой
pointer dd 0               ; указатель (адрес)
```

DQ (Define Quad Word) - определяет переменную размером 8 байт (64 бита / учетверенное слово):

```
very_big dq 9876543210    ; очень большое число
double_val dq 3.141592653589793 ; double precision float
```

DT (Define Ten Bytes) - определяет переменную размером 10 байт (80 бит):

```
extended dt 3.14159265358979323846 ; extended precision float
```

Примеры использования:

```
; Стока символов (массив байт)
message db 'Hello, World!', 0

; Массив чисел
numbers db 1, 2, 3, 4, 5

; Комбинированное использование
data_mixed db 10, 'A', 255, 0

; Размер в шестнадцатеричной системе
hex_value dd 0x12345678
```

6. Какое произойдёт действие при выполнении инструкции mov eax, esi?

При выполнении инструкции `mov eax, esi` произойдет **копирование значения** из регистра-источника (`esi`) в регистр-приемник (`eax`).

- Детальное описание:**
- Содержимое регистра ESI будет скопировано в регистр EAX - Регистр ESI останется без изменений (значение сохранится)
 - Предыдущее значение регистра EAX будет потеряно (перезаписано) - Оба регистра 32-битные, поэтому все 32 бита будут скопированы

Пример:

```
; До выполнения:  
; ESI = 0x12345678  
; EAX = 0xAAAAABBBB  
  
mov eax, esi  
  
; После выполнения:  
; ESI = 0x12345678 (не изменился)  
; EAX = 0x12345678 (скопировано значение из ESI)
```

Важные замечания: - Инструкция `mov` не выполняет арифметических операций, только копирование - Размеры operandов должны совпадать (нельзя: `mov eax, cl` - разные размеры) - Нельзя копировать напрямую из памяти в память (нужны две инструкции `mov`) - Флаги процессора не изменяются при выполнении `mov`

7. Для чего используется инструкция `int 80h`?

Инструкция `int 80h` используется для **вызыва прерывания с номером 80h (128 в десятичной системе)**, что в операционных системах Linux означает **системный вызов** (system call).

Назначение: - Передача управления ядру операционной системы - Выполнение привилегированных операций (ввод-вывод, работа с файлами, управление процессами) - Взаимодействие программы пользовательского уровня с ядром ОС

Механизм работы:

1. **Перед вызовом** в регистры помещаются параметры:
 - **EAX** - номер системного вызова
 - **EBX, ECX, EDX, ESI, EDI, EBP** - аргументы системного вызова
2. **Выполнение int 80h** - передача управления ядру
3. **Ядро обрабатывает запрос** и выполняет соответствующую функцию
4. **Возврат результата** в регистр EAX

Основные системные вызовы:

| EAX | Системный вызов | Описание | Параметры |
|-----|-----------------|----------------------|--|
| 1 | sys_exit | Завершение программы | EBX = код возврата |
| 3 | sys_read | Чтение данных | EBX = дескриптор, ECX = буфер, EDX = количество байт |
| 4 | sys_write | Запись данных | EBX = дескриптор, ECX = буфер, EDX = количество байт |
| 5 | sys_open | Открытие файла | EBX = имя файла, ECX = флаги, EDX = права |

| EAX | Системный вызов | Описание | Параметры |
|-----|-----------------|----------------|------------------------|
| 6 | sys_close | Закрытие файла | EBX = дескриптор файла |

Примеры использования:

Вывод строки на экран:

```
mov eax, 4          ; sys_write
mov ebx, 1          ; stdout (стандартный вывод)
mov ecx, message   ; адрес строки
mov edx, msg_len   ; длина строки
int 80h             ; вызов ядра
```

Ввод данных с клавиатуры:

```
mov eax, 3          ; sys_read
mov ebx, 0          ; stdin (стандартный ввод)
mov ecx, buffer     ; адрес буфера
mov edx, 80         ; максимальная длина
int 80h             ; вызов ядра
```

Завершение программы:

```
mov eax, 1          ; sys_exit
mov ebx, 0          ; код возврата 0 (успех)
int 80h             ; вызов ядра
```

Примечание: В современных 64-битных системах Linux используется инструкция `syscall` вместо `int 80h`, но для 32-битных программ `int 80h` остается актуальной.

1.7 5. Выводы

В ходе выполнения лабораторной работы №5 были приобретены практические навыки работы в Midnight Commander и освоены базовые инструкции языка ассемблера NASM:

1. Освоен файловый менеджер Midnight Commander:

- Изучены основные функциональные клавиши (F1-F10)
- Освоена навигация по файловой системе
- Практически применены операции создания, копирования, редактирования файлов
- Использован встроенный текстовый редактор для работы с исходным кодом

2. Изучена структура программы на языке ассемблера NASM:

- Понята роль секций .data, .bss и .text
- Освоены директивы объявления данных (DB, DW, DD и RESB, RESW, RESD)
- Изучена структура программы с точкой входа _start

3. Освоены базовые инструкции ассемблера:

- Инструкция **mov** для копирования данных между регистрами и памятью
- Инструкция **int 80h** для выполнения системных вызовов
- Системные вызовы sys_write (вывод), sys_read (ввод), sys_exit (завершение)

4. Практически применено использование внешних файлов:

- Подключен файл in_out.asm с помощью директивы %include
- Использованы подпрограммы sprint, sprintLF, sread и quit

- Понята разница между различными вариантами подпрограмм (с переводом строки и без)

5. Созданы и протестированы четыре программы:

- lab5-1.asm - базовая программа с системными вызовами
- lab5-2.asm - программа с использованием внешних подпрограмм
- lab5-1-mod.asm - модифицированная версия с выводом введенной строки
- lab5-2-mod.asm - улучшенная версия с подпрограммами

6. Выполнена работа с системой контроля версий:

- Все файлы программ загружены в репозиторий GitHub
- Соблюдена структура каталогов проекта

Полученные навыки являются фундаментальными для дальнейшего изучения программирования на языке ассемблера и понимания низкоуровневой работы программ в операционной системе Linux.

1.8 6. Список файлов работы

Программы: - lab5-1.asm - программа вывода сообщения и ввода строки (базовая версия) - lab5-2.asm - программа с использованием in_out.asm - lab5-1-mod.asm - модифицированная программа без внешнего файла - lab5-2-mod.asm - модифицированная программа с внешним файлом - in_out.asm - файл с подпрограммами ввода-вывода

Отчет: - report.md - отчет в формате Markdown - report.pdf - отчет в формате PDF - report.docx - отчет в формате DOCX

Ссылка на репозиторий GitHub:

https://github.com/username/study_2025-2026_arch-pc

Конец отчета