

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: Армихос Гонзалез Карла

Группа: НКАбд-02-24

МОСКВА

2024 г.

Содержание

1. Цель работы.....	4
2. Задание.....	5
3. Технические введение.....	6
4. Порядок выполнения лабораторной работы.....	8
5. Задание для самостоятельной работы.....	15
6. Выводы.....	19
7. Список используемой литературы.....	20

Список иллюстраций

рис. 4.1 Midnight Commander.....	8
рис. 4.2 Войти в каталог лабораторная работа 4.....	9
рис. 4.3 Создать файл lab05.....	10
рис. 4.4 Создать lab05-1.asm (команда touch).....	11
рис. 4.5 Открыть lab 05-1. asm (F4).....	11
рис. 4.6 Часть 1.....	11
рис. 4.7 Часть 2.....	12
рис. 4.8 Проверить, что файл содержит текст программы.....	13
рис 4.9 Исполнение файла.....	14
рис 4.10 копирование и изменение имени.....	14
рис. 5.1 Копирование файла.....	15
рис. 5.2 Редактировать текст.....	16
рис. 5.3 Запустить программу.....	16
рис 5.4 Копирование файла lab5-2.asm.....	17
рис 5.5 Ввести строку.....	17
рис 5.6 Вывести введённую строку на экран.....	18

1. Цель работы

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

2. Задание

1. Откройте Midnight Commander
2. Перейдите в каталог с помощью клавиш.
3. С создайте файл -asm.
4. Создать копию файла, переместить его в другое место, переименовать с помощью клавиш.
5. Использовать такие команды, как `nano mcedit`, для ввода текста программы из списка

3. Технические введение

Команда **mc** (Midnight Commander) — это файловый менеджер для терминала, который предоставляет удобный интерфейс для работы с файлами. Основное назначение **mc** — упростить навигацию по файловой системе и выполнение операций с файлами, таких как копирование, перемещение, удаление, создание директорий и многое другое. Он особенно полезен для тех, кто предпочитает работать в текстовом режиме и часто использует командную строку.

Некоторые операции с файлами можно выполнить как с помощью стандартных команд **bash**, так и через интерфейс **mc** с использованием сочетаний клавиш:

- Копирование файлов: В *bash* — *cp file1 file2*, в *mc* — *F5*.
- Перемещение файлов: В *bash* — *mv file1 file2*, в *mc* — *F6*.
- Удаление файлов: В *bash* — *rm file*, в *mc* — *F8*.
- Создание новых каталогов: В *bash* — *mkdir newdir*, в *mc* — *F7*.
- Эти операции могут быть выполнены с помощью командной строки или комбинаций клавиш, что делает работу с файлами в **mc** более интуитивной и удобной.

Программа на языке ассемблера **NASM** обычно состоит из нескольких секций:

- **section .data** — секция для хранения инициализированных данных.
- **section .bss** — секция для хранения неинициализированных данных.
- **section .text** — секция, где находится код программы, обычно включает точку входа **_start** или **main**, с которой начинается выполнение.

Эта структура позволяет организовать код и данные в программе, упрощая управление памятью.

Секция **.data** используется для описания *инициализированных данных*, таких как строки или константы, которые сразу известны на этапе написания программы.

- Секция **.bss** используется для описания *неинициализированных данных*, таких как переменные, которые будут инициализированы позже, во время выполнения программы.

Эти компоненты используются для определения данных разных типов и размеров в **NASM**:

- **db** (Define Byte) — задаёт байт (1 байт).
- **dw** (Define Word) — задаёт слово (2 байта).
- **dd** (Define Doubleword) — задаёт двойное слово (4 байта).
- **dq** (Define Quadword) — задаёт квадро-слово (8 байт).
- **dt** (Define Ten Bytes) — задаёт десятибайтовое значение (10 байт).

Эти компоненты помогают задавать данные разных типов и размеров в зависимости от требований программы.

- Инструкция **mov eax, esi** копирует значение регистра **esi** в регистр **eax**. При этом содержимое **esi** остаётся неизменным, а значение **eax** перезаписывается.

- Инструкция **int 80h** используется для вызова системных прерываний в Linux. Она позволяет программе взаимодействовать с операционной системой для выполнения различных системных вызовов, таких как чтение, запись, открытие файлов и завершение программы. Регистр **eax** обычно содержит номер системного вызова, а другие регистры передают необходимые параметры.

4. Порядок выполнения лабораторной работы

Откройте Midnight Commander “mc” (рис. 4.1)

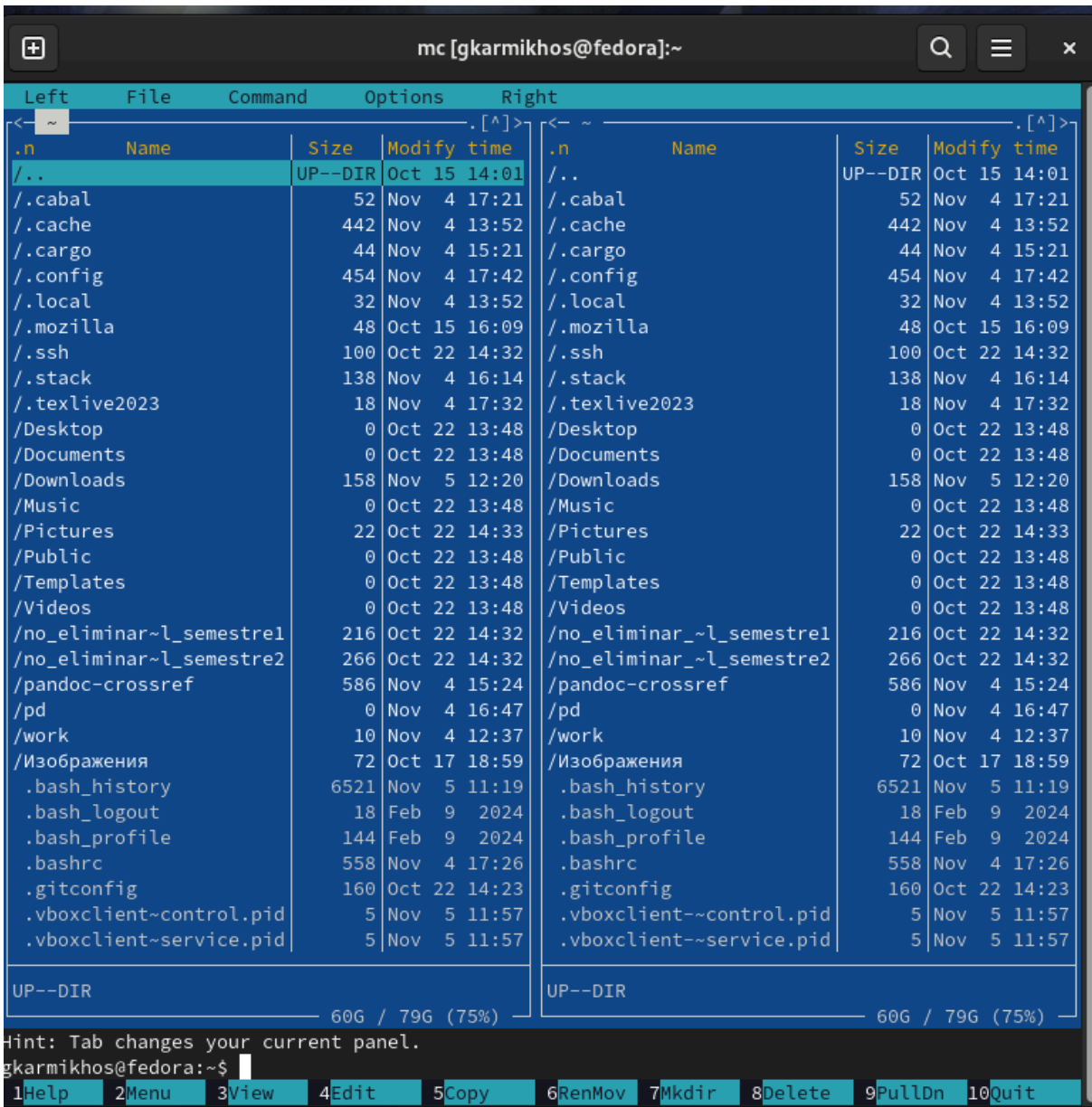


рис. 4.1 Midnight Commander

С помощью клавиш ↑, ↓ и Enter заходим в каталог work/arch/pc Что создается в лаборатории 4 (рис. 4.2)



Создать файл с помощью клавиши f7 с именем lab 05 (рис.4.3)

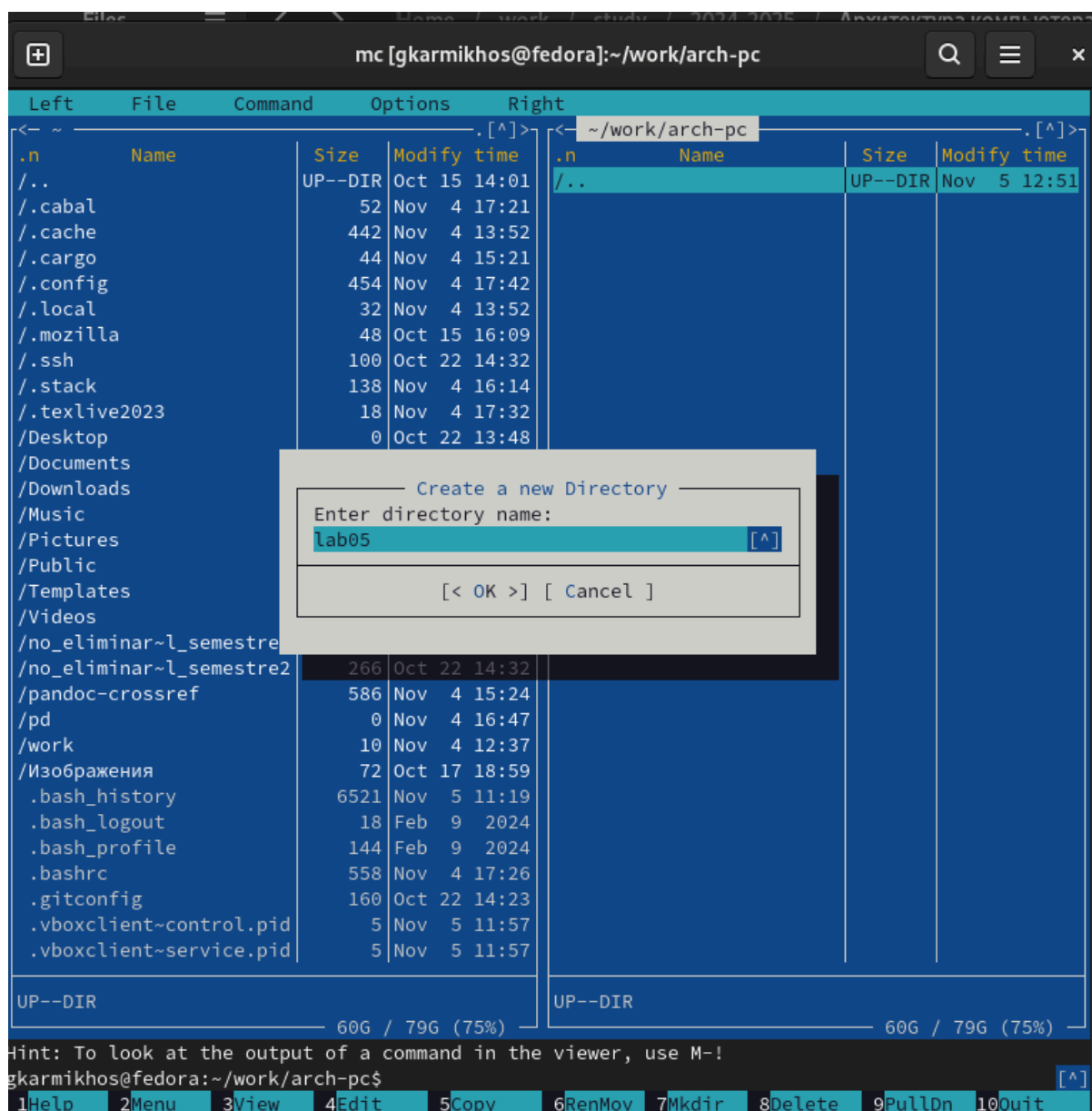


рис. 4.3 Создать файл lab05

Внутри созданной папки мы создаем lab05-1.asm, набрав команду touch в строке ввода. (рис. 4.4)

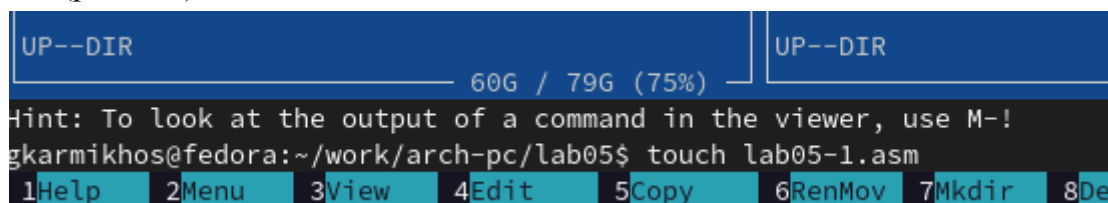


рис. 4.4 Создать lab05-1.asm (команда touch)

Откройте файл lab05-1.asm с помощью клавиши F4, чтобы иметь возможность редактировать его (рис. 4.5)

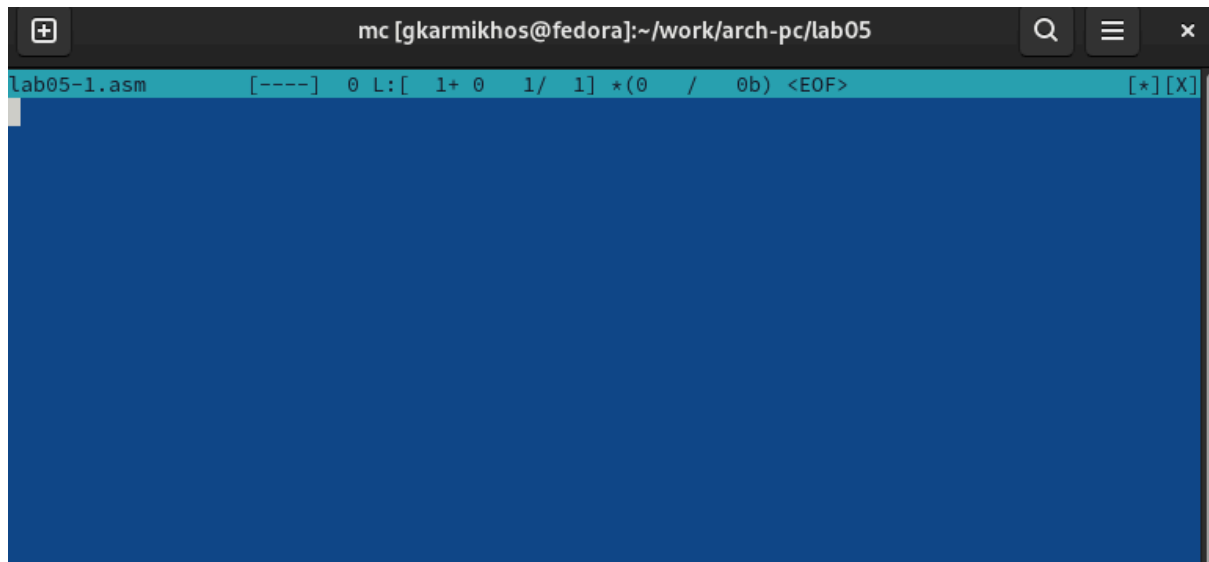


рис. 4.5 Открыть lab 05-1. asm (F4)

Введите текст программы из листинга, который отображается на следующих изображениях (рис. 4.6 и рис 4.7)

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 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
```

рис. 4.6 Часть 1

```

;----- 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

```

рис. 4.7 Часть 2

Убедитесь, что то, что мы вводим с помощью клавиши F3, было успешно сохранено (рис. 4.8)

```
mc [gkarmikhos@fedora]:~/work/arch-pc/lab05
/home/gkarmikhos/work/arch-pc/lab05/lab05-1.asm 2663/2707 98%
; 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 программ
;
;----- 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)
1Help      2UnWrap  3Quit  4Hex      5Goto  6      7Search  8Raw      9Format 10Quit
```

рис. 4.8 Проверить, что файл содержит текст программы

Открой в файле lab5-1.asm. Выполните компоновку объектного файла и запустите получившийся исполняемый файл. Выводит строку 'Введите строку:' введите ваши ФИО.(рис 4.9)

```

gkarmikhos@fedora:~$ cd work/arch-pc/
gkarmikhos@fedora:~/work/arch-pc$ cd lab05
gkarmikhos@fedora:~/work/arch-pc/lab05$ nasm -f elf lab5-1.asm
gkarmikhos@fedora:~/work/arch-pc/lab05$ ld -m elf_i386 -o lab5-1 lab5-1.o
gkarmikhos@fedora:~/work/arch-pc/lab05$ .lab5-1/
bash: .lab5-1/: No such file or directory
gkarmikhos@fedora:~/work/arch-pc/lab05$ ./lab5-1
Введите строку:
Карла
gkarmikhos@fedora:~/work/arch-pc/lab05$

```

рис 4.9 Исполнение файла

Создайте копию файла lab5-1.asm с именем lab5-2.asm.(рис 4.10)

~ /work/arch-pc/lab05		.[^]>	
.n	Name	Size	Modify time
./..		UP--DIR	Nov 7 22:32
	in_out.asm	3942	Nov 5 12:20
*lab5-1		8744	Nov 8 17:45
	lab5-1.asm	2707	Nov 7 22:14
	lab5-1.o	752	Nov 8 17:44
	lab5-2.asm	2707	Nov 7 22:14

рис 4.10 копирование и изменение имени

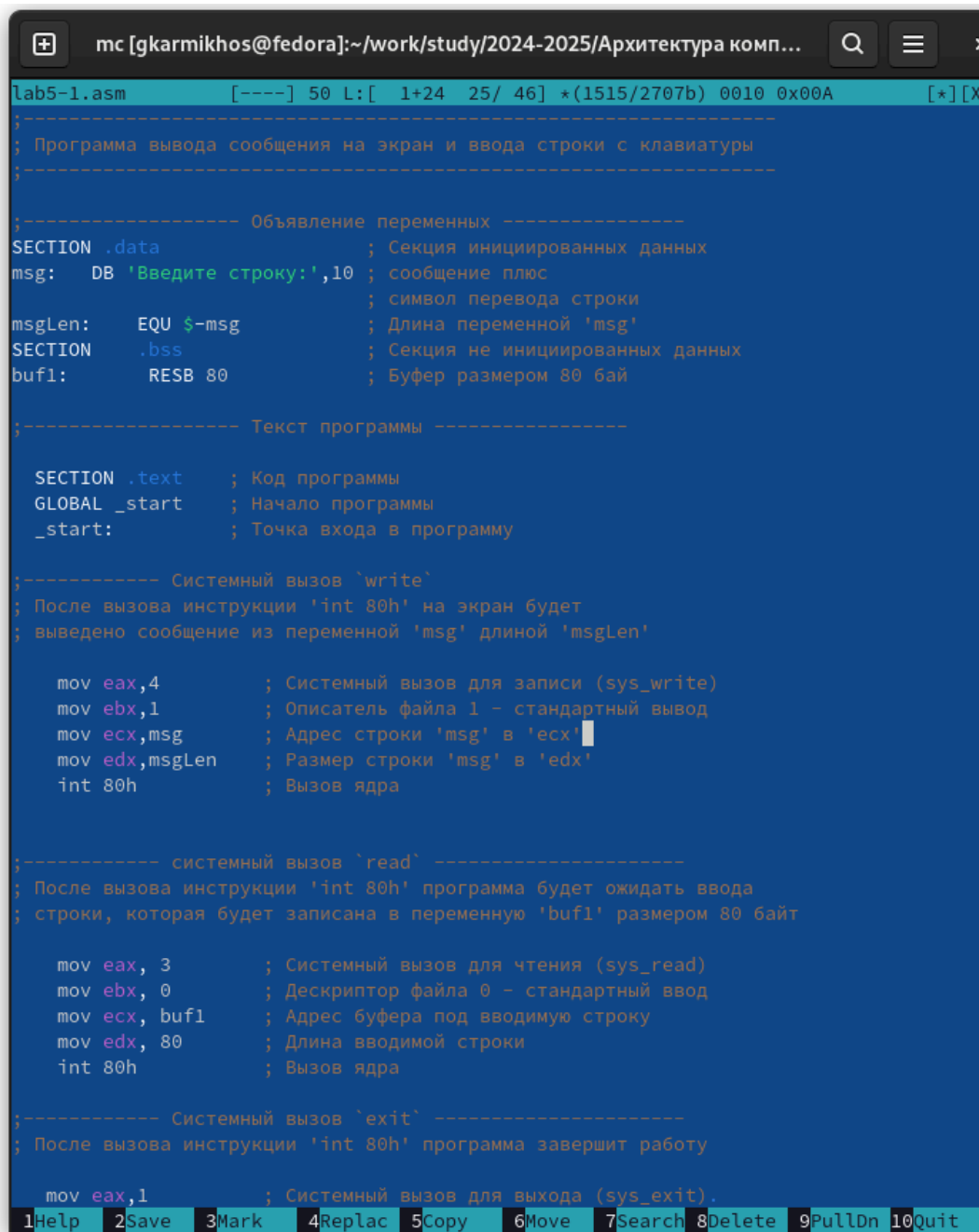
5. Задание для самостоятельной работы

Создать копию файла lab5-1.asm с помощью функциональной клавиши F5 (рис. 5.1)

Left	File	Command	Options	Right
<code><- ...ьютера/arch-pc/labs/lab05 -.^[></code>				<code><- ~/work/arch-pc/lab05 -.^[></code>
	Name	Size	Modify time	.n Name Size Modify time
	/..	UP--DIR	Nov 4 13:08	/.. UP--DIR Nov 7 22:32
	/report	82	Nov 7 22:16	in_out.asm 3942 Nov 5 12:20
	/presentation	100	Nov 4 13:08	*lab5-1 8744 Nov 8 17:45
	lab5-1.asm	2707	Nov 7 22:14	lab5-1.o 752 Nov 8 17:44
				lab5-2.asm 1384 Nov 8 18:08

рис. 5.1 Копирование файла

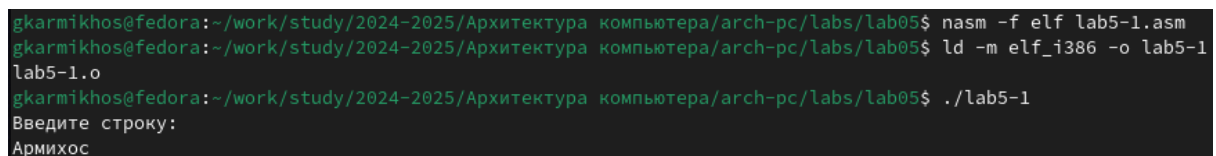
Отредактируйте текст, чтобы при его запуске отображались мои имя и фамилия(рис. 5.2)



```
lab5-1.asm [----] 50 L: [ 1+24 25/ 46] *(1515/2707b) 0010 0x00A [*][X]
;
; 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 ;Descriptor 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).
```

рис. 5.2 Редактировать текст

Воспроизвести отредактированный текст.(рис. 5.3)



```
gkarmikhos@fedora:~/work/study/2024-2025/Архитектура компьютера/arch-pc/labs/lab05$ nasm -f elf lab5-1.asm
gkarmikhos@fedora:~/work/study/2024-2025/Архитектура компьютера/arch-pc/labs/lab05$ ld -m elf_i386 -o lab5-1
lab5-1.o
gkarmikhos@fedora:~/work/study/2024-2025/Архитектура компьютера/arch-pc/labs/lab05$ ./lab5-1
Введите строку:
Армихос
```

рис. 5.3 Запустить программу

Создайте копия файла lab5-2.asm (рис 5.4)

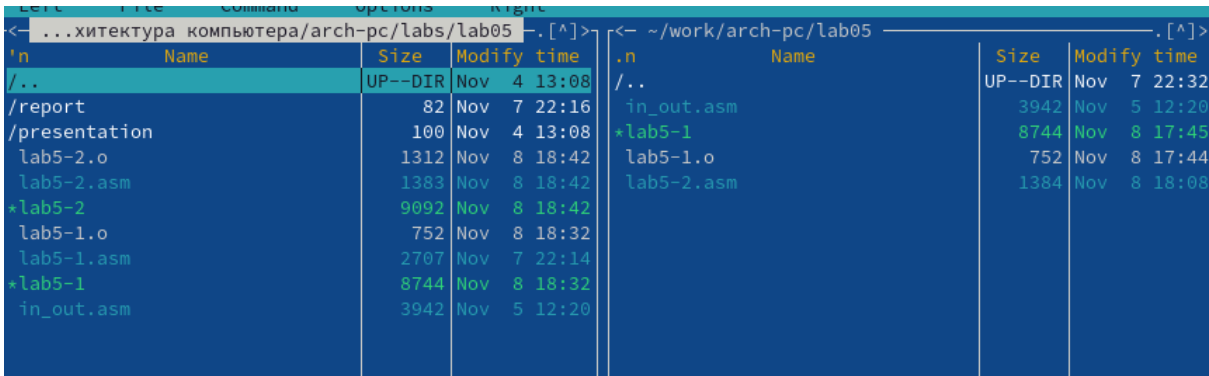


рис 5.4 Копирование файла lab5-2.asm

Отобразить текстовую строку в болоте с помощью клавиши F3 (рис 5.5)

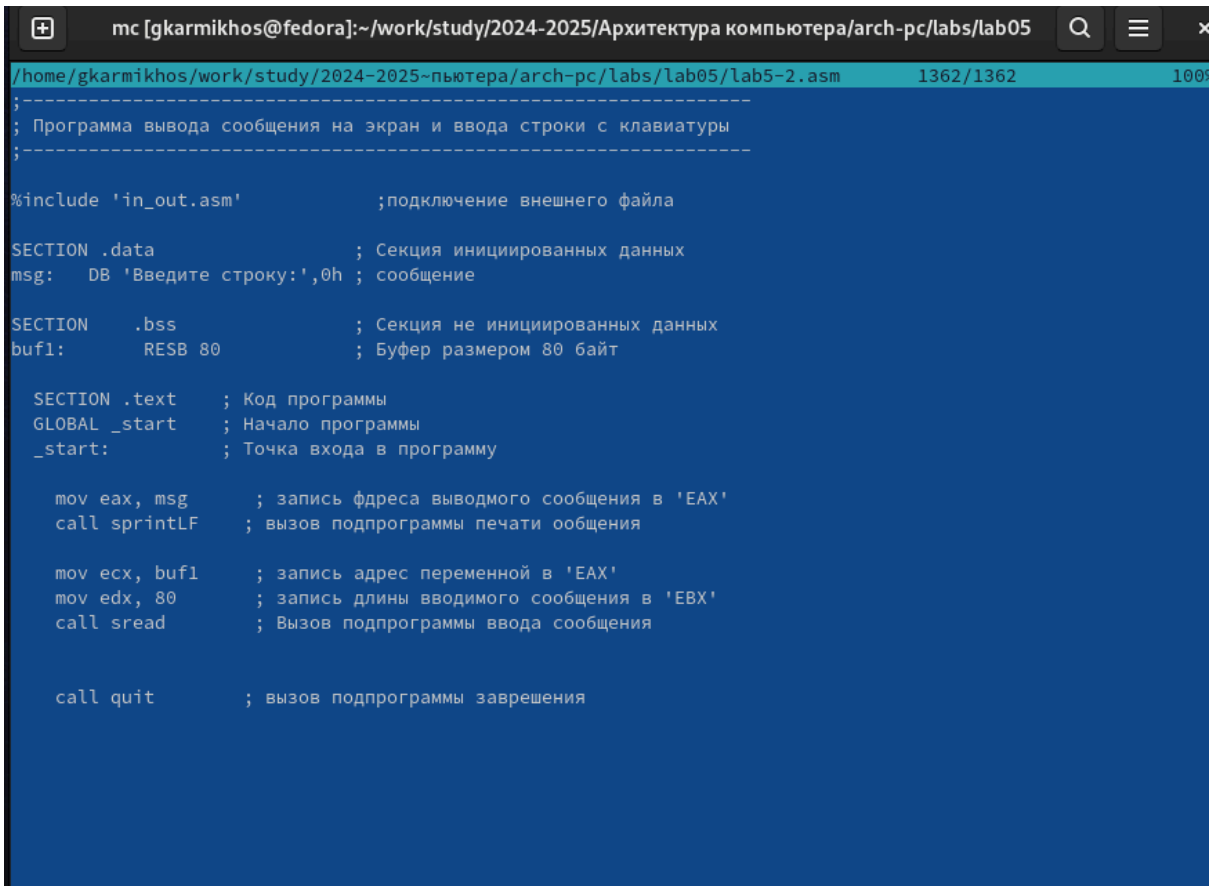


рис 5.5 Ввести строку

На приглашение ввести строку введите свою фамилию(рис 5.6)

```
gkarmikhos@fedora:~/work/study/2024-2025/Архитектура компьютера/arch-pc/labs/lab05$ nasm -f elf lab5-2.asm
gkarmikhos@fedora:~/work/study/2024-2025/Архитектура компьютера/arch-pc/labs/lab05$ ld -m elf_i386 -o lab5-2
lab5-2.o
gkarmikhos@fedora:~/work/study/2024-2025/Архитектура компьютера/arch-pc/labs/lab05$ ./lab5-2
Введите строку:
Армихос Гонзалез Карла
gkarmikhos@fedora:~/work/study/2024-2025/Архитектура компьютера/arch-pc/labs/lab05$
```

рис 5.6 Вывести введённую строку на экран

6. Выводы

В общем, **mc** - это инструмент управления файлами в терминальном режиме, позволяющий выполнять ряд обычных операций, таких как копирование, перемещение и удаление файлов, как с помощью команд `bash`, так и с помощью определенных комбинаций клавиш. Что касается языка ассемблера NASM, то его структура состоит из ключевых секций (**.data**, **.bss** и **.text**), которые позволяют определять различные типы данных и кода, оптимизируя использование памяти.

В этой лаборатории мы смогли познакомиться с работой NASM и Midnight Commander.

Зная функции клавиш F (1, 2, 5...10), мы смогли получить доступ к программе и работать над ней, а также использовать команды, ранее использовавшиеся в других лабораториях, такие как `touch`, `nano`.

7. Список используемой литературы

- 1) Архитектура ЭВМ