

Отчет по лабораторной работе №5

Дисциплина: архитектура компьютера

Акмурадов Тимур

Содержание

Цель работы	1
Задание	1
Теоретическое введение	1
Выполнение лабораторной работы	2
Основы работы с mc	2
Структура программы на языке ассемблера NASM	4
Подключение внешнего файла	8
Выполнение заданий для самостоятельной работы	12
Выводы	17
Список литературы	17

Цель работы

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

Задание

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

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 инициализированных 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 (четверное 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.

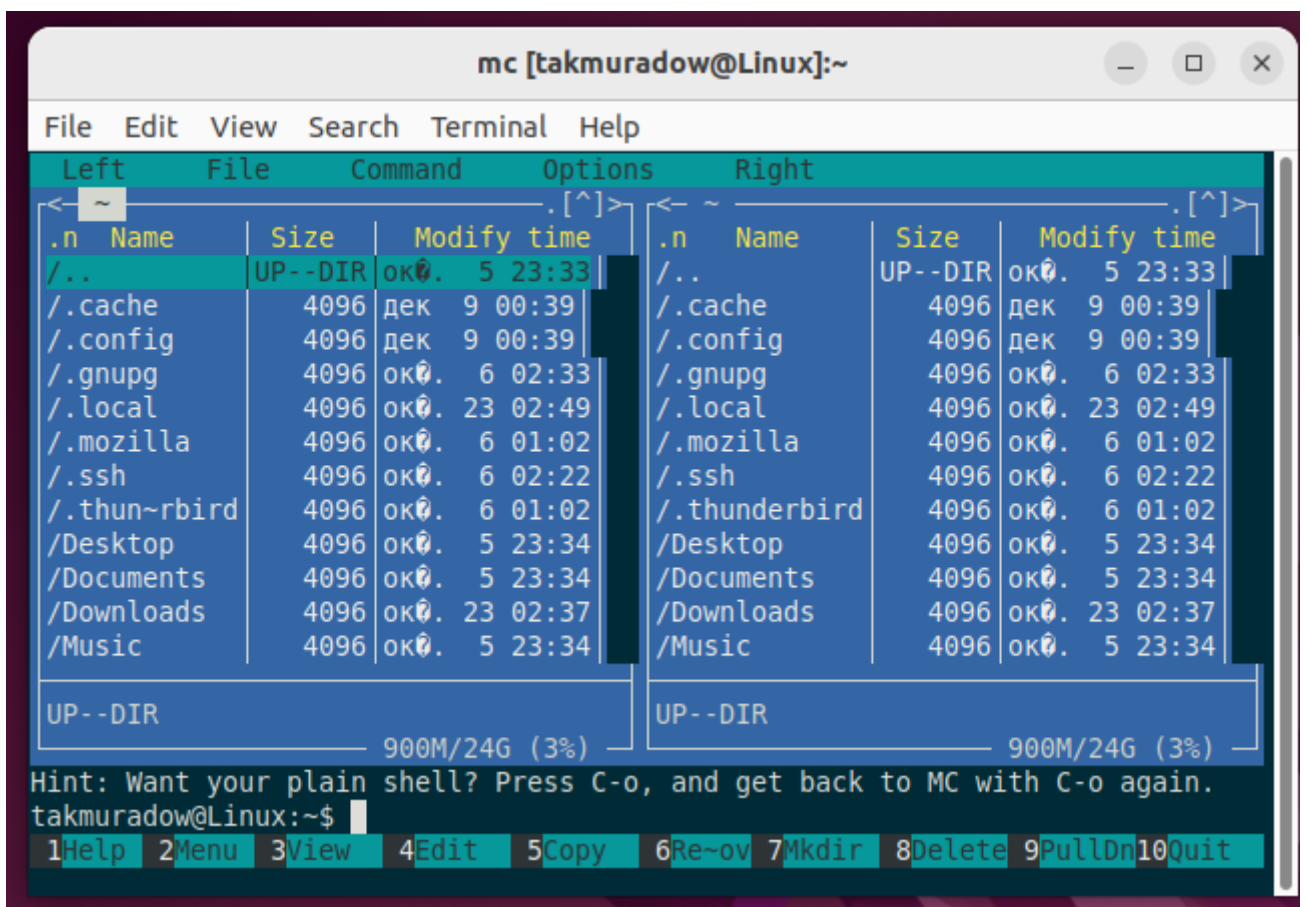
```
int n
```

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).

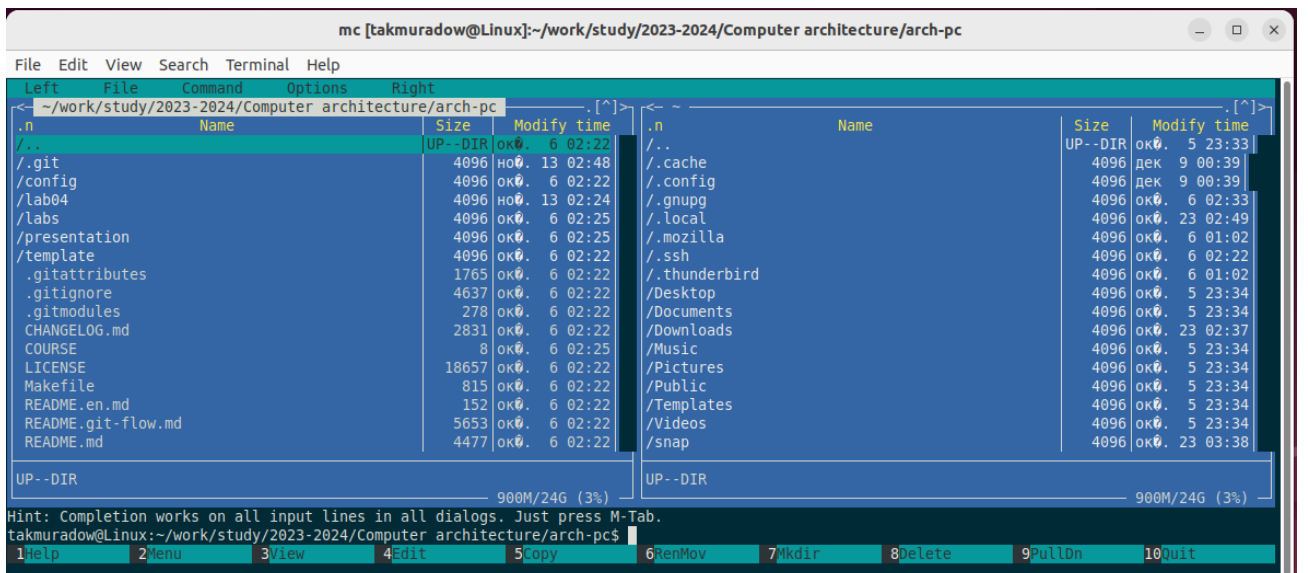
Выполнение лабораторной работы

Основы работы с mc

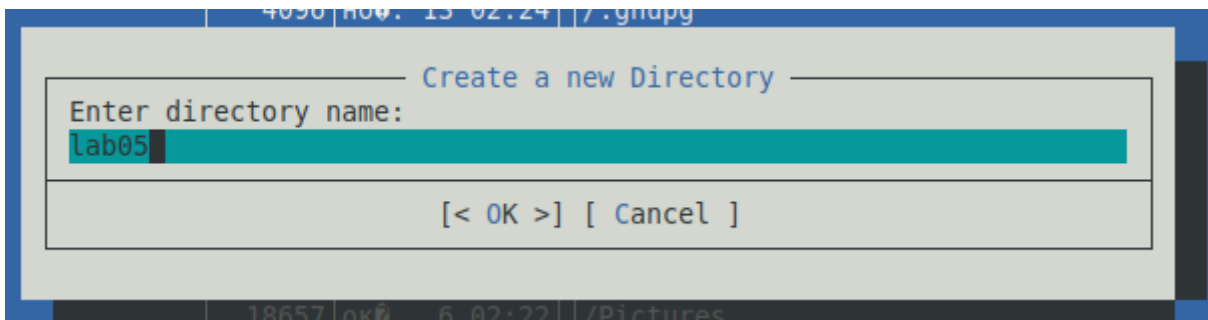
Открываю Midnight Commander, введя в терминал mc (рис. 01).



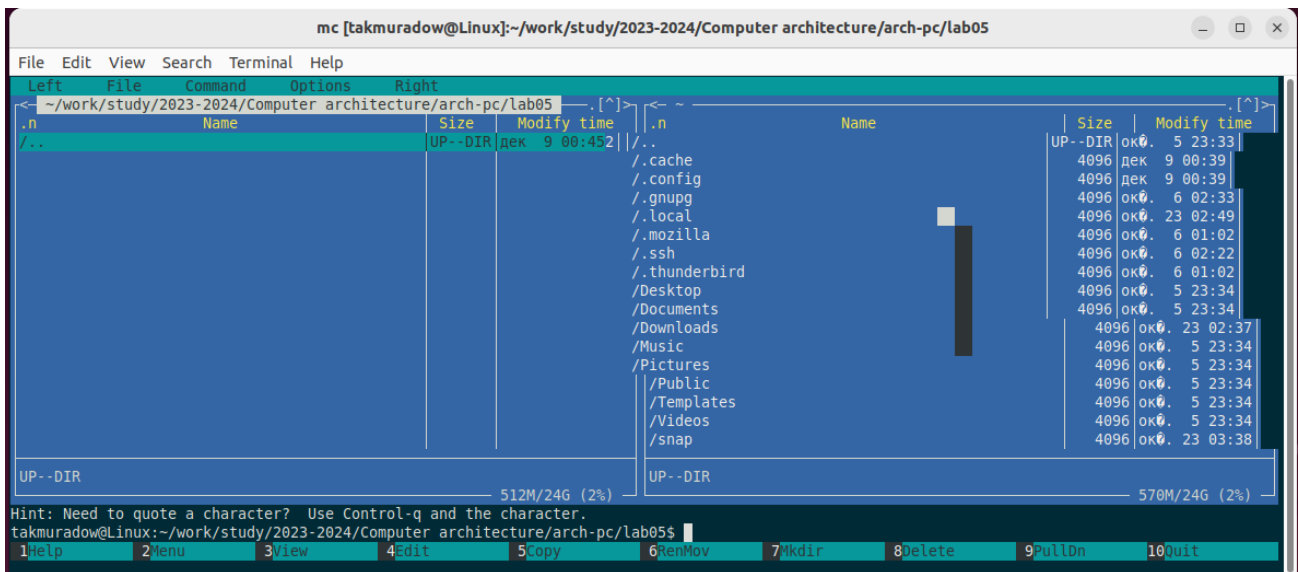
Перехожу в каталог ~/work/study/2022-2023/Архитектура Компьютера/arch-рс, используя файловый менеджер mc (рис. 02)



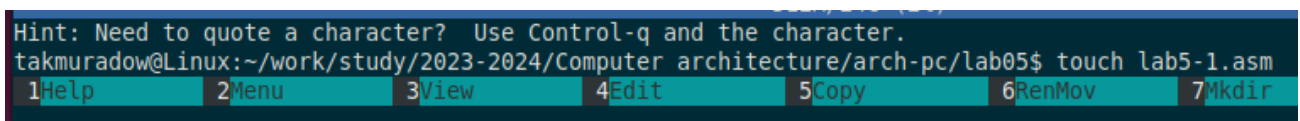
С помощью функциональной клавиши F7 создаю каталог lab05 (рис. 03).



Переходу в созданный каталог (рис. 05).

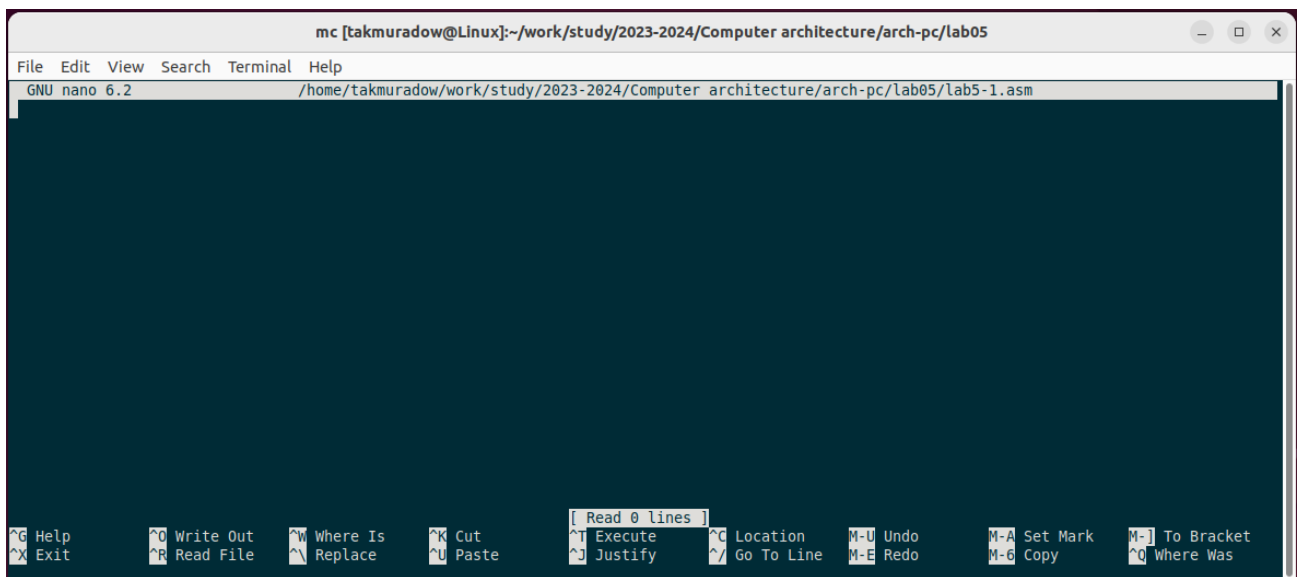


В строке ввода прописываю команду `touch lab5-1.asm`, чтобы создать файл, в котором буду работать (рис. 06).



Структура программы на языке ассемблера NASM

С помощью функциональной клавиши F4 открываю созданный файл для редактирования в редакторе nano (рис. 07).

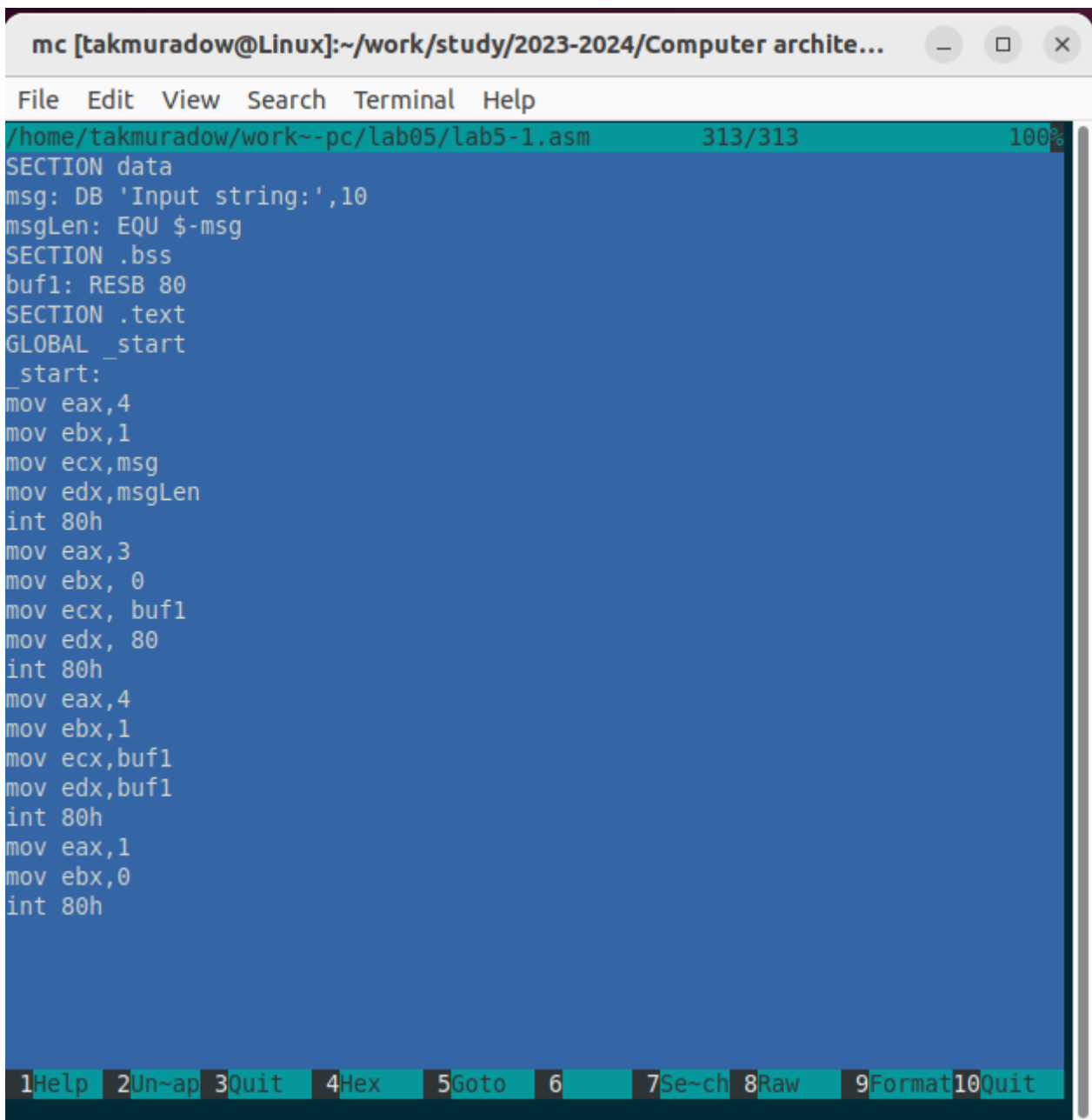


Ввожу в файл код программы для запроса строки у пользователя (рис. 08). Далее выхожу из файла (Ctrl+X), сохраняя изменения (Y, Enter).

```
mc [takmuradow@Linux]:~/work/study/2023-2024/Computer archite...
File Edit View Search Terminal Help
./rk/study/2023-2024/Computer architecture/arch-pc/lab05/lab5-1.asm *
SECTION data
msg: DB 'Input string:',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
mov ebx,0
int 80h

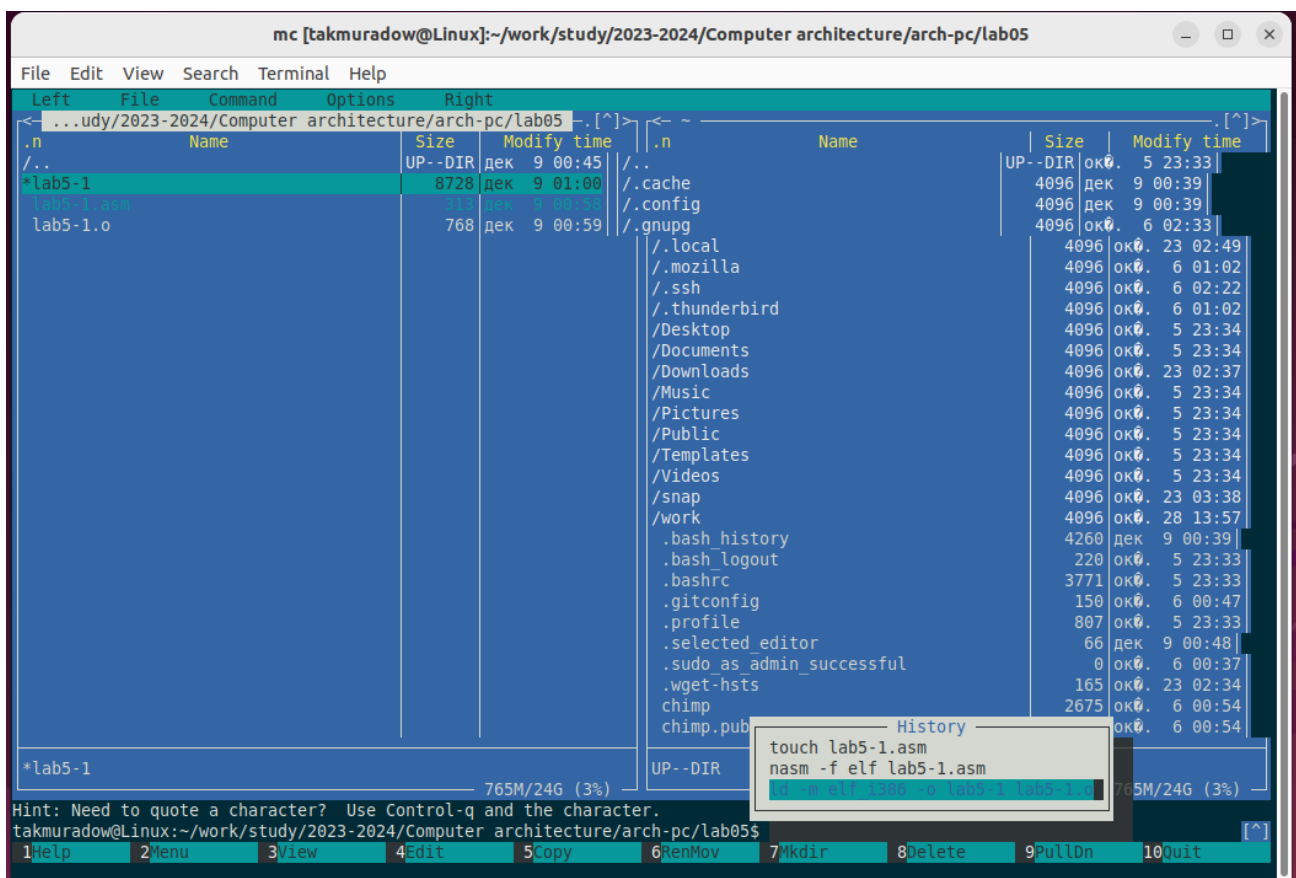
^G Help      ^O Write Out  ^W Where Is   ^K Cut        ^T Execute
^X Exit      ^R Read File  ^\ Replace    ^U Paste      ^J Justify
```

помощью функциональной клавиши F3 открываю файл для просмотра, чтобы проверить, содержит ли файл текст программы (рис. 09).

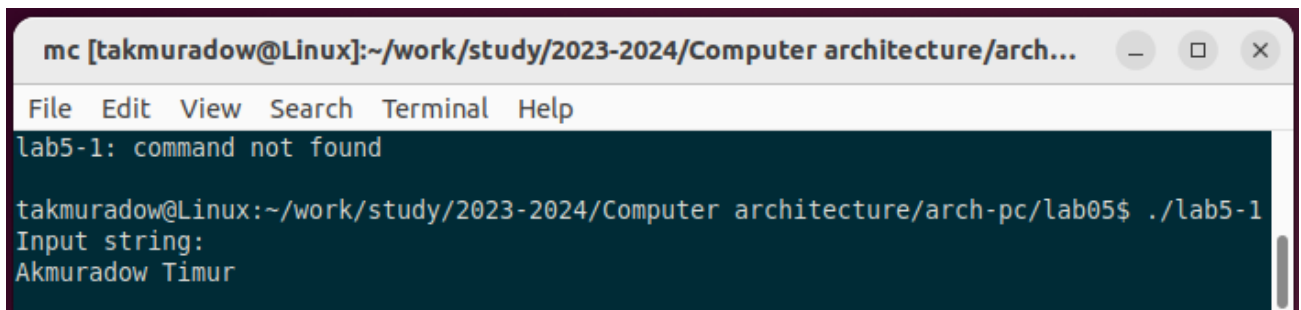


```
mc [takmuradow@Linux]:~/work/study/2023-2024/Computer archite...
File Edit View Search Terminal Help
/home/takmuradow/work~pc/lab05/lab5-1.asm 313/313 100%
SECTION data
msg: DB 'Input string:',10
msgLen: EQU $-msg
SECTION .bss
buf1: RESB 80
SECTION .text
GLOBAL _start
_start:
mov eax,4
mov ebx,1
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
mov ebx,0
int 80h
1Help 2Un~ap 3Quit 4Hex 5Goto 6 7Se~ch 8Raw 9Format10Quit
```

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` (рис. 10). Создался исполняемый файл `lab5-1`.

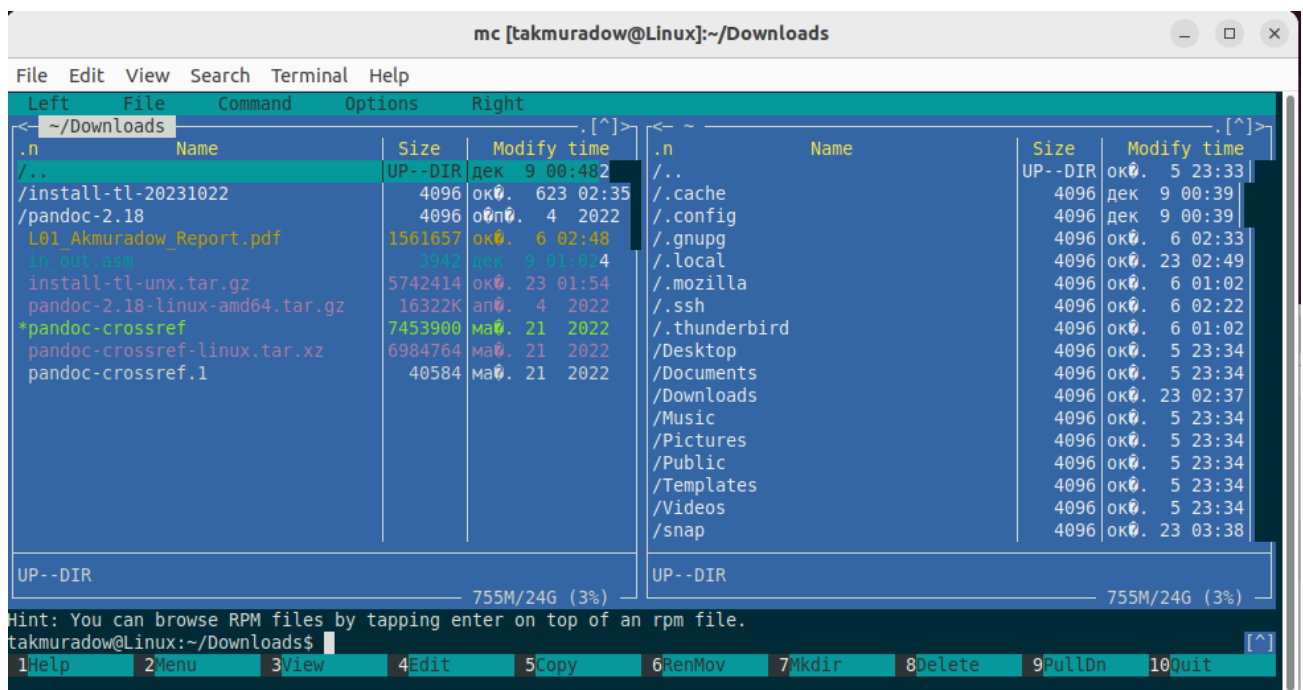


Запускаю исполняемый файл. Программа выводит строку “Введите строку:” и ждет ввода с клавиатуры, я ввожу свои ФИО, на этом программа заканчивает свою работу (рис. 11).

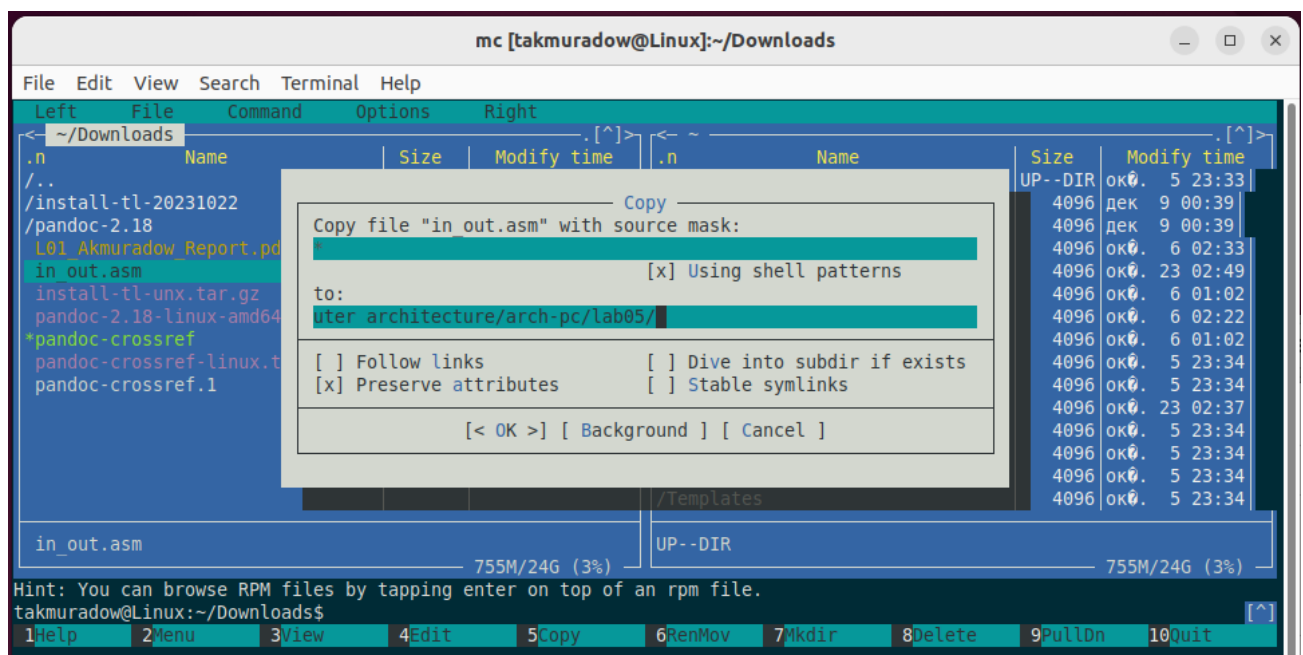


Подключение внешнего файла

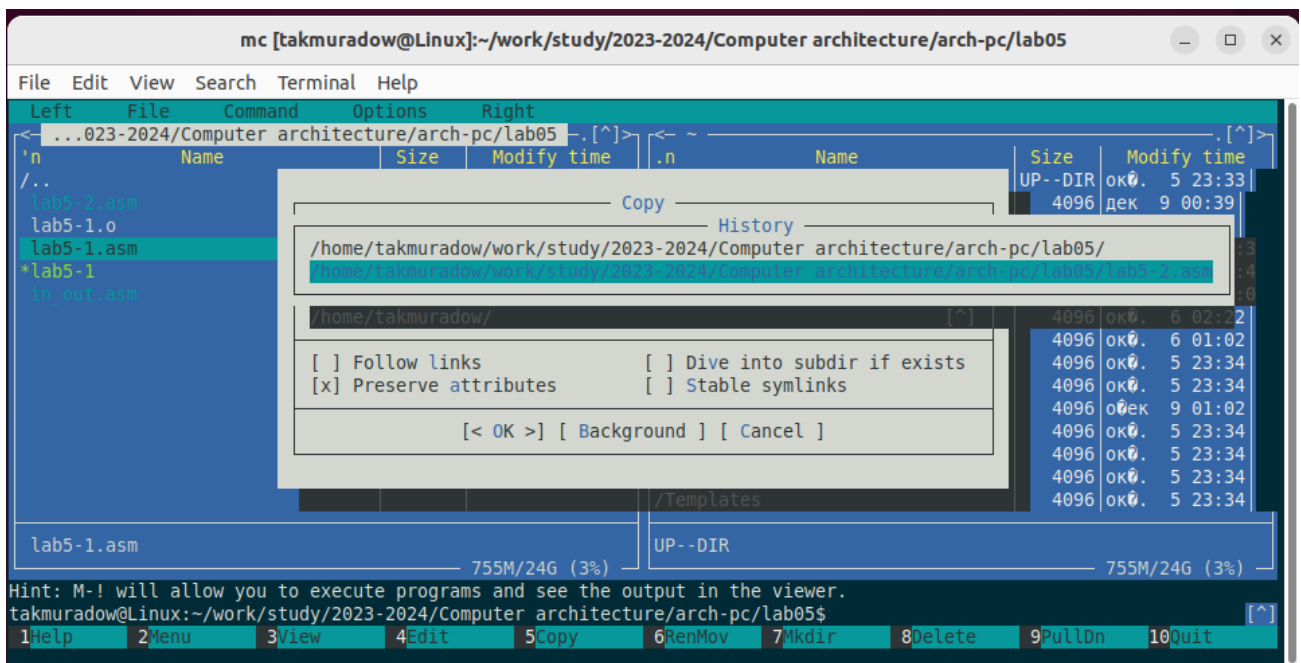
Скачиваю файл in_out.asm со страницы курса в ТУИС. Он сохранился в каталог “Загрузки” (рис. 12).



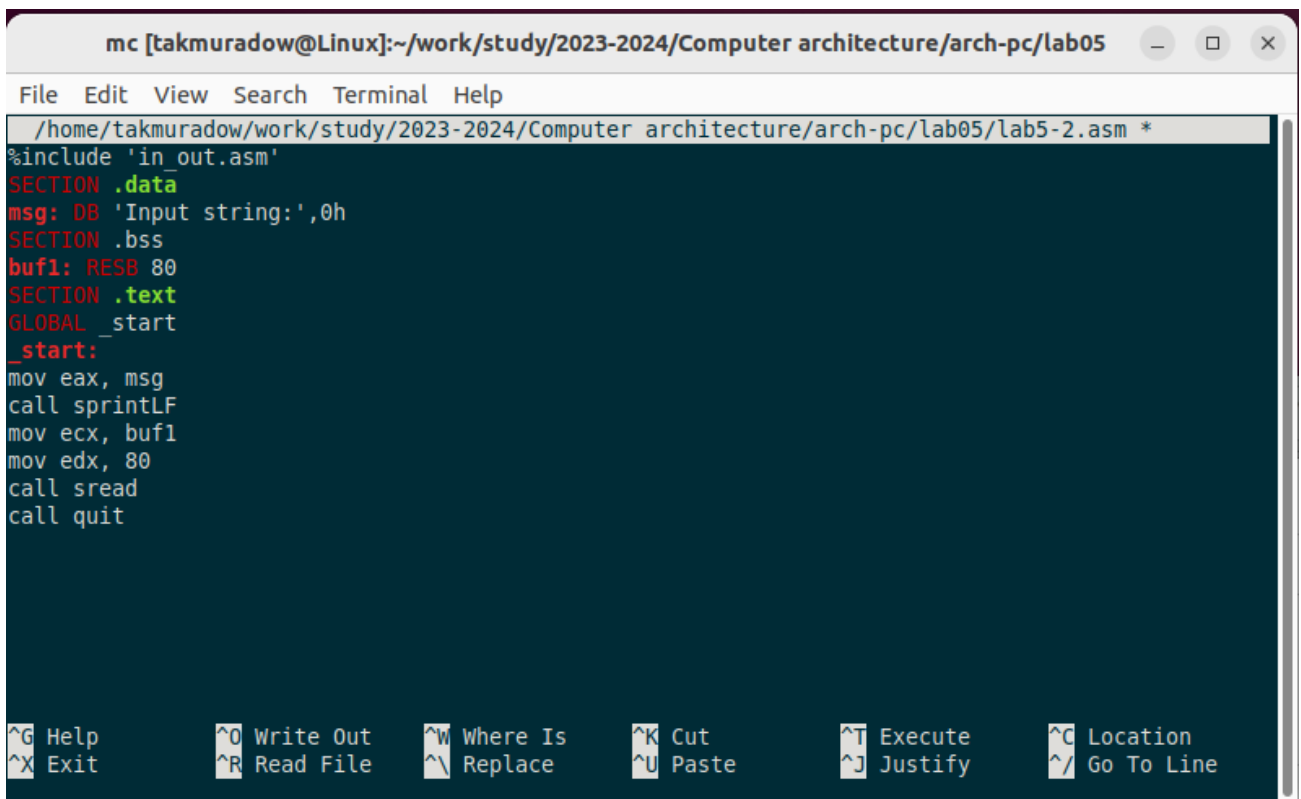
С помощью функциональной клавиши F5 копирую файл in_out.asm из каталога Загрузки в созданный каталог lab05 (рис. 13).



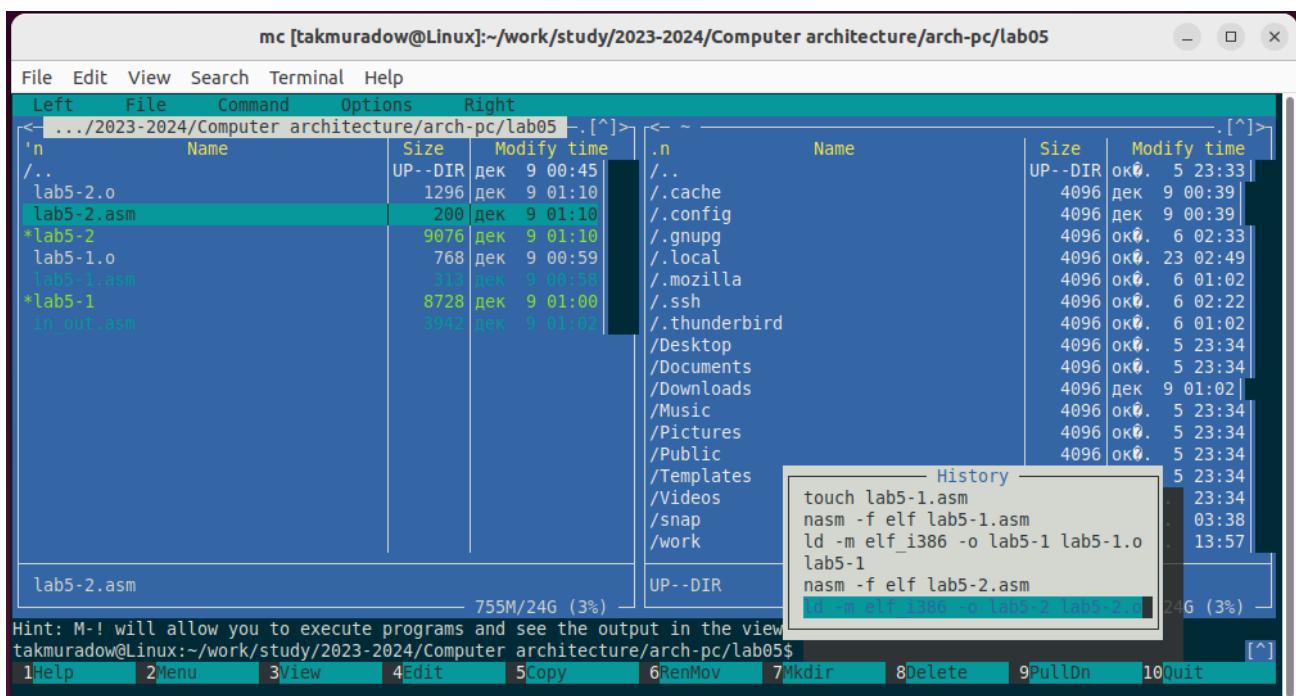
С помощью функциональной клавиши F5 копирую файл lab5-1 в тот же каталог, но с другим именем, для этого в появившемся окне mc прописываю имя для копии файла (рис. 14).



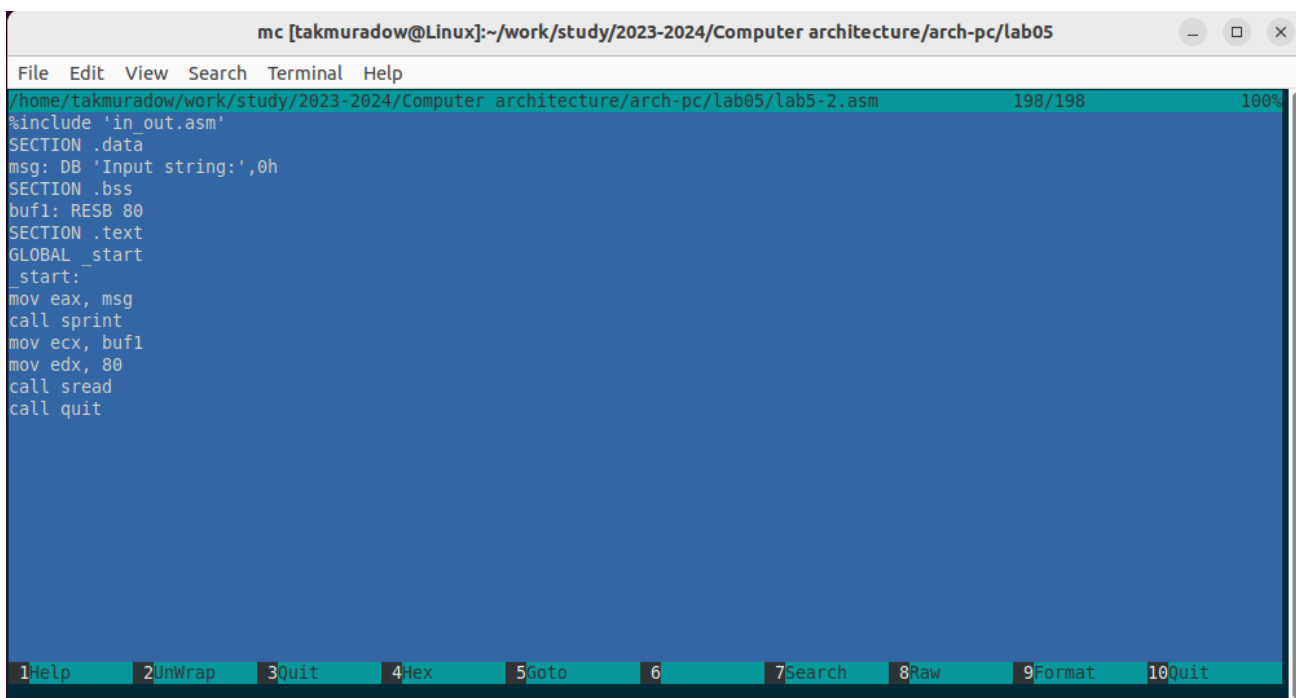
Изменяю содержимое файла lab5-2.asm во встроенном редакторе nano (рис. 15), чтобы в программе использовались подпрограммы из внешнего файла in_out.asm.



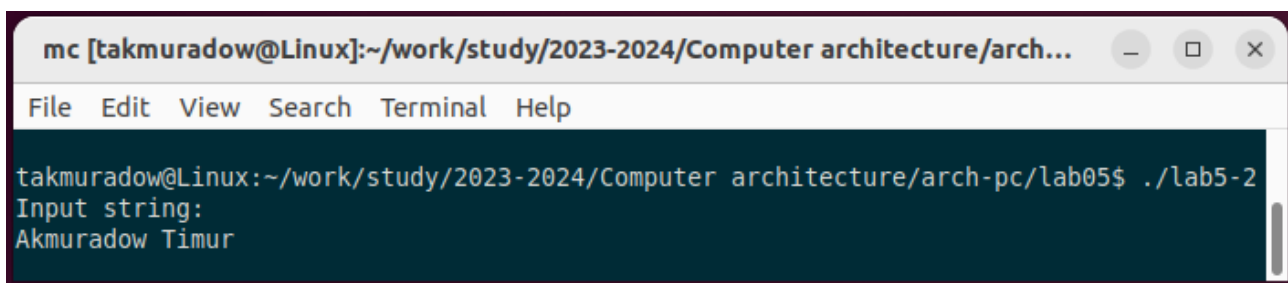
Транслирую текст программы файла в объектный файл командой `nasm -f elf lab5-2.asm`. Создался объектный файл lab5-2.o. Выполняю компоновку объектного файла с помощью команды `ld -m elf_i386 -o lab5-2 lab5-2.o` Создался исполняемый файл lab5-2. Запускаю исполняемый файл (рис. 16).



Открываю файл lab5-2.asm для редактирования в nano функциональной клавишей F4. Изменяю в нем подпрограмму sprintLF на sprint. Сохраняю изменения и открываю файл для просмотра, чтобы проверить сохранение действий (рис. 17).



Снова транслирую файл, выполняю компоновку созданного объектного файла, запускаю новый исполняемый файл (рис. 18).



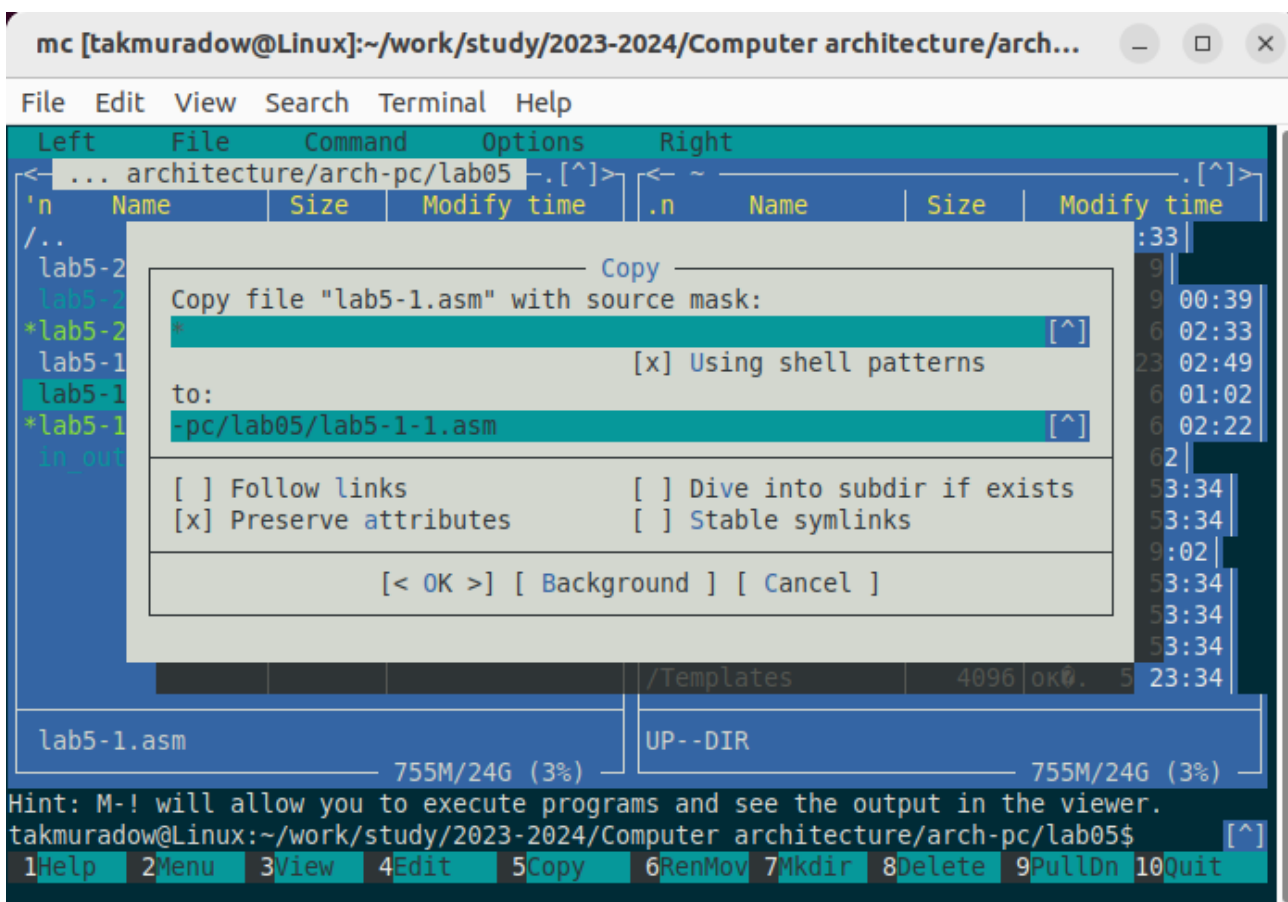
```
mc [takmuradow@Linux]:~/work/study/2023-2024/Computer architecture/arch...
File Edit View Search Terminal Help

takmuradow@Linux:~/work/study/2023-2024/Computer architecture/arch-pc/lab05$ ./lab5-2
Input string:
Akmuradow Timur
```

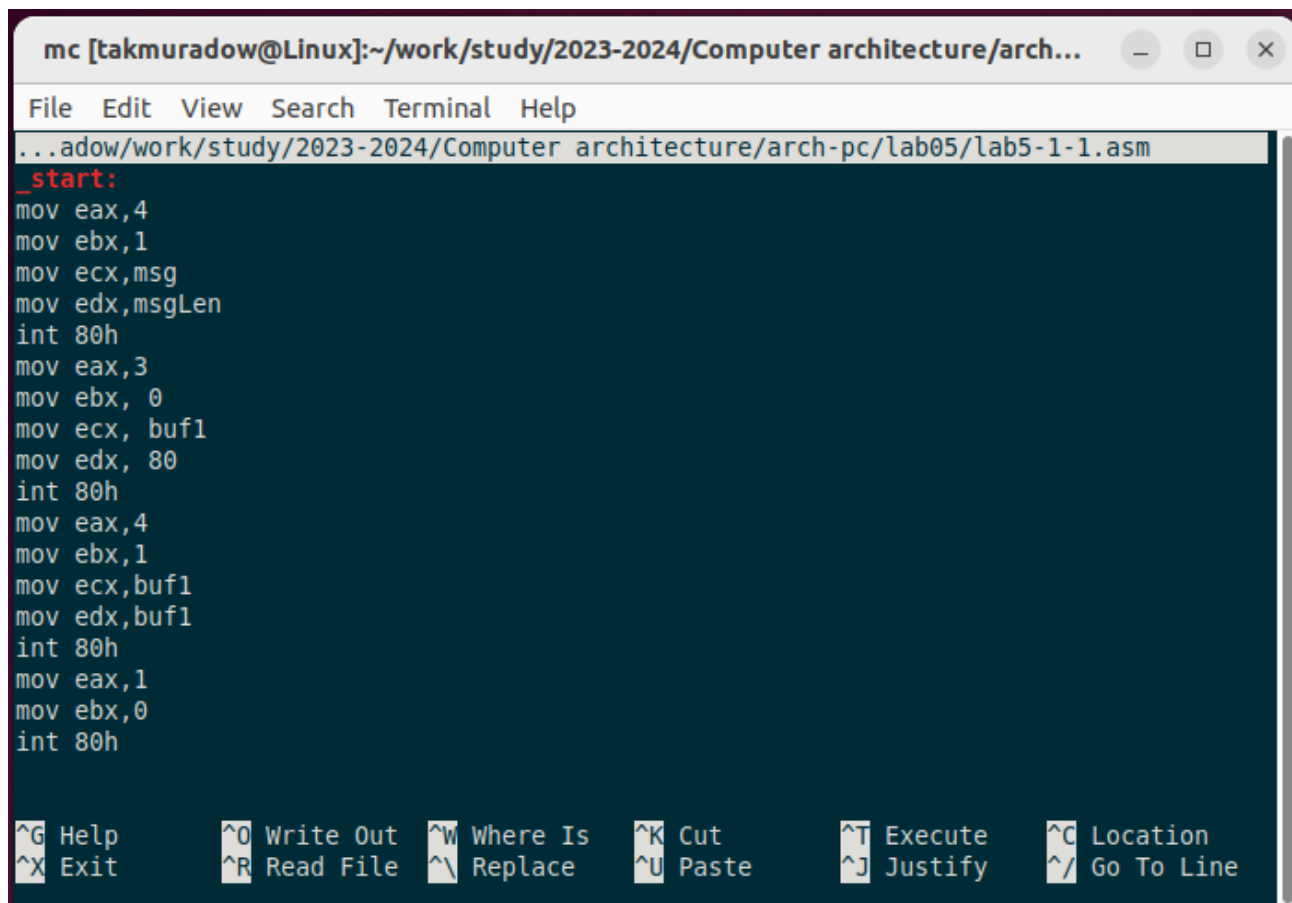
Разница между первым исполняемым файлом lab5-2 и вторым lab5-2-2 в том, что запуск первого запрашивает ввод с новой строки, а программа, которая выполняется при запуске второго, запрашивает ввод без переноса на новую строку, потому что в этом заключается различие между подпрограммами `sprintLF` и `sprint`.

Выполнение заданий для самостоятельной работы

1. Создаю копию файла lab5-1.asm с именем lab5-1-1.asm с помощью функциональной клавиши F5 (рис. 19).



С помощью функциональной клавиши F4 открываю созданный файл для редактирования. Изменяю программу так, чтобы кроме вывода приглашения и запроса ввода, она выводила вводимую пользователем строку (рис. 20).

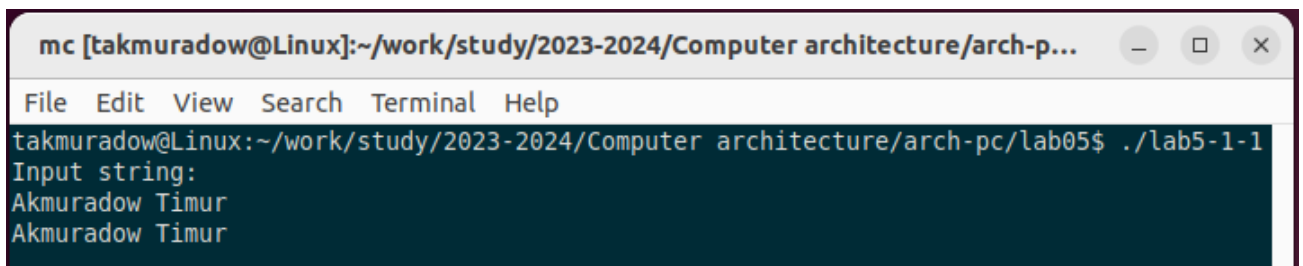


```
mc [takmuradow@Linux]:~/work/study/2023-2024/Computer architecture/arch-...
File Edit View Search Terminal Help
...adow/work/study/2023-2024/Computer architecture/arch-pc/lab05/lab5-1-1.asm
_start:
mov eax,4
mov ebx,1
mov ecx,msg
mov edx,msgLen
int 80h
mov eax,3
mov ebx, 0
mov ecx, buf1
mov edx, 80
int 80h
mov eax,4
mov ebx,1
mov ecx,buf1
mov edx,buf1
int 80h
mov eax,1
mov ebx,0
int 80h

^G Help      ^O Write Out  ^W Where Is   ^K Cut        ^T Execute    ^C Location
^X Exit      ^R Read File  ^\ Replace    ^U Paste      ^J Justify    ^_ Go To Line
```

2.

Создаю объектный файл lab5-1-1.o, отдаю его на обработку компоновщику, получаю исполняемый файл lab5-1-1, запускаю полученный исполняемый файл. Программа запрашивает ввод, ввожу свои ФИО, далее программа выводит введенные мною данные (рис. 21).



```
mc [takmuradow@Linux]:~/work/study/2023-2024/Computer architecture/arch-p...
File Edit View Search Terminal Help
takmuradow@Linux:~/work/study/2023-2024/Computer architecture/arch-pc/lab05$ ./lab5-1-1
Input string:
Akmuradow Timur
Akmuradow Timur
```

Код программы из пункта 1:

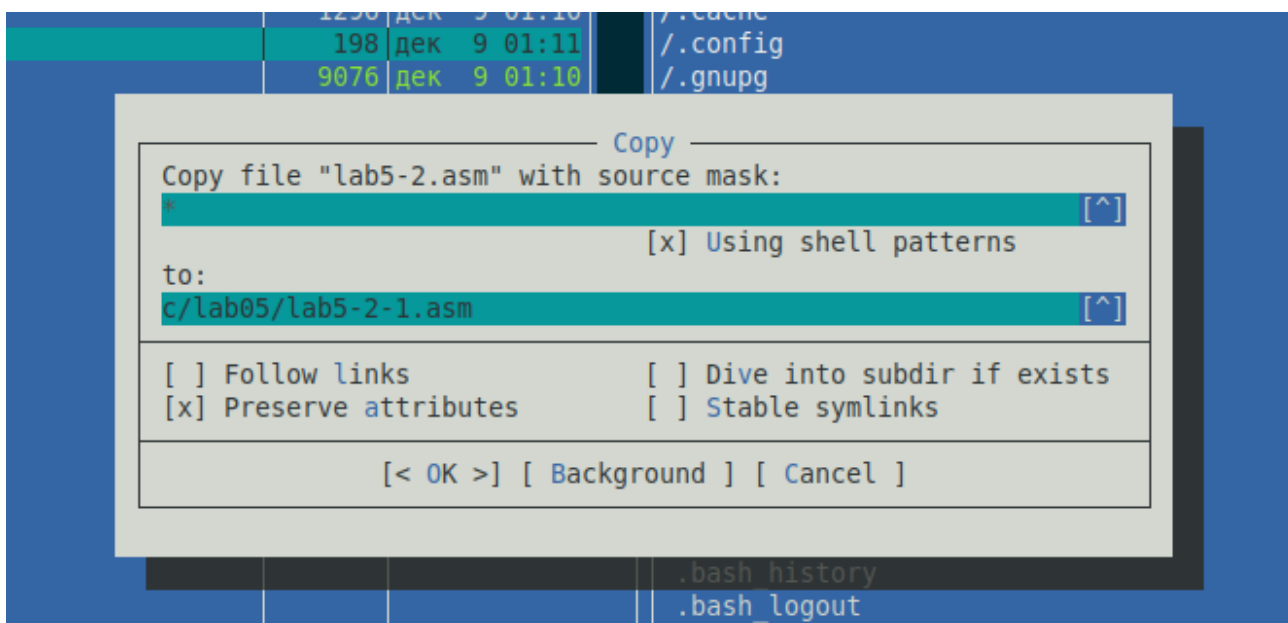
```
SECTION .data ; Секция иницированных данных
msg: DB 'Введите строку:',10
msgLen: EQU $-msg ; Длина переменной 'msg'
SECTION .bss ; Секция не иницированных данных
buf1: RESB 80 ; Буфер размером 80 байт
SECTION .text ; Код программы
GLOBAL _start ; Начало программы
_start: ; Точка входа в программу
mov eax,4 ; Системный вызов для записи (sys_write)
mov ebx,1 ; Описание файла 1 - стандартный вывод
mov ecx,msg ; Адрес строки 'msg' в 'ecx'
mov edx,msgLen ; Размер строки 'msg' в 'edx'
```

```

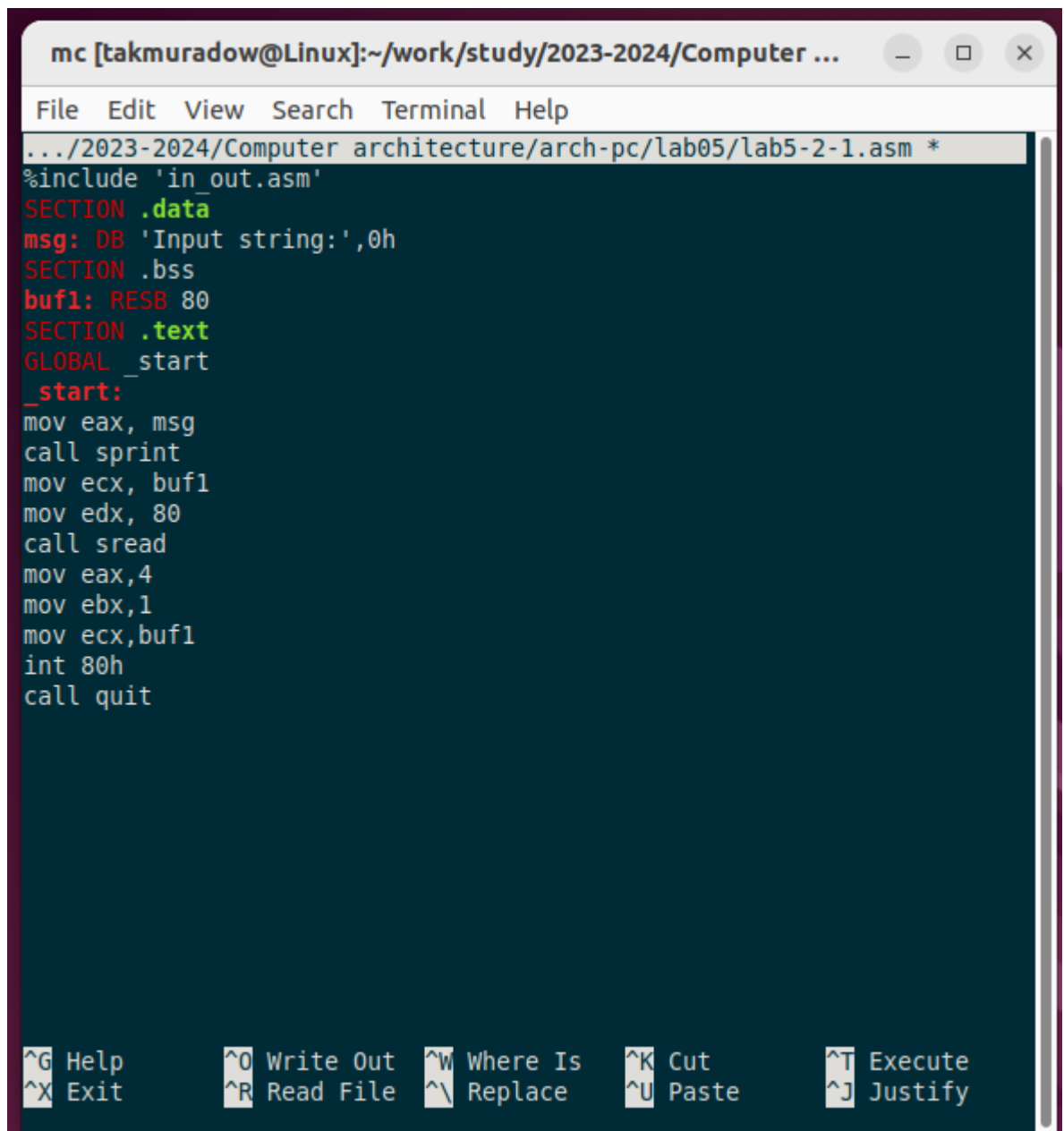
int 80h ; Вызов ядра
mov eax, 3 ; Системный вызов для чтения (sys_read)
mov ebx, 0 ; Дескриптор файла 0 - стандартный ввод
mov ecx, buf1 ; Адрес буфера под вводимую строку
mov edx, 80 ; Длина вводимой строки
int 80h ; Вызов ядра
mov eax, 4 ; Системный вызов для записи (sys_write)
mov ebx, 1 ; Описатель файла '1' - стандартный вывод
mov ecx, buf1 ; Адрес строки buf1 в ecx
mov edx, buf1 ; Размер строки buf1
int 80h ; Вызов ядра
mov eax, 1 ; Системный вызов для выхода (sys_exit)
mov ebx, 0 ; Выход с кодом возврата 0 (без ошибок)
int 80h ; Вызов ядра

```

3. Создаю копию файла lab5-2.asm с именем lab5-2-1.asm с помощью функциональной клавиши F5 (рис. 22).



С помощью функциональной клавиши F4 открываю созданный файл для редактирования. Изменяю программу так, чтобы кроме вывода приглашения и запроса ввода, она выводила вводимую пользователем строку (рис. 23).

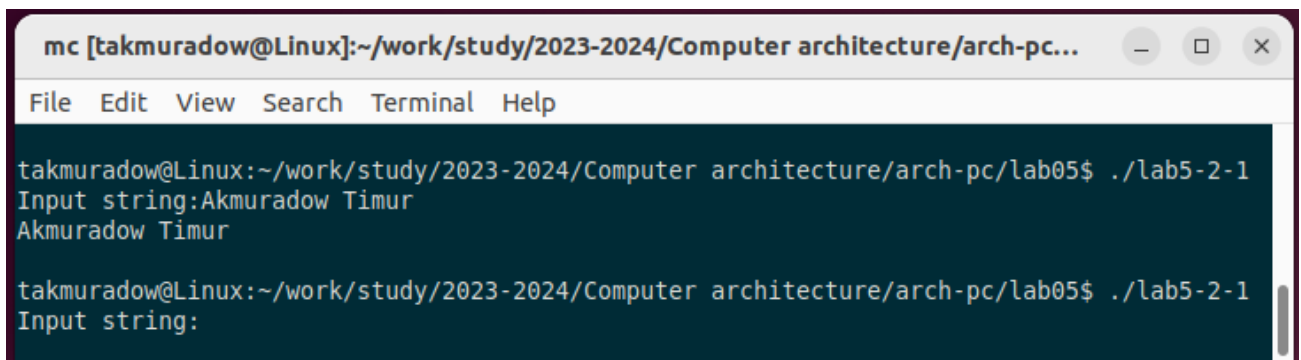


```
mc [takmuradow@Linux]:~/work/study/2023-2024/Computer ...
File Edit View Search Terminal Help
.../2023-2024/Computer architecture/arch-pc/lab05/lab5-2-1.asm *
%include 'in_out.asm'
SECTION .data
msg: DB 'Input string:',0h
SECTION .bss
buf1: RESB 80
SECTION .text
GLOBAL _start
_start:
mov eax, msg
call sprint
mov ecx, buf1
mov edx, 80
call sread
mov eax, 4
mov ebx, 1
mov ecx, buf1
int 80h
call quit
```

^G Help ^O Write Out ^W Where Is ^K Cut ^T Execute
^X Exit ^R Read File ^\ Replace ^U Paste ^J Justify

4.

Создаю объектный файл lab5-2-1.o, отдаю его на обработку компоновщику, получаю исполняемый файл lab5-2-1, запускаю полученный исполняемый файл. Программа запрашивает ввод без переноса на новую строку, ввожу свои ФИО, далее программа выводит введенные мною данные (рис. 24).



```
mc [takmuradow@Linux]:~/work/study/2023-2024/Computer architecture/arch-pc...
File Edit View Search Terminal Help

takmuradow@Linux:~/work/study/2023-2024/Computer architecture/arch-pc/lab05$ ./lab5-2-1
Input string:Akmuradow Timur
Akmuradow Timur

takmuradow@Linux:~/work/study/2023-2024/Computer architecture/arch-pc/lab05$ ./lab5-2-1
Input string:
```

Код программы из пункта 3:

```
%include 'in_out.asm'
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: ; 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 `EAX`
mov edx, 80 ; запись длины вводимого сообщения в `EBX`
call sread ; вызов подпрограммы ввода сообщения
mov eax,4 ; Системный вызов для записи (sys_write)
mov ebx,1 ; Описатель файла '1' - стандартный вывод
mov ecx,buf1 ; Адрес строки buf1 в ecx
int 80h ; Вызов ядра
call quit ; вызов подпрограммы завершения
```

5. С помощью команд `git add .`, `git commit -m 'Add files'`, `git push` добавляю файлы лабораторной в репозиторий GitHub


```
takmuradow@Linux: ~/work/study/2023-2024/Computer architecture/arch-pc
File Edit View Search Terminal Help
takmuradow@Linux:~/work/study/2023-2024/Computer architecture/arch-pc$ git add .
takmuradow@Linux:~/work/study/2023-2024/Computer architecture/arch-pc$ git commit -m 'Add files'
[master a768269] Add files
14 files changed, 462 insertions(+), 208 deletions(-)
create mode 100644 lab05/in_out.asm
create mode 100755 lab05/lab5-1
create mode 100755 lab05/lab5-1-1
create mode 100644 lab05/lab5-1-1.asm
create mode 100644 lab05/lab5-1-1.o
create mode 100644 lab05/lab5-1.asm
create mode 100644 lab05/lab5-1.o
create mode 100755 lab05/lab5-2
create mode 100755 lab05/lab5-2-1
create mode 100644 lab05/lab5-2-1.asm
create mode 100644 lab05/lab5-2-1.o
create mode 100644 lab05/lab5-2.asm
create mode 100644 lab05/lab5-2.o
takmuradow@Linux:~/work/study/2023-2024/Computer architecture/arch-pc$ git push
Enumerating objects: 24, done.
Counting objects: 100% (24/24), done.
Delta compression using up to 6 threads
Compressing objects: 100% (19/19), done.
Writing objects: 100% (19/19), 5.67 KiB | 5.67 MiB/s, done.
Total 19 (delta 11), reused 0 (delta 0), pack-reused 0
remote: Resolving deltas: 100% (11/11), completed with 5 local objects.
To github.com:takmuradow/study_2023-2024_arch-pc.git
   e986ddd..a768269  master -> master
takmuradow@Linux:~/work/study/2023-2024/Computer architecture/arch-pc$
```

Выводы

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

Список литературы

1. Лабораторная работа №5