

# Отчет по лабораторной работе №5

## Дисциплина: архитектура компьютера

Гусейнов Тагир

### 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 .....	5
Выполнение заданий для самостоятельной работы.....	8
Выводы.....	11
Список литературы.....	11

### 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.

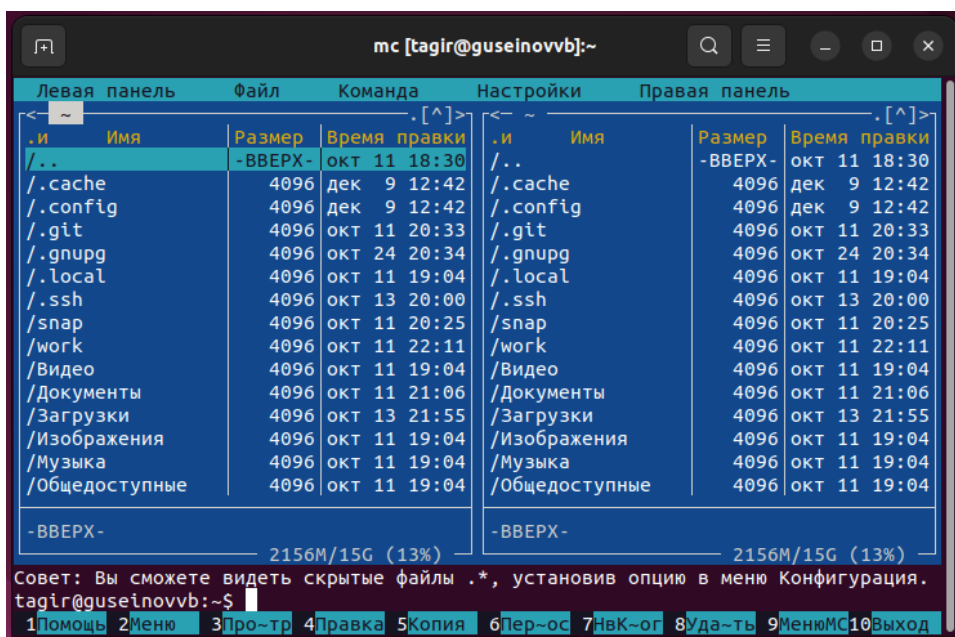
```
int n
```

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).

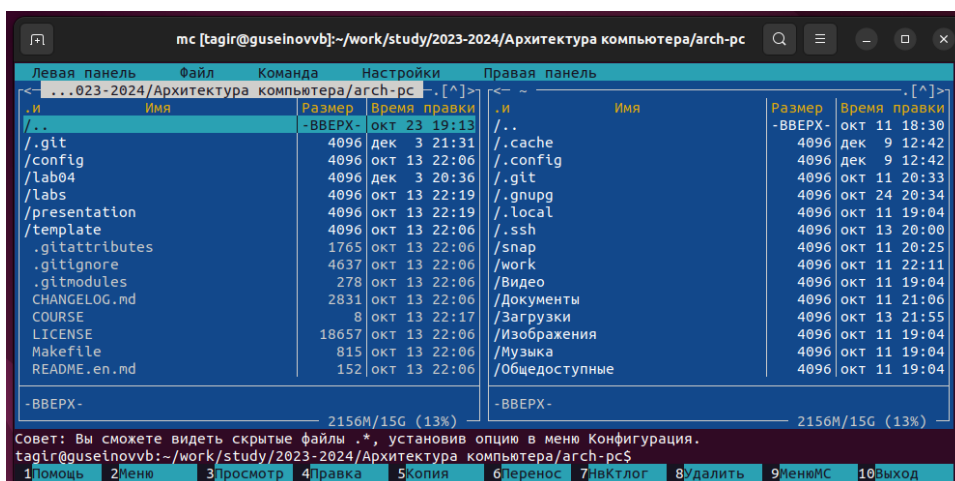
## Выполнение лабораторной работы

### Основы работы с mc

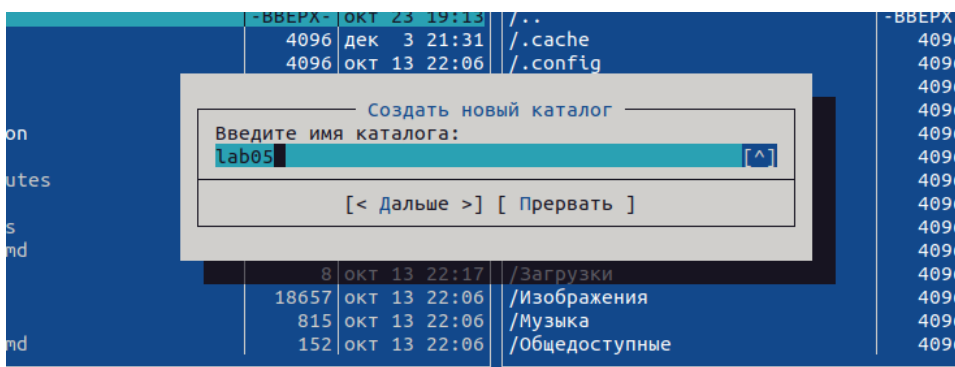
Открываю Midnight Commander, введя в терминал mc (рис. 01).



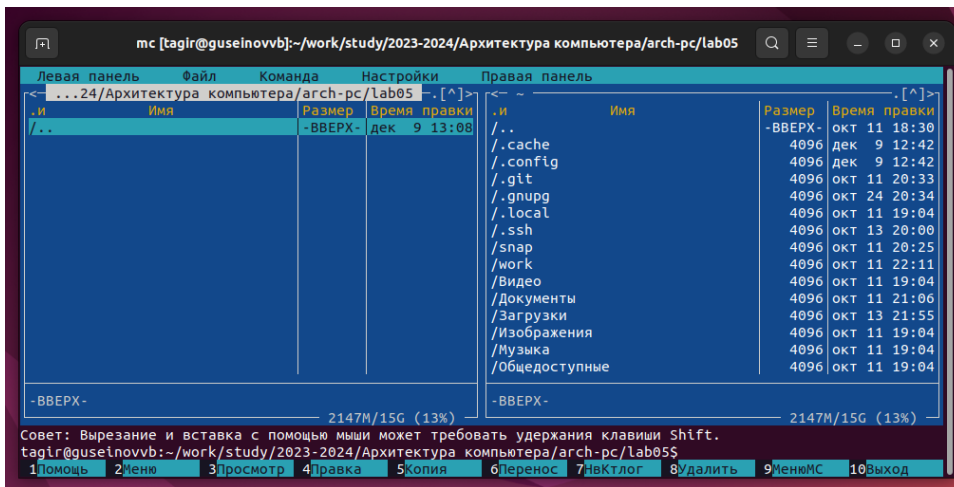
Перехожу в каталог ~/work/study/2022-2023/Архитектура Компьютера/arch-rc, используя файловый менеджер mc (рис. 02)



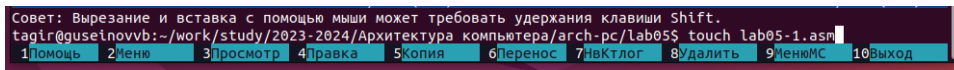
С помощью функциональной клавиши F7 создаю каталог lab05 (рис. 03).



Перехожу в созданный каталог (рис. 04).

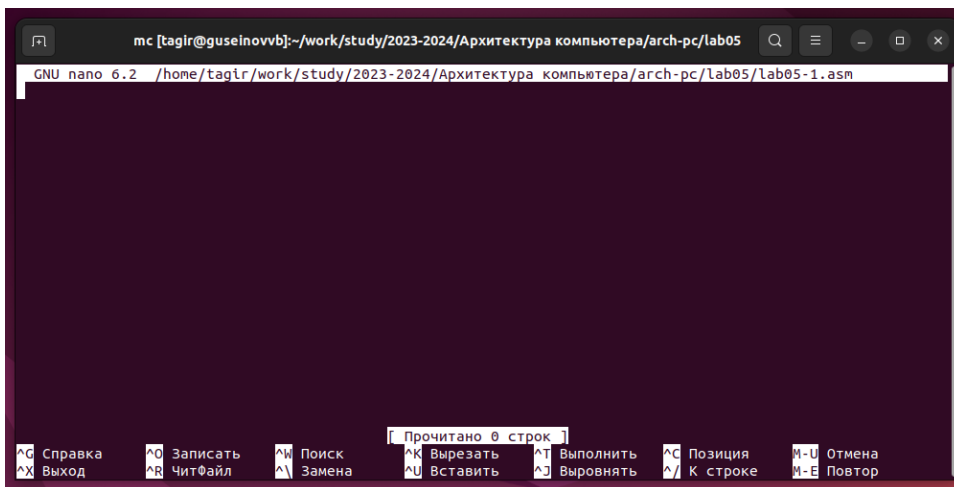


В строке ввода прописываю команду `touch lab05-1.asm`, чтобы создать файл, в котором буду работать (рис. 05).

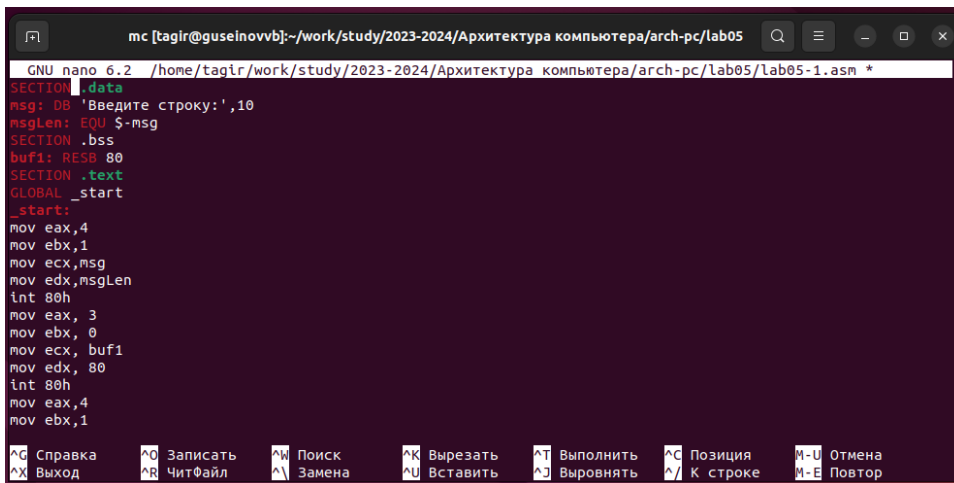


## Структура программы на языке ассемблера NASM

С помощью функциональной клавиши F4 открываю созданный файл для редактирования в редакторе nano (рис. 06).

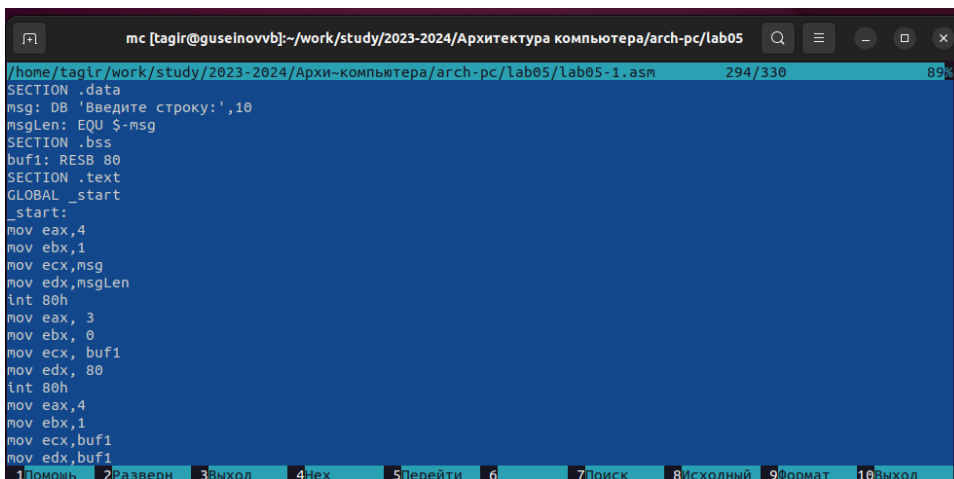


Ввожу в файл код программы для запроса строки у пользователя (рис. 07). Далее выхожу из файла (Ctrl+X), сохраняя изменения (Y, Enter).



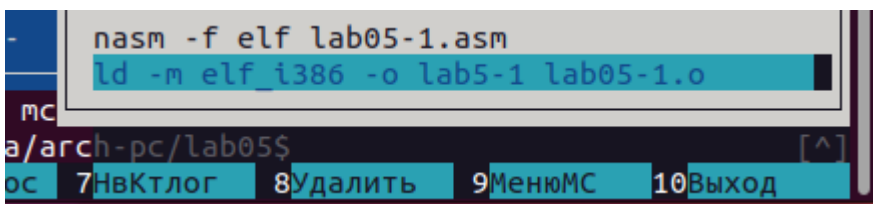
```
GNU nano 6.2 /home/taglr/work/study/2023-2024/Архитектура компьютера/arch-pc/lab05/lab05-1.asm *
SECTION .data
msg: DB 'Введите строку:',10
msgLen: EQU $-msg
SECTION .bss
buf1: RESB 80
SECTION .text
GLOBAL _start
_start:
mov eax,4
mov ebx,1
mov ecx,msg
mov edx,msgLen
int 80h
mov eax, 3
mov ebx, 0
mov ecx, buf1
mov edx, 80
int 80h
mov eax,4
mov ebx,1
```

С помощью функциональной клавиши F3 открываю файл для просмотра, чтобы проверить, содержит ли файл текст программы (рис. 08).



```
/home/taglr/work/study/2023-2024/Архитектура компьютера/arch-pc/lab05/lab05-1.asm 294/330 89%
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
```

Транслирую текст программы файла в объектный файл командой `nasm -f elf lab05-1.asm`. Создался объектный файл `lab5-1.o`. Выполняю компоновку объектного файла с помощью команды (рис. 9). Создался исполняемый файл `lab5-1`.

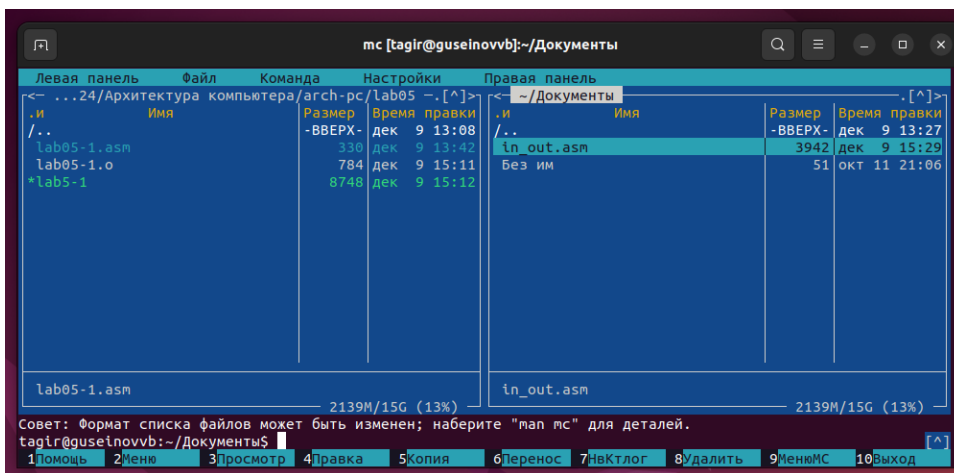


```
- nasm -f elf lab05-1.asm
ld -m elf_i386 -o lab5-1 lab05-1.o
MC
a/arch-pc/lab05$
ос 7ВклТлог 8Удалить 9МенюМС 10Выход
```

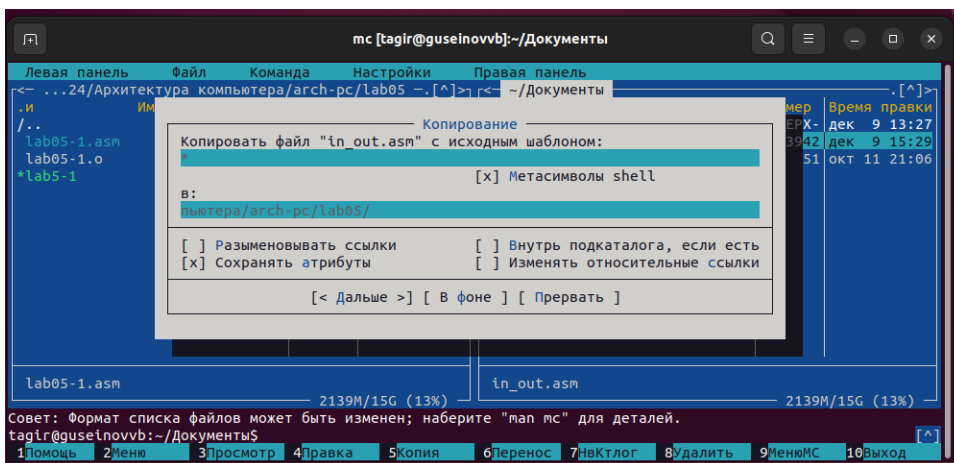
Запускаю исполняемый файл. Программа выводит строку “Введите строку:” и ждет ввода с клавиатуры, я ввожу свои ФИО, на этом программа заканчивает свою работу

## Подключение внешнего файла

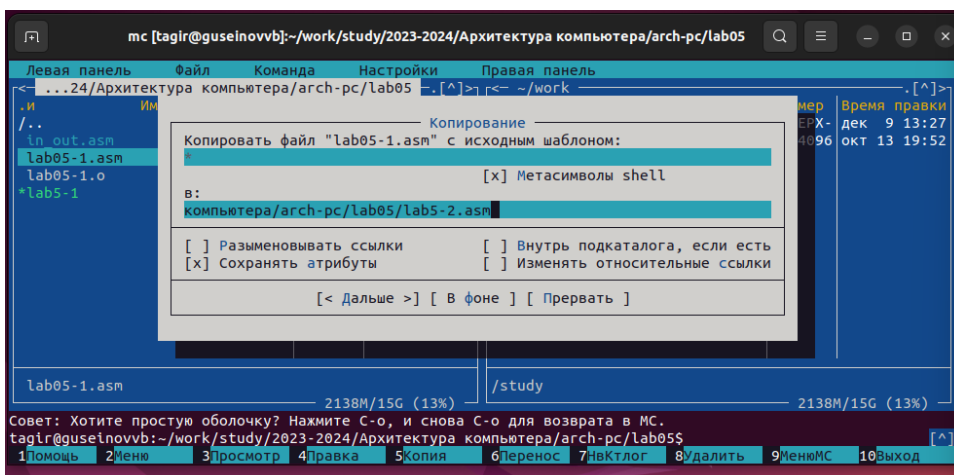
Скачиваю файл `in_out.asm` со страницы курса в ТУИС. (рис. 11).



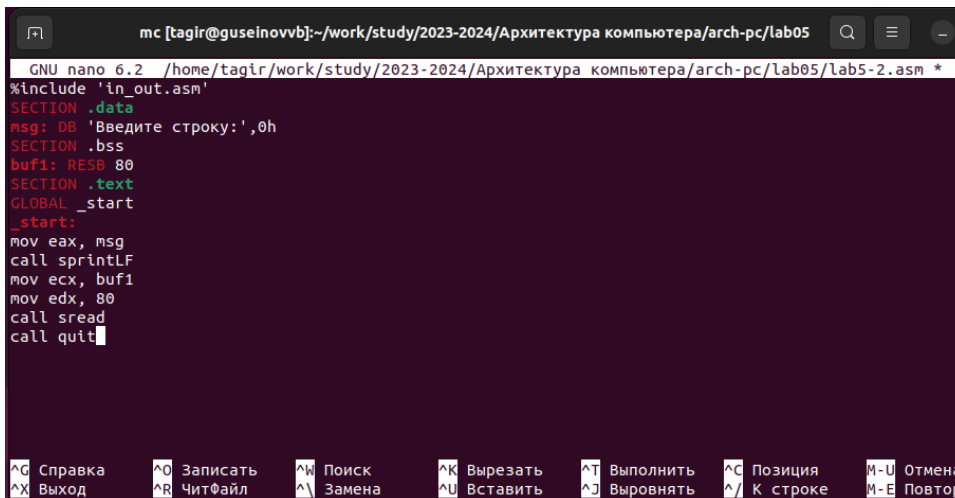
С помощью функциональной клавиши F5 копирую файл in\_out.asm из каталога Загрузки в созданный каталог lab05 (рис. 12).



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. 13).



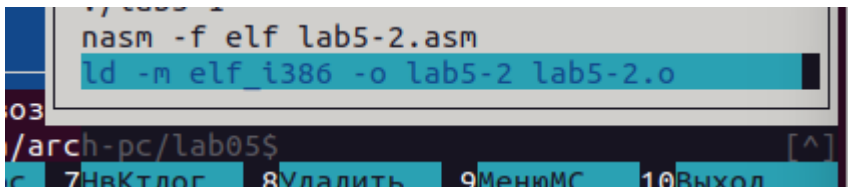
Изменяю содержимое файла lab5-2.asm во встроенном редакторе nano (рис. 14), чтобы в программе использовались подпрограммы из внешнего файла in\_out.asm.



```
mc [tagir@guseinovvb]:~/work/study/2023-2024/Архитектура компьютера/arch-pc/lab05
GNU nano 6.2 /home/tagir/work/study/2023-2024/Архитектура компьютера/arch-pc/lab05/lab5-2.asm *
#include 'in_out.asm'
SECTION .data
msg: DB 'Введите строку:',0h
SECTION .bss
buf1: RESB 80
SECTION .text
GLOBAL _start
_start:
mov eax, msg
call sprintLF
mov ecx, buf1
mov edx, 80
call sread
call quit
```

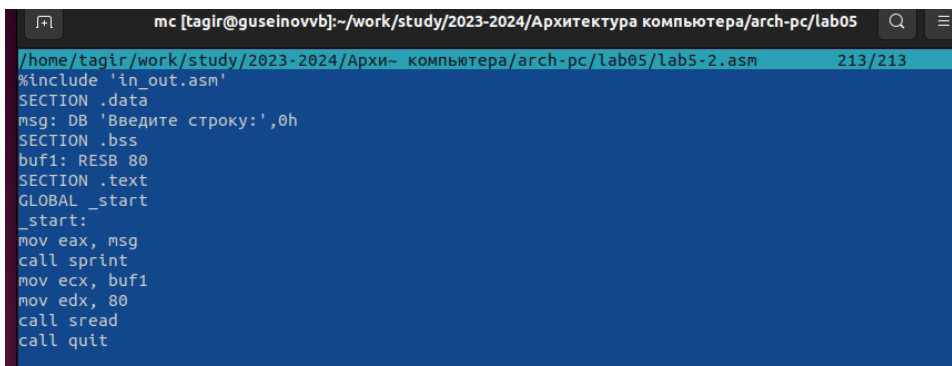
Справка Записать Поиск Вырезать Выполнить Позиция Отмена  
Выход ЧитФайл Замена Вставить Выводить К строке М-Е Повтор

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`. (рис. 15).



```
lab5-2.o
nasm -f elf lab5-2.asm
ld -m elf_i386 -o lab5-2 lab5-2.o
/arch-pc/lab05$
```

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. 16).



```
mc [tagir@guseinovvb]:~/work/study/2023-2024/Архитектура компьютера/arch-pc/lab05
/home/tagir/work/study/2023-2024/Архи~ компьютера/arch-pc/lab05/lab5-2.asm 213/213
#include 'in_out.asm'
SECTION .data
msg: DB 'Введите строку:',0h
SECTION .bss
buf1: RESB 80
SECTION .text
GLOBAL _start
_start:
mov eax, msg
call sprint
mov ecx, buf1
mov edx, 80
call sread
call quit
```

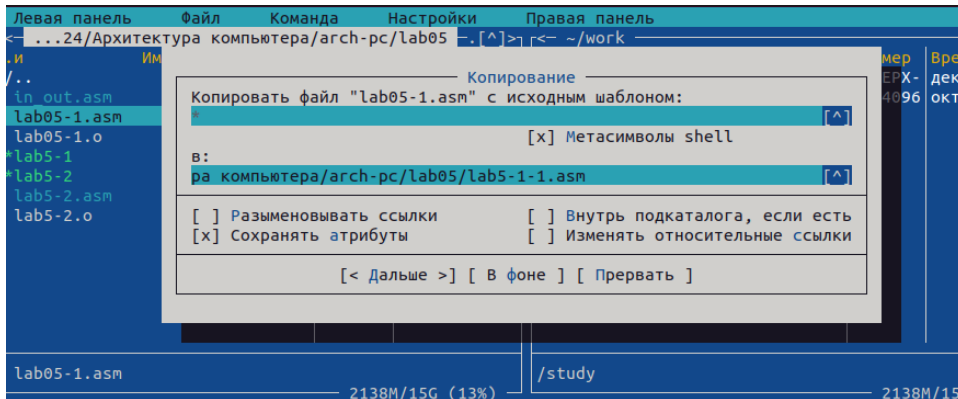
Снова транслирую файл, выполняю компоновку созданного объектного файла, запускаю новый исполняемый файл

на след строке вывел.

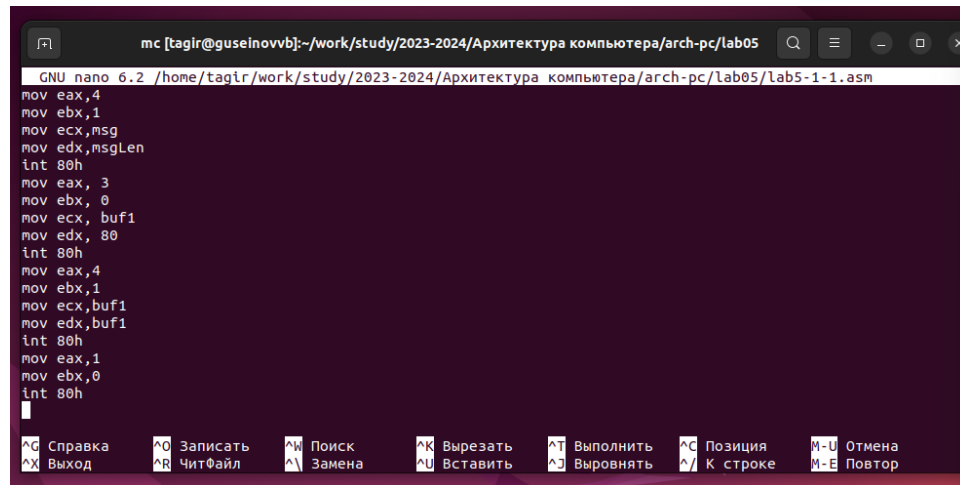
Разница между первым исполняемым файлом `lab5-2` и вторым `lab5-2-2` в том, что запуск первого запрашивает ввод с новой строки, а программа, которая выполняется при запуске второго, запрашивает ввод без переноса на новую строку, потому что в этом заключается различие между подпрограммами `sprintLF` и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

## Выполнение заданий для самостоятельной работы

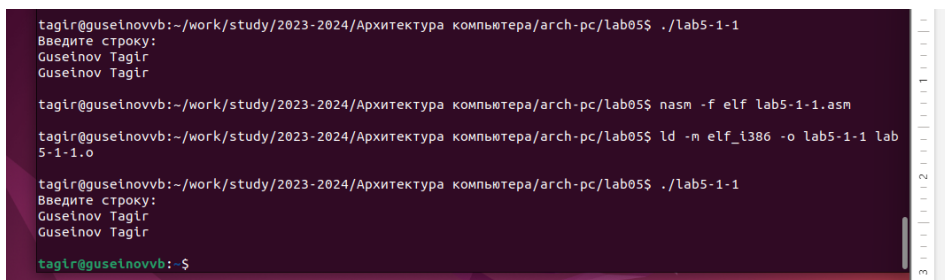
1. Создаю копию файла lab5-1.asm с именем lab5-1-1.asm с помощью функциональной клавиши F5 (рис. 18).



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. 19).



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. 20).



Код программы из пункта 1:

```
SECTION .data ; Секция инициализированных данных
msg: DB 'Введите строку:',10
msgLen: EQU $-msg ; Длина переменной 'msg'
```

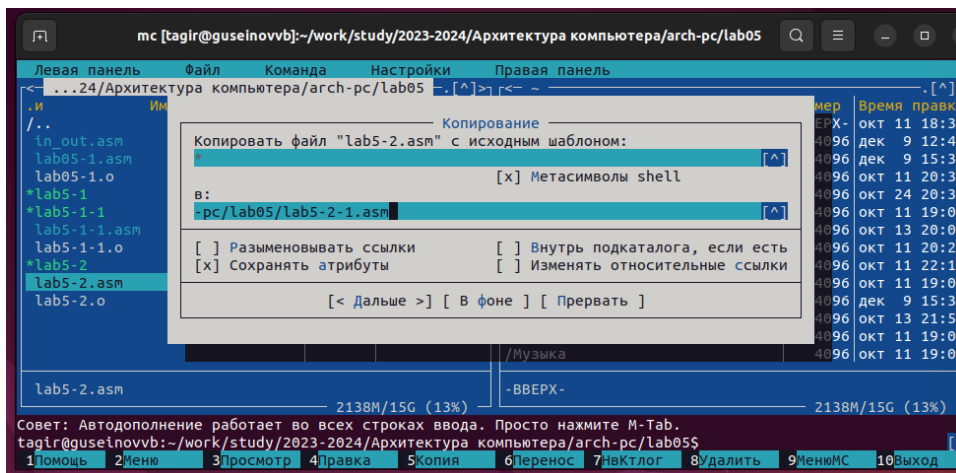


```

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 ; Описание 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. 21).



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. 22).

```

tagir@guseinovvb: ~
GNU nano 6.2 /home/tagir/work/study/2023-2024/Архитектура компьютера/arch-pc/lab05/lab5-2-1.asm *
#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

```

^G Справка    ^O Записать    ^W Поиск    ^K Вырезать    ^T Выполнить    ^C Позиция    M-U Отмена  
 ^X Выход    ^R ЧитФайл    ^\ Замена    ^U Вставить    ^J Выводить    ^/ К строке    M-E Повтор

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. 23).

```

tagir@guseinovvb:~/work/study/2023-2024/Архитектура компьютера/arch-pc/lab05$ ./lab5-2-1
Введите строку: Guseinov T
Guseinov T
tagir@guseinovvb:~$

```

Код программы из пункта 3:п

```

#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 (рис. 24)

```
tagir@guseinovvb:~/work/study/2023-2024/Архитектура компьютера/arch-pc$ git config pull.rebase true
tagir@guseinovvb:~/work/study/2023-2024/Архитектура компьютера/arch-pc$ git pull
Успешно перемещён и обновлён refs/heads/master.
tagir@guseinovvb:~/work/study/2023-2024/Архитектура компьютера/arch-pc$ git push
Перечисление объектов: 28, готово.
Подсчет объектов: 100% (28/28), готово.
При сжатии изменений используется до 2 потоков
Сжатие объектов: 100% (27/27), готово.
Запись объектов: 100% (27/27), 8.21 КиБ | 2.74 МИБ/с, готово.
Всего 27 (изменений 15), повторно использовано 0 (изменений 0), повторно использовано пакетов 0
remote: Resolving deltas: 100% (15/15), completed with 1 local object.
To github.com:bstwshs/study_2023-2024_arh-pc.git
   bbe813e..b0122f6  master -> master
tagir@guseinovvb:~/work/study/2023-2024/Архитектура компьютера/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