Отчет по лабораторной работе №5

Дисциплина: архитектура компьютера

Бизев Никита Владимирович

Содержание

# 1 Цель работы

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

# 2 Задание

1. Основы работы с mc
2. Структура программы на языке ассемблера NASM
3. Подключение внешнего файла

# 3 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 инициированных 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 размеров 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 (учетве- рённое 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 intпредназначена для вызова прерывания с указанным номером.

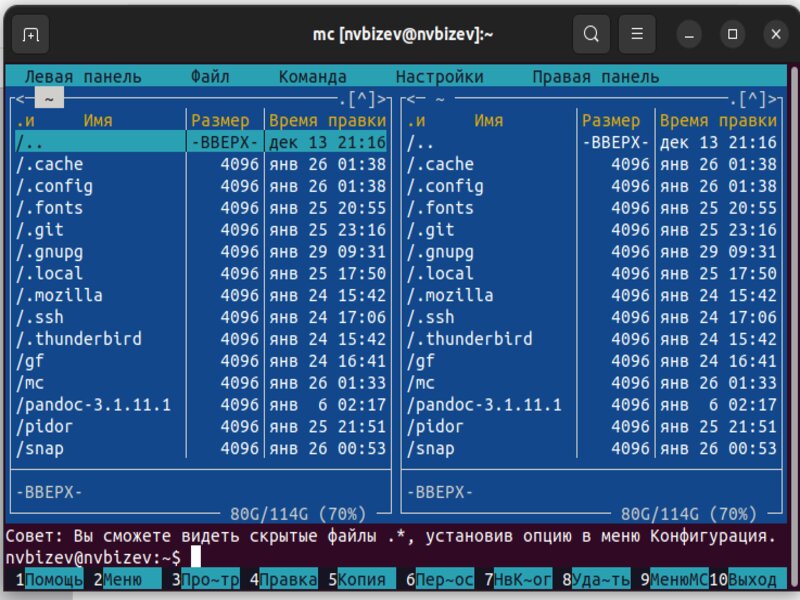
int n

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).

# 4 Выполнение лабораторной работы

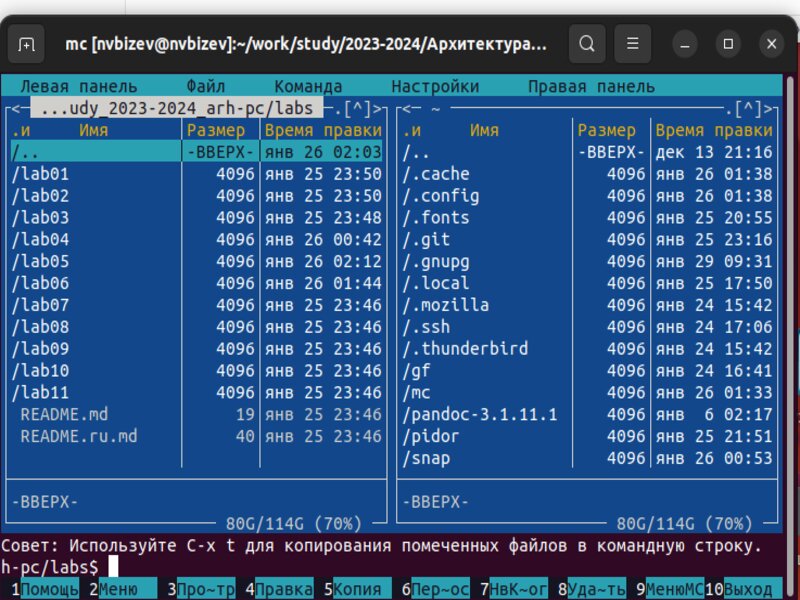
## 4.1 Основы работы с mc

Открываю Midnight Commander, введя в терминал mc (рис. [??]).



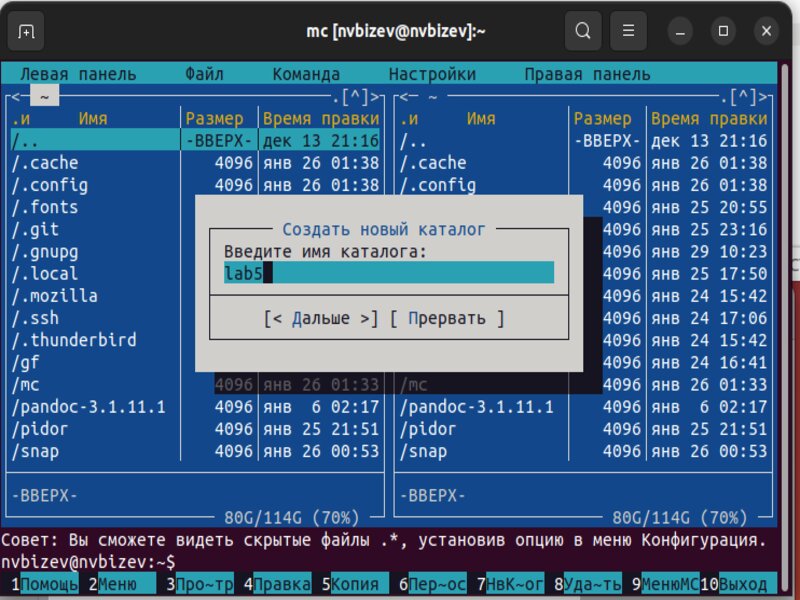
Открытый mc

Перехожу в каталог ~/work/study/2023-2024/Архитектура Компьютера/arch-pc, используя файловый менеджер mc (рис. [??])



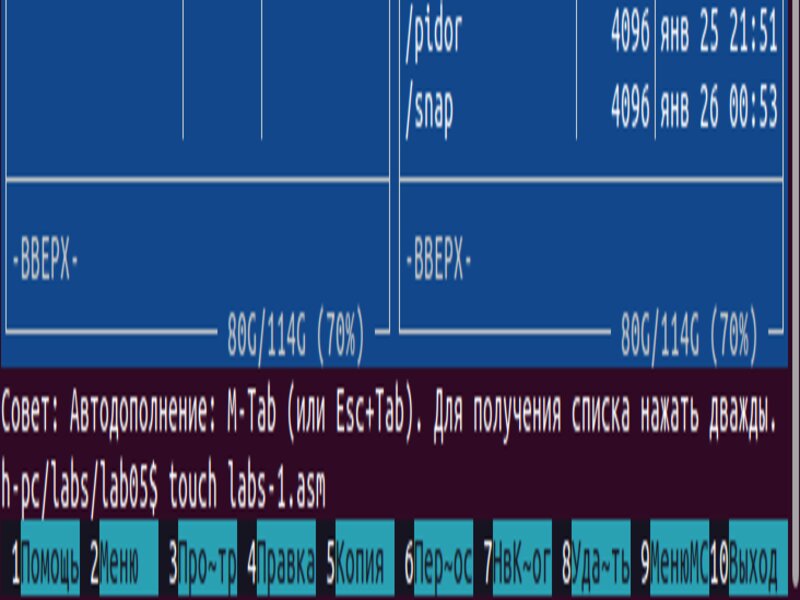
Перемещение между директориями

С помощью функциональной клавиши F7 создаю каталог lab5 (рис. [??]).



Создание каталога

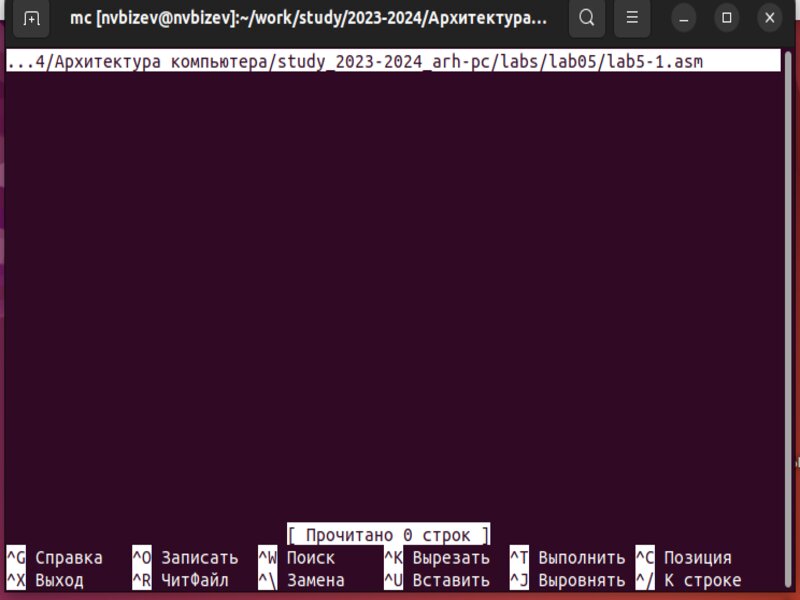
В строке ввода прописываю команду touch lab5-1.asm, чтобы создать файл, в котором буду работать (рис. [??]).



Создание файла

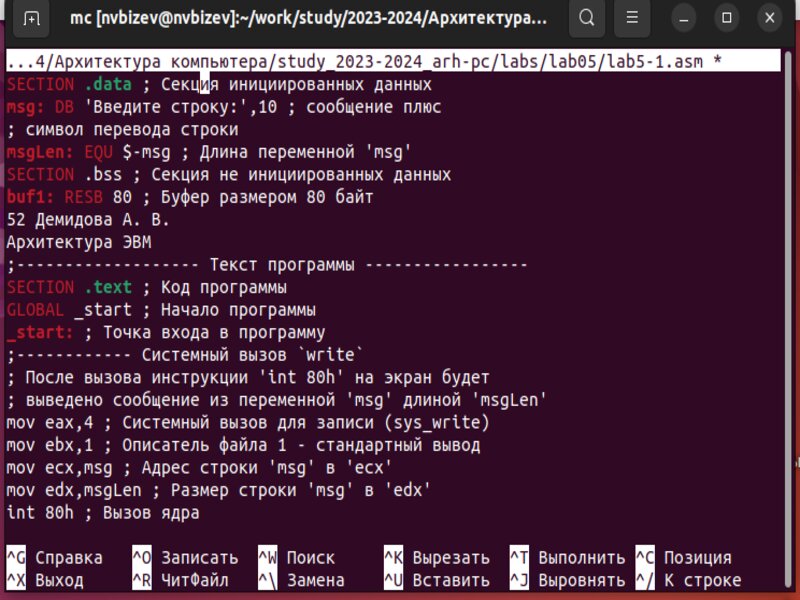
## 4.2 Структура программы на языке ассемблера NASM

С помощью функциональной клавиши F4 открываю созданный файл для редактирования в текстовом редакторе (рис. [??]).



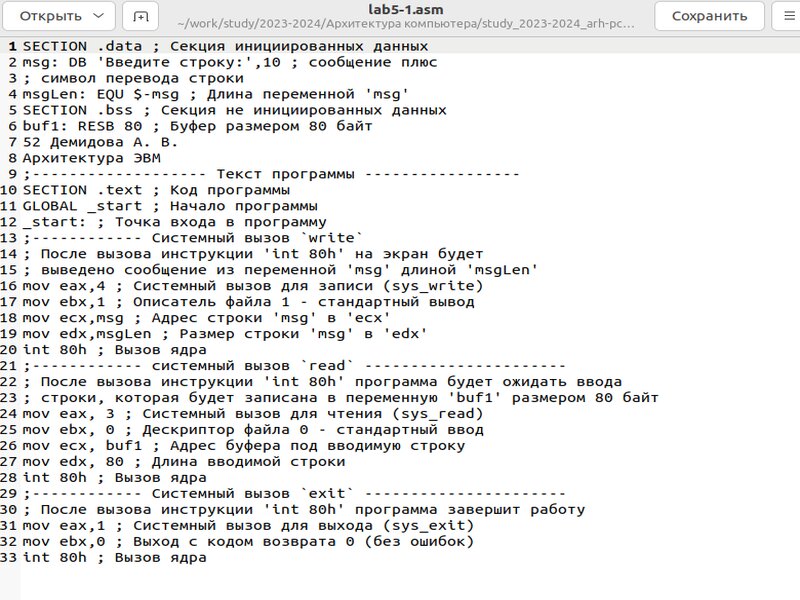
Открытие файла для редактирования

Ввожу в файл код программы для запроса строки у пользователя (рис. [??]).



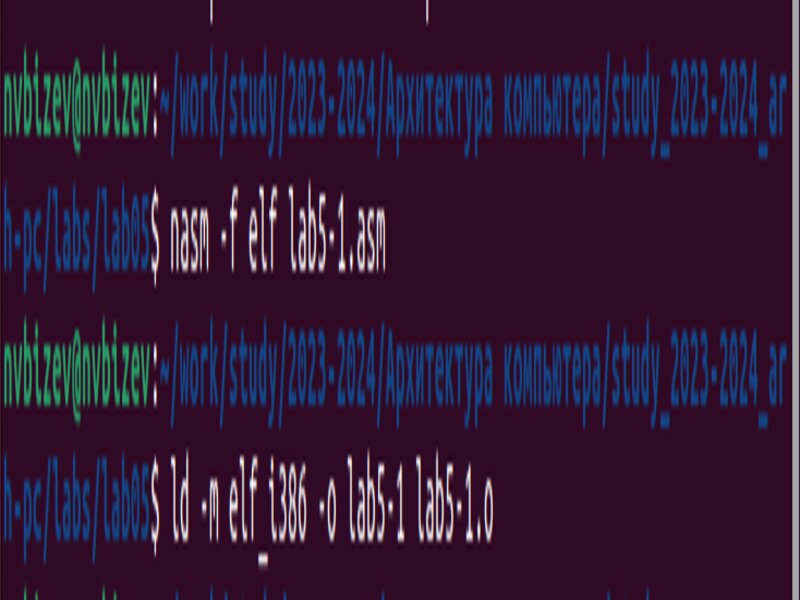
Редактирование файла

Открываю файл в текстовом редакторе для проверки. (рис. [??]).



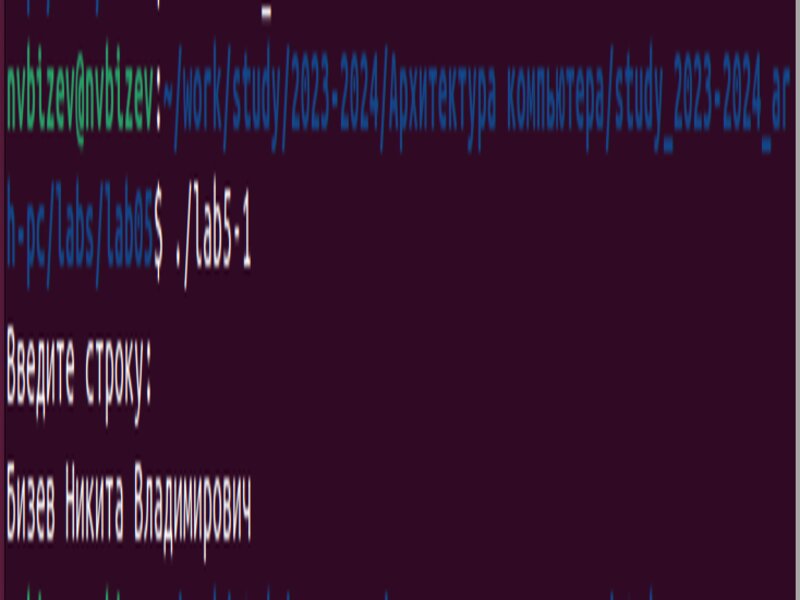
Открытие файла для просмотра

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. [??]). Создался исполняемый файл lab6-1.



Компиляция файла и передача на обработку компоновщику

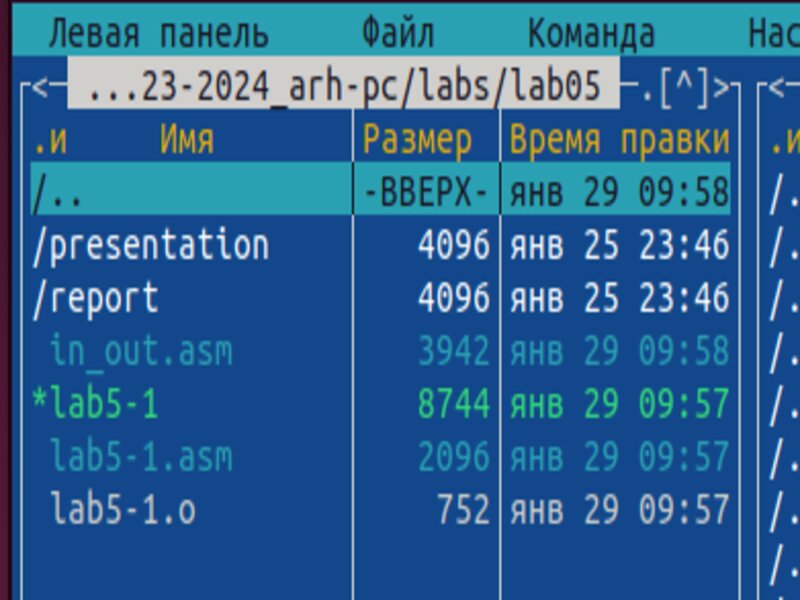
Запускаю исполняемый файл. Программа выводит строку “Введите строку:” и ждет ввода с клавиатуры, я ввожу свои ФИО, на этом программа заканчивает свою работу (рис. [??]).



Исполнение файла

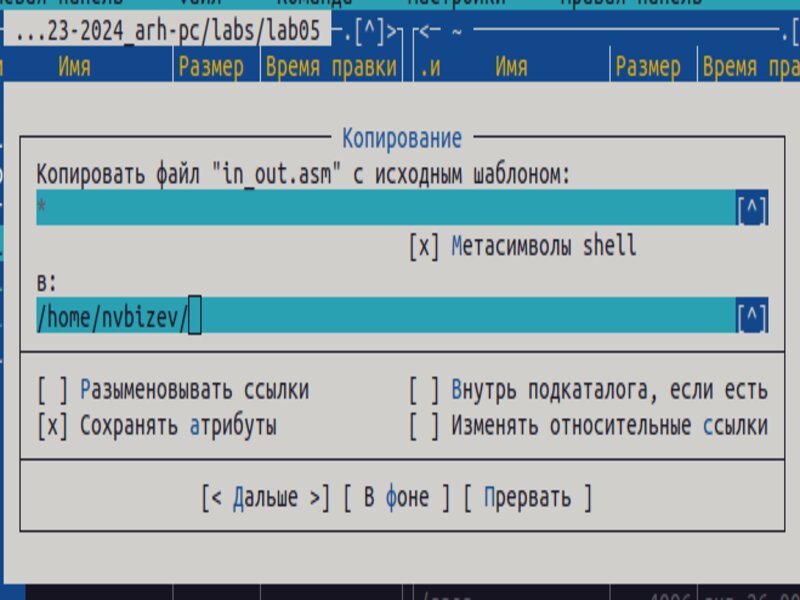
## 4.3 Подключение внешнего файла

Скачиваю файл in\_out.asm со страницы курса в ТУИС. (рис. [??]).



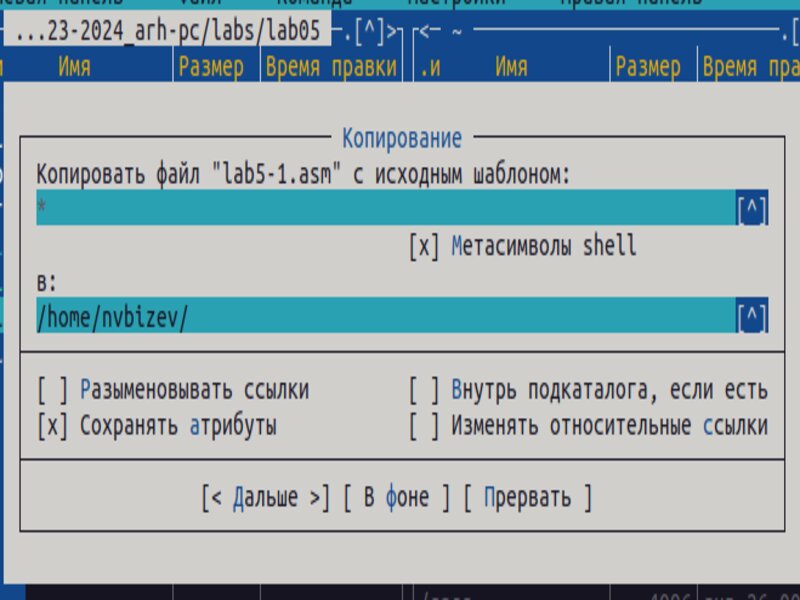
Скачанный файл

С помощью функциональной клавиши F5 копирую файл in\_out.asm в созданный каталог lab05 (рис. [??]).



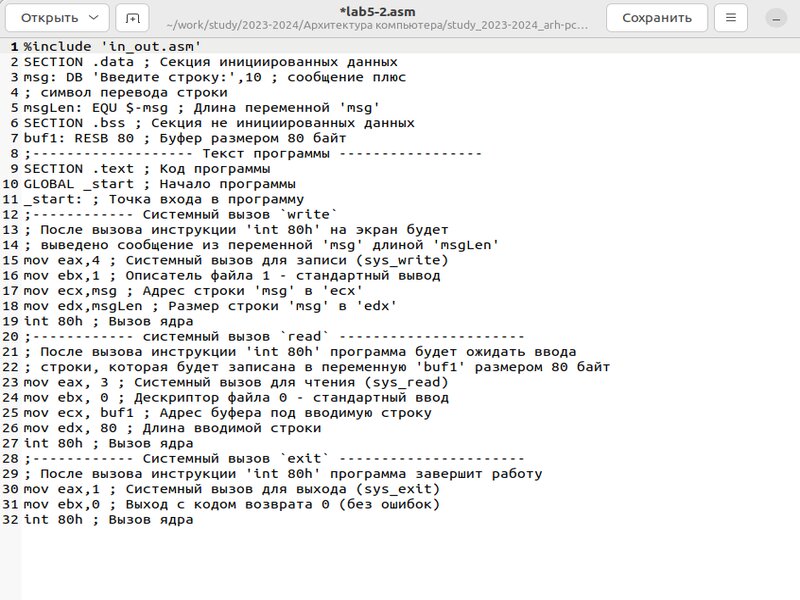
Копирование файла

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. [??]).



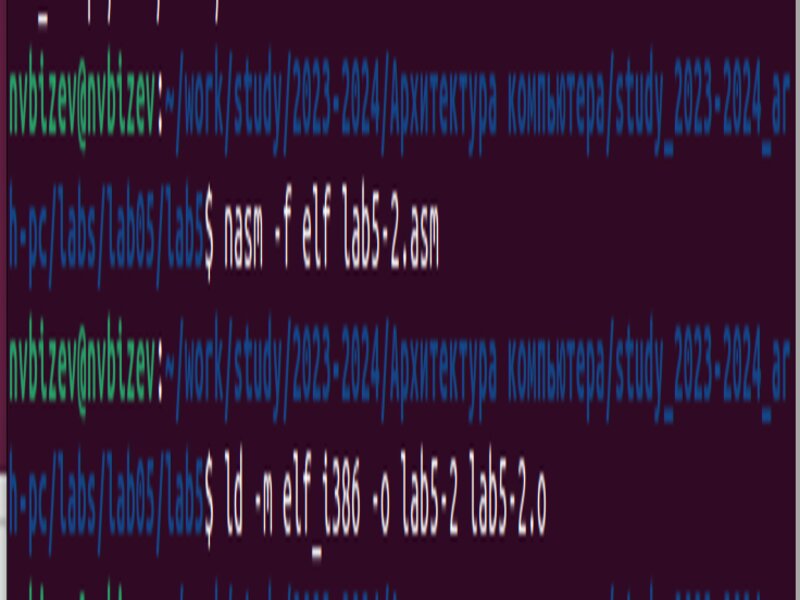
Копирование файла

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



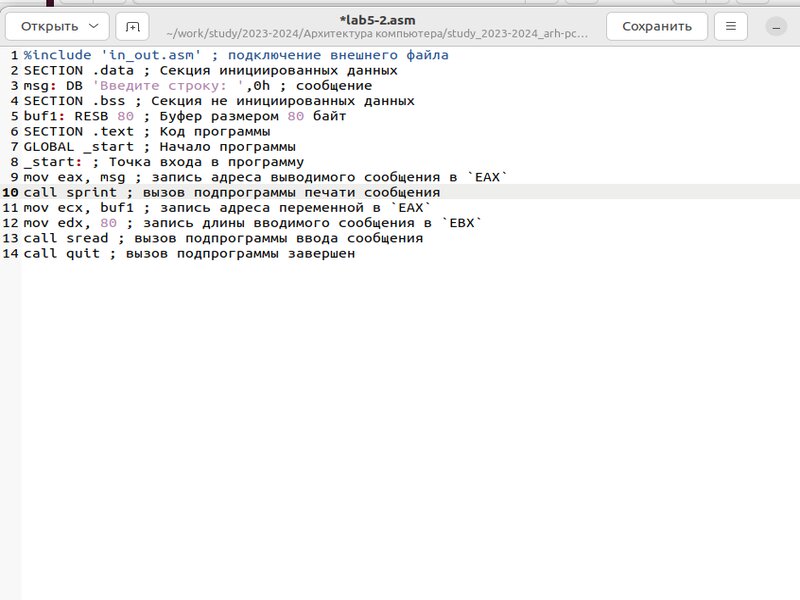
Редактирование файла

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. [??])



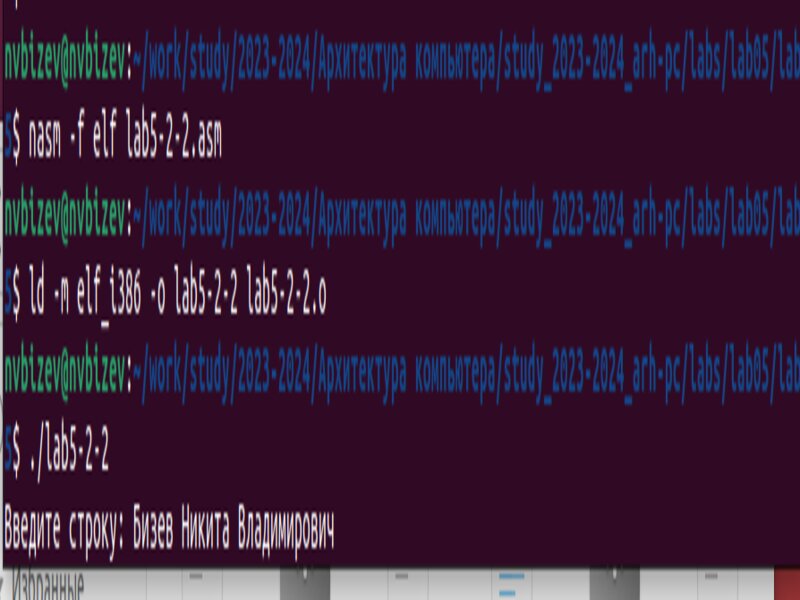
Исполнение файла

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



Отредактированный файл

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

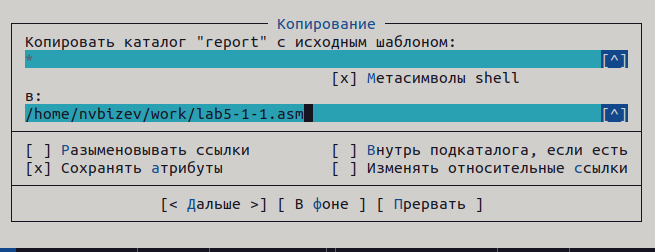


Исполнение файла

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 исполняется при запуске второго, запрашивает ввод без переноса на новую строку, потому что в этом заключается различие между подпрограммами sprintLF и sprint.

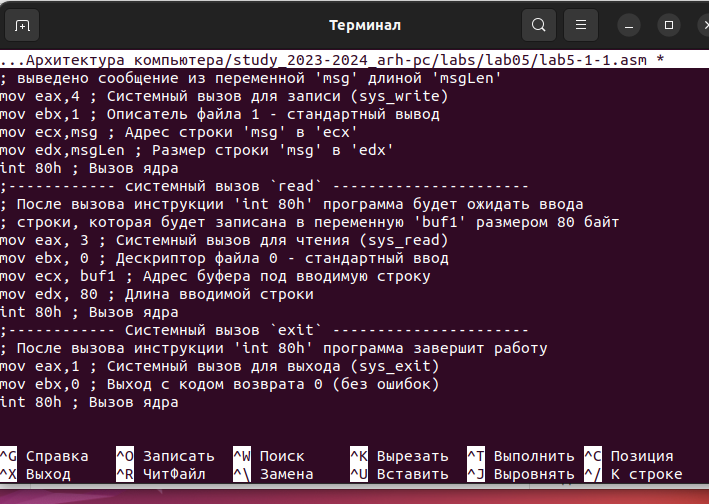
## 4.4 **Задания для самостоятельной работы**

1. Создаю копию файла lab5-1.asm. (рис. ??).



Создание копии файла

Вношу изменения в программу так, чтобы она выводила введённую строку на экран. (рис. ??).



Внесение изменений в программу

Сам код:

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

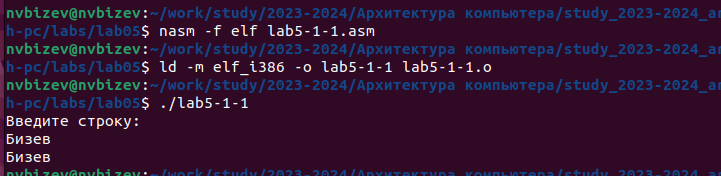
int 80h

mov eax,1

mov ebx,0

int 80h

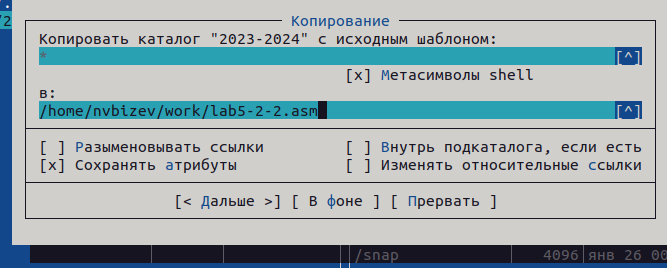
1. Получаю исполняемый файл и проверяю его работу. На приглашение ввести строку ввожу свою фамилию. (рис. ??).



Запуск исполняющего файла

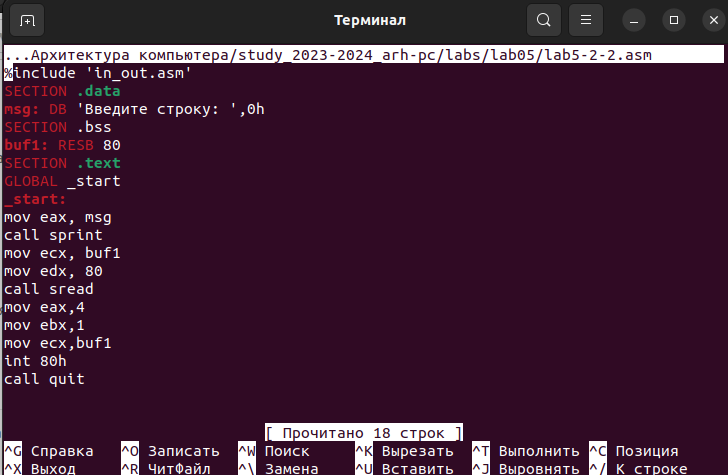
Программа работает.

1. Создаю копию файла lab5-2.asm. (рис. ??).



Создание копии файла

Вношу изменения в программу с использование подпрограмм из внешнего файла in\_out.asm так, чтобы она выводила введённую строку на экран. (рис. ??).



Внесение изменений в программу

Сам код:

%include ‘in\_out.asm’

SECTION .data

msg: DB ‘Введите строку:’,0h

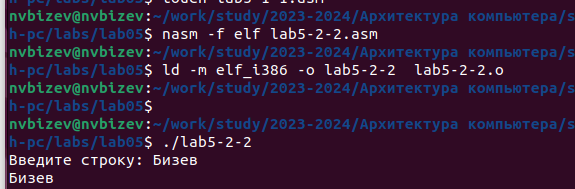
SECTION .bss

buf1: RESB 80

SECTION .text

GLOBAL \_start  
  
\_start:  
  
mov eax, msg  
  
call sprint  
  
mov ecx, buf1  
  
mov edx, 80  
  
call sread  
  
mov eax,4  
  
mov ebx,1  
  
mov ecx,buf1  
  
int 80h  
  
call quit

1. Создаю исполняемый файл и проверяю его работу. (рис. ??).



Запуск исполняющего файла

Программа работает.

# 5 Выводы

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

# 6 Список литературы

1. https://esystem.rudn.ru/pluginfile.php/2089085/mod\_resource/content/0/%D0%9B%D0%B0%D0%B1%D0%BE%D1%80%D0%B0%D1%82%D0%BE%D1%80%D0%BD%D0%B0%D1%8F%20%D1%80%D0%B0%D0%B1%D0%BE%D1%82%D0%B0%20%E2%84%965.%20%D0%9E%D1%81%D0%BD%D0%BE%D0%B2%D1%8B%20%D1%80%D0%B0%D0%B1%D0%BE%D1%82%D1%8B%20%D1%81%20Midnight%20Commander%20%28%29.%20%D0%A1%D1%82%D1%80%D1%83%D0%BA%D1%82%D1%83%D1%80%D0%B0%20%D0%BF%D1%80%D0%BE%D0%B3%D1%80%D0%B0%D0%BC%D0%BC%D1%8B%20%D0%BD%D0%B0%20%D1%8F%D0%B7%D1%8B%D0%BA%D0%B5%20%D0%B0%D1%81%D1%81%D0%B5%D0%BC%D0%B1%D0%BB%D0%B5%D1%80%D0%B0%20NASM.%20%D0%A1%D0%B8%D1%81%D1%82%D0%B5%D0%BC%D0%BD%D1%8B%D0%B5%20%D0%B2%D1%8B%D0%B7%D0%BE%D0%B2%D1%8B%20%D0%B2%20%D0%9E%D0%A1%20GNU%20Linux.pdf