РОССИЙСКИЙ УНИВЕРСИТЕТ ДРУЖБЫ НАРОДОВ

Факультет физико-математических и естественных наук

Кафедра прикладной информатики и теории вероятностей

ОТЧЕТ
ПО ЛАБОРАТОРНОЙ РАБОТЕ №5

***Дисциплина*: Архитектура компьютеров**

Студент: Болдырева Дельгир

Группа: НКАбд-01-25

Москва

2025 г.

Оглавление

1 Цель работы

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

2 Задание

1. Создайте копию файла lab5-1.asm. Внесите изменения в программу (без использования внешнего файла in_out.asm), так чтобы она работала по следующему алгоритму: • вывести приглашение типа “Введите строку:”; • ввести строку с клавиатуры; • вывести введённую строку на экран.

2. Получите исполняемый файл и проверьте его работу. На приглашение ввести строку введите свою фамилию.

3. Создайте копию файла lab5-2.asm. Исправьте текст программы с использование подпрограмм из внешнего файла in_out.asm, так чтобы она ра- бо- тала по следующему алгоритму:

• вывести приглашение типа “Введите строку:”;

• ввести строку с клавиатуры;

• вывести введённую строку на экран.

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

4 Выполнение лабораторной работы

4.1 Откроем Midnight Commander (рис. 4.1)

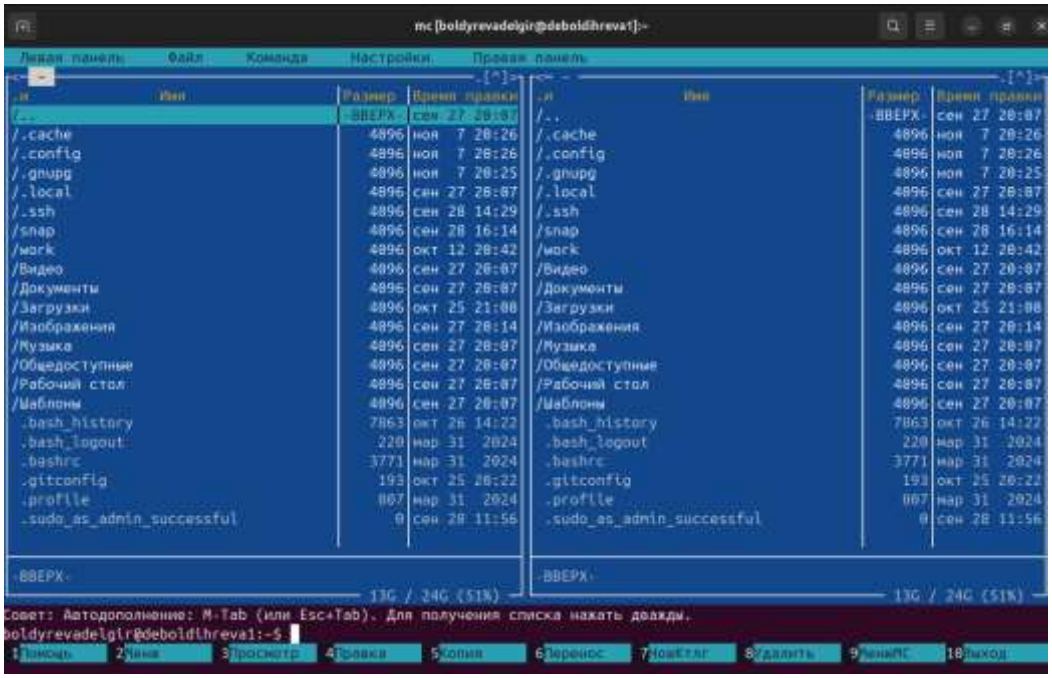


Рис 4.1

4.2 Воспользуемся клавишами движения вверх-вниз и клавишей Enter перейдем в каталог ~/work/arch-pc, который был создан при выполнении лабораторной №4 (рис 4.2-4.2.1)

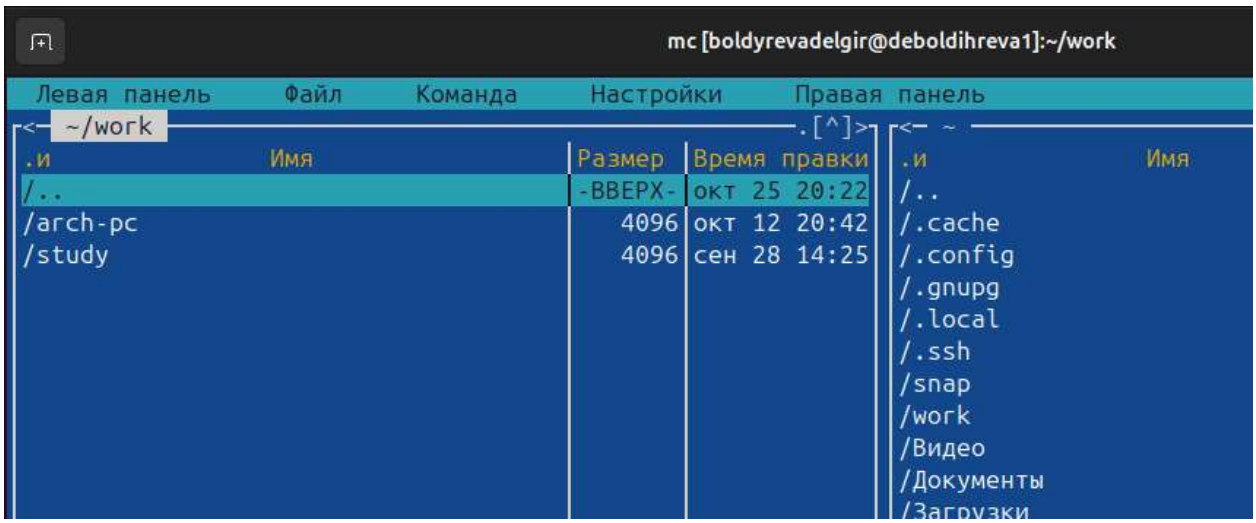


Рис 4.2

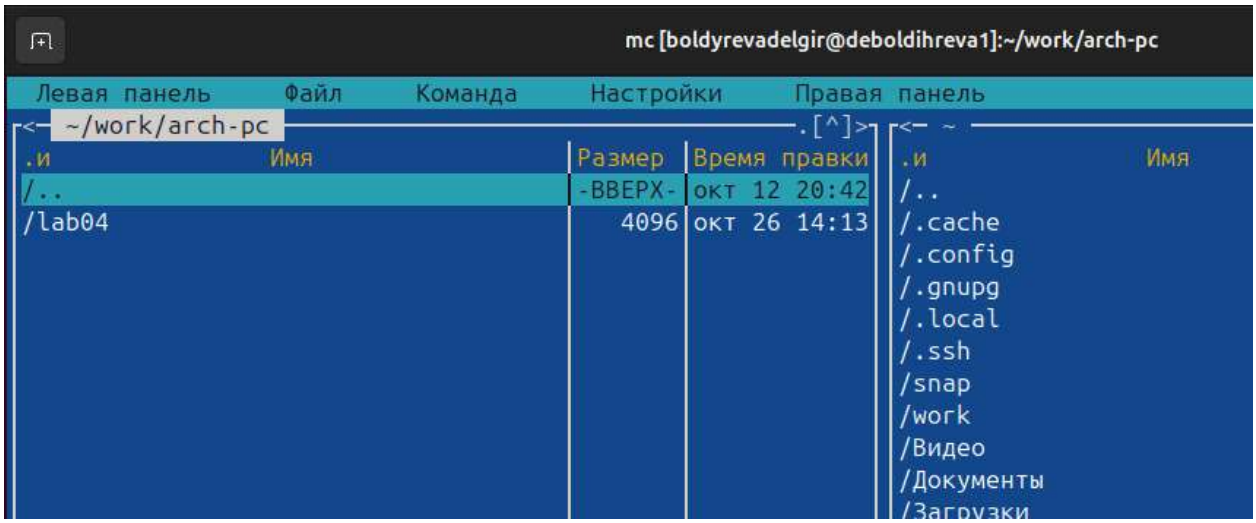


Рис 4.2.1

4.3 Далее создаем папку lab05 и перейдем в созданный каталог (рис 4.3-4.3.1)

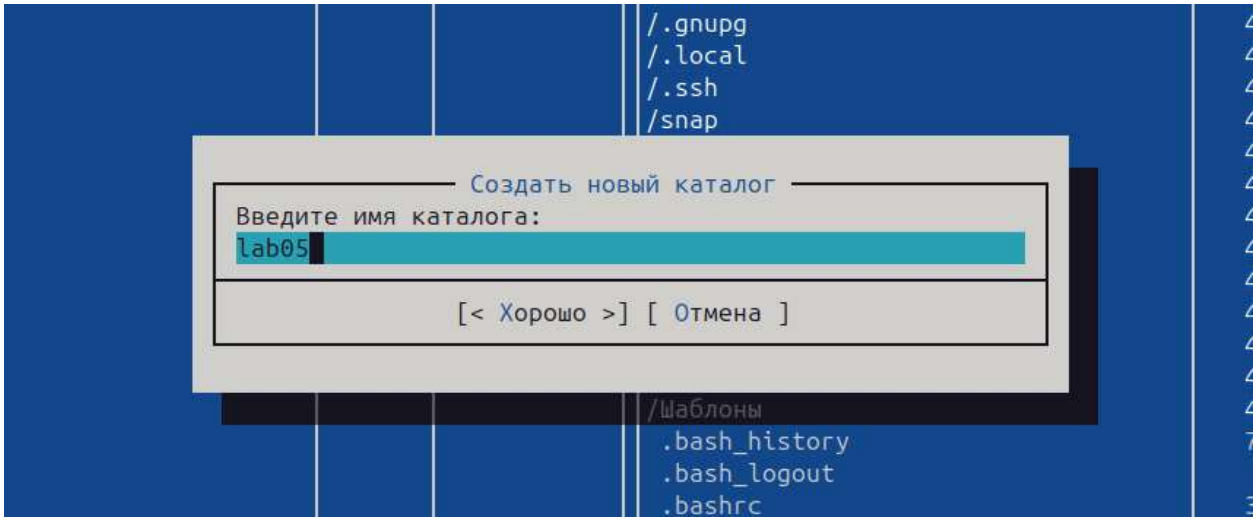
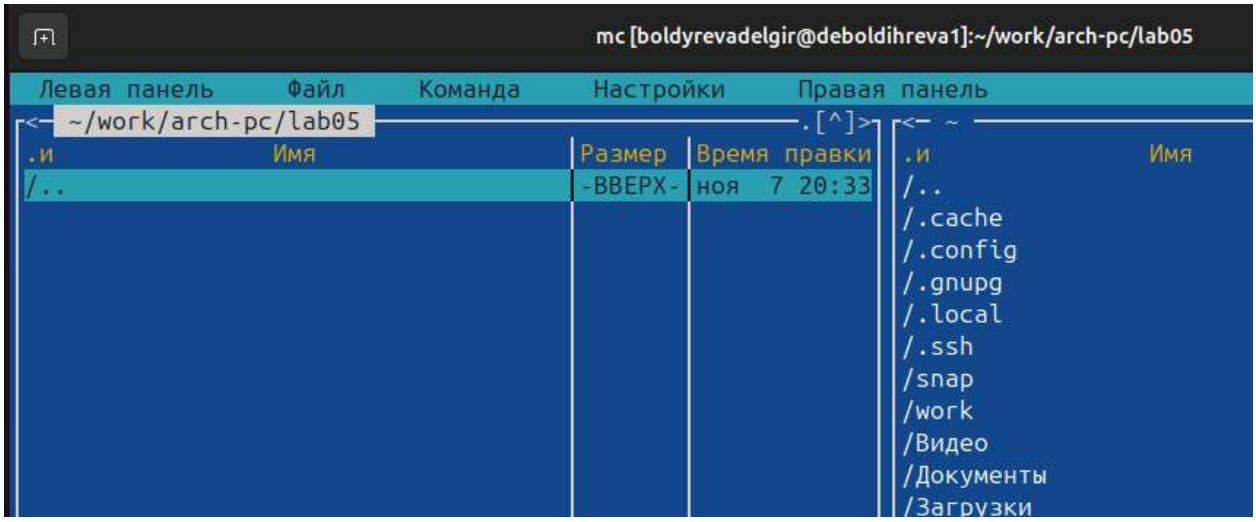


Рис 4.3



4.4 Воспользуемся строкой ввода и командой touch создадим файл lab5-1.asm (Рис 4.4)

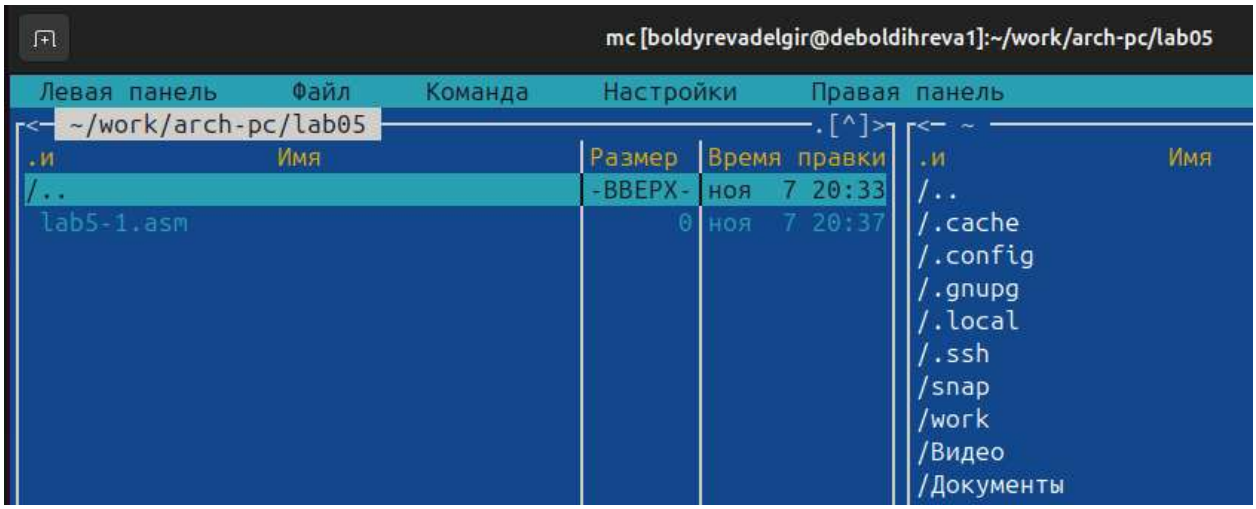


Рис 4.4

4.5 Далее откроем файл lab5-1.asm во встроенном редакторе для редактирования и введем текст программы из листинга, сохраняем изменения и закрываем файл (Рис 4.5)

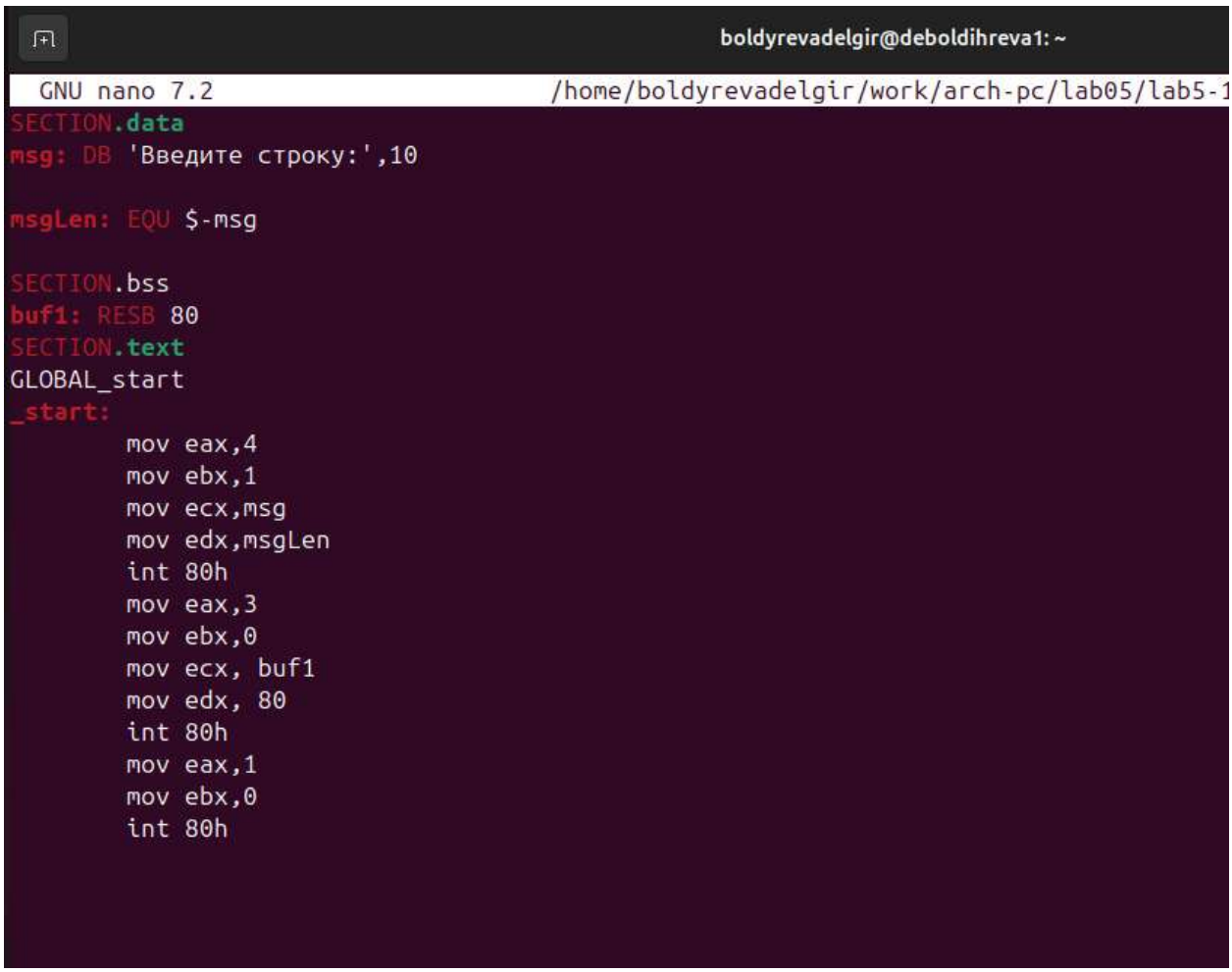


Рис 4.5

4.6 Оттранслируем текст программы lab5-1.asm в объектный файл. Выполним компоновку объектного файла и запустим получившийся исполняемый файл. Программа выводит строку 'Введите строку:' и ожидает ввода с клавиатуры. На запрос введем ФИО (рис 4.6-4.6.1)

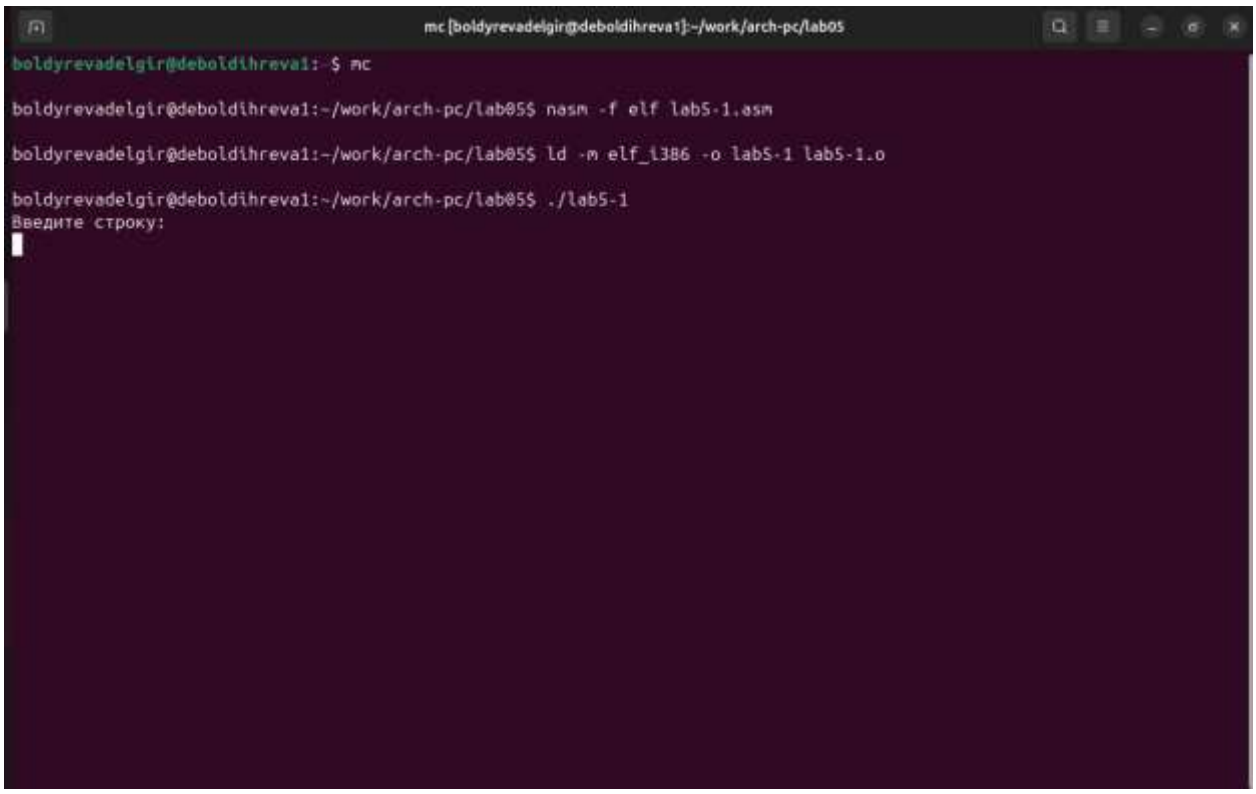


Рис 4.6

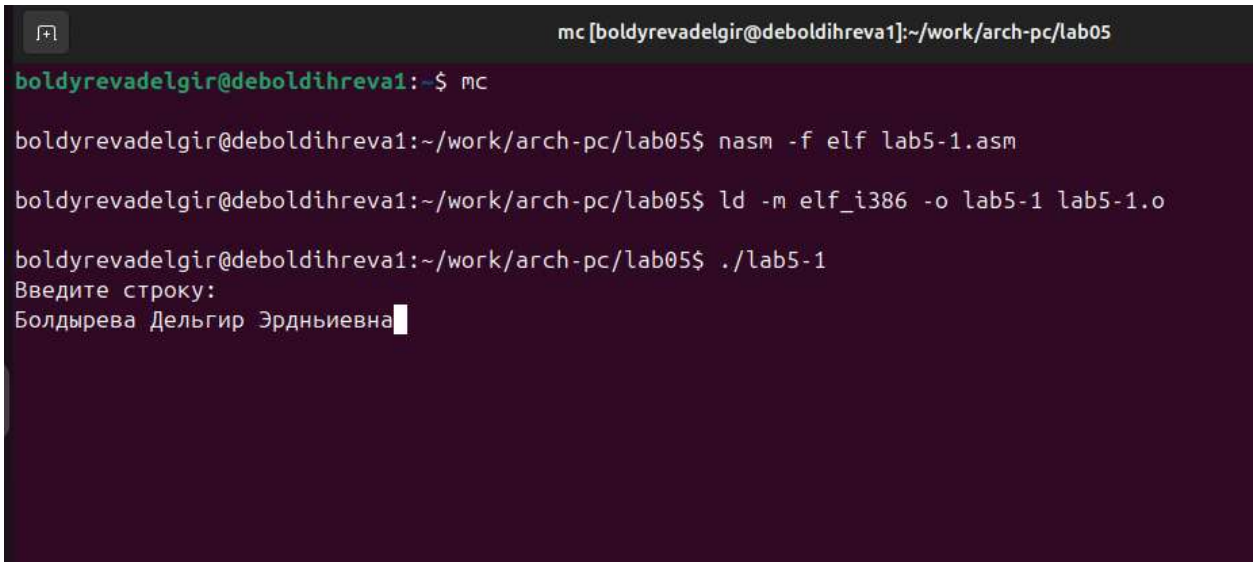


Рис 4.6.1

4.7 Скачиваем файл in_out.asm со страницы курса в Туис. Подключаемый файл in_out.asm должен лежать в том же каталоге, что и файл с программой, в которой он используется. С помощью функциональной клавиши f6 создадим копию файла lab5-1.asm с именем lab5-2.asm. Выделим файл lab5-1.asm, нажмем клавишу f6, введем имя файла lab5-2.asm и нажмем клавишу enter (рис 4.7)

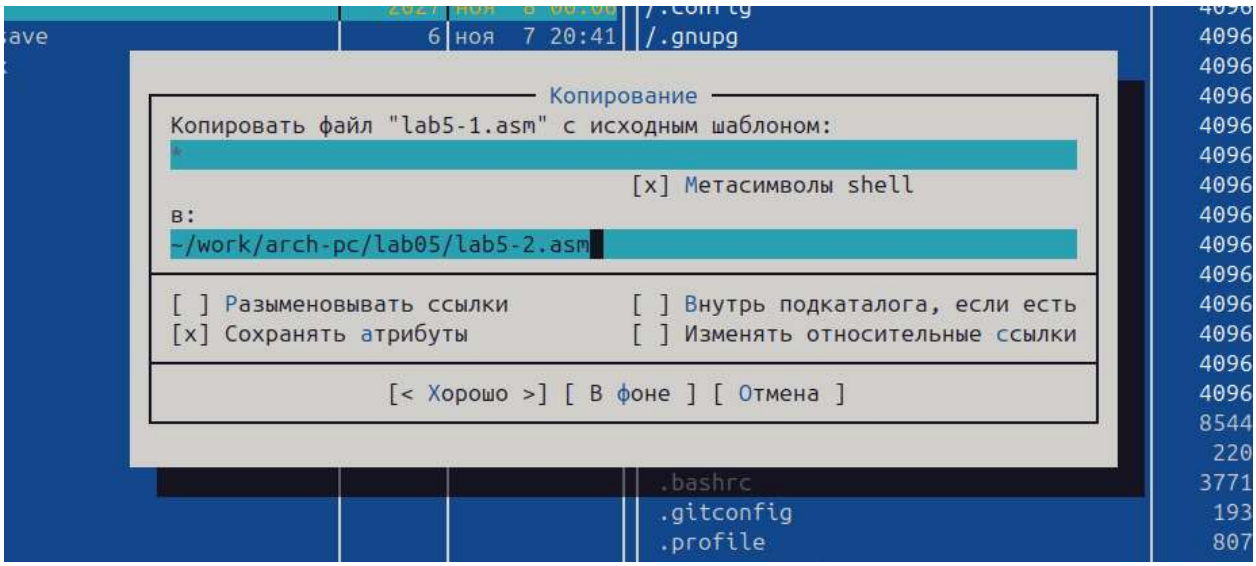


Рис 4.7

4.8 Исправим текст программы в файле lab5-2.asm с использование подпрограмм из внешнего файла in_out.asm в соответствии с листингом. Создадим исполняемый файл и проверим его работу.

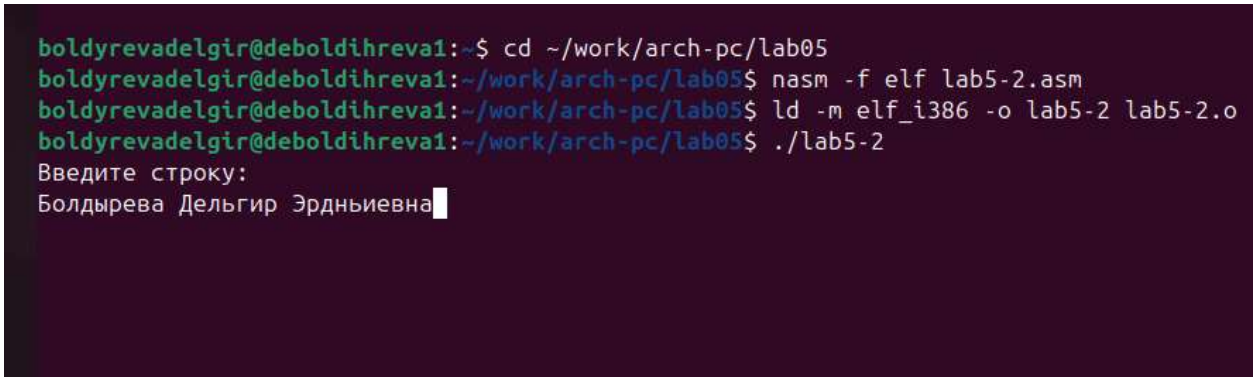


Рис 4.8

5 Задания для самостоятельной работы

1 Создадим копию файла lab5-1.asm. Внесем изменения в программу, так чтобы она работала по следующему алгоритму:

• вывести приглашение типа “Введите строку:”;

• ввести строку с клавиатуры;

• вывести введённую строку на экран

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

```
Вы можете вставить изображение из буфера обмена.

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,4 ; Системный вызов для выхода (sys_exit)
mov ebx,1; Выход с кодом возврата 0 (без ошибок)
mov ecx,buf1
mov edx,buf1
int 80h ; Вызов ядра
mov eax,1
mov ebx,0
int 80h
```

Рис 1

```
boldyrevadelgir@deboldihreva1:~/work/arch-pc/lab05$ nasm -f elf lab5-1.asm

boldyrevadelgir@deboldihreva1:~/work/arch-pc/lab05$ ./lab5-1
Введите строку:
Болдырева Дельгир Эрдниевна
```

Рис 1.1

2 Создадим копию файла lab5-2.asm. Исправим текст программы с использование под программ из внешнего файла in_out.asm, так чтобы она работала по следующему алгоритму: • вывести приглашениетипа“Введите строку:”;

• ввести строку с клавиатуры;

• вывести введённую строку на экран

mc [boldyrevadelgir@deboldihreva1]:~/work/arch-pc/lab05

```
/home/boldyrevadelgir/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 инициированных данных
msg: DB 'Введите строку:',0h ; сообщение
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

mov eax, msg ; запись адреса выводимого сообщения в `EAX`
call sprintLF ; 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, buf1; запись адреса переменной в 'EAX'
mov ebx, 80; запись длины вводимого сообщения в 'EBX'
call sprintLF
call quit ; вызов подпрограммы завершения
```

Рис 2

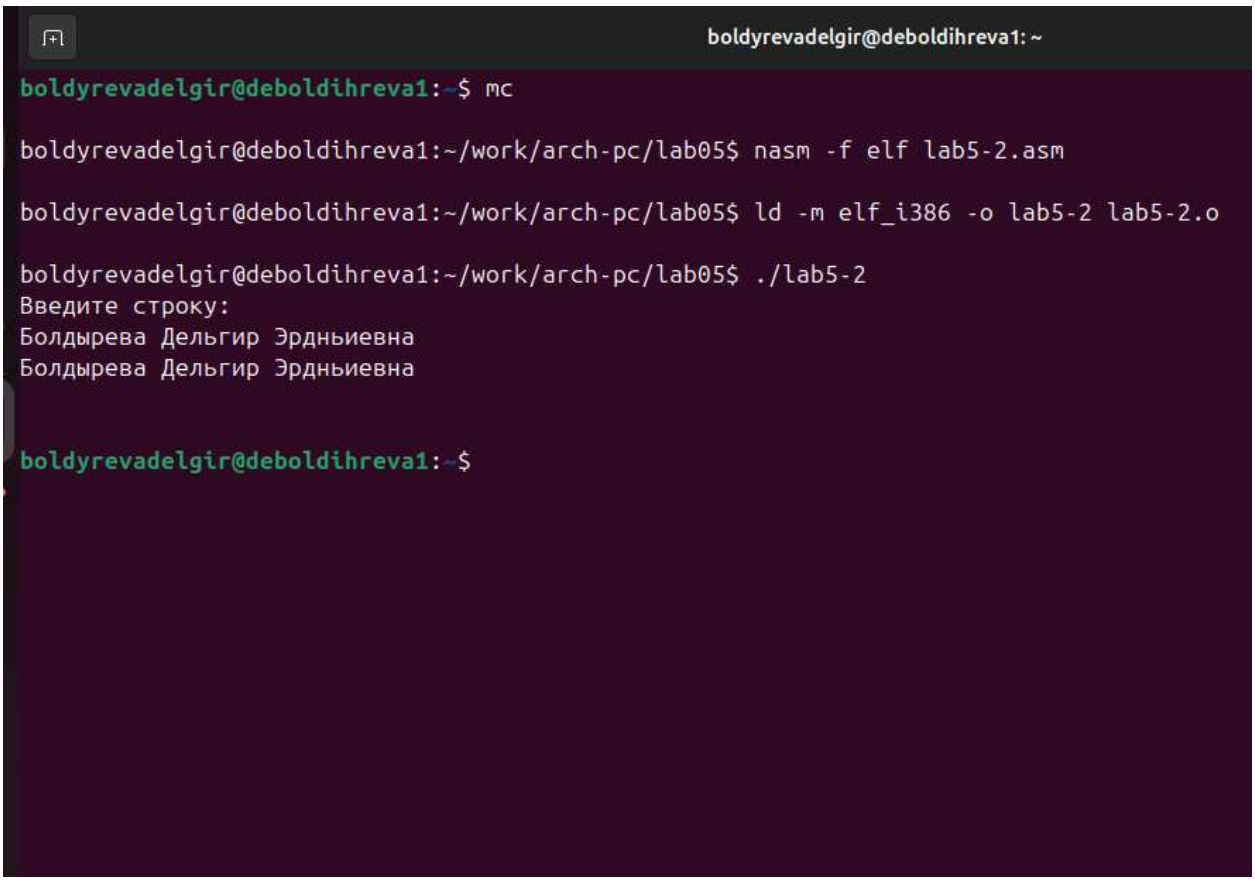

boldyrevadelgir@deboldihreva1: ~
boldyrevadelgir@deboldihreva1:~$ mc
boldyrevadelgir@deboldihreva1:~/work/arch-pc/lab05$ nasm -f elf lab5-2.asm
boldyrevadelgir@deboldihreva1:~/work/arch-pc/lab05$ ld -m elf_i386 -o lab5-2 lab5-2.o
boldyrevadelgir@deboldihreva1:~/work/arch-pc/lab05$ ./lab5-2
Введите строку:
Болдырева Дельгир Эрдньиевна
Болдырева Дельгир Эрдньиевна
boldyrevadelgir@deboldihreva1:~$

Рис 2.1

6 Вывод

В ходе выполнения данной лабораторной работы я приобрела практические навыки работы в Midnight Commander и освоила инструкции языка ассемблера mov и int.

7 Список литературы

1. GDB:TheGNUProjectDebugger.—URL:https://www.gnu.org/software/gdb/.

2. GNUBashManual.—2016.—URL:https://www.gnu.org/software/bash/manual/.

3. Midnight CommanderDevelopment Center.—2021.—URL: https://midnight-commander.org/.

4. NASMAssemblyLanguageTutorials.—2021.—URL: https://asmtutor.com/.

6. RobbinsA. Bash Pocket Reference.—O'Reilly Media,2016.—156 c.—ISBN 978-1491941591.

7. TheNASMdocumentation.—2021.—URL:https://www.nasm.us/docs.php.