Отчет по лабораторной работе №5

Дисциплина: Архитектура компьютера

Мустафина Аделя Юрисовна

Содержание

Цель работы

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

Задание

1. Изучение программы Midnight Commander и выполнение кода на языке ассемблера NASM.
2. Выполнение 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. Для активации оболочки Midnight Commander достаточно ввести в командной строке mc и нажать клавишу Enter. В Midnight Commander используются функциональные клавиши F1 — F10 , к которым привязаны часто выполняемые операции. Следующие комбинации клавиш облегчают работу с Midnight Commander: • Tab используется для переключениями между панелями; • ↑ и ↓ используется для навигации, Enter для входа в каталог или открытия файла (если в файле расширений mc.ext заданы правила связи определённых расширений файлов с инструментами их запуска или обработки); • Ctrl + u (или через меню Команда > Переставить панели) меняет местами содержимое правой и левой панелей; • Ctrl + o (или через меню Команда > Отключить панели) скрывает или возвращает панели Midnight Commander, за которыми доступен для работы командный интерпретатор оболочки и выводимая туда информация. • Ctrl + x + d (или через меню Команда > Сравнить каталоги) позволяет сравнить содержимое каталогов, отображаемых на левой и правой панелях.

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).

Таким образом, общая структура программы имеет следующий вид:

SECTION .data ; Секция содержит переменные, для … ; которых задано начальное значение SECTION .bss ; Секция содержит переменные, для … ; которых не задано начальное значение SECTION .text ; Секция содержит код программы GLOBAL _start _start: ; Точка входа в программу … ; Текст программы mov eax,1 ; Системный вызов для выхода (sys_exit) mov ebx,0 ; Выход с кодом возврата 0 (без ошибок) int 80h ; Вызов ядра 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 (учетверённое слово); • DT (define ten bytes) — определяет переменную размером в 10 байт.

Выполнение лабораторной работы

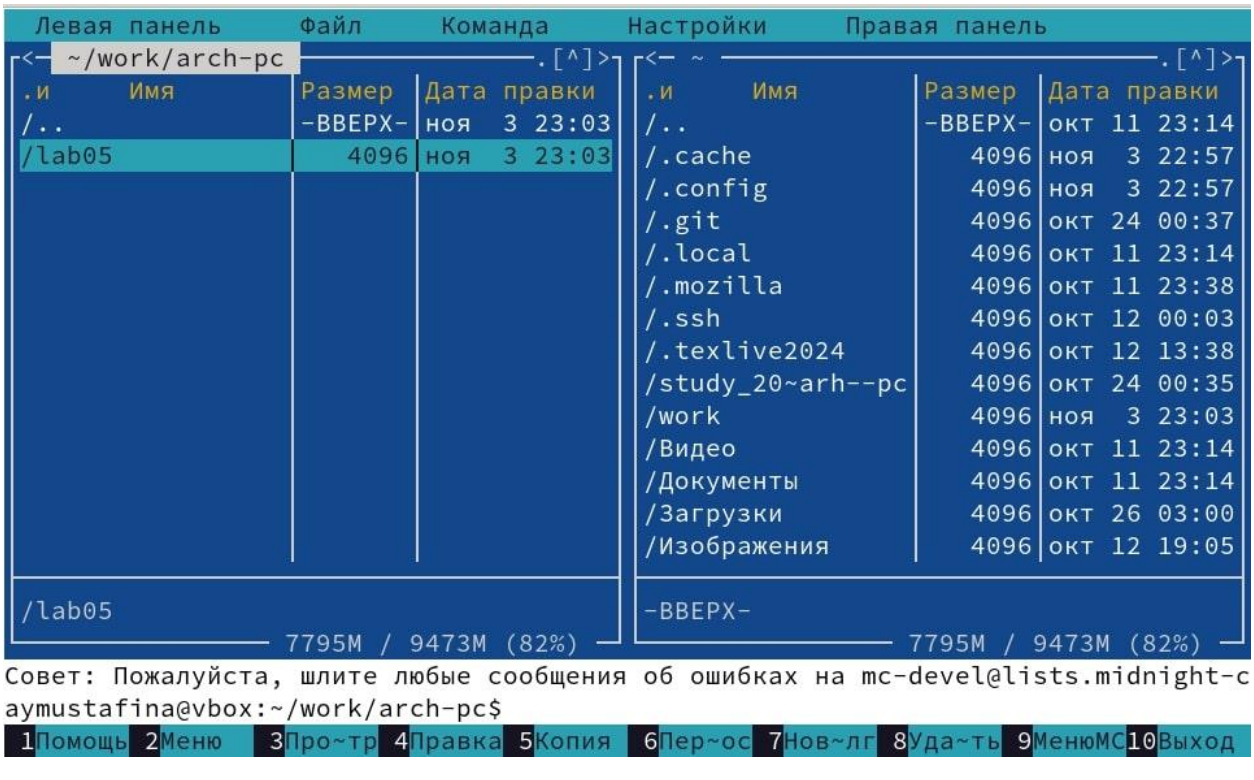
5.3. Порядок выполнения лабораторной работы

Открываю Midnight Commander с помощью команды mc (рис. [-@fig:001]).

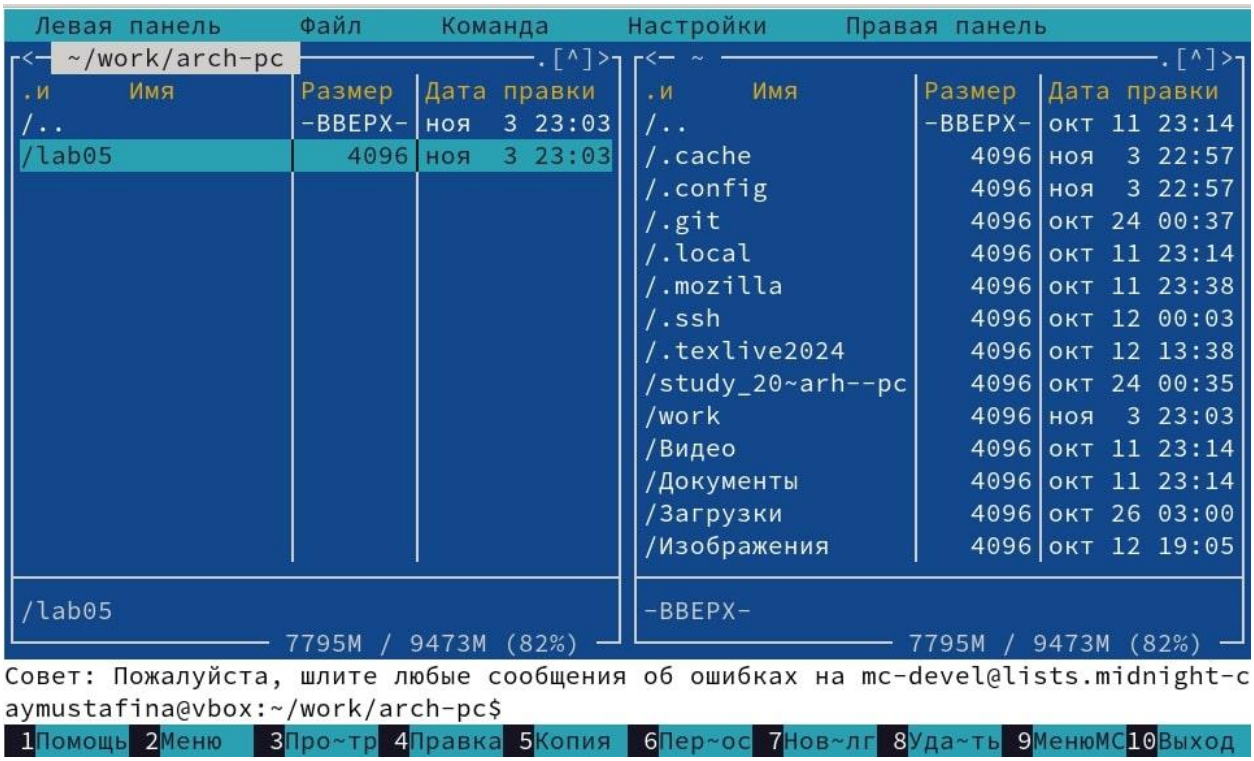
```
aymustafina@vbox:~$ mc
bash: mc: команда не найдена...
Установить пакет «mc», предоставляющий команду «mc»? [N/y] y


 * Ожидание в очереди...
Следующие пакеты должны быть установлены:
 gpm-libs-1.20.7-46.fc40.x86_64 Dynamic library for gpm
 mc-1:4.8.31-1.fc40.x86_64      User-friendly text console file manager and visu
al shell
 slang-2.3.3-5.fc40.x86_64      Shared library for the S-Lang extension language
Продолжить с этими изменениями? [N/y] y


 * Ожидание в очереди...
 * Ожидание аутентификации...
 * Ожидание в очереди...
 * Загрузка пакетов...
```

1

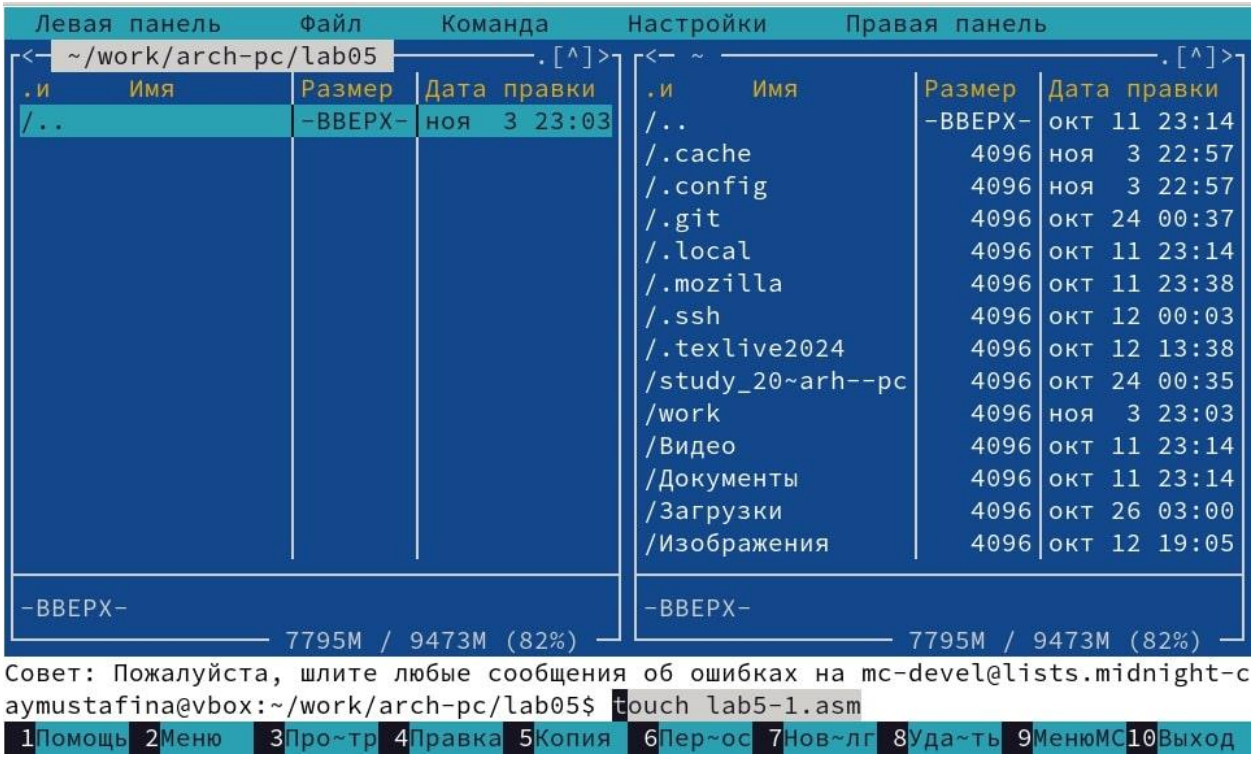
Захожу в директорию ~/work/arch-pc (рис. [-@fig:002]).



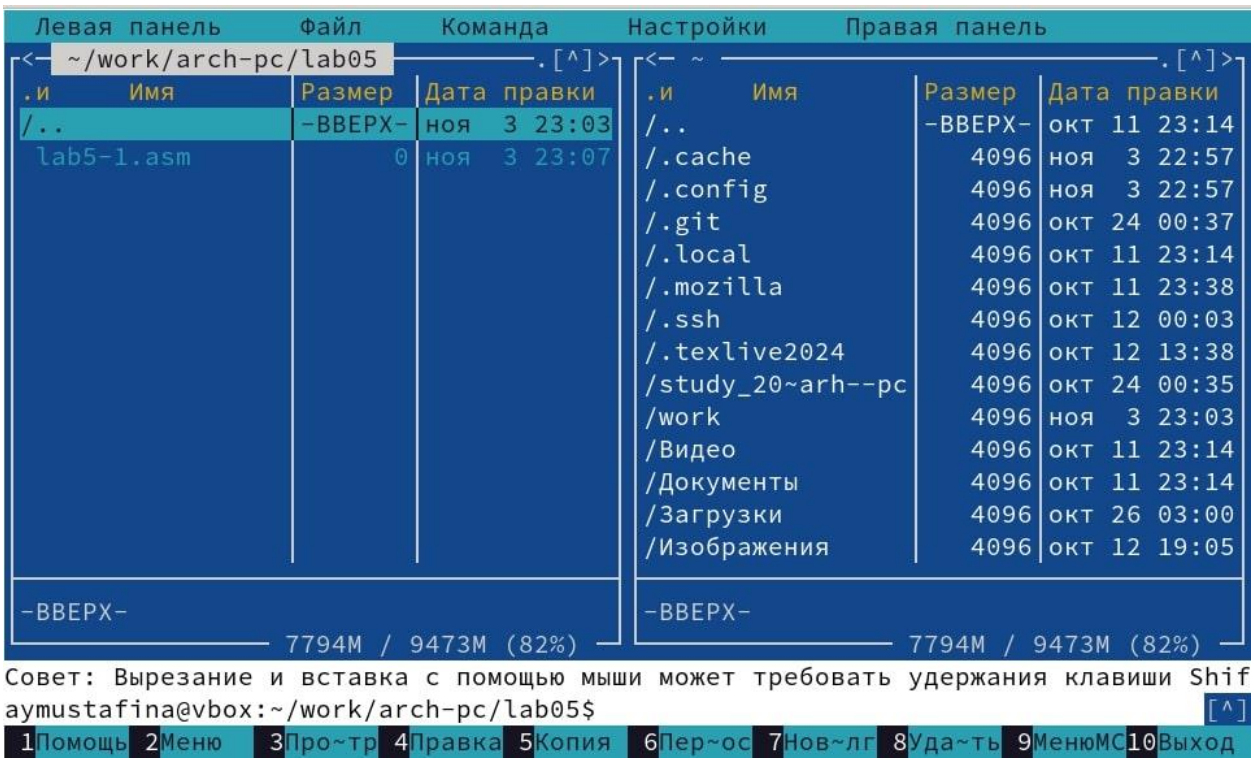
Создаю в этой директории папку с новым файлом с названием lab5-1.asm (рис. [-@fig:003]).



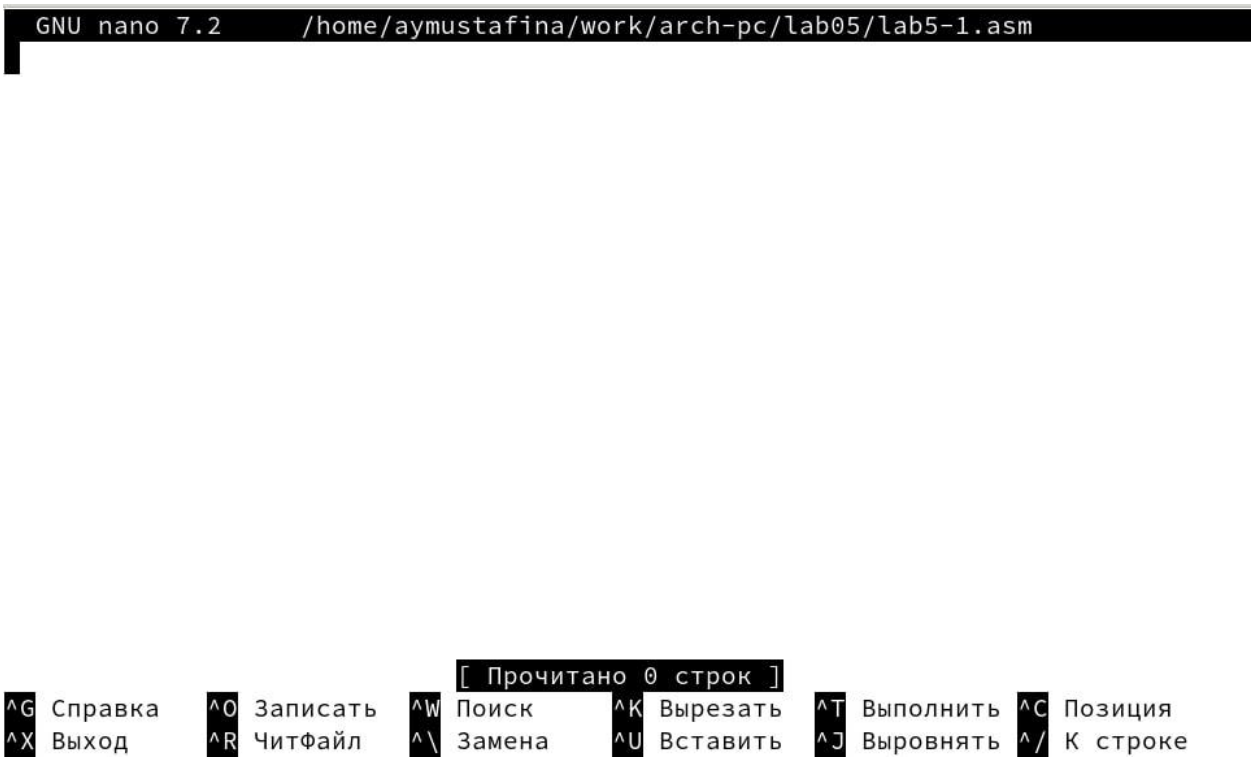
Создание файла с помощью функции touch (рис. [-@fig:004]).



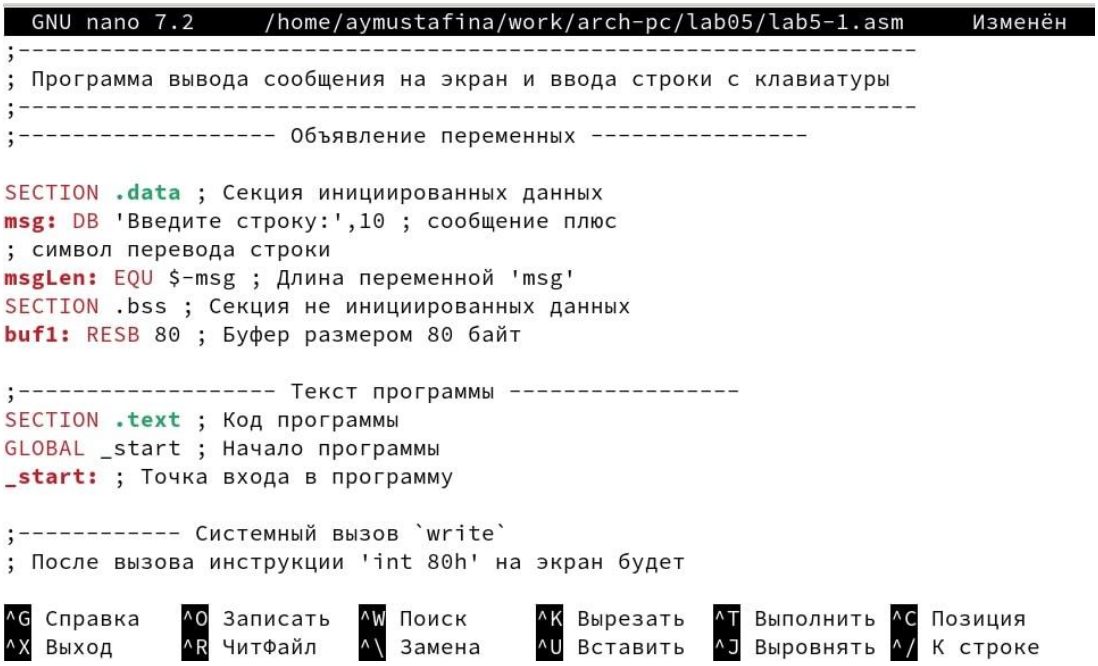
Созданный файл (рис. [-@fig:005]).



Открываю созданный файл с помощью функциональной клавиши F4 (рис. [-@fig:006]).



Ввожу текст из листинга (рис. [-@fig:007]).



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 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

;————————- 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` ————————- ;

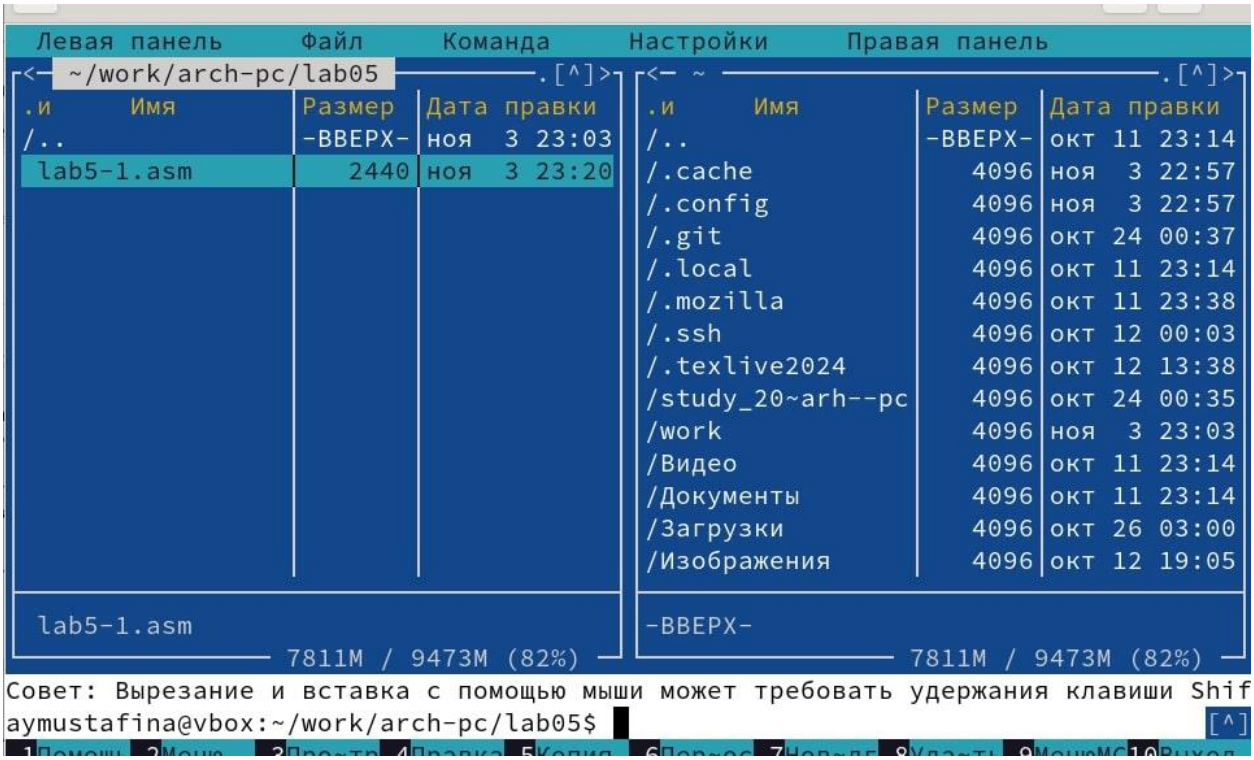
После вызова инструкции ‘int 80h’ программа завершит работу

mov eax,1 ; Системный вызов для выхода (sys_exit)

mov ebx,0 ; Выход с кодом возврата 0 (без ошибок)

int 80h ; Вызов ядра

Убеждаюсь, что файл содержит текст программы (рис. [-@fig:008]).



Транслирую текст программы lab5-1.asm в объектный файл, выполняю компановку объектного файла и запускаю его (рис. [-@fig:009]).

```
aymustafina@vbox:~$ cd ~/work/arch-pc/lab05/lab5-1.asm
bash: cd: /home/aymustafina/work/arch-pc/lab05/lab5-1.asm: Это не каталог
aymustafina@vbox:~$ cd ~/work/arch-pc/lab05
aymustafina@vbox:~/work/arch-pc/lab05$ ls
lab5-1.asm
aymustafina@vbox:~/work/arch-pc/lab05$ nasm -f elf64 lab5-1.asm
aymustafina@vbox:~/work/arch-pc/lab05$ ls
lab5-1.asm  lab5-1.o
aymustafina@vbox:~/work/arch-pc/lab05$ nasm -o obj.o -f elf64 -g -l list.ls
5-1.asm
aymustafina@vbox:~/work/arch-pc/lab05$ ls
lab5-1.asm  lab5-1.o  list.lst  obj.o
aymustafina@vbox:~/work/arch-pc/lab05$ ld -m elf_x86_64 lab5-1.o -o lab5-1
aymustafina@vbox:~/work/arch-pc/lab05$ ls
lab5-1  lab5-1.asm  lab5-1.o  list.lst  obj.o
aymustafina@vbox:~/work/arch-pc/lab05$ ld -m elf_x86_64 obj.o -o lab5
aymustafina@vbox:~/work/arch-pc/lab05$ ls
lab5  lab5-1  lab5-1.asm  lab5-1.o  list.lst  obj.o
```

Запуск файла (рис. [-@fig:010]).

```
aymustafina@vbox:~/work/arch-pc/lab05$ ./lab5-1
Введите строку:
Мустафина Аделя Юрисовна
aymustafina@vbox:~/work/arch-pc/lab05$
```

5.3.1. Подключение внешнего файла in_out.asm

Скопировала файл ab5-1.asm с именем lab5-2.asm (рис. [-@fig:011]).

```
aymustafina@vbox:~/work/arch-pc/lab05$ cp lab5-1.asm lab5-2.asm
aymustafina@vbox:~/work/arch-pc/lab05$ ls
lab5  lab5-1  lab5-1.asm  lab5-1.o  lab5-2.asm  list.lst  obj.o
```

Я скачала файл in_out.asm со страницы курса в ТУИС и переместила его в каталог с программами лабораторной работы (рис. [-@fig:012]).

```
aymustafina@vbox:~/Загрузки$ cp in_out.asm ~/work/arch-pc/lab05
aymustafina@vbox:~/Загрузки$ cd ~
aymustafina@vbox:~$ cd ~/work/arch-pc/lab05
aymustafina@vbox:~/work/arch-pc/lab05$ ls
in_out.asm  lab5  lab5-1  lab5-1.asm  lab5-1.o  lab5-2.asm  list.lst  obj.o
aymustafina@vbox:~/work/arch-pc/lab05$
```

Меняю текст в программе lab5-2.asm в соответствии с листингом 5.2 (рис. [-@fig:013]).

```
aymustafina@vbox:~/work/arch-pc/lab05$ nasm -f elf64 lab5-2.asm
aymustafina@vbox:~/work/arch-pc/lab05$ nasm -o obj.o -f elf64 -g -l list.lst lab
5-2.asm
aymustafina@vbox:~/work/arch-pc/lab05$ ld -m elf_x86_64 lab5-2.o -o lab5-2
aymustafina@vbox:~/work/arch-pc/lab05$ ld -m elf_x86_64 obj.o -o lab52
aymustafina@vbox:~/work/arch-pc/lab05$ ls
in_out.asm  lab5-1      lab5-1.o  lab52       lab5-2.o  obj.o
lab5        lab5-1.asm  lab5-2    lab5-2.asm  list.lst
aymustafina@vbox:~/work/arch-pc/lab05$
```

Транслирую текст программы lab5-2.asm в объектный файл, выполняю компановку объектного файла (рис. [-@fig:014]).

```
 GNU nano 7.2     /home/aymustafina/work/arch-pc/lab05/lab5-2.asm     Изменён
;------------------------------------------------------------------------
; 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
call quit ; вызов подпрограммы завершения


^G Справка   ^O Записать  ^W Поиск     ^K Вырезать  ^T Выполнить ^C Позиция
^X Выход     ^R ЧитФайл   ^\ Замена    ^U Вставить  ^J Выровнять ^/ К строке
```

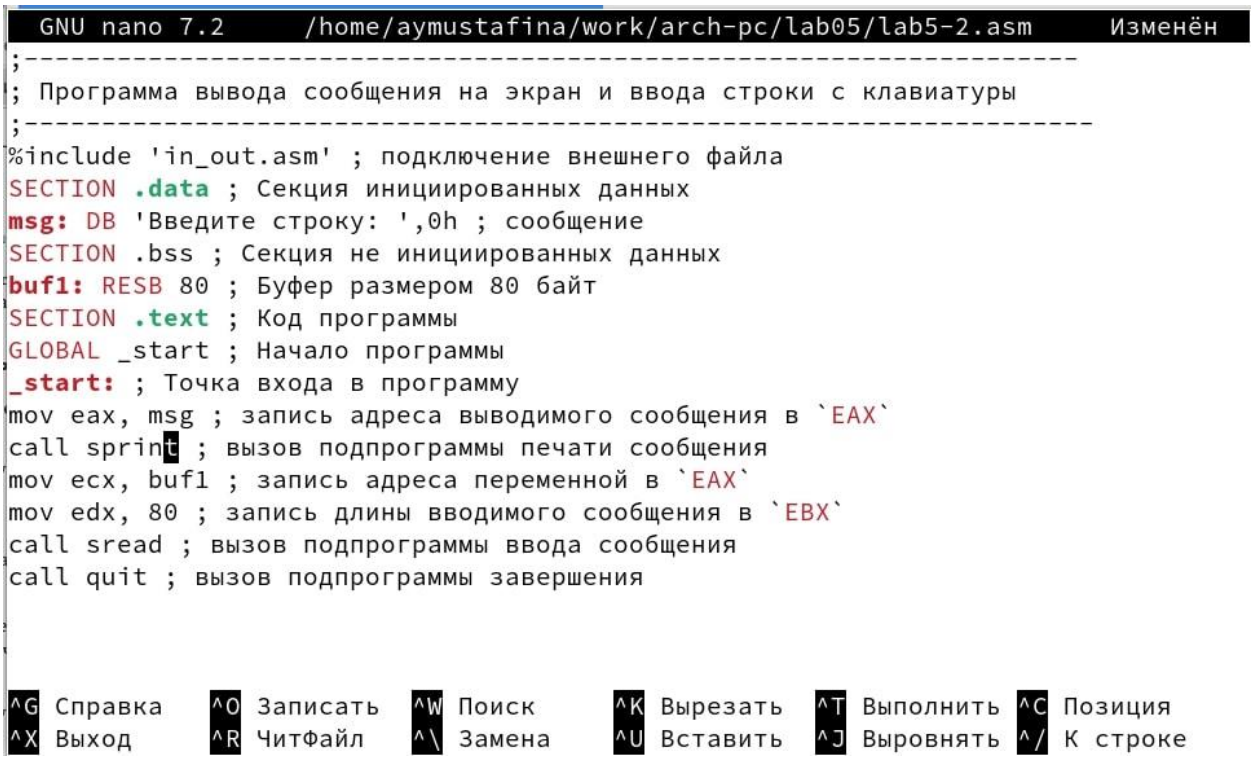
Листинг 5.2. Программа вывода сообщения на экран и ввода строки с клавиатуры с использованием файла in_out.asm

```
;------------------------------------------------
; 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
call quit ; вызов подпрограммы завершения
```

Запуск программы lab5-2.asm (рис. [-@fig:015]).

```
aymustafina@vbox:~/work/arch-pc/lab05$ ./lab5-2
Введите строку:
Мустафина Аделя Юрисовна
aymustafina@vbox:~/work/arch-pc/lab05$
```

Меняю текст в программе lab5-2.asm, заменив подпрограмму sprintLF на sprint (рис. [-@fig:016]).



Листинг 5.3. Измененная программа вывода сообщения на экран и ввода строки с клавиатуры с использованием файла in_out.asm

;—————————————————————————————

; Измененная 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 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

call quit ; вызов подпрограммы завершения

Запуск измененной программы lab5-2.asm (рис. [-@fig:017]).

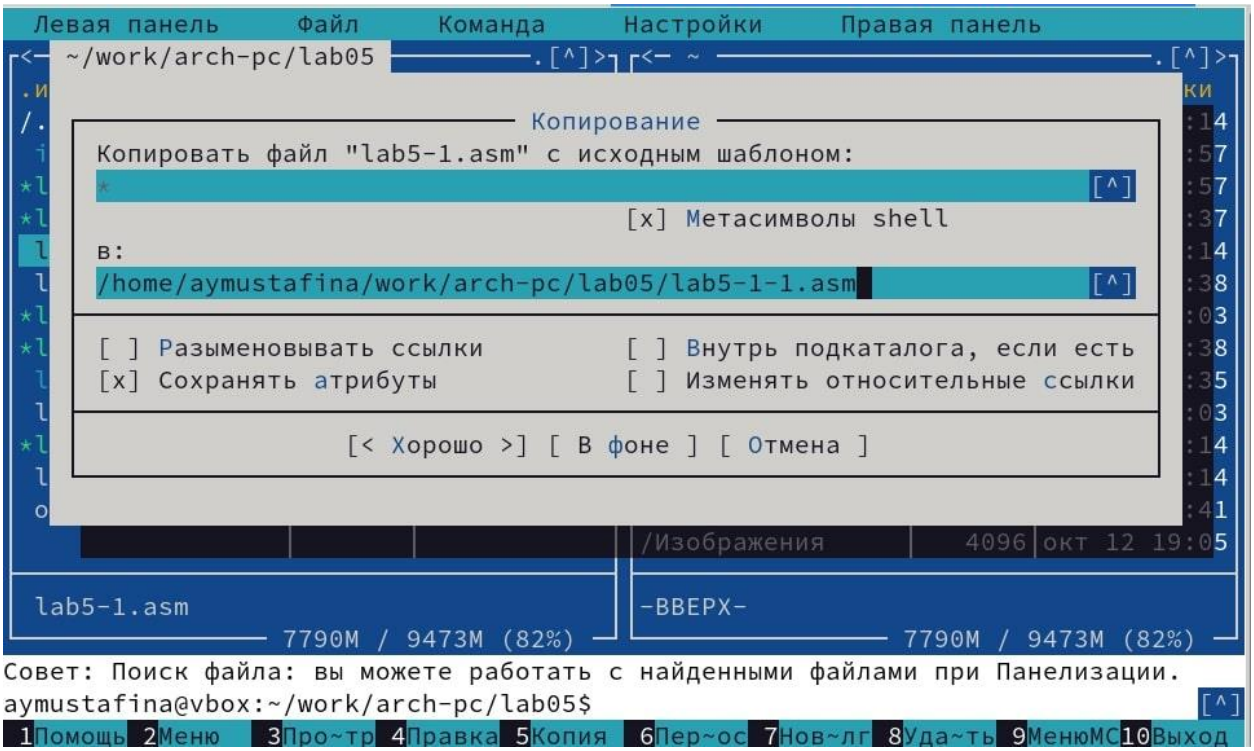
```
aymustafina@vbox:~/work/arch-pc/lab05$ nasm -f elf lab5-2.asm
aymustafina@vbox:~/work/arch-pc/lab05$ ld -m elf_i386 lab5-2.o -o lab5-2-2
aymustafina@vbox:~/work/arch-pc/lab05$ ./lab5-2-2
Введите строку: Мустафина Аделя Юрисовна
aymustafina@vbox:~/work/arch-pc/lab05$
```

Разница между этими исполняемыми файлами заключается в том, что в первом варианте при запуске запрашивается ввод с новой строки, а во втором при запуске ввод происходит без перехода на новую строку.

Выполнение заданий для самостоятельной работы

1.

Создаю копию файла lab5-1.asm с названием lab5-1-1.asm (рис. [-@fig:018]).



И вношу изменения в эту программу такие, что при запуске она выводит приглашение и просит ввести строку с клавиатуры, а после снова выводит введенную пользователем строку (рис. [-@fig:019]).

```
aymustafina@vbox:~/work/arch-pc/lab05$ nasm -f elf lab5-1-1.asm
aymustafina@vbox:~/work/arch-pc/lab05$ ld -m elf_i386 lab5-1-1.o -o lab5-1-1
aymustafina@vbox:~/work/arch-pc/lab05$ ./lab5-1-1
Введите строку:
Мустафина Аделя Юрисовна
Мустафина Аделя Юрисовна
aymustafina@vbox:~/work/arch-pc/lab05$
```

Листинг для первой программы для самостоятельной работы

;————————————————————————

; Программа вывода сообщения на экран и ввода строки с клавиатуры

;————————————————————————

;————————- Объявление переменных ————————-

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

;————————- 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

mov eax, 4;

mov ebx, 1;

mov ecx, buf1;

mov edx, buf1;

int 80h;

;———— Системный вызов `exit` ————————-

; После вызова инструкции ‘int 80h’ программа завершит работу

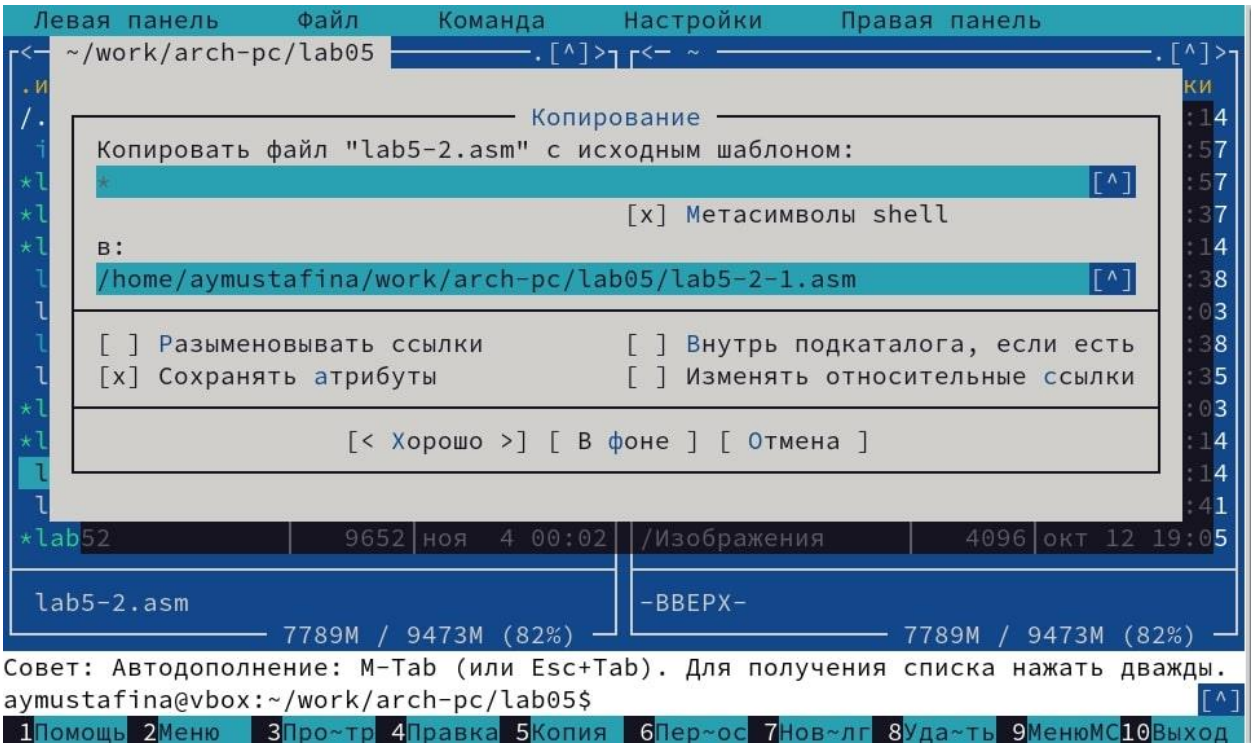
mov eax,1 ; Системный вызов для выхода (sys_exit)

mov ebx,0 ; Выход с кодом возврата 0 (без ошибок)

int 80h ; Вызов ядра

2.

Создаю копию файла lab5-2.asm с названием lab5-2-1.asm (рис. [-@fig:020]).



И вношу изменения в эту программу такие, что при запуске она выводит приглашение и просит ввести строку с клавиатуры, а после снова выводит введенную пользователем строку без перехода на новую строку (рис. [-@fig:021]).

```
lab5-2-1.asm       [-M--]  0 L:[  4+20  24/ 24] *(1273/1273b) <EOF>       [*][X]
%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 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;
mov ebx, 1;
mov ecx, buf1;
int 80h;

call quit ; вызов подпрограммы завершения
 1Помощь 2Сохран 3Блок  4Замена 5Копия  6Пер~ть 7Поиск  8Уда~ть 9МенюМС10Выход
```

Запуск программы lab5-2-1.asm (рис. [-@fig:022]).

```
aymustafina@vbox:~/work/arch-pc/lab05$ nasm -f elf lab5-2-1.asm
aymustafina@vbox:~/work/arch-pc/lab05$ ld -m elf_i386 lab5-2-1.o -o lab5-2-1
aymustafina@vbox:~/work/arch-pc/lab05$ ./lab5-2-1
Введите строку: Мустафина Аделя Юрисовна
Мустафина Аделя Юрисовна
aymustafina@vbox:~/work/arch-pc/lab05$
```

Листинг для второй программы для самостоятельной работы

;——————————————————————————

; 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 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;

mov ebx, 1;

mov ecx, buf1;

int 80h;

call quit ; вызов подпрограммы завершения
```

Выводы

При выполнении лабораторной работы я научилась работать в Midnight Commander. И изучила основы программ для вывода и ввода на языке ассемблера.

Список литературы

1. Лабораторная работа №6