Отчет по лабораторной работе №5

Дисциплина: Архитектура компьютера

Мустафина Аделя Юрисовна

Содержание

[1 Цель работы 2](#__RefHeading___Toc705_2757482149)

[2 Задание 2](#__RefHeading___Toc707_2757482149)

[3 Теоретическое введение 2](#__RefHeading___Toc709_2757482149)

[4 Выполнение лабораторной работы 3](#__RefHeading___Toc711_2757482149)

[4.1 5.3. Порядок выполнения лабораторной работы 3](#__RefHeading___Toc713_2757482149)

[4.1.1 Листинг 5.1. Программа вывода сообщения на экран и ввода строки с клавиатуры 6](#__RefHeading___Toc715_2757482149)

[4.2 5.3.1. Подключение внешнего файла in\_out.asm 8](#__RefHeading___Toc717_2757482149)

[4.2.1 Листинг 5.2. Программа вывода сообщения на экран и ввода строки с клавиатуры c использованием файла in\_out.asm 9](#__RefHeading___Toc719_2757482149)

[4.2.2 Листинг 5.3. Измененная программа вывода сообщения на экран и ввода строки с клавиатуры с использованием файла in\_out.asm 10](#__RefHeading___Toc721_2757482149)

[5 Выполнение заданий для самостоятельной работы 11](#__RefHeading___Toc723_2757482149)

[5.1 1. 11](#__RefHeading___Toc725_2757482149)

[5.1.1 Листинг для первой программы для самостоятельной работы 12](#__RefHeading___Toc727_2757482149)

[5.2 2. 13](#__RefHeading___Toc729_2757482149)

[5.2.1 Листинг для второй программы для самостоятельной работы 14](#__RefHeading___Toc731_2757482149)

[6 Выводы 14](#__RefHeading___Toc733_2757482149)

[7 Список литературы 14](#__RefHeading___Toc735_2757482149)

# 1 Цель работы

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

# 2 Задание

1. Изучение программы Midnight Commander и выполнение кода на языке ассемблера NASM.
2. Выполнение самостоятельной работы.

# 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. 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 содер- жимое 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.

# 4 Выполнение лабораторной работы

## 4.1 5.3. Порядок выполнения лабораторной работы

Открываю Midnight Commander с помощью команды mc (рис. 1).

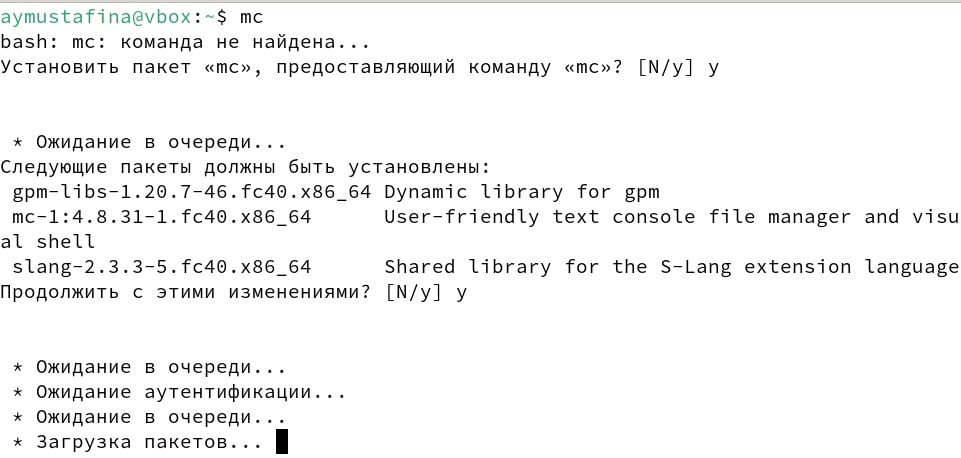


Рис. 1: 1. Открытие

Захожу в директорию ~/work/arch-pc (рис. 2).

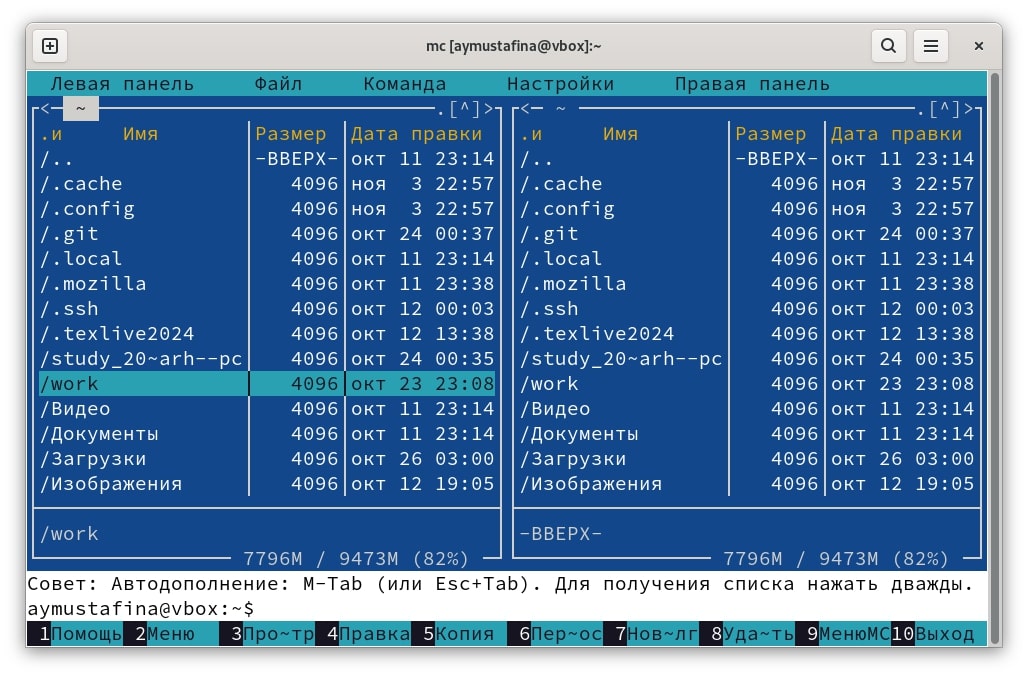


Рис. 2: 2. Перемещение по директории

Создаю в этой директории папку с новым файлом с названием lab5-1.asm (рис. 3).

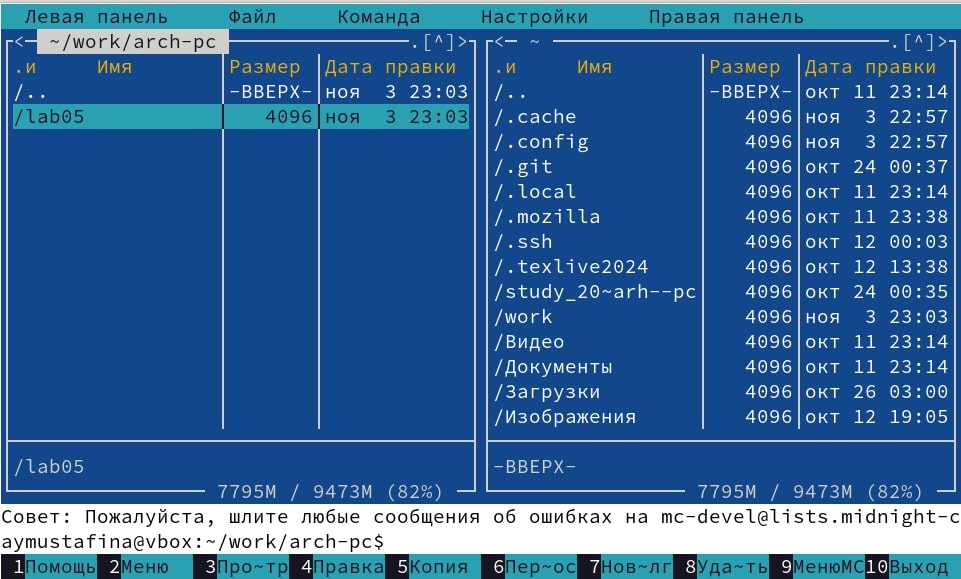


Рис. 3: 3. Создание папки

Создание файла с помощью функции touch (рис. 4).

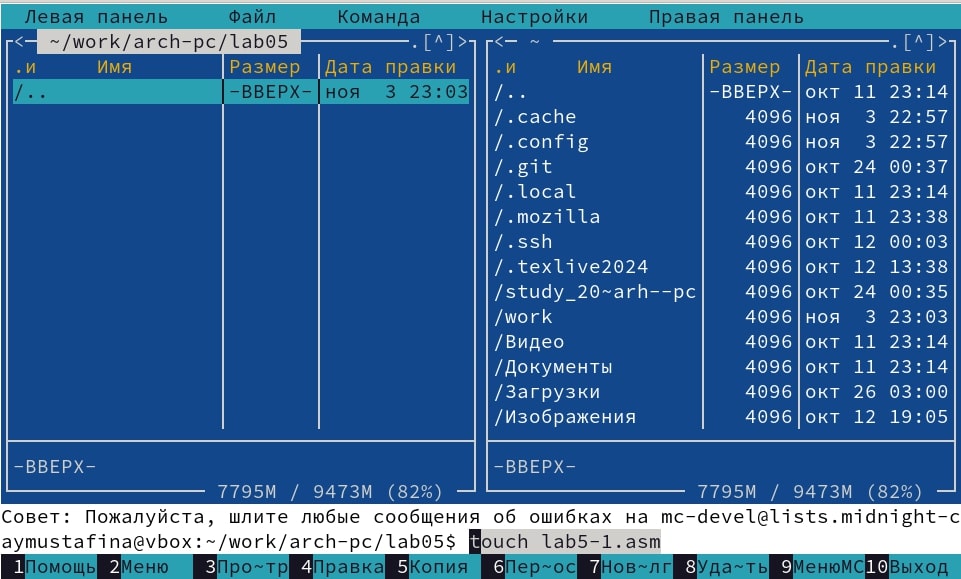


Рис. 4: 4. Создание файлы

Созданный файл (рис. 5).

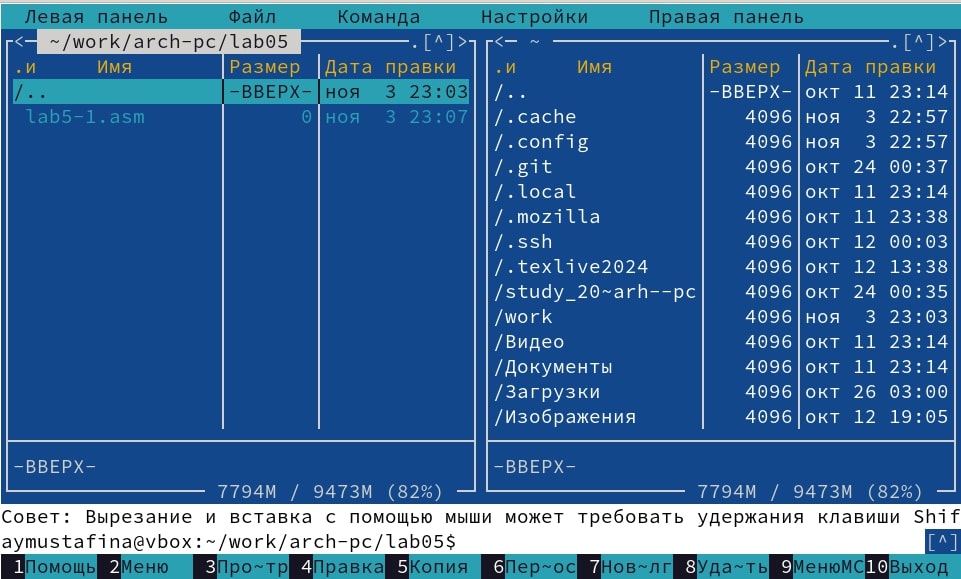


Рис. 5: 5. Файл

Открываю созданный файл с помощью функциональной клавиши F4 (рис. 6).

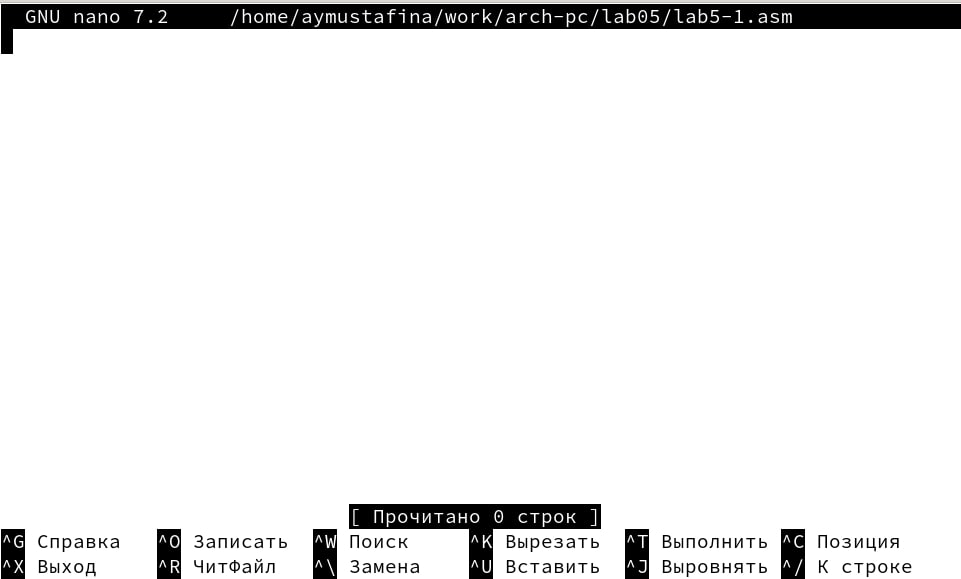


Рис. 6: 6. Файл

Ввожу текст из листинга (рис. 7).

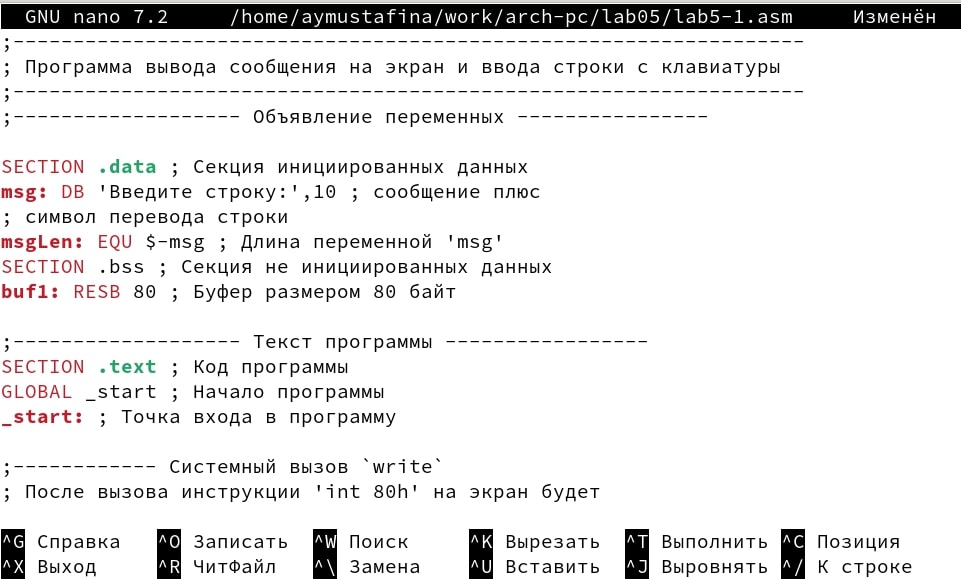


Рис. 7: 7. Ввод текста

### 4.1.1 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

;———— Cистемный 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. 8).

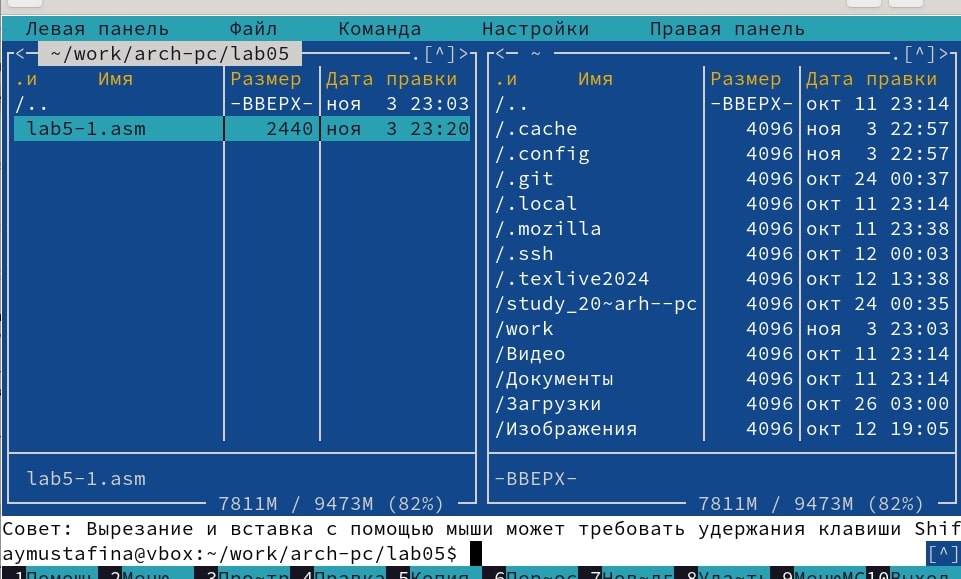


Рис. 8: 8. Текст программы

Транслирую текст программы lab5-1.asm в объектный файл, выполняю компоновку объектного файла и запускаю его (рис. 9).

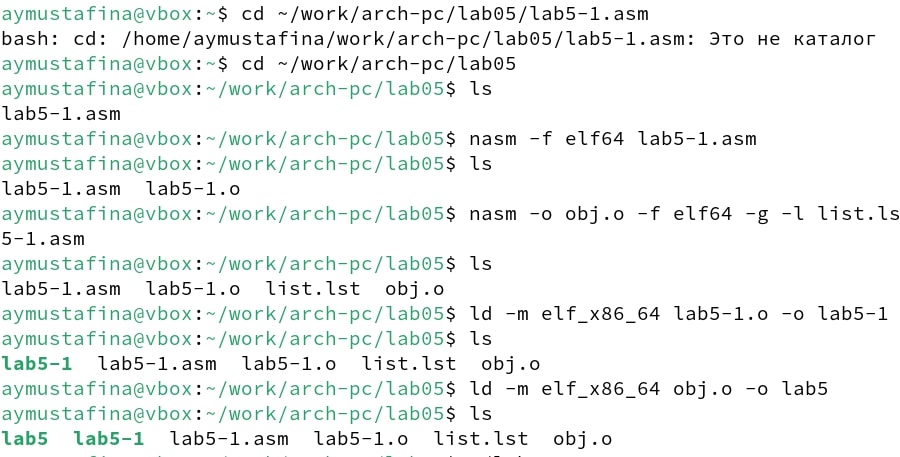


Рис. 9: 9. Компоновка файла

Запуск файла (рис. 10).

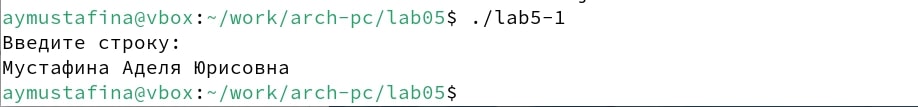


Рис. 10: 10. Запуск файла

## 4.2 5.3.1. Подключение внешнего файла in\_out.asm

Скопировала файл ab5-1.asm с именем lab5-2.asm (рис. 11).

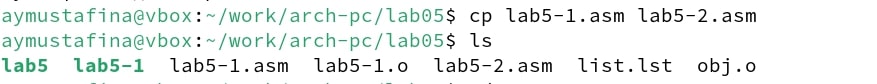


Рис. 11: 11. Копирование файла

Я скачала файл in\_out.asm со страницы курса в ТУИС и переместила его в каталог с программами лабораторной работы (рис. 12).

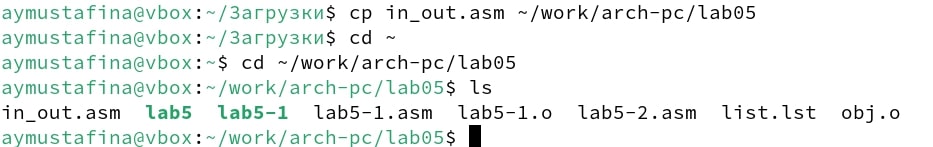


Рис. 12: 12. Перемещение

Меняю текст в программе lab5-2.asm в соответствии с листингом 5.2 (рис. 13).

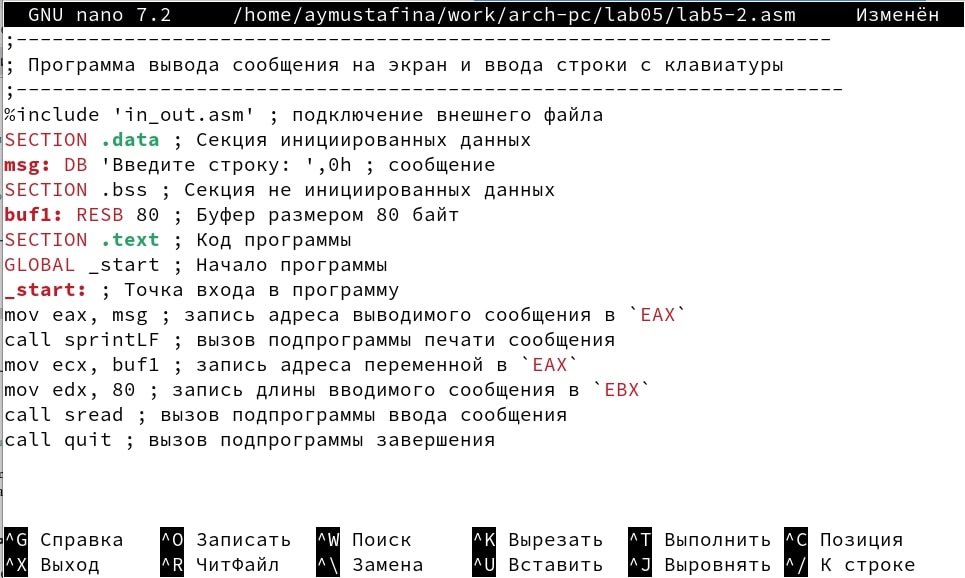


Рис. 13: 13. Текст программы

Транслирую текст программы lab5-2.asm в объектный файл, выполняю компоновку объектного файла (рис. 14).

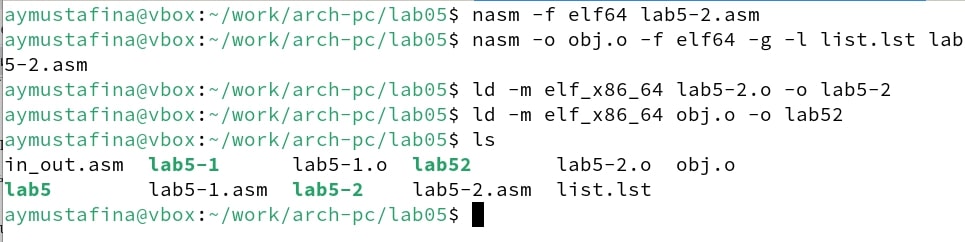


Рис. 14: 14. Компоновка файла

### 4.2.1 Листинг 5.2. Программа вывода сообщения на экран и ввода строки с клавиатуры c 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. 15).

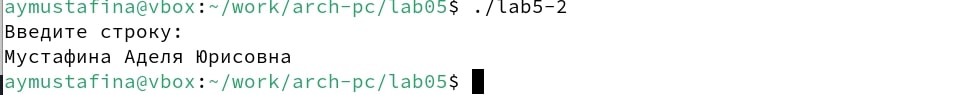


Рис. 15: 15. Запуск программы

Меняю текст в программе lab5-2.asm, заменив подпрограмму sprintLF на sprint (рис. 16).

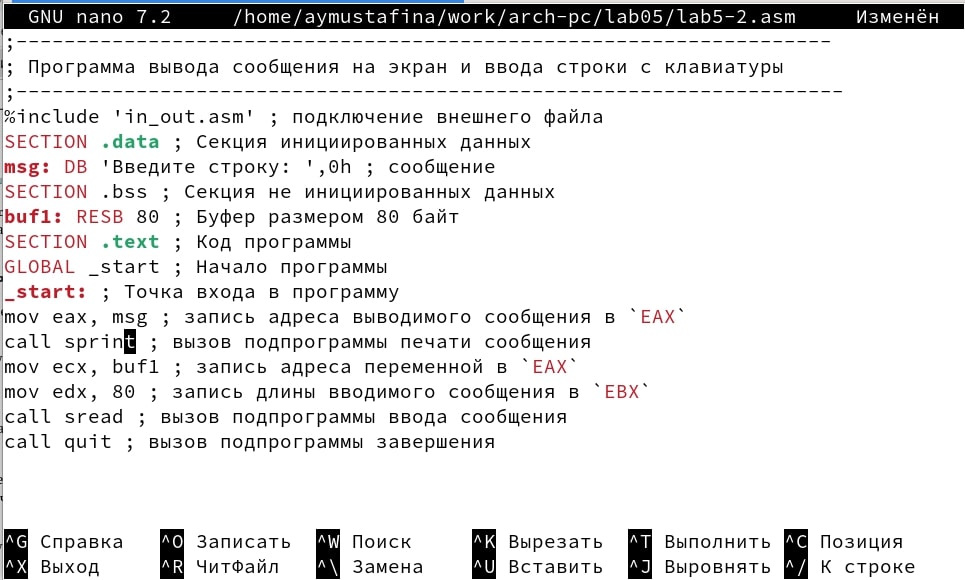


Рис. 16: 16. Изменения в тексте

### 4.2.2 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. 17).

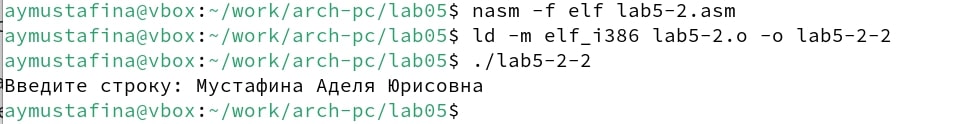


Рис. 17: 17. Запуск

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.

# 5 Выполнение заданий для самостоятельной работы

## 5.1 1.

Создаю копию файла lab5-1.asm с названием lab5-1-1.asm (рис. 18).

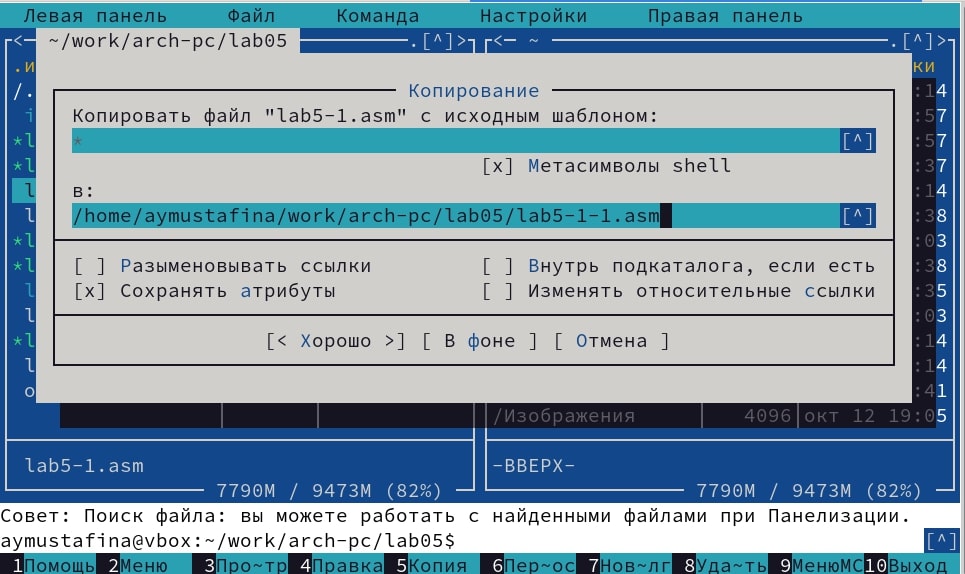


Рис. 18: 18. Копирование файла

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. 19).

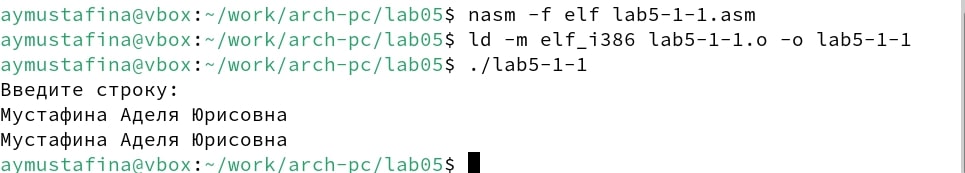


Рис. 19: 19. Запуск файла

### 5.1.1 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

;———— Cистемный 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

## 5.2 2.

Создаю копию файла lab5-2.asm с названием lab5-2-1.asm (рис. 20).

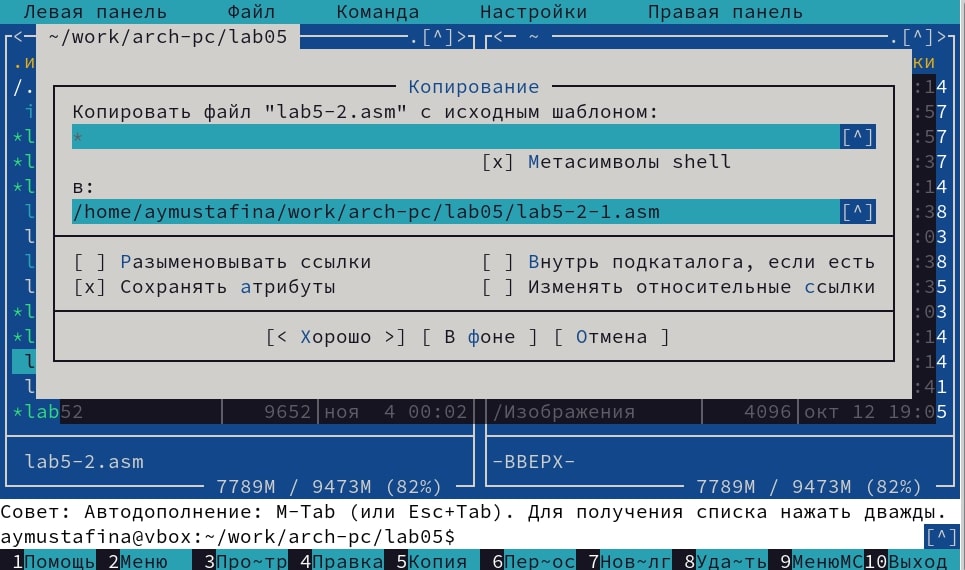


Рис. 20: 20. Копирование файла

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. 21).

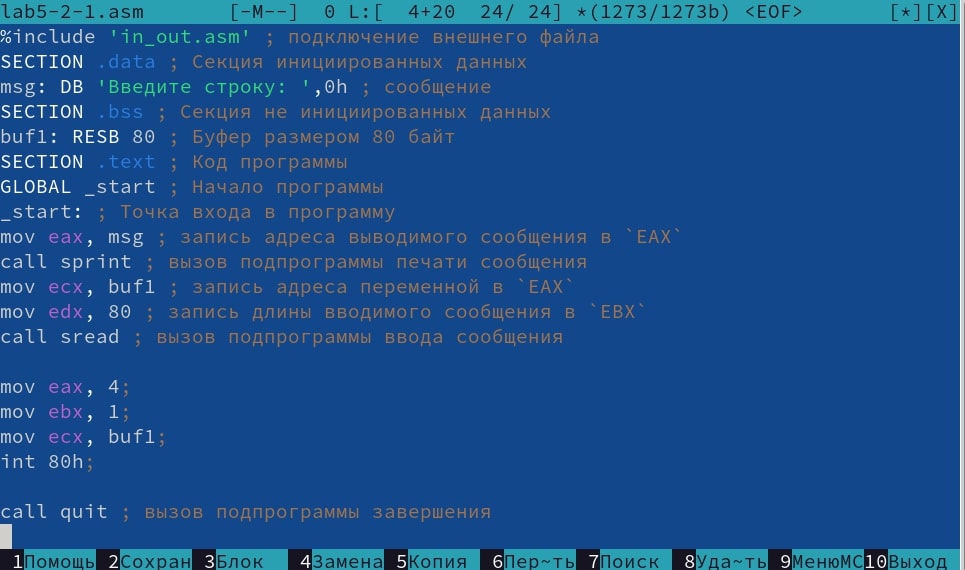


Рис. 21: 21. Изменения в файле

Запуск программы lab5-2-1.asm (рис. 22).

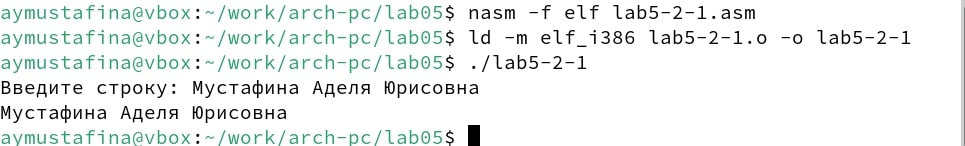


Рис. 22: 22. Запуск

### 5.2.1 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

# 6 Выводы

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

# 7 Список литературы

1. [Лабораторная работа №6](https://esystem.rudn.ru/pluginfile.php/2089085/mod_resource/content/0/Лабораторная%20работа%20№5.%20Основы%20работы%20с%20Midnight%20Commander%20%28%29.%20Структура%20программы%20на%20языке%20ассемблера%20NASM.%20Системные%20вызовы%20в%20ОС%20GNU%20Linux.pdf)