**Отчёт по лабораторной работе №5**

#### Дисциплина: архитектура компьютеров и операционные системы

### Хань Цзянтао

**Содержание**

1. [Цель работы](#_bookmark0) 5
2. [Задание](#_bookmark1) 6
3. [Теоретическое введение](#_bookmark2) 7
4. [Выполнение лабораторной работы](#_bookmark4) 11
   1. [Начало работы с Midnight Commander](#_bookmark5) 11
   2. [Подключение внешнего файла in\_out.asm](#_bookmark12) 14
   3. [Задания для самостоятельной работы](#_bookmark19) 17
5. [Выводы](#_bookmark26) 23

[Список литературы](#_bookmark27) 24

**Список иллюстраций**

* 1. [Интерфейс Midnight Commander](#_bookmark6) 11
  2. [Создание каталога в Midnight Commander](#_bookmark7) 12
  3. [Создание файла lab5-1.asm](#_bookmark8) 12
  4. [Редактирование текстового файла в mc](#_bookmark9) 13
  5. [Проверка наличия программы в файле](#_bookmark10) 13
  6. [Запуск исполняющего файла](#_bookmark11) 14
  7. [Копирование файла in\_out.asm](#_bookmark13) 14
  8. [Создание файла lab5-2.asm](#_bookmark14) 15
  9. [Исправление текста программы](#_bookmark15) 16
  10. [Запуск программы исполняемого файла](#_bookmark16) 16
  11. [Изменения в коде программы](#_bookmark17) 16
  12. [Запуск программы измененного файла](#_bookmark18) 17
  13. [Создание копии файла](#_bookmark20) 17
  14. [Внесение изменений в программу](#_bookmark21) 18
  15. [Запуск исполняющего файла](#_bookmark22) 19
  16. [Создание копии файла](#_bookmark23) 20
  17. [Внесение изменений в программу](#_bookmark24) 20
  18. [Запуск исполняющего файла](#_bookmark25) 22

**Список таблиц**

* 1. [Функциональные клавиши F1 — F10](#_bookmark3) . . . . . . . . . . . . . . . . 7

# Цель работы

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

# Задание

1. Начало работы с Midnight Commander.
2. Подключение внешнего файла in\_out.asm.
3. Задания для самостоятельной работы.

# Теоретическое введение

Midnight Commander (или просто mc) — это программа, которая позволяет про- сматривать структуру каталогов и выполнять основные операции по управлению файловой системой, т.е. mc является файловым менеджером. Для активации обо- лочки Midnight Commander достаточно ввести в командной строке mc и нажать клавишу Enter.

В табл. [3.1](#_bookmark3) приведено краткое описание функциональных клавиш F1 — F10 , к которым привязаны часто выполняемые операции в Midnight Commander.

Таблица 3.1: Функциональные клавиши F1 — F10

Функ- цио- наль- ные клави-

ши Выполняемое действие

F1 вызов контекстно-зависимой подсказки

F2 вызов меню, созданного пользователем

F3 просмотр файла, на который указывает подсветка в активной панели

F4 вызов встроенного редактора для файла, на который указывает подсветка в активной панели

F5 копирование файла или группы отмеченных файлов из каталога, отображаемого в активной панели, в каталог, отображаемый на второй

Функ- цио- наль- ные клави-

ши Выполняемое действие

панели

F6 перенос файла или группы отмеченных файлов из каталога, отображаемого

в актив- ной панели, в каталог, отобра- жае- мый на второй панели

F7 создание подкаталога в каталоге, отображаемом в активной панели

F8 удаление файла (подкаталога) или группы отмеченных файлов

F9 вызов основного меню программы

F10 выход из программы

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 дирек- тивы DB, DW, DD, DQ и DT, которые резервируют память и указывают, какие значения должны храниться в этой памяти. Директивы используются для объяв- ления 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. Инструкция языка ассемблера int предназначена для вызова прерывания с указанным номером. После вызова инструкции int 80h выполняется системный вызов какой-либо функции ядра Linux. При этом происходит передача управления ядру операционной систе- мы. Чтобы узнать, какую именно системную функцию нужно выполнить, ядро извлекает номер системного вызова из регистра eax. Поэтому перед вызовом прерывания необходимо поместить в этот регистр нужный номер. Кроме того, многим системным функциям требуется передавать какие-либо параметры. По принятым в ОС Linux правилам эти параметры помещаются в порядке следо- вания в остальные регистры процессора: ebx, ecx, edx. Если системная функция должна вернуть значение, то она помещает его в регистр eax. Простейший диалог с пользователем требует наличия двух функций — вывода текста на экран и ввода текста с клавиатуры. Простейший способ вывести строку на экран— использовать системный вызов write. Этот системный вызов имеет номер 4, поэтому перед вы- зовом инструкции int необходимо поместить значение 4 в регистр eax. Первым аргументом write, помещаемым в регистр ebx, задаётся дескриптор файла. Для вывода на экран в качестве дескриптора файла нужно указать 1 (это означает

«стандартный вывод», т. е. вывод на экран). Вторым аргументом задаётся адрес выводимой строки (помещаем его в регистр ecx, например, инструкцией mov ecx, msg). Строка может иметь любую длину. Последним аргументом (т.е. в реги- стре edx) должна задаваться максимальная длина выводимой строки. Для ввода строки с клавиатуры можно использовать аналогичный системный вызов read. Его аргументы –такие же, как у вызова write,только для «чтения» с клавиатуры

используется файловый дескриптор 0 (стандартный ввод). Системный вызов exit является обязательным в конце любой программы на языке ассемблер. Для обозначения конца программы перед вызовом инструкции int 80h необходимо поместить в регистр еах значение 1, а в регистр ebx код завершения 0.

# Выполнение лабораторной работы

## Начало работы с Midnight Commander

Открываю Midnight Commander. (рис. [4.18).](#_bookmark25)

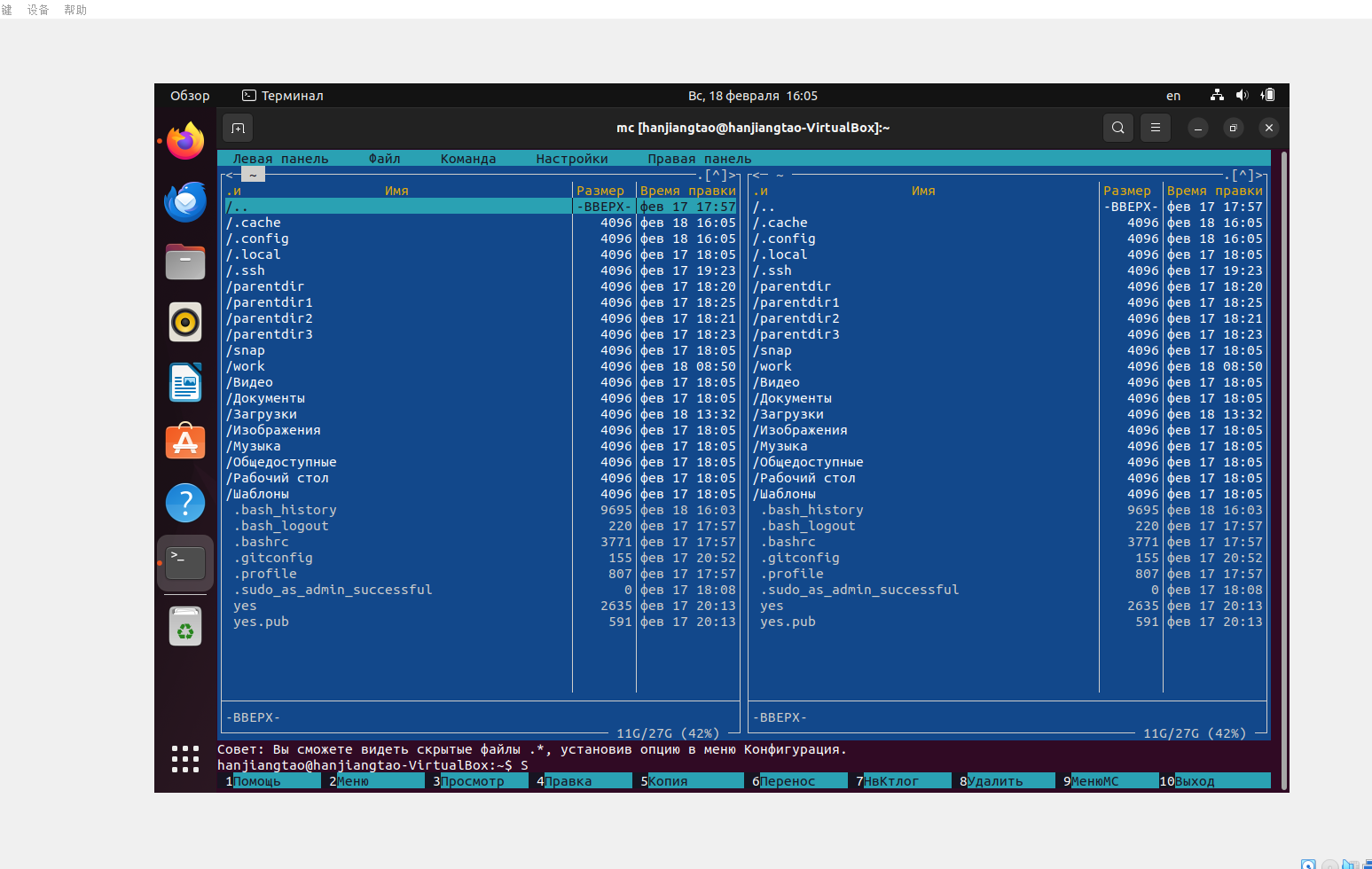


Рис. 4.1: Интерфейс Midnight Commander

Пользуясь клавишами вверх, вниз и Enter перехожу в каталог ~/work/arch-pc, созданный при выполнении лабораторной работы №4, с помощью функцио- нальной клавиши F7 создаю папку lab05 и перехожу в созданный каталог. (рис. [4.18).](#_bookmark25)

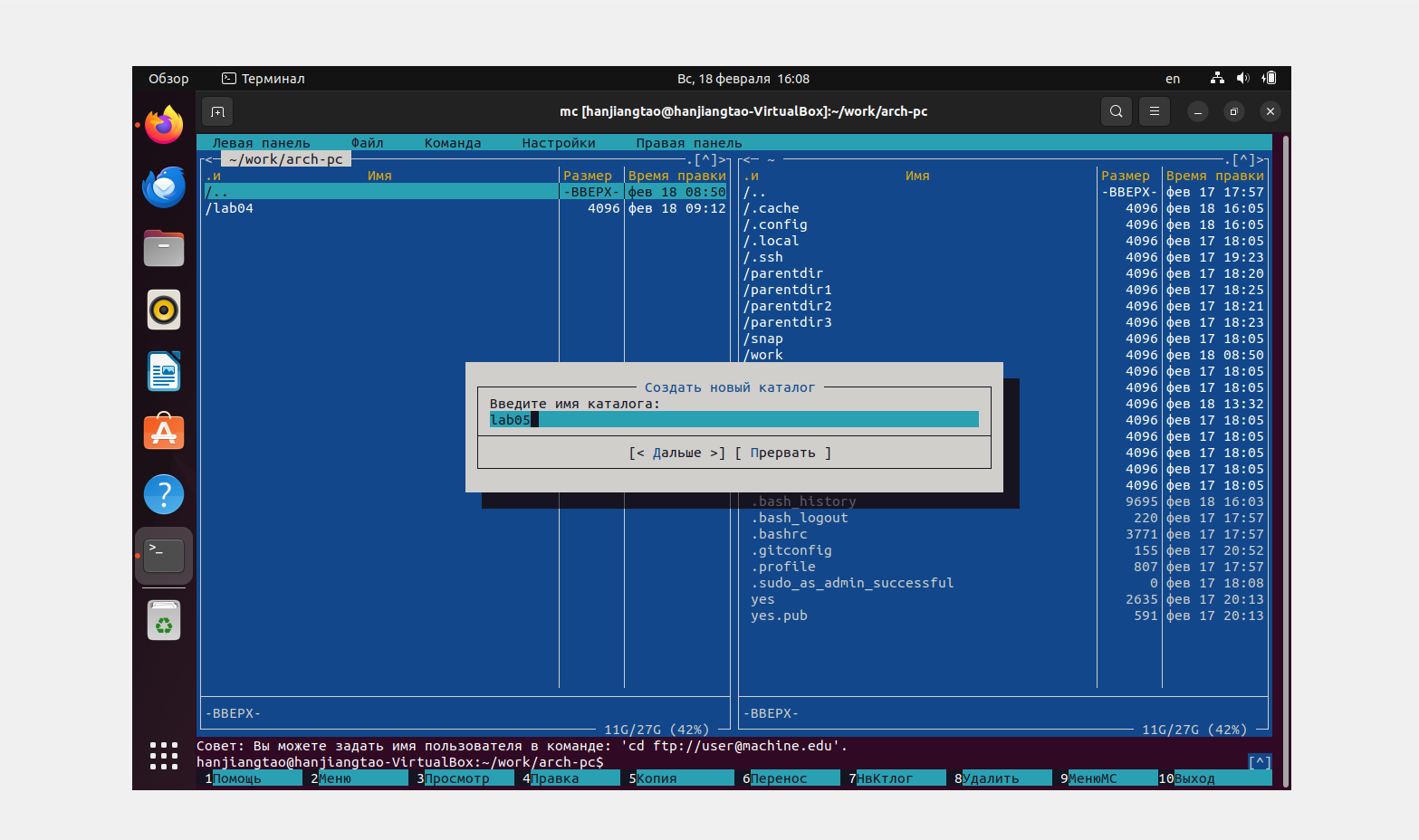


Рис. 4.2: Создание каталога в Midnight Commander

Пользуясь строкой ввода и командой touch создаю файл lab5-1.asm. (рис. [4.18).](#_bookmark25)

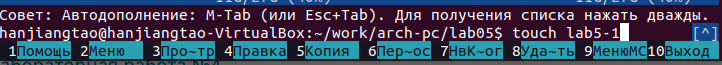


Рис. 4.3: Создание файла lab5-1.asm

С помощью функциональной клавиши F4 открываю файл lab5-1.asm для ре- дактирования во встроенном редакторе, ввожу текст программы из листинга 5.1, сохраняю изменения с помощью функциональной клавиши F2 и закрываю файл. (рис. [4.18).](#_bookmark25)

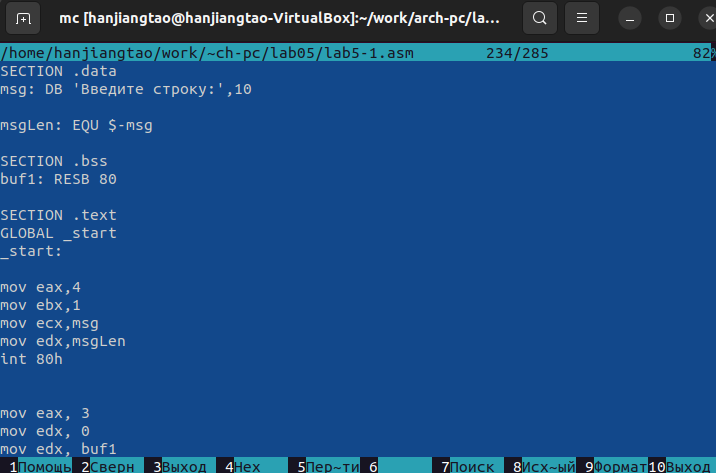


Рис. 4.4: Редактирование текстового файла в mc

С помощью функциональной клавиши F3 открываю файл lab5-1.asm для про- смотра и убеждаюсь, что файл содержит текст программы. (рис. [4.18).](#_bookmark25)

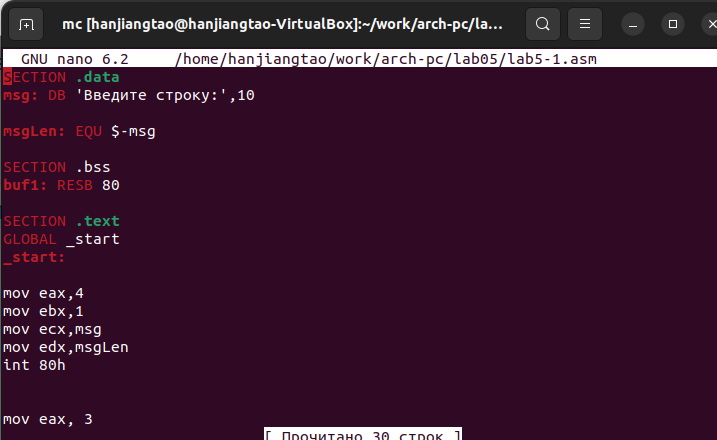


Рис. 4.5: Проверка наличия программы в файле

Компилирую текст программы lab5-1.asm в объектный файл, выполняю компо- новку объектного файла и запускаю получившийся исполняемый файл, в строку ввожу свои ФИО. (рис. [4.18).](#_bookmark25)

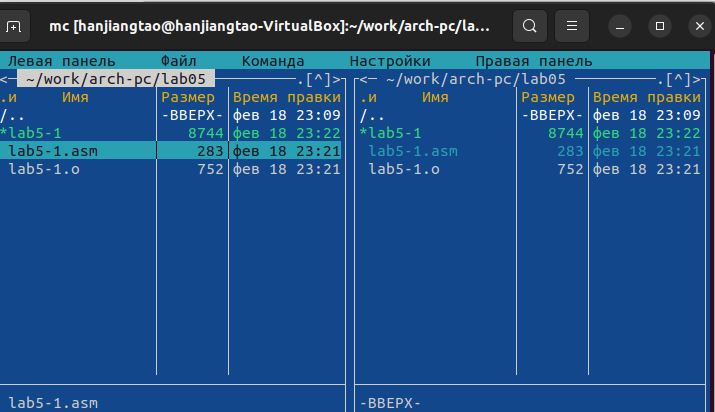


Рис. 4.6: Запуск исполняющего файла

## Подключение внешнего файла in\_out.asm

Скачиваю файл in\_out.asm со страницы курса в ТУИС. В первой панели mc открываю каталог с файлом lab5-1.asm, а во второй панели каталог со скачанным файлом in\_out.asm. Копирую файл in\_out.asm в каталог с файлом lab5-1.asm с помощью функциональной клавиши F5. (рис. [4.18).](#_bookmark25)

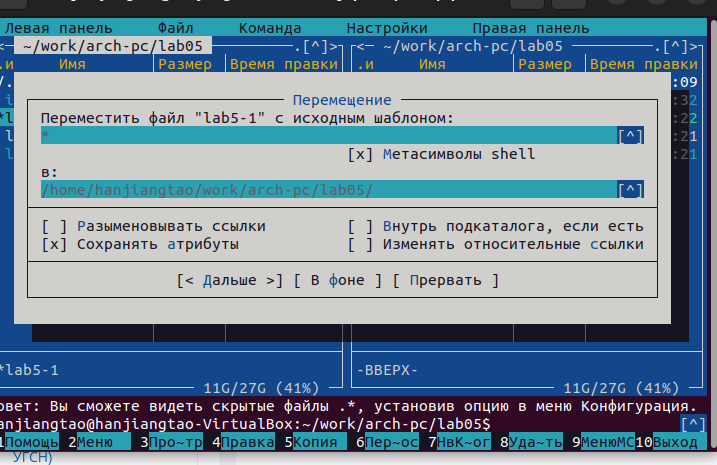


Рис. 4.7: Копирование файла in\_out.asm

С помощью функциональной клавиши F6 создаю копию файла lab5-1.asm с именем lab5-2.asm. (рис. [4.18).](#_bookmark25)

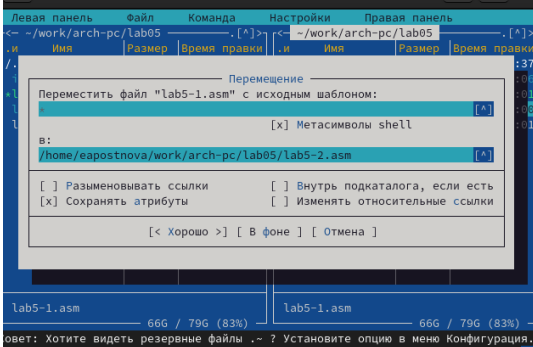


Рис. 4.8: Создание файла lab5-2.asm

Исправляю текст программы в файле lab5-2.asm с использование подпрограмм из внешнего файла in\_out.asm в соответствии с листингом 5.2. (рис. [4.18).](#_bookmark25)

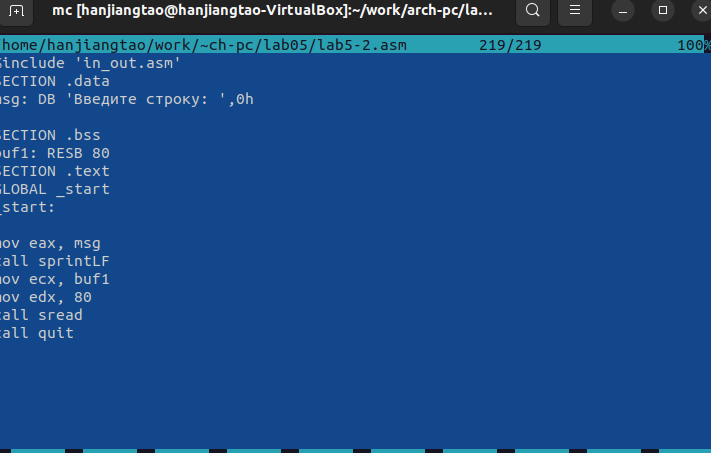


Рис. 4.9: Исправление текста программы

Создаю исполняемый файл и проверяю его работу. (рис. [4.18).](#_bookmark25)

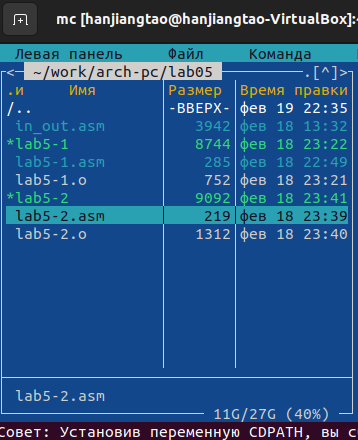


Рис. 4.10: Запуск программы исполняемого файла

Затем в файле lab5-2.asm заменяю подпрограмму sprintLF на sprint. (рис. [4.18).](#_bookmark25)



Рис. 4.11: Изменения в коде программы

Создаю исполняемый файл и проверяю его работу. (рис. [4.18).](#_bookmark25)

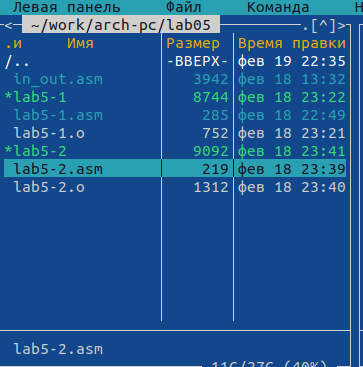


Рис. 4.12: Запуск программы измененного файла

##### Разница состоит в том, что в изначальной программе ввод текста про- исходил с новой строки, в измененной же программе перехода на новую строку нет.

## Задания для самостоятельной работы

* + 1. Создаю копию файла lab5-1.asm. (рис. [4.18).](#_bookmark25)

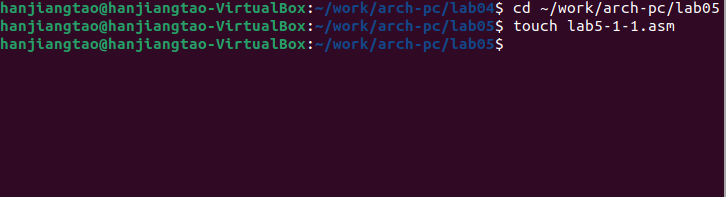


Рис. 4.13: Создание копии файла

Вношу изменения в программу так, чтобы она выводила введённую строку на экран. (рис. [4.18).](#_bookmark25)

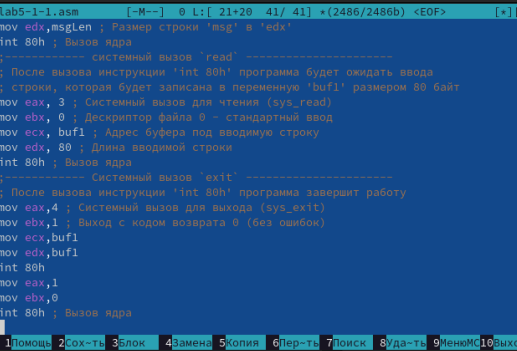


Рис. 4.14: Внесение изменений в программу

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. [4.18).](#_bookmark25)

15

Рис. 4.15: Запуск исполняющего файла

Программа работает.

* + 1. Создаю копию файла lab5-2.asm. (рис. [4.18).](#_bookmark25)

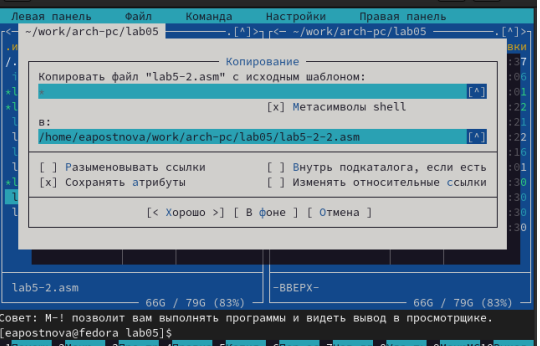


Рис. 4.16: Создание копии файла

Вношу изменения в программу с использование подпрограмм из внешнего файла in\_out.asm так, чтобы она выводила введённую строку на экран. (рис. [4.18).](#_bookmark25)

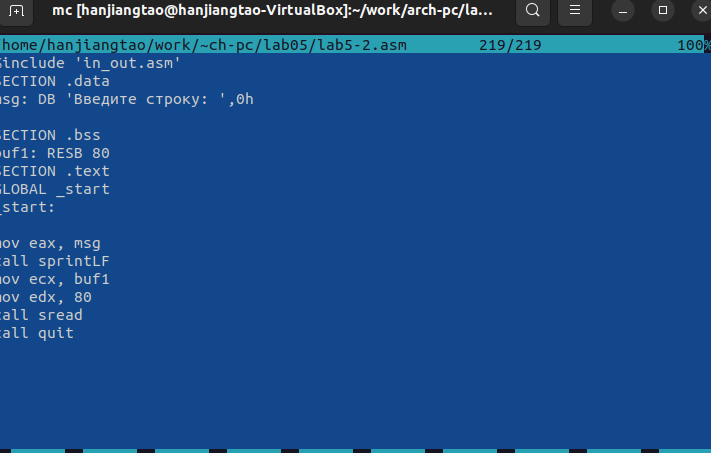


Рис. 4.17: Внесение изменений в программу

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. [4.18).](#_bookmark25)

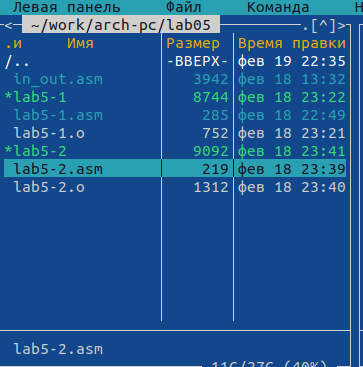


Рис. 4.18: Запуск исполняющего файла

Программа работает.

# Выводы

Благодаря данной лабораторной работе я приобрела навыки работы в Midnight Commander и освоила инструкции языка ассемблер mov и int, что поможет мне при выполнении последующих лабораторных работ.

# Список литературы

1. GDB: The GNU Project Debugger. — URL: ht[tps://w](http://www.gnu.org/software/gdb/)ww[.gnu.org/software/gdb/.](http://www.gnu.org/software/gdb/)
2. GNU Bash Manual. — 2016. — URL: ht[tps://w](http://www.gnu.org/software/bash/manual/)ww[.gnu.org/software/bash/manual/.](http://www.gnu.org/software/bash/manual/)
3. Midnight Commander Development Center. — 2021. — URL: https://midnight- commander. org/.
4. NASM Assembly Language Tutorials. — 2021. — URL: https://asmtutor.com/.
5. Newham C. Learning the bash Shell: Unix Shell Programming. — O’Reilly Media, 2005. — 354 с. — (In a Nutshell). — ISBN 0596009658. — URL:

[http://www.amazon.com/Learningbash-Shell-Programming-Nutshell/dp/0596009658.](http://www.amazon.com/Learningbash-Shell-Programming-Nutshell/dp/0596009658)

1. Robbins A. Bash Pocket Reference. — O’Reilly Media, 2016. — 156 с. — ISBN 978-1491941591.
2. The NASM documentation. — 2021. — URL: ht[tps://w](http://www.nasm.us/docs.php)ww[.nasm.us/docs.php.](http://www.nasm.us/docs.php)
3. Zarrelli G. Mastering Bash. — Packt Publishing, 2017. — 502 с. — ISBN 9781784396879.
4. Колдаев В. Д., Лупин С. А. Архитектура ЭВМ. — М. : Форум, 2018.
5. Куляс О. Л., Никитин К. А. Курс программирования на ASSEMBLER. — М. : Солон-Пресс, 2017.
6. Новожилов О. П. Архитектура ЭВМ и систем. — М. : Юрайт, 2016.
7. Расширенный ассемблер: NASM.— 2021.— URL: [https://w](http://www.opennet.ru/docs/RUS/nasm/)ww[.opennet.ru/docs/RUS/nasm/.](http://www.opennet.ru/docs/RUS/nasm/)
8. Робачевский А., Немнюгин С., Стесик О. Операционная система UNIX. — 2-е изд. — БХВПетербург, 2010. — 656 с. — ISBN 978-5-94157-538-1.
9. Столяров А. Программирование на языке ассемблера NASM для ОС Unix.— 2-

е изд.— М.: МАКС Пресс, 2011.— URL: [http://www.stolyarov.info/books/asm\_unix.](http://www.stolyarov.info/books/asm_unix)

1. Таненбаум Э. Архитектура компьютера. — 6-е изд. — СПб. : Питер, 2013. — 874 с. — (Классика Computer Science).
2. Таненбаум Э., Бос Х. Современные операционные системы. — 4-е изд. — СПб. : Питер,2015. — 1120 с. — (Классика Computer Science).