Лабораторная работа 5

Основы работы с Midnight Commander (mc). Структура программы на языке ассемблера NASM. Системные вызовы в ОС GNU Linux

Головина Мария Игоревна

Содержание

Список иллюстраций

Список таблиц

1 Цель работы

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

2 Задание

1. Открыть Midnight Commander. Создать папку lab05.
2. Пользуясь строкой ввода и командой touch создать файл lab5-1.asm.
3. С помощью функциональной клавиши F4 открыть файл lab5-1.asm для редактирования во встроенном редакторе. Ввести текст программы, сохранить изменения и закрыть файл.
4. Оттранслировать текст программы lab5-1.asm в объектный файл. Выполнить компоновку объектного файла и запустить получившийся исполняемый файл. Программа выводит строку ‘Введите строку:’ и ожидает ввода с клавиатуры. На запрос ввести свое ФИО.
5. Скачать 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. В одной из панелей mc открыть каталог с файлом lab5-1.asm. В другой панели каталог со скаченным файлом in_out.asm. Скопировать файл in_out.asm в каталог с файлом lab5-1.asm. С помощью функциональной клавиши F6 создать копию файла lab5-1.asm с именем lab5-2.asm.
6. Исправить текст программы в файле lab5-2.asm с использованием подпрограмм из внешнего файла in_out.asm (использовать подпрограммы sprintLF, sread и quit). Создать исполняемый файл и проверить его работу.
7. В файле lab5-2.asm заменить подпрограмму sprintLF на sprint. Создать исполняемый файл и проверить его работу.

Задание для самостоятельной работы

1. Создать копию файла lab5-1.asm. Внести 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.

Получить исполняемый файл и проверить его работу. На приглашение ввести строку ввели свою фамилию.

2. Создать копию файла lab5-2.asm. Исправить текст программы с использованием подпрограмм из 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.

Создать исполняемый файл и проверить его работу.

3 Теоретическое введение

Основы работы с Midnight Commander 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.

В Midnight Commander используются функциональные клавиши F1 — F10 , к которым привязаны часто выполняемые операции [3.1].

Таблица 3.1: Функциональные клавиши Midnight Commander

Функц.клавиши	Описание каталога
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, отображ.в активной панели, в каталог, отображ. на 2-й панели
F6	перенос файла или группы отмеченных файлов из каталога, отображ. в активной панели, в каталог, отображ. на 2-й панели
F7	создание подкаталога в каталоге, отображаемом в активной панели
F8	удаление файла (подкаталога) или группы отмеченных файлов

Функц.клавиши	Описание каталога
F9	вызов основного меню программы
F10	выход из программы

Структура программы на языке ассемблера NASM

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 (учетверённое 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.

Синтаксис директив определения данных следующий:

DB [,] [,]

Для объявления неинициированных данных в секции .bss используются директивы resb, resw, resd и другие, которые сообщают ассемблеру, что необходимо зарезервировать за данное количество ячеек памяти.

Элементы программирования

Описание инструкции mov

Инструкция языка ассемблера mov предназначена для дублирования данных источника в приёмнике. В общем виде эта инструкция записывается в виде

mov dst,src

Здесь операнд dst — приёмник, а src — источник.

В качестве операнда могут выступать регистры (register), ячейки памяти (memory) и непосредственные значения (const).

Необходимо переслать значение из одной ячейки памяти в другую нельзя, для этого необходимо использовать две инструкции

mov: mov eax, x

mov y, eax

Также необходимо учитывать то, что размер операндов приемника и источника должны совпадать. Использование следующих примеров приведет к ошибке:

mov al,1000h — ошибка, попытка записать 2-байтное число в 1-байтный регистр;

mov eax,cx — ошибка, размеры операндов не совпадают.

Описание инструкции int

Инструкция языка ассемблера int предназначена для вызова прерывания с указанным номером. В общем виде она записывается в виде

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). После вызова инструкции int 80h выполняется системный вызов какой-либо функции ядра Linux. При этом происходит передача управления ядру операционной системы. 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 систем-

ным функциям требуется передавать какие-либо параметры. По принятым в ОС Linux правилам эти параметры помещаются в порядке следования в остальные регистры процессора: ebx, ecx, edx. Если системная функция должна вернуть значение, то она помещает его в регистр eax.

Системные вызовы для обеспечения диалога с пользователем

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 вызовом 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 регистре 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 eax значение 1, а в регистр ebx код завершения 0.

4 Выполнение лабораторной работы

1. Открыли Midnight Commander. Создали папку lab05 (рис. 4.1 Создание папки lab 5).

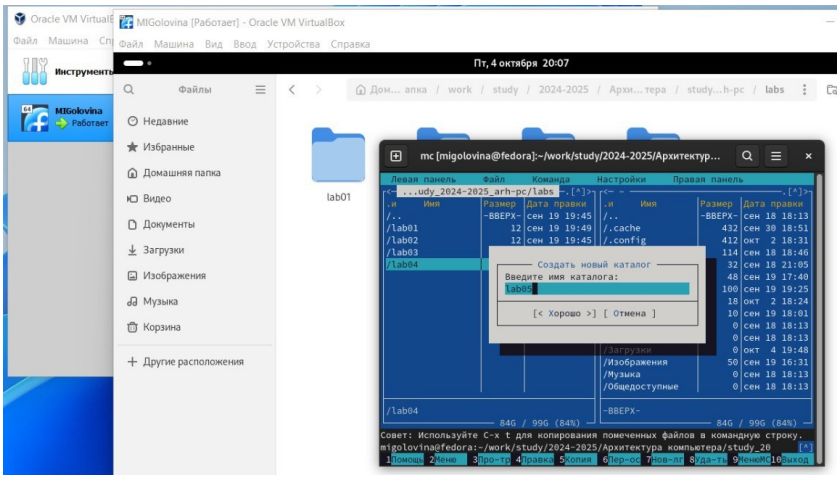

Рис. 4.1: Создание папки lab 5

2. Пользуясь строкой ввода и командой touch создали файл lab5-1.asm (рис. 4.2 Созданный файл lab5-1.asm).

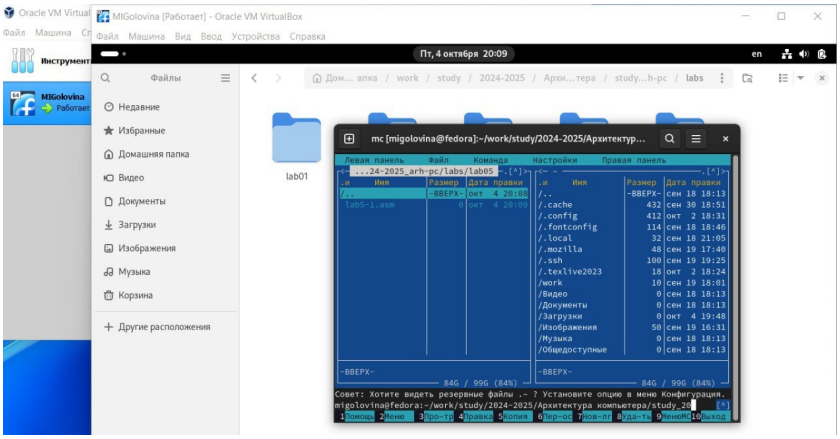

Рис. 4.2: Созданный файл lab5-1.asm

3. С помощью функциональной клавиши F4 открыли файл lab5-1.asm для редактирования во встроенном редакторе. Ввели текст программы, сохранили изменения и закрыли файл (рис. 4.3 Файл lab5-1.asm с изменениями).

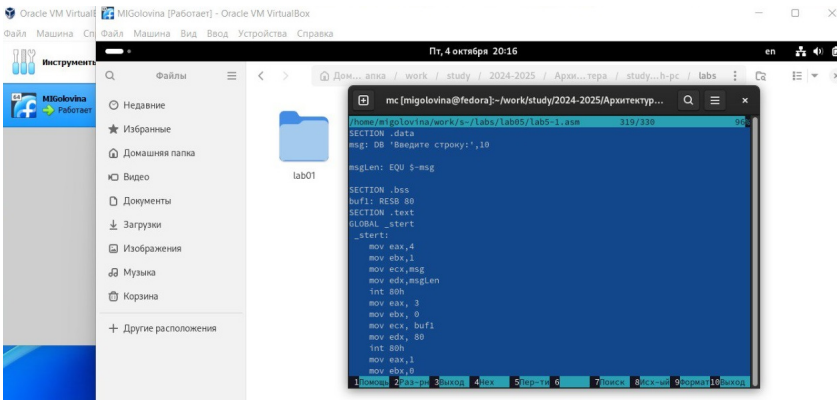

Рис. 4.3: Файл lab5-1.asm с изменениями

4. Оттранслировали текст программы lab5-1.asm в объектный файл. Выполнили компоновку объектного файла и запустили получившийся исполняемый файл. Программа выводит строку ‘Введите строку:’ и ожидает ввода с клавиатуры. На запрос ввели свое ФИО (рис. 4.4 Проверка работы программы).

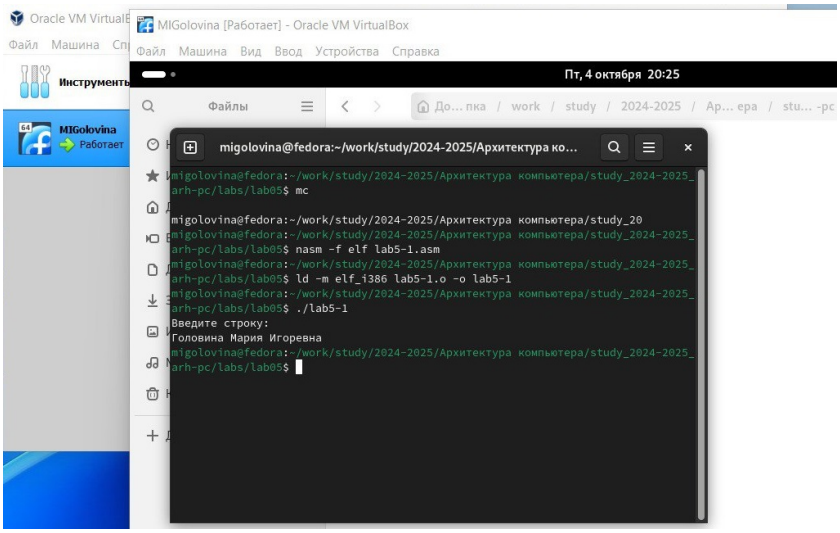

Рис. 4.4: Проверка работы программы

5.Скачали 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. В одной из панелей mc открыли каталог с файлом lab5-1.asm. В другой панели каталог со скаченным файлом in_out.asm. Скопировали файл in_out.asm в каталог с файлом lab5-1.asm. С помощью функциональной клавиши F6 создали копию файла lab5-1.asm с именем lab5-2.asm (рис. 4.5 Создание копии файла lab5-1.asm с именем lab5-2.asm).

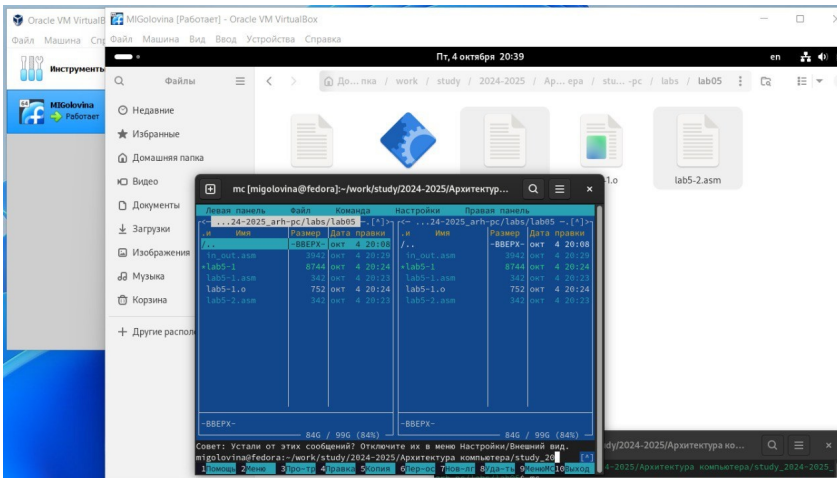

Рис. 4.5: Создание копии файла lab5-1.asm с именем lab5-2.asm

6. Исправили текст программы в файле lab5-2.asm с использованием подпрограмм из внешнего файла in_out.asm (использовали подпрограммы sprintLF, sread и quit). Создали исполняемый файл и проверили его работу (рис. 4.6 Результат работы исполняемого файла).

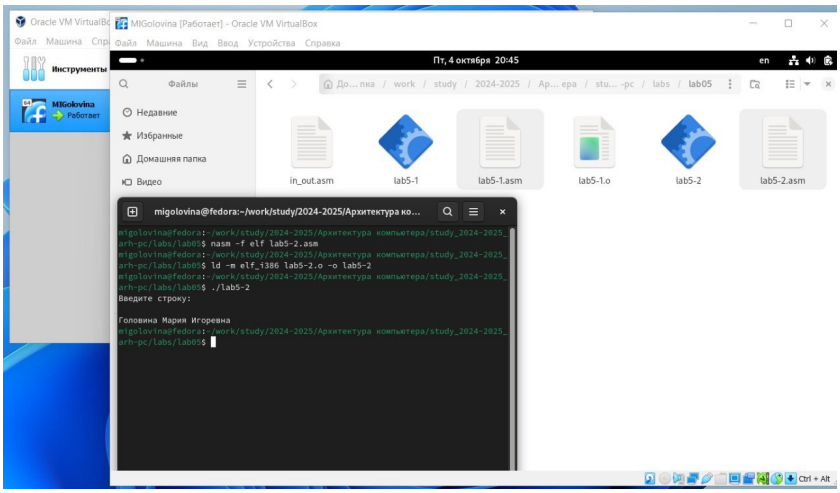

Рис. 4.6: Результат работы исполняемого файла

7. В файле lab5-2.asm заменили подпрограмму sprintLF на sprint. Создали исполняемый файл и проверили его работу (рис. 4.7 Результат работы измененного исполняемого файла).

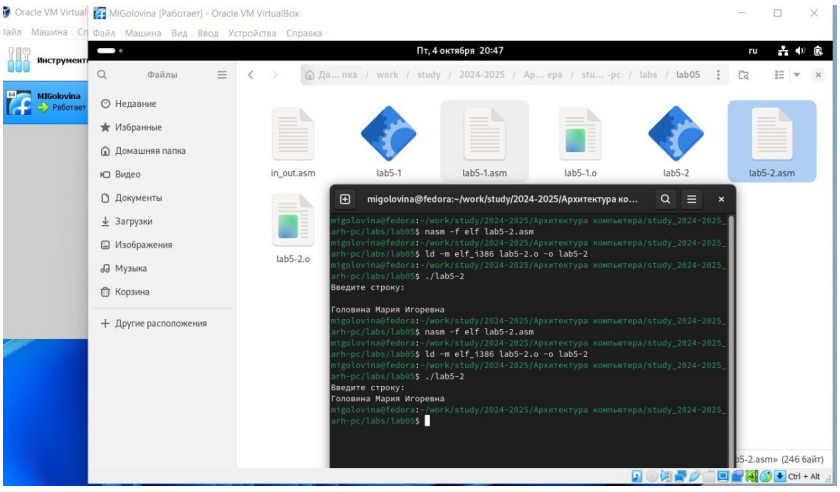

Рис. 4.7: Результат работы измененного исполняемого файла

Задание для самостоятельной работы

1. Создали копию файла lab5-1.asm. Внесли 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.

Получили исполняемый файл и проверили его работу. На приглашение ввести строку ввели свою фамилию (рис. 4.8 Результат работы исполняемого файла (без использования внешнего файла)).

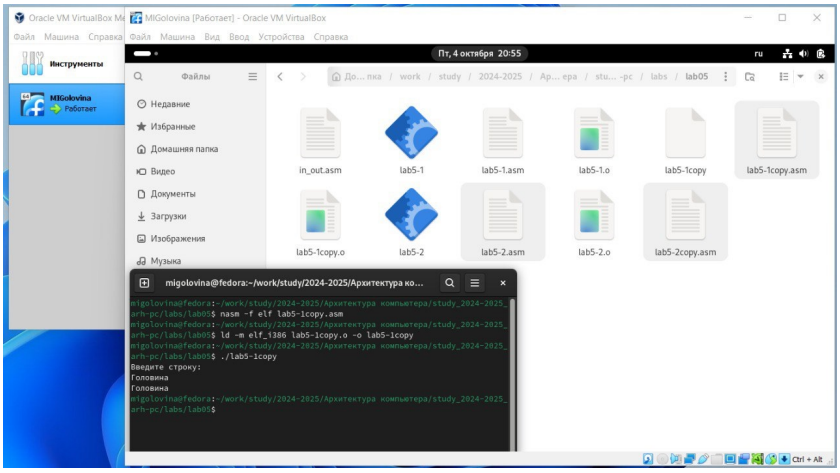

Рис. 4.8: Результат работы исполняемого файла (без использования внешнего файла)

Листинг написанной программы (без использования внешнего файла) (рис. 4.9 Листинг написанной программы (без использования внешнего файла)).

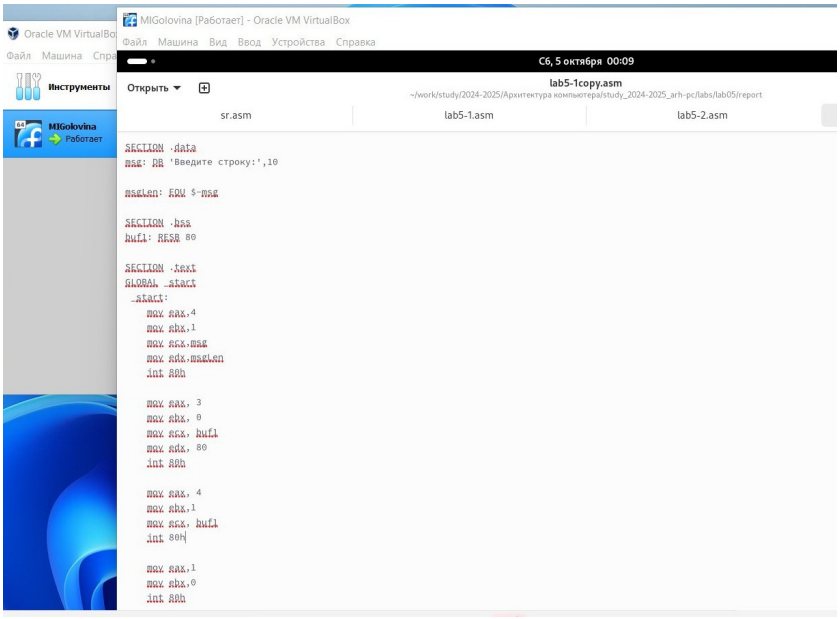

Рис. 4.9: Листинг написанной программы (без использования внешнего файла)

2. Создали копию файла lab5-2.asm. Исправили текст программы с использованием подпрограмм из 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.

Создали исполняемый файл и проверили его работу (рис. 4.10 Результат работы исполняемого файла (с использованием внешнего файла)).

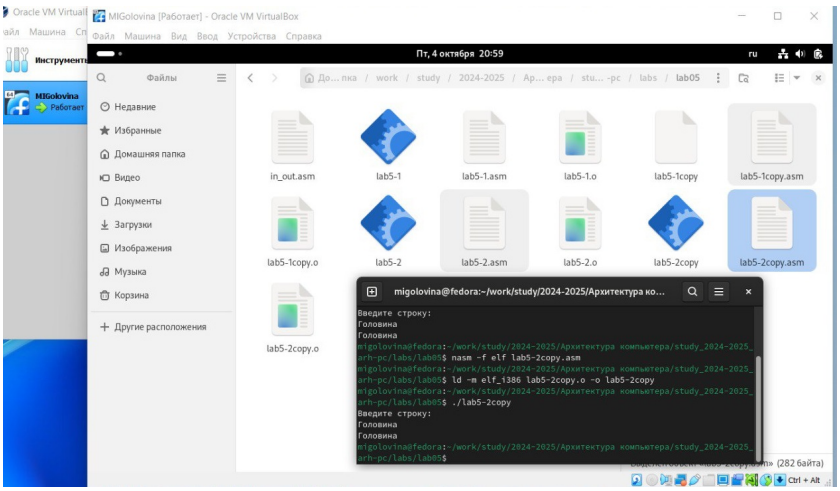

Рис. 4.10: Результат работы исполняемого файла (с использованием внешнего файла)

Листинг написанной программы (с использованием внешнего файла) (рис. 4.11 Листинг написанной программы (с использованием внешнего файла)).

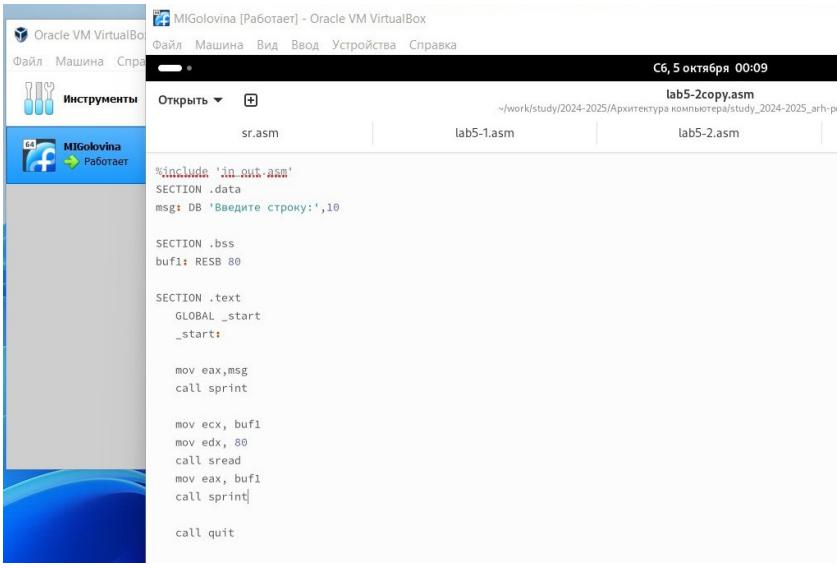

Рис. 4.11: Листинг написанной программы (с использованием внешнего файла)

5 Выводы

Приобрели практические навыки работы в Midnight Commander. Освоили инструкции языка ассемблера mov и int.

Список литературы

1. GDB: The GNU Project Debugger. — URL:https://www.gnu.org/software/gdb/.
2. GNU Bash Manual. — 2016. — URL: https://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 c. — (In a Nutshell). — ISBN 0596009658. — URL: http://www.amazon.com/Learningbash-Shell-Programming-Nutshell/dp/0596009658.
6. Robbins A. Bash Pocket Reference. — O'Reilly Media, 2016. — 156 c. — ISBN 978-1491941591.
7. The NASM documentation. — 2021. — URL: https://www.nasm.us/docs.php.
8. Zarrelli G. Mastering Bash. — Packt Publishing, 2017. — 502 c. — ISBN 9781784396879.
9. Колдаев В. Д., Лупин С. А. Архитектура ЭВМ. — М. : Форум, 2018.
10. Куляс О. Л., Никитин К. А. Курс программирования на ASSEMBLER. — М. : Солон-Пресс, 2017.
11. Новожилов О. П. Архитектура ЭВМ и систем. — М. : Юрайт, 2016.
12. Расширенный ассемблер: NASM. — 2021. — URL: https://www.opennet.ru/docs/RUS/nasm/.
13. Робачевский А., Немнюгин С., Стесик О. Операционная система UNIX. — 2-е изд. — БХВПетербург, 2010. — 656 с. — ISBN 978-5-94157-538-1.
14. 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.

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