Лабораторная работа 7

Команды безусловного и условного переходов в Nasm. Программирование ветвлений

Головина Мария Игоревна

Содержание

# 1 Цель работы

Изучение команд условного и безусловного переходов. Приобретение навыков написания программ с использованием переходов. Знакомство с назначением и структурой файла листинга.

# 2 Задание

1. Создать каталог для программ лабораторной работы № 6, перейти в него и создать файл lab7-1.asm.
2. Ввести в файл lab7-1.asm текст программы из листинга 1 методического указания. Создать исполняемый файл и запустить его. Посмотреть результаты работы. Написать вывод.
3. Изменить программу таким образом, чтобы она выводила сначала ‘Сообщение № 2’, потом ‘Сообщение № 1’ и завершала работу. Ввести текст программы в соответствии с листингом. Создать исполняемый файл и запустить его. Посмотреть результаты работы.
4. Создать файл lab7-2.asm. Ввести в него текст из листинга 3 методического указания. Создать исполняемый файл и запустить его. Проверить его работу при разных значениях В.
5. Создать файл листинга для программы из файла lab7-2.asm. Открыть его с помощью текстового редактора и изучили. Ознакомить с содержимым файла, описать любые три строки листинга.
6. Открыть файл с программой lab7-2.asm и в инструкции с двумя операндами удалили один операнд. Выполнить трансляцию с получением файла листинга. Написать вывод по заданию.

Задание для самостоятельной работы

1. Напишите программу нахождения наименьшей из 3 целочисленных переменных a, b, c. Значения переменных выбрать из таблицы в соответствии с вариантом, полученным при выполнении лабораторной работы № 6. Создайте исполняемый файл и проверьте его работу.
2. Напишите программу, которая для введенных с клавиатуры значений х и а вычисляет значение заданной функции f(x) и выводит результат вычислений. Вид функции f(x) выбрать из таблицы вариантов заданий в соответствии с вариантом, полученным при выполнении лабораторной работы № 6. Создайте исполняемый файл и проверьте его работу для значений x и a.

# 3 Теоретическое введение

Для реализации ветвлений в ассемблере используются так называемые команды передачи управления или команды перехода. Можно выделить 2 типа переходов: условный переход – выполнение или не выполнение перехода в определенную точку программы в зависимости от проверки условия; безусловный переход – выполнение передачи управления в определенную точку про граммы без каких-либо условий.

Команды безусловного перехода

Безусловный переход выполняется инструкцией jmp (от англ. jump – прыжок), которая включает в себя адрес перехода, куда следует передать управление:

jmp адрес перехода

Адрес перехода может быть либо меткой, либо адресом области памяти, в которую предварительно помещен указатель перехода. Кроме того, в качестве операнда можно использовать имя регистра, в таком случае переход будет осуществляться по адресу, хранящемуся в этом регистре.

В табл. [[1](#tbl:std-dir)] приведены типы операндов инструкции jmp.

Table 1: Типы операндов инструкции jmp

| Тип операнда | Описание |
| --- | --- |
| jmp label | переход на метку label |
| jmp [label] | переход по адресу в памяти, помеченному меткой label |
| jmp eax | переход по адресу из регистра eax |

Команды условного перехода

Как отмечалось выше, для условного перехода необходима проверка какого-либо условия. В ассемблере команды условного перехода вычисляют условие перехода анализируя флаги из регистра флагов.

Флаг – это бит, принимающий значение 1 («флаг установлен»), если выполнено некоторое условие, и значение 0 («флаг сброшен») в противном случае. Флаги работают независимо друг от друга, и лишь для удобства они помещены в единый регистр — регистр флагов, отражающий текущее состояние процессора. Флаги состояния (биты 0, 2, 4, 6, 7 и 11) отражают результат выполнения арифметических инструкций, таких как ADD, SUB, MUL, DIV.

Описание инструкции cmp

Инструкция cmp является одной из инструкций, которая позволяет сравнить операнды и выставляет флаги в зависимости от результата сравнения. Инструкция cmp является командой сравнения двух операндов и имеет такой же формат, как и команда вычитания:

cmp операнд\_1, операнд\_2

Команда cmp, так же как и команда вычитания, выполняет вычитание операнд\_2-операнд\_1, но результат вычитания никуда не записывается и единственным результатом команды сравнения является формирование флагов.

Описание команд условного перехода.

Команда условного перехода имеет вид

j мнемометка перехода label

Мнемоника перехода связана со значением анализируемых флагов или со способом формирования этих флагов.

Файл листинга и его структура

Листинг (в рамках понятийного аппарата NASM) — это один из выходных файлов, создаваемых транслятором. Он имеет текстовый вид и нужен при отладке программы, так как кроме строк самой программы он содержит дополнительную информацию.

Структура листинга:

номер строки — это номер строки файла листинга (нужно помнить, что номер строки в файле листинга может не соответствовать номеру строки в файле с исходным текстом программы);

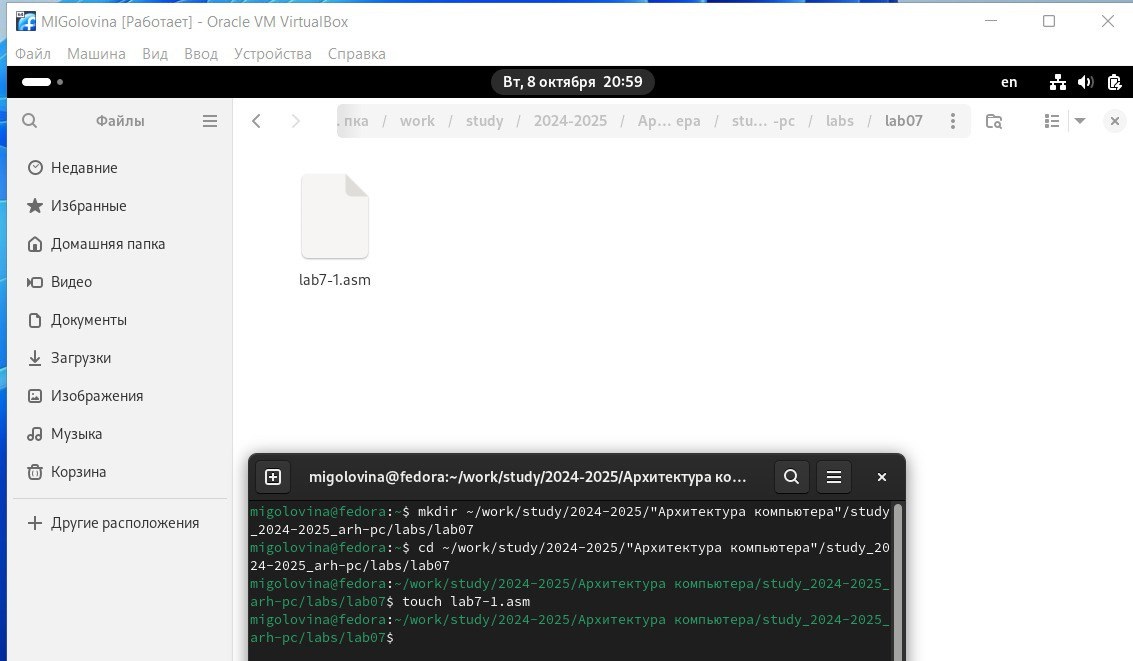
адрес — это смещение машинного кода от начала текущего сегмента;

машинный код представляет собой ассемблированную исходную строку в виде шестнадцатеричной последовательности. (например, инструкция int 80h начинается по смещению 00000020 в сегменте кода; далее идёт машинный код, в который ассемблируется инструкция, то есть инструкция int 80h ассемблируется в CD80 (в шестнадцатеричном представлении); CD80 — это инструкция на машинном языке, вызывающая прерывание ядра);

исходный текст программы — это просто строка исходной программы вместе с комментариями (некоторые строки на языке ассемблера, например, строки, содержащие только комментарии, не генерируют никакого машинного кода, и поля «смещение» и «исходный текст программы» в таких строках отсутствуют, однако номер строки им присваивается).

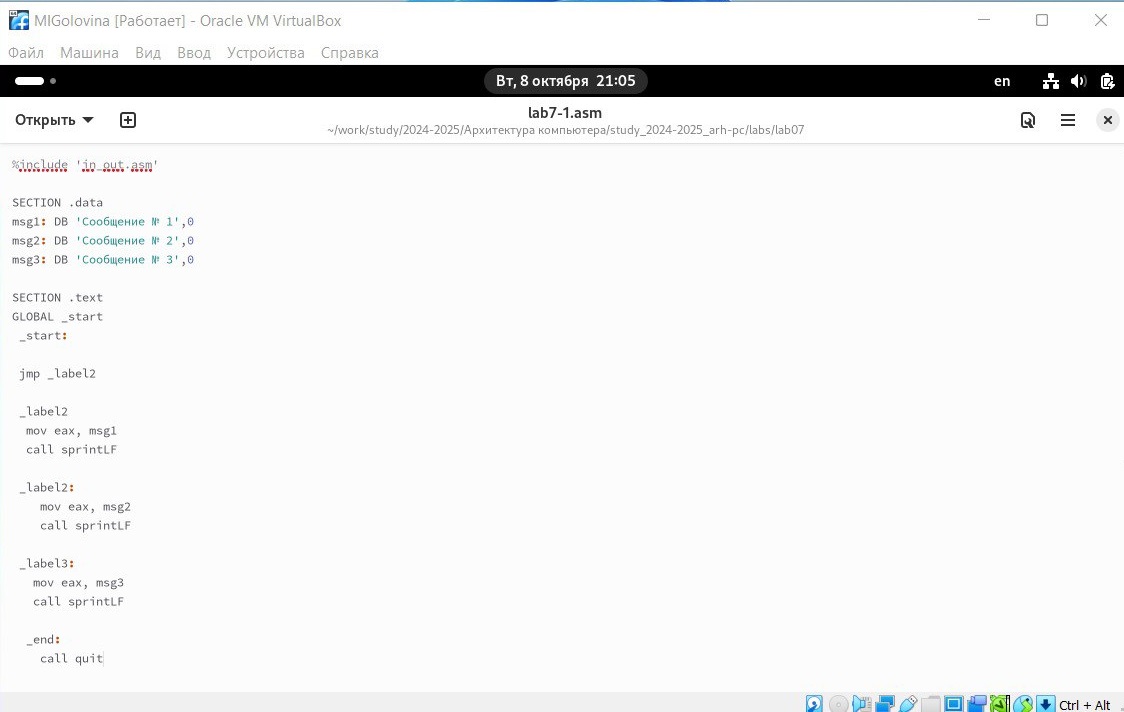
# 4 Выполнение лабораторной работы

1. Создали каталог для программ лабораторной работы № 7, перешли в него и создали файл lab7-1.asm (рис. 4.1 Создание каталога и файла для выполнения лабораторной работы).

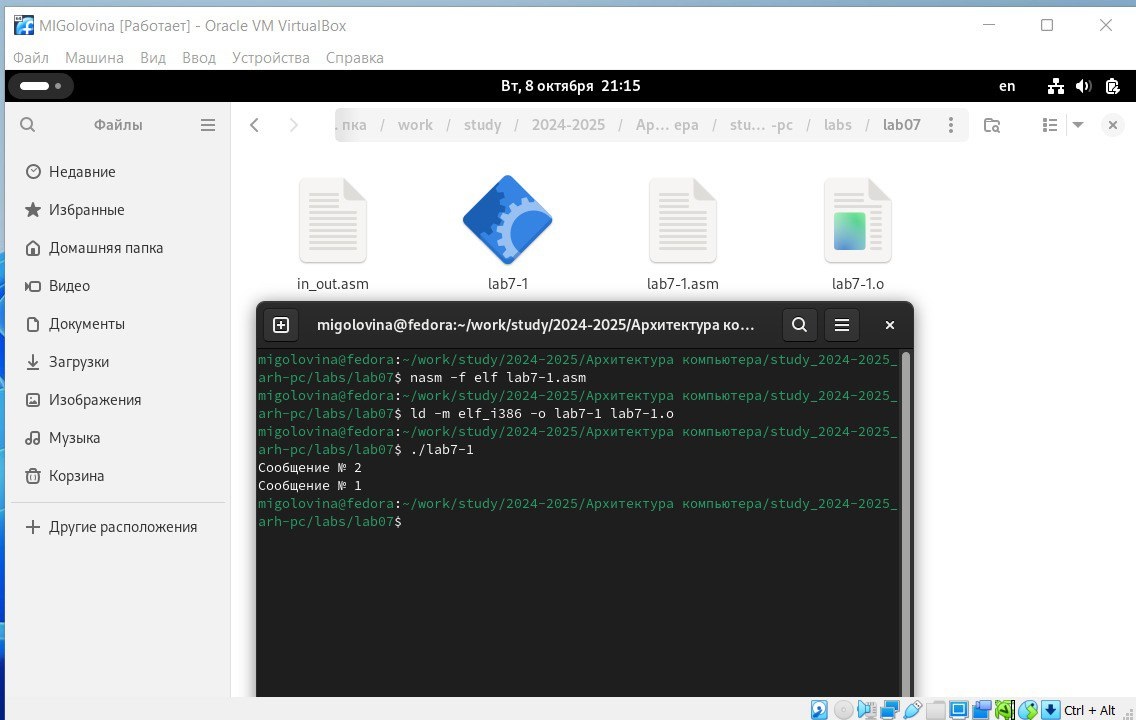


Создание каталога и файла для выполнения лабораторной работы

1. Ввели в файл lab7-1.asm текст программы из листинга 1 (рис. 4.2 Листинг 1) методического указания. Создали исполняемый файл и запустили его. Посмотрели результаты работы (рис. 4.3 Результаты работы программы из листинга 1 методического указания).



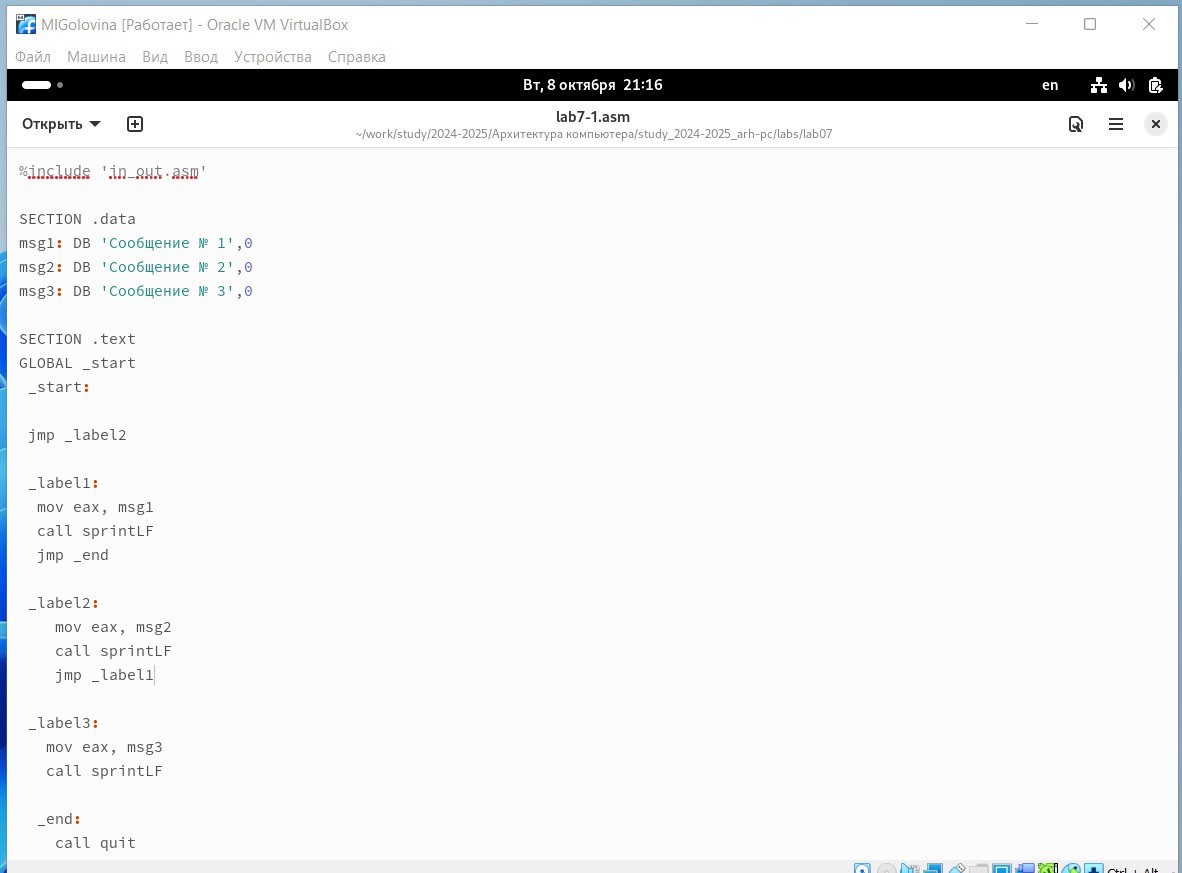
Листинг 1



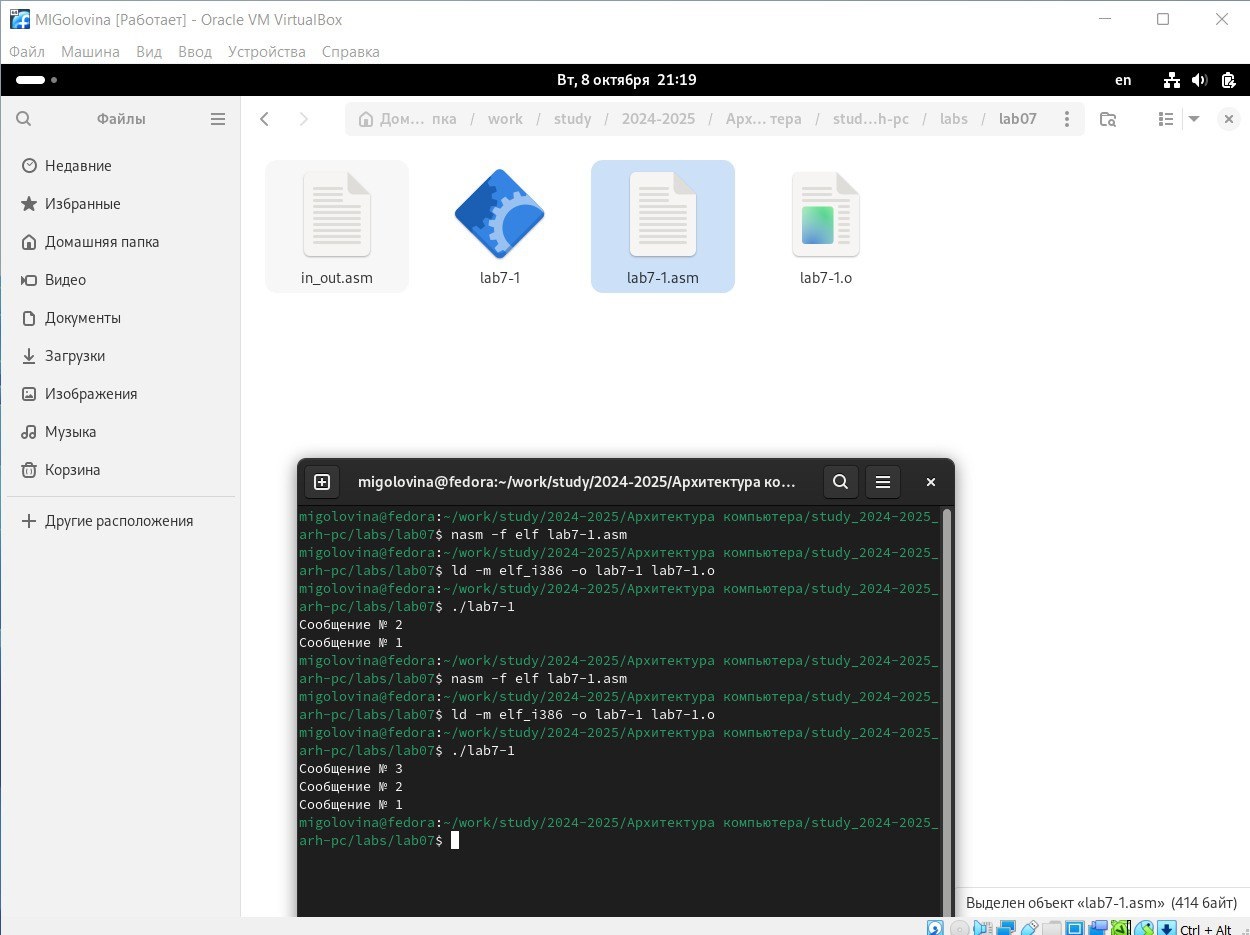
Результаты работы программы из листинга 1 методического указания

Таким образом, использование инструкции jmp \_label2 меняет порядок исполнения инструкций и позволяет выполнить инструкции, начиная с метки \_label2, пропустив вывод первого сообщения.

1. Изменили программу таким образом, чтобы она выводила сначала ‘Сообщение № 2’, потом ‘Сообщение № 1’ и завершала работу. Для этого в текст программы после вывода сообщения № 2 добавили инструкцию jmp с меткой \_label1 (т.е. переход к инструкциям вывода сообщения № 1) и после вывода сообщения № 1 добавили инструкцию jmp с меткой \_end (т.е. переход к инструкции call quit). Изменили текст программы в соответствии с листингом 2 (рис.4.4 Листинг 2). Создали исполняемый файл и запустили его. Посмотрели результаты работы (рис. 4.5 Результаты работы программы из листинга 2 методического указания).

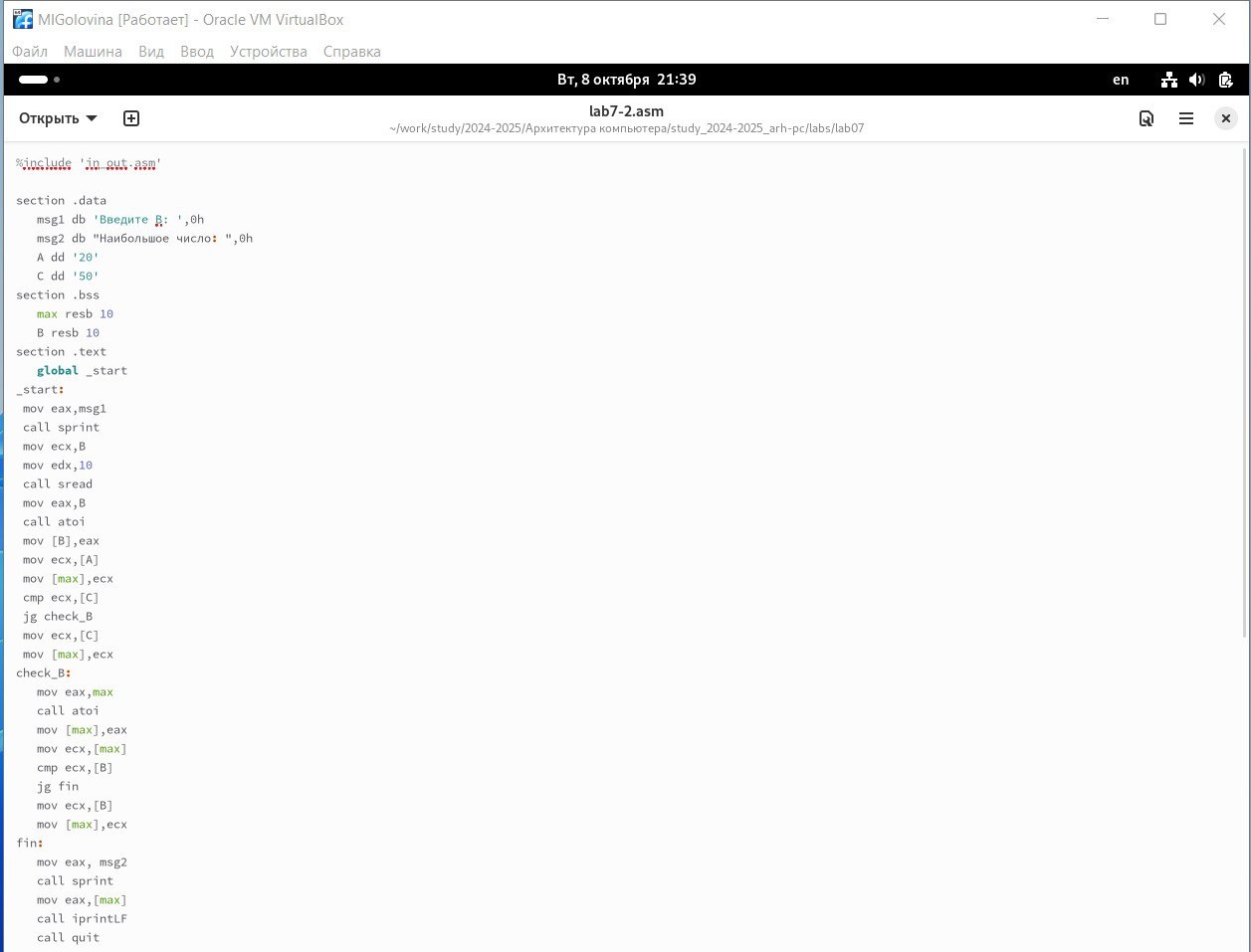


Листинг 2

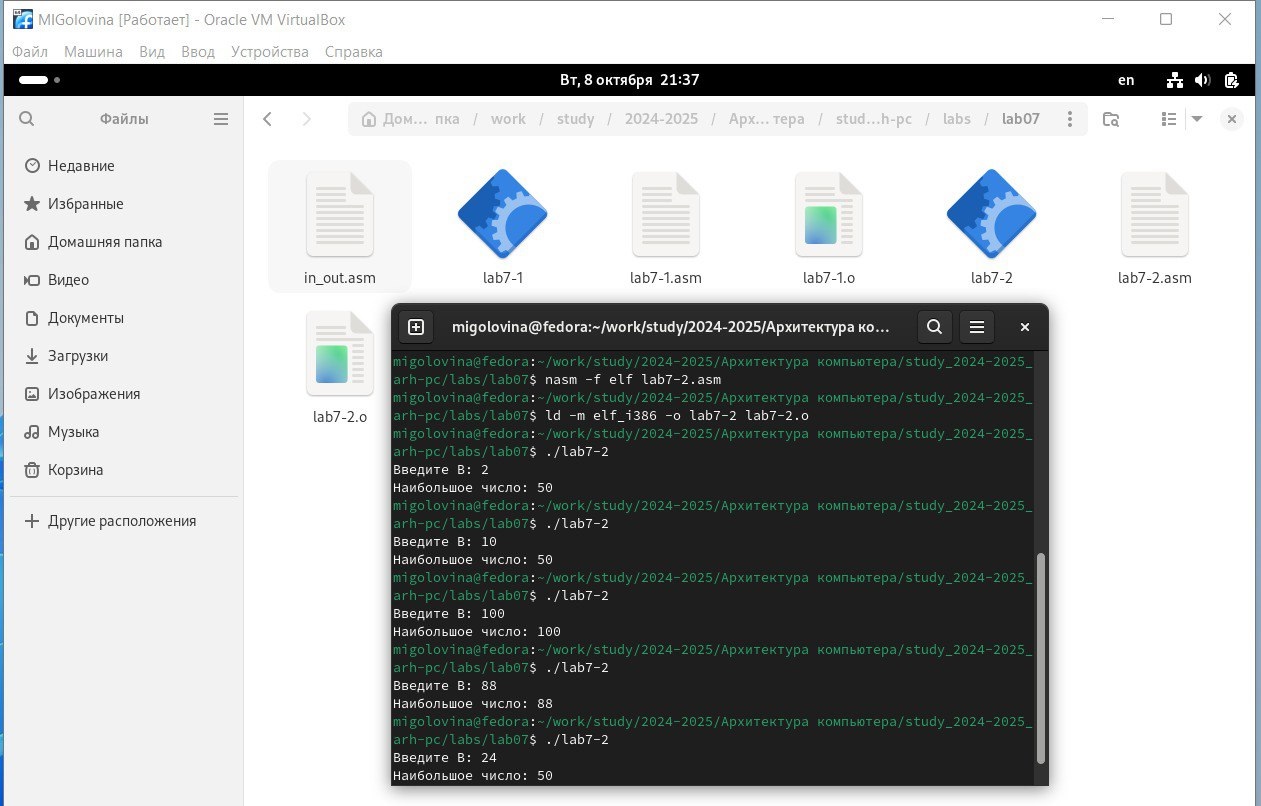


Результаты работы программы из листинга 2 методического указания

1. Создали файл lab7-2.asm. Ввели в него текст из листинга 3 методического указания (рис. 4.6 Листинг 3). Создали исполняемый файл и запустили его. Проверили его работу при разных значениях В (рис. 4.7 Результаты работы программы из листинга 3 методического указания).

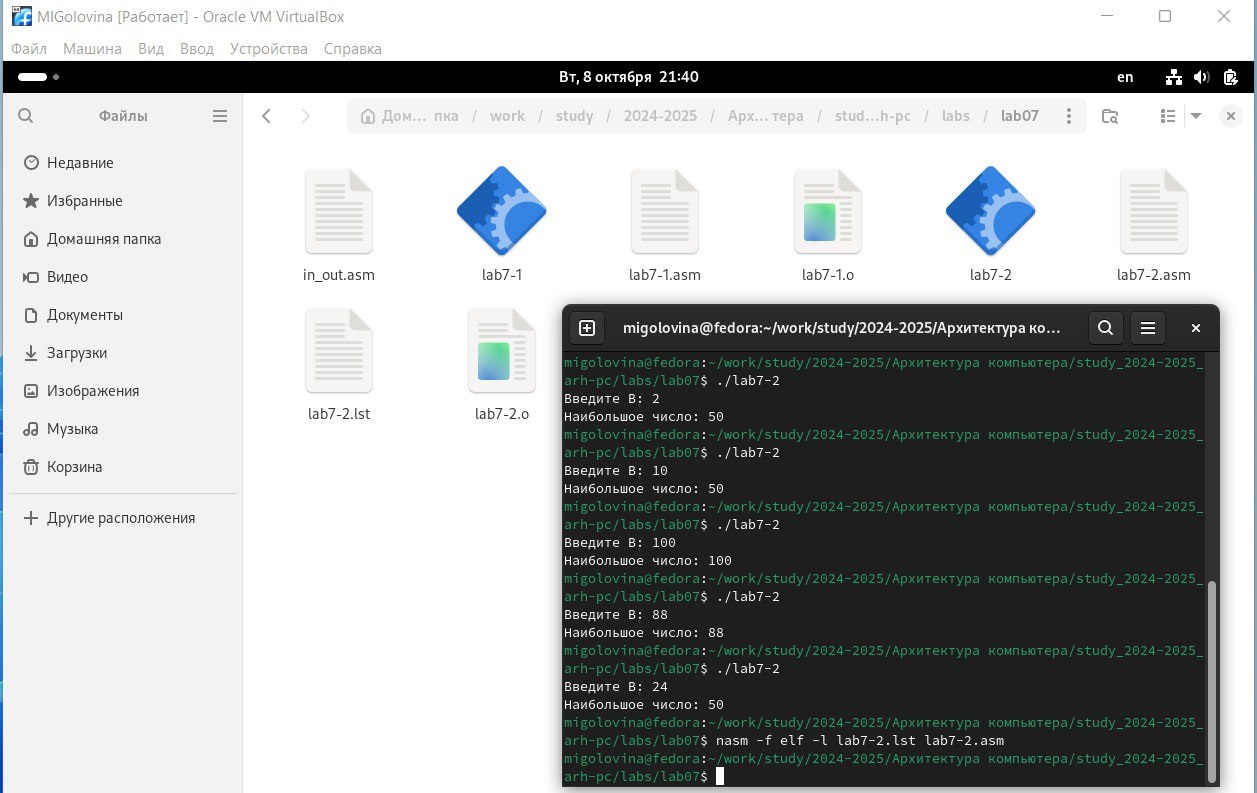


Листинг 3

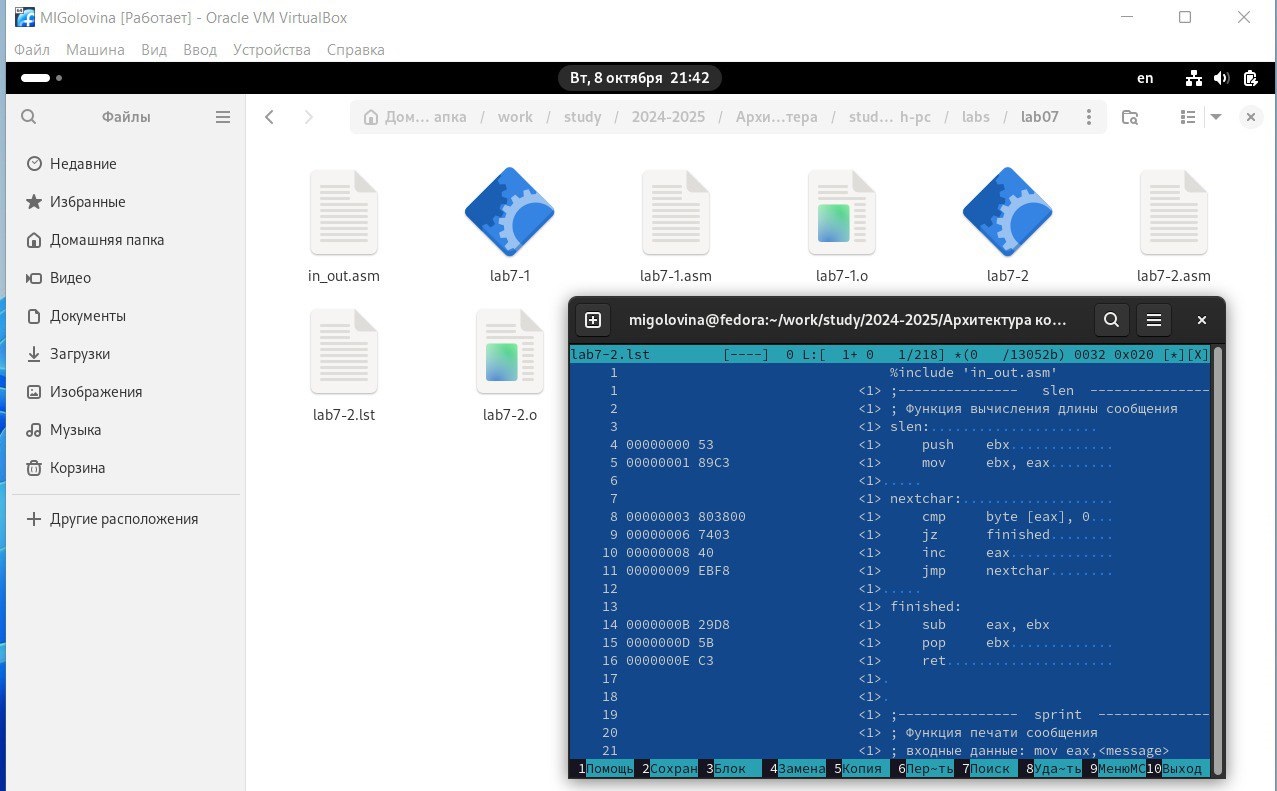


Результаты работы программы из листинга 3 методического указания

1. Создали файл листинга для программы из файла lab7-2.asm (рис. 4.8 Создание файл листинга для программы для программы из файла lab7-2.asm). Открыли его с помощью текстового редактора (рис. 4.9 Просмотр файла lab7-2.asm с помощью текстового редактора) и изучили.



Создание файл листинга для программы для программы из файла lab7-2.asm



Просмотр файла lab7-2.asm с помощью текстового редактора

Ознакомились с форматом и содержимым файла:

Эта строка находится на 23 месте, ее адрес “00000101”, Машинный код- В8[0A000000], а mov eax,B - исходный текст программы, означающий, что в регистр eax мы вносим значения переменной B (рис.4.10 Объяснение 1-й выбранной строки с листинга файла).

Объяснение 1-й выбранной строки с листинга файла

Объяснение 1-й выбранной строки с листинга файла

Эта строка находится на 37 месте, ее адрес “0000012F”, Машинный код - E868FFFFFF, а call atoi - исходный текст программы, означающий, что символ лежащий в строке выше переводится в число (рис.4.11 Объяснение 2-й выбранной строки с листинга файла).

Объяснение 2-й выбранной строки с листинга файла

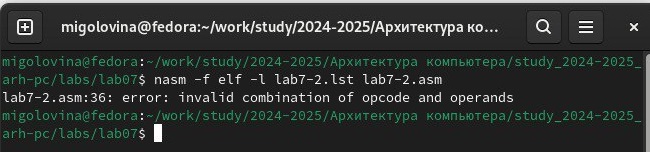
Объяснение 2-й выбранной строки с листинга файла

Эта строка находится на 48 месте, ее адрес “0000015D”, Машинный код - A1[00000000], а mov eax,[max] - исходный текст программы, означающий что число хранившееся в переменной max записывается в регистр eax (рис. 4.12 Объяснение 3-й выбранной строки с листинга файла).

Объяснение 3-й выбранной строки с листинга файла

Объяснение 3-й выбранной строки с листинга файла

1. Открыли файл с программой lab7-2.asm и в инструкции с двумя операндами удалили один операнд. Выполнили трансляцию с получением файла листинга (рис. 4.13 Создание файла без одного операнда).



Создание файла без одного операнда

В файле листинга показывает, где именно ошибка и с чем она связана (рис. 4.14 Файл листинга без одного операнда).

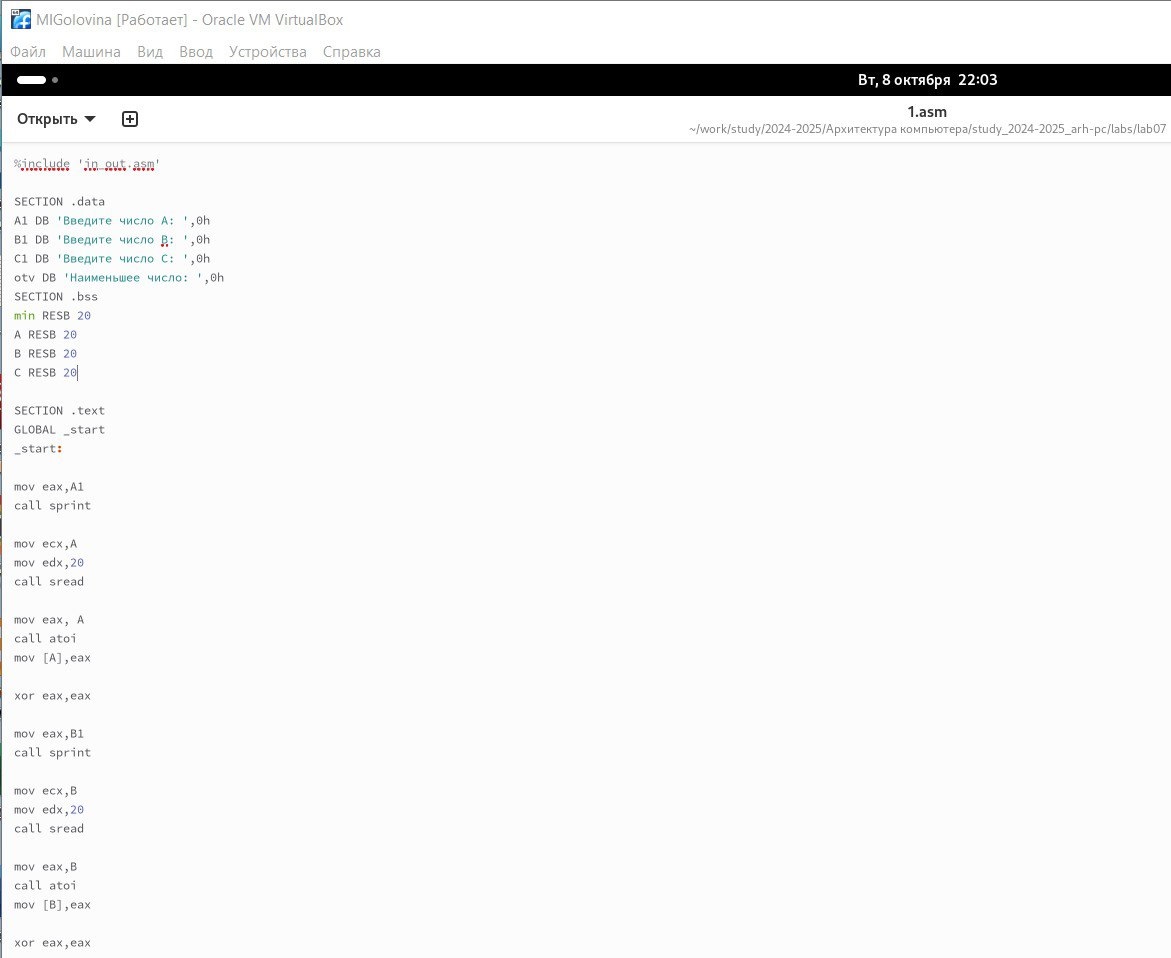
Файл листинга без одного операнда

Файл листинга без одного операнда

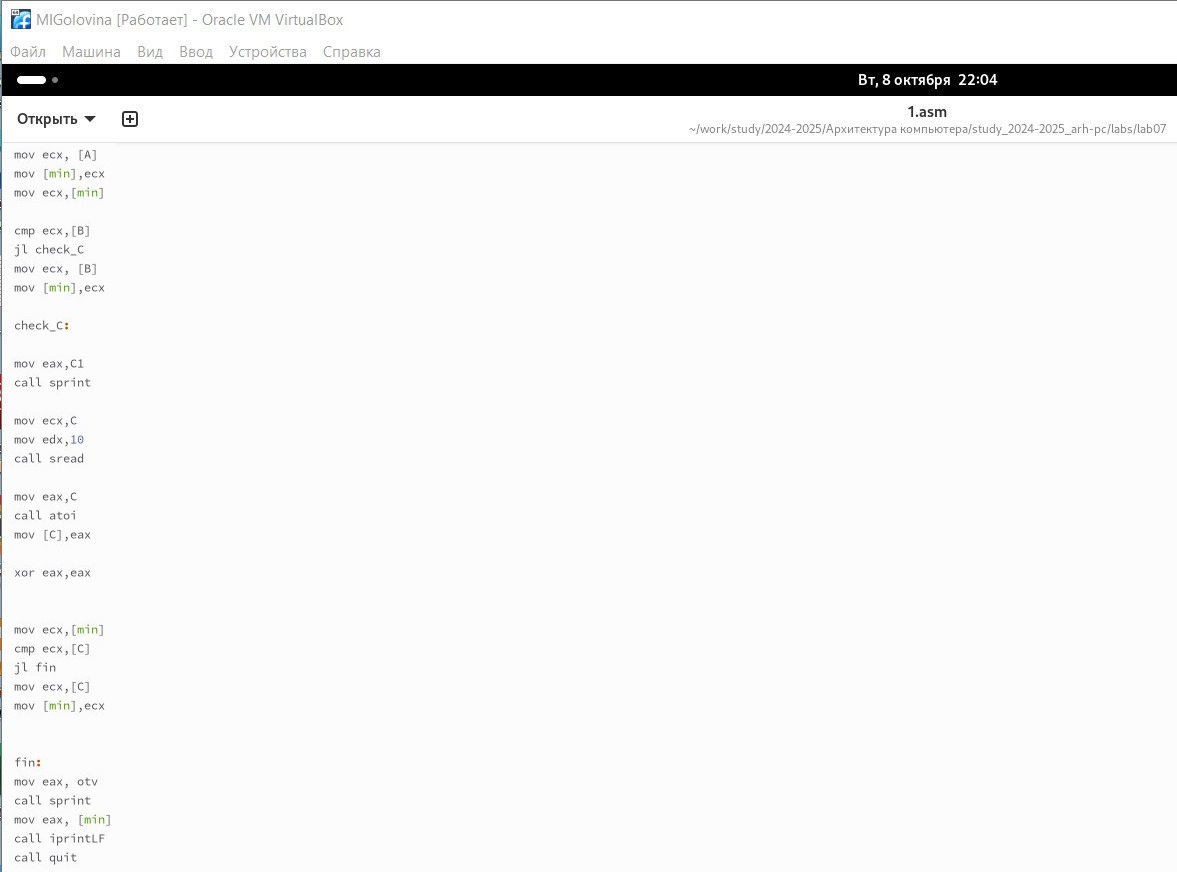
Задание для самостоятельной работы

1.Напишите программу нахождения наименьшей из 3 целочисленных переменных a, b, c. Значения переменных выбрать из таблицы в соответствии с вариантом, полученным при выполнении лабораторной работы № 6. Создайте исполняемый файл и проверьте его работу.

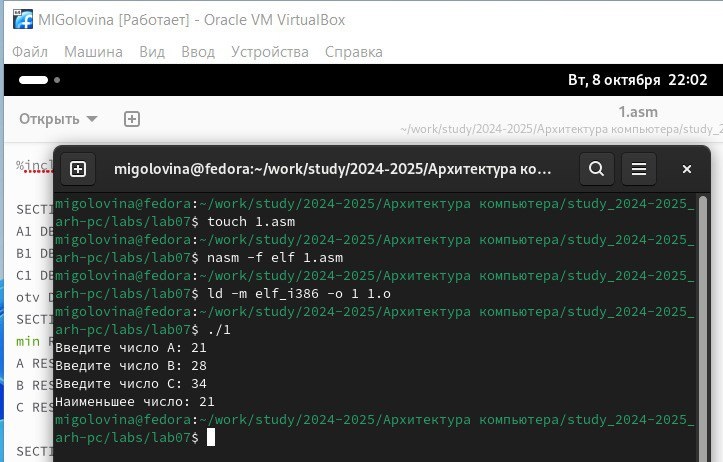
При выполнении лабораторной работы №6 у меня получился вариант 11, соответственно для варианта №11 значение переменных: a=21, b=28, c=34.



Листинг самостоятельного задания №1



Продолжение листинга самостоятельного задания №1



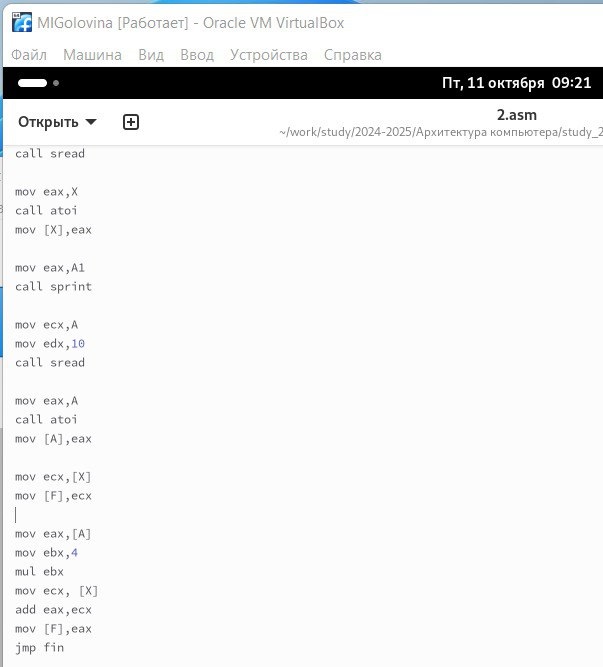
Результаты работы программы по самостоятельному заданию №1

1. Напишите программу, которая для введенных с клавиатуры значений х и а вычисляет значение заданной функции f(x) и выводит результат вычислений. Вид функции f(x) выбрать из таблицы вариантов заданий в соответствии с вариантом, полученным при выполнении лабораторной работы № 6. Создайте исполняемый файл и проверьте его работу для значений x и a.

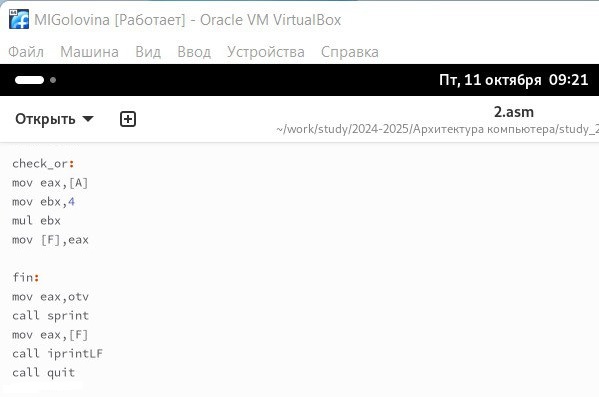
При выполнении лабораторной работы №6 у меня получился вариант 11, соответственно для варианта №11 была написана программа.



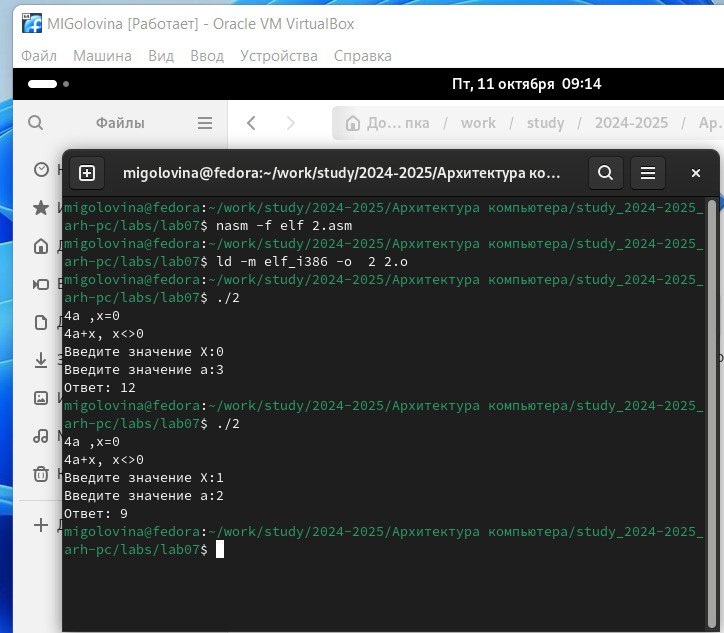
Листинг самостоятельного задания №2



Продолжение листинга самостоятельного задания №2



Продолжение листинга самостоятельного задания №2



Результаты работы программы

# 5 Выводы

Изучили команды условного и безусловного переходов. Приобрели навыки написания программ с использованием переходов. Познакомились с назначением и структурой файла листинга.

# 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 с. — (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 с. — ISBN 978-1491941591.
7. The NASM documentation. — 2021. — URL: https://www.nasm.us/docs.php.
8. Zarrelli G. Mastering Bash. — Packt Publishing, 2017. — 502 с. — 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).