



МИНОБРНАУКИ РОССИИ

Федеральное государственное бюджетное образовательное учреждение  
высшего образования

«МИРЭА - Российский технологический университет»

РТУ МИРЭА

---

Институт Информационных Технологий  
Кафедра Вычислительной Техники (ВТ)

**ОТЧЁТ ПО ПРАКТИЧЕСКИМ РАБОТАМ**

по дисциплине

«Архитектура вычислительных машин и систем»

Выполнил студент группы  
ИКБО-42-23

Туляшева А.Т.

Принял преподаватель кафедры ВТ

Рыжова А.А.

Отчет выполнен

«\_\_»\_\_\_\_\_2024 г.

«Зачтено»

«\_\_»\_\_\_\_\_2024 г.

Москва 2024 г.

## **Оглавление**

<b>ПРАКТИЧЕСКАЯ РАБОТА №1 (Q11+12) .....</b>	<b>3</b>
<b>ПРАКТИЧЕСКАЯ РАБОТА №2 (Q21+22) .....</b>	<b>8</b>
<b>ПРАКТИЧЕСКАЯ РАБОТА №3 (Q3).....</b>	<b>11</b>
<b>ПРАКТИЧЕСКАЯ РАБОТА №4 (VM1) .....</b>	<b>14</b>
<b>ПРАКТИЧЕСКАЯ РАБОТА №5 (VM2) .....</b>	<b>17</b>
<b>ПРАКТИЧЕСКАЯ РАБОТА №6 (VM3) .....</b>	<b>25</b>
<b>ПРАКТИЧЕСКАЯ РАБОТА №7 (CPU580) .....</b>	<b>31</b>

# ПРАКТИЧЕСКАЯ РАБОТА №1 (Q11+12)

## Теоретическое введение

Математической основой цифровой электроники и вычислительной техники является алгебра логики или булева алгебра (по имени английского математика Джона Буля).

В булевой алгебре независимые переменные или аргументы (X) принимают только два значения: «0» или «1». Зависимые переменные или функции (Y) также могут принимать только два значения: «0» или «1».

Функция алгебры логики (ФАЛ) представляется в виде:

$$Y = F(X_1 ; X_2 ; X_3 \dots X_n)$$

Данная форма задания ФАЛ называется алгебраической.

Схемы, реализующие логические функции, называются логическими элементами. Основные логические элементы имеют, как правило, один выход (Y) и несколько входов, число которых равно числу аргументов.

На электрических схемах логические элементы рисуют в виде прямоугольников с выводами для входных (слева) и выходных (справа) переменных. В середине прямоугольника изображается символ, обозначающий функциональное назначение элемента.

Язык описания аппаратуры AHDL разработан фирмой Altera и предназначен для описания комбинационных и последовательностных логических устройств, групповых операций, цифровых автоматов (state machine) и таблиц истинности с учетом архитектурных особенностей ПЛИС фирмы Altera. Он полностью интегрируется с системой автоматизированного проектирования ПЛИС QUARTUS II. Файлы описания аппаратуры, написанные на языке AHDL, имеют расширение \*.TDF (Text design file). Для создания TDF-файла можно использовать как текстовый редактор системы QUARTUS II, так и любой другой. Проект, выполненный в виде TDF-файла, компилируется, отлаживается и используется для формирования файла программирования или загрузки ПЛИС фирмы Altera.

Операторы и элементы языка AHDL являются достаточно мощным и универсальным средством описания алгоритмов функционирования цифровых

устройств, удобным в использовании. Язык описания аппаратуры AHDL дает возможность создавать иерархические проекты в рамках одного этого языка или же в иерархическом проекте использовать как TDF-файлы, разработанные на языке AHDL, так и другие типы файлов.

При распределении ресурсов устройств разработчик может пользоваться командами текстового редактора или операторами языка AHDL для того, чтобы сделать назначения ресурсов и устройств. Кроме того, разработчик может только проверить синтаксис или выполнить полную компиляцию для отладки и запуска проекта. Любые ошибки автоматически обнаруживаются обработчиком сообщений и высвечиваются в окне текстового редактора.

Зарезервированные ключевые слова используются для следующих целей:

- для обозначения начала, конца и переходов в объявлениях языка AHDL;
- для обозначения предопределенных констант, т.е. GND и VCC.

Ключевые слова можно использовать, как символические имена, только если они заключены в символы одинарных кавычек (''). Их можно также использовать в комментариях.

### **Вариант 1**

1. Нарисовать синтезированную схему в графическом редакторе САПР QUARTUS II, описанного уравнением в алгебраической форме варианта 1:

$$Y = \overline{AB} + CD + ABD$$

2. Произвести симуляцию работы схемы. Зарисовать ее временную диаграмму.

3. Сделать описание заданной электрической схемы при помощи текстового редактора САПР QUARTUS II.

4. Произвести симуляцию работы кода. Зарисовать ее временную диаграмму.

5. Сравнить результаты, полученные в ходе выполнения лабораторной работы с таблицей истинности.

### **Таблица истинности выражения**

Построим таблицу истинности выражения  $Y = \overline{AB} + CD + ABD$

Таблица 1 – Таблица истинности логического выражения

A	B	C	D	Y
0	0	0	0	1
0	0	0	1	1
0	0	1	0	1
0	0	1	1	1
0	1	0	0	1
0	1	0	1	1
0	1	1	0	1
0	1	1	1	1
1	0	0	0	1
1	0	0	1	1
1	0	1	0	1
1	0	1	1	1
1	1	0	0	0
1	1	0	1	1
1	1	1	0	0
1	1	1	1	1

### Построение схемы

Построим схему для нашего логического выражения.

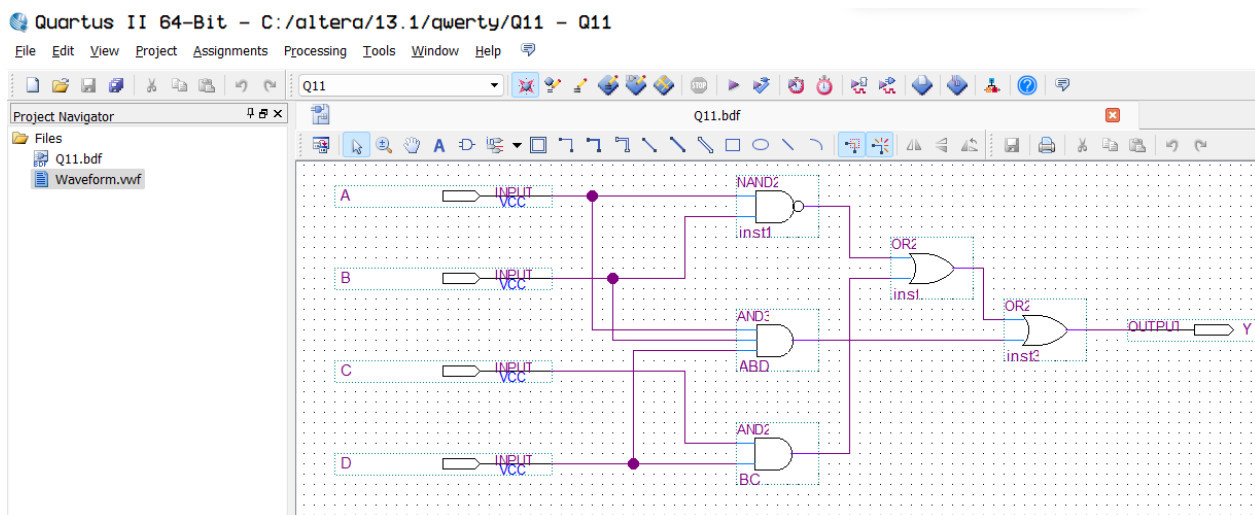


Рисунок 1 – Схема логического выражения

### Временная диаграмма схемы

Произведем симуляцию работы схемы и нарисует ее временную диаграмму.

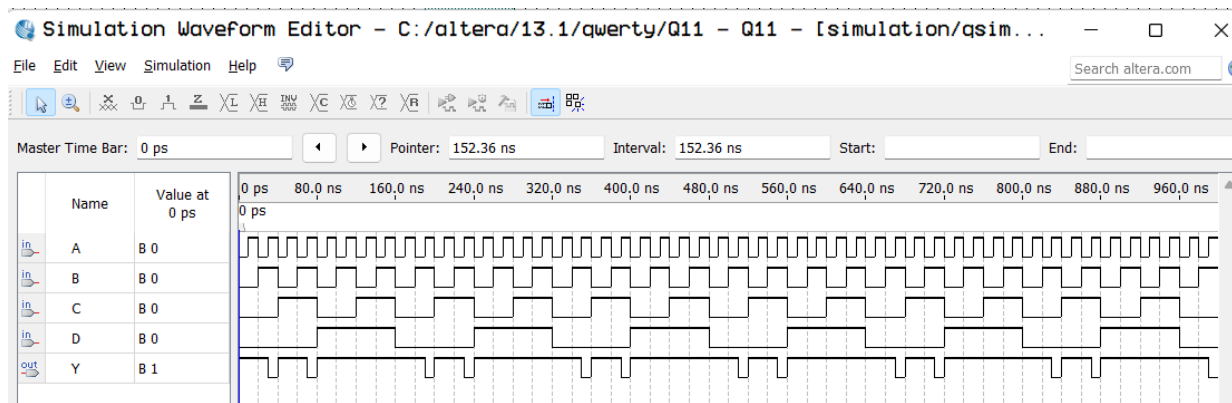


Рисунок 2 – Временная диаграммы схемы

## Написание кода

Напишем код, который реализует работу данной схемы.

### Quartus II 64-Bit - C:/altera/13.1/Q12 - Q12

File Edit View Project Assignments Processing Tools Window Help

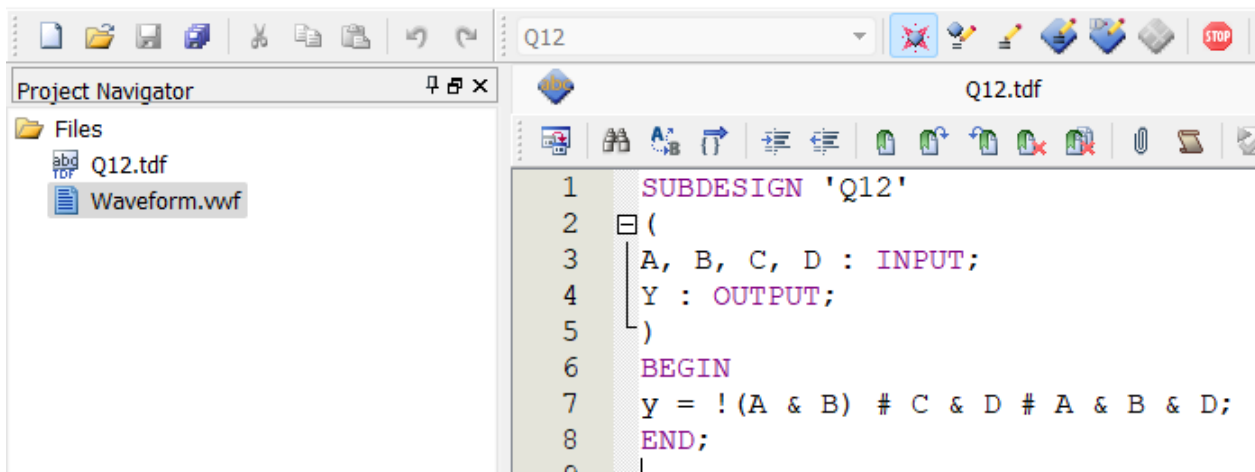


Рисунок 3 – Код схемы

## Временная диаграмма кода

Произведем симуляцию работы кода и нарисуем ее временную диаграмму.

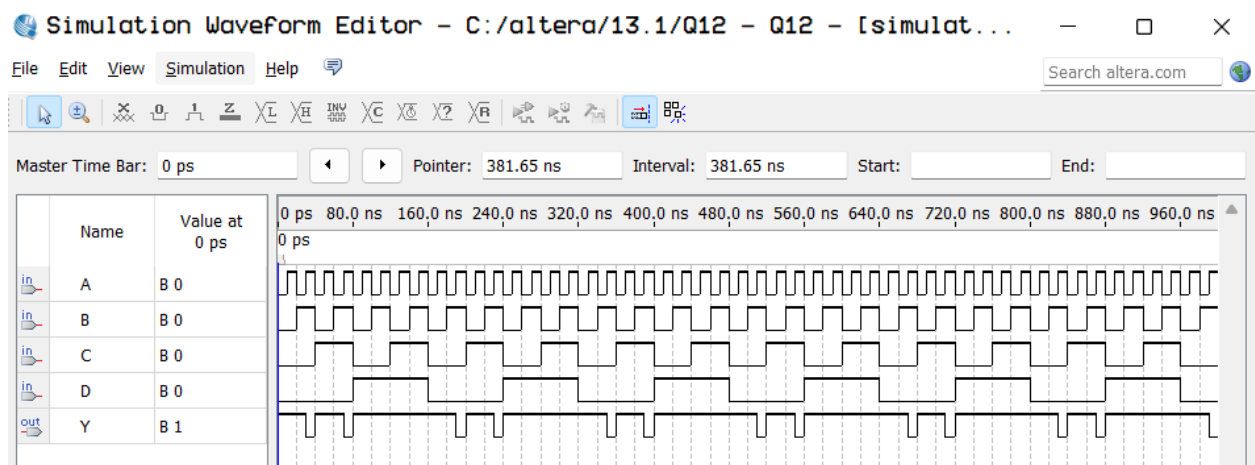


Рисунок 4 – Временная диаграмма кода

## **Сравнение временных диаграмм с таблицей истинности**

Заметим, что временные диаграммы для схемы и кода полностью совпадают. Также, при любых наборах (A, B, C, D), значение Y полностью совпадает с таблицей истинности.

### **Выводы**

В ходе выполнения практической работы мы научились строить схемы для логических выражений при помощи графического редактора САПР QUARTUS II, писать код для них в текстовом редакторе и зарисовывать к ним временные диаграммы. Результаты временных диаграмм сошлись с таблицей истинности, а значит работа выполнена правильно.

## ПРАКТИЧЕСКАЯ РАБОТА №2 (Q21+22)

### Теоретическое введение

При выполнении практических работ Q11 и Q12, мы научились строить схемы и писать для них код. Полученных знаний хватит, чтобы реализовать простейшие Комбинационные Схемы и Цифровые Автоматы, а также построить для них временные диаграммы.

### Вариант 31

1. Создать синтезированную схему в графическом редакторе САПР QUARTUS II в соответствии с вариантом 31 (2xMUX)
2. Произвести симуляцию работы схемы. Зарисовать ее временную диаграмму.
3. Сделать описание заданной электрической схемы при помощи текстового редактора САПР QUARTUS II.
4. Произвести симуляцию работы кода. Зарисовать ее временную диаграмму.
5. Сравнить результаты, полученные в ходе выполнения лабораторной работы с таблицей истинности.

### Таблица истинности узла

Построим таблицу истинности Мультиплексора 2xMUX, где A0, A1 – адресные входы; A, B, C, D – информационные входы; E – выход.

Таблица 2 – Таблица истинности 2xMUX

A1	A0	E
X	X	0
0	0	A
0	1	B
1	0	C
1	1	D

### Построение схемы

Построим схему для нашего мультиплексора.



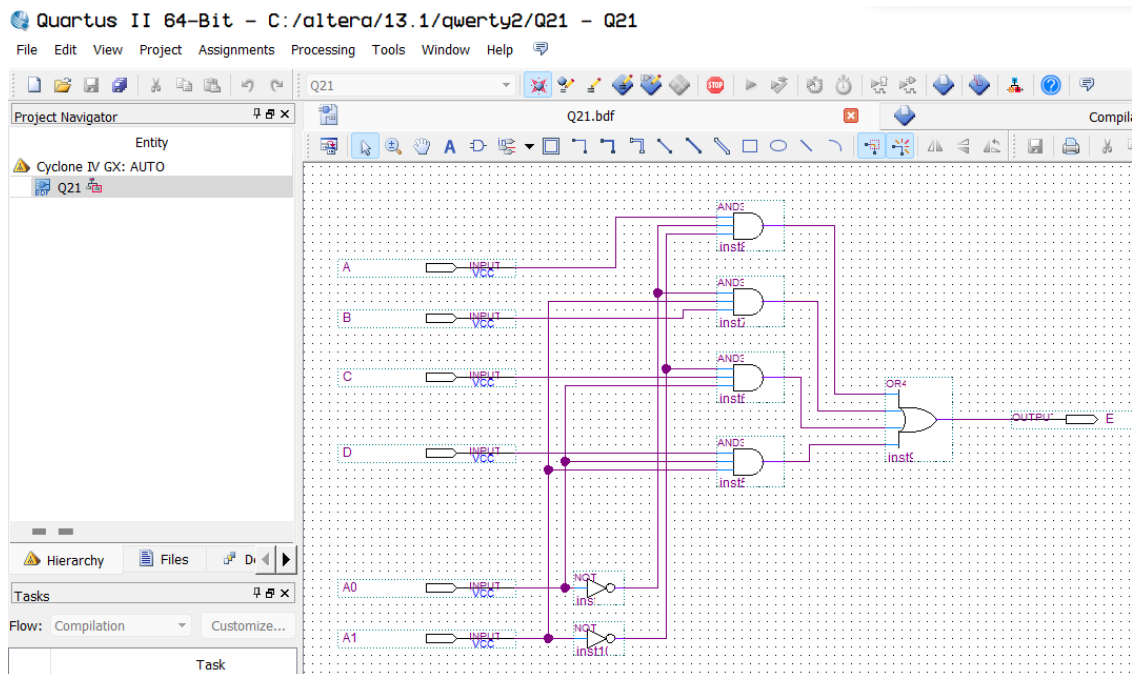


Рисунок 5 – Схема 2xMUX

### Временная диаграмма схемы

Произведем симуляцию работы схемы и построим для нее временную диаграмму.

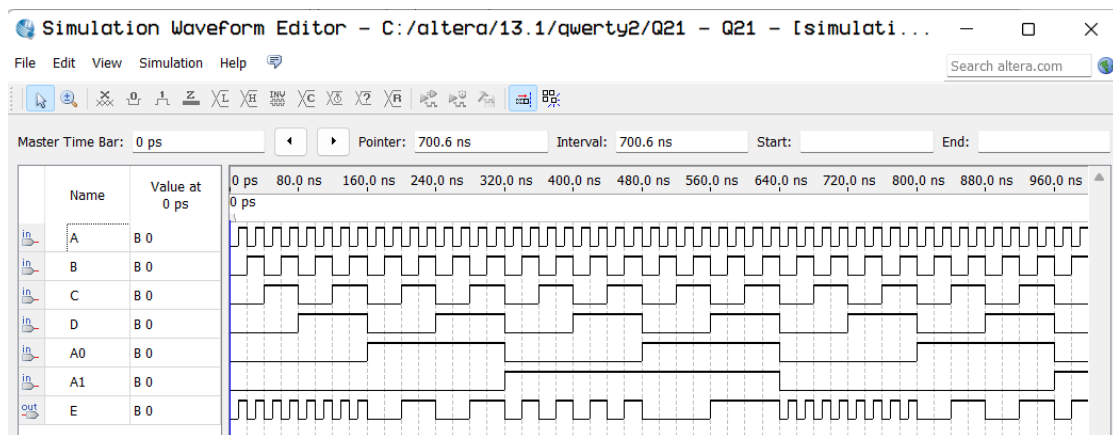


Рисунок 6 – Временная диаграмма работы схемы

### Написание кода

Напишем код, который реализует работу данной схемы.

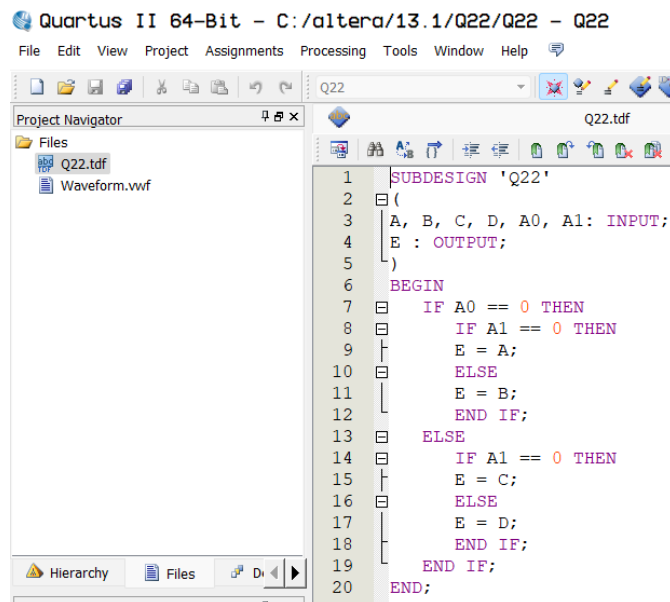


Рисунок 7 – Код схемы 2xMUX

### Временная диаграмма кода

Произведем симуляцию работы кода и построим для нее временную диаграмму.

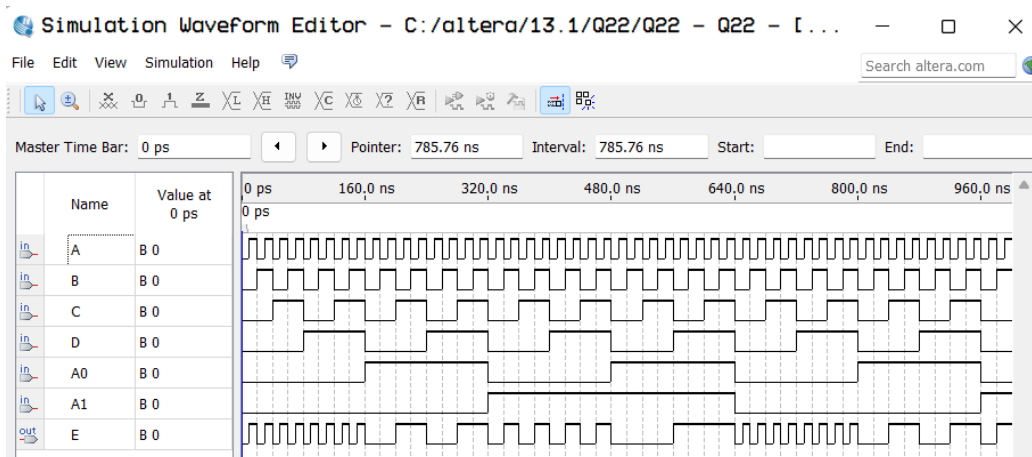


Рисунок 8 – Временная диаграмма работы кода

### Сравнение временных диаграмм с таблицей истинности

Заметим, что временные диаграммы для схемы и кода полностью совпадают. Также, при любых наборах (A0, A1, A, B, C, D), значение Q полностью совпадает с таблицей истинности.

### Выводы

В ходе выполнения практической работы мы построили мультиплексор при помощи графического редактора САПР QUARTUS II и написали код для него в текстовом редакторе, а также зарисовали к ним временные диаграммы. Результаты временных диаграмм сошлись с таблицей истинности, а значит работа выполнена правильно.

# ПРАКТИЧЕСКАЯ РАБОТА №3 (Q3)

## Вариант 31

1. Спроектировать электрическую схему 2xMUX с использованием параметрических элементов при помощи графического редактора САПР QUARTUS II.

2. Зарисовать временную диаграмму работы схемы и сравнить ее с таблицей истинности.

### Таблица истинности узла

Таблица истинности MUX 3-1 представлена в Практической работе №2 в Таблице 2.

### Этап настройки узла

Для того, чтобы вставить готовую схему 2xMUX, зайдём в Symbol, далее megafunctions -> gates -> lpm\_mux.

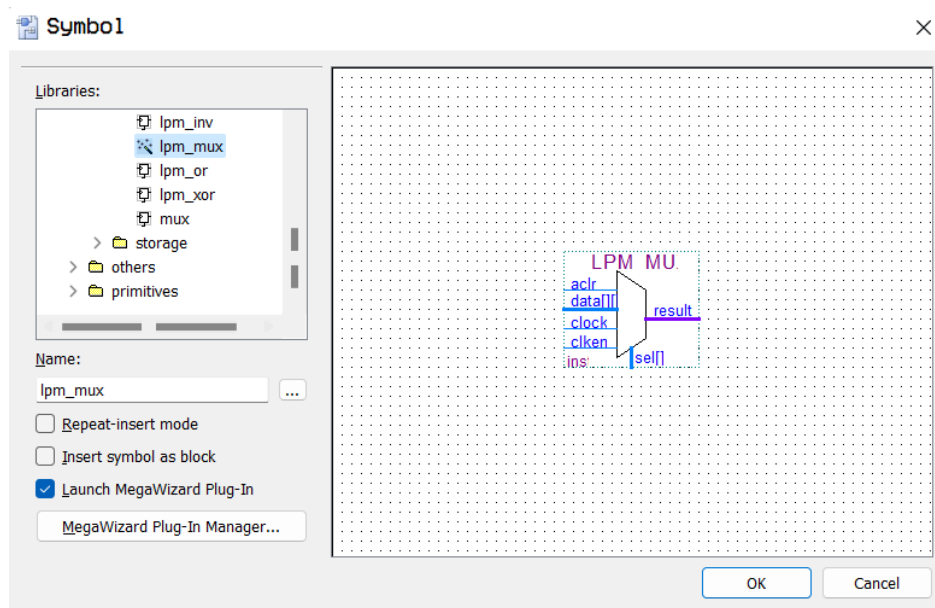


Рисунок 9 – Создание мультиплексора

Далее переходим к его настройке. Так как наш мультиплексор имеет 4 информационных входа, в поле data выбираем 4 информационных входа. Количество адресных входов программа определит сама.

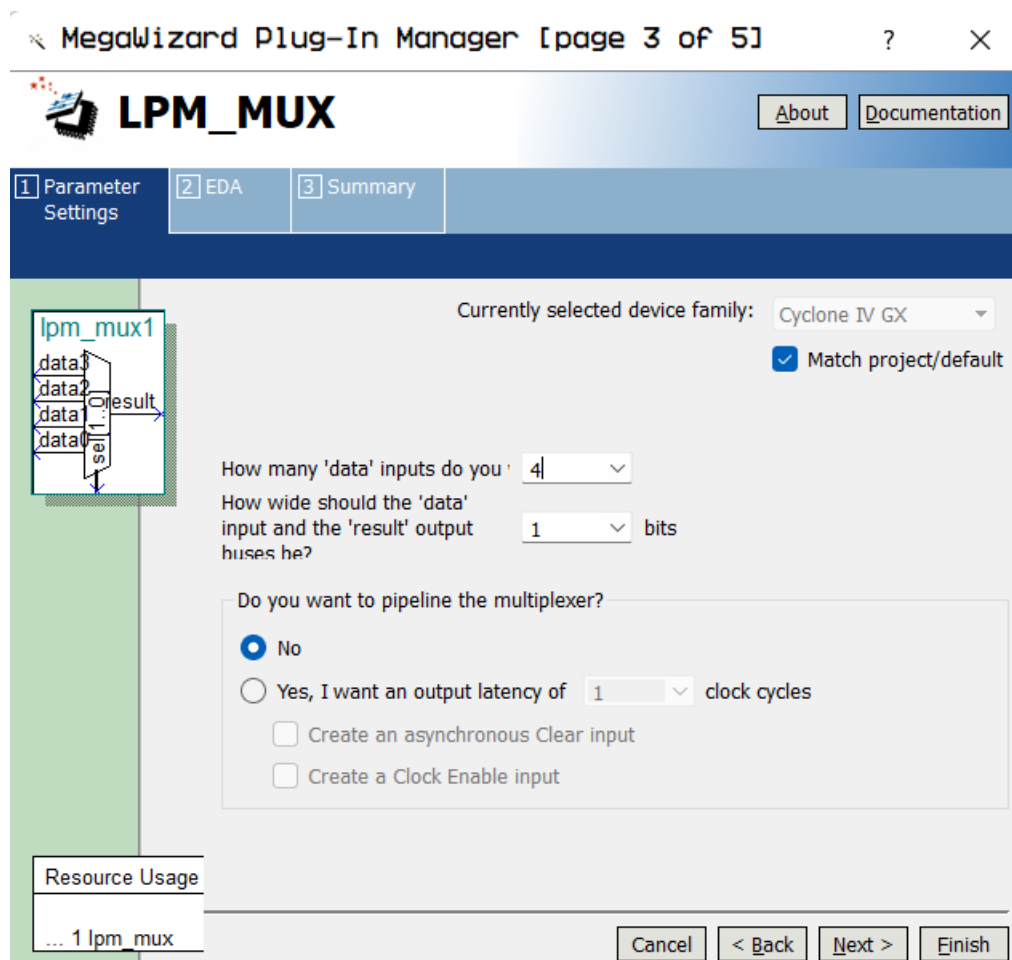


Рисунок 10 – Создание MUX 4-1

После этих шагов нажимаем Finish и программа даст нам разместить наш мультиплексор в графическом редакторе. Стоит заметить, что адресные входы подключаются к мультиплексору с помощью шины.

### Построение схемы

Добавим к нашему мультиплексору 4 информационных входа D0..D3 и шину снизу для адресных входов A[1..0]. Также разместим выход Q к проводу result.

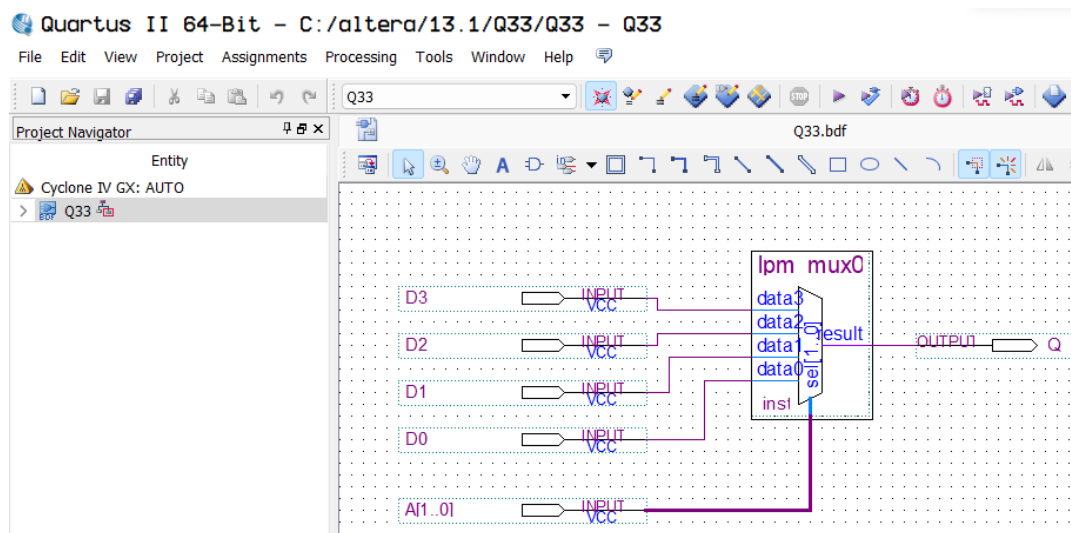


Рисунок 11 – Готовая схема MUX 3-1

### Временная диаграмма схемы

Произведем симуляцию работы схемы и построим для нее временную диаграмму.

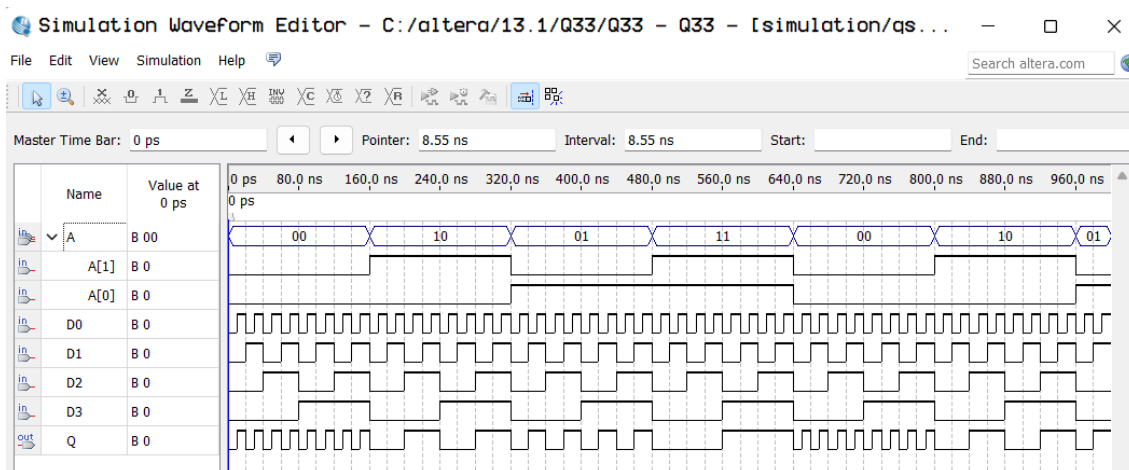


Рисунок 12 – Временная диаграмма схемы

### Сравнение временных диаграмм с таблицей истинности

Заметим, что во временной диаграмме при любых наборах (A0..A1, D0..D3), значение Q полностью совпадает с таблицей истинности.

### Выводы

В ходе выполнения практической работы мы спроектировали электрическую схему мультиплексора 4-1 с использованием параметрических элементов при помощи графического редактора САПР QUARTUS II, а также зарисовали к нему временную диаграмму. Результаты временной диаграммы сошлись с таблицей истинности, а значит работа выполнена правильно.

# ПРАКТИЧЕСКАЯ РАБОТА №4 (VM1)

## Теоретическое введение

Виртуальная машина – это программа, которая эмулирует реальный (физический) компьютер со всеми его компонентами (жёсткий диск, DVD/CD привод, BIOS, сетевые адаптеры и т.д.). На такой виртуальный компьютер можно установить, например, операционную систему, драйверы, программы и т.д. Таким образом, вы можете запустить на своем реальном компьютере еще несколько виртуальных компьютеров, с такой же или другой операционной системой. Вы можете без проблем осуществить обмен данными между вашим реальным и виртуальным компьютером.

Виртуальную машину используют для различных целей и задач:

- установка второй/другой операционной системы;
- тестирование программного обеспечения;
- безопасный запуск подозрительных программ;
- эмуляция компьютерной сети;
- запуск приложений, которые нельзя запустить в вашей операционной системе.

Существует достаточно обширный список средств для создания и управления виртуальными машинами. В данной лабораторной работе мы рассмотрим одно из них – виртуальную машину VirtualBox фирмы Oracle.

VirtualBox – это бесплатный программный продукт, с помощью которого можно создавать виртуальные машины и установить на них все самые популярные операционные системы. VirtualBox поддерживает работу с Windows, Linux, FreeBSD, Mac OS.

## Вариант

Целью данной практической работы является получение практических навыков установки и создания виртуальных машин в Oracle VirtualBox, а также изучение принципов инсталляции и начальной настройки операционной системы Ubuntu Linux.

В результате выполнения практической работы мы познакомимся с процессом установки на персональный компьютер виртуальной машины

OracleVirtualBox, получим представление о процессе создания и настройки виртуального окружения. На примере операционной системы UbuntuLinux будет выполнен процесс установки и базовой настройки операционной системы.

### Знакомство с ВМ

После установки виртуальной машины по методическим указаниям, мы можем ее настроить. В своих настройках для ВМ, я выделила 4гб ОЗУ, 6 ЦП и 12гб внутренней памяти.

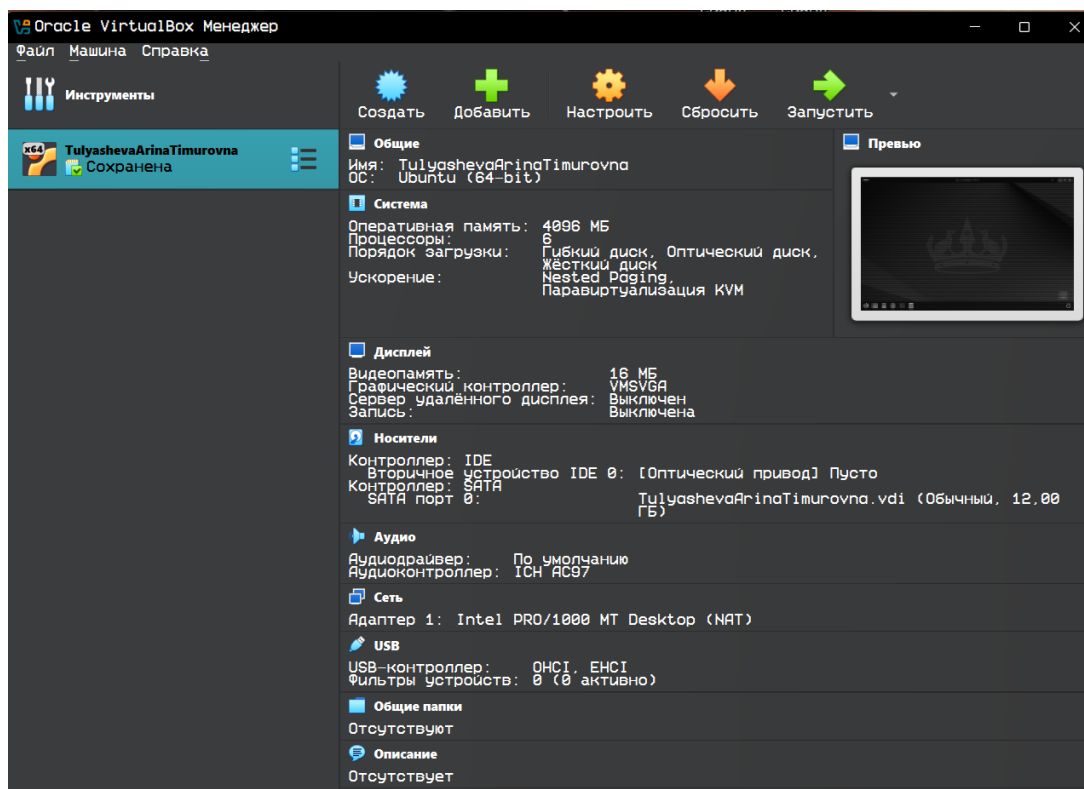


Рисунок 13 – Настройки ВМ

Готовая к работе ВМ представлена на рисунке 14.

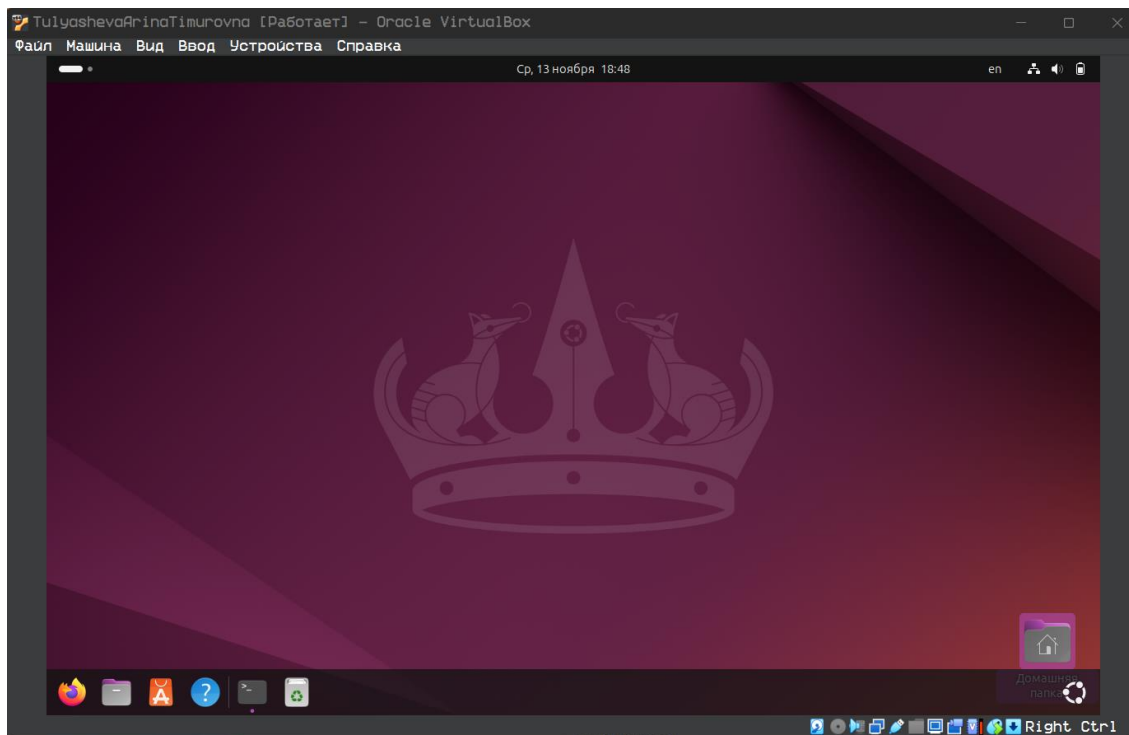


Рисунок 14 – Виртуальная машина

### **Выводы**

В данной лабораторной работе мы приобрели навыки установки OracleVirtualBox, познакомились с процессом создания виртуальной машины. Выполнили установку и начальную настройку операционной системы UbuntuLinux. Созданная в процессе выполнения виртуальная машина понадобится для выполнения последующих лабораторных работ.



# ПРАКТИЧЕСКАЯ РАБОТА №5 (VM2)

## Теоретическое введение

Файловая система — порядок, определяющий способ организации, хранения и именования данных на носителях информации в компьютерах, а также в другом электронном оборудовании: цифровых фотоаппаратах, мобильных телефонах и т. п. Файловая система определяет формат содержимого и способ физического хранения информации, которую принято группировать в виде файлов. Конкретная файловая система определяет размер имен файлов (и каталогов), максимальный возможный размер файла и раздела, набор атрибутов файла. Некоторые файловые системы предоставляют сервисные возможности, например, разграничение доступа или шифрование файлов.

### Основные команды для работы с файловой системой

Для получения подробной справки по каждой из команд необходимо набрать команду `man "имя команды"`. В справке содержится описание команды, область ее применения, синтаксис вызова, возможные параметры вызова.

`.` – ссылка на текущий каталог. Текущим называется каталог, с которым работает операционная система, если ей не указать другого каталога.

`..` – ссылка на родительский каталог. Родительским каталогом называется каталог, в котором находится текущий.

**cat** – команда объединения/слияния данных. Имя команды является сокращением от английского слова concatenate.

Есть возможность перенаправить вывод на устройство или в файл, используя оператор `>`. Пример: `cat filename1.txt filename2.txt > filename3.txt`. В данном примере содержимое файлов `filename1.txt` и `filename2.txt` будет объединено и записано в файл `filename3.txt`. При этом если файл `filename3.txt` существовал, он будет перезаписан. Если необходимо дописать информацию в конец файла, необходимо использовать оператор `>>`. Пример: `cat filename1.txt >> filename2.txt`. Данные из файла `filename1.txt` будут дописаны в конец файла `filename2.txt`. Если `filename2.txt` не существовало, он будет создан.

Также команда `cat` используется для организации конвейера для ввода информации с клавиатуры в файл. В этом случае формат команды следующий:

`cat > filename.txt << EOF`. По выполнении команды последовательно будет запрашиваться информация с клавиатуры. Для завершения ввода необходимо с новой строки ввести последовательность “EOF”.

**cd** – команда для изменения текущего каталога. В качестве аргумента команды задается абсолютное или относительное имя каталога, который необходимо сделать текущим.

**echo**– команда, предназначенная для вывода строки текста в стандартный поток вывода. Команда поддерживает возможность перенаправления вывода (см. примеры для команды `cat`).

**tree** - команда выводит содержимое текущего каталога в виде дерева.

**grep** – команда строковой фильтрации текстовых данных. Она использует компактный недетерминированный алгоритм сопоставления. В качестве параметра принимает строку шаблона для поиска, сформированную в соответствии с правилами составления паттернов для регулярных выражений (стандарт PERL). Команда может использоваться как самостоятельно, принимая на вход имя файла, так и в составе конвейера.

**ls** – команда для вывода в стандартный поток вывода содержимого каталога.

**mkdir** – команда для создания директории. Для выполнения команды необходимо обладать правами на запись для текущего каталога. Идентификатор владельца и группы нового каталога устанавливаются соответственно равными реальным идентификаторам владельца и группы процесса, в контексте которого выполняется команда.

**nano** – консольный текстовый редактор для Unix и Unix-подобных операционных систем.

**pwd** – команда UNIX-подобных системах, которая выводит полный путь от корневого каталога к текущему рабочему каталогу: в контексте которого (по умолчанию) будут исполняться вводимые команды.

**sort**– команда для сортировки содержимого файла в алфавитном или нумерологическом порядке. Если задать несколько файлов, то команда `sort` соединит их и, рассортировав, выдаст единым выводом. По умолчанию,

объектом сортировки будут строки, однако опции позволяют выбирать объект сортировки: колонки, столбцы и прочие элементы форматирования файла. Разделителем между ними служат пробелы, однако соответствующие опции позволяют задать иные разделители.

**uniq** – команда, с помощью которой можно вывести или отфильтровать повторяющиеся строки в файле. Если входной файл задан как («-») или не задан вовсе, чтение производится из стандартного потока ввода. Если выходной файл не задан, запись производится в стандартный поток вывода. Вторая и последующие копии повторяющихся соседних строк не записываются. Повторяющиеся входные строки не распознаются, если они не следуют строго друг за другом, поэтому может потребоваться предварительная сортировка файлов.

**wc** – команда подсчета строк, слов и символов С помощью команды **wc** можно подсчитать число строк, слов и символов в указанном файле. Если указано более одного файла в командной строке, то команда **wc** осуществляет подсчет строк, слов и символов в каждом файле и затем выдает общее число.

### Вариант

1. Все этапы выполнения работы необходимо фиксировать.
2. Войти в систему от имени своей учетной записи. В случае, если вход осуществлен в графическую оболочку, переключиться на текстовую консоль или запустить терминал. Вся дальнейшая работа выполняется исключительно в терминале. Использование root прав запрещено.
3. Создать родительский каталог. В качестве имени каталога задать свою фамилию. Все остальные действия данной лабораторной работы будут выполняться внутри данного каталога.
4. Внутри каталога, созданного на 2-м шаге создать структуру каталогов, представленную на рисунке 15. Вывести на экран содержимое текущего каталога и убедиться, что все созданные каталоги созданы без ошибок. Для отображения используйте утилиту **tree**. При необходимости произведите обновление компонентов и установите утилиту **tree** вручную.

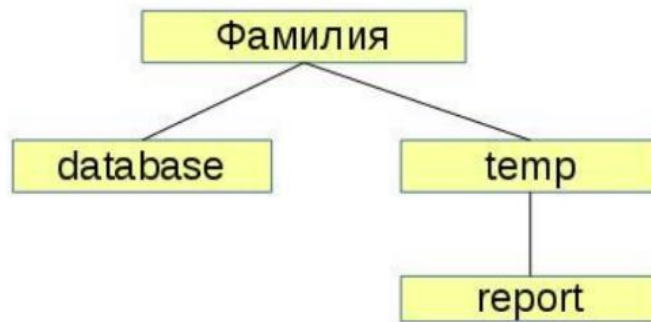


Рисунок 15 – Дерево каталога

5. Перейти в каталог temp. Убедиться, что он является текущим. Вывести на экран содержимое каталога.

6. С помощью встроенного текстового редактора внутри каталога temp создать файл базы данных dataset1.txt. Заполнить файл данными в соответствии с 10 вариантом задания - Экспорт (наименование товара, объем поставки, стоимость единицы продукции, страна экспорта). В качестве разделителя столбцов данных в файле использовать символ “;” без пробелов. Файл должен содержать не менее 3-х строк.

7. С помощью конвейера команд внутри каталога temp создать файл базы данных dataset2.txt. Заполнить файл данными в соответствии с 10 вариантом задания. В качестве разделителя столбцов данных в файле использовать символ “;” без пробелов. Файл должен содержать не менее 4-х строк. Данные должны отличаться от введенных ранее.

8. С помощью перенаправления вывода в файл, либо используя команду echo создать файл базы данных dataset3.txt. Заполнить файл данными в соответствии с 10 вариантом задания. В качестве разделителя столбцов данных в файле использовать символ “;” без пробелов. Файл должен содержать не менее 3-х строк. Данные должны отличаться от введенных ранее.

9. Вывести на экран содержимое всех созданных файлов базы данных.

10. Объединить содержимое всех созданных файлов базы данных в один файл data.txt и поместить его в каталог /database.

11. Перейти в каталог /database. Убедиться, что он является текущим. Вывести на экран содержимое каталога. Убедиться, что созданный файл data.txt содержит все необходимые данные.

12. Подсчитать количество строк файла data.txt. Результат подсчета вывести на экран и в файл отчета output.txt, расположенный в каталоге report.

13. С помощью любого из использованных выше способов дополнить файл data.txt 2-я строками данных в соответствии с 10 вариантом задания. В качестве разделителя столбцов данных в файле использовать символ “;” без пробелов. Убедиться, что файл data.txt содержит все необходимые данные.

14. Повторно подсчитать количество строк файла data.txt. Результат подсчета вывести на экран и дописать в конец файла отчета output.txt, расположенного в каталоге report.

15. Осуществить фильтрацию данных файла data.txt в соответствии с 10 вариантом задания - Поиск по наименованию товара. Результат фильтрации вывести на экран и в файл отчета filtered.txt, расположенный в каталоге report. Повторить фильтрацию с различными значениями фильтра. Результаты фильтрации выводить на экран и дописывать в файл отчета filtered.txt.

16. Выполнить сортировку содержимого файла data.txt в соответствии с 10 вариантом задания – Сортировка по объемам поставки. Результат сортировки вывести на экран и в файл отчета sorted.txt, расположенный в каталоге report.

17. Выполнить фильтрацию содержимого файла data.txt с сортировкой результата фильтрации. Фильтрацию и сортировку выполнить в соответствии с 10 вариантом задания. Результат вывести на экран и в файл отчета filteredsorted.txt, расположенный в каталоге report.

18. Исследовать самостоятельно команды: date, cal, pwd, who, clear, exit.

19. Выполнить команду вывода календаря на экран и любым известным способом записать значение в файл calendar.txt, находящийся в каталоге /database. Результат вывести на экран.

### **Выполнение заданий**

В соответствии с моим вариантом, выполним задания практической работы. Ход и результаты выполнения заданий указаны на рисунках 16-20.

```
TulyashevaArinaTimurovna [Работает] - Oracle VirtualBox
Файл Машина Вид Ввод Устройства Справка

Ср, 13 ноября 18:58

tulyasheva@TulyashevaArinaTimurovna-VB: ~/Tulyasheva/temp/report

tulyasheva@TulyashevaArinaTimurovna-VB:~$ mkdir Tulyasheva
tulyasheva@TulyashevaArinaTimurovna-VB:~/Tulyasheva$ cd Tulyasheva
tulyasheva@TulyashevaArinaTimurovna-VB:~/Tulyasheva$ mkdir temp database
tulyasheva@TulyashevaArinaTimurovna-VB:~/Tulyasheva$ cd temp
tulyasheva@TulyashevaArinaTimurovna-VB:~/Tulyasheva/temp$ mkdir report
tulyasheva@TulyashevaArinaTimurovna-VB:~/Tulyasheva/temp$ cd ../Tulyasheva
bash: cd: ../Tulyasheva: Нет такого файла или каталога
tulyasheva@TulyashevaArinaTimurovna-VB:~/Tulyasheva/temp$ cd ../../
tulyasheva@TulyashevaArinaTimurovna-VB:~/Tulyasheva$ cd Tulyasheva
tulyasheva@TulyashevaArinaTimurovna-VB:~/Tulyasheva$ tree
.
├── database
├── temp
└── report

4 directories, 0 files
tulyasheva@TulyashevaArinaTimurovna-VB:~/Tulyasheva$ cd temp
tulyasheva@TulyashevaArinaTimurovna-VB:~/Tulyasheva/temp$ pwd
/home/tulyasheva/Tulyasheva/temp
tulyasheva@TulyashevaArinaTimurovna-VB:~/Tulyasheva/temp$ ls
report
tulyasheva@TulyashevaArinaTimurovna-VB:~/Tulyasheva/temp$ nano dataset1.txt
tulyasheva@TulyashevaArinaTimurovna-VB:~/Tulyasheva/temp$ cat dataset1.txt
Товар1;500;5000;Страна1
Товар2;10;1000;Страна2
tulyasheva@TulyashevaArinaTimurovna-VB:~/Tulyasheva/temp$ echo -e "Товар4;200;200;Страна4\nТовар5;120;12;Страна5\nТовар6;600;1
0;Страна6\nТовар7;100;350;Страна7" > dataset2.txt
tulyasheva@TulyashevaArinaTimurovna-VB:~/Tulyasheva/temp$ cat dataset2.txt
Товар4;200;200;Страна4
Товар5;120;12;Страна5
Товар6;600;10;Страна6
```

Рисунок 16 – VM2

```
TulyashevaArinaTimurovna [Работает] - Oracle VirtualBox
Файл Машина Вид Ввод Устройства Справка

Ср, 13 ноября 18:59

tulyasheva@TulyashevaArinaTimurovna-VB: ~/Tulyasheva/temp/report

tulyasheva@TulyashevaArinaTimurovna-VB:~/Tulyasheva/temp$ cat dataset2.txt
Товар4;200;200;Страна4
Товар5;120;12;Страна5
Товар6;600;10;Страна6
Товар7;100;350;Страна7
tulyasheva@TulyashevaArinaTimurovna-VB:~/Tulyasheva/temp$ cat dataset1.txt
Товар1;500;5000;Страна1
Товар2;10;1000;Страна2
Товар3;3000;3;Страна3
tulyasheva@TulyashevaArinaTimurovna-VB:~/Tulyasheva/temp$ echo -e "Товар8;550;150;Страна8\nТовар9;900;11;Страна9\nТовар10;60;3
10;Страна10" > dataset3.txt
tulyasheva@TulyashevaArinaTimurovna-VB:~/Tulyasheva/temp$ cat dataset3.txt
Товар8;550;150;Страна8
Товар9;900;11;Страна9
Товар10;60;310;Страна10
tulyasheva@TulyashevaArinaTimurovna-VB:~/Tulyasheva/temp$ ls
dataset1.txt dataset2.txt dataset3.txt report
tulyasheva@TulyashevaArinaTimurovna-VB:~/Tulyasheva/temp$ cd dataset1.txt dataset2.txt dataset3.txt > ../database/data.txt
tulyasheva@TulyashevaArinaTimurovna-VB:~/Tulyasheva/temp$ cd ../database
tulyasheva@TulyashevaArinaTimurovna-VB:~/Tulyasheva/database$ pwd
/home/tulyasheva/Tulyasheva/database
tulyasheva@TulyashevaArinaTimurovna-VB:~/Tulyasheva/database$ ls
data.txt
tulyasheva@TulyashevaArinaTimurovna-VB:~/Tulyasheva/database$ cat data.txt
Товар1;500;5000;Страна1
Товар2;10;1000;Страна2
Товар3;3000;3;Страна3
Товар4;200;200;Страна4
Товар5;120;12;Страна5
Товар6;600;10;Страна6
Товар7;100;350;Страна7
```

Рисунок 17 – VM2

```
TulyashevaArinaTimurovna [Работает] - Oracle VirtualBox
Файл Машина Вид Ввод Устройства Справка
Ср, 13 ноября 19:00
en
tulyasheva@TulyashevaArinaTimurovna-VB: ~/Tulyasheva/temp/report

tulyasheva@TulyashevaArinaTimurovna-VB: ~/Tulyasheva/database$ cat data.txt
Товар1;500;5000;Страна1
Товар2;10;1000;Страна2
Товар3;3000;3;Страна3
Товар4;200;200;Страна4
Товар5;120;12;Страна5
Товар6;600;10;Страна6
Товар7;100;350;Страна7
Товар8;550;150;Страна8
Товар9;900;11;Страна9
Товар10;60;310;Страна10
tulyasheva@TulyashevaArinaTimurovna-VB: ~/Tulyasheva/database$ wc -l < data.txt
10
tulyasheva@TulyashevaArinaTimurovna-VB: ~/Tulyasheva/database$ wc -l < data.txt >> ../temp/report/output.txt
tulyasheva@TulyashevaArinaTimurovna-VB: ~/Tulyasheva/database$ cd -
/home/tulyasheva/Tulyasheva/temp
tulyasheva@TulyashevaArinaTimurovna-VB: ~/Tulyasheva/temp$ cd report
tulyasheva@TulyashevaArinaTimurovna-VB: ~/Tulyasheva/temp/report$ ls
output.txt
tulyasheva@TulyashevaArinaTimurovna-VB: ~/Tulyasheva/temp/report$ cat output
cat: output: Нет такого файла или каталога
tulyasheva@TulyashevaArinaTimurovna-VB: ~/Tulyasheva/temp/report$ cat output.txt
10
tulyasheva@TulyashevaArinaTimurovna-VB: ~/Tulyasheva/temp/report$ cd ../database
bash: cd: ../database: Нет такого файла или каталога
tulyasheva@TulyashevaArinaTimurovna-VB: ~/Tulyasheva/temp/report$ cd -
/home/tulyasheva/Tulyasheva/temp
tulyasheva@TulyashevaArinaTimurovna-VB: ~/Tulyasheva/temp$ cd ../database
tulyasheva@TulyashevaArinaTimurovna-VB: ~/Tulyasheva/database$ echo -e "Товар11;1550;8;Страна11\nТовар12;55;90;Страна12" >> data.txt
tulyasheva@TulyashevaArinaTimurovna-VB: ~/Tulyasheva/database$ wc -l < data.txt
```

Рисунок 18 – VM2

```
TulyashevaArinaTimurovna [Работает] - Oracle VirtualBox
Файл Машина Вид Ввод Устройства Справка
Ср, 13 ноября 19:01
en
tulyasheva@TulyashevaArinaTimurovna-VB: ~/Tulyasheva/temp/report

tulyasheva@TulyashevaArinaTimurovna-VB: ~/Tulyasheva/database$ grep "Товар5" data.txt | tee ../temp/report/filtered.txt
Товар5;120;12;Страна5
tulyasheva@TulyashevaArinaTimurovna-VB: ~/Tulyasheva/database$ sort -t';' -k2n data.txt > ../temp/report/sorted.txt
tulyasheva@TulyashevaArinaTimurovna-VB: ~/Tulyasheva/database$ cat ../temp/report/sorted.txt
Товар2;10;1000;Страна2
Товар12;55;90;Страна12
Товар10;60;310;Страна10
Товар7;100;350;Страна7
Товар5;120;12;Страна5
Товар4;200;200;Страна4
Товар1;500;5000;Страна1
Товар8;550;150;Страна8
Товар6;600;10;Страна6
Товар9;900;11;Страна9
Товар11;1550;8;Страна11
Товар3;3000;3;Страна3
tulyasheva@TulyashevaArinaTimurovna-VB: ~/Tulyasheva/database$ cat ../temp/report/filtered.txt
Товар5;120;12;Страна5
tulyasheva@TulyashevaArinaTimurovna-VB: ~/Tulyasheva/database$ grep "Товар10" data.txt | tee ../temp/report/filteredsorted.txt
Товар10;60;310;Страна10
tulyasheva@TulyashevaArinaTimurovna-VB: ~/Tulyasheva/database$ cd -
/home/tulyasheva/Tulyasheva
tulyasheva@TulyashevaArinaTimurovna-VB: ~/Tulyasheva$ cd /temp/report
bash: cd: /temp/report: Нет такого файла или каталога
tulyasheva@TulyashevaArinaTimurovna-VB: ~/Tulyasheva$ cd temp
tulyasheva@TulyashevaArinaTimurovna-VB: ~/Tulyasheva/temp$ cd report
tulyasheva@TulyashevaArinaTimurovna-VB: ~/Tulyasheva/temp/report$ ls
filteredsorted.txt filtered.txt output.txt sorted.txt
tulyasheva@TulyashevaArinaTimurovna-VB: ~/Tulyasheva/temp/report$ sort -t';' -k2n filteredsorted.txt > ../temp/report/filteredsorted.txt
```

Рисунок 19 – VM2

```
tulyasheva@TulyashevaArinaTimurovna-VB: ~/Tulyasheva/database$ grep "Товар10" data.txt | sort -t';' -k2n > ../temp/report//filteredsorted.txt
Товар10;60;310;Страна10
tulyasheva@TulyashevaArinaTimurovna-VB:~/Tulyasheva/database$ cat ../temp/report/filteredsorted.txt
Товар10;60;310;Страна10
tulyasheva@TulyashevaArinaTimurovna-VB:~/Tulyasheva/database$ cal
Ноябрь 2024
Вс Пн Вт Ср Чт Пт Сб
      1  2
 3  4  5  6  7  8  9
10 11 12 13 14 15 16
17 18 19 20 21 22 23
24 25 26 27 28 29 30

tulyasheva@TulyashevaArinaTimurovna-VB:~/Tulyasheva/database$ cal > calendar.txt
tulyasheva@TulyashevaArinaTimurovna-VB:~/Tulyasheva/database$ cat calendar.txt
Ноябрь 2024
Вс Пн Вт Ср Чт Пт Сб
      1  2
 3  4  5  6  7  8  9
10 11 12 13 14 15 16
17 18 19 20 21 22 23
24 25 26 27 28 29 30

tulyasheva@TulyashevaArinaTimurovna-VB:~/Tulyasheva/database$ date
Вт 12 ноя 2024 13:17:28 MSK
tulyasheva@TulyashevaArinaTimurovna-VB:~/Tulyasheva/database$ pwd
/home/tulyasheva/Tulyasheva/database
tulyasheva@TulyashevaArinaTimurovna-VB:~/Tulyasheva/database$ who
tulyasheva seat0      2024-11-11 22:57 (login screen)
tulyasheva tty2      2024-11-11 22:57 (tty2)
```

Рисунок 20 – VM2

В ходе выполнения 19 задания, мы протестировали разные команды, которые не тестировались в заданиях 3-18. Команда `clear` очищает терминал, а команда `exit` закрывает его.

### Выводы

В данной практической работе мы приобрели навыки работы с файлами и каталогами, познакомились с некоторыми командами манипулирования данными на примере текстовой базы данных.

Задания практической работы были успешно выполнены.



# ПРАКТИЧЕСКАЯ РАБОТА №6 (VM3)

## Теоретическое введение

AWK — это интерпретируемый скриптовый C-подобный язык построчного разбора и обработки входного потока (например, текстового файла) по заданным шаблонам (регулярным выражениям). Используется в bash (SH) скриптах.

Благодаря AWK в нашем распоряжении оказывается язык программирования, а не довольно скромный набор команд, отдаваемых редактору. С помощью языка программирования AWK можно выполнять следующие действия:

- объявлять переменные для хранения данных;
- использовать арифметические и строковые операторы для работы с данными;
- использовать структурные элементы и управляющие конструкции языка, такие, как условные операторы и циклы;
- реализовать сложные алгоритмы обработки данных;
- создавать форматированные отчёты.

AWK может запоминать контекст, делать сравнения, создавать форматированные отчёты, которые удобно читать и анализировать. Это оказывается очень кстати при работе с лог-файлами, которые могут содержать миллионы записей. При надлежащей сноровке, она может объединять множество строк. Awk – это инструмент, предоставляющий несколько очень удобных способов обработки текстовых данных, которые могут пригодиться в повседневной жизни.

В общем виде программа awk состоит из операторов (правил), имеющих вид:

шаблон {действие}

шаблон {действие}

...

Шаблон задает правила для отбора обрабатываемых строк в потоке данных. Строки несоответствующие шаблону не обрабатываются. При

составлении шаблона используется синтаксис схожий с синтаксисом регулярных выражений языка программирования PERL.

В случае, если шаблон не задан, обрабатываются все строки потока данных.

Пример:

```
awk '{print}' f-awk # выдает весь текст;
```

```
awk '/до/ {print}' f-awk # выдает строки, где есть "до".
```

```
awk '/до/ {}' f-awk # выдает строки, где есть "до"
```

```
awk '/до/ {print("Привет!")}' f-awk
```

Действие – последовательность команд манипулирования с данными, заключенная в фигурные скобки. Команды разделяются точкой с запятой, переводом строки или закрывающей скобкой.

Внутри awk программы возможны комментарии (как в shell "#.....").

Скрипты awk, которые можно писать прямо в командной строке, оформляются в виде текстов команд, заключённых в фигурные скобки. Кроме того, так как awk предполагает, что скрипт представляет собой текстовую строку, его нужно заключить в одинарные кавычки.

Одна из основных функций awk заключается в возможности манипулировать данными в текстовых файлах. Делается это путём автоматического назначения переменной каждому элементу в строке. По умолчанию awk назначает следующие переменные каждому полю данных, обнаруженному им в записи:

\$0 — представляет всю строку текста (запись).

\$1 — первое поле.

\$2 — второе поле.

\$n — n-ное поле.

Поля выделяются из текста с использованием символа-разделителя. По умолчанию — это пробельные символы вроде пробела или символа табуляции.

Вызов awk с одной командой обработки текста — подход очень ограниченный. Awk позволяет обрабатывать данные с использованием

многострочных скриптов. Awk позволяет хранить скрипты в файлах и ссылаться на них, используя ключ -f.

Иногда нужно выполнить какие-то действия до того, как скрипт начнёт обработку записей из входного потока. Для этого можно воспользоваться ключевым словом BEGIN. Команды, которые следуют за BEGIN, будут исполнены до начала обработки данных. Ключевое слово END позволяет задавать команды, которые надо выполнить после окончания обработки данных.

Утилита awk использует встроенные переменные, которые позволяют настраивать процесс обработки данных и дают доступ, как к обрабатываемым данным, так и к некоторым сведениям о них. Некоторые позиционные переменные из наиболее часто используемых:

- FIELDWIDTHS — разделённый пробелами список чисел, определяющий точную ширину каждого поля данных с учётом разделителей полей;
- FS — уже знакомая вам переменная, позволяющая задавать символ-разделитель полей;
- RS — переменная, которая позволяет задавать символ-разделитель записей;
- OFS — разделитель полей на выводе awk-скрипта;
- RS — разделитель записей на выводе awk-скрипта.

Переменные, которые предоставляют сведения о данных и об окружении, в котором работает awk:

- ARGV — количество аргументов командной строки;
- ARGV — массив с аргументами командной строки;
- ARGIND — индекс текущего обрабатываемого файла в массиве ARGV;
- ENVIRON — ассоциативный массив с переменными окружения и их значениями;
- ERRNO — код системной ошибки, которая может возникнуть при чтении или закрытии входных файлов;
- FILENAME — имя входного файла с данными;
- FNR — номер текущей записи в файле данных;

- IGNORECASE — если эта переменная установлена в ненулевое значение, при обработке игнорируется регистр символов;

- NF — общее число полей данных в текущей записи;

- NR — общее число обработанных записей.

Команда `printf` в `awk` позволяет выводить форматированные данные. Она даёт возможность настраивать внешний вид выводимых данных благодаря использованию шаблонов, в которых могут содержаться текстовые данные и спецификаторы форматирования. Спецификатор форматирования — это специальный символ, который задаёт тип выводимых данных и то, как именно их нужно выводить. `Awk` использует спецификаторы форматирования как указатели мест вставки данных из переменных, передаваемых `printf`.

Первый спецификатор соответствует первой переменной, второй спецификатор — второй, и так далее.

Спецификаторы форматирования записывают в таком виде:

%[modifier]control-letter

Вот некоторые из них:

`c` — воспринимает переданное ему число как код ASCII-символа и выводит этот символ;

`d` — выводит десятичное целое число;

`i` — то же самое, что и `d`;

`e` — выводит число в экспоненциальной форме;

`f` — выводит число с плавающей запятой;

`g` — выводит число либо в экспоненциальной записи, либо в формате с плавающей запятой, в зависимости от того, как получается короче;

`o` — выводит восьмеричное представление числа;

`s` — выводит текстовую строку;

### Вариант

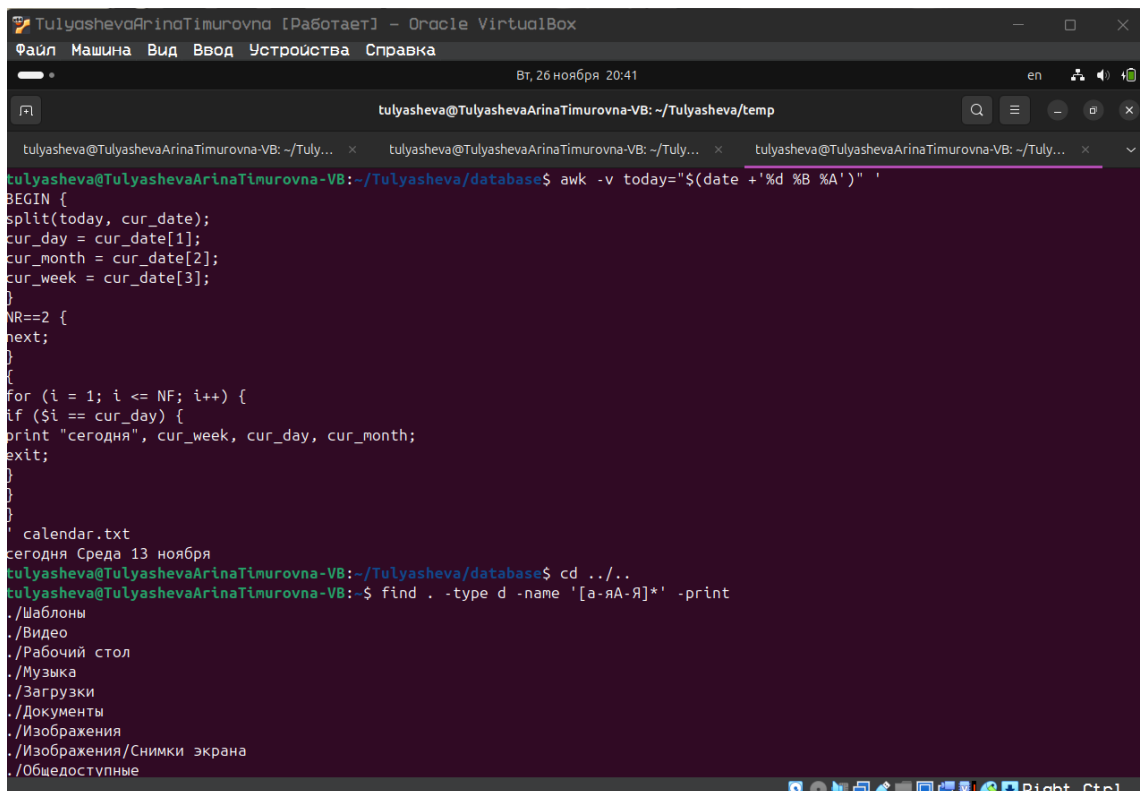
Используя `AWK`:

1. вывести на экран из файла `calendar.txt` день недели и текущее число в виде «сегодня вторник ... августа»;

2. вывести список каталогов, имена которых состоят из русских букв, без дополнительных полей;
3. определить количество(сумму) байтов, занятых всеми вашими текстовыми файлами (txt) в каталогах и подкаталогах;
4. определить количество блоков, содержащих ваш текущий каталог;
5. изменить права доступа для некоторых файлов текущего каталога и провести сортировку списка по возможностям доступа;
6. напечатать список каталогов, в которых обнаружены файлы с именами data\*.txt;
7. подсчитать, сколько раз пользователь входил в систему;
8. напечатать список пользователей, отсортированный по времени.

### Выполнение заданий

Выполним задания практической работы. Ход и результаты выполнения заданий указаны на рисунках 21-22.



```
tulyasheva@TulyashevaArinaTimurovna-VB: ~/Tulyasheva/temp
tulyasheva@TulyashevaArinaTimurovna-VB: ~/Tulyasheva/database$ awk -v today="$(date +%d %B %A)" '
BEGIN {
    split(today, cur_date);
    cur_day = cur_date[1];
    cur_month = cur_date[2];
    cur_week = cur_date[3];
}
NR==2 {
    next;
}
{
    for (i = 1; i <= NF; i++) {
        if ($i == cur_day) {
            print "сегодня", cur_week, cur_day, cur_month;
            exit;
        }
    }
}
' calendar.txt
сегодня Среда 13 ноября
tulyasheva@TulyashevaArinaTimurovna-VB:~/Tulyasheva/database$ cd ../../
tulyasheva@TulyashevaArinaTimurovna-VB:~$ find . -type d -name '[а-яА-Я]*' -print
./Шаблоны
./Видео
./Рабочий стол
./Музыка
./Загрузки
./Документы
./Изображения
./Изображения/Снимки экрана
./Общедоступные
```

Рисунок 21 – VM3

```
tulyasheva@TulyashevaArinaTimurovna-VB: ~/Tulyasheva
./Изображения/Снимки экрана
./Общедоступные
tulyasheva@TulyashevaArinaTimurovna-VB:~$ find . -type f -name "*.txt" -exec du -b {} + | awk '{sum += $1} END {print sum}'
3363
tulyasheva@TulyashevaArinaTimurovna-VB:~$ du -s | awk '{print $1 " блоков"}'
25816 блоков
tulyasheva@TulyashevaArinaTimurovna-VB:~$ chmod 644 dataset*.txt
chmod: невозможно получить доступ к 'dataset*.txt': Нет такого файла или каталога
tulyasheva@TulyashevaArinaTimurovna-VB:~$ cd Tulyasheva/temp
tulyasheva@TulyashevaArinaTimurovna-VB:~/Tulyasheva/temp$ chmod 644 dataset*.txt
tulyasheva@TulyashevaArinaTimurovna-VB:~/Tulyasheva/temp$ ls -l | sort
drwxrwxr-x 2 tulyasheva tulyasheva 4096 ноя 12 13:04 report
-rw-r--r-- 1 tulyasheva tulyasheva 102 ноя 12 12:40 dataset1.txt
-rw-r--r-- 1 tulyasheva tulyasheva 102 ноя 12 12:46 dataset3.txt
-rw-r--r-- 1 tulyasheva tulyasheva 134 ноя 12 12:44 dataset2.txt
итого 16
tulyasheva@TulyashevaArinaTimurovna-VB:~/Tulyasheva/temp$ grep 'session opened for user tulyasheva' /var/log/auth.log | wc -l
2
tulyasheva@TulyashevaArinaTimurovna-VB:~/Tulyasheva/temp$ last | awk '{print $1}' | sort | uniq -c | sort -nr
      2 tulyashe
      1 wtmp
      1 reboot
      1
tulyasheva@TulyashevaArinaTimurovna-VB:~/Tulyasheva/temp$ cd -
/home/tulyasheva
tulyasheva@TulyashevaArinaTimurovna-VB:~$ cd Tulyasheva
tulyasheva@TulyashevaArinaTimurovna-VB:~/Tulyasheva$ find . -type f -name "data*.txt" -printf '%h\n' | sort -u
./database
./temp
tulyasheva@TulyashevaArinaTimurovna-VB:~/Tulyasheva$
```

Рисунок 22 – VM3

### Выводы

В данной практической работе мы познакомились с возможностями программируемого фильтра `awk`. Фильтр широко применяется для обработки данных и формирования различного вида отчетов.

Задания практической работы были успешно выполнены.

# ПРАКТИЧЕСКАЯ РАБОТА №7 (CPU580)

## Теоретическое введение

Эмулятор 8-разрядного процессора реализован на ЭВМ типа IBM — РСАТ. При входе в эмулятор открывается окно ПМК КР580ВМ80, в котором изображена структура «стенда». В верхней части окна указываются режимы работы: «файл», «структурная схема», «система команд», «помощь».

Строкой ниже указывается адрес (состояние шины адреса в шестнадцатеричной системе счисления), данные (состояние шины данных в шестнадцатеричной системе счисления).

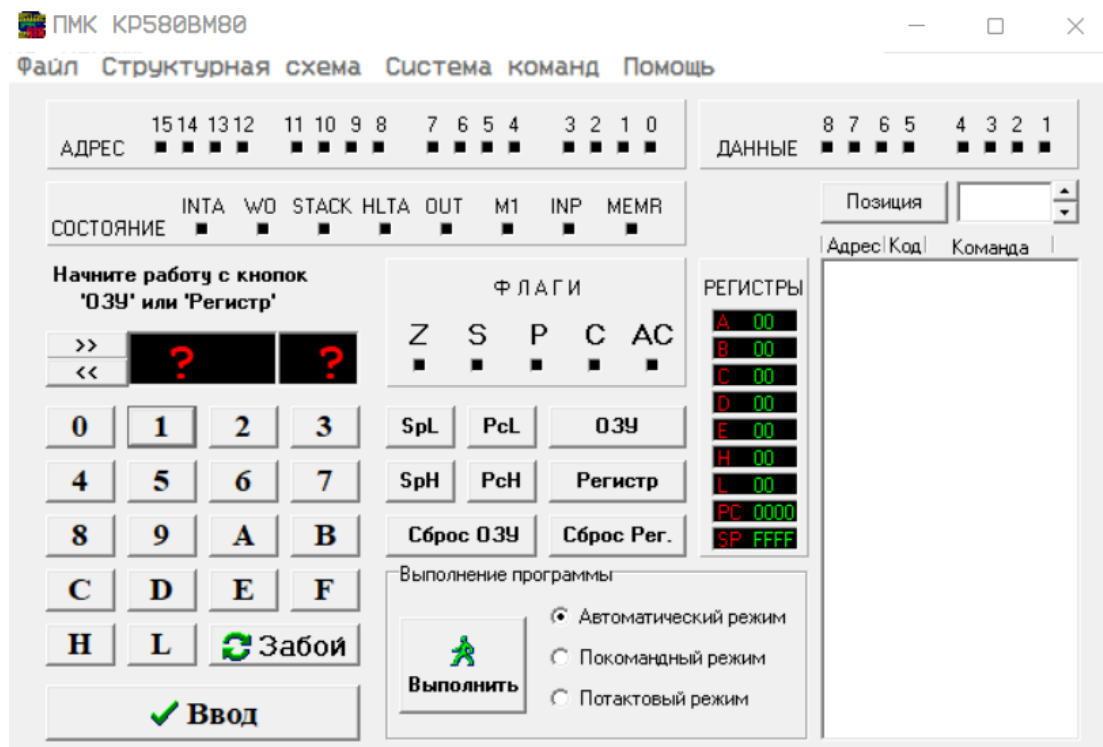


Рисунок 23 - Стенд эмулятора микропроцессора КР580ВМ80

Далее показано окно, в котором указывается адрес ОЗУ и его содержимое, или номер регистра и его содержимое. В окне флагов записывается состояние флагов Z, S, P, C, AC на данный момент выполнения команды процессором, где:

- Z — признак нулевого результата;
- S — знак;
- P — признак четного количества единиц результата;
- C — признак переноса из старшего разряда;
- AC — признак дополнительного переноса из старшего разряда младшей тетрады байта.

Для ввода информации используется клавиатура, которая подразделяется на цифровую от 0 до F и режимную SpL, SpH, PcL, PcH, ОЗУ, Сброс ОЗУ, Регистр, Сброс рег.

Для ввода информации ОЗУ необходимо нажать кнопку «ОЗУ», затем кнопку «Ввод». После этого ввести в данную ячейку код команды или данные.

Для ввода информации в регистры требуется нажать клавишу «Регистр», клавишу «Ввод», а затем на цифровой клавиатуре произвести ввод данных.

Клавиши SpL, SpH и PcL, PcH позволяют записывать адреса в указатель стека и счетчик команд. При вводе любой информации необходимо выполнить нажатия «мышкой» на клавишу «ввод».

В первой части «окна» расположены регистры процессора:

- А — аккумулятор;
- В, С, D, E, H, L — общие регистры R0N, PC — программный счетчик;
- SP — указатель стека — указывает вершину стека FFFF (перевернутый стек).

В области окна «выполнение программы» задается режим:

- автоматический режим — при нажатии кнопки «автоматический режим» выполняется программа до команды HLT;
- покомандный режим — при нажатии кнопки «покомандный режим» реализуется весь цикл исполнения команды;
- потактный режим — при каждом нажатии кнопки «выполнить» осуществляется выполнение одного такта работы процессора.

В правой части окна экрана расположено «окно», в котором указана «Позиция» адресного пространства ОЗУ, где производится набор программы: адрес команды, КОП, ассемблерный код команды. «Позиция» указывает область адресного пространства ОЗУ, в котором прокруткой «Δ» или «∇» можно указывать начало требуемого адресного пространства.

При указании «мышью» позиции «структурная схема» раскрывается следующее окно, в котором показана структурная схема процессора.

Структурная схема 8-разрядного процессора включает АЛУ с входными регистрами («Аккумулятор», «Буф.Регистр 1», «Буф.Регистр 2»),



«Рег.Признаков», «Схема десятичной коррекции»; «Регистр команд», «Деш. Команд», «Блок синхронизации и управления», «Регистры временного хранения W и Z», «регистры общего назначения B, C, D, E, H, L», указатель стека, счетчик команд, схемы инкремента и декремента.

Для организации режимов работы, как и в предыдущем окне, в данном расположена клавиатура ввода информации, аналогичная клавиатуре предыдущего окна, и соответствующие режимные клавиши, а также клавиши режима работы:

- автоматический режим — АТ;
- покомандный режим — КМ;
- потактный режим. При потактном режиме можно исследовать выполнение любой команды.

При каждом нажатии кнопки «Вып.» реализуется очередной такт работы процессора. При этом фиксируется на экране все состояния его регистров, номер такта, номер цикла, состояния управляющих сигналов, регистр состояния и PSW.

Режим работы с клавиатурой соответствует предыдущему окну. При указании «мышью» в окне «ПМК КР580ВМ80» на позицию «Система команд» раскрывается окно «Команды», в котором предоставлена таблица (рис. 24). системы команд. Код команды определяется по матрице в 16-ричной системе счисления по столбцу и строке таблицы, например, MOV B, C-41 или HLT-76.

Система команд																
Команды																
	0	1	2	3	4	5	6	7	8	9	A	B	C	D	E	F
0	NOP	LXI B,d16	STAX B	INX B	INR B	DCR B	MM B,d8	RLC	-	DAD B	LDAX B	DCX B	INR C	DCR C	MM C,d8	RRC
1	-	LXI D,d16	STAX D	INX D	INR D	DCR D	MM D,d8	RAL	-	DAD D	LDAX D	DCX D	INR E	DCR E	MM E,d8	RAR
2	-	LXI H,d16	SHLD adr	INX H	INR H	DCR H	MM H,d8	DAA	-	DAD H	LHLD adr	DCX H	INR L	DCR L	MM L,d8	CMA
3	-	LXI SP,d16	STA adr	INX SP	INR M	DCR M	MM M,d8	STC	-	DAD SP	LDA adr	DCX SP	INR A	DCR A	MM A,d8	CMC
4	MOV B,B	MOV B,C	MOV B,D	MOV B,E	MOV B,H	MOV B,L	MOV B,M	MOV B,A	MOV C,B	MOV C,C	MOV C,D	MOV C,E	MOV C,H	MOV C,L	MOV C,M	MOV C,A
5	MOV D,B	MOV D,C	MOV D,D	MOV D,E	MOV D,H	MOV D,L	MOV D,M	MOV D,A	MOV E,B	MOV E,C	MOV E,D	MOV E,E	MOV E,H	MOV E,L	MOV E,M	MOV E,A
6	MOV H,B	MOV H,C	MOV H,D	MOV H,E	MOV H,H	MOV H,L	MOV H,M	MOV H,A	MOV L,B	MOV L,C	MOV L,D	MOV L,E	MOV L,H	MOV L,L	MOV L,M	MOV L,A
7	MOV M,B	MOV M,C	MOV M,D	MOV M,E	MOV M,H	MOV M,L	MOV M,M	MOV M,A	MOV A,B	MOV A,C	MOV A,D	MOV A,E	MOV A,H	MOV A,L	MOV A,M	MOV A,A
8	ADD B	ADD C	ADD D	ADD E	ADD H	ADD L	ADD M	ADD A	ADC B	ADC C	ADC D	ADC E	ADC H	ADC L	ADC M	ADC A
9	SUB B	SUB C	SUB D	SUB E	SUB H	SUB L	SUB M	SUB A	SBB B	SBB C	SBB D	SBB E	SBB H	SBB L	SBB M	SBB A
A	ANA B	ANA C	ANA D	ANA E	ANA H	ANA L	ANA M	ANA A	XRA B	XRA C	XRA D	XRA E	XRA H	XRA L	XRA M	XRA A
B	ORA B	ORA C	ORA D	ORA E	ORA H	ORA L	ORA M	ORA A	CMP B	CMP C	CMP D	CMP E	CMP H	CMP L	CMP M	CMP A
C	RNZ	POP B	JNZ	JMP	CNZ	PUSH B	ADI d8	RST 0	RZ	RET	JZ	-	CZ	CALL	ACI d8	RST 1
D	RNC	POP D	JNC	OUT	CNC	PUSH D	SUI d8	RST 2	RC	-	JC	IN	CC	-	SBI d8	RST 3
E	RPO	POP H	JPO	XTHL	CPO	PUSH H	ANI d8	RST 4	RPE	PCHL	JPE	XCHG	CPE	-	XRI d8	RST 5
F	RP	POP PSW	JP	DI	CP	PUSH PSW	ORI d8	RST 6	RM	SPHL	JM	EI	CM	-	CPI d8	RST 7

Рисунок 24 – Система команд эмулятора микропроцессора КР580ВМ80

## Вариант

В соответствии с 1 вариантом, требуется найти сумму натурального ряда первых 10 целочисленных элементов начиная с «1» с помощью эмулятора микропроцессора KP580BM80.

## Решение задачи

Таблица 3 – Команды, передаваемые эмулятору

Ячейка ОЗУ	Код команды	Расшифровка команды	Комментарий
0000	1E	MVI E, d8	Загружаем в регистр E число $46_{16}$
0001	0A		Операнд $10_{10} = 0A_{16}$
0002	19	DAD D	Сложение конкатенаций пар регистров DE и HL
0003	1D	DCR E	Уменьшить содержимое регистра E на 1
0004	C2	JNZ adr	Перейти к началу цикла, если содержимое регистра E не 0
0005	02		Младший байт адреса перехода
0006	00		Старший байт адреса перехода
0007	76	HLT	Остановка

Далее, поочередно введем коды команд в нужные ячейки ОЗУ уже в самом эмуляторе CPU850.

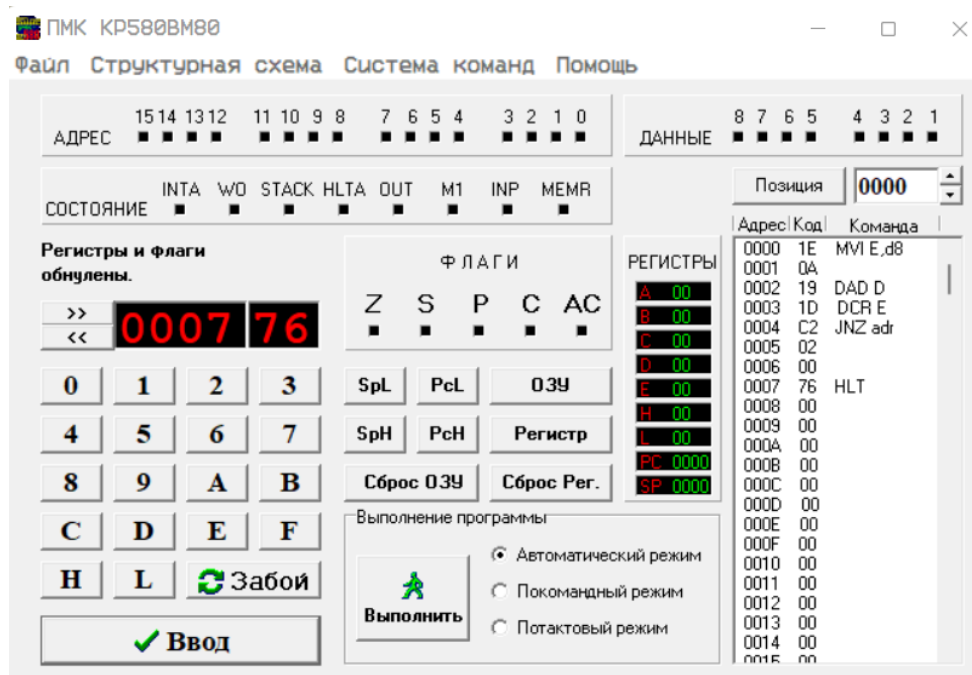


Рисунок 25 – Ввод кодов команд в эмулятор

После нажатия кнопки «Выполнить», в регистре L получим нашу конечную сумму чисел от 1 до 10 в шестнадцатеричной системе счисления.

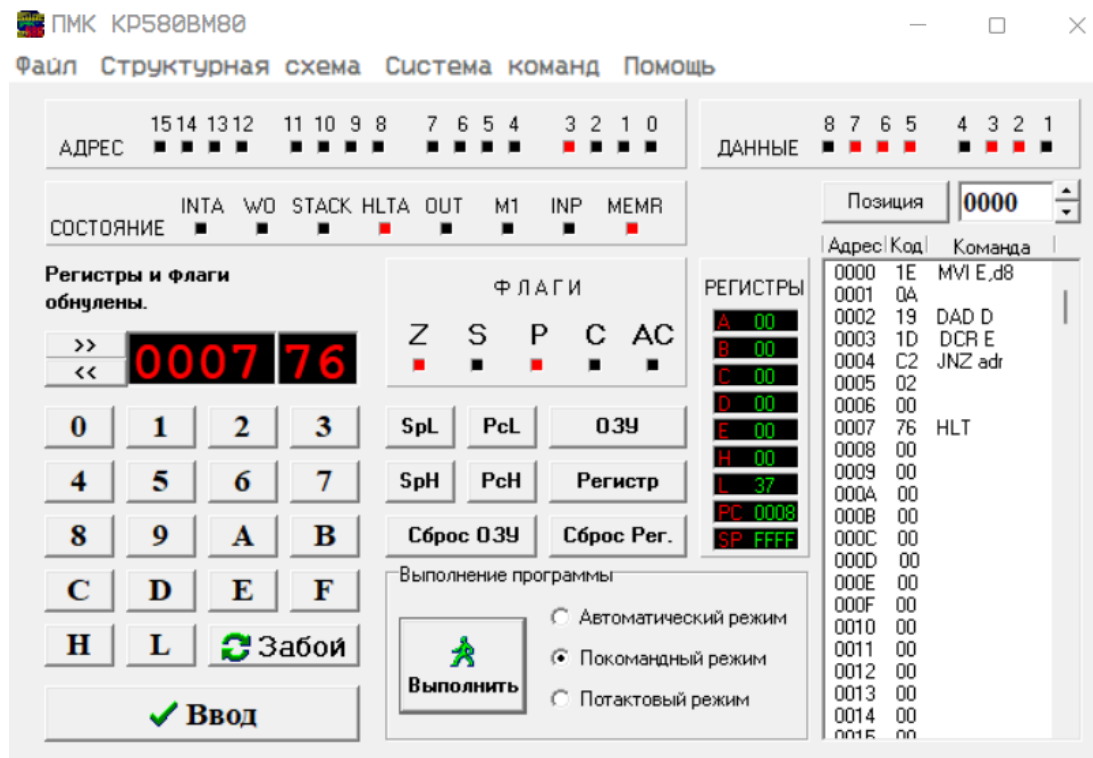


Рисунок 26 – Результат работы эмулятора

Действительно,  $37_{16} = 55_{10}$  является суммой натуральных чисел от 1 до 10, а значит работа эмулятора выполнена корректно.

### Выводы

В данной практической работе мы научились работать с эмулятором CPU580, основанного на микропроцессоре KP580BM80, а также с помощью его команд выполнять задачи.

Задание практической работы было успешно выполнено.