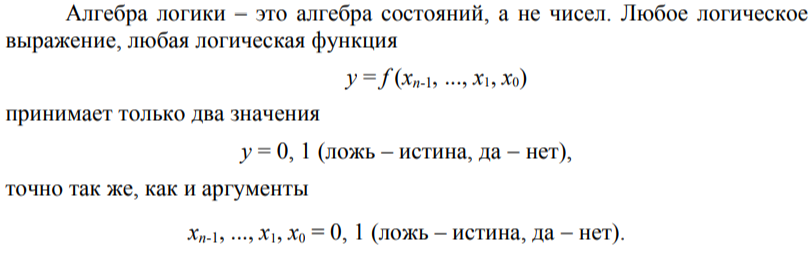
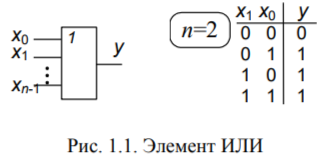
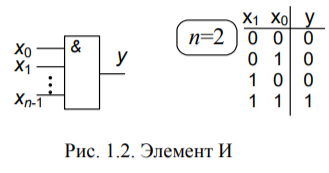
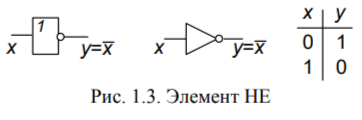
**1) Понятие логической операции или функции. Дизъюнкция, конъюнкция, инверсия. Их обозначение на схемах. Теорема де Моргана. Получение СДНФ и СКНФ для любой логической функции.**



1) логическое сложение (дизъюнкция, ИЛИ)

2) логическое умножение (конъюнкция, И)

3) инверсия (отрицание, НЕ)

**СДНФ** — совершенная дизъюнктивная нормальная форма формулы.

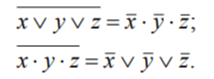
СДНФ представляет функцию в виде дизъюнкции всех комбинаций значений переменных, при которых функция принимает значение 1 (истина).

Пример: Пусть F(A, B) = A∨B, тогда её СДНФ: (A∧¬B)∨(¬A∧B)∨(A∧B).

СКНФ — совершенная конъюнктивная нормальная форма

СКНФ представляет функцию в виде конъюнкции всех комбинаций значений переменных, при которых функция принимает значение 0 (ложь), с инверсией для каждой переменной, участвующей в этой комбинации.

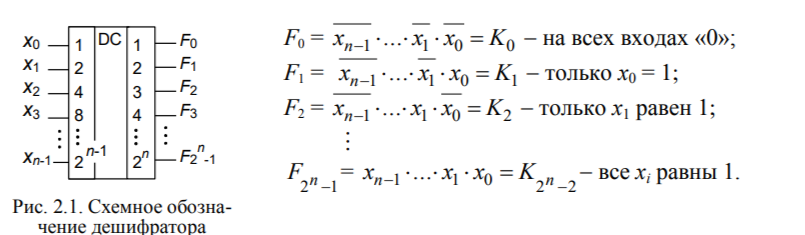
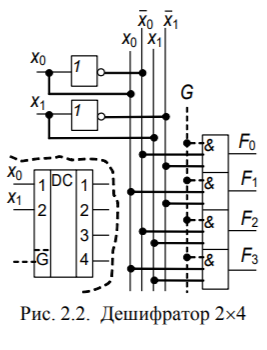
Пример: Пусть F(A, B) = A∧B, тогда её СКНФ: (A∨¬B)∧(¬A∨B)∧(¬A∨¬B).

****Теорема де Моргана

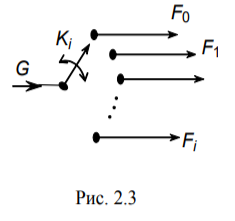
**2) Дешифратор, демультиплексор, мультиплексор. Устройство и назначение. Их обозначение на схемах.**

**Дешифратор**

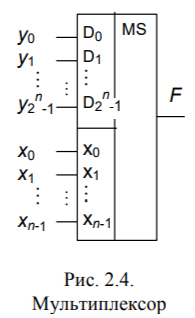
дешифратор можно рассматривать как преобразователь двоичного кода

****

В этом случае на выходах Fi будет результат дешифрации только при условии G = 1. Если G = 0, то всегда и на всех выходах будут логические нули. Такое устройство относится к классу стробируемых дешифраторов

Если вход G принять за информационный, а двоичный код считать кодом переключения выходов (рис. 2.3), то получаем устройство, функция которого состоит в передаче цифровых сигналов по одному из нескольких, задаваемых кодом направлений. Это функция демультиплексора, являющегося разновидностью коммутаторов цифровых сигналов.

**Мультиплексор**

мультиплексор также выполняет функцию коммутации, но противоположную той, которая свойственна демультиплексору

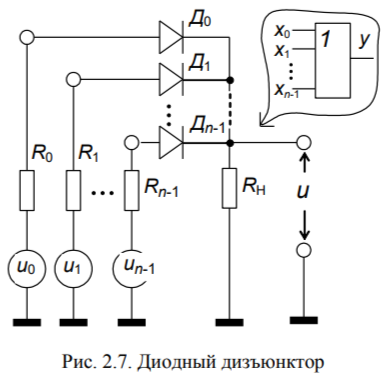
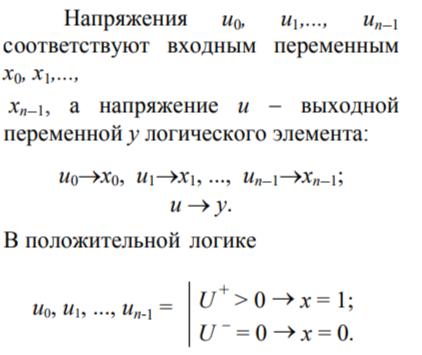
коммутация цифровых сигналов, идущих от n источников, на одну информационную линию F

Подаваемые на мультиплексор (рис. 2.4) сигналы разбиты на две группы. Одна из них образована варьируемыми комбинациями двоичного кода

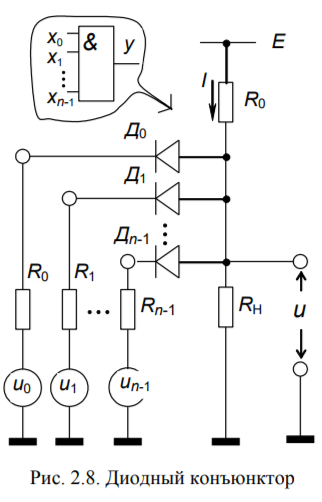
Другая − фиксированной комбинацией значений

**3) Диодные дизъюнкторы и конъюнкторы. Устройство, принцип работы.**

Диодный дизъюнктор в положительной логике выполняется операция логического сложения (операция ИЛИ)

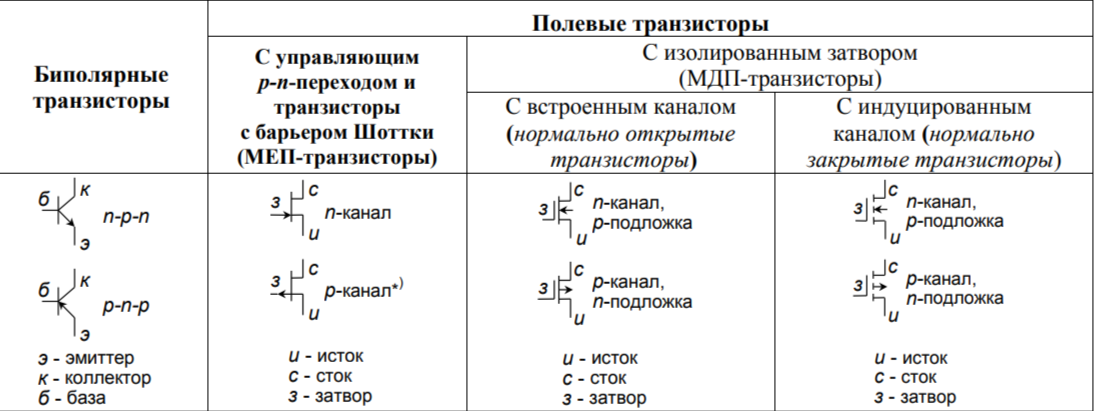
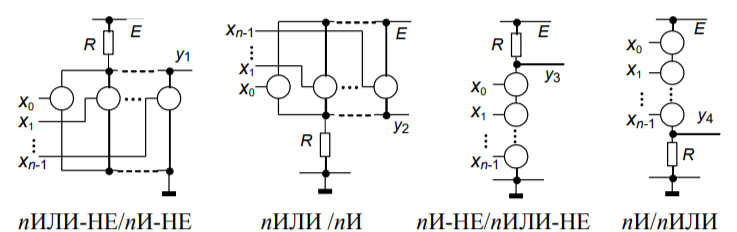
 

Диодный дизъюнктор представляет собой логический элемент, состоящий из нескольких диодов, соединенных параллельно, таким образом, что при наличии хотя бы одного "1" на входе, на выходе будет присутствовать "1" (используется в логике "пассивного уровня"). Когда напряжение присутствует хотя бы на одном из входов, соединённом с диодом, то это напряжение будет присутствовать и на выходе. Диодные дизъюнкторы в положительной логике выполняется операция И

Диодный конъюнктор (И-элемент на диодах):

Диодный конъюнктор включает в себя несколько диодов, соединенных последовательно, чтобы на выходе получить "1" (высокий уровень напряжения), все входы должны быть на "1".

смена логики на отрицательную превращает конъюнктор в дизъюнктор

 **4) Классификация биполярных и полевых транзисторов, их внутренне устройство, обозначения на схемах**.

Биполярные транзисторы делятся на два основных типа: NPN и PNP.

1. NPN-транзистор

Внутренне устройство NPN-транзистора состоит из трех мажоритарных зон: две "N"-типа зоны, между которыми находится "P"-тип зона. Эти зоны образуют два перехода: база-эмиттер (BE) и база-коллектор (BC). При этом, NPN транзисторы управляются положительным напряжением на базе по отношению к эмиттеру.

2. PNP-транзистор

В PNP-транзисторе порядок типов проводимости противоположен NPN-транзистору: сначала идет "P"-тип, затем две "N"-типа зоны. Структура электродов устроена так, что PNP-транзисторы управляются отрицательным напряжением на базе относительно эмиттера.

### Полевые транзисторы (FET):

Полевые транзисторы также имеют два основных типа: JFET (p-n-p или n-p-n тип) и MOSFET (Metal-Oxide-Semiconductor Field-Effect Transistor).

1. JFET

Внутренне устройство JFET (p-n-p или n-p-n) представляет собой полупроводниковый кристалл, в котором образуется канал между двумя другими областями, непроводящими. Управление проводимостью канала в JFET осуществляется путем изменения напряжения на затворе.

2. MOSFET

MOSFET сделан на основе диэлектрического слоя оксида между затвором и каналом. Он имеет три вывода: исток (source), сток (drain) и затвор (gate). Управление проводимостью канала осуществляется путем приложения напряжения на затвор.

**5) Работа транзисторов в ключевом режиме.**

Работа транзисторов в ключевом режиме связана с использованием их в качестве переключающих устройств, когда они находятся в одном из двух состояний: насыщения или отсечки. Этот режим работы является основой многих цифровых и управляющих устройств, таких как микроконтроллеры, схемы памяти, источники питания и другие.

### Режим насыщения:

Когда транзистор находится в режиме насыщения, он обеспечивает минимальное сопротивление между коллектором и эмиттером (в случае биполярного транзистора) или между стоком и истоком (в случае полевого транзистора). Это состояние соответствует "замкнутому" положению транзистора, и в этом случае транзистор способен пропускать максимальный ток.

### Режим отсечки:

Когда транзистор находится в режиме отсечки, он обеспечивает высокое сопротивление между коллектором и эмиттером (или стоком и истоком), практически ограничивая ток до минимального значения. Это состояние соответствует "разомкнутому" положению транзистора, и в этом случае транзистор не позволяет проходить току.

**6) Устройство простейшего инвертора, дизъюнктора и конъюнктора на транзисторах.**

Простейший инвертор может быть создан с использованием одного транзистора и нескольких резисторов.

Устройство:

- Для простого инвертора на биполярном транзисторе можно использовать npn-транзистор.

- К заголовку и базе транзистора подключаются резисторы.

- Входной сигнал подается на базу транзистора, а нагрузка подключается к коллектору.

Принцип работы:

- Когда входной сигнал на базу транзистора равен 1 (высокий уровень), транзистор находится в открытом (насыщенном) состоянии, и на выходе нагрузки будет 0 (низкий уровень).

- Когда входной сигнал на базу транзистора равен 0 (низкий уровень), транзистор находится в закрытом (отсечке) состоянии, и на выходе нагрузки будет 1 (высокий уровень).

Простейший дизъюнктор на транзисторах:

Простой дизъюнктор может быть создан с использованием нескольких транзисторов и нескольких резисторов.

Устройство и принцип работы:

- В данной схеме у нас два npn-транзистора и источник питания Vcc.

- Когда на вход подается хотя бы один высокий уровень, соответствующий транзистор переходит в насыщенное состояние, и напряжение на соответствующем выходе становится высоким.

### Простейший конъюнктор на транзисторах:

Простой конъюнктор также может быть создан с использованием нескольких транзисторов и резисторов.

Устройство и принцип работы:

- Здесь также используются два npn-транзистора и источник питания Vcc.

- Когда все входы находятся на высоком уровне, все транзисторы находятся в насыщенном состоянии, и напряжение на выходе становится высоким.

**7) Назначение выходных цепей, не отвечающих за выполнение логических операций, в составе логических элементов.**

1. Усиление сигнала:

Во многих логических элементах выходные цепи могут усиливать слабый выходной сигнал, что позволяет подавать этот сигнал на другие элементы или на внешние устройства с минимальными потерями. Это особенно важно в современных электронных системах, где большая часть работ выполняется на микроуровне.

2. Регулирование времени задержки:

Некоторые выходные цепи настраиваются для обеспечения определенных временных задержек в сигналах, что может быть полезным при согласовании логических сигналов в цепях с различными скоростями работы. Иногда это также позволяет улучшить стабильность и согласование времени в системах.

3. Электрическая защита:

В некоторых случаях выходные цепи могут обеспечивать защиту от электрических помех или избыточного напряжения, предотвращая повреждение логических элементов или внутренних устройств. Например, они могут включать схемы ограничения напряжения или фильтрации, чтобы предотвратить повреждение элементов при внешних электрических сбоях.

4. Формирование формы сигнала:

Выходные цепи могут также преобразовывать форму сигнала, делая его более совместимым с другими устройствами или подсистемами, которые могут требовать определенной формы сигнала для своей работы.

5. Управление питанием:

Некоторые выходные цепи могут управлять питанием внешних устройств или даже других частей логического элемента, включая режимы сна, режимы ожидания и другие аспекты управления энергопотреблением.

**8) Понятия пассивной, динамической, активной нагрузки в ключе на транзисторах.**

1. Пассивная нагрузка:

- Пассивная нагрузка представляет собой тип нагрузки, который определяется емкостями, индуктивностями или резисторами и не требует внешнего питания.

- В контексте работы с транзисторами, пассивная нагрузка может представлять собой простой резистивный нагрузочный резистор, индуктивность или емкость, которые подключены к выходу транзистора. Пассивная нагрузка используется для получения выходного сигнала, подавления колебаний и других аналогичных целей.

2. Динамическая нагрузка:

- Динамическая нагрузка обычно обозначает нагрузку, представленную схемой, которую можно изменять или настраивать.

- В контексте транзисторов, динамическая нагрузка может быть электронным устройством, таким как операционный усилитель, который может менять свою реакцию на входной сигнал в зависимости от условий, что предоставляет больше гибкости в управлении процессом.

3. Активная нагрузка:

- Активная нагрузка, напротив, включает в себя элементы, требующие внешнего питания и способные усиливать или регулировать входной сигнал.

- В контексте транзисторов, активная нагрузка обычно представлена усилителями или другими подобными устройствами, которые могут не только получать входной сигнал, но и усиливать его или выполнять другие функции управления

**9) Понятие комплементарных структур. Их назначение в цифровой электронике, преимущества и недостатки по сравнению с другими технологиями. Пример конъюнктора и дизъюнктора на КМДП транзисторах.**

Комплементарные структуры относятся к технологии, в которой используются парные транзисторы типа p-n-p (последовательно p-n-p-n) для формирования логических элементов. Каждый набор из двух транзисторов обычно составляет инверсную пару, где один транзистор является p-канальным, а другой - n-канальным.

Назначение в цифровой электронике:

- Комплементарные структуры (CMOS) широко используются для создания цифровых интегральных схем, таких как микропроцессоры, оперативная память, логические элементы и другие компоненты электроники.

- Они обеспечивают минимальное потребление энергии в статическом состоянии, что делает их идеальным выбором для длительной автономной работы в мобильных устройствах, сенсорных устройствах и других энергоэффективных устройствах.

Преимущества:

1. Энергоэффективность: CMOS потребляет очень небольшое количество энергии в статическом состоянии благодаря отсутствию постоянного тока в ключевом состоянии логических элементов.

2. Высокая скорость работы: CMOS предлагает высокую скорость работы, поскольку транзисторы могут переключаться быстро.

3. Надежность: Области перекрывания в CMOS обычно меньше, что способствует меньшему потреблению энергии и повышению надежности.

Недостатки:

1. Технологические сложности: Производство CMOS технологии требует высокой точности и технологических сложностей, что может сделать их более дорогими и трудоемкими в производстве.

2. Подверженность шумам: в некоторых случаях CMOS может быть более подвержен влиянию различных шумов из-за структуры инверсных пар.

**10) Назначение и устройство элементов с тремя состояниями.**

Элемент с тремя состояниями выхода кроме логического состояния 0 и 1 имеет состояние «отключено», в котором ток его выходной цепи пренебрежимо мал (состояние с высоким выходным импедансом). В это (третье) состояние элемент переводится специальным управляющим сигналом (OE – Output Enable).

### Назначение:

1. Шина данных: наиболее распространенное применение элементов с тремя состояниями — это формирование шины данных, где несколько устройств могут управлять шиной и не вмешиваться друг в друга.

2. Мультиплексирование и демультиплексирование: Элементы с тремя состояниями также используются в системах мультиплексирования и демультиплексирования данных.

3. Управление шинами устройства ввода-вывода: они широко применяются в устройствах ввода-вывода, где ускоряют и упрощают структуру работы шин данных.

**11) Двоичные сумматоры. Устройство одноразрядных и многоразрядных сумматоров.**

Двоичный сумматор — это комбинационное устройство, которое выполняет операцию сложения для двух двоичных чисел. Они широко используются в цифровых системах, таких как процессоры, схемы сравнения, схемы управления и т. д. Позволь мне рассказать о структуре одноразрядного и многоразрядного двоичных сумматоров.

Одноразрядный двоичный сумматор:

Одноразрядный двоичный сумматор выполняет сложение двух одноразрядных двоичных чисел (A и B) вместе с учетом возможного внесения переноса из предыдущего бита.

**Устройство:**

- Одноразрядный сумматор состоит из двух входов A и B, входа для внешнего переноса C<sub>in</sub> (старший перенос) и двух выходов - суммы S и переноса C<sub>out</sub> (младший перенос).

**Принцип работы:**

- Выходная сумма (S) для двух битов A и B рассчитывается по модулю 2 (исключающее ИЛИ) операции A ⊕B ⊕C<sub>in</sub>.

- Перенос (C<sub>out</sub>) рассчитается как A AND B OR (A ⊕B) AND C<sub>in</sub>.

Многоразрядный двоичный сумматор:

Многоразрядный сумматор принимает два n-разрядных двоичных числа (A<sub>n-1</sub>...A<sub>0</sub> и B<sub>n-1</sub>...B<sub>0</sub>) и выполняет их сложение.

**Устройство:**

- Он состоит из нескольких одноразрядных сумматоров и логики управления для обработки взаимодействия переносов между разрядами.

**Принцип работы:**

- Выполняется сначала сложение младших разрядов A<sub>0</sub> и B<sub>0</sub> с учетом C<sub>in</sub> (входной перенос).

- При этом каждый последующий старший разряд складывается с учетом выходного переноса предыдущего разряда.

**Примечание:**

- Многоразрядные сумматоры обычно имеют возможность преобразования в сумматор/вычитатель, используя инверсию второго входного числа и подачей логической "1" в качестве входа внешнего переноса.

**12) Матричные умножители. Устройство множительно-суммирующего блока.**

Устройство множительно-суммирующего блока:

Множительно-суммирующий блок или, как его иногда называют, умножительно-аккумуляторный блок, состоит из умножителя (Multiplier) и сумматора (Accumulator).

1. Умножитель (Multiplier):

- Умножитель обычно состоит из параллельно работающих умножителей для каждого бита входной последовательности.

- Для двоичных операций умножение сводится к сложению этих частичных произведений.

2. Сумматор (Accumulator):

- Сумматор принимает и суммирует все результаты умножений, представленные в виде частичных сумм, чтобы получить итоговый выход.

Принцип работы:

1. Умножение:

- Входные сигналы подаются на умножитель, где они умножаются на соответствующие коэффициенты.

- Полученные частичные произведения затем передаются на сумматор.

2. Сложение:

- Сумматор внутри множительно-суммирующего блока выполняет сложение частичных произведений, произведенных умножителем, для получения итогового результата умножения.

Применение:

Множительно-суммирующие блоки встречаются в матричных умножителях, потому что они обрабатывают каждую пару элементов входной матрицы для вычисления каждого элемента выходной матрицы

**13) Программируемые логические матрицы (ПЛМ). Назначение, устройство. Реализация МДНФ любой логической функции на ПЛМ.**

Назначение и устройство:

1. Назначение:

- ПЛМ используются для реализации логических функций, что позволяет их быструю и гибкую конфигурацию для выполнения различных задач, таких как обработка сигналов, управление данными, арифметические операции и т. д.

2. Устройство:

- Каждый элемент ПЛМ представлен комбинацией транзисторов и других элементов, обеспечивающих реализацию комбинационной логики.

- Программируемые соединения позволяют настраивать ПЛМ для выполнения конкретных функций, что делает их универсальными и гибкими.

### Реализация МДНФ на ПЛМ:

1. МДНФ (минимизированная дизъюнктивная нормальная форма) - это способ представления логической функции в виде конъюнкции минтермов, которая, когда она приведена к канонической форме, соответствует минимальному набору таких конъюнкций, которые истинны для заданной функции.

2. Реализация:

- Для реализации МДНФ любой логической функции на ПЛМ необходимо выполнить программирование конфигураций ПЛМ таким образом, чтобы каждый минтерм был представлен соответствующим набором соединений.

- Для этого используется процесс настройки переключателей или специальных устройств, которые могут программировать связи между элементами ПЛМ.

- Настройка проводится в соответсвии с МДНФ, т.е. включение тех элементов, которые соответствуют истинным минтермам функции.

**14) Базовые матричные кристаллы (БМК) и программируемые логические интегральные схемы (ПЛИС). Что это, чем отличаются, зачем нужны?**

Базовые матричные кристаллы (БМК):

Что это:

- Базовый матричный кристалл (БМК) представляет собой интегральную схему, которая содержит основные микросхемы (например, И-НЕ, ИЛИ, НЕ, "Исключающее ИЛИ" и др.), основанные на металл-оксид-полупроводниковой (MOS) технологии.

Назначение:

- БМК используются для создания основных комбинационных логических схем в цифровой электронике.

Программируемые логические интегральные схемы (ПЛИС):

Что это:

- Программируемая логическая интегральная схема (ПЛИС) представляет собой высокодисперсный интегральный элемент, который может быть программирован для конфигурации как логических, так и комбинационных схем на микросхеме после ее производства. Она состоит из большой сети программироваемых логических блоков (ПЛБ).

Назначение:

- ПЛИС позволяют инженерам проектировать цифровые схемы различной сложности, перенося конфигурацию на чип, что делает их очень гибкими и универсальными.

Различия:

1. Архитектура:

- БМК обычно предлагают фиксированную архитектуру только с базовыми логическими функциями, в то время как ПЛИС предлагают гибкую архитектуру с возможностью программирования специфических логических и комбинационных схем.

2. Гибкость:

- ПЛИС обеспечивают гибкость в выборе конфигурации логических элементов, в то время как БМК имеют фиксированный набор логических элементов.

### Зачем нужны:

- БМК используются для решения базовых комбинационных логических задач.

- ПЛИС применяются в широком спектре задач, начиная от прототипирования и разработки до создания множества разнообразных цифровых устройств, таких как ЦПУ, DSP, микроконтроллеры, ASIC-модули и другие.

**15) Бистабильные ячейки (триггеры). Назначение, классификация. Обозначение на схемах.**

Назначение:

Бистабильные ячейки используются для запоминания одного бита информации. Их основное использование включает:

1. Хранение состояния: Триггеры используются для хранения и обновления состояний в цифровых устройствах, таких как регистры, счетчики, управляющие схемы и другие.

2. Управление сигналами: Они также используются в устройствах, где необходим контроль переключения состояний, таких как таймеры, управляющие устройства и многие другие.

Классификация:

1. RS-триггеры:

- Их можно считать самыми базовыми бистабильными ячейками. Они используются в случаях, когда необходимо хранить состояние.

- Используют два входа: R (reset) и S (set), которые устанавливают или сбрасывают выход ячейки в зависимости от активности сигналов.

2. D-триггеры:

— Это бистабильные ячейки синхронного типа, где данные хранятся только на фронте тактового сигнала.

- Имеют один вход данных (D) и сигнал такта (clock), используемый для записи данных.

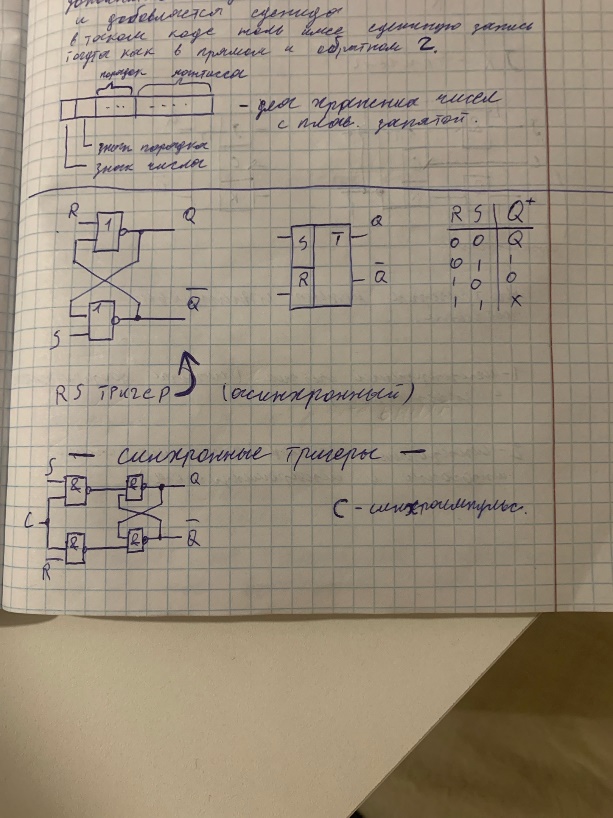
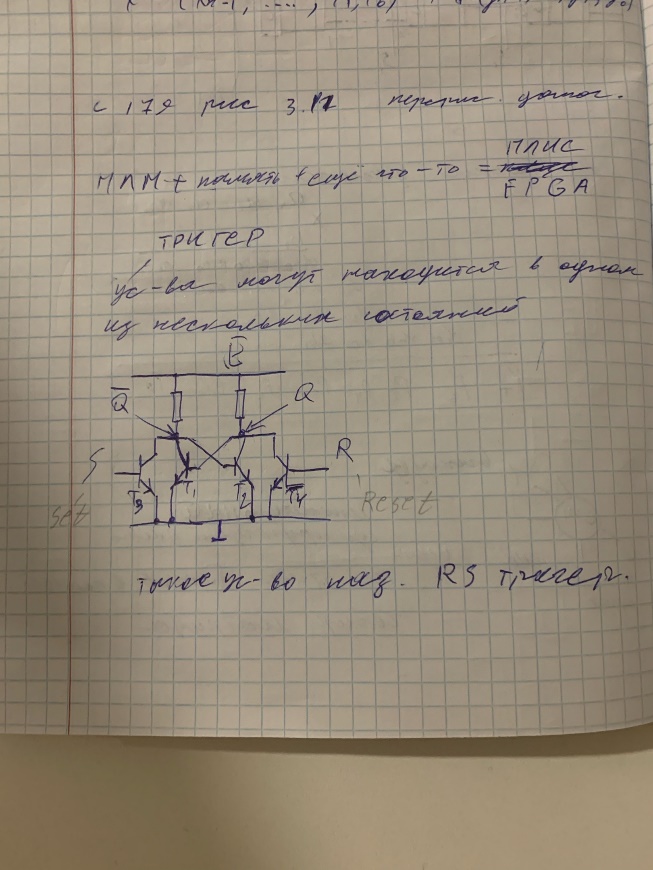
3. JK-триггеры:

— Это бистабильные ячейки, которые представляют собой улучшенную версию RS-триггера, обеспечивающую более надежное поведение при некоторых комбинациях входных сигналов.

- Имеют три входа: J (set), K (reset), и сигнал такта (clock).

4. T-триггеры:

- Имеют один вход T, который представляет собой "инвертированный" вход: если сигнал на входе T равен 1, состояние триггера инвертируется.

**16) Транзисторная схема асинхронного RS-триггера. Построение асинхронного и синхронного RS-триггера из логических элементов.**

**17) Назначение синхронизации при работе триггеров и других логических схем. Проблемы, возникающие при работе цифровых схем без синхронизации.**

Назначение синхронизации:

1. Координация работы элементов: Синхронизация требуется для координации работы триггеров и других логических элементов в цифровых схемах, чтобы обеспечить правильный порядок выполнения операций и избежать конфликтов при обращении к данным.

2. Предотвращение метастабильности: Синхронизация помогает уменьшить вероятность возникновения метастабильного состояния в цифровых схемах при работе с асинхронными сигналами.

3. Управление тактированием: Синхронизация позволяет эффективно управлять тактовыми сигналами, что является ключевым фактором при работе с синхронными цифровыми системами.

Проблемы при работе без синхронизации:

1. Метастабильность: При работе с асинхронными сигналами без синхронизации возникает риск метастабильности, когда триггер или элемент не удается установить определенное состояние вовремя, что может привести к непредсказуемым результатам.

2. Нарушение порядка выполнения операций: Без синхронизации работы цифровых схем могут возникнуть проблемы с порядком выполнения операций, что может привести к неправильным результатам и ошибкам.

3. Шумы и помехи: В асинхронных системах сигналы могут подвергаться воздействию шумов и помех, что делает работу с данными менее надежной и предсказуемой.

**18) Схема параллельного (статического) регистра и последовательного (сдвигающего) регистра. Их назначение.**

начнем с параллельного (статического) регистра. Этот тип регистра используется для хранения битовых данных. Он имеет несколько входов данных, обычно обозначаемых как D<sub>0</sub>, D<sub>1</sub>, ..., D<sub>n</sub>, и один общий вход тактового импульса (обычно CLK), который синхронизирует операцию записи в регистр.

Когда поступает тактовый сигнал, данные на всех входах D<sub>0</sub>...D<sub>n</sub> записываются одновременно в регистр. Это позволяет осуществлять одновременную загрузку всех битов регистра. Параллельные регистры широко используются в цифровых системах, таких как микропроцессоры, ввод-вывод, управление памятью и другие системы.

Теперь перейдем к последовательному (сдвигающему) регистру. Этот тип регистра используется для последовательной передачи данных по цепи. Он обычно имеет один вход данных (D) и два сигнала управления: тактовый сигнал (CLK) и сигнал сдвига (SHIFT). Когда активируется тактовый сигнал, данные на входе D записываются в первый бит регистра, а все остальные биты сдвигаются на одну позицию вправо (вниз). Это позволяет осуществлять последовательную передачу данных в регистре.

Назначение последовательного регистра включает использование его для хранения данных, которые должны быть переданы последовательно, например, в цепи передачи данных (например, при использовании последовательного интерфейса SPI) или для реализации сдвиговых операций в цифровых устройствах.

**19) Двоичные счётчики. Устройство и назначение. Схемы последовательного и параллельного счётчиков.**

двоичные счётчики применяются во всевозможных областях, от деления частоты до создания секвенсоров в цифровой технике и микроконтроллерах. А ещё их используют для адресации памяти и управления периферийными устройствами.

Устройство

Устройство двоичного счётчика обычно состоит из набора флип-флопов, которые могут сдвигаться от одного состояния к другому в соответствии с тактовым импульсом. Когда один флип-флоп переходит от "1" к "0", он передаёт сигнал следующему флип-флопу, что и создаёт двоичный счёт

**20) Машины состояния. Что это такое, каким образом описываются? Классификация машин состояния. Какие задачи они могут решать?**

Описание машин состояния и их классификация

Машины состояния обычно описываются с помощью конечных автоматов, которые классифицируются на:

1. Конечные автоматы Мили - где выходные сигналы ассоциированы с переходами между состояниями.

2. Конечные автоматы Мура - где выходные сигналы ассоциированы с текущим состоянием.

Также их можно классифицировать по сложности (конечные автоматы, конечные автоматы с выходом, конечные автоматы с двумя лентами, и т.д.) и применению (регулярные языки, управление аппаратурой, микропрограммирование и другие).

Решаемые задачи

Машины состояния широко применяются для моделирования, анализа и реализации различных систем. Они могут использоваться для:

- Управления (автоматические двери, лифты, умные устройства и т.д.)

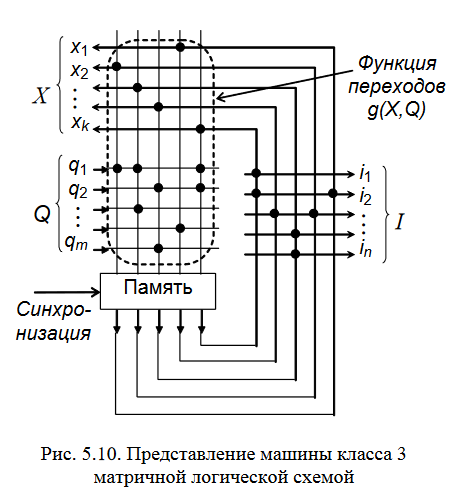
- Распознавания (распознавание речи, обработка сигналов и изображений)

- Протоколов связи (реализация протоколов передачи данных)

- Моделирования поведения (моделирование бизнес-процессов, игровых сценариев и т.д.)

- Дизайна электронных систем (управление состояниями цифровых систем)

**21) Пример реализация машины состояния на матричной логике.**



**22) Понятие микроопераций и микрокоманд.**

Микрооперации

Микрооперации представляют собой базовые операции, выполняемые в процессоре или других цифровых устройствах. Это операции с данными, которые могут быть выполнены за один такт. Например, это может быть операция сложения двух чисел, передачи данных из одного регистра в другой, выполнения логических операций и т.д.

Микрокоманды

Микрокоманды используются для управления выполнением микроопераций и обычно представляют собой битовые комбинации, которые кодируют операции управления в процессоре. Каждая микрокоманда может управлять выполнением нескольких микроопераций. То есть, микрокоманда содержит управляющую информацию о том, какие микрооперации должны быть выполнены в данном такте.

**23) Запоминающие устройства. Назначение и классификация, Основные структуры адресных запоминающих устройств.**

Назначение и классификация

Запоминающие устройства (или память) предназначены для хранения, чтения и записи данных. Они классифицируются, в основном, по способу хранения информации:

1. Постоянное запоминающее устройство (ROM): Информация в ROM устройствах может быть записана один раз в процессе производства и после этого хранится постоянно.

2. Оперативное запоминающее устройство (RAM): RAM позволяет хранить и изменять данные в процессе работы устройства, но информация в нем теряется при отключении питания.

3. Flash-память: обладает возможностью как чтения, так и перемаркировки информации, что делает её гибридом между ROM и RAM, сочетая в себе их преимущества.

Основные структуры адресных запоминающих устройств

Адресное запоминающее устройство способно хранить информацию под определенными адресами и обеспечивает доступ к этой информации по запросу. Они включают в себя:

1. Регистры: Самая быстрая, но емкая форма памяти в процессоре. Используется для хранения промежуточных результатов и вспомогательных данных.

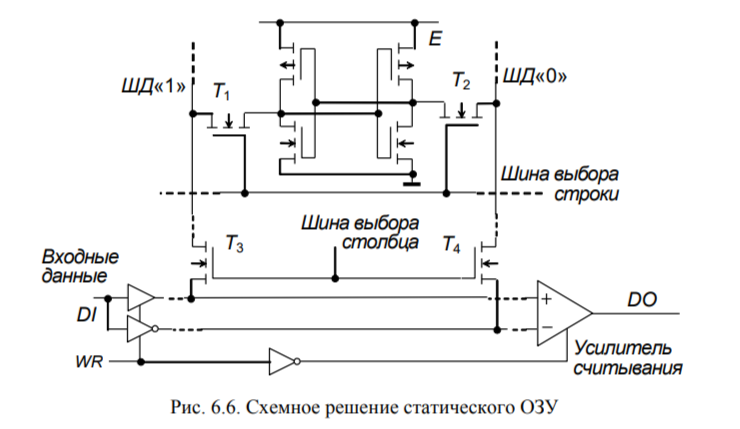
2. Кэш-память: представляет собой небольшой объем быстрой памяти, используемый для хранения данных, наиболее часто используемых процессором.

3. Динамическая RAM (DRAM): Один из наиболее распространенных типов оперативной памяти. Требует перезаписи (refresh) данных периодически из-за утечки заряда с конденсаторов.

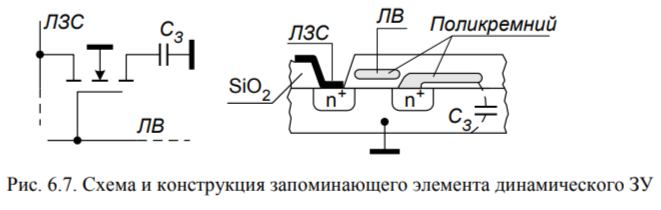
4. Статическая RAM (SRAM): использует триггеры для хранения информации, поэтому не требует перезаписи данных. Поэтому она быстрее, чем DRAM, но и более дорога и имеет больший размер.

**24) Оперативные запоминающие устройства. Динамические и статические (DRAM и SRAM): что это такое, и чем отличаются? Строение одного запоминающего элемента статической ОЗУ. Упрощенная схема одного элемента динамического ОЗУ.**

Адресные ЗУ делятся на статические и динамические оперативные запоминающие устройства (ОЗУ, или Random Access Memory – RAM) и память 222 типа постоянных ЗУ (ПЗУ, или Read-Only Memory – ROM)

Статические запоминающие устройства относятся к числу наиболее быстродействующих. В частности, они широко используются в кэш-памяти, которая при сравнительно малой емкости должна иметь максимальное быстродействие. Статические ОЗУ (SRAM), как правило, имеют структуру 2DM, часть их при небольшой информационной емкости строится по структуре 2D

Запоминающий элемент двумерного ЗУ на КМОП-транзисторах (рис. 6.6) представляет собой RS-триггер с транзисторами выборки Т1 и Т2. При обращении к ЗЭ активизируются сигналы выбора столбца и выбора строки, которые открывают каналы транзисторов Т1, Т2 и Т3, Т4. В режиме записи сигнал WR = 1 переводит входные буферы в состояние прямой (для одного) и инверсной (для другого) передачи информационного сигнала DI на шины данных ШД«0» и ШД«1» выбранного столбца. Передаваемый по линиям ШД«0» и ШД«1» сигнал воспринимается той ячейкой запоминающего массива, у которой открытыми окажутся транзисторы Т1 и Т2. Поступивший через эти транзисторы бифазный сигнал установит триггер ячейки в соответствующее DI состояние.

В динамических ЗУ (DRAM) данные хранятся в виде зарядов емкостей МОП-структур и основой ЗЭ является просто конденсатор небольшой емкости. Такой ЗЭ значительно проще триггерного, что позволяет разместить на кристалле намного больше ЗЭ (в 4...5 раз) и обеспечивает динамическим ЗУ максимальную емкость. Вместе с тем конденсатор неизбежно теряет со временем свой заряд и хранение данных требует их периодического (через несколько миллисекунд) восстановления.

В режиме хранения транзистор выборки заперт. При выборке данного ЗЭ на затвор по линии выборки (ЛВ) подается напряжение, отпирающее транзистор. Запоминающая емкость через проводящий канал подключается к 228 линии записи-считывания, оказывая тем самым влияние на потенциал ЛЗС. При записи потенциал линии записи-считывания передается на конденсатор, определяя его состояние.

**25) Постоянные и перезаписываемые элементы энергонезависимой памяти. Назначение и классификация. Устройство транзисторов, использующихся в энергонезависимой памяти.**

Энергонезависимая память (EEPROM - Electrically Erasable Programmable Read-Only Memory) представляет собой тип памяти, который сохраняет свои данные даже после отключения питания. Это позволяет энергонезависимой памяти сохранять информацию, даже если устройство, в котором она находится, не подключено к источнику питания.

Постоянные и перезаписываемые элементы энергонезависимой памяти:

1. Постоянная - ROM (Read-Only Memory): данные записаны один раз, не могут быть изменены. Применяются для хранения постоянных инструкций или данных.

2. Перезаписываемая - EEPROM (Electrically Erasable Programmable Read-Only Memory): данные могут быть записаны, изменены и стерты электрическим способом. Этот тип памяти может быть перепрограммирован несколько раз.

Устройство транзисторов, использующихся в энергонезависимой памяти:

EEPROM использует специальные EEPROM-транзисторы, которые являются основными элементами памяти. Каждый EEPROM-транзистор обычно состоит из полевого транзистора и запирающего (контроля) транзистора. Это создает возможность индивидуальной записи и стирания каждой ячейки памяти. Программирование и стирание EEPROM-транзисторов осуществляется путем приложения определенных напряжений и токов через электроды.

**26) Центральный процессор (ЦП). Назначение, строение (архитектура), принцип (алгоритм) работы**.

Центральный процессор (ЦП) - это основной вычислительный компонент компьютера, который выполняет инструкции программ и управляет остальными устройствами. Он является "мозгом" компьютера, обрабатывая данные и инструкции, выполнение математических и логических операций, а также координируя работу различных устройств.

Строение:

Центральный процессор выполняет следующие действия:

* загрузка/сохранение регистров и результатов операций в памяти;
* присваивание, логические и арифметические операции;
* безусловные и условные переходы, циклы;
* безусловные и условные вызовы процедур;
* ввод/вывод

1. Устройство регистров: ЦП содержит набор регистров для временного хранения данных и адресов. Регистры могут использоваться для выполнения операций над данными, хранения адресов памяти и управления выполнением программы.

2. Арифметико-логическое устройство (ALU): осуществляет математические и логические операции, такие как сложение, вычитание и логические действия, например, AND, OR, NOT.

3. Устройство управления: осуществляет управление работой ЦП, включая выполнение инструкций, чтение/запись данных в память, обмен данными с другими устройствами.

4. Часы: ЦП имеет встроенные часы, которые управляют таймингом выполнения инструкций, чтобы обеспечить синхронизацию работы всех компонентов.

Принцип работы:

1. Исполнение команд: ЦП последовательно исполняет инструкции, которые хранятся в памяти. Каждая инструкция описывает операцию, которую нужно выполнить (например, загрузка данных из памяти, выполнение операции с данными, сохранение результата).

2. Извлечение и декодирование: Процессор извлекает инструкцию из памяти и декодирует её, понимая, какую операцию нужно выполнить.

3. Исполнение: ЦП выполняет операцию, описанную в инструкции, используя свои регистры, ALU и управляющие сигналы.

4. Обмен данными: Процессор может обмениваться данными с памятью и внешними устройствами для получения и передачи информации.

Эти шаги повторяются многократно в течение работы компьютера, позволяя выполнять сложные программы и обрабатывать большое количество информации.

**27) Арифметико-логическое устройство (АЛУ). Что это такое, и зачем нужно? Что подаётся на вход АЛУ?**

Арифметико-логическое устройство (АЛУ) — это часть центрального процессора (ЦП), которая выполняет арифметические и логические операции над данными. Оно является одной из ключевых частей процессора, так как именно здесь происходят вычисления, необходимые для обработки информации компьютером.

АЛУ получает на вход операнды, над которыми должны быть выполнены операции. Операнды - это данные, над которыми производятся вычисления. Эти данные могут быть целыми числами, числами с плавающей запятой, битовыми последовательностями и т.д. Также АЛУ получает на вход управляющие сигналы, которые определяют, какую операцию следует выполнить.

Входные данные для АЛУ могут включать:

1. Данные из регистров процессора: Это могут быть числа, адреса памяти, или временные данные, которые используются для выполнения операций.

2. Константы: Фиксированные значения, которые могут быть использованы для выполнения операций.

3. Результаты предыдущих операций: Некоторые операции в АЛУ могут быть последовательными, и результат одной операции может быть использован в качестве входного значения для другой операции.

Операции, которые могут выполняться в АЛУ, включают в себя:

1. Арифметические операции, такие как сложение, вычитание, умножение и деление.

2. Логические операции, такие как И, ИЛИ, НЕ, и сдвиги битов.

3. Сравнение чисел.

4. Другие операции, зависящие от конкретной архитектуры и назначения процессора.

**28) Регистры ЦП. Что это такое, и зачем нужны? Классификация. Перечислите основные регистры, их назначение.**

Регистры ЦП (центрального процессора) - это маленькие и очень быстрые встроенные хранилища для временного хранения данных и адресов. Они используются непосредственно процессором для хранения промежуточных результатов вычислений, адресов оперативной памяти, и другой временной информации.

Зачем нужны регистры ЦП:

1. Ускорение обработки данных: Регистры находятся непосредственно внутри самого процессора, что делает доступ к данным в регистрах быстрее, чем доступ к памяти.

2. Хранение операндов: Операции над данными выполняются над данными, хранящимися в регистрах.

3. Хранение адресов: Регистры используются для хранения адресов памяти, что позволяет процессору обращаться к нужным данным в памяти.

Классификация регистров:

1. Регистры общего назначения (General Purpose Registers): Эти регистры используются для хранения данных и адресов, а также для выполнения различных операций. Они являются наиболее распространенными и широко используются в процессорах. Примеры: регистры AX, BX, CX, DX в архитектуре x86.

2. Регистры индексов (Index Registers): Используются для хранения адресов для доступа к данным в памяти. Примеры: регистры SI, DI в архитектуре x86.

3. Регистры указателей (Pointer Registers): Содержат указатели на различные структуры данных. Пример: регистр SP (стековый указатель) в архитектуре x86.

4. Регистры сегментов (Segment Registers): Используются для хранения адресов базы различных сегментов памяти. Примеры: CS (Code Segment), DS (Data Segment), SS (Stack Segment), ES (Extra Segment) в архитектуре x86.

5. Флаговые регистры (Flags Register): Хранят флаги состояния процессора и результата предыдущих операций. Пример: регистр FLAGS в архитектуре x86.

Основные регистры и их назначение в архитектуре x86:

- Регистр AX (и его расширения EAX, RAX): Используется для хранения результатов арифметических операций и данных.

- Регистр BX (и его расширения EBX, RBX): Используется в качестве базового регистра.

- Регистр CX (и его расширения ECX, RCX): Используется в качестве счетчика в циклах.

- Регистр DX (и его расширения EDX, RDX): Используется для хранения данных и ввода-вывода.

- Регистр SI: Регистр источника (Source Index) для работы с строками.

- Регистр DI: Регистр цели (Destination Index) для работы с строками.

- Регистр SP: Стековый указатель (Stack Pointer), используется для управления стеком.

- Регистр BP: Базовый указатель (Base Pointer), используется для адресации данных в стеке и кадрах стека.

- Регистр FLAGS: Содержит флаги состояния процессора (например, флаги переноса, знака, нуля и т.д.).

**29) Команды центрального процессора. Что это такое, и зачем нужны? Классификация.**

Команды центрального процессора (ЦП) являются инструкциями, которые управляют работой процессора в процессе выполнения программ. Они представляют собой набор операций, которые процессор может выполнить, такие как арифметические операции, операции сдвига и сравнения, доступ к памяти, управление вводом-выводом и т.д. Команды ЦП являются ключевыми для выполнения программ и обработки данных.

Зачем нужны команды ЦП:

1. Выполнение операций: Команды ЦП позволяют процессору выполнять различные операции, такие как сложение, вычитание, умножение, деление, сравнение, логические операции и т.д.

2. Управление памятью: Команды позволяют процессору считывать и записывать данные в оперативную память, взаимодействовать с периферийными устройствами и управлять стеком.

3. Управление программой: Команды ЦП позволяют переходить к другим частям программы, выполнять условные и безусловные переходы, а также вызывать подпрограммы.

Классификация команд:

1. Арифметические операции: Команды для выполнения арифметических действий, таких как сложение, вычитание, умножение и деление.

2. Логические операции: Команды для выполнения логических операций, таких как И, ИЛИ, НЕ, сдвиг битов и т.д.

3. Загрузка и сохранение данных: Команды для загрузки данных из памяти в регистры и сохранения данных из регистров в память.

4. Управление переходами: Команды для управления последовательностью выполнения кода, включая условные и безусловные переходы.

5. Управление стеком: Команды для управления стеком, включая помещение и извлечение данных из стека.

6. Управление вводом-выводом: Команды для взаимодействия с периферийными устройствами, такими как чтение и запись в порты ввода-вывода.

**30) Из каких шагов состоит выполнение команды центральным процессором?**

1. Получение команды из памяти

2. Декодирование команды

3. подготовка исходных данных для выполнения инструкции

4. Выполнение команды

5. Обновление состояния регистров и памяти

**31) Системная шина. Что это такое, и зачем нужна? Почему устройства на шине не мешают работе друг другу?**

Системная шина — это коммуникационный канал, который соединяет различные компоненты компьютерной системы, такие как процессор, память, устройства ввода-вывода и другие устройства. Шина позволяет передавать данные и управляющие сигналы между компонентами системы.

Системная шина нужна для обеспечения связи между различными устройствами и компонентами компьютерной системы. Она обеспечивает передачу данных между процессором и памятью, между процессором и устройствами ввода-вывода, а также между устройствами ввода-вывода и памятью.

Устройства, подключенные к системной шине, не мешают друг другу, потому что шина использует специальные протоколы и методы доступа к данным, чтобы обеспечить безопасную и эффективную передачу информации между различными устройствами.

**32) Схема, описывающая простейшую архитектуру компьютера. Какие устройства в ней присутствуют, как они соединены, и каково их назначение?**

Простейшая архитектура компьютера обычно состоит из основных компонентов: центрального процессора (CPU), оперативной памяти (RAM), устройства ввода-вывода (например, клавиатура, мышь, монитор), устройств хранения данных (например, жесткий диск), и системной шины для передачи данных и управляющих сигналов между ними.

Центральный процессор (CPU) - основной вычислительный узел компьютера, который выполняет инструкции и обрабатывает данные.

Оперативная память (RAM) - используется для временного хранения данных и инструкций, к которым процессор имеет быстрый доступ.

Устройства ввода-вывода (например, клавиатура, мышь, монитор) - позволяют пользователю вводить данные в компьютер и получать информацию от компьютера.

Устройства хранения данных (например, жесткий диск) - используются для долгосрочного хранения данных и программ.

Системная шина - обеспечивает связь между центральным процессором, оперативной памятью, устройствами ввода-вывода и устройствами хранения данных.

В простейшей архитектуре эти устройства соединены через системную шину. CPU обращается к оперативной памяти для доступа к данным и инструкциям, а также обращается к устройствам ввода-вывода, чтобы получать данные от пользователей и отображать информацию на экране. Устройства хранения данных используются для хранения данных и программ, которые не используются непосредственно центральным процессором.

**33) Где находятся команды (программы) и данные в компьютере? Каким образом реализовано выполнение находящейся в памяти программы посредством ЦП? Какие регистры используются?**

Команды (программы) и данные в компьютере обычно находятся в оперативной памяти (RAM) и в устройствах хранения данных, таких как жесткий диск или SSD. Программы хранятся в виде инструкций, которые выполняются процессором, а данные могут быть как временно загруженные из внешних источниках (например, клавиатура, сеть) или извлеченные из внешних источников.

Когда процессор должен выполнить программу, он загружает инструкции из памяти. Процессор получает команды из оперативной памяти посредством системной шины и сохраняет их в своем кэше для быстрого доступа. После этого процессор декодирует команды, выполняет операции и обновляет свои регистры и данные в памяти. Процессор использует адреса памяти, инструкции, управляющие сигналы и данные для выполнения программы.

В процессе выполнения программы процессор использует различные регистры для временного хранения данных и адресов. Например:

- Регистр инструкций (Instruction Pointer) - содержит адрес следующей инструкции, которую процессор должен выполнить.

- Регистр адреса (Address Register) - используется для хранения адреса памяти, куда нужно выполнить операцию.

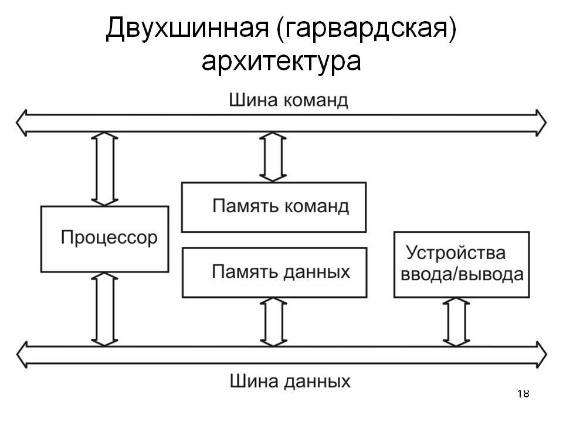
- Регистр данных (Data Register) - для хранения промежуточных результатов и данных, над которыми выполняются операции.

- Регистр флагов (Flags Register) - для хранения информации о состоянии процесса выполнения программы, такой как условия выполнения операций (например, флаги переполнения, нулевого результата и т. д.).

**34) Гарвардская архитектура и архитектура фон Неймана (достаточно общих представлений).**

Архитектура фон Неймана:

* Основная идея архитектуры фон Неймана заключается в том, что команды (инструкции) и данные хранятся в одной общей памяти. Для выполнения операций процессор должен загружать как данные, так и инструкции из одной и той же памяти.
* Архитектура фон Неймана использует один и тот же канал для передачи данных и команд. Это означает, что процессор получает команды и данные через один и тот же канал.

Гарвардская архитектура:

* В гарвардской архитектуре данные и команды хранятся в разных памяти. Это означает, что для хранения и передачи данных и инструкций используются разные шины.
* Такая архитектура позволяет одновременно получать данные и инструкции, что может увеличить производительность.
* - Главный недостаток заключается в том, что для хранения данных и команд может потребоваться больше цепочек памяти (RAM) и процессор должен использовать разные каналы для доступа к данным и командам

**35) Как реализован вызов процедур (подпрограмм)? Понятие стека, принцип его работы.**

Вызов процедур (или подпрограмм) реализуется с использованием стека. Стек - это структура данных, которая работает по принципу Last In, First Out (LIFO), то есть последний элемент, помещенный в стек, будет первым, который будет извлечен.

При вызове процедуры (или функции) в программе, текущее состояние процесса сохраняется на стеке. Обычно на стек помещаются адрес возврата (адрес следующей инструкции после вызова подпрограммы) и текущие значения регистров. После этого управление передается внутрь процедуры, которая выполняет свои задачи, используя регистры и другие ресурсы. Когда выполнение процедуры заканчивается, сохраненные значения извлекаются со стека, возвращая предыдущее состояние процесса, и управление передается обратно в вызывающую программу.

Принцип работы стека следующий:

1. При вызове процедуры текущее состояние процесса (адреса возврата и значения регистров) помещается на вершину стека.

2. Выполняется код процедуры, при этом процедура может использовать регистры и другие ресурсы.

3. По завершении работы процедуры сохраненное состояние восстанавливается, извлекая данные со стека и восстанавливая предыдущее состояние процесса.

4. После восстановления состояния процессор передает управление по адресу, извлеченному из стека.

**36) Назначение механизма прерываний. Понятие аппаратного прерывания. Простейшая реализация аппаратных прерываний через контроллер прерываний.**

Механизм прерываний в компьютерной системе используется для обработки внешних событий, которые могут потребовать немедленного вмешательства центрального процессора. Прерывания позволяют процессору временно остановить выполнение текущей программы, обработать входящее событие (например, сигнал от устройства ввода-вывода или событие таймера) и вернуться к выполнению программы с места, где было прервано выполнение.

Аппаратное прерывание - это вид прерывания, инициируемый аппаратным устройством (например, сетевой адаптер, жесткий диск, таймер и др.), для уведомления процессора о возникновении события, требующего внимания или обработки.

Простейшая реализация аппаратных прерываний включает в себя использование контроллера прерываний. Контроллер прерываний - это устройство, которое управляет входящими аппаратными прерываниями от различных устройств в компьютерной системе. Примером контроллера прерываний является система прерываний Intel 8259A, который широко используется для управления аппаратными прерываниями в системах, построенных на архитектуре x86.

Принцип работы контроллера прерываний заключается в следующем:

1. Аппаратное устройство инициирует прерывание, отправляя сигнал на контроллер прерываний.

2. Контроллер прерываний определяет источник прерывания и устанавливает его приоритет.

3. При возникновении аппаратного прерывания контроллер прерываний посылает сигнал процессору для прерывания выполнения текущей операции и передачи управления обработчику прерывания.

4. Процессор выполняет обработчик прерывания, который обрабатывает входящее событие, выполняет необходимые действия, такие как чтение данных, запись данных, и т.д., и затем возвращает управление прерванной программе.

**37)Реализация условного перехода при выполнении программы с точки зрения системы команд процессора и регистра состояний (флагов).**

Условный переход - это инструкция в ассемблерном или машинном коде, которая позволяет программе принимать решение о том, выполнять ли какое-то действие или переходить к другой части программы на основе определенного условия. Условный переход реализуется с помощью системы команд процессора и регистров состояний (флагов).

Во время выполнения программы, процессор выполняет инструкции, проверяет условия и выполняет переход к другой части программы ("ветвление") в зависимости от значения определенного флага или комбинации флагов.

Процессор использует регистр состояния (или регистр флагов) для хранения информации о завершении выполнения предыдущей инструкции. Флаги — это биты в регистре состояния, которые представляют различные условия, такие как заем, переполнение, нулевое значение, отрицательное значение и др.

CMP AX, BX; сравнить содержимое регистров AX и BX

JE Label; перейти к метке Label, если результат был равен

В этом примере инструкция CMP производит сравнение двух значений, а инструкция JE (jump if equal) выполняет условный переход к метке Label, если результат сравнения был равен.

Когда процессор выполняет инструкцию CMP, он изменяет флаги в регистре состояний в соответствии с результатом сравнения. В данном случае, если в результате сравнения обнаруживается равенство, флаг равенства устанавливается, и при выполнении инструкции JE, процессор передается к метке Label.

**38) Примеры команд ассемблера, перемещающих данные из одной ячейки памяти в другую, арифметических команд, команд условного и безусловного перехода.**

Примеры команд ассемблера, перемещающих данные из одной ячейки памяти в другую:

MOV AX, BX; перемещает содержимое регистра BX в регистр AX

MOV dest, AX; перемещает содержимое регистра AX в ячейку памяти, адрес которой задан в dest

Примеры арифметических команд:

ADD AX, BX; складывает содержимое регистра AX с содержимым регистра BX и сохраняет результат в регистре AX

SUB AX, BX; вычитает содержимое регистра BX из содержимого регистра AX и сохраняет результат в регистре AX

MUL CX; умножает содержимое регистра AX на содержимое регистра CX и сохраняет результат в регистрах DX:AX

Примеры команд условного и безусловного перехода:

CMP AX, BX; сравнивает содержимое регистра AX с содержимым регистра BX

JE Loop; переходит к метке Loop, если предыдущее сравнение показало равенство

JNE Loop; переходит к метке Loop, если предыдущее сравнение показало неравенство

JMP Loop; безусловный переход к метке Loop

**39) Пример реализации программного цикла на ассемблере. То, что на языках высокого уровня реализуется оператором for.**

MOV R12,10 // перемещаем в регистр R12 количество итераций по циклу

MOV R11,0 // перемещаем в регистр R11 0 для окончания

Loop // метка, на которую цикл возвращается

Тело цикла

CMP R12, R11 // сравнивает содержимое регистра R12 с значением R11

JNE Loop // переходит к метке Loop, если предыдущее сравнение показало неравенство

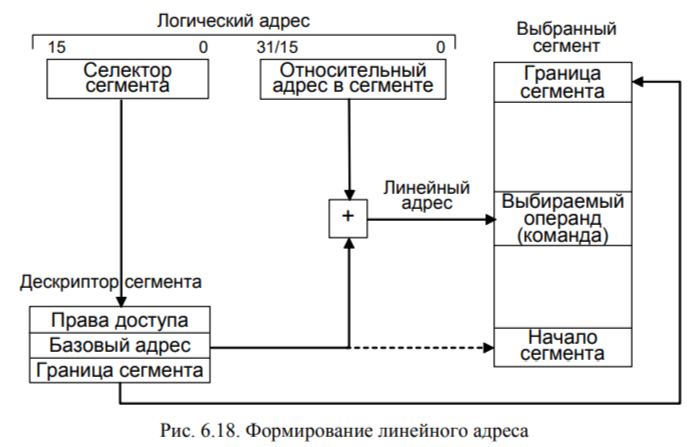
**40) Назначение таймера в работе компьютера**

**41) Механизм виртуальной памяти. Что это, и зачем это нужно? Сегментная и страничная организация памяти (достаточно общих представлений).**

Он позволяет использовать больше памяти, чем физически установлено на компьютере, за счет представления виртуальной памяти как расширения физической.

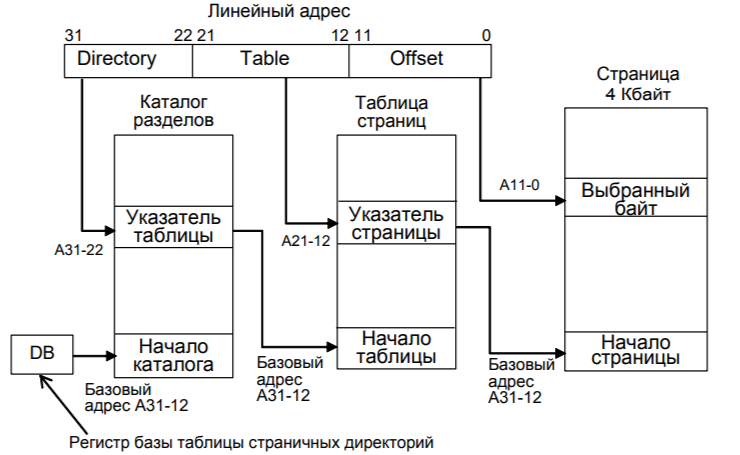
**Сегментная организация памяти**

Сегментная организация памяти представляет собой разбиение виртуальной памяти программы на логически связанные сегменты, такие как код, данные, стек и др. Каждый сегмент может иметь различный размер, что позволяет удобно управлять памятью программы, но может приводить к фрагментации памяти.

В системах с виртуальным режимом адресации память делится на отдельные сегменты. Процессор, обращаясь к памяти, формирует линейный (виртуальный) адрес, исходя из логического адреса, определяемого по содержимому специального регистра – селектора сегмента и относительному адресу. Линейный адрес находится как сумма базового адреса (адреса начала) сегмента и относительного адреса (смещения в сегменте)

В памяти может быть выделено несколько сегментов, например, сегмент кода, сегмент данных, сегмент стека и т.д., Кроме того, в многозадачной среде каждой задаче выделяется свой сегмент памяти. В таком случае желательно иметь несколько регистров для хранения селекторов соответствующих сегментов, что имеет место на практике в процессорах, поддерживающих виртуальную адресацию.

**Страничная организация памяти**

Страничная организация памяти разбивает как физическую, так и виртуальную память на фиксированные блоки фиксированного размера, называемые страницами. Эта организация позволяет эффективно управлять памятью и обеспечивать защиту памяти, а также реализовать механизм подкачки страниц.

Линейный адрес разбивается на три поля: поле каталога разделов – директории (Directory), поле таблицы страниц (Table) и поле адреса в странице (Offset). Базовый адрес каталога разделов задается значением 243 специального регистра DB (Directory Base) процессора. Регистр программно доступен. Его содержимое определяется действиями на стадии инициализации системы и не меняется при ее работе

**42) Что такое операционная система? Из каких модулей она состоит, и какие преимущества с точки зрения написания и исполнения программ она даёт? (Достаточно общих представлений).**

Операционная система (ОС) — это программное обеспечение, которое управляет ресурсами компьютерной системы и предоставляет интерфейс для взаимодействия с пользователем и другими программами.

ОС состоит из нескольких модулей:

1. Ядро (kernel): ядро ОС — это основной компонент, отвечающий за управление ресурсами, планирование задач, управление памятью и вводом-выводом.

2. Драйверы (drivers): драйверы обеспечивают взаимодействие между аппаратными устройствами и операционной системой.

3. Системные библиотеки (system libraries): системные библиотеки предоставляют набор функций и процедур, которые могут быть использованы программами для выполнения различных задач, таких как работа с файлами, сетью и т. д.

4. Интерфейс пользователя (user interface): интерфейс пользователя предоставляет средства для взаимодействия с операционной системой, например, командную строку или графический интерфейс.

Преимущества операционной системы с точки зрения написания и исполнения программ:

1. Управление ресурсами: ОС управляет доступом к ресурсам компьютера, таким как процессор, память и ввод-выводные устройства, что позволяет программам эффективно использовать доступные ресурсы.

2. Абстракция аппаратных устройств: ОС предоставляет абстракцию аппаратных устройств, скрывая сложности и различия между разными устройствами. Это позволяет программистам разрабатывать программы, которые могут работать на разных компьютерных системах без необходимости переписывать код.

3. Управление задачами: ОС позволяет программам выполняться параллельно и управляет их выполнением, планируя их работу и обеспечивая справедливое распределение ресурсов.

4. Обеспечение безопасности: ОС обеспечивает безопасность путем контроля доступа к ресурсам и защиты от вредоносных программ.

**43) За счёт каких модулей операционной системы и аппаратных особенностей компьютера возможна многозадачность на компьютере с одним процессором?**

1. Планировщик задач (Task Scheduler): Этот модуль операционной системы отвечает за планирование и управление выполнением задач. Он определяет, какие задачи будут выполняться в каждый момент времени и каким образом будет распределено процессорное время между ними.

2. Прерывания и таймеры: Аппаратные особенности компьютера, такие как прерывания и таймеры, позволяют операционной системе переключаться между различными задачами. Прерывания позволяют операционной системе реагировать на события, такие как нажатие клавиши или завершение операции ввода-вывода, и переключаться на другую задачу. Таймеры используются для периодического вызова планировщика задач и позволяют ему переключать контекст выполнения между задачами.

3. Контекстное переключение (Context Switching): Когда операционная система переключается с одной задачи на другую, она сохраняет текущее состояние задачи (контекст) и восстанавливает состояние следующей задачи. Это позволяет операционной системе эффективно переключаться между задачами и продолжать их выполнение с того места, где они были приостановлены.

**44) Кэш-память. Что это такое, и зачем нужно? Какие данные туда попадают, и чем это улучшает работу компьютера? (Достаточно общих представлений).**

Кэш-память запоминает копии информации, передаваемой между устройствами (прежде всего между процессором и основной памятью). Она имеет небольшую емкость в сравнении с основной памятью и более высокое быстродействие (реализуется на триггерных элементах памяти).

При чтении данных сначала выполняется обращение к кэш-памяти. Если в кэше имеется копия адресованной ячейки основной памяти, то кэш вырабатывает сигнал Hit (попадание) и выдает данные на общую шину данных. В противном случае сигнал Hit не вырабатывается и выполняется чтение из основной памяти и одновременное помещение считанных данных в кэш.

В основу работы кэш-памяти положен принцип локальности программы или, как еще говорят, гнездовой характер обращений, имея в виду, что адреса последовательных обращений к основной памяти (ОП) образуют компактную группу. При обращении к ОП в кэш-память копируются не отдельные данные, а блоки цифровой информации, включающие те данные, которые с большой степенью вероятности будут использованы в ЦП на последующих шагах работы

Объем кэш-памяти много меньше емкости основной памяти и любая единица информации, помещаемая в кэш, должна сопровождаться дополнительными данными (тегом), определяющими, копией содержания какой ячейки основной памяти является эта единица информации.

Кэш-память используется для хранения часто используемых данных или инструкций, что позволяет процессору более быстро получить к ним доступ, уменьшая задержки при выполнении операций. Это особенно важно в многозадачных системах, где процессор должен быстро переключаться между различными задачами, а также в случаях, когда данные часто обращаются из оперативной памяти или диска.

**45) Микроконтроллеры. Что это такое, зачем нужно, и как устроено? Чем отличаются от персонального компьютера?**

Микроконтроллеры — это маленькие компьютеры, которые используются для управления различными электронными устройствами, их используют, когда не требуется большой вычислительный масштаб

они специализированы для выполнения конкретных задач, часто в режиме реального времени, и обычно работают в устройствах с ограниченными ресурсами

состоят:

* Центральный процессор
* Оперативная память (RAM)
* Постоянное хранилище (Flash или EEPROM)
* Вводно-выводные порты

Отличие в том, что у ПК более широкий спектр задач, следовательно больше мощности.

**46) CISC и RISC машины. Что это такое, чем они отличаются? Их относительные достоинства и недостатки. Важные преимущества RISС архитектуры.**

CISC (Complex Instruction Set Computing) и RISC (Reduced Instruction Set Computing) являются двумя различными архитектурами микропроцессоров, которые отличаются в своем подходе к инструкциям и выполнению операций.

CISC-машину выполняют меньшее количество инструкций, которые более сложны, в то время как RISC-машину выполняют большее количество простых инструкций. CISC часто включает в себя множество различных инструкций, некоторые из которых могут выполнять несколько операций за один шаг.

RISC упрощает инструкции до базовых операций, таких как загрузка, хранение, арифметические операции и передача данных между регистрами.

Преимущества RISC-архитектуры:

1. Простота: RISC-процессоры обычно имеют более простую структуру, что облегчает их разработку, снижает стоимость производства и позволяет достичь более высокой производительности при меньших затратах.

2. Высокая скорость выполнения: из-за простоты инструкций и более легкой оптимизации процессоры RISC обычно могут выполнять инструкции более быстро.

3. Эффективное использование ресурсов: RISC-процессоры обычно могут эффективнее использовать ресурсы, такие как кэш-память и регистры, что позволяет повысить производительность.

Недостатки RISC включают в себя более объемный код, так как более простые инструкции могут потреблять больше места в программе, а также увеличение количества инструкций для выполнения сложных задач.

**47) Конвейеризация при выполнении команд в ЦП.**

Конвейеризация в центральном процессоре (ЦП) представляет собой метод организации выполнения команд, который позволяет одновременно выполнять несколько этапов операции для увеличения производительности и сокращения времени выполнения команды.

Преимущества:

* Разделение задач на отдельные шаги
* Параллельная обработка
* Разделение шин для каждого устройства

Шаги выполнения:

1. Выборка команды (Instruction Fetch): Процессор выбирает следующую команду из памяти и загружает ее в свой внутренний буфер команд.

2. Декодирование команды (Instruction Decode): Выбранная команда декодируется для определения ее типа, операндов и операций, которые нужно выполнить.

3. Выполнение операции (Execute): Процессор выполняет действия, соответствующие операции, указанной в команде. Например, это может быть арифметическая операция, обращение к памяти и т.д.

4. Обращение к памяти (Memory Access): Если команда требует доступа к памяти (например, загрузка или сохранение данных), процессор осуществляет такой доступ.

5. Запись результата (Write Back): Результирующие данные от операции записываются обратно в регистры или в память.