**АЛОВС**

**Погрешности метаматематических операций в цифровых системах. Способы оценки.**

1. **Методы оценки погрешностей математических операций:**
   * **Абсолютная погрешность**: разница между истинным и приближённым значением.
   * **Относительная погрешность**: отношение абсолютной погрешности к истинному значению.
   * **Ошибка округления**: результат ограничений представления чисел в памяти.
2. **Почему при выполнении математических операций в двоичном коде наблюдаются погрешности?**
   * Из-за ограниченности разрядности чисел в компьютере, невозможно точно представить некоторые числа, особенно иррациональные и бесконечно повторяющиеся десятичные дроби.
3. **Источники погрешностей при выполнении операции умножения в двоичном коде:**
   * Ошибка округления из-за ограниченной точности представления чисел.
   * Погрешности при работе с числами с плавающей точкой.
4. **Источники погрешностей при выполнении операции деления в двоичном коде:**
   * Потеря точности при вычислениях с ограниченным числом разрядов.
   * Ошибка округления при делении на числа, которые не могут быть точно представлены в двоичной системе.

**Арифметические операции над двоичными числами. Операция сложения и вычитания в двоичной системе исчисления.**

1. **Общие правила проведения сложений в двоичном коде:**
   * 0 + 0 = 0, 1 + 0 = 1, 1 + 1 = 10 (перенос единицы в следующий разряд).
2. **Общие правила проведения вычитания в двоичной системе:**
   * 0 - 0 = 0, 1 - 0 = 1, 1 - 1 = 0, 0 - 1 = 1 с заимствованием из старшего разряда.

**Операция умножения в двоичной системе исчисления**

1. **Общие правила выполнения умножения чисел в двоичном коде:**
   * Умножение двоичных чисел аналогично умножению десятичных, только с использованием 0 и 1. Применяется метод "сдвигов и сложений".
2. **Почему операции деления и умножения производят обычно в формате с плавающей точкой?**
   * Формат с плавающей точкой позволяет представлять очень большие или очень маленькие числа с хорошей точностью и решает проблемы переполнения и потери точности.

**Деление двоичных чисел**

1. **Общие правила деления двоичных чисел:**
   * Алгоритм деления в двоичной системе аналогичен делению в десятичной, но используется сдвиг влево и вычитание.
2. **От чего зависит количество итераций при делении двоичных чисел?**
   * От количества разрядов в делимом и делителе.
3. **Алгоритм деления с восстановлением остатка:**
   * Деление производится с остатком, который восстанавливается после каждого шага деления.
4. **Преимущество и недостатки деления с восстановлением остатка:**
   * Преимущество: позволяет более точно разделить число.
   * Недостаток: более сложный и ресурсоёмкий процесс.
5. **Алгоритм деления без восстановления остатка:**
   * Деление без восстановления остатка включает только вычитание и сдвиги, без необходимости сохранять остаток.
6. **Преимущество и недостатки деления без восстановления остатка:**
   * Преимущество: проще и быстрее, но может быть менее точным.
   * Недостаток: возможно более низкое качество вычислений.
7. **Методы деления двоичных чисел:**
   * Метод с восстановлением остатка.
   * Метод без восстановления остатка.
   * Метод с использованием битовых операций.

**Операции с двоичными числами в дополнительном и обратном кодах**

1. **Чем отличается алгоритм выполнения сложения в обратном и дополнительном кодах?**
   * В обратном коде сложение чисел выполняется с учётом отрицательных значений, и для получения правильного результата используется инвертирование старшего бита.
2. **Почему современные процессоры выполняют операции вычитания в дополнительном (обратном) коде?**
   * Дополнительный код упрощает операции вычитания, поскольку позволяет преобразовать вычитание в сложение.

**Логические операции с двоичными кодами: логическое суммирование, логическое умножение, логическое отрицание**

1. **Перечислите основные логические операции с двоичными кодами:**
   * Логическое И (AND), Логическое ИЛИ (OR), Логическое НЕ (NOT), XOR (исключающее ИЛИ), NAND, NOR.

19-24. **Провести логические операции для чисел:** - Для выполнения этих операций вам необходимо представить числа в двоичном виде и использовать соответствующие логические операции, описанные выше.

(

**OR**: 110111111

**AND**: 010001000

**NOT (110011100)**: 001100011

**XOR**: 100110111

**NAND**: 101110111

**NOR**: 001000000

)

1. **В чем отличие логического сдвига от арифметического?**
   * Логический сдвиг сдвигает биты числа, заполняя освободившиеся разряды нулями.
   * Арифметический сдвиг сохраняет знак числа, т.е. заполняет старшие биты значением знакового бита.

**26. Какие есть альтернативные названия логических операций: логическое суммирование, логическое умножение, логическое отрицание, суммирование по модулю два?**

* **Логическое суммирование** также называется **OR** (ИЛИ).
* **Логическое умножение** также называется **AND** (И).
* **Логическое отрицание** также называется **NOT** (НЕ).
* **Суммирование по модулю два** также называется **XOR** (исключающее ИЛИ).

1. **Какие есть методы вычисления сложных математических функций?**

* **Разложение в ряды** (например, ряд Тейлора или Маклорена).
* **Метод Ньютона** (итерационный метод для нахождения корней функции).
* **Метод приближений** (например, использование интерполяции).

1. **Методы вычисления квадратного корня:**

* **Алгоритм Ньютона** (метод приближений).
* **Метод деления пополам** для квадратных корней (бинарный поиск).
* **Ряд Тейлора** для приближённого вычисления.

1. **Простейшие варианты вычисления тригонометрических функций:**

* **Ряд Тейлора** для синуса и косинуса.
* **Алгоритм преобразования угла в радианы**.
* **Использование таблиц значений** или встроенных функций в процессорах.

1. **Что такое ряды и как это может быть использовано при выполнении арифметических операций?**

* Ряды (например, ряд Тейлора или Фурье) — это бесконечные суммы, которые могут быть использованы для приближённого вычисления функций. В вычислительных системах эти ряды могут быть обрезаны для получения приближённых значений.

1. **Законы и правила алгебры Буля:**

* Основные законы: **идемпотентность**, **ассоциативность**, **дистрибутивность**, **коммутативность** и **законы о нуле и единице**.
* **Законы поглощения**: A+AB=AA + AB = A, A⋅(A+B) = A.

1. **Аксиомы булевой алгебры:**

* **1. Закон идемпотентности**: A⋅A=A,A + A = A.
* **2. Закон двойного отрицания**: ¬(¬A)= A.
* **3. Закон коммутативности**: A⋅B=B+A,A + B = B + A.

1. **Законы нулевого и универсального множества булевой алгебры:**

* A⋅0=0, A+1= 1.
* A⋅1=A,A +0= A.

1. **Теорема де Моргана:**

* ¬(A⋅B)=¬A+¬B
* ¬(A+B)=¬A⋅¬B

**35. Законы повторения и двойной инверсии булевой алгебры:**

* A⋅A= A, A+A=A
* ¬(¬A)=A

1. **Коммуникативный и ассоциативный законы булевой алгебры:**

* Коммутативность: A⋅B=B⋅A, A+B=B+A
* Ассоциативность: A⋅(B⋅C)=(A⋅B)⋅C, A+(B+C)=(A+B)+C

1. **Дистрибутивный закон и закон поглощения булевой алгебры:**

* Дистрибутивность: A⋅(B+C)=(A⋅B)+(A⋅C), A+(B⋅C)=(A+B)⋅(A+C)
* Поглощение: A⋅(A+B)=A, A+A⋅B=A

**Арифметические сдвиги положительных и отрицательных чисел.**

1. **Правила для положительных двоичных чисел, представленных в прямом коде:**

* Арифметический сдвиг вправо: удаление последнего бита, заполняя справа нулями.
* Арифметический сдвиг влево: добавление нулей справа.

1. **Правила для отрицательных двоичных чисел, представленных в прямом коде:**

* При сдвиге отрицательных чисел в прямом коде результат может быть неадекватным, так как прямой код не учитывает знак.

1. **Правила для положительных двоичных чисел, представленных в обратном коде:**

* При сдвиге влево и вправо сохраняются правила аналогичные прямому коду, но нужно учитывать инвертирование всех битов для отрицательных чисел.

1. **Правила для отрицательных двоичных чисел, представленных в обратном коде:**

* Операции сдвига на отрицательных числах требуют инвертирования битов для получения корректных значений.

1. **Правила для положительных двоичных чисел, представленных в дополнительном коде:**

* При сдвиге влево или вправо, заполняются нулями, как и в случае с прямым кодом.

1. **Правила для отрицательных двоичных чисел, представленных в дополнительном коде:**

* Для отрицательных чисел в дополнительном коде при сдвиге используется сохранение знакового бита (для арифметического сдвига).

1. **Чем отличается логический сдвиг от арифметического?**

* Логический сдвиг сдвигает все биты, заполняя освободившиеся биты нулями.
* Арифметический сдвиг сохраняет старший бит (знак) для отрицательных чисел.

1. **Для чего используются арифметические сдвиги?**

* Арифметические сдвиги используются для умножения или деления числа на 2, сохраняя его знак.

**Представление чисел с фиксированной точкой. Арифметические операции над числами, представленными с фиксированной точкой.**

1. **Что отличает формат чисел с фиксированной и плавающей точкой?**

* В фиксированной точке количество знаков после запятой фиксировано.
* В плавающей точке позиция десятичной (или двоичной) точки может быть перемещена, что позволяет работать с более широким диапазоном значений.

1. **Арифметика для фиксированной или плавающей точкой требует больше ресурсов от процессора? Почему?**

* Арифметика с плавающей точкой требует больше вычислительных ресурсов, так как требует работы с экспонентой и мантиссой, что значительно усложняет операции.

**Принцип двойственности. Суперпозиция логических функций.**

1. **В чем заключается двойственность логических функций?**

* Двойственность заключается в том, что для любой булевой функции можно составить её двойственную функцию, заменив операции И на ИЛИ, а 0 на 1 и наоборот.

1. **Что такое суперпозиция логических функций?**

* Суперпозиция логических функций означает комбинирование нескольких логических операций для получения новой логической функции.

**Нормальные и совершенно нормальные логические функции.**

1. **Что такое нормальная и совершенно нормальная форма логических функций? Признак совершенно нормальной формы логической функции.**

* Нормальная форма логической функции представляется как конъюнкция дизъюнкций (или наоборот).
* Совершенно нормальная форма имеет строгие правила представления функций, где все переменные должны быть представлены в полном составе.

1. **Почему для логических функций получают совершенно нормальную форму?**

* Это позволяет упростить анализ и синтез логических схем, а также избежать избыточности.

1. **Сколько может быть вариантов нормальной и совершенно нормальной формы для любой логической функции?**

* Для функции с nn переменными существует 2n2^n возможных комбинаций значений.

**Минимизация булевых функций. Основные понятия.**

1. **Зачем проводят минимизацию булевых функций?**

* Минимизация позволяет упростить логические схемы, что снижает потребление ресурсов, улучшает производительность и снижает стоимость реализации.

1. **Основные понятия минимизации булевых функций:**

* **Импликанты** — минимальные группы, которые можно выделить для упрощения выражения.
* **Карты Карно** и **алгоритм Куайна-Мак-Класки** — методы минимизации.

1. **Наиболее известные методы минимизации булевых функций:**

* **Метод карт Карно**.
* **Алгоритм Куайна-Мак-Класки**.
* **Метод булевых минимизаций** (с использованием логических базисов).

**89. Что значит минимизация частично определённых булевых функций?**

* **Минимизация частично определённых булевых функций** относится к процессу упрощения булевых функций, которые не определены для всех возможных входных значений. Это означает, что для некоторых значений переменных результат функции не имеет значения (неопределён), и такие значения не учитываются при минимизации. В этом случае процесс минимизации может учитывать только те строки, для которых функция определена.

**90. Что значит минимизация системы логических булевых функций?**

* **Минимизация системы логических булевых функций** предполагает упрощение не одной, а группы или системы булевых функций. Это может включать минимизацию каждой функции по отдельности или объединение нескольких функций в одну более простую с помощью логических операций. Система минимизированных функций будет обладать минимальной сложностью, что может быть полезно для реализации в логических схемах или при разработке программного обеспечения.

**91. Алгоритм минимизации с использованием карт Карно?**

* **Алгоритм минимизации с использованием карт Карно** включает несколько шагов:
  1. **Построение карты Карно** — для каждого значения функции (всех возможных комбинаций входных переменных) наносится единичное или нулевое значение.
  2. **Группировка единиц** — ищутся группы из единичных значений, которые можно объединить, создавая более простую логическую функцию.
  3. **Минимизация** — определяется минимальная система логических выражений, которая описывает все единичные группы.
  4. **Получение минимальной функции** — результатом минимизации является упрощённая булевая функция, которая представляет собой минимальную схему.

**92. В чем преимущества и недостатки минимизации с использованием карт Карно? Когда её обычно используют?**

* **Преимущества:**
  + Простой и наглядный метод для минимизации булевых функций с небольшим числом переменных.
  + Легко визуализируется, что упрощает поиск минимальных выражений.
* **Недостатки:**
  + Для больших функций (с большим количеством переменных) метод становится громоздким, так как количество ячеек в карте Карно увеличивается экспоненциально.
* **Когда используется?**
  + Обычно применяется для булевых функций с 3-6 переменными, поскольку в этом диапазоне количество ячеек в карте Карно управляемо.

**93. Как можно объединять импликанты в карте Карно?**

* **Объединение импликант** в карте Карно — это процесс группировки единичных значений (единиц), которые могут быть объединены в более простое логическое выражение. Это делается путём поиска смежных единичных значений в горизонтальном, вертикальном или диагональном направлении. Каждая группа импликант является минимальной (неделимой).

**94. Правила построения карты Карно?**

* **Размер карты Карно** зависит от количества переменных. Для nn переменных карта Карно будет содержать 2^n ячеек.
* В каждой ячейке указывается значение функции для соответствующих значений входных переменных.
* Правила:
  1. Разделите таблицу на ячейки в зависимости от всех возможных комбинаций входных переменных.
  2. Заполните ячейки значениями функции (единицы или нули).
  3. Сгруппируйте единичные значения в прямоугольные или квадратные блоки.
  4. Каждая группа должна содержать количество ячеек, которое является степенью двойки (1, 2, 4, 8, 16 и т. д.).

**95. Какие значения могут быть в ячейках карты Карно?**

* В ячейках карты Карно могут быть два значения:
  + **1** — это значение функции для данной комбинации входных переменных.
  + **0** — это значение функции для данной комбинации входных переменных.
  + В некоторых случаях также могут быть **X** (неопределённые значения), если для соответствующей комбинации входных переменных функция не определена.

**96. Правила формирования результата минимизации с картой Карно по «1»?**

* При минимизации с картой Карно по **1** мы ищем группировки единичных значений. Эти группировки (импликанты) приводят к минимальной логической функции, которая представляет исходную функцию, используя меньшее количество операций.

**97. Правила формирования результата минимизации с картой Карно по «0»?**

При минимизации с картой Карно **по 0** (минимизация отрицательных импликант) выполняются следующие шаги:

1. **Заполнение карты Карно** — как и в случае минимизации по 1, сначала заполняется карта Карно значениями функции, но в данном случае для минимизации выбираются только ячейки, где функция равна 0. Ячейки с нулями должны быть помечены на карте.
2. **Группировка нулевых значений** — нужно найти группы нулевых значений, которые могут быть объединены. Эти группы должны быть максимально большими и содержать только нули. Группировка выполняется по тем же правилам, что и при минимизации по 1 — смежные нули объединяются в прямоугольные или квадратные блоки, и блоки должны быть степенями двойки (1, 2, 4, 8 и т. д.).
3. **Формирование минимальных отрицательных импликант** — каждая группа нулевых значений будет представлять собой минимальную отрицательную импликанту. Это выражение будет минимальным логическим выражением, которое описывает, когда функция будет равна нулю.
4. **Получение минимальной функции** — по собранным минимальным импликантам строится булева функция, которая является минимизированной функцией для тех значений, где функция принимает значение 0.

Продолжаю отвечать на вопросы:

**98. Что такое булевы базис? Какие есть варианты?**

* **Булев базис** — это набор логических операций, который является достаточным для представления всех возможных булевых функций. В теории логики важным моментом является, что из любого базиса можно составить любую булеву функцию, используя только операции из этого набора.
* **Варианты булевых базисов:**
  1. **Базис И-НЕ (AND-NOT):** Операции AND и NOT являются достаточными для представления всех булевых функций.
  2. **Базис ИЛИ-НЕ (OR-NOT):** Операции OR и NOT также могут использоваться для представления всех булевых функций.
  3. **Базис ИЛИ-И (OR-AND):** Операции OR и AND могут быть использованы для построения любых булевых функций.

**99. Какой теоремой руководствуются при выборе булевого базиса?**

* При выборе **булевого базиса** руководствуются **теоремой о полноте базиса**. Согласно этой теореме, если набор логических операций является достаточным для представления всех булевых функций, то этот набор называется полным. Например, комбинации операций **AND**, **OR**, **NOT** образуют полный булев базис, поскольку с их помощью можно выразить любую булеву функцию.

**100. Что необходимо быть в полной системе логических функций (базисе)?**

В **полной системе логических функций** (или **полном булевом базисе**) необходимо, чтобы система содержала операции, которые могут быть использованы для выражения всех возможных булевых функций. Это значит, что из любой логической функции можно составить комбинацию операций из выбранного набора базиса.

Чтобы базис был полным, должны быть выполнены следующие условия:

1. **Отрицание (NOT)** — базис должен включать операцию логического отрицания, позволяющую инвертировать значения логических переменных (например, если функция принимает значение 1, то операция NOT преобразует его в 0 и наоборот).
2. **Или (OR)** или **И (AND)** — базис должен включать хотя бы одну из этих операций (ИЛИ или И), так как с их помощью можно выразить основные логические операции, такие как конъюнкция (И) или дизъюнкция (ИЛИ).
3. **Комбинированные операции** — из базиса можно должно быть возможно комбинировать операции, например, можно получить все остальные операции, комбинируя AND и NOT (или OR и NOT). С помощью этих операций можно выразить другие логические функции, такие как **XOR** (исключающее ИЛИ), **NAND** и **NOR**.

Таким образом, для того чтобы базис был полным, в нем должны быть такие операции, которые в сочетании могут выразить любую возможную булеву функцию. Например:

* Базис **AND-NOT** (или И-НЕ) является полным, так как с помощью этих двух операций можно выразить все остальные логические операции.
* Базис **OR-NOT** (или ИЛИ-НЕ) также является полным, так как с помощью этих операций можно выразить все другие булевы функции.

Примером **полного булева базиса** может служить базис, состоящий из операций **AND**, **OR**, **NOT**, или из **NAND** или **NOR**, так как эти операции являются функционально полными, и их достаточно для реализации любой булевой функции.

**101. В чем преимущества базиса И-НЕ и ИЛИ-НЕ?**

* **И-НЕ** и **ИЛИ-НЕ** являются **функционально полными базисами**, позволяя реализовать любую булеву функцию.
* Преимущества:
  1. **Простота реализации**: современные цифровые схемы проще проектировать с использованием базисов И-НЕ или ИЛИ-НЕ.
  2. **Минимизация логических элементов**: меньшее количество логических элементов требуется для реализации сложных функций.
  3. **Универсальность**: возможность строить сложные схемы из одного типа операций.

**102. Приведите пример функции в базисе И-НЕ.**

Рассмотрим функцию F = (A ∨ B) ∧ C в базисе И-НЕ:

1. Представим дизъюнкцию: A ∨ B = ¬(¬A ∧ ¬B).
2. Представим конъюнкцию: F = ¬(¬(¬A ∧ ¬B) ∧ ¬C).

**103. Приведите пример функции в базисе ИЛИ-НЕ.**

Рассмотрим функцию F = (A ∧ B) ∨ C в базисе ИЛИ-НЕ:

1. Представим конъюнкцию: A ∧ B = ¬(¬A ∨ ¬B).
2. Представим дизъюнкцию: F = ¬(¬(¬(¬A ∨ ¬B) ∨ ¬C)).

**Основные понятия алгебры логики. Способы задания логической функции.**

**104. Что включают основные понятия алгебры логики?**

1. **Переменные**: принимают значения 0 или 1.
2. **Операции**: логические функции (например, AND, OR, NOT).
3. **Функции**: булевы выражения, описывающие связь между переменными.
4. **Законы алгебры**: коммутативность, ассоциативность, дистрибутивность, двойственность и т.д.
5. **Таблицы истинности**: определяют значения функции для всех возможных наборов входных данных.

**105. Какие есть способы задания логической функции? Пример.**

1. **Таблица истинности**: задает значения функции для всех наборов входных переменных.
2. **Аналитическая форма**: использование булевых операций. Пример: F(A,B)=A∧¬BF(A, B) = A \land \neg BF(A,B)=A∧¬B.
3. **Карты Карно**: графическое представление для упрощения функций.
4. **Логические схемы**: представление через элементы (И, ИЛИ, НЕ).

Пример: для функции F(A,B)=A∧BF(A, B) = A \land BF(A,B)=A∧B:

* Таблица истинности:  
  A=0,B=0,F=0A = 0, B = 0, F = 0A=0,B=0,F=0,  
  A=1,B=1,F=1A = 1, B = 1, F = 1A=1,B=1,F=1.

**Минимизация логических выражений методом Квайна.**

**106. Алгоритм минимизации Квайна?**

1. **Построение импликант**: выделение всех простых импликант (совпадающих частей минимальных выражений).
2. **Построение таблицы покрытия**: сопоставление импликант с набором истинных значений.
3. **Минимизация**: выбор минимального числа импликант, которые покрывают все наборы истинных значений функции.

**107. Зачем нужна импликантная таблица в методе Квайна?**

Импликантная таблица позволяет:

1. Систематизировать связь между импликантами и наборами истинных значений функции.
2. Упростить процесс выбора минимального покрытия.

**108. Как составляется импликантная таблица в методе Квайна?**

1. По горизонтали указываются наборы значений переменных, при которых функция равна 1.
2. По вертикали указываются простые импликанты.
3. Отмечаются ячейки, где импликанта покрывает набор.

**109. Что такое тупиковая функция в методе Квайна?**

Тупиковая функция — функция, у которой нельзя выделить простую импликанту, покрывающую дополнительные наборы значений, без увеличения длины минимального выражения.

**110. Алгоритм выбора импликант из импликантной таблицы при минимизации методом Квайна?**

1. Выделяются обязательные импликанты (покрывающие уникальные наборы).
2. Проверяются пересечения между оставшимися импликантами.
3. Выбирается минимальное подмножество, покрывающее все наборы.

**111. Достоинства и недостатки метода Квайна?**

**Достоинства**:

* Обеспечивает минимизацию функций.
* Формализованный алгоритм.

**Недостатки**:

* Высокая вычислительная сложность для большого числа переменных.
* Непрактично для функций с большим количеством входных данных.

**Полная система логических функций.**

**112. Что понимают под полной системой логических функций? Сколько их?**

Полная система логических функций — это множество операций, с помощью которых можно выразить любую булеву функцию.

Примеры:

1. **И, НЕ**.
2. **ИЛИ, НЕ**.
3. **Штрих Шеффера (НИ)**.
4. **Стрелка Пирса (ИЛИ-НЕ)**.

Всего таких систем **16** (если учитывать только функционально полные базисы).

**113. Как строится последовательность в полной системе логических функций?**

1. Выбирается базис.
2. Логическая функция представляется в виде комбинации операций из базиса.
3. Построение осуществляется по правилам булевой алгебры и минимизации логического выражения.

**RISK, CISC, MISC, VLIW.**

**114. Что понимают под архитектурой процессоров? Что важно знать программисту?**

**Архитектура процессоров** — это структура, организация и набор принципов, определяющих работу процессора.

Программисту важно знать:

* Набор команд процессора.
* Разрядность процессора.
* Способы адресации памяти.
* Устройство памяти и методы оптимизации производительности.

**115. Какие есть варианты классификации процессоров?**

1. По архитектуре: RISC, CISC, MISC, VLIW.
2. По разрядности: 32-битные, 64-битные.
3. По количеству ядер: одноядерные, многоядерные.
4. По назначению: универсальные, специализированные.
5. По разрядности шины данных: 16-битные, 32-битные, 64-битные.

**116. Что понимают под набором команд? На что в процессоре влияет набор команд?**

**Набор команд** — это перечень операций, которые процессор может выполнять.  
Влияет на:

* Производительность процессора.
* Простоту разработки программного обеспечения.
* Совместимость с другими устройствами.

**117. Какими параметрами может отличаться процессор в рамках одной архитектуры? Чем может отличаться память?**

Процессоры могут отличаться:

* Тактовой частотой.
* Количеством ядер.
* Энергопотреблением.
* Объемом кэш-памяти.

Память может отличаться:

* Типом (DRAM, SRAM).
* Объемом.
* Скоростью доступа.
* Энергозависимостью.

**118. Что такое шина процессора? Какие они бывают? Что определяет разрядность шины данных для арифметических операций?**

**Шина процессора** — это набор линий, через которые передаются данные, адреса и управляющие сигналы.  
Бывают:

1. **Данные**.
2. **Адреса**.
3. **Управления**.

Разрядность шины данных определяет максимальную длину слова, обрабатываемую процессором за один такт.

**119. Что определяется на аппаратном уровне в многоуровневой архитектуре?**

* Распределение памяти (уровни кэша, оперативная память, долговременная память).
* Алгоритмы обработки инструкций (конвейеризация, суперскалярность).
* Способы передачи данных между уровнями (шины и контроллеры).

**120. Приведите классификацию архитектур с точки зрения системы команд.**

1. **RISC**: ограниченный набор простых команд, высокая производительность.
2. **CISC**: сложные команды, каждая из которых выполняет несколько операций.
3. **VLIW**: команды фиксированной длины для параллельного выполнения.
4. **MISC**: минимальный набор команд.

**121. Приведите отличительные признаки архитектур команд.**

* **RISC**: фиксированная длина команд, одно тактное выполнение.
* **CISC**: переменная длина команд, сложные операции.
* **VLIW**: инструкции группируются для параллельного выполнения.
* **MISC**: минимальная сложность набора команд.

**122. Какому процессору потребуется меньше и больше микрокоманд для вычисления простейшего арифметического выражения по сложению нескольких чисел?**

* **RISC**: меньше микрокоманд, так как команды выполняются быстро и эффективно.
* **CISC**: больше микрокоманд, так как команды могут быть сложнее и выполнять дополнительные операции.

**123. В чем особенность и преимущества процессоров с RISC архитектурой?**

* Особенность: простой и ограниченный набор команд.
* Преимущества:
  1. Высокая производительность за счет конвейеризации.
  2. Простота реализации в аппаратуре.
  3. Меньшее энергопотребление.

**124. Где в основном используются процессоры с RISC архитектурой и почему они не вытеснили процессоры CISC?**

* Используются: в мобильных устройствах, встроенных системах.
* Не вытеснили CISC из-за совместимости последних с существующим ПО и их эффективности в сложных вычислительных задачах.

**125. В чем преимущества CISC процессоров? Где их можно встретить?**

* Преимущества:
  1. Более сложные инструкции.
  2. Поддержка сложных алгоритмов.
* Встречаются: в настольных ПК, серверах, системах с большим объемом памяти.

**126. MISC процессоры. Достоинства. Где используются?**

* Достоинства:
  1. Минимальный набор команд.
  2. Простота реализации и низкое энергопотребление.
* Используются: в специализированных устройствах, например, системах управления.

**127. VLIW процессоры. Достоинства. Где используются?**

* Достоинства:
  1. Высокая параллельность.
  2. Простота в проектировании аппаратуры.
* Используются: в мультимедийных устройствах, цифровой обработке сигналов.

**128. Что такое виртуальные архитектуры? Зачем их используют? В чем достоинства и недостатки?**

* **Виртуальные архитектуры** — модели вычислительных систем, абстрагированные от конкретного оборудования.
* Используют для:
  1. Универсальности ПО.
  2. Оптимизации использования ресурсов.

**Достоинства**:

* Универсальность.
* Упрощение разработки.

**Недостатки**:

* Потеря производительности.
* Сложность реализации.

**129. Что понимают под командами (инструкциями) процессора? Какие они бывают?**

Команды (инструкции) — набор операций, выполняемых процессором.  
Виды:

1. Арифметические.
2. Логические.
3. Управления потоком.
4. Работа с памятью.

**130. Минимальный набор арифметических операций в АЛУ? Почему их обычно больше?**

Минимальный набор: сложение, вычитание.  
Дополнительно включают умножение, деление для повышения производительности.

**131. Кто предоставляет инструкции процессору? Сколько инструкций может выполнить за один такт процессора?**

* Инструкции предоставляет компилятор.
* Количество зависит от архитектуры процессора (обычно 1 для RISC, несколько для VLIW).

**132. Что такое тактирование процессора? На что это влияет? Как это сейчас используют?**

Тактирование — синхронизация работы процессора по тактовому сигналу.  
Влияет на:

* Скорость выполнения инструкций.
* Общую производительность.

**133. Что такое Spectre и Meltdown? Из-за чего они появлялись?**

**Spectre** и **Meltdown** — это классы уязвимостей процессоров, позволяющие злоумышленникам получать доступ к защищенным данным.

Причины появления:

1. Использование спекулятивного исполнения инструкций.
2. Неправильное разделение данных между ядрами процессора.
3. Ошибки в механизмах кэширования.

**134. В чем сложность устранения уязвимостей типа Spectre и Meltdown?**

1. Затруднения в исправлении на аппаратном уровне.
2. Значительное снижение производительности при программной защите.
3. Затрагивают множество процессоров, включая старые модели.

**Представление чисел с плавающей точкой.**

**135. Алгоритм сложения двух чисел в формате с плавающей точкой?**

1. Выровнять порядки чисел, сместив мантиссу меньшего числа.
2. Сложить или вычесть мантиссы.
3. Нормализовать результат.
4. Корректировать порядок результата.
5. Применить округление.

**136. Алгоритм умножения двух чисел в формате с плавающей точкой?**

1. Сложить порядки чисел.
2. Перемножить мантиссы.
3. Нормализовать результат.
4. Корректировать порядок результата.
5. Применить округление.

**137. Алгоритм деления двух чисел в формате с плавающей точкой?**

1. Вычесть порядок делителя из порядка делимого.
2. Разделить мантиссы.
3. Нормализовать результат.
4. Корректировать порядок результата.
5. Применить округление.

**IEEE 754.**

**138. Из каких фрагментов состоит формат с плавающей точкой?**

1. **Знак**: определяет положительное или отрицательное число.
2. **Порядок**: показывает степень числа.
3. **Мантисса**: дробная часть, определяющая точность.

**139. В чем преимущества и недостатки использования формата с плавающей точкой?**

**Преимущества:**

* Большой диапазон чисел.
* Универсальность для разных типов вычислений.

**Недостатки:**

* Потеря точности из-за округлений.
* Сложность аппаратной реализации.

**140. Как формируется мантисса числа с плавающей точкой?**

1. Число переводится в нормализованную форму: 1.XXX⋅2^n
2. 111 перед запятой опускается для экономии места.

**141. Зачем в формат с плавающей точкой ввели ненормализованные числа? Какой у них признак?**

**Зачем:** для представления малых чисел, которые не могут быть нормализованы.  
**Признак:** порядок устанавливается на минимальное значение, а мантисса не нормализуется.

**142. В районе какого числа наблюдаются проблемы с точностью у числа с плавающей точкой? Как решают эту проблему?**

**Проблемы:** возникают вблизи нуля и при больших порядках.  
**Решение:** использовать денормализованные числа и корректные алгоритмы округления.

**143. Какие есть проблемы при проведении арифметических операций с плавающей точкой?**

1. Потеря точности при сложении чисел с разным порядком.
2. Ошибки округления.
3. Накопление погрешностей при многократных вычислениях.

**144. Что является источником погрешности операций сложения с плавающей точкой? Как можно минимизировать погрешность при больших объемах вычислений?**

**Источник:** потеря значимых цифр при выравнивании порядков.  
**Минимизация:**

1. Упорядочивание чисел по величине перед сложением.
2. Применение компенсирующих алгоритмов (например, алгоритм Кэхэна).

**145. Какие специальные числа присутствуют в стандарте IEEE 754? Зачем они?**

1. **+0 и -0**: различие при асимптотических вычислениях.
2. **Infinity**: для представления переполнения.
3. **NaN**: для обозначения некорректных операций.

**146. Когда можно получить NaN?**

1. Деление 0 на 0.
2. Извлечение корня из отрицательного числа.
3. Операции с участием NaN.

**147. Почему в формате с плавающей точкой присутствует два варианта числа 0? Какие это решает проблемы?**

* Обеспечивает корректность при вычислениях с направлением градиентов (например, 1/(+0)не равно1/(−0)).

**148. Особенности округления в IEEE 754? Как можно настроить?**

**Особенности:**

* Выбор ближайшего числа.
* Сохранение симметрии.

**Настройка:** через параметры компилятора или инструкции процессора.

**149. Какие есть варианты округления в IEEE 754?**

1. К ближайшему.
2. Вверх.
3. Вниз.
4. К нулю.

**150. Что такое неассоциативность арифметических операций? Для какого формата она не соблюдается?**

**Неассоциативность:** результат операции зависит от порядка вычислений.  
Проявляется в числах с плавающей точкой из-за округлений.

**151. В чем есть проблема использования констант в формате с плавающей точкой в цикле? Почему?**

**Проблема:** накопление погрешности из-за округления при каждом шаге цикла.

**152. В чем есть проблема сравнения констант в формате с плавающей точкой? Как решать проблему?**

**Проблема:** точность представления числа может быть недостаточной.  
**Решение:** использовать сравнение с учетом допустимой погрешности (эпсилон).

**153. Что такое CPU и GPU? И при чем тут формат с плавающей точкой?**

* **CPU**: универсальный процессор.
* **GPU**: специализированный для массовых параллельных вычислений.
* Формат с плавающей точкой используется для сложных вычислений в обеих архитектурах.

**154. Что определяет стандарт IEEE 754?**

1. Формат хранения чисел с плавающей точкой.
2. Алгоритмы округления.
3. Исключения и специальные числа.

**155. Три основных двоичных формата с плавающей запятой стандарта IEEE 754?**

1. Одинарная точность (32 бита).
2. Двойная точность (64 бита).
3. Расширенная точность (80 бит и больше).

**156. Примерный диапазон чисел формата с одинарной точностью в стандарте IEEE 754?**

±1.4×10−45 до ±3.4×1038\pm 3.4 \times 10^{38}±3.4×1038.

**157. Самый точный и не точный формат стандарта IEEE 754?**

* Самый точный: **quadruple precision (128-битный формат)**.
* Наименее точный: **одинарная точность (32-битный формат)**.

**158. Какие форматы добавились в последних редакциях стандарта IEEE 754?**

1. Десятичные форматы (Decimal32, Decimal64, Decimal128).
2. Полуточный формат (Half-precision, 16 бит).

**159. Примерный график зависимости точности арифметических операций для формата с одинарной точностью и двойной точностью?**

* С увеличением числа операций точность уменьшается быстрее для **одинарной точности**, чем для **двойной точности**.

**160. Десятичный32, десятичный64, десятичное128. В чем их особенность? Чем они отличаются от бинарных?**

Особенность: основаны на десятичной арифметике, что минимизирует погрешности при работе с денежными значениями.  
Отличие: прямое представление десятичных дробей.

**161. Как округляет процессор, если не указать вариант округления при настройке компилятора?**

Процессор использует округление **к ближайшему значению с округлением к чётному** ("round to nearest, ties to even") как стандартное поведение в IEEE 754.

**162. Необходимые операции, которые определяет стандарт IEEE 754?**

1. Основные арифметические операции: сложение, вычитание, умножение, деление.
2. Операции сравнения.
3. Извлечение квадратного корня.
4. Конверсии между форматами.
5. Манипуляции с битами (например, копирование знака).

**163. Какие есть редакции стандарта IEEE 754? В чем их отличия?**

1. **1985**: первый выпуск стандарта, ввод базовых форматов (одинарная, двойная точность).
2. **2008**: добавлены десятичные форматы и полуточные (half-precision).
3. **2019**: уточнения по исключениям, расширенные форматы, улучшенные рекомендации по округлению.

**164. Перечислите пять исключений стандарта IEEE 754?**

1. Деление на ноль.
2. Переполнение (overflow).
3. Подпоток (underflow).
4. Потеря точности (inexact).
5. Некорректная операция (invalid operation), например, sqrt(-1).

**165. Что происходит, если при выполнении операций в формате с плавающей точкой АЛУ попадает на исключение?**

* Устанавливается соответствующий флаг исключения.
* Выдаётся результат (например, NaN или бесконечность).
* Программный обработчик может обработать исключение.

**166. Какие есть два типа NaN в стандарте IEEE 754? В чем их отличия?**

1. **Quiet NaN (qNaN):** передаётся через вычисления без генерации ошибки.
2. **Signaling NaN (sNaN):** вызывает ошибку при использовании в вычислениях.

**Параллелизм. Виды, организация.**

**167. Параллелизм. Зачем он нужен? Какие есть варианты его построения?**

**Зачем:** увеличить производительность вычислительных систем.

**Варианты:**

1. **Временной:** последовательное выполнение этапов на разных частях данных.
2. **Пространственный:** одновременное выполнение операций на разных устройствах.
3. **Потоковый:** параллелизм инструкций и данных (SIMD, MIMD).

**168. В чем достоинство и какие возможности временного параллелизма?**

1. **Достоинства:**
   * Простота реализации.
   * Возможность использования существующих процессоров.
2. **Возможности:**
   * Увеличение скорости выполнения.
   * Улучшение загрузки вычислительных блоков.

**169. Что ограничивает параллелизм реальных арифметических задач?**

1. Зависимости данных.
2. Ограниченная пропускная способность памяти.
3. Накладные расходы на синхронизацию.
4. Необходимость в сложной программной и аппаратной поддержке.

**Типы памяти.**

**170. Что такое матрица памяти? Из чего она состоит?**

**Матрица памяти** — структура для хранения данных в памяти, организованная в виде сетки ячеек.

**Состоит из:**

1. Ячеек памяти.
2. Линий адресации.
3. Линий данных.
4. Линий управления (например, запись/чтение).

**171. Какие типы памяти можно встретить в вычислительных системах? Их особенности применения.**

1. **Оперативная память (RAM):** временное хранение данных.
2. **Постоянная память (ROM):** неизменяемые данные.
3. **Кэш-память:** ускорение доступа к часто используемым данным.
4. **Видеопамять (VRAM):** хранение данных для графической обработки.
5. **SSD/HDD:** долговременное хранение данных.

**172. Как работает запись-чтение из памяти?**

1. **Чтение:**
   * Устройство передаёт адрес.
   * Память возвращает данные по этому адресу.
2. **Запись:**
   * Устройство передаёт адрес и данные.
   * Память записывает данные по указанному адресу.

**173. Чем характеризуются матрицы памяти и на какие классы делятся?**

**Характеристики:**

1. Скорость доступа.
2. Энергопотребление.
3. Плотность хранения данных.

**Классы:**

1. Энергозависимые.
2. Энергонезависимые.

**174. Основные классы ОЗУ? В чем особенность «запоминания»?**

1. **DRAM (динамическое):** требует периодической перезарядки.
2. **SRAM (статическое):** сохраняет данные без перезарядки, но более дорогая.

**175. Сравните достоинства и недостатки энергозависимых запоминающих устройств.**

**Достоинства:**

* Высокая скорость.
* Низкая стоимость хранения на бит.

**Недостатки:**

* Потеря данных при отключении питания.

**Устройства ЭВМ. Состав АЛУ.**

**176. Какие основные элементы ЭВМ? Зачем они нужны?**

1. **Процессор:** выполнение инструкций.
2. **Память:** хранение данных.
3. **Ввод/вывод:** взаимодействие с пользователем и внешними устройствами.
4. **АЛУ:** выполнение арифметических и логических операций.

**177. Как обозначается АЛУ и для чего используются входы и выходы?**

**Обозначение:** Arithmetic Logic Unit (АЛУ).  
**Входы:** принимают данные и управляющие сигналы.  
**Выходы:** возвращают результат операций.

**178. Какую роль выполняет АЛУ в процессорах? От чего зависит разрядность управляющего сигнала?**

**Роль:** выполнение базовых арифметических и логических операций.  
**Разрядность управляющего сигнала** зависит от количества поддерживаемых операций.

**179. Минимальный набор операций АЛУ. Зачем на входе в одном канале присутствует инвертор?**

**Минимальный набор:** сложение, вычитание, логические операции (AND, OR, NOT).  
**Инвертор:** позволяет реализовать операции вычитания через сложение и отрицание.

**Сумматор. Многоразрядный сумматор. Ускорение выполнения математических операций.**

**180. Обозначение и таблица истинности сумматора? Для чего используются электронные схемы сумматоров?**

1. **Обозначение:** сумматор обозначается как логическая схема с входами A,B,A, B,A,B, и CarryCarryCarry и выходами SumSumSum и Carry OutCarry\ OutCarry Out.
2. **Таблица истинности (полусумматор):**

| **AAA** | **BBB** | **Sum** | **Carry** |
| --- | --- | --- | --- |
| 0 | 0 | 0 | 0 |
| 0 | 1 | 1 | 0 |
| 1 | 0 | 1 | 0 |
| 1 | 1 | 0 | 1 |

1. **Использование:** выполняют сложение двоичных чисел, ключевой компонент АЛУ.

**181. Какие бывают сумматоры? Что такое бит переноса? Сколько нужно сумматоров для выполнения сложения одного байта?**

1. **Типы сумматоров:**
   * Полусумматоры.
   * Полные сумматоры.
   * Многоразрядные (с последовательным или параллельным переносом).
2. **Бит переноса:** дополнительный бит, возникающий при сложении, если сумма превышает максимальное значение разряда.
3. Для сложения одного байта нужно **8 полных сумматоров.**

**182. Какие виды сумматоров используются в вычислительных системах? Их достоинства и недостатки?**

1. **Последовательный:**
   * **Достоинства:** простота реализации.
   * **Недостатки:** медленная работа из-за последовательной обработки битов.
2. **Параллельный:**
   * **Достоинства:** высокая скорость работы.
   * **Недостатки:** сложная схема из-за генерации и передачи переносов.
3. **Сумматоры с ускоренным переносом (CLA):**
   * **Достоинства:** высокая скорость за счёт предвычисления переносов.
   * **Недостатки:** сложность реализации.

**183. В чем смысл работы сумматоров с ускоренным переносом?**

Сумматоры с ускоренным переносом (Carry Lookahead Adder, CLA) используют предвычисление переносов, что позволяет выполнять сложение быстрее, чем в схемах с последовательным переносом.

**184. В чем смысл работы префиксного сумматора?**

Префиксный сумматор строит дерево переносов, чтобы определить биты переносов одновременно для всех разрядов, что минимизирует задержки.

**185. Зачем используют компараторы? Какой сигнал на выходе?**

1. **Зачем:** для сравнения чисел (равно, больше, меньше).
2. **Выходной сигнал:**
   * EQ=1EQ = 1EQ=1, если числа равны.
   * GT=1GT = 1GT=1, если первое число больше.
   * LT=1LT = 1LT=1, если первое число меньше.

**186. На каких элементах строится компаратор равенства? Как он работает?**

1. **Элементы:** XOR-гейты и логические элементы AND.
2. **Принцип работы:** если все разряды двух чисел равны, на выходе EQ=1EQ = 1EQ=1.

**187. Как строится и работает компаратор больше/меньше?**

1. **Построение:** состоит из цепочек логических элементов для последовательного сравнения битов.
2. **Принцип работы:**
   * Если старший разряд одного числа больше, то GT=1GT = 1GT=1.
   * Если старший разряд меньше, то LT=1LT = 1LT=1.
   * Если равны, проверяются младшие разряды.

**Языки описания аппаратуры. ПЛИС (FPGA) модули.**

**188. Какие есть языки описания аппаратуры? Для чего они применяются?**

1. **Языки:** VHDL, Verilog, SystemVerilog.
2. **Применение:** разработка цифровых схем, моделирование поведения аппаратуры, синтез схем для FPGA или ASIC.

**189. Основные цели HDL. Их место в разработке вычислительных систем.**

1. **Основные цели:**
   * Проектирование и симуляция аппаратуры.
   * Оптимизация и генерация конечных схем.
2. **Место в разработке:** используются на этапах проектирования и реализации вычислительных систем.

**190. Что такое FPGA? Зачем их применяют в вычислительных системах?**

1. **FPGA (Field Programmable Gate Array):** программируемая логическая матрица.
2. **Применение:** для создания специализированных вычислительных устройств, когда требуется высокая производительность и гибкость.

**191. В чем преимущество и недостатки выполнения вычислительных операций в FPGA в сравнении с процессорами?**

1. **Преимущества:**
   * Высокая параллельность.
   * Низкая задержка выполнения.
2. **Недостатки:**
   * Ограничения в размере памяти.
   * Высокая сложность разработки.

**192. Как строятся современные вычислительные структуры с применением FPGA? Пример.**

1. **Построение:**
   * Используются HDL для описания логики.
   * Загружается конфигурация в FPGA для выполнения специфических задач.
2. **Пример:** ускорители машинного обучения, обработка сигналов.

**193. Где применяются FPGA?**

1. Системы реального времени.
2. Цифровая обработка сигналов.
3. Разработка прототипов микропроцессоров.
4. Ускорение научных вычислений.

**194. Из чего состоит FPGA? Как формируется логическая функция в нём?**

1. **Состав:**
   * Программируемые логические блоки.
   * Матрица коммутации.
   * Блоки памяти.
   * Входы/выходы.
2. **Формирование логики:**
   * Логическая функция описывается на HDL.
   * Синтезируется и загружается в FPGA.

**Код Грея.**

**195. Что такое код Грея? Пример трёхбитного кода Грея?**

**Код Грея:** бинарный код, в котором соседние числа отличаются только одним битом.

**Пример:**

| **Двоичный** | **Код Грея** |
| --- | --- |
| 000 | 000 |
| 001 | 001 |
| 010 | 011 |
| 011 | 010 |
| 100 | 110 |
| 101 | 111 |
| 110 | 101 |
| 111 | 100 |

**196. Для чего и когда используется код Грея?**

Используется в:

1. Цифровых системах с уменьшением ошибок при переходах между состояниями.
2. Кодировании позиций вращающихся объектов (энкодеры).

**197. Какие варианты кода Грея есть? Что такое рефлексивный код Грея?**

1. **Варианты:**
   * Прямой.
   * Обратный.
2. **Рефлексивный код Грея:**
   * Код строится отражением предыдущей строки.

**АЦП и ЦАП. Предназначение. Параметры сравнения и выбора.**

**198. Что такое АЦП и ЦАП? Для чего они нужны?**

1. **АЦП (аналогово-цифровой преобразователь):** преобразует аналоговый сигнал в цифровой.
   * Применяется для обработки аналоговых сигналов в цифровой форме.
2. **ЦАП (цифро-аналоговый преобразователь):** преобразует цифровой сигнал в аналоговый.
   * Применяется для управления аналоговыми устройствами с помощью цифровых сигналов.

**199. Основные параметры АЦП?**

1. **Разрядность:** определяет точность преобразования (количество уровней квантования).
2. **Частота дискретизации:** максимальная частота, с которой АЦП может измерять сигнал.
3. **Диапазон входного сигнала:** максимально возможное значение аналогового сигнала.
4. **Линейность:** определяет точность преобразования по всему диапазону.
5. **Шум и помехоустойчивость:** уровень влияния внешних факторов на точность преобразования.

**200. Что такое квантование и дискретизация?**

1. **Дискретизация:** преобразование непрерывного сигнала в последовательность отсчётов во времени.
2. **Квантование:** процесс округления значений аналогового сигнала к ближайшим уровням цифрового представления.
   * В процессе возникает **погрешность квантования**.

**Параметры АЦП и ЦАП. Сравнение и выбор**

**201. Как правильно выбирать скорость АЦП? В чем смысл теоремы Шеннона-Котельникова?**

1. **Выбор скорости АЦП:**
   * Скорость АЦП должна быть как минимум вдвое больше максимальной частоты сигнала, чтобы избежать потерь информации (дискретизации).
   * Частота дискретизации выбирается в соответствии с теоремой Шеннона-Котельникова.
2. **Суть теоремы Шеннона-Котельникова:**
   * Для восстановления сигнала из дискретных отсчетов частота дискретизации должна быть не менее чем в 2 раза выше максимальной частоты спектра сигнала.

**202. Зачем нужны корректирующие коды?**

1. Они обеспечивают защиту данных от ошибок, возникающих при передаче или хранении.
2. Применяются в сетях, хранилищах данных, и устройствах связи для восстановления исходной информации.

**203. В чем суть кода Хэмминга? Область применения.**

1. **Суть:**
   * Код Хэмминга добавляет избыточные биты для обнаружения и исправления одиночных ошибок.
2. **Применение:**
   * Защита данных в памяти компьютеров.
   * Средства связи (например, спутниковые каналы).

**204. Что такое контрольные суммы? Какие есть варианты их формирования. Область применения.**

1. **Контрольные суммы:**
   * Итоговые значения, рассчитанные на основе содержимого данных для проверки их целостности.
2. **Методы:**
   * Простое суммирование.
   * CRC (циклический избыточный код).
3. **Применение:**
   * Передача данных по сети, сохранение данных на диске.

**205. Система контроля. Виды ошибок результата арифметических операций.**

1. **Система контроля:**
   * Механизм проверки результата выполнения арифметических операций.
2. **Виды ошибок:**
   * Переполнение.
   * Ошибки округления.
   * Ошибки разрядности.

**206. Систематические коды. Как они работают?**

1. **Систематические коды:**
   * Код, в котором данные сохраняются в неизменном виде, а к ним добавляются контрольные биты.
2. **Принцип работы:**
   * Данные передаются вместе с проверочными кодами, которые позволяют обнаружить и исправить ошибки.

**207. Условие обнаружения ошибки систематическим кодом.**

* Ошибка может быть обнаружена, если число ошибок не превышает кодировочную способность систематического кода.

**208. Простейший метод контроля ошибок. Как считается? Их недостаток.**

1. **Простейший метод:**
   * Суммирование данных или вычисление паритета.
2. **Недостаток:**
   * Невозможность обнаружить сложные множественные ошибки.

**209. Самоконтролирующиеся коды. Какую логическую операцию используют?**

* Используют операцию **XOR** (исключающее ИЛИ) для формирования и проверки кодов.

**210. Какие самоконтролирующиеся коды сейчас используются? В чем их преимущество?**

1. **Примеры:**
   * Код Хэмминга, CRC.
2. **Преимущества:**
   * Обнаружение и исправление ошибок в реальном времени.

**211. При использовании кода Хэмминга, что передается помимо информации?**

* Передаются контрольные (избыточные) биты, используемые для обнаружения и исправления ошибок.

**212. Что определяет количество контрольных бит в коде Хэмминга?**

* Количество контрольных битов зависит от длины информационного слова: 2r≥r+k+12^r \geq r + k + 12r≥r+k+1 где rrr — число контрольных битов, kkk — длина данных.

**213. Что такое логический вентиль? Пример обозначения в разных стандартах. Таблица истинности.**

1. **Логический вентиль:**
   * Основной элемент цифровой схемы, выполняющий логическую операцию (AND, OR, NOT).
2. **Пример:**
   * ANSI: & для AND.
   * IEC: прямоугольный блок с описанием.

**214. Что такое электронный буфер? Его предназначение.**

* Устройство, усиливающее сигнал или изолирующее цепи, чтобы предотвратить взаимное влияние.

**215. Логический вентиль AND. Пример обозначения в разных стандартах.**

1. **Логический вентиль AND:**
   * Выполняет логическую операцию "И", где выход равен 1, если оба входа равны 1.
2. **Пример обозначения:**
   * **ANSI:** Стандартное обозначение — полукруглый символ с двумя входами слева и выходом справа.
   * **IEC:** Треугольная форма с двумя входами и одним выходом.

**216. Логический вентиль AND. Таблица истинности.**

| **A** | **B** | **A AND B** |
| --- | --- | --- |
| 0 | 0 | 0 |
| 0 | 1 | 0 |
| 1 | 0 | 0 |
| 1 | 1 | 1 |

**217. Логический вентиль OR. Пример обозначения в разных стандартах.**

1. **Логический вентиль OR:**
   * Выполняет логическую операцию "ИЛИ", где выход равен 1, если хотя бы один из входов равен 1.
2. **Пример обозначения:**
   * **ANSI:** Обозначается как полукруглый символ с двумя входами и выходом.
   * **IEC:** Обозначение похоже на символ AND, но с закругленными углами.

**218. Логический вентиль OR. Таблица истинности.**

| **A** | **B** | **A OR B** |
| --- | --- | --- |
| 0 | 0 | 0 |
| 0 | 1 | 1 |
| 1 | 0 | 1 |
| 1 | 1 | 1 |

**219. Конъюнкция. Способы отображения в формулах. Диаграмма Венна.**

1. **Конъюнкция (AND):**
   * Логическая операция, которая возвращает истину только тогда, когда все операнды истинны.
   * В формуле: A∧BA \land BA∧B.
2. **Диаграмма Венна:**
   * Пересечение двух кругов показывает элементы, которые принадлежат обоим множествам.

**220. Дизъюнкция. Способы отображения в формулах. Диаграмма Венна.**

1. **Дизъюнкция (OR):**
   * Логическая операция, которая возвращает истину, если хотя бы один операнд истин.
   * В формуле: A∨BA \lor BA∨B.
2. **Диаграмма Венна:**
   * Объединение двух кругов показывает все элементы, которые принадлежат хотя бы одному из множеств.

**221. Исключающее ИЛИ. Альтернативные названия. Пример обозначения в разных стандартах.**

1. **Исключающее ИЛИ (XOR):**
   * Логическая операция, которая возвращает истину, если только один из операндов истинен.
2. **Альтернативные названия:**
   * "Или, но не оба".
3. **Пример обозначения:**
   * **ANSI:** Два входа, форма буквы "V".
   * **IEC:** Символ с двумя входами, который пересекает друг друга, образуя форму "X".

**222. Исключающее ИЛИ. Таблица истинности. Способ отображения в формулах.**

| **A** | **B** | **A XOR B** |
| --- | --- | --- |
| 0 | 0 | 0 |
| 0 | 1 | 1 |
| 1 | 0 | 1 |
| 1 | 1 | 0 |

**223. Логический вентиль NAND4. Пример обозначения. Таблица истинности.**

1. **Логический вентиль NAND4:**
   * Выполняет операцию отрицания конъюнкции для 4 входов.
2. **Пример обозначения:**
   * Стандартное обозначение в виде прямоугольного элемента с 4 входами и выходом.
3. **Таблица истинности:**  
   | A | B | C | D | A NAND B NAND C NAND D | |---|---|---|---|-----------------------| | 0 | 0 | 0 | 0 | 1 | | 0 | 0 | 0 | 1 | 1 | | 0 | 0 | 1 | 0 | 1 | | 0 | 1 | 0 | 0 | 1 | | 1 | 1 | 1 | 1 | 0 |

**224. Логический вентиль NOR3. Пример обозначения. Таблица истинности.**

1. **Логический вентиль NOR3:**
   * Выполняет операцию отрицания дизъюнкции для 3 входов.
2. **Пример обозначения:**
   * Стандартное обозначение в виде треугольной формы с 3 входами и выходом.
3. **Таблица истинности:**  
   | A | B | C | A NOR B NOR C | |---|---|---|----------------| | 0 | 0 | 0 | 1 | | 0 | 0 | 1 | 0 | | 0 | 1 | 0 | 0 | | 1 | 0 | 0 | 0 |

**225. Какие есть уровни питания цифровых микросхем? Почему стараются снизить напряжение питания цифровых микросхем?**

1. **Уровни питания:**
   * От 3.3 В до 1.8 В и ниже для современных микросхем.
2. **Зачем снижать напряжение?**
   * Снижение напряжения уменьшает потребляемую мощность и тепловыделение, улучшая эффективность и долговечность устройства.

**226. Логические уровни. Что они определяют? На какие зоны делятся?**

1. **Логические уровни:**
   * Определяют значения сигнала (0 или 1) в цифровых системах.
2. **Зоны деления:**
   * **Логический 0:** Низкий уровень напряжения.
   * **Логический 1:** Высокий уровень напряжения.

**227. Почему у цифровых микросхем разные логические уровни у приемника и источника?**

* Это позволяет избежать коротких замыканий и обеспечивает правильную работу схем при взаимодействии разных компонентов с различными логическими уровнями.

**228. Какие есть четыре основных семейства цифровых микросхем?**

1. **Bipolar Logic (TTL).**
2. **CMOS (Complementary Metal-Oxide-Semiconductor).**
3. **ECL (Emitter-Coupled Logic).**
4. **BiCMOS (Bipolar CMOS).**

**229. Что такое передаточная характеристика? Приведите пример.**

1. **Передаточная характеристика:**
   * Это зависимость выходного сигнала от входного для конкретного элемента или устройства. Она описывает, как система реагирует на изменение входных значений.
2. **Пример:**
   * Например, для операционного усилителя передаточная характеристика может быть линейной, если усилитель работает в линейном режиме, и имеет вид прямой линии на графике Vout=A⋅VinV\_{out} = A \cdot V\_{in}Vout​=A⋅Vin​, где AAA — коэффициент усиления.

**230. Что понимают под статической дисциплиной в цифровой схемотехнике?**

1. **Статическая дисциплина:**
   * Это набор правил, которые регулируют корректную работу цифровых схем при постоянных значениях сигналов (например, в момент времени, когда сигналы не меняются). Она включает в себя правила работы логических элементов, чтобы гарантировать корректное функционирование схемы на стационарных состояниях.

**231. Что такое комплементарная пара транзисторов? Схема для инвертора. Биполярные и КМОП транзисторы. Полупроводники. Конденсаторы. n-МОП и p-МОП транзисторы.**

1. **Комплементарная пара транзисторов:**
   * Это пара транзисторов, один из которых типа n, а другой типа p. Эти транзисторы обычно используются в логических схемах, таких как инверторы и другие элементы, для уменьшения потребления энергии.
2. **Схема для инвертора:**
   * В схеме инвертора используется комплементарная пара: один транзистор n-МОП (проводит при логическом "0"), другой p-МОП (проводит при логическом "1").
3. **Биполярные и КМОП транзисторы:**
   * **Биполярные:** используют два типа носителей заряда (электроны и дырки).
   * **КМОП:** используют только один тип носителей (электроны или дырки), что делает их более энергоэффективными.
4. **Полупроводники:**
   * Материалы, которые проводят ток в зависимости от условий (например, кремний).
5. **Конденсаторы:**
   * Хранят электрический заряд и используются для сглаживания напряжений или временных задержек.
6. **n-МОП и p-МОП транзисторы:**
   * **n-МОП:** Проводят ток, когда на управляющем электроде подается высокое напряжение.
   * **p-МОП:** Проводят ток при низком напряжении на управляющем электроде.

**232. Какие встречаются транзисторы? Чем они отличаются?**

1. **Виды транзисторов:**
   * **Биполярные:** Используют два типа носителей заряда (электроны и дырки). Основные виды: NPN, PNP.
   * **Полевые транзисторы (MOSFET):** Используют электрическое поле для управления проводимостью. Основные виды: n-МОП, p-МОП.
2. **Отличия:**
   * **Биполярные:** Высокая скорость переключения, но потребляют больше энергии.
   * **MOSFET:** Энергоэффективны, особенно в интегральных схемах.

**233. В чем отличие идеального от реального источника тока?**

1. **Идеальный источник тока:**
   * Предоставляет постоянный ток, независимо от изменений нагрузки и напряжения.
2. **Реальный источник тока:**
   * Имеет ограничение по напряжению и может изменять выходной ток в зависимости от изменения сопротивления или напряжения на нагрузке.

**234. Как конденсаторы используются в вычислительных системах?**

1. **Использование в вычислительных системах:**
   * Конденсаторы используются для сглаживания пиков напряжения, фильтрации сигналов и создания временных задержек. В динамических системах памяти, например, DRAM, конденсаторы служат для хранения информации в виде заряда.

**235. В чем отличие идеального от реального источника напряжения?**

1. **Идеальный источник напряжения:**
   * Обеспечивает постоянное напряжение независимо от изменения нагрузки или тока.
2. **Реальный источник напряжения:**
   * Может изменять напряжение в зависимости от изменений в цепи, например, из-за сопротивления или изменений нагрузки.

**236. Что такое операционные усилители? Как они используются в современных вычислительных системах?**

1. **Операционные усилители (ОУ):**
   * Это усилители с высокой входной импедансой и низкой выходной импедансой, которые используются для выполнения математических операций (сложение, вычитание, интеграция, дифференцирование).
2. **Использование:**
   * ОУ используются в фильтрах, усилителях, генераторах сигналов и в других устройствах, где требуется усиление и обработка сигналов.

**237. В чем преимущество использования матриц операционных усилителей для вычисления?**

1. **Преимущество:**
   * Матрицы операционных усилителей позволяют эффективно реализовать параллельные операции, такие как сложение и вычитание сигналов в большом объеме, что ускоряет вычислительные процессы, особенно в системах обработки сигналов и в алгоритмах обработки данных.

**238. Какие основные операции реализуют при помощи операционных усилителей?**

1. **Основные операции:**
   * Интеграция.
   * Дифференциация.
   * Сложение и вычитание сигналов.
   * Усиление сигналов.
   * Преобразования импеданса (например, преобразование сопротивления).

**239. Почему по возможности стоит при проектировании использовать элементы OR-NOT или AND-NOT?**

1. **Преимущество:**
   * Использование вентилей OR-NOT (NOR) и AND-NOT (NAND) предпочтительно, поскольку они являются универсальными логическими элементами. Это означает, что все остальные логические функции (например, AND, OR, NOT) можно реализовать с помощью только этих двух типов элементов, что упрощает проектирование.

**240. Что влияет на потребляемую мощность цифровых схем? Из чего она складывается?**

1. **Факторы, влияющие на потребляемую мощность:**
   * Напряжение питания.
   * Частота работы.
   * Тип логических элементов.
   * Степень загрузки схемы.
2. **Компоненты потребляемой мощности:**
   * **Динамическая мощность:** Происходит при переключении состояний элементов.
   * **Статическая мощность:** Происходит из-за тока утечек в элементах.

**241. Что дает использование псевдо n-МОП-логики? В чем недостатки?**

1. **Преимущество:**
   * Псевдо n-МОП-логика сочетает низкое потребление энергии с более высокой скоростью переключения по сравнению с традиционными n-МОП логическими элементами. Это делает схемы более эффективными в некоторых случаях.
2. **Недостатки:**
   * Потери в эффективности из-за дополнительных сложностей в проектировании и более высокие требования к компоновке.

**242. Чем отличается псевдо n-МОП-логика от классического варианта?**

1. **Псевдо n-МОП-логика:**
   * В псевдо n-МОП-логике используется один тип транзисторов (n-МОП), в отличие от классической схемы, где используются как n-МОП, так и p-МОП транзисторы.
2. **Отличия:**
   * **Псевдо n-МОП-логика:** Чаще всего применяется в схемах с низким энергопотреблением, но имеет ограничения по скорости и функциональности.
   * **Классическая n-МОП логика:** Более универсальна, так как сочетает два типа транзисторов (n-МОП и p-МОП), что позволяет достигать лучшей производительности и меньшего энергопотребления при высокой скорости.

**243. Что такое Z состояние у цифровой микросхемы? Как оно реализуется?**

1. **Z состояние:**
   * Это высокоимпедансное состояние, при котором выход микросхемы не оказывает воздействия на цепь (выход не подключен к логическому уровню). Часто используется для создания триовходных шин, где несколько устройств могут делить одну шину.
2. **Реализация:**
   * Z состояние обеспечивается через специальные буферы с высоким сопротивлением, что позволяет предотвратить влияние одного устройства на шину, когда оно не активно.

**244. Как подразделяют цифровые схемы? В чем их особенность в формировании выходного сигнала?**

1. **Подразделение схем:**
   * **Комбинационные схемы:** Выходной сигнал зависит только от текущих входных сигналов. Примеры: логические вентили, мультиплексоры.
   * **Последовательностные схемы:** Выходной сигнал зависит как от текущих, так и от предыдущих входных сигналов, что требует хранения информации. Примеры: триггеры, регистры, конечные автоматы.

**245. Каковы признаки комбинационных цифровых схем (цепей)?**

1. **Признаки комбинационных схем:**
   * Выход зависит только от текущих входных сигналов.
   * Не имеют памяти, не запоминают предыдущее состояние.
   * Примеры: логические элементы, мультиплексоры, дешифраторы.

**246. Правила изображения цифровых схем (цепей)?**

1. **Правила изображения:**
   * Схема должна быть упрощенной и понятной, с четкими метками на входах, выходах и соединениях.
   * Логические элементы должны быть обозначены стандартными символами.
   * Важные элементы должны быть расположены так, чтобы минимизировать пересечения проводов.

**247. Зачем перемещать инверсию? И каковы правила этой операции?**

1. **Перемещение инверсии:**
   * Инверсию можно перемещать для упрощения логических выражений, например, с помощью законов де Моргана. Это позволяет преобразовать логические выражения в более простые и эффективные формы.
2. **Правила:**
   * Применяются преобразования, такие как:
     + ¬(A∧B)=¬A∨¬B\neg (A \land B) = \neg A \lor \neg B¬(A∧B)=¬A∨¬B (Закон де Моргана)
     + ¬(A∨B)=¬A∧¬B\neg (A \lor B) = \neg A \land \neg B¬(A∨B)=¬A∧¬B

**248. Где используется входы с Z состоянием? Почему это мало используется в современных вычислительных системах?**

1. **Использование Z состояния:**
   * Используется в многоканальных системах, таких как шины данных, где несколько устройств могут подключаться к одной линии и не мешать друг другу при отсутствии активной передачи данных.
2. **Почему редко используется:**
   * Требует дополнительных компонентов для управления состоянием и может создавать сложности в синхронизации сигналов.

**249. Что значит микросхема с тремя состояниями?**

1. **Микросхема с тремя состояниями:**
   * Микросхема может быть в одном из трех состояний: логическое "1", логическое "0" или высокоимпедансное состояние (Z), при котором она не взаимодействует с внешней цепью.

**250. Для чего используется символ X при анализе или синтезе цифровых схем?**

1. **Использование символа X:**
   * Символ "X" используется для обозначения неопределенного или неизвестного состояния на входе или выходе схемы. Это может происходить в случае многозначных логических операций или при наличии ошибок в логическом выражении.

**251. Какие варианты могут быть использованы при проектировании комбинационной логики?**

1. **Варианты проектирования:**
   * **Минимизация логических выражений:** Использование карт Карно, булевых алгебраических преобразований.
   * **Использование готовых логических элементов:** Применение стандартных вентилей (AND, OR, NOT) для реализации требуемых функций.
   * **Использование ПЛИС (Программируемых логических интегральных схем):** Программирование схем в соответствии с требуемыми характеристиками.

**252. Какие временные характеристики определяют у цифровых микросхем?**

1. **Временные характеристики:**
   * **Задержка распространения:** Время, необходимое для того, чтобы сигнал от входа дошел до выхода через все промежуточные элементы.
   * **Время отклика (реакции):** Время, необходимое для того, чтобы выход микросхемы изменился в ответ на изменение входных сигналов.
   * **Частота переключений:** Максимальная частота, с которой микросхема может переключаться между состояниями.

**253. Нарисовать временной график переключения цифровой микросхемы.**

1. **Временной график:**
   * Он обычно включает оси для времени и логических уровней (0 и 1). В нем изображается переход сигналов на входах и выходах, показывая временные задержки, пики и спады.

**254. По какой причине возникают импульсные помехи? Как можно этого избежать?**

1. **Причины импульсных помех:**
   * Импульсные помехи возникают из-за резких изменений в электрических цепях, таких как переключение больших токов или напряжений.
2. **Как избежать:**
   * Использование фильтров, экранирования, а также проектирование с учетом минимизации переходных процессов.

**255. Чем отличается задержка распространения от задержки реакции?**

1. **Задержка распространения:**
   * Это время, которое требуется сигналу для того, чтобы пройти через цепь от входа до выхода.
2. **Задержка реакции:**
   * Это время, которое требуется элементу для того, чтобы откликнуться на изменение входных данных.

**256. Что такое критический путь в цифровой схеме? Почему важно обращать внимание на это при проектировании?**

1. **Критический путь:**
   * Это наибольшее время задержки в цепи, через которую проходят все сигналы. Этот путь определяет максимальную скорость работы всей схемы.
2. **Значение:**
   * Важно, чтобы критический путь был как можно короче, так как именно он ограничивает максимальную рабочую частоту схемы. Для улучшения производительности необходимо минимизировать критический путь.

**257. От чего может зависеть выбор варианта реализации цифровой схемы?**

1. **Зависимости:**
   * **Сложность логики:** Простота схемы и ее реализация могут зависеть от доступных ресурсов и требований к производительности.
   * **Потребление энергии:** Разные схемы могут потреблять разное количество энергии.
   * **Скорость работы:** Важно учитывать требования по скорости выполнения операций.
   * **Надежность:** Требования к надежности и устойчивости схемы в условиях различных помех.
   * **Производственные ограничения:** Например, технологии изготовления или доступность компонентов.

**258. Приведите пример временной диаграммы работы цифровой схемы.**

1. **Пример временной диаграммы:**
   * Временная диаграмма может показать изменение уровня логических сигналов (например, "0" и "1") на разных входах и выходах цифровой схемы с течением времени. Она может также включать временные задержки между сигналами.

**259. Что включают базовые комбинационные блоки?**

1. **Базовые комбинационные блоки:**
   * Это такие элементы, как логические вентили (AND, OR, NOT), мультиплексоры, дешифраторы, кодировщики, демультиплексоры, которые выполняют операции над входными сигналами для формирования выходных сигналов.

**260. Что такое мультиплексор? Как он обозначается на схемах?**

1. **Мультиплексор:**
   * Мультиплексор (MUX) — это логическая схема, которая принимает несколько входных сигналов и передает один из них на выход в зависимости от состояния управляющего сигнала (выборки).
2. **Обозначение:**
   * На схемах мультиплексор обычно изображается прямоугольной коробкой с несколькими входами и одним выходом. Управляющие входы также обозначаются.

**261. Область применения мультиплексоров и демультиплексоров?**

1. **Мультиплексоры:**
   * Мультиплексоры широко используются в системах, где необходимо выбирать один из нескольких источников данных для передачи через один канал. Например, в шинах данных.
2. **Демультиплексоры:**
   * Демультиплексоры выполняют противоположную операцию: они принимают один сигнал и распределяют его на несколько выходов в зависимости от управляющих сигналов. Используются в системах, где один источник данных должен быть разделен на несколько целевых каналов.

**262. Что такое дешифратор? Как он обозначается на схемах?**

1. **Дешифратор:**
   * Это схема, которая преобразует бинарный код на входах в активный сигнал на одном из нескольких выходов. Например, дешифратор может принимать 2-битный код и активировать один из четырех выходов.
2. **Обозначение:**
   * На схемах дешифратор обозначается прямоугольной коробкой с несколькими входами и выходами, где количество выходов обычно больше, чем количество входов.

**263. Что такое демультиплексор? Как он обозначается на схемах?**

1. **Демультиплексор:**
   * Это схема, которая принимает один сигнал и направляет его на один из нескольких выходов в зависимости от управляющего сигнала.
2. **Обозначение:**
   * На схемах демультиплексор изображается как элемент с одним входом и несколькими выходами, с управляющими входами для выбора выхода.

**264. Область применения дешифраторов?**

1. **Область применения дешифраторов:**
   * Дешифраторы применяются в цифровых устройствах для выделения определенных сигналов в зависимости от входного бинарного кода. Например, в системах адресации памяти, в кодировании/декодировании сигналов для управления устройствами.

**265. Какие есть подходы к проектированию последовательной логики?**

1. **Подходы к проектированию:**
   * **Статическая последовательная логика:** Схемы, в которых состояние системы зависит от внешних сигналов и предшествующего состояния.
   * **Динамическая последовательная логика:** Схемы, в которых состояние системы изменяется с течением времени, обычно с использованием тактовых сигналов.

**266. Какие есть типы триггеров?**

1. **Типы триггеров:**
   * **RS-триггер:** Имеет два входа — Reset и Set, и может хранить одно из двух состояний.
   * **D-триггер:** Использует вход данных и тактовый сигнал для переключения состояния.
   * **JK-триггер:** Развивает возможности RS-триггера с улучшенной работой при различных входных состояниях.
   * **T-триггер:** Используется для переключения состояния на каждом такте.

**267. Что такое D-Триггер? И почему преимущество синтеза схем на D-Триггер а не на RS?**

1. **D-Триггер:**
   * D-Триггер имеет один вход данных (D) и тактовый вход (Clock). Его состояние зависит от входа данных на момент тактового импульса.
2. **Преимущество:**
   * D-триггер проще в проектировании и синтезе, так как не имеет таких состояний, как у RS-триггера (неопределенность), и гарантирует стабильное поведение.

**268. Что отличает последовательные схемы от комбинационных?**

1. **Отличие последовательных схем от комбинационных:**
   * **Комбинационные схемы:** Выход зависит только от текущих входных значений.
   * **Последовательные схемы:** Выход зависит от входных значений и состояния, которое сохраняется в памяти (например, в триггерах).

**269. Нарисовать временной график срабатывания по фронту и по спаду цифровой схемы.**

1. **Временной график:**
   * В графике отображаются временные интервалы, где сигнал на выходе схемы изменяется на фронте (возрастающий) или на спаде (убывающий) тактового импульса.

**270. Что лежит в основе построения RS-триггера?**

1. **Основы RS-триггера:**
   * RS-триггер состоит из двух логических элементов (например, NOR или NAND), которые реализуют два состояния: Set (установить) и Reset (сбросить). Входы триггера управляют его состоянием.

**271. Какие есть состояния у RS-триггера?**

1. **Состояния RS-триггера:**
   * **Set:** Триггер установлен в логическое "1".
   * **Reset:** Триггер сброшен в логическое "0".
   * **Неопределенное состояние:** При одновременном активировании входов Set и Reset, что не рекомендуется использовать.

**272. Что такое регистр? Какие они бывают?**

1. **Регистр:**
   * Это устройство для хранения данных в цифровых системах. Регистр может хранить несколько битов данных.
2. **Типы регистров:**
   * **Сдвиговый регистр:** Выполняет операции сдвига данных.
   * **Параллельный регистр:** Позволяет записывать и считывать все биты одновременно.
   * **Сдвигово-регистровый:** Комбинирует сдвиг и параллельную запись.

**273. Чем защелка отличается от триггера?**

1. **Отличие защелки от триггера:**
   * **Защелка:** Сохраняет состояние до следующего тактового импульса, но не обязательно на основе такта.
   * **Триггер:** Обновляет свое состояние только по тактовому импульсу, является более стабильным.

**274. Чем определяется схема (цепь)? Что характеризуют параметры?**

1. **Определение схемы:**
   * Схема определяется как совокупность элементов (вентили, триггеры, регистры), соединенных между собой проводниками для выполнения определенной функции.
2. **Параметры:**
   * **Тип схемы:** Комбинационная или последовательная.
   * **Входы и выходы:** Количество и тип сигналов.
   * **Логическая функция:** Задача, которую выполняет схема.
   * **Время отклика:** Скорость реакции на изменение входных сигналов.
   * **Энергопотребление:** Количество энергии, потребляемое схемой.

**275. Зачем нужны триггеры с функцией сброса и как они обозначаются?**

1. **Триггеры с функцией сброса:**
   * Эти триггеры могут быть сброшены в начальное состояние по специальному сигналу, независимо от других входных сигналов. Это полезно для инициализации состояния схемы.
2. **Обозначение:**
   * Обычно на схемах триггер с функцией сброса обозначается специальным входом, помеченным как "Reset" или "Clr".

**276. Зачем нужны триггеры с функцией разрешения и как они обозначаются?**

1. **Триггеры с функцией разрешения:**
   * Эти триггеры требуют дополнительного сигнала разрешения для выполнения изменения состояния. Без разрешающего сигнала триггер не меняет свое состояние, что важно для синхронизации с другими частями системы.
2. **Обозначение:**
   * На схемах обычно используется обозначение "Enable" или "E", которое указывает на вход разрешения.

**277. Чем отличаются синхронные и асинхронные схемы вычислительных систем? Достоинства и недостатки.**

1. **Синхронные схемы:**
   * В синхронных схемах все изменения состояния происходят в такт с тактовым сигналом.
   * **Достоинства:** Простота проектирования, высокая предсказуемость работы.
   * **Недостатки:** Требования к точности тактового сигнала, ограничение частоты работы.
2. **Асинхронные схемы:**
   * В асинхронных схемах изменения происходят по мере поступления сигналов, без привязки к тактовому сигналу.
   * **Достоинства:** Нет зависимости от тактового сигнала, высокая гибкость.
   * **Недостатки:** Более сложное проектирование, возможно возникновение метастабильности.

**278. Синхронные последовательные схемы? Зачем они нужны?**

1. **Синхронные последовательные схемы:**
   * Это схемы, в которых состояния изменяются синхронно с тактовым сигналом.
   * **Зачем нужны:** Они обеспечивают стабильную и предсказуемую работу системы, так как все изменения происходят в определенный момент времени, что упрощает управление сложными процессами.

**279. Что такое конечный автомат? Какие еще названия встречаются этой технологии?**

1. **Конечный автомат:**
   * Это математическая модель, которая описывает поведение системы с конечным количеством состояний. В зависимости от входных сигналов автомат переходит из одного состояния в другое.
   * **Другие названия:** Машина состояний, FSM (Finite State Machine).

**280. Алгоритм проектирования конечного автомата Мура?**

1. **Алгоритм проектирования конечного автомата Мура:**
   * **Шаг 1:** Определение состояний и их переходов.
   * **Шаг 2:** Назначение выходов для каждого состояния.
   * **Шаг 3:** Построение таблицы состояний и таблицы переходов.
   * **Шаг 4:** Реализация схемы на основе таблицы состояний.

**281. Алгоритм проектирования конечного автомата Мили?**

1. **Алгоритм проектирования конечного автомата Мили:**
   * **Шаг 1:** Определение состояний и возможных переходов между ними.
   * **Шаг 2:** Присвоение выходных сигналов не каждому состоянию, а каждому переходу.
   * **Шаг 3:** Построение таблицы состояний с соответствующими выходами для каждого перехода.
   * **Шаг 4:** Реализация схемы с учетом выходов на каждом переходе.

**282. Что указывается в узлах, а что около стрелок при построении графов?**

1. **В узлах указывается:**
   * Состояние автомата.
2. **Около стрелок указывается:**
   * Условия переходов между состояниями, то есть входные сигналы, при которых происходит переход.

**283. Какие таблицы составляются при синтезе конечного автомата?**

1. **Таблицы:**
   * **Таблица состояний:** Показывает все возможные состояния автомата и их связи с другими состояниями.
   * **Таблица переходов:** Указывает, при каких входных сигналах происходит переход между состояниями.
   * **Таблица выходов:** Содержит информацию о выходных значениях в зависимости от состояния или перехода.

**284. Чем отличается автоматы Мура и Мили?**

1. **Автомат Мура:**
   * Выходы зависят только от текущего состояния автомата.
2. **Автомат Мили:**
   * Выходы зависят не только от состояния, но и от входных сигналов, что делает автомат более гибким.

**285. Какие есть варианты кодирования состояния конечных автоматов? В чем преимущества?**

1. **Варианты кодирования состояния:**
   * **Прямое кодирование:** Каждому состоянию присваивается уникальный двоичный код.
   * **Кодирование с использованием минимальной длины:** Применяется с целью сокращения количества используемых бит.
2. **Преимущества:**
   * Прямое кодирование обеспечивает простоту реализации, но может требовать много битов.
   * Кодирование с минимальной длиной эффективно использует ресурсы, но может потребовать более сложной реализации.

**286. Какой из автоматов дает выигрыш во времени и на сколько?**

1. **Выигрыш во времени:**
   * **Автомат Мили** может дать выигрыш во времени, поскольку его выход зависит не только от состояния, но и от входных сигналов, что позволяет быстрее реагировать на изменения.

**287. Что такое декомпозиция конечных автоматов?**

1. **Декомпозиция конечных автоматов:**
   * Это процесс разбиения сложного конечного автомата на несколько более простых автоматов, каждый из которых выполняет часть задачи. Это упрощает проектирование и повышает эффективность.

**288. Что дает декомпозиция конечных автоматов?**

1. **Преимущества декомпозиции:**
   * Упрощение анализа и синтеза сложных систем.
   * Повышение гибкости и масштабируемости системы.
   * Уменьшение числа состояний в каждом из под-автоматов.

**289. Алгоритм восстановления конечного автомата по электрической схеме.**

1. **Алгоритм восстановления:**
   * **Шаг 1:** Анализ электрической схемы для выявления элементов, соответствующих состояниям и переходам.
   * **Шаг 2:** Построение таблицы состояний и таблицы переходов на основе соединений элементов схемы.
   * **Шаг 3:** Определение выходных сигналов для каждого состояния или перехода.

**290. Зачем нужна синхронизация в вычислительных системах?**

1. **Необходимость синхронизации:**
   * Синхронизация нужна для того, чтобы все компоненты системы работали согласованно, что предотвращает ошибки и гарантирует правильную работу при взаимодействии различных частей системы.

**291. Что такое временная характеристика работы цифровой системы?**

1. **Временная характеристика:**
   * Это зависимость состояния выходного сигнала от времени, которая характеризует поведение цифровой системы при изменении входных сигналов.

**292. Что такое расфазировка тактовых сигналов?**

1. **Расфазировка тактовых сигналов:**
   * Это ситуация, когда два или более тактовых сигнала, которые должны быть синхронизированы, оказываются смещенными по фазе. Это может привести к сбоям в работе системы, особенно в сложных многозадачных вычислительных системах.

**293. Почему для синхронизации современных микропроцессоров учитывают физическое расположение отдельных элементов?**

1. **Причины:**
   * В современных микропроцессорах элементы могут находиться на разных расстояниях друг от друга, и время распространения сигнала зависит от этого расстояния. Для оптимальной синхронизации важно учитывать эти задержки, чтобы избежать метастабильных состояний и нарушений в работе системы.

**294. Почему частота современных микропроцессоров стала расти медленно? Как увеличивают производительность современных вычислительных устройств?**

1. **Причины замедления роста частоты:**
   * Увеличение частоты приводит к повышенному энергопотреблению и тепловым проблемам, что затрудняет дальнейшее повышение производительности только за счет частоты.
2. **Методы увеличения производительности:**
   * Использование многоядерных процессоров, улучшение архитектуры, повышение эффективности инструкций, использование специализированных вычислительных блоков, таких как графические процессоры (GPU) или тензорные ядра.

**295. Что такое динамическая дисциплина? В чем его суть? И почему следуют таким правилам?**

1. **Динамическая дисциплина:**
   * Это подход к проектированию цифровых схем, при котором обеспечивается стабильность работы системы в условиях изменения внешних факторов, таких как температура и напряжение питания.
2. **Суть:**
   * Включает в себя стратегии, направленные на минимизацию ошибок и сбоев в работе схемы за счет адаптивного управления параметрами системы.

**296. Что такое апертурное время последовательностных схем? Из чего оно состоит?**

1. **Апертурное время:**
   * Это время, в течение которого входной сигнал может изменяться, прежде чем система завершит обработку его изменения. Оно состоит из времени, необходимого для перехода через состояние, и времени распространения сигнала.

**297. Период тактовых импульсов. Что влияет на выбор его значения?**

1. **Факторы, влияющие на выбор периода тактовых импульсов:**
   * Скорость работы логических элементов, задержки распространения сигналов, требования к синхронизации, энергопотребление, требования к точности.

**298. Что такое расфазировка и по какой причине она может возникать?**

1. **Причины расфазировки:**
   * Расфазировка может возникать из-за различных временных задержек в компонентах системы, различных физических характеристик проводников или изменений в частоте работы.

**299. Что такое метастабильность? Когда она нужна? И как ее обеспечивают?**

1. **Метастабильность:**
   * Это состояние, когда цифровой элемент находится в неопределенном состоянии, не соответствующем логическому уровню (0 или 1), и может оставаться в этом состоянии до тех пор, пока не произойдут дополнительные события.
2. **Когда она нужна:**
   * Метастабильность используется в некоторых схемах для временной стабилизации и синхронизации, например, в синхронизаторах.
3. **Обеспечение метастабильности:**
   * Для минимизации метастабильности используются специальные элементы, такие как двуступенчатые синхронизаторы, которые помогают стабилизировать сигнал перед его использованием в логике.

**300. Что такое регистр?**

1. **Регистр:**
   * Регистр — это элемент цифровой схемы, предназначенный для хранения данных (обычно в виде битов). Регистры могут быть использованы для временного хранения данных в процессе обработки, а также для передачи данных между различными частями системы.

**301. Какие бывают регистры?**

1. **Виды регистров:**
   * **Сдвиговые регистры:** Используются для сдвига битов данных.
   * **Параллельные регистры:** Могут загружать и извлекать данные параллельно по всем битам.
   * **Циклические регистры:** Данные в циклических регистрах циркулируют по регистру после определенного времени.

**302. Как обозначают регистры на электрических схемах в разных стандартах?**

1. **Обозначения:**
   * На схемах регистры обычно обозначаются прямоугольниками с несколькими входами и выходами, где каждый вывод представляет собой один бит данных. Обозначения могут варьироваться в зависимости от стандарта, но принцип остается схожим.

**303. Зачем нужны мультиплексоры?**

1. **Нужны для:**
   * Мультиплексоры (MUX) позволяют выбрать один из нескольких входных сигналов и передать его на выход. Они часто используются для многократного выбора и передачи данных через один канал, что снижает потребность в дополнительных проводах и упрощает схему.

**304. Зачем нужны демультиплексоры?**

1. **Нужны для:**
   * Демультиплексоры (DEMUX) выполняют обратную операцию мультиплексора: принимают один сигнал и направляют его на один из нескольких выходов в зависимости от управляющего сигнала.

**305. Как обозначаются мультиплексоры в разных стандартах?**

1. **Обозначения:**
   * На электрических схемах мультиплексоры часто изображаются как прямоугольники с несколькими входами и одним выходом. Управляющий сигнал или сигналы также отображаются как входы.

**306. Какие есть регистры процессора?**

1. **Типы регистров процессора:**
   * **Общие регистры:** Используются для хранения операндов и промежуточных результатов.
   * **Регистр состояния:** Содержит флаги, такие как флаг переноса или флаг нуля.
   * **Регистр счетчика команд:** Хранит адрес следующей инструкции для выполнения.
   * **Регистр данных и адреса:** Используются для хранения данных и адресов в памяти.

**307. Зачем нужны регистры процессора?**

1. **Назначение регистров процессора:**
   * Регистры процессора служат для быстрого хранения данных и промежуточных результатов в ходе выполнения программ. Они обеспечивают быстрый доступ к данным, что значительно ускоряет выполнение операций по сравнению с обращением к основной памяти.

**308. Предназначение шин в микропроцессорных системах?**

1. **Назначение шин:**
   * Шины в микропроцессорных системах служат для передачи данных, адресов и управляющих сигналов между различными компонентами системы, такими как процессор, память и устройства ввода-вывода.

**309. Виды шин микропроцессорных систем?**

1. **Виды шин:**
   * **Адресная шина:** Для передачи адресов памяти и устройств.
   * **Данные шина:** Для передачи данных между процессором и другими компонентами.
   * **Управляющая шина:** Для передачи управляющих сигналов, таких как сигналы чтения/записи и сигналы синхронизации.

**310. Что такое кэш микропроцессора?**

1. **Кэш микропроцессора:**
   * Это специализированная память, которая используется для хранения часто запрашиваемых данных и инструкций, чтобы ускорить доступ к ним и повысить общую производительность процессора.

**311. Чем различаются кэши микропроцессорных систем?**

1. **Различия в кэшах:**
   * **L1 Cache (уровень 1):** Это самый быстрый кэш, находящийся непосредственно в процессоре. Он хранит данные и инструкции, к которым процессор обращается наиболее часто.
   * **L2 Cache (уровень 2):** Кэш с большей емкостью, который часто используется для хранения более общих данных, но имеет немного больший доступный латентность по сравнению с L1.
   * **L3 Cache (уровень 3):** Этот кэш часто используется в многозадачных процессорах и хранит еще большие объемы данных с меньшей скоростью доступа, чем L1 и L2.

**312. Что такое суперскалярная архитектура?**

1. **Суперскалярная архитектура:**
   * Это архитектура процессора, которая позволяет выполнять несколько инструкций за один такт процессора, улучшая параллельную обработку данных и увеличивая производительность. Процессоры с суперскалярной архитектурой могут одновременно декодировать и исполнять несколько инструкций.

**313. В чем особенность суперскалярной архитектуры?**

1. **Особенность:**
   * Главной особенностью суперскалярной архитектуры является способность процессора выполнять несколько инструкций в параллели за один цикл, что повышает его производительность, особенно при выполнении сложных вычислений и многозадачности.

**314. Зачем в компьютере предсказатели переходов?**

1. **Предсказатели переходов:**
   * Предсказатели переходов используются для увеличения производительности процессора за счет предсказания результатов условных переходов (ветвлений) в программах. Это позволяет заранее загрузить и выполнить инструкции, которые, вероятно, будут выполнены, снижая время ожидания.

**315. Зачем используют иерархию памяти?**

1. **Использование иерархии памяти:**
   * Иерархия памяти в вычислительных системах используется для балансировки между скоростью и стоимостью хранения данных. Высокоскоростная, но дорогая память (например, кэш) используется для хранения часто используемых данных, в то время как более медленная и дешевле память (например, жесткие диски или SSD) хранит данные, к которым обращаются реже.

**316. Что такое гетерагенные вычисления?**

1. **Гетерагенные вычисления:**
   * Гетерагенные вычисления — это вычисления, использующие различные типы процессоров и вычислительных блоков (например, центральные процессоры (CPU) и графические процессоры (GPU)) для выполнения различных задач, оптимизируя процесс на основе того, какой тип вычислительной мощности наиболее эффективен для данной задачи.

**317. FPGA-акселератор. Что это такое?**

1. **FPGA-акселератор:**
   * FPGA-акселератор — это аппаратное решение, использующее программируемые логические устройства (FPGA), которые могут быть настроены для выполнения специфических вычислительных задач с высокой эффективностью, таких как обработка сигналов, криптография или машинное обучение.

**318. Сфера применения FPGA-акселераторов.**

1. **Сфера применения:**
   * FPGA-акселераторы применяются в области обработки больших данных, искусственного интеллекта, криптографии, научных вычислений, телекоммуникаций и других областях, где требуется высокая производительность при низкой задержке.

**319. Особенность квантовых вычислений?**

1. **Особенности квантовых вычислений:**
   * Квантовые вычисления используют принципы квантовой механики, такие как суперпозиция и запутанность, для решения задач, которые невозможно эффективно решить с помощью традиционных компьютеров. Квантовые алгоритмы могут теоретически решать определенные задачи (например, факторизацию больших чисел) гораздо быстрее, чем классические алгоритмы.

**320. В каком прикладном направлении могут помочь квантовые вычисления?**

1. **Прикладные направления квантовых вычислений:**
   * Квантовые вычисления могут быть полезны в области криптографии, моделировании молекул и химических реакций, оптимизации сложных систем, машинном обучении и других областях, где требуется обработка огромных объемов данных с высокой вычислительной нагрузкой.

**321. Что такое тензорные ядра?**

1. **Тензорные ядра:**
   * Тензорные ядра — это специализированные вычислительные блоки, предназначенные для эффективного выполнения операций с тензорами, которые широко используются в машинном обучении, особенно для обработки и обучения нейронных сетей.

**322. Что такое вычисление со смешанной точностью?**

1. **Вычисление со смешанной точностью:**
   * Это подход, при котором в процессе вычислений используются различные типы числовых представлений, например, операции с числами с плавающей запятой различной точности (например, 16-битные и 32-битные числа), что позволяет достичь оптимального баланса между производительностью и точностью вычислений.

**323. Сфера применения тензорных ядер?**

1. **Сфера применения тензорных ядер:**
   * Тензорные ядра в первую очередь используются для ускорения операций в области искусственного интеллекта и машинного обучения, включая обучение нейронных сетей и инференс, где важна высокая производительность при работе с большими массивами данных.

**324. Когда эффективно использование тензорных вычислений?**

1. **Эффективность тензорных вычислений:**
   * Тензорные вычисления наиболее эффективны в приложениях, связанных с искусственным интеллектом, глубоким обучением и нейронными сетями, где требуется обработка многомерных массивов данных с высокой производительностью и низкой задержкой.

[**подписка сюда быстро**](https://github.com/m0loko)