|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| 1. Системы счисления. Перевод чисел из одной системы счисления в другую. Метод преобразования из одной системы счисления в другую. Метод преобразования с использованием весов разрядов.  Система счисления – это знаковая система, в которой числа записываются по определенным правилам с помощью символов некоторого алфавита, называемых цифрами.  Методы преобразований:  1) преобразования с использованием весов разрядов в исходной и в искомой записи числа;  2) деления (умножения) на новое основание;  3) с использованием особого соотношения заданной и искомой систем счисления.  Метод преобразования с использованием весов разрядов имеет 2 разновидности в зависимости от того, какая система счисления (исходная или искомая) является более привычной. Если более привычной является искомая система, то на основании расширенной записи исходного числа подсчитываются значения ее отдельных разрядов в новой системе счисления. Далее полученные значения суммируются.  При преобразовании правильных дробей в принципе используется тот же подход, но при расчете весов отдельных разрядов берутся отрицательные степени основания счисления. | 2. Перевод чисел из одной системы счисления в другую. Метод деления (умножения) на новое основание.  Система счисления – это знаковая система, в которой числа записываются по определенным правилам с помощью символов некоторого алфавита, называемых цифрами.  Методы преобразований:  1) преобразования с использованием весов разрядов в исходной и в искомой записи числа;  2) деления (умножения) на новое основание;  3) с использованием особого соотношения заданной и искомой систем счисления.  Метод деления (умножения) имеет 2 разновидности:  1.Преобразование целых чисел  *Пример:* найти запись в двоичной форме десятичного числа *N*10= 27.  *Решение:*  Делим сначала исходное число, а затем получаемые частные на значение нового основания 2 до получения частного со значением, меньше, чем 2:  27/2 → int(27/2) = 13 и rest (27/2) = 1;  13/2→int(13/2) = 6 и rest (13/2) = 1;  6/2 → int(6/2) = 3 и rest (6/2) = 0;  3/2 → int(3/2) = 1 и rest (3/2) = 1.  Таким образом: 436 = 11 0110100.  2.Преобразование дробных чисел  *Пример:* найти запись в двоичной форме десятичного числа *M*10= 0,7.  *Решение:*  Определяем количество разрядов числа *M*2. Так как исходная запись числа содержит один десятичный разряд, то запись данного числа в двоичном основании должна содержать четыре разряда. Учитывая округление, ищется предварительный двоичный эквивалент с пятью разрядами.  Умножаем исходное число *M*10, а затем дробные части последовательно получаемых произведений на новое основание 2. Выполняется пять таких операций умножения, в результате получаем:  0,7 *×*2 = 1,4 (int(0,7*×*2) = 1 и DF (0,7*×*2) = 0,4);  0,4 *×*2 = 0,8 (int(0,4*×*2) = 0 и DF(rest (0,4*×*2) = 0,8);  0,8 *×*2 = 1,6 (int(0,8*×*2) = 1 и DF(rest (0,8*×*2) = 0,6);  0,6 *×*2 =1,2 (int(0,6*×*2) = 1 и DF(rest (0,6*×*2) = 0,2);  0,2 *×*2 = 0,4 (int(0,2*×*2) = 0 и DF(rest (0,2*×*2) = 0,4).  Таким образом, 0,7 = 0,10110, а окончательный рез-т перехода в двоичную систему будет 0,710= 0,10112. | 3. Перевод чисел из одной системы счисления в другую. Метод с использованием особого соотношения оснований исходной и искомой систем счисления.  Система счисления – это знаковая система, в которой числа записываются по определенным правилам с помощью символов некоторого алфавита, называемых цифрами.  Методы преобразований:  1) преобразования с использованием весов разрядов в исходной и в искомой записи числа;  2) деления (умножения) на новое основание;  3) с использованием особого соотношения заданной и искомой систем счисления.  Метод с использованием особого соотношения оснований исходной и искомой систем счисления применим в тех случаях, когда исходное *q*1 и новое *q*2 основания могут быть связаны через целую степень, т.е. когда выполняются условия: *q*1*m* = *q*2 (*условие 1*) или *q*2*m* = *q*1 (*условие 2*). Если имеет место *условие 2*, то для заданного в системе с основанием *q*1 числа *Nq*1 *= аn аn*-1 *аn*-2... *а*1*а*0 запись его в системе в новом основании *q*2 определяется следующим образом:  - каждому разряду *ai* исходной записи числа ставится в соот-вие его *m-*разрядный эквивалент в системе счисления с основанием *q*2;  - искомая запись всего заданного числа формируется за счет объединения всех полученных *m*-разрядных групп.  ***Пример:*** найти двоичный эквивалент 8-ричного числа 67401.648.  *Решение*  Основания исходной и новой систем счисления можно выразить через целую степень:23 = 8.  Поэтому применяем третий метод для случая перехода из системы с большим основанием в систему с меньшим основанием. Ставим в соответствие каждой цифре исходной записи числа трехразрядный двоичный код (*триаду*):  6 7 4 0 1 6 4  110 111 100 000 001 110 100  Формируем окончательный результат посредством объединения полученных трехразрядных двоичных чисел в единый двоичный эквивалент:  67401.648 = 110111100000001.110100. | 4. Арифметические операции над двоичными числами. Операция сложения и вычитания в двоичной системе исчисления.  При выполнении любой операции результат ищется согласно соответствующим правилам, которые удобно представлять в табличной форме, где для всех возможных комбинаций значений одноразрядных операндов приводятся значения результата.  ***Правила сложения и вычитания в двоичной системе счисления***   |  |  |  |  |  |  |  | | --- | --- | --- | --- | --- | --- | --- | | + | 0 | 1 |  | - | 0 | 1 | | 0 | 0 | 1 |  | 0 | 0 | 1 | | 1 | 1 | 0*\** |  | 1 | 1\* | 0 |   Все возможные значения первого слагаемого задаются во второй и третьей строках первой колонки; все возможные значения второго слагаемого – во второй и третьей колонках первой строки. На пересечении отмеченных значениями операндов строк и колонок располагается результат их сложения. В таблице знаком «*\**» отмечен случай, когда в текущем разряде результата получен ноль и имеет место перенос в ближайший старший разряд  *Пример*  https://studfile.net/html/20260/740/html_gatsd6GBaT.XiuA/img-EoAMQ3.png  Звездочкой отмечен случай, когда в текущем разряде получена единица, путем займа из ближайшего старшего разряда.  *Пример.*  \_ 1000111001  0101101101  0011001100 |
| 5. Операция умножения в двоичной системе исчисления.  Умножение одноразрядных двоичных чисел выполняется по следующим правилам: 0 \* 0 = 0; 1 \* 0 = 0; 0 \* 1 = 0; 1 \* 1 = 1  При умножении многоразрядных операндов, используется метод, при котором формирование произведения выполняется за счет суммирования частичных произведений, которые оформляются посредством умножения множимого на отдельные разряды множителя с учетом веса соотв-вующего разряда множителя. **Возможные методы реализации умножения можно классифицировать по 2 признакам:** Начиная с какого выполняется отработка множителя и что сдвигается, следственно, существует 4 вида умножения:   * [Операция умножения в 2-й СС начиная со старшего разряда множителя со сдвигом множимого.](#kix.o2t9t257jg9f) (умножение идет от старшего к младшему разряду 2-го множителя; все частные произведения записываются под разрядом 2-го множителя и после все произведения суммируются). * [Операция умножения в 2-й СС начиная с младшего разряда множителя со сдвигом множимого.](#stxvo1j5s6l4) (умножение идет от младшего к старшему разряду 2-го множителя и происходит по алгоритму умножил-> сложил, результаты частных произведений записываются с разряда 2-го множителя) * [Операция умножения в 2-й СС начиная со старшего разряда множителя со сдвигом промежуточных результатов](#m3a62pn98jqs) (умножение идет от старшего к младшему разряду 2-го множителя; все частные произведения записываются в столбик под младшим разрядом и происходит по алгоритму умножил-> сложил-> сдвинул на 1 вправо).   + - [Операция умножения в 2-й СС начиная с младшего разряда множителя со сдвигом частичных произведений.](#axck4sdhu2wn) | 6. IEEE754. Специальные числа. Зачем нулю знак.  ***IEEE754***  Разработчики «K-C-S» победили и теперь их детище воплотилось в стандарт IEEE754. Числа с плавающей запятой в нем представлены в виде знака (s), мантиссы (M) и порядка (E) следующим образом:  (-1)s × 1.M × 2E  https://habrastorage.org/getpro/habr/post_images/112/403/33b/11240333b7f7b8cc9fe2d3339ca06bfa.gif   * Знак s=0 (положительное число) * Порядок E=011111002-12710 = -3 * Мантисса M = 1.012 (первая единица не явная) * В результате наше число F = 1.012e-3 = 2-3+2-5 = 0,125 + 0,03125 = 0,15625   ***Специальные числа: ноль, бесконечность и неопределенность*** В IEEE754 число «0» -- E=Emin-1 (для single это -127) и нулевой мантиссой. Введение нуля как самостоятельного числа (т.к. в нормализованном представлении нельзя представить ноль) позволило избежать многих странностей в арифметике. И хоть операции с нулем нужно обрабатывать отдельно, обычно они выполняются быстрее, чем с обычными числами. Также в IEEE754 предусмотрено представление для специальных чисел, работа с которыми вызывает исключение. К таким числам относится бесконечность (±∞) и неопределенность (NaN). *Неопределенность* или NaN (от not a number) – это представление, придуманное для того, чтобы арифметическая операция могла всегда вернуть какое-то не бессмысленное значение. В IEEE754 NaN представлен как число, в котором E=Emax+1, а мантисса не нулевая. Как можно получить NaN? Одним из следующих способов: ∞+(- ∞); 0 × ∞; 0/0, ∞/∞; sqrt(x), где x<0.  ***Зачем нулю знак (или +0 vs -0)***  Так, 3·(+0)=+0, а 3·(-0)=-0. Но при сравнении +0=-0. В стандарте знак сохранили умышленно, чтобы выражения, которые в результате переполнения или потери значимости превращаются в бесконечность или в ноль, при умножении и делении все же могли представить максимально корректный результат. Например, если бы у нуля не было знака, выражение 1/(1/x)=x не выполнялось бы верно при x=±∞, так как 1/∞ и 1/-∞ равны 0. | 7. Деление двоичных чисел (общие правила)  Деление - неточная операцией, поэтому при её выполнении прежде всего устанавливается количество разрядов частного, которые подлежат определению. Деление в двоичной системе счисления может выполняться точно так же, как и в десятичной, однако формирования частного двоичных операндов реализуется гораздо проще, чем в десятичной системе, т.к.:  • Упрощается процедура подбора очередной цифры вследствие того, что в двоичной системе очередной цифрой может быть одна из двух - либо 0, либо 1;  • Упрощается процедура умножения найденной цифры частного на делитель. | 8. Деление двоичных чисел с восстановлением остатка  ***Алгоритм деления с восстановлением остатка:***  1. Выполняется пробное вычитание с формированием первого остатка A1=[Дм]доп+[-Дт]доп. Далее, если А1 < 0, то в первый разряд, расположенный слева от запятой, заносится ноль (0), иначе единица (1) – переполнение и переход к пункту 5.  2. Если Аi < 0, то восстанавливаем предыдущий остаток Ai=Ai+[Дт]доп.  3. Формирование очередного остатка. Ai+1=Ai∙2+[-Дт]допесли Ai+1<0, то в очередной разряд частного справа от запятой записывается ноль (Чт(n)=0), иначе записывается единица (Чт(n)=1).  4. Если достигнута заданная точность частного или получен нулевой остаток Ai+1, то процесс деления окончен и осуществляется переход к пункту 5, иначе переходим к пункту 2 алгоритма.  5. Окончание алгоритма.  Из рассмотренного алгоритма видно следующее:  1) необходимо затрачивать время на восстановление остатка;  2) процесс деления нерегулярный, в зависимости от делимого и делителя частное будет содержать нулей больше или меньше, и чем больше нулей, тем больше требуется времени на восстановление остатков.  https://www.ok-t.ru/studopediaru/baza7/1445215431351.files/image386.gif |
| 17. Представление чисел с фиксированной точкой. Арифметические операции над числами, представленными с фиксированной точкой.  ***Представление чисел с фиксированной точкой***  **Арифметические операции над числами, представленными с фиксированной точкой**  К числу основных арифметических операций, непосредственно реализуемых в ЭВМ, относятся операции сложения, умножения, деления. Остальные операции (например, такие, как возведение в степень, извлечение квадратного корня) реализуются программным способом.  Выполнение длинных операций, таких, как умножение и деление, реализуется в два этапа:  • на первом этапе формируется знак искомого результата,  • на втором этапе, используя абсолютные значения операндов, ищем результат  • присваивание результату знак.  Как правило, представлены в прямом коде, и знак результата, не зависимо от того, частное это или произведение, ищется за счет сложения по модулю 2 знаковых разрядов операндов. В результате этого знак результата положителен, если операнды имеют одинаковые знаки, или отрицательный, если операнды имеют разные знаки. | 18. Представление чисел с плавающей точкой. Сложение чисел, представленных в формате с плавающей точкой  ***Представление чисел с плавающей точкой***    ***Сложение чисел, представленных в формате с плавающей точкой*** | 19. Умножение чисел, представленных в формате с плавающей точкой. Деление чисел, представленных в формате с плавающей точкой.  Последовательность действий, обеспечивающих получение произведение двух чисел, заключается в следующем:   * определяется знак произведения как сумма по модулю двух знаковых разрядов мантисс сомножителей; * определяется предварительное значение порядка произведения посредством суммирования порядков сомножителей; * определяется предварительное значение мантиссы произведения как произведения мантисс операндов; * устраняется нарушение нормализации мантиссы произведения (если нарушение имеет место) соответствующей корректировкой предварительного значения порядка и мантиссы искомого произведения. * При формировании мантиссы произведения нормализованных чисел с плавающей точкой возможен только один вид нарушения нормализации – нарушение нормализации справа от точки с появлением нуля только в старшем разряде мантиссы.   Последовательность действий, обеспечивающих получение отношение двух чисел, заключается в следующем:   * определяется знак по произведению знаков мантисс * производится сложение двух мантисс * совершение сдвига влево с записью сдвинутого числа в ответ в формате знак итогового числа. Сдвинутое число 1 … сдвинутое число 6 * если результатом сдвига является 1, добавляем число А, если 0, модуль числа В | 20. Неосновные арифметические операции. Вычисление квадратного корня  К числу основных арифметических операций, непосредственно реализуемых в ЭВМ, относятся операции сложения, умножения, деления. Остальные операции (например, такие, как возведение в степень, извлечение квадратного корня) реализуются программным способом.  Имеются 2 пути вычисления квадратного корня:   1. Связан с разработкой микропрограммы извлечения квадратного корня с использованием набора простых арифметических операций. При этом микропрограмма реализует один из известных итерационных методов извлечения квадратного корня с помощью базовой аппаратуры. Например, формулой Ньютона: Bi+1=0.5(Bi+A/Bi), где Bi+1 есть (i+1)-e приближение B=sqrt(A), а i=0,1,2… 2. Наиболее простой алгоритм сводится к подбору цифр в результате разряд за разрядом, начиная со старшего, т.е с 2-1. При этом вычисление i=q цифры D происходит следующим образом. После получения (i-1)-й цифры bi-1 в i-й разряд В для пробы помещается 1. Вычисляется разность (A-Bi2)=Rt. Если Rt>0, то Bi есть число, у которого цифры всех i разрядов совпадают с цифрами искомого результата B. Если Ri<0, то в i-м разряде bi нужно поставить 0 и переходить к вычислению (I +1)-го разряда.Т.к вычисление этого разряда снова начинается с подстановки пробной 1, то в случае (A-Bi2)=R<0 можно вместо «стирания» 1 в i-м разряде вычесть 1 из (i+1)-го разряда. |

|  |  |  |  |
| --- | --- | --- | --- |
| 9. Деление двоичных чисел без восстановления остатка.  ***Алгоритм деления без восстановления остатка***  1. Выполняется пробное вычитание с формированием первого остатка A1=[Дм]доп+[-Дт]доп. Далее, если А1 < 0, то в первый разряд, расположенный слева от запятой, заносится ноль (0), иначе единица (1) – что является признаком переполнение и осуществляется переход к пункту 5.  2. Формирование очередного остатка. Если Аi < 0, то Ai+1=Ai∙2+[Дт]доп, иначе Ai+1=Ai∙2+[-Дт]доп.  3. Если Аi+1 < 0, то в очередной разряд частного справа от запятой записывается ноль (Чт(n)=0), иначе записывается единица (Чт(n)=1).  4. Если достигнута заданная точность частого или получен нулевой остаток Ai+1, то процесс деления окончен и осуществляется переход к пункту 5, иначе переходим к пункту 2 алгоритма.  5. Окончание алгоритма.  https://www.ok-t.ru/studopediaru/baza7/1445215431351.files/image388.gif | 10. Двоично-десятичная арифметика. Сложение и вычитание двоично-десятичных чисел.  ***Сложение двоично-десятичных чисел:***  Рассмотрим на конкретном примере реализацию этой операции. Пример: найти сумму двух десятичных чисел с использованием двоично-десятичной системы счисления: A = D + C, где D = 3927; C = 4856. Решение: составляем двоично-десятичную запись для чисел D и C: D =3927 = 0011 1001 0010 0111: C = 4856 =0100 1000 0101 0110. Найти значение А можно, реализовав следующую последовательность операций из двоичного сложения и операции коррекции:  https://studfile.net/html/20260/740/html_gatsd6GBaT.XiuA/img-EP5ScZ.png  Для получения двоично-десятичной суммы A на основании результата сложения операндов по правилам двоичной арифметики необходимо добавить шестерку (0110) в те тетрады, из которых был перенос, и в которых получилось значение >9.  Вычитание двоично-десятичных чисел:  https://studfile.net/html/20260/740/html_gatsd6GBaT.XiuA/img-O1gBhW.png | 11. Кодирование алгебраических чисел. Дополнительный и обратный коды двоичных чисел. ***Кодирование алгебраических чисел:***  Для представления чисел со знаком используются специальные коды: прямой код; дополнительный код; обратный код.  Во всех 3 случаях используется следующий формат представления числа, содержащий два поля - поле знака и поле модуля. *Поле знака* представлено одним разрядом, в котором устанавливается 0, если число положительное, и 1, если число отрицательно. *Поле модуля* отражает количественную оценку числа и для каждого кода формируется по*–*разному. Количество разрядов поля модуля определяется диапазоном изменения отображаемых чисел или точностью их представления.  В *прямом и дополнительном коде* запись целого числа*А* формируется по следующим правилам:  https://studfile.net/html/20260/740/html_gatsd6GBaT.XiuA/img-C2EBxE.pnghttps://studfile.net/html/20260/740/html_gatsd6GBaT.XiuA/img-O6LW5C.png  где *n* – разрядность модульного поля; *q* – основание системы счисления; *qn* – максимальная не включенная граница диапазона изменения представляемых чисел, т. к. диапазон изменение чисел *А* определяется как *qn*> |*А*| ≥ 0 .  В *обратном коде*:  https://studfile.net/html/20260/740/html_gatsd6GBaT.XiuA/img-FnMl_M.png  где *n* – разрядность модульного поля; *q* – основание системы счисления; (2*n-*1) – максимальная включенная граница диапазона изменения представляемых чисел, т. е. диапазон изменения чисел *А* определяется как (*qn–*1) ≥ | *A* | ≥ 0.  Легко показать, что перевод отрицательного числа из обратного или дополнительного кода в прямой выполняется по тем же правилам, что и перевод числа из прямого кода в обратный или *дополнительный*:  – для перевода отрицательного числа из обратного в *прямой код* необходимо дополнить его модуль до включенной границы;  – для перевода отрицательного числа из обратного в *прямой код* необходимо дополнить его модуль до невключенной границы. | 12. Операции с двоичными числами в дополнительном и обратном кодах  ***Операции с двоичными числами в дополнительном коде***  При использовании дополнительного или обратного кода операция вычитания заменяется операцией сложения с изменением знака второго операнда. Переполнение знакового поля игнорируется!  ***Операции с двоичными числами в обратном коде***  При сложении чисел, представленных в *обратном* коде, выполняется сложение разрядов, представляющих запись операндов, по правилам двоичной арифметики по всей длине записи чисел, не обращая внимания на границу, разделяющую знаковое и модульные поля. Переполнение знакового поля, т.е. перенос, возникший из крайнего левого разряда, должен быть учтен как +1 в младший разряд полученной суммы. В результате такого сложения будет получен *обратный* код суммы заданных операндов.  **Пример**  *С*1 = *А*+ *В*, *С*2 = *А*- *В*, *С*3 = *В*- *А*, *С*4 = - *А*- *В*, если А=5710, В = - 21010. При выполнении операций использовать двоичный обратный код. Результат представить в прямом коде.  **Решение**  Обратные коды операндов имеют вид  [*А*]ок = 0.000111001, [-*А*]ок = 1.111000110,  [*В*]ок = 1.100101101, [-*В*]ок = 0.011010010.  https://studfile.net/html/20260/740/html_gatsd6GBaT.XiuA/img-UuMQhW.png |
| 13. Модифицированные коды  Рассмотрим пример:  https://studfile.net/html/20260/740/html_gatsd6GBaT.XiuA/img-73mtKQ.jpg  Пример, выполненный по всем формальным правилам, дал абсурдный результат, так как получена положительная сумма двух отрицательных операндов. Аналогичная ситуация может возникать и при использовании дополнительного кода.  Ситуацию переполнения можно обнаруживать по факту появления «абсурдного» результата, но для этого необходимо помнить то, что в суммировании принимают участие операнды с одинаковыми знаками и знак полученного при этом результата отличен от знака операндов.  Более просто ситуация переполнения определяется при применении модифицированного кода (обратного или дополнительного). Модифицированные коды отличаются от базовых кодов только тем, что поле знака операндов имеет два разряда, и эти разряды имеют одинаковые значения:  00 – для положительных чисел;  11 – для отрицательных чисел. | 14. Логические операции с двоичными кодами: логическое суммирование, логическое умножение, логическое отрицание, суммирование по модулю два, логические сдвиги.  **Логическое суммирование**  Операция *логического суммирования* выполняется над двумя кодами и генерирует код той же разрядности, что и операнды, у которого в некотором *i*-м разряде находится единица, если хотя бы в одном операнде в *i*-м разряде имеет место единица: 10001101 ∨ 11110000 = 11111101.  **Логическое умножение**  Операция *логического умножения* выполняется над двумя кодами и генерирует код той же разрядности, что и операнды, у которого в некотором *i*-м разряде находится единица, если оба операнда в этом *i*-м разряде имеются единицу, и ноль во всех других случаях: 10001101 ∧ 11110000 = 10000000  **Суммирование по модулю 2**  Операция *суммирования по модулю 2* выполняется над двумя кодами и генерирует код той же разрядности, что и операнды, у которого в некотором *i*-м разряде находится единица, если два заданных операнда в *i*-м разряде имеют противоположные значения. Иногда эта операция называется «исключающее ИЛИ»: 10001101 ⊕ 11110000 = 01111101.  **Логическое отрицание**  Операция логического отрицания выполняется над одним кодом и генерирует результирующий код той же разрядности, что и операнд, в некотором *i*-м разряде которого находится значение, противоположное значению в *i*-м разряде отрицаемого кода.  **Операция сдвига**  Операции *сдвига* в свою очередь, подразделяются на:  1) логические сдвиги, которые имеют разновидности – сдвиг вправо, сдвиг влево, циклический сдвиг вправо, циклический сдвиг влево;  2) арифметические сдвиги вправо и влево, выполнение которых зависит от знака и кода сдвигаемого числа. | 15. Арифметические сдвиги положительных двоичных чисел, представленных в прямом коде. Арифметические сдвиги двоичных чисел, представленных в обратном коде.  Для представления чисел со знаком используются специальные коды: прямой код, дополнительный код, обратный код. Во всех трех случаях используется определенный формат представления числа, содержащий два поля – поле знака и поле модуля.  Сдвиг, при котором уходящий бит исчезает, не влияя на оставшиеся биты, а на месте появившегося бита записывается бит 0. В большинстве процессоров уходящий бит сохраняется во флаге переноса.  Арифметические сдвиги обеспечивают выполнение умножения или операции деления двоичных кодов на два. Если сдвигается положительное число, то сдвиг (вправо или влево) выполняется как соответствующий логический сдвиг (влево или вправо), с той лишь разницей, что предусматриваются средства определения факта переполнения при сдвиге влево, что реализуется и при всех других арифметических операциях. При любом сдвиге вправо предусматриваются средства для округления после завершения нужного количества сдвигов и средства обнаружения обнуления сдвигаемой величины после очередного сдвига. Арифметические сдвиги влево положительных двоичных чисел выполняются независимо от используемого кода (прямого, обратного, дополнительного).  При арифметическом сдвиге влево отрицательного двоичного числа, представленного в обратном коде, осуществляется циклический сдвиг исходного кода с контролем за переполнением, например, сдвиг влево отрицательного двоичного числа 11.1100110 (2510), представленного в обратном коде, дает в результате 11.1001101 (5010). При арифметическом сдвиге вправо отрицательного двоичного числа, представленного в обратном коде, осуществляется сдвиг только модульной части записи числа с установкой единицы в освобождающийся разряд. При этом может осуществляется контроль за обнулением результата сдвига (появление единичных значений во всех разрядах) и округление результата после выполнения заданного количества сдвигов. | 16. Арифметические сдвиги двоичных чисел, представленных в дополнительном коде. Сдвиг отрицательных чисел с переполнением.  При арифметическом сдвиге влево отрицательного двоичного числа, представленного в дополнительном коде, осуществляется логический сдвиг влево модуля исходного кода (освобождающийся разряд заполняется нулем) с контролем за переполнением, например, сдвиг влево отрицательного двоичного числа 11.11001110 (50 в 10 с/c), представленного в дополнительном коде, дает в результате 11.10011100 (100 в 10 с/с).  При арифметическом сдвиге вправо отрицательного двоичного числа, представленного в дополнительном коде, осуществляется логический сдвиг вправо модуля записи числа с установкой единицы в освобождающийся разряд. При этом может осуществляется контроль за обнулением результата сдвига (появление единичных значений во всех разрядах).  При выполнении сдвига вправо нечетного целого числа результат получается с точностью до младшего разряда кода, причем ошибка положительная.  Арифметический сдвиг вправо может выполняться над отрицательными числами с переполнением (такие числа в модифицированном прямом, обратном или дополнительном коде имеют в знаковом поле 10). В этом случае после сдвига в знаковом поле будет 11, а в старшем разряде – 0, если число представлено в обратном или дополнительном коде, или 1, если число представлено в прямом коде. |
| 21. Методы вычисления элементарных функций.  Методы вычисления элементарных функций:   * Разложение в ряд Тейлора (степенные полиномы). Полиномы вычисляются по схеме Горнера. При этом требуется выполнить m операций умножения и m операций сложений (m- степень полинома). Используется редко, т.к. ЭВМ долго считает * Аппроксимация с помощью различных полиномов. Используется наиболее часто. Характеризуется достаточно высоким единообразием вычисления всех ЭФ, однако при этом в памяти необходимо хранить большое количество коэффициентов всех полиномов. * Табличные методы. Основаны главным образом на кусочно-линейной и криволинейной аппроксимации. Для вычисления ЭФ этим методом требуется выполнить малое число арифметических операций, однако объем таблиц и время поиска в них может v быть большим. * Рациональные приближения ЭФ. При методе рационального приближения ЭФ функцию представляют в виде отношения двух полиномов, причем число членов в каждом полиноме намного меньше, чем при соответствующем разложении в ряд Тейлора. Однако коэффициенты полиномов должны обязательно храниться в памяти. * Использование цепных дробей. Метод цепных дробей характеризуется однообразием вычисления всех ЭФ, при этом количество констант мало. По сравнению с методом степенных полиномов в этом методе число шагов меньше, однако на каждом шаге выполняется операция деления, занимающая много времени. * Итерационные (рекуррентные). Итерационные методы предполагают вычисление последовательных приближений функции по итерационной формуле Yj+i = f(Yi). | 22. Денормализованные числа. Подводные камни в арифметике с плавающей запятой.  Денормализованное число - вид чисел с плавающей точкой, определенный в стандарте IEEE 754. Для получения их значения не требуется использование неявной единицы; мантисса просто умножается на наименьшую для данного формата экспоненту.  **Подводные камни в арифметике с плавающей точкой**   * **Округление:** Особенности округления в IEEE754. Математически показано, что если 0,5 округлять до 1, существует набор операций, при которых ошибка округления будет возрастать. Поэтому в IEEE754 применяется правило округления до четного. Самая опасная операция с точки зрения округления в арифметике с плавающей запятой — это вычитание. При вычитании близких чисел значимые разряды могут потеряться. * **Неассоциативность арифметических операций:** В арифметике с плавающей запятой правило (a\*b)\*c = a\*(b\*c) не выполняется для любых арифметических операций. Например, (1020+1)-1020=0 ≠ (1020-1020)+1=1 * **Числовые константы:** не все десятичные числа имеют двоичное представление с плавающей запятой. Например, число «0,2» будет представлено как «0,200000003» в одинарной точности. Соответственно, «0,2 + 0,2 ≈ 0,4». * **Выбор минимума из 2 значений:** Допустим из двух значений нам нужно выбрать минимальное. В Си это можно сделать одним из следующих способов:1. x < y? x: y 2. x <= y? x: y 3. x > y? y: x 4. x >= y? y: x Часто компилятор считает их эквивалентными и всегда использует первый вариант, так как он выполняется за одну инструкцию процессора. * **Сравнение чисел:** Лучший, но все равно ошибочный способ, это сравнивать разницу с допустимой абсолютной погрешностью. | 23. Погрешности, обусловленные форматом с плавающей точкой  В случае с плавающей точкой число Х представляется в виде двух частей: мантисса и порядок. Количественная оценка числа Х определяется как Х = qxп × хм, где q – основание системы счисления. Для плавающей точки максимальные значения абсолютной и относительной ошибок определяются следующим образом:  **Максимальная абсолютная погрешность представления чисел:**  Dmax = 2-(s+1) × 2p; **Абсолютной погрешностью** называется разность между этим числом и его точным значением (из большего числа вычитается меньшее)\*. **Δ = x2 - x1.**  **Максимальная относительная погрешность:**  dmax = D max / Аmin = 2-(s+1) × 2p / (хм min × 2p)= 2­(s+1) × 2p / ( 2-1 × 2p) = 2­(s+1) / (2-1) = =2-s. **Относительной погрешностью** приближенного числа называется отношение абсолютной погрешности приближенного числа к самому этому числу. **δ = Δ/a.**  Относительная ошибка при представлении чисел в форме с плавающей точкой существенно меньше, чем в случае с фиксированной точкой. Это, а также больший диапазон изменения представляемых чисел, является основным преимуществом представления чисел с плавающей точкой. | 24. Основные понятия алгебры логики. Способы задания логической функции.  Алгебра логики используется при анализе и синтезе схем ЭВМ по двум причинам. Во-первых, это объясняется соответствием представления переменных и функций алгебры логики. Во-вторых, двоичным представлением информации и характером работы отдельных компонентов вычислительной техники. Основные понятия алгебры логики:   * **Логическая переменная** — это такая переменная, которая может принимать одно из двух значений * **Логическая константа** — это такая постоянная величина, значением которой может быть истинно или ложно * **Логическая функция** — это такая функция, которая может принимать одно из двух значений: истинно или ложно в зависимости от текущих значений ее аргументов, в кач-ве которых используются логические переменные.   Зависимость логической функции от переменных может задаваться по–разному:   * словесным описанием (используется в случае сравнительно несложной логической функции) * таблицей истинности (универсальное средство задания логической функции. Она включает все наборы для заданного количества переменных, определяющих значение логической функции, с указанием значений, которые принимает функция для каждого набора, Макс. количество полностью определенных функций от «n» переменных определяется как M = (2^2)^n) * логическим выражением (комбинация логических переменных и констант, связанных элементарными базовыми логическими функциями (или логическими операциями), которые могут разделяться скобками.) |

|  |  |  |  |
| --- | --- | --- | --- |
| 25. Понятие о принципе двойственности. Суперпозиция логических функций.  Определение: Двойственной функцией для f(x1, x1,…,xn) называется функция f\* = ¬f(¬x1, ¬x2,…,¬xn) ¬ - отрицание  Теорема: Функция, двойственная к суперпозиции функций, есть суперпозиции функций, двойственных к функциям, составляющим эту суперпозицию.  Принцип двойственности  Если функция f задана формулой, построенной с помощью &,∨,¬,0,1 и переменных, то по теореме о суперпозиции двойственных функций и ввиду того, что для функций x&y, ¬x, 1,0 двойственными являются x∨y ,x ,0,1 соответственно, то f\* получается из f заменой & на ∨, 0 на 1 и т.д. при сохранении исходной расстановки скобок.  Две функции алгебры логики называются двойственными, если одна получается из другой заменой каждой операции конъюнкции на операцию дизъюнкции и наоборот. Принцип двойственности формулируется так: если функции F1 и F2 равносильны, то равносильны и двойственные им функции F1\* и F2\*  Функция, полученная из некоторых функций путем применения принципа суперпозиции, называется суперпозицией этих функций. Все логические функции двух переменных, а также конъюнкции и дизъюнкции n переменных называются элементарными логическими функциями. Они позволяют строить любые новые функции алгебры логики, являющиеся суперпозициями элементарных функций  Суперпозиция функций (или сложная функция) — это функция, полученная из некоторого множества функций путем подстановки одной функции в другую или отождествления (равенство) переменных. | 26. Нормальная и совершенные нормальные логических функций.  Конъюнкция (дизъюнкция), в которой каждая переменная с инверсией или без встречается не более одного раза, называется *элементарной*. Число входящих в нее переменных определяет *ранг* конъюнкции (дизъюнкции). Дизъюнкция любого числа элементарных конъюнкций называет-ся *дизъюнктивной нормальной формой*(ДНФ), например:  https://helpiks.org/helpiksorg/baza1/703749610099.files/image926.gif  Конъюнкция любого числа элементарных дизъюнкций называется *конъюнктивной нормальной формой* (КНФ). Например:  https://helpiks.org/helpiksorg/baza1/703749610099.files/image928.gif  Для каждой функции может существовать несколько равносильных ДНФ (или КНФ), например:  https://helpiks.org/helpiksorg/baza1/703749610099.files/image930.gif  ДНФ (КНФ), содержащая наименьшее число переменных по сравнению с другими равносильными ДНФ (КНФ), называется *минимальной*. ДНФ логической функции, состоящая из элементарных конъюнкций одинакового ранга, называется *совершенной дизъюнктивной нормальной формой* (СДНФ). В СДНФ каждая элементарная конъюнкция включает все переменные с инверсиями или без, причем одинаковых конъюнкций нет, например:  https://helpiks.org/helpiksorg/baza1/703749610099.files/image932.gif  От любой ДНФ можно перейти к СДНФ. Для этого необходимо ввести недостающие переменные в конъюнкции младших рангов умножением их на равносильность вида формула (х - недостающая переменная), раскрыть скобки и избавиться от повторяющихся конъюнкций в соответствии с правилом х+х=х.  СДНФ обладает следующими свойствами:  - если при каком-то наборе f = 1, то СДНФ только одна из элементарных конъюнкций принимает значение единицы;  - если для данного набора f = 0, то в СДНФ ни один из членов не будет равен единице.  Свойства СКНФ:  - если для данного набора f=0, то в СКНФ только одна из элементарных конъюнкций принимает нулевое значение;  - если для данного набора f=1, то в СКНФ ни один из членов не принимает нулевое значения. | 27. Минимизация булевых функций. Основные понятия.  Задача мин-ии булевых функций: найти аналитическое выражение заданной бул. функции в форме, содержащей минимально возможное число букв. **Элементарная конъюнкция -** конъюнкция конечного числа различных между собой булевых переменных, каждая из которых может иметь или не иметь отрицания. **ДНФ -** дизъюнкция элементарных конъюнкций. **Минимальная ДНФ** булевой функции - ДНФ, содержащая минимальное число букв (по отношению ко всем другим ДНФ, представляющим заданную булеву функцию). **Булева функция** g(x1,x2…xn)называется импликантной булевой функции f(x1,x2…xn), если для любого набора переменных, на некотором g=1, справедливо f=1. **Импликанта** g булевой функции f, являющаяся элементарной конъюнкцией, называется простой, если никакая часть импликанты g не является импликантой функции f. **Сокращенная ДНФ** булевой функции называется тупиковой, если в ней отсутствуют лишние простые импликанты. **Наиболее известные методы минимизации:** методом Квайна; С использованием диаграмм Вейча; не полностью определенных функций; Конъюнктивных нормальных форм; Метод кубического задания функций алгебры логики; Методом Метод Куайна—Мак-Класки; Методом Нельсона; С использованием алгоритма извлечения; ФАЛ методом преобразования логических функций.  **Минимизация системы логических функций: 1)** Построить полное множество А элементарных конъюнкций минимизируемой системы ф-ий, считая что вначале каждая из ф-ий системы представлена в СДНФ. Каждой единице констнтуента мн-ва присвоить знак, содержащий номера ф-ий системы, в которые входит рассматриваемая констнтуента. **2)** Произвести минимизацию СДНФ функции. **3)** Построить импланкантную матрицу функции, аналогичную матрице Квайна с той разницей, что для каждой констнтуенты единицы выделяются столько столбцов, сколько разл. номеров функций сод-ит ее знак.  **Мин-ия частично опред. Ф-ий: 1)** Нахождение любым способом сокращенную ДНФ/КНФ функции, получающейся до определения единицами исходной функции f на всех неопределенных наборах. **2)** Выбрать минимальную ДНФ/КНФ по импликантной матрице, где в столбцах записаны лишь те конституанты единицы функции f, которые соответствуют полностью опред. единичным наборам. | 28. Минимизация логических выражений методом Квайна.  Метод Квайна выполняется в два этапа.  Первый этап имеет своей целью получение тупиковой формы, представляющей собой дизъюнкцию, в качестве слагаемых которой используются конъюнкции (каждая из них не склеивается ни с одной другой конъюнкцией, входящей в это выражение). Такие конъюнкции называются простыми импликантами.  Данный этап выполняется за счет реализации отдельных шагов. На каждом шаге на основании выражения, полученного на предыдущем шаге, выполняются все возможные операции склеивания для пар имеющихся конъюнкций. Каждый шаг понижает ранг исходных конъюнкций на единицу. Шаги повторяются до получения тупиковой формы.  Второй этап имеет своей целью устранение из тупиковой формы всех избыточных простых импликант, что дает в результате минимальное логическое выражение. |
| 29. Минимизация логических выражений с использованием Карт Карно (диаграммами Вейча).  Карта Карно для «*n*» логических переменных представляет собой множество квадратов (клеток), объединенных в близкую к квадрату прямоугольную форму. Каждая такая клетка соответствует одному набору логических переменных, причем наборы двух соседних клеток должны отличаться на значение одной переменной (их наборы образуют склеивающиеся конъюнкции).  Для выполнения минимизации представленной в карте Карно функции необходимо выполнить два этапа:   * охватить множество клеток карты Карно контурами; * записать минимальное выражение для заданной функции в виде дизъюнкции конъюнкций, где каждая конъюнкция соответствует одному из введенных на карте контуров.   Охват клеток карты контурами выполняется с соблюдением следующих правил:   * контур должен иметь прямоугольную форму; * в контур может входить количество клеток, равное целой степени числа «2»; * в контур могут входить клетки, являющиеся логическими соседями; * в контур необходимо включить максимальное количество клеток с учетом вышеприведенных требований; * контурами необходимо охватить все клетки с единичными значениями; * контуров должно быть минимальное количество; * количество клеток в контуре должно быть равно 2ΔR, где ΔR –разность ранга (дельта ранга) конституент единицы заданной функции и ранга конъюнкции, соответствующей контуру.   *Логическими соседями* являются такие две клетки, наборы которых отличаются только одной переменной – в одном эта переменная должна иметь прямое, в другом – обратное значение. | 30. Синтез логических схем по логическим выражениям в булевом базисе. Логический базис И-НЕ. Логический базис ИЛИ-НЕ.  Логические схемы строятся на основе логических элементов, набор которых определяется заданным логическим базисом. Для базиса Буля в качестве логических элементов используются элементы, реализующие базовые логические функции И, ИЛИ, НЕ.    Булевый базис не является единственной функционально полной системой логических функций. Среди других наибольшее распространение получили базис И–НЕ и базис ИЛИ–НЕ.    При синтезе логических схем в заданном базисе логических элементов (например, в базисах И–НЕ, или ИЛИ–НЕ) целесообразно предварительно исходное выражение привести к форме, в которой в выражении будут использованы только логические операции, соответствующие используемым логическим элементам в заданном базисе. | 31. Законы и правила алгебры Буля.  При работе с булевыми логическим выражениями используются следующие законы, правила и операции.   * *Переместительный* (коммутативный) *закон. (*Закон справедлив как для конъюнкции, так и для дизъюнкции, справедлив для любого количества операндов)   от перемены мест логических слагаемых сумма не меняется х1 + х2 + х3 + х4 = х4 + х3 + х2 + х1  от перемены мест логических сомножителей их произведение не меняется   * *Сочетательный* (ассоциативный) *закон. (*Справедлив как для конъюнкции, так и для дизъюнкции.)   при логическом сложении отдельные слагаемые можно заменить их суммой х1 + х2 + х3 + х4 = (х2 + х3) + х1 + х4 = (х1 + х4 ) + (х2 + х3)  при логическом умножении отдельные логические сомножители можно заменить их произведением   * *Распределительный* (дистрибутивный) *закон*. (х1 + х2) х3 = х1х3 + х2х3; (х1 + х2) (х1 + х3) = х1 + х2х3; * *Правило де Моргана:* отрицание суммы равно произведению отрицаний, отрицание произведения равно сумме отрицаний; * *Операция склеивания:* операция склеивания для конъюнкций, где А – переменная или любое логическое выражение, и операция склеивания для дизъюнкций; * *Операции с отрицаниями:* двойное отрицание равносильно отсутствию отрицания; * *Операции с константами;* * *Операции с одинаковыми операндами.* | 32. Параллелизм. Виды, организация  Параллелизм – основа высокопроизводительной работы всех подсистем вычислительных машин. Организация памяти любого уровня иерархии, организация системного ввода/вывода, организация мультиплексирования шин и т.д. базируются на принципах параллельной обработки запросов. Современные операционные системы являются многозадачными и многопользовательскими, имитируя параллельное исполнение программ посредством механизма прерываний.  Виды параллелизма:   * параллелизм операций, * векторный параллелизм, * параллелизм потоков управления, * параллелизм задач.   Уровни параллелизма  1) Микроуровневый (команды разд. на фазы)  2) Уровень команд (параллельное выполнение нескольких команд)  3) Уровень потоков (задачи разб. на части, которые выполн. парал.)  4) Уровень заданий (независ. задания одноврем. на неск. процессор.) |
| 33. Устройства ЭВМ. Состав АЛУ.  **Устройства ЭВМ**  Классическая ЭВМ состоит из трех основных устройств: арифметико-логического устройства, устройства управления и запоминающего устройства.  **Состав АЛУ**  АЛУ состоит из следующих типовых узлов:  – Регистры (R), служащие для хранения операндов и результатов;  – Сумматор (SM), служащий для выполнения операции суммирования многоразрядных кодов;  – Операционные узлы (ОУ), служащие для выполнения логических операций;  – Мультиплексор (MS);  – Счетчик (Сч), обеспечивающий подсчет тактов длинных операций;  – Регистр флажков (RF), служащий для фиксации особой информации, характеризующей полученный результат. | 34. Типы памяти.  Памятью ЭВМ называется совокупность устройств, служащих для запоминания, хранения и выдачи информации. Отдельные устройства, входящие в эту совокупность, называются запоминающими устройствами (ЗУ) того или иного типа. К основным параметрам, характеризующим запоминающие устройства, относятся емкость (это максимальное количество данных, которое в ней может храниться.) и быстродействие(Быстродействие памяти определяется продолжительностью операции обращения, то есть временем, затрачиваемым на поиск нужной информации в памяти и на ее считывание, или временем на поиск места в памяти, предназначенного для хранения данной информации)  **Регистровая память** – Регистры CPU программно доступны и хранят информацию, наиболее часто используемую при выполнении программы  **Оперативная память** – устройство, которое служит для хранения информации (программ, исходных данных, промежуточных и конечных результатов обработки), непосредственно используемой в ходе выполнения программы в процессоре  **Кэш-память** организована как более быстродействующая статическая оперативная память со специальным механизмом записи и считывания информации и предназначена для хранения информации, наиболее часто используемой при работе программы.  **Внешняя память** организуется, как правило, на магнитных и оптических дисках, магнитных лентах. | 35. Код Грея.  Код Грея — двоичный код, в котором две «соседние» (в упорядоченном, то есть лексикографическом, наборе) кодовые комбинации различаются только цифрой в одном двоичном разряде.  Двоичным кодом Грея порядка n называется последовательность всех 2n n-битных кодов, в которой любые два соседних кода различаются ровно в 1 разряде. Пример кодов Грея порядка 2: 00, 01, 11, 10.  Наиболее часто на практике применяется рефлексивный двоичный код Грея, хотя в общем случае существует бесконечное множество кодов Грея со значениями цифр в разрядах, взятых из различных алфавитов. В большинстве случаев, под термином «код Грея» понимают именно рефлексивный бинарный код Грея.  Преобразование двоичного кода в код Грея  Коды Грея легко получаются из двоичных чисел путём побитовой операции «Исключающее ИЛИ» с тем же числом, сдвинутым вправо на один бит и в котором старший разряд заполняется нулём. Следовательно, i-й бит кода Грея Gi выражается через биты двоичного кода Bi след. образом:    Преобразование кода Грея в двоичный код  Обратный алгоритм — преобразование кода Грея в двоичный код — можно выразить рекуррентной формулой  Действительно, если подставить в эту формулу вышеприведённое выражение для i-го бита кода Грея, получим | 36. Обратная польская запись.  Обратная польская запись — форма записи математических и логических выражений, в которой операнды расположены перед знаками операций.  Общий порядок:  Автоматизация вычисления выражений в обратной польской нотации основана на использовании стека. Алгоритм вычисления для стековой машины элементарен:  1. Обработка входного символа   * Если на вход подан операнд, он помещается на вершину стека. * Если на вход подан знак операции, то соответствующая операция выполняется над требуемым количеством значений, извлечённых из стека, взятых в порядке добавления. Результат выполненной операции кладётся на вершину стека.   2. Если входной набор символов обработан не полностью, перейти к шагу 1.  3. После полной обработки входного набора символов результат вычисления выражения лежит на вершине стека. |

|  |  |  |  |
| --- | --- | --- | --- |
| 37. АЦП и ЦАП.  ЦАП – цифро-аналоговые преобразователи – устройства, предназначенные для преобразования дискретного (цифрового) сигнала в непрерывный (аналоговый) сигнал. Преобразование производится пропорционально двоичному коду сигнала.  АЦП – аналого-цифровые преобразователи – устройства, предназначенные для преобразования непрерывных (аналоговых) сигналов в цифровые.  ЦАП (цифро-аналоговый преобразователь) и АЦП (аналого-цифровой преобразователь) входят в состав кодека и непосредственно выполняют дискретизацию:  — при воспроизведении ЦАП переводит цифровой код в аналоговый сигнал;  — при записи АЦП выполняет обратное преобразование.  Чем качественнее АЦП, тем более чистый и детальный звук будет литься из колонок.  Чем качественнее ЦАП, тем точнее будет переведен в цифру аналоговый сигнал.  АЦП и ЦАП являются основными устройствами ввода-вывода ин­формации в цифровых системах, предназначенных для обработки анало­говой информации или управления каким-либо технологическим процес­сом. | 38. Корректирующие коды. Код Хэмминга. Область применения.  **Корректирующие коды**  Корректирующий код (также помехоустойчивый код) — [код](https://ru.wikipedia.org/wiki/%D0%9A%D0%BE%D0%B4), предназначенный для [обнаружения и исправления ошибок](https://ru.wikipedia.org/wiki/%D0%9A%D0%BE%D0%BD%D1%82%D1%80%D0%BE%D0%BB%D1%8C_%D0%BE%D1%88%D0%B8%D0%B1%D0%BE%D0%BA).  Фактически любой код, исправляющий ошибки, может быть также использован для обнаружения ошибок (при этом он будет способен обнаружить большее число ошибок, чем был способен исправить). Коды, исправляющие ошибки, применяются в системах [цифровой связи](https://ru.wikipedia.org/wiki/%D0%A6%D0%B8%D1%84%D1%80%D0%BE%D0%B2%D0%B0%D1%8F_%D1%81%D0%B2%D1%8F%D0%B7%D1%8C), в том числе: спутниковой, радиорелейной, сотовой, передаче данных по телефонным каналам, а также в системах хранения информации, в том числе магнитных и оптических. Коды, обнаруживающие ошибки, применяются в [сетевых протоколах](https://ru.wikipedia.org/wiki/%D0%A1%D0%B5%D1%82%D0%B5%D0%B2%D0%BE%D0%B9_%D0%BF%D1%80%D0%BE%D1%82%D0%BE%D0%BA%D0%BE%D0%BB) различных [уровней](https://ru.wikipedia.org/wiki/%D0%9C%D0%BE%D0%B4%D0%B5%D0%BB%D1%8C_OSI).  **Код Хэмминга**  Код Хэмминга — вероятно, наиболее известный из первых самоконтролирующихся и самокорректирующихся кодов. Построены они применительно к двоичной системе счисления.  **Применение**  Код Хэмминга используется в некоторых прикладных программах в области хранения данных, особенно в RAID 2; кроме того, метод Хэмминга давно применяется в памяти типа ECC и позволяет «на лету» исправлять однократные и обнаруживать двукратные ошибки. | 39. Языки описания аппаратуры. ПЛИС (FPGA) модули.  **Языки описания аппаратуры**  Язык описания аппаратуры (HDL от [англ.](https://ru.wikipedia.org/wiki/%D0%90%D0%BD%D0%B3%D0%BB%D0%B8%D0%B9%D1%81%D0%BA%D0%B8%D0%B9_%D1%8F%D0%B7%D1%8B%D0%BA) hardware description language) — специализированный компьютерный язык, используемый для описания структуры и поведения [электронных схем](https://ru.wikipedia.org/wiki/%D0%AD%D0%BB%D0%B5%D0%BA%D1%82%D1%80%D0%BE%D0%BD%D0%BD%D0%B0%D1%8F_%D1%81%D1%85%D0%B5%D0%BC%D0%B0), чаще всего цифровых лог. схем.  Языки описания аппаратуры внешне похожи на такие [языки программирования](https://ru.wikipedia.org/wiki/%D0%AF%D0%B7%D1%8B%D0%BA_%D0%BF%D1%80%D0%BE%D0%B3%D1%80%D0%B0%D0%BC%D0%BC%D0%B8%D1%80%D0%BE%D0%B2%D0%B0%D0%BD%D0%B8%D1%8F), как [Си](https://ru.wikipedia.org/wiki/%D0%A1%D0%B8_(%D1%8F%D0%B7%D1%8B%D0%BA_%D0%BF%D1%80%D0%BE%D0%B3%D1%80%D0%B0%D0%BC%D0%BC%D0%B8%D1%80%D0%BE%D0%B2%D0%B0%D0%BD%D0%B8%D1%8F)) или [Паскаль](https://ru.wikipedia.org/wiki/%D0%9F%D0%B0%D1%81%D0%BA%D0%B0%D0%BB%D1%8C_(%D1%8F%D0%B7%D1%8B%D0%BA_%D0%BF%D1%80%D0%BE%D0%B3%D1%80%D0%B0%D0%BC%D0%BC%D0%B8%D1%80%D0%BE%D0%B2%D0%B0%D0%BD%D0%B8%D1%8F)), написанные на них программы также состоят из выражений, операторов, управляющих структур.  **ПЛИС (FPGA) модули**  ПЛИС — электронный компонент, используемый для создания цифровых интегральных схем. В отличие от обычных цифровых микросхем, логика работы ПЛИС не определяется при изготовлении, а задаётся посредством программирования.  Структура ПЛИС основана на ФАЛ (булевой алгебры) и содержит следующие логические элементы: — логическое «И» (коньюнкторы); — логическое «ИЛИ» (дизъюнкторы); — логическое «НЕ» (инверторы); — буферные элементы с прямыми, инверсными и тристабильными выходами; — «Исключающее ИЛИ»; — триггеры D- и T-типа; — мультиплексоры конфигурации. | 40. Сумматор. Многоразрядный сумматор. Ускорение выполнения математических операций.  **Сумматор**  Сумматор — устройство, преобразующее [информационные сигналы](https://ru.wikipedia.org/wiki/%D0%A1%D0%B8%D0%B3%D0%BD%D0%B0%D0%BB) (аналоговые или цифровые) в сигнал, эквивалентный сумме этих сигналов; устройство, производящее операцию [сложения](https://ru.wikipedia.org/wiki/%D0%A1%D0%BB%D0%BE%D0%B6%D0%B5%D0%BD%D0%B8%D0%B5). Т.е. сумматорами называют лог. устройства, выполняющие арифметические суммирование кодов двоичного числа.  сумматор  В электронике сумматоры делятся на две группы:   * сумматоры аналоговых сигналов * сумматоры цифровых сигналов   **Многоразрядный сумматор**  Многоразрядный двоичный сумматор строится на основе одноразрядных сумматоров с введением соответствующих связей между разрядами. |
| 41. Полная система логических функций.  Функционально полная система логических элементов — это такой набор элементов, используя который можно реализовать любую сколь угодно сложную логическую функцию. Поскольку любая логическая функция представляет собой комбинацию простейших функций — дизъюнкции, конъюнкции и инверсии, то набор из элементов трех типов, реализующих соответственно функции И, или и НЕ, естественно, является функционально полным.  Система функций называется полной, если любую лог. функцию можно представить в виде формулы, содержащей функции этой системы. Имеет место теорема Поста: для того чтобы некоторый набор функций был полным, необходимо и достаточно, чтобы в него входили функции, не принадлежащие каждому из классов Т0, Т1, L, M, S. Т0 – класс функций сохраняющих 0. В него входят функции, у которых f (0,0,...,0) = 0 . Т1 – класс функций, сохраняющих 1, т.е. таких у которых f (1,1,...,1) = 1. | 42. Искусство упр-я сложностью. Цифровая абстракция.  Одной из характеристик, отличающих профессионального инженера-электронщика или программиста от дилетанта, является систематический подход к управлению сложностью многоуровневой системы.  Критически важный принцип управления сложностью системы – абстракция, подразумевающая исключение из рассмотрения тех элементов, которые в данном конкретном случае несущественны для понимания работы этой системы. Любую систему можно рассматривать с различных уровней абстракции. | 43. Логические элементы. Таблицы истинности. Обозначения элементов в разных представления.  **Логические элементы**  **Логический элемент** — элемент, осуществляющий определенные логические зависимость между входными и выходными сигналами. **Логический элемент** – это такая схемка, у которой несколько входов и один выход. Каждому состоянию сигналов на входах, соответствует определенный сигнал на выходе.  **Таблицы истинности**  Для того, чтобы понять как он работает, нужно нарисовать таблицу, в которой будут перечислены состояния на выходе при любой комбинации входных сигналов. Такая таблица называется «**таблица истинности**».  **Обозначения элементов в разных представлении**  **Элемент «И» (AND) / Элемент «ИЛИ» (OR) (**По-другому, его зовут «дизъюнктор».)  Элемент ИЛИЭлемент И  **Элемент «НЕ» (NOT) (**Чаще, его называют «инвертор».**) / Элемент «И-НЕ» (NAND)**  Элемент НЕЭлемент И-НЕ  **Элемент «ИЛИ-НЕ» (NOR) / Элемент «Исключающее ИЛИ» (XOR)**  Элемент ИЛИ-НЕЭлемент исключающее ИЛИ | 44. За пределами цифровой абстракции. Напряжение питания. Логические уровни. Допускаемые уровни шумов.  Цифровая система оперирует дискретными переменными. Однако для представления этих переменных используются непрерывные физические величины, такие как напряжение в электрической цепи, положение шестеренок в механической передаче или уровень жидкости в гидравлическом цилиндре. Задача разработчика цифровой системы – определить, каким образом непрерывно-меняющаяся величина соотносится с конкр. знач. дискретной переменной.  **Напряжение питания**  Предположим, что минимальное напряжение в электронной цифровой системе, называемое также напряжением земли (GND), составляет 0 В. Самое высокое напряжение в системе поступает от блока питания и обозначается VDD.  **Логические уровни**  Источник выдает выходной сигнал низкого напряжения (0) в диапазоне от 0 В до VOL или выходной сигнал высокого напряжения в диапазоне от VOH до VDD. Если приемник получает на вход сигнал в диапазоне от 0 до VIL, он рассматривает такой сигнал как нуль. Если приемник получает на вход сигнал в диапазоне от VIH до VDD, он рассматривает такой сигнал как единицу. Если же по какой-либо причине, например, наличия шумов или неисправности одного из элементов схемы, напряжение сигнала на входе приемника падает настолько, что попадает в запретную зону между VIL и VIH, то поведение этого логического элемента становится непредсказуемым. VOH и VOL называются соответственно высоким и низким логическими уровнями, а VIH и VIL называются соответственно высоким и низким логическими уровнями входа.  **Допускаемые Уровни Шумов**  Допускаемый уровень шумов – это то максимальное количество шума, присутствие которого в выходном сигнале источника не мешает приемнику корректно интерпретировать значение полученного сигнала. Значения нижнего допускаемого уровня шумов и верхнего допускаемого уровня шумов определяются следующим образом: NML = VIL – VOL; NMH = VO – VIH. |
| 45. Передаточная характеристика. Статическая дисциплина.  **Передаточная Характеристика**  Для понимания предела цифровой абстракции мы должны рассмотреть поведение логических вентилей с аналоговой точки зрения.  Передаточная характеристика (DС transfer characteristics) какого-либо логического вентиля описывает напряжение на выходе этого элемента как функцию напряжения на его входе, когда входной сигнал изменяется настолько медленно, что выходной сигнал успевает изменяться вслед за ним. Такая характеристика называется передаточной, поскольку описывает взаимосвязь между входным и выходным напряжением.  **Статическая Дисциплина**  Для того, чтобы избежать попадания входных сигналов в запретные зоны, логические вентили должны разрабатываться в соответствии с принципом статической дисциплины (static discipline). Принцип статической дисциплины требует, чтобы при условии наличия логически корректных сигналов на входе каждый элемент системы выдавал логически корректные сигналы на выходе.  Применение принципа статической дисциплины ограничивает свободу разработчика в выборе аналоговых элементов для построения цифровых систем, однако помогает обеспечить простоту и надежность разрабатываемых цифровых схем. Используя этот принцип, разработчик поднимается с аналогового уровня абстракции на цифровой, что увеличивает производительность проектировщика, избавляя его от рассмотрения излишних деталей. | 46. Биполярные и КМОП транзисторы. Полупроводники. Конденсаторы. n-МОП и p-МОП-транзисторы.  **Биполярные и КМОП транзисторы**  Биполярный транзистор – это полупроводниковый прибор с двумя взаимодействующими между собой р-п–переходами и тремя или более выводами.  КМОП (комплементарная логика на [транзисторах](https://science.fandom.com/ru/wiki/%D0%A2%D1%80%D0%B0%D0%BD%D0%B7%D0%B8%D1%81%D1%82%D0%BE%D1%80) металл-оксид-полупроводник) — техн-ия построения электрон. схем.  Полупроводник — [материал](https://ru.wikipedia.org/wiki/%D0%9C%D0%B0%D1%82%D0%B5%D1%80%D0%B8%D0%B0%D0%BB), по [удельной проводимости](https://ru.wikipedia.org/wiki/%D0%AD%D0%BB%D0%B5%D0%BA%D1%82%D1%80%D0%B8%D1%87%D0%B5%D1%81%D0%BA%D0%B0%D1%8F_%D0%BF%D1%80%D0%BE%D0%B2%D0%BE%D0%B4%D0%B8%D0%BC%D0%BE%D1%81%D1%82%D1%8C) занимающий промежуточное место между [проводниками](https://ru.wikipedia.org/wiki/%D0%9F%D1%80%D0%BE%D0%B2%D0%BE%D0%B4%D0%BD%D0%B8%D0%BA_(%D1%8D%D0%BB%D0%B5%D0%BA%D1%82%D1%80%D0%B8%D1%87%D0%B5%D1%81%D1%82%D0%B2%D0%BE)) и [диэлектриками](https://ru.wikipedia.org/wiki/%D0%94%D0%B8%D1%8D%D0%BB%D0%B5%D0%BA%D1%82%D1%80%D0%B8%D0%BA), и отличающийся от проводников сильной зависимостью удельной проводимости от концентрации примесей, температуры и воздействия различных видов [излучения](https://ru.wikipedia.org/wiki/%D0%98%D0%B7%D0%BB%D1%83%D1%87%D0%B5%D0%BD%D0%B8%D0%B5). Основным свойством полупроводников является увеличение электр. проводимости с ростом температуры.  **Конденсатор** представляет из себя пассивный радиоэлемент, который обладает таким свойством, как сохранение электрического заряда на своих обкладках, если, конечно, перед этим его зарядить каким-нибудь источником питания.  **n-МОП и p-МОП-транзисторы.**  Полное название такого радиоэлемента на английский манер звучит как Metal Oxide Semiconductor Field Effect Transistors (MOSFET), что в переводе полевой транзистор со структурой Металл Оксид Полупроводник или просто МОП-транзистор. Почему МОП-транзистор также называют МДП-транзистором и транзистором с изолированным затвором.  строение полевого транзистора | 47. Логический вентиль НЕ и другие на КМОП-транзисторах. Псевдо n-МОП-Логика. Потребляемая Мощность.  Логический вентиль — базовый элемент цифровой схемы, выполняющий элементарную логическую операцию, преобразуя таким образом множество входных логических сигналов в выходной логический сигнал. Логика работы вентиля основана на битовых операциях с входными цифровыми сигналами в качестве операндов.  Схема демонстрирует, как можно построить логический элемент НЕ, используя КМОП-транзисторы. На этой схеме треугольник обозначает напряжение земли GND, а горизонтальная линия обозначает напряжение питания VDD.  **Псевдо n-МОП-Логика**  При использовании псевдо n-МОП-логики, или просто псевдо-логики, медленный стек из p-МОП транзисторов заменяют одним «слабым» p-МОП транзистором, который всегда находится во включенном состоянии. Такой транзистор часто называют слабым подтягивающим транзистором.  **Потребляемая мощность**  Потребляемая мощность – это количество энергии, потребляемой системой в единицу времени. Именно потребляемая мощность определяет время автономной работы без подзарядки батареи любого портативного устройства, такого как сотовый телефон или ноутбук. Не стоит думать, однако, что потребляемая мощность – второстепенный параметр для стационарных устройств. | 48. Проектирование комбинационной логики. От логики к логическим элементам. Что такое Х и Z: способы сопряжения микросхем в ЭВМ.  **Проектирование комбинационной логики**  Правила комбинационной композиции говорят нам, как мы можем построить большую комбинационную схему из более маленьких комбинационных элементов. Схема является комбинационной, если она состоит из соединенных между собой элементов и выполнены следующие условия:  Каждый элемент схемы сам является комбинационным;  Каждое соединение схемы является или входом, или подсоединено к одному-единственному выходу другого элемента схемы;  Схема не содержит циклических путей: каждый путь в схеме проходит через любое соединение не более одного раза.  **Что за X и Z**  Булева алгебра ограничена значениями 0 и 1. Однако реальные схемы могут также иметь недопустимое и плавающее состояния, представляемые символами X и Z соответственно.  Символ X обозначает неизвестное логическое значение или недопустимое значение физического напряжения в соединении, не соответствующее уровням логических 0 и 1. Символ Z указывает, что напряжение в цепи не определяется ни источником ВЫСОКОГО, ни источником НИЗКОГО напряжения. Говорят, что такая цепь отключена, находится в состоянии высокого импеданса или в третьем состоянии. Буфер с тремя состояниями имеет вход A, выход Y и сигнал управления E. |

|  |  |  |  |
| --- | --- | --- | --- |
| 49. Временные характеристики цифровых микросхем. Задержка распр-ия и задержка реакции. Импульсные помехи.  **Временные характеристики цифровых микросхем**  Переход от НИЗКОГО уровня к ВЫСОКОМУ называется положительным перепадом или фронтом. Аналогично, переход от ВЫСОКОГО уровня к НИЗКОМУ называется соответственно отрицательным перепадом или срезом. Величина задержки измеряется от момента времени, когда входной сигнал А достигает уровня 50%, до момента достижения уровня 50% выходным сигналом Y. Уровень 50% – это точка, в которой сигнал находится ровно посередине между НИЗКИМ и ВЫСОКИМ логическими уровнями.  **Задержка распространения и задержка реакции**  Комбинационная логика характеризуется задержкой распространения и задержкой реакции, или отклика. Задержка распространения tpd – это максимальное время от начала изменения входа до момента, когда все выходы достигнут установившихся значений. Задержка реакции tcd – это мин. время от момента, когда вход изменился, до момента, когда любой из выходов начнет изменять свое значение.  **Импульсные помехи**  Однако может оказаться, что одиночное изменение на входе вызывает несколько выходных изменений. Это называется импульсной помехой или паразитным импульсом. Хотя паразитный импульс обычно не вызывает проблем, важно понимать, что он есть, и уметь распознавать его на временных диаграммах. | 50. Базовые комбинационные блоки. Мультиплексоры. Логика на мультиплексорах. Дешифраторы.  Комбинационные логические элементы часто группируются в «строительные блоки», используемые для создания сложных систем. Это позволяет абстрагироваться от излишней детализации уровня логических элементов и подчеркнуть функцию «строительного блока».  **Мультиплексоры** являются одними из наиболее часто используемых комбинационных схем. Они позволяют выбрать одно выходное значение из нескольких входных в зависимости от значения сигнала выбора.  Мультиплексоры могут использоваться как таблицы преобразования для выполнения логических функций.  **Дешифраторы**  В общем случае у дешифратора имеется N входов и 2N выходов. Он выдает единицу строго на один из выходов в зависимости от набора входных значений. Выходы образуют прямой унитарный код (one-hot code), называемый так потому, что в любое время только один из выходов может принимать высокий уровень. | 51. Проектирование последовательностной логики. Защёлки и триггеры. RS-триггер. D-защёлка. D-Триггер. Регистр.  **Проектирование последовательностной логики**  Последовательнстные логические схемы обладают памятью. Они могут явно запоминать предыдущие значения определенных входов, а могут «сжимать» предыдущие значения опред. входов в меньшее количество информации, называемое состоянием системы. Состояние цифр. послед. схемы – набор бит, называемый переменными состояния. Защелки и триггеры - простые последовательностные схемы, запоминающие 1 бит информации.  **Защёлки и триггеры**  Основным блоком для построения памяти является бистабильная ячейка – элемент с двумя устойчивыми состояниями.  **RS-триггер**  Одной из простейших последовательностных схем является RS-триггер, (от англ. Reset и Set), состоящий из двух перекрестно включенных элементов ИЛИ-НЕ. У защелки есть два входа – R и S и два выхода Q и Q¯. Принципы работы RS-триггера и схемы с перекрестно включенными инверторами аналогичны, но состояние защелки контролируются R и S входами, которые сбрасывают и устанавливают выход Q.  **D-защёлка**  Проблема состоит в том, что вопросы ЧТО и КОГДА в контексте изменения состояния триггера объединены его R и S входами. Подача логической единицы на эти входы определяет не только, ЧТО произойдет, но и КОГДА это произойдет. Разработка схем упрощается, если эти вопросы ЧТО и КОГДА разделены. D-триггер-защелка решает эти проблемы. У триггера есть два входа: вход данных D, определяющий, каким будет следующее состояние, и вход тактового сигнала CLK, определяющий, когда оно изменится.  **D-триггер** может быть построен из двух включенных последовательно D-защелок.  **Регистр**  N-разрядный регистр – набор из N триггеров с общим тактовым сигналом. Таким образом, все биты регистра обновляются одновременно. Регистр является ключ. блоком при построении больш-ва последовательностных схем. | 52. Триггер с функцией разрешения. Триггер с функцией сброса. Проектирование синхронных логических схем. Синхронные послед. схемы. Синх. и асинхр. схемы.  У некоторых триггеров имеется еще один вход, называемый EN, или ENABLE (разрешить). Этот вход определяет, будут ли данные загружены по фронту или нет. Такие триггеры полезны, если мы хотим загружать значения в триггер только на протяжении какого-то времени, а не по каждому фронту тактовому импульсу.  В триггере с функцией сброса добавляется еще один вход, называемый RESET (сброс). Когда на RESET подан 0, сбрасываемый триггер ведет себя как обычный D-триггер. Когда на RESET подана 1, такой триггер игнорирует вход D и сбрасывает выход в 0. Триггеры с функцией сброса полезны, когда мы хотим ускорить установление определенного состояния во всех триггерах системы при первом включении.  **Проектирование синхронных логических схем**  Вообще говоря, последовательностные схемы включают в себя все схемы, которые не являются комбинационными, то есть последовательностные схемы – это те, значение выхода которых нельзя однозначно определить, зная лишь текущие значения входов.  **Синхронные последовательностные схемы**  Это скорее последовательностные, чем комбинационные схемы. В комбинационной логике нет циклических путей и нет зависимостей состояния выхода от времени прохождения сигнала. Если на входы комбинационной логической схемы поданы определенные сигналы, то ее выход спустя некоторое время всегда установится в определенное корректное состояние.  **Синхронные и асинхронные схемы**  Синхронные схемы проектировать и использовать проще, чем асинхронные, так же как цифровые схемы проще проектировать, чем аналоговые. Почти все современные цифровые схемы являются синхронными. Асинхронные схемы иногда используются для связи между собой систем с разными тактовыми сигналами или для считывания значений со входов в произвольное время, так же как аналоговые схемы необходимы для взаимодействия с реальным миром аналоговых (непрерывных) напряжений. |
| 53. Конечные автоматы. Пример проектирования конечного автомата.  **Конечные автоматы** получили свое название из-за того, что схема с k-регистрами может находиться в одном из 2k, то есть в конечном числе, состояний. У КА М входов, N выходов и k бит состояний. На вход КА так же подается тактовый сигнал и, возможно, сигнал сброса. КА состоит из двух блоков комбинационной логики: логики перехода в следующее состояние и выходной логики, – и из регистра, в котором хранится текущее состояние. Существует два основных класса конечных автоматов, которые отличаются своими функциональными описаниями. В автомате Мура выходные значения зависят лишь от текущего состояния, в то время как в автомате Мили выход зависит как от текущего состояния, так и от входных данных. Конечные автоматы предоставляют систематический способ проектирования синхронных последоват. схем по заданному функциональному описанию.  **Пример проектирования конечного автомата**  Для того чтобы проиллюстрировать процесс проектирования конечного автомата, рассмотрим проблему создания контроллера светофора для загруженного перекрестка в студенческом городке. Студенты-инженеры гуляют по Академической улице, на которой расположены учебные корпуса и общежитие. У них нет времени читать про конечные автоматы, и они не смотрят под ноги во время передвижения. Футболисты носятся между спортзалом и столовой по Беговой улице. Они гоняют мяч туда-сюда и тоже не смотрят под ноги. Несколько студентов уже получили серьезные травмы на перекрестке, и декан попросил Бена Битдидла установить светофор, пока не произошли инциденты с летальным исходом. | 54. Конечные автоматы. Кодирование состояний. Автоматы Мура и Мили.  **Конечные автоматы** получили свое название из-за того, что схема с k-регистрами может находиться в одном из 2k, то есть в конечном числе, состояний. У КА М входов, N выходов и k бит состояний. На вход КА так же подается тактовый сигнал и, возможно, сигнал сброса. КА состоит из двух блоков комбинационной логики: логики перехода в следующее состояние и выходной логики, – и из регистра, в котором хранится текущее состояние. Существует два основных класса конечных автоматов, которые отличаются своими функциональными описаниями. В автомате Мура выходные значения зависят лишь от текущего состояния, в то время как в автомате Мили выход зависит как от текущего состояния, так и от входных данных. Конечные автоматы предоставляют систематический способ проектирования синхронных последоват. схем по заданному функциональному описанию.  **Кодирование состояний**  Основная проблема заключается в том, как определить кодировку, которая потребует наименьшее количество элементов и приведет к наименьшим задержкам в схеме. К сожалению, простого способа найти самую лучшую кодировку не существует, кроме как перепробовать все возможные, что нерационально в случае, если число состояний велико. Однако зачастую возможно найти хорошую кодировку так, чтобы связанные состояния или выходы имели общие биты. Наилучший выбор кодирования зависит от особенностей конкретного автомата.  **Автоматы Мура и Мили**  Автоматы Мили очень похожи на автоматы Мура, но значения на их выходах могут зависеть от значений на входах таким же образом, как они зависят от текущего состояния системы. Поэтому на диаграммах переходов для автоматов Миля значения выходов пишутся над стрелками. В блоке комбинационной логики, который вычисляет выходные значения, используются значения текущего состояния и входов.  Простым способом запомнить разницу между двумя типами конечных автоматов состояний является тот факт, что у автомата Мура обычно больше (Moore – more) состояний, чем у автомата Мили, решающего ту же задачу. | 55. Декомпозиция конечных автоматов. Восстановление конечных автоматов по электрической схеме.  **Декомпозиция конечных автоматов**  Проектирование сложных конечных автоматов часто упрощается, если их можно разбить на несколько более простых автоматов, взаимодействующих друг с другом таким образом, что выход одних автоматов является входом других. Такое применение принципов иерархической организации и модульного проектирования называется декомпозицией конечных автоматов. | 56. Синхронизация последовательностных схем. Временные характеристики системы. Расфазировка тактовых сигналов. Метастабильность. Синхронизаторы.  **Синхронизация последовательных схем**  Триггер копирует сигнал с D-входа на Q-выход по переднему фронту тактового сигнала. Этот процесс называется фиксацией D-сигнала по фронту тактового импульса. Поведение триггера корректно, если сигнал на D-входе стабилен (равен 0 или 1 и не изменяется) в течение переднего фронта тактового сигнала.  **Временные характеристики системы**  Периодом тактового сигнала или длительностью цикла синхронизации, Tc , называется промежуток времени между передними фронтами последовательных тактовых импульсов. Обратная величина, fc = 1/Tc , называется тактовой частотой. Увеличение тактовой частоты без изменения остальных параметров схемы приводит к увеличению ее производительности. Частота измеряется в герцах (Гц), или в циклах за одну секунду: 1 мегагерц (МГц) = 106 Гц и 1 гигагерц (ГГц) = 109 Гц.  **Расфазировка тактовых сигналов**  В предыдущих разделах предполагалось, что тактовые импульсы поступают на все регистры в одно и то же время. В действительности существует некоторый разброс этого времени. Эта неодновременность фронтов называется расфазировкой.  **Метастабильность**  **Метастабильное состояние** — состояние квазиустойчивого равновесия физ. системы, в кот. система может находиться длит. время.  **Синхронизаторы**  Наличие асинхронных входов цифровой системы, которые принимают информацию из внешнего мира, неизбежно. Например, сигналы, которые формирует человек, асинхронны. Такие асинхронные входы, если к ним относиться небрежно, могут привести к появлению метастабильных состояний в системе, что приведет к ее непредсказуемым отказам, которые крайне сложно отследить и исправить. |
| 57. Типы триггеров. Классификация триггеров. RS-триггер на элементах И-НЕ и ИЛИ–НЕ. T-, JK-, D-триггеры.  Триггер – электронная схема, обладающая двумя устойчивыми состояниями. Переход из одного устойчивого состояния в другое происходит скачкообразно под воздействием управляющих сигналов.  Типы триггеров   * RS-триггеры (триггер, который сохраняет своё предыдущее состояние при неактивном состоянии обоих входов и изм. своё состояние при подаче на 1 из его входов актив. уровня) * D-триггер (запоминает состояние входа и выдаёт его на выход. D-триггеры имеют 2 входа: информационный **D** и синхронизации **С**. Вход синхронизации С может быть статическим (потенциальным) и динамическим.) * Т-триггеры (часто называют счётным триггером, так как он является простейшим [счётчиком](https://ru.wikipedia.org/wiki/%D0%A1%D1%87%D1%91%D1%82%D1%87%D0%B8%D0%BA_(%D1%8D%D0%BB%D0%B5%D0%BA%D1%82%D1%80%D0%BE%D0%BD%D0%B8%D0%BA%D0%B0)) по модулю) * **JK-триггер** (работает так же, как RS-триггер, с одним лишь исключением: при подаче логической единицы на оба входа J и K состояние выхода триггера изменяется на противоположное, то есть выполняется операция инверсии)   Классификация триггеров:  Триггеры подразделяются на две большие группы — динамические (управляемый генератор, одно из состояний которого (единичное) характеризуется наличием на выходе непрерывной последовательности импульсов определённой частоты, а другое (нулевое) — отсутствием выходных импульсов. Смена состояний производится внешними импульсами) и статические (устройства, каждое состояние которых харак-ся неизменными уровнями выходного напряжения (выходными потенциалами): высоким — близким к напряжению питания и низким — около 0).  Триггерные схемы классифицируют также по следующим признакам:   * числу целочисленных устойчивых * числу уровней * по способу реакции на помехи — прозрачные и непрозрачные. Непрозрачные, в свою очередь, делятся на проницаемые и непроницаемые; * по составу логических элементов (триггеры на элементах И-НЕ, ИЛИ-НЕ и др.). | 58. Параллельные и последовательные регистры. Отличия в обозначения цифровых элементов в разных стандартах.  **Параллельный регистр**  В данном типе регистров триггеры соединены параллельно, то есть каждый внутренний триггер имеет свой вход D и свой выход Q, которые не зависят от других триггеров, а также вход С, который называется **тактовым входом** и для всех входящих в регистр триггеров он является общим. Параллельные регистры бывают двух типов:   * **тактируемые регистры**, которые срабатывают по **фронту сигнала** управления (вход С); * **стробируемые регистры**, которые срабатывают по **уровню входного сигнала** (вход С);   **Регистры, срабатывающие по фронту сигнала**  Данные типы регистров практически ничем не отличаются от D триггеров. Напомню, что на выходе Q устанавливается такой уровень напряжения, который был на входе D, в момент положительного фронта тактового сигнала на входе С. Так как регистр состоит из нескольких D триггеров, то и количество (4,6,8,16) запоминаемых сигналов (бит) у регистра больше.  **Регистры, срабатывающие по уровню сигнала**  Данные типы регистров являются как бы гибридом между буфером и регистром (ещё такие регистры называют **регистры-защёлки**). Принцип работы данных регистров состоит в следующем: когда на **стробирующем входе** высокий логический уровень, сигнал **поступает с входов** на выходы регистра, а если на стробирующем входе низкий уровень сигнала, регистр переходит в **режим хранения** последнего из пропущенных значений входного сигнала. Регистры-защелки не очень распространены из-за своего специфического характера работы, но в некоторых случаях удобнее, **регистров, срабатывающих по фронту сигнала** на тактовом входе. | 59. Мультиплексоры и демультиплексоры. Отличия в обозначения цифровых элементов в разных стандартах.  **Мультиплексор** – коммутатор логических сигналов, обеспечивающий передачу информации, поступающей по нескольким входным линиям связи, на одну выходную линию. Выбор вход­ной линии Аi осуществляется в соответствии с поступающим адресным кодом. При наличии m адресных входов можно реализовать M=2m комбинаций адресных сигналов, каждая из которых обеспечивает выбор одной из М вводных линий. Мультиплексор состоит из дешифратора адреса входной линии, схем И и схемы объединения ИЛИ. Мультиплексоры можно использовать для синтеза логических функций от нескольких переменных.  **Демультиплексор** – коммутатор логических сигналов, обеспечивающий передачу информации, поступающей по одному входу, и в соответствии с адресом направляющий в одну из выходных линий.  При передачах данных по общему каналу с разделением во времени нужны не только мультиплексоры, но и демультиплексоры, распределяющие данные из одного канала между несколькими приемниками информации.  Демультиплексор имеет один информационный вход n адресующих (управляющих) входов и 2n выходов. | 60. Погрешность математических операций в цифровых системах. Способы оценки. Округление.  **Абсолютной погрешностью** или, погрешностью приближенного числа называется разность между этим числом и его точным значением (из большего числа вычитается меньшее).  **Относительной погрешностью** приближенного числа называется отношение абсолютной погрешности приближенного числа к самому этому числу.  Число, заведомо превышающее абсолютную погрешность (или в худшем случае равное ей), называется **предельной абсолютной погрешностью**. Число, заведомо превышающее относительную погрешность (или в худшем случае равное ей), называется **предельной относительной погрешностью**.  **Округление** применяется для представления значений и результатов вычислений с тем количеством знаков, которое соответствует реальной точности измерений или вычислений, либо той точности, которая требуется в конкретном приложении. Округление в ручных расчётах также может использоваться для упрощения вычислений в тех случаях, когда погрешность, вносимая за счёт ошибки округления, не выходит за границы допустимой погрешности расчёта.  **Оценка погрешности**  В зависимости от характеристик измеряемой величины для определения погрешности измерений используют различные методы. Часто для оценки случайной погрешности используют стандартное отклонение, или среднеквадратическое отклонение, для которого обычно используют один из двух способов оценки (оба термина применяются как к одному, так и к другому способу):  На основании несмещённой оценки дисперсии:    На основании смещённой оценки дисперсии: |

|  |  |  |  |
| --- | --- | --- | --- |
| 61. Архитектура процессора. Основные компоненты…  **Архитектура процессора**  Архитектура процессора — количественная составляющая компонентов микроархитектуры вычисл. машины, рассматр. IT-специалистами в аспекте прикладной деятельности.  **Основные** составные части типичного персонального **компьютера**: монитор, материнская плата, центральный процессор, оперативная память, карты расширений, блок питания, оптический привод, жёсткий диск, 9 — компьютерная мышь, 10 — клавиатура.  **Способы классификации**  Имеются различные классификации архитектур процессоров как по организации (например, по количеству и сложности отдельных команд: RISC, CISC; по возможности доступа команд к памяти), так и по назначению (например, специализир. графические, математические или предназначенные для цифровой обработки сигналов).  **Многоуровневая организация**    **Контроллеры ввода-вывода**  Super I/O — название класса сопроцессоров, которые начали использоваться после 1980-х годов на материнских платах IBM PC-совместимых компьютеров. Super I/O объединяет интерфейсы различных низкоскоростных устройств. | 62. RISK, CISK, MISC, VLIW. Отличительные особенности, сфера применения. Что такое Spectre и Meltdown.  **RISK** — архитектура процессора, в котором быстродействие увеличивается за счёт упрощения инструкций: их декодирование становится более простым, а время выполнения — меньшим. Первые RISC-процессоры не имели даже инструкций умножения и деления и не поддерживали работу с числами с плавающей запятой (RISC быстрее CISC)  **CISK** — тип процессорной архитектуры, в первую очередь, с нефиксированной длиной команд, а также с кодированием арифметических действий в одной команде и небольшим числом регистров, многие из которых выполняют строго определенную функцию (В CISC процессорах одна команда может быть заменена ей аналогичной, либо группой команд, выполняющих ту же функцию)  **MISC** - ещё более простая архитектура, используемая в первую очередь для ещё большего уменьшения итоговой цены и энергопотребления процессора. Используется в IoT-сегменте и недорогих компьютерах, например, роутерах. (“спекулятивное исполнение команд” - это выполнение команды до того, как станет известно, понадобится эта команда или нет)  **VLIW** — архитектура процессоров с несколькими вычислительными устройствами. Характеризуется тем, что одна инструкция процессора содержит несколько операций, которые должны выполняться параллельно. По сути является архитектурой CISC со своим аналогом спекулятивного исполнения команд, только сама спекуляция выполняется во время компиляции, а не во время работы программы, из-за чего уязвимости Meltdown и Spectre невозможны для этих процессоров.  **Spectre** — группа аппаратных уязвимостей, ошибка в большинстве современных процессоров, имеющих спекулятивное выполнение команд и развитое предсказание ветвлений, позволяющих проводить чтение данных через сторонний канал в виде общей иерархии кэш-памяти.  **Meltdown** — аппаратная уязвимость категории утечка по стороннему каналу, обнаруженная в ряде микропроцессоров, в частности, производства Intel и архитектуры ARM. | 63. Сравнительная характеристика архитектур. В чем преимущества. Преимущества RISC. Какова проблема лицензирования архитектур.  **Сравнительная характеристика архитектур**    **Преимущества RISC**  RISC проще, легче оптимизируется, соответственно быстрее, меньше потребление, простота наращивания и отладки, инструкции фиксированной длины, не нужно делать выравнивание, проще работа с памятью, более богатая регистровая архитектура, легче делать 32/64/128 разрядов (далее везде)  **Какова проблема лицензирования архитектур**  Конкуренция | 64. Виртуальные архитектуры. Команды…  Магические JAR-файлы, которые можно запустить на любой машине - это пример виртуальной JVM-архитектуры, которая, по сути, эмулируется на целевой реальной машине.  **Команды (инструкции), предназначение, виды**  Инструкция - это не что иное, как действие, которое мы отправляем процессору. Инструкции могут быть арифметическими операциями с различными типами данных, такими как с плавающей запятой, целыми числами, вектором, скаляром, логическими операциями, операциями перемещения данных, операциями перемещения битов, операциями перехода и т. д. Они бывают нескольких типов:  • Арифметические: сложение, вычитание, умножение и т. д.  • Логические: И (логическое умножение/конъюнкция), ИЛИ (логическое суммирование/дизъюнкция), отрицание и т. д.  • Информационные: move, input, outptut, load и store.  • Команды перехода: goto, if ... goto, call и return.  • Команда останова: halt.  **Тактирование процессоров**  Быстродействие компьютера определяется тактовой частотой его процессора. Тактовая частота — количество тактов (соответственно и исполняемых команд) за секунду.  Частота нынешних процессоров измеряется в ГГц (Гигагерцы). 1 ГГц = 10⁹ Гц — миллиард операций в секунду.  **Выполнение инструкций**  Инструкции хранятся в ОЗУ в последовательном порядке. Для гипотетического процессора инструкция состоит из кода операции и адреса памяти/регистра. Внутри управляющего устройства есть 2 регистра инструкций, в кот. загружается код команды и адрес текущей исполняемой команды.  **Поток инструкций**  Современные процессоры могут параллельно обрабатывать несколько команд. Пока одна инструкция находится в стадии декодирования, процессор может успеть получить другую инструкцию. Однако такое решение подходит только для тех инструкций, которые не зависят друг от друга. Если процессор многоядерный, это означает, что фактически в нём находятся несколько отдельных процессоров с некоторыми общими ресурсами, например кэшем. |
| 65. Регистр процессора: предназначение, виды. Шины: предназначение, виды. Кэш: предназначение, виды.  **Регистр процессора**  Регистры - это память, ближайшая к существующему процессору и, следовательно, самая быстрая. Они используются для выполнения всех видов общих задач, а не только для выполнения арифметических операций.  • Регистры типа аккумулятора: используется для арифметических операций. Каждое семейство имеет разное количество записей типа аккумулятора.  • Регистры доступа к памяти: содержат адрес памяти данных, к которым мы хотим получить доступ из ОЗУ.  • Регистры данных в или из памяти: Содержат данные, скопированные из памяти (чтение) или для записи по определенному адресу памяти (запись).  • Регистры общего назначения: это регистры памяти без специальной утилиты, которые служат для хранения данных, которые должны быть вызваны как можно быстрее.  • Счетчик команд : указывает следующую инструкцию для выполнения.  **Шины**    **Кэш**  Кэш процессора — кэш, используемый процессором компьютера для уменьшения среднего времени доступа к компьютерной памяти. Является одним из верхних уровней иерархии памяти. Кэш использует небольшую, очень быструю память (обычно типа SRAM), которая хранит копии часто  используемых данных из основной памяти. У процессора есть механизмсохранения инструкций в кэш. | 66. Что такое суперскалярная архитектура. Ее особенности. Предсказатели переходов. Иерархия памяти. Ветвление  **Суперскалярная архитектура**  Это означает, что в любой момент времени процессор одновременно выполняет на каждом этапе конвейера множество инструкций. Он может также ожидать ещё сотни других, чтобы начать их выполнение, и для того, чтобы иметь возможность одновременного выполнения нескольких инструкций внутри процессоров есть несколько копий каждого этапа конвейера. Если процессор видит, что к выполнению готовы две инструкции, и между ними нет зависимости, то он не ждёт, пока они завершатся по отдельности, а выполняет их одновременно.  **Предсказатели переходов**  Инструкции переходов (ветвлений) схожи с конструкциями «if» для процессора. Один набор инструкций выполняется, если условие истинно, а другой — если оно ложно. Например, нам нужно сравнить два числа, и если они равны, выполнить одну функцию, а если не равны, то выполнить другую. Эти инструкции ветвления применяются чрезвычайно часто и могут составлять примерно 20% всех инструкций в программе.  **Иерархия памяти**  Кэш L1 — самый маленький и быстрый, L2 находится посередине, а L3 — самый крупный и медленный из всех кэшей. Выше кэшей в иерархии находятся мелкие регистры, хранящие во время вычислений единственное значение данных. По порядку величин эти регистры являются самыми быстрыми устройствами хранения в системе. Когда компилятор преобразует высокоуровневую программу в язык ассемблера, он определяет наилучший способ использования этих регистров. | 67. Что такое гетерогенные вычисления. FPGA-акселератор? Сфера применения. Перспективные направления развития вычислительных систем.  **Гетерогенные вычисления**  Эта методика заключается во включении в одну систему множества различных вычислительных элементов. Большинство из нас пользуется преимуществами такого подхода в виде отдельных GPU в компьютерах. Центральный процессор очень гибок и может с приличной скоростью выполнять широкий диапазон вычислительных задач. С другой стороны, GPU спроектированы специально для выполнения графических вычислений, например, перемножения матриц. Они очень хорошо с этим справляются и на порядки величин быстрее ЦП в подобных видах инструкций. Перенеся часть графических вычислений с ЦП на GPU, мы можем ускорить расчёты. Любой программист может оптимизировать ПО, изменив алгоритм, но оптимизировать оборудование гораздо сложнее.  **FPGA-акселератор**  В отличие от традиционных вычислительных элементов наподобие ЦП и GPU, имеющих фиксированную внутреннюю архитектуру, FPGA гибки. Это почти программируемое оборудование, которое можно настраивать в соответствии с нуждами компании.  Если кому-то нужно распознавание изображений, то он реализует эти алгоритмы в «железе». Если кто-то хочет симулировать работу новой аппаратной архитектуры, то перед изготовлением её можно протестировать на FPGA. FPGA обеспечивает большую производительность и энергоэффективность, чем GPU, но всё равно меньше, чем у ASIC (application specific integrated circuit — интегральная схема специального назначения). Другие компании, например, Google и Nvidia, разрабатывают отдельные ASIC машинного обучения для ускорения распознавания и анализа изображений. |