**1.Системы счисления. Перевод чисел из одной системы счисления в другую. Метод преобразования с использованием весов разрядов.**

**Система счисления** — символический метод записи чисел, представление чисел с помощью письменных знаков.

Все **системы счисления** можно подразделить на **позиционные (Количественная оценка числа** определяется как сумма произведений значения цифр, составляющих запись числа, умноженных на вес позиции, в которой̆ располагается цифра. (10сс, 2сс, 2-10сс)) и **непозиционные** (вес в количественном измерении записанного числа не зависит от местоположения данной̆ цифры в записи этого числа (римская сс))

Наличие различных систем счисления предполагает использование способов перевода записи числа из одной системы в другую. Для этой цели применяются следующие методы преобразований:

* преобразования с использованием весов разрядов в исходной и в искомой записи числа
* деления (умножения) на новое основание
* с использованием особого соотношения заданной и искомой систем счисления.

Метод имеет две 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. Например, при преобразовании целого двоичного числа *N2* = 110011010 в десятичную систему счисления исходное число представляется в расширенной записи *N* = 28 + 27 + 24 + 23 + 21 и рассчитывается вес отдельных (ненулевых) двоичных разрядов в десятичной системе счисления: 256, 128, 16, 8, 2. Затем искомая запись числа определяется как сумма весов всех ненулевых разрядов записи числа в заданной системе счисления: 256 + 128 + 16 + 8 + 2 = 410. 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.**

Наличие различных систем счисления предполагает использование способов перевода записи числа из одной системы в другую. Для этой цели применяются следующие методы преобразований:

* преобразования с использованием весов разрядов в исходной и в искомой записи числа
* деления (умножения) на новое основание
* с использованием особого соотношения заданной и искомой систем счисления.

Метод деления (умножения) имеет две разновидности соответственно для преобразования целых и дробных чисел.

* Преобразование целых чисел.

1. необходимо разделить исходное число N q1на новое основание q2, при этом получив целое частное и остаток;
2. полученный остаток снова необходимо разделить на q2, процесс деления продолжается до тех пор, пока частное будет не меньше нового основания q2. Если очередное сформированное частное будет меньше , чем q2, то процесс формирования записи заданного числа в новой системе с основанием q2 считается законченным, а в качестве искомых разрядов новой записи числа используются результаты выполненных операций деления следующим образом:
3. в качестве старшего разряда берется значение последнего частного, для остальных разрядов используются значения остатков в порядке, обратном порядку их получения.

* Преобразование дробных чисел.

1. определяется количество разрядов «n» в записи числа Mq2 в новой системе счисления (см. подраздел 1.2.1);
2. исходное число Mq1 умножается на q2, при этом будет получено смешанное число;
3. дробная часть полученного произведения снова умножается на q2 и т.д.; процесс умножения повторяется n раз. В качестве искомых разрядов новой записи числа используются результаты выполненных операции деления следующим образом:
4. в качестве первого старшего разряда искомой записи числа в новом основании берется значение целой части первого произведения, в качестве второго старшего разряда искомой записи числа в новом основании берется значение целой части второго произведения и т.д.**3.Перевод чисел из одной системы счисления в другую. Метод с использованием особого соотношения оснований исходной и искомой систем счисления.**

Наличие различных систем счисления предполагает использование способов перевода записи числа из одной системы в другую. Для этой цели применяются следующие методы преобразований:

* преобразования с использованием весов разрядов в исходной и в искомой записи числа
* деления (умножения) на новое основание
* с использованием особого соотношения заданной и искомой систем счисления.

Данный метод 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 или *q*2*m* = *q*1 (метод, когда есть промежуточное преобразование в другую сс)

* Для 1-го условия (запись заданного числа *N*=*аnаn*-1*аn-*2...*а*1*а*0 в системе с новым основанием *q*2 формируется следующим образом)

1. исходная запись числа разбивается на группы по *m* разрядов, двигаясь от точки вправо и влево (недостающие разряды в крайних группах (слева и справа) дополняются нулями;
2. каждой полученной группе ставится в соответствие цифра новой системы счисления;
3. искомая запись заданного числа в новой системе счисления образуется из цифр, соответствующих группам, на которые была разбита исходная запись.

* Для 2-го условия (для заданного в системе с основанием *q*1 числа *Nq*1 *= аnаn*-1*аn*-2... *а*1*а*0 запись его в системе с новым основанием *q*2 определяется следующим образом)

1. каждому разряду *ai*исходной записи числа ставится в соответствие его *m-*разрядный эквивалент в системе счисления с основанием *q*2;
2. искомая запись всего заданного числа формируется за счет объединения всех полученных *m*-разрядных групп.

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

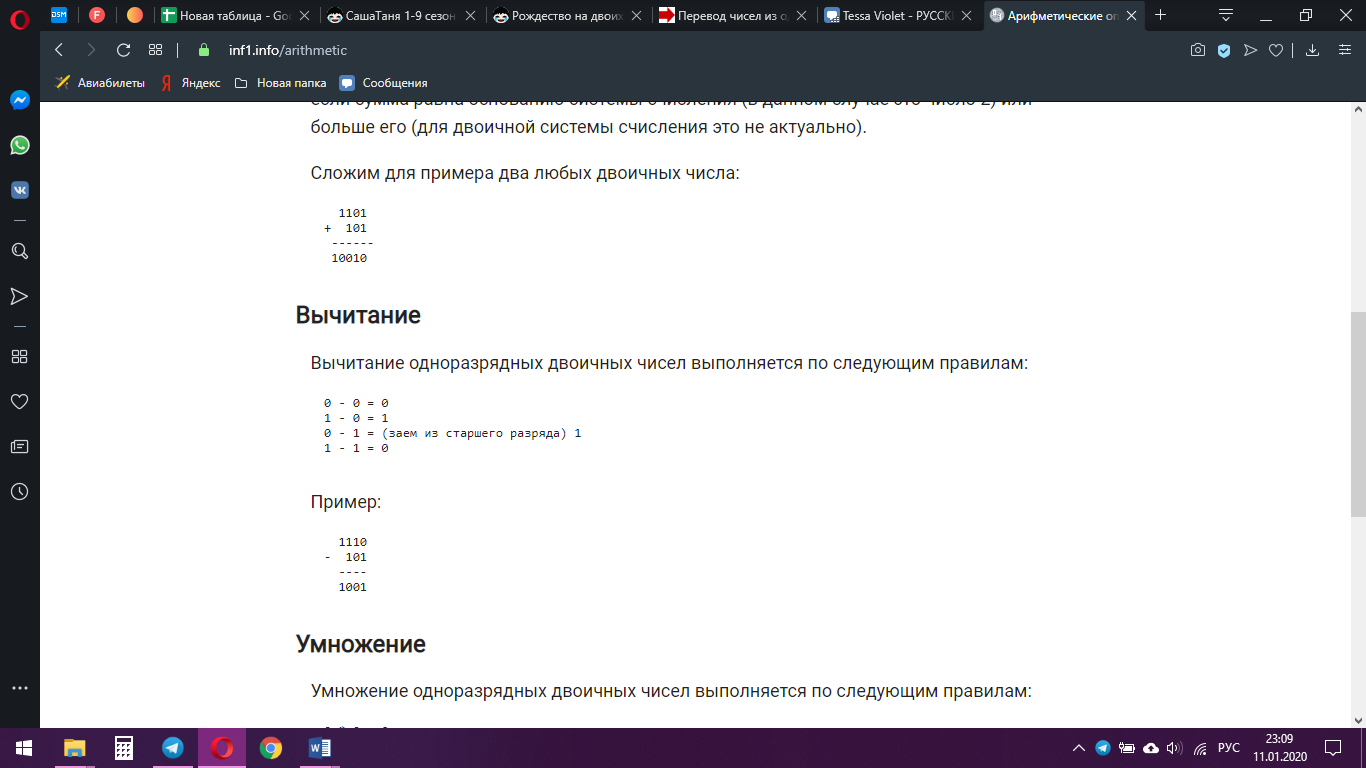
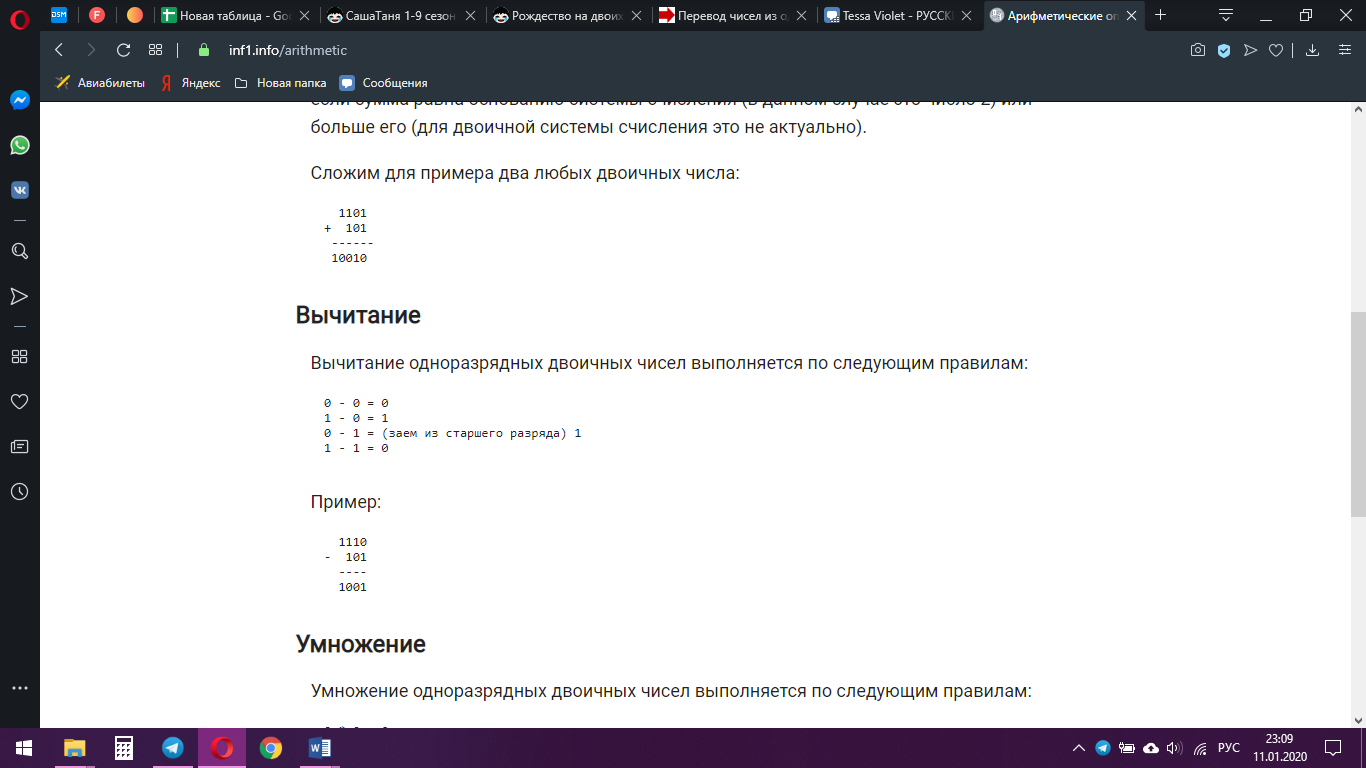
При выполнении любой операции результат ищется согласно соответствующим правилам, которые удобно представлять в табличной форме, где для всех возможных комбинаций значений одноразрядных операндов приводятся значения результата

* Правила сложения в двоичной системе счисления

Сложение производится как сложение в столбик в 10 сс, записываются необходимые числа построчно, далее по правилам сложения производится сложение каждого разряда чисел (0+0=0; 1+0=1; 0+1=1; 1+1=10 (пишется 0, 1 переносится на старший разряд, т.к происходит переполнение младшего разряда). Переполнение возникает в случае, если сумма равна основанию системы счисления или больше его

* Правила вычитания в двоичной системе счисления

Вычитание производится как сложение в столбик в 10 сс, записываются необходимые числа построчно, далее по правилам вычитания производится вычитание каждого разряда чисел (0-0=0;  
1-0=1; 0-1= 11(1 пишем в младший разряд и занимаем из старшего разряда 1);1 - 1 = 0)



**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 соответствующего разряда множителя. **Возможные методы реализации операции умножения можно классифицировать по двум 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)
* Для реализации умножения с младшего разряда требуется: требуется n-разрядный сумматор, 2n-разрядный регистр промежуточного произведения, n-разрядный регистр для хранения множимого и n-разрядный регистр для хранения множителя
* Для реализации умножения со старшего разряда требуется: 2n-разрядный сумматор, 2n-разрядный регистр промежуточного произведения, 2n- разрядный регистр для хранения и сдвига множимого и n-разрядный регистр для хранения множителя;

**6. IEEE754. Специальные числа. Зачем нулю знак.**

**IEEE 754** — стандарт, описывающий формат представления чисел с [плавающей точкой](https://ru.wikipedia.org/wiki/%D0%9F%D0%BB%D0%B0%D0%B2%D0%B0%D1%8E%D1%89%D0%B0%D1%8F_%D0%B7%D0%B0%D0%BF%D1%8F%D1%82%D0%B0%D1%8F). Используется в программных и аппаратных реализациях арифметических, стандарт описывает:

* формат [чисел с плавающей точкой](https://ru.wikipedia.org/wiki/%D0%A7%D0%B8%D1%81%D0%BB%D0%BE_%D1%81_%D0%BF%D0%BB%D0%B0%D0%B2%D0%B0%D1%8E%D1%89%D0%B5%D0%B9_%D0%B7%D0%B0%D0%BF%D1%8F%D1%82%D0%BE%D0%B9): [мантисса](https://ru.wikipedia.org/wiki/%D0%AD%D0%BA%D1%81%D0%BF%D0%BE%D0%BD%D0%B5%D0%BD%D1%86%D0%B8%D0%B0%D0%BB%D1%8C%D0%BD%D0%B0%D1%8F_%D0%B7%D0%B0%D0%BF%D0%B8%D1%81%D1%8C), экспонента (показатель), знак числа;
* представление специальных чисел
* методы, используемые для преобразования числа при выполнении математических операций;
* исключительные ситуации, например [деление на ноль](https://ru.wikipedia.org/wiki/%D0%94%D0%B5%D0%BB%D0%B5%D0%BD%D0%B8%D0%B5_%D0%BD%D0%B0_%D0%BD%D0%BE%D0%BB%D1%8C), [переполнение](https://ru.wikipedia.org/wiki/%D0%90%D1%80%D0%B8%D1%84%D0%BC%D0%B5%D1%82%D0%B8%D1%87%D0%B5%D1%81%D0%BA%D0%BE%D0%B5_%D0%BF%D0%B5%D1%80%D0%B5%D0%BF%D0%BE%D0%BB%D0%BD%D0%B5%D0%BD%D0%B8%D0%B5), [потеря значимости](https://ru.wikipedia.org/wiki/%D0%90%D0%BD%D1%82%D0%B8%D0%BF%D0%B5%D1%80%D0%B5%D0%BF%D0%BE%D0%BB%D0%BD%D0%B5%D0%BD%D0%B8%D0%B5), работа с [денормализованными числами](https://ru.wikipedia.org/wiki/%D0%94%D0%B5%D0%BD%D0%BE%D1%80%D0%BC%D0%B0%D0%BB%D0%B8%D0%B7%D0%BE%D0%B2%D0%B0%D0%BD%D0%BD%D1%8B%D0%B5_%D1%87%D0%B8%D1%81%D0%BB%D0%B0" \o "Денормализованные числа) и другие;
* операции: арифметические и другие.

Специальные числа: ноль, бесконечность и неопределенность

* число 0

В IEEE754 число «0» представляется значением с порядком, равным E=Emin-1 (для single это -127) и нулевой мантиссой. Введение нуля как самостоятельного числа позволило избежать многих странностей в арифметике.

* Бесконечность и неопределенность   
  В IEEE754 предусмотрено представление для специальных чисел, работа с которыми вызывает исключение. К таким числам относится бесконечность (±∞) и неопределенность (NaN). Эти числа позволяет вернуть адекватное значение при переполнении. Бесконечности представлены как числа с порядком E=Emax+1 и нулевой мантиссой, а неопределенность - как число, в котором E=Emax+1, а мантисса не нулевая.

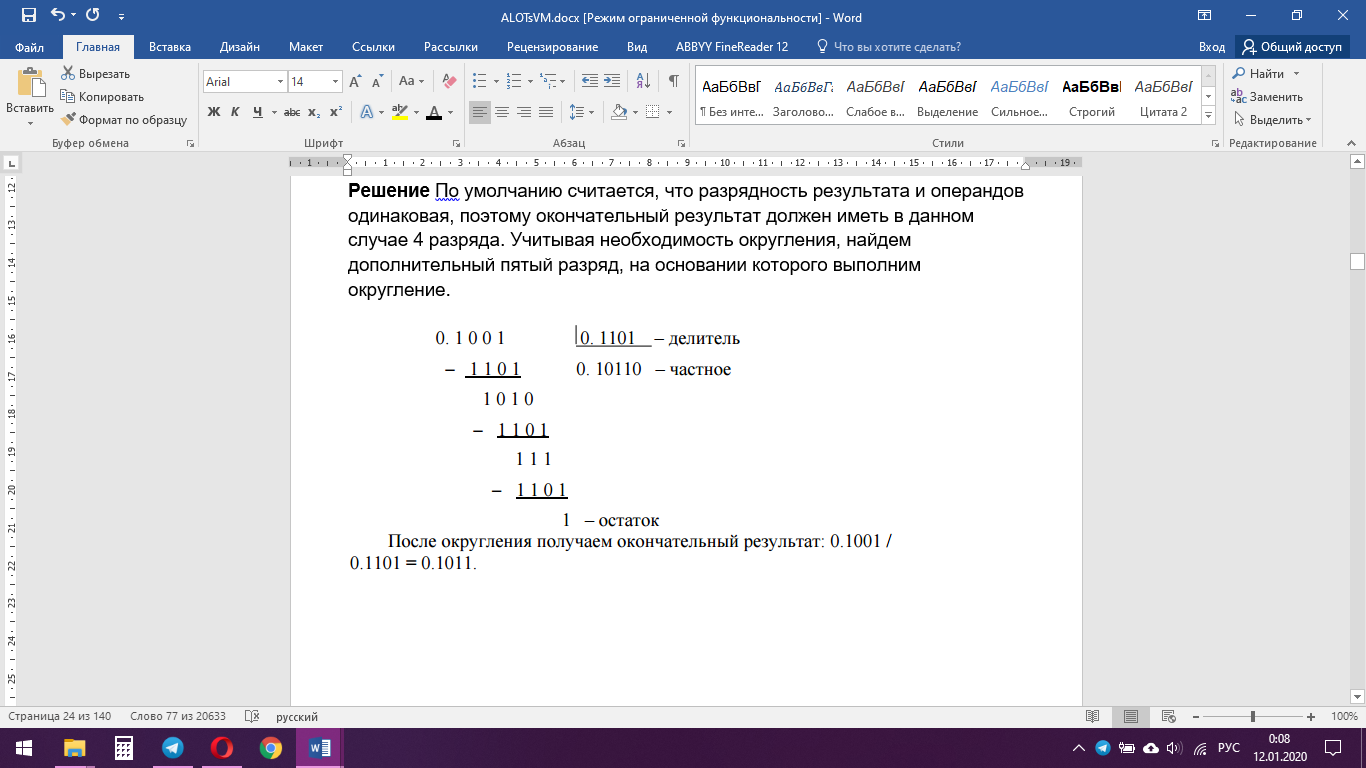
«Отрицательный ноль» и «положительный ноль» — это не [числа](https://ru.wikipedia.org/wiki/%D0%92%D0%B5%D1%89%D0%B5%D1%81%D1%82%D0%B2%D0%B5%D0%BD%D0%BD%D1%8B%D0%B5_%D1%87%D0%B8%D1%81%D0%BB%D0%B0) в обычном смысле, а абстракции, представляющие бесконечно малую функцию, [сходящуюся](https://ru.wikipedia.org/wiki/%D0%9F%D1%80%D0%B5%D0%B4%D0%B5%D0%BB_(%D0%BC%D0%B0%D1%82%D0%B5%D0%BC%D0%B0%D1%82%D0%B8%D0%BA%D0%B0)) к [нулю](https://ru.wikipedia.org/wiki/0_(%D1%87%D0%B8%D1%81%D0%BB%D0%BE)), соответственно. 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.

**7.Деление двоичных чисел (общие правила).**

Деление в принципе является неточной 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 описывается следующим образом. Пусть А - делимое, В - делитель, С - частное.

A=0,α1α2...αn; B=0,b1b2...bm, C=0,c1c2...cr.

На каждом шаге определяется остаток Аi=Аi–1 – В\*2-i, производится анализ, если остаток Аi>0, то старший разряд частного Ci=l, производится левый сдвиг и снос следующего разряда числа А, переход к определению следующего остатка. Если Ai<0, то Сi=0 и восстанавливается остаток Ai=Ai-1+B\*2-iна следующем шаге после сдвига определяется новый остаток и т.д.

**9.Деление двоичных чисел без восстановления остатка**

Метод деления бинарных чисел без восстановления промежуточных остатков выполняется в последовательности:

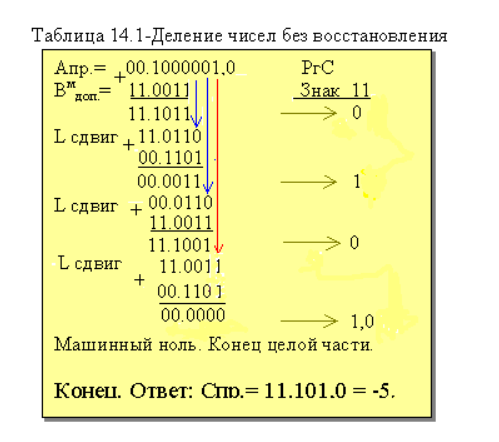
* определить знак частного по формуле SgC= SgA+SgB;;
* представить числа в дополнительном коде в машинном изображении, делимое берется в прямом коде с положительным знаком, а делитель берется в дополнительном коде с отрицательным знаком;
* присвоить сумматору значение См:=Амдоп , РгВ := Bмдoп; РгС:=0;
* устраняем дробную часть в делителе, перенося запятую вправо на n разрядов (по аналогии с десятичной системой счисления) и, чтобы дробь не изменилась, в делимом также переносим вправо запятую на n разрядов;
* начиная со старших разрядов, к делимому прибавляют делитель в дополнительном коде, что равносильно вычитанию из делимого делителя и анализируют знак промежуточного остатка:

1) если знак промежуточного остатка 00, то в регистр частного РгС записывается 1, начиная со старшего разряда. Остаток сдвигается на один разряд влево и сносится последующий разряд делимого не участвующий до этого в делении. После этого, промежуточный остаток подготовлен к последующему прибавлению делимого в дополнительном коде;

2) если знак промежуточного остатка 11, то в регистр частного Рг.С записывается 0, начиная со старшего разряда. Остаток сдвигается на один разряд влево и сносится последующий разряд делимого не участвующий до этого в делении. После этого, промежуточный остаток подготовлен к последующему прибавлению к нему делимого в прямом коде со знаком 00;

-действия предыдущего пункта повторяются до получения машинного нуля или заданной точности вычисления (количество разрядов дроби после запятой целой части числа). Запятая дроби устанавливается в частном после сноса последнего разряда целой части делимого.

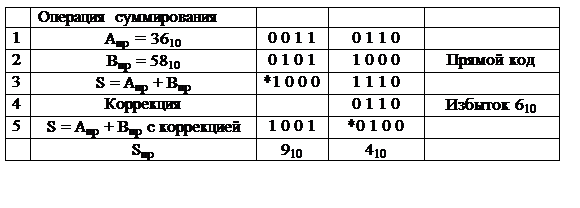
- знак результату присваивается в соответствии с пунктом 1. Результат деления представлен в регистре частного в прямом коде.



**10.Двоично-десятичная арифметика. Сложение и вычитание двоично-десятичных чисел.**

В ЭВМ часто предусматривается обработка чисел не только в двоичной системе счисления, но в двоично-десятичной. Двоично-десятичная сс –это сс похожая на 2-ю, только число переводится не целиком, а по цифрам, число в 2-10 сс записывается по 4 знака(тетрада). В отличие от сложения в двоичной сс необходимо производить коррекцию итогового результата, коррекцию производят по тетрадам если итог сложения тетрад получился больше 9 или на тетраду был перенос/займ 1 из младшей тетрады, коррекцию производят путем добавления или наоборот, вычитания 6 (0110) к тетраде.   
**Сложение** происходит по правилам сложения в двоичной сс (Сложение производится как сложение в столбик в 10 сс, записываются необходимые числа построчно, далее по правилам сложения производится сложение каждого разряда чисел (0+0=0; 1+0=1; 0+1=1; 1+1=10 (пишется 0, 1 переносится на старший разряд, т.к происходит переполнение младшего разряда).

**Вычитание** производится как сложение в столбик в 10 сс, записываются необходимые числа построчно, далее по правилам вычитания производится вычитание каждого разряда чисел (0-0=0;  
1-0=1; 0-1= 11(1 пишем в младший разряд и занимаем из старшего разряда 1);1 - 1 = 0)



**11.** **Кодирование алгебраических чисел. Дополнительный и обратный коды двоичных чисел.**

Для представления чисел со знаком используются специальные коды: прямой код (кодируется только знаковая информация), обратный код(используется для [выполнения арифметических операций вычитания, умножения, деления, через сложение](https://math.semestr.ru/inf/operation.php)) и дополнительный код(имеет такое же назначение, как и обратный код числа). Во всех трёх случаях используется специальный 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.

Форматирование числа в дополнительный и обратные коды зависит от его знака. Если число положительное, число в обратном и дополнительном коде будет таким же, как и в прямом. Если число отрицательное, то для его форматирования в обратный код, необходимо произвести замену всех цифр, входящих в число на обратные (0 на 1, и 1 на 0), форматирование числа в дополнительный код происходит путем прибавления 1 к младшему разряду числа в обратном коде.

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

Дополнительный

При использовании дополнительного или обратного кода операция вычитания заменяется операцией сложения с изменением знака второго операнда. При сложении чисел, представленных в дополнительном коде, выполняется сложение разрядов по правилам двоичной арифметики по всей длине записи чисел, не обращая внимание на границу, разделяющую знаковое и модульные поля. **Переполнение знакового поля игнорируется!** Исходя из абсолютного значения операндов, разрядность представления модульной части n должна быть равна количеству разрядов большего операнда. Мы используем две операции: сложение и вычитание, поэтому возможно переполнение из-за переноса из старшего разряда, необходимо длину модульной части на один разряд больше, т.е n+1

Обратный

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.

**13.Модифицированные коды**

При расчете разрядности n модульного поля весьма трудно бывает учесть диапазон значений результатов, в следствии чего, возникает переполнение. 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.

Если в результате сложения чисел в модифицированном коде полученный результат имеет в поле знака одинаковые значения в обоих разрядах (00 или 11), то переполнения нет, если же разряды знакового поля имеют не одинаковые значения (10 или 01), то имеет место переполнение. При этом, если в поле знака имеет место значение 01 – результат положительный, а если 10, то полученный результат отрицательный (основным носителем знака числа является левый разряд знакового поля).

**14.Логические операции с двоичными кодами: логическое суммирование, логическое умножение, логическое отрицание, суммирование по модулю два, логические сдвиги**.

Над двоичными кодами могут выполняться различные логические операции, среди которых особое место занимают:

* *логическое суммирование* (ИЛИ, ОR, «∨»,дизъюнкция);

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, но может и в нескольких операндах в *i*-м разряде имеет место единица. 10001101 ∨ 11110000 = 11111101.

* *логическое умножение* (И, AND, «∧»);

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

* *отрицание* (НЕТ, NOT, «*x*», т.е. штрих над отрицаемым *x*); 11001->00110
* *суммирование по модулю* 2 (mod 2, « ⊕ », *исключающее или ,* строгая дизъюнкция, XOR)

Операция выполняется над двумя кодами и генерирует код той же разрядности, что и операнды, у которого в некотором *i*-м разряде находится единица, если только одном операнде в *i*-м разряде имеет место единица. 10001101 ∨ 11110000 = 01111101.

* *операции сдвига*.

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 вофлаге 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).

Результат получается в обратном дополнительном коде, чтобы получить нормальный код нужно отнять 1 и инвертировать биты!!!

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

Числовая информация представляется в машине в форме с фиксированной или с плавающей точкой. При представлении с фиксированной точкой положение последней в записи числа фиксировано. Как правило, при использовании фиксированной точки числа представляются в виде целого числа или правильной дроби Точка в записи числа не отображается, а так как она находится всегда в одном месте, то указание на её положение в записи числа отсутствует. При *n*-разрядном представлении модульной части формат с фиксированной точкой обеспечивает диапазон изменения абсолютного значения числа *А*, для которого выполняется неравенство2*n* > ú*A*ú ≥ 0.Одним из важнейших параметров представления чисел является ошибка представления. Ошибка представления может быть абсолютной (D) или относительной (d). Для фиксированной точки максимальные значения этих ошибок.в худшем случае относительная ошибка при фиксированной точке может достигать сравнительно большого значения – 50%.

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**

При представлении числа *с плавающей точкой* число в общем случае представляет собой смешанную дробь и имеет формат 1р. 2р. 3р…кр. «.» (к+1р) (к+2р)…(n-1р) nр.

Местоположение точки в записи числа может быть различным, а так как сама точка в записи числа не присутствует, то для однозначного задания числа необходима не только его запись, но и информация о том, где в записи числа располагается точка, отделяющая целую и дробную части. Поэтому в случае с плавающей точкой число *Х* представляется в виде двух частей: ***мантисса* (*х*м),** отображающая запись числа, представляется в виде правильной дроби с форматом фиксированной точки; ***порядок*** **(*х*п)**, отображающий местоположение в этой записи точки, представляется в виде целого числа с форматом фиксированной точки. Количественная оценка числа *Х* определяется как: *Х* = *qx*п × *х*м, где *q* – основание системы счисления. 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.

При сложении чисел с плавающей точкой существует следующий алгоритм:

* сложить порядок в любом модифицированном коде
* Сдвинуть меньшую мантиссу вправо на итог сложения
* сложить 2 мантиссы в любом модифицированном коде
* записать ответ в виде: знак итогового числа. порядок большего числа. мантисса...

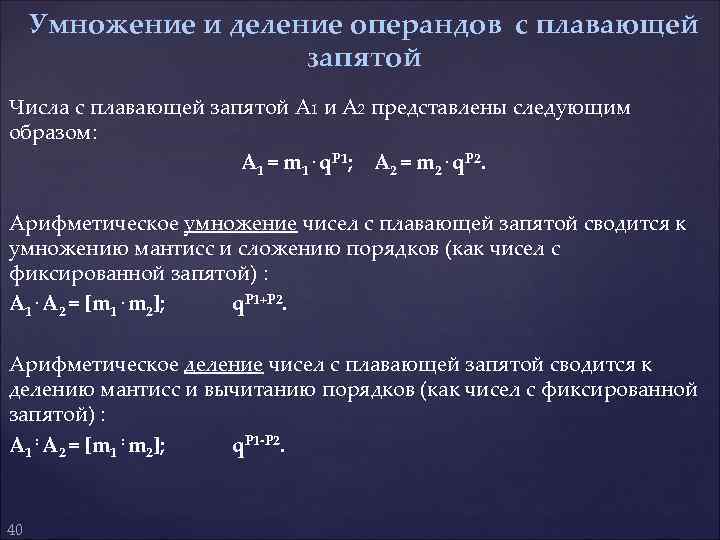
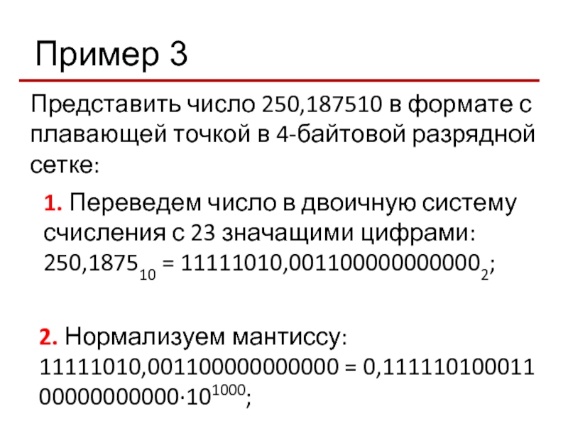
**19.Умножение чисел, представленных в формате с плавающей точкой. Деление чисел, представленных в формате с плавающей точкой.**

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



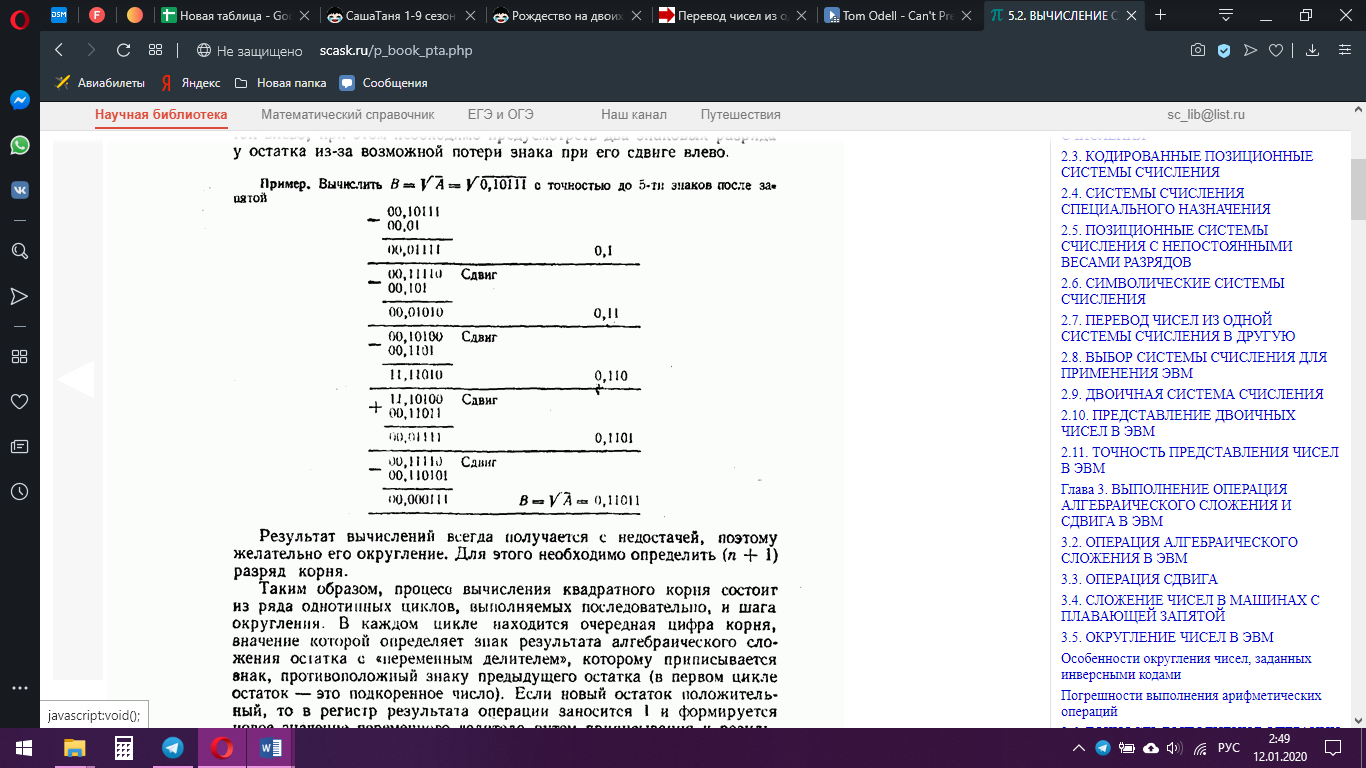
**Степень представлена в двоичном коде**

**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.



**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. Для ускорения сходимости полинома аргумент предварительно сводится в более узкую область.

* 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
* Использование цепных дробей
* Итерационные (рекуррентные)

**22.Денормализованные числа. Подводные камни в арифметике с плавающей запятой**

Денормализованное число - вид чисел с плавающей точкой, определенный в стандарте IEEE 754. При записи в форматах float, double, long double их экспонента будет записана как 0. 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. Так, 12,5 будет округлено до 12, а 13,5 – до 14. Самая опасная операция с точки зрения округления в арифметике с плавающей запятой — это вычитание. При вычитании близких чисел значимые разряды могут потеряться. Для многих широко распространенных математических формул математики разработали специальную форму, которая позволяет значительно уменьшить погрешность при округлении. Например, расчет формулы «x2-y2» лучше вычислять используя формулу «(x-y)(x+y)».
* **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». Абсолютная погрешность в отдельном случае может и не высока, но если использовать такую константу в цикле, можем получить накопленную погрешность.
* **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. Но если мы учтем ±0 и NaN, эти операции никак не эквивалентны: x y x < y? x: y x <= y? x: y x > y? y: x x >= y? y: x; +0 -0 -0 +0 +0 -0; NaN 1 1 1 NaN NaN
* **Сравнение чисел**Лучший, но все равно ошибочный способ, это сравнивать разницу с допустимой абсолютной погрешностью. Недостаток такого подхода в том, что погрешность представления числа увеличивается с ростом самого этого числа. Так, если программа ожидает «10000», то приведенное равенство не будет выполняться для ближайшего соседнего числа (10000,000977). Это особенно актуально, если в программе имеется преобразование из одинарной точности в двойную. Выбрать правильную процедуру сравнения сложно.

**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 качестве 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, Максимальное 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**

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 иF2 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.

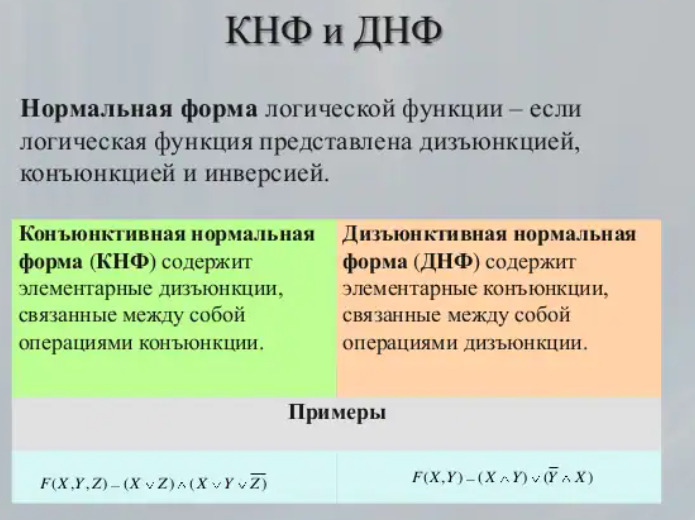
**26. Нормальная и совершенные нормальные логических функций**

* Дизъюнкция любого числа элементарных конъюнкций называется дизъюнктивной нормальной формой (ДНФ) Например: а+б\*с+(не а)б\*с+а\*(не б \* не с)
* Конъюнкция любого числа элементарных дизъюнкций называется конъюнктивной нормальной формой (КНФ). Например: ф(ф+б)(не б+с)(не а+ б+ не с)

Для каждой функции может существовать несколько дизъюнктивных и конъюнктивных нормальных форм (являющиеся равносильными друг другу)

Существует один вид дизъюнктивной нормальной формы и один вид конъюнктивной нормальной формы, в которых функция может быть записана только единственным образом. Они называются совершенными нормальными формами. В совершенной дизъюнктивной (конъюнктивной) нормальной форме: каждая элементарная конъюнкция (дизъюнкция) включает все переменные (с инверсиями или без них)

СДНФ – дизъюнкция конституентов единицы тех наборов значений переменных, где данная функция равна единице. СКНФ – конъюнкция конституентов нуля тех наборов, где данная функция равна нулю.



**27.Минимизация булевых функций. Основные понятия. Наиболее известные методы минимизации. Минимизация системы логических функций. Минимизация частично определенных функций.**

Задача минимизации булевых функций: найти аналитическое выражение заданной булевой функции в форме, содержащей минимально возможное число букв.

* Элементарной конъюнкцией называется конъюнкция конечного числа различных между собой булевых переменных, каждая из которых может иметь или не иметь отрицания
* ДНФ называется дизъюнкция элементарных конъюнкций
* Минимальной ДНФ булевой функции называется 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 происходит по следующему алгоритму:**

* + 1. Построить полное множество А элементарных конъюнкций минимизируемой системы функций, считая что вначале каждая из функций системы представлена в СДНФ. Каждой единице констнтуента множества присвоить знак, содержащий номера функций системы, в которые входит рассматриваемая констнтуета
    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 различных номеров функций содержит ее знак

**Минимизация частично определённых функций происходит по алгоритму:**

1. Нахождение любым способом сокращенную ДНФ/КНФ функции, получающейся до определением 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 определенным единичным наборам

**28.Минимизация логических выражений методом Квайна.**

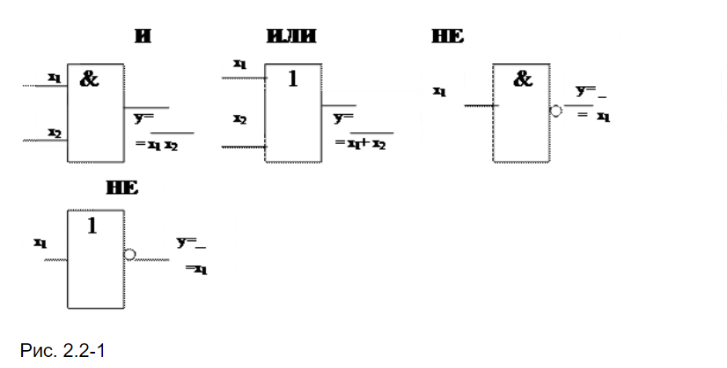
В качестве исходной формы представления логического выражения используется СДНФ. Метод Квайна выполняется в два этапа.

* *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*.
* *Второй этап* имеет своей целью устранение из тупиковой формы всех избыточных простых импликант, что дает в результате минимальное логическое выражение.

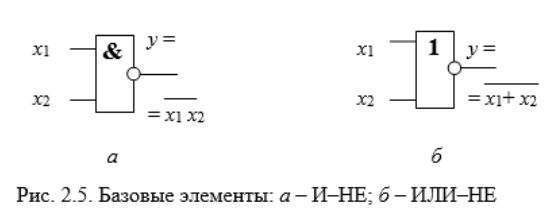
**29.Минимизация логических выражений с использованием Карт Карно (диаграммами Вейча).**Минимизация этим методом предполагает использование специальных форм – диаграмм Вейча (или карт *Карно*). Карта Карно для «*n*» логических переменных представляет собой множество квадратов (клеток), объединенных в близкую к квадрату прямоугольную форму.Записываемая функция должна быть представлена в СДНФ. Запись функции в карту осуществляется за счет установки «1» в клетки карты. 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 2DR, где DR –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**.З**апись минимального выражения по заданной функции имеет вид дизъюнкции простых конъюнкций, соответствующих контурам на карте, и формируется следующим образом:-конъюнкция, соответствующая контуру, должна включать только те переменные, которые имеют постоянное значение во всех клетках, охваченных рассматриваемым контуром,- или по другому: в конъюнкцию, соответствующую контуру, не должны входить переменные, которые имеют разные значения для клеток, охваченных рассматриваемым контуром.

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

операция склеивания для дизъюнкций

* *Операции с отрицаниями.*

двойное отрицание равносильно отсутствию отрицания

* *Операции с константами.*
* *Операции с одинаковыми операндами.*

**32.Параллелизм применения  
Пространственный параллелизм:** Бен просит Алису Хакер помочь ему.У нееесть собственная печь и противень. **Временной параллелизм:** Бену дали второй противень.Как только он ставитодин противень в печь, он начинает сворачивать печенье на другом противне, а не ожидает окончания выпекания печенья на первом противне.Использование параллелизма ограничивается *взаимозависимостями* реальных задач. Если текущая задача зависит от результатов предыдущей задачи, а не только от своих предыдущих шагов, то выполнение задачи не может быть начато до завершения предыдущей задачи. Например, если Бен Битдидл хочет проверить, достаточно ли вкусны печенья из первого противня, перед приготовлением второго, он имеет взаимозависимость, которая препятствует использованию конвейера или параллелизму. Параллелизм – один из самых важных методов проектирования высокопроизводительных цифровых систем.

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.**

Классическая ЭВМ состоит из трех основных устройств: арифметико-логического устройства, устройства управления и запоминающего устройства.

АЛУ **(Арифметико-логическое устройство)** 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.

Для передачи информации между отдельными узлами используются шины Ш1 – Ш3. Шина Ш3 обеспечивает также связь с запоминающими устройствам ЗУ (ЭВМ).

**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. Как правило, часть кэш-памяти располагается непосредственно на кристалле микропроцессора (внутренний кэш), а часть – вне его (внешняя кэш-память). Кэш-память программно недоступна. Для обращения к ней используются аппаратные средства процессора и компьютера.

**Внешняя память** организуется, как правило, на магнитных и оптических дисках, магнитных лентах. Емкость дисковой памяти достигает тысяч гигабайт при времени обращения менее 1 мкс. Магнитные ленты вследствие своего малого быстродействия и большой емкости используются в настоящее время в основном только как устройства резервного копирования данных, обращение к которым происходит редко, а может быть и никогда. Время обращения для них может достигать нескольких десятков секунд.

**35.Код Грея**

**Код Грея** — [двоичный код](https://ru.wikipedia.org/wiki/%D0%94%D0%B2%D0%BE%D0%B8%D1%87%D0%BD%D1%8B%D0%B9_%D0%BA%D0%BE%D0%B4), в котором две «соседние» кодовые комбинации различаются только цифрой в одном двоичном разряде. Иными словами, [расстояние Хэмминга](https://ru.wikipedia.org/wiki/%D0%A0%D0%B0%D1%81%D1%81%D1%82%D0%BE%D1%8F%D0%BD%D0%B8%D0%B5_%D0%A5%D1%8D%D0%BC%D0%BC%D0%B8%D0%BD%D0%B3%D0%B0) между соседними кодовыми комбинациями равно 1. Используется для упрощения выявления и [исправления ошибок](https://ru.wikipedia.org/wiki/%D0%9E%D0%B1%D0%BD%D0%B0%D1%80%D1%83%D0%B6%D0%B5%D0%BD%D0%B8%D0%B5_%D0%B8_%D0%B8%D1%81%D0%BF%D1%80%D0%B0%D0%B2%D0%BB%D0%B5%D0%BD%D0%B8%D0%B5_%D0%BE%D1%88%D0%B8%D0%B1%D0%BE%D0%BA) в системах связи, а также в формировании сигналов обратной связи в системах управления.

Код Грея используется в передаче меняющихся цифровых сигналов в отсутствие [тактового сигнала синхронизации](https://ru.wikipedia.org/wiki/%D0%A2%D0%B0%D0%BA%D1%82%D0%BE%D0%B2%D1%8B%D0%B9_%D1%81%D0%B8%D0%B3%D0%BD%D0%B0%D0%BB). Например, код (обычный двоичный) перескакивает 3→4, или 0112 → 1002. Если из-за несовершенства считывателя мы прочитаем первый бит от 011, а остальные два — от 100, мы получим 0002=0 — число, далёкое от реальных значений. В коде Грея никаких посторонних значений не будет: перескок будет в одном разряде, 010G → 110G, и мы считаем либо старое 010G=3, либо новое 110G=4.

**36.Обратная польская запись**

**Обратная польская запись**— форма записи [математических](https://ru.wikipedia.org/wiki/%D0%9C%D0%B0%D1%82%D0%B5%D0%BC%D0%B0%D1%82%D0%B8%D1%87%D0%B5%D1%81%D0%BA%D0%BE%D0%B5_%D0%B2%D1%8B%D1%80%D0%B0%D0%B6%D0%B5%D0%BD%D0%B8%D0%B5) и [логических](https://ru.wikipedia.org/wiki/%D0%9B%D0%BE%D0%B3%D0%B8%D1%87%D0%B5%D1%81%D0%BA%D0%BE%D0%B5_%D0%B2%D1%8B%D1%80%D0%B0%D0%B6%D0%B5%D0%BD%D0%B8%D0%B5) выражений, в которой [операнды](https://ru.wikipedia.org/wiki/%D0%9E%D0%BF%D0%B5%D1%80%D0%B0%D0%BD%D0%B4) расположены перед знаками [операций](https://ru.wikipedia.org/wiki/%D0%9E%D0%BF%D0%B5%D1%80%D0%B0%D1%86%D0%B8%D1%8F_(%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%D0%B5)).

Пример вычисления выражений: (1+2)\*4+3 в ОПН может быть записано так: 1 2 + 4 × 3 +

**37. АЦП и ЦАП**

**Цифро-аналоговый преобразователь**— устройство для преобразования цифрового кода в [аналоговый сигнал](https://ru.wikipedia.org/wiki/%D0%90%D0%BD%D0%B0%D0%BB%D0%BE%D0%B3%D0%BE%D0%B2%D1%8B%D0%B9_%D1%81%D0%B8%D0%B3%D0%BD%D0%B0%D0%BB). Цифро-аналоговые преобразователи являются [интерфейсом](https://ru.wikipedia.org/wiki/%D0%98%D0%BD%D1%82%D0%B5%D1%80%D1%84%D0%B5%D0%B9%D1%81) между дискретным цифровым миром и аналоговыми сигналами.

**Аналого-цифровой преобразователь** — устройство, преобразующее входной [аналоговый сигнал](https://ru.wikipedia.org/wiki/%D0%90%D0%BD%D0%B0%D0%BB%D0%BE%D0%B3%D0%BE%D0%B2%D1%8B%D0%B9_%D1%81%D0%B8%D0%B3%D0%BD%D0%B0%D0%BB) в дискретный код ([цифровой сигнал](https://ru.wikipedia.org/wiki/%D0%A6%D0%B8%D1%84%D1%80%D0%BE%D0%B2%D0%BE%D0%B9_%D1%81%D0%B8%D0%B3%D0%BD%D0%B0%D0%BB)).

АЦП преобразует [напряжение](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%BE%D0%B5_%D0%BD%D0%B0%D0%BF%D1%80%D1%8F%D0%B6%D0%B5%D0%BD%D0%B8%D0%B5) в двоичный цифровой код, а ЦАП наоборот.

**38. Корректирующие коды. Код Хэмминга. Область применения.**

Код Хэмминга —первый самоконтролирующийся и самокорректирующийся (Коды, в которых возможно автоматическое исправление ошибок) код. Построен применительно к двоичной системе счисления. Позволяет исправлять одиночную ошибку и находить двойную.

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.  
**Характеристики самокорректирующихся кодов являются:**

1. Число разрешенных и запрещенных комбинаций.

2. Избыточность кода. Величину k (число информационных символов)/n (число информационных символов) называют избыточностью корректирующего кода.

3. Минимальное кодовое расстояние (минимальное число искаженных символов, необходимое для перехода одной разрешенной комбинации в другую)

4. Число обнаруживаемых и исправляемых ошибок.

5. Корректирующие возможности кодов. • Граница Плоткина даёт верхнюю границу кодового расстояния • Есть еще Граница Хемминга устанавливает максимально возможное число разрешенных кодовых комбинаций и Граница Варшамова — Гилберта для больших n определяет нижнюю границу числа проверочных символов.

Построение кодов Хэмминга основано на принципе проверки на четность числа единичных символов: к последовательности добавляется такой элемент, чтобы число единичных символов в получившейся последовательности было четным.

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

**Язык описания аппаратуры** — специализированный [компьютерный язык](https://ru.wikipedia.org/wiki/%D0%9A%D0%BE%D0%BC%D0%BF%D1%8C%D1%8E%D1%82%D0%B5%D1%80%D0%BD%D1%8B%D0%B9_%D1%8F%D0%B7%D1%8B%D0%BA), 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 логических](https://ru.wikipedia.org/wiki/%D0%9B%D0%BE%D0%B3%D0%B8%D1%87%D0%B5%D1%81%D0%BA%D0%B8%D0%B9_%D0%B2%D0%B5%D0%BD%D1%82%D0%B8%D0%BB%D1%8C) 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)) Важнейшим отличием является явное включение концепции времени в языки описания аппаратуры.

FPGA — программируемая логическая интегральная схема (ПЛИС), конфигурация которой может быть загружена после включения питания.

**Микросхема ПЛИС состоит из:**

* конфигурируемых логических блоков, реализующих требуемую логическую функцию;
* программируемых электронных связей между конфигурируемыми логическими блоками;
* программируемых блоков ввода/вывода, обеспечивающих связь внешнего вывода микросхемы с внутренней логикой.

**40.Сумматор. Многоразрядный сумматор**

**Сумматор** – это логическая электронная схема, выполняющая сложение двоичных чисел.

Одноразрядный двоичный сумматор обеспечивает сложение одноименных разрядов операндов с учетом переноса, поступающего от ближайшего младшего разряда. Сумматор вырабатывает значение соответствующего разряда суммы (S) и перенос (P), который должен быть учтен в соседнем старшем разряде. Синтез схемы, реализующей функции одноразрядного сумматора, можно выполнить на основании таблицы истинности. Исходя из реализуемой функции сумматор представляет собой логический узел с двумя выходами (выход суммы S и выход переноса Р) и тремя входами: а - разряд первого операнда; b - разряд второго операнда; р - перенос из младшего разряда.

Многоразрядный двоичный сумматор строится на основе одноразрядных сумматоров с введением соответствующих связей между разрядами. Сумматор обладает малым быстродействием из-за последовательного учета переноса, возникшего в младшем разряде, в непрерывной цепочке старших разрядов, имеющих значение поразрядной суммы, равное единице. Такие разряды называются разрядами, пропускающими перенос. В худшем случае перенос, возникший в младшем разряде, распространяется до самого старшего разряда формируемой суммы.

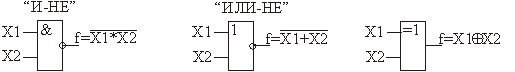
**41. Полная система логических функций.**

Функционально полная система логических функций представляет собой набор логических функций, с помощью которых можно записать любую, сколь угодно сложную функцию. В этом случае говорят, что этот набор образует базис. Функционально полными являются 3 базиса:

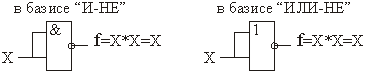
1) "И-ИЛИ-НЕ" (базис конъюнкции, дизъюнкции, инверсии)

2) "И-НЕ" (базис Шеффера)

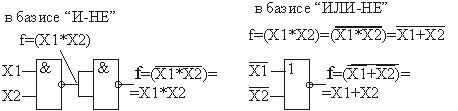
3) "ИЛИ-НЕ" (базис Пирса или функция Вебба).



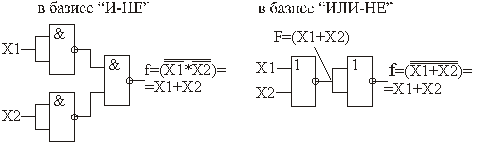
Примеры реализации логических операций в базисах “И-НЕ” и “ИЛИ-НЕ”.



**Реализация операции “НЕ”:**



**Реализация операции “И”:**



**Реализация операции “ИЛИ”:**

Пример реализации комбинационного устройства в базисе "И-НЕ". Пусть задана функция, реализуемая комбинационным устройством, в аналитической форме

Используя закон де Моргана и с учетом закона двойного инвертирования, запишем эту функцию в виде



Как следует из полученного аналитического выражения, логическое устройство должно содержать три двухвходовых и один трехвходовой элемент И-НЕ. Функциональная схема комбинационного устройства, построенная в базисе И-НЕ

**42. Искусство управления сложностью. Цифровая абстракция.**

**ИСКУССТВО УПРАВЛЕНИЯ СЛОЖНОСТЬЮ**

Важен систематический подход к управлению сложностью многоуровневой системы. Современные цифровые системы построены из миллионов и миллиардов транзисторов. Человеческий мозг не в состоянии предсказать поведение подобных систем, поэтому чтобы разработать удачный микропроцессор, необходимо научиться управлять сложностью разрабатываемой системы.

**Абстракция** – принцип управления сложностью системы, подразумевающий исключение из рассмотрения тех элементов, которые в данном конкретном случае несущественны для понимания работы этой системы.

Именно принцип абстрагирования от маловажных деталей позволяет вашей бабушке общаться с внуками в Интернете, не задумываясь о квантовых колебаниях электронов или организации памяти компьютера.

**Три базовых принципа** для управления сложностью системы: **иерархичность, модульность конструкции и регулярность**. Эти принципы применительны как к ПО, так и к аппаратной части компьютерных систем.

**Иерархичность** – принцип иерархичности предполагает разделение системы на отдельные модули, а затем последующее разделение каждого такого модуля на фрагменты до уровня, позволяющего легко понять поведение каждого конкретного фрагмента.

**Модульность** – принцип модульности требует, чтобы каждый модуль в системе имел четко определенную функциональность и набор интерфейсов и мог быть легко и без непредвиденных побочных эффектов соединен с другими модулями системы.

**Регулярность** – принцип регулярности требует соблюдения единообразия при проектировании отдельных модулей системы. Стандартные модули общего назначения, например, такие как блоки питания, могут использоваться многократно, во много раз снижая количество модулей, необходимых для разработки новой системы.

**ЦИФРОВАЯ АБСТРАКЦИЯ**

Большинство физических величин изменяется непрерывно. Цифровые системы, представляют информацию в виде дискретно меняющихся переменных с конечным числом строго определённых значений.

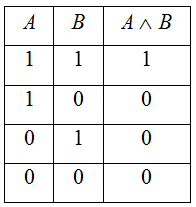
Джордж Буль разработал систему логики, использующую двоичные переменные, и эту систему сегодня называют его именем – Булева логика. Булевы переменные могут принимать значения ИСТИНА (TRUE)(1) или ЛОЖЬ (FALSE)(0).

**Преимущества цифровой абстракции** заключаются в том, что разработчик цифровой системы может сосредоточиться исключительно на единицах и нулях, игнорируя, каким образом булевы переменные представлены на физическом уровне. Программист может продуктивно работать, не располагая детальной информацией об аппаратном обеспечении компьютера. Однако, понимание того, как работает это аппаратное обеспечение, позволяет программисту гораздо лучше оптимизировать программу для конкретного компьютера.

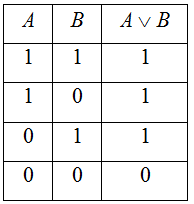
**43. Логические элементы. Таблицы истинности. Обозначения элементов в разных представления.**

**Таблицы истинности для основных двоичных логических функций**

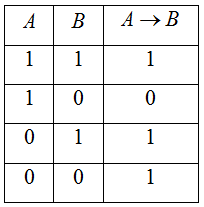
1. **Конъюнкция** (логическое умножение) – сложное логическое выражение, которое является истинным только в том случае, когда истинны оба входящих в него простых выражения



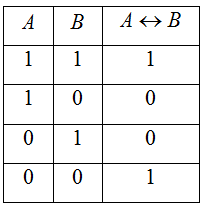
2. **Дизъюнкция** (логическое сложение) – логическая операция, по своему применению максимально приближённая к союзу «или» в смысле «или то, или это, или оба сразу».



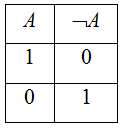
3. **Импликация** (логическое следствие) – это сложное логическое выражение, которое является ложным тогда и только тогда, когда условие истинно, а следствие ложно.



4. **Эквиваленция** – это сложное логическое высказывание, которое является истинным только при одинаковых значениях истинности простых выражений, входящих в него.



5. **Логическое** отрицание (инверсия) делает истинное высказывание ложным и, наоборот, ложное – истинным.



**44. За пределами цифровой абстракции. Напряжение питания. Логические уровни. Допускаемые уровни шумов.**

Цифровая система оперирует дискретными переменными(0, 1). Однако для представления этих переменных используются непрерывные физические величины, такие как напряжение в электрической цепи. Задача разработчика цифровой системы – определить, каким образом непрерывно-меняющаяся величина соотносится с конкретным значением дискретной переменной.

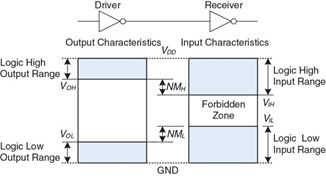
**Напряжение питания**

Предположим, что минимальное напряжение в электронной цифровой системе, напряжение земли (GND), составляет 0 В. Самое высокое напряжение в системе поступает от блока питания и, как правило, обозначается VDD.

Транзисторные технологии семидесятых и восьмидесятых годов прошлого века в основном использовали VDD равное 5 В. С переходом на транзисторы меньшего размера, VDD последовательно снижали до 3,3 В, 2,5 В, 1,8 В, 1,5 В, 1,2 В и даже ниже для экономии электроэнергии и во избежание перегрузки транзисторов.

**Логические уровни**

Отображение непрерывно-меняющейся переменной на различные значения дискретной двоичной переменной выполняется путем определения логических уровней, как показано на Рис.

Первый логический элемент в рассматриваемой схеме называется источник (driver), а второй – приемник (receiver). Выходной сигнал источника подключается ко входу приемника. Источник выдает выходной сигнал низкого напряжения (0) в диапазоне от 0 В до VOL или выходной сигнал высокого напряжения (1) 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 (forbidden zone) между VIL и VIH, то поведение этого логического элемента становится непредсказуемым. VOH и VOL называются соответственно высоким и низким логическими уровнями выхода (output high and low logic levels), а VIH и VIL называются соответственно высоким и низким логическими уровнями входа (input high and low logic levels).

**Допускаемые Уровни Шумов**

Для того чтобы выходной сигнал источника был правильно интерпретирован на входе приемника, необходимо, чтобы VOL < VIL и VOH > VIH. В этом случае, даже если выходной сигнал источника будет загрязнен шумами, приемник по-прежнему сможет правильно определить логический уровень входного сигнала. **Допускаемый уровень шумов** (noise margin) – это то максимальное количество шума, присутствие которого в выходном сигнале источника не мешает приемнику корректно интерпретировать значение полученного сигнала. Значения нижнего допускаемого уровня шумов (low noise margin) и верхнего допускаемого уровня шумов (high noise margin) определяются следующим образом:

NML = VIL – VOL и NMH = VOL – VIH

**45. Передаточная характеристика. Статическая дисциплина**

**Передаточная характеристика** *(DС transfer characteristics)* какого-либологического 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.

В случае идеального инвертора переключение будет резким в точке *VDD*/2,как показано на**Рис(a)**.Для *V*(*A*) > *VDD*/2, *V*(*Y*) = 0.В этом случае, *VIH* = *VIL* = *VDD*/2. *VOH* = *VDD* и *VOL* = 0.

*DC* указывает на состояние,когда напряжение на входе электроннойсистемы поддерживается постоянным или изменяется так медленно, что остальные параметры системы плавно изменяются вместе с ним.

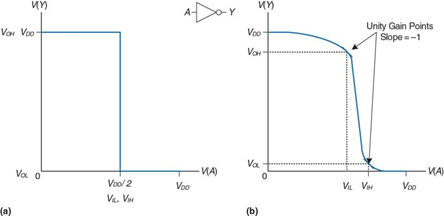
Напряжение при переключении реального инвертора изменяется постепенно между граничными значениями – так, как показано на **Рис(b)**.Если входное напряжение*V*(*A*)равно0,то напряжение на

выходе *V*(*Y*) = *VDD*. Если *V*(*A*) = *VDD*, то *V*(*Y*) = 0. Однако, переход между этими конечными точками плавный и может находиться правее или левее значения *VDD*/2.

Как в этом случае определить логические уровни. Разумно выбрать в качестве логических уровней те две точки, где наклон передаточной характеристики d*V*(*Y*)/d*V*(*A*) равен −1. Такие точки называются *граничные коэффициенты передачи* *(unity gain points)*. Подобный выбор обычно максимизирует допускаемые уровни шумов. При уменьшении *VIL* *VOH* увеличивается незначительно. Однако, если *VIL* растет, *VOH* падает практически отвесно.

**Статическая Дисциплина**

Для того, чтобы избежать попадания входных сигналов в запретные зоны, логические вентили должны разрабатываться в соответствии с **принципом статической дисциплины** *(static discipline)*.Принципстатической дисциплины требует, чтобы при условии наличия логически корректных сигналов на входе каждый элемент системы выдавал логически корректные сигналы на выходе.



**Передаточные характеристики и уровни шума**

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.

Выбор *VDD* и логических уровней может быть произвольным, однако этот выбор должен обеспечить совместимость всех логических вентилей, обменивающихся данными в пределах одной цифровой системы. Поэтому вентили обычно группируются в *семейства логики* *(logic families)* таким образом,что любой элемент из одного семействапри соединении с любым другим элементом из этого же семейства автоматически обеспечивает соблюдение принципа статической дисциплины.

**46.Биполярные и КМОП транзисторы. Полупроводники. Конденсаторы. n-МОП и p-МОП-транзисто**

В технологии КМОП используется «полевой транзистор» управляется он полем, электрическим полем затвора, в то время как [биполярный транзистор управляется током базы](http://electrik.info/main/praktika/1381-bipolyarnye-tranzistory-shemy-rezhimy-modelirovanie.html).

МОП-транзисторы изготавливаются из кремния.

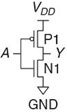
Конденсатор состоит из двух проводников, отделенных друг от друга изолятором. Если к одному из проводников приложить напряжение, то через некоторое время этот проводник накопит электрический заряд, а другой проводник накопит противоположный электрический заряд −Q. Емкость – это очень важный параметр электрической схемы, поскольку зарядка или разрядка любого проводника требует времени и энергии. Более высокая емкость означает, что электрическая схема будет работать медленнее и потребует для своего функционирования больше энергии.

n-МОП и p-МОП-транзисторы

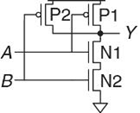
Полевой МОП-транзистор ведет себя как переключатель, управляемый приложенным к нему напряжением. Он представляет собой «сэндвич» из нескольких слоёв проводящих и изолирующих материалов. Существуют два вида полевых МОП-транзисторов: n-МОП и p-МОП. В транзисторах n-типа, области, где расположены полупроводниковые примеси n-типа – в свою очередь называемые истоком и стоком – находятся рядом с затвором, причем вся эта структура размещается на подложке p-типа. В транзисторах же p-МОП и исток, и сток – это области p-типа, размещенные на подложке n-типа.

**47. Логический вентиль НЕ и другие на КМОП-транзисторах. Псевдо n-МОП-Логика Потребляемая мощность**

Схема на **Рис.** 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*.

**Схема вентиля НЕ**

n-МОП-транзистор N1 включен между землей GND и выходным контактом *Y*. В свою очередь, p-МОП-транзистор P1 включен между напряжением питания *VDD* и выходным контактом *Y*. Напряжение на входном контакте *А* управляет переходами обоих транзисторов. Если напряжение на *А* равно 0, то транзистор N1 выключен, транзистор P1 включен. При этом, напряжение на контакте Y равно напряжению питания *VDD*, а не земли, что соответствует логической единице. В этом случае говорят, что Y «подтянут» к единице.Включенный транзисторP1хорошо передает логическуюединицу (равную напряжению питания), то есть напряжение на контакте *Y* очень близко к *VDD*.Если же напряжение на контакте А равнологической единице, то транзистор N1 включен, а транзистор P1 выключен, и напряжение на контакте *Y* равно напряжению земли, что соответствует логическому нулю. В этом случае говорят, что *Y* «подтянут» к нулю. Включенный транзистор N1 хорошо передает логический ноль, то есть напряжение на контакте *Y* очень близко к GND. Проверка в таблице истинности подтверждает, что мы действительно имеем дело с логическим вентилем НЕ.

**Другие логические вентили на КМОП-транзисторах**

На **Рис.** показана схема для построения с помощью МОП-транзисторов логического элемента И-НЕ с двумя входными контактами. На электронных схемах принято, что если нет никаких дополнительных замечаний или обозначений, то подразумевается, что две линии соединяются друг с другом в том случае, если одна из линий заканчивается в точке пересечения (пересечение в форме буквы Т). Если же обе линии продолжаются точкой пересечения, то для обозначения контакта этих двух линий в точке пересечения ставится точка. Если точка отсутствует, то это означает, что линии не пересекаются, и одна из линий проходит над другой. На **Рис.** n-МОП-транзисторы N1 и N2 соединены последовательно. Причем, чтобы замкнуть выходной контакт на землю GND – то есть понизить логический уровень, оба этих транзистора должны быть включены. В то время как p-МОП-транзисторы P1 и P2 соединены параллельно, и только один из них должен быть включен, чтобы соединить выходной контакт с напряжением питания *VDD* – то есть повысить логический уровень.

**Псевдо n-МОП-Логика**

Построенный по технологии КМОП логический вентиль ИЛИ-НЕ, число входных контактов которого равно N, использует N параллельно включенных n-МОП-транзисторов и N последовательно включенных p-МОП-транзисторов. Последовательно включенные транзисторы передают сигнал медленнее, чем транзисторы, включенные параллельно. Кроме того, p-МОП-транзисторы передают сигналы медленнее, чем n-МОП-транзисторы, поскольку дырки не могут перемещаться по кристаллической решетке кремния так же быстро, как электроны. В результате, соединенные параллельно n-МОП-транзисторы работают быстро, а соединенные последовательно p-МОП-транзисторы работают медленно, особенно если их много.

**Потребляемая мощность**–это количество энергии,потребляемойсистемой в единицу времени.

Цифровая система потребляет энергию как в динамическом режиме, когда выполняет какие-либо операции, так и в статическом, когда система находится в состоянии покоя *(idle)*. В динамическом режиме энергия расходуется на зарядку емкостей элементов системы, когда эти элементы переключаются между 0 и 1. И хотя в статическом режиме никаких переключений не происходит, система все равно расходует электрическую энергию.

**48.Проектирование комбинационной логики. От логики к логическим элементам, Что такое Х и Z: способы сопряжения микросхем в ЭВМ.**

Принципиальная схема – это изображение цифровой схемы, показывающее элементы и соединяющие их проводники. Изображая принципиальные схемы в унифицированном виде, нам становится легче читать их и отлаживать. В большинстве случаев мы будем придерживаться следующих правил:

* Входы изображаются на левой (или верхней) части схемы;
* Выходы изображаются на правой (или нижней) части схемы;
* Всегда, когда это возможно, элементы необходимо изображать слева направо;
* Проводники лучше изображать прямыми линиями
* Проводники всегда должны соединяться в виде буквы «Т»;
* Точка в месте пересечения проводников обозначает их соединение;

Символ «X» используется не только для обозначения переменных, но и для обозначения недопустимых состояний сигналов при симуляции логических схем. Булева алгебра ограничена значениями 0 и 1. Однако реальные схемы могут также иметь недопустимое и плавающее состояния, представляемые символами X (обозначает неизвестное логическое значение или недопустимое значение физического напряжения в соединении, не соответствующее уровням логических 0 и 1. Это обычно происходит, если к соединению подключены выходы других элементов схемы, выдающие значения 0 и 1 одновременно) и 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. Типично неправильное представление – это что неподключенная, или плавающая цепь имеет значение логического) соответственно.

**49.Временные характеристики цифровых микросхем. Задержка распространения и задержка реакции. Импульсные помехи.**

* Задержка распространения tpd – это максимальное время от начала изменения входа до момента, когда все выходы достигнут установившихся значений.(сумма задержек распространения всех элементов в критическом пути)
* Задержка реакции tcd – это минимальное время от момента, когда вход изменился, до момента, когда любой из выходов начнет изменять свое значение (сумма задержек реакции всех элементов в кратчайшем пути)

Временные характеристики мультикомплексора:

* Критических путь
* Задержки распространения и реакции
* Импульсные помехи (явление одиночного изменения на входе вызывает несколько выходных изменений)
* Резюме (Цифровая схема – это модуль с дискретными значениями входов и выходов и спецификацией, описывающей его функциональные и временные характеристики. Функциональное описание комбинационной схемы может быть задано таблицей истинности или логическим выражением. Логическое выражение для любой таблицы истинности может быть получено в виде совершенной дизъюнктивной нормальной формы или совершенной конъюнктивной нормальной формы)

**50.Базовые комбинационные блоки. Мультиплексоры. Логика на мультиплексорах. Дешифраторы**

**Мультиплексор** - логическое устройство, предназначенное для **поочерёдной** передачи на один выход одного из нескольких входных сигналов, т.е. их мультиплексирование. **Количество мультиплексируемых входов** от 2 до 16. **Количество выходов** от 1 до 4. **Управление** осуществляется при помощи входного кода. Связь между **n**(кол-во каналов) и **m**(число разрядов) **n = 2^m**. **Функции** записываются как **MS**.**Разряды 1,2,4** и т.д. Входная информация 0,1,2,3,4… **Выходы** бывают **прямыми и инверсными**. Применяют для операций **коммутации, преобразование параллельного кода в логический**

**Дешифратор -** логическая комбинационная схема,имеющая n входов и 2^n выходов.Если имеется комбинация входных сигналов,то появляется сигнал на том контакте, который соответствует коду,который был на входе. Особенностью двоичного дешифратора является то, что логический сигнал появляется только на выходе, соответствующем номеру двоичной комбинации. Существует несколько разновидностей дешифраторов: прямоугольные, матричные, пирамидальные.

**51.Проектирование последовательности логики. Защелки и триггеры. RS-триггер. D-защелка. D-Триггер. Регистр.**

* Защелка – элемент, который запоминает свое состояние.
* Триггер – электронная схема обладающая двумя устойчивыми состояниями. Изменение состояния происходит скачкообразно под воздействием управляющих сигналов

Триггеры служат основой для построения регистров , счетчиков и других элементов обладающих функцией хранения. Главной частью триггера является запоминающая ячейка

Их разделяют по способу управления (RS, JK. D…), типу синхронизации (синхронные (статическая/динамическая) синхронизация), асинхронные), по внутренней организации (1-/2- ступенчатые). Триггер называется синхронным, если таблица переходов хотя бы по одному управляющему входу осуществляется под воздействием синхронизирующего сигнала

Асинхронные RS(по названию входов Reset (установить выход в 0) и Set (установить в 1)) триггеры одни из самых простых и часто являются основой для построения более сложных. Триггеры этого типа построены на двух типах логических элементов: ИЛИ-НЕ – триггер с прямым входом, И-НЕ – триггер с прямым входом

D-триггер (от слова delay) – триггер с одним входом, работающий так, что сигнал на выходе после переключения равен сигналу на входе D до переключения.

D-защелка – D триггер, работающий по сигналу на входе C

Регистр – устройство для записи хранения считывания и выполнения операций над n-разрядными двоичными данными. Фактически любое цифровое устройство можно представить как совокупность регистров объединенных при помощи комбинационных логических устройств, а сам регистр как упорядоченный набор триггеров

**52.Триггер с функцией разрешения. Триггер с функцией сброса. Проектирование синхронных логических схем. Синхронные последовательные схемы. Синхронные и асинхронные схемы.**

У триггеров с функцией разрешения имеется еще один вход, называемый EN (ENABLE т.е. разрешить). Этот вход определяет, будут ли данные загружаться по фронту. Когда на EN подается логическая 1, то такой D-триггер поведет себя как и обычный D-триггер. Если же поступает логический 0, то триггер игнорирует тактовый сигнал и сохраняет свое состояние. Такие триггеры полезны, если мы хотим загружать значения в триггер только на протяжении определенного времени.

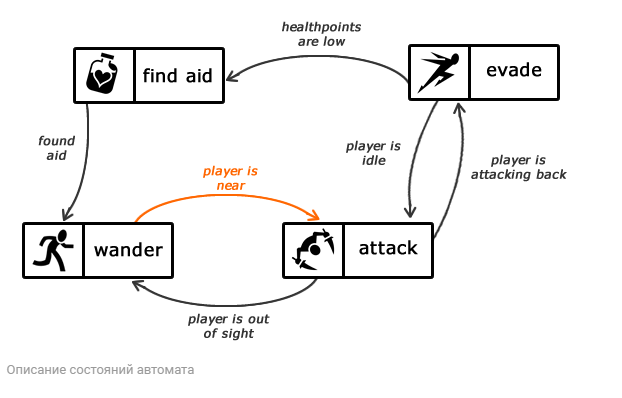
В триггере с функцией сброса добавляется вход, называемый RESET (сброс). Когда на RESET подан 0, сбрасываемый триггер едет себя как обычный D-триггер. Когда же поступает единица, триггер игнорирует вход D и сбрасывает выход к 0. Такие триггеры полезны, если мы хотим ускорить установления определенного состояния (0) во всех триггерах системы при первом включении. Такие триггеры могут сбрасываться как синхронно, так и асинхронно. В первом случае триггеры сбрасываются только по фронту сигнала CLK. Во втором сброс происходит сразу при поступлении логической единицы на вход RESET, вне зависимости от тактового сигнала.

Схема является синхронной последовательностной, если ее элементы удовлетворяют следующим условиям: как минимум один элемент схемы является регистром, остальные – комбинационной схемой, все регистры тактируются единственным тактовым сигналом, в каждом циклическом пути есть как минимум 1 регистр.

Последовательностные схемы, не являющиеся синхронными, называются асинхронными.

**53.Конечные автоматы. Пример проектирования конечного автомат**

Конечный автомат- это модель вычислений, основанная на гипотетической машине состояний. В один момент времени только одно состояние может быть активным. Следовательно, для выполнения каких-либо действий машина должна менять свое состояние. Конечные автоматы обычно используются для организации и представления потока выполнения чего-либо. Это особенно полезно при реализации ИИ в играх. Например, для написания «мозга» врага: каждое состояние представляет собой какое-то действие (напасть, уклониться и т. д.).



**54. Конечные автоматы. Кодирование состояний. Автоматы Мура и Мили.  
Конечный автомат- это модель вычислений, основанная на гипотетической машине состояний**Кодирование состояний

Одно из важных решений в кодировании состояний – выбор между двоичным кодированием (00, 01, 10) и прямым кодированием (001, 010, 100)( для каждого состоянияиспользуется один бит состояния), которое также называется кодированием «1 из *N*». При *двоичном кодировании*,как в примере сконтроллером светофора, каждому состоянию ставится в соответствие двоичное число (номер этого состояния). Так как K двоичных чисел можно записать в log2*K* разрядах, системе с *K* состояниями нужно всегоlog2*K* битовсостояния. При использовании прямого кодирования схема определения следующего состояния и схема формирования выходных сигналов часто упрощается; таким образом, требуется меньше элементов. Наилучший выбор кодирования зависит от особенностей конкретного автомата .

Автоматы Мура и Мили

**Автомат Мура** в [теории вычислений](https://ru.wikipedia.org/wiki/%D0%A2%D0%B5%D0%BE%D1%80%D0%B8%D1%8F_%D0%B2%D1%8B%D1%87%D0%B8%D1%81%D0%BB%D0%B5%D0%BD%D0%B8%D0%B9) — [конечный автомат](https://ru.wikipedia.org/wiki/%D0%9A%D0%BE%D0%BD%D0%B5%D1%87%D0%BD%D1%8B%D0%B9_%D0%B0%D0%B2%D1%82%D0%BE%D0%BC%D0%B0%D1%82), выходное значение сигнала в котором зависит лишь от текущего состояния данного автомата, и не зависит напрямую

**Автомат Мили** — [конечный автомат](https://ru.wikipedia.org/wiki/%D0%9A%D0%BE%D0%BD%D0%B5%D1%87%D0%BD%D1%8B%D0%B9_%D0%B0%D0%B2%D1%82%D0%BE%D0%BC%D0%B0%D1%82), выходная последовательность которого (в отличие от [автомата Мура](https://ru.wikipedia.org/wiki/%D0%90%D0%B2%D1%82%D0%BE%D0%BC%D0%B0%D1%82_%D0%9C%D1%83%D1%80%D0%B0)) зависит от состояния автомата и входных сигналов.

у автомата Мура обычно больше состояний, чем у автомата Мили, решающего ту же задачу.

**55.Декомпозиция конечных автоматов. Восстановление конечных автоматов по электрической схеме.**

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 декомпозициейконечных автоматов.

Восстановление конечных автоматов по электрической схеме

Восстановление конечных автоматов по электрической схеме практически является процессом, обратным проектированию КА. Этот процесс необходим, например, при рассмотрении проекта с неполной документацией или для реверсивного проектирования чьей-то системы

1. Проанализировать схему, возможные состояния входов, выходов и регистра состояний.
2. Составить выражения для следующего состояния и для выходов.
3. Составить таблицу выходов и таблицу переходов.
4. Вычеркнуть из таблицы переходов состояния, в которые система никогда не попадает.
5. Присвоить имя каждому используемому набору бит-состояний.
6. Переписать таблицы выходов и переходов, используя эти обозначения.
7. Нарисовать диаграмму переходов.
8. Описать словами то, что делает автомат

На последнем шаге развернуто описать цели и функции автомата, чтобы избежать простого переформулирования каждого перехода из диаграммы переходов.

**56. Синхронизация последовательностных схем. Временные характеристики системы. Расфазировка тактовых сигналов. Метастабильность. Синхронизаторы.**

**СИНХРОНИЗАЦИЯ ПОСЛЕДОВАТЕЛЬНОСТНЫХ СХЕМ**

Последовательностный элемент имеет апертурное время до и после фронта тактового сигнала, в течение которого его информационные входные сигналы должны быть стабильными, чтобы на выходе триггера сформировался корректный сигнал.

Часть апертурного времени последовательностного элемента до фронта тактового импульса называется временем предустановки (setup time), после фронта – временем удержания (hold time). Динамическая дисциплина позволяет использовать только те сигналы, которые изменяются вне апертурного времени. При выполнении требований динамической дисциплины мы можем оперировать дискретными единицами времени, которые называются тактовыми циклами, аналогично тому, как мы оперируем с дискретными логическими уровнями 1 и 0. Сигнал может изменяться и осциллировать в течение некоторого ограниченного промежутка времени. При выполнении требований динамической дисциплины важно лишь его значение в конце цикла тактового сигнала, когда он уже принял стабильное значение. Период тактовых импульсов должен быть достаточно большим, чтобы переходные процессы всех сигналов успели завершиться. Это требование ограничивает быстродействие всей системы. В реальных системах тактовые импульсы поступают на входы триггеров неодновременно. Этот разброс по времени, который называется **расфазировкой** или **разбросом фаз тактового сигнала** (clock skew), заставляет разработчиков дополнительно увеличивать период тактовых сигналов.

Иногда невозможно удовлетворить требованиям динамической дисциплины, особенно в устройствах сопряжения цифровой системы с реальным миром. Решением проблемы асинхронных входов является использование синхронизатора, на выходе которого некорректный логический уровень может появиться с очень малой (но не нулевой) вероятностью.

**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, and 1 гигагерц (ГГц) = 109 Гц.

|  |  |
| --- | --- |
| tpd ≤Tc−(tpcq +tsetup) |  |

Слагаемое в скобках, tpcq + tsetup, называется потерями на упорядочение (sequencing overhead). В идеальном случае весь период тактового сигнала может быть затрачен на вычисления в комбинационной логике (время tpd). Однако, потери на упорядочение в триггерах уменьшают это время. Это неравенство называется **ограничением времени предустановки** или **ограничением максимальной задержки**, поскольку оно зависит от времени предустановки и ограничивает максимальную задержку распространения в комбинационной логической схеме.

Если задержка распространения в комбинационной схеме слишком велика, то вход D2 может не успеть принять свое установившееся состояние ко времени, когда регистр R2 ожидает стабильности и фиксирует его. Таким образом, R2 может зафиксировать некорректный результат или даже логический уровень в запретной зоне. В таком случае схема будет работать некорректно. Проблему можно решить увеличением периода тактового сигнала или пересмотром комбинационной схемы с целью добиться меньшей задержки распространения.

**Ограничение времени удержания**

Регистр R2 на **Рис.** (a) имеет также ограничение времени удержания. Его вход, D2, не должен изменяться в течение некоторого времени thold после переднего фронта тактового импульса.

**Рис. Непосредственное последовательное соединение триггеров**

Тем не менее, ограничения времени удержания критически важны. Если они нарушаются, то единственным решением является увеличение задержки реакции комбинационной схемы, что требует ее перепроектирования. Такие нарушения, в отличие от нарушений ограничений времени предустановки, не могут быть исправлены изменением периода тактового сигнала. Перепроектирование интегральной микросхемы и производство ее исправленного варианта занимает несколько месяцев и требует затрат в несколько миллионов долларов при современных технологиях, поэтому к нарушениям ограничения времени удержания нужно относиться крайне серьезно.

**Заключение**

Последовательностные схемы имеют ограничения времен предустановки и удержания, которые устанавливают максимальную и минимальную задержки в комбинационной логической схеме между триггерами. Современные триггеры обычно спроектированы так, что минимальная задержка в комбинационной логике равна нулю, то есть триггеры могут быть размещены непосредственно друг за другом. Максимальная задержка ограничивает число последовательных логических элементов, включенных один за другим в критическом пути быстродействующей схемы.

**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**. Например, длина проводников, по которым тактовые сигналы поступают на разные регистры, может быть разной, это приводит к разным временам задержки.

Шум также приводит к различным задержкам. Стробирование тактовых сигналов, приводит к их дополнительной задержке. Если в схеме используются стробированные нестробированные тактовые сигналы, то между ними будет существенное рассогласование. При выполнении временного анализа мы рассматриваем наихудший случай, что позволяет гарантировать, что схема будет работать при всех условиях.

В итоге, расфазировка тактовых импульсов приводит к эффективному увеличению как времени предустановки, так и времени удержания. Это, в свою очередь, приводит к росту потерь на упорядочение и уменьшает время, доступное для обработки данных комбинационной схемой. Чтобы предотвратить такие серьезные нарушения ограничений времени удержания, проектировщик должен ограничивать расфазировку тактовых сигналов. Иногда триггеры специально проектируются медленными (время tccq велико), чтобы избежать проблем времени удержания, даже если расфазировка тактовых сигналов существенна.

**Метастабильность**

Метастабильное состояние- состояние, когда состояние информационного входа триггера изменяется в течение апертурного времени, на его выходе Q может на некоторое время появиться напряжение в диапазоне от 0 до VDD, то есть в запретной зоне. Такое состояние называется метастабильным. Со временем выход триггера перейдет в стабильное состояние 0 или 1. Однако время разрешения, необходимое для достижения стабильного состояния, не ограничивается.

Метастабильное состояние триггера подобно состоянию шарика на вершине между двумя впадинами. Положения во впадинах являются стабильными, поскольку шарик будет находиться в них неограниченно долго при отсутствии внешнего возмущения.

Положение на вершине возвышенности называется метастабильным, потому что шарик будет находиться в нем только при условии идеальной балансировки. Но, поскольку в мире нет ничего совершенного, со временем шарик скатится в одну из впадин. Необходимое для этого время зависит от степени первоначальной балансировки шарика. Каждое бистабильное устройство имеет метастабильное состояние между двумя стабильными.

**Синхронизаторы**

Синхронизатор - устройство, на вход которого поступает асинхронный сигнал D и тактовый сигнал CLK. За ограниченное время он формирует выходной сигнал Q, который с очень высокой вероятностью имеет корректный логический уровень. Если вход D стабилен в течение апертурного времени, то выход Q должен принять значение входа. Если D изменяется в течение апертурного времени, то Q может принять значение 0 или 1, но не должен быть метастабильным.

Надежность системы обычно измеряют средним временем наработки на отказ (mean time between failures, MTBF). Как понятно из названия, MTBF – это среднее время между отказами системы. Эта величина обратна вероятности сбоя системы за любую заданную секунду:

MTBF растет экспоненциально с ростом времени ожидания синхронизатора, Tc. Для большинства систем синхронизатор, который ожидает один период тактового сигнала, обеспечивает достаточную величину MTBF. В высокоскоростных системах может понадобиться ожидание на большее количество периодов тактового сигнала.

**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. При этом также скачкообразно изменяется уровень напряжения на выходе триггера.

Типы триггеров

* RS-триггеры (триггер, который сохраняет своё предыдущее состояние при неактивном состоянии обоих входов и изменяет своё состояние при подаче на один из его входов активного уровня и )
* D-триггер (запоминает состояние входа и выдаёт его на выход. D-триггеры имеют, как минимум, два 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 (чем он отличается от RS-триггеров с доопределённым состоянием, которые строго переходят в логический ноль или единицу, независимо от предыдущего состояния))

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 характеризуется неизменными уровнями выходного напряжения (выходными потенциалами): высоким — близким к напряжению питания и низким — около нуля. Статические триггеры по способу представления выходной информации часто называют потенциальными.).

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.**

**Регистры** представляют собой несколько триггеров соединённых между собой различными способами. Два основных вида регистров:

* параллельный регистр;
* последовательный регистр или регистр сдвига.

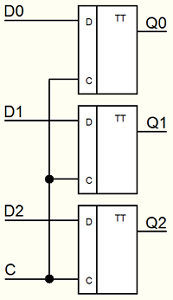
Параллельный регистр

Схема соединения триггеров в параллельном регистре

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);

Наибольшее распространение получили тактируемые параллельные регистры.

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.

Параллельные регистры, срабатывающие по фронту импульса.

Основное применение регистры с тактируемым входом нашли в схемах где необходимо хранить некоторый код в течении необходимого времени, а по фронту импульса на тактовом входе С выдавать на выходы Q.

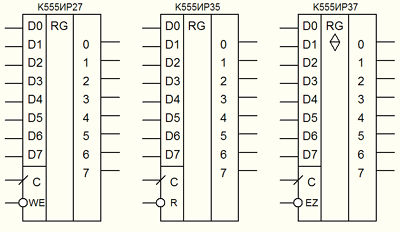
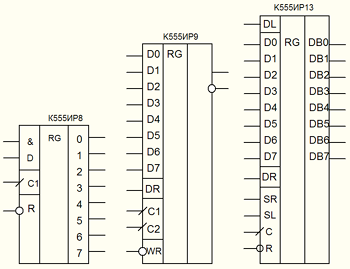
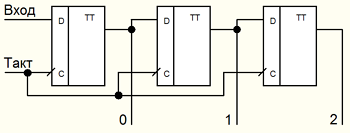
**Регистр сдвига** или **последовательный** **регистр** (англ. Shift Register), представляет собой схему, в которой внутренние триггеры соединены последовательно. Схема работы сдвигового регистра заключается в следующем: по импульсу тактового сигнала происходит сдвиг на один разряд цифрового кода, который записан на входном выводе. У обычных сдвиговых регистров, сдвиг происходит от младших разрядов к старшим, но есть также и **реверсивные сдвиговые регистры**, у которых сдвиг идет, наоборот, от старших разрядов к младшим.

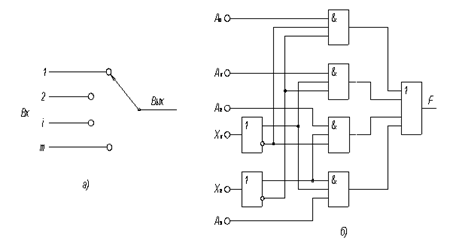
Схема сдвигового регистра от младших разрядов к старшим.

Большинство регистров сдвига имеют восемь разрядов, различаются режимами работы, режимами записи, чтения и сдвига, а также типом выходного каскада.

Регистры сдвига

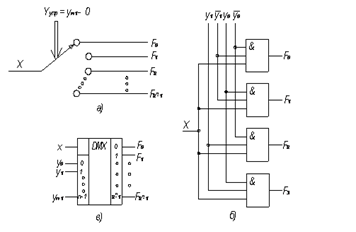


**59. Мультиплексоры и демультиплексоры. Отличия в обозначениях цифровых элементов в разных стандартах.**

**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. Функциональная схема мультиплексора приведена на рисунке б. Двоичный код, воздействующий на адресные входа, откроет одну из схем И, которая соединит с выходом соответствующую входную линию. При этом информация на выходе определяется состоянием выбранного входного канала и не зависит от состояния других каналов.

Мультиплексоры можно использовать для синтеза логических функций от нескольких переменных (x1, x2, …, xn). Если число адресных входов мультиплексора m(адр) , то из общего числа n переменных функции m(адр) можно подать на адресные входы. Тогда на информационные входы мультиплексора через дополнительную логическую схему подаются n-m(адр) 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. **Рис.**

Построение демультиплексора «1>4» на элементах И показано на рисунке б. Работа демультиплексора описывается логическими выражениями:, где mi – минтермы n адресующих переменных.

Применительно к мультиплексорам и демультиплексорам пользуются также термином «селекторы данных».

**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.

В большинстве случаев невозможно узнать точное значение приближенного числа, а значит, и точную величину погрешности. Однако почти всегда можно установить, что погрешность (абсолютная или относительная) не превосходит некоторого числа.

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.**

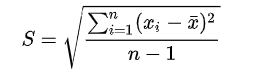
**Округление** применяется для представления значений и результатов вычислений с тем количеством знаков, которое соответствует реальной точности измерений или вычислений, либо той точности, которая требуется в конкретном приложении. Округление в ручных расчётах также может использоваться для упрощения вычислений в тех случаях, когда погрешность, вносимая за счёт ошибки округления, не выходит за границы допустимой погрешности расчёта.

**Оценка погрешности**

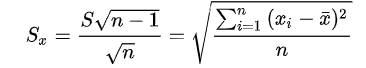
В зависимости от характеристик измеряемой величины для определения погрешности измерений используют различные методы.

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:



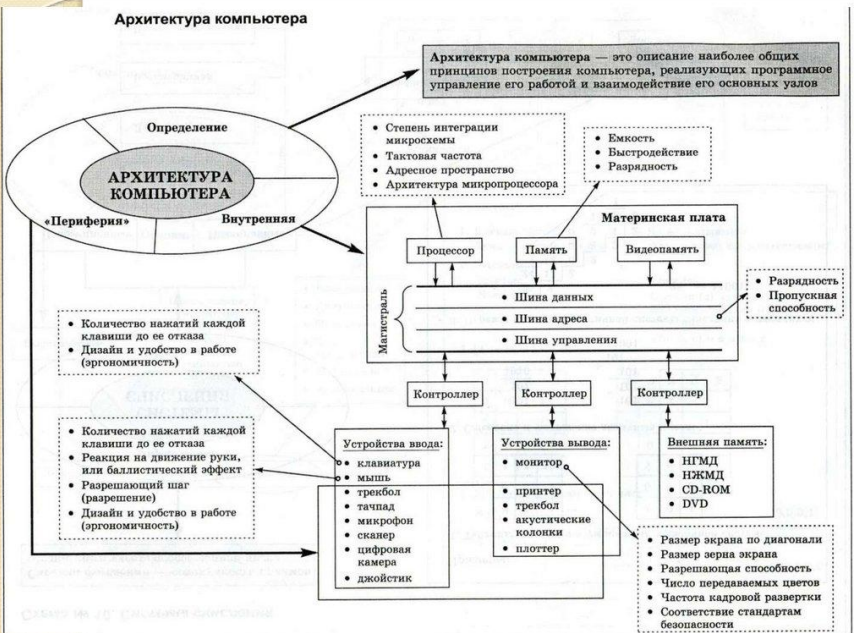
На основании смещённой оценки дисперсии:



61. Архитектура процессора. Основные компоненты. Способы классификации. Много уровневая организация. Контроллеры вввода-вывола

Архитекту́ра проце́ссора — количественная составляющая компонентов микроархитектуры вычислительной машины (процессора компьютера) (например, регистр флагов или регистры процессора), рассматриваемая IT-специалистами в аспекте прикладной деятельности.

**Основные компоненты**



**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[1]), так и по назначению (например, специализированные графические, математические или предназначенные для цифровой обработки сигналов)

**Многоуровневая организация**

**Контроллеры ввода-вывода**

1)Super I/O (англ. Super Input/output) — 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.

Как правило, включает в себя следующие функции:

• контроллер дисковода гибких дисков (floppy);

• контроллер параллельного порта (LPT-порт);

• контроллер последовательных (COM) портов и портов клавиатуры и мыши (PS/2).

• игровой (MIDI или джойстик) или инфракрасный порты.

• контроллер Ethernet

62. RISK, CISK, MISC, VLIW. Отличительные особенности, сфера применения. Что такое Spectre и Meltdown.

**RISK**

RISC (англ. Reduced Instruction Set Computer — «компьютер с сокращённым набором команд») — 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**

CISC (англ. Complex Instruction Set Computer — «компьютер с полным набором команд») — 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**

MISC (англ. Minimal Instruction Set Computer — «компьютер с минимальным набором команд») - 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**

VLIW (англ. Very Long Instruction Word — «очень длинная машинная команда») — 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.

**Что такое Spectre и Meltdown**

Spectre — группа аппаратных уязвимостей, ошибка в большинстве современных процессоров, имеющих спекулятивное выполнение команд (англ.)рус. и развитое предсказание ветвлений, позволяющих проводить чтение данных через сторонний канал в виде общей иерархии кэш-памяти. Затрагивает большинство современных микропроцессоров, в частности, архитектур х86/x86\_64 (Intel и AMD) и некоторые процессорные ядра ARM.

Meltdown — аппаратная уязвимость категории утечка по стороннему каналу, обнаруженная в ряде микропроцессоров, в частности, производства Intel и архитектуры ARM. Meltdown использует ошибку реализации спекулятивного выполнения команд (англ.) рус. в некоторых процессорах Intel и ARM (но не AMD), из-за которой при спекулятивном выполнении инструкций чтения из памяти процессор игнорирует права доступа к страницам.

63. Сравнительная характеристика архитектур. В чем преимущества. Преимущества RISK. 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 (инструкции), предназначение, виды. Тактирование процессоров. Выполнение инструкций. Поток инструкций

**Виртуальные архитектуры**

Магические JAR-файлы, которые можно запустить на любой машине - это пример виртуальной JVM-архитектуры, которая, по сути, эмулируется на целевой реальной машине. Поэтому достаточно JVM-машины для целевой архитектуры для запуска на ней любой Java-программы. Другим примером виртуальной архитектуры является .NET CIL.

Из минусов виртуальных архитектур можно выделить меньшую производительность по сравнению с реальными архитектурами. Однако большим плюсом будет кроссплатформенность.

**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 (где бит изменяет положение), 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.

Чтобы уменьшить время выполнения программы, нужно либо оптимизировать (уменьшить) её, либо увеличить тактовую частоту. У части процессоров есть возможность увеличить частоту (разогнать процессор), однако такие действия физически влияют на процессор и нередко вызывают перегрев и выход из строя.

**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 два регистра инструкций, в которые загружается код команды и адрес текущей исполняемой команды.

Т.е., независимо от того, какой процессор использует наша система, все они читают двоичный код определенным образом, соответствующим своему семейству. Что они делают, так это берут определенное количество бит двоичного кода, который они выполняют, и интерпретируют их значение в соответствии с его расположением. Каждая инструкция кодируется следующим образом: первые цифры соответствуют коду инструкции и способу его выполнения, а последние биты - это сами данные или место, где находятся данные, на которых мы хотим выполнить инструкцию.

**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; Это очень маленькие блоки памяти, которыми можно управлять напрямую с помощью блока управления процессора. Они используются для выполнения всех видов общих задач, а не только для выполнения арифметических операций.

Наиболее распространенные регистры в процессоре независимо от его ISA:

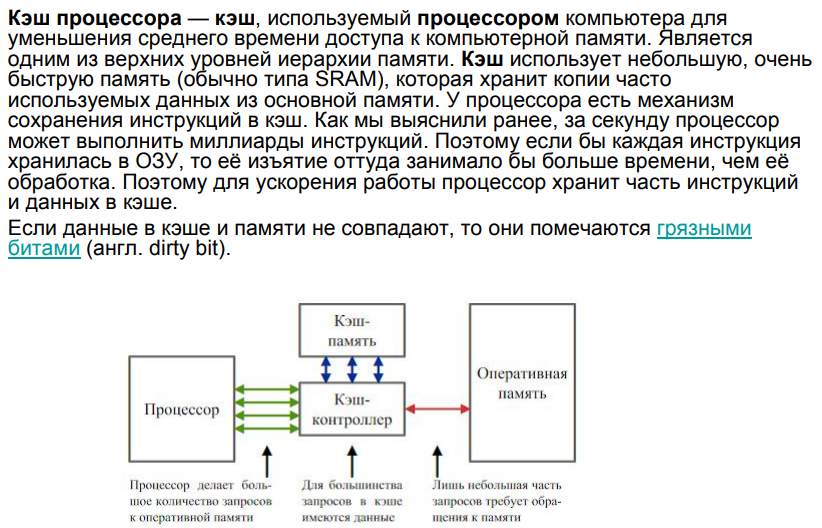
• Регистры типа аккумулятора : используется для арифметических операций. Каждое семейство имеет разное количество записей типа аккумулятора.

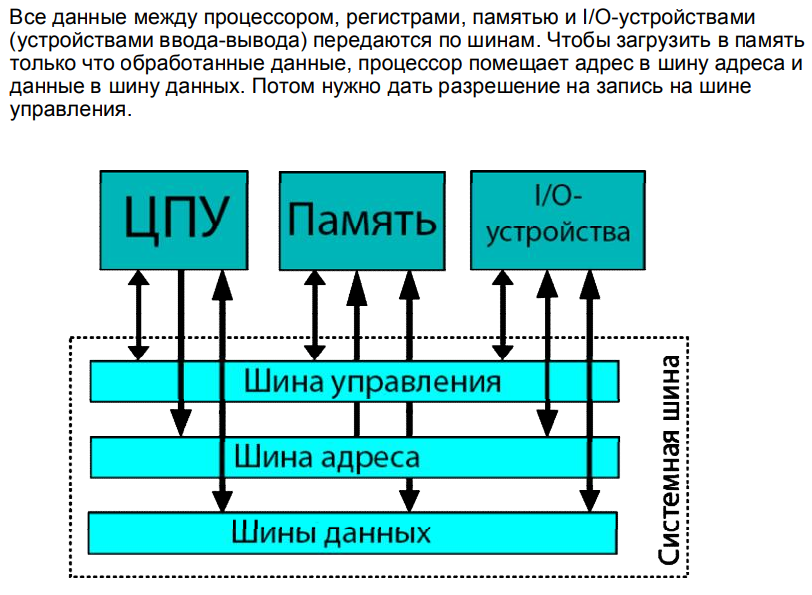
• Регистры доступа к памяти : содержат адрес памяти данных, к которым мы хотим получить доступ из ОЗУ.

• 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; Команды перехода изменяют их, когда вы хотите получить доступ не к следующей инструкции, а к другой части программы. В каждом полном командном цикле адрес памяти увеличивается на 1 и связывается с адресной шиной процессора.

**Шины**



**Кэш**

66. Что такое суперскалярная архитектура. Ее особенности. Предсказатели переходов. Иерархия памяти. Ветвление

**Суперскалярная архитектура**

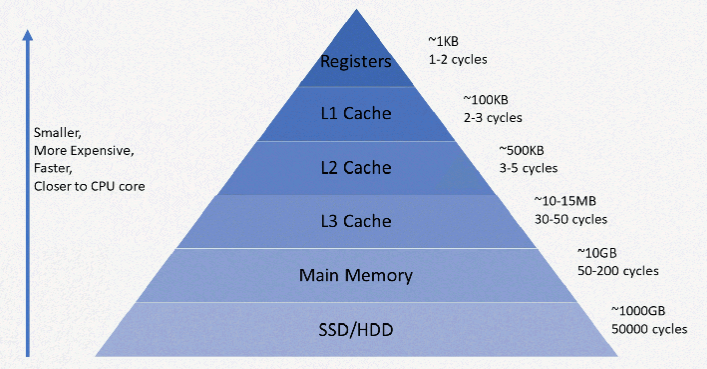
Это означает, что в любой момент времени процессор одновременно выполняет на каждом этапе конвейера множество инструкций. Он может также ожидать ещё сотни других, чтобы начать их выполнение, и для того, чтобы иметь возможность одновременного выполнения нескольких инструкций внутри процессоров есть несколько 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 гетерагенные 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.