**1. Системы счисления**

Все **системы счисления**(далее СС) можно подразделить на **позиционные** и **непозиционные**.

В **непозиционных** СС «доля» цифры или ее вес в количественном измерении записанного числа не зависит от местоположения данной цифры в записи этого числа. Примером такой СС является римская. В ней используются цифры:

I V X L C D M –римские цифры

1 5 10 50 100 500 1000 – их десятичные эквиваленты

При количественной оценке числа его значение определяется как сумма значений цифр, составляющих запись числа, кроме пар, состоящих из цифры меньшего веса, предшествующей цифре большего веса, значение которой определяется как разность веса большей и меньшей цифр.

**Количественная оценка числа**, записанного в позиционной СС, определяется как сумма произведений значения цифр, составляющих запись числа, умноженных на вес позиции, в которой располагается цифра.

Примером такой СС является широко используемая десятичная система счисления.

*Пример.*

Количественная оценка десятичного числа 395910 определяется как 3 ⋅ 1000 + 9 ⋅ 100 + 5 ⋅ 10 + 9 ⋅ 1, где 1000, 100, 10, 1 – соответственно веса четвертого, третьего, второго, первого разрядов записи оцениваемого числа

***Десятичная система счисления***является также системой с равномерно распределенными весами, которые характеризуются тем, что соотношение весов двух любых соседних разрядов имеет для такой системы одинаковое значение. Это соотношение называется **основанием системы счисления**, которое далее будем обозначать как «*q*».

Общая запись числа в системе с равномерно распределенными весами имеет вид

*Nq = Аn Аn–*1 *... А*2 *А*1 *А*0(кодированная запись). Значение такого числа определяется как

*Nq =Аn* ⋅*qn +Аn–*1 ⋅*qn–*1 *+Аn*–2 ⋅*qn*–1 *+...А*2 ⋅*q*2 *+А*1 ⋅*q*1 *+А*0

Помимо *q* = 10 (десятичная СС), возможны другие значения для основания СС:

− двоичная, восьмеричная, шестнадцатеричная и т. д.

*Примеры записи чисел в различных СС:*

*N*2 =10011011=1⋅27 +0⋅26 +0⋅25 +1⋅24 +1⋅23 +0⋅22 +1⋅21 +1⋅20;

*N*8 =471025=4⋅85 +7⋅84 +1⋅83 +0⋅82 +2⋅81 +5⋅80;

*N*10 =35491=3⋅104 +5⋅103 +4⋅102 +9⋅101 +1⋅100.

*N*16 =84FE4A=8⋅165 +4⋅164 +F⋅163 +E⋅162 +4⋅161 +A⋅160;

***Метод преобразования с использованием весов разрядов***

Метод преобразования с использованием весов разрядов записи числа в исходной и в искомой системах предполагает применение расширенной записи числа в некоторой системе 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.***

***Преобразование целых чисел***

Задачу представления числа *N*, заданного в системе *q*1, в системе счисления с основанием *q*2 можно рассматривать как задачу поиска коэффициентов полинома, представляющего собой расширенную запись числа *N* в системе счисления *q*2:

*Nq*1 *= а*0 *+ а*1 *× q* 21 *+ а*2 *× q*22 *+....+ аn–*2 *× q* 2*n–*2*+ аn–*1 *× q*2*n–*1*+*

*+ аn × q*2*n = Nq*2

Отсюда вытекает правило формирования коэффициентов полинома

или разрядов записи заданного числа *N* в системе счисления с основанием *q2*:

- необходимо разделить исходное число *N* *q*1 на новое основание *q*2, при этом получив целое частное и остаток;

- полученный остаток снова необходимо разделить на *q*2, процесс деления продолжается до тех пор, пока частное будет не меньше нового основания *q*2*.* Если очередное сформированное частное будет меньше, чем *q*2, то процесс формирования записи заданного числа в новой системе с основанием *q*2 считается законченным, а в качестве искомых разрядов новой записи числа используются результаты выполненных операций деления следующим образом:

- в качестве старшего разряда берется значение последнего частного, для остальных разрядов используются значения остатков в порядке, обратном порядку их получения.

***Преобразование дробных чисел***

Задача представления дробного числа *Mq*1, заданного в системе *q*1, в системе счисления с основанием *q*2, можно рассматривать как задачу поиска коэффициентов полинома, представляющего собой расширенную запись числа *M* в системе счисления *q*2:

*B*1 *× q*2*–*1 *+ B*2 *× q*2*–*2 *+ B*3 *× q*2*–*3 *+ ....* *+ Bn–*2 *× q*2*–(n–*2*) + B n–*1 *× q*2*–(n–*1*) + Bn × q*2*–n* = *Mq* (1.4)

Отсюда вытекает следующее правило формирования коэффициентов полинома, которые одновременно являются разрядами записи заданного числа *M* в системе счисления с основанием *q*2:

- определяется количество разрядов «*n*» в записи числа *Mq*2 в новой системе счисления;

- исходное число *Mq*1 умножается на *q*2, при этом будет получено смешанное число;

- дробная часть полученного произведения снова умножается на *q*2 и т. д.; процесс умножения повторяется *n* +1 раз. В качестве искомых разрядов новой записи числа используются результаты выполненных операции деления следующим образом:

- 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.***

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 (*условие 1*) или *q*2 *m* = *q*1 (*условие 2*). Если имеет место *условие 2*, то для заданного в системе с основанием *q*1 числа *Nq*1 *= аn аn*-1 *аn*-2... *а*1*а*0 запись его в системе с новом основании *q*2 определяется следующим образом:

- каждому разряду *ai* исходной записи числа ставится в соответствие его *m-*разрядный эквивалент в системе счисления с основанием *q*2;

- искомая запись всего заданного числа формируется за счет объединения всех полученных *m*-разрядных групп.

Если имеет место *условие 1*, то запись заданного числа *N* =*аnаn*-1*аn-*2...*а*1*а*0 в системе с новом основании *q*2 формируется следующим образом:

- 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;

- каждой полученной группе ставится в соответствие цифра новой системы счисления;

- искомая запись заданного числа в новой системе счисления образуется из цифр, соответствующих группам, на которые была разбита исходная запись.

**4.Арифметические операции над двоичными числами. Операция сложения в 2-й СС**

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*

|  |  |  |
| --- | --- | --- |
| + | 0 | 1 |
| 0 | 0 | 1 |
| 1 | 1 | 0*\** |

Все возможные значения первого слагаемого задаются во второй и третьей строках первой колонки; все возможные значения второго слагаемого – во второй и третьей колонках первой строки. На пересечении отмеченных значениями операндов строк и колонок располагается результат их сложения. В таблице знаком «*\**» отмечен случай, когда в текущем разряде результата получен ноль и имеет место перенос в ближайший старший разряд

В общем случае при формировании значения в текущем разряде результата приходится дважды применять приведенную таблицу сложения: первый раз при сложении соответствующих разрядов операндов, формируя так называемую поразрядную сумму, и второй раз *–* при сложении разряда сформированной поразрядной суммы и переноса, пришедшего из ближайшего младшего разряда.

При формировании поразрядной суммы и учете возникших переносов используется следующая классификация разрядов складываемых операндов:

- разряд, генерирующий перенос (оба операнда в этом разряде имеют «1»);

- разряд, пропускающий перенос (операнды в этом разряде име­ют разные значения);

- разряд, блокирующий распространение переноса (операнды в этом разряде имеют одинаковые значения).

***Операция вычитания в СС.***

Правила вычитания в СС.

|  |  |  |
| --- | --- | --- |
| - | 0 | 1 |
| 0 | 0 | 1 |
| 1 | 1\* | 0 |

Звездочкой отмечен случай, когда в текущем разряде получена единица, путем займа из ближайшего старшего разряда.

*Пример.*

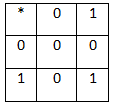
\_ 1000111001

0101101101

0011001100

В ЭВМ никогда в перечне выполняемых операций арифметического устройства не присутствует одновременно операция сложения и операция вычитания. При этом, как правило, присутствует только операции сложения. Что же касается операции вычитания, то она реализуется за счет прибавления к уменьшаемому значения вычитаемого,   
взятого с противоположным знаком.

**5. Операция умножения в СС начиная со старшего разряда множителя со сдвигом множимого.**



При умножении многоразрядных операндов, как правило (особенно в десятичной системе счисления), 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. При последовательном способе формирования частичных произведений последние могут рассчитываться поочередно для отдельных разрядов множителя начиная с младшего или старшего разряда. При десятичном основании, как правило, формирование частичных произведений осуществляется начиная с младшего разряда множителя.

**Операция умножения в 2-й СС начиная с младшего разряда множителя со сдвигом множимого.**

При последовательном способе формирования частичных произведений последние могут рассчитываться поочередно для отдельных разрядов множителя начиная с младшего или старшего разряда. При десятичном основании, как правило, формирование частичных произведений осуществляется начиная с младшего разряда множителя.

Реализация данного метода умножения требует использовать 2n-разрядный сумматор для последовательного, от такта к такту, формирования 2n-разрядного произведения и 2n-разрядный регистр для хранения и сдвига влево множимого. В данном примере для того, чтобы учесть то, что очередной разряд множителя имеет вес, в два раза больший, чем предыдущий разряд, его частичное произведение учитывается со сдвигом множимого на один разряд влево при суммировании с промежуточным результатом. В таком случае говорят, что умножение выполняется со сдвигом множимого.

**Возможные методы реализации операции умножения можно классифицировать по двум признакам:**

– Начиная с какого разряда (со старшего или младшего) выполняется отработка множителя;

– Что сдвигается – множимое или промежуточное произведение.

Используя эти два классификационных признака, **можно выделить четыре метода умножения**:

– **Умножение с младших разрядов множителя со сдвигом множимого**; при реализации данного метода требуется 2n- разрядный сумматор, 2n- разрядный регистр промежуточного произведения, 2n - разрядный регистр для хранения и сдвига множимого и n- разрядный регистр для хранения множителя;

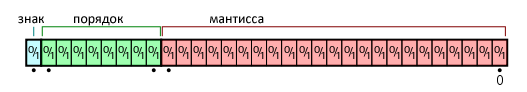
– **Умножение с младших разрядов множителя со сдвигом промежуточного произведения**; при реализации данного метода требуется n-разрядный сумматор, 2n-разрядный регистр промежуточного произведения, n-разрядный регистр для хранения множимого и n-разрядный регистр для хранения множителя;

– **Умножение со старшего разряда множителя со сдвигом множимого**; при реализации данного метода требуется 2n-разрядный сумматор, 2n-разрядный регистр промежуточного произведения, 2n- разрядный регистр для хранения и сдвига множимого и n-разрядный регистр для хранения множителя;

– **Умножение со старшего разряда множителя со сдвигом промежуточного произведения**; при реализации данного метода требуется n-разрядный сумматор, 2n-разрядный регистр промежуточного произведения, n-разрядный регистр для хранения множимого и n-разрядный регистр для хранения множителя.

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

Число с плавающей точкой состоит из набора отдельных разрядов, условно разделенных на знак, ~~экспоненту~~ порядок и мантиссу. Порядок и мантисса — целые числа, которые вместе со знаком дают представление числа с плавающей запятой в следующем виде:



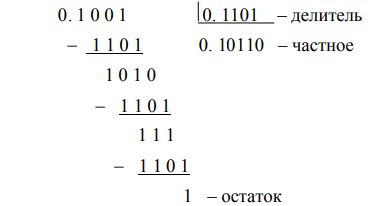
*Мантисса* – это целое число фиксированной длины, которое представляет старшие разряды действительного числа

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

Также 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 переполнени.

**7. Деление двоичных чисел.**

При 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.



После округления получаем окончательный результат: 0.1001 / 0.1101 = 0.1011.

**8. Деление двоичных чисел с восстановлением остатка**

Деление выполняется по алгоритму с восстановлением остатка на сумматоре дополнительного кода в следующей последовательности:

-определяется знак частного

-представление делимого и делителя в машинных кодах, когда делимое всегда, независимо от его знака, берется в прямом коде с положительным знаком, а делитель

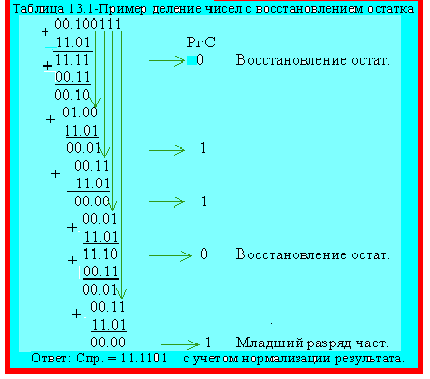
всегда, независимо от его знака, берется в дополнительном коде с отрицательным знаком;

-устранение дробной части в делителе, путем переноса запятой вправо на n разрядов (по аналогии с десятичной системой счисления). Чтобы дробь не изменилась, в делимом также переносят 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:

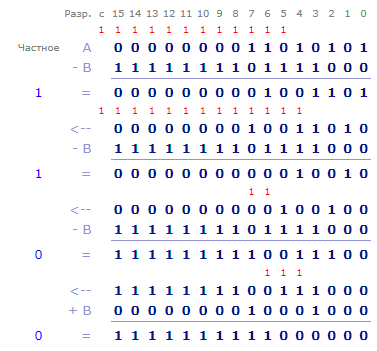
1) если знак промежуточного остатка 00 (положительный), то в регистр частного РгС записывается 1, начиная со старшего разряда. Остаток сдвигается на один разряд влево

2) если знак промежуточного остатка 11 (отрицательный), то в регистр частного РгС записывается 0, начиная со старшего разряда. Остаток восстанавливается путем прибавления к нему делителя в прямом коде с положительным знаком. Восстановленный остаток сдвигается влево на один разряд

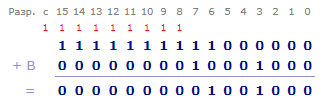
****

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

**Процесс деления** будет следующий:  
6.1) Вычитаем из делимого А делитель В (т.е. прибавляем -В).  
6.2) Анализируем знак полученного частичного остатка (15-й разряд). В регистр результата записываем "0" если остаток отрицательный и единицу в противном случае. Помним, что отрицательному числу соответствует наличие единицы в 15-м разряде и наоборот.  
6.3) Сдвигаем частичный остаток на один разряд влево. При этом крайний правый (младший) разряд заполняется нулем, а знаковый разряд (15-й) в процессе сдвига не участвует.  
6.4) Прибавляем к частичному остатку делитель В если остаток отрицательный либо вычитаем делитель в противном случае.  
6.5) Анализируем знак полученного частичного остатка (15-й разряд). В регистр результата записываем "0" если остаток отрицательный и единицу в противном случае.  
6.6) Действия описанные в пунктах 6.3-6.5 выполняем k раз (если k=0, то ни разу не выполняем).  
Весь процесс деления выглядит следующим образом :



Определяем **остаток от деления**. Для этого анализируем последний частичный остаток. В нашем случае он равен **"1111111111000000"**.

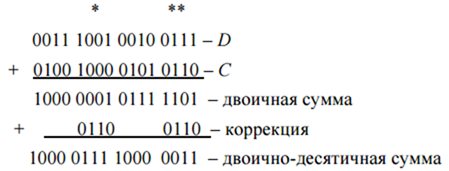


Так как в процессе деления частичные остатки были сдвинуты 3 раза влево, то для получения верного значения последний полученный остаток необходимо сдвинуть 3 раза вправо

Определяем **знак результата**. Если знаки исходных операндов одинаковы, то результирующее частное положительно и наоборот. В нашем случае знаки совпадают, следовательно результирующее частное положительно.

**10. Сложение двоично-десятичных чисел**

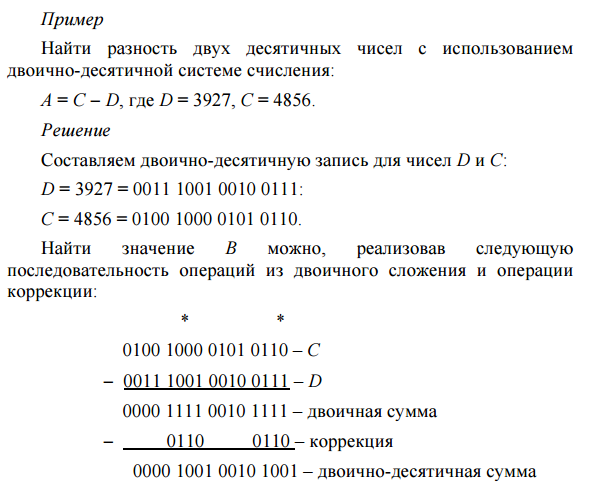
Рассмотрим на конкретном примере реализацию этой операции. Пример Найти сумму двух десятичных чисел с использованием двоично-десятичной системы счисления: A = D + C, где D = 3927; C = 4856. Решение Составляем двоично-десятичную запись для чисел D и C: D =3927 = 0011 1001 0010 0111: C = 4856 =0100 1000 0101 0110. Найти значение А можно, реализовав следующую последовательность операций из двоичного сложения и операции коррекции:



Для получения двоично-десятичной суммы A на основании результата сложения операндов по правилам двоичной арифметики необходимо добавить шестерку (0110) в те тетрады, из которых был перенос. В данном примере это вторая тетрада (отмечена \*). Необходимость такой коррекции обусловливается тем, что перенос, сформированный по правилам двоичного суммирования, унес из тетрады шестнадцать, а для десятичного сложения перенос должен был унести десять, т.е. перенос, сформированный по правилам двоичной арифметики, унес лишнюю шестерку. Кроме этого шестерка добавляется в те тетрады, в которых получено значение, большее девяти. Такая коррекция обуславливается тем, что по правилам десятичной арифметики в25 таких тетрадах должен быть выработан перенос и, чтобы его выработать по правилам двоичной арифметики, в тетраду нужно добавить шестерку. Для рассмотренного примера такой тетрадой является и четвертая тетрада (отмечена \*\*)

**Двоично-десятичная арифметика. Вычитание двоично-десятичных чисел.**

В ЭВМ часто предусматривается обработка чисел не только в двоичной системе счисления, но в двоично-десятичной. При этом, как правило, стремятся реализовать двоично-десятичную арифметику по правилам двоичной с введением ограниченного количества коррекций.



Для получения двоично-десятичной разности «A» на основании результата вычитания операндов по правилам двоичной арифметики необходимо вычесть шестерку (0110) из тетрад, в которые пришел заем. Это обусловливается тем, что заем, сформированный по правилам двоичного вычитания, приносит в тетраду шестнадцать, а для десятичного сложения заем должен был принести в тетраду десять, т.е. заём, сформированный по правилам двоичной арифметики, принес лишнюю шестерку. Для рассмотренного примера тетрадами, в которые пришел заем и в которых необходимо выполнить коррекцию (вычесть шестерку), являются вторая и четвертая тетрады (отмечены \*).

***11 .Кодирование алгебраических чисел .***

Для представления чисел со знаком используются специальные коды:

- прямой код;

- дополнительный код;

- обратный код.

Во всех трёх случаях используется следующий формат представления числа, содержащий два поля - поле знака и поле модуля.

*Поле знака* представлено одним разрядом, в котором устанавливается 0, если число положительное, и 1, если число отрицательно.

*Поле модуля* отражает количественную оценку числа и для каждого кода формируется по*–*разному. Количество разрядов поля модуля определяется диапазоном изменения отображаемых чисел или точностью их представления.

Перевод отрицательного числа из обратного или дополнительного кода в прямой выполняется по тем же правилам, что и перевод числа из прямого кода в обратный или *дополнительный*:

– для перевода отрицательного числа из обратного в *прямой код* необходимо дополнить его модуль до включенной границы;

– для перевода отрицательного числа из обратного в *прямой код* необходимо дополнить его модуль до невключенной границы.

***Дополнительный и обратный коды двоичных чисел***

При переводе двоичных чисел в качестве включенной и не включенной границы диапазона изменения абсолютных значений представляемых чисел используется соответственно 2n и 2n - 1.

Обратный код **положительного** числа **совпадает с его прямым кодом,** обратный код **отрицательного** числа формируется **по правилам:** в знаковом разряде **записывается “1**”; цифровые **значения меняются** на противоположные(**0 на 1,1 на 0),т.е.** необходимо **проинвертировать модуль прямого кода.**

Переход от обратного кода отрицательного числа к представлению в прямом коде осуществляется по тому же правилу, т.е. необходимо проинвертировать модуль записи числа в дополнительном коде.

*Дополнительный код числа*, имеет такое же назначение, как и обратный код числа. Формируется по следующим правилам: положительные числа в дополнительном коде выглядят также как и в обратном и в прямом коде, т.е. не изменяются. Отрицательные числа кодируются следующим образом: к обратному коду отрицательного числа (к младшему разряду) прибавляется 1, по правилу двоичной арифметики.

Переход от дополнительного кода отрицательного числа к прямому осуществляется по тому же правилу, т.е. необходимо проинвертировать модуль записи числа в дополнительном коде, и к полученному коду прибавить 1 в младший разряд.

При выполнении операций над числами со знаком в ЭВМ используются прямой, обратный и дополнительный коды. Как правило, информация в памяти хранится в прямом коде, а при выполнении операций применяется или обратный, или дополнительный код.

***12. Операции с двоичными числами в дополнительном коде***

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!

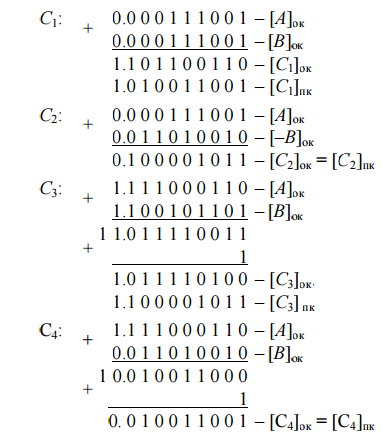
1. Исходя из абсолютного значения операндов, разрядность представления модульной части n должна быть равна количеству разрядов большего операнда. Учитывай то, что мы используем две операции: сложение и вычитание, поэтому возможно переполнение из-за переноса из старшего разряда, БЕРИ длину модульной части на один разряд больше, т.е n+1

2. Избавься от операции вычитания ( A-B = A + (-B) )

3. Далее свои величины A и B представляешь в дополнительном коде.  
**НЕ ЗАБЫВАЙ!** Если при выполнении сложения у тебя возникла единица переполнения знакового поля. При работе с дополнительным кодом она игнорируется (в примере она подчёркнута).

***Операции с двоичными числами в обратном коде***

При сложении чисел, представленных в *обратном* коде, выполняется сложение разрядов, представляющих запись операндов, по правилам двоичной арифметики по всей длине записи чисел, не обращая внимания на границу, разделяющую знаковое и модульные поля. Переполнение знакового поля, т.е. перенос, возникший из крайнего левого разряда, должен быть учтен как +1 в младший разряд полученной суммы. В результате такого сложения будет получен *обратный* код суммы заданных операндов.



В данном случае также возникло переполнение знакового разряда, которое должно быть учтено как +1 в младший разряд сформированной суммы

***13. Модифицированные коды***

При расчете разрядности n модульного поля весьма трудно бывает учесть диапазон значений результатов, особенно когда последовательность операции, представленных в подлежащих реализации выражениях, достаточно сложны.

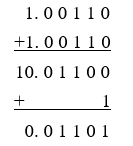
При несоответствии выбранной разрядности n диапазону изменения представляемых чисел при выполнении операции сложении чисел с одинаковыми знаками возможно появление ситуации переполнения, когда подлежащий представлению результат выходит за диапазон представления, определенный некорректно выбранной разрядностью n поля модуля.

Например, в случае сложения двух чисел, представленных в обратном коде:

[D1]ок = 1.00110 и [D2]ок = 1.00110.

Сумма этих чисел F1 = D1 + D2 будет подсчитана следующим образом:

F1:



Пример, выполненный по всем формальным правилам, дал абсурдный результат, так как получена положительная сумма двух отрицательных операндов. Аналогичная ситуация может возникать и при использовании дополнительного кода.

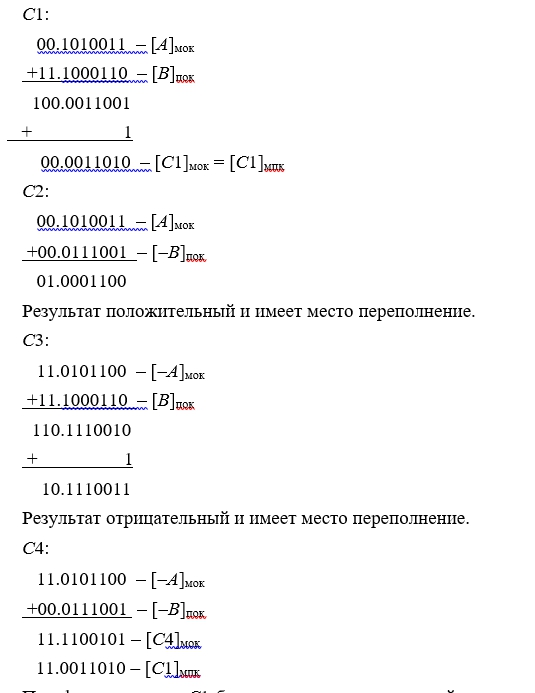
Ситуацию переполнения можно обнаруживать по факту появления «абсурдного» результата, но для этого необходимо помнить то, что в суммировании принимают участие операнды с одинаковыми знаками и знак полученного при этом результата отличен от знака операндов.

Более просто ситуация переполнения определяется при применении модифицированного кода (обратного или дополнительного). Модифицированные коды отличаются от базовых кодов только тем, что поле знака операндов имеет два разряда, и эти разряды имеют одинаковые значения:

00 – для положительных чисел;

11 – для отрицательных чисел.

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).



При формировании С1 был получен положительный результат (без переполнения).

При формировании С4 был получен отрицательный результат (без переполнения).

Факт переполнения при формировании С3 и С2 устанавливается по наличию в разрядах знакового поля различных значений.

**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:

1) *логическое суммирование* (обозначения – ИЛИ, ОR, «∨»);

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

3) *отрицание* (обозначения – НЕТ, NOT, «*x*», т.е. штрих над отрицаемым *x*);

4) *суммирование по модулю* 2 (обозначается mod 2, « ⊕ »);

5) *операции сдвига*.

**Логические операции**

Операция *логического суммирования* выполняется над двумя кодами и генерирует код той же разрядности, что и операнды, у которого в некотором *i*-м разряде находится единица, если хотя бы в одном операнде в *i*-м разряде имеет место единица.

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

Операция *суммирования по модулю 2* выполняется над двумя кодами и генерирует код той же разрядности, что и операнды, у которого в некотором *i*-м разряде находится единица, если два заданных операнда в *i*-м разряде имеют противоположные значения. Иногда эта операция называется «исключающее ИЛИ».

Операция логического отрицания выполняется над одним кодом и генерирует результирующий код той же разрядности, что и операнд, в некотором *i*-м разряде которого находится значение, противоположное значению в *i*-м разряде отрицаемого кода.

Операции *сдвига* в свою очередь, подразделяются на:

1) логические сдвиги, которые имеют разновидности – сдвиг вправо, сдвиг влево, циклический сдвиг вправо, циклический сдвиг влево;

2) арифметические сдвиги вправо и влево, выполнение которых зависит от знака и кода сдвигаемого числа.

**15. Арифметические сдвиги положительных двоичных чисел, представленных в прямом коде.**

Арифметические сдвиги обеспечивают выполнение умножения (сдвиги влево) или операции деления (сдвиги вправо) двоичных кодов на два, точно так же, как сдвиги вправо и влево десятичного числа обеспечиваю выполнение деления и умножение на 10. 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).

Пример: Найти результат арифметического сдвига влево на три разряда двоичного прямого кода числа [А]пк = 00.00000101

Решение: Процесс выполнения заданного сдвига дает следующие промежуточные и конечное значения:

первый сдвиг: 00.00000101 ← 00.00001010;

второй сдвиг: 00.00001010 ← 00.00010100;

третий сдвиг: 00.00010100 ← 00.00101000.

**Арифметические сдвиги двоичных чисел, представленных в обратном коде.**

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.

*Пример3*

Выполнить сдвиг вправо на четыре разряда двоичного числа 11.1001101 (десятичный эквивалент – 5010), представленного в обратном коде.

Первый сдвиг дает 11.11001101 (5010) à 11.11100110 (2510).

Второй сдвиг дает 11.11100110 (2510) à 11.11110011 (1210).

Третий сдвиг дает 11.11110011 (1210) à 11.11111001 (610).

Четвертый сдвиг дает 11.11111001 (610) à 11.11111100 (310).

При выполнении сдвига вправо нечетного числа результат получается с точностью до младшего разряда кода, причем ошибка отрицательная.

После выполнения последнего, четвертого сдвига выполняется округление, при котором, если последний «вытолкнутый» разряд имел значение 0, к результату последнего сдвига прибавляется – 1.

Данное округление можно выполнить за счет прибавления единицы к прямому коду, соответствующему результату последнего сдвига исходного обратного кода.

В рассмотренном примере корректировать на единицу результат четвертого сдвига не надо, так как «вытолкнутый» разряд при последнем (четвертом) сдвиге равен единице. В данном случае конечный результат сдвига заданного отрицательного числа, представленного в обратном коде, равен 11.11111100.

***16. Арифметические сдвиги отрицательных двоичных чисел, представленных в прямом коде.***

Арифметические сдвиги влево и вправо реализуются по-разному в зависимости как от знака числа, так и от используемого кода (прямого обратного, дополнительного).

При арифметическом сдвиге отрицательного двоичного числа, представленного в прямом коде, осуществляется соответствующий сдвиг только модульного поля записи числа.

Реализация этого типа сдвига иллюстрируется следующими примерами.

***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).

**Пример**

Выполнить сдвиг вправо на четыре разряда двоичного числа 11.11001110 (десятичный эквивалент – 50 в 10 с/c), представленного в дополнительном коде.

Решение

Первый сдвиг дает 11.11001110 → 11.11100111 (25 в 10 с/c).

Второй сдвиг дает 11.11100111 → 11.11110011 (13 в 10 с/c).

Третий сдвиг дает 11.11110011 → 11.11111001 (7 в 10 с/c).

Четвертый сдвиг дает 11.11111001 → 11.11111100 (4 в 10 с/c).

При выполнении сдвига вправо нечетного целого числа результат получается с точностью до младшего разряда кода, причем ошибка положительная.

Арифметический сдвиг вправо может выполняться над отрицательными числами с переполнением (такие числа в модифицированном прямом, обратном или дополнительном коде имеют в знаковом поле 10). В этом случае после сдвига в знаковом поле будет 11, а в старшем разряде – 0, если число представлено в обратном или дополнительном коде, или 1, если число представлено в прямом коде.

**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, форматы которых приведены на рис. 1.3.К заданному виду (целым числам или правильной дроби) исходные числа приводятся за счет введения масштабных коэффициентов.

|  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- |
| Зн | 1р | 2р | 3р | 4р | .... | (*n*-1) | *n*р | «.» |

*а*

|  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- |
| Зн | «.» | 1р | 2р | 3р | 4р | .... | (*n*­1) | *n*р |

*б*

Представление чисел с фиксированной точкой: *а* – формат целого числа; *б –* формат дробного числа

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*

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 определяются следующим образом.

В случае целых чисел:

Dmax = 0.5; dmax = Dmax / *А* min = 0.5, где *А*min – минимальное, отличное от нуля, значение числа.

В случае дробных чисел: Dmax = 0.5×2*n* = 2(*n*+1); dmax = Dmax / *А*min = 2(*n*+1) / 2-*n* = 0.5,

т.е. в худшем случае относительная ошибка при фиксированной точке может достигать сравнительно большого значения – 50%.

**Арифметические операции над числами, представленными с фиксированной точкой**

К числу основных арифметических операций, непосредственно реализуемых в ЭВМ, относятся операции сложения, умножения, деления. Остальные операции (например, такие, как возведение в степень, извлечение квадратного корня) реализуются программным способом.

Выполнение операций с числами, представленными с фиксированной точкой, рассмотрено в рамках материала по выполнению операций с алгебраическими числами. 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 результата (произведение или частное), которому затем присваивается предварительно определенный знак.

Операнды, 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.**

При представления числа *с плавающей точкой* число в общем случае представляет собой смешанную дробь и имеет формат, приведенный на рис. 1.4

|  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| 1р | 2р. | 3р. | ... | *к* р. | «.» | (*к*+1) р. | (*к*+2) р. | ... | (*n*­1) р. | *n*р |

Рис. 1.4. Формат представления числа с плавающей точкой

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.

Для двоичной системы счисления имеет место

*Х* = *2x*п × *х*м.

При *s*-разрядном представлении модуля записи мантиссы и *k*-разрядном представлении модуля записи порядка форма с плавающей точкой обеспечивает диапазон изменения абсолютного значения числа *А*

В ЭВМ числа с плавающей точкой представляются в так называемой нормализованной форме, при которой в прямом коде мантисса нормализованного числа в старшем разряде модуля имеет ненулевое значение, а для двоичной системы счисления – нормализованная мантисса должна иметь в старшем разряде модуля прямого кода значение 1, т.е. для двоичной системы мантисса должна удовлетворять неравенству:

Для плавающей точки максимальные значения абсолютной и относительной ошибок определяются следующим образом.

Максимальная абсолютная погрешность представления чисел:

Dmax = 2-(*s*+1) × 2*p*;

Максимальная относительная погрешность:

dmax = D max / *А*min = 2-(*s*+1) × 2*p* / (*х*м min × 2*p*)= 2­(s+1) × 2*p* / ( 2-1 × 2*p*) = 2­(*s*+1) / (2-1) = =2-*s*.

Отсюда видно, что 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.

**Сложение чисел, представленных в формате с плавающей точкой.**

Числа, представленные в формате с плавающей точкой (запятой) имеют две части – мантиссу и порядок. Поэтому, операция алгебраического сложения выполняется отдельно над мантиссой и над порядком. Следовательно, в цифровом автомате может быть два суммирующих устройства, для мантиссы и для порядка.

Для чисел с плавающей точкой справедливо условие нормализации: **q-1< | mA| < 1,** где q - основание системы счисления; mA - мантиссы числа.

При сложении чисел, результат сложения может выйти из нормализации как справа уравнения **q-1< | mA| < 1,**  так и слева.

Обозначим через *γ -*признак нарушения нормализации числа справа, указывающий на необходимость сдвига числа вправо на один разряд для восстановления знака числа.

Признаком нарушения нормализации числа слева является наличие одинаковых комбинаций в разряде переполнения и старшем разряде цифровой части сумматора.

Сложение мантиссы осуществляют на сумматоре ДСДК или ДСОК по правилу сложения чисел аналогично в формате с фиксированной запятой. Если после сложения мантисса результата удовлетворяет условию нормализации, то к этому результату приписывается порядок любого из операндов. В противном случае производится нормализация числа.

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

Сначала рассмотрим алгоритм умножения. Сразу же после начала операции проверяется, не равен ли нулю один из сомножителей. Если это так, то произведение также будет равно нулю. Следующий шаг — суммирование порядков. Поскольку, как правило, для хранения порядков используется смещенное представление, при суммировании двух смещенных представлений результат будет смещен дважды. Поэтому после суммирования кодов порядков из суммы вычитается значение смещения. При суммировании может возникнуть как переполнение порядка, так и потеря значимости. В обоих случаях формируется соответствующий сигнал.

Если порядок произведения не выходит из диапазона, определенного форматом, далее перемножаются мантиссы сомножителей с учетом их знаков. Умножение мантисс выполняется по тому же алгоритму, что и умножение целых чисел в прямом коде, т.е. фактически перемножаются числа без знака, а затем произведению приписывается знак "плюс" или "минус" в зависимости от сочетания знаков сомножителей. Произведение мантисс имеет разрядность, вдвое большую, чем каждый из сомножителей. Лишние младшие разряды отбрасываются при округлении. После того как будет получено произведение мантисс, результат нормализуется и округляется. Эти операции выполняются так же, как и при сложении или вычитании. Необходимо учесть, что при нормализации может возникнуть переполнение или потеря значимости порядка.

Теперь рассмотрим алгоритм деления. Как и ранее, первый этап — анализ операндов на равенство нулю. Если нулю равно делимое, то результату сразу присваивается значение 0. Если же нулю равен делитель, то в зависимости от конкретной реализации АЛУ результату может быть присвоено значение "бесконечность" с соответствующим знаком или сформирован сигнал арифметической ошибки. Следующий этап — вычитание кода порядка делителя из кода порядка делимого. При этом получится несмещенный код разности, который нужно скорректировать — сложить с кодом смещения. После завершения операций с порядком результата проверяется, не возникло ли переполнение порядка или потеря значимости. Следующий этап — деление мантисс. За ним следуют обычные операции нормализации и округления.

**20. Неосновные арифметические операции. Вычисление квадратного корня**

Есть два способа аппаратной реализации процедуры извлечения квадратного корня. Первый способ связан с разработкой микропрограммы извлечения квадратного корня, основанной на использовании основных арифметических операций и являющейся реализацией того или иного итерационного метода извлечения квадратного корня. Другой путь состоит в создании алгоритма извлечения квадратного корня сходного по структуре с алгоритмами основных арифметических действий. Предполагается, что второй вариант алгоритма выполняется быстрее, но он обычно требует достаточно существенного усложнения блока управления арифметического устройства цифрового автомата.

Весь процесс извлечения кадратного корня из мантиссы операнда разбивается на ряд однотипных шагов, выполняемых последовательно, и на каждом шаге находится очередная цифра результата, начиная со старшей. Этот алгоритм очень похож на обычный алгоритм деления, за тем исключением, что на каждом очередном шаге реализации алгоритма все время меняется "делитель", который формируется в регистре SR из текущего значения результата извлечения квадратного корня. Окончательный результат получается за *m* -1 элементарных шагов алгоритма или же после того шага, на котором очередной остаток получился равнм нулю.

**21. Методы вычисления элементарных функций**

Разложение в ряд Тейлора

Разложение в ряд Тейлора вычисляются в ЭВМ чаще всего по схеме Горнера. При этом требуется выполнить m операций умножения и m операций сложения (m – степень полинома ) Методическая погрешность повышается вместе с ростом аргумента . Достоинства : можно вычислять коэффициенты членов ряда при вычислении функции и не хранить их в памяти ЭВМ, однако при этом возрастает время вычисления ЭФ.

Метод полиномиальной аппроксимации

Характеризуется высоким единообразием вычислений всех ЭФ, однако при этом в памяти необходимо хранить большое количество коэффициентов всех полиномов.

Табличный метод

Основан на кусочно-линейной и криволинейной аппроксимации. Для вычисления Эф требуется выполнить малое число арифметических операций, однако объем таблиц и время поиска может быть большим

Метод рационального приближения

Функцию представляют в виде отношения двух полиномов, причем число членов в каждом полиноме намного меньше, чем при соответствующем разложении в ряд Тейлора. Однако коэффициенты должны обязательно храниться в памяти. Для вычисления Эф необходимо вычислить два полинома и выполнить операцию деления.

Метод цепных дробей

Характеризуется единообразием вычисления всех ЭФ, при этом количество констант мало. Метод используется в малых машинах, где быстродействие не очень важно.

Итерационны метод

Предполагает вычисление последовательных приближений функции по итерационной формуле . Загрузка памяти наименьшая, т.к константы можно вычислить непосредственно перед счетом функции по той же схеме. Это снижает быстродействие, но является решающим преимуществом при использовании в машинах с произвольной разрядностью.

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

Денормализованные числа - вид [чисел с плавающей запятой](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), определённый в стандарте [IEEE 754](https://ru.wikipedia.org/wiki/IEEE_754).

Причина, по которой были введены денормализованные числа, является частью более общей вычислительной проблемы нахождения суммы чисел при использовании ограниченной точности (см., например, [алгоритм Кэхэна](https://ru.wikipedia.org/wiki/%D0%90%D0%BB%D0%B3%D0%BE%D1%80%D0%B8%D1%82%D0%BC_%D0%9A%D1%8D%D1%85%D1%8D%D0%BD%D0%B0)). Причём проблема, решаемая введением денормализованных чисел, возникает в сравнительно узком диапазоне чисел — вблизи границы [потери значимости](https://ru.wikipedia.org/wiki/%D0%98%D1%81%D1%87%D0%B5%D0%B7%D0%BD%D0%BE%D0%B2%D0%B5%D0%BD%D0%B8%D0%B5_%D0%BF%D0%BE%D1%80%D1%8F%D0%B4%D0%BA%D0%B0). Но аналогичная проблема может быть связана и с переполнением: например, если мы захотим сравнить два числа **a и b** разного знака, [порядок](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) которых максимальный, то условие **a = b** будет

успешно проверено и даст отрицательный результат, а условие **a-b=0** может привести к ошибке переполнения. Таким образом, в общем случае денормализованные числа не решают проблему зависимости результата от перестановки слагаемых, и от программиста при работе с плавающей арифметикой всё равно требуется определённая аккуратность. В случае, если программисту приходится работать с числами на грани [потери значимости](https://ru.wikipedia.org/wiki/%D0%98%D1%81%D1%87%D0%B5%D0%B7%D0%BD%D0%BE%D0%B2%D0%B5%D0%BD%D0%B8%D0%B5_%D0%BF%D0%BE%D1%80%D1%8F%D0%B4%D0%BA%D0%B0), более целесообразным представляется переход к формату с более широким диапазоном [порядков](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) либо использование специальных мер для контроля за величиной [порядка](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), чем рассчитывать на денормализованные числа. К тому же нужно помнить, что денормализованные числа имеют меньше значащих цифр [мантиссы](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) по сравнению с обычным для данного формата, а это чревато существенной потерей точности.

**23. Погрешности обусловленные форматом с плавающей точкой**

**Абсолютная погрешность представления**- это*разность между истинным значением входной величины А и ее значением, полученным из машинного изображения*Ам, т.е. [A] =A-Aм

Тогда **относительная погрешность представления**равна: [A] = [A]/Av

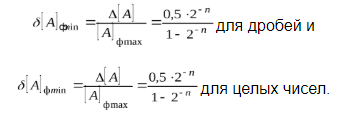
*Максимальная погрешность перевода десятичной информации в двоичную не превышает единицы младшего разряда разрядной сетки цифрового автомата. Минимальная погрешность перевода равна нулю.*

**Усредненная абсолютная погрешность**перевода чисел в двоичную систему счисления будет равна:

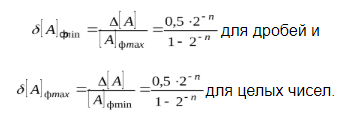
[A]с=0.52^-n – для дробей

[A]c=-0.52^0 – для целых чисел

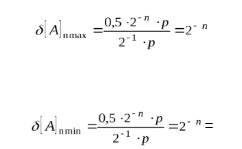
 Относительные погрешности представления для минимального значения числа равны:



Аналогично для максимального значения:



Так как погрешность представления числа зависит только от разрядности мантиссы, то для нахождения погрешности представления числа в форме с плавающей запятой величину этой погрешности надо умножить на величину порядка числа *p*:



где *n*- количество разрядов для представления мантиссы числа. Из последней формулы следует, что относительная точность представления чисел в форме с плавающей запятой почти не зависит от величины числа.

**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. Эти компоненты могут пропускать или не пропускать ток, иметь на выходе высокий или низкий уровень сигнала (напряжения или тока).

Приведем основные понятия алгебры логики.

**Логическая переменная** — это такая переменная, которая может принимать одно из двух значений: истинно или ложно (да или нет, единица или ноль).

**Логическая константа** — это такая постоянная величина, значением которой может быть истинно или ложно (да или нет, единица или ноль).

**Логическая функция** — это такая функция, которая может принимать одно из двух значений: истинно или ложно (да или нет, единица или ноль) в зависимости от текущих значений ее аргументов, в качестве которых используются логические переменные.

Логическая функция может быть одного **(n = 1)** или нескольких **(n > 2)** аргументов. Значение логической функции определяется комбинацией конкретных значений переменных, от которых она зависит. Комбинация конкретных значений переменных (аргументов функции) называется набором. Количество различных наборов N для «n» переменных вычисляется по формуле **N = 2^n.**

Зависимость логической функции от переменных может задаваться по–разному:

– словесным описанием;

– таблицей истинности;

– логическим выражением.

Словесное описание используется в случае сравнительно несложной логической функции.

Таблица истинности является универсальным средством задания логической функции. Она включает все наборы для заданного количества переменных, определяющих значение логической функции, с указанием значений, которые принимает функция для каждого набора. В одной таблице истинности может задаваться несколько логических функций, зависящих от одних и тех же переменных. Таблица истинности для нескольких функций y трёх переменных **х1, х2, х3** может быть задана следующим образом (табл. 2.1)

Таблица 2.1

**Таблица истинности трех переменных**

|  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- |
| № | х1 | х2 | х3 | y1 | y2 | y3 | … | yn |
| 0 | 0 | 0 | 0 | 0 | 1 | 1 |  | 0 |
| 1 | 0 | 0 | 1 | 1 | 1 | 0 |  | 1 |
| 2 | 0 | 1 | 0 | 1 | 1 | 1 |  | 0 |
| 3 | 0 | 1 | 1 | 0 | 1 | 0 |  | – |
| 4 | 1 | 0 | 0 | 1 | 0 | 1 |  | 0 |
| 5 | 1 | 0 | 1 | 0 | 0 | 0 |  | 1 |
| 6 | 1 | 1 | 0 | 0 | 0 | 1 |  | – |
| 7 | 1 | 1 | 1 | 1 | 1 | 0 |  | 1 |

В приведенной таблице истинности во второй, третей и четвертой колонках, помеченных соответственно х1, х2, х3 , приведены все возможные наборы этих переменных. В следующих колонках приводятся значения функций y1, y2, yn для каждого набора.

Логическая функция называется «полностью определенной», если для нее заданы значения по всем возможным наборам. Функция называется «частично определенной», если для некоторых наборов значения функции не заданы. В приведенной таблице истинности функции y1, y2, y3 являются полностью определенными, а функция **yn – частично определенная** (знак «–» означает неопределенность значения функции).

**Максимальное количество полностью определенных функций** от «n» переменных определяется как **M = (2^2)^n .**

**Логическим выражением** называется комбинация логических переменных и констант, связанных элементарными базовыми логическими функциями (или логическими операциями), которые могут разделяться скобками.

**Например**, логическую функцию у1, определенную в вышеприведенной таблице истинности, можно представить в виде логического выражения

**Набор элементарных логических операций**, с помощью которых можно задать любую, сколь угодно сложную логическую функцию, **называется функционально полной системой логических функций**. Иногда такую систему называют базисом.

В качестве элементарных логических функций функционально полных систем этих функций используются функции одной или двух логических переменных.

Все возможные функции одной переменной приведены в табл. 2.2.

Таблица 2.2

**Функции одной переменной**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| x | y0 | y1 | y2 | y3 |
| 0 | 0 | 0 | 1 | 1 |
| 1 | 0 | 1 | 0 | 1 |

Из таблицы видно, что:

y0 = 0 – константа; y1 равна значению переменной; y2 равна значению, обратному значению переменной « х »; y3 = 1 – константа.

С точки зрения базовых функций интерес представляет только функция y2, она называется функцией отрицания, читается как «не х» и обозначается как «», т. е. можно записать y2 =.

Все возможные функции двух переменных приведены в табл. 2.3

Таблица 2.3

**Функции двух переменных**

|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| № | х1 | х2 | y0 | y1 | y2 | y3 | y4 | y5 | y6 | y7 | y8 | y9 | y10 | y11 | y12 | y13 | y14 | y15 |
| 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 |
| 1 | 0 | 1 | 0 | 0 | 0 | 0 | 1 | 1 | 1 | 1 | 0 | 0 | 0 | 0 | 1 | 1 | 1 | 1 |
| 2 | 1 | 0 | 0 | 0 | 1 | 1 | 0 | 0 | 1 | 1 | 0 | 0 | 1 | 1 | 0 | 0 | 1 | 1 |
| 3 | 1 | 1 | 0 | 1 | 0 | 1 | 0 | 1 | 0 | 1 | 0 | 1 | 0 | 1 | 0 | 1 | 0 | 1 |

**25.Законы и правила алгебры Буля**

В алгебре Буля логические выражения включают логические операции И, ИЛИ, НЕ, которые могут быть использованы в самых различных сочетаниях. При оценке значения такого выражения необходимо решить его для конкретного набора переменных. В алгебре Буля применяется следующая приоритетность выполнения операций: сначала рассчитываются значения имеющих место отрицаний и скобок, затем выполняется операция И (логическое умножение); самый низший приоритет имеет операция ИЛИ (логическая сумма).

При работе с булевыми логическим выражениями используются следующие законы, правила и операции.

***Переместительный* (коммутативный) *закон.***

Закон справедлив как для конъюнкции, так и для дизъюнкции.

– 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

х1х2х3х4 = х4х3х2х1

Этот закон справедлив для любого количества логических операндов.

***Сочетательный* (ассоциативный) *закон.***

Справедлив как для конъюнкции, так и для дизъюнкции.

– при логическом сложении отдельные слагаемые можно заменить их суммой

– 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



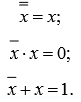
Если в качестве А используется простая конъюнкция, т. е. конъюнкция, представляющая собой логическое произведение переменных и их отрицаний, то имеет место



Как видно, в результирующем выражении количество переменных на единицу меньше, чем в склеенных конъюнкциях. Количество переменных в простой конъюнкции называется рангом конъюнкции, т. е. операция склеивания, примененная к простым конъюнкциям, дает результат с рангом, на единицу меньшим ранга исходных конъюнкций.

***Операции с отрицаниями.***

– двойное отрицание равносильно отсутствию отрицания



***Операции с константами.***



***Операции с одинаковыми операндами.***





***Функционально полной системой булевых функций*** (ФПСБФ) называется совокупность таких булевых функций (f1, f2, ..., fk), посредством которых можно записать произвольную булеву функцию f. Как уже было сказано, ФПСБФ являются «Стрелка Пирса» и «Штрих Шеффера».

**26.Понятие о принципе двойственности. Суперпозиция логических**

В булевых алгебрах существуют двойственные утверждения, которые либо одновременно верны, либо одновременно неверны. Именно, если в формуле, которая верна в некоторой булевой алгебре, поменять все конъюнкции на дизъюнкции, 0 на 1, ≤ на ≥ и наоборот, то получится формула также истинная в этой булевой алгебре. Это следует из симметричности аксиом относительно таких замен.

Пусть https://konspekta.net/lektsiiorgimg/baza14/2117058209327.files/image611.png ― произвольная формула булевой алгебры, содержащая булевы переменные https://konspekta.net/lektsiiorgimg/baza14/2117058209327.files/image612.png .

Формула

https://konspekta.net/lektsiiorgimg/baza14/2117058209327.files/image613.png

называется двойственной формуле https://konspekta.net/lektsiiorgimg/baza14/2117058209327.files/image611.png

Для двойственныхформул справедлив*принцип двойственности.*Если

https://konspekta.net/lektsiiorgimg/baza14/2117058209327.files/image614.png

то и обязательно

https://konspekta.net/lektsiiorgimg/baza14/2117058209327.files/image615.png

а также, если

https://konspekta.net/lektsiiorgimg/baza14/2117058209327.files/image616.png

то и обязательно

https://konspekta.net/lektsiiorgimg/baza14/2117058209327.files/image617.png

Так как любая формула булевой алгебры определяет булеву функцию, то понятие двойственности можно использовать и для функций.

Булева функция https://konspekta.net/lektsiiorgimg/baza14/2117058209327.files/image618.png называется двойственной функции https://konspekta.net/lektsiiorgimg/baza14/2117058209327.files/image619.png , если

https://konspekta.net/lektsiiorgimg/baza14/2117058209327.files/image620.png .

Пусть имеется некоторый набор *K*, состоящий из конечного числа булевых функций. Суперпозицией функций из этого набора называются новые функции, полученные с помощью конечного числа применения двух операций;

Суперпозицию еще иначе называют сложной функцией.

**27.Полная система логических функций.**

Система логических функций называется Функционально полной системой, если любая логическая функция может быть выражена через функции с помощью их суперпозиции.

Очевидно, что любая система, через функции которой можно выразить конъюнкцию, дизъюнкцию и отрицание, будет полной.

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

Существует 2 формы представления логических функций:

- дизъюнктивная нормальная форма (ДНФ);

- конъюнктивная нормальная форма (КНФ).

Дизъюнктивная нормальная форма (ДНФ)– это форма представления логической функции в виде дизъюнкции ряда членов, каждый из которых представляет собой простую конъюнкцию аргументов или инверсий аргументов.

http://ok-t.ru/studopediaru/baza3/660670099329.files/image183.gif

Совершенная ДНФ – (СДНФ) - это ДНФ, в каждом члене которой присутствуют все аргументы.

http://ok-t.ru/studopediaru/baza3/660670099329.files/image185.gif

Чтобы получить СДНФ функции заданной таблицей необходимо записать столько дизъюнктивных членов, сколько единиц содержит функция в таблице. Для каждой единицы записать простую конъюнкцию аргументов, или инверсии аргументов, если их значения равны нулю.

http://ok-t.ru/studopediaru/baza3/660670099329.files/image187.gifhttp://ok-t.ru/studopediaru/baza3/660670099329.files/image189.gif

Конъюнктивная нормальная форма (КНФ) - это форма представления логических функций в виде конъюнкции ряда членов, каждый из которых представляет собой простую дизъюнкцию аргументов или инверсий аргументов.

http://ok-t.ru/studopediaru/baza3/660670099329.files/image191.gif

Совершенная КНФ-(СКНФ)-это КНФ, в каждом члене которой присутствуют все аргументы.

http://ok-t.ru/studopediaru/baza3/660670099329.files/image193.gif

Чтобы получить СКНФ функции, заданной таблицей, необходимо записать столько конъюнктивных членов, сколько нулей содержит функция в таблице. Для каждого нуля записать простую дизъюнкцию аргументов или инверсий аргументов, если их значения равны единице.

http://ok-t.ru/studopediaru/baza3/660670099329.files/image195.gif

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

Индексом простоты (или сложностью) булевой функции называется функция https://studfile.net/html/2706/241/html_56Y_DPoQff.stbV/img-WbyoqT.png, определенная на множестве всех ДНФ и удовлетворяющая следующим свойствам:

1. Свойство неотрицательности: https://studfile.net/html/2706/241/html_56Y_DPoQff.stbV/img-4KaHGD.pnghttps://studfile.net/html/2706/241/html_56Y_DPoQff.stbV/img-w5XRGB.png;
2. Свойство монотонности: если https://studfile.net/html/2706/241/html_56Y_DPoQff.stbV/img-RfCTHq.png, где https://studfile.net/html/2706/241/html_56Y_DPoQff.stbV/img-0qu7Rx.png -элементарная конъюнкция, https://studfile.net/html/2706/241/html_56Y_DPoQff.stbV/img-6CRrHC.png.
3. Свойство выпуклости: https://studfile.net/html/2706/241/html_56Y_DPoQff.stbV/img-kVHWXv.png
4. Свойство инвариантности: если ДНФ =https://studfile.net/html/2706/241/html_56Y_DPoQff.stbV/img-1bqohF.png получена из ДНФ =https://studfile.net/html/2706/241/html_56Y_DPoQff.stbV/img-E9R21n.png путем переименования переменных без отождествления, то https://studfile.net/html/2706/241/html_56Y_DPoQff.stbV/img-RZs5pL.png.

## **Метод неопределенных коэффициентов**

Метод применим для функций от любого числа переменных. Метод неопределенных коэффициентов эффективен, если число аргументов функции не больше, чем 5 – 6. Это связано с тем, что число уравнений равно 2n. Более эффективным является выписывание не всех возможных конъюнкций для функции, а только тех, которые могут присутствовать в ДНФ данной функции.

**Метод Квайна**

Выписывание не всех возможных конъюнкций для функции, а только тех, которые могут присутствовать в ДНФ данной функции. При этом предполагается, что функция задана в виде СДНФ. В данном методе элементарные конъюнкции рангаn, входящие в ДНф, называются минитермами рангаn. Метод Квайна состоит из последовательного выполнения следующих этапов:

1)Нахождение первичных импликант

2)Расстановка меток

3)Нахождение существенных импликант

4)Вычеркивание лишних столбцов и строк

5)Выбор минимального покрытия максимальными интервалами

# **Метод Петрика**

Метод Петрика позволяет исключить процедуру построения и анализа таблицы меток. После получения всех первичных импликант, для каждого минитерма строится дизъюнкция из тех импликант, которые входят в данный минитерм. Затем все дизъюнкции логически перемножаются, раскрываются скобки, выполняются все поглощения, т. е. получается некоторая ДНФ. При этом, каждая конъюнкция, входящая в ДНФ, соответствует одной из тупиковых ДНФ. Выбирается самая короткая, которая соответствует МДНФ.

**Метод Блека**

Метод Блека - Порецкого позволяет строить Сокр. ДНФ не по СДНФ, а по произвольной ДНФ.

**30. Минимизация логических выражений методом Квайна.**

В качестве исходной формы представления логического выражения используется СДНФ. Если подлежащее минимизации выражение имеет другую форму, то приведение к СДНФ осуществляется за счет открытия скобок, избавления от отрицаний логических выражений, более сложных чем отрицание переменной (используется правило де Моргана).

Метод Квайна выполняется в два этапа.

***Первый этап*** имеет своей целью получение тупиковой формы, представляющей собой дизъюнкцию, в качестве слагаемых которой используются конъюнкции (каждая из них не склеивается ни с одной другой конъюнкцией, входящей в это выражение). Такие конъюнкции называются *простыми импликантами*.

Данный этап выполняется за счет реализации отдельных шагов. На каждом шаге на основании выражения, полученного на предыдущем шаге, выполняются все возможные операции склеивания для пар имеющихся конъюнкций. Каждый шаг понижает ранг исходных конъюнкций на единицу. Шаги повторяются до получения тупиковой формы.

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

Над конъюнкциями проставлены их номера; в скобках под каждой конъюнкцией (*i–j*) указывают, что данная конъюнкция является результатом склеивания *i-*й и *j-*й конъюнкций исходного выражения.

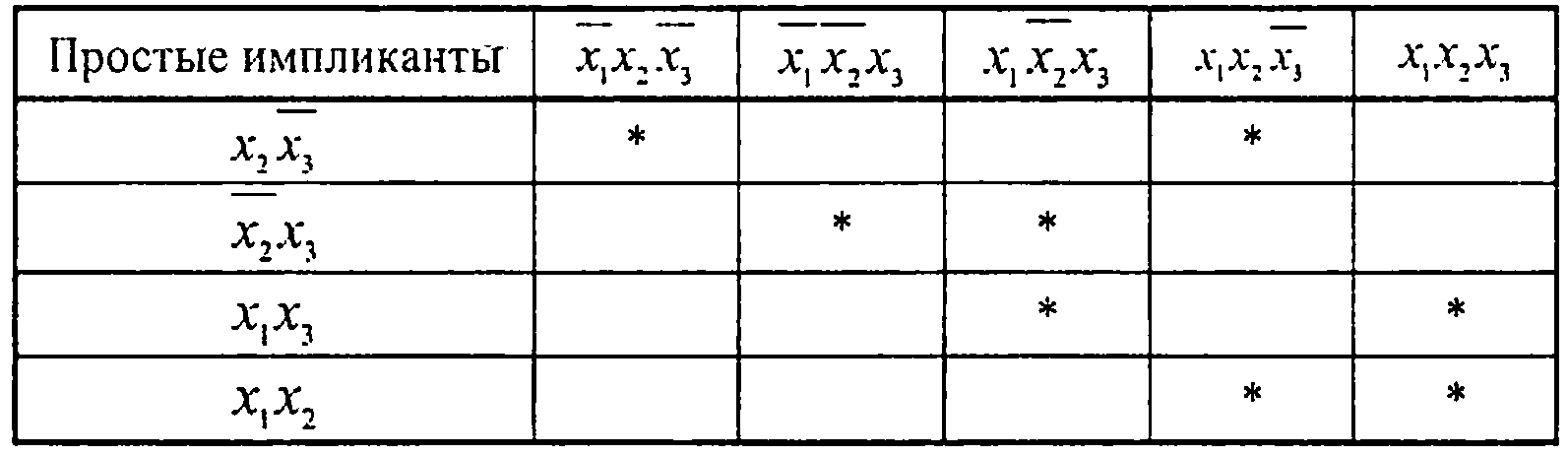
К результатам склеивания логически добавлен ни с чем не склеенный пятый член исходного выражения; несколько одинаковых конъюнкций представляются одной конъюнкцией.

Последнее выражение получено из предыдущего посредством удаления повторяющихся членов.

2*-*й этап:

На основании исходного выражения и полученной тупиковой формы составляется и заполняется импликантная таблица (табл.2.7).

**Импликантная таблица**



Колонки приведенной таблицы помечены конституентами единицы, имеющимися в исходном логическом выражении.

Строки таблицы помечены простыми импликантами полученной тупиковой формы.

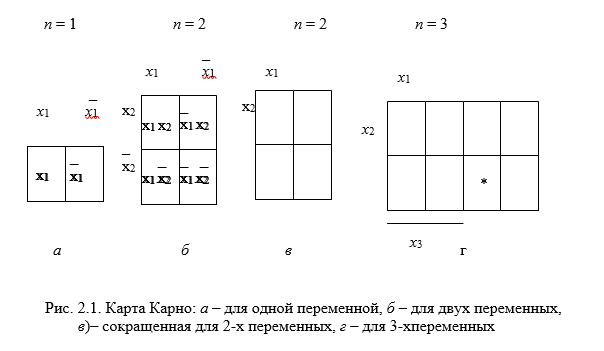
Звездочками в каждой строке отмечены те конституенты единицы, которые покрываются соответствующей простой импликантой (практически отмечаются те конституенты единицы, которые включают простую импликанту как свою составную часть)

**31. Минимизация логических выражений с 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. Каждая такая клетка соответствует одному набору логических переменных, причем наборы двух соседних клеток должны отличаться на значение одной переменной (их наборы образуют склеивающиеся конъюнкции).

На рис. 2.1 приведены карты Карно для *n* = 1, 2, 3. На рис.2.1, *а*, 2.1,*б* показана разметка колонок и строк, а также указан для каждой составляющей клетки соответствующие ей набор. Разметка колонок (строк) указывает, какие значения данная переменная имеет в клетках, находящихся в данной колонке (строке). На рис. 2.1,*в* приведен пример компактной разметки карты, соответствующей карте на рис.2.1, *б*. Здесь помечаются колонки (строки), в которых соответствующая переменная имеет прямое значение. На рис. 2.1, *г* приведена карта Карно для *n* = 3, сформированная посредством зеркального отображения карты Карно для *n* = 2 (рис. 2.1, *в*) относительно правой границы. Этот прием универсальный; его можно использовать для построения карты для заданного «*n*» на основании имеющейся карты Карно для «n ˗ 1» переменной. Клетка, отмеченная знаком «\*», соответствует набору . Карты Карно используются для представления и минимизации логических функций.



Записываемая функция должна быть представлена в СДНФ. Запись функции в карту осуществляется за счет установки «1» в клетки карты, соответствующие конституентам единиц записываемой функции. 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.

Для того чтобы быть логическими соседями, клеткам достаточно быть геометрическими соседями. Считая, что карта является пространственным объектом и заворачивается по горизонтали и вертикали, сливаясь своими крайними горизонтальными и крайними вертикальными границами, можно считать, что соответствующие крайние горизонтальные и вертикальные клетки являются геометрическими соседями. Логическими соседями могут быть клетки, которые не являются геометрическими соседями. К числу таких клеток относятся клетки, которые по горизонтали или вертикали симметричны относительно линий зеркального отображения, которые были использованы при переходе от «*n*» к «*n*+1» переменным.

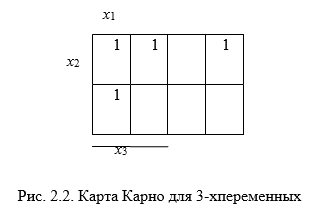
Запись минимального выражения по заданной функции имеет вид дизъюнкции простых конъюнкций, соответствующих контурам на карте, и формируется следующим образом:

- 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.

Например, если задана логическая функция «*y*» трех переменных в виде выражения

то её запись в карту Карно будет иметь вид, приведенный на рис.2.2.



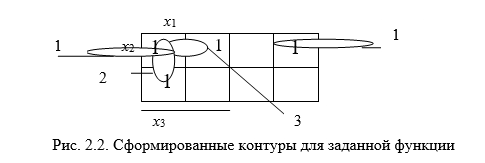
Для функции, заданной в карте Карно, приведенной на рис.2.2, контуры имеют вид, приведенный на рис.2.3.

Для примера, контур 1 представлен на рисунке в виде двух клеток: клетки, соответствующей набору *x*1*x*2*x*3, и клетки, соответствующей набору , поэтому данному контуру будет соответствовать конъюнкция x1 x2.

Минимальное логическое выражение для функции имеет вид:

*y* = *x*1 *x*2+ *x*1 *x*3 + *x*2 *x*3.

1 2 3



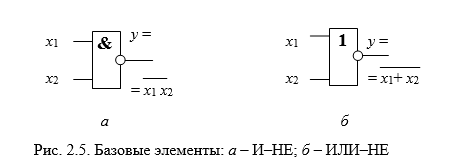
Конъюнкции минимального выражения помечены внизу цифрами, соответствующими номерам контуров, которые они представляют.

**32. Логический базис И-НЕ. Синтез логических схем по логическому выражению в базисе И-НЕ.**

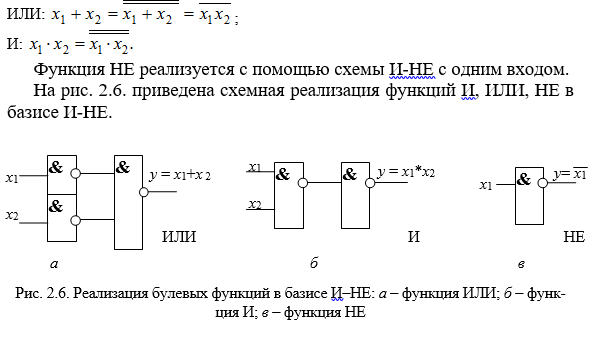
Булевый базис не является единственной функционально полной системой логических функций. Среди других наибольшее распространение получили базис И–НЕ и базис ИЛИ–НЕ.

Чтобы доказать логическую полноту любого базиса, достаточно показать, что в этом базисе можно реализовать базовые функции И, ИЛИ, НЕ.

Для базиса И-НЕ в качестве базового элемента используется элемент приведенный на рисунке рис. 2.5,а.



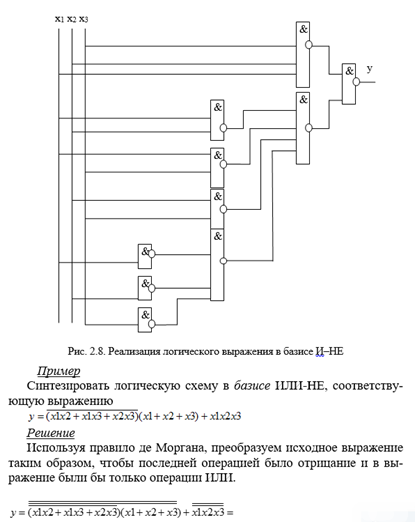
Реализация с помощью функции И-НЕ базовых функций алгебры Буля осуществляется следующим образом.

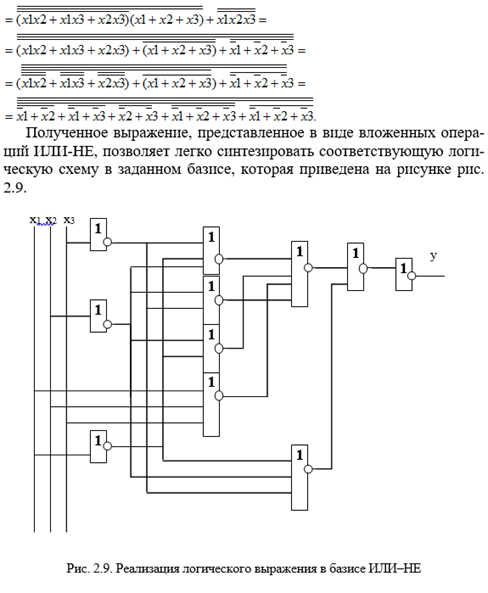


Реализация с помощью логической функции ИЛИ-НЕ базовых функций алгебры Буля осуществляется следующим образом.



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.





**33.Код Грея. Обратная польская запись**

**Код Гре́я** — [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), иначе зеркальный код, он же код с отражением, 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.

Наиболее часто на практике применяется рефлексивный [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) Грея, хотя в общем случае существует бесконечное множество кодов Грея со значениями цифр в разрядах, взятых из различных алфавитов. В большинстве случаев, под термином «код Грея» понимают именно рефлексивный бинарный код Грея.

Изначально предназначался для защиты от ложного срабатывания электромеханических переключателей. Сегодня коды Грея широко используются 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.

**Обра́тная по́льская 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)). Также именуется как обратная бесскобочная запись, постфиксная нотация, бесскобочная символика Лукасевича, польская инверсная запись, ПОЛИЗ.

Отличительной особенностью обратной польской нотации является то, что все аргументы (или [операнды](https://ru.wikipedia.org/wiki/%D0%9E%D0%BF%D0%B5%D1%80%D0%B0%D0%BD%D0%B4)) расположены перед знаком операции.

**34. АЦП и ЦАП**

Цифро-аналоговые преобразователи (ЦАП) служат для преобразования информации из цифровой формы в аналоговый сигнал. ЦАП широко применяется в различных устройствах автоматики для связи контроллеров, вырабатывающих сигналы управления в виде цифрового кода, с аналоговыми элементами системы.

Принцип работы ЦАП состоит в суммировании аналоговых сигналов, пропорциональных весам разрядов входного цифрового кода, с коэффициентами, равными нулю или единице в зависимости от значения соответствующего разряда кода.

ЦАП преобразует цифровой двоичный код а0, а1, а2, .. ап-1 в аналоговую величину, обычно напряжение Uвых.. Каждый разряд двоичного кода имеет определенный вес i-го разряда вдвое больше, чем вес (i-1)-го. Работу ЦАП можно описать следующей формулой:

https://siblec.ru/img/105/02.files/image121.png,

где e = https://siblec.ru/img/105/02.files/image122.png - напряжение, соответствующее весу младшего разряда, аi - значение i -го разряда двоичного кода (0 или 1).

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)).

Обратное преобразование осуществляется при помощи [цифро-аналогового преобразователя](https://ru.wikipedia.org/wiki/%D0%A6%D0%B8%D1%84%D1%80%D0%BE-%D0%B0%D0%BD%D0%B0%D0%BB%D0%BE%D0%B3%D0%BE%D0%B2%D1%8B%D0%B9_%D0%BF%D1%80%D0%B5%D0%BE%D0%B1%D1%80%D0%B0%D0%B7%D0%BE%D0%B2%D0%B0%D1%82%D0%B5%D0%BB%D1%8C)

Как правило, АЦП — [электронное](https://ru.wikipedia.org/wiki/%D0%AD%D0%BB%D0%B5%D0%BA%D1%82%D1%80%D0%BE%D0%BD%D0%B8%D0%BA%D0%B0) устройство, преобразующее [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. Тем не менее, некоторые неэлектронные устройства с цифровым выходом следует также относить к АЦП, например, некоторые типы [преобразователей угол-код](https://ru.wikipedia.org/wiki/%D0%94%D0%B0%D1%82%D1%87%D0%B8%D0%BA_%D1%83%D0%B3%D0%BB%D0%B0_%D0%BF%D0%BE%D0%B2%D0%BE%D1%80%D0%BE%D1%82%D0%B0). Простейшим одноразрядным двоичным АЦП является [компаратор](https://ru.wikipedia.org/wiki/%D0%9A%D0%BE%D0%BC%D0%BF%D0%B0%D1%80%D0%B0%D1%82%D0%BE%D1%80).

**35. Искусство управления сложностью. цифровая абстракция.**

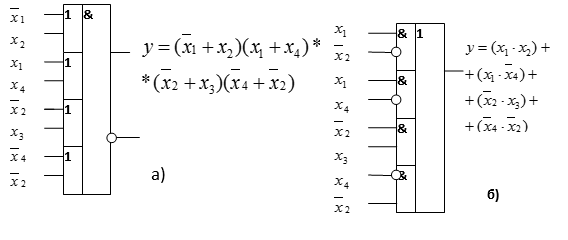
**36.Логические элементы.**

Логические элементы — устройства, предназначенные для [обработки информации](https://ru.wikipedia.org/w/index.php?title=%D0%9E%D0%B1%D1%80%D0%B0%D0%B1%D0%BE%D1%82%D0%BA%D0%B0_%D0%B8%D0%BD%D1%84%D0%BE%D1%80%D0%BC%D0%B0%D1%86%D0%B8%D0%B8&action=edit&redlink=1) в [цифровой форме](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) (последовательности сигналов высокого — «1» и низкого — «0» уровней в [двоичной логике](https://ru.wikipedia.org/wiki/%D0%94%D0%B2%D0%BE%D0%B8%D1%87%D0%BD%D0%B0%D1%8F_%D0%BB%D0%BE%D0%B3%D0%B8%D0%BA%D0%B0), последовательность «0», «1» и «2» в [троичной логике](https://ru.wikipedia.org/wiki/%D0%A2%D1%80%D0%BE%D0%B8%D1%87%D0%BD%D0%B0%D1%8F_%D0%BB%D0%BE%D0%B3%D0%B8%D0%BA%D0%B0), последовательностями «0», «1», «2», «3», «4», «5», «6», «7», «8» и «9» в [десятичной логике](https://ru.wikipedia.org/w/index.php?title=%D0%94%D0%B5%D1%81%D1%8F%D1%82%D0%B8%D1%87%D0%BD%D0%B0%D1%8F_%D0%BB%D0%BE%D0%B3%D0%B8%D0%BA%D0%B0&action=edit&redlink=1)). Физически логические элементы могут быть выполнены механическими, электромеханическими (на [электромагнитных реле](https://ru.wikipedia.org/wiki/%D0%AD%D0%BB%D0%B5%D0%BA%D1%82%D1%80%D0%BE%D0%BC%D0%B0%D0%B3%D0%BD%D0%B8%D1%82%D0%BD%D0%BE%D0%B5_%D1%80%D0%B5%D0%BB%D0%B5)), электронными (в частности, на [диодах](https://ru.wikipedia.org/wiki/%D0%94%D0%B8%D0%BE%D0%B4) или [транзисторах](https://ru.wikipedia.org/wiki/%D0%A2%D1%80%D0%B0%D0%BD%D0%B7%D0%B8%D1%81%D1%82%D0%BE%D1%80)), [пневматическими, гидравлическими](https://ru.wikipedia.org/wiki/%D0%A1%D1%82%D1%80%D1%83%D0%B9%D0%BD%D0%B0%D1%8F_%D0%BB%D0%BE%D0%B3%D0%B8%D0%BA%D0%B0), [оптическими](https://ru.wikipedia.org/wiki/%D0%A4%D0%BE%D1%82%D0%BE%D0%BD%D0%BD%D0%B0%D1%8F_%D0%BB%D0%BE%D0%B3%D0%B8%D0%BA%D0%B0) и другими.

Можно выделить три основные разновидности элементов – логические элементы, запоминающие, специальные.

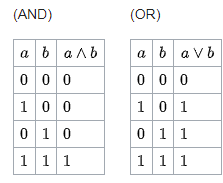
Логические элементы, так же как и элементы алгебры логики, реализуют логические функции, но эти функции, оставаясь сравнительно простыми, все же сложней, чем базовые функции в алгебре логики. В одном логическом элементе может быть реализовано несколько простых функций. Кроме того, логические элементы характеризуются дополнительными параметрами, такими, как количество входов, нагрузочная способность (количество входов других элементов, к которым можно подключать выход данного элемента).

На рис. 1.1. приведены примеры некоторых логических элементов.



а – ИЛИ-И-НЕ; б – И-ИЛИ-НЕ

**Таблица истинности** — таблица, описывающая логическую функцию. Под «логической функцией» в данном случае понимается функция, у которой значения переменных (параметров функции) и значение самой функции выражают логическую истинность. Например, в двузначной логике они могут принимать значения «истина» либо «ложь»



**37. За пределами цифровой абстракции. Напряжение питания. Логические уровни. Допускаемые уровни шумов. Передаточная характеристика. Статическая дисциплина**

**38-39 Биполярные и КМОП транзисторы. Полупроводники. Конденсаторы. n-МОП и p-МОП-транзисторы**

**Биполя́рный транзи́стор** — трёхэлектродный [полупроводниковый прибор](https://ru.wikipedia.org/wiki/%D0%9F%D0%BE%D0%BB%D1%83%D0%BF%D1%80%D0%BE%D0%B2%D0%BE%D0%B4%D0%BD%D0%B8%D0%BA%D0%BE%D0%B2%D1%8B%D0%B5_%D0%BF%D1%80%D0%B8%D0%B1%D0%BE%D1%80%D1%8B), один из типов [транзисторов](https://ru.wikipedia.org/wiki/%D0%A2%D1%80%D0%B0%D0%BD%D0%B7%D0%B8%D1%81%D1%82%D0%BE%D1%80). В полупроводниковой структуре сформированы два [p-n-перехода](https://ru.wikipedia.org/wiki/P-n-%D0%BF%D0%B5%D1%80%D0%B5%D1%85%D0%BE%D0%B4), перенос [заряда](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%B8%D0%B9_%D0%B7%D0%B0%D1%80%D1%8F%D0%B4) через которые осуществляется носителями двух полярностей — [электронами](https://ru.wikipedia.org/wiki/%D0%AD%D0%BB%D0%B5%D0%BA%D1%82%D1%80%D0%BE%D0%BD) и [дырками](https://ru.wikipedia.org/wiki/%D0%94%D1%8B%D1%80%D0%BA%D0%B0). Именно поэтому прибор получил название «биполярный» (от [англ.](https://ru.wikipedia.org/wiki/%D0%90%D0%BD%D0%B3%D0%BB%D0%B8%D0%B9%D1%81%D0%BA%D0%B8%D0%B9_%D1%8F%D0%B7%D1%8B%D0%BA) bipolar), в отличие от [полевого (униполярного) транзистора](https://ru.wikipedia.org/wiki/%D0%9F%D0%BE%D0%BB%D0%B5%D0%B2%D0%BE%D0%B9_%D1%82%D1%80%D0%B0%D0%BD%D0%B7%D0%B8%D1%81%D1%82%D0%BE%D1%80).

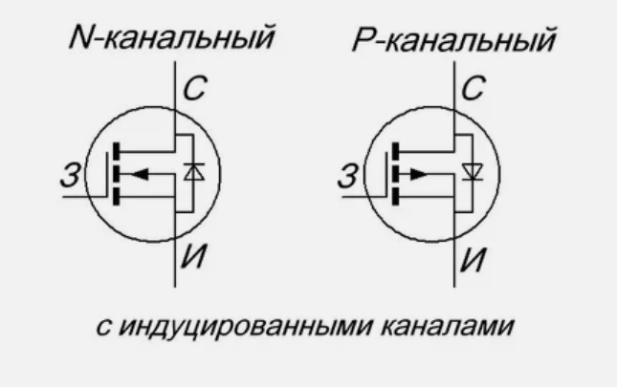
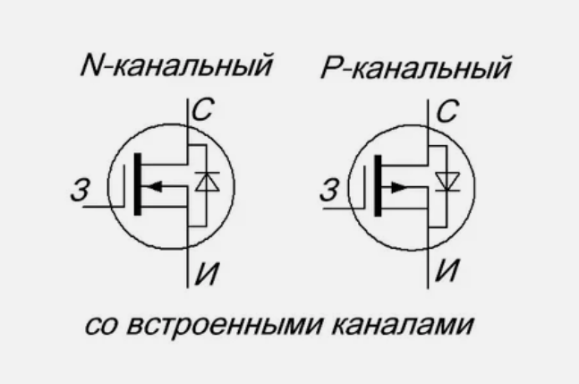
Применяется в электронных устройствах для усиления или генерации электрических колебаний, а также в качестве коммутирующего элемента

**КМОП** -  набор [полупроводниковых технологий](https://ru.wikipedia.org/wiki/%D0%A2%D0%B5%D1%85%D0%BD%D0%BE%D0%BB%D0%BE%D0%B3%D0%B8%D1%87%D0%B5%D1%81%D0%BA%D0%B8%D0%B9_%D0%BF%D1%80%D0%BE%D1%86%D0%B5%D1%81%D1%81_%D0%B2_%D1%8D%D0%BB%D0%B5%D0%BA%D1%82%D1%80%D0%BE%D0%BD%D0%BD%D0%BE%D0%B9_%D0%BF%D1%80%D0%BE%D0%BC%D1%8B%D1%88%D0%BB%D0%B5%D0%BD%D0%BD%D0%BE%D1%81%D1%82%D0%B8) построения [интегральных микросхем](https://ru.wikipedia.org/wiki/%D0%98%D0%BD%D1%82%D0%B5%D0%B3%D1%80%D0%B0%D0%BB%D1%8C%D0%BD%D0%B0%D1%8F_%D0%BC%D0%B8%D0%BA%D1%80%D0%BE%D1%81%D1%85%D0%B5%D0%BC%D0%B0) и соответствующая ей [схемотехника](https://ru.wikipedia.org/wiki/%D0%A1%D1%85%D0%B5%D0%BC%D0%BE%D1%82%D0%B5%D1%85%D0%BD%D0%B8%D0%BA%D0%B0) микросхем. Подавляющее большинство современных цифровых микросхем — КМОП. В технологии КМОП используются [полевые транзисторы](https://ru.wikipedia.org/wiki/%D0%9F%D0%BE%D0%BB%D0%B5%D0%B2%D0%BE%D0%B9_%D1%82%D1%80%D0%B0%D0%BD%D0%B7%D0%B8%D1%81%D1%82%D0%BE%D1%80) с изолированным затвором с каналами разной проводимости. Отличительной особенностью схем КМОП по сравнению с биполярными технологиями  является очень малое энергопотребление в статическом режиме. Отличительной особенностью структуры КМОП по сравнению с другими МОП-структурами является наличие как n-, так и p-канальных полевых транзисторов в одной локации кристалла; вследствие меньшего расстояния между элементами КМОП-схемы обладают более высокой скоростью действия и меньшим энергопотреблением, однако при этом характеризуются более сложным технологическим процессом изготовления и меньшей плотностью упаковки.

**Полупроводни́к** — [материал](https://ru.wikipedia.org/wiki/%D0%9C%D0%B0%D1%82%D0%B5%D1%80%D0%B8%D0%B0%D0%BB), по [удельной проводимости](https://ru.wikipedia.org/wiki/%D0%AD%D0%BB%D0%B5%D0%BA%D1%82%D1%80%D0%B8%D1%87%D0%B5%D1%81%D0%BA%D0%B0%D1%8F_%D0%BF%D1%80%D0%BE%D0%B2%D0%BE%D0%B4%D0%B8%D0%BC%D0%BE%D1%81%D1%82%D1%8C) занимающий промежуточное место между [проводниками](https://ru.wikipedia.org/wiki/%D0%9F%D1%80%D0%BE%D0%B2%D0%BE%D0%B4%D0%BD%D0%B8%D0%BA_(%D1%8D%D0%BB%D0%B5%D0%BA%D1%82%D1%80%D0%B8%D1%87%D0%B5%D1%81%D1%82%D0%B2%D0%BE)) и [диэлектриками](https://ru.wikipedia.org/wiki/%D0%94%D0%B8%D1%8D%D0%BB%D0%B5%D0%BA%D1%82%D1%80%D0%B8%D0%BA), и отличающийся от проводников сильной зависимостью [удельной проводимости](https://ru.wikipedia.org/wiki/%D0%AD%D0%BB%D0%B5%D0%BA%D1%82%D1%80%D0%B8%D1%87%D0%B5%D1%81%D0%BA%D0%B0%D1%8F_%D0%BF%D1%80%D0%BE%D0%B2%D0%BE%D0%B4%D0%B8%D0%BC%D0%BE%D1%81%D1%82%D1%8C) от концентрации примесей, температуры и воздействия различных видов [излучения](https://ru.wikipedia.org/wiki/%D0%98%D0%B7%D0%BB%D1%83%D1%87%D0%B5%D0%BD%D0%B8%D0%B5). Основным свойством полупроводников является увеличение электрической проводимости с ростом температуры

Конденса́тор - устройство для накопления [заряда](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%B8%D0%B9_%D0%B7%D0%B0%D1%80%D1%8F%D0%B4) и энергии электрического поля. нденсатор является пассивным электронным компонентом. [Ёмкость](https://ru.wikipedia.org/wiki/%D0%AD%D0%BB%D0%B5%D0%BA%D1%82%D1%80%D0%B8%D1%87%D0%B5%D1%81%D0%BA%D0%B0%D1%8F_%D1%91%D0%BC%D0%BA%D0%BE%D1%81%D1%82%D1%8C) конденсатора измеряется в [фарадах](https://ru.wikipedia.org/wiki/%D0%A4%D0%B0%D1%80%D0%B0%D0%B4).

**МОП** (металл-оксид-полупроводник) - один из видов полевого [транзистора](https://ru.wikipedia.org/wiki/%D0%A2%D1%80%D0%B0%D0%BD%D0%B7%D0%B8%D1%81%D1%82%D0%BE%D1%80), в котором управляющий электрод (затвор) отделён от канала слоем диэлектрика, в простейшем случае, [диоксида кремния](https://ru.wikipedia.org/wiki/%D0%94%D0%B8%D0%BE%D0%BA%D1%81%D0%B8%D0%B4_%D0%BA%D1%80%D0%B5%D0%BC%D0%BD%D0%B8%D1%8F). Транзисторы МОП-структуры лучше других активных полупроводниковых приборов подходили для создания логических [БИС](https://ru.wikipedia.org/wiki/%D0%91%D0%BE%D0%BB%D1%8C%D1%88%D0%B0%D1%8F_%D0%B8%D0%BD%D1%82%D0%B5%D0%B3%D1%80%D0%B0%D0%BB%D1%8C%D0%BD%D0%B0%D1%8F_%D1%81%D1%85%D0%B5%D0%BC%D0%B0) и СБИС, и ранний прогресс цифровой техники обусловлен микросхемами на транзисторах с МОП-структурой. В отличие от биполярного транзистора, выходной ток которого управляется входным током, МОП-транзистор, как и другие полевые транзисторы, управляется напряжением, этим он напоминает электровакуумный триод. В зависимости от типа носителей зарядов, МОП-транзисторы могут быть n-канальными или p-канальными, в первых используются электроны, во вторых — дырки.



 P-канальный работает точно также, как и N-канальный, но вся разница в том, что основными носителями будут являться уже дырки. В этом случае все напряжения в схеме меняем на инверсные, в отличие от N-канального транзистора

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

Комбинационной схемой (КС) называется схема из логических (переключательных) элементов, реализующая булеву функцию или совокупность булевых функций. Под логическим (переключательным) элементом чаще всего понимают техническое устройство, реализующее одну элементарную булеву функцию.

В ИИС информационный контакт между ЭВМ и объектом измерения осуществляется с помощью датчиков и исполнительных органов, которые подключаются к ЭВМ с помощью специальных технических средств сопряжения. Технические средства сопряжения ЭВМ с объектами включают разнообразный набор преобразователей, из которых строят необходимые подсистемы сопряжения — устройства связи с объектами. Основные функции устройств связи с объектами: преобразование непрерывных (аналоговых) сигналов, поступающих от датчиков в цифровой код; преобразование цифровых кодов ЭВМ в непрерывные аналоговые сигналы для управления исполнительными органами; формирование сигналов дискретных датчиков, ввод их в ЭВМ и вывод их на исполнительные органы и т. д. На АЭС в АСУТП комплексы связи с объектом представляют собой программируемые иерархические комплексы типа м-64 и улу2-эвм, реализованные на базе микропрограммируемых контроллеров мпк. Всего в системе пять таких комплексов. Терминалы связи с операторами представляют собой программируемые комплексы типа РМОТ. В системе предусмотрено семь таких терминалов.

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

К основным параметрам цифровых микросхем относятся :

1. Реализуемая логическая функция или назначение.
2. Нагрузочная способность – характеризуемая коэффициентом разветвления Краз. Он обозначает максимальное число интегральных микросхем, аналогично рассматриваемой, которые допускают подключать одновременно к ее выходу без искажения передачи информации.
3. Мощность, потребляемая микросхемой (среднее значение) – Рср.
4. максимально допустимый ток на выходе микросхемы в режиме логического нуля – I0вых.
5. I0вх – то же на входе.
6. Значение уровня логического нуля на выходе элемента – U0вых.
7. U1вых – то же , логической единицы.
8. Максимальная частота переключений элемента – fмакс.
9. Номинальное напряжение источника питания - Uип.

Задержкой распространения сигнала через элемент/схему называют время между перепадом цифрового сигнала на входе элемента/схемы и вызванным им перепадом сигнала на выходе. Задержка распространения вызвана конечным временем срабатывания транзисторных ключей внутри элемента. Она будет больше, чем больше количество таких ключей по пути распространения сигнала внутри элемента. Задержки распространения для современных интегральных микросхем находятся в диапазоне от десятков наносекунд до десятков пикосекунд

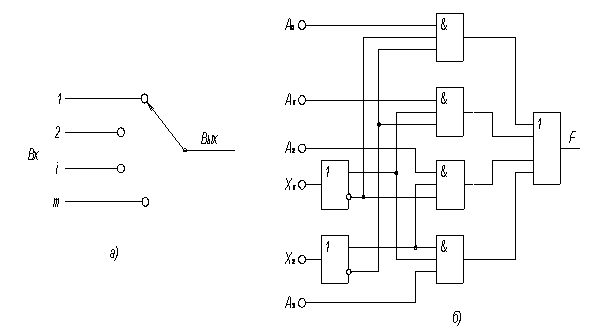
Задержка реакции - с технической точки зрения представляет собой проявление влияния [латентности](https://ru.wikipedia.org/wiki/%D0%9B%D0%B0%D1%82%D0%B5%D0%BD%D1%82%D0%BD%D0%BE%D1%81%D1%82%D1%8C) операций передачи и обработки данных на качество работы [системы реального времени](https://ru.wikipedia.org/wiki/%D0%A0%D0%B5%D0%B0%D0%BB%D1%8C%D0%BD%D0%BE%D0%B5_%D0%B2%D1%80%D0%B5%D0%BC%D1%8F).

Главными источниками импульсных помех являются включения и выключения осветительных и электромеханических бытовых приборов, аварии линий сильного тока, удары молнии и др. Импульсные помехи обычно проявляются как ошибки передачи и могут быть причиной перерывов в линии [DSL](http://www.xdsl.ru/), требующих её повторного запуска retrain. Особенно чувствительны к импульсным помехам видео приложения, где эти помехи могут вызывать потерю синхронизации, «замораживание», пикселизацию изображения и др. Признаком влияния импульсных помех на сигнал [ADSL](http://xdsl.ru/articles/adsl.htm) могут быть также жалобы абонентов на малую скорость сети Интернет.

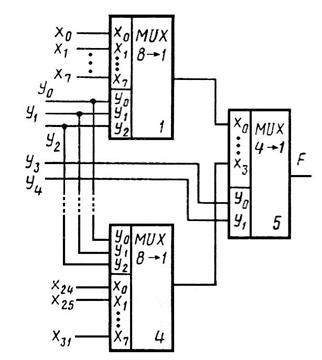
**42. Базовые комбинационные блоки. Мультиплексоры. Логика на мультиплексорах. Дешифраторы**

Узел ЭВМ представляют собой совокупность нескольких логических схем и, в общем случае, элементов памяти, формирующих выходные сигналы, соответствующие нескольким логическим функциям от входных сигналов.  
  
Характерной особенностью узлов комбинационного типа является то, что их выходные сигналы определяются только действующими в данный момент входными сигналами (не зависят от «истории» входных сигналов).  
  
Характерной особенностью узлов накапливающие типа является то, что их выходные сигналы определяются не только действующими в данный момент входными сигналами, но и тем, какие входные сигналы поступали на узел ранее, т.е. зависят от «истории» входных сигналов. Свойство хранить историю обеспечивается у накапливающих узлов наличием память, представленной некоторой совокупностью запоминающих элементов.

*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.

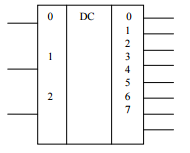


Наращивание размерности мультиплексора.



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. Структуру такой логической схемы можно определить табличным методом или с помощью диаграмм Вейча.

Дешифратором называется комбинационная схема, имеющая n входов и 2n выходов и преобразующая двоичный код на своих входах в унитарный код на выходах. Унитарным называется двоичный код, содержащий одну и только одну единицу, например 00100000. Условно-графическое обозначение дешифратора на три входа приведено на рис. 1.20



Номер разряда, в котором устанавливается "1" на выходе дешифратора, определяется кодом на его входах. Ниже приведена таблица истинности трехвходового дешифратора:

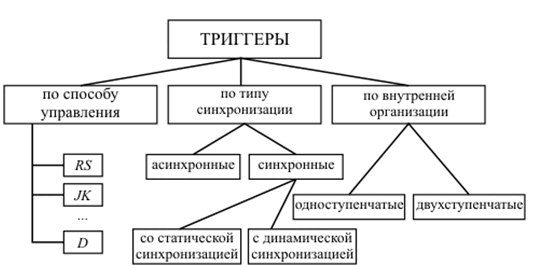


**43. Проектирование последовательностной логики. Защелки и триггеры. RS-триггер. D-защелка. D-Триггер. Регистр**

**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.

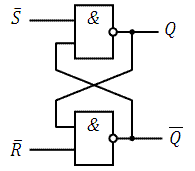
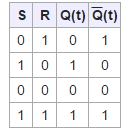
Триггеры служат основой для построения регистров, счетчиков и других элементов, обладающих функцией хранения. Главной частью любого триггера является запоминающая ячейка (ЗЯ).

Типы триггеров



**Одноступенчатый RS-триггер на элементах ИЛИ–НЕ**

#### **RS-триггер асинхронный**



.**RS-триггер**, или **SR-триггер** — триггер, который сохраняет своё предыдущее состояние при нулевых входах и меняет своё выходное состояние при подаче на один из его входов единицы.

RS-триггер используется для создания сигнала с положительным и отрицательным фронтами, отдельно управляемыми посредством стробов, разнесенных во времени. Также RS-триггеры часто используются для исключения так называемого явления дребезга контактов.

RS-триггеры иногда называют RS-фиксаторами

#### **2. RS-триггер синхронный**

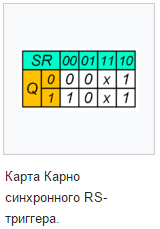
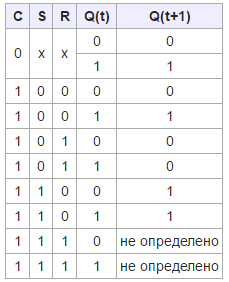
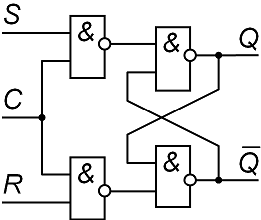


Схема синхронного RS-триггера совпадает со схемой одноступенчатого парафазного (двухфазного) D-триггера, но не наоборот, так как в парафазном (двухфазном) D-триггере не используются комбинации S=0, R=0 и S=1, R=1.

Алгоритм функционирования синхронного RS-триггера можно представить формулой

Безымянный.png где x — неопределённое состояние

D-триггер по-другому называют элементом задержки.

Триггер D может работать по уровню сигнала, он еще называется защелка. В таком устройстве нужно ограничивать длительность синхронизирующего сигнала, потому что пока синхросигнал подается - переходной процесс со входа поступает на выход.

Регистр — устройство для записи, хранения и считывания n-разрядных [двоичных](https://ru.wikipedia.org/wiki/%D0%94%D0%B2%D0%BE%D0%B8%D1%87%D0%BD%D0%B0%D1%8F_%D1%81%D0%B8%D1%81%D1%82%D0%B5%D0%BC%D0%B0_%D1%81%D1%87%D0%B8%D1%81%D0%BB%D0%B5%D0%BD%D0%B8%D1%8F) данных и выполнения других операций над ними.

Регистр представляет собой упорядоченный набор [триггеров](https://ru.wikipedia.org/wiki/%D0%A2%D1%80%D0%B8%D0%B3%D0%B3%D0%B5%D1%80), обычно [D-триггеров](https://ru.wikipedia.org/wiki/D-%D1%82%D1%80%D0%B8%D0%B3%D0%B3%D0%B5%D1%80), число которых соответствует числу разрядов в [слове](https://ru.wikipedia.org/wiki/%D0%9C%D0%B0%D1%88%D0%B8%D0%BD%D0%BD%D0%BE%D0%B5_%D1%81%D0%BB%D0%BE%D0%B2%D0%BE). С регистром может быть связано [комбинационное цифровое устройство](https://ru.wikipedia.org/wiki/%D0%9A%D0%BE%D0%BC%D0%B1%D0%B8%D0%BD%D0%B0%D1%86%D0%B8%D0%BE%D0%BD%D0%BD%D0%B0%D1%8F_%D0%BB%D0%BE%D0%B3%D0%B8%D0%BA%D0%B0), с помощью которого обеспечивается выполнение некоторых операций над словами.

**44. Триггер с функцией разрешения. Триггер с функцией сброса. Проектирование синхронных логических схем. Синхронные последовательностные схемы. Синхронные и асинхронные схемы.**

**45. Конечные автоматы. Пример проектирования конечного автомата**

Одним из критериев сложности конечного автомата является число его состояний. Чем меньше это число, тем проще дискретное устройство, реализующее данный автомат. Поэтому одной из важных задач теории конечных автоматов является построение автомата с наименьшим числом состояний.

Поскольку в современных компьютерах любая информация представ­ляется в виде двоичных кодов, то для построения автомата можно исполь­зовать элементы, имеющие лишь два различных устойчивых состояния, одно из которых соответствует цифре 0,а другое цифре 1.

Приведём несколько примеров конечных автоматов.

1)Элемент задержки (элемент памяти)

2)Двоичный сумматор последовательного действия

3) Схема сравнения на равенство

Элементы задержки представляют собой устройство, имеющее один вход и один выход. Причем значение выходного сигнала в момент времени ***t***совпадает со значением сигнала в предыдущий момент.

**46. Конечные автоматы. Кодирование состояний. Автоматы Мура и Мили. Декомпозиция конечных автоматов. Восстановление конечных автоматов по электрической схеме.**

Кодирование внутренних состояний автомата заключается в сопоставлении каждому состоянию автомата набора значений соответствующих состояний элементарных автоматов памяти. При функционировании автомата могут появиться так называемые состязания, которые возникают вследствие того, что

1. Элементарные автоматы памяти имеют различные, хотя достаточно близкие, времена срабатывания.

2. Различные времена задержки сигналов (функции возбуждения), поступающих на входы элементов памяти.

Если при переходе автомата из одного состояния в другое должны изменить свои состояния сразу несколько элементов памяти, то между ними начинаются состязания. Элемент памяти, который изменил своё состояние раньше других, может через цепь обратной связи (КС1) изменять сигналы на входах других запоминающих элементов. Это может привести к переходу автомата в состояние непредусмотренное графом его функционирования.

**Автомат Мили** описывается следующими формулами:

1. внутреннее состояние автомата в следующий момент времени зависит от внутреннего состояния автомата в настоящий момент времени и входного сигнала в настоящий момент времени.

https://studfile.net/html/2706/58/html_3hBcbthBt4.FgrG/img-lth4Qs.png

2. выходной сигнал автомата в настоящий момент времени зависит от входного сигнала в настоящий момент времени и внутреннего состояния автомата в настоящий момент времени.

https://studfile.net/html/2706/58/html_3hBcbthBt4.FgrG/img-sD6T0O.png

Понятие состояния автомата в момент времени t определяется внутренним состоянием автомата и состоянием входа автомата в тот же момент времени.

https://studfile.net/html/2706/58/html_3hBcbthBt4.FgrG/img-Psp_vu.png

**Автомат Мура**

Для автомата Мура функции переходов и выходов выглядят следующим образом:

https://studfile.net/html/2706/58/html_3hBcbthBt4.FgrG/img-1URuGQ.png

https://studfile.net/html/2706/58/html_3hBcbthBt4.FgrG/img-m9IM6h.png

Функция выходов для автомата Мура определяется внутренним состоянием автомата.

Задача декомпозиции – задача построения сети автоматов *N*, реализующей заданный автомат. Таким образом, задача декомпозиции состоит в том, чтобы для заданного автомата построить сеть из более простых автоматов, которая бы реализовала заданный автомат. При декомпозиции используется аппарат разбиения множества состояний

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

Последовательностные схемы или цифровые автоматы (ЦА) с памятью составляют другой, более сложный класс преобразователей дискретной информации. В отличие от КС они имеют некоторое конечное число различных внутренних состояний. Выходные сигналы ЦА в данном такте определяются в общем случае входными сигналами, поступившими на вход ЦА в этом такте, и внутренним состоянием автомата, которое явилось результатом воздействия на автомат входных сигналов в предыдущие такты.

Эффект **метастабильности** может возникнуть в нескольких классических ситуациях:

1. Явное нарушение параметров Ts и Th триггера. Обычно возникает, когда устройство пытаются заставить работать на частоте, на которой оно физически работать не может из-за слишком длинных путей распространения сигналов. Данная ситуация контролируется временным анализатором и при нормальной работе (при выполнении временных требований) встречаться не должна.
2. Нарушение временных параметров по входам асинхронного сброса. Удивительно, но несмотря на то, что сброс *асинхронный*, он должен быть *синхронен* тактовому сигналу того триггера, который сбрасывает, т.е. его временные параметры декларируются относительно фронта тактового импульса. Об этом часто забывают, подавая сброс откуда ни попадя. Опять-таки, если сигнал сброса порожден с помощью того же тактового сигнала, как и тактовый сигнал сбрасываемого триггера, то временной анализатор разберется самостоятельно.
3. Сигналы, подаваемые синхронно (на той же частоте с известной задержкой) из других устройств — здесь простейший временной анализ (т.е. декларирование только тактовой частоты) не работает и необходимо явно задавать временные характеристики ваших входных/выходных сигналов. Это является отдельной непростой (но вполне разрешимой) задачей временного анализа.
4. И, наконец, самый тяжелый случай — сигналы передаются асинхронно, может быть совсем на другой тактовой частоте (и с неизвестными сдвигами фазы) относительно приемных триггеров. Их источником может быть как внешнее оборудование (хоть кнопка на плате или последовательный порт) так и блоки той же микросхемы, работающие на другой тактовой частоте. Именно об этой ситуации и пойдет дальнейший разговор.

**48. Параллелизм**

Параллелизм - это свойство систем, при котором несколько вычислений выполняются одновременно, и при этом, возможно, взаимодействуют друг с другом. Вычисления могут выполняться на нескольких [ядрах](https://ru.wikipedia.org/wiki/%D0%9C%D0%BD%D0%BE%D0%B3%D0%BE%D1%8F%D0%B4%D0%B5%D1%80%D0%BD%D1%8B%D0%B9_%D0%BF%D1%80%D0%BE%D1%86%D0%B5%D1%81%D1%81%D0%BE%D1%80) одного [чипа](https://ru.wikipedia.org/wiki/%D0%98%D0%BD%D1%82%D0%B5%D0%B3%D1%80%D0%B0%D0%BB%D1%8C%D0%BD%D0%B0%D1%8F_%D1%81%D1%85%D0%B5%D0%BC%D0%B0) с [вытесняющим разделением времени](https://ru.wikipedia.org/wiki/%D0%92%D1%8B%D1%82%D0%B5%D1%81%D0%BD%D1%8F%D1%8E%D1%89%D0%B0%D1%8F_%D0%BC%D0%BD%D0%BE%D0%B3%D0%BE%D0%B7%D0%B0%D0%B4%D0%B0%D1%87%D0%BD%D0%BE%D1%81%D1%82%D1%8C) [потоков](https://ru.wikipedia.org/wiki/%D0%9F%D0%BE%D1%82%D0%BE%D0%BA_%D0%B2%D1%8B%D0%BF%D0%BE%D0%BB%D0%BD%D0%B5%D0%BD%D0%B8%D1%8F) на одном процессоре, либо выполняться на физически отдельных [процессорах](https://ru.wikipedia.org/wiki/%D0%9F%D1%80%D0%BE%D1%86%D0%B5%D1%81%D1%81%D0%BE%D1%80). Для выполнения параллельных вычислений разработаны ряд математических моделей, в том числе [сети Петри](https://ru.wikipedia.org/wiki/%D0%A1%D0%B5%D1%82%D0%B8_%D0%9F%D0%B5%D1%82%D1%80%D0%B8), [исчисление процессов](https://ru.wikipedia.org/wiki/%D0%98%D1%81%D1%87%D0%B8%D1%81%D0%BB%D0%B5%D0%BD%D0%B8%D0%B5_%D0%BF%D1%80%D0%BE%D1%86%D0%B5%D1%81%D1%81%D0%BE%D0%B2), модели параллельных случайных доступов к вычислениям.

**49. Корректирующие коды. Код Хэ́мминга. Область применения.**

Корректирующими называются коды, которые позволяют обнаруживать ошибки и исправлять их на приемной стороне, не прибегая к повторной передаче ошибочных кодовых комбинаций.

Исходя из основных параметров и способов кодирования и декодирования, корректирующие коды в первую очередь можно разделить на блочные и непрерывные.

Блочные коды характеризуются тем, что каждая кодовая комбинация состоит из двух частей (блоков), первая состоит из информационных символов, вторая – из контрольных. Особенностью непрерывных кодов является то, кодовая комбинация не разделяется на блоки, а контрольные символы размещаются по определенному правилу между информационными.

Для корректирующих кодов справедливо неравенство

*N = Km https://studfile.net/html/2706/241/html_b9dRJ4ulkd.Fhm5/img-SmLMEE.pngM*,

где *М*– количество сообщений;*N*– количество кодовых комбинаций;*К*– основание кода;

*m*– длина кодовой комбинации:*m = n + k*;*n*– число информационных разрядов,*k*– число контрольных разрядов, обеспечивающих локализацию и исправление искаженных элементов кодовой комбинации.

**Код Хэ́мминга** — вероятно, наиболее известный из первых самоконтролирующихся и самокорректирующихся [кодов](https://ru.wikipedia.org/wiki/%D0%9A%D0%BE%D0%B4). Построен применительно к [двоичной системе счисления](https://ru.wikipedia.org/wiki/%D0%94%D0%B2%D0%BE%D0%B8%D1%87%D0%BD%D0%B0%D1%8F_%D1%81%D0%B8%D1%81%D1%82%D0%B5%D0%BC%D0%B0_%D1%81%D1%87%D0%B8%D1%81%D0%BB%D0%B5%D0%BD%D0%B8%D1%8F).

Позволяет исправлять одиночную ошибку (ошибка в одном бите) и находить двойную Коды Хэмминга являются самоконтролирующимися кодами, то есть кодами, позволяющими автоматически обнаруживать ошибки при передаче данных. Для их построения достаточно приписать к каждому слову один добавочный (контрольный) двоичный разряд и выбрать цифру этого разряда так, чтобы общее количество единиц в изображении любого числа было, например, нечетным. Одиночная ошибка в каком-либо разряде передаваемого слова (в том числе, может быть, и в контрольном разряде) изменит четность общего количества единиц. Счетчики по модулю 2, подсчитывающие количество единиц, которые содержатся среди двоичных цифр числа, могут давать сигнал о наличии ошибок.

При этом невозможно узнать, в каком именно разряде произошла ошибка, и, следовательно, нет возможности исправить её. Остаются незамеченными также ошибки, возникающие одновременно в двух, четырёх, и т. д. — в четном количестве разрядов. Впрочем, двойные, а тем более четырёхкратные ошибки полагаются маловероятными.

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.

Код Хэмминга используется в некоторых прикладных программах в области хранения данных, особенно в [RAID 2](https://ru.wikipedia.org/wiki/RAID); кроме того, метод Хэмминга давно применяется в памяти типа [ECC](https://ru.wikipedia.org/wiki/ECC) и позволяет «на лету» исправлять однократные и обнаруживать двукратные ошибки.

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

Язык описания аппаратуры (HDL от [англ.](https://ru.wikipedia.org/wiki/%D0%90%D0%BD%D0%B3%D0%BB%D0%B8%D0%B9%D1%81%D0%BA%D0%B8%D0%B9_%D1%8F%D0%B7%D1%8B%D0%BA) hardware description language) — специализированный [компьютерный язык](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)), написанные на них программы также состоят из выражений, операторов, управляющих структур. Важнейшим отличием между обычными языками программирования и языками HDL является явное включение концепции времени в языки описания аппаратуры.

Языки описания аппаратуры являются неотъемлемой частью [САПР](https://ru.wikipedia.org/wiki/%D0%90%D0%B2%D1%82%D0%BE%D0%BC%D0%B0%D1%82%D0%B8%D0%B7%D0%B0%D1%86%D0%B8%D1%8F_%D0%BF%D1%80%D0%BE%D0%B5%D0%BA%D1%82%D0%B8%D1%80%D0%BE%D0%B2%D0%B0%D0%BD%D0%B8%D1%8F_%D1%8D%D0%BB%D0%B5%D0%BA%D1%82%D1%80%D0%BE%D0%BD%D0%B8%D0%BA%D0%B8), особенно для таких сложных схем, как [специализированные интегральные схемы](https://ru.wikipedia.org/wiki/ASIC), [микропроцессоры](https://ru.wikipedia.org/wiki/%D0%9C%D0%B8%D0%BA%D1%80%D0%BE%D0%BF%D1%80%D0%BE%D1%86%D0%B5%D1%81%D1%81%D0%BE%D1%80) и [программируемые логические устройства](https://ru.wikipedia.org/wiki/%D0%9F%D0%9B%D0%98%D0%A1).

Основные практически используемые языки описания аппаратуры — [Verilog](https://ru.wikipedia.org/wiki/Verilog) и [VHDL](https://ru.wikipedia.org/wiki/VHDL); также существует несколько десятков альтернативных языков.

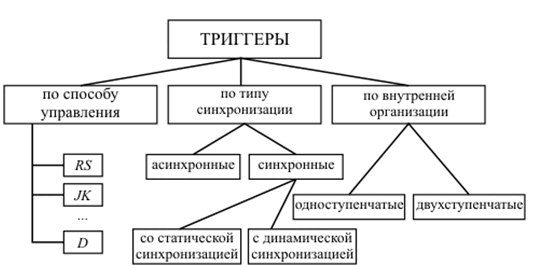
FPGA  (Field-Programmable Gate Array) — программируемая логическая интегральная схема (ПЛИС), конфигурация которой может быть загружена после включения питания. Большинство FPGA не имеют встроенной энергонезависимой памяти, поэтому повторное включение устройства требует повторной загрузки конфигурации FPGA (информация объёмом от сотен Кбайт и более). Наличие в изделии FPGA означает феноменальную гибкость логики этого изделия (в большинстве случаев это заводская возможность коррекции логики работы устройства без проведения каких-либо электромонтажных операций, позволяющая учесть потребности потребителей). Ресурсы современных FPGA (даже младших в выбранном семействе) уже позволяют реализовывать сложнейшие алгоритмы

**51. Типы триггеров. Классификация триггеров.**

**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.

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

Типы триггеров



**Одноступенчатый RS-триггер на элементах ИЛИ–НЕ**

Основным триггером, на котором базируются все остальные триггеры является RS-триггер.

RS-триггер имеет два логических входа:

R - установка 0 (от слова reset);

S - установка 1 (от слова set).

RS-триггер имеет два выхода:

Q - прямой;

· Q(со штрихом) - обратный (инверсный).

Состояние триггера определяется состоянием прямого выхода. Простейший RS-триггер состоит из двух логических элементов, охваченных перекрёстной положительной обратной связью

Для описания работы триггера используют таблицу состояний (переходов).

Обозначим:

Q(t) - состояние триггера до поступления управляющих сигналов (изменения на входах R и S);

·Q(t+1) - состояние триггера после изменения на входах R и S.

Таблица 2.1 - Таблица переходов RS триггера в базисе ИЛИ-НЕ

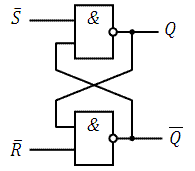
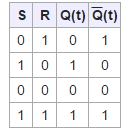
|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| R | S | Q(t) | Q(t+1) | Пояснения |
| 0 | 0 | 0 | 0 | Режим хранения информации R=S=0 |
| 0 | 0 | 1 | 1 |
| 0 | 1 | 0 | 1 | Режим установки единицы S=1 |
| 0 | 1 | 1 | 1 |
| 1 | 0 | 0 | 0 | Режим установки нуля R=1 |
| 1 | 0 | 1 | 0 |
| 1 | 1 | 0 | \* | R=S=1 запрещённая комбинация |
| 1 | 1 | 1 | \* |

**Одноступенчатый RS-триггер на элементах И–НЕ.**

**Триггер** (триггерная система) — класс электронных устройств, обладающих способностью длительно находиться в одном из двух устойчивых состояний и чередовать их под воздействием внешних сигналов. Каждое состояние триггера легко распознаётся по значению выходного напряжения. По характеру действия триггеры относятся к импульсным устройствам — их активные элементы (транзисторы, лампы) работают в ключевом режиме, а смена состояний длится очень короткое время.

ВИДЫ ОДНОСТУПЕНЧАТЫХ ТРИГГЕРОВ:

#### **RS-триггер асинхронный**



.**RS-триггер**, или **SR-триггер** — триггер, который сохраняет своё предыдущее состояние при нулевых входах и меняет своё выходное состояние при подаче на один из его входов единицы.

RS-триггер используется для создания сигнала с положительным и отрицательным фронтами, отдельно управляемыми посредством стробов, разнесенных во времени. Также RS-триггеры часто используются для исключения так называемого явления дребезга контактов.

RS-триггеры иногда называют RS-фиксаторами

#### **2. RS-триггер синхронный**

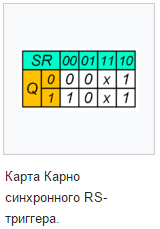
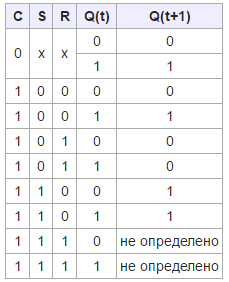
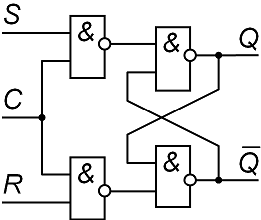


Схема синхронного RS-триггера совпадает со схемой одноступенчатого парафазного (двухфазного) D-триггера, но не наоборот, так как в парафазном (двухфазном) D-триггере не используются комбинации S=0, R=0 и S=1, R=1.

Алгоритм функционирования синхронного RS-триггера можно представить формулой

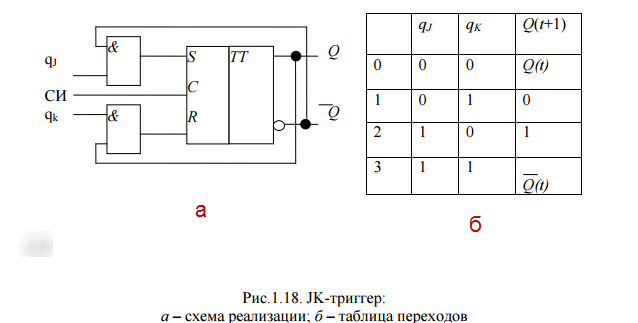
Безымянный.png где x — неопределённое состояние.

***T- JK-, D-триггеры.***

Т-триггер представляет собой триггер, имеющий один вход «Т», поступление единичного сигнала на который переводит Т- триггер в состояние, противоположное его исходному состоянию (фигурально говоря, по каждому входному сигналу триггер «кувыркается», меняя свое состояние на противоположное).

Т-триггер можно рассматривать как счетчик, считающий по модулю два количество импульсов, поступающих на его вход. Действительно, если в исходном состоянии триггер находится в «0», то при поступлении на его вход нечетного количества импульсов триггер будет находиться в «1», а при четном в– «0», что соответствует суммированию по модулю «2» количества поступающих импульсов.

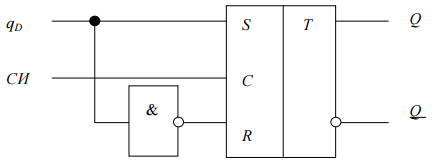
JK-триггер. Реализация JK-триггера и соответствующая таблица истинности приведена на рис. 1.18. Вход «J» – это вход установки «1», вход «K» – вход установки «0».



Таким образом, JK-триггер представляет собой универсальный триггер, объединяющий в себе свойства и RS-триггера и Т-триггера.

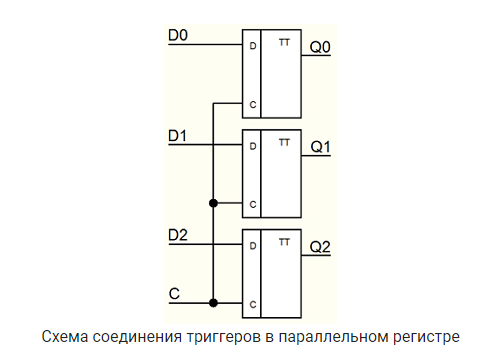
D-триггер по-другому называют элементом задержки.

Использование подачи сигнала установки «1» через логику НЕ на вход установки «0» приводит к тому, что на входы R и S базового RS-триггера подаются сигналы, имеющие противоположные значения.



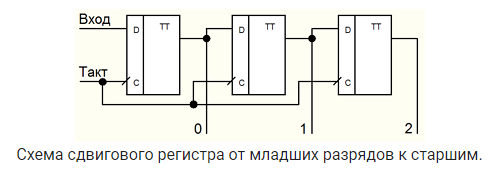
**52. Параллельные и последовательные регистры. Мультиплексоры и демультиплексоры. Отличия в обозначения цифровых элементов в разных стандартах.**

Параллельный регистр



В данном типе регистров триггеры соединены параллельно, то есть каждый внутренний триггер имеет свой вход D и свой выход Q, которые не зависят от других триггеров, а также вход С, который называется **тактовым входом** и для всех входящих в регистр триггеров он является общим. Параллельные регистры бывают двух типов:

Последовательный регистр



Регистр сдвига или сдвиговый 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.

**Мультиплексор** - это устройство, которое осуществляет выборку одного из нескольких входов и подключает его к своему единственному выходу, в зависимости от состояния двоичного кода. Другими словами, мультиплексор - переключатель сигналов, управляемый двоичным кодом и имеющий несколько входов и один выход. К выходу подключается тот вход, чей номер соответствует управляющему двоичному коду.

**мультиплексор** - это устройство, преобразующее параллельный код в последовательный

**Демультиплексор** - устройство, обратное мультиплексору. Т. е., у демультиплексора один вход и много выходов. Двоичный код определяет, какой выход будет подключен ко входу.

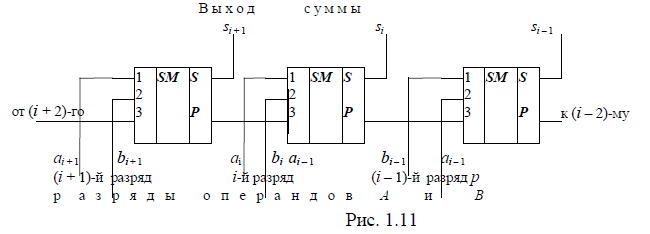
Другими словами, **демультиплексор** - это устройство, которое осуществляет выборку одного из нескольких своих выходов и подключает его к своему входу или, ещё, это переключатель сигналов, управляемый двоичным кодом и имеющий один вход и несколько выходов.

Ко входу подключается тот выход, чей номер соответствует состоянию двоичного кода. И частное определение: **демультиплексор** - это устройство, которое преобразует последовательный код в параллельный.

Обычно в качестве демультиплексора используют **дешифраторы**двоичного кода в позиционный, в которых вводят дополнительный вход стробирования.

**53. Сумматор. Многоразрядный сумматор.**

**Многоразрядный двоичный сумматор** строится на основе одноразрядных сумматоров с введением соответствующих связей между разрядами. На рис. 1.11 приведена простейшая схема такого сумматора. На схеме показана только часть сумматора, относящаяся к некоторому i-му разряду и его соседям: (i + 1)-й соседний младший разряд и (i - 1)-й соседний старший разряд.



Приведенная схема многоразрядного сумматора называется **схемой сумматора с последовательным переносом**. Схема очень простая. **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.

**54. Устройства ЭВМ. Состав АЛУ. Типы памяти**

**Устройства ЭВМ**

Классическая ЭВМ состоит из трех основных устройств: арифметико-логического устройства, устройства управления и запоминающего устройства. Рассмотрим особенности организации этих устройств. Прежде всего, рассмотрим структуру арифметико-логического устройства:

**Состав АЛУ**

АЛУ состоит из следующих типовых узлов:

– Регистры (R), служащие для хранения операндов и результатов;

– Сумматор (SM), служащий для выполнения операции суммирования многоразрядных кодов;

– Операционные узлы (ОУ), служащие для выполнения логических операций;

– Мультиплексор (MS);

– Счетчик (Сч), обеспечивающий подсчет тактов длинных операций;

– Регистр флажков (RF), служащий для фиксации особой информации, характеризующей полученный результат.

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

Управляющий блок осуществляет выработку множества управляющих сигналов Y, обеспечивающих выполнение элементарных операций (микроопераций) типовыми узлами операционного блока.

При работе управляющая часть АЛУ использует код заданной операции (например сложение, умножение, вычитание и т. п.), а также информацию о состоянии операционного блока, представленную в виде множества Х признаков, формируемых типовыми узлами. К признакам, вырабатываемым регистром и посылаемым в управляющую часть, относятся:

– «ноль регистра» (R{0...n} = 0) характеризует состояние, при котором во всех разрядах регистра имеет место нулевое значение;

– «ноль знака» (R{зн} = 0) – в знаковом разряде регистра находится значение 0;

– «единица старшего разряда» (R{1} = 1) – в старшем разряде регистра находится значение единица;

– «единица младшего разряда» (R{n} = 1) – в младшем разряде регистра находится значение единица.

К микрооперациям, которые может выполнять регистр при поступлении соответствующего управляющего сигнала уi , относятся:

– прием кода;

– выдача прямого кода;

– выдача инверсного кода;

– установка единицы в некотором разряде регистра;

– обнуление знакового разряда;

– сдвиг кода влево;

– сдвиг кода вправо;

– обнуление регистра (во все разряды регистра устанавливается нулевое значение).

Счетчик может выполнять следующие операции, инициируемые по управляющим сигналам, поступающим из управляющего блока:

– Установка нуля в счетчике;

– Установка в счетчике некоторого начального значения;

– Установка режима счета (обратный или прямой счет);

– Изменение находящегося в счетчике текущего значения на единицу

К признакам, вырабатываемым счетчиком и посылаемым в управляющую часть, относятся:

– «ноль счетчика» («0» Сч) – характеризует состояние, при котором во всех разрядах регистра имеет место нулевое значение;

– «переполнение счетчика» – при поступлении очередного счетного сигнала счетчик переходит от максимального значения к значению «0».

Счетчик может выполнять следующие операции, инициируемые по управляющим сигналам, поступающим из управляющего блока:

– установка нуля в счетчике;

– установка в счетчике некоторого начального значения;

– установка режима счета (обратный или прямой счет);

– изменение находящегося в счетчике текущего значения на единицу.

К признакам, вырабатываемым сумматором и посылаемым в управляющую часть, относятся:

– признак нулевого результата;

– единичных значений во всех разрядах результата;

– единицы в первом знаковом разряде результата;

– единицы во втором знаковом разряде результата;

– переноса из старшего разряда сумматора;

– наличия в тетраде значения, большего 9;

– межтетрадного переноса.

Каждому из перечисленных состояний может соответствовать отдельный разряд (флажок) в регистре флажков.

Сумматор может выполнять следующие микрооперации, инициируемые по управляющим сигналам, поступающим из управляющего блока:

– прием кода двух операндов на свои входы;

– формирование поразрядной суммы операндов, поступающих на его входы;

– генерирование поразрядного переноса;

– распространение переносов через разряды поразрядной суммы, пропускающие перенос;

– прибавление единицы в младший разряд;

– прибавление корректирующих кодов в тетрады при сложении двоично-десятичных кодов.

Выполнение любой арифметической операции в АЛУ реализуется за счет выполнения определенной последовательности микроопераций в узлах операционной части АЛУ.

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.

Запоминающие устройства играют важную роль в общей структуре ЭВМ. По некоторым оценкам производительность компьютера на разных классах задач на 40– 50% определяется характеристиками ЗУ различных типов, входящих в его состав.

К основным параметрам, характеризующим запоминающие устройства, относятся емкость и быстродействие.

**Емкость памяти** – это максимальное количество данных, которое в ней может храниться.

Емкость запоминающего устройства измеряется количеством адресуемых элементов (ячеек) ЗУ и длиной ячейки в битах.

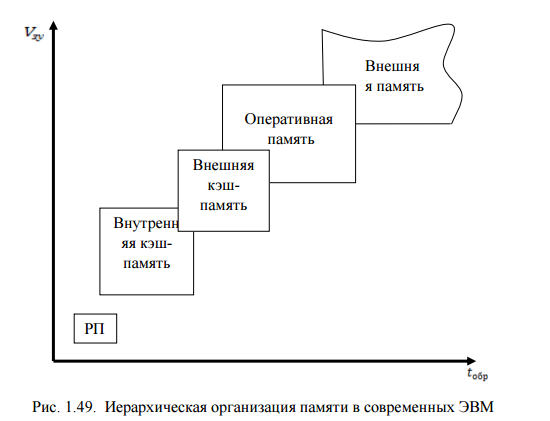
Быстродействие памяти определяется продолжительностью операции обращения, то есть временем, затрачиваемым на поиск нужной информации в памяти и на ее считывание, или временем на поиск места в памяти, предназначенного для хранения данной информации, и на ее запись:



где быстродействие ЗУ при считывании информации;

 быстродействие ЗУ при записи

**Оперативная память** – устройство, которое служит для хранения информации (программ, исходных данных, промежуточных и конечных результатов обработки), непосредственно используемой в ходе выполнения программы в процессоре. В настоящее время объем ОП персональных компьютеров составляет несколько гигабайт. Оперативная память работает на частоте системной шины и требует 6– 8 циклов синхронизации шины для обращения к ней. Так, при частоте работы системной шины 100 МГц (при этом период равен 10 нс) время обращения к оперативной памяти составит несколько десятков наносекунд.



**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.