[1.Системы счисления.](#g4gi4vfh7wx0)

[2.Перевод чисел из одной системы счисления в другую. Метод преобразования с использованием весов разрядов.](#fptvsvebdw7g)

[3.Перевод чисел из одной системы счисления в другую. Метод деления (умножения) на новое основание.](#u9jc85wgpwbe)

[4.Перевод чисел из одной системы счисления в другую. Метод с использованием особого соотношения оснований исходной и искомой систем счисления.](#1v39aefsjjl0)

[5.Арифметические операции над двоичными числами.Операция сложения в 2-й СС](#kix.ii1p31neonqr)

[6.Операция вычитания в 2-й СС](#rx98uczid9s)

[7.Операция умножения в 2-й СС начиная со старшего разряда множителя со сдвигом множимого.](#kix.o2t9t257jg9f)

[8.Операция умножения в 2-й СС начиная с младшего разряда множителя со сдвигом множимого.](#stxvo1j5s6l4)

[9.Операция умножения в 2-й СС начиная со старшего разряда множителя со сдвигом промежуточных результатов](#m3a62pn98jqs)

[10.Операция умножения в 2-й СС начиная с младшего разряда множителя со сдвигом частичных произведений.](#axck4sdhu2wn)

[11.Деление двоичных чисел.](#guccx61v3995)

[12.Двоично-десятичная арифметика. Сложение двоично-десятичных чисел.](#z5x38igmjm9h)

[13.Двоично-десятичная арифметика. Вычитание двоично-десятичных чисел.](#p2hy2bwh03q5)

[14.Кодирование алгебраических чисел.](#pd3r77es5uxw)

[15.Дополнительный и обратный коды двоичных чисел](#l2f522v0yhw)

[16.Операции с двоичными числами в дополнительном коде](#911lw4xtoyrk)

[17.Операции с двоичными числами в обратном коде](#jswbz2v0o6s0)

[18.Модифицированные коды](#kix.fuf0zycwtsi0)

[19.Логические операции с двоичными кодами: логическое суммирование, логическое умножение, логическое отрицание, суммирование по модулю два, логические сдвиги.](#9o8xuce65wrq)

[20.Арифметические сдвиги положительных двоичных чисел, представленных в прямом коде](#kix.rmzspp1lmyjr).

[21.Арифметические сдвиги отрицательных двоичных чисел, представленных в прямом коде.](#9sfwgvy4o7nd)

[22.Арифметические сдвиги двоичных чисел, представленных в обратном коде.](#dhyttkv1t34m)

[23.Арифметические сдвиги двоичных чисел, представленных в дополнительном коде. Сдвиг отрицательных чисел с переполнением.](#l99im61vilh8)

[24.Представление чисел с фиксированной точкой. Арифметические операции над числами, представленными с фиксированной точкой.](#kix.1gn6r9mzr8jn)

[25.Представление чисел с плавающей точкой.](#kix.8hozlhjmwu3h)

26.Сложение чисел, представленных в формате с плавающей точкой

[27.Умножение чисел, представленных в формате с плавающей точкой.](#kix.xhh2ma1c681t)

[28.Основные понятия алгебры логики. Способы задания логической функции.](#26iotxuwymux)

[29.Законы и правила алгебры Буля](#qkp9xfrb3qmy)

[30.Канонические формы представления логических функций.Преобразование форм представления логических функций.](#tnc8y7ruqv5p)р

[31.Синтез логических схем по логическим выражениям в булевом базисе.](#3mh3z1ua1san)

[32.Минимизация логических выражений методом Квайна.](#no04uxmct1lu)

[33.Минимизация логических выражений с диаграммами Вейча.](#u1e6v38barpc)

[34.Логический базис И-НЕ.Синтез логических схем по логическому выражению в базисе И-НЕ.](#pnxjmv438ul8)

[35.Логический базис ИЛИ-НЕ.Синтез логических схем по логическому выражению в базисе ИЛИ-НЕ.](#wwek4ye8847y)

[36.Логические элементы. Реализация простейших логических элементов с помощью резистивно-диодной логики.](#jvvox3rsoha1)

[37.Реализация простейших логических элементов на полупроводниковых триодах (транзисторах).](#h6aswb2zke20)

[38.Запоминающие элементы. Типы триггеров. Классификация триггеров](#kix.oqe56gfhhpnz).

[39.Одноступенчатый RS-триггер на элементах ИЛИ–НЕ.](#k27fflx8uht5)

[40.Одноступенчатый RS-триггер на элементах И–НЕ.](#kix.5jxpr9gaxme2)

[41.Синхронный одноступенчатый RS-триггер. Условно-графическое обозначение RS-триггера с различной синхронизацией. Временная диаграмма работы RS-триггера с различными типами синхронизации.](#mma9oj7j0pel)

[42.Двухступенчатый RS-триггер.](#in9d62618tvh)

[43.T-, JK-, D-триггеры.](#t6thzzb0sioz)

[44.Комбинационные узлы. Дешифратор. Шифратор.](#kix.m2qca8efn38t)

[45.Комбинационные узлы. Сумматор по модулю 2](#kix.ly8o3af20sge)

[46.Комбинационные узлы. Мультиплексор. Демультиплексор.](#qz98cou09qam)

[47.Счетчики. Схема, временная диаграмма работы, оценка быстродействия асинхронного счетчика.](#kix.6ka73z2onn0x)

[48.Сумматор. Синтез одноразрядного двоичного сумматора. Многоразрядный сумматор.](#oguvlr31uk6d)

[49.Программируемые логические матрицы.](#lak8rpx8xc1d)

[50.Накапливающие узлы. Регистр хранения.](#wb41vqrlqj7t)

[51.Накапливающие узлы. Регистр сдвига.](#kix.rsew1jjuqer8)

[52.Устройства ЭВМ. Состав АЛУ. Регистр флажков.](#xucxu2jdglu3)

[53.Синтез АЛУ.](#lnwdci9pcuzg)

[54.Устройство управления схемного типа.](#haw2pd3erjjy)

[55.Микропрограммное устройство управления](#qm03azlnv8hp).

[56.Запоминающие устройства. Характеристики, классификация ЗУ.](#5wz30jdgix2g)

[57.Иерархия памяти в современных ЭВМ.](#trbym3j400rt)

[58.Построение ЗУ с заданной организацией. ЗУ 8К×8 на БИС 1К×8.](#u633t29bj1t0)

[59.Построение ЗУ с заданной организацией. ЗУ 8К×8 на БИС 1К×1.](#ajlgs7m22q1b)

**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.

*Пример.*

Значение числа МММСМLIХ определяется как сумма: 1000 + 1000 + 1000 + (1000 *–* 100) + 50 + (10 *–* 1),что соответствует десятичному эквиваленту 3959.

**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.ьтло

***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 ⋅*q*0(расширенная запись), где *Аi* – цифра записи числа, удовлетворяющая условию 0 ≤ *Аi* ≤ (*q*–1), где *q* – основание СС.

При *q* =10 *А* изменяется в диапазоне от 0 до 9.

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

− двоичная, восьмеричная, шестнадцатеричная и т. д.

В различных СС в качестве цифр используются обозначения соответствующих цифр десятичной СС – 0, 1, 2, 3, 4, 5, 6, 7, 8, 9, а в случае, когда десятичных цифр «не хватает» (для систем счисления с основанием *q*, большим чем 10), для цифр, превышающих 9, вводятся дополнительные обозначения, например, для *q* = 16 это будут обозначения А, В, C, D, E, F, которые соответствуют шестнадцатеричным цифрам (десятичные эквиваленты их равны соответственно 10, 11, 12, 13, 14, 15).

В связи с тем, что в дальнейшем изложении будут использоваться различные СС, примем такое обозначение: *Nq* – число *N*, представленное в СС с основанием *q*.

*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;

На основании вышеизложенного можно заключить, что запись одного и того же числа в различных СС будет тем длиннее, чем меньше основание СС. Например, *N* = 206310 = 1000000011112 = 40178 = 80F16.

Человек в своей практической деятельности наиболее часто использует *десятичную* СС. *Двоичная СС* является удобной для обработки информации в ЭВМ. Промежуточное место между ними занимает *двоично*-*десятичная СС*, которая является десятичной, но отдельные десятичные цифры в ней записываются в виде набора двоичных разрядов.

***2.Перевод чисел из одной системы счисления в другую.***

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

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

- преобразования с использованием весов разрядов в исходной и в искомой записи числа;

- деления (умножения) на новое основание;

- с использованием особого соотношения заданной и искомой систем счисления.

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

Метод преобразования с использованием весов разрядов записи числа в исходной и в искомой системах предполагает применение расширенной записи числа (1.2) 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.

*Пример*

Найти двоичный эквивалент десятичного числа:

43610 = \_\_\_\_?2.

*Решение*

Первый (старший) разряд, имеющий значение 1 в искомой двоичной записи числа, будет разряд весом 28 = 256. С помощью остальных (младших) разрядов искомой записи числа необходимо представить значение 180 (180 – остаток, полученный как 436 *–* 256).

Второй разряд с весом 27 = 128 будет иметь в искомой двоичной записи числа значение 1. С помощью остальных (более младших) разрядов искомой записи числа необходимо представить значение 52 (остаток, полученный как 180 *–* 128).

Третий разряд с весом 26 = 64 будет иметь в искомой двоичной записи числа значение 0.

Четвертый разряд с весом 25 = 32 будет иметь в искомой двоичной записи числа значение 1, а остаток *–* 20.

Пятый разряд с весом 24 = 16 будет иметь в искомой двоичной записи числа значение 1, а остаток *–* 4.

Шестой разряд с весом 23 = 8 будет иметь в искомой двоичной записи числа значение 0.

Седьмой разряд с весом 22 = 4 будет иметь в искомой двоичной записи числа значение 1, а остаток *–* 0.

Восьмой разряд с весом 21 = 2 будет иметь в искомой двоичной записи числа значение 0.

Девятый разряд с весом 20 = 1 будет иметь в искомой двоичной записи числа значение 0.

Таким образом

43610 = 1101101002.

*Пример*

Найти двоичный эквивалент числа 0,710 = ?2.

*Решение*

Предварительный результат ищется с точностью до пяти двоичных разрядов, причем пятый разряд используется только для округления при переходе к четырехразрядному окончательному результату.

Первый (старший) разрядом весом 2*–*1 = 0,5 искомой двоичной записи числа будет иметь значение 1. С помощью остальных (младших) разрядов искомой записи числа необходимо представить значение 0,2 (0,2 – остаток, полученный как 0,7 *–* 0,5 = 0,2).

Второй (старший) разряд с весом 2*–*2 = 0,25 в искомой двоичной записи числа будет иметь значение 0.

Третий разрядом с весом 2*–*3 = 0,13 в искомой двоичной записи числа будет иметь значение 1. С помощью остальных (более младших) разрядов искомой записи числа необходимо представить значение 0,07 (0,07 – остаток, полученный как 0,2 *-* 0,13).

Четвертый разрядом с весом 2*–*4 = 0,06 в искомой двоичной записи числа будет иметь значение 1, а остаток 0,01.

Пятый разряд с весом 2*–*5 = 0,03 искомой двоичной записи числа будет иметь значение 0.

Таким образом, десятичное число 0,710 = 0,101102.

После округления имеет место 0,710 = 0,10112.

***3. Перевод чисел из одной системы счисления в другую. Метод деления (умножения) на новое основание.***

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

- преобразования с использованием весов разрядов в исходной и в искомой записи числа;

- деления (умножения) на новое основание;

- с использованием особого соотношения заданной и искомой систем счисления.

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

***1.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 *.* (1.3)

Введем скобочную форму для выражения (1.3):

*Nq*1 = *Nq*2 = *а*0 + *q*2 (*а*1 +q2 (*а*2 +*q*2 (*а*3 + . +*q*2(*an*-1 + *q*2(*аn*))…));

Обозначим выражение в первой скобке как *N*1, выражение во второй скобке как *N*2, в третьей – как *N3* и т. д., выражение в (*n–*1)*-*й скобке – как *N*(*n–*1), выражение в *n-*й скобке – как *Nn*. Теперь, основываясь на выражении (1.3), можно утверждать, что при делении *Nq*1*/q*2 будет получена целая часть частного int(*Nq*1/*q*2) и остаток rest(*Nq*1/*q*2). Это можно записать:

*Nq*1/*q*2 → int(*Nq*1/*q*2) — целая часть частного *N*1, и остаток rest(*Nq*1/*q*2), равный *а*0.

Аналогично для остальных скобок:

*N1*/*q* 2 → int(*N*1/*q*2) равное *N*2 и остаток rest(*N*1/*q*2), равный а1;

*N*2/*q*2 → int(*N*2/*q*2) равное *N*3, и остаток rest(*N*2/*q*2), равный а2;

*N*(*n–*2)/*q* 2 → int(*N*(*n–*2)/*q*1) равное *N*(*n–1)*, остаток rest(*N*(*n* = 2)/*q*1), равный а(*n–*2);

*N*(*n–1*)/*q*2 → int(*N*(*n–*1)/*q*2) = *Nn* = а*n* и остаток rest(*N*(*n–*1)/*q*2), равный

*а*(*n–1*), при этом *Nn*< *q*2.

Отсюда вытекает правило формирования коэффициентов полинома (1.3) 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.

*Пример*

Найти запись в двоичной форме десятичного числа *N*10 = 436.

*Решение*

Делим сначала исходное число *N*10, а затем получаемые частные на значение нового основания 2 до получения частного со значением, меньше чем 2:

436/2 → int(436/2) = 218 и rest (436/2) = 0;

218/2 → int(218/2) = 109 и rest (218/2) = 0;

109/2 →int(109/2) = 54 и rest (109/2) = 1;

54/2 → int(54/2) = 27 и rest (54/2) = 0;

27/2 → int(27/2) = 13 и rest (27/2) = 1;

13/2→int(13/2) = 6 и rest (13/2) = 1;

6/2 → int(6/2) = 3 и rest (6/2) = 0;

3/2 → int(3/2) = 1 и rest (3/2) = 1.

Таким образом: 436 = 11 0110100.

***2. Преобразование дробных чисел***

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)

Введем скобочную форму для выражения (1.4). Обозначим выражение в первой скобке как *M*1, выражение во второй *–* как *M*2, в третьей скобке *–* как *M*3 и т. д., выражение в (*n –* 1)*-*й скобке как *M* *n–*1, выражение в *n-*й скобке – как *Mn*:

*Mq*2 *= Mq*1 *= q*2*–*1(*B*1 *+ q*2*–*1(*B*2 *+ q*2*–*1(*B*3*+...+q*2*–*1(*Bn–*1 *+ q2–*1(*Bn*))...))).

Число *Mq*1 *–* правильная дробь, поэтому при умножении *Mq*1 *×* *q*2 будет получено произведение, в общем случае состоящее из целой части int(*Mq*1 *× q*2) и дробной части DF (*M* *q*1 *×* q2). Использование введенных обозначений позволяет записать:

*Mq*1 *×* *q*2 *=* (int(*M* *q*1 *× q*2) *= B*1) *+* (DF(*Mq*1 *×* *q*2) *= M*1 ),

аналогично для остальных скобок будем иметь следующее:

*M*1 *× q*2 = (int(*M*1 *× q*2) = *B*2) + (DF(*M*1 *× q*2) = *M*2);

*M*2 *× q*2 = (int(*M*2 *× q*2) *= B*3) + (DF(*M*2 *× q*2) *= M*3);

*M*3 *× q*2 = (int*(M*3 *× q*2) *= B*4) *+* (DF(*M*3 *× q*2) *= M*4);

*Mn–*2 *× q*2 = (int(*Mn*–2 *× q*2) *= Bn–*1) *+* (DF*(Mn–*2 *× q*2) *= Mn–*1);

*Mn–*1*× q*2 *=* (int(*Mn–*1 *× q*2) *= Bn*) *+* (DF(*Mn–*1 *× q*2) *= Mn*);

*Mn × q*2 = (int(*Mn× q*2) *= Bn+*1) *+* (DF(*Mn × q*2) *= Mn*+1).

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.

*Пример*

Найти запись в двоичной форме десятичного числа *M*10 = 0,7.

*Решение*

Определяем количество разрядов числа *M*2. Так как исходная запись числа содержит один десятичный разряд, то запись данного числа в двоичном основании должна содержать четыре разряда. Учитывая округление, ищется предварительный двоичный эквивалент с пятью разрядами.

Умножаем исходное число *M*10, а затем дробные части последовательно получаемых произведений на новое основание 2. Выполняется пять таких операций умножения, в результате получаем:

0,7 *×* 2 = 1,4 (int(0,7 *×* 2) = 1 и DF (0,7 *×* 2) = 0,4);

0,4 *×* 2 = 0,8 (int(0,4 *×* 2) = 0 и DF(rest (0,4 *×* 2) = 0,8);

0,8 *×* 2 = 1,6 (int(0,8 *×* 2) = 1 и DF(rest (0,8 *×* 2) = 0,6);

0,6 *×* 2 =1,2 (int(0,6 *×* 2) = 1 и DF(rest (0,6 *×* 2) = 0,2);

0,2 *×* 2 = 0,4 (int(0,2 *×* 2) = 0 и DF(rest (0,2 *×* 2) = 0,4).

Таким образом, 0,7 = 0,10110, а окончательный результат перехода в двоичную систему будет 0,710 = 0,10112.

**4.*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.

***Пример:***Найти двоичный эквивалент восьмеричного числа 67401.648.

*Решение*

Основания исходной и новой систем счисления можно выразить через целую степень:23 = 8.

Поэтому применяем третий метод для случая перехода из системы с большим основанием в систему с меньшим основанием. Ставим в соответствие каждой цифре исходной записи числа трехразрядный двоичный код (*триаду*):

6 7 4 0 1 6 4

110 111 100 000 001 110 100

Формируем окончательный результат посредством объединения полученных трехразрядных двоичных чисел в единый двоичный эквивалент:

67401.648 = 110111100000001.110100.

***Пример*** Найти шестнадцатеричный эквивалент двоичного числа N =11100101110110.1110110012.

Решение

Основания исходной и новой систем счисления можно выразить через целую степень:24=16.

Поэтому применяем третий метод для случая перехода из системы с меньшим основанием в систему с большим основанием. Разбиваем исходную запись числа на группы по четыре разряда (*тетрады*) вправо и влево от точки, в крайних левой и правой группах недостающие разряды заполняем нулями и каждой полученной группе из четырех разрядов ставим в соответствие цифру шестнадцатеричной системы счисления

0011 1001 0111 0110 . 1110 1100 1000

3 9 7 6 EC 8

Формируем окончательный результат посредством объединения полученных цифр в единый шестнадцатеричный эквивалент 11100101110110.1110110012 = 3976.EC816.

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

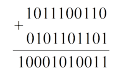
При выполнении любой операции результат ищется согласно соответствующим правилам, которые удобно представлять в табличной форме, где для всех возможных комбинаций значений одноразрядных операндов приводятся значения результата.

*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

*Пример*



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 машинной реализации операции сложения сначала формируется поразрядная сумма операндов без учета переноса, далее формируется код переноса и затем с помощью специальных логических цепей учитываются возникшие переносы. При этом перенос, возникший в некотором разряде, может изменить не только ближайший старший разряд, но и целую группу старших разрядов. В худшем случае перенос, возникший в самом младшем разряде, может изменить значение старших разрядов сформированной поразрядной суммы вплоть до самого старшего.

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).

***6. Операция вычитания в 2-й СС.***

Правила вычитания в 2-й 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.*

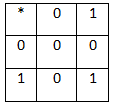
\_ 1000111001

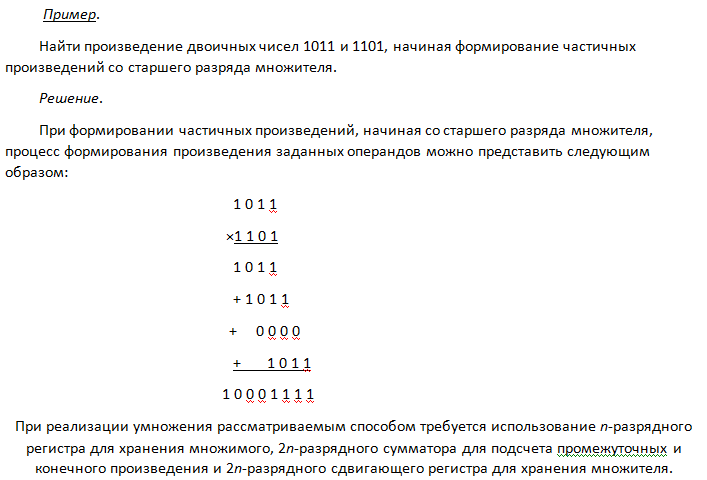
0101101101

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.

**7. Операция умножения в 2-й СС начиная со старшего разряда множителя со сдвигом множимого.**

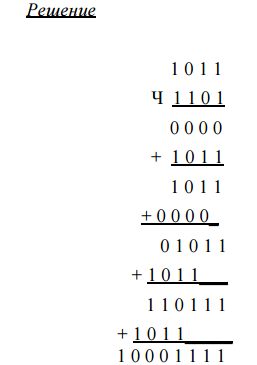


При умножении многоразрядных операндов, как правило (особенно в десятичной системе счисления), используется метод, при котором формирование произведения выполняется за счет суммирования частичных произведений, которые оформляются посредством умножения множимого на отдельные разряды множителя с учетом 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.

---------------------------------------------------------------------------------------------------------------

**8. 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;

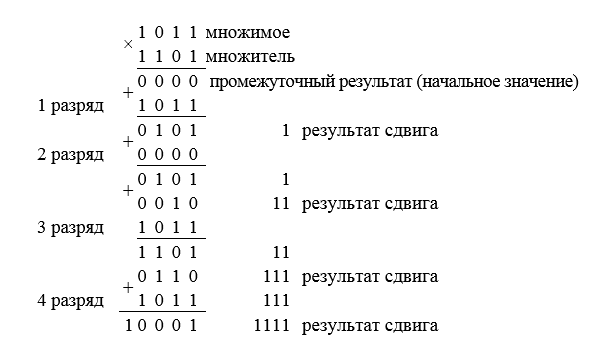
– **Умножение со старшего разряда множителя со сдвигом множимого**; при реализации данного метода требуется 2n-разрядный сумматор, 2n-разрядный регистр промежуточного произведения, 2n- разрядный регистр для хранения и сдвига множимого и n-разрядный регистр для хранения множителя;

– **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.

**9. Операция умножения в 2-й СС начиная со старшего разряда множителя со сдвигом промежуточных результатов**

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.

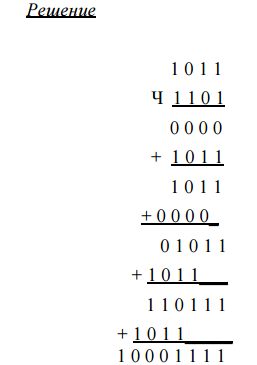
Для учета того, что очередное частичное произведение имеет вес, в два раза больший веса предыдущего частичного произведения, можно при суммировании сдвигать вправо промежуточный результат. В таком случае говорят: умножение выполняется со сдвигом промежуточного результата. При использовании данного подхода умножение чисел 1101 и 1011 представляется в виде следующих действий:



В первой колонке приведены номера отрабатываемых разрядов множителя начиная с младшего. Эти номера отмечают строки, в которых учитывается частичное произведение, соответствующее этому разряду множителя. В этой же первой колонке расположены единицы переполнения, возникающие при суммировании промежуточного результата и очередного частичного произведения, сформированного для соответствующего отрабатываемого разряда множителя (в данном случае единица переполнения имеется при отработке четвертого, самого старшего, разряда множителя). Вторая колонка отражает длину основной разрядной сетки (n = 4). В третьей колонке представлены разряды промежуточных и конечного произведений, «вытолкнутых» за пределы основной разрядной сетки в процессе выполнения очередного сдвига промежуточного произведения. Из приведенного примера видно, что выталкиваемые за пределы разряды промежуточных произведений в дальнейшем не изменяются и их значение не влияет на значение суммы, формируемой в пределах основной разрядной сетки. Поэтому для реализации этого метода умножения требуется n-разрядный сумматор, обеспечивающий суммирование только в пределах основной разрядной сетки. Аналогично умножению начиная с младшего разряда множителя, при умножении со старших разрядов можно заменить сдвиг вправо множителя на сдвиг влево промежуточного произведения. Операция умножения в общем случае дает точный результат – 2nразрядное произведение, где n-разрядность операндов.

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

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.

**11. Деление двоичных чисел.**

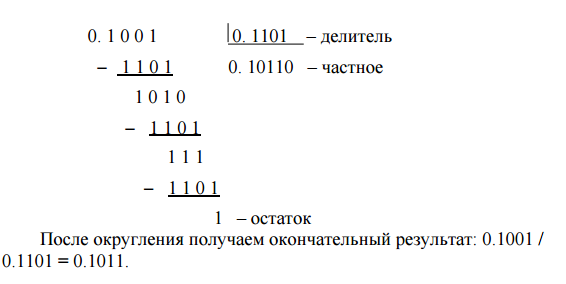
Деление в принципе является неточной операцией, поэтому при её выполнении прежде всего устанавливается количество разрядов частного, которые подлежат определению. Деление в двоичной системе счисления может выполняться точно так же, как и в десятичной, однако формирования частного двоичных операндов реализуется гораздо проще, чем в десятичной системе, т.к.:

• Упрощается процедура подбора очередной цифры вследствие того, что в двоичной системе очередной цифрой может быть одна из двух - либо 0, либо 1;

• Упрощается процедура умножения найденной цифры частного на делитель.

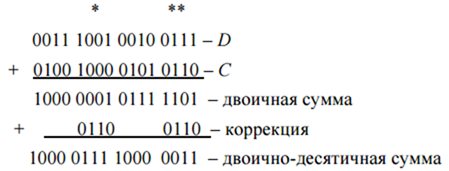
**Пример** Найти частное от деления двоичных чисел 0.1001 на 0.1101.

**Решение** По умолчанию считается, что разрядность результата и операндов одинаковая, поэтому окончательный результат должен иметь в данном случае 4 разряда. Учитывая необходимость округления, найдем дополнительный пятый разряд, на основании которого выполним округление.



**12. Сложение двоично-десятичных чисел**

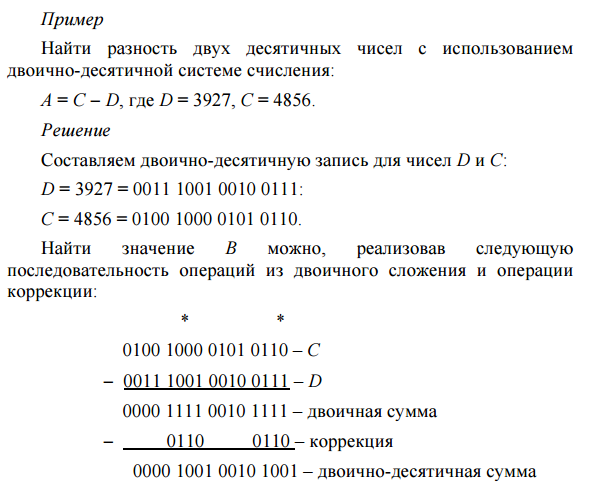
Рассмотрим на конкретном примере реализацию этой операции. Пример Найти сумму двух десятичных чисел с использованием двоично-десятичной системы счисления: A = D + C, где D = 3927; C = 4856. Решение Составляем двоично-десятичную запись для чисел D и C: D =3927 = 0011 1001 0010 0111: C = 4856 =0100 1000 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 \*\*)

**13. 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 \*).

***14 .Кодирование алгебраических чисел .***

Для представления чисел со знаком используются специальные коды:

- прямой код;

- дополнительный код;

- обратный код.

|  |  |
| --- | --- |
| Поле знака | Поле модуля |

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.

В *прямом(1) и дополнительном(2) коде* запись целого числа *А* формируется по следующим правилам:

(1)(2)

где *n* – разрядность модульного поля;

*q* – основание системы счисления;

*qn* – максимальная не включенная граница диапазона изменения представляемых чисел, т. к. диапазон изменение чисел *А* определяется как *qn* > |*А*| ≥ 0 .

Для случая правильной дроби запись числа *А* в дополнительном коде имеет вид:



где 1 – максимальная невключенная граница диапазона изменения представляемых чисел, т. е. диапазон изменение чисел *А* определяется как 1 > |*A* | ≥ 0.

В *обратном коде* запись целого числа *А* формируется по следующему правилу:



где *n* – разрядность модульного поля;

*q* – основание системы счисления;

(2*n* *-* 1) – максимальная включенная граница диапазона изменения представляемых чисел, т. е. диапазон изменения чисел *А* определяется как

(*qn* *–* 1) ≥ | *A* | ≥ 0.

Для случая правильной дроби запись числа *А* в обратном коде имеет вид



где *n –* разрядность поля модуля;

1 *- q-n* – верхняя включенная граница представляемых чисел.

Т. о., диапазон изменения чисел *А* определяется с. о.:

(1 *– q-n*) ≥ | *А* | ≥ 0.

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

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

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

***15. 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.

Представление двоичных чисел в прямом и обратном кодах поясняется следующими примерами.

**Пример**

Найди запись чисел **А = 532 и В = - 150** в прямом, дополнительном и обратном двоичных кодах.

**Решение**

Найдем **запись** заданных **чисел в двоичной системе:**

А = 532 = 10000101002, В = - 150 = - 100101102.

Если считать, что представляются в заданных кодах только А и В, то разрядность n модульного поля должна соответствовать разрядности двоичной записи большего числа, т.е. n = 10.

Найдем запись заданных чисел **в прямом коде:**

[A]пк = 0.1000010100, [B]пк = 1.0010010110.

Найдем запись заданных чисел в **дополнительном коде:**

[A]дк = 0.1000010100,

[B]дк: для определения модульной части **прибавим к невключенной границе диапазона** (2n = 10000000000) **число В:**

10000000000 +(-10010110) = 1101101010.

Тогда [B]дк = 1.1101101010.

Найдем запись **заданных чисел в обратном коде**:

[A]ок = 0.1000010100,

[B]ок: для определения модульной **части прибавим к включенной границе диапазона** (2n - 1= 1111111111) **число В**:

1111111111 + (- 10010110) = 1101101001,

и тогда [B]ок = 1.1101101001.

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.

***16. 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

Пример: у тебя есть число в двоичном представлении

A=100110 оно имеет 6 разрядов

B=11010101 оно имеет 8 разрядов  
Количество разрядов между двумя операндами должно совпадать, поэтому в прямом коде выглядит так: A=0.**00**100110

B=0.11010101

2. Избавься от операции вычитания, т.е если есть такая запись:  
С1 = А + В, С2 = А – В, С3 = В – А, С4 = –А – В, то нужно сделать так:  
С1 = А + В, С2 = А + (–В), С3 = В + (–А), С4 = (–А) + (-В)

Таким образом, у тебя будут величины: A, –A, B, –B

3. Далее свои величины A и B представляешь в дополнительном коде.

4. Используя сформированный дополнительный код, реализуем выражения для С1, С2, С3, С4.   
**НЕ ЗАБЫВАЙ!** Если при выполнении сложения у тебя возникла единица переполнения знакового поля. При работе с дополнительным кодом она игнорируется (в примере она подчёркнута).

С4: 1. 1 1 1 0 0 0 1 1 1 – [–A]дк

* 0. 0 1 1 0 1 0 0 1 0 – [–В]дк

-------------------------

**1** 0. 0 1 0 0 1 1 0 0 1 – [–С4]дк

КОНЕЦ.

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

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

**Пример**

Найти значения для *С*1, *С*2, *С*3, *С*4, определяемых выражениями

*С*1 = *А* + *В*, *С*2 = *А* - *В*, *С*3 = *В* - *А*, *С*4 = - *А* - *В*,

если А=5710, В = - 21010. При выполнении операций использовать двоичный обратный код. Результат представить в прямом коде.

**Решение**

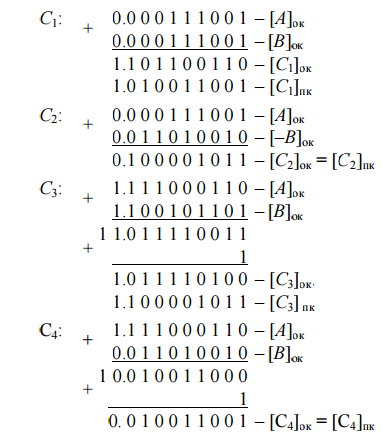
В данном примере используются те же выражения и те же операнды, что и в предыдущем примере, поэтому при его решение используются уже найденные ранее двоичные представления операндов и их прямые коды.

Обратные коды операндов имеют вид

[*А*]ок = 0.000111001, [-*А*]ок = 1.111000110,

[*В*]ок = 1.100101101, [-*В*]ок = 0.011010010.

Используя сформированный дополнительный код, реализуем выражения для *С*1, *С*2, *С*3, *С*4.



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

***18. Модифицированные коды***

Л

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

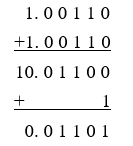
При несоответствии выбранной разрядности n диапазону изменения представляемых чисел при выполнении операции сложении чисел с одинаковыми знаками возможно появление ситуации переполнения, когда подлежащий представлению результат выходит за диапазон представления, определенный некорректно выбранной разрядностью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

Пример

Найти значения выражений

С1 = А + В, С2 = А - В, С3 = В - А, С4 = - А - В,

используя модифицированный обратный код, если

[А]пк = 0.1010011,

[B]пк = 1.0111001.

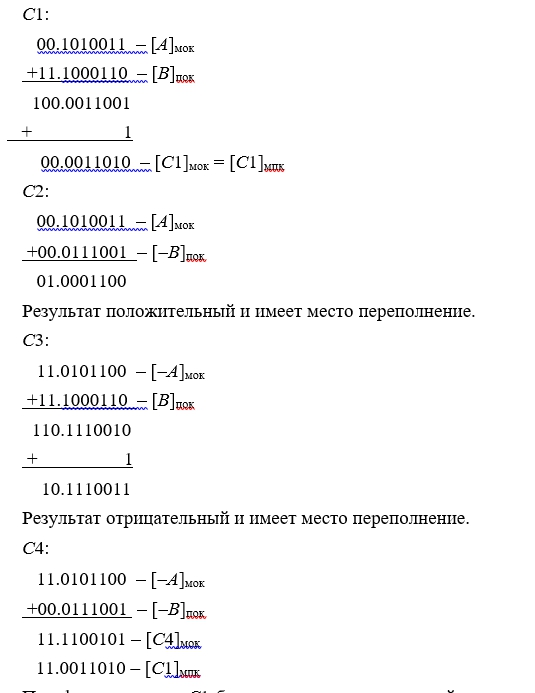
Решение

Модифицированный обратный код для всех операндов, используемых в приведенных выражениях, имеет вид

[А]мпк = 00.1010011, [А]мок = 00.1010011, [-А]мок = 11.0101100,

[B]мпк = 11.0111001, [B]мок =11.1000110, [-B]мок = 00.0111001.

Выполним действия, указанные в приведенных выражениях:



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.

**19. 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*.

**4.1. 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.

*Пример:*

10001101 ∨ 11110000 = 11111101.

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.

*Пример*:

10001101 ∧ 11110000 = 10000000

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».

*Пример:*

10001101 ⊕ 11110000 = 01111101.

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.

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

Для представления чисел со знаком используются специальные коды:

– прямой код;

– дополнительный код;

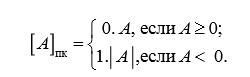
– обратный код.

Во всех трех случаях используется следующий формат представления числа, содержащий два поля – поле знака и поле модуля

|  |  |
| --- | --- |
| Поле знака | Поле модуля |

Общий формат представления числа в ЭВМ

В прямом коде запись целого числа А формируется по следующему правилу:



Сдвиг, при котором уходящий бит исчезает, не влияя на оставшиеся биты, а на месте появившегося бита записывается бит **0**. В большинстве процессоров уходящий бит сохраняется вофлаге переноса.

Арифметические сдвиги обеспечивают выполнение умножения (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.

***21. 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.

**Пример 1**

Выполнить арифметический сдвиг влево двоичного числа А = 11. 001010 (соответствует 1010), представленного в модифицированном прямом коде.

**Решение**

Заданный сдвиг, имеющий своей целью получение результата, в два раза превышающего по абсолютному значению значение исходного кода, дает в результате 11.010100 (2010), которое получается за счет логического сдвига влево только модульной части исходного кода.

Факт получения переполнения устанавливается по наличию единичного значения старшего разряда в сдвигаемом коде перед очередным сдвигом.

**Пример 2**

Выполнить арифметический сдвиг вправо двоичного числа А = 11.01110 (1410), представленного в модифицированном прямом коде.

**Решение**

Заданный сдвиг, имеющий своей целью получение кода, в два раза меньшего по абсолютному значению по отношению к значению исходного кода, дает в результате число 11.00111 (710), которое получается за счет логического сдвига вправо только модульной части исходного кода.

**22. 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.

***23.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.

**Пример1**

Выполнить сдвиг вправо на 2 разряда числа [А]пк = 10.01000110 (А10 = 326).

**Решение**

1-й сдвиг: 10. 01000110 → 11.10100011 (-163 в 10 с/c);

2-й сдвиг: 11.10100011 → 11.11010001 (-81 в 10 с/c) и последний вы-толкнутый разряд равен 1).

С учетом округления имеем окончательный результат:

[А2]пк=11.100100101

**Пример2**

Выполнить сдвиг вправо на 2 разряда числа [А]ок = 10.10111001 (А10 = – 326).

**Решение**

1-й сдвиг: 10.10111001 → 11.01011100 (-163 в 10 с/c);

2-й сдвиг: 11.01011100 → 11.10101110 (-82 в 10 с/c).

**Пример3**

Выполнить сдвиг вправо на 2 разряда число [А]ок= 10. 1011101

(А10 = - 326).

**Решение**

1-й сдвиг: 10.0111010 → 11.01011101 (-163 в 10 с/c);

2-й сдвиг: 11.01011101 → 11.10101110 (-81 в 10 с/c) и последний вытолкнутый разряд равен 1).

С учетом округления имеем окончательный результат

[А2]ок = 11.10101101.

**24. 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*

Рис. 1.3. 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*, для которого выполняется неравенство2*n* > ú*A*ú ≥ 0.

Одним из важнейших параметров представления чисел является ошибка представления. Ошибка представления может быть абсолютной (D) или относительной (d). Для фиксированной точки максимальные значения этих ошибок 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%.

5.2. 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 (подраздел 1.3). 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.

**25.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*, для которого выполняется неравенство:

2ú *х*пúmax × ú*х*мúmax = 2*p* × (1 - 2-*s*) ≥ ú*Х*ú ≥ 0 ,

где *p* = 2*k* - 1.

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:

1 > ú*x*мú ≥ 0.5.

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.

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

**27.Умножение чисел, представленных в формате с плавающей точкой.**

С точки зрения представления чисел с плавающей точкой поиск произведения

*С*2 =*А* × *В*

сводится к поиску *С*2п и *С*2м, соответственно порядку и мантиссе произведения на основании порядка *а*п и мантиссы *а*м множимого и порядка *в*п и мантиссы *в*м множителя. Учитывая общую запись чисел с плавающей точкой, произведение двух операндов представляется в виде

*С*2 = *А* × *В* = 2*а*п × *а*м × 2*в*п × *в*м = 2*а* п+вп × *а*м× *в*м = 2*с*

п × *с*2м.

Отсюда вытекает, что порядок произведения определяется как сумма порядков сомножителей, а мантисса произведения – как произведение мантисс сомножителей. Однако, учитывая возможность нарушения нормализации при умножении мантисс, в результате указанных действий будет найдено предварительное значения порядка и мантиссы искомого произведения, и окончательное значение произведения будет найдено только после устранения нарушения нормализации.

Таким образом, имеем:

*С*2п` = *а* п + *в* п;

*С*2м`= *а* м × *в*м.

Отсюда последовательность действий, обеспечивающих получение произведение двух чисел, заключается в следующем:

- определяется знак произведения как сумма по модулю двух знаковых разрядов мантисс сомножителей;

- определяется предварительное значение порядка произведения посредством суммирования порядков сомножителей;

- определяется предварительное значение мантиссы произведения как произведения мантисс операндов;

- устраняется нарушение нормализации мантиссы произведения (если нарушение имеет место) соответствующей корректировкой предварительного значения порядка и мантиссы искомого произведения.

При формировании мантиссы произведения нормализованных чисел с плавающей точкой возможен только один вид нарушения нормализации – нарушение нормализации справа от точки с появлением нуля только в старшем разряде мантиссы.

*Пример*

Найти произведение *С* чисел *А* и *В*, представленных с плавающей точкой, если *А* и *В* представлены в виде порядков, соответственно [*а*п]пк и [*в*п]пк и мантисс, соответственно [*а*м]пк и [*в*м]пк,

где [*а*п]пк = 1.010, [*а*м]пк = 1.1010, [*в*п]пк = 0.001, [*в*м]пк = 0.1001.

При выполнении операций использовать *обратный код*. При умножении мантисс использовать метод умножения, начиная *со старшего разряда* множителя со сдвигом промежуточного результата.

*Решение*

Знак искомого произведения, представляемого знаком его мантиссы, отрицательный, так как знаки мантисс сомножителей неодинаковые.

Предварительное значение порядка произведения определяется следующим образом:

*С*п' = *а* п + *в*п:

11.101 – [*а*п]мок

+ 00.001 – [*в*п]мок

11.110 – [*С*п`]мок

11.001 – [*С* п`]мпк, т.е. [*С*п`]пк = 1.001.

Абсолютное значение предварительного значения мантиссы произведения определяется следующим образом:

[*С*м`]:

0.1010 – ú*а*мú

× 0.1001 – ú*в*мú

.0000 – начальное значение промежуточного произведения

+ 1010 – первый младший разряд множителя равен единице

1010 – промежуточное произведение с учетом первого разряда

01010 – сдвинутое промежуточное произведение

001010 – второй разряд множителя равен нулю, поэтому выполняется только сдвиг

0001010 – третий разряд равен нулю, поэтому выполняется только сдвиг

+ 1010 – четвертый разряд равен единице

1011010 – промежуточное произведение с учетом старшего разряда

01011 010 – сдвинутое промежуточное произведение.

Таким образом,

[*С*м`]пк = 0.01011010.

С учетом округления имеем

[*С* м]пк = 0.01011.

Мантисса произведения ненормализованная, поэтому необходимо сдвинуть мантиссу влево на один разряд, а предварительное значение порядка произведения уменьшить на единицу. После нормализации с учетом ранее полученного знака окончательные значения мантиссы и порядка произведения будут следующими:

[*С*м]пк = 1.1011.

[*С*п]пк = 1.010.

**28. 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 |

**29.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

х1 + х2 + х3 + х4 = (х2 + х3) + х1 + х4 = (х1 + х4 ) + (х2 + х3)

– при логическом умножении отдельные логические сомножители можно заменить их произведением

х1 х2 х3 х4 = (х2 х3)х1х4 = (х1 х4) (х2 х3)

***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



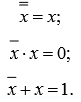
Если в качестве А используется простая конъюнкция, т. е. конъюнкция, представляющая собой логическое произведение переменных и их отрицаний, то имеет место



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.***





Законы, правила и операции алгебры Буля могут быть доказаны путем логического рассуждения, однако такое доказательство применимо только для простейших случаев. Доказать справедливость того или иного правила можно, если с помощью различных преобразований привести правую часть правила к выражению в левой части (или наоборот). Универсальным приемом доказательства является использование таблицы истинности. Это основано на том утверждении, что два выражения (правая и левая часть правила или закона) эквивалентны, если они принимают одинаковые значения на всех наборах логических переменных. Например, правило двойного отрицания, которое справедливо не только относительно одной переменной, но и любого логического выражения, можно доказать следующим рассуждением: если неверно утверждение, что выражение ложно, то очевидно утверждение, что это выражение истинно. Доказать справедливость распределительного закона в интерпретации выражением (х1 + х2)(х1 + х3) = х1 + х2х3 можно за счет приведения левой части к выражению правой части, раскрыв скобки:

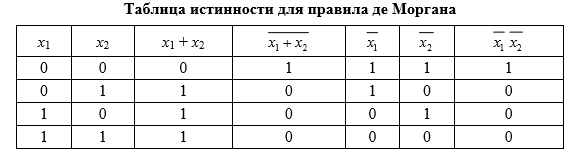
(х1 + х2)(х1 + х3) = х1х2 + х1х1 + х1х3 + х2х3 = = х1х2 + х1 + х1х3 + х2х3 = х1(х2 + 1 + х3) + х2х3.

Помня, что логическая сумма с одним слагаемым, равным константе «1», равна «1», можно записать х1 + х2х3.

Используем таблицу истинности для доказательства правила де Моргана в варианте отрицание суммы равно произве

дению отрицаний.

Составим таблицу истинности для правой и левой частей и составляющих их функций



Из таблицы истинности видно, что правая и левая части доказываемого правила принимают одинаковые значения на всех наборах, следовательно они эквивалентны.

***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».

**30. Канонические формы представления логических функций. Преобразование форм представления логических функций.**

Одну и ту же логическую функцию можно представить различными логическими выражениями. Среди множества выражений, которыми представляется логическая функция, особое место занимают две канонические формы:

* Совершенная конъюнктивная нормальная форма (СКНФ)
* Совершенная дизъюнктивная нормальная форма (СДНФ)

Совершенная дизъюнктивная нормальная форма представляет собой дизъюнкцию (дизъюнкция - логическое ИЛИ - даёт значение 1 когда хотя бы один из элементов равен 1, иначе 0) простых конъюнкций (конъюнкция - логическое И - результат операции равен 1, если все операнды равны 1, в остальных случаях результат равен 0), где под термином простая конъюнкция имеется в виду конъюнкция переменных или их отрицаний. В СДНФ простые конъюнкции содержат все переменные в своей прямой или инверсной форме и отражают собой наборы, на которых представляемая функция имеет единичное значение. Такие конъюнкции называются конституентами единицы рассматриваемой функции. Поэтому СДНФ представляет собой дизъюнкцию (логическую сумму), слагаемыми которой являются конституенты единицы. Общая запись СДНФ функции «y» имеет вид

y1 = ;

y2 = ;

y3 = .

Совершенная конъюнктивная нормальная форма - это конъюнкция простых дизъюнкций, где под термином простая дизъюнкция имеется в виду дизъюнкция переменных или их отрицаний. В СКНФ простые дизъюнкции содержат все переменные в своей прямой или инверсной форме и представляют собой отрицание конституент нуля. Общая запись СКНФ функции «y» имеет вид:

y1 = ;

y2 = ;

y3 = .

СКНФ строится на основе конституентa нуля. Конституент нуля представляет набор логических переменных, на котором логическая функция принимает значение «0». Каждая скобка в приведенных выражениях представляет собой отрицание конституенты нуля соответствующей функции, а запись функции в виде конъюнкция таких скобок представляет собой условие, при котором отсутствуют все конституенты нуля определяемой функции, при выполнении которого функция имеет единичное значение.

Например, для y1 выражение первой скобки представляет собой отрицание набора значений переменных второй строки, на котором функция y1 имеет нулевое значение, выражение второй скобки представляет собой отрицание набора значений переменных четвертой строки, на котором функция y1 также имеет нулевое значение, выражение третьей скобки представляет собой отрицание набора значений переменных пятой строки, на котором функция y1 имеет нулевое значение.

Из вышеизложенного следует, что любую функцию можно представить или в СДНФ, или в СКНФ, а так как эти формы представлены в базисе Буля, то отсюда значит, что этот базис (базис И, ИЛИ, НЕ) является функционально полным.

Если функция задана в СДНФ и требуется найти ее СКНФ, то такой переход можно выполнить, составив по заданной СДНФ таблицу истинности для этой функции, а на основе полученной таблицы составить СКНФ заданной функции.

Однако в некоторых случаях может оказаться более удобным подход, который поясняется следующим примером.

По заданной СДНФ функции y3 = 

Найти запись этой функции в СКНФ.

Решение:

Запишем логические выражение отрицания заданной функции, т.е. найдем логическое условие, при котором эта функция имеет нулевое значение. В качестве такого выражения можно взять дизъюнкцию конъюнкций, где каждая конъюнкция представляет собой конституенту нуля заданной функции. Очевидно, что конституенты нуля это те наборы, которые не являются наборами, соответствующими конституентам единицы, которые использованы в СДНФ. Таким образом, можно записать:



Эту запись можно интерпретировать как словесное описание функции: функция у равна нулю, если имеет место хотя бы одна из конституент нуля.

В этой записи представлена дизъюнкция тех наборов, которые не использовались в записи функции y3. Возьмем отрицание правой и левой частей полученного уравнения и применим к правой части правило де Моргана.



Применим правило Де Моргана к отрицаниям конъюнкций, полученным в правой части:

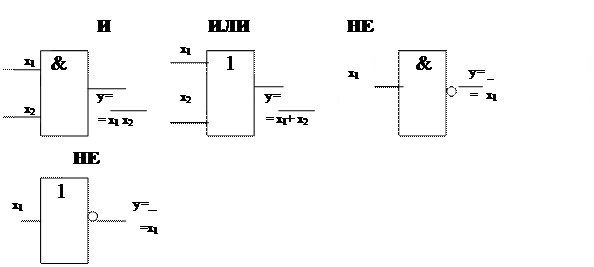
.

P.S. Правила Де Моргана:

Отрицание суммы равно произведению отрицаний: 

Отрицание произведения равно сумме отрицаний: 

**31. Синтез логических схем по логическим выражениям в булевом базисе.**

Логические схемы строятся на основе логических элементов, набор которых определяется заданным логическим базисом.  
  
Для базиса Буля в качестве логических элементов используются элементы, реализующие базовые логические функции И, ИЛИ, НЕ, которые имеют приведенные обозначения.  
  
При синтезе схемы по логическому выражению, составляющие логические операции представляются в виде соответствующих логических элементов, связи между которыми определяются последовательностью выполнения логических операций в заданном выражении.  
Пример:  
Синтезировать логическую схему в базисе И, ИЛИ, НЕ, реализующую логическое выражение:

*Решение*

Входными сигналами синтезируемой схемы являются x1, x2, x3, а выходным - y1.

Реализацию заданного выражения в виде логической схемы можно начать или с последней операции, или с первой.

Последней операцией в заданном выражении является операция логического сложения двух операндов

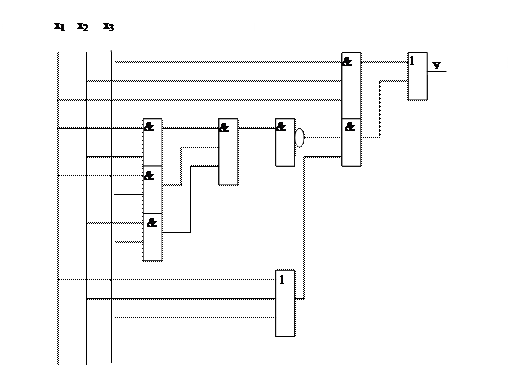
|  |
| --- |
| y= ( x1x2 +x1x3 +x2 x3 )(x1 +x2 +x3 ) и х1 x2 x3, |

поэтому для её реализации требуется элемент ИЛИ(1) с двумя входами, на выходе которого будет сформирован сигнал, соответствующий y1, если на его входы будут поданы эти два слагаемые (например, первое слагаемое на второй вход, а второе слагаемое на первый вход).

На первый вход ИЛИ(1) подается логическое произведение х1 x2 x3, для реализации которого необходимо использовать логический элемент И с тремя входами, на которые подаются входные переменные х1, x2, x3. Аналогичным образом рассматривается последовательность формирования выражения

|  |
| --- |
| y= ( x1x2 +x1x3 +x2 x3 )(x1 +x2 +x3 ), |

которое соответствует сигналу, подаваемому на второй вход элемента ИЛИ(1). В результате синтезируется схема для заданного выражения, приведенная на рис.2.2-2 .



**32. Минимизация логических выражений методом Квайна.**

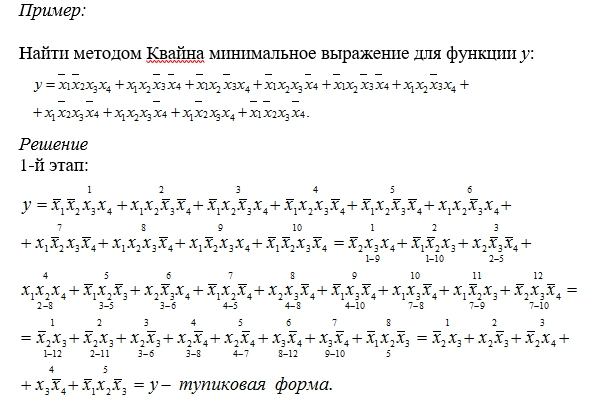
В качестве исходной формы представления логического выражения используется СДНФ. Если подлежащее минимизации выражение имеет другую форму, то приведение к СДНФ осуществляется за счет открытия скобок, избавления от отрицаний логических выражений, более сложных чем отрицание переменной (используется правило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

Таблица 2.7

**Импликантная таблица**

|  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
|  | \_ \_  *х*1*х*2*х*3*х*4 | \_ \_  *х*1*х*2*х*3*х*4 | \_ \_  *х*1*х*2*х*3*х*4 | \_ \_  *х*1*х*2*х*3*х*4 | \_ \_ \_  *х*1*х*2*х*3*х*4 | \_  *х*1*х*2*х*3*х*4 | \_ ­\_  *х*1*х*2*х*3*х*4 | \_  *х*1*х*2*х*3*х*4 | \_  *х*1*х*2*х*3*х*4 | \_ \_ \_  *х*1*х*2*х*3*х*4 |
|  | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | 10 |
| *\_*  *х*2*х*3 | \* |  |  |  |  |  | \* |  | \* | \* |
| \_  *х*2*х*3 |  | \* | \* |  | \* | \* |  |  |  |  |
| \_  *х*2*х*4 |  | \* |  | \* | \* |  |  | \* |  |  |
| \_  *х*3*х*4 |  |  |  | \* |  |  | \* | \* |  | \* |
| \_ \_  *х*1*х*2*х*3 |  |  | \* |  | \* |  |  |  |  |  |

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

Строки таблицы помечены простыми импликантами полученной тупиковой формы.

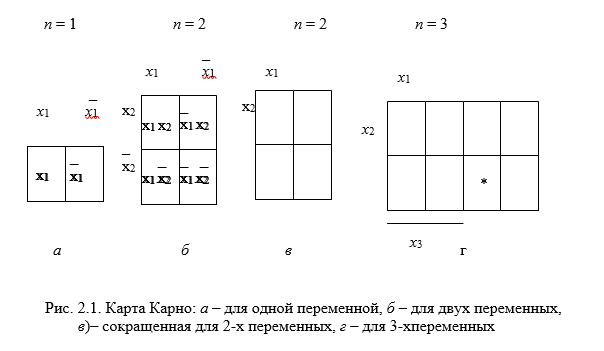
Звездочками в каждой строке отмечены те конституенты единицы, которые покрываются соответствующей простой импликантой (практически отмечаются те конституенты единицы, которые включают простую импликанту как свою составную часть)

**33. 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.

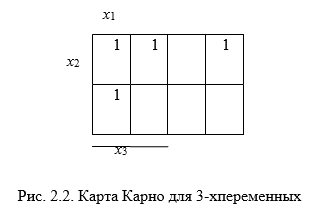
Запись минимального выражения по заданной функции имеет вид дизъюнкции простых конъюнкций, соответствующих контурам на карте, и формируется следующим образом:

- 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.



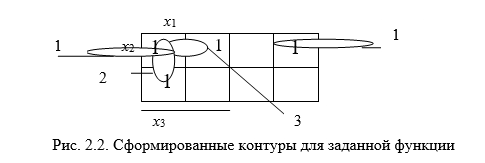
Для функции, заданной в карте Карно, приведенной на рис.2.2, контуры имеют вид, приведенный на рис.2.3.

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



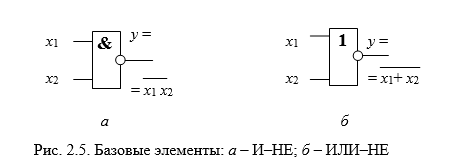
Конъюнкции минимального выражения помечены внизу цифрами, соответствующими номерам контуров, которые они представляют.

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

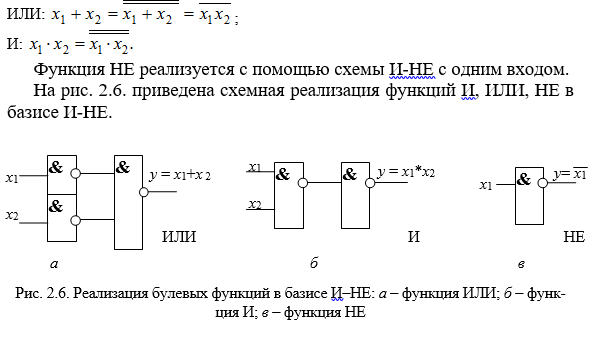
Булевый базис не является единственной функционально полной системой логических функций. Среди других наибольшее распространение получили базис И–НЕ и базис ИЛИ–НЕ.

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.



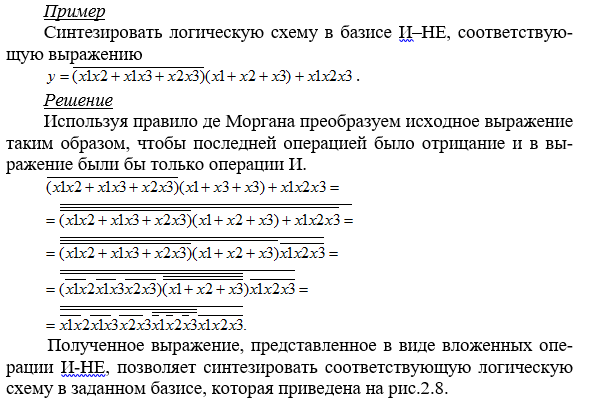
Реализация с помощью функции И-НЕ базовых функций алгебры Буля осуществляется следующим образом.

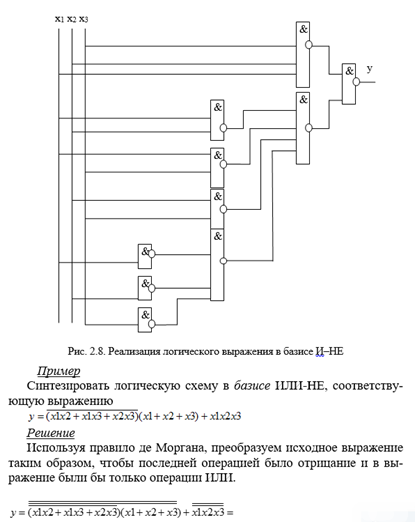


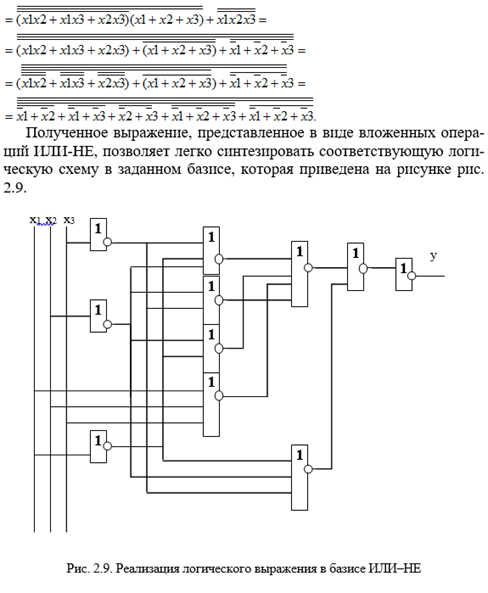
Реализация с помощью логической функции ИЛИ-НЕ базовых функций алгебры Буля осуществляется следующим образом.



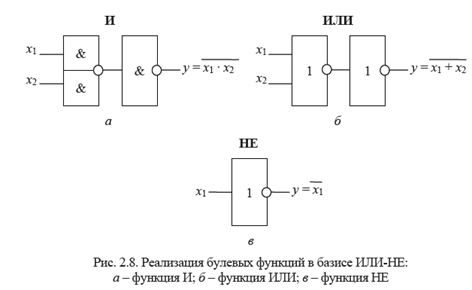
При синтезе логических схем в заданном базисе логических элементов (например, в базисах И–НЕ, или ИЛИ–НЕ) целесообразно предварительно исходное выражение привести к форме, в которой в выражении будут использованы только логические операции, соответствующие используемым логическим элементам в заданном базисе.







**Вопрос 35.Логический базис ИЛИ-НЕ.Синтез логических схем по логическому выражению в базисе ИЛИ-НЕ.**

Реализация с помощью логической функции ИЛИ-НЕ базовых

функций алгебры Буля осуществляется следующим образом.

ИЛИ: x1 + x2 =

И:

Функция НЕ реализуется с помощью схемы ИЛИ-НЕ с одним

входом.

На рис. 2.8. приведена схемная реализация операций И, ИЛИ, НЕ

в базисе ИЛИ-НЕ

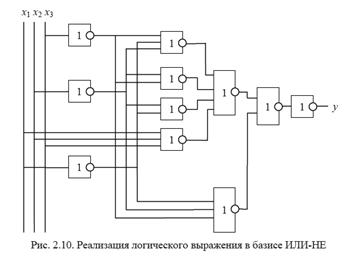
При синтезе логических схем в заданном базисе логических элементов (например, в базисах И–НЕ, или ИЛИ–НЕ) целесообразно предварительно исходное выражение привести к форме, в которой в выражении будут использованы только логические операции, соответствующие используемым логическим элементам в заданном базисе.

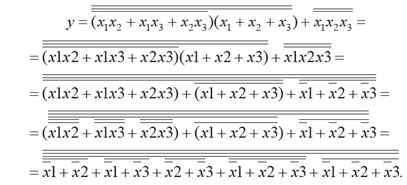
Пример:

Синтезировать логическую схему в базисе ИЛИ-НЕ, соответствующую выражению:



Решение

Используя правило де Моргана, преобразуем исходное выражение таким образом, чтобы последней операцией было отрицание и в выражение были бы только операции ИЛИ.



Полученное выражение, представленное в виде вложенных операций ИЛИ-НЕ, позволяет легко синтезировать соответствующую логическую схему в заданном базисе, которая приведена на рисунке рис. 2.10.

**36.Логические элементы.**

**Реализация простейших логических элементов с помощью резистивно-диодной логики.**

### Большинство современных ЦВМ состоят из двух и более архитектурных уровней (до шести).

Элементы ЭВМ, являющиеся представителями низшего иерархического уровня архитектуры компьютера – цифрового логического уровня (схемотехнического), реализованные на радиотехнических деталях, представляют собой мельчайшие компоненты, на основе которых строятся более крупные составляющие вычислительной машины.

- 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.

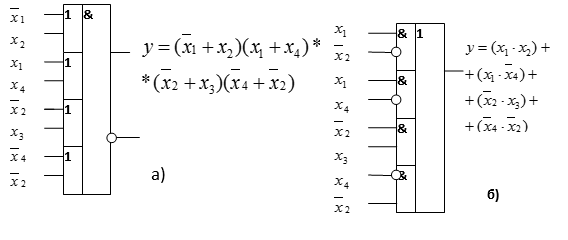
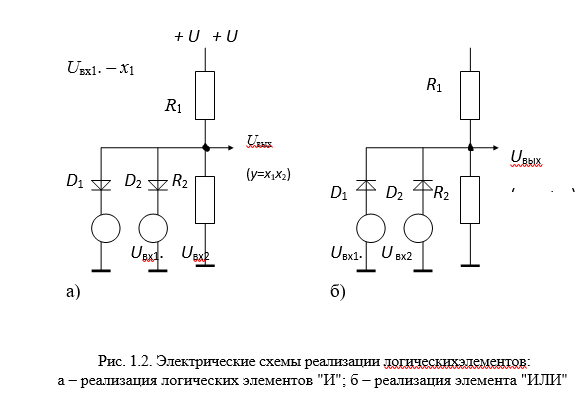


Рис. 1.1. Примеры логических элементов:

а – ИЛИ-И-НЕ; б – И-ИЛИ-НЕ

На выходах элементов указаны логические выражения для выходных сигналов в соответствии с приведенными входными сигналами. На рис. 1, б приведен логический элемент с инверсными входами (в логическом выражении сигнал по такому входу используется в обратном значении).

Примеры реализации простейших логических элементов с помощью диодно*-*резисторной схемы приведены на рис. 1.2.



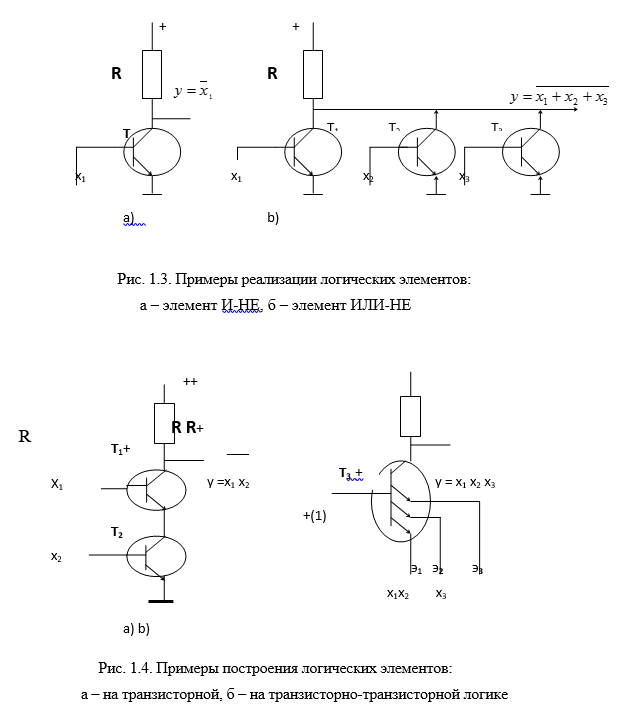
На рис.1.2,а приведена реализация логических элементов И. Реализация элемента ИЛИ приведена на рис.1.2,б. Схемы логических элементов построены с условием, что логическая «1» соответствует высокому уровню («+»), а логический ноль – низкому уровню напряжения, близкому «земле». Это соответствие используется и в других реализациях. На рис. 1.2,а соотношение сопротивления резисторов R1 и R2 при заданном напряжении «+U» выбирается таким образом, что без учета шунтирующего действия диодных цепочек напряжение на выходе имеет значение высокого уровня (уровня, соответствующего логической «1»). Источники входных сигналов Uвх1 и Uвх2 имеют малое внутреннее сопротивления. Поэтому, если один или оба источника подают низкий уровень (логический «0»), то из-за шунтирующего воздействия диодных цепочек на резистора R2 на выходе будет иметь место низкий уровень напряжения, соответствующий логическому нулю. Высокий уровень на выходе (логическая «1») будет иметь место только тогда, когда на оба входа подаются единицы, так как соответствующие им высокие уровни напряжения закрывают оба диода. Таким образом, единица на выходе будет иметь место только тогда, когда и x1, и x2 имеют единичные значения. Это означает, что рассматриваемая схема реализует логику И.

Для схемы на рис. 1.2,б соотношение сопротивления резисторов R1 и R2 при заданном напряжении «+U» выбирается таким образом, что без учета воздействия диодных цепочек напряжение на выходе имеет значение низкого уровня (уровня, соответствующего логическому «0»). Если хотя бы один или оба источника входных сигналов подают высокий уровень (логическая «1»), то этот высокий уровень проходит через открытый диод и появляется на выходе. Низкий уровень, т.е. логический «0», будет иметь место только тогда, когда оба входных сигнала имеют низкий уровень. Это означает, что рассматриваемая схема реализует логику ИЛИ.

На рис.1.3 приведены примеры реализации логических функций НЕ (рис. 1.3,а) и ИЛИ-НЕ (рис. 1.3,b) на транзисторах. Транзисторы обозначены символом «Т».

На рис. 1.3,а транзистор открыт, следовательно, на его коллекторе напряжение, близкое к нулевому уровню, тогда, когда на его базе высокий уровень логической единицы, и, наоборот, транзистор закрыт, а следовательно на его коллекторе высокий уровень тогда, когда входной сигнал соответствует низкому уровню нуля. Выходом схемы является коллектор транзистора, поэтому выходной сигнал реализует функцию НЕ.

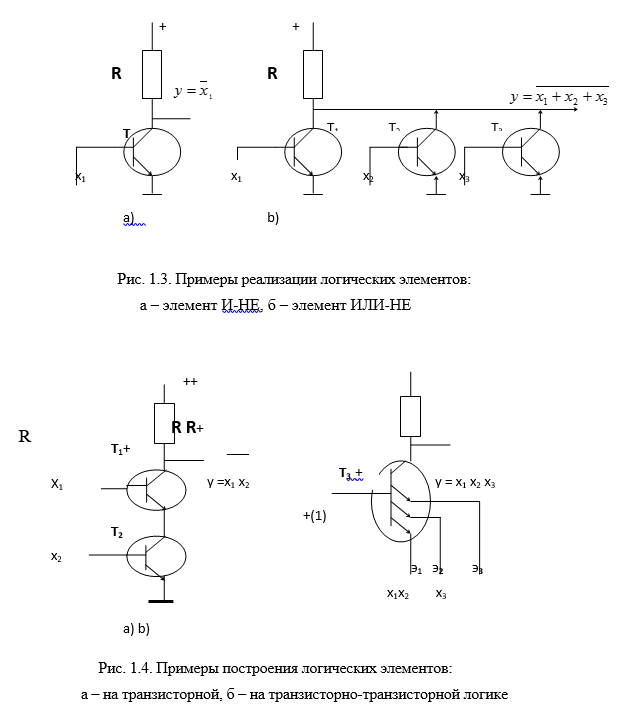
На рис. 1.3,б на выходе схемы «*y*» будет низкий уровень (логический нуль) тогда, когда открыт хотя бы один транзистор T1 , T2 , T3 , т.е. тогда, когда хотя бы одна из входных переменных *x*1, *x*2, *x*3, имеет значение логической единицы. Это означает, что выходной сигнал «y» зависит от входных сигналов по логике ИЛИ- НЕ.



На рис. 1.4 приведены примеры реализации логических функции И-НЕ и функции И на транзисторах.

На рис. 1.4,а на выходе схемы «*y*» будет низкий уровень (логический нуль) только тогда, когда открыты оба транзистора T1 , T2 , т.е. тогда, когда обе входные переменные *x*1, *x*2 имеют значение логической единицы. Это означает, что выходной сигнал «*y*» зависит от входных сигналов по логике И-НЕ.

На рис.1.4,б приведена схема, использующая многоэмиттерный транзистор T3 .Транзистор такого типа пропускает ток только тогда, когда имеет место высокий уровень на его базе и низкий уровень хотя бы на одном из его эмиттеров. В приведенной схеме на базу T3 подается постоянный высокий уровень (логическая константа, равная «1»). В этом случае на выходе схемы «y» будет низкий уровень (логический нуль) тогда, когда есть условия протекания тока хотя бы по одному из его эмиттеров, т.е. хотя бы одна из входных переменных x1, x2 , x3 имеет значение логического нуля. Если на все эмиттеры подается логическая единица, то T3 закрыт, а на выходе схеме имеет место высокий уровень, т.е. логическая единица.Это означает, что выходной сигнал «y» зависит от входных сигналов по логике И.



**37.Реализация простейших логических элементов на полупроводниковых триодах (транзисторах).**

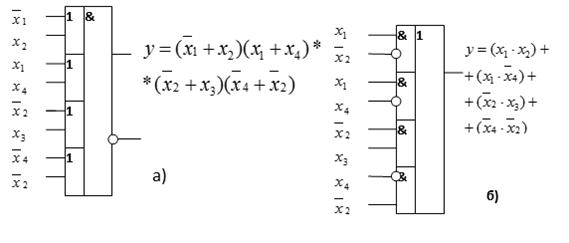
*Логические элементы*

*Большинство современных ЦВМ состоят из двух и более архитектурных уровней (до шести).*

Элементы ЭВМ, являющиеся представителями низшего иерархического уровня архитектуры компьютера – цифрового логического уровня (схемотехнического), реализованные на радиотехнических деталях, представляют собой мельчайшие компоненты, на основе которых строятся более крупные составляющие вычислительной машины.

- 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).



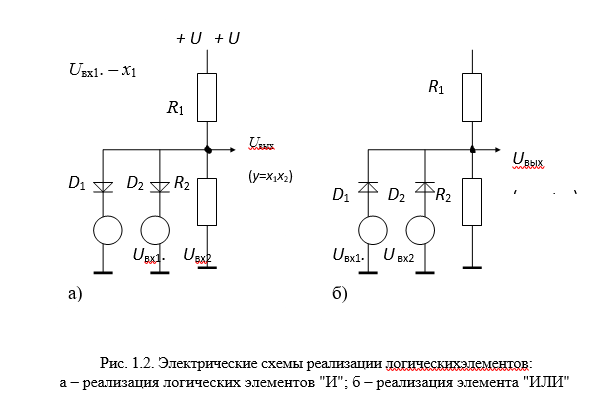
**На рис. 1.1**. приведены примеры некоторых логических элементов.

**Рис. 1.1.** Примеры логических элементов:

а – ИЛИ-И-НЕ; б – И-ИЛИ-НЕ

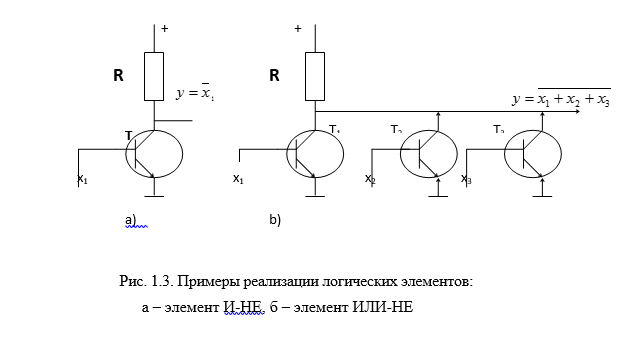
На выходах элементов указаны логические выражения для выходных сигналов в соответствии с приведенными входными сигналами. На **рис. 1, б** приведен логический элемент с инверсными входами (в логическом выражении сигнал по такому входу используется в обратном значении).

Примеры реализации простейших логических элементов с помощью диодно*-*резисторной схемы приведены на **рис. 1.2.**



**На рис.1.2,а** приведена реализация логических элементов И. Реализация элемента ИЛИ приведена на **рис.1.2,б.** Схемы логических элементов построены с условием, что логическая «1» соответствует высокому уровню («+»), а логический ноль – низкому уровню напряжения, близкому «земле». Это соответствие используется и в других реализациях. **На рис. 1.2,а** соотношение сопротивления резисторов R1 и R2 при заданном напряжении «+U» выбирается таким образом, что без учета шунтирующего действия диодных цепочек напряжение на выходе имеет значение высокого уровня (уровня, соответствующего логической «1»). Источники входных сигналов Uвх1 и Uвх2 имеют малое внутреннее сопротивления. Поэтому, если один или оба источника подают низкий уровень (логический «0»), то из-за шунтирующего воздействия диодных цепочек на резистора R2 на выходе будет иметь место низкий уровень напряжения, соответствующий логическому нулю. Высокий уровень на выходе (логическая «1») будет иметь место только тогда, когда на оба входа подаются единицы, так как соответствующие им высокие уровни напряжения закрывают оба диода. Таким образом, единица на выходе будет иметь место только тогда, когда и x1, и x2 имеют единичные значения. Это означает, что рассматриваемая схема реализует логику И.

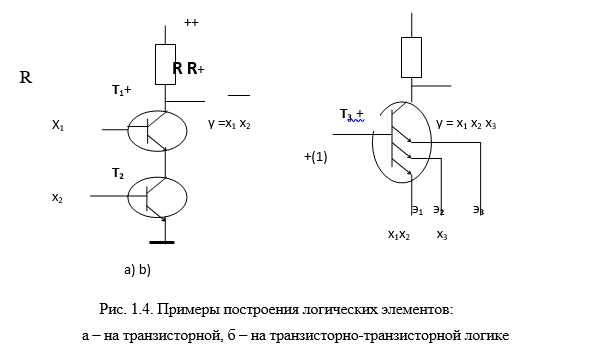
Для схемы на **рис. 1.2,б** соотношение сопротивления резисторов R1 и R2 при заданном напряжении «+U» выбирается таким образом, что без учета воздействия диодных цепочек напряжение на выходе имеет значение низкого уровня (уровня, соответствующего логическому «0»). Если хотя бы один или оба источника входных сигналов подают высокий уровень (логическая «1»), то этот высокий уровень проходит через открытый диод и появляется на выходе. Низкий уровень, т.е. логический «0», будет иметь место только тогда, когда оба входных сигнала имеют низкий уровень. Это означает, что рассматриваемая схема реализует логику ИЛИ.



На **рис.1.3** приведены примеры реализации логических функций НЕ (**рис. 1.3,а**) и ИЛИ-НЕ (рис. 1.3,b) на транзисторах. Транзисторы обозначены символом «Т».

На **рис. 1.3**,а транзистор открыт, следовательно, на его коллекторе напряжение, близкое к нулевому уровню, тогда, когда на его базе высокий уровень логической единицы, и, наоборот, транзистор закрыт, а следовательно на его коллекторе высокий уровень тогда, когда входной сигнал соответствует низкому уровню нуля. Выходом схемы является коллектор транзистора, поэтому выходной сигнал реализует функцию НЕ.

На **рис. 1.3,б** на выходе схемы «*y*» будет низкий уровень (логический нуль) тогда, когда открыт хотя бы один транзистор T1 , T2 , T3 , т.е. тогда, когда хотя бы одна из входных переменных *x*1, *x*2, *x*3, имеет значение логической единицы. Это означает, что выходной сигнал «y» зависит от входных сигналов по логике ИЛИ- НЕ.



**На рис. 1.4** приведены примеры реализации логических функции И-НЕ и функции И на транзисторах.

**На рис. 1.4,а** на выходе схемы «*y*» будет низкий уровень (логический нуль) только тогда, когда открыты оба транзистора T1 , T2 , т.е. тогда, когда обе входные переменные *x*1, *x*2 имеют значение логической единицы. Это означает, что выходной сигнал «*y*» зависит от входных сигналов по логике И-НЕ.

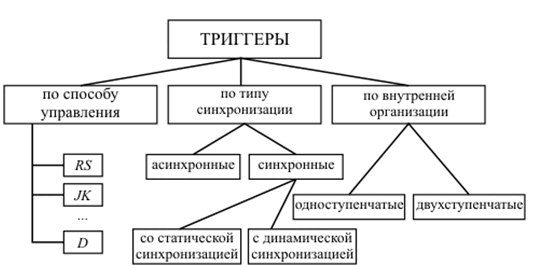
**На рис.1.4,б** приведена схема, использующая многоэмиттерный транзистор T3 .Транзистор такого типа пропускает ток только тогда, когда имеет место высокий уровень на его базе и низкий уровень хотя бы на одном из его эмиттеров. В приведенной схеме на базу T3 подается постоянный высокий уровень (логическая константа, равная «1»). В этом случае на выходе схемы «y» будет низкий уровень (логический нуль) тогда, когда есть условия протекания тока хотя бы по одному из его эмиттеров, т.е. хотя бы одна из входных переменных x1, x2 , x3 имеет значение логического нуля. Если на все эмиттеры подается логическая единица, то T3 закрыт, а на выходе схеме имеет место высокий уровень, т.е. логическая единица. Это означает, что выходной сигнал «y» зависит от входных сигналов по логике И.

**38. Запоминающие элементы. 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



Триггер называется синхронным, если его таблица переходов хотя бы по одному управляющему входу реализуется под воздействием синхронизирующего сигнала.

Основу синхронного одноступенчатого триггера составляет рассмотренная выше запоминающая ячейка (элементы 1, 2). Комбинационная схема преобразует управляющие сигналы триггера, а также, для некоторых типов триггеров, сигналы Q и с выходов ЗЯ в сигналы S и R на входах запоминающей ячейки.

Обобщенная схема синхронного одноступенчатого триггера:

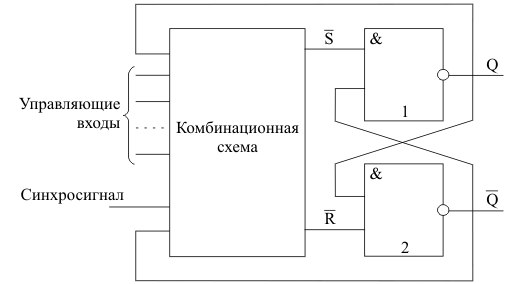
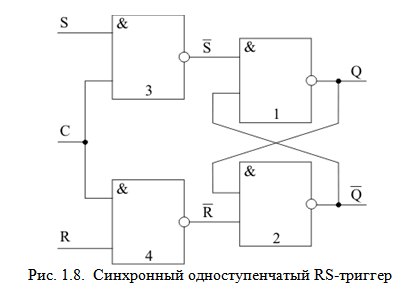
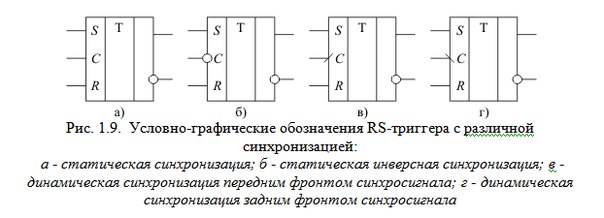


Рис. 1.7. Обобщенная схема синхронного одноступенчатого триггера

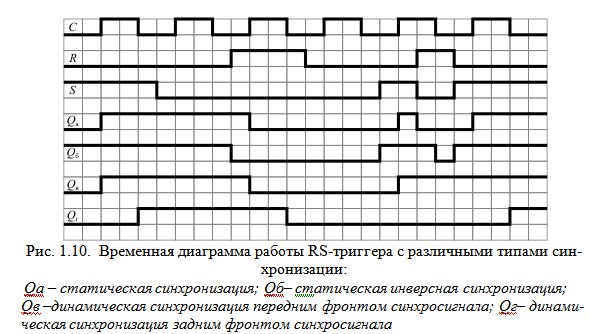
Схема синхронного одноступенчатого RS-триггера:



## УГО RS-триггера с различной синхронизацией:



Идеализированная (без учета задержек) временная диаграмма работы RS-триггеров с различными типами синхронизации:



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

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)..

**2.1 RS-триггер**

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:

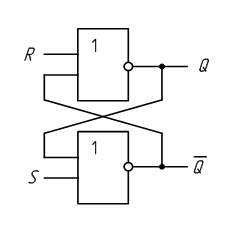
Q - прямой;

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

Состояние триггера определяется состоянием прямого выхода. Простейший RS-триггер состоит из двух логических элементов, охваченных перекрёстной положительной обратной связью (рисунок 2.1).

Рисунок 2.1 - Схема простейшего RS- триггера

Рассмотрим работу триггера:



Пусть R=0, S=1. Нижний логический элемент выполняет логическую функцию ИЛИ-НЕ, т.е. 1 на любом его входе приводит к тому, что на его выходе будет логический ноль Q=0. На выходе Q будет 1 (Q=1), т.к. на оба входа верхнего элемента поданы нули (один ноль - со входа R, другой - с выхода ). Триггер находится в единичном состоянии. Если теперь убрать сигнал установки (R=0, S=0), на выходе ситуация не изменится, т.к. несмотря на то, что на нижний вход нижнего логического элемента будет поступать 0, на его верхний вход поступает 1 с выхода верхнего логического элемента. Триггер будет находиться в единичном состоянии, пока на вход R не поступит сигнал сброса. Пусть теперь R=1, S=0. Тогда Q=0, а Q(со штрихом)=1. Триггер переключился в "0". Если после этого убрать сигнал сброса (R=0, S=0), то все равно триггер не изменит своего состояния.

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 | \* |

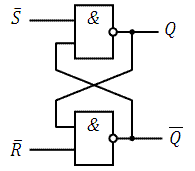
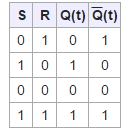
Триггер, построенный на базе элементов ИЛИ-НЕ, называют также дизьюнктивной бистабильной ячейкой. Бистабильные ячейки, помимо самостоятельного применения, входят в качестве составного узла в триггеры других типов.

**40.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.

При подаче единицы на вход **S** (от англ. *Set* — установить) выходное состояние становится равным логической единице. А при подаче единицы на вход **R** (от англ. *Reset* — сбросить) выходное состояние становится равным логическому нулю. Состояние, при котором на оба входа **R** и **S** одновременно поданы логические единицы не определено и зависит от реализации, например в триггере на элементах «или-не» оба выхода переходят в состояние логического «0», которое является неустойчивым и переходит в одно из устойчивых состояний при снятии управляющего сигнала с одного из входов.

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**

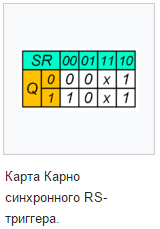
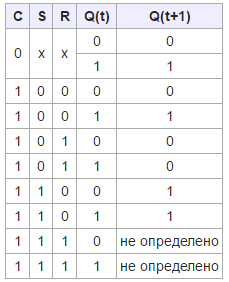
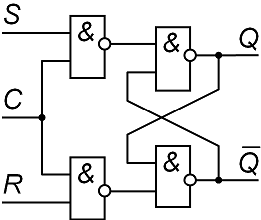
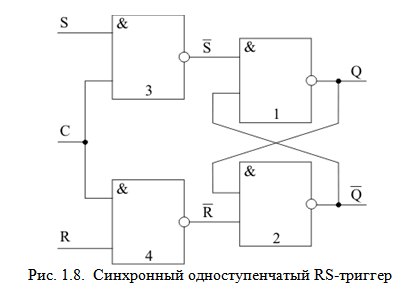


Схема синхронного RS-триггера совпадает со схемой одноступенчатого парафазного (двухфазного) D-триггера, но не наоборот, так как в парафазном (двухфазном) D-триггере не используются комбинации S=0, R=0 и S=1, R=1.

Алгоритм функционирования синхронного RS-триггера можно представить формулой

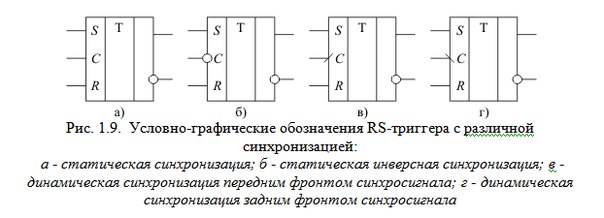
Безымянный.png где x — неопределённое состояние.

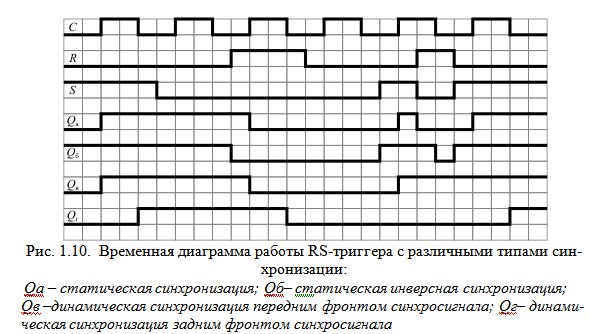
***41. Синхронный одноступенчатый RS-триггер. Условно-графическое обозначение RS-триггера с различной синхронизацией. Временная диаграмма работы RS-триггера с различными типами синхронизации.***



**Синхронный одноступенчатый RS-триггер** отличается от асинхронного наличием С-входа для синхронизирующих ( тактовых) импульсов. Синхронный триггер состоит из асинхронного - триггера и двух логических элементов на его входе.

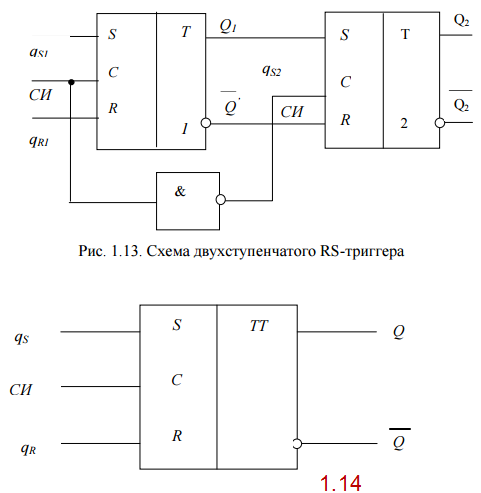
Особенность **одноступенчатых RS-триггеров** - записываемая в них информация практически сразу ( лишь с небольшой задержкой, обусловленной временем срабатывания) появляется на выходах, что не всегда желательно.

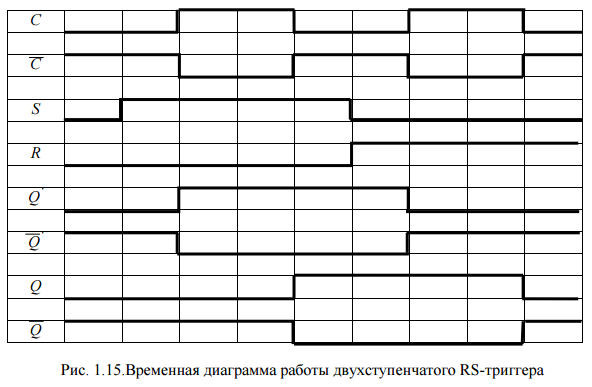




**42. Двухступенчатый RS-триггер.**

Двухступенчатый RS-триггер (рис. 1.13 и рис. 1.14) строится на основе двух одноступенчатых триггеров с прямой статической синхронизацией. Информация в первую ступень триггера (элемент 1) заносится во время действия высокого уровня синхросигнала. После того как синхросигнал на входе принимает низкое значение, элемент 1 переходит в режим хранения, а значение высокого сигнала на выходе инвертора 3 обеспечивает запись состояния триггера 1 в триггер 2. Идеализированная временная диаграмма работы двухступенчатого RS- триггера приведена на рис. 1.15

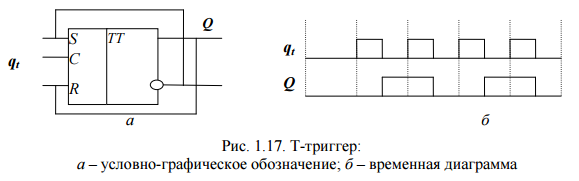




Следует отметить, что первая ступень представляет собой одноступенчатый триггер, реализующий заданную таблицу переходов, в то время как вторая ступень – это всегда одноступенчатый синхронный RS-триггер.

***43.T-, JK-, D-триггеры.***

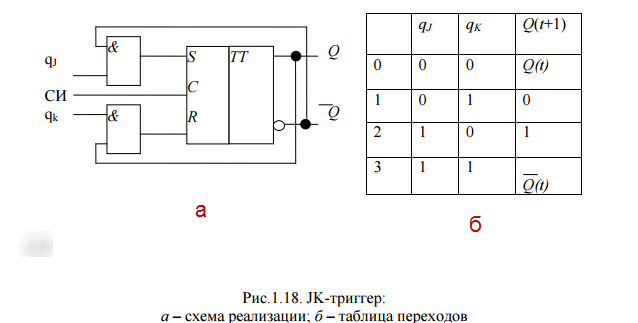
Т-триггер представляет собой триггер, имеющий один вход «Т», поступление единичного сигнала на который переводит Т- триггер в состояние, противоположное его исходному состоянию (фигурально говоря, по каждому входному сигналу триггер «кувыркается», меняя свое состояние на противоположное). На рис. 1.17 приведена реализация Т-триггера на базе двухтактного RS- триггера (а) и временная диаграмма его работы (б). Имеющиеся на схеме обратные связи создают ситуацию, при которой сигналы на входах R и S стремятся перевести триггер в состояние, противоположное текущему. Поэтому при приходе очередного сигнала qT триггер воспринимает сигналы, имеющиеся на его входах. Выходные сигналы триггера изменяются после снятия единичного сигнала на его входе qT, так как триггер двухтактный.



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».

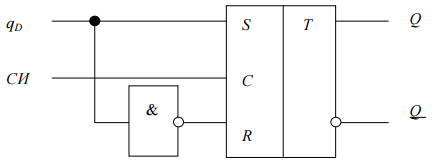
Из приведенной схемы видно, что сигналы qJили qK, стремящиеся установить триггер, соответственно, в «1» или «0», поступают на соответствующий вход «S» или «R» базового RS-триггера только тогда, когда его исходное состояние противоположно тому, в которое стремится перевести JK-триггер комбинация входных сигналов. В противном случае сигналы qJили qK на соответствующий вход S или R базового триггера не поступают. В связи с этим комбинация входных сигналов «1, 1» не является запретной, так как в этом случае на соответствующий вход базового RS-триггера поступит только тот сигнал, который стремится установить триггер в состояние, противоположное его исходному состоянию. Этот момент отражен в таблице истинности:при комбинации входных сигналов «1,1» триггер меняет исходное состояние, то есть работает как T-триггер



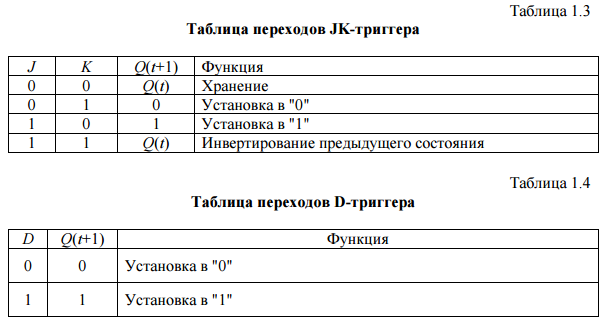
Таким образом, JK-триггер представляет собой универсальный триггер, объединяющий в себе свойства и RS-триггера и Т-триггера.

D-триггер. 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.



Это означает, что, когда есть «1» на входе qD, на входах базового триггера будут сигналы: «1»–на входе S и «0»–на входе «R». Поэтому по переднему фронту сигнала СИ в триггере устанавливается «1», если есть «1» на входе qD, в противном случае в триггере будет установлен «0». Состояние, которое имеется у триггера в момент заднего фронта сигнала СИ, будет сохраняться («задерживаться») до поступления очередного сигнала синхронизации СИ. Таблицы переходов JK- и D- триггеров приведены в таблиц.1.3 и 1.4 соответственно.



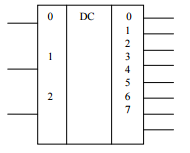
**44. Комбинационные узлы. Дешифратор. Шифратор.**

Узлы ЭВМ можно подразделить на два типа – комбинационные узлы и накапливающие.

Узел ЭВМ представляет 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 накапливающего 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 памяти, представленной некоторой совокупностью запоминающих элементов.

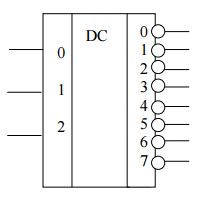
Дешифратором называется комбинационная схема, имеющая n входов и 2n выходов и преобразующая двоичный код на своих входах в унитарный код на выходах. Унитарным называется двоичный код, содержащий одну и только одну единицу, например 00100000. Условно-графическое обозначение дешифратора на три входа приведено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



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

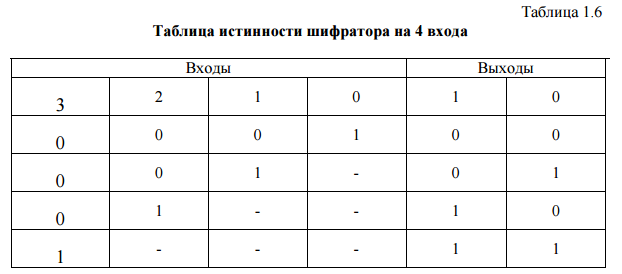
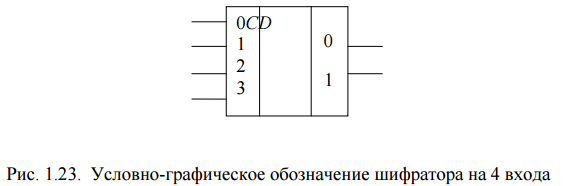


В большинстве реальных интегральных микросхем реализованы именно дешифраторы с инверсными выходами. Обозначение такого дешифратора показано на рис.:



На выходах такого дешифратора образуется унитарный код, содержащий один и только один ноль. Например, если входные сигналы имеют значение 1102=610, то выходы дешифратора, представленного на рис. 1.3, будут находиться в состоянии 10111111, то есть выход 6 будет иметь значение, отличное от остальных выходов. Дешифраторы широко применяются в различных устройствах компьютеров. Прежде всего, они используются для выбора ячейки запоминающего устройства, к которой производится обращение для записи или считывания информации. При этом часть разрядов адресного кода может дешифрироваться дешифраторами, выполненными в виде отдельных интегральных схем, а другая часть разрядов (обычно младшая) дешифрируется с помощью дешифраторов, встроенных непосредственно в БИС запоминающего устройства. Кроме того, дешифраторы находят применение в устройстве управления для определения выполняемой операции, построения распределителей импульсов и в других блоках.

Шифратор – схема, имеющая 2n (два в степени n) входов и n выходов, функции которой во многом противоположны функции дешифратора (рис. 1.23). Эта комбинационная схема в соответствии с унитарным кодом на своих входах формирует позиционный код на выходе (таблица 1.6)



**Вопрос 45.Комбинационные узлы. Сумматор по модулю 2**

Узлы ЭВМ можно подразделить на два типа – комбинационные узлы и накапливающие.

Узел ЭВМ представляет собой совокупность нескольких логических схем и, в общем случае, элементов памяти, формирующих выходные сигналы, соответствующие нескольким логическим функциям от входных сигналов.

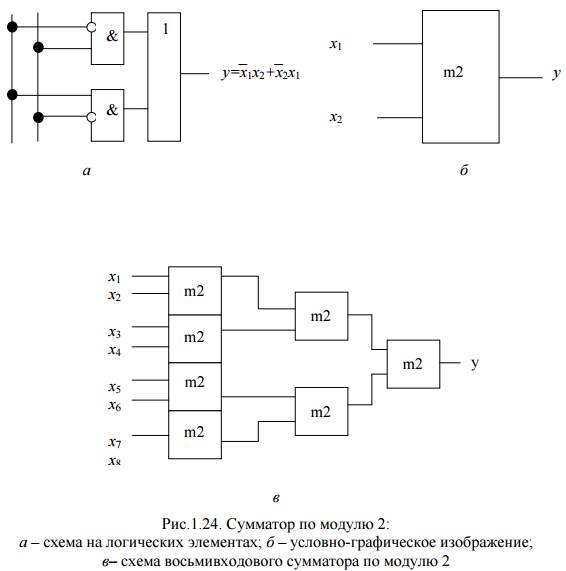
Характерной особенностью узлов комбинационного типа является то, что их выходные сигналы определяются только действующими в данный момент входными сигналами (не зависят от «истории» входных сигналов).

Характерной особенностью узлов накапливающего 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 памяти, представленной некоторой совокупностью запоминающих элементов.

Сумматор по модулю 2 вырабатывает на своем выходе сигнал логической единицы, если количество его входов с сигналом логической единицы нечетное. На рис. 1.24,а приведена схема сумматора по модулю 2 на два входа, а на рис. 1.24,б – его условное обозначение. На рис. 1.24,в приведен сумматор по модулю 2 с восемью входами, построенный по принципу каскадирования из двухвходовых сумматоров по модулю 2.

Сигнал на выходе схемы (см. рис. 1.3,в) у определяется логическим выражением

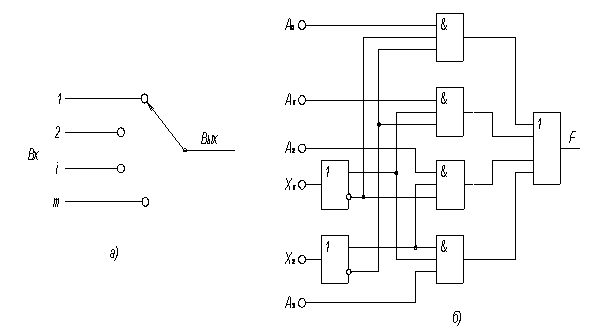




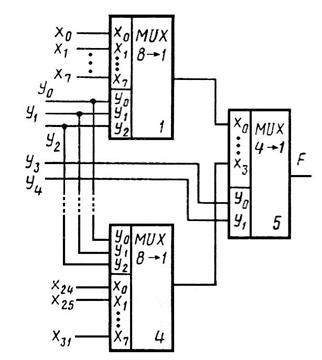
**46.Комбинационные узлы. Мультиплексор. Демультиплексор.**

Узлы ЭВМ можно подразделить на два типа:  
  
- комбинационные узлы;  
  
- накапливающие узлы.  
  
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.

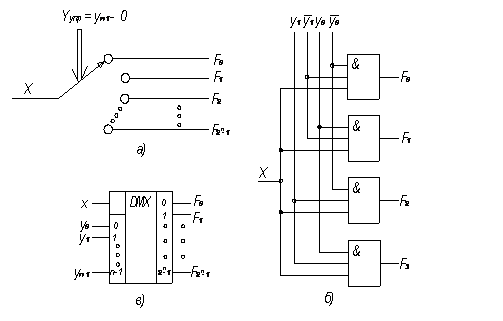


Наращивание размерности мультиплексора.



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.

*Демультиплексор* – коммутатор логических сигналов, обеспечивающий передачу информации, поступающей по одному входу, и в соответствии с адресом направляющий в одну из выходных линий.  
  
При передачах данных по общему каналу с разделением во времени нужны не только мультиплексоры, но и устройства обратного назначения, распределяющие данные из одного канала между несколькими приемниками информации. Эта задача реализуется демультиплексорами, функционирование которых можно пояснить с помощью рисунка а. Демультиплексор имеет один информационный вход n адресующих (управляющих) входов и 2n выходов.  
  
Построение демультиплексора «1>4» на элементах И показано на рисунке б. Работа демультиплексора описывается логическими выражениями:

, где mi – минтермы n адресующих переменных.  


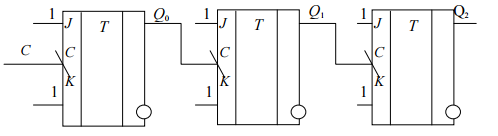
Принцип работы (а), схемная реализация (б) и условное обозначение для демультиплексора (в)  
В условных изображениях демультиплексор обозначается буквами ДМХ (рисунок в).  
Применительно к мультиплексорам и демультиплексорам пользуются также термином «селекторы данных».

**47.Счетчики. Схема, временная диаграмма работы, оценка быстродействия асинхронного счетчика.**

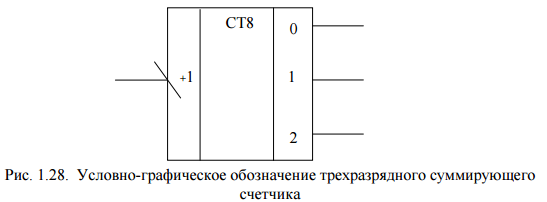
Счетчиком называется электронная схема, предназначенная для подсчета числа сигналов, поступающих на его счетный вход. Счетчики используются в устройстве управления компьютера при построении распределителей импульсов и организации циклов, в счетчиках команд для формирования адреса выполняемой команды при естественном порядке выполнения и в некоторых других устройствах ЭВМ. Также счетчики широко применяются в качестве самостоятельных узлов в различных системах цифровой автоматики.

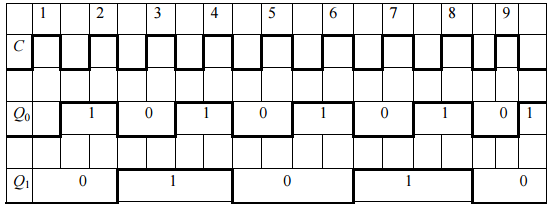
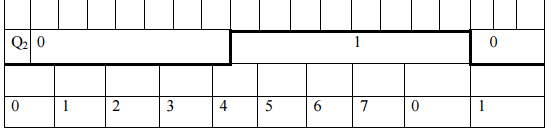
Суть работы счетчика заключается в изменении на единицу зафиксированного в нем значения с приходом каждого счетного сигнала. Счетчики принято подразделять на суммирующие, вычитающие и реверсивные. Суммирующие счетчики увеличивают свое значение, вычитающие – уменьшают, а реверсивные могут работать как на прибавление, так и на вычитание в зависимости от сигналов управления. Параметром, определяющим информационную емкость счетчика, является модуль пересчета, равный числу внутренних состояний. Это значение проставляется на УГО после аббревиатуры CT.

Пример асинхронного трехразрядного двоичного суммирующего счетчика приведен на рис. 1.27, а его условно-графическое обозначение – на рис. 1.28. Для построения этого счетчика использованы JK- триггеры с динамической синхронизацией по спаду синхросигнала. Каждый JK-триггер в счетчике включен в режим инвертирования своего состояния при переключении синхросигнала с высокого уровня на низкий (см. табл. 1.5). Идеализированная временная диаграмма работы этого счетчика показана на рис. 1.29



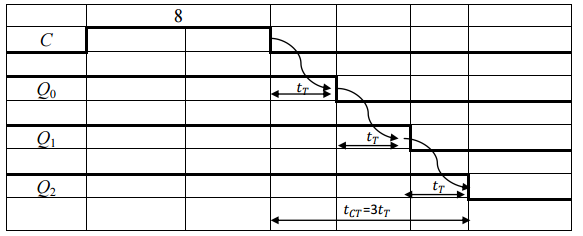






Быстродействие асинхронного счетчика определяется максимальной задержкой от изменения сигнала на его счетном входе до полного установления состояния всех его выходов. Проведем оценку быстродействия на примере переключения выходов счетчика после поступления восьмого синхросигнала на его вход (рис. 1.30), так как именно в этом такте время переключения выходов счетчика будет максимальным.





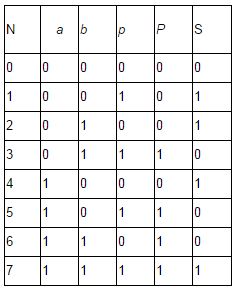
По фронту 1-го сигнала С(8) с задержкой сигнала, равной tT, на триггере Q0 (см. рис. 2.1) происходит изменение сигнала на выходе Q0. Это изменение, в свою очередь, приведет к переключению сигнала Q1 с соответствующей задержкой относительно переключения Q0. Вслед за этим с задержкой сигнала на следующем триггере переключится сигнал Q2. То есть общее время задержки переключения сигналов на выходе трехразрядного счетчика составит 3tT. Очевидно, что для n–разрядного счетчика время задержки составит:



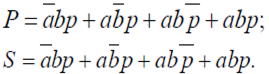
Таким образом, с увеличением разрядности асинхронного счетчика увеличивается его задержка и, следовательно, уменьшается быстродействие. Этот недостаток устраняется в счетчиках, работающих по синхронной схеме. В них за счет дополнительных управляющих комбинационных схем обеспечивается одновременное переключение всех разрядов при поступлении сигнала на счетный вход (с задержкой, равной задержке одного триггера вне зависимости от разрядности счетчика). Обычно счетчик имеет вход установки в нулевое состояние (асинхронный сброс составляющих его триггеров в "0"). Некоторые счетчики имеют цепи установки в произвольное начальное состояние, с которого уже будет начинаться операция счета.

**48. Сумматор. Синтез одноразрядного двоичного сумматора. Многоразрядный сумматор.**

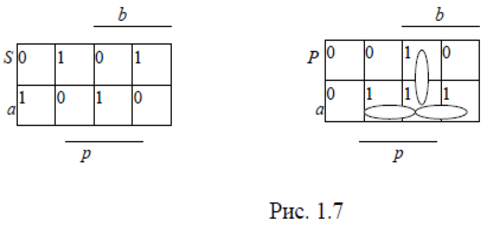
**Одноразрядный двоичный сумматор** обеспечивает сложение одноименных разрядов операндов с учетом переноса, поступающего от ближайшего младшего разряда. Сумматор вырабатывает значение соответствующего **разряда суммы (S) и перенос (P)**, который должен быть учтен в соседнем старшем разряде. Синтез схемы, реализующей функции одноразрядного сумматора, можно выполнить на основании таблицы истинности.



Исходя из реализуемой функции сумматор представляет собой логический узел с двумя выходами (выход суммы S и выход переноса Р) и тремя входами: а - разряд первого операнда; b - разряд второго операнда; р - перенос из младшего разряда. На основании таблицы истинности можно записать логические выражения для формируемых суммы и переноса, которые будут иметь следующий вид:



Полученные функции удобно минимизировать с помощью карты Карно, так как количество переменных невелико. Карты Карно с представленными в них функциями S и P приведены на рис. 1.7.



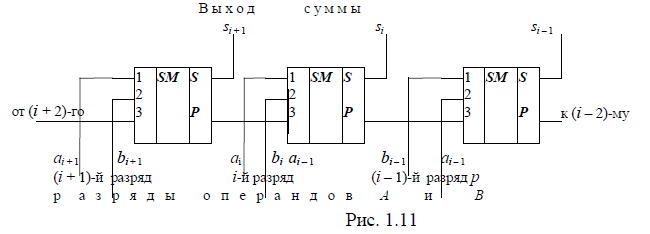
На основании представления функции S на карте можно заключить, что логическое выражение для этой функции не минимизируется.

Минимизированная функция переноса с учетов введенных контуров имеет вид:



Ввиду того что функции P и S формируются в одном и том же узле, при формировании S целесообразно использовать средства, примененные для реализации функции Р. С этой целью рассмотрим функцию Р как переменную для функции S. Тогда модифицированная функция S, зависящая теперь от четырех переменных a, b, p, P, будет записываться в карту Карно как функция четырех переменных.

**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.

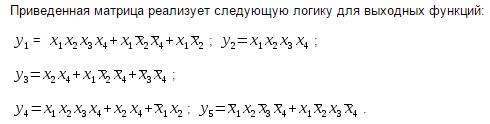
**49.Программируемые логические матрицы.**

Программируемая логическая матрица (ПЛМ) - комбинационный зел, обеспечивающий формирование нескольких функциональных зависимостей на основе заданных переменных.

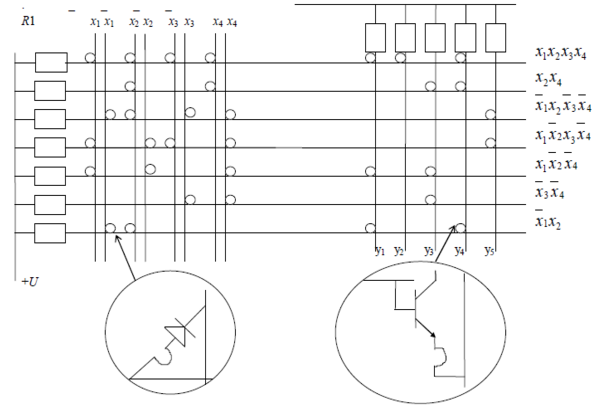
В состав ПЛМ входят дизъюктивная (ДМ) и конъюктивная (КМ) матрицы.

КМ - формирует множество неповторяющихся конъюкций, используемых во всех формируемых логических функциях.

ДМ - для каждой выходной функции формирующие логическое сложение.



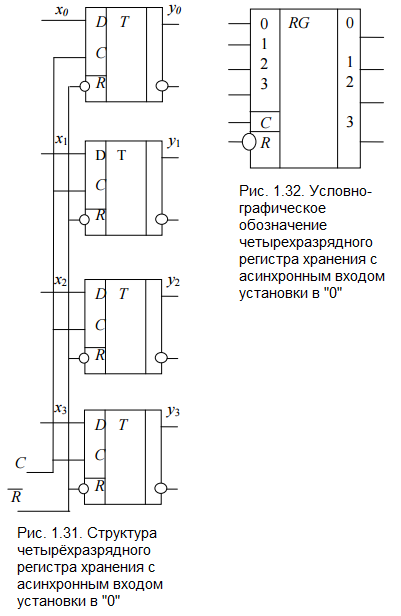
Программирование ПЛМ выполняется следующим образом. При производстве одним из методов интегральной технологии создается заготовка ПЛМ, в которой на пересечениях горизонтальных и вертикальных шин дизъюнктивной матрицы имеет место диодная цепочка, а на пересечениях горизонтальных и вертикальных шин конъюнктивной матрицы располагается транзисторная цепочка. Пользователь в зависимости от логики, которую он собирается реализовать, удаляет перемычки в «ненужных» цепочках. Удаление перемычки осуществляется посредством пропускания по соответствующей горизонтальной и вертикальной шинам мощного тока, который разогревает и испаряет соответствующую легкоплавкую перемычку.



***Вопрос 50. Накапливающие узлы. Регистр хранения***

Для накапливающего узла характерна зависимость выходных сигналов не только от входных, но и от состояния, в котором находился данный узел в момент воздействия входного сигнала. Это означает, что такие узлы могут хранить «историю» входных сигналов, т. е. узлы данного типа обладают памятью. К типовым накапливающим узлам, используемым в вычислительной технике, относятся счетчики и регистры.

Регистры хранения используются для приема, хранения и выдачи многоразрядного кода. Они представляют собой совокупность одноступенчатых триггеров (как правило, D-типа) с общим входом ++8синхронизации. Иногда в регистре имеется также и общий вход асинхронной установки всех триггеров в "0". Схема четырехразрядного регистра хранения приведена на рис. 1.31, а его условно-графическое обозначение – на рис. 1.32

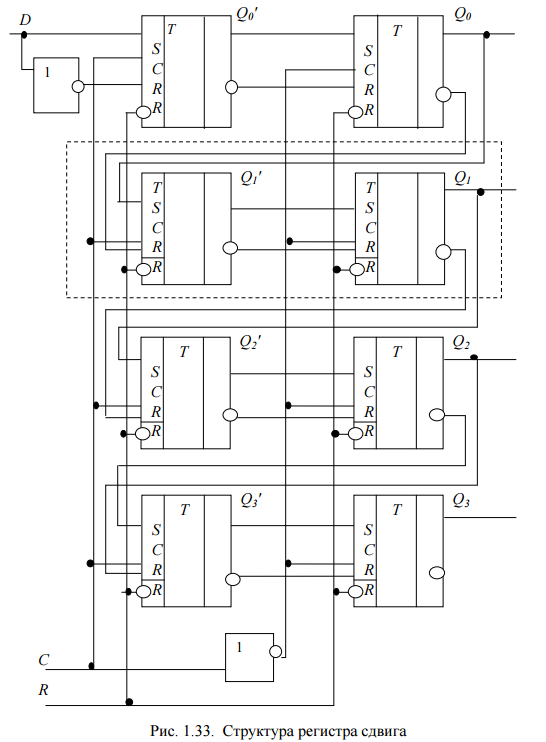


***51. Накапливающие узлы. Регистр сдвига.***

Для накапливающего узла характерна зависимость выходных сигналов не только от входных, но и от состояния, в котором находился данный узел в момент воздействия входного сигнала. Это означает, что такие узлы могут хранить «историю» входных сигналов, т. е. узлы данного типа обладают памятью. К типовым накапливающим узлам, используемым в вычислительной технике, относятся счетчики и регистры.

**Регистр сдвига** – регистр, обеспечивающий помимо хранения информации сдвиг влево или вправо всех разрядов одновременно на одинаковое число позиций. При этом выдвигаемые за пределы регистра разряды теряются, а в освобождающиеся разряды заносится информация, поступающая по отдельному внешнему входу регистра сдвига. Обычно эти регистры обеспечивают сдвиг кода на одну позицию влево или вправо. Но существуют и универсальные регистры сдвига, которые выполняют сдвиг как влево, так и вправо в зависимости от значения сигнала на специальном управляющем входе или при подаче синхросигналов на разные входы регистра. Регистр сдвига может быть спроектирован и таким образом, чтобы выполнять сдвиг одновременно не на одну, а на несколько позиций.

Регистры сдвига строятся на двухступенчатых триггерах. Схема четырехразрядного регистра, выполняющего сдвиг на один разряд от разряда 0 к разряду 3, показана на рис. 1.33, а его условно-графическое обозначение – на рис. 1.34. Ввод информации в данный регистр – последовательный через внешний вход D0. Регистр имеет вход асинхронной установки всех разрядов в "0". Для наглядности каждый двухступенчатый регистр представлен двумя одноступенчатыми с соответствующей организацией синхронизации первой и второй ступеней. Пунктиром обозначен реальный двухступенчатый триггер.





**52. Устройства ЭВМ. Состав АЛУ. Регистр флажков.**

**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. Такие последовательности образуют алгоритм выполнения операций на уровне микроопераций. **Регистр флажков**

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

**53. Синтез АЛУ.**

В современных ЭВМ арифметико- логическое устройство не является самостоятельным схемотехническим блоком. Оно входит в состав микропроцессора, на котором строится компьютер. Однако знание структуры и принципов работы АЛУ весьма важно для понимания работы компьютера в целом.

Для лучшего понимания этих вопросов проведем синтез арифметического устройства, предназначенного для выполнения только одной операции – умножения чисел с фиксированной запятой, заданных в прямом коде, со старших разрядов множителя и сдвигом множимого. В ходе этого процесса также обратим внимание на особенности использования рассмотренных выше основных схемотехнических элементов ЭВМ.

Синтез АЛУ проходит в несколько этапов:

1. Выбор и/или описание метода, по которому предполагается выполнение операции.

2. Составление алгоритма соответствующих действий в соответствии с выбранным (заданным) методом.

3. Определение набора составляющих АЛУ функциональных элементов и узлов, а также разрядность узлов исходя из алгоритма и формата исходных данных.

4. Определение связей между элементами и узлами и отображение их на схеме.

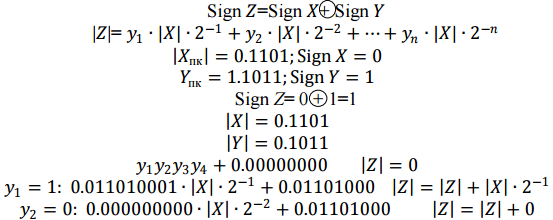
5. Установление порядка функционирования устройства и разработка временной диаграммы управляющих сигналов, которые должны быть поданы на АЛУ от устройства управления.

Пусть операнды имеют вид:

ее.png

где x0, y0 – знаковые разряды.

Операция умножения чисел с фиксированной запятой, заданных в прямом коде, со старших разрядов множителя выполняется по следующей формуле:



дд.png

Каждой переменной, представленной в алгоритме, в схеме должен соответствовать элемент хранения. Разрядность модуля произведения равна сумме разрядностей сомножителей. Умножение двоичного числа на2– i обеспечивается сдвигом этого числа вправо на соответствующее количество разрядов. Переход к анализу очередного разряда множителя (i = i + 1) может быть обеспечен сдвигом регистра множителя на один разряд в сторону старших разрядов. Исходя из этого, определим состав оборудования, необходимого для реализации АЛУ заданного типа для n = 4

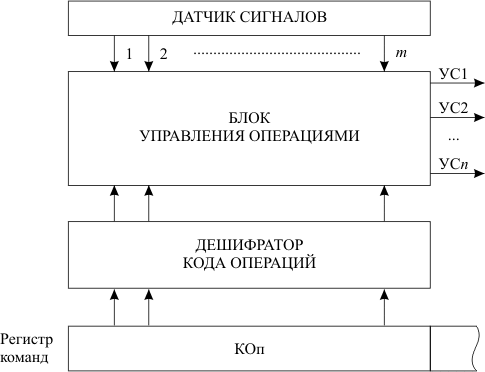
**54.Схемное устройство управления**

Устройство управления схемного типа (рис. 4.1) состоит из:

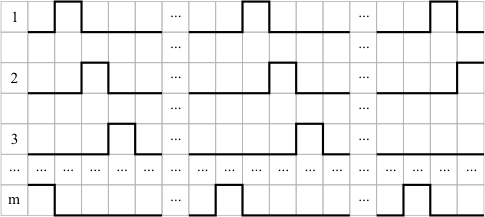
· ***датчика сигналов***, вырабатывающего последовательность импульсов, равномерно распределенную во времени по своим шинам ( рис. 4.2) ( n - общее количество управляющих сигналов, необходимых для выполнения любой операции; m - количество тактов, за которое выполняется самая длинная операция);

· блока управления операциями, осуществляющего выработку управляющих сигналов, то есть коммутацию сигналов, поступающих с ДС, в соответствующем такте на нужную управляющую шину;

· дешифратора кода операций, который дешифрирует код операции команды, присутствующей в данный момент в *регистре команд*, и возбуждает одну шину, соответствующую данной операции; этот сигнал используется блоком управления операциями для выработки нужной последовательности управляющих сигналов



**Рис 4.1** Функциональная схема схемного устройства управления.

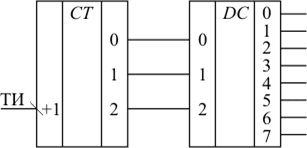


**Рис. 4.2.** Временная диаграмма работы датчика сигналов

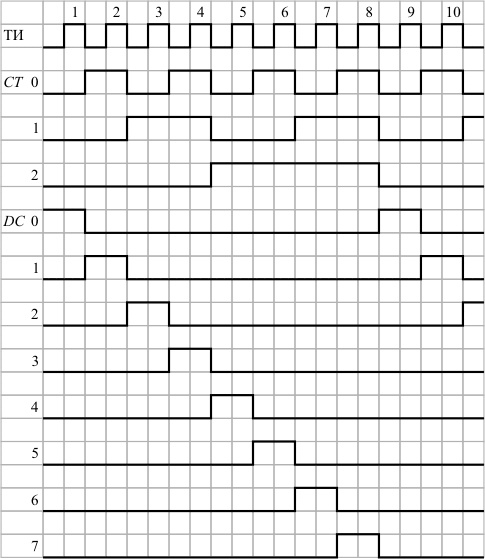
*Датчик сигналов* обычно реализуется на основе счетчика с *дешифратором* или на *сдвиговом регистре*.

#### **Датчик сигналов на основе счетчика с дешифратором**

Реализация *датчика сигналов* на основе счетчика с *дешифратором* представлена на рис. 4.3. По заднему фронту каждого тактового импульса, поступающего на *устройство управления* с системного *генератора импульсов*, счетчик увеличивает свое состояние; выходы счетчика соединены со входами дешифратора, выходы которого и являются выходами *датчика сигналов* (рис. 4.4).

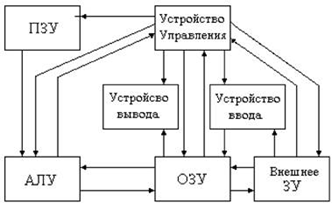


**Рис. 4.3.** Схема датчика сигналов на основе счетчика с дешифратором



**Рис. 4.4.** Временная диаграмма работы датчика сигналов на основе счетчика с дешифратором

**55.Микропрограммное устройство управления.(МУУ)**

В микропроцессорах используют два метода выработки совокупности функциональных управляющих сигналов: программный и микропрограммный.

Микропрограммное управление. Классическое вычислительное устройство состоит из арифметического устройства (АУ), устрой­ства управления (УУ), запоминающего устройства (ЗУ) и уст­ройства ввода-вывода (УВВ); АУ и УУ образуют процессор лю­бой ЭВМ, т. е. ее управляющую и обрабатывающую части .

В соответствии с микропрограммным принципом управления любая сложная операция делится на последовательность более простых действий. Такое простое действие называется микроопе­рацией и выполняется за один такт работы АУ. Для задания оче­редности следования микроопераций вводятся специальные пе­ременные, называемые логическими условиями. Совокупность микроопераций, выполняемых за один цикл (несколько тактов) рабо­ты устройства, называется микрокомандой (МК). Микрокоманда представляет собой двоичное n-разрядное слово, содержащее код операции (КОП), выполняемой АУ, а также коды адресов исход­ных данных и результата. Микрокоманда поступает на вход АУ, которое дешифрует ее и вырабатывает управляющие сигналы. Эти сигналы стробируются импульсами внутреннего блока синхрони­зации, который формирует временные такты выполнения микро­операций. Микрооперации жестко связаны со структурой АУ и не могут быть изменены.

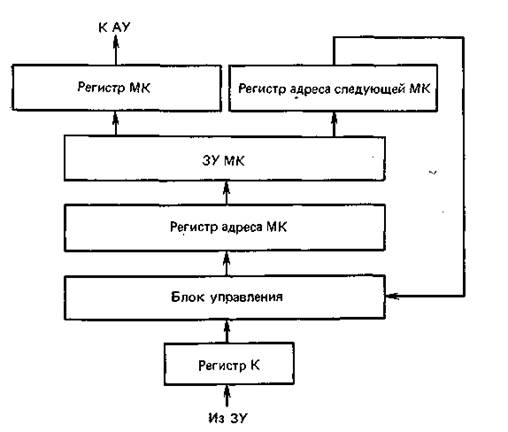
После выполнения действия, определенного КОП(код операции), АУ инфор­мирует об окончании выполнения микрокоманды. Каждому АУ присущ только свой, конкретный набор МК, который называется системой микрокоманд.

Устройство, предназначенное для записи, хранения и считы­вания МК(микрокоманд) называется микропрограммным устройством управления (МУУ). В простейшем случае МУУ представляет собой большую интегральную схему постоянного запоминающего устройства(ПЗУ) или ППЗУ(перепрограммируемое ПЗУ), в которой записаны МК. Для считывания этих МК необходимо устройство формирования адреса, например счет­чик. Любую операцию можно представить последовательностью МК (микропрограммой). Необходимо отметить два основных от­личия микропрограммного управления от жесткого: смена выпол­няемой операции обеспечивается заменой микропрограммы; при считывании каждой МК требуется обращение к ПЗУ, что снижает быстродействие УУ. Итак, микропрограммное управление заменя­ет аппаратные средства программными и обеспечивает высокую гибкость, но при снижении быстродействия.

Микрокомандный уровень управления АУ является самым низ­ким уровнем, доступным разработчику МПУ. МК представляет собой достаточно мелкую детализацию выпол­няемой операции, например «обнулить регистр», «содержимое ре­гистра переслать в аккумулятор» и т. д. Поэтому для программи­рования сложных алгоритмов, которыми являются алгоритмы об­работки сигналов, потребовалось бы составить микропрограммы,, содержащие сотни, тысячи микрокоманд. Отладить такую микро­программу очень сложно.

Для повышения уровня детализации выполняемой операции вводится командный уровень управления. Символом этого уровня является команда, которая представляет собой (как и микроко­манда) m-разрядное двоичное слово (обычно m<n). В общем случае формат команды должен включать КОП, адреса операн­дов, над которыми выполняется операция, и адрес результата. Ко­манда представляет собой последовательность МК (микропрог­рамму). С целью экономии емкости микропрограммной памяти запись МК осуществляется в ячейки памяти, содержание которых, учитывает код текущей МК.

Структурная схема МУУ изображена на рисунке. Команда, считанная из ЗУ, поступает на регистр команд и далее на блок управления. В соответствии с принятыми сигналами блок управле­ния формирует адрес первой МК микропрограммы, соответствую­щей принятой команде. Этот адрес через регистр поступает в ЗУ МК. Считанная из ЗУ МК состоит из двух частей: операционной (или собственно микрокоманды, которая поступает на АУ) и ад­ресной, которая поступает на блок управления. Приняв адресную часть МК, блок управления формирует адрес следующей МК. Вновь считанная МК имеет свою адресную часть, которая посту­пает на блок управления. Этот процесс продолжается до тех пор, пока не будет считана последняя МК данной программы. После этого МУУ готово к приему следующей команды. Длина микро­программы определяется разрядностью кода адреса следующей МК.

В целом же, принцип микропрограммного управления (ПМУ) включает следующие позиции: 

1) любая операция, реализуемая устройством, является последовательностью элементарных действий - микроопераций;

2) для управления порядком следования микроопераций используются логические условия;

3) процесс выполнения операций в устройстве описывается в форме алгоритма, представляемого в терминах микроопераций и логических условий, называемого микропрограммой;

4) микропрограмма используется как форма представления функции устройства, на основе которой определяются структура и порядок функционирования устройства во времени.

**56. Запоминающие устройства. Характеристики, классификация ЗУ**

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.

Термин “запоминающее устройство” обычно используется, когда речь идет о принципе построения некоторого устройства памяти (например, полупроводниковое ЗУ, ЗУ на жестком магнитном диске и т.п.), а термин “память” – когда хотят подчеркнуть выполняемую устройством памяти логическую функцию или место расположения в составе оборудования ЭВМ (например, оперативная память – ОП, внешняя память и т.п.). В тех вопросах, где эти отличия не имеют принципиального значения, термины “память” и “запоминающее устройство” мы будем использовать как синонимы.

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. В настоящее время практически все запоминающие устройства в качестве минимально адресуемого элемента используют 1 байт (1 байт = 8 двоичных разрядов (бит)). Поэтому емкость памяти обычно определяется в байтах, килобайтах (1Кбайт=210 байт), мегабайтах (1Мбайт = 220 байт), гигабайтах (1Гбайт = 230 байт) и т.д. За одно обращение к запоминающему устройству производится считывание или запись некоторой единицы данных, называемой словом, различной для устройств разного типа. Это определяет разную организацию памяти. Например, память объемом 1 мегабайт может быть организована как 1М слов по 1 байту, или 512К слов по 2 байта каждое, или 256К слов по 4 байта и т.д.

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:

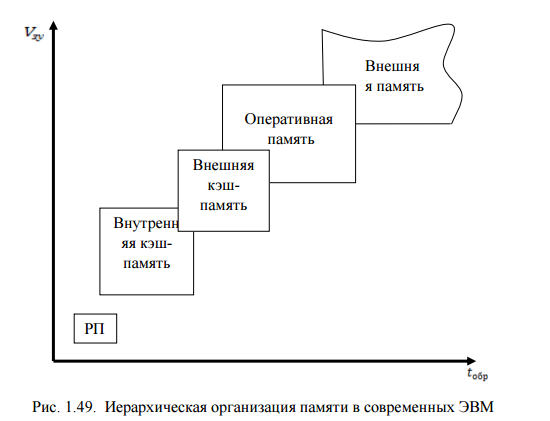


где быстродействие ЗУ при считывании информации;

 быстродействие ЗУ при записи.

## **Классификация запоминающих устройств.**

Как отмечалось выше, основные характеристики запоминающих устройств – это емкость и быстродействие. Идеальное запоминающее устройство должно обладать бесконечно большой емкостью и иметь бесконечно малое время обращения. На практике эти параметры находятся в противоречии друг другу: в рамках одного типа ЗУ улучшение одного из них ведет к ухудшению значения другого. К тому же следует иметь в виду и экономическую целесообразность построения запоминающего устройства с теми или иными характеристиками при данном уровне развития технологии. Поэтому в настоящее время запоминающие устройства компьютера, как это и предполагал Нейман, строятся по иерархическому принципу



Иерархическая структура памяти позволяет экономически эффективно сочетать хранение больших объемов информации с быстрым доступом к информации в процессе ее обработки.

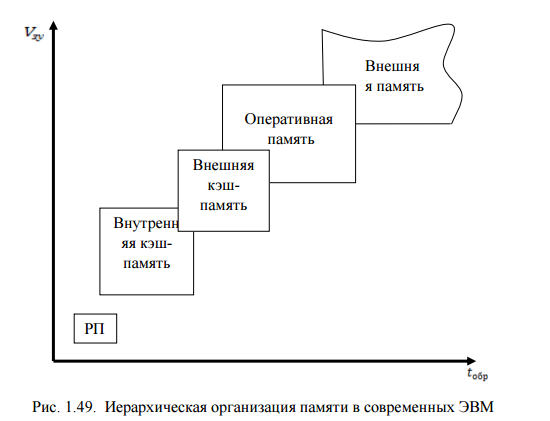
На нижнем уровне иерархии находится **регистровая память** – набор регистров, входящих непосредственно в состав микропроцессора (центрального процессора – CPU). Регистры CPU программно доступны и хранят информацию, наиболее часто используемую при выполнении программы: промежуточные результаты, составные части адресов, счетчики циклов и т.д. Регистровая память имеет относительно небольшой объем (до нескольких десятков машинных слов). РП работает на частоте процессора, поэтому время доступа к ней минимально. Например, при частоте работы процессора 2 ГГц время обращения к его регистрам составит всего 0,5 нс.

**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.

Для заполнения пробела между РП и ОП по объему и времени обращения в настоящее время используется **кэш-память**, которая организована как более быстродействующая (и, следовательно, более дорогая) статическая оперативная память со специальным механизмом записи и считывания информации и предназначена для хранения информации, наиболее часто используемой при работе программы. Как правило, часть кэш-памяти располагается непосредственно на кристалле микропроцессора (внутренний кэш), а часть – вне его (внешняя кэш-память). Кэш-память программно недоступна. Для обращения к ней используются аппаратные средства процессора и компьютера.

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

**57. Иерархия памяти в современных ЭВМ.**



Иерархическая структура памяти позволяет экономически эффективно сочетать хранение больших объемов информации с быстрым доступом к информации в процессе ее обработки.

На нижнем уровне иерархии находится **регистровая память** – набор регистров, входящих непосредственно в состав микропроцессора (центрального процессора – CPU). Регистры CPU программно доступны и хранят информацию, наиболее часто используемую при выполнении программы: промежуточные результаты, составные части адресов, счетчики циклов и т.д. Регистровая память имеет относительно небольшой объем (до нескольких десятков машинных слов). РП работает на частоте процессора, поэтому время доступа к ней минимально. Например, при частоте работы процессора 2 ГГц время обращения к его регистрам составит всего 0,5 нс.

**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.

Для заполнения пробела между РП и ОП по объему и времени обращения в настоящее время используется **кэш-память**, которая организована как более быстродействующая (и, следовательно, более дорогая) статическая оперативная память со специальным механизмом записи и считывания информации и предназначена для хранения информации, наиболее часто используемой при работе программы. Как правило, часть кэш-памяти располагается непосредственно на кристалле микропроцессора (внутренний кэш), а часть – вне его (внешняя кэш-память). Кэш-память программно недоступна. Для обращения к ней используются аппаратные средства процессора и компьютера.

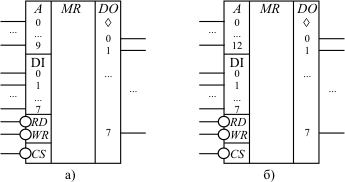
**Внешняя память** организуется, как правило, на магнитных и оптических дисках, магнитных лентах. Емкость дисковой памяти достигает тысяч гигабайт при времени обращения менее 1 мкс. Магнитные ленты вследствие своего малого быстродействия и большой емкости используются в настоящее время в основном только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

***58. Построение ЗУ с заданной организацией. ЗУ 8К×8 на БИС 1К×8.***

В современных ЭВМ минимальной адресуемой единицей памяти является, как правило, 1 байт. В связи с этим обмен с памятью организуется блоками, кратными этой величине: байтами, словами, двойными словами, учетверенными словами, в зависимости от выполняемой процессором команды и разрядности внешней шины данных. Такой обмен происходит под управлением специальных сигналов, поступающих по системной шине. Преобразование информации из формата ее представления на шине данных в формат, учитывающий организацию конкретных схем памяти, осуществляется специальными интерфейсными схемами. Большие интегральные схемы (БИС), на которых строятся модули памяти, являются изделиями электронной промышленности и могут иметь различную организацию. Разработчики средств вычислительной техники должны учитывать имеющуюся у них номенклатуру БИС памяти, чтобы построить запоминающее устройство необходимой емкости и организации. Для этой цели может проводиться объединение нескольких БИС либо с целью увеличения количества слов в модуле памяти, либо для наращивания разрядности каждого слова, либо с той и другой целью одновременно.

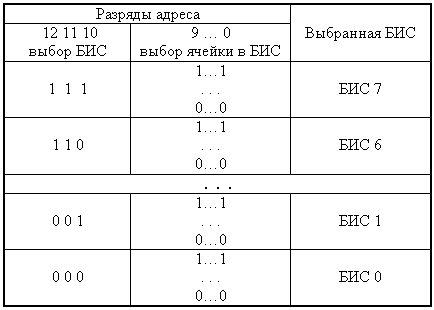
Рассмотрим варианты построения блока памяти необходимой организации при наличии заданных БИС памяти.

Построить ОЗУ с организацией 8К\*8 разрядов на БИС с организацией 1К\*8 разрядов (рис.5.3).



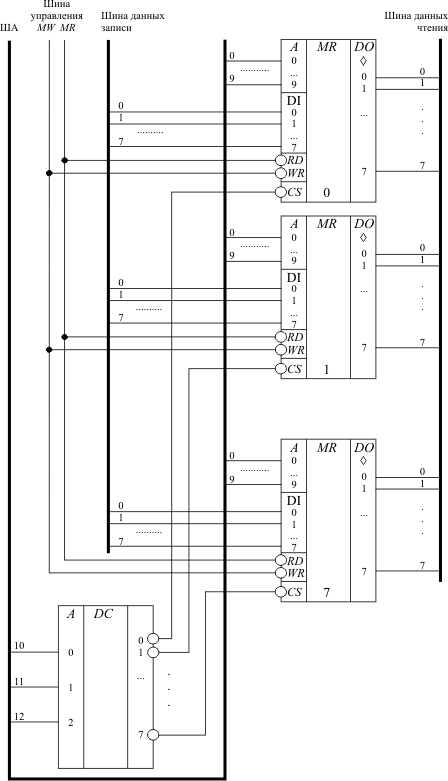
**Рис. 5.3.** Условно-графические обозначения запоминающих устройств с различной организацией:а - 1К\*8 разрядов; б - 8К\*8 разрядов

В данном случае требуется построить модуль памяти, имеющий большее число слов, чем в составляющих его БИС. Модуль памяти будет состоять из восьми БИС. Для обращения к модулю памяти используется 13-разрядный адрес ( А12 А0 ), поступающий по шине адреса (ША). Три старших разряда ( А12-А10 ) определяют ту схему, которая в данный момент включается в работу, а каждая ячейка внутри любой БИС определяется 10-ю младшими разрядами адреса ( А9-А0 ) (рис. 5.4).



**Рис. 5.4.** Организация модуля памяти

При единичном значении сигнала на входе выбора кристалла БИС ( CS=1 ) выходные разряды данных находятся в третьем состоянии, то есть как бы отключены от шины ( DO=Z ).Таким образом, при любом значении кода на шине адреса всегда в работе находится одна и только одна из восьми БИС (рис. 5.5).

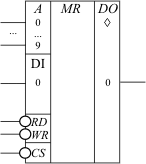


**Рис. 5.5.** Запоминающее устройство объемом 8К\*8 разрядов на БИС с организацией 1К\*8 разрядов

В реальных микросхемах шины данных записи и чтения ( DI и DO ) обычно представляют собой общую двунаправленную шину.

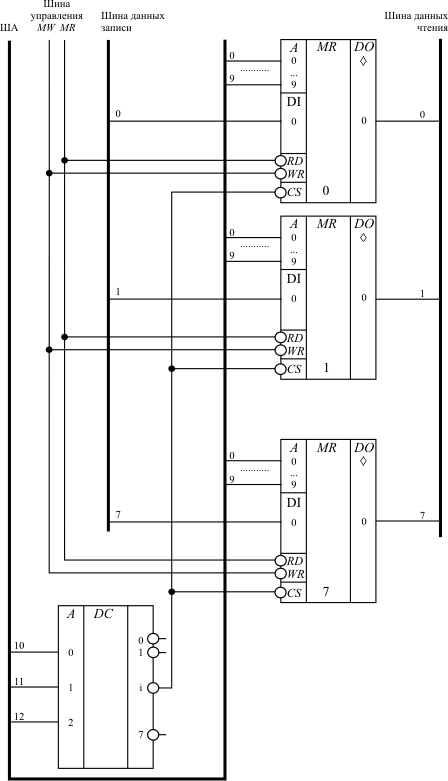
Сигналы на шине управления означают: MW - сигнал записи в память, MR - сигнал чтения из памяти.

Построить ОЗУ с организацией 1К\*8 разрядов на БИС с организацией 1К\*1 разряд (рис.5.6).



**Рис. 5.6.** Условно-графическое обозначение БИС с организацией 1К\*1 разряд

В данном случае требуется увеличить разрядность слова памяти. Так как все разряды одного слова должны записываться и считываться одновременно, то все БИС должны работать параллельно. Модуль памяти будет состоять из восьми БИС (рис. 5.7). Если разрабатываемый блок является частью модуля памяти, имеющего объем больше, чем 1К слов (например, 8К), то необходим специальный дешифратор, который будет дешифрировать старшие разряды адреса аналогично тому, как показано на рис. 5.5 и включать в работу данный блок.



**Рис. 5.7.** Запоминающее устройство объемом 1К\*8 разрядов на БИС с организацией 1К\*1 разряд

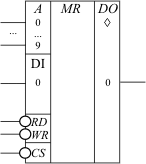
**59. Построение ЗУ с заданной организацией. ЗУ 8К×8 на БИС 1К×1.**

В современных ЭВМ минимальной адресуемой единицей памяти является, как правило, 1 байт. В связи с этим обмен с памятью организуется блоками, кратными этой величине: байтами, словами, двойными словами, учетверенными словами, в зависимости от выполняемой процессором команды и разрядности внешней шины данных. Такой обмен проходит под управлением специальных сигналов, поступающих по системной шине. Преобразование информации из формата ее представления на шине данных в формат, учитывающий организацию конкретных схем памяти, осуществляется специальными интерфейсными схемами. Большие интегральные схемы (БИС), на которых строятся модули памяти, являются изделиями электронной промышленности и могут иметь различную организацию. Разработчики средств вычислительной техники должны учитывать имеющуюся у них номенклатуру БИС памяти, чтобы построить запоминающее устройство необходимой емкости и организации. Для этой цели может проводиться объединение нескольких БИС либо с целью увеличения количества слов в модуле памяти, либо для наращивания разрядности каждого слова, либо с той и другой целью одновременно. Рассмотрим варианты построения блока памяти необходимой организации при наличии заданных БИС памяти

В реальных микросхемах шины данных записи и чтения (DI и DO) обычно представляют собой общую двунаправленную шину.

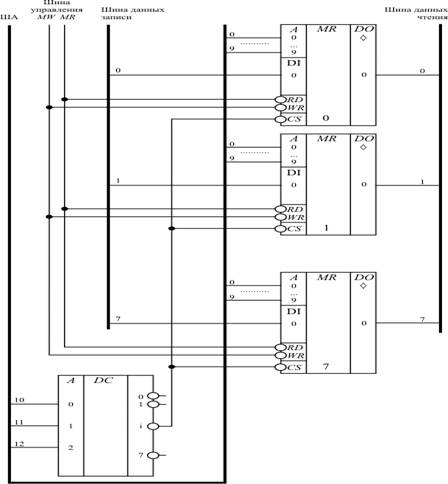
Сигналы на шине управления означают: MW - сигнал записи в память, MR - сигнал чтения из памяти.

Построить ОЗУ с организацией 1К\*8 разрядов на БИС с организацией 1К\*1 разряд (рис.11).



**Рис. 11.** Условно-графическое обозначение БИС с организацией 1К\*1 разряд

В данном случае требуется увеличить разрядность слова памяти. Так как все разряды одного слова должны записываться и считываться одновременно, то все БИС должны работать параллельно. Модуль памяти будет состоять из восьми БИС (рис. 11.1). Если разрабатываемый блок является частью модуля памяти, имеющего объем больше, чем 1К слов (например, 8К), то необходим специальный дешифратор, который будет дешифрировать старшие разряды адреса аналогично тому, как показано на рис. 10.2 и включать в работу данный блок.



**Рис. 11.1.** Запоминающее устройство объемом 1К\*8 разрядов на БИС с организацией 1К\*1 разряд