Позиционные системы счисления

Информация в компьютерах кодируется, как правило, в двоичной системе счисления. Система счисления – это способ представления любого числа с помощью символов, имеющих определенные количественные значения и называемых цифрами. Различают позиционные и непозиционные системы счисления. В непозиционной системе счисления цифры не меняют своего количественного значения при изменении их расположения в числе. Примером такой системы является римская система счисления.

В позиционной системе счисления количественное значение каждой цифры зависит от её места (позиции) в числе. Количество S различных цифр, употребляемых в позиционной системе, называется ее основанием. Эти цифры обозначают s целых чисел, обычно 0, 1, ..., (S-1). В десятичной системе используются десять цифр: 0, 1,2, 3,4,5,6,7,8,9; эта система имеет основанием число десять.

В общем случае в позиционной системе с основанием s любое число x может быть представлено в виде полинома от основания S:

$X_{(s)}= \alpha_n S^n + \alpha_{n-1} S^{n-1} + \dots + \alpha_1 S^1 + \alpha_0 S^0 + \dots + \alpha_{-m} S^{-m} = \sum_{k=-m}^{n} \alpha k S k$ где в качестве коэффициентов A_k могут стоять любые из S цифр, используемых в системе счисления.

Принято представлять числа в виде соответствующей последовательности цифр:

$x= \alpha_n \alpha_{n-1} \dots \alpha_1 \alpha_0 , \alpha_{-1} \dots .$

В этой последовательности запятая (точка) отделяет целую часть числа от дробной (коэффициенты при положительных степенях, включая нуль, от коэффициентов при отрицательных степенях). Запятая опускается, если нет отрицательных степеней. Позиции цифр, отсчитываемые от запятой, называют разрядами. В позиционной системе счисления значение каждого разряда больше значения соседнего справа разряда в число раз, равное основанию S системы.

В компьютерах применяют позиционные системы счисления с недесятичным основанием: двоичную, шестнадцатиричную и восьмеричную. В дальнейшем для обозначения используемой системы счисления число будет заключено в скобки и в индексе указано основание системы счисления.

Наибольшее распространение в ЭВМ имеет *двоичная система счисления.* В этой системе используются только две («двоичные») цифры: 0 и 1.

В двоичной системе любое число может быть представлено последовательностью двоичных цифр

$x = \alpha_m \alpha_{m-1} \dots \alpha_1 \alpha_0, \alpha_{-1} \alpha_{-2} \dots,$ где α_i, либо 0, либо 1.

Эта запись соответствует сумме степеней числа 2, взятых с указанными в ней коэффициентами:

$x = \alpha_m 2^m + \alpha_{m-1} 2^{m-1} + \dots + \alpha_1 2^1 + \alpha_0 2^0 + \alpha_{-1} 2^{-1} + \alpha_{-2} 2^{-2} + \dots$ Например,

двоичное число

$(10101101, 101)_2 = 1\cdot 2^7 + 0\cdot 2^6 + 1\cdot 2^5 + 0\cdot 2^4 + 1\cdot 2^3 + 1\cdot 2^2 + 0\cdot 2^1 + 1\cdot 2^0 + 1\cdot 2^{-1} + 0\cdot 2^{-2} + 1\cdot 2^{-3}$

как следует из приведенного разложения его по степеням числа 2, соответствует десятичному числу

$(173, 625)_{10}$.

Двоичное изображение числа требует большего (для многоразрядного числа примерно в 3,3 раза) числа разрядов, чем его десятичное представление. Тем не менее применение двоичной системы создает большие удобства для проектирования компьютеров, так как для представления в машине разряда двоичного числа может быть использован любой простой элемент, имеющий всего два устойчивых состояния. Другим важным достоинством двоичной системы является простота двоичной арифметики. Соответствие цифр отмеченных систем счисления можно получить из следующей таблицы.

Десятичная	Двоичная	Восьмеричная	Шестнадцатеричная
0	000	0	0
1	001	1	1
2	010	2	2
3	011	3	3
4	100	4	4
5	101	5	5
6	110	6	6
7	111	7	7
8	1000	10	8
9	1001	11	9
10	1010	12	A
11	1011	13	B
12	1100	14	C
13	1101	15	D
14	1110	16	E
15	1111	17	F
16	10000	20	10

В *восьмеричной системе,* употребляется восемь цифр: 0, 1, 2, 3, 4, 5, 6, 7. Любое число в восьмеричной системе представляется последовательностью цифр

$x = \alpha_q\, \alpha_{q-1} \ldots \alpha_1\alpha_0, \alpha_{-1}\, \alpha_{-2} \ldots,$

в которой α_i могут принимать значения от 0 до 7.

Например, восьмеричное число

$(703, 04)_8 = 7\cdot 8^2+0\cdot 8^1+3\cdot 8^0+0\cdot 8^{-1}+4\cdot 8^{-2} = (451, 0625)_{10}$.

В шестнадцатиричной системе для изображения чисел употребляется 16 цифр: от 0 до 15. При этом, чтобы одну цифру не изображать двумя знаками, введены специальные обозначения для цифр, больших девяти. Первые десять цифр этой системы обозначаютсяцифрами от 0 до 9, а старшие пять цифр — латинскими буквами: десять — *A,* одиннадцать — *B,* двенадцать — C, тринадцать — *D,* четырнадцать — *E,* пятнадцать — *F.* Например, шестнадцатиричное число

$(B2E, 4)_{16} = 11\cdot 16^2+2\cdot 16^1+14\cdot 16^0+4\cdot 16^{-1} = (2862, 25)_{10}$.

Для перевода восьмеричного (шестнадцатиричного) числа в двоичную форму достаточно заменить каждую цифру этого числа соответствующим трехразрядным (четырехразрядным) двоичным числом, при этом отбрасывают ненужные нули, например

$$\left(\underbrace{3}_{011}\ \underbrace{0}_{000}\ \underbrace{5}_{101}\ \underbrace{4}_{100}\right)_8 = (11000101, 1)_2.$$

$$\left(\underbrace{7}_{0111}\ \underbrace{B}_{1011}\ \underbrace{2}_{0010}\ \underbrace{E}_{1110}\right)_{16} = (11110110010, 111)_2.$$

Для перехода от двоичной к восьмеричной (или шестнадцатиричной) системе поступают следующим образом: двигаясь от запятой влево и вправо, разбивают двоичное число на группы по три (четыре) разряда, дополняя при необходимости нулями крайние левую и правую группы. Затем каждую группу из трех (четырех) разрядов заменяют соответствующей восьмеричной (шестнадцатиричной) цифрой.

Приведем примеры:

а) перевод двоичного числа 1101111001, 1101 в восьмеричное:

$$\underbrace{001}_{1}\ \underbrace{101}_{5}\ \underbrace{111}_{7}\ \underbrace{001}_{1}\ \underbrace{110}_{6}\ \underbrace{100}_{4} = (1571, 64)_8;$$

б) перевод двоичного числа 11111111011, 100111 в шестнадцатиричное:

$$\underbrace{0111}_{7}\ \underbrace{1111}_{F}\ \underbrace{1011}_{B}\ \underbrace{1001}_{9}\ \underbrace{1100}_{C} = (7FB, 9C)_{16}$$

В настоящее время в большинстве ЭВМ используются двоичная система и двоичный алфавит для представления и хранения чисел, команд и другой информации, а также при выполнении арифметических и логических операций.

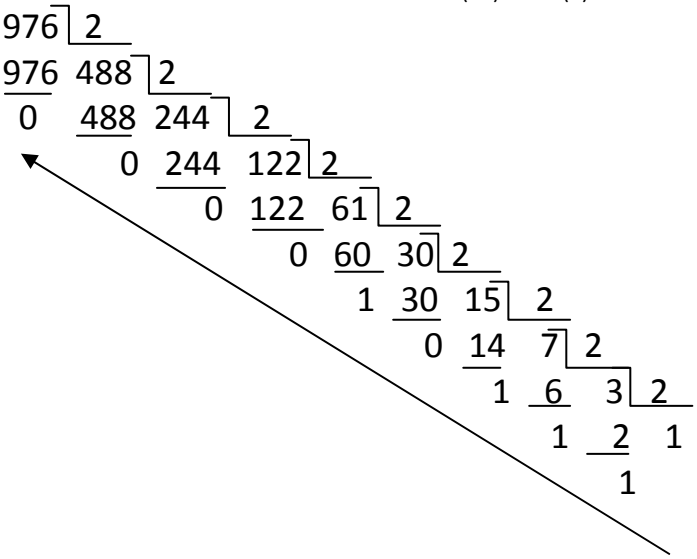
Шестандатиричная (и восьмеричная) система применяется в текстах программ для более короткой и удобной записи двоичных кодов команд, адресов и операндов. Кроме того, эти системы применяются в ЭВМ при некоторых формах представления чисел .

Перевод целых чисел

Для перевода целого числа из S-й системы счисления в систему

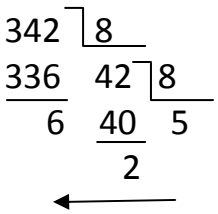
счисления с основанием *q* надо переводимое число последовательно делить на основание *q-й* системы счисления, в которую это число переводится, до тех пор, пока не будет получено частное, меньше основания *q*. Число в новой системе счисления запишется в виде остатков от деления, начиная с последнего частного, представляющего собой старшую цифру числа.

Пример 1. Переведем число 976 из десятичной системы счисления в двоичную систему счисления ($976_{(10)} \rightarrow X_{(2)}$)



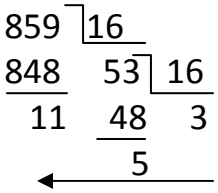
$976_{(10)}=1111010000_{(2)}$

Пример 2. Переведем число 342 из десятичной системы счисления в восьмеричную систему счисления (342(10) $\rightarrow X_{(8)}$):



$342_{(10)} \rightarrow 526_{(8)}$

Пример 3. Переведём число $859_{(10)} \rightarrow X_{(16)}$

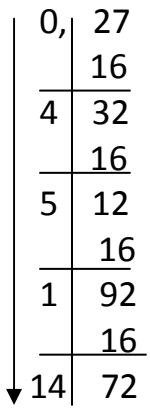


$859_{(10)} \rightarrow 35B_{(16)}$

Перевод правильных дробей

Для перевода правильных дробей в систему счисления с основание *q* умножают исходную дробь (а дальше только дробные части произведения, выделяя целые части) последовательно на основание системы счисления *q.* Полученные в результате умножения целые части произведения являются соответствующими разрядами дробного числа в системе счисления с основанием *q.*

Пример 2.4. Переведем число $0,27_{(10)} \rightarrow X_{(16)}$:

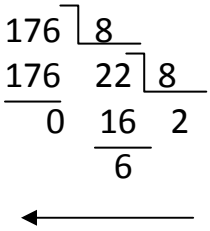


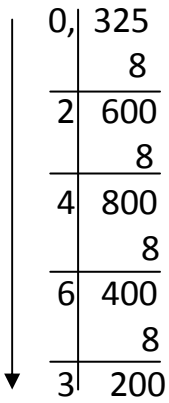
$0,27_{(10)} \rightarrow 451E_{(16)}$

Перевод неправильных дробей

Перевод неправильных дробей в систему счисления с основанием *q* выполняется отдельно для целой и дробной частей числа по вышеизложенным правилам с последующим соединением этих частей в одну неправильную дробь, представленную уже в новой системе счисления.

Пример 2.6. Переведем число $176,325(10) \rightarrow X_{(8)}$





$176.325_{(10)} \rightarrow 260.2463_{(8)}$

ДВОИЧНАЯ АРИФМЕТИКА

Правила выполнения арифметических действий над двоичными числами задаются таблицами двоичных сложения, вычитания и умножения:

Двоичные операции

сложение	вычитание	умножение
0+0=0	0-0=0	0x0=0
0+1=1	1-0=1	0x1=0
1+0=1	1-1=0	1x0=0
1+1=0+ед.переноса	10-1=1	1x1=1

Правила арифметики во всех позиционных системах аналогичны. При сложении в каждом разряде в соответствии с таблицей двоичного сложения производится сложение двух цифр слагаемых или двух этих цифр и 1, если имеется перенос из соседнего младшего разряда. В результате получается цифра соответствующего разряда суммы и, возможно, также 1 переноса в старший разряд. Приведем пример сложения двух двоичных чисел:

Переносы

111

110111.01 55.25

+ 10011.10 + 19.5

1001010.11 74.75

Справа показано сложение тех же чисел, представленных в десятичной системе.

При вычитании двоичных чисел в данном разряде при необходимости занимается 1 из следующего старшего разряда. Эта занимаемая 1 равна двум 1 данного разряда. Заем производится каждый раз, когда цифра в разряде вычитаемого больше цифры в том же разряде уменьшаемого. Поясним сказанное примером:

11011,10

1101,01

1110,01

Умножение двоичных многоразрядных чисел производится путем образования частичных произведений и последующего их суммирования. В соответствии с таблицей двоичного умножения каждое частичное произведение равно 0, если в соответствующем разряде множителя стоит 0, или равно множимому, сдвинутому на соответствующее число разрядов влево, если в разряде множителя стоит 1. Таким образом, операция умножения многоразрядных двоичных чисел сводится к операциям сдвига и сложения. Положение запятой определяется так же, как при умножении десятичных чисел. Сказанное поясняется примером:

1011,1 x 101,01 = 111100,011

10111

```
                    x 10101
      10111
     00000
  +  10111
    00000
   10111
   111100011
```

Особенности выполнения деления двоичных чисел поясняются следующим примером:

```
1100,011:10,01 = ?  1100011 |10010
                   -10010    101,1
                     11011
                    -10010
                     10010
                    -10010
                     00000
```

Благодаря простоте правил двоичного сложения, вычитания и умножения применение в ЭВМ двоичной системы счисления позволяет упростить схемы устройств, выполняющих арифметические операции.

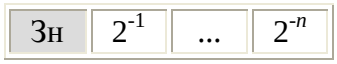
ФОРМЫ ПРЕДСТАВЛЕНИЯ ЧИСЕЛ В КОМПЬЮТЕРЕ

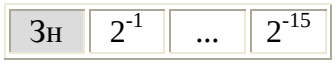
Любая информация (числа, команды, алфавитно-цифровые записи и т. п.) представляется в компьютере в виде двоичных кодов (двоичных слов) фиксированной или переменной длины. Отдельные элементы двоичного кода, имеющие значение 0 или 1, называют разрядами или битами. В компьютере слова часто разбивают на части, называемые *байтами.* В современных ЭВМ широко используется байт, содержащий 8 бит.

Двоичный разряд представляется в компьютере некоторым техническим устройством, например триггером, двум различным состояниям которого приписывают значения 0 и 1. Набор соответствующего количества таких устройств служит для представления многоразрядного двоичного числа (слова).

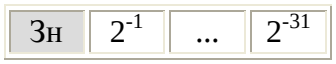
В компьютере применяют две формы представления чисел: с фиксированной запятой (точкой) и с плавающей запятой (точкой). Эти формы называют также соответственно естественной и полулогарифмической.

При представлении чисел с фиксированной запятой положение запятой фиксируется в определенном месте относительно разрядов числа. Обычно подразумевается, что запятая находится или перед старшим разрядом, или после младшего. В первом случае могут быть представлены только числа, которые по модулю меньше 1, во втором — только целые числа.





< 2 байта, 16 разрядов >



< 4 байта, 32 разряда >

Форматы данных для представления двоичных чисел с фиксированной запятой (точкой)

На рис. 1.1 показаны примеры форматов данных для представления двоичных чисел с фиксированной запятой и соответствующие разрядные сетки. По сложившейся в вычислительной технике традиции нумерация разрядов (бит) в разрядной сетке в машинах общего назначения (ЕС ЭВМ) ведется слева направо, а в малых ЭВМ, микро-ЭВМ и микропроцессорах — справа налево. На разрядной сетке указаны веса разрядов.

При представлении числа со знаком для кода знака выделяется «знаковый» разряд (обычно крайний слева). В этом разряде 0 соответствует плюсу, а 1 — минусу.

На рис. 1.1, а показан формат для чисел с запятой, фиксированной перед старшим разрядом. В этом формате могут быть с точностью до $2^{-(n-1)}$ представлены числа (правильные дроби) в диапазоне

$2^{-(n-1)} \le |x| \le 1 - 2^{-(n-1)}$

Первые компьютеры были машинами с фиксированной запятой, причем запятая фиксировалась перед старшим разрядом числа. В настоящее время, как правило, форму с фиксированной запятой применяют для представления целых чисел (запятая фиксирована после младшего разряда).

Используют два варианта представления целых чисел: со знаком и без знака. В последнем случае все разряды разрядной сетки служат для представления модуля числа.

Представление чисел с фиксированной запятой используется как основное и единственное лишь в сравнительно небольших по своим вычислительным возможностям машинах, применяемых в системах передачи данных, для управления технологическими процессами и обработки измерительной информации в реальном масштабе времени.

В машинах, предназначенных для решения широкого круга вычислительных задач, основным является представление чисел с плавающей запятой, не требующее масштабирования данных. Однако в таких машинах часто наряду с этой формой представления чисел используется также и представление с фиксированной запятой для целых чисел, поскольку операции с такими числами выполняются за меньшее время. В частности, к операциям с целыми числами сводятся операции над кодами адресов (операции индексной арифметики).

Представление числа с плавающей запятой в общем случае имеет вид

$$x = s^p q; \; |q| < 1, \qquad (2.3)$$

где *q — мантисса* числа *x, s^p* — характеристика числа *x; p - порядок-, s* — основание характеристики (обычно целая степень числа 2).

Мантисса (правильная дробь со знаком) и порядок (целое число со знаком) представляются в системе счисления с основанием, равным *s* (в соответствующей двоично-кодированной форме). Знак числа совпадает со знаком мантиссы.

Порядок p, который может быть положительным или отрицательным целым числом, определяет положение запятой в числе *x.*

На рис. 1.2 показаны примеры форматов данных для чисел с плавающей запятой. Одна часть бит формата используется для представления порядка, а другая — для мантиссы.

Арифметические действия над числами с плавающей запятой требуют выполнения помимо операций над мантиссами определенных операций над порядками (сравнение, вычитание и др.). Для упрощения операций над порядками их сводят к действиям над целыми положительными числами (целыми числами без знаков), применяя представление чисел с плавающей запятой со *«смещенным порядком».*

В случае представления числа с плавающей запятой со смещенным порядком к его порядку p прибавляется целое число - смещение $N = 2^k$, где *k* — число двоичных разрядов, используемых для модуля порядка.

Смещенный порядок $p_{см}=p+N$ всегда положителен. Для его представления необходимо такое же число разрядов, как и для модуля и знака порядка *p.*

Важная особенность смещенных порядков состоит в том, что если для порядков *p'* и *p",* представляющих собой целые числа со знаками, выполняется соотношение

$$p' \geq p'',$$

то и для положительных целых чисел соответствующих смещенных порядков $p'_{см}$ и $p''_{см}$ всегда $p'_{см} \geq p''_{см}$. Это представление числа называют также полулогарифмическим, так как часть числа - характеристика - выражена в логарифмической форме.

Зн.п	2^{ln-2}	...	2^0	Зн.m	2^{-1}	2^{lm}

< Код порядка > < Код мантиссы >

< Длина поля порядка > < Длина поля мантиссы >

Знак '-' кодируется единицей, знак '+' - нулем.

Представление чисел с плавающей запятой

При фиксированном числе разрядов мантиссы любая величина представляется в машине с наибольшей возможной точностью нормализованным числом.

Число $x = s''q$ называется *нормализованным,* если мантисса q удовлетворяет условию

$$1 > |q| \geq 1/s, \qquad (2.4)$$

т. е. старший разряд мантиссы в s-ричной системе отличен от нуля. В процессе вычислений может получаться ненормализованное число. В этом случае машина, если это предписано командой, автоматически нормализует его («нормализация результата» операции).

Пусть r старших разрядов s-ричной мантиссы равны 0. Тогда нормализация заключается в сдвиге мантиссы на r разрядов влево и уменьшении порядка на r единиц, при этом в младшие r разрядов мантиссы записывается 0. После такой операции число не меняется, а условие (2.4) выполняется. При нулевой мантиссе нормализация невозможна.

В различных ЭВМ применяются представления чисел с плавающей запятой в системах счисления с различными основаниями, но равными целой степени числа 2 ($s = 2^w$), при этом порядок p представляется целым числом, а мантисса q — числом, в котором группы по w двоичных разрядов изображают цифры мантиссы с основанием системы счисления $s= 2^w$.

Примерами применяемых форм чисел с плавающей запятой с различными основаниями системы счисления являются

$$x=2^p q \ (1 > |q| \geq 1/2);$$
$$x=8^p q \ (1 > |q| \geq 1/8);$$
$$x = 16^p q \ (1 > |q| \geq 1/16).$$

В скобках указаны соответствующие условия получения нормализованных чисел.

Использование для чисел с плавающей запятой недвоичного основания несколько уменьшает точность вычислений (при заданном числе разрядов мантиссы), но позволяет увеличить диапазон представляемых в машине чисел и ускорить выполнение некоторых операций, в частности

нормализации, за счет того, что сдвиг может производиться сразу на несколько двоичных разрядов (на четыре разряда для s = 16). Кроме того, уменьшается вероятность появления ненормализованных чисел в ходе вычислений.

Диапазон представимых в машине чисел с плавающей запятой зависит от основания системы счисления и числа разрядов, выделенных для изображения порядка. Точность вычислений при плавающей запятой определяется числом разрядов мантиссы. С увеличением числа разрядов мантиссы увеличивается точность вычислений, но увеличивается и время выполнения арифметических операций.

Задачи, решаемые на ЭВМ, предъявляют различные требования к точности вычислений. Поэтому во многих машинах используется несколько форматов с плавающей запятой с различным числом разрядов мантиссы.

Прямой, обратный и дополнительный коды

В компьютерах с целью упрощения арифметических операций применяют специальные коды для представления чисел. При помощи этих кодов упрощается определение знака результата операции, операция вычитания (или алгебраического сложения) чисел сводится к арифметическому сложению их кодов, облегчается выработка признаков переполнения разрядной сетки. В результате упрощаются устройства компьютеров, выполняющие арифметические операции.

Для представления отрицательных чисел в компьютерах применяют прямой, обратный и дополнительный коды. Положительные числа представляются в прямом коде. Во всех этих кодах выделяются цифровые разряды и знаковый (крайний слева), представляющий знак числа, причем знак плюс кодируется цифрой 0, а знак минус цифрой 1.

Прямой код двоичного числа G с (n-1) цифровыми разрядами определяется как

$$G_{пр} = \begin{cases} G & при \quad G \geq 0; \\ A + |G| & при \quad G \leq 0, \end{cases}$$

где A — величина, равная весу знакового разряда. Для дробных чисел $A = 1$, а для целых $A = 2^{n-1}$.

Сложение в прямом коде чисел, имеющих одинаковые знаки, выполняется достаточно просто. Числа складываются, и сумме присваивается код знака слагаемых. Значительно более сложной является операция алгебраического сложения в прямом коде чисел с различными знаками. В этом случае приходится определять большое по модулю число, производить вычитание чисел и присваивать разности знак большего по модулю числа.

Операция вычитания (алгебраического сложения) сводится к операции простого арифметического сложения при помощи обратного и

дополнительного кодов, используемых для представления отрицательных чисел в машине.

Чтобы представить двоичное отрицательное число в обратном коде, нужно поставить в знаковый разряд 1, а во всех других разрядах заменить 1 нулями, а нули единицами.

Обратный код, если рассматривать его как число, является дополнением модуля исходного числа до наибольшего числа без знака, помещающегося в разрядную сетку. Для n-разрядной сетки имеем

$$G_{обр}=2-2^{-(n-1)}-|G^{-}|,$$

если G^{-} — двоичная дробь, и

$$G^{-}_{обр}=2^{n}-1-|G^{-}|$$

если G^{-} — целое двоичное число.

При представлении отрицательного двоичного числа в дополнительном коде ставят 1 в разряд знака, а цифровую часть числа заменяют дополнением модуля числа до 1 или 2^{n-1} соответственно для дробей и целых чисел. Дополнительный код отрицательного числа G определяется выражением

$$G^{-}_{доп}=2-|G^{-}|$$

если G^{-} — двоичная дробь, и

$$G^{-}_{доп}=2^{n}-|G^{-}|$$

если G^{-} — целое двоичное число.

Обратный и дополнительный коды числа можно рассматривать как двоичные числа без знаков, при этом для двоичных дробей $G^{-}_{доп}=G^{-}_{обр}+2^{-(n-1)}$, а для двоичных целых чисел $C^{-}_{доп}=C^{-}_{обр}+1$.

Таким образом, дополнительный код числа может быть получен из обратного путем прибавления 1 к младшему разряду обратного кода.

При выполнении расчетов на машине могут возникнуть как «положительный», так и «отрицательный» 0. Положительный 0 в прямом коде имеет вид

$$(+0)_{пр}=000...0.$$

Отрицательный 0 изображается в прямом коде

$$(-0)_{пр}=100...0,$$

в обратном

$$(-0)_{обр}=111...1;$$

в дополнительном коде отрицательный 0 отсутствует.

При представлении положительных чисел прямым кодом, а отрицательных дополнительным нуль имеет единственное изображение. При применении обратного кода «положительный» и «отрицательный» 0 имеют разные изображения.

Изменению знака отрицательного числа соответствует инвертирование его кода, если число представлено в обратном коде, и инвертирование и

добавление 1 младшего разряда, если отрицательное число представлено в дополнительном коде. В результате получается прямой код соответствующего положительного числа. Сказанное следует из соотношений:

для дробей

$-G^{-}_{пр} = |G^{-}| = 2 - 2^{-(n-1)} - G^{-}_{обр}$

$-G^{-}_{пр} = |G^{-}| = 2 - G^{-}_{доп}$

для целых чисел

$-G^{-}_{пр} = |G^{-}| = 2^{n} - 1 - G^{-}_{обр}$

$-G^{-}_{пр} = |G^{-}| = 2^{n} - G^{-}_{доп}$

Рассмотрим применение обратного и дополнительного кодов при алгебраическом сложении n-разрядных двоичных чисел G и *Q,* когда одно из них или оба числа отрицательны. Могут быть сформулированы следующие правила (предполагаем, что модуль алгебраической суммы меньше 1 для дробей и меньше 2^{n-1} для целых чисел, и, следовательно, код суммы представим в n-разрядной сетке).

При алгебраическом сложении двух двоичных чисел с использованием обратного (или дополнительного) кода положительные слагаемые представляются в прямом коде, а отрицательные - в обратном (дополнительном) и производится арифметическое суммирование этих кодов, включая разряды знаков, .которые при этом рассматриваются как старшие разряды. При возникновении переноса из разряда знака единица переноса прибавляется к младшему разряду суммы кодов при использовании обратного кода и отбрасывается при использовании дополнительного кода. В результате получается алгебраическая сумма в прямом коде, если эта сумма положительна, и в обратном (дополнительном), если она отрицательна.

Пример.

Пусть требуется выполнить две операции:

176- 154 и 176- 215

В первом случае результат будет положительный, а во втором – отрицательный. Переведем эти числа в двоичный код, например, через восьмеричную систему.

176 | 8	154 | 8	215 | 8
176 | 22 | 8	152 | 19 | 8	208 | 26 | 8
0 | 16 | 2	2 | 16 | 2	7 | 24 | 3
6	3	2

$176_{10}=260_{8}$ $154_{10}=232_{8}$ $215_{10}=327_{8}$

010110000_{2} 010011010_{2} 011010111_{2}

инвертируем вычитаемые в обратный и дополнительный коды

$$\begin{array}{r} 101100101_{\text{обр}} \\ + \quad\quad 1 \\ \hline 101100110_{\text{доп}} \end{array} \qquad \begin{array}{r} 100101000_2 \\ + \quad\quad 1 \\ \hline 100101001_{\text{доп}} \end{array}$$

теперь складываем уменьшаемое с дополнительными кодами обоих вычитаемых с учётом знаков

$$\begin{array}{r|l} 0 & 010110000 \\ + \ 1 & 101100110 \\ \hline 10 & 000010110 \end{array} \qquad \begin{array}{r|l} 0 & 010110000 \\ + \ 1 & 100101001 \\ \hline 1 & 111011001 \end{array}$$

в первой операции возникла единица переполнения, но знак результата плюс, поэтому переводим число в восьмеричную систему и далее в десятичную: $26_8=2\cdot8^1+6\cdot8^0=22_{10}$, что соответствует результату первой операции.

во второй операции знак результата отрицательный, поэтому его придется перевести в дополнительный код:

$$\begin{array}{r} 111011001 \\ 000100110_{\text{обр}} \\ + \quad\quad 1 \\ \hline 000100111_{\text{доп}} \end{array} =47_8=39_{10}$$

или с учётом знака -39.

Признак переполнения разрядной сетки. При алгебраическом сложении двух чисел, помещающихся в разрядную сетку, может возникнуть переполнение, т. е. образоваться сумма, требующая для своего представления на один цифровой разряд больше по сравнению с разрядной сеткой слагаемых. Можно сформулировать следующее правило (признак) для обнаружения переполнения разрядной сетки.

При алгебраическом сложении двух двоичных чисел с использованием прямого кода для представления положительных и дополнительного (обратного) кода для представления отрицательных чисел признаком переполнения является наличие переноса в знаковый разряд суммы при отсутствии переноса из ее знакового разряда (положительное переполнение) или наличие переноса из знакового разряда суммы при отсутствии переноса в ее знаковый разряд (отрицательное переполнение). Если и в знаковый, и из знакового разряда суммы есть переносы или нет этих переносов, то переполнение отсутствует. При положительном переполнении результат операции положительный, а при отрицательном отрицательный.

Представление информации в компьютере.

Оперативная память (ОП) компьютера предназначена для временного хранения двух типов данных: программ и данных для обработки. Программы используются процессором для выполнения действий над данными. Данные

позволяют хранить информацию различных типов: чисел, текста, даты и времени. Структура ОП состоит из элементарных единиц памяти – *байтов*, и средства доступа к ним – *адреса*. Более подробно память компьютера будет рассмотрена ниже.

В компьютере данные представляются последовательностью нескольких битов или байтов называемых полем данных. Поля могут быть постоянной и переменной длины. К полям постоянной длины относят:

Слово – 2 байта

Полуслово – 1 байт

Двойное слово – 4 байта

Расширенное слово – 8 байтов.

Числа с фиксированной запятой чаще всего имеют формат слова и полуслова, числа с плавающей запятой – формат двойного и расширенного слова.

Поля переменной длины имеют любой размер от 0 до 256 байт, по обязательно равный целому числу байт.

Двоично-кодированные десятичные числа могут быть представлены в компьютере полями переменной длины в упакованном и распакованном форматах.

В упакованном формате для каждой десятичной цифры отводится по 4 разряда, при этом знак числа кодируется в крайнем правом полубайте числа (1100 – знак "+" и 1101 – знак "-"). Упакованный формат используется в ПК при выполнении операций сложения и вычитания.

В распакованном формате для каждой десятичной цифры отводится 1 байт, при этом старшие полубайты (зона) каждого байта (кроме самого младшего) в ПК заполняется кодом 0011 (в соответствии с ASCII-кодом), а в младших (левых) полубайтах обычным образом кодируются десятичные цифры. Старший полубайт (зона) самого младшего (правого) байта используется для кодирования знака числа. Распакованный формат используется при вводе-выводе информации в компьютере, и также при выполнении операций умножения и деления двоично-десятичных чисел.

Пример. Число – 172 = - 0001. 0111. 0010 в ПК будет представлено:

В упакованном формате

0001	0111	0010	1101

В распакованном формате

0011	0001	0011	0111	0011	0010	0011	1101

Распакованный формат представления двоично-десятичных чисел является следствием использования в компьютере ASCII-кода для

представления символьной информации.

Код ASCII (American Standard Code for Information Intechange – Американский стандартный код для обмена информацией) имеет основной стандарт и его расширение.

В данном разделе были рассмотрены первичные представления хранения данных. Принципы хранения других типов данных будут рассмотрены далее.

Принцип программного управления

Пусть требуется вычислить выражение Y = $\frac{a+b}{x^2-d}$ при известных значениях составных переменных. Не прибегая к помощи компьютера данное выражение можно вычислить с помощью последовательности инструкций (словесного алгоритма).

1. Сложить данные с именами а и ь, результат сохранить, присвоив ему имя P1.
2. Умножить данное с именем x на это же данное, результат сохранить, присвоив ему имя P2.
3. Из промежуточного результата P2 вычесть данное с именем d, результат сохранить, присвоив ему имя P3.
4. Разделить данное с именем P1 на данное с именем P3, полученный результат сохранить под именем R.
5. Выдать окончательный результат с именем R.
6. Закончить вычисления.

Для реализации этого алгоритма в компьютере необходимо все его буквенные обозначения закодировать цифрами, поскольку любой компьютер оперирует только с цифрами. Вначале закодируем все элементарные операции данного алгоритма (для наглядности используем числа в десятичной системе счисления):

сложение (+) - 01; деление (:) - 04;
вычитание (-) - 02; выдача результата (Pr) - 05;
умножение (*) - 03; конец вычислений (E) - 06.
Эти номера называются КОП - коды операций.
Вместо букв имен данных введем их адреса:
01 - a; 02 - ь; 03 - *; 04 - d;
05 - P1; 06 - P2; 07 - P3; 08 - R.

Таким образом, первую инструкцию в закодированном виде можно записать последовательностью чисел:

КОП	A1	A2	A3

01	01	02	05

где

первое число (КОП) обозначает операцию (+);

второе число (операнд А1) обозначает адрес первого операнда а;

третье число (операнд А2) - адрес второго операнда в;

четвертое число (операнд А3) - адрес первого промежуточного результата Р1.

Закодированная в виде последовательности чисел инструкция о том, какую операцию и над какими операндам» необходимо выполнять и где сохранить результат, называется командой.

Таким образом, команда - это машинное слово, которое содержит код соответствующей операции и адреса операндов. В свою очередь, операнд - это машинное слово, в котором хранится данное для выполнения операции.

Каждая команда содержит КОП и адресную часть. Если в адресной части отведено место для трех адресов, то такие машины называют трехадресными. Однако существуют машины и с большей, и с меньшей адресностью. Правило записи команды называют форматом команды.

Последовательность команд называют *программой*. Для команд программы выделяют последовательность байтов с последовательными номерами так, чтобы их выборку можно было бы осуществить последовательно друг за другом. Применительно к рассмотренному примеру алгоритм и программа имеют следующий вид:

Алгоритм	КОП	А1	А2	А3
+ a b P1	01	01	02	05
▪ x x P2	03	03	03	06
- P2 d P3	02	06	04	07
: P3 P1 R	04	05	07	08
Pr R - -	05	08	-	-
E - - -	06	-	-	-

Черточки означают, что числа, заданные на этих позициях, на работу машины не влияют.

Из сопоставления алгоритма и программы следует, что программа - это алгоритм решения задачи, описанный на языке цифровых слов (команд). И программа и аппаратура компьютера строятся по единому замыслу, который основывается на следующих ключевых идеях:

1. Числовое кодирование и операций, и операндов, а также хранение и тех, и других в одной ОП (принцип числовое кодирования).
2. Введение адресности операндов и промежуточных результатов (принцип адресности).

3. Хранение команд в ОП, последовательная их выборка и исполнение (принцип хранимости программ и последовательнго их исполнения).

Совокупность перечисленных трех идей называют принципом программного управления, который состоит в том, что компьютер автоматически, без участия человека, реализует предусмотренную программу. Этот принцип положен в основу функционирования компьютеров.

Логические основы функционирования компьютеров

В компьютерах принцип программного управления реализуется с помощью арифметико – логического устройства (процессора), в основе устройства которого лежат аппаратные средства преобразования двоичных кодов. Математической основой работы этих устройств является алгебра логики.

В основе алгебры логики лежат понятия зависимой и независимой переменных (функции и аргумента). Их отличия от соответствующих понятий обычной алгебры в том, что переменные могут принимать лишь одноиз двух значений: 0 или 1. Пусть $x_1, x_2,..., x_n$ - двоичные аргументы, тогда функция $y= f(x_1...x_n)$ является двоичной функцией двоичных аргументов, которая также может принимать значение либо 0, либо 1. Данная функция задается или аналитически, либо таблично. При табличном представлении значение функции должно быть задано на всех возможных наборах значений аргумента. Общее число этих наборов равно 2^n, где n - число аргументов, а общее число всех возможных двоичных функций n двоичных аргументов будет 2^{2n}.

Существует всего четыре функции одного двоичного аргумента, которые можно представить таблично или аналитически. Функция Y_0, принимающая нулевые значения на всех наборах, называется константой нуля $Y_0=0$ (Y_0 тождественна 0).

X	Y_0	Y_1	Y_2	Y_3
0	0	0	1	1
1	0	1	0	1

Функция Y_1, значения которой совпадают со значениями аргумента, называется тавтологией, или повторением, и записывается аналитически

Y_1= X (Y_1 тождественна **X).** Функция, значения которой противоположны значениям аргумента, называется инверсией, или отрицанием. В таблице такой функцией является **$Y_2= \bar{X}$ (Y_2** тождественна не **X).** Черта над аргументом означает отрицание или инверсное значение. Значение **Y_3** не изменяется при

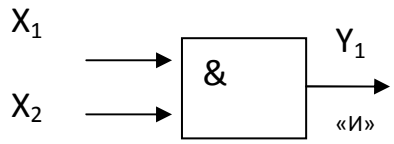
изменениях аргумента и всегда тождественно единице **Y_3=1** Это функция константа единицы.

Рассмотрим двоичные функции двух (п=2) двоичных аргументов. Всего таких функций 16, каждая из которых определена на 2^2=4 наборах значений аргумента.

Среди множества указанных функций существует три, которые позволяют записать аналитически любую логическую функцию. Это функции конъюнкции, дизъюнкции и инверсии. Рассмотрим их более подробно.

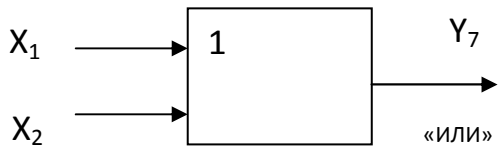
X_1	X_2	Y_1	Y_7	Y_{12}	Y_{10}
0	0	0	0	1	1
0	1	0	1	1	0
1	0	0	1	0	1
1	1	1	1	0	0

Функция $Y_1=X_1 \wedge X_2$ соответствует операции логического умножения и называется конъюнкцией. Знак Λ (или &) обозначает действие логического умножения. Иногда она обозначается как «·». Это действие выполняет устройство, которое называется автомат "И". На схемах это устройство обозначается так:

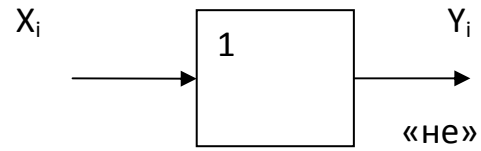


Функция Y_1 обращается в единицу только при единичных значениях аргумента, во всех остальных случаях Y_1 тождественна нулю и ничем не отличается от арифметического произведения.

Функция $Y_7 = X_1 \vee X_2$ похожа (за исключением последней строки на операцию арифметического сложения и соответствует операции логического сложения (v - знак логического сложения). Данная функция называется дизъюнкцией и обращается в нуль только тогда, когда аргументы равны нулю, и в единицу - во всех остальных случаях. Это действие выполняет автомат "ИЛИ", имеющий следующее обозначение



Функции $Y_{12}= \overline{X}_1$ и $Y_{10}= \overline{X}_2$ соответствуют операции инверсии (или отрицания) и выполняются автоматом "НЕ", со следующим обозначением.



Но если операции "И" и "ИЛИ" могут выполняться над произвольным числом аргументов n > 2, то операция «НЕ» всегда выполняется над одним аргументом.

Среди множества двоичных функций есть такие, которые принимают значение 1 лишь на одном наборе двоичных аргументов. Такие функции называются конституентами единицы. Для n =2 данные функции приведены в следующей таблице.

X_1	X_2	Y_1	Y_2	Y_4	Y_8
0	**0**	**0**	**0**	**0**	**1**
0	**1**	**0**	**0**	**1**	**0**
1	**0**	**0**	**1**	**0**	**0**
1	**1**	**1**	**0**	**0**	**0**

$Y_8=\overline{X}_1 \cdot \overline{X}_2$
$Y_4=\overline{X}_1 \cdot X_2$
$Y_2=X_1 \cdot \overline{X}_2$
$Y_1=X_1 \cdot X_2$

Аналитически их записывают по следующему правилу. Из таблицы выбирается строка, в которой функция принимает единичное значение и для нее записывается конъюнкция всех аргументов. Если в строке аргумент имеет нулевое значение, то над ним выполняется еще операция отрицания.

Аналитическое выражение для любой заданной таблично двоичной функции можно записать с помощью совершенной дизъюнктивной нормальной формы (СДНФ). СДНФ - это дизъюнкция конституент единицы, записанных для тех наборов двоичных аргументов, на которых функция принимает единичное значение. Пусть функция y= $f(x_1,x_2,x_3)$ задана в виде нижеприведённой таблицы.

X_1	X_2	X_3	Y
0	0	0	0
0	0	1	0
0	1	0	1
0	1	1	0
1	0	0	1
1	0	1	1
1	1	0	0
1	1	1	0

Аналитическая запись данной функции в СДНФ будет представлять дизъюнкцию трех конституент единицы, записанных для **3,5** и **6** строк

таблицы, т.е.

Y=$\overline{X}_1 \cdot X_2 \cdot \overline{X}_3 \; V \; X_1 \cdot \overline{X}_2 \cdot \overline{X}_3 \; V \; X_1 \cdot \overline{X}_2 \cdot X_3$

Таким образом, с помощью трех функций и соответствующих им трех логических действий можно записать или формально выразить любую логическую функцию. Другими словами, в алгебре логики определено только три логических действия, никаких других действий нет. Порядок выполнения действий: сначала выполняется инверсия, затем конъюнкция и последним - дизъюнкция. Если есть скобки, то сначала выполняются операции в скобках.

При выполнении логических действий необходимо знать следующие основные соотношения для логических сложения и умножения:

x v 0 = x x v 1 =1 x v x = x x v $\overline{x}$ = 1

x Λ 0 = 0 x Λ 1 = x x Λ x = x x Λ $\overline{x}$ = 0

$\overline{\overline{x}}$ = x x v x v … v x = x x Λ x Λ … Λ x = x

Из последних соотношений следует, что в любом логическом выражении можно сколько угодно раз добавлять любой из логических членов. Например,

$$Y = \overline{X}_1 X_2 \overline{X}_3 \; = \; \overline{X}_1 X_2 X_2 \overline{X}_3 \overline{X}_3$$

Основные законы алгебры логики:

1. *Переместительный закон*. От перестановки мест двоичных аргументов значение логического выражения не изменяется:

$X_1 V X_2 = X_2 V X_1$

2. *Сочетательный закон*. Значение логического выражения не зависит от последовательности действий над логическими переменными.

$X_1 X_2 X_3 = X_1 (X_2 X_3) = (X_1 X_2) X_3$

$X_1 \text{ v } X_2 \text{ v } X_3 = (X_1 \text{ v } X_2) V X_3 = X_1 \text{ v } (X_2 \text{ v } X_3)$

3. *Первый распределительный закон*: $X_1 (X_2 \text{ v } X_3) = X_1X_2 \text{ v } X_1X_3$
   Из приведенных законов следует, что, как и в обычной алгебре, логические переменные можно менять местами и выносить за скобки. Однако в алгебре логики есть еще законы, которые не аналогов в обычной алгебре.
4. *Второй распределительный закон*: $X_1 \text{ v } X_2X_3 = (X_1 \text{ v } X_2)(X_1 \text{ v } X_3)$
5. *Закон инверсии*. Этот закон базируется на теореме де Моргана,

которая формулируется следующим образом. При замене в исходной, логической функции аргументов их отрицаниями, знаков логического сложения знаками логического умножения, а знаков логического умножения знаками логического сложения получается функция, являющаяся инверсной от исходной :

$$\overline{X}_1\overline{X}_2 \dots \overline{X}_n = \overline{X_1 \, V \, X_2 \, V \dots V \, X_n}$$
$$\overline{X}_1 \, V \, \overline{X}_2 \, V \dots V \, \overline{X}_n = \overline{X_1 X_2 \dots X_n}$$

Указанные соотношения и законы позволяют проводить анализ и синтез логических схем, одним из этапов которых является построение СДНФ. Рассмотрим способы образования СДНФ для заданных аналитически логических выражений.

Первый способ заключается в том, что для заданной аналитически функции строится таблица истинности, из которой по рассмотренному выше правилу записывается СДНФ.

Пример. Построить СДНФ для функции $Y = \overline{X_1 X_2} \, V \, X_3$

Функция содержит три аргумента, для которых в таблице истинности заполняем 2^3=8 строк. Подставляя входные наборы аргументов в заданную функцию, определяем значения Y**.**

X1	X2	X3	Y
0	0	0	1
0	0	1	1
0	1	0	1
0	1	1	1
1	0	0	1
1	0	1	1
1	1	0	0
1	1	1	1

Так, для первой строки $Y(0,0,0) = \overline{0 \cdot 0} \, V \, 0 = \overline{0} \, V \, 0 = 1 \, V \, 0 = 1$ Проводя аналогичные действия для всех строк, заполняем столбец Y. Столбец для Y мог быть заполнен и исходя из анализа исходной функции. Так, функция Y будет принимать значение 1 в том случае, если X_3= 1 либо $\overline{X_1 X_2} = 1$. Последнее тождество возможно, когда либо X_1 =0, либо X_2=0. Т.е. Y=1 на тех наборах, когда либо X_1=0**,** либо X_2 =0**,** либо X_3=1. Составив таблицу, СДНФ запишем как дизъюнкцию семи конституент единицы:

$$Y = \overline{X}_1\overline{X}_2\overline{X}_3 \, V \, \overline{X}_1\overline{X}_2 X_3 \, V \, \overline{X}_1 X_2\overline{X}_3 \, V \, \overline{X}_1 X_2 X_3 \, V \, X_1\overline{X}_2\overline{X}_3 \, V \, X_1\overline{X}_2 X_3 \, V \, X_1 X_2 X_3$$

Второй способ образования СДНФ заключается в том, что:

- на основании теоремы де Моргана инверсии дизъюнкций и конъюнкций заменяются на конъюнкции и дизъюнкции инверсий аргументов;
- раскрываются скобки во всех логических выражениях;
- образуются конституенты единицы домножением членов, не содержащих X_i , i= 1,2, … , n на $\left(X_i \, V \, \overline{X}_i\right) = 1$ с последующим раскрытием скобок;
- упорядочиваются соответствующие индексы во всех наборах аргументов.

Пример. Получить СДНФ для функции $Y = \overline{X_1 X_2} \, V \, \overline{X_3 (X_1 \, V \, X_3)}$

Пользуясь теоремой де Моргана и проводя упрощения, получим

$$Y = \overline{X_1 X_2} \, V \, \overline{X}_3 \, V \, \overline{X}_1 \overline{X}_2 = \overline{X}_1 \, V \, \overline{X}_2 V \, \overline{X}_3 V \overline{X}_1 \overline{X}_2 = \overline{X}_1 \left(1 \, V \, \overline{X}_2\right) V \, \overline{X}_2 V \, \overline{X}_3$$
$$= \overline{X}_1 V \, \overline{X}_2 V \, \overline{X}_3$$

Домножим все члены на недостающее значение аргументов:

$$Y = \overline{X}_1 \left(X_2 V \, \overline{X}_2\right)\left(X_3 V \, \overline{X}_3\right) V \, \overline{X}_2 \left(X_1 V \, \overline{X}_1\right)\left(X_3 V \, \overline{X}_3\right) V \, \overline{X}_3 (X_1 V \, \overline{X}_1)(X_2 V \overline{X}_2)$$

Раскрывая скобки и приводя подобные, СДНФ получим в виде

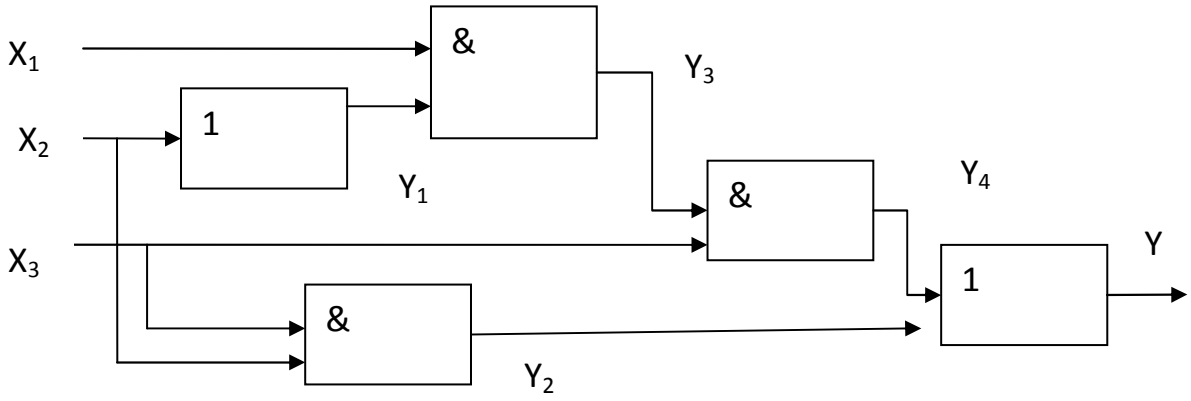
$$Y = \overline{X}_1 \overline{X}_2 \overline{X}_3 V \, \overline{X}_1 \overline{X}_2 X_3 V \, \overline{X}_1 X_2 \overline{X}_3 V \, \overline{X}_1 X_2 X_3 V \, X_1 \overline{X}_2 \overline{X}_3 V \, X_1 \overline{X}_2 X_3 V \, X_1 X_2 X_3$$

Анализ комбинационных схем

Проанализировать логическую схему это означает: по заданной структурной схеме определить таблицу истинности и (или) логическое выражение, реализуемое данным автоматом. Т.е. установить зависимость выходной функции от входных значений аргументов. Анализ осуществляется в следующей последовательности:

1. Для каждого из элементарных автоматов записывается логическое выражение. При этом образуется система логических уравнений.
2. Система логических уравнений решается методом подстановок
3. По полученному выражению строится таблица истинности и если необходимо, записывается СДНФ.
4. Находится тупиковая форма, т.е. логическое выражение, не допускающее дальнейшего упрощения.

Пример. Для представленного автомата найти логическую функцию, которую воспроизводит данный автомат, и, если возможно, упростить его структуру для реализации той же функции.



1. Обозначим выход каждого из элементарных автоматов через Y_1 и запишем реализуемые ими логические выражения

$$Y_1 = \overline{X}_2, Y_2 = X_2X_3, Y_3 = X_1Y_1, Y_4 = X_3Y_3, Y = Y_2 V\ Y_4$$

2. Получим систему логических уравнений, которую решаем методом подстановки, начиная с выходного элементарного автомата:

$$Y = Y_2 V\ Y_4 = X_2X_3 V\ X_3Y_3 = X_2X_3 V\ X_3X_1Y_1 = X_2X_3 V\ X_3\overline{X}_1X_2$$

3. Составим таблицу истинности С этой целью запишем в СДНФ, для чего домножим первый член на $X_1 V\ \overline{X}_1 = 1$

$$Y = \left(X_1 V\ \overline{X}_1\right)X_2X_3 V\ X_1\overline{X}_2X_3 = X_1X_2X_3 V\ \overline{X}_1X_2X_3 V\ X_1\overline{X}_2X_3$$

Получили три конъюнктивных члена, которые являются конституентами единицы, т.е. будет принимать значение единицы три раза на наборах аргументов, соответствующих конъюнктивным членам СДНФ. По выражению заполняем столбец .

X1	X2	X3	У
0	0	0	0
0	0	1	0
0	1	0	0
0	1	1	1
1	0	0	0
1	0	1	1
1	1	0	0
1	1	1	1

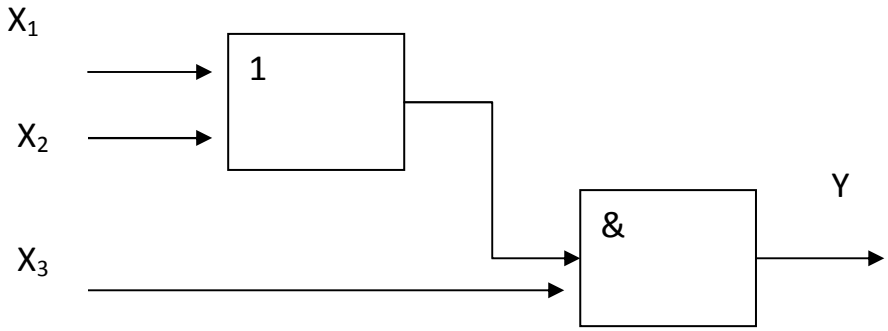
4. Упрощение полученного выражения и нахождение тупиковой формулы. Из выражения следует, что, вынося X_3 за скобки применяя к выражению в скобках 2-й распределительный закон получим тупиковую формулу

$$Y = X_3\left(X_2\ V\ X_1\overline{X}_2\right) = X_3\left(\left(X_2\ V\ X_1\right)\left(X_2\ V\ \overline{X}_2\right)\right) = X_3(X_2\ V\ X_1)$$

Можно тупиковую формулу получить и из выражения СДНФ. На основании основных соотношений алгебры логики в формуле можно добавить любое число конъюнктивных членов, входящих в исходное выражение. Добавив конъюнктивный член $X_1X_2X_3$ и проведя операции выноса за скобки, получим

$$Y = X_1X_2X_3\ V\ \overline{X}_1X_2X_3\ V\ X_1\overline{X}_2X_3\ V\ X_1X_2X_3 = X_2X_3\left(X_1\ V\ \overline{X}_1\right)V\ X_1X_3\left(\overline{X}_2\ V\ X_2\right)$$
$$= X_3(X_2\ V\ X_1)$$

По данному выражению можно построить автомат, который реализует ту же функцию Y, но имеет значительно более простую структуру, состоящую только из двух элементарных автоматов.



Синтез комбинационных схем

Задача синтеза формулируется следующим образом. По заданной словесной формулировке задачи определить структурную схему автомата с минимальным числом заданных логических элементов. Наиболее рациональной при этом является следующая последовательность:

1. Словесное описание задачи.
2. Переход от словесного описания к описанию с помощью таблицы истинности.
3. Запись и минимизация СДНФ.
4. Составление структурной схемы автомата.

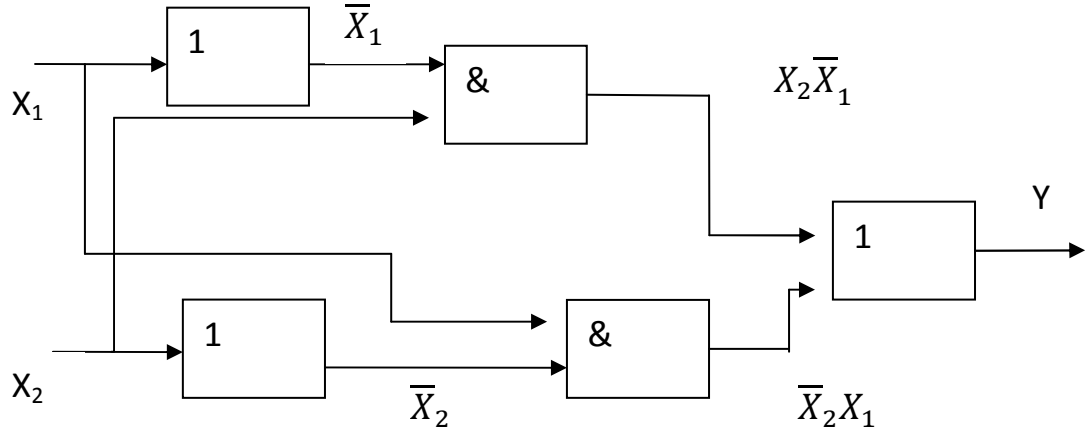
Пример. Составить автомат, определяющий знак произведения двух сомножителей.

Словесная формулировка задачи предполагает, что два сомножителя с одинаковыми знаками дают положительное значение произведения, а сомножители с разными знаками - отрицательное.

В качестве двух аргументов принимаем знак первого сомножителя X_1 и знак второго сомножителя X_2, причем положительное значение будем кодировать "0", а отрицательное "1". В качестве логической функции Y примем знак произведения. Исходя из приведенной словесной форму лировки задачи составляется таблица истинности.

X_1	X_2	Y
0	0	0
0	1	1
1	0	1
1	1	0

На основе получаемой таблицы истинности может быть записана СДНФ $Y = \overline{X}_1 X_2 \, V \, X_1 \overline{X}_2$ из которой следует, что произведение отрицательно (Y=1), если знаки сомножителей (X_1 и X_2) различны. Полученная аналитическая зависимость является тупиковой и не допускает дальнейшего упрощения.



Последний этап синтеза предполагает построение структурной схемы автомата, который будет содержать два элемента "не", элемента "и" и один элемент "или".