МАТЕМАТИЧЕСКИЕ И ЛОГИЧЕСКИЕ ОСНОВЫ ВЫЧИСЛИТЕЛЬНОЙ ТЕХНИКИ

Оглавление

ВВЕДЕНИЕ

Считать приходилось человеку уже очень давно. При этом использовались некоторые вполне определенные способы наименования и записи чисел, то есть определенная система счисления.

Первые приспособления для вычислений облегчали не столько сам процесс счета, сколько запоминание исходных чисел и промежуточных результатов. Числа изображались в привычной цифровой форме в определенной системе счисления (почти всегда — десятичной, которая стала общепризнанной). Такими же были и первые вычислительные машины.

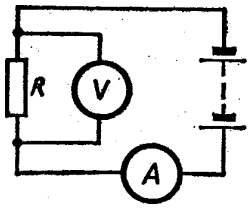
Существует два различных способа изображения чисел в вычислительных машинах и приборах. Первый из них условно называется ***цифровым.*** Этот способ реализуется в русских счетах и большом количестве различных приспособлений и машин.

Второй способ изображения чисел называют ***непрерывным*** или ***аналоговым.*** Здесь для изображения числа применяется некоторая физическая величина, имеющая непрерывный характер. Это может быть длина отрезка или угол поворота вала, ток или напряжение в электрической цепи, температура, давление и другие физические величины. Простейшим примером вычислительного устройства с непрерывным способом изображения чисел является логарифмическая линейка. На ней число изображается длиной отрезка.

Различие в способах представления чисел на русских счетах и логарифмической линейке состоит в физической природе величин. Непрерывные величины, в отличие от цифровых (дискретных) величин, могут принимать любые вещественные значения. Это различие положено в основу классификации вычислительных машин: существуют ***цифровые (дискретные) машины*** и ***аналоговые (непрерывные) машины***. Применение того или иного способа изображения чисел диктует свои принципы устройства.

Современные вычислительные машины являются цифровыми и используют электронные элементы. В них используется двоичная система счисления. Если изображать числа с помощью каких-либо электрических величин, то нужно создать электрические схемы, позволяющие выполнять над этими величинами те или иные операции.

Рассмотрим простой пример схемы, которую можно использовать для умножения и деления и в которой числа изображаются электрическими величинами. Электрическая цепь состоит из источника, например гальванической батареи, и резистора. Известно, что ток такой цепи подчиняется закону Ома $I = \frac{E}{R}$, где I — ток; E — напряжение и R — сопротивление.



Включив в цепь амперметр и вольтметр, мы можем по показаниям амперметра находить частное от деления E на R, а по показаниям вольтметра — произведение IR, придавая остальным величинам нужные значения. Таким образом, эта простейшая электрическая схема может быть использована для умножения и деления. Числа в ней изображаются электрическими величинами — напряжением, током и сопротивлением.

Машинные элементы делятся на ***логические***, ***запоминающие*** и ***вспомогательные***. Из логических элементов строят операционные схемы, выполняющие арифметические и иные операции. Запоминающие элементы хранят информацию. Вспомогательные элементы предназначены для формирования стандартных сигналов и согласования режимов работы операционных схем. Это деление условно, так как одни и те же элементы могут использоваться различным образом: запоминающий элемент — ***триггер*** — может быть построен из логических элементов, а операционная схема — ***регистр*** — может рассматриваться как запоминающее устройство.

Результат любой операции, выполняемой в машине, есть двоичное число. Поэтому операционная схема представляет собой функциональный преобразователь, на входах и выходах которого двоичные числа. Для удобства аргументами и значениями функции считают отдельные разряды двоичных чисел. Таким образом, речь идет о переменных и функциях, принимающих лишь значения 0 и 1. Такие переменные и функции носят название ***логических*** или ***булевых функций*** — по имени английского математика

и логика Джорджа Буля (1815—1864) — создателя современной символической логики. Математическая логика, которую называют иногда алгеброй логики, занимается подробным изучением булевых функций.

В данном учебном пособии мы рассмотрим арифметические и логические основы построения вычислительных машин (ВМ).

1. АРХИТЕКТУРА ПЕРСОНАЛЬНОГО КОМПЬЮТЕРА

Под архитектурой ЭВМ понимают описание устройства и принципов работы компьютера, достаточное для пользователя и программиста. Всякий компьютер имеет следующие основные блоки:

• ***устройства ввода,*** предназначенные для кодирования и передачи в оперативную память данных, необходимых для решения задачи, а также программы для их обработки;

• ***оперативное запоминающее устройство,*** служащее для хранения вводимых в машину данных, результатов промежуточных и окончательных вычислений и программ обработки данных;

• ***процессор,*** включающий в себя устройство управления, организующее последовательность действий по обработке данных, и арифметико-логическое устройство, непосредственно выполняющее операции;

• ***устройства вывода,*** предназначенные для декодирования и вывода результатов обработки данных и программ;

• ***внешние запоминающие устройства,*** обеспечивающие неограниченно долгое хранение любой информации (данных, программ);

• ***системный канал,*** служащий для передачи закодированной информации между устройствами в виде электрических импульсов низкого и высокого уровней.

Для представления информации в вычислительной технике преимущественное распространение получило *равномерное двоичное кодирование*, при котором символы вводимой в ЭВМ информации представляются средствами двоичного алфавита — **0** и **1**. Каждому вводимому символу ставится в соответствие определенная кодовая комбинация — некоторая цепочка двоичных знаков.

Работа современных компьютеров базируется на следующих принципах:

1) ***принцип программного управления:*** программа состоит из набора команд, которые выполняются процессором автоматически в определенной последовательности;

2) ***принцип адресности:*** память состоит из отдельных ячеек, каждая из которых имеет свой номер (адрес);

3) ***принцип однородности памяти:*** и программы и данные, представленные в двоичном коде, хранятся в памяти; не важно, что хранится в данной ячейке памяти — число, текст или команда.

Далее мы попытаемся ответить на вопрос, каким же образом представляются в компьютере данные (слова, числа, рисунки, звуки) и программы.

2. ИНФОРМАЦИЯ. ЕДИНИЦЫ ПРЕДСТАВЛЕНИЯ ИНФОРМАЦИИ

Информация — одно из фундаментальных понятий. Термин «информация» этимологически происходит от латинскому слова «informatio» — разъяснение, осведомление, изложение.

«Словарь иностранных слов» (М., 2004) так толкует значение этого термина:

1) сообщение о чем-либо;

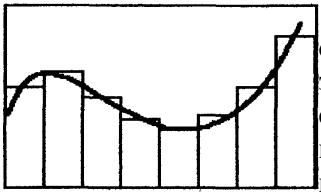
2) сведения, являющиеся объектом хранения, переработки и передачи;

3) в математике, кибернетике — количественная мера устранения неопределенности (***энтропии***), мера организации системы.

Информация — это сведения, знания, которые мы получаем из книг, газет, радио, телевидения, от людей, с которыми мы общаемся. Передача информации присуща всем явлениям природы. Информация может быть представлена в аналоговой или дискретной форме. Если электрическое напряжение или ток изменяется по тому же закону, что и некоторая другая физическая величина, то их называют электрическими аналогами этой физической величины**.**

При аналоговом представлении информации используемая в качестве ее носителя физическая величина принимает бесконечное множество значений, она изменяется непрерывно.

При дискретном (цифровом) представлении информации используемая в качестве ее носителя физическая величина принимает конечное множество значений, которые обозначаются какими-либо символами.



На рисунке наклонная плоскость соответствует бесконечному количеству значений высоты. Увеличивая число ступенек, лестницу с какой угодно точностью можно приблизить к наклонной плоскости. Так же и любая аналоговая величина может быть заменена конечным набором дискретных значений.

Дискретную информацию можно передавать двумя принципиально разными способами:

1) последовательно (сигналы передаются один за другим по единственной линии связи);

2) параллельно (группа сигналов передается по нескольким линиям связи одновременно).

Представление дискретной информации в стандартных символических формах называется ***кодированием***. ***Декодирование*** — процесс, обратный кодированию.

Числа — удобная форма представления информации, природа которой может быть самой различной: от показаний всевозможных датчиков до сообщений на привычном нам языке.

Различают ***равномерные и неравномерные коды***. Равномерные коды в кодовых комбинациях содержат одинаковое число знаков, неравномерные — неодинаковое. Примером неравномерного кода может служить азбука Морзе, в которой для каждой буквы и цифры определена последовательность коротких и длинных сигналов. Так, букве Е соответствует короткий сигнал (точка), а букве Ш — четыре длинных сигнала (четыре тире). Неравномерное кодирование позволяет повысить скорость передачи сообщений за счет того, что наиболее часто встречающиеся в передаваемой информации символы имеют самые короткие кодовые комбинации.

Информация играет определяющую роль в жизни современного общества. Но безошибочная обработка огромных информационных потоков трудна и утомительна для человека в силу его психологических особенностей.

Современная вычислительная машина (компьютер) — это комплекс аппаратных и программных средств, предназначенный для автоматизации процесса обработки информации.

Информацию, представленную различными устойчивыми состояниями некоторого физического носителя в форме, воспринимаемой и обрабатываемой компьютером или человеком, называют ***данными.***

Информацию о последовательности операций, которые необходимо осуществить для получения по исходным данным требуемого результата, называют ***программой.***

Автоматическая обработка данных означает, что весь процесс вплоть до получения результата происходит без участия человека. При этом исходные данные и программы могут быть заданы, составлены или подготовлены человеком.

Количество введенной в ЭВМ информации измеряется ее «длиной», выраженной в двоичных знаках — ***битах*** (от англ. binary digit — двоичная цифра). Последовательность из восьми двоичных битов называется ***байтом.*** Для кодирования любого символа текста требуется 1 байт информации.

Большие объемы информации измеряются с помощью производных единиц: килобайт, мегабайт и гигабайт.

1 Кбайт $= 2^{10}$ байт = 1024 байт; 1 Мбайт $= 2^{20}$ байт = 1024 Кбайт;

1 Гбайт $= 2^{30}$ байт = 1024 Мбайт.

П р и м е р. Измерим в байтах объем текстовой информации в «Современном словаре иностранных слов» из 740 страниц, если на одной странице размещается в среднем 60 строк по 80 символов (включая пробелы).

Подсчитаем количество символов во всем словаре:

$80 \cdot 60 \cdot 740 = 3\,552\,000$ байт = 3468,8 Кбайт = 3,39 Мбайт.

Таким образом, объем информации в словаре в двоичной форме равен примерно 3,4 Мбайт.

3. АРИФМЕТИЧЕСКИЕ ОСНОВЫ ВЫЧИСЛИТЕЛЬНОЙ ТЕХНИКИ

Известно множество способов представления чисел. В любом случае число изображается символом или группой символов (словом) некоторого алфавита. Такие символы называются *цифрами*, символические изображения чисел — *кодами*, а правила их получения — *системами счисления* (кодирования).

Система счисления **—** это совокупность приемов и правил для обозначения и наименования чисел.

Алфавит системы счисления **—** это множество всех символов (знаков), используемых для записи чисел в данной системе счисления.

Цифра — это любой символ (знак), входящий в алфавит данной системы счисления.

Системы счисления делятся на следующие виды:

1) системы бирок (унарные);

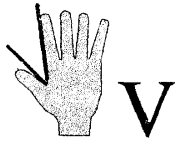
2) кодовые (непозиционные) системы;

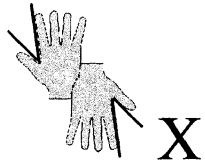
3) позиционные системы.

Простейшая и самая древняя — так называемая *унарная система счисления*. В ней для записи любых чисел используется всего один символ — палочка, узелок, зарубка, камешек. Если какое-нибудь число должны были запомнить два человека, то брали деревянную бирку, делали на ней соответствующее число зарубок, а потом раскалывали бирку пополам. Этим кодом пользуются малыши, показывая на пальцах свой возраст.

Система счисления называется непозиционной, если количественный эквивалент значения каждого символа не зависит от его положения (места, позиции) в коде числа.

П р и м е р 1. Примером непозиционной системы может служить система счисления, которая применялась более двух с половиной тысяч лет назад в Древнем Риме. В основе римской системы счисления лежали знаки I (один палец) для числа 1, V (раскрытая ладонь) для числа 5, X (две сложенные ладони) для 10, а обозначения C для 100 и M для 1000 — это первые буквы соответствующих латинских слов (centum — сто, mille — тысяча).





Чтобы записать число, римляне разлагали его на сумму тысяч, полутысяч, сотен, полусотен, десятков, пятерок, единиц. Напри-

мер, десятичное число 28 представляется следующим образом: XXVIII = 10 + 10 + 5 + 1 + 1 + 1 (два десятка, пятерка, три единицы).

Пример 2. Культура Древней Руси была тесно связана с византийской (греческой) культурой. Поэтому и обозначения чисел (буквами) были похожи на греческие. Для обозначения высших десятичных разрядов в славянском языке использовались следующие названия: 10 тысяч — тьма, 10 тем — легион, 10 легионов — леорд и др. Чтобы обозначить тьмы, буквы, соответствующие числам от 1 до 10, обводились кружком, для обозначения легионов эти же буквы обводились кружком из точек, для обозначения леордов — кружком из лучей и т. д.

Система счисления называется позиционной, если количественный эквивалент (значение) символа зависит от его положения (места, позиции) в записи числа.

В привычной нам системе счисления для записи чисел используются десять различных знаков (цифры 0, 1, 2, 3, 4, 5, 6, 7, 8 и 9). Поэтому ее называют ***десятичной***.

Не только сама цифра, но и ее место, ее позиция имеют определяющее значение: из двух написанных рядом цифр левая выражает единицы, в десять раз большие, чем правая. Поэтому данную систему счисления называют ***позиционной***.

Арифметические действия над десятичными числами производятся с помощью операций, в основе которых лежат таблицы умножения и сложения, а также ***правило переноса:*** если в результате сложения двух цифр получается число, которое больше или равно 10, то оно записывается с помощью нескольких цифр, находящихся на соседних позициях.

Кроме десятичной, известны другие позиционные системы счисления, в том числе двадцатеричная и шестидесятеричная. Остатки последней мы находим в сохранившемся до наших дней обыкновении делить один час на 60 минут, одну минуту — на 60 секунд, полный угол — на 360 градусов. В некоторых областях Украины еще несколько десятков лет назад продавали яблоки, яйца и многое другое на «копы» — кучи по 60 штук.

Следует отметить, что позиционная шестидесятеричная система счисления возникла раньше десятичной. Ее применяли в Древнем Вавилоне. В Китае долгое время пользовались пятеричной системой счисления.

До первой трети XX в. имели элементы двенадцатеричной системы счисления. При этом число 12 (дюжина) даже составляло конкуренцию числу 10 в борьбе за почетный пост основания общеупотребительной системы счисления. Дело в том, что число 12 имеет больше делителей (2, 3, 4, 6), чем 10 (2 и 5). Поэтому в двенадцатеричной системе счисления гораздо удобнее производить расчеты, нежели в десятичной.

Основные достоинства любой позиционной системы счисления — простота выполнения арифметических операций и ограниченное количество символов, необходимых для записи любых чисел.

3.1. Представление чисел в позиционных системах счисления

3.1.1. Базис и основание системы счисления

Наша десятичная система счисления характеризуется тем, что в ней 10 единиц какого-либо разряда образуют единицу следующего, старшего разряда. Другими словами, единицы различных разрядов представляют собой различные степени числа 10.

В десятичном числе $A_{10} = 464 = 4 \cdot 10^2 + 6 \cdot 10^1 + 4 \cdot 10^0$ цифры 4, находящиеся на разных позициях, имеют различные количественные значения — 4 сотни и 4 единицы. При перемещении цифры на соседнюю позицию ее вес (количественный эквивалент) изменяется в 10 раз.

Базисом позиционной системы счисления называется последовательность чисел, каждое из которых определяет количественный эквивалент (вес) символа в зависимости от его места в коде числа.

Базис десятичной системы счисления: $\ldots 10^n, 10^{n-1}, \ldots, 10^1, 10^0, 10^{-1}, \ldots, 10^{-m}, \ldots$

Базис произвольной позиционной системы счисления: $\ldots q^n, q^{n-1}, \ldots, q^1, q^0, q^{-1}, \ldots, q^{-m}, \ldots$

Основанием позиционной системы счисления называется целое число q, которое возводится в степень.

Основание в любой системе изображается как 10, но имеет разное количественное значение. Оно показывает, во сколько раз изменяется количественное значение цифры при перемещении ее на соседнюю позицию. Возможно множество позиционных систем, так как за основание системы счисления можно принять любое число, не меньшее 2.

Наименование системы счисления соответствует ее основанию (десятичная, двоичная, пятеричная и т. д.).

В системе счисления с основанием q (q-ичная система счисления) единицами разрядов служат последовательные степени числа q, иначе говоря, q единиц какого-либо разряда образуют единицу следующего разряда.

Для записи чисел в q-ичной системе счисления требуется q различных знаков (цифр), изображающих числа 0, 1, ..., $q - 1$.

Следовательно, основание позиционной системы счисления равно количеству символов (знаков) в ее алфавите. Запись числа q в q-ичной системе счисления имеет вид 10.

П р и м е р 1. Восьмеричная система счисления.

Основание: $q = 8$.

Алфавит: 0, 1, 2, 3, 4, 5, 6 и 7.

Числа: например, $45023{,}152_8$; $751{,}001_8$.

П р и м е р 2. Пятеричная система счисления.

Основание: $q = 5$.

Алфавит: 0, 1, 2, 3 и 4.

Числа: например, 20304_5; $324{,}03_5$.

П р и м е р 3. Шестнадцатеричная система счисления.

Основание: $q = 16$.

Алфавит: 0, 1, 2, 3, 4, 5, 6, 7, 8, 9, A, B, C, D, E, F.

Здесь только десять цифр из шестнадцати имеют общепринятое обозначение 0—9. Для записи остальных символов алфавита (10, 11, 12, 13, 14 и 15) обычно используются первые пять букв латинского алфавита.

Числа: например, $B5C3{,}1A2_{16}$; $355{,}0FA01_8$.

3.3.2. Развернутая и свернутая формы записи чисел

В позиционной системе счисления любое вещественное число может быть представлено в следующем виде:

$$A_q = \pm(a_{n-1}\cdot q^{n-1} + a_{n-2}\cdot q^{n-2} + \ldots + a_0\cdot q^0 + a_{-1}\cdot q^{-1} + a_{-2}\cdot q^{-2} + \ldots + a_{-m}\cdot q^{-m}), \qquad (1)$$

или $A_q \pm \sum_{i-m}^{n-1} a_i q^i$.

Здесь A — само число; q — основание системы счисления; a_i — цифры, принадлежащие алфавиту данной системы счисления; n — количество целых разрядов числа; m — количество дробных разрядов числа.

Разложение числа по формуле (1) называется ***развернутой формой записи***. Иначе такую форму записи называют *многочленной* или *степенной.*

Пример 1. Десятичное число A_{10} = 5867,91 по формуле (1) представляется следующим образом:

$$A_{10} = 5\cdot 10^3 + 8\cdot 10^2 + 6\cdot 10^1 + 7\cdot 10^0 + 9\cdot 10^{-1} + 1\cdot 10^{-2}.$$

Пример 2. Формула (1) для восьмеричной системы счисления имеет вид:

$$A_8 = \pm(a_{n-1}\cdot 8^{n-1} + a_{n-2}\cdot 8^{n-2} + \ldots + a_0\cdot 8^0 + a_{-1}\cdot 8^{-1} + a_{-2}\cdot 8^{-2} + \ldots + a_{-m}\cdot 8^{-m}),$$

где a_i — цифры 0–7.

Восьмеричное число A_8 = 7064,3 в виде (1) запишется так:

$$A_8 = 7\cdot 8^3 + 0\cdot 8^2 + 6\cdot 8^1 + 4\cdot 8^0 + 3\cdot 8^{-1}.$$

Пример 3. Пятеричное число A_5 = 2430,21 по формуле (1) запишется так:

$$A_5 = 2\cdot 5^3 + 4\cdot 5^2 + 3\cdot 5' + 0\cdot 5^\circ + 2\cdot 5^{-1} + 1\cdot 5^{-2}.$$

Вычислив это выражение, можно получить десятичный эквивалент указанного пятеричного числа: $365{,}44_{10}$.

Пример 4. В шестнадцатеричной системе счисления запись $3AF_{16}$ означает:

$$3AF_{16} = 3\cdot 16^2 + 10\cdot 16^1 + 15\cdot 16^0 = 768 + 160 + 15 = 943_{10}.$$

3.3.3. Двоичная система счисления

Из всех позиционных систем счисления особенно проста и поэтому интересна двоичная система счисления. В ней для записи чисел используются всего две цифры: 0 и 1. Запись 10 означает число 2, так как две единицы данного разряда составляют единицу старшего разряда.

В двоичной системе счисления основание $q = 2$. В этом случае формула (1) принимает следующий вид:

$A_2 = \pm(a_{n-1} \cdot 2^{n-1} + a_{n-2} \cdot 2^{n-2} + \ldots + a_0 \cdot 2^0 + a_{-1} \cdot 2^{-1} + a_{-2} \cdot 2^{-2} + \ldots + {}_{-m} \cdot 2^{-m})$,

где a_i — цифры 0 или 1.

Выпишем начало натурального ряда чисел в десятичной и двоичной системах счисления:

A_{10}	A_2	A_{10}	A_2
0	0	8	1000
1	1	9	1001
2	10	10	1010
3	11	11	1011
4	100	12	1100
5	101	13	1101
6	110	14	1110
7	111	15	1111

Итак, двоичное число представляет собой цепочку из нулей и единиц. При этом оно имеет достаточно большое число разрядов. Быстрый рост числа разрядов — самый существенный недостаток двоичной системы счисления.

3.2. Связь между позиционными системами счисления

Человек привык работать в десятичной системе счисления, а ЭВМ ориентирована на двоичную систему. Поэтому общение человека с машиной было бы невозможно без создания простых и надежных алгоритмов перевода чисел из одной системы счисления в другую.

3.2.1. Перевод целых десятичных чисел в двоичную систему счисления

Первый способ. Пусть $A_ц$ — целое десятичное число. Тогда в его разложении отсутствуют коэффициенты с отрицательными индексами, и его можно представить в виде:

$$A_ц = a_{n-1} \cdot 2^{n-1} + a_{n-2} \cdot 2^{n-2} + \ldots + a_0 \cdot 2^0. \qquad (2)$$

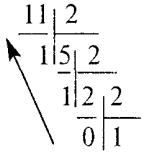
Разделим число $A_ц$ на 2. Частное будет равно

$$a_{n-1} \cdot 2^{n-1} + a_{n-2} \cdot 2^{n-2} + \ldots + a^1,$$

а остаток равен a_0.

Полученное неполное частное опять разделим на 2, остаток от деления будет равен a_1. Если продолжить этот процесс деления, то на n-м шаге получим набор цифр a_0, a_1, a_2, … a_{n-1}, которые входят в двоичное представление числа $A_ц$ и совпадают с остатками при последовательном делении данного числа на 2. Но мы их получили в порядке, обратном порядку расположения в двоичном представлении числа $A_ц$: $A_ц = a_{n-1}\, a_{n-2}\, ...\, a_1\, a_0$.

Пример 1. Перевести десятичное число 11 в двоичную систему счисления. Рассмотренную выше последовательность действий (алгоритм перевода) удобнее изобразить так:



Записывая остатки от деления в направлении, указанном стрелкой, получим: $11_{10} = 1011_2$.

Пример 2. Если десятичное число достаточно большое, то можно порекомендовать следующий способ записи рассмотренного выше алгоритма:

Делимое	363	181	90	45	22	11	5	2	1
Делитель	2	2	2	2	2	2	2	2	2
Частное	1	1	0	1	0	1	1	0	1

$363_{10} = 101101011_2$.

Второй способ. Представим десятичное число 1579 в двоичной системе счисления. Для этого составим таблицу степеней числа 2.

n	0	1	2	3	4	5	6	7	8	9	10
2^n	1	2	4	8	16	32	64	128	256	512	1024

Воспользуемся так называемым *методом разностей*. Берем ближайшую к переводимому числу степень числа 2 и составляем разность: 1579 – 1024 = 555. Затем находим следующую степень числа 2, меньшую 555, и составляем разность: 555 – 512 = 43.

Три очередные степени числа 2 (256, 128, 64) больше остатка 43 и поэтому пропускаются. Последующие разности: 43 – 32 = 11, 11 – 8 = 3, 3 – 2 = 1. В итоге:

$$1579 = 1024 + 512 + 32 + 8 + 2 + 1 = 1\cdot2^{10} + 1\cdot2^9 + 0\cdot2^8 + 0\cdot2^7 + \\ + 0\cdot2^6 + 1\cdot2^5 + 0\cdot2^4 + 1\cdot2^3 + 0\cdot2^2 + 1\cdot2^1 + 1\cdot2^0.$$

Это число в свернутой форме записи будет иметь следующий вид: 11000101011_2.

3.2.2. Перевод правильных дробей

Пусть $A_{др}$ — десятичная дробь, тогда в ее разложении отсутствуют коэффициенты с положительными индексами:

$$A_{др} = a_{-1} \cdot 2^{-1} + a_{-2} \cdot 2^{-2} + \ldots + a_{-m} \cdot 2^{-m}. \quad (3)$$

Таким образом, необходимо найти коэффициенты a_{-1}, a_{-2}, ... , входящие в запись числа в двоичной системе счисления.

Умножим правую и левую части выражения (3) на 2. В результате в правой части получим: $a_{-1} + a_{-2} \cdot 2^{-1} + a_{-3} \cdot 2^{-2} + \ldots$

Целая часть и даст нам старший коэффициент в разложении числа $A_{др}$ по степеням двойки.

Оставшуюся дробную часть умножим на 2: $a_{-2} + a_{-3} \cdot 2^{-1} + \ldots$

Цифра a_{-2} представляет собой второй коэффициент после запятой в двоичном представлении исходного числа.

Описанный процесс необходимо продолжать до тех пор, пока в правой части не получим нуль или пока не будет достигнута необходимая точность вычислений.

Пример 3. Перевести десятичную дробь 0,5625 в двоичную систему счисления. Вычисления лучше всего оформлять по следующей схеме:

0	5625 * 2
1	1250 * 2
0	2500 * 2
0	5000 * 2
1	0000

Пример 4. Перевести в двоичную систему счисления десятичную дробь 0,7.

0,	7 *2
1	4

	*2
0	8
	*2
1	6
	*2
1	2

……..

Очевидно, что этот процесс может продолжаться бесконечно, давая все новые и новые знаки в изображении двоичного эквивалента числа $0{,}7_{10}$. Так, за четыре шага мы получаем число $0{,}1011_2$, а за семь шагов — число $0{,}1011001_2$, которое является более точным представлением числа $0{,}7_{10}$ в двоичной системе счисления, и т. д. Такой бесконечный процесс обрывают на некотором шаге, когда считают, что получена требуемая точность представления числа.

3.2.3. Перевод смешанных чисел

Перевод смешанных чисел, содержащих целую и дробную части, осуществляется в два этапа. Отдельно переводится целая часть, отдельно — дробная. В итоговой записи полученного числа целая часть отделяется от дробной запятой (точкой). См. подразд. 2.3.

Пример 5. Перевести число 17,25 в двоичную систему счисления.

Переводим целую часть:

17	2
1	8 | 2
0	4 | 2
0	2 | 2
0	1

Переводим дробную часть:

0,	25
	*2
0	50
	*2
1	00

Ответ: $17{,}25_{10}=10001{,}01_2$.

3.2.4. Перевод целых чисел из системы счисления с основанием *p* в систему счисления с основанием *q*

По аналогии с рассмотренными алгоритмами можно сформулировать ***алгоритм перевода целых чисел из системы счисления с основанием p в систему счисления с основанием q:***

1) основание новой системы счисления выразить цифрами исходной системы счисления и все последующие действия производить в исходной системе счисления;

2) последовательно выполнять деление данного числа и получаемых целых частных на основание новой системы счисления до тех пор, пока не получим частное, меньшее делителя;

3) полученные остатки, являющиеся цифрами числа в новой системе счисления, привести в соответствие с алфавитом новой системы счисления;

4) составить число в новой системе счисления, записывая его начиная с последнего остатка.

Под исходной подразумевается система счисления с основанием *p,* под новой — с основанием *q.*

Пример 1. Перевести десятичное число 173 в восьмеричную систему счисления:

$$\begin{array}{r|l} 173 & 8 \\ \hline 5 & 21 \mid 8 \\ & \underline{5} \quad 2 \end{array}$$

Ответ: $173_{10} = 255_8$.

Пример 2. Перевести десятичное число 173 в шестнадцатеричную систему счисления:

173	16
13	10
(D)	(A)

Ответ: $173_{10} = AD_{16}$.

3.2.5. Перевод дробных чисел из системы счисления с основанием *p* в систему счисления с основанием *q*

Можно сформулировать следующий алгоритм ***правило перевода правильной дроби из системы счисления с основанием p в систему счисления с основанием q:***

1) основание новой системы счисления выразить цифрами исходной системы счисления и все последующие действия производить в исходной системе счисления;

2) последовательно умножать данное число и получаемые дробные части произведений на основание новой системы до тех пор, пока дробная часть произведения не станет равной нулю или пока не будет достигнута требуемая точность представления числа;

3) полученные целые части произведений, являющиеся цифрами числа в новой системе счисления,

4) привести в соответствие с алфавитом новой системы счисления;

5) составить дробную часть числа в новой системе счисления начиная с целой части первого произведения.

Пример 1. Перевести число $0{,}65625_{10}$ в восьмеричную систему счисления.

0,	65625
	*8
5	25000
	*8
2	00000

Ответ: $0{,}65625_{10} = 0{,}52_8$.

Пример 2. Перевести число $0{,}65625_{10}$ в шестнадцатеричную систему счисления.

0,	65625
	*16
10	50000
(A)	*16
8	00000

Ответ: $0{,}65625_{10} = 0{,}A8_{16}$.

Пример 3. При переводе смешанных чисел отдельно переводятся целые и дробные части. Перевести число $124{,}25_{10}$ в восьмеричную систему.

124	8
4	15
	8
7	1

0,	25
	× 8
2	00

Ответ: $174{,}2_8$

3.2.6. Перевод чисел из двоичной системы счисления в систему счисления с основанием 2^n и обратно

Если основание *n*-ичной системы счисления является степенью числа 2, то перевод чисел из двоичной системы счисления в *n*-ичную и обратно можно проводить по более простым правилам.

Пусть $q = 2^3$ и дано некоторое двоичное число

$$\overline{abcdef}_2 = a*2^5 + b*2^4 + c*2^3 + b*2^2 + e*2^1 + f*2^0 =$$

$$= (a*2^2 + b*2^1 + c*2^0)*2^3 + (b*2^2 + e*2^1 + f*2^0) =$$

$$= A_8*2^3 + B = A*8 + B = \overline{AB}_8,$$

$$A_8 = a*2^2 + b*2^1 + c*2^0 = \overline{abc}_2,$$

$$B_8 = d*2^2 + e*2^1 + f*2^0 = \overline{def}_2.$$

Исходя из приведенных выше рассуждений, можно сформулировать следующий алгоритм перевода целых двоичных чисел в систему с основанием $q = 2^n$:

Для того чтобы целое двоичное число записать в системе счисления с основанием q = 2^n, нужно:

1) данное двоичное число разбить справа налево на группы по *n* цифр в каждой;

2) если в последней левой группе окажется меньше *n* разрядов, то ее надо дополнить слева нулями до нужного числа разрядов;

3) рассмотреть каждую группу как *n*-разрядное двоичное число и записать ее соответствующей цифрой в системе счисления с основанием $q = 2^n$.

Пример 1. Число 101100001000110010_2 заменим равным ему числом восьмеричной системы счисления.

Разбиваем число справа налево на триады (группы по 3 цифры) и под каждой из них записываем соответствующую восьмеричную цифру:

101	100	001	000	110	010
5	4	1	0	6	2

Получаем восьмеричное представление исходного числа: 541062_8.

Пример 2. Число 1000000000111110000111_2 переведем в шестнадцатеричную систему счисления.

Разбиваем число справа налево на *тетрады* (группы по четыре цифры) и под каждой из них записываем соответствующую шестнадцатеричную цифру:

0010	0000	0000	1111	1000	0111
4	0	0	F	8	7

Получаем шестнадцатеричное представление исходного числа: $400F87_{16}$.

Для того чтобы дробное двоичное число записать в системе счисления с основанием $q = 2^n$, нужно:

1) данное двоичное число разбить слева направо на группы по n цифр в каждой;

2) если в последней правой группе окажется меньше n разрядов, то ее надо дополнить справа нулями до нужного числа разрядов;

3) рассмотреть каждую группу как n-разрядное двоичное число и записать ее соответствующей цифрой в системе счисления с основанием $q = 2^n$.

Пример 3. Число $0{,}10110001_2$ заменим равным ему числом восьмеричной системы счисления.

Разбиваем число слева направо на триады и под каждой из них записываем соответствующую восьмеричную цифру:

000,	101	100	010
0,	5	4	2

Получаем восьмеричное представление исходного числа: $0{,}542_8$.

Пример 4. Число $0{,}100000000011_2$ переведем в шестнадцатеричную систему счисления. Разбиваем число слева направо на тетрады и под каждой из них записываем соответствующую шестнадцатеричную цифру:

0,	1000	0000	0011
0,	8	0	3

Получаем шестнадцатеричное представление исходного числа: $0{,}803_{16}$.

Для того чтобы произвольное двоичное число записать в системе счисления с основанием $q = 2^n$, нужно:

1) целую часть данного двоичного числа разбить справа налево, а дробную - слева направо на группы по *n* цифр в каждой;

2) если в последней левой и/или правой группе окажется меньше *n* разрядов, то их надо дополнить слева и/или справа нулями до нужного числа разрядов;

3) рассмотреть каждую группу как n-разрядное двоичное число и записать ее соответствующей цифрой в системе счисления с основанием $q = 2^n$.

Пример 5. Число $111100101{,}0111_2$ заменим равным ему числом восьмеричной системы счисления.

Разбиваем целую и дробную части числа на триады и под каждой из них записываем соответствующую восьмеричную цифру:

111	100	101,	011	100
7	4	5,	3	4

Получаем восьмеричное представление исходного числа: $745{,}34_8$.

Пример 6. Число $1001000{,}1101001_2$ переведем в шестнадцатеричную систему счисления. Разбиваем целую и дробную части числа на тетрады и под каждой из них записываем соответствующую шестнадцатеричную цифру:

0100	1000,	1101	0010
4	8,	D	2

Получаем шестнадцатеричное представление исходного числа: $48{,}D2_{16}$.

Для того чтобы произвольное число, записанное в системе счисления с основанием $q = 2^n$, перевести в двоичную систему счисления, нужно каждую цифру этого числа заменить ее *n*-значным эквивалентом в двоичной системе счисления.

Пример 7. Переведем шестнадцатеричное число 4AC35 в двоичную систему счисления. В соответствии с алгоритмом:

4 | A | C | 3 | 5

0100	1010	1100	0011	0101

Получаем двоичное представление исходного числа:

1001010110000110101_2.

П р и м е р 8. Переведем восьмеричное число 1024_8 в двоичную систему счисления.

1	0	2	4
001	000	010	100

Получаем двоичное представление исходного числа:

1000010100_2.

Описанные алгоритмы позволяют достаточно быстро и просто осуществлять переводы десятичных чисел в двоичную систему счисления и обратно с использованием в качестве промежуточной восьмеричной или шестнадцатеричной системы счисления.

3.2.7. Перевод чисел из произвольной позиционной системы счисления в десятичную

Существует несколько способов перевода чисел из позиционной системы счисления с произвольным основанием в десятичную.

Перевод по правилам (описан выше). Такой способ перевода достаточно затруднителен для систем с основанием $q > 2$, так как требует хорошего знания таблиц сложения и умножения в соответствующих системах счисления.

П р и м е р 1. Перевести двоичное число 10101101,1 в десятичную систему счисления.

Основание десятичной системы в двоичной имеет вид 1010.

```
 10101101 | 1010                 0, | 1000
-1010     | 10001 | 1010        ×    | 1010
     1101  -1010  | 1          101   | 0000
    -1010    111               (5)
       11    (7)
      (3)
```

Ответ: $173{,}5_{10}$

Перевод по степенному ряду (формула (1)).

П р и м е р 2. Перевести двоичное число 1010110,11 в десятичную систему счисления.

$$A_2 = 1*2^6 + 1*2^4 + 1*2^2 + 1*2^1 + 1*2^{-1} + 1*2^{-2} = 86{,}75_{10}.$$

3.3. Двоичная арифметика

Рассмотрим более подробно арифметические операции в двоичной системе счисления.

Арифметика двоичной системы счисления основывается на использовании следующих таблиц сложения, вычитания и умножения цифр.

+	0	1
0	0	1
1	1	10

–	0	1
0	0	$\bar{1}$ 1
1	1	0

*	0	1
0	0	0
1	0	1

Рассмотрим подробно каждую операцию.

Сложение. Таблица двоичного сложения предельно проста: $1_2 + 1_2 = 10_2$ остается в данном разряде, а 1 переносится в следующий разряд.

П р и м е р ы:

$+1001_2$	$+1101_2$	$+11111_2$	$+1010011,111_2$
1010_2	1011_2	1	$11001,110_2$
10011_2	11000_2	100000_2	$1101101,101_2$

Вычитание. *Первый способ.* При выполнении операции вычитания всегда из большего по абсолютной величине числа вычитается меньшее и ставится соответствующий знак. В таблице вычитания 1 с чертой означает заем в старшем разряде.

Примеры:

$$10111001{,}1_2 - 10001101{,}1_2 = 101100{,}0_2;$$

$$101011111_2 - 110101101_2 = -1010110_2;$$

$-1011100 1,1_2$	$-1101101 01_2$
$1000110 1,1_2$	$1010111 11_2$
$1010110 0,0_2$	$0010101 10_2$

Второй способ. Вычитание чисел на компьютере сводится к сложению уменьшаемого и дополнительного кода вычитаемого с последующим отбрасыванием старшего разряда.

Например: $1101101_2 - 110111_2$ ($109_{10} - 55_{10}$).

1. Количество цифр вычитаемого должно совпадать с количеством цифр уменьшаемого; для этого, при необходимости, добавляем слева нужное число нулей (в нашем примере вычитаемое содержит 6 цифр, а уменьшаемое – 7, поэтому добавляем один нуль слева) и получаем 0110111_2.

2. Находим обратный код (инверсию) вычитаемого:

0110111 – вычитаемое

1001000 – инверсия вычитаемого;

3. Находим дополнительный код вычитаемого прибавлением 1 к обратному коду вычитаемого:

$$\begin{array}{rr} & 1001000 \\ + & \underline{\quad\quad 1} \\ & 1001001 \end{array}$$

4. Складываем дополнительный код вычитаемого и уменьшаемое

$$\begin{array}{rr} & 1101101 \\ + & 1001001 \\ \hline & 10110110 \end{array}$$

5. Отбрасываем старшую единицу.

Ответ: 110110_2. Проверка: $109_{10} - 55_{10} = 54_{10}$.

Умножение. Операция умножения выполняется с использованием таблицы умножения по обычной схеме, применяемой в десятичной системе счисления с последовательным умножением множимого на очередную цифру множителя.

Пример.

$11001_2 * 1101_2 = 101000101_2$;

$1100{,}01_2 * 11{,}01_2 = 1010010{,}001_2$;

$$\begin{array}{r} 11001_2 \\ \times\ 1101_2 \\ \hline 11001 \\ 11001 \\ 11001 \\ \hline 101000101_2 \end{array} ; \qquad \begin{array}{r} 11001{,}01_2 \\ \times\ 11{,}01_2 \\ \hline 1100101 \\ 1100101 \\ 1100101 \\ \hline 1010010{,}0001_2 \end{array}$$

Вы видите, что умножение сводится к сдвигам множимого и сложениям.

Деление. Операция деления выполняется по алгоритму, подобному алгоритму выполнения операции деления в десятичной системе счисления.

П р и м е р.

$1010001012:11012 = 11001_2$.

$$\begin{array}{l|l} 101000101_2 & 1101_2 \\ \underline{-\ 1101} & 11001_2 \\ 1110 & \\ \underline{-\ 1101} & \\ 1101 & \\ \underline{-\ 1101} & \\ 0 & \end{array}$$

3.4. Системы счисления, используемые в ЭВМ

От того, какая система счисления будет использована в ЭВМ, зависят скорость вычислений, емкость памяти, сложность алгоритмов выполнения арифметических операций.

Дело в том, что для физического представления (изображения) чисел необходимы элементы, способные находиться в одном из нескольких устойчивых состояний. Число этих состояний должно быть равно основанию используемой системы счисления. Тогда каждое состояние будет представлять соответствующую цифру из алфавита данной системы счисления.

Десятичная система счисления, привычная для нас, не является наилучшей для использования в ЭВМ. Для изображения любого числа в десятичной системе счисления требуется десять различных символов. При реализации в ЭВМ этой системы счисления необходимы функциональные элементы, имеющие ровно десять устойчивых состояний, каждое из которых ставится в соответствие определенной цифре. Так, в арифмометрах используются вращающиеся шестеренки, для которых фиксируется десять устойчивых положений. Но арифмометр и другие подобные механические устройства имеют серьезный недостаток — низкое быстродействие.

Создание электронных функциональных элементов, имеющих много устойчивых состояний, затруднено. Наиболее простыми с точки зрения технической реализации являются так называемые двухпозиционные элементы, способные находиться в одном из двух устойчивых состояний, например:

• электромагнитное реле замкнуто или разомкнуто;

• ферромагнитная поверхность намагничена или размагничена;

• магнитный сердечник намагничен в некотором направлении или в противоположном ему;

• транзисторный ключ находится в проводящем или запертом состоянии и т. д.

Одно из этих устойчивых состояний может представляться цифрой 0, другое — цифрой 1. С двоичной системой связаны и другие существенные преимущества. Она обеспечивает максимальную помехоустойчивость в процессе передачи информации как между отдельными узлами автоматического устройства, так и на большие расстояния. В ней предельно просто выполняются арифметические действия и возможно применение аппарата булевой алгебры для выполнения логических преобразований информации. Благодаря таким особенностям двоичная система стала стандартом при построении ЭВМ.

Широкое применение в ЭВМ нашли также восьмеричная и шестнадцатеричная системы счисления. Обмен информацией между устройствами большинства ЭВМ осуществляется путем передачи двоичных слов. Пользоваться такими словами из-за их большой длины и зрительной однородности человеку неудобно.

Поэтому специалисты (программисты, инженеры) как на этапах составления несложных программ для микроЭВМ, их отладки, ручного ввода-вывода данных, так и на этапах разработки, создания, настройки вычислительных систем заменяют коды машинных команд, адреса и операнды на эквивалентные им величины в восьмеричной или шестнадцатеричной системе счисления. В результате длина исходного слова сокращается в 3 или 4 раза соответственно. Это делает информацию более удобной для рассмотрения и анализа.

Таким образом, восьмеричная и шестнадцатеричная системы счисления выступают в качестве простейшего языка общения человека с ЭВМ, достаточно близкого как к привычной для человека десятичной системе счисления, так и к двоичному «языку» машины.

Двоично-десятичное кодирование. Как правило, пользователь ЭВМ вводит исходную информацию и получает результат решения задачи в десятичной системе счисления. При вводе информации в ЭВМ каждая десятичная цифра заменяется ее двоичным эквивалентом в виде тетрады (четыре двоичных разряда). Десятичное число требует для своего изображения стольких тетрад, сколько имеется десятичных разрядов в числе.

Таким образом, десятичные цифры представляются в двоичной системе счисления, а все разряды без изменения — в десятичной системе счисления. Это позволяет выполнять арифметические операции в десятичной системе счисления, используя двоичные элементы для хранения и переработки числовой информации. Такая форма представления данных называется ***двоично-десятичной***. Говорят о двоично-десятичном коде (ДДК) или о смешанной двоично-десятичной системе счисления.

П р и м е р. Число 38 в смешанной двоично-десятичной системе будет иметь вид: $0011\ 1000_{2–10}$.

Обратите внимание на то, что приведенная запись не соответствует двоичному представлению десятичного числа 38:

$$38_{10} = 100110_2.$$

При выводе информации из ЭВМ наблюдается обратный процесс: двоичное число переводится в ДДК и затем десятичное число выводится на печать.

4. ПРЕДСТАВЛЕНИЕ И ОБРАБОТКА ИНФОРМАЦИИ В ЭВМ

4.1. Представление текстовой информации

Мониторы современных компьютеров могут работать в двух режимах: текстовом и графическом.

В текстовом режиме экран обычно разбивается на 25 строк по 80 символов в строке. В каждую позицию экрана (знакоместо) может быть помещен один из 256 символов, запрограммированных в знакогенераторе. В текстовом режиме на экран монитора можно выводить символьные тексты, таблицы и простые рисунки, составленные из символов псевдографики.

Для кодирования текстовой информации в вычислительной технике используются равномерные двоичные коды длиной восемь двоичных разрядов. Например, при нажатии клавиши с латинской буквой A в оперативную память передается двоичный код 01000001. При выводе символа на экран дисплея производится декодирование: по двоичному коду символа строится его изображение на экране.

Кодирование и декодирование происходят в компьютере автоматически за миллионные доли секунды. Ощутить эти процессы практически невозможно.

Для сокращения записей вместо двоичных кодов используют шестнадцатеричные. Например, код той же латинской буквы A задается шестнадцатеричным числом 41.

Кодировочные таблицы. Используемые в вычислительной технике кодовые комбинации символов представляют в виде кодировочных (кодовых) таблиц. Кодировочные таблицы имеют 16 строк и 16 столбцов, которые нумеруются шестнадцатеричными цифрами от 0 до F. Место символа в таблице определяет его шестнадцатеричный код. Например, если символ стоит в строке 7 и столбце D, то его код 7D.

Всего кодировочные таблицы содержат 256 различных кодовых комбинаций (таково число различных цепочек из восьми нулей и единиц). Это коды управляющих символов, служебных символов и цифр, латинских и русских (заглавных и строчных) букв, псевдографических символов и математических знаков. Каждую комбинацию можно интерпретировать и как десятичное число от 0 до 255.

На разных типах компьютеров используют разные кодировочные таблицы. В качестве одного из стандартов во всем мире принята таблица ASCII (American Standard Code for Information Interchange), кодирующая ровно половину возможных символов — от 0 до 127. Во второй половине таблицы содержатся коды национальных алфавитов, символы псевдографики и некоторые математические знаки.

DEC HEX	0 00	16 10	32 20	48 30	64 40
0	NUL	DLE		0	@
1	SOH	DC1	!	1	A
2	STX	DC2	“	2	B
3	ETX	DC3	#	3	C
4	EOT	DC4	$	4	D
5	ENQ	S	%	5	E
6	ACK	SYN	&	6	F
7	BEL	ETB	‘	7	G
8	BS	CAN	(	8	H
9	HT	EM	)	9	I
A	LF	SUB	*	:	J
B	VT	ESC	+	;	K
C	FF	FS	,	<	L
D	CR		-	=	M
E	SO	RS	.	>	N
F	SI		/	?	O

80 50	96 60	112 70	128 80	144 90	160 A0	224 E0	240 F0
P	`	p	А	Р	а	р	Ё
Q	a	q	Б	С	б	с	ё

R	b	r	В	Т	в	т	Є
S	c	s	Г	У	г	у	є
T	d	t	Д	Ф	д	ф	Ї
U	e	u	Е	Х	е	х	ї
V	f	v	Ж	Ц	ж	ц	Ў
W	g	w	З	Ч	з	ч	ў
X	h	x	И	Ш	и	ш	°
Y	i	y	Й	Щ	й	щ	•
Z	j	z	К	Ъ	к	ъ	·
[	k	{	Л	Ы	л	ы	_
\	l	|	М	Ь	м	ь	№
]	m	}	Н	Э	н	э	¤
^	n	~	О	Ю	о	ю	_
_	o	DEL	П	Я	п	я	_

4.2. Представление чисел

Числа в компьютере могут быть представлены двумя способами. Если, к примеру, число 7 используется в тексте, то его код $37_{16} = 00110111_2$ получается с помощью кодовой таблицы. Если число 7 используется для вычислений, то по известным вам правилам оно переводится из десятичной системы в двоичную — 0111 — и его код по определенным правилам размещается в памяти компьютера.

Память ЭВМ построена из двоичных запоминающих элементов, обладающих двумя устойчивыми состояниями, одно из которых соответствует нулю, а другое — единице. Таким физическим элементом (битом) представляется в памяти ЭВМ каждый разряд двоичного числа. Совокупность определенного количества этих элементов служит для представления многоразрядных двоичных чисел и составляет ***разрядную сетку ЭВМ***.

Каждая группа из восьми запоминающих элементов (байт) пронумерована. Номер байта называется его ***адресом***. Определенное число последовательно расположенных байтов называется ***словом***.

Человек осуществляет арифметические операции над числами последовательно — цифра за цифрой; ЭВМ производит операции над числами параллельно, сразу в некотором количестве разрядов. Для разных ЭВМ длина слова различна — два, четыре или восемь байт. Разбиение памяти на слова для четырехбайтовых ЭВМ:

Байт 0	Байт 0	Байт 0	Байт 0	Байт 0	Байт 0	Байт 0	Байт 0
Полуслово		Полуслово		Полуслово		Полуслово	
Слово				Слово			
Двойное слово							

4.3. Числа с фиксированной точкой

Различают две формы записи чисел: ***естественную*** и ***экспоненциальную***.

Например, длина некоторого отрезка в зависимости от единиц измерения может быть представлена в естественной форме записи следующим образом: 478 000 микрон; 0,478 м.

При представлении в ЭВМ чисел в естественной форме устанавливается фиксированная длина разрядной сетки. Запятую (точку) можно зафиксировать в начале, середине или конце разрядной сетки, при этом распределение разрядов между целой и дробной частями остается неизменным для любых чисел.

Вместо термина «фиксированная точка» иногда используется термин «фиксированная запятая». Это означает одно и то же, так как в математике принято целую часть числа от дробной части отделять запятой, а в языках программирования для этой цели используется точка. Работая на компьютере, мы можем вводить числа с фиксированной запятой в любом виде. Они будут также высвечиваться и на экране компьютера, но в памяти компьютера они хранятся и обрабатываются либо с запятой, фиксированной после последнего разряда (целые числа), либо с запятой перед старшим разрядом (правильные дроби).

Любые правильная дробь и целое число в двоичной системе счисления имеют соответственно вид:

$$A_{\text{др}} = \pm \sum_{i=1}^{n} a_{-i} \cdot 2^{-i} \quad \text{и} \quad A_{\text{ц}} = \pm \sum_{i=1}^{n} a_i \cdot 2^{n-i}.$$

Анализ этих формул показывает:

1) минимальное положительное число $A_{\min}$ равно 0,00...1 для дробных и единице для целых чисел; числа, по абсолютной величине меньшие $A_{\min}$ (единицы младшего разряда n-разрядной машинной сетки), называются машинным нулем;

2) максимальное положительное число $A_{\max}$ равно 0,11...1 ($1-2^{-n}$) для дробных чисел (во всех разрядах должны быть записаны единицы) и 11...1 ($2^n - 1$) для целых чисел;

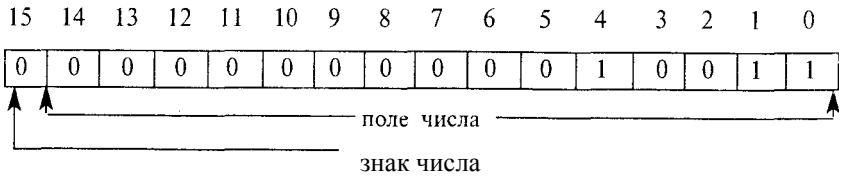
3) общее количество чисел, которое можно записать в n-разрядную сетку, равно $M = 2^n$.

Обычно целые числа занимают в памяти ЭВМ 1, 2 или 4 байта. Один, как правило, старший бит отводится под знак числа. Знак положительного числа «+» кодируется нулем, а знак отрицательного числа «–» — единицей.

Целые числа без знака в двухбайтовом формате могут принимать значения от 0 до $2^{16}-1$ (до 65535), а со знаком «–» от -2^{15} до $+2^{15}-1$, т. е. от –32768 до 32767.

ЭВМ производит операции над числами параллельно, сразу во всех разрядах, поэтому во всех разрядах всегда должно быть что-то записано, даже если это «незначащий» ноль. Число располагается так, что его самый младший двоичный разряд записывается в крайний правый бит разрядной сетки.

Например, десятичное число 19 (10011_2) в 16-разрядной машине записывается так:



4.4. Числа с плавающей точкой

Для очень больших и очень маленьких чисел неудобно и нецелесообразно выписывать много нулей в конце числа или в начале после запятой. В связи с этим и возникла экспоненциальная форма записи. (Эту форму иногда называют *научной,* так как именно в научных задачах часто приходится иметь дело с очень маленькими или очень большими числами.)

Длину отрезка в экспоненциальной форме записи можно представить так: $478 \cdot 10^3$ микрон; $4{,}78 \cdot 10^{-1}$ м.

Точность числа определяется не его длиной, а количеством верных значащих цифр. Задание же всех величин с точностью до 256 бит (или примерно 76 десятичных цифр) — дело не только нереальное, но и бессмысленное, хотя бы потому, что многие из этих величин получаются в результате измерений не очень точными приборами. Не случайно в практических расчетах редко используют более трех значащих цифр, соответствующим образом округляя промежуточные результаты.

Для хранения в памяти ЭВМ чисел с небольшим количеством значащих цифр целесообразно представлять их в экспоненциальной форме. В приведенном выше примере это представление может иметь вид:

$4{,}72 \cdot 10^5$ микрон; $472 \cdot 10^3$ микрон; $4720 \cdot 10^2$ микрон;

$4{,}72 \cdot 10^{-4}$ км; $47{,}2 \cdot 10^{-5}$ км; $472 \cdot 10^{-6}$ км.

Из этого примера видно, что положение запятой в записи числа может изменяться. Поэтому представление в ЭВМ числа в экспоненциальной форме называется ***представлением с плавающей точкой (запятой).*** Кроме того, экспоненциальную форму называют еще *полулогарифмической* или *нормальной.*

Любое число А в экспоненциальной форме представляется в виде

$$A = m_A \cdot q^p,$$

где m_A — мантисса числа; q — основание системы счисления, p — порядок числа.

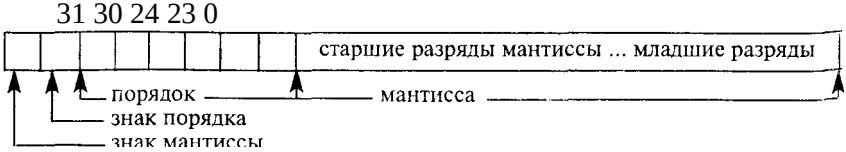
Для однозначности представления чисел с плавающей точкой используется *нормализованная форма,* при которой мантисса отвечает условию:

$$q^{-1} <= |m_A| < 1.$$

Это означает, что мантисса должна быть правильной дробью и иметь после запятой цифру, отличную от нуля.

Правило нормализации мантиссы: если мантисса больше единицы, то она сдвигается вправо и к порядку при каждом сдвиге прибавляется 1. Если после запятой находятся нули, то мантисса сдвигается влево, а из порядка при каждом сдвиге вычитается 1.

Запись числа «нуль» является нормализованной, если и мантисса и порядок равны нулю. Число в форме с плавающей точкой занимает в памяти ЭВМ 4 или 8 байт. При записи числа с плавающей точкой выделяются разряды для хранения знака мантиссы, знака порядка, порядка и мантиссы.



Порядок располагается так, что его самый младший двоичный разряд записывается в крайний правый бит из выделенных под порядок. Мантисса располагается так, что ее самый старший двоичный разряд записывается в крайний левый бит из выделенных под мантиссу.

Оценим диапазон представления чисел по максимальному значению:

$$A_{\max} = m_{\max} * q^{P\max},$$

где $q = 2$; $m_{\max} = 2^6 - 1 = 63.$

Тогда $A_{\max} = (1 - 2^{-24}) * 2^{63} = 1 * 2^{63} = 10^{19}.$

Если для размещения порядка выделяется 7 разрядов, то

$$P_{\max} = 2^7 - 1 = 127 \text{ и } A_{\max} = 10^{38}.$$

с точностью около 7 десятичных разрядов. Когда такой точности не хватает, используется формат удвоенной точности, в котором для записи мантиссы отводится дополнительная область. Это позволяет получить большее число значащих цифр в мантиссе при том же диапазоне порядков.

4.5. Арифметика нормализованных чисел

Пусть $A = m_A * q^{P_A}, B = m_B * q^{P_B}$ — два нормализованных числа.

Для выполнения ***операции сложения (вычитания)*** необходимо выполнить действия по следующему алгоритму:

1) составить разность порядков $Dp = P_A - P_B$;

2) если $Dp > 0$, сдвинуть мантиссу числа B на Dp разрядов вправо; если $Dp < 0$, сдвинуть мантиссу числа A на Dp разрядов вправо; если $Dp = 0$, мантиссы не сдвигаются;

3) выполнить операцию сложения (вычитания) над мантиссами и произвести округление по $n + 1$ разряду;

4) присвоить результату порядок большего разряда;

5) нормализовать при необходимости мантиссу суммы (разности) с одновременным изменением порядка.

Умножение нормализованных чисел производится значительно проще:

1) перемножить мантиссы сомножителей;

2) вычислить порядок произведения $P_C = P_A + P_B$;

3) нормализовать мантиссу произведения с одновременным изменением порядка.

Операция деления реализуется такой последовательностью действий:

1) мантиссу делимого разделить на мантиссу делителя;

2) вычислить порядок частного $P_C = P_A - P_B$;

3) нормализовать мантиссу частного с одновременным изменением порядка.

5. ЭЛЕМЕНТЫ МАТЕМАТИЧЕСКОЙ ЛОГИКИ

5.1. Высказывания и высказывательные формы

Алгебра в широком смысле этого слова — наука об общих операциях, аналогичных сложению и умножению, которые могут выполняться не только над числами, но и над другими математическими объектами.

Примеры алгебр: алгебра натуральных чисел, алгебра рациональных чисел, алгебра многочленов, алгебра векторов, алгебра матриц, алгебра множеств и т. д.

Объектами алгебры логики, или булевой алгебры, являются высказывания.

Высказывание — это любое предложение какого-либо языка (утверждение), содержание которого можно определить как истинное или ложное.

Всякое высказывание или истинно, или ложно; быть одновременно и тем и другим оно не может.

Формулировка любой теоремы является высказыванием. Высказывания могут выражаться с помощью математических, физических, химических и прочих знаков.

Из двух числовых выражений можно составить высказывания, соединив их знаками равенства или неравенства. Сами числовые выражения высказываниями не являются. Например, предложение $X < 12$ становится высказыванием при замене переменной каким-либо конкретным значением. Такие предложения называют ***высказывательными формами***.

Примеры высказываний:

1) {Город Вашингтон — столица США} (истинное высказывание);

2) {Число 2 является делителем числа 7} (ложное высказывание);

3) $\{3 + 5 = 2 \cdot 4\}$ (истинное высказывание);

4) $\{2 + б > 10\}$ (ложное высказывание);

5) {II + VI > VIII} (ложное высказывание);

6) {Сумма чисел 2 и 6 больше числа 8} (ложное высказывание);

7) {Two plus six is eight} (истинное высказывание);

8) {Na — металл} (истинное высказывание).

Высказывание называется ***простым*** (**элементарным**), если никакая его часть сама не является высказыванием. Если это условие не выполняется, высказывание называется ***сложным***.

Высказывания, приведенные выше, являются простыми. Они обозначаются заглавными латинскими буквами:

A = {Аристотель — основоположник логики}.

B = {На яблонях растут бананы}.

Читать приведенные записи нужно так:

А есть высказывание «Аристотель — основоположник логики».
В есть высказывание «На яблонях растут бананы».

Обоснование истинности или ложности простых высказываний решается вне алгебры логики.

Например, истинность или ложность высказывания «Сумма углов треугольника равна 180 градусам» устанавливается геометрией, причем в геометрии Евклида это высказывание является истинным, а в геометрии Лобачевского — ложным. Истинному высказыванию ставится в соответствие 1, ложному — 0.

Таким образом, $A = 1$, $B = 0$.

5.2. Логические операции

Будем считать, что уже имеется некоторый запас элементарных высказываний, относительно каждого из которых известно, истинно оно или ложно. В обычной речи мы часто используем слова, называемые логическими связками, — «не», «и», «или», «следует», «влечет», «эквивалентно», «равносильно», «тогда и только тогда, когда...» и т. п.

Примеры сложных высказываний:

1) {В автобусе можно доехать до школы и почитать журнал};
2) {Число 376 четно или двузначно};
3) {Неверно, что Солнце движется вокруг Земли};
4) {Если сумма цифр числа делится на 3, то число делится на 3}.

В алгебре логики, как и в обычной алгебре, вводится ряд операций. Рассмотрим пять основных логических операций.

1. Логическая операция ***конъюнкция*** (лат. conjunctio — «связываю»):

• в естественном языке соответствует союзам **и, а, но, хотя**;
• обозначение: & или $\wedge$;
• иное название: ***логическое умножение.***

Конъюнкция — это логическая операция, ставящая в соответствие каждым двум элементарным высказываниям новое высказывание, являющееся истинным тогда и только тогда, когда оба исходных высказывания истинны.

Это определение распространяется и на случай *n* высказываний (*n* > 2, *n* —целое число). В соответствии с определением правила выполнения действий для операции конъюнкции можно представить в виде истинностной таблицы:

A	B	A∧B
0	0	0
0	1	0
1	0	0
1	1	1

Истина будет лишь в том случае, когда оба человека не лгут.

2. Логическая операция ***дизъюнкция*** (лат. disjunctio — «различаю»):

• в естественном языке соответствует союзу **или**;

• обозначение;

• иное название: ***логическое сложение***.

Дизъюнкция — это логическая операция, которая каждым двум элементарным высказываниям ставит в соответствие новое высказывание, являющееся истинным тогда и только тогда, когда хотя бы одно из двух образующих его высказываний истинно.

Правила действия для операции дизъюнкции можно представить в виде истинностной таблицы:

A	B	A∨B
0	0	0
0	1	1
1	0	1
1	1	1

Выбирая между истиной и ложью, мы останавливаемся на истине.

В отличие от рассмотренной выше операции дизъюнкции можно рассмотреть ***строгую дизъюнкцию*** (двойное «или»), которой в естественном языке соответствует связка «либо..., либо...»). Суть этой операции ясна из приведенной ниже таблицы:

A	B	A⊕B
0	0	0
0	1	1
1	0	1
1	1	0

Данная операция реализует сложение разряда двоичного числа без переноса в старший разряд.

3. Логическая операция ***импликация*** (лат. implicatio — «тесно связываю»):

• в естественном языке соответствует обороту **если...,** то...;

• обозначение: =>;

• иное название: ***логическое следование.***

Импликация — это логическая операция, ставящая в соответствие каждым двум элементарным высказываниям новое высказывание, являющееся ложным тогда и только тогда, когда условие (первое высказывание) истинно, а следствие (второе высказывание) ложно. Таблица истинности импликации:

A	B	A⇒B
0	0	1
0	1	1
1	0	0
1	1	1

Из лжи может следовать что угодно, даже истина, но из истины не может следовать ложь.

4. Логическая операция э***квиваленция*** (лат. aequivalens — «равноценное»):

• в естественном языке соответствует оборотам речи **тогда и только тогда, в том и только в том случае;**

• обозначение:⇔;

• иное название: ***равнозначность.***

Эквиваленция — это логическая операция, ставящая в соответствие каждым двум элементарным высказываниям новое высказывание, являющееся истинным тогда и только тогда, когда оба исходных высказывания одновременно истинны или одновременно ложны. Эквивалентны ли высказывания, то есть одинаковы ли значения высказываний?

Таблица истинности эквиваленции:

A	B	A⇔B
0	0	1
0	1	0
1	0	0
1	1	1

5. Логическая операция ***инверсия*** (лат. inversio — «переворачиваю»):

• в естественном языке соответствует словам **неверно, что...** и частице **не**;

• обозначение: $\overline{A}$;

• иное название: ***отрицание***.

Отрицание **—** это логическая операция, которая каждому данному высказыванию ставит в соответствие новое высказывание, которое истинно, если данное высказывание ложно, и ложно, если данное высказывание истинно.

Таблица истинности инверсии:

A	$\overline{A}$
0	1
1	0

Логические операции имеют следующий приоритет: действия в скобках, отрицание, ∧, v, =>,<=>.

5.2. Таблицы истинности

Логические функции могут быть заданы табличным способом или аналитически — в виде соответствующих формул.

Истинность или ложность сложных высказываний, образованных в результате выполнения логических операций над простыми высказываниями, не зависит от смыслового содержания исходных высказываний и определяется только их значениями (истинностью или ложностью).

Поэтому любое сложное высказывание можно рассматривать как некоторую логическую функцию *F(X*1*, X*2*, ..., Xn).*

Определим количество различных логических функций с заданным числом переменных *n.* Логическая функция на каждом наборе переменных принимает значение 0 или 1.

Следовательно, отличающихся друг от друга функций может быть ровно столько, сколько существует различных комбинаций из $m = 2^n$ нулей и единиц.

Таких комбинаций 2^n, и они представляют собой последовательность *n*-разрядных двоичных чисел от 0 до $2^n - 1$.

5.3.1. Логические функции от двух переменных

Пусть $n = 2$. Существует 16 различных логических функций от двух переменных. Рассмотрим их подробно:

Аргументы		**Функции**															
X_1	X_2	F_1	F_2	F_3	F_4	F_5	F_6	F_7	F_8	F_9	F_{10}	F_{11}	F_{12}	F_{13}	F_{14}	F_{15}	F_{16}
0	0	0	0	0	0	0	0	0	0	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
1	0	0	0	1	1	0	0	1	1	0	0	1	1	0	0	1	1
1	1	0	1	0	1	0	1	0	1	0	1	0	1	0	1	0	1

F_1 — константа 0.

F_2 — конъюнкция.

F_3 — отрицательные импликации X_1 и X_2.

F_4 — функция, повторяющая переменная X_1.

F_5 — отрицание импликации X_2 и X_1.

F_6 — переменная X_2.

F_7 — строгая дизъюнкция или отрицание эквивалентности (неравнозначность) переменных X_1 и X_2. Значение этой функции получается поразрядным сложением двоичных переменных X_1 и X_2 по модулю 2, то есть без учета переноса в старший разряд.

F_8 — дизъюнкция.

F_9 — отрицание дизъюнкции (функция ИЛИ-НЕ); эта функция называется также ***функцией Пирса*** («стрелка» Пирс).

F_{10} — эквивалентность.

F_{11} — отрицание переменной X_2.

F_{12} — импликация X_1 и X_2.

F_{13} — отрицание X_1.

F_{14} —импликация X_1 и X_2.

F_{15} — отрицание конъюнкции (функция И-НЕ); эта функция называется также ***функцией Шеффера*** («штрих» Шеффера).

F_{16} — константа 1.

С увеличением числа аргументов количество логических функций резко возрастает. Так, при $n = 3$ их будет уже 256. Но изучать их все нет никакой необходимости. Дело в том, что функция любого количества переменных может быть выражена через функции только двух переменных. Делается это с помощью приема суперпозиции, состоящего в том, что, во-первых, на место

переменных подставляются функции, во-вторых, переменные меняются местами.

Минимальное количество функций двух переменных, через которое можно выразить все другие логические функции, называется ***функционально полным набором логических функций***.

Вот несколько примеров функционально полных наборов:

1) F2 и F11; 2) F13 и F8; 3) F9 и F15.

При желании всю алгебру логики можно свести к одной функции. Но чаще всего логические функции записываются в виде логического выражения через инверсию, конъюнкцию и дизъюнкцию.

Введенные пять логических операций дают возможность из простых высказываний строить сложные. Всякое сложное высказывание принимает значение 1 или 0 в зависимости от значения простых высказываний, из которых оно построено.

Таблицу, показывающую, какие значения принимает сложное высказывание при всех сочетаниях (наборах) значений входящих в него простых высказываний, называют ***таблицей истинности*** сложного высказывания. Сложные высказывания часто называют ***формулами логики высказываний***. Для любой формулы алгебры логики достаточно просто построить таблицу истинности.

5.3.2. Алгоритм построения таблицы истинности

1) Подсчитать n — количество переменных в формуле.

2) Определить число строк в таблице $m = 2^n$.

3) Подсчитать количество логических операций в формуле.

4) Установить последовательность выполнения логических операций с учетом скобок и приоритетов.

5) Определить количество столбцов в таблице: число переменных плюс число операций.

6) Выписать наборы входных переменных с учетом того, что они представляют собой натуральный ряд n-разрядных двоичных чисел от 0 до 2^n—1.

7) Провести заполнение таблицы истинности по столбцам, выполняя логические операции в соответствии с установленной в п. 4 последовательностью.

Пример. Для формулы $A \wedge (B \vee B \wedge C)$ построить таблицу истинности.

A	B	C	$\overline{B}$	$\overline{C}$	$B \wedge C$	$B \wedge B \wedge C$	$A \wedge (B \vee B \wedge C)$
0	0	0	1	1	1	1	0
0	0	1	1	0	0	0	0
0	1	0	0	1	0	1	0
0	1	1	0	0	0	1	0
1	0	0	1	1	1	1	1
1	0	1	1	0	0	0	0
1	1	0	0	1	0	1	1
1	1	1	0	0	0	1	1

Наборы входных переменных во избежание ошибок иногда рекомендуют перечислять следующим образом:

1) определить количество наборов входных переменных;

2) разделить колонку значений первой переменной пополам и заполнить верхнюю часть колонки нулями, а нижнюю — единицами;

3) разделить колонку значений второй переменной на четыре части и заполнить каждую четверть чередующимися группами нулей или единиц, начиная с группы нулей;

4) продолжать деление колонок значений последующих переменных на 8, 16 и т. д. частей и заполнение их группами нулей или единицами до тех пор, пока группы нулей и единиц не будут состоять из одного символа.

Процедура составления таблиц истинности может быть существенно сокращена, если воспользоваться следующим приемом.

Пример. Для $(X_1 \Rightarrow X_2) \Leftrightarrow (\overline{X}_1 \,\&\, X_2)$ получаем:

$$(X_1 \Rightarrow X_2) \quad \Rightarrow \quad (\overline{X}_1 \,\&\, X_2)$$

0	1	0	0	1	0	0
0	1	1	1	1	1	1
1	0	0	1	0	0	0
1	1	1	0	0	0	1

5.4. Законы логики высказываний

Сложные высказывания (формулы) A и B называются ***равносильными***, если их истинностные значения совпадают на любых наборах истинностных значений простейших высказываний, вхо-

дящих в эти формулы. В алгебре логики имеется ряд законов, позволяющих производить равносильные преобразования формул.

5.4.1. Формулировки логических законов

1. Закон двойного отрицания:

$$A = \overline{\overline{A}}.$$

Двойное отрицание исключает отрицание.

2. Переместительный (коммутативный) закон:

• для логического сложения:

$$A \vee B = B \vee A;$$

• для логического умножения:

$$A \& B = B \& A.$$

Результат операции над высказываниями не зависит от того, в каком порядке берутся эти высказывания. В обычной алгебре $a + b = b + a$, $a \cdot b = b \cdot a$.

3. Сочетательный (ассоциативный) закон:

• для логического сложения:

$$(A \vee B) \vee C = A \vee (B \vee C;$$

• для логического умножения:

$$(A \& B) \& C = A \& (B \& C).$$

При одинаковых знаках скобки можно ставить произвольно или вообще опускать. В обычной алгебре

$$(A + B) + C = A + (B + C) = A + B + C,$$
$$A \cdot (B \cdot C) = A \cdot (B \cdot C) = A \cdot B \cdot C.$$

4. Распределительный (дистрибутивный) закон:

• для логического сложения:

$$(A \vee B) \& C = (A \& C) \vee (B \& C);$$

• для логического умножения:

$$(A \& B) \vee C = (A \vee C) \& (B \vee C).$$

Закон определяет правило выноса общего высказывания за скобку.

В обычной алгебре $(a + b) * c = a + b * c$.

5. *Закон общей инверсии (законы де Моргана):*

• для логического сложения:

$$\overline{A \vee B} = \overline{A} \& \overline{B};$$

• для логического умножения:

$$\overline{A \& B} = \overline{A} \vee \overline{B}.$$

6. Закон идемпотентности (от латинских слов idem — «тот же самый» и potens — «сильный»; дословно — «равносильный»):

• для логического сложения:

$$A \vee A = A;$$

• для логического умножения:

$$A \& A = A.$$

Закон означает отсутствие показателей степени

7. Законы исключения констант:

• для логического сложения:

$$A \vee 1 = 1, A \vee 0 = A;$$

• для логического умножения:

$$A \& 1 = A, A \& 0 = 0.$$

8. Закон противоречия:

$$A \& \overline{A} = 0.$$

Невозможно, чтобы противоречивые высказывания были одновременно истинными.

9. Закон исключения третьего:

$$A \vee \overline{A} = 1.$$

Из двух противоречивых высказываний об одном и том же предмете одно всегда истинно, а второе — ложно, третьего не дано.

10. Закон поглощения:

• для логического сложения:

$$A \vee (A \& B) = A;$$

• для логического умножения:

$$A \& (A \vee B) = A.$$

11. Закон исключения (склеивания):

• для логического сложения:

$$(A \& B) \vee (\overline{A} \& B) = B;$$

• для логического умножения:

$$(A \vee B) \& (\overline{A} \vee B) = B.$$

12. Закон контрапозиции (правило перевертывания):

$$(A \Rightarrow B) = (\bar{B} \Rightarrow \bar{A}).$$

Переместительный, сочетательный (для логических сложения и умножения) и распределительный (для логического сложения) законы имеют полную аналогию с обычной алгеброй. Для других законов такой аналогии нет.

5.4.2. Доказательство логических законов

Справедливость приведенных законов можно доказать табличным способом: надо выписать все наборы значений *A* и *B,* вычислить для них значения левой и правой частей доказываемого выражения и убедиться, что результирующие столбцы совпадут.

Пример. Докажем справедливость закона инверсии для логического сложения:

$$\overline{A \vee B} = \bar{A} \& \bar{B}$$

A	B	A∨B	$\overline{A \vee B}$	$\bar{A}$	$\bar{B}$	$\bar{A} \& \bar{B}$
0	0	0	1	1	1	1
0	1	1	0	1	0	0
1	0	1	0	0	1	0
1	1	1	0	0	0	0

5.5. Совершенная дизъюнктивная нормальная форма (СДНФ) и совершенная конъюнктивная нормальная форма (СКНФ)

Если логическая функция представлена дизъюнкцией, конъюнкцией и инверсией, то такая форма представления называется ***нормальной***.

Элементарная конъюнкция — конъюнкция конечного множества логических переменных и их инверсий.

Элементарная дизъюнкция — дизъюнкция конечного множества логических переменных и их инверсий.

Число аргументов, образующих элементарную дизъюнкцию или конъюнкцию, называется ***ее рангом.***

Пример 1. *X & Y & Z, X &. Y & Z* — элементарные конъюнкции третьего ранга. *X* v *Y, X* v *Y* — элементарные дизъюнкции второго ранга.

Дизъюнктивная нормальная форма (ДНФ) содержит элементарные конъюнкции, связанные между собой операцией дизъюнкции.

Конъюнктивная нормальная форма (КНФ) содержит элементарные дизъюнкции, связанные между собой операцией конъюнкции.

Одну и ту же логическую функцию можно представить разными ДНФ и КНФ.

Пример 2. Нетрудно убедиться (построив таблицы истинности для каждой из логических формул или проведя преобразования на основании логических законов), что приведенные ниже формулы определяют одну и ту же логическую функцию *F(X, Y, Z)*:

1) $(X \,\&\, Y) \vee (\bar{X} \,\&\, Y) \vee (X \,\&\, Y \,\&\, Z)$;

2) $(X \,\&\, Y) \vee \quad (\bar{X} \,\&\, Y) \vee (X \,\&\, Z)$.

Для исключения неоднозначности записи логические функции могут быть представлены в совершенных дизъюнктивной и конъюнктивной нормальных формах.

Совершенная дизъюнктивная нормальная форма (СДНФ) отвечает следующим требованиям:

1) в ней нет двух одинаковых элементарных конъюнкций;

2) ни одна элементарная конъюнкция не содержит двух одинаковых переменных;

3) ни одна элементарная конъюнкция не содержит переменную вместе с ее инверсией;

4) все конъюнкции имеют один и тот же ранг.

Аналогичным требованиям подчиняется и ***совершенная конъюнктивная нормальная форма*** (СКНФ).

Пример 3. Если логическая функция содержит конъюнкции разных рангов, то для получения СДНФ следует повысить ранг младших конъюнкций, используя закон исключения третьего.

$$F(X,Y,Z) = (\bar{X} \,\&\, Y) \vee (X \,\&\, Y \,\&\, Z) = (\bar{X} \,\&\, Y) \,\&\, (Z \vee \bar{Z}) \vee$$

$$(X \,\&\, Y \,\&\, Z) = (\bar{X} \,\&\, Y \,\&\, Z) \vee (\bar{X} \,\&\, Y \,\&\, \bar{Z}) \vee (X \,\&\, Y \,\&\, Z).$$

СДНФ и СКНФ можно получить по табличному представлению логической функции.

5.5.1. Алгоритм образования СДНФ по таблице истинности

1. Выделить в таблице истинности все наборы переменных, на которых функция принимает единичные значения.

2. Для каждого выбранного набора записать элементарные конъюнкции, содержащие без инверсии переменные, принимающие в соответствующем наборе значение 1 и с инверсией — переменные, принимающие значение 0.

3. Соединить элементарные конъюнкции знаком дизъюнкции.

5.5.2. Алгоритм образования СКНФ по таблице истинности

1. Выделить в таблице истинности все наборы переменных, на которых функция принимает нулевые значения.

2. Для каждого выбранного набора записать элементарные дизъюнкции, содержащие без инверсии переменные, принимающие в соответствующем наборе значение 0 и с инверсией — переменные, принимающие значение 1.

3. Соединить элементарные дизъюнкции знаком конъюнкции.

Пример 4. Пусть логическая функция F задана таблицей истинности:

X	Y	Z	F	СДНФ	СКНФ
0	0	0	0	–	$X \vee Y \vee Z$
0	0	1	0	–	$X \vee Y \vee \bar{Z}$
0	1	0	1	–	$X \vee \bar{Y} \vee Z$
0	1	1	1	$\bar{X} \& Y \& Z$	–
1	0	0	0	–	$\bar{X} \vee Y \vee Z$
1	0	1	0	$X \& \bar{Y} \& Z$	–
1	1	0	0	$X \& Y \& \bar{Z}$	–
1	1	1	1	$X \& Y \& Z$	–

В соответствии с приведенными выше алгоритмами логическую функцию *F(X, Y, Z)*, заданную таблицей истинности, можно представить аналитически:

1) в СДНФ —

$$F(X,Y,Z) = (X \& Y \& Z) \vee (X \& Y \& \bar{Z}) \vee (X \& \bar{Y} \& Z) \vee (\bar{X} \& Y \& Z);$$

2) в СКНФ —

$$F(X,Y,Z) = (X \vee Y \vee Z) \& (X \vee Y \vee \bar{Z}) \& (X \vee \bar{Y} \vee Z) \& (\bar{X} \vee Y \vee Z).$$

Обратите внимание на тот факт, что СДНФ и СКНФ являются инверсными по отношению друг к другу, т. е. если одна из них в некотором наборе равна 1, то другая на этом же наборе равна 0.

Пример 5. Покажем, как для логической функции, заданной аналитически, можно построить таблицу истинности по СДНФ.

$$F(X, Y, Z) = (X \& Y \& Z) v (X \& Y \& Z) \vee (X \& Y \& Z).$$

По определению СДНФ только на наборах 011, 010, 111 логическая функция *F (X, Y, Z)* принимает значение 1; во всех остальных случаях — значение 0.

X	Y	Z	F
0	0	0	0
0	0	1	0
0	1	0	1
0	1	1	1
1	0	0	0
1	0	1	0
1	1	0	0
1	1	1	1

5.6. Минимизация логических функций

Используя законы алгебры логики, можно упрощать сложные выражения, определяющие логические функции.

Преобразование логической функции с целью упрощения ее аналитического представления называется ***минимизацией.***

Пример. Пусть некоторая логическая функция представлена в СДНФ.

$$F(X,Y,Z) = \bar{X}\,\bar{Y}\,\bar{Z} \vee X\,\bar{Y}\,\bar{Z} \vee \bar{X}\,Y\,\bar{Z} \vee \bar{X}\,YZ.$$

Элементарные конъюнкции называются ***соседними,*** если они отличаются только одной переменной. Применение к соседним конъюнкциям правила склеивания понижает их ранг на единицу.

Первая и вторая, а также третья и четвертая конъюнкции — соседние. В результате их склеивания получим:

$$F = \bar{X}\,\bar{Y}(\bar{Z} \vee Z) \vee \bar{X}\,\bar{Y}(\bar{Z} \vee Z) = \bar{X}\,\bar{Y} \vee \bar{X}\,Y = \bar{X}(\bar{Y} \vee Y) = \bar{X}.$$

Минимизация функций алгебры логики (ФАЛ) в более общем смысле — это процедура нахождения наиболее простого представления ФАЛ в виде суперпозиции функций, составляющих функционально полную систему, при одновременной оптимизации ее технической реализации по некоторым критериям в условиях ряда ограничений.

Критериями оптимизации могут быть объем оборудования (количество вентилей, корпусов), габариты, вес, энергопотребление, стоимость, быстродействие, надежность.

В качестве ***ограничений*** могут выступать допустимые к использованию системы элементов, число элементов в корпусе, коэффициенты объединения по входу и разветвления по выходу логических элементов, необходимость реализации системы ФАЛ, а также ряд перечисленных выше критериев оптимизации.

Решение задачи минимизации ФАЛ в полном объеме является трудной проблемой хотя бы потому, что ряд критериев оптимизации находятся в противоречивом отношении друг к другу, например, одновременное снижение энергопотребления и повышение быстродействия.

На практике обычно решается задача оптимизации по нескольким или даже одному из критериев. Наиболее часто это делается по минимуму необходимого числа базовых логических элементов И, ИЛИ, НЕ, так как при этом в большинстве случаев удовлетворяются требования получения минимальных габаритов, веса, энергопотребления, стоимости, а также повышения быстродействия и надежности.

Иногда ограничиваются еще более простой задачей представления ФАЛ в дизъюнктивной или конъюнктивной форме, содержащей наименьшее возможное число букв, когда, например, для

дизъюнктивных форм, в выражении присутствует как можно меньше слагаемых, являющихся элементарными произведениями, которые в свою очередь содержат как можно меньше сомножителей. Такую задачу принято называть ***канонической задачей минимизации ФАЛ***.

Существует несколько методов минимизации, например, расчетный метод и табличный метод минимизации ФАЛ, основанный на использовании карт, впервые предложенных Вейтчем и модернизированных Карно.

5.7. Алгебра переключательных схем

Долгое время алгебра логики была известна достаточно «узкому» классу специалистов. Прошло почти 100 лет, прежде чем в 1938 году выдающийся американский математик и инженер Клод Шеннон обнаружил, что алгебра логики применима для описания самых разнообразных процессов. Например, 0 и 1 могут кодировать включенные и выключенные переключатели, высокое и низкое напряжение, годную и бракованную продукцию и т. д.

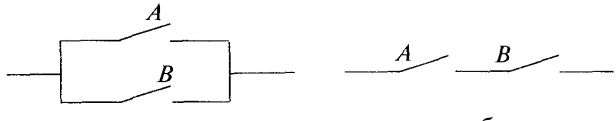
Прежде всего алгебра логики была использована для преобразования релейно-контактных и электронно-ламповых схем. Отвлекаясь от физической природы этих схем, будем называть их переключательными. Другими словами, под ***переключательной схемой*** мы будем понимать схематическое изображение какого-либо устройства, содержащего только двухпозиционные переключатели, т. е. переключатели, которые могут находиться только в двух состояниях: в замкнутом (ток проходит) и в разомкнутом (ток не проходит).

5.7.1. Связь между переключательными схемами и алгеброй высказываний

Любую переключательную схему можно разбить на участки из последовательно или параллельно соединенных переключателей. Каждому переключателю поставим в соответствие элементарное высказывание, истинное тогда, когда переключатель замкнут, и ложное, если переключатель разомкнут. На схемах переключатели будем обозначать теми же буквами, что и соответствующие им элементарные высказывания. Если в цепи

содержится несколько переключателей *A,* то все эти переключатели должны быть одновременно замкнуты или разомкнуты.

Переключателям, соединенным параллельно, поставим в соответствие операцию дизъюнкции: ток в этой цепи (рис. 1, *а*) будет протекать при замкнутом переключателе *A,* или переключателе *B,* или замкнутых переключателях *A* и *B* одновременно.



а *б*

Рис. 1

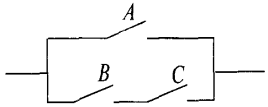
Переключателям, соединенным последовательно, поставим в соответствие операцию конъюнкции: ток в цепи (рис. 1, *б*) потечет только тогда, когда замкнут переключатель *A* и замкнут переключатель *B.*

Если два переключателя работают так, что один из них замкнут, когда другой разомкнут и наоборот, то им ставятся в соответствие высказывания *A* и *A.*

5.7.2. Чтение и анализ переключательных схем

Прочитать переключательную схему — это значит определить, протекает по ней ток или нет при определенных состояниях переключателей.

П р и м е р. Дана схема:



Состояния переключателей задаются таблицей:

A	B	C	Состояние схемы
0	0	1	
1	1	0	
1	1	1	

Требуется прочитать переключательную схему — заполнить колонку «Состояние схемы». Будем последовательно рассматри-

вать все строки таблицы, описывающие состояние переключателей данной схемы.

1. Переключатели *A* и *B* разомкнуты, C замкнут. В этом случае ток не может проходить ни по одной из двух цепочек параллельного соединения. Состояние схемы 0.

2. Так как переключатель *A* замкнут, ток будет проходить по верхней цепочке параллельного соединения. Состояние схемы 1.

3. Ток может проходить по обеим цепочкам параллельного соединения. Состояние схемы 1.

Две схемы, содержащие одни и те же переключатели *A, B,...*, мы будем считать одинаковыми или равными, если при одном и том же состоянии переключателей *(A* замкнут, *B* разомкнут и т. д.) обе схемы одновременно пропускают или не пропускают ток. Естественно считать из двух схем более простой ту, которая содержит меньше переключателей.

В алгебре «переключательных схем» выполняются все законы алгебры логики. В этом достаточно просто убедиться, если прочитать приведенные ниже схемы и сравнить столбец состояния каждой схемы с результирующим столбцом таблицы истинности для соответствующей логической формулы.

Справедливость коммутативного закона для параллельного и последовательного соединения контактов видна на рис. 1, ассоциативного закона — на рис. 2.

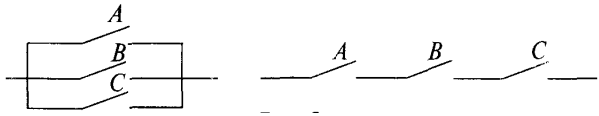


Рис. 2

Равенство схем, приведенных на рис. 3 и 4, говорит о справедливости дистрибутивного закона.

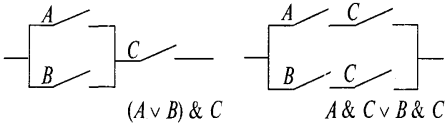


Рис. 3

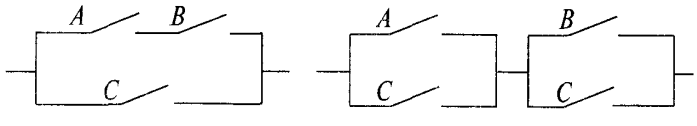


Рис. 4

На рис. 5 приведены переключательные схемы, реализующие следующие соотношения:

а) $1 \vee A = 1$ в) $\bar{A} \vee A = 1$ д) $1 \& A = A$ ж) $A \& \bar{A} = 0$

б) $0 \vee A = A$ г) $A \vee A = A$ е) $0 \& A = 0$ з) $A \& A = A$

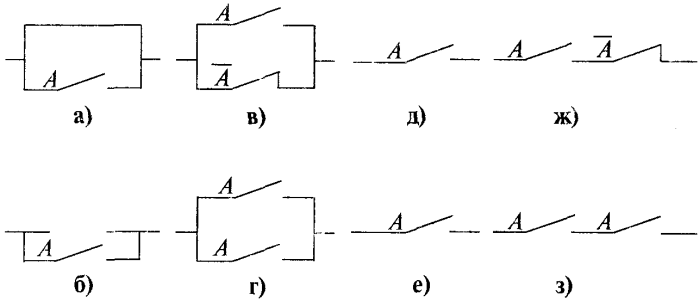
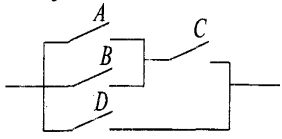


Рис. 5

5.7.3. Упрощение переключательных схем

Каждой переключательной схеме можно поставить в соответствие сложное высказывание, истинное тогда и только тогда, когда схема проводит ток. Это сложное высказывание можно исследовать методами математической логики. Если такое сложное высказывание удается упростить, то и соответствующая схема допускает аналогичное упрощение.



Пример 1. Составим формулу логической функции для следующей схемы. Переключатели *A* и *B* соединены параллельно, что соответствует дизъюнкции соответствующих высказываний: *A* v *B*. Далее следует последовательное соединение с переключателем C: *(A* v *B) & C*. Рассмотренный участок цепи параллельно соединяется с переключателем *D: (A* v *B) & C* v *D.*

Окончательный результат: *F(A,B,C,D) = (A* v *B) &* C v *D.*

Анализ переключательной схемы состоит в определении всех возможных условий, при которых соответствующая электрическая цепь будет пропускать электрический ток. Это удобно делать с помощью соответствующей логической функции.

5.7.4. Алгоритм анализа переключательной схемы

1. Записать логическую формулу, соответствующую данной переключательной схеме.

2. По полученной формуле построить таблицу истинности.

3. Выбрать в таблице истинности наборы (строки), где функция принимает единичные значения. Единицы в рассматриваемых наборах соответствуют замкнутым переключателям, а нули — разомкнутым.

П р и м е р 2. Проанализируем переключательную схему из предыдущего примера.

1. $F(A,B,C,D) = ((A \vee B) \,\&\, C) \vee D.$

2.

((**A**	$\vee$	**B**)	&	**C**)	$\vee$	**D**
0	0	0	0	0	0	0
0	0	**0**	0	**0**	**1**	**1**
0	0	0	0	1	0	0
0	0	**0**	0	**1**	**1**	**1**
0	1	1	0	0	0	0
0	1	**1**	0	**0**	**1**	**1**
0	1	**1**	1	**1**	**1**	**0**
0	1	**1**	1	**1**	**1**	**1**
1	1	0	0	0	0	0
1	1	**0**	0	**0**	**1**	**1**
1	1	**0**	1	**1**	**1**	**0**
1	1	**0**	1	**1**	**1**	**1**
1	1	1	0	0	0	0
1	1	**1**	0	**0**	**1**	**1**
1	1	**1**	1	**1**	**1**	**0**
1	1	**1**	1	**1**	**1**	**1**

3. Электрическая цепь пропускает ток, если состояния переключателей соответствуют наборам: 0001, 0011, 0101, 0110, 0111, 1001, 1010, 1011, 1101, 1110, 1111.

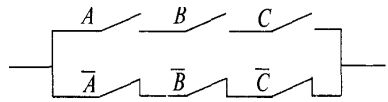
Синтез переключательной схемы — это разработка схемы, условия работы которой заданы таблицей истинности или словесным описанием.

Пример. Построим схему электрической цепи, включающей переключатели *A, B, C,* такую, что цепь замыкается лишь в том случае, если замкнуты все переключатели или ни один из этих переключателей.

Решение. Оформим в виде таблицы все возможные соположения переключателей *A, B,* C и в отдельном столбце отметим требуемое состояние для каждого набора состояний переключателей:

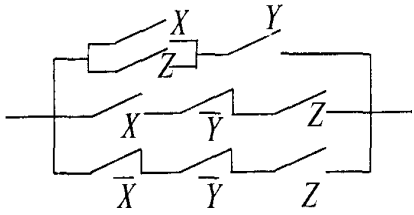
A	**B**	**C**	**F**
0	0	0	1
0	0	1	0
0	1	0	0
0	1	1	0
1	0	0	0
1	0	1	0
1	1	0	0
1	1	1	1

Построим СДНФ: F(A,B,C)=A & B & C V A & B & C. Этой логической функции соответствует переключательная схема:



Упрощение (минимизация) переключательной схемы сводится к упрощению соответствующей ей формулы на основании законов алгебры логики.

Пример. Упростить схему:



Решение. Исходная схема имеет девять переключателей. Выпишем логическую формулу, соответствующую данной схеме и упростим ее.

$$F(X,Y,Z) = ((X \vee Z) \& Y) \vee (X \& \bar{Y} \& Z) \vee (\bar{X} \& \bar{Y} \& Z) =$$

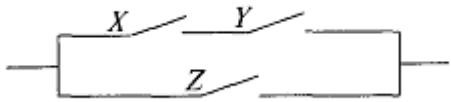
$$= ((X \vee Z) \& Y) \vee (\bar{Y} \& Z) \& (X \vee \bar{X}) = ((X \vee Z) \& Y) \vee$$

$$\vee (\bar{Y} \& Z) \& 1 =$$

$$= ((X \vee Z) \& Y) \vee (\bar{Y} \& Z) = (X \& Y) \vee (Y \& Z) \vee (\bar{Y} \& Z) =$$

$$= (X \& Y) \vee Z \& (Y \vee \bar{Y}) = (X \& Y) \vee Z \& 1 = (X \& Y) \vee Z.$$

Рассматриваемая схема может быть заменена более простой, содержащей только три включателя

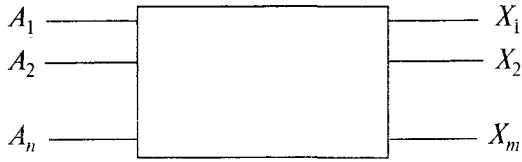


5.8. Использование алгебры логики в вычислительной технике

При разработке вычислительной техники широкое применение находит алгебра логики. Любая информация может быть представлена дискретным сигналом — в виде фиксированного набора отдельных значений. Устройства, в которых действуют такие сигналы, называются дискретными.

Всякий преобразователь информации — некая искусственная система обработки информации, устройство, имеющее входы и выходы. Простейшими преобразователями информации являются

логические преобразователи дискретного действия. Их условное обозначение:



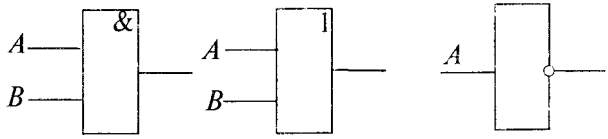
Здесь $A_1, A_2, ..., A_n$ — входы, $X_1, X_2, ..., X_n$ — выходы.

На каждый вход подаются сигналы только двух видов, обозначаемые 0 и 1. Такие же сигналы появляются на каждом выходе.

5.9. Логические элементы

Дискретный преобразователь, который выдает после обработки двоичных сигналов значение одной из логических операций, называется ***логическим элементом***.

Ниже приведены условные обозначения (схемы) логических элементов, реализующих логическое умножение, логическое сложение и отрицание.



Логический элемент **И** *(конъюнктор)* реализует операцию - логического умножения. Единица на выходе этого элемента будет только тогда, когда на всех входах будут единицы. Если хотя бы на одном входе будет нуль, на выходе также будет нуль.

Логический элемент **ИЛИ** *(дизъюнктор)* реализует операцию логического сложения. Если хотя бы на одном входе будет единица, то на выходе элемента также будет единица.

Знак 1 на схеме элемента — остаток устаревшего обозначения операции ИЛИ (результат операции ИЛИ равен единице, если сумма значений операндов больше или равна 1).

Логический элемент **НЕ** *(инвертор)* реализует операцию отрицания. Если на входе элемента нуль, то на выходе единица; если на входе элемента единица, то на выходе нуль.

Описанные логические элементы могут быть реализованы многими способами — на пневматической, гидравлической, механической или электрической основе. Важно лишь, чтобы нули и единицы на входе были идентичны нулям и единицам на выходе.

Любой преобразователь информации, имеющий n входов и m выходов, можно заменить набором из m преобразователей, каждый из которых имеет n входов и один выход.

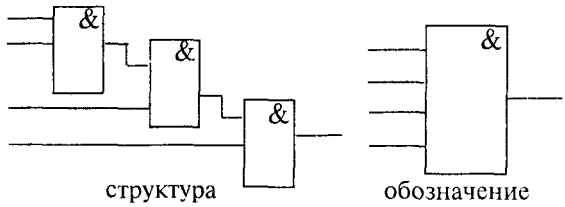
Следовательно, для построения произвольного преобразователя информации достаточно уметь синтезировать преобразователи с одним выходом.

5.10. Комбинационные схемы

Схему преобразователя информации, сигнал на выходе которой однозначно определяется комбинацией сигналов на входах, называют ***комбинационной.***

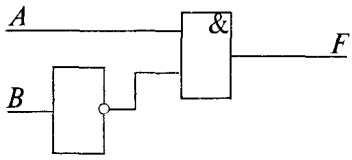
Однотипность сигналов на входах и выходах позволяет подавать сигнал, вырабатываемый одним элементом, на вход другого элемента. Это позволяет из двухвходовых схем **И** и **ИЛИ** строить многовходовые схемы, а также синтезировать произвольные комбинационные схемы, соединяя в цепочки отдельные логические элементы.

Пример 1. Покажем, как из двухвходовых схем И можно построить схему И с любым количеством входов.



Анализируя комбинационную схему, можно понять, как работает логическое устройство.

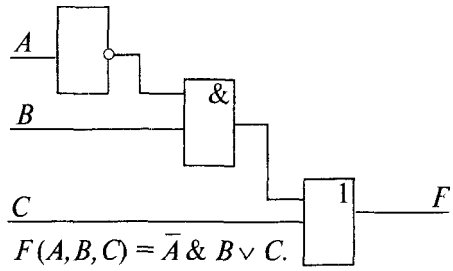
Пример 2. Проанализируем комбинационную схему, т. е. выясним, какой сигнал должен быть на выходе при каждом возможном наборе сигналов на входе.



A	B	F
0	0	0
0	1	0
1	0	1
1	1	0

Заполненная таблица истинности полностью описывает рассматриваемую комбинационную схему. Другая форма описания логических устройств — структурная формула.

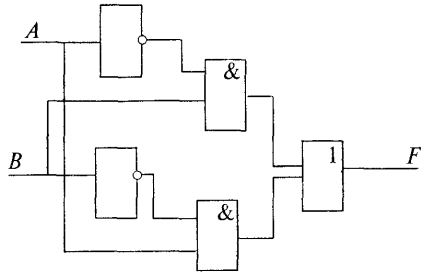
Пример 3. По заданной комбинационной схеме выпишем структурную формулу.



Пример 4. По структурной формуле вычертим соответствующую комбинационную схему:

$$F(A,B) = \bar{B} \& A \vee B \& \bar{A}.$$

Решение. Нам потребуются два инвертора, два конъюнктора и один дизъюнктор.



5.11. Синтез комбинационных схем

Предположим, что имеется таблица истинности некоторой логической функции, где указывается, какой сигнал должен быть на

выходе при каждом возможном наборе сигналов на входах. ***Синтез*** произвольной комбинационной схемы по заданной таблице истинности заключается в получении структурной формулы, описывающей работу схемы, в упрощении этой формулы и в составлении искомой схемы по минимизированной формуле.

Структурные формулы могут быть упрощены на основании законов алгебры логики. Эта работа требует большой аккуратности и внимания, так как даже малейшая неточность приводит к ошибочному результату.

Основным звеном проектирования логической схемы цифрового устройства является синтез

Пример 1. По заданной таблице истинности синтезировать комбинационную схему:

A	**B**	**F**
0	0	1
0	1	1
1	0	1
1	1	0

Решение.

1-й способ: $F(A,B)_{\text{СДНФ}} = \bar{A}\&\bar{B}$

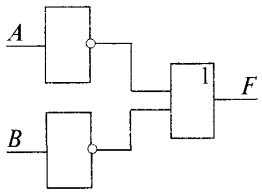
2-й способ:

$$F(A,B)_{\text{СДНФ}} = \bar{A}\&\bar{B} \vee B\&\bar{A} \vee \bar{A}\&B = \bar{A}\&\bar{B} \vee A\&\bar{B} \vee \bar{A}\&B =$$

$$= (\bar{A}\&\bar{B} \vee \bar{A}\&B) \vee (\bar{A}\&\bar{B} \vee A\&\bar{B}) = \bar{A}\&(\bar{B} \vee B) \vee \bar{B}\&(A \vee \bar{A})$$

$$= \bar{A}\&1 \vee = \bar{B}\&1\ \bar{A} \vee \bar{B}$$

Заметим, что запись СКНФ для этого примера сразу же дает нам нужный результат. Комбинационная схема имеет следующий вид.



5.12. Примеры комбинационных схем

5.12.1. Сумматор

Сумматор – комбинационное устройство, обеспечивающее суммирование двоичных чисел. Рассмотрим схему сложения двух *n*-разрядных двоичных кодов.

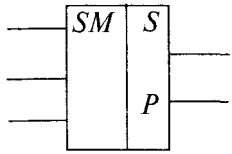
$+ a_u \ldots$	a_i	$\ldots a_1$
		a_0
$b_n \ldots$	b_i	$\ldots b_1\ b_0$
$c_{n+1} c_n \ldots$	c_i	$\ldots c_1\ c_0$

При сложении цифр i-го разряда складываются a_i и b_i. Результатом будет c_i и p_i — перенос в старший разряд.

Таким образом, одноразрядный двоичный сумматор — это устройство с тремя входами и двумя выходами. Его работа может быть описана следующей таблицей истинности:

Вход			**Выход**	
1-е слагаемое	2-е слагаемое	Перенос	Сумма	Перенос
0	0	0	0	0
0	0	1	1	0
0	1	0	1	0
0	1	1	0	1
1	0	0	1	0
1	0	1	0	1
1	1	0	0	1
1	1	1	1	1

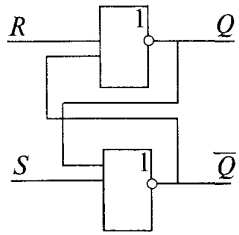
Ниже приведено условное обозначение сумматора:



Если складывать *n*-разрядные двоичные слова, то можно использовать последовательное соединение таких сумматоров.

5.12.2. Триггер

Триггером называют узел, способный хранить один разряд двоичного числа. Самый простой триггер — *RS*. Он состоит из двух элементов ИЛИ—НЕ. Его схема и таблица истинности приведены ниже:



Входы		Состояние Q
R	S	
0	0	Q
0	1	1
1	0	0
1	1	Недопустимо

Обычно на входы поступают сигналы $R = 0$ *и* $S = 0$, и триггер хранит старое состояние. Если на вход *S* поступает на короткое время сигнал 1, то триггер переходит в состояние 1, и после того как сигнал *S* станет равен 0, он будет сохранять это состояние. При подаче 1 на вход R триггер перейдет в состояние 0. Подача на оба входа логической единицы может привести к неоднозначному результату, поэтому такая комбинация входных сигналов запрещена.

Для запоминания 1 байта информации необходимо 8 триггеров, для 1 Кбайта — 8—1024 триггера. Оперативная память современных ЭВМ содержит миллионы триггеров.

В заключение отметим, что ЭВМ состоит из огромного числа отдельных логических элементов, образующих все ее узлы и память.

5.12.3. Дешифратор

Дешифратор — комбинационное устройство, преобразующее комбинацию входных переменных в активный сигнал только на одном из его выходов.

Максимальное количество выходов дешифратора равно 2^n, где n – число входов.

Таблица истинности дешифратора

Входы				Выходы			
A	B	C	D	1	2	3	4
0	0	0	0	1	0	0	0
0	0	0	1	0	1	0	0
0	0	1	0	0	0	1	0

5.12.4. Шифратор

Шифратор *(coder)* — комбинационное устройство, выполняющее функцию, обратную по отношению к дешифратору, т. е. формирование двоичного кода на выходах при появлении сигнала на одном из входов.

Таблица истинности шифратора

Входы							Выходы		
1	2	3	4	5	6	7	1	2	3
0	0	0	0	0	0	0	0	0	0
1	0	0	0	0	0	0	0	0	1
0	1	0	0	0	0	0	0	1	0

СПИСОК РЕКОМЕНДУЕМОЙ ЛИТЕРАТУРЫ

Е.В. Ширшов. Учебный практикум по вычислительной технике : метод. указания и задания к контрольным работам / Е.В. Ширшов О.В. Чурбанова. – Ростов н/Д : Феникс, 2006. – 256 с.

Калиш Г.Г. Основы вычислительной техники : учеб. пособие / Г.Г. Калиш. – М. : Высш шк., 2000. – 271 с.

Кузин А.В. Микропроцессорная техника : учебник для сред. проф. образования / А.В. Кузин, М.А. Жаворонков. – М. : Академия, 2004. – 304 с.