ТЕМА 2
1. СИСТЕМЫ СЧИСЛЕНИЯ.

Содержание

1. Понятие о системах счисления
2. Примеры базисов позиционных систем счисления
3. Перевод десятичных чисел в другие системы счисления
4. Алгоритм перевода целого десятичного числа *N* в позиционную систему с основанием *p*
5. Алгоритм перевода правильной десятичной дроби *N* в позиционную систему с основанием *p*
6. Перевод чисел любых позиционных систем счисления в десятичную систему
7. Выполнение арифметических операций в позиционных системах счисления
8. Представление чисел в памяти компьютера. Представление целых чисел
9. Представление вещественных чисел
10. Контрольные вопросы

1. Понятие о системах счисления

Система счисления – **это способ представления чисел и правила действий над ними.**

Знаки, используемые при записи чисел, называются цифрами.

В качестве знаков, используемых при записи чисел, применяются арабские цифры от 1 до 9 и строчные (заглавные) буквы латинского алфавита: A =10, B = 11, C = 12, D = 13 и так далее.

Полный набор символов определенной системы счисления называется ее алфавитом.

Позиционная система счисления – это та, в которой величина, обозначаемая цифрой в записи числа («вес» цифры), зависит от ее позиции в числе.

Основание системы счисления – количество используемых цифр или знаков в алфавите системы счисления.

Значение числа складывается как сумма цифр числа, умноженных на основание системы в степени, обозначающей номер позиции этой цифры в числе.

Пример: $395{,}65 = 3 \cdot 10^2 + 9 \cdot 10^1 + 5 \cdot 10^0 + 6 \cdot 10^{-1} + 5 \cdot 10^{-2}$

Базис системы счисления **–** последовательность степеней основания**.** Каждое из чисел базиса задает количественное значение, или «вес», соответствующего разряда.

2. Примеры базисов позиционных систем счисления

Десятичная система:…,10^{-2},10^{-1},10^{0},10^{1},10^{2},10^{3},…
Двоичная система:…,2^{-2},2^{-1},2^{0},2^{1},2^{2},2^{3},…
Восьмеричная система:…,8^{-2},8^{-1},8^{0},8^{1},8^{2},8^{3},…
Шестнадцатеричная система:…,16^{-2},16^{-1},16^{0},16^{0},16^{2},…
Пример. Разложить по базису системы счисления числа:
$101{,}01_2$; $673{,}2_8$; $15FC_{16}$.
Решение

$$101{,}01_2 = 1\cdot2^2 + 0\cdot2^1 + 1\cdot2^0 + 0\cdot2^{-1} + 1\cdot2^{-2};$$

$$673{,}2_8 = 6\cdot8^2 + 7\cdot8^1 + 3\cdot8^0 + 2\cdot8^{-1};$$

$$15FC_{16} = 1\cdot16^3 + 5\cdot16^2 + F\cdot16^1 + C\cdot16^0.$$

Если выразить шестнадцатеричные цифры через их десятичные значения *(F = 15; C = 12), то* $15FC_{16} = 1\cdot16^3 + 5\cdot16^2 + 15\cdot16^1 + 12\cdot16^0$.

В позиционных системах счисления основание системы определяет, во сколько раз различаются значения соседних разрядов.

Разряд – это номер позиции цифры в числе. Разряд растет влево от десятичной точки (запятой) и уменьшается вправо, принимая отрицательное значение.

Можно сделать вывод, что умножение или деление числа на основание системы приведет к перемещению запятой, отделяющей целую часть от дробной, вправо или влево соответственно, например:

$$536{,}15_{10} \cdot 10 = 5361{,}5_{10};\quad 1001{,}11_2 / 10_2 = 100{,}111_2$$

3. Перевод десятичных чисел в другие системы счисления

Перевод десятичного числа в другую систему счисления может выполняться разными способами. При этом надо учитывать, что алгоритмы перевода целых чисел и правильных дробей будут отличаться. Для смешанного числа целая и дробная части переводятся отдельно по соответствующим алгоритмам. В итоговой записи искомого числа они объединяются и разделяются запятой.

Так называемый метод поэтапного деления заключается в последовательном делении исходного числа и получаемых неполных частных на основание той системы счисления, в которую осуществляется перевод. Остатки деления составляют искомое число.

4. Алгоритм перевода целого десятичного числа *N* в позиционную систему с основанием *p*

1. Разделить число *N* на *p*
2. Полученный остаток дает цифру, стоящую в нулевом разряде *p* –ичной записи числа *N*.
3. Полученное частное снова разделить на *p* и снова запомнить полученный остаток – это цифра первого разряда, и т.д.
4. Такое последовательное деление производится до тех пор, пока частное не станет равным 0.
5. Цифрами искомого числа являются остатки от деления, выписанные слева направо начиная с последнего полученного остатка.

Пример. Перевести десятичное число 26 в двоичную, троичную и шестнадцатеричную системы счисления.

Решение

$26_{10} \rightarrow X_2$ $26_{10} \rightarrow X_3$ $26_{10} \rightarrow X_{16}$

26/2| 0 26/3| 2 26/16| 10 → A

13/2| 1 8/3| 2 1| 1

6/2| 0 2| 2

3/2| 1

1| 1

Результат

$X_2 = 11010$ $X_3 = 222$ $X_7 = 1A_{16}$

5. Алгоритм перевода правильной десятичной дроби *N* в позиционную систему с основанием *p*

1. Умножить данное число (дробную часть смешанного числа) на основание заданной позиционной системы *p*.
2. Целая часть полученного произведения является цифрой старшего разряда искомой дроби
3. Оставшаяся дробная часть полученного произведения вновь умножается на *p*, и целая часть результата считается следующей цифрой искомой дроби.
4. Операция продолжается до тех пор, пока дробная часть не окажется равной нулю или не будет достигнута требуемая точность

Пример 1. Перевести десятичную дробь 0,375 в двоичную, троичную и шестнадцатеричную систему счисления. Перевод выполнить с точностью до третьего знака.

Решение

$0{,}375_{10} \rightarrow 0{,}X_2$ $0{,}375_{10} \rightarrow 0{,}X_3$ $0{,}375_{10} \rightarrow 0{,}X_{16}$

$p = 2$	$p = 3$	$p = 16$
$0{,}375 \cdot 2 = 0{,}75$	$0{,}375 \cdot 3 = 1{,}125$	$0{,}375 \cdot 16 = 6{,}0$
$0{,}75 \cdot 2 = 1{,}5$	$0{,}125 \cdot 3 = 0{,}375$	
$0{,}5 \cdot 2 = 1{,}0$	$0{,}375 \cdot 3 = 1{,}125$	
$0{,}125 \cdot 3 = 0{,}375$		

Результат: $0{,}375_{10} = 0{,}011_2$ $0{,}375_{10} = 0{,}101_3$ $0{,}375_{10} = 0{,}6_{16}$

Пример 2. Перевести десятичное число 26,375 в двоичную, троичную и шестнадцатеричную систему счисления.

Решение

Из рассмотренных выше примеров следует:

$26{,}375_{10} = 11010{,}011_2 = 222{,}101_3 = 1A{,}6_{16}$

6. Перевод чисел любых позиционных систем счисления в десятичную систему

Представление чисел в развернутой форме одновременно является способом перевода чисел в десятичную систему из любой другой позиционной системы счисления. Достаточно подсчитать результат по правилам десятичной арифметики.

Пример. Получить десятичные эквиваленты чисел:

$101{,}01_2$; $673{,}2_8$; $15AC_{16}$.

Решение

$101{,}01_2 = 1\cdot2^1 + 0\cdot2^0 + 0\cdot2^{-1} + 1\cdot2^{-2} = 4 + 0 + 1 + 0 + 1/4 = 5 + 0{,}25 = 5{,}25_{10}$

$673{,}2_8 = 6\cdot8^2 + 7\cdot8^1 + 3\cdot8^0 + 2\cdot8^{-1} = 384 + 56 + 3 + 2\cdot0{,}25 = 443{,}25_{10}$

$15AC_{16} = 1\cdot16^3 + 5\cdot16^2 + 10\cdot16^1 + 12\cdot16^0 = 4096 + 1280 + 160 + 12 = 5548_{10}$

7. Выполнение арифметических операций в позиционных системах счисления

Арифметические действия в позиционных системах счисления выполняются по общим правилам.

При выполнении арифметических действий числа, представленные в разных системах счисления, нужно сначала привести к одному основанию.

Перенос в следующий разряд при сложении и заем из старшего разряда при вычитании определяется величиной основания системы счисления.

Переполнение разряда наступает тогда, когда значение числа в нем становится равным или больше основания.

Правила арифметики многоразрядных чисел основывается на использовании таблиц сложения и умножения одноразрядных чисел.

Пример 1. Вычислить выражение $10111_2 – 51_{16}/33_8$, записав результат в двоичной системе счисления.

Решение

Приведем числа выражения в единую систему счисления, например, десятичную:

$10111_2 = 23_{10}$

$51_{16} = 1010001_2 = 81_{10}$

$33_8 = 11001_2 = 27_{10}$

Вычислим выражение: $23 – 81/27 = 20$

Результат $20_{10} = 10100_2$

Пример 2. Сложить числа: 17_8 и 17_{16}.

Решение

Приведем число 17_{16} к основанию 8, используя двоичную систему.

$17_{16} = 23_{10} = 10111_2 = 27_8$

$17_8 = 15_{10} = 1111_2$

$27_8 + 17_8 = 23_{10} + 15_{10} = 38_{10} = 46_8$

Проверка решения:

$17_8 = 1 \cdot 8 + 7 = 15_{10}$ $17_{16} = 1 \cdot 16 + 7 = 23_{10}$ $15 + 23 = 38$

$38_{10} = 4 \cdot 8 + 6 = 32 + 6 = 46_8$

8. Представление чисел в памяти компьютера
Представление целых чисел

Любая информация в ЭВМ представляется в виде двоичных кодов. Отдельные элементы двоичного кода, принимающие значение 0 или 1, называют *разрядами* или битами. Память компьютера условно делится на отсеки или ячейки, каждая из которых имеет свой номер. Нумерация начинается с нуля.

Минимальной адресуемой ячейкой памяти является байт – 8 двоичных разрядов (бит). Порядковый номер байта называется его *адресом*.

Наибольшую последовательность битов, которую процессор может обрабатывать как единое целое, называют ***машинным словом***. Длина машинного слова может быть равной – 8, 16, 32 бит и т.д. Адрес машинного слова равен адресу младшего байта, входящего в это слово. Двоичные разряды в любой ячейке памяти нумеруются справа налево, начиная с нуля.

Существует два основных формата представления чисел в памяти компьютера. Один из них используется для кодирования целых чисел, второй используется для задания некоторого подмножества действительных чисел (*формат с плавающей точкой*).

Для размещения целых положительных чисел отводится, как правило, один или два байта. Существует ограничение на множество целых чисел, представимых в памяти компьютера. Максимальное число, которое можно поместить в одном байте, – это $255_{10} = 11111111_2 = 2^8 - 1$. Такие числа могут применяться, например, для организации различных счетчиков, записи адресов ячеек, даты и времени, размеров графических изображений в пикселях.

Для положительных и отрицательных чисел существует знаковый способ представления числа. Под знак отводится ***старший*** разряд ячейки: 0 – для положительных чисел, 1 – для отрицательных чисел. Числа со знаком, представленные в одном байте, попадают в диапазон от -128 до 127, в двух байтах попадают в диапазон от -32768 до 32767, а положительные числа попадают в диапазон от 0 до 65535. Следует отметить, что модули положительных и отрицательных чисел не равны. Почему?

Для упрощения реализации арифметических операций в компьютере целые числа представляются специальными кодами – ***прямым, обратным*** **и** ***дополнительным***.

Для положительного целого числа прямой, обратный и дополнительный коды совпадают с двоичным представлением числа. В знаковом разряде числа будет 0.

Для отрицательного целого числа:

- прямой код совпадает с двоичным представлением числа. В знаковом разряде числа – 1;
- обратный код получается инвертированием значений всех разрядов, кроме знакового разряда;
- дополнительный код получается путем прибавления единицы к младшему разряду обратного кода. Перенос в знаковый разряд при этом теряется.

Пример 1. Определить прямой, обратный и дополнительный коды для следующих чисел: $36_{10} = 100100_2$, $-36_{10} = -100100_2$, $-35_{10} = -100011_2$.

Решение

Будем считать, что число размещается в одном байте. Старший бит – знак разряда. Незначащие нули добавляются слева от числа.

Число	**Прямой код**	**Обратный код**	**Дополнительный код**
100100	00100100	00100100	00100100
-100100	10100100	11011011	11011100
-100011	10100011	11011100	11011101

Пример 2. Как будет представлено в памяти компьютера целое число 12345_{10}?

Решение

Для размещения числа потребуется два байта (16 бит или разрядов).

Поскольку число положительное, то в старшем бите будет 0.

Переведем число в двоичную систему счисления:

$12345_{10} = 11000000111001_2$.

Результат: $0110000\ \ 00111001_2$

Целочисленная двоичная арифметика в ЭВМ

Особенности двоичной системы счисления позволяют создавать специфические алгоритмы вычитания и умножения двоичных чисел, наиболее подходящие для аппаратной реализации.

Целочисленная двоичная арифметика используется при изучении программирования, в процессе освоения операторов цикла, выбора, стандартных процедур ***val*** и ***str***, операций над целыми числами ***div*** и ***mod***, операций над строковыми величинами.

Сложение чисел производится в дополнительных кодах поразрядно.

При выполнении арифметических операций число может выйти за указанные границы. Произойдет ***переполнение разрядной сетки***, поэтому при работе с большими целыми числами под них выделяется больше места, например 4 байта. Факт переполнения всегда фиксируется установкой в единицу специального управляющего бита, который программа может проанализировать. Процессор “заметит” переполнение, но предоставит программе право выбора дальнейших действий.

Чтобы избежать ситуации переполнения, в языках программирования предусмотрено строгое описание ***типа переменной***, которым определяется набор возможных ее значений. При выходе за границы допустимых значений возникает ошибка выполнения программы. В программах, где предусматривается обработка числовой информации, например *Excel*, при переполнении разрядной сетки производится автоматическое преобразование целого числа в вещественный вид. Например, 123000000000 = 1,23E+11 = $1{,}23 \cdot 10^{11}$.

Вычитание целых чисел эквивалентно сложению с отрицательным числом. Отрицательное число может быть представлено в прямом коде. Однако использование прямого кода усложняет структуру команд процессора. При выполнении сложения чисел с разными знаками требуется выбрать из них большее по модулю, затем вычесть из него меньшее, выяснить знак большего и присвоить этот знак остатку. По этой причине в компьютерах используется представление отрицательного числа в ***дополнительном*** коде. Таким образом, операция вычитания выполняется как сложение с дополнительным кодом вычитаемого.

Пример. Выполнить операцию вычитания 25 – 34.

Решение

Учтем, что 25 – 34 = 25 + (–34).

Переведем числа 25 и 34 в двоичную систему счисления:

$25_{10} = 11001_2$ и $34_{10} = 100010_2$

Запишем прямые, обратные и дополнительные коды, воспользовавшись 8-разрядной сеткой:

После сложения дополнительных кодов получим код 11110111. Единица в старшем разряде (бите) полученного кода означает, что число отрицательное. Следовательно, результат надо перевести в обратный, а затем в прямой код: 11110111 → 10001000 →10001001.

Полученный результат в десятичном представлении равен:

$-1001_2 = -9_{10}$.

Число	**Прямой код**	**Обратный код**	**Дополнительный код**
25	00011001	00011001	00011001
-34	00100010	11011101	11011110

Операции ***умножения*** и ***деления*** выполняются в прямом коде с использованием итерационных алгоритмов (ряда повторяющихся шагов).

Умножение двоичных чисел сводится к двум операциям: сложения и сдвига. Первый сомножитель складывается сам с собой столько раз, сколько единиц содержится во втором сомножителе. Каждый шаг итерации содержит очередной сдвиг влево на двоичный разряд и прибавление первого сомножителя, если соответствующий разряд второго сомножителя содержит 1, и сдвиг без прибавления, если в таком разряде 0. Таким образом, реализации отдельной операции умножения в процессоре не требуется. Знаки произведения и частного определяются по тем же правилам, что и в десятичной системе счисления.

Операция ***деления*** для целых чисел однозначно не определена, поскольку в общем случае приводит к появлению нецелых (вещественных) чисел. В разных процессорах существуют различные методы и алгоритмы реализации этой операции.

9. Представление вещественных чисел

В отличие от целых чисел, которые представляются в памяти компьютера абсолютно точно, значения вещественных чисел являются *приближенными*. В некоторых областях вычислений требуются очень большие или малые действительные числа. Для получения большей точности применяют *запись чисел с плавающей точкой*.

В общем случае в формате с плавающей точкой число представляется в виде произведения двух сомножителей: $R=mP^n$,

где m – *мантисса* числа;

P – основание системы счисления;

n – порядок, указывающий, на какое количество позиций и в каком направлении должна сместиться точка, отделяющая дробную часть в мантиссе.

Например, число 5,14 может быть записано $0{,}514\cdot10^1$ или $51{,}4\cdot10^{-1}$ и тому подобное. Запятая (десятичная точка) перемещается, или “плавает”, вправо и влево в зависимости от порядка числа.

При работе с числами в языках программирования в вычислительных системах используется *экспоненциальная* форма записи: $R = mE\pm n$,

Где E – десятичное основание системы.

Например, 3,1467890000E+2 = 314, 6789.

Для увеличения количества значащих цифр в числе мантиссу обычно подвергают ***нормализации***. Нормализованная мантисса меньше единицы и первая значащая цифра не ноль.

Пример 1. Записать числа в нормализованном виде:

а) $159{,}16 = 0{,}15916 \cdot 10^{3}$; в) $0{,}05975 = 0{,}5975 \cdot 10^{-1}$;

б) $36{,}256 = 0{,}36256 \cdot 10^{2}$; г) $0{,}000142 = 0{,}142 \cdot 10^{-3}$.

Пример 2. Нормализовать двоичное число $-101{,}01_2$.

Запись двоичного нормализованного числа выглядит так:

$-101{,}01_2 = -0{,}10101_2 \cdot 2^{11}{}_2$.

Выполнение арифметических действий над числами с плавающей запятой (десятичной точкой) гораздо сложнее целочисленной арифметики. Для некоторых процессоров операции над вещественными числами вынесены в отдельный узел, который называют математическим сопроцессором.

Сложение чисел с плавающей запятой выполняется в соответствии со следующим алгоритмом.

1. Представить числа A и B в нормализованном виде, записав отдельно значения мантисс и порядков.
2. Выровнять порядки по числу с большим порядком.
3. Выровнять число цифр в мантиссах по числу, порядок которого не изменился.
4. Сложить числа.

5. Нормализовать сумму, оставив число цифр в мантиссе таким, как у числа, порядок которого не изменился.

Пример. Найти сумму чисел A = 9,6098 и B = 98,009 по правилу сложения чисел с плавающей запятой.

Решение

Шаг	**Число**	**Нормализованное число**	**Порядок**	**Мантисса**	**Число цифр в мантиссе**
1	A=9,6098	$0{,}96098 \cdot 10^{1}$	1	96098	5
	B=98,009	$0{,}98009 \cdot 10^{2}$	2	98009	5
2	A	$0{,}096098 \cdot 10^{2}$	2	096098	6
3	A	$0{,}09609 \cdot 10^{2}$	2	09609	5
4	A+B	$1{,}07618 \cdot 10^{2}$	2	-	
5	A+B	$0{,}10761 \cdot 10^{3}$	3	10761	5

Размещение чисел с плавающей запятой (точкой)

Метод представления вещественных чисел в памяти компьютера предполагает хранение двух чисел: *мантиссы* и *порядка*. Чем больше разрядов отводится под запись мантиссы, тем выше точность представления числа. Чем больше разрядов занимает порядок, тем шире диапазон чисел, представимых в компьютере при заданном формате. Правила кодирования мантиссы и порядка отличаются для различных типов компьютеров.

Для размещения вещественного числа могут использоваться четыре байта (32 бита) – короткий формат, 8 байтов – длинный формат, 16 байтов – формат повышенной точности. В любом случае старший байт остается постоянным, а изменяется область, отведенная под мантиссу. Старший байт включает в себя:

один бит (старший) – знак числа;

один бит – знак порядка;

шесть битов – порядок числа.

Положительные и отрицательные значения порядка существенно усложняют обработку вещественных чисел. Поэтому во многих современных компьютерах используют не прямое значение порядка, а *смещенное*. Его называют *характеристикой* числа. Для разных типов ЭВМ существуют разные варианты смещения порядка.

Использование смещенной формы позволяет производить операции над порядками как над беззнаковыми числами, что упрощает операции сравнения, сложения и вычитания порядков, а также упрощает операцию сравнения самих нормализованных чисел.

Поскольку мантисса нормализована, ее двоичное значение всегда начинается с единицы. Поэтому во многих ЭВМ эта единица даже не записывается в оперативное запоминающее устройство (но подразумевается), что дает дополнительный разряд мантиссы (“скрытая единица”) или порядка.

Точность представления вещественного числа зависит от количества разрядов, отведенных под мантиссу.

Диапазон числа зависит от размера порядка.

Смещенный порядок имеет только положительные значения. Смещение выбирается так, чтобы минимальному значению истинного порядка соответствовал нуль.

Наименьшее по абсолютной величине вещественное число равно нулю. Наибольшее по абсолютной величине число в форме с плавающей точкой – это число с самой большой мантиссой и самым большим порядком.

Диапазон вещественных чисел значительно шире диапазона целых чисел.

10. Контрольные вопросы

1. Параметры позиционных систем счисления (основание, алфавит, базис).
2. Как связаны между собой виды информации?
3. Чему равна разность десятичного числа 205 и восьмеричного числа 105?
4. Найдите произведение десятичного числа 101 на двоичное число 110.
5. Какому двоичному числу соответствует десятичное число 254?
6. Сколько разрядов используется для кодирования вещественных чисел?

2. ЛОГИЧЕСКИЕ ОСНОВЫ ЭВМ

Содержание

1. Понятие алгебры логики
2. Основные логические операции
3. Логические основы компьютеров
4. Контрольные задачи

1. Понятие алгебры логики

Алгебра логики (математическая логика или булева алгебра, или алгебра высказываний) используется в информатике для того, чтобы можно было определять истинность или ложность составных высказываний, не вникая в их содержание.

Каждое составное высказывание можно выразить в виде формулы (логического выражения), в которую войдут логические переменные, обозначающие высказывания, и знаки логических операций, обозначающие логические функции. Для записи составных высказываний в виде логических выражений на формальном языке (языке алгебры логики) в составном высказывании нужно выделить простые высказывания и логические связи между ними. Истинность или ложность составных высказываний можно определять чисто формально, руководствуясь законами алгебры высказываний, не обращаясь к смысловому содержанию высказываний.

В алгебре логики (высказываний) суждениям (простым высказываниям) ставятся в соответствие переменные (заглавные буквы латинского алфавита), которые могут принимать только два значения "истина (true)" (1) и "ложь (false)" (0).

Пример: A – "2 × 2 = 4" – истина (1) A = 1

B – "2 × 2 = 5" – ложь (0) B = 0

Простые высказывания называются ***переменными***, а сложные – ***логическими функциями***.

2. Основные логические операции

Логическое умножение (конъюнкция) – объединение 2-х (или нескольких высказываний) в одно с помощью союза "**И**". Конъюнкция истинна тогда и только тогда, когда истинны входящие в нее простые высказывания. Обозначения: **&**, ^, *, **and**.

Например, F = A&B, F = A^B, F = A × B, F = A and B.

Ниже приведена таблица истинности для операции логического умножения.

A	B	F = A and B
0	0	**0**
0	1	**0**
1	0	**0**
1	**1**	**1**

Логическое сложение (дизъюнкция) – объединение 2-х (или нескольких) высказываний в одно с помощью союза "**ИЛИ**". Дизъюнкция истинна тогда, когда истинно хотя бы одно из входящих в выражение простых высказываний. Обозначения: **v**, **+**, **or**.

Например, F = A v B, F = A + B, F = A or B. Ниже приведена таблица истинности для операции логического сложения.

A	B	F = A or B
0	0	**0**
0	1	**1**
1	0	**1**
1	**1**	**1**

Логическое отрицание (инверсия) – присоединение частицы "**НЕ**" к высказыванию. Инверсия делает истинным ложное высказывание, ложным – истинное высказывание. Обозначения: **¬**, **not**.

Например, ¬F = A, F = ¬A, F = not A. Ниже приведена таблица истинности для операции логического отрицания.

A	**F = not A**
0	**1**
1	**0**

Обычный приоритет выполнения логических операций:

1) инверсия;
2) конъюнкция;
3) дизъюнкция.

Значение логической функции для разных сочетаний значений входных переменных или наборов входных переменных обычно задается ***таблицей истинности***. Таблицы истинности для всех логических операций приведены выше.

Равносильные логические выражения

Логические выражения, у которых таблицы истинности совпадают, называются равносильными (эквивалентными). Обозначение – знак "=".

Пример: ¬A&¬B = ¬(A v B).

Импликация и эквиваленция. В обыденной и научной речи кроме базовых логических связок «**И**», «**ИЛИ**», «**НЕ**», используются и некоторые другие: «**ЕСЛИ..., ТО...**», «**ТОГДА... И ТОЛЬКО ТОГДА, КОГДА...**» и др. Некоторые из них имеют свое название и свой символ и им соответствуют определенные логические функции.

Логическое следование (импликация) образуется соединением двух высказываний в одно с помощью оборота речи «**ЕСЛИ**..., **ТО**...». Логическая операция импликации «**ЕСЛИ** A, **ТО** B», обозначается A ® B.

Составное высказывание, образованное с помощью операции логического следования (импликации), ложно тогда и только тогда, когда из истинной предпосылки (первого высказывания) следует ложный вывод (второе высказывание).

Например:

1) высказывание «Если число делится на 10, то оно делится на 5» истинно, так как истинны и первое высказывание (предпосылка), и второе высказывание (вывод);

2) высказывание «Если число делится на 10, то оно делится на 3» ложно, так как из истинной предпосылки делается ложный вывод.

Однако операция логического следования несколько отличается от обычного понимания слова «следует». Если первое высказывание (предпосылка) ложно, то вне зависимости от истинности или ложности второго высказывания (вывода) составное высказывание истинно. Это можно понимать таким образом, что из неверной предпосылки может следовать что угодно.

В алгебре высказываний все логические функции могут быть сведены путем логических преобразований к трем базовым: логическому умножению, логическому сложению и логическому отрицанию.

Логическое равенство (эквивалентность) образуется соединением двух высказываний в одно «...**ТОГДА И ТОЛЬКО ТОГДА**, **КОГДА**...». Логическая операция эквивалентности «А **ЭКВИВАЛЕНТНО** В» обозначается $A \sim B$ и выражается с помощью логической функции, которая задается соответствующей таблицей истинности.

A	B	$A \sim B$
0	0	1
0	1	0
1	0	0
1	1	1

Логические основы компьютеров

Логическими элементами компьютеров являются электронные схемы **НЕ**, **ИЛИ**, **И**, **ИЛИ-НЕ**, **И-НЕ** и другие (называемые также **вентилями**).

С помощью этих схем можно реализовать любую логическую функцию, описывающую работу устройств компьютера. Обычно у вентилей бывает от двух до восьми входов и один выход.

Чтобы представить два логических состояния — “**1**” и “**0**” в вентилях, соответствующие им входные и выходные сигналы имеют один из двух установленных уровней напряжения. Например, **+5** вольт и **0** вольт.

Высокий уровень обычно соответствует значению “**истина**” (“**1**”), а низкий — значению “**ложь**” (“**0**”).

Каждый логический элемент имеет свое условное обозначение, которое выражает его логическую функцию, но не указывает на то, какая именно электронная схема в нем реализована. Это упрощает запись и понимание сложных логических схем.

Работу логических элементов описывают с помощью таблиц истинности.

Таблица истинности это табличное представление логической схемы (операции), в котором перечислены все возможные сочетания значений истинности входных сигналов (операндов) вместе со значением истинности выходного сигнала (результата операции) для каждого из этих сочетаний.

Схема НЕ (инвертор) реализует операцию отрицания. Связь между входом **X** этой схемы и выходом **Z** можно записать соотношением $Z = \neg X$, где «¬» читается как «**не X**» или «**инверсия X**». Если на входе схемы 0, то на выходе 1. Когда на входе 1, на выходе 0.

Схема ИЛИ реализует дизъюнкцию двух или более логических значений. Когда хотя бы на одном входе схемы ИЛИ будет единица, на её выходе также будет единица.

Связь между входами **X**, **Y** этой схемы и выходом **Z** можно записать соотношением **X \/ Y = Z**

Схема ИЛИ-НЕ. Когда хотя бы на одном входе схемы ИЛИ-НЕ будет единица, на её выходе будет ноль. Связь между входами **X**, **Y** этой схемы и выходом **Z** можно записать соотношением **¬(X \/ Y) = Z.** Логический элемент **ИЛИ-НЕ**

Схема И реализует конъюнкцию двух или более логических значений.

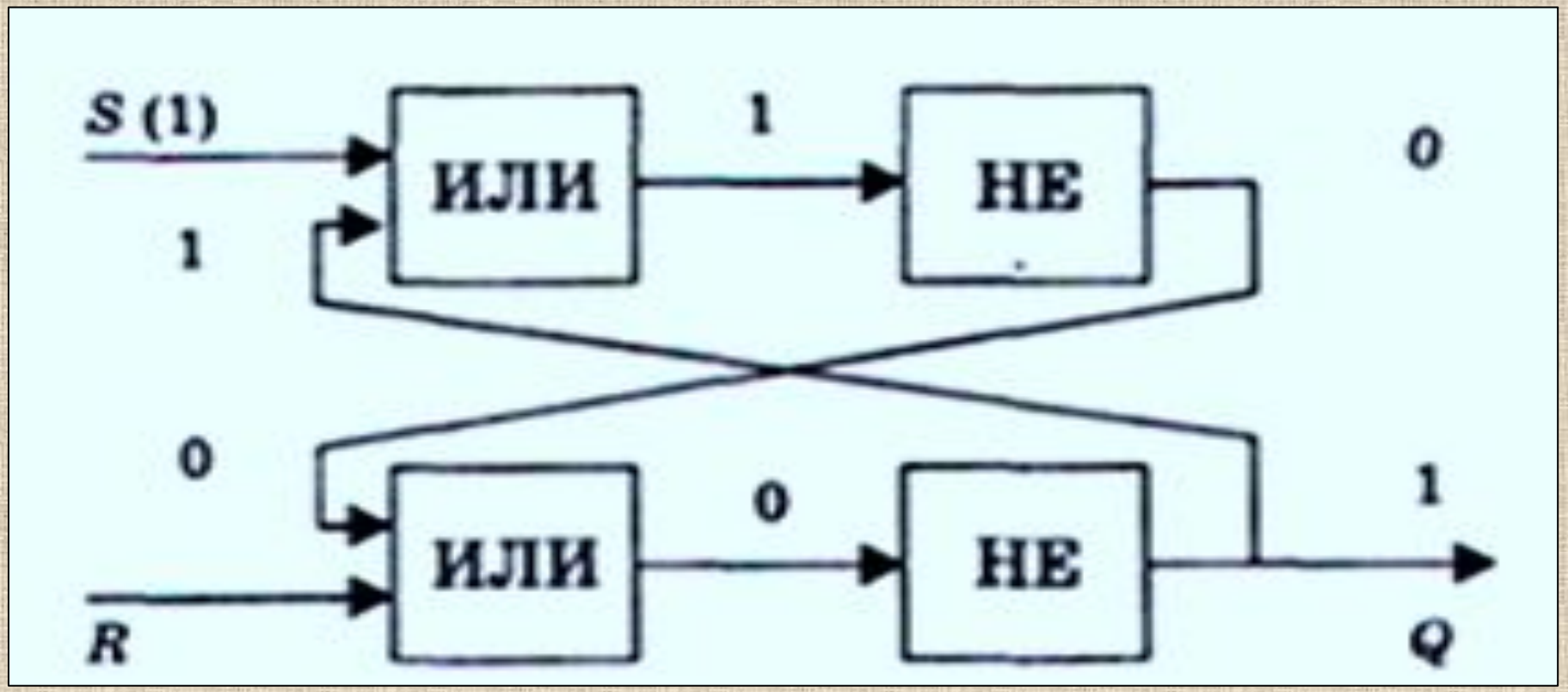
Единица на выходе схемы И будет тогда и только тогда, когда на всех входах будут единицы**.** Когда хотя бы на одном входе будет ноль, на выходе также будет ноль. Связь между входами **X**, **Y** этой схемы и выходом **Z** можно записать соотношением **Z = X /\ Y.** Операция конъюнкции на функциональных схемах обозначается знаком «**&**» (читается как «**амперсанд**»), являющимся сокращенной записью английского слова **and**.

Схема И-НЕ. Ноль на выходе схемы И-НЕ будет тогда и только тогда, когда на всех входах будут единицы. Связь между входами **X**, **Y** этой схемы и выходом **Z** можно записать соотношением **¬(X /\ Y) = Z.**

Триггер — элемент оперативной памяти компьютера, способный запомнить и сохранить один бит информации. Триггер был изобретен в 1918 г. М.А. Бонч-Бруевичем, руководителем Нижегородской лаборатории связи.

Триггер имеет два устойчивых состояния, в которые он поочередно переходит под воздействием входных сигналов при записи информации.

Существует множество типов триггеров. Один из них, RS-триггер, построен на двух элементах ИЛИ-НЕ.



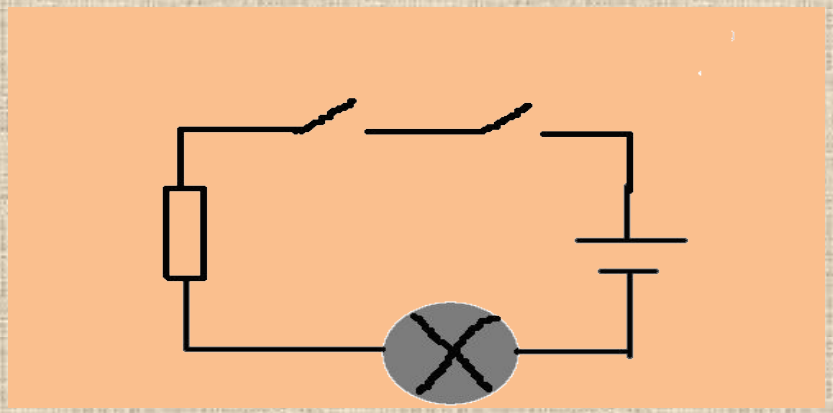
Вход *R* называют входом установки триггера в нулевое состояние, а вход S - в единичное.

Триггер имеет два выхода: Q - прямой выход,

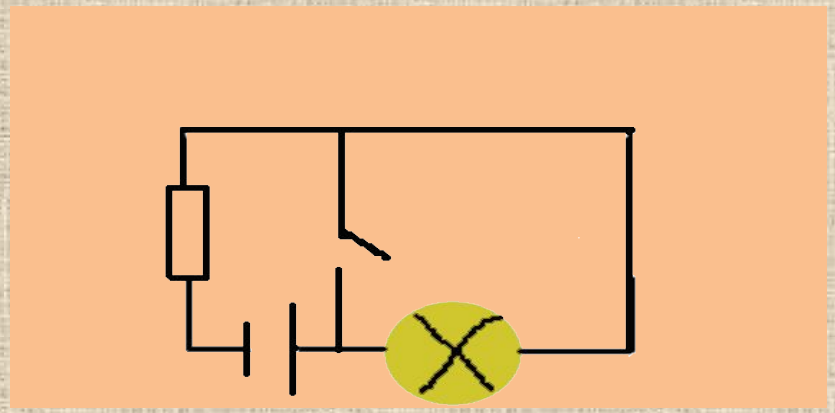
P - инверсный.

Триггер — элемент оперативной памяти компьютера, способный запомнить и сохранить один бит информации.

Физические аналоги логических элементов



Конъюнктор



Инвертор

Контрольные задачи

Пусть a, b, c — логические величины, которые имеют следующие значения: a = истина, b = ложь, c = истина. Нарисуйте логические схемы для следующих логических выражений и вычислите их значения:

1. a и b;
2. a или b;
3. не a или b;
4. a и b или c;
5. a или b и c;
6. не a или b и c;
7. (a или b) и (c или b);
8. не (a или b) и (c или b);
9. не (a и b и c).