Представление данных в ЭВМ

Все данные в электронных устройствах кодируются числами.
При проведении математических расчетов числа внутри ЭВМ могут быть представлены с помощью

естественной

и

нормальной

форм записи.

Примером записи числа в естественной форме может служить число:

173,856

Для записи такого числа отводится машинное слово.

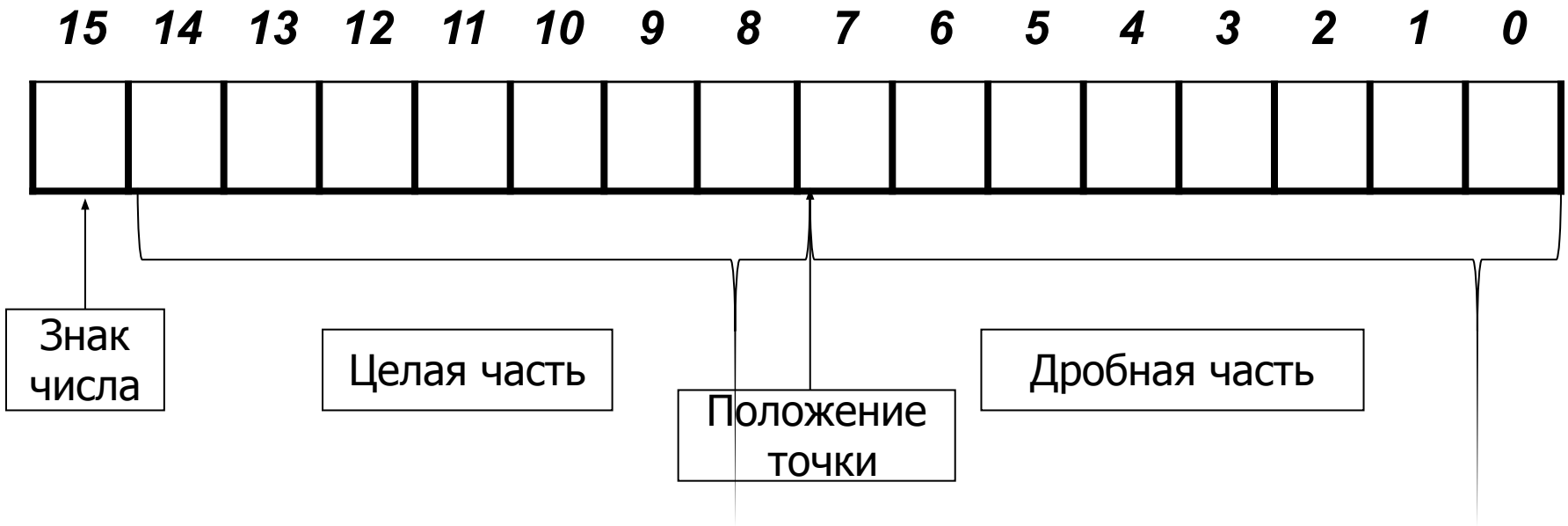
Машинное слово — машиннозависимая и платформозависимая величина, измеряемая в битах или байтах, равная разрядности регистров процессора и/или разрядности шины данных (обычно некоторая степень двойки). На ранних компьютерах размер слова совпадал также с минимальным размером адресуемой информации (разрядностью данных, например 8 бит).

Данные, находящиеся в машинном слове, обрабатываются как единое целое за одну операцию. Например, считываются на быстрые регистры, к одному числу, хранящемуся в машинном слове на быстрых регистрах, может быть прибавлено другое число.

Машинное слово является структурной единицей информации ЭВМ. В первых ЭВМ фирмы IBM машинное слово состояло из 16 разрядов, т.е. его длина составляла 16 бит или 2 байта. В современных ЭВМ длина машинных слов составляет 32..128 разрядов. Такие слова называют ***удвоенным*** словом(32 разряда), ***учетверенным*** словом(64 разряда), и.т. далее.

Для записи числа в естественной форме машинное слово делится на два фиксированных поля (части). Первое поле отводится для записи целой части числа, второе - для записи дробной части числа. Старший разряд используется для указания знака числа. Ниже на рисунке номерами от 0..15 обозначены разряды (биты) машинного слова.

При таком представлении положение запятой между целой и дробной частью четко определено. Такое представление чисел называют представлением с ***фиксированной точкой.***



На предыдущем рисунке для упрощения показано машинное слово длиной 16 разрядов. Физически каждый разряд машинного слова представляет собой отдельный элемент памяти –***триггер.***

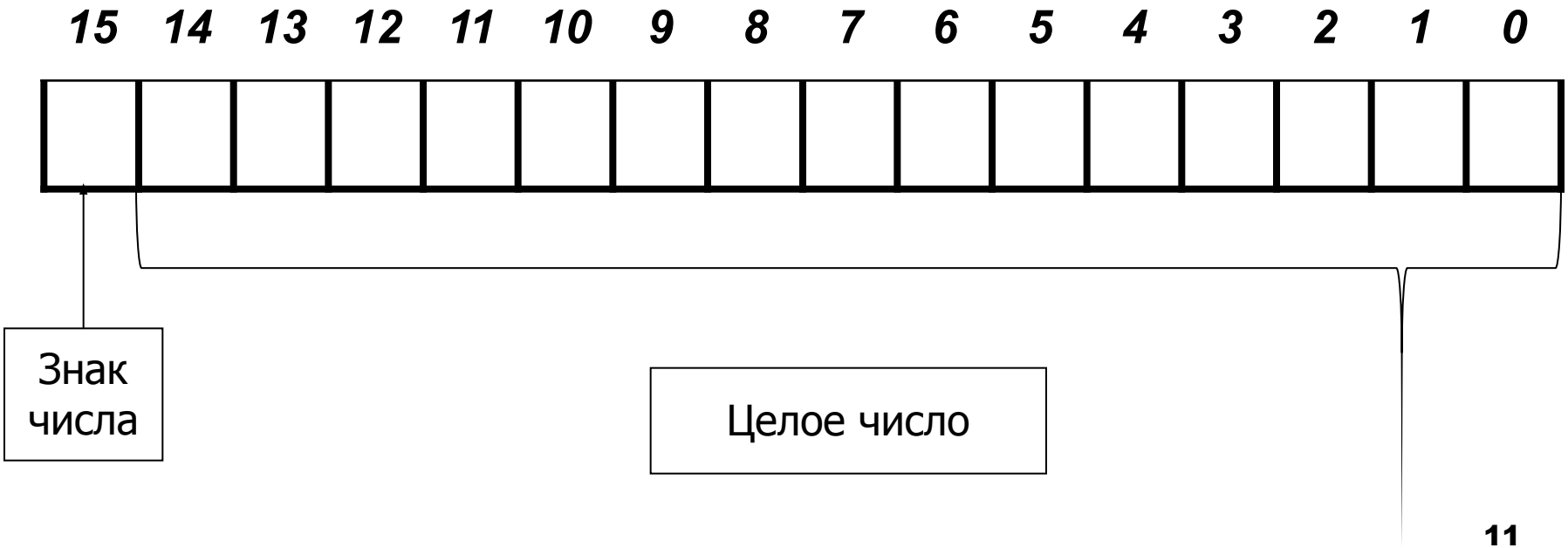
Триггер – электронный микроэлемент, который хранит не заряд, а состояние

(включен/выключен).Приняв одно из состояний за «1», а другое за «0», можно считать, что триггер хранит (помнит) один разряд числа, записанного в двоичном коде.

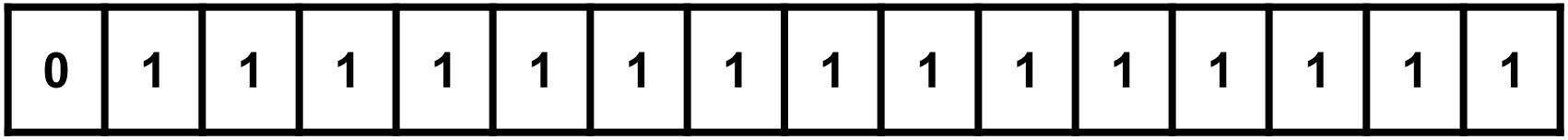
При изготовлении триггеров применяются преимущественно полупроводниковые приборы - транзисторы, в прошлом — электромагнитные реле, электронные лампы.

Недостатком формы с фиксированной точкой является малый диапазон представления чисел. Как правило, в этой форме записывают только ***целые числа***.

При записи целых чисел отпадает необходимость отводить часть машинного слова для записи дробной части числа.



В одном машинном слове (16 разрядов) можно разместить целые числа в диапазоне от -32768..32767



На рисунке изображено целое положительное число 32767 в двоичной системе счисления.

Нормальная форма записи числа имеет следующий вид:

$$0{,}173856 \cdot 10^{3}$$

т.е. $N = m \cdot d^{p}$

N – число;

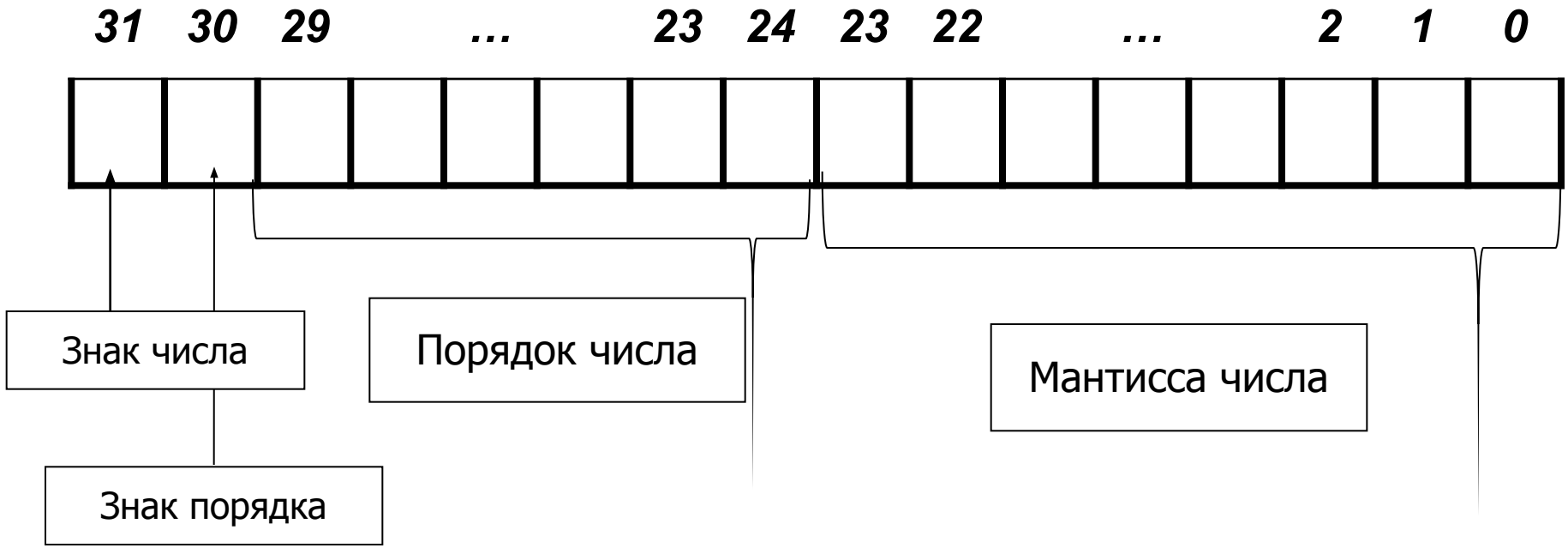
m – мантисса числа<1;

p – порядок;

d – основание системы счисления.

Порядок указывает местоположение в числе запятой, отделяющей целую часть числа от дробной. В зависимости от порядка запятая передвигается (плавает) по мантиссе. Такая форма представления числа называется формой с ***плавающей точкой.***

Следующий рисунок иллюстрирует форму числа с плавающей точкой на примере 32-разрядного машинного слова.



Пример:

Пусть m=0,3, d=10, а порядок будем брать разным:

p=-1 $0.3 \cdot 10^{-1}=0.03$

p=-2 $0.3 \cdot 10^{-2}=0.003$

p=2 $0.3 \cdot 10^{2}=30$

p=3 $0.3 \cdot 10^{3}=300$

Из приведенного примера видно, что благодаря изменению порядка точка перемещается по мантиссе. Благодаря порядку получаем различные десятичные числа.

В этом случае машинное слово делится на два основных поля. В одном поле записывается мантисса числа, во втором указывается порядок числа. Диапазон представления чисел с плавающей точкой значительно больше диапазона представления чисел с фиксированной точкой.

Однако быстродействие ЭВМ при обработке чисел с плавающей точкой гораздо ниже, чем при обработке чисел с фиксированной точкой. Почему это так станет понятно из дальнейшего рассказа.

Арифметические основы компьютера

Все данные в электронных устройствах кодируются числами.

Все электронные устройства в настоящее время оперируют только двоичными числами.

В вычислительной технике широко применяют ***двоичную***, ***восьмеричную*** и ***шестнадцатеричную*** системы счисления.

Операции над данными(сложение, вычитание и пр.) осуществляются только в двоичной системе счисления, для хранения данных используется ***восьмеричная*** и ***шестнадцатеричная*** системы счисления.

В простых электронных устройствах может применяться ***двоично-десятичная*** система.

Немного истории

Полный набор из 8 триграмм и 64 гексаграмм, аналог 3-битных и 6-битных цифр, был известен в древнем Китае в классических текстах книги Перемен. Порядок гексаграмм в книге Перемен, расположенных в соответствии со значениями соответствующих двоичных цифр (от 0 до 63), и метод их получения был разработан китайским учёным и философом Шао Юн в XI веке.

Триграмма	Символ Unicode	Название	Сторона света	Стихия
☰	☰ (U+2630)	乾 Цянь	Северо-Запад	Небо
☷	☷ (U+2637)	坤 Кунь	Юго-Запад	Почва
☳	☳ (U+2633)	震 Чжэнь	Восток	Дерево
☵	☵ (U+2635)	坎 Кань	Север	Вода
☶	☶ (U+2636)	艮 Гэнь	Северо-Восток	Гора
☴	☴ (U+2634)	巽 Сюнь	Юго-Восток	Ветер
☲	☲ (U+2632)	离 Ли	Юг	Огонь
☱	☱ (U+2631)	兑 Дуй	Запад	Металл

Современная двоичная система была полностью описана ***Готфридом Лейбницем*** в XVII веке в работе ***Explication de l'Arithmétique Binaire***. В системе счисления Лейбница были использованы цифры 0 и 1, как и в современной двоичной системе. Как человек, увлекающийся китайской культурой, Лейбниц знал о книге Перемен и заметил, что гексаграммы соответствуют двоичным числам от 0 до 111111.

Он восхищался тем, что это кодирование является свидетельством крупных китайских достижений в философской математике того времени.

Лейбниц также был изобретателем счетного устройства, которое производило арифметические операции в двоичной системе счисления.

Двоичная система счисления имеет основание 2, и, следовательно, две разных цифры - 0 и 1;

Восьмеричная - восемь разных цифр - 0, 1, 2, 3, 4, 5, 6, 7;

Шестнадцатеричная - шестнадцать цифр - десять арабских цифр от 0 до 9 и еще шесть символов -

A (цифра, изображающая десять),
B (цифра одиннадцать),
C (цифра двенадцать),
D (цифра тринадцать),
E (цифра четырнадцать),
F (цифра пятнадцать).

Проще всего сопоставить запись одних и тех же чисел в этих системах счисления можно с использованием таблицы, приведенной на следующем слайде

Система счисления			
10	*2*	*8*	*16*
0	0	0	0
1	1	1	1
2	1 0	2	2
3	1 1	3	3
4	1 0 0	4	4
5	1 0 1	5	5
6	1 1 0	6	6
7	1 1 1	7	7
8	1 0 0 0	1 0	8
9	1 0 0 1	1 1	9
10	1 0 1 0	1 2	A
11	1 0 1 1	1 3	B
12	1 1 0 0	1 4	C

Мы уже говорили о том, что современные цифровые ЭВМ все используют в качестве основной двоичную систему счисления. ***К ее достоинствам относится:***

- *простота выполнения арифметических и логических операций, что влечет за собой простоту устройств, реализующих эти операции;*

- *возможность использования аппарата алгебры логики (булевой алгебры) для анализа и синтеза операционных устройств ЭВМ.*

К неудобствам двоичной системы счисления относится необходимость перевода чисел из десятичной в двоичную и наоборот, а также то, что запись числа в двоичной системе громоздка (требует большего числа разрядов, чем привычная для человека десятичная). По этой и ряду других причин, кроме двоичной применяются восьмеричная и шестнадцатеричная системы счисления.

Совместное использование указанных систем обусловлено двумя причинами:

- *в восьмеричной и шестнадцатеричной системах любое число записывается более компактно, нежели двоичное;*
- *простотой преобразования из двоичной в восьмеричную (шестнадцатеричную) систему счисления и наоборот*.

Правила перевода из одной СС в другую СС

Правила перевода целых и дробных чисел не совпадают, поэтому приведем три правила перевода чисел из системы счисления с основанием **R** в систему счисления с основанием **Q**.

Правило 1. Перевод целых чисел

Для перевода целого числа **N**, представленного в СС с основанием **R** в СС с основанием **Q**, необходимо данное число делить на основание Q по правилам СС с основанием R до получения целого остатка, меньшего Q. Полученное целое снова необходимо делить на основание Q до получения нового целого остатка, меньшего Q, и т.д., до тех пор, пока последнее целое станет равно нулю. Число N в СС с основанием Q представляется в виде последовательности остатков деления на Q в порядке, обратном их получению.

Пример перевода числа 532 из десятичной СС в двоичную СС

532:2=266(остаток 0)

266:2=133(остаток 0)

133:2=66 (остаток 1)

66:2=33 (остаток 0)

33:2=16 (остаток 1)

16:2=8 (остаток 0)

8:2=4 (остаток 0)

4:2=2 (остаток 0)

2:2=1 (остаток 0)

1:2=0 (остаток 1)

Получаем число 1000010100

Перевод числа 1000010100 из двоичной СС в десятичную СС

Пронумеруем разряды:

9	*8*	*7*	*6*	*5*	*4*	*3*	*2*	*1*	*0*
1	**0**	**0**	**0**	**0**	**1**	**0**	**1**	**0**	**0**

Перевод:

$$1 \cdot 2^9 + 1 \cdot 2^4 + 1 \cdot 2^2 = 532$$

Пример перевода числа 532 из десятичной СС в восьмеричную СС

532:8=66(остаток 4)

66:8=8 (остаток 2)

8:8=1 (остаток 0)

1:8=0 (остаток 1)

Получаем число **1024**

Перевод числа 1024 из восьмеричной СС в десятичную СС

Пронумеруем разряды:

3	*2*	*1*	*0*
1	**0**	**2**	**4**

Перевод:

$$1 \cdot 8^3 + 2 \cdot 8^1 + 4 \cdot 8^0 = 532$$

Пример перевода числа 532 из десятичной СС в шестнадцатеричную СС

532:16=33(остаток 4)

33:16=2 (остаток 1)

2:16=0 (остаток 2)

Получаем число **214**

Перевод числа 214 из шестнадцатеричной СС в десятичную СС

Пронумеруем разряды:

2	*1*	*0*
2	**1**	**4**

Перевод:

$$2 \cdot 16^2 + 1 \cdot 16^1 + 4 \cdot 16^0 = 532$$

Общее правило перевода в десятичную СС

Если $a_{n-1}a_{n-2}\ldots a_1a_0$ запись целого числа в СС с основанием Q, то перевод в СС с снованием 10 осуществляется по формуле:

$$N_{(10)}=a_{n-1}\,Q^{n-1}+a_{n-2}\,Q^{n-2}+\ldots+a_1\,Q^1+a_0Q^0$$

Правило 2. Перевод правильной дроби

Перевод правильной дроби **D**, представленной в СС с основанием **R**, в СС с основанием **Q** заключается в последовательном умножении этой дроби на основание Q по правилам системы счисления с основанием R, причем перемножают только дробные части. Дробь D в СС с основанием Q представляется в виде последовательности целых частей произведений в порядке их получения. Умножение прекращают, когда дробная часть станет, равна нулю.

Для многих чисел указанный процесс умножения потенциально никогда не кончается. Поэтому он продолжается до тех пор, пока не будет получено необходимое число цифр дробной части. Количество последовательных произведений определяет количество цифр в полученном числе.

Пример перевода в двоичную СС правильной десятичной дроби 0,7243.

Основание исходной системы счисления R=10. Основание новой системы счисления Q=2.

Согласно приведенного правила исходное число 0,7243 надо умножать на 2 по правилам десятичной системы счисления.

Выполним серию умножений до получения, например, шести цифр в двоичном числе:
Искомые цифры дроби:

0,7243 * 2 = **1**,4486 1 (целая часть)
0,4486 * 2 = **0**,8972 0 (целая часть)
0,8972 * 2 = **1**,7944 1 (целая часть)
0,7944 * 2 = **1**,5888 1 (целая часть)
0,5888 * 2 = **1**,1776 1 (целая часть)
0,1776 * 2 = **0**,3552 0 (целая часть)
0,3552 * 2 = **0**,7104 0 (целая часть)

Искомое представление числа 0,7243 в двоичной системе счисления **0,101110**.

Обратите внимание, что ***для получения шести цифр дроби выполнено семь умножений***. Это связано с необходимостью выполнить округление, чтобы представить дробь заданной длины более точно.

Перевод правильной дроби 0,101110 из двоичной СС в десятичную СС

Пронумеруем разряды:

	1	*2*	*3*	*4*	*5*	*6*
0,	**1**	**0**	**1**	**1**	**1**	**0**

Перевод:

$$1 \cdot 2^{-1} + 1 \cdot 2^{-3} + 1 \cdot 2^{-4} + 1 \cdot 2^{-5} = 0{,}71875 \approx 0{,}7188$$

Исходное число было: 0,7243

Общее правило перевода

Если $0,a_1a_2 \ldots a_{n-1}a_n$ запись правильной дроби в СС с основанием Q, то перевод в СС с снованием 10 осуществляется по формуле:

$$D_{(10)} = a_1 Q^{-1} + a_2 Q^{-2} + \ldots + a_{n-1} Q^{1-n} + a_n Q^{-n}$$

Для достижения требуемой точности шести знаков явно недостаточно, кроме того данная дробь в двоичной системе СС может оказаться бесконечной.

Правило 3. Перевод неправильной дроби

Перевод неправильной дроби из одной системы счисления в другую осуществляется отдельно для целой и дробной части по правилам, изложенным выше.

Правило перевода из двоичной СС в восьмеричную СС

При переводе многоразрядного двоичного числа в восьмеричную форму поступают следующим образом: Исходное число разбивают на триады. При этом для целой части числа разбиение проводят от местонахождения запятой влево, а для дробной части - от этого же места вправо. Затем самая левая группа при необходимости дополняется незначащими нулями до образования триады, а самая правая группа только в дробной части дополняется нулями справа также до образования полной триады. После этого каждая триада заменяется соответствующей восьмеричной цифрой. Местоположение запятой сохраняется.

Пример:

Представить двоичное число

1101100,01111101

в форме восьмеричного.

Разобьем исходное число на группы по три цифры, приняв в качестве точки отсчета местоположение запятой (для наглядности между триадами поместим пробелы):

1 101 100 , 011 111 01

Теперь дополним до трех цифр нулями самую левую группу слева и самую правую группу справа

001 101 100 , 011 111 010

И, наконец, заменим каждую триаду соответствующей восьмеричной цифрой из таблицы.

Получим **154,372** $_{(8)}$

Правило перевода из восьмеричной СС в двоичную СС

При переводе многоразрядного числа каждую цифру исходного восьмеричного числа можно всегда представить точно тремя двоичными цифрами, взятыми из приведенной выше таблицы. При этом, если для записи соответствующей восьмеричной цифры в виде двоичной требуется менее трех двоичных цифр, двоичный эквивалент дополняется слева нулями. Таким образом, например, при записи четырехразрядного восьмеричного числа должно получиться двенадцатиразрядное двоичное. После окончания такого преобразования можно отбросить старшие для всего числа незначащие двоичные цифры.

Пример:

Преобразуем восьмеричное число

371,62.

Для этого запишем для каждой цифры соответствующую триаду:

3 → 011

7 → 111

1 → 001

6 → 110

2 → 010

Теперь можно записать число в двоичной форме (для наглядности между триадами поместим пробелы):

371,62 → 011 111 001 , 110 010

И, наконец, запишем полученное двоичное число так, как это принято в математике, без незначащих нулей, а также отбросив правые нули в дробной части числа:

371,62 → 11111001,11001

Правило перевода из двоичной СС в шестнадцатеричную СС

При переводе многоразрядного двоичного числа в шестнадцатеричную форму поступают следующим образом. Исходное число разбивают на тетрады. При этом для целой части числа разбиение проводят от местонахождения запятой влево, а для дробной части от этого же места вправо. Затем самая левая группа при необходимости дополняется незначащими нулями до образования тетрады, а самая правая группа только в дробной части дополняется нулями справа также до образования полной тетрады. После этого каждая тетрада заменяется соответствующей шестнадцатиричной цифрой из таблицы. Местоположение запятой сохраняется.

Пример:

Представим двоичное число

1101100,01111101

в форме шестнадцатеричного.

Разобьем исходное число на группы по четыре цифры, приняв в качестве точки отсчета местоположение запятой (для наглядности между тетрадами поместим пробелы):

110 1100 , 0111 1101

Теперь дополним до четырех цифр нулями слева самую левую группу:

0110 1100 , 0111 1101

И, наконец, заменим каждую тетраду соответствующей шестнадцатеричной цифрой:

0110 1100 , 0111 1101 → 6C,7D.

Шестнадцатеричная и восьмеричная системы счисления используются для более компактной и удобной записи двоичных чисел.

Правило перевода из шестнадцатеричной СС в двоичную СС

При переводе многоразрядного шестнадцатеричного числа в двоичную форму каждую цифру исходного числа заменяют точно группой из четырех двоичных цифр (***тетрадой*** двоичных цифр). Местоположение запятой сохраняется. В окончательной записи можно отбросить самые левые (незначащие) нули и самые правые нули дробной части.

Пример:

Преобразуем шестнадцатеричное число 6C,7D в двоичную форму.

Для этого запишем для каждой цифры соответствующую тетраду:

6 →0110

C→1100

7→ 0111

D→1101

Теперь можно записать число в двоичной форме (для наглядности между тетрадами поместим пробелы):

6C,7D→0110 1100 , 0111 1101

И, наконец, запишем полученное двоичное число так, как это принято в математике, без незначащих нулей:

6C,7D → 1101100,01111101

Математическая запись двоичных чисел

Положительное целое двоичное число	11011011
Отрицательное целое двоичное число	-11011011
Положительное дробное двоичное число	110,1101101
Отрицательное дробное двоичное число	-110,1101101

Закон продвижения единицы

0	0
1	1
2	10
3	11
4	100
5	101
6	110
7	111
8	1000
9	1001
10	1010
11	1011
12	1100
13	1101
14	1110
15	1111
16	10000
17	10001

Закон продвижения единицы состоит в следующем - для вычисления следующего двоичного числа, единицу ставим на место ближайшего 0 с конца (т.е. как бы продвигаем единицу), при этом все единицы стоящие за ним обращаются в 0.
Пользуясь этим законом , можно всегда вычислить следующее двоичное число.

Правила сложения, вычитания, умножения двоичных чисел

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

Сложение	Вычитание	Умножение
0+0=0	0-0=0	0*0=0
0+1=1	1-0=1	0*1=0
1+0=1	1-1=0	1*0=0
1+1=10	10-1=1	1*1=1

Пример сложения чисел 17 и 13 в десятичной и двоичной СС.

17		**10001**
+ 13	→	**+ 1101**
30		**11110**

Пример вычитания чисел 17 и 13 в десятичной и двоичной СС.

17		**10001**
- 13	→	**- 1101**
4		**0100**

Пример умножения чисел 13 и 17 в десятичной и двоичной СС.

13		**10001**
***17**		*** 1101**
91		**10001**
+13		**10001**
221		**10001**
		11011101

Таким образом, **операцию умножения** двоичных чисел в ЭВМ можно свести к операциям **сдвига** и **сложения**.

Арифметика цифровых вычислительных машин(ЦВМ)

Для того, чтобы более просто, и, следовательно, более экономично реализовать устройство **АЛУ** применяют несколько разных кодов чисел. Это связано с тем, что разные операции в ЭВМ более просто реализуются в разных кодах.
При выполнении арифметических операций в ЭВМ применяют ***прямой, обратный*** и ***дополнительный*** коды чисел.

В устройствах, реализующих операцию арифметического сложения двоичных чисел, операнды представляют числами определенной разрядности (одинаковой для обоих операндов). При этом неиспользуемые старшие разряды заполняются нулями.

В ЭВМ используются 8-,16-,32-, 64-разрядные числа.

Прямой код двоичного числа - это само двоичное число, в котором все цифры, изображающие его значение, записываются как в математической записи, а знак числа записывается двоичной цифрой 0(+), 1(-).

Примеры:

В примерах коды изображаются восемью цифрами.

Изображаемое число	Прямой код
+13	0000 1101
-13	1000 1101

Итак, прямой код почти не отличается от принятого в математике: для выявления абсолютной величины (модуля) числа, надо поменять цифру, обозначающую его знак на противоположную.

Однако применительно к операциям сложения и вычитания прямой код неудобен: правила счета для положительных и отрицательных чисел различаются.

Например, 13+(-7)=6. А вот что получим мы.

00001101

+ 10000111

10010100 (получили -20)

Прямой код используется для хранения чисел в памяти ЭВМ, а также при выполнении операций над положительными числами.

Чтобы построить более простые схемы **АЛУ** предложены и активно применяются ***обратный*** и ***дополнительный*** коды.

Обратный код положительного числа совпадает с прямым, а при записи отрицательного числа все его цифры, кроме цифры, изображающей знак числа, заменяются на противоположные (0 заменяется на 1, а 1 - на 0).

Пример:

Число	*Прямой код*	*Обратный код*
$13_{(10)}$	0000 1101	0000 1101
$-13_{(10)}$	10001101	1111 0010

Восстановить абсолютную величину (модуль) отрицательного числа в обратном коде также довольно просто – заменить все 0 на 1, а 1 на 0. В этом коде как к положительным, так и к отрицательным числам можно применять одни и те же правила, а операцию A-B можно заменить операцией сложения чисел A и “минус B”.

Например, 13+(-7)=6.

00001101

+ 11111000

100000101

00000101

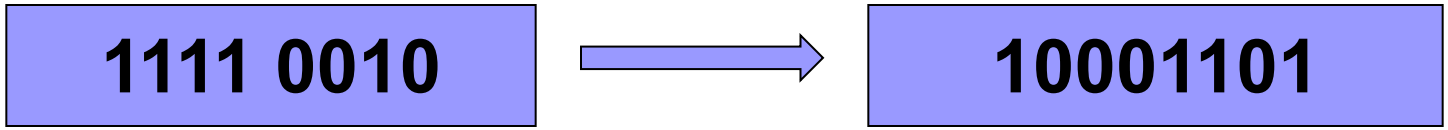
+ 1

00000110

получили 6

По правилу сложения чисел в обратном коде, при появлении 1 в дополнительном разряде, эта1 отбрасывается, а к числу прибавляется еще 1. Получаем $\mathbf{110_{(2)}=2^2+2=6_{(10)}}$

Для восстановления прямого кода отрицательного числа из обратного кода надо все цифры, кроме цифры, изображающей знак числа, заменить на противоположные.



Дополнительный код положительного числа совпадает с прямым, а код отрицательного числа образуется как результат увеличения на 1 его обратного кода.

Пример:

Число	*Прямой код*	*Обратный код*	*Дополнительный код*
$13_{(10)}$	0000 1101	0000 1101	0000 1101
$-13_{(10)}$	1000 1101	1111 0010	1111 0011

Для восстановления прямого кода числа из дополнительного нужно полностью повторить (и именно в том же порядке!) действия, которые использовались при переводе из прямого в дополнительный код: сначала все цифры, кроме цифры, изображающей знак, заменить на противоположные, а затем прибавить 1.

11110011 (дополнительный для -13)

10001100 (промежуточное действие)

10001101 (прямой для -13)

Сложение чисел в обратном и дополнительном кодах выполняется с использованием обычного правила арифметического сложения многоразрядных чисел. Общей для этих кодов особенностью (и очень удобной особенностью) является лишь то, что при поразрядном сложении чисел разряды, изображающие знаки чисел рассматриваются как равноправные разряды двоичного числа и участвуют в операции сложения.

Различие же обратного и дополнительного кодов, заключается в том, что делается с единицей, появляющейся в дополнительном разряде. Например, 13+(-7)=6 в дополнительном коде выглядит так:

00001101

+ 11111001

(1)00000110

При сложении чисел в дополнительном коде единица из дополнительного разряда игнорируется. Получаем $\mathbf{110_{(2)}=2^2+2=6_{(10)}}$

Например, 7-13=-6 в дополнительном коде выглядит так:

$$\begin{array}{r} 00000111 \\ +\ 11110011 \\ \hline 11111010 \end{array}$$

В результате получено отрицательное число в дополнительном коде. Переведем его в прямой:
11111010 , далее 10000101, далее 10000110.
Получили $-(2^2+2^1)=-6$

Основными достоинствами дополнительного кода является:

в нем единообразно реализуются операции сложения чисел разных знаков (алгебраическое сложение);

перевод из прямого кода в дополнительный и обратно, производится по одному и тому же алгоритму.

Немного истории

В ранних компьютерах не было заложено средств для автоматического перевода чисел из десятичной системы счисления в двоичную. Исходные данные, коды команд, адреса вводились в компьютер в двоичной системе, результаты программы также выводились в двоичном виде.

Первый серийный персональный компьютер Альтаир 8800

Выпущенный в **1975** г., он продавался в сборе за **397** дол., а в виде деталей, которые можно было получить по почте, за **297** дол.

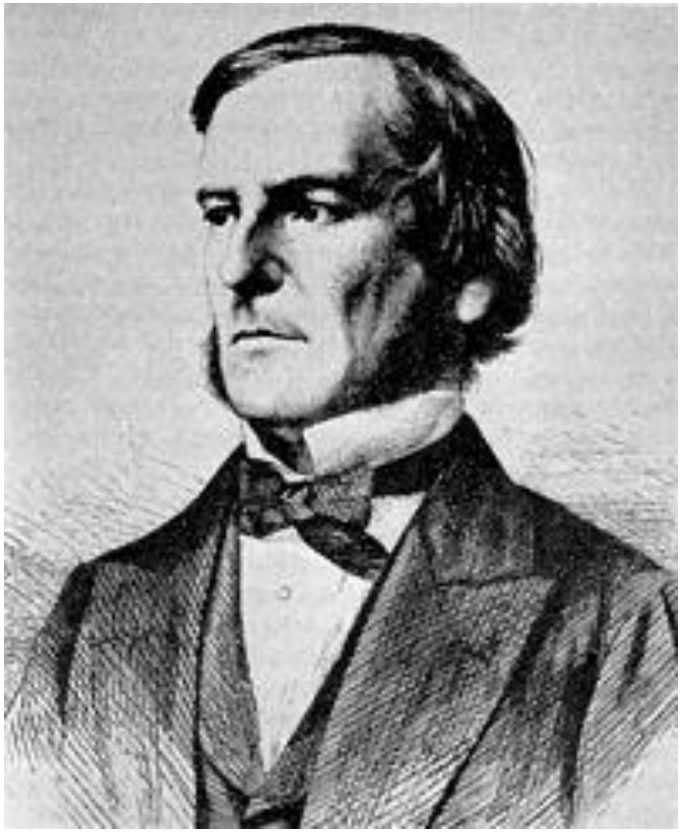
- В компьютере не было ни клавиатуры, ни дисплея, ни долговременной памяти. Весь объём **ОЗУ** составлял **256 байт**. Программы вводились переключением тумблеров на передней панели, а результаты считывались со светодиодных индикаторов.

Логические основы компьютера

Алгебра логики

Алгебра логики — это раздел математики, изучающий высказывания, рассматриваемые со стороны их логических значений (истинности или ложности) и логических операций над ними.

Алгебра логики возникла в середине **XIX** века в трудах английского математика ***Джорджа Буля***. Ее создание представляло собой попытку решать традиционные логические задачи алгебраическими методами.



Джордж Буль
англ. George Boole
Дата рождения:
2 ноября 1815
Дата смерти:
8 декабря 1864 (49 лет)
Место работы:
Королевский колледж в Корке

Логическое высказывание — это любое повествовательное предложение, в отношении которого можно однозначно сказать, ***истинно*** оно или ***ложно***.

Логические высказывания могут быть ***истинными*** или ***ложными***.

Так, например, предложение ***«6 — четное число»*** следует считать истинным высказыванием.

Предложение ***«Рим — столица Франции»*** – ложное высказывание.

Разумеется, не всякое предложение является логическим высказыванием. Высказыванием не является, например, предложение «***информатика — интересный предмет***». Для кого-то он интересный предмет, а у некоторых он вызывает только головную боль.

Данное высказывание использует слишком неопределённое понятие «***интересный предмет***».

Алгебра логики рассматривает любое высказывание только с одной точки зрения — является ли оно ***истинным*** или ***ложным***

Употребляемые в обычной речи слова и словосочетания **“не”, “и”, “или”, ”если..., то”, “тогда и только тогда”** и другие позволяют из уже заданных высказываний строить новые высказывания. Такие слова и словосочетания называются ***логическими связками.***

Высказывания, образованные из других высказываний с помощью логических связок, называются ***составными*****.** Высказывания, не являющиеся составными, называются ***элементарными*****.**

Так, например, из элементарных высказываний «***Сидоров — студент***», «***Сидоров — спортсмен***» при помощи связки «***и***» можно получить составное высказывание «***Сидоров— студент и спортсмен***», понимаемое как «***Сидоров— студент, занимающийся спортом***».

При помощи связки «***или***» из этих же высказываний можно получить составное высказывание «***Сидоров — студент или спортсмен***», понимаемое в алгебре логики как «***Сидоров или студент, или спортсмен, или и студент и спортсмен одновременно***».

Истинность или ложность получаемых таким образом составных высказываний зависит от истинности или ложности элементарных высказываний.

Чтобы обращаться к логическим высказываниям, им назначают имена.

Например:

Пусть через **А** обозначено высказывание ***«Сидоров – студент»,*** а через **В** — высказывание ***«Сидоров – спортсмен».*** Тогда составное высказывание ***«Сидоров — студент и спортсмен»,*** можно кратко записать как **А и В**. Здесь «**и**» — логическая связка, **А, В** — логические переменные, которые могут принимать только два значения **«истина»** или **«ложь».**

Каждая логическая связка рассматривается как операция над логическими высказываниями и имеет свое название и обозначение:

«НЕ» - операция, выражаемая словом *“не”,* называется **отрицанием** и обозначается чертой над высказыванием (или знаком ¬). Высказывание ¬А истинно, когда А ложно, и ложно, когда А истинно.

Пример. «*Луна — спутник Земли»* (А);

«*Луна — не спутник Земли»* (¬А).

«И» - операция, выражаемая связкой *“и”*, называется **конъюнкцией** (лат. conjunctio — соединение) или логическим умножением и обозначается **“∧”** (может также обозначаться знаками **“·”** или **&**).

Высказывание **A ∧ B** истинно тогда и только тогда, когда оба высказывания **A** и **B** истинны.

Пусть A=«12 четное число»,
B=«12 делится на 3».

Тогда **A ∧ B**=«12 четное число и делится на 3» - истинно, т.к. A - истинно, B – истинно.

«ИЛИ» - операция, выражаемая связкой “*или”,* называется **дизъюнкцией** (лат. disjunctio — разделение) или логическим сложением и обозначается знаком **v** (или +).

Высказывание **A v B** ложно тогда и только тогда, когда оба высказывания A и B ложны.

Пусть A=«12 делится на 5»,
B=«12 делится на 7».

Тогда **A v B** = «12 делится на 5 или на 7» - ложно, т.к. A - ложно, B – ложно.

«ЕСЛИ-ТО» - операция, выражаемая связками *“если ..., то”, “из ... следует”, “… влечет …”,* называется **импликацией** (лат. *implico* — тесно связаны) и обозначается знаком → . Высказывание A → B ложно тогда и только тогда, когда **A** истинно, а **B** ложно, то есть из истины не может следовать ложь.

«РАВНОСИЛЬНО» - операция, выражаемая связками “*тогда и только тогда*”, “*необходимо и достаточно*”, “... *равносильно* ...”, называется **эквиваленцией** и обозначается знаком ↔ или **~.** Высказывание истинно тогда и только тогда, когда логические значения **A** и **B** совпадают.

Например, высказывание *"24 делится на 2 тогда и только тогда, когда 4 делится на 2» -* истинно.

Логическая формула

С помощью логических переменных и символов логических операций любое высказывание можно формализовать, то есть заменить логической формулой.

Логической формулой называется -

1. Всякая логическая переменная и символы “истина“, “T”, “1” и "ложь“, “F”, “0” - формулы.

2. Если A и B - формулы, то $\neg A$, $A \wedge B$, $A \vee B$, $A \rightarrow B$, $A \leftrightarrow B$ - формулы.

3. Никаких других формул в алгебре логики нет.

В п.1 определены элементарные формулы;

в п.2 даны правила образования из любых данных формул новых формул. В логических формулах используются круглые скобки, для указания приоритета.

Примеры логических формул:

$$\mathbf{F},\quad \mathbf{0},\quad \mathbf{A},\ \mathbf{B},\ \mathbf{A} \wedge \mathbf{B},\ (\mathbf{B} \vee \mathbf{A}) \rightarrow \mathbf{A}$$

Если высказывания A и B могут принимать различные логические значения, то их заменяют переменными X и Y. От переменных X и Y можно описать логические функции одной переменной F(x), двух переменных F(x,y). Правые части которых являются логическими формулами.

Например:

$F(x)=x \vee \neg x$ (Был или не был?)

$F(x,y)=x \wedge y \rightarrow x$ (Сидоров студент и спортсмен, следовательно Сидоров – студент)

Логическая функция может принимать только два значения “истина” (“1”) или “ложь” (“0”).

“истина” (“1”) и “ложь” (“0”) в алгебре логики образуют множество значений логической функции и называются ***логическими константами.***

Элементарные логические функции одной переменной

Функции F**0**(x) = 0 и F**3**(x) = 1 являются константами (функции не изменяются при изменении аргумента).

Функция F**1**(x) = x повторяет значение аргумента x.

Функция F**2**(x)= ¬x называется ***отрицанием переменной*** или ***инверсией***.

Любая логическая функция может быть задана с помощью ***таблицы истинности***.
Например, таблица истинности для функции F**2**(x)= ¬x .

“НЕ”

X	F2(X)
0	1
1	0

В таблице задаются все возможные значения аргументов функции – слева, а соответствующие им значения функции – справа. В таблице могут быть столбцы для вычисления промежуточных значений.

Таблица истинности - это табличное представление логической функции (элементарной или составной).

F(x,y)=(x v y) ∧ ¬x

x	y	(x v y)	¬x	(x v y) ∧ ¬x
0	0	0	1	0
0	1	1	1	1
1	0	1	0	0
1	1	1	0	0

Элементарные логические функции двух переменных

Элементарных функций двух переменных x_1 и x_2 всего 16. Важными из них являются функции

$$F1(x_1, x_2)= x_1 \wedge x_2 \quad \text{и} \quad F7(x_1, x_2)=x_1 \vee x_2.$$

Таблица истинности $F1(x_1, x_2) = x_1 \wedge x_2$

“И”

x1	x2	x1 ∧ x2
0	0	0
0	1	0
1	0	0
1	1	1

Таблица истинности $F7(x_1, x_2) = x_1 \vee x_2$

“ИЛИ”

x1	x2	x1 v x2
0	0	0
0	1	1
1	0	1
1	1	1

Таблица истинности $F11(x_1, x_2) = x_1 \rightarrow x_2$

“Импликация”

x1	x2	x1→ x2
0	0	1
0	1	1
1	0	0
1	1	1

Таблица истинности $F14(x_1, x_2) = x_1 \leftrightarrow x_2$

“Эквиваленция”

x1	x2	x1↔ x2
0	0	1
0	1	0
1	0	0
1	1	1

С помощью этих функций можно представить (аналитически выразить) любую сколь угодно сложную логическую функцию:

$$F(x,y)=(x \vee y) \wedge \neg x$$

x	y	(x v y)	¬x	(x v y) ∧ ¬x
0	0	0	1	0
0	1	1	1	1
1	0	1	0	0
1	1	1	0	0

Цветом выделены столбцы для вычисления промежуточных значений функции.
Функция принимает истинное значение, когда **x** – “ложь”, а **y** “истина”.

А с помощью таблиц истинности можно получить все значения функции и проверить эквивалентность функций.

Пример:

F(x)=x v ¬x

равносильна константной функции

F(x)=1

x	F(x)
0	1
1	1

Очень важными для вычислительной техники являются логические операции **исключающее ИЛИ** (**неравнозначность**, **сложение по модулю два**), обозначаемая символом $\oplus$, а так же штрих Шеффера |, и стрелка Пирса ↓.

Таблица истинности для **«исключающего ИЛИ».**

x	y	x ⊕ y
0	0	0
0	1	1
1	0	1
1	1	0

Таблица истинности для **«штрих Шеффера».**

x	y	x | y
0	0	1
0	1	1
1	0	1
1	1	0

Таблица истинности для **«стрелка Пирса ↓».**

x	y	x ↓ y
0	0	1
0	1	0
1	0	0
1	1	0

Упрощение логических выражений

В алгебре логики выполняются следующие основные законы, позволяющие производить тождественные преобразования логических выражений:

Аксиомы

1. $\bar{\bar{x}} = x$, инволютивность отрицания, закон снятия двойного отрицания
2. $x \vee \bar{x} = 1$
3. $x \vee 1 = 1$
4. $x \vee x = x$
5. $x \vee 0 = x$
6. $x \wedge \bar{x} = 0$
7. $x \wedge x = x$
8. $x \wedge 0 = 0$
9. $x \wedge 1 = x$

Свойства логических операций

1. Коммутативность: $x \circ y = y \circ x$, $\circ \in \{\&, \vee, \oplus, \sim, |, \downarrow\}$.
2. Идемпотентность: $x \circ x = x$, $\circ \in \{\&, \vee\}$.
3. Ассоциативность: $(x \circ y) \circ z = x \circ (y \circ z)$, $\circ \in \{\&, \vee, \oplus, \sim\}$.
4. Дистрибутивность конъюнкций и дизъюнкции относительно дизъюнкции, конъюнкции и суммы по модулю два соответственно:
   - $x \wedge (y \vee z) = x \wedge y \vee x \wedge z$,
   - $x \vee y \wedge z = (x \vee y) \wedge (x \vee z)$,
   - $x \wedge (y \oplus z) = x \wedge y \oplus x \wedge z$.
5. Законы де Морган:
   - $\overline{x \wedge y} = \bar{x} \vee \bar{y}$,
   - $\overline{x \vee y} = \bar{x} \wedge \bar{y}$.
6. Законы поглощения:
   - $x \wedge (x \vee y) = x$,
   - $x \vee x \wedge y = x$.
7. Другие (1):
   - $x \wedge \bar{x} = x \wedge 0 = x \oplus x = 0$.
   - $x \vee \bar{x} = x \vee 1 = x \sim x = x \to x = 1$.
   - $x \vee x = x \wedge x = x \wedge 1 = x \vee 0 = x \oplus 0 = x$.
   - $x \oplus 1 = x \to 0 = x \sim 0 = x \mid x = x \downarrow x = \bar{x}$.
   - $\bar{\bar{x}} = x$, инволютивность отрицания, закон снятия двойного отрицания.
8. Другие (2):
   - $x \oplus y = x \wedge \bar{y} \vee \bar{x} \wedge y = (x \vee y) \wedge (\bar{x} \vee \bar{y})$.
   - $x \sim y = \overline{x \oplus y} = 1 \oplus x \oplus y = x \wedge y \vee \bar{x} \wedge \bar{y} = (x \vee \bar{y}) \wedge (\bar{x} \vee y)$.
   - $x \to y = \bar{x} \vee y = x \wedge y \oplus x \oplus 1$.
   - $x \vee y = x \oplus y \oplus x \wedge y$.
9. Другие (3) (Дополнение законов де Морган):
   - $x \mid y = \overline{x \wedge y} = \bar{x} \vee \bar{y}$.
   - $x \downarrow y = \overline{x \vee y} = \bar{x} \wedge \bar{y}$.

Существуют методы упрощения логической функции: например, Карта Карно, метод Куайна - Мак-Класки

Решение логических задач

Если логическую задачу удается формализовать. То её можно решить с помощью алгебры логики.

Задача 1.

Трое друзей, болельщиков автогонок "Формула-1", спорили о результатах предстоящего этапа гонок.

— Вот увидишь, Шумахер не придет первым, — сказал Джон. Первым будет Хилл.

— Да нет же, победителем будет, как всегда, Шумахер, — воскликнул Ник. — А об Алези и говорить нечего, ему не быть первым.

Питер возмутился:

— Хиллу не видать первого места.

По завершении этапа гонок оказалось, что каждое из предположений двоих друзей подтвердилось, а предположение третьего оказалось неверным. ***Кто выиграл этап гонки?***

Решение. Введем обозначения для логических высказываний:

Ш — победит Шумахер;
Х — победит Хилл;
А — победит Алези.

Формализуем высказывания каждого из друзей:

$$\text{Джон: } \overline{\text{Ш}} \wedge \text{Х} \quad \text{Ник: } \text{Ш} \wedge \overline{\text{А}} \quad \text{Питер: } \overline{\text{Х}}$$

Учитывая то, что предположения двух друзей подтвердились, а предположения третьего неверны объединяем высказывания друзей связкой «и», а гипотезы связкой «или». Запишем и упростим логическое высказывание:

$$(\overline{\text{Ш}} \wedge \text{Х}) \wedge (\text{Ш} \wedge \overline{\text{А}}) \wedge \overline{\overline{\text{Х}}} \vee (\overline{\text{Ш}} \wedge \text{Х}) \wedge \overline{(\text{Ш} \wedge \overline{\text{А}})} \wedge \overline{\text{Х}} \vee \overline{(\overline{\text{Ш}} \wedge \text{Х})} \wedge (\text{Ш} \wedge \overline{\text{А}}) \wedge \overline{\text{Х}}$$
$$= (\overline{\text{Ш}} \wedge \text{Ш} \wedge \text{Х} \wedge \text{Х} \wedge \overline{\text{А}}) \vee \left(\overline{\text{Ш}} \wedge \text{Х} \wedge \overline{\text{Х}} \wedge (\overline{\text{Ш}} \vee \text{А})\right) \vee \left((\text{Ш} \vee \overline{\text{Х}}) \wedge \text{Ш} \wedge \overline{\text{А}} \wedge \overline{\text{Х}}\right)$$
$$= 0 \vee 0 \vee \text{Ш} \wedge \overline{\text{А}} \wedge \overline{\text{Х}} = \text{Ш} \wedge \overline{\text{А}} \wedge \overline{\text{Х}}$$

Высказывание истинно только при Ш=1, А=0, Х=0.
Ответ. Победителем этапа гонок стал Шумахер.

Решение логических задач методом рассуждений

Сводится к составлению различного вида таблиц и нахождению противоречий.

	Шумахер	**Хилл**	**Алези**
Джон	0	1	
Ник	1		0
Питер		0	

Рассмотрим все возможные пары угадавших исход гонок: **ДН, ДП, НП.**
ДН – противоречие: «Шумахер не мог победить и проиграть одновременно»;
ДП – противоречие: «Хилл не мог победить и проиграть одновременно»;
НП – противоречий нет, следовательно победил Шумахер.

Задача 2.
Виновник ночного дорожно-транспортного происшествия скрылся с места аварии.
Первый из опрошенных свидетелей сказал работникам ГАИ, что это были “Жигули”, первая цифра номера машины — единица.
Второй свидетель сказал, что машина была марки “Москвич”, а номер начинался с семёрки.
Третий свидетель заявил, что машина была иностранная, номер начинался не с единицы.
При дальнейшем расследовании выяснилось, что каждый из свидетелей правильно указал либо только марку машины, либо только первую цифру номера.
Какой марки была машина и с какой цифры начинался номер?

Решение. Введем обозначения для логических высказываний:
Ж – были жигули;
М – был москвич;
N1 – номер начинался с 1;
N7 – номер начинался с 7.
Формализуем показания каждого из свидетелей с учетом того, что при дальнейшем расследовании выяснилось, что каждый из свидетелей правильно указал либо только марку машины, либо только первую цифру номера.

Первый свидетель $(Ж \wedge \overline{N1}) \vee (\overline{Ж} \wedge N1)$

Второй свидетель $(М \wedge \overline{N7}) \vee (\overline{М} \wedge N7)$

Третий свидетель $(\overline{Ж} \wedge \overline{М} \wedge \overline{\overline{N1}}) \vee (\overline{\overline{Ж} \wedge \overline{М} \wedge N1})$

Объединяем показания связкой «и», так как они верны одновременно.

$$(Ж \wedge \overline{N1}) \vee (\overline{Ж} \wedge N1) \wedge (М \wedge \overline{N7}) \vee (\overline{М} \wedge N7) \wedge (\overline{Ж} \wedge \overline{М} \wedge \overline{\overline{N1}}) \vee (\overline{\overline{Ж} \wedge \overline{М} \wedge N1})$$

$$(Ж \wedge \overline{N1}) \vee (\overline{Ж} \wedge N1) \wedge (M \wedge \overline{N7}) \vee (\overline{M} \wedge N7) \wedge \left(\overline{Ж} \wedge \overline{M} \wedge \overline{\overline{N1}}\right) \vee \left(\overline{\overline{Ж} \wedge \overline{M}} \wedge \overline{N1}\right)$$

$$= \overline{\overline{Ж \wedge \overline{N1}} \wedge \overline{\overline{Ж} \wedge N1} \wedge M \wedge \overline{N7} \wedge \overline{M} \wedge N7 \wedge (\overline{Ж \vee M} \wedge N1)} \vee (Ж \vee M \wedge \overline{N1})$$

$$= \overline{(\overline{Ж} \vee N1) \wedge (Ж \vee \overline{N1}) \wedge (\overline{M} \vee N7) \wedge (M \vee \overline{N7}) \wedge \left(\overline{\overline{Ж \vee M} \wedge N1}\right) \wedge \left(Ж \vee M \wedge \overline{N1}\right)}$$

$$= \overline{Ж \leftrightarrow N1} \wedge \overline{M \leftrightarrow N7} \wedge \overline{(Ж \vee M) \leftrightarrow N1}$$

Для решения задачи необходимо найти при каких Ж, M, N1, N7 логическое выражение принимает истинное значение.

$$\overline{Ж \leftrightarrow N1} \wedge \overline{M \leftrightarrow N7} \wedge \overline{(Ж \vee M) \leftrightarrow N1} = \mathbf{1}$$

Выражение истинно, когда все три операнда истинны, а это возможно только при следующих значениях Ж, M, N1, N7. Первые два набора противоречат условию задачи.

Ж	N1	M	N7	$\overline{Ж \leftrightarrow N1}$	$\overline{M \leftrightarrow N7}$	$Ж \vee M$	$\overline{(Ж \vee M) \leftrightarrow N1}$	$\overline{Ж \leftrightarrow N1} \wedge \overline{M \leftrightarrow N7} \wedge \overline{(Ж \vee M) \leftrightarrow N1}$
1	0	1	0					
0	1	0	1					
1	0	0	1	1	1	1	1	1
0	1	1	0	1	1	1	0	0

Самая сложная логическая задача

Самая сложная логическая задача — название логической задачи, предложенной американским философом и логиком Джорджем Булосом в итальянской газете «la Repubblica» в 1992 году:

Есть три бога: A, B и C, которые являются богами истины, лжи и случая в произвольном порядке. Бог истины всегда говорит правду, бог лжи — всегда обманывает, бог случая может говорить и правду, и ложь в произвольном порядке. Требуется определить богов, задав 3 вопроса, на которые можно ответить «да» или «нет». Каждый вопрос задаётся только одному богу. Боги понимают язык, но отвечают на своём языке, в котором есть 2 слова «da» и «ja», причём неизвестно, какое слово обозначает «да», а какое «нет».

Источник Википедия

Какая связь между алгеброй логики и двоичным кодированием?

Математический аппарат алгебры логики очень удобен для описания того, как функционируют аппаратные средства компьютера, поскольку основной системой счисления в компьютере является двоичная, в которой используются цифры 1 и 0, а значений логических переменных тоже два: “1” и “0”.

Из этого следует два вывода:

- одни и те же устройства компьютера могут применяться для обработки и хранения как числовой информации, представленной в двоичной системе счисления, так и логических переменных;
- на этапе конструирования аппаратных средств алгебра логики позволяет значительно упростить логические функции, описывающие функционирование схем компьютера, и, следовательно, уменьшить число элементарных логических элементов, входящих в состав АЛУ.

Что такое логический элемент компьютера?

Логический элемент компьютера — это часть электронной логической схемы, которая реализует элементарную логическую функцию.

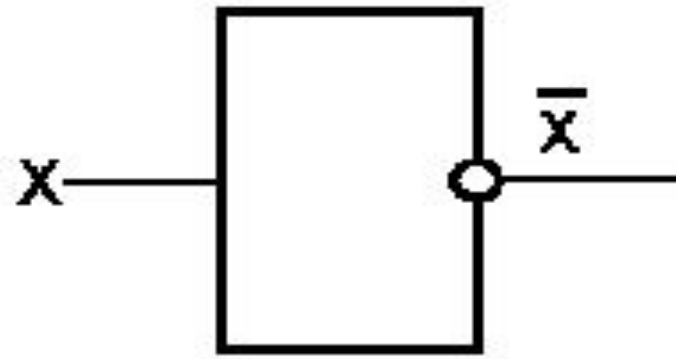
Логическими элементами компьютеров являются электронные схемы **И, ИЛИ, НЕ, И-НЕ, ИЛИ-НЕ, ИСКЛЮЧАЮЩЕЕ ИЛИ, ИСКЛЮЧАЮЩЕЕ ИЛИ-НЕ** (называемые также вентилями).

Работу логических элементов (вентилей) 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.
Рассмотрим несколько схем.

Схема «НЕ»

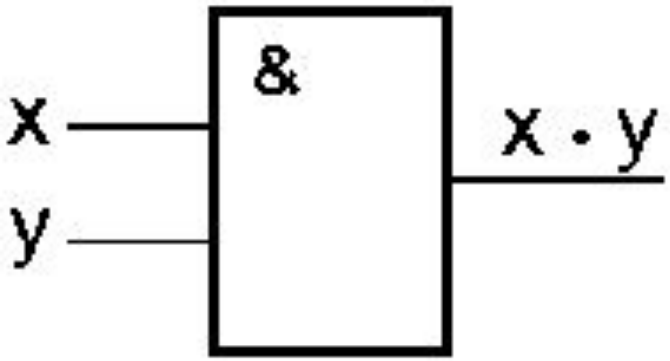
Схема «**НЕ**» (инвертор) реализует операцию отрицания. Связь между входом **x** этой схемы и выходом **z** можно записать соотношением $z = \overline{x}$, где $\overline{x}$ читается как **"не x"** или **"инверсия x".**



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

Схема «И»

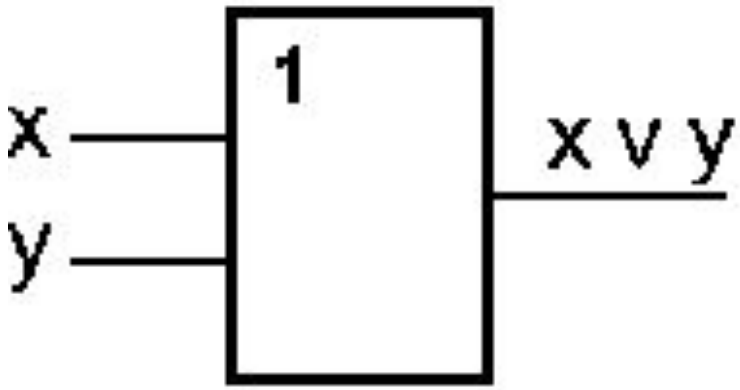
Схема «И» реализует конъюнкцию двух или более логических значений. Условное обозначение на структурных схемах схемы «И» с двумя входами представлено на рисунке.



X	У	X ● У
0	0	0
0	1	0
1	0	0
1	1	1

Схема «ИЛИ»

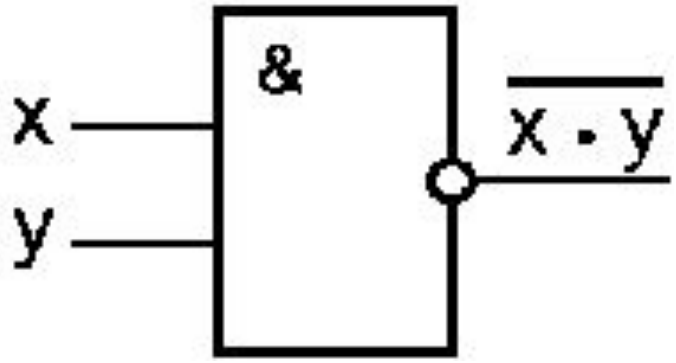
Схема «ИЛИ» реализует дизъюнкцию двух или более логических значений. Когда хотя бы на одном входе схемы «ИЛИ» будет единица, на её выходе также будет единица.



x	y	x v y
0	0	0
0	1	1
1	0	1
1	1	1

Схема «И—НЕ»

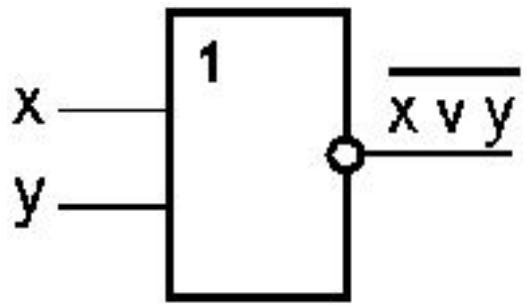
Схема «**И—НЕ**» состоит из элемента «**И**» и инвертора и осуществляет отрицание результата схемы «**И**». Связь между выходом **z** и входами **x** и **y** схемы записывают следующим образом: $z=x \cdot y$, где $x \cdot y$ читается как **"инверсия x и y".**



X	У	$\overline{X \bullet У}$
0	0	1
0	1	1
1	0	1
1	1	0

Схема «ИЛИ—НЕ»

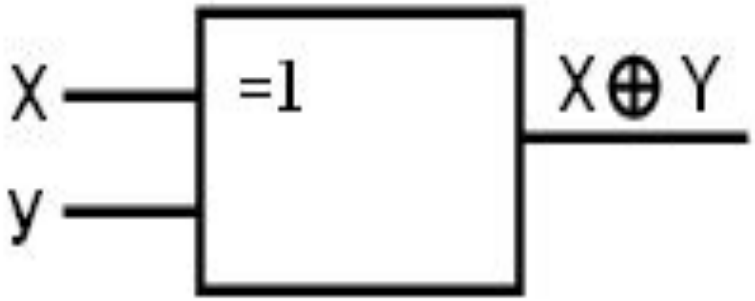
Схема «**ИЛИ—НЕ»** состоит из элемента «**ИЛИ»** и инвертора и осуществляет отрицание результата схемы «**ИЛИ».** Связь между выходом z и входами **x** и **y** схемы записывают следующим образом: $z=x \vee y$, где $\overline{x \vee y}$, читается как **"инверсия x или y ".**



x	y	$\overline{x \vee y}$
0	0	1
0	1	0
1	0	0
1	1	0

Схема «ИСКЛЮЧАЮЩЕЕ ИЛИ»

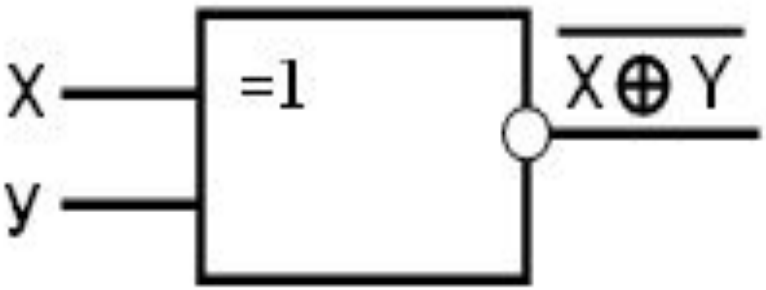
Схема «**ИСКЛЮЧАЮЩЕЕ ИЛИ**» реализует схему сложение по модулю 2. Связь между выходом **z** и входами **x** и **y** схемы записывают следующим образом: $z=x \oplus y$.



x	y	x⊕ y
0	0	0
0	1	1
1	0	1
1	1	0

Схема «ИСКЛЮЧАЮЩЕЕ ИЛИ-НЕ»

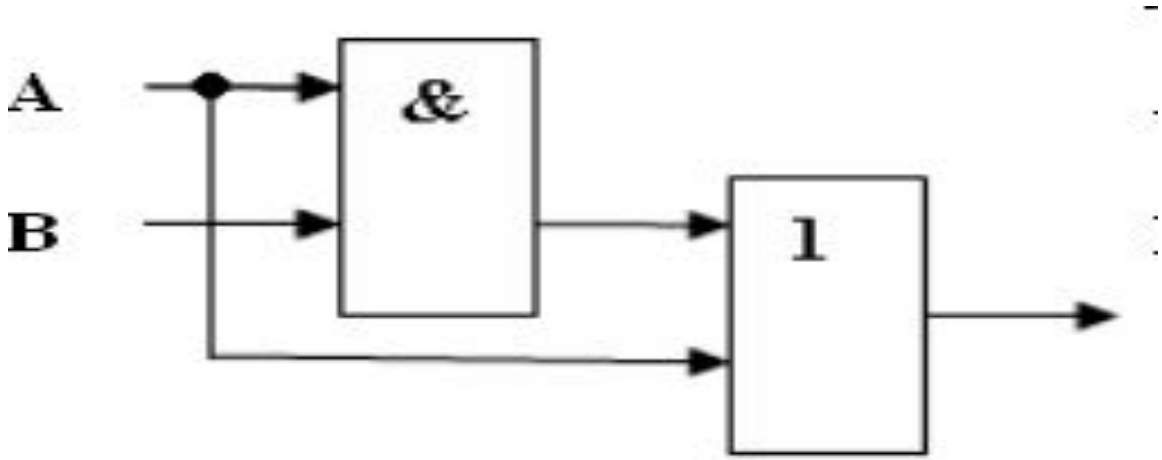
Схема «**ИСКЛЮЧАЮЩЕЕ ИЛИ-НЕ»** состоит из элемента «**ИСКЛЮЧАЮЩЕЕ ИЛИ»** и инвертора и осуществляет отрицание результата схемы «**ИСКЛЮЧАЮЩЕЕ ИЛИ».** Связь между выходом **z** и входами **x** и **y** схемы записывают следующим образом: $z=\overline{x \oplus y}$.



x	y	$\overline{x \oplus y}$
0	0	1
0	1	0
1	0	0
1	1	1

Примеры схем и соответствующие им таблицы истинности

Из вентилей составляют более сложные схемы, которые позволяют выполнять сложные арифметические операции



Данной логической схеме соответствует логическое выражение F=A & B v A.

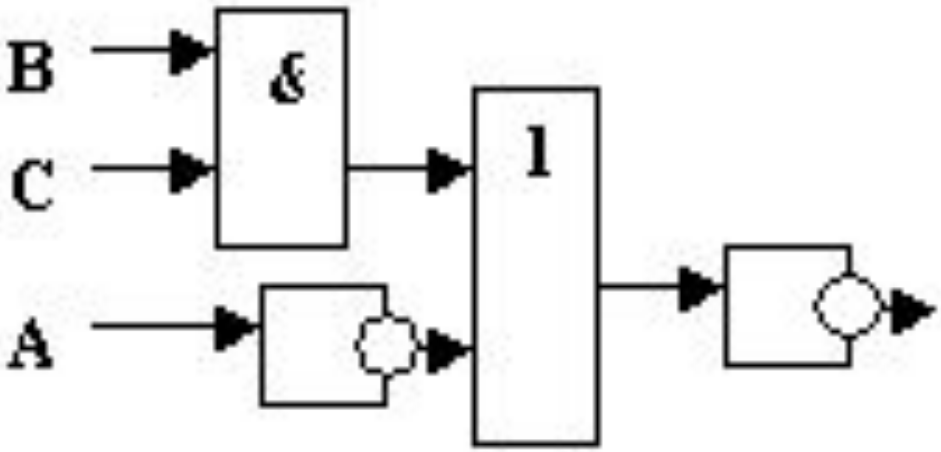
и таблица истинности:

A	B	F=A & B v A
0	0	0
0	1	0
1	0	1
1	1	1

Алгоритм построения логической схемы по логическому выражению

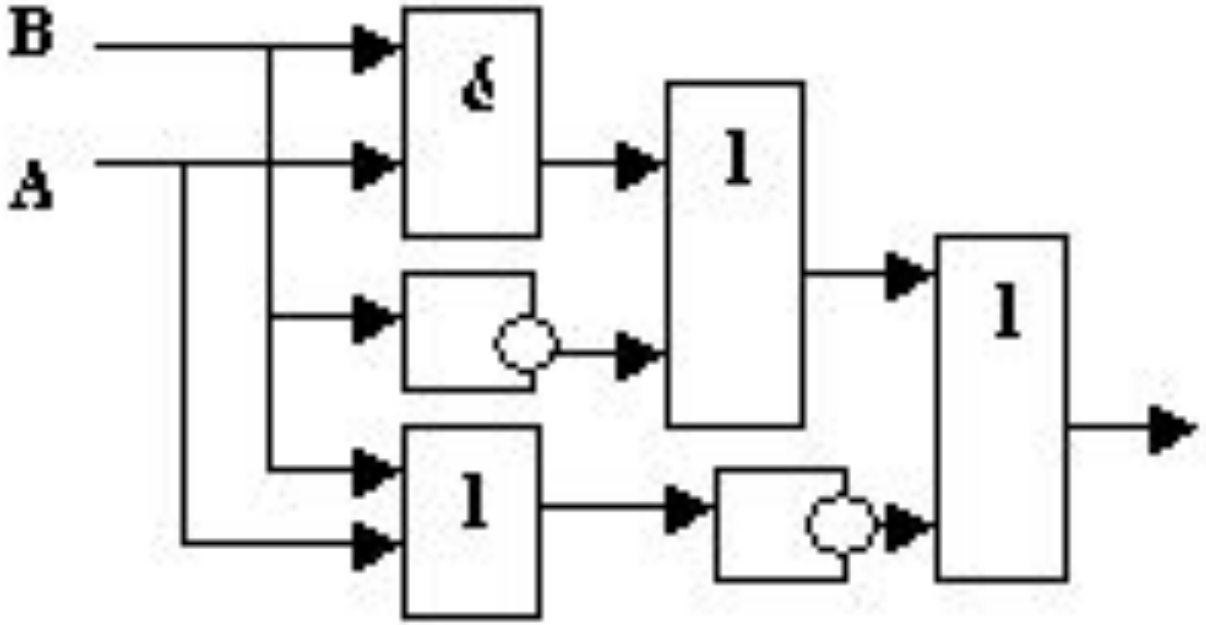
1. Определить число логических переменных.
2. Определить количество базовых логических операций и их порядок.
3. Изобразить для каждой логической операции соответствующий ей вентиль.
4. Соединить вентили в порядке выполнения логических операций.

Пример. Составить логическое выражение по схеме:



Ответ: $\overline{(B \wedge C) \vee \overline{A}}$

Составьте логическое выражение?



$$((B \wedge A)\vee\overline{B})\vee\overline{(B\vee A)}$$

Значения A и B хранятся на триггерах, они несут один бит информации 0 или 1.

Каждый разряд двоичного числа – это часть электронной схемы - **триггер**.

Триггер — это электронная схема, широко применяемая в регистрах компьютера для надёжного запоминания одного разряда двоичного кода. Триггер имеет два устойчивых состояния, одно из которых соответствует двоичной единице, а другое — двоичному нулю.

Сумматор — это электронная логическая схема, выполняющая суммирование двоичных чисел.

Сумматор служит, прежде всего, центральным узлом арифметико-логического устройства (АЛУ) компьютера.

Многоразрядный двоичный сумматор, предназначенный для сложения многоразрядных двоичных чисел, представляет собой комбинацию одноразрядных сумматоров

Рассмотрим схему одноразрядного сумматора.

При сложении чисел A и B в одном *i*-ом разряде приходится иметь дело с тремя цифрами:

1. цифра a_i первого слагаемого;

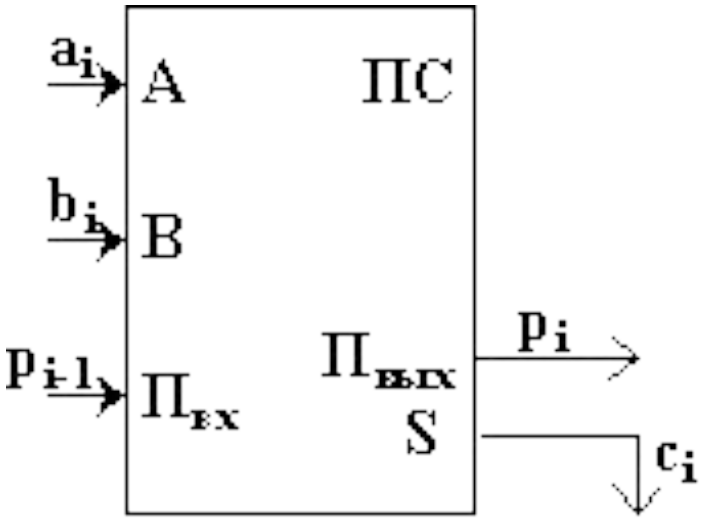
2. цифра b_i второго слагаемого;

3. перенос p_{i-1} из младшего разряда.

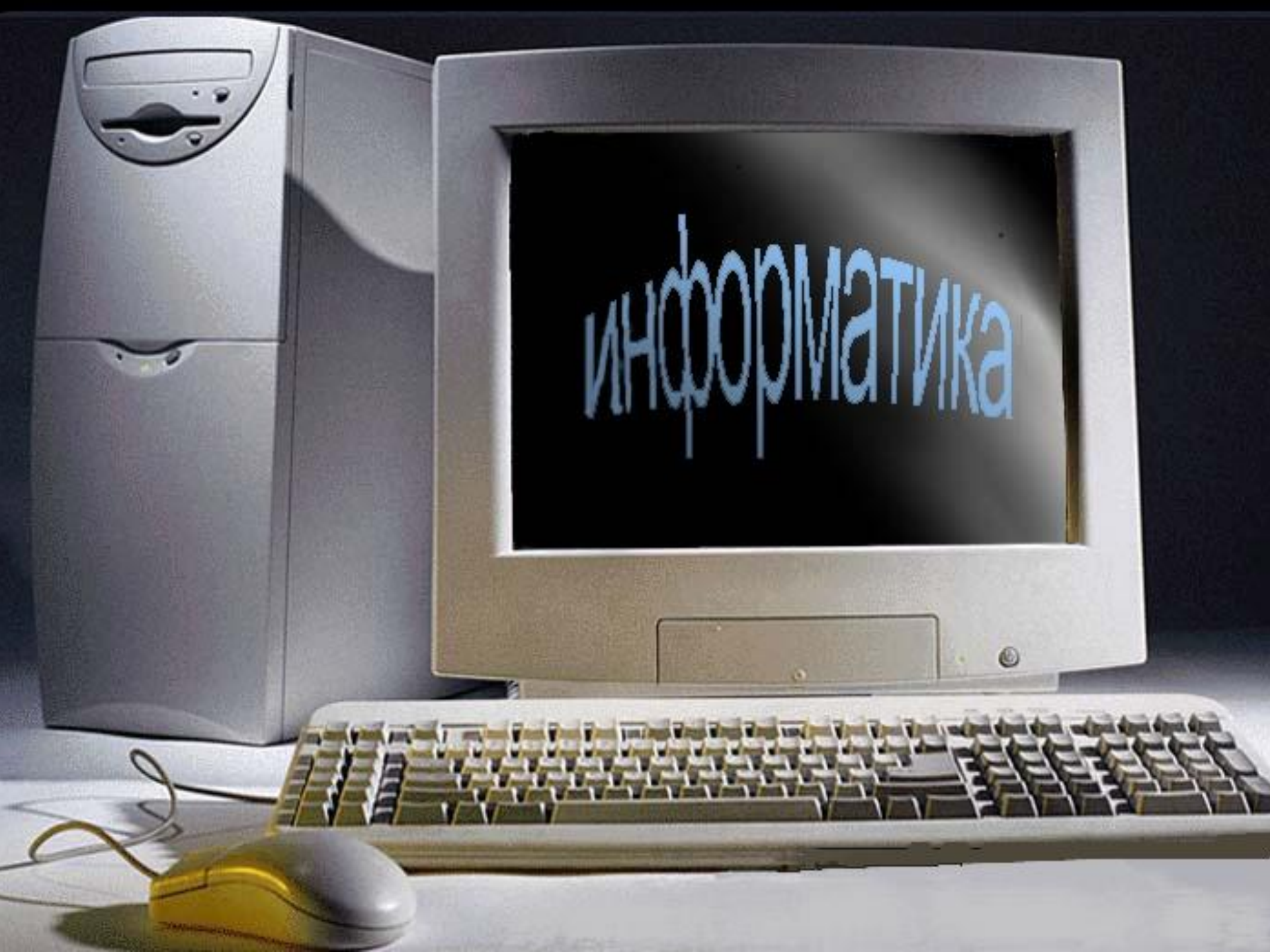
В результате сложения получаются две цифры:

1. цифра c_i для суммы;

2. перенос p_i из данного разряда в старший.



Таким образом, ***одноразрядный двоичный сумматор*** есть устройство с тремя входами и двумя выходами.
Если требуется складывать двоичные слова длиной два и более бит, то можно использовать последовательное соединение таких сумматоров, причём для двух соседних сумматоров выход переноса одного сумматора является входом для другого.


информатика