Оглавление

Двоичная система счисления

В двоичной (binary) системе счисления всего две цифры, называемые **двоичными** (binary digits). Сокращение этого наименования привело к появлению термина **бит**, ставшего названием разряда двоичного числа. Веса разрядов в двоичной системе изменяются по степеням двойки. Поскольку вес каждого разряда умножается либо на 0, либо на 1, то в результате значение числа определяется как сумма соответствующих значений степеней двойки. Если какой-либо разряд двоичного числа равен 1, то он называется значащим разрядом. Запись числа в двоичном виде намного длиннее записи в десятичной системе счисления.

Арифметические действия, выполняемые в двоичной системе, подчиняются тем же правилам, что и в десятичной системе. Только в двоичной системе перенос единиц в старший разряд возникает чаще, чем в десятичной. Вот как выглядит таблица сложения в двоичной системе:

0 + 0 = 0 0 + 1 = 1
1 + 0 = 1 1 + 1 = 10 (перенос в старший разряд)

Таблица умножения для двоичных чисел еще проще:
0 * 0 = 0 0 * 1 = 0 1 * 0 = 0 1 * 1 = 1

Пример выполнения операции сложения в двоичной системе счисления:

1 1 1
1 0 1 1$_2$
+ 1 1 0$_2$
---
1 0 0 0 1$_2$

Красным цветом показан перенос из младших разрядов в старшие

Для проверки правильности выполнения операции переведем все три числа из двоичной системы в 10-ую:

$1011 = 1*2^3 + 1*2^1 + 1 = 8 + 2 + 1 = 11_{10}$
3 2 1 0

$110 = 1*2^2 + 1*2^1 = 4 + 2 = 6_{10}$
2 1 0

$10001 = 1*2^4 + 1 = 16 + 1 = 17_{10}$
4 3 2 1 0

Сумма первых двух чисел (11 и 6) равна третьему числу (17), следовательно операция выполнена верно.

Обратите внимание на то, что при добавлении к числу, состоящему из единиц (11…1), еще одной единицы, получается число, равное 1 с количеством нулей, равным количеству единиц исходного числа, например:

$1111\ 1111_2 + 1 = 1\ 0000\ 0000_2 = 2^8$

Пример выполнения операции вычитания в двоичной системе счисления:

Вычитание выполняется по тем же правилам, что и в 10-ой системе, но в 10-й системе при заеме единицы старшего разряда она превращается в 10 единиц младшего разряда, а в 2-й системе – в 2 единицы. Если нужно произвести заем не в соседнем разряде, а далее влево, то из каждых двух единиц текущего разряда одна остается в этом разряде, а вторая передается вправо. Сравните:

9 9 10
1 0 0 0$_{10}$
- 1
---
9 9 9$_{10}$

1 1 2
1 0 0 0$_2$
- 1
---
1 1 1$_2$

Выполним в 2-й системе счисление вычитание $17_{10} – 6_{10}$:
0 1 1 2
1 0 0 0 1$_2$

$$\begin{array}{r} - \quad 1\,1\,0_2 \\ \hline 1\,0\,1\,1_2 = 11_{10} \end{array}$$

Проверка показывает, что вычитание выполнено верно.

Если в двоичной системе счисления из числа, являющегося степенью двойки, вычесть 1, то получается число, состоящее из единиц, количество которых равно количеству нулей двоичного числа, например:

$2^8 - 1 = 1\,0000\,0000_2 - 1 = 1111\,1111_2$

$1023 = 1024 - 1 = 2^{10} - 1 = 11\,1111\,1111_2$

Пример выполнения операции умножения в двоичной системе счисления:

$$\begin{array}{r} 1\,1\,0\,1_2 = 13_{10} \\ * \quad 1\,0\,1_2 = 5_{10} \\ \hline 1\,1\,0\,1 \\ +1\,1\,0\,1 \\ \hline 1\,0\,0\,0\,0\,0\,1_2 \end{array} = 2^6 + 1 = 64 + 1 = 65_{10} \qquad (13 * 5 = 65)$$

6 5 4 3 2 1 0

Рассмотрим подробнее, как процессор выполняет умножение двоичных чисел. Пусть надо умножить число 1101 на 101 (оба числа в двоичной системе счисления). Машина делает это следующим образом: она берет число 1101 и, если первый справа элемент второго множителя равен 1, то она заносит его в сумму. Затем сдвигает число 1101 влево на одну позицию, получая тем самым 11010, и, если, второй элемент второго множителя равен единице, то добавляет его к сумме. Если элемент второго множителя равен нулю, то сумма не изменяется. Этот процесс сдвигов и сложений повторяется.

Пример выполнения операции деления в двоичной системе счисления:

Двоичное деление основано на методе, знакомом вам по десятичному делению, т. е. сводится к выполнению операций умножения и вычитания. Выполнение основной процедуры - выбор числа, кратного делителю и предназначенного для уменьшения делимого, здесь проще, так как таким числом могут быть только либо 0, либо сам делитель.

В качестве примера разделим $143_{10} = 10001111_2$ на $13_{10} = 1101_2$

$$\begin{array}{r|l} 1\,0\,0\,0\,1\,1\,1\,1 & 1\,1\,0\,1 \\ -1\,1\,0\,1 & 1\,0\,1\,1_2 = 11_{10} \\ \hline 1\,0\,0\,1\,1 & \\ -1\,1\,0\,1 & \\ \hline 1\,1\,0\,1 & \\ -1\,1\,0\,1 & \\ \hline 0 & \end{array}$$

Проверка показывает, что деление выполнено верно (143 / 13 = 11).

Умножение или деление двоичного числа на 2 приводит к перемещению запятой, отделяющей целую часть от дробной на один разряд соответственно вправо или влево:

$1011_2 * 10_2 = 10110_2$.

$1011_2 / 10_2 = 101.1_2$.

8-ая система счисления

При наладке аппаратных средств ЭВМ или создании новой программы возникает необходимость "заглянуть внутрь" памяти машины, чтобы оценить ее текущее состояние. Но там все заполнено длинными последовательностями нулей и единиц двоичных чисел. Эти последовательности очень неудобны для восприятия человеком, привыкшим к более короткой записи десятичных чисел. Кроме того, естественные возможности человеческого мышления не позволяют оценить быстро и точно величину числа, представленного, например, комбинацией из 16 нулей и единиц.

Для облегчения восприятия двоичного числа решили разбивать его на группы разрядов, например, по три или четыре разряда. Эта идея оказалась очень удачной, так как последовательность из трех бит имеет 8 комбинаций, а последовательность из 4 бит - 16. Числа 8 и 16 являются степенями двойки, поэтому легко находить соответствие с двоичными числами. Развивая эту идею, пришли к выводу, что группы разрядов можно закодировать, сократив при этом длину последовательности знаков. Для кодировки трех битов требуется восемь цифр, поэтому взяли цифры от 0 до 7 десятичной системы. Для кодировки же четырех битов необходимо шестнадцать знаков; для этого взяли 10 цифр десятичной системы и 6 букв латинского алфавита: A, B, C, D, E, F. Полученные системы, имеющие основания 8 и 16, назвали соответственно восьмеричной и шестнадцатеричной.

В **восьмеричной** (octal) системе счисления используются восемь различных цифр: 0, 1, 2, 3, 4, 5, 6, 7. Основание системы - 8. При записи отрицательных чисел перед последовательностью цифр ставят знак минус. Сложение, вычитание, умножение и деление чисел, представленных в восьмеричной системе, выполняются весьма просто, подобно тому, как это делают в общеизвестной десятичной системе счисления.

Пример выполнения операции сложения в восьмеричной системе счисления:

```
  1 1
  4 7 6
+ 3 4
-------
  5 3 2
```

Красным цветом показан перенос из младших разрядов в старшие.

Выполнение операции в каждом разряде:

1) $6 + 4 = 10 = 1*8 + 2 = 12_8$

2) $1 + 7 + 3 = 1*8 + 3 = 13_8$

3) $1 + 4 = 5$

Проверим результат путем перевода чисел в десятичную систему счисления :

$476_8 = 4*8^2 + 7*8 + 6 = 318$

$34_8 = 3*8 + 4 = 28$

$532 = 5*8^2 + 3*8 + 2 = 346$

318
+28
346

Пример выполнения операции вычитания в восьмеричной системе счисления:

```
  7 8
5 3 2
- 3 4
4 7 6
```

Красным цветом показан перенос из старших разрядов в младшие.

Выполнение операции в каждом разряде:

1) 8 + 2 – 4 = 6

2) 7 + 2 - 3 = $1*8 + 3 = 13_8$

3) 1 + 4 = 5

Пример выполнения операции умножения в восьмеричной системе счисления:

```
  5 4
* 3 4
2 6 0
+2 0 4
2 3 2 0
```

```
 54
*  4
260
```

4*4 = 16 = $2*8 + 0 = 20_8$ (записываем 0)

2+ 5*4 = 22 = $2*8 + 6 = 26_8$

```
 54
*  3
204
```

4*3 = 12 = $1*8 + 4 = 14_8$ (записываем 4)

1 + 5*3 = 16 = $2*8 + 0 = 20_8$

Выполним проверку:

$54_8 = 5*8 + 4 = 44_{10}$

$34_8 = 3*8 + 4 = 28_{10}$

$2320_8 = 2*8^3 + 3*8^2 + 2*8 = 1232_{10}$

```
 44
*28
352
+88
```

$= 1232_{10}$

Пример выполнения операции деления в восьмеричной системе счисления:

```
 2 3 2 0 | 5 4
-2 0 4   | 3 4
  2 6 0
- 2 6 0
      0
```

Деление в восьмеричной системе близко делению в десятичной системе: нужно подобрать цифры частного. 232 делим на 54, в десятичной системе мы получили бы целое частное 4, но из предидущего примера мы знаем, что в восьмеричной системе 54*4 = 260, это много, попробуем взять цифру поменьше – 3, умножаем 54*3 = 204, эта цифра подходит, и т.д.

В различных языках программирования запись восьмеричных чисел начинается с 0, например, запись 011 означает десятичное число 9.

16-ая система счисления

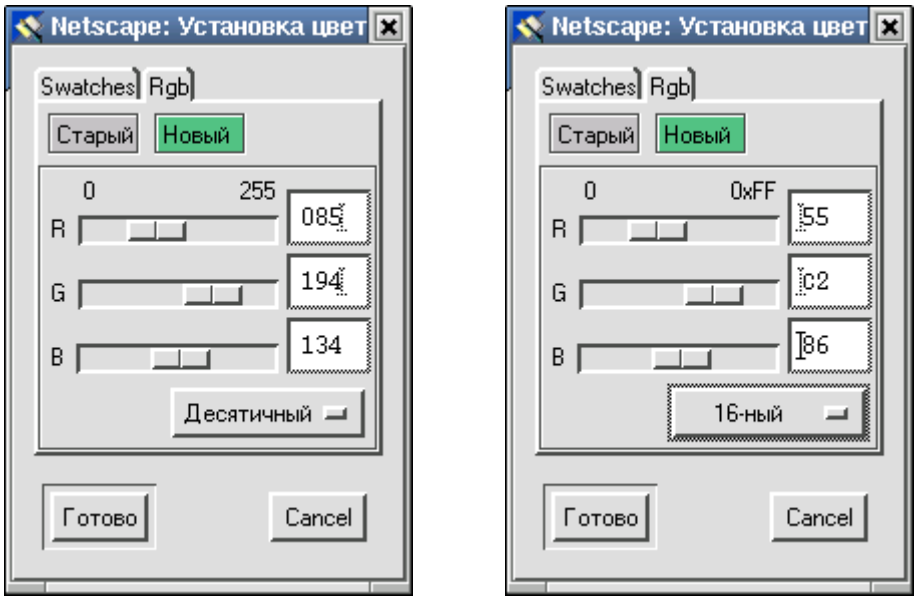
В **шестнадцатеричной** (hexadecimal) системе счисления применяются десять цифр от 0 до 9 и шесть первых букв латинского алфавита:

10 – A 11 – B 12 – C 13 – D 14 – E 15 – F.

При записи отрицательных чисел слева от последовательности цифр ставят знак минус.

Для того чтобы при написании **компьютерных программ** отличить числа, записанные в шестнадцатеричной системе, от других, перед числом ставят 0x. То есть 0x11 и 11 - это разные числа.

Шестнадцатеричная система счисления широко используется при задании различных оттенков цвета при кодировании графической информации (модель RGB). Так, в редакторе гипертекста Netscape Composer можно задавать цвета для фона или текста как в десятичной, так и шестнадцатеричной системах счисления (см. рисунок).



Пример выполнения операции сложения в 16-ой системе счисления:

1 1 — Красным цветом показан перенос из младших разрядов

$A\ 7\ B_{16}$

$+\ C\ 8_{16}$

$B\ 4\ 3_{16}$

Выполнение операции в каждом разряде:

B + 8 = 11 + 8 = 19 = 1*16 + 3 = 13_{16} (записываем 3)

1+7+C = 8+12 = 20 = 1*16 + 4 = 14_{16} (записываем 4)

1 + A = B

Проверим результат путем перевода чисел в 10-ю систему:

$A7B_{16} = 10*16^2 + 7*16 + 11 = 2683$

2 1 0

$C8_{16} = 12*16 + 8 = 200$

1 0

$B43_{16} = 11*16^2 + 4*16 + 3 = 2883$

2 1 0

2683

+ 200

2883

Пример выполнения операции вычитания в 16-ой системе счисления:

15 16 Красным цветом показан заем из старших разрядов

B 4 3_{16} Выполнение операции в каждом разряде:

- A 7 B_{16} 16 + 3 – B = 19 -11 = 8

C 8_{16} 15 + 4 – 7 = 12 = C

B - 1 – A = 0

Умножение и деление в 16-ой системе обычно не выполняется ввиду сложности вычислений.

Перевод чисел из одной системы счисления в другую

Перевод числа из системы счисления с основанием q в 10-ю систему счисления выполняется путем вычисления значения многочлена по степеням q, коэффициенты которого равны цифрам числа.

Рассмотрим различные способы перевода чисел из одной системы счисления в другую на конкретных примерах.

Перевод из 2-ой системы в 10-ую

$1\ 0\ 1\ 1\ .\ 1\ 0\ 1_2 = 1*2^3 + 0*2^2 + 1*2 + 1*2^0 + 1*2^{-1} + 0*2^{-2} + 1*2^{-3} =$

3 2 1 0 -1 -2 -3

$= 8 + 2 + 1 + 0.5 + 0.125 = 11.625$

Для того, чтобы быстро переводить числа из двоичной системы счисления в 10-ую, необходимо запомнить степени двойки : 2, 4, 8, 16, 32, 64, 128, 256, 512, 1024 и т.д. Отрицательные степени двойки: .5, .25, .125, .0625, .03125 и т.д.

Перевод из 8-ой системы в 10-ую

$6\ 3\ 2.4\ 5_8 = 6*8^2 + 3*8 + 2 + 4*8^{-1} + 5*8^{-2} = 6*64 + 24 + 2 + 4/8 + 5/64 =$

2 1 0 -1 -2

$= 410.578125$

Перевод из 16-ой системы в 10-ую

$E\ 7\ F.8_{16} = 14*16^2 + 7*16 + 15 + 8/16 = 14*256 + 7*16 + 15 + .5 = 3711.5$

2 1 0 -1

Перевод из 10-ой системы в 2-ую

Перевод из 10-ой системы целой и дробной частей выполняется по различным алгоритмам, поэтому будем рассматривать их отдельно.

Перевод целой части

Пусть требуется перевести число 567 из десятичной в двоичную систему. Сначала определим максимальную степень двойки, такую, чтобы два в этой степени было меньше или равно исходному числу. В нашем случае это 9, т. к. 2^9=512, а 2^{10}=1024, что больше начального числа. Таким образом, мы получим число разрядов результата. Оно равно 9+1=10. Поэтому результат будет иметь вид 1xxxxxxxxx, где вместо x могут стоять любые двоичные цифры. Найдем вторую цифру результата. Возведем двойку в степень 9 и вычтем из исходного числа: $567-2^9=55$. Остаток сравним с числом 2^8=256. Так как 55 меньше 256, то девятый разряд будет нулем, т. е. результат примет вид 10xxxxxxxx. Рассмотрим восьмой разряд. Так как $2^7=128>55$, то и он будет нулевым.

Седьмой разряд также оказывается нулевым. Искомая двоичная запись числа принимает вид 1000xxxxxx. $2^5=32<55$, поэтому шестой разряд равен 1

(результат 10001xxxxx). Для остатка 55-32=23 справедливо неравенство $2^4 = 16 < 23$, что означает равенство единице пятого разряда. Действуя аналогично, получаем в результате число 1000110111. Мы разложили данное число по степеням двойки:

$$567=1*2^9 + 0*2^8 + 0*2^7 + 0*2^6 + 1*2^5 + 1*2^4 + 0*2^3 + 1*2^2 + 1*2^1 + 1*2^0$$

При другом способе перевода чисел используется операция деления в столбик. Рассмотрим то же самое число 567. Разделив его на 2, получим частное 283 и остаток 1. Проведем ту же самую операцию с числом 283. Получим частное 141, остаток 1. Опять делим полученное частное на 2, и так до тех пор, пока частное не станет меньше делителя. Теперь для того, чтобы получить число в двоичной системе счисления, достаточно записать последнее частное, то есть 1, и приписать к нему в обратном порядке все полученные в процессе деления остатки.

```
567 | 2
566 | 283 | 2
  1   282 | 141 | 2
        1   140 | 70 | 2
              1   70 | 35 | 2
                    0   34 | 17 | 2
                          1   16 | 8 | 2
                                1   8 | 4 | 2
                                    0   4 | 2 | 2
                                        0   2 | 1
                                            0
```

Результат, естественно, не изменился: 567 в двоичной системе счисления записывается как 1000110111.

Поскольку делить на 2 несложно, этот процесс можно записать более компактно:

Частное	Остаток
567	1
283	1
141	1
70	0
35	1
17	1
8	0
4	0
2	0
1	1

$567 = 1000110111_2$

Перевод дробной части

Алгоритм перевода дробной части:

1) последовательно умножать дробную часть на основание новой системы счисления, пока не получим нулевую дробную часть или не будет достигнута требуемая точность вычислений.
2) Записать полученные целые части произведений в прямой последовательности

Примеры:

1) перевести 0.65625 в 2-ю систему счисления.

Умножаем дробную часть на 2:

целая часть произведения	дробная часть произведения
	65625
1	3125
0	625
1	25
0	5
1	0

Умножаем только дробную часть!

$0.65625 = 0.10101_2$

2) перевести 0.1 в 2-ю систему счисления.

Умножаем дробную часть на 2:

целая часть произведения	дробная часть произведения
	1
0	2
0	4
0	8
1	6
1	2
0	4
0	8
1	6
1	2
. . .	

Умножаем только дробную часть!

С этого места процесс повторяется

0.1 = 0. 0 0011 0011 0011 …

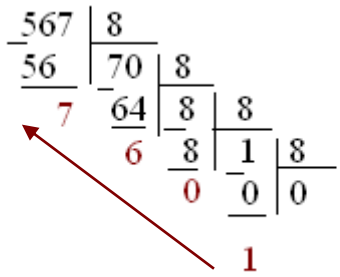
В результате перевода большинства десятичных чисел, имеющих дробную часть, получается число с бесконечной дробью, поэтому действительные (вещественные) числа в компьютере хранятся не точно!

Перевод из 10-ой системы в 8-ую

Перевод целой части

Алгоритм перевода из десятичной системы в систему счисления с основанием q путем деления и записи остатков в обратном порядке более удобен, поэтому для перевода числа в 8-ю и 16-ую системы мы будем использовать его.

Рассмотрим перевод числа 567 в систему счисления с основанием 8.



$567 = 1067_8$

Перевод дробной части

Переведем 0.65625 в 8-ю систему счисления.

Умножаем дробную часть на 8:

целая часть произведения	дробная часть произведения
	65625
5	25
2	0

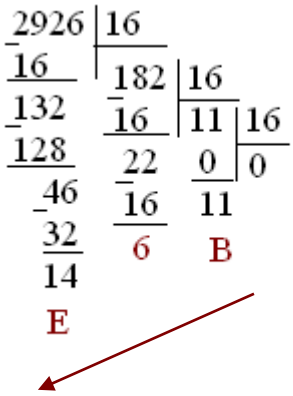
Умножаем только дробную часть!

$0.65625 = 0.52_8$

Перевод из 10-ой системы в 16-ую

Перевод целой части

Делим число на 16 и записываем остатки в обратном порядке:



В шестнадцатеричной системе счисления необходимо заменить 10 на A, 11 на B и так далее.

Перевод дробной части

Переведем 0.65625 в 16-ю систему счисления.
Умножаем дробную часть на 16:

целая часть произведения	дробная часть произведения
	65625
10(A)	5
8	0

Умножаем только дробную часть!

$0.65625 = 0.A8_{16}$

Перевод из 2-ой системы в 8-ю или 16-ю и обратно

Пожалуй, проще всего осуществляется перевод чисел из двоичной системы в системы с основанием, равным степеням двойки (8 или 16), и наоборот. Для того чтобы целое двоичное число записать в системе счисления с основанием 2^n, нужно

- данное двоичное число разбить на группы по n-цифр в каждой справа налево в целой части и слева-направо в дробной;
- если в последней группе окажется меньше n разрядов, то дополнить ее нулями до нужного числа разрядов;
- рассмотреть каждую группу, как n-разрядное двоичное число, и заменить ее соответствующей цифрой в системе счисления с основанием 2^n.

Таблица перевода из двоичной системы в 16-ю и обратно

Десятичное значение	Двоичный код	Шестнадцатеричная цифра
0	0000	0
1	0001	1
2	0010	2
3	0011	3

4	0100	4
5	0101	5
6	0110	6
7	0111	7
8	1000	8
9	1001	9
10	1010	A
11	1011	B
12	1100	C
13	1101	D
14	1101	E
15	1111	F

Часть таблицы, выделенная бирюзовым, может использоваться для перевода из 2-й системы в 8-ю и обратно.

Примеры:

1) Переведем число 11101.00111_2 из двоичной системы в восьмеричную.

Разбиваем двоичное число на тройки цифр:

$11101.00111_2 = 011\ 101.001\ 110_2 = 35.16_8$

Заменяем каждую тройку двоичных цифр соответствующей 8-й цифрой (см. таблицу).

Для перевода числа из 8-й системы счисления в 2-ю нужно каждую 8-ю цифру заменить тройкой двоичных цифр (рассмотрите тот же пример справа-налево).

2) Переведем число 10000.110111_2 в 16-ю систему.

Разбиваем двоичное число на четверки цифр:

$10000.110111_2 = 0001\ 0000.1101\ 1100_2 = 10.DC_{16}$

Заменяем каждую четверку двоичных цифр соответствующей 16-й цифрой (см. таблицу).

Для перевода числа из 16-й системы счисления в 2-ю нужно каждую 16-ю цифру заменить четверкой двоичных цифр (рассмотрите тот же пример справа-налево).

Примеры двоичного кодирования информации

Среди всего разнообразия информации, обрабатываемой на компьютере, значительную часть составляют числовая, текстовая, графическая и аудиоинформация. Познакомимся с некоторыми способами кодирования этих типов информации в ЭВМ.

Кодирование чисел

Существуют два основных формата представления чисел в памяти компьютера. Один из них используется для кодирования целых чисел, второй

(так называемое представление числа в формате с плавающей точкой) используется для задания некоторого подмножества действительных чисел.

Кодирование целых чисел

Множество целых чисел, представимых в памяти ЭВМ, ограничено. Диапазон значений зависит от размера области памяти, используемой для размещения чисел. В k-разрядной ячейке может храниться 2^k различных значений целых чисел.

Целые числа могут занимать 1, 2, 4 или 8 байт (для 64-разрядных машин).

Чтобы получить внутреннее представление целого положительного числа N, хранящегося в k-разрядном машинном слове, необходимо:

1. перевести число N в двоичную систему счисления;
2. полученный результат дополнить слева незначащими нулями до k разрядов.

Код целого числа может рассматриваться как двоичное число со знаком или без знака.

При беззнаковом представлении все разряды используются для записи значения числа.

Пример:

Число 107 = 1101011_2 будет записано:

в 1 байт как 01101011

в 2 байта как 00000000 01101011
1-й байт 0-й байт

в 4 байта как 00000000 00000000 00000000 01101011
3-й байт 2-й байт 1-й байт 0-й байт

Минимальное беззнаковое число равно 0. Максимальное беззнаковое число равно $2^n - 1$, где n – кол-во двоичных разрядов, используемых для записи числа.

Например для 2-хбайтового представления max =11111111 11111111_2 = 1 00000000 00000000 – 1 = $2^{16} - 1$ = 65 535

Для записи чисел со знаком старший (левый) разряд отводится под знак числа. Если число неотрицательное, то в знаковый разряд записывается 0, в противном случае – 1, т.е. единица в знаковом разряде означает знак “минус”.

Целые числа со знаком могут быть записаны в прямом, обратном и дополнительном коде.

В прямом коде число хранится в виде: знак+абсолютное значение (модуль) числа.

В обратном коде в значении числа нули заменяют на единицы, а единицы на нули.

Дополнительный код получают путем прибавления 1 к обратному.

Обратный и дополнительный код неотрицательных чисел совпадает с прямым.

Обратный и дополнительный коды чисел позволяют заменить операцию вычитания сложением с отрицательным числом, что существенно упрощает устройство процессора. Варианты арифметических операций будут рассмотрены ниже.

Пример. Рассмотрим внутреннее представление целого отрицательного числа: $-6 = 110_2$.

Однобайтовое:

Прямой код: 1000 0110
Обратный код: 1111 1001
Дополнительный: 1111 1001
$+1$
1111 1010

Четырехбайтовое:

Прямой код: 10000000 00000000 00000000 00000110
Обратный код: 1111111 1111111 11111111 11111001
Дополнительный: 1111111 1111111 11111111 11111001
$+1$
1111111 1111111 11111111 11111010

Для того, чтобы получить значение отрицательного числа, записанного в дополнительном коде, можно использовать один из двух алгоритмов:

1) вычесть 1 из дополнительного кода (получаем обратный код) и заменить все нули на единицы, а единицы на нули;

2) сначала заменить все нули на единицы, единицы на нули, затем прибавить единицу к результату.

Пример: возьмем однобайтовый доп. код : 1111 1010 и используем второй алгоритм: 1111 1010 -- > - (0000 0101 + 1) = - 110_2 = -6.

Диапазон значений знаковых чисел

Рассмотрим однобайтовое представление. Возможные дополнительные коды знаковых чисел:

0111 1111
. . .
0000 0001
0000 0000
1111 1111
1111 1110 ↕ Отрицательные числа
. . .
1000 0000

Рассмотрим десятичные значения этих чисел:

0111 1111 = $2^7 - 1 = 128 - 1 = 127$
0000 0001 = 1
0000 0000 = 0
1111 1111 -> -(000 0000 + 1) = -1
1111 1110 -> -(000 0001 + 1) = -2

1000 0000 -> -(111 1111 + 1) = -(1000 0000) = -2^7 = -128

Таким образом диапазон значений знаковых однобайтовых чисел: от -128 до 127.

Аналогично, диапазон значений двухбайтовых целых чисел: -2^{15} - +(2^{15} -1) (от -32768 до 32767).

Диапазон значений четырехбайтовых целых чисел со знаком: -2^{31} - +(2^{31} – 1) (от -2 147 483 648 до 2 147 483 647)

Сложение и вычитание целых чисел

В большинстве компьютеров операция вычитания не используется. Вместо нее производится сложение обратных или дополнительных кодов уменьшаемого и вычитаемого. Это позволяет существенно упростить конструкцию арифметико-логического устройства процессора.

Сложение обратных кодов. Здесь при сложении чисел А и В имеют место четыре основных и два особых случая:

1. А и В положительные. При суммировании складываются все разряды, включая разряд знака. Так как знаковые разряды положительных слагаемых равны нулю, разряд знака суммы тоже равен нулю. Например:

Десятичная запись	Двоичные коды
+ 3 7 10	+ 0 0000011 0 0000111 0 0001010

Получен правильный результат.

2. А положительное, В отрицательное и по абсолютной величине больше, чем А. Например:

Десятичная запись	Двоичные коды	
+ 3 -10 -7	+ 0 0000011 1 1110101 1 1111000	 Обратный код числа -10 Обратный код числа -7

Получен правильный результат в обратном коде. При переводе в прямой код биты цифровой части результата инвертируются: 1 0000111 = -7_{10}.

3. А положительное, В отрицательное и по абсолютной величине меньше, чем А. Например:

Десятичная запись	Двоичные коды	
+ 10 -3 7	+ 0 0001010 1 1111100 0 0000110 +1 0 0000111	 Обратный код числа -3

Компьютер исправляет полученный первоначально неправильный результат (6 вместо 7) переносом единицы из знакового разряда в младший разряд суммы.

4. A и B отрицательные. Например:

Десятичная запись	Двоичные коды	
-3	1 1111100	Обратный код числа -3
+ -7	+ 1 1111000	Обратный код числа -7
-10	1 1110100	
	→ +1	
	1 1110101	Обратный код числа -10

Полученный первоначально неправильный результат (обратный код числа -11_{10} вместо обратного кода числа -10_{10}) компьютер исправляет переносом единицы из знакового разряда в младший разряд суммы. При переводе результата в прямой код биты цифровой части числа инвертируются: 1 0001010 = -10_{10}.

Переполнение

При сложении может возникнуть ситуация, когда старшие разряды результата операции не помещаются в отведенной для него области памяти. Такая ситуация называется **переполнением разрядной сетки формата числа.** Для обнаружения переполнения и оповещения о возникшей ошибке в компьютере используются специальные средства. Ниже приведены два возможных случая переполнения.

5. A и B положительные, сумма A+B больше, либо равна 2^{n-1}, где n — количество разрядов формата чисел (для однобайтового формата n=8, $2^{n-1} = 2^7 = 128$). Например:

Десятичная запись	Двоичные коды	
65	0 1000001	
+ 97	+ 0 1100001	
162	1 0100010	Переполнение

Обратите внимание: в результате сложения положительных чисел получен отрицательный результат!

Семи разрядов цифровой части числового формата **недостаточно** для размещения восьмиразрядной суммы ($162_{10} = 10100010_2$), поэтому **старший разряд суммы оказывается в знаковом разряде.** Это вызывает **несовпадение знака суммы и знаков слагаемых**, что **является свидетельством переполнения разрядной сетки**.

6. A и B отрицательные, сумма абсолютных величин A и B больше, либо равна 2^{n-1}. Например:

Десятичная запись		Двоичные коды	
+ -63		+ 1 1000000	Обратный код числа -63
-95		1 0100000	Обратный код числа -95
158		0 1100000	Переполнение
		→ +1	

В результате сложения отрицательных чисел получен результат > 0!

Здесь знак суммы тоже не совпадает со знаками слагаемых, что свидетельствует о переполнении разрядной сетки.

Сложение дополнительных кодов. Здесь также имеют место рассмотренные выше шесть случаев:

1. A и B положительные. Здесь нет отличий от случая 1, рассмотренного для обратного кода (коды неотрицательных чисел совпадают).

2. A положительное, B отрицательное и по абсолютной величине больше, чем A. Например:

Десятичная запись		Двоичные коды	
+ 3		+ 0 0000011	
-10		1 1110110	Дополнительный код числа -10
-7		1 1111001	Дополнительный код числа -7

Получен правильный результат в дополнительном коде. При переводе в прямой код биты цифровой части результата инвертируются и к младшему разряду прибавляется единица: 1 0000110 + 1 = 1 0000111 = -7_{10}.

3. A положительное, B отрицательное и по абсолютной величине меньше, чем A. Например:

Десятичная запись		Двоичные коды	
+ 10		+ 0 0001010	
-3		1 1111101	Дополнительный код числа -3
7		0 0000111	
		→ перенос отбрасывается	

Получен правильный результат. Единицу переноса из знакового разряда компьютер отбрасывает.

4. A и B отрицательные. Например:

Десятичная запись		Двоичные коды	
+ -3		+ 1 1111101	Дополнительный код числа -3
-7		1 1111001	Дополнительный код числа -7
-10		1 1110110	Дополнительный код числа -10
		→ перенос отбрасывается	

Получен правильный результат в дополнительном коде. **Единицу переноса** из знакового разряда компьютер **отбрасывает**.

Случаи переполнения

Для обнаружения переполнения разрядной сетки знаковый разряд дублируется. Такое представление чисел называется модифицированным дополнительным кодом:

1)	65	00 100 0001
	+97	+ 00 110 0001
	162	01 010 0010

Разные цифры в знаковых разрядах свидетельствуют о том, что произошло переполнение.

2)	-65	11 011 1111
	+-97	+ 11 001 1111
	-162	10 101 1110
		Переполнение!

Для проверки знаковых разрядов используют результат операции “исключающее ИЛИ”, которая дает значение 1 только если операнды различны.

Сравнение рассмотренных форм кодирования целых чисел со знаком показывает:

на преобразование отрицательного числа в обратный код компьютер затрачивает меньше времени, чем на преобразование в дополнительный код, так как последнее состоит из двух шагов — образования обратного кода и прибавления единицы к его младшему разряду;

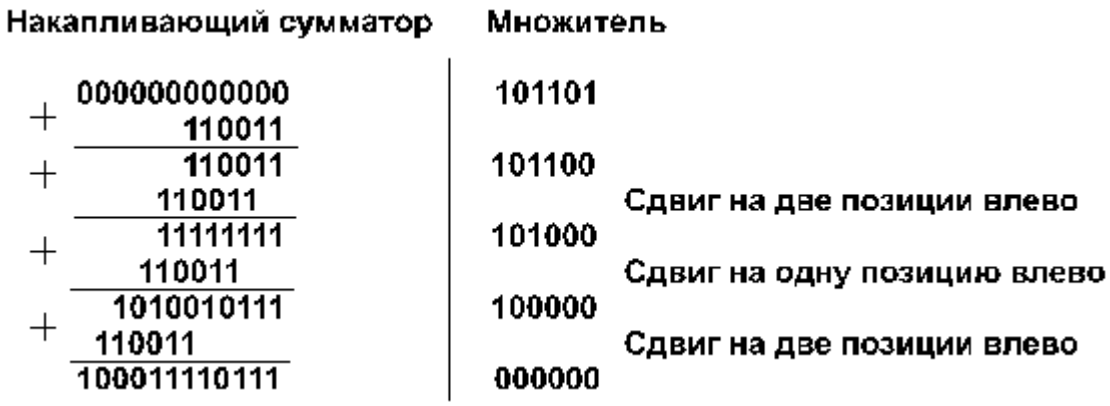
время выполнения сложения для дополнительных кодов чисел меньше, чем для их обратных кодов, потому что в таком сложении нет переноса единицы из знакового разряда в младший разряд результата, поэтому для ускорения выполнения расчетов используют **дополнительный код**.

Умножение и деление

Во многих компьютерах умножение производится как последовательность сложений и сдвигов. Для этого в АЛУ имеется регистр, называемый накапливающим сумматором, который до начала выполнения операции содержит число ноль. В процессе выполнения операции в нем поочередно размещаются множимое и результаты промежуточных сложений, а по завершении операции — окончательный результат.

Другой регистр АЛУ, участвующий в выполнении этой операции, вначале содержит множитель. Затем по мере выполнения сложений содержащееся в нем число уменьшается, пока не достигнет нулевого значения.

Для иллюстрации умножим 110011_2 на 101101_2.



Деление для компьютера является трудной операцией. Обычно оно реализуется путем многократного прибавления к делимому дополнительного кода делителя.

Кодирование вещественных чисел

Формат с плавающей точкой использует представление вещественного числа R в виде произведения мантиссы m на основание системы счисления q в некоторой целой степени p, которую называют порядком: $R = m * q^{p}$.

Представление числа в форме с плавающей точкой неоднозначно. Например, справедливы следующие равенства:

$$12.345 = 0.0012345 * 10^{4} = 1234.5 * 10^{-2} = 0.12345 * 10^{2}$$

Чаще всего в ЭВМ используют нормализованное представление числа в форме с плавающей точкой. Мантисса в таком представлении должна удовлетворять условию: $0.1_{p} <= m < 1$. Иначе говоря, мантисса должна быть меньше 1 и первая значащая цифра - не ноль (p - основание системы счисления).

В памяти компьютера мантисса представляется как целое число, содержащее только значащие цифры (0 целых и запятая не хранятся), так для числа 12.345 в ячейке памяти, отведенной для хранения мантиссы, будет сохранено число 12345. Для однозначного восстановления исходного числа остается сохранить только его порядок, в данном примере - это 2.

Диапазон и точность представления чисел зависят от числа разрядов, отводимых под порядок и мантиссу. Обычно число в формате с плавающей запятой занимает в памяти 4 (float) или 8 (double) байтов.

В большинстве вычислительных машин для упрощения операций над порядками их приводят к целым положительным числам, применяя так называемый смещенный порядок. Для этого к истинному порядку добавляется целое положительное число, равное половине представимого диапазона порядков.

Числа с плавающей запятой в разных вычислительных машинах (ВМ) имеют различные форматы. В настоящее время для всех ВМ рекомендован стандарт, разработанный международным центром стандартизации IEEE (Institute of Electrical and Electronics Engineers).

Стандарт IEEE 754

Рекомендуемый для всех ВМ формат представления чисел с плавающей запятой определен стандартом IEEE 754. Этот стандарт был разработан с целью облегчить перенос программ с одного процессора на другие и нашел широкое применение практически во всех процессорах и арифметических сопроцессорах.

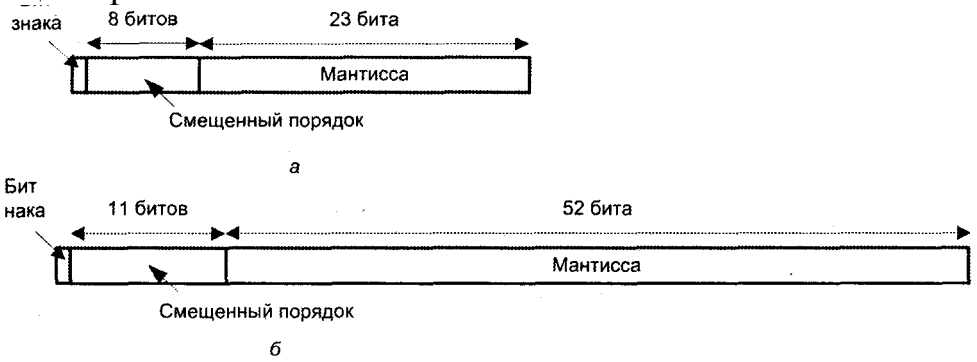


Рис. 2.24. Основные форматы IEEE 754: а — одинарный; б — двойной

Стандарт определяет 32-битовый (одинарный) и 64-битовый (двойной) форматы (рис. 2.24) с 8- и 11-разрядным порядком соответственно. Самый левый бит хранит знак числа. Основанием системы счисления является 2.

Смещение равно соответственно 127 и 1023.

Максимальный порядок, который может иметь число: 127 и 1023.

Для повышения точности представления мантиссы используют прием скрытой единицы: поскольку в нормализованной мантиссе старшая цифра всегда равна 1, ее можно не хранить. Следовательно, при 4-хбайтовом представлении, мантисса фактически состоит из 24 разрядов. Скрытая единица при выполнении арифметических операций восстанавливается, а при записи результата удаляется.

Пример: рассмотрим 4-хбайтовый код числа 20.5:

$20.5 = 10100.1_2 = 0.101001 * 2^5$

Порядок (смещенный): $5+127 = 132 = 1000\ 0100_2$

Мантисса: 101001 → 010010…0 (первая единица – скрытая, в конец мантиссы добавляются нули).

4-хбайтовое представление:

0	1	0	0	0	0	1	0	0	0	1	0	0	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
	порядок								мантисса																							

Знак

В 16-ом виде этот код будет выглядеть так: 42240000.

Определим максимальное число и его точность при 4-хбайтовом представлении.

Максимальное число:

$.1\ldots1 * 2^{127} = 1 * 2^{127} = 1.7 * 10^{38}$

Максимальное значение мантиссы:

$1\ldots1$ (24 единицы) $= 2^{24} - 1 = 2^{10*2.4} = 1024^{2.4} = 1.7*10^{7}$, следовательно точность представления мантиссы 7-8 значащих цифр.

Арифметические операции с числами в формате с плавающей запятой

Сложение и вычитание

Производятся в несколько этапов:

1) Выравниваются порядки чисел в сторону большего (чтобы не получить мантиссы > 1)
2) Складываются мантиссы. Для представления отрицательных чисел используется модифицированный дополнительный код. Порядок суммы будет равен общему порядку слагаемых.
3) Нормализуется результат: порядок и мантисса изменяются так, чтобы первая значащая цифра результата попала в первый разряд после запятой.

Пример 1: Вычесть из числа A = 20.0 число B = 11.0.

$A = 20 = 10100_2 = .101 * 2^5 = .101 * 10^{101}$ (все числа –двоичные)

$B = 11 = 1011_2 = .1011 * 2^4 = .1011 * 10^{100}$

Процессор для определения разности порядков вычитает из порядка числа A порядок числа B и получает 1. Т.к. порядок числа A на единицу больше порядка числа B, порядок числа B увеличивается на 1 и мантисса при этом сдвигается на 1 разряд вправо относительно точки:

$B = .01011 * 10^{101}$

Мантисса числа B должна быть записана как отрицательное число (нужно выполнить вычитание):

B = -010110…0 = 1|101001…1 = 1|101010…0

Обратный код Дополнительный

Сложение мантисс в модифицированном дополнительном коде:

```
  00| 1010 00…0   (число A)
+ 11| 1010 10…0   (число B)
1| 00| 0100 10…0   (сумма, порядок = 101₂)
```

Произошло нарушение нормализации.

Нормализация результата: мантисса сдвигается влево, порядок уменьшается: $A - B = .1001 * 10^{100} = 1001_2 = 9.0$

Пример 2: Сложить A = 5.0 и B = 28.0.

$A = 5 = 101_2 = .101 * 2^5 = .101 * 10^{11}$ (все числа –двоичные)

$B = 28 = 11100_2 = .111 * 2^5 = .111 * 10^{101}$

Процессор для определения разности порядков вычитает из порядка числа A порядок числа B и получает -2. Т.к. порядок числа A на 2 меньше порядка числа B, порядок числа A увеличивается на 2 и мантисса при этом сдвигается на 2 разряда вправо относительно точки:

$A = .00101 * 10^{101}$

Сложение мантисс в модифицированном коде:

00| 0010 10…0 (число A)

\+ 00| 1110 00…0 (число B)

01| 0000 10…0 (сумма, порядок = 101_2)

Произошло нарушение нормализации.

Нормализация результата: мантисса сдвигается вправо, порядок увеличивается: $A + B = .100001* 10^{110} = 100001_2 = 33.0$

При сложении и вычитании чисел с плавающей запятой при сложении мантисс переполнение не фиксируется. Переполнение может возникнуть в процессе нормализации, если порядок превысит максимально допустимый.

Умножение и деление

При умножении чисел в формате с плавающей запятой порядки складываются, а мантиссы перемножаются, затем результат нормализуется.

При делении из порядка делимого вычитается порядок делителя, а мантисса делимого делится на мантиссу делителя, затем результат нормализуется.

Двоично-десятичное кодирование информации

Двоично-десятичный код — (binary-coded decimal [BCD]) форма записи целых чисел, когда каждый десятичный разряд числа записывается в виде его четырёхбитного двоичного кода (вместо каждой десятичной цифры записывают ее двоичное значение). Например, десятичное число 310 будет записано в двоичном коде как 100110110_2, а в двоично-десятичном коде как $0011\ 0001\ 0000_{BCD}$.

Преимущества и недостатки

Преимущества

- Упрощён вывод чисел на индикацию — вместо последовательного деления на 10 требуется просто вывести на индикацию каждый полубайт. Аналогично, проще ввод данных с цифровой клавиатуры.
- Для дробных чисел (как с фиксированной, так и с плавающей запятой) при переводе в человекочитаемый десятичный формат и наоборот не теряется точность.
- Упрощены умножение и деление на 10, а также округление.

По этим причинам двоично-десятичный формат применяется в калькуляторах — калькулятор в простейших арифметических операциях должен выводить в точности такой же результат, какой подсчитает человек на бумаге.

Недостатки

- Усложнены арифметические операции.
- Требует больше памяти.
- В двоично-десятичном коде BCD существуют запрещённые комбинации битов:

  Запрещённые в BCD битовые комбинации:
  1010 1011 1100 1101 1110 1111

Запрещённые комбинации возникают обычно в результате операций сложения, так как в BCD используются только 10 возможных комбинаций 4-х битового поля вместо 16. Поэтому, при сложении и вычитании чисел формата BCD действуют следующие правила:

- При сложении двоично-десятичных чисел каждый раз, когда происходит перенос бита в старший полубайт, необходимо к полубайту, от которого произошёл перенос, добавить корректирующее значение 0110.
- При сложении двоично-десятичных чисел каждый раз, когда встречается недопустимая для полубайта комбинация, необходимо к каждой недопустимой комбинации добавить корректирующее значение 0110 с разрешением переноса в старшие полубайты.
- При вычитании двоично-десятичных чисел, для каждого полубайта, получившего заём из старшего полубайта, необходимо провести коррекцию, вычитая значение 0110.

Пример операции сложения двоично-десятичных чисел:
Требуется: Найти число A = D + C, где D = 3927, C = 4856
Решение: Представим числа D и C в двоично-десятичной форме: D = 3927 = 0011 1001 0010 0111 C = 4856 = 0100 1000 0101 0110
Суммируем числа D и C по правилам двоичной арифметики:

```
         *          **
  0011 1001 0010 0111
+ 0100 1000 0101 0110
  ____________________
= 1000 0001 0111 1101 - Двоичная сумма
+      0110      0110 - Коррекция
  ____________________
  1000 0111 1000 0011
```

'*' — тетрада, из которой был перенос в старшую тетраду

'**' — тетрада с запрещённой комбинацией битов

В тетраду, помеченую символом *, добавляем шестёрку т.к по правилам двоичной арифметики перенос унёс с собой 16, а по правилам десятичной арифметики должен был унести 10. В тетраду, помеченую символом ** , добавляем шестёрку, так как комбинация битов 1101 (что соответствует десятичному числу 13) является запрещённой.