Представление чисел в ЭВМ

А.А. Вылиток

1 Информация и данные

Информация (от лат. information — разъяснение, изложение) — содержание (смысл) сообщения или сигнала, сведения, рассматриваемые в процессе их передачи или восприятия. При помощи компьютеров (ЭВМ) информация может передаваться в различных формах: число, текст, аудио, видео и другие.

Информация воспринимается человеком и может вызвать у него ту или иную реакцию. Разные люди из одного и того же сообщения могут извлечь разное количество информации. Профессиональный музыкант, например, способен по нотной записи музыкального произведения получить больше сведений о его исполнении, чем программист, а программист по тексту программы может рассказать о ее поведении больше, чем музыкант.

Компьютеры имеют дело не с информацией, а с данными. Получив исходные данные, они «механически» перерабатывают их по определенным алгоритмам в выходные данные (результаты), из которых человеку обычно легче извлекать информацию, чем из исходных данных. Получив на входе арифметическое выражение “(13+27)•2”, компьютер на выход выдает эквивалентное выражение “80”, значение (смысл) которого человеку воспринять легче (не надо совершать в уме или на бумаге арифметические действия).

2 Представление данных в компьютере

Обычно входные и выходные данные представляются в форме, удобной для человека. Числа люди привыкли изображать в десятичной системе счисления. Для компьютера удобнее двоичная система. Это объясняется тем, что технически гораздо проще реализовать устройства (например, запоминающий элемент) с двумя, а не с десятью устойчивыми состояниями (есть электрический ток — нет тока, намагничен — не намагничен и т.п.). Можно считать, что одно из двух состояний означает единицу, другое — ноль.

Любые данные (числа, символы, графические и звуковые образы) в компьютере представляются в виде последовательностей из нулей и единиц. Эти последовательности можно считать словами в алфавите {0,1}, так что обработку данных внутри компьютера можно воспринимать как преобразование слов из нулей и единиц по правилам, зафиксированным в микросхемах процессора. Такой взгляд роднит вычислительные машины с абстрактными вычислителями. Вспомните машины Тьюринга или нормальные алгоритмы Маркова.

Элемент последовательности из нулей и единиц (член такой последовательности) называют *битом*. Именительный падеж — бит.

Отображение внешней информации во внутреннее представление называется кодированием. *Кодом* (франц. code, от лат. codex — свод законов) называют как сам способ отображения, так и множество слов (кодовых комбинаций), используемых при кодировании.

3 Представление целых чисел

Для представления чисел в ЭВМ обычно используют битовые наборы — последовательности нулей и единиц фиксированной длины. Организовать обработку наборов фиксированной длины технически легче, чем наборов переменной длины. Позиция в битовом наборе называется *разрядом*. В ЭВМ разрядом называют также часть регистра (или ячейки памяти), хранящую один бит.

3.1 Целые числа без знака

Как определить, какое целое число представляет тот или иной битовый набор? Возможны разные способы. Например, можно считать, что представляемое число равно количеству единиц в битовом наборе ("единичная" система счисления). Такой способ позволяет представить всего k различных целых чисел от 0 до k–1, где k — длина набора. Очевидно, что этот способ неэкономный — одному и тому же числу могут соответствовать несколько различных наборов. Количество всевозможных битовых наборов длины k равно 2^k, поэтому выгоднее различным наборам поставить в соответствие различные числа. Это позволит представить 2^k различных чисел. Обычно рассматривают диапазон целых чисел [N, N+2^k). При N=0 имеем представление беззнаковых (неотрицательных) чисел от 0 до 2^k–1.

Существует всего $(2^k)!$ (количество перестановок из 2^k элементов) способов закодировать беззнаковые числа битовыми наборами. Среди всех этих теоретически возможных способов представления чисел наиболее удобен такой: битовый набор, соответствующий числу, является k-разрядной записью этого числа в двоичной системе счисления. Таким образом, можно реализовать арифметические операции над числами, используя известные школьные алгоритмы поразрядной обработки для битовых наборов.

3.2 Целые числа со знаком

Для представления знаковых целых чисел используются три способа:

1) *прямой* код;

2) *обратный* код;

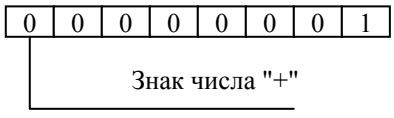
3) *дополнительный* код.

Все три способа используют самый левый (старший) разряд битового набора длины k для кодирования знака числа: знак “плюс” кодируется нулем, а “минус” — единицей. Остальные k-1 разрядов (называемые *мантиссой* или цифровой частью) используются для представления абсолютной величины числа.

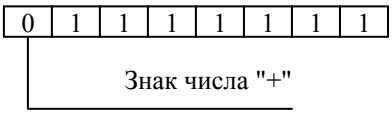
3.2.1 Положительные целые числа (и число 0)

Положительные числа в прямом, обратном и дополнительном кодах изображаются одинаково — цифровая часть содержит двоичную запись числа, в знаковом разряде содержится 0. Например, для k = 8:

Число $1_{10} = 1_2$



Число $127_{10} = 1111111_2$



Диапазон представимых чисел: 0.. 2^{k-1}– 1

3.2.2 Отрицательные целые числа

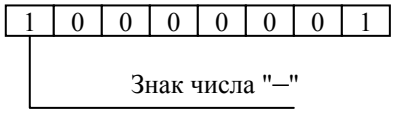
Отрицательные числа в прямом, обратном и дополнительном кодах имеют разное изображение.

3.2.2.1 Прямой код отрицательных чисел

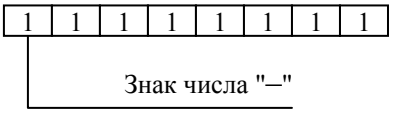
В знаковый разряд помещается цифра 1, а в разряды цифровой части числа — двоичный код его абсолютной величины.

Пример (при $k = 8$):

Прямой код числа –1



Прямой код числа –127



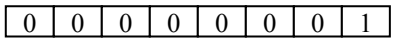
Диапазон представимых чисел: $-(2^{k-1}-1)..0$

3.2.2.2 Обратный код отрицательных чисел

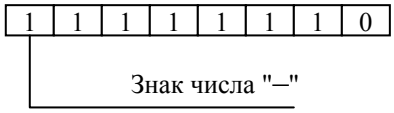
Получается инвертированием всех цифр двоичного кода абсолютной величины числа, включая разряд знака: нули заменяются единицами, а единицы — нулями. Пример ($k = 8$):

Число: –1

Код модуля числа:

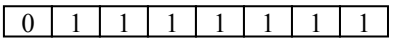


Обратный код числа:

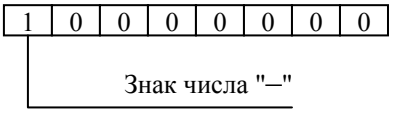


Число: –127

Код модуля числа:



Обратный код числа:

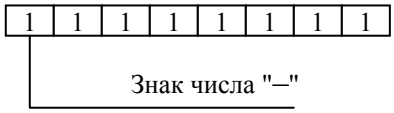


Диапазон представимых чисел: $-(2^{k-1}-1)..0$

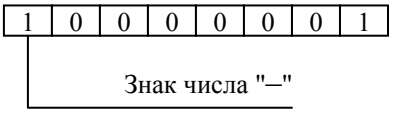
3.2.2.3 Дополнительный код отрицательных чисел

Получается образованием обратного кода с последующим прибавлением единицы к его младшему разряду. Например:

Дополнительный код числа –1:



Дополнительный код числа –127:



Диапазон представимых чисел: $-2^{k-1}.. -1$

Заметим, что ноль имеет два представления в прямом и обратном коде, а в дополнительном коде представление нуля единственно.

3.2.3 Вычисление обратного и дополнительного кодов

Один и тот же битовый набор длины k можно интерпретировать по-разному: 1) как представление некоторого числа без знака; 2) как представление некоторого числа со знаком (в прямом, обратном или дополнительном коде). Компьютер не знает, что именно представляет тот или иной битовый набор — для него это просто слово в

алфавите {0,1}, а смысл этого слова известен программисту. Числовым значением такого слова будем называть неотрицательные целое, двоичная (*k*-разрядная) запись которого совпадает с данным словом.

Пусть x — число со знаком. Тогда числовое значение его обратного и дополнительного кодов можно определить с помощью функций *обр (x)* и *доп(x)* :

$$обр(x) = \begin{cases} x, \text{ если } x \geq 0 \\ 2^k - 1 - |x|, \text{ если } x < 0 \end{cases} \qquad доп(x) = \begin{cases} x, \text{ если } x \geq 0 \\ 2^k - |x|, \text{ если } x < 0 \end{cases}$$

Например, при $k = 8$ $обр(+1) = доп(+1) = 1 = 00000001_2$;

$обр(-127) = 255 - 127 = 128 = 10000000_2$; $доп(-127) = 256 - 127 = 129 = 10000001_2$;

$обр(-1) = 255 - 1 = 254 = 11111110_2$; $доп(-1) = 256 - 1 = 255 = 11111111_2$;

$доп(-128) = 256 - 128 = 128 = 10000000_2$. В обратном коде число –128 не представимо.

Ниже приведена таблица, демонстрирующая различные интерпретации битовых наборов длины 3.

Что представляет / Битовый набор (k =3)	Беззнаковое целое	Знаковое целое в прямом коде	Знаковое целое в обратном коде	Знаковое целое в дополнительном коде
0 0 0	0	+0	+0	+0
0 0 1	1	+1	+1	+1
0 1 0	2	+2	+2	+2
0 1 1	3	+3	+3	+3
1 0 0	4	–0	–3	–4
1 0 1	5	–1	–2	–3
1 1 0	6	–2	–1	–2
1 1 1	7	–3	–0	–1

3.2.4 Диапазоны значений целых чисел

Целые числа обычно занимают в памяти компьютера один, два или четыре байта. В суперкомпьютерах могут быть и более «длинные» целые.

Формат числа в байтах	Диапазон			
	Запись с порядком		Обычная запись	
	Со знаком	Без знака	Со знаком	Без знака
1	$-2^7 .. 2^7-1$	$0..2^8-1$	–128 .. 127	0..255

2	$-2^{15} .. 2^{15}-1$	$0..2^{16}-1$	–32768 .. 32767	0..65535
4	$-2^{31} .. 2^{31}-1$	$0..2^{32}-1$	–2147483648 .. 2147483647	0..4294967295

3.3 Арифметические действия над целыми числами

Обратный и дополнительный коды применяются особенно широко, так как позволяют упростить конструкцию арифметико-логического устройства (АЛУ) компьютера путем замены некоторых арифметических операций сложением.

Обычно десятичные числа при вводе в машину автоматически преобразуются в двоичный код (целые без знака), обратный или дополнительный код (целые со знаком) и в таком виде хранятся, перемещаются и участвуют в операциях. При выводе результатов из машины происходит обратное преобразование в десятичные числа.

3.3.1 Сложение и вычитание

3.3.1.1 Сложение и вычитание чисел без знака

Сложение и вычитание беззнаковых чисел происходит по обычным для позиционных систем счисления алгоритмам. Примеры (для k =3):

$$001_2+100_2=101_2; \qquad 101_2-010_2=011_2.$$

Ситуации, когда уменьшаемое меньше вычитаемого или когда результат суммы не умещается в k разрядов, считаются ошибочными и должны отслеживаться устройством компьютера. Реакция на такие ошибки может быть различной в разных типах компьютеров.

3.3.1.2 Сложение и вычитание чисел со знаком в обратном коде

Сложение в обратном коде происходит следующим образом: по обычному алгоритму складываются все разряды, включая знаковый. Результат такого сложения для k-разрядных наборов имеет длину k+1 (самый левый разряд результата равен единице, если был перенос при сложении старших разрядов операндов, иначе – нулю). Значение левого k+1-го разряда добавляется к младшему разряду результата. Получаем k-разрядный набор, который и будет суммой двух чисел в обратном коде.

Пример (k =3): $+3_{10}+(-1_{10})=011_2+110_2=\mathbf{1}\,001_2 \Rightarrow 001_2+1=010_2=+2_{10}$.

Вычитание чисел в обратном коде $x-y$ сводится к сложению $x+(-y)$.

3.3.1.3 Сложение и вычитание чисел со знаком в дополнительном коде

В дополнительном коде сложение происходит так: по обычному алгоритму складываются все разряды, включая знаковый; единица переноса в k+1-й разряд отбрасывается (т.е. сложение по модулю 2^k).

Пример (k =3): $+3_{10}+(-1_{10})=011_2+111_2=\mathbf{1}010_2 \Rightarrow 010_2=+2_{10}$.

При вычитании тоже действует обычный алгоритм, причем если уменьшаемое меньше вычитаемого, к двоичному коду уменьшаемого слева приписывается единица

(т.е. добавляется 2^k) и только после этого производится вычитание (такой способ называется вычитание по модулю 2^k).

Пример (k =3): $1_{10} - 3_{10} = 001_2 - 011_2 \Rightarrow \mathbf{1}001_2 - 011_2 = 110_2 = -2_{10}$.

Если x и y — числовые значения дополнительного кода знаковых чисел, то числовые значения дополнительных кодов суммы и разности определяются по следующим формулам:

$$(x+y) \bmod 2^k = \begin{cases} x+y, \textit{если } x+y < 2^k, \\ x+y-2^k, \textit{если } x+y \geq 2^k \end{cases}$$

$$(x-y) \bmod 2^k = \begin{cases} x-y, \textit{если } x \geq y, \\ (2^k + x) - y, \textit{если } x < y \end{cases}$$

3.3.2 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.

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

3.3.3 Ошибки при выполнении арифметических операций

При выполнении арифметических операций могут возникать ситуации, когда старшие разряды результата операции не помещаются в отведенной для него области памяти. Ниже приводятся примеры ошибочных вычислений (k =3).

Сложение знаковых чисел в обратном коде:

$-3_{10} + (-2_{10}) = 100_2 + 101_2 = \mathbf{1}001_2 \Rightarrow 001_2 + 1 = 010_2 = +2_{10}$

Вычитание знаковых чисел в обратном коде:

$+2_{10} - (-3_{10}) = 010_2 - 101_2 \Rightarrow \mathbf{1}010_2 - 101_2 = 101_2 = -3_{10}$.

Такая ситуация называется переполнением цифровой части (мантиссы) формата числа. Для обнаружения переполнения и оповещения о возникшей ошибке в компьютере используются специальные средства. Реакция на разные ошибки может быть разная. Так, в некоторых ЭВМ при делении на ноль вычисления прекращаются (фатальная ошибка), а при переполнении мантиссы устанавливается признак переполнения в так называемом регистре флагов и вычисления продолжаются.

4 Представление вещественных чисел

Вещественными числами (в отличие от целых) в компьютерной технике называются числа, имеющие дробную часть. При их изображении во многих языках программирования вместо запятой принято ставить точку. Так, например, число 5 — целое, а числа 5.1 и 5.0 — вещественные. Для удобства отображения чисел, принимающих значения из достаточно широкого диапазона (то есть, как очень маленьких, так и очень больших), используется форма записи чисел с порядком основания системы счисления. Например, десятичное число 1.75 можно в этой форме представить так:

$$1.75 \cdot 10^0 = 0.175 \cdot 10^1 = 0.0175 \cdot 10^2 = \ldots ,$$

или так:

$$17.5 \cdot 10^{-1} = 175.0 \cdot 10^{-2} = 1750.0 \cdot 10^{-3} = \ldots .$$

Любое число N в системе счисления с основанием q можно записать в виде $N = M \cdot q^p$, где M называется *мантиссой* числа, а p — *порядком*. Такой способ записи чисел называется представлением с плавающей точкой. Если “плавающая” точка расположена в мантиссе перед первой значащей цифрой, то при фиксированном количестве разрядов, отведённых под мантиссу, обеспечивается запись максимального количества значащих цифр числа, то есть максимальная точность представления числа в машине. Из этого следует, что мантисса должна быть правильной дробью, первая цифра которой отлична от нуля: $M \in [0.1, 1)$. Такое, наиболее выгодное для компьютера, представление вещественных чисел называется нормализованным. Мантиссу и порядок q-ичного числа принято записывать в системе с основанием q, а само основание — в десятичной системе.

4.1 Примеры нормализованного представления:

Десятичная система Двоичная система

$752.15 = 0.75215 \cdot 10^3$; $-101.01 = -0.10101 \cdot 2^{11}$ (порядок $11_2 = 3_{10}$)

$-0.000039 = -0.39 \cdot 10^{-4}$; $-0.000011 = 0.11 \cdot 2^{-100}$ (порядок $-100_2 = -4_{10}$)

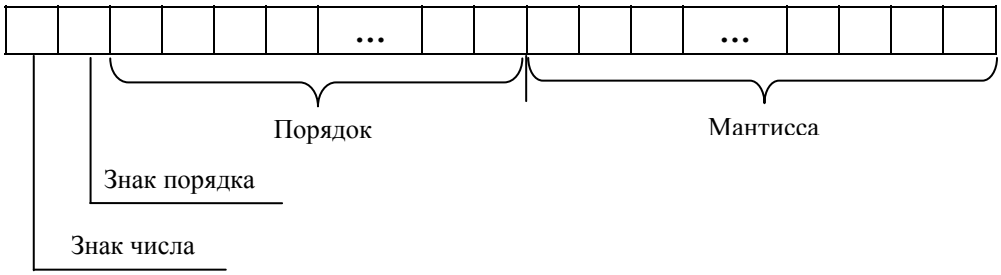
Вещественные числа в компьютерах различных типов записываются по-разному. При этом компьютер обычно предоставляет программисту возможность выбора из нескольких числовых форматов наиболее подходящего для конкретной задачи — с использованием четырех, шести, восьми или десяти байтов.

В качестве примера приведем характеристики форматов вещественных чисел, используемых IBM-совместимыми персональными компьютерами:

Форматы вещественных чисел	Размер в байтах	Примерный диапазон абсолютных значений	Количество значащих десятичных цифр
Одинарный	4	$10^{-45} \ldots 10^{38}$	7 или 8
Вещественный	6	$10^{-39} \ldots 10^{38}$	11 или 12
Двойной	8	$10^{-324} \ldots 10^{308}$	15 или 16
Расширенный	10	$10^{-4932} \ldots 10^{4932}$	19 или 20

4.2 Представление в виде набора битов

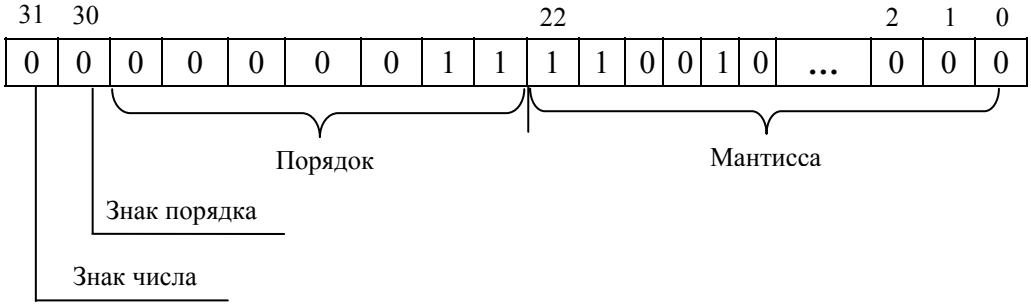
Числа с плавающей точкой представляются в виде битовых наборов, в которых отводяся разряды для мантиссы, порядка, знака числа и знака порядка:



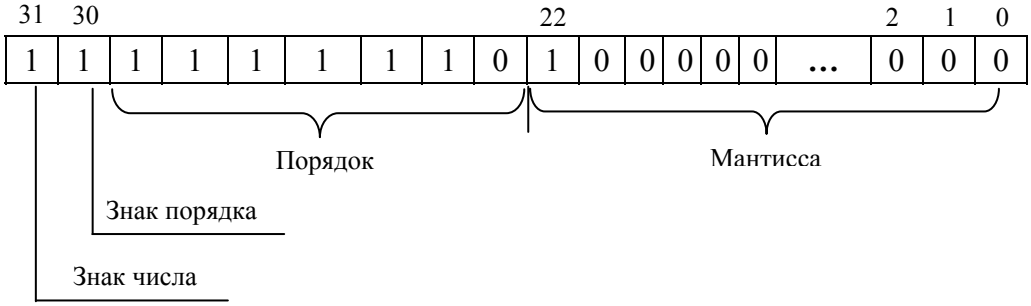
Чем больше разрядов отводится под запись мантиссы, тем выше точность представления числа. Чем больше разрядов занимает порядок, тем шире диапазон от наименьшего отличного от нуля числа до наибольшего числа, представимого в машине при заданном формате.

Покажем на примерах, как записываются некоторые числа в нормализованном виде в четырехбайтовом формате с семью разрядами для записи порядка.

Число $6.25_{10} = 110.01_2 = 0.11001 \cdot 2^{11}$:



Число $-0.125_{10} = -0.001_2 = -0.1 \cdot 2^{-10}$ (отрицательный порядок записан в дополнительном коде):



4.3 Арифметические действия над нормализованными числами

К началу выполнения арифметического действия операнды операции помещаются в соответствующие регистры АЛУ.

При сложении и вычитании сначала производится подготовительная операция, называемая выравниванием порядков. В процессе выравнивания порядков мантисса числа с меньшим порядком сдвигается в своем регистре вправо на количество разрядов, равное разности порядков операндов. После каждого сдвига порядок увеличивается на единицу.

В результате выравнивания порядков одноименные разряды чисел оказываются расположенными в соответствующих разрядах обоих регистров, после чего мантиссы складываются или вычитаются.

В случае необходимости полученный результат нормализуется путем сдвига мантиссы результата влево. После каждого сдвига влево порядок результата уменьшается на единицу.

Пример 1. Сложить двоичные нормализованные числа $0.10111 \cdot 2^{-1}$ и $0.11011 \cdot 2^{10}$. Разность порядков слагаемых здесь равна трем, поэтому перед сложением мантисса первого числа сдвигается на три разряда вправо:

$$\begin{array}{rl} & 0.00010111 \cdot 2^{10} \\ + & \underline{0.11011 \quad \cdot 2^{10}} \\ & 0.11101111 \cdot 2^{10} \end{array}$$

Пример 2. Выполнить вычитание двоичных нормализованных чисел $0.10101 \cdot 2^{10}$ и $0.11101 \cdot 2^{1}$. Разность порядков уменьшаемого и вычитаемого здесь равна единице, поэтому перед вычитанием мантисса второго числа сдвигается на один разряд вправо:

$$\begin{array}{rl} & 0.10101 \quad \cdot 2^{10} \\ - & \underline{0.011101 \cdot 2^{10}} \\ & 0.001101 \cdot 2^{10} \end{array}$$

Результат получился не нормализованным, поэтому его мантисса сдвигается влево на два разряда с соответствующим уменьшением порядка на две единицы: $0.1101 \cdot 2^{0}$.

При умножении двух нормализованных чисел их порядки складываются, а мантиссы перемножаются.

Пример 3. Выполнить умножение двоичных нормализованных чисел:

$$(0.11101 \cdot 2^{101}) \cdot (0.1001 \cdot 2^{11}) = (0.11101 \cdot 0.1001) \cdot 2^{(101+11)} = 0.100000101 \cdot 2^{1000}$$

При делении двух нормализованных чисел из порядка делимого вычитается порядок делителя, а мантисса делимого делится на мантиссу делителя. Затем в случае необходимости полученный результат нормализуется.

Пример 4. Выполнить деление двоичных нормализованных чисел:

$$0.1111 \cdot 2^{100} : 0.101 \cdot 2^{11} = (0.1111 : 0.101) \cdot 2^{(100-11)} = 1.1 \cdot 2^{1} = 0.11 \cdot 2^{10}$$

Использование представления чисел с плавающей точкой существенно усложняет схему арифметико-логического устройства.