5.3 ПРЕДСТАВЛЕНИЕ ЧИСЕЛ

Компьютер работает как с целыми, так и с дробными числами. До настоящего момента мы рассматривали только представления знаковых и беззнаковых целых чисел, которые были описаны в **параграфе 1.4**. В данном разделе вводится представление чисел с фиксированной и с плавающей точкой, с помощью которого можно представить рациональные числа. Числа с фиксированной точкой – это аналог десятичных чисел; некоторые биты представляют целую часть, а оставшиеся – дробную. Числа с плавающей точкой являются аналогом экспоненциального представления числа с мантиссой и порядком (прим. переводчика: в англоязычных странах в качестве разделителя целой и дробной частей чисел используется точка, а не запятая. Так сложилось, что в современной русскоязычной технической литературе термин «точка» используется чаще, поэтому и мы будем его использовать. В некоторых других книгах используется термин «запятая»).

5.3.1 Числа с фиксированной точкой

Представление «с фиксированной точкой» подразумевает двоичную запятую между битами целой и дробной части, аналогично десятичной точке между целой и дробной частями обычного десятичного числа.

Например, на **Рис. 5.21 (a)** показано число с фиксированной точкой с 4-мя битами целой части и 4-мя дробной.

На **Рис. 5.21 (b)** голубым цветом показана двоичная запятая, а на **Рис. 5.21 (c)** изображено эквивалентное десятичное число.

Знаковые числа с фиксированной точкой можно использовать как в прямом, так и в дополнительном коде.

На **Рис. 5.22** показаны оба представления числа –2.375 с фиксированной запятой с использованием 4-х целых бит и 4-х дробных бит. Неявная двоичная запятая для ясности изображена голубым цветом.

(a) 01101100

(b) 0110.1100

(c) $2^2+2^1+2^{-1}+2^{-2}=6.75$

Рис. 5.21 Представление числа 6.75 с фиксированной точкой с четырьмя битами целой части и четырьмя дробной

(a) 0010.0110

(b) 1010.0110

(c) 1101.1010

Рис. 5.22 Представление числа –2.375 с фиксированной точкой: (a) абсолютное значение, (b) прямой код, (c) дополнительный код

В прямом коде знаковый бит используется для указания знака. Дополнительный код двоичного числа получается инверсией битов

абсолютного значения и добавления 1 к младшему разряду. В этом примере младший разряд соответствует 2^{-4}.

Как и все представления двоичных чисел, числа с фиксированной точкой являются лишь набором бит. Не существует способа узнать о существовании двоичной точки кроме как из соглашения между людьми, интерпретирующими число.

Пример 5.4 АРИФМЕТИЧЕСКИЕ ОПЕРАЦИИ С ЧИСЛАМИ С ФИКСИРОВАННОЙ ТОЧКОЙ

Вычислим выражение 0.75 ±0.625, используя числа с фиксированной точкой.

Решение: Сначала преобразуем 0.625, абсолютное значение второго числа, в стандартное представление двоичного числа с фиксированной точкой. $0.625 \geq 2^{-1}$, следовательно, ставим 1 в разряд 2^{-1}, оставляя $0.625 – 0.5 = 0.125$. Так как $0.125 < 2^{-2}$, то ставим 0 в разряд 2^{-2}. Так как $0.125 \geq 2^{-3}$, то ставим 1 в разряд 2^{-3}, оставляя $0.125 – 0.125 = 0$. Таким образом, в разряде 2^{-4} будет 0. Таким образом, $0.62510 = 0000.1010_2$.

На **Рис. 5.23** показано преобразование числа –0.625 в двоичное представление в дополнительном коде. На **Рис. 5.24** показано сложение чисел с фиксированной запятой и, для сравнения, десятичный эквивалент.

Заметьте, что первый единичный бит в двоичном представлении числа с фиксированной точкой на **Рис. 5.24 (а)** отброшен в 8-битовом результате.

0000.1010	Binary Magnitude
1111.0101	One's Complement
+ 1	Add 1
1111.0110	Two's Complement

Рис. 5.23 **Представление числа в дополнительном коде**

0000.1100	0.75
+ 1111.0110	+ (–0.625)
10000.0010	0.125
(a)	**(b)**

Рис. 5.24 **Сложение: (a) двоичных чисел с фиксированной точкой, (b) десятичный эквивалент**

Для корректных вычислений с использованием чисел с фиксированной точкой используется двоичное представление в дополнительном коде.

5.3.2 Числа с плавающей точкой

$\pm M \times B^E$

Рис. 5.25 **Числа с плавающей точкой**

Числа с плавающей точкой соответствуют экспоненциальному представлению. В этом представлении преодолены ограничения наличия только фиксированного количества целых и дробных бит, поэтому оно позволяет представлять очень большие и очень маленькие числа. Как и в экспоненциальном представлении, числа с плавающей запятой имеют знак, мантиссу (M), основание (B) и порядок (E), что показано на **Рис. 5.25**.

К примеру, число $4.1*10^3$ является десятичным экспоненциальным представлением числа 4100. Мантиссой является 4.1, основание

равно 10, а порядок равен 3. Десятичная запятая «переплывает» на позицию правее самого значимого (старшего) разряда. У чисел с плавающей точкой основание будет равно 2, а мантисса будет двоичным числом. 32 бита используются для представления 1 знакового бита, 8 бит порядка и 23 бит мантиссы.

Пример 5.5 32-БИТНОЕ ЧИСЛО С ПЛАВАЮЩЕЙ ТОЧКОЙ

Найдите представление десятичного числа 228 в виде числа с плавающей точкой.

Решение: Для начала преобразуем десятичное число в двоичное:

$228_{10} = 11100100_2 = 1.11001_2 * 2^7$. На **Рис. 5.26** показано 32-битное кодирование, которое далее для эффективности будет модифицировано. Знаковый бит положительный, равен 0, 8 бит порядка дают значение 7, а оставшиеся 23 бита – это мантисса.

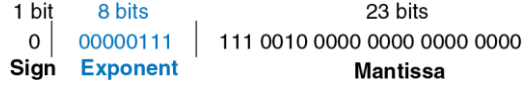

Рис. 5.26 **32-разрядной кодирование числа с плавающей точкой: версия 1**

В двоичных числах с плавающей точкой первый бит мантиссы (слева от точки) всегда равен 1, и поэтому его можно не сохранять.

Это называется неявная старшая единица. На **Рис. 5.27** изображено модифицированное представление:

$228_{10} = 11100100_2 * 2^0 = 1.11001_2 * 2^7$. Неявная старшая единица не входит в 23 бита мантиссы. Сохраняются только дробные биты. Это освобождает дополнительный бит для полезных данных.

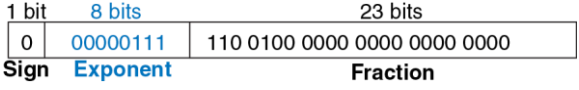

Рис. 5.27 Кодирование числа с плавающей точкой: версия 2

Сделаем последнюю модификацию представления порядка. Порядок должен представлять, как положительный показатель степени, так и отрицательный. Для этого в формате с плавающей точкой используется смещенный порядок, который представляет собой первоначальный порядок плюс постоянное смещение. 32-битное представление с плавающей точкой использует смещение 127. Например, для порядка 7, смещенный порядок будет выглядеть так: $7 + 127 = 134 = 10000110_2$, для порядка -4 смещенный порядок равен- $4 + 127 = 123 = 01111011_2$.

На **Рис. 5.28** показано представление числа $1.11001_2 * 2^7$ в формате с плавающей точкой с неявной старшей единицей и смещенным порядком 134 (7 + 127). Это представление соответствует стандарту IEEE 754.

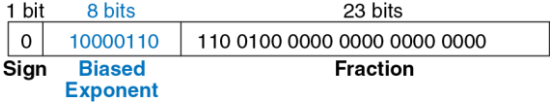

Рис. 5.28 Представление числа с плавающей точкой по стандарту IEEE 754

Особые случаи: 0, ±∝ и NaN

Стандарт IEEE для чисел с плавающей точкой включает в себя особые случаи представления таких чисел, как 0, бесконечность и недопустимые результаты. К примеру, представить число 0 в виде числа с плавающей точкой невозможно из-за наличия неявной старшей единицы. Для этих случаев зарезервированы специальные коды: порядок состоит только из нулей или единиц. В **Табл. 5.2** показано обозначение 0, ±∝, NaN. Как и в знаковых числах, плавающая запятая имеет и положительный и отрицательный 0. NaN используется для чисел, которые не существуют, например корень из -1 и $\log_2(-5)$.

Табл. 5.2 Обозначение 0, ±∞ и NaN в соответствии со стандартом IEEE 754

Number	Sign	Exponent	Fraction
0	X	00000000	00000000000000000000000
∝	0	11111111	00000000000000000000000
±∝	1	11111111	00000000000000000000000
NaN	X	11111111	Non-zero

Очевидно, что существует много разумных способов представления чисел с плавающей точкой. Много лет производители компьютеров использовали несовместимые форматы. Результат от одного компьютера не мог быть непосредственно интерпретирован другим. Институт инженеров электротехники и электроники (Institute of Electricaland Electronics Engineers, IEEE) решил эту проблему, определив в 1985 году стандарт IEEE754. Сейчас этот формат используется повсеместно, и он будет обсуждаться в данном разделе.

Форматы одинарной и двойной точности

Ранее мы рассматривали 32-битные числа с плавающей точкой. Такой формат также называют форматом одинарной точности. Стандарт IEEE 754 также определяет 64-битные числа двойной точности, которые позволяют представить большой диапазон чисел с большой точностью. В **Табл. 5.3** приведено число бит, используемых в полях разных форматов.

Табл. 5.3 Числа с плавающей точкой с одинарной и двойной точностью

Format	Total Bits	Sign Bits	Exponent Bits	Fraction Bits
single	32	1	8	23
double	64	1	11	52

Если исключить специальные случаи, упомянутые ранее, обычные числа одинарной точности охватывают диапазон от $\pm 1.175494 * 10^{-38}$ до

±3.402824 * 10^{38}. Их точность составляет около 7 десятичных разрядов, так как $2^{-24} \approx 10^{-7}$. Числа с двойной точностью охватывают диапазон от ±2.22507385850720 * 10^{-308} до ±1.79769313486232 * 10^{308} и имеют точность около 15 десятичных разрядов.

Некоторые числа нельзя точно представить в виде числа с плавающей точкой, как, например, 1.7. Однако, когда вы вводите 1.7 на калькуляторе, вы видите точно 1.7, не 1.69999… Для этого большинство приложений, как например калькулятор и различные финансовые программы, используют двоично-десятичный формат (BCD) или формат с основанием 10. Числа в таком формате кодируют каждый десятичный разряд с помощью 4 бит со значением от 0 до 9. Например, число 1.7 в формате BCD с четырьмя целыми и четырьмя дробными битами представляет собой 0001.0111. Конечно, не все так просто. Ценой является усложнение арифметических схем и неполное использование кодировки (не используются кодировки A-F), следовательно, снижается эффективность. Таким образом, для ресурсоемких приложений числа в формате с плавающей точкой гораздо эффективнее.

Округление

Арифметические результаты, которые выходят за пределы доступной точности, необходимо округлять до наиболее близких чисел. Существуют следующие способы округления: округление в меньшую сторону (1), округление в большую сторону (2), округление до нуля (3) и

округление к ближайшему числу (4). По умолчанию принято округление к ближайшему числу. В этом случае, если два числа находятся на одинаковом расстоянии, то выбирается то, у которого будет ноль в младшем разряде дробной части.

Напомним, что число переполняется, когда его величина слишком велика для какого-либо представления. Аналогично, число является изчезающе малым, когда оно слишком мало для представления. При округлении (4) переполненные числа округляются до $\pm\infty$, а изчезающе малые округляются до нуля.

Сложение чисел с плавающей точкой

Сложение чисел с плавающей точкой не такая простая операция, как в случае представления чисел в дополнительном коде. Для выполнения сложения двух таких чисел необходимо выполнить следующие шаги:

1. Выделить биты порядка и мантиссы.
2. Присоединить неявную старшую единицу к мантиссе.
3. Сравнить порядки.
4. При необходимости сдвинуть мантиссу числа, имеющего меньший порядок.
5. Сложить мантиссы.

6. При необходимости нормализовать мантиссу и порядок.
7. Округлить результат.
8. Собрать обратно порядок и мантиссу в итоговое число с плавающей точкой.

На **Рис. 5.29** показан процесс сложения чисел с плавающей точкой 7.875 ($1.11111 * 2^2$) и 0.1875 ($1.1 * 2^{-3}$). Результат равен 8.0625 ($1.0000001 * 2^3$). После извлечения мантиссы и порядка, присоединения неявной старшей единицы (шаги 1 и 2), порядки сравнивается путем вычитания меньшего порядка из большего. Результатом будет число бит, на которое необходимо сдвинуть мантиссу меньшего число вправо (шаг 4) для выравнивания двоичной точки (т.е. чтобы сделать порядки равными). Выровненные значения складываются. Так как мантисса суммы больше или равна 2.0, результат нужно нормализовать, сдвинув его вправо на 1 бит и увеличить порядок на 1. В этом пример результат точный и никаких округлений не требуется. Он сохраняется в формате с плавающей точкой, после удаления неявной старшей единицы мантиссы и добавления знакового бита.

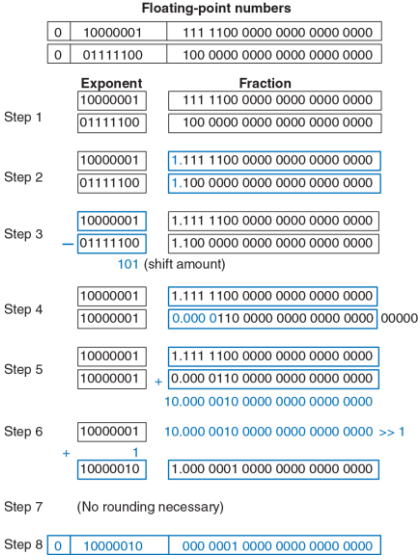

Рис. 5.29 Сложение чисел с плавающей точкой

Вычисления при использовании чисел в формате с плавающей точкой обычно выполняются с помощью специальных аппаратных средств для увеличения скорости. Такая аппаратура называется FPU (floating-point unit), она, как правило, отличается от CPU (central processing unit). Печально известный баг FDIV (floating-point division) в FPU процессора Pentium стоил компании Intel $475 миллионов, которые она вынуждена была потратить на отзыв и замену дефектных микросхем. Ошибка произошла только потому, что не была правильно загружена таблица преобразования.