

IEEE 754-2019

Стандарт IEEE для арифметики с плавающей запятой (IEEE 754) — это технический стандарт для арифметики с плавающей запятой, установленный в 1985 году Институтом инженеров по электротехнике и электронике (IEEE). Стандарт решил многие проблемы, обнаруженные в различных реализациях с плавающей запятой, которые затрудняли их надежное и переносимое использование. Используется в программных (компиляторы разных языков программирования) и аппаратных (CPU и FPU) реализациях арифметических действий (математических операций).

Стандарт определяет:

арифметические форматы: наборы двоичных и десятичных данных с плавающей запятой, которые состоят из конечных чисел (включая нули со знаком и субнормальные числа), бесконечности и специальных значений «не числа» (NaN)
форматы обмена: кодировки (битовые строки), которые могут использоваться для обмена данными с плавающей запятой в эффективной и компактной форме.
правила округления: свойства, которые должны выполняться при округлении чисел во время арифметических операций и преобразований.
операции: арифметические и другие операции (например, тригонометрические функции) над арифметическими форматами.
обработка исключений: индикация исключительных условий (таких как деление на ноль, переполнение и т. д.)

Разработка стандарта

IEEE 754-2008, опубликованный в августе 2008 г., включает почти весь исходный стандарт IEEE 754-1985, а также стандарт IEEE 854-1987 для независимой от системы счисления арифметики с плавающей запятой. Текущая версия IEEE 754-2019 была опубликована в июле 2019 г. Это доработанная версия предыдущей версии, включающая в основном уточнения, исправления дефектов и новые рекомендуемые операции. Текущая версия IEEE 754—2008 была опубликована в 2008 году. Международный стандарт ISO/IEC/IEEE 60559:2011 (с идентичным IEEE 754—2008) был одобрен и опубликован для JTC1/SC 25 под соглашением ISO/IEEE PSDO. Бинарные форматы в первоначальном стандарте включены в новый стандарт наряду с тремя новыми основными форматами (одним бинарным и двумя десятичными). Для того, чтобы соответствовать текущему стандарту, реализация должна реализовать по крайней мере один из основных форматов.

Стандартная разработка

Первый стандарт для арифметики с плавающей запятой, IEEE 754-1985, был опубликован в 1985 году. Он охватывал только двоичную арифметику с плавающей запятой.

Новая версия IEEE 754-2008 была опубликована в августе 2008 г. после семилетнего процесса пересмотра под председательством Дэна Зурса и под редакцией Майка Каулишоу. Он заменил как IEEE 754-1985 (двоичная арифметика с плавающей запятой), так и стандарт IEEE 854-1987 для независимой от системы счисления

арифметики с плавающей запятой. Двоичные форматы исходного стандарта включены в этот новый стандарт вместе с тремя новыми базовыми форматами, одним двоичным и двумя десятичными. Чтобы соответствовать текущему стандарту, реализация должна реализовать по крайней мере один из основных форматов как арифметический формат, так и формат обмена.

Международный стандарт ISO/IEC 60559:2020 (с содержанием, идентичным IEEE 754-2019) был одобрен для принятия через ISO/IEC JTC 1/SC 25 и опубликован.

Следующая запланированная редакция стандарта запланирована на 2028 год.

Формат

Формат IEEE 754 представляет собой «совокупность представлений числовых значений и символов». Формат может также включать в себя способ кодирования.

Формат с плавающей запятой определяется

основание b (b может быть 2 или 10);

точность p ;

диапазон показателей степени от e_{\min} до e_{\max} , где $e_{\min} = 1 - e_{\max}$ для всех форматов IEEE 754.

Формат включает:

Числа, которые могут рассматриваться в двоичной или десятичной системе счисления.

Вещественное число представляется тремя целыми числами s , c и q , где s — знак (0 для положительного и 1 для отрицательного), c — мантисса (коэффициент), имеющая не более p цифр при записи по основанию b (т. е. целое число в диапазоне от 0 до $b^p - 1$), q — экспонента. Для заданных целых чисел s , c и q значением соответствующего вещественного числа является: $(-1)^s \times c \times b^q$.

Например, число с основанием 10, битом знака 1 (число отрицательное), мантиссой 12345 и экспонентой -3 определяют число

$$(-1)^1 \times 12345 \times 10^{-3} = -12.345.$$

Положительный ноль $+0$ и отрицательный ноль -0 .

Две бесконечности: $+\infty$ и $-\infty$.

Например, если $b = 10$, $p = 7$ и $e_{\max} = 96$, то $e_{\min} = -95$, мантиссы удовлетворяют условию $0 \leq c \leq 9999999$, а показатель степени удовлетворяет условию $-101 \leq q \leq 90$.

Два вида NaN (не число): тихий NaN (qNaN) и сигнальный NaN (sNaN).

Возможные конечные значения, которые могут быть представлены в формате,

определяются основанием b , числом знаков в мантиссе (с точностью p) и

максимальным значением E_{\max} :

c должен быть целым числом в диапазоне от нуля до $b^p - 1$ (если $b = 10$ $p = 7$ тогда c может быть от 9999999)

q должно быть целым числом, чтобы $1 - E_{\max} \leq q + p - 1 \leq E_{\max}$ (если $p = 7$ и $E_{\max} = 96$, то q может быть от -101 до 90).

Следовательно, наименьший ненулевой положительное число, которое может быть представлено, равно 1×10^{-101} , а наибольшее — 9999999×10^{90} ($9,999999 \times 10^{96}$), поэтому полный диапазон чисел составляет от $-9,999999 \times 10^{96}$ до $9,999999 \times 10^{96}$.

Числа $-b^{-E_{\max}}$ и $b^{-E_{\max}}$ (здесь -1×10^{-95} и 1×10^{-95}) являются наименьшими (по величине) нормальными числами; ненулевые числа между этими наименьшими числами называются субнормальными числами.

Представление и кодирование в памяти

Некоторые числа могут иметь несколько представлений в формате, в котором они были только что описаны. Например, если $b = 10$ и $p = 7$, то $-12,345$ может быть представлено как -12345×10 , -123450×10 и -1234500×10 . Однако для большинства операций, таких как арифметические операции, результат (значение) не зависит от представления входных данных.

Для десятичных форматов допустимо любое представление, и набор этих представлений называется когортой. Если результат может иметь несколько представлений, стандарт определяет, какой член когорты выбран.

Для двоичных форматов представление становится уникальным путем выбора наименьшей представимой экспоненты, позволяющей точно представить значение. Кроме того, показатель степени не представлен напрямую, но добавляется смещение, так что наименьший представимый показатель представлен как 1, а 0 используется для субнормальных чисел. Для чисел с показателем степени в нормальном диапазоне (поле показателя не может содержать ни все единицы, ни все нули), ведущий бит мантиссы всегда будет равен 1. Следовательно, ведущая единица может подразумеваться, а не присутствовать явно в кодировании памяти, и по стандарту явно представленная часть мантиссы будет находиться между 0 и 1. Это правило называется соглашением о начальных битах, неявным соглашением о битах или соглашением о скрытых битах. Это правило позволяет двоичному формату иметь дополнительную точность. Соглашение о начальных битах не может использоваться для субнормальных чисел, поскольку они имеют показатель степени за пределами диапазона нормального показателя и масштабируются по наименьшей представленной экспоненте, используемой для наименьших нормальных чисел.

Из-за возможности множественного кодирования (по крайней мере, в форматах, называемых форматами обмена), NaN может нести другую информацию: знаковый бит (который не имеет значения, но может использоваться некоторыми операциями) и полезные данные, который предназначен для диагностической информации, указывающей на источник NaN (но полезная нагрузка может иметь другое использование, например NaN-бокс).

Для десятичных форматов любое представление справедливо, и совокупность этих представлений называется когортой. Когда результат может иметь несколько представлений, стандарт определяет, какой выбран членом когорты.

Для бинарных форматов представление делается уникальным путём выбора наименьшего представляемого показателя. Для чисел с показателем в нормальном диапазоне (не все из них или все нули), ведущий бит мантиссы всегда будет равен 1. Следовательно, ведущий 1 бит может подразумеваться, а не сохраняться явно в памяти. Это правило называется ведущей битной конвенцией или скрытой битной конвенцией. Правило позволяет сберечь 1 бит памяти, чтобы иметь ещё один бит точности. Ведущий бит конвенции не используется для субнормальных чисел; их показатель находится за пределами нормального диапазона значений.

Базовые форматы и форматы обмена

Стандарт определяет пять основных форматов, названных по их числовой базе и количеству битов, используемых при их обменном кодировании. Существует три основных двоичных формата с плавающей запятой (с 32-, 64- или 128-битной

кодировкой) и два основных десятичных формата с плавающей запятой (с 64- или 128-битной кодировкой). Форматы binary32 и binary64 представляют собой одинарный и двойной форматы IEEE 754-1985 соответственно. Соответствующая реализация должна полностью реализовывать хотя бы один из основных форматов.

Стандарт также определяет форматы обмена, которые обобщают эти базовые форматы. Для двоичных форматов требуется соглашение о начальных битах. В следующей таблице приведены самые маленькие форматы обмена (включая основные).

Имя	Общее имя	Базовое	Знаки и биты или цифры	Десятичные цифры	Биты экспоненты
двоичное16	половинная точность	2	11	3,31	5
двоичный32	одинарная точность	2	24	7,22	8
binary64	Двойная точность	2	53	15,95	11
binary128	Четверная точность	2	113	34,02	15
двоичный 256	восьмеричная точность	2	237	71,34	19
десятичный32		10	7	7	7,58
десятичный64		10	16	16	9,58
десятичное 128		10	34	34	13,58

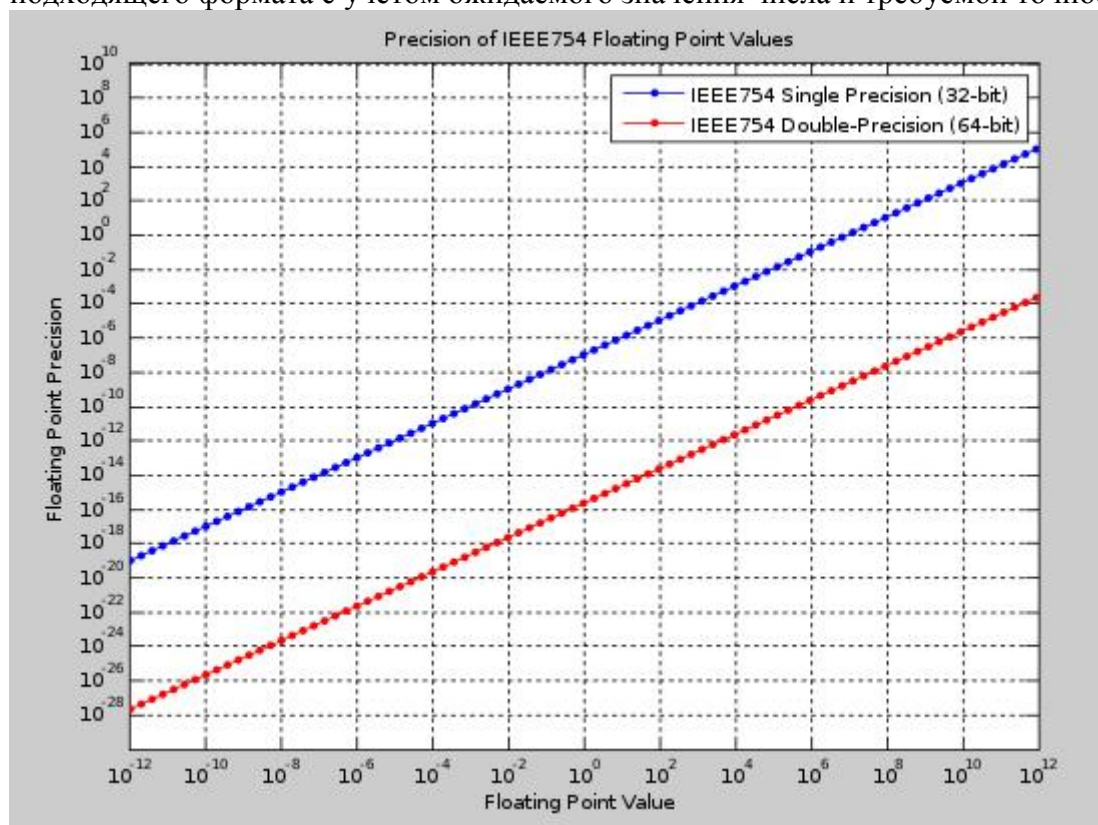
Имя	Десятичное E макс.	Экспоненциальное смещение	E min	E max	Примечания
двоичное16	4,51	$2^{-1} = 15$	-14	+15	не базовый
двоичный32	38,23	$2^{-1} = 127$	-126	+127	
binary64	307,95	$2^{-1} = 1023$	-1022	+1023	
binary128	4931,77	$2^{-1} = 16383$	-16382	+16383	
двоичный 256	78913,2	$2^{-1} = 262143$	-262142	+262143	не базовый
десятичный32	96	101	-95	+96	не базовый
десятичный64	384	398	- 383	+384	
десятичное 128	6144	6176	-6143	+6144	

Обратите внимание, что в приведенной выше таблице минимальные показатели степени указаны для нормальных чисел; специальное представление субнормального числа позволяет представлять даже меньшие числа (с некоторой потерей точности). Например, наименьшее положительное число, которое может быть представлено в двоичном формате, равно 2; Вклады в фигуру -1074 включают значение E min -1022 и все, кроме одного, из 53 значащих битов ($2 = 2$).

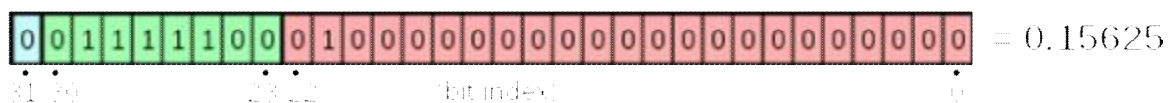
Десятичные цифры - это цифры $\times \log_{10}$ основание. Это дает приблизительную точность в количестве десятичных цифр.

Десятичное E_{\max} равно $E_{\max} \times \log_{10}$ основание. Это дает приблизительное значение максимального десятичного показателя степени.

Форматы binary32 (одиначный) и binary64 (двойной) - два наиболее распространенных формата, используемых сегодня. На рисунке ниже показана абсолютная точность для обоих форматов в диапазоне значений. Этот рисунок может использоваться для выбора подходящего формата с учетом ожидаемого значения числа и требуемой точности.



Пример макета для 32-битной с плавающей запятой
sign: exponent: 8 bits: fraction: 23 bits:



Расширенные и расширяемые форматы точности

Стандарт определяет дополнительные расширенные и расширяемые форматы точности, которые обеспечивают большую точность, чем базовые форматы. Формат расширенной точности расширяет базовый формат за счет большей точности и большего диапазона экспонент. Расширяемый формат точности позволяет пользователю определять точность и диапазон экспоненты. Реализация может использовать любое внутреннее представление, выбранное для таких форматов; все, что необходимо определить, - это его параметры (b , p и e_{\max}). Эти параметры однозначно описывают набор конечных чисел (комбинации знака, значащей и экспоненты для данного основания системы счисления), которые он может представлять.

Стандарт рекомендует, чтобы языковые стандарты обеспечивали метод определения p и e_{\max} для каждой поддерживаемой базы b . Стандарт рекомендует, чтобы языковые стандарты и реализации поддерживали расширенный формат, который имеет большую точность, чем самый большой базовый формат, поддерживаемый для каждого основания b . Для расширенного формата с точностью между двумя основными форматами диапазон экспоненты должен быть таким же большим, как и для следующего более широкого базового формата. Так, например, 64-битное двоичное число с расширенной точностью должно иметь ' e_{\max} ' не менее 16383. Этому требованию отвечает x8780-битный расширенный формат .

Форматы обмена

Форматы обмена предназначены для обмена данными с плавающей запятой с использованием битовой строки фиксированной длины для данного формата.

Двоичный

Для обмена двоичными числами с плавающей запятой определены форматы обмена длиной 16 бит, 32 бита, 64 бита и любое кратное 32 бит ≥ 128 . 16-битный формат предназначен для обмена или хранения небольших чисел (например, для графики).

Схема кодирования для этих двоичных форматов обмена такая же, как и в IEEE 754-1985: знаковый бит, за которым следуют w битов экспоненты, которые описывают смещение экспоненты на смещение e_{bias} , и $p - 1$ бит, описывающий значение. Ширина поля экспоненты для k -битного формата вычисляется как $w = \text{round}(4 \log_2(k)) - 13$. Существующие 64- и 128-битные форматы следуют этому правилу, но 16- и 32-битные форматы имеют больше битов экспоненты (5 и 8 соответственно), чем дает эта формула (3 и 7 соответственно).

Как и в IEEE 754-1985, поле смещенной экспоненты заполняется всеми 1 битами, чтобы указать либо бесконечность (конечное поле значимости = 0), либо NaN (конечное значение значимости 0). Для NaN, тихие NaN и сигнальные NaN различаются исключительно использованием самого старшего бита конечного поля значимости, а полезная нагрузка переносится в оставшихся битах.

Десятичный

Для обмена десятичными числами с плавающей запятой определены форматы обмена, кратные 32 битам. Как и в случае двоичного обмена, схема кодирования для форматов десятичного обмена кодирует знак, показатель степени и значение. Определены две разные кодировки битового уровня, и обмен затруднен тем фактом, что может потребоваться некоторый внешний индикатор используемой кодировки.

Эти две опции позволяют кодировать мантиссу как сжатую последовательность десятичных цифр с использованием плотно упакованного десятичного числа или, альтернативно, как двоичное целое число . Первый более удобен для прямой аппаратной реализации стандарта, а второй больше подходит для программной эмуляции на двоичном компьютере. В любом случае набор чисел (комбинации знака, значащей и экспоненты), который может быть закодирован, идентичен, и специальные значения (\pm ноль с минимальным показателем, \pm бесконечность, тихие NaN и сигнализация NaN) имеют одинаковую кодировку.

Правила округления

Стандарт определяет пять правил округления. Первые два правила округляются до ближайшего значения; остальные называются направленным округлением :

Округлением до ближайшего

Округлением до ближайшего, привязкой к четному - округлением до ближайшего значения; если число выпадает на полпути, оно округляется до ближайшего значения с четной младшей цифрой; это значение по умолчанию для двоичных чисел с плавающей запятой и рекомендуемое значение по умолчанию для десятичных.

Округление до ближайшего, отнесение к нулю - округление до ближайшего значения; если число падает на полпути, оно округляется до ближайшего значения выше (для положительных чисел) или ниже (для отрицательных чисел); это предназначено как опция для десятичных чисел с плавающей запятой.

Направленное округление

Округление в сторону 0 - направленное округление до нуля (также известное как усечение).

Округление в сторону $+\infty$ - направленное округление в сторону положительной бесконечности (также известное как округление вверх или потолок).

Округление в сторону $-\infty$ - направленное округление в сторону отрицательной бесконечности (также известное как округление вниз или пол).

Пример округления до целых чисел с использованием IEEE 754 правила

Режим	Пример значения			
	+11,5	+12,5	-11,5	-12,5
до ближайшего, связи до четного	+12.0	+12.0	-12.0	-12.0
до ближайшего, увязывает от нуля	+12.0	+13.0	-12.0	-13.0
в сторону 0	+11.0	+12.0	-11.0	-12.0
в сторону $+\infty$	+12.0	+13.0	-11.0	- 12.0
в сторону $-\infty$	+11.0	+12.0	-12.0	-13.0

Если не указано иное, результат операции с плавающей запятой определяется путем применения функции округления к бесконечно точному (математическому) результату. Такая операция называется правильно округленной. Это требование называется правильным округлением.

Необходимые операции

Необходимые операции для поддерживаемого арифметического формата (включая базовые форматы) включают:

Арифметические операции (сложение, вычитание, умножение, деление, квадратный корень, объединенное умножение – сложение, остаток)
Преобразования (между форматами, в и из строк и т. д.)
Масштабирование и (для десятичных) квантование
Копирование и изменение знака (абс., Отрицание и т. д.)
Сравнение и полное упорядочение
Классификация и тестирование для NaN и т. Д.
Тестирование и настройка flags
Разные операции.

Предикаты сравнения

Стандарт предоставляет предикаты сравнения для сравнения одного элемента данных с плавающей запятой с другим в поддерживаемом арифметическом формате. Любое сравнение с NaN считается неупорядоченным. -0 и $+0$ сравниваются как равные.

Предикат общего порядка

Стандарт предоставляет предикат `totalOrder`, который определяет общий порядок канонических элементов поддерживаемого арифметического формата. Предикат согласуется с предикатами сравнения, когда одно число с плавающей запятой меньше другого. Предикат `totalOrder` не устанавливает тотальный порядок для всех кодировок в формате. В частности, он не делает различий между разными кодировками одного и того же представления с плавающей запятой, как когда одна или обе кодировки являются неканоническими. IEEE 754-2019 включает пояснения к `totalOrder`.

Обработка исключений

Стандарт определяет пять исключений, каждое из которых возвращает значение по умолчанию и имеет соответствующий флаг состояния, который поднимается при возникновении исключения. Никакой другой обработки исключений не требуется, но рекомендуются дополнительные альтернативы не по умолчанию (см. § Альтернативная обработка исключений).

Пять возможных исключений:

Недопустимая операция: математически не определено, например, квадратный корень из отрицательного числа. По умолчанию возвращает `qNaN`.

Деление на ноль: операция с конечными операндами дает точный бесконечный результат, например, $1/0$ или $\log(0)$. По умолчанию возвращает \pm бесконечность.

Переполнение: результат слишком велик для правильного представления (т.е. его показатель степени с неограниченным диапазоном экспоненты будет больше, чем `emax`). По умолчанию возвращает \pm бесконечность для режимов округления до ближайшего (и следует правилам округления для режимов направленного округления).
Незаполнение: результат очень мал (вне нормального диапазона) и неточен. По умолчанию возвращает субнормальное или ноль (в соответствии с правилами округления).

Неточный: точный (т. Е. Неокругленный) результат не представляется точно. По умолчанию возвращает правильно округленный результат.

Это те же пять исключений, которые были определены в IEEE 754-1985, но исключение деления на ноль распространено на операции, отличные от деления.

Для десятичных чисел с плавающей запятой существуют дополнительные исключения:

Ограничено: показатель степени результата слишком велик для формата назначения.

По умолчанию к коэффициенту добавляются завершающие нули, чтобы уменьшить показатель степени до наибольшего полезного значения. Если это невозможно (потому что это приведет к тому, что количество цифр должно быть больше, чем формат назначения), тогда возникает исключение переполнения.

Округленное: коэффициент результата требует большего количества цифр, чем предоставляет формат назначения. О неточном исключении сигнализируется, если отбрасываются любые ненулевые цифры.

Кроме того, такие операции, как квантование, когда один из операндов бесконечен или когда результат не соответствует формату назначения, также будут сигнализировать об исключении недопустимой операции.

Рекомендации

Альтернативная обработка исключений

Стандарт рекомендует дополнительную обработку исключений в различных формах, включая предварительную подстановку пользовательских значений по умолчанию и ловушек (исключения, которые каким-то образом изменяют поток управления) и другие модели обработки исключений, которые прерывают поток, например try / catch. Ловушки и другие механизмы исключения остаются необязательными, как в IEEE 754-1985.

Рекомендуемые операции

В разделе 9 стандарта рекомендуются дополнительные математические операции, которые должны быть определены в языковых стандартах. Ничего не требуется для соответствия стандарту.

Рекомендуемые арифметические операции, которые должны правильно округляться:

- $e^x, 2^x, 10^x$
- $e^x - 1, 2^x - 1, 10^x - 1$
- $\ln x, \log_2 x, \log_{10} x$
- $\ln(1 - x), \log_2(1 - x), \log_{10}(1 + x)$
- $\sqrt{x^2 + y^2}$
- \sqrt{x}
- $(1 - x)^n$
- $x^{\frac{1}{n}}$
- x^a, x^b
- $\sin x, \cos x, \tan x$
- $\arcsin x, \arccos x, \arctan x, \operatorname{atan2}(y, x)$
- $\sin \pi x - \sin \pi x, \cos \pi x - \cos \pi x, \tan \pi x - \tan \pi x$ (see also: [Multiples of \$\pi\$](#))
- $\operatorname{asin} \pi x = \frac{\arcsin x}{\pi}, \operatorname{acos} \pi x = \frac{\arccos x}{\pi}, \operatorname{atan} \pi x = \frac{\arctan x}{\pi}, \operatorname{atan2} \pi(y, x) = \frac{\operatorname{atan2}(y, x)}{\pi}$
- $\sinh x, \cosh x, \tanh x$
- $\operatorname{arsinh} x, \operatorname{arcosh} x, \operatorname{artanh} x$

Функции `asinPi`, `acosPi` и `tanPi` не были частью стандарта IEEE 754-2008, потому что казалось, что они менее необходимы. Первые два были хотя бы упомянуты в абзаце, но это считалось ошибкой, пока они не были добавлены в редакцию 2019 года.

Эти операции также включают в себя установку направления округления в динамическом режиме и доступ к нему, а также операции уменьшения вектора, определенные реализацией, такие как сумма, масштабированное произведение и скалярное произведение, точность которых не указана стандартом.

С 2019 года также рекомендуются расширенные арифметические операции для двоичных форматов. Эти операции, указанные для сложения, вычитания и умножения, производят пару значений, состоящую из результата, правильно округленного до ближайшего в формате, и члена ошибки, который можно точно представить в этом формате. На момент публикации стандарта аппаратные реализации не были известны, но очень похожие операции уже были реализованы в программном обеспечении с использованием хорошо известных алгоритмов. История и мотивация их стандартизации объясняются в справочном документе.

По состоянию на 2019 год ранее необходимые `minNum`, `maxNum`, `minNumMag` и `maxNumMag` в IEEE 754-2008 теперь удалены из-за их неассоциативности. Вместо этого рекомендуется два набора новых минимальных и максимальных операций. Первый набор содержит минимум, минимум, максимум и максимум. Второй набор содержит `minimumMagnitude`, `minimumMagnitudeNumber`, `maximumMagnitude` и `maximumMagnitudeNumber`. История и мотивация этого изменения объясняются в справочном документе.

Оценка выражений

Стандарт рекомендует, как языковые стандарты должны определять семантику последовательностей операций, и указывает на тонкости буквального смысла и оптимизации, которые меняют ценность результата. Напротив, предыдущая версия стандарта 1985 оставляла аспекты языкового интерфейса неопределенными, что приводило к несогласованному поведению между компиляторами или различным уровням оптимизации в одном компиляторе.

Языки программирования должны позволять пользователю указывать минимальную точность для промежуточных вычислений выражений для каждого основания. В стандарте это называется «предпочтительной шириной», и должна быть возможность устанавливать ее для каждого блока. Промежуточные вычисления в выражениях должны быть рассчитаны, а любые временные значения должны быть сохранены с использованием максимальной ширины операндов и предпочтительной ширины, если она установлена. Таким образом, например, компилятор, ориентированный на оборудование с плавающей запятой x87, должен иметь средства указания, что промежуточные вычисления должны использовать двойной расширенный формат. Сохраненное значение переменной должно всегда использоваться при вычислении последующих выражений, а не любого предшественника до округления и присвоения переменной.

Воспроизводимость

IEEE 754-1985 допускает множество вариантов реализации (например, кодирование некоторых значений и обнаружение определенных исключений). IEEE 754-2008 усилил многие из них, но некоторые варианты все еще остаются (особенно для двоичных форматов). Пункт о воспроизводимости рекомендует, чтобы языковые стандарты предоставляли средства для написания воспроизводимых программ (т. Е. Программ, которые будут давать одинаковый результат во всех реализациях языка), и описывает, что необходимо сделать для достижения воспроизводимых результатов.

Символьное представление

Стандарт требует операций для преобразования между базовыми форматами и внешними форматами последовательности символов. Преобразование в формат десятичных символов и обратно требуется для всех форматов. Преобразование во внешнюю последовательность символов должно быть таким, чтобы обратное преобразование с округлением до четного восстановило исходное число. Нет требования сохранять полезную нагрузку в виде тихого NaN или сигнального NaN, а преобразование из внешней последовательности символов может превратить сигнальный NaN в тихий NaN.

Исходное двоичное значение будет сохранено путем преобразования в десятичное и обратно с использованием:

5 десятичных цифр для binary16,

9 десятичных цифр для binary32,

17 десятичные цифры для двоичного64,

36 десятичных цифр для двоичного 128.

Для других двоичных форматов необходимое количество десятичных цифр составляет

$$1 + \lceil p \log_{10}(2) \rceil$$

где p - количество значащих битов в двоичном формате, например 237 бит для двоичного 256.

(Примечание: в качестве ограничения реализации правильное округление гарантируется только для количества десятичных цифр выше плюс 3 для наибольшего поддерживаемого двоичного формата. Например, если binary32 является наибольшим поддерживаемым двоичным форматом, то преобразование из десятичной внешней последовательности с 12 десятичными цифрами гарантированно правильно округляется при преобразовании в binary32; но преобразование последовательности из 13 десятичных цифр - нет; однако стандарт рекомендует, чтобы реализации не налагали такого ограничения.)

При использовании десятичного формата с плавающей запятой десятичное представление будет сохранено с использованием:

7 десятичных цифр для decimal32,

16 десятичных цифр для decimal64,

34 десятичных цифр для decimal128 .

Алгоритмы с кодом для правильно округленного преобразования из двоичного в десятичное и из десятичного в двоичное обсуждают Гей, а для тестирования - Паксон и Кахан. Википедия site:wiki5.ru

Воспроизводимость

IEEE 754-1985 допускает множество вариантов реализации (например, кодирование некоторых значений и обнаружение определенных исключений). IEEE 754-2008 усилил многие из них, но некоторые варианты все еще остаются (особенно для двоичных форматов). Пункт о воспроизводимости рекомендует, чтобы языковые стандарты предоставляли средства для написания воспроизводимых программ (т. Е. Программ, которые будут давать одинаковый результат во всех реализациях языка), и описывает, что необходимо сделать для достижения воспроизводимых результатов.

NaN

Два вида NaN: тихий NaN (qNaN) и сигнализационный NaN (sNaN). NaN может нести полезную нагрузку, предназначенную для диагностической информации, указывающей источник, вызвавший NaN. Знак NaN не имеет никакого значения, но может быть предсказуемым в некоторых случаях.

$\text{NaN} = s \ 111 \ 1111 \ 1xxx \ xxxx \ xxxx \ xxxx \ xxxx$

где s - это знак (чаще всего игнорируемый в приложениях), а последовательность x представляет ненулевое число (нулевое значение кодирует бесконечности). Первый бит из x используется для определения типа NaN: «тихий NaN» или «сигнальный NaN». Остальные биты кодируют полезную нагрузку (чаще всего игнорируются в приложениях).

Операции с плавающей запятой, отличные от упорядоченных сравнений, обычно передают тихий NaN (qNaN). Большинство операций с плавающей запятой в сигнальном NaN (sNaN) сигнализируют об исключительной ситуации недопустимой операции; тогда действие исключения по умолчанию такое же, как и для операндов qNaN, и они производят qNaN, если производят результат с плавающей запятой.

Распространение тихих NaN посредством арифметических операций позволяет обнаруживать ошибки в конце последовательности операций без тщательного тестирования на промежуточных этапах. Например, если кто-то начинает с NaN и добавляет 1 пять раз подряд, каждое добавление приводит к NaN, но нет необходимости проверять каждое вычисление, потому что можно просто отметить, что окончательный результат - NaN. Однако, в зависимости от языка и функции, NaN можно незаметно удалить из цепочки вычислений, в которой одно вычисление в цепочке даст постоянный результат для всех других значений с плавающей запятой. Например, вычисление x может привести к результату 1, даже если x равно NaN, поэтому проверка только окончательного результата может скрыть тот факт, что вычисление перед x привело к NaN. В общем, тогда необходим более поздний тест на наличие установленного недопустимого флага, чтобы обнаружить все случаи, когда вводятся NaN.

В разделе 6.2 старого стандарта IEEE 754-2008 есть две аномальные функции (функции `maxNum` и `minNum`, которые возвращают максимум два операнда, которые, как ожидается, будут числами), которые предпочитают числа - если только один из операндов является NaN, то возвращается значение другого операнда. В версии IEEE

754-2019 эти функции были заменены, поскольку они не являются ассоциативными (когда в операнде появляется сигнальное NaN).

Сравнение с NaN

Сравнение с NaN всегда возвращает неупорядоченный результат даже при сравнении с самим собой. Предикаты сравнения либо сигнализируют, либо не сигнализируют о тихих операндах NaN; версии сигнализации сигнализируют об исключении недопустимой операции для таких сравнений. Предикаты равенства и неравенства не сигнализируют, поэтому $x = x$, возвращающий false, можно использовать для проверки, является ли x тихим NaN. Все другие стандартные предикаты сравнения сигнализируют о получении операнда NaN. Стандарт также предоставляет несигнальные версии этих других предикатов. Предикат `isNaN(x)` определяет, является ли значение NaN, и никогда не сигнализирует об исключении, даже если x является сигнальным NaN.

Сравнение NaN и любого значения x с плавающей запятой (включая NaN и $\pm \infty$)						
Сравнение	$\text{NaN} \geq x$	$\text{NaN} \leq x$	$\text{NaN} > x$	$\text{NaN} < x$	$\text{NaN} = x$	$\text{NaN} \neq x$
Результат	Всегда Ложь	Всегда Ложь	Всегда Ложь	Всегда Ложь	Всегда Ложь	Всегда Истина

Операции, генерирующие NaN

Есть три типа операций, которые может возвращать NaN:

- Большинство операций хотя бы с одним операндом NaN.
- Неопределенные формы :
 - Деления $(\pm 0) / (\pm 0)$ и $(\pm \infty) / (\pm \infty)$.
 - Умножения $(\pm 0) \times (\pm \infty)$ и $(\pm \infty) \times (\pm 0)$.
 - Остаток $x \% y$, когда x равен бесконечности или y равен нулю.
 - Сложение $(+\infty) + (-\infty)$, $(-\infty) + (+\infty)$ и эквивалентные вычитания $(+\infty) - (+\infty)$ и $(-\infty) - (-\infty)$.
 - В стандарте есть альтернативные функции для степеней:
 - § Стандартная функция `pow` и функция целочисленной экспоненты `powi` определяют 0, 1 и ∞ как 1.
 - § Функция `powr` определяет все три неопределенных для `ms` как недопустимые операции и поэтому возвращает NaN.
- Действительные операции с сложными результатами, например:
 - Квадратный корень из отрицательного числа.
 - логарифм отрицательного числа.
 - обратный синус или обратный косинус числа, которое меньше -1 или больше 1.

NaN также могут быть явно присвоены переменным, обычно как представление пропущенных значений. До стандарта IEEE программисты часто использовали специальные значения (например, -99999999) для представления неопределенных или отсутствующих значений, но не было гарантии, что они будут обрабатываться согласованно или правильно.

NaN не обязательно генерируются во всех перечисленных случаях. Если операция может вызвать исключительную ситуацию и ловушки не замаскированы, тогда операция вызовет ловушку. Если операнд является тихим NaN, а также нет сигнального операнда NaN, тогда нет условия исключения и результатом является тихий NaN. Явные присвоения не вызовут исключения даже для сигнализации NaN.

Тихие NaN

Тихие NaN, или qNaN, не вызывают никаких дополнительных исключений, поскольку они распространяются через большинство операций. Исключение составляют случаи, когда NaN нельзя просто передать в неизменном виде на вывод, например, при преобразовании формата или некоторых операциях сравнения.

NaN сигнализации

NaN сигнализации или sNaN - это особые формы NaN, которые при использовании большинством операций должны вызывать исключение недопустимой операции, а затем, при необходимости, «подавляться» в qNaN, который затем может распространяться. Они были введены в IEEE 754. Было несколько идей, как их можно использовать:

- Заполнение неинициализированной памяти сигнальными NaN приведет к исключению недопустимой операции, если данные используются до инициализации
- Использование sNaN в качестве заполнителя для более сложный объект , например:
 - Представление числа, у которого отсутствует
 - Представление числа, которое имеет переполнение
 - Число в формат с более высокой точностью
 - А комплексное число

При обнаружении обработчик прерывания может декодировать sNaN и вернуть индекс для вычисленного результата. На практике такой подход сталкивается со многими сложностями. Обработка знакового бита NaN для некоторых простых операций (таких как абсолютное значение) отличается от обработки для арифметических операций. Стандарт не требует ловушек. Есть и другие подходы к такого рода проблемам, которые были бы более переносимыми.

Определение функции

Существуют разногласия по поводу правильного определения результата числовой функции, которая принимает тихий NaN в качестве входных данных. Одна точка зрения состоит в том, что NaN должно распространяться на выход функции во всех случаях, чтобы распространять индикацию ошибки. Другой взгляд, принятый стандартами ISO C99 и IEEE 754-2008 в целом, заключается в том, что если функция имеет несколько аргументов и вывод однозначно определяется всеми не -NaN (включая бесконечность), тогда это значение должно быть результатом. Таким образом, например, значение, возвращаемое $\text{hypot}(\pm \infty, \text{qNaN})$ и $\text{hypot}(\text{qNaN}, \pm \infty)$, равно $+\infty$.

Проблема особенно остро стоит для функции возведения в степень $\text{pow}(x, y) = x^y$. Выражения 0, ∞ и 1 считаются неопределенными формами, когда они встречаются как пределы (точно так же, как $\infty \times 0$), и вопрос о том, является ли от нуля до нулевой степени должен быть определен как 1 разделились мнения.

Если вывод считается неопределенным, когда параметр не определен, то $\text{pow}(1, \text{qNaN})$ должно давать qNaN . Однако математические библиотеки обычно возвращают 1 для $\text{pow}(1, y)$ для любого действительного числа y , и даже когда y равно бесконечности. Точно так же они производят 1 для $\text{pow}(x, 0)$, даже если x равен 0 или бесконечности. Обоснованием для возврата значения 1 для неопределенных форм было то, что значение функций в особых точках может быть принято как конкретное значение, если это значение находится в предельном значении для всех, кроме исчезающе малой части шара вокруг предельного значения. параметров. В версии стандарта IEEE 754 2008 года говорится, что оба $\text{pow}(1, \text{qNaN})$ и $\text{pow}(\text{qNaN}, 0)$ должны возвращать 1, поскольку они возвращают 1 независимо от else используется вместо тихого NaN. Более того, ISO C99, а затем IEEE 754-2008, решили указать $\text{pow}(-1, \pm \infty) = 1$ вместо qNaN ; причина этого выбора указана в обосновании языка C: «Как правило, C99 избегает результата NaN, когда полезно числовое значение.... Результат $\text{pow}(-2, \infty)$ равен $+\infty$, потому что все большие положительные значения с плавающей запятой являются целыми числами ».

Чтобы удовлетворить тех, кто желает более строгой интерпретации того, как должна действовать степенная функция, стандарт 2008 определяет две дополнительные степенные функции: $\text{pow}(\text{pow}(x, n))$, где показатель степени должен быть целое число и $\text{pow}(\text{pow}(x, y))$, которое возвращает NaN всякий раз, когда параметр является NaN, или возведение в степень даст неопределенную форму.

Integer NaN

Наиболее фиксированный -size целочисленные форматы не могут явно указывать недопустимые данные. В таком случае при преобразовании NaN в целочисленный тип стандарт IEEE 754 требует, чтобы сообщалось о недопустимой операции исключение. Например, в Java такие операции вызывают экземпляры `java.lang.ArithmeticException`. В C они приводят к неопределенному поведению, но если приложение F поддерживается, операция приводит к «недопустимому» исключению с плавающей запятой (в соответствии с требованиями стандарта IEEE) и неопределенному ценности.

Пакет Perl Math :: BigInt использует "NaN" для результата строк, которые не представляют действительные целые числа.

```
>perl -mMath :: BigInt -e "print Math :: BigInt->new('foo') "NaN
```

Display

Различные операционные системы и языки программирования могут иметь разные строковые представления NaN.

```
nan NaN NaN% NAN NaNQ NaNs qNaN sNaN 1. # SNAN 1. # QNAN -1. # IND + nan.0
```

Поскольку на практике закодированные NaN имеют знак, тихий / сигнальный бит и необязательная «диагностическая информация» (иногда называемая полезной нагрузкой), они также часто встречаются в строковых представлениях NaN, например:

-NaN NaN12345 -sNaN12300 -NaN (s1234)

(существуют другие варианты).

Кодирование

В форматах хранения с плавающей запятой, соответствующих стандарту IEEE 754, NaN идентифицируются с помощью определенных заранее заданных битовых шаблонов, уникальных для NaN. Знаковый бит не имеет значения. Двоичный формат NaN представлены экспоненциальным полем, заполненным единицами (например, значениями бесконечности), и некоторым ненулевым числом в значении поля (чтобы отличить их от значений бесконечности). Исходный стандарт IEEE 754 1985 года (IEEE 754-1985) описывал только двоичные форматы с плавающей запятой и не определял, как должно быть помечено состояние сигнализации / молчания. На практике старший бит значимого поля определяет, является ли NaN сигнальным или нет. В результате были реализованы две разные реализации с обратным смыслом:

- большинство процессоров (включая те из Intel и AMD семейства x86, Семейство Motorola 68000, семейство AIM PowerPC, семейство ARM, семейство Sun SPARC и, необязательно, новые процессоры MIPS) устанавливают бит сигнализации / молчания в ненулевое значение, если NaN не используется, и в ноль, если NaN сигнализирует. Таким образом, на этих процессорах бит представляет собой флаг is_quiet;
- в NaN, генерируемых PA-RISC и старыми процессорами MIPS, бит сигнализации / молчания равен ноль, если NaN тихо, и ненулевое значение, если NaN сигнализирует. Таким образом, на этих процессорах этот бит представляет собой флаг is_signaling.

Первый вариант предпочтительнее, поскольку он позволяет реализации заглушить сигнальный NaN, просто установив бит сигнализации / молчания в 1. обратное невозможно с последним выбором, потому что установка бита сигнализации / молчания в 0 может дать бесконечность.

Версия стандарта IEEE 754 от 2008 г. (IEEE 754-2008) дает формальные рекомендации по кодированию состояния сигнализации / молчания.

- Для двоичных форматов наиболее значимым битом значимого поля должен быть флаг is_quiet. То есть, этот бит не равен нулю, если NaN неактивен, и равен нулю, если NaN сигнализирует.
- Для десятичных форматов, как в двоичном, так и в десятичном кодировании, NaN идентифицируется по пяти старшим битам поле комбинации после бита знака установлено в единицы. Шестой бит поля - это флаг is_quiet. Стандарт следует интерпретации как флаг is_signaling. То есть бит сигнализации / молчания равен нулю, если NaN является тихим, и ненулевым, если NaN сигнализирует. Сигнализация NaN подавляется очисткой этого шестого бита.

Для соответствия IEEE 754-2008 значение бита сигнализации / молчания в последних процессорах MIPS теперь можно настраивать через поле NAN2008 регистра FCSR. Эта поддержка является необязательной в MIPS версии 3 и требуется в версии 5.

Состояние / значение остальных битов значимого поля не определены стандартом. Это значение называется «полезной нагрузкой» NaN. Если операция имеет единственный вход NaN и передает его на выход, полезная нагрузка результата NaN должна быть полезной нагрузкой входного NaN (это не всегда возможно для двоичных форматов, когда состояние сигнализации / молчания кодируется с помощью is_signalingфлаг, как описано выше). Если имеется несколько входов NaN, полезная нагрузка результата NaN должна быть из одного из входных NaN; в стандарте не указано, какие именно.