МАТЕМАТИЧНЕ МОДЕЛЮВАННЯ В ПРИРОДНИЧИХ НАУКАХ ТА ІНФОРМАЦІЙНІ ТЕХНОЛОГІЇ

И.И. ЖУЛЬКОВСКАЯ, к.т.н., доцент, inivzh@ukr.net
О.А. ЖУЛЬКОВСКИЙ, к.т.н., доцент, olalzh@ukr.net
А.Д. ЖУРАВСКИЙ, студент
Днепровский государственный технический университет, г. Каменское

Современные средства аппаратной и программной поддержки IEEE-стандарта

Проведенное исследование показывает возможности современных процессоров общего назначения и компиляторов языков высокого уровня поддерживать форматы повышенной точности и современные технологии потоковой обработки для повышения эффективности математического моделирования в целом.

The study shows the capabilities of modern general-purpose processors and high-level language compilers to support high-precision formats and modern streaming processing technologies to improve the efficiency of mathematical modeling.

Постановка проблемы

Математическое моделирование является альтернативой дорогостоящим физическим экспериментам. Именно поэтому численный эксперимент стал наиболее распространенным и актуальным методом исследования всевозможных объектов и процессов, происходящих во всех сферах человеческой деятельности.

Решение современных задач, формализованных в виде математических моделей, требует крайне сложных вычислений над огромными массивами данных, циклы обработки которых содержат значительное число итераций (10^6 и более) с огромным количеством шагов. Ошибка округления, за редким исключением, проявляется на каждом таком расчетном шаге. При этом точность машинных вычислений становится неудовлетворительной, а вычислительная погрешность определяет основную долю ошибки в получаемом решении.

Как известно, большинство современных вычислительных систем, в том числе и суперкомпьютеров, строится на базе процессоров общего назначения. Вычислительные ошибки возникают в связи с точностью представления действительных (вещественных) чисел в памяти ЭВМ при вводе данных и/или при проведении арифметических операций. Одним из источников вычислительных погрешностей является приближенное представление действительных чисел в ЭВМ, обусловленное конечностью разрядной сетки.

Таким образом, при проведении математического моделирования на ЭВМ нужно иметь представление о погрешностях машинной арифметики.

Анализ последних исследований и публикаций

Вычисления с действительными числами были одним из приоритетных применений компьютеров с момента их появления. Область научных вычислений, а также численное решение большинства практических задач связаны, в основном, с выполнением операций над действительными числами.

Действительные данные в памяти вычислительной системы представлены в формате с плавающей запятой (точкой). В 1985 г. рабочей группой *Floating-Point Working Group* комитета стандартов *IEEE* (*Institute of Electrical and Electronics Engineers*) был разработан и внедрен единый стандарт для представления в двоичном коде чисел с плавающей запятой *IEEE Std* 754–1985, нашедший воплощение в текущей версии *IEEE Std* 754–2008 [1]. Позднее этот стандарт был утвержден Международной организацией по стандартизации *ISO* (*International Organization for Standardization* — крупнейший в мире разработчик и издатель международных стандартов) как *ISO/IEC/IEEE* 60559:2011.

Авторские стандарты *IEEE* с определенных пор стали коммерческим продуктом и не находятся в свободном распространении и обращении (официальный текст стандарта распространяется на платной основе). Ограничение доступности указанных стандартов не могло не сказаться на компетентности постановки вычислительных задач и достоверности результатов численного экспериментирования.

Авторы многих публикаций [2, 3 и др.] предлагают различные подходы к стандартизации математических функций, работающих с числами с плавающей запятой в форматах *IEEE* 754. Наряду с этим, множество работ, [3—5] посвящено тестированию на соответствие стандарту, вопросам погрешностей результатов при вычислении функций [6]. Авторами настоящей работы ранее исследован и описан алгоритм современного подхода к формированию машинного представления и хранения числовой информации в формате с плавающей запятой, рассмотрены особенности представления, а также вычислены граничные (максимальные и минимальные) значения субнормальных и нормализованных чисел [7, 8 и др.].

Формулирование цели исследования

Повышение достоверности результатов моделирования на основе оценки современных, наиболее часто используемых, аппаратных и программных средств на предмет получения наибольшей точности вычислений и снижения погрешностей, обусловленных представлением действительных данных в памяти компьютера.

Изложение основного материала

Как известно, значение двоичного числа с плавающей запятой определяется выражением:

$$A = \pm a_0.a_1a_2a_3...a_{n-1} \times S^e, \qquad (1)$$

где S — основа системы счисления; n — число значащих разрядов мантиссы; a_i — цифры ($0 \le a_i < S$); e — порядок или экспонента (не путать с числом e).

Стандарт *IEEE* 754 определяет основные параметры форматов представления чисел с плавающей запятой, которые отличаются диапазоном представимых в них значений. Параметры основных двоичных форматов (*basic format*), определенных стандартом, приведены в табл.1. К ним относятся числа одинарной (*single precision*), двойной (*double precision*) и четырехкратной точности (*quadruple precision*).

Таблица 1. Параметры основных двоичных форматов чисел с плавающей запятой

Формат	**Всего бит**	**Бит в порядке**	**Бит в мантиссе (*m*)**	**Смещение порядка (*bias*)**
*binary*32 *single*	32	8	23	127
*binary*64 *double*	64	11	52	1023
*binary*128 *quadruple*	128	15	112	16383

В стандарте *IEEE* 754 порядок представлен в виде беззнакового числа, называемого смещенным порядком (E), которое отличается от порядка (e) на фиксированную для данного формата величину, называемую смещением (*bias*).

Стандарт рекомендует (но не требует) реализации предоставлять расширенный формат(ы) (*extended precision format*), являющийся надмножеством некоторого базового (более конкретно — расширенный тип должен иметь диапазон экспонент следующего по размеру базового типа, и диапазон мантисс между данным и следующим базовыми типами). Конкретное битовое представление такого формата определяется реализацией. Этот формат может быть использован для повышения точности промежуточных вычислений.

Примером *extended precision format* является 80-битный формат с явным ведущим битом мантиссы, расширяющий тип *binary*64 (называемый поэтому *double-extended precision*).

Некоторые архитектуры имеют поддержку такого формата, но детали реализации отличаются от производителя к производителю. Так, например, микропроцессоры *Intel* поддерживают формат расширенной точности с 1 битом знака, 15-битной экспонентой и 64-битной мантиссой, и хранят их в регистрах модуля *FPU* (*Floating Point Unit* — модуль операций с плавающей запятой) шириной 80 бит.

При кодировании мантиссы в десятибайтовом поле, в отличие от кодирования в двух других полях, ее целая часть не отбрасывается, а указывается в отведенных под код мантиссы битах вместе со всеми остальными цифрами дробной части. Это объясняется тем, что десятибайтовая разрядная сетка используется не только для хранения действительных чисел, но и для выполнения над ними различных операций. Если первоначально число в формате с плавающей запятой хранится в четырех- или восьмибайтовом поле, то перед выполнением какого-либо действия в коде мантиссы этого числа восстанавливается единица целой части и производится расширение кода до десятибайтовой сетки. Таким приемом обеспечиваются правильность выполнения всех арифметических операций над числами в формате с плавающей запятой и наивысшая возможная точность вычислений.

Внутри *FPU* числа хранятся в 80-битном формате с плавающей запятой, а вычисления ведутся *FPU* исключительно в этом формате. Запись в память и чтение из неё других форматов приводит к автоматическому преобразованию. При этом субнормальные числа *binary*32/*binary*64 нормализуются.

Естественно, не каждое действительное число можно представить в формате с плавающей запятой. К представимым относятся числа, не имеющие дробной части (т.е. множество целых чисел), а также числа, которые можно представить в виде конечной двоичной дроби. Точнее, представимые в формате с плавающей запятой числа должны быть двоично-рациональными, т.е. должны иметь вид $p/2^m$, где p и m (m>0) — целые числа. Только в этом случае число представляется в формате с плавающей запятой без округления и, соответственно, без потери точности. Остальные же числа представлены приближенно, т.е. они округляются до точно представимых в формате с плавающей запятой.

Как известно, стандарт *IEEE* 754 описывает операции сложения, умножения, вычитания, деления, вычисления остатка от деления, извлечения квадратного корня и преобразований между различными типами чисел. Общий принцип всех операций заключается в том, что результат получается из точного путем приведения к представимому числу согласно установленному режиму округления. При этом исходное число модифицируют так, дабы результат поместился в целевой формат.

Для представления результатов вычислений в виде чисел с плавающей запятой стандарт *IEEE* 754 определяет следующие четыре режима округления.

1. Округление к ближайшему (*round to nearest*). В этом режиме должно возвращаться ближайшее к результату представимое значение.

2. Округление к отрицательной бесконечности. В этом случае результатом будет ближайшее представимое число, не превосходящее точного значения, т.е. ближайшее меньшее представимое число.

3. Округление к положительной бесконечности. Режим такого округления предусматривает возвращение ближайшего представимого числа, которое не меньше точного значения, т.е. ближайшее большее представимое число.

4. Округление к нулю. Такой режим округления возвращает ближайшее представимое число, не превосходящее по абсолютной величине точного значения, что означает выбор после усечения числа, ближайшего к нулю.

Любая реализация стандарта *IEEE* 754 обязана

поддерживать режим *round to nearest* в качестве дефолтного (*default settings*).

Основным показателем качества машинной арифметики с плавающей запятой считается точность (*accuracy*), с которой арифметика округляет действительные числа. Чем больше расстояние от исходного числа до ближайшего представимого, тем с меньшей точностью оно может быть представлено.

Расстояние между соседними представимыми числами, т.е. числами с единым десятичным значением порядка и с различающимися в один бит мантиссами, называют шагом числа. Следовательно, максимальная абсолютная погрешность округления для числа в формате *IEEE* 754 равна половине шага. Таким образом, вычисленный результат отличается от точного значения не более чем на половину единицы последнего разряда мантиссы результата (*unit in the last place*, *ulp*) [9]. В тексте стандарта эта величина носит название *quantum*.

Таким образом, максимальная абсолютная погрешность округления определяется по формуле

$$\Delta^{\max} = 0{,}5ulp\,. \qquad (2)$$

Шаг чисел удваивается с увеличением экспоненты двоичного числа на единицу. Т.е. чем дальше от нуля, тем шире шаг чисел в формате *IEEE* 754 по числовой оси.

Для повышения точности вычислений при работе с «маленькими» числами в стандарте предусмотрена возможность использования так называемых субнормальных (*subnormal numbers*), т.е. ненормализованных чисел.

Описание машинного представления и особенностей использования субнормальных чисел в стандарте *IEEE* 754, а также получение формул для вычисления и сам расчет граничных значений (максимальных и минимальных) субнормальных чисел в десятичной системе счисления для различных форматов рассмотрены ранее [7].

Также авторами настоящей работы рассмотрены [10] особенности стандартных способов округления чисел с плавающей запятой при вычислениях на компьютере и получены выражения для вычисления максимальных абсолютных погрешностей округления чисел, представленных в базовых форматах стандарта *IEEE* 754 (табл.2).

Таблица 2. Выражения для вычисления максимальных абсолютных погрешностей округления чисел базовых форматов стандарта *IEEE*

Формат	**Субнормальные числа**	**Нормализованные числа**
*binary*32	2^{-150}	2^{E-151}
*binary*64	2^{-1075}	2^{E-1076}
*binary*128	2^{-16495}	$2^{E-16496}$

Исторически поддержка арифметики с плавающей запятой в x86-командах, впервые появившаяся в процессорах компании *Intel*, была реализована в отдельном сопроцессоре *Intel* 8087 и в следующих его модификациях, вплоть до *Intel* 486*DX*, начиная с которого *FPU*-модуль был интегрирован в центральный процессор. При отсутствии сопроцессора генерировалось прерывание, обработчик которого мог вызвать «медленную» программную эмуляцию соответствующей операции. Как и другие расширения базового набора инструкций процессора, инструкции x87 не являются строго необходимыми для построения рабочей программы, но, будучи аппаратно реализованными, позволяют компиляторам генерировать более эффективный программный код и благодаря этому оптимизировать и ускорять выполнение общих математических задач.

Инструкции x87 совместимы со стандартом *IEEE* 754, но выполняют операции не в строгом соответствии с форматами *IEEE* 754 из-за использования более широких регистров. Так, ппоследовательность арифметических операций может выполняться несколько по-разному на наборе x87 и на x86, строго следующему формату *IEEE* 754. Сопроцессор организует свои регистры не как массив, характерный большинству других архитектур, а как регистровый стек, работающий по принципу обратной польской записи. Такая организация вычислений есть удобной для программистов, но трудоемка для построения компилятором эффективного кода на базе x87-инструкций.

В процессоре *Pentium* III с ядром *Katmai* появилось расширение инструкций для потоковой обработки *SSE* (*Streaming SIMD Extensions*). Благодаря режиму *SIMD* (*Single Instruction Multiple Data* — одна инструкция, множество данных) началось использование принципа параллелизма на уровне данных (когда требуется применять однотипные операции к потоку данных) в компьютерных вычислениях, в т.ч. и с плавающей запятой.

В общем случае к архитектуре процессора добавляется ряд инструкций и несколько 128-битных регистров с различной интерпретацией. Тем не менее, хотя изначально каждый регистр трактуется как два значения с плавающей запятой двойной точности (2*64 бит), операции могут применяться практически ко всем типам, «помещающимся» в 16 байт.

Это означает, например, что появляется возможность одновременно сложить или умножить с помощью всего одной инструкции два операнда из четырех чисел с плавающей запятой одинарной точности, двух — с двойной, двух 64-битных целочисленных, шестнадцати 8-битных целых и т.п.

Таким образом, для получения максимальной отдачи от *SSE* следует использовать такие структуры данных, которые максимально укладываются в развитые 128-битные регистры. В противном случае используются специальные *SSE*-инструкции. Для программ с большим количеством ветвлений и условных операций рекомендуется, по возможности, заменять условные ветвления на логические и вычислительные операции.

Преимущество в производительности достигается в том случае, когда необходимо произвести одну и ту же последовательность действий над разными данными. В таком случае блоком *SSE* осуществляется распараллеливание вычислительного процесса между данными. Версии *SSE* и процессоры, в которых они реализованы впервые, показаны в табл. 3.

Таблица 3. Версии *SIMD*-расширения *Intel*

Версия	**Процессор**
SSE	*Intel Pentium* III
*SSE*2	*Intel Pentium* IV
*SSE*3	*Intel Pentium* IV *(Prescott)*
*SSE*4	*Intel Core (Penryn)*

Компиляторы языка *C* от *Intel*, начиная с версии 10, а также *Sun Studio* от *Sun Microsystems* с версии 12 *update* 1 генерирует инструкции *SSE*4 с помощью специальных опций. Компилятор *GCC* (*GNU Compiler Collection* — набор компиляторов для различных языков программирования, разработанный в рамках проекта *GNU*, свободное программное обеспечение, используемое как стандартный компилятор для свободных *UNIX*-подобных операционных систем) поддерживает *SSE*4.1 и *SSE*4.2 с версии 4.3.

Большинство устройств, имеющих аппаратную поддержку вычислений с плавающей запятой, на данный момент поддерживают именно *IEEE* 754. Большая часть остальных случаев сводится к поддержке формата, похожего на *IEEE* 754, но с некоторыми отличиями в семантике, например, отсутствие *NaN* и/или бесконечностей, принудительное *denormals-are-zero*, неизменяемое направление округления к нулю. Большинство устройств поддерживают *binary*32 и *binary*64 (*float*/*single* и *double* соответственно). Аппаратная поддержка *binary*128 чрезвычайно редка, например *z/Architecture* и *POWER*9; в *SPARC V*8 и *V*9 заявлена поддержка, но реальная аппаратная поддержка отсутствует.

Стандарты *C*/*C++* не требуют, чтобы арифметика с плавающей запятой реализовывала именно *IEEE* 754. *Appendix F* стандарта *C* подробно излагает соответствие между операциями *IEEE* 754 и конструкциями/функциями языка *C* (в *C++* соответствующий раздел отсутствует). Однако может случиться так, что конкретная реализация не отвечает требованиям; например, по причине того, что это не позволяет установленное оборудование.

Языки *C*/*C++* согласно стандарту предоставляют три типа с плавающей запятой: *float, double и long double.* Типы *float* и *double* практически всюду соответствуют *binary*32 и *binary*64 стандарта *IEEE* 754.

Тип *long double* менее однозначен. Для x86/x86_64 в *GCC* и *Clang* он по умолчанию соответствует 80-битному расширенному формату *Intel* x87. Для *Visual Studio* и *ICC* (*Intel C++ compiler*) он физически соответствует тому же типу, что и *double*, хотя логически является самостоятельным типом. В частности по *double* и *long double* возможна перегрузка функций. Для *ICC* для *Windows* можно воспользоваться флагом /Qlong-double, который меняет *long double* на 80-битный x87 (тип занимает 16 байт из соображений выравнивания). Для *Visual Studio* поддержка 80-битного расширенного типа x87 отсутствует, начиная с 32-битных версий (присутствовала в 16-битных).

Размер типа *long double* в смысле значения sizeof(long double) для 80-битного типа x87 типично не равен 10 байтам из соображений выравнивания — реальные значения обычно равны 12 или 16 байт. Для *GCC* значение можно изменить соответствующими опциями компиляции. Неиспользуемые байты имеют произвольные значения.

На некоторых платформах *long double* означает *binary*128, либо использует пару 64-битных действительных чисел для достижения 106-битной точности, но с диапазоном обычного *double*.

На многих платформах (в частности x86/x86_64) *GCC* по умолчанию поддерживает (предоставляет в качестве расширения) нестандартный тип __float128 (а также __float80). Реализация этого типа (на архитектурах, не поддерживающих *binary*128) — программная и, соответственно, медленная.

В некоторых версиях *ICC* есть поддержка 128-битного типа с плавающей запятой _Quad.

Выводы

Для современных цифровых вычислительных систем очень важной проблемой является ограниченная размерная сетка, используемая для представления данных. Это, в свою очередь, приводит к дополнительной погрешности результатов вычисления, которая может оказаться соизмеримой с величиной исходных данных.

Проведенное исследование показывает возможности современных процессоров общего назначения и компиляторов языков высокого уровня поддерживать форматы повышенной точности и современные технологии потоковой обработки для повышения эффективности математического моделирования в целом.

ЛИТЕРАТУРА

1. IEEE Standard for Floating-Point Arithmetic. – New York, 2008.– 70p.
2. Some Notes for a Proposal for Elementary Function Implementation in Floating-Point Arithmetic / G. Hanrot, V. Lefevre, J.-M.Muller, N. Revol and other // Proc. of Workshop IEEE 754 and Arithmetic Standardization, in ARITH-15.– 2001.
3. Proposal for a standardization of mathematical function implementation in floating-point arithmetic / D. Defour, G. Hanrot, V. Lefevre, J.-M.Muller and other // Numerical Algorithms.– 2004.– №37(1–4).– P.367–375.
4. A Test Generation Framework for Datapath Floating-Point Verification / M. Aharoni, S. Asaf, L. Fournier, A. Koifman and other // IEEE International High Level Design Validation and Test Workshop.– 2003.– P.17–22.
5. Stehle D. Searching Worst Cases of a One-Variable Function Using Lattice Reduction / D.Stehle, V.Lefevre, P. Zimmermann // IEEE Transactions on Computers.– 2005.– №54(3).– P.340–346.
6. Никонов О.Я. Оценка точности вычислений специальных функций при разработке компьютерных программ математического моделирования / О.Я. Никонов, О.В. Мнушка, В.М. Савченко // Вісник НТУ «ХПІ». Тематичний випуск: Інформатика і моделювання.– Харків: НТУ.– 2011.– №17.– С.115–121.
7. Жульковская И.И. Вычисление граничных значений субнормальных чисел в IEEE-стандарте / И.И. Жульковская, О.А. Жульковский, Р.Г. Шаганенко // Математичне моделювання. – Дніпродзержинськ: ДДТУ. – 2015.– №1 (32).– С.41–44.
8. Жульковская И.И. Вычисление граничных значений действительных числовых данных в IEEE-стандарте / И.И. Жульковская, О.А. Жульковский, Ю.В. Николаенко // Зб. наук. праць ДДТУ (технічні науки).– Дніпродзержинськ, ДДТУ.– 2015.–№.1 (26).– С.240–245.
9. Goldberg D. What Every Computer Scientist Should Know about Floating-Point Arithmetic / D. Goldberg // ACM Computing Surveys.– 1991.– №.23(1).– P.5–48.
10. Жульковская И.И., Жульковский О.А. Вычисление максимальных абсолютных погрешностей округления чисел в IEEE-стандарте / И.И. Жульковская, О.А. Жульковский // Математичне моделювання. – 2015.– №2 (33).– С.33–36.

пост. 15.09.2017