ФОРМАТ ПРЕДСТАВЛЕНИЯ МНОГОРАЗРЯДНЫХ ЧИСЕЛ С ПЛАВАЮЩЕЙ ТОЧКОЙ, ОРИЕНТИРОВАННЫЙ НА ПАРАЛЛЕЛЬНУЮ ОБРАБОТКУ

А.Н. Мальцев[1], К.С. Исупов

Вятский Государственный Университет

Введение.

Для большинства научных вычислений, особенно тех, что используют эмпирические данные, 32-битная арифметика спецификации IEEE-754 имеет достаточную точность. Она предпочтительна, так как экономит память, время вычислений и затраты энергии. В других приложениях для достижения нужной точности может потребоваться 64-битная арифметика. Однако некоторые ученые и инженеры, производящие объемные вычисления, с сожалением отмечают, что с быстрым ростом масштабов вычислений возникают численные трудности, приводящие к получению результатов сомнительной точности даже при использовании 64-битной арифметики. Часто бывает, что при условном переходе происходит выбор неправильной ветви, то есть, при определенных обстоятельствах, результат получается абсолютно неверным.

Вычисления, требующие повышенной точности, обычно отличаются наличием одной или более из следующих составляющих: (а) плохо обусловленные линейные системы; (б) крупные суммирования; (в) длительное моделирование; (г) масштабные высокопараллельные расчеты, или (д) экспериментальные математические расчеты. В настоящее время для выполнения таких вычислений наиболее часто применяется арифметика с точностью примерно в два раза больше стандартной 64-битной арифметики с плавающей точкой. Одним из вариантов является 128-битный IEEE формат, в котором поле мантиссы расширено до 112 разрядов. Однако аппаратная поддержка этого формата требует значительных затрат. Гораздо более распространенный вариант, реализованный в программном обеспечении, известен как формат «double-double» (точность около 31-й десятичной цифры). Этот формат состоит из пары 64-битных чисел (s, t), где s является числом с плавающей точкой, ближайшим к истинному значению, а t – разница (положительная или отрицательная) между истинным значением и s. Кроме этого существует несколько свободно доступных программных пакетов, поддерживающих арифметику многократной или произвольной точности, в частности ARPREC (поддерживает вещественные, целые и комплексные числа произвольной точности, а также многие алгебраические и трансцендентные функции), GMP (поддерживает вычисления с целыми, рациональными числами и числами с плавающей точкой), MPFR (поддерживает вычисления с плавающей точкой с различной точностью и точным округлением), QD (включает в себя процедуры «double-double» и «quad-double» арифметики, а также многие алгебраические и трансцендентные функции), NTL (включает структуры данных и алгоритмы обработки целых чисел любой длины, векторов, матриц и полиномов над целыми числами и над конечными полями, а также арифметику с плавающей точкой произвольной точности).

Однако очевидно, что позиционная арифметика высокой точности имеет недостатки. Вычисления в формате «double-double» обычно в 5 раз медленнее, чем в 64-битном формате, в формате «quad-double» в 25 раз медленнее. При использовании произвольной точности время вычислений может возрастать в сотни и тысячи раз [1]. Таким образом, возникает потребность в новых программных средствах, которые бы позволили ускорить вычисления с многократной или произвольной точностью. Одним из способов достижения этой цели может быть использование форматов представления длинных чисел, ориентированных на параллельную обработку. Далее рассматривается формат представления чисел, основанный на системе остаточных классов.

1. Формат представления длинных чисел с плавающей точкой.

Для представления чисел большой разрядности при организации высокоточных вычислений предлагается следующий модулярно-позиционный формат с плавающей точкой (далее МП-формат):

$$x \to \{s, M, e, I(M/P)\}, \qquad (1)$$

где s – знак числа x, $M = (m_1,m_2,...,m_n)$ – мантисса, представленная в системе остаточных классов (СОК) [2] набором независимых остатков по взаимно-простым модулям $p_1,p_2,...,p_n$; e – порядок (экспонента числа), изменяющийся в пределах интервала, ограниченного двумя целыми числами $e_{\min}$ и $e_{\max}$; $I(M/P) = [b(M), t(M)]$ – интервально-позиционная характеристика (ИПХ) модулярной мантиссы, являющаяся интервальной аппроксимацией ее относительной величины [3], причем $b(M)$ и $t(M)$ – соответственно верхняя и нижняя границы ИПХ, представляющие собой машинные числа с плавающей точкой. Значение модулярной мантиссы – целое число, изменяющееся в пределах диапазона $[0, P–1]$, где $P = p_1p_2...p_n$. Таким образом, выбор различного числа модулей позволяет задавать произвольно высокую точность вычислений. Значение чисел вида (1) определяется выражением

$$x = (-1)^s \cdot |m_1B_1+m_2B_2+...+m_nB_n|_P \cdot 2^e,$$

где $B_1,B_2,...,B_n$ – ортогональные базисы СОК. Схема МП-формата представлена на рисунке 1.

Рис. 1. МП-формат: INTEGER – целочисленный тип данных (long, int, long int), SHORT – целочисленный короткий тип, REAL – любой вещественный тип (float, double или long double)

В МП-формате мантисса, многоразрядная в позиционной системе, представляется в виде нескольких малоразрядных остатков. Все модульные операции над остатками по каждому модулю выполняются отдельно и независимо, без необходимости распространения межразрядных переносов, следовательно, просто и быстро. При этом появляется возможность применения методов параллельной обработки разрядов мантиссы, путем векторизации, либо распределения вычислений по ядрам многоядерных процессоров и графических ускорителей, что делает перспективным использование МП-формата для организации высокопроизводительных параллельных вычислений высокой точности при решении больших задач.

Стоит отметить, что в разные периоды времени было предложено несколько способов представления рациональных / вещественных чисел большой разрядности на основе модулярных систем [4-9]. Существенным отличием МП-формата от них является включение ИПХ модулярной мантиссы (двух чисел с плавающей точкой $b(M)$ и $t(M)$) в представление числа. Это позволяет ускорить вычисление результата наиболее трудоемких немодульных операций над мантиссами, таких как сравнение, определение знака, оценка переполнения допустимого диапазона представления, округление и пр. [3, 10]. Поясним сказанное. Одним из наиболее распространенных методов выполнения немодульных операций в СОК является преобразование модулярного кода в код системы со смешанными основаниями (Mixed Radix Conversion, MRC-метод). Для выполнения такого преобразования требуется выполнить порядка n^2 арифметических операций над остатками, где n – количество модулей [11]. Соответственно сложность основных немодульных операций определяется функцией $O(n^2)$. В свою очередь, вычисление ИПХ в соответствии со специальным алгоритмом [10] требует $O(n)$ арифметических операций с плавающей точкой. При этом выполнение немодульных операций сводится в общем случае к сопоставлению противоположных границ ИПХ операндов и, следовательно, выполняется также за время $O(n)$. При использовании МП-формата необходимость алгоритмического вычисления ИПХ в большинстве случаев отсутствует (поскольку появляется возможность использования интервальной арифметики), что обеспечивает выполнение базовых немодульных операций над модулярными мантиссами за время $O(1)$. Кроме этого, включение ИПХ в формат числа обеспечивает новую организацию арифметической обработки чисел, в основе которой заложена более эффективная схема предвычислительного округления.

Для МП-формата помимо числовых кодировок определены способы представления бесконечностей и нечисловых величин (таблица 1). Нечисловые величины, в отличие от бесконечностей, не могут являться исходными данными арифметических операций, но могут быть получены в ходе их выполнения. Несмотря на одинаковую мантиссу, неоднозначности при интерпретации нуля и бесконечности не возникает в силу различных значений порядка.

Таблица 1 – Кодировки значений модулярно-позиционных величин с плавающей точкой

Класс	Порядок, e	Модулярная мантисса со знаком, $\pm M$
Ноль со знаком	0	$\pm(0,0,...,0)$
Числовые величины	$e_{\min} \le e \le e_{\max}$	$\pm(m_1,m_2,...,m_n)$, есть $m_i \neq 0$
Бесконечность со знаком	$e_{\max}+1$	$\pm(0,0,...,0)$
Нечисловые величины (NaN)	$e_{\max}+1$	$\pm(m_1,m_2,...,m_n)$, есть $m_i \neq 0$

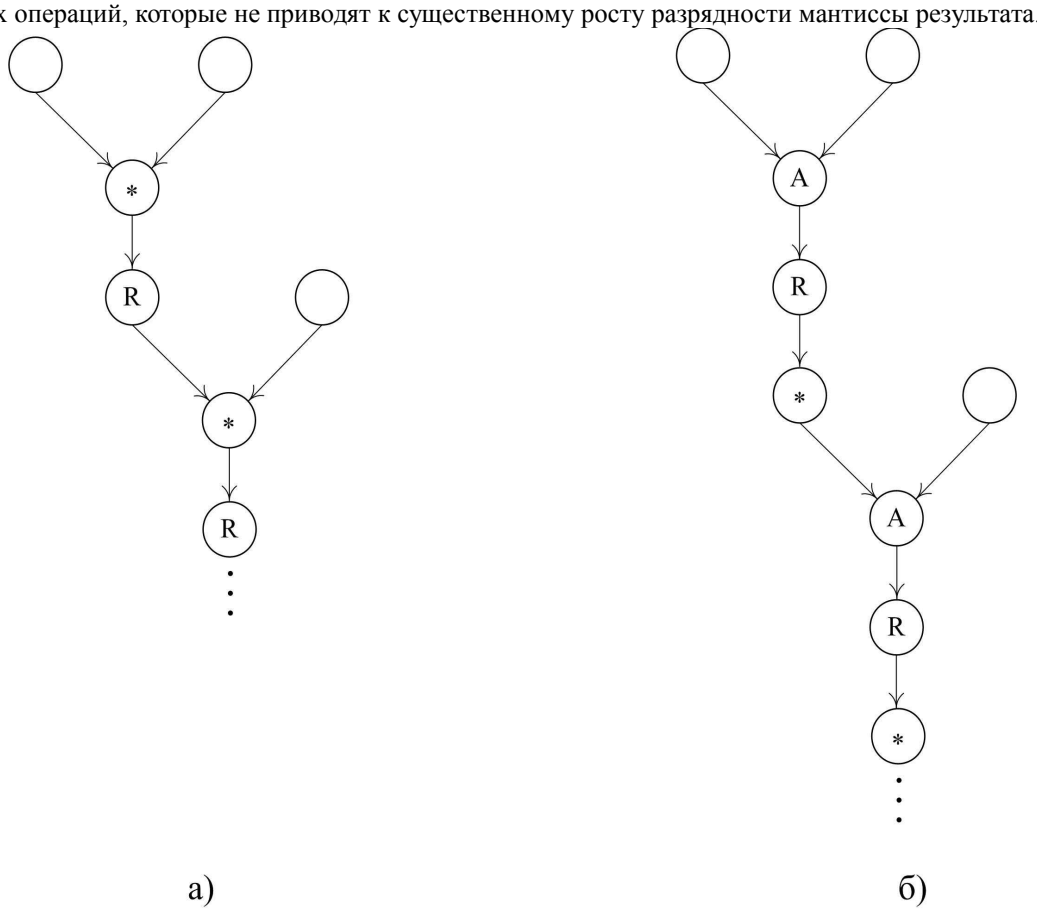
2. Алгоритмы многоразрядной арифметики.

Базовые алгоритмы выполнения арифметических операций и округления чисел, представленных в МП-формате, поддерживают арифметику бесконечностей, а также обработку основных исключительных ситуаций, определенных спецификацией IEEE-754 (переполнение, деление на ноль и пр.). Рассмотрим их.

2.1. Округление.

Округление – ключевая операция при реализации любых нецелочисленных вычислений, в особенности итерационных, поэтому основным требованием к алгоритму его выполнения является высокая скорость. Различают две противоположных схемы округления: поствычислительная и предвычислительная (рисунок 2). Согласно поствычислительной схеме округление результата производится после выполнения арифметической операции так, чтобы его мантисса не превышала заданного предела. Для МП-формата этот предел равен sqrt(P–1), где sqrt – квадратный корень, иначе при умножении не гарантируется отсутствие переполнения. Такой подход используется в позиционной арифметике с плавающей точкой. Напротив, предвычислительная схема

предполагает, что округляются один или оба исходных операнда, причем таким образом, чтобы результатная мантисса находилась в пределах допустимого диапазона представления. С точки зрения точности вычислений и снижения количества округлений предпочтительна предвычислительная схема, так как она позволяет избежать ненужных округлений в том случае, если результат последующей операции будет находится в допустимом диапазоне $[0, P–1]$. Наибольший эффект предвычислительного округления достигается при выполнении аддитивных операций, которые не приводят к существенному росту разрядности мантиссы результата.



а) б)

Рис. 2. Поствычислительная (а) и предвычислительная (б) схемы округления: * – арифметическая операция, A – проверка необходимости округления, R – округление

Проблемы реализации быстрого предвычислительного округления в многоразрядных позиционных форматах связаны с необходимостью дополнительного анализа операндов с целью определения количества позиций для сдвига мантисс. Однако эти проблемы естественным образом решаются при работе с числами в МП-формате, поскольку наличие малоразрядной интервально-позиционной характеристики в представлении числа позволяет дать быструю оценку величины его мантиссы. Таким образом, общая задача предвычислительного округления чисел разбивается на две подзадачи:

- проверка необходимости округления, цель которой заключается в определении значения функции rsh(M) – показателя наименьшей степени двойки, на которую необходимо поделить двоичное представление модулярной мантиссы для того, чтобы при выполнении следующего арифметического действия над ней не произошло переполнения;
- собственно округление, состоящее в уменьшении значения модулярной мантиссы в $2^{\mathrm{rsh}(M)}$ раз с сопутствующим увеличением порядка.

Способ определения значения функции rsh(M) зависит от арифметической операции, которая будет выполнена. Рассмотрим для примера определение необходимости округления при умножении. Пусть дано число x формата (1). Обозначим $P' = \mathrm{floor}(\mathrm{sqrt}(P–1))$, где floor – округление к меньшему целому, P – произведение модулей СОК. Разобьем полный модулярный диапазон $D = [0, P–1]$ на два интервала: $D_{work} = [0, P']$ и $D_{crit} = [P'+1, P–1]$. Интервал D_{work} назовем рабочим диапазоном, и если $M \in D_{work}$, то округление числа x не требуется. Интервал D_{crit} – это критический диапазон, поскольку при $M \in D_{crit}$ необходимо выполнить округление, иначе при выполнении умножения на второе такое же число мантисса результата может выйти за пределы D. Возникает необходимость нахождения значения целочисленной функции rsh(M), определяющей показатель коэффициента, которым необходимо масштабировать M для того, чтобы вернуть ее в D_{work}. Значение rsh(M) определяется равенством $2^{\mathrm{rsh}(M)} = M/P'$. Логарифмируя обе его части с учетом целочисленности показателя степени имеем: $\mathrm{rsh}(M) = \mathrm{ceil}(\log_2(M/P'))$, где ceil – округление к большему целому. Далее, если rsh(M) больше

нуля, необходимо выполнить операцию округления round(x), которая осуществляется в соответствии со следующим алгоритмом.

1. Если rsh(M) $\leq$ 0, то принять round(x) = x и перейти к шагу 6, иначе – к следующему шагу.

2. Масштабировать модулярную мантиссу M коэффициентом $2^{\mathrm{rsh}(M)}$: $M_r = \mathrm{floor}(M/2^{\mathrm{rsh}(M)})$.

3. Увеличить порядок: $e_r = e + \mathrm{rsh}(M)$.

4. Если $e_r > e_{\max}$, то это воспринимается, как выполнение условия арифметического переполнения. В этом случае следует сформировать и занести в регистр состояния соответствующий код исключения, установить $e_r = e_{\max}+1$, $M_r = 0$, $I(M_r/P) = [0, 0]$ (кодировка бесконечности) и перейти к шагу 6.

5. Определить ИПХ округленной мантиссы M_r в соответствии с высокоточным алгоритмом [10].

6. Завершить выполнение алгоритма, вернув округленное число round(x) $\rightarrow \{s, M_r, e_r, I(M_r/P)\}$.

Наиболее трудоемким является шаг 2. Для его реализации разработан новый итерационный метод модулярного масштабирования чисел [12], основная суть которого состоит в следующем. Пусть дана мантисса в модулярном представлении $M = (m_1,m_2,...,m_n)$ и требуется определить частное $M_r = (r_1,r_2,...,r_n)$ такое, что $M_r = \mathrm{floor}(M/2^{\mathrm{rsh}(M)})$. Зафиксируем 2^h – основной (максимальный) шаг масштабирования. Предварительно вычислим модулярные коды мультипликативных инверсий первых h натуральных степеней двойки (при нечетных модулях все степени двойки обратимы в кольце вычетов $\boldsymbol{Z}/P\boldsymbol{Z}$), которые для удобства представим в виде hn-элементной матрицы

$$\mathbf{H} = (|(2^j)^{-1}|_{p1}, |(2^j)^{-1}|_{p2},..., |(2^j)^{-1}|_{pn}), \quad j = 1,2,...,h.$$

Дале вычислим матрицу модулярных поправок:

$$\mathbf{S} = (|\mathrm{ceil}(jP/2^h)|_{p1}, |ceil(jP/2^h)|_{p2},..., |ceil(jP/2^h)|_{pn}), \qquad j = 1,2,...,2^h-1.$$

Тогда весь процесс масштабирования распадается на два этапа: итерационное приближение и уточнение.

I. На итерационном этапе используется основной шаг. При этом числовой диапазон [0, P–1] разбивается на 2^h интервалов, а вычисления выполняются в соответствии с рекуррентным соотношением $M_i = \mathrm{floor}(M_{i-1}/2^h)$, $i = 1,2,\ldots,a$, где $a = \mathrm{floor}(\mathrm{rsh}(M)/h)$, $M_0=M$. Каждая итерация этапа состоит из трех шагов.

1. Определение формального частного (ФЧ) $M'_i = (m_1^{(i)},m_2^{(i)},...,m_n^{(i)})$ умножением предыдущего приближения M_{i-1} на мультипликативную инверсию $(2^h)^{-1}$, модулярное представление которой определяется h-й (последней) строкой в $\mathbf{H}$.

2. Вычисление ИПХ $I(M'_i/P)$ ФЧ в соответствии с алгоритмом [10] и определение номера k интервала его локализации по формуле: $k = \mathrm{floor}(t(M_i')\cdot 2^h)$, где $t(M_i')$ – верхняя граница ИПХ $I(M'_i/P)$, $k \in \{1,2,...,2^h-1\}$.

3. Корректировка ФЧ вычитанием поправки, определяемой k-й строкой матрицы $\mathbf{S}$:

$$M_i = (|m_1^{(i)}-S_{k,1}|_{p1}, |m_2^{(i)}-S_{k,2}|_{p2} ,..., |m_n^{(i)}-S_{k,n}|_{pn}), \qquad S_{k,t} = |ceil(kP/2^h)|_{pt},\ t = 1,2,...,n,$$

где n – количество модулей СОК.

Результат первого этапа – приближение $M_a = \mathrm{floor}(M/2^{h\cdot\mathrm{floor}(\mathrm{rsh}(M)/h)})$.

II. На втором этапе выполняется уточнение результата с шагом 2^b, где $b = |\mathrm{rsh}(M)|_h$. Здесь вычисляется выражение $M_r = \mathrm{floor}(M_a/2^b)$. Для этого приближение M_a (результат первого этапа), умножается на мультипликативную инверсию $(2^b)^{-1}$, модулярное представление которой определяется b-й строкой матрицы $\mathbf{H}$: $M'_r = M_a\cdot(2^b)^{-1}$. Номер интервала k, которому принадлежит M'_r, определяется также как и на первом этапе, а поправочный модулярный код выбирается из матрицы $\mathbf{S}$ со смещением $2^h/2^b$, т.е. номер нужной строки $S_{j,}$. определяется выражением: $j = k\cdot 2^h/2^b$. В результате выполнения второго этапа находится уточненный результат масштабирования:

$$M_r = \mathrm{floor}(M_a/2^b) = M'_r - S_{j,}\text{.}, \quad \text{где } S_{j,}\text{.} - j\text{-я строка матрицы } \mathbf{S}.$$

Количество итераций масштабирования по представленному алгоритму меньше в $q/(\mathrm{floor}(q/h)+1)$ раз по сравнению с алгоритмом деления отрезка пополам [12]. Алгоритм не требует преобразования модулярной мантиссы в позиционную систему счисления и обладает высоким быстродействием, а основные его шаги могут быть эффективно распараллелены.

2.2. Выравнивание порядков.

При сложении и вычитании разномасштабных величин существенную роль играет операция выравнивания порядков. Разработанный алгоритм выравнивания позволяет минимизировать потерю точности и ускорить вычисления за счет компенсации разности порядков чисел путем корректировки их мантисс. Основные этапы алгоритма состоят в следующем.

1. Для заданной разности порядков $\Delta e = e_x - e_y$ (предполагается, что $\Delta e < 0$) проверяется условие $I(M_y/P) < 2^{\Delta e}$. Если оно выполняется, то принимается $r = 0$ (где r – количество разрядов округления) и осуществляется переход к шагу 3.

2. Интервал $I(M_y/P) = [b(M_y), t(M_y)]$ уточняется с использованием алгоритма [10], далее вычисляется $r = \mathrm{ceil}(t(M_y)+|\Delta e|)$. Если $\log_2 b(M_x) > r-\log_2 P$, где $b(M_x)$ – нижняя граница ИПХ $I(M_x/P)$, то выполняется переход к шагу 3, иначе число x обнуляется, а алгоритм завершается.

3. Мантисса числа y умножается на 2^a, где $a = |\Delta e|-r$. Из порядка e_y вычитается a. Число x округляется масштабированием мантиссы M_x коэффициентом 2^r. К порядку e_x прибавляется r.

В результате работы алгоритма либо будут получены ненулевые числа x и y с одинаковыми порядками, либо одно из них будет нулем, а второе не изменится. Это позволяет применять к их мантиссам аддитивные операции.

2.3. Сложение.

Алгоритм сложения $z = x + y$ формулируется следующим образом.

1. Проверяется равенство слагаемых нулю или бесконечности. Если $x = 0$ или $y = \pm\infty$, то $z = y$ и алгоритм завершается. Если же $y = 0$ или $x = \pm\infty$, то $z = x$. Если оба операнда конечные ненулевые числа, то выполняется переход к следующему шагу.

2. Если после выравнивания одно из чисел обратилось в нуль, то алгоритм завершается и возвращает в качестве суммы исходное слагаемое, которое после выравнивания не обратилось в нуль. Иначе выполняется переход к следующему шагу.

3. Порядок суммы e_z равен порядку выровненных чисел. Поскольку знаки слагаемых равны, знак суммы определяется знаком любого из них.

4. Выполняется предвычислительный контроль переполнения, для этого ИПХ мантисс складываются по интервальной формуле

$$I(M_z/P) = [b(M_z),\, t(M_z)] = [b(M_x)+b(M_y),\, t(M_x)+t(M_y)],$$

Полученный интервал-сумма анализируется по следующим правилам.

4.1. Если верхняя граница $t(M_z)$ меньше единицы, то переполнения при сложении мантисс не возникнет, поэтому выполняется переход к шагу 5.

4.2. Если $t(M_z) \geq 1$, то для исключения переполнения мантиссы чисел x и y, а также $I(M_z/P)$ делятся на двойку, а порядок e_z увеличивается на единицу и выполняется переход к шагу 5.

5. Вычисляется сумма мантисс в модулярном преставлении:

$$M_z = M_x + M_y = (|x_1 + y_1|_{p1},\, |x_2 + y_2|_{p2},\ldots,|x_n + y_n|_{pn}).$$

2.4. Умножение.

Алгоритм умножения определяется следующим образом.

1. Вычисляются значения предикатов

$$A : |x| = 0, \qquad C : |x| = \pm\infty, \qquad E : |x| = \pm 1,$$
$$B : |x| = 0, \qquad D : |x| = \pm\infty, \qquad F : |x| = \pm 1$$

и булевы функции

$$K : (A \text{ and } \overline{D}) \text{ or } (\overline{C} \text{ and } B), \qquad L : (\overline{A} \text{ and } D) \text{ or } (\overline{B} \text{ and } C), \qquad N : (A \text{ and } D) \text{ or } (B \text{ and } C).$$

Если $E = 1$, то $|z| = |y|$; если $F = 1$, то $|z| = |x|$; если $K = 1$, то $|z| = 0$; если $L = 1$, то $|z| = \infty$; если $N = 1$, то $|z| = \text{NaN}$. Если любое из перечисленных условий выполняется, то осуществляется переход к шагу 5, иначе – к шагу 2 (если результат NaN, то также формируется сообщение об ошибке некорректной операции). Ситуации, когда один из операндов равен единицы ($E = 1$ или $F = 1$), являются штатными, поэтому могут и не обрабатываться отдельно. Однако их проверка позволяет исключить нежелательное расширение ИПХ, которое возникает при многократном умножении одного и того же числа на единицу.

2. Вычисляется ИПХ мантиссы произведения:

$$I(M_z/P) = [b(M_z),\, t(M_z)] = [\downarrow b(M_x)\cdot b(M_y)/t(1),\, \uparrow t(M_x)\cdot t(M_y)/b(1)],$$

где стрелки соответствуют направленным округлениям, а $t(1)$ и $b(1)$ – соответственно верхняя и нижняя граница ИПХ-константы $I(1/P)$. Интервал-произведение $I(M_z/P)$ анализируется по следующим правилам.

2.1. Если верхняя граница $t(M_z)$ меньше единицы, то переполнения при умножении мантисс не возникнет, поэтому необходимо сразу перейти к шагу 3.

2.2. Если верхняя граница $t(M_z)$ не меньше единицы, то определяется необходимое число итераций округления чисел x и y применением к ним алгоритма проверки округления (см. пункт 2.1) . Далее числа округляются. После округления вычисляется $I(M_z/P)$ по алгоритму [10] и выполняется переход к шагу 3.

3. Вычисляется модулярное произведение:

$$M_z = M_x M_y = (|x_1 y_1|_{p1},\, |x_2 y_2|_{p2},\ldots,|x_n y_n|_{pn}).$$

4. Позиционные порядки операндов складываются:

$$e_z = e_x + e_y.$$

5. Знаки сомножителей складываются по модулю два.

2.5. Деление.

1.Вычисляются значения предикатов

$$A : |x| = 0, \qquad C : |x| = \pm\infty, \qquad B : |x| = 0, \qquad D : |x| = \pm\infty.$$

и булевы функции

$$K : (A \text{ and } B) \text{ or } (C \text{ and } B), \qquad L : \overline{B} \text{ or } C \text{ or } \overline{D}, \qquad N : (A \text{ and } \overline{B}) \text{ or } (\overline{C} \text{ and } D), \qquad Q : \overline{A} \text{ and } B.$$

Если $K = 1$, то принимается $|z| = \text{NaN}$ и формируется сообщение «некорректная операция»; если $L = 1$, то принимается $|z| = \infty$; если $N = 1$, то $|z| = 0$; если $L = 1$, то $|z| = \infty$; если $N = 1$, то $|z| = 0$, если $Q = 1$, то $|z| = \infty$ и формируется сообщение об ошибке «деление на ноль». Если любое из перечисленных условий выполняется, осуществляется переход к завершающему шагу 6, иначе – к шагу 2.

2. На основании верхней границы ИПХ вычисляется $v = -\text{ceil}(\log_2 t(M_x))$. Мантисса делимого M_x умножается на 2^v, а из порядка e_x вычитается v.

3. Мантисса числа y округляется до floor(sqrt(P–1)). При этом возможна потеря точности.

4. Выполняется целочисленное модулярное деление: $M_z = M_x/M_y = (z_1, z_2, ..., z_n)$.

5. Вычисляется алгебраическая разность показателей: $e_z = e_x - e_y$.

6. Знаки делимого и делителя складываются по модулю два.

3. Оценка эффективности.

Для оценки эффективности разработанных алгоритмических решений были проведены эксперименты, в ходе которых выполнялись как элементарные высокоточные операции (сложение, вычитание и пр.), а также операции с накоплением. В операциях с накоплением один из операндов используется для накопления результата, а второй не изменяется. Тестовая конфигурация: Intel Core i5 3570K (3.4 GHh, 6 Mb cache L3). В качестве аналогов рассматривалась позиционная библиотека A Library for doing Number Theory (NTL) и система компьютерной алгебры Wolfram Mathematica 10. Результаты в виде временной сложности выполнения различных арифметических операций представлены в таблице 1.

Таблица 2. Результаты эксперимента

Операция	Время выполнения, нс			
	МП-формат без векторизации	МП-формат с векторизацией	NTL	Wolfram Mathematica
Сложение	125,9	58,6	161	480,2
Вычитание	172,8	61,2	167,4	656,5
Сравнение	103,9	51.9	85,9	245,5
Умножение	89.9	45,9	546,2	415,9
Сложение с накоплением	96.5	45.6	190	660,5
Вычитание с накоплением	105.4	68.8	196,7	946
Умножение с накоплением	130,8	70.2	694	632,4

Заключение.

1. Предложен новый модулярно-позиционный формат с плавающей точкой для представления чисел большой разрядности при организации высокоточных арифметических вычислений. В основе базирующейся на предложенном формате арифметики лежат принципы разрядно-параллельной обработки чисел в системах остаточных классов. Данный формат может быть использован для повышения скорости расчетов, критичных к ошибкам округления, на многоядерных вычислительных системах, графических процессорах и специализированных аппаратных ускорителях.

2. Получено экспериментальное подтверждение эффекта повышения скорости выполнения основных операций многоразрядной арифметики в новом формате по сравнению с аналогами на основе позиционных систем счисления и символьных вычислений.

3. На примере использования векторизации была показана эффективность применения параллельной обработки. При выполнении арифметических операций в среднем было получено двукратное ускорение по сравнению с расчетами без использования векторизации.

Работа выполнена при финансовой поддержке РФФИ в рамках научного проекта № 14-07-31075 мол_а.

ЛИТЕРАТУРА:

1. D.H. Bailey, J.M. Borwein High-Precision Arithmetic: Progress and Challenges.– Electronic text data. – 2013. – Mode of access: http://www.davidhbailey.com/dhbpapers/hp-arith.pdf. – The title from the screen.
2. И.Я. Акушский, Д.И. Юдицкий Машинная арифметика в остаточных классах. – М. : Сов. Радио, 1968. – 440 с.
3. К.С. Исупов Методика выполнения базовых немодульных операций в модулярной арифметике с применением интервальных позиционных характеристик // Известия высших учебных заведений. Поволжский регион. Технические науки. – 2013. – № 3 (27). – С. 26–39.
4. Ш.А. Оцоков, Структурно-алгоритмические методы организации высокоточных вычислений на основе теоретических обобщений в модулярной системе счисления [Текст] : дис. … доктора технических наук : 05.13.05, 05.13.15 / Оцоков Шамиль Алиевич. – М., 2010. – 287 с.
5. A. Sasaki The Basis for Implementation of Additive operations in the Residue Number System [Text] / A. Sasaki // IEEE Transactions on Computers. − 1968. − Vol. 17, No. 11. – P. 1066−1073. − ISSN 0018-9340.
6. М.К. Буза, О плавающей запятой в системе счисления в остаточных классах [Текст] / М.К. Буза, В.К. Кравцов, Н.Н. Поснов // Вестник БГУ. – 1969. − Серия 1, № 3. – С. 21−27. – ISSN 1994-0866.

7. E.A. Kinoshita, Floating-Point Arithmetic Algorithms in the Symmetric Residue Number System [Text] / E. Kinoshita, H. Kosako, Y. Kojima // IEEE Transactions on Computers. – 1974. – Vol. C−23, No. 1. – P. 9−20. – ISSN 0018-9340.
8. J.-S. Chiang, Floating-Point Numbers in Residue Number Systems [Text] / J.-S. Chiang, M. Lu // Computers and Mathematics with Applications. – 1991. – Vol. 22, No. 5. – P. 127−140. – ISSN 0898–1221.
9. E.A. Kinoshita, Residue Arithmetic Extension for Reliable Scientific Computation [Text] / E. Kinoshita, K.-J. Lee // IEEE Transactions on Computers. − 1997. − Vol. 46, No. 2. – P. 129−138. − ISSN 0018-9340.
10. К.С. Исупов Алгоритм вычисления интервально-позиционной характеристики для выполнения немодульных операций в системах остаточных классов // Вестник ЮУрГУ. Серия: Компьютерные технологии, управление, радиоэлектроника. – Челябинск : Издательский центр ЮУрГУ. – 2014. – Т. 14, № 1. – С.89–97.
11. H.M. Yassine, Improved mixed-radix Conversion for Residue Number Architectures [Text] / H. M. Yassine, W. R. Moore // Circuits, Devices and Systems, IEEE proceedings. – 1991. – Vol. 138, Issue 1. – P. 120–124. – ISSN 0956-3768.
12. К.С. Исупов, А.Н. Мальцев Модулярное масштабирование степенью двойки с произвольным шагом [Электронный ресурс] // Общество, наука, инновации (НПК-2014) : Сб. материалов ежегодной Всероссийской научно-практической конф. (15−26 апреля 2014 г., г. Киров). − Электрон. текстовые дан. – Киров : Изд-во ВятГУ, 2014. − С. 1179−1184. − 1 электрон. опт. диск (CD-ROM). − Загл. с экрана. − ISBN 978-5-98228-073-2.