УДК 519.6

СПОСОБ ПРЕДСТАВЛЕНИЯ ЧИСЕЛ С ПЛАВАЮЩЕЙ ТОЧКОЙ БОЛЬШОЙ РАЗРЯДНОСТИ, ОРИЕНТИРОВАННЫЙ НА ПАРАЛЛЕЛЬНУЮ ОБРАБОТКУ

К. С. Исупов[1], **А. Н. Мальцев**[2]

При решении многих научных и инженерных задач возникает необходимость повышения точности вычислений, при этом критичным параметром является время решения, что требует разработки новых быстродействующих методов многоразрядной арифметики. В настоящей статье предложен новый модулярно-позиционный формат для представления многоразрядных чисел с плавающей точкой. В его основе лежит использование систем остаточных классов для представления и разрядно-параллельной обработки мантисс чисел. Для повышения скорости при выполнении сложных немодульных операций используется метод интервально-позиционных характеристик. Рассматриваются алгоритмы выполнения арифметических операций и округления чисел в модулярно-позиционном формате с плавающей точкой. Приводятся результаты исследования эффективности их векторизации, а также быстродействия по сравнению с аналогами: MPFR (Multiple Precision Floating-Point Reliable library), NTL (Number Theory Library) и Wolfram Mathematica.

Ключевые слова: система остаточных классов, высокоточные вычисления, модулярно-позиционный формат с плавающей точкой, многоразрядные числа, арифметические операции, высокое быстродействие.

1. Введение. Для большинства научных вычислений, особенно тех, что используют эмпирические данные, 32-битная арифметика спецификации IEEE-754 имеет достаточную точность. Эта арифметика 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, т.е. при определенных обстоятельствах результат получается абсолютно неверным [1]. Ситуация осложняется тем, что даже при небольшом объеме вычислений может быть получен результат, не содержащий ни одной значащей цифры [2].

В общем случае вычисления, требующие повышенной точности, отличаются наличием одной или более из следующих составляющих [3, 4]:

а) плохо обусловленные линейные системы;

б) крупные суммирования;

в) длительное моделирование;

г) масштабные высокопараллельные расчеты;

д) экспериментальные математические расчеты.

В настоящее время для выполнения таких вычислений наиболее часто применяется арифметика с точностью, примерно в два раза большей, чем стандартная 64-битная арифметика 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 [1]. Более распространенный вариант длинной арифметики, реализованный в программном обеспечении, известен как формат “double-double” [5]. В этом формате число представляется в виде пары 64-битных чисел x_h и x_l, где x_h является числом с плавающей точкой, ближайшим к истинному значению, а x_l — разница (положительная или отрицательная) между истинным значением и x_h. Для сложения и умножения чисел в таком формате используются алгоритмы Деккера. На подобных принципах основан формат “quad-double”. Кроме этого, существуют несколько свободно доступных программных пакетов, поддерживающих арифметику многократной или произвольной точности. Среди таких пакетов упомянем следующие: ARPREC

[1] Вятский государственный университет, факультет автоматики и вычислительной техники, ул. Московская, 36, 610000, Россия, Кировская обл., г. Киров; ассистент, e-mail: isupov.k@gmail.com

[2] Вятский государственный университет, факультет автоматики и вычислительной техники, ул. Московская, 36, 610000, Россия, Кировская обл., г. Киров; аспирант, e-mail: maltsev_a@list.ru

(Arbitrary Precision computation package — поддерживает вещественные, целые и комплексные числа произвольной точности, а также многие алгебраические и трансцендентные функции); GMP (GNU Multiple Precision library — поддерживает вычисления с целыми числами, рациональными числами и числами с плавающей точкой); MPFR (Multiple Precision Floating-Point Reliable library — поддерживает вычисления с плавающей точкой с различной точностью и правильным округлением), QD (Quad-Double library — включает в себя процедуры "double-double" и "quad-double" арифметики, а также многие алгебраические и трансцендентные функции), NTL (Number Theory Library — включает в себя 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 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 один такой формат, основанный на системе остаточных классов.

2. 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):

$$x \to \{s, M, \lambda, I(M/P)\}, \tag{1}$$

где $s \in \{0, 1\}$ — знак числа x; $M = \langle m_1, \ldots, m_n \rangle$ — мантисса, представленная в системе остаточных классов (СОК) [6, 7] набором независимых остатков по взаимно-простым модулям $p_1, \ldots, p_n$; λ — порядок числа, изменяющийся в пределах интервала, ограниченного двумя целыми числами $\lambda_{\min}$ и $\lambda_{\max}$; $I(M/P) = \left[\underline{M/P}, \overline{M/P}\right]$ — интервально-позиционная характеристика (ИПХ) модулярной мантиссы, являющаяся интервальной аппроксимацией ее относительной величины [8, 9], причем $\underline{M/P}$ и $\overline{M/P}$ — верхняя и нижняя границы ИПХ, представленные машинными числами с плавающей точкой соответственно.

Значение модулярной мантиссы может быть найдено в соответствии с китайской теоремой об остатках и представляет собой целое число, изменяющееся в пределах диапазона $[0, P-1]$, где $P = \prod_{i=1}^{n} p_i$. Таким образом, варьирование количества модулей позволяет задавать произвольно высокую точность вычислений. Значение числа в формате (1) определяется выражением

$$x = (-1)^s \cdot |m_1 B_1 + m_2 B_2 + \ldots + m_n B_n| \cdot 2^{\lambda},$$

где $B_1, B_2, \ldots, B_n$ — ортогональные базисы системы остаточных классов. Схема МП-формата представлена на рис. 1.

Знак	Порядок	Модулярная мантисса			Интервально-позиционная характеристика	
integer	integer	integer	...	integer	real	real

Рис. 1. Модулярно-позиционный формат с плавающей точкой для представления многоразрядных чисел: integer — целочисленный тип данных, real — вещественный тип (с плавающей точкой)

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 переносов, следовательно, просто и быстро. При этом появляется возможность применения методов параллельной обработки знакопозиций мантиссы путем векторизации или 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.

Основным отличием МП-формата от ранее известных способов представления рациональных/вещественных чисел на основе модулярных систем [10–15] является включение ИПХ модулярной мантиссы (двух чисел с плавающей точкой $\underline{M/P}$ и $\overline{M/P}$) в представление числа. Это позволяет ускорить вычисление результата трудоемких немодульных операций, таких как сравнение, определение знака, оценка

переполнения допустимого диапазона представления, округление и пр. [8, 9]. 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 [16]. Таким образом, сложность основных немодульных операций определяется функцией $O(n^2)$. В свою очередь, вычисление ИПХ в соответствии со специальным алгоритмом [17] 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 тоже 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 (рассматривается далее).

Помимо числовых кодировок, для МП-формата определены способы представления бесконечностей и нечисловых величин (табл. 1). Нечисловые величины не могут быть исходными данными арифметических операций, но могут быть получены в ходе их выполнения. Мантиссы нуля и бесконечностей представлены кодом $\langle 0, 0, \ldots, 0\rangle$, поэтому для однозначной интерпретации значения модулярно-позиционной величины необходим анализ порядка λ: для нуля $\lambda = 0$, а для бесконечности $\lambda = \lambda_{\max} + 1$.

Таблица 1

Кодировки значений модулярно-позиционных величин с плавающей точкой

Класс	Порядок, λ	Модулярная мантисса со знаком, $\pm M$
Ноль со знаком	0	$\pm\langle 0, 0, \ldots, 0\rangle$
Числовые величины	$\lambda_{\min} \leqslant \lambda \leqslant \lambda_{\max}$	$\pm\langle m_1, \ldots, m_n\rangle$, имеются $m_i \neq 0$
Бесконечность со знаком	$\lambda_{\max} + 1$	$\pm\langle 0, 0, \ldots, 0\rangle$
Нечисловые величины (NaN)	$\lambda_{\max} + 1$	$\pm\langle m_1, \ldots, m_n\rangle$, имеются $m_i \neq 0$

3. 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.

3.1. Округление. Округление — ключевая операция при реализации любых нецелочисленных вычислений, в особенности итерационных, поэтому основным требованием к алгоритму его выполнения является высокая скорость. Существуют две схемы округления: поствычислительная и предвычислительная (рис. 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 $\sqrt{P-1}$, 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 находиться в допустимом диапазоне $[0, P-1]$. Наибольший эффект предвычислительного округления достигается при выполнении аддитивных операций, которые не приводят к существенному росту разрядности мантиссы результата.

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 M для того, чтобы при выполнении следующего арифметического действия над ней не произошло переполнения;

— собственно округление, состоящее в уменьшении значения модулярной мантиссы в $2^{rsh(M)}$ 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 $P' = \lfloor\sqrt{P-1}\rfloor$, где $\lfloor\cdot\rfloor$ — округление к меньшему целому, P — произведение модулей СОК. Разобьем полный модулярный диапазон $D = [0, P-1]$ на два непересекающихся интервала:

$$D_{\text{work}} = [0, P'], \quad D_{\text{crit}} = [P'+1, P-1].$$

Интервал D_{work} назовем рабочим диапазоном; если $M \in D_{\text{work}}$, то округления числа x не требуется. Интервал D_{crit} — это критический диапазон, поскольку при $M \in D_{\text{crit}}$ 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 $2^{rsh(M)} = M/P'$. Логарифмируя обе его части с учетом целочисленности показателя степени, получим

$$rsh(M) = \lceil \log_2(M/P') \rceil, \qquad (2)$$

где $\lceil\cdot\rceil$ — округление к большему целому. Равенство (2) определяет алгоритм проверки необходимости округления при выполнении операции умножения, согласно которому

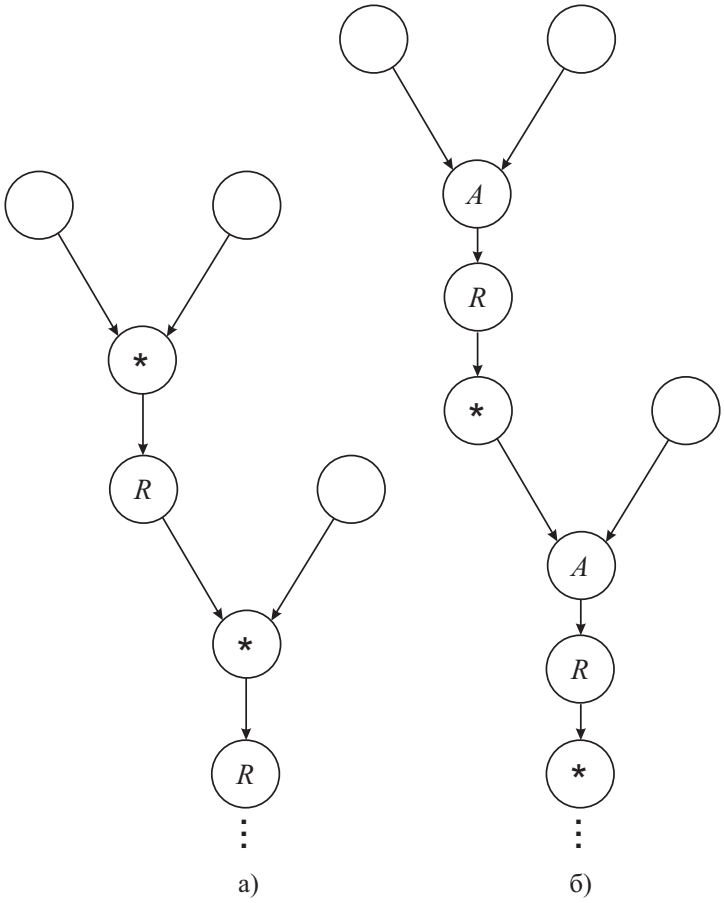

Рис. 2. Поствычислительная (а) и предвычислительная (б) схемы округления: $*$ — арифметическая операция, A — проверка необходимости округления, R — округление

$$rsh(M) = \max\left\{ \left\lceil \log_2 \underline{M/P} - \log_2 \overline{\alpha} \right\rceil, \left\lceil \log_2 \overline{M/P} - \log_2 \underline{\alpha} \right\rceil, 0 \right\},$$

где $\log_2 I(\alpha) = [\log_2 \underline{\alpha}, \log_2 \overline{\alpha}]$, $\alpha = \lfloor\sqrt{P-1}\rfloor / P$ — константы. При известной интервально-позиционной характеристике $I(M/P)$ сложность алгоритма инвариантна к числу модулей СОК.

Далее, если $rsh(M)$ больше нуля, то необходимо выполнить операцию округления, которая осуществляется в соответствии с алгоритмом 1.

Алгоритм 1. Округление чисел в модулярно-позиционном формате с плавающей точкой.

Входные данные: $x \to \{s, M, \lambda, I(M/P)\}$, $rsh(M)$.

Выходные данные: округленное число $\text{round}(x) \to \{s_r, M_r, \lambda_r, I(M_r/P)\}$.

1. Если $rsh(M) \leqslant 0$, то принять $\text{round}(x) = x$ и завершить алгоритм, иначе — к следующему шагу.
2. Масштабировать модулярную мантиссу M коэффициентом $2^{rsh(M)}$: $M_r = \lfloor M/2^{rsh(M)} \rfloor$.
3. Увеличить порядок: $\lambda_r = \lambda + rsh(M)$, принять $s_r = s$.

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

5. Вычислить ИПХ округленной мантиссы M_r в соответствии с высокоточным алгоритмом [17].

Наиболее трудоемким является шаг 2, поскольку операция масштабирования чисел в СОК является немодульной для модулярной арифметики. Для модулярного масштабирования чисел разработан новый итерационный метод [18], основная суть которого состоит в следующем. Пусть дана модулярная мантисса $M = \langle m_1, \ldots, m_n \rangle$ и требуется определить частное $M_r = \langle r_1, \ldots, r_n \rangle$, такое, что $M_r = \lfloor M/2^{rsh(M)} \rfloor$. Зафиксируем 2^h — основной (максимальный) шаг масштабирования. Предварительно вычислим модулярные коды мультипликативных инверсий первых h натуральных степеней двойки (в теории чисел доказано, что при нечетных модулях $p_1, p_2, \ldots, p_n$ все степени двойки обратимы в кольце вычетов Z/PZ), которые для удобства представим в виде матрицы размера $h \times n$, каждая i-я строка которой определяет модулярный код мультипликативной инверсии i-й степени двойки:

$$H = \begin{pmatrix} \left|\left(2^1\right)^{-1}\right|_{p_1} & \left|\left(2^1\right)^{-1}\right|_{p_2} & \cdots & \left|\left(2^1\right)^{-1}\right|_{p_n} \\ \left|\left(2^2\right)^{-1}\right|_{p_1} & \left|\left(2^2\right)^{-1}\right|_{p_2} & \cdots & \left|\left(2^2\right)^{-1}\right|_{p_n} \\ \cdots & \cdots & \ddots & \cdots \\ \left|\left(2^h\right)^{-1}\right|_{p_1} & \left|\left(2^h\right)^{-1}\right|_{p_2} & \cdots & \left|\left(2^h\right)^{-1}\right|_{p_n} \end{pmatrix}. \tag{3}$$

Кроме того, вычислим матрицу модулярных поправок размера $\left(2^h - 1\right) \times n$:

$$S = \begin{pmatrix} \left|\lceil P/2^h \rceil\right|_{p_1} & \left|\lceil P/2^h \rceil\right|_{p_2} & \cdots & \left|\lceil P/2^h \rceil\right|_{p_n} \\ \left|\lceil 2P/2^h \rceil\right|_{p_1} & \left|\lceil 2P/2^h \rceil\right|_{p_2} & \cdots & \left|\lceil 2P/2^h \rceil\right|_{p_n} \\ \cdots & \cdots & \ddots & \cdots \\ \left|\lceil (2^h-1)P/2^h \rceil\right|_{p_1} & \left|\lceil (2^h-1)P/2^h \rceil\right|_{p_2} & \cdots & \left|\lceil (2^h-1)P/2^h \rceil\right|_{p_n} \end{pmatrix}. \tag{4}$$

Тогда процесс масштабирования распадается на два этапа: итерационное приближение и уточнение результата, которые реализуются алгоритмом 2.

Алгоритм 2. Модулярное масштабирование чисел степенью двойки [18].

Входные данные: $M = \langle m_1, \ldots, m_n \rangle$, $rsh(M)$.

Выходные данные: $M_r = \lfloor M/2^{rsh(M)} \rfloor = \langle r_1, \ldots, r_n \rangle$.

I. *Итерационный этап.* Используется основной (максимальный) шаг масштабирования. При этом числовой диапазон $[0, P-1]$ разбивается на 2^h интервалов, а вычисления выполняются в соответствии с рекуррентным соотношением $M_i = \lfloor M_{i-1}/2^h \rfloor$, $i = 1, 2, \ldots, a$, где $a = \lfloor rsh(M)/h \rfloor$, $M_0 = M$. Каждая итерация этапа состоит из трех шагов.

1. Вычисление формального частного (ФЧ) $\dot{M}_i = \left\langle \dot{m}_1^{(i)}, \dot{m}_2^{(i)}, \ldots \dot{m}_n^{(i)} \right\rangle$ умножением предыдущего приближения M_{i-1} на мультипликативную инверсию числа 2^h, модулярное представление которой определяется h-й (последней) строкой матрицы (3).
2. Высокоточное вычисление ИПХ ФЧ $I(\dot{M}_i/P) = \left[\underline{\dot{M}_i/P}, \overline{\dot{M}_i/P}\right]$ по алгоритму [17] и определение номера $k \in \left\{1, 2, \ldots, 2^h - 1\right\}$ интервала его локализации по формуле $k = \left\lfloor \overline{\dot{M}_i/P} \cdot 2^h \right\rfloor$.
3. Корректировка ФЧ вычитанием поправки, код которой определяется k-й строкой матрицы (4): $M_i = \left\langle \left|\dot{m}_1^{(i)} - S_{k1}\right|_{p_1}, \left|\dot{m}_2^{(i)} - S_{k2}\right|_{p_2}, \ldots, \left|\dot{m}_n^{(i)} - S_{kn}\right|_{p_n} \right\rangle$, $S_{ij} = \left|\lceil iP/2^h \rceil\right|_{p_j}$, $j = 1, 2, \ldots, n$, где n — количество модулей СОК.

Результат первого этапа — приближение $M_a = \left\lfloor M/2^{h\lfloor rsh(M)/h\rfloor}\right\rfloor$.

II. *Уточняющий этап.* Выполняется уточнение результата модулярного масштабирования с шагом 2^b, где $b = |rsh(M)|_h$. Здесь вычисляется выражение $M_r = \left\lfloor M_a/2^b\right\rfloor$. Для этого приближение M_a (результат первого этапа) умножается на мультипликативную инверсию числа 2^b, модулярное представление которой определяется b-й строкой матрицы H : $\dot{M}_r = M_a \cdot \left(2^b\right)^{-1}$. Номер интервала k, которому принадлежит $\dot{M}_r$, определяется так же, как и на первом этапе, а поправочный модулярный код выбирается из матрицы S со смещением $2^h/2^b$, т.е. номер нужной строки $S_{j,\bullet}$ определяется выражением $j = k2^h/2^b$. В результате выполнения второго этапа находится уточненный результат масштабирования: $M_r = \left\lfloor M_a/2^b\right\rfloor = \dot{M}_r - S_{j,\bullet}$, где $S_{j,\bullet}$ — j-я строка матрицы S.

Количество итераций масштабирования по представленному алгоритму меньше в $q/\left(\lfloor q/h\rfloor+1\right)$ раз по сравнению с алгоритмом деления отрезка пополам. Эта оценка в полной мере подтверждена результатами экспериментов [18]. Приведенный алгоритм масштабирования примечателен тем, что не требует преобразования модулярной мантиссы в позиционную систему счисления и обладает высоким быстродействием, а основные его шаги могут быть эффективно распараллелены по модулям СОК.

3.2. Выравнивание порядков. При сложении и вычитании разномасштабных величин существенную роль играет операция выравнивания порядков. Алгоритм выравнивания 3 позволяет минимизировать потерю точности и ускорить вычисления за счет компенсации разности порядков чисел путем корректировки их мантисс.

Алгоритм 3. Выравнивание порядков чисел в модулярно-позиционном формате с плавающей точкой.

Входные данные: $x \to \left\{s_x, M_x, \lambda_x, I(M_x/P)\right\}$, $y \to \left\{s_y, M_y, \lambda_y, I(M_y/P)\right\}$.

Выходные данные: числа x, y с выравненными порядками $\lambda_x = \lambda_y$.

1. Для заданной разности порядков $\Delta\lambda = \lambda_x - \lambda_y$ (предполагается, что $\Delta\lambda < 0$, иначе числа x и y меняются местами) проверяется условие $I(M_y/P) < 2^{\Delta\lambda}$. Если оно выполняется, то принимается $r = 0$ (где r — количество разрядов округления) и осуществляется переход к шагу 3.

2. Интервал $I(M_y/P) = \left[\underline{M_y/P}, \overline{M_y/P}\right]$ уточняется с использованием алгоритма [17], далее вычисляется $r = \left\lceil \log_2 \overline{M_y/P} + |\Delta\lambda|\right\rceil$. Если $\log_2 \underline{M_x/P} > (r - \log_2 P)$, где $\underline{M_x/P}$ — нижняя граница $I(M_x/P)$, то выполняется переход к шагу 3, иначе число x обнуляется, а алгоритм завершается.

3. Мантисса числа y умножается на 2^a, где $a = |\Delta\lambda| - r$. Из порядка λ_y вычитается показатель a. Число x округляется масштабированием мантиссы M_x коэффициентом 2^r. К порядку λ_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 выполнять над ними аддитивные операции.

3.3. Сложение. Результат арифметического сложения двух чисел с плавающей точкой одинаковых знаков, представленных в формате (1), определяется указанной справа таблицей истинности, где x и y — конечные ненулевые числа. Рассмотрим алгоритм, реализующий эту таблицу.

	0	x	$\pm\infty$
0	0	x	$\pm\infty$
y	y	$(x+y) \vee \pm\infty$	$\pm\infty$
$\pm\infty$	$\pm\infty$	$\pm\infty$	$\pm\infty$

Алгоритм 4. Сложение чисел в модулярно-позиционном формате с плавающей точкой.

Входные данные: $x \to \left\{s_x, M_x, \lambda_x, I(M_x/P)\right\}$, $y \to \left\{s_y, M_y, \lambda_y, I(M_y/P)\right\}$.

Выходные данные: $z = x + y \to \left\{s_z, M_z, \lambda_z, I(M_z/P)\right\}$.

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. Порядки чисел выравниваются. Если после выравнивания одно из чисел равно нулю, 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 λ_z равен порядку выравненных чисел. Поскольку знаки слагаемых равны, знак суммы s_z определяется знаком любого из них.

4. Выполняется предвычислительный контроль переполнения, для этого ИПХ мантисс складываются по интервальной формуле $I(M_z/P) = \left[\underline{M_z/P}, \overline{M_z/P}\right] = \left[\underline{M_x/P} + \underline{M_y/P}, \overline{M_x/P} + \overline{M_y/P}\right]$,

   (a) если верхняя граница $\overline{M_z/P}$ меньше единицы, то переполнения при сложении мантисс не возникнет, поэтому выполняется переход к шагу 5;

   (b) если $\overline{M_z/P} \geqslant 1$, то (при условии, что $\lambda_z < \lambda_{\max}$) мантиссы чисел x и y, а также обе границы ИПХ $I(M_z/P)$ делятся на двойку, а порядок λ_z увеличивается на единицу; если же $\lambda_z = \lambda_{\max}$, то формируется код переполнения, сумме z присваивается кодировка бесконечности (см. табл. 1), и алгоритм завершается.

5. Вычисляется сумма мантисс $M_x = \langle x_1, \ldots, x_n\rangle$ и $M_y = \langle y_1, \ldots, y_n\rangle$ в модулярном представлении:

$$M_z = M_x + M_y = \left\langle \left|x_1 + y_1\right|_{p_1}, \left|x_2 + y_2\right|_{p_2}, \ldots, \left|x_n + y_n\right|_{p_n} \right\rangle.$$

Алгоритмы выполнения операций вычитания и сравнения во многом схожи с представленным алгоритмом сложения и выполняются по двухэтапной схеме: вначале производится выравнивание порядков операндов, а затем непосредственно выполнение арифметических действий над мантиссами. При вычитании мантиссы представляются в симметричной системе остаточных классов [6], а для определения знака результата используется техника интервально-позиционных характеристик.

3.4. Умножение. Результат умножения двух чисел, представленных в МП-формате (1), определяется указанной справа таблицей истинности, где x и y — конечные ненулевые числа. Рассмотрим алгоритм 5.

	0	x	$\pm\infty$
0	0	0	NaN
y	0	xy	$\pm\infty$
$\pm\infty$	NaN	$\pm\infty$	$\pm\infty$

Алгоритм 5. Умножение чисел в модулярно-позиционном формате с плавающей точкой.

Входные данные: $x \to \left\{s_x, M_x, \lambda_x, I(M_x/P)\right\}$, $y \to \left\{s_y, M_y, \lambda_y, I(M_y/P)\right\}$.

Выходные данные: $z = x \cdot y \to \left\{s_z, M_z, \lambda_z, I(M_z/P)\right\}$.

1. Вычисляются значения следующих булевых функций:

$$\begin{aligned} &A:\ |x| = 0, && B:\ |y| = 0, && K:\ (A \wedge \neg D) \vee (\neg C \wedge B),\\ &C:\ x = \pm\infty, && D:\ y = \pm\infty, && L:\ (\neg A \wedge D) \vee (\neg B \wedge C),\\ &E:\ x = \pm 1, && F:\ y = \pm 1, && N:\ (A \wedge D) \vee (B \wedge C). \end{aligned}$$

   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 (E, F, K, L, N) выполняется, то алгоритм завершает работу (если результат NaN, то формируется сообщение об ошибке некорректной операции); знак результата при этом определяется сложением по модулю два знаков операндов.

2. По интервальной формуле вычисляется ИПХ мантиссы произведения:

$$I(M_z/P) = \left[\underline{M_z/P}, \overline{M_z/P}\right] = \left[\downarrow \frac{\underline{M_x/P} \cdot \underline{M_y/P}}{\overline{1/P}}, \uparrow \frac{\overline{M_x/P} \cdot \overline{M_y/P}}{\underline{1/P}}\right],$$

   где стрелки соответствуют направленным округлениям, а константы $\overline{1/P}$ и $\underline{1/P}$ — верхняя и нижняя границы ИПХ $I(1/P)$ соответственно. Далее полученный интервал-произведение $I(M_z/P)$ анализируется по следующим правилам.

   (a) Если $\overline{M_z/P} < 1$, то переполнения при умножении мантисс не возникнет, поэтому осуществляется переход к шагу 3.

   (b) Если $\overline{M_z/P} \geqslant 1$, то определяется необходимое количество итераций округления чисел x и y (см. пункт 3.1). Далее числа округляются, после округления вычисляется $I(M_z/P)$ по алгоритму [17], и выполняется переход к шагу 3.

3. Вычисляется модулярное произведение: $M_z = M_x \cdot M_y = \left\langle \left|x_1 \cdot y_1\right|_{p_1}, \left|x_2 \cdot y_2\right|_{p_2}, \ldots, \left|x_n \cdot y_n\right|_{p_n} \right\rangle$.

4. Вычисляется алгебраическая сумма порядков: $\lambda_z = \lambda_x + \lambda_y$.

5. Знаки сомножителей складываются по модулю два: $s_z = s_x \oplus s_y$.

3.5. Деление. Результат деления двух чисел определяется указанной справа таблицей истинности (значения делимого записаны по столбцам, а делителя — по строкам), где x и y — конечные ненулевые числа; знак результата — сумма по модулю два знаков операндов.

	0	x	$\pm\infty$
0	NaN	$\pm\infty$	$\pm\infty$
y	0	x/y	$\pm\infty$
$\pm\infty$	0	0	NaN

Пусть x — делимое, а y — делитель, представленные в формате (1). Поскольку позиционные значения мантисс M_x и M_y — целые числа, остаток от их деления отбрасывается; во избежание потери точности результата мантисса делимого должна быть значительно больше мантиссы делителя (так, чтобы значение отбрасываемого остатка было малым по отношению к значению частного). Для этого необходимо умножить M_x на максимально допустимую степень двойки, а число y округлить.

Алгоритм 6. Деление чисел в модулярно-позиционном формате с плавающей точкой.

Входные данные: $x \to \{s_x, M_x, \lambda_x, I(M_x/P)\}$, $y \to \{s_y, M_y, \lambda_y, I(M_y/P)\}$.

Выходные данные: $z = x/y \to \{s_z, M_z, \lambda_z, I(M_z/P)\}$.

1. Вычисляются булевы функции:

$$A:\ |x| = 0, \qquad B:\ |y| = 0, \qquad K:\ (A \wedge B) \vee (C \wedge D), \qquad L:\ \neg B \wedge C \wedge \neg D$$
$$C:\ x = \pm\infty, \qquad D:\ y = \pm\infty, \qquad N:\ (A \wedge \neg B) \vee (\neg C \wedge D), \qquad Q:\ \neg A \wedge B.$$

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 $Q = 1$, то $|z| = \infty$ и формируется сообщение об ошибке “деление на ноль”. Если любое из этих условий выполняется, то знак результата определяется сложением по модулю два знаков делимого и делителя и алгоритм завершает работу. Иначе (ни одно из условий не выполнимо) осуществляется переход к шагу 2.

2. На основании верхней границы ИПХ числа x вычисляется $v = -\left\lceil \log_2 \overline{M_x/P} \right\rceil$. Мантисса делимого M_x умножается на 2^v, а из порядка λ_x вычитается v.

3. Мантисса числа y округляется до $\left\lfloor \sqrt{P-1} \right\rfloor$ (при этом возможна потеря точности).

4. Выполняется целочисленное модулярное деление $M_z = M_x/M_y = \langle z_1, \ldots, z_n \rangle$ и вычисляется интервально-позиционная характеристика мантиссы частного $I(M_z/P)$.

5. Вычисляется алгебраическая разность порядков: $\lambda_z = \lambda_x - \lambda_y$.

6. Знаки делимого и делителя складываются по модулю два: $s_z = s_x \oplus s_y$.

Наиболее сложный шаг алгоритма — это деление модулярных мантисс, состоящее в определении такого числа M_z, что $M_x = M_y M_z + R$ и $0 \leqslant R < M_z$. Данная операция требует в общем случае преобразования мантисс в позиционную систему счисления и имеет сложность $O(n^2)$.

Замечание. Известен широкий спектр методов модулярного деления, не требующих в явном виде преобразования чисел из СОК в позиционную систему. К ним относятся, в частности, SRT [7, с. 201], метод на основе приближенного определения знака числа [19], метод итераций Ньютона [20], метод на базе техники контроля четности и бинарного поиска [21], метод бисекции [22]. Однако многие из них чувствительны к величине делимого/делителя и на практике часто оказываются не менее затратными, чем классический метод на основе преобразования в позиционную систему. Поэтому задача разработки быстрого алгоритма модулярного деления на сегодня является актуальной темой для научных исследований.

4. Оценка эффективности. Для оценки эффективности разработанного формата представления чисел и алгоритмов высокоточной арифметики был проведен ряд экспериментов. Конфигурация тестовой ЭВМ: Intel Core i5-3570K 3.40 GHz / 6M Cache / 8 Gb RAM / Intel C++ Compiler v. 13.0. Во всех экспериментах точность вычислений составляла 239 бит (≈ 72 десятичные цифры).

Первый эксперимент. Цель — получение показателей эффективности векторизации алгоритмов. Исследовались операции сложения (add), вычитания (sub), умножения (mult), сравнения (cmp), сложения с накоплением (aac, $x = x + y$), вычитания с накоплением (sac, $x = x - y$), умножения с накоплением (mac, $z = z + xy$) и деления (div). Тестовые данные (псевдослучайные 239-битные числа) генерировались алгоритмом "Вихрь Мерсенна", а СОК для представления модулярных мантисс была задана 32 модулями с произведением $P \approx 2^{479}$. Остатки мантисс представлялись в 32-битном целочисленном формате, а границы ИПХ — в формате с плавающей точкой двойной точности. Для оценки эффективности векторизации циклов обработки модулярных мантисс и интервально-позиционных характеристик запуск программных реализаций алгоритмов выполнялся в двух конфигурациях: при установленном запрете на векторизацию (прописыванием директив #pragma novector) и при использовании средств автоматической векторизации компилятора Intel C++ Compiler (#pragma simd). Эффективность векторизации оценивалась как отношение полученного ускорения к максимальному числу одновременно обрабатываемых пар операндов. Результаты экспериментов представлены в табл. 2. Низкая эффективность векторизации операции деления объясняется необходимостью преобразования модулярных мантисс в позиционную систему счисления.

Таблица 2

Результаты исследования эффективности векторизации

Операция	Время выполнения без векторизации, нс	Время выполнения с векторизацией, нс	Ускорение	Эффективность
add	106	42	2.51	0.63
sub	97	42	2.31	0.58
mult	72	28	2.59	0.65
div	6910	6333	1.09	0.27
cmp	91	34	2.68	0.67
aac	108	42	2.54	0.64
sac	100	45	2.21	0.55
mac	561	309	1.82	0.45

Таким образом, при векторизации циклов вычисления модулярных мантисс и интервально-позиционных характеристик скорость выполнения операций сложения, вычитания, умножения, сравнения, сложения с накоплением, вычитания с накоплением и умножения с накоплением увеличилась в среднем в 2.38 раз с эффективностью 0.60. Хуже всех векторизуется деление (ускорение 1.09 раз, эффективность 0.27).

Второй эксперимент. Цель — сравнение производительности алгоритмов вычислений в МП-формате (с векторизацией) со следующими аналогами: NTL v. 6.1.0, MPFR v. 3.0.1 [23] и Wolfram Mathematica v. 10.0. В результате эксперимента (рис. 3) получены следующие усредненные оценки ускорения алгоритмов высокоточной арифметики по сравнению с аналогами (ввиду сильного разброса в качестве среднего использована медиана распределения): 4.35 раз по сравнению с пакетом NTL, 1.32 раза по сравнению с библиотекой MPFR и 13.15 раз по сравнению с системой компьютерной алгебры Wolfram Mathematica. Наиболее эффективная операция — это умножение, которое выполняется быстрее в 7.39 раз самого быстродействующего из исследованных аналогов — библиотеки MPFR. Деление, напротив, является наиболее сложной операцией, что связано с потребностью преобразования модулярной мантиссы в позиционную систему. Снижение быстродействия операции умножения с накоплением (mac) в МП-формате объясняется необходимостью выполнения множественных округлений (при выполнении каждой итерации, начиная со второй, возникает необходимость округления операндов в среднем на 238 бит).

Третий эксперимент. Цель — получение оценок времени высокоточного умножения квадратных плотных матриц. Порядок матриц a изменялся в интервале от 100 до 800 с шагом 100. Количество частичных произведений, которые необходимо просуммировать для получения результатного элемента, линейно возрастает с увеличением a, что позволяет оценить эффективность алгоритмов выравнивания порядков и округления. Исходные матрицы были плотно заполнены псевдослучайными 239-битными числами. Результаты эксперимента представлены на рис. 4. Среднее ускорение разработанных алгоритмов составило 2.34 раза по сравнению с MPFR и 5.93 раза по сравнению с NTL.

Четвертый эксперимент. Цель — получение оценок времени высокоточного решения задачи термодинамики. Решалась краевая задача для дифференциального уравнения теплопроводности с однородны-

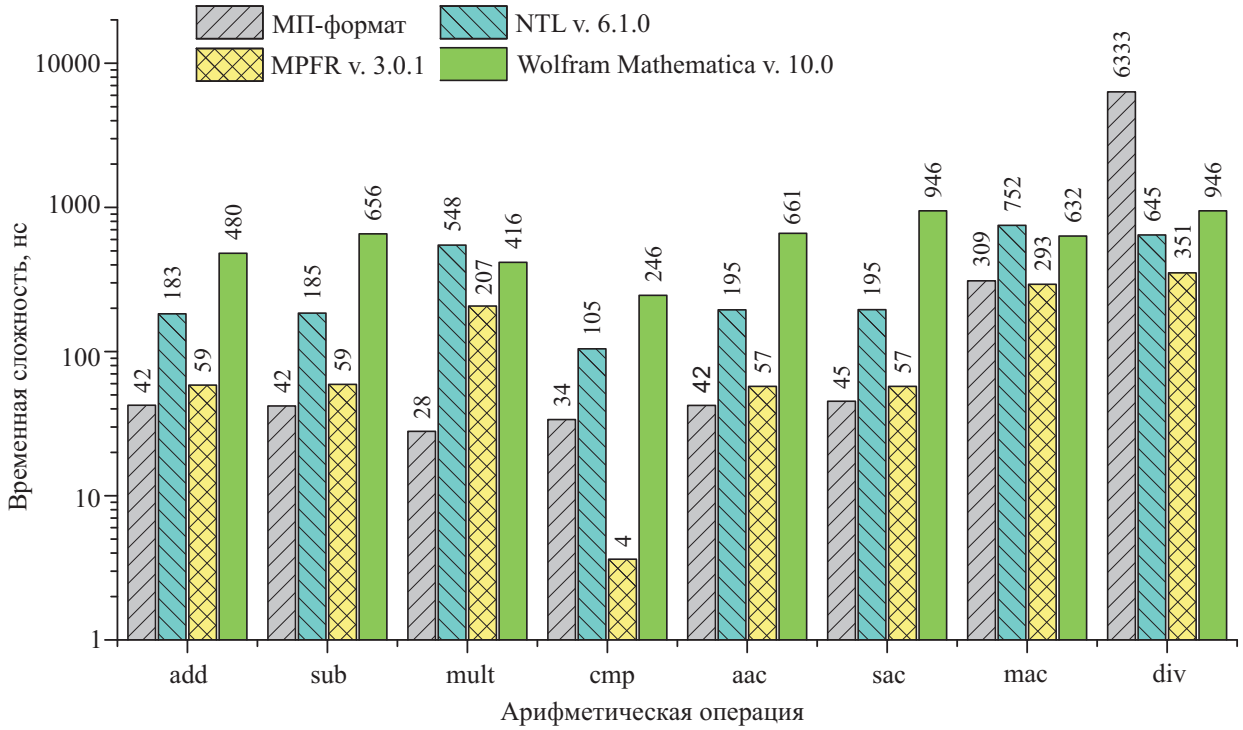

Рис. 3. Сравнительные оценки времени выполнения операций многоразрядной арифметики

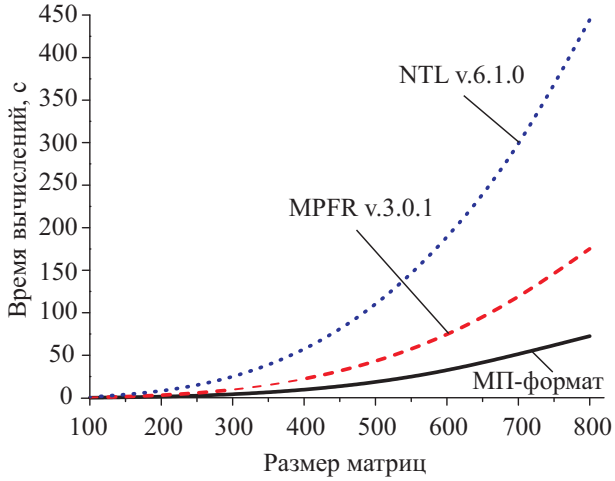

Рис. 4. Результаты исследования времени высокоточного умножения плотных матриц

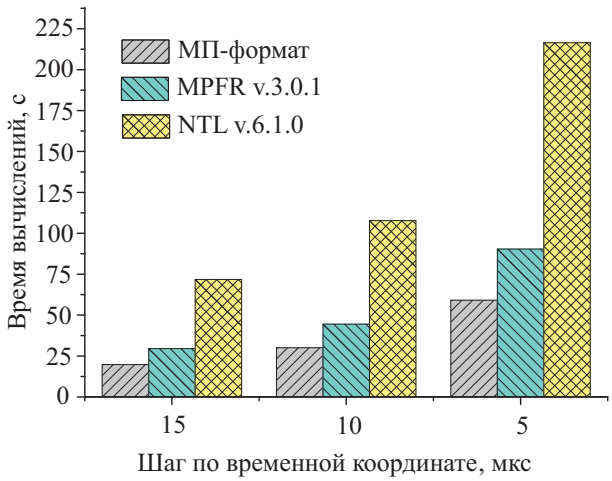

Рис. 5. Результаты исследования времени высокоточного решения уравнения теплопроводности

ми граничными условиями первого рода:

$$\frac{\partial u}{\partial t} = D\,\frac{\partial^2 u}{\partial x^2}\,, \quad u(t,0) = u(t,1) = 0. \tag{5}$$

Уравнение (5) описывает динамику температуры $u(t,x)$ в стержне единичной длины из материала с теплопроводностью D. В начальный момент времени центральная пятая часть стержня была нагрета до температуры 5000 градусов Цельсия. Использовалась четырехточечная явная разностная схема на прямоугольной сетке с равномерным шагом. Коэффициент теплопроводности был принят постоянным ($D = \mathrm{const} = 0.03$). Моделировалось 0.75 секунд физического процесса с постоянным шагом по пространственной координате $h = 0.001$, но с различным шагом по времени: 5 мкс, 10 мкс и 15 мкс. Для всех параметров сетки условия устойчивости схемы выполнялись. Результаты экспериментов представлены на рис. 5. Среднее ускорение разработанных алгоритмов составило 1.51 раза по сравнению с MPFR и 3.63 раза по сравнению с NTL.

5. Заключение. В настоящей статье предложен новый способ представления чисел большой разрядности — модулярно-позиционный формат с плавающей точкой (МП-формат). Для нового формата разработаны кодировки чисел, бесконечностей и нечисловых величин.

Представлены алгоритмы выполнения арифметических операций и округления чисел в МП-формате, в основе которых лежат принципы разрядно-параллельной обработки чисел в системах остаточных классов. Для выполнения сложных немодульных операций использован эффективный метод интервально-позиционных характеристик. В результате проведенных экспериментов получены следующие усредненные по медиане распределения оценки ускорения разработанных алгоритмов: 4.35 раза по сравнению с библиотекой NTL, 1.32 раза по сравнению с библиотекой MPFR и 13.15 раз по сравнению с системой компьютерной алгебры Wolfram Mathematica.

Представленные алгоритмы сохраняют высокое быстродействие как при выполнении одиночных операций, так и при длительных итерационных расчетах, требующих многократного выравнивания порядков, контроля переполнения диапазона и сравнения, что подтверждается проведенными экспериментами: при высокоточном матричном умножении получено среднее ускорение 2.34 раза по сравнению с библиотекой MPFR и 5.93 раза по сравнению с пакетом NTL; при решении задачи теплопроводности среднее ускорение составило 1.51 раза по сравнению с MPFR и 3.63 раза по сравнению с NTL.

На примере использования векторизации циклов вычисления модулярных мантисс и интервально-позиционных характеристик показана эффективность применения параллельной обработки многоразрядных чисел в МП-формате. При выполнении арифметических операций в среднем было получено ускорение 2.38 раза по сравнению с расчетами без использования векторизации.

Разработанный формат и алгоритмы высокоточной арифметики могут использоваться для повышения скорости расчетов, критичных к ошибкам округления, на многоядерных вычислительных системах, графических процессорах и аппаратных ускорителях во многих прикладных областях, в частности при решении плохо обусловленных систем линейных уравнений больших порядков (например, в задачах строительной механики), при вычислении значений функций с использованием рядов Тейлора и в других задачах, требующих выполнения больших суммирований, при вычислении преобразования Фурье и сверток в численно неустойчивых задачах цифровой фильтрации, при вычислении интерполяционных полиномов высоких степеней, при решении жестких систем дифференциальных уравнений, возникающих, например, в задачах химической кинетики и теории ядерных реакторов.

Работа выполнена при поддержке РФФИ (проект № 14–07–31075-мол_а). Статья рекомендована к публикации Программным комитетом Международной суперкомпьютерной конференции “Научный сервис в сети Интернет: многообразие суперкомпьютерных миров” (http://agora.guru.ru/abrau2014).

СПИСОК ЛИТЕРАТУРЫ

1. *Bailey D.H., Borwein J.M.* High-precision arithmetic: progress and challenges. [Electronic resource]: 2013. URL: http://www.davidhbailey.com/dhbpapers/hp-arith.pdf (date of access 19.06.2014).
2. *Ghazi K.R., Lefèvre V., Thevény P., Zimmermann P.* Why and how to use arbitrary precision // Computing in Science & Engineering. 2010. **12**, N 3. 62–65.
3. *Bailey D.H., Borwein J.M.* Experimental mathematics: examples, methods and implications // Notices of the AMS. 2005. **52**, N 5. 502–514.
4. *Bailey D.H., Borwein J.M., Barrio R.* High-precision computation: mathematical physics and dynamics // Applied Mathematics and Computation. 2012. **218**, N 20. 10106–10121.
5. *Muller J.-M. et al.* Handbook of floating-point arithmetic. Boston: Birkhäuser, 2010.
6. *Акушский И.Я., Юдицкий Д.И.* Машинная арифметика в остаточных классах. М.: Сов. Радио, 1968.
7. *Omondi A., Premkumar B.* Residue number systems: theory and implementation. London: Imperial College Press, 2007.
8. *Исупов К.С.* Методика выполнения базовых немодульных операций в модулярной арифметике с применением интервальных позиционных характеристик // Известия высших учебных заведений. Поволжский регион. Технические науки. 2013. **27**, № 3. 26–39.
9. *Исупов К.С.* Об одном алгоритме сравнения чисел в системе остаточных классов // Вестн. Астрахан. гос. техн. ун-та. Сер.: Управление, вычисл. техн. информ. 2014. № 3. 40–49.
10. *Оцоков Ш.А.* Структурно-алгоритмические методы организации высокоточных вычислений на основе теоретических обобщений в модулярной системе счисления. Дис. ... докт. техн. наук. М., 2010.
11. *Sasaki A.* The basis for implementation of additive operations in the residue number system // IEEE Transactions on Computers. 1968. **C-17**, N 11. 1066–1073.
12. *Поснов Н.Н., Буза М.К., Кравцов В.К.* О плавающей запятой в системе счисления в остаточных классах // Вестник Белорусского гос. ун-та. 1969. Серия 1. № 3. 21–27.

13. *Kinoshita E., Kosako H., Kojima Y.* Floating-point arithmetic algorithms in the symmetric residue number system // IEEE Transactions on Computers. 1974. **C-23**, N 1. 9–20.
14. *Chiang J.-S., Lu M.* Floating-point numbers in residue number systems // Computers and Mathematics with Applications. 1991. **22**, N 10. 127–140.
15. *Kinoshita E., Lee K.-J.* A residue arithmetic extension for reliable scientific computation // IEEE Transactions on Computers. 1997. **46**, N 2. 129–138.
16. *Gbolagade K.A., Cotofana S.D.* An $O(n)$ residue number system to mixed radix conversion technique // IEEE International Symposium on Circuits and Systems (24–27 May, 2009). New York : IEEE Press, 2009. 521–524.
17. *Исупов К.С.* Алгоритм вычисления интервально-позиционной характеристики для выполнения немодульных операций в системах остаточных классов // Вестник ЮУрГУ. Серия: Компьютерные технологии, управление, радиоэлектроника. 2014. **14**, № 1. 89–97.
18. *Исупов К.С., Мальцев А.Н.* 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). Киров: Изд-во ВятГУ, 2014. 1179–1184.
19. *Hung C.Y., Parhami B.* An approximate sign detection method for residue numbers and its application to RNS division // Computers and Mathematics with Applications. 1994. **27**, N 4. 23–35.
20. *Kaltofen E., Hitz M.* Integer division in residue number systems // IEEE Transactions on Computers. 1995. **44**, N 8. 983–989.
21. *Lu M., Chiang J.-S.* A novel division algorithm for the residue number system // IEEE Transactions on Computers. 1992. **41**, N 8. 1026–1032.
22. *Chang C.-C., Yang J.-H.* A division algorithm using bisection method in residue number system // International Journal of Computer, Consumer and Control. 2013. **2**, N 1. 59–66.
23. *Fousse L., Hanrot G., Lefèvre V., Pélissier P., Zimmermann P.* MPFR: a multiple-precision binary floating-point library with correct rounding // ACM Transactions on Mathematical Software. 2007. **33**, N 2. Article No. 13.

Поступила в редакцию
9.10.2014

A Parallel-Processing-Oriented Method for the Representation of Multi-Digit Floating-Point Numbers

K. S. Isupov [1] **and A. N. Maltsev** [2]

[1] *Vyatka State University, Faculty of Automation and Computing Machines; ulitsa Moskovskaya 36, Kirov, 610000, Russia; Ph.D., Assistant, e-mail: isupov.k@gmail.com*

[2] *Vyatka State University, Faculty of Automation and Computing Machines; ulitsa Moskovskaya 36, Kirov, 610000, Russia; Postgraduate Student, e-mail: maltsev_ a@list.ru*

Received October 9, 2014

Abstract: The extended precision of calculations is required in solving many scientific and engineering problems. The solution time is a critical parameter to accomplish and, therefore, new methods should be developed for fast high-precision arithmetic. In this paper a new modular-positional format for the representation of floating-point multi-digit numbers is proposed. The main concept of this format is to represent and ensure the digit-parallel processing of floating-point mantissas in residue number systems. The method of interval-positional characteristics is used to increase the speed of complex non-modular operations. Several algorithms for performing arithmetic operations and rounding in the new modular-positional floating-point format are considered. The results of studies of their vectorization efficiency and performance compared to some analogs (MPFR — Multiple Precision Floating-Point Reliable library, NTL — Number Theory Library, and Wolfram Mathematica) are discussed.

Keywords: residue number system, high-precision computations, modular-position floating-point format, multi-digit numbers, arithmetic operations, high performance.

References

1. D. H. Bailey and J. M. Borwein, "High-Precision Arithmetic: Progress and Challenges," http://www.davidhbailey.com/dhbpapers/hp-arith.pdf. Cited June 19, 2014.

2. K. R. Ghazi, V. Lefèvre, P. Théveny, and P. Zimmermann, "Why and How to Use Arbitrary Precision," Comput. Sci. Eng. **12** (3), 62–65 (2010).

3. D. H. Bailey and J. M. Borwein, "Experimental Mathematics: Examples, Methods and Implications," Notices Amer. Math. Soc. **52** (5), 502–514 (2005).

4. D. H. Bailey, J. M. Borwein, and R. Barrio, "High-Precision Computation: Mathematical Physics and Dynamics," Appl. Math. Comput. **218** (20), 10106–10121 (2012).

5. J.-M. Muller, N. Brisebarre, F. de Dinechin, et al., *Handbook of Floating-Point Arithmetic* (Birkhäuser, Boston, 2010).

6. I. Ya. Akushskii and D. I. Yuditskii, *Computer Arithmetic in Residue Classes* (Sov. Radio, Moscow, 1968) [in Russian].

7. A. Omondi and B. Premkumar, *Residue Number Systems: Theory and Implementation* (Imperial College Press, London, 2007).

8. K. S. Isupov, "Methods of Basic Non-Modular Operations in Modular Arithmetic Using Interval Positional Characteristics," Izv. Vyssh. Uchebn. Zaved., Tekh. Nauki **27** (3), 26–39 (2013).

9. K. S. Isupov, "On an Algorithm for Number Comparison in the Residue Number System," Vestn. Astrakhan Tekh. Univ., Ser. Upravl. Vychisl. Tekh. Inform., No. 3, 40–49 (2014).

10. Sh. A. Otsokov, *Structural-Analytic Methods of High-Precision Computing on the Basis of a Modular Number System*, Doctoral Dissertation in Technical Sciences (Moscow Power Eng. Inst., Moscow, 2010).

11. A. Sasaki, "The Basis for Implementation of Additive Operations in the Residue Number System," IEEE Trans. Comput. **C-17** (11), 1066–1073 (1968).

12. N. N. Posnov, M. K. Buza, and V. K. Kravtsov, "On the Floating Point in the Residue Number System," Vestn. Beloruss. Univ., Ser. 1, No. 3, 21–27 (1969).

13. E. Kinoshita, H. Kosako, and Y. Kojima, "Floating-Point Arithmetic Algorithms in the Symmetric Residue Number System," IEEE Trans. Comput. **C-23** (1), 9–20 (1974).

14. J.-S. Chiang and M. Lu, "Floating-Point Numbers in Residue Number Systems," Comput. Math. Appl. **22** (10), 127–140 (1991).

15. E. Kinoshita and K.-J. Lee, " A Residue Arithmetic Extension for Reliable Scientific Computation," IEEE Trans. Comput. **46** (2), 129–138 (1997).

16. K. A. Gbolagade and S. D. Cotofana, "An $O(n)$ Residue Number System to Mixed Radix Conversion Technique," in *Proc. IEEE Int. Symp. on Circuits and Systems, Taipei, Taiwan, May 24–27, 2009* (IEEE Press, New York, 2009), pp 521–524.

17. K. S. Isupov, "Calculation Interval-Positional Characteristic Algorithm for Implementation Nonmodular Operations in Residue Number Systems," Vestn. Yuzhno-Ural. Univ., Ser.: Komp. Tekhnol. Upravl. Radioelektron. **14** (1), 89–97 (2014).

18. K. S. Isupov and A. N. Mal'tsev, "Modular Scaling by the Power of Two with an Arbitrary Step," in *Proc. All-Russian Conf. on Society, Science and Innovations, Kirov, Russia, April 15–26, 2014* (Vyatka Gos. Univ., Kirov, 2014), pp. 1179–1184.

19. C. Y. Hung and B. Parhami, "An Approximate Sign Detection Method for Residue Numbers and its Application to RNS Division," Comput. Math. Appl., **27** (4), 23–35 (1994).

20. E. Kaltofen and M. A. Hitz, "Integer Division in Residue Number Systems," IEEE Trans. Comput., **44** (8), 983–989 (1995).

21. M. Lu and J.-S. Chiang, "A Novel Division Algorithm for the Residue Number System," IEEE Trans. Comput., **41** (8), 1026–1032 (1992).

22. C.-C. Chang and J.-H. Yang, "A Division Algorithm Using Bisection Method in Residue Number System," Int. J. Comput. Consum. Control **2** (1), 59–66 (2013).

23. L. Fousse, G. Hanrot, V. Lefèvre, et al., "MPFR: A Multiple-Precision Binary Floating-Point Library With Correct Rounding," ACM Trans. Math. Softw. **33** (2) (2007). doi acm.org/10.1145/1236463.1236468