УДК 519.688

Использование модулярной арифметики для ускорения выполнения операций над числами большой разрядности

М. А. Дерябин[1], **А. А. Зайцев**[2]

[1] maxim.deryabin@gmail.com, [2] collard.90@gmail.com

ФГАОУ ВПО «Северо-Кавказский федеральный университет» (СКФУ)

Поступила в редакцию 22.01.2013

Аннотация. Описывается один из методов ускорения вычислений над числами большой разрядности, основанный на применении системы остаточных классов. Рассмотрены современные методы и алгоритмы модулярной арифметики. Приведен пример использования модулярной арифметики для ускорения работы алгоритма RSA. Представлены результаты тестирования разработанных на основе исследования программ.

Ключевые слова: параллельные вычисления; модулярная арифметика; система остаточных классов.

Решение современных задач науки и техники влечет за собой необходимость работы с большими объемами данных. Например, чем выше вычислительная мощность, тем проще злоумышленнику взламывать различные криптографические алгоритмы. Для повышения их криптостойкости необходимо увеличивать длину ключей и тем самым – величину операндов математических преобразований. Это приводит к замедлению выполнения операций, лежащих в основе рассматриваемых алгоритмов, что отрицательно сказывается на скорости работы для легальных пользователей. Возникает необходимость поиска путей ускорения выполняемых преобразований. Часто для этого используется декомпозиция и распараллеливание прямых последовательных алгоритмов выполнения операций. Но в некоторых случаях стандартные способы распараллеливания вычислений малоэффективны.

Одним из перспективных путей решения задач сокращения времени обработки данных и повышения надежности вычислительных средств является применение различных форм параллельной обработки данных на основе числовых систем с параллельной структурой. Среди современных подходов к созданию высокопроизводительных средств обработки данных можно отметить использование системы остаточных классов (СОК).

1. ОСНОВЫ МОДУЛЯРНОЙ АРИФМЕТИКИ

Система остаточных классов (Residue number system) является непозиционной системой представления чисел [1]. Пусть задана некоторая система взаимно-простых модулей $\{p_1, p_2, \ldots, p_n\}$. Число A в СОК по данным модулям представляется в виде кортежа чисел $(a_1, a_2, \ldots, a_n)$, где $a_i = A \bmod p_i$, $b_i = B \bmod p_i$ для $i = 1, 2, \ldots, n$. В соответствии с Китайской теоремой об остатках, такое представление числа A является единственным, если $0 \le A < P = p_1 p_2 \ldots p_n$, где P называется диапазоном СОК. При этом операции $C = A + B \bmod P$ и $D = A \cdot B \bmod P$ для чисел $A = (a_1, a_2, \ldots, a_n)$ и $B = (b_1, b_2, \ldots, b_n)$, представленных в СОК, определяются следующим образом:

$$C = A + B \bmod P = (a_1 + b_1 \bmod p_1, a_2 + b_2 \bmod p_2, \ldots, a_n + b_n \bmod p_n),$$

$$D = A \cdot B \bmod P = (a_1 b_1 \bmod p_1. a_2 b_2 \bmod p_2, \ldots, a_n b_n \bmod p_n).$$

Описанное представление эффективно использовать при выполнении операций умножения и сложения, так как числа a_i и b_i имеют гораздо меньшее число разрядов, чем исходные числа A и B. При этом обработка данных происходит по параллельным каналам связи. Таким образом, модулярная арифметика позволяет проводить декомпозицию системы большого

динамического диапазона на ряд параллельных независимых каналов меньшей разрядности. Использование такого подхода увеличивает эффективность вычислений.

Ввиду малости остатков СОК эффективно использовать табличные методы реализации. В ряде случаев бывает удобно внести результат операции в таблицу, строки и столбцы в которой определяются операндами. В таких случаях операция поиска по таблице должна быть эффективнее непосредственно арифметических операций в данной архитектуре.

СОК находит применение во многих теоретико-числовых системах. Например, специальным образом преобразованный код СОК можно использовать для организации помехоустойчивых вычислений. Похожие принципы применяются в криптографической схеме разделения секрета. Некоторые свойства кода СОК полезны при цифровой обработке сигналов и при проектировании пороговых схем. Кроме того, модель данных СОК все чаще используется при построении относительно нового вида структур, сочетающих в себе модели нейронных сетей и модулярной арифметики, – модулярных нейрокомпьютеров.

2. МОДУЛЬНЫЕ И НЕМОДУЛЬНЫЕ ОПЕРАЦИИ В СОК

Операции, выполняемые над данными, представленными в системе остаточных классов, принято разделять на два класса: модульные и немодульные. Модульные характеризуются тем, что при их выполнении не происходит переносов между разрядами. Это позволяет осуществлять такие операции параллельно относительно каждого разряда, что увеличивает быстродействие всего алгоритма в целом. Примерами таких операций служат элементарные сложение и умножение. Однако деление в СОК имеет иной характер. Его можно отнести к немодульным операциям, имеющим позиционную природу.

Основная сложность выполнения немодульных операций заключается в том, что СОК есть непозиционная система. Здесь операция сравнения приравнивается по сложности к переводу в непозиционную систему счисления. Так, если число A делится нацело на число B, то деление можно реализовать достаточно легко. Однако если заранее неизвестно, разделимо ли число A без остатка на B, то стратегия поиска частного и остатка резко усложняется. Для преодоления сложившихся сложностей можно пользоваться различными точными и приближенными методами [2, 3].

Одной из наиболее часто используемых операций при работе с большими целыми числами является модульное возведение в степень. Алгоритмы, реализующие данную операцию, используются в различных областях, одним из примеров которых является шифрование и расшифрование передаваемой информации в схеме RSA.

Модульное возведение в степень представляет собой последовательное повторение операций умножения. Так как операция умножения выполняется по каждому модулю параллельно, то возведение в степень по модулю P можно представить как n независимых возведений в степень по модулям $p_1, p_2, \ldots, p_n$. Таким образом, возведение в степень – модульная операция.

Рассмотрим один из алгоритмов быстрого модульного возведения в степень. Пусть требуется вычислить $z = a^s \bmod p$. Число s может быть записано как

$$s = \sum_{i=0}^{h-1} s_i 2^i. \qquad (1)$$

Положим старший разряд $s_{n-1} = 1$ по определению. Числа s_i могут принимать значения 0 и 1. Число h является длиной бинарного представления числа s. Из выражения (1) следует следующая формула для возведения в степень:

$$z = a^{\left(\sum_{i=0}^{h} s_i 2^i\right)} \bmod p = \prod_{i=0}^{h-1} \left(a^{2^i}\right)^{s_i} \bmod p. \qquad (2)$$

Формула (2) лежит в основе алгоритма быстрого модульного возведения в степень [4].

Еще один способ ускорения операции возведения в степень в СОК заключается в сокращении показателя степени. Такая возможность следует из Малой теоремы Ферма. При этом требуется, чтобы все модули системы являлись простыми числами. В Малой теореме Ферма говорится, что для любого числа, не делящегося на простое число, верно равенство $a^{p-1} \equiv 1 \bmod p$.

Данная теорема является очень полезной для расчета мультипликативной обратной величины для целого числа a, потому что $a^{p-2} \equiv a^{-1} \bmod p$. Но более интересным для нас является следствие из данной теоремы: если целое число a не делится на p и если $n \equiv m \bmod (p-1)$, то $a^n \equiv a^m \bmod p$. В следствии говорится, что при работе по простому модулю p показатели могут быть снижены до $\bmod (p-1)$.

Принимая во внимание вышеизложенные факты, выражение $Z = A^B \bmod P$, где $P = p_1, p_2, \ldots, p_n$ и p_i – простые числа ($i = 1, 2, ..., n$) можно заменить вычислениями в СОК по модулям $p_1, p_2, \ldots, p_n$:

$$Z = (a_1^{B_1} \bmod p_1, a_2^{B_2} \bmod p_2, \ldots, a_n^{B_n} \bmod p_n),$$

где $B_i = B \bmod p_i - 1$.

3. ПЕРЕВОД ЧИСЕЛ ИЗ СОК В ПОЗИЦИОННОЕ ПРЕДСТАВЛЕНИЕ И ОБРАТНО

Особенностью адаптации алгоритмов для СОК является необходимость перевода данных из позиционной системы в СОК и обратно. Прямые преобразования из позиционной системы и обратно в случае обработки больших целых чисел являются непараллельными, что увеличивает сложность реализации преобразуемого алгоритма. В [5] предлагается эффективный метод преобразования двоичного числа в СОК на основе разбиения исходного двоичного числа на отдельные форматы, для которых отводится заранее известное количество двоичных разрядов B. Тогда n-битное двоичное число может быть выражено как комбинация n / B взвешенных (позиционных) форматов размерностью B бит (разрядов). При этом позициям каждого формата n / B присваивается определенный вес 2^j, где $j = 0, B, 2B, \ldots, MB$.

Прямое преобразование двоичного числа в модулярное осуществляется с помощью модульного суммирования остатков по модулю p_i ($i = 1, 2, \ldots, n$) для B разрядов и n / B форматов с учетом их весов.

На основании сказанного любое двоичное число может быть записано в виде

$$X = \sum_{j=0}^{M} \left(\sum_{i=0}^{B-1} x_{jB+i} 2^i \right) 2^{jB},$$

где B – количество разрядов выбранного формата; M – степень формата; x_i – коэффициент 0 или 1; $j = 0, B, 2B, \ldots, MB$ – позиция формата; i – позиция разряда в формате. Основываясь на этом выражении, можно записать формулу для вычисления остатка по модулю p:

$$|X|_p = \left| \sum_{j=0}^{M} \left(\sum_{i=0}^{B-1} x_{jB+i} 2^i \right) 2^{jB} \right|_p =$$

$$= \left| \sum_{j=0}^{M} \left(\left| \sum_{i=0}^{B-1} x_{jB+i} 2^i \right|_p \cdot \left| 2^{jB} \right|_p \right) \right|_p .$$

При выполнении операций следует учитывать, что $\omega_i = \left| 2^{jB} \right|_p$ есть заранее вычисленные константы. Нахождение остатка от деления для каждого формата можно находить независимо относительно остальных форматов. После чего вычисленные остатки по каждому из форматов складываются по модулю p. При таком подходе к нахождению остатков операции проводятся над числами гораздо меньшей разрядности.

Пример. Пусть дано число X = 2460034527 и модуль p = 7.

Для решения представим X в двоичной системе:

X = 1001 0010 1010 0001 0010 0101 1101 1111.

Разобьем двоичное представление числа X на 4 формата по B = 8 бит в каждом. Для дальнейших вычислений, найдем константы ω:

$$\omega_0 = \left| 2^0 \right|_7 = 1; \ \omega_1 = \left| 2^8 \right|_7 = 4;$$

$$\omega_2 = \left| 2^{16} \right|_7 = 2; \ \omega_3 = \left| 2^{24} \right|_7 = 1.$$

Дальнейшие вычисления представлены в виде следующей схемы (рис. 1).

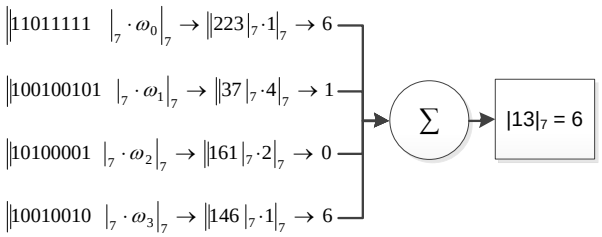

Рис. 1. Пример параллельного вычисления вычета по модулю

Согласно данным вычислениям, $|X|_7 = 6$, что соответствует действительности.

Из рассмотренного примера видно, что данный способ вычисления остатка от деления хорошо подходит для работы с числами большой разрядности. В данном случае за счет параллельной обработки данных в несколько раз сокращается разрядность операндов математических преобразований. Такой подход позволяет ускорить вычислительные операции и, кроме того, упрощает представление чисел большой разрядности за счет использования более коротко разрядной сетки.

Обратное преобразование числа из модулярного представления в двоичную форму базируется на классической теореме из теории чисел, которая называется Китайской теоремой об остатках (КТО). На основании известного пред-

ставления чисел в СОК (α_1, α_2, …, α_n) КТО делает возможным определение числа в позиционную систему счисления (ПСС) $|X|_p$, если наибольший общий делитель любой пары модулей равен 1.

Китайская теорема об остатках имеет вид

$$|X|_p = \left| \sum_{i=1}^{n} P_i \left| \frac{\alpha_i}{p_i} \right|_{p_i} \right|_P, \qquad (3)$$

где $P_i = \frac{P}{p_i}$, $P = \prod_{i=1}^{n} p_i$ для $(p_i, p_j) = 1$ для $i \neq j$. Такая форма КТО предполагает лишь последовательные вычисления по модулю P, который является достаточно большим. Снизить разрядность модуля можно с помощью применения позиционной системы со смешанными основаниями. Под обобщенной позиционной системой счисления (ОПСС) понимается такая система, в которой целое число N представляется в виде

$$N = a_{n-1}\pi_{n-1}\pi_{n-2} \ldots \pi_2\pi_1 + a_{n-2}\pi_{n-2}\pi_{n-3} \ldots \pi_2\pi_1 + \ldots + a_2\pi_2\pi_1 + a_1\pi_1 + a_0,$$

где a_j – цифры 0, 1, …, π_{j-1} (j = 1, 2, …, n). Отображение из СОК в ОПСС может быть определено рекурсивно с помощью операций по малым модулям p_i. Для перехода от вычислений по модулю P к вычислению по модулям p_i предлагается метод восстановления чисел на основе совместного использования КТО и обобщенной позиционной системы счисления (ОПСС) [5].

Цифры ОПСС можно искать, например, пользуясь формулой: $a_i = \left\| \left[N \Big/ \prod_{j=0}^{i} \pi_j \right] \right\|_{\pi_{i+1}}$, для i = 0, 1, …, n–1 и положив π_0 = 1. Так, для N = 758 и системы модулей {2, 5, 7, 11} ОПСС представления будет выглядеть следующим образом: N = [0, 4, 5, 10]. В случае если имеется представление числа N в СОК по модулям $\{\pi_i\}$ $i = \overline{1,n}$, то перевод в ОПСС можно проводить более оптимизированным способом, представленным в [6].

Пусть задана система оснований p_1, p_2,…, p_n с диапазоном $P = p_1p_2\ldots p_n$ и ортогональными базисами B_1, B_2, …, B_n, которые определяются как

$$B_i = \frac{m_i P}{p_i} \equiv 1 \bmod p_i,\ i = \overline{1,n},$$

где m_i – веса ортогональных базисов. Представим ортогональные базисы B_i в ОПСС, тогда

$$B_i = b_{i1} + b_{i2}p_1 + b_{i2}p_1p_2 + \ldots + b_{in}p_1p_2\ldots p_n, \qquad (4)$$

где b_{ij} – коэффициенты ОПСС, i, j = 1, 2, …, n. Из (4) и (3) следует $X_{\text{ОПСС}} = a_1[b_{11}, b_{12}, \ldots, b_{1n}] + a_2[0, b_{22}, \ldots, b_{2n}] + \ldots + a_n[0, 0, \ldots, b_{nn}]$.

Так как $B_i \bmod p_i = 0$, $\forall j > i$, то перед первым значащим разрядом будет i – 1 нулей. Для удобства вычислений базисы можно представить в виде матрицы. Тогда $X_{\text{ОПСС}}$ можно вычислить следующим образом: $\tau_i = \sum_{j=1}^{n} \alpha_i b_{ij}$ для i = 1, 2, …, n, $a_1 = \tau_1 \bmod p_1$,

$$a_i = \left(\tau_1 + \left[\frac{\tau_{i-1}}{p_{i-1}} \right] \right) \bmod p_i \text{ для } i = 2, 3, \ldots, n,$$

где a_i – коэффициенты ОПСС числа X; α_i – вычеты числа X по mod p_i; b_{ij} – ортогональные базисы, представленные в ОПСС. Получение позиционного представления числа осуществляется по формуле

$$X = a_n p_1p_2 \cdot \ldots \cdot p_{n-1} + a_{n-1}p_1p_2 \cdot \ldots \cdot p_{n-2} + \ldots + a_2 p_1 + a_1 = a_1 + \sum_{k=1}^{n-1} q_k a_{k+1},$$

где $q_k = \prod_{i=1}^{k} p_i$, для k = 1, 2, …, n–1 являются константами.

Пример. Пусть дано число X = (0, 3, 2, 10) в системе классов с основаниями {2, 5, 7, 11}. Требуется найти позиционное представление числа X. В данном случае ортогональные базисы равны B_1 = (1, 0, 0, 0) = 358; B_2 = (0, 1, 0, 0) = = 616, B_3 = (0, 0, 1, 0) = 330, B_4 = (0, 0, 0, 1) = = 210. В обобщенной позиционной системе:

$$B1 = [b_{11}, b_{12}, b_{13}, b_{14}] = [1, 2, 3, 5],$$
$$B2 = [b_{21}, b_{22}, b_{23}, b_{24}] = [0, 3, 5, 8],$$
$$B3 = [b_{31}, b_{32}, b_{33}, b_{34}] = [0, 0, 5, 4],$$
$$B4 = [b_{41}, b_{42}, b_{43}, b_{44}] = [0, 0, 0, 3].$$

Умножим каждую строку матрицы (b_{ij}) на соответствующие цифры числа X:

$$X \to (b_{ij}) \Rightarrow \begin{pmatrix} 0 \\ 3 \\ 2 \\ 10 \end{pmatrix} \times \begin{pmatrix} 1 & 2 & 3 & 5 \\ 0 & 3 & 5 & 8 \\ 0 & 0 & 5 & 4 \\ 0 & 0 & 0 & 3 \end{pmatrix} = \begin{pmatrix} 0 & 0 & 0 & 0 \\ 0 & 9 & 15 & 24 \\ 0 & 0 & 10 & 8 \\ 0 & 0 & 0 & 30 \end{pmatrix}.$$

После этого сложим числа в столбцах и, учитывая переносы разрядов и отбросив избыточные, получим представление X в ОПСС. Получаем, что X = [0, 4, 5, 10], откуда X = 0 + 4·2 + 5·2·5 + + 10·2·5·7 = 758. Таким образом найден ответ:

X = 758. Рассмотренный пример демонстрирует основные преимущества данного алгоритма: частичная параллельность и работа с небольшими модулями.

Следует отметить: использование усовершенствованных алгоритмов перевода в СОК и обратно эффективно лишь в определенных условиях. Например, для нахождения остатка от деления числа с использованием разбиения на форматы необходимо, чтобы число форматов (т. е. параллельных потоков) было достаточно большим, чтобы уменьшить время выполнения элементарных операций компенсировало увеличение их количества.

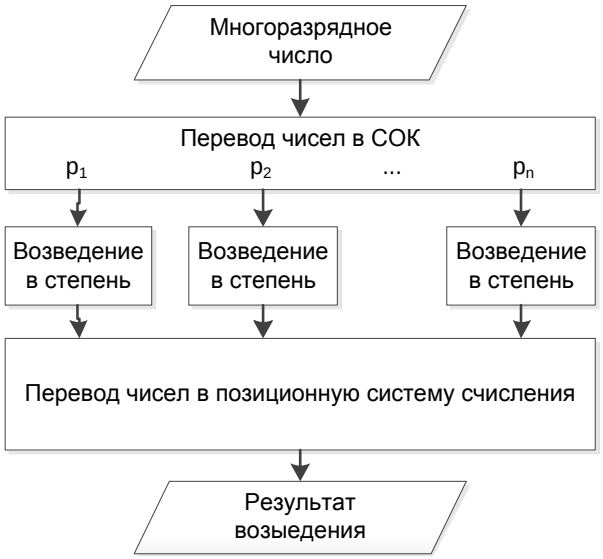

Рис. 2. Алгоритм возведения в степень с применением СОК

На основе рассмотренных выше алгоритмов было разработано две программы. Первая реализует алгоритм быстрого модульного возведения в степень над числами различной разрядности. Вторая реализует тот же алгоритм, но с применением СОК, схема которого приведена на рис. 2. На рис. 3 приведена сравнительная диаграмма ускорения, получаемого при использовании СОК на основе тестовых данных разработанных программ для системы оснований, состоящей из двух, трех и четырех модулей по 100, 150 и 200 бит каждый. Из диаграммы видно, что при увеличении числа задействованных оснований и их разрядности ускорение резко возрастает. Причем оно превосходит количество используемых параллельных каналов.

Например, при использовании двух оснований ускорение в среднем составляет примерно 3,6 раз, при использовании трех оснований оно уже превышает 4 раза. Таким образом, если даже последовательный алгоритм выполняется параллельно над независимыми данными, использование СОК все равно приведет к значительному ускорению.

Однако использование СОК целесообразно только при соблюдении ряда условий. Например, если общий модуль системы есть простое число или его разложение на простые множители неизвестно, то применение СОК невозможно.

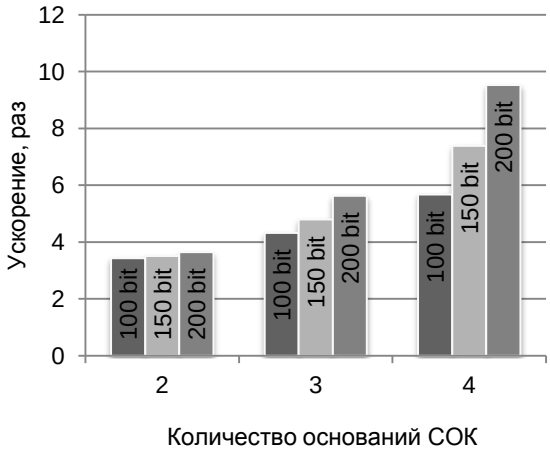

Рис. 3. Среднее ускорение операции умножения с использованием СОК относительно последовательной реализации

4. ПРИМЕНЕНИЕ СОК ДЛЯ УСКОРЕНИЯ АЛГОРИТМА РАСШИФРОВАНИЯ В СХЕМЕ RSA

Модульные операции с большими целыми числами требуются во многих алгоритмах, классическим примером которых является схема шифрования RSA. Основной отличительной чертой RSA является тот факт, что модуль, используемый в схеме шифрования RSA, представляет собой произведение двух простых чисел. Это позволяет использовать двухмодульную СОК для ускорения операций с закрытым ключом.

Для простых P и Q любое сообщение $M < N$, где $N = PQ$, единственным образом представляется парой $[M_P, M_Q]$, где $M_P = M \bmod P$ и $M_Q = M \bmod Q$. Таким образом можно получить M по вычислениям с M_P, M_Q и с их последующей «сборкой», а не обычным возведением $M = C^D \bmod N$. С помощью следствия Малой теоремы Ферма размер показателя может быть уменьшен:

$$M_P = M \bmod P = \left(C^D \bmod N\right) \bmod P =$$
$$= C^D \bmod P = C^{D \bmod (P-1)} \bmod P = C^{D_P} \bmod P.$$

Со значениями $C_P = C \bmod P$ и $C_Q = C \bmod Q$, а также $D_P = D \bmod (P - 1)$ и $D_Q = D \bmod (Q - 1)$, мы получаем следующие выражения для M_P и M_Q: $M_P = C_P^{D_P} \bmod P$ и $M_Q = C_Q^{D_Q} \bmod Q$.

Если предположить, что показатели $D_P = D \bmod (P - 1)$ и $D_Q = D \bmod (Q - 1)$ были предварительно вычислены, RSA расшифрование, основанное на использовании СОК, может иметь место в соответствии со следующими шагами:

1. Вычислить $C_P = C \bmod P$ и $C_Q = C \bmod Q$.

2. Возвести в степень $M_P = C_P^{D_P} \bmod P$ и $M_Q = C_Q^{D_Q} \bmod Q$.

3. Перевести полученные результаты из СОК в ПСС.

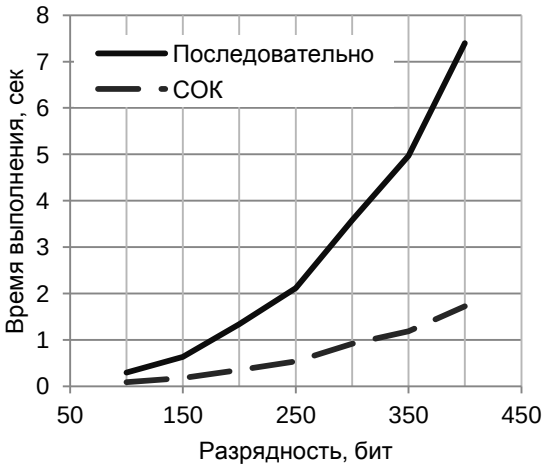

Рис. 4. Время выполнения обычного алгоритма RSA и RSA с применением СОК

Два возведения в степень (шаг 2) можно вычислить независимо друг от друга и параллельно. На рис. 4 представлены результаты тестов предложенного алгоритма и его последовательной версии. При этом достигается ускорение, в среднем равное четырем.

ВЫВОД

Современные достижения модулярной арифметики делают ее крайне эффективным средством для ускорения алгоритмов, использующих целые числа большой разрядности. При увеличении числа модулей ускорение выполнения операций возрастает, что ставит этот способ представления чисел в ряду наиболее перспективных методов в данной области.

СПИСОК ЛИТЕРАТУРЫ

1. **Omondi A., Premkumar B.** Residue number systems. Theory and Implementation. London: Imperial College Press, 2007.

2. **Червяков Н. И., Лавриненко И. Н, Лавриненко С. В., Мезенцева О. С.** Методы и алгоритмы округления, масштабирования и деления чисел в модулярной арифметике. // 50 лет модулярной арифметике : матер. Междунар. науч.-техн. конф. Зеленоград: МИЭТ, 2005. С. 291–310.

3. **Червяков Н. И.** Методы, алгоритмы и техническая реализация основных проблемных операций, выполняемых в системе остаточных классов // Инфокоммуникационные технологии. Самара: ПГУТиИ, 2011. № 4. С. 4–12.

4. **Schneier B.** Applied Cryptography: Protocols, Algorithms, and Source Code in C. 2nd Ed. New York: John Wiley & Sons, 1995. 662 p.

5. **Червяков Н. И.** Реализация высокоэффективной модулярной цифровой обработки сигналов на основе программируемых логических интегральных схем // Нейрокомпьютеры: разработка и применение. 2006. № 10. С. 24–36.

6. **Червяков Н. И., Сахнюк П. А., Шапошников А. В., Макоха А. Н.** Нейрокомпьютеры в остаточных классах. М.: Радиотехника, 2003. 272 с.

ОБ АВТОРАХ

ДЕРЯБИН Максим Анатольевич, асп. каф. высш. алгебры и геометрии Ин-та математики и естеств. наук. Дипл. магистр математики (СКФУ, 2013). Иссл. в обл. модулярной арифметики, параллельных вычислений.

ЗАЙЦЕВ Александр Александрович, асп. той же каф. Дипл. магистр математики (СКФУ, 2013) . Иссл. в обл. модулярной арифметики, параллельных вычислений.

METADATA

Title: Using of the modular arithmetic for acceleration of execution operations on high numbers.

Authors: M. A. Deryabin, A. A. Zaytsev.

Affiliation: North-Caucasian Federal University (NCFU), Russia.

Email: maxim.deryabin@gmail.com, collard.90@gmail.com.

Language: Russian.

Source: Vestnik UGATU (scientific journal of Ufa State Aviation Technical University), vol. 17, no. 5 (58), pp. 245-251, 2013. ISSN 2225-2789 (Online), ISSN 1992-6502 (Print).

Abstract: The paper describes one of the methods to accelerate calculations on numbers of larger capacity, based on the application of the residue number system. Modern methods and algorithms for modular arithmetic are discussed. In paper given an example of using modular arithmetic to speed up the algorithm RSA. Presented the

results of the tests of programs developed through research.

Key words: parallel computing; modular arithmetic; residue number system.

References (English transliteration):

1. A. Omondi and B. Premkumar, *Residue Number Systems. Theory and Implementation.* London: Imperial College Press, 2007.
2. N. I. Chervyakov, I. N. Lavrynenko, S. V. Lavrynenko, and O. S. Mesentseva, "Methods and algorithms to rounding, scaling, and divide numbers in modular arithmetic," (in Russian), in *Proc. Int. Sci. Conf. "50 years of modular arithmetic,"* MIET, Zelenograd, Moscow region, 2005, pp. 291-310.
3. N. I. Chervyakov, "Methods, algorithms and technical implementation of the basic problem of operations performed in the remaining classes," (in Russian), *Infocommunikatsionnye Tekhnologii*, no. 4, pp. 4-12, 2011.
4. B. Schneier, *Applied Cryptography: Protocols, Algorithms, and Source Code in C*, 2nd Ed. New York: John Wiley & Sons, 1995.
5. N. I. Chervyakov, "Modular implementation of high-performance digital signal processor-based programmable logic integrated circuits," *Neyrokomputery: Razrabotka i Primenenie*, no. 10, pp. 24-36, 2006.
6. N. I. Chervyakov, P. A. Cakhniuk, A. V. Shaposhnikov, and A. N. Makokha, *Neurocomputers in Residual Classes*, (in Russian). Moscow: Radiotekhnika, 2003.

About authors:

DERYABIN, Maxim Anatolievich, postgraduate student of Department of Algebra and geometry of the Institute of Mathematics and Natural Sciences, a certified master of mathematics (NCFU, 2013).

ZAYTSEV, Alexander Alexandrovich, postgraduate student of Department of Algebra and geometry of the Institute of Mathematics and Natural Sciences, a certified master of mathematics (NCFU, 2013).