ИСПОЛЬЗОВАНИЕ МОДУЛЯРНОЙ АРИФМЕТИКИ ДЛЯ УСКОРЕНИЯ ВЫПОЛНЕНИЯ ОПЕРАЦИЙ НАД ЧИСЛАМИ БОЛЬШОЙ РАЗРЯДНОСТИ

М.А. Дерябин, А.А. Зайцев

Северо-Кавказский федеральный университет, Ставрополь

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.

Введение

Совершенствование вычислительной техники влечет за собой необходимость работы с большими объемами данных, с числами большой разрядности. Это приводит к замедлению операций, лежащих в основе многих алгоритмов. Возникает необходимость поиска путей ускорения преобразований. Часто для этого используется декомпозиция и распараллеливание последовательных алгоритмов выполнения операций. Но в некоторых случаях стандартные способы распараллеливания вычислений малоэффективны. Одним из перспективных путей решения задачи сокращения времени обработки данных является применение различных форм параллельной обработки данных на основе числовых систем с параллельной структурой. Современным подходом к созданию отказоустойчивых высокопроизводительных средств обработки данных является использование системы остаточных классов (СОК).

1. Основы модулярной арифметики. Модульное возведение в степень

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 $a_i = A(\mathrm{mod}\, p_i)$ 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 $C = A + B(\mathrm{mod}\, P)$ и $D = A \cdot B(\mathrm{mod}\, P)$ 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:

$c_i = a_i + b_i(\mathrm{mod}\, p_i),\ d_i = a_i \cdot b_i(\mathrm{mod}\, p_i),\ i = 1,2,...,n.$

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, так как в таком представлении нет межразрядных связей, эти операции можно выполнять параллельно по разным модулям. Таким образом, модулярная арифметика позволяет проводить декомпозицию системы большого динамического диапазона на ряд параллельных независимых каналов меньшей разрядности. Использование такого подхода увеличивает эффективность вычислений.

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

Рассмотрим один из алгоритмов быстрого модульного возведения в степень. Пусть требуется вычислить $z = a^s (\mathrm{mod}\ p)$. Число s может быть записано как

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

Положим старший разряд $s_{h-1} = 1$ 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:

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

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

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 a, не делящегося на простое число, верно равенство $a^{p-1} \equiv 1 (\mathrm{mod}\ p)$.

Интересным для нас является следствие из данной теоремы: если целое число a не делится на p и если $n \equiv m (\mathrm{mod}(p-1))$, то $a^n \equiv a^m (\mathrm{mod}\ p)$. В следствии говорится, что при работе по простому модулю p показатели могут быть снижены до $\mathrm{mod}(p-1)$.

Принимая во внимание вышеизложенные факты, выражение $Z = A^B (\mathrm{mod}\ P)$, где $P = p_1 p_2 \cdot \ldots \cdot p_n$ и p_i – простые числа $(i = 1,2,\ldots,n)$, можно заменить вычислениями в СОК по модулям $p_1, p_2, \ldots, p_n$: $Z = (a_1^{B_1} (\mathrm{mod}\ p_1), a_2^{B_2} (\mathrm{mod}\ p_2), \ldots, a_n^{B_n} (\mathrm{mod}\ p_n))$, где $B_i = B(\mathrm{mod}(p_i - 1))$.

2. 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 [3] 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. 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. Остаток 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.

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 представления чисел в СОК $(\alpha_1, \alpha_2, ..., \alpha_n)$ 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 $(p_i, p_j) = 1$, $i \neq j$. 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

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

где a_j – цифры $0,1,...\pi_{j-1}$ $(j = 1,2,...n)$. 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) [3].

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

$$B_i = \frac{m_i P}{p_i} \equiv 1 (\text{mod } p_i), \quad i = \overline{1, n},$$

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

$$B_i = b_{i1} + b_{i2} p_1 + b_{i2} p_1 p_2 + ... + b_{in} p_1 p_2 ... p_n , \qquad (4)$$

где b_{ij} – коэффициенты ОПСС, $i,\ j = 1,2,...,n.$ Из (4) и (3) следует

$$X_{ОПСС} = a_1 (b_{11}, b_{12}, ..., b_{1n}) + a_2 (0, b_{22}, ..., b_{2n}) + ... + a_n (0,0,..., b_{nn}).$$

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 $X_{ОПСС}$ можно вычислить следующим образом

$\tau_i = \sum_{j=1}^{n} \alpha_i b_{ij}$ для $i = 1,2,...,n$, $a_1 = \tau_1 (\mathrm{mod}\, p_1)$, $a_i = \left(\tau_1 + \left[\frac{\tau_{i-1}}{p_{i-1}} \right] \right) (\mathrm{mod}\, p_i)$ для $i = 2,3,...,n$, где a_i – коэффициенты ОПСС числа x; α_i – вычеты числа x по $\mathrm{mod}\, p_i$; b_{ij} – ортогональные базисы, представленные в ОПСС. Получение позиционного представления числа осуществляется по формуле

$$X = a_n p_1 p_2 \cdot ... \cdot p_{n-1} + a_{n-1} p_1 p_2 \cdot ... \cdot p_{n-2} + ... + a_2 p_1 + a_1 = a_1 + \sum_{k=1}^{n-1} q_k a_{k+1},$$

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

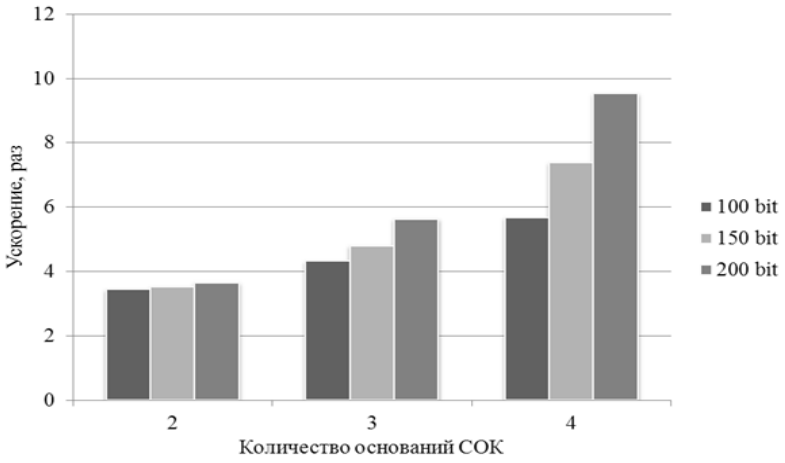

Рис. 1. Среднее ускорение тестовых программ, использующих СОК, относительно последовательных программ

На основе рассмотренных выше алгоритмов было разработано две программы. Первая реализует рассмотренный выше алгоритм модульного возведения в степень над числами различной разрядности. Вторая реализует тот же алгоритм, но с применением СОК. На рис. 1 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 потоков.

3. Ускорение алгоритма расшифрования в схеме RSA

Модульные операции с большими целыми числами требуются во многих алгоритмах, классическим примером 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 $M < N = PQ$ единственным образом представляется парой $[M_P, M_Q]$, где $M_P = M (\mathrm{mod}\, P)$ и $M_Q = M (\mathrm{mod}\, Q)$. Таким образом можно получить M по вычислениям с M_P, M_Q и с их последующей «сборкой», а не обычным возведением $M = C^D (\mathrm{mod}\, N)$. С помощью следствия Малой теоремы Ферма размер показателя может быть уменьшен. Со значениями $C_P = C (\mathrm{mod}\, P)$ и

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

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

Вычислить $C_P = C(\mathrm{mod}\, P)$ и $C_Q = C(\mathrm{mod}\, Q)$.

Возвести в степень $M_P = C_P^{D_P}(\mathrm{mod}\, P)$ и $M_Q = C_Q^{D_Q}(\mathrm{mod}\, Q)$.

Перевести полученные результаты из СОК в ПСС.

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

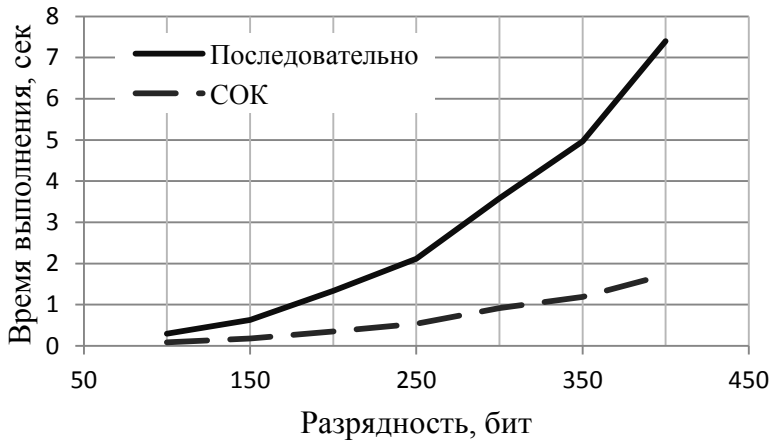

Рис. 2. Скорость выполнения обычного алгоритма RSA и RSA с применением СОК

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 ряд наиболее перспективных методов в данной области.

Литература

1. Omondi A., Premkumar B. Residue number systems. Theory and Implementation. – London: Imperial College Press, 2007.
2. Schneier Bruce. Applied Cryptography: Protocols, Algorithms, and Source Code in C, Second Edition. – New York: John Wiley & Sons, 1995. – 662 p.
3. Червяков Н.И. Реализация высокоэффективной модулярной цифровой обработки сигналов на основе программируемых логических интегральных схем // Нейрокомпьютеры: разработка и применение. №10, 2006. С. 24–36.