УДК 621.391

Н.И. Червяков, М.Г. Бабенко, П.А. Ляхов,
И.Н. Лавриненко, А.М. Лягин

УМНОЖЕНИЕ И ДЕЛЕНИЕ В СИСТЕМЕ ОСТАТОЧНЫХ КЛАССОВ С ИСПОЛЬЗОВАНИЕМ ПОЛЕЙ ГАЛУА GF(*p*)

N.I. Chervyakov, M.Gr. Babenko, P.A. Lyakhov,
I.N. Lavrinenko, A.M. Lyagin

MULTIPLICATION AND DIVISION IN THE RESIDUE NUMBER SYSTEM USING GALOIS FIELDS GF(*p*)

Предложен алгоритм умножения и деления в системе остаточных классов, основанный на теории полей Галуа GF(*p*). Применение полей Галуа GF(*p*) для решения проблем арифметического умножения и деления устраняет многие ограничения существующих алгоритмов. Преимущество предложенного алгоритма заключается в том, что он не имеет ограничений на делимое и делитель, не использует обобщенную позиционную систему счисления и расширение системы остаточных классов.

КОМПЬЮТЕРНАЯ АРИФМЕТИКА; СИСТЕМА ОСТАТОЧНЫХ КЛАССОВ; МОДУЛЯРНАЯ АРИФМЕТИКА; ПОЛЯ ГАЛУА; ТЕОРИЯ ЧИСЕЛ; ПАРАЛЛЕЛЬНЫЕ ВЫЧИСЛЕНИЯ.

The current paper presents an algorithm of multiplication and division in the residual classes based on the theory of Galois fields GF(*p*). The use of Galois fields GF(*p*) to solve the problems of arithmetic multiplication and division eliminates a lot of limitations of existing algorithms. The advantage of the proposed algorithm is that it has no restrictions on the dividend and the divisor and it does not use the generalized positional notation and the expansion of the residue number systems.

COMPUTER ARITHMETIC; RESIDUE NUMBER SYSTEM; MODULAR ARITHMETIC; GALOIS FIELD; NUMBER THEORY; PARALLEL COMPUTING.

Проблема деления в СОК в общем виде привлекает внимание многих исследователей при разработке высокопроизводительных многомодульных арифметико-логических устройств (АЛУ). Цифровые системы, построенные на основе арифметики СОК, могут сыграть важную роль в высокоскоростных системах обработки данных в режиме реального времени, с поддержкой параллельной обработки целочисленных данных [1].

Некоторые алгоритмы деления в СОК в общем случае были разработаны ранее. Эти алгоритмы можно разделить на две категории: с использованием умножения и с использованием вычитания [2, 3]. Большинство алгоритмов на основе умножения предварительно вычисляют обратную величину делителя, после чего эта величина умножается на делимое. Алгоритмы на основе вычитания используют вычитание кратных делителя из делимого до тех пор, пока полученный результат не будет меньше делителя. Некоторые известные алго-

ритмы деления в СОК на основе умножения изложены в [4–8]. Все эти алгоритмы используют преобразование в обобщенную позиционную систему счисления (ОПСС) для нахождения обратной величины делителя и сравнения чисел. Кроме того, эти алгоритмы являются медленными, т. к. требуют выполнения большого количества арифметических действий. Некоторые алгоритмы на основе вычитания представлены в [9, 10]. Эти алгоритмы не требуют вычислений в ОПСС, однако используют некоторые другие немодульные операции [11, 12]. Алгоритм с вычитанием, представленный в [10], кажется наиболее привлекательным для применения на практике, поскольку использует эффективный метод проверки четности для сравнения и обнаружения переполнения диапазона. Большинство существующих алгоритмов обладают различными недостатками, что делает их менее пригодными для решения задачи деления в СОК.

Анализ известных алгоритмов деления в СОК показывает, что большинство из них работает на основе метода спуска Ферма и итераций Ньютона, каждая из которых содержит трудные операции: масштабирования, расширения, округления и другие, включающие большое количество арифметических действий.

Так как эти методы являются итерационными, то их сравнение можно провести по двум критериям:

по общему количеству выполненных итераций;

по количеству операций, выполненных за одну итерацию

Сравнительная оценка по количеству итераций, выполненных, в алгоритмах Ферма и Ньютона приведена на основе компьютерного моделирования, результаты которого показаны на рис. 1 и 2.

Вычислительные средства, построенные на основе арифметики системы остаточных классов (СОК), могут сыграть важную роль в высокоскоростных и надежных системах обработки данных в режиме реального времени, с поддержкой параллельной обработки целочисленных данных. Операции сложения, вычитания и умножения, называемые *модульными операциями*, могут быть реализованы очень быстро, без распространения межразрядных переносов. К немодульным относятся операции деления, сравнения чисел, определения знака, определения переполнения динамического диапазона и др. Любое улучшение скорости этих медленных операций значительно улучшит производительность многомодульных арифметико-логических устройств.

В десятичной арифметике логарифмы часто используются для умножения и деления. В СОК используется аналогичный метод, называемый *индексными вычислениями* [13]. Использование индексного преобразования над полем Галуа GF(p) сводит операции умножения и деления к операциям сложения и вычитания, соответственно. Операция умножения является модульной и поэтому может быть реализована в СОК как сложение. С точки зрения аппаратной реализации, сложение в СОК реализуется легче, чем умножение [13, 14]. Деление, напротив, является немодульной операцией в СОК, поэтому использование индексных преобразований над GF(p) значительно улучшит время выполнения операции и снизит аппаратурные затраты.

При построении умножителей, работающих в модулярной арифметике, можно воспользоваться индексным или «дискретно-логарифмическим» представлением операндов и заменить операцию модулярного умножения операцией модулярного сложения индексов или дискретных логарифмов. Индексное представление модулярного числа основывается на понятии первообразного «корня по простому модулю». Таким корнем является целое число, возведение которого в степень 1, 2, ..., $p-1$ дает неповторяющиеся вычеты по модулю [13].

Индексный метод позволяет получить очень простую реализацию с помощью метода, аналогичного методу вычисления логарифмов. Умножители по модулю p_i реализуются с помощью табличного поиска при малом p_i (5 бит и менее) и индексного сложения при больших p_i (6–10 бит) [14]. Поэтому в тех случаях, когда диапазон обрабатываемых данных лежит в преде-

лах 6–10 бит, целесообразно использовать умножители и деления на основе индексного исчисления. Однако индексное исчисление может быть использовано только в том случае, когда модуль p_i – простое число.

На практике встречаются также случаи, когда модуль p_i – произвольное число, поэтому появляется необходимость разработать эффективные методы умножения, пригодные при любом модуле, независимо от того, является ли последний простым числом или нет. Так, например, если реализовать умножитель по модулю p_i= 249, который не является простым числом, т. к. 249 = 83·3, то непосредственное использование индексного счисления невозможно. Так как числа 83 и 3 являются простыми, то возможно индексное исчисление по разложенным модулям.

При умножении (делении) по модулю p_i сначала определяются индексы чисел, затем эти индексы складываются (вычитаются) по модулю p_i–1 и по обратной таблице определяется окончательный результат [11, 12, 15].

Если модуль не является простым, то необходимо разложить его на меньшие модули, провести вычисления с помощью обычных методов, а затем вновь перейти к исходному модулю. Метод индексного исчисления эффективен при обработке данных 6–10 бит и когда модуль представляет собой простое число [14].

Сравнительная оценка общего количества итераций, выполняемых в алгоритмах Ферма и Ньютона, приведена на рис. 1 и 2.

На рис. 1 приведена сравнительная оценка в случае, когда значение делителя фиксировалось, а значение делимого менялось. На рис. 2 наоборот, неизменным оставалось значение делимого, делитель принимал различные значения.

Проведенная сравнительная оценка показала, что количество итераций метода Ферма растет вместе с ростом разницы между разрядностями делимого и делителя. Количество итераций метода Ньютона от этого не зависит. Поэтому, если разница между разрядностями делимого и делителя является достаточно большой, то в этом случае хороший результат дает применение метода Ньютона. Однако если разница небольшая, то выгоднее применять для выполнения операции деления метод Ферма.

Количество операций в каждой итерации в основном определяются операциями расширения и масштабирования.

Операция масштабирования состоит в выполнении деления с нулевым остатком

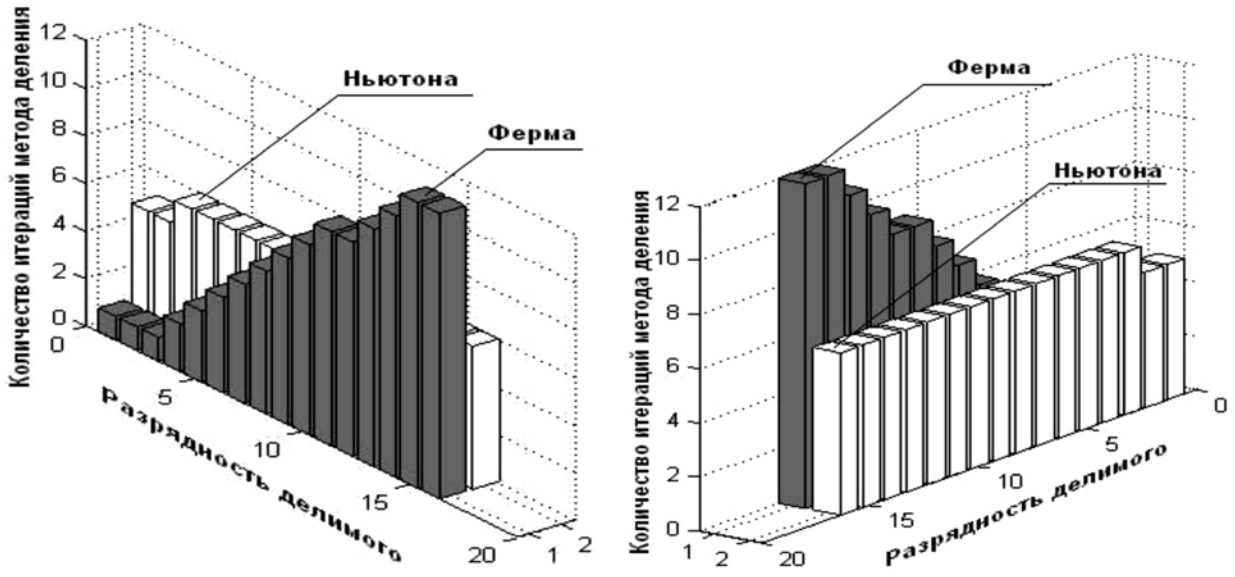

Рис. 1. Оценка количества итераций методов Ферма и Ньютона в зависимости от разрядности делимого

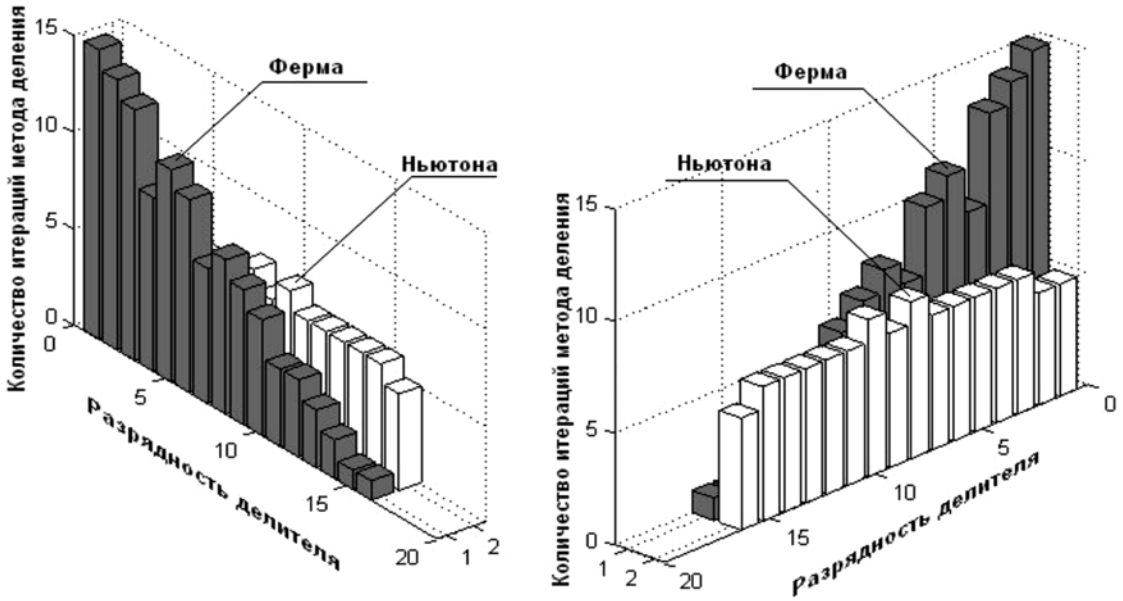

Рис. 2. Оценка количества итераций методов Ферма и Ньютона в зависимости от разрядности делителя

(Z вычитаний и Z сложений, где Z – количество модулей СОК, по которым проводится масштабирование).

Операции расширения выполняются на основе метода Гарнера, который содержит *r* операций вычитания и *r* операций умножения, где *r* – количество модулей исходной СОК.

Трудность выполнения операций модулярного деления, связанная с итерационными процессами, наводит на мысль об исключении итераций при делении изоморфизмом, позволяющем операцию деления заменить операцией вычитания остатков делимого и делителя как разность их остатков.

В настоящей статье рассматривается очень быстрый алгоритм деления в СОК с использованием индексов в GF(p). Улучшенный алгоритм обладает следующими свойствами: очень быстр по сравнению с известными алгоритмами, не использует предварительную оценку частного, обратную величину для делителя и операцию расширения базы СОК.

Постановка задачи

Модулярная арифметика обладает большими возможностями обработки данных за счет дробления операндов на несколько более мелких остатков и независимого параллельного выполнения арифметических операций сложения, вычитания и умножения с остатками.

Большой интерес к системе остаточных классов в последнее время обусловлен внедрением в цифровых системах микроэлектроники с высокой синхронизирующей частотой и низким потреблением энергии. Поскольку система остаточных классов не является взвешенной системой счисления, то операция деления, включающая такие операции, как сравнение, расширение, масштабирование и другие, не может считаться простой. Все известные алгоритмы деления характеризуются использованием итерационных принципов, что существенно понижает способность обработки данных. Для сокращения времени выполнения операции деления желательно исключить итерации. Это можно реализовать с помощью полей Галуа. Для этого необходимо использовать изоморфизм, который позволяет операцию деления заменить операцией вычитания остатков делимого и делителя как разность их остатков. В данной работе предлагается алгоритм модулярного деления без использования итерационных процессов.

Индексы в полях Галуа GF(p)

Число I, являющееся решением сравнения $q^x \equiv A(\mathrm{mod}\, p)$, называется индексом числа A и обозначается $I = \mathrm{ind}\, A$. Первообразный корень q называется основанием индекса. Для нахождения индекса числа A по модулю p надо найти первообразный корень q и затем найти решение этого сравнения для данного первообразного корня.

Пример 1. Вычислить индексы по модулю 7 чисел 0, 1, 2, 3, 4, 5, 6. Первообразные корни числа 7 суть 3 и 5.

Решение. Примем основание $q = 3$. Вычислим 3^x для $x = 0, 1, 2, 3, 4, 5, 6$.

$$3^0 \equiv 1 \bmod 7,\ 3^1 \equiv 3 \bmod 7,\ 3^2 \equiv 2 \bmod 7,$$
$$3^3 \equiv 6 \bmod 7,\ 3^4 \equiv 4 \bmod 7,\ 3^5 \equiv 5 \bmod 7,$$
$$3^6 \equiv 6 \bmod 7.$$

Из этих сравнений следует, что

$$\mathrm{ind}\, 1 = 0,\ \mathrm{ind}\, 2 = 2,\ \mathrm{ind}\, 3 = 1,\ \mathrm{ind}\, 4 = 4,$$
$$\mathrm{ind}\, 5 = 5,\ \mathrm{ind}\, 6 = 3.$$

Аналогично, индексы по первообразному корню 5:

$$\mathrm{ind}\, 1 = 0,\ \mathrm{ind}\, 2 = 4,\ \mathrm{ind}\, 3 = 5,\ \mathrm{ind}\, 4 = 2,$$
$$\mathrm{ind}\, 5 = 1,\ \mathrm{ind}\, 6 = 3.$$

Антииндексом числа I называется число a, такое что

$$I = \mathrm{ind}\, a \text{ или } a = \mathrm{ind}^{-1} I.$$

Если антииндекс обозначить через $N(I)$, то $N(\mathrm{ind}\, a) = a$.

Пример 2. Вычислить антииндексы по модулю 7 чисел 0, 1, 2, 3, 4, 5.

Решение. В предыдущем примере приведены индексы чисел от 0 до 6, которые равны 0, 1, 2, 3, 4, 5. Естественно, что число нуль не может иметь индекса, потому что нет такого показателя степени, возведя в которую конечное, отличное от нуля основание, можно было бы получить нуль.

Первообразный корень 3:

$$N(0) = 1,\ N(1) = 3,\ N(2) = 2,\ N(3) = 6,$$
$$N(4) = 4,\ N(5) = 5.$$

Первообразный корень 5:

$$N(0) = 1,\ N(1) = 5,\ N(2) = 4,\ N(3) = 6,$$
$$N(4) = 2,\ N(5) = 3.$$

При помощи индексов можно вычислять умножение, деление, возведение в степень.

Таблица 1

Индексы по модулю 7

I	q	
	3	5
0	–	–
1	0	0
2	2	4
3	1	5
4	4	2
5	5	1
6	3	3

Пример 3. Вычислить выражение

$$c = \frac{ab^3}{k^2 d} (\mathrm{mod}\, p),$$

где $a = 2$, $b = 5$, $k = 3$, $d = 4$, $p = 7$.

Решение.

1. Находим индексы величин, входящих в вычисляемое выражение, приняв первообразный корень равным 5.

Из табл. 1 индексов по модулю 7 находим:

$$\mathrm{ind}\, 2 = 4,\ \mathrm{ind}\, 5 = 1,\ \mathrm{ind}\, 3 = 5,\ \mathrm{ind}\, 4 = 2.$$

2. Вычисляем индекс результата:

$$\mathrm{ind}\, c = 4 + 1 \cdot 3 - 5 \cdot 2 - 2 (\mathrm{mod}\, 6) \equiv 1 (\mathrm{mod}\, 6).$$

3. Находим антииндекс 1: $N(1) = 5$.

Непосредственное вычисление показывает, что

$$c = \frac{2 \cdot 5^3}{3^2 \cdot 4} (\mathrm{mod}\, 7) = 5.$$

Конечные поля Галуа бывают двух типов: простые поля GF(p) и полиномиальные поля GF(p^n), где p – простое число, n – целое положительное. Все поля Галуа обладают свойством: все ненулевые элементы могут быть построены с использованием примитивного элемента, который будем обозначать g. Это свойство может быть использовано при делении над полем GF(p). Свойство определяется следующим образом [16]:

если p – любое простое число, а g – любой примитивный корень в GF(p), то для

каждого целого a, взаимно простого с p, существует единственное целое i, обозначаемое $i = \mathrm{ind}_g a$, такое что

$$a = |g^i|_p,\ 0 \le i \le p-1. \qquad (1)$$

Индексы над полем GF(p) обладают следующими важными свойствами:

(1) $\mathrm{ind}_g 1 = 0$,

(2) $\mathrm{ind}_g(ab) = |\mathrm{ind}_g a + \mathrm{ind}_g b|_{p-1}$,

(3) $\mathrm{ind}_g a^k = |k \cdot \mathrm{ind}_g a|_{p-1}$,

(4) $\mathrm{ind}_g a = |\mathrm{ind}_g g' + \mathrm{ind}_g a|_{p-1}$, где g' – любой другой примитивный корень.

В тех случаях, когда сумма индексов превышает самое большое значение в GF(p) используется теорема Ферма. При этом:

если p – простое число, то

$$|a^p|_p = |a|_p, \text{ для всех целых } a; \qquad (2)$$

если p простое число и a – целое, то

$$|a^{p-1}|p = 1; \qquad (3)$$

если p – простое число, k и a – целые числа, то

$$\left|a^k\right|_p = \left|a^{|k|_{p-1}}\right|_p. \qquad (4)$$

В СОК проблема арифметического деления делится на три категории:

деление с нулевым остатком (ДН);

масштабирование;

деление в общем случае.

Деление с нулевым остатком и масштабирование применяются для ограниченного круга приложений [17]. Для деления с нулевым остатком необходимо знать априори, что остаток равен нулю. Масштабирование реализует деление только на фиксированный делитель, при этом делитель может представляться в виде произведения нескольких модулей. Таким образом, общее деление используется, когда неизвестно априори, что делимое делится нацело на делитель, и когда неизвестно, что делитель принадлежит множеству пригодных для масштабирования значений. Эффективный алгоритм деления на основе приближенного метода рассмотрен в работах [18, 19].

Алгоритм деления с нулевым остатком над полем Галуа GF(p)

Деление с нулевым остатком требует вычисления частного $\left\lfloor \frac{x}{y} \right\rfloor$, когда известно априори, что остаток равен нулю. Поле Галуа GF(p) может быть использовано весьма эффективно при решении проблемы деления с нулевым остатком. Так, если $\{q_n\} = \{1, 2, ..., p-1\}$ – мультипликативная группа GF(p), $\{i_n\} = \{0, 1, ..., p-2\}$ – ассоциативная изоморфная группа с обозначением $q_n = \left|q^{i_n}\right|_p$, где g – примитивный элемент GF(p), а q_x и q_y – два целых числа, таких что q_y делит q_x, тогда деление с нулевым остатком $\left|\frac{q_x}{q_y}\right|_p$ может быть определено как

$$\left|\frac{q_x}{q_y}\right|_p = \left|g^{|i_x - i_y|_{p-1}}\right|_p = \left|g^{|i_x + (p-1-i_y)|_{p-1}}\right|_p. \qquad (5)$$

Если $p - 1 = \prod_{j=0}^{r-1} p_j$, так что $(p_0, p_1, ..., p_{r-1})$ взаимно простые, $\mathrm{ind}_g x = (x_1, x_2, ..., x_r)$ и $\mathrm{ind}_g y = (y_1, y_2, ..., y_r)$ и y делит x, арифметическое деление $\left\lfloor \frac{x}{y} \right\rfloor_p$ может быть определено как

$$\left|\frac{x}{y}\right|_p = \left|g^{(|x_1 - y_1|_{p_1}, |x_2 - y_2|_{p_2}, ..., |x_r - y_r|_{p_r})}\right|_p. \qquad (6)$$

Пример 4. Пусть простое число $p = 43$ и $g = 3$. Группа$\{q_n\} = \{1, 2, ..., 42\}$ – ассоциативная изоморфная группа $\{i_n\} = \{0, 1, ..., 41\}$. Так как $p-1 = 42$, возьмем модули $p_1 = 2$, $p_2 = 3$ и $p_3 = 7$. Любое целое i_n может быть единственным образом представлено в виде тройки $r_n = (|i_n|_2, |i_n|_3, |i_n|_7)$. Множество всех троек образует группу $\{r_n\}$. Изоморфизм между группами $\{q_n\}$ и $\{r_n\}$ показан в табл. 2.

В табл. 3 показан обратный переход к GF(43).

Вычисление табличных значений покажем для $q_n = 27$. Первый индекс вычисляется следующим образом:

$i_n = \mathrm{ind}_3 27 = 3$, то есть $27 = \left|3^{i_n}\right|_{43} = \left|3^3\right|_{43}$.

Тройка, полученная из i_n:

$r_n = (|i_n|_2, |i_n|_3, |i_n|_7) = (|3|_2, |3|_3, |3|_7) = (1, 0, 3)$.

В табл. 2 $q_n = 27$ соответствует запись $(1, 0, 3) = (1, 00, 011)_2$. Результат записан

Таблица 2

Изоморфизм между группами $\{q_n\}$ и $\{r_n\}$ для GF(43), $p = 43$, $g = 3$, $p_1 = 2$, $p_2 = 3$ и $p_3 = 7$

q_n	r_n	q_n	r_n	q_n	r_n	q_n	r_n	q_n	r_n
1 000001	(0, 0, 0) 0 00 000	10 001010	(0, 1, 3) 0 01 011	19 010011	(1, 1, 6) 1 01 110	28 011100	(1, 2, 5) 1 10 101	37 100101	(1, 1, 0) 1 01 000
2 000010	(1, 0, 6) 1 00 110	11 001011	(0, 0, 2) 0 00 010	20 010100	(1, 1, 2) 1 01 010	29 011101	(1, 2, 6) 1 10 110	38 100110	(0, 1, 4) 0 01 100
3 000011	(1, 1, 1) 1 01 001	12 001100	(1, 1, 5) 1 01 101	21 010101	(0, 0, 1) 0 00 001	30 011110	(1, 2, 4) 1 10 100	39 100111	(1, 0, 5) 1 00 101
4 000100	(0, 0, 5) 0 00 101	13 001101	(0, 2, 4) 0 10 100	22 010110	(1, 0, 1) 1 00 001	31 011111	(0, 1, 6) 0 01 110	40 101000	(0, 1, 1) 0 01 001
5 000101	(1, 1, 4) 1 01 100	14 001110	(0, 2, 6) 0 10 110	23 010111	(0, 1, 2) 0 01 010	32 100000	(1, 0, 2) 1 00 010	41 101001	(0, 0, 6) 0 00 110
6 000110	(0, 1, 0) 0 01 000	15 001111	(0, 2, 5) 0 10 101	24 011000	(0, 1, 5) 0 01 101	33 100001	(1, 1, 3) 1 01 011	42 101010	(1, 0, 0) 1 00 000
7 000111	(1, 2, 0) 1 10 000	16 010000	(0, 0, 3) 0 00 011	25 011001	(0, 2, 1) 0 10 001	34 100010	(1, 2, 2) 1 10 010		
8 001000	(1, 0, 4) 1 00 100	17 010001	(0, 2, 3) 0 10 011	26 011010	(1, 2, 3) 1 10 101	35 100011	(0, 0, 4) 0 00 100		
9 001001	(0, 2, 2) 0 10 010	18 010010	(1, 2, 1) 1 10 001	27 011011	(1, 0, 3) 1 00 011	36 100100	(0, 2, 0) 0 10 000		

Таблица 3

Обратное преобразование в GF(43), $p = 43$, $g = 3$, $p_1 = 2$, $p_2 = 3$ и $p_3 = 7$

r_n	q_n	r_n	q_n	r_n	q_n	r_n	q_n	r_n	q_n
(0, 0, 0) 0 00 000 = 0	1	(0, 1, 2) 0 01 010 = 10	23	(0, 2, 4) 0 10 100 = 20	13	(1, 0, 6) 1 00 110 = 38	2	(1, 2, 1) 1 10 001 = 49	18
(0, 0, 1) 0 00 001 = 1	21	(0, 1, 3) 0 01 011 = 11	10	(0, 2, 5) 0 10 101 = 21	15	(1, 1, 0) 1 01 000 = 40	37	(1, 2, 2) 1 10 010 = 50	34
(0, 0, 2) 0 00 010 = 2	11	(0, 1, 4) 0 01 100 = 12	38	(0, 2, 6) 0 10 110 = 22	14	(1, 1, 1) 1 01 001 = 41	3	(1, 2, 3) 1 10 101 = 51	26
(0, 0, 3) 0 00 011 = 3	16	(0, 1, 5) 0 01 101 = 13	24	(1, 0, 0) 1 00 000 = 32	42	(1, 1, 2) 1 01 010 = 42	20	(1, 2, 4) 1 10 100 = 52	30
(0, 0, 4) 0 00 100 = 4	35	(0, 1, 6) 0 01 110 = 14	31	(1, 0, 1) 1 00 001 = 33	22	(1, 1, 3) 1 01 011 = 43	33	(1, 2, 5) 1 10 101 = 53	28
(0, 0, 5) 0 00 101 = 5		(0, 2, 0) 0 10 000 = 16	6	(1, 0, 2) 1 00 010 = 34	2	(1, 1, 4) 1 01 100 = 44		(1, 2, 6) 1 10 110 = 54	9
(0, 0, 6) 0 00 110 = 6	1	(0, 2, 1) 0 10 001 = 17	5	(1, 0, 3) 1 00 011 = 35	7	(1, 1, 5) 1 01 101 = 45	2		
(0, 1, 0) 0 01 000 = 8		(0, 2, 2) 0 10 010 = 18		(1, 0, 4) 1 00 100 = 36		(1, 1, 6) 1 01 110 = 46	9		
(0, 1, 1) 0 01 001 = 9	0	(0, 2, 3) 0 10 011 = 19	7	(1, 0, 5) 1 00 101 = 37	9	(1, 2, 0) 1 10 000 = 48			

в двоичной форме и содержит 1 бит, 2 бита и 3 бита для записи значения по mod 2, mod 3 и mod 7 соответственно.

Вычитание по модулю p выполняется одновременно по всем модулям

$$p-1=\prod_{i=1}^{r} p_i.$$

Существует множество вариантов, из которых можно выбрать набор модулей для СОК систем. Скорость и эффективность деления нацело увеличивается, если подбирать модули на основе следующих критериев:

1) $p-1=\prod_{i=1}^{r} p_i$;

2) они должны быть взаимно простыми $(p_i, p_j)=1$, если $i \neq j$;

3) $\sum_{j=1}^{r} \log(p_j-1) \to \min$.

Пример 5. При $p = 43$, $g = 3$, $p_1 = 2$, $p_2 = 3$ и $p_3 = 7$. Найти $\frac{36}{2}$. Это задача деления с нулевым остатком.

По табл. 2 получаем:

для 36: $\text{ind}_3 36 = (0, 2, 0)$;
для 2: $\text{ind}_3 2 = (1, 0, 6)$.

Используя вычитание по модулям 2, 3 и 7, получаем

$$\text{ind}_3\left(\frac{36}{2}\right)=(0,2,0)-(1,0,6)=(1,2,1),$$

Из табл. 3 следует, что результат деления чисел $\frac{36}{2}$ равен

$$(1,2,1)=(1\ 10\ 001)_2=49.$$

Схема для реализации процесса с нулевым остатком представлена на рис. 3. Реализация операции сложения проще, чем операции вычитания, поэтому используется дополнительный код делителя, позволяющий заменить операцию вычитания операцией сложения.

На вход $LUT_1 p_i$ поступает двоичный код делимого, а на вход $LUT_2 p_i$ – двоичный код делителя. На выходе $LUT_1 p_i$ формируется прямой код индекса делимого, а на вы-

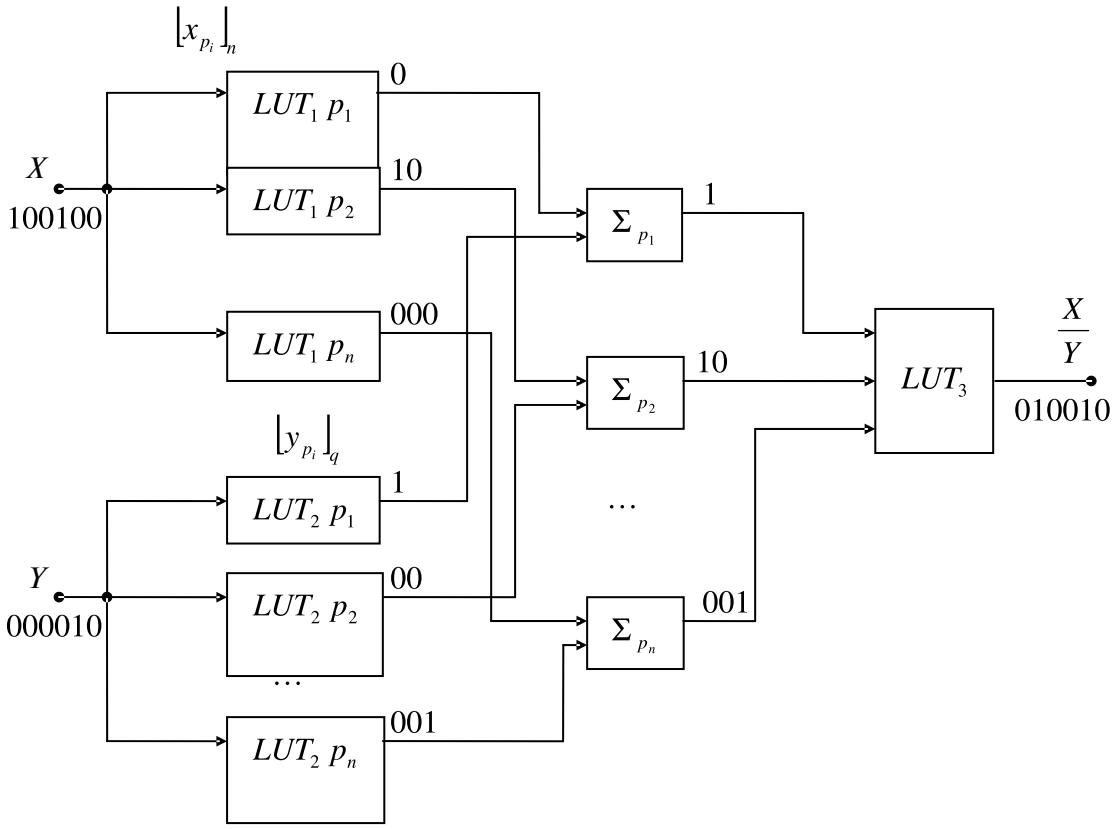

Рис. 3. Деление без остатка по модулям СОК в GF(p) при n двоичных разрядах

ходе LUT_2p_i формируется дополнительный код индекса делителя. Индексы делимого и делителя суммируются модульными сумматорами Σ_{p_i} и поступают на вход LUT_3p_i, где формируется частное $\frac{X}{Y}$.

На рис. 3 показаны конкретные значения индексов и антииндексов для примера 5.

Масштабирование чисел может быть выполнено как последовательность решения задач деления с нулевым остатком, если делитель является произведением некоторых модулей, причем порядок деления с нулевым остатком не имеет значения.

Общее арифметическое деление над полем Галуа GF(p) используется в случае, когда заранее неизвестно, какой делитель представляет кратное число делимого, или когда делимое есть произведение некоторых модулей. Общая проблема деления $\frac{X}{Y}$ может быть сведена к процессу масштабирования $\frac{X}{Y'}$, где Y' удовлетворяет следующим условиям:

1. $Y' > Y$.

2. Y' является произведением некоторых модулей СОК.

Выбор Y' подробно рассмотрен в [11].

Оценим вычислительную сложность предлагаемого алгоритма и сравним с известным алгоритмом, предложенным в работе [4]. Анализ сложности алгоритма базируется на числе модулярных вычислений и логической глубине (количество последовательно соединенных модульных операций). Вычислительная сложность определяется вычислением в полях Галуа GF(p), которые выполняются параллельно. В представленном алгоритме логическая глубина равна трем и имеет представление $LUT_{(1,2)}P \to \sum p_i \to LUT_3p$, при этом длительность выполнения операции деления $t_g = t_1' + t_2 + t_1''$, где $t_1' = t_1'' = O(\log p)$ – такт синхронизации доступа к памяти, $t_2 = O\left(\sum_{i=1}^{r} \lceil \log p_i \rceil\right)$ – такт синхронизации модульного сложения. Разрядность данных $R = O(\lceil \log p_i \rceil)$. Реализация таблиц $LUT_{(1,2)}$ требует памяти объемом $M = n2^n$.

В известном алгоритме [4] используется одна операция сложения, одна операция деления и процедура расширения базы, состоящая в нахождении остатков представления числа по модулю делителя. Эта проблема по сути межмодульная и в классическом подходе ее сложность линейно зависит от числа модулей, т. к. она включает все остатки текущего представления и вычисляется последовательно по алгоритму Гарнера в котором необходимы ряд шагов того же порядка, что и r, где r – количество модулей СОК. При этом каждый шаг состоит из операций сложения и умножения, сложность которых составляют $O\left(2 + 2r \cdot \sum_{i=1}^{r} \lceil \log p_i \rceil\right)$. Логическая глубина равна $2r + 2$.

Вычислительная сложность предложенного метода без учета доступа к памяти примерно в n раз меньше, чем известного. Логическая глубина представленного метода равна трем, а известного – $2r + 2$. Современная вычислительная база позволяет приблизить время доступа к времени вычисления выполнения модульных операций в СОК.

В современных ПЛИС Xilinx время распространения сигнала через LUT-таблицу составляет около 0,5 нс, через блок метки ускоренного переноса – 0,1 нс, а время переключения триггера – до 0,5 нс.

Время суммирования двух 16-разрядных операндов составляет 5 нс.

Время доступа ROM на основе таблиц, близкое к 10 нс [20].

Программируемые ПЗУ семейства XC 4000 решают функции комбинационного устройства, поэтому обладают высокой скоростью.

При использовании таблиц для суммирования двух операндов, реализованных на базе ПЗУ, общая емкость ПЗУ представляется в виде выражения $V_{\text{ПЗУ}} = \left(\sum_{i=1}^{r} p_i \log_2 p_i\right)_{bit}$. Быстродействие сумматоров определяется максимальным модулем p_i.

При нежестком допущении приведем сравнительную оценку по быстродействию предлагаемого алгоритма с известными.

Быстродействие предлагаемого алгоритма t_A определяется как $t_A = t_A(LUT_{(1,2)}) + t_2 +$

$+ t_A(LUT_3) = 10$ нс $+ 5$ нс $+ 10$ нс $= 25$.

Быстродействие известных алгоритмов в основном определяется операцией расширения, которая базируется на алгоритме Гарнера и включает r операций вычитания и n операций умножения. Тогда быстродействие одной итерации известных алгоритмов $t'_{A'}$ определяется как $t'_A = nt_a + nt_y$. Для нашего случая $n = 3$, $t_a = 5$ нс и $t_y = 0,5$ нс (в качестве умножителей на константу используются LUT-таблицы), тогда $t'_{A'} = 3 \cdot 5 + 3 \cdot 1,5 = 15 + 4,5 = 19,5$ нс.

При делении чисел разрядностью 6–10 бит необходимо (рис. 1 и 2) 6–8 итераций, тогда быстродействие известных алгоритмов $t'_A \approx (6 - 8) \cdot 19,5$ нс $= (117 - 156)$ нс.

Соотношение $\frac{t'_A}{t_A} \approx (5 - 6)$ в пользу предложенного алгоритма арифметического деления над полем Галуа. При увеличении количества модулей p_i, соотношение $\frac{t'_A}{t_A}$ растет, и уже при $n > 5$ выигрыш в скорости вычислений представленного метода выше, чем известного.

Предложенный метод модулярного деления на основе использования индексов легко реализуем на специальных наборах оснований СОК и может дать значительное преимущество не только в тех приложениях, в которых основная доля вычислений приходится на точное умножение, возведение в степень в сочетании со сложением и вычитанием, но и в тех приложениях, в которых часто появляется необходимость в делении, сравнении и определении знака числа.

Рассмотренный алгоритм над полем Галуа GF(p) обеспечивает эффективность общего деления в системе остаточных классов. Полученные новые результаты эффективного выполнения модулярного деления в диапазоне 6–10 бит являются развитием теории математических основ разработки и проектирования точных и безошибочных теоретико-числовых арифметико-логических устройств процессоров, функционирующих в СОК.

Исследования выполнены при финансовой поддержке РФФИ в рамках научного проекта № 13-07-00478-а и ФЦП № 14.B37.21.1128.

СПИСОК ЛИТЕРАТУРЫ

1. **Claudio E.D., Piazza F., Orlandi G.** Fast combinatorial RNS processors for DSP applications // IEEE Trans. Comput. 1995. Vol. 44. No. 5. Pp. 624–633.

2. **Waser S., Flynn M.J.** Introduction to Arithmetic for Digital System Designers. Holt, Rinehart Winston. New York, 1982.

3. **Taylor F.J.** Residue arithmetic: A tutorial with examples // IEEE Comput. 1984. Vol. 17. No. 5. Pp. 50–62.

4. **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. Юбилейная междунар. науч.-техн. конф. Сб. трудов. М., 2000. С. 291–310.

5. **Chren W.A.** A new residue number system division algorithm // Computers Math. Applic. 1990. Vol. 19. No. 7. Pp. 13–29.

6. **Banerji D.K., Cheung T.Y., Ganesan V.** A high-speed division method in residue arithmetic // IEEE Symp. Comput. Arithmetic. 1981. No. 5. Pp. 158–164.

7. **Hits M.A., Kaltofen E.** Integer division in residue number systems // IEEE Trans. Comput. 1995. Vol. 44. No. 8. Pp. 983–989.

8. **Kinoshita E., Kosako H., Kojima Y.** General division in the symmetric residue number system // IEEE Trans. Comput. 1973. Vol. 22. Pp. 134–142.

9. **Lin M.L., Leiss E., McInnis B.** Division and sign detection algorithm for residue number systems // Computers Math. Applic. 1984. Vol. 10. No. 4/5. Pp. 331–342.

10. **Radhakrishnan D., Yuan Y.** Novel approaches to the design of VLSI RNS multipliers // IEEE Trans. Circuits and System. 1992. Vol. 39. No. 1. Pp. 52–57.

11. **Hung C., Parhami B.** Fast RNS division algorithms for fixed divisors with application to RSA encryption // Information Processing Letters. 1994. No. 51. Pp. 163–169.

12. **Hung C., Parhami B.** An approximate sign detection method for residue numbers and its application to RNS division // Computers Math. Applic. 1994. Vol. 27. No. 4. Pp. 23–35.

13. **Акушский И.Я., Юдицкий Д.И.** Машинная арифметика в остаточных классах. М.: Сов. радио, 1968. 440 с.

14. **Содерстренд М.А., Верниа К.** Недорогой быстродействующий умножитель по модулю и его применение в арифметических устройствах на основе системы счисления в остаточных

классах. ТИИЭР, 1980. № 4. Т. 68.

15. **Julien G.A.** Residue number scaling and other operations using ROM arrays // IEEE Trans. on Comput. 1978. Vol. 27. No. 4. Pp. 325–336.

16. **Beyer W.** CRC-Standard Mathematical Tables and Formulae. Edition 91-101. CRC Press, 1991.

17. **Talameh S., Siy P.** Arithmetic Division in RNS Using Galois Field GF(p) // Computers and Mathematics with Applications. 2000. Vol. 39. Pp. 227–238.

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

19. **Червяков Н.И., Ляхов П.А.** Метод определения знака числа в системе остаточных классов на основе приближенных вычислений // Нейрокомпьютеры: разработка, применение. 2012. № 12. С. 56–64.

20. **Alia G., Martinelli E.** Neuron: a Rom based RNS oligital neuron // Neural Networks. 2005. Pp. 174–189.

REFERENCES

1. **Claudio E.D., Piazza F., Orlandi G.** Fast combinatorial RNS processors for DSP applications. *IEEE Trans. Comput*, 1995, Vol. 44, No. 5, Pp. 624–633.

2. **Waser S., Flynn M.J.** *Introduction to Arithmetic for Digital System Designers.* Holt, Rinehart Winston. New York, 1982.

3. **Taylor F.J.** Residue arithmetic: A tutorial with examples, *IEEE Comput.*, 1984, Vol. 17, No. 5, Pp. 50–62.

4. **Chervyakov N.I., Lavrinenko S.V., Mezentseva O.S.** Metody i algoritmy okrugleniya, masshtabirovaniya i deleniya chisel v modulyarnoy arifmetike, *50 let modulyarnoy arifmetike. Yubileynaya mezhdunarodnaya nauchno-tekhnicheskaya konferentsiya,* Moscow, 2000, Pp. 291–310. (rus)

5. **Chren W.A.** A new residue number system division algorithm, *Computers Math. Applic*, 1990, Vol. 19, No. 7, Pp. 13–29.

6. **Banerji D.K., Cheung T.Y., Ganesan V.** A high-speed division method in residue arithmetic, *IEEE Symp. Comput. Arithmetic*, 1981, No. 5, Pp. 158–164.

7. **Hits M.A., Kaltofen E.** Integer division in residue number systems, *IEEE Trans. Comput.*, 1995, Vol. 44, No. 8, Pp. 983–989.

8. **Kinoshita E., Kosako H., Kojima Y.** General division in the symmetric residue number system, *IEEE Trans. Comput.*, 1973, Vol. 22, Pp. 134–142.

9. **Lin M.L., Leiss E., McInnis B.** Division and sign detection algorithm for residue number systems, *Computers Math. Applic*, 1984, Vol. 10, No. 4/5, Pp. 331–342.

10. **Radhakrishnan D., Yuan Y.** Novel approaches to the design of VLSI RNS multipliers, *IEEE Trans. Circuits and System*, 1992, Vol. 39, No. 1, Pp. 52–57.

11. **Hung C., Parhami B.** Fast RNS division algorithms for fixed divisors with application to RSA encryption, *Information Processing Letters*, 1994, No. 51, Pp. 163–169.

12. **Hung C., Parhami B.** An approximate sign detection method for residue numbers and its application to RNS division, *Computers Math. Applic.*, 1994, Vol. 27, No. 4, Pp. 23–35.

13. **Akushskiy I. Ya., Yuditskiy D.I.** *Mashinnaya arifmetika v ostatochnykh klassakh.* Moscow: Sovetskoye radio Publ., 1968. 440 p. (rus)

14. **Soderstrend M.A., Vernia K.** Nedorogoy bystrodeystvuyushchiy umnozhitel po modulyu i yego primeneniye v arifmeticheskikh ustroystvakh na osnove sistemy schisleniya v ostatochnykhklassakh, *TIIER*, 1980, No. 4, Vol. 68. (rus)

15. **Julien G.A.** Residue number scaling and other operations using ROM arrays, *IEEE Trans. on Comput.*, 1978, Vol. 27, No. 4, Pp. 325–336. (rus)

16. **Beyer W.** *CRC-Standard Mathematical Tables and Formulae. Edition 91-101,* CRC Press, 1991.

17. **Talameh S., Siy P.** Arithmetic Division in RNS Using Galois Field GF(p), *Computers and Mathematics with Applications*, 2000, Vol. 39, Pp. 227–238.

18. **Chervyakov N.I.** Metody, algoritmy I tekhnicheskaya realizatsiya osnovnykh problemnykh operatsiy, vypolnyayemykh v sisteme ostatochnykh klassov, *Infokommunikatsionnyye tekhnologii*, 2011, No. 4, Pp. 4–12. (rus)

19. **Chervyakov N.I., Lyakhov P.A.** Metod opredeleniya znaka chisla v sisteme ostatochnykh klassov na osnove priblizhennykh vychisleniy, *Neyrokompyutery: razrabotka, primeneniye*, 2012, No. 12, Pp. 56–64. (rus)

20. **Alia G., Martinelli E.** Neuron: a Rom Based RNS Oligital Neuron, *Neural Networks*, 2005, Pp. 174–189.

ЧЕРВЯКОВ Николай Иванович – *заведующий кафедрой прикладной математики и математического моделирования Института математики и естественных наук Северо-Кавказского федерального университета, профессор, доктор технических наук.*
355029, Россия, г. Ставрополь, пр. Кулакова, д. 2.
E-mail: k-fmf-primath@stavsu.ru

CHERVYAKOV, Nikolay I. *North-Caucasus Federal University.*
355029, Kulakov Ave. 2, Stavropol, Russia.
E-mail: k-fmf-primath@stavsu.ru

БАБЕНКО Михаил Григорьевич – *доцент кафедры высшей алгебры и геометрии Северо-Кавказского федерального университета, кандидат физико-математических наук.*
355029, Россия, г. Ставрополь, пр. Кулакова, д. 2.
E-mail: whbear@yandex.ru

BABENKO, Mikhail Gr. *North-Caucasus Federal University.*
355029, Kulakov Ave. 2, Stavropol, Russia.
E-mail: whbear@yandex.ru

ЛЯХОВ Павел Алексеевич – *доцент кафедры прикладной математики и математического моделирования Института математики и естественных наук Северо-Кавказского федерального университета, кандидат физико-математических наук.*
355029, Россия, г. Ставрополь, пр. Кулакова, д. 2.
E-mail: ljahov@mail.ru

LYAKHOV, Pavel A. *North-Caucasus Federal University.*
355029, Kulakov Ave. 2, Stavropol, Russia.
E-mail: ljahov@mail.ru

ЛАВРИНЕНКО Ирина Николаевна – *доцент кафедры высшей алгебры и геометрии Института математики и естественных наук Северо-Кавказского федерального университета, кандидат физико-математических наук.*
355029, Россия, г. Ставрополь, пр. Кулакова, д. 2.
E-mail: Algebra223@yandex.ru

LAVRINENKO, Irina N. *North-Caucasus Federal University.*
355029, Kulakov Ave. 2, Stavropol, Russia.
E-mail: Algebra223@yandex.ru

ЛЯГИН Алексей Михайлович – *доцент кафедры прикладной математики и математического моделирования Института математики и естественных наук Северо-Кавказского федерального университета, кандидат технических наук.*
355029, Россия, г. Ставрополь, пр. Кулакова, д. 2.
E-mail: kfmf-primath@stavsu.ru

LYAGIN, Aleksey M. *North-Caucasus Federal University.*
355029, Kulakov Ave. 2, Stavropol, Russia.
E-mail: kfmf-primath@stavsu.ru