# On the Possibility of Implementing High-Precision Calculations in Residue Numeral System

Otsokov Sh.A[1]
Dept. of Computing Machines, Systems and Networks
National Research University "Moscow Power Engineering Institute" Moscow, Russian Federation

Magomedov Sh.G[2]
Dept. of Intelligent Information Security Systems
MIREA Russian Technological University
Moscow, Russian Federation

***Abstract*—This article proposes a method for accelerating high-precision calculations by parallelizing arithmetic operations of addition, subtraction and multiplication. The proposed approach allows us to apply the advantages of the residue numeral system: absence of carry-overs when adding, subtracting, multiplying and reducing high-precision calculations with numbers of high digit capacity to parallel and independent execution of arithmetic operations with numbers of low digit capacity across many modules. Due to the complexity of performing non-modular operations such as: inverse transformation into a positional numeral system, number comparisons, sign identification and number rank calculation in a residue numeral system, the effect of acceleration of high-precision calculations is possible when solving some computational problems with a small number of non-modular operations, for example: determination of the scalar product of vectors, discrete Fourier transformation, iterative solution of systems of linear equations by the methods of Jacoby, Gaussa-Zeidel, etc. Implementation of the proposed method are demonstrated by the example of finding the scalar product of vectors.**

***Keywords*—High-precision calculations; residue numeral system; positional numeral system; number conversion; rank determination***

## I. Introduction

The double-precision floating-point format, supported by modern computer processors, is sufficient to solve many computational problems. However, certain tasks exist in computational practice, for example, in the fields of nanoelectronics, nuclear physics, robotics, computational geometry and others, where high-precision computer calculations are required [1-4] and traditional double-precision floating-point computer calculations provide the wrong result [5-6].

High-precision calculations have been programmatically implemented and for various programming languages there are libraries and packages supporting floating-point calculations of arbitrary accuracy, for example, ZREAL, MParith, GMP etc. [7-11]

A significant drawback of such libraries and packages is a sharp decrease in the computational speed with increasing accuracy or the length of the mantissa of a floating-point number. Attempts to speed up high-precision calculations at the level of floating-point arithmetic algorithms do not provide significant gains due to the fact that arithmetic operations of addition, subtraction and multiplication of numbers in a positional numeral system are poorly parallelized due to inter-digit carry.

These studies were conducted with the support of RTU MIREA within the framework of the initiative research work of MSEC-5 "Development of an automated procurement management system".

The purpose of this article is to propose a method for speeding up high-precision calculations by switching to a residue number system (RNS) in which the operations of addition, subtraction and multiplication can be parallelized. In RNS integers are represented by their values modulo several pairwise coprime integers called the moduli, and arithmetic operations are performed in parallel and independently for each of the moduli [12-16]. The final result of these calculations is converted to a positional numeral system.

Speeding up of arithmetic operations is achieved due to the fact that parallel calculations are performed with low-digit numbers.

The disadvantages of RNS include the difficulty of performing such operations as division, comparison, left and right shifts, rounding, converting a number to a positional numeral system and others, that are called non-modular operations [12]. Therefore, modular arithmetic is mainly used for tasks in which no or a small number of non-modular operations are required, for example, determining the scalar product of vectors, etc. [17].

Modular arithmetic is integer, but arithmetic operations with rational numbers can also be carried out in RNS [16].

Authors in [18-19] describe the method for representing numbers and the algorithms for performing arithmetic operations in RNS, including division. In this article two ways of representing floating-point numbers in RNS are discussed and the possibilities for parallelizing arithmetic operations of addition, subtraction and multiplication are explored.

In the next section a possible way of representing numbers in RNS and a mixed numeral system is considered.

## II. Residue Number System

First, confirm that you have the correct template for your paper size. This template has been tailored for output on the US-letter paper size.

Consider a set of integers $p_1$, $p_2$, ..., $p_n$, called the moduli, such that the following inequality holds:

$$2=p_1<p_2<, ..., <p_n \quad (1)$$

Let P be the product of all the moduli:

$$P=\prod_{i=1}^{n} p_i \quad (2)$$

According to the Chinese remainder theorem [14], all integers belonging to the range.

$$[0, \ldots, P-1] \quad (3)$$

have a unique representation in the residue number system by the moduli (1).

Since $p_1$, $p_2$, ..., $p_n$ are prime numbers and one of them equals 2, their product $P$ is an even number and the following ranges are used to represent positive and negative numbers [12]:

$$[0, ..., P/2-1], [P/2, ..., P-1] \quad (4)$$

for positive and negative numbers respectively.

Thus, in RNS all integers from the following range are unambiguously represented:

$$[-P/2, ..., P/2] \quad (5)$$

Let $P/2>q^{(n_f)}-1$.

Any integer belonging to the range (2) has a unique representation in the mixed number system.

## III. Representation of Floating-Point Numbers in RNS

Consider the following representation of floating-point numbers:

$$A=K\cdot q^t \quad (6)$$

where $A$ is a floating-point number, $K$ is the mantissa of $A$, an integer such that $|K|\le q^{(n_f)}-1$, $q$ is the base of the numeral system, $t$ is the order, an integer such that $|t|\le k_f$, $n_f$ is a natural number characterizing the length of the mantissa of the floating-point number, $k_f$ is a natural number characterizing the maximum order of representable numbers.

The floating-point format (6) differs from the traditional floating-point number format [6] and is more convenient for representation in RNS.

Table I shows the maximum and minimum positive and negative numbers representable in (6):

The range of representable numbers in (6) is the following:

$$(-q^{(n_f+k_f)}, q^{(n_f+k_f)}) \quad (7)$$

Consider the following representation of the floating-point format (6) in RNS:

$$A=[(\alpha_1,\alpha_2,...,\alpha_i,...,\alpha_n),t] \quad (8)$$

where $\alpha_i=|K|_{pi}$.

TABLE. I. Representable Number Ranges

| | |
|---|---|
| Maximum positive | $q^{(n_f+k_f)}$ |
| Minimum positive | $q^{(-k_f-n_f)}$ |
| Minimum negative | $-q^{(n_f+k_f)}$ |
| Maximum negative | $-q^{(-k_f-n_f)}$ |

Given the complexity of performing non-modular operations in RNS, in particular the left shift, alignment of orders during addition and subtraction is performed by the right shift.

For the same reason, because of the complexity of the effective implementation of the normalization operation, this article describes an unnormalized floating-point format (8) and there is no normalization operation.

In the next section the rules for performing arithmetic operations of addition, subtraction and multiplication are considered.

## IV. Rules for Performing Arithmetic Operations with a Floating Point in RNS

Consider two floating-point numbers in the format (8):

$$A_1=[(\alpha_1,\alpha_2,...,\alpha_i,...,\alpha_n),t_1] \quad (9)$$

$$A_2=[(\beta_1,\beta_2,...,\beta_i,...,\beta_n),s_1]$$

*1)* The product of the numbers $A_1\cdot A_2$ is

$$A_3=A_1\cdot A_2=[(\chi_1,\chi_2,...,\chi_i,...,\chi_n),t_1+s_1] \quad (10)$$

where

$$(\chi_1,\chi_2,...,\chi_n)=(\alpha_1\cdot\beta_1 \bmod p_1,\alpha_2\cdot\beta_2 \bmod p_2,...,\alpha_n\cdot\beta_n \bmod p_n)$$

*2)* The sum of the numbers $A_1+A_2$ is

$$A_4=A_1+A_2=K_4\cdot q^z$$

Let $s_1>t_1$. Then

$$A_1\pm A_2=K_1\cdot q^{(t_1)}\pm K_2\cdot q^{(s_1)}=q^{(t_1)}\cdot(K_1\pm q^{(s_1-t_1)}\cdot K_2)=$$

$$=[(\delta_1,\delta_2,...,\delta_n),t_1] \quad (11)$$

where

$$(\delta_1,\delta_2,...,\delta_n)=(\alpha_1\pm q^{(s_1-t_1)}\cdot\beta_1 \bmod p_1,\alpha_2\pm q^{(s_1-t_1)}\cdot\beta_2 \bmod p_2,...,\alpha_n\pm e\cdot q^{(s_1-t_1)} \bmod p_n),\ z=t_1$$

The diagram of high-precision calculations in modular arithmetic is shown in Fig. 1.

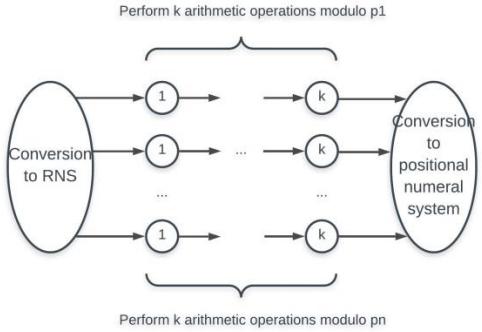


Fig. 1. Diagram of High-Precision Calculations in Modular Arithmetic.

In accordance with this diagram, the source data is converted from a positional numeral system to RNS by moduli (1). Then k arithmetic operations are carried out and the result is converted to a positional numeral system.

Using formulas (10) and (11), it is possible to carry out arithmetic operations with floating-point numbers according to the rules of modular arithmetic in parallel and independently for individual moduli. In this case, the mantissa and orders are calculated in parallel and independently from each other by formulas (10) and (11). Complexity arises when the result is outside of the allowed range defined by the moduli. In this regard, two questions arise:

*1)* How to choose the moduli so that the result is not outside the allowed range?

*2)* How to convert results from RNS to a positional numeral system?

To answer the first question, it is enough to estimate the top boundary of the result and choose the moduli so that in RNS all numbers smaller than this estimate are uniquely represented. In the next section an estimation method using the example of calculating the scalar product of two vectors is considered [20].

## V. Method for Choosing the Moduli

Let us estimate the order of the result of the scalar product of two vectors.

Consider two vectors $X=(x_1,x_2,...,x_k)$, $Y=(y_1,y_2,...,y_k)$

The scalar product is determined by the formula:

$$(X,Y)=x_1\cdot y_1+x_2\cdot y_2+...+x_k\cdot y_k \quad (12)$$

Consider two floating-point numbers in the format (6).

Then for multiplication the following is true:

$$A_3=A_1\cdot A_2=K_1\cdot K_2\cdot q^{(t+s)}<q^{(n_f)}\cdot q^{(n_f)}\cdot q^{(t+s)}=$$

$$=q^{(2\cdot n_f)}\cdot q^{(t+s)}$$

which means

$$A_3=K_3\cdot q^{(t+s)},\ |K_3|\le q^{(2\cdot n_f)},\ |t+s|\le 2\cdot k_f \quad (13)$$

Substituting $x_i\cdot y_i=K_i\cdot q^{(t_i)}$ in (12) using the format (6) and considering (13) the scalar product can be presented as:

$$(X,Y)=K_1\cdot q^{(t_1)}+K_2\cdot q^{(t_2)}+...+K_k\cdot q^{(t_k)}$$

Let $t_1=\min(t_1,...,t_k)$. Then

$$(X,Y)=q^{(t_1)}\ (K_1+K_2\cdot q^{(t_2-t_1)}+...+K_k\cdot q^{(t_k-t_1)})\le$$

$$\le q^{(t_1)}\ (K_1+q^{(2\cdot n_f)}\cdot q^{(4\cdot k_f)}+...+q^{(2\cdot n_f)}\cdot q^{(4\cdot k_f)})\le$$

$$\le q^{(t_1)}\ k\cdot q^{(2\cdot n_f)}\cdot q^{(4\cdot k_f)}$$

From this expression it follows that the maximum value of the mantissa for the scalar product (12) is

$$k\cdot q^{(2\cdot n_f)}\cdot q^{(4\cdot k_f)}$$

Then to represent the mantissa of the result of the scalar product in RNS the moduli should be chosen so that the following inequality holds:

$$P/2\ge k\cdot q^{(2\cdot n_f)}\cdot q^{(4\cdot k_f)} \quad (14)$$

To answer the second question regarding the conversion of numbers from RNS to a positional numeral system, an auxiliary method for determining the rank of a number that is used in the conversion process is considered.

## VI. Auxiliary Method to Determine the Rank of a Number

Any integer $A\in[0,P-1]$ can be represented as:

$$A=\sum_{i=1}^{n} B_i\cdot\gamma_i - rank\cdot P \quad (15)$$

Where $B_i$ are orthogonal bases, *rank* is the largest positive integer such that $A<P$.

Orthogonal bases $B_i$ are constants for RNS with given moduli and are determined by the formulas:

$$B_i=m_i\cdot P/p_i, \quad (16)$$

$$m_i=\left|\frac{P}{p_i}\right|_{p_i}^{-1}$$

Maximum possible value of *rank* is determined by the following equations:

$$\sum_{i=1}^{n} B_i\cdot\beta_i \le \sum_{i=1}^{n} B_i\cdot(p_i-1)<$$

$$<\sum_{i=1}^{n} B_i\cdot p_i=\sum_{i=1}^{n} m_i\cdot\frac{P}{p_i}\cdot p_i=\sum_{i=1}^{n} m_i\cdot P=$$

$$=P\cdot(\sum_{i=1}^{n} p_i-n)$$

This shows that

$$rank<(\sum_{i=1}^{n} p_i-n)$$

Consider an auxiliary method to determine the rank of a number.

An additional module is introduced

$$p_{n+1}>\sum_{i=1}^{n} p_i-n$$

From (15) follows:

$$rank=\left(\sum_{i=1}^{n} B_i\cdot\gamma_i - A\right)\cdot P^{-1}$$

then:

$$rank=\left(\sum_{i=1}^{n}|B_i|_{p_{n+1}}\cdot|\gamma_i|_{p_{n+1}}-|A|_{p_{n+1}}\right)\cdot|P^{-1}|_{p_{n+1}} \quad (17)$$

In (17) all calculations are performed modulo $p_{n+1}$.

The diagram for quick rank calculation is shown in Fig. 2.

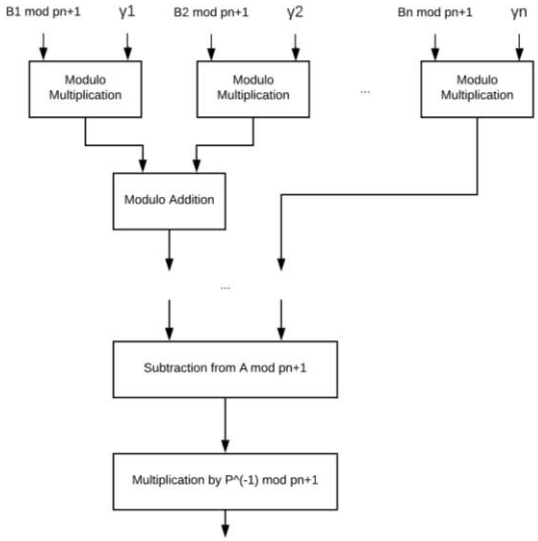


Fig. 2. Diagram for quick rank calculation.

According to the above diagram, the rank value can be calculated by the formula (17) using $\lceil \log_2 n \rceil + 2$ modular operations of addition and multiplication.

Next, a method of converting numbers from RNS to a positional numeral system is considered.

## VII. Converting Numbers from RNS to a Positional Numeral System

Let $A$ be a number in the format (8) to be converted to a positional numeral system in the floating-point format with the mantissa length $n_f$

$A=[(\gamma_1,\gamma_2,...,\gamma_i,...,\gamma_n), t]$

A method for accelerated calculation of the value of expression (15) in a signed-digit numeral system with numbers in the range [-6, ..., 6] is considered.

Let $B_i$, $P$ be constants in the floating-point format with the mantissa length $n_f$.

The expressions

$$j \cdot B_i, j=1..p_i-1, \quad (18)$$

$$j \cdot P, j = 1..(\sum_{i=1}^{n} p_i - n)$$

Are constants and can be stored in computer memory, then the conversion process can be represented in the form of the following diagram (Fig. 3).

According to the diagram, the result of the conversion can be calculated by the formula (15) in $\lceil \log_2 n \rceil + 1$ steps for a known value of *rank*. The adjustment block shown in Fig. 3 normalizes the mantissa of the result. If the mantissa contains more than $n_f$ digits, then the order of the result should be reduced by the number of discarded digits.

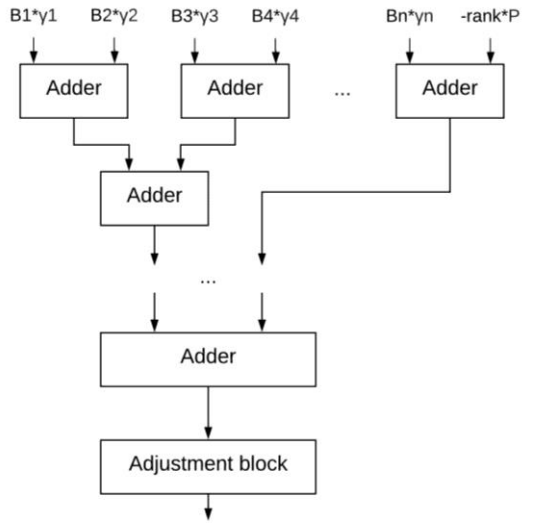


Fig. 3. Diagram for Conversion from RNS to a Positional Numeral System.

## VIII. Example

Given:

nf=2, kf=2, q=10

Two vectors X=(1· 102,5· 10-1,3· 102), Y=(1,2,3)

Then according to (14)

P/2≥3· 104· 108=3· 1012

The chosen moduli are:

p1=2269, p2=2437, p3=2791, p4=3169

P=48907121298487

B1=35737332354734

B2=32129791218251

$B_3$=42756494435080

$B_4$=36097745887397

The result of the scalar product in RNS is

$(X,Y)=[(1736,1586,1214,836), -1]$

After conversion from RNS to the floating-point format with a given length of the mantissa the following can be obtained:

$(X,Y)=40\cdot 10^1$

## IX. Conclusion

This article proposes an approach to speeding up high-precision computing with the following limitations:

*1)* It should be used for tasks with a predetermined number of operations for which the order of the result can be estimated.

*2)* Three modular arithmetic operations are considered: addition, subtraction and multiplication (as well as division by a constant, which can be replaced by multiplication).

*3)* The remaining non-modular operations such as rounding and comparison are difficult to implement in RNS. Therefore, the proposed approach will be effective for the tasks that contain a small number of non-modular operations. In addition to the scalar product, such tasks include, for example, the discrete Fourier transform, iterative solution of linear equation systems by Jacoby, Gaussa-Zeidel methods, solution of the Cauchy problem by Euler method, etc.

*4)* Possible direction of further research is the software implementation of high-precision calculations in RNS arithmetic on GPU.

## REFERENCES

[1] D.H. Bailey, R. Barrio, J.M. Borwein, “High-precision computation: Mathematical physics and dynamics” in Applied Mathematics and Computation, Vol. 218, No. 20, 2012, pp. 10106-10121.

[2] D. H. Bailey, J. M. Borwein, “High-Precision Arithmetic in Mathematical Physics” in Mathematics, 3 (2015), pp. 337–367.

[3] D.H. Bailey, “High-Precision Computation and Mathematical Physics” Lawrence Berkeley National Laboratory, 2009.

[4] Demidova L., Nikulchev E., Sokolova Yu. (2016). BIG DATA classification using the svm classifiers with the modified particle swarm optimization and the svm ensembles. international journal of advanced computer science and applications T.7, №5, P. 294-312.

[5] “Non-obvious features of real numbers” [www.delphikingdom.com/asp/viewitem.asp?catalogid=374]

[6] A.A. Amosov, Y.A. Dubinskiy, N.V. Kopchenkova, Computational Methods for Engineers. Moscow: “High School”, p. 544, 1994.

[7] L. Fousse, G. Hanrot, V. Lefèvre, P. Pélissier, P. Zimmermann, “MPFR: a multiple-precision binary floating-point library with correct rounding” in ACM Transactions on Mathematical Software, Vol. 33, No. 2, Article No. 13, 2007.

[8] MParithm - package for high precision computation, 2015 [www.wolfgang-ehrhardt.de/mp_intro.html]

[9] GNU Scientific Library 2.5 released, 2018 [https://savannah.gnu.org/forum/forum.php?forum_id=9175]

[10] “Operations with multi-bit real numbers of ZReal type” [http://ishodniki.ru/list/index.php?action=name&show=pascal-math&cat=11]

[11] D.H. Bailey, X.S. Li, B. Thompson, “ARPREC: An arbitrary precision computation package” Sep 2002 [http://crd.lbl.gov/~dhbailey/dhbpapers/arprec.pdf].

[12] I.I. Dzegelenok, Sh.A. Otsokov, “Algebraization of numerical representations in providing high-precision supercomputer calculations” in Vestnik MPEI, No 3, 2010, pp. 107-116.

[13] I.Ya. Akushskiy, D.I. Yuditskiy, Machine arithmetic in residual classes. Moscow: Soviet Radio, p. 440, 1968.

[14] R. Graham, D. Knuth, O. Patashnik, Concrete math. Computer Science Foundation. Williams, p. 784, 2015.

[15] A.R. Omondi, B. Premkumar, “Residue Number Systems: Theory and Implementation”, Imperial College Press, 2007.

[16] Magomedov Sh. Organization of secured data transfer in computers using sign-value notation. ITM Web of Conferences. 2017. T. 10. DOI: 10.1051/itmconf/20171004004.

[17] R.A. Solovyev, E.S. Balaka, D.V. Telpukhov, “A device for calculating the scalar product of vectors with error correction based on a system of residual classes” in Problems of developing promising micro- and nanoelectronic systems. Vol. IV, A.L. Stempkovskiy Ed. Moscow: The Institute for Design Problems in Microelectronics RAS, 2014, pp. 173-178.

[18] E. Kinoshita, H. Kosako, Y. Kojima, “Floating-point arithmetic algorithms in the symmetric residue number system” in Computers, IEEE Transactions on, vol. C-23, No. 1, 1974, pp. 9–20.

[19] K. S. Isupov, A. N. Mal’tsev, “A parallel-processing-oriented method for the representation of multi-digit floating-point numbers” in Vychislitel’nyye metody i programmirovaniye, 15:4 (2014), pp. 631–643.

[20] A. Lebedev, S. Magomedov. “A tool for automatic parallelization of affine programs for systems with shared and distributed memory”. Russian Journal of Technology. 2019;7(5):P.7-19. https://doi.org/10.32362/2500-316X-2019-7-5-7-19.