A HIGH-SPEED DIVISION METHOD IN RESIDUE ARITHMETIC

Dilip K. Banerji*, To-Yat Cheung, and V. Ganesan

Department of Computer Science
University of Ottawa
Ottawa, Ontario K1N 6N5

ABSTRACT

This paper is concerned with the operation of division in residue number systems. Residue codes are introduced and the basic residue arithmetic operations are defined. Previous results on residue division are outlined. A well-known integer division algorithm is used and adapted for residue division. A new method is proposed for choosing an approximate divisor, approximate dividends and the partial quotients. The proposed method yields correct quotients faster than the existing methods and is general in its application i.e., it is not restricted by the choice of moduli as long as they are relatively prime.

1. INTRODUCTION

Residue Number Systems (RNS), though known to ancient mathematicians,[1,2] were not exploited for machine computation until quite recently. The characteristics of RNS have been under investigation for use in computer arithmetic since late 1950's.[3-5] The inherent "carry-free" property of residue arithmetic makes the study of RNS attractive from the point of view of high-speed arithmetic unit design. A major drawback of RNS is that the operation of sign detection and the related operations of division, relative magnitude comparison, and additive overflow are complicated and slow. In this paper, we consider the problems associated with the operation of division, and we propose a simple and efficient algorithm to perform division.

1.1 Residue Codes

Let $M = \{m_1, m_2, \ldots, m_n\}$ be an ordered set of positive integers, where $m_i \geq 2$, for $i = 1,2,\ldots,n$. The m_i are called moduli or radices, and the corresponding ordered set $(x_1, x_2, \ldots, x_n)$ of least positive residues of a natural number X, with respect to the moduli, forms the residue code of X. The residue code of X with respect to m_i is denoted by $|X|_{m_i} = x_i$. For example, if $M = \{2,3,5\}$ and X=14, then $|14|_2=0$, $|14|_3=2$, and $|14|_5=4$. Thus, residue code for 14 in this RNS is (0,2,4).

In order to avoid redundancy (unless redundancy is desirable for some reason) the moduli of a residue system must be pair-wise relatively prime, that is, the greatest common divisor of each pair of moduli must be 1. In this case, the number of integers that can be uniquely coded is given by $M = \prod_{i=1}^{n} m_i$.[7] In the example, therefore, a total of 30 integers can be represented uniquely. These can correspond to the numbers 0 through 29.

1.2 Residue Arithmetic

Let $(x_1,x_2,\ldots,x_n)$ and $(y_1,y_2,\ldots,y_n)$ denote the residue codes for X and Y respectively in the RNS defined by M. Let $(z_1,z_2,\ldots,z_n)$ denote the residue code for Z, the result of an arithmetic operation on X and Y.

Addition or Multiplication:

The ith digit z_i of the result is given by $z_i = |x_i * y_i|_{m_i}$, where * denotes either addition or multiplication. The fact that z_i depends on x_i & y_i only, implies the absence of any carry from one residue digit position to another, and thus addition or multiplication with respect to different moduli can be performed in parallel. Furthermore, multiplication does not require any partial products.

---

* School of Computer & Systems Science, Jawaharlal Nehru University, New Delhi - 110067.
Currently, Visiting Professor, Department of Computing and Information Science, University of Guelph, Guelph, Ontario N1G 2W1.
This work was supported by the National Science and Engineering Research Council of Canada, Grant #A8306.

CH1630-3/81/0000/0158$00.75 © 1981 IEEE

Subtraction:

Subtraction in RNS is defined as:
$z_i = |x_i - y_i|_{m_i} = |x_i + y'_i|_{m_i}$ where $y'_i = m_i - y_i$.
Thus subtraction is also a carry-free operation.

Division:

This is one of the most difficult operations in residue arithmetic and forms the subject of this paper; it will be discussed later on.

1.3 Mixed Radix Conversion and Base Extension

Mixed radix conversion is a very useful process for performing division and is briefly discussed here. It converts a residue number into its representation in a mixed radix system, where the positional weights are products of moduli.

A number X can be represented in its mixed radix form as: $X = r_n m_1 m_2 \ldots m_{n-1} + \ldots + r_2 m_1 + r_1$ (1.1)
where r_i are called the mixed radix digits, and $0 \le r_i < m_i$. The weight associated with any mixed radix digit r_i is $m_1 m_2 \ldots m_{i-1}$ (note $m_0 = 1$). Such a system has the same range of representation as a residue system $M = \{m_1, m_2, \ldots m_n\}$.[7] We note that $|X|_{m_1} = x_1 = r_1$. Hence, the first mixed radix digit is the same as the first residue digit.
$r_2 = \left|\frac{X - r_1}{m_1}\right|_{m_2}$ from eq. (1.1). Division by m_1 is actually multiplication by the multiplicative inverse of m_1 with respect to m_2. By successively subtracting r_i and dividing by m_i, all the mixed radix digits can be computed.

Example 1.1: Let $M = \{2,3,5\}$ and $X = 7$. The corresponding mixed-radix expression is:
$X = r_3(2 \times 3) + r_2(2) + r_1$. We now find the mixed-radix digits as follows:

	Moduli: 2	3	5
Residue Representation of X	$1 = r_1$	1	2
Subtract $r_1 = 1$	1	1	1
$X - r_1$	0	0	1
multiply by $\left\|\frac{1}{2}\right\|_{m_i}$		2	3
$\frac{X - r_1}{2}$	$0 = r_2$	3	
Subtract $r_2 = 0$	0	0	
$\frac{X - r_1}{2} - r_2$	0	3	
multiply by $\left\|\frac{1}{3}\right\|_{m_i}$		2	
$\frac{\frac{X - r_1}{2} - r_2}{3}$		$1 = r_3$	

Hence the mixed-radix representation of X is <1,0,1> and one obtains $X = 1(2 \times 3) + 0(2) + 1 = 7$.

Extension of Base

It is frequently necessary to find the residue digits for a new set of moduli, given the residue digits relative to another set of moduli. Usually, the new base will be an extension of the original base. The procedure consists of one mixed-radix conversion with an additional final step. For $M = \{m_1, m_2, \ldots, m_n\}$, the range of definition is 0 to $\prod_{i=1}^{n} m_i - 1$. If another modulus m_{n+1} is included, the range of definition becomes 0 to $\prod_{i=1}^{n+1} m_i - 1$, and the mixed-radix expression will be of the form:

$$X = r_{n+1} \prod_{i=1}^{n} m_i + r_n \prod_{i=1}^{n-1} m_i + \ldots + r_2 m_1 + r_1.$$

In performing mixed-radix conversion to find $|X|_{m_{n+1}}$ we use the fact that for any number in the original interval of definition, r_{n+1} will be equal to zero.[7]

2. DIVISION OPERATION

In this section, we consider the problem of division in RNS. Since a RNS is not a weighted number system, the operation of division which involves magnitude comparison of two operands is not straightforward. We first discuss the existing methods of division and their limitations, followed by our proposal method discussed in Sec. 3.

2.1 Categorization of Division

The operation of division in RNS can be categorized into three distinct types.

Category 1: Division Remainder Zero

In this category[7] the dividend is known to be an integer multiple of the divisor and the divisor is known to be relatively prime to M. This category is of restricted use, since it must be known a priori whether its prerequisites are satisfied in order to perform the operation. For this algorithm the following theorem applies.

Theorem

If y divides x without remainder and the G.C.D. of y and m_i is 1, then

$$\left|\frac{x}{y}\right|_{m_i} = \left|\frac{1}{y} x\right|_{m_i} \quad (2.1)$$

for all m_i, where $\left|\frac{1}{y}\right|_{m_i}$ is the multiplicative inverse of y modulo m_i. If y does not divide x, the quantity $\frac{x}{y}$ is not an integer, and $\left|\frac{x}{y}\right|_{m_i}$ is not defined. Consequently, (2.1) has no meaning.

Cateogry 2: Scaling

In this category[7], the dividend is arbitrary and the divisor is any factor of M which is a product of the first powers of some of the moduli. This division is analogous to division by a power of 2 in binary arithmetic, in the sense that division by the defined restricted set of numbers is faster than that by an arbitrary divisor. Division in any integer number system is defined by $x = [\frac{x}{y}]y + |x|_y$, where x is the dividend, y the divisor, $[\frac{x}{y}]$ the integer value of x over y (quotient), and $|x|_y$ is the (least positive integer) remainder. The object of the scaling algorithm is to find $[\frac{x}{y}]$ for restricted y values. We note that: $[\frac{x}{y}] = \frac{x-|x|_y}{y}$. Hence the residue representation of $[\frac{x}{y}]$ is $(\left|\frac{x-|x|_y}{y}\right|_{m_1}, \left|\frac{x-|x|_y}{y}\right|_{m_2}, \ldots, \left|\frac{x-|x|_y}{y}\right|_{m_n})$, where the values of $\left|\frac{x-|x|_y}{y}\right|_{m_i}$ are integers. If y is one of the m_i's or the product of the first powers of some of the moduli m_i, then $|x|_y$ can be found. Then by the theorem used in division remainder zero category, for all i for which the G.C.D. of m_i and y is 1, one obtains:

$$\left|\frac{x-|x|_y}{y}\right|_{m_i} = [\frac{x}{y}]_{m_i} = \left|\frac{1}{y}\cdot(x-|x|_y)\right|_{m_i}.$$

This equation expresses the residue digits of $[\frac{x}{y}]$ for all digits for which the G.C.D. of m_i and y is 1. The remaining digits may be found by the base extension method. Thus the scaling algorithm consists of two steps: 1) Division Remainder Zero, and 2) Base extension.

Category 3: General Division

Both Category 1 and Category 2 are concerned with special cases and are not applicable when both the dividend and the divisor are arbitrary integers; Category 3 represents the latter case.

A method for approximating the quotient by another integer has been described in literature.[7] This method consists of choosing some product of moduli as an approximation for the divisor, and applying the scaling algorithm to obtain either the greatest integer less than or equal to the quotient, or the integer nearest to the quotient (that is, the quotient rounded-off to the nearest integer). Since the approximated divisor $\bar{y}$ is not equal to the given divisor y, an error is introduced in the quotient which is iteratively reduced to zero. The basic assumptions made are that both the dividend x and the divisor y are positive and that a value for $\bar{y}$ can be found such that $y \le \bar{y} < 2y$, where $\bar{y}$ is a permissible divisor under the scaling algorithm.

The first step in the algorithm is to compute, by the scaling algorithm, $z_1 = [\frac{x}{\bar{y}}]$. Once z_1 is found, this quantity is used in the recursive relationships: $x_i = x_{i-1} - yz_i$, $x_0 = x$, and $z_i = [\frac{x_{i-1}}{\bar{y}}]$ to obtain z_2, z_3, ..., etc. This iterative procedure is continued until either $z_i = 0$, or $x_i = 0$. If this occurs on the rth iteration, then

$$z = [\frac{x}{y}] = \sum_{i=1}^{r-1} z_i + z'_r \qquad (2.4)$$

$$z'_r = \begin{cases} z_r & \text{if } z_r \neq 0 \text{ and } x_r = 0 \qquad (2.5) \\ 1 & \text{if } z_r = 0 \text{ and } x_{r-1} \geq y \text{ for any } \bar{y} \neq y \qquad (2.6) \\ 0 & \text{otherwise} \qquad (2.7) \end{cases}$$

The validity of this algorithm hinges on three premises:

1. Either z_i or x_i becomes zero after a finite number of iterations.
2. The series $\sum_{i=1}^{r-1} z_i + z'_r$ must be equal to $[\frac{x}{y}]$.
3. For every y there exists a $\bar{y}$ and this $\bar{y}$ can be found.

Due to limitations of space, we will not consider an example showing the application of this method. The main drawbacks of this method are:

1) It requires a table look-up where the size of the table for n moduli is $2^n - 2$.
2) A $\bar{y}$ must be found such that $y \le \bar{y} < 2y$. This requirement cannot be satisfied for every set of moduli; for example, for the RNS with $m_1 = 9$, $m_2 = 11$, the condition $y \le \bar{y} < 2y$ is not satisfied for y=4.

This method will henceforth be referred to as the Tanaka algorithm.

THE PROPOSED METHODS

We now propose two methods which overcome the drawbacks of the Tanaka algorithm. The major advantages of these two methods are:

1) They are more general, that is, the approximate divisor need not be a product of the moduli and the relation $y \le \bar{y} < 2y$ is automatically satisfied.
2) They are faster than the existing methods.

The mathematical proofs for our methods are given in the Appendix; for the second method, only the first two termination conditions are proved (the third termination condition has not been found).

3.1 Proposed Method 1

In this method, we start with both an approximate dividend and an approximate divisor. Since, in the general case, the dividend and the divisor are not equal to our approximate dividend and approximate divisor, respectively, an error is introduced in the quotient. This error is then iteratively reduced to zero.

3.1.1 The Algorithm

Let x_{i-1} be the dividend and y be the divisor in the ith iteration, i=1,2,... .

Approximate Dividend: Let x_{i-1} be denoted in its mixed radix representation as:

$x_{i-1} \leftrightarrow <0,0,\ldots,\alpha_k,\alpha_{k-1},\cdots,\alpha_1>$ where α_k is the most significant non-zero mixed radix coefficient of x_{i-1}. The approximate dividend $\bar{x}_{i-1}$ is then chosen to be: $\bar{x}_{i-1}=\alpha_k m_1 m_2 \cdots m_{k-1}$.

Approximate Divisor: If the mixed radix digits of y are assumed to be: $y \leftrightarrow <0,0,\ldots,\beta_\ell,\beta_{\ell-1},\ldots,\beta_1>$, where β_ℓ is the most significant non-zero mixed radix coefficient of y, then the approximate divisor is chosen to be $\bar{y}=(\beta_\ell+1)m_1 m_2 \cdots m_{\ell-1}$.

Approximate Quotient: The approximate quotient z_i in the ith iteration is determined by the interrelationship between k and ℓ.

Case 1: k=ℓ

In this case, $$z_i = \frac{\bar{x}_{i-1}}{\bar{y}} = \frac{\alpha_k m_1 m_2 \cdots m_{\ell-1}}{(\beta_\ell+1) m_1 m_2 \cdots m_{\ell-1}} = \frac{\alpha_k}{(\beta_\ell+1)}$$

Case 2: k=ℓ+1

$$z_i = \frac{\bar{x}_{i-1}}{\bar{y}} = \frac{\alpha_k m_1 m_2 \cdots m_{\ell-1} m_\ell}{(\beta_\ell+1) m_1 m_2 \cdots m_{\ell-1}} = \alpha_k \frac{m_\ell}{(\beta_\ell+1)}$$

Case 3: k>ℓ+1

$$z_i = \frac{\bar{x}_{i-1}}{\bar{y}} = \alpha_k \frac{m_\ell m_{\ell+1} m_{\ell+2} \cdots m_{k-1}}{(\beta_\ell+1)}$$

Recursive Relationship: We use the approximate quotient z_i in the following recursive relationships to iteratively reduce the error in computing the quotient.

$$x_i = x_{i-1} - y z_i, \quad x_0 = x \text{ (given dividend)} \qquad (3.1)$$

and, $$z_{i+1} = \left[\frac{\bar{x}_i}{\bar{y}}\right] \qquad (3.2)$$

We first find z_1, and use relations (3.1) and (3.2) repeatedly to obtain $z_2, z_3, \ldots$, etc. This iterative procedure is continued until either $z_i=0$, or $x_i=0$. If this occurs in the rth iteration, then relation (2.4) and conditions (2.5)-(2.7) can be used to obtain the quotient z.

3.2 Table Look-ups

We need two types of table look-ups to perform our algorithms. One is of the form $m_{\ell+1} m_{\ell+2} \cdots m_{k-1}$ and the other is of the form

$$\frac{m_\ell}{(\beta_\ell+1)}, \quad \frac{\alpha_k}{(\beta_\ell+1)}, \quad \frac{m_\ell}{\beta_\ell}, \quad \frac{\alpha_k}{\beta_\ell} .$$

3.2.1 Table Look-ups of the Form $m_{\ell+1} m_{\ell+2} \cdots m_{k-1}$

For k>ℓ+1, (ℓ=1,2,...,n-2) and (k=ℓ+2,ℓ+3,...,n), there are $(n-2)\frac{(n-1)}{2}$ integers of the form $m_{\ell+1} m_{\ell+2} \cdots m_{k-1}$. All such products of moduli are stored in a table and for a given value of k and ℓ, the corresponding product is read out.

3.2.2 Table Look-ups of the Form $\frac{m_\ell}{(\beta_\ell+1)}, \frac{\alpha_k}{(\beta_\ell+1)}, \frac{m_\ell}{\beta_\ell}$ and $\frac{\alpha_k}{\beta_\ell}$

Normally, such a table would contain $m_{max} \times m_{max}$ entries where m_{max} is the largest modulus. However, the table would contain 0 entries for the cases $\alpha_k < \beta_\ell+1$, and $\alpha_k < \beta_\ell$. These 0 entries need not be stored, reducing the number of entries in the table to $\frac{m_{max}(m_{max}+1)}{2}$.

It should be noted that the proposed algorithm computes quotients that are exact. Furthermore, for the examples considered, the total number of operations required is, on the average, less than that required by the Tanaka algorithm. Approximately 500 residue division operations were performed using randomly selected dividends and divisors with 10 different sets of moduli. For each set of selected moduli, 50 dividend-divisor pairs were chosen and it was found that our algorithm is not only faster, but it also computes the quotient without any error for the entire set of 500 division operations. Over the 500 experiments, 18,209 basic operations were required by our algorithm and 21,000 by the Tanaka method. Thus the average saving is (21,000-18,209)/500=5.82 operations per division. Another advantage of our method is that it is applicable to any set of moduli, as opposed to the Tanaka method which must satisfy the condition $y \le \bar{y} < 2y$. The method for choosing the approximate divisor $\bar{y}$ in the Tanaka algorithm does not assume that this condition would be satisfied for all moduli, whereas our procedure for selecting it automatically ensures that this condition is satisfied.

3.3 Speed-up Procedure (Method 2)

A modification to Method 1 further reduces the number of basic operations required, thus providing a further speed-up compared to the Tanaka algorithm. This procedure is essentially

similar to Method 1. The difference lies in the selection of the approximate divisor and in the computation of z_r' (see relation 2.4). The approximate divisor for Method 2 is chosen as: $\bar{y}=\beta_\ell m_1 m_2 \cdots m_{\ell-1}$, where β_ℓ is the most significant non-zero mixed radix digit of y. The stopping conditions and the corresponding values of z_r' are given as follows:

1. If $z_r=0$ and $x_{r-1}\geq y$, then $z_r'=1$
2. If $z_r=0$ and $y>x_{r-1}\geq 0$, then $z_r'=0$

When a third condition, $z_r\neq 0$ and $x_r<0$, with $x_{r-1}>0$ is encountered, we were unable to obtain a suitable correction factor z_r'. It was found empirically that for some set of moduli, setting z_r' to -1 when this condition is encountered, would produce the correct quotient. It is, however, suggested that Method 1 be used when this condition is encountered in order to ensure that the computed quotient is correct.

For the set of 500 randomly selected operands and different sets of moduli, Method 2 resulted in an average saving of 8.14 basic operations as compared to the Tanaka algorithm; compared to Method 1, there was an average saving of 2.32 basic operations.

For the residue system

$$M=\{23,19,17,13,11,7,5,3,2\}$$

which is a particularly "good" system for the Tanaka algorithm since the condition $y\leq\bar{y}<2y$ is satisfied for all possible divisors, it was found that Method 2 computes the quotients (for the 50 cases tried) without any error; the Tanaka algorithm produced an average error of 29 per cent for the same cases.

CONCLUSION

We have proposed two methods for division that are, in general, faster than the existing methods. Furthermore, our algorithms are not constrained by the set of moduli as long as they are relatively prime.

APPENDIX

We provide here the complete mathematical proof for Method 1 and partial proof for Method 2. For Method 2 we are unable to obtain analytically the correction factor for termination condition (2.7). We want to show that the iterations

$$x_i=x_{i-1}-yz_i \quad \text{where} \quad x_0=x \tag{3.1}$$

and, $$z_i=\left[\frac{\bar{x}_{i-1}}{\bar{y}}\right] \tag{3.2}$$

converge to yield the quotient $z=\left[\frac{x}{y}\right]=\sum_{i=1}^{r-1} z_i+z_r'$, where z_r' is obtained using conditions (2.5)-(2.7).

The first part of the proof consists of showing that either z_i or x_i becomes zero after a finite number of steps. Let us suppose $z_i\neq 0$ for all i. Then we would how that $x_i=0$ for some i. This follows from the fact that a) $x_i\geq 0$ for all i, and b) $x_0,x_1,\ldots,x_{i-1},\ldots$ is a strictly decreasing sequence of integers.

We will prove a) by mathematical induction. We note that $x_0=x>0$ (by assumption). Now, let us assume that $x_{i-1}\geq 0$. Then from (3.1) we get

$x_i=x_{i-1}-y\left[\frac{\bar{x}_{i-1}}{\bar{y}}\right]\geq x_{i-1}-\bar{y}\frac{\bar{x}_{i-1}}{\bar{y}}$, since $y<\bar{y}$ (proved later) and $[I]\leq I$ for any I. Hence $x_i\geq x_{i-1}-\bar{x}_{i-1}\geq 0$, which proves assertion a).

To prove assertion b), we note that $x_i=x_{i-1}-yz_i<x_{i-1}$, since $y>0$ and $z_i\neq 0$ (by assumption). We have $\bar{x}_{i-1}\geq 0$ and $\bar{y}>0$. Therefore, $z_i=\left[\frac{\bar{x}_{i-1}}{\bar{y}}\right]\geq 0$. But $z_i\neq 0$ by assumption. Hence, $z_i>0$. Therefore, $x_1<x_0, x_2<x_1,\ldots,x_i<x_{i-1}$ i.e., $x_0,x_1,x_2,\ldots,x_i,\ldots$ is a strictly decreasing sequence of integers. This proves that $x_i=0$ for some i.

Since $x_0,x_1,\ldots,x_i,\ldots$ is a strictly decreasing sequence, it is clear that for some i, $x_{i-1}<\bar{y}$. Since $\bar{x}_{i-1}\leq x_{i-1}$, it is obvious that $z_i=\left[\frac{\bar{x}_{i-1}}{\bar{y}}\right]=0$. Therefore, either x_i or z_i becomes zero after a finite number of iterations.

Next, we will show that $\sum_{i=1}^{r-1} z_i+z_r'=\left[\frac{x}{y}\right]$.

From (3.1), we have: $x_1=x_0-yz_1$, $x_2=x_1-yz_2,\ldots$, $x_{r-1}=x_{r-2}-yz_{r-1}$, $x_r=x_{r-1}-yz_r$. Adding, we have: $x_r=x_0-y\left(\sum_{i=1}^{r-1} z_i+z_r\right)$. Since $x_0=x$, we get:

$\frac{x}{y}=\sum_{i=1}^{r-1} z_i+z_r+\frac{x_r}{y}$. Hence $\left[\frac{x}{y}\right]=\sum_{i=1}^{r-1} z_i+z_r+\left[\frac{x_r}{y}\right]$ $=\sum_{i=1}^{r-1} z_i+z_r'$, where $z_r'=z_r+\left[\frac{x_r}{y}\right]$. To complete the proof, we will show that z_r' can be obtained using conditions (2.5)-(2.7).

Case 1 $z_r\neq 0$ and $x_r=0$ (2.5)

Then $\left[\frac{x}{y}\right]=\sum_{i=1}^{r-1} z_i+z_r$ i.e., $z_r'=z_r$.

Case 2 $z_r=0$ and $x_{r-1}\geq y$ (2.6)

Since $z_r=0$, we have $[\frac{\bar{x}_{r-1}}{\bar{y}}]=0$ or $\frac{\bar{x}_{r-1}}{\bar{y}}<1$. Therefore, $\bar{x}_{r-1}<\bar{y}$. Now, we will show that $\bar{x}_{r-1}<\bar{y}$ and $x_{r-1}\geq y$ implies that $x_r<2y$. Let

$x_{r-1}=\alpha_j m_1 m_2\cdots m_{j-1}+\alpha_{j-1} m_1 m_2\cdots m_{j-2}+\ldots+\alpha_2 m_1+\alpha_1$

$=\alpha_j m_1 m_2\cdots m_{j-1}+\bar{\alpha}_j$ where,

$\bar{\alpha}_j=\alpha_{j-1} m_1 m_2\cdots m_{j-2}+\ldots+\alpha_2 m_1+\alpha_1$. Similarly, let $y=\beta_k m_1 m_2\cdots m_{k-1}+\bar{\beta}_k$, where $\bar{\beta}_k=\beta_{k-1} m_1 m_2\cdots m_{k-2}+\ldots+\beta_2 m_1+\beta_1$.

First, we will show that $j=k$. Since $x_{r-1}\geq y$, we have $j\geq k$. Suppose, $j=k+1$. Since $\bar{x}_{r-1}<\bar{y}$, we get $\alpha_j m_1 m_2\cdots m_{j-1}<(\beta_k+1) m_1 m_2\cdots m_{k-1}$ or $\alpha_{k+1} m_1 m_2\cdots m_k<(\beta_k+1) m_1 m_2\cdots m_{k-1}$. Therefore,

$\alpha_{k+1}\cdot m_k<1\cdot(\beta_k+1)$ (A1)

But, $\alpha_{k+1}\geq 1$ since it represents the most significant non-zero mixed radix digit of x_{r-1}. Also, $m_k\geq\beta_k+1$ since $0\leq\beta_k\leq m_k-1$. Therefore, $\alpha_{k+1}\cdot m_k\geq 1\cdot(\beta_k+1)$, contradictory to (A1).

Similarly, we can show that $j=k+n$ leads to contradiction for any $n>1$. Hence, $j=k$.

Next, we shall show that $\alpha_k=\beta_k$. $\bar{x}_{r-1}<\bar{y}$ implies $\alpha_k m_1 m_2\cdots m_{k-1}<(\beta_k+1) m_1\cdots m_{k-1}$.

Therefore, $\alpha_k<\beta_k+1$. (A2)

Suppose $\alpha_k=\beta_k-i$, $i=1,2,\ldots$. $x_{r-1}\geq y$ implies $\alpha_k m_1 m_2\cdots m_{k-1}+\bar{\alpha}_k\geq\beta_k m_1 m_2\cdots m_{k-1}+\bar{\beta}_k$ or

$(\beta_k-i) m_1 m_2\cdots m_{k-1}+\bar{\alpha}_k\geq\beta_k m_1 m_2\cdots m_{k-1}+\bar{\beta}_k$ or

$-i m_1 m_2\cdots m_{k-1}+\bar{\alpha}_k\geq\bar{\beta}_k$ (A3)

We have $\bar{\alpha}_k=\alpha_{k-1} m_1 m_2\cdots m_{k-2}+\ldots+\alpha_2 m_1+\alpha_1$

$\leq(m_{k-1}-1) m_1 m_2\cdots m_{k-2}+\ldots+(m_2-1) m_1+m_1-1$

$=m_1 m_2\cdots m_{k-1}-1$. Hence $\bar{\alpha}_k<m_1 m_2\cdots m_{k-1}$.

This means the left-hand side of relation (A3) is less than zero. However, the right-hand side of A(3), $\bar{\beta}_k$ is nonnegative. Therefore, relation (A3) is impossible. Hence, $\alpha_k\geq\beta_k$. This together with relation (A2) implies that $\alpha_k=\beta_k$.

We now have $x_{r-1}=\alpha_k m_1\cdots m_{k-1}+\bar{\alpha}_k$ and $2y=2\beta_k m_1\cdots m_{k-1}+2\bar{\beta}_k=2\alpha_k m_1\cdots m_{k-1}+2\bar{\beta}_k$. But $\bar{\alpha}_k<m_1 m_2\cdots m_{k-1}$. Hence $x_{r-1}<2y$. Since $z_r=0$, (by assumption), $x_r=x_{r-1}$. Therefore, we get $y\leq x_r<2y$ or $1\leq\frac{x_r}{y}<2$. Hence $[\frac{x_r}{y}]=1=z_r'$, which shows the validity of condition (2.6).

Case 3 $z_r=0$, and $x_{r-1}<y$.

These conditions imply $z_r'=0+[\frac{x_{r-1}}{y}]=0$. This proves the validity of condition (2.7).

To prove $y<\bar{y}<2y$

$y=\beta_k m_1\cdots m_{k-1}+\ldots+\beta_2 m_1+\beta_1$; $\bar{y}=(\beta_k+1) m_1\cdots m_{k-1}$; $2y=2\beta_k m_1\cdots m_{k-1}+\ldots+2\beta_2 m_1+2\beta_1$. Therefore, $2y-\bar{y}=(\beta_k-1) m_1 m_2\cdots m_{k-1}+\ldots+2\beta_2 m_1+2\beta_1$. Except for the case $\beta_k=1$ and $\beta_i=0$, for $1\leq i\leq k-1$, we have $2y-\bar{y}>0$ or $\bar{y}<2y$. The case $\beta_k=1$ and $\beta_i=0$, $1\leq i\leq k-1$, represents a special divisor (product of moduli). In this case the scaling method should be used to obtain the quotient.

$\bar{y}=\beta_k m_1 m_2\cdots m_{k-1}+m_1 m_2\cdots m_{k-1}$; $y=\beta_k m_1 m_2\cdots m_{k-1}+\bar{\beta}_k$. It is easily shown (as in the case of $\bar{\alpha}_k$) that $\bar{\beta}_k<m_1 m_2\cdots m_{k-1}$. Hence, $y<\bar{y}$.

For Method 2, it can also be shown that $x_0, x_1, \ldots, x_i, \ldots$ is a decreasing sequence. Therefore, for some i, $x_{i-1}<\bar{y}$ or $z_i=0$.

Case 1: $z_r=0$ and $x_{r-1}\geq y$ for some r.

Then $z_r=[\frac{\bar{x}_{r-1}}{\bar{y}}]=0 \Rightarrow \bar{x}_{r-1}<\bar{y}\leq y$. Also $x_r=x_{r-1}\leq 2\bar{x}_{r-1}<2y$. Hence, we have $y\leq x_{r-1}=x_r<2y$; or $1\leq\frac{x_r}{y}<2$. Therefore, $z_r'=0+[\frac{x_r}{y}]=1$.

Case 2: $z_r=0$ and $y>x_{r-1}\geq 0$

Then $x_r=x_{r-1}<y$, or $[\frac{x_r}{y}]=0$. Hence $z_r'=0$.

Case 3: $z_r\neq 0$, $x_r<0$, $x_{r-1}>0$

We are unable to analytically obtain a value for z_r' for this case.

BIBLIOGRAPHY

1. Deskins, W.E., "Abstract Algebra", Collier-MacMillan, 1964.

2. Hardy, G.H., and Wright, E.M., "An Introduction to the Theory of Numbers". Oxford Press, 1954.

3. Aiken, H., and Semon, W., "Advanced Digital Computer Logic", Report No. WADC TR-59-472, Computing Lab., Harvard University, July 1959.

4. Garner, H.L., "The Residue Number System", IRE Transactions on Electronic Computers, Vol. EC-8, June 1959, pp. 140-147.

5. Svoboda, A., "The Numerical Systems of Residual Classes (SRC)", Digital Information Processors, Walter Hoffman (Ed.), John Wiley, 1962.

6. Peterson, W.W., "Error-Correcting Codes", MIT Press, 1961.

7. Szabo, N.S., and Tanaka, R.I., "Residue Arithmetic and Its Application to Computer Technology", McGraw-Hill, 1967.

8. Avizienis, A., "Design of Fault-Tolerant Computers", Proceedings of AFIPS 1967 FJCC, Vol. 31, Thompson Books, Washington, D.C., pp. 733-743.

9. Baugh, R.A. and Day, E.C., "Electronic Sign Evaluator for Residue Number System", RCA Aerospace Communications & Control Division, TR-60-597-32, Camden, N.J., and Burlington, Mass., Jan. 1961.

10. Eastman, W.L., "Sign Detection in a Modular Number System", Proceedings of Harvard Symposium on Digital Computers and their Applications, 3-6 April 1961, pp. 136-162, Harvard University Press, Cambridge, Mass., 1962.

11. Tanaka, R.I., et al., "Interim Technical Report on Modular Arithmetic Techniques", Lockhead Missiles & Space Company; 2-38-61-1, ASD TR61-472, Sunnyvale, Calif., June 1961.

12. Garner, H.L., et al., "Residue Number Systems for Computers", University of Michigan, Information Systems Laboratory., ASD TR61-483, Ann Arbor, Mich., October 1961.

13. Banerji, D.K. and Brzozowski, J.A., "Sign Detection in Residue Number Systems", IEEE Transactions on Computers, Vol. C-18, April 1969, pp. 313-320.

14. Kleir, Y.A., Cheney, P.W., and Tannenbaum, M., "Division and Overflow Detection in Residue Number Systems", IRE Trans. Electron. Computers, Vol. EC-11, No. 4, August 1962.

15. Levine, M., Marx, J., and Levy, S., "New Techniques in Residue Arithmetic", Conference Proceedings of the 4th National Convention on Military Electronics, 1960.

16. Banerji, D.K., "Residue Arithmetic in Computer Design", Ph.D. Thesis, Department of Applied Analysis and Computer Science, University of Waterloo, March 1971.

17. Kinoshita, E., Kosako, H., and Kojima, Y., "General Division in the Symmetric Residue Number Systems", IEEE Transactions on Computers, Vol. C-22, No. 2, Feb. 1973, pp. 134-142.