A High Speed Division Method in Residue Arithmetic

by

Vasudevan Ganesan

Submitted to the School of Graduate Studies
in partial fulfillment for the requirements of
the degree of Master of Applied Science.

Department of Electrical Engineering
Faculty of Science and Engineering
University of Ottawa
Ottawa, Ontario

June 1978

ABSTRACT

This thesis is concerned with the operation of division in residue (modular) number systems. Theoretically, residue arithmetic offers the possibility of "carry-free" arithmetic as far as the operations of addition, subtraction and multiplication are concerned. Since a residue number system is not a weighted number system, the operations of division and sign detection are not simple. Therefore, in special purpose applications, where addition, subtraction and multiplication operations constitute a major part of the computations, residue arithmetic can potentially offer a fairly high speed gain over weighted arithmetic.

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.

VITA

Name: Vasudevan Ganesan

Date of Birth: April 24, 1948

Place of Birth: India

Education: B.E.(Electronics & Communication Eng.)(1971)
M.Sc.(Eng) (Applied Electronics Eng.) (1973)
University of Madras, Madras, India

ACKNOWLEDGEMENTS

The author wishes to express his very sincere thanks to Professor D.K. Banerji for his supervision during the course of this work. Thanks are also due to Professor T.Y. Cheung for his interest and many helpful suggestions. The author is particularly grateful to Claudette Henderson for her speedy and excellent typing assistance. The financial assistance of the National Research Council of Canada, under grant #A8306, is acknowledged.

LIST OF SYMBOLS

GCD	Greatest Common Divisor
[I]	Smallest Integer ≥ I
LCM	Least Common Multiple
m	Modulus
$M=\{m_1,m_2,m_n\}$	Set of moduli
m_i	i^{th} modulus in M
M	Product of moduli
n	Number of moduli
$\|x\|_{m_i}=x_i$	Least positive residue of x modulo m_i
$(x)_M=(x_1,\ldots x_n)$	Residue Code for x
$<x>=<x_1,\ldots,x_n>$	Mixed Radix Digits for x

INDEX

CHAPTER 1

BASIC CONCEPTS

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 [3,4,5].

RNS received considerable attention in the study of error detecting and correcting codes in computers based on a weighted number systems, since it has been shown by Peterson [6] that every separate checking code is a residue code. More references on this subject can be found in [7]. This aspect of RNS has found applications in the JPL STAR computer project [8]. However, the subject of error checking using RNS is outside the scope of this thesis.

There is a very important property which is inherent in the structure of RNS. The property is that in the operations of addition, subtraction, and multiplication, any digit of the resultant is dependent solely on the corresponding digits of the operands. This results in the elimination of any carry from one residue digit position to another in all the three operations. Furthermore, it removes the need for partial products in multiplication and hence multiplication can be performed in essentially the same amount of time as required for addition. It is this inherent "carry-free" property of residue arithmetic that makes the study of RNS attrac-

tive 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 thesis, we consider the problems associated with the operation of division, and we propose a simple and efficient algorithm to perform division. We begin by discussing the fundamental notions of residue codes and residue arithmetic.

1.1 FUNDAMENTALS

We now introduce residue codes. The description is mainly based on [4,5,7].

The number theoretical concept of congruences is the basis for residue codes.

Definition 1.1.1: If a, b, and m are integers, and $m>0$, then a is congruent to b modulo m, written as $a \equiv b \bmod m$, if and only if m divides (a-b), i.e., $a-b=km$ or $a=b+km$, where k is some integer. b is called a residue of a with respect to m, and m is called the modulus.

Definition 1.1.2: If, in the relation $a=b+km$, b is such that $0 \leq b < m$, then b is called the least positive residue of a, modulo m. We denote this by $|a|_m$, or a mod m.

The least positive residues of a number with respect to different moduli are used to represent the number in the residue system defined by the moduli.

For further properties of congruence the reader is referred

to the literature [1,2,7].

1.2 RESIDUE CODES

Consider an ordered set of moduli $M = \{m_1, m_2, \ldots, m_n\}$ instead of just one modulus m. The corresponding ordered n-typle $(x_1, x_2, \ldots, x_n)$ of least positive residues of a number X with respect to the moduli forms the residue representation or residue code for X.

Example 1.2.1: The residue representation for 9 in the system $M = \{2,3,5\}$ is obtained as follows:

$$9 = 4 \times 2 + 1$$

$$9 = 3 \times 3 + 0$$

$$9 = 1 \times 5 + 4$$

1, 0 and 4 are the least positive residues of 9 with respect to the moduli 2, 3, and 5 respectively. Therefore, the residue representation for 9 in this residue system is (1, 0, 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 i.e., the greatest common divisor of each pair of moduli must be unity. If this is so, then the number of integers that can be uniquely coded in a system defined by $\{m_1, \ldots, m_n\}$ equals the product $M = \prod_{i=1}^{n} m_i$. This is a direct consequence of the Chinese Remainder Theorem [1,3,4,7]. In Example 1.2.1, therefore, a total of 30 integers can be represented uniquely. These correspond to the numbers 0 through 29.

When the moduli are not pair-wise relatively prime, the total

number of unique representations is not equal to their product. Instead, it equals the LCM of the moduli [7], where LCM denotes the least common multiple.

1.3 RESIDUE AIRTHMETIC

Let $(x_1,x_2,\ldots,x_n)$ and $(y_1,y_2,\ldots,y_n)$ denote the residue codes for two numbers X and Y respectively in the residue system of moduli $\{m_1,m_2,\ldots,m_n\}$. Let $(z_1,z_2,\ldots,z_n)$ denote the residue code for Z, the result of an arithmetic operation on X and Y.

Rules for Residue Arithmetic

This discussion is mainly based on [7]. The basic properties are given here without proof. For details the reader is referred to [7].

I. Residue of Multiples of m: For any integers k and a, the following applies:

(a) $|km|_m = 0$

(b) $[\frac{kX}{km}] = [\frac{X}{m}]$

(c) $kX \bmod km = k \times (X \bmod m)$

(d) $[\frac{a}{m}] = 0$ if and only if $0 \le a < m$

(e) $a \bmod m = a$ if and only if $0 \le a < m$

(f) $\bmod(X \bmod m)\ m = X \bmod m$

II. Addition of Multiples of m:

(a) $|X \pm km|_m = |X|_m$

(b) $[\frac{X \pm km}{m}] = [\frac{X}{m}] \pm k$

III. Additive inverse of X modulo m:

$$|-X|_m = |(m-1)X|_m = |m-X|_m$$

Where $|m-X|_m$ is called the additive inverse of X mod m. Every number has a unique additive inverse modulo m.

IV. Multiplicative inverse: If $|ab|_m = 1$, then a is the multiplicative inverse of b mod m, if and only if $0 \leq a < m$, and the G.C.D. of b and m is equal to 1, hence b mod $m \neq 0$. We denote the multiplicative inverse of b as

$$a = \left|\frac{1}{b}\right|_m \text{ or } \frac{1}{b} \text{ (when m is implicitly assumed)}$$

Also note that:

$$\left|\frac{1}{a}\,\frac{1}{b}\right|_m = \left|\frac{1}{ab}\right|_m = \left|\left|\frac{1}{ab}\right|_m\right|_m$$

V. Solving linear equations: An equation of the form $|X+a|_m = |b|_m$ has the solution $|X|_m = |b-a|_m$.

An equation of the form $|aX|_m = |b|_m$ has a unique solution for $|X|_m$ if and only if the G.C.D. of a and m is equal to 1.

VI. Law of cancellation: Let $|ka|_m = |kb|_m$. Then $|a|_m = |b|_m$ if and only if the G.C.D. of k and m is equal to 1.

VII. Fermat's theorem: $|a^p|_p = |a|_p$ for all integers a if and only if p is prime.

Fermat's theorem is useful for finding multiplicative inverses (if the modulus is prime). The multiplicative inverse of $|a|_p$ is $|a^{p-2}|_p$, because by Fermat's theorem $|a^{p-2}\ a|_p = 1$

VIII. Addition: The ith digit z_i, $1 \leq i \leq n$, of the residue representation of the sum X+Y is given by

$$z_i = |x_i + y_i|_{m_i}.$$

The fact that z_i depends on x_i and y_i only, implies the absence

of any carry from one residue digit position to another. This also means that addition with respect to different moduli can be performed simultaneously. A block diagram of the addition mechanism is shown below in Fig. 1.3.1.

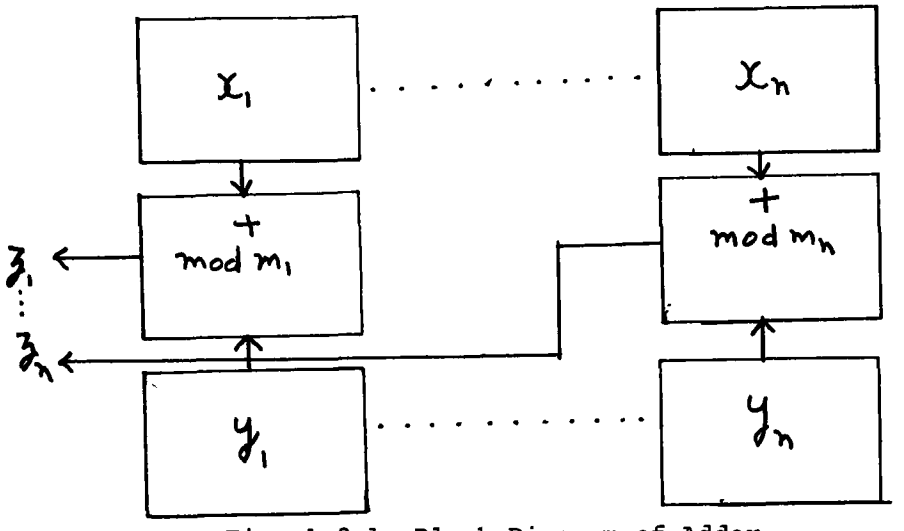

Fig. 1.3.1 Block Diagram of Adder

IX. Multiplication: The i^{th} digit of the residue representation of the product is given by

$$z_i = |x_i \cdot y_i|_{m_i}, \quad 1 \le i \le n.$$

We note that there is no carry from one digit position to another and that multiplication with respect to different moduli can be performed simultaneously. Furthermore, no partial products need be generated. A block diagram of the multiplication mechanism is shown in Fig. 1.3.2.

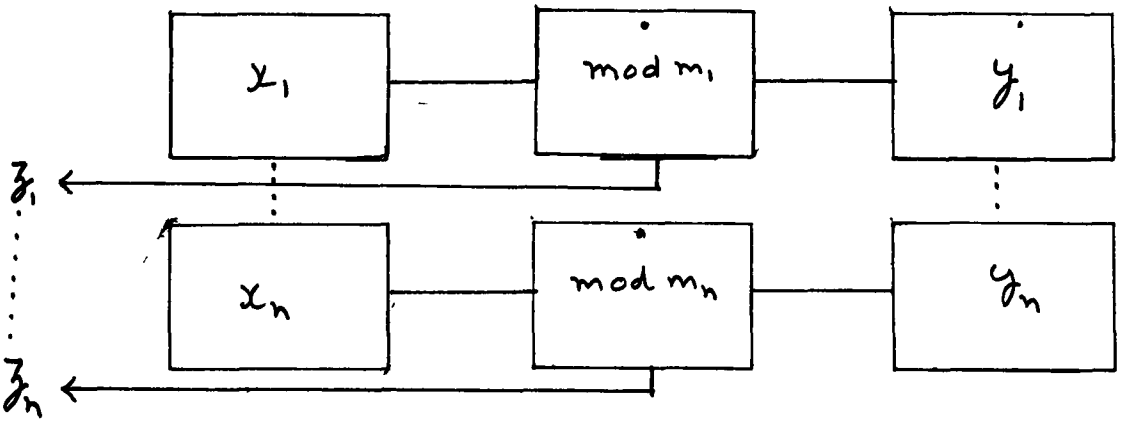

Fig. 1.3.2 Block Diagram of Multiplier

X. Subtraction: Before we discuss the operation of subtraction, we briefly discuss the representation of negative numbers.

One way of representing negative numbers is to divide the range of residue numbers into two parts. One part is assigned to positive numbers and the other to negative numbers. The negative numbers are then represented in terms of their additive inverses.

The i^{th} digit of the residue representation of the difference X-Y is given by

$$z_i = |x_i - y_i|_{m_i} = |x_i + y_i'|_{m_i} \quad \text{(where } y_i' \text{ is the additive inverse of } y_i\text{)}$$

We shall consider examples illustrating the above three arithmetic operations.

Example 1.3.1: Let $M = \{2,3,5\}$. Let X=7, Y=5

$$\begin{array}{rcl} 7 & : & (1,1,2) \\ + & = & + \\ \underline{5} & : & \underline{(1,2,0)} \\ 12 & & (0,0,2) \end{array}$$

Example 1.3.2: Let $M = \{2,3,5\}$. Let X=7, and Y=2

$$\begin{array}{rcl} 7 & : & (1,1,2) \\ \cdot & = & \cdot \\ \underline{2} & : & \underline{(0,2,2)} \\ 14 & & (0,2,4) \end{array}$$

Example 1.3.3: Let $M = \{2,3,5\}$. Let X=7, and Y=2

$$\begin{array}{rcccl} 7 & : & (1,1,2) & & (1,1,2) \\ - & = & - & = & + \\ \underline{2} & : & \underline{(0,2,2)} & & \underline{(0,1,3)} \\ 5 & & & & (1,2,0) \end{array}$$

XI. Division: This is one of the most difficult operations in residue arithmetic and forms the subject of this thesis. We will discuss this operation in Chapters 2 and 3.

1.4 TWO WELL-KNOWN RESULTS FOR NUMBER REPRESENTATION CONVERSION

The following two results provide a mathematical method for converting a number from its residue representation into its weighted representation.

I. Chinese Remainder Theorem: "A system of congruences $X \equiv x_i \bmod m_i$, $1 \le i \le n$, has a unique solution for X mod M if and only if m_i are pair-wise relatively prime". The following describes the process of getting the unique solution [1,2,3,4,7]:

$$x_1 A_1 \frac{M}{m_1} + \ldots + x_n A_n \frac{M}{m_n} \equiv X \bmod M$$

where $A_i \cdot \frac{M}{m_i} = 1 \bmod m_i$.

Example 1.4.1: Let $M = \{2,3,5\}$. Then M=30. We have

$$A_1 \cdot \frac{30}{2} = 1 \bmod 2. \quad \text{Hence } A_1 = 1$$

$$A_2 \cdot \frac{30}{3} = 1 \bmod 3. \quad \text{Hence } A_2 = 1$$

$$A_3 \cdot \frac{30}{5} = 1 \bmod 5. \quad \text{Hence } A_3 = 1$$

Therefore for a residue number (0,2,4), we get for example:

$$0 \times 15 + 2 \times 10 + 4 \times 6 = X \bmod 30$$

Therefore, X=14.

There exists a corollary of the Chinese Remainder Theorem which allows us to obtain the weighted representation from the residue representation when the moduli are not relatively prime. In this case, the set of congruences $X \equiv x_i \bmod m_i$ has a unique solution mod LCM of the m_i's [7].

The Chinese Remainder Theorem is an impractical method for residue number conversion since residue computer would be equipped to perform arithmetic operations modulo m_i, not mod M.

II. Mixed Radix Conversion: An alternative method requiring only mod m_i operations is the mixed radix conversion process [3,7]. This process converts a residue number into its representation in a mixed radix system where the digit weights are not powers of the same bases. Instead, the weights are products of moduli.

A number X can be represented in its mixed radix form as:

$$X = r_n m_1 m_2 \ldots, m_{n-1} + \ldots + r_2 m_1 + r_1 \qquad (1.1)$$

where r_i are called the mixed radix digits, and $0 \leq r_i < m_i$. 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.

$$r_2 = \left| \frac{X-r_1}{m_1} \right|_{m_2} \qquad \text{(from eq. (1.1))}$$

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 1.4.2: Let $M = \{2,3,5\}$ and X=7.

The corresponding mixed-radix expression is:

$$X = r_3(2\times3) + r_2(2) + r_1.$$

We now find the mixed-radix digits as follows:

Moduli:	2	3	5
Residue Representation of X	$\overline{1=r_1}$	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
$\frac{X-r_1}{2}$		$\overline{0=r_2}$	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
$\frac{\frac{X-r_1}{2} - r_2}{3}$			$\overline{1=r_3}$

Hence the mixed-radix representation of X is <1,0,1> and one obtains

$$X = 1(2\times3) + 0(2) + 1 = 7$$

1.5 EXTENSION OF BASE

It is frequently necessary to find the residue representation of a number on one base when this number is represented in another base. The problem is 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, as explained below.

Let M be $\{m_1,m_2,\ldots,m_n\}$. The range of definition is $[0, \prod_{i=1}^{n} m_i-1]$. If another modulus m_{n+1} is included, the range of definition becomes $[0, \prod_{i=1}^{n+1} m_i-1]$, 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.

Example 1.5.1: $M = \{2,3,5\}$. Let us extend the base to $m_4=7$ for the residue representation of $X = (1,1,2)$.

The residue representation of X for the new moduli system will be $(1,1,2,|X|_7)$. The process is started by performing the mixed-radix conversion as described before, and also including $|X|_7$ value in the operations.

Moduli	2	3	5	7
Residue representation of X	$\overline{1=r_1}$	1	2	$\|X\|_7$
Subtract $r_1=1$	1	1	1	1
	0	0	1	$\|X\|_7+6$
multiply by $\|\frac{1}{2}\|_{m_i}$		2	3	4
		$\overline{0=r_2}$	3	$4\|X\|_7+3$
Subtract $r_2=0$		0	0	0
			3	$4\|X\|_7+3$
multiply by $\|\frac{1}{3}\|_{m_i}$			2	5
			$\overline{1=r_3}$	$6\|X\|_7+1$
Subtract $r_3=1$			1	1
			0	$6\|X\|_7+0$
multiply by $\|\frac{1}{5}\|_{m_i}$				3
				$4\|X\|_7$

However, since X must fall into the interval $[0, \prod_{i=1}^{n} m_i - 1]$;

$$r_4 = 4|X|_7 = 0.$$

Multiplying by $|\frac{1}{4}|_7$; i.e. 2, yields

$$|X|_7 = |0 \times 2|_7 = 0.$$

CHAPTER 2

DIVISION OPERATION

In this chapter, we consider the problem of division in RNS. A residue number system is not a weighted number system, so the operation of division which involves magnitude comparison of two operands is not straightforward. We discuss the existing methods for the operation of division and their limitations in this chapter.

2.1 CATEGORIZATION OF DIVISION

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

2.1.1 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 [7]:

Theorem

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

$$\left|\frac{x}{y}\right|_{m_i} = \left|\frac{1}{y}x\right|_{m_i} \tag{2.1}$$

for all m_i

If y does not divide x, the residue of the quantity $\frac{x}{y}$ is not an integer, and $|\frac{x}{y}|_{m_i}$ is not defined. Consequently, (2.1) has no meaning.

Example 2.1:

For $M = \{2,3,5\}$ we divide 14 by 7. The residue representation of 14 is (0,2,4) and that of 7 is (1,1,2).

$$|\tfrac{1}{7}|_2 = 1, \qquad |\tfrac{1}{7}|_3 = 1, \qquad |\tfrac{1}{7}|_5 = 3$$

Therefore, for $\frac{14}{7} = 2$, the residue code is

$$(|0\times1|_2,\ |2\times1|_3,\ |4\times3|_5) \text{ or } (0,2,2) \text{ i.e. } 2.$$

On the other hand, if we divide 17 by 7 (where 17 is not divisible by 7 without remainder), we have: the residue code for 17: (1,2,2). Then for $\frac{17}{7}$ we get:

$$(|1\times1|_2,\ |2\times1|_3,\ |2\times3|_5) \text{ or } (1,2,1) \text{ i.e. } 11.$$

which is obviously wrong.

2.1.2 Category 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 computers, in the sense that division by the defined restricted set of numbers is faster than by an arbitrary divisor.

Scaling Positive Integers

Division in any integer number system is defined by the relation:

$$x = [\frac{x}{y}]y + |x|_y$$

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 Cateogry, for all i for which the G.C.D. of m_i and y is equal to 1, one obtains:

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

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 equal to 1. The remaining digits may be found by base extension method. Thus the scaling algorithm consists of two steps:

1) Division Remainder Zero, and

2) Base extension

Example 2.2

For $M = \{2,3,5\}$, we want to determine the residue representation of the integer value $[\frac{26}{5}]$. Note that the divisor in this case is simply m_3.

We first have to determine the residue representation of the number which is divisible by 5 and is closest to x without exceeding x, that is, $x-|x|_5$. This can be found by subtracting the residue representation of $|x|_5=1$ from x. In the residue code we have:

moduli	2	3	5
Residue representation of 26	(0	2	1)
Subtract $\|x\|_5=1$	(1	1	1)
$x-\|x\|_5$	(1	1	0)

The result is divisible by 5. Except for modulus m_3, which is itself the divisor, all moduli are relatively prime to the divisor. Hence division by division-remainder-zero method may be employed if the residue digit in the m_3 position is temporarily disregarded.

Multiplying by $|\frac{1}{5}|_{m_i}$ one obtains:

$x-|x|_5$ i.e. (1 1 -)

multiplying by $|\frac{1}{5}|_{m_i}$ i.e. (1 2 -)

$\frac{x-|x|_5}{5}$ i.e. (1 2 -)

The original interval of definition for the full set of moduli was [0,29]. Since x must be in this interval of definition, it follows that $[\frac{x}{5}]$ must therefore be in the interval [0,5]. Since the moduli 2, and 3 have an interval of definition [0,5], the residue representation (1 2 -) is unambiguous. The m_3 digit may then be found by base-extension. Denoting $[\frac{x}{5}]$ by z, one obtains:

moduli	2	3	5
Subtract $r_1=1$	$1=r_1$	2	$\|z\|_5$
	1	1	1
Multiply $\|\frac{1}{2}\|_{m_i}$	0	1	$\|z\|_5+4$
		2	3
Subtract $r_2=2$		$2=r_2$	$3\|z\|_5+2$
		2	2
Multiply $\|\frac{1}{3}\|_{m_i}$		0	$6\|z\|_6$
			2
			$2\|z\|_5$

However, r_3 corresponds to r_{N+1}; therefore, $r_3=0$. Hence

$$r_3=2|z|_5=0$$

therefore, $|z|_5=0$.

Hence the residue representation of $[\frac{x}{5}]$ is (1 2 0) i.e. 5.

Scaling a Positive Number by Several Moduli

In the preceding example, the scaling factor was only one modulus, but if the scale factor is to be the product of first powers of some of the moduli, for example, say $m_2 \times m_3$, then first the closest multiple of m_2 not exceeding the dividend is located and this number is divided by m_2. The quotient then becomes the new dividend, and its closest multiple of m_3 (not exceeding the new dividend) is determined. Division by m_3 yields the integer value of the quotient. Base extension completes the operation, yielding the rounded down, scaled value. Changing the sequence, first dividing by m_3 and then by m_2, does not affect the result.

Scaling Numbers of Either Sign

If it is known a priori that x is negative, one can determine

x from M-x, scale by y to obtain $[\frac{x}{y}]$, and then represent the result by $M+[\frac{x}{y}]$. If we do not know a priori that x is negative, and scale it as though it were positive, the result is $\frac{M}{y}+[\frac{x}{y}]$ instead of the desired $M+[\frac{x}{y}]$. One solution to the problem is to determine the sign of x before scaling.

This extra sign detection process can be avoided, if we note that division by y maps all numbers in the interval [0,M/2-1] into $[0,\frac{M/2-1}{y}]$, and all numbers in the interval [M/2, M -1] into $[\frac{M/2}{y}, \frac{M-1}{y}]$. Hence, it is possible to divide the number by y first. Then by noting the interval in which $|[\frac{x}{y}]|_M$ lies, the sign of x can be determined. If x is negative, $|-\frac{M}{y}|_M$ is added, modulo M, to $\frac{M}{y}+[\frac{x}{y}]$ to obtain the desired $M+[\frac{x}{y}]$.

Ordinarily, determining the interval in which $[\frac{x}{y}]$ lies requires the same amount of time as sign detection. However, as previously mentioned, the scaling process requires a base-extension operation, and, as a result, the mixed-radix digits are available. Therefore, one can determine the location of $[\frac{x}{y}]$ by examining the mixed-radix digits.

Example 2.3

For $\bar{M}$ = {13,9,11,7,2} scale -979 by 7×11, rounding down to nearest integer (e.g., -17.1 becomes -18).

Moduli:	13	9	11	7	2	
Residue Representation of x	9	2	0	1	1	
Subtract $\|x\|_7$=1	1	1	1	1	1	
	8	1	10	0	0	$x-\|x\|_7$
Multiply by $\|\frac{1}{7}\|_{m_i}$	2	4	8	-	1	

Moduli:	13	9	11	7	2	
	3	4	3	-	0	$\frac{x-\lvert x\rvert_7}{7}$
Subtract $\lvert x\rvert_{11}=3$	3	3	3	-	1	
	0	1	0	-	1	$\frac{x-\lvert x\rvert_7}{7}-\lvert x\rvert_{11}$
Multiply by $\lvert\frac{1}{11}\rvert_{m_i}$	6	5	-	-	1	
	0	5	-	-	1	$\frac{\frac{x-\lvert x\rvert_7}{7}-\lvert x\rvert_{11}}{11}$

Let $[\frac{x}{y}]=z$

We now base extend:

	$0=r_1$	5	$\lvert z\rvert_{11}$	$\lvert z\rvert_7$	1
Subtract $r_1=0$	0	0	0	0	0
	0	5	$\lvert z\rvert_{11}$	$\lvert z\rvert_7$	1
Multiply by $\lvert\frac{1}{13}\rvert_{m_i}$		7	6	6	1
		$8=r_2$	$6\lvert z\rvert_{11}$	$6\lvert z\rvert_7$	1
Subtract $r_2=8$		8	8	1	0
			$6\lvert z\rvert_{11}+3$	$6\lvert z\rvert_7+6$	1
Multiply by $\lvert\frac{1}{9}\rvert_{m_i}$			5	4	1
			$8\lvert z\rvert_{11}+4$	$3\lvert z\rvert_7+3$	$1=r_3$
Subtract $r_3=1$			1	1	1
			$8\lvert z\rvert_{11}+3$	$3\lvert z\rvert_7+2$	

Since $\lvert 8\lvert z\rvert_{11}+3\rvert_{11}=0$

$8\lvert z\rvert_{11}=8$

$\lvert z\rvert_{11}=\lvert 8\times 7\rvert_{11}=0$

and $\lvert 3\lvert z\rvert_7+2\rvert_7=0$

$3\lvert z\rvert_7=5$

$\lvert z\rvert_7=\lvert 5\times 5\rvert_7=4$

Hence, the residue representation of z is (0,5,1,4,1). Depending

on the sign of x, the residue representation of z will be either $[\frac{x}{y}]$ or $\frac{M}{y}+[\frac{x}{y}]$. The mixed-radix digits of z for the moduli 13, 9 and 2 were generated during the base extension process. z can be represented for the moduli 13, 9 and 2 as:

$$z = 1(13\times9)+8(13)+0$$

If x is positive, $|x|_M$ must lie in the interval [0, M/2-1]. Therefore $|z|_M$ should lie in the interval $[0, \frac{M/2-1}{77}]$. Since $|z|_M$ does not lie in this interval, x must be negative. Therefore, $|-\frac{M}{y}|_M$ i.e. (0,0,8,4,0) is added to z i.e. (0,5,1,4,1). The result is (0,5,9,1,1) which corresponds to

$$[\frac{-979}{77}] = -13.$$

Rounding to the Closest Integer

The scaling algorithm can be modified so that the quotient is rounded to the closest integer value, rather than to the integer value which is closest but less than the complete answer. The modifications require that M should be even, and y the scale factor be odd [7,11].

The main difference between this method and the scaling method is that the first step in the scaling method was to subtract $|x|_{m_i}$ from $|x|_M$. In the modification, $|x|_{m_i}$ is subtracted from $|x|_M$ only if $0\leq|x|_{m_i} < \frac{m_i}{2}$, and $|m_i-x|_{m_i}$ is added if $m_i/2\leq|x|_{m_i}<m_i$. This procedure is repeated for each modulus which is a factor of the divisor. This method produces an integer closest to the quotient provided no overflow occurs.

Example 2.4

For $M = \{2,9,13,11,7\}$, scale and round to the nearest integer

x=5517 divided by y=11*7.

Moduli	2	9	13	11	7
Residue representation of x:	1	0	5	6	1
Subtract $\|x\|_7=1$	1	1	1	1	1
	0	8	4	5	0
Multiply by $\|\frac{1}{7}\|_{m_i}$	1	4	2	8	-
	0	5	8	7	-
Since $\|x\|_{11}$ is $> \frac{11}{2}$ we add $\|m_i-7\|_{m_i}=4$	0	4	4	4	-
	0	0	12	0	-
Multiply by $\|\frac{1}{11}\|_{m_i}$	1	5	6	-	-
	0	0	7	-	-
Let $[\frac{x}{y}]=z$					
we now base extend	$0=r_1$	0	7	$\|z\|_{11}$	$\|z\|_7$
Subtract 7	1	7	7	7	0
	1	2	0	$\|z\|_{11}+4$	$\|z\|_7$
Multiply by $\|\frac{1}{13}\|_{m_i}$	1	7	-	6	6
	1	5	-	$6\|z\|_{11}+2$	$6\|z\|_7$
Subtract 5	1	5	-	5	5
	0	0	-	$6\|z\|_{11}+8$	$6\|z\|_7+2$

Since $|6|z|_{11}+8|_{11}=0$

$6|z|_{11}=3$

$|z|_{11}=|3*2|_{11}=6$

and $|6|z|_7+2|_7=0$

$6|z|_7=5$

$|z|_7=|5*6|_7=2$

Therefore z i.e. (0,0,7,6,2)

2.1.3 Category 3: GENERAL DIVISION

In this cateogry, both the dividend and the divisor are arbitrary integers.

Method 1: DIVISION USING POWERS OF TWO

A division procedure which uses a stored table of powers of 2 has been described in [4,7]. This method, which is essentially a variation of the usual method of division by repeated subtraction of the divisor from the dividend, generates exactly one binary bit of the quotient per iteration. It is not subject to overflow error and seems to require little extra hardware, but is not as fast as might be desired.

The method is based on determining the residues of the binary equivalent of z. The general approach used to accomplish this is to approximate T(z) (binary equivalent of z) by T'(z), form T'(z)y, and compare the latter with x. Depending on whether $x \geq T'(z)y$ or $x < T'(z)y$, a new estimate of T(z) is made. After T(z) has been determined, the process is continued by determining T(z-T(z)), T(z-T(z-T(z))), etc. in a similar manner.

The conventional binary representation of z,

$$z=a_N 2^N+a_{N-1}2^{N-1}+ \ldots +a_0 \quad \text{where } a_i=1 \text{ or } 0, \text{ and N is an integer} \qquad (2.2)$$

In terms of T(z), z can be written as

$$z=T(z)+T(z-T(z))+T(z-T(z)-T(z-T(z)))+ \ldots \qquad (2.3)$$

The first non-zero term of Eq. (2.2) is equal to the first term of Eq. (2.3), the second non-zero term of Eq. (2.2) is equal to second term of Eq. (2.3), etc. The division algorithm consists of finding the terms of Eq. (2.3) where now z is the integer value of the quotient, i.e. $z=[\frac{x}{y}]$. The first step in the division algorithm is to find a least upper bound for T(z).

Determination of the Least Upper Bound for T(z)

Since $z=[\frac{x}{y}]$, $z\leq\frac{x}{y}$, it follows that $x\geq zy$. Since $T(z)\leq z$, we have $x\geq T(z)y$. Furthermore, since x is positive, $x<M/2$ and it is obvious that $T(y)\leq y$. Hence we can write $\frac{M/2}{T(y)}>T(z)$. Since T(z) is an integer power of 2, it follows that

$$\frac{T(M/2)}{T(y)} \geq T(z).$$

Therefore, $\frac{T(M/2)}{T(y)}$ is an upper bound for T(z). The quantity T(y) can be found from the stored table by using repeated sign detections to determine the location of y in the table. The ratio $\frac{T(M/2)}{T(y)}$ can then be found if T(M/2) is available as a permanently stored value

While $\frac{T(M/2)}{T(y)}$ is an upper bound for T(z), it is <u>possible</u> that a lesser upper bound may exist. To check for the existence of a lesser upper bound, we note that $M/2>x$, and, hence, $M/2>T(z)y$. $M/2\leq\frac{T(M/2)}{T(y)}y$ implies that the upper bound $T(M/2)/T(y)$ is too high and, consequently, a lesser upper bound exists for T(z), namely, $T(z)\leq 1/2\frac{T(M/2)}{T(y)}$. Conversely, $\frac{T(M/2)}{T(y)}y<M/2$, implies that the original bound $\frac{T(M/2)}{T(y)}$ is a least upper bound for T(z). Hence, by performing a mixed-radix conversion on the quantity, $\frac{T(M/2)}{T(y)}y$, a least upper bound for T(z) may be found. This least upper bound is designated by $\overline{T(z)}$.

Determination of the Terms of Eq. 2.3

After $\overline{T(z)}$ is found, this bound is used as T'(z), the <u>first</u> estimate for the quantity T(z). If $(x-T'(z)y)\geq 0$, this implies that $T(z)\geq T'(z)$. Conversely, if $(x-T'(z)y)<0$, this implies that $T'(z)>T(z)$. Hence by determining the sign of quantities of the

form $(x-T^i(z)y)$ where $T^i(z)$ are different estimates, it is possible to find T(z).

Having determined the first term of Eq. 2.3, we can proceed to determine the second term in a similar manner. An estimate $T'(z-T(z))=1/2T(z)$, is made of $T(z-T(z))$. Then

$$(x-(T(z)+T'(z-T(z)))y)\geq 0$$

implies that $T'(z-T(z))\leq T(z-T(z))$. Conversely

$$(x-(T(z)+T'(z-T(z)))y)<0$$

implies that $T'(z-T(z))>T(z-T(z))$. The remaining terms of Eq. 2.3 may be determined in a similar manner.

Execution Time

Most of the time required for this division algorithm is spent on determining the highest power of 2 contained in some number. On the average, $1+\log_2\sqrt{c}+c/2$, mixed radix conversions and $3+c+\log_2\sqrt{c}$ additions or multiplications are required for division using this method [4,7], where $c=[\log_2(\frac{M}{2}-1)]$. A considerable improvement in speed is possible if the computer is equipped with a table, which lists the highest power of 2 for all combinations of the most significant mixed-radix digit. The highest power of 2 contained in any number x may then be found by performing a mixed-radix conversion on x and finding the table entry for the most significant mixed-radix digit.

Example 2.5

For the moduli $M = \{2,5,7,9,11,13\}$ divide 41235 by 781.

Step 1 Determine T(y). We find that the largest $k>0$ for which $2^k\leq 781$ is k=9. Therefore, we choose

$T(y)=512 \doteq (0,2,1,8,6,5)$.

Step 2 Form $\frac{T(M/2)}{T(y)}$. Since T(M/2) is a multiple of T(y), the division-remainder zero method may be used. We obtain $\frac{T(M/2)}{T(y)} = 64$.

Step 3 Find $\overline{T(z)}$. Determine the sign of $\frac{T(M/2)}{T(y)}y = 64*781 = 49984$. Since M/2=45045, 49984 corresponds to a negative representation. This implies that $\overline{T(z)} = \frac{1}{2}\frac{T(M/2)}{T(y)} = 32$.

Step 4 Find T(z). Estimate T(z) as T'(z)=32. Then test the sign of the quantity:

$$(x-T'(z)y) = (41235-32*781) = 16243$$

This is positive representation. Hence T(z)=32.

Step 5 Determine T(z-T(z)). Estimate T(z-T(z)) as T'(z-T(z))=16, then:

$$x-(T(z)+T'(z-T(z)))y = 3747$$

Since this is positive representation, T(z-T(z))=16.

Step 6 Determine T(z-T(z)-T(z-T(z))). Estimate T(z-T(z)-T(z-T(z))) as T'(z-T(z)-T(z-T(z)))=8. Then $x-((T(z)+T(z-T(z)))+T'(z-T(z)-T(z-T(z)))y = -2501$. This is a negative number, hence the estimate was too high. Make a new estimate: T"(z-T(z)-T(z-T(z)))=4. $x-(T(z)+T(z-T(z))+T"(z-T(z-T(z))))y = 623$. This is a positive number, hence T(z-T(z)-T(z-T(z)))=4.

By repeating the previous procedure it can be seen that Eq. 2.3 does not have any more terms. Therefore, z=32+16+4=52.

Method 2: DIVISION USING AN APPROXIMATE DIVISOR

A method for approximating the quotient by another integer has been described in [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 $y \leq \bar{y} < 2y$, 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:

$$x_i = x_{i-1} - yz_i, \quad x_0 = x$$

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 r^{th} iteration, then

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

where

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

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.

Example 2.6

For the moduli $M = \{23, 19, 17, 13, 11, 7, 5, 3, 2\}$, divide $x = 10304312$ by $y=1401$.

Step 1 Form the table of approximate divisors.

Table 1 is used when all but one mixed-radix digits are equal to zero. Table 2 is used when more than one mixed-radix digit is non-zero. a_p is the most significant non-zero mixed radix digit.

Step 2 Obtain the mixed-radix digits of y (listed in the order of decreasing significance).

$y = <0,0,0,0,0,0,3,3,21>$

Using the Table 2, with $a_p=a_3$ we obtain $\bar{y}=Q\sum_{i=1}^{p-1} m_i$, where Q is given in the table, i.e. $\bar{y}=5(23\times19)=2185$. We can then find $z_1=[\frac{x}{\bar{y}}]$ by the scaling algorithm to be 4715.

Step 3 Find $x_1=x_0-yz_1=3698597$.

Step 4 Find $z_2=[\frac{x_1}{\bar{y}}]=[\frac{3698597}{2185}]=1692$.

Step 5 Find $x_2=x_1-yz_2=1328105$.

Step 6 Find $z_3=[\frac{x_2}{\bar{y}}]=607$.

Step 7 Find $x_3=x_2-yz_3=477698$.

Step 8 Find $z_4=[\frac{x_3}{\bar{y}}]=218$.

Step 9 Find $x_4=x_3-yz_4=172280$.

TABLE 1		TABLE 2	
If $a_i=0$ for $i\neq p$		If $a_i\neq 0$ for some $i\neq p$	
a_p	Q	a_p	Q
1	1	1	2
2	2	2	3
3	3	3	5
4	5	4	5
5	5	5	3*2
6	3*2	6	7
7	5*2	7	5*2
8	5*2	8	5*2
9	5*2	9	5*2
10	5*2	10	11
11	11	11	13
12	13	12	13
13	13	13	7*2
14	7*2	14	5*3
15	5*3	15	17
16	17	16	17
17	17	17	19
18	19	18	19
19	19	19	7*3
20	7*3	20	7*3
21	7*3	21	23
22	11*2	22	23

Step 10 Find $z_5=[\frac{x_4}{\bar{y}}]=78$.

Step 11 Find $x_5=x_4-yz_5=63002$.

Step 12 Find $z_6=[\frac{x_5}{\bar{y}}]=28$.

Step 13 Find $x_6=x_5-yz_6=23774$.

Step 14 Find $z_7=[\frac{x_6}{y}]=10$.

Step 15 Find $x_7=x_6-yz_7=9764$.

Step 16 Find $z_8=[\frac{x_7}{y}]=4$.

Step 17 Find $x_8=x_7-yz_8=4160$.

Step 18 Find $z_9=[\frac{x_8}{y}]=1$.

Step 19 Find $x_9=x_8-yz_9=2759$.

Step 20 Find $z_{10}=[\frac{x_9}{y}]=1$.

Step 21 Find $x_{10}=x_9-yz_{10}=1358$.

Step 22 Find $z_{11}=[\frac{x_{10}}{y}]=0$.

Since $z_r=0$ (i.e., $z_{11}=0$), but $x_{r-1}\neq y$, then $z_r'=0$.

Hence

$$z=\sum_{i=1}^{10} z_i=4715+1692+607+218+78+28+10+4+1+1=7354.$$

2.2 LIMITATIONS OF EXISTING METHODS FOR DIVISION

2.2.1 Category 1: Division Remainder Zero Method is of very restricted use since it must be known a priori that its prerequisites are satisfied in order to perform the operation.

2.2.2 Cateogry 2: Scaling Method is analogous to division by a power of 2 in binary computers in the sense that division by the defined restricted set of numbers is faster than by an

arbitrary divisor. In the binary system, of course, division by a power of 2 is simply a shifting operation. In the residue system, division by a factor of M is not so simple but is nonetheless much faster than division by an arbitrary number.

2.2.3 Cateogry 3:

A. Division using the method powers-of-two requires the availability of a table of residue representations of the integer powers of 2 (up to M/2-1), along with the exponent of each entry. This method also requires sign detection to find the highest power of 2 contained in any number.

B. Division using an approximate divisor requires a table-look-up. The table would necessarily consist of all products of the first n (number of moduli) primes taken 1,2,...,n-1 at a time. Such a table would then contain (2^n-2) words, a rather formidable number for all but small n. Also $\bar{y}$ should be found, which is a product of the first powers of some of the moduli and which lies in the interval $y \leq \bar{y} < 2y$. It is easily seen that this requirement cannot be satisfied for every set of moduli; for example, for the moduli system $m_1=9$, $m_2=11$, the condition $y \leq \bar{y} < 2y$ is not satisfied for y=4.

CHAPTER 3

TWO MODIFIED METHODS FOR DIVISION

The problem of division in residue arithmetic is a major problem, as we saw in Chapter 2. We shall now propose two modified methods which overcome most of the restrictions encountered in the existing methods. These methods are based on the approximate divisor method mentioned in Chapter 2. A disadvantage of the modified methods is that they also require a table-look-up. The major advantages are, 1) they are more general, that is, the approximate divisor need not be a product of the first powers of some of the moduli and the relation $y \leq \bar{y} \leq 2y$ is automatically satisfied, 2) they are much faster than the existing methods. In addition one of the two proposed methods produces zero error for the derived quotient. The complete mathematical proof for method 1 is given in the appendix. For method 2, only the first two termination conditions are proved (the third termination condition has not been found).

3.1 THE PROPOSED METHOD (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 i^{th}

iteration ; i=1,2,...

Approximate Dividend: Let x_{i-1} be denoted in its mixed radix representation as: $x_{i-1} \leftrightarrow <0,0,\ldots,\alpha_k,\alpha_{k-1},\ldots\alpha_1>$ 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 choosen to be:

$\bar{x}_{i-1}=\alpha_k m_1 m_2,\ldots m_{k-1}$

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 choosen to be

$$\bar{y}=(\beta_\ell+1)m_1 m_2\ldots m_{\ell-1}$$

Approximate Quotient: The approximate quotient z_i in the i^{th} iteration is determined by the inter-relationship between k and ℓ.

Case 1: k=ℓ

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

Case 2: k=ℓ+1

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

Case 3A: k=ℓ+2

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

Case 3B: k=ℓ+3

$$z_i = \frac{\bar{x}_{i-1}}{\bar{y}} = \frac{\alpha_k m_1 m_2\ldots m_{\ell-1} m_\ell m_{\ell+1} m_{\ell+2}}{(\beta_\ell+1)m_1 m_2\ldots m_{\ell-1}} = \alpha_k \frac{m_\ell m_{\ell+1} m_{\ell+2}}{(\beta_\ell+1)}$$

and so on. In general, we would get:

Case 3: $k>\ell+1$

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

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

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

and,

$$z_{i+1} = [\frac{\bar{x}_i}{\bar{y}}] \tag{3.2}$$

We first find z_1, and use relations (3.1) and (3.2) repeatedly to obtain z_{i+1}, z_{i+2}, ..., etc. This iterative procedure is continued until either $z_i=0$, or $x_i=0$. If this occurs in the r^{th} 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 $\frac{m_\ell}{(\beta_\ell+1)}$, $\frac{\alpha_k}{(\beta_\ell+1)}$, $\frac{m_\ell}{\beta_\ell}$, $\frac{\alpha_k}{\beta_\ell}$.

3.2.1 TABLE-LOOK-UPS OF THE FORM $m_{\ell+1} m_{\ell+2} \cdots m_{k-1}$

For $k>\ell+1$, ($\ell=1,2,\ldots,n-2$) and ($k=\ell+2,\ell+3,\ldots,n$), 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-UP OF THE FORMS $\frac{m_\ell}{(\beta_\ell+1)}$, $\frac{\alpha_k}{(\beta_\ell+1)}$, $\frac{m_\ell}{\beta_\ell}$ 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 $m_\ell < \beta_{\ell+1}$, $\alpha_k < \beta_{\ell+1}$, $m_\ell < \beta_\ell$, 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 approximate divisor algorithm outlined in Chapter 2 (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 choosen and it was found (when averaged over these 50 pairs) 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 $y \leq \bar{y} < 2y$. 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. Table 3.1 summarizes the comparison, between the Tanaka method and our method for a typical moduli system, for cases that are known to be slow via our method and fast via the Tanaka method.

TABLE 3.1

$M=\{29,23,5,3,2\}$

Deviation = True Quotient - Computed Quotient
Basic Operation Saved = Basic Operation (Tanaka Method) - Basic Operation (Method One)

Comparison of Method Tanaka with Method One

		Method Tanaka			Method One		
Numerator	Denominator	Deviation	Percentage Error	No. of Basic Operations	Deviation	Percentage Error	No. of Basic Operations
8809	1255	0	0.0	32	0	0.0	44
12883	3582	0	0.0	45	0	0.0	43
6505	1921	0	0.0	31	0	0.0	54
1867	583	0	0.0	44	0	0.0	43
8659	3440	0	0.0	32	0	0.0	31
3395	1423	0	0.0	32	0	0.0	32
609	294	0	0.0	32	0	0.0	31
5225	2875	0	0.0	31	0	0.0	31
4441	2498	0	0.0	31	0	0.0	31
9537	5521	0	0.0	31	0	0.0	30
17467	11013	0	0.0	32	0	0.0	20
4545	2910	0	0.0	31	0	0.0	31
4427	2853	0	0.0	31	0	0.0	31
15809	11169	0	0.0	32	0	0.0	20
5369	3883	0	0.0	32	0	0.0	20
18617	14994	0	0.0	32	0	0.0	20
11291	9435	0	0.0	31	0	0.0	31
11195	10028	0	0.0	32	0	0.0	20
11905	11510	0	0.0	32	0	0.0	20
14863	12443	0	0.0	32	0	0.0	20
14763	11313	0	0.0	32	0	0.0	20
14730	10571	0	0.0	32	0	0.0	20
6236	4385	0	0.0	32	0	0.0	20
16762	11771	0	0.0	32	0	0.0	20
2988	1929	0	0.0	31	0	0.0	30
13981	8307	0	0.0	31	0	0.0	31
3198	1843	0	0.0	31	0	0.0	30

(Continued)

TABLE 3.1

(Continued)

		Method Tanaka			Method One		
Numerator	Denominator	Deviation	Percentage Error	No. of Basic Operations	Deviation	Percentage Error	No. of Basic Operations
16171	9233	0	0.0	31	0	0.0	31
19411	10369	0	0.0	32	0	0.0	20
12826	6785	0	0.0	31	0	0.0	31
12986	6457	0	0.0	32	0	0.0	32
3995	1747	0	0.0	44	0	0.0	42
13756	5705	0	0.0	31	0	0.0	42
12547	5067	0	0.0	44	0	0.0	42
19064	7449	0	0.0	44	0	0.0	43
12426	4761	0	0.0	44	0	0.0	42
17126	6187	0	0.0	31	0	0.0	43
895	307	0	0.0	31	0	0.0	31
18296	6097	0	0.0	32	0	0.0	44
112	35	0	0.0	45	0	0.0	42
8753	2569	0	0.0	44	0	0.0	43
11408	3257	0	0.0	31	0	0.0	31
11529	3097	0	0.0	31	0	0.0	31
10075	2379	0	0.0	32	0	0.0	42
86	19	0	0.0	44	0	0.0	55
19589	4267	0	0.0	57	0	0.0	66
17544	3697	0	0.0	45	0	0.0	56
3552	739	0	0.0	57	0	0.0	53
10391	1977	0	0.0	31	0	0.0	54
2526	459	0	0.0	57	0	0.0	55

Average Deviation for Tanaka Method = 0.0
Average Deviation for Method One = 0.0
Average Percentage Error for Tanaka Method = 0.0
Average Percentage Error for Method One = 0.0
Average No. of Basic Operations for Tanaka Method = 35.5999908447
Average No. of Basic Operations for Method One = 34.8999938965
Average No. of Basic Operations saved in Method One relative to Tanaka Method = 0.6999969482

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 choosen as: $\bar{y}=\beta_\ell m_1 m_2 \ldots m_{\ell-1}$, 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 emperically 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. Table 3.2 summarizes the comparison between the Tanaka method and Method Two for a typical moduli system for cases that are known to be slow via our method and fast via the Tanaka method.

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,

TABLE 3.2

(Continued)

		Method Tanaka			Method Two			Method One And Two			
Numerator	Denominator	Deviation	Percentage Error	No. of Basic Operations	Deviation	Percentage Error	No. of Basic Operations	Method Two Failed	Deviation	Percentage Error	No. of Basic Operations
16171	9233	0	0.0	31	0	0.0	32	Yes	0	0.0	60
19411	10369	0	0.0	32	0	0.0	30	No	0	0.0	30
12826	6785	0	0.0	31	0	0.0	31	No	0	0.0	31
12986	6457	0	0.0	32	0	0.0	21	Yes	0	0.0	51
3995	1747	0	0.0	44	0	0.0	31	No	0	0.0	31
13756	5705	0	0.0	31	0	0.0	21	Yes	0	0.0	61
12547	5067	0	0.0	44	0	0.0	21	Yes	0	0.0	61
19064	7449	0	0.0	44	0	0.0	43	No	0	0.0	43
12426	4761	0	0.0	44	0	0.0	21	Yes	0	0.0	61
17126	6187	0	0.0	31	0	0.0	21	Yes	0	0.0	62
895	307	0	0.0	31	0	0.0	31	No	0	0.0	31
18296	6097	0	0.0	32	0	0.0	31	No	0	0.0	31
112	35	0	0.0	45	0	0.0	30	No	0	0.0	30
8753	2569	0	0.0	44	0	0.0	43	No	0	0.0	43
11408	3257	0	0.0	31	0	0.0	31	No	0	0.0	31
11529	3097	0	0.0	31	0	0.0	31	No	0	0.0	31
10075	2379	0	0.0	32	0	0.0	42	No	0	0.0	42
86	19	0	0.0	44	0	0.0	55	No	0	0.0	55
19589	4267	0	0.0	57	0	0.0	32	Yes	0	0.0	95
17544	3697	0	0.0	45	0	0.0	42	No	0	0.0	42
3552	739	0	0.0	57	0	0.0	21	Yes	0	0.0	72
10391	1977	0	0.0	31	0	0.0	21	Yes	0	0.0	73
2526	459	0	0.0	57	0	0.0	55	No	0	0.0	55

(Continued)

TABLE 3.2

$M=\{29,23,5,3,2\}$

Deviation = True Quotient - Computed Quotient
Basic Operation Saved = Basic Operation (Tanaka Method) - Basic Operation (Method Two)

Comparison of Method Tanaka with Methods One and One and Two

		Method Tanaka			Method Two			Method One And Two			
Numerator	Denominator	Deviation	Percentage Error	No. of Basic Operations	Deviation	Percentage Error	No. of Basic Operations	Method Two Failed	Deviation	Percentage Error	No. of Basic Operations
8809	1255	0	0.0	32	-2	-28.57143	21	Yes	0	0.0	63
12883	3582	0	0.0	45	0	0.0	31	No	0	0.0	31
6505	1921	0	0.0	31	0	0.0	42	No	0	0.0	42
1867	583	0	0.0	44	0	0.0	43	No	0	0.0	43
8659	3440	0	0.0	32	0	0.0	30	No	0	0.0	30
3395	1423	0	0.0	32	0	0.0	31	No	0	0.0	31
609	294	0	0.0	32	0	0.0	30	No	0	0.0	30
5225	2875	0	0.0	31	0	0.0	31	No	0	0.0	31
4441	2498	0	0.0	31	0	0.0	31	No	0	0.0	31
9537	5521	0	0.0	31	0	0.0	20	Yes	0	0.0	48
17467	11013	0	0.0	32	0	0.0	30	No		0.0	30
4545	2910	0	0.0	31	0	0.0	31	No	0	0.0	31
4427	2853	0	0.0	31	0	0.0	31	No	0	0.0	31
15809	11169	0	0.0	32	0	0.0	30	No	0	0.0	30
5369	3883	0	0.0	32	0	0.0	30	No	0	0.0	30
18617	14994	0	0.0	32	0	0.0	30	No	0	0.0	30
11291	9435	0	0.0	31	0	0.0	31	No	0	0.0	31
11195	10028	0	0.0	32	0	0.0	30	No	0	0.0	30
11905	11510	0	0.0	32	0	0.0	30	No	0	0.0	30
14863	12443	0	0.0	32	0	0.0	30	No	0	0.0	30
14763	11313	0	0.0	32	0	0.0	30	No	0	0.0	30
14730	10571	0	0.0	32	0	0.0	30	No	0	0.0	30
6236	4385	0	0.0	32	0	0.0	30	No	0	0.0	30
16762	11771	0	0.0	32	0	0.0	30	No	0	0.0	30
2988	1929	0	0.0	31	0	0.0	20	Yes	0	0.0	48
13981	8307	0	0.0	31	0	0.0	31	No	0	0.0	31
3189	1843	0	0.0	31	0	0.0	20	Yes	0	0.0	48

(Continued)

TABLE 3.2

(Continued)

Average Deviation for Tanaka Method	= 0.0
Average Deviation for Method Two	= 0.0399999991
Average Deviation for Method One And Two	= 0.0
Average Percentage Error for Tanaka Method	= 0.0
Average Percentage Error for Method Two	= 0.5714284778
Average Percentage Error for Method One And Two	= 0.0
Average No. of Basic Operations for Tanaka Method	= 35.5999908447
Average No. of Basic Operations for Method Two	= 30.8199920654
Average No. of Basic Operations for Method One And Two	= 41.0399932861
Average No. of Basic Operations Saved in Method Two relative to Tanaka Method	= 4.7799987793
Average No. of Basic Operations Saved in Method One And Two relative to Tanaka Method	= -5.4400024414
Average No. of Times Method Two Failed and Method One And Two was called	= 0.2599999905

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.

Example 3.1

For the moduli system $M=\{23,19,17,13,11,7,5,3,2\}$ ($M=223092870$), divide x=10304312 by y=1401.

Method 1

TABLE OF THE FORM $m_{\ell+1}m_{\ell+2}\cdots m_{k-1}$

The total number of entries = $(n-2)\frac{(n-1)}{2} = 28$.

TABLE 3.3

Moduli	Product
$m_2m_3m_4m_5m_6m_7m_8$	4849845
$m_2m_3m_4m_5m_6m_7$	1616615
$m_2m_3m_4m_5m_6$	323323
$m_2m_3m_4m_5$	46189
$m_2m_3m_4$	4199
m_2m_3	323
m_2	19
$m_3m_4m_5m_6m_7m_8$	255255
$m_3m_4m_5m_6m_7$	85085
$m_3m_4m_5m_6$	17017
$m_3m_4m_5$	2431
m_3m_4	221

Moduli	Product
m_3	17
$m_4m_5m_6m_7m_8$	15015
$m_4m_5m_6m_7$	5005
$m_4m_5m_6$	1001
m_4m_5	143
m_4	13
$m_5m_6m_7m_8$	1155
$m_5m_6m_7$	385
m_5m_6	77
m_5	11
$m_6m_7m_8$	105
m_6m_7	35
m_6	7
m_7m_8	15
m_7	5
m_8	3

Table 3.4 lists the values for the quantities of the form $\frac{\alpha_k}{\beta_\ell}$, etc. The table is shown with 329 entries; however only 276 of these values need actually be stored since the rest are zero. We first perform mixed radix conversion (MRC) on y:

$$y \leftrightarrow \langle 0,0,0,0,0,0,3,3,21 \rangle$$

Hence, the value of $\ell=3$, and the value of $\beta_\ell=3$.

TABLE 3.4

$\beta_\ell \backslash \alpha_k$	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23
1	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23
2	0	1	1	2	2	3	3	4	4	5	5	6	6	7	7	8	8	9	9	10	11	11	11
3	0	0	1	1	1	2	2	2	3	3	3	4	4	4	5	5	5	6	6	6	7	7	7
4	0	0	0	1	1	1	1	2	2	2	2	3	3	3	3	4	4	4	4	5	5	5	5
5	0	0	0	0	1	1	1	1	1	2	2	2	2	2	3	3	3	3	3	4	4	4	4
6	0	0	0	0	0	1	1	1	1	1	1	2	2	2	2	2	2	3	3	3	3	3	3
7	0	0	0	0	0	0	1	1	1	1	1	1	1	2	2	2	2	2	2	2	3	3	3
8	0	0	0	0	0	0	0	1	1	1	1	1	1	1	1	2	2	2	2	2	2	2	2
9	0	0	0	0	0	0	0	0	1	1	1	1	1	1	1	1	1	2	2	2	2	2	2
10	0	0	0	0	0	0	0	0	0	1	1	1	1	1	1	1	1	1	1	2	2	2	2
11	0	0	0	0	0	0	0	0	0	0	1	1	1	1	1	1	1	1	1	1	1	2	2
12	0	0	0	0	0	0	0	0	0	0	0	1	1	1	1	1	1	1	1	1	1	1	1
13	0	0	0	0	0	0	0	0	0	0	0	0	1	1	1	1	1	1	1	1	1	1	1
14	0	0	0	0	0	0	0	0	0	0	0	0	0	1	1	1	1	1	1	1	1	1	1
15	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	1	1	1	1	1	1	1	1
16	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	1	1	1	1	1	1	1
17	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	1	1	1	1	1	1
18	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	1	1	1	1	1
19	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	1	1	1	1
20	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	1	1	1
21	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	1	1
22	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	1
23	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1

Iteration 1

We perform MRC on x

$x \leftrightarrow <0,0,1,2,7,9,0,12,13>$

The value of k=7, and the values of α_k=1. Since k>ℓ+1, we look-up Table 3.3 to find $m_4m_5m_6$=1001. From Table 3.4, we find $\frac{m_\ell}{(\beta_\ell+1)}=4$. Therefore $z(1)=\alpha_k \times m_4m_5m_6 \times \frac{m_\ell}{(\beta_\ell+1)}=4004$.

$x(1)=x(0)-yz(1)=10304312-140\times4004=4694708$.

Iteration 2

We perform MRC on x(1)

$x(1) \leftrightarrow <0,0,0,4,4,7,16,0,17>$

The value of k=6, and the value of α_k=4. Since k>ℓ+1, we look-up table 3.3 to find m_4m_5=143. From Table 3.4, we find $\frac{m_\ell}{(\beta_\ell+1)}=4$. Therefore $z(2)=\alpha_k \times m_4m_5 \times \frac{m_\ell}{(\beta_\ell+1)}=2288$.

$x(2)=x(1)-y\times z(2)=4694708-1401\times2288=1489220$.

Iteration 3

We perform MRC on x(2)

$x(2) \leftrightarrow <0,0,0,1,4,5,7,15,16>$

The value of k=6, and the value of α_k=1. Since k>ℓ+1, we look-up Table 3.3 to find m_4m_5=143. From Table 3.4, we find $\frac{m_\ell}{(\beta_\ell+1)}=4$. Therefore $z(3)=\alpha_k \times m_4m_5 \times \frac{m_\ell}{(\beta_\ell+1)}=572$.

$x(3)=x(2)-y\times z(3)=1489220-1401\times572=687848$.

Iteration 4

We perform MRC on x(3)

$x(3) \leftrightarrow <0,0,0,0,7,1,10,0,10>$

The value of k=5, and the value of α_k=7. Since k>ℓ+1, we look-up Table 3.3 to find m_4=13. From Table 3.4, we find $\frac{m_\ell}{(\beta_\ell+1)}=4$.

Therefore $z(4)=\alpha_k \times m_4 \times \frac{m_\ell}{(\beta_\ell+1)} = 364.$

$x(4)=x(3)-y\times z(4)=687848-1401\times 364=177884$

Iteration 5

We perform MRC on x(4)

$x(4) \leftrightarrow <0,0,0,0,1,10,16,1,2>$

The value of k=5, and the value of $\alpha_k=1$. Since $k>\ell+1$, we look-up Table 3.3 to find $m_4=13$. From Table 3.4, we find $\frac{m_\ell}{(\beta_\ell+1)}=4$.

Therefore $z(5)=\alpha_k \times m_4 \times \frac{m_\ell}{(\beta_\ell+1)} = 52.$

$x(5)=x(4)-y\times z(5)=177884-1401\times 52=105032.$

Iteration 6

We perform MRC on x(5)

$x(5) \leftrightarrow <0,0,0,0,1,1,2,6,14>$

The value of k=5, and the value of $\alpha_k=1$. Since $k>\ell+1$, we look-up Table 3.3, to find $m_4=13$. From Table 3.4, we find $\frac{m_\ell}{(\beta_\ell+1)}=4$.

Therefore $z(6)=\alpha_k \times m_4 \times \frac{m_\ell}{(\beta_\ell+1)} = 52.$

$x(6)=x(5)-y\times z(6)=105032-1401\times 52=32180.$

Iteration 7

We perform MRC on x(6)

$x(6) \leftrightarrow <0,0,0,0,0,4,5,12,3>$

The value of k=4, and the value of $\alpha_k=4$. Since $k=\ell+1$, we look-up Table 3.4, to find $\frac{m_\ell}{(\beta_\ell+1)}=4$. Therefore $z(7)=\alpha_k \times \frac{m_\ell}{(\beta_\ell+1)} = 16.$

$x(7)=x(6)-y\times z(7)=32180-1401\times 16=9764.$

Iteration 8

We perform MRC on x(7)

$x(7) \leftrightarrow <0,0,0,0,0,1,5,6,12>$

The value of k=4, and the value of $\alpha_k=1$. Since $k=\ell+1$, we look-up

Table 3.4, to find $\frac{m_\ell}{(\beta_\ell+1)}=4$. Therefore $z(8)=\alpha_k \times \frac{m_\ell}{(\beta_\ell+1)}=4$.

$x(8)=x(7)-y\times z(8)=9764-1401\times 4=4160$.

Iteration 9

We perform MRC on x(8)

$x(8) \leftrightarrow <0,0,0,0,0,0,9,9,20>$

The value of k=3, and the value of $\alpha_k=9$. Since $k=\ell$, we look-up Table 3.4, to find $\frac{\alpha_k}{(\beta_\ell+1)}=2$. Therefore $z(9)=\frac{\alpha_k}{(\beta_\ell+1)}=2$.

$x(9)=x(8)-y\times z(9)=4160-1401\times 2=1358$.

Iteration 10

We perform MRC on x(9)

$x(9) \leftrightarrow <0,0,0,0,0,0,3,2,1>$

The value of k=3 and the value of $\alpha_k=3$. Since $k=\ell$, we look-up Table 3.4 to find $\frac{\alpha_k}{(\beta_\ell+1)}=0$. Therefore $z(10)=\frac{\alpha_k}{(\beta_\ell+1)}=0$.

Since the value of z(10) has become zero, we terminate. According to the stopping criteria, since $z_r=0$ and $x_{r-1}<y$, the value of $z_r'=0$. Therefore

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

$$=7354+0=7354$$

which is the correct quotient. The total number of iterations is 10 and the total number of basic operations is 212. For the same operands and residue system, the Tanaka algorithm took 11 iterations, and the number of basic operations was 245. The number of basic operations saved using our algorithm (Method 1) is equal to 32, which represents a saving of about 13 per cent.

Method 2

We perform MRC on y

$y \leftrightarrow <0,0,0,0,0,0,3,3,21>$

The value of $\ell=3$ and the values of $\beta_\ell=3$.

Iteration 1

We perform MRC on x(0)=x

$x(0) \leftrightarrow <0,0,1,2,7,9,0,12,13>$

The value of k=7, and the value of $\alpha_k=1$. Since $k>\ell+1$, we look-up Table 3.3, to find $m_4m_5m_6=1001$. From Table 3.4, we find $\frac{m_\ell}{\beta_\ell}=5$.

Therefore, $z(1)=\alpha_k \times m_4m_5m_6 \times \frac{m_\ell}{\beta_\ell}=5005$.

$x(1)=x(0)-y \times z(1)=10304312-1401 \times 5005=3292307$.

Iteration 2

We perform MRC on x(1)

$x(1) \leftrightarrow <0,0,0,3,1,1,2,16,18>$

The value of k=6, and the value of $\alpha_k=3$. Since $k>\ell+1$, we look-up Table 3.3, to find $m_4m_5=143$. From Table 3.4, we find $\frac{m_\ell}{\beta_\ell}=5$.

Therefore $z(2)=\alpha_k \times m_4m_5 \times \frac{m_\ell}{\beta_\ell}=2145$.

$x(2)=x(1)-y \times z(2)=3292307-1401 \times 2145=287162$.

Iteration 3

We perform MRC on x(2)

$x(2) \leftrightarrow <0,0,0,0,2,12,11,2,7>$

The value of k=5 and the value of $\alpha_k=2$. Since $k>\ell+1$, we look-up Table 3.3 to find $m_4=13$. From Table 3.4, we get $\frac{m_\ell}{\beta_\ell}=5$.

Therefore $z(3)=\alpha_k \times m_4 \times \frac{m_\ell}{\beta_\ell}=130$.

$x(3)=x(2)-y \times z(3)-287162-1401 \times 130=105032$.

Iteration 4

We perform MRC on x(3)

$x(3) \leftrightarrow <0,0,0,0,1,1,2,6,14>$

The value of k=5, and the value of α_k=1. Since k>ℓ+1, we look-up Table 3.3, to find m_4=13. From Table 3.4, we find $\frac{m_\ell}{\beta_\ell}=5$. Therefore $z(4)=\alpha_k \times m_4 \times \frac{m_\ell}{\beta_\ell}=65$.

$x(4)=x(3)-y\times z(4)=105032-1401\times 65=13967$.

Iteration 5

We perform MRC on x(4)

$x(4) \leftrightarrow <0,0,0,0,0,1,14,18,6>$

The value of k=4 and the value of α_k=1. Since k=ℓ+1, we look-up Table 3.4, to find $\frac{m_\ell}{\beta_\ell}=5$. Therefore, $z(6)=\alpha_k \times \frac{m_\ell}{\beta_\ell}=5$.

$x(5)=x(4)-y\times z(5)=13967-1401\times 5=6962$.

Iteration 6

We perform MRC on x(5)

$x(5) \leftrightarrow <0,0,0,0,0,0,15,17,16>$

The value of k=3, and value of α_k=15. Since k=ℓ, we look-up Table 3.4 to find $\frac{\alpha_k}{\beta_\ell}=5$. Therefore, $z(6)=\frac{\alpha_k}{\beta_\ell}=5$.

$x(6)=x(5)-y\times z(6)=6962-1401\times 5=1223092827$.

Since x(6) is greater than x(5) we terminate the algorithm. Note that the value of x(6) represents a negative number in this residue system since $x(6)>\frac{M}{2}$. According to our emperical termination condition $z'_r=-1$.

Therefore, $z=[\frac{x}{y}]=\sum_{i=1}^{r} z(i)+z'$

$=7355-1=7354$.

The number of basic operations required is 136. Note that, in this particular case, the emperical correction factor $z'_r=-1$ yielded the correct quotient. However, this will not be true in general and Method 1 should be used to obtain the correct quotient. The total number of operations for obtaining the correct quotient would then become 136+212-32=316. The quantity "-32" appears because the mixed radix digits for x and y have already been computed and stored. It might appear, therefore, that whenever this situation is encountered the number of basic operations exceeds that required for the Tanaka algorithm. However, in actual computations it was found that, for the examples we performed, the Tanaka algorithm and Method 1 were slower on the average than the combined Method 1 - Method 2. Table 3.2 summarizes the comparison between Method Tanaka and Method One and Two for a typical moduli system, for cases that are known to be slow via our method and fast via the Tanaka method.

A flow-chart for Method 1 is shown in Fig. 3.1 and that for Method 2 is shown in Fig. 3.2.

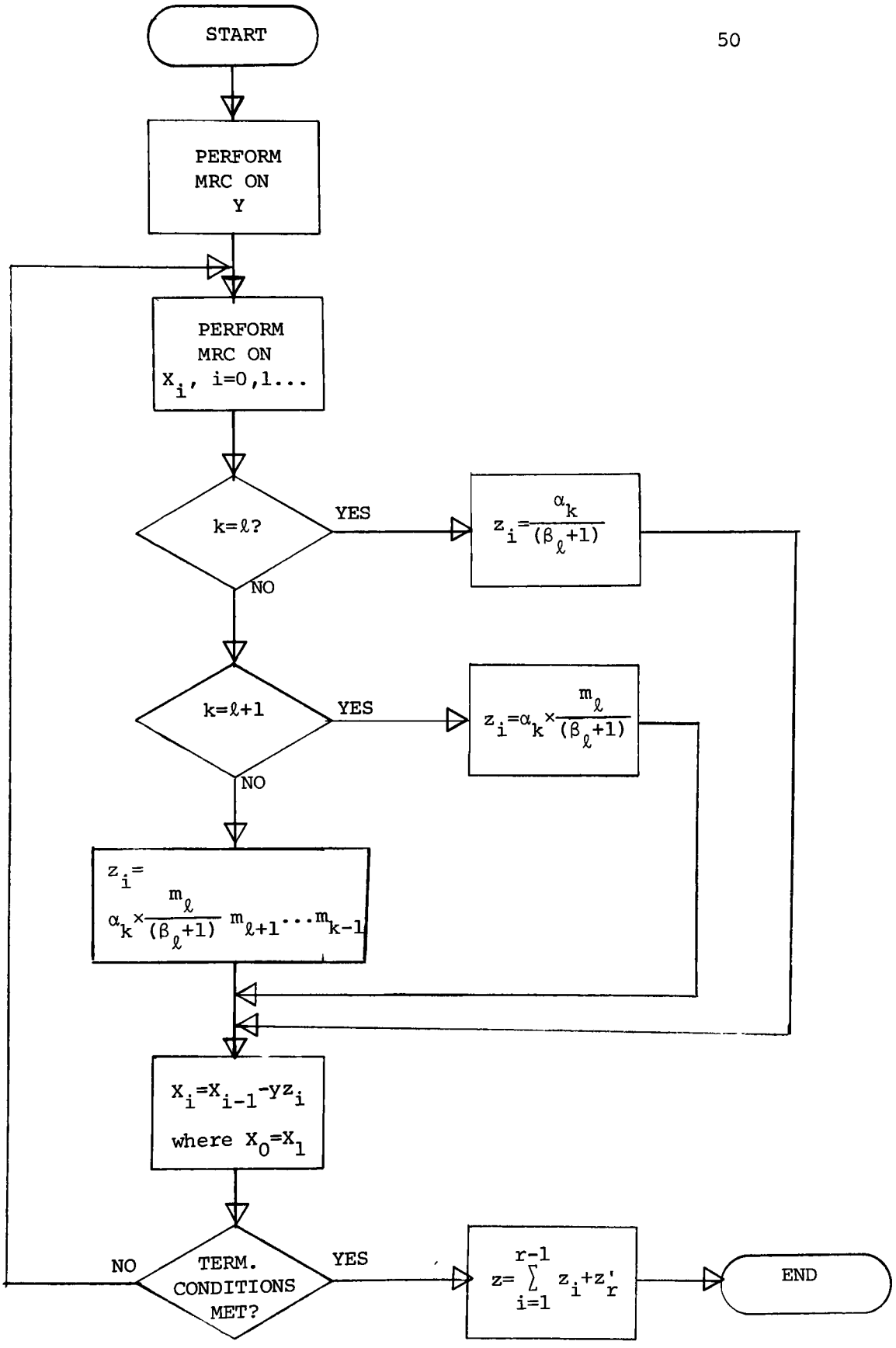

Fig. 3.1 Flowchart for Method 1

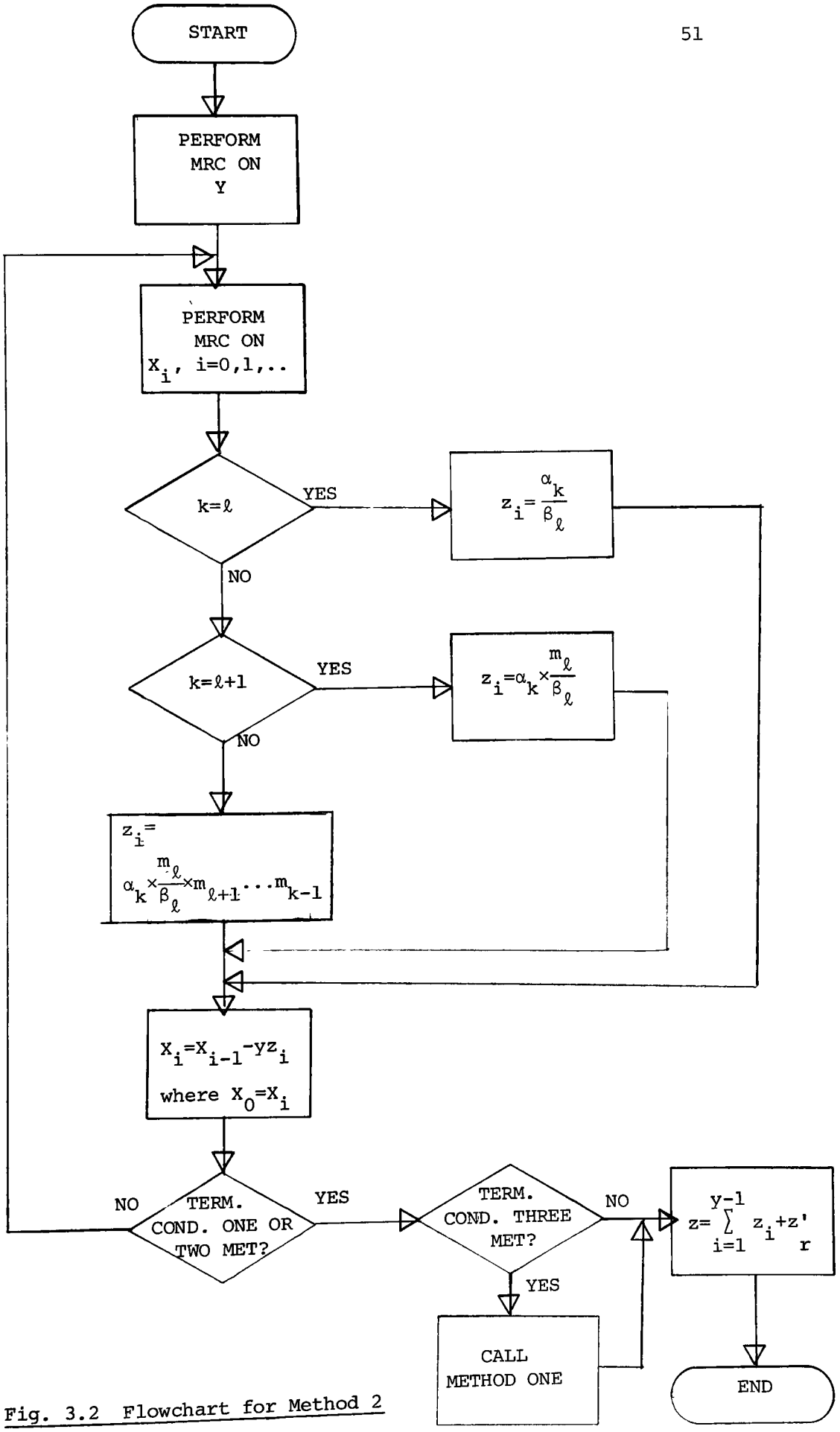

Fig. 3.2 Flowchart for Method 2

APPENDIX

We give here the complete mathematical proofs for Method One and partial proof for Method Two. We are unable to obtain analytically the correction factor for termination condition (2.7) described in Chapter 3. We want to show that the iterations

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

and

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

converge to yield the quotient z, for a certain r

$$z = \left[\frac{x}{y}\right] = \sum_{i=1}^{r-1} z_i + z'_r$$

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 show that $x_i = 0$ for some i. This follows from the fact that

a) $x_i \geq 0$ for all i.

and b) $x_0, x_1, \ldots, x_{i-1}, x_i, \ldots$ 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

$$x_i = x_{i-1} - y\left[\frac{\bar{x}_{i-1}}{\bar{\bar{y}}}\right] \geq x_{i-1} - \bar{y}\,\frac{\bar{x}_{i-1}}{\bar{\bar{y}}}$$

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 $\bar{\bar{y}} > 0$. Therefore, $z_i = \left[\frac{\bar{x}_{i-1}}{\bar{\bar{y}}}\right] \geq 0$. But $z_i \neq 0$ by assumption. Hence, $z_i > 0$. Therefore, $x_1 < x_0$, $x_2 < x_1$, ..., $x_i < x_{i-1}$

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 $z_i=[\frac{\bar{x}_{i-1}}{\bar{y}}]=0$. Therefore, either x_i or z_i becomes zero after a finite number of iterations.

Next, we will show that

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

From (3.1), we have:

$$x_1=x_0-yz_1$$

$$x_2=x_1-yz_2$$

$$\ldots\ldots$$

$$x_{r-1}=x_{r-2}-yz_{r-1}$$

$$x_r=x_{r-1}-yz_r$$

Adding, we have:

$$x_r=x_0-y(\sum_{i=1}^{r-1} z_i+z_r)$$

Since $x_0=x$, we get:

$$\frac{x}{y}=\sum_{i=1}^{r-1} z_i+z_r+\frac{x_r}{y}$$

Hence $[\frac{x}{y}]=\sum_{i=1}^{r-1} z_i+z_r+[\frac{x_r}{y}]$

$$=\sum_{i=1}^{r-1} z_i+z'_r$$

where $z'_r=z_r+[\frac{x_r}{y}]$

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 $[\frac{x}{y}] = \sum_{i=1}^{r-1} z_i + z_r$

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

$$\bar{x}_{r-1} < \bar{y} \text{ and } x_{r-1} \geq y \text{ implies that } x_r < 2y.$$

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

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

where, $\bar{\alpha}_j = \alpha_{j-1} m_1 m_2 \ldots m_{j-2} + \ldots + \alpha_2 m_1 + \alpha_1$.

Similarly, let

$$y = \beta_k m_1 m_2 \ldots m_{k-1} + \bar{\beta}_k, \text{ where } \bar{\beta}_k = \beta_{k-1} m_1 m_2 \ldots m_{k-2} + \ldots + \beta_2 m_1 + \beta_1.$$

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

$$\alpha_j m_1 m_2 \ldots m_{j-1} < (\beta_k + 1) m_1 m_2 \ldots m_{k-1}$$

or $$\alpha_{k+1} m_1 m_2 \ldots m_k < (\beta_k + 1) m_1 m_2 \ldots m_{k-1}$$

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

$$\alpha_k m_1 m_2 \ldots m_{k-1} < (\beta_k + 1) m_1 \ldots m_{k-1}.$$

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

Suppose $\alpha_k=\beta_k-i$, $i=1,2,\ldots$

$x_{r-1}\geq y$ implies

$$\alpha_k m_1 m_2\ldots m_{k-1}+\bar{\alpha}_k \geq \beta_k m_1 m_2\ldots m_{k-1}+\bar{\beta}_k$$

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

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

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

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

$$=m_1 m_2\ldots m_{k-1}-1.$$

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

This means the left-hand side of relation (A3) is less than zero. However, the right-hand side of (A3) $\bar{\beta}_k$ 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

$$x_{r-1}=\alpha_k m_1\ldots m_{k-1}+\bar{\alpha}_k$$

and $$2y=2\beta_k m_1\ldots m_{k-1}+2\bar{\beta}_k$$

$$=2\alpha_k m_1\ldots m_{k-1}+2\bar{\beta}_k$$

But $$\bar{\alpha}_k < m_1 m_2\ldots m_{k-1}$$

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

$$y=\beta_k m_1\cdots m_{k-1}+\cdots+\beta_2 m_1+\beta_1$$

$$\bar{y}=(\beta_k+1)m_1\cdots m_{k-1}$$

$$2y=2\beta_k m_1\cdots m_{k-1}+\cdots+2\beta_2 m_1+2\beta_1$$

Therefore, $2y-\bar{y}=(\beta_k-1)m_1m_2\cdots m_{k-1}+\cdots+2\beta_2 m_1+2\beta_1$. Except for the case $\beta_k=1$ and $\beta_i=0$, for $1\le i\le k-1$, we have

$$2y-\bar{y}>0 \quad \text{or} \quad \bar{y}<2y.$$

The case in which $\beta_k=1$ and $\beta_i=0$, $1\le i\le k-1$, represents a special divisor (product of moduli). In this case the scaling method should be used to obtain the quotient.

$$\bar{y}=\beta_k m_1m_2\cdots m_{k-1}+m_1m_2\cdots m_{k-1}$$

$$y=\beta_k m_1m_2\cdots m_{k-1}+\bar{\beta}_k$$

It is easily shown (as in the case of $\bar{\alpha}_k$) that

$$\bar{\beta}_k<m_1m_2\cdots m_{k-1}.$$

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 $x_{r-1}\ge y$ for some r.

Then $z_r=[\frac{\bar{x}_{r-1}}{\bar{y}}]=0 \Rightarrow \bar{x}_{r-1}<\bar{y}\le y.$

Also $x_r=x_{r-1}\le 2\bar{x}_{r-1}<2y$

Hence, we have $y\le x_{r-1}=x_r<2y$

or $1\le\frac{x_r}{y}<2$

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: $z_r=0$ and $x_{r-1}<y$

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] Eartman, 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", Lockheed 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, Informations 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] Keir, Y.A., Cheney, P.W., and Tannendaum, 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 Report, 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.

[18] Gregory, R.T., and Matula, D.W., "Base Conversion in residue number systems", BIT, Bind 17, Hefte Np. 3, 1977, pp. 286-302.