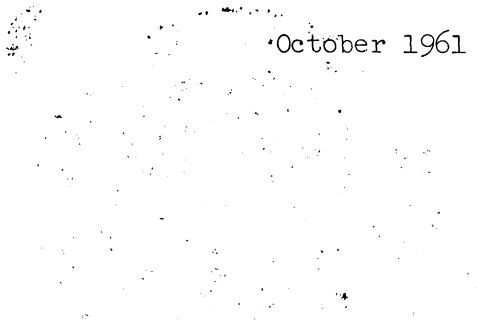
ASD TECHNICAL REPORT 61-483

RESIDUE NUMBER SYSTEMS FOR COMPUTERS

H. L. Garner
R. F. Arnold
B. C. Benson
C. G. Brockus
R. J. Gonzalez
D. P. Rozenberg

The University of Michigan



October 1961

Electronic Technology Laboratory

Contract No. AF 33(616)-7340
Project No. 7062, Task No. 706205

AERONAUTICAL SYSTEMS DIVISION
AIR FORCE SYSTEMS COMMAND
UNITED STATES AIR FORCE
WRIGHT-PATTERSON AIR FORCE BASE, OHIO

FOREWORD

This report represents the results of research performed by the group at The University of Michigan under the direction of Professor H. L. Garner. Concurrently, research on the same subject was being conducted at Harvard University under the direction of Professor Howard Aiken, and at the Lockheed Missile System Division under the direction of Dr. Richard Tanaka.

There was a considerable exchange of information among the above groups during the course of the research effort. The efforts attained exhibit little overlap, rather they are complementary.

A portion of this report was extracted from the doctoral dissertation of D. P. Rozenberg. His work was supported by this contract, and led to the Ph.D.

ABSTRACT

The purpose of the research performed under this contract was to investigate the feasibility of residue number systems in their applications to digital computers.

The problems of such an application are the ones of magnitude determination, sign determination, overflow, scaling, and division. These problems are not independent, but are found to be quite interrelated.

A theoretical treatment of residue number systems is given which lays the foundation for a unified study of the complete problem. Treatments of an organizational nature are given which deal with multiplication, division, and scaling. The matter of correlating the theoretical and organizational studies to physical realizations involving networks is treated also.

The question of whether the residue number system can be successfully applied to general purpose computers is still an open one. Their application to special purpose machines is considered both feasible and practical.

TABLE OF CONTENTS

TABLE OF CONTENTS (Continued)

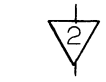
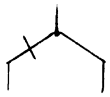
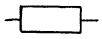
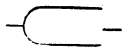
TABLE OF CONTENTS (Concluded)

LIST OF TABLES

LIST OF FIGURES

LIST OF FIGURES (Concluded)

LIST OF SYMBOLS

$[x]$	The integer part of x (*)
$\{x\}$	The fractional part of x (*)
$(a,b) = c$	The greatest common divisor of a and b is c (*)
$\lvert x \rvert_m$	The residue of x with respect to m
$a \mid b$; $(a \nmid b)$	a divides b; (a does not divide b)
$a \equiv b \bmod m$	a is congruent to b modulo m
$a \cong b$	a is isomorphic to b
$a \simeq b$	a is on the order of b (is approximately equal to b)
x^*, $\overline{x}$	The modular complement of x
$G: R^n \rightarrow S^m$	G is a linear transformation from R^n into S^m
$a \longleftrightarrow a'$	a corresponds to a'
	Two levels of switching elements
	Switching element
	Delay element
	Binary Encoder
	Modulo five operation

(*)The brackets are also used at times to indicate association. The square bracket is used also to indicate a matrix. The particular meaning should be clear from context.

CHAPTER I

INTRODUCTION

1.1 INTRODUCTION TO THE PROBLEM

The aspects of the residue class of the number system which makes this class of number systems suitable for high-speed computation has been long known to mathematicians. The first consideration of the residue class of numbers for a computer system is due to Valach.[1] This initial work was soon followed by a paper by Svoboda,[2] and later a paper by Valsch and Svoboda.[3] Svoboda must be given credit for the most fundamental and comprehensive study of the application of residue class to practical computing machines. In his paper, "Rational Number Systems of Residue Classes," he considers both fractional and integer number systems. The residue number system is most naturally interpreted as an integer system, though the fractional interpretation allows greater flexibility with respect to the problem of scaling. The paper by Garner[4] considers only the integer interpretation of the residue number system and suggests a unified approach to the problem of sign detection. The first part of this research report is a continuation of the study of the vector-space approach to the sign detection problem.

Carry propagation accounts for the major portion of execution time in a conventional arithmetic system. The residue number system, based on the algebra

---

Manuscript released by the University of Michigan, October 1961, for publication as an ASD Technical Report.

of residue classes, permits addition and multiplication to be performed without the existence of carries. The main advantage to be gained from the residue number system follows from the fact that it should be possible to execute multiplication as rapidly as addition. The usefulness of the residue number system for general purpose computation is still an open question. However, there exist many favorable applications of a special nature. Cheney[5] has designed a digital correlator based on the residue number system and has estimated that a correlator of similar accuracy, based on the binary system, would have been 10 to 100 times slower, depending on whether the organization were parallel or serial. The residue number system has several characteristics which have prevented the adoption of this system for general purpose computation. The problems of sign detection, the detection and handling of additive and multiplicative overflow, and division are of extreme practical importance. In this report the algebraic and numeric properties of the residue number system are investigated in order to determine algorithmic solutions for the above problems. Considerable attention has been given to the nature of possible algorithmic solutions. This is, in many cases, more important than the actual algorithms.

In the remainder of this chapter, the residue number system is shown to be a ring. Various well-known ring theorems are interpreted in terms of the residue number system to provide fundamental information concerning the problems of sign detection, the representation of negative numbers, and division. Other theorems provided in this section relate to the construction of residue number systems.

In Chapter II, the basic algebraic structure required for the investigation of the residue number system and other number systems is developed. In Chapter III the carry structure of the class of non-redundant number systems is determined. It is shown that the residue number system is a member of the class of carry-type number systems, for which the carries are congruent to zero modulo base of the number system. In Chapter IV, the characteristics of the mixed-base number system are determined. In Chapter V, particular attention is given to the problem of sign detection. It is shown that the only non-redundant number system with simple sign detection characteristics is the mixed-base number system. This fact is used to establish a general definition for the sign detection process. In Chapter VI the problem of multiplication for a residue number system is studied. In particular, two different multiplication schemes are presented. The first is based on the decomposition of the ring structure. This obtains a simple multiplication algorithm at the expense of complicating the addition algorithm. The second approach uses redundancy to speed the multiplication process and is termed A-coding. Both of the techniques offer advantage over conventional residue coding when the moduli are large.

In Chapter VII, division for a residue number system is considered. Two division schemes are presented. One is based on a Newton-Raphson iterative procedure, while the other depends on approximation to the integral part of the quotient. In Chapter VIII specific attention is given to the problem of multiplicative overflow. Various schemes are considered and a special purpose application of the residue number system is given. In Chapter IX the problem of sign detection is considered in terms of physical realization.

1.2 THE RESIDUE NUMBER SYSTEM

As a preliminary step in the introduction of the residue number system, the ring I_M of integers modulo M will be discussed.[6] The elements of the ring are the integers 0, 1, 2, ..., M-1. Ring addition and ring multiplication of two elements of I_M are performed by reducing the conventional sum and product modulo M. That is, the sum or product is divided by M and the remainder is retained as the result. Thus, for example, the ring I_6 consists of the integers 0, 1, 2, 3, 4, 5. Dividing the ordinary sum of 4 and 5 by 6, one obtains 3 for the remainder. Therefore, in I_6, $4+5 = 3$. Similarly $2+4 = 0$, $2 \cdot 2 = 4$, and $4 \cdot 2 = 2$. The proof that I_M is actually a ring follows from the fact that in the division of an integer by M, the remainder is unique. The complete proof may be found in the literature. The order of I_M is M.

Consider the ring I of integers and the ring I_M of integers modulo M. An arbitrary element m of I determines a unique element $\bar{m}$ of I_M; namely, the remainder upon division by M. If one denotes the correspondence of m to $\bar{m}$ by $m \longleftrightarrow \bar{m}$, it is clear that if $m_1 \longleftrightarrow \bar{m}_1$, $m_2 \longleftrightarrow \bar{m}_2$, then $m_1+m_2 \longleftrightarrow \overline{m_1+m_2} = \bar{m}_1+\bar{m}_2$ and $m_1 m_2 \longleftrightarrow \overline{m_1 m_2} = \bar{m}_1 \bar{m}_2$. Therefore, the correspondence between I and I_M is a homomorphism.

Fixed point digital computers do not operate upon the ring, I, of integers but rather on the ring I_M. If one thinks of the radix point as being on the right, then the largest number which can be represented is M-1. That the homomorphism is a many-to-one mapping is painfully apparent when overflow occurs and the most significant digit(s) are lost. Fortunately, one can detect such an overflow by sensing the carry from the most significant digit position. The

ring I_M possesses an additive inverse of every element. If m is an element of I_M, M-m is the additive inverse. Thus we map the integers less than M/2 onto the elements of I_M less than M/2, and the negatives greater than -M/2 onto the remaining elements of I_M. Thus the sign is readily determined and thereby magnitude comparisons can be performed. Since one can perform overflow detection, sign determination, and magnitude comparisons, division is possible.

Let S_1 and S_2 be two rings and consider the ordered pairs of symbols (s_1,s_2) where $s_1 \epsilon S_1$ and $s_2 \epsilon S_2$. If one defines addition and multiplication to be

$$(s_1,s_2) + (t_1,t_2) = (s_1 + t_1, s_2 + t_2)$$

and

$$(s_1,s_2)(t_1,t_2) = (s_1t_1, s_2t_2)$$

this set of ordered pairs becomes the ring S termed the direct sum of S_1 and S_2 and denoted $S_1 \oplus S_2$. In the above definition the operations s_i+t_i and s_it_i are the ring operations of S_i.

An especially important theorem is the following:

Theorem I.[6] If a ring S has positive characteristic $n = n_1 \cdot n_2$ where n_1 and n_2 are greater than 1 and relatively prime, then S is isomorphic ($\cong$) to $S_1 \oplus S_2$ where S_i is a ring of characteristic $n_i (i = 1,2)$.

This theorem states, for instance, that I_6 is isomorphic to the direct sum $I_2 \oplus I_3$ with the following mapping:

$0 \longleftrightarrow (0.0)$	$3 \longleftrightarrow (1.0)$
$1 \longleftrightarrow (1,1)$	$4 \longleftrightarrow (0,1)$
$2 \longleftrightarrow (0,2)$	$5 \longleftrightarrow (1,2)$

Theorem I may be extended as follows:

Theorem II. If the ring I_M has positive characteristic $M = m_1m_2 \ldots m_n$

where the m_i are integers greater than 1 and pairwise prime, then

$$I_M \cong I_{m_1} \oplus I_{m_2} \oplus \ldots \oplus I_{m_n}.$$

Proof: If n = 2, the result follows from Theorem I. Assume the result for n = k and consider $M = m_1 m_2 \ldots m_k m_{k+1}$. M may be rewritten $M = m_1 m_2 \ldots m_k'$, where $m_k' = m_k m_{k+1}$. Thus we have

$$I_M \cong I_{m_1} \oplus I_{m_2} \oplus \ldots \oplus I_{m_k'}.$$

but by Theorem I,

$$I_{m_k'} \cong I_{m_k} \oplus I_{m_{k+1}}.$$

Therefore,

$$I_M \cong I_{m_1} \oplus I_{m_2} \oplus \ldots \oplus I_{m_{k+1}}.$$

Thereby, the conclusion is proved.

Theorem II completely defines the residue number system. Elements of I_M are mapped into n-tuples of the direct sum (elements of the residue number system) according to the following scheme:

$$x \longleftrightarrow (x \bmod m_1, x \bmod m_2, \ldots, x \bmod m_n). \qquad (1\text{-}1)$$

If

$$x \longleftrightarrow (x_1, x_2, \ldots, x_n),$$

then

$$x + y \longleftrightarrow (x_1 + y_1 \bmod m_1, \ldots, x_n + y_n \bmod m_n)$$

and

$$x \cdot y \longleftrightarrow (x_1 y_1 \bmod m_1, \ldots, x_n y_n \bmod m_n).$$

It is clear from both the example following Theorem I and Expression (1-1) that the components of the residue representations have no positional significance.

Consider the following examples of additions in the ring I_{210} with $m_1 = 2$ $m_2 = 3$, $m_3 = 5$, and $m_4 = 7$.

(a)	$209 \longleftrightarrow (1,2,4,6)$	(b)	$209 \longleftrightarrow (1,2,4,6)$
	$\underline{1 \longleftrightarrow (1,1,1,1)}$		$\underline{105 \longleftrightarrow (1,0,0,0)}$
	$210 \longleftrightarrow (0,0,0,0)$		$104 \longleftrightarrow (0,2,4,6)$
(c)	$105 \longleftrightarrow (1,0,0,0)$		$23 \longleftrightarrow (1,2,3,1)$
	$\underline{120 \longleftrightarrow (0,0,0,1)}$		$\underline{162 \longleftrightarrow (1,2,2,6)}$
	$015 \longleftrightarrow (1,0,0,1)$		$185 \longleftrightarrow (0,1,0,0)$

In example (a), the sum produces an overflow and each component ring indicates an overflow. The sum in (b) overflows but only one component overflows. In (c), overflow occurs but the component subrings do not indicate overflow. No overflow accompanies the addition in (d); however, overflow is present in each component. These examples indicate that overflow in the component subrings is unrelated to overflow of I_M. Similar statements may be made concerning multiplicative overflow.

1.3 THEOREMS ON THE BASIC PROPERTIES OF RESIDUE NUMBER SYSTEMS

Theorem III. The period of a residue class number system is equal to the least common multiple (lcm) of the bases.

Proof: Given bases $m_1,\ldots,m_n$. Assume $m_i \mid m_j$, $i \neq j$; then the digits associated with modulus $m_i m_j$ have a period equal to m_j. Thus, m_i has no effect on the period and may be dropped from further consideration. The process may be continued for remaining bases until for all pairs $m_i \mid m_j$, $i \neq j$. The period is given by the product $M = \prod_{k=1}^{n} m_k$. The product is the lcm of the original bases since, by the fundamental theorem of arithmetic, the product of the bases is

$$P = p_1^{\alpha_1} \; p_1^{\alpha_2} \ldots p_1^{\alpha_q} \; p_2^{\beta_1} \; p_2^{\beta_2} \ldots p_2^{\beta_r} \ldots p_m^{\delta_s}$$

the lcm of P, denote by <P>, is

$$\langle P \rangle = P_1^{\alpha \max} \; p_2^{\beta \max} \ldots p_m^{\delta \max}.$$

Obviously, <P> = M.

Corollary: The maximum length period is obtained when the bases are relatively prime by pairs. i.e., $(m_i, m_j) = 1$.

Theorem IV. The replacement of all residue digits, r_i, associated with base m_i by the modulo m_i product $\left|\alpha r_i\right|_{m_i}$ where $\alpha \neq 0$ and $\alpha < m_i$, yields a number system with period M', where

$$M' = \frac{M}{(\alpha, m_i)}$$

Proof:

1. $N \equiv a_i \bmod m_i$
2. $\alpha N \equiv \alpha a_i \bmod m_i$
3. $\dfrac{\alpha N}{(\alpha, m_i)} \equiv \dfrac{\alpha a_i}{(\alpha, m_i)} \quad \left(\bmod \dfrac{m_i}{(\alpha, m_i)}\right)$
4. $M' = m_1 \ldots \dfrac{m_i}{(\alpha, m_i)} \ldots m_n = \dfrac{M}{(\alpha, m_i)}$

Theorem V.

The successor of X an element of a residue number system R may be defined as

$$X' = X + \delta \qquad \begin{array}{l} X' \in R \\ X \in R \\ \delta \in R \end{array}$$

The period of R is given by

$$M' = \frac{M}{(M, \delta)}$$

Proof:

For the standard residue number system

$$X' = X + 1$$

Multiplication by δ yields

$$X' = X + \delta$$

Hence, the change of successor is equivalent to multiplication of each element of the residue code by the residue representation for δ. This situation is covered by Theorem IV. It is necessary to verify that

$$(M,\delta) = (m_1,\alpha_1) \ldots (m_n,\alpha_n)$$

where δ is represented in residue code by $\alpha_1,\ldots,a_n$. But this is obvious since

$$(M,\delta) = (m_1,\delta) \ldots (m_n,\delta) = (m_1,\alpha_1) \ldots (m_n,\alpha_n)$$

$$(a_1a_2,b) = (a_1,b)(a_2,b) \Longrightarrow (a_1,a_2) = 1$$

Theorem VI. Isomorphic mapping between the ring of integers I_M and the residue number system consisting of $I_{m_1},\ldots,I_{m_n}$ exists for any successor value $\delta,(\delta M) = 1$ if the operation is addition. If the operation is multiplication, the isomorphic mapping exists only if $\delta = 1$.

Proof:

$$\delta a + \delta b = (a + b)$$

but

$$(\delta a) \times (\delta b) \neq \delta(ab)$$

except when $\delta^2 \equiv \delta \bmod M$, $\delta = 1$ since $(\delta,M) = 1$.

Corollary: Isomorphic mapping is obtainable between I_M and $I_{m_1},\ldots,I_{m_n}$ if each multiplication is followed by a corrective multiplication by δ.

The results of Theorem VI, and the corollary indicate that the integer interpretation is the natural interpretation for the residue number system.

Any other interpretation will require multiplicative correction to preserve the interpretation.

Theorem VII. Consider the residue digits r_i and r_j associated with bases m_i and m_j; then the operation $\left|r_i - r_j\right|_{m_i}$ partitions the set of $m_i m_j$ residue digits into j subsets. Each subset has i elements. The subsets are ordered modulo m_i.

Proof:

1. $N \equiv r_i \bmod m_i$

   $N \equiv r_j \bmod m_j$

2. $N = r_i + m_i t_i$

   $N = r_j + m_j t_j$

3. $r_i - r_j = m_j t_j - m_i t_i$

4. $r_i - r_j \equiv m_j t_j \bmod m_i$

5. $(r_i - r_j)/m_j \equiv t_j \bmod m_i$

Note: If $N < m_i m_j$ then $t_j < m_i$ and $\left|(r_i - r_j)/m_j\right|_{m_k} = t$

Example: $m_1 = 2, m_3 = 7$

r_1	r_2	$\left\|r_1 - r_2\right\|_{m_1}$	$\left\|r_2 - r_1\right\|_{m_2}$	$4\left\|\left\|r_2 - r_1\right\|_{m_2}\right\|_{m_2}$
0	0	0	0	0
1	1	0	0	0
0	2	0	2	1
1	3	0	2	1
0	4	0	4	2
1	5	0	4	2
0	6	0	6	3
1	0	1	6	3
0	1	1	1	4
1	2	1	1	4
0	3	1	3	5
1	4	1	3	5
0	5	1	5	6
1	6	1	5	6

Theorem VIII. Consider the ring R. For every a and b elements of R

$$a^*b^* = ab$$

$$a^*b = (ab)^*$$

where a^* is the additive inverse of a defined as $a^*+a = 0$. Every element of a ring has a unique additive inverse. This is usually given as one of the postulates of a ring. This postulate and Theorem VIII taken together indicate two possible correspondences between the elements of the ring, the inverse elements of the ring and the positive and negative signs. The usual convention specifies that an element interpreted as an additive inverse of a represents -a. Thus the additive inverse provides a code for distinguishing positive and negative numbers. The code is valid for both addition and multiplication but is useful only when the ring is partitioned into elements always interpreted as magnitudes and elements always interpreted as inverses, and a simple means exists for determining the correct partition for a given element.

Theorem IX. For every a and b; $b \neq 0$, elements of a ring there exist x and y such that

$$a = bx + y$$

However x and y are not unique.

Theorem X. For every a and b; $b \neq 0$ there exist a unique x and y such that

$$a = bx + y$$

if $y < b$. $y < b$ may be interpreted in terms of the successor concept. Let $y = Z^{\alpha}$ and $b = Z^{\beta}$. Then $y < b$ if $\alpha|_m < |\beta|_m$. m is the cardinality of the ring.

Theorem IX and Theorem X provide some insight into the nature of the division algorithms associated with any finite number system. The residue number

system is included in the class of finite number systems. Observe that

$$a \equiv y \bmod b$$

and if $y < b$ then

$$|a|_b = y$$

Furthermore

$$\frac{a-|a|_b}{b} = \left[\frac{a}{b}\right] = x$$

CHAPTER II

THE R SPACE

There are many aspects of the residue number system which suggest the appropriateness and usefulness of the vector space model. This approach is pursued in this chapter. It will be shown that strictly speaking the residue number system is not a vector space. However, a development including much of the conventional vector space concept is a useful tool for the investigation of the residue and related number systems. The appropriate axiomatic system has been called the R space. The basic properties of the R space are developed in the chapter and are applied in the next three chapters to the basic problems of the residue number system. We have found no property of the residue number system which cannot be validated using either the R space approach or the number theoretic approach. However, depending on the nature of the problem, usually one approach is more suggestive.

2.1 BASIC PROPERTIES OF THE R SPACE

Let any two residue representations be

$$x = (x_1, x_2, \ldots, x_n)$$

$$y = (y_1, y_2, \ldots, y_n)$$

then $x+y \triangleq [x_1+y_1(m_1), \ldots, x_n+y_n(m_n)]$ where the m_j are pairwise relatively prime. The component x_i is said to be associated with the base modulus m_i. The residue number system representations from an additive Abelian group which is denoted by R^n. From S, a ring with identity, we select the integers and define multi-

plication by a scalar to be

$$ax \triangleq [ax_1(m_1),\ldots,ax_n(m_n)]$$

With these definitions it is seen that

a. $ax \epsilon R^n$,

b. $a(x+y) = ax + ay$

for

$$\begin{aligned} a(x+y) &= \{a[(x_1+y_1)(m_1)](m_1),\ldots,a[(x_n+y_n)(m_n)](m_n)\} \\ &= \{[(ax_1)(m_1) + (ay_1)(m_1)](m_1),\ldots, \\ &\quad [(ax_n)(m_n) + (ay_n)(m_n)](m_n)\} \\ &= ax + ay \end{aligned}$$

c. $(a+b)x = ax + bx$

since

$$\begin{aligned} (a+b)x &= [(a+b)x_1(m_1),\ldots,(a+b)x_n(m_n)] \\ &= \{[(ax_1)(m_1) + bx_1(m_1)](m_1),\ldots, \\ &\quad [(ax_n)(m_n) + bx_n(m_n)](m_n)\} \end{aligned}$$

d. $(ab)x = a(bx)$

for

$$\begin{aligned} (ab)x &= [(abx_1)(m_1),\ldots,(abx_n)(m_n)] \\ &= a[bx_1(m_1),\ldots,bx_n(m_n)] \\ &= a(bx), \text{ and} \end{aligned}$$

e. All elements of R^n are uniquely expressible as linear forms $a_1\epsilon_1 + \ldots + a_n\epsilon_n$ by means of a fixed basis elements with $0 \leqslant a_i \leqslant m_i$ where ϵ_j has one for its jth component and zero for the other components.

The five properties which must be satisfied for R^n to be a n-dimensional vector

space a, b, c, d above, and:

e'. All elements of R^n are uniquely expressible as linear forms $a_1u_1 + \ldots + a_nu_n$ by means of a fixed "basis elements" $u_1,\ldots,u_n$ and $a_i \in S$.

This property is not satisfied, and thus R^n is not a true vector space. The pseudo-vector space R^n will be called the R-space and all vector space terms which follow will have an interpretation in the R-space which is analogous to the vector space definitions.

Consider the set of vectors $\langle\alpha_1,\ldots,\alpha_m\rangle$ with each α_i having n components. Form a square array of the components by placing the components of α_i in the ith row. If this array can be made triangular (specifically a lower triangular array) by reordering rows and columns, the set $\langle\alpha_1,\ldots,\alpha_n\rangle$ is termed semi-triangular and the reordered set termed triangular.

Theorem XI. If the set $\{\alpha_1,\ldots,\alpha_n\}$ is triangular and the elements on the principal diagonal are relatively prime to the associated base moduli, then any linear form $a_1\alpha_1 + a_2\alpha_2 + \ldots + a_n\alpha_n$ where the a_i are integers can be uniquely expressed as

$$\sum_{i=1}^{n} c_i\alpha_i \text{ where } 0 \leqslant c_i < m_i.$$

Proof: $k_{nn}c_n \equiv k_{nn}a_n \bmod m_n$

where k_{ij} is the jth component of α_i.*

Thus $c_n \equiv a_n \bmod m_n$ since m_n is relatively prime to k_{nn}, and $0 \leqslant c_n < m_n$ is uniquely determined.

*$a \equiv b \bmod m$ (a is congruent to b modulo m) if and only if $m \mid a-b$ (m divides a-b or $a = b + km$).

Assume the conclusion true for c_j for $j = m, m+1,\ldots,n$ and also that these c_j have been determined and consider the congruence

$$k_{m-1,m-1}c_{m-1} + k_{m,m-1}c_m + \cdots + k_{n,m-1}c_n \equiv$$
$$k_{m-1,m-1}a_{m-1} + k_{m,m-1}a_m + \cdots + k_{n,m-1}a_n \bmod m_{m-1}.$$

By adding to both sides of the above congruence the additive inverse of $(k_{m,m-1}c_m + \cdots + k_{n,m-1}c_n) \bmod m_{m-1}$, this congruence takes on the form $k_{m-1,m-1}c_{m-1} = A \bmod m_{m-1}$. Since $(k_{m-1,m-1}, m_{m-1}) = 1$, c_{m-1} is uniquely determined mod m_{m-1} and

$$0 \leqslant c_{m-1} < m_{m-1}^*$$

Theorem XI is of key importance for it allows us to abandon the ring S and to concentrate our attention on linear forms of triangular sets of vectors; namely, $\sum_{i=1}^{n} c_i\alpha_i$ where $<\alpha_1,\ldots,\alpha_n>$ is triangular and $0 \leqslant c_i < m_i$. Except where indicated, the following discussion will be limited to sets of triangular vectors and linear combinations with restricted scalars.

Definition: The vectors $\alpha_1,\ldots,\alpha_n$ of a triangular array are <u>linearly independent</u> if and only if

$$c_1\alpha_1 + \cdots + c_n\alpha_n = 0 \quad \text{where } 0 \leqslant c_j < m_j$$

implies $c_1 = c_2 = \cdots = c_n = 0$. Otherwise the vectors $\alpha_1,\ldots,\alpha_n$ are termed <u>linearly dependent</u>.

<u>Theorem XII</u>. The triangular set of vectors $<\alpha_1,\alpha_2,\ldots,\alpha_n>$ is linearly independent if and only if each element on the principal diagonal is relatively prime to the associated base modulus.

---

*The greatest common divisor (d) of a and b is written (a,b) = d.

Proof: Consider the equation

$$c_1\alpha_1 + \ldots + c_n\alpha_n = 0. \qquad (2\text{-}1)$$

Equation (2-1) is equivalent to the following simultaneous linear congruences

$$k_{nn}c_n \equiv 0 \bmod m_n$$

$$k_{ii}c_i + \sum_{j=i+1}^{n} k_{ji}c_j \equiv 0 \bmod m_i$$

for

$$i = n-1,\ldots,1.$$

For $k_{nn}c_n \equiv 0 \bmod m_n$ to yield the unique solution $c_n = 0$, it is sufficient that $(k_{nn},m_n) = 1$. Assume $(k_{\ell\ell},m_\ell) = 1$ and $c_i = 0$ for $i = 2\ell + 1,\ \ell + 2,\ldots,n$. The ℓth congruence becomes $k_{\ell\ell}c_\ell \equiv 0 \bmod m_\ell$ and $c_\ell = 0$ is the unique solution. This proves the sufficiency of the hypothesis.

Assume r to be the least index for which $(k_{ii},m_i) \neq 1$. Let $c_i = 0$ for $i > r$. The above congruences become

$$k_{rr}c_r \equiv 0 \bmod m_r \qquad (2\text{-}2)$$

$$k_{ii}c_i + \sum_{j=i+1}^{r} k_{ji}c_j \equiv 0 \bmod m_i \quad \text{for } i = r-1,r-2,\ldots,1.$$

Congruence (2-2) may be solved with $c_r \neq 0$. Since $(k_{ii},m_i) = 1$ for $i = r-1,\ r-2,\ldots,1$, the remaining congruences assume the form

$$k_{ii}c_i \equiv A_i \bmod m_i$$

The condition $(k_{ii},m_i) = 1$ for $i < r$ guarantees the existence of a solution for each congruence. This completes the proof.

Definition: A set of vectors $\{\alpha_1, \alpha_2,\ldots,\alpha_n\}$ is said to span R^n if and only if there exists a set of coefficients c_i in the range $0 \leqslant c_i < m_i$ satisfying the equation

$$\sum_{i=1}^{n} c_i \alpha_i = r_i$$

where r_i is any residue number.

Theorem XIII. For a triangular set of vectors $< \alpha_1, \ldots, \alpha_n >$ to span R^n it is necessary and sufficient that each element on the principal diagonal be relatively prime to the associated base modulus.

Proof: The existence of solutions c_i to the following congruences will be shown

$$k_{nn} c_n \equiv a_n \bmod m_n \tag{2-3}$$

and $k_{ii} c_i + \sum_{j=i+1}^{n} k_{ji} c_j \equiv a_i \bmod m_i$ for all a_i in the range $0 \leqslant a_i < m_i$. The necessary and sufficient condition for the existence of a solution to (2-3) is $(k_{nn}, m_n) | a_n$. Since a_n (an integer) will range $0 \leqslant a_n < m_n$, it is necessary and sufficient for $(k_{nn}, m_n) = 1$. Assume that solutions c_i exist for $i > \ell$. These solutions are substituted into the ℓth congruence yielding an expression of the form

$$k_{\ell\ell} c_\ell + D \equiv a_\ell \bmod m_\ell$$

or

$$k_{\ell\ell} c_\ell \equiv a_\ell + E \bmod m_\ell$$

Again $(a_\ell + E) \bmod m_\ell$ will include all integers in the range 0 through $m_\ell - 1$. Thus to guarantee solution it is necessary and sufficient that $(k_{\ell\ell}, m_\ell) = 1$. The proof is complete.

Corollary: The triangular set of vectors $<\alpha_1, \ldots, \alpha_n>$ is a spanning set of R^n if and only if it is an independent set.

Definition: A basis of an R-space is a linearly independent set of vectors

which generate the R-space.

Theorem XIII may be rephrased in more conventional terms as follows:

Theorem XIV. For a triangular set of vectors $<\alpha_1,\ldots,\alpha_n>$ to be a basis of R^n it is necessary and sufficient that each element on the principal diagonal of the array be relatively prime to the associated modulus.

Theorem XV: If $\alpha_1,\ldots,\alpha_n$ forms a basis for R^n, then every vector $\beta \epsilon R^n$ has a unique expression

$$\beta = x_1\alpha_1 + x_2\alpha_2 + \ldots + x_n\alpha_n, \quad 0 \leqslant x_i < m_i.$$

Proof: It will be shown that the congruences which must be satisfied for the above expression to hold will provide x_j which are unique modulo m_j. Let $\beta = (b_1,b_2,\ldots,b_n)$.

The first congruence is $x_n k_{n,n} \equiv b_n \bmod m_n$. Since k_{nn} and m_n are relatively prime, x_n is uniquely specified modulo m_n. Assume $x_{m+1},x_{m+2},\ldots,x_n$ have been uniquely determined. Then to be considered is the congruence

$$k_{m,m}x_m + k_{m+1,m}x_{m+1} + \ldots + k_{n,m}x_n \equiv b_m \bmod m_m.$$

Add to each side the additive inverse modulo m_m of

$$(k_{m+1,m}x_{m+1} + \ldots + k_{n,m}x_n)$$

to obtain

$$k_{m,m}x_m \equiv B \bmod m_n.$$

Since $(k_{mm},m_m) = 1, x_m$ is uniquely determined modulo m_m.

Theorem XV states that every residue number has unique coordinates relative to a given basis. Thus every basis serves to define a number system related to the residue number system. It is now shown that triangularity is a necessary as well as a sufficient condition for a nonredundant number system.

Theorem XVI. If the set of residue numbers, $\langle \alpha_1, \alpha_2, \ldots \alpha_n \rangle$, spans R^n (i.e., $r \epsilon R^n \exists \{a_i\} \ni \sum_{i=1}^{n} a_i \alpha_i = r$) then the set is triangular.

Proof: If

$$(a_1, a_2, \ldots, a_n) \longleftrightarrow a_1\alpha_1 + a_2\alpha_2 + \ldots + a_n\alpha_n$$

$$(b_1, b_2, \ldots, b_n) \longleftrightarrow b_1\alpha_1 + b_2\alpha_2 + \ldots + b_n\alpha_n$$

From the properties of R^n it follows that

$$a_i\alpha_i + b_i\alpha_i = (a_i + b_i)\alpha_i$$

then

$$(a_1\alpha_1 + a_2\alpha_2 + \ldots + a_n\alpha_n) + (b_1\alpha_1 + \ldots + b_n\alpha_n) =$$

$$(a_1 + b_1)\alpha_1 + (a_2 + b_2)\alpha_2 + \ldots + (a_n + b_n)\alpha_n =$$

$$d_1\alpha_1 + d_2\alpha_2 + \ldots + d_n\alpha_n \triangleq S$$

Consider the form $d_i = m_i q_i + r_i$ in which $m_i q_i$ is an integer; then

$$m_i q_i = \sum_{j=1}^{n} e_{ij}\alpha_j$$

and

$$S = \sum_{i=1}^{n} r_i\alpha_i + \sum_{i=1}^{n}\sum_{j=1}^{n} q_j e_{ji}\alpha_i$$

$$= \sum_{i=1}^{n} \left(r_i + \sum_{j=1}^{n} q_j e_{ji} \right) \alpha_i$$

The above expression yields the residue number which is congruent to the sum S modulo M; it is not, however, the required expression for the sum. The desired sum has restricted coefficients, $0 \leqslant a_i < m_i$. So consider the above expression to be

$$S^1 = d_1^1\alpha_1 + d_2^1\alpha_2 + \ldots + d_n^1\alpha_n$$

This sum is manipulated to yield S^2, and the process is continued until $S^{k+1} = S^k$. It will be shown that the existence of this sum is equivalent to

triangularity.

The next step of the proof will be to show that no carry cycles can exist. Assume that carry cycles of length k, when $k \leqslant n$, exist. That is

$$e_{k,k-1} \neq 0$$

$$e_{k-1,k-2} \neq 0$$

$$\vdots$$

$$e_{i,k} \neq 0$$

and

$$e_{j,k} = 0 \text{ for } j > k$$

Consider the element $(x_1,x_2,\ldots,x_k,0,0,0)$, where $x_i < m_i - 1$. Since R^n is an additive Abelian group, the additive inverse of the above element exists. Assume it to be

$$(y_1,y_2,\ldots,y_n)$$

Further it is known that the sum

$$(x_1,x_2,\ldots,x_k,\ldots,0,0,0) + (y_1,y_2,\ldots,y_n) = (0,0,\ldots,0)$$

No carries can enter this initially from outside the group of k under consideration.

Case I. $k = 1$: here, $e_{ii} \neq 0$; $e_{ji} = 0$, $j > k$.

A. $x_i + y_i$ must produce a carry into the first position, thus the first component of the sum is non-zero.

B. If the sum, reduced modulo m, plus the carry is $\geqslant m_i$, there must be a carry in the first position and the result is non-zero.

Case II. The general case.

A. It will be necessary to have

a. $y_k = x_k^*$ to get a zero in the kth position. This will produce a

carry to (k-1)th position.

b. If jth position produces a carry, the (j-1)st will also produce a carry, for the (j-1)st unreduced sum is non-zero and will have to be made congruent to zero by the addition of an appropriate y_{j-1}.

It can be seen that there will be <u>at least</u> one full cycle of carries.

B. Assume we have had ℓ full cycles of carries.

a. The sum of carries plus the previously reduced sum is in the kth position. This is non-zero. If this sum is $< m_k$ the desired contradiction has been obtained. That is, the existence of the additive inverse has been contradicted. A carry must propagate from k to k-1, otherwise the inverse does not exist.

b. Again it may be argued that, if the jth position produces a carry, the (j-1)st position must also produce a carry.

Therefore, there are $\ell + 1$ carry cycles and the carry process does not converge; the proper sum is not obtained. This is the desired contradiction which proves that it is not possible for carry cycles to exist.

From the fact that carry cycles of length 1 cannot exist it can be seen that the principal diagonal of the carry matrix consists of zero elements only.

Since there can be no carry cycles of length 2, $e_{ij} \neq 0 \Longrightarrow e_{ji} = 0$.

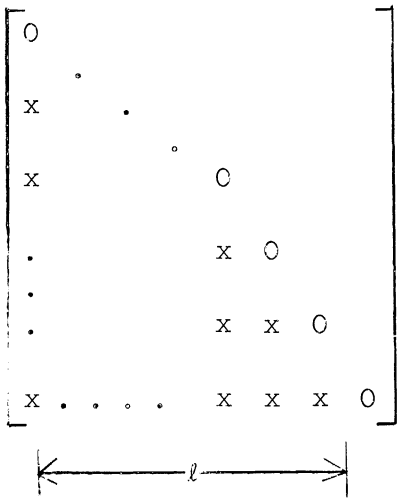
The next step in the proof of the theorem is to show that the number system must have a carry matrix which is a lower triangular array with zero elements on the principal diagonal.

1. <u>Initial Step</u>. There must be at least one column which is composed

entirely of zeros. If this were not the case, there would be carries into every position; and thus carry cycles.

2. General Step. Assume there is one column with at most k-1 non-zero elements for $k = 1,2,\ldots,\ell$. There is a column with at most ℓ-1 non-zero elements.

The array may be rearranged by placing the column, with k non-zero elements in the old array, into the kth column of the new array for $k = \ell, \ell+1, \ldots,n$. The rows are interchanged in a similar manner, giving



Consider a non-zero element $e_{i,\ell-1}$ above the principal diagonal in the (ℓ-1)st column. Since $e_{ij} \neq 0$ implies $e_{ji} = 0$; row i may be interchanged with row ℓ-1, and column i with column ℓ-1, while retaining $e_{\ell-1,\ell-1} = 0$ and $e_{i,\ell-i} = 0$. Thus all non-zero elements are placed below the principal diagonal.

To complete the proof it is only necessary to note that since the base moduli are pairwise relatively prime, multiplying a vector by the ith base

modulus can create a zero only in the ith position. Thus a carry matrix with all non-zero elements below the principal diagonal implies that the set $< \alpha_1, \alpha_2, \ldots, \alpha_n >$ is triangular.

2.2 LINEAR TRANSFORMATIONS AND MATRIX MULTIPLICATION IN THE R SPACE

Definition: A linear transformation $G: R^n \to S^m$ of an R-space R^n into an R-space S^m is a transformation which satisfies

$$(\zeta + \eta)G = \zeta G + \eta G$$

$$(c\zeta)G = c(\zeta G).$$

Theorem XVII. If $\alpha_1, \ldots, \alpha_n$ is any basis of R^n and $\gamma_1, \ldots, \gamma_n$ are any vectors in S^m, then there is one and only one linear transformation $G: R^n \to S^m$ with

$$\begin{aligned} \alpha_1 G &= \gamma_1 \\ &\cdot \\ &\cdot \\ &\cdot \\ \alpha_n G &= \gamma_n \end{aligned}$$

This transformation is defined by

$$(x_1\alpha_1 + \ldots + x_n\alpha_n)G = x_1\gamma_1 + x_2\gamma_2 + \ldots + x_n\gamma_n.$$

Proof: Let $A = \{\alpha_1, \ldots, \alpha_n\}$ be a basis for R^n

and $B = \{\beta_1, \ldots, \beta_m\}$ be a basis for S^m,

then

$$\begin{aligned} \alpha_1 G &= \gamma_1 \\ \alpha_2 G &= \gamma_2 \\ &\cdot \\ &\cdot \\ &\cdot \\ \alpha_n G &= \gamma_n, \end{aligned}$$

where

$$\gamma_i = a_{i1}\beta_1 + \ldots + a_{1m}\beta_m$$

for

$$i = 1,\ldots,n.$$

If

$$\zeta = (x_1,\ldots,x_n)_A \epsilon R^n,$$

then

$$\zeta = x_1\alpha_1 + \ldots + x_n\alpha_n,$$

from which results

$$\zeta G = x_1(\alpha_1 G) + x_2(\alpha_2 G) + \ldots + x_n(\alpha_n G).$$

By substituting for $(\alpha_i G)$, one obtains

$$\begin{aligned} &= x_1(a_{11}\beta_1 + \ldots + a_{1m}\beta_m) \\ &+ x_2(a_{21}\beta_1 + \ldots + a_{2m}\beta_m) \\ &\quad \vdots \\ &+ x_n(a_{n1}\beta_1 + \ldots + a_{nm}\beta_m), \end{aligned}$$

and by rearranging terms,

$$\begin{aligned} \zeta G &= (x_1a_{11} + x_2a_{21} + \ldots + x_na_{n1})\beta_1 \\ &+ (x_1a_{12} + x_2a_{22} + \ldots + x_na_{n2})\beta_2 \\ &\quad \vdots \\ &+ (x_1a_{1m} + x_2a_{2m} + \ldots + x_na_{nm})\beta_m. \end{aligned}$$

By Theorem XI the image ζG can be uniquely expressed

$$\sum_{i=1}^{m} d_i\beta_i$$

where

$$0 \leq d_i < m_i$$

and, therefore, $\zeta G \epsilon S^m$ and the transformation G is a linear transformation $R^n \rightarrow S^m$.

Let

$$\eta = (y_1,\ldots,y_n)_A$$

then

$$\eta = y_1\alpha_1 + \ldots + y_n\alpha_n.$$

Thus

$$\begin{aligned}
\eta G &= y_1(\alpha_1 G) + \ldots + y_n(\alpha_n G) \\
&= y_1\gamma_1 + \ldots + y_n\gamma_n, \\
\zeta + \eta &= (x_1 + y_1)\alpha_1 + \ldots + (x_n + y_n)\alpha_n \\
(\zeta + \eta)G &= (x_1 + y_1)\alpha_1 G + \ldots + (x_n + y_n)\alpha_n G \\
&= (x_1 + y_1)\,\gamma_1 + \ldots + (x_n + y_n)\gamma_n \\
&= x_1\gamma_1 + \ldots + x_n\gamma_n + y_1\gamma_1 + \ldots + y_n\gamma_n \\
&= \zeta G + \eta G
\end{aligned}$$

and

$$\begin{aligned}
c\zeta &= cx_1\alpha_1 + \ldots + cx_n\alpha_n \\
(c\zeta)G &= (cx_1)\alpha_1 G + \ldots + (cx_n)(\alpha_n G) \\
&= cx_1\gamma_1 + \ldots + cx_n\gamma_n \\
&= c(\zeta G).
\end{aligned}$$

Thus the transformation is linear. It is to be noted that it is not necessary that the set $\{\gamma_1,\ldots,\gamma_n\}$ be a triangular set.

Since every vector in R^n can be expressed uniquely as $x_1\alpha_1 + \ldots + x_n\alpha_n$ for $0 \leqslant x_i < m_i$, the transformation is single valued and, therefore, no other transformation from R^n into S^m yields B_iG. The proof is complete.

Let $A = \{\alpha_1,\ldots,\alpha_n\}$ be a basis for R^n and $B = \{\beta_1,\ldots,\beta_m\}$ be a basis for S^m. We then have the equations

$$\alpha_1 G = a_{11}\beta_1 + \ldots + a_{1m}\beta_m$$

$$\alpha_2 G = a_{21}\beta_1 + \cdots + a_{2m}\beta_m$$
$$\vdots$$
$$\alpha_n G = a_{n1}\beta_1 + \cdots + a_{nm}\beta_m$$

The form of these equations suggests writing the square matrix

$$a = \begin{bmatrix} a_{11}a_{12} \cdots a_{1m} \\ a_{21} \cdots \quad a_{2m} \\ \quad \cdots \\ a_{n1} \cdots \quad a_{nm} \end{bmatrix}$$

If R^n, S^n, and T^n are three n-dimensional R-spaces, G is a transformation of R^n into S^n, and H is a linear transformation of S^n into T^n. The product of the transformations is defined

$$\alpha(GH) = (\alpha G)H$$

Theorem XVIII. If the product of two transformations is defined, then it is a linear transformation.

Proof: Let G and H be linear transformations whose product exists, let c be a scalar, and let α_1, α_2 be vectors in the domain of G. Then we have

$$(\alpha_1 + \alpha_2)(GH) = [(\alpha_1 + \alpha_2)G]H$$
$$= [(\alpha_1 G) + (\alpha_2 G)]H$$
$$= (\alpha_1 G)H + (\alpha_2 G)H$$
$$= \alpha_1(GH) + \alpha_2(GH)$$

and

$$(c\alpha_1)(GH) = [(c\alpha_1)G]H$$
$$= [c(\alpha_1 G)]H$$
$$= c[(\alpha_1 G)]H$$
$$= c[\alpha_1(GH)].$$

Multiplication of linear transformations has been defined and will now be used as a guide to defining multiplication of matrices. To this end let R^n, S^n, and T^n be three R-spaces of dimension n with respective bases

$$A = \{\alpha_1,\ldots,\alpha_n\}$$

$$B = \{\beta_1,\ldots,\beta_n\}$$

and

$$C = \{\gamma_1,\ldots,\gamma_n\}.$$

Relative to these bases G has the matrix α and H has the matrix β.

Consider the matrix $P = \|P_{ij}\|$ of the product transformation J = GH relative to the bases for R^n and S^n. A development of the rows of P will be given in terms of a_{ij} and b_{ij}.

$$\alpha_1 G = a_{11}\beta_1 + \ldots + a_{1n}\beta_n$$

$$\alpha_2 G = a_{21}\beta_1 + \ldots + a_{2n}\beta_n$$

$$\vdots$$

$$\alpha_n G = a_{n1}\beta_1 + \ldots + a_{nn}\beta_n$$

and

$$\beta_1 H = b_{11}\gamma_1 + \ldots + b_{1n}\gamma_n$$

$$\beta_2 H = b_{21}\gamma_1 + \ldots + b_{2n}\gamma_n$$

$$\vdots$$

$$\beta_n H = b_{n1}\gamma_1 + \ldots + b_{nn}\gamma_n.$$

$$\text{Thus } (\alpha_1 G)H = a_{11}(\beta_1 H) + a_{12}(\beta_2 H) + \ldots + a_{1n}(\beta_n H)$$

$$= a_{11}(b_{11}\gamma_1 + b_{12}\gamma_2 + \ldots + b_{1n}\gamma_n)$$

$$+ a_{12}(b_{21}\gamma_1 + b_{22}\gamma_2 + \ldots + b_{2n}\gamma_n)$$

$$\vdots$$

$$+ a_{1n}(b_{n1}\gamma_1 + b_{n2}\gamma_2 + \ldots + b_{nn}\gamma_n).$$

Therefore

$$(\alpha_1 GH) = \left(\sum_{k=1}^{n} a_{1k} b_{k1}, \sum_{k=1}^{n} a_{1k} b_{k2}, \ldots, \sum_{k=1}^{n} a_{1k} b_{kn} \right).$$

Similarly

$$(\alpha_2 GH) = \left(\sum_{k=1}^{n} a_{2k} b_{k1}, \sum_{k=1}^{n} a_{2k} b_{k2}, \ldots, \sum_{k=1}^{n} a_{2k} b_{kn} \right).$$

Thus

$$(\alpha_n GH) = \left(\sum_{k=1}^{n} a_{nk} b_{k1}, \sum_{k=1}^{n} a_{nk} b_{k2}, \ldots, \sum_{k=1}^{n} a_{nk} b_{kn} \right).$$

The above equations are preliminary results in the determination of the rows of P. Each of the linear forms indicated above must be expressed as linear forms with restricted coefficients. The linear form with restricted coefficients corresponding to $(\alpha_i GH)$ constitutes the ith row of P.

The justification for restricting the discussion to the multiplication of matrices which correspond to linear transformations from n-dimensional R-spaces into n-dimensional R-spaces is the projected application of such multiplication. The principal application will be in the change of basis operation (conversion from one number system into another). In this application the linear transformation is an automorphism from R^n onto R^n.

In the interest of completeness, the result for the most general case will be given. Let R^x, S^w, and T^z be three R-spaces with respective bases

$$A = \{\alpha_1, \ldots, \alpha_x\},$$

$$B = \{\beta_1, \ldots, \beta_w\},$$

and

$$C = \{\gamma_1, \ldots, \gamma_z\}.$$

The linear transformations G and H are defined by

$$\alpha_i G = \sum_{v=1}^{w} a_{iv} \beta_v (i = 1,2,\ldots,x),$$

and

$$\beta_i H = \sum_{u=1}^{z} b_{iu}\gamma_u (i = 1,2,\ldots,w).$$

Therefore,

$$(\alpha_i GH) = \left(\sum_{v=1}^{w} a_{iv}\beta_v\right)H = \sum_{v=1}^{w} a_{iv}(\beta_v H) = \sum_{v=1}^{w} a_{iv} \sum_{u=1}^{z} b_{vu}\gamma_u$$

$$= \sum_{u=1}^{z} \sum_{v=1}^{w} a_{iv}b_{vu}\gamma_u = \left(\sum_{k=1}^{w} a_{ik}b_{k1}, \sum_{k=1}^{w} a_{ik}b_{k2}, \ldots, \sum_{k=1}^{w} a_{ik}b_{kz}\right).$$

The above linear form when expressed with restricted coefficients constitutes the ith row of the product matrix.

If $A = \|a_{ij}\|$, $B = \|b_{ij}\|$, and $C = \|c_{ij}\|$ are n x n matrices relative to the natural basis, the product (AB)C is developed in the following manner:

$AB = E = \|e_{ij}\|$ relative to the natural basis and the ith row of E is obtained from the linear form

$$\left(\sum_{k=1}^{n} a_{ik}b_{k1}, \sum_{k=1}^{n} a_{ik}b_{k2}, \ldots, \sum_{k=1}^{n} a_{ik}b_{kn}\right).$$

Since this linear form is to be reduced relative to the natural basis, the ith row of E is

$$\left[\left(\sum_{k=1}^{n} a_{ik}b_{k1}\right)(m_1), \left(\sum_{k=1}^{n} a_{ik}b_{k2}\right)(m_2), \ldots, \left(\sum_{k=1}^{n} a_{ik}b_{kn}\right)(m_n)\right].$$

The product $(AB)C = F = \|f_{ij}\|$ is similarly developed and the ith row of F is

$$\left[\left(\sum_{k=1}^{n} e_{ik}c_{ki}\right)(m_1), \ldots, \left(\sum_{k=1}^{n} e_{ik}c_{kn}\right)(m_n)\right]$$

$$= \left\{\sum_{k=1}^{n}\left[\left(\sum_{r=1}^{n} a_{ir}b_{bk}\right)(m_k)\right]c_{k1}\right\}(m_1), \ldots,$$

$$\left\{\sum_{k=1}^{n}\left[\left(\sum_{r=1}^{n} a_{ir}b_{rk}\right)(m_k)\right]c_{kn}\right\}(m_n). \qquad (2\text{-}4)$$

Consideration of the product BC gives rise to the following linear form with

restricted coefficients.

$$\left[\left(\sum_{k=1}^{n} b_{ik}c_{k1}\right)(m_1),\ldots,\left(\sum_{k=1}^{n} b_{ik}c_{kn}\right)(m_n)\right].$$

Also

$$A(BC) = D = \|d_{ij}\|$$

where the ith row of D is

$$\left[\left(\sum_{r=1}^{n} a_{ir}\sum_{k=1}^{n} b_{rk}c_{k1}\right)(m_1),\ldots,\left(\sum_{r=1}^{n} a_{ir}\sum_{k=1}^{n} b_{rk}c_{kn}\right)(m_n)\right]. \qquad (2\text{-}5)$$

As shown by equation (2-4) and (2-5), multiplication of matrices associated with the residue system is not associative, for the ranges of the components in (2-4) and (2-5) differ.

CHAPTER III

CARRY FUNCTIONS IN RESIDUE NUMBER SYSTEM AND RELATED NUMBER SYSTEMS

3.1 THE CARRY ALGORITHM

In the second chapter many of the results depended upon the existence of a linear form with restricted coefficients which was equivalent to a given linear expression. This chapter will discuss an algorithm (the Carry Algorithm) for finding the linear form with restricted coefficients when any linear combination of the basis vectors $\{\alpha_n,\ldots,\alpha_n\}$ is given. The algorithm involves the notion of carries from some components of the representations into other components of the representation.

If the linear form to be reduced is $a_1\alpha_1 + a_2\alpha_2 + \ldots + a_n\alpha_n$, one proceeds by expressing $a_n\alpha_n$ as $b_1\alpha_1 + b_2\alpha_2 + \ldots + b_n\alpha_n$ where $0 \leqslant b_i < m_i$ and making the substitution to obtain $(a_1+b_1)\alpha_1 + \ldots + b_n\alpha_n$. The process is then repeated focusing attention on $(b_{n-1}+a_{n-1})\alpha_{n-1}$ and continued until one has the desired result.

Dividing a_j by m_j, one obtains $a_j = m_jq + r$, $0 \leqslant r < m_j$. Any multiple of $m_j\alpha_j$ will yield a vector in the subspace $(x_1,x_2,\ldots,x_{j-1},0,\ldots,0)$. Since the set of vectors $\{\alpha_i\}$ is a basis, the set $\{\alpha_1,\ldots,\alpha_j\}$ is a basis of the mentioned subspace by Theorem XII. Thus the term m_jq will not affect the multiplier of α_j in the reduced expression; that coefficient must be r. The product $m_j\alpha_j$ can be expressed as a linear combination of α_1 through α_{j-1} and $qm_j\alpha_j$ is merely q times each term of that combination. The linear combination for $(qm_j)\alpha_j$ is added to the original linear combination and $a_j\alpha_j$ is replaced by

$r\alpha_j$. Thus one applies the above procedure to first a_n, then to the new coefficient of α_{n-1}, and so on until the desired result if obtained.

The Carry Algorithm may be stated as follows:

1. Express $m_j\alpha_j$ as a linear combination of $\alpha_1,\ldots,\alpha_{j-1}$ denoted $b_{j1}\alpha_1 + b_{j2}\alpha_2 + \ldots + b_{jj-1}\alpha_{j-1}$ for $j = 2,3,\ldots,n$. Beginning with $j = n$ and $a'_n = a_n$.

2. Divide a'_j by m_j and obtain

$$a'_j = q_j m_j + r_j \text{ with } 0 \leqslant r_j < m_j.$$

3. Replace $a'_j\alpha_j$ with $r_j\alpha_j$ and

$$a'_i \text{ with } (a'_i + q_i b_{ji}) \text{ for } i = 1,2,\ldots,j-1.$$

4. Repeat steps 2 and 3 substituting j-1 for j. Stop after executing steps 2 and 3 with $j = 1$.

Theorem XIX. The linear form produced by the application of the Carry Algorithm expresses the same residue number as the original linear combination.

Proof: The proof will be the demonstration that the coefficients of the resulting linear form satisfy the congruences indicated in the proof of Theorem XI.

Consider $c_n \equiv a_n \bmod m_n$ with $0 \leqslant c_n < m_n$. By the uniqueness of division r_n is the residue of a_n modulo m_n and $r_n = c_n$.

Assume that $r_j = c_j$ for $j = m,m+1,\ldots,n$. The congruence which then must be satisfied is

$$k_{m-1,m-1}c_{m-1} + k_{m,m-1}c_m + \ldots + k_{n,m-1}c_n$$
$$\equiv k_{m-1,m-1}a_{m-1} + k_{m,m-1}a_m + \ldots + k_{n,m-1}\, a \bmod m_{m-1}. \qquad (3\text{-}1)$$

The quantity a'_j which enters the division in step 2 of the Carry Algorithm is

$$a'_j = a_j + q_n b_{nj} + q_{n-1} b_{n-1,j} + \cdots + q_{j+1} b_{j+1,j}$$

thus

$$\begin{aligned}
a'_n &= a_n \\
a'_{n-1} &= a_{n-1} + q_n b_{n,n-1} \\
a'_{n-2} &= a_{n-2} + q_n b_{n,n-2} + q_{n-1} b_{n-1} b_{n-1,n-2} \\
&\vdots \\
a'_{m-1} &= a_{m-1} + q_n b_{n,m-1} + \cdots + q_m b_{m,m-1}.
\end{aligned}$$

In expressing

$$p_j \alpha_j = b_{j1} \alpha_K + b_{j2} \alpha_2 + \cdots + b_{jj-1} \alpha_{j-1},$$

one solved the congruences

$$\begin{aligned}
k_{j-1,j-1} b_{j,j-1} &\equiv m_j k_{j,j-1} \bmod m_{j-1} \\
k_{j-2,j-2} b_{j,j-2} + k_{j-1,j-2} b_{j,j-1} &\equiv m_j k_{j,j-2} \bmod m_{j-2} \\
&\vdots \\
k_{m-1,m-1} b_{j,m-1} + k_{m-2,m-1} b_{j,m-2} + \cdots + k_{j-1,m-1} b_{j,n-1} & \\
\equiv m_j k_{j,m-1} \bmod m_{m-1} & \\
&\vdots \\
k_{11} b_{j1} + k_{21} b_{j2} + \cdots + k_{j-1,1} b_{j,j-1} &\equiv m_j k_{j1} \bmod m_1.
\end{aligned}$$

Using the induction assumption, relation (3-1) can be written

$$\begin{aligned}
&k_{m-1,m-1} c_{m-1} + k_{m,m-1} r_m + k_{n,m-1} r_n \\
&\quad \equiv k_{m-1,m-1} a_{m-1} + k_{m,m-1} a_m + \cdots + k_{n,m-1} a_n \bmod m_{m-1}
\end{aligned}$$

which can be manipulated to yield

$$\begin{aligned}
&k_{m-1,m-1} c_{m-1} + k_{m,m-1} r_m + \cdots + k_{n-1,m-1} r_{n-1} \\
&\quad \equiv k_{m-1,m-1} a_{m-1} + k_{m,m-1} a_m + \cdots + k_{n,m-1} (a_n - r_n) \bmod m_{m-1}.
\end{aligned} \tag{3-2}$$

Since $a_n = a'_n, a_n - r_n = q_n m_n$, (3-2) becomes

$$k_{m-1,m-1}c_{m-1} + k_{m,m-1}r_m + \cdots + k_{n-1,m-1}r_{n-1}$$

$$\equiv k_{m-1,m-1}(a_{m-1} + q_n b_{n,m-1}) + k_{m,m-1}(a_m + q_n b_{nm})$$

$$+ \cdots + k_{n-1,m-1}(a_{n-1} + q_n b_{n,n-1}) \bmod m_{m-1} \qquad (3\text{-}3)$$

upon the substitution

$$q_n(k_{m-1,m-1}b_{n,m-1} + k_{m,m-1}b_{nm} + \cdots + k_{n-1,m-1\ n,}b_{n-1})$$

$$\equiv q_n m_n k_{n,m-1} \pmod{m_{m-1}}.$$

Once again we transpose (add the inverse) $k_{n-1,m-1}r_{n-1}$ and recognize that

$$a_{n-1} + \alpha_n b_{nn-1} - r_{n-1} = a'_{n-1} - r_{n-1} = q_{n-1}m_{n-1}.$$

Making the following substitution:

$$q_{n-1}(k_{m-1,m-1}b_{n-1,m-1} + k_{m,m-1}b_{n-1,m} + \cdots + k_{n-1,m-1}b_{n-1,n-2}$$

$$\equiv q_{n-1}m_{n-1}k_{n-1,m-1} \bmod m_{m-1},$$

we obtain

$$k_{m-1,m-1}c_{m-1} + k_{m,m-1}r_m + \cdots + k_{n-2,m-1}r_{n-2}$$

$$\equiv k_{m-1,m-1}(a_{m-1} + q_n b_{n,m-1} + q_{n-1}b_{n-1,m-1})$$

$$+ k_{m,m-1}(a_m + q_n b_{n,m} + q_{n-1}b_{n-1,m}) + \cdots$$

$$+ k_{n-2,m-1}(a_{n-2} + q_n b_{n,n-2} + q_{n-1}b_{n-1,n-2}) \bmod m_{m-1}.$$

Again we identify the last quantity in parenthesis as a'_{n-2}, transpose and substitute. By repeating these manipulations, one finally obtains

$$k_{m-1,m-1}c_{m-1} \equiv k_{m-1,m-1}(a_{m-1} + q_n b_{n,m-1} + \cdots + q_m b_{m,m-1}) \bmod m_{m-1}$$

or

$$c_{m-1} \equiv a'_{m-1} \bmod m_{m-1}$$

which yields the desired result $c_{m-1} = r_{m-1}$.

By showing that $r_m = c_m$, we have shown that the linear form produced by the Carry Algorithm is identical to the linear form of the conclusion of Theorem

XI. Therefore the proof is completed.

Without the Carry Algorithm, one can perform operations such as addition, multiplication, and matrix multiplication by resorting to the defined operations of addition and scalar multiplication of residue numbers. The only alternative is to solve a set of simultaneous linear congruences every time one wishes to express a linear form with restricted coefficients.

With the Carry Algorithm, it is necessary to solve only n sets of simultaneous linear congruences, and further those sets of congruences may be solved immediately upon the selection of the base moduli and the basis vectors. Thereafter, any linear form may be reduced to one with restricted coefficients by the application of steps 2, 3, and 4 of the Carry Algorithm. It is thus possible to select a set of basis vectors, perform step 1 of the reduction algorithm (i.e., determine the carry functions), and thereafter perform addition of two vectors by addition of the components followed by the application of the Carry Algorithm. Scalar multiplication is effected by multiplying each component of the vector by the scalar and then applying the Carry Algorithm. The Carry Algorithm will prove significant in the multiplication of representations for it will provide a means of combining partial products.

3.2 THE BORROW ALGORITHM AND COMPLEMENTATION

The question arises: Given two representations X and Y of positive integers, how does one obtain the remainder X - Y? This question will now be considered in some detail.

Let X be represented by $(x_1,x_2,\ldots,x_n)\alpha$ and Y by $(y_1,y_2,\ldots,y_n)\alpha$. Initially one forms $(x_1-y_1,x_2-y_2,\ldots,x_n-y_n)$. This last expression denotes

$(x_1-y_1)\alpha_1 + (x_2-y_2)\alpha_2 + \ldots + (x_n-y_n)\alpha_n$. Consider the jth component of an expression to be negative. Since

$$m_j\alpha_j = b_{j1}\alpha_1 + b_{j2}\alpha_2 + \ldots + b_{j,j-1}\alpha_{j-1}$$

one may add to the above expression zero in the form

$$d_j[m_j\alpha_j - (b_{j1}\alpha_1 + b_{j2}\alpha_2 + b_{j,j-1}\alpha_{j-1})] = 0$$

where d_j is the smallest multiple of m_j which is larger than the magnitude of the jth component. This addition yields

$$(x_1 - y_1 - d_n b_{n1}, x_2 - y_2 - d_n b_{n2}, \ldots, x_{n-1} - y_{n-2} - d_n b_{n,n-1},$$
$$x_n - y_n + d_n m_n)$$

for j = n,

$$(x_1 - y_1 - d_n b_{n1} - d_{n-1} b_{n-1,1}, x_2 - y_2 - d_n b_{n2} - d_{n-1} b_{n-1,2},$$
$$\ldots, x_{n-1} - y_{n-1} - d_n b_{n,n-1} + d_{n-1} m_{n-1}, x_n - y_n + d_n m_n)$$

for j = n-1, and finally

$$(x_1 - y_1 - d_n b_{n1} - d_{n-1} b_{n-1,1} - \ldots - d_2 b_{21} + d_1 m_1,$$
$$x_2 - y_2 - d_n b_{n2} - d_{n-1} b_{n-1,2} - \ldots - d_3 b_{32} + d_2 m_2,$$
$$\ldots, x_{n-1} - y_{n-1} - d_n b_{n,n-1} + d_{n-1} m_{n-1}, x_n - y_n + d_n m_n) \text{ for } j = 1.$$

This final expression will be a linear form with restricted coefficients which is the representation of X - Y if $X \geq Y$. If $X < Y$ the above procedure yields a representation of -(Y - X). Thus complementation quite naturally enters the picture.

By dividing the range of integers which can be represented by residue numbers into those integers less than M/2 and those integers larger than M/2, one can designate the first group of vectors to be representations of positive integers and the second group, codings for the complements of the elements of the

first group. If M is even, M/2 is self complement.

To find the complement of a given representation Z, one generates the remainder (0-Z) as indicated above.

The procedure for performing subtraction and finding complements indicated above suggests the statement of a Borrow Algorithm as follows:

1. Express $m_j\alpha_j$ as a linear combination of $\alpha_1,\ldots,\alpha_{j-1}$ denoted $b_{j1}\alpha_1 + b_{j2}\alpha_2 + \ldots + b_{j,j-1}\alpha_{j-1}$ for $j = 2,3,\ldots,n$. Beginning with $j = n$ and $a'_j = a_n$.
2. Perform steps 3 and 4 if $a'_j < 0$, otherwise skip to step 5.
3. Determine d_j such that d_j is the smallest multiple of m_j which is larger than $|a'_j|$.
4. Add to the linear form the expression $(-d_jb_{j1}, -d_jb_{j2},\ldots, -d_jb_{j,j-1}, +d_jm_j,0,\ldots,0)$.
5. Repeat steps 2, 3, and 4 substituting j-1 for j. Stop after executing steps 2, 3, and 4 for $j = 1$.

Example: With the basis < (1,0,0,0), (1,2,0,0), (1,1,2,0), (1,1,1,1) > where the moduli are $m_1 = 2$, $m_2 = 3$, $m_3 = 5$ and $m_4 = 7$, the carry functions are $3\alpha_2 = \alpha_1$, $5\alpha_3 = \alpha_2$, and $7\alpha_4 = \alpha_3$. An example of subtraction using the Borrow Algorithm directly will be given as well as subtraction by using the complement of the subtrahend.

Subtraction using the borrow algorithm:

$$
\begin{array}{rl}
 & (\ 1,\ 2,\ 2,\ 1) \longleftrightarrow\ 190 \\
- & \underline{(\ 0,\ 2,\ 4,\ 6) \longleftrightarrow -104} \\
 & (\ 1,\ 0,-2,-5) \\
+ & \underline{(\ 0,\ 0,-1,\ 7)} \\
 & (\ 1,\ 0,-3,\ 2) \\
+ & \underline{(\ 0,-1,\ 5,\ 0)} \\
 & (\ 1,-1,\ 2,\ 2) \\
+ & \underline{(-1,\ 3,\ 0,\ 0)} \\
 & (\ 0,\ 2,\ 2,\ 2) \longleftrightarrow\ 86
\end{array}
$$

Complement of (0,2,4,6):

$$\begin{array}{r} (\ 0,-2,-4,-6) \\ +\ (\ 0,\ 0,-1,\ 7) \\ \hline (\ 0,-2,-5,\ 1) \\ +\ (\ 0,-1,\ 5,\ 0) \\ \hline (\ 0,-3,\ 0,\ 1) \\ +\ (-1,-3,\ 0,\ 0) \\ \hline (-1,\ 0,\ 0,\ 1) \\ +\ (\ 2,\ 0,\ 0,\ 0) \\ \hline (\ 1,\ 0,\ 0,\ 1) \end{array}$$

Subtraction utilizing the complement:

$$\begin{array}{rl} (\ 1,\ 2,\ 2,\ 1) & \longleftrightarrow\ 190 \\ +\ (\ 1,\ 0,\ 0,\ 1) & \longleftrightarrow -104 \\ \hline (\ 0,\ 2,\ 2,\ 2) & \longleftrightarrow\ \ 86 \end{array}$$

3.3 CHANGE OF BASIS

One quite important use of the carry algorithm and matrix multiplication is the change of basis operation. Quite often it is desirable to convert from one number system to another, i.e., express a vector in coordinates relative to a different set of basis vectors. One might wish to make the conversion because different number systems are more advantageous for particular operations than others. More will be said concerning this later.

Let a vector X be represented by $(x_1,x_2,\ldots,x_n)$ relative to the basis $< \alpha_1,\alpha_2,\ldots,\alpha_n >$. It is desired to find coordinates $(y_1,y_2,\ldots,y_n)$, relative to the basis $< \beta_1,\beta_2,\ldots,\beta_n >$. Each vector α_i of the old basis can be expressed as a linear combination of the vectors of the new basis in the form

$$\alpha_i = q_{i1}\beta_1 + q_{i2}\beta_2 + \ldots + q_{in}\beta_n. \qquad (3\text{-}4)$$

However, since both bases are triangular, $q_{ik} = 0$ for $k > i$. The vector X with

coordinates $(x_1,x_2,\ldots,x_n)$ relative to the basis

$$< \alpha_1,\alpha_2,\ldots,\alpha_n > \text{ is } x_1\alpha_1 + x_2\alpha_2 + \ldots + x_n\alpha_n.$$

Substituting from above, one obtains for X

$$x_1q_{11}\beta_1 + x_2[q_{21}\beta_1 + q_{22}\beta_2] + \ldots + x_n[q_{n1}\beta_1 + \ldots + q_{nn}\beta_n]$$

which can be written

$$(x_1q_{11} + x_2q_{21} + \ldots + x_nq_{n1})\beta_1$$

$$+ (x_2q_{22} + \ldots + x_nq_{n2})\beta_2 + \ldots + x_nq_{nn}\beta_n. \qquad (3\text{-}5)$$

The carry algorithm is then applied to linear form (3-5) and the result is

$X = y_1\beta_1 + y_2\beta_2 + \ldots + y_n\beta_n.$

The y_i are the coordinates of X relative to the basis $< \beta_1,\beta_2,\ldots,\beta_n >$.

If the zero coefficients are retained in expression (3-4), expression (3-5) becomes

$$(x_1q_{11} + x_2q_{21} + \ldots + x_nq_{n1})\beta_1 + (x_1q_{12} + x_2q_{22} + \ldots + x_nq_{n2})\beta_2$$

$$+ \ldots + (x_1q_{1n} + x_2q_{2n} + \ldots + x_nq_{nn})\beta_n.$$

This expression is an interpretation of

$$(x_1q_{11} + x_2q_{21} + \ldots + x_nq_{n1}),(x_1q_{12} + x_2q_{22} + \ldots + x_nq_{n2}),$$

$$\ldots + (x_1q_{1n} + \ldots + x_nq_{nn})$$

which is the matrix product

$$[x_1,x_2,\ldots,x_n] \begin{bmatrix} q_{11}q_{12} \cdots q_{1n} \\ q_{21}q_{22} \cdots q_{2n} \\ \cdot \\ \cdot \\ \cdot \\ q_{n1}q_{n2} \cdots q_{nn} \end{bmatrix} = [y_1,y_2,\ldots,y_n]$$

denoted $X \cdot Q = Y$.

The above procedure constitutes an effective procedure for executing

change of basis.

The vectors β_i of the new basis can be written as linear combinations of the old vectors,

$$\beta_i = \sum_{j=1}^{n} p_{ij}\alpha_j \tag{3-6}$$

Thus for a change of basis from $< \beta_1,\ldots,\beta_n >$ to $< \alpha_1,\ldots,\alpha_n >$, the appropriate matrix product is $Y \cdot P = X$ where $P = \|P_{ij}\|$.

Substituting equation (3-6) into equation (3-4) one obtains

$$\begin{aligned}\alpha_i &= q_{i1}\sum_{j=1}^{n} p_{ij}\alpha_j + q_{i2}\sum_{j=1}^{n} p_{2j}\alpha_j + \ldots + q_{in}\sum_{j=1}^{n} p_{nj}\alpha_j \\ &= \sum_{j=1}^{n} q_{i1}p_{1j}\alpha_j + \sum_{j=1}^{n} q_{i2}p_{2j}\alpha_j + \ldots + \sum_{j=1}^{n} q_{in}p_{nj}\alpha_j \\ &= \sum_{k=1}^{n}\sum_{j=1}^{n} q_{ik}p_{kj}q_j\end{aligned}$$

One may interchange summations to obtain

$$\begin{aligned}\alpha_i &= \sum_{j=1}^{n}\sum_{k=1}^{n} q_{ik}p_{kj}q_j \\ &= \sum_{k=1}^{n} q_{ik}p_{kj}\alpha_1 + \sum_{k=1}^{n} q_{ik}p_{k2}\alpha_2 + \ldots + \sum_{k=1}^{n} q_{ik}p_{kn}\alpha_n .\end{aligned} \tag{3-7}$$

Equation (3-7) written in n-tuple form with restricted coefficients is the ith row of the matrix product QP. Since the α_i constitute a basis, the reduced form of equation (3-7) must be $\alpha_i = \alpha_i$. As a consequence, it is seen that

$$Q \cdot P = I. \tag{3-8}$$

By advancing a dual argument, one deduces

$$P \cdot Q = I. \tag{3-9}$$

Equations (3-8) and (3-9) can be used as a check on the determination of the P and Q matrices. These equations are necessary but not sufficient conditions.

3.4 MULTIPLICATION

To multiply two elements of a number system related to the residue number system, one forms partial products, one for each component of the multiplier, and adds them together producing a linear combination of the basis vectors which is then reduced by means of the Carry Algorithm. The algorithm for determining the form of the partial products will be called the Multiplication Algorithm.

The Multiplication Algorithm may be stated as follows:

Consider the most general multiplicand $(y_1,y_2,\ldots,y_n)$ and multiplier $(x_1,x_2,\ldots,x_n)$.

1. Write the multiplicand $(y_1,y_2,\ldots,y_n)$ as the vector sum $y_1\alpha_1 + y_2\alpha_2 + \ldots + y_n\alpha_n$.

Beginning with $j = n$

2. Write the partial multiplier $(0,\ldots,0,x_j,0,\ldots,0)$ as the vector $x_j\alpha_j$.
3. Multiply, component by component, $x_j\alpha_j \cdot y_i\alpha_i$, $i = 0,\ldots,n$.
4. Express the vector $x_j\alpha_j \cdot y_i\alpha_i$ as the linear combination $Z_i\alpha_1 + \ldots + Z_i\alpha_1$, $i = 0,\ldots,n$.
5. Sum the linear forms produced in Step 4 to determine the jth partial sum.
6. Reduce j by one and repeat Steps 2 through 5. Terminate the procedure after doing the above steps with $j = 1$.

Example: Consider the multiplication

$$(y_1,y_2,\ldots,y_4)_M \cdot (x_1,x_2,\ldots,x_4)_M$$

in the mixed base number system where $m_1 = 2$, $m_2 = 3$, $m_3 = 5$, $m_4 = 7$. Here

the basis is $< (1,0,0,0), (1,2,0,0), (1,1,2,0), (1,1,1,1) >$.

The following steps are numbered to correspond to the statement of the Multiplication Algorithm.

1. $(y_1,0,0,0) + (y_2,2y_2,0,0) + (y_3,y_3,2y_3,0) + (y_4,y_4,y_4,y_4)$

2. (x_4,x_4,x_4,x_4)

3. $(y_1x_4,0,0,0)$

   $(y_2x_4,2y_2,x_4,0,0)$

   $(y_3x_4,y_3x_4,2y_3x_4,0)$

   $(y_4x_4,y_4x_4,y_4x_4,y_4x_4)$

4. $(y_1x_4,0,0,0) = (y_1x_4,0,0,0)_M$

   $(y_2x_4,2y_2x_4,0,0) = (0,y_2x_4,0,0)_M$

   $(y_3x_4,y_3x_4,2y_3x_4,0) = (0,0,y_3x_4,0)_M$

   $(y_4x_4,y_4x_4,y_4x_4,y_4x_4) = (0,0,0,y_4x_4)_M$

5. $(y_1,y_2,y_3,y_4)_M \cdot (0,0,0,x_4)_M = (x_4,y_1,x_4y_2,x_4y_3,x_4y_4)_M$

2. $(x_3,x_3,2x_3,0)$

3. $(y_1x_3,0,0,0) = A$

   $(y_2x_3,2y_2x_3,0,0) = B$

   $(y_3x_3,y_3x_3,2(2y_3x_3),0) = C$

   $(y_4x_3,y_4x_3,2y_4x_3,0) = D$

4. $A = (y_1x_3,0,0,0)_M$

   $B = (0,y_2x_3,0,0)_M$

   $C = (0,0,2y_3x_3,0)_M + (0,x_3y_3,0,0)_M$

   since $(0,0,2x_3y_3,0)_M + (0,x_3y_3,0,0)_M$

   $= (2x_3y_3,2x_3y_3,4x_3y_3,0) + (x_3y_3,2x_3y_3,0,0)$

$= (y_3x_3, y_3x_3, 2(2y_3x_3), 0)$

$D = (0,0,x_3y_4,0)_M$.

5. $(y_1,y_2,y_3,y_4)_M \cdot (0,0,x_3,0)_M$

$= (y_1x_3, y_2x_3 + y_3x_3, 2x_3y_3 + x_3y_4, 0)_M$

2. $(x_2, 2x_2, 0, 0)$

3. $(x_2y_1, 0, 0, 0) = A'$

$(x_2y_2, 4x_2y_2, 0, 0) = B'$

$(x_2y_3, 2x_2y_3, 0, 0) = C'$

$(x_2x_4, 2x_2y_4, 0, 0) = D'$

4. $A' = (y_1x_2, 0, 0, 0)_M$

$B' = 2(0, x_2y_2, 0, 0)_M + (x_2y_2, 0, 0, 0)_M$

for $(x_2y_2, 4x_2y_2, 0, 0)_M$

$= (x_2y_2, 2x_2y_2, 0, 0) + (0, 2x_2y_2, 0, 0)$

$= (x_2y_2, 2x_2y_2, 0, 0) + (x_2y_2, 2x_2y_2, 0, 0)$

$+ (x_2y_2, 0, 0, 0)$

$C' = (0, x_2y_3, 0, 0)_M$

$D' = (0, x_2y_4, 0, 0)_M$

5. $(y_1,y_2,y_3,y_4)_M \cdot (0,x_2,0,0)_M$

$= (x_2y_1 + x_2y_2, 2x_2y_2 + x_2y_2 + x_2y_4, 0, 0)_M$

2. $(x_1, 0, 0, 0)$

3. $(x_1y_1, 0, 0, 0)$

$(x_1y_2, 0, 0, 0)$

$(x_1y_3, 0, 0, 0)$

$(x_1y_4, 0, 0, 0)$

5. $(y_1,y_2,y_3,y_4)_M \cdot (x_1,0,0,0)_M = (x_1y_1 + x_1y_2 + x_1y_3 + x_1y_4,0,0,0)_M$

The results of the Multiplication Algorithm in this example are the following for rules for the formation of partial products:

$(y_1,y_2,y_3,y_4)_M \cdot (x_1,0,0,0)_M = (x_1y_1 + x_1y_2 + x_1y_3 + x_1y_4,0,0,0)_M$

$(y_1,y_2,y_3,y_4)_M \cdot (0,x_2,0,0)_M = (x_2y_1 + x_2y_2,2x_2y_2 + x_2y_3 + x_2y_4,0,0)_M$

$(y_1,y_2,y_3,y_4)_M \cdot (0,0,x_3,0)_M = (x_3y_1,x_3y_2 + x_3y_3,2x_3y_3 + x_3y_4,0)_M$

$(y_1,y_2,y_3,y_4)_M \cdot (0,0,0,x_4)_M = (x_4y_1,x_4y_2,x_4y_3,x_4y_4)_M.$

To multiply two mixed base elements $(y_1,y_2,y_3,y_4)_M$ and $(x_1,x_2,x_3,x_4)_M$ one proceeds as follows:

1. Form the above partial products.

2. Reduce each partial product by employing the Carry Algorithm.

3. Sum the partial products again employing the Carry Algorithm.

As an example, consider the product $(1,2,3,4)_M \cdot (0,2,4,6)_M$. The partial products are

$(1,2,3,4)_M \cdot (0,0,0,6)_M = (6,12,18,24) = (1,1,1,3)_M \bmod M$

$(1,2,3,4)_M \cdot (0,0,4,0)_M = (4,28,40,0) = (1,1,0,0)_M \bmod M$

$(1,2,3,4)_M \cdot (0,2,0,0)_M = (6,22,0,0) = (1,1,0,0)_M \bmod M.$

Therefore,

$(1,2,3,4)_M \cdot (0,2,4,6)_M = (0,0,1,3)_M \bmod M$

$(0,2,4,6)_M \longleftrightarrow 104$

and

$(1,2,3,4)_M \longleftrightarrow 200$

$104 \cdot 200 = 20800 \equiv 10 \bmod M$

$(0,0,1,3)_M \longleftrightarrow 10.$

Let us look again at the question of associativity of matrix multiplication. Consider the three matrices $A = \|a_{ij}\|$, $B = \|b_{ij}\|$, and $C = \|c_{ij}\|$. Let the basis of the vector spaces by U, $< \alpha_1,\alpha_2,\ldots,\alpha_n >$; V, $< \beta_1,\beta_2,\ldots,\beta_n >$; W, $< \gamma_1,\gamma_2,\ldots,\gamma_n >$; and Z, $< \delta_1,\delta_2,\ldots,\delta_n >$. T_A is a transformation from $U \to V$, T_B: $V \to W$, and T_C: $W \to Z$.

The ith row of the product A·B is

$$\left(\sum_{k=1}^{n} a_{ik}b_{k1}, \sum_{k=1}^{n} a_{ik}b_{k2}, \ldots, \sum_{k=1}^{n} a_{ik}b_{kn}\right)$$

reduced relative to the basis $< \gamma_1,\ldots,\gamma_n >$. Designating the reduced form obtained $(e_{i1},e_{i2},\ldots,e_{in})$, we obtain

$$e_{in} = \left\{\frac{\sum_{k=1}^{n} a_{ik}b_{kn}}{m_n}\right\}$$

Define

$$f_{in} = \left[\frac{\sum_{k=1}^{n} a_{ik}b_{kn}}{m_n}\right]$$

$$e_{i,n-1} = \left\{\frac{\sum_{k=1}^{n} a_{ik}b_{k,n-1} + f_{in}g_{n,n-1}}{m_{n-1}}\right\}$$

$$f_{i,n-1} = \left[\frac{\sum_{k=1}^{n} a_{ik}b_{k,n-1} + f_{in}g_{n,n-1}}{m_{n-1}}\right]$$

$$\vdots$$

$$e_{i1} = \left\{\frac{\sum_{k=1}^{n} a_{ik}b_{k1} + f_{in}g_{n,1} + f_{i,n-1}g_{n-1} + \cdots + f_{i,2}g_{21}}{m_1}\right\}$$

where

$$m_j \cdot \gamma_j = g_{j1}\gamma_1 + g_{j2}\gamma_2 + \cdots + g_{j,j-1}\gamma_{j-1}.$$

The ith row of the product (A·B)C is

$$\left(\sum_{k=1}^{n} e_{ik}c_{k1}, \sum_{k=1}^{n} e_{ik}c_{k2}, \ldots, \sum_{k=1}^{n} e_{ik}c_{kn}\right)$$

reduced relative to the basis $< \gamma_1, \gamma_2, \ldots, \gamma_n >$. This gives rise to the reduced from

$$(h_{i1}, h_{i2}, \ldots, h_{in})$$

where

$$h_{in} = \left\{\frac{\sum_{k=1}^{n} e_{ik}c_{kn}}{m_n}\right\}$$

$$\ell_{in} = \left[\frac{\sum_{k=1}^{n} e_{ik}c_{kn}}{m_n}\right]$$

$$h_{i,n-1} = \left\{\frac{\sum_{k=1}^{n} e_{ik}c_{k,n-1} + \ell_{in}r_{n,n-1}}{m_{n-1}}\right\}$$

and

$$\ell_{i,n-1} = \left[\frac{\sum_{k=1}^{n} e_{ik}c_{k,n-1} + \ell_{in}r_{n,n-1}}{m_{n-1}}\right]$$

$$h_{i1} = \left\{\frac{\sum_{k=1}^{n} e_{ik}c_{k1} + \ell_{in}r_{n,1} + \ell_{i,n-1}r_{n-1,1} + \cdots + f_{i2}r_{21}}{m_1}\right\}$$

where

$$m_j\delta_j = r_{j1}\delta_1 + \cdots + r_{j,j-1}\delta_{j-1}.$$

Similarly the ith row of the product B·C is

$$\left(\sum_{k=1}^{n} b_{ik}c_{k1}, \sum_{k=1}^{n} b_{ik}c_{k2}, \ldots, \sum_{k=1}^{n} b_{ik}c_{kn}\right)$$

reduced relative to the basis $< \delta_1, \ldots, \delta_n >$. The reduced form is

$(s_{i1}, s_{i2}, \ldots, s_{in})$, where

$$s_{in} = \left\{ \frac{\sum_{k=1}^{n} b_{ik} c_{kn}}{m_n} \right\}$$

$$t_{in} = \left[\frac{\sum_{k=1}^{n} b_{ik} c_{kn}}{m_n} \right]$$

$$s_{i,n-1} = \left\{ \frac{\sum_{k=1}^{n} b_{ik} c_{k,n-1} + t_{in} r_{n,n-1}}{m_{n-1}} \right\}$$

$$t_{i,n-1} = \left[\frac{\sum_{k=1}^{n} b_{ik} c_{k,n-1} + t_{in} r_{n,n-1}}{m_{n-1}} \right]$$

$$\vdots$$

$$s_{i1} = \left\{ \frac{\sum_{k=1}^{n} b_{ik} c_{k1} + t_{in} r_{n1} + t_{i,n-1} r_{n-1,1} + \cdots + t_{i2} r_{21}}{m_1} \right\}$$

The ith row of A(BC) is

$$\left(\sum_{k=1}^{n} a_{ik} s_{k1}, \sum_{k=1}^{n} a_{ik} s_{k2}, \ldots, \sum_{k=1}^{n} a_{ik} s_{kn} \right)$$

reduced relative to the basis $< \delta_1, \ldots, \delta_n >$. The reduced form is

$$(v_{i1}, v_{i2}, \ldots, v_{in})$$

where

$$v_{in} = \left\{ \frac{\sum_{k=1}^{n} a_{ik} s_{kn}}{m_n} \right\}$$

$$v_{i,n-1} = \left\{ \frac{\sum_{k=1}^{n} a_{ik} s_{k,n-1} + u_{in} r_{n,n-1}}{m_{n-1}} \right\}$$

$$u_{i,n-1} = \left[\frac{\sum_{k=1}^{n} a_{ik} s_{k,n-1} + u_{in} r_{n,n-1}}{m_{n-1}} \right]$$

$$\vdots$$

$$v_{i1} = \left(\frac{\sum_{k=1}^{n} a_{ik} s_{k1} + u_{in} r_{n,1} + u_{i,n-1} r_{n-1,1} + \cdots + u_{i2} r_{21}}{m_1} \right)$$

Selecting particular components for comparison

$$h_{in} = \left(\sum_{k=1}^{n} e_{ik} c_{kn} \right) \bmod m_n$$

$$= \left[\left(\sum_{k=1}^{n} a_{ik} b_{kn} \right) \bmod m_n \cdot c_n \right.$$

$$+ \left(\sum_{k=1}^{n} a_{ik} b_{k,n-1} + f_{in} g_{n,n-1} \right) \bmod m_{n-1} \cdot c_{n-1,n} + \cdots$$

$$+ \left(\sum_{k=1}^{n} a_{ik} b_{k1} + f_{in} g_{n1} + f_{i,n-1} g_{n-1,1} + \cdots \right.$$

$$\left. \left. + f_{i2} g_{21} \right) \bmod m_1 \cdot c_{1,n} \right] \bmod m_n$$

or

$$h_{in} = \sum_{r=1}^{n} \left(\sum_{r=1}^{n} a_{ik} b_{kr} \bmod m_r \right) c_{rn} \bmod m_n$$

$$+ \sum_{\ell=1}^{n-1} \left(\sum_{j=\ell}^{n} f_{ij} g_j \bmod m_\ell \right) \cdot c_{\ell n} \bmod m_n .$$

Removing one term and changing indices, one finds

$$h_{in} = \sum_{k=1}^{n} a_{ik} b_{kn} c_{nn} \bmod m_n$$

$$+ \left[\sum_{r=1}^{n-1} \left(\sum_{k=1}^{n} a_{ik} b_{kr} \bmod m_r \right) \cdot c_{rn} \right] \bmod m_n$$

$$+ \sum_{r=1}^{n-1} \left(\sum_{j=r}^{n} f_{ij} g_{jr} \bmod m_r \right) c_{rn} \bmod m_n$$

$$= \sum_{k=1}^{n} a_{ik} b_{kn} c_{nn} \bmod m_n$$

$$+ \sum_{r=1}^{n-1} \left[\sum_{k=1}^{n} a_{ik} b_{kr} + \sum_{j=r}^{n} f_{ij} g_{jr} \right] \bmod m_r \cdot c_{rn} \bmod m_n .$$

Also

$$v_{in} = \left(\sum_{k=1}^{n} a_{ik} s_{kn} \right) \bmod m_n$$

$$= \left[s_{i1} \left(\sum_{k=1}^{n} b_{ik} c_{kn} \bmod m_n \right) + a_{i2} \left(\sum_{k=1}^{n} b_{2k} c_{kn} \bmod m_n \right) \right.$$

$$\left. + \ \ldots \ 2\, a_{in} \left(\sum_{k=1}^{n} b_{nk} c_{kn} \bmod m_n \right) \right] \bmod m_n$$

or

$$v_{in} = \sum_{r=1}^{n} a_{ir} b_{rn} c_{nn} \bmod m_n + \sum_{k=1}^{n-1} \sum_{r=1}^{n} a_{ir} b_{rk} c_{kn} \bmod m_n .$$

Changing indices for clarity, one obtains

$$v_{in} = \sum_{k=1}^{n} a_{ik} b_{kn} c_{nn} \bmod m_n + \sum_{r=1}^{n-1} \sum_{k=1}^{n} a_{ik} b_{kr} c_{rn} \bmod m_n .$$

Since the first terms of h_{in} and v_{in} are the same, it is sufficient to look at

$$\left[\sum_{r=1}^{n} a_{ir} b_{rk} + \sum_{j=k}^{n} f_{ij} g_{ik} \right] \bmod m_k \qquad (3\text{-}10)$$

and

$$\left[\sum_{r=1}^{n} a_{ir} b_{rk} \right] \bmod m_n . \qquad (3\text{-}11)$$

It is seen that the above expressions are not in general equal, for the range of expression (3-10) is the positive integers less than m_k whereas the range of (3-11) is the positive integers less than m_n. Since the m_j are relatively prime, one is led to conclude that the ith row of (AB)C is not equal in general to the corresponding row of A(BC). This was shown independent of the

selection of the basis for the various image spaces involved. Therefore, no selection of basis will guarantee associativity of matrix multiplication.

CHAPTER IV

THE MIXED BASE NUMBER SYSTEM

4.1 GENERAL CHARACTERISTICS

In order to remove ambiguity the elements of a finite number system are partitioned into two classes, those representations defined as positive and those defined as negative. Normal convention assigns a magnitude interpretation to positive elements and a complement interpretation to the negative elements. For sign determination and magnitude comparison, it is necessary to identify the classification of each and every element. The structure of the residue number system makes immediate classification difficult. The structure of the mixed base system facilitates immediate classification. In addition, the mixed base system allows one to handle the problems of additive and multiplicative overflow, and division. We shall see that the mixed base system extracts payment in the form of carries for these advantages.

The basis vectors for the mixed base system are generated in the following manner:

1. Order the primes $m_1, m_2, \ldots, m_n$.
2. Set α_n equal to the vector consisting of all 1's. Beginning with $j = n$.
3. Obtain $\alpha_{j-1} = m_j \alpha_j$.
4. Repeat step 3 with j replaced by j-1. Stop after performing 3 with $j = 1$.

Theorem XX. The vectors $\{\alpha_1,\ldots,\alpha_n\}$ produced by the above scheme constitute a basis.

In order to prove the theorem, Lemma 1 will be proven.

Lemma 1: If $(a,b) = 1$, then $(a \bmod b, b) = 1$.

Proof: $a = bg + r$, $0 \leqslant r < b$ and by definition r is the residue of a mod b

$$a \equiv r \bmod b$$

or

$$a \equiv (a \bmod b) \bmod b.$$

Since $x \equiv y \bmod z \Rightarrow (x,z) = (y,z)$, $(a \bmod b, b) + (a,b)$. The conclusion follows.

Proof of Theorem XX: α_n is a representation of the integer 1. The vector $\alpha\ell$ is the residue representation of $A = \prod_{i=\ell+1}^{n} m_i$. Each m_i for $i \leqslant \ell$ is relatively prime to $A\ell$. Thus it is seen that

$$k_{\ell i} = 0 \text{ for } i > \ell$$

and $(k_{\ell i}, m_i) = 1$ for $i \geqslant \ell$. By Theorem XII the set $< \alpha_1,\alpha_2,\ldots,\alpha_n >$ is a basis of R^n.

Theorem XXI. An element $(x_1,x_2,\ldots,x_n)$ of the mixed base number system is a representation of an integer X in the range $C\frac{M}{m_1} \leqslant X < (C+1)\frac{M}{m_1}$ if and only if $x_1 = C$.

Proof: The element $X = (x_1,x_2,\ldots,x_n)$ is a coding of the integer

$$x_1 \frac{M}{m_1} + x_2 \frac{M}{m_1m_2} + x_3 \frac{M}{m_1m_2m_3} + \ldots + x_n \text{ modulo } M. \qquad (4\text{-}1)$$

Consider the quantity

$$x_1 \frac{M}{m_1} + x_2 \frac{M}{m_1m_2} + x_3 \frac{M}{m_1m_2m_3} + \ldots + x_n. \qquad (4\text{-}2)$$

The largest value expression (4-2) can assume is attained when

$$x_i = m_i - 1.$$

Upon substitution one obtains

$$(m_1 - 1)\frac{M}{m_1} + (m_2 - 1)\frac{M}{m_1 m_2} + (m_3 - 1)\frac{M}{m_1 m_2 m_3} + \ldots + (m_n - 1)$$

This is then rewritten as

$$(m_n - 1) + m_n(m_{n-1} - 1) + m_n m_{n-1}(m_{n-2} - 1) + \ldots + \frac{M}{m_1 m_2}(m_2 - 1) + \frac{M}{m_1}$$

remembering the scheme for generating the base vectors. All but the following terms add out:

$$-1 + \frac{M}{m_1} + \frac{M}{m_1} + M - 1.$$

This means that expression (4-2) is equivalent to expression (4-1). Consider next the quantity:

$$(m_2 - 1)\frac{M}{m_1 m_2} + (m_3 - 1)\frac{M}{m_1 m_2 m_3} + \ldots + (m_n - 1)$$

which is equal to

$$(m_n - 1) + m_n(m_{n-1} - 1) + m_n m_{n-1}(m_{n-2} - 1) + \ldots + (m_2 - 1)\frac{M}{m_1 m_2}.$$

The above expression reduces to

$$-1 + \frac{M}{m_1}.$$

From this the conclusion is clear.

To determine the sign of a mixed base number it is necessary to have a partitioning of elements representing consecutive integers into two classes. The first coordinate gives such a partitioning if m_1 is even, i.e.,

$x_1 < m_1/2 \equiv 0 < X < M/2$.

When applying the Carry Algorithm to the reduction of the expression

$$a_1\alpha_1 + a_2\alpha_2 + \ldots + a_n\alpha_n$$

(α_i is a base vector of the mixed base system) one must increase by one a'_j for every multiple of m_{j+1} contained in a'_{j+1}. (The notation here is consistent with that contained in the discussion of the Carry Algorithm.) This indicates that a carry may be propagated from the nth coordinate to the first. Therefore, when adding two elements of the mixed base system, one unit of time is required to produce the unreduced sum, and up to n-1 units of time may be required to propagate carries. Borrowing is accomplished by reducing a'_j by 1 and increasing a'_{j+1} by m_{j+1}. Again up to n-1 units of time may be required to perform subtraction or complementation.

Multiplication of two elements of the mixed base system was discussed and an example given in Chapter III.

Theorem XXII. When two mixed base numbers are added, the carries are binary and a position which produces a carry cannot also propagate a carry.

Proof: The largest jth component of the unreduced sum occurs when the jth components of the addend and the augend are maximum, i.e., equal to m_j-1. The maximum sum will be m_j-2.

Let j = n. The maximum unreduced nth component will be $2m_n$-2; therefore, the carry can only be zero or one. Consider j = n-1. The component $2m_{n-1}$-2 will produce a carry. If a carry was generated by the nth position m_{n-1}-2 + $1m_{n-1}$; therefore, no carry is both propagated and generated.

Assume the results true for j = m. In this case $2m_{m-1}$-2 will generate a

carry of one, and $2m_{m-1}-2 + 1$ will also give rise to a carry of one.

4.2 OVERFLOW

A problem of the residue number system which can be solved with the mixed base system is overflow. In a mixed base number system where $m_1 = 2$, the integers less than M/2 are represented in the system. Therefore, overflow is defined to be the condition where the true arithmetic result is an integer larger than or equal to M/2. First additive overflow will be discussed and then various conditions for the absence of multiplicative overflow will be demonstrated. (Due to sign detection and overflow conditions it will be convenient to assume $m_1 = 2$ for the remainder of this chapter. In this chapter all n-tuples will be elements of the mixed base number system.)

Theorem XXIII. If the sum of two positive elements of the mixed base system is $(z_1,\ldots,z_n)$ additive overflow occurs if and only if $z_1 = 1$.

Proof: $(z_1,z_2,\ldots,z_n)$ represents the integer

$$z_1\frac{M}{2} + z_2\frac{M}{2m_2} + \ldots + z_n \bmod M \qquad (4\text{-}3)$$

and overflow occurs when

$$z_1\frac{M}{2} + z_2\frac{M}{2m_2} + \ldots + z_n \geq \frac{M}{2}$$

This will clearly be the case if $z_1 = 1$.

The largest value that $z_2\frac{M}{2m_2} + \ldots + z_n$ can attain has been shown to be $\frac{M}{2} - 1$. The largest sum possible is $2\left(\frac{M}{2} - 1\right) = M-2 < M$. Thus expression (4-3) becomes $z_1\frac{M}{2} + z_2\frac{M}{2m_2} + \ldots + z_n$ and the other conclusion follows.

To insure that multiplicative overflow will not occur, conditions will be

given which will insure that no multiplicative overflow will occur as the partial products are formed. It is also necessary that overflow does not occur when the partial products are summed. This will also be treated.

Consider $(y_1,y_2,\ldots,y_n) \cdot (x_1,0,\ldots,0)$. Since $(x_1,0,\ldots,0) \longleftrightarrow x_1\frac{M}{2}$, overflow will occur unless $y_1 = y_2 = \ldots = y_n = 0$.

Consider next $(y_1,y_2,\ldots,y_n) \cdot (0,x_2,0,\ldots,0)$ and note that $(0,x_2,0,\ldots,0) \longleftrightarrow x_2\frac{M}{2m_2}$. Therefore, the condition which is necessary and sufficient for the absence of multiplicative overflow in this partial product is:

$$y_1 = y_2 = \ldots y_{n-1} = 0 \text{ and } x_2 \cdot y_n < m_2.$$

Since $(0,0,x_3,0,\ldots,0)$ represents $x_3\frac{M}{2m_2m_3}$ and $(y_1,y_2,\ldots,y_n)$ represents $Y = y_n + y_{n-1}m_n + y_{n-2}m_nm_{n-1} + \ldots + y_1\frac{M}{m_1}$, a necessary and sufficient condition to prevent overflow is

$$x_3 \frac{M}{2m_2m_3} Y < \frac{M}{2} \tag{4-4}$$

or

$$x_3 Y < m_2m_3$$

Thus it is seen that condition (4-4) becomes

$$x_3(y_n + y_{n-1}m_n) < m_2m_3 \tag{4-5}$$

and

$$y_1 = y_2 = \ldots = y_{n-2} = 0.$$

Since $y_n < m_n$ we may substitute for expression (4-5)

$$x_3m_n(y_{n-1} + 1) < m_2m_3$$

to obtain a sufficient condition.

Consider now the general partial product $(y_1,y_2,\ldots,y_n) \cdot (0,\ldots,0,x_j,0,\ldots,0)$. In this case $(0,\ldots,0,x_j,0,\ldots,0)$ represents $x_j\frac{M}{2m_2\ldots m_j}$.

To guarantee that no overflow will occur we must satisfy the inequality

$x_j Y < m_2 \cdots m_j$ or

$$x_j(y_n + y_{n-1}m_n + \cdots + y_{n-(j-2)}m_n \cdots m_{n-(j-3)} + y_{n-j+1}m_n m_{n-1} \cdots m_{n-j-2} + \cdots + y_1 \frac{M}{2}) < m_2 m_3 m_j \tag{4-6}$$

The condition becomes

$$y_{n-j+1} = y_{n-j} = \cdots = y_1 = 0$$

and

$$x_j(y_n + y_{n-1}m_n + \cdots + y_{n-(j-2)}m_n \cdots m_{n-(j-3)}) < m_2 m_3 m_j. \tag{4-7}$$

This constitutes a necessary and sufficient condition that this partial product does not produce an overflow. Again there are simpler inequalities the validity of which will imply the validity of (4-7). These inequalities are

$$x_j[(y_{n-1} + 1)m_n + \cdots + y_{n-(j-\omega)}m_n \cdots m_{n-(j-3)}] < m_2 m_3 \cdots m_j.$$

$$x_j[(y_{n-2} + 1)m_n m_{n-1} + \cdots + y_{n-(j-2)}m_n \cdots m_{n-(j-3)}] < m_2 m_3 \cdots m_j.$$

$$\vdots$$

$$x_j[(y_{n-(j-2)} + 1)m_n \cdots m_{n-(j-3)}] < m_2 m_3 \cdots m_j.$$

The sufficiency of the above inequalities stems from Theorem XXIV.

Theorem XXIV. If $x_i < m_i$, then

$$x_1 + x_2 m_1 + x_3 m_1 m_2 + \cdots + x_{k-1}m_1 m_2 \cdots m_{k-2} < m_1 m_2 \cdots m_{k-1}.$$

Proof: The maximum value that

$$x_1 + x_2 m_1 + x_3 m_1 m_2 + \cdots + x_{k-1}m_1 m_2 \cdots m_{k-2}$$

can attain is

$$(m_1 - 1) + (m_2 - 1)m_1 + (m_3 - 1)m_1m_2 + \ldots + (m_{k-1} - 1)m_1m_2\ldots m_{k-2}$$

$$= m_1m_2\ldots m_{k-2}m_{k-1} - 1 < m_1m_2\ldots m_{k-1}.$$

Even when none of the partial products of a multiplication involve multiplicative overflow, overflow may result when the partial products are summed. Such overflow will be avoided only if the first coefficient of the unreduced sum is zero and no carry is propagated into that position.

4.3 DIVISION IN THE MIXED BASE SYSTEM

Utilizing the overflow rules given in this chapter, one may now state rules for performing division in the mixed base system. The conditions for the absence of multiplicative overflow and rules for forming partial products provide a means for estimating trial divisors. The subtraction rules then allow one to determine the new dividend. The method will be demonstrated before the formal rules are stated.

Example: Divide 95 by 14 using the mixed base system where $m_1 = 2$, $m_2 = 3$, $m_3 = 5$, and $m_4 = 7$.

$$95 \longleftrightarrow (0,2,3,4)$$
$$14 \longleftrightarrow (0,0,2,0)$$

Step 1: Determine first divisor

$$0,0,2,0 \quad \overset{\alpha}{\overline{)0,2,3,4}}$$

It is seen that it is necessary for $\alpha = 0$ to avoid overflow.

Step 2: Determine second divisor

$$0,0,2,0 \quad \overset{0,\beta}{\overline{)0,2,3,4}}$$

Again to avoid overflow $\beta = 0$.

Step 3: Determine third divisor

$$0,0,2,0 \quad \overline{)0,2,3,4} \quad 0,0,\gamma$$

From overflow considerations it is seen that $\gamma = 1$ is satisfactory.

$$\begin{array}{l} 0,0,2,0 \quad \overline{)0,2,3,4} \quad 0,0,1 \\ \qquad\qquad\qquad\quad \underline{0,2,4,0} \end{array}$$

It is seen that $\gamma = 1$ is too large; therefore, $\gamma = 0$.

Step 4: Determine fourth divisor

$$0,0,2,0 \quad \overline{)0,2,3,4} \quad 0,0,0,\zeta$$

From reference to overflow rules and multiplication the estimate is $\zeta = 6$.

$$\begin{array}{l} 0,0,2,0 \quad \overline{)0,2,3,4} \quad 0,0,0,6 \\ \qquad\qquad\qquad\quad \underline{0,2,2,0} \\ \qquad\qquad\qquad\quad 0,0,1,4 \end{array}$$

The division is completed giving $95 = 6 \cdot 14 + 11$.

The procedure for determining the trial divisor is as follows:

1. Consult the conditions for the absence of multiplicative overflow to determine the possible range of trial divisors.
2. Use the rules for forming partial products to determine a trial divisor which will yield the required zero(s) in the most significant place(s) and a non-zero component in the most significant non-zero position (kth) of the dividend. (The kth component must be less than or equal to the kth component of the dividend.)
3. Subtract the partial product from the dividend and if necessary

revise the quotient so that the least possible non-negative result is achieved.

4.4 DIGIT FILL-IN

An addition problem which can be solved by using the mixed base system is the problem of digit fill-in. If one is given the coordinates of a vector representing the integer relative to the basis $A = \{\alpha_1,\alpha_2,\ldots,\alpha_n\}$ and base moduli $m_1,m_2,\ldots,m_n$, the problem is to express the integer s in terms of coordinates relative to the basis $\{\beta_1,\beta_2,\ldots,\beta_n,\beta_{n+1},\ldots,\beta_{n+m}\} = B$ and base moduli $m_1,m_2,\ldots,m_n,m_{n+1},\ldots,m_{n+m}$. The m_i are pairwise relatively prime.

Since one can represent s as a vector relative to the base moduli $m_1,m_2,\ldots,m_n$, s is less than $\prod_{i=1}^{n} m_1$. Therefore, if s is expressed in the mixed base system relative to the moduli $m_{n+1},m_{n+2},\ldots,m_{n+m},m_1,m_2,\ldots,m_n$, the coordinates with weights greater than or equal to $\prod_{i=1}^{n} m_i$ must be zeros. The weights of the last n components of the mixed system will be in reverse order, $1,m_n,m_nm_{n-1},\ldots,m_nm_{n-1}\ldots m_2$. These weights are the same as the weights of the components in the mixed base system relative to the primes $m_1,m_2,\ldots,m_n$. Therefore, digit fill-in is accomplished as follows:

1. Perform the change of basis operation from the basis A to the mixed base system.
2. Prefix the m zeros to produce the correct representation in the extended mixed base system.
3. Perform the change of basis operation, this time from the extended mixed base system to the basis B.

CHAPTER V

OTHER NUMBER SYSTEMS RELATED TO THE RESIDUE NUMBER SYSTEM

5.1 PARTITIONING PROPERTIES

In Chapter IV it was noted that the mixed base number system representations corresponding to consecutive integers are partitioned by the first coordinate. It is this property which permits a rapid solution to the sign detection problem. A question arises whether other number systems exist which possess the desirable partitioning while having simpler rules of arithmetic manipulation. It will be shown that the only non-redundant number system to achieve a partitioning of elements representing consecutive integers is the mixed base system.

Lemma 2: For any integer k within the range $1 < k < p$, there exists an integer ℓ such that

$$p \leqslant \ell k < 2p \qquad \text{where } \ell < p \qquad \text{and } p > 2.$$

Proof: It is evident that k cannot lie in the range

$$1 \leqslant k < \frac{p}{p-1},$$

for

$$p > 2,\ p-2 > 0$$

$$2p-< > p$$

$$2(p-1) > p;$$

therefore,

$$\frac{p}{p-1} < 2.$$

Thus there exists an integer ℓ such that

$$\frac{P}{\ell} \leqslant k < \frac{P}{\ell-1} \qquad \text{with } p > \ell$$

and one concludes $p \leqslant \ell k < \frac{\ell}{\ell-1}\, p \leqslant 2p$.

Theorem XXV. If the base moduli are ordered $m_1, m_2, \ldots, m_n$, only the mixed base number system gives a partitioning of the elements into those elements which represent integers in the range less than $\frac{(c+1)M}{m_1}$ but greater than or equal to $c\,\frac{M}{m_1}$, where c is the first coordinate of the element.

Proof: We will consider all number systems which give the desired partitioning and conclude that they are all identical to the mixed base system.

Consider the number system based on the vectors $< \beta_1, \beta_2, \ldots, \beta_n >$. It is assumed that this number system achieves the desired partitioning. Here β_1 corresponds to B_1; β_2 to B_2 etc. An element of this number system $Y = (y_1, y_2, \ldots, y_n)$ represents the integer.

$$y_1B_1 + y_2B_2 + \ldots + y_nB_n \bmod M.$$

For the set $< \beta_1, \ldots, \beta_n >$ to be a basis it must be triangular; therefore, one may state

$$\begin{aligned} B_n &= k_n \\ B_{n-1} &= k_{n-1}m_n \\ &\vdots \\ B_j &= k_j \prod_{i=j+1}^{n} m_i \\ &\vdots \\ B_1 &= k_1 \prod_{i=2}^{n} m_i \end{aligned}$$

where

$$0 < k_j < \prod_{i=1}^{j} m_i .$$

Consider $y_1 = c$ and $y_i = 0$ for $i \neq 1$. The expression $y_1B_1 + y_2B_2 + \cdots y_nB_n$ mod M becomes $ck_1 \frac{M}{m_1}$ mod M. Clearly, $\frac{cM}{m_1} \leqslant ck_1 \frac{M}{m_1} < (c+1) \frac{M}{m_1}$ for $c < m_1$ requires $k_1 = 1$. One condition which must be satisfied is $y_1 = 0 \cdot \equiv \cdot Y < \frac{M}{m_1}$. This requirement becomes

$$y_2B_2 + y_3B_3 + \cdots + y_nB_n \equiv z \bmod M \qquad (5\text{-}1)$$

where $z < \frac{M}{m_1}$ for all $y_2, y_3, \cdots, y_n$.

Consider first the case

$$y_2 \neq 0, \; y_3 = \cdots = y_n = 0.$$

We have

$$y_2B_2 = y_2k_2 \frac{M}{m_1m_2}.$$

If there is an integer ℓ in the range $0 < \ell < m_2$ that

$$\frac{M}{m_1} < \ell k \frac{M}{m_1m_2} < M \qquad (5\text{-}2)$$

condition (5-1) is violated. Expression (5-2) may be written

$$m_2 < \ell k_2 < m_1m_2 \qquad (5\text{-}3)$$

By Lemma 2, it is seen that for $k_2 > 1$, we can find an integer in the range $0 < \ell < m$ such that $m_2 \leqslant \ell k_2 < 2m_2$. Since $m_1 \geqslant 2$, inequality (5-3) is satisfied. Therefore, $k_2 = 1$.

Assume that we have shown $k_i = 1$ for $i = 1,2,\ldots,m$. Let $y_i = m_i - 1$ for $i = 1,2,\ldots,m, y_{m+1} \neq 0$, and $y_{m+2} = y_{m+3} = \cdots = y_n = 0$. Consider

$$(m_2 - 1) \frac{M}{m_1m_2} + (m_3 - k) \frac{M}{m_1m_2m_3} + \cdots + \frac{(m_{m-1})M}{m_1m_2\cdots m_m} + y_{m+1}k_{m+1} \frac{M}{m_1m_2\cdots m_m}$$

or

$$(m_2 - 1)m_3\cdots m_n + (m_3 - 1)m_4\cdots m_n + \cdots + (m_m - 1)m_{m+1}\cdots m_n + y_{m+1}k_{m+1}m_{m+2}\cdots m_n$$

which yields

$$(m_2 m_3 \cdots m_n) - (m_{m+1} \cdots m_n) + y_{m+1} k_{m+1} m_{m+2} \cdots m_n .$$

But $\frac{M}{m_1} = m_2 m_3 \cdots m_n$.

Thus the sum s is

$$s = \frac{M}{m_1} - m_{m+1} \cdots m_n + y_{m+1} k_{m+1} m_{m+2} \cdots m_n .$$

If

$$\frac{M}{m_1} \leqslant s < M \tag{5-4}$$

condition (5-1) will be compromised. Expression (5-4) becomes

$$m_{m+1} \cdots m_n \leqslant y_{m+1} k_{m+1} m_{m+2} \cdots m_n < (m_1 - 1) \frac{M}{m_1} + m_{m+1} \cdots m_n$$

which upon substitution for M yields

$$m_{m+1} \cdots m_n \leqslant y_{m+1} k_{m+1} m_{m+2} \cdots m_n < [m_2 \cdots m_m (m_1 - 1) + 1] m_{m+1} \cdots m_n .$$

Removing the common factor in the above expression one obtains

$$m_{m+1} \leqslant y_{m+1} k_{m+1} < [m_2 \cdots m_m (m_1 - 1) + 1] m_{m+1} . \tag{5-5}$$

If $m_{m+1} < k_{m+1} < (m_1 - 1) m_2 \cdots m_{m+1}$, (5-5) is satisfied by $y_{m+1} = 1$. If $k_{m+1} < m_{m+1}$ and $k_{m+1} > 1$, Lemma 2 guarantees that there exists a y_{m+1} such that

$$m_{m+1} < y_{m+1} k_{m+1} < 2 m_{m+1}$$

but since

$$2 < [m_2 \cdots m_m (m_1 - 1) + 1].$$

It is necessary that $k_{m+1} = 1$ or that

$$(m_1 - 1) m_2 \cdots m_{m+1} < k_{m+1} < m_1 m_2 \cdots m_{m+1} .$$

To see that k_{m+1} cannot be in the range

$$(m_1 - 1) m_2 \cdots m_{m+1} < k_{m+1} < m_1 m_2 \cdots m_{m+1},$$

consider $y_{m+1} = 1$ and $y_i = 0$ for $i \neq m + 1$. Then

$$s = B_{m+1}k_{m+1}m_{m+2}\cdots m_n$$

and

$$\frac{M}{m_1} \leqslant (m_1 - 1)m_2\ldots m_{m+2}\cdots m_n < s < m_1m_2\ldots m_{m+2}\cdots m_n = M.$$

Again we have satisfied condition (5-4); it is necessary that $k_{m+1} = 1$.

Therefore, the only number system which effects the partition described is that system having $k_i = 1$ for $i = 1,\ldots,n$. This system is the mixed base number system.

Theorem XXVI. If the base moduli are ordered $m_1,m_2,\ldots,m_n$, there is only one number system with the property that the first coordinate partitions elements which represent consecutive integers into m_1 classes. That number system is the mixed base number system.

Proof: Theorem XXV shows that there exists but one number system which partitions the elements into those elements which represent integers in the range less than $(c + 1)\frac{M}{m_1}$ but greater than or equal to $c\frac{M}{m_1}$, where c is the first coordinate of the element. By definition this number system is the mixed base number system.

It remains to show that no number system exists which effects the same partitioning for $y_1 \neq c$. Assume such a number system exists with basis $\beta_1',\beta_2',\ldots,\beta_n'$. An element of the number system represents the integer

$$y_1\beta_1' + y_2B_2' + \ldots + y_nB_n' \bmod M \qquad (5\text{-}6)$$

Again we may state

$$B_n' = k_n$$

$$B_{n-1}' = k_{n-1}m_n$$

$$\vdots$$

$$B_j' = k_j \prod_{i=j+1}^{n} m_i$$

$$\vdots$$

$$B_1' = k_1 \prod_{i=2}^{n} m_i$$

where $0 < k_j < \prod_{i=1}^{j} m_i$. The range imposed on k_1 is $0 < k_1 < m_1$. Thus we have

$$B_1' = k_1 M/m_1$$

and expression (5-6) becomes

$$y_1 k_1 \frac{M}{m_1} \bmod M \text{ when } y_1 \neq 0,\ y_2 = \ldots = y_n = 0.$$

It will now be shown that there exists an integer c in the range $0 \leqslant c < m_1$ such that

$$c \frac{M}{m_1} \leqslant y_1 k_1 \frac{M}{m_1} \bmod M < (c + 1) \frac{M}{m_1} \tag{5-7}$$

cannot be satisfied for $0 < k_1 < m_1$, $0 < y_1 < m_1$, and $y_1 \neq c$. Take $c = 0$. Expression (5-7) becomes

$$0 \leqslant y_1 k_1 \frac{M}{m_1} \bmod M < \frac{M}{m_1}. \tag{5-8}$$

Condition (5-8) will be satisfied only if

$$y_1 k_1 \equiv 0 \bmod m_1.$$

Since $y_1 \neq 0$, (5-9) requires that $k_1 = 0$. This is the desired contradiction which completes the proof.

Corollary 1: If the base moduli are ordered $m_1 = 2$, m_2, $m_3, \ldots, m_n$ there is one and only one number system with the property that the first coordinate partitions the elements into two classes, the representations of the integers less than $\frac{M}{2}$, and the representations of integers greater than or equal to $\frac{M}{2}$.

Proof: This corollary follows from Theorem XXVI with $m_1 = 2$.

One now asks whether a number system with base moduli $m_1, m_2, \ldots, m_n$ exists which partitions elements which represent consecutive integers into m_j classes with the first coordinate where $j \neq 1$?

The number of elements having the same jth coordinate in any system having m_j as a base prime is

$$\frac{M}{m_j} = \prod_{\substack{i=1 \\ i \neq j}}^{n} m_i$$

Likewise, the number of elements associated with a particular value of the first coordinate is

$$M/m_1 = \prod_{i=2}^{n} m_i$$

The number of elements in the two cases differs and the greatest common multiple is one. Therefore, the answer to the question posed in the preceding paragraph is negative.

A similar argument shows that no number system with base moduli $m_1, m_2, \ldots, m_n$ where $(m_1, m) = (m_2, m) = \ldots = (m_n, m) = 1$ exists which partitions with the first coordinate elements which represent consecutive integers into two classes, one class the elements of which represent integers less than M/m and the other class representing integers greater than or equal to M/m. The argument follows:

Since M is relatively prime to m, $m \nmid M$, but $m/M - \ell$ when $0 < \ell < m$. Let $m - \ell = d_m$. If $m < m_1$, $m_1 \nmid M - \ell$. If $m > m_1$

$$m_1 \nmid M - \ell \text{ for } cm_1 < \ell < (c + 1)m_1.$$

Suppose $km_1 = \ell$, then $m - \ell = m_1, m_2, \ldots, m_n - m_1 k$ but $(m, m_1) = 1$; therefore, $m \nmid M - \ell$ which is a contradiction. The only related number system which produces a partitioning is the mixed base number system and; therefore, the proof is completed.

From the theorems and arguments advanced thus far in this chapter, one concludes that the mixed base number system is the only number system which partitions the elements representing consecutive integers. In particular only the mixed base system with m_1 even partitions the elements representing consecutive integers into two groups--(1) elements corresponding to positive integers and (2) elements representing complements. Thus if one wishes to use a number system to determine the sign of the residue element, he will find it necessary to use the mixed base system.

5.2 NUMBER SYSTEMS ALLOWING SIGN DETECTION WITH FEWER THAN N-CARRIES

It has been suggested that the use of number systems which are neither strictly residue nor strictly mixed base might ease the carry situation in sign determination. Such systems are those in which a certain number of carries are eliminated from the operation of expressing a vector in the mixed base system.

As developed in the chapter concerning the Carry Algorithm, a vector X with coordinates $(x_1, x_2, \ldots, x_n)$ relative to the basis $< \beta_1, \ldots, \beta_n >$ is expressed with coordinates $(y_1, y_2, \ldots, y_n)$ relative to the mixed base basis $< \alpha_1, \ldots, \alpha_n >$ by determining the $q = \|q_{ij}\|$ matrix and following with the matrix multiplication $X \cdot Q = Y$. The elements of the Q matrix are governed by the equation

$$\beta_i = q_{i1}\alpha_1 + \ldots + q_{in}\alpha_n \text{ for } i = 1,2,\ldots,n. \qquad (5\text{-}10)$$

The Y so obtained will not in general be a linear form with restricted coefficients. Carrys must then be propagated from each position to the next more significant position. The general conversion will require up to n-1 carries. To reduce the maximum possible number of carries which can occur by say m-1 carries, it will be necessary and sufficient to guarantee

$$y'_k < m_k \text{ for } k = n\text{-}m,\ldots,n. \qquad (5\text{-}11)$$

This is in turn equivalent to the condition

$$q_{ik} = \delta_{ik} \text{ for } i,k = n\text{-}m,\ldots,n. \qquad (5\text{-}12)$$

Condition (5-12) may be expressed as

$$\beta_k = \alpha_k + \sum_{i=1}^{n-m-1} c_{ik}\alpha_i \text{ for } k = n\text{-}m,\ldots,n. \qquad (5\text{-}13)$$

If $\|a_{ij}\|$ is the array of the mixed base basis vectors, consider the m x m sub-array in the lower right corner. Equation (5-13) states that this sub-array must be preserved in $\|b_{ij}\|$, the array of the β vectors. The only other requirement is that $\|b_{ij}\|$ be triangular. Other considerations which affect the selection of the remaining elements in the β array stem from a desire to simplify the carry structure in the β system.

Since $a_{ij} = b_{ij}$ for i, j = n-m,...,n the carry structure in the last m positions is fixed. Carries from the kth components to the jth components where

$$k = n\text{-}m,\ldots,n$$

$$j = 1,2,\ldots,n\text{-}m\text{-}1$$

can be prevented by the following constraint

$$b_{ij} = 0 \text{ for } i - n-m,\ldots,n$$

$$j = 1,2,\ldots,n-m-1.$$

Further carries can be eliminated by making the upper right partition of the array the identity matrix.

Since $a_{ij} = b_{ij}$ for i, j = n-m,...,n carries from the ℓth component to the (ℓ-1)st component will occur for ℓ = n-m+1,...,n. Therefore, at least m-1 carries will occur in the β system. The total number of carries in the β system for a subtraction followed by sign detection is at least as great as for subtraction in the residue number system and conversion to the mixed base system.

Such number systems do not appear practical, for nothing is gained in addition and sign detection. In fact, from the Multiplication Algorithm of Chapter III, it is clear that much speed is sacrificed in multiplication.

5.3 GENERALIZATION OF SIGN DETECTION FOR THE RESIDUE NUMBER SYSTEM

Let a residue number system be defined as

$$S_{RN} = y_1 \frac{P_n}{m_1} + \ldots + y_n \frac{P_n}{m_n} \tag{5-14}$$

$$(m_i, m_j) = 1 \text{ except for } i = j$$

where

$$\left|S_R\right|_m = X$$

$$\left|X\right|_{m_i} = x_i$$

$$\left|x_i k_i\right|_{m_i} = y_i$$

$$\left|k_i \frac{P_n}{m_i}\right|_{m_i} = 1$$

$$\mathbf{P}_u = \prod_{i=1}^{u} m_i, 1 \leq [u] \leq n; \quad P_o = 1$$

$$m = p_n$$

The residue number system is a member of the general class of nonredundant weighted number systems which satisfy the following relationship:

$$\left|S\right|_m = z_1\rho_1 + \ldots + z_n\rho_n \tag{5-15}$$

The necessary and sufficient conditions for the validity of this relationship is the triangularity of the weight matrix (See Theorem XV and XVI).

Consider next the special mixed base number system which satisfies the relationship. It has been shown in the previous sections of this chapter that the only non-redundant number systems having simple sign properties is the mixed base system.

$$S_M = \sum_{i=0}^{n} Z_i \frac{P_n}{P_i} \tag{5-16}$$

Z_0 is identified as A(x) and z_1 is the high order digit of X. Any non-redundant weighted number system S must satisfy the relation

$$S = S_M. \tag{5-17}$$

where $S = z_1\rho_1 + \ldots + z_n\rho_n$.

The following relations are also valid if $S = S_M$

$$\begin{aligned} kS &= kkS_M \\ [kS] &= [kS_M] \\ \{kS\} &= \{kS_M\} \\ \left|kS\right|_{m_j} &= \left|kS_M\right|_{m_j} \end{aligned} \tag{5-18}$$

Consider the normalized residue code

$$S_{RN} = S_M \tag{5-19}$$

$$\left[\frac{S_{RN}\, P_i}{P_n}\right] = \left[\frac{S_M\, P_i}{P_n}\right]$$

$$\left[\sum_{j=1}^{n} \frac{y_i P_i}{m_j}\right] = \sum_{\ell=0}^{i} z_\ell \frac{P_i}{P_\ell} \tag{5-20}$$

It is possible to further reduce the left hand side of the equation when $m_j | P_i$. This occurs when $i \geqslant j$.

$$\sum_{j \leqslant i} y \frac{P_i}{m_j} + \left[\sum_{j>i}^{n} \frac{Y_j P_i}{m_j}\right] = \sum_{\ell=0}^{i} z_\ell \frac{P_i}{P_\ell} \tag{5-21}$$

The i-1 term associated with equation (5-20) is

$$\left[\sum_{j=1}^{n} \frac{y_j P_{i-1}}{m_j}\right] = \sum_{\ell=0}^{i-1} z_\ell \frac{P_{i-1}}{P_\ell} \tag{5-22}$$

Both sides of equation (5-21) are multiplied by m_i and the resulting equation is subtracted from equation (5-20) to give

$$\left[\sum_{j=1}^{n} \frac{y_j P_i}{m_j}\right] - m_i \left[\sum_{j=1}^{n} \frac{y_j P_{i-1}}{m_j}\right] = z_i \quad 1 \leqslant [i] \leqslant n \tag{5-23}$$

Reduction of the left side of equation (5-23) yields

$$y_i P_{i-1} + \left[\sum_{j \geqslant i}^{n} \frac{y_j P_i}{m_j}\right] - m_i \left[\sum_{j \geqslant i-1}^{n} \frac{y_j P_{i-1}}{m_j}\right] = z_i \tag{5-24}$$

The identity

$$[\alpha] - m_i \left[\frac{\alpha}{m_i}\right] = |[\alpha]|_{m_i} \tag{5-25}$$

may be applied to equation (5-23) to give

$$\left|\left[\sum_{j=1}^{n} \frac{y_j P_i}{m_j}\right]\right|_{m_i} = z_i \tag{5-26}$$

$$0 \leqslant [i] \leqslant n \text{ where } m_0 = \infty$$

Equation (5-26) may be reduced to

$$\left|\sum_{j<i} y_j \frac{P_i}{m_j} + \left[\sum_{j>i}^{n} \frac{y_j P_i}{m_j}\right]\right|_{m_i} = z_i$$

$$\left| y_i P_{i-1} + \left[\sum_{j>i}^{n} \frac{y_j P_i}{m_j} \right] \right|_{m_i} = z_i \qquad (5\text{-}27)$$

The special case $2 \mid m_1$, $z_1 = z_{11} \frac{m_1}{2} + Z_{12}$; $0 \leqslant \left[z_{11} \right] \leqslant 1$; $0 \leqslant \left[z_{12} \right] < \frac{m_1}{2}$ may be handled within the framework of the previous formulas. In particular

$$z_{11} = \left[\frac{2y_1}{m_1} + \ldots + \frac{2y_n}{m_n} \right] - 2 \left[\frac{y_1}{m_1} + \ldots + \frac{y_n}{m_n} \right] \qquad (5\text{-}28)$$

and

$$z_{11} = \left| \left[\frac{2y_1}{m_1} + \ldots + \frac{2y_n}{m_n} \right] \right|_2 \qquad (5\text{-}29)$$

or in general when $2 \nmid m_1$

$$z_1 = \left| y_1 + \left[\frac{m_1}{m_2} y_2 + \ldots + \frac{m_1}{m_n} y_n \right] \right|_{m_1} \qquad (5\text{-}30)$$

Equation (5-29) and (5-30) should be regarded as the fundamental definition of the sign digit. All known methods of sign detection may be interpreted in terms of these equations.

CHAPTER VI

DIGIT ENCODING IN ARITHMETIC OPERATIONS

This chapter discusses some coding schemes for individual moduli and the resulting facility of arithmetic operations. Two types of natural encoding result depending upon how one represents the integers with respect to multiplication. The first approach analyzes the abstract structure of the given prime subring, i.e., residue digit. It is recognized that still a further decomposition is possible with respect to multiplication. A representation scheme is suggested by the direct sum decomposition of the component multiplicative subgroups. The second approach treats multiplication as a linear transformation on the additive subring. Such a treatment suggests a weighted code so that the fundamental property of multiplication, its distributivity, may be employed. This is the more conventional approach. It will be shown how codes may be constructed for arbitrary moduli by the use of redundancy. Furthermore, the redundancy may itself be used to speed up arithmetic operations.

6.1 CODES BASED UPON THE GROUP STRUCTURE OF THE MULTIPLICATIVE SUBGROUP

The fundamental idea of the residue number system is the ring decomposition:

$$R_M = R_{p_1^{\alpha_1}} \oplus R_{p_2^{\alpha_2}} \oplus \cdots \oplus R_{p_m^{\alpha_m}}$$

$$\prod p_i^{\alpha_i} = M \qquad (p_i \cdot p_j) = 1 \qquad i \neq j$$

where R_i denotes the ring of integers modulo i.

If the p_i are taken prime, there is no further ring decomposition possible. Nevertheless, the subrings $R_{p_i^{\alpha_i}}$ possess multiplicative groups which may be further decomposed. In general, the order of the multiplicative group of $R_{p_i^{\alpha_i}}$ is a group of order $\phi(p_i^{\alpha_i})$ where ϕ is the Euler function. If p is prime and $\alpha = 1$, then $\phi(p) = p-1$ and the multiplicative subgroup includes all but the zero of the subring.

Since p-1 is not prime, this cyclic group may be represented as the product of its prime power cyclic subgroups. In number theoretic terms we are just saying that multiplication may be done by taking the index function of the additively expressed integers and performing a residue encoding.

As an example, take R_{31}. Column I contains the additive code for R_{31}, column II the index or multiplicative representation, and columns II a, b, c the decomposed or residue representation of II.

If $\alpha_i \neq 1$ then the multiplicative subgroup will no longer include all but the zero of $R_{p_i^{\alpha_i}}$, since p, 2p, 3p, etc., will be "divisors of zero." In this case, a larger system may be constructed of which $R_{p_i^{\alpha_i}}$ is a homomorphic image. A semigroup acting like an exponent of p is adjoined to the multiplicative subgroup representation. This semigroup $E_\alpha = \{0,1,\ldots,\alpha\}$ has the operation defined as follows:

$$x \cdot y = \max(\alpha; x+y) \qquad x,y \epsilon E_\alpha$$

Every element in $R_{p\alpha}$ is represented in the form $x = yp^i$ where y is in the multiplicative subgroup and i is in E_α. The whole system is then

$$C_{q_1} \oplus C_{q_2} \cdots \oplus C_{q_n} \oplus E_\alpha$$

$$\prod_{i=1}^{n} q_i = \phi(p^{\alpha_i})$$

TABLE 1

ENCODING OF THE MULTIPLICATIVE SUBGROUP

I	II	III			I	II	III		
		a	b	c			a	b	c
C_{31}	C_{30}	C_2	C_3	C_5	C_{31}	C_{30}	C_2	C_3	C_5
0	--	--	--	--	16	6	0	1	1
1	0	0	0	0	17	7	1	1	2
2	24	0	0	4	18	26	0	2	1
3	1	1	1	1	19	4	0	1	4
4	18	0	0	3	20	8	0	2	3
5	20	0	2	0	21	29	1	2	4
6	25	1	1	0	22	17	1	2	2
7	28	0	1	3	23	27	1	0	2
8	12	0	0	2	24	13	1	1	3
9	2	0	2	2	25	10	0	1	0
10	14	0	2	4	26	5	1	2	0
11	23	1	2	3	27	3	1	0	3
12	19	1	1	4	28	16	0	1	1
13	11	1	2	1	29	9	1	0	4
14	22	0	1	2	30	15	1	0	0
15	21	1	0	1					

As an example take $R_{27} = R_3^3$. Column I gives their additive representation, column II the index if it exists or a factorization yp^i if $i \neq 0$. Column III a, b the residue representation of the index and column IV the exponent. Notice that the system contains a certain amount of redundancy. The number six could have just as easily been represented at 11.3^1 or 20.3^1.

TABLE 2

FACTORIZATION PROCEDURE

I	II	III a	III b	IV	I	II	III a	III b	IV
C_{27}	CL_8 or Factorization	C_9	C_2	E_3	C_{27}	CL_8 or Factorization	C_9	C_2	E_3
0	$\underline{1.3^3}$	0	0	3	14	17	8	1	0
1	0	0	0	0	15	$\underline{5.3^1}$	5	1	1
2	11	1	0	0	16	4	4	0	0
3	$\underline{1.3^1}$	0	0	1	17	15	6	1	0
4	2	2	0	0	18	$\underline{2.3^2}$	1	0	2
5	5	5	1	0	19	12	3	0	0
6	$\underline{2.3^1}$	1	0	1	20	7	7	1	0
7	16	7	0	0	21	$\underline{7.3^1}$	7	0	1
8	3	3	1	0	22	14	5	0	0
9	$\underline{1.3^2}$	0	0	2	23	11	2	1	0
10	6	6	0	0	24	$\underline{8.3^1}$	3	1	1
11	13	4	1	0	25	10	1	0	0
12	$\underline{4.3^1}$	2	0	1	26	9	1	0	0
13	8	8	0	0					

There appears to be at least two possibilities for application of the multiplication decomposition exhibited above.

The most obvious is to use a machine code that is based on this decomposition. The desirability of such a scheme appears questionable except in rather exceptional circumstances. The fundamental question, of course, is how one adds in such a code. If a reasonably fast technique for addition were forthcoming, it might be reasonable to consider using the multiplication code since multiplication in the multiplication code is considerably more easily accomplished than addition in the normal code due to reduction of carry problems. Nevertheless we are not too hopeful since there is no distributive law operat-

ing so that addition can be done by only multiplication as one can the converse in the normal situation.

The other possibility is to somehow use the multiplicative decomposition in the logical design of the multipliers where a more conventional code is used as the basic machine language. This is essentially dependent on conversion from conventional to multiplication code and reconversion. This problem ressembles the radix based-residue conversion problem except on a much reduced scale and the techniques developed there are applicable. Some techniques which appear quite expensive in time and hardware for the gross conversion problems will be applicable at this level because the order of the subring being decomposed will be of very moderate magnitude. A further distinction is that the completely reduced codes would only have to be added whereas in the ring decomposition one must both add and multiply in the subrings. The use of this code within residue multipliers would be indicated if mappings were used rather than sequential methods.

6.2 CODES BASED ON MULTIPLICATION AS A LINEAR TRANSFORMATION

The type of weighted binary code with which this section deals in the following. It consists of a set $X = \{x_0, \ldots x_{n-1}\}$, $x \epsilon \{0;1\}$ of 2^n elements. The set is interpreted by a set of weights $W = \{w_1 \ldots w_n\}$ and a modulus M where the w_i and M are integers with $w_i < M$. We further require the following relation on the weights

$$1 \epsilon W \qquad \ell \epsilon W \rightleftharpoons (2\ell \bmod M) \epsilon W$$

This latter property ensures that addition of code elements may be done using ordinary carry procedures. Also we shall consider only the case where M is an odd prime. The work is easily generalized to the case of composite moduli, at the expense of a certain amount of notational inconvenience.

A code or a subset of a code is said to be complet if for every positive integer $k < M$ there is at least one $x \varepsilon X$ such that

$$k \equiv \left(\sum_{l=0}^{n-1} x_i w_i \right) \bmod M.$$

It can be noted immediately that if $M < 2^m$, the entire code will be complete. We shall be primarily interested in the completeness of certain subsets of codes.

If $w_o = i$ and $w_{i+1} \equiv 2w_i \bmod M$, then because of the restriction on the weights and modulus, it must be that $2w_{n-1} \equiv 1 \bmod M$. Since for each w_i, there exists a w_j such that $w_j \equiv 2w_i \bmod M$ then a permutation $x_i \longrightarrow x_j$ results in multiplication by 2. Repeated shifts will therefore generate all powers of 2 modulo M. Thus there is set up a correspondence between certain elements of the code set and a cyclic permutation of the bits of the code. This correspondence is the basis of the ordinary pencil and paper method of doing multiplication as well as conventional machine multiplication with the slight difference that in our case we have a true permutation whereas conventionally one bit is "lost" and a zero introduced.

It should be clear at this point that we have described what is sometimes called a reduced radix code. This is the special case obtained when $M = 2^n - 1$. We mention another commonly known example of this class which is not usually

thought of as a weighted code at all. It is the one digit per wire code. In this case n = M-1, and the weight set includes all possible weights.

These two well known coding schemes are the extremes of all codes of this class. In the case of the former, there is almost no redundancy (zero has two forms) but the arithmetic must be sequentially. The one digit per wire code on the other hand has extremely high redundancy while permitting simple combinational arithmetic. It is our purpose to investigate those codes lying somewhere inbetween.

Consider the code M = 17, n = 8, W = {1,2,4,8,16,15,13,9}. In this case the weights are entirely generated by powers of two. This code may be viewed as using the reduced radix code for $2^8 = 255$ to represent numbers modulo a divisor of 255. Notice, however, that by introducing all this redundancy it is possible to select a complete subset of this code with some very nice properties. The following subset has the useful property that no element has more than two bits which take the value one.

This property ressembles the property of the one digit per wire code in that it facilitates arithmetic. It is not quite as easy to do arithmetic in this code, but on the other hand it is much less redundant. Addition may be done quite easily in this code using adders in each position consisting of one two input "and" and one three input "or."

At most, 2 levels of logic are required regardless of carries because carries can propagate only one level. This results from the additional property of the set chosen that no element has two adjacent "ones."

TABLE 3

THE CODE M = 17, n = 8

w_i	1 x_0	2 x_1	4 x_2	8 x_3	16 x_4	15 x_5	13 x_6	9 x_7
0	0	0	0	0	0	0	0	0
1	1	0	0	0	0	0	0	0
2	0	1	0	0	0	0	0	0
3	0	0	1	0	1	0	0	0
4	0	0	1	0	0	0	0	0
5	1	0	1	0	0	0	0	0
6	0	0	0	1	0	1	0	0
7	0	0	0	0	0	1	0	1
8	0	0	0	1	0	0	0	0
9	0	0	0	0	0	0	0	1
10	0	1	0	1	0	0	0	0
11	0	1	0	0	0	0	0	1
12	0	0	0	0	1	0	1	0
13	0	0	0	0	0	0	1	0
14	1	0	0	0	0	0	1	0
15	0	0	0	0	0	1	0	0
16	0	0	0	0	1	0	0	0

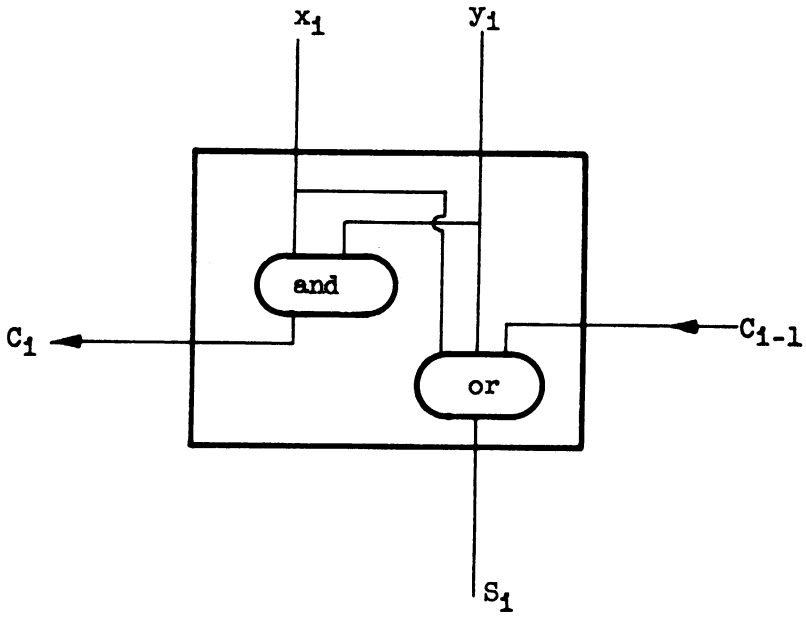


Figure 1. An adder for the M = 17, n = 8 code.

Unfortunately the sum produced may no longer be a member of this special subset. It is necessary to retrieve this proper form prior to doing further arithmetic. The nature of the complexity of this problem is not yet known sufficiently to make final judgement. It is known that in special cases, reduction networks may be designed such that the total addition time plus reduction is competitive if not superior to conventional binary addition. Futthermore the complexity of the coding networks seems to increase only linearly with the number of bits.

Notice that complementation may be done either by complementing each bit individually or by a simple permutation since "-1" is included as one of the weights.

If a complete set exists having only at most 2"one" bits per digit, multiplication is reduced to a single addition of the two appropriate permutations of the multiplicand. In the example given above, the weights were generated by successive powers of two. This is not the only manner in which a set of weights meeting the criteria may be generated. The set of powers of two form a multiplicative subgroup. Any coset of this subgroup is also closed under multiplcation by two and therefore may be included in the weight set. Consider the set of weights $\begin{Bmatrix}1,2,4,8,16\\3,6,12,24,17\end{Bmatrix}$ and M = 31. This code has a complete subset having the 2 "one" bit property. The only difficulty here is that there are no permutations corresponding to the coset weights {3,6,12,24,17}, and therefore multiplication cannot be carried out by successive shifts and adds. In order for this property to be retained, it is necessary that the cosets themselves form a subgroup under multiplication. Thus the set $\begin{Bmatrix}1,2,4,8,16\\30,29,27,23,15\end{Bmatrix}$

would be a better choice since the cosets are now isomorphic to the 2 element group {1,30}.

In general, it is easy to determine closed sets of weights of arbitrary order dividing M, using powers of primitive roots as generators. The problem of determing whether a code has a complete "2 one's" subset is more difficult. The coset decomposition of the entire multiplicative group allows a short cut to be used. If any element of a coset is representable with only 2 one bits, then every other element in the coset also is, since each is only a permutation away. Thus, only the least element in each coset need be examined.

Consider as a final example a coding for M = 127. Certainly the weight set must include {1,2,4,16,32,64}. If no additional weights are used, the almost non-redundant reduced radix code of seven bits results by adding the subgroup generated by $3^{63} \equiv 126$, the 14 element subgroup $\left\{\begin{matrix}1,2,4,8,16,32,64\\126,125,123,119,111,\\95,63\end{matrix}\right\}$ results.

This code has a "3 one's" complete subset but no "2 one's" complete set. If the subgroup of order 3 generated by 3^{42} is used, the resulting 21 weight set does yield a "2 one's" complete subset.

In concluding these remarks about redundant weighted codes it should be emphasized again that the recovery of the cannonical form of a number is the big question in the practical utilization of these codes. If moderate success is met here, the way is open to use very large moduli with a resulting saving of scaling and base extension time.

CHAPTER VII

DIVISION

7.1 INTRODUCTION

The definition of a division in Euclidean rings suggests that algorithms for performing it will be crucially tied up with magnitude comparison. Given two elements (a,b) it is desirable to find a (q,r) such that:

$$a = bq + r \text{ and } r < b.$$

The difficulty of recovering the natural magnitude ordering from the residue representation had initially led us to suppose that division within the system would be very difficult also. Recent workers have recognized essentially three different approaches.

a. Devise a really efficient magnitude comparison scheme; more conventional algorithms will then prove feasible.

b. Do division in some other closely related systems such as a mixed base system.

c. Find an algorithm which is independent of magnitude tests.

This chapter is devoted almost entirely to the approaches of the third type. The brief discussion of the first two approaches is given for purposes of contrast.

Considerations of the nature of the sign detection problem have been made in other parts of this report. While it is not so fair to say that no improvement can be made in the presently known sign determination scheme, it is clear that sign detection will remain essentially as time consuming as

division by a divisor of the moduli of the system.

Since, the division algorithms used in class C also use these "division by a constant" procedures any improvements in sign determination techniques would yield a corresponding improvement in techniques of type C also. With regard to algorithms of type B, mixed base system division seems to be awkward at best. It certainly is more complex than any binary division even without consideration of translation time. This approach appears to be the least valuable of the three, since it has rather extensive hardware requirements without really being very fast.

The magnitude determination free techniques known to us are all iterative in nature and resemble most closely the non-replacement techniques of conventional binary division. With respect to these algorithms, several important questions must be asked.

1. What is the rate of convergence?
2. What are the hardware requirements?
3. Are any initial estimates needed, and if so, how are they determined?
4. What is the stopping procedure?
5. What type of arithmetic systems can this algorithm easily be fit into?

7.2 AN APPROXIMATE DIVISOR METHOD

This algorithm utilizes the fact that inverse multiplication corresponds to ring division if the remainder is zero.

Let:

$$M = \prod_{i=1}^{m} p_i \qquad (i = 1,2...m)$$

be the modulus of a residue number system. Let Z be the set of 2^m numbers less than M of the form

$$\prod_{i=1}^{m} p_i^{b_i} \qquad b_i \in \{0,1\}.$$

It is well known that division of any number in the ring of integers, modulo M by $z \epsilon Z$ may be performed in a straightforward way by successively reducing the dividend by an amount necessary to make it exactly divisible by one of the component primes of the divisor, and multiplying by the reciprocal of that prime. In general, the division of x by z will take either 2n or n steps depending on whether one counts substraction and multiplication as 1 or 2 steps. If m primes compose z, then the remainder of the n-m stages are necessary to re-extend the quotient to full length. Using the set Z to approximate y for divisors the following algorithm may now be employed. To divide x/y; $x,y \epsilon R$; $2y > z > y$; $z \epsilon Z$

$$q_o = \left[\frac{x}{Z}\right] \qquad r_o = X - \left[\frac{x}{z}\right]y$$

$$q_i = q_{i-1} + \left[\frac{r_{i-1}}{z}\right] \qquad r_i = r_{i-1} - \left[\frac{r_{i-1}}{z}\right]x$$

stop when $q_i = q_{i-1}$.

An error of 1 will occur if every $y < r_i < z$ which may be corrected by comparison of r_i with y following $q_i = q_{i-1}$ if an exact answer is necessary.

It is not necessary that $z > y$; however, the partial quotients will be of opposite signs on successive iterations if $z < y$.

The rate of convergence depends on z/y. Let $y = (1-\epsilon)z$ and $x/y = q$.

The s_i are the partial sumands $q_i - q_{i-1}$.

$$s_0 = \frac{(1-\epsilon)x}{y} = q(1-\epsilon) \qquad r_0 = x - \frac{(1-\epsilon)x}{y}\, y = x\epsilon$$

$$s_1 = \frac{x\epsilon(1-\epsilon)}{y} = q(1-\epsilon) \qquad r_1 = x\epsilon - \frac{x\epsilon(1-\epsilon)}{y}\, y$$

$$= x[\epsilon - \epsilon(1-\epsilon)] = x\epsilon^2$$

$$s_i = \frac{x\epsilon^i(1-\epsilon)}{y} = q(1-\epsilon)\epsilon^i \qquad r_i = x\epsilon^i - \frac{x\epsilon^i(1-\epsilon)}{y}\, y$$

$$= x[\epsilon^i - (1-\epsilon)\epsilon^i] = x\epsilon^{i+1}$$

The error after i iterations will be

$$q(1-\epsilon)(\epsilon^i + \epsilon^{i+1} + \ldots) = q\epsilon^i$$

for

$$q < 10^a$$

then for

$$q\epsilon^i < 1$$

$$\epsilon^i < 10^{-a} \qquad i \log\epsilon < -a \qquad i > a/\log\epsilon^{-1}$$

expressing ϵ as a function of i and a, then $\epsilon < 10^{-a/i}$.

The convergence of this algorithm is crucially dependent upon how well the set Z approximates arbitrary divisors. In the partical cases where only a few moduli are used it would appear that this algorithm would not be feasible if a fast division were sought. However, in systems including a large number of different moduli the closeness of approximation is remarkable. The system consisting of the primes less than or equal to 31 was considered in detail. The set Z was tabulated and ordered. A set of twenty-five random divisors in the range 10^5 to 10^6 was examined to see how closely they might

be approximated. Assuming these numbers to be divisors of dividends of the order 10^{11} a quotient of less than 10^6 will be produced. The approximating $z \epsilon Z$ for each number was determined along with the necessary number of iterations. For all numbers examined the number of iterations was either 3 or 2. As an example consider the computation of 10,000,000,000/815,392

$$z = 817,190$$

$$q_0 = \frac{10,000,000,000}{817,190} = 12,237 \qquad r_0 = 10^{10} - (12,237)(815,392)$$

$$= 22,048,096$$

$$q_1 = 12,237 + \frac{22,048,096}{817,190} \equiv 12,263 \qquad r_1 = 22,048,096 - (26)(815,392)$$

$$= 847,904$$

$$q_2 = 12,263 + \frac{847,904}{817,190} = 12,264 \qquad r_2 = 847,904 - (1)(815,392)$$

$$q = 12,264 \qquad = 32,512$$

$$r = 32,512$$

Note that for this particular divisor, which was chosen at random, the probability of an error of one in the quotient for a random dividend is less than 1%.

There are three possible means of terminating this procedure. A fixed number of iterations can be used. If indeed it were shown that three is the largest ever needed, such procedure would be the best. For a very small divisor however, this may not be sufficient as the approximations are not as good. The test $r_{i+1} = r_i$ does not involve either additional hardware nor significant additional controls. However, given $r_{i+1} = r_i$, one additional unnecessary iteration has already been performed. Nevertheless, the insurance

that the error in the quotient is at most one, makes it more desirable than the fixed length procedure, unless the particular arithmetic system used scales all numbers into a range where the approximations are always good. The third method involves comparison of r_i with a real divisor and therefore, an additional cycle of n operations following the conditions $r_i = r_{i+1}$. This is the only method for determining an exact quotient. It appears that this additional test would best be left to the programmer just in case an exact quotient is required, since for most uses the error involved is unimportant. The hardware requirements for this algorithm are moderate in addition to the residue multipliers and adders which will already be available. A control register of n bits is necessary to indicate the base primes used in the approximate divisor. The hardware for producing the approximate divisor is an additional problem. It appears that this may most efficiently be implemented by a table-look-up procedure. Indexing this table requires the first two or three digits of the mixed base representation of the divisor and requires the equivalent of an additional iteration. The magnitude of the table required for this system illustrated would be about 2000 eleven bit words, which is not prohibitive.

The remaining two algorithms are essentiality unrelated to the residue number systems itself, but it is of interest to see what difficulties ensue in their realization. Both methods require a fractional multiply operation. It is not yet certain that all possible utilizations of residue arithmetic would entail having a fractional multiply, however, for any conventional numerical analysis it seems likely. Fractional multiplication schemes all in-

volve division by one of the divisors of the moduli. It is fair to compare the number of iterations of the above described algorithm with the number of fractional multiplications of the two algorithms described below. In a system that permits postponement of scaling, it is reasonable to count the number of scalings necessary, rather than the number of multiplications.

7.3 A NEWTON-RAPHSON METHOD

A Newton-Raphson method[8] is derivable from

$$\frac{b}{ax} - 1 = 0 \qquad x \triangleq \frac{b}{a}$$

i.e.,

$$\begin{aligned} x_{i+1} &= x_i - f(x_i)/f^1(x_i) \\ &= x_i - (\frac{b}{ax} - 1)/(-\frac{b}{ax^2}), \qquad \left|\frac{ax_o}{b} - 1\right| < 1 \\ &= x_i(2 - \frac{ax_i}{b}) \end{aligned}$$

This is useless directly since it involves the calculation of a/b. If b = 1 however, then

$$x_{i+1} = x_i \cdot (2 - ax_i), \qquad |ax_o - 1| < 1$$

is quite satisfactory since one may compute 1/a and then multiply by b.

These procedures converge exponentially since if

$$x_i = (1-\epsilon) \cdot x$$

then

$$\begin{aligned} x_{i+1} &= (1-\epsilon) \cdot x[2 - (a/b)(1-\epsilon)x] \\ &= (1-\epsilon^2) \cdot x \end{aligned}$$

so

$$\frac{\epsilon_{i+1}}{\epsilon_i} = \epsilon_i .$$

Given an x_0 such that $\epsilon_0 < 2^{-5}$ then $\epsilon_3 < 2^{-40}$ which is of the order required. This contrasts favorably with the third algorithm

$$x_{i+1} = x_i \cdot (1+b-ax_i), \qquad b < 1$$

for $x_i = (1-\epsilon) \cdot x$.

$$x_{i+1} = x(1-\epsilon) \cdot [1+b-a(1-\epsilon)x]$$

$$= x(1-\epsilon) \cdot (1+b\epsilon) = x[1-\epsilon(1-b)-b\epsilon^2]$$

so

$$\frac{\epsilon_{i+1}}{\epsilon_i} = \frac{\epsilon(1-b)+b\epsilon^2}{\epsilon} = 1 - b + b\epsilon$$

which is very poor if $(1-b)$ is not very close to 0.

It is possible to scale b by one of the divisors of the modulus of the residue system, however this still results in a geometric convergence to better than that provided by the approximate divisor method which does not require fractional multiplication. The scaling would require hardware similar to the obtaining of the approximate divisor as in the first method described.

7.4 INITIAL APPROXIMATIONS

It is possible to use a uniform x_0 in the reciprocation formula. For integral a, then a very small x_0 will always satisfy $|1-ax_0| < 1$. A decent approximation, however, is clearly necessary since if $(1-\epsilon)$ is the required

precision then the number of iteration i must be such that

$$\epsilon_o^{2i} \leqslant \epsilon \; 2^i \geqslant (\log\epsilon)/\log\epsilon_o$$

$$i \geqslant \log_2[\log\epsilon/\log\epsilon_o]$$

$$i \geqslant \log_2\log\frac{1}{\epsilon} - \log_2\log\frac{1}{\epsilon_o}$$

which for ϵ_o close to 1 becomes large.

As an initial approximation, one may use a table-look-up; however, the index to such a table must reflect relative magnitude and therefore, for residue numbers, requires a conversion to mixed base. Still, such an approximation is superior to a linear approximation which would require about the same time but no table. We consider now a system which is mentioned else where in the report which involves the use of a "double length plus" expression of numbers when in the arithmetic unit. The use of this system is convenient in both the initial approximation procedure and in performing the iterations themselves.

If the size of the system is such that the double length magnitude M^* is of the order LM^2 where M is the single length magnitude then in approximately the same time, one can compute a quadratic approximation where the coefficient of the second order term is a small integer less than L. This is possible since where the fraction x is represented as [xM], the first coefficient is represented directly [a] while the second and third as bM and cM^2, so that

$$ax^2 + bx + c$$

is represented before scaling

$$[a][xM]^2 + [bM][xM] + [cM^2]$$

$$= [ax^2M^2 + bxM^2] + [cM^2]$$

$$= [(ax^2 + bx)M^2] + [cM^2]$$

which becomes

$$[(ax^2 + bx + c)M^2]$$

Overflow will not occur as long as $[(a+b) < L]$. A comparison of approximation techniques is somewhat difficult because of the measure of goodness of fit. Ideally one wishes to minimize the expected number of iterations which is

$$c + \int_0^1 [- \log \log \frac{1}{\epsilon_0} (a)1]da$$

given a uniform distribution of X. If f(a) is the approximation of 1/a then $\epsilon_0 = 1\text{-}af(a)$ and only a numerical solution appears feasible. A simple least squares measure should, however, give a rough comparison.

The best-fitting quadratic using a least squares criterion results from the solution of the system of linear partial differential equations:

$$\frac{\partial\phi}{\partial a} = \frac{\partial\phi}{\partial b} = \frac{\partial\phi}{\partial c} = 0$$

$$\phi = \int_0^1 [1 - x(ax^2 + bx + c)]^2 dx.$$

These equations easily reduce to three linear equations in the coefficients. The solution yields a nonintegral value for a, however the linearity of the system insures a unique local maximum, and the best integer values for a may be determined by inspecting the two nearest integers.

The hardware requirements in addition to those already needed for the fractional multiple are almost negligible. The only additional equipment is

that associated with determining the initial approximation. Depending on the type of instructional organization used in the machine, it may be possible to do away with the division instruction per se, altogether. The condition under which this would be possible without loss of speed would essentially involve a streaming mode of operation, similiar to that used in computing the vector dot product.

Before concluding this chapter, it is important to emphasize again the impossibility of deciding on a best division algorithm without considering the system on a whole. In our opinion, the Newton-Raphson Reciprocation procedure is clearly superior to any other known if a fast fractional multiplication has already been purchased. Its exponential rather than geometrical convergence plus its lower hardware requirements make it quite satisfactory. The other algorithms might have use in certain limited special purpose applications where a full fractional multiplication is not desired.

CHAPTER VIII

THE MULTIPLICATIVE OVERFLOW PROBLEM

8.1 INTRODUCTION

The problem of multiplicative overflow detection is one that is unique to systems in which the multiplication operation is accomplished in one step. In conventional systems, those in which multiplication is performed as a series of additions, the detection of multiplicative overflow can be handled by the detection of additive overflow.

One of the main advantages of a Residue Number System (RNS) is its adaptability to a one-step multiplication operation. It follows that one of the more pressing problems of a RNS is that of detection of multiplicative overflow.

In any finite number system, an obvious way to prevent overflow under multiplication is to require that every number to be processed be less than the square-root of the system's limit. The product of any two such numbers will be less than the limit of the system, and no multiplicative overflow will exist. But this restriction must apply to every step of each calculation, and merely represents the substitution of a scaling problem for the one of overflow detection.

A universal problem of finite machines, whether their organization is integer or fractional, is the one of scaling the double-length result of multiplication back into single-length form. This problem has a solution

that takes the form of multiplication by a constant. In a RNS of single-length capacity m, where the residue representations are interpreted as integers, that constant is the inverse of m.

In his work on RNS, Svoboda[2] suggests handling this scaling problem by employing a double-length multiplier whose capacity is at least as great as the square of the capacity of the single-length system. He then employs a procedure which is essentially division by m (the single-length capacity) to rescale the result of multiplication into suitable form for further calculations. The unfortunate feature of this method is the amount of time required for the digit-fill-in operations (single-to-double-length and vice versa) used to implement the scaling process.

Cheney[9] has suggested a method which involves scaling by combinations of the individual base primes. The precision achieved by this method is not as good as Svoboda s, but the comparison is somewhat unfair since Cheney does not use the double-length idea. The question of precision will be treated more fully in the last section of this chapter.

The individual-base scaling procedure is awkward and cumbersome. As a result it is time-consuming. Even if precision were equal for the two procedures, this one suffers by speed-comparison.

Consider now an extension of Svoboda's concept to include what will be termed multiple-precision. This title may be misleading, as the direct result is a gain in speed while precision is increased non-uniformly. That is, the increase is slight (or non-existent) in some cases, while present to a greater degree in others.

The proposed extension is accomplished by attaching positional significance to each of a group of RNS representations (as in, for example, a decimal number) and allowing the propagation of carries between subgroups of these representations.

Such an arrangement, with the operations of truncation and scale-factor multiplication, constitutes a Floating-Point number system. The arithmetic of this system is apparently retarded by the necessity for carries, but carry assimilation is accomplished as a part of the scaling operation.

The speed advantage of this system comes from the fact that the subgroups of RNS representations (the coefficients) are small. This permits the standard RNS operations to be accomplished rapidly. The digit-fill-in and scaling operations are performed on the entire number by processing several small parts of its simultaneously. A complete description of these operations is continued in the following sections.

The two parts of this chapter which follow are both methods of employing this parallel structure. The system contained in Sections 8.2 through 8.8 is primarily a general purpose structure, while that of the remaining sections of the chapter is intended for special purpose use. That is, the one considers everything that might occur, while the other ignores or minimizes certain problems under the assumption that they will occur infrequently, if at all.

The original starting point for the two developments was different. The former is a result of an inquiry into the general problem of multiplicative overflow. The latter came about as a use for the redundancy established by

an extra-length representation.

There are two main areas of dissimilarity between the two methods. The first is the question of whether the sign of a number should be carried along explicitly in the representation, or be allowed to remain implicit within the representation. The second is a question of the size of the double-length system; whether its capacity should be the square of the capacity of the single-length system, or slightly larger than the square. Rather closely connected to these two areas is the question of when and/or how often the internal scaling procedure must occur.

Perhaps each system is superior for its intended use, or even for a series of specified uses. Given a well-specified purpose, the two systems could be carefully compared to determine equality or the superiority of one over the other.

8.2 APPLICATION OF THE SQUARE-ROOT CRITERION

This direction of attack upon the overflow problem branches off from the square-root criterion.

Consider, for the bases of a RNS, some sequence, β, of composite numbers

$$\beta = \beta_1, \ldots \beta_n$$

subject to two restrictions:

$$(\beta_i, \beta_j) = 1, \qquad i \neq j \tag{8-1}$$

$$\beta_i = p_i^2 \tag{8-2}$$

The first restriction is equivalent to, and may be replaced by, the following restriction on the set, P, of sub-bases.

$$(p_i,p_j) = 1, \qquad i \neq j \tag{8-3}$$

The set, P, may be thought of as a sequence of prime numbers with no resulting loss of generality.

Consider also, in the matter of coding the modular representations of the RNS with respect to the base β_i, the use of a two-digit uniformly-based number with base p_i. Such a coding does indeed accomplish the representation, although it requires one carry-digit within each composite-base representation.

The following formulas serve as a summary of the discussion of this section:

$$N \equiv \alpha_i \bmod \beta_i, \qquad 0 \leqslant N < M \tag{8-4}$$

$$M = \prod_{i=1}^{n} \beta_i, \qquad \beta_i = p_i^2 \tag{8-5}$$

$$\alpha_i = a_{i1}p_i + a_{io} \tag{8-6}$$

It follows that the zero power, or units, digit (a_{io}) represents the residue of N with respect to p_i. Thus there exists a readily available n-digit number which represents the residue of N with respect to m, where

$$m = \sqrt{M} = \prod_{i=1}^{n} p_i \tag{8-7}$$

These n digits are the units digits (a_{io}) of the n representations of N modulo $\beta_i (i = 1,2,\ldots,n)$.

8.3 CONSIDERATIONS OF THE DOUBLE-LENGTH SYSTEM

If a number, $A(0 \leqslant A < M)$, is considered to be of the form

$$A = a_1 m + a_0 \tag{8-8}$$

it is clear that a_0 may be found by some reduction process involving the single-length system m: if you will, the "square-root" system m. That is, since $0 \leqslant a_0 < m$ and the n units-digits of $A(a_{i0}; i = 1,2,\ldots,n)$ are available by inspection, the search to find a_0 may be limited to the m-system. Digit-fill-in of a_0 from the m-system into the M-system acquires the first-power digits of a_0, and subtraction of this result from the original representation of A leaves $a_1 m$ as the new representation.

Because $a_1 m$ is an integer multiple of m, the units-digits of this representation are all zeros. For this reason, consideration may be limited once more to an m-system representation, in this case involving the first-power digits. Division of this representation by the first-power digits of the representation of m, along with m-system to M-system digit-fill-in, yields a_1.

The above paragraphs describe a complete reduction of the M-system representation of a number to more conventional form by means of m-system operations. With the background established so far, investigation of some properties of this "square" system may be undertaken.

The redundancy established by this particular system, the easy availability of the square-root residue, is gained only at the expense of providing a carry-digit within each composite-moduli group. But this system is useful in the scaling aspect of the problem. The redundancy provides the

necessary magnitude information for scaling, which makes division by m a simple operation.

8.4 MULTIPLICATION

Consider the multiplication of two numbers A and B ($0 \leqslant A, B < M$) in algebraic form.

$$A = a_1 m + a_0, \qquad B = b_1 m + b_0 \tag{8-9}$$

$$AB = a_1 b_1 m^2 + (a_0 b_1 + a_1 b_0) m + a_0 b_0 \tag{8-10}$$

Each of the ordered pairs $(a_i b_j)$ represents a non-overflow situation since, by the nature of the system, every $a_i, b_j < m$.

The interesting part of this is not in the possibilities for detection of overflow, but in other possibilities suggested by the ordered pairs. These numbers are all readily available in the system, and the products are guaranteed non-overflow. Through some system of manipulations they may be carried along to give a complete description of any number, though it be arbitrarily larger than the finite limit of the sub-system.

Such factors as truncation and round-off may be considered if an approximate answer is acceptable. For example, it may be considered necessary to keep only the four most significant coefficients, any remaining terms then being discarded.

Now consider a more general multiplication, AB, with A and B not restricted in magnitude as they were previously. Let R be defined as the reduction operation of Section 8.3, and understand R_1 to be the first (a_0) portion of this operation, while R_2 shall be the remaining (a_1) portion.

Let

$$A = \sum_{i=0}^{r} a_i m^i \tag{8-11}$$

$$B = \sum_{j=0}^{s} b_j m^j \tag{8-12}$$

$$AB = C = \sum_{d=0}^{r+s+1} c_d m^d \tag{8-13}$$

where

$$0 \leqslant a_i, b_j, c_d < m. \tag{8-14}$$

The multiplication AB results in several ordered pairs, $a_i b_j$, and by the restriction of equation (8-14)

$$0 \leqslant a_i b_j < M \tag{8-15}$$

so that the operation R, when applied to these products, gives

$$a_i b_j \xrightarrow{R} c_{ij1} m + c_{ijo} \tag{8-16}$$

in which

$$0 \leqslant c_{ij1} < m-1; \qquad 0 \leqslant c_{ijo} < m. \tag{8-17}$$

Define, for the collected term of multiplication:

$$c_k = \sum_{i+j+t=k} c_{ijt} \tag{8-18}$$

Since the number of terms collected is very small compared to m,

$$0 \leqslant c_k \ll m^2 \tag{8-19}$$

Now, under a further reduction step:

$$c_k \xrightarrow{R} c_{k1}m + c_{ko}. \tag{8-20}$$

The restrictions upon these coefficients follow from equation (8-19)

$$0 < c_{k1} \ll m; \qquad 0 < c_{ko} < m. \tag{8-21}$$

At this point, AB = C may be expressed as follows:

$$C = \sum_{k=0}^{r+s} (c_{k1}m+c_{ko})m^k. \tag{8-22}$$

Since the largest number expressible in this form is given when

$$A = m^{r+1} - 1 \tag{8-23}$$

$$B = m^{s+1} - 1 \tag{8-24}$$

$$AB = m^{r+s+2} - m^{r+1} - m^{s+1} + 1 \tag{8-25}$$

it follows that

$$C = AB < m^{r+s+2} \tag{8-26}$$

and

$$c_{r+s+2} = C. \tag{8-27}$$

Thus, finally, the form of equation (8-13) is obtained

$$C = \sum_{d=0}^{r+s+1} c_d m^d.$$

Since the c_d's must satisfy the restriction of equation (8-14), they must result from a process of reduction upon the c_{kt}'s and completion of the carry

(or scaling) process. An exception is c_{r+s+1} which will never propagate a carry, as a result of equation (8-27).

An estimate of truncation error would be in order at this point. Let the decision be made to store only the four most significant coefficients of any number, the highest-order of which is non-zero. Thus if

$$A = a_r m^r + a_{r-1}m^{r-1} + a_{r-2}m^{r-2} + a_{r-3}m^{r-3} + a_{r-4}m^{r-4} + \ldots \quad (8\text{-}28)$$

$$B = b_s m^s + b_{s-1}m^{s-1} + b_{s-2}m^{s-2} + b_{s-3}m^{s-3} + b_{s-4}m^{s-4} + \ldots \quad (8\text{-}29)$$

certainly

$$AB < a_r b_s m^{r+s} \quad (8\text{-}30)$$

while the highest-order neglected term is

$$(a_{r-4}b_s + a_r b_{s-4})m^{r+s-4}. \quad (8\text{-}31)$$

Therefore the error is given approximately by

$$\frac{a_r b_{s-4} + a_{r-4}b_s}{a_r b_s m^4} \quad (8\text{-}32)$$

In the worst case, with a_r, $b_s \simeq 1$ and a_{r-4}, $b_{s-4} \simeq m$, a rough upper bound on the error as obtained from equation (8-32) is $2/m^3$. While if a_r, $b_s \simeq m$ and a_{r-4}, $b_{s-4} \simeq 1$, a bound on the error is $2/m^5$. In any case m need not be too large to result in quite a few significant demical digits with such an arrangement. For example, if the set P of sub-bases is (2, 3, 5, 7), m = 210 and $2/m^3 \simeq 2.16 \times 10^{-7}$. And if P is (3, 5, 7, 11), m = 1155, $2/m^3 \simeq 1.3 \times 10^{-9}$. This demonstrates the feasibility of a "small" RNS.

Figure 2 is a schematic description of multiplication in the proposed, floating-point RNS. The justapositions (a_ib_j) all indicate non-overflow RNS multiplications, the R operations are as defined earlier in this section, and the additions indicated are also non-overflow RNS operations.

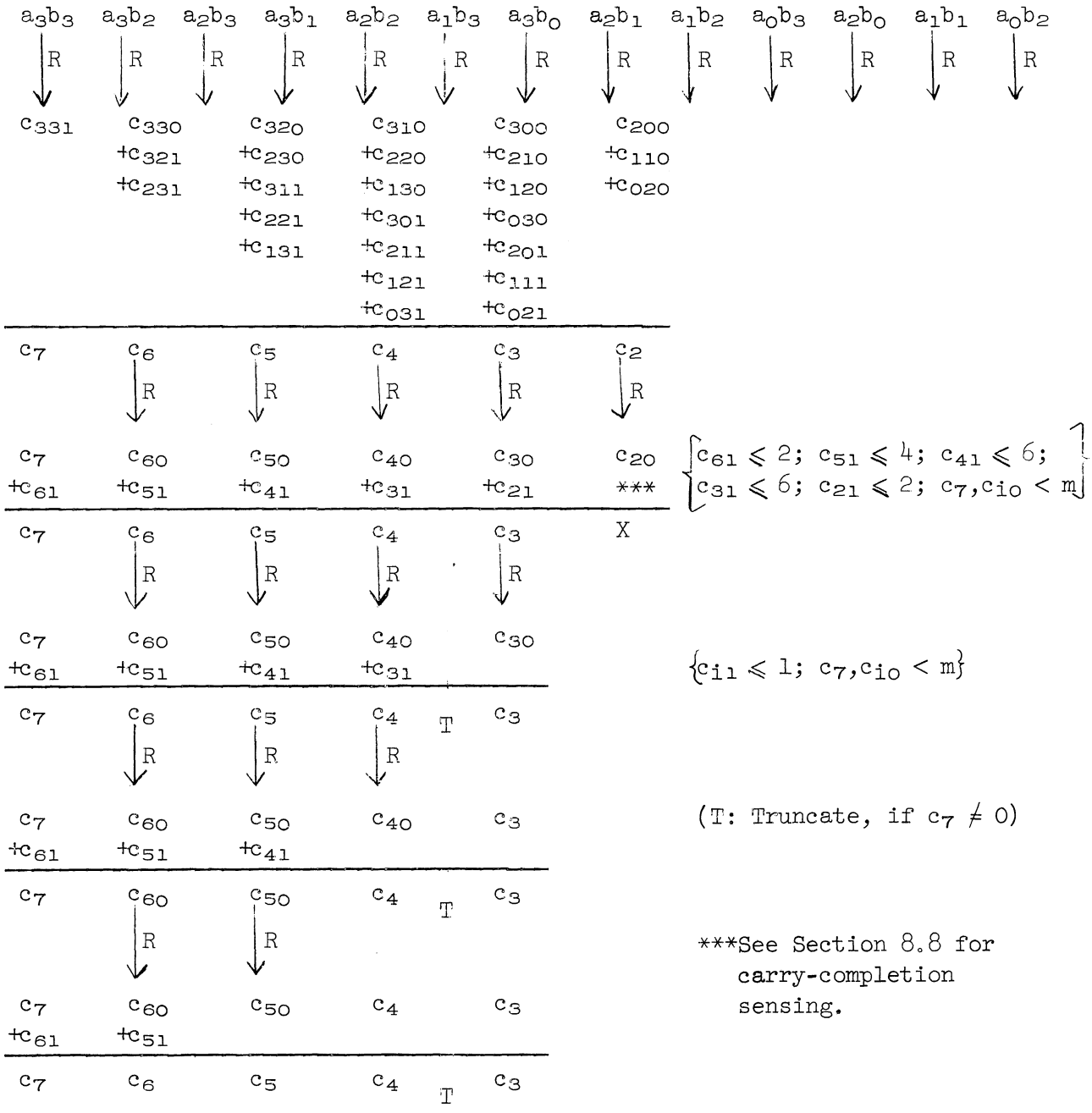


Figure 2. Multiplication in the floating point RNS.

The first level shown is actually the only multiplication involved, the remainder of the operation can be regarded as scaling to get the respective digits back into the $0 \leqslant c_i < m$ range. This scaling can take place quite separately from the multiplication, leaving the thirteen side-by-side M-system multiplication units free immediately for a new operation. The results are not immediately available for further computation, however, but must first be processed through the reduction "grinder."

The reduction machine will consist of various levels and may operate as a unit or as several semi-independent stages. Some of the stages may prove useful for operations on added numbers, etc. The "grinder" may also be processing a sequence of numbers at the same time in different stages, with a continuous flow of results from the output.

In Figure 2, A and B are taken to be of the following form:

$$A = m^r(a_3m^3 + a_2m^2 + a_1m + a_0) \tag{8-33}$$

$$B = m^s(b_3m^3 + b_2m^2 + b_1m + b_0) \tag{8-34}$$

Manipulation of the "exponent" (m^{r+s}) is not shown, but understood.

At the first level of reduction, thirteen side-by-side R-type units are necessary. These are followed by five adders, followed in turn by five R-type units. Next come five adders, followed by four R-type units. As was seen in equation (8-27), further attention to c_7 is unnecessary as it cannot overflow.

At this level, four adders are necessary and truncation can occur by dropping c_3. In the worst case three more levels are necessary, with R fol-

lowed by adder. The number of elements needed would decrease with each stage. It is possible that carry-completion sensing would be both possible and profitable. That subject will be taken up in a later section.

8.5 NUMBER FORMAT AND MAGNITUDE COMPARISON

A description of number format is now presented in order to lay the groundwork for a discussion of some other operations in the system. The system-description of a number, A, shall consist of a magnitude-plus-sign arrangement. The magnitude is to be represented by four coefficients with positional significance, and an exponent term. S(A) is defined to be the sign of A, and may be represented by one bit.

$$S(A) \quad a_3 \quad a_2 \quad a_1 \quad a_0 \quad r \tag{8-35}$$

where

$$0 \leqslant a_1 < m,$$

while

$$r = 0, \pm 1, \pm 2, \ldots \pm q \tag{8-36}$$

Equation (8-35) is then defined to mean

$$A = S(A)(a_3 m^3 + a_2 m^2 + a_1 m + a_0) m^r \tag{8-37}$$

The magnitude comparison operation may now be accomplished by a three-step procedure. If A and B are represented as suggested in equations (8-35, 36, 37), then:

1. Inspect the signs;
   a. If they are different, the positive one is larger.
   b. If they are the same, proceed to step 2.

2. Inspect the exponents;

   a. If they are different, the more-positive one corresponds to the larger number.
   b. If they are the same, proceed to step 3.

3. Complement $b_i (i = 0, 1, 2, 3)$ with respect to M, add $A+\overline{B} = C$. Operate on c_i with R_1. Inspect $c_{31}m$.

   a. If $c_{31}m \neq 0$, $B > A$.
   b. If $c_{31}m = 0$, inspect c_{30}.

      1. If $c_{30} = 0$, $A > B$.
      2. If $c_{30} = 0$, inspect $c_{21}m$.

The complete operation R is not needed in step 3, because the question of magnitude is determined by whether or not $c_{i1}m$ is zero. This information is available at the output of R_1.

Figure 3 is a graphical aid for step 3 above. The X under sign and exponent indicates that they are of no interest in this step.

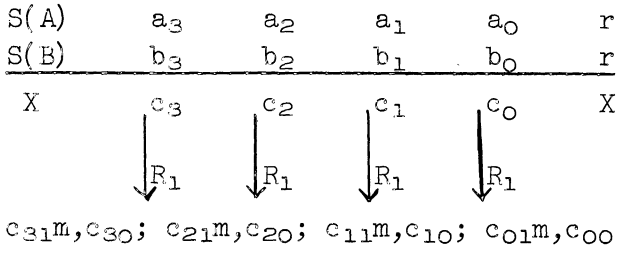


Figure 3. Step 3 of the magnitude comparison.

8.6 THE FOUR CASES OF ADDITION

For the addition of two numbers, there are four cases of interest. Inspection of the signs and exponents of A and B determines the case in any particular instance. For convenience and simplicity, A is assumed always positive, with more-positive relative exponent if the exponents are different. The four cases are:

1. Same exponent, same sign;
2. Same exponent, different signs;
3. Different exponents, same sign;
4. Different exponents, different signs.

Addition in case 1 is accomplished by positional addition of corresponding coefficients. The sign of the result is the same as the signs of the two numbers. The exponent must be determined in the scaling process. Truncation (T), the right-shift of coefficients, and augmentation of the exponent as shown in the final stage of Figure 4, take place only if $c_{31} \neq 0$. No further carries are possible out of c_3 or c_{-1}, but in the worst possible case, three more sequential (R, Add) operations will be necessary to complete the carry process. It can be seen that only one carry (and of unit magnitude) can be transmitted from any position. That is, a position that generates a carry cannot propagate one. Thus, carry-completion sensing is suggested.

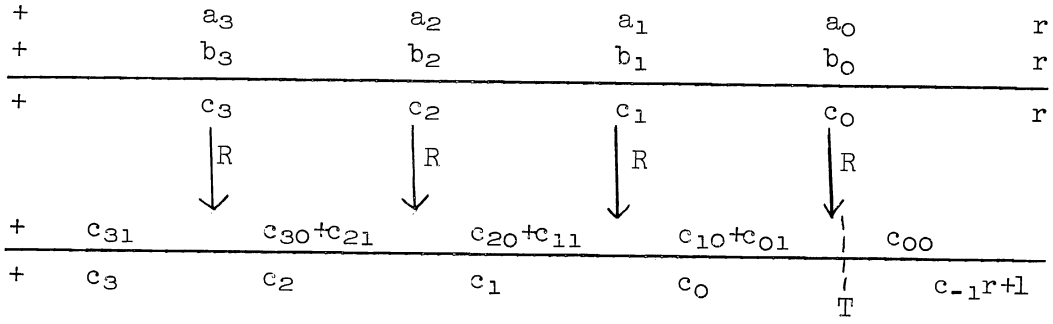


Figure 4. Addition in the floating point RNS.

Addition in case 2 is accomplished by, first, step three of the magnitude comparison operation. The next step is complementation of the smaller number, and then positional addition. This is followed by the R_1 operation as in Figure 3. Then a check must be made on $c_{i1}(i = 0,1,2)$. If $c_{i1} = 0$, the process is complete. If $c_{i1} \neq 0$:

1. Add m to c_i;
2. Add M-1 to c_{i+1}.

This reduction process must be carried out three times in the worst possible case. However, a borrow can be propagated through the ith position only if $c_{i1} = c_{io} = 0$, which suggests a possible borrow-completion sensing method. The sign of C is that of the larger number, A or B. The determination of r must, upon borrow completion, be determined by the highest-order non-zero c_i, and a shift accomplished to put that c_i into the c_3 position.

Addition in case 3 involves a relative shift to the right, of (r-s) positions, for the smaller number. At most four reduction processes are necessary to complete carry-propagation. Sign is carried along unchanged, while truncation and r-augmentation are accomplished as the last step.

Addition in case 4 combines the right-shift of case 3 and the complementation of b_i with respect to M as in case 2. Once again, borrow-completion requires at most three reductions (R_1). The resulting sign is that of the larger number, while positional-shift and r-determination are accomplished as the final step.

8.7 DIVISION

The division procedure in this floating-point system must be some combination of the familiar and the unfamiliar. The familiar portion is due to the similarity of the gross system-structure to the decimal system. The unfamiliar portion is due to the modular arithmetic of the sub-structure.

One possible division procedure is graphically displayed in the flow-diagram of Figure 5. Some prior attention must be given to the operations

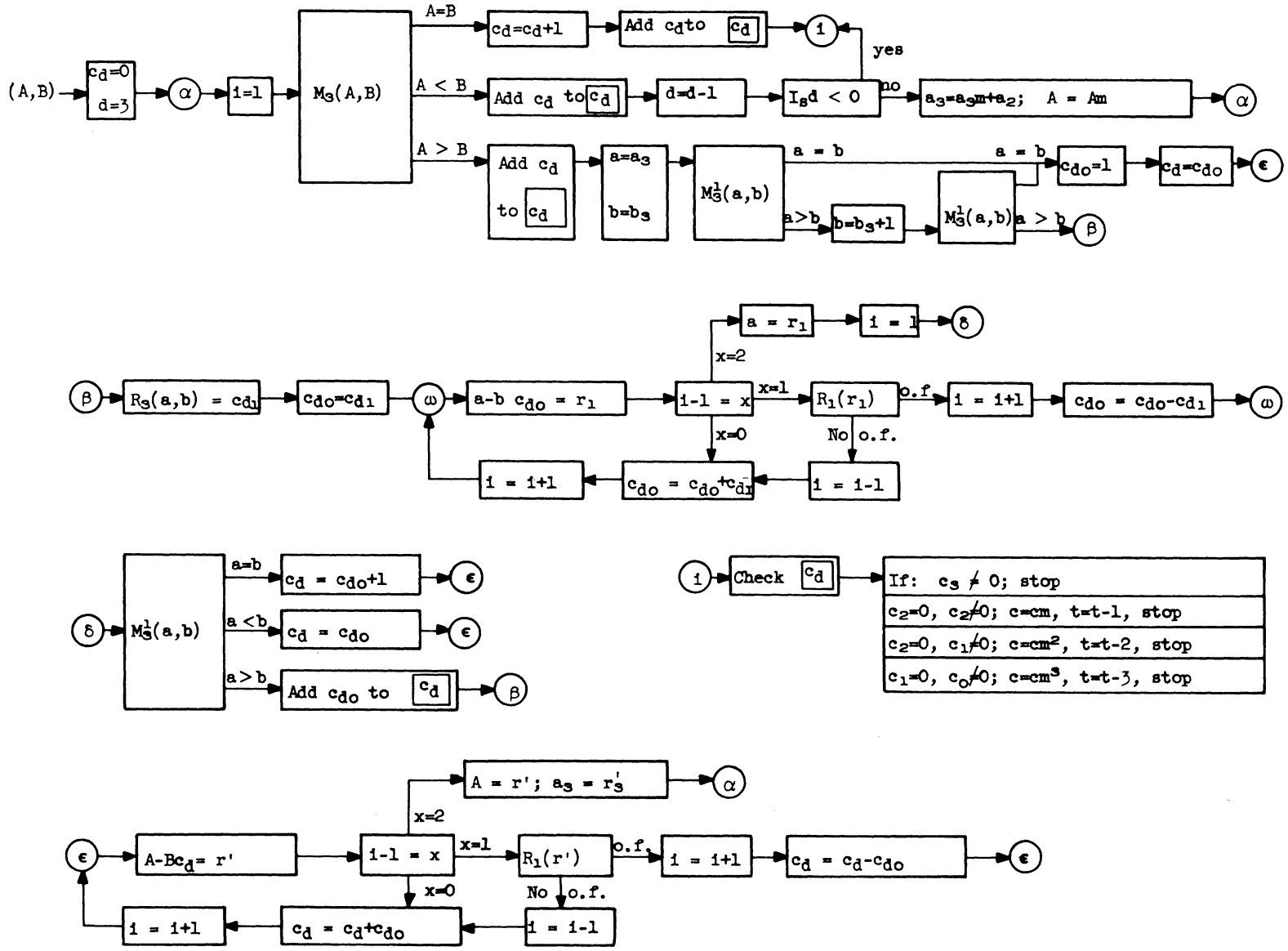


Figure 5. Division in the floating point RNS.

involved, and the intent of the over-all operation explained, in order that the flow-diagram may be more easily interpreted.

Given two numbers, A and B, in the form of equation (8-35), the sign of A/B = C will be determined, as in multiplication:

If

$$S(A) = S(B), \qquad S(C) = + \tag{8-38}$$

$$S(A) \neq S(B), \qquad S(C) = - \tag{8-39}$$

While the initial determination of the exponent (t) of C will be accomplished as

$$t = r\text{-}s\text{-}3 \tag{8-40}$$

and may or may not require modification in the final step of the division procedure. This the number, C, will be represented by

$$S(C) \quad \boxed{c_3} \quad \boxed{c_2} \quad \boxed{c_1} \quad \boxed{c_0} \quad t \tag{8-41}$$

where the box around $c_d (d = 0, 1, 2, 3)$, here and in Figure 5, implies that the number is in the storage location reserved for that particular final result.

For the calculation as shown in the flow-diagram, the signs of the numbers are ignored and the exponents are treated as though they were zero (excepting the final step). In other words, only the stems of the numbers (their coefficients) are treated in the calculation.

The operation $M_3(A,B)$ is the step-three magnitude comparison operation of Section 8.5. The operation $M_3^1(a,b)$ is a reduced version of the same thing,

operating on a pair of single coefficients rather than on two complete numbers. The A = Am operation is a single left-shift of the dividend, while the operation $a_3 = a_3 m + a_2$ is a sub-system computation to obtain the new coefficient a_3. The operation, Add c_d to $\boxed{c_d}$, is understood to leave $c_d = 0$ upon completion. $R_1(x)$ is the reduction operation of Section 8.3, but is used only to detect overflow of the m-system in this procedure.

The multiplications used in this procedure are not the full-scale multiplications of Section 8.4. The multiplication in the ω sequence involves no carries at all and is a single-step operation involving two m-system coefficients. The ϵ sequence multiplication does involve carries, but it is the multiplication of a number, B, by a single coefficient, c_d, and might be thought of as a "scalar multiplication" as opposed to the complete "vector multiplication" described in Section 8.4.

The $R_3(a,b)$ operation is one for the selection of a reasonable first approximation for the quotient digit. It could be done by a table-look-up procedure; since the m-system may be relatively small, such a table could also be of reasonable size. However, it can be done by a simultaneous-reduction process, the reduction being to a non-uniformly-based number system. If a_3 is of the form $(0 \leqslant a_3 < m)$

$$a_3 \equiv \alpha_1 \ \alpha_2 \ \alpha_3 \ \alpha_4 \tag{8-42}$$

corresponding to the definition of equation (8-4), then the process

$$\frac{\dfrac{a_3 - a_4}{p_4} - \alpha_3'}{p_3} - \alpha_2''}{p_2} - \alpha_1''' \tag{8-43}$$

an operation in the m-system, must go to zero at one of the steps. If the pair of coefficients (a_i,b_j; $a_i > b_j$) are reduced simultaneously, when the b-sequence goes to zero the remainder of the a-sequence may be re-expanded into the M-system and used as the desired approximation. If both sequences go to zero at the same time, unity may be taken as the approximation.

The (β) and (ε) sequences of Figure 5 are used to find and improve a series of quotient-digit approximations. The first approximation in each case is chosen in such a way as to be smaller than (if not equal to) the correct one. The loops increase the approximations by consecutive-integer-multiples until overflow results, thereby bracketing the correct result. Restoration is used, upon overflow, and a new approximation is then found from the remainder term. The remainders are always compared with the divisor for relative magnitude to determine the subsequent operation. When the remainder is smaller, the correct quotient-digit is the current sum in $\boxed{cd}$.

Note that the $M_3^1(a,b)$ operations in the A > B portion of the (α) sequence show only two alternatives, because the third one (a < b) does not exist as a possibility at that point.

The convergence of this division scheme is roughly geometric. A Newton-Raphson scheme converges nearly exponentially and would thus be more desirable. This method has been presented in the spirit that these straightforward algorithms serve to demonstrate that the system is feasible with rather simple operations.

8.8 CARRY-COMPLETION SENSING

There are several tools that might be of use in the carry-completion prob-

lem. For example, borrow-completion, in addition cases 2 and 4, may be sensed by testing every c_{i1} and c_{io} after the application of R_1 (as in Figure 3). If, for all i; c_{i1}, $c_{io} \neq 0$ then no further steps are necessary.

Carry completion, in addition cases 1 and 3, is aided by the fact that carries cannot be both generated and propagated by the ith position. Obviously, if $c_{i1} = 0$ or 1 (for all i) the process is completed. Moreover, if any $c_{i1} = 1$ (it cannot be greater) then it need not be checked further. Figure 6 is a chart specifying the situation in detail, following the additional rule that if $c_{i1} = 0$, $c_{(i-1)} \neq 0$ a further step is necessary.

If i equals	3	2	1	0	N	
And c_i equals	0	0	0	0	0	Then N corresponds to the number of further possible steps.
	0	0	0	1	3	
	0	0	1	0	2	
	0	0	1	1	2	
	0	1	0	0	1	
	0	1	0	1	1	
	0	1	1	0	1	
	0	1	1	1	1	
	1	0	0	0	0	
	1	0	0	1	2	
	1	0	1	0	1	
	1	0	1	1	1	
	1	1	0	0	0	
	1	1	0	1	1	
	1	1	1	0	0	
	1	1	1	1	0	

Figure 6. Carry completion.

The scaling operation after multiplication is not as slow as it looks in Figure 2. It may be seen that carries occur only if $m-6 \leqslant c_{io} < m$ (in the worst case) at the starred level. Under the assumption of a uniform distribution of numbers in the range $0 \leqslant c_{io} < m$, and taking m to be 1155, this

represents less than a 1 percent probability. This may be detected by the criterion that $c_{i1} = 0$ (for all i) represents completion of the scaling process. Moreover, if carries do exist at that level, at the next level Figure 6 is directly applicable to the completion problem, under a change of the subscripts given by $i = i+3$.

8.9 POSTPONED SCALING AND SPECIAL PURPOSE MACHINES

The most striking characteristic of RNS arithmetic is the substitution of the magnitude-control problem for the one of pure integer multiplication. The first approach discussed in this chapter demonstrates how RNS arithmetic may be used to construct a conventionally-organized machine. This means that the machine instruction code employed might be indistinguishable from a conventional machine instruction code. The difficulties in the RNS, however, might well demand a machine organization of a different type. The direction of exploration here is based on the maximum postponement of the time-consuming magnitude-control operation.

The primary characteristic of a system using such an organization is an arithmetic unit capable of operating on longer-than-double-length numbers. This feature permits addition and subtraction operations on a set of products before scaling. Postponement of scaling necessitates "remembering," between operations, which numbers have been properly scaled and which have not. The best method of handling this difficulty depends on whether a general-purpose or special-purpose use is intended.

General-purpose organization will require special scaling instructions, or perhaps tag bits in the ordinary instructions, indicating whether or not

a result is to be scaled. The difficulties for the programmer, in keeping track of magnitudes, should be overcome by the use of compilers. Such compilers initially would need information as to the possible range of all input variables; they would keep track of each variable, inserting scaling instructions where necessary.

The postponement of scaling requires an implicit sign scheme, since sign-determination is as difficult, in general, as scaling itself. An ordinary complement-system appears applicable to a multiple-precision RNS, with one slight difficulty. The base-extension procedure must differentiate between negative and positive numbers. The former must then be placed in the upper half of the double-length system.

There are certain numerical-analysis procedures which can take quite natural advantage of scaling-postponement. The computation of the dot-product of two vectors is typical of these procedures. This operation occurs in relaxation methods for differential equations, in the formation of a weighted sum of neighbors. This procedure is basic to several matrix inversion algorithms.

The design of a special-purpose machine for matrix inversion was considered in some detail during the past year. That portion of this study having to do with organization is included here. The suggested configuration appears to be considerably faster, at no great cost than conventionally-organized machines. The "streaming mode" of operation, with its resulting savings in instruction fetch, is of particular interest, apart from the use of the RNS. In some respects this organization resembles the "Harvest" device of IBM.[10]

8.10 A MATRIX-INVERSION COMPUTER

The matrix-inversion problem involves little input-output data, but a great deal of arithmetic computation. The proposed computer is capable of handling up to 100 by 100 matrices.

In any residue machine, the digits of the residue number may be coded in binary form. Only prospective moduli which are equal to (or slightly less than) powers of 2 can be coded efficiently in binary form. If a sufficient number of digits are carried throughout the computation (in this case the equivalent of 16 decimal digits will be carried) to insure the desired accuracy, either the number of residue digits required, or the size of the individual moduli, becomes excessively large. The time required for magnitude and/or sign determination is almost directly proportional to the number of residue digits.

A multiple-precision system was selected as the best way to reduce the number of residue digits and to insure an efficiently coded number for storage. Sufficient accuracy can be obtained, using multiple-precision, to carry the equivalent of 16 decimal digits throughout the computation. The basic machine number will consist of four base "b" digits, where "b" is 127 x 128. A scaling factor, α, will also be part of the number. Each base "b" digit will be expressed in a two-digit residue code, the two digits of which are 127 and 128. These residue digits are then binary-coded, using 7 bits for the 127-digit and the same number for the 128-digit. A 64-bit storage word allows an additional 8 bits for exponent.

The problem of matrix inversion can be broken down into a series of

matrix multiplications. Matrix multiplication can be broken down into a series of "sum of products" operations:

$$c_{ij} = \sum_{k-1}^{n} a_{ik} b_{kj}.$$

If the arithmetic unit can be organized to compute this "sum of products" operation efficiently, it will be ideally suited for the matrix inversion problem.

In a conventional machine, a double-length accumulator is provided to insure that multiplicative overflow does not occur. The accumulator is made somewhat larger than double-length so that it is possible to compute the sum of several products without overflow. To this end each base "b" digit (expressed in a 127, 128 residue code) will be extended to a 127, 128, 63, 61, 59, and 31 code as soon as it is obtained from storage. These numbers are pairwise prime. Furthermore, their product is greater than 400 $(127 \times 128)^2$. The extended-base representation of each base "b" digit will be carried throughout the arithmetic unit. A block diagram of the proposed arithmetic unit is shown in Figure 7.

The machine will be basically a two-address machine. The general storage is to be word organized, and divided into X and Y halves. The cycles of the two halves can be interlaced to reduce the time required to obtain both operands.

The "sum of products" operation will proceed as follows:

1. The two operands X and Y are obtained successively from storage.
2. Each operand passes through the base-extension unit and then to one

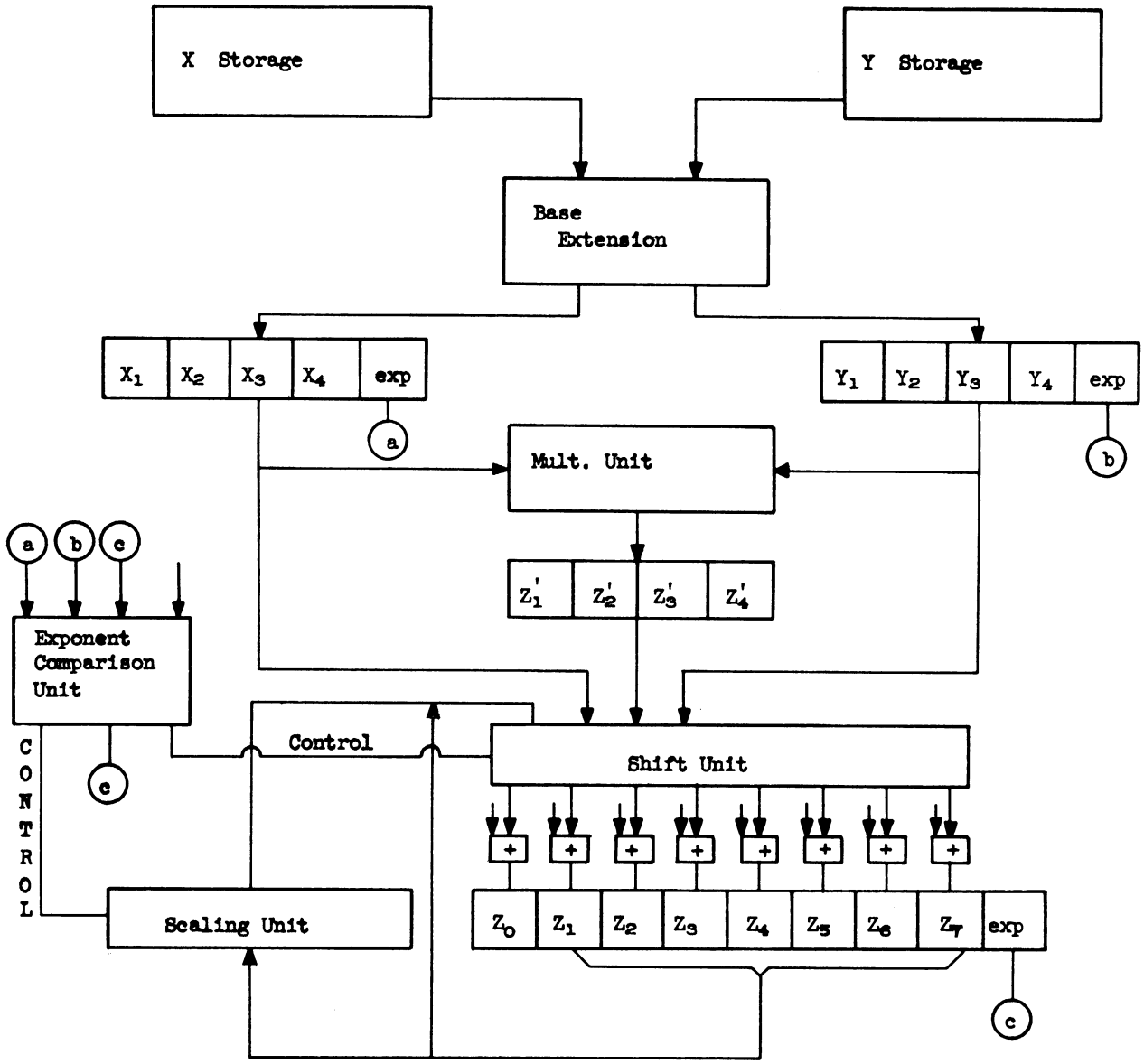


Figure 7. Block diagram of a computer.

of the operand registers.

3. When both operands are present in the operand registers, the actual multiplication can begin. The multiplication operation will involve forming several sets of cross products. As each set of cross products is formed, they are added into the Z registers through the shift unit (SU). To multiply two numbers together, it will be necessary to form four sets of four cross products each. It is evident that in the worst case, after one multiplication, some one of the Z registers will contain a number of order $4b^2$.

4. The SU is controlled by the Exponent-Comparing unit (ECU). The ECU adds the exponents of the X and Y operands and compares the results with the exponent already present in the Z register.

a. If the exponent already present in the Z register is greater than or equal to that of the new product, the cross products are added through SU into the appropriate Z registers.

b. If the exponent of the new product is greater than that already present in the Z register, the contents of the Z register are shifted right enough positions so that the most-significant digit of the product will add into the Z_1 position.

5. This process is continued until a maximum of 100 of these products have been summed.

At the end of the "sum of products" operation, each of the Z registers will in general contain a number of order $400\ b^2$. But since the capacity of the register exceeds $400\ b^2$, no overflow will have occurred. Before the result can be stored, it must be scaled so that the number in each of the Z

registers is less than b. When this condition is met, the 127, and 128 digits will express the number uniquely and the 63, 61, 59, and 31 digits may be dropped. The scaling operation proceeds as follows: consider the set of digits (Z_0, Z_1, Z_2, Z_3, Z_4, Z_5, Z_6, Z_7).

Z_0 represents an overflow digit and will be zero at the start of the scaling operation. Each of the Z_i digits will be divided by b. The remainder in the nth position will be added to the quotient from the (n-1)th position and the sum placed in the nth register. The resulting number is exactly the same as the original number except that each digit is smaller by a factor of b. The only exception will be the Z_0 digit which will be larger if there was a quotient in the Z_1 position. This process is repeated until all of the quotients are zero, at which point the contents of any digit position will be less than b. Consider the set of digits:

$$Z_1\ Z_2\ Z_3\ \ldots\ Z_7$$

which are of order 400 b^2 where b = 127 x 128. If Z_1 is replaced by:

$$R_i + \frac{Z_i\text{-}1}{b}.$$

Where R_i is the remainder when Z_i is divided by b, and Z_i-1/b is the integer part of the division Z_i-1/b. The procedure for scaling should proceed as follows:

1. Decode each Z digit
2. Divide by 127
3. Divide the quotient of step 2 by 128
4. The quotient from step 3 consists only of the modulo 63, 61, 59, and 31 digits. These are now extended to include modulo 127 and 128 digits.

5. The remainder of this division is taken as the number expressed by the 127 and 128 digits before division. This number is extended to include modulo 63, 61, 59, and 31 digits.
6. The quotients and remainders are now added.

A block diagram of this operation is shown in Figure 8.

A floating-point mode of operation has been implied in the discussion. However, the arithmetic structure as defined will work equally well in a fixed point mode.

Floating-point round-off may be accomplished by adding a constant $K = 1/2b$ into position Z_5 just prior to the scaling operation. After scaling, if Z_0 is non-zero, the contents of the Z registers are to be shifted right and the exponent increased by one. The four high-order digits and the exponent are then to be stored.

8.11 A METHOD-PRECISION COMPARISON

In Cheney's[9] procedure, the scaling is accomplished before multiplication. Because of this feature, Cheney's method is the worst in terms of precision.

If A is scaled by m_1 and B by m_2

$$A = \left[\frac{A}{m_1}\right] m_1 + |A|_{m_1} = a_1 \circ m_1 + a_0 \qquad (8\text{-}44)$$

$$B = \left[\frac{B}{m_2}\right] m_2 + |B|_{m_2} = b_1 \circ m_2 + b_0 \qquad (8\text{-}45)$$

where

$$m_1 \circ m_2 = m \qquad (8\text{-}46)$$

then

$$\frac{AB}{m} = a_1 b_1 + \frac{a_0 b_1}{m_1} + \frac{a_1 b_0}{m_2} + \frac{a_0 b_0}{m} \qquad (8\text{-}47)$$

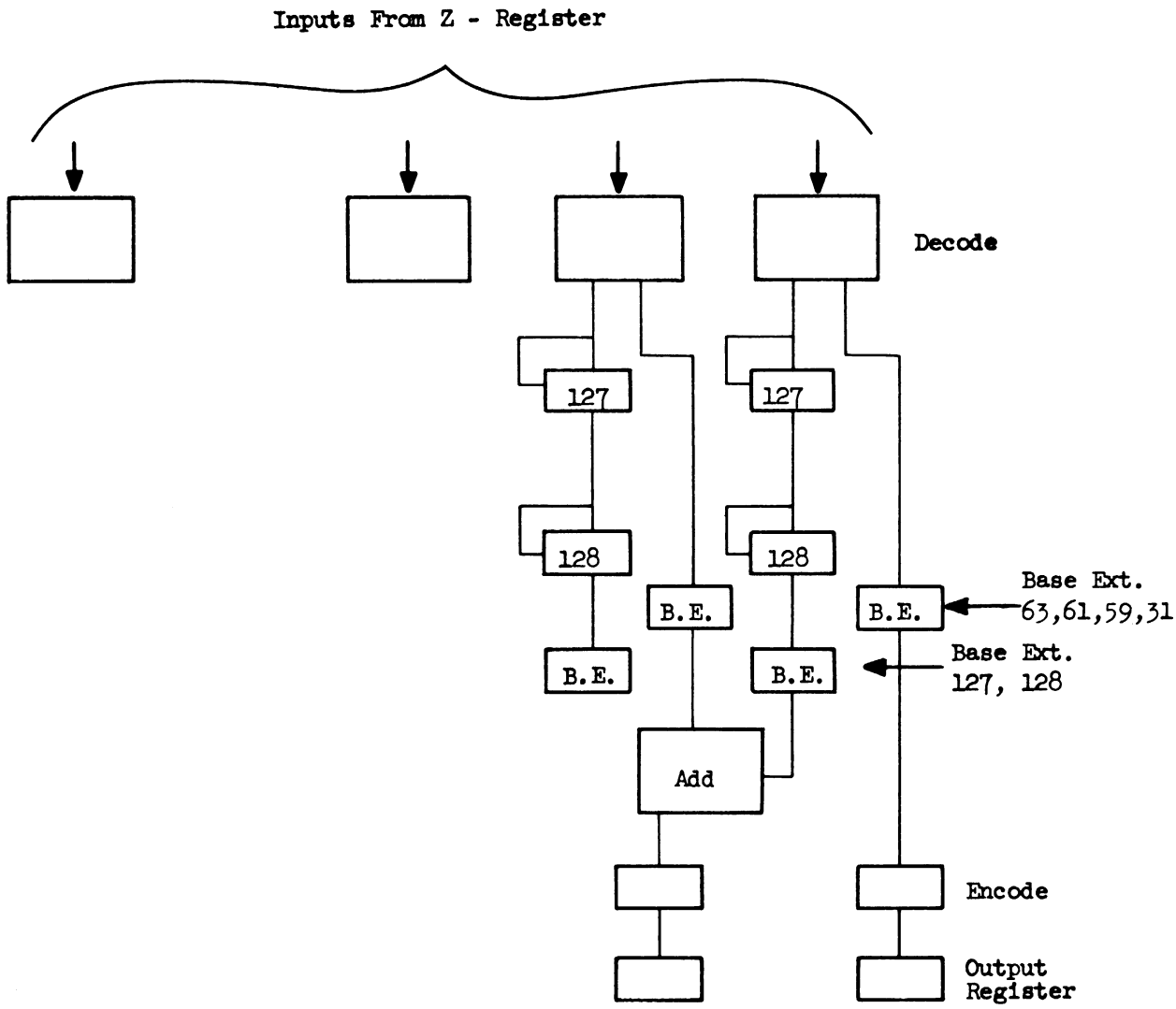


Figure 8. Block diagram for scaling operation.

By scaling before multiplication only $a_1 \cdot b_1$ is kept. Thus the contributions of

$$\frac{a_0 b_1}{m_1} + \frac{a_1 b_0}{m_2} \quad \text{to} \quad \left[\frac{AB}{m}\right]$$

are lost.

In Svoboda's[2] method, AB = C is completely available in the double-length form. C is then scaled by m, with the result that

$$\left[\frac{C}{m}\right] = \left[\frac{AB}{m}\right]$$

suffers no contribution losses as above. However, it should be observed that the comparison between the two systems does not take into account that the Cheney method employs a single-length operand, while the Svoboda method involves a double-length operand.

The precision advantage of the floating-point system over Svoboda's method follows from the degree to which $|AB|_m$ must be discarded.

In the most extreme case, when $\left[\frac{AB}{m}\right]$ is on the order of unity, the loss of precision in Svoboda's method can approach m/2. The corresponding worst case in the floating-point system occurs when the most-significant coefficient is on the order of unity. The error is then bounded by $2/m^3$, as was demonstrated in Section 8.4.

Certain assumptions must be made about the capacities of the systems in order to make further comparisons. If one takes the magnitude of Svoboda's single-length capacity to be equal to the mantissa-magnitude (approximately m^5 in Section 8.4) of the floating-point system, there exists a common basis for comparison.

The precision of Svoboda's method will, at best, be equal to that of the floating-point system. This occurs only when

$$\left[\frac{AB}{m}\right] = 0 \quad \text{or} \quad \left[\frac{AB}{m}\right] \simeq m$$

In the ideal case, with $\left[\frac{AB}{m}\right] = 0$, neither method loses any precision. This case is one of no multiplicative overflow, and represents only 1/m of the total number range. In the larger range, with $\left[\frac{AB}{m}\right] = E(E \neq 0)$, Svoboda's method loses precision approximately linearly as compared to the floating-point system; from equal precision when $E \simeq m$, to the extreme case (m/2 when $E \simeq 1$) described above.

CHAPTER IX

IMPLEMENTATION OF SIGN DETERMINATION METHODS

9.1 INTRODUCTION

The problem of sign determination in a RNS has received a great deal of attention, and five different methods have been proposed for the solution of that problem in an efficient manner.

Efficiency here means the capability of operating in a very short time, of the order of one or two pulse-times, and the possibility of implementing in practice the conversion network with a reasonable number of switching elements. The methods referred to are the following:

(1) Implementation of the conversion equations;

(2) Change of representation;[11]

(3) Addition of $\frac{1}{m_i} \left|\frac{X}{m_i}\right|_{m_i}$ quantities;[11]

(4) Method of estimates;[12] and

(5) Successive reduction.[13]

These five methods can be grouped in three categories, according to the means by which the conversion is achieved:

(a) Methods implementing the conversion equations: (1) (2)

(b) Methods using stored tables: (3) (4)

(c) Methods using a reduction procedure (5).

If one is interested in extreme economy of equipment, the methods under (a) must be considered, taking into account that any gain in simplicity is

paid for by a loss in speed. In general, this method takes (n-1) pulse-times to be completed, where n is the number of moduli used in the system.

This type of conversion procedure has been dealt with only as an elegant but impractical solution because it has been considered that for a practical RNS, for example one with a range equal to 2^{30}, the number of elements would become impractically large. This statement appears in practically every paper on the subject, and must have been suggested by a wrong association with the idea of a decoding network in the form of a complete tree.

This type of network requires 2^{n-1} elements for n variables (levels in this case). In a system with a range of 2^{30}, a set of at least 8 prime moduli is needed if small moduli are used, and an examination of the number of binary digits necessary to encode the coefficients of the associated mixed radix representation leads to the conclusion that the decoding tree for performing conversion from the mixed radix to the residue system comprises 29 levels.

If this network were a complete tree, the number of elements would be $536,870,911 = 2^{29}-1$.

The networks for obtaining the other residues have a progressively-diminishing number of elements, but the number of elements in the very first network is enough to discourage any further attempt.

It will be shown that the number of elements for such a conversion is actually very small and depends on the encoding used. For a RNS with a range of 2^{30} it is never much greater than 1,000.

An implementation of this method, using current steering techniques, is given in Section 9.2, and Section 9.3 deals with the procedure for determining

the number of elements of the conversion network for the general case of several moduli and large range.

A criterion for the choice of moduli, resulting in an economy in the number of switching elements, is given in Section 9.4.

If a very fast conversion procedure is desired, only those methods which are inherently non-sequential should be considered. In this case only those under (b) are capable of being implemented with a reasonable amount of circuitry.

Section 9.5 covers the application of fractional-binary-coding, to the necessary tables for further reduction of the number of elements, at the expense of one additional pulse-time, bringing the total time needed for sign-determination to two pulse-times. This procedure, although presented as part of an algorithm using a table of $\frac{1}{m_i} \left|\frac{x}{m_i}\right|_{m_i}$ quantities, actually can be applied to any method using a table-look-up operation together with subsequent addition of the quantities thus obtained.

Section 9.6 explains the general structure of the network, and Sections 9.7, 9.8 and 9.9 present a systematic procedure for determining the number of required elements without actually drawing the network. This procedure is based on a general algorithm for developing a carry determining network (for any number of variables) from a smaller network that is easily drawn.

9.2 RESIDUE TO MIXED BASIS CONVERSION USING CURRENT STEERING CIRCUITRY

The networks used to perform the conversion from residue to mixed basis representation implement the equations of procedure number 1.

$$a_n = |x|_{m_n} \tag{9-1}$$

$$a_{n-1} = \left| |\frac{1}{m_n}|_{m_{n-1}} \cdot |(x_{m_{n-1}} - a_n)|_{m_{n-1}} \right|_{m_{n-1}} \tag{9-2}$$

$$a_{n-2} = \left| \left| |\frac{1}{m_n}|_{m_{n-2}} \cdot |\frac{1}{m_{n-1}}|_{m_{n-2}} \right|_{m_{n-2}} \cdot \left| |x|_{m_{n-2}} - a_n - |m_n|_{m_{n-2}} \cdot a_{n-1} \right|_{m_{n-2}} \right|_{m_{n-2}} \tag{9-3}$$

In general:

$$a_i = \left| \left| |\frac{1}{m_n}|_{m_i} \cdot |\frac{1}{m_{n-1}}|_{m_i} \cdot |\frac{1}{m_{n-2}}|_{m_i} \cdot |\frac{1}{m_{n-1+1}}|_{m_i} \right|_{m_i} \cdot \left| |x|_{m_i} - a_n - |m_n|_{m_i} \cdot a_{n-1} - |m_n \cdot m_{n-1}|_{m_i} \cdot a_{n-2} - |m_n \cdot m_{n-1} \cdots m_{n-i+2}|_{m_i} \cdot a_{i+1} \right|_{m_i} \right|_{m_i}$$

If one considers the case where only three moduli are used, the preceding equations take the form:

$$a_3 = |x|_{m_3} \tag{9-4}$$

$$a_2 = \left| |\frac{1}{m_3}|_{m_2} \cdot \left| |x|_{m_2} - a_3 \right|_{m_2} \right|_{m_2} \tag{9-5}$$

$$a_1 = \left| |\frac{1}{m_3} \cdot \frac{1}{m_2}|_{m_1} \cdot \left| |x|_{m_1} - a_3 - |m_3|_{m_1} \cdot a_2 \right|_{m_1} \right|_{m_1} \tag{9-6}$$

Each of the equations (9-1), (9-2) and (9-3) is implemented by an independent network, but as a_n is always equal to $|x|_{m_n}$, only n-1 networks are required.

The inputs to each network are the residues, coded as one number per wire, therefore the number of input lines is equal to the corresponding modulus.

As an example, Figures 9 and 10 represent the conversion networks to perform conversion from the system m_i: (4,3,5) to the associated mixed basis system.

The broken line is the path of information when the number 53 in residue notation (1,2,3) is converted to the mixed basis representation: (3,1,3).

The procedure includes the following steps, as indicated on Figures 9 and 10:

1. The input (a_1) is received on wire number 1.

2. The first binary digit of $|a_3|_{m_1} = |3|_4 = 3 = \underline{11}$ is inspected.

3. The first binary digit of $|a_3|_{m_1}$ is subtracted.

4. The interchange of wires in this step has no arithmetic meaning. It is only a rearrangement to simplify the drawing. Every wire retains its original weight.

5. The second binary digit of $|a_3|_{m_1}$ is inspected. In this case $|a_3|_{m_1}$ in binary is $\underline{11}$, and the second digit is also a $\underline{1}$.

6. The second binary digit of $|a_3|_{m_1}$ is subtracted. Weight = 2.

7. Multiplication by $|1/m_3|_{m_1}$ is performed. In this example $|1/m_3|_{m_1} = |1/5|_4 = 1$.

8. The first binary digit of $|a_2|_{m_1}$ is inspected. $|a_2|_{m_1} = 1$. First binary digit is 1.

9. The first binary digit of $|a_2|_{m_1}$ is subtracted, with a weight of 1.

10. No operation.

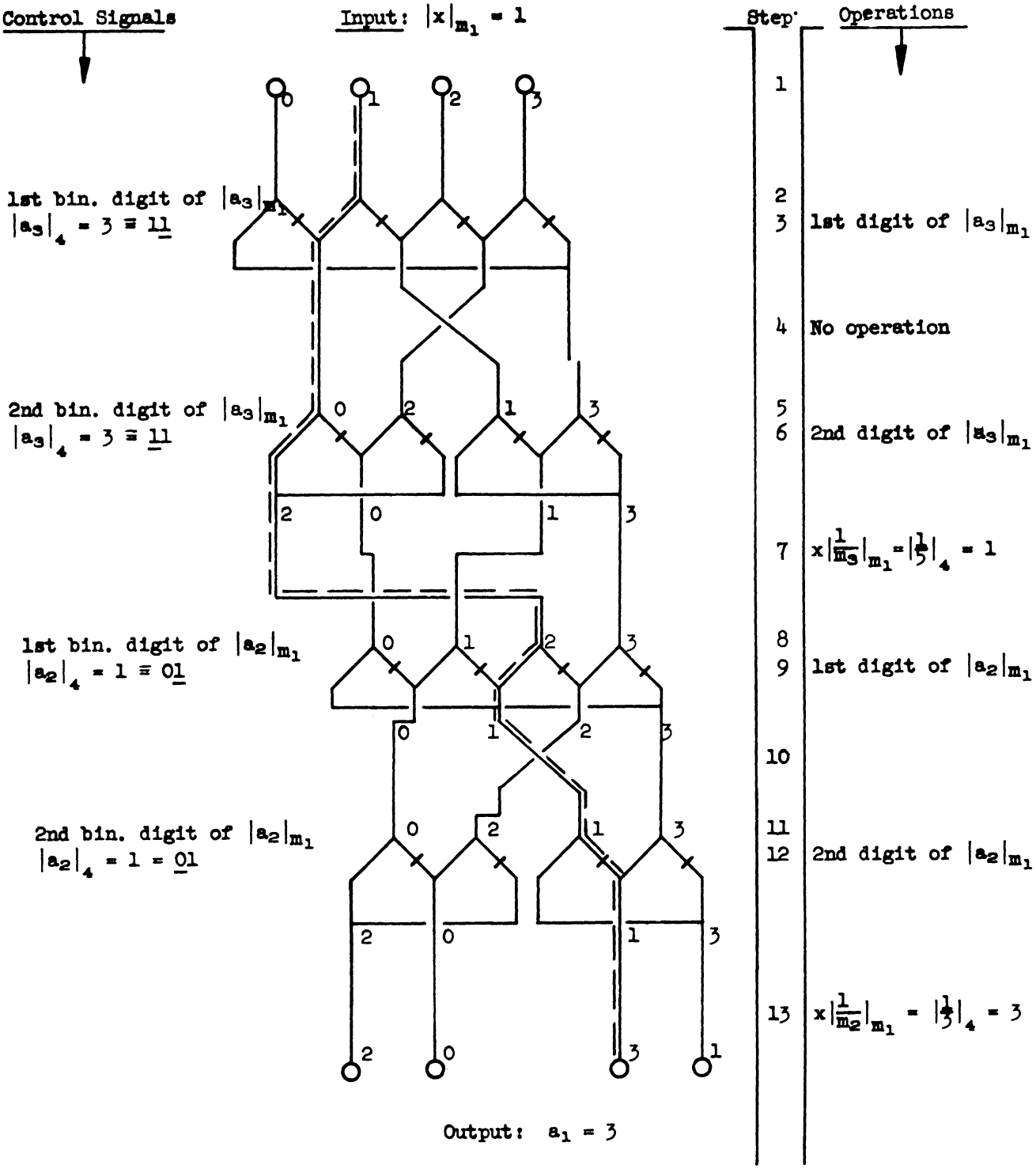


Figure 9. Switching network for residue to mixed basis conversion network (1st part).

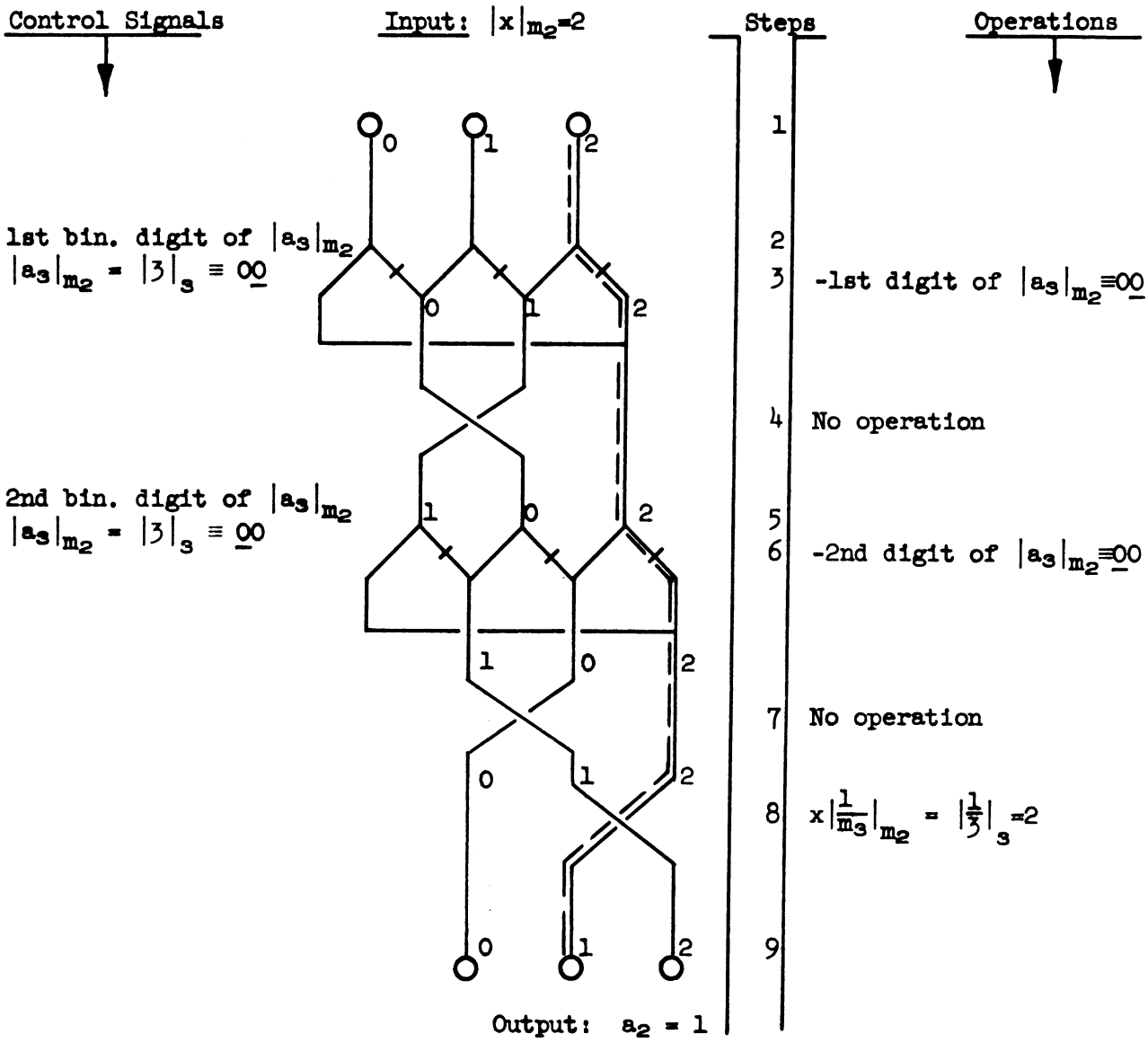


Figure 10. Switching network for residue to mixed basis conversion network (2nd part).

11. The second binary digit of $|a_2|_{m_1}$ is inspected. $|a_2|_{m_1} = 1$.

Therefore the second binary digit is 0.

12. The second binary digit, with a weight of 2, is subtracted.

13. Multiplication by $|1/m_2|_{m_1}$ is performed: $|1/m_2|_{m_1} = |1/3|_1 = 3$.

14. The output is $a_1 = 3$.

It is to be noted that the number of levels that are necessary depends on the number of a coefficients to be inspected, which is (n-1) for the first network, and on the number of binary bits needed to represent the a coefficients modulo m_1. As $m_1 = 4$, the largest magnitude to be dealt with is 3, and therefore only two binary bits are necessary for each coefficient.

The complete network to perform conversion in this system is shown in Figure 11, and in more detail in Figure 12 where the ∩ and ⊐ elements and connections indicate auxiliary equipment performing notation changes, encoding or modulo operations not essential to the conversion procedure.

The operation is sequential, that is, the coefficient a_8 appears first at the output; after a certain time interval a_7 is obtained, and so forth.

Therefore the total delay to obtain the complete conversion is the sum of the propagation times through the path shown as a broken line in Figure 12.

All the $|x|_{m_i}$ are applied simultaneously as inputs, but they are delayed by the delay elements ⊏⊐ to allow time for the modulo and conversion operations to take place on $|x|_{m_8}$.

After that delay, both signals, that is, the individual $|x|_{m_i}$ and the corresponding setting signal for the switching elements obtained from $|x|_{m_8}$,

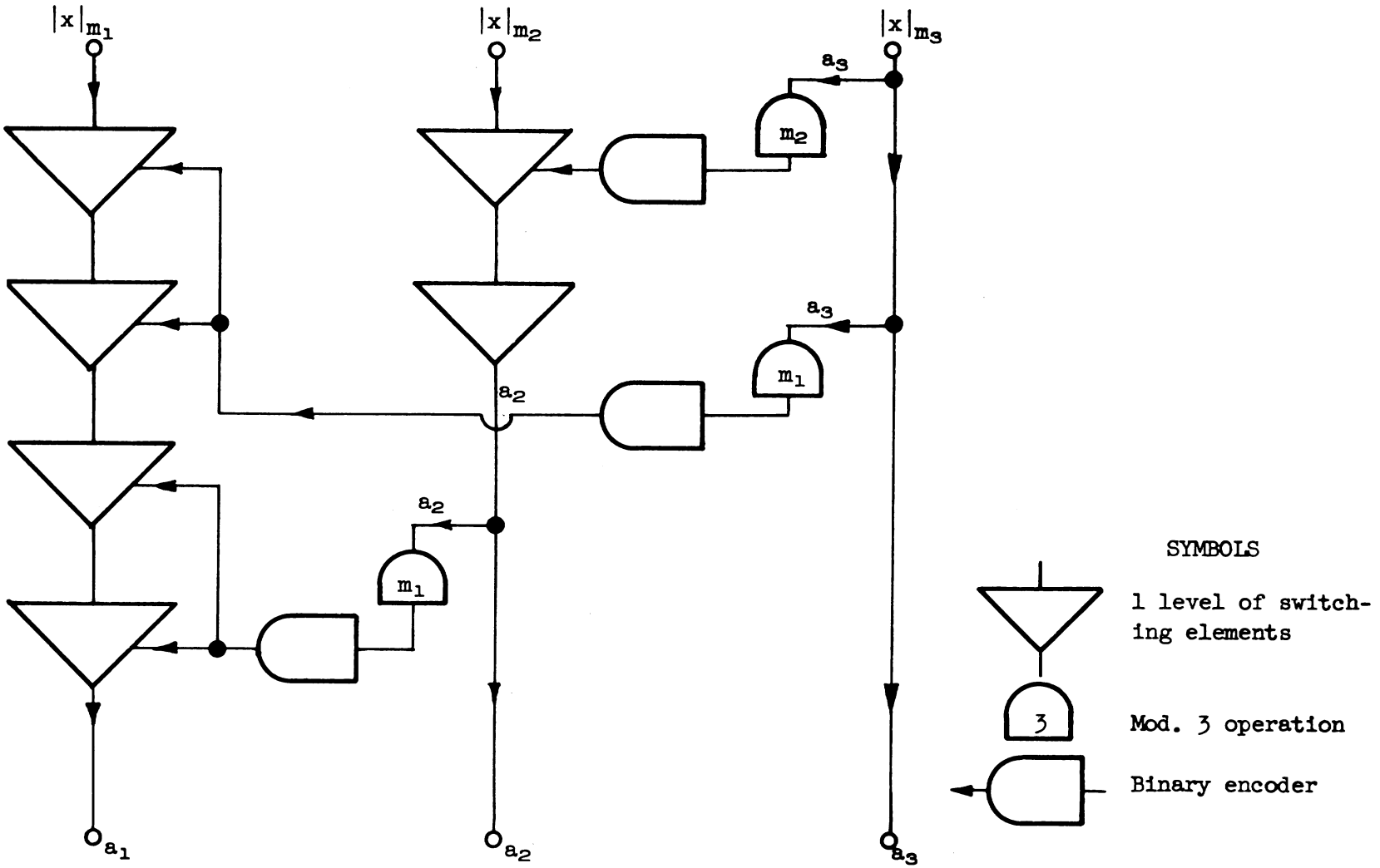


Figure 11. Simplified schematic of the complete residue to mixed basis conversion network.

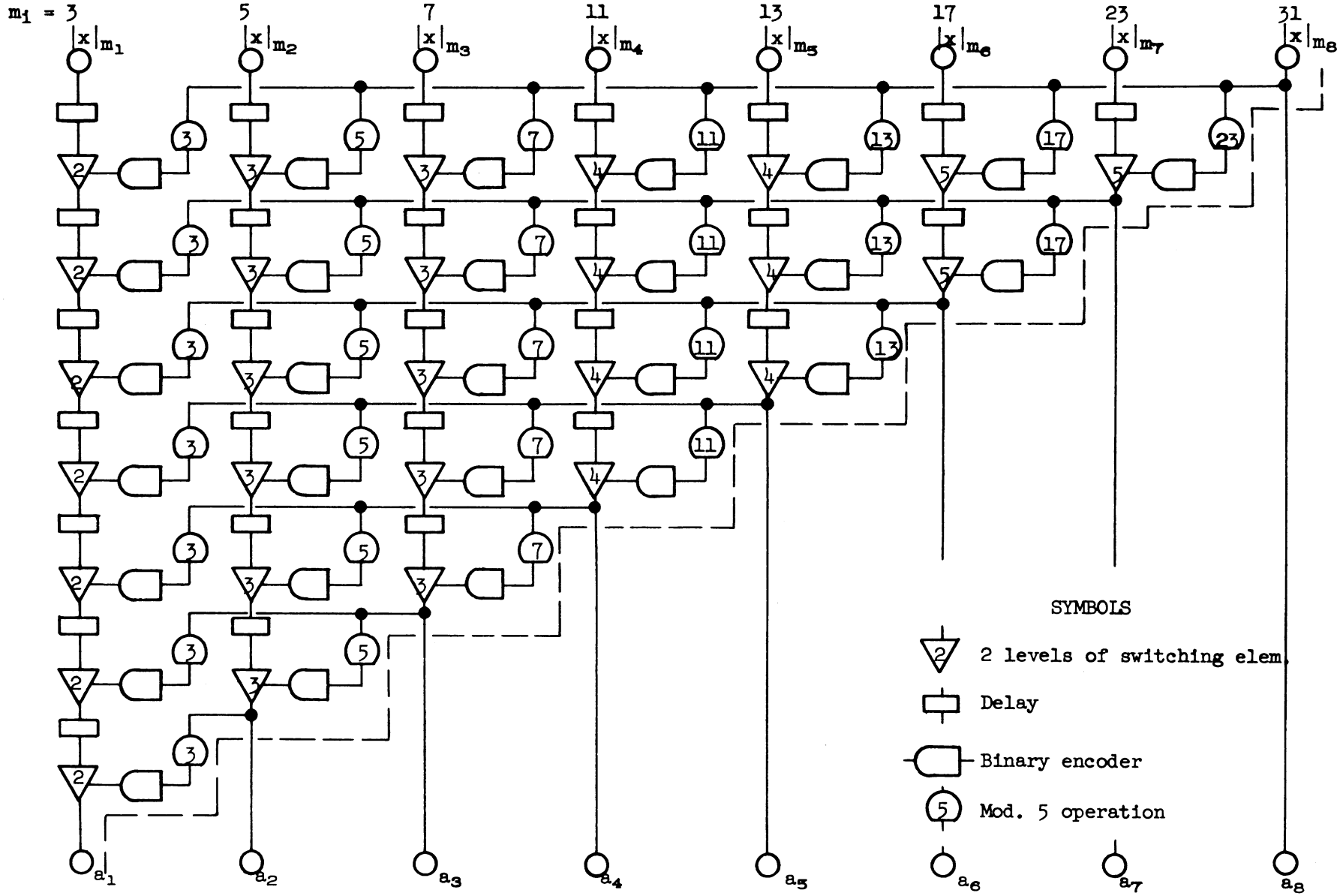
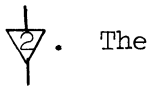


Figure 12. Block diagram of the conversion network from the residue system to mixed basis.

are applied to the first group of switching levels, indicated in Figure 2 as:  . The number inside the  indicates the number of levels actually comprising the group.

The output of the modulo 23 column is already a_7, from which the setting signals for the second group (second row) of switching elements are derived by modulo and conversion operations.

Therefore, initially there are no outputs, and the a_i coefficients begin to appear successively in decreasing order until they are all available after a time equal to the total propagation time through the chain of elements indicated by the broken line in Figure 12.

A slightly simpler network can be obtained by omitting the delay elements  in Figure 12; but then the output has to be strobed after a time equal to the total delay Δ, because in this case there are output signals present since the moment the $|x|_{m_i}$ input signals are applied, but it is only after a time Δ that they are all correct.

The initial values present at the output depend on the state in which previous operations left the switching elements, and they change successively into the correct value as the signal travels through the broken line diagonal path of Figure 12.

9.3 NUMBER OF ELEMENTS IN A CONVERSION NETWORK

When performing conversion from a residue number representation, the number of independent conversion networks is one less than the number of moduli of the system, because the coefficient a_n in the mixed radix representation in equal to the residue $|x|_{m_n}$.

The number of input lines to the ith network is equal to the corresponding modulus m_i. For example, if for a network $m_i = 3$, the number of input lines will be 3, because the input is a_i, and this can take any value up to m_i-1; therefore the largest possible value is 2, but an additional line is needed for the 0 (zero) signal, which brings the total back to 3.

The number of levels is the product of the number of binary digits necessary to represent moduli i the coefficients a_i from a_{i+1} to a_n, times the number of coefficients to be added up, which is equal to (n-i).

The itemized calculation for a residue system with 8 moduli is given in Table 8.

For the first network, $m_i = 3$, and the number of coefficients to be added is 7: a_2 through a_8. All these coefficients are taken moduli 3 and therefore are represented in a binary notation by only two digits. The total number of levels needed is therefore: $2x7 = 14$.

The following general formula can be used to calculate the total number of switching elements of a conversion network for a system of any number of moduli:

$$\text{No. of elements} = \sum_{i=1}^{(n-1)} (n-i) \times m_i \times F(m_i) \qquad (9\text{-}7)$$

$F(m_i)$ is a function giving the number of binary digits necessary to represent a number modulus m_i.

When operating moduli m_i, the largest number that can occur is evidently (m_i-1), and for systems with a small number of moduli, $F(m_i)$ can be obtained from the following table:

TABLE 4

TABLE OF THE FUNCTION $F(m_i)$

Any m_i up to	2	4	8	16	32	64	128
Largest No. poss.	1	3	7	15	31	63	127
$F(m_i)$	1	2	3	4	5	6	7

In those cases where the number of moduli is very large, or the calculation is to be included in a program as a subroutine, $F(m_i)$ can be obtained from the expression:

$$F(m_i) = \text{integer part of } \frac{\log (m_i+1)}{0.3010} + 0.99 \qquad (9\text{-}8)$$

It is interesting to note that given a residue number system and the associated mixed basis system, the conversion either way takes the same number of switching elements, although the networks are quite different in their circuitry, and in the time required for the conversion.

Modular system with moduli: 3, 5, 7, 11, 13, 17, 23, 31.

TABLE 5

PROCEDURE FOR THE CALCULATION OF THE TOTAL NUMBER OF ELEMENTS

1	2	3	4	5 = 3x4	6	7 = 5x6
1	3	7	2	14	3	42
2	5	6	3	18	5	90
3	7	5	3	15	7	105
4	11	4	4	16	11	176
5	13	3	4	12	13	156
6	17	2	5	10	17	170
7	23	1	5	5	23	115
					Total number of elem. =	854

Column 1: network number
Column 2: m_i
Column 3: number of coefficients to be added
Column 4: number of binary digits necessary to represent the coefficients moduli m_i
Column 5: number of levels
Column 6: number of input lines
Column 7: number of elements.

The table beginning on page 145 gives the number of elements necessary to implement the conversion network for different sets of moduli, all giving a range equal or greater than a specified value. The results are a direct application of the equation (9-7) and have been calculated for values of M from 2^{27} through 2^{34}. Only the table for 2^{27} is included here.

For the calculation of these tables, the first 400 prime numbers, excluding 2, have been considered as numbered from 1 to 400. Therefore, the numbers in columns 1 and 2 identify the first and last moduli of the set by their ordinal position; i.e., 3 indicates the third prime number, that is 7.

The tables contain the following information:

Column 1: The first modulus.

Column 2: The last modulus.

Column 3: The number of moduli in the set.

Column 4: The actual range obtained, in floating point notation.

Column 5: The number of elements in the conversion network.

Column 6: The number of elements normalized for a range of 10^{10}.

Column 7: The minimum range, in floating point notation, as prescribed in the program.

The same type of information is presented in Figure 13, beginning on page 146.

Here only the first and last moduli are indicated, together with the number of elements. The length of the lines of 8's in the scale indicated, is proportional to the number of elements.

TABLE 6

NUMBER OF ELEMENTS IN THE CONVERSION NETWORK FOR DIFFERENT SETS OF MODULI

First Modulus Prime No.	Last Modulus Prime No.	Number of Moduli	Range Obtained	Number of Elements	Number of Elements for a Range = 10^{10}	2^{27}
3	9	7	215656259	1114	51656	134217759
4	10	7	955048459	1524	15957	134217759
6	11	6	247110459	1595	64546	134217759
7	12	6	595971659	1902	31914	134217759
8	13	6	134877760	2310	17126	134217759
9	14	6	275619560	2761	10017	134217759
11	15	5	1[illegible]2489759	2424	149178	134217759
12	16	5	259105159	2[illegible]40	101889	134217759
13	17	5	385497559	2868	74397	134217759
14	18	5	600658559	3156	52542	134217759
15	19	5	907376959	[illegible]535	38958	134217759
16	20	5	124978260	3949	31597	134217759
17	21	5	167343660	4376	26149	134217759
18	22	5	227696860	4942	21704	134217759
19	23	5	302462760	5208	17218	134217759
20	24	5	413223560	5488	15280	134217759
21	25	5	571720060	5880	10284	134217759
22	26	5	743406960	[illegible]258	8395	134217759
23	27	5	960945460	6664	6934	134217759
24	28	5	117688661	7028	5971	134217759
26	29	4	135743659	3879	285759	134217759
27	30	4	167373059	3474	207560	134217759
28	31	4	204914459	3125	152502	134217759
29	32	4	257552659	5199	201861	134217759
30	33	4	316812259	5859	184936	134217759
31	34	4	371692859	6448	173476	134217759
32	35	4	428439559	6704	156474	134217759
33	36	4	490985159	6928	141104	134217759
34	37	4	575759259	7248	125885	134217759
35	38	4	645313759	7440	115292	134217759
36	39	4	739332659	7386	99900	134217759
37	40	4	804900059	7968	94507	134217759
38	41	4	936017259	8208	87690	134217759
39	42	4	1070533460	8464	79063	134217759
40	43	4	11942946[illegible]	8720	73013	134217759
42	44	4	131439060	8944	68046	134217759
42	45	4	144310260	9248	63995	134217759
43	46	4	159642160	9376	58731	134217759
44	47	4	184456960	9600	52044	134217759
45	48	4	212546760	9936	46747	134217759
46	49	4	244588660	10448	42716	134217759
47	50	4	270090460	10816	40045	134217759
48	51	4	289468860	10976	37917	134217759
49	52	4	307321260	11136	36235	134217759
50	53	4	336845060	11344	33677	134217759
51	54	4	371541260	11600	31221	134217759
52	55	4	408850460	12113	29626	134217759
53	56	4	456351360	13017	28524	134217759
54	57	4	492713760	14094	28604	134217759
55	58	4	531056760	14382	27081	134217759
56	59	4	567402260	14634	25791	134217759
57	60	4	596931960	14832	24847	134217759
58	61	4	645390860	15084	23371	134217759
59	62	4	715288360	15318	21415	134217759
60	63	4	791653160	15678	19804	134217759
61	64	4	875573360	16236	18543	134217759
62	65	4	947292060	16704	17633	134217759
63	66	4	1021347[illegible]	16884	16931	134217759
64	67	4	1106731[illegible]	17136	15483	134217759
65	68	4	1226949[illegible]	17550	14303	134217759
66	69	4	1350804[illegible]	18126	13418	134217759
67	70	4	1440584[illegible]	18486	12832	134217759
68	71	4	1534626[illegible]	18828	12268	134217759
69	72	4	1623076[illegible]	19008	11711	134217759
70	73	4	1734690[illegible]	19296	11123	134217759
71	74	4	1862455[illegible]	19656	10553	134217759
72	75	4	1986962[illegible]	20034	10082	134217759
73	76	4	2106069[illegible]	20340	9657	134217759
74	77	4	2241579[illegible]	20628	9202	134217759
75	78	4	2371696[illegible]	20916	8819	134217759
76	79	4	2532696[illegible]	21258	8393	134217759
77	80	4	2728018[illegible]	21618	7924	134217759
78	81	4	2892935[illegible]	21960	7590	134217759
79	82	4	3109361[illegible]	22374	7195	134217759
80	83	4	3291816[illegible]	22770	6917	134217759
81	84	4	3448942[illegible]	23022	6675	134217759
82	85	4	3629170[illegible]	23382	6442	134217759
83	86	4	3780735[illegible]	23580	6236	134217759
84	87	4	3990288[illegible]	23868	5981	134217759
85	88	4	4190255[illegible]	24156	5764	134217759
86	89	4	4379430[illegible]	24498	5593	134217759
87	90	4	4553499[illegible]	24804	5445	134217759
88	91	4	4774268[illegible]	24984	5233	134217759
89	92	4	5043523[illegible]	25218	5000	134217759
90	93	4	5348522[illegible]	25614	4788	134217759
91	94	4	5719006[illegible]	26118	4570	134217759
92	95	4	6001344[illegible]	26478	4412	134217759
93	96	4	6272446[illegible]	26766	4267	134217759
94	97	4	6655681[illegible]	27108	4072	134217759
96	98	3	138683659	14372	1036315	134217759
97	99	3	147402359	15650	1061720	134217759
98	100	3	1547581[illegible]	15870	1025471	134217759
99	101	3	1648187[illegible]	16290	988358	134217759

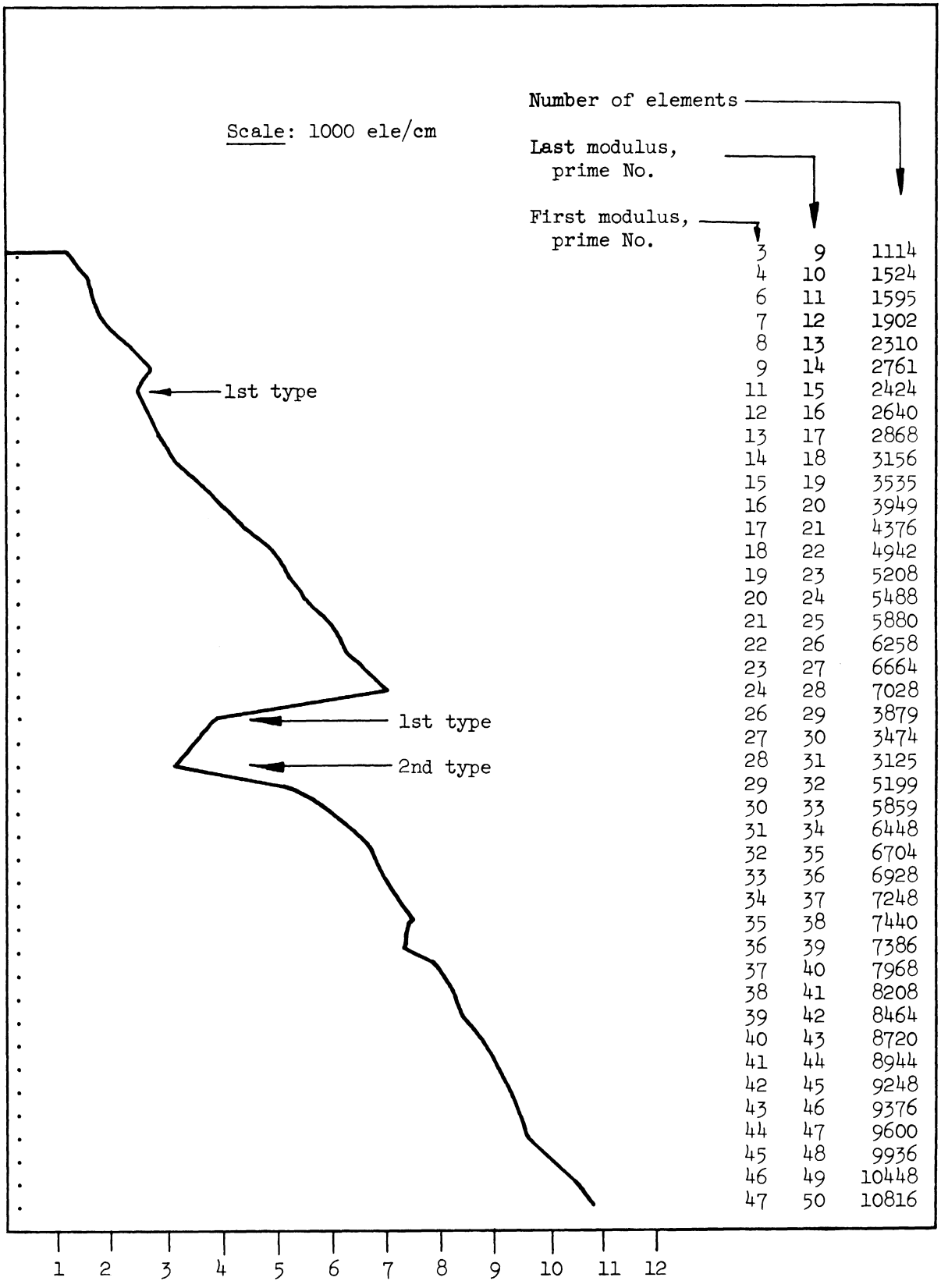


Figure 13. Variation of the number of elements in the conversion network.

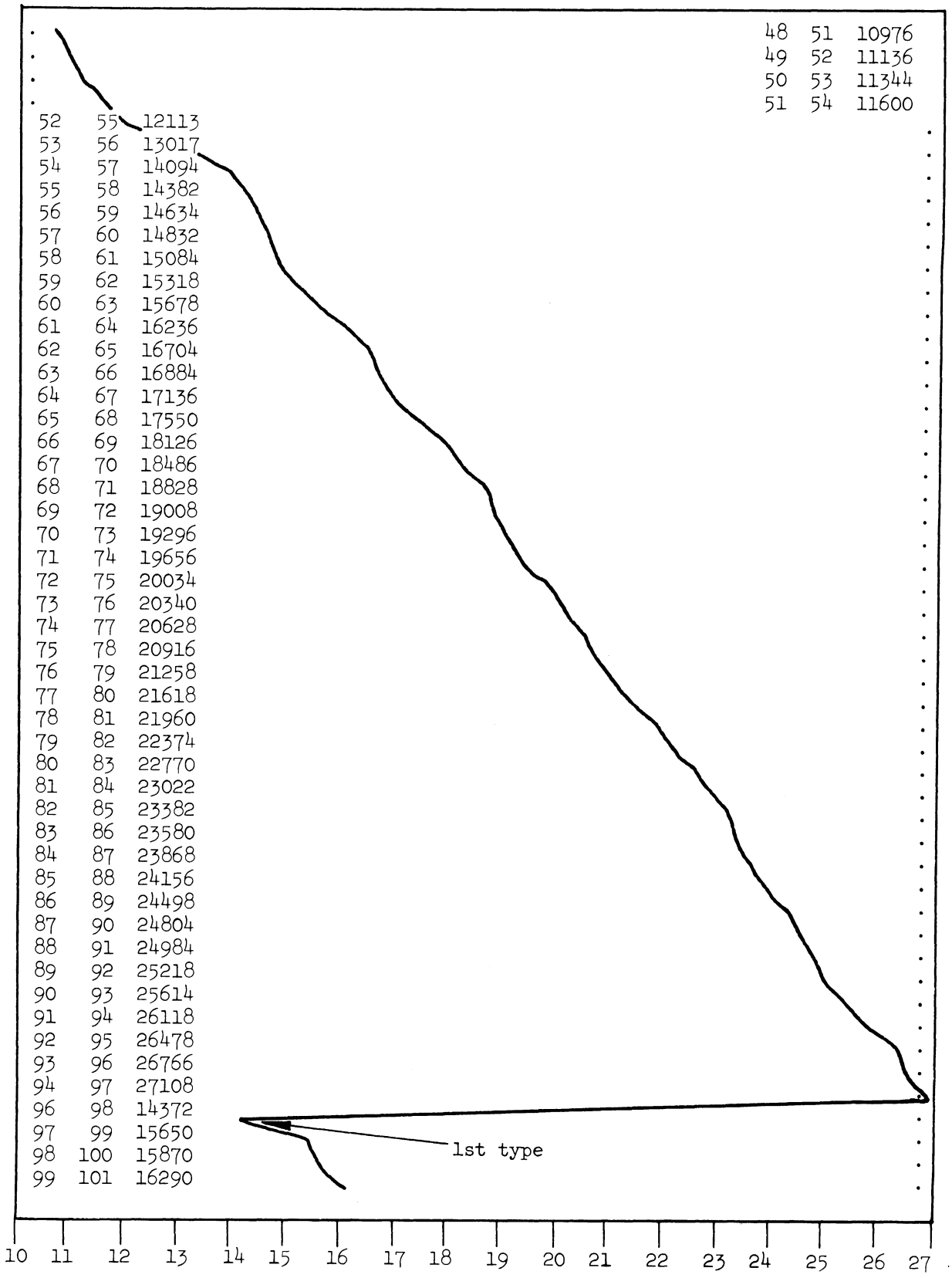


Figure 13 (Concluded).

9.4 SELECTION OF MODULI FOR MAXIMUM ECONOMY

It is evident that some combinations of moduli require a minimum of elements, and that two types of relative minimums can be distinguished:

One type is produced when the number of moduli in the set drops by one, because as larger moduli are considered suddenly the range can be obtained with one less modulus.

The second type occurs even when the number of moduli remains constant, and is produced when the nth-1 modulus takes the largest value not exceeding one of the following magnitudes: 2^k-1. That is when the nth-1 modulus takes the maximum value less than or equal to: 3, 7, 15, 31,....

This can be explained in the following way: The nth modulus does not influence the number of elements because it gives directly the last coefficient in the mixed radix system. When the nth-1 modulus takes for example the value 15, it is possible to encode it using four bits to full capacity. If the modulus were 16, it would be necessary to use 5 bits and therefore five levels of switching elements.

The rest of the moduli are also encoded in the same number of bits or in less, because they are all smaller than the nth-1.

In the graph for a range of 2^{27}, the first type of minimum occurs for the sets having the following first moduli: 6, 11, 26, 96. In those cases, the number of moduli is reduced by one.

The second type is evident in the set having prime number 28 as the first modulus. This set is composed of the following primes: 109, 113, 127, and 131. We see that the nth-1 modulus is 127, which is the maximum value representable by 7 binary bits.

9.5 THE APPLICATION OF BINARY CODING TO CONVERSION PROCEDURES USING STORED TABLES

One of the procedures used to obtain the mixed radix coefficients corresponding to a given residue representation consists of the addition of the corresponding values of the quantities $\frac{1}{m_i}\left|\frac{x}{\hat{m}_i}\right|_{m_i}$ in mixed basis notation.

This procedure requires the storage of n tables, n being the number of moduli in the system. Each table contains the values of $\frac{1}{m_i}\left|\frac{x}{\hat{m}_i}\right|_{m_i}$ for all possible values of $|x|_{m_i}$, expressed in mixed basis notation.

From these tables one obtains the mixed basis coefficients corresponding to each of the residues. They are added in a modular adder, taking into account the carries that may occur.

The main disadvantage of this method is that it reintroduces the problem of carries. But when the problem is sign determination rather than change of representation, the method can be modified to perform the determination of sign in only one pulse time, assuming that the corresponding quantities have already been obtained from the tables.

This modification is based on the fact that when a number x is represented as:

$$x = M \cdot \sum \frac{1}{m_i}\left|\frac{x}{\hat{m}_i}\right|_{m_i},$$

where M is the range of the system, the summation takes values less than 1/2 for the first half of the range, and values greater or equal to 1/2 for the second half. Now if the quantities stored in the tables are represented in binary form, it is not necessary to perform the actual addition of all the quantities, but only the determination of the first binary bit after the dec-

imal point. If this is a zero, the summation is less than 1/2 no matter what the other bits are. If the first bit is a 1, the sum is always equal to or greater than 1/2.

When all the quantities have been obtained from the tables, the carries are determined for all the columns except the first one, and the resulting carry is added to the digits of the first column.

For example, if the (2,3,5) modular system is used, the corresponding tables, expressed in binary notation, are as follows:

$\|x\|_2$	$\frac{1}{2}\|\frac{x}{1}\|_2$
0	.00000
1	.10000

$\|x\|_3$	$\frac{1}{3}\|\frac{x}{1}\|_3$
0	.00000
1	.01011
2	.10101

$\|x\|_5$	$\frac{1}{5}\|\frac{x}{5}\|_5$
0	.00000
1	.00110
2	.01101
3	.10011
4	.11010

For x = 16, residue representation is (0,1,1), and from the tables we obtain:

			1	1	1	
For	0	.0	0	0	0	0
For	1	.0	1	0	1	1
For	1	.0	0	1	1	0
		.1	0	0	0	1

As only the first digit after the decimal point is to be inspected, the complete sum is not needed. It suffices to determine the carries up to the first column, and then to add the last carry to the first column.

This simplification makes it possible to perform sign determination in a single pulse-time using a current steering network, because the necessary setting signals for all the levels are available at the same time, and there is no carry propagation problem.

9.6 GENERAL STRUCTURE OF THE NETWORK

The switching network for carry determination is very simple because it does not have to handle large numbers since the largest carry for a sum of N binary numbers is N-1, which happens only after several columns have been added.

For the first columns, the network has to handle carries of increasing magnitude. This necessitates an increasing number of wires to represent the carries, but after the maximum value of the carry has been reached, the complexity remains stationary. Then the same type of circuitry is repeated for the following columns.

Three well defined regions can be distinguished in the network. This begins with a single wire representing the original "zero" carry into the first column from the right. In the first region where the number of values that the carry can assume keeps growing, the individual networks representing the columns are all different, because the possibility of carries of larger magnitude demands more wires for their representation. This region ends with the network corresponding to the column where the carry first assumes its maximum value: N-1.

The second region is formed by a number of networks, all alike, corresponding to the columns up to and including the next to the last, for which the same number of values are possible for the carries.

The third region comprises only one network in which the resulting carry from the previous columns is added to the binary digits of the last column. The result is interpreted as (+) if it is a zero, and as (-) if it is a one.

Note that the networks in regions 1 and 2 perform binary carry determination, while the network of region 3 performs binary addition.

The configuration of the sign determining network for the example of Section 9.5 is shown in Figure 14.

A simplified network for the case where 4 moduli are used is shown in Figure 15.

9.7 DETERMINATION OF THE NUMBER OF ELEMENTS OF THE NETWORK

The three sections of the network, as described in 9.6, contain four different types of individual networks:

Region 1: The first network is of the type (1→M), indicating that it is of the type having only one input and M outputs. From the second network on, only the type (M→M+W) occurs, indicating a network having more output wires than input ones.

Region 2: All networks are of the type (M→M), that is all have equal number of input and output wires.

Region 3: The network is always of the type (M→2).

Each of these types of networks requires a separate calculation to determine the number of elements necessary for its implementation.

Networks of the first type (1→M) are the most difficult to evaluate because they are asymmetric and include many cases of "don't care" conditions, which tend to simplify the network but introduce difficulties for the systematic determination of the number of elements.

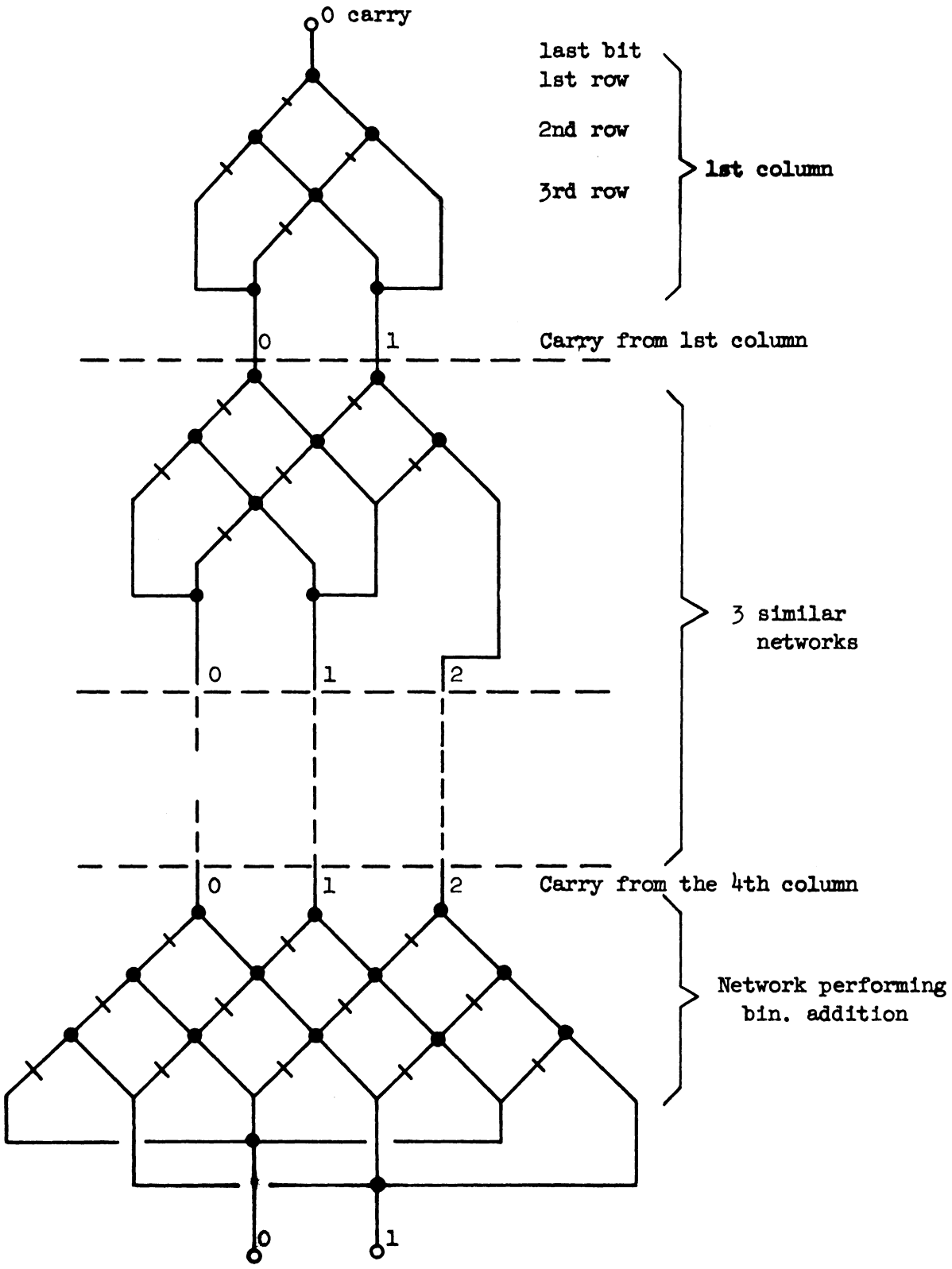


Figure 14. Switching network for sign determination for a residue system with 3 moduli.

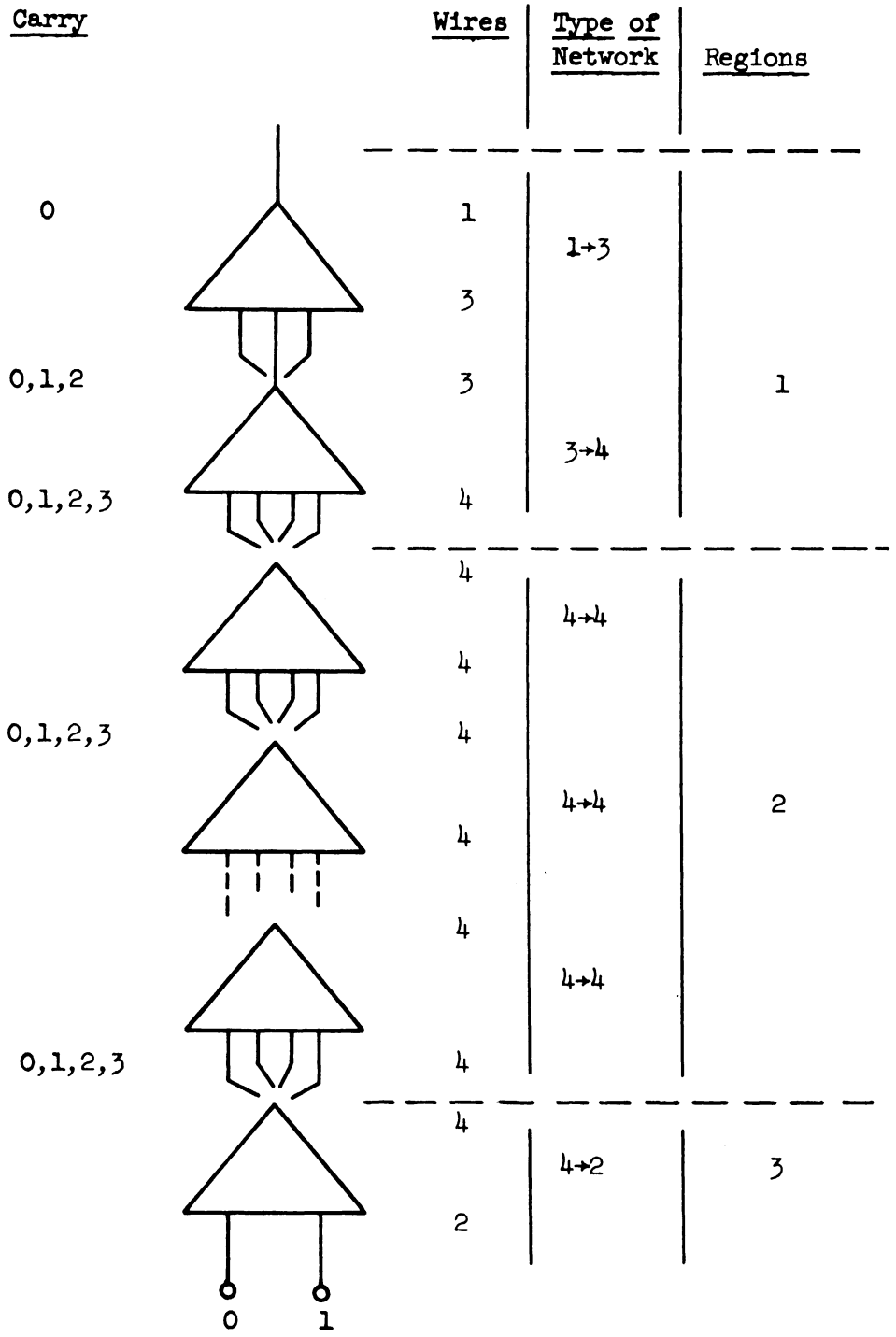
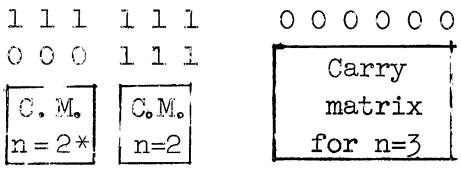


Figure 15. Block diagram of sign determination network for a system with 4 moduli.

The following is an explanation of a procedure by which the number of elements can be determined for any number of variables (levels) without actually drawing the complete network.

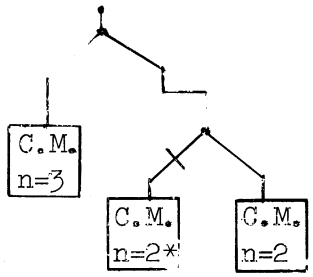
Figure 16 shows the carry matrix and the corresponding network for the case n = 3, while Figure 17 shows the case n = 4.

It can be seen that the carry matrix for the case of four variables can be thought of as being formed in the following manner:



The asterisk (*) in the carry matrix of 2* is just a symbol to indicate that the matrix is derived from the matrix for n = 2 by interchanging 1's and 0's in the first row.

The corresponding diagram in network form, for the previous diagram can be indicated as:



The procedure can be extended for a larger number of variables, and the general structure of the network for the first column is indicated in Figure 18.

n = 3

0 0 0 1 1 1
1 1 0 1 0 0
1 0 E E 1 0
1 0 0 1 1 0

E = either 0 or 1

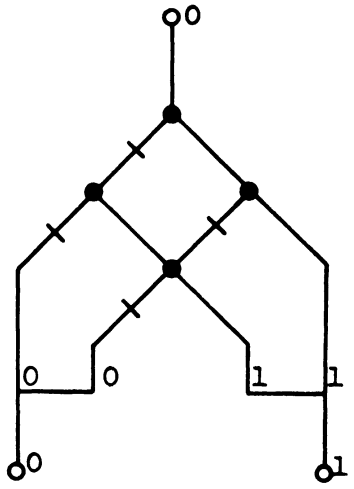


Figure 16. Carry matrix for three addends and corresponding network.

n = 4

0 0 0 0 0 0 1 1 1 1 1 1
0 0 0 1 1 1 0 0 0 1 1 1
1 1 0 1 0 0 1 0 0 1 1 0
1 0 E E 1 0 E 1 0 1 0 E
1 0 0 1 1 0 1 1 0 2 1 1

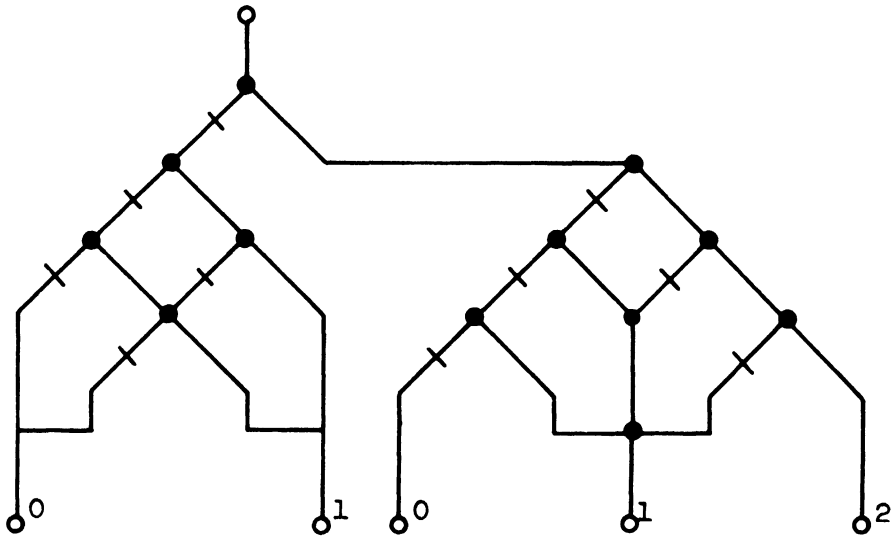


Figure 17. Carry matrix for four addends and corresponding network.

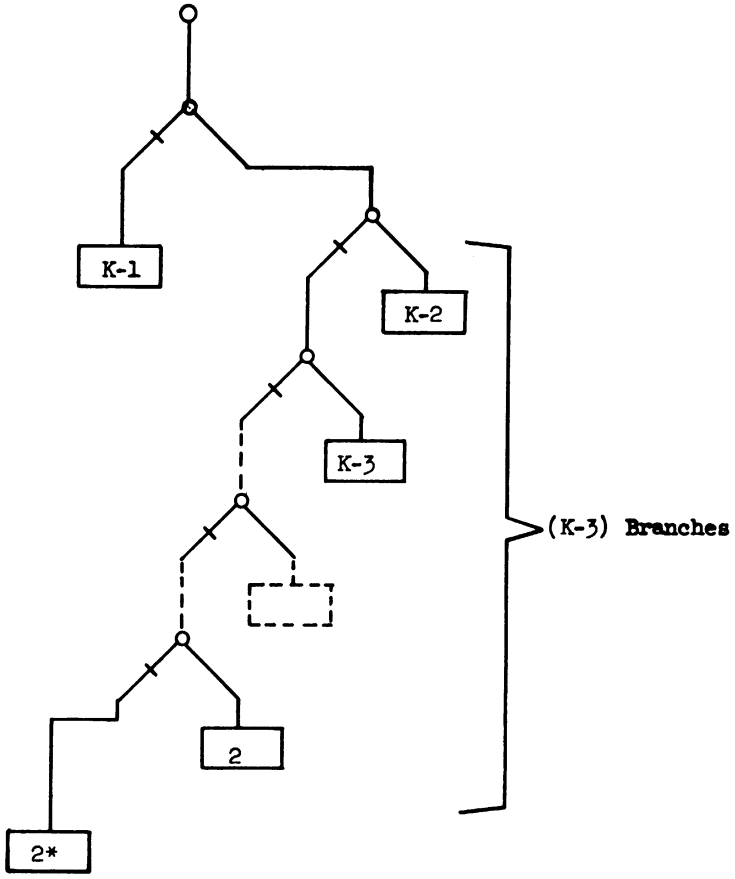


Figure 18. General structure of the carry determining network for the first column of K binary bits.

The number of elements in this network can be obtained from the following table:

TABLE 7

NUMBER OF ELEMENTS FOR NETWORKS OF THE TYPE (1→M)

Output wires, M	No. of levels = No. of moduli, N	Number of elements
2	3	4
3	4	10
4	5	21
5	6	43
6	7	87
7	8	175
8	9	351

For values of N = 6, the following formula can be used:

$$\text{Number of elements} = 17 \times 2^{(N-5)} + 4 \times 2^{(N-6)} + (N-1) + 5 \times (N-6) \quad (9\text{-}9)$$

Second type: Networks of the form (M→M+W).

These networks implement the carry determination procedure from the second column up to and including the column in which the carry assumes its maximum value.

The number and type of networks that are needed in each case can be determined from the following table, which gives the maximum carry from every column and at which column the carry stabilizes at its maximum value of N-1.

TABLE 8

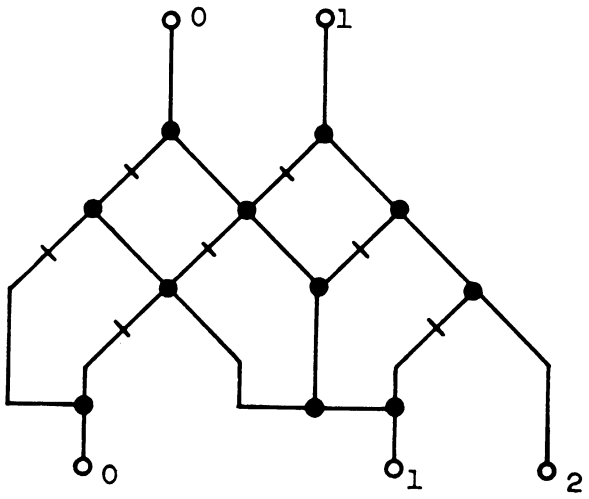
MAXIMUM CARRY AND CARRY PROPAGATION LENGTH

No. of binary bits in a column	Max carry from column to column						Max carry at column No.
2	1	1	1	1	1	1	1
3	1	2	2	2	2	2	2
4	2	3	3	3	3	3	2
5	2	3	4	4	4	4	3
6	3	4	5	5	5	5	3
7	3	5	6	6	6	6	3
8	4	6	7	7	7	7	3
9	4	6	7	8	8	8	4
10	5	7	8	9	9	9	4

It must be remembered that the table gives the maximum carry, not the number of wires, which is one greater.

Figure 19 shows the corresponding network for the case (2→3) for 3 levels.

Figure 20 shows the network for the case (3→4) for 4 levels. The



2→3 type for
3 levels

Figure 19. Carry determining network
for the (2 to 3) case.

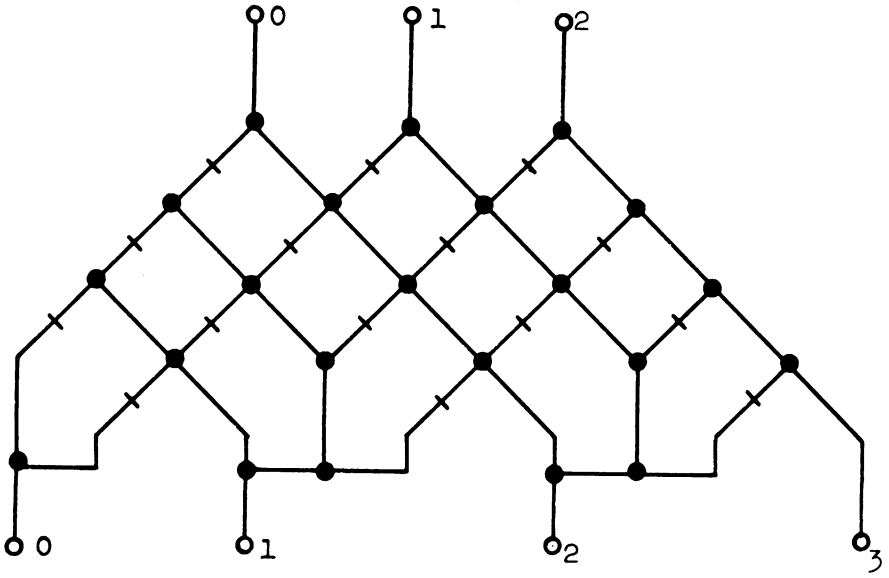


Figure 20. Carry determining network
for the (3 to 4) case.

preceding table indicates that for 4 levels, that is four binary bits per column, the carry from the first column into the second is a 2. Therefore the network for the second column must have 3 input wires (for the values 0,1,2) and 4 output wires (for the values 0,1,2,3). Therefore the network of Figure 20 represents the second network for the case n = 4.

Number of elements:

The number of elements of a network of the type (M→N+W) for N levels depends only on M and N, and is calculated by means of the following formula:

Number of elements:

$$M + (M+1) + (M+2) + \ldots + \left[\frac{M+N-1}{2}\right] \qquad (9\text{-}10)$$

Third type: Networks of the form: (M→M).

For this type of network, the same formula (9-10) applies.

Fourth type: Networks of the form: (M→2).

This network performs binary addition, but no carry read our wires are needed, because only the resulting digit is inspected.

The number of elements is determined by applying:

Number of elements:

$$M + (M+1) + (M+2) + \ldots + (M+P-1) \qquad (9\text{-}11)$$

9.8 THE NECESSARY NUMBER OF BINARY BITS

In order to attain the required discrimination between the positive and negative regions, it is necessary to determine the number of binary bits of the $\frac{1}{m_i}\left|\frac{x}{\Delta m_i}\right|_{m_i}$ quantities. At the same time, it is convenient to use the minimum

possible number of bits in order to save space in the tables, and to economize in equipment.

The necessary discrimination is 1/M. In the worst case, two representations may differ by only one bit in the last place. The weight of this bit is $1/2^k$, where k is the number of bits in use.

Therefore k is the number of bits that produces a suitable discrimination, and must satisfy the relation: $2^k \geqslant M$.

This number of bits insures an adequate discrimination in all cases, but when maximum simplification is desired, it is possible to obtain accurate sign determination even when several of the final bits are dropped. In that case the magnitudes are represented only approximately, but the sign is still correctly determined. The following considerations apply:

In a RNS, using 2 as one of the moduli, the representation for the first number of the second half of the range is always:

$$(1;\ 0;\ 0;\ldots.0)$$

Therefore, the summation of the quantities from the tables is always equal to 1/2 for this number, no matter how many digits there are in use. If the following numbers are also represented with less than k bits, their magnitude will be represented only as an approximation. But as long as the summation results in values of 1/2 or greater, the sign determination procedure will continue to operate correctly.

As an example, let us suppose k is being determined for the system (2; 3; 5). Here M = 30, therefore k must be 5, because $2^5 = 32$ which is greater than 30.

But it is possible to use only four bits in the table and still obtain correct sign determination. The following approximate representations are calculated using the table on page 150, but considering only four bits:

First half of the range	10	5/16	$< 1/2$
	11	5/16	
	12	6/16	
	13	6/16	
	14	7/16	
	15	8/16	
Second half of the range	16	8/16	$\geq 1/2$
	17	8/16	
	18	9/16	
	19	10/16	
	20	10/16	

9.9 SYSTEMATIC PROCEDURE FOR THE DEVELOPMENT OF THE NETWORK AND FOR THE DETERMINATION OF THE NUMBER OF ELEMENTS

The only data available are the set of moduli in the system. The following steps should be followed, as illustrated in the next example.

1. Draw the schematic of the network, as in Figure 21, with as many ▽ (indicating individual networks) as the number K of binary bits used in the tables of the $\frac{1}{m_i}\left|\frac{X}{\hat{m}_i}\right|_{m_i}$ quantities.

2. Determine the sequence of maximum carries from Table 8 for the number of moduli used in the system.

3. Write on the left side of the network the sequence of carries and on the right side the same numbers increased by one, indicating the number of wires.

4. Identify and separate the three regions in the general network.

5. For the first network, obtain the number of elements from Table 7, or apply equation (9-9).

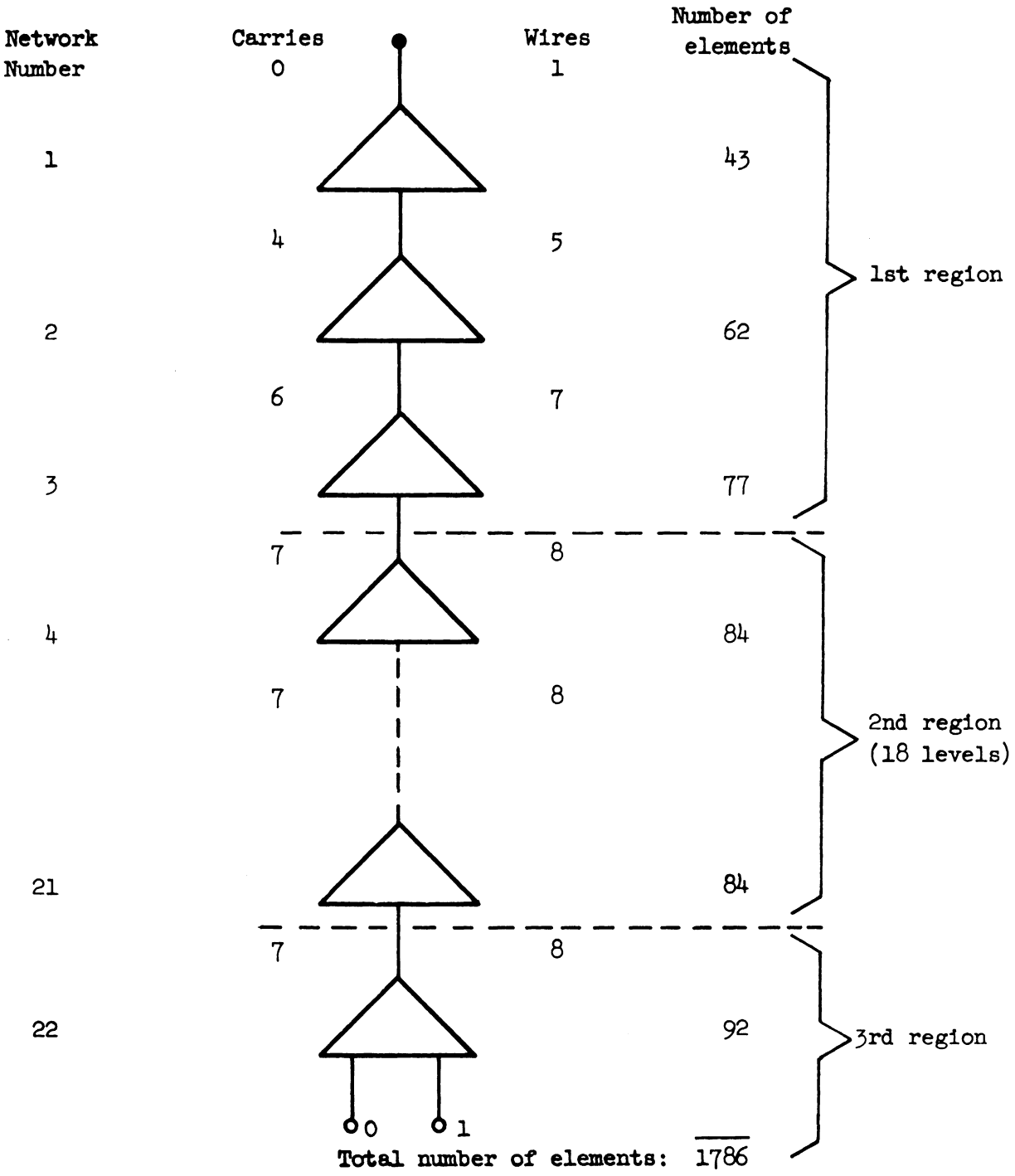


Figure 21. Determination of the number of elements in the carry network.

6. Calculate the number of elements for the networks in the rest of region 1 and 2, using equation (9-10).

7. Determine the number of elements for the network in the 3rd region using equation (9-11).

8. Add the partial results obtained in steps 5, 6, and 7. The number of terms to be added is equal to the number of individual networks.

Example: Given the system: 2, 3, 5, 7, 11, 13, 17, 23, draw a diagram of the general network and determine the number of elements.

The numbers on the left side refer to the steps of the preceding procedure.

1. Number of bits in the $\frac{1}{m_i}\left|\frac{x}{m_i}\right|_{m_i}$ quantities = K = 22.

2. Sequence of carries for n = 8: from the Table 7, we obtain: 4, 6, 7, 7, 7, 7.....

3. This sequence is written on the left side of the network, and on the right side the same quantities appear increased by one.

4. The three regions are separated by broken lines.

5. From the Table 7, for M = 5, the number of elements is found to be 43.

6. Applying equation (9-10), we obtain:

For the 2nd network: $5+6+7+8+9+10+11+[12/6] = 62$

For the 3rd network: $7+8+9+10+11+12+13+[14/2] = 77$

For the 4th up to the 21st network: $8+9+10+11+12+13+14+[15/2] = 84$

7. For the 22nd network: $8+9+10+11+12+13+14+15 = 92$.

8. Adding the partial results of 5, 6, and 7:

$$43 + 62 + 77 + (18 \times 84) + 92 = 1786$$

APPENDIX I

THEOREMS ON THE GENERATION OF SEQUENCES OF RELATIVELY PRIME NUMBERS

Theorem I. Given an ordered monotone sequence of positive and negative integers for which the difference between successive numbers is $\pm\delta$. If there exists one number in the sequence divisible by p_m then the maximum length L or a subsequence of numbers not divisible by p_m is $L = \frac{p_m}{(p_m,\delta)} - 1$; note $(p_m,\delta) = 1$ or p_m.

Proof: Given $p_m|\alpha$ then the next number divisible by p_m is β,

$$\beta = K_1 p_m + a\delta = K_2 p_m$$

$$(k_1-k_2)p_m + a\delta = 0$$

$$(k_1-k_2)\frac{p_m}{(p_m,\delta)} + a\frac{\delta}{(p_m,\delta)} = 0$$

Hence $a = \frac{p_m}{(p_m,\delta)}$ and there exists $\frac{p_m}{(p_m,\delta)}$ numbers between α and β.

Corollary: If one element of an ordered sequence of constant difference δ is divisible by p_m then the length of the maximum subsequence containing only one number divisible by p_m is $L_m = \frac{2p_m}{(p_m,\delta)} - 1$.

Theorem II. Given a subsequence of length $2p_m-1$ of the form $\alpha-\delta(p_m-1), \ldots,\alpha,\ldots,\alpha+\delta(p_m-1)$. There is at least one number divisible p_m.

Proof: $|\alpha\pm\delta K|_{p_m}$ assumes all values between 0 and p_m-1 as k goes from zero to p_m-1.

Corollary: If $p_m \nmid \delta$ in the subsequence of length $2p_m-1$ then two numbers of the subsequence are divisible by p_m.

Theorem III. A necessary condition for a relatively prime sequence of numbers of length $L_m = \frac{2p_m}{(p_m,\delta)} = 1$; $(p_m,\delta) = 1$ is that no number of the sequence is divisible by a prime less than p_m.

Proof: Assume $p_n|\alpha$, $p_n < p_m$; then it follows from Theorem I that the length of the subsequence including two numbers divisible by p_n is $L_{n_2} = \frac{p_n}{(p_n,\delta)} - 1$. The length of the subsequence including 3 numbers divisible by p_n is $L_{n_3} = \frac{2p_n}{(p_n,\delta)} + 1$. Both L_{n_2} and $L_{n_3} \leqslant L_m$

Lemma 1: Let $p_\ell > p_m$ divide one number of the subsequence of length $L = 2p_m-1$. p_ℓ never divides 3 numbers of the subsequence if $(p_\ell;\delta) = 1$.

Proof:

$$2p_\ell + 1 > 2p_m - 1$$

Lemma 2: Let $p_\ell > p_m$ divide one number of the subsequence of length $L = 2p_m-1$. The subsequence is partitioned into two parts $\alpha-\delta(p_m-1),\ldots,\alpha-\delta$ and $\alpha,\alpha+\delta,\ldots,\alpha+\delta(p_m-1)$, where $p_m|\alpha$. Neither part has two numbers divisible by p_ℓ if $(p_\ell,\delta) = 1$.

Proof:

$$p_\ell\delta > (p_m-1)\delta.$$

Theorem IV. If $p_\ell > p_m$ divides one number of the subsequence of length $L-2p_m-1$ then the necessary condition for the division of two numbers in the sequence by p_ℓ is

$$\frac{p_\ell}{(p_\ell,\delta)} + 2 \leqslant 2p_m$$

The sufficient condition is either

$$\frac{p_\ell}{(p_\ell,\delta)} + 1 \leqslant p_m$$

or if $(p_\ell,\delta) = 1$ then integers k_1 and k_2 must exist such that

$$|k_2\delta|_{p_\ell} = |\alpha|_{p_\ell} \qquad k_2 \leqslant p_m - 1$$

$$|k_1\delta|_{p_\ell} = p_\ell - |\alpha|_{p_\ell} \qquad k_1 \leqslant p_m - 1$$

Corollary 1: If $p_\ell > p_m$ divides one number of the subsequence of length $L-2p_m-1$ then a sufficient condition that only one number of the subsequence is divisible by p_ℓ is

$$\frac{p_\ell}{(p_\ell,\delta)} + 2 > 2p_m$$

Corollary 2: A sufficient condition that only one number of the subsequence of length $L = 2p_m-1$ is divisible by $p_\ell > p_m$ for the case $p_\ell+2 \leqslant 2p_m$; $(\delta,p_\ell) = (\delta;p_m) = 1$; is

$$|\alpha|_{p_\ell} = 0,\ |\delta|_{p_\ell},\ldots,|(p_\ell - p_m)\delta|_{p_\ell}$$

or

$$|\alpha|_{p_\ell} = |(p_\ell - 1)\delta|_{p_\ell},\ldots,|p_m\delta|_{p_\ell}$$

Theorem V. The necessary and sufficient conditions for a relatively prime sequence of length $L = p_m-1$ of the form $\alpha-\delta(p_m-1),\ldots,\alpha,\ldots,\alpha+\delta(p_m-1)$ are:

1. $(p_m,\delta) = 1$
2. $p_m|\alpha$
3. for all $p_i < p_m$, $p_i \nmid \alpha$ and $(p_i,\delta) = p_i$
4. for all $p_\ell > p_m$ satisfying $p_\ell + 2 \leqslant 2p_m$

$$|\delta|_{p_\ell} = 0,\ |\delta|_{p_\ell},\ldots,|(p_\ell - p_m)\delta|_{p_\ell}$$

or

$$|\alpha|_{p_\ell} = |(p_\ell - 1)\delta|_{p_\ell},\ldots,|p_m\delta|_{p_\ell}$$

Proof:

1. Corollary, Theorem I

2. Corollary, Theorem II

3. $p_i \nmid \alpha$ follows from Theorem III

   $(p_i,\delta) = p_i$ follows from the fact that

   $|\delta K|_{p_i}$ for $k = 0, 1,\ldots, p_m-1$

   assumes all values of p_i is δ is not restricted and hence for some k_1, $|\alpha \pm k_1\delta|_{p_i} = 0$. Theorem III states that if one k exists satisfying $|\alpha \pm k\delta|_{p_i} = 0$ than a second k also exists.

4. Corollary 2, Theorem IV.

Corollary: A relatively prime sequence meeting the conditions of Theorem V may be modified by the multiplication of one and only one number in the sequence by each of the p_i, $i < n$, without destroying the relatively prime relationship between the numbers in the sequence.

Example: $p_m = 5$, $\alpha = k5$, $\delta = 2.3 = 6$, $2p_m-2 = 8$ so $p_\ell = 7$ and $|\alpha|_7 = 0, 6, 5, 1, 2$. Thus, suitable α's are 25, 55, 5, 85, 65, and the resulting sequences of relatively prime numbers are:

11,	17,	23,	29,	35,	41,	47,	53,	59
31,	37,	43,	49,	55,	61,	67,	73,	79
-19,	-13,	-7,	-1,	5,	11,	17,	23,	29, etc.

A choice of α such that $|\alpha|_7 = 3$ or 4 or α such that $p_i|\alpha$, $i < n$ results in a sequence of relatively prime numbers shorter than 9, consider $\alpha = 25$.

-5, 1, 7, 13, 19, 25, 31, 37, 43, 49, 55

APPENDIX II

A MIXED BASE AND MODULAR COMPARISON

The following theorem specifies the conditions that must be satisfied for a number S to have identical mixed base and modular representations.

Theorem I. Let $p_1, p_2, \ldots, p_N$ be the bases of a modular number system of total modulus $p = p_1 p_2 p_3 \ldots p_N$ and let the modular representation of a number S be given by:

$$S = (a_1 a_2 a_3 \ldots a_N), \quad a_K \equiv S \bmod p_K \qquad 0 \leq a_k \leq p_K - 1$$

Let the mixed base representation of S be given by:

$$S = b_1 b_2 \ldots b_N$$

where

$$S = b_1 p_{N1} + b_2 p_{N2} + \ldots b_{N-1} p_N + b_N$$

and

$$p_{NK} = p_N p_{N-1} p_{N-2} \ldots p_{K+1}.$$

S has identical mixed base and modular representations if:

$$a_k = b_K, \quad K = 1, 2, \ldots, N$$

Then S has identical mixed base and modular representations if, and only if,

$$\frac{S - \left\lfloor \frac{S}{p_{NK}} \right\rfloor}{p_K} \quad \text{is an integer} \quad K = 1, 2, \ldots, N-1.$$

Define the quantity t_{NK}:

$$t_{NK} \equiv \frac{S-\left[\frac{S}{p_{NK}}\right]}{p_K} \qquad K = 1, 2,\ldots,N-1$$

Now since $a_K = b_K$ only when t_{NK} is an integer, the following theorem relates to the number of integer t_{NK}'s which appear as S goes through the integers 1, 2,...,p and K is fixed.

Theorem II. As S goes through the integers 1, 2,...,p, the number t_{NK}, given by:

$$t_{NK} = \frac{S\left[-\frac{S}{p_{NK}}\right]}{p_K}$$

assumes the value of all integers $1,2,\ldots,(p_1p_2p_3\ldots p_{K-1})(p_{NK}-1)$ at least once and at most twice.

Proof: Examine the quantity

$$A = S - \left[\frac{S}{p_{NK}}\right]$$

For

$$0 \leqslant S < p_{NK},\ A = S$$

For

$$p_{NK} \leqslant S < 2p_{NK},\ A = S-1$$

and in general

$$ap_{NK} \leqslant S < (a+1)p_{NK},\ A = S-a$$

It is easily seen from the above that as S goes through the integers 1, 2,... bp_{NK}, A goes through the integers 1, 2,...,bp_{NK}-b. Therefore, as S goes through the integers 1, 2,...p and since $p = (p_1p_2p_3\ldots p_K)(p_{NK})$, A goes

through the integers 1, 2,...,$(p_1p_2\cdots p_K)(p_{NK}-1)$.

Now $t_{NK} = A/p_K$ hence, it follows that as S goes through the integers 1, 2,...p, t_{NK} assumes integer values at least

$$\frac{(p_1p_2\cdots p_K)(p_{NK}-1)}{p_K} \text{ times}$$

or t_{NK} assumes every integer 1, 2,...,$(p_1p_2\cdots p_{K-1})(p_{NK}-1)$ at least once as S goes through the integers 1, 2,...p.

Now examine the case of two integers S_1 and S_2 where $1 \leqslant S_1 < S_2 \leqslant p$ to see under what conditions their respective t_{NK}'s are the same.

Let $1 \leqslant S_1 < S_2 \leqslant p$ where S_1 and S_2 are integers. If t_{NK} for S_1 equals t_{NK} for S_2 then

$$\frac{S_1 - \left[\frac{S_1}{p_{NK}}\right]}{p_K} = \frac{S_2 - \left[\frac{S_2}{p_{NK}}\right]}{p_K} \qquad (2\text{-}1)$$

$$S_1 - \left[\frac{S_1}{p_{NK}}\right] = S_2 - \left[\frac{S_2}{p_{NK}}\right]$$

Let

$$\left[\frac{S_1}{p_{NK}}\right] = a_1$$

and

$$\left[\frac{S_2}{p_{NK}}\right] = a_2$$

Then

$$S_1 = a_1p_{NK} + r_1, \qquad 0 \leqslant r_1 \leqslant p_{NK} - 1 \qquad (2\text{-}2)$$

$$S_2 = a_2p_{NK} + r_2, \qquad 0 \leqslant r_2 \leqslant p_{NK} - 1 \qquad (2\text{-}3)$$

Further more, since $S_1 < S_2$, this implies $a_1 \leqslant a_2$. Substituting a_1 and a_2 into equation (2-1) gives

$$S_1 - a_1 = S_2 - a_2$$

$$S_2 - S_1 = a_2 - a_1 \qquad (2\text{-}4)$$

Using equations (2-2) and (2-3) to develop an expression for S_2-S_1 gives

$$S_2 - S_1 = (a_2-a_1)p_{NK} + r_2 - r_1 \qquad (2\text{-}5)$$

Equating equation (2-4) and equation (2-5)

$$a_2 - a_1 = (a_2-a_1)p_{NK} + r_2 - r_1$$

$$a_2 - a_1 = \frac{r_2-r_1}{1-p_{NK}}$$

Now $a_2 \geqslant a_1$ implies $a_2-a_1 \geqslant 0$. Furthermore, since a_1 and a_2 are integers, a_2-a_1 is an integer. If all the bases are greater than one, $p_{NK} > 1$ and $1-p_{NK} < 0$. Thus $r_2-r_1 \leqslant 0$. Recalling the restriction on the range of r_1 and r_2 from equations (2-2) and (2-3) it is seen that in general $1-p_{NK} \leqslant r_2-r_1 \leqslant p_{NK}-1$. In this case r_2-r_1 must satisfy the additional conditions:

(1) $r_2 - r_1 \leqslant 0$

(2) $1 - p_{NK}$ must divide r_2-r_1. This leaves only two allowable choices of r_2-r_1:

Case (1) $r_2 - r_1 = 0$

Case (2) $r_2 - r_1 = 1 - p_{NK}$

Case (1): Let $r_2-r_1 = 0$, then $r_2 = r_1$ and $a_2-a_1 = \frac{0}{1-p_{NK}} = 0$ thus for Case (1) $r_2 = r_1$ and $a_2 = a_1$.

Now since

$$\left.\begin{aligned} S_1 &= a_1 p_{NK} + r_1 \\ S_2 &= a_2 p_{NK} + r_2 \end{aligned}\right\} \Longrightarrow S_1 = S_2$$

But $S_1 = S_2$ contradicts the original hypothesis that $S_1 < S_2$. Therefore,

$$r_2 - r_1 \neq 0.$$

Case (2): Let

$$r_2 - r_1 = 1 - p_{NK}$$

then

$$a_2 - a_1 = \frac{1-p_{NK}}{1-p_{NK}} = 1$$

By equation (2-4)

$$S_2 - S_1 = 1$$

Now if

$$r_2 - r_1 = 1 - p_{NK}$$

$$r_1 - r_2 = p_{NK} - 1$$

Recalling the limits on r_1 and r_2 it is seen that

$$r_1 = p_{NK} - 1$$

$$r_2 = 0$$

Hence,

$$S_1 = a_1 p_{NK} + p_{NK} - 1$$

$$S_2 = (a_1+1) p_{NK}$$

when t_{NK} for S_1 equals t_{NK} for S_2.

It has been established that two integers S_1 and S_2 will have the same t_{NK} for some $1 \leqslant K \leqslant N-1$ when S_2 is a multiple of p_{NK} and $S_1 = S_2-1$. Now assume that there is a third number, S_3, which has the same t_{NK} as S_1 and S_2. Since S is always and integer, it is clear that if such a S_3 exists it is either less than S_1 or greater than S_2. If S_3 is less than S_1 it can only be S_1-1. If $S_3 = S_1-1$, then to satisfy the conditions already established S_1 must be a multiple of p_{NK}. But $S_1 \equiv (p_{NK}-1) \bmod p_{NK}$ which is a contradiction. Furthermore, if $S_3 > S_2$, then $S_3 = S_2+1$ and S_3 must also be a multiple of p_{NK}. Now since S_2 is a multiple of p_{NK}, $S_3 = S_2+1$ can only be a multiple of p_{NK} if $p_{NK} = 1$. This contradicts the hypothesis that $p_{NK} > 1$. Hence, no such S_3 exists.

In summary, it has been shown that as S goes through the integers 1, 2,...,p, t_{NK} assumes integer values at least $(p_1p_2...p_{K-1})(p_{NK}-1)$ times and further that no more than two numbers have the same t_{NK}. Hence, the theorem is proved.

Define M_{NK} as the number of integer t_{NK} for some fixed K as S goes through the integers 1, 2,...,p. On the basis of the previous theorem we can say that

$$M_{NK} = (p_1p_2...p_{K-1})(p_{NK} - 1) + m_{NK}$$

where m_{NK} is the number of <u>integer</u> t_{NK} which appear twice as S goes through 1, 2,...,p.

It has been shown that t_{NK} will be the same for two numbers if they are successive and if p_{NK} divides the greater one. Note that t_{NK} in this case is not necessarily an integer. As S goes through 1, 2,...p there will be p/p_{NK} integers which are multiples of p_{NK}.

Now examine t_{NK} when S is some multiple c of p_{NK}.

$$t_{NK} = \frac{cp_{NK} - \left[\frac{cp_{NK}}{p_{NK}}\right]}{p_K} = \frac{c(p_{NK}-1)}{p_K}$$

It is clear that if $p_K | (p_{NK}-1)$ all t_{NK} for $S = cp_{NK}$ will be integers. Since there are $p/p_{NK} = p_1p_2\cdots p_K$ multiples of p_{NK}, $M_{NK} = p_1p_2\cdots p_K$ and $M_{NK} = (p_1p_2 \cdots p_{K-1})(p_{NK}-1)+(p_1p_2\cdots p_K)$, $M_{NK} = (p_1p_2\cdots p_{K-1})(p_{NK}+p_K-1)$, $p_k | (p_{NK}-1)$. For the case $p_K \nmid (p_{NK}-1)$, t_{NK} will be an integer only for those values of c which are divisible by p_K and there will be $\frac{p_1p_2\cdots p_K}{p_K}$ of these.

$$M_{NK} = (p_1p_2\cdots p_{K-1}), \quad p_K \nmid (p_{NK} - 1)$$

$$M_{NK} = (p_1p_2\cdots p_{K-1})(p_{NK} - 1) \div (p_1p_2\cdots p_{K-1})$$

$$= (p_1p_2\cdots p_{K-1})(p_{NK}), \quad p_K \nmid (p_{NK} - 1)$$

Given some set of bases $p_1p_2p_3\cdots p_N$, the set M_{N1}, M_{N2}, M_{N3},...M_{NN-1} can be calculated. The number of identical mixed base and modular representations of integers S such that $1 \leqslant S \leqslant p$ cannot exceed the minimum M_{NK}.

Example:

$p_1 = 3$, $p_2 = 5$, $p_3 = 7$, $p_4 = 11$

$p_{43} = 11$, $p_{42} = 7 \cdot 11 = 77$, $p_{41} = 77 \cdot 5 = 385$

$7 \nmid 11-1 \Longrightarrow M_{43} = 3 \cdot 5 \cdot 11 = \underline{165}$

$5 \nmid 77-1 \Longrightarrow M_{42} = 3 \cdot 77 = \underline{231}$

$3 | 385-1 \Longrightarrow M_{41} = 385+3-1 = \underline{387}$

min M_{NK} = 165. The actual number of identical elements = 27. Although the minimum M_{NK} does provide an upper bound for the maximum number of identical mixed base and modular representations, it is a poor estimate of the actual number. Furthermore, to determine the maximum number of identical elements for all the N: mixed base systems associated with N bases requires (N-1): computations. The M_{NK} function does give some insight into the problem of choosing the bases to achieve the maximum number of identical elements. Comparing the two expressions for M_{NK} seems to indicate that choosing the bases and their order so that $p_K | p_{NK}$-1 will lead to the greatest probability of attaining a high degree of correspondance between mixed base and modular representations in a system.

APPENDIX III

SIGN DETERMINATION BY A TWO-DIGIT REDUCTION PROCESS

INTRODUCTORY REMARKS

This report presents a method for determining the magnitude of a number from the decimal system which has been transformed into a residue number system. The residue representation does not permit a trivial recognition of magnitude, otherwise there would be no need of such a method.

This method determines which half of the system the number is in.

TERMINOLOGY

The bases are taken as primes, where p_1 is taken to be the number 2 while $p_i (i \neq 1)$ represents any arbitrary prime. The primes are to be arranged in increasing order, after arbitrary selection, such that

$$p_1 < p_2 < \cdots\cdots\cdots < p_i < p_{i+1} < \cdots\cdots\cdots < p_n \qquad (3\text{-}1)$$

The letter n will be taken to mean the total number of prime bases, and N will mean any number representable in the system. The letter a_i shall represent the residue of N with respect to p_i, and the range of N to be considered shall be

$$0 \leqslant N < M \qquad (3\text{-}2)$$

where

$$M = \prod_{i=1}^{n} p_i$$

In symbolic form, the number shall be represented by

$$\text{Bases:} \qquad p_1 p_2 p_3 \ldots\ldots\ldots\ldots p_n \tag{3-3}$$

$$N \equiv = a_1 a_2 a_3 \ldots\ldots\ldots\ldots a_n$$

Use will also be made of further terminology. The letter m shall indicate the product of two base primes, and to be orderly let

$$\begin{aligned} m_1 &= p_n p_{n-1} \\ m_2 &= p_{n-2} p_{n-3} \\ &\vdots \\ m_k &= p_4 p_3 \end{aligned} \tag{3-4}$$

The case under consideration is seen to be n an even number, with k thus being equal to (n/2)-1.

This orderly pairing of primes and assignment of subscripts to the m's is not essential and does not effect the generality of the result. It is merely an aid to simplification of symbolism.

The letter b_i shall represent the residue of N with respect to m_i, and s_i, t_i and d_j represent integers.

OPERATIONAL REMARKS

From the terminology section, a_i is the residue of N with respect to p_i. This may be written as:

$$N = s_i p_i + a_i \tag{3-5}$$

and, from the interpretation of b_i;

$$N = t_i m_i + b_i \tag{3-6}$$

By the nature of the residue system, certain operations are easily performed. If the representation has a zero in the ith digit position, division by p_i is simple and guarantees an integer result. Thus

$$\frac{N-a_i}{p_i} = s_i \qquad (3\text{-}7)$$

is easily performed, because $N-a_i$ does indeed have a zero in the ith position.

Similarly, the operation

$$\frac{N-b_i}{m_i} = t_i \qquad (3\text{-}8)$$

is easily performed, since m_i is the product of two base primes and $N-b_i$ has zero digits corresponding to these primes.

However, b_i is not as readily available as was a_i. But b_i may be found by equations (3-12) and (3-13). In the development of these equations, the subscripts on b, t and m have been changed to avoid confusion with p, d and a.

Let

$$m_o = p_i p_j$$

where

$$2 \leqslant i < j \leqslant n$$

and

$$0 \leqslant d_j \leqslant p_i - 1$$

By the definition of a and b

$$b_o \equiv a_i \bmod p_i$$

$$b_o \equiv a_j \bmod p_j \qquad (3\text{-}9)$$

These equations (3-9) may also be interpreted as follows

$$b_o = d_i p_i + a_i$$
$$b_o = d_j p_j + a_j \qquad (3\text{-}10)$$

Combining equations (3-10) yields

$$a_i - a_j = d_j p_j - d_i p_i$$

which may be also interpreted as

$$a_i - a_j \equiv d_j p_j \bmod p_i \qquad (3\text{-}11)$$

and thus the procedure for the determination of b_o, from equations (3-10) and (3-11) is:

$$d_j = \left| \frac{a_i - a_j}{p_j} \right|_{p_i} \qquad (3\text{-}12)$$

$$b_o = d_j p_j + a_j \qquad (3\text{-}13)$$

ADDITIONAL REMARKS

If care is taken in selection of the primes for m_i, a savings will result. For if, in equation (3-12), $p_j \equiv \pm 1 \bmod p_i$ the congruence relation is solved merely by subtracting a_j from a_i in the mod p_i portion of the system, a fast and simple step.

In connection with the forthcoming expansion, note that the machine step of equation (3-7) destroys the ith digit position. If the system is assumed capable of re-expanding s_i to reclaim the lost digit position, then a reduction to a uniformly-based number system can be made. The following

reduction to a non-uniformly-based number system does not require such an assumption.

REDUCTION TO NON-UNIFORMLY BASED SYSTEM

The number N will now be developed from its residue representation into a non-uniformly-based number. The starting point is the use of equations (3-6) and (3-8), and the following iterative procedure:

$$N = t_1 m_1 + b_1 \qquad t_1 = \frac{N - b_1}{m_1}$$

$$t_1 = t_2 m_2 + b_2 \qquad t_2 = \frac{t_1 - b_2}{m_2}$$

$$\vdots \qquad\qquad \vdots$$

$$t_{k-1} = t_k m_k + b_k \qquad t_k = \frac{t_{k-1} - b_k}{m_k}$$

$$t_k = \cfrac{\cfrac{\cfrac{\cfrac{N - b_1}{m_1} - b_2}{m_2} - b_3}{\vdots \qquad\qquad \vdots} - b_{k-1}}{m_{k-1}} - b_k \Bigg/ m_k \qquad (3\text{-}14)$$

Equation (3-14) thus shows the final result of the iterative process. By combining this equation it can be seen that

$$t_k = \frac{N - b_1 - m_1 b_2 - m_1 m_2 b_3 - \cdots - \prod_{i=1}^{k-1} m_i b_k}{\prod_{i=1}^{k} m_i} \qquad (3\text{-}15)$$

Then by solving (3-15) for N, the result is seen to be a non-uniformly-based number

$$N = t_k \prod_{i=1}^{k} m_i + b_k \prod_{i=1}^{k-1} m_i + \cdots + b_3 m_2 m_1 + b_2 m_1 + b_1 \qquad (3\text{-}16)$$

in which

$$0 \leqslant b_i \leqslant m_i - 1.$$

DEVELOPMENT OF THE FINAL STEP

There are exactly (p_1p_2)-1 occurrences of the case $b_i = m_i$-1 for $0 < N < M$, two of which will be helpful in establishing certain limits. The two cases are now to be examined, and they are:

$$(a) \quad N = \frac{M}{2} - 1,$$

and

$$(b) \quad N = M - 1.$$

In each case, add unity to the representation of (3-16), resulting in:

$$(a) \quad \frac{M}{2} = (1 + t_k^{(a)}) \prod_{i=1}^{k} m_i \tag{3-17}$$

$$(b) \quad M = (1 + t_k^{(b)}) \prod_{i=1}^{k} m_i \tag{3-18}$$

In the equations (3-17) and (3-18) the letters $t_k^{(a)}$ and $t_k^{(b)}$ refer to specific constants, respectively the most significant digit of $\frac{M}{2} - 1$ and M-1 in the non-uniformly-based system.

By equations (3-2) and (3-4)

$$\prod_{i=1}^{k} m_i = \prod_{i=3}^{n} p_i \tag{3-19}$$

$$M = \prod_{i=1}^{n} p_i$$

And, since $p_i = 2$ by definition

$$\frac{M}{2} = \prod_{i=2}^{n} \tag{3-20}$$

By combining equations (3-17--3-20)

$$t_k^{(a)} = p_2 - 1 (N = \frac{M}{2} - 1)$$

$$t_k^{(b)} = p_1 p_2 - 1 (N = M - 1) \tag{3-21}$$

It now follows from equations (3-21) and as a direct result of the carry properties of the non-uniformly-based number system of equation (3-16) that for

$$\begin{aligned} 0 \leqslant N < \frac{M}{2}, \qquad & 0 \leqslant t_k < p_2 \\ \frac{M}{2} \leqslant N < M, \qquad & p_2 \leqslant t_k < p_1 p_2 \end{aligned} \tag{3-22}$$

Since the orderly use of the primes in the formation of the m_i's was not essential to any of the steps involved up to now, as long as each prime is used once and only once in that formation except for p_1 and one other prime p_r, the equations become in the more general case

$$\begin{aligned} 0 \leqslant N < \frac{M}{2}, \qquad & 0 \leqslant t_k < p_r \\ \frac{M}{2} \leqslant N < M, \qquad & p_r \leqslant t_k < p_1 p_r \end{aligned} \tag{3-23}$$

The only difference resulting from a given selection of p_r will be in the weights assigned to the digit positions of the non-uniformly-based number system.

Each step of this procedure eliminates two digit positions from further

consideration. After k steps, upon the derivation of t_k, there are two digits remaining for its representation, a_1' and a_r'. The primes on these letters merely serve to underline the fact that they are not the a_1 and a_r of the number N.

A brief study of equations (3-23), bearing in mind the cyclic nature of a residue representation, shows that for

$$0 \leqslant t_k < p_r;\ a_1' \text{ is } \begin{matrix}\text{even}\\ \text{odd}\end{matrix} \text{ when } a_r' \text{ is } \begin{matrix}\text{even}\\ \text{odd}\end{matrix}$$

while

$$p_r \leqslant t_k < p_1 p_r;\ a_1' \text{ is } \begin{matrix}\text{even}\\ \text{odd}\end{matrix} \text{ when } a_r' \text{ is } \begin{matrix}\text{odd}\\ \text{even}\end{matrix}.$$

And therefore when

$$a_1' + a_r' \equiv N_o \bmod 2 \tag{3-24}$$

it is seen that when

$$N_o = \begin{cases} 0, & 0 \leqslant N < \frac{M}{2} \\ 1, & \frac{M}{2} \leqslant N < M \end{cases} \tag{3-25}$$

And it is for this reason that $p_1 = 2$ was saved for the last (k+1) step.

EXAMPLE

An example is now given to illustrate pairing the prime bases to satisfy the criteria of the additional remarks section.

For the bases (2, 3, 5, 7, 11, 13, 17, 19, 23, 29), pairs $m_1 = 29x7$; $m_2 = 23x11$; $m_3 = 13x3$; $m_4 = 19x5$; while $p_1 = 2$ and $p_r = p_7 = 17$.

The assignment of subscripts to the m's in this example is arbitrary. Each definite assignment leads to a separate number system.

CONCLUDING REMARKS

An answer to this particular magnitude question is guaranteed in K+1 steps, where for

$$\begin{aligned} &\text{n even:} \quad K+1 = \frac{n}{2} \\ &\text{n odd:} \quad K+1 = \frac{n+1}{2} \end{aligned} \tag{3-26}$$

In the latter case, the final step becomes merely an inspection of the a_1' digit with the conditions of equation (3-25) becoming

$$\text{If } a_1' = \begin{cases} 0, & 0 \leqslant N < \frac{M}{2} \\ 1, & \frac{M}{2} \leqslant N < M \end{cases}$$

But, while K+1 steps are required in general, the procedure may terminate earlier, depending upon the size of N. For when any

$$t_i = 0 (i \leqslant K), \quad 0 \leqslant N < \frac{M}{2}$$

since

$$t_i = 0 \text{ if } 0 \leqslant N < \prod_{n=1}^{i} m_j$$

Now a time estimate for the complete operation gives, for equation (3-8) 1 add time; while for equations (3-12) and (3-13); 2 add times; for a total of 3 add times per ith step $1 \leqslant i \leqslant K$. The final step for n even takes 1 add time, while for n odd does not require an addition, but an inspection. These

facts coupled with equations (3-26) give

$$\begin{aligned} &\text{n even:} \quad \left(\frac{n}{2} - 1\right)3 + 1 \quad \text{add times} \\ &\text{n odd:} \quad \left(\frac{n-1}{2}\right)3 \quad \text{add times} \end{aligned} \tag{3-27}$$

to guarantee an answer.

However, if only one digit is removed at once, by the use of equation (3-7) alone, only two add times per step are required for n-2 steps, and one add time for the final step (n both even and odd) and equations (3-27) become, for this case

$$\text{every n:} \quad (n - 1) \quad \text{add times.}$$

APPENDIX IV

COMMENTS ON PARITY CHECKING IN THE RNS

A first thought is to apply normal checking procedures to the Residue Number System. Any conventional method may be applied to the transmission of digits from one point to another. That is, simple parity, multiple parity, and Hamming checks may be applied by considering the binary coding to be a message, with no consideration given its semantic character. Nothing new or special is added merely because the message is a residue coding.

This discussion will be on arithmetic checking. The problem is essentially to find a function F(a) having the property

$$F(a) + F(b) = F(a + b).$$

The most obvious example of such a function is F(a) = a. This is interpreted as duplication of the addition circuitry. A second and more significant example is $F(a) \equiv a \bmod m$. If the moduli of the RNS are $m_1, m_2, \ldots m_n$ and $m = m_j$ for some j, then this function is not particularly useful. It is a duplication of one of the component sums and suffers from the same type of limitation as the identity function. Attention will be focused on the case where $m \neq m_j$ (for all j). Here if $m \nmid m_j$ (for all j), then F(a) is a function of all n components and, therefore, requires a conversion to the mixed base system. The carries involved make this system very unattractive. A "useful" check will either involve a conversion to the mixed base system or will require a separate check on each of the digits. This latter case will be discussed in considerable detail.

In Garner's[14] paper on Generalized Parity Checking, the dimished-base check is discussed, in which:

$$\sigma_i b_j^k \equiv \sigma_i \bmod (b-1)$$

Also the augmented-base check, in which:

$$\sigma_i b_j^k \equiv \sigma_i \bmod (b+1)$$

or

$$\equiv -\sigma_i \bmod (b+1)$$

Neither of these can be used when

$$b_j = m_j$$

for then k = 0 and these relations are trivial. However, were the components to be binary-coded, it would be possible to use the augmented-base check.

It is more intriging to consider a binary-coded base-3 system of components with the dimished-base check. Moduli up to 81 could then be employed with a maximum of 4 such binary groups, and up to 27 with 3 base-3 digits. Only the least significant bit of each group would be involved in the checking. Three or four digits could conveniently be handled with "carry-look-ahead," if not by purely combinational circuits. Consider the RNS with moduli $m_1^{j_1}$, $m_2^{j_2}$,...$m_n^{j_n}$.

If

$$x \equiv \alpha_i \bmod m_i^{j_i}$$

then

$$x = \alpha_i + km_i^{j_i}$$

and

$$x = \alpha_i + (km_i^{j_i-\ell})m_i^{\ell}, \text{ where } \ell < j_i$$

yielding

$$x \equiv \alpha_i \bmod m_i^{\ell}$$

Thus, the residue of x mod m_i^{ℓ} may be obtained from the residue of x mod $m_i^{j_i}$. From this it follows that such a RNS could be used to obtain the residue of x mod m_i as a function of the individual components. This derived residue could be used to implement a check. It will now be shown that the only moduli which would work in the above scheme are those which divide $m_i^{j_i}$.

If

$$x \equiv a_1 \bmod m$$

and

$$y \equiv b_1 \bmod m$$

then

$$x \div y \equiv a_1 \div b_1 \bmod m$$

and the retained residue is

$$\left\{\frac{a_1 \div b_1}{m}\right\}$$

Now

$$a_1 \equiv a_2 \bmod m'$$

$$b_1 \equiv b_2 \bmod m'$$

and

$$a_1 + b_1 \equiv a_2 + b_2 \bmod m'.$$

The residue of $a_2 \div b_2 \bmod m'$ equals the residue of $a_1 \div b_1 \bmod m'$. The residue produced in the check procedure is

$$\left\{\frac{a_1 + b_1}{m'}\right\}.$$

Choose for consideration an a_1 and a b_1 such that

$$a_1 + b_1 = m$$

Such a selection is possible because $a_1 + b_1$ will span the integers I

$$0 \leqslant I < 2(m-1)$$

If $m' \mid m$ then

$$\left[\frac{a_1 + b_1}{m'}\right] = k\left[\frac{a_1 + b_1}{m}\right]$$

and

$$\left\{\frac{a_1 + b_1}{m'}\right\} = \left\{\frac{a_1 + b_1}{m}\right\}$$

However, if $m' \nmid m$

$$\left\{\frac{a_1 + b_1}{m}\right\} = 0$$

and

$$\left\{\frac{a_1 + b_1}{m'}\right\} \neq 0.$$

The hypothesis has been demonstrated.

REFERENCES

1. Valach, M. Vznik Kodu A Ciselne Soustavy Zbytkovych Trid. Stroje Na Zpracovani Informaci, Sbornik III; 1955.

2. Svoboda, A. Rational Number Systems of Residual Classes. Stroje Na Zpracovani Informaci, Sbornik, 1957.

3. Svoboda, A. and Valach, M. Operatorove Obvody. Stroje Na Zpracovani Informaci, Sbornik III; 1955.

4. Garner, H. L. "The Residue Number System." IRE Trans. PGEC, Vol. EC-8, June 1959, p. 140-47.

5. Cheney, P. W. A Digital Correlator Based on the Residue Number System. Technical Document LMSD-702670, Lockheed Aircraft Corporation, Palo Alto, Calif., 1960.

6. McCoy, N. H. Rings and Ideals. Buffalo, New York: The Mathematical Association of America, 1948.

7. van der Waerden, B. L. Modern Algebra. New York: Frederick Ugar Publishing Company, Vol. I and II, 1950.

8. Hildebrand, F. B. Introduction to Numerical Analysis. McGraw-Hill Book Co.; 1956.

9. Cheney, P. W. Scaling by Deletion of Bases. Modular Arithmetic Note 10, Lockheed Aircraft Corporation, Palo Alto, Calif.; August 1960.

10. Herwitz, P. S. and Pomerene, J. H. The Harvest System. Proc. WJCC, pp. 23-32; May 1960.

11. Report BL-25, Harvard University Communications Laboratory.

12. Valach, M. and Svoboda, A. Operational Circuits. Institute of Mathematical Machines; Oct. 1954; p. 330.

13. WADC TR-59-472 Project 7062; July 1959.

14. Garner, H. L. Generalized Parity Checking. IRE Trans. PGEC, Vol. EC-7, Sept. 1958, pp. 209-13.