ON A FLEXIBLE IMPLEMENTATION OF DIGITAL COMPUTER ARITHMETIC

A. AVIZENSIS
Jet Propulsion Laboratory, California Institute of Technology, Pasadena, Calif., USA.

INTRODUCTION
Most implementations of computer arithmetic deal with numerical values which are represented as machine words of a fixed length. If the number of significant digits is less than the word length, non-significant zero digits are used to fill the remaining positions of the word; if it is greater than the word length, multiple-precision operations are programmed. This paper presents a method for a flexible implementation of arithmetical operations in a digital computer. Arithmetical operations are defined for any length (number of significant digits) of the operands. The result of the operation is generated sequentially, with the most significant digits appearing first. As arithmetical operation may be concluded when the required number of significant digits of the result has been generated. Furthermore, the time required by an addition in a parallel adder is constant for any number of digits. The length of the adder may be altered without changing the rules of implementation or the addition time; this permits a reorganization of the adder in case of a failure, or a reappropriation of adder units to suit a particular problem in a parallel system with several adders.

2. SIGN-SEGMENT NUMBER REPRESENTATIONS
The class of sign-digit number representations is uniquely suitable for a flexible implementation of arithmetical. Signed-digit representations are position number representations with a constant integer radix \( r > 3 \), in which the allowed values of the individual digits \( z_i \) are a sequence of \( r (r - 2) < q < 2r - 1 \) integers: \((-r, -r + 1, \ldots, 0, 1, \ldots, r - 1)\). The range of the minimum magnitude of a digit is \( \left[(r + 1) - q, r - 1\right) \) for odd integers, \( r > 3 \); and \( \left[q, r - 1\right) \) for even integers, \( r > 4 \). Both positive and negative digit values are allowed. The individual digits each contain sign information and therefore, a special sign digit is not required. For example, only one set of allowed values exists for radix \( r = 3 \) (values: \(-2, -1, 0, 1, 2\)), and for radix \( r = 4 \) (values: \(-3, -2, -1, 0, 1, 2, 3\)); for radix \( r = 10 \) there are four sets, from 13 values (\(-6 \) to \( \pm 6 \)) to 19 values (\(-9 \) to \( 9 \)).

Signed-digit representations are redundant, that is, each radix digit \( z_i \) may assume more than \( r \) different values. In a conventional (non-redundant) representation only \( r \) values of a digit \( 0, 1, \ldots, r - 1 \) are allowed. Signed-digit numbers have minimal redundancy (the digits assume \( r + 2 \) or \( r + 3 \) values) when

\[ s = s_{max} + r z + 1 \]

and they have maximal redundancy (the digits assume

\[ 2r - 1 \text{ or } 2r - 1 \text{ values} \]

when:

\[ s = s_{max} + r z - 1 \]

with:

\[ s_{max} = r^{l-1} - 1 \text{ or } s_{max} = r^{l-1} - 1. \]

The most important properties of signed-digit representations are the following:

1. The algebraic value of a number \( z \) is \( Z = \sum_{i=1}^{l} a_i r^{-i} \).

2. Algebraic value of \( Z = 0 \) if, and only if, all \( z_i = 0 \).

3. Sign of the algebraic value \( Z \) is the sign of the most significant non-zero digit.

4. To form the representation of \( -Z \), change the sign of every non-zero digit \( z_i \).

5. The addition and subtraction of digits may be totally-parallel: \( a_i = f(z_i, y_i, z_n, y_n, x_i) \) for all positions \( i \), where \( a_i \) are digits in the representation of the sum or difference \( S = Z \pm Y \).

There are no carry-propagation chains in totally-parallel addition (or subtraction), that is, any digit of the sum is a function of only two adjacent digits of the operands. Subtraction is performed as a change of sign followed by an addition. The time of one addition is independent of the length of the operands and is equal to the addition time of two digits. The procedure and block diagram of a totally-parallel adder are shown in Fig. 1. Here \( s \) is the transfer digit and may assume the values \( 0, 1, \) and \(-1\); \( w \) is the intermediate sum digit and may assume the sequence of values:

\[ (-w_{max}, \ldots, -1, 0, 1, \ldots, w_{max}). \]

For representations of minimal redundancy \( (s = s_{min}) \), we have \( w_{max} = s_{max} = -1 \); in all other cases \( (s > s_{min}) \) the value of \( w_{max} \) is restricted to the range \( -1 < w_{max} < s - 1 \).

To perform conversions between conventional (sign and magnitude) and signed-digit representations, we may treat each digit \( z_i \) of a conventional number as the sum of two digits of signed-digit numbers and apply the rules of totally-parallel addition \( (z_i = w_i + z_{n-i}) \). Conversely, we may consider a signed-digit number to be the sum of a positive and a negative number in conventional representations and obtain a single conventional number by adding them in a conventional adder.

The allowed range of the algebraic values

\[ Z = \sum_{i=1}^{l} a_i r^{-i} \]

for fixed-point numbers is required to cover the range \( 1 > Z > -1 \). An easily implementable method of over-
flow detection is also necessary. The two most significant
digit $z_0$ and $z_1$ are inspected to detect overflow.

Positive overflow occurs when:

$$ z_0 = 1; \text{ or } z_0 = -1 \text{ and } z_1 > 0. \tag{6} $$

Negative overflow occurs when:

$$ z_0 = -1; \text{ or } z_0 = 1 \text{ and } z_1 < 0. \tag{7} $$

where $z_0$ is the least significant digit (the number is
as $= 1$ digits long). In the range of values of $|Z|$
between these limits, overflow may or may not be
indicated, because signed-digit representations are re-
dundant and some algebraic values may be represented
in more than one way. For example, given radix $r = 10$
and $a = 6$ (minimal redundancy), we shall have a

For these overflow detection rules, no overflow will be
indicated for the algebraic values of

$$ |Z| < 1 + (1/r) - (a(a-1))r^{-2} = r^{-m}, $$

and overflow will always be indicated for

$$ |Z| > 1 + (a/(a-1))r^{-2} = r^{-m}, $$
certain overflow indication for $|Z| > (32/30)$ and no
overflow indication for $|Z| < (31/30)$.

In the following sections we employ minimal re-
dundancy representations, which require least storage
for the values of one digit and have the least magnitude
of $a (a = 15)$. The familiar radix 10, with 13 digits

Fig. 1. Totally-parallel addition and subtraction.

Fig. 2. Adder for $k$ digits.

Figures 1 and 2.
values (−6 to 6) is the preferred choice for practical applications. A detailed description of signed-digit representations and of conventional arithmetic with signed-digit numbers has been presented in an earlier paper).

3. IMPLEMENTATION OF ARITHMETIC

First we consider the addition (including subtraction) of two n digit-long signed-digit numbers (n ≥ 1), using an adder of k × 1 digit length, and beginning with the k most significant (highest weighted) digits of the operands. The addition consists of a sequence of k steps, where k is the least integer such that k ≥ \frac{b}{b}.

During each step, k digits of the sum are generated; the duration of one step is the unit time interval of the arithmetic unit, independent of \(a\). The most significant digits of the sum are generated first and overflow is immediately detectable. The detection or correction of errors can be independently implemented for each digit-adder circuit. Fig. 2 shows the diagram of an adder for \(k\) digits. The adder consists of \(k\) digit-circuits (\(A_0\) and \(B_0\) to \(A_{k-1}\) and \(B_{k-1}\)), and registers \(Z^\prime\), \(Z\)′, and \(Z\). The two most significant digits which are to be added during the next step (as inputs to \(A_0\)) are held in storage locations \(ZA\) and \(YS\) and serve as inputs to circuit \(A_0\) to form \(TA\). The circuits \(A_{k-1}\) and \(B_{k-1}\) and associated storage locations are used in multiplication only.

The adder is also employed to implement multiplication and division. In these operations, it is necessary to add a multiple \(\pm kY\) to the added \(\pm Y\) to the sum. Multiples \(\pm 2^k\) (0 ≤ \(k ≤ \log_2 m\)) of the digits \(y\) are added \(\pm y\) to be added to the digits \(z\). If the adder of \(k\) is enabled, \(\pm Y\) is added \(\pm z\) times. However, it is also possible to devise a circuit which forms the multiples \(\pm 2^k\) at once. The addition of \(\pm z\) is then completed in one additional time. One digit-adder with a multiple-forming circuit \(M\) for the digit \(z\), \(z\) is shown in Fig. 3. Two outputs are generated by the circuit \(M\): \((z, y) = z + y, z - y\). These outputs are added to the digit \(z\) in the circuits \(A_{k-1}\) and \(B_{k-1}\)

\(z + y + \pm = z + z + y + \pm = z - z + y + \pm = z - z - y + \pm \)

according to the rules of \(z\). The allowed values of \(y\) and \(z\) are chosen to satisfy the totally-parallel addition requirements. The only difference when the digit-adders of Fig. 3 are used to form \(z = \pm Y\) is that now \(z = z + z + y, z - z + y\) and the input \(z\) must be also available to form \(z\); during the next step the digit from \(Y = 0\) is shifted to \(Y\). The outputs \(A_{k-1}\) and \(B_{k-1}\) are employed only to form product digits.

\[P_i = P_{i-1} + r \cdot Y_{i-1}, i = 0, 1, 2, \ldots, m - 1\]

where \(Y_i\) is the multiplier, \(z_{i-1}\) and \(z_{i-1}\) are the digits of the multiplier \(X, P_i (0, 1, \ldots, m - 1)\) are partial products, \(P_i = 0, r = \pm Yi\) is the product. When the adder of Fig. 2 is used to perform multiplication, the \(\pm Y\) register holds the multiplicand, while the partial products are accumulated in the \(\pm Z\) register and its extension to the left, i.e., the \(\pm Z\) register. The multiplier \(X\) is held in a shift register \(\pm X\) and its digits are sensed sequentially, starting with the most significant digit \(x_0\). The product digits are immediately available when \(m\) is implemented as

\[P_i = P_{i-1} + Y_{i-1}\]

that is, one step of multiplication is performed as the addition of \(\pm Y\) followed by a left shift of the sum. All \((X, z_0) = 0, \ldots, 0)\) are partial products, \(P_i = 0, r = \pm Y_{i-1}\) is the product. When the adder of Fig. 2 is used to perform division, the \(\pm Y\) register holds the quotient digit \(q_{i-1}\) and its extension to the left, i.e., the \(\pm Y\) register. The divisor \(X\) is held in a shift register \(\pm X\) and its digits are sensed sequentially, starting with the most significant digit \(x_0\). The product digits are immediately available when \(m\) is implemented as

\[P_i = P_{i-1} + Y_{i-1}\]

Division is also implemented as a sequence of additions and shifts. The Robinson method of division \(r\) is most conveniently applicable to signed-digit numbers. In this method the representation of the quotient digit may be redundant, and then the selection of the quotient digit values may be based on the comparison of the approximate magnitudes of the divisor and the dividend, or a partial remainder. Given the radix \(r\), divisor \(Z\), and divisor \(Y\), one step of division is described by the algorithm:

\[R_i = kZ_{i+1} - X_i, i = 0, 1, \ldots, m - 1\]

where \(R_i = R_i, X_i, Y_i = 0, 1, \ldots, m - 1\) is the remainder, \(Z_i, X_i, Y_i = 0, 1, \ldots, m - 1\) is the allowed values of quotient digit \(q_i\); this requires that \(|R_i| < r \cdot K\|F\|\) must be satisfied, where \(K\) is the range \(R_i < K < 2K\|F\|\). We allow the quotient digit \(q_i = 0, 1, \ldots, m - 1\) as long as \(R_i < 2k\|F\|\). Then the quotient digit \(q_i = 0, 1, \ldots, m - 1\) as long as \(q_i = \pm Y_i\) for even values \(r_i < 2k, \) and \(s = 0, 1, \ldots, m - 1\) for odd values \(r_i < 2k\). Since the re-
presentation of the quotient is redundant, the value of \( q \) is selected on the basis of a comparison between

\[
|Y_{d,r} - \frac{1}{2} r - q| \quad \text{and} \quad |R_{d,r} - \frac{1}{2} r - q|.
\]

Here the digits \( y_d(0, 1, 2, 3) \) are the first four digits of the normalized divisor: \( |X_d| > |r| \); and the digits \( n(0, 1, 2, 3) \) are the first four digits of the partial remainder \( R_{d,r} \).

When the adder of fig. 3 is employed to perform division, the magnitude of \( X_d \) (field in "2") is diminished by repeated addition of \( \pm X_d \) (field in "0") until the condition \(|R_{d,r} \pm y X_d | < |Y_{d,r}| \) is detected by a comparison circuit; then \( \pm X_d \) is shifted left. If \( g \) additions are performed, \( |g| = g \); the sign of \( g \) is chosen to be such that the signs of \( R_{d,r} \) and of \( Y_d \) agree. If the digi-adder of fig. 3 is available, one step of division may be completed in one additiontime. We employ a total of 3 comparison circuits, in which the test \( |R_{d,r}| < |Y_d| + |g| \) is performed for the values of \( g \) which satisfies the test gives \( |g| = g \); if no value of \( g \) satisfies the test, \( |g| = 0 \). The sign of \( g \) is chosen as above.

4. Significant Digit Operations

In this section we consider an implementation of arithmetic in which the operations are performed with significant digits only. The propagation of error during a sequence of calculations may be more readily estimated in this case; furthermore, the least time will be taken by an arithmetic operation if a computation signal occurs as soon as the required number of significant digits of a sum, product or quotient has been generated.

To determine the number of significant digits in the number of significant digits in the input operands must be known. This information may be incorporated into the representations of numbers by a special digit \( \varphi \), designated as the space-zero. The number of significant digit positions at the right (low-significance) ends of the input operands and partial results are identified by this special digit \( \varphi \). The relative locations of the \( \varphi \) digits in all the operands supply the required information to conclude an arithmetical operation.

There exist two methods of applying the digit \( \varphi \). In the first method, addition rules

a) \( z + \varphi = (z + \varphi) \) for all values of \( z \) and \( \varphi \)

b) \( z + \varphi = \varphi \)

apply to the new digit \( \varphi \), and the sum of two signed-digit numbers is rounded off by truncation to the length of the number with fewer significant digits. If every allowed value of the digit \( \varphi \) occurs with the same probability, then the average error which is introduced by truncation is zero, and the round-off is without bias. The end of an addition is signalled by the detection of the digit \( \varphi \) as an input to the adder.

The digit \( \varphi \) may also be interpreted as a zero value:

\[ z \pm \varphi = z \pm 0 \]

for all values of \( z + \varphi \).

In this case round-off is avoided and the sum has the length of the number with more significant digits. End of addition is signalled by the detection of the digit \( \varphi \) as both inputs to one position of the adder. Rule (11) is applied in the implementation of multiplication and division.

The preceding development leads to a floating-point arithmetic in which the numbers are normalized and the \( \varphi \) digits indicate the precision of the fractional parts. The proposed implementation follows the rules of significant digit arithmetic which have been developed by Metropolitan and Attherhurst [4].

In a floating-point (radix \( r \)) number system, a numerical value \( Z^* \) is represented as a fractional part \( Z_2 \) and an integral exponent \( Z_1 \) such that the pair \( (Z_2, Z_1) \) represents \( Z^* = Z_2 \times Z_1 \). We require the fractional part to be normalized, since leading zeros are not significant. The fractional part \( Z_2 \) is in normal form when 2 conditions are satisfied: a) \( Z_2 \neq 0 \) and \( Z_2 = 1 \) or overflow is indicated when \( Z_2 = 0 \) is substituted for \( Z_2 \) and \( Z_2 \) is substituted for \( Z_1 \) in (6) and (7). According to this definition, the range of the normalized fractional part \( Z_2 \) is:

\[ r^{k_1 + 1} - (r(-1)) + r^k(r(-1)) < |Z_2| < r^{k_1 + 1} - r^k(r(-1)). \] (12)

To define the normal form of \( Z = 0 \), we add the rule that \( Z_2 = \varphi \) is an overflow indication when substituted for \( Z_2 \) in the test for normalization. Consequently, all fractional parts have at least \( 2 \) significant digits \( Z_2 \) and \( Z_2 \) and the fractional part \( Z_2 \) is 0 uniquely represented by \( Z_2 = 0, z_2 = 0 \) and \( Z_2 = \varphi \).

The execution of floating-point arithmetic now consists of three parts. The provisional exponent \( E_2 \) of the result \( Z_2^* \) must be calculated. In the addition process \( Z_2^* = Z_2^* \times Z_2^* \), we have \( E_2 = max \left( E_2, E_2 \right) \). The exponent difference \( d = |E_2 - E_2| \) must also be obtained.

In multiplication, \( Z_2^* = Z_2^* \times Z_2^* \) and we have \( E_2 = E_2 + E_2 \), while in division,

\( Z_2^* = Z_2^* \times Z_2^* \) and we have \( E_2 = E_2 - E_2 \).

The fractional part \( Z_2 \) of the result \( Z_2^* \) is calculated according to the rules of the preceding section, with completion signals provided by the \( \varphi \) digits in \( Z_2 \) and \( Z_2 \) (fractional parts of the operands). In addition, the most significant digit of \( Z_2 \) (where \( E_2 \geq E_2 \)) are added to zero, until digit \( z_2 \) of \( Z_2 \) is aligned with the most significant digit \( z_2 \) of \( Z_2 \); then addition proceeds as described above, and rule (10) applies to the \( \varphi \) digits. In multiplication and division, \( z_2 \) (the number of significant digits in \( Z_2 \)) is determined as \( z_2 = min (n_2, n_2) \), where \( n_2 \) and \( n_2 \) are the numbers of significant digits in \( Z_2 \) and \( Z_2 \) respectively. To determine \( n_2 \), it is sufficient to scan the operands serially (from the most significant digit) for the first occurrence of the digit \( \varphi \) or \( \varphi \). The \( \varphi \) digits are inspected before the execution of the \( t \)th step if one digit has the value \( \varphi \), \( n_2 \) steps have been completed.

The additions of (1) or (2) are performed according to rule (11) to avoid a truncation of the \( \varphi \) or \( \varphi \). Since the divided \( Z = 1 \), is altered by the first step of (9), it should be stored elsewhere for scanning, possibly in the multiplier register "X".
The result $Z_n$ may become denormalized, that is, the first three digits may indicate overflow or underflow (rounding zeros). These $Z_n$ must be normalized and the new exponent $E_n$ obtained by a correction of $E$. One significant digit may be gained or several may be lost during addition; appropriate corrections are also necessary during multiplication and division if denormalization occurs. 2.

5. CONCLUSION

A signed-digit number of any length consists of a positionally ordered and weighted array of complete one-digit-long numbers, carrying their own sign information. As a result, procedures exist for operations with variable length numbers and for significant digit arithmetic. The most significant digits of a result become available before the entire arithmetical operation is completed. Consequently, it is now possible to consider the organization of arithmetical nets, in which the digits of intermediate results are immediately processed further, and a numerical computation proceeds in a parallel asynchronous manner, at the maximum speed of the components (digit-adding, shift registers and control circuits) of the net. The organization of such arithmetical nets and the logical design of digit-adding and control circuits present interesting problems for further research.

6. ACKNOWLEDGMENT

This paper presents results of one phase of research carried out at the Jet Propulsion Laboratory, California Institute of Technology, under Contract NASw-6, sponsored by the National Aeronautics and Space Administration. A detailed study of feasible implementation will be published as a Technical Report of the Jet Propulsion Laboratory.

7. REFERENCES


ABSTRACTS

A method is presented for the implementation of arithmetical operations in a digital computer. Addition, subtraction, multiplication and division are performed with operands of variable multiple precision with respect to the length of the adder in an arithmetic unit. The result of the operation is generated sequentially, with the most significant digit appearing first.

A signed-digit number representation is employed to represent the operands and results. It is possible to add (or subtract) signed-digit numbers so that any digit of the sum (or difference) is the function of only two adjacent digits of the input operands. Consequently, carry-propagation chains of variable length do not occur during addition or subtraction.

When a special digit is employed to indicate the first non-significant position of each operand, an arithmetical operation can be terminated when all required significant digits of the result have been generated. It is expected that this implementation of arithmetic will be applicable in asynchronous computers, and in complex parallel systems, in which several arithmetic units are required to perform calculations synchronously at their own maximum speeds.

> В данном докладе определяется метод выполнения арифметических операций в цифровой вычислительной машине. Сложение, вычитание, умножение и деление произоходят с арифметическими числами переменной длины (т.е. с различной в разрядном отношении длиной операндов и длиной сумматора в арифметическом блоке). Результат операции (сумма, разность, произведение или частное) вычисляется последовательно, начиная с самых младших разрядов. Для представления входных данных и результата вычисления используется специально представленная вида входных разрядов. Обнаружение значимых разрядов представляет собой вычисление в определенных условиях следующих неравенств. Если это условие выполняется, то все разряды числовой части данных, которые не являются целыми в достаточно малом диапазоне, являются значимыми. Поскольку $g > 0$, представленные значения вида входных разрядов обеспечивают необходимость. Можно определить и вычислить число значимых разрядов того, что любые разряды из них были бы асинхронное произведение вычисления при максимальных скоростях этих блоков.
This communication presents a method for the realization of arithmetical operations on an electronic computer. Addition, subtraction, multiplication, and division are performed with operand values in the range between the smallest and largest representable numbers. The result of an operation is stored in the same range as the operand values, and the result is obtained by performing arithmetic operations on the operands. The method is simple and efficient, and it can be used in a wide range of applications.

In this paper, we present a method for the implementation of arithmetical operations in machine arithmetic. The method is based on the use of arithmetic operations on the operands. The result of an operation is obtained by performing arithmetic operations on the operands. The method is simple and efficient, and it can be used in a wide range of applications.

**DISCUSSION**

M. TAKAGI (Japan): In any redundant number representation, the determination of the sign of the sign-digit representation is difficult. The method presented in this paper avoids the determination of the sign by using the least significant non-zero digit. This is an important advantage of the method.

A. VAREZZA (U.S.A.). In the class of sign-digit representations, the determination of the sign is straightforward. The method presented in this paper avoids the determination of the sign by using the least significant non-zero digit. This is an important advantage of the method.

K. L. TISCH (Germany): A property of sign-digit representations is that, if two sign-digit values are equal, then the sum of the sign-digit values is equal to the sum of the sign-digit values.

C. L. TISCH (U.S.A.). Do you use the redundancy of the
representation to obtain error detection or correction of the arithmetic processes carried out?

A. Avizienis (U.S.A.). The original purpose of redundancy in this class of number representations was to limit the propagation of carries during addition. When there are no carry-propagation chains, the problem of error detection and correction is considerably simplified, since error detection or correction can be implemented independently for each digit-adder circuit. An investigation of this problem is now in progress.