(19) Europäisches Patentamt
European Patent Office
Office européen des brevets

(11) EP 1 728 149 B1

# (12) EUROPEAN PATENT SPECIFICATION

(45) Date of publication and mention of the grant of the patent:
**03.06.2009 Bulletin 2009/23**

(21) Application number: **04718386.8**

(22) Date of filing: **08.03.2004**

(51) Int Cl.:
***G06F 7/49*** *(2006.01)*

(86) International application number:
**PCT/IT2004/000116**

(87) International publication number:
**WO 2005/085991 (15.09.2005 Gazette 2005/37)**

## (54) COMPUTER SYSTEM FOR STORING INFINITE, INFINITESIMAL, AND FINITE QUANTITIES AND EXECUTING ARITHMETICAL OPERATIONS WITH THEM

COMPUTERSYSTEM ZUM SPEICHERN VON UNENDLICHEN, INFINITISIMALEN, UND ENDLICHEN GRÖSSEN UND DURCHFÜHREN VON ARITHMETISCHEN OPERATIONEN DAMIT

SYSTEME INFORMATIQUE PERMETTANT DE STOCKER DES QUANTITES INFINIES, INFINITESIMALES, ET FINIES ET EXECUTION D'OPERATIONS ARITHMETIQUES AVEC CELLES-CI

(84) Designated Contracting States:
**AT BE BG CH CY CZ DE DK EE ES FI FR GB GR HU IE IT LI LU MC NL PL PT RO SE SI SK TR**

(43) Date of publication of application:
**06.12.2006 Bulletin 2006/49**

(73) Proprietor: **YAROSLAV, Sergeev**
**I-87036 Arcavacata di Rende (IT)**

(72) Inventor: **YAROSLAV, Sergeev**
**I-87036 Arcavacata di Rende (IT)**

(74) Representative: **Leone, Mario**
**Studio Associato Leone & Spadaro**
**Viale Europa, 15**
**00144 Roma (IT)**

(56) References cited:

- **Y. D. SERGEYEV: "Arithmetic of Infinity" October 2003 (2003-10), EDIZIONI ORRIZONTI MERIDIONALI , COSENZA, ITALY , XP002300455 ISBN: 8889064013 paragraph [0003]**
- **COLLINS: "PM, A System for Polynomial Manipulation" COMMUNICATIONS OF THE ACM, vol. 9, no. 8, August 1966 (1966-08), pages 578-589, XP002303379**
- **ODA ET AL: "Polynomial Manipulation in AL-1E" REVIEW OF THE ELECTRICAL COMMUNICATION LABORATORIES, vol. 26, no. 7-8, August 1978 (1978-08), pages 1016-1026, XP008038021**



EP 1 728 149 B1

Printed by Jouve, 75001 PARIS (FR)

## Description

### FIELD OF THE INVENTION

**[0001]** The present invention describes a new type of computer which is able to execute arithmetical operations with infinite, infinitesimal, and finite numbers.

### BACKGROUND OF THE INVENTION

**[0002]** Problems related to the idea of infinity are among the most fundamental and have attracted the attention of the most brilliant thinkers throughout the whole history of humanity. Numerous trials have been done in order to evolve existing numeral systems and to include infinite and infinitesimal numbers in them. Such eminent researchers as Aristotle, Archimedes, Euclid, Eudoxus, Parmenides, Plato, Pythagoras, Zeno, Cantor, Dedekind, Descartes, Leibniz, Newton, Peano, Cohen, Frege, Gelfond, Gödel, Robinson, and Hilbert worked hard on these topics. To emphasize importance of the subject it is sufficient to mention that the Continuum Hypothesis related to infinity has been included by David Hilbert as the problem number one in his famous list of 23 unsolved mathematical problems that have influenced strongly development of the mathematics in the 20th century.

**[0003]** The point of view on infinity accepted nowadays is based on the ideas of Georg Cantor who has shown that there exist infinite sets having different number of elements. Particularly, he has shown that the infinite set of natural numbers, N, has less elements than the set, $\mathbb{R}$, of real numbers. There exist different ways to generalize arithmetic for finite numbers to the case of infinite numbers. However arithmetics developed for infinite numbers are quite different with respect to the arithmetic we are used to deal with (see examples in [1]). Moreover, very often they leave undetermined many operations where infinite numbers take part (for example, infinity minus infinity, infinity divided by infinity, sum of infinitely many items, etc.) or use representation of infinite numbers based on infinite sequences of finite numbers. These crucial difficulties did not allow to people to construct computers that would be able to work with infinite and infinitesimal numbers in the same manner as we are used to do with finite real numbers.

**[0004]** In fact, in modem computers, only arithmetical operations with finite numbers or with intervals having finite numbers as their limits (see, for example, [2]) are realized. Traditional real numbers can be represented in computer systems in various ways. Many of them use positional numeral systems with the finite radix *b*. Note that *numeral* is a symbol or group of symbols that represents a *number.* The difference between numerals and numbers is the same as the difference between words and the things they refer to. A *number* is a concept that a *numeral* expresses. The same number can be represented by different numerals. For example, the symbols '3', 'three', and 'III' are different numerals, but they all represent the same number.

**[0005]** In positional numeral systems fractional numbers are expressed by the record

$$(a_n a_{n-1} \ldots a_1 a_0 . a_{-1} a_{-2} \ldots a_{-(q-1)} a_{-q})_b \qquad (1)$$

where numerals $a_i$, $-q \le i \le n$, are called *digits,* belong to the alphabet {0,1, ..., *b*-1}, and the dot is used to separate the fractional part from the integer one. Thus, the numeral (1) is equal to the sum

$$a_n b^n + a_{n-1} b^{n-1} + \ldots + a_1 b^1 + a_0 b^0 + a_{-1} b^{-1} + \ldots + a_{-(q-1)} b^{-(q-1)} + a_{-q} b^{-q}. \qquad (2)$$

**[0006]** In modern computers, the radix $b = 2$ with the alphabet {0,1} is mainly used to represent numbers. There exist numerous ways to represent and to store numbers in computers. Particularly, a floating-point representation expresses a number in four parts: a sign, a mantissa, a radix, and an exponent. The sign is either a 1 or -1. The mantissa, always a positive number, holds the significant digits of the floating-point number. The exponent indicates the positive or negative power of the radix that the mantissa and sign should be multiplied by.

**[0007]** The above mentioned limits in the prior art are obviated by a computing system as defined in annexed claim 1.

## DESCRIPTION OF THE INVENTION

**[0008]** In this invention we describe a new type of computer - *infinity computer* - that is able to operate with infinite, infinitesimal, and finite numbers in such a way that it becomes possible to execute the usual arithmetical operations with all of them. For the new computer it is shown how the memory for storage of these numbers is organized and how the new arithmetic logic unit (NALU) executing arithmetical operations with them works.

**[0009]** In order to describe the infinity computer able to work with infinite, infinitesimal, and finite numbers we proceed as follows. First, we introduce a new positional system with infinite base allowing us to write down not only finite but infinite and infinitesimal numbers too. Second, we describe arithmetical operations for all of them. Third, we explain how the computer memory is organized to store these numbers. Fourth, we describe the NALU for operating with infinite, infinitesimal, and finite numbers.

**[0010]** The infinite radix of the new positional numeral system is introduced as an infinite number ① defined by the following axioms:

(i) For any finite natural number $n$ it follows $n < ①$.

(ii) The following relations link ① to identity elements 0 and 1

$$0 \cdot ① = ① \cdot 0 = 0, \quad ① - ① = 0, \quad \frac{①}{①} = 1, \quad ①^{0} = 1, \quad 1^{①} = 1. \qquad (3)$$

(iii) For any finite natural n the number $\frac{①}{n}$ is defined as the nth part of ①.

**[0011]** Examples of mathematical objects that can be taken as the radix ①: the number of elements of the set of natural numbers (see [1]), the number of elements of the set of integer numbers, the number of elements of the set of odd numbers. Note that the axioms (i) - (iii) added to the axioms of real numbers describe a class of new mathematical objects allowing to consider the process of operating with infinite numbers in a way different from the traditional one where $\infty + \infty = \infty$. Theories wheres $\infty + \infty \neq \infty$ are studied in the field of mathematics called non-standard analysis.

**[0012]** In order to fix our arithmetic, we suppose hereinafter without loss of generality that ① is the number of elements of the set of natural numbers. Then, due to the axiom (iii) the number $\frac{①}{n}$ is defined as the number of elements of the $n$th part of the set, N, of natural numbers where its nth parts are determined as the following sets

$$\mathbb{N}_n = \{k, k+n, k+2n, k+3n, \ldots\}, \qquad 1 \leq k \leq n.$$

It is important to emphasize that to introduce $\frac{①}{n}$ we do not try to count elements $k, k+n, k+2n, k+3n, \ldots$ In fact, we cannot do it since our possibilities to count are limited and, therefore, we are not able to count for infinity. In contrast, we postulate (in order to have the situation occurring in finite sets) that the infinite number of elements of the nth part of the set, i.e., $\frac{①}{n}$, is n times less than the number of elements of the whole set, i.e., than ①. Note also that, since $\frac{①}{n}$ has been introduced as the number of elements of a set, it is an integer number. This axiomatization means, for example, that the sets of even and odd numbers have $\frac{①}{2}$ elements each and the set of natural numbers, N, being the union of these two sets, has $\frac{①}{2} + \frac{①}{2} = ①$ elements.

**[0013]** The axioms (i)-(iii) are added to the axioms of real numbers and, therefore, it is possible to operate with ① as with a usual finite number. For example, it is possible to define the following infinite numbers that can also be interpreted in the terms of sets of finite numbers: ① - 2 is defined as the number of elements of the set of natural numbers, $\mathbb{N}$, from which two any numbers have been taken out; ① + 3 as the number of elements of the set $\mathbb{N} \cup \{a, b, c\}$ where numbers $a, b, c \notin \mathbb{N}$; and $①^2$ as the number of elements of the set $\mathbb{N} \times \mathbb{N}$.

**[0014]** Numbers obtained in such a way can be ordered. In our example we have ① - 2 < ① < ① + 3 < $①^2$. Let us show, for instance, that ① < $①^2$. We can write the difference

$$\square^2 - \square = \square(\square - 1). \qquad (4)$$

Due to (i), □ is greater than any finite natural number, therefore, □ > 1 and as a consequence □ -1 > 0. It follows from this inequality and (4) that the number $0^2$ - 0 is a positive number and, therefore, $\square^2 > \square$.

**[0015]** To express infinite and infinitesimal numbers we shall use records that are similar to (1) and (2) but have some peculiarities. In order to construct a number *C* in the new numeral positional system with base □ we subdivide *C* into groups corresponding to powers of □:

$$C = c_{p_m}\square^{p_m} + \ldots + c_{p_1}\square^{p_1} + c_{p_0}\square^{p_0} + c_{p_{-1}}\square^{p_{-1}} + \ldots + c_{p_{-k}}\square^{p_{-k}}. \qquad (5)$$

Then, the record

$$C = c_{p_m}\square^{p_m} \ldots c_{p_1}\square^{p_1} c_{p_0}\square^{p_0} c_{p_{-1}}\square^{p_{-1}} \ldots c_{p_{-k}}\square^{p_{-k}}. \qquad (6)$$

represents the number *C*, symbols $c_i$ are called *infinite grossdigits,* symbols $p_i$ are called *grosspowers.* The numbers $p_i$ are such that $p_i > 0$, $p_0 = 0$, $p_{-i} < 0$ and

$$p_m > p_{m-1} > \ldots p_2 > p_1 > p_{-1} > p_{-2} > \ldots p_{-(k-1)} > p_{-k}.$$

In the traditional record (1) there exists a convention that a digit $a_i$ shows how many powers $b^i$ are present in the number and the radix *b* is not written explicitly. In the record (6) we write $\square^{pi}$ explicitly because in the new numeral positional system the number i in general is not equal to the grosspower $p_i$. This gives possibility to write, for example, such numbers as $7\square^{244.5}3\square^{-32}$ where $p_1 = 244.5$, $p_{-1} = -32$.

**[0016]** Finite numbers in this new numeral system are represented by numerals having only one grosspower equal to zero. In fact, if we have a number *C* such that $m = k = 0$ in representation (6), then due to (3) we have $C = c_0\square^0 = c_0$. Thus, the number C in this case does not contain infinite units and is equal to the grossdigit $c_0$ which being a conventional finite number can be expressed in the form (1), (2) by any positional system with finite base *b* (or by another numeral system). It is important to emphasize that the grossdigit $c_0$ can be integer or fractional and can be expressed by a few symbols in contrast to the traditional record (1) where each digit is integer and is represented by one symbol from the alphabet {0,1, 2, ... , *b* -1 }. Thus, the grossdigit $c_0$ shows how many finite units and/or parts of the finite unit, $1 = \square^0$, there are in the number *C*. Grossdigits can be written in positional systems, in the *form* $\frac{p}{q}$ where *p* and *q* are integer numbers, or in any other finite numeral system.

**[0017]** Analogously, in the general case, all grossdigits $c_i$, $-k \le i \le m$, can be integer or fractional and expressed by many symbols. For example, the number $\frac{7}{3}\square^4\frac{84}{19}\square^{-3.1}$ has grossdigits $c_4 = \frac{7}{3}$ and $c_{-3.1} = \frac{84}{19}$. All grossdigits show how many corresponding units take part in the number *C* and it is not important whether this unit is finite or infinite.

**[0018]** Infinite numbers are written in this numeral system as numerals having grosspowers grater than zero, for example $7\square^{244.5}3\square^{-32}$ and $-2\square^{74}3\square^{0}37\square^{-2}11\square^{-15}$ are infinite numbers. In the following example the left-hand expression presents the way to write down infinite numbers and the right-hand shows how the value of the number is calculated:

$$15\square^{14}17.2045\square^{3}52.1\square^{-6} = 15\square^{14} + 17.2045\square^{3} + 52.1\square^{-6}.$$

If a grossdigit $c_{pi}$ is equal to 1 then we write $\square^{pi}$, instead of $1\square^{pi}$. Analogously, if power $\square^0$ is the lowest in a number then we often use simply the corresponding grossdigit $c_0$ without $\square^0$, for instance, we write $23\square^{14}5$ instead of $23\square^{14}5\square^0$ or 3 instead of $3\square^0$.

**[0019]** Numerals having only negative grosspowers represent infinitesimal numbers. The simplest number from this

group is $\square^{-1} = \frac{1}{\square}$ being the inverse element with respect to multiplication for $\square$:

$$\frac{1}{\square} \cdot \square = \square \cdot \frac{1}{\square} = 1. \qquad (7)$$

Note that all infinitesimals are not equal to zero. Inverse elements of more complex numbers including grosspowers of $\square$ are defined by a complete analogy. The following two numbers are examples of infinitesimals $30^{-32}$, $37\square^{-2}\text{-}11\square^{-15}$.

**[0020]** The above examples show how we can write down infinite numbers with all grossdigits being finite numbers. Let us see now how we can express a number including infinite grossdigits. The number

$$-14\square^2(0.5\square+3)\square^1(\square-4.5)\square^{-1} \qquad (8)$$

has $m=2, k=1$, and the following grossdigits

$$c_2 = -14, \quad c_1 = 0.5\square+3, \quad c_{-1} = \square-4.5,$$

where $c_2$ is finite and $c_1$, $c_{-1}$ are infinite. The record (8) is correct but not very elegant because the system base $\square$ appears in the expressions of grossdigits. In order to overcome this unpleasantness and to introduce a more simple structure of infinite numerals, we rewrite the number (8) in the explicit form (5)

$$-14\square^2(0.5\square+3)\square^1(\square-4.5)\square^{-1} = -14\square^2+(0.5\square+3)\square^1+(\square-4.5)\square^{-1}.$$

Then we open the parenthesis, collect the items having the same powers of $\square$ (taking into account that $\square\square^{-1} = \square^0$), and finally obtain

$$-14\square^2+(0.5\square+3)\square^1+(\square-4.5)\square^{-1} =$$

$$-14\square^2+0.5\square^2+3\square^1+\square\square^{-1}-4.5\square^{-1} =$$

$$-13.5\square^2+3\square^1+\square^0-4.5\square^{-1} = -13.5\square^2 3\square^1\square^0(-4.5)\square^{-1}. \qquad (9)$$

**[0021]** As can be seen from the record (9), there are no infinite grossdigits in it but negative grossdigits have appeared. Since the record (8) using infinite grossdigits (called hereinafter *record of the type 1*) is more cumbersome, we introduce the notion of *finite grossdigit* as a finite number $c_i$ expressed by a finite number of symbols in a numeral system and showing how many infinite units of the type $\square^{k_1}$, $-k \le i \le m$, should be added or subtracted in order to compose infinite numbers. The record (9) using finite grossdigits is called *record of the type 2* and, since it is more flexible than the record of the type 1, it will be mainly used hereinafter to express infinite numbers.

**[0022]** Let us now introduce arithmetical operations for infinite, infinitesimal, and finite numbers to be used in the infinity computer. The operation of *addition* of two given infinite numbers $A$ and $B$ returns as the result an infinite number $C$ constructed as follows (the operation of subtraction is a direct consequence of that of addition and is thus omitted). The numbers $A$, $B$, and their sum $C$ are represented in the record of the type 2:

$$A = \sum_{i=1}^{K} a_{k_i}\square^{k_i}, \qquad B = \sum_{j=1}^{M} b_{m_j}\square^{m_j}, \qquad C = \sum_{i=1}^{L} c_{l_i}\square^{l_i}. \qquad (10)$$

Then the result $C$ is constructed (see Example 1) by including in it all items $a_{k_i}\,\square^{k_i}$ from A such that $k_i \neq m_j$, $1 \le j \le M$, and all items $b_{mj}\,\square^{mj}$ from $B$ such that $m_j \neq k_i, 1 \le i \le K$. If in $A$ and $B$ there are items such that $k_i = m_j$ for some $i$ and $j$ then this grosspower $k_i$ is included in $C$ with the grossdigit $b_{k_i} + a_{k_i}$, i.e., as $(b_{k_i} + a_{k_i})\square^{k_i}$. It can be seen from this definition that the introduced operation enjoys the usual properties of commutativity and associativity due to definition of grossdigits and the fact that addition for each grosspower of □ is executed separately.

**[0023]** The operation of *multiplication* of two given infinite numbers $A$ and $B$ from (10) returns as the result the infinite number $C$ constructed as follows (see Example 2).

$$C = \sum_{j=1}^{M} C_j, \quad C_j = b_{m_j}\square^{m_j} \cdot A = \sum_{i=1}^{K} a_{k_i} b_{m_j} \square^{k_i + m_j}, \quad 1 \le j \le M. \qquad (11)$$

Similarly to addition, the introduced multiplication is commutative and associative. It is easy to show that the distributive property is also valid for these operations.

**[0024]** In the operation of *division* of a given infinite number $C$ by an infinite number $B$ we obtain an infinite number $A$ and a reminder $R$ that can be also equal to zero, i.e., $C = A \cdot B + R$.

**[0025]** The number $A$ is constructed as follows (see Example 3). The numbers $B$ and $C$ are represented in the form (10). The first grossdigit $a_{k_k}$ and the corresponding maximal exponent $k_K$ are established from the equalities

$$a_{k_K} = c_{l_L} / b_{m_M}, \quad k_K = l_L - m_M. \qquad (12)$$

Then the first partial reminder $R_1$ is calculated as

$$R_1 = C - a_{k_K}\square^{k_K} \cdot B. \qquad (13)$$

If $R_1 \neq 1$ 0 then the number $C$ is substituted by $R_1$ and the process is repeated by a complete analogy. The grossdigit $a_{kk\text{-}i}$, the corresponding grosspower $k_{K\text{-}i}$ and the partial reminder $R_{i+1}$ are computed by formulae (14) and (15) obtained from (12) and (13) as follows: $l_L$ and $c_{l_L}$ are substituted by the highest grosspower $n_i$ and the corresponding grossdigit $r_{n_i}$ of the partial reminder $R_i$ that in its turn substitutes $C$:

$$a_{k_{K-i}} = r_{n_i} / b_{m_M}, \quad k_{K-i} = n_i - m_M. \qquad (14)$$

$$R_{i+1} = R_i - a_{k_{K-i}}\square^{k_{K-i}} \cdot B, \quad i \ge 1. \qquad (15)$$

The process stops when a partial reminder equal to zero is found (this means that the final reminder $R = 0$) or when a required accuracy of the result is reached.

**[0026]** Now we are ready to present the infinity computer itself. It differs from traditional computers by its ability to store and elaborate infinite, infinitesimal, and finite numbers. We start by describing how its memory unit is organized.

**[0027]** The infinity computer has a special memory to store infinite, infinitesimal, and finite numbers where the following three rules of storage are fixed: (i) for the radix, □, of the infinite positional system; (ii) for grossdigits; (iii) for grosspowers. Hereinafter we suppose that infinity computer works with a base-two representation of grossdigits and grosspowers, however, the whole discussion holds for computing systems using any radix for storage of finite numbers.

**[0028]** The radix, □, of the infinite positional system is not stored. Its meaning (for example, assumption that □ is the number of elements of the set of even numbers or the number of elements of the set of natural numbers) is a convention for realization and usage of the infinity computer.

Every infinite, infinitesimal, or finite number

$$C = c_L \square^{p_L} + \ldots + c_1 \square^{p_1}, \qquad p_L > \ldots > p_1, \qquad (16)$$

is represented by a set of *2L* registers for storage grossdigits and grosspowers. Each ith part $c_i \square^{p_i}, 1 \le i \le L$, of the number *C* is represented by two registers (for example, with floating-point having single or higher precision): the first one for the grossdigit $c_i$ and the second for the grosspower $p_i$. Registers representing the parts $c_i \square^{p_i}$ of the number *C* are linked by pointers. If the number *C* contains *m* positive grosspowers, *k* negative grosspowers, and the grosspower equal to zero, then *L* registers used to store grossdigits are related to the *L* corresponding registers used to store grosspowers: *m* for the grossdigits corresponding to positive grosspowers, *k* corresponding to negative grosspowers, and one register corresponding to the grosspower equal to zero if it is present in the number.

**[0029]** If a grosspower $p_i$ is itself an infinite number *A*, then the register of this grosspower contains the pointer to the address where the number *A* is stored in the same manner as the number *C*. If the number *A* has again an infinite grosspower $A_1$, we again put in the register corresponding to this grosspower a pointer to the address where the number $A_1$ is stored. The obtained chains of pointers finish when all finite grosspowers are found. By using the terminology of data structures, we can say that as the result, the number *C* is represented by a tree with branches being pointers and leafs being grosspowers and grossdigits. An example of such a data structure is shown in Fig. 1.

**[0030]** The record of the type 2 for writing down infinite numbers (see definition at p. 8) ensures that all grossdigits are finite. In the case of the record of the type 1 (see definition at p. 7), not only grosspowers $p_i$ but also grossdigits $c_i$ can be infinite numbers. If this is the case and a grossdigit $c_i$ is an infinite number *B*, then the register of $c_i$ contains the pointer to the address where the number *B* is stored in the same manner as the number *C*.

**[0031]** Infinitesimal numbers are stored analogously. The only difference with infinite numbers is in the fact that infinitesimal numbers do not contain grosspowers greater or equal to zero. Finite numbers are stored in the same manner but they contain only one grossdigit corresponding to the grosspower equal to zero. Other grosspowers and grossdigits are absent in the record and, therefore, only two registers are sufficient to store finite numbers: the first one contains the grossdigit and the second register responsabile for the corresponding grosspower contains zero.

**[0032]** In traditional computers every number *r* is represented by a string of digits and the maximal length of this string allowed us by the computer gives boundaries for representation of the numbers, i.e., minimal and maximal absolute values, $\gamma$ and $\Gamma$, such that $\gamma \le |r| \le \Gamma$. In the infinity computers only finite grossdigits and grosspowers being leafs in the tree data structure are represented by strings of digits and, therefore, are also such that $\gamma \le |r| \le \Gamma$. However, if one needs to express, for example, a value greater than $\Gamma\square^{\Gamma}$, it is sufficient to substitute the finite value $\Gamma$ in the grosspower position by an infinite number, for instance, by $\square$. Thus, since $\square > \Gamma$, the resulting number $\Gamma\square^{\square} > \Gamma\square^{\Gamma}$. In general, when it is necessary to obtain greater (or smaller) quantities, the strings representing finite grossdigits or grosspowers can be substituted by pointers to new infinite (or infinitesimal) numbers.

**[0033]** In each concrete technical realization of the infinity computer we can represent a finite number *M* of infinite, infinitesimal, and finite numbers. The number *M* can be increased by enlarging the memory of the computer and by augmenting the length of the string of digits for representation of finite grossdigits and grosspowers.

**[0034]** In order to execute arithmetical operations with infinite, infinitesimal, and finite numbers the infinity computer uses the NALU which is different from that of classical computers. The NALU works with numbers of the form (16) by executing arithmetical operations introduced above. It consists of a control unit (CU), an internal memory unit (IMU) for storage intermediate results, and traditional ALUs (TALUs) working as low level tools inside the NALU and executing usual arithmetical operations only with finite numbers.

**[0035]** During every arithmetical operation the NALU reads numbers (16) from external memory to its IMU. Then, it decomposes the operands in finite grossdigits and the corresponding finite grosspowers using, if necessary, chains of pointers leading to final finite numbers (see the description of the memory unit above). Then the NALU calculates intermediate results and the final result according to the introduced operations: addition and subtraction by formula (10), multiplication by (10), (11), division by (10),(12)-(15). In order to execute these operations with infinite, infinitesimal, or finite numbers the NALU uses TALUs as follows.

**[0036]** The CU sends the operands (i.e., finite grossdigits and grosspowers) obtained after decomposition to TALUs and communicates to them which standard low level operations (i.e., operation of addition, subtraction, multiplication, or division with standard finite operands) should be executed with the sent operands in order to realize one of the required operations (10)-(15). When a TALU has calculated a result of the required operation, it sends it back to the CU which receives it and writes in the corresponding register of the IMU (i.e., as a grossdigit or as a grosspower) of the infinite number. This process of communications between the CU, IMU, and TALUs continues until the IMU will contain all grossdigits and grosspowers of the infinite result. Then, if the obtained number is the final result, the NALU stores it in the external memory, otherwise it is used as an intermediate result in further calculations.

**[0037]** The following two special situations can occur when TALUs calculate results of arithmetic operations with finite numbers. First, a TALU can obtain zero as the value of a grossdigit, i.e., $a_i = 0$ for a term $a_i \square^{p_i}$. Then, the CU does not

store the corresponding term $a_i \square^{p_i}$. Second, the underflow or overflow situations during calculating finite grossdigits or grosspowers hold. In this case the CU sends a message to the user describing the event and the user (or the compiler) makes a decision about elaboration of the situation.

**[0038]** The introduced new memory and the NALU can be used in different ways in computer systems with various architectures, for example, in the following forms: single processor computers, parallel computers, distributed computer systems, single processor quantum computers, parallel and distributed quantum computers.

## DESCRIPTION OF THE FIGURE

**[0039]** Figure 1 shows a tree data structure representing the following infinite number

$$C = 5\square^{4\square^{15}(-16)\square^{-1.5}}(-2)\square^{6}3\square^{7\square^{-2}}.$$

This number has 3 parts, i.e., $L = L(C) = 3$ in the form (16). The first grossdigit is 5 and it corresponds to the grosspower $4\square^{15}(-16)\square^{-15}$ being an infinite number which we call $A$. Since A is infinite, there is a pointer to its location in the memory shown in Figure 1 by the arrow $\rightarrow$. In its turn, $A$ consists of two parts, i.e., $L(A) = 2$, and all its grossdigits and grosspowers are finite. The symbol Ø shows that there are no other pointers in a number representation. The arrow links the first part of the number $C$ to the second one.

**[0040]** The second part, $-2\square^{6}$, of the number $C$ has both grossdigit and grosspower finite and they are stored in the corresponding registers. The third part, $3\square^{7\square^{-2}}$, of the number $C$ has the pointer to its infinite grosspower $7\square^{-2}$ which, in its turn, has both grossdigit and grosspower finite.

## EXAMPLES

**[0041]**

**1. Addition.** In order to simplify the presentation, the radix $b = 10$ is used for writing down grossdigits. We consider two infinite numbers $A$ and $B$ where

$$A = 16.5\square^{44.2}(-12)\square^{12}17\square^{0}1.17\square^{-3},$$

$$B = 23\square^{14}6.23\square^{3}10.1\square^{0}(-1.17)\square^{-3}11\square^{-43}.$$

Their sum $C$ is calculated as follows

$$C = A + B = 16.5\square^{44.2} + (-12)\square^{12} + 17\square^{0} + 1.17\square^{-3} +$$

$$23\square^{14} + 6.23\square^{3} + 10.1\square^{0} - 1.17\square^{-3} + 11\square^{-43} =$$

$$16.5\square^{44.2} + 23\square^{14} - 12\square^{12} + 6.23\square^{3} +$$

$$(17 + 10.1)\square^{0} + (1.17 - 1.17)\square^{-3} + 11\square^{-43} =$$

$$16.5\square^{44.2} + 23\square^{14} - 12\square^{12} + 6.23\square^{3} + 27.1\square^{0} + 11\square^{-43} =$$

$$16.5\square^{44.2}23\square^{14}(-12)\square^{12}6.23\square^{3}27.1\square^{0}11\square^{-43}.$$

**2. Multiplication.** We consider two infinite numbers

$$A = \square^{18}(-5)\square^{2}(-3)\square^{1}0.2, \qquad B = \square^{2}(-1)\square^{1}7\square^{-3}$$

and calculate the product $C = B \cdot A$. The first partial product $C_1$ is equal to

$$C_1 = 7\square^{-3} \cdot A = 7\square^{-3}(\square^{18} - 5\square^{2} - 3\square^{1} + 0.2) =$$

$$7\square^{15} - 35\square^{-1} - 21\square^{-2} + 1.4\square^{-3} = 7\square^{15}(-35)\square^{-1}(-21)\square^{-2}1.4\square^{-3}.$$

The other two partial products, $C_2$ and $C_3$, are computed analogously:

$$C_2 = -\square^{1} \cdot A = -\square^{1}(\square^{18} - 5\square^{2} - 3\square^{1} + 0.2) = -\square^{19}5\square^{3}3\square^{2}(-0.2)\square^{1},$$

$$C_3 = \square^{2} \cdot A = \square^{2}(\square^{18} - 5\square^{2} - 3\square^{1} + 0.2) = \square^{20}(-5)\square^{4}(-3)\square^{3}0.2\square^{2}.$$

Finally, by taking into account that grosspowers $\square^3$ and $\square^2$ belong to both $C_2$ and $C_3$ and, therefore, it is necessary to sum up the corresponding grossdigits, the product $C$ is equal (due to its length, the number $C$ is written in two lines) to

$$C = C_1 + C_2 + C_3 = \square^{20}(-1)\square^{19}7\square^{15}(-5)\square^{4}2\square^{3}3.2\square^{2}$$
$$(-0.2)\square^{1}(-35)\square^{-1}(-21)\square^{-2}1.4\square^{-3}.$$

**3. Division.** In the first example we divide the number $C = -10\square^{3}160^{0}42\square^{-3}$ by the number $B = 5\square^{3}7$. For these numbers we have

$$l_L = 3, \quad m_M = 3, \quad c_{l_L} = -10, \quad b_{m_M} = 5.$$

It follows immediately from (12) that $a_{k_k}\square^{k_k} = -2\square^{0}$. The first partial reminder $R_1$ is calculated as

$$R_1 = -10\square^{3}16\square^{0}42\square^{-3} - (-2\square^{0}) \cdot 5\square^{3}7 =$$

$$-10\square^{3}16\square^{0}42\square^{-3} + 10\square^{3}14\square^{0} = 30\square^{0}42\square^{-3}.$$

By a complete analogy we should construct $a_{k_{K-1}}\square^{k_{K-1}}$ by rewriting (12) for $R_1$. By doing so we obtain equalities

$$30 = a_{k_{K-1}} \cdot 5, \quad 0 = k_{K-1} + 3$$

and, as the result, $a_{k_{K-1}}\square^{k_{K-1}} = 6\square^{-3}$. The second partial reminder is

$$R_2 = R_1 - 6\square^{-3} \cdot 5\square^{3}7 = 30\square^{0}42\square^{-3} - 30\square^{0}42\square^{-3} = 0.$$

Thus, we can conclude that the reminder $R = R_2 = 0$ and the final result of division is $A = -2\square^0 6\square^{-3}$.
Let us now substitute the grossdigit 42 by 40 in $C$ and divide this new number $\tilde{C} = -10\square^3 16\square^0 400^{-3}$ by the same number $B = 5\square 37$. This operation gives us the same result $\tilde{A}_2 = A = -2\square^0 6\square^{-3}$ (where subscript 2 indicates that two partial reminders have been obtained) but with the reminder $\tilde{R} = \tilde{R}_2 = -2\square^{-3}$. Thus, we obtain $\tilde{C} = B \cdot \tilde{A}_2 + \tilde{R}_2$. If we want to continue the procedure of division, we obtain $\tilde{A}_3 = -2\square^0 6\square^{-3}(-0.4)\square^{-6}$ with the reminder $\tilde{R}_3 = 0.280\square^{-6}$. Naturally, it follows $\tilde{C} = B \cdot \tilde{A}_3 + \tilde{R}_3$. The process continues until a partial reminder $\tilde{R}_i = 0$ is found or when a required accuracy of the result will be reached.

## References

**[0042]**

[1] Ya.D. Sergeyev, Arithmetic of infinity, Edizioni Orizzonti Meridionali, CS, 2003.

[2] G.W. Walster, *Method and apparatus for representing arithmetic intervals with a computer system,* US Patent 6,658,443 B1, 2003.

## Claims

**1.** A computing system adapted to operate on data structures representing respective infinite, infinitesimal and/or finite operands (X), said system operating on data structures representing a first operand (A) and a second operand (B) in order to obtain a data structure representing a result operand (C), wherein each operand is expressed by a formula

$$X = x_L\square^{PL} + x_{L-1}\square^{PL-1} + \ldots + x_i\square^{Pi} + \ldots + x_1\square^{P1} .$$

where $\square$ represents an infinite radix;
$x_i$ are coefficients (grossdigits) of said radix;
$P_i$ are powers (grosspowers) of said radix, each of them corresponding to one of said $x_i$; and
said coefficients and said powers being infinite, infinitesimal and/or finite operands, where said computer system includes a memory (IMU) comprising:

at least a first series of registers;
at least a second series of registers;

wherein each data structure comprises a linked list of pairs of registers and each part $x_i\square^{Pi}$ of said operands is represented by a respective pair,
each first one of said respective pair of registers is a register from the first series, storing $P_i$ if $P_i$ is a finite operand and storing a pointer to a further data structure, representing $P_i$, if $P_i$ is an infinite or infinitesimal operand, and
each second one of said respective pair of registers is a register from the second series, storing $x_i$ if $x_i$ is a finite operand and storing a pointer to a further data structure, representing $x_i$, if $x_i$ is an infinite or infinitesimal operand, and where said computer system further includes a logical unit (NALU) comprising:

a control unit (CU) adapted to read said finite powers and finite coefficients from said data structures representing said first and second operands (A, B); and
at least one arithmetical logical unit (TALU) adapted to execute arithmetical operations among said finite powers and finite coefficients to obtain results and storing said results as respective finite powers and finite coefficients in the registers comprised in said data structure representing said result operand (C).

**2.** The computing system according to claim 1, so adapted that said second series of registers contains a number of coefficients (grossdigits) equal to the number of powers (grosspowers) stored in said first series of registers, the registers of said first series of registers for storing powers (grosspowers) being related to corresponding registers of said second series of registers for storing coefficients (grossdigits).

**3.** The computing system according to claim 2, so adapted that an operand comprising positive powers (grosspowers) in number equal to m, negative powers (grosspowers) in a number equal to k and a power equal to zero, is stored

in registers of the first series in a number equal to m for said positive powers (grosspowers), registers of the first series in a number equal to k for said negative powers (grosspowers), and a register of the first series for said power equal to zero, and the coefficients (grossdigits) of said operand are stored in corresponding registers of said second series in number equal to m for coefficient (grossdigits) corresponding to said positive powers (grosspowers), corresponding registers of said second series in number equal to k for coefficients (grossdigits) corresponding to said negative powers (grosspowers), and a register for a coefficient corresponding to said power equal to zero.

**4.** The computing system according to any of claim 1 to 3, so adapted that the operations between said operands (A, B) comprise the operations of addition, multiplication and division, and that:

• in said operation of addition said result (C) comprises all powers (grosspowers) of said operands and respective coefficient (grossdigits) if powers (grosspowers) from the first operand (A) differs from powers (grosspowers) from the second operand (B), or the additions of said coefficients (grossdigits) if said powers (grosspowers) from said first operand (A) are equal to said powers (grosspowers) from the second operand (B), the additions of said coefficients (grossdigits) being obtained by arithmetical additions of finite coefficients (grossdigits) performed by said arithmetical logical unit (TALU):
• in said operation of multiplication the resulting powers (grosspowers) are obtained by additions of every single power (grosspower) of said first operand (A) and every single power (grosspower) of said second operand (B), and said resulting coefficients (grossdigits) are obtained by multiplications of every single coefficient (grossdigit) of said first operand (A) by every single coefficient (grossdigit) of said second operand (B), said additions and multiplications being obtained by respective arithmetical additions and multiplications of finite powers (grosspowers) and finite coefficients (grossdigits) performed by said low-level arithmetical logical unit (TALU);
• in said operation of division a first resulting coefficient (grossdigit) $C_{I_L}$ of said result (C) is obtained by arithmetical division of a first coefficient (grossdigit) of said first operand (A) by a first coefficient (grossdigit) of said second operand (B) and a respective maximal power (grosspowers) $I_L$ of said results (C) is obtained by arithmetical subtraction of a maximal power (grosspowers) of said second operand (B) from a maximal power (grosspowers) of said first operand (A), so that a first partial reminder ($R_1$) is obtained as

$$R_1 = A - c_{I_L} \square^{I_L} - B$$

said process being repeated so that an $i+1^{th}$ resulting coefficient (grossdigit) is obtained by the division of an $i^{th}$ coefficient (grossdigit) of an $i^{th}$ partial reminder ($R_i$) by a first coefficient (grossdigit) of said second operand (B) and a generic $i+1^{th}$ resulting power (grosspower) of said result (C) is obtained by the subtraction of a maximal power (grosspower) of said second operand (B) from a maximal power (grosspower) of said $i^{th}$ partial reminder ($R_i$), said process being repeated, until said partial reminder ($R_i$) is equal to zero or a required accuracy of the result (C) is reached, said subtractions and divisions being obtained by respective arithmetical subtractions and divisions of finite powers (grosspowers) and finite coefficients (grossdigits) performed by said arithmetical logical unit (TALU).

**5.** The computing system according to any of claim 1 to 4, so adapted that said radix is not stored during the operations of said logical units (NALU, TALU) and said storage memories.

**6.** The computing system according to any of claim 1 to 5, so adapted that said finite coefficients (grossdigits) and finite powers (grosspowers) are represented in a base-two representation.

**Patentansprüche**

**1.** Computersystem für die Durchführung von Operationen mit Datenstrukturen, die infinite, infinitesimale und/oder finite Operanden (X) repräsentieren, wobei dieses System mit Datenstrukturen arbeitet, die einen ersten Operanden (A) und einen zweiten Operanden (B) haben, um eine Datenstruktur zu erhalten, die einen Ergebnisoperanden (C) repräsentiert, wobei jeder Operand ausgedrückt wird durch die Formel

$$X = x_L \square^{p_L} + x_{L-1} \square^{p_{L-1}} + \ldots + x_i \square^{p_i} + \ldots + x_1 \square^{p_1},$$

in der

□ eine infinite Basis darstellt,
$x_i$ Koeffizienten (grossdigits) dieser Basis sind,
Pi Potenzen (grosspowers) dieser Basis sind, wobei jede von diesen einem $x_i$ entspricht, und wobei diese Koeffizienten und Potenzen infinite, infinitesimale und/oder finite Operanden sind, wobei das Computersystem einen Speicher (IMU) mit
wenigstens einer ersten Reihe von Registern und
wenigstens einer zweiten Reihe von Registern hat,

wobei jede Datenstruktur eine verknüpfte Liste von Registerpaaren umfasst und jeder Teil $x_i \square^{Pi}$ der Operanden durch ein entsprechendes Paar repräsentiert wird,
wobei jedes erste Paar des entsprechenden Registerpaares ein Register aus der ersten Reihe ist, in der Pi gespeichert ist, wenn Pi ein finiter Operand ist, und in der ein Zeiger (pointer) zu einer weiteren, Pi repräsentierenden Datenstruktur gespeichert ist, wenn Pi ein infiniter oder infinitesimale Operand ist,
wobei jedes zweite Paar des entsprechenden Registerpaares ein Register aus der zweiten Reihe ist, in der $x_i$ gespeichert ist, wenn $x_i$ ein finiter Operand ist, und in der ein Zeiger (pointer) zu einer weiteren, $x_i$ repräsentierenden Datenstruktur gespeichert ist, wenn $x_i$ ein infiniter oder infinitesimaler Operand ist,
und wobei das Computersystem außerdem eine logische Einheit (NALU) hat, umfassend:

eine Steuereinheit (CU), die so ausgelegt ist, dass sie die genannten finiten Potenzen und finiten Koeffizienten aus den Datenstrukturen liest, welche die ersten und zweiten Operanden (A, B) repräsentieren, und
wenigstens eine arithmetische Logikeinheit (TALU), die so ausgelegt ist, dass sie mit den finiten Potenzen und den finiten Koeffizienten arithmetische Operationen durchführt, um Ergebnisse zu erzeugen und diese Ergebnisse als entsprechende finite Potenzen und finite Koeffizienten in den Registern zu speichern, die in den Datenstrukturen enthalten sind, welche den Ergebnisoperanden (C) repräsentieren.

**2.** Computersystem nach Anspruch 1, das so ausgelegt ist, dass die zweite Reihe der Register eine Anzahl von Koeffizienten (grossdigits) umfasst, die gleich der Anzahl von Potenzen (grosspowers) ist, die in der ersten Reihe der Register gespeichert sind, wobei die Register der ersten Registerreihe für die Speicherung der Potenzen (grosspowers) in Beziehung stehen zu entsprechenden Registern der zweiten Registerreihe für die Speicherung der Koeffizienten (grossdigits).

**3.** Computersystem nach Anspruch 2, das so ausgelegt ist, dass ein Operand, der positive Potenzen (grosspowers) mit einer Anzahl m, negative Potenzen (grosspower) mit einer Anzahl k und eine Potenz Null hat, gespeichert wird in Registern der ersten Reihe mit einer Zahl gleich m für die positiven Potenzen (grosspowers), in Registern der ersten Reihe mit einer Zahl gleich k für die negativen Potenzen (grosspowers) und in einem Register der ersten Reihe für die Potenz Null, während die Koeffizienten (grossdigits) des Operanden gespeichert werden in entsprechenden Registern der zweiten Reihe mit einer Zahl m für die Koeffizienten (grossdigits), die den positiven Potenzen (grosspowers) entsprechen, in entsprechenden Registern der zweiten Reihe mit einer Zahl k für die Koeffizienten (grossdigits), die den positiven Potenzen (grosspowers) entspricht, in entsprechenden Registern der zweiten Reihe mit einer Zahl k für die Koeffizienten (grossdigits), die den negativen Potenzen (grossdigits) entsprechen, und in ein Register für einen Koeffizienten, der der Potenz Null entspricht.

**4.** Computersystem nach einem der Ansprüche 1 bis 3, das so ausgelegt ist, dass die Operationen zwischen den Operanden (A, B) die Operationen Addition, Multiplikation und Division umfassen, und dass

• bei der Additionsoperation das Ergebnis (C) alle Potenzen (grosspowers) der Operanden und entsprechenden Koeffizienten (grossdigits), wenn die Potenzen (grosspowers) aus dem ersten Operanden (A) verschieden von den Potenzen (grosspowers) aus dem zweiten Operanden (B) sind, oder die Addition der Koeffizienten (grossdigits) enthält, wenn die Potenzen (grosspowers) aus dem ersten Operanden (A) gleich den Potenzen (grosspowers) aus dem zweiter Operanden (B) sind, wobei die Addition dieser Koeffizienten (grossdigits) erhalten wird durch arithmetische Additionen der finiten Koeffizienten (grossdigits), die von der arithmetischen Logikeinheit (TALU) ermittelt werden;
• bei der Multiplikationsoperation die resultierenden Potenzen (grosspowers) erhalten werden durch Additionen jeder einzelnen Potenz (grosspower) aus dem ersten Operanden (A) und jeder einzelnen Potenz (grosspower) aus dem zweiten Operanden (B) und die resultierenden Koeffizienten (grossdigits) erhalten werden durch Multiplikation jedes einzelnen Koeffizienten (grossdigits) des ersten Operanden (A) mit jedem einzelnen Koef-

fizienten (grossdigit) des zweiten Operanden (B), wobei diese Additionen und Multiplikationen durch arithmetische Additionen bzw. Multiplikationen der finiten Potenzen (grosspowers) und der finiten Koeffizienten (grossdigits), die von der arithmetischen Logikeinheit (TALU) ermittelt werden;

• bei der Divisionsoperation ein erster resultierender Koeffizient (grossdigit) $C_{IL}$ des Ergebnisses (C) erhalten wird durch arithmetische Division eines ersten Koeffizienten (grossdigit) des ersten Operanden (A) durch einen ersten Koeffizienten (grossdigit) des zweiten Operanden (B) und eine entsprechende maximale Potenz (grospower) $I_L$ des Ergebnisses (C) erhalten wird durch arithmetische Subtraktion einer maximalen Potenz (grospower) des zweiten Operanden (B) von einer maximalen Potenz (grosspower) des ersten Operanden (A), so dass ein ersten Teilrest ($R_1$) erhalten wird als

$$R_1 = A\text{-}c_{IL} \square^{IL} \cdot B,$$

wobei dieser Prozess wiederholt wird, so dass ein i + 1 resultierender Koeffizient (grossdigit) erhalten wird mittels Division des i-ten Koeffizienten (grossdigit) eines i-ten Teilrestes ($R_i$) durch einen ersten Koeffizienten (grossdigit) des zweiten Operanden (B) und eine resultierende allgemeine i + 1 Potenz (grosspower) des Ergebnisses (C) erhalten wird durch Subtraktion einer maximalen Potenz (grosspower) der zweiten Operanden (B) von einer maximalen Potenz (grosspower) des i-ten Teilrestes ($R_i$), wobei dieser Prozess wiederholt wird, bis der Teilrest ($R_i$) Null ist oder eine geforderte Genauigkeit des Ergebnisses (C) erreicht ist, wobei die Subtraktionen und Divisionen erhalten werden durch arithmetische Subtraktionen bzw. Divisionen der finiten Potenzen (grosspowers) und finiten Koeffizienten (grossdigits), die von der arithmetischen Logikeinheit (TALU) ermittelt werden.

**5.** Computersystem nach einem der Ansprüche 1 bis 4, das so ausgelegt ist, dass die Basis während der Operationen durch die arithmetischen Logikeinheiten (NALU, TALU) und der Speicher nicht gespeichert werden.

**6.** Computersystem nach einem der Ansprüche 1 bis 5, das so ausgelegt ist, dass die finiten Koeffizienten (grossdigits) und die finiten Potenzen (grosspowers) in einer Widergabe mit Basis 2 repräsentiert werden.

## Revendications

**1.** Système informatique apte à fonctionner sur des structures de données représentant des opérandes infinis, infinitésimaux et/ou finis respectifs (X), ledit système fonctionnant sur des structures de données représentant un premier opérande (A) et un deuxième opérande (B) pour obtenir une structure de données représentant un opérande de résultat (C), dans lequel chaque opérande est exprimé par une formule :

$$X = x_L \square^{PL} + x_{L-1} \square^{PL-1} + \ldots + x_i \square^{Pi} + \ldots + x_1 \square^{P1}$$

où □ représente une base infinie ;
$x_i$ sont des coefficients (chiffres bruts) de ladite base ;
$P_i$ sont des puissances (puissances brutes) de ladite base, chacune d'elles correspondant à l'un desdits $x_i$ ; et lesdits coefficients et lesdites puissances sont des opérandes infinis, infinitésimaux et/ou finis,
dans lequel ledit système informatique comprend une mémoire (IMU) comprenant :

au moins une première série de registres ;
au moins une deuxième série de registres ;

dans lequel chaque structure de données comprend une liste liée de paires de registres et chaque partie $x_2 \square^{pi}$ desdits opérandes est représentée par une paire respective,
chaque premier de ladite paire respective de registres est un registre de la première série, stockant $P_i$, si $P_i$ est un opérande fini et stockant un pointeur vers une autre structure de données, représentant $P_i$, si $P_i$ est un opérande infini ou infinitésimal, et
chaque deuxième de ladite paire respective de registres est un registre de la deuxième série stockant $x_i$ si $x_i$ est un opérande fini et stockant un pointeur vers une autre structure de données, représentant $x_i$ si $x_i$ est un opérande infini ou infinitésimal,

et dans lequel ledit système informatique comprend en outre une unité logique (NALU) comprenant :

une unité de commande (CU) apte à lire lesdites puissances finies et lesdits coefficients finis desdites structures de données représentant lesdits premier et deuxième opérandes (A, B) ; et
au moins une unité logique arithmétique (TALU) apte à exécuter des opérations arithmétiques parmi lesdites puissances finies et lesdits coefficients finis pour obtenir des résultats et stocker lesdits résultats en tant que puissances finies respectives et coefficients finis respectifs dans les registres compris dans ladite structure de données représentant ledit opérande de résultat (C).

**2.** Système informatique selon la revendication 1, apte à ce que ladite deuxième série de registres contienne un nombre de coefficients (chiffres bruts) égal au nombre de puissances (puissances brutes) stockées dans ladite première série de registres, les registres de ladite première série de registres pour stocker des puissances (puissances brutes) étant liés à des registres correspondants de ladite deuxième série de registres pour stocker des coefficients (chiffres bruts).

**3.** Système informatique selon la revendication 2, apte à ce qu'un opérande comprenant des puissances positives (puissances brutes) d'un nombre égal à m, des puissances négatives (puissances brutes) d'un nombre égal à k et une puissance égale à zéro, soit stocké dans des registres de la première série d'un nombre égal à m pour lesdites puissances positives (puissances brutes), des registres de la première série d' un nombre égal à k pour lesdites puissances négatives (puissances brutes), et un registre de la première série pour ladite puissance égale à zéro, et les coefficients (chiffres bruts) dudit opérande sont stockés dans des registres correspondants de ladite deuxième série d'un nombre égal à m pour les coefficients (chiffres bruts) correspondant aux dites puissances positives (puissances brutes), des registres correspondants de ladite deuxième série d'un nombre égal à k pour les coefficients (chiffres bruts) correspondant aux dites puissances négatives (puissances brutes), et un registre pour un coefficient correspondant à ladite puissance égale à zéro.

**4.** Système informatique selon l'une quelconque des revendications 1 à 3, apte à ce que les opérations entre lesdits opérandes (A, B) comprennent les opérations d'addition, de multiplication et de division, et à ce que :

- dans ladite opération d'addition, ledit résultat (C) comprend toutes les puissances (puissances brutes) desdits opérandes et les coefficients respectifs (chiffres bruts) si des puissances (puissances brutes) du premier opérande (A) diffèrent des puissances (puissances brutes) du deuxième opérande (B), ou les additions desdits coefficients (chiffres bruts) si lesdites puissances (puissances brutes) dudit premier opérande (A) sont égales aux dites puissances (puissances brutes) du deuxième opérande (B), les additions desdits coefficients (chiffres bruts) étant obtenues par des additions arithmétiques de coefficients finis (chiffres bruts) effectuées par ladite unité logique arithmétique (TALU) ;
- dans ladite opération de multiplication, les puissances résultantes (puissances brutes) sont obtenues par des additions de chaque puissance unique (puissance brute) dudit premier opérande (A) et de chaque puissance unique (puissance brute) dudit deuxième opérande (B), et lesdits coefficients résultants (chiffres bruts) sont obtenus par des multiplications de chaque coefficient unique (chiffre brut) dudit premier opérande (A) par chaque coefficient unique (chiffre brut) dudit deuxième opérande (B), lesdites additions et multiplications étant obtenues par des additions et multiplications arithmétiques respectives de puissances finies (puissances brutes) et de coefficients finis (chiffres bruts) effectuées par ladite unité logique arithmétique de bas niveau (TALU) ;
- dans ladite opération de division, un premier coefficient résultant (chiffre brut) $C_u$ dudit résultat (C) est obtenu par division arithmétique d'un premier coefficient (chiffre brut) dudit premier opérande (A) par un premier coefficient (chiffre brut) dudit deuxième opérande (B) et une puissance maximale respective (puissance brute) 1L dudit résultat (C) est obtenue par soustraction arithmétique d'une puissance maximale (puissance brute) dudit deuxième opérande (B) à une puissance maximale (puissance brute) dudit premier opérande (A), de sorte qu'un premier reste partiel $(R_1)$ soit obtenu par :

$$R_1 = A - c_{iL} \cdot B$$

ledit processus étant répété de sorte qu'un i+1ème coefficient résultant (chiffre brut) soit obtenu par la division d'un ième coefficient (chiffre brut) d'un ième reste partiel $(R_i)$ par un premier coefficient (chiffre brut) dudit deuxième opérande (B) et une i+1ème puissance résultante (puissance brute) dudit résultat (C) soit obtenue par la soustraction d'une puissance maximale (puissance brute) dudit deuxième opérande (B) à une puissance maximale

(puissance brute) dudit i$^{éme}$ reste partiel **($R_i$),** ledit processus étant répété jusqu'à ce que ledit reste partiel **($R_i$)** soit égal à zéro ou jusqu'à ce qu'une précision requise du résultat (C) soit atteinte, lesdites soustractions et divisions étant obtenues par des soustractions et divisions arithmétiques respectives de puissances finies (puissances brutes) et de coefficients finis (chiffres bruts) effectuées par ladite unité logique arithmétiques (TALU).

**5.** Système informatique selon l'une quelconque des revendications 1 à 4, apte à ce que ladite base ne soit pas stockée pendant les opérations desdites unités logiques (NALU, TALU) et desdites mémoires de stockage.

**6.** Système informatique selon l'une quelconque des revendications 1 à 5, apte à ce que lesdits coefficients finis (chiffres bruts) et lesdites puissances finies (puissances brutes) soient représentés dans une représentation de base deux.

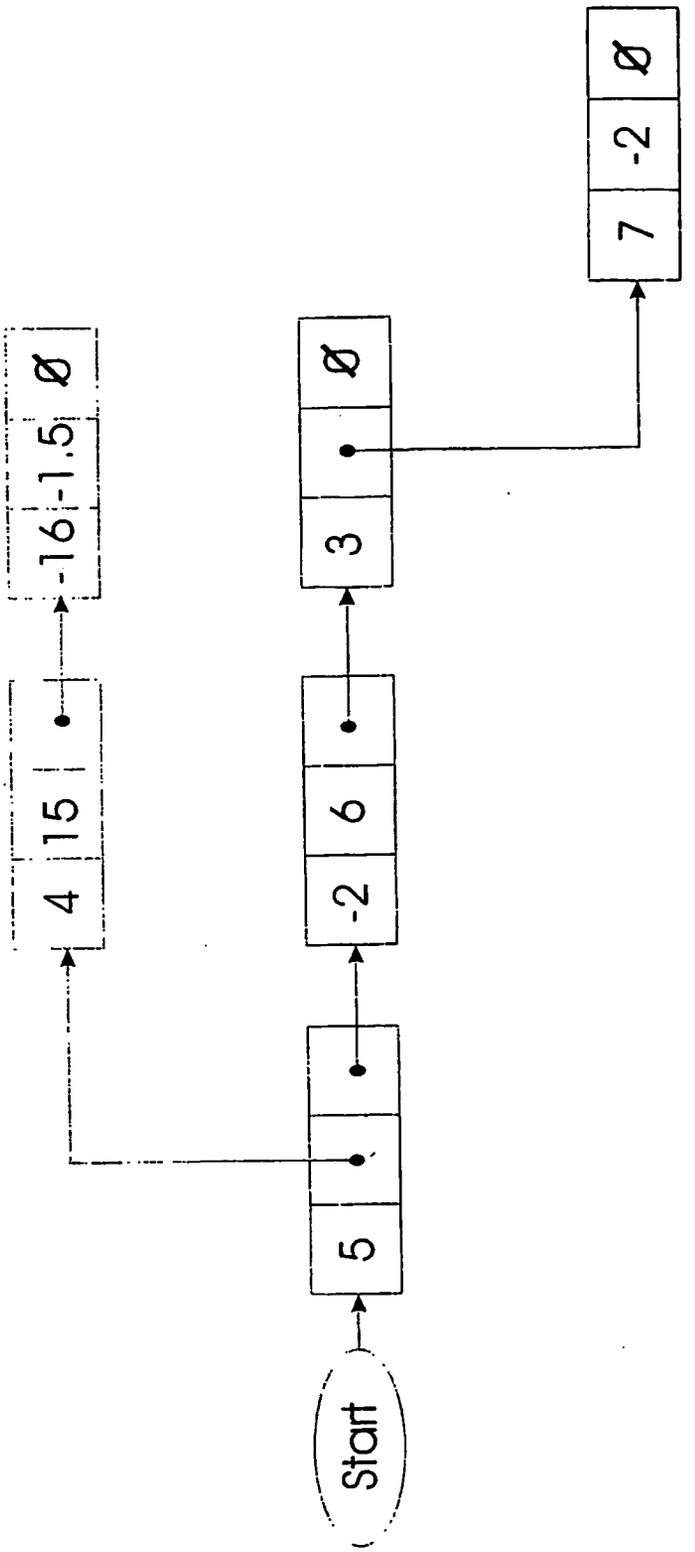


Figure 1

## REFERENCES CITED IN THE DESCRIPTION

*This list of references cited by the applicant is for the reader's convenience only. It does not form part of the European patent document. Even though great care has been taken in compiling the references, errors or omissions cannot be excluded and the EPO disclaims all liability in this regard.*

### Patent documents cited in the description


- US 6658443 B1 **[0042]**
- US 2003 B1 **[0042]**


### Non-patent literature cited in the description


- **Ya.D. Sergeyev.** Arithmetic of infinity. 2003 **[0042]**