NASA TECHNICAL MEMORANDUM

NASA TM X-1885

NASA TM X-1885

HARDWARE (OR SOFTWARE) PROVISIONS FOR MULTIPLE PRECISION FLOATING-POINT ARITHMETIC

by L. Richard Turner

Lewis Research Center
Cleveland, Ohio

NATIONAL AERONAUTICS AND SPACE ADMINISTRATION • WASHINGTON, D. C. • SEPTEMBER 1969

HARDWARE (OR SOFTWARE) PROVISIONS FOR MULTIPLE PRECISION FLOATING-POINT ARITHMETIC

by L. Richard Turner

Lewis Research Center

SUMMARY

A variant of floating-point arithmetic is described which permits the easy realization of indefinite precision within the framework of the floating-point arithmetic. The principal difference of this arithmetic involves the details of floating-point addition. It has been realized in hardware and also simulated. It is proposed that this feature be considered as an option or mode for future computers.

INTRODUCTION

In 1960, the Lewis Research Center was faced with the decision of whether to upgrade the capability of an existing UNIVAC 1103 computer, or to replace it in order to keep pace with a growing workload of the evaluation of experimental research data. Because at that time there were no obvious candidate machines for replacement, it was decided to add a transistorized floating-point arithmetic unit to the UNIVAC.

Although the major workload of the 1103 computer dealt with relatively short word-length data, experience, especially with the solution of problems in least squares had shown that the relatively short fraction length (27 bits) of most floating-point arithmetics was inadequate. Most problems of this type had been successfully processed with simulated floating-point arithmetic using a 35 bit fraction and separate exponent although even this precision was occasionally marginal. Because the basic computer had only 4096 words of core store, it was decided to implement the 35 bit form and even higher precisions directly in the hardware. It proved to be very easy to achieve high precision integer or fractional arithmetic within the framework of the hardwarized floating-point arithmetic.

Until quite recently, there had been little interest in precision of arithmetic. This situation has been dramatically changed as a result of the severe difficulty of problem solution in the floating-point arithmetic of the 32 bit word length computers. This difficulty led to several investigations of the precision of computation (mostly unpublished) with the resulting realization that users had rather generally been more optimistic about the quality of the results of computation than was justified. Among other results of these studies is a growing interest in extended precision arithmetic. Because of this change in interest, it has been decided to describe the pertinent features of the floating-point hardware and to show how easily extended precision arithmetic can be achieved. We may note that almost the same arithmetic has been coded in an 18 bit computer which did not have floating-point hardware.

PROPERTIES OF THE ARITHMETIC

No attempt will be made to describe all of the features of the hardware. Many of them were designed to reduce the demands on core store which is now a somewhat less pressing problem. They are largely irrelevant to this discussion in any case.

The principle hardware consisted of two storage registers designated as R and OP. Each arithmetic operation produced two results with a separate exponent for each part. The principal result was always returned to R and a secondary result was returned to OP. As will be noted, the identity of these registers could be logically interchanged. The actual arithmetic was performed in one 74-bit arithmetic register with an additional 37-bit transfer register.

The relevant operations consisted of LOAD, LOAD NORMALIZE, ADD, MULTIPLY, DIVIDE, REVERSE DIVIDE, STORE(R), and STORE INTERCHANGE, which caused the storage of R followed by interchange of the identity of R and OP. In addition to the ability to address the general store, it was also possible to address the internal storage registers except for the operation STORE.

The instruction code also contained bits which permitted the change of the sign of an operand read from store or to control whether the result of any operation was truncated or rounded. When rounding was elected, it was a postnormalized rounding. In every operation an exact residue of the computation was stored in OP. The residue was adjusted to compensate for rounding if rounding increased the value of the more significant part of the result. It is crucial to the achievement of multiple precision arithmetic that the residue be exact for every operation except division. When the other multiple precision operations are correctly executed, multiple precision division is essentially trivial.

The actual hardware performed only one form of arithmetic on numbers with a long biased exponent part and signed 35 bit fractions part. The actual exponent length was 30 bits and what is adequately long cannot be reliably specified. There is no adequately long exponent to permit the arbitrary approximation of zero or infinity which is a part of the theory of mathematics. It is suggested that the minimum desirable length ranges from 12 to 18 bits.

We note first that although a residue was always formed and stored in OP, this residue was destroyed by reading the next operand into OP for any subsequent operation; the execution of multiple precision arithmetic requires that the content of OP be stored before any further operation is initiated. In the execution of the usually coded sequences of instructions the residue was not required.

The operation of multiplication was trivial and was different from the methods commonly employed only to the extent that the results were always normalized. Because rounding might cause arithmetic overflow the exponent part of the residue was computed first and stored. If the 36th bit of the product was a "1" and rounding were ordered, one unit was added to the 35th bit position. Two bits were included in the register to separately provide a sign and an overflow position. In any case, if roundup occurred, the low order part of the result was complemented (2's complement) and its sign set opposite to the principal result. In any case it was shifted one bit to the right and the result stored as the fraction of the residue in OP.

Finally, the exponent of the principal part was computed. If arithmetic overflow occurred because of rounding, the principal exponent was increased by one count and the principal result shifted right one bit. The adjusted exponent and fraction with a sign was then stored in R.

The method of performing addition differed from conventional practice in a crucial way. The first detailed operation was to subtract the exponent of the addend (in OP) from the exponent of the augend (in R). If this was negative, the register identity was interchanged and the exponent difference complemented. If the exponent difference then exceeded the fraction length, $(35)_{10}$, the operation terminated. Otherwise, the fraction of OP was loaded into the arithmetic unit and shifted right to align the fractional parts. These were then added algebraically and the result and sign complemented if necessary. Arithmetic overflow could occur and if it did, the entire result was shifted one bit right and the exponent adjusted. The final steps were then performed as in multiplication.

Division was also somewhat unusual in that it was desired to form a rounded quotient if so specified and to compute a correct residue. We note that Reverse Division required only that the register identity be interchanged prior to an actual division.

In the division operation an unnormalized divisor produced a conditional trap. Because the registers were addressable, the option existed to consider the trap condition a fault or to normalize the divisor and continue. Application-related coding is required to effect the decision.

The dividend fraction was first normalized. Then if the absolute value of the fraction was less than the absolute value of the fraction of the divisor the dividend shifted left by one bit. If rounding had been elected, the absolute value of the divisor fraction was attached to the dividend. In any case, a 35 bit quotient and remainder was developed. Finally, if rounding had been elected, the absolute value of the divisor was subtracted from the remainder to produce a correctly signed residue. The quotient and residue with appropriate exponents were then stored in R and OP, respectively.

We may note that in the softwarized version of this arithmetic the remainder is not generated because for division it is really not required. We also note that both implementations use absolute value and attached sign for the fraction and a biased exponent.

MULTIPLE PRECISION OPERATIONS

Multiple precision operation depends fundamentally only on three subroutines. One provides for the storage of the parts of the entire result. This proves to be convenient by the use of array operations but would be even easier if a pushdown stack or some form of variable length dynamic storage allocation were available. The latter seems preferable but is not usually at hand.

The second subroutine provides for the addition of individual floating point numbers to a collection of previously computed and added parts.

The third is a "standardize" subroutine which uses the second of these subroutines to produce a result which depends only on the value and length of the multiple part floating-point result and removes the possibility of nonunique representation.

The first subroutine deserves no detailed account.

For the addition of two numbers, after a work space is initially set to zero the sum of two multiple part numbers consists merely in delivering their individual parts in sequence to the addition subroutine.

Multiplication requires that for two numbers that are respectively N and M words long, the 2NM separate pieces be delivered to the second subroutine.

Division is executed as a simple form of Fourier division. The dividend is placed in a buffer space preferably in standard form. A partial quotient Q_1 is then formed and stored in a reference store. The product of Q_1 times all elements of the divisor is then subtracted from the initial dividend using the second subroutine. As many partial quotients are formed as are desired or permitted by a repetition of this algorithm. Note that division does not ordinarily generate a finite length quotient so that a limitation on the length of the quotient is mandatory.

If the remainder were available in the division operation, some simplification is possible if following each division step the dividend element were replaced by the remainder.

The standardize subroutine merely acts like a dummy arithmetic operation which delivers zero to the second or general addition routine and as the process goes on, usually in several steps checks to determine whether the result is in standard form.

It is apparent that the second or generalized addition routine is the important routine and performs most of the work. It's structure, arrived at after considerable investigation in the hardware-software design period proves to be somewhat different than had at first been expected.

A critical part of the objective is that the subroutine will ultimately produce a standardized result. A standardized result is defined to be one in which, of the several floating-point results which make up a single multiple-precision number the absolute values of the first (leftmost, if you wish) is the largest and that the absolute values of the parts decrease monotonically as successive words are encountered. This is a possible definition for any number base. In a binary implementation it is desirable that the exponent parts of successive words differ by at least $n + 1$ where n is the fraction length. In the implemented cases the minimum difference is 36.

In a nonbinary computer the best that can be guaranteed is that the difference is equal to the number of digits of the fractional part. This leads to a different condition for the addition operation in that the addition of two floating-point words is not attempted if the difference of the exponents exceeds the fractional word length or if the difference of exponents equals the fractional word length and the signs of the parts are alike.

For the purpose of definition, a result can also be said to be standardized if the individual floating-point words are disjoint in the sense that if they were added, no actual addition is required.

What proved to be somewhat surprising is that the feasible process started by adding a new element of the final sum to the most significant part of an existing multiple word number even though the new element had a very small value. If its value relatively is quite large, then the ordering aspect of the addition algorithm replaced the previously most significant word by a different value with a residue which was now added to the next word in the augend file and so on until all of the data were exhausted. The most convenient control of the sequence was generated by replacing each word in the augend file by zero as it is read from store. If the space allocated to the augend ends with a floating-point zero then the encounter of a zero-valued word is a sentinal that the present step of addition is complete.

Attention is called to the fact that in reducing the initial dividend by subtracting the product of the first quotient and the entire divisor, it is to be expected that first subtraction will produce a quite small result. If the initial dividend was standardized, then subtraction tends to move a word with the most significant part of the adjusted dividend into the proper position for a further division step.

Multiple precision division is one process in which there is a tendency for strong cancellation of value. Another process is the multiplication of a matrix by its approximate inverse. Here, if the inverse is a reasonable one, all the diagonal elements of the product matrix tend toward unity and all the off-diagonal elements tend toward zero. This trend toward zero is a property of any iterative method for the refinement of the solution of an equation or set of equations.

Attention is called to a few further details of the addition algorithm. When the partial operands of any step are of opposite sign, it will frequently happen that the immediate sum is only one fraction long. In this case, the residue is zero and before any result is stored the next word of the augend should be added. In any case, the first word stored is placed in the leftmost position of the augend file and successive nonzero words in successive positions.

It is important to note that despite the tendency of the addition algorithm to reduce the total number of words required, the count of total number is the number initially in the augend file plus one for the new value which is being added. This may increase the number of words in the file so that the assigned file length may be exceeded. This usually requires that the least significant word be discarded.

It may have been noticed that the addition operation tends to move the more significant values toward the leftmost (first) part of the augend file. Because of this property the simplest standardization subroutine merely repeatedly adds zero to the augend file until the results are appropriately disjoint.

One important observation concerning this method of executing multiple precision arithmetic is that the successive parts are usually not all of the same sign. This is unavoidable if only one fraction length is used in the elementary floating-point addition. Advantage of this property is usable in a binary computer to increase the average effective length of a fraction. Even without rounding the minimum effective length of the successive floating-point words is one bit more than the nominal hardware length. With rounding the average effective length is one additional bit.

This property does not carry over to higher values of the base such as 8 or 16 and, in any case, the gain in apparent length is not adequate to compensate for the many times repeated exponent parts. The advantage of the method is that it uses only a basic floating-point arithmetic if the floating-point addition is properly designed.

One additional feature of the system is that if the initial values of the operands are integers and the partial quotients accepted in division are constrained to be integers, the same algorithm is quite adequate to perform all of the ordinary operations of integer or rational fraction arithmetic.

CONCLUDING REMARKS

The hardware floating-point arithmetic unit constructed at the Lewis Research Center (or equivalent simulations of the hardware) have made the realization of multiple precision floating-point arithmetic extremely easy. The principal difference between this hardware and several others is the treatment of floating-point addition when the difference of the exponent parts of the augend and addend exceed the number of characters of the standard computer word.

It is suggested that the NASA addition algorithm either as the only method of addition or as optional mode of addition be considered as a design characteristic of future computers.

Lewis Research Center,
National Aeronautics and Space Administration,
Cleveland, Ohio, July 18, 1969,
129-04.

1. Report No.	2. Government Accession No.	3. Recipient's Catalog No.
NASA TM X-1885		

4. Title and Subtitle	5. Report Date
HARDWARE (OR SOFTWARE) PROVISIONS FOR MULTIPLE PRECISION FLOATING-POINT ARITHMETIC	September 1969
	6. Performing Organization Code
7. Author(s) L. Richard Turner	8. Performing Organization Report No. E-5184
9. Performing Organization Name and Address Lewis Research Center National Aeronautics and Space Administration Cleveland, Ohio 44135	10. Work Unit No. 129-04
	11. Contract or Grant No.
12. Sponsoring Agency Name and Address National Aeronautics and Space Administration Washington, D. C. 20546	13. Type of Report and Period Covered Technical Memorandum
	14. Sponsoring Agency Code

15. Supplementary Notes

16. Abstract

A variant of floating-point arithmetic is described which permits the easy realization of indefinite precision within the framework of the floating-point arithmetic. The principal difference of this arithmetic involves the details of floating-point addition. It has been realized in hardware and also simulated. It is proposed that this feature be considered as an option or mode for future computers.

17. Key Words (Suggested by Author(s))	18. Distribution Statement
	Unclassified - unlimited

19. Security Classif. (of this report)	20. Security Classif. (of this page)	21. No. of Pages	22. Price*
Unclassified	Unclassified	8	$3.00

*For sale by the Clearinghouse for Federal Scientific and Technical Information
Springfield, Virginia 22151

NATIONAL AERONAUTICS AND SPACE ADMINISTRATION
WASHINGTON, D. C. 20546

OFFICIAL BUSINESS

FIRST CLASS MAIL

POSTAGE AND FEES PAID
NATIONAL AERONAUTICS AND
SPACE ADMINISTRATION

POSTMASTER: If Undeliverable (Section 158 Postal Manual) Do Not Return

"The aeronautical and space activities of the United States shall be conducted so as to contribute . . . to the expansion of human knowledge of phenomena in the atmosphere and space. The Administration shall provide for the widest practicable and appropriate dissemination of information concerning its activities and the results thereof."

— NATIONAL AERONAUTICS AND SPACE ACT OF 1958

NASA SCIENTIFIC AND TECHNICAL PUBLICATIONS

TECHNICAL REPORTS: Scientific and technical information considered important, complete, and a lasting contribution to existing knowledge.

TECHNICAL NOTES: Information less broad in scope but nevertheless of importance as a contribution to existing knowledge.

TECHNICAL MEMORANDUMS: Information receiving limited distribution because of preliminary data, security classification, or other reasons.

CONTRACTOR REPORTS: Scientific and technical information generated under a NASA contract or grant and considered an important contribution to existing knowledge.

TECHNICAL TRANSLATIONS: Information published in a foreign language considered to merit NASA distribution in English.

SPECIAL PUBLICATIONS: Information derived from or of value to NASA activities. Publications include conference proceedings, monographs, data compilations, handbooks, sourcebooks, and special bibliographies.

TECHNOLOGY UTILIZATION PUBLICATIONS: Information on technology used by NASA that may be of particular interest in commercial and other non-aerospace applications. Publications include Tech Briefs, Technology Utilization Reports and Notes, and Technology Surveys.

Details on the availability of these publications may be obtained from:

SCIENTIFIC AND TECHNICAL INFORMATION DIVISION

NATIONAL AERONAUTICS AND SPACE ADMINISTRATION

Washington, D.C. 20546