数理解析研究所講究録
339巻 1978年 9-12

OVF (Over Flow) Free Computing

by Tetsuo Ida** and Eiichi Goto*,**

* Department of Information Science,
The University of Tokyo, Hongo, Tokyo 113 Japan

** The Institute of Physical and Chemical Research
Hirosawa, Wako-shi, Saitama 351 Japan

September 12, 1977

10

1. Motivation

Overflows can be quite troublesome to numerical programming because most contemporary computer systems do not provide adequate means for handling overflow. The computing centre of a Japanese university received a great number of overflow complaints upon the replacement of machine by one with fewer exponent bits. Transporting FORTRAN programs from a larger machine to a smaller machine is often troublesome because of the difference in the word length of integers. Even the language specifications are compatible, these troubles are caused by the incompatibility in numerical data specifications.

We encountered an overflow trouble when we were running a performance evaluation program for a hashing hardware to be used with parallel memory banks, J in number [1]. The FORTRAN program produced beautiful results up to J = 16 but started to print "EXPONENT OVER FLOW ..." for J > 16.

2. Software Aids

We resolved our trouble by recompiling the program with the aid of a nonstandard FORTRAN feature called "TYPE" statement [2], which declares some variables to be regarded as arrays similarly to a DIMENSION statement. Arithmetic and logical operations on "TYPE"d variables are compiled into calls on subroutines, which are to be written by the user. The only things we had to do were to add TYPE statements and to write some subroutines for extended exponent arithmetic. Actually a full word integer exponent of 36 bits was used instead of 9 bits.

The "SUPER PRECISION" FORTRAN pre-compiler package of Wyatt [3] would also provide software aids for overflow, if "super exponent" feature were added.

The "BIG FLOAT" package developed by Fateman [4] for the formula manipulation system MACSYMA can handle arbitrarily long integers and floating point numbers with arbitrarily long exponents and mantissas.

A portable overflow free FORTRAN compiler with numerical data specification similar to the "BIG FLOAT" is now being written by our group.

3. A New Architecture for Efficiency

In all pure software means for eliminating overflows disclosed in 2, arithmetic operations would be slowed down by a large factor S, at least 10, in comparison with a standard FORTRAN code, because of large software overheads in run-time data type checks.

The situation would be greatly improved in tag machines. Arithmetic operations on "untagged" small number(s) are to be made with standard hardware at high speed while those on "tagged" big numbers are to be trapped and handled by software or microprogramming. Untagged small numbers are likely to appear with high probability P in most programs. Hence, the slow down factor would be improved as $S' = P + (1 - P)S \leq S$.

Bit-loss, a common objection against tag-bit(s), can be remedied by using a specific exponent value as a "trapping tag".
In case of 7-bit exponent $-63 \leq p \leq 64$, $p = 64$ may be used as the "tag", with the mantissa part being a pointer to a data structure representing a "big" number. Bit or entropy loss in this scheme is only
$\log_2 (127/128) = -0.01$ bits.

4. Concluding Remarks

Besides causing incompatibility of numerical programs, overflows have often hampered the design and implementation of some important classes of algorithms [6, 7]. The importance of variable precision

algorithms in multiprecision arithmetic has been pointed out by Brent [8]. The "trapping tag" would be also useful for implementing such algorithms efficiently.

We are urging some local venders to consider "trapping tag" architectures, so far without success.

In a microprogrammed machine called FLATS being build by ourselves, "trapping tag" will be incorporated besides some other new features such as hardware hashing [1].

References

[1] T. Ida and E. Goto: "Performance of a Parallel Hash Hardware with Key Deletion", Proc. IFIP Congress 77, North-Holland (1977).

[2] "FACOM-230/60 FORTRAN Manual" SP-061-4-5, Fujitsu Ltd (June, 1968) in Japanese.

[3] W. T. Wyatt, P. W. Lozier and P. J. Orser: "A Portable Extended Precision Arithmetic Package and Library With Fortran Precompiler", ACM Trans. Math. Software, Vol. 2 (1976) pp.209-231.

[4] R. J. Fateman: "The MACSYMA 'Big-Floating-Point' Arithmetic System" in Proceeding of ACM-SYMSAC '76, Yorktown Heights NY, Aug. 1976.

[5] E. Goto and T. Ida: "Trap-NUM a Method for Handling Big Numbers" Programming Symposium of the Information Proc. Soc. of Japan (Jan., 1977 in Japanese).

[6] H. Takahasi and M. Mori: "Double Exponential Formulas for Numerical Integration", Pub. Research Institute for Math. Science, Kyoto University 9 721 (1974).

[7] A. A. Grau: "Algorithm 256, Modified Graeffe Method", Collected Algorithms from CACM.

[8] R. P. Brent: "Fast Multiple-Precision Evaluation of Elementary Functions", JACM Vol. 23 (1976) pp.242-251.