CGTM 70
John Ehrman
May 1969

A STUDY OF FLOATING-POINT CONVERSIONS IN SOME OS/360 COMPONENTS.

At the SHARE XXXII meeting in Los Angeles (March 3-7, 1969) I introduced two resolutions in the Fortran Project which dealt with the problems of accurate conversions of floating-point numbers between external (decimal) and internal (hexadecimal) representations. The problem which leads to these resolutions is as follows:

> In the Fortran G and H compilers and their supporting library modules, the algorithm used for converting decimal data to internal form is such that inaccurate results are obtained for occasional arguments. While these arguments form a very small subset of the total range of convertible values, no indication is given of the possible errors, nor does the programmer have any control over the magnitudes of errors he is willing to allow.

In the discussions that followed it became clear that there was no reliable data as to the accuracy of the conversions produced by any of the above modules, so the resolutions were tabled for further discussion at a later time. This memorandum presents the results of a detailed study of conversions from decimal to long floating-point form in the G and H-Level Fortran compilers, the Fortran library, the F-Level PL/I compiler and its run-time library, and the F-Level Assembler.

Method

A routine was written in Fortran which generated random strings of 15 decimal digits. These were prefixed by the characters "+.", and followed by the letter "D". Then, using an in-core formatting routine,

integer exponents from -75 to +75 were placed following the "D". Using the same in-core formatting routine (reference 1), the character string was converted to a long (REAL*8) floating-point number. At the same time, the internal value was calculated in multiple precision using a set of multiple-precision floating-point arithmetic routines (reference 2). The resulting values were compared, and if the difference in the low-order digits exceeded a preset tolerance, the character string was written onto a scratch file, with some other information that allowed its use in an assignment statement. After a preset number of test cases had been run, some other Fortran statements were attached to the scratch file so as to form a complete Fortran Program.

This Fortran program was then compiled by the G and H compilers, and when executed, each wrote the compiler-produced values for the long floating-point numbers onto a result file in hexadecimal. The compiler input was also run through another short program with appropriate Format statements which wrote the converted values produced by the Fortran library routines.

To test the PL/I and Assembler conversions, a program was written in PL/I to convert the Fortran-style input file to both a PL/I program and a sequence of DC statements acceptable to the Assembler. In each case the hexadecimal values were again written onto the result file.

Finally, a Fortran program was written which read the result file, and also recomputed in multiple precision the corresponding "exact" value from the original decimal digit string. These values were then compared among one another, and the results tabulated.

Results

Table 1 shows the statistics for the original test-case generation process. In each case the difference in the low-order digits was calculated, and the number of cases with that difference is shown.

TABLE 1. Instances of Low-Order Differences for Initial Test Cases

Difference	Cases	Difference	Cases
0	264895	1	145690
2	33985	3	12406
4	6250	5	2872
6	1870	7	989
8	790	9	487
10	322	11	316
12	199	13	55

Except for an occasional floating-point result of zero (this is caused when the value "underflowed" due to too many leading zeros in the input digit string), no differences greater than 13 were observed.

Table 2 summarizes the results of comparing the values produced by each of the components. The row and column headings of FH, FG, FL, PL, PF, AF, and MP refer respectively to Fortran H compiler, Fortran G Compiler, Fortran Library, PL/I Library, PL/I F Compiler, Assembler F, and Multiple Precision values. Each box contains three entries: the first is the number of discrepancies observed in the pairwise comparisons, the second is the algebraic sum of the discrepancies over all cases, and the third is the sum of the magnitudes of the discrepancies over all cases. The sign of the discrepancy was determined by subtracting from the value produced by the processor at the top of the column that value produced by the processor at the left of the row. For example, the data in the lower left corner was formed by subtracting the MP values from those produced by the Fortran H compiler, which can be seen to have a slight positive bias.

The values produced by the multiple-precision routines were not rounded, since no attempt is made (except in the Assembler) to round the floating-point result.

TABLE 2. Summary of Conversion Discrepancies for 892 Test Values

	FH	FG	FL	PL	PF	AF
FG	520 -6320 6320					
FL	520 -6320 6320	0 0 0				
PL	889 5574 9200	892 11894 15520	892 11894 15520			
PF	892 5764 9390	892 12084 15710	892 12084 15710	92 190 870		
AF	838 -340 4748	892 5980 9440	892 5980 9440	814 -5914 6088	804 -6104 6278	
MP	726 139 4453	892 6459 9799	892 6459 9799	840 -5435 5727	829 -5625 5923	479 479 479

Analysis

A number of simple observations may be made immediately.

1. Because the results produced by the Assembler are never in error by more than 1 in the low-order digit, an examination was made of those values for which a discrepancy existed. In each case it was due to the fact that the Assembler produces a rounded result, whereas the MP value was unrounded. Thus the Assembler produced the correct value in all cases.

2. Because the average algebraic error for the Fortran H compiler is zero somewhere "between" the AF and MP values, we may conclude that

Fortran H produces a relatively unbiased value, even though the mean magnitude of the errors is about 5 in the low-order digit.

3. The only case where there was complete agreement between a compiler and its run-time library was for the Fortran G compiler. The mean magnitude of the errors is large, however, being about 11 in the low-order digit.

4. Because the total discrepancy between the FH and FG values is the same as its magnitude, we may conclude that the Fortran G compiler and the Fortran Library always give a value larger than that produced by the Fortran H compiler.

5. The mean discrepancies between Fortran and PL/I components is very large; this could be a troublesome source of problems in converting programs from one language to another.

6. The errors produced by Fortran and PL/I have opposite signs in almost all cases.

Conclusions

The only reliable component of the operating system is the Assembler, as far as floating-point conversions are concerned.

Configuration

The tests were run on a Model 91 under MVT (release 16). The status of the processors was:

Fortran H: Level 16 (1 July 68) with PTFs 22244, 21537, 21940, 21765, 22395, 22771, 22494, 21967, 20464

Fortran G: Release 16

Fortran Library: Release 16, ZAP to remove "END=" error message

PL/I F: Version 4 Release 16

PL/I Library: Release 16

Assembler F: Release 16, F01AUG68, with PTF 19077

References

1. SLAC Library Routine FI0999.

2. SLAC Library Routine MPA (A1-27), available from PID as 360D-40.4.003