Comments on version 1.1 of Draft Proposal for a Language-Based Arithmetic Standard

B A Wichmann 87-11-06

Introduction

The proposal is an excellent starting point. However, being a specific proposal, it lacks any rationale for the various decisions that have been made. In particular, it is very specific in one issue which I believe is of critical importance - the degree of abstraction from an implementation.

Abstraction versus Concrete specification.

The IEEE standards are essentially the specification of an actual implementation. Relatively few choices would have to be made to provide a complete logical specification. In contrast, the Brown model in very abstract and only just manages to specify enough to guarantee that the properties of floating point are provided by a hardware unit which satisfies the Brown axioms. In my view, both specifications are inappropriate for an international standard.

The reasons against a concrete standard such as IEEE is that an implementation is too constrained - for instance, the majority of floating point systems do not adhere to it, and hence compliance is restricted in a way that is of little concern to most users. Of course, suppliers producing new systems can no doubt use IEEE (as they do), but the very precise details may unduly constrain the hardware. This appears to be the case as the Weitek chips have a go-fast or IEEE mode. One must remember that standards tend to last much longer than specific hardware and hence one must try to allow some flexibility for future hardware (which may work in ways not yet envisaged).

The reasons against abstraction is that the model used may be too weak to satisfy the user requirements. For instance, the Brown model does not require that any specific action is taken on floating point overflow. Since virtually no practical computation can be guaranteed to be overflow free, the Brown model itself is not adequate to meet the needs of users. (Since one needs to ensure that overflow has not occurred if the results are to be meaningful). Similarly, the Brown model does not specify any action on underflow. The variant of the Brown model used in the Ada language definition ignores underflow so that a machine may perform either gradual underflow or give zero (but cannot give an exception). Hence this version of the Brown model mirrors systems without an underflow trap set.

It is clear from the above that I advocate a version of the Brown model which takes an intermediate position. One needs more flexibility than can be obtained from a logical specification of a complete system, and yet a strong enough set of requirements to meet the reasonable aspirations of the user. Hence the remaining part of this paper proposes an alternative approach based upon the version 1.1 proposal.

Characteristics of a Standard

Several aspects of the standard are non-controversial, but need to be discussed:

a) The Radix
The IEEE 854 standard restricts this to 2 or 10 and thus ignores the IBM architecture. Of course, radix 16 has a number of undesirable characteristics, but the net effect is to lose 3 bits from the last place compared with a binary equivalent. This is quite acceptable in the sense that an approximation is obtained. I do not see any real justification of restricting the radix at all (to >= 2). Hardware already has 2, 8, 10 and 16. A software floating point system on a byte machine could easily use a radix of 256 to avoid bit-level normalization. Similarly, I have written radix 10000 routines myself mainly to avoid radix conversion on output and yet avoid complex normalization. A large radix gives a relatively efficient software implementation.

b) The number of places
Brown states this should be greater than or equal to 2. I believe that a precision corresponding to 5 decimal places is a useful minimum, but I see no point in specifying this (after all, existing language standards do not give a lower limit for integers).

c) Physical representation
No requirements should be given. For instance, with the byte software floating point system, the size taken for a value could depend upon the value itself (up to the maximum for the precision provided).

d) Exponent range
Brown places some very weak restrictions on the range. The Version 1.1 paper gives another restriction required for conversions to/from integers. I think that this last restriction is inappropriate, but that could be rephrased to apply only if there is an appropriate integer format. This is not a critical issue.

The controversial aspects

Version 1.1 is not a model (like Brown) but a specification which does not specify rounding. Hence with 1.1, one can round either way, but this must be deterministic and the precision is never greater than that of the specified precision. In contrast, Brown allows machine numbers in addition to model numbers (which have exactly the precision specified). This means that with Brown a radix 2, 30 place system is also a radix 2, 29 place system. Also, a radix 16, D place system is also a radix 2, 4*D-3 place system. (In this discussion, I am ignoring the exponent range which must be adjusted also.) I believe that this property of the Brown model is very important for two major reasons:

1) If a system has a small defect, then this may be masked by means of a penalty (i.e, pretending that the machine is less accurate that the representation). This property is very useful in testing.
2) An implementation is always free the provide more accuracy than that specified. This is essential on some machines where the accumulator works to 'double' precision while values can be stored to 'single' precision. On such a machine, the accuracy given to be user of a high level language depends upon the register optimization performed. I believe that this is a reasonable implementation strategy which should not be excluded by a standard. (It provides very hardware-cheap 'single' precision.)

Another controversial aspect (in my view!) is that of underflow detection. I believe that an acceptable floating point system produces gradual underflow or zero on underflow and that detection need not be mandated. On the other hand, being able to determine the behaviour without execution is convenient. (Hence the availablility of the parameters of the standard to the program needs consideration.)

The small print

1. I believe that there are problems in defining the accuracy of exponentiate if non-integer exponents are involved. Are there some references to this?
2. The definition of 'trap' as a hardware transfer of control is inappropriate since the entire system could be implemented in software.
3. I would much prefer to define a single precision and then specify how different precisions have to fit together. This would allow DEC to claim conformance for four precisions on the Vax, and IBM three on the 360s. This also avoids terms such as "short" and "long" which may not correspond to the terms used by the supplier.
4. The identities listed in section 4.1 are not satisfied in the Brown Model. It is obvious to see how these can fail if one considers a machine with an overlength accumulator.
5. A common source of problems can be the conversion of decimal to internal format performed by the compiler or run-time system. Although radix conversion is difficult, I do not think this issue can be avoided.
6. The Brown Model permits subtraction (with overflow) to be used to implement comparisons. This results in a penalty in the exponent range (i.e, the claimed range smaller than the physical range). This is not satisfactory, since a user cannot creditably avoid the unsafe exponent range. Hence I believe that comparison should be overflow free.
7. I am not sure if it is necessary to distinguish the exception of division by zero from overflow.
8. The Pascal standard uses the terms 'implementation defined' (meaning that a conforming implementation must define it in the compliance statement); and 'implementation dependent' meaning that no definition or consistency is required. I believe that the standard should specify the compliance statement in detail.

Surveys

In order to make a more rational choice, two surveys should be undertaken: of standard programming languages and computer architectures. Companies like NAG and IMSL may be able to provide much of the data.