July 26, 2022

Abstract

Consideration of language features and commands drawn from HP 2000F BASIC (TBS) towards an implementation of a BASIC interpreter system for FRAM-based MSP430 MCUs.

This is a work-in-progress and it will likely start out (if not end) as a walk along my own path of considering both implementation details as well as larger issues, encountered here in a similar order to how I also did.

1 Floating Point

I'm fairly comfortable with most of the elements of developing a BASIC interpreter such as the HP 2000F TSB. The MSP430 doesn't have any hardware specifically dedicated for floating-point computations and so it must all be done in software.

So far as I'm aware (at the time of this writing) no open-source and freely available floating point library, optimized and written in assembly code, exists for the MSP430.

I'll be writing that code.

Conclusion: ***Writing a floating point support library is the first step along the solution path ahead.***

The essential details of IEEE 754-1985 standard had already become the *de facto* standard years before it was finally approved. Much effort and thought went into it and the subject of denormals (subnormals) was perhaps its most contentious facet.

But aside from the additions of reserved symbols for qNaN, sNaN, INF, and Indef, and the use of hidden-bit notation for everything except denormals, the semantic behavior of HP's 2100 floating-point and IEEE 754 are quite similar.

This fact, plus the relatively small amount of extra coding required to support unpacking and packing two different formats, led me to consider more seriously the idea of supporting both.

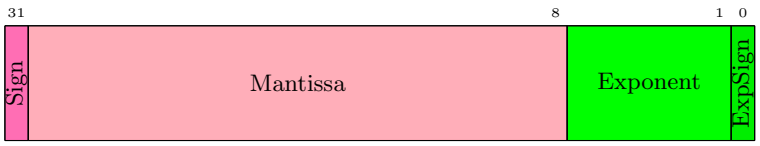
1.1 Hidden-bit Notation

Aside from the minor changes in code required to support two different formats, a potential added cost may be a re-evaluation of the existing non-linear minimax constants applied to the Chebyshev polynomials to generate transcendental function results. The use of hidden-bit notation improves precision but may also require similarly improved polynomial constants.

Whether or not this causes me to recant and support only the HP 2100 format is yet to be seen.

1.2 HP 2116 FP Format

The HP floating-point format is laid out like this:

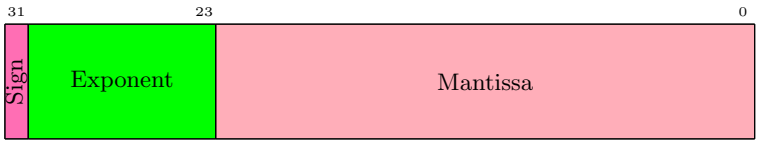


By today's standards, it's a little odd:

- The exponent is not stored in *excess-127* notation (which can be sorted more easily) but is instead stored in direct twos-complement form (except as noted in the next bullet below.)
- The sign bit of the exponent is moved to the lowest-order bit rather than the more expected place for twos-complement form. This would be as if the upper sign-bit of the traditional twos-complement form were always moved to the lowest-order bit, instead of where it is traditionally placed, today.
- The mantissa doesn't support hidden-bit notation, so it loses one bit of potential precision.
- The mantissa has the assumed radix point just to the left of the mantissa bits. But in this case with an implied "0." prefixed to it. In general, this means the exponent value is "off by one" (ignoring for now the *excess-127* notation) when compared to IEEE 754. (IEEE 754 assumes a "1." prefixed to the mantissa unless the value is an exact zero.)

1.3 IEEE-754 FP "Single" Format

The IEEE-754 single-precision floating-point format is laid out like this:



Here:

- The exponent is stored in *excess-127* notation. This means that when packing the result into the format, the value 127 is first added to the computed exponent. In general, this makes sorting easier.
- The stored exponent (in excess notation) is a simple unsigned integer. To restore it, the value 127 is subtracted from the stored value in order to compute a signed resulting exponent value.
- The mantissa does support hidden-bit notation, so it gains one bit of precision.
- The mantissa has the assumed radix point just to the left of the mantissa bits. But with an implied "1." prefixed (unless the entire value is zero.) This means the computed exponent is "off by one" with respect to the HP FP format.

IEEE-754 also supports other features: qNaN, sNaN, INF, and Indef. But those won't be supported, instead using the extra symbol in the exponent as an added power of 2. Denormals (subnormals) will be supported, though, as HP's 2100 behavior is similar.

1.4 Denormals-Are-Zero (DAZ) and Flush-To-Zero (FTZ)

These are also hot topics in numerical circles and I've decided to support them, as optional during assembly/build. They were not supported by HP's 2100 floating-point unit. But the behavior has value in certain areas (audio processing, for example) and the support isn't complicated.

1.5 Summary

Either format would work fine in HP 2000F TSB. The only difference in terms of supporting code is that there is a slight difference in the way that the stored formats are packed and unpacked and there is one more bit of support required in the basic mathematical operations.

This isn't a big difference in terms of execution time. The main differences are related to the HP 2000F TSB Chebyshev polynomial constants used in approximating the transcendental functions. (They may need to be slightly different to satisfy non-linear minimax requirements – which require some added work on my part.) And in the behavior of some very old code that depended on very specific behaviors of the HP 2000F TSB system's floating-point computations.

Tentative Conclusion: ***Provide settable options to select floating-point behavior and storage format.***