Floating-Point Numbers - History of IEEE Floating-Point Format*

Charles Severance
Kevin Dowd

1 History of IEEE Floating-Point Format

Prior to the RISC microprocessor revolution, each vendor had their own floating- point formats based on their designers' views of the relative importance of range versus accuracy and speed versus accuracy. It was not uncommon for one vendor to carefully analyze the limitations of another vendor's floating-point format and use this information to convince users that theirs was the only "accurate" floating- point implementation. In reality none of the formats was perfect. The formats were simply imperfect in different ways.

During the 1980s the Institute for Electrical and Electronics Engineers (IEEE) produced a standard for the floating-point format. The title of the standard is "IEEE 754-1985 Standard for Binary Floating-Point Arithmetic." This standard provided the precise definition of a floating-point format and described the operations on floating-point values.

Because IEEE 754 was developed after a variety of floating-point formats had been in use for quite some time, the IEEE 754 working group had the benefit of examining the existing floating-point designs and taking the strong points, and avoiding the mistakes in existing designs. The IEEE 754 specification had its beginnings in the design of the Intel i8087 floating-point coprocessor. The i8087 floating-point format improved on the DEC VAX floating-point format by adding a number of significant features.

The near universal adoption of IEEE 754 floating-point format has occurred over a 10-year time period. The high performance computing vendors of the mid 1980s (Cray IBM, DEC, and Control Data) had their own proprietary floating-point formats that they had to continue supporting because of their installed user base. They really had no choice but to continue to support their existing formats. In the mid to late 1980s the primary systems that supported the IEEE format were RISC workstations and some coprocessors for microprocessors. Because the designers of these systems had no need to protect a proprietary floating-point format, they readily adopted the IEEE format. As RISC processors moved from general-purpose integer computing to high performance floating-point computing, the CPU designers found ways to make IEEE floating-point operations operate very quickly. In 10 years, the IEEE 754 has gone from a standard for floating-point coprocessors to the dominant floating-point standard for all computers. Because of this standard, we, the users, are the beneficiaries of a portable floating-point environment.

2 IEEE Floating-Point Standard

The IEEE 754 standard specified a number of different details of floating-point operations, including:

*Version 1.3: Aug 25, 2010 10:29 am -0500
†http://creativecommons.org/licenses/by/3.0/

- Storage formats
- Precise specifications of the results of operations
- Special values
- Specified runtime behavior on illegal operations

Specifying the floating-point format to this level of detail insures that when a computer system is compliant with the standard, users can expect repeatable execution from one hardware platform to another when operations are executed in the same order.

3 IEEE Storage Format

The two most common IEEE floating-point formats in use are 32- and 64-bit numbers. Table 1: Parameters of IEEE 32- and 64-Bit Formats gives the general parameters of these data types.

Parameters of IEEE 32- and 64-Bit Formats

IEEE75	FORTRAN	C	Bits	Exponent Bits	Mantissa Bits
Single	REAL*4	float	32	8	24
Double	REAL*8	double	64	11	53
Double-Extended	REAL*10	long double	>=80	>=15	>=64

Table 1

In FORTRAN, the 32-bit format is usually called REAL, and the 64-bit format is usually called DOUBLE. However, some FORTRAN compilers double the sizes for these data types. For that reason, it is safest to declare your FORTRAN variables as `REAL*4` or `REAL*8`. The double-extended format is not as well supported in compilers and hardware as the single- and double-precision formats. The bit arrangement for the single and double formats are shown in Figure 1 (IEEE754 floating-point formats).

Based on the storage layouts in Table 1: Parameters of IEEE 32- and 64-Bit Formats, we can derive the ranges and accuracy of these formats, as shown in Table 2.

IEEE754 floating-point formats

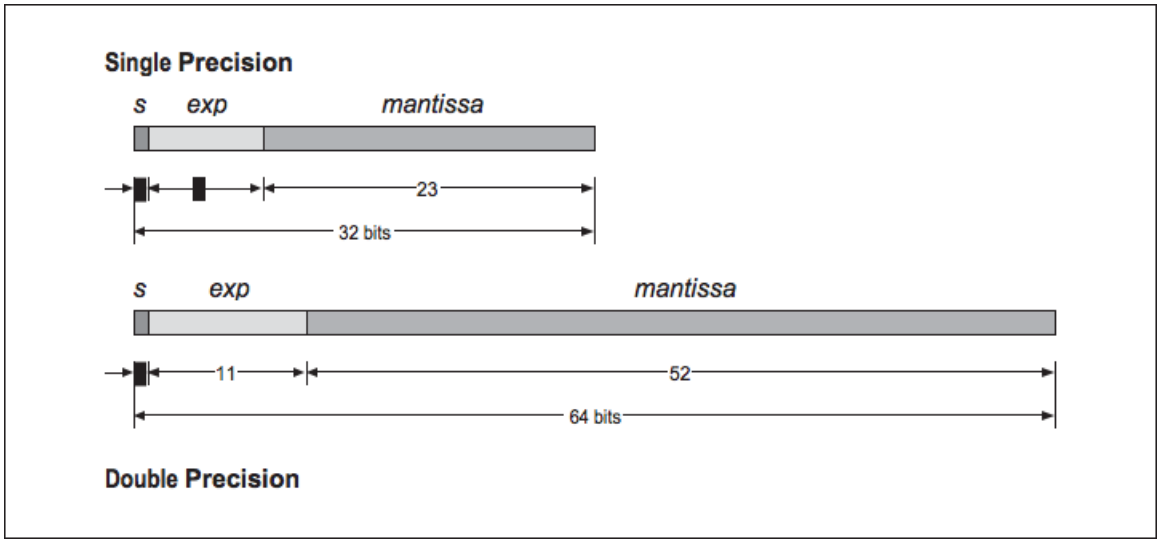

Figure 1

IEEE754	Minimum Normalized Number	Largest Finite Number	Base-10 Accuracy
Single	1.2E-38	3.4 E+38	6-9 digits
Double	2.2E-308	1.8 E+308	15-17 digits
Extended Double	3.4E-4932	1.2 E+4932	18-21 digits

Table 2: Range and Accuracy of IEEE 32- and 64-Bit Formats

3.1 Converting from Base-10 to IEEE Internal Format

We now examine how a 32-bit floating-point number is stored. The high-order bit is the sign of the number. Numbers are stored in a sign-magnitude format (i.e., not 2's - complement). The exponent is stored in the 8-bit field biased by adding 127 to the exponent. This results in an exponent ranging from -126 through +127.

The mantissa is converted into base-2 and normalized so that there is one nonzero digit to the left of the binary place, adjusting the exponent as necessary. The digits to the right of the binary point are then stored in the low-order 23 bits of the word. Because all numbers are normalized, there is no need to store the leading 1.

This gives a free extra bit of precision. Because this bit is dropped, it's no longer proper to refer to the stored value as the mantissa. In IEEE parlance, this mantissa minus its leading digit is called the *significand*.

Figure 2 (Converting from base-10 to IEEE 32-bit format) shows an example conversion from base-10 to IEEE 32-bit format.

Converting from base-10 to IEEE 32-bit format

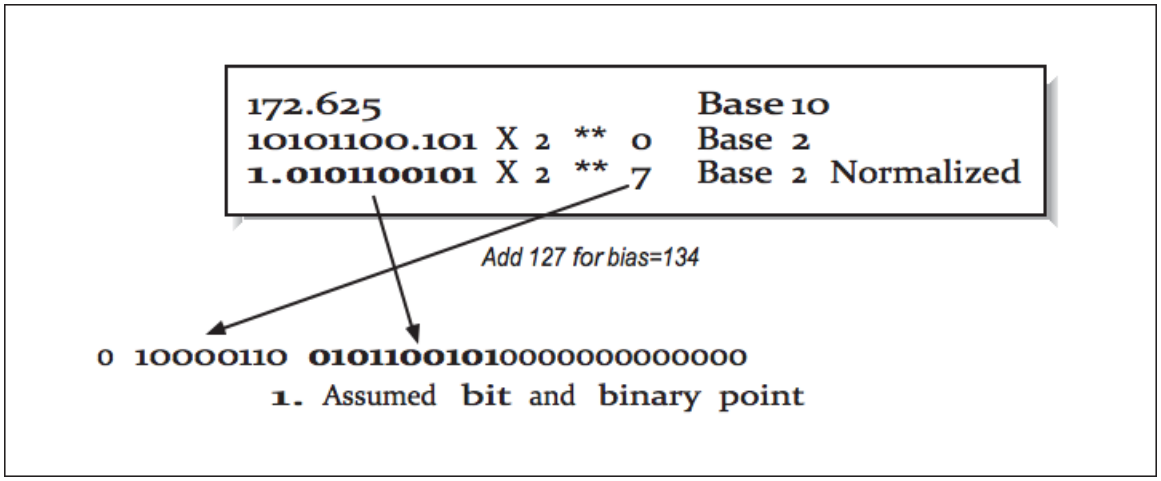

Figure 2

The 64-bit format is similar, except the exponent is 11 bits long, biased by adding 1023 to the exponent, and the significand is 54 bits long.