1.11 Write the number 0.001220703125 in the following forms (in part (*c*) follow the IEEE-754 standard):
(*a*) Binary form. (*b*) Base 2 floating point representation. (*c*) 64 bit double-precision string.

Solution

(*a*) The largest power of 2 that can be divided into 0.001220703125 is $2^{-10} = 0.0009765625$. Subtract $0.001220703125 - 2^{-10} = 0.000244140625$. The highest power of 2 that divides into 0.000244140625 is $2^{-12} = 0.000244140625$. Thus, the number 0.001220703125 in binary form is

$$0.001220703125 = 1\times 2^{-10} + 1\times 2^{-12}$$

or 0.000000000101.

(*b*) Using part (*a*), the binary floating point representation of 0.001220703125 is:

$$\frac{0.001220703125}{2^{-10}}\times 2^{-10} = \frac{0.001220703125}{0.0009765625}\times 2^{-10} = 1.25\times 2^{-10}$$

(*c*) According to the IEEE-754 standard, 0.001220703125 in double precision form is as follows:

- Since the number is positive, the first bit is 0
- From part (*b*), the exponent is -10. Adding a bias of 1023, the value of the exponent that must be stored is -10+1023 = 1013. The number 1013 in binary form is:

$$\begin{aligned} 1013 &= 1\times 2^9 + 1\times 2^8 + 1\times 2^7 + 1\times 2^6 + 1\times 2^5 + 1\times 2^4 + 1\times 2^2 + 1\times 2^0 \\ &= 512 + 256 + 128 + 64 + 32 + 16 + 4 + 1 \end{aligned}$$

  Thus the number 1013 in binary form is 1111110101. In double precision, 11 bits can be used to store the exponent so that 1013 is stored as 01111110101 without the need for rounding or chopping.
- Next, the mantissa 0.25 is easily converted to binary form:

$$1\times 2^{-2}$$

  or, 0.01.
- Since 52 bits are allocated for the mantissa, the binary number stored is 0100000000000000000000000000000000000000000000000000

Thus, the number 0.001220703125 in double precision is stored as: |0|01111110101|0100000000000000000000000000000000000000000000000000|.