**Storing Numbers in the Computer**

Integers are stored using sign-magnitude format while floating point numbers are stored using the IEEE 754 standard.

|  |  |
| --- | --- |
| 1 bit | 31 bits or 63 bits (depending on machine operating system |
| Sign | Magnitude |

Note: sign bit is 0 for positive numbers and 1 for negative numbers

IEEE 754 single precision uses 32 bits while double precision uses 64 bits. There is also an 80 bit data implementation specified by IEEE 754.

**Single Precision**

|  |  |  |
| --- | --- | --- |
| 1 bit | 8 bits (bias = 127) | 23 bits |
| Sign | Exponent | Mantissa |

**Double Precision**

|  |  |  |
| --- | --- | --- |
| 1 bit | 11 bits (bias = 1023) | 52 bits |
| Sign | Exponent | Mantissa |

Note: sign bit is 0 for positive numbers and 1 for negative numbers

**Converting to IEEE 754 Form**

*Put 0.085 in single-precision format*

1. **The first step is to look at the sign of the number.**  
   Because 0.085 is positive, the sign bit =0.

**Note:** A negative number would have a sign bit of 1

1. **Write 0.085 in base-2 scientific notation.**  
   This means that we must factor it into a number in the range [1 <= n < 2] and a power of 2.  
     
   0.085 / 2power = (1+fraction)  
     
   So we can divide 0.085 by a power of 2 to get the (1 + fraction).   
     
   0.085 / 2-1 = 0.17  
   0.085 / 2-2 = 0.34  
   0.085 / 2-3 = 0.68  
   **0.085 / 2-4 = 1.36**  
     
   Therefore, 0.085 = 1.36 \* 2-4
2. **Find the exponent.**  
   The power of 2 is -4, and the bias for the single-precision format is 127. This means that the exponent = 123ten, or 01111011bin
3. **Write the fraction in binary form**  
   The fraction = 0.36 . Unfortunately, this is not a "pretty" number, like those shown in the book. The best we can do is to approximate the value. Single-precision format allows 23 bits for the fraction.  
     
   Binary fractions look like this:  
     
   0.1 = (1/2) = 2-1  
   0.01 = (1/4) = 2-2  
   0.001 = (1/8) = 2-3  
     
   To approximate 0.36, we can say:  
     
   0.36 = (0/2) + **(1/4)** + (0/8) + **(1/16)** + **(1/32)** +...  
   0.36 = 2-2 + 2-4 + 2-5+...   
     
   0.36ten ~ 0.01011100001010001111011bin .  
     
   The binary string we need is: 01011100001010001111011.  
     
   It's important to notice that you will not get 0.36 exactly. This is why floating-point numbers have error when you put them in IEEE 754 format.
4. **Now put the binary strings in the correct order -**  
   1 bit for the sign, followed by 8 for the exponent, and 23 for the fraction. The answer is:

|  |  |  |  |
| --- | --- | --- | --- |
|  | **Sign** | **Exponent** | **Fraction** |
| **Decimal** | 0 | 123 | 0.36 |
| **Binary** | **0** | **01111011** | **01011100001010001111011** |