CS 140

Class Notes 2

1 IEEE 754 Number Representation

As you can see in your textbook, the IEEE754 Floating Point representation is composed of three parts, the Mantissa Sign, S, the Signed Exponent, E, and the Mantissa Magnitude, M. In single precision floating point representation, the Signed Exponent, E, is 8 bits, whereas the Mantissa Magnitude, M, is composed of the remaining 23 bits. In double precision floating point representation, the Signed Exponent, E is 11 bits, whereas the Mantissa Magnitude, M, is composed of the remaining 52 bits. In both cases, the hidden-1 representation for the Mantissa Magnitude holds, effectively extending its representational power by one bit.

The value of a single precision IEEE754 Floating Point number is typically given by the following formula:

$$N = (-1)^S 2^{E-127}(1.M) \tag{1}$$

Yet, one of the things to keep in mind is that this interpretation only holds for $0 < E < 255$. For $E = 0$ (i.e., E being the bit string "00000000") and for $E = 1$ (i.e., E being the bit string "11111111") alternate value interpretations hold as given below.

Condition	N value
$E = 255$ and $M \neq 0$	NaN
$E = 255$ and $M = 0$	$(-1)^S\infty$
$E = 0$ and $M \neq 0$	$(-1)^S 2^{-126}(0.M)$
$E = 0$ and $M = 0$	$(-1)^S 0$

Similarly, the following interpretations hold for the case of *double precision* IEEE754 Floating Point numbers:

Condition	N value
$E = 2047$ and $M \neq 0$	NaN
$E = 2047$ and $M = 0$	$(-1)^S\infty$
$0 < E < 2047$	$(-1)^S 2^{E-1023}(1.M)$
$E = 0$ and $M \neq 0$	$(-1)^S 2^{-1022}(0.M)$
$E = 0$ and $M = 0$	$(-1)^S 0$

Adding and subtraction are the most difficult of the elementary operations for floating-point operands. Here, we deal only with addition, since subtraction can be converted to addition by flipping the sign of the subtrahend. Consider the addition:

$$(\pm s1 \times b^{e1}) + (\pm s2 \times b^{e2}) = \pm s \times b^{e} \tag{2}$$

Assuming $e1 \geq e2$, we begin by aligning the two operands through right-shifting of the significand $s2$ of the number with the smaller exponent:

$$\pm s2 \times b^{e2} = \frac{\pm s2}{b^{e1-e2}} \times b^{e1} \tag{3}$$

If the exponent base b and the number representation radix r are the same, we simply shift $s2$ to the right by $e1 - e2$ digits. When $b = r^a$ the shift amount, which is computed through direct subtraction of the biased exponents, is multiplied by a. In either case, this step is referred to as alignment shift, or preshift (in contrast to normalization shift or postshift, which is needed when the resulting significand

s is unnormalized). After the alignment shift, the significands of the two operands are added to get the significand of the sum.

When the operand signs are alike, a single-digit normalizing shift is always enough. For example, with IEEE754 format, we have $1 \leq s < 4$, which may have to be reduced by a factor of 2 through a single-bit right shift (and adding 1 to the exponent to compensate). However, when the operands have different signs, the resulting significand may be very close to 0 and left shifting by many positions may be needed for normalization.

Figure 1 shows a floating-point addition example:

	E=10001010;	S=1.11100000000000000000000	
+	E=10001000;	S=1.10000000000000000000000	
	E=10001010;	S=1.11100000000000000000000	
+	E=10001010;	S=0.01100000000000000000000	Alignment shifting
	E=10001010;	S=10.01000000000000000000000	Sum
	E=10001011;	S=1.00100000000000000000000	Normalization

Figure 1: floating-point addition

Figure 2 shows a floating-point subtraction example:

	E=10001010;	S=1.11100000000000000000000	
-	E=10001010;	S=1.11000000000000000000000	
	E=10001010;	S=0.00100000000000000000000	Difference
	E=10000111;	S=1.00000000000000000000000	Normalization

Figure 2: floating-point subtraction

Floating-point multiplication is simpler than floating-point addition; it is performed by multiplying the significands and adding the exponents:

$$(\pm s1 \times b^{e1}) \times (\pm s2 \times b^{e2}) = \pm(s1 \times s2)b^{e1+e2} \tag{4}$$

Postshifting may be needed, since the product $s1 \times s2$ of the two significands can be unnormalized. For example, with the IEEE format, we have $1 \leq s1 \times s2 < 4$, leading to the possible need for a single-bit right shift. Also, the computed exponent needs adjustment if a normalization shift is performed.

Figure 3 shows a floating-point multiplication example:

	E=10001010;	S=1.10000000000000000000000	
*	E=00010001;	S=1.10000000000000000000000	
	E=10011011;	S=10.01000000000000000000000	Product
	E=10011100;	S=1.00100000000000000000000	Normalization

Figure 3: floating-point multiplication

Similarly, floating-point division is performed by dividing the significands and subtracting the exponents:

$$\frac{\pm s1 \times be1}{\pm s2 \times b^{e2}} = \pm\frac{s1}{s2} \times b^{e1-e2} \tag{5}$$

The ratio $s1/s2$ of the significands may have to be normalized. With the IEEE754 format, we have $1/2 < s1/s2 < 2$ and a single-bit left shift is always adequate. The computed exponent needs adjustment if a normalizing shift is performed.