Computer Architecture Floating Point Notes (Section 3.5)

Binary Real Numbers:

A decimal point shows the dividing line between non-negative and negative exponents in decimal numbers

$15.345 = 1 \times 10^1 + 5 \times 10^0 + 3 \times 10^{-1} + 4 \times 10^{-2} + 5 \times 10^{-3}$

$= 10 + 5 + 3/10 + 4/100 + 5/1000$

A *binary point* shows the dividing line between non-negative and negative exponents in binary numbers

$10.101 = 1 \times 2^1 + 0 \times 2^0 + 1 \times 2^{-1} + 0 \times 2^{-2} + 1 \times 2^{-3}$

$= 2 + 0 + 1/2 + 0/4 + 1/8$

$=$ 2.625 decimal

Normalized Floating Point Representation:

$1.xxxxx \times 2^{yyyy}$

.xxxx	is the *fraction* (or *mantissa*)	given here in binary
yyyy	is the *exponent*	given here in decimal

For a normalized representation, the value of yyyy is increased or decreased to assure that 1 (not 0 or 10 or 11 or 100 etc.) is to the left of the binary point

The fraction represents an unsigned value between 0 and 1 (the sign bit is handled separately, floating point does not use 2's complement)

IEEE 754 Floating-Point Standard:

Single Precision

32 bits -> 1 sign bit, 8 exponent bits, 23 fraction bits

Double Precision

64 bits -> 1 sign bit, 11 exponent bits, 52 fraction bits

Sign Bit

0 for positive, 1 for negative

Exponent Bits

1) 1-254 (single precision) or 1-2046 (double precision)

Bias signed exponent value

1	most negative exponent value
254, 2046	most positive exponent value
127, 1023	exponent value of 0

Using biased notation for exponents and placing bits in the order sign, then exponent, then fraction allows standard integer sorting algorithms to work properly on floating point numbers

2) All zeros

a) Fraction all zeros -> floating point 0.0

b) Fraction not all zeros -> denormalized number (don't worry about what this means - allows for some very large and very small magnitude values)

3) All ones (255 single precision or 2047 double precision)

a) Fraction all zeros -> floating point infinity

b) Fraction not all zeros -> floating point NaN (not a number)

IEEE 754 Single Precision Floating-Point to Decimal Conversion Example:

What is -4.25 decimal in IEEE 754 single precision notation?

-4.25 decimal	=	-100.01 binary	$(1X2^2 + 1X2^{-2})$
	=	$-1.0001 X 2^2$	normalized floating point
Sign bit	=	1	(negative value)
Fraction bits	=	0001 0000 0000 0000 0000 000	(23 bits)
Exponent	=	127 + 2 = 129	(bias plus actual)

Exponent bits = 1000 0001 (8 bits)

IEEE 754 S.P. = 1 1000 0001 0001 0000 0000 0000 0000 000

= 1100 0000 1000 1000 0000 0000 0000 0000

= 0xC088 0000

Floating-Point Addition:

1.xxxx is called the ***significand*** (1 + fraction)

1) Shift significand of number with smaller exponent right by difference in exponent values

2) Add the signifcands

3) Normalize the sum of significands

4) Determine if overflow/underflow occurs (exponent too large or too small)

5) Round such that fraction has the proper number of bits

Round values half way between representations up such that rounded values have the same mean as the unrounded values

If the bit to the right of rounded LSb is

0 -> truncate
1 -> truncate and add 1 to the LSb

1.11<u>0</u>0	(1.75 decimal)	->	1.11	(1.75 decimal)
1.11<u>0</u>1	(1.8125 decimal)	->	1.11	(1.75 decimal)
1.11<u>1</u>0	(1.875 decimal)	->	10.00	(2.0 decimal)
1.11<u>1</u>1	(1.9375 decimal)	->	10.00	(2.0 decimal)
10.0<u>0</u>0	(2.0 decimal)	->	10.00	(2.0 decimal)

Note that adding 1 to LSb can result in un-normalized value

6) If value is no longer normalized, repeat steps 3 and 4

Floating-Point Addition Example:

0x40E0 0000 + 0x3F40 0000 (IEEE 754 S. P.)

0100 0000 1110 0000 0000 0000 0000 0000
\+ 0011 1111 0100 0000 0000 0000 0000 0000

	sign	exponent	fraction
	0	1000 0001	1100 0000 0000 0000 0000 000
+	0	0111 1110	1000 0000 0000 0000 0000 000
	positive	129 = 127 + 2	1100 0000 0000 0000 0000 000
+	positive	126 = 127 - 1	1000 0000 0000 0000 0000 000

1.11×2^2 + 1.1×2^{-1}

1.11×2^2 + 0.0011×2^2 [right 3 = 2 -(-1) places]

$(1.1100 + 0.0011) \times 2^2$

1.1111×2^2

Round fraction to 23 binary places (already done)

Sum : positive exponent is 129 = 127 + 2 fraction is 1111000...

0 1000 0001 1111 0000 0000 ...

0x40F8 0000 (IEEE 754 S. P.)