Information Representation

Number systems and values
Floating point arithmetic

Unsigned Binary

$$Value = \sum_{i=0}^{n-1} b_i \times 2^i$$

11110000 = 240

1111000011110000 = 61,680

Unsigned Binary Patterns

0000 0001 = 1

0000 0010 = 2

0000 0100 = 4

0000 1000 = 8

0000 1010 = 10

0001 0000 = 16

0001 1010 = 16 + 10 = 26

Unsigned Binary, Fixed Point

$$Value = \sum_{i=0}^{n-1} b_i \times 2^i \times 2^{-p}$$

1111.0000 = 15

11110000.11110000 = 240.9375

1111.000011110000 = 15.05859375

1.111000011110000 = 1.88232421875

Eighth's:		Sixteenth's:	
1	0.125	1	0.0625
2	0.250	3	0.1875
3	0.375	5	0.3125
4	0.500	7	0.4375
5	0.625	9	0.5625
6	0.750	11	0.6875
7	0.875	13	0.8125
		15	0.9375

0.1111 1111 1111 1111	base 2
0.FFFF	base 16
0.9999847412109375	base 10

Two’s Complement

$$Value = -b_{n-1} \times 2^{n-1} + \sum_{i=0}^{n-2} b_i \times 2^i$$

11110000 = -16

1111000011110000 = -3,856

Two’s Complement Patterns

0000 0001 = 1

0000 0010 = 2

0000 0100 = 4

0000 1000 = 8

0000 1010 = 10

0001 0000 = 16

0001 1010 = 16 + 10 = 26

Two’s Complement Patterns

1000 0001 = 1 + -128 = -127

1000 0010 = 2 + -128 = -126

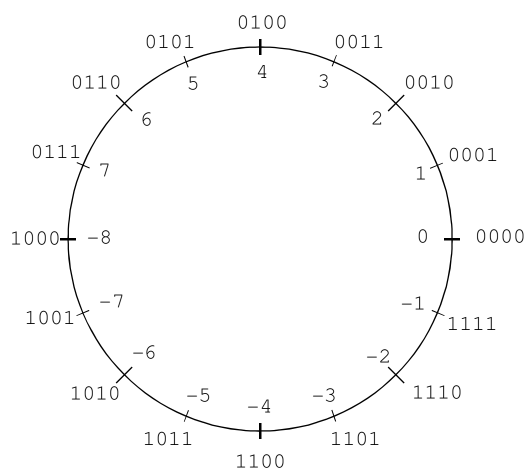
1000 0100 = 4 + -128 = -124

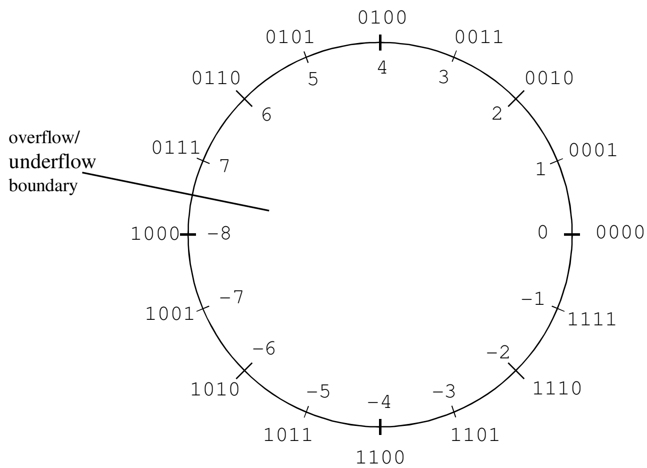
1000 1000 = 8 + -128 = -120

1000 1010 = 10 + -128 = -118

1001 0000 = 16 + -128 = -112

1001 1010 = 16 + 10 + -128 = -102





Two's Complement, Fixed Point

$$Value = (-b_{n-1} \times 2^{n-1} + \sum_{i=0}^{n-2} b_i \times 2^i) \times 2^{-p}$$

1111.0000 = -1

11110000.11110000 = -15.0625

1111.000011110000 = -0.94140625

1.111000011110000 = -0.11767578125

Excess Code

$$Value = StoredVal_{UB} - Excess$$

11110000 in excess 128 = 112

11111110 in excess 127 = 127

00000001 in excess 127 = -126

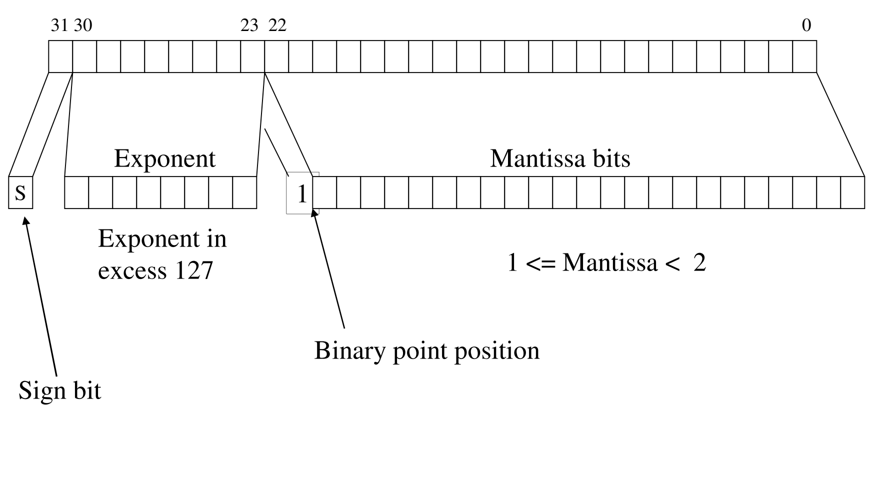
Floating Point Numbers – Coding for Range

- Follows basic scientific notation ideas
- Min, max determined by base, exponent
- Delta R determined by exponent, mantissa
- Number of representable values less than integer methods with same number of bits
- IEEE allows un-normalized numbers close to zero

Floating Point Number

$$Value = (-1)^s \times M \times 2^E$$

IEEE Floating Point Format (32 bit)



Addition of Floating Point Numbers

Add together the following numbers:

1634.75

498.0625

Addition of Floating Point Numbers

1634.75 = 11001100010.1100

= $1.10011000101100 \times 2^{10}$

IEEE: $10 + 127 = 137_{10} = 10001001_2$ so,

0 10001001 10011000101100000000000

Addition of Floating Point Numbers

1634.75 = 11001100010.1100
= $1.10011000101100 \times 2^{10}$

IEEE: $10 + 127 = 137_{10} = 10001001_2$ so,
0 10001001 10011000101100000000000

498.0625 = 111110010.0001
= $1.111100100001 \times 2^{8}$

IEEE: $8 + 127 = 135_{10} = 10000111_2$ so,
0 10000111 11110010000100000000000

Addition of Floating Point Numbers

Step 1: Determine larger of two numbers by
comparing the exponents

Number A: 10001001
Number B: 10000111
A – B = 00000010

Number A is bigger than Number B
by a factor of about 4 (2^2)

Addition of Floating Point Numbers

Step 2: Represent numbers in same format of user's choosing. Choice here is two words (64 bits) with p=56.
Note: keep track of fact that this is x 2^{10}

0000 0001. 1001 1000 1011 0000 0000 0000 0000 0000
0000 0000. 0111 1100 1000 0100 0000 0000 0000 0000

Addition of Floating Point Numbers

Step 3: do the addition:

0000 0001. 1001 1000 1011 0000 0000 0000 0000 0000
0000 0000. 0111 1100 1000 0100 0000 0000 0000 0000

0000 0010. 0001 0101 0011 0100 0000 0000 0000 0000

Addition of Floating Point Numbers

Step 4: Post normalize: (restore normalized condition)
and adjust exponent
0000 0001 0000 1010 1001 1010 0000 0000 0000 0000
x 2^1

So, final IEEE representation:
0 10001010 00001010100110100000000

Addition of Floating Point Numbers

Add together the following numbers:
1634.75
-1555.55

Addition of Floating Point Numbers

$1634.75 = 11001100010.1100$
$= 1.10011000101100 \times 2^{10}$

IEEE: $10 + 127 = 137_{10} = 10001001_2$ so,
0 10001001 10011000101100000000000

$-1555.55 = 11000010011.10001100110011001100$
$= 1.100001001110001100110011001100 \times 2^{10}$

IEEE: $10 + 127 = 137_{10} = 10001001_2$ so,
1 10001001 10000100111000110011001

Addition of Floating Point Numbers

Step 1: Determine larger of two numbers by comparing the exponents

Number A: 10001001
Number B: 10001001
A – B = 00000000

Number A is same order of magnitude as Number B; no alignment necessary

Addition of Floating Point Numbers

Step 2: Represent numbers in same format of user’s choosing. Choice here is two words (64 bits) with p=56.
Note: keep track of fact that this is x 2^{10}

0000 0001. 1001 1000 1011 0000 0000 0000 0000 0000
0000 0001. 1000 0100 1110 0011 0011 0011 0011 0000

Addition of Floating Point Numbers

Step 3: do the addition (in this case, subtraction):

0000 0001. 1001 1000 1011 0000 0000 0000 0000 0000
0000 0001. 1000 0100 1110 0011 0011 0011 0011 0000

0000 0000. 0001 0011 1100 1100 1100 1100 1101 0000

Addition of Floating Point Numbers

Step 4: Post normalize: (restore normalized condition)
and adjust exponent

0000 0001 0011 1100 1100 1100 1100 1101 0000
x 2^{-4}

So, final IEEE representation:

0 10000101 00111100110011001100110

Floating Point Addition

$$
\begin{aligned}
Value &= A + B \\
&= M_A \times 2^{EXP_A} + M_B \times 2^{EXP_B} \\
&= M_A \times 2^{EXP_A} + M_B \times 2^{EXP_B + EXP_A - EXP_A} \\
&= (M_A + M_B \times 2^{EXP_B - EXP_A}) \times 2^{EXP_A}
\end{aligned}
$$

Steps in Floating Point Addition

- Break out A, B into sign, exponent, mantissa
- Determine which number bigger, smaller
- Align smaller mantissa with respect to larger
- Do addition/subtraction (double precision)
- Do post normalization (and adjust exponent)
- Put result together

Basic Block Diagram – Floating Point Addition