Floating Point Numbers

Review of Numbers

- Computers are made to deal with numbers
- What can we represent in N bits?
  - Unsigned integers:

    0 to $2^N - 1$
  - Signed Integers (Two's Complement)

    $-2^{(N-1)}$ to $2^{(N-1)} - 1$

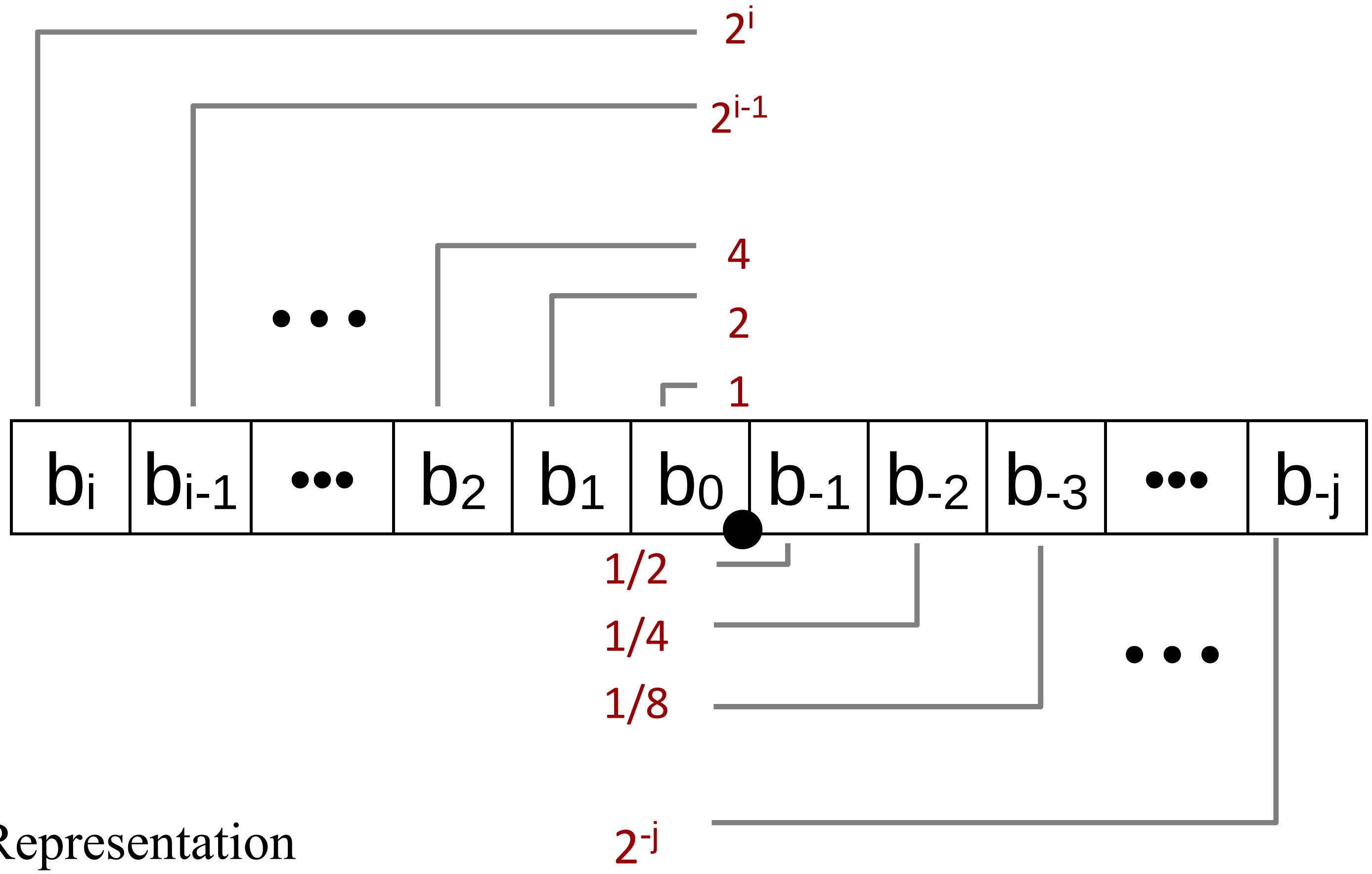
Signed Integers

$-2^{(N-1)} - 1$ to $2^{(N-1)} - 1$

Other Numbers

- What about other numbers?
  - Very large numbers? (seconds/century)
    $3{,}155{,}760{,}000_{10}$ ($3.15576_{10} \times 10^9$)
  - Very small numbers? (atomic diameter)
    0.00000001_{10} ($1.0_{10} \times 10^{-8}$)
  - Rationals (repeating pattern)
  - 2/3 (0.666666666. . .)
  - Irrationals
    $2^{1/2}$ (1.414213562373. . .)
  - Transcendentals
  - e (2.718...), π (3.141...)
- All represented in scientific notation

Fractional Binary Numbers



- Representation
  - Bits to right of "binary point" represent fractional powers of 2
  - Represents rational number:

$$\sum_{k=-j}^{i} b_k \times 2^k$$

Fractional Binary Numbers: Examples

- Value / Representation

Value		Representation	
5 3/4	= 23/4	$\texttt{101.11}_2$	= 4 + 1 + 1/2 + 1/4
2 7/8	= 23/8	$\texttt{10.111}_2$	= 2 + 1/2 + 1/4 + 1/8
1 7/16	= 23/16	$\texttt{1.0111}_2$	= 1 + 1/4 + 1/8 + 1/16

- Observations
  - Divide by 2 by shifting right (unsigned)
  - Multiply by 2 by shifting left
  - Numbers of form $0.111111\ldots_2$ are just below 1.0
    - $1/2 + 1/4 + 1/8 + \ldots + 1/2^i + \ldots \rightarrow 1.0$
    - Use notation $1.0 - \varepsilon$

Representable Numbers

- Limitation #1
  - Can only exactly represent numbers of the form $x/2^k$
    - Other rational numbers have repeating bit representations

  - Value — Representation
    - 1/3 — **`0.0101010101[01]…`**$_2$
    - 1/5 — **`0.001100110011[0011]…`**$_2$
    - 1/10 — **`0.0001100110011[0011]…`**$_2$

- Limitation #2
  - Just one setting of binary point within the *w* bits
    - Limited range of numbers (very small values? very large?)

Objective

- To understand the fundamentals of floating-point representation
- To know the IEEE-754 Floating Point Standard

Patriot Missile

- Gulf War I
- Failed to intercept incoming Iraqi scud missile (Feb 25, 1991)
- 28 American soldiers killed

GAO Report: GAO/IMTEC-92-26 Patriot Missile Software Problem
http://www.fas.org/spp/starwars/gao/im92026.htm

Patriot Design

- Intended to operate only for a few hours
  - Defend Europe from Soviet aircraft and missile
- Four 24-bit registers (1970s design!)
- Kept time with integer counter: incremented every 1/10 second
- Calculate speed of incoming missile to predict future positions:

  $$\text{velocity} = \text{loc}_1 - \text{loc}_0/(\text{count}_1 - \text{count}_0) * 0.1$$

- But, cannot represent 0.1 exactly!

Floating Imprecision

- 24-bits:

$$0.1 = 1/2^4 + 1/2^5 + 1/2^8 + 1/2^9$$
$$+ 1/2^{12} + 1/2^{13} + 1/2^{16} + 1/2^{17}$$
$$+ 1/2^{20} + 1/2^{21}$$
$$= 209715 \,/\, 2097152$$

Error is 0.2/2097152 = 1/10485760

One hour = 3600 seconds

3600 * 1/10485760 * 10 = 0.0034s

20 hours = 0.0687s

Miss target! (137 meters)

Two weeks before the incident, Army officials received Israeli data indicating some loss in accuracy after the system had been running for 8 consecutive hours. Consequently, Army officials modified the software to improve the system's accuracy. However, the modified software did not reach Dhahran until February 26, 1991--the day after the Scud incident.

GAO Report

http://fas.org/spp/starwars/gao/im92026.htm

Floating Point Representation

Example:
$15213_{10} = (-1)^0 \times 1.1101101101101_2 \times 2^{13}$

- Numerical Form:

$$(-1)^s \, M \; 2^E$$

  - Sign bit s determines whether number is negative or positive
  - Significand M normally a fractional value in range [1.0,2.0).
  - Exponent E weights value by power of two

- Encoding
  - MSB s is sign bit s
  - exp field encodes E (but is not equal to E)
  - frac field encodes M (but is not equal to M)

s	exp	frac

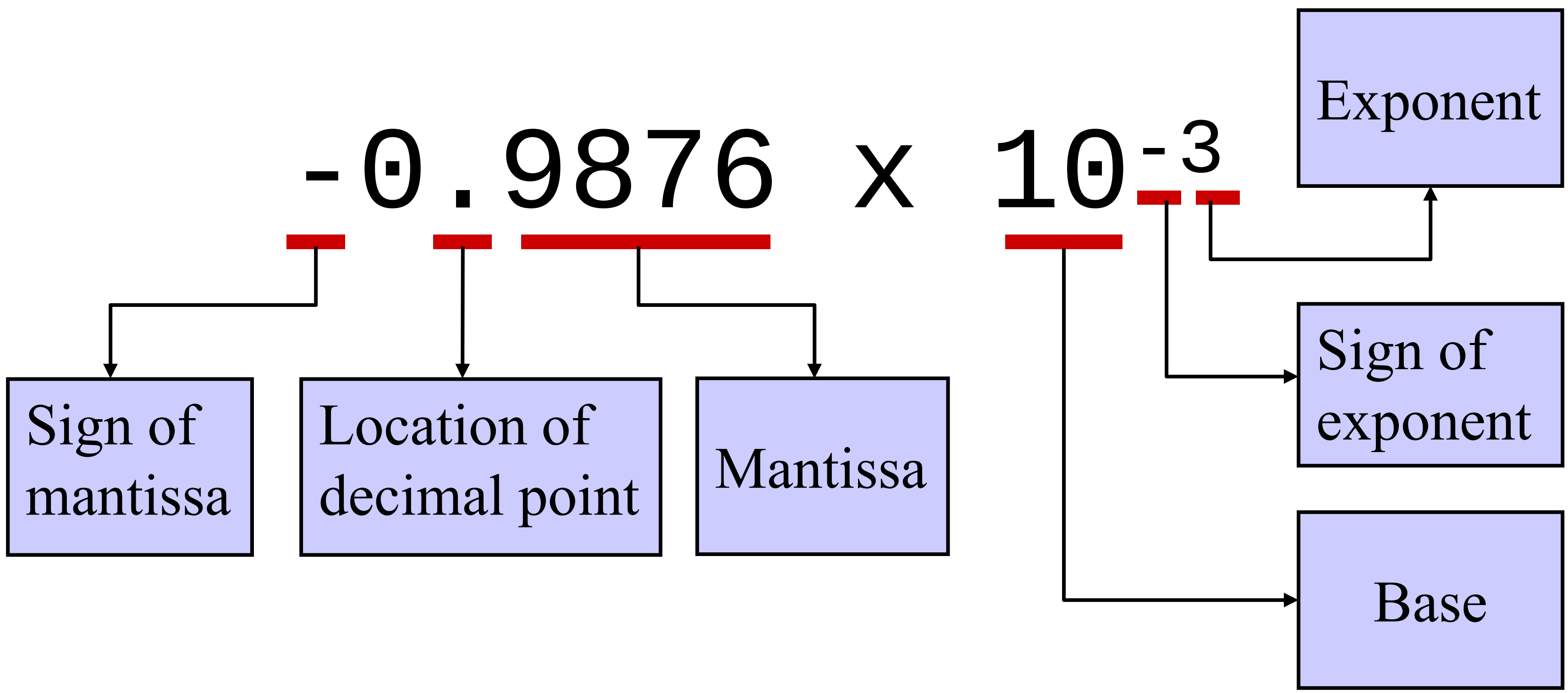
Exponential Notation

- The following are equivalent representations of 1,234

123,400.0	x	10^{-2}
12,340.0	x	10^{-1}
1,234.0	x	10^{0}
123.4	x	10^{1}
12.34	x	10^{2}
1.234	x	10^{3}
0.1234	x	10^{4}

The representations differ in that the decimal place – the “point” -- “floats” to the left or right (with the appropriate adjustment in the exponent).

Parts of a Floating Point Number

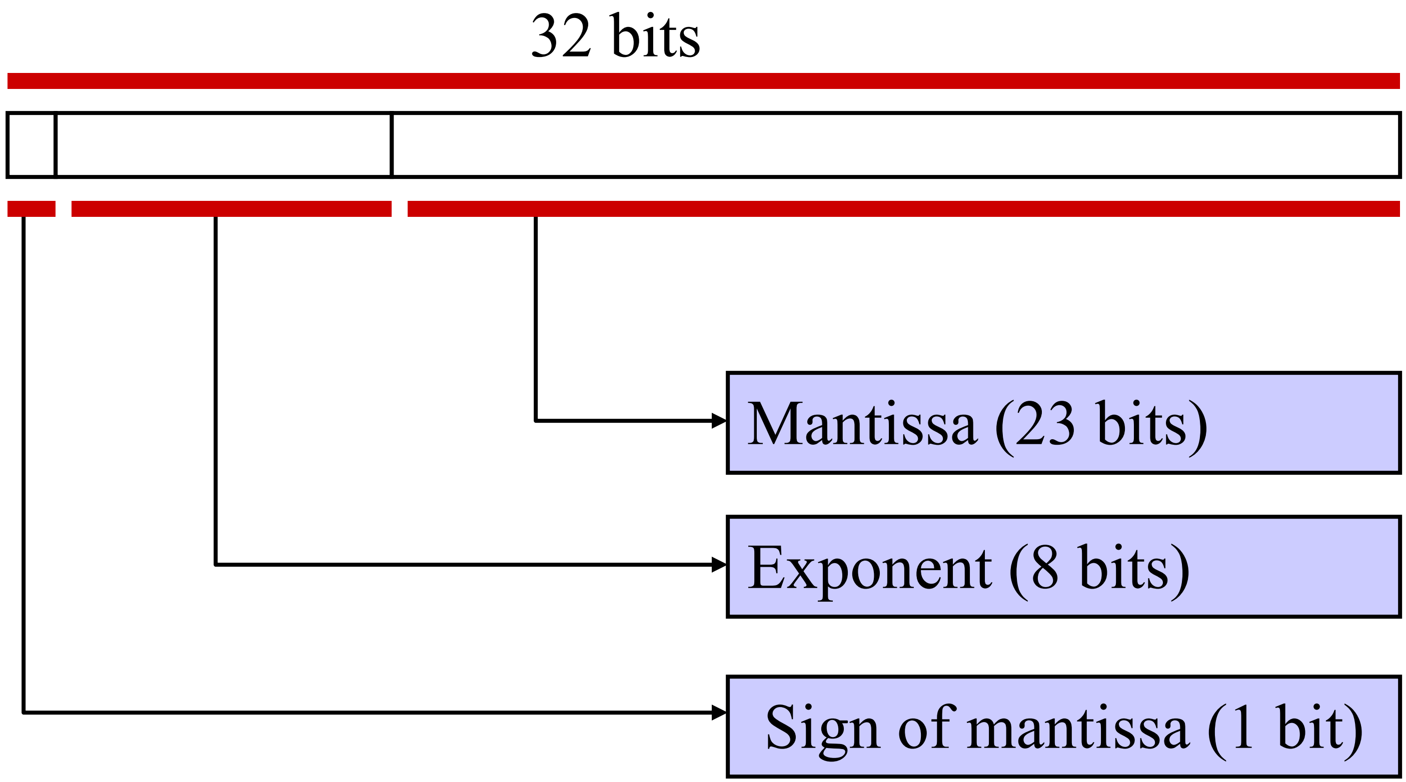


IEEE 754 Standard

- Most common standard for representing floating point numbers
- Single precision: 32 bits, consisting of...
    - Sign bit (1 bit)
    - Exponent (8 bits)
    - Mantissa (23 bits)
- Double precision: 64 bits, consisting of…
    - Sign bit (1 bit)
    - Exponent (11 bits)
    - Mantissa (52 bits)

Prof. Willian Kahan

Single Precision Format



Normalization

- The mantissa is *normalized*
- Has an implied decimal place on left
- Has an implied "1" on left of the decimal place
- E.g.,
  - Mantissa → `10100000000000000000000`
  - Represents… $1.101_2 = 1.625_{10}$
- Normalized form: no leadings 0s
  (exactly one digit to left of decimal point)
  - Normalized: 1.0×10^{-9}
  - Not normalized: $0.1 \times 10^{-8}, 10.0 \times 10^{-10}$

Excess Notation

- To include +ve and –ve exponents, “excess” notation is used
- Single precision: excess 127
- Double precision: excess 1023
- The value of the exponent stored is larger than the actual exponent
- E.g., excess 127,
  - Exponent $\rightarrow$ `10000111`
  - Represents… `135 – 127 = 8`

Example

- Single precision

`0 10000010 11000000000000000000000`

1.11_2

$130 - 127 = 3$

0 = positive mantissa

$$+1.11_2 \times 2^3 = 1110.0_2 = 14.0_{10}$$

Hexadecimal

- It is convenient and common to represent the original floating point number in hexadecimal
- The preceding example…

```
0 10000010 11000000000000000000000
```

0100	0001	0110	0000	0000	0000	0000	0000
4	1	6	0	0	0	0	0

Converting <u>from</u> Floating Point

- E.g., What decimal value is represented by the following 32-bit floating point number?

$$\texttt{C17B0000}_{16}$$

- Step 1
  - Express in binary and find S, E, and M

$C17B0000_{16}$ –

$\underbrace{1}_{S}\ \underbrace{10000010}_{E}\ \underbrace{11110110000000000000000}_{M}{}_2$

1 = negative
0 = positive

- Step 2
  - Find "real" exponent, n
  - $n = E - 127$

$$\begin{aligned} n &= E - 127 \\ &= 10000010_2 - 127 \\ &= 130 - 127 \\ &= 3 \end{aligned}$$

- Step 3
  - Put S, M, and *n* together to form binary result
  - (Don’t forget the implied “1.” on the left of the mantissa.)

$-1.1111011_2 \times 2^n =$

$-1.1111011_2 \times 2^3 =$

-1111.1011_2

- Step 4
  - Express result in decimal

$$\underbrace{-1111}_{-15}.\underline{1011}_2$$

$$
\begin{aligned}
2^{-1} &= 0.5 \\
2^{-3} &= 0.125 \\
2^{-4} &= \underline{0.0625} \\
& 0.6875
\end{aligned}
$$

Answer: -15.6875

Converting from Floating Point

- E.g., What decimal value is represented by the following 32-bit floating point number?

42808000_{16}

Converting to Floating Point

- E.g., Express 36.5625_{10} as a 32-bit floating point number (in hexadecimal)

- Step 1
  - Express original value in binary

    $36.5625_{10} =$

    100100.1001_2

- Step 2
  - Normalize

$100100.1001_2 =$

$1.001001001_2 \times 2^5$

- Step 3
  - Determine S, E, and M

$$\underbrace{+}_{S}1.\underbrace{001001001}_{M}{}_2 \times 2^{\underbrace{5}_{n}}$$

$$\begin{aligned} E &= n + 127 \\ &= 5 + 127 \\ &= 132 \\ &= 10000100_2 \end{aligned}$$

S = 0 (because the value is positive)

- Step 4
  - Put S, E, and M together to form 32-bit binary result

$$\underbrace{\texttt{0}}_{S}\ \underbrace{\texttt{10000100}}_{E}\ \underbrace{\texttt{00100100100000000000000}}_{M}{}_2$$

- Step 5
  - Express in hexadecimal

$0\ 10000100\ 00100100100000000000000_2 =$

$0100\ 0010\ 0001\ 0010\ 0100\ 0000\ 0000\ 0000_2 =$

$4\quad 2\quad 1\quad 2\quad 4\quad 0\quad 0\quad 0_{16}$

Answer: 42124000_{16}

Converting <u>to</u> Floating Point

- E.g., Express 6.5_{10} as a 32-bit floating point number (in hexadecimal)

Converting <u>to</u> Floating Point

- E.g., Express 0.1 as a 32-bit floating point number (in hexadecimal)

Zero, Infinity, and NaN

- Zero
  - Exponent field $E = 0$ and fraction $F = 0$
  - +0 and –0 are possible according to sign bit S
- Infinity
  - Infinity is a special value represented with maximum E and $F = 0$
    - For single precision with 8-bit exponent: maximum $E = 255$
    - For double precision with 11-bit exponent: maximum $E = 2047$
  - Infinity can result from overflow or division by zero
  - $+\infty$ and $-\infty$ are possible according to sign bit S
- NaN (Not a Number)
  - NaN is a special value represented with maximum E and $F \neq 0$
  - Result from exceptional situations, such as 0/0 or sqrt(negative)
  - Operation on a NaN results is NaN: Op(X, NaN) = NaN

Simple 6-bit Floating Point Example

S	Exponent^3	Fraction^2

- 6-bit floating point representation
  - Sign bit is the most significant bit
  - Next 3 bits are the exponent with a bias of 3
  - Last 2 bits are the fraction
- Same general form as IEEE
  - Normalized, denormalized
  - Representation of 0, infinity and NaN
- Value of normalized numbers $(-1)^S \times (1.F)_2 \times 2^{E-3}$
- Value of denormalized numbers $(-1)^S \times (0.F)_2 \times 2^{-2}$

Values Related to Exponent

Exp.	exp	E	2^E	
0	000	-2	¼	Denormalized
1	001	-2	¼	Normalized
2	010	-1	½	
3	011	0	1	
4	100	1	2	
5	101	2	4	
6	110	3	8	
7	111	n/a		Inf or NaN

Dynamic Range of Values

s	exp	frac	E	value	
0	000	00	-2	0	
0	000	01	-2	1/4*1/4=1/16	smallest denormalized
0	000	10	-2	2/4*1/4=2/16	
0	000	11	-2	3/4*1/4=3/16	largest denormalized
0	001	00	-2	4/4*1/4=4/16=1/4=0.25	smallest normalized
0	001	01	-2	5/4*1/4=5/16	
0	001	10	-2	6/4*1/4=6/16	
0	001	11	-2	7/4*1/4=7/16	
0	010	00	-1	4/4*2/4=8/16=1/2=0.5	
0	010	01	-1	5/4*2/4=10/16	
0	010	10	-1	6/4*2/4=12/16=0.75	
0	010	11	-1	7/4*2/4=14/16	

Dynamic Range of Values

s	exp	frac	E	value
0	011	00	0	4/4*4/4=16/16=1
0	011	01	0	5/4*4/4=20/16=1.25
0	011	10	0	6/4*4/4=24/16=1.5
0	011	11	0	7/4*4/4=28/16=1.75
0	100	00	1	4/4*8/4=32/16=2
0	100	01	1	5/4*8/4=40/16=2.5
0	100	10	1	6/4*8/4=48/16=3
0	100	11	1	7/4*8/4=56/16=3.5
0	101	00	2	4/4*16/4=64/16=4
0	101	01	2	5/4*16/4=80/16=5
0	101	10	2	6/4*16/4=96/16=6
0	101	11	2	7/4*16/4=112/16=7

Dynamic Range of Values

s	exp	frac	E	value
0	110	00	3	4/4*32/4=128/16=8
0	110	01	3	5/4*32/4=160/16=10
0	110	10	3	6/4*32/4=192/16=12
0	110	11	3	7/4*32/4=224/16=14
0	111	00		∞
0	111	01		NaN
0	111	10		NaN
0	111	11		NaN

largest normalized

Floating Point Addition Example

- Consider adding: $(1.111)_2 \times 2^{-1} + (1.011)_2 \times 2^{-3}$
  - For simplicity, we assume 4 bits of precision (or 3 bits of fraction)
- Cannot add significands … Why?
  - Because exponents are not equal
- How to make exponents equal?
  - Shift the significand of the lesser exponent right until its exponent matches the larger number
- $(1.011)_2 \times 2^{-3} = (0.1011)_2 \times 2^{-2} = (0.01011)_2 \times 2^{-1}$
  - Difference between the two exponents = $-1 - (-3) = 2$
  - So, shift right by 2 bits
- Now, add the significands:

```
          1.111
      +   0.01011
      -----------
Carry → 10.00111
```

Addition Example

- So, $(1.111)_2 \times 2^{-1} + (1.011)_2 \times 2^{-3} = (10.00111)_2 \times 2^{-1}$
- However, result $(10.00111)_2 \times 2^{-1}$ is NOT normalized
- Normalize result: $(10.00111)_2 \times 2^{-1} = (1.000111)_2 \times 2^{0}$
  - In this example, we have a carry
  - So, shift right by 1 bit and increment the exponent
- Round the significand to fit in appropriate number of bits
  - We assumed 4 bits of precision or 3 bits of fraction
- Round to nearest: $(1.000111)_2 \approx (1.001)_2$
  - Renormalize if rounding generates a carry
- Detect overflow / underflow
  - If exponent becomes too large (overflow) or too small (underflow)

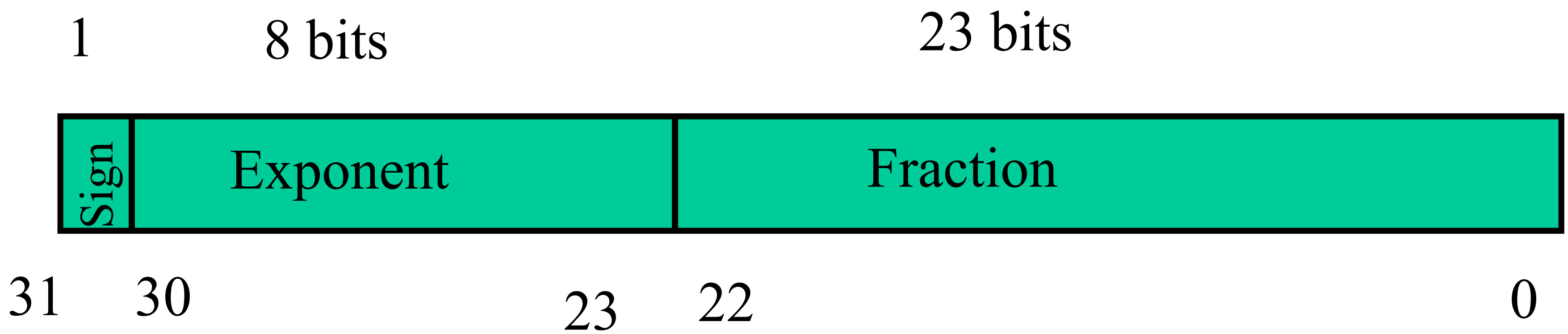
```
  1.000 | 111
+     1
-------
  1.001
```

Item	Single precision	Double precision
Bits in sign	1	1
Bits in exponent	8	11
Bits in fraction	23	52
Bits, total	32	64
Exponent system	Excess 127	Excess 1023
Exponent range	−126 to +127	−1022 to +1023
Smallest normalized number	2^{-126}	2^{-1022}
Largest normalized number	approx. 2^{128}	approx. 2^{1024}
Decimal range	approx. 10^{-38} to 10^{38}	approx. 10^{-308} to 10^{308}
Smallest denormalized number	approx. 10^{-45}	approx. 10^{-324}

Figure B-5. Characteristics of IEEE floating-point numbers.

Summary: IEEE Floating Point

Single Precision (32 bits)



Exponent values:

Exponent values:	0	zeroes
	1-254	exp + 127
	255	infinities, NaN

Value = $(1 - 2*Sign)\ (1 + Fraction)^{Exponent\ -\ 127}$

Denormalized Values

- Condition
  - `exp` = `000…0`
- Value
  - Exponent value $E = -Bias + 1$
  - Significand value $M =$ `0.xxx…x`$_2$
    - `xxx…x`: bits of `frac`
- Cases
  - `exp` = `000…0`, `frac` = `000…0`
    - Represents value 0
    - Note that have distinct values +0 and –0
  - `exp` = `000…0`, `frac` ≠ `000…0`
    - Numbers very close to 0.0

Special Values

- Condition
  - `exp` = `111`…`1`
- Cases
  - `exp` = `111`…`1`, `frac` = `000`…`0`
    - Represents value ∞ (infinity)
    - Operation that overflows
    - Both positive and negative
    - E.g., $1.0/0.0 = -1.0/-0.0 = +\infty$, $1.0/-0.0 = -\infty$
  - `exp` = `111`…`1`, `frac` $\neq$ `000`…`0`
    - Not-a-Number (NaN)
    - Represents case when no numeric value can be determined
    - E.g., sqrt(–1), $\infty - \infty$

Interesting Numbers

Description	`exp`	`frac`	Numeric Value
Zero	00…00	00…00	0.0
Smallest Pos. Denorm.	00…00	00…01	$2^{-\{23,52\}} \times 2^{-\{126,1022\}}$

- Single $\approx 1.4 \times 10^{-45}$
- Double $\approx 4.9 \times 10^{-324}$

Description	`exp`	`frac`	Numeric Value
Largest Denormalized	00…00	11…11	$(1.0 - \varepsilon) \times 2^{-\{126,1022\}}$

- Single $\approx 1.18 \times 10^{-38}$
- Double $\approx 2.2 \times 10^{-308}$

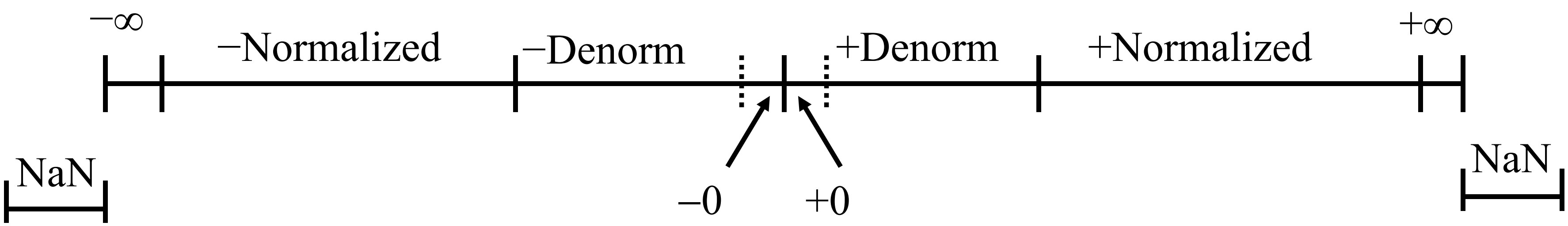
Description	`exp`	`frac`	Numeric Value
Smallest Pos. Normalized	00…01	00…00	$1.0 \times 2^{-\{126,1022\}}$

- Just larger than largest denormalized

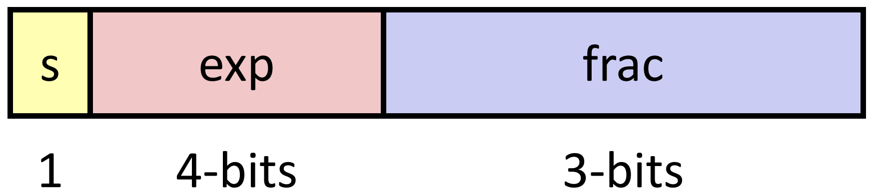
Description	`exp`	`frac`	Numeric Value
One	01…11	00…00	1.0
Largest Normalized	11…10	11…11	$(2.0 - \varepsilon) \times 2^{\{127,1023\}}$

- Single $\approx 3.4 \times 10^{38}$
- Double $\approx 1.8 \times 10^{308}$

Visualization: Floating Point Encodings



Tiny Floating Point Example



- 8-bit Floating Point Representation
  - the sign bit is in the most significant bit
  - the next four bits are the `exp`, with a bias of 7
  - the last three bits are the `frac`

- Same general form as IEEE Format
  - normalized, denormalized
  - representation of 0, NaN, infinity

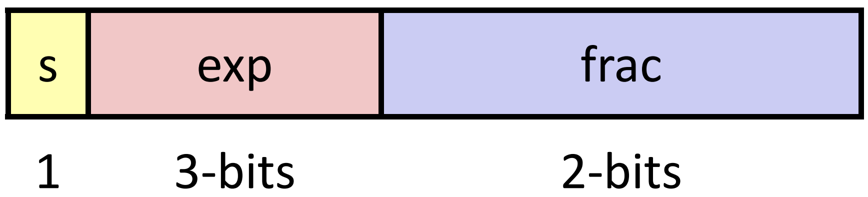
Dynamic Range (s=0 only)

$v = (-1)^s M\ 2^E$
norm: E = exp – Bias
denorm: E = 1 – Bias

	s	exp	frac	E	Value	
	0	0000	000	-6	0	
	0	0000	001	-6	1/8*1/64 = 1/512	closest to zero
Denormalized numbers	0	0000	010	-6	2/8*1/64 = 2/512	$(-1)^0(0+1/4)*2^{-6}$
	…					
	0	0000	110	-6	6/8*1/64 = 6/512	
	0	0000	111	-6	7/8*1/64 = 7/512	largest denorm
	0	0001	000	-6	8/8*1/64 = 8/512	smallest norm
	0	0001	001	-6	9/8*1/64 = 9/512	$(-1)^0(1+1/8)*2^{-6}$
	…					
	0	0110	110	-1	14/8*1/2 = 14/16	
	0	0110	111	-1	15/8*1/2 = 15/16	closest to 1 below
Normalized numbers	0	0111	000	0	8/8*1 = 1	
	0	0111	001	0	9/8*1 = 9/8	closest to 1 above
	0	0111	010	0	10/8*1 = 10/8	
	…					
	0	1110	110	7	14/8*128 = 224	
	0	1110	111	7	15/8*128 = 240	largest norm
	0	1111	000	n/a	inf	

Distribution of Values

- 6-bit IEEE-like format
  - e = 3 exponent bits
  - f = 2 fraction bits
  - Bias is $2^{3-1}-1 = 3$



- Notice how the distribution gets denser toward zero.