CSE 140: Computer Arithmetic Algorithms and Hardware Design

Lecture 19: Floating Point Numbers

Instructor:
Prof. Chung-Kuan Cheng

Motivation

- Maximal information with given bit numbers.
- Arithmetic with proper precision.
- Fairness of rounding.
- Features at the expenses of the complexity of the operations.

Topics:

- Floating Point Numbers (IEEE P754)
  - Standard
  - Operations
  - Exceptional Situations
  - Rounding Modes
- Numerical Computing with IEEE Floating Point Arithmetic, Michael L. Overton, SIAM

Standard

2^{32} → Typically

- Goal: Dynamic Range:

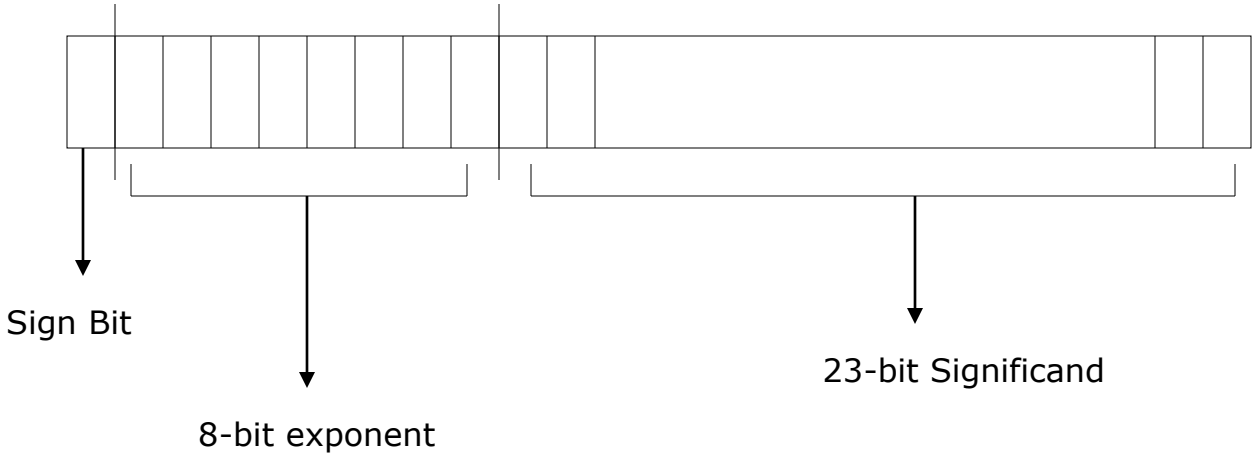
  largest #/ smallest #

- If too large, holes between #'s

Standard

- ulp (unit in the last place)
  - Difference between two consecutive values of the significand.

3 Parts→ x = ~s b^e:sign, significand, exponent



Standard: nomalization

- $\pm e_1e_2e_3e_4e_5e_6e_7e_8s_1s_2s_3\ldots s_{22}s_{23}$
  - $1.s_1s_2s_3\ldots s_{22}s_{23}$ normalized number
  - $0.s_1s_2s_3\ldots s_{22}s_{23}$ denormalized number

Id	$e_1e_2e_3e_4e_5e_6e_7e_8$	
0	0 0 0 0 0 0 0 0	$x=0.s_1s_2s_3\ldots s_{22}s_{23}\ 2^{-126}$
1	0 0 0 0 0 0 0 1	$x=1.s_1s_2s_3\ldots s_{22}s_{23}\ 2^{-126}$
2	0 0 0 0 0 0 1 0	$x=1.s_1s_2s_3\ldots s_{22}s_{23}\ 2^{-125}$
.		
126	0 1 1 1 1 1 1 0	$x=1.s_1s_2s_3\ldots s_{22}s_{23}\ 2^{-1}$
127	0 1 1 1 1 1 1 1	$x=1.s_1s_2s_3\ldots s_{22}s_{23}\ 2^{0}$
128	1 0 0 0 0 0 0 0	$x=1.s_1s_2s_3\ldots s_{22}s_{23}\ 2^{1}$
.		
253	1 1 1 1 1 1 0 1	$x=1.s_1s_2s_3\ldots s_{22}s_{23}\ 2^{126}$
254	1 1 1 1 1 1 1 0	$x=1.s_1s_2s_3\ldots s_{22}s_{23}\ 2^{127}$
255	1 1 1 1 1 1 1 1	x= Inf if $(s_1\ldots s_{23})= 0$, else NaN

NaN → Not a Number

Standard: normalization

$0.01 \times 2^{-3} = 0.001 \times 2^{-2}$

- Same number, so normalize to remove redundancy
- Use a default 1 in front for one more bit precision.
- Smallest Number

$$0.00\ldots01 \times 2^{-126} = 1.0 \times 2^{-23} \times 2^{-126}$$
$$= 1 \times 2^{-149}$$

Standard - Example

±eeeeeeee sssss ssssss ssssss ssssss sss

0 00000000 00000000000000000000000 = 0.000…0x2^{-126}

1 00000000 00000000000000000000000 =-0.000…0x2^{-126}

minimum

0 00000000 00000000000000000000001 = 1x2^{-149}

normalized minimum

0 00000001 00000000000000000000000 = 1.000…0x2^{-126}

0 00000001 00000000000000000000001 = 1.000…1x2^{-126}

.

.

0 01111111 00000000000000000000000 = 1.000…0x2^{0}

0 01111111 00000000000000000000001 = 1.000…1x2^{0}

0 10000000 00000000000000000000001 = 1.000…1x2^{1}

Standard – Example Cont.

0 11111110 00000000000000000000000 = $1.000\ldots0 \times 2^{127}$

0 11111110 00000000000000000000001 = $1.000\ldots1 \times 2^{127}$

0 11111110 11111111111111111111111 = $1.111\ldots1 \times 2^{127}$

- Normalized Maximum

0 11111111 00000000000000000000000 = Inf

Tiniest= 1×2^{-149}

$N_{min} = 1.0 \times 2^{-126}$

$N_{max} = (2 - 2^{-23})2^{127}$

Double Floating Point

$\pm\ e_1e_2\ldots e_{11}\ s_1s_2\ldots s_{52}$

0 00…000 $s_1s_2\ldots s_{52}$ $x=0.s_1s_2\ldots s_{52}\ 2^{-1022}$

0 00…001 $s_1s_2\ldots s_{52}$ $x=1.s_1s_2\ldots s_{52}\ 2^{-1022}$

.

.

0 01…111 $s_1s_2\ldots s_{52}$ $x=1.s_1s_2\ldots s_{52}\ 2^{0}$

0 10…000 $s_1s_2\ldots s_{52}$ $x=1.s_1s_2\ldots s_{52}\ 2^{1}$

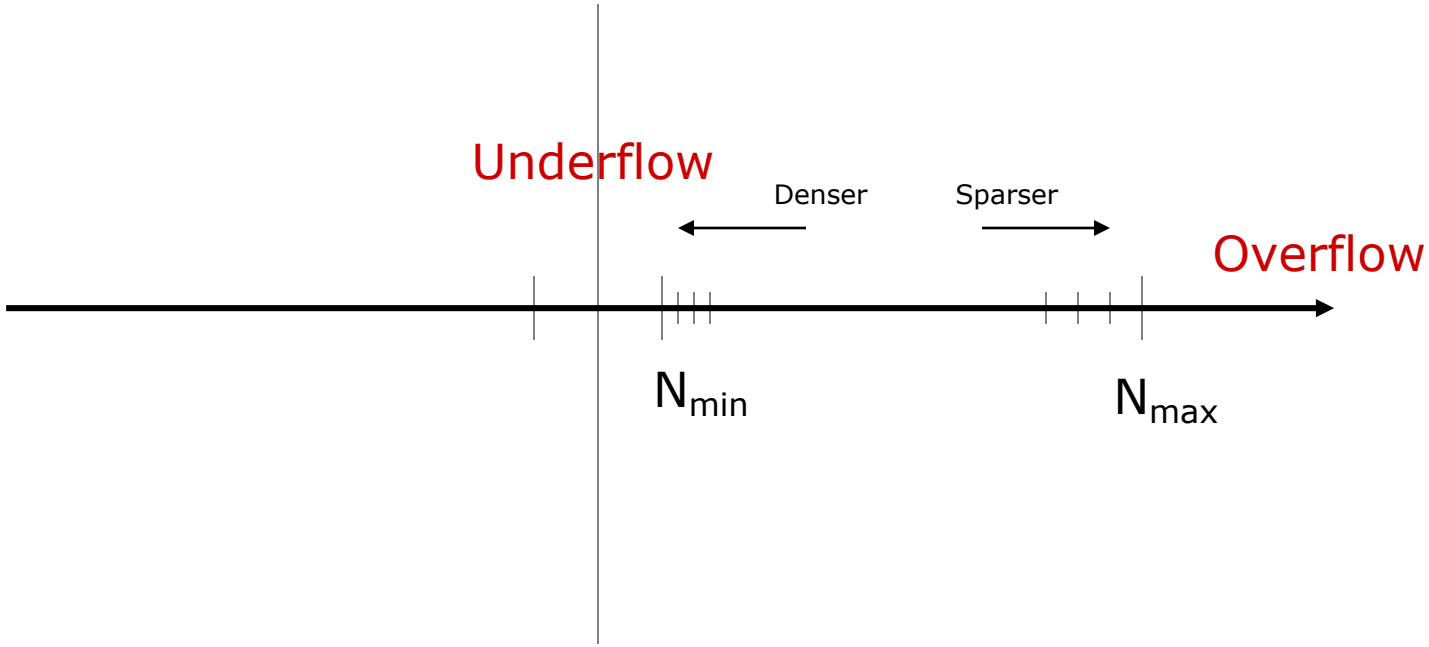
.

.

0 11…110 $s_1s_2\ldots s_{52}$ $x=1.s_1s_2\ldots s_{52}\ 2^{1023}$

0 11…111 $s_1s_2\ldots s_{52}$ $x=\text{Inf if } (s_1\ldots s_{52})=0$

Overflow/Underflow



Addition/Multiplication

- $\sim s_1 x b^{e1} + (\sim s_2 x b^{e2}) = \sim s x b^{e}$

  $= \sim s_1 x b^{e1} + \sim s_2 / b^{e1-e2} \times b^{e1}$

  $= (\sim s_1 + \sim s_2 / b^{e1-e2}) \times b^{e1}$

- $(\sim s_1 x b^{e1}) \times (\sim s_2 x b^{e2}) = \sim (s_1 x s_2) b^{e1+e2}$

Exceptions

a/0 = Inf if a > 0

a/Inf = 0 if a != 0

a·0 = 0

a·Inf = Inf if a > 0

a + Inf = Inf

0·Inf = invalid operation (NaN)

0/0 = invalid operation (NaN)

Inf - Inf = NaN

NaP op a = NaN

Rounding Mode

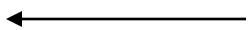
- Adder Output = $C_{out}\ z_1 z_0 . z_{-1} z_{-2} \ldots z_{-l}$ GRS

  Guard Bit

  Round Bit

  Sticky Bit, OR of all bits below bit R

$$
\begin{array}{r}
1.101 \times 2^3 \\
+1.110 \times 2^3 \\
\hline
11.011 \times 2^3 \\
1.1011 \times 2^4
\end{array}
$$



Normalize – need to round ↑ or ↓

Rounding

$1.110\ 2^3$

$-\ 1.101\ 2^3$

$0.001\ 2^3$

$1.000\ 2^0$ normalize

$1.101\ 2^3$

$-\ 1.101\ 2^2$

$1.101\ 2^3$

$-\ 0.1101\ 2^3$ Guard bit

$0.1101\ 2^3$

$1.101\ 2^2$

Rounding

- Round to the nearest even
  - 1.10111
  - toward 0 1.1011
  - Toward +Inf 1.1100
  - Toward -Inf 1.1011

Conventional Rounding Error

Rounding	Error
1.10100 → 1.101	= 0
1.10101 → 1.101	= -0.25
1.10110 → 1.110	= +0.5
1.10111 → 1.110	= +0.25

Average Error = 0.5/4 = 0.125

Assuming 3 bit precision.

Conventional Rounding Error

Rounding	Error
1.10100 → 1.101	= 0
1.10101 → 1.101	= -0.25
1.10110 → 1.110	= +0.5
1.10111 → 1.110	= +0.25

Average Error = 0.5/4 = 0.125

Correction: round up only if there is one or more 1s at the right.

1.101100 → 1.101	= -0.5
1.101101 → 1.110	= +0.375

Rounding: round to even

- Round up conditions: 1.bbb…bbbGRXXXX
  - Round = 1, Sticky = 1 ➔ > 0.5
  - Guard = 1, Round = 1, Sticky = 0 ➔ Round to even

Value	Fraction	GRS	Incr?	Rounded
128	1.0000000	000	N	1.000
15	1.1010000	100	N	1.101
17	1.0001000	010	N	1.000
19	1.0011000	110	Y	1.010
138	1.0001010	011	Y	1.001
63	1.1111100	111	Y	10.000

Round up= R(G+S)

Guard bit: LSB of result
Round bit: 1st bit removed
Sticky bit: OR of remaining bits

Rounding: Round to even

- 1.bbb…bbbGRS
  - Round up = R(G+S)
  - GRS Rouding Error
  - 000 0 0
  - 001 0 -0.25
  - 010 0 -0.5
  - 011 +1 +0.25
  - 100 0 0
  - 101 0 -0.25
  - 110 +1 +0.5
  - 111 +1 +0.25

Rounding

- We need only one guard bit for normalization after addition.
- Assumption: Operands are normalized.

Example 2

$1.00001\ 2^3$

$-1.01011\ 2^{-1}$

Normalize according to exponent (assuming 5 bit precision)

$1.00001 \quad 2^3$

$-0.000101011\ 2^3$

$0.111100101\ 2^3$

Bit on the boundary

Round bit

Renormalize

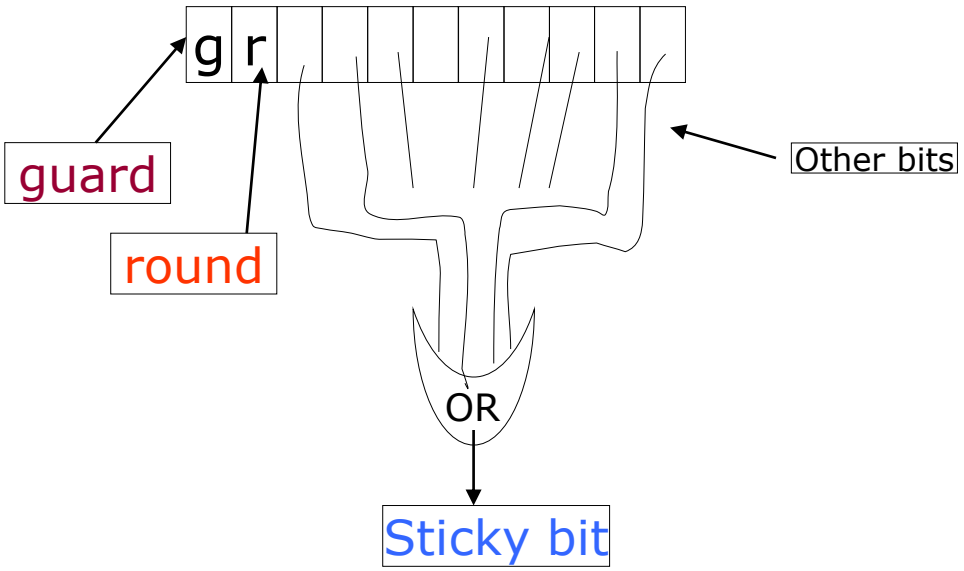
$1.11100101\ 2^2$

Take 5 bits after decimal

Result = $1.11101\ 2^2$

Non-zero => round-up

Theory behind it



- When shifting right, don't need to remember anything more than 3 bits below

Reference

- Michael L. Overton, Numerical Computing with IEEE Floating Point Arithmetic, SIAM, 2001
- Behrooz Parhami, Computer Arithmetic, second edition, Oxford, 2010
- Israel Koren, Computer Arithmetic Algorithms, second edition, A K Peters, Ltd., 2002