Computer Architecture

CH3 Computer Arithmetic (III)
Floating Point

Prof. Ren-Shuo Liu
NTHU EE
Fall 2017

Outline

- Overview
- IEEE 754 standard
  - Single-precision
  - Double-precision
  - Special numbers
- Floating-point operations
  - Addition
  - Multiplication
  - Rounding

Outline

- Overview
  - RISC-V floating-point instructions
  - Fixed-point and floating-point representations
- IEEE 754 standard
- Floating-point operations

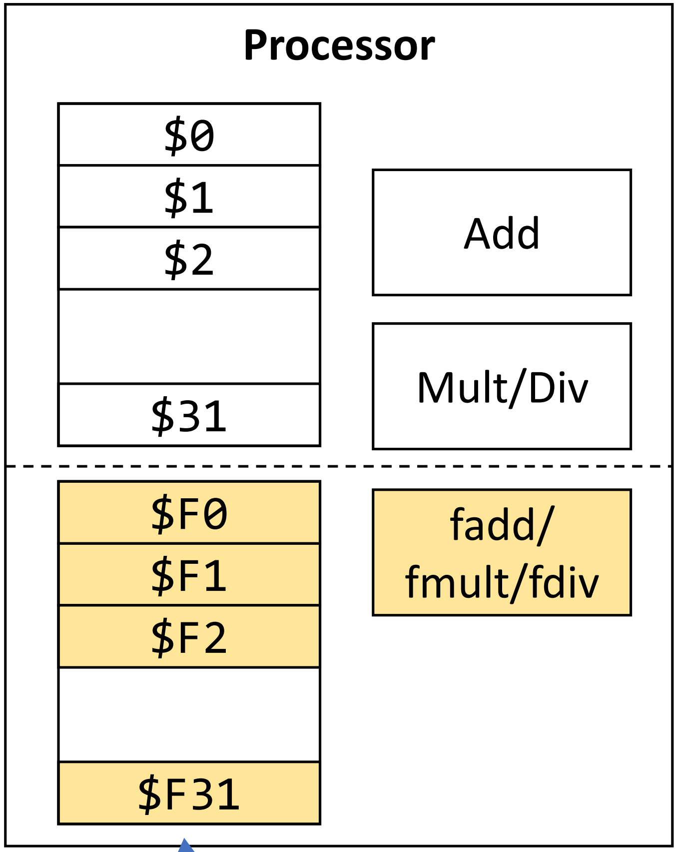
RISC-V Floating Instructions

- Arithmetic
  - fadd.s, fsub.s, fmul.s, fdiv.s # s means single-precision
  - fadd.d, fsub.d, fmul.d, fdiv.d # d means double-precision
- Comparisons
  - feq.s, feq.d # equal
  - flt.s, flt.d # less than
  - fle.s, fle.d # less than or equal
- Load / store
  - flw, fsw
  - fld, fsd

Floating Point Unit and Register Files

- Separate 32 registers for floating-point
  - Register pairs (e.g., $F0 and $F1) for double precision
  - $F0 is not always zero

- Floating-point instructions can be optional
  - Many embedded systems do not utilize them



整數和浮點數無法直接做運算，但在寫C code的時候，Compiler會幫你把整數轉成浮點數。

Fixed-Point

- Integers scaled by an implicit (隱含的) factor
- The scaling factor for each variable does not change (i.e., fixed) during the entire computation

計算時小數點位置不變，隱含著計算的每個數值都有相同的scaling factor

- Examples
  - 3.14 is represented as

把小數表示成一個整數跟一個scaling factor

    - 314 (scaling factor = 1/100)
    - 3140 (scaling factor = 1/1000)
  - 5,000,000 is represented as
    - 5 (scaling factor = 1,000,000)
    - 50 (scaling factor = 100,000)

Floating-Point ~= Scientific Notation (科學記號表示法)

	sign	significand		radix	exponent	unit
• 光速	+	2.99792458	×	10	8	(m/s)
• 電子電量	-	1.60217733	×	10	-19	(C)
• 0.5莫耳碳原子	+	6.00000000	×	10	-3	(kg)

sign

significand (有效數)
or fraction (小數),
or mantissa (尾數)

exponent (指數)

radix or base (底數)

- Normalized form
  - Exactly one non-zero significant digit to the left of the point
  - 29.9×10^7 and 0.299×10^9 are not normalized forms

Floating-Point ~= Scientific Notation (科學記號表示法)

- 光速 $+\ 2.99792458 \times 10^{8}$ (m/s)
- 電子電量 $-\ 1.60217733 \times 10^{-19}$ (C)
- 0.5莫耳碳原子 $+\ 6.00000000 \times 10^{-3}$ (kg)

sign

significand (有效數)
or fraction (小數),
or mantissa (尾數)

radix or base (底數)

exponent (指數)

- Scaling factor (the exponent) is explicit
- "Floating"
  - Scaling factor can change during computation

Floating Point Number

- IEEE 754 standard
  - Single-precision

    32 bits

S	Exp.	Significand

  - Double-precision

    64 bits

S	Exp.	Significand

只存小數部分，因為Normalized form規定只有一個數能在小數點左邊，且此為二進位(故直接寫1)

用Excess-K表示出小的數字

  - Represented value: $(-1)^S \times \boxed{1.}\mathrm{Significant}_{two} \times 2^{(\mathrm{Exponent}\ -\ \mathrm{Bias})}$

	Sign bit	Exp. bits	Significand bits	Bias
Single precision (*float* in C/C++)	1	0-255 8 unsigned	23	127
Double precision (*double* in C/C++)	1	11	52	1023

Single Precision: Exponent-Bias = -127->128

Special Floating Point Numbers

1. 32bit的浮點數可以比32bit的整數有更大的數值range(因為有Exponent)

2. 32bit浮點數和32bit整數一樣有2^32種組合。但是浮點數的0有重複(+-0)且正負無限大和NaN都不是數值。所以32bit整數能表示的數值個數比浮點數多

32 bits

S	Exp.	Significand

zero	S	0…00	0…00
Denormalized value	S	0…00	non-zero
+∞	0	1…11	0…00
-∞	1	1…11	0…00
Not a number (NaN)	d	1…11	non-zero

用於表示很小的數字(0.000000X)

i.e., the maximal and minimal exponent values are reserved for special floating-point numbers

Denormalized Value

- | S | 0…00 | Significand | are denormalized values
  - $(-1)^S \times 0.\text{Significant} \times 2^{(1\text{-bias})}$
  - No leading one to the left of the point

- Objective
  - Represent very small value
  - Gradual underflow

Floating Point Examples

S	Exp.	Significand
0	01111000	10100........000

  (32 bits)

$$= 1.1010......000_{two} \times 2^{(120-127)}$$
$$= 1.1010_{two} \times 2^{-7}$$
$$= 1.625_{ten} \times 2^{-7}$$
$$= 0.0126953125_{ten}$$

Floating Point Examples

- Convert -3.14 to 32-bit float
  - $3 = 11_{two}$
  - 3.14
    $= 11.0010\_0011\_1101\_0111\_0000\_1010..._{two}$
    $= 1.\underbrace{1001\_0001\_1110\_1011\_1000\_010}_{\text{23-bit significand (assume not rounded)}} \times 2^1$

=	1	10000000	

32 bits

要考慮Bias(128)

fraction part x2

小數部分乘以2，
取整數部分當作
輸出Binary

0.14

0.28	0.88
0.56	1.76
1.12	1.52
0.24	1.04
0.48	0.08
0.96	0.16
1.92	0.32
1.84	0.64
1.68	1.28
1.36	0.56
0.72	1.12
1.44	0.24

IEEE 754 Online Converter

IEEE 754 Converter

This page allows you to convert between the decimal representation of numbers (like "1.02") and the binary format used by all modern CPUs (IEEE 754 floating point). The conversion is limited to single precision numbers (32 Bit). The purpose of this webpage is to help you understand floating point numbers.

IEEE 754 Converter (JavaScript), V0.13

Note: This JavaScript-based version is still under development, please report errors here.

	Sign	Exponent	Mantissa
Value:	+1	2^{-7}	1.625
Encoded as:	0	120	5242880
Binary:	☐	☐☑☑☑☑☐☐☐	☑☐☑☐☐☐☐☐☐☐☐☐☐☐☐☐☐☐☐☐☐☐☐

Decimal Representation	0.0126953125
Binary Representation	00111100010100000000000000000000
Hexadecimal Representation	0x3c500000
After casting to double precision	0.0126953125

https://www.h-schmidt.net/FloatConverter/IEEE754.html

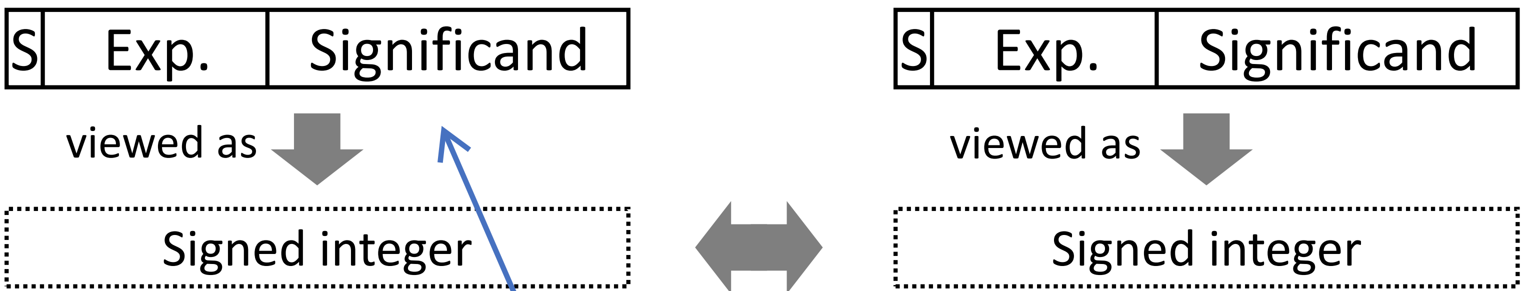
http://babbage.cs.qc.cuny.edu/IEEE-754.old/Decimal.html

Floating Point Operations

- Comparisons
- Addition
- Multiplication

Comparisons

- Almost identical to signed integer comparison



直接把他看成一個Signed Integer:
(1)先看正負號
(2)看Exponent
(3)最後看Significand

- Rationales
  - Positive > negative
  - Between two positive floating point numbers
    - One with larger {exponent, significand} is greater
  - Between two negative floating point numbers
    - One with smaller {exponent, significand} is greater

Comparisons (Cont'd)

- Cases directly supported by unsigned comparisons
  - $+\infty == +\infty$
  - $-\infty == -\infty$
  - $-\infty <$ all numbers $< \infty$
- Special cases that signed comparisons do not directly support
  - != involving any NaN yields true
  - All other comparisons involving NaN yield false
    - NaN < 10 ? → false
    - NaN > NaN? → false
    - NaN == NaN? → false
  - 0 == -0

Addition

- Steps
  1. Align (shift smaller number right)
  2. Perform addition
  3. Normalize
  4. Round
  5. Re-normalize

- Examples
  - $9.999_{ten} \times 10^{1} + 1.610_{ten} \times 10^{-1}$
  - $1.101_{two} \times 2^{9} + 1.110_{two} \times 2^{12}$
  - Assume the four-digit significands

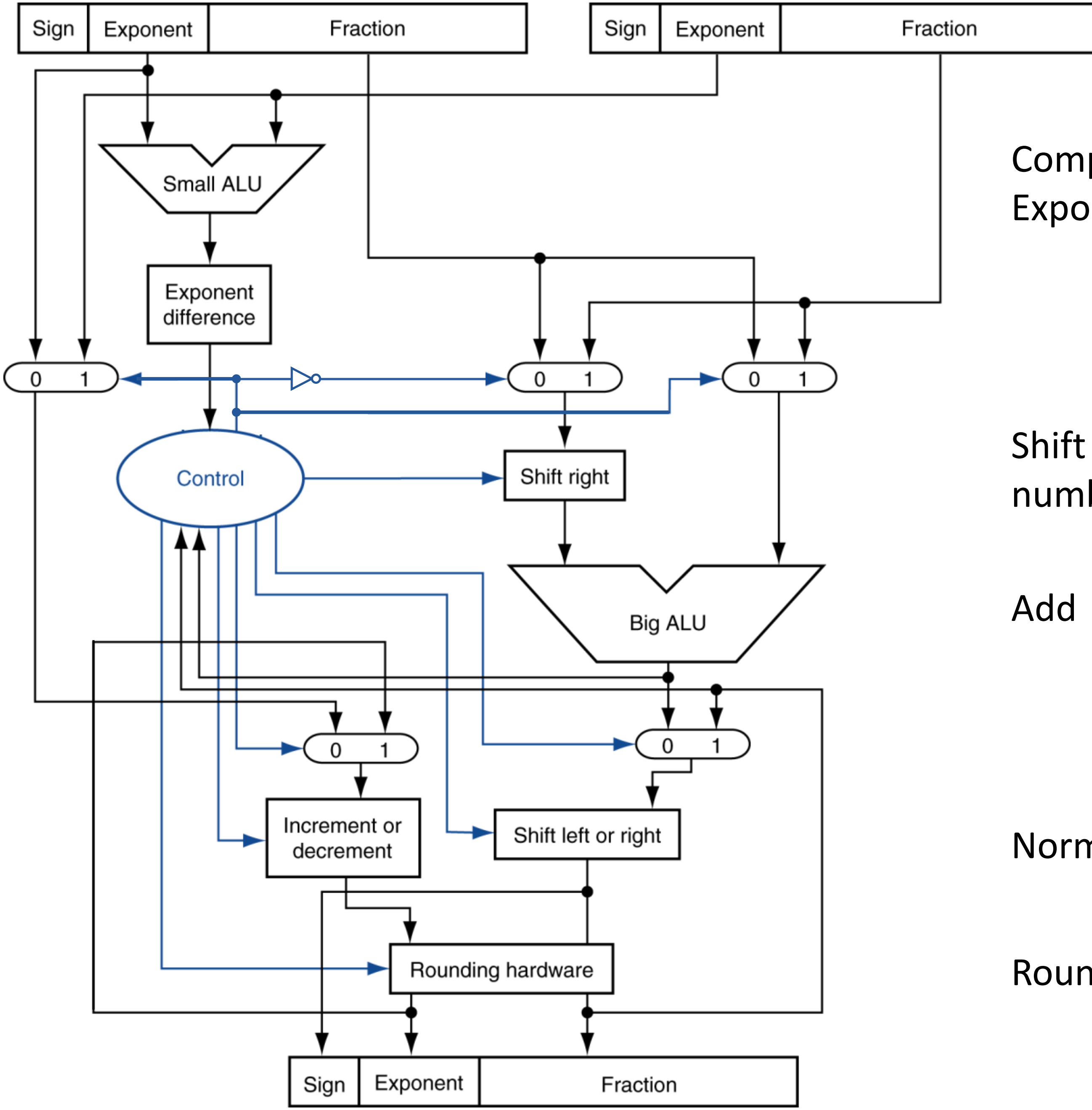
Decimal Example

	$9.999_{ten} \times 10^{1} \quad + 1.610_{ten} \times 10^{-1}$
Align 小數字往大數字對齊	$= 9.999_{ten} \times 10^{1} \quad + 0.01610_{ten} \times 10^{1}$
Add	$= 10.01500_{ten} \times 10^{1}$
Normalize	$= \underline{1.001}500_{ten} \times 10^{2}$
Round	$= 1.002_{ten} \times 10^{2}$
Renormalize	$= 1.002_{ten} \times 10^{2}$ (no change)

Renormalize至多做一次

Binary Example

	$1.101_{two} \times 2^{9}$	$+\ 1.110_{two} \times 2^{12}$
Align	$= 0.001101_{two} \times 2^{12}$	$+\ 1.110_{two} \times 2^{12}$
Add	$= 1.111101_{two} \times 2^{12}$	
Normalize	$= \underline{1.111}101_{two} \times 2^{12}$	(no change)
Round	$= 10.000_{two} \times 2^{12}$	101大於一半(四捨五入)
Renormalize	$= 1.000_{two} \times 2^{13}$	



Compare Exponents

Shift smaller number right

Add

Normalize

Round

Multiplication

- Steps
  1. Add exponents without bias
  2. Multiply the significands (with sign determined)
  3. Normalize (and check over/underflow)
  4. Round
  5. Re-normalize (and re-check over/underflow)

- Examples
  - $1.110_{ten} \times 10^{10} \times 9.200_{ten} \times 10^{-5}$
  - $1.000_{two} \times 2^{-1} \times (-1.110_{two}) \times 2^{-2}$
  - Inputs and outputs have four-digit significand

Decimal Example

$$1.110_{ten} \times 10^{10} \qquad \times 9.200_{ten} \times 10^{-5}$$

Exponent $\quad 10 + (-5) = 5$

Multiply $\quad 1.110_{ten} \times 9.200_{ten} = 10.212_{ten}$

Normalize $\quad = \underline{1.021}2_{ten} \times 10^{6}$

Round $\quad = 1.021_{ten} \times 10^{6}$

Renormalize $\quad = 1.021_{ten} \times 10^{6}$ (no change)

Binary Example

$$1.000_{two} \times 2^{-1} \qquad \times (-1.110_{two}) \times 2^{-2}$$

Exponent $(-1) + (-2) = (-3)$

Multiply $1.000_{two} \times (-1.110_{two}) = (-1.110000_{two})$

Normalize $= (-\underline{1.110}000_{two}) \times 2^{-3}$ (no change)

Round $= (1.110_{two}) \times 2^{-3}$ (no change)

Renormalize $= (1.110_{two}) \times 2^{-3}$ (no change)

Internal Format with Extra Bits

- Extra bits are needed during arithmetic operations to increase the arithmetic accuracy
- e.g., $1.101_{two} \times 2^9 + 1.110_{two} \times 2^{12}$

without extra bits

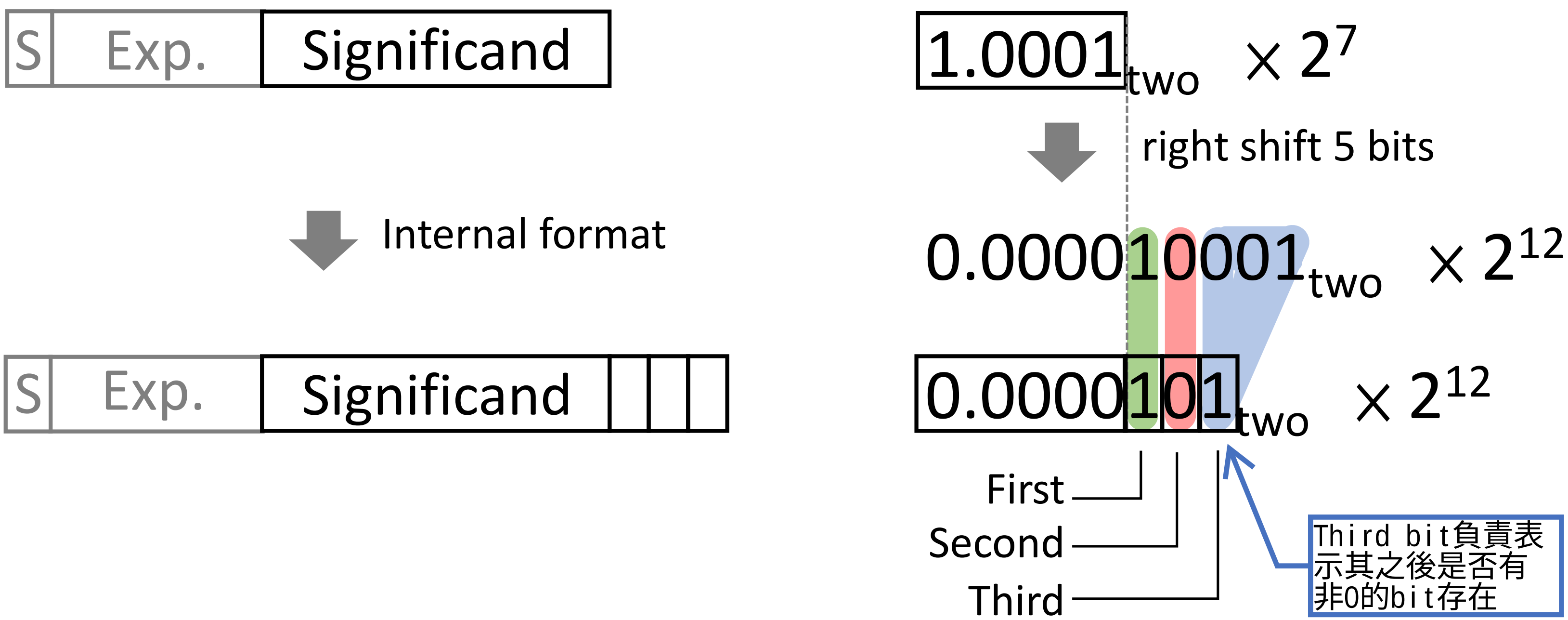
$$
\begin{array}{rl}
 & 0.001_{two} \times 2^{12} \\
+ & 1.110_{two} \times 2^{12} \\
\hline
= & \underline{1.111}_{two} \times 2^{12}
\end{array}
$$

with extra bits

$$
\begin{array}{rl}
 & 0.001\overbrace{101}_{two} \times 2^{12} \\
+ & 1.110000_{two} \times 2^{12} \\
\hline
= & \underline{1.111}101_{two} \times 2^{12} \\
= & 10.00_{two} \times 2^{12} \\
= & 1.000_{two} \times 2^{13}
\end{array}
$$

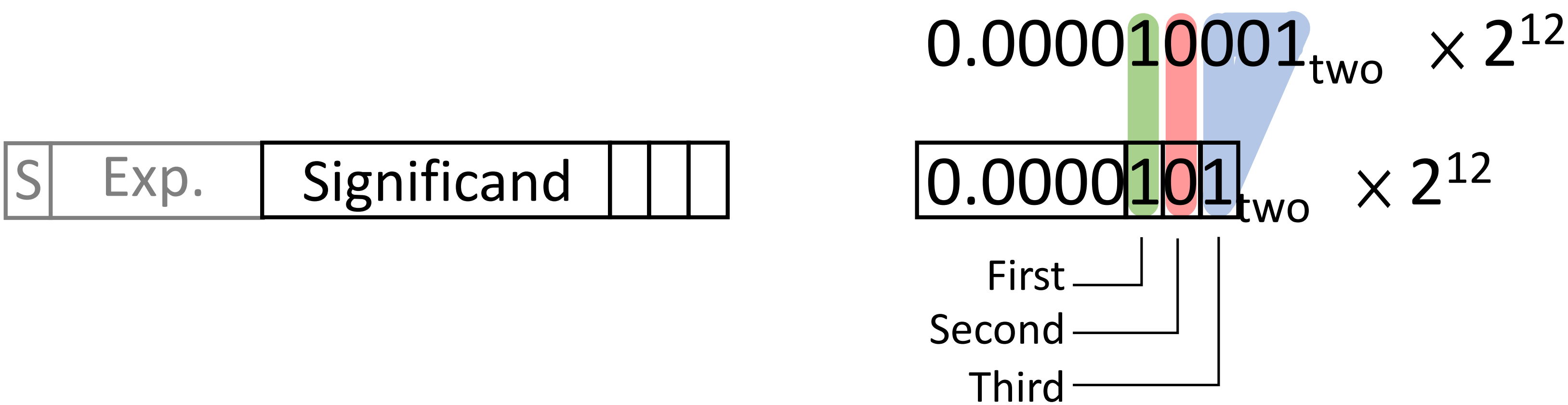
IEEE 754 Internal Format

- Three extra bits
  - The 3rd one represents any remaining nonzero bits to the right



IEEE 754 Internal Format

- Roles/names of the three extra bits
  - First: Guard
  - Second: Round
  - Third: Sticky



IEEE 754 Rounding Mode

- Four modes can be chosen by programmers
  - Toward 0 (also called truncation) 無條件捨去
  - Toward $+\infty$ 正數:無條件進位、負數:無條件捨去 (往正方向)
  - Toward $-\infty$ 負數:無條件進位、正數:無條件捨去 (往負方向)
  - Toward nearest even (default mode)
    - Choose the even one if there are two equally nearest values

Round Toward Nearest Even

- Binary examples

剛好一半的Case，看前面的數經過
運算後是否為偶數來決定

四捨六入

捨去後為偶數

進位後為偶數

	10.00~~011~~	10.00101	10.10~~100~~	10.11100
Results	10.00	10.01	10.10	11.00

- Reduce the statistical biases of rounding noises

Outline

- Overview
- IEEE 754 standard
  - Single-precision
  - Double-precision
  - Special numbers
- Floating-point operations
  - Addition
  - Multiplication
  - Rounding