CSARCH Lecture Series: Floating-Point Operation and Guard Bits

Sensei RL Uy

College of Computer Studies

De La Salle University

Manila, Philippines

‘2110

Copyright Notice

Overview

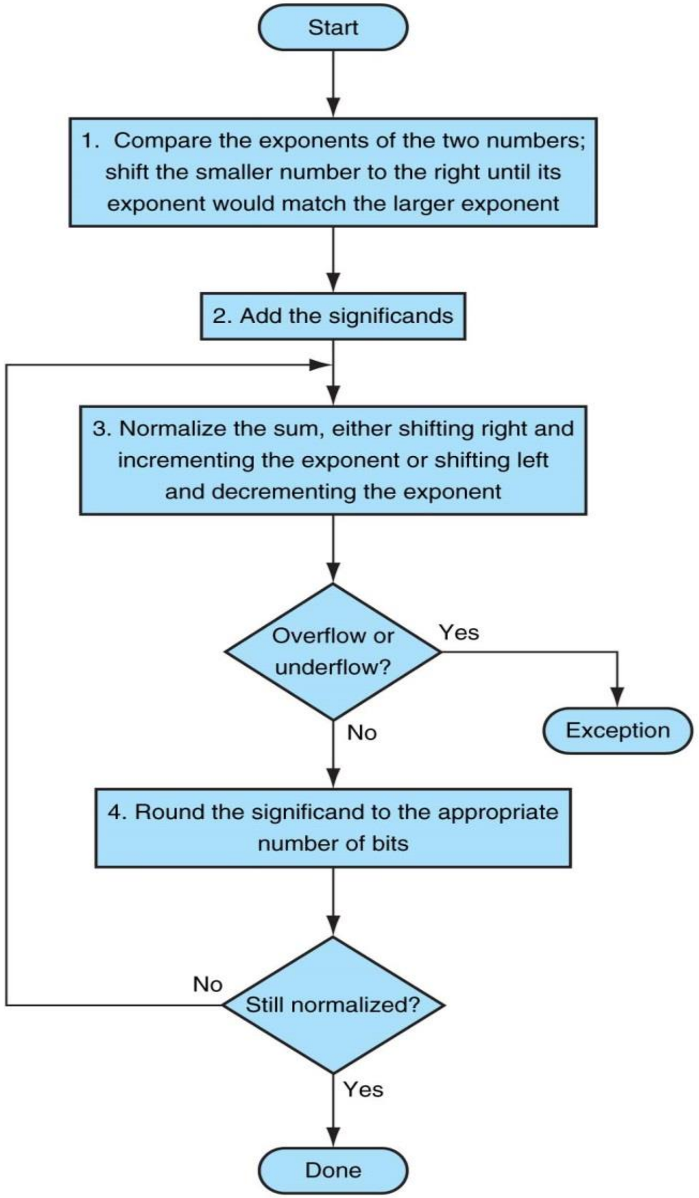
Reflect on the following question:

- Consider a system that can support 7 bits, perform FP addition using guard, round and sticky bits
  - $1.0111110010_2 \times 2^5 + 1.0011111110_2 \times 2^3$

Overview

- This sub-module introduces floating-point arithmetic operations and the concepts of guard, round and sticky bits
- The objectives are as follows:
  - ✓Describe the process of performing floating-point arithmetic operations
  - ✓Describe the process of using guard, round and sticky bits

FP Addition Algorithm

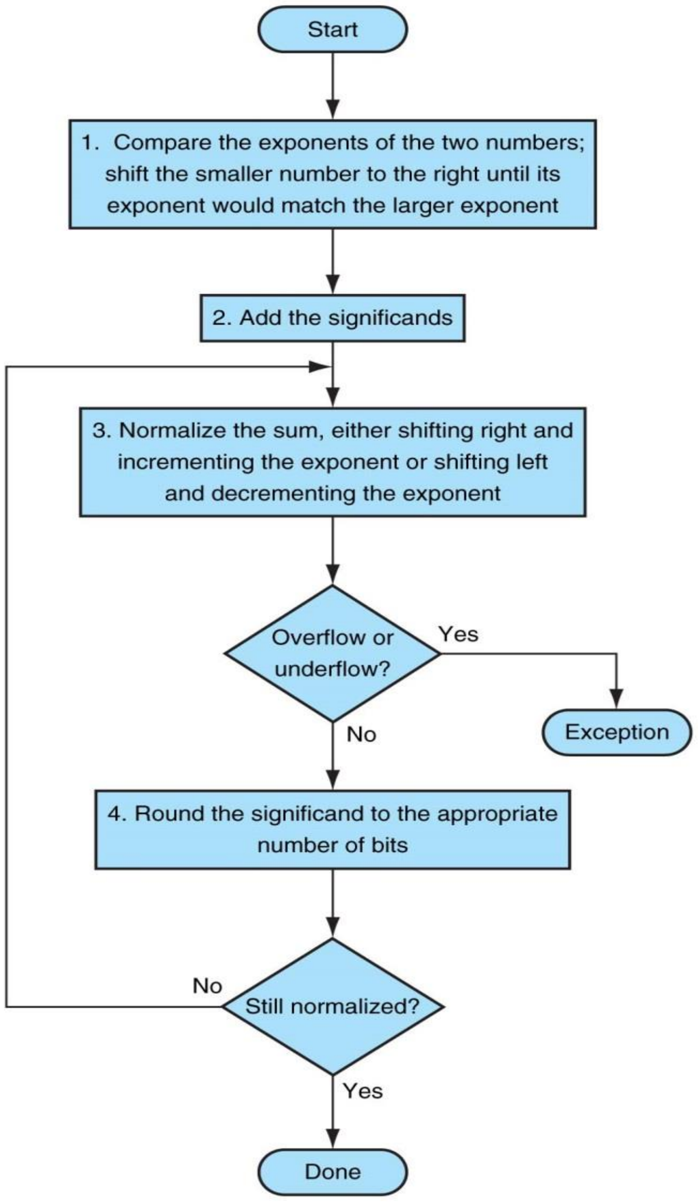


Floating-point addition. The normal path is to execute steps 3 and 4 once, but if rounding causes the sum to be unnormalized, we must repeat step 3.

Floating-Point Addition

- Consider a system that can support 4 decimal digits
  - $9.999 \times 10^{1} + 1.610 \times 10^{-1}$

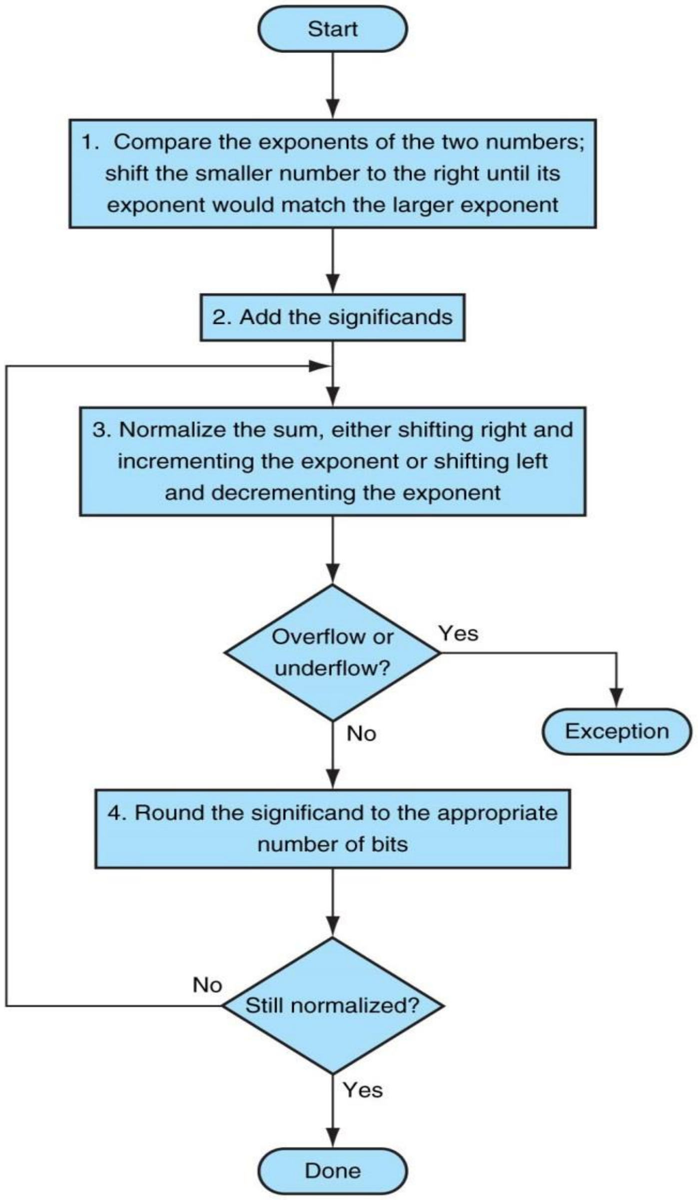
1. Align decimal points
   - $9.999 \times 10^{1} + 0.016 \times 10^{1}$
2. Add the significands/significand
   - $9.999 \times 10^{1} + 0.016 \times 10^{1} = 10.015 \times 10^{1}$
3. Normalize result & check for over/underflow
   - 1.0015×10^{2}
4. Round and renormalize if necessary
   - 1.002×10^{2}



Floating-Point Addition

- Consider a system that can support 4 bits:
  - $1.000_2 \times 2^{-1} + (-1.110_2) \times 2^{-2}$

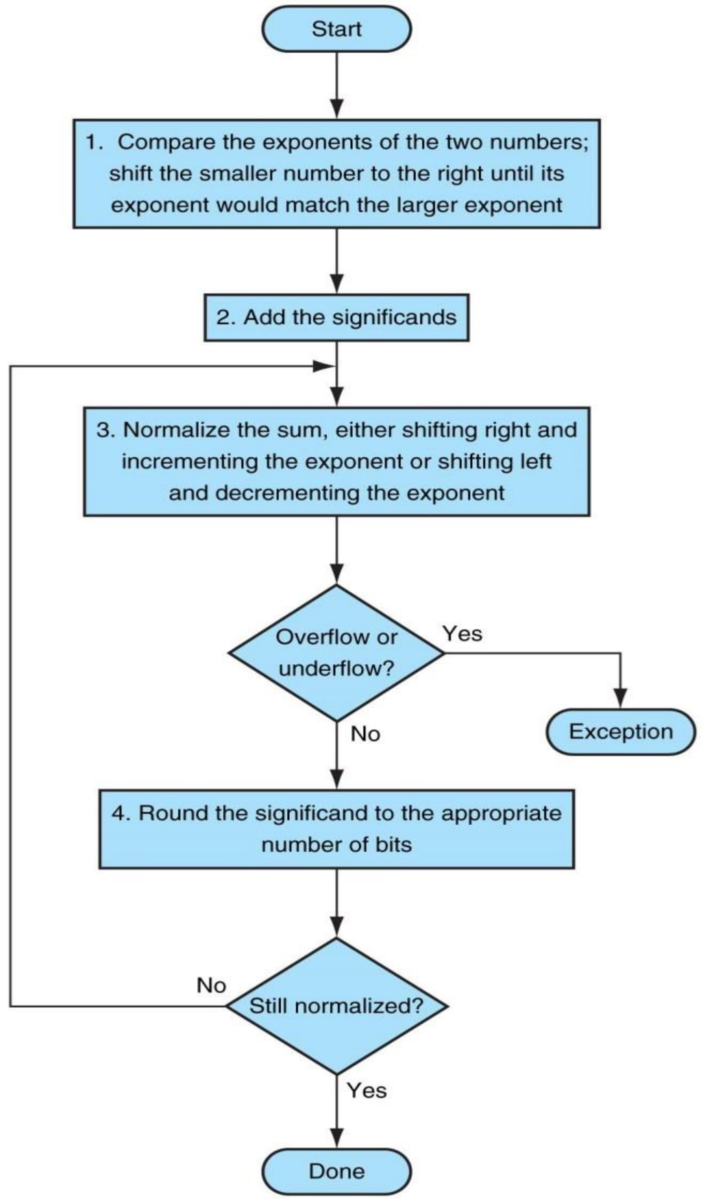
1. Align binary points
   - $1.000_2 \times 2^{-1} + (-0.111_2) \times 2^{-1}$
2. Add significands
   - $1.000_2 \times 2^{-1} + (-0.111_2) \times 2^{-1} = 0.001_2 \times 2^{-1}$
3. Normalize result & check for over/underflow
   - $1.000_2 \times 2^{-4}$
4. Round and renormalize if necessary
   - $1.000_2 \times 2^{-4}$ (no change)



Floating-Point Subtraction

- Consider a system that can support 7 decimal digits
  - 123457.1467 - 123456.659

1. Align decimal points
   - $1.234571467 \times 10^5 - 1.23456659 \times 10^5$
2. Subtract significands
   - $1.234571 \times 10^5 - 1.234567 \times 10^5 = 0.000004 \times 10^5$
3. Normalize result & check for over/underflow
   - 4.000000×10^{-1}
4. Round and renormalize if necessary
   - 4.000000×10^{-1} (same)



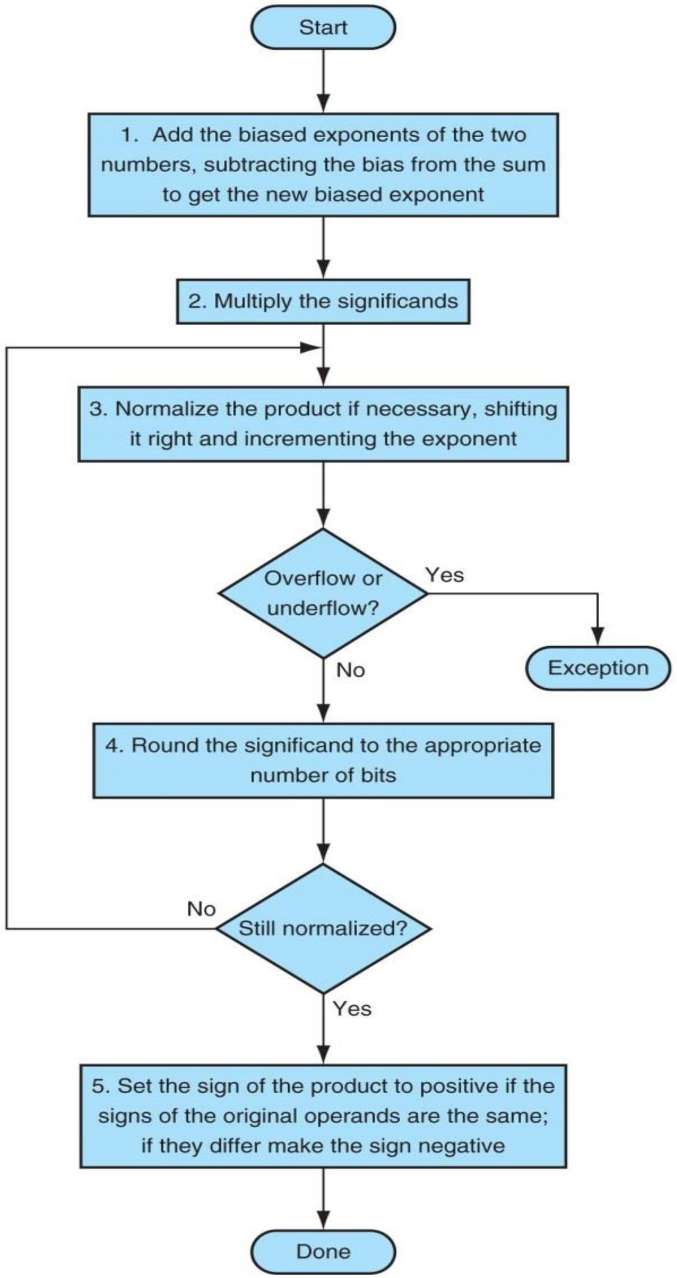
Floating-Point Addition

- Reflection: Floating-point is not associative

A= 1234.567, b=45.67834, c=0.0004

Is (A+B)+C = A+(B+C)?

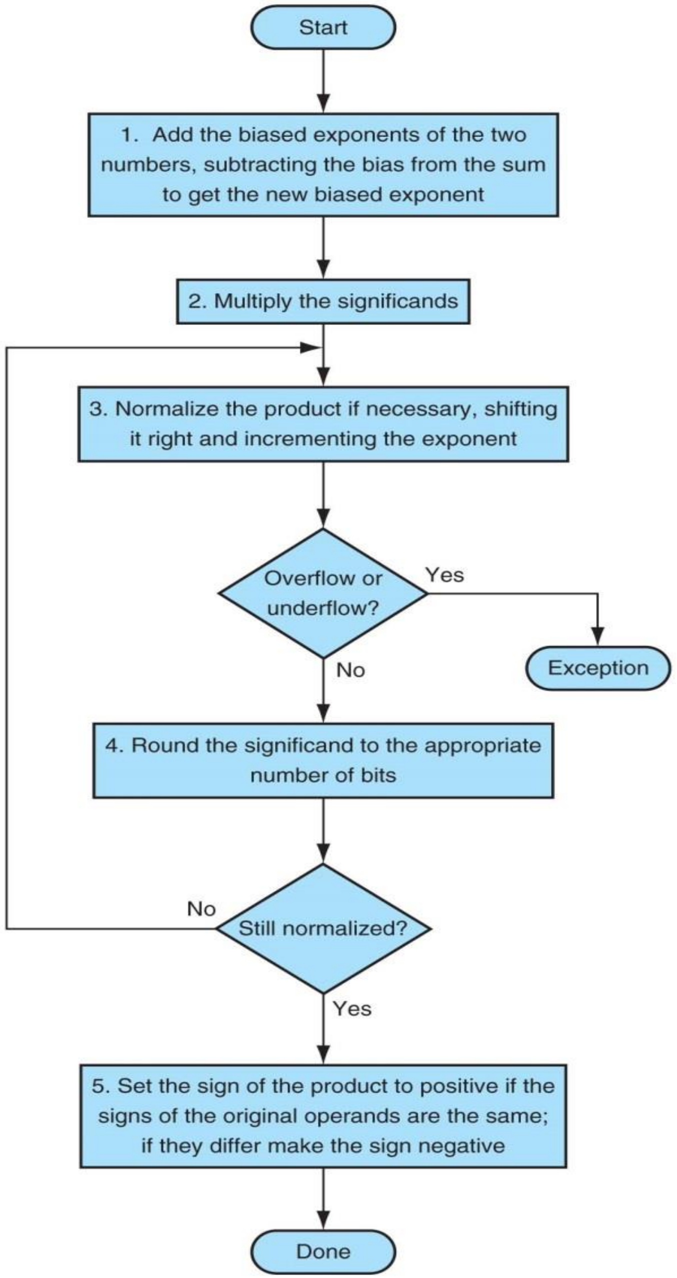
FP Multiplication Algorithm



Floating-Point Multiplication

- Consider a system that can support 4 decimal digits
  - $1.110 \times 10^{10} \times 9.200 \times 10^{-5}$

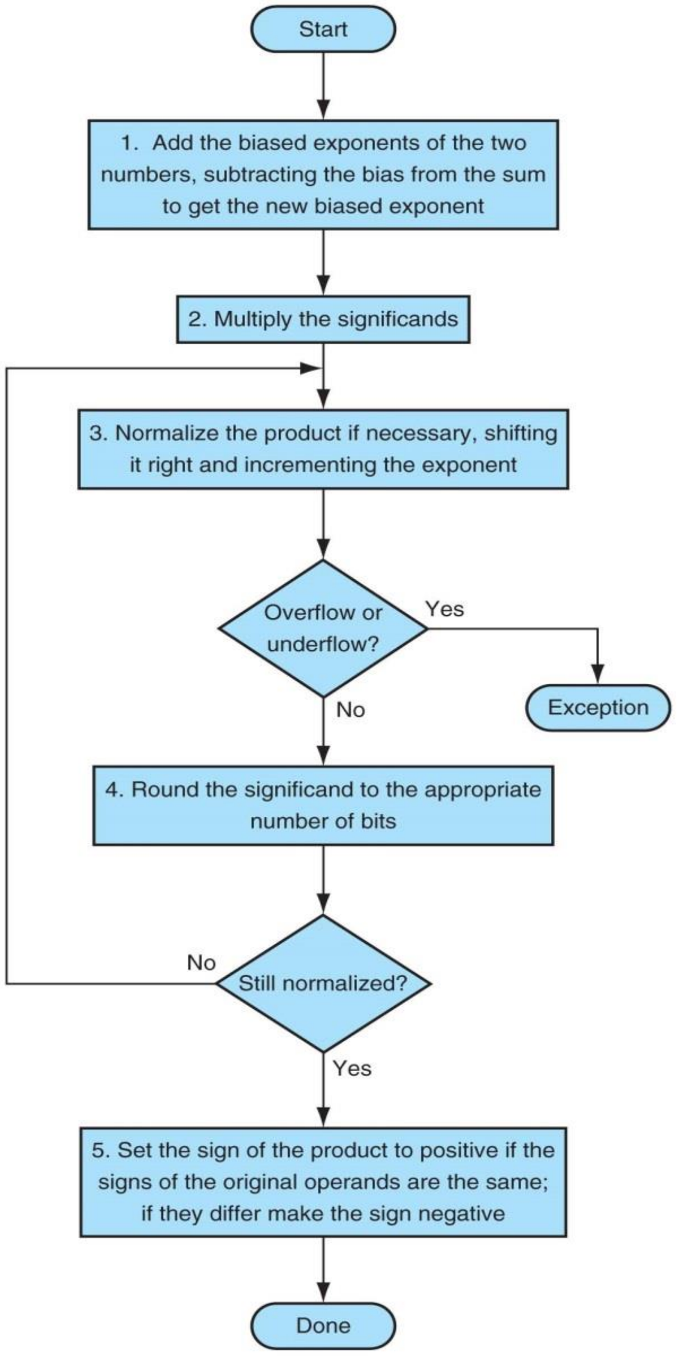
1. Add exponents
   - New exponent = $10 + -5 = 5$
2. Multiply significands
   - $1.110 \times 9.200 = 10.212 \Rightarrow 10.212 \times 10^{5}$
3. Normalize result & check for over/underflow
   - 1.0212×10^{6}
4. Round and renormalize if necessary
   - 1.021×10^{6}
5. Determine sign of result from signs of operands
   - $+1.021 \times 10^{6}$



Floating-Point Multiplication

- Consider a system that can support 4 bits
  - $1.000_2 \times 2^{-1} \times -1.110_2 \times 2^{-2}$

1. Add exponents
   - Unbiased: $-1 + -2 = -3$
2. Multiply significands
   - $1.000_2 \times 1.110_2 = 1.110_2 \;\Rightarrow\; 1.110_2 \times 2^{-3}$
3. Normalize result & check for over/underflow
   - $1.110_2 \times 2^{-3}$ (no change)
4. Round and renormalize if necessary
   - $1.110_2 \times 2^{-3}$ (no change)
5. Determine sign: +operand × –operand ⇒ –operand
   - $-1.110_2 \times 2^{-3}$



Guard, Round & Sticky Bits

- *Guard bit* the first of the two extra bits kept on the right during intermediate calculations of floating point numbers; used to improve rounding accuracy.
- *Round bit* the second of the two extra bits kept on the right during intermediate calculations of floating point numbers ; used to improve rounding accuracy.
- *Sticky bit* – bit used in rounding in addition to guard and round bit that is set whenever there are nonzero **bits to the right** of the round bit.

Guard, Round & Sticky Digits

- For decimal, the sticky digit is always 1 if there is a non-zero in the least significant digit

Rounding with guard, round & sticky bits

- Consider a binary that can support 5 bits
  - $1.00111101 \times 2^5 + 1.00111101 \times 2^3$

1. Align binary points
   - Shift number with smaller exponent
   - $1.00111101 \times 2^5 + 0.0100111101 \times 2^5$
2. Add significands (with guard, round and sticky bits)
   - $1.0011111 \times 2^5 + 0.0100111 \times 2^5 = 1.1000110 \times 2^5$
3. Normalize result & check for over/underflow
   - 1.1001×2^5
4. Round and renormalize if necessary
   - 1.1001×2^5

Rounding without guard, round & sticky bits

- Consider a binary that can support 5 bits
  - $1.00111101 \times 2^5 + 1.00111101 \times 2^3$

1. Align binary points
   - Shift number with smaller exponent
   - $1.00111101 \times 2^5 + 0.0100111101 \times 2^5$
2. Add significands
   - $1.0100 \times 2^5 + 0.0101 \times 2^5 = 1.1001 \times 2^5$
3. Normalize result & check for over/underflow
   - 1.1001×2^5
4. Round and renormalize if necessary
   - 1.1001×2^5

Rounding with guard, round & sticky digits

- Consider a system that can support 7 decimal digits
  - 123457.1467 - 123456.659

1. Align decimal points
   - Shift number with smaller exponent
   - $1.234571467 \times 10^5 - 1.23456659 \times 10^5$
2. Add significands (with guard, round and sticky digits)
   - $1.234571461 \times 10^5 - 1.234566590 \times 10^5 = 0.000004871 \times 10^5$
3. Normalize result & check for over/underflow
   - 4.871000×10^{-1}
4. Round and renormalize if necessary
   - 4.871000×10^{-1} vs. [compare to 4.000000×10^{-1} without guard & round digits]

Try

Consider a system that can support 7 bits

- $1.0111110010_2 \text{ x}2^5 + 1.0011111110_2 \text{ x}2^3$

		Base^{Exp}
Operand 1		
Operand 2		
Sum		
Sum (normalize)		

Perform without guard, round and sticky bits

Try

Consider a system that can support 7 bits

- $1.0111110010_2 \times 2^5 + 1.0011111110_2 \times 2^3$

		Base^{Exp}
Operand 1	1.011111	2^5
Operand 2	0.010100	2^5
Sum	1.110011	2^5
Sum (normalize)	1.110011	2^5

Perform without guard, round and sticky bits

Try

Consider a system that can support 7 bits

- $1.0111110010_2 \times 2^5 + 1.0011111110_2 \times 2^3$

		G	R	S	Base^{Exp}
Operand 1					
Operand 2					
Sum					
Sum (normalize)					

Perform with guard(G), round(R) and sticky(S) bits

Try

Consider a system that can support 7 bits

- $1.0111110010_2 \times 2^5 + 1.0011111110_2 \times 2^3$

		G	R	S	Base^{Exp}
Operand 1	1.011111	0	0	1	2^5
Operand 2	0.010011	1	1	1	2^5
Sum	1.110011	0	0	0	2^5
Sum (normalize)	1.110011	-	-	-	2^5

Perform with guard(G), round(R) and sticky(S) bits

To recall …

- What have we learned:
- The objective are as follows:
  - ✓Describe the process of performing FP arithmetic operations
  - ✓Describe the process of using guard, round and sticky bits