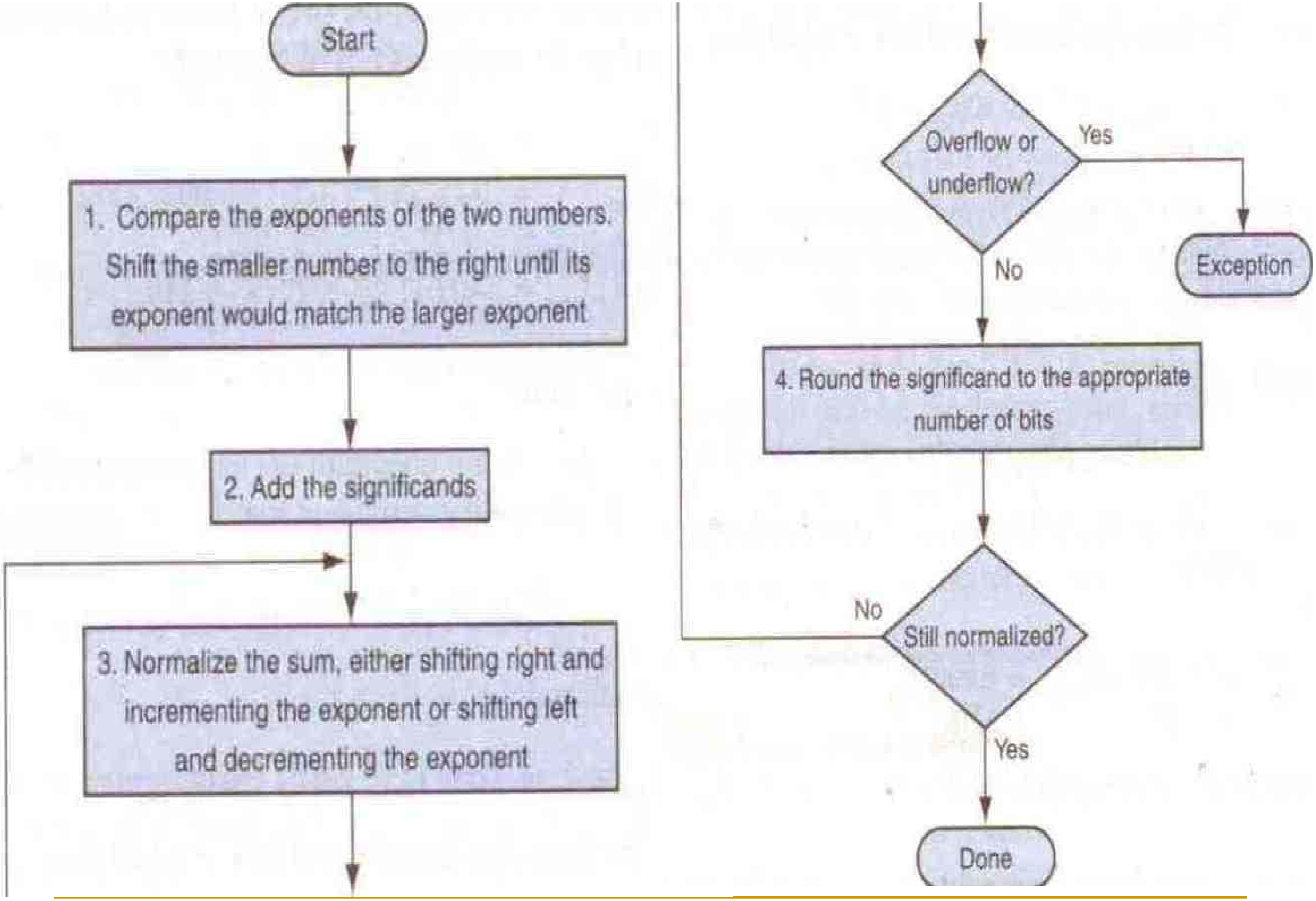
COMPUTER ORGANIZATION (IS F242)

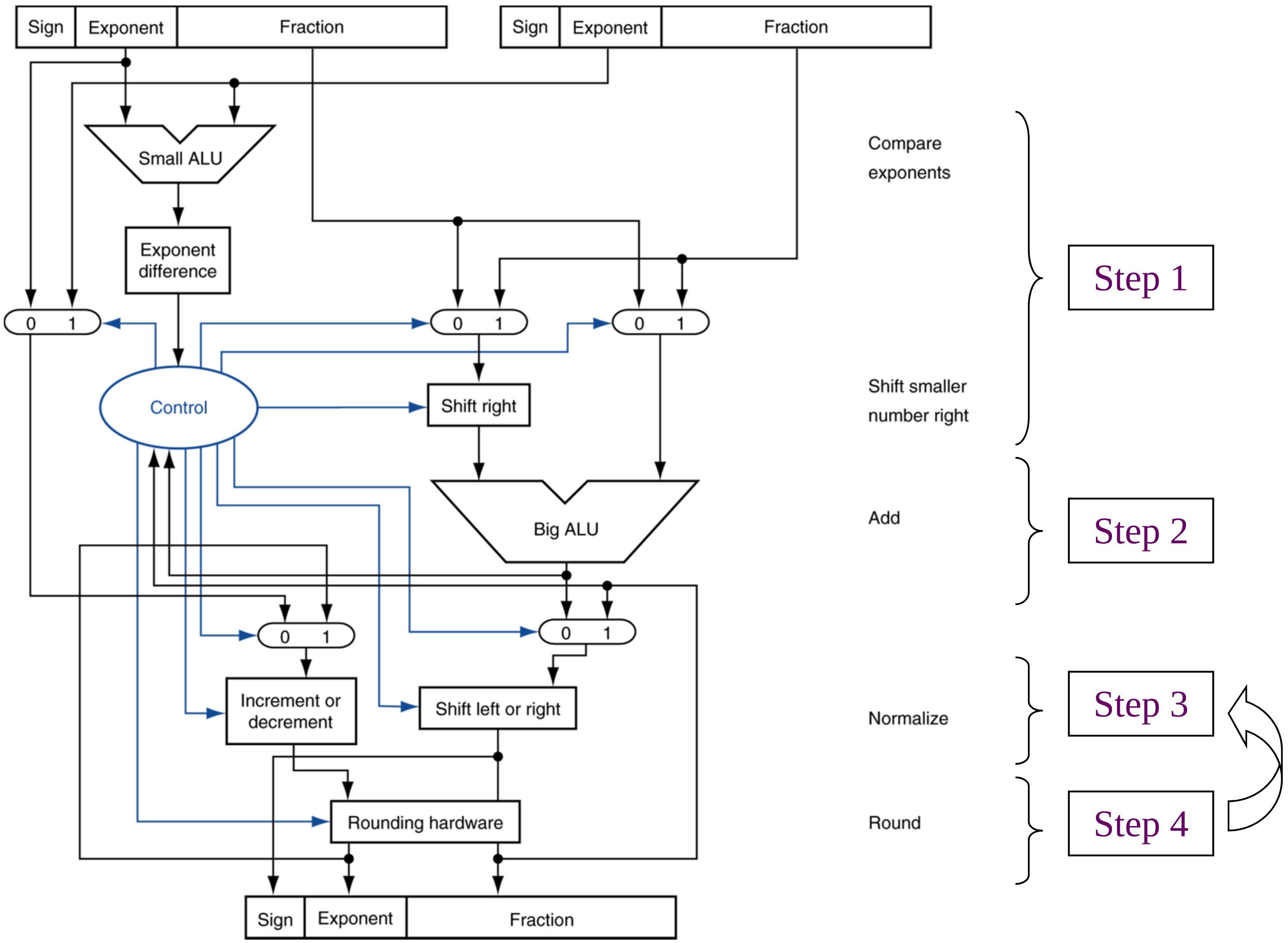
LECT 13: FLOATING POINT

Arithmetic Operations

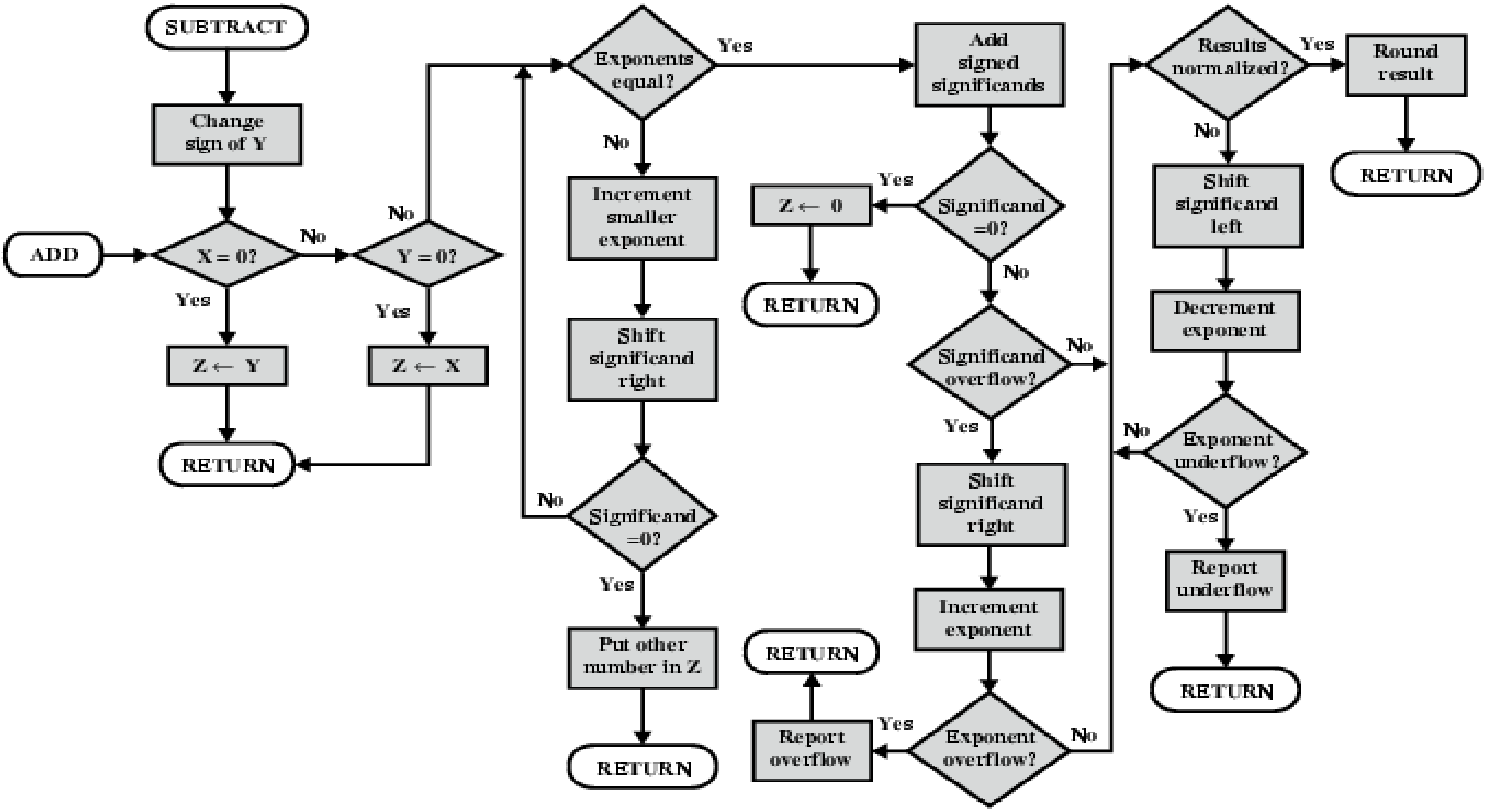
- Addition & Subtraction
  - Check for Zeros
  - Align the Mantissas
  - Add or Subtract the Mantissas
  - Normalize the result
  - Example
    - $X = 0.3 * 10^2 \quad Y = 0.2 * 10^3$
    - $X = (0.1 * 2^0)_2 \; Y = (-0.0111 * 2^0)_2$
    - $12.5 \times 10^1 + 346 \times 10^{-3}$


Start
1. Compare the exponents of the two numbers. Shift the smaller number to the right until its exponent would match the larger exponent
2. Add the significands
3. Normalize the sum, either shifting right and incrementing the exponent or shifting left and decrementing the exponent
Overflow or underflow?
Yes
Exception
No
4. Round the significand to the appropriate number of bits
No
Still normalized?
Yes
Done

FP Adder Hardware



FP Addition & Subtraction Flowchart



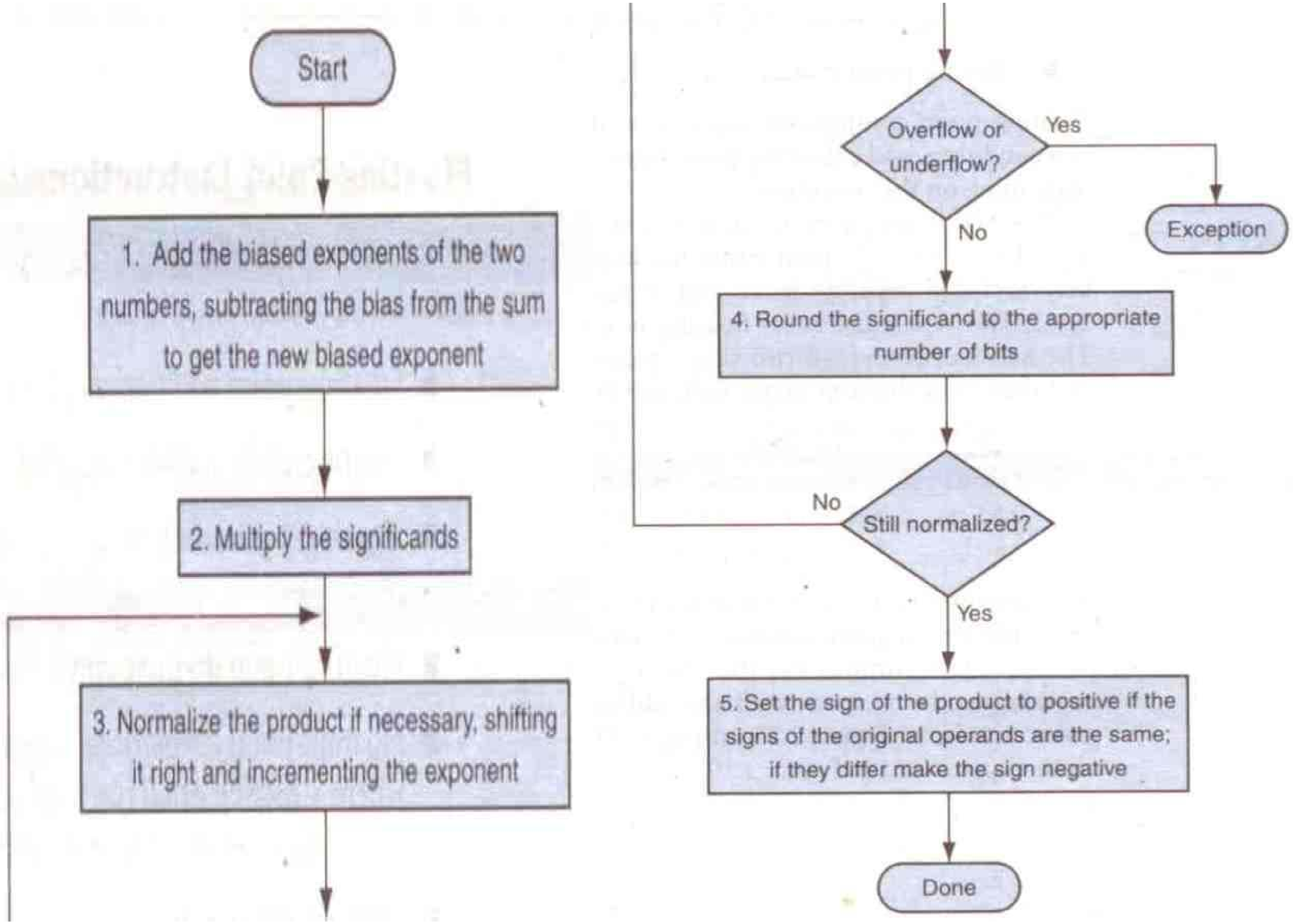
Arithmetic Operations

- Multiplication & Division
  - Check for Zero
  - Add or Subtract the exponents
  - Multiply or Divide the Mantissas (significands) (watch sign)
  - Normalize the result
  - Round
  - All intermediate results should be in double length storage
  - Example
    - X = 0.3 * 10^2
    - Y = 0.2 * 10^3

Biased Exponent

- If $X = 1.110 * 2^{10}$

  $Y = 0.001 * 2^{-5}$
- Bias = 127
- New Exponent?
- Adjust the exponent
- What about Division?



Start

1. Add the biased exponents of the two numbers, subtracting the bias from the sum to get the new biased exponent

2. Multiply the significands

3. Normalize the product if necessary, shifting it right and incrementing the exponent

Overflow or underflow?

Yes

Exception

No

4. Round the significand to the appropriate number of bits

No

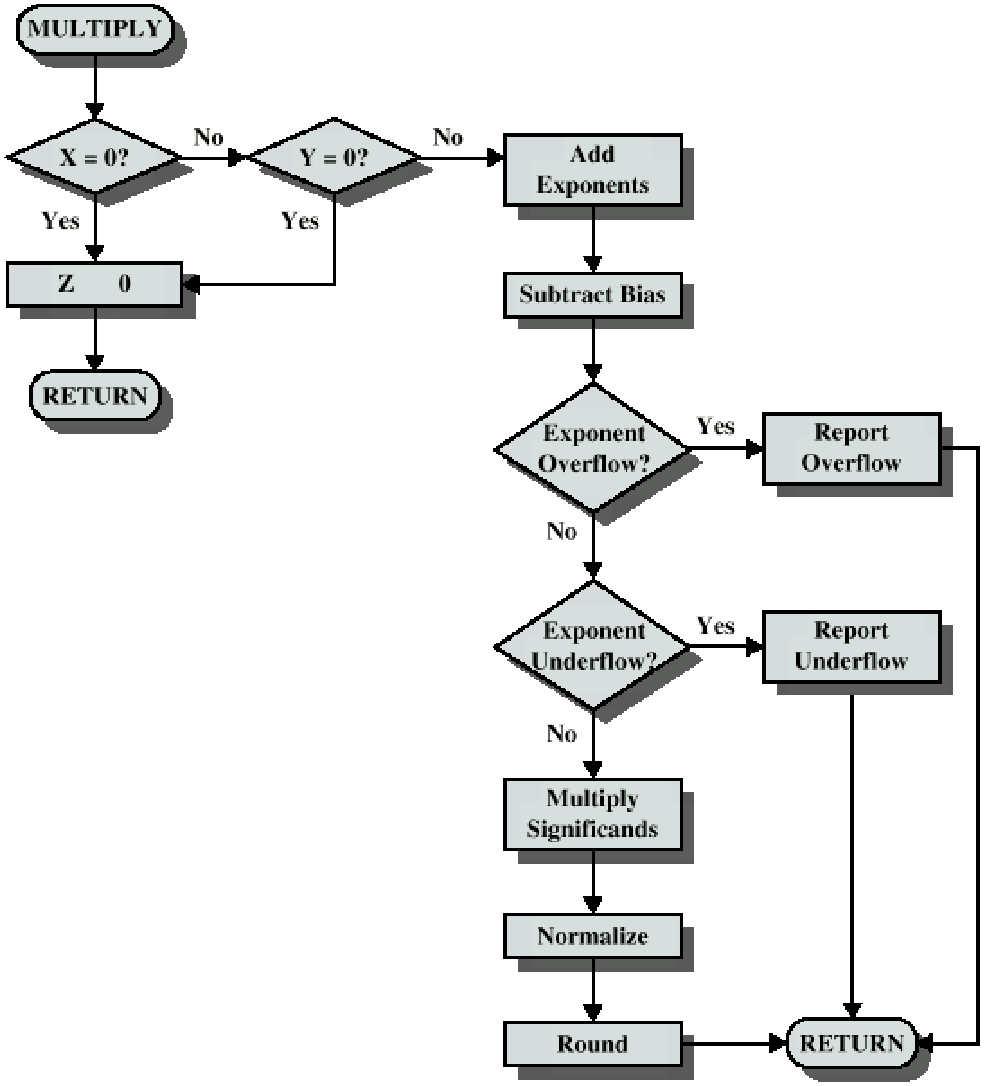
Still normalized?

Yes

5. Set the sign of the product to positive if the signs of the original operands are the same; if they differ make the sign negative

Done

Floating Point Multiplication



MULTIPLY

X = 0?

No

Y = 0?

No

Add Exponents

Yes

Yes

Z 0

Subtract Bias

RETURN

Exponent Overflow?

Yes

Report Overflow

No

Exponent Underflow?

Yes

Report Underflow

No

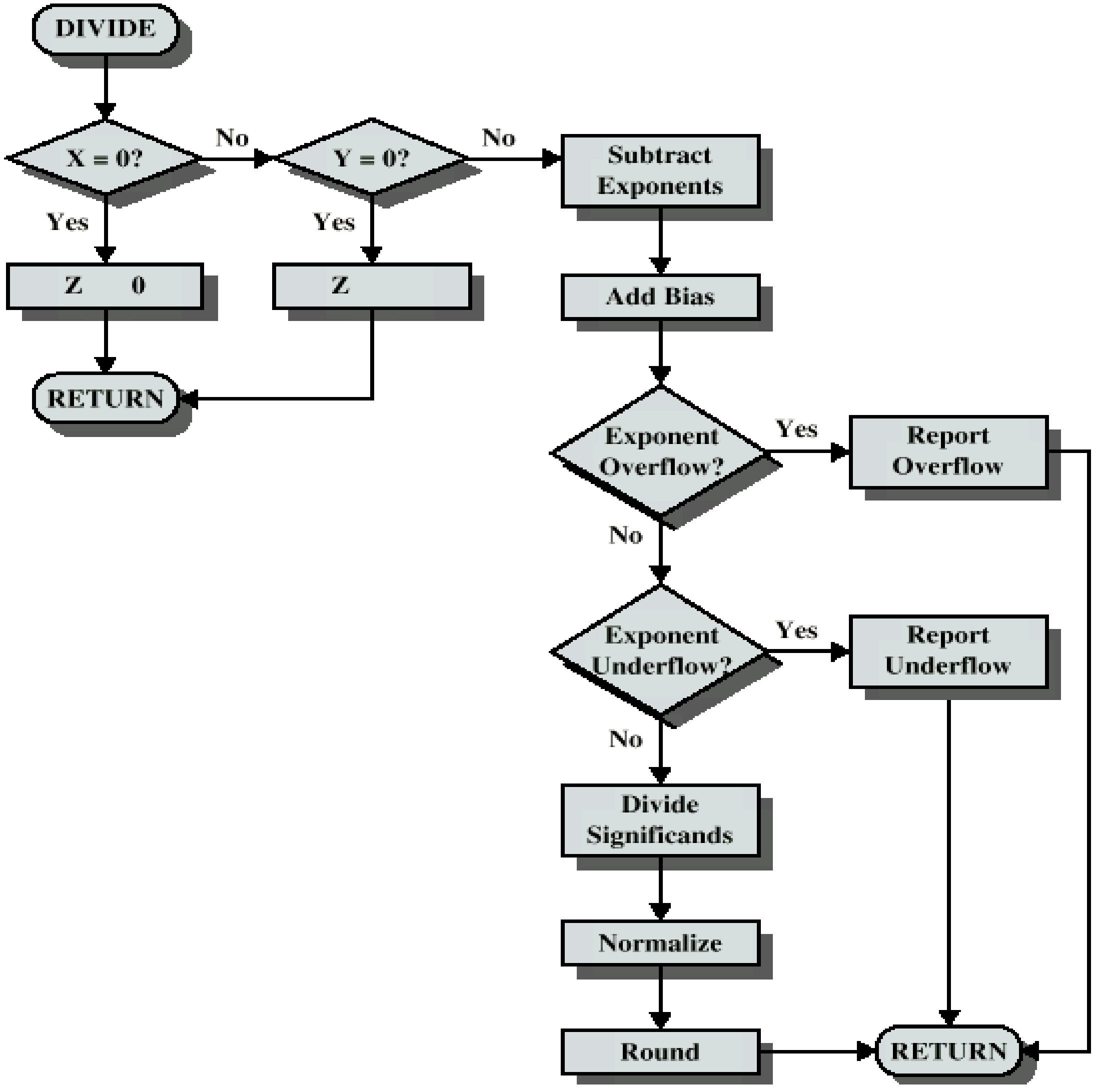
Multiply Significands

Normalize

Round

RETURN

Floating Point Division



Guard Bits

- Guard Bits
  - Additional bits used to pad out the right end of the significand
- Example:
  - Consider:
    - X = 1.00000000000000000000000 * 2^{1}
    - Y = 1.11111111111111111111111 * 2^{0}
  - Equalize Exponents and Subtract
    - X = 1.00000000000000000000000 * 2^{1}
    - Y = 0.11111111111111111111111 * 2^{1}
    - Z = 0.00000000000000000000001 * 2^{1}
  - Normalize
    - Z = 1.00000000000000000000000 * 2^{-23}

Guard Bits

- Now add few bits extra to Mantissa
  - X = 1.00000000000000000000000**000** * 2^1
  - Y = 1.11111111111111111111111**000** * 2^0
- Equalize Exponents and Subtract
  - X = 1.00000000000000000000000**000** * 2^1
  - Y = 0.11111111111111111111111**100** * 2^1
  - Z = 0.00000000000000000000000**100** * 2^1
- Normalize
  - **Z = 1.00000000000000000000000 * 2^{-24}**

Precision Considerations[2]

- Rounding
  - Disposal of extra bits
  - How???
    - If extra bit amount to more than one half of the last representable bit position then rounding up to the next representable number.
    - If extra bit amount to less than one half of the last representable bit position then rounding down to the next representable number.

Rounding IEEE Standards

- Rounding to nearest
  - Result is rounded to the nearest representable number
  - If rounding bits are exactly in the midway
    - Round to the nearest EVEN (IEEE standard)
- Rounding towards Zero
  - Result is rounded towards zero (Simple truncation)
- Rounding towards +Infinity
  - Result is rounded up towards plus infinity
- Rounding towards –Infinity
  - Result is rounded down towards negative infinity

LC-3 Single Cycle Design

Instruction

- The instruction is the fundamental unit of work.
- Control Unit interprets instruction
- Specifies two things:
  - *opcode*: Operation to be performed (what the instruction does)
    - e.g. ADD, SUB, LOAD, STORE
  - *operands*: Data/locations to be used for operation
    - e.g ADD *dest scr1 scr2*
- An instruction is encoded as a sequence of bits.
- Often, but not always, instructions have a fixed length, such as 16 or 32 bits.
- ***Note:***
- A computer's instructions and their formats is known as its ***Instruction Set Architecture*** **(*ISA*)**.