COMPUTER ARITHMETIC

Floating Point Representation

Computer Science Department

National University of Computer and Emerging Sciences

Islamabad

REAL NUMBERS

- Numbers with fractions
- Could be done in pure binary
  - 1001.1010 = $2^4 + 2^0 + 2^{-1} + 2^{-3}$ =9.625
- Where is the binary point?
- Fixed?
  - Very limited
- Moving?
  - How do you show where it is?

FLOATING POINT NOTATION

- Decimal
  - 12.4568_{ten} (decimal notation) means
    - 10*1 + 2 + 4/10 + 5/100 + 6/1000 + 8/10000
  - In scientific notation
  - 12.4568 =
    - $124568 * 10^{-4} = 1245680 * 10^{-5} =$
    - $12456.8 * 10^{-3} = 1245.68 * 10^{-2} =$
    - $124.568 * 10^{-1} = 12.4568 * 10^{0}$
    - $1.24568 * 10^{1}$
  - $1.24568*10^{1}$ is an example of ***normalised*** scientific notation.

FLOATING POINT IN BINARY

- Binary
  - 0.010011_{two} =

    $(0/2) + (1/2^2) + (0/2^4) + (1/2^5) + (1/2^6)$
    - $0 + 1/4 + 0 + 1/32 + 1/64 =$
    - $(0.25 + 0.03125 + 0.015625)_{ten} =$
    - 0.296875_{ten}
- In scientific notation
  - $10011*2^{-6} = 1001.1*2^{-5} =$

  = $100.11*2^{-4}$

  = $1.0011*2^{-2}$ *normalised*

NORMALIZATION

Every binary number, **except the one corresponding the number zero**, can be normalized by choosing the exponent so that the radix point falls to the right of the leftmost 1 bit.

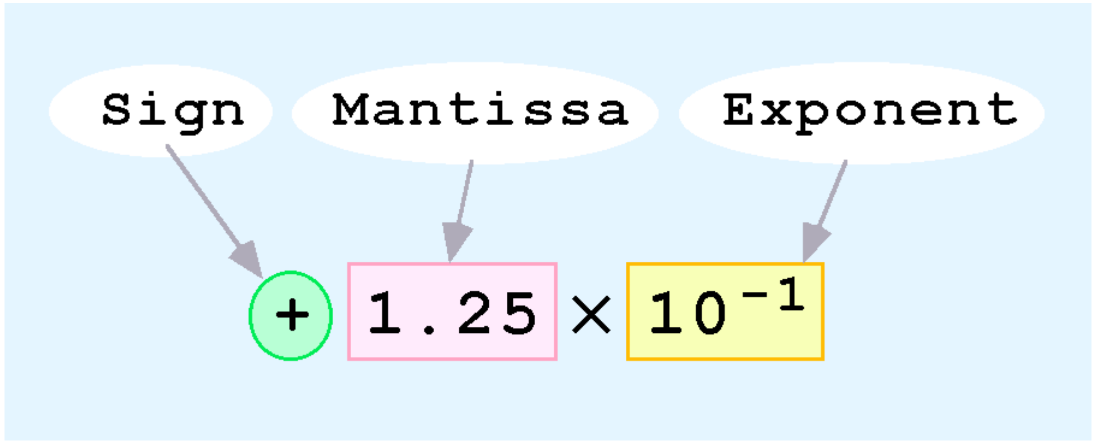
$37.25_{10} = 100101.01_2 = 1.0010101 \times \mathbf{2^5}$

$7.625_{10} = 111.101_2 = 1.11101 \times \mathbf{2^2}$

$0.3125_{10} = 0.0101_2 = 1.01 \times \mathbf{2^{-2}}$

FLOATING-POINT REPRESENTATION

- Computers use a form of scientific notation for floati ng-point representation
- Numbers written in scientific notation have three co mponents:



FLOATING POINT

Sign bit	Biased Exponent	Significand or Mantissa

- +/- .significand x 2^{exponent}
- Point is actually fixed between sign bit and body of mantissa
- Exponent indicates place value (point position)

FLOATING-POINT STANDARDS

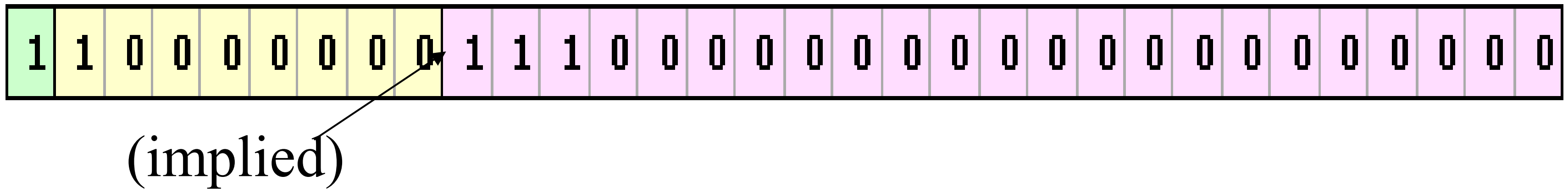
- The IEEE has established a standard for floating-point numbers
- The IEEE-754 *single precision* floating point standard uses an 8-bit exponent (with a bias of 127) and a 23-bit significand. Bias is 2^{k-1}-1
- The IEEE-754 *double precision* standard uses an 11-bit exponent (with a bias of 1023) and a 52-bit significand.

FLOATING-POINT REPRESENTATION

- In both the IEEE single-precision and double-precision floating-point standard, the significand has an implied 1 to the LEFT of the radix point.
  - The format for a significand using the IEEE format is: 1.xxx…
  - For example, $4.5 = .1001 \times 2^3$ in IEEE format is $4.5 = 1.001 \times 2^2$.
  - The 1 is implied, which means it does not need to be listed in the significand (the significand would include only 001).

FLOATING-POINT REPRESENTATION

- Example: Express -3.75 as a floating point number using IEEE single precision.
- First, let's normalize according to IEEE rules:
  - $-3.75 = -11.11_2 = -1.111 \times 2^1$
  - The bias is 127, so we add 127 + **1** = 128 (this is our exponent)
  - The first 1 in the significand is implied, so we have:



  - Since we have an implied 1 in the significand, this equates to **$-(1).111_2 \times 2^{(128-127)} = -1.111_2 \times 2^1 = -11.11_2 = -3.75$.**

DECIMAL FLOATING POINT TO IEEE STANDARD CONVERSION

Ex 1: Find the IEEE FP representation of 40.15625

Step 1.

Compute the binary equivalent of the whole part and the fractional part. (i.e. convert 40 and .15625 to their binary equivalents)

DECIMAL FLOATING POINT TO IEEE STANDARD CONVERSION

40	.15625
Result:	**Result:**
101000	.00101

So: $40.15625_{10} = 101000.00101_{2}$

DECIMAL FLOATING POINT TO IEEE STANDARD CONVERSION

Step 2. Normalize the number by moving the decimal point to the right of the leftmost one.

$$101000.00101 = 1.0100000101 \times \mathbf{2}^{5}$$

DECIMAL FLOATING POINT TO IEEE STANDARD CONVERSION

Step 3. Convert the exponent to a biased exponent

$127 + 5 = 132$

And convert biased exponent to 8-bit unsigned binary:

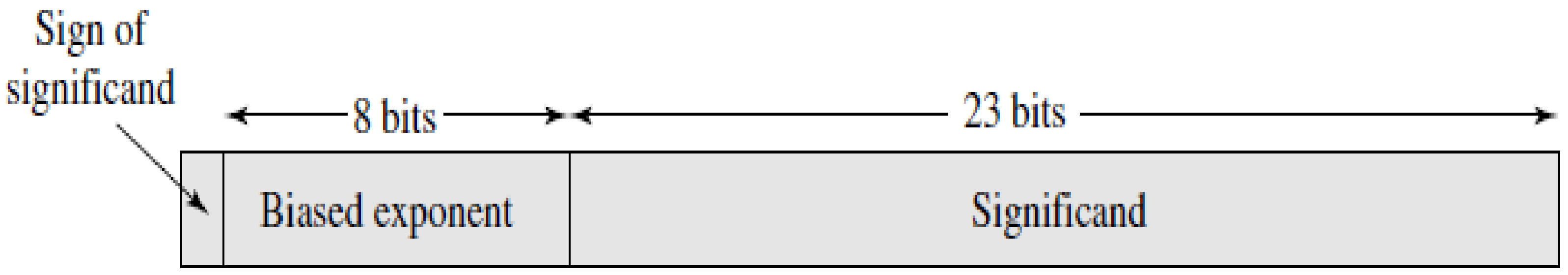
$132_{10} = 10000100_2$

DECIMAL FLOATING POINT TO IEEE STANDARD CONVERSION

Step 4. Store the results from steps 1-3:

Sign	Exponent	Mantissa
	(from step 3)	(from step 2)
0	**10000100**	**01000001010000000000000**

FLOATING POINT EXAMPLES



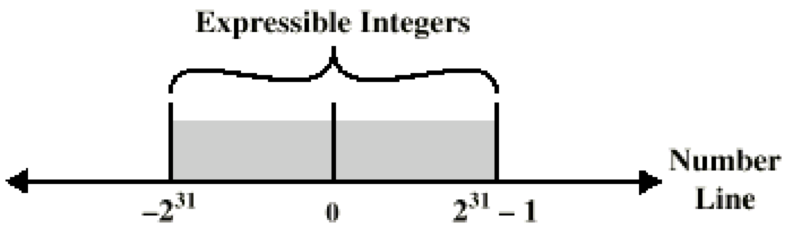
(a) Format

$$
\begin{array}{rcl}
1.1010001 \times 2^{10100} & = 0\ 10010011\ 10100010000000000000000 = & 1.6328125 \times 2^{20} \\
-1.1010001 \times 2^{10100} & = 1\ 10010011\ 10100010000000000000000 = & -1.6328125 \times 2^{20} \\
1.1010001 \times 2^{-10100} & = 0\ 01101011\ 10100010000000000000000 = & 1.6328125 \times 2^{-20} \\
-1.1010001 \times 2^{-10100} & = 1\ 01101011\ 10100010000000000000000 = & -1.6328125 \times 2^{-20}
\end{array}
$$

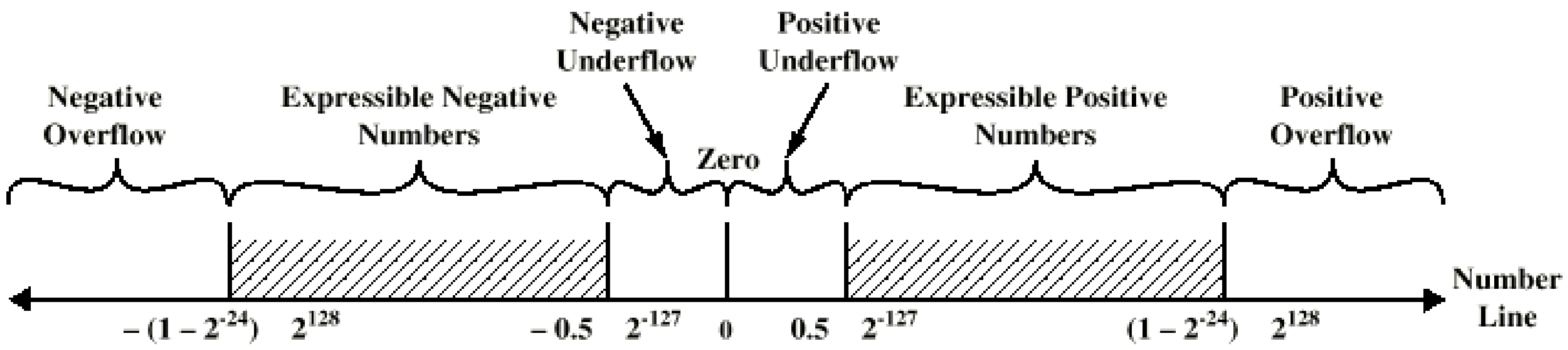
FP RANGES

- For a 32 bit number
  - 8 bit exponent (-127 to 128)
  - 24 bit fraction ($-(1-2^{-24})$ to $(1-2^{-24})$)

EXPRESSIBLE NUMBERS

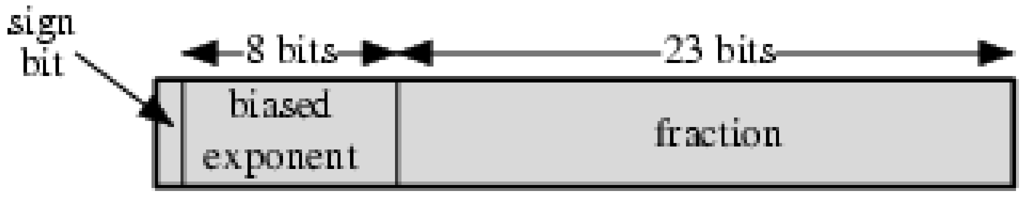


(a) Twos Complement Integers

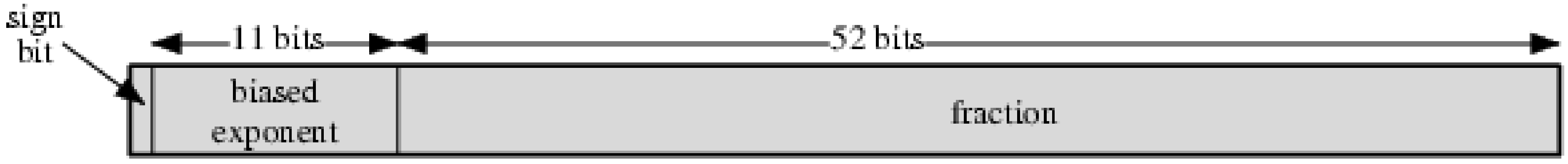


(b) Floating-Point Numbers

IEEE 754 FORMATS



(a) Single format



(b) Double format

IEEE 754

- Standard for floating point storage
- 32 and 64 bit standards
- 8 and 11 bit exponent respectively
- Extended formats (both mantissa and exponent) for intermediate results

Table 9.4 Interpretation of IEEE 754 Floating-Point Numbers

	Single Precision (32 bits)				Double Precision (64 bits)			
	Sign	Biased exponent	Fraction	Value	Sign	Biased exponent	Fraction	Value
positive zero	0	0	0	0	0	0	0	0
negative zero	1	0	0	−0	1	0	0	−0
plus infinity	0	255 (all 1s)	0	∞	0	2047 (all 1s)	0	∞
minus infinity	1	255 (all 1s)	0	$-\infty$	1	2047 (all 1s)	0	$-\infty$
quiet NaN	0 or 1	255 (all 1s)	$\neq 0$	NaN	0 or 1	2047 (all 1s)	$\neq 0$	NaN
signaling NaN	0 or 1	255 (all 1s)	$\neq 0$	NaN	0 or 1	2047 (all 1s)	$\neq 0$	NaN
positive normalized nonzero	0	$0 < e < 255$	f	$2^{e-127}(1.f)$	0	$0 < e < 2047$	f	$2^{e-1023}(1.f)$
negative normalized nonzero	1	$0 < e < 255$	f	$-2^{e-127}(1.f)$	1	$0 < e < 2047$	f	$-2^{e-1023}(1.f)$
positive denormalized	0	0	$f \neq 0$	$2^{e-126}(0.f)$	0	0	$f \neq 0$	$2^{e-1022}(0.f)$
negative denormalized	1	0	$f \neq 0$	$-2^{e-126}(0.f)$	1	0	$f \neq 0$	$-2^{e-1022}(0.f)$

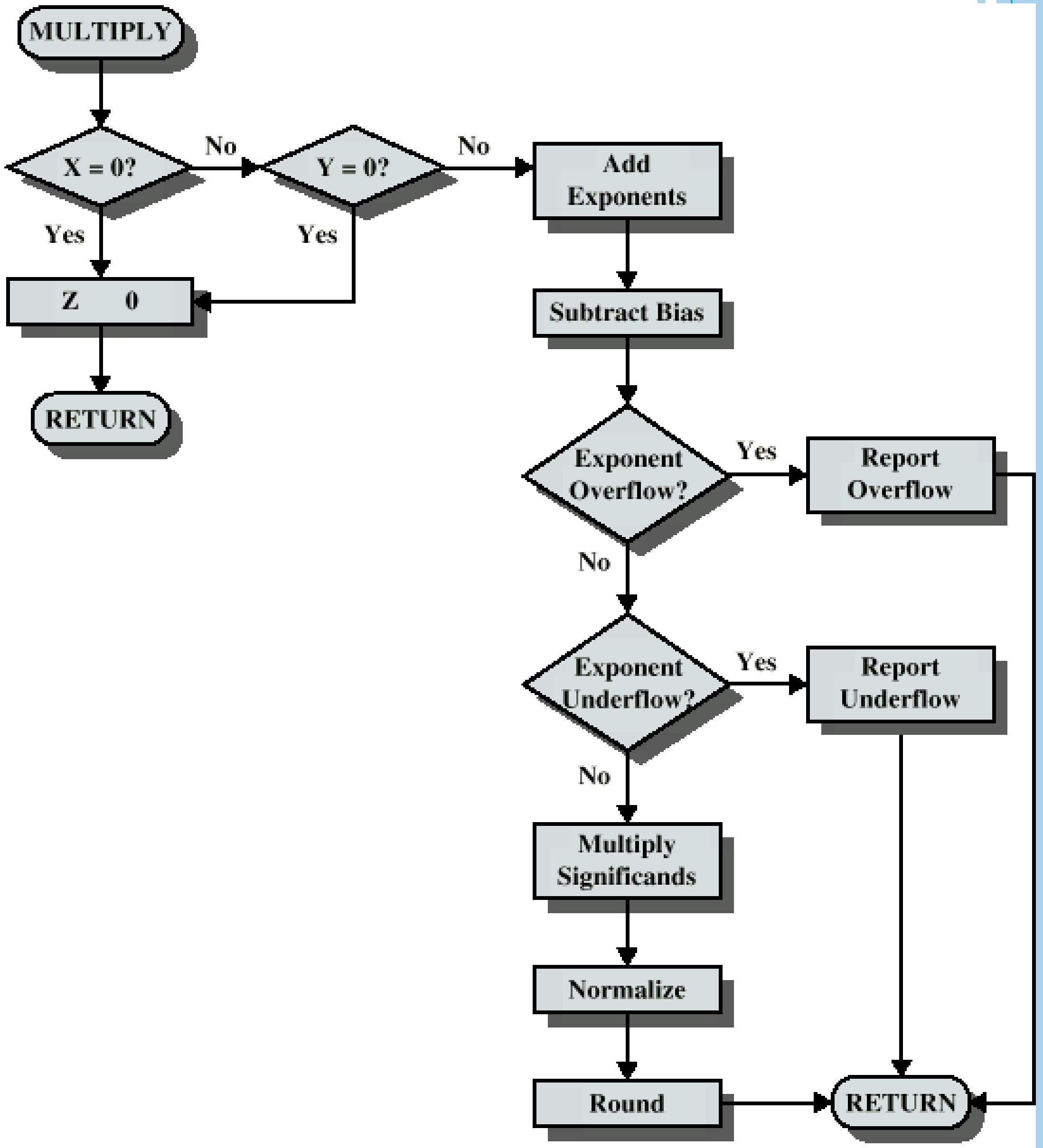
FP ARITHMETIC +/-

- Check for zeros
- Align significands (adjusting exponents)
- Add or subtract significands
- Normalize result

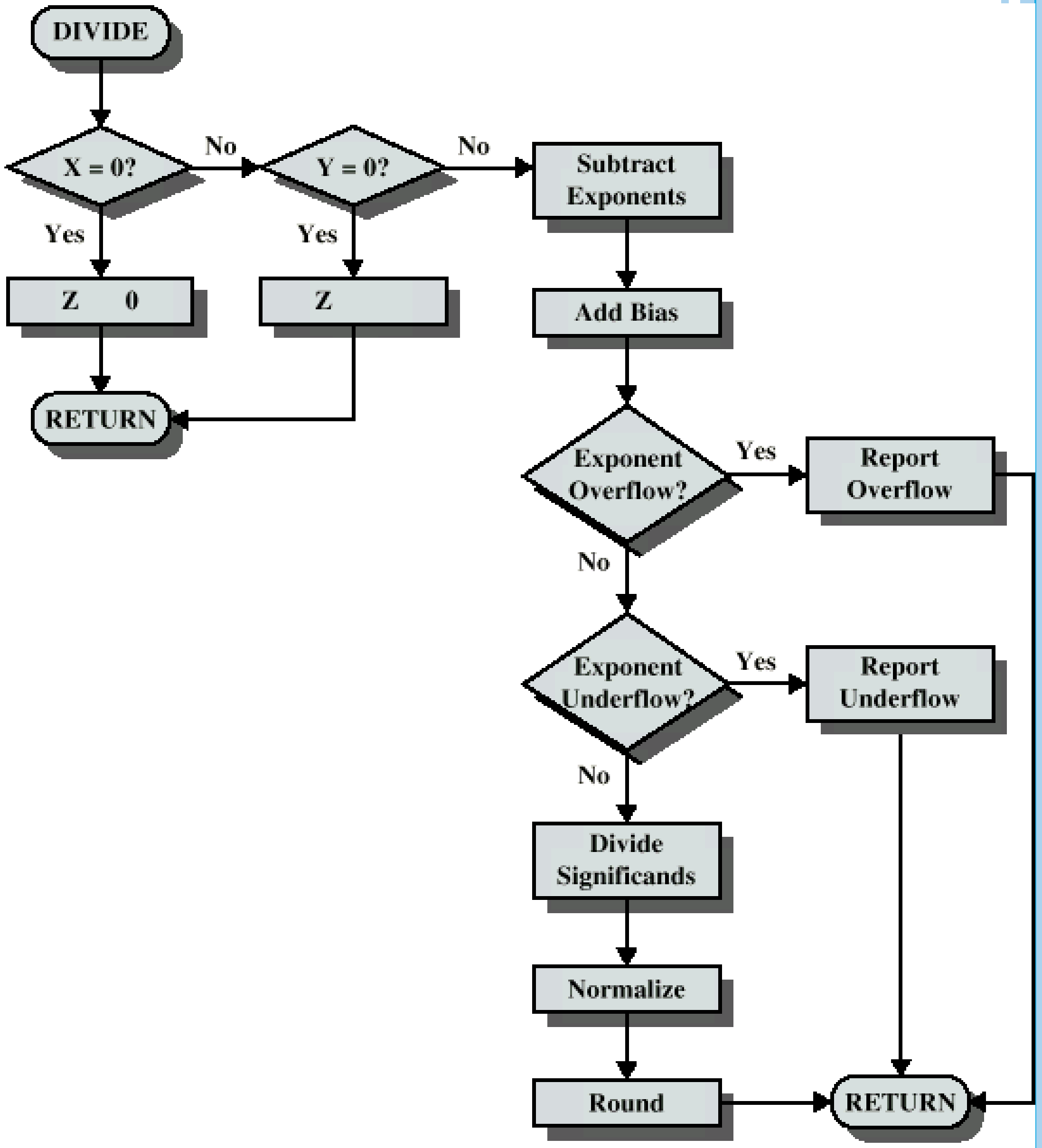
FP ARITHMETIC X/÷

- Check for zero
- Add/subtract exponents
- Multiply/divide significands (watch sign)
- Normalize
- Round
- All intermediate results should be in double length storage

FLOATING POINT MULTIPLICATION



FLOATING POINT DIVISION



REQUIRED READING

- Stallings Chapter 10
- IEEE (Institute of Electrical and Electronics

Engineers) Computer 754 on IEEE Web site