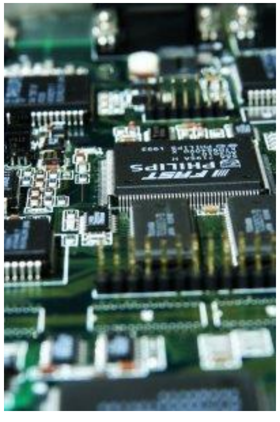
CSE 321b

Computer Organization (2)

تنظيم الحاسب (2)



3rd year, Computer Engineering

Spring 2018

Lecture #9

Dr. Ahmed Amer Shahin

http://www.aashahine.faculty.zu.edu.eg

Credits to Dr. Ahmed Abdul-Monem & Dr. Hazem Shehata for the slides

Chapter 10. Computer Arithmetic (*Cont.*)

Outline

- Integer Representation
  - Sign-Magnitude, Two's Complement, Biased
- Integer Arithmetic
  - Negation, Addition, Subtraction
  - Multiplication, Division
- Floating-Point Representation
  - IEEE 754
- Floating-Point Arithmetic
  - Addition, Subtraction
  - Multiplication, Division
  - Rounding

Real Numbers

- Numbers with fractions.
- Could be done in pure binary
  - $1001.1010 = 2^3 + 2^0 + 2^{-1} + 2^{-3} = 9.625$
- Where is the binary point?
- Fixed? 0 0 1 0 1 1 0 1 0 0.1 1 1 0 1 0
  - Very large/small numbers cannot be represented.
    - e.g., 0.0000001, 10000000000
  - Fractional part of the quotient in dividing very large numbers will be lost.
- Moving/floating?
  - How do you show where it is?
  - $976{,}000{,}000{,}000{,}000 = 9.76 \times 10^{14}$
  - $0.0000000000000976 = 9.76 \times 10^{-14}$

Can do the same with binary numbers. What do we need to store?

Floating-Point Representation

$$\pm S \times 2^E$$

Sign	Exponent	Significand (Mantissa)

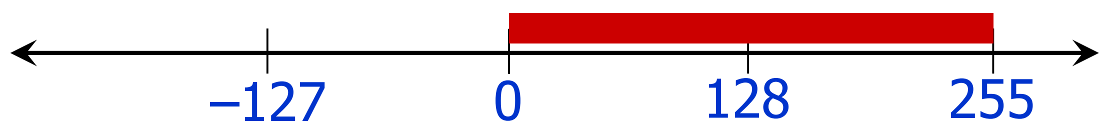
- The base 2 is the same for all numbers ➔ need not be stored.
- Number is stored in a binary word with 3 fields:
  - Sign: +/–
  - Significand S
  - Exponent E
- Normal number: most significant digit of the significand (mantissa) is nonzero ➔ 1 for base 2 (binary).
- What number to store in the significand field? **0.001011**
  - Normal form: 1.011×2^{-3} ➔ Store only 011 in the significand field!
- There is an implicit 1 to the left of the binary point (normalized).
- Exponent indicates place value (floating-point position).

Floating-Point Representation
Biased Exponent

$\pm S \times 2^{E}$

Sign	Biased Exponent	Significand (Mantissa)

- k-bit unsigned exponent E′ ranges from 0 to $2^k - 1$
  - e.g., 8-bit exponent: $0 \le E' \le 255$
- The stored exponent E′ is a biased exponent
  - $E' = E + (2^{k-1} - 1)$ bias
  - e.g., for 8-bit exponent, $E' = E + 127$
  - $-127 \le E \le 128$



- Why?
  - Nonnegative floating-point numbers can be treated as unsigned integers for comparison purposes.
  - This is not true for 2′s comp. or sign-magnitude representations.

Normalization

- FP numbers are usually normalized.
  - i.e., exponent is adjusted so that leading bit (MSB) of mantissa is non-zero, i.e., 1.
  - c.f., Scientific notation where numbers are normalized to give a single digit before the decimal point, e.g. 3.123×10^3.
- Since the MSB of mantissa is always 1, there is no need to store it!

Floating-Point Examples

1717698.56

1.638125×2^{20}

$1.1010001 \times 2^{10100}$

Positive ➔ sign bit = 0
E′ = E + 127 = 10100 + 1111111 = 10010011
Mantissa = 1010001 0000000000000000

0 10010011 10100010000000000000000

-1717698.56

-1.638125×2^{20}

$-1.1010001 \times 2^{10100}$

Negative ➔ sign bit = 1
E′ = E + 127 = 10100 + 1111111 = 10010011
Mantissa = 1010001 0000000000000000

1 10010011 10100010000000000000000

…

1.638125×2^{-20}

$1.1010001 \times 2^{-10100}$

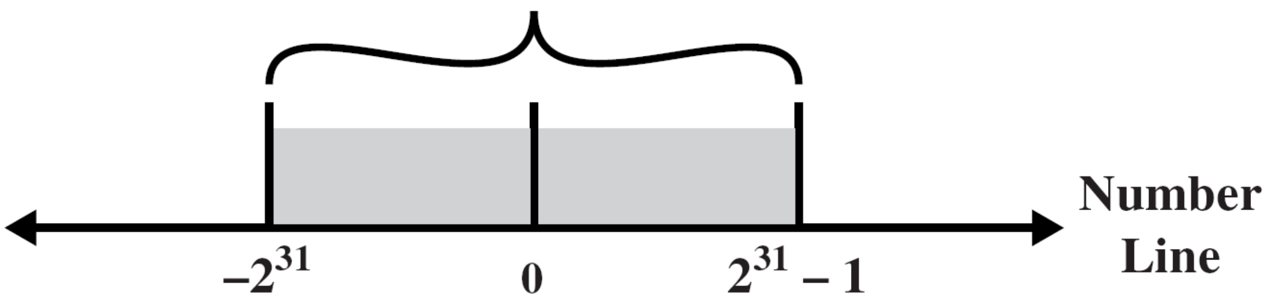
Positive ➔ sign bit = 0
E′ = E + 127 = −10100 + 1111111 = 01101011
Mantissa = 1010001 0000000000000000

0 01101011 10100010000000000000000

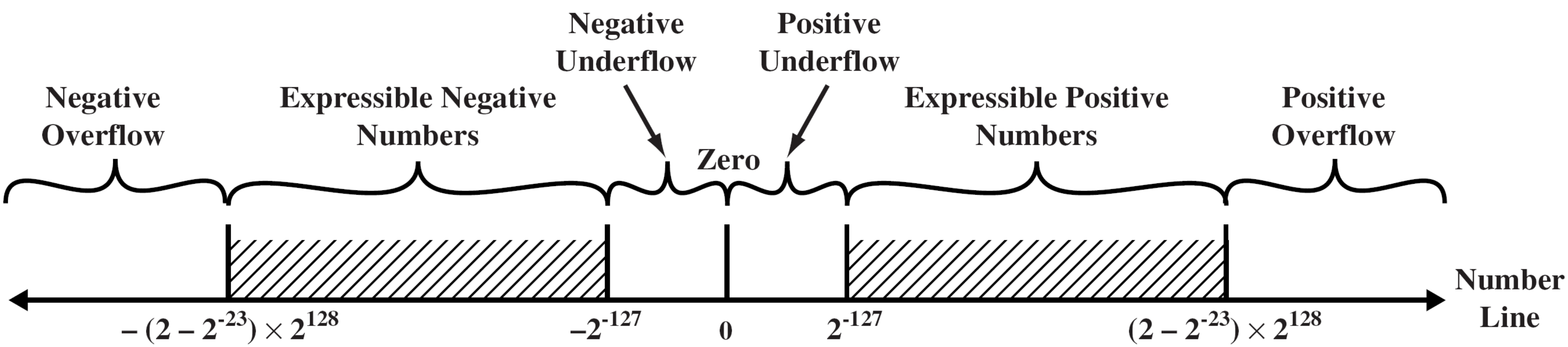
FP Ranges (32-bit)

- 32-bit FP number, 8-bit exponent, 23-bit mantissa.
- Largest +ve number $(2 – 2^{–23}) \times 2^{128}$
  - Largest true exponent: 128
  - Largest mantissa: $1 + (1 – 2^{–23}) = 2 – 2^{–23}$ (0.111...11)
- Smallest +ve number $2^{–127}$
  - Smallest true exponent: –127
  - Smallst mantissa: 1
- Smallest –ve number $– (2 – 2^{23}) \times 2^{128}$
- Largest –ve number $–2^{–127}$
- Accuracy
  - The effect of changing LSB of mantissa.
  - 23-bit mantissa $2^{-23} \approx 1.2 \times 10^{-7}$
  - About 6 decimal places.

Expressible Numbers (32-bit)



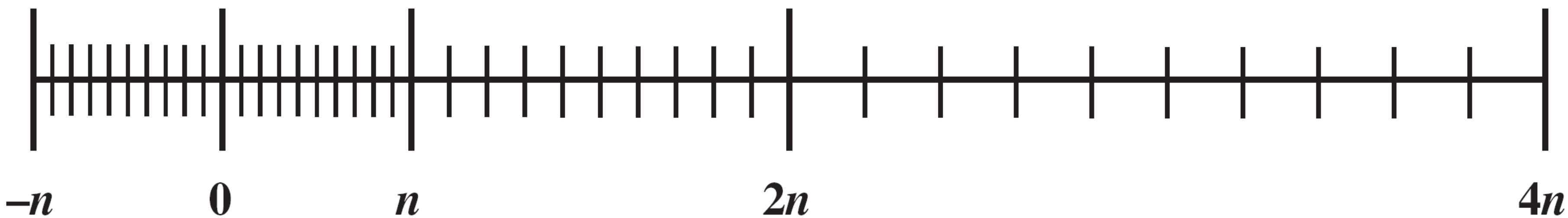
(a) Twos Complement Integers



(b) Floating-Point Numbers

Density of Floating Point Numbers

- 32-bit FP number ➔ 2^{32} different values represented.
- No more individual values are represented with floating-point numbers. Numbers are just spread out.
- Numbers represented in the FP representation are not spaced evenly along the line number. Why?
- Range-precision tradeoff
  - More bits for exponent ➔ wider range & less precision
  - Reason: there is a fixed number of values that can be represented!
  - To increase both range and precision ➔ use more bits!!!

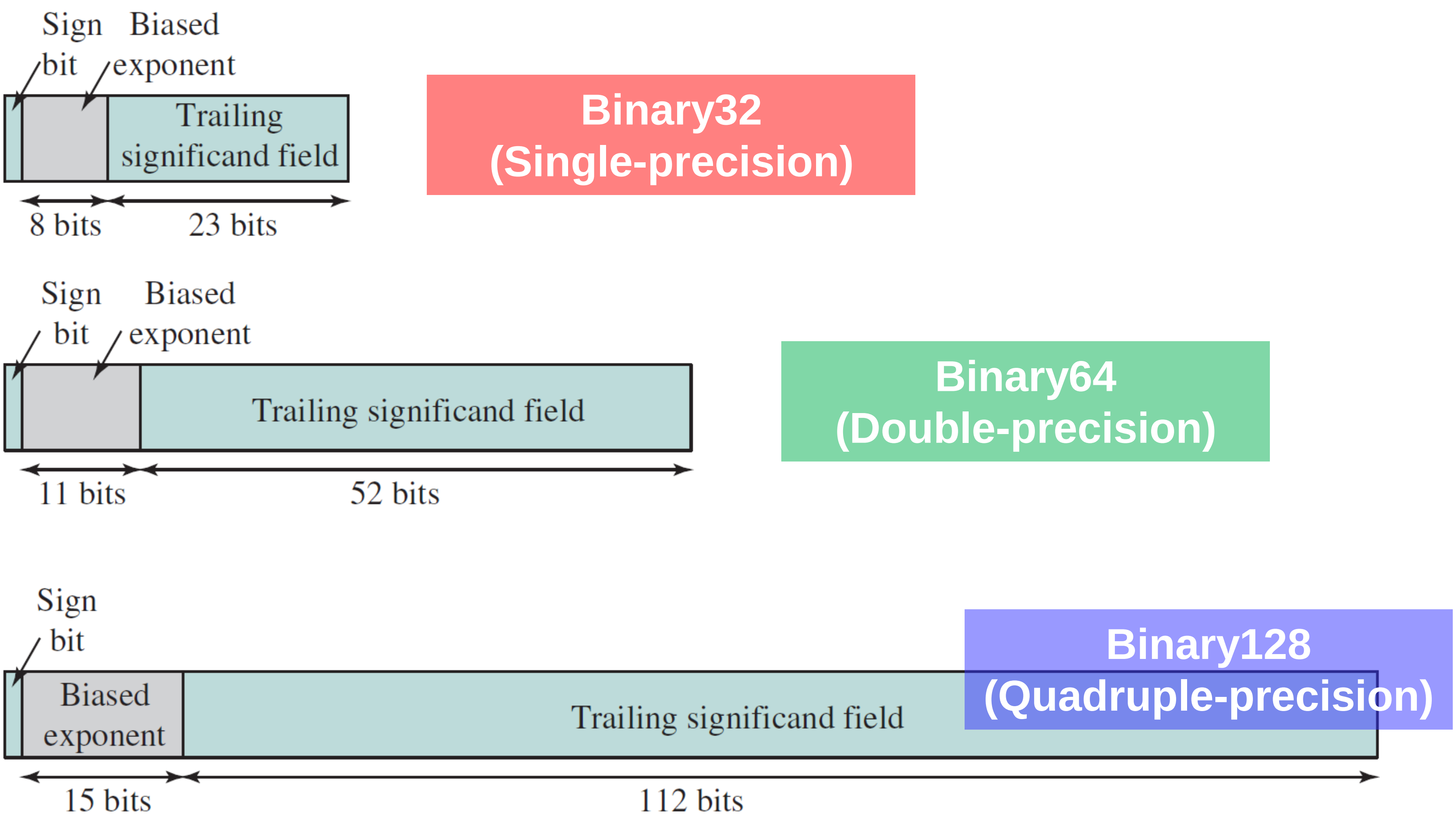


IEEE 754

- Standard for floating-point representation.
- Adopted 1985 and revised 2008.
- IEEE 754-2008 defines many FP formats for different purposes:

Format	Format Type		
	Arithmetic Format	Basic Format	Interchange Format
binary16			X
binary32	X	X	X
binary64	X	X	X
binary128	X	X	X
binary{k} ($k = n \times 32$ for $n > 4$)	X		X
decimal64	X	X	X
decimal128	X	X	X
decimal{k} ($k = n \times 32$ for $n > 4$)	X		X
extended precision	X		
extendable precision	X		

IEEE 754 - Binary32/64/128 Formats



IEEE 754 - Binary32/64/128 Interpretations

	Sign	Biased Exponent	Fraction	Value	
positive zero	0	0	0	0	
negative zero	1	0	0	-0	
plus infinity	0	all 1s	0	∞	
minus infinity	1	all 1s	0	$-\infty$	
quiet NaN	0 or 1	all 1s	$\neq 0$; first bit $= 1$	qNaN	
signaling NaN	0 or 1	all 1s	$\neq 0$; first bit $= 0$	sNaN	
positive normal nonzero	0	$0 < e < 255$	f	$2^{e-127}(1.f)$	32
negative normal nonzero	1	$0 < e < 255$	f	$-2^{e-127}(1.f)$	
positive subnormal	0	0	$f \neq 0$	$2^{-126}(0.f)$	
negative subnormal	1	0	$f \neq 0$	$-2^{-126}(0.f)$	
positive normal nonzero	0	$0 < e < 2047$	f	$2^{e-1023}(1.f)$	64
negative normal nonzero	1	$0 < e < 2047$	f	$-2^{e-1023}(1.f)$	
positive subnormal	0	0	$f \neq 0$	$2^{-1022}(0.f)$	
negative subnormal	1	0	$f \neq 0$	$-2^{-1022}(0.f)$	
positive normal nonzero	0	$0 < e < 32767$	f	$2^{e-16383}(1.f)$	128
negative normal nonzero	1	$0 < e < 32767$	f	$-2^{e-16383}(1.f)$	
positive subnormal	0	0	$f \neq 0$	$2^{-16382}(0.f)$	
negative subnormal	1	0	$f \neq 0$	$-2^{-16382}(0.f)$	

IEEE 754 - Binary32/64/128 Parameters

Parameter	Format		
	Binary32	Binary64	Binary128
Storage width (bits)	32	64	128
Exponent width (bits)	8	11	15
Exponent bias	127	1023	16383
Maximum exponent	127	1023	16383
Minimum exponent	−126	−1022	−16382
Approx normal number range (base 10)	$10^{-38}, 10^{+38}$	$10^{-308}, 10^{+308}$	$10^{-4932}, 10^{+4932}$
Trailing significand width (bits)*	23	52	112
Number of exponents	254	2046	32766
Number of fractions	2^{23}	2^{52}	2^{112}
Number of values	1.98×2^{31}	1.99×2^{63}	1.99×2^{128}
Smallest positive normal number	2^{-126}	2^{-1022}	2^{-16362}
Largest positive normal number	$2^{128} - 2^{104}$	$2^{1024} - 2^{971}$	$2^{16384} - 2^{16271}$
Smallest subnormal magnitude	2^{-149}	2^{-1074}	2^{-16494}

Note: *not including implied bit and not including sign bit

IEEE 754 - NaNs

- NaN:
  - Symbolic entity encoded in FP format
  - Types: Signaling (sNaN) or Quiet (qNaN)
  - Both types have the same format:

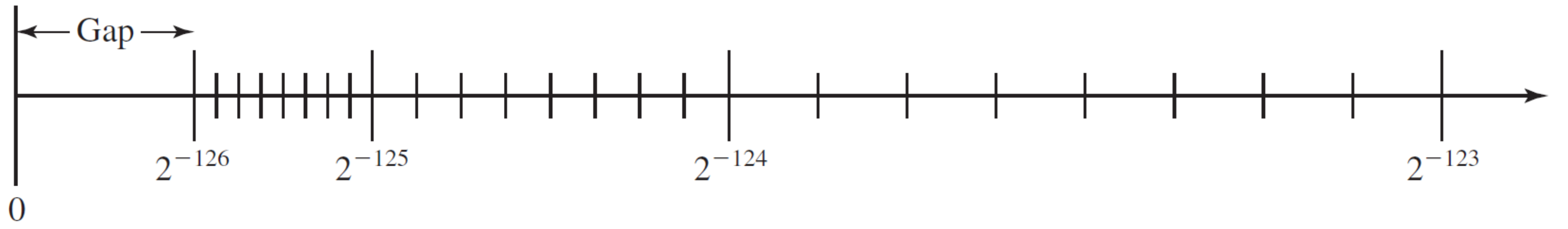
S= 0 or 1	E= 1111...11	F ≠ 0000..00

  - F distinguishes between the two types:
    - F=**0**xxxx..xx ➔ sNaN, F=**1**xxxx..xx ➔ qNaN
- Signaling NaN:
  - Signals an invalid operation exception whenever it appears as an operand. Ex.: uninitialized variables
- Quite NaN:
  - Propagates without signaling exceptions.

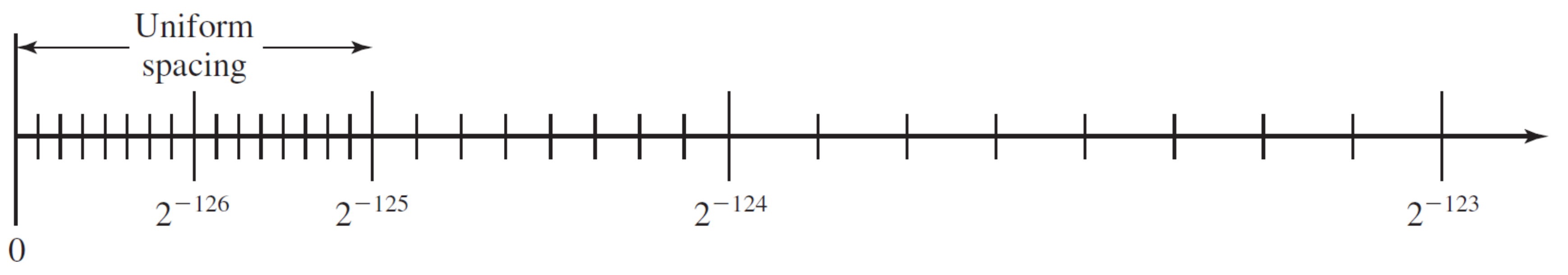
IEEE 754 - Quiet NaN

Operation	Quiet NaN Produced By
Any	Any operation on a signaling NaN
Add or subtract	Magnitude subtraction of infinities: $(+\infty) + (-\infty)$ $(-\infty) + (+\infty)$ $(+\infty) - (+\infty)$ $(-\infty) - (-\infty)$
Multiply	$0 \times \infty$
Division	$\frac{0}{0}$ or $\frac{\infty}{\infty}$
Remainder	x REM 0 or ∞ REM y
Square root	$\sqrt{x}$, where $x < 0$

IEEE 754 - Effect of Subnormal Numbers



(a) 32-Bit format without subnormal numbers

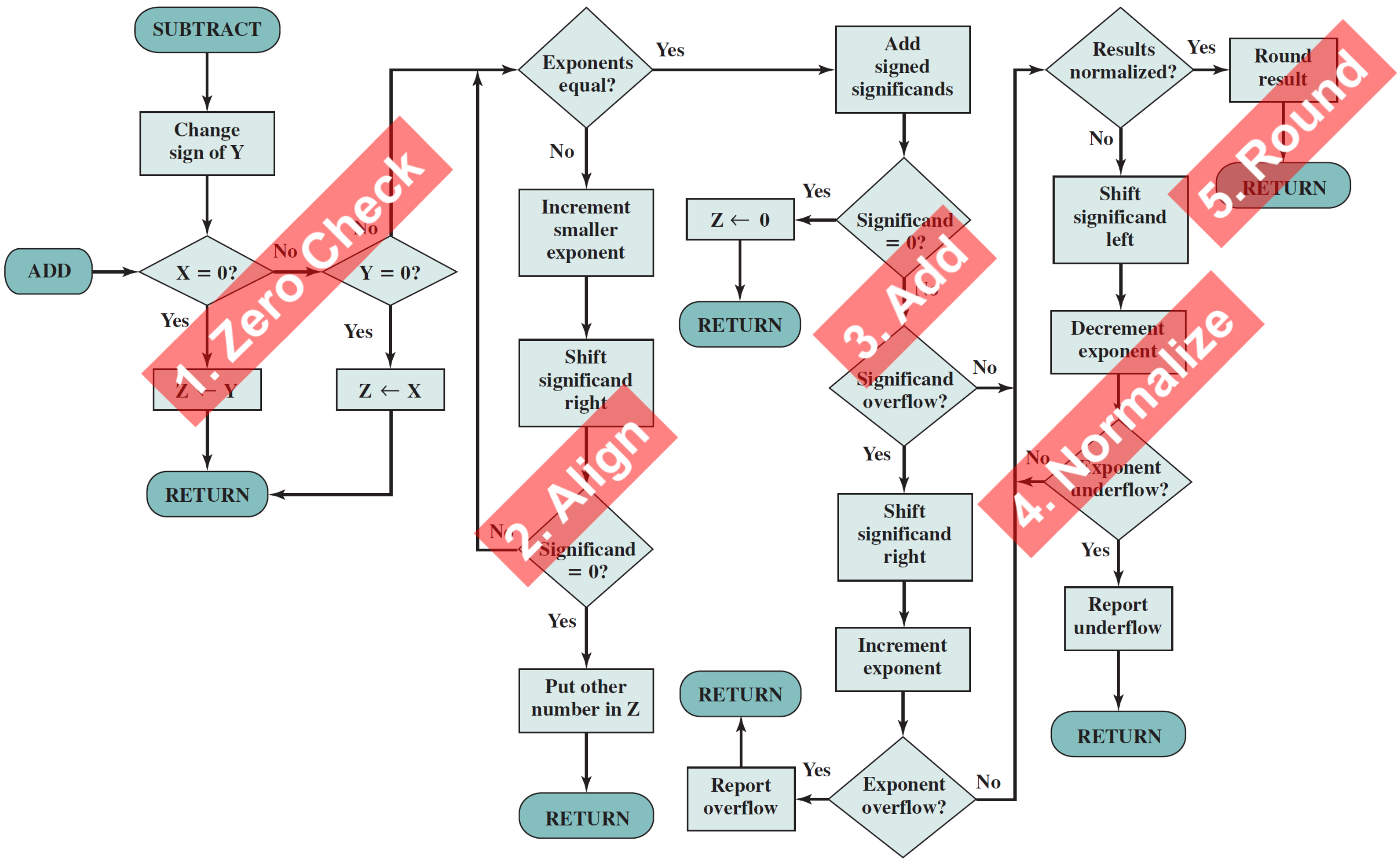


(b) 32-Bit format with subnormal numbers

FP Arithmetic +/-

- Algorithm:
  1. Check for zeros.
  2. Align significands (adjusting exponents).
  3. Add or subtract significands.
  4. Normalize result.
  5. Round result.

FP Addition & Subtraction Flowchart



Reading Material

- Stallings, Chapter 10:
  - Pages 341-352
  - Pages 356-358