Computer Organization & Architecture

Chapter 9 – FP Numbers and Operations

Zhang Yang 张杨

cszyang@scut.edu.cn

Autumn 2021

Floating-point Numbers and Operations (1)

- What can be represented in *n* bits?
  - Unsigned: 0 to 2^n-1
  - 2's complement: -2^{n-1} to $2^{n-1} - 1$
  - But, what about?
    - Very large numbers? 9,349,398,989,787,762,244,859,087,678
    - Very small number? 0.00000000000000000000045691
    - Fractional values? 0.35
    - Mixed numbers? 10.28
    - Irrationals(无理数)? π

Floating point ‡ Real numbers

- Floating point numbers in a computer do not behave like their conceptual counterparts, the real numbers:
  - Limited space to represent the numbers
  - Finite length encoding (e.g., 1/3 = 0.333333333…..)
  - Limited resolution: can only represent exactly values of the form x $\times 2^y$
  - Numbers like 1/10 cannot be represented precisely
- Floating point number is a major source of errors and bugs in programs.

Number Format

- According to whether the position of binary point is fixed, there are two number formats:
  - Fixed-point numbers
    - E.g., integers, have an implied binary point at the right end of them.
  - Floating-point numbers

Fixed-point Representation (1)

- A fixed-point notation is any number in which the number of bits to the right of the binary point does not change.
  - Unsigned integers: with no bits to the right of the binary point
  - Signed integers: with no bits to the right of the binary point.
  - Signed fractions:the binary point is to the right of the sign bit.

Fixed-point Representation (2)

- Let $X = x_n \ldots x_0$ be a fixed-point number

  - If X is a pure fraction
    - The binary point is between x_n and x_{n-1}
    - Range: $-1 \le V(X) \le 1 - 2^{-n}$
  - If X is an integer
    - The binary point is to the right of x_0
    - Range: $-2^n \le V(X) \le 2^n - 1$

Fixed-point Representation (3)

- Limitation
  - Very large integers can not be represented, nor can very small fractions.
  - Example: Consider the range of values representable in a 32-bit, signed, fixed-point format.
    - Interpreted as integers $-2^{31} \le V(X) \le 2^{31}-1$

$$V(X) \in [-2.15\times10^{9},0],[0,+2.15\times10^{9}]$$

    - Interpreted as fractions $-1 \le V(X) \le 1-2^{-31}$

$$V(X) \in [-1,-4.55\times10^{-10}],[+4.55\times10^{-10},+1]$$

Fixed-point Representation (4)

- Fixed-point Representation (ctd.)
  - Limitation
    - Example: Consider the range of values representable in a 32-bit, signed, fixed-point format.
      - In scientific calculations

      Avogadro's constant $6.0247 \times 10^{23}\ \text{mol}^{-1} =$

      $0.60247 \times 10^{24}\ \text{mol}^{-1}$

      Planck's constant $6.6254 \times 10^{-27}\text{erg.s} =$

      $0.66254 \times 10^{-26}\text{erg.s}$

Floating-point Representation (1)

- Scientific Notation (Decimal Numbers)
  - A notation that renders numbers with a single digit to the left of the decimal point.
  - The following are equivalent representations of 1,234

$$
\begin{aligned}
123{,}400.0 &\times 10^{-2} \\
12{,}340.0 &\times 10^{-1} \\
1{,}234.0 &\times 10^{0} \\
123.4 &\times 10^{1} \\
12.34 &\times 10^{2} \\
1.234 &\times 10^{3} \\
0.1234 &\times 10^{4}
\end{aligned}
$$

The representations differ in that the decimal place – the "point" -- "floats" to the left or right (with the appropriate adjustment in the exponent).

Floating-point Representation (2)

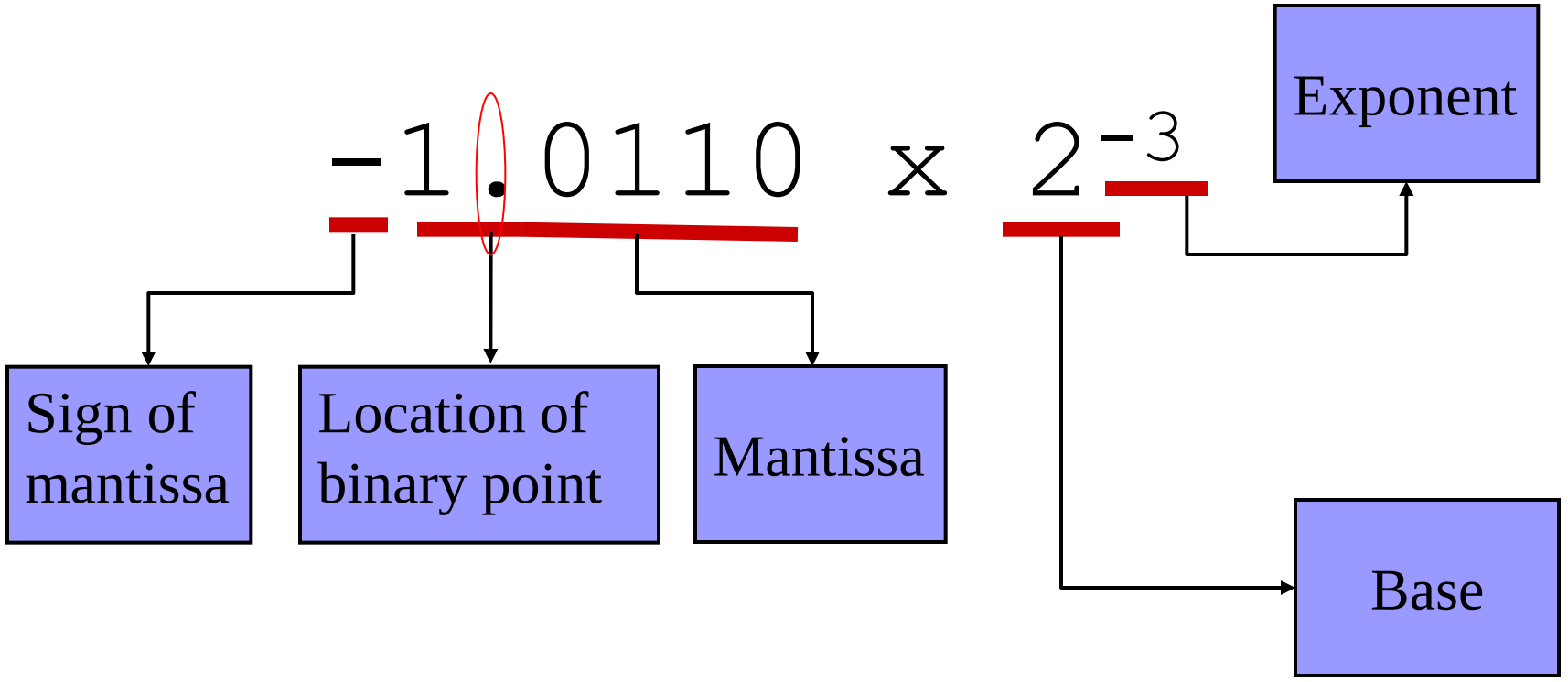
- Scientific Notation (Decimal Numbers) (ctd.)
  - Normalized Number
    - A number in scientific notation that has no leading 0s is called a normalized number.
      - 1.0×10^{-9} (√) a normalized scientific notation
      - 0.1×10^{-10} (X) not a normalized scientific notation
    - Normalizing means
      - Shifting the decimal point until we have the right number of digits to its left (normally one).
      - Adding or subtracting from the exponent to reflect the shift.

Floating-point Representation (3)

- Scientific Notation (Binary Numbers)
  - E.g. 1.0×2^{-1}
  - The position of the binary point is variable and is automatically adjusted as computation proceeds.
  - The position of the binary point must be given explicitly in the floating-point representation.

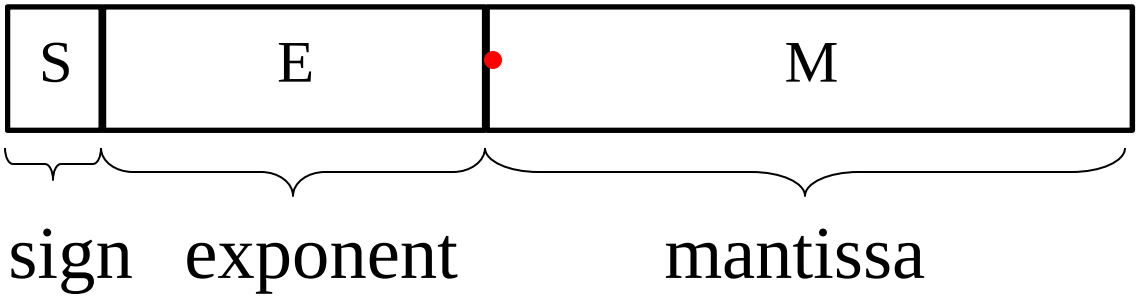
Floating-point Representation (4)

- Parts of a floating-point number



Floating-point Numbers in Computers (1)

- Numerical Form
  - $(-1)^S \times M \times 2^e$
    - Sign bit *S* determines whether number is negative or positive
    - Mantissa M, a fraction in sign-magnitude or 2's complement representation, containing the significant digits
    - Exponent E, in 2's complement or biased notation, the power of base
- Encoding



Floating-point Numbers in Computers (2)

- Excess or Biased Notation
  - A negative exponent in 2's complement looks like a large exponent.
  - A fixed value is subtracted from the exponent field to get the true exponent.
  - $E = e + (2^{k-1} - 1)$
    - e is the actual exponent
    - k is the number of bits in the exponent
  - Note
    - When the bits of a biased representation are treated as unsigned integers, the relative magnitudes of the numbers do not change.

Decimal Representation	Biased Representation	Two's complement representation
+7	1111	0111
+6	1110	0110
+5	1101	0101
+4	1100	0100
+3	1011	0011
+2	1010	0010
+1	1001	0001
+0	1000	0000

- 0	0111	0000
- 1	0110	1111
- 2	0101	1110
- 3	0100	1101
- 4	0011	1100
- 5	0010	1011
- 6	0001	1010
- 7	0000	1001
- 8	_	1000

Floating-point Numbers in Computers (3)

- Normalization of Floating-point Numbers
  - In normalized binary, the first bit of the mantissa should always be a 1.
    - Example: normalize 0 000010 000110101
    - 0 111111 110101000

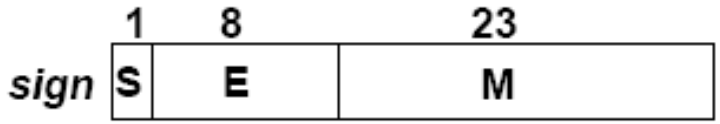
IEEE 754 Standard (1)

- Institute of Electrical and Electronics Engineers
- Most common standard for representing floating point numbers.
- Established in 1985 as uniform standard for floating point arithmetic and revised in 2008.
- This standard was developed to facilitate the portability of programs from one processor to another and encourage the development of sophisticated, numerically oriented programs.
- Supported by all major CPUs

IEEE 754 Standard (2)

- Single Precision

single precision
(32 bits, float in C)



exponent:
excess 127
binary integer

Mantissa or significand
sign + magnitude, normalized
binary significand w/ hidden
one bit: 1.M

actual exponent is
e = E – 127 (bias)

$0 < E < 255$

$$X = (-1)^S \, 2^{E-127} \, (1.M)$$

Magnitude of numbers that can be represented is in the range:

$$2^{-126}(1.0) \quad \text{to} \quad 2^{127}(2 - 2^{-23})$$

Positive Numbers Range

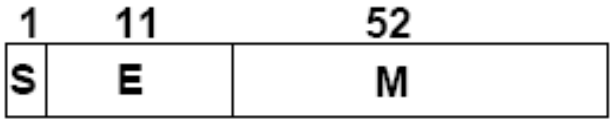
which is approximately:

$$1.8 \times 10^{-38} \quad \text{to} \quad 3.40 \times 10^{38}$$

IEEE 754 Standard (3)

- Double Precision

double precision
(64 bits, double in C)



exponent:
excess 1023
binary integer

Mantissa or significand
sign + magnitude, normalized
binary significand w/ hidden
one bit: 1.M

actual exponent is
e = E – 1023 (bias)

$0 < E < 2047$

$$X = (-1)^S \, 2^{E-1023} (1.M)$$

Magnitude of numbers that can be represented is in the range:

$2^{-1022}(1.0)$ to $2^{1024}(2-2^{-52})$ Positive Numbers Range

which is approximately:

2.2×10^{-308} to 1.8×10^{308}

IEEE 754 Standard (4)

- A computer must provide at least single-precision representation to conform to the IEEE standard.
- Double-precision representation is optional.
- Extended single-precision (more than 32 bits) /Extended double-precision (more than 64 bits)
  - Help to reduce the size of the accumulated round-off error in a sequence of calculations.
  - Enhance the accuracy of evaluation of elementary functions such as sine, cosine, and so on.

IEEE 754 Standard (5)

- Trade-off between “accuracy” and “range”
  - Increasing the size of **mantissa** enhances **accuracy.**
  - Increasing the size of **exponent** increases the **range.**

IEEE 754 Standard (6)

- Special Values
  - Zero
    - S = 0/1, E = 0, M = 0 (0.M) Value = ± 0
    - An exponent field of zero is special; it indicates that there is no implicit leading 1 on the mantissa.
  - Infinity
    - Operation that overflows
      - E.g., +1.0/+0.0 = +infinity
    - S = 0/1, E = 255 or 2047, M = 0 Value = ±infinity

IEEE 754 Standard (7)

- Special Values (ctd.)
  - NaN (Not a Number)
    - Represents case when no numeric value can be determined (invalid operation)
      - E.g., sqrt(–1),
      - E.g., the logarithm of a negative number
      - E.g., the inverse sine or cosine of a number that is less than -1, or greater than +1.
    - Represents a missing value, may also be explicitly be assigned to variables.
    - S = 0/1, E = 255 or 2047, M ≠ 0 Value = NaN

IEEE 754 Standard (8)

- Special Values (ctd.)
  - Denormal Numbers
    - There is no implied 1 to the left of the binary point.
    - All denormalized numbers are assumed to have an exponent field of 1 – bias.
    - Numbers very close to 0.0
    - Note that we cannot normalize this value.
    - Zero is effectively a denormal number.
    - Lose precision as get smaller
    - "Gradual underflow"
    - S = 0/1, E = 0, M ≠ 0
      - Value = $\pm$ 0.M $\times 2^{-126}$
      - Value = $\pm$ 0.M $\times 2^{-1022}$

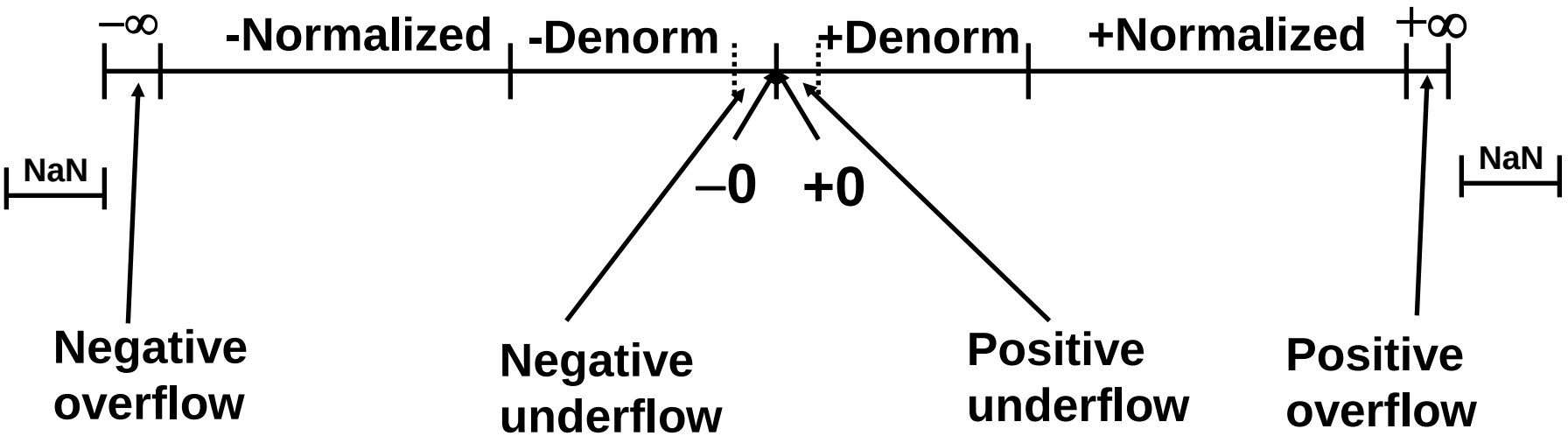
IEEE 754 Standard (9)

- Summary

Normalized:	±	0<E<max	Any bit pattern
Denormalized:	±	0	Any nonzero bit pattern
zero:	±	0	0
Infinity:	±	11...1	0
NaN:	±	11...1	Any nonzero bit pattern

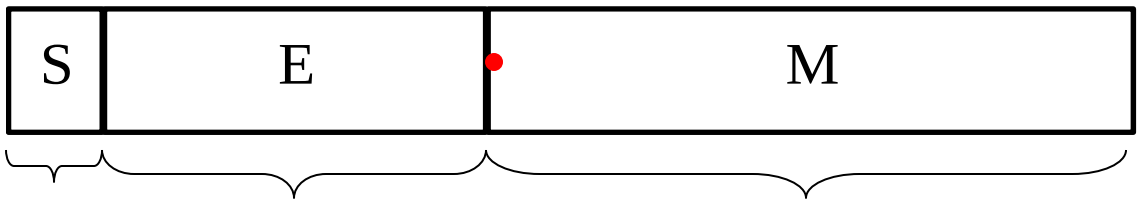
IEEE 754 Standard (10)

- Summary (ctd.)



Summary (1)

- 知识点FP Representation
  - $(-1)^S \times M \times 2^e$



  - IEEE 754 Standard
    - Single Precision
    - Double Precision
    - Four Special Values

Summary (2)

- 掌握程度
  - 给定一个十进制小数，熟练转换成给定浮点格式的浮点数。
  - 给定一个机器数表示的单精度/双精度浮点数，熟练转换成十进制数。
  - IEEE单精度浮点数表示方法和表示范围
  - IEEE双精度浮点数表示方法和表示范围

Exercise (1)

1. In single-precision format of IEEE 754 floating point number standard, instead of the signed exponent e, what is the value E actually stored in the exponent field?
   - A. E=e+255
   - B. E=e+127
   - C. E=e+256
   - D. E=e+128

Exercise (2)

2. In IEEE 754 standard for representing floating-point numbers of 32 bits, the sign of the number is given 1 bit, the exponent of the scale factor is allocated 8 bits, and the mantissa is assigned 23 bits. What is the maximum normalized positive number that 32-bit representation can represent?

A. $+(2-2^{-23})\times 2^{127}$

B. $+(1-2^{-23})\times 2^{127}$

C. $+(2-2^{-23})\times 2^{255}$

D. $2^{127}-2^{-23}$

Exercise (3)

3. In double-precision format of IEEE 754 floating point number standard, instead of the signed exponent e, what is the value E actually stored in the exponent field?
   - A. E=e+2047
   - B. E=e+1023
   - C. E=e+2048
   - D. E=e+1024

Exercise (4)

4. In IEEE 754 standard for representing floating-point numbers of 64 bits, the sign of the number is given 1 bit, the exponent of the scale factor is allocated 11 bits, and the mantissa is assigned 52 bits. What is the maximum normalized positive number that 64-bit representation can represent?

A. $+(2-2^{-50})\times 2^{1025}$

B. $+(1-2^{-52})\times 2^{1025}$

C. $+(2-2^{-52})\times 2^{1024}$

D. 2^{-1022}

Content of this lecture

- 9.7 Floating-Point Numbers and Operations
  - Fix-point Representation
  - Floating-point Representation
  - Floating-point Numbers in Computers
  - IEEE-754 Standard
  - Summary of FP Representation
  - Arithmetic Operations on FP
  - Problems Considerable in FP Arithmetic
  - Implementing Floating-Point Operations
  - Solved Problems Example 9.5 (P376)
  - Summary of FP Arithmetic

Arithmetic Operations on FP (1)

- Addition and Subtraction
  - Alignment
    - If their exponents differ, it is necessary to manipulate the two summands so that the two exponents are equal.
    - Example: Decimal addition $(123 \times 10^{0})+(456 \times 10^{-2})$

$(123 \times 10^{0})+(456 \times 10^{-2}) = (123 \times 10^{0})+(4.56 \times 10^{0})$

    - The alignment is achieved by shifting the magnitude portion of the mantissa right 1 digit and incrementing the exponent until the two exponents are equal.

Arithmetic Operations on FP (2)

- Add/Subtract Rule (for IEEE single precision)
  - ① Choose the number with the smaller exponent and shift its mantissa right a number of steps equal to the difference in exponents.
  - ② Set the exponent of the result equal to the larger exponent.
  - ③ Perform addition/subtraction on the mantissas and determine the sign of the result.
  - ④ Normalize the resulting value, if necessary.

Arithmetic Operations on FP (3)

- Multiplication and Division
  - Multiply Rule (for IEEE single precision)
    - ① Add the exponents and subtract 127.
    - ② Multiply the mantissas and determine the sign of the result.
    - ③ Normalize the resulting value, if necessary.
  - Divide Rule (for IEEE single precision)
    - ① Subtract the exponents and add 127.
    - ② Divide the mantissas and determine the sign of the result.
    - ③ Normalize the resulting value, if necessary.

Content of this lecture

- 9.7 Floating-Point Numbers and Operations
  - Fix-point Representation
  - Floating-point Representation
  - Floating-point Numbers in Computers
  - IEEE-754 Standard
  - Summary of FP Representation
  - Arithmetic Operations on FP
  - Problems Considerable in FP Arithmetic
  - Implementing Floating-Point Operations
  - Solved Problems Example 9.5 (P376)
  - Summary of FP Arithmetic

Problems Considerable in FP Arithmetic (1)

- Guard Bits
- Truncation
- Normalization
- Overflow

Problems Considerable in FP Arithmetic (2)

- Guard Bits
  - The additional bits retained in the mantissa are referred to as guard bits.
  - The guard bits are used to pad out the right end of the mantissa with 0s.
  - It is important to retain guard bits during the intermediate steps. This yields maximum accuracy in the final results.
  - Example: X= 1.00…00 $\times$ 2^1 , Y= 1.11…11 $\times$ 2^0 (X and Y are all in IEEE single precision format). Calculate Z= X – Y.

Problems Considerable in FP Arithmetic (3)

- Guard Bits (ctd.)
  - Example (ctd.)
    - Without guard bits

$$X = 1.000\ldots00 \times 2^{1}$$
$$-Y = 0.111\ldots11 \times 2^{1}$$
$$Z = 0.000\ldots01 \times 2^{1}$$
$$= 1.000\ldots00 \times 2^{-22}$$

    - With guard bits

$$X = 1.000\ldots00\ \ \underline{0} \times 2^{1}$$
$$-Y = 0.111\ldots11\ \ \underline{1} \times 2^{1}$$
$$Z = 0.000\ldots00\ \ \underline{1} \times 2^{1}$$
$$= 1.000\ldots00\ \ \underline{0} \times 2^{-23}$$

Problems Considerable in FP Arithmetic (4)

- Truncation
  - Chopping
    - Remove the guard bits and make no changes in the retained bits.
    - Example: Truncate $0.b_{-1}b_{-2}b_{-3}$ 010 to three bits.
      - $0.b_{-1}b_{-2}b_{-3}$ 010 $\longrightarrow$ $0.b_{-1}b_{-2}b_{-3}$
      - Actually, all fractions in the range $0.b_{-1}b_{-2}b_{-3}$ 000 to $0.b_{-1}b_{-2}b_{-3}$ 111 are truncated to $0.b_{-1}b_{-2}b_{-3}$.
      - The error range in the 3-bit result is from 0 to 0.000111.
    - Error Range
      - The error range in chopping is from 0 to almost 1 in the least significant position of the retained bits.

Problems Considerable in FP Arithmetic (5)

- Von Neumann Rounding
  - ① If the bits to be removed are all 0s, they are simply dropped, with no changes to the retained bits.
  - ② If any of the bits to be removed are 1, the least significant bit of the retained bits is set to 1.
  - Example: Truncate $0.b_{-1}b_{-2}b_{-3}$ 000 – $0.b_{-1}b_{-2}b_{-3}$ 111 to three bits.
    - ① $0.b_{-1}b_{-2}b_{-3}$ 000 ⟶ $0.b_{-1}b_{-2}b_{-3}$
    - ② $0.b_{-1}b_{-2}b_{-3}$ 001 – $0.b_{-1}b_{-2}b_{-3}$ 111 ⟶ $0.b_{-1}b_{-2}1$
  - Error Range
    - The error range in Von Neumann rounding is between – 1 and +1 in the LSB position of the retained bits.

Problems Considerable in FP Arithmetic (6)

- Truncation (ctd.)
  - Rounding
    - If there is a 0 in the MSB position of the bits being removed, like chopping method.
    - If there is a 1 in the MSB position of the bits being removed, a 1 is added to the LSB position of the bits to be retained.
    - Example: Truncate $0.b_{-1}b_{-2}b_{-3}\ 000 - 0.b_{-1}b_{-2}b_{-3}\ 111$ to three bits.
      - ① $0.b_{-1}b_{-2}b_{-3}\ 000 - 0.b_{-1}b_{-2}b_{-3}\ 011 \longrightarrow 0.b_{-1}b_{-2}b_{-3}$
      - ② $0.b_{-1}b_{-2}b_{-3}\ 101 - 0.b_{-1}b_{-2}b_{-3}\ 111 \longrightarrow 0.b_{-1}b_{-2}b_{-3} + 0.001$

Problems Considerable in FP Arithmetic (7)

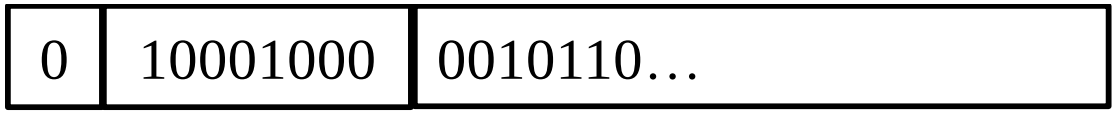
- Truncation (ctd.)
  - Rounding (ctd.)
    - Example: Truncate $0.b_{-1}b_{-2}b_{-3}\ 000 - 0.b_{-1}b_{-2}b_{-3}\ 111$ to three bits.
      - ③ $0.b_{-1}b_{-2}b_{-3}\ 100$(tie situation) : Choose the retained bits to be nearest even number.

      $0.b_{-1}b_{-2}0100 \longrightarrow 0.b_{-1}b_{-2}0$

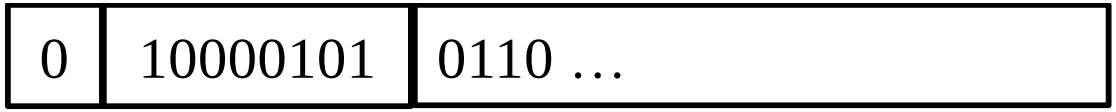
      $0.b_{-1}b_{-2}1100 \longrightarrow 0.b_{-1}b_{-2}1 + 0.001$
    - Error Range
      - The error range is – ½ to + ½ in the LSB position of the retained bits.
    - Rounding achieves the closest approximation to the number being truncated and is an unbiased technique.
    - Rounding is the default mode for truncation specified in the IEEE floating-point standard.

Problems Considerable in FP Arithmetic (8)

- Normalization
  - If a number is not normalized, it can always be put in normalized form by shifting the mantissa and adjusting the exponent.
  - Example
    - Value =+0.0010110… $\times 2^{9}$



    - Value =+1.0110… $\times 2^{6}$



Problems Considerable in FP Arithmetic (9)

- Overflow
  - As computation proceed, a number that does not fall in the representable range of normal numbers might be generated.
  - Exponent Overflow
    - A positive exponent exceeds the maximum possible exponent value.
    - Example: In IEEE single precision, e > 127
    - In some systems, it may be designated as plus infinity or minus infinity.
  - Exponent Underflow
    - A negative exponent is less than the minimum possible exponent value.
    - Example: In IEEE single precision, e < – 126
    - This means that the number is too small to be represented, it may be reported as 0.

Problems Considerable in FP Arithmetic (10)

- Problems Considerable in FP Arithmetic
  - Overflow(ctd.)
    - Mantissa Overflow
      - The addition of two mantissas of the same sign may result in a carry out of the most significant bit.
      - If so, the mantissa of the result is shifted right and the exponent is incremented.
    - Mantissa Underflow
      - In the process of aligning mantissas, digits may flow off the right end of the mantissa.
      - This can be resolved by using guard bits and some method of truncation.

Problems Considerable in FP Arithmetic (11)

- Some Useful and Interesting Pointers
  - How Java's Floating-Point Hurts Everyone Everywhere (W. Kahan, J.D. Darcy)

http://www.cs.berkeley.edu/~wkahan/JAVAhurt.pdf

  - Tutorial to Understand IEEE Floating-Point Errors

http://support.microsoft.com/kb/42980

  - What Every Computer Scientist Should Know About Floating-Point Arithmetic

http://docs.sun.com/source/806-3568/ncg_goldberg.html

  - Floating-Point Computing: A Comedy of Errors?

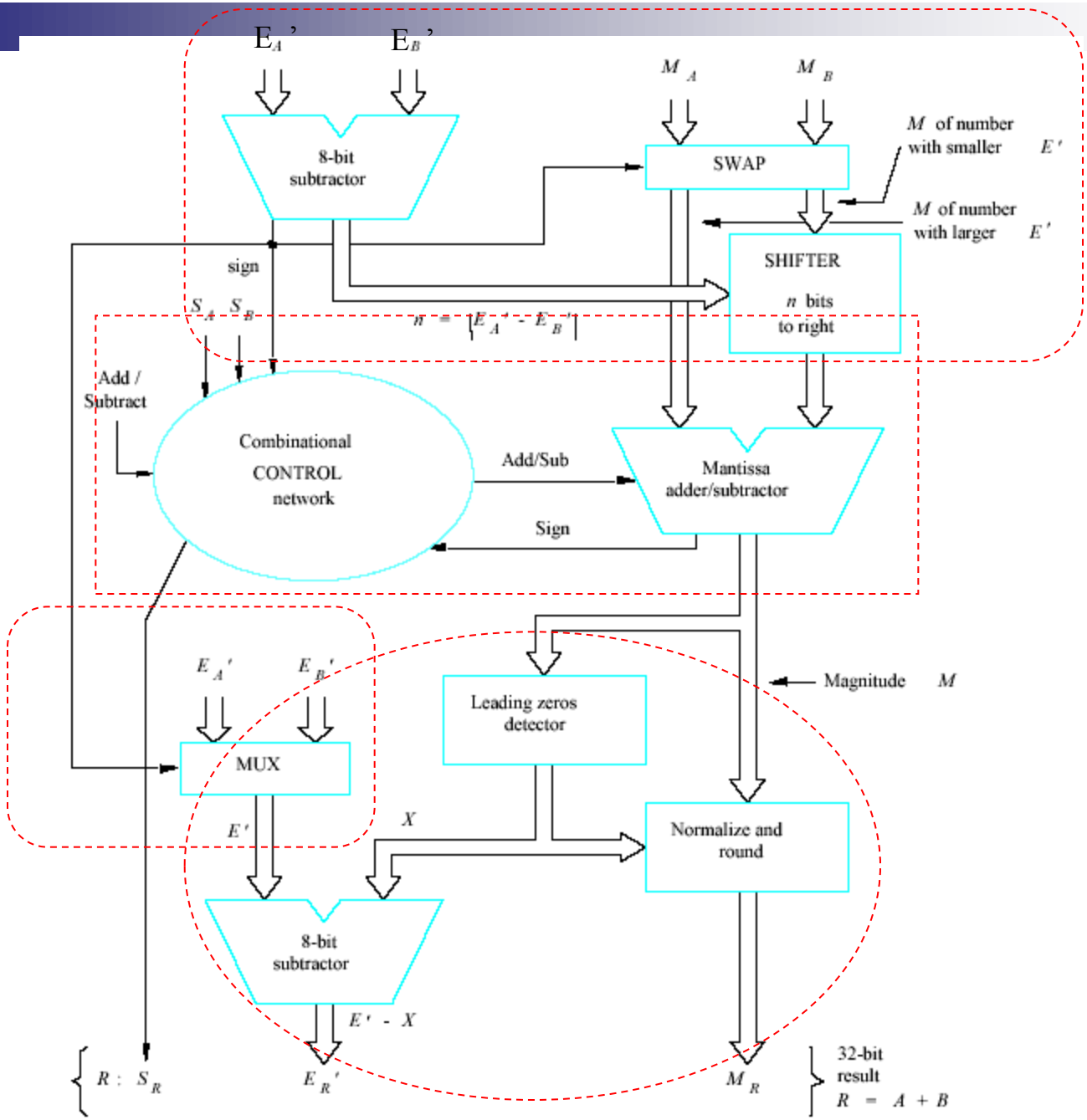
http://developers.sun.com/sunstudio/articles/fp_errors.html

Content of this lecture

- 9.7 Floating-Point Numbers and Operations
  - Fix-point Representation
  - Floating-point Representation
  - Floating-point Numbers in Computers
  - IEEE-754 Standard
  - Summary of FP Representation
  - Arithmetic Operations on FP
  - Problems Considerable in FP Arithmetic
  - Implementing Floating-Point Operations
  - Solved Problems Example 9.5 (P376)
  - Summary of FP Arithmetic

Implementing Floating-Point Operations

- Implementation Methods
  - Software Routines
  - Hardware Circuits
    - Computers will provide machine instructions for floating-point operations.
  - Note
    - In either case, the computer must be able to convert input and output from and to the use's decimal representation of numbers.
- Example of Hardware Implementation
  - 32-bit operands , {A: S_A, E_A', M_A}, {B: S_B, E_B', M_B}


E_A'
E_B'
M_A
M_B
8-bit subtractor
SWAP
M of number with smaller E'
M of number with larger E'
SHIFTER
n bits to right
sign
S_A S_B
$n = |E_A' - E_B'|$
Add / Subtract
Combinational CONTROL network
Add/Sub
Mantissa adder/subtractor
Sign
Magnitude M
E_A'
E_B'
MUX
Leading zeros detector
E'
X
Normalize and round
8-bit subtractor
$E' - X$
E_R'
M_R
$R : S_R$
32-bit result
$R = A + B$

Content of this lecture

- 9.7 Floating-Point Numbers and Operations
  - Fix-point Representation
  - Floating-point Representation
  - Floating-point Numbers in Computers
  - IEEE-754 Standard
  - Summary of FP Representation
  - Arithmetic Operations on FP
  - Problems Considerable in FP Arithmetic
  - Implementing Floating-Point Operations
  - Solved Problems Example 9.5 (P376)
  - Summary of FP Arithmetic

Solved Problems (1)

- Example 9.5

Problem: Consider the following 12-bit floating-point number representation format that is manageable for working through numerical exercises. The first bit is the sign of the number. The next five bits represent an excess-15 exponent for the scale factor, which has an implied base of 2. The last six bits represent the fractional part of the mantissa, which has an implied 1 to the left of the binary point.

Perform Subtract and Multiply operations on the operands

$A =$	0	10001	011011
$B =$	1	01111	101010

which represent the numbers

$$A = 1.011011 \times 2^2$$

and

$$B = -1.101010 \times 2^0$$

Solved Problems (2)

Example 9.5 (ctd.)

Solution: The required operations are performed as follows:

- Subtraction

  According to the Add/Subtract rule in Section 9.7.1, we perform the following four steps:

  1. Shift the mantissa of B to the right by two bit positions, giving 0.01101010.
  2. Set the exponent of the result to 10001.
  3. Subtract the mantissa of B from the mantissa of A by adding mantissas, because B is negative, giving

$$\begin{array}{rcccccccccc} & 1 & . & 0 & 1 & 1 & 0 & 1 & 1 & 0 & 0 \\ + & 0 & . & 0 & 1 & 1 & 0 & 1 & 0 & 1 & 0 \\ \hline & 1 & . & 1 & 1 & 0 & 1 & 0 & 1 & 1 & 0 \end{array}$$

  and set the sign of the result to 0 (positive).

  4. The result is in normalized form, but the fractional part of the mantissa needs to be truncated to six bits. If this is done by rounding, the two bits to be removed represent the tie case, so we round to the nearest even number by adding 1, obtaining a result mantissa of 1.110110. The answer is

$$A - B = \boxed{0} \quad \boxed{10001} \quad \boxed{110110}$$

Solved Problems (3)

Example 9.5 (ctd.)

- Multiplication

  According to the Multiplication rule in Section 9.7.1, we perform the following three steps:

  1. Add the exponents and subtract 15 to obtain 10001 as the exponent of the result.
  2. Multiply mantissas to obtain 10.010110101110 as the mantissa of the result. The sign of the result is set to 1 (negative).
  3. Normalize the resulting mantissa by shifting it to the right by one bit position. Then add 1 to the exponent to obtain 10010 as the exponent of the result. Truncate the mantissa fraction to six bits by rounding to obtain the answer

$A \times B =$

0	10010	001011

Summary of FP Arithmetic

- 知识点Floating-point Arithmetic Operation
  - Addition
  - Subtraction
  - Multiplication
  - Division
- 掌握程度
  - 给定两个浮点数，根据加减运算规则或乘除运算规则计算出结果。

Homework

- P380
  - 9.21 (a) (d)两个小题，只做Add和Subtract运算。
  - 9.22