Computer Organization & Architecture

2-9 Floating-point Number Arithmetic

Wang Guohua

School of Software Engineering

Contents of this lecture

- Alignment
- Add/Subtract Rule
- Multiply Rule
- Divide Rule
- Problems Considerable in FP Arithmetic
  - Guard Bits
  - Truncation
  - Normalization
  - Overflow

Alignment

- First step of addition and subtraction operations in floating-point numbers.
- Alignment
  - Before addition/subtraction, if exponents of two floating-point numbers differ, it is necessary to manipulate the two summands so that the two exponents are equal.
  - Example: Decimal addition $(123 \times 10^0)+(456 \times 10^{-2})$
    - $(123 \times 10^0)+(456 \times 10^{-2}) = (123 \times 10^0)+(4.56 \times 10^0)$
    - $(123 \times 10^0)+(456 \times 10^{-2}) = (12300 \times 10^{-2})+(456 \times 10^{-2})$
  - 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.

Add/Subtract Rule

- For IEEE Single Precision Floating-point Numbers
  - 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.

Multiply Rule

- For IEEE Single Precision Floating-point Numbers
  - 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 Floating-point Numbers
  - 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.

Problems Considerable in FP Arithmetic (1)

- Guard Bits
  - To improve the precision of floating-point computations, guard bits are used.
  - 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.

Problems Considerable in FP Arithmetic (2)

- Guard Bits (ctd.)
  - Example: $X = 1.00\ldots00 \times 2^1$, $Y = 1.11\ldots11 \times 2^0$ (X and Y are all in IEEE single precision format). Calculate $Z = X - Y$.
    - Without guard bits

$$\begin{array}{rl} X = & 1.000\ldots00 \times 2^1 \\ -Y = & 0.111\ldots11 \times 2^1 \\ \hline Z = & 0.000\ldots01 \times 2^1 \end{array} \quad = 1.000\ldots00 \times 2^{-22}$$

    - With guard bits

$$\begin{array}{rl} X = & 1.000\ldots00\ \ \underline{0} \times 2^1 \\ -Y = & 0.111\ldots11\ \ \underline{1} \times 2^1 \\ \hline Z = & 0.000\ldots00\ \ \underline{1} \times 2^1 \end{array} \quad = 1.000\ldots00\ \ \underline{0} \times 2^{-23}$$

Problems Considerable in FP Arithmetic (3)

- Truncation
  - Mantissa is restricted to a specific length (23 bit/single-precision, 52 bit/double-precision)
  - An arithmetic operation may result in a mantissa with a larger precision
  - Before storing a floating-point number, the excessive bits have to be discarded $\Rightarrow$ truncation
  - A truncation method should be unbiased
    - The errors compensate each other
  - Truncation Methods
    - Chopping
    - Von Neumann Rounding
    - Rounding

Problems Considerable in FP Arithmetic (4)

- Chopping
  - Remove the guard bits and make no changes in the retained bits.
  - Easy to implement
  - Biased, since all values are rounded towards a lower mantissa value
  - The error range in chopping is from 0 to almost 1 in the least significant position of the retained bits.
  - Not the optimum method!

Problems Considerable in FP Arithmetic (5)

- Chopping (ctd.)
  - Example: Truncate $0.b_{-1}b_{-2}b_{-3}\,010$ to three bits.
    - $0.b_{-1}b_{-2}b_{-3}\,010 \Rightarrow 0.b_{-1}b_{-2}b_{-3}$
    - 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.

Problems Considerable in FP Arithmetic (6)

- Von Neumann Rounding
  - Distinguishes an *exact* representation and *rounding* towards next odd boundary:
    - 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.
  - Still moderately easy to implement
  - Unbiased, rounding is equally distributed towards positive and negative values
  - The error range is between – 1 and +1 in the LSB position of the retained bits.
  - Better than chopping, higher absolute errors

Problems Considerable in FP Arithmetic (7)

- Von Neumann Rounding (ctd.)
  - Example: Truncate $0.b_{-1}b_{-2}b_{-3}\,000$ – $0.b_{-1}b_{-2}b_{-3}\,111$ to three bits.
    - ① $0.b_{-1}b_{-2}b_{-3}\,000 \Rightarrow 0.b_{-1}b_{-2}b_{-3}$
    - ② $0.b_{-1}b_{-2}b_{-3}\,001$ – $0.b_{-1}b_{-2}b_{-3}\,111 \Rightarrow 0.b_{-1}b_{-2}1$

Problems Considerable in FP Arithmetic (8)

- Rounding
  - Rounding maps a value to its nearest value representable, distinguishing three cases:
    - 1. MSB of the mantissa part to be truncated is zero ⇒ perform chopping
    - 2. MSB of the truncated mantissa part is one, and *any* other bit to be truncated is one ⇒ add one to retained LSB
    - 3. MSB of the truncated mantissa part is one, and *all* other bits to be truncated are zero:
      - ⇒ chop if retained LSB is zero
      - ⇒ add one to retained LSB if one
  - Some effort to implement
  - Unbiased, rounding is equally distributed towards positive and negative values
  - The error range is – ½ to + ½ in the LSB position of the retained bits.

Problems Considerable in FP Arithmetic (9)

- Rounding (ctd.)
  - Example: Truncate $0.b_{-1}b_{-2}b_{-3}\,000$ – $0.b_{-1}b_{-2}b_{-3}\,111$ to three bits.
    - ① $0.b_{-1}b_{-2}b_{-3}\,000$ – $0.b_{-1}b_{-2}b_{-3}\,011 \Rightarrow 0.b_{-1}b_{-2}b_{-3}$
    - ② $0.b_{-1}b_{-2}b_{-3}\,101$ – $0.b_{-1}b_{-2}b_{-3}\,111 \Rightarrow 0.b_{-1}b_{-2}b_{-3}+0.001$
    - ③ $0.b_{-1}b_{-2}b_{-3}\,100$
      - $0.b_{-1}b_{-2}\,0100 \Rightarrow 0.b_{-1}b_{-2}\,0$
      - $0.b_{-1}b_{-2}1100 \Rightarrow 0.b_{-1}b_{-2}\,1 + 0.001$
  - Rounding is the default mode for truncation specified in the IEEE floating-point standard.

Problems Considerable in FP Arithmetic (10)

- Normalization
  - IEEE Single Precision Normalized FP Numbers
    - E≠ 000…0 (8 bits)and E ≠ 111…1(8 bits)
    - E is encoded with bias value
    - The normalized significand is 1.M
  - If a number is not normalized, it can always be put in normalized form by shifting the mantissa and adjusting the exponent.

Problems Considerable in FP Arithmetic (11)

- Normalization (ctd.)
  - Example

	excess-127 exponent	
0	1 0 0 0 1 0 0 0 •	0 0 1 0 1 1 0 . . .

(There is no implicit 1 to the left of the binary point.)

$$\text{Value represented} = +0.0010110\ldots \times 2^{9}$$

(a) Unnormalized value

0	1 0 0 0 0 1 0 1 •	0 1 1 0 . . .

$$\text{Value represented} = +1.0110\ldots \times 2^{6}$$

(b) Normalized version

Figure 9.27 Floating-point normalization in IEEE single-precision format.

Problems Considerable in FP Arithmetic (12)

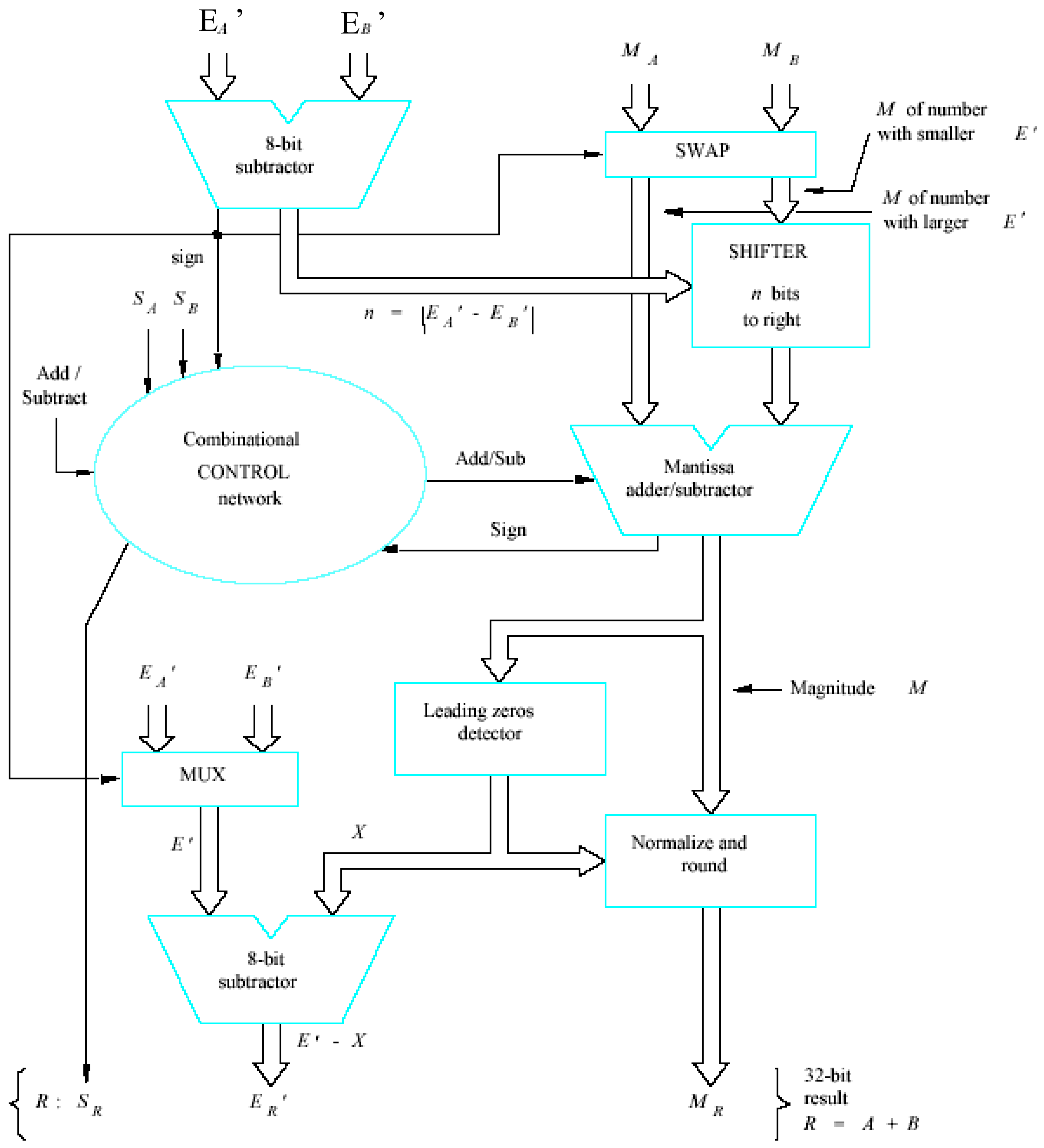
- Overflow
  - As computation proceed, a number that does not fall in the representable range of normal numbers might be generated.
  - Exponent Overflow
    - A positive exponent exceeds the maximum possible exponent value.
    - Example: In IEEE single precision, $e > 127$
    - In some systems, it 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 (13)

- 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.

Implementing Floating-Point Operations

- Implementation Methods
  - Software Routines
  - Hardware Routines
    - 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
Sign
Mantissa adder/subtractor
Magnitude M
Leading zeros detector
E_A'
E_B'
MUX
E'
X
Normalize and round
8-bit subtractor
E' - X
R : S_R
E_R'
M_R
32-bit result
R = A + B

Quiz (1)

1. True or False? An overflow of the mantissa field of a floating-point number means a real case of overflow.

   浮点数的尾数部分溢出并不是真正的溢出，可以通过移位来解决；

   而指数部分溢出才是真正的溢出。

Quiz (2)

2. Consider a reduced 8-bit IEEE floating-point format, with 1 bit for the sign, 3 bits of the exponent and 4 bits for the mantissa. Note that the mantissa is normalized with an implied 1 to the left of the binary point.

   (1) Express A=2.38 and B=1.6 in this floating-point format.

   (2) Write the computation process of A-B and give the result in normalized form. Use **Rounding** method as needed.

Quiz (2)

Solution:

(1) $+2.38=+10.0110000=+1.0011\times 2^1$

0 100 0011

$+1.6=+1.1001100=+1.1010\times 2^0$

0 011 1010

Quiz (2)

Solution:

(2) ①Shift the mantissa of B to the right by one bit position, giving 0.1101<u>0</u>

②Set the exponent of the result to 100.

③Subtract the mantissa of B from the mantissa of A, giving

$$\begin{array}{r} 1.0011\underline{0} \\ -\ \ 0.1101\underline{0} \\ \hline 0.0110\underline{0} \end{array}$$

and set the sign of the result to 0.

④Shift the mantissa to the left by two bit positions. We obtain a result mantissa of 1.1000. So the exponent of the result is 010. The answer is 0 010 1000

Homework

- 9.1
- 9.9 (a) (b)
- 9.20
- 9.21
- 9.22