Chapter 3

Dr. Md Abu Sayeed
EET 340

Binary Addition

Four Basic Rules of Binary Addition:

0 + 0 = 0 sum = 0, carry = 0

0 + 1 = 1 sum = 1, carry = 0

1 + 0 = 0 sum = 1, carry = 0

1 + 1 = 10 sum = 0, carry = 1

For addition, in the computer, it is a bitwise addition. It adds one bit by one bit every time.

Example 1: 110 + 100 = ?

```
        carry
 1

    1   1   0
+   1   0   0
--------------
1   0   1   0
```

```
     6
+    4
------
 1   0
```

Example 2: 111 + 11 = ?

```
            carry
1   1   1

    1   1   1
    +   1   1
--------------
1   0   1   0
```

```
     7
+    3
------
 1   0
```

Binary Subtraction

Example 1: 8 - 3 = ?

```
   1 0 0 0    : 8
 + 1 1 0 1    : 2's complement of 3 (-3)
 ---------
   0 1 0 1    : 5
```

Example 2: 7 - 6 = ?

```
   0 1 1 1    : 7
 + 1 0 1 0    : 2's complement of 6 (-6)
 ---------
   0 0 0 1    : 5
```

Example 1: 101 - 011 = ?

$$\begin{array}{rccc} & \overset{0}{\cancel{1}} & \overset{10}{\cancel{0}} & 1 \\ - & 0 & 1 & 1 \\ \hline & 0 & 1 & 0 \end{array}$$

borrow is required

Example 2: 110 - 101 = ?

$$\begin{array}{rccc} & 1 & \overset{0}{\cancel{1}} & \overset{10}{\cancel{0}} \\ - & 1 & 0 & 1 \\ \hline & 0 & 0 & 1 \end{array}$$

borrow is required

Binary Multiplication

Four Basic Rules of Binary Multiplication:

0 x 0 = 0

0 x 1 = 0

1 x 0 = 0

1 x 1 = 1

- It involves forming partial products, shifting each successive partial product left one place, and then adding all the partial products.
- The computer can only do addition and shift operation. So, for multiplication, it will be converted to simple shift and addition operation.

Example 1: 11 X 01 = ?

```
    1 1
  X 1 0
  ---------
    0 0
  1 1
  ---------
  1 1  0
```

```
    3
  X 2
  -----
    6
```

Example 2: 111 X 101 = ?

```
      1 1 1
    X 1 0 1
  -----------
      1 1 1
    0 0 0
  1 1 1
  -----------
1 0 0 0 1 1
```

```
    7
  X 5
  -----
    35
```

Floating Point Numbers

Scientific notation: A notation that renders numbers with a single digit to the left of the decimal point.
Normalized Number: A number in scientific notation that has no leading 0s is called a normalized number

For example, 0.000000001 is floating point number (decimal).
$1.0_{ten} \times 10^{-9}$ is in normalized scientific notation, but $0.1_{ten} \times 10-8$ and $10.0_{ten} \times 10^{-10}$ are not.

For example, 1001.001 is fractional binary number.
$1.001001_{two} \times 2^{3}$ is in normalized scientific notation

Floating Point Numbers

- Floating-point number (also known as a real number) consists of two parts plus a sign. The mantissa is the part of a floating-point number that represents the magnitude of the number and is between 0 and 1. The exponent is the part of a floating-point number that represents the number of places that the decimal point (or binary point) is to be moved.
- For binary floating-point numbers, the format is defined by ANSI/IEEE Standard 754-1985 in three forms:
  - single-precision
  - double-precision

IEEE 754 Single Precision Format

S	EXPONENT	FRACTION
1 bit	8 bits	23 bits

- S: Sign bit (0 → Nonnegative, 1→ Negative)
- Exponent = Actual Exponent + Bias
  For single precision, Bias = 127
- Fraction : 23 bit fractions from normalized numbers

The value of the floating-point number can be determined by the following expression:

$$(-1)^S \times (1 + \text{Fraction}) \times 2^{(\text{Exponent} - \text{Bias})}$$

Example: Convert the decimal value 2.75 to IEEE-754 single precision format. Write your converted result in hexadecimal format.

① Convert to binary: 0.75 * 2 = 1.5

0.5 * 2 = 1.0

$\rightarrow$ 10.11

② Normalized: $1.011 * 2^1$

③ Calculation of the three components: $(-1)^S * (1 + Fraction) * 2^{(Exponent-Bias)}$

Single Precision:

- S = 0;
- Fraction = 0.011;
- Bias = 127;
- Exponent = 128;

$\underbrace{0}_{S},\ \underbrace{1000,\ 0000,}_{Exponent}\ \underbrace{0110,\ 0000,\ 0000,\ 0000,\ 0000,\ 000_2}_{Fraction} = 0X40300000_{hex}$

Example: Convert the decimal value -4.25 to IEEE-754 single precision format. Write your converted result in hexadecimal format.

① Convert to binary: 0.25 * 2 = 0.5
0.5 * 2 = 1.0
→ 100.01

② Normalized: $1.0001 * 2^2$

③ Calculation of the three components: $(-1)^S * (1 + Fraction) * 2^{(Exponent-Bias)}$

Single Precision:

- S = 1;
- Fraction = 0.0001;
- Bias = 127;
- Exponent = 129;

$$\underbrace{1}_{\text{S}},\ \underbrace{1000,\ 0001}_{\text{Exponent}},\ \underbrace{0001,\ 0000,\ 0000,\ 0000,\ 0000,\ 000}_{\text{Fraction}}{}_2 = 0XC0880000_{hex}$$

Example: Convert each of the following IEEE-754 floating point representation to decimal values.

a. 0x41380000

b. 0xC0E80000

a.

①Convert to binary: 0100, 0001, 0011, 1000, 0000, 0000, 0000, 0000

②Find three components:

- S = 0;
- Fraction = 0.0111;
- Bias = 127;
- Exponent = 130;

③Substitution the three components in the equation: $(-1)^S * (1 + Fraction) * 2^{(Exponent-Bias)}$

$$(-1)^0 * (1 + 0.4375) * 2^{(130-127)} = 11.5$$

b.

①Convert to binary: 1100, 0000, 1110, 1000, 0000, 0000, 0000, 0000

②Find three components:

S = 1;

Fraction = 0.1101;

Bias = 127;

Exponent = 129;

③Substitution the three components in the equation: $(-1)^{S} * (1 + Fraction) * 2^{(Exponent-Bias)}$

$$(-1)^{1} * (1 + 0.8125) * 2^{(129-127)} = -7.25$$

IEEE 754 Double Precision Format

S	EXPONENT	FRACTION
1 bit	11 bits	52 bits

- S: Sign bit (0 → Non-negative, 1→ Negative)
- Exponent = Actual Exponent + Bias
  For double precision, Bias = 1023
- Fraction : 52-bit fractions from normalized number

The value of the floating-point number can be determined by the following expression:

$$(-1)^S \times (1 + \text{Fraction}) \times 2^{(\text{Exponent} - \text{Bias})}$$

Example: Convert the decimal value 2.75 to IEEE-754 double precision format. Write your converted result in hexadecimal format.

① Convert to binary: $0.75 * 2 = 1.5$
$0.5 * 2 = 1.0$ → 10.11

② Normalized: $1.011 * 2^1$

Double Precision:

- S = 0;
- Fraction = 0.011;
- Bias = 1023;
- Exponent = 1024;

S

Exponent

0, 1000, 0000, 000, 0110, 0000, 0000, 0000, 0000, 0000, 0000, 0000, 0000, 0000, 0000, 0000, 0000$_2$ = 0X4006000000000000$_{hex}$

Fraction

Example: Convert the decimal value -4.25 to IEEE-754 double precision format. Write your converted result in hexadecimal format.

① Convert to binary: $0.25 * 2 = 0.5$

$0.5 * 2 = 1.0$ → 100.01

② Normalized: $1.0001 * 2^2$

Double Precision:

S = 1;

Fraction = 0.0001;

Bias = 1023;

Exponent = 1025;

S

Exponent

$1, 1000, 0000, 001, 0001, 0000, 0000, 0000, 0000, 0000, 0000, 0000, 0000, 0000, 0000, 0000, 0000_2 = 0XC011000000000000_{hex}$

Fraction

Example: Convert the following IEEE-754 floating point representation to decimal values.

a. **0X4035000000000000**

Solution:

①Convert to binary: 0100, 0000, 0011, 0101, 0000, 0000, 0000, 0000, 0000, 0000, 0000, 0000, 0000, 0000, 0000, 0000

②Find three components:

- S = 0;
- Fraction = 0.0101;
- Bias = 1023;
- Exponent = 1027;

③Substitution the three components in the equation: $(-1)^S * (1 + Fraction) * 2^{(Exponent-Bias)}$

$$(-1)^0 * (1 + 0.3125) * 2^{(1027-1023)} = 21$$

Floating Point Addition (Decimal)

Consider a 4-digit decimal example: $9.999 \times 10^{1} + 1.610 \times 10^{-1}$

1. Align decimal points:
   - Align decimal point of the number with smaller exponent
   - $9.999 \times 10^{1} + 0.016 \times 10^{1}$

2. Add significands:

   $9.999 + 0.016 = 10.015$

   result $= 10.015 \times 10^{1}$

3. Normalize result: 1.0015×10^{2}

4. Round and renormalize if necessary: 1.002×10^{2}

NB. We assumed that significands can be only 4 bits or digits.

Floating Point Addition (Binary)

1. Align binary points:
   - Align binary point of the number with smaller exponent
2. Add significands:
3. Normalize result:
4. Round and renormalize if necessary:

NB. We assumed that significands can be only 4 bits or digits.

Problem: Perform binary floating-point addition and convert the resulting values to IEEE-754 single and IEEE-754 double precision format. Finally convert them to hexadecimal values.

0.5 + 0.4375

SOLUTION:

Convert 0.5 to Binary:

$0.5 \times 2 = 1.0 \quad \} \quad 0.1$

Normalized: 1.0×2^{-1}

Convert 0.4375 to Binary:

$$\left.\begin{array}{ll} 0.4375 \times 2 & = 0.875 \\ 0.875 \times 2 & = 1.75 \\ 0.75 \times 2 & = 1.5 \\ 0.5 \times 2 & = 1.0 \end{array}\right\} \quad 0.0111$$

Normalized: 1.11×2^{-2}

1. Align Binary Points:

$$0.5 \quad = 1.000 \text{ X } 2^{-1}$$
$$0.4375 = 0.111 \text{ X } 2^{-1}$$

2. Add Significands:

$1.000 + 0.111 = 1.111$

Resulting value = $1.111 \text{ X } 2^{-1}$

3. Normalized Result = $1.111 \text{ X } 2^{-1}$

4. Round and Renormalize: No change

Single Precision Format (from Normalized result):

Sign = 0

Exponent = -1 + 127 $=126_{10}=01111110_2$

Fraction = 0.111

Sign (1)	Exponent (8)	Fraction (23)
0	**01111110**	**111**00000000000000000000

0011 1111 0111 0000 0000 0000 0000 0000 = 0X3F700000

Single Precision Format (from Normalized result):

Sign = 0

Exponent = -1 + 1023 $=1022_{10}=01111111110_2$

Fraction = 0.111

Sign (1)	Exponent (11)	Fraction (52)
0	01111111110	**111**0000000000000………0

0011 1111 1110 1110 0000 0000 0000 0000 0000 0000
0000 0000 0000 0000 0000 0000 = 0X3FEE000000000000

Floating Point Subtraction (Binary)

1. Align binary points:
   - Align binary point of the number with smaller exponent
2. Add significands:
3. Normalize result:
4. Round and renormalize if necessary:

NB. We assumed that significands can be only 4 bits or digits.

Problem: Perform binary floating-point subtraction and convert the resulting values to IEEE-754 single. Finally convert them to hexadecimal values.

0.5 – 0.4375

SOLUTION:

Convert 0.5 to Binary:

$0.5 \times 2 = 1.0$ 0.1

Normalized: 1.0×2^{-1}

Convert 0.4375 to Binary:

$0.4375 \times 2 = 0.875$
$0.875 \times 2 = 1.75$
$0.75 \times 2 = 1.5$
$0.5 \times 2 = 1.0$

0.0111

Normalized:

$-0.4375_{10} = -1.11 \times 2^{-2}$

1. Align Binary Points:

$$0.5 \quad = \quad 1.000 \text{ X } 2^{-1}$$
$$-\ 0.4375 = -\ 0.111 \text{ X } 2^{-1}$$

2. Add Significands:

2's complement of 0.111 = 1.000 +1= 1.001

1.000 + 1.001 = 10.001

Resulting value = 0.001 X 2^{-1}

3. Normalized Result = 1.000 X 2^{-4}

4. Round and Renormalize: No change

Single Precision Format (from Normalized result):

Sign = 0

Exponent = -4 + 127 = $123_{10} = 01111011_2$

Fraction = 0.0

0	01111011	00000000000000000000000
Sign (1)	**Exponent (8)**	**Fraction (23)**

0011 1101 1000 0000 0000 0000 0000 0000 = 0X3D800000

Floating Point Multiplication

1. Add exponents:
2. Multiply significands:
3. Normalize result:
4. Round and renormalize if necessary:
5. Determine the sign of result from signs of operands

NB. We assumed that significands can be only 4 digits of the significands and two digits of the exponents .

Floating Point Multiplication (Decimal)

Consider a 4-digit decimal example: (1.110×10^{10}) X (9.2×10^{-5})

1. Add exponents:
   - e1 =10 e2=-5
   - e = e1 + e2 = 10 – 5 = 5

2. Multiply significands:

   1.110 X 9.2 = 10.212

   result = 10.212×10^{5}

3. Normalize result: 1.0212×10^{6}

4. Round and renormalize if necessary: 1.021×10^{6}

5. Determine the sign= (+) * (+) = +

$$1.021 \times 10^{6}$$

Problem: Perform binary floating-point multiplication and convert the resulting values to IEEE-754 single precision format. Finally convert them to hexadecimal values.

0.5 * -0.4375

SOLUTION:

Convert 0.5 to Binary:

$0.5 \times 2 = 1.0$ → 0.1

Normalized: 1.0×2^{-1}

Convert 0.4375 to Binary:

$$\left.\begin{aligned} 0.4375 \times 2 &= 0.875 \\ 0.875 \times 2 &= 1.75 \\ 0.75 \times 2 &= 1.5 \\ 0.5 \times 2 &= 1.0 \end{aligned}\right\} 0.0111$$

Normalized: 1.11×2^{-2}

1. Add exponents:
   - e1 = -1 e2= -2
   - e = -1 - 2 = -3

2. Multiply significands:

   1.000 X 1.110 = 1.110000

   result = 1.11 $\times$ 2^{-3}

3. Normalize result: 1.11 $\times$ 2^{-3}

4. Round and renormalize if necessary: No change

5. Determine the sign= (+) * (-) = -

Single Precision Format (from Normalized result):

Sign = 1

Exponent = -3 + 127 = 124_{10} = 01111100_2

Fraction = 0.11

1	01111100	11000000000000000000000
Sign (1)	**Exponent (8)**	**Fraction (23)**

1011 1110 0110 0000 0000 0000 0000 0000 = 0XBE600000

Note: The figures, text etc included in slides are borrowed from books, other sources for academic purpose only. The author does not claim any originality.

Source:

1.Computer Organization and Design (ARM Edition) by David A. Patterson