

Binary Arithmetic

2. Binary Arithmetic

2.1 Complement

2.2 Un-signed binary number

2.3 Signed binary number

2.4 Binary Arithmetic

- Addition
- Subtraction
- Multiplication
- Division

2.5 Interpretation of arithmetic result

2.1 Complement

- 2 types of complement for any radix- r number system
 - For a n -digit number a
 - Its $(r-1)$'s complement number: $r^n - 1 - a$
 - Its r 's complement number: $r^n - a$

Note: All systems are implemented with fixed digit (bit) length.

The length of complement is the same as the number itself.

- For decimal number $r = 10$

- 9's complement of a : $b = 10^n - 1 - a$

Examples:

$$a = 43, \quad b = 100 - 1 - 43 = 56 \quad (n = 2)$$

$$a = 7629, \quad b = 10000 - 1 - 7629 = 2370 \quad (n = 4)$$

- 10's complement of a : $c = 10^n - a$

Examples:

$$a = 43, \quad c = 100 - 43 = 57 \quad (n = 2)$$

$$a = 7629, \quad c = 10000 - 7629 = 2371 \quad (n = 4)$$

- For binary number $r = 2$
 - 1's complement of a : $b = 2^n - 1 - a$

Examples:

$$a = 1001$$

$$\begin{aligned} b &= 10000 - 1 - 1001 \\ &= 0110 \quad (n = 4, \text{ same bit length}) \end{aligned}$$

$$a = 1100101$$

$$\begin{aligned} b &= 10000000 - 1 - 1100101 \\ &= 0011010 \quad (n = 7, \text{ same bit length}) \end{aligned}$$

- OR turn “1” to “0” and “0” to “1”

- For binary number $r = 2$
 - 2's complement of a : $c = 2^n - a$
 - OR Its 1's complement plus "1", i.e. $b + 1$

Examples:

$$a = 1001$$

$$c = 0110 + 1$$

$$= 0111 \quad (n = 4, \text{ same bit length})$$

$$a = 1100101$$

$$c = 0011010 + 1$$

$$= 0011011 \quad (n = 7, \text{ same bit length})$$

- A fast way to obtain the 2's complement
 - i. Starting from the LSB,
 - ii. Copy all digits including the first “1” encountered,
 - iii. Then complement the rest of the bits.

$$a = 1001,$$

$$c = 0111$$

$$a = 1100101,$$

$$c = 0011011$$

$$a = 101110010100$$

$$c = 010001101100$$

- Property of r 's complement of a n -digit number a .

10's complement: $c = 10^n - a$

Sum of a and c : $a + c = a + 10^n - a = 10^n$

Examples:

$$a = 43, c = 10^2 - 43 = 57, a + c = 100 \quad (1=10^2, n=2)$$

$$a = 7629, c = 10^4 - 7629 = 2371, a + c = 10000 \quad (1=10^4, n=4)$$

For hardware system, c is also a n -digit number, and so as the result of $a + c$, therefore 1 is discarded from the result.

NOTE: $1 = 10^n$, discard $1 \Rightarrow a + c - 10^n \Rightarrow a + c = 0$

$n = 2, a + c = 00$ (for 2-digit result)

$n = 3, a + c = 000$ (for 3-digit result)

$n = 4, a + c = 0000$ (for 4-digit result)

$\Rightarrow$ "10's of a " = $c \equiv -a$ for n -digit input and output system

For a binary number a :

2's complement: $c = 2^n - a$

Sum of a and c : $a + c = a + 2^n - a = 2^n$

1001,	1100101	101110010100
<u>+ 0111</u>	<u>+ 0011011</u>	<u>+ 010001101100</u>
10000	10000000	10000000000000

NOTE: 1 = 2^n , discard 1 $\Rightarrow a + c - 2^n \Rightarrow a + c = 0$

$n = 2$, $a + c = 00$ (for 2-bit result)

$n = 3$, $a + c = 000$ (for 3-bit result)

$n = 4$, $a + c = 0000$ (for 4-bit result)

$\Rightarrow$ "2's of a " = $c \equiv -a$ **for n -bit input and output system**

Formal proof:

- For a n -bit binary number a ,
it's 2's complement is $b = 2^n - a$
- $a + b = a + 2^n - a = 2^n$
- 2^n is the weight of the $(n+1)^{th}$ bit of the sum.
- If the $(n+1)^{th}$ bit is discarded for a n -digit system,
then $a + b = 0$ and b can be treated as $-a$.

2.2 Un-signed binary number

- For a n -bit number, its value range is:
 - $0, 1, 2, \dots, 2^n - 1$

$$a_{n-1} r^{n-1} + a_{n-2} r^{n-2} + \dots + a_2 r^2 + a_1 r + a_0$$

$$n = 2 \Rightarrow 0, 1, 2, 3$$

$$n = 5 \Rightarrow 0, 1, 2, \dots, 30, 31$$

$$n = 8 \Rightarrow 0, 1, 2, \dots, 254, 255$$

$$n = 10 \Rightarrow 0, 1, 2, \dots, 1022, 1023$$

2.3 Signed number

2.3.1 signed magnitude representation

- The MSB is a sign bit (which does not carry any value)
 - 0 means **positive** “+”
 - 1 means **negative** “-”



Sign-magnitude format

Example:

4-bit system (**+5** = **0101**; **-5** = **1101**)

Last 3 bits used for magnitude

- For a n -bit number, range value is:
 $-(2^{n-1}-1), -2^{n-1}+2, \dots, -2, -1, 0, +1, +2, \dots, +2^{n-1}-2, +2^{n-1}-1$

■ Problems of Sign-Magnitude representation

- Digital circuit cannot differentiate sign bit from number bit.

e.g. compute the sum of +5 and -3 using 4-bit system

$$+5 = 0101$$

$$-3 = 1011$$

Result = 10000 – **WRONG ANSWER!!**

Problem : The sign bit do not carry value, adding two sign bits **DOES NOT** make sense !!!

- Cannot perform arithmetic operation with signed-magnitude representation !!!
- NOT used in integer number system !!!
- Only used in floating-point system $(-1)^S$!!!

2.3.2 Signed number - 2's complement format

- MSB is a sign bit with weight
 - 0 means **positive** “+” with weight of 0
 - 1 means **negative** “−” with weight of 2^{n-1}



Sign-magnitude format

- value: $S \times (-2^{n-1}) + \text{magnitude}$

i.e.,

sign bit = 0: value of $0xx...x = 0 + xx...x$

sign bit = 1: value of $1xx...x = -2^{n-1} + xx...x$

Example:

$$01001 \equiv 0 + 9 = +9$$

$$11110111 \equiv -2^{8-1} + 119 = -9$$

- Range value of 2's complement number

$$-2^{n-1}, -2^{n-1}+2, \dots, -2, -1, 0, +1, +2, \dots, +2^{n-1}-2, +2^{n-1}-1$$

$$-8 \leq x[4] \leq +7$$

$$-128 \leq x[8] \leq +127$$

$$-32768 \leq x[16] \leq +32767$$

- Sign extension - extend the bit length of a 2's complement number
- Repeatedly Duplicate the sign bit for the required bit length, i.e.

$$1001 \Rightarrow 11111001$$

$$111001010 \Rightarrow 11111111111111111001010$$

- Important for having the same bit length to perform arithmetic operation using digital circuit.

■ Different number representation (3-bit example)

<u>Unsigned</u>	<u>(Binary)</u>		<u>2's complement (Binary)</u>	<u>Invert MSB</u>
7	111	3	011	111
6	110	2	010	110
5	101	1	001	101
4	100	0	000	100
3	011	-1	111	011
2	010	-2	110	010
1	001	-3	101	001
0	000	-4	100	000

- Complementing the MSB of a 2's complement no turns it to unsigned representation, or vice versa.

■ 2's complement of unsigned number (3-bit example)

<u>Unsigned</u>	<u>Binary</u>	<u>2's complement (Binary)</u>	<u>Value</u>	<u>Sum</u>
0	000	000	0	0
1	001	111	7	0
2	010	110	6	0
3	011	101	5	0
4	100	100	4	0
5	101	011	3	0
6	110	010	2	0
7	111	001	1	0
				3-bit result

■ 2's complement of signed number (3-bit example)

<u>signed</u>	<u>(Binary)</u>	<u>2's complement (Binary)</u>	<u>Value</u>	<u>Sum</u>
3	011	101	-3	0
2	010	110	-2	0
1	001	111	-1	0
0	000	000	0	0
-1	111	001	1	0
-2	110	010	2	0
-3	101	011	3	0
-4	100	100	-4	0

3-bit

result

Signed number is represented in 2's complement form

Signed number also has its 2's complement

2.4 Binary Arithmetic

2.4.1 Binary number addition - unsigned number

$$9 + 5 = 14$$

$$\begin{array}{r} 1\ 0\ 0\ 1 \\ +\ 0\ 1\ 0\ 1 \\ \hline 1\ 1\ 1\ 0 \end{array}$$

$$12 + 7 = 19$$

$$\begin{array}{r} 1\ 1\ 0\ 0 \\ +\ 0\ 1\ 1\ 1 \\ \hline 1\ 0\ 0\ 1\ 1 \end{array}$$

carry & sum

- For a n -bit number, its value range is:
 - $0, 1, 2, \dots, 2^{n-1} - 1, \Rightarrow$ 4-bit : $0, 1, 2, \dots, 15$
- If result exceed n -bit, carry will be generated.
- For n -bit system, result is also interpreted with n -bit. Hence the result is wrong if carry is produced.

2.4.2 Binary number addition - 2's complement number

$$3 + 4 = 7$$

$$\begin{array}{r} 0011 \\ + 0100 \\ \hline 0111 \end{array}$$

$$-7 + 3 = -4$$

$$\begin{array}{r} 1001 \\ + 0011 \\ \hline 1100 \end{array}$$

$$\begin{aligned} 1 &= -8, 100 = 4 \\ -8 + 4 &= -4 \end{aligned}$$

$$-3 + (-4) = -7$$

$$\begin{array}{r} 1101 \\ + 1100 \\ \hline 11001 \end{array}$$

$$\begin{aligned} 1 &= -8, 001 = 1 \\ -8 + 1 &= -7 \end{aligned}$$

Discard the carry for 2's complement number addition.

2.4.3 Overflow - only for 2's complement number

- If the sum exceeds the bit length value range, overflow occurs.
- (+ve no) + (-ve no) $\Rightarrow$ overflow never occur
- (+ve no) + (+ve no) **or** (-ve no) + (-ve no) $\Rightarrow$ overflow may occur

same sign

$$\begin{array}{r}
 \text{00110} \\
 + \text{00110} \\
 \hline
 \text{01100}
 \end{array}
 \quad
 \begin{array}{r}
 + 6 \\
 + 6 \\
 \hline
 +12
 \end{array}$$

same sign

$$\begin{array}{r}
 \text{11010} \\
 + \text{11010} \\
 \hline
 \text{110100}
 \end{array}
 \quad
 \begin{array}{r}
 - 6 \\
 - 6 \\
 \hline
 -12
 \end{array}$$

overflow does not occur

diff. sign

$$\begin{array}{r}
 \text{00110} \\
 + \text{01101} \\
 \hline
 \text{010011}
 \end{array}
 \quad
 \begin{array}{r}
 + 6 \\
 +13 \\
 \hline
 +19
 \end{array}$$

diff. sign

$$\begin{array}{r}
 \text{11010} \\
 + \text{10011} \\
 \hline
 \text{101101}
 \end{array}
 \quad
 \begin{array}{r}
 - 6 \\
 -13 \\
 \hline
 -19
 \end{array}$$

overflow occurs

2.4.4 Binary number subtraction - Unsigned number

$$\begin{array}{r}
 9 - 5 = 4 \\
 1\ 0\ 0\ 1 \\
 -\ 0\ 1\ 0\ 1 \\
 \hline
 0\ 1\ 0\ 0
 \end{array}$$

$$\begin{array}{r}
 6 - 11 = -5 \\
 0\ 1\ 1\ 0 \\
 -\ 1\ 0\ 1\ 1 \\
 \hline
 1\ 0\ 0\ 1\ 1
 \end{array}$$

Wrong result (borrower) !!

If the result is negative, use 2's complement to help

$$\begin{array}{r}
 6 \\
 -11 \\
 \hline
 -5
 \end{array}
 \quad
 \begin{array}{r}
 0\ 1\ 1\ 0 \\
 +\ 0\ 1\ 0\ 1 \\
 \hline
 1\ 0\ 1\ 1
 \end{array}
 \leftarrow \text{2's complement of } 1011$$

- Answer is the negative of the 2's complement of the sum
i.e. 2's complement of **1011** $\Rightarrow$ Answer = **-0101** = **-5**

2.4.5 Binary number subtraction - 2's complement number

- Addition by means of using the 2's complement the subtrahend.

$$\begin{array}{r}
 2 - 6 = -4 \\
 0010 \\
 - 0110 \\
 \hline
 1100
 \end{array}$$

$$\begin{array}{r}
 2 + (-6) = -4 \\
 0010 \\
 + 1010 \leftarrow \text{2's complement of } 0110 \\
 \hline
 1100
 \end{array}$$

$$\begin{array}{r}
 -6 - 5 = -11 \\
 1010 \\
 - 0101 \\
 \hline
 10101
 \end{array}$$

$$\begin{array}{r}
 -6 + (-5) = -11 \\
 1010 \\
 + 1011 \leftarrow \text{2's complement of } 0110 \\
 \hline
 10101
 \end{array}$$

Exceed the range value !!!

Result overflow !!!

2.4.6 BCD number addition

- If the sum of two BCD digit addition greater than 9, add 0110 (6) to the digit sum.

$$\begin{array}{r}
 0101 \quad 0110 \\
 + 0011 \quad 0111 \\
 \hline
 1000 \quad 1101 \\
 + 0000 \quad 0110 \\
 \hline
 1001 \quad 0011 \\
 9 \qquad 3
 \end{array}$$

$$\begin{array}{r}
 56 \\
 + 37 \\
 \hline
 93
 \end{array}$$

2.4.7 BCD number subtraction

- If the difference of two BCD digit subtraction greater than 9, subtract 0110 (6) from the digit difference.

$$\begin{array}{r}
 0110 \quad 0101 \\
 - 0001 \quad 1001 \\
 \hline
 0100 \quad 1100 \\
 - 0000 \quad 0110 \\
 \hline
 0100 \quad 0110 \\
 \text{4} \qquad \qquad \text{6}
 \end{array}$$

$$\begin{array}{r}
 65 \\
 - 19 \\
 \hline
 46
 \end{array}$$

2.4.8 Binary multiplication

- Two one bit multiplication
- Two n-bit multiplication

A	B	$A \times B$
0	0	0
0	1	0
1	0	0
1	1	1

$$\begin{array}{r}
 1\ 1\ 1\ 0 \\
 \times 1\ 0\ 1\ 1 \\
 \hline
 1\ 1\ 1\ 0 \\
 1\ 1\ 1\ 0 \\
 0\ 0\ 0\ 0 \\
 + 1\ 1\ 1\ 1\ 0 \\
 \hline
 1\ 0\ 0\ 1\ 1\ 0\ 1\ 0
 \end{array}$$

Results is in 2n-bit !!

$$\begin{array}{r}
 \overline{101010} \quad \leftarrow \text{quotient} \\
 1101 \overline{)1000101001} \\
 \underline{1101} \\
 10001 \\
 \underline{1101} \\
 10000 \\
 \underline{1101} \\
 10000 \\
 \underline{1101} \\
 111 \quad \leftarrow \text{remainder}
 \end{array}$$

Fractional
part depends
on precision
requirement.

2.4.10 Floating-point addition

Example:

- Find the sum of 12_{10} and 1.25_{10} using the 32-bit floating-point format.

$$12_{10} = 1.100 \times 2^3, \quad 1.25_{10} = 1.01 \times 2^0 = 0.00101 \times 2^3.$$

0	10000010	100000000000000000000000000000
+	0	10000010
		001010000000000000000000000000
0	10000010	101010000000000000000000000000

$$\text{Ans} = 1.10101 \times 2^3 = 13.25$$

The exponents for the two number must be equal for operation.

2.5 Interpretation of arithmetic result

- For a **n-bit** hardware system, extra bit may be produced
- Addition/subtraction may generate carry/borrow bit
- Hardware cannot differentiate signed or unsigned numbers
- Condition flags will be generated from the ALU (Arithmetic and Logic Unit) of a CPU (Central Processing Unit) to help programmer to make decision
- C: carry/borrow (result is larger than unsigned **n-bit** number)
- Z: the **n-bit** result is zero
- N: the **n-bit** result is negative
- V: overflow (result is larger than signed **n-bit** number)

For unsigned number: C, Z

For signed number: Z, N, V