School of Engineering and Computer Science

Numerical Applied Mathematics

(Arithmetic Operations of Unsigned Binary Numbers)

Kamel ATTAR
attar.kamel@gmail.com

Week **#5** ✦ 30/NOV/2022✦ Week **#6 − 7** ✦2-7/DEC/2022 ✦

- The arithmetic operations of binary numbers, namely, **addition**, **subtraction**, **multiplication** and **division** of binary numbers are almost similar to those of decimal system.
- Addition, subtraction, multiplication and division of binary numbers can be made by following the usual rules of arithmetic.

Just like the decimal system, the following basic laws hold good in binary system:

1. **Unique Existence Law:** The sum and product of any two numbers exist uniquely.
2. **Neutral element:** **0** is the identity element for **'+'** and **1** is the identity element for **'×'**

for addition $\mathbf{0} + \boldsymbol{a} = \boldsymbol{a} + \mathbf{0} = \boldsymbol{a}$ and for multiplication $\mathbf{1} \times \boldsymbol{a} = \boldsymbol{a} \times \mathbf{1} = \boldsymbol{a}$.

3. **Associative Law:** Addition and multiplication of binary numbers are associative.

for addition $(\boldsymbol{a} + \boldsymbol{b}) + \boldsymbol{c} = \boldsymbol{a} + (\boldsymbol{b} + \boldsymbol{c})$ and for multiplication $(\boldsymbol{a} \times \boldsymbol{b}) \times \boldsymbol{c} = \boldsymbol{a} \times (\boldsymbol{b} \times \boldsymbol{d})$.

4. **Commutative Law:** Addition and multiplication of binary numbers are commutative.

for addition $\boldsymbol{a} + \boldsymbol{b} = \boldsymbol{b} + \boldsymbol{a}$ and for multiplication $\boldsymbol{a} \times \boldsymbol{b} = \boldsymbol{b} \times \boldsymbol{a}$.

5. **Distributive Law:** Multiplication of binary numbers is distributive over two terms in addition.

$\boldsymbol{a} \times (\boldsymbol{b} + \boldsymbol{c}) = \boldsymbol{a} \times \boldsymbol{b} + \boldsymbol{a} \times \boldsymbol{c}$.

★ Binary Addition ★

Binary addition is preformed in the same manner as decimal addition.

- But the main difference between these two is, binary number system uses two digits like **0** and **1** of base **2** whereas the decimal number system uses digits from **0** to **9** and the base of this is **10**.
- "Carry-overs" of binary addition are performed in the same manner as in decimal addition.

A	B	$A + B$	Carry-over
0	0	0 + 0 = 0	0
0	1	0 + 1 = 1	0
1	0	1 + 0 = 1	0
1	1	1 + 1 = 0	1
1 + 1 + 1 = 1			1

Example

	Carry					
	21	1	0	1	0	1
+	**27**	1	1	0	1	1
	Result					

$\Longrightarrow$

★ Binary Addition ★

Binary addition is preformed in the same manner as decimal addition.

- But the main difference between these two is, binary number system uses two digits like **0** and **1** of base **2** whereas the decimal number system uses digits from **0** to **9** and the base of this is **10**.
- "Carry-overs" of binary addition are performed in the same manner as in decimal addition.

A	B	$A + B$	Carry-over
0	0	0 + 0 = 0	0
0	1	0 + 1 = 1	0
1	0	1 + 0 = 1	0
1	1	1 + 1 = 0	1
1 + 1 + 1 = 1			1

Example

Carry						
21		1	0	1	0	1
+ 27		1	1	0	1	1
Result						

$\Longrightarrow$

Carry		1	1	1	1	
21		1	0	1	0	1
+ 27		1	1	0	1	1
48	1	1	0	0	0	0

★ Binary Addition ★

✎ **Exercise 1.** Add the following binary numbers:

a. **10101** and **11011**.
b. **110**, **1010**, and **1001**.
c. **111**, **1111**, and **111**.

✎ **Exercise 2.** Here are **3** binary numbers: **1110101** , **1011110** , **1010011** (Working in binary).

a. add together the two smaller numbers,
b. add together the two larger numbers,
c. take the smallest number away from the largest number,
d. add together all three numbers.

✎ **Exercise 3.** Add the following binary numbers:

a. **10101.101** and **1101.011**.
b. **111.0111** and **10011.001**.

Binary Subtraction

First Method : Straight Forward subtraction

Example

	Borrow				
		1	2	5	1
−			2	7	1
	Result				

$\Longrightarrow$

Binary Subtraction

First Method : Straight Forward subtraction

Example

	Borrow				
−		1	2	5	1
			2	7	1
	Result				

$\Longrightarrow$

	Borrow		11	15	
−		$\cancel{1}$0	$\cancel{2}$1	5	1
			2	7	1
	Result	0	9	8	0

Thus **1251** − **271** = **980**

- We can not directly subtract **7** from **5** in the first column as **7** is greater than **5**, so we have to borrow a **10**, the base number, from the next column and add it to the minuend to produce **15** minus **7**.
- This "borrowed" **10** is then return back to the subtrahend of the next column once the difference is found. Simple school math's, borrow a **10** if needed, find the difference and return the borrow.

Binary Subtraction

First Method : Straight Forward subtraction

- The subtraction of **one binary number** from another is exactly the same idea as that for subtracting two decimal numbers, with the difference that **10** represents **2**.

- When **1** is subtracted from **0**, it is necessary to borrow **1** from the next higher order bit and that bit is reduced by **1** (or **1** is added to the next bit of subtrahend) and the remainder is **1**.

A	B	$A - B$	Borrow
0	**0**	**0 − 0 = 0**	**0**
1	**0**	**1 − 0 = 1**	**0**
1	**1**	**1 − 1 = 0**	**0**
0	**1**	**10 − 1 = 1**	**1**

Example

	Borrow				
−	**9**	**1**	**0**	**0**	**1**
	5		**1**	**0**	**1**
	Result				

$\Longrightarrow$

Binary Subtraction

First Method : Straight Forward subtraction

- The subtraction of **one binary number** from another is exactly the same idea as that for subtracting two decimal numbers, with the difference that **10** represents **2**.

- When **1** is subtracted from **0**, it is necessary to borrow **1** from the next higher order bit and that bit is reduced by **1** (or **1** is added to the next bit of subtrahend) and the remainder is **1**.

A	B	$A - B$	Borrow
0	**0**	**0 − 0 = 0**	**0**
1	**0**	**1 − 0 = 1**	**0**
1	**1**	**1 − 1 = 0**	**0**
0	**1**	**10 − 1 = 1**	**1**

Example

	Borrow				
−	**9**	**1**	**0**	**0**	**1**
	5		**1**	**0**	**1**
	Result				

$\Longrightarrow$

	Borrow		**10**		
−	**9**	$\cancel{1}$**0**	**0**	**0**	**1**
	5		**1**	**0**	**1**
	4	**0**	**1**	**0**	**0**

Binary Subtraction

First Method : Straight Forward subtraction

✎ **Exercise 4.** Subtract the following unsigned binary numbers:

a. **101** from **1100**.

b. **101** from **1000**.

c. **101011** from **1000111**.

d. **10101.101** from **1011.11**

✎ **Exercise 5.** Solve the following equations, where all numbers, including x, are binary:

a. $x - 1101 = 11011$

b. $x + 1110 = 10001$

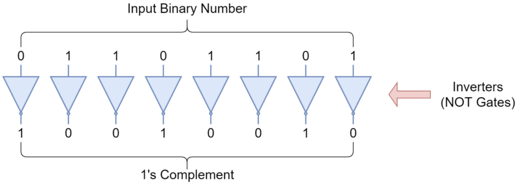
c. $x + 111 = 11110$

d. $x - 1001 = 11101$

Binary Subtraction

Second Method: One's Complement

- First, confirm that the digits in the subtrahend and minuends should be equal.
- A **1**'s complement of a number can be achieved by complementing each digit of the number like zero's to ones and ones to zeros.



- Finally, we add this with the minuend.
  - If the result of addition has a carry over then it is dropped and an **1** is added in the last bit.
  - If there is no carry over, then **1**'s complement of the result of addition is obtained to get the final result and it is negative.

Binary Subtraction

Second Method: one's Complement(Case-**1**: When Carry bit **1** Minuend > Subtrahend)

Example (Case-1: When Carry bit 1)

Subtract **100101** from **110101** by using **1**'s complement.

Solution

Binary	1's complement
100101	**011010**

	Carry						
	53	1	1	0	1	0	1
−	**37**	1	0	0	1	0	1
	16						

$\Longleftrightarrow$

	Carry	**1**	**1**					
			1	1	0	1	0	1
+			0	1	1	0	1	0
	Result		0	0	1	1	1	1

The result of addition has a carry over then it is dropped and an **1** is added in the last bit. Thus the required difference is **001111 + 1 = 10000 = 16**.

Binary Subtraction

Second Method: one's Complement (Case-**2**: When no Carry bit Minuend < Subtrahend)

Example (Case-**2**: When no Carry bit)

Subtract **111001** from **101011** by using **1**'s complement.

Solution

Binary	1's complement
111001	**000110**

	Carry						
	43	**1**	**0**	**1**	**0**	**1**	**1**
−	**57**	**1**	**1**	**1**	**0**	**0**	**1**
	−14						

$\Longleftrightarrow$

	Carry		**1**	**1**	**1**		
		1	**0**	**1**	**0**	**1**	**1**
+		**0**	**0**	**0**	**1**	**1**	**0**
	Result	**1**	**1**	**0**	**0**	**0**	**1**
	1's	**0**	**0**	**1**	**1**	**1**	**0**

There is no carry over, then **1**'s complement of the result of addition is obtained to get the final result and it is negative. Thus the difference is $-(110001)^{1s} = -001110 = -14$.

1. Binary Addition
2. Binary Subtraction
   - First Method : Straight Forward subtraction
   - Second Method: One's Complement
   - Third Method: Two's Complement
     - Case-**1**: When Carry bit **1**
     - Case-**1**: When Carry bit **1** Minuend > Subtrahend
     - Case-**2**: When no Carry bit Minuend < Subtrahend
   - Differences between 1 's complement and 2's complemen
3. Binary Multiplication
4. Binary Division

★ Binary Subtraction ★

Third Method: Two's Complement

With the help of subtraction by **2**'s complement method we can easily subtract two binary numbers.

- First, confirm that the digits in the subtrahend and minuends should be equal.
- A **2**'s complement of a number can be achieved by complementing each digit of the number like zero's to ones and ones to zeros and add one.

$$\text{Binary} \Longrightarrow \underbrace{\mathbf{1}\text{'s complement}}_{\substack{\mathbf{1} \longrightarrow \mathbf{0} \\ \mathbf{0} \longrightarrow \mathbf{1}}} \Longrightarrow \underbrace{\mathbf{2}\text{'s complement}}_{\mathbf{1}\text{'s complement}+\mathbf{1}}.$$

- Finaly, add the subtrahend to the minuend.
  - If the final carry over of the sum is **1**, it is dropped and the result is positive.
  - If there is no carry over, the two's complement of the sum will be the result and it is negative.

Binary Subtraction

Third Method: Two's Complement (Case-1: When Carry bit 1 Minuend > Subtrahend)

Example (Case-1: When Carry bit 1 Minuend > Subtrahend)

Subtract **11011** from **1101101** by using **2**'s complement.

Solution

11011 = 0011011 $\Longrightarrow$

Binary	1's complement	2's complement
0011011	**1100100**	**1100101**

	Carry							
−	**109**	1	1	0	1	1	0	1
	27			1	1	0	1	1
	82							

$\Longleftrightarrow$

	Carry	1	1		1	1		1	
+			1	1	0	1	1	0	1
			1	1	0	0	1	0	1
	Result		1	0	1	0	0	1	0

After dropping the carry over we get the result of subtraction to be **+1010010 = 82**.

Binary Subtraction

Third Method: Two's Complement (Case-**1**: When Carry bit **1** Minuend < Subtrahend)

Example (Case-2: When no Carry bit Minuend < Subtrahend)

Subtract **11010** from **10110** by using **2**'s complement.

Solution

Binary	1's complement	2's complement
11010	**00101**	**00110**

	Carry					
−	**22**	**1**	**0**	**1**	**1**	**0**
	26	**1**	**1**	**0**	**1**	**0**
	−4					

$\Longleftrightarrow$

	Carry		**1**	**1**		
+		**1**	**0**	**1**	**1**	**0**
		0	**0**	**1**	**1**	**0**
	Result	**1**	**1**	**1**	**0**	**0**

As there is no carry over, the result of subtraction is negative and is obtained by writing the **2**'s complement of **11100** i.e.($(\mathbf{11100})^{2's} = \mathbf{00011} + \mathbf{1} = \mathbf{00100}$). Hence the difference is $-\mathbf{100} = -\mathbf{4}$.

Binary Subtraction

Differences between 1 's complement and 2's complement

1's complement	2's complement
end-around-carry-bit addition occurs in 1's complement arithmetic operations. It added to the LSB of result.	end-around-carry-bit addition does not occur in 2's complement arithmetic operations. It is ignored.
1's complement arithmetic operations are not easier than 2's complement because of addition of end-around-carry-bit.	2's complement arithmetic operations are much easier than 1's complement because of there is no addition of end-around-carry-bit.

✎ **Exercise 6.** Find the answer of the following operations (using three different methods):

a. $A = 1110 - 1101$ b. $B = 1010 - 1100$ c. $C = -1010 - 101$.

★ Binary Multiplication ★

- The procedure for binary multiplication is similar to that in decimal system.
- The rules of binary multiplication are given by the following table:

×	1	0
1	**1**	**0**
0	**0**	**0**

- As in decimal system, the multiplication of binary numbers is carried out by multiplying the multiplicand by one bit of the multiplier at a time and the result of the partial product for each bit is placed in such a manner that the LSB is under the corresponding multiplier bit.
- Finally the partial products are added to get the complete product. The placement of the binary point in the product of two binary numbers having fractional representation is determined in the same way as in the product of decimal numbers with fractional representation. The total number of places after the binary point in the multiplicand and the multiplier is counted.

Example

Multiply **11010110** by **101** .

Solution

$$
\begin{array}{cccccccccccc}
 & & & 1 & 1 & 0 & 1 & 0 & 1 & 1 & 0 \\
\times & & & & & & & & 1 & 0 & 1 \\
\hline
 & & & 1 & 1 & 0 & 1 & 0 & 1 & 1 & 0 \\
 & & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & \color{red}{0} \\
+ & 1 & 1 & 0 & 1 & 0 & 1 & 1 & 0 & \color{red}{0} & \color{red}{0} \\
\hline
1 & 0 & 0 & 0 & 0 & 1 & 0 & 1 & 1 & 1 & 0
\end{array}
$$

Example

Multiply **10111** by **1101** .

Solution

$$
\begin{array}{cccccccccc}
 & & & & & 1 & 0 & 1 & 1 & 1 \\
\times & & & & & & 1 & 1 & 0 & 1 \\
\hline \hline
 & & & & & 1 & 0 & 1 & 1 & 1 \\
+ & & & & 0 & 0 & 0 & 0 & 0 & \color{red}{0} \\
 & & & 1 & 0 & 1 & 1 & 1 & \color{red}{0} & \color{red}{0} \\
+ & & 1 & 0 & 1 & 1 & 1 & \color{red}{0} & \color{red}{0} & \color{red}{0} \\
\hline \hline
 & & 0 & 0 & 0 & 1 & 0 & 1 & 1 & 1 \\
+ & 1 & 0 & 0 & 0 & 1 & 0 & 1 & 0 & 0 \\
\hline \hline
 & \mathbf{1} & \mathbf{0} & \mathbf{0} & \mathbf{1} & \mathbf{0} & \mathbf{1} & \mathbf{0} & \mathbf{1} & \mathbf{1}
\end{array}
$$

✎ **Exercise 7.** Calculate the binary numbers

a. **101(110 + 1101)**
b. **1101(1111 − 110)**
c. **111(1000 − 101)**
d. **1011(10001 − 1010)**

✎ **Exercise 8.** Here are **3** binary numbers:

11011 **11100** **10011**

Working in binary

a. multiply the two larger numbers
b. multiply the two smaller numbers.

✎ **Exercise 9.**

a. Multiply the base **10** numbers **45** and **33**.
b. Convert your answer to a binary number.
c. Convert **45** and **33** to binary numbers.
d. Multiply the binary numbers obtained in part (c) and compare this answer with your answer to part (b).

★ Binary Multiplication ★

Multiplying Binary Fractions

- Align both rows by the least significant bit and multiply the same way as in decimal multiplication.
- When multiplying, binary fractions do not need to be lined up by the radix point.
- The final position of the radix point is the sum of the number of radix point places from both factors.
- As for adding, we pad with **0**s to help avoid alignment mistakes.
- We could add two rows at a time, but in this case, there is no need. A row consisting of all **0**s can be ignored, so we add the top and bottom rows in one step. Use binary addition

		1	0	1	·	1
×			1	·	0	1
			1	0	1	1
		0	0	0	0	0
+	1	0	1	1	0	0
	1	1	0	1	1	1

- The top factor has a radix point at one place from the right, and the bottom factor has a radix point three places from the right. Then the radix point will be placed three places from the least significant bit

1	1	0	·	1	1	1

★ Binary Multiplication ★

Multiplying Binary Fractions

Example

Evaluate $\mathbf{10 \cdot 10} \times \mathbf{1 \cdot 01}$

Solution

★ Binary Multiplication ★

Multiplying Binary Fractions

Example

Evaluate $\mathbf{10 \cdot 10} \times \mathbf{1 \cdot 01}$

Solution

$$
\begin{array}{rcccccc}
 & & 1 & 0 & \cdot & 1 & 0 \\
\times & & & 1 & \cdot & 0 & 1 \\
\hline\hline
 & & & 1 & 0 & 1 & 0 \\
 & & 0 & 0 & 0 & 0 & \color{red}{0} \\
+ & 1 & 0 & 1 & 0 & \color{red}{0} & \color{red}{0} \\
\hline\hline
1 & 1 & 0 & 0 & 1 & 0 &
\end{array}
\Longrightarrow \boxed{1}\boxed{1}\boxed{\cdot}\boxed{0}\boxed{0}\boxed{1}\boxed{0}
$$

✎ **Exercise 11.** Evaluate a. $\mathbf{1.1} \times \mathbf{1.1}$ b. $\mathbf{0.001} \times \mathbf{0.0001}$

★ Binary Division ★

Non fractional Division

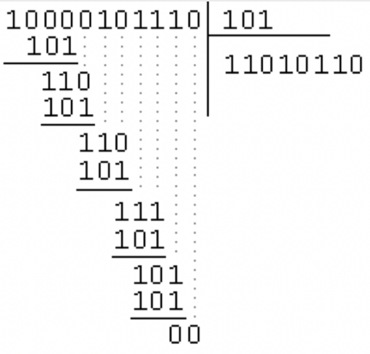
The method followed in binary division is also similar to that adopted in decimal system. However, in the case of binary numbers, the operation is simpler because the quotient can have either 1 or 0 depending upon the divisor.

÷	1	0
1	1	Meaning less
0	0	Meaning less

$$
\begin{array}{r|l}
\;1\;1\;0\;0\;1 & 101 \\ \cline{2-2}
-\;1\;0\;1\downarrow\;\; & 101 \\ \cline{1-1}
0\;1\;0\;\;\; & \\
-\;0\;0\;0\;\downarrow & \\ \cline{1-1}
1\;0\;1 & \\
-\;1\;0\;1 & \\ \cline{1-1}
0\;0\;0 &
\end{array}
$$

$\Longrightarrow$ **11001 $\div$ 101 = 101 .**

Example


10000101110
101
101
11010110
110
101
110
101
111
101
101
101
00

★ Binary Division ★

Fractional DIvision

- Like decimal division, we must remove the radix point from the divisor. The number of fractional places in the divisor tells us how many places to move the radix point (Include the trailing zeros in the fractional divisor when counting places).
- For example, if we have $\frac{10.11}{0.010}$ we move the radix point three places to the right since the divisor has three fractional places, we get $\frac{10.11}{0.010} = \frac{10110}{010}$.
- When we do this, we must also move the radix point the same number of places to the right in the dividend (three places). Append 0 bits as needed to fill the new places.
- We can discard leading zeros from the divisor, so the result looks like this $\frac{10110}{010}$
- Now, we are ready to divide. Division is the same as before. Take it two bits at a time since the divisor is two bits in length.

$$
\begin{array}{rccccc|c}
 & 1 & 0 & 1 & 1 & 0 & 10 \\
\cline{7-7}
- & 1 & 0 & & & & 1011 \\
\cline{2-6}
 & & 0 & 1 & 1 & & \\
- & & 0 & 1 & 0 & & \\
\cline{2-6}
 & & 0 & 0 & 1 & 0 & \\
- & & & & 1 & 0 & \\
\cline{2-6}
 & & & & 0 & 0 & \\
\end{array}
$$

$$\frac{10.11}{0.010} = 1011$$

1. Binary Addition
2. Binary Subtraction
3. Binary Multiplication
4. Binary Division
   - Non fractional Division
   - Fractional DIvision
   - Irrational DIvision

★ Binary Division ★

Irrational DIvision

- Now, let's try an irrational number. We will need to divide into fractional values and choose a location to terminate the quotient because it results in a repetend — a number pattern that repeats indefinitely.
- Divide **1010** by **11**, it is like we are dividing **10** by **3** in decimal number. This produces a quotient of **3.3333333** $\cdots$ in decimal. The fractional **3** continues forever with each additional decimal place producing a quotient more precise, but it will never terminate in even division.
- The same happens in binary. We can divide **1010** by **11** and add fractional binary places until the quotient is so close to even division that it is barely noticeable in real-world problems, but it will never divide evenly.

★ Binary Division ★

Irrational DIvision

$$
\begin{array}{c|cccccccc|l}
 & 1 & 0 & 1 & 0 & .0 & 0 & 0 & 0 & 11 \\
- & & 1 & 1 & & & & & & \overline{11.0101} \\
\hline
 & & 1 & 0 & 0 & & & & & \\
- & & & 1 & 1 & & & & & \\
\hline
 & & & 0 & 1 & 0 & 0 & & & \\
- & & & & & 1 & 1 & & & \\
\hline
 & & & & & & 1 & 0 & 0 & \\
- & & & & & & & 1 & 1 & \\
\hline
 & & & & & & & & 1 &
\end{array}
$$

$$\Longrightarrow \quad \frac{1010}{11} = 11.\overline{01}$$

✎ **Exercise 12.** Evaluate a. $1.1 \div 10$ b. $11 \div 1.1$ c. $111010 \div 101$

Thank you! Questions?