2. Number System

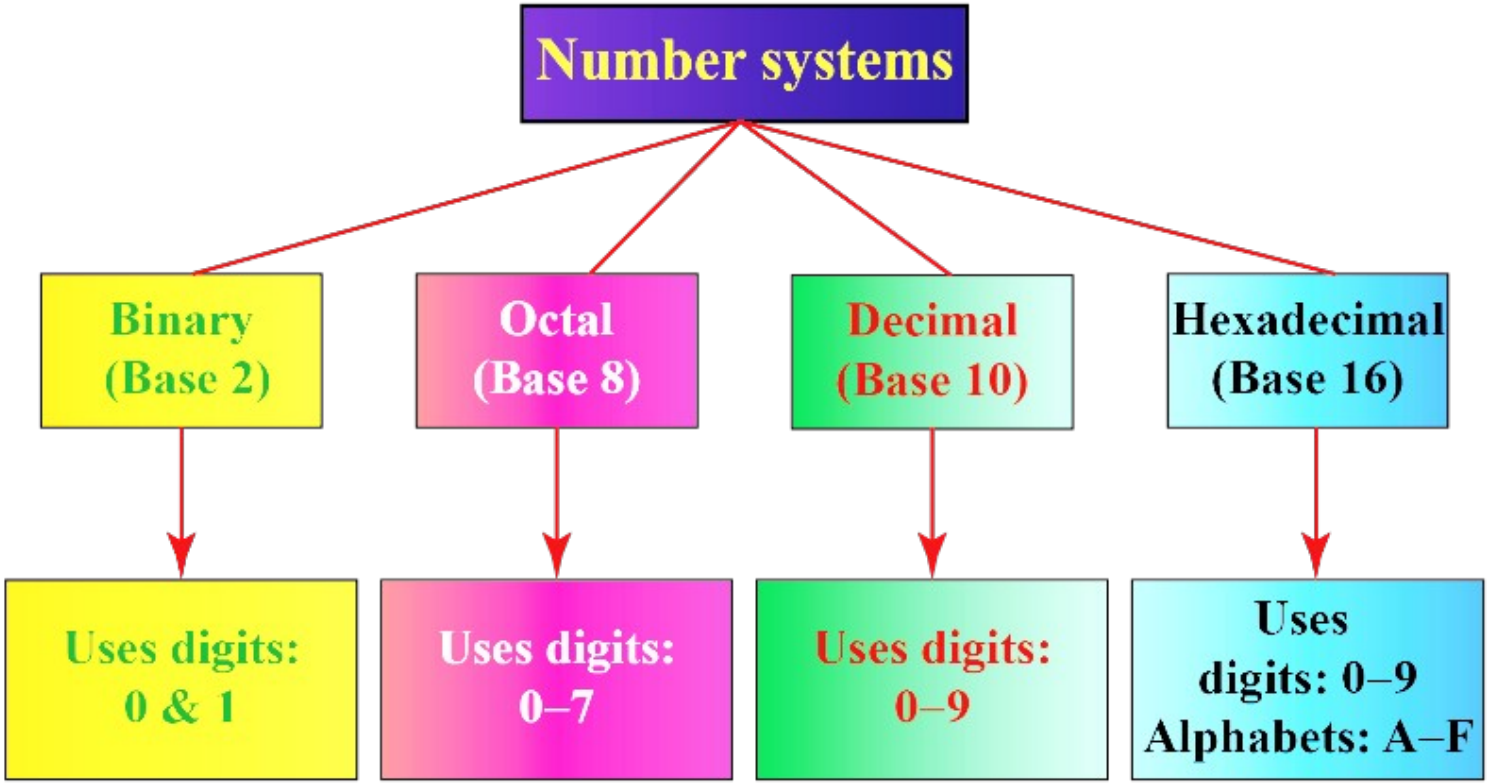
Presented by Nimesh Shiwakoti

What is Number

Number are arithmetical value, expressed by a word, symbol, or figure, representing a particular quantity and used in counting and making calculations.

What is Number System

The number system is simply a system to represent or express numbers. There are various types of number systems and the most commonly used ones are decimal number system, binary number system, octal number system, and hexadecimal number system.



Number Systems		
System	Base	Digits
Binary	2	0 1
Octal	8	0 1 2 3 4 5 6 7
Decimal	10	0 1 2 3 4 5 6 7 8 9
Hexadecimal	16	0 1 2 3 4 5 6 7 8 9 A B C D E F

Base of number system

A number base is the number of digits or combination of digits that a system of counting uses to represent numbers. A base can be any whole number greater than 0. The most commonly used number system is the decimal system, commonly known as base 10. Its popularity as a system of counting is most likely due to the fact that we have 10 fingers.

Decimal	Binary	Octal	Hexadecimal
0	0000	0	0
1	0001	1	1
2	0010	2	2
3	0011	3	3
4	0100	4	4
5	0101	5	5
6	0110	6	6
7	0111	7	7
8	1000	10	8
9	1001	11	9
10	1010	12	A
11	1011	13	B
12	1100	14	C
13	1101	15	D
14	1110	16	E
15	1111	17	F

Types of number system

- **Binary number system**

The binary number system, also called the base-2 number system, is a method of representing numbers that counts by using combinations of only two numerals: zero (0) and one (1). Computers use the binary number system to manipulate and store all of their data including numbers, words, videos, graphics, and music.

- **Octal number system**
- The octal numeral system, or octal for short, is the base-8 number system, and uses the digits 0 to 7

- **Decimal number system**
- A number system which uses digits from 0 to 9 to represent a number with base 10 is the decimal system number.

- **Hexadecimal number system**
- The word “Hexadecimal” means sixteen because this type of digital numbering system uses 16 different digits from 0-to-9, and A-to-F.

Decimal to Binary

1. To convert a decimal number to a binary number, you can follow these steps:
2. **Divide the decimal number by 2.**
3. **Record the remainder.**
4. **Update the decimal number to the quotient obtained from the division.**
5. **Repeat the process until the quotient is 0.**
6. **The binary number is the remainders read in reverse order (from last to first).**

fractional decimal number to binary

1. Converting the Integer Part

This process is the same as converting a whole number to binary as previously done.

2. Converting the Fractional Part

a. Multiply the fractional part by 2.
b. Record the integer part of the result.
c. Update the fractional part to the remainder after removing the integer part.
d. Repeat the process until the fractional part becomes 0 or you reach the desired precision.

13.9654 to $(?)_2$

First convert 13 to binary we get 1101

Now for fractional part 0.9654

Step	Multiplication	Result	Integer Part	Fractional Part
1	0.9654 × 2	1.9308	1	0.9308
2	0.9308 × 2	1.8616	1	0.8616
3	0.8616 × 2	1.7232	1	0.7232
4	0.7232 × 2	1.4464	1	0.4464
5	0.4464 × 2	0.8928	0	0.8928
6	0.8928 × 2	1.7856	1	0.7856
7	0.7856 × 2	1.5712	1	0.5712
8	0.5712 × 2	1.1424	1	0.1424
9	0.1424 × 2	0.2848	0	0.2848
10	0.2848 × 2	0.5696	0	0.5696

We'll stop here after 10 steps, but you can continue to get more precision.

inary Fractional Part: Reading the integer parts in order: 1111011100

13.9654=1101. 1111011100

Decimal to Octal

1. Divide the decimal number by 8.
2. Record the remainder.
3. Update the decimal number to the quotient obtained from the division.
4. Repeat the process until the quotient is 0.
5. The octal number is the remainders read in reverse order (from last to first)

Convert 7326_{10} to $(?)_8$

Fractional Decimal to octal

Converting the Integer Part

The process is similar to converting a decimal number to octal.

Converting the Fractional Part

1. **Multiply the fractional part by 8.**
2. **Record the integer part of the result.**
3. **Update the fractional part to the remainder after removing the integer part.**
4. **Repeat the process until the fractional part becomes 0 or you reach the desired precision.**

Converting 0.9654 to Octal

Step	Multiply by 8	Result	Integer Part	Fractional Part
1	0.9654 × 8	7.7232	7	0.7232
2	0.7232 × 8	5.7856	5	0.7856
3	0.7856 × 8	6.2848	6	0.2848
4	0.2848 × 8	2.2784	2	0.2784
5	0.2784 × 8	2.2272	2	0.2272
6	0.2272 × 8	1.8176	1	0.8176
7	0.8176 × 8	6.5408	6	0.5408
8	0.5408 × 8	4.3264	4	0.3264
9	0.3264 × 8	2.6112	2	0.6112
10	0.6112 × 8	4.8896	4	0.8896

$0.9654_{10} \approx 0.75622164248$

Decimal to Hexadecimal

1. **Divide the decimal number by 16.**
2. **Record the remainder.** (This will be a hexadecimal digit)
3. **Update the decimal number to the quotient obtained from the division.**
4. **Repeat the process until the quotient is 0.**
5. **The hexadecimal number is the remainders read in reverse order (from last to first).**
6. **Replace**

   10 → A 11 → B 12 → C 13 → D 14 → E 15 → F

fractional decimal number to hexadecimal

Converting the Fractional Part

1. Multiply the fractional part by 16.
2. Record the integer part of the result.
3. Update the fractional part to the remainder after removing the integer part.
4. Repeat the process until the fractional part becomes 0 or you reach the desired precision.

In Integer column replace

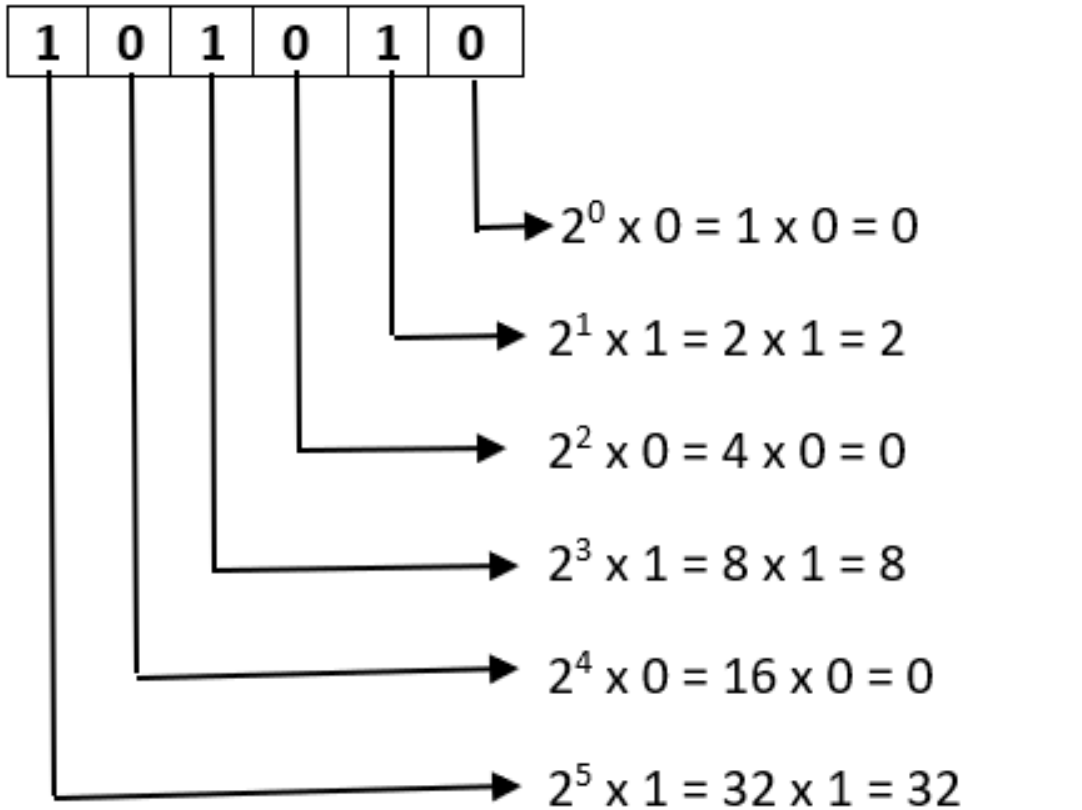
5. Replace

10 → A 11 → B 12 → C 13 → D 14 → E 15 → F

Step	Fractional Part	Multiplication Result	Integer Part	Hexadecimal Digit
1	0.9654	0.9654×16 = 15.4464	15	F
2	0.4464	0.4464×16 = 7.1424	7	7
3	0.1424	0.1424×16 = 2.2784	2	2
4	0.2784	0.2784×16 = 4.4544	4	4

Binary to decimal

Convert $(101010)_2 = (?)_{10}$



Resultant decimal number= 0+2+0+8+0+32 = **42**

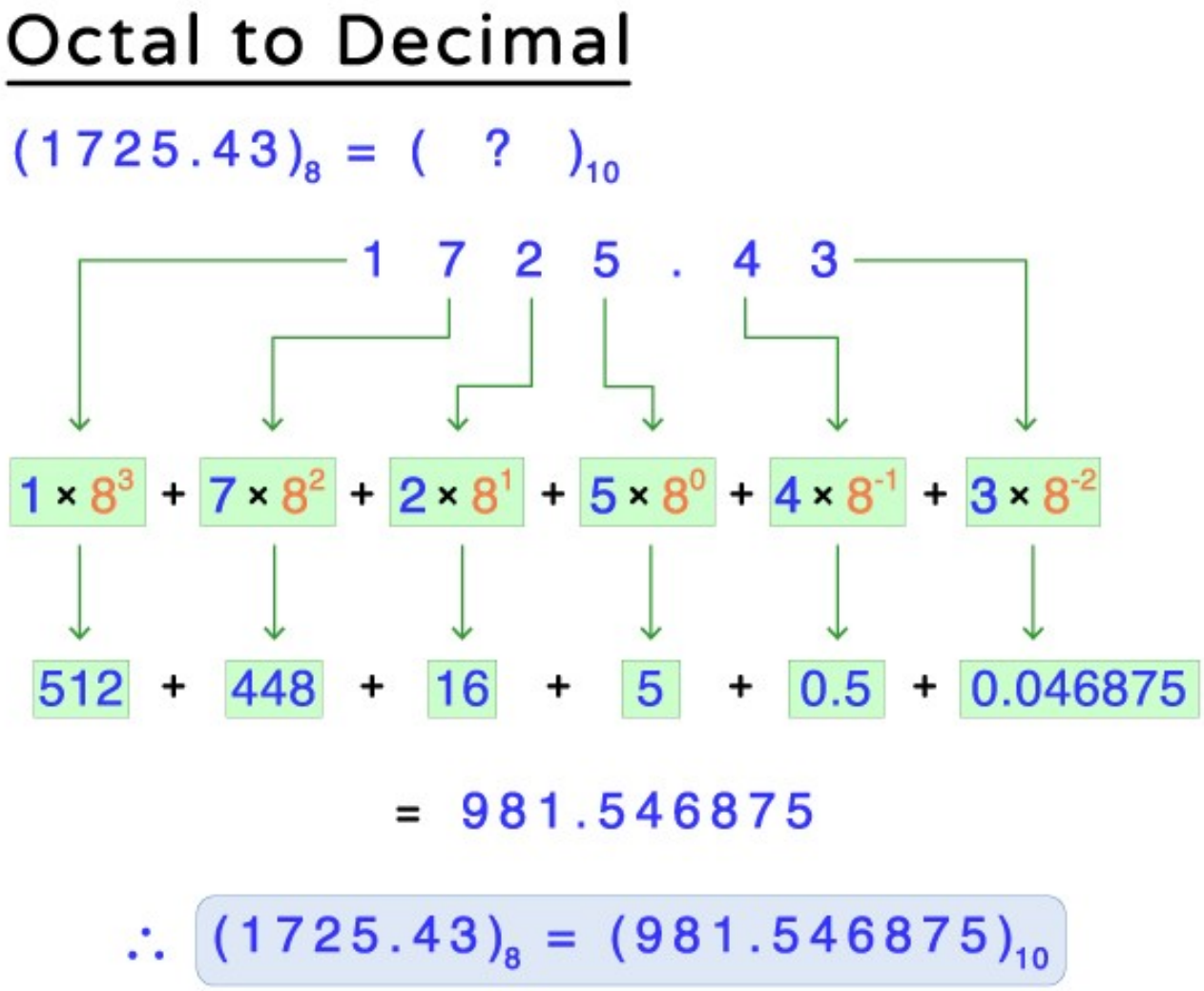
fractional binary number to decimal

To convert a fractional binary number to decimal, follow these steps:

1. **Separate the Binary Number into Integer and Fractional Parts:**
    - For example, consider the binary number 101.101_2. The integer part is 101_2 and the fractional part is 0.101_2.
2. **Convert the Integer Part to Decimal:**
    - Convert the integer part 101_2 to decimal using the method described earlier:
        - $1 \times 2^2 = 4$
        - $0 \times 2^1 = 0$
        - $1 \times 2^0 = 1$
        - Sum: $4 + 0 + 1 = 5$
    - So, 101_2 in decimal is 5_{10}.

3. **Convert the Fractional Part to Decimal:**
   - Convert the fractional part 0.101_2 by multiplying each digit by 2^{-n}, where n is the position of the digit after the binary point (starting from 1):
     - The first digit (1) is multiplied by 2^{-1}: $1 \times \frac{1}{2} = 0.5$
     - The second digit (0) is multiplied by 2^{-2}: $0 \times \frac{1}{4} = 0$
     - The third digit (1) is multiplied by 2^{-3}: $1 \times \frac{1}{8} = 0.125$
   - Sum of the results: $0.5 + 0 + 0.125 = 0.625$
   - So, 0.101_2 in decimal is 0.625_{10}.
4. **Combine the Results:**
   - Add the integer part and fractional part together: $5 + 0.625 = 5.625$

convert an octal number (base 8) to decimal (base 10)



Hexadecimal to decimal

Digit	5	4	D	2
Place value	16^1	16^0	16^{-1}	16^{-2}

$54.D2_{16}$

$= 5 \cdot 16^1 + 4 \cdot 16^0 + D \cdot 16^{-1} + 2 \cdot 16^{-2}$

$= 5 \cdot 16^1 + 4 \cdot 16^0 + 13 \cdot 16^{-1} + 2 \cdot 16^{-2}$

$= 80 + 4 + 0.8125 + 0.0078125$

$= 84.8203125$

Binary to octal

- To convert a binary number to octal, you can use the following steps:
- **Steps for Conversion:**
- **Group the Binary Digits:**
  - Binary numbers are grouped into sets of three digits each, starting from the binary point (if any). If the number of digits is not a multiple of three, add leading zeros to make it so.
  - For example, convert 110101101_2 to octal.
- **Convert Each Group of Three Binary Digits to Octal:**
  - Each group of three binary digits can be directly converted to a single octal digit.
- **Combine the Octal Digits:**
  - Combine the octal digits to form the final octal number.

Example Conversion:

Convert 110010110_2 to octal.

- **Group Binary Digits into Sets of Three:**
  - Starting from the right: 011, 010,110
  - If there are leading digits, pad them with zeros: 011, 010,110

- **Convert Each Binary Group to Octal:**
  - $011_2=3_8$
  - $010_2=2_8$
  - $110_2=6_8$
- Combine the results to form the octal number: 626_8

Fractional binary to octal

- **Steps for Conversion:**
- **Separate the Binary Number:**
  - Split the binary number into integer and fractional parts. For example, 101.1012101.101_2101.1012.
- **Convert the Integer Part to Octal:**
  - Group the integer part into sets of three binary digits from right to left. Convert each group to octal.
- **Convert the Fractional Part to Octal:**
  - Group the fractional part into sets of three binary digits from left to right. Convert each group to octal.
- **Combine the Results:**
  - Combine the octal results from both the integer and fractional parts to get the final octal number.

Convert 101.101_2 to octal.

1. **Convert the Integer Part:**
   - Integer part: 101_2
   - Group into sets of three (already grouped): 101
   - Convert 101_2 to octal:
     - $101_2 = 5_8$
2. **Convert the Fractional Part:**
   - Fractional part: 0.101_2
   - Group into sets of three (add trailing zeros if needed): 101
   - Convert 101_2 to octal:
     - $101_2 = 5_8$
3. **Combine the Results:**
   - Integer part in octal: 5
   - Fractional part in octal: 5

The final octal representation is 5.5_8.

binary number to hexadecimal

- To convert a binary number to hexadecimal, follow these steps:
- **Steps for Conversion:**
- **Group the Binary Digits:**
  - Group the binary digits into sets of four, starting from the binary point (if any). If the number of digits is not a multiple of four, add leading zeros to make it so.
- **Convert Each Group of Four Binary Digits to Hexadecimal:**
  - Each group of four binary digits can be directly converted to a single hexadecimal digit.
- **Combine the Hexadecimal Digits:**
  - Combine the hexadecimal digits to form the final hexadecimal number.

Example Conversion:

Convert 110101101_2 to hexadecimal.

1. **Group Binary Digits into Sets of Four:**
   - Starting from the right: 1101, 0110, 1
   - Pad the leftmost group with zeros to make it four digits: 0001, 1011, 0110
2. **Convert Each Binary Group to Hexadecimal:**
   - $0001_2 = 1_{16}$
   - $1011_2 = B_{16}$
   - $0110_2 = 6_{16}$

   Combine the results to form the hexadecimal number: $1B6_{16}$

fractional binary number to hexadecimal

- **Steps for Conversion:**
- **Separate the Binary Number:**
  - Split the binary number into integer and fractional parts. For example, consider 101.101_2.
- **Convert the Integer Part to Hexadecimal:**
  - Group the integer part into sets of four binary digits from right to left. Convert each group to hexadecimal.
- **Convert the Fractional Part to Hexadecimal:**
  - Group the fractional part into sets of four binary digits from left to right. Add trailing zeros if needed, then convert each group to hexadecimal.
- **Combine the Results:**
  - Combine the hexadecimal results from both the integer and fractional parts to get the final hexadecimal number.

Example Conversion:

Convert 101.101_2 to hexadecimal.

1. Convert the Integer Part:

- Integer part: 101_2
- Pad with leading zeros if needed: 0101_2
- Group into sets of four: 0101
- Convert 0101_2 to hexadecimal:
  - $0101_2 = 5_{16}$

2. Convert the Fractional Part:

- Fractional part: 0.101_2
- Pad with trailing zeros if needed: 0.1010_2 (Add one zero to make a full group)
- Group into sets of four: 1010
- Convert 1010_2 to hexadecimal:
  - $1010_2 = A_{16}$

3. Combine the Results:

- Integer part in hexadecimal: 5
- Fractional part in hexadecimal: A

So, 101.101_2 in binary is $5.A_{16}$ in hexadecimal.

octal number to binary

- **Steps for Conversion:**
- **Separate the Octal Number into Individual Digits:**
  - For example, consider the octal number 254_8.
- **Convert Each Octal Digit to Binary:**
  - Each octal digit is directly converted to a 3-bit binary number.
- **Combine the Binary Results:**
  - Combine the binary equivalents of each octal digit to form the final binary number.

Detailed Conversion Table:

Octal Digit	Binary
2	010
5	101
4	100

fractional octal number to binary

- **Steps for Conversion:**
- **Separate the Octal Number into Integer and Fractional Parts:**
  - For example, consider 34.5_8.
- **Convert the Integer Part to Binary:**
  - Convert each octal digit in the integer part to its 3-bit binary equivalent.
- **Convert the Fractional Part to Binary:**
  - Convert each octal digit in the fractional part to its 3-bit binary equivalent.
- **Combine the Results:**
  - Combine the binary results from both the integer and fractional parts.

Detailed Conversion Table:

Octal Digit (Integer)	Binary
3	011
4	100

Octal Digit (Fractional)	Binary
5	101

Summary:

1. Convert each octal digit to its 3-bit binary equivalent.
2. Combine the binary digits from the integer and fractional parts to form the final binary number.

So, 34.5_8 in octal is 011100.101_2 in binary.

hexadecimal number to binary

- **Steps for Conversion:**
- **Separate the Hexadecimal Number into Individual Digits:**
  - For example, consider the hexadecimal number $2F3_{16}$.
- **Convert Each Hexadecimal Digit to Binary:**
  - Each hexadecimal digit converts directly to a 4-bit binary number.
- **Combine the Binary Results:**
  - Combine the binary equivalents of each hexadecimal digit to form the final binary number.

Detailed Conversion Table:

Hexadecimal Digit	Binary
2	0010
F	1111
3	0011

Summary:

1. Convert each hexadecimal digit to its 4-bit binary equivalent.
2. Combine the binary digits to form the final binary number.

So, $2F3_{16}$ in hexadecimal is 001011110011_2 in binary.

fractional hexadecimal number to binary

- **Steps for Conversion:**
- **Separate the Hexadecimal Number into Integer and Fractional Parts:**
  - For example, consider $1A.3F_{16}$.
- **Convert the Integer Part to Binary:**
  - Convert each hexadecimal digit in the integer part to its 4-bit binary equivalent.
- **Convert the Fractional Part to Binary:**
  - Convert each hexadecimal digit in the fractional part to its 4-bit binary equivalent.
- **Combine the Results:**
  - Combine the binary results from both the integer and fractional parts.

Detailed Conversion Table:

Hexadecimal Digit (Integer)	Binary
1	0001
A	1010

Hexadecimal Digit (Fractional)	Binary
3	0011
F	1111

Summary:

1. Convert each hexadecimal digit to its 4-bit binary equivalent.
2. Combine the binary digits from the integer and fractional parts to form the final binary number.

So, $1A.3F_{16}$ in hexadecimal is $0001\ 1010.0011\ 1111_2$ in binary.

Octal to hexadecimal

- **Steps for Conversion:**
- **Convert Octal to Binary:**
  - Convert each octal digit to its 3-bit binary equivalent.
- **Group Binary Digits into Sets of Four:**
  - Group the binary digits into sets of four, starting from the binary point (if any). Add leading or trailing zeros if necessary.
- **Convert Binary to Hexadecimal:**
  - Convert each group of four binary digits to its hexadecimal equivalent.

Detailed Conversion Table:

Octal Digit	Binary	Hexadecimal
2	010	2
5	101	5
7	111	7

Summary:

1. Convert each octal digit to its 3-bit binary equivalent.
2. Group the binary digits into sets of four.
3. Convert each 4-bit binary group to its hexadecimal equivalent.

So, 257_8 in octal is 57_{16} in hexadecimal.

hexadecimal number to octal

- To convert a hexadecimal number to octal, follow these steps:
- **Steps for Conversion:**
- **Convert Hexadecimal to Binary:**
  - Convert each hexadecimal digit to its 4-bit binary equivalent.
- **Group Binary Digits into Sets of Three:**
  - Group the binary digits into sets of three, starting from the binary point (if any). Add leading or trailing zeros if necessary.
- **Convert Binary to Octal:**
  - Convert each group of three binary digits to its octal equivalent.

Detailed Conversion Table:

Hexadecimal Digit	Binary	Octal
1	0001	1
F	1111	17
3	0011	3

Summary:

1. Convert each hexadecimal digit to its 4-bit binary equivalent.
2. Group the binary digits into sets of three (pad with leading zeros if needed).
3. Convert each 3-bit binary group to its octal equivalent.

So, $1F3_{16}$ in hexadecimal is $1F3_{8}$ in octal.

Calculations in Binary

Binary addition is similar to decimal addition but operates on binary numbers (base-2).

Steps for Binary Addition:

Align the Binary Numbers:

Write the numbers you want to add, aligning them by their least significant bits (rightmost).

Add Each Column:

Add the digits in each column from right to left, just like decimal addition. Remember the binary addition rules:

0+0=0

0+1=1

1+1=10 (which is 0 with a carry of 1)

1+1+1=11 (which is 1 with a carry of 1)

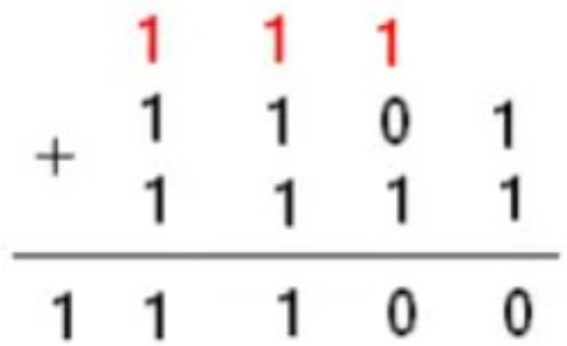
- **Carry Over:**

If the sum of the column exceeds 1, write down the result and carry over the excess to the next column.

- **Write Down the Result:**

Continue this process until all columns are added.

Example



Binary subtraction

- **Steps for Binary Subtraction:**
- **Align the Binary Numbers:**
  - Write the numbers you want to subtract, aligning them by their least significant bits (rightmost).
- **Subtract Each Column:**
  - Subtract the digits in each column from right to left. If the minuend (top number) is smaller than the subtrahend (bottom number), you need to borrow from the next column.
- **Borrowing:**
  - If you need to borrow, subtract 1 from the next higher column and add 222 (binary 101010) to the current column.
- **Write Down the Result:**
  - Continue this process until all columns are subtracted.

Binary Subtraction

The binary subtraction can be performed in two ways—direct subtracting and by using complement methods. Direct subtraction can be done as decimal subtraction where,

0 - 0 = 0, 1 - 1 = 0, 1 – 0 = 1, 10 – 1 =1 and 0 – 1 = 1 if we can borrow 1 from next column otherwise we cannot perform the 0 – 1 subtraction. To solve this type of problem we have to follow the 1's or 2's complement method of binary subtraction.

For example,

a. Subtract $(10)_2$ from $(100)_2$ by using direct method.

Here,

$$\begin{array}{r} 100 \\ -10 \\ \hline 10 \end{array}$$

b. Subtract $(1110)_2$ from $(1001)_2$ by using direct method.

Here,

$$\begin{array}{r} 1001 \\ -1110 \end{array}$$

Here, we cannot perform the operation because second number is greater than first number. To subtract this number in direct method simply interchange two number and place negative sign after subtraction. That is:

$$\begin{array}{r} 1110 \\ -1001 \end{array}$$

0101, we reverse the number hence answer is negative i.e. $(-0101)_2$

One's and Two's Complement Methods of Binary Subtraction

1's Complement Method

A 1's complement of the given number is obtained simply by subtracting each bit from 1 or making 0 to 1 and 1 to 0.

The algorithm or procedure for subtracting binary number by using 1's complement

Step 1. Make the number of digit equal in both Minuend and Subtrahend by adding 0 at the leftmost of the least number of digit number.

Step 2. Calculate 1's complement of Subtrahend.

Step 3. Calculate sum of Minuend and the 1's complement of Subtrahend

Step 4. Check the overflow digit/bit on the leftmost

- If there is overflow digit (more number of digit), add the leftmost bit to the remaining part of the sum and the final sum is the answer.
- If there is no overflow digit then the result must be negative, so the calculated value of 1's complement of the sum is the answer.

Example Subtract 1011 from 11100 by using 1's complement method.

Step 1. Making the number of digit equal in both Minuend and Subtrahend by adding 0 at the leftmost of the least number of digit number:

Minuend	1 1 1 0 0	Main number
Subtrahend	0 1 0 1 1	Negative number

Step 2. Calculation of 1's complement of Subtrahend i.e. 0 1 0 1 1 = 1 0 1 0 0

Step 3. Calculation of sum of Minuend and the 1's complement of Subtrahend

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

Step 4. While checking, it is found the overflow mode, so we add the leftmost bit to the remaining part of the sum for the final answer i.e.

$$\begin{array}{r} 1\,0\,0\,0\,0 \\ +1 \\ \hline (1\,0\,0\,0\,1)_2 \end{array}$$

The final answer is:

Example Subtract 11100 from 1011 by using 1's complement method.

Step 1. Making the number of digit equal in both Minuend and Subtrahend by adding 0 at the leftmost of the least number of digit number:

Minuend	0 1 0 1 1	Main number
Subtrahend	1 1 1 0 0	Negative number

Step 2. Calculation of 1's complement of Subtrahend i.e. 1 1 1 0 0 = 0 0 0 1 1

Step 3. Calculation of sum of Minuend and the 1's complement of Subtrahend

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

Step 4. While checking, found there is no overflow mode, hence answer is in negative and it obtains by converting the sum into 1's complement i.e. $(-\,1\,0\,0\,0\,1)_2$ (with negative mark).

2's Complement Method

A 2's complement of a given binary number is simply obtained by adding 1 to the 1's complement of the given binary number. Example: the 2's complement of 1101 is : 0010 + 1 = 0011.

The algorithm or procedure for subtracting decimal number by using 1's complement

Step 1. Make the number of digit equal in both Minuend and Subtrahend by adding 0 at the leftmost of the least number of digit number.

Step 2. Calculate 2's complement of Subtrahend
OR add 1 to the 1's complement of subtrahend.

Step 3. Calculate sum of Minuend and the 2's complement of subtrahend.

Step 4. Check the overflow digit/bit on the leftmost

- If there is overflow digit (more number of digit), remove the leftmost bit and the remaining part of the sum is the final answer.
- If there is no overflow digit then the result must be negative, so the calculated value of 2's complement of the sum is the answer.

Example Subtract 1011 from 11100 by using 2's complement method.

Step 1. Making the number of digit equal in both Minuend and Subtrahend by adding 0 at the leftmost of the least number of digit number:

Minuend	1 1 1 0 0	Main number
Subtrahend	0 1 0 1 1	Negative number

Step 2. Calculation of 2's complement of Subtrahend i.e. 0 1 0 1 1 = 1 0 1 0 0 + 1 = 1 0 1 0 1

Step 3. Calculation of sum of Minuend and the 2's complement of Subtrahend

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

Step 4. While checking, it is found the overflow mode, so we just remove the leftmost bit and the remaining part of the sum is the final answer i.e. $(1\,0\,0\,0\,1)_2$

Example Subtract 11100 from 1011 by using 2's complement method.

Step 1. Making the number of digit equal in both Minuend and Subtrahend by adding 0 at the leftmost of the least number of digit number:

Minuend	0 1 0 1 1	Main number
Subtrahend	1 1 1 0 0	Negative number

Step 2. Calculation of 2's complement of Subtrahend i.e. 1 1 1 0 0 = 0 0 0 1 1 + 1 = 0 0 1 0 0

Step 3. Calculation of sum of Minuend and the 1's complement of Subtrahend

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

Step 4. While checking, there is no overflow digit then the result must be negative, so the calculated value of 2's complement of the sum is the answeri.e. 0 1 1 1 1= 1 0 0 0 0 + 1 = 1 0 0 0 1

Now, place negative mark in answer i.e. $(-\,1\,0\,0\,0\,1)_2$.

Introduction to Boolean algebra

Boolean algebra is a branch of algebra that deals with binary variables and logical operations. It is fundamental in computer science, digital logic design, and electrical engineering. Boolean algebra simplifies and manipulates logical expressions and is essential for designing and understanding digital circuits.

Key Concepts in Boolean Algebra

1. Boolean Variables:

Definition: Variables that can take on one of two possible values: 000 (false) or 111 (true).

Usage: Represent the basic elements in Boolean algebra and digital logic circuits.

Examples: AAA, BBB, XXX, YYY.

- **2. Boolean Expressions:**
- **Definition:** Combinations of Boolean variables and logical operators (AND, OR, NOT) that represent a logical relationship.
- **Operators:**
  - **AND (· or &):** Results in true if both operands are true. Example: A·B
  - **OR (+ or |):** Results in true if at least one operand is true. Example: A+B
  - **NOT ($\bar{A}$ or A’):** Inverts the value of the operand. Example: $\bar{A}$
- **Examples:** A·B+$\bar{C}$, (A+B)·$\bar{C}$

3. Boolean Functions:

Definition: A function that maps a set of Boolean variables to a single Boolean output.

Examples:

$F(A,B)=A \cdot B$

$G(X,Y,Z)=(X+Y) \cdot \overline{Z}$

Purpose: Used to define the output of logic circuits based on the inputs.

4. Truth Tables:

Definition: A table that shows all possible values of the Boolean variables and the corresponding output of a Boolean function.

Construction:

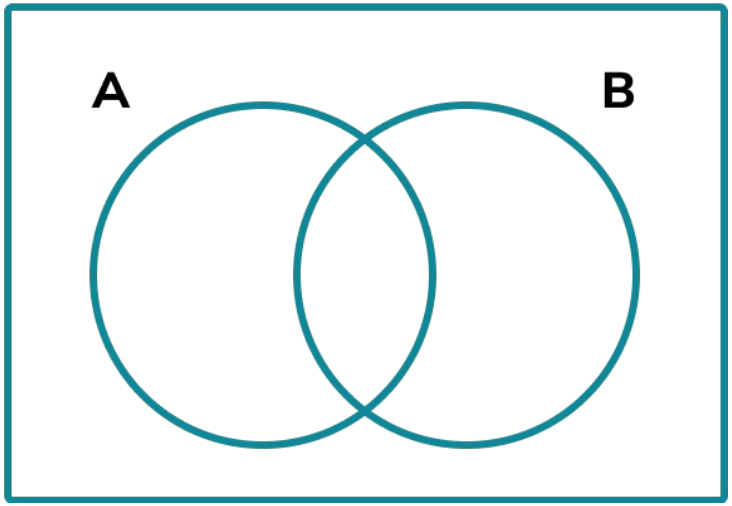
List all possible combinations of input values.

Compute the output for each combination.

Example for F(A,B)=A·B

A	B	A·B
0	0	0
0	1	0
1	0	0
1	1	1

- **5. Venn Diagrams:**
- **Definition:** A graphical tool used to represent Boolean expressions and their relationships visually.
- **Components:**
  - **Circles:** Represent Boolean variables or sets.
  - **Overlapping Areas:** Show the logical operations between sets (AND, OR, NOT).
- **Usage:**
  - **AND Operation:** Overlapping regions of circles.
  - **OR Operation:** Union of circles.
  - **NOT Operation:** Area outside a circle.



Basic Laws and Properties:

In Boolean algebra, there are several fundamental laws and properties that simplify Boolean expressions and are crucial for designing and analyzing digital circuits. Here’s an overview of these basic laws and properties:

Identity Law

1. Identity Law

AND Identity:

$A \cdot 1 = A$

AAA	1	A·1
0	1	0
1	1	1

OR Identity:

$A + 0 = A$

A	0	A+0
0	0	0
1	0	1

Explanation:

ANDing any value with 1 results in the original value.

ORing any value with 0 results in the original value.

Complement Law

AND Complement:

$A \cdot \bar{A} = 0$

A	$\bar{A}$	$A \cdot \bar{A}$
0	1	0
1	0	0

OR Complement:

A	$\bar{A}$	$A + \bar{A}$
0	1	1
1	0	1

- **Explanation:**
- ANDing a variable with its complement results in 0.
- ORing a variable with its complement results in 1.

Associative Law

AND Associative:

$$(A \cdot B) \cdot C = A \cdot (B \cdot C)$$

A	B	C	(A·B)·C	A·(B·C)
0	0	0	0	0
0	0	1	0	0
0	1	0	0	0
0	1	1	0	0
1	0	0	0	0
1	0	1	0	0
1	1	0	0	0
1	1	1	1	1

OR Associative:

(A+B)+C=A+(B+C)

- **Explanation:**

The grouping of variables does not affect the result of AND or OR operations.

A	B	C	(A+B)+C	A+(B+C)
0	0	0	0	0
0	0	1	1	1
0	1	0	1	1
0	1	1	1	

Distributive Law

Distributive over OR:

$A \cdot (B+C) = (A \cdot B) + (A \cdot C)$

A	B	C	B+C	A·(B+C)	A·B	A·C	(A·B)+ (A·C)
0	0	0	0	0	0	0	0
0	0	1	1	0	0	0	0
0	1	0	1	0	0	0	0
0	1	1	1	0	0	0	0
1	0	0	0	0	0	0	0
1	0	1	1	1	0	1	1
1	1	0	1	1	1	0	1
1	1	1	1	1	1	1	1

Distributive over AND:

$A+(B\cdot C)=(A+B)\cdot(A+C)$

Explanation:

Distributive laws allow for the expansion or factoring of Boolean expressions.

A	B	C	B·C	A+(B·C)	A+B	A+C	(A+B)·(A+C)
0	0	0	0	0	0	0	0
0	0	1	0	0	0	1	0
0	1	0	0	0	1	0	0
0	1	1	1	1	1	1	1
1	0	0	0	1	1	1	1
1	0	1	0	1	1	1	1
1	1	0	0	1	1	1	1
1	1	1	1	1	1	1	1

Commutative Law

AND Commutative:

$A \cdot B = B \cdot A$

A	B	A·B	B·A
0	0	0	0
0	1	0	0
1	0	0	0
1	1	1	1

OR Commutative:

A+B=B+A

Explanation:

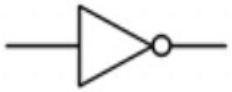
The order of variables does not affect the result of AND or OR operations.

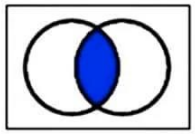
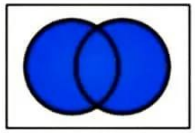
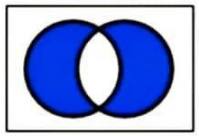
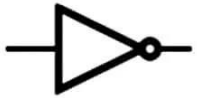
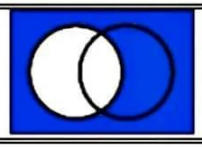
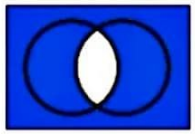
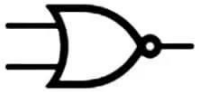
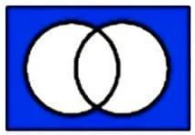
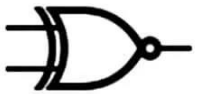
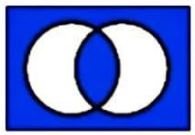
A	B	A+B	B+A
0	0	0	0
0	1	1	1
1	0	1	1
1	1	1	1

Logic gate

A logic gate is an electronic component that performs a basic logical function in digital circuits. It operates on one or more binary inputs to produce a single binary output. The primary role of logic gates is to process and control digital signals, performing operations based on Boolean algebra.

- **Key Characteristics of Logic Gates:**
- **Binary Operation:** Logic gates work with binary inputs, which are represented as either 0 (low) or 1 (high). The output is also binary.
- **Boolean Functions:** Each logic gate implements a specific Boolean function. Boolean algebra is used to describe the relationships between inputs and outputs.
- **Symbolic Representation:** Logic gates are represented by standard symbols in circuit diagrams, and they can be implemented using various technologies, such as transistors.
- **Digital Logic Circuits:** Logic gates are combined to build more complex digital systems, including arithmetic logic units (ALUs), memory units, and other components in computers and digital devices.

Logic Gate	Symbol	Description	Boolean
AND		Output is at logic 1 when, and only when all its inputs are at logic 1,otherwise the output is at logic 0.	$X = A \cdot B$
OR		Output is at logic 1 when one or more are at logic 1.If all inputs are at logic 0,output is at logic 0.	$X = A+B$
NAND		Output is at logic 0 when,and only when all its inputs are at logic 1,otherwise the output is at logic 1	$X = \overline{A \cdot B}$
NOR		Output is at logic 0 when one or more of its inputs are at logic 1.If all the inputs are at logic 0,the output is at logic 1.	$X = \overline{A+B}$
XOR		Output is at logic 1 when one and Only one of its inputs is at logic 1. Otherwise is it logic 0.	$X = A \oplus B$
XNOR		Output is at logic 0 when one and only one of its inputs is at logic1.Otherwise it is logic 1. Similar to XOR but inverted.	$X = \overline{A \oplus B}$
NOT		Output is at logic 0 when its only input is at logic 1, and at logic 1 when its only input is at logic 0.That's why it is called and INVERTER	$X = \overline{A}$

Expression	Symbol	Venn diagram	Boolean algebra	Values		
AND			$A \cdot B$	A	B	Output
				0	0	0
				0	1	0
				1	0	0
				1	1	1
OR			$A + B$	A	B	Output
				0	0	0
				0	1	1
				1	0	1
				1	1	1
XOR			$A \oplus B$	A	B	Output
				0	0	0
				0	1	1
				1	0	1
				1	1	0
NOT			$\overline{A}$	A		Output
				0		1
				1		0
NAND			$\overline{A \cdot B}$	A	B	Output
				0	0	1
				0	1	1
				1	0	1
				1	1	0
NOR			$\overline{A + B}$	A	B	Output
				0	0	1
				0	1	0
				1	0	0
				1	1	0
XNOR			$\overline{A \oplus B}$	A	B	Output
				0	0	1
				0	1	0
				1	0	0
				1	1	1

De Morgan’s Theorem

De Morgan’s Theorems are two of the most important rules in Boolean algebra and digital logic design. These theorems help simplify expressions involving **NOT**, **AND**, and **OR** operations. There are two De Morgan’s Theorm

De Morgan’s First Theorem

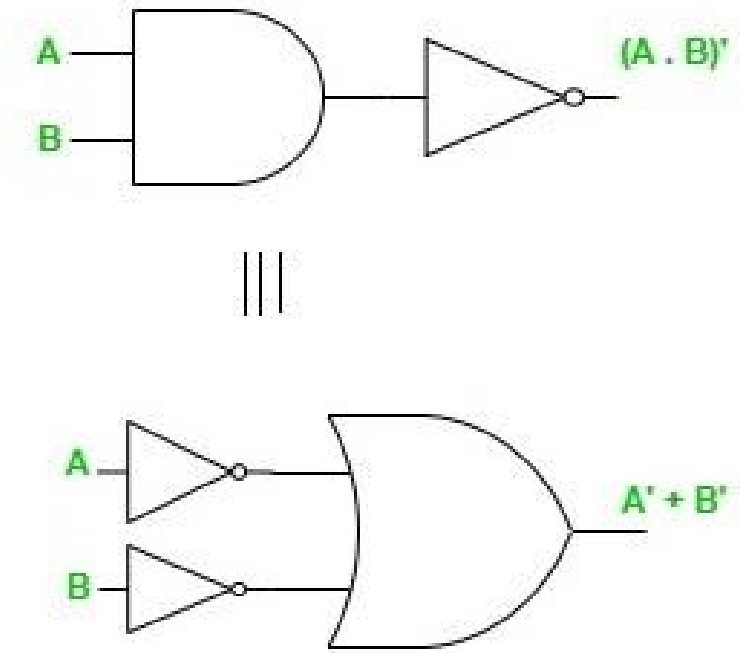
The first theorem states that the inversion of the product is the same as the sum of the inversions

NOT (A AND B) = (NOT A) OR (NOT B)

In symbols:

$(A \cdot B)' = A' + B'$

De morgan's first theorem Logic Gates



A	B	A · B	(A · B)'	A'	B'	A' +B'
0	0	0	1	1	1	1
0	1	0	1	1	0	1
1	0	0	1	0	1	1
1	1	1	0	0	0	0

De Morgan’s Second Theorem

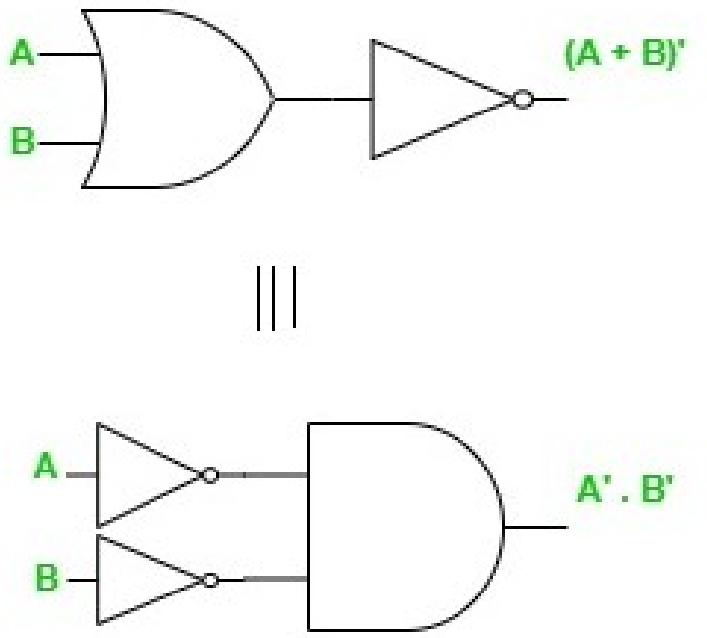
The second theorem states that the inversion of the sum is the same as the product of the inversions.

NOT (A OR B) = (NOT A) AND (NOT B)

In symbols:

$(A+B)'=A'\cdot B'$

De morgan's second theorem Logic Gates



A	B	A + B	(A + B)'	A'	B'	A' · B'
0	0	0	1	1	1	1
0	1	1	0	1	0	0
1	0	1	0	0	1	0
1	1	1	0	0	0	0