How can I calculate the bitwise XOR (exclusive OR) of two cells in Excel?

Authored by

stats writer

June 28, 2024

The bitwise XOR (exclusive OR) operation is a logical operation that compares two binary values and returns a result based on their differences. In Excel, this operation can be applied to two cells by using the XOR function. To calculate the bitwise XOR of two cells in Excel, simply enter the formula "=XOR(cell1,cell2)" into a third cell, where "cell1" and "cell2" are the two cells you want to compare. This will return a result of either TRUE or FALSE, indicating whether the two cells have different binary values. The XOR function is useful for comparing and evaluating data in a binary format, such as binary numbers or text strings.

This article describes the formula syntax and usage of the **BITXOR** function in Microsoft Excel.

Description

Returns a bitwise 'XOR' of two numbers.

Syntax

BITXOR(number1, number2)

The BITXOR function syntax has the following arguments.

Number1 Required. Must be greater than or equal to 0.

Number2 Required. Must be greater than or equal to 0.

Remarks

BITXOR returns a decimal number that is the result of the sum of a bitwise 'XOR' (exclusive XOR) of its parameters.

If either argument is outside its constraint, BITXOR returns the #NUM! error value.

If either argument is greater than (2^48)-1, BITXOR returns the #NUM! error value.

If either argument is a non-numeric value, BITXOR returns the #VALUE! error value.

In the result, each bit position is 1 if the values of the parameters at that bit position are not equal; in other words, one value is 0 and the other is 1. For example, using BITXOR(5,3), 5 is expressed as 101 in binary and 3 as 11 in binary. To help with comparison, you can consider 3 as 011. From right to left, the bit values at the three positions in this example are the same (1) only at the rightmost position. A 'not equal' result returns a 1 for the second and third positions from the right, and an 'equal' result returns 0 for the rightmost position.

Values of 1 returned from the bit positions progress from right to left as powers of 2. The rightmost bit returns 1 (2^0), the bit to the left returns 2 (2^1), and so on.

Using the same example, 0 is returned for the rightmost bit position because it is a 0, 2 (2^1) is returned for the second bit position from the right (a 1 value), and 4 (2^2) is returned for the leftmost bit (also a 1 value). The total is 6, in decimal representation.