DEFINITION

bitwise

By Rahul Awati

What is bitwise?

Bitwise is a level of operation that involves working with individual bits which are the smallest units of data in a computing system. Each bit has single binary value of 0 or 1. Most programming languages manipulate groups of 8, 16 or 32 bits. These bit multiples are known as bytes.

The arithmetic logic unit (ALU) is a part of a computer's CPU. Inside the ALU, mathematical operations like addition, subtraction, multiplication and division are all done at bit level. For those operations, bitwise operators are used.

Bitwise operations

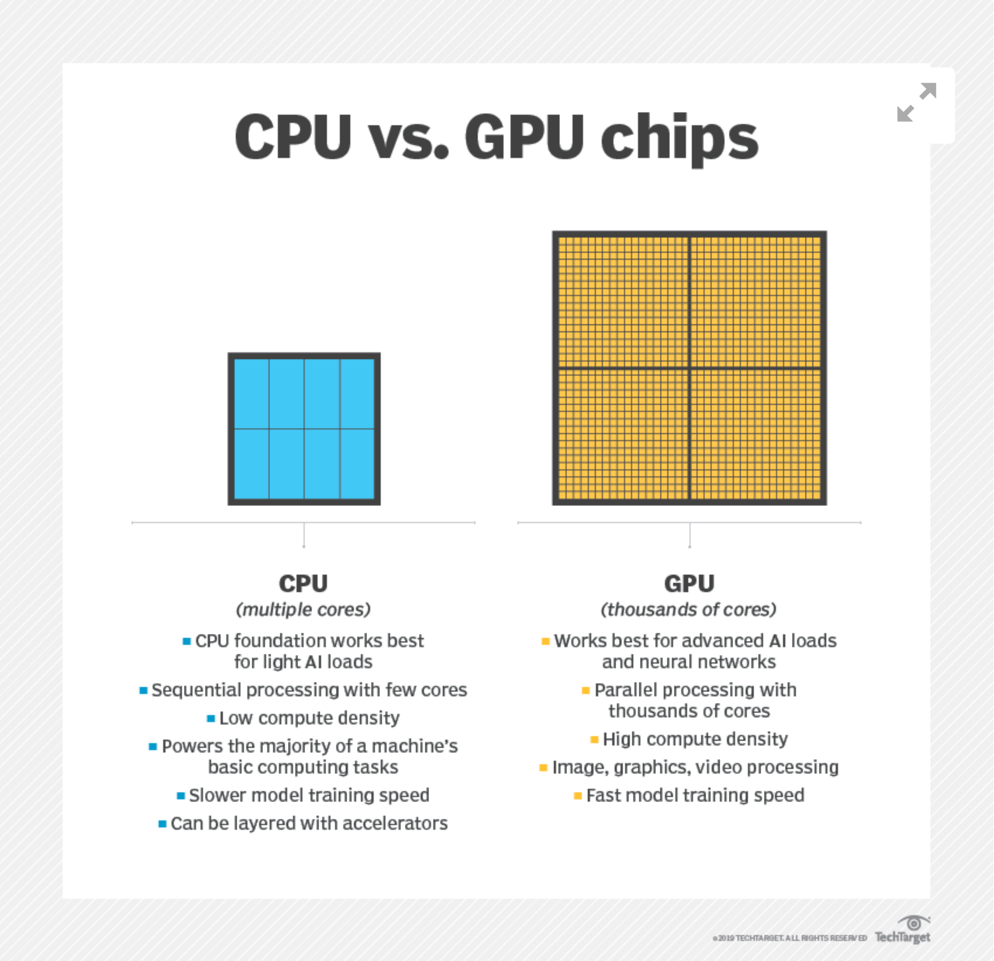
A bitwise operation operates on two-bit patterns of equal lengths by positionally matching their individual bits. For example, a logical AND (&) of each bit pair results in a 1 if both the first AND second bits are 1. If only one bit is a 1, the result is 0. AND can also be used to test individual bits in a bit string to see if they are 0 or 1.

A logical OR (|) operation functions differently from the AND operations. For each bit pair, the result is 1 if the first OR second bit is 1. If neither bit is 1, the result is 0.

A logical XOR (~) of each bit pair results in a 1 if the two bits are different, and 0 if they are the same (both zeros or both ones).

Logical NOT is represented as ^.

Left shift (<<), right shift (>>) and zero-fill right shift (>>>) bitwise operators are also known as bit shift operators.



CPU vs. GPU chips

CPU
(multiple cores)

- CPU foundation works best for light AI loads
- Sequential processing with few cores
- Low compute density
- Powers the majority of a machine's basic computing tasks
- Slower model training speed
- Can be layered with accelerators

GPU
(thousands of cores)

- Works best for advanced AI loads and neural networks
- Parallel processing with thousands of cores
- High compute density
- Image, graphics, video processing
- Fast model training speed

Arithmetic logic unit, part of a computer CPU, is where bitwise operators used to p
mathematical operations.

Bitwise operators

Bitwise operators are characters that represent actions (bitwise operations) to be performed on single bits. They operate at the binary level and perform operations on bit patterns that involve the manipulation of individual bits. Thus, unlike common logical operators like + or - which work with bytes or groups of bytes, bitwise operators can check each individual bit within a byte.

The most common bitwise operators used in C/C++ are given in the table below.

Operator	Name	Description	Application
&	Bitwise AND	Copies a bit to the result if it exists in both operands. The result is 1 only if both bits are 1.	To set up a mask to check the values of specific bits

Operator	Name	Description	Application
|	Bitwise OR	Copies a bit to the result if it exists in either operand. The result is 1 if either bit is 1.	To add two numbers if there is no carry involved
^	Bitwise Exclusive OR (XOR)	Copies a bit to the result if it exists in either operand. So, if one of the operands is TRUE, the result is TRUE. If neither operand is TRUE, the result is FALSE.	To toggle bits or swap two variables without using a third temporary variable To find specific types of numbers (e.g., odd) in a series of numbers (e.g., all even) To find nonrepeating elements To detect if two integers have opposite signs
~	Bitwise NOT	Also known as bitwise complement and bitwise inversion, it flips zeros into ones and ones into zeros.	To flip or invert bits
<<	Shift left	The left operand value is shifted left by the number of bits	To align bits

Operator	Name	Description	Application
		specified by the right operand.	
>>	Shift right	The left operand value is shifted right by the number of bits specified by the right operand.	To align bits

Multiple bitwise operators are used in bit manipulation. These operations happen very fast and optimize system performance and time complexity.

It's important to keep in mind that the left shift and right shift operators should not be used for negative numbers. Doing this can result in undefined behaviors in the programming language.

Also, bitwise operators should not be used in place of logical operators because they work differently. Logical operators consider non-zero operands as 1 and their result is either 0 or 1. In contrast, bitwise operators return an integer value.

The table below defines the JavaScript bitwise operators.

Operator	Name	Type	Action
&	Bitwise AND	Binary	If bits of both operands are ones, returns a one in each bit position
|	Bitwise OR	Binary	If bits of either operand are ones, returns a one in a

Operator	Name	Type	Action
			bit position
^	Bitwise XOR	Binary	If a single operand is a one, returns a one in a bit position
~	Bitwise NOT	Unary	Flips the bits in the operand
<<	Left shift	Binary	Shifts first operand a number of bits to the left as specified in the second operand, shifting in zeros from the right
>>	Right shift	Binary	Shifts first operand a number of bits to the right as specified in the second operand, and discards

Operator	Name	Type	Action
			displaced bits
>>>	Zero-fill right shift	Binary	Shifts first operand a number of bits to the right as specified in the second operand, discards displaced bits, and shifts in zeros from the left

Applications of bitwise operations and operators

There are many applications of bitwise operations and operators. For one, they are used in data compression where data is converted from one representation to another to reduce the amount of storage space required. Bitwise operations are also used in encryption algorithms to encrypt data and protect it from unauthorized use, manipulation or exfiltration.

The following are some other common applications:

- low-level programming for device drivers, memory allocators and compression software;
- maintaining large integer sets for search and optimization;
- ability to store multiple Boolean flags on limited memory devices;
- embedded software in chips and microcontrollers;
- communications where individual header bits carry sensitive or important information; and
- converting text cases, such as uppercase to lowercase or lowercase to uppercase.

Bitwise AND

The bitwise AND operator produces an output of 1 if the corresponding bits of both the operands are 1. If not, the output is 0.

Example 1: Bitwise AND operation of two one-bit operands.

Left operand	Right operand	Result
0	0	0
0	1	0
1	0	0
1	1	1

Example 2: Bitwise AND operation of two integers: 28 and 17; the & operator compares each binary digit of these integers.

Binary digits								
28	0	0	0	1	1	1	0	0
17	0	0	0	1	0	0	0	1
Are both digits 1?	No	No	No	Yes	No	No	No	No
Bitwise AND output	0	0	0	1	0	0	0	0

Thus: 28 & 17 (bitwise AND) = 00010000 (binary) = 16 (decimal).

Bitwise OR

The bitwise OR operator produces an output of 1 if either one of the corresponding bits is 1. Otherwise, the output is zero.

Example 1: The bitwise OR operation of two one-bit operands.

Left operand	Right operand	Result
0	0	0
0	1	1
1	0	1
1	1	1

Example 2: Let's consider the previous example of two integers: 28 and 17.

Binary digits								
28	0	0	0	1	1	1	0	0
17	0	0	0	1	0	0	0	1
Is either digit 1?	No	No	No	Yes	Yes	Yes	No	Yes
Bitwise OR output	0	0	0	1	1	1	0	1

Thus: 28 | 17 (bitwise OR) = 00011101 (binary) = 29 (decimal).

Bitwise exclusive OR (XOR)

The bitwise exclusive OR (XOR) operator returns 1 if the bits of both operands are opposite. Otherwise, it returns 0.

Example 1: The bitwise XOR operation of two one-bit operands.

Left operand	Right operand	Result
0	0	0

Left operand	Right operand	Result
0	1	1
1	0	1
1	1	0

Example 2: Let's see how bitwise XOR works for our two integers 28 and 17.

Binary digits								
28	0	0	0	1	1	1	0	0
17	0	0	0	1	0	0	0	1
Are the two digits opposite of each other?	No	No	No	No	Yes	Yes	No	Yes
Bitwise XOR output	0	0	0	0	1	1	0	1

Thus: 28 ^ 17 (bitwise XOR) = 00001101 (binary) = 13 (decimal).

Bitwise NOT

The bitwise NOT operator reverses the bits. Unlike other bitwise operators, it accepts only one operand.

Example: Let's consider the bitwise NOT operation of the integer 28.

Binary digits								
28	0	0	0	1	1	1	0	0
Bitwise NOT output	1	1	1	0	0	0	1	1

Thus: ~28 (bitwise NOT) = 11100011 (binary) = 227 (decimal).

Bitwise left shift

The bitwise left shift operator shifts the bits left by the bits specified by the right operand. The positions vacated by the left shift operator are filled with 0.

Example: Let's perform the bitwise left shift operation on the integer 6. Each bit will be shifted left by 1.

6 = 0110

6<<1 = 1100 (binary) = 12 (decimal)

Bitwise right shift

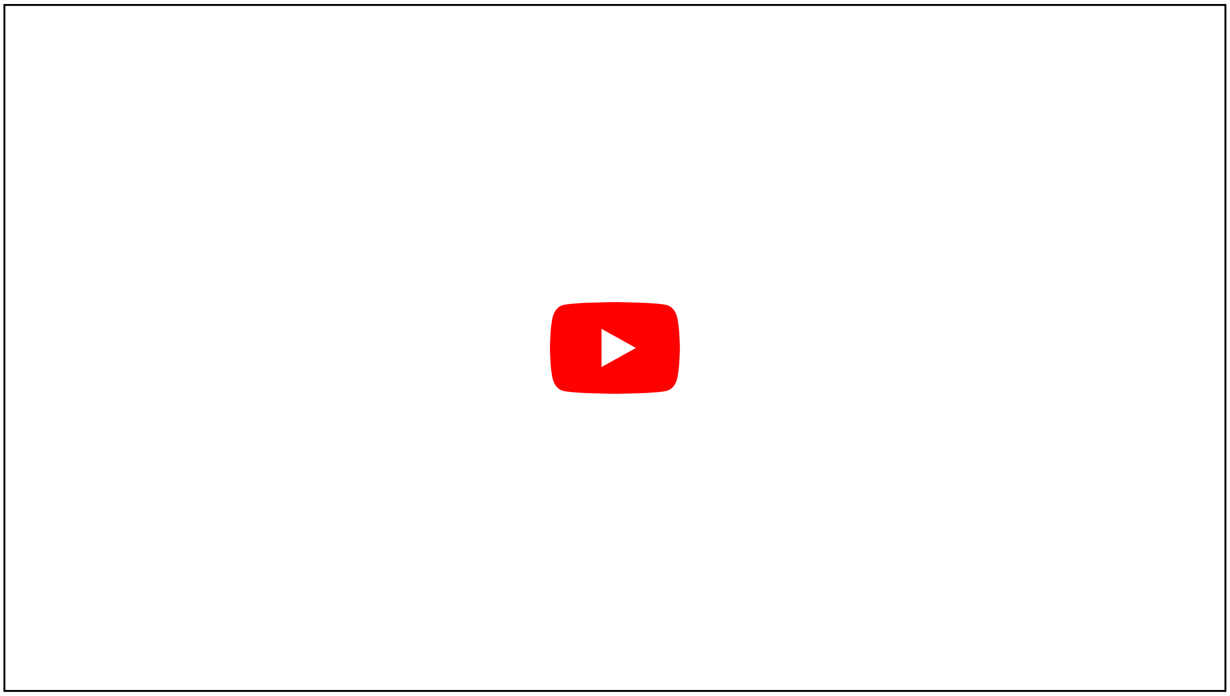
Like the left shift operator, the bitwise right shift operator shifts the bits right by the bits

0.

Example: Let's perform the right shift by two bits operations on the integer 8. Each bit will be shifted right by 2.

8 = 1000

8>>2 = 0010 (binary) = 2 (decimal)



See also: bit stuffing, logic gate (AND, OR, XOR, NOT, NAND, NOR and XNOR), How many bytes for..., classical computing, Advanced Business Application Programming

This was last updated in July 2022

Continue Reading About bitwise

- Binary and hexadecimal numbers explained for developers
- TB vs. GB: Is a terabyte bigger than a gigabyte?
- A brief breakdown of declarative vs. imperative programming
- A primer on DNA data storage and its potential uses
- 11 cloud programming languages developers need to know

Related Terms

What is machine learning? Guide, definition and examples

Machine learning is a branch of AI focused on building computer systems that learn from data.

See complete definition

What is natural language processing (NLP)?

Natural language processing (NLP) is the ability of a computer program to understand human language as it’s spoken and written --... See complete definition

What is project planning?

Project planning is a project management discipline that addresses how to complete a project in a certain time frame, usually ... See complete definition

Latest TechTarget **resources**

- NETWORKING ▶
- SECURITY
- CIO
- HR SOFTWARE
- CUSTOMER EXPERIENCE

Networking

What is a URL (Uniform Resource Locator)?

A URL (Uniform Resource Locator) is a unique identifier used to locate a resource on the internet.

What is FTP?

File Transfer Protocol (FTP) is a network protocol for transmitting files between computers over TCP/IP connections.

Browse by Topic | **Browse Resources**

About Us Meet The Editors Editorial Ethics Policy Contact Us Advertisers Business Partners Events Media Kit Corporate Site Reprints

Privacy Policy

Do Not Sell or Share My Personal Information