Phép toán thao tác bit

Bởi:

Wiki Pedia

Trong ngôn ngữ máy tính, các phép toán thao tác bit (tiếng Anh: bitwise operation) thực hiện tính toán (theo từng bit) trên một hoặc hai chuỗi bit, hoặc trên các số nhị phân. Với nhiều loại máy tính, việc thực thi các phép toán thao tác bit thường nhanh hơn so với khi thực thi các phép toán cộng, trừ, nhân, hoặc chia.

Các toán tử thao tác bit

Các toán tử thao tác bit (tiếng Anh: bitwise operator) là các toán tử được sử dụng chung với một hoặc hai số nhị phân để tạo ra một phép toán thao tác bit. Hầu hết các toán tử thao tác bit đều là các toán tử một hoặc hai ngôi.

NOT

Toán tử thao tác bit NOT còn được gọi là toán tử lấy phần bù (complement) là một toán tử một ngôi có nhiệm vụ phủ định luận lí từng bit của toán hạng của nó - tức đảo 0 thành 1 và ngược lại. Ví dụ, thực hiện phép toán NOT với số nhị phân 0111:

NOT 0111

= 1000

Trong các ngôn ngữ lập trình C, C++, Java, C#, toán tử thao tác bit NOT được biểu diễn bằng kí hiệu "~" (dấu ngã). Trong Pascal, toán tử này là "not". Ví dụ:

```
x = ~y; // C
```

Hay

```
x := not y; { Pascal }
```

Câu lệnh trên sẽ gán cho x giá trị "NOT y" - tức phần bù của y. Chú ý rằng, toán tử này không tương đương với toán tử luận lí "not" (biểu diễn bằng dấu chấm than "!" trong C/C++). Về vấn đề này, xin xem ở bài toán tử hoặc các bài về ngôn ngữ C/C++.

Toán tử NOT hữu dụng khi ta cần tìm bù 1 của một số nhị phân. Nó cũng có thể được sử dụng làm bước đầu tiên để tìm số bù 2.

OR

Toán tử thao tác bit OR là một toán tử hai ngôi, có nhiệm vụ thực hiện tính toán (trên từng bit) với hai chuỗi bit có cùng độ dài để tạo ra một chuỗi bit mới có cùng độ dài với hai chuỗi bit ban đầu. Trên mỗi cặp bit tương ứng nhau của hai toán hạng, toán tử OR sẽ trả về 1 nếu có một trong hai bit là 1, còn trong tất cả các trường hợp khác, OR sẽ tạo ra một bit 0. Ví dụ, thực hiện phép toán OR với hai số nhị phân 0101 và 0011:

0101

OR 0011

= 0111

Trong C, C++, Java, C#, toán tử thao tác bit OR được biểu diễn bằng kí hiệu "|" (vạch đứng). Trong Pascal, toán tử này là "or". Ví dụ:

x = y | z; // C

Hay:

x := y or z; { Pascal }

Câu lệnh trên sẽ gán cho x kết quả của "y OR z". Chú ý rằng toán tử này không tương đương với toán tử luận lí "or" (biểu diễn bằng cặp vạch đứng "||" trong C/C++). Về vấn đề này, xin xem ở bài toán tử hoặc các bài về ngôn ngữ C/C++.

Ứng dụng điển hình của toán tử thao tác bit OR là dùng để bật (set) một bit cụ thể trong một mẫu bit cho trước. Ví dụ: giả sử ta có mẩu bit 0010. Ta thấy, bit thứ nhất, thứ hai và thứ tư của mẩu chưa được bật (0), bây giờ, nếu ta muốn bật bit đầu tiên của mẩu, ta có thể sử dụng toán tử OR như minh họa sau:

0010

OR 1000

1010

Khi làm việc với các máy không có nhiều không gian bộ nhớ trống, các lập trình viên thường áp dụng kĩ thuật trên. Lúc đó, thay vì khai báo tám biến kiểu bool (C++) độc lập, người ta sử dụng từng bit riêng lẽ của một byte để biểu diễn giá trị cho tám biến đó.

XOR

Cũng giống OR, toán tử thao tác bit XOR (còn gọi là OR có loại trừ - exclusive OR) cũng là một toán tử hai ngôi, có nhiệm vụ thực hiện tính toán (trên từng bit) với hai chuỗi bit có cùng độ dài để tạo ra một chuỗi bit mới có cùng độ dài với hai chuỗi bit ban đầu. Tuy nhiên, trên mỗi cặp bit tương ứng nhau của hai toán hạng, toán tử XOR sẽ trả về 1 nếu chỉ có một trong hai bit là 1 (và bit còn lại là 0), ngược lại, XOR trả về bit 0. Ví dụ:

0101

XOR 0011

0110

(cách nhớ dễ nhất là: 2 bit giống nhau trả về 0, 2 bit khác nhau trả về 1) Trong C, C++, Java, C#, toán tử thao tác bit XOR được biểu diễn bằng kí hiệu "^" (dấu mũ). Trong Pascal, toán tử này là "xor". Ví dụ:

x = y ^ z; // C

Hay:

x := y xor z; { Pascal }

Câu lệnh trên sẽ gán cho x kết quả của "y XOR z". Các lập trình viên hợp ngữ (Assembly) thường sử dụng toán tử XOR để gán giá trị của một thanh ghi (register) về 0. Khi thực hiện phép toán XOR cho một mẫu bit với chính bản thân nó, mẫu nhị phân nhận được sẽ toàn bit 0. Trên nhiều kiến trúc máy tính, sử dụng XOR để gán 0 cho một thanh ghi sẽ được CPU xử lí nhanh hơn so với chuỗi thao tác tương ứng để nạp và lưu giá trị 0 vào thanh ghi.

Dịch chuyển và quay bit

Phép dịch bit được ký hiệu: >> (dịch phải) hoặc << (dịch trái) Ví dụ: 5 >> 1 = 2; 2 >> 1 = 1; 1 >> 1 = 0;

Giải thích: 5b = 0101 sau khi dịch 1 trở thành 0010 (=2d) và cứ tiếp tục như vậy.