Bitwise operations

By: Yehia M. Abu Eita

Outlines

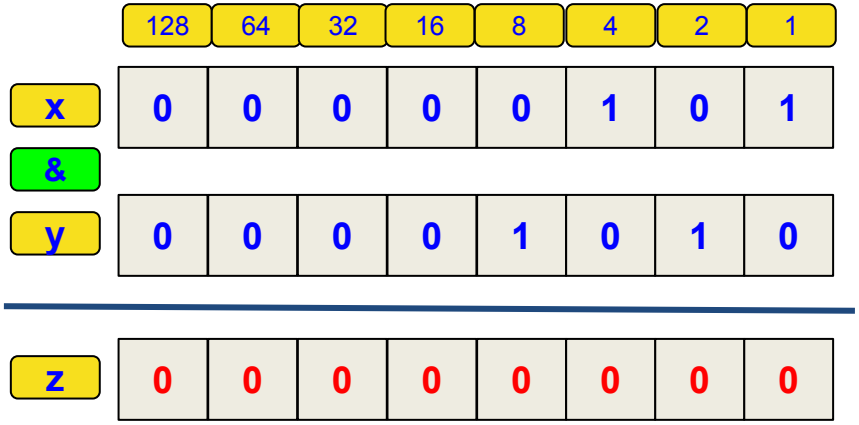
- Introduction
- Bitwise AND
- Bitwise OR
- Bitwise XOR
- Bitwise NOT
- Bitwise shift left and right

Introduction

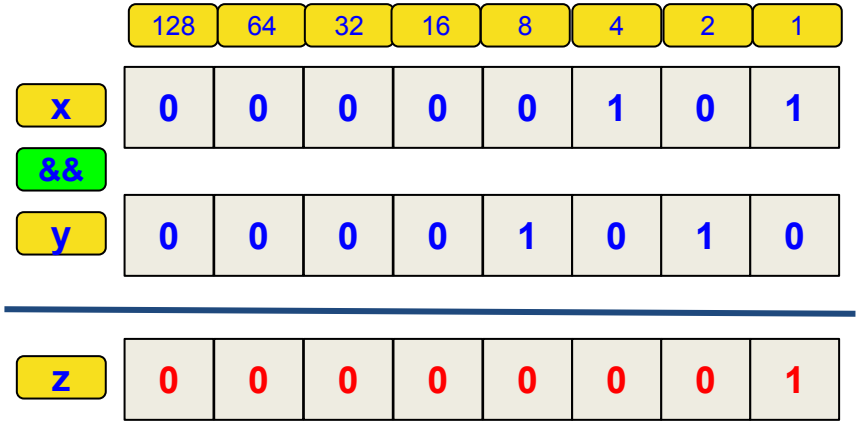
- The bitwise operators are used to **manipulate** values on the **bits level.**
- Bitwise operations are **widely used in embedded systems**.
- There are **six** bitwise operations you can use in C:
  - **AND (&)**
  - **OR (|)**
  - **XOR (^)**
  - **NOT (~)**
  - **Shift right (>>)**
  - **Shift left (<<)**

Bitwise AND

```
unsigned char x = 5, y = 10, z;
z = x & y;
```

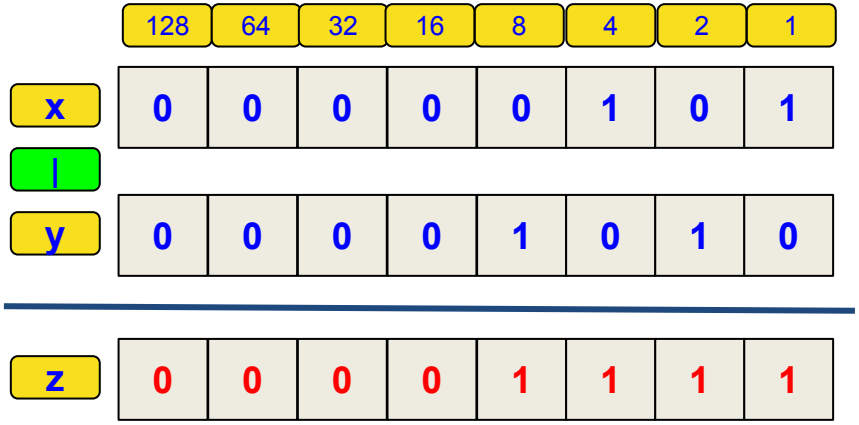



```
unsigned char x = 5, y = 10, z;
z = x && y; // Logical AND
```

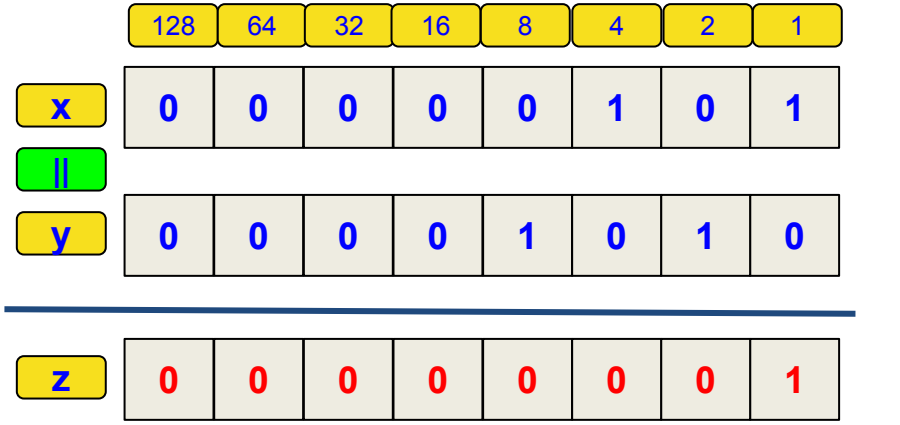


Bitwise OR

```
unsigned char x = 5, y = 10, z;
z = x | y;
```

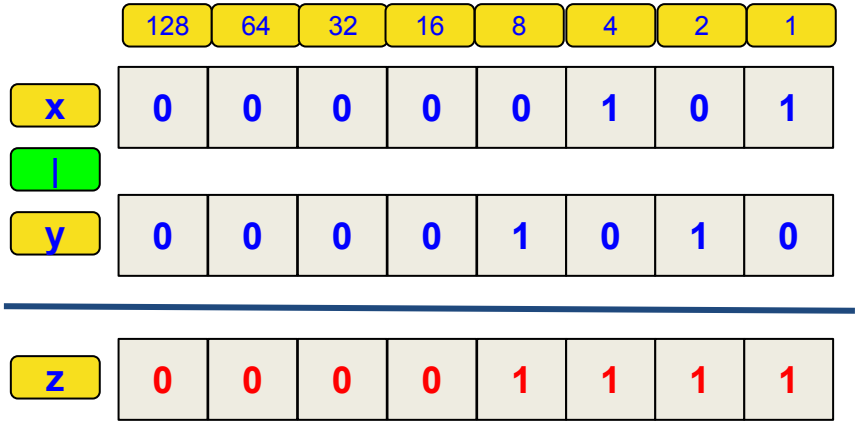



```
unsigned char x = 5, y = 10, z;
z = x || y; // Logical OR
```



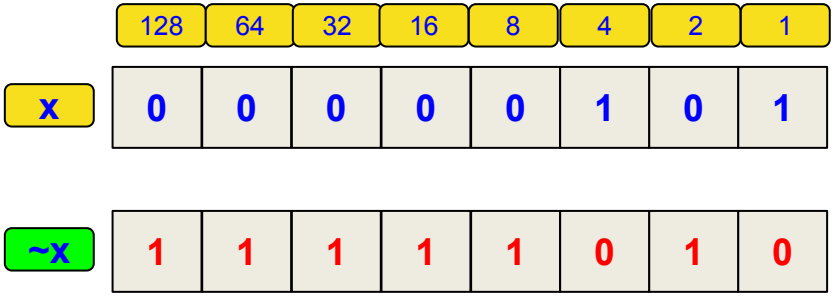
Bitwise XOR

```
unsigned char x = 5, y = 10, z;
z = x ^ y;
```

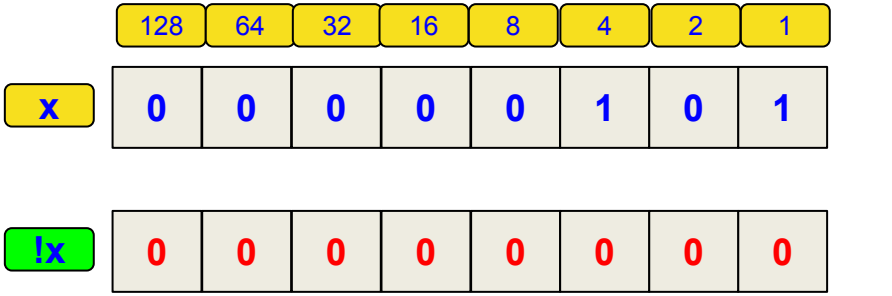


Bitwise NOT

```
unsigned char x = 5;
x = ~x;
```




```
unsigned char x = 5;
x = !x; // Logical NOT
```

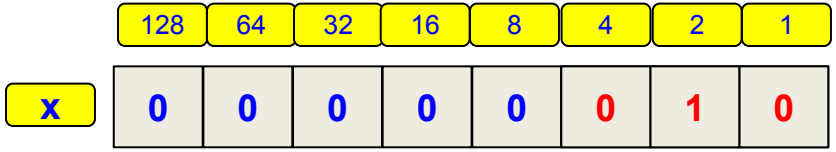


Bitwise Shift right

- Shifting right acts like an integer division by 2.

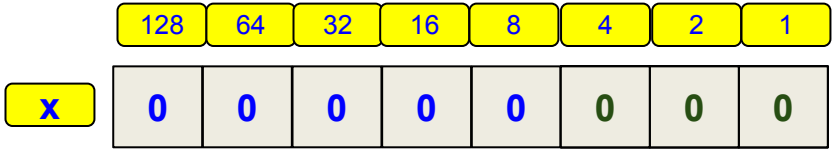
```
unsigned char x = 5, z;

z = x >> 1;
```




```
unsigned char x = 5, z;

z = x >> 3;
```

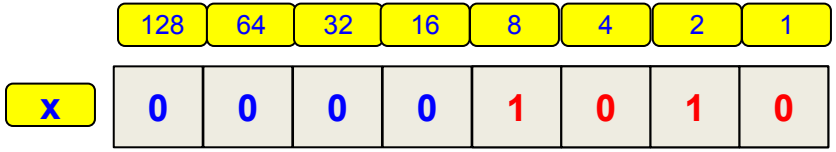


Bitwise Shift left

- Shifting left acts like an integer multiplication by 2.

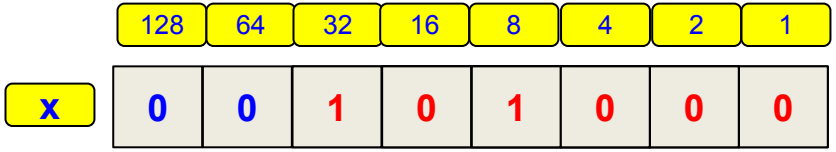
```
unsigned char x = 5, z;

z = x << 1;
```




```
unsigned char x = 5, z;

z = x << 3;
```



Summary

- You have learned that bitwise operation change values on the bit level.
- Now you can distinguish between bitwise and logical operators.