14. Bitwise Operators

[ECE10002/ITP10003] C Programming

Agenda

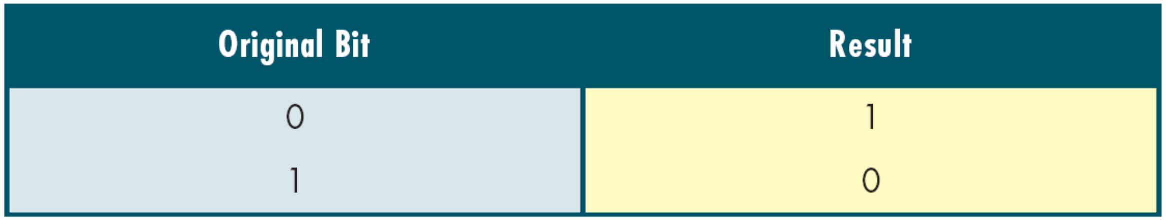
- Logical Operations
- Bitwise Logical Operators
- Bitwise Shift Operators

Logical Operations

- A AND B, A OR B

A	B	A AND B	A OR B
0	0	0	0
0	1	0	1
1	0	0	1
1	1	1	1

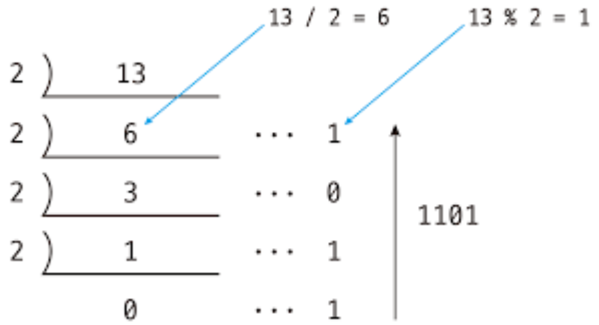
- NOT



Original Bit	Result
0	1
1	0

Decimal ⇔ Binary

- Decimal number ➔ binary number



- Binary number ➔ decimal number

$$\begin{aligned} 1101_2 &= 1 * 2^3 + 1 * 2^2 + 0 * 2^1 + 1 * 2^0 \\ &= \quad 8 \; + \quad 4 \; + \quad 0 \; + \quad 1 \; = \; 13 \end{aligned}$$

Representation of integers

- Representation of integers

  Ex) Assuming a short integer takes 2 bytes

  $599_{10} = 1001010111_2$

  → $\underline{0000\ 0010}\ \underline{0101\ 0111}_2 = \mathrm{0x0257}$

  $42083_{10} = 1010010001100011_2$

  → $\underline{1010\ 0100}\ \underline{0110\ 0011}_2 = \mathrm{0xA463}$

- Advanced topics
  - Representation of negative numbers
  - Representation of floating point numbers

Agenda

- Logical Operations
- Bitwise Logical Operators
- Bitwise Shift Operators

Bitwise Logical Operators

- Bitwise operators: logical operators that manipulate individual bits
  - Bitwise AND (&)

    Ex) 1100_2 & $1010_2 = 1000_2$
  - Bitwise OR (|)

    Ex) $1100_2 \mid 1010_2 = 1110_2$
  - Bitwise XOR (^)

    Ex) $1100_2 \wedge 1010_2 = 0110_2$
  - Bitwise NOT (~)

    Ex) $\sim 1100_2 = 0011_2$

Bitwise Logical Operators

- Example

```
short a = 0x0257;              // 0000 0010 0101 0111_2
short b = 0xA463;              // 1010 0100 0110 0011_2
printf("0x%hx & 0x%hx = 0x%hx₩n", a, b, a & b);
printf("0x%hx | 0x%hx = 0x%hx₩n", a, b, a | b);
printf("0x%hx ^ 0x%hx = 0x%hx₩n", a, b, a ^ b);
printf("~0x%hx = 0x%hx₩n", a, ~a);

0x257 & 0xa463 = 0x43      // 0000 0000 0100 0011_2
0x257 | 0xa463 = 0xa677    // 1010 0110 0111 0111_2
0x257 ^ 0xa463 = 0xa634    // 1010 0110 0011 0100_2
~0x257 = 0xfda8            // 1111 1101 1010 1000_2
```

Logical Operators vs. Bitwise Logical Operators

Example)

short a = 0x01; // $\underline{0000\ 0000}\ \underline{0000\ 0001}_2$

short b = 0x04; // $\underline{0000\ 0000}\ \underline{0000\ 0100}_2$

- Logical AND vs. bitwise AND
  - □ a && b == 1; // true && true == true
  - □ a & b == 0; // 0001_2 & 1000_2 == 0000_2
- Logical OR vs. bitwise OR
  - □ a || b == 1; // true || true == true
  - □ a | b == 5 // 0001_2 | 0100_2 == 0101_2

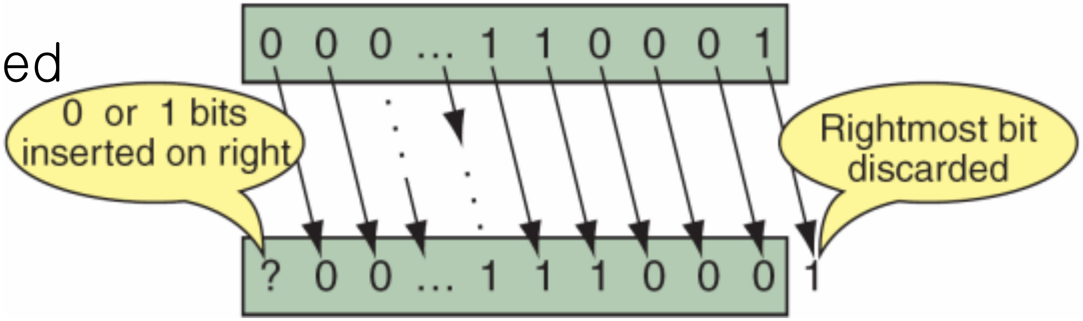
Agenda

- Logical Operations
- Bitwise Logical Operators
- Bitwise Shift Operators

Shift Operators

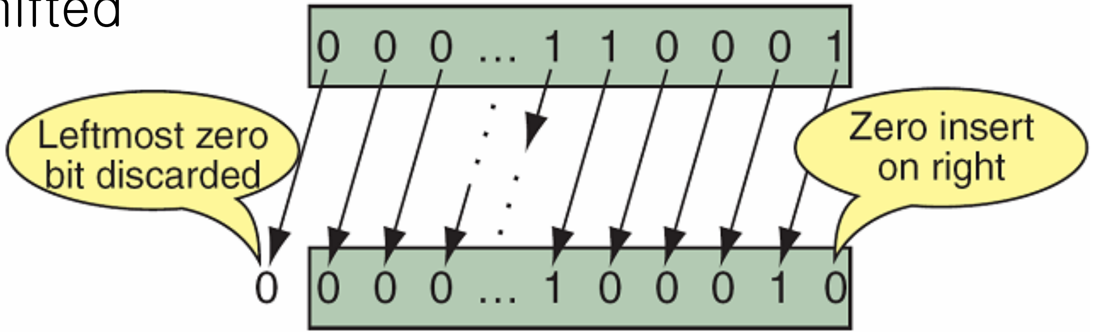
- Bitwise shift-right operator (a >> b)
  - a: value to be shifted
  - b: # of bits to be shifted

  Ex) x = x >> 1;



- Bitwise shift-left operator (a << b)
  - a: value to be shifted
  - b: # of bits to be shifted

  Ex) x = x << 1;



Shift Operators

Example)

- short x = 1; // 0000 0000 0000 0001
- x << 1 == 2; // 0000 0000 0000 0010
- x << 2 == 4; // 0000 0000 0000 0100
- x << 3 == 8; // 0000 0000 0000 1000
- x << 4 == 16; // 0000 0000 0001 0000

Compound Shift Operators

- Right-shift + assignment
  - >>=

```
Ex) x >>= 2;        // x = x >> 2;
```

- Left-shift + assignment
  - <<=

```
Ex) x <<= 3;        // x = x << 3;
```

Precedence and Associativity

Operators	Associativity
() [] -> .	left to right
! ~ ++ -- + - * & (type) sizeof	right to left
* / %	left to right
+ -	left to right
<< >>	left to right
< <= > >=	left to right
== !=	left to right
&	left to right
^	left to right
|	left to right
&&	left to right
||	left to right
?:	right to left
= += -= *= /= %= &= ^= |= <<= >>=	right to left
,	left to right

Exercises

- Read an integer. Then print it as a binary number.