Bitwise Operations

Instructor: Jeeho Ryoo

Bitwise operations

Logical NOT: !

A	!A
T	F
F	T

Bitwise operations

Logical NOT: !

A	!A
non-zero	0
zero	1

Bitwise operations

Bitwise NOT: ~

A	~A
00	11
01	10
10	01
11	00

Bitwise operations

Logical AND: &&

A	B	A && B
T	T	T
T	F	F
F	T	F
F	F	F

Bitwise operations

Logical AND: &&

A	B	A && B
non-zero	non-zero	1
non-zero	zero	0
zero	non-zero	0
zero	zero	0

Bitwise operations

Bitwise AND: &

A	B	A & B
1	1	1
1	0	0
0	1	0
0	0	0

Example: 1100 & 0101 = 0100

Bitwise operations

Logical OR: ||

A	B	A || B
T	T	T
T	F	T
F	T	T
F	F	F

Bitwise operations

Logical OR: ||

A	B	A || B
non-zero	non-zero	1
non-zero	zero	1
zero	non-zero	1
zero	zero	0

Bitwise operations

Bitwise OR: |

A	B	A | B
1	1	1
1	0	1
0	1	1
0	0	0

Example: 1100 | 0101 = 1101

Bitwise operations

Logical XOR:

A	B	A XOR B
T	T	F
T	F	T
F	T	T
F	F	F

Bitwise operations

Bitwise XOR: ^

A	B	A ^ B
1	1	0
1	0	1
0	1	1
0	0	0

Example: 1100 ^ 0101 = 1001

Bitwise operations

Two’s Complement Representation:

The left most bit is 0 for non-negative numbers. It will be 1 for negative numbers.

The Two’s complement representation of any number ***x*** with ***N*** bits is:

$$2^N - x$$

It can also be calculated using:

$$\sim x + 1$$

Bitwise operations

Examples (N = 4 bits):

0000 = $-0*2^3 + 0*2^2 + 0*2^1 + 0*2^0 = 0$

1000 = $-1*2^3 + 0*2^2 + 0*2^1 + 0*2^0 = -8$

1111 = $-1*2^3 + 1*2^2 + 1*2^1 + 1*2^0 = -1$

1010 = $-1*2^3 + 0*2^2 + 1*2^1 + 0*2^0 = -6$

0010 = $-0*2^3 + 0*2^2 + 1*2^1 + 0*2^0 = 2$

Bitwise operations

Two’s complement values to keep in mind:

00000…0 = 0

11111…1 = -1

01111…1 = Biggest positive integer with N bits.

10000…0 = Smallest negative integer with N bits.

Bitwise operations

Exercise:
Can you write a function that checks whether the third bit (from right) of a number is 1 or 0? Signature should be:

```
int thirdBitFromRight(int n);
```

Bitwise operations

Example:

```
#include <stdio.h>
int thirdBitFromRight(int n) {
    int mask = 4;
    return (n & mask) == 4;
}
void runTest(int n) {
    printf("n = %d, thirdBitFromRight = %d\n", n,
thirdBitFromRight(n));
}
int main() {
    runTest(4);
    runTest(15);
    runTest(0);
    runTest(11);
    runTest(-1);
    return 0;
}
```

Bitwise operations

Bit Masking:

A bit mask is an integer whose binary representation is intended to combine with another value using &, | or ^ to extract or set a particular bit or set of bits.

For example mask = 4 in the code from previous slide.

Bitwise operations

Another exercise:

Write a function that turns “on” the third bit (from right):

Practice Problems

MEDIUM – Write a function that takes a number and turns ***on*** the first and third binary digits (from right) for this number. Here are some examples:

$8 = (1000)_2 \rightarrow 13 = (1101)_2$
$0 = (0)_2 \rightarrow 5 = (101)_2$
$17 = (10001)_2 \rightarrow 21 = (10101)_2$
$29 = (11101)_2 \rightarrow 29 = (11101)_2$