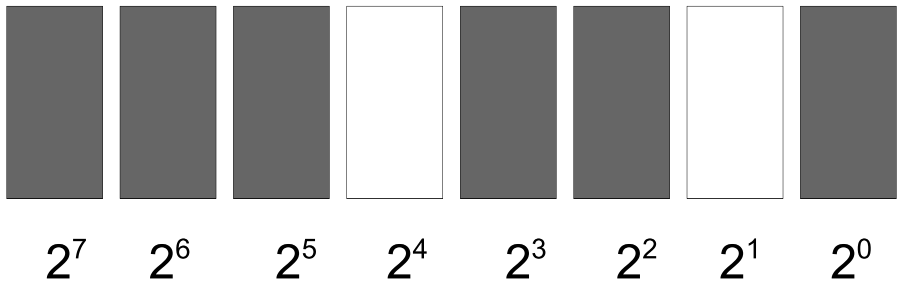
Binary Base

Introduction

- “A binary numeral system is a numeral system that represents numeric values using two symbols, usually 0 and 1. The binary system is used internally by all computers.” (wikipedia.org)

Bits & Bytes

- Binary numbers can be represented via a sequence of bits (binary digits) that can be represented by any mechanism capable of being in two mutually states (e.g. small magnetic field).



Binary & Decimal

- Converting a binary number into a decimal one is done by multiplying each one of the digits (starting with the rightmost digit representing 2^0) by 2^n.

  The 'n' stands for the digit position. The position of the rightmost digit is 0.

1	0	0	1	0	1	0	1	1	1
2^9	2^8	2^7	2^6	2^5	2^4	2^3	2^2	2^1	2^0

$1*2^9+1*2^6+1*2^4+1*2^2+1*2^1+1*2^0 = 512 +64+16+4+2+1=599$

Binary & Decimal

- Converting a decimal number into a binary one is done by dividing the decimal number by 2 while keeping the residual aside and continue with dividing by 2 each result we get.

25 / 2 = 12	1
12 / 2 = 6	0
6 / 2 = 3	0
3 / 2 = 1	1
1 / 2 = 0	1

$25_{10} = 11001_{2}$

Bitwise Operations

- NOT

  This is an unary operation that goes over each bit and change it into 1 (instead of 0) and into 0 (instead of 1).

  NOT 01111 = 10000

  NOT 10101 = 01010

  In many programming languages the bitwise NOT operator is '~'.

Bitwise Operations

- OR

  This operation takes two bit patterns with the same length and produce a new one with the same length by matching each one of the bits and performing a logical inclusive OR. For each pair the result is 1 if at least one of the two bits is 1. In all other cases the result is 0.

  110010 OR 111100 = 111110

  101 OR 100 = 101

  In many programming languages the bitwise NOT operator is '|'.

Bitwise Operations

- AND

  This operation takes two bit patterns with the same length and produce a new one with the same length by matching each one of the bits and performing a logical AND. For each pair the result is 1 if each one of the two bits is 1. In all other cases the result is 0.

  110010 AND 111100 = 110000

  101 OR 100 = 100

  In many programming languages the AND operator is '&'.

Bitwise Operations

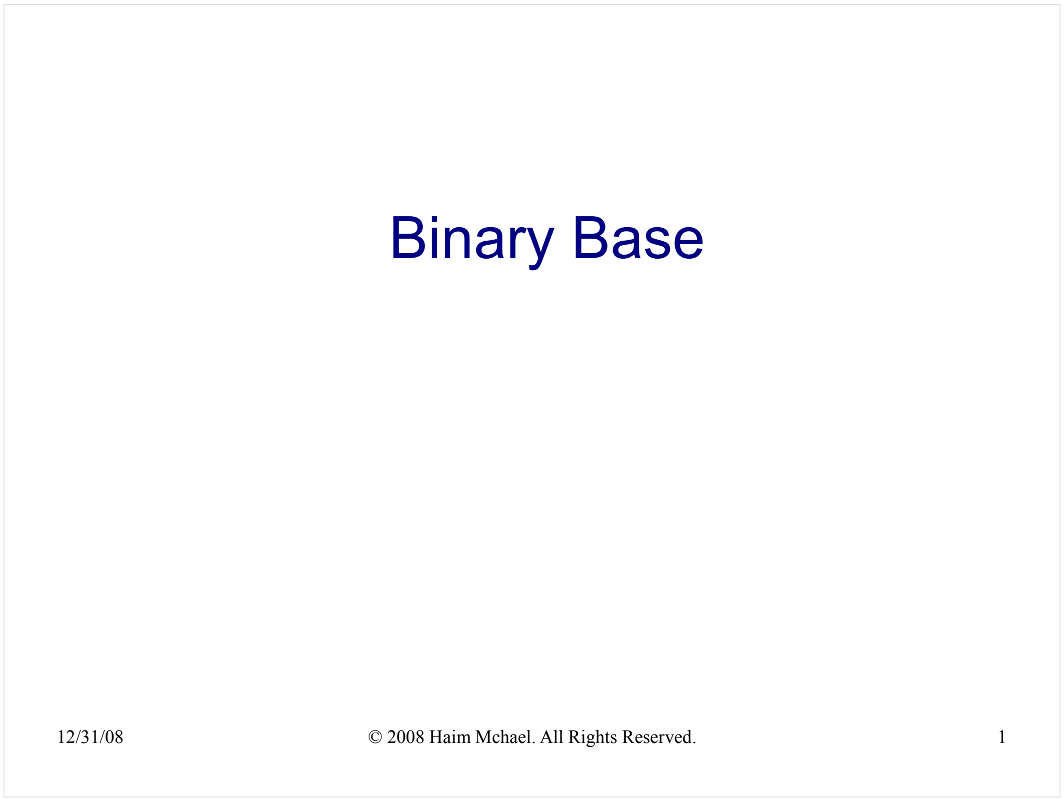
- XOR

  This operation takes two bit patterns with the same length and produce a new one with the same length by matching each one of the bits and performing a logical bitwise exclusive XOR operation. For each pair the result is 1 if the two bits are different. In all other cases the result is 0.

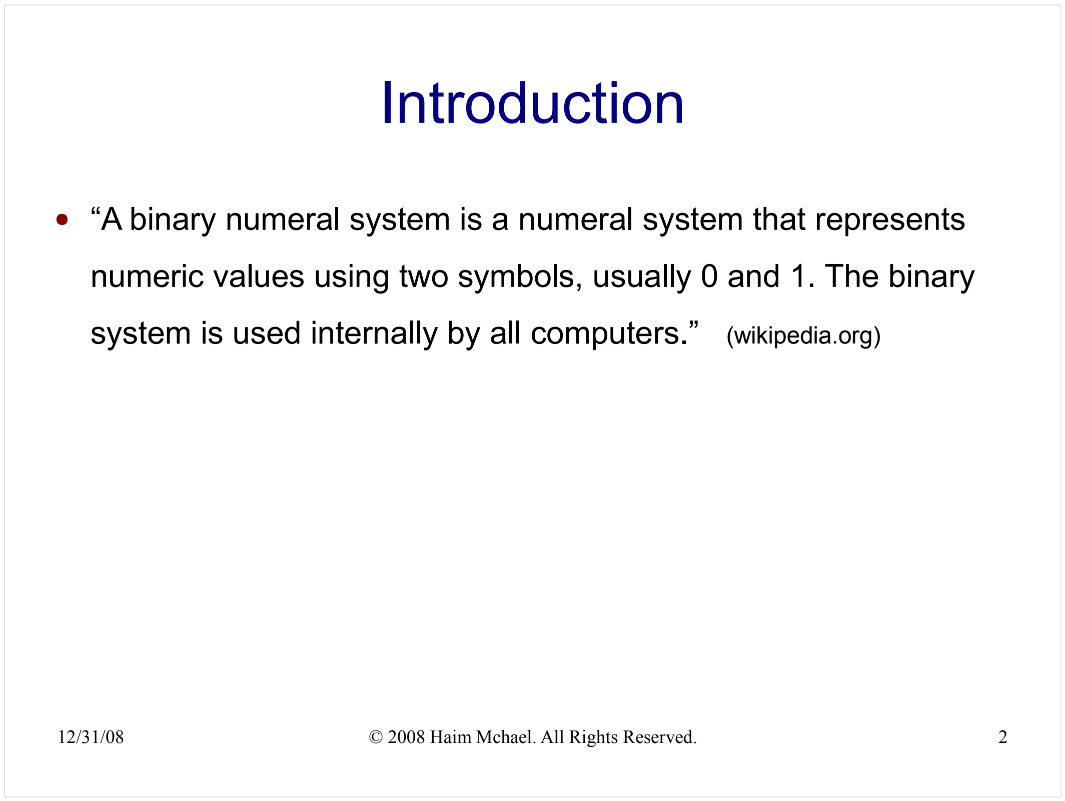
  110 XOR 111 = 001

  0101 OR 1100 = 1001

  In many programming languages the XOR operator is '^'.

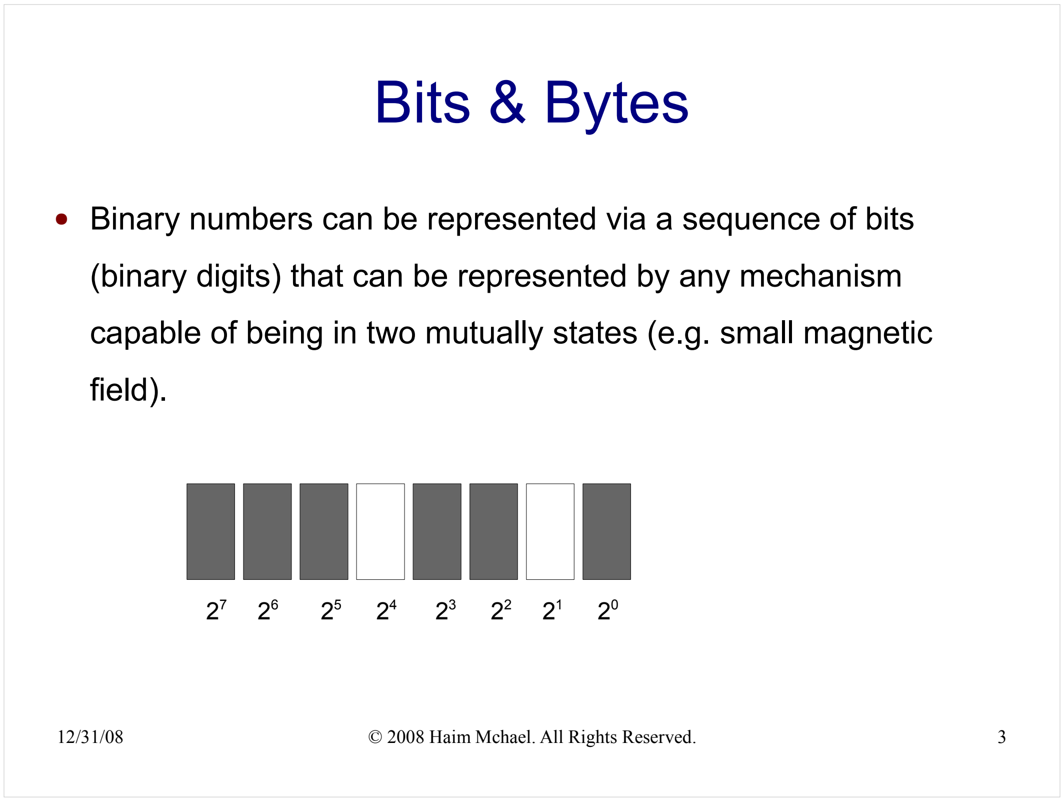


Binary Base



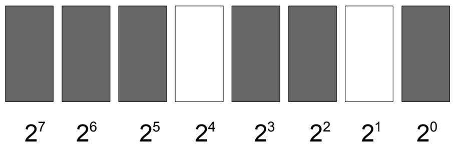
Introduction

- “A binary numeral system is a numeral system that represents numeric values using two symbols, usually 0 and 1. The binary system is used internally by all computers.” (wikipedia.org)

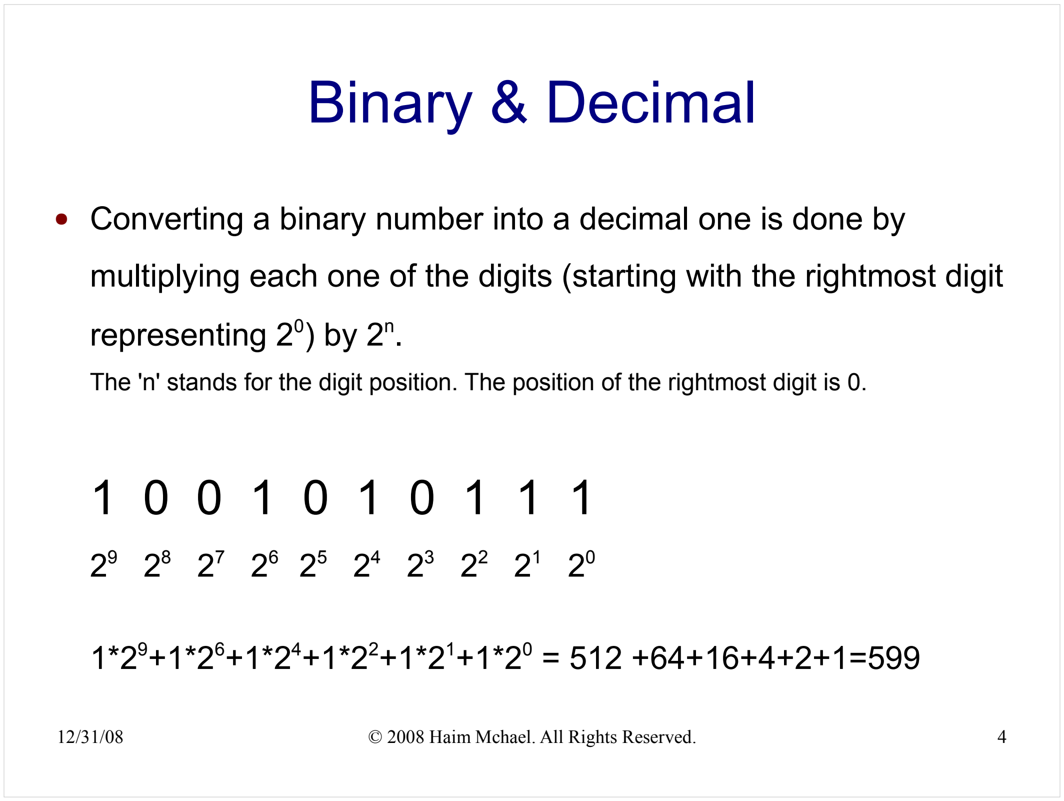


Bits & Bytes

- Binary numbers can be represented via a sequence of bits (binary digits) that can be represented by any mechanism capable of being in two mutually states (e.g. small magnetic field).



12/31/08 © 2008 Haim Mchael. All Rights Reserved. 3



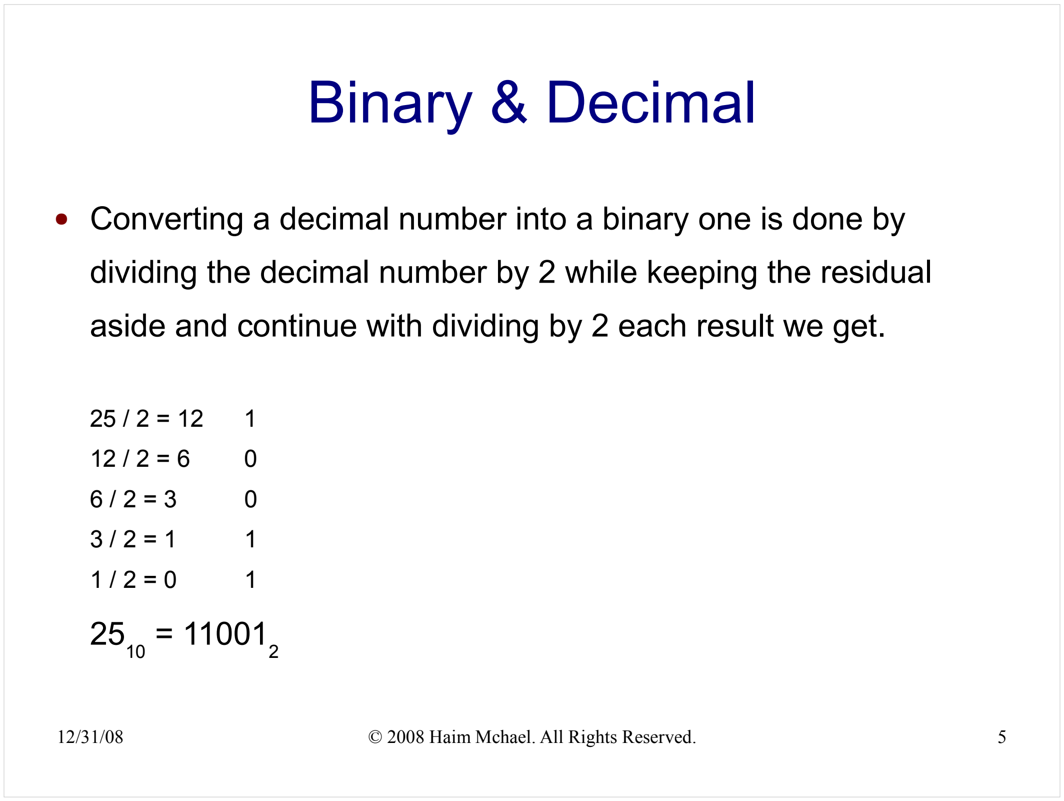
Binary & Decimal

- Converting a binary number into a decimal one is done by multiplying each one of the digits (starting with the rightmost digit representing 2^0) by 2^n.
  The 'n' stands for the digit position. The position of the rightmost digit is 0.

1	0	0	1	0	1	0	1	1	1
2^9	2^8	2^7	2^6	2^5	2^4	2^3	2^2	2^1	2^0

$1*2^9+1*2^6+1*2^4+1*2^2+1*2^1+1*2^0 = 512 +64+16+4+2+1=599$

12/31/08 © 2008 Haim Mchael. All Rights Reserved. 4

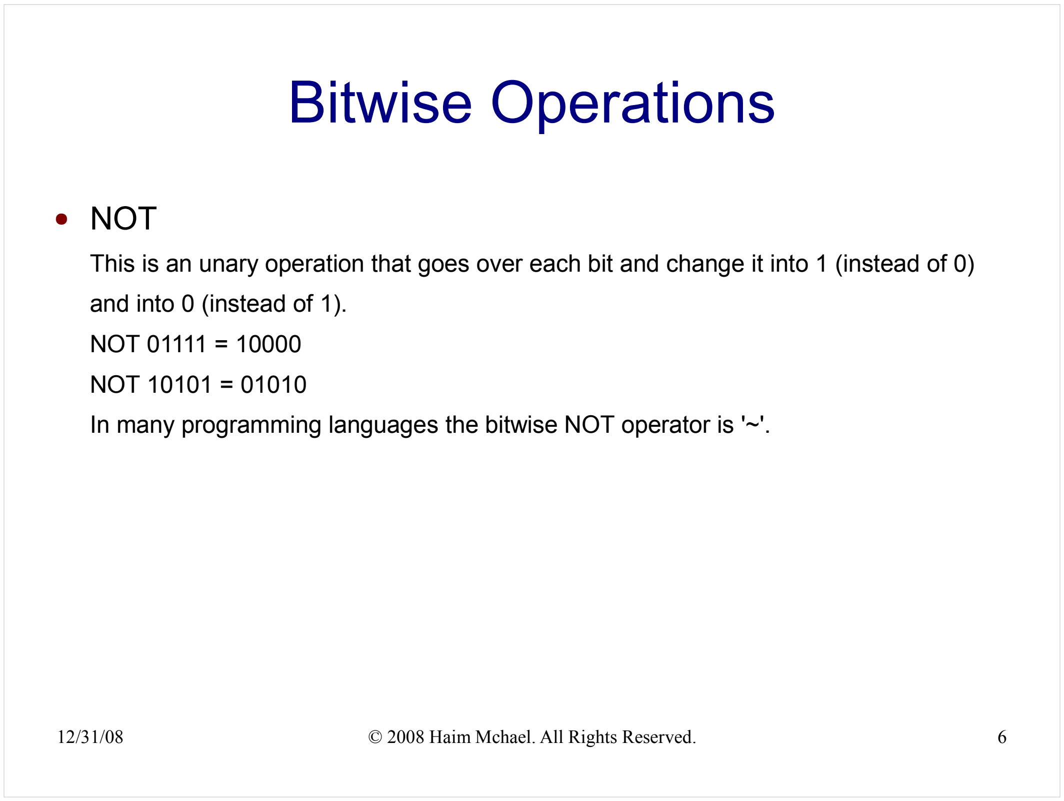


Binary & Decimal

- Converting a decimal number into a binary one is done by dividing the decimal number by 2 while keeping the residual aside and continue with dividing by 2 each result we get.

25 / 2 = 12	1
12 / 2 = 6	0
6 / 2 = 3	0
3 / 2 = 1	1
1 / 2 = 0	1

$25_{10} = 11001_2$



Bitwise Operations

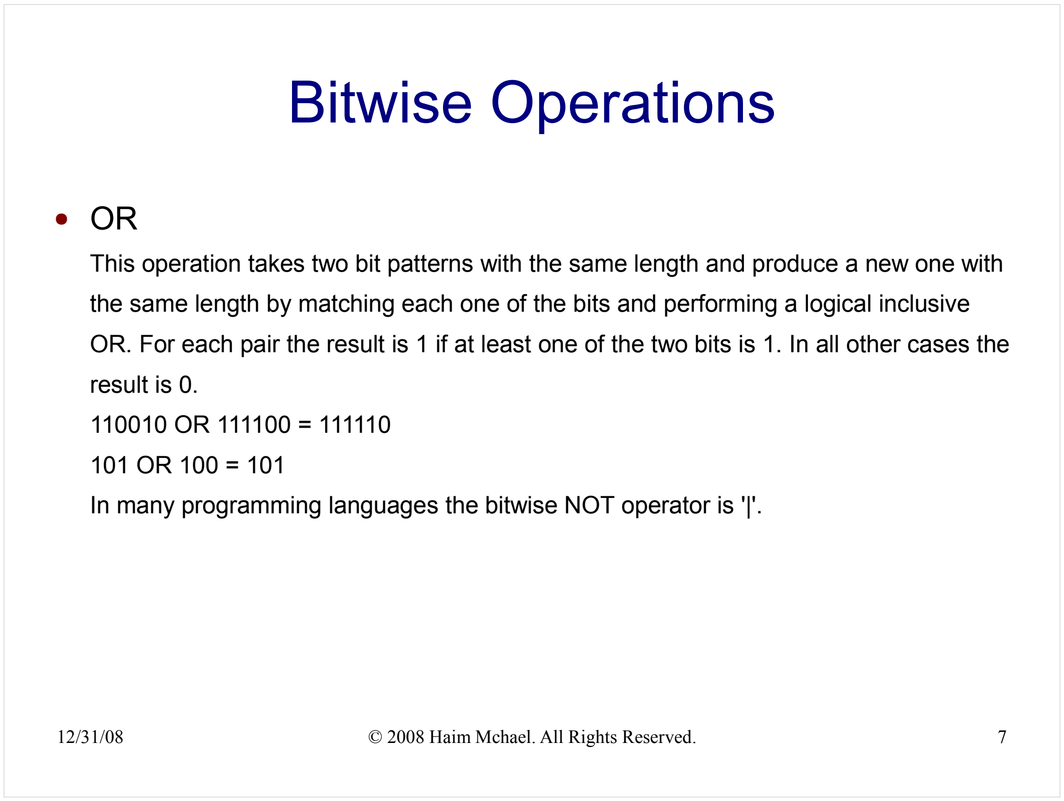
- NOT

  This is an unary operation that goes over each bit and change it into 1 (instead of 0) and into 0 (instead of 1).

  NOT 01111 = 10000

  NOT 10101 = 01010

  In many programming languages the bitwise NOT operator is '~'.



Bitwise Operations

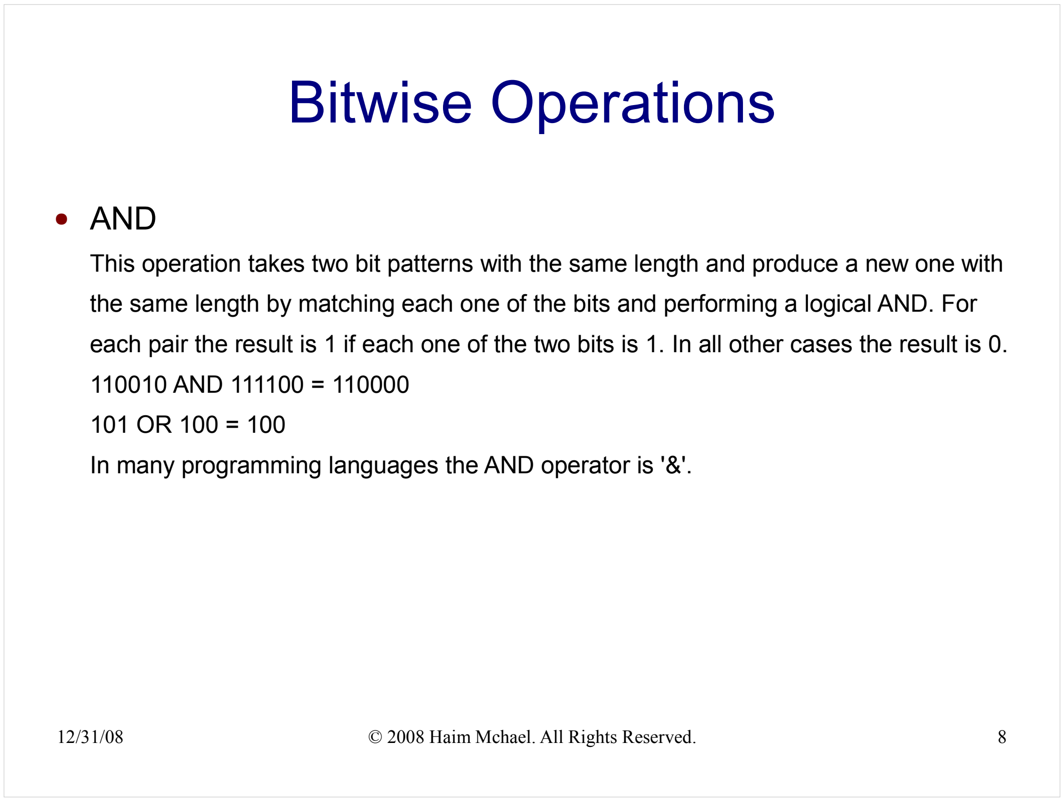
- OR

  This operation takes two bit patterns with the same length and produce a new one with the same length by matching each one of the bits and performing a logical inclusive OR. For each pair the result is 1 if at least one of the two bits is 1. In all other cases the result is 0.

  110010 OR 111100 = 111110

  101 OR 100 = 101

  In many programming languages the bitwise NOT operator is '|'.



Bitwise Operations

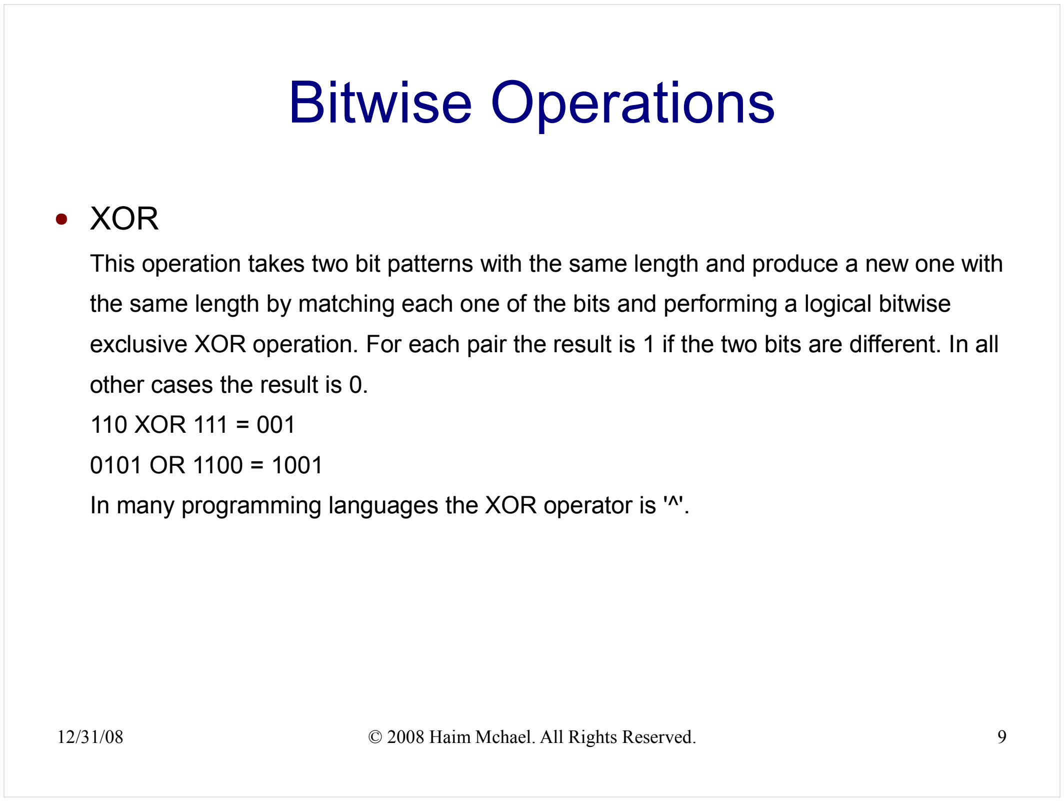
- AND

  This operation takes two bit patterns with the same length and produce a new one with the same length by matching each one of the bits and performing a logical AND. For each pair the result is 1 if each one of the two bits is 1. In all other cases the result is 0.

  110010 AND 111100 = 110000

  101 OR 100 = 100

  In many programming languages the AND operator is '&'.



Bitwise Operations

- XOR

  This operation takes two bit patterns with the same length and produce a new one with the same length by matching each one of the bits and performing a logical bitwise exclusive XOR operation. For each pair the result is 1 if the two bits are different. In all other cases the result is 0.

  110 XOR 111 = 001

  0101 OR 1100 = 1001

  In many programming languages the XOR operator is '^'.