LOGICAL OPERATIONS ON BITS

So far

* Bits interpreted as integers
* Arithmetic operations on bits

Today

* Bits interpreted as logical values
* Logical operations on bits
Logical values

True $\rightarrow$ 1 (convention)
False $\rightarrow$ 0

Logical operations

Also known as Boolean functions

A Boolean function $F$ with operands $A$ and $B$ can be specified by a truth table:

<table>
<thead>
<tr>
<th>A</th>
<th>B</th>
<th>F</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td>(Binary function)</td>
</tr>
</tbody>
</table>

We can also have unary operations:

<table>
<thead>
<tr>
<th>A</th>
<th>F</th>
</tr>
</thead>
</table>
NOT function
**AND function**

Truth table: 

<table>
<thead>
<tr>
<th>A</th>
<th>B</th>
<th>AND</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

When applied to bit patterns, do it bit by bit:

\[ 11010100 \text{ AND } \]

\[ 11110000 \]

AND allows masking out bits to isolate a particular location while ignoring other bits. To do that, we use a bitmask.

**OR function (inclusive OR)**

Truth table: 

<table>
<thead>
<tr>
<th>A</th>
<th>B</th>
<th>OR</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

When applied to bit patterns, do it bit by bit:

\[ 11011010 \text{ OR } \]

\[ 11110000 \]
Logical operations on bits

1 1 0 1 1 0 1 0
OR
1 1 1 1 0 0 0 0

XOR function (exclusive OR)

Truth table:

<table>
<thead>
<tr>
<th>A</th>
<th>B</th>
<th>XOR</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>1</td>
<td>0</td>
</tr>
</tbody>
</table>

Symbol:

When applied to bit patterns, do it bit by bit:

0 1 0 1 XOR 0 1 0 1 XOR
0 1 0 1     0 1 1 1

Special use: two bit patterns are identical if XOR output is all zeros

©2017 Juan Jose Jaramillo. All rights reserved.
We will only use the notation highlighted in **RED**
Logical completeness

How many Boolean functions exist for $N$ variables?

* For one variable ($x$):

* For two variables ($x, y$):
* For $N$ variables:

If there are so many functions, why NOT, AND, OR are so important?

Because we can implement any Boolean function with only those 3 functions! (Logical completeness)
Question: can you memorize the following 15-bit pattern in ten seconds?

0 1 1 1 0 0 0 1 0 1 0 1 1 1 1
Hexadecimal notation

1) Group bits in sets of 4 bits each
2) With 4 bits we can represent

\[ 2^4 = 16 \text{ numbers} \]

Use digits 0-9 and letters A-F to represent a set of 4 bits

Example: for the bit pattern above

1) 0111 1000 1010 1111

2) \times \_ \_ \_ \_ \_ \_ \_