Bit-Wise Operators
Bit-Wise Operators

If A and B are bytes, what does this code mean?

\[ C = A \& B; \]
Bit-Wise Operators

If A and B are bytes, what does this code mean?

\[ C = A \& B; \]

The corresponding bits of A and B are ANDed together.
Bit-Wise AND

\[
\begin{array}{c}
\text{0 1 0 1 1 1 1 0} & \text{A} \\
\text{1 0 0 1 1 0 1 1} & \text{B} \\
\hline
\text{?} & \text{C = A \& B}
\end{array}
\]
Bit-Wise AND

\[
\begin{array}{ll}
0 & 1 & 0 & 1 & 1 & 1 & 1 & 0 & \text{A} \\
1 & 0 & 0 & 1 & 1 & 0 & 1 & 1 & \text{B} \\
\hline
1 & 0 & 0 & 1 & 1 & 0 & 1 & 1 & \text{C} = \text{A} \& \text{B}
\end{array}
\]
Bit-Wise AND

A

B

C = A & B
Bit-Wise AND

A = 01011110
B = 10011011

C = A & B = 10
Bit-Wise AND

01011110 \quad A

10011011 \quad B

00011010 \quad C = A \& B
Logical AND

\[ 0 \ 1 \ 0 \ 1 \ 1 \ 1 \ 1 \ 0 \quad A \]
\[ 1 \ 0 \ 0 \ 1 \ 1 \ 0 \ 1 \ 1 \quad B \]

\[ ??? \quad C = A \land \land B \]
Logical AND

\[ A = 01011110 \]
\[ B = 10011011 \]
\[ C = A \&\& B \]

\[ C \]

true
Logical AND

\[
\begin{array}{cccccccc}
0 & 1 & 0 & 1 & 1 & 1 & 1 & 0 \\
\text{A} & \text{true} \\
1 & 0 & 0 & 1 & 1 & 0 & 1 & 1 \\
\text{B} & \text{true} \\
\hline
\text{C} = A \&\& B
\end{array}
\]
Logical AND

\[ \begin{array}{ccccccccc}
0 & 1 & 0 & 1 & 1 & 1 & 1 & 0 \\
A \\
\text{true}
\end{array} \quad \begin{array}{ccccccccc}
1 & 0 & 0 & 1 & 1 & 0 & 1 & 1 \\
B \\
\text{true}
\end{array} \quad \begin{array}{ccccccccc}
\text{false} \\
\text{true} \\
\text{true} \\
C = A \&\& B
\end{array} \]
Logical AND

\[ A = 01011110 \]
\[ B = 10011011 \]

\[ C = A \land B = 00000001 \]

NOTE: we are assuming an 8-bit value
Representing Logical Values

Most of the time, we represent logical values using a multi-bit value. (e.g., using 8 or 16 bits). The rules are:

• A value of zero is interpreted as \textit{false}
• A non-zero value is interpreted as \textit{true}
Representing Logical Values

A logical operator will give a result of **true** or **false**:

- **false** is represented with a value of zero (0)
- **true** is represented with a value of one (1)
Other Operators

<table>
<thead>
<tr>
<th>LOGICAL</th>
<th>Bit-Wise</th>
</tr>
</thead>
<tbody>
<tr>
<td>• OR:</td>
<td></td>
</tr>
<tr>
<td>• NOT:</td>
<td>!</td>
</tr>
<tr>
<td>• XOR:</td>
<td>^</td>
</tr>
<tr>
<td>• Shift left:</td>
<td>&lt;&lt;</td>
</tr>
<tr>
<td>• Shift right:</td>
<td>&gt;&gt;</td>
</tr>
</tbody>
</table>

When coding: keep this distinction straight
Putting the Bit-Wise Operators to Work: Bit Manipulation

Assume a variable A is declared as such:

```c
uint8_t A;
```

What is the code that allows us to set bit 2 of A to 1? (we start counting bits from 0)
Bit Manipulation

What is the code that allows us to set bit 2 of A to 1? (we start counting bits from 0)

\[ A = A \mid 4; \]
Bit Manipulation

What is the code that allows us to set bit 2 of A to 0?
Bit Manipulation

What is the code that allows us to set bit 2 of A to 0?

\[ A = A \& 0xFB; \]

or

\[ A = A \& \sim 4; \]
Bit Shifting

```c
uint8_t A = 0x5A;
uint8_t B = A << 2;
uint8_t C = A >> 5;
```

What are the values of B and C?
What mathematical operations have we performed?