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

\[
\begin{array}{cccccccc}
0 & 1 & 0 & 1 & 1 & 1 & 1 & 0 \\
1 & 0 & 0 & 1 & 1 & 0 & 1 & 1 \\
\hline
? & & & & & & & \\
\end{array}
\]

C = A \& B
Bit-Wise AND

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

C = A & B
Bit-Wise AND

\[ A = 01011110 \]
\[ B = 10011011 \]
\[ C = A \& B = 00001000 \]
Bit-Wise AND

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

0 1 0 1 1 1 1 0  \hspace{1cm} A

1 0 0 1 1 0 1 1 \hspace{1cm} B

\hline

0 0 0 1 1 0 1 0 \hspace{1cm} C = A \& B
Logical AND

A

B

C = A && B
Logical AND

$$0 \ 1 \ 0 \ 1 \ 1 \ 1 \ 1 \ 0$$  \hspace{0.5cm} A

$$1 \ 0 \ 0 \ 1 \ 1 \ 0 \ 1 \ 1$$  \hspace{0.5cm} B

$$\text{true}$$

$$\text{C} = \ A \ & \ & \ B$$
Logical AND

\[
\begin{array}{cccccccc}
0 & 1 & 0 & 1 & 1 & 1 & 1 & 0 \\
1 & 0 & 0 & 1 & 1 & 0 & 1 & 1 \\
\end{array}
\]

A

\text{true}

B

\text{true}

C = A \&\& B
Logical AND

\[
\begin{array}{cccccccc}
0 & 1 & 0 & 1 & 1 & 1 & 1 & 0 \\
\hline
1 & 0 & 0 & 1 & 1 & 0 & 1 & 1 \\
\end{array}
\]

\[
C = A \land B
\]

true
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)

\[ A = A | 4; \]
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?