

# 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

0 1 0 1 1 1 1 0

A

1 0 0 1 1 0 1 1

B

---

?

C = A & B

# Bit-Wise AND

0 1 0 1 1 1 1 0

A

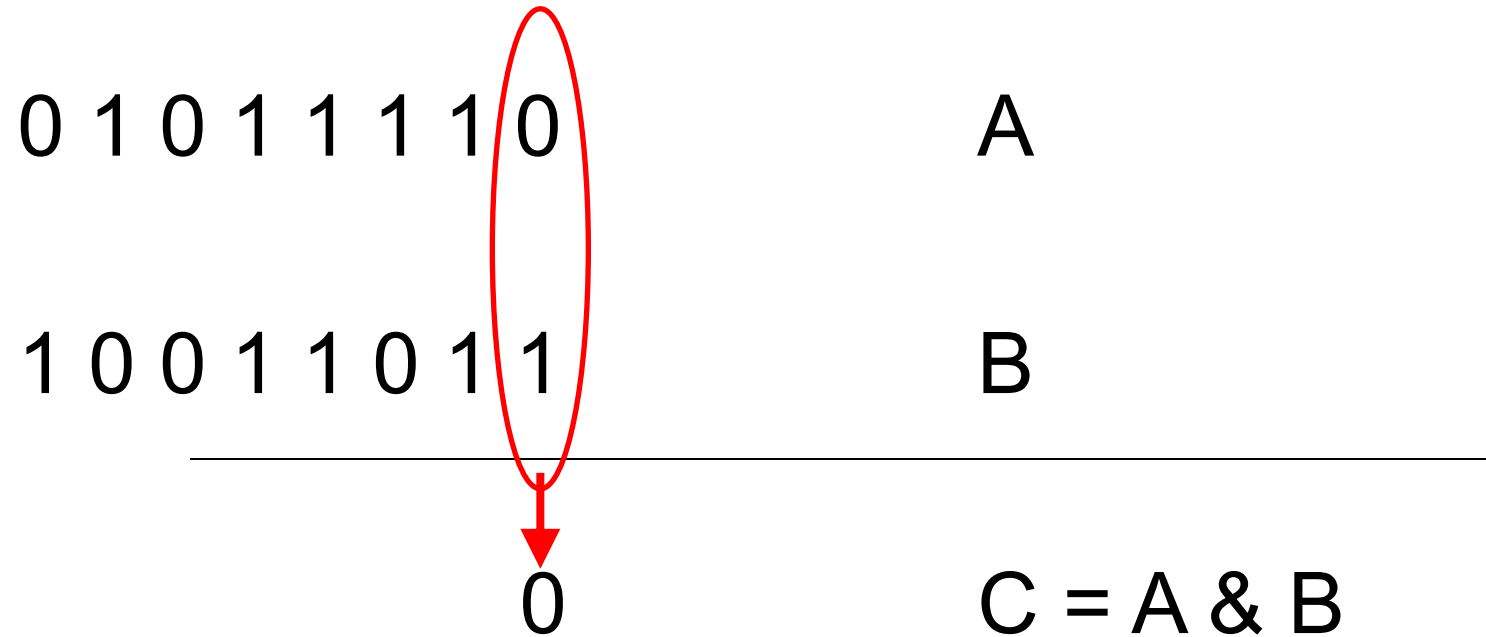
1 0 0 1 1 0 1 1

B

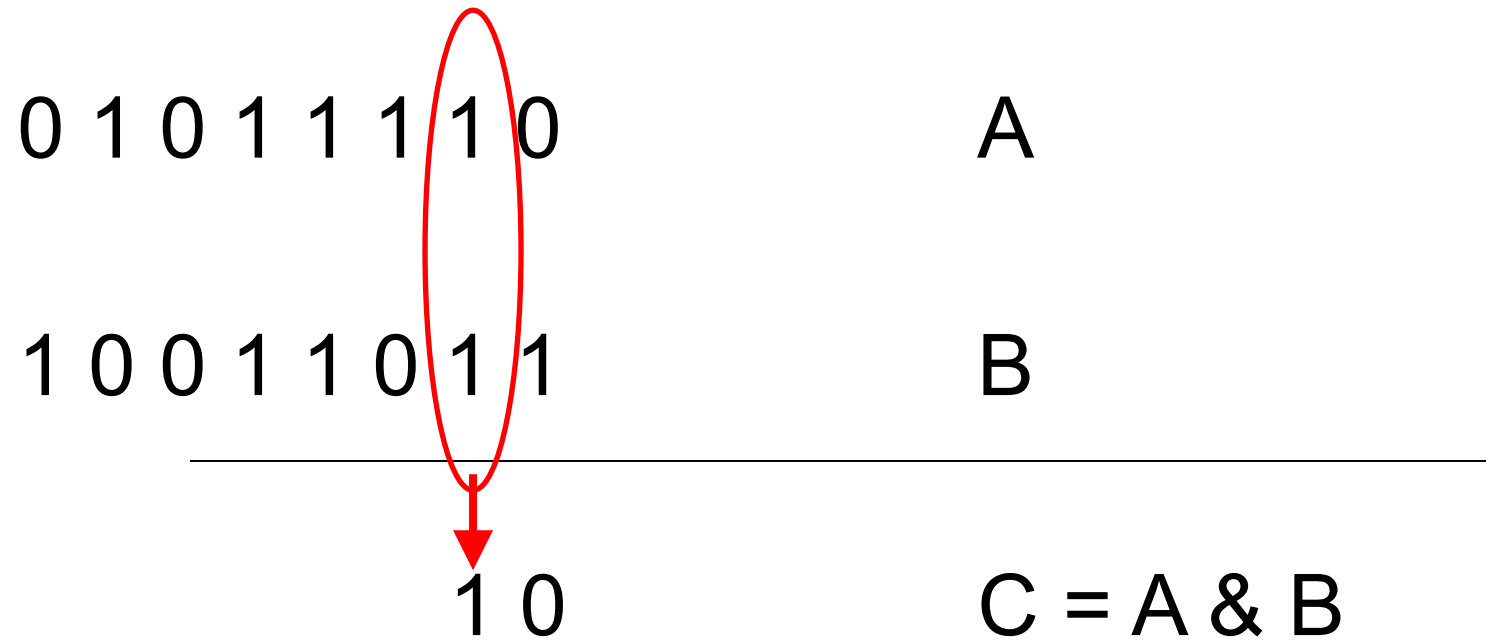
---

C = A & B

# Bit-Wise AND



# Bit-Wise AND



# Bit-Wise AND

0 1 0 1 1 1 1 0

A

1 0 0 1 1 0 1 1

B

---

0 0 0 1 1 0 1 0

C = A & B



# Logical AND

0 1 0 1 1 1 1 0

A

1 0 0 1 1 0 1 1

B

---


???

C = A && B



# Logical AND

0 1 0 1 1 1 1 0

 A  
true

1 0 0 1 1 0 1 1

B


---

???


C = A && B

# Logical AND

0 1 0 1 1 1 1 0

 A  
true

1 0 0 1 1 0 1 1


 B  
true

???


C = A && B

# Logical AND

0 1 0 1 1 1 1 0


 A  
true

1 0 0 1 1 0 1 1

 B  
true

???

---


 true  
C = A && B

# Logical AND

0 1 0 1 1 1 1 0



 **A**  
true

1 0 0 1 1 0 1 1

 **B**  
true

---

0 0 0 0 0 0 0 1

  
 **C = A && B**

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 ***false***
- A non-zero value is interpreted as ***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

	LOGICAL	Bit-Wise
• OR:		
• NOT:	!	~
• XOR:		^
• Shift left:		<<
• Shift right:		>>

When coding: keep this distinction straight



# Putting the Bit-Wise Operators to Work: Bit Manipulation

Assume a variable A is declared as such:

```
u_int8_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 & ~4;
```

# Bit Shifting

```
u_int8_t A = 0x5A;
```

```
u_int8_t B = A << 2;
```

```
u_int8_t C = A >> 5;
```

What are the values of B and C?

What mathematical operations have we performed?