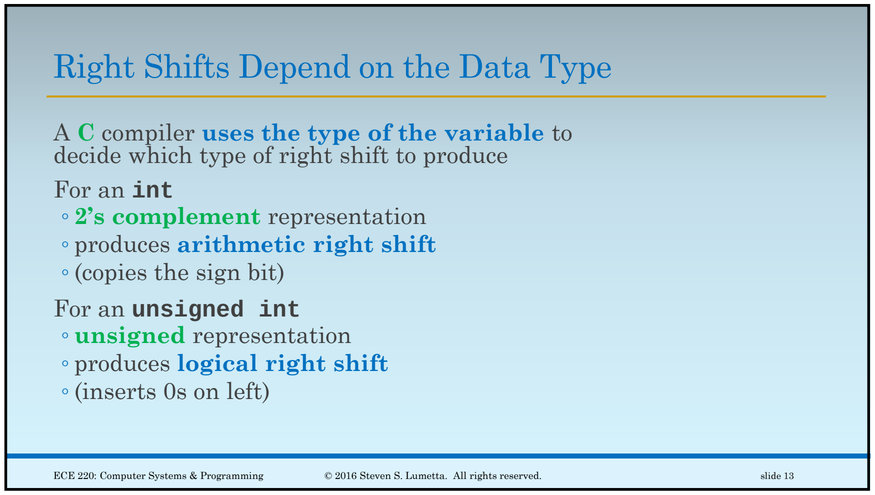
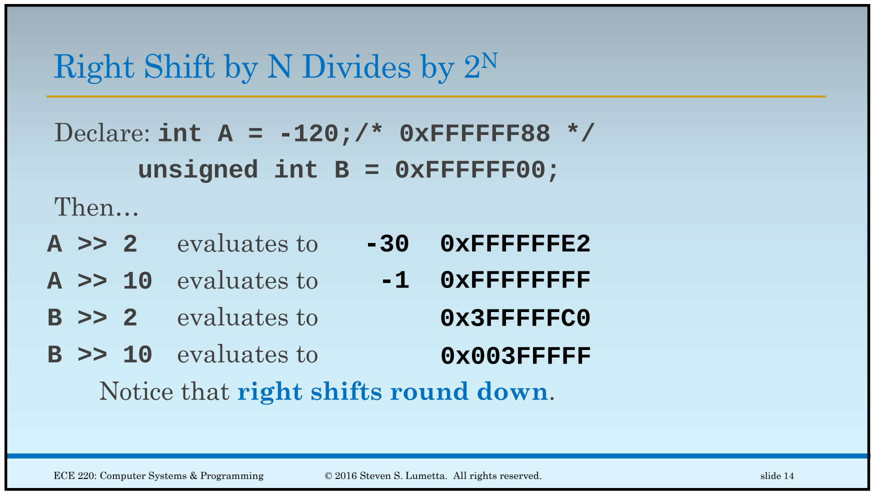


Right Shifts Depend on the Data Type

A **C** compiler **uses the type of the variable** to decide which type of right shift to produce

For an `int`
- **2's complement** representation
- produces **arithmetic right shift**
- (copies the sign bit)

For an `unsigned int`
- **unsigned** representation
- produces **logical right shift**
- (inserts 0s on left)



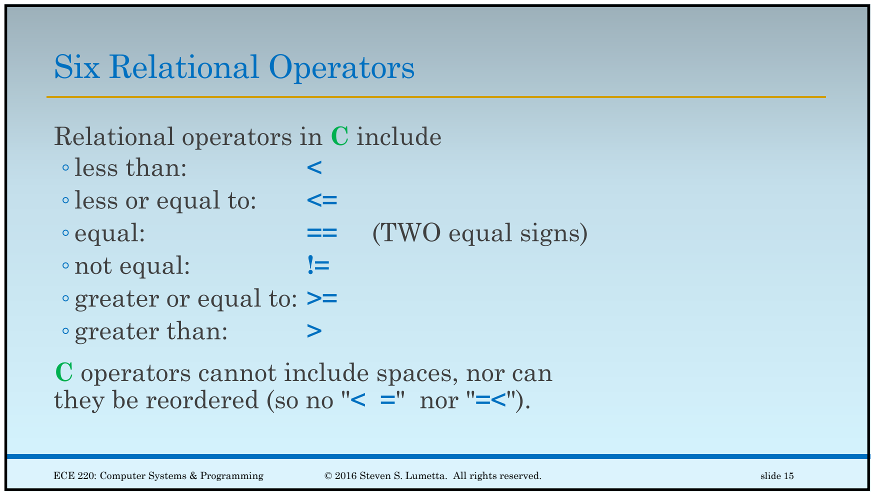
Right Shift by N Divides by 2^N

Declare: `int A = -120;/* 0xFFFFFF88 */`
`unsigned int B = 0xFFFFFF00;`

Then…

`A >> 2`	evaluates to	`-30`	`0xFFFFFFE2`
`A >> 10`	evaluates to	`-1`	`0xFFFFFFFF`
`B >> 2`	evaluates to		`0x3FFFFFC0`
`B >> 10`	evaluates to		`0x003FFFFF`

Notice that **right shifts round down**.

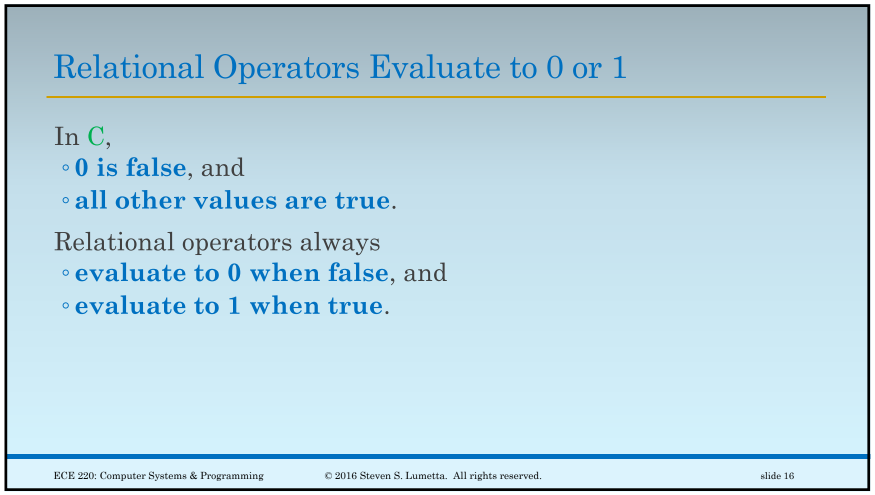


Six Relational Operators

Relational operators in **C** include

less than:	`<`	
less or equal to:	`<=`	
equal:	`==`	(TWO equal signs)
not equal:	`!=`	
greater or equal to:	`>=`	
greater than:	`>`	

C operators cannot include spaces, nor can they be reordered (so no "`< =`" nor "`=<`").



Relational Operators Evaluate to 0 or 1

In C,
- **0 is false**, and
- **all other values are true**.

Relational operators always
- **evaluate to 0 when false**, and
- **evaluate to 1 when true**.