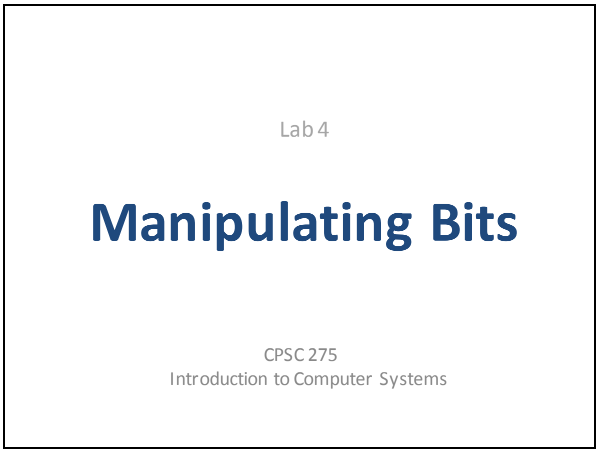
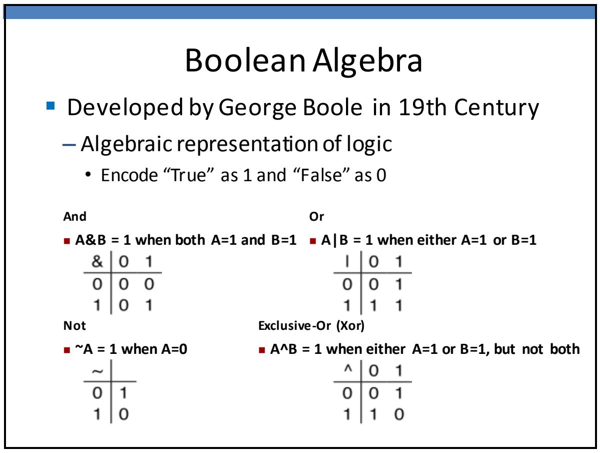


Lab 4

Manipulating Bits

CPSC 275
Introduction to Computer Systems



Boolean Algebra

- Developed by George Boole in 19th Century
  - Algebraic representation of logic
    - Encode “True” as 1 and “False” as 0

And

- A&B = 1 when both A=1 and B=1

&	0	1
0	0	0
1	0	1

Or

- A|B = 1 when either A=1 or B=1

|	0	1
0	0	1
1	1	1

Not

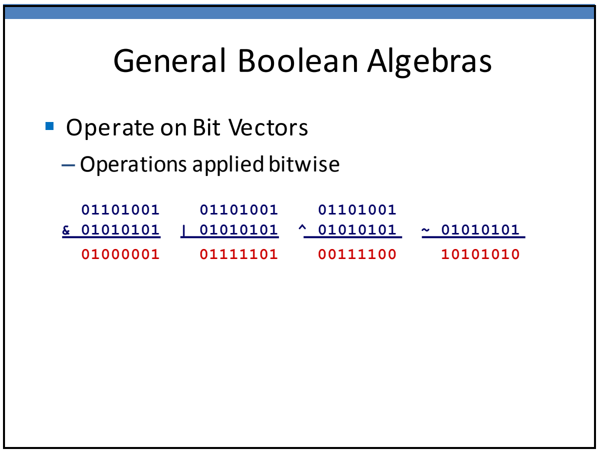
- ~A = 1 when A=0

~	
0	1
1	0

Exclusive-Or (Xor)

- A^B = 1 when either A=1 or B=1, but not both

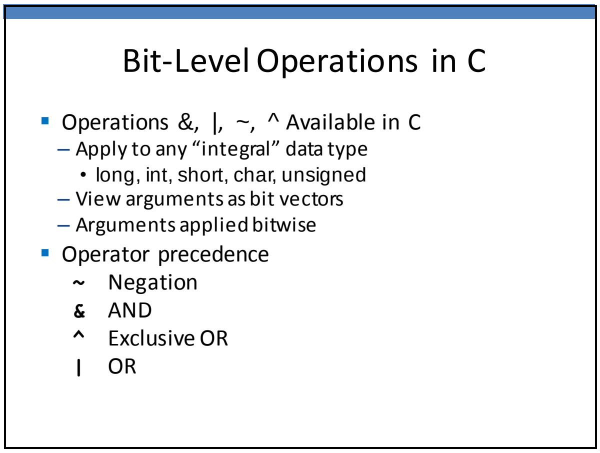
^	0	1
0	0	1
1	1	0



General Boolean Algebras

- Operate on Bit Vectors
  - Operations applied bitwise

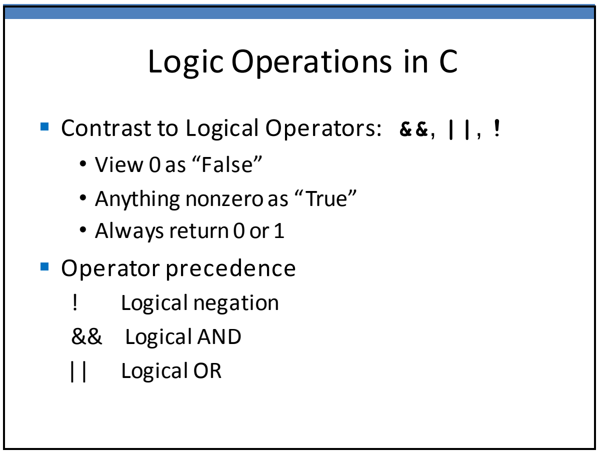
```
  01101001     01101001     01101001
& 01010101   | 01010101   ^ 01010101   ~ 01010101
  01000001     01111101     00111100     10101010
```



Bit-Level Operations in C

- Operations &, |, ~, ^ Available in C
  - Apply to any “integral” data type
    - long, int, short, char, unsigned
  - View arguments as bit vectors
  - Arguments applied bitwise
- Operator precedence

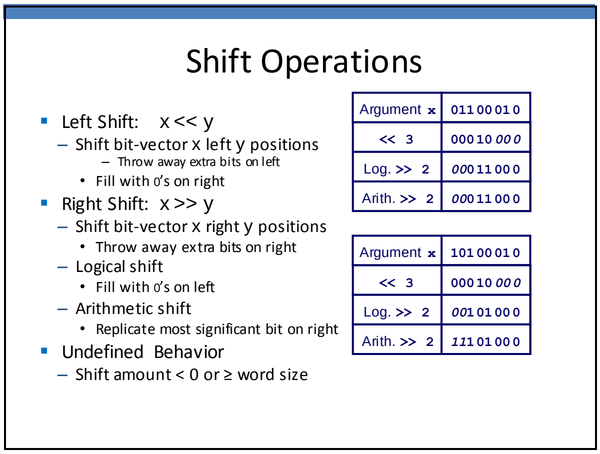
~	Negation
&	AND
^	Exclusive OR
|	OR



Logic Operations in C

- Contrast to Logical Operators: &&, ||, !
  - View 0 as “False”
  - Anything nonzero as “True”
  - Always return 0 or 1
- Operator precedence

!	Logical negation
&&	Logical AND
||	Logical OR

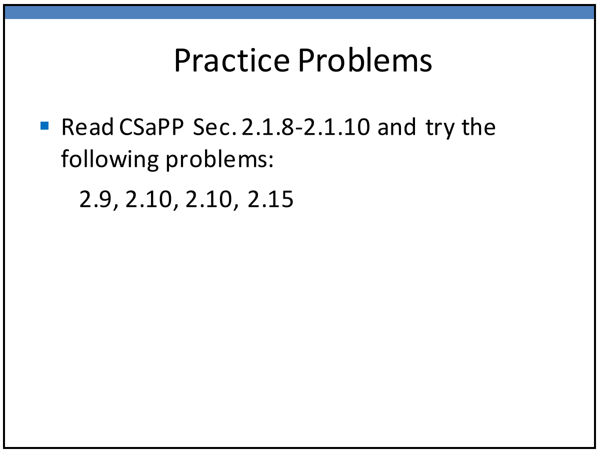


Shift Operations

- Left Shift: x << y
  - Shift bit-vector x left y positions
    - Throw away extra bits on left
    - Fill with 0’s on right
- Right Shift: x >> y
  - Shift bit-vector x right y positions
    - Throw away extra bits on right
  - Logical shift
    - Fill with 0’s on left
  - Arithmetic shift
    - Replicate most significant bit on right
- Undefined Behavior
  - Shift amount < 0 or ≥ word size

Argument x	01100010
<< 3	00010*000*
Log. >> 2	*00*011000
Arith. >> 2	*00*011000

Argument x	10100010
<< 3	00010*000*
Log. >> 2	*00*101000
Arith. >> 2	*11*101000



Practice Problems

- Read CSaPP Sec. 2.1.8-2.1.10 and try the following problems:

  2.9, 2.10, 2.10, 2.15