Bitwise Operators

Bitwise Operators

- What are they? Operators that work at the bit level
- &, |, ^, <<, >>, and ~ (Not to be confused with logical && and ||)
- Used to:
  - turn a bit on (1) or off (0)
  - find out if a bit is on or off
- Commonly used in low level programs (e.g. device drivers, embedded systems) and where space is scarce. Also for media compression.
- Typical Usage:
  - Assume an int has n bits.
  - Use it to store n true/false values (concisely!)
  - Decide on a meaning for each bit (e.g. SOUND_ON)
  - Assign 1 to to a bit to indicate true, 0 for false.

AND, OR, Exclusive OR Truth Table

A	B	A & B	A | B	A ^ B
		AND	OR	EXCLUSIVE OR
0	0	0	0	0
1	0	0	1	1
0	1	0	1	1
1	1	1	1	0

Try These Operations

```
  1 0 1 0                 1 0 1 0                 1 0 1 0
& 0 1 1 0               | 0 1 1 0               ^ 0 1 1 0
---------               ---------               ---------
  0 0 1 0
```

One's Complement

~

A	~A
0	1
1	0

Try These Operations

```
~ 1 0 1 0                ~ 0 0 1 0                ~ 1 1 1 1
---------                ---------                ---------
  0 1 0 1
```

Masks

- A “mask” is used to turn bits on and off
- The code below has 3 masks: READY, WAITING, RUNNING

```
#define READY 1
#define WAITING 2
#define RUNNING 4

int status;

// Turn status’ READY bit on (leaving other bits unchanged)
status = status | READY;

// Toggle the READY bit.
// If it’s on, turn it off. If it’s off, turn it on.
status = ???

// Clear all bits except the READY bit; leave READY bit unchanged.
status = ???
```

Masks

- A "mask" is used to turn bits on and off
- The code below has 3 masks: READY, WAITING, RUNNING

```
#define READY 1
#define WAITING 2
#define RUNNING 4

int status;

// Turn status' READY bit on (leaving other bits unchanged)
status = status | READY;

// Toggle the READY bit.
// If it's on, turn it off. If it's off, turn it on.
status = status ^ READY;

// Clear all bits except the READY bit; leave READY bit unchanged.
status = status & READY;
```

Bit Shift Left <<
Bit Shift Right >>

A	A << 1	A << 2	A <<3	A << 4
1101	1010	0100	1000	0000