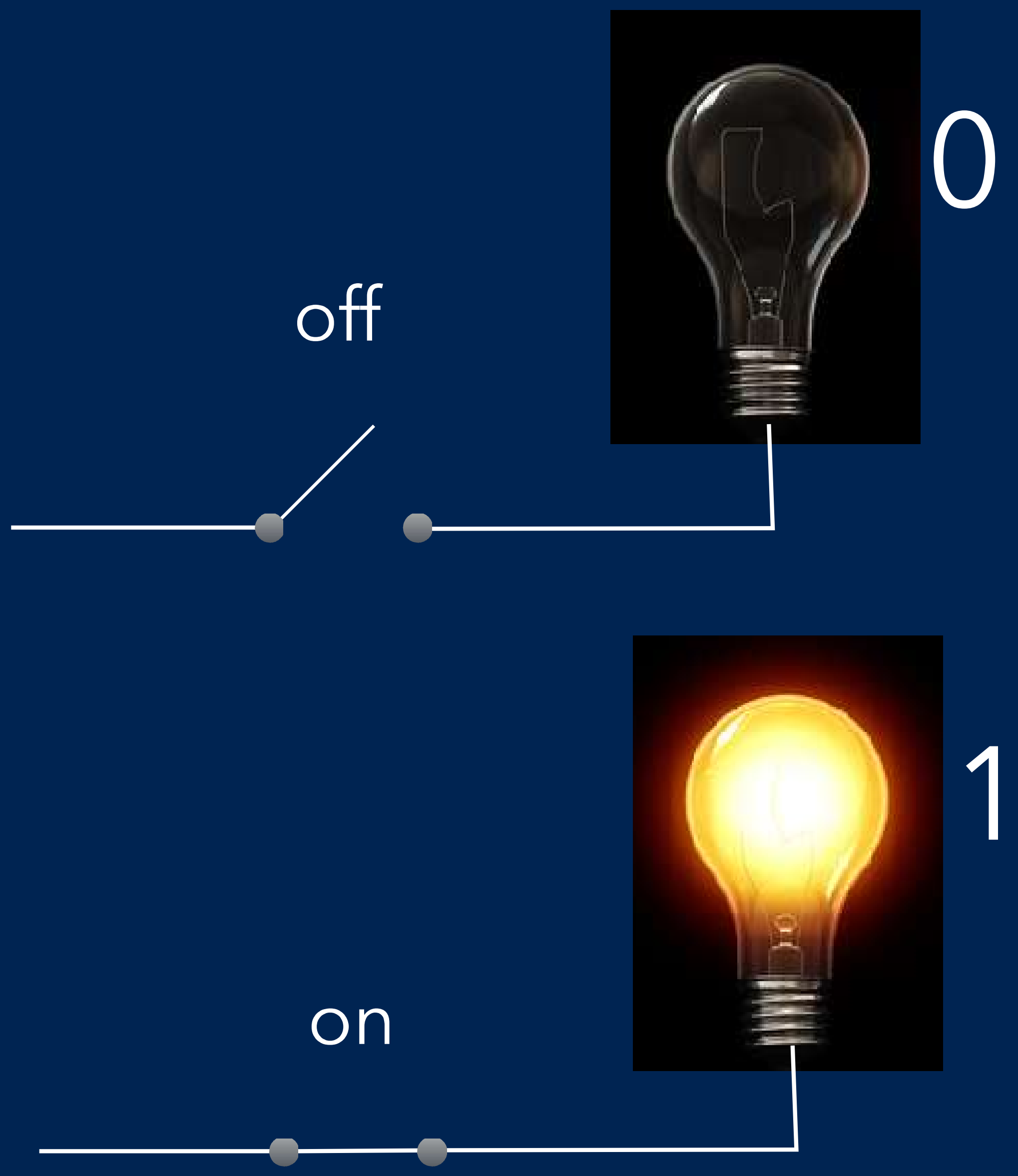
Binary arithmetic

CS 101 – Spring 2018

One bit



Two bits

bit 2	bit 1
0	0
0	1
1	0
1	1

Three bits

bit 3	bit 2	bit 1
0	0	0
0	0	1
0	1	0
0	1	1
1	0	0
1	0	1
1	1	0
1	1	1

bit 3	bit 2	bit 1
0	0	0
0	0	1
0	1	0
0	1	1
1	0	0
1	0	1
1	1	0
1	1	1

one bit: 2 patterns

two bits: 2 * 2 = 4 patterns

three bits: 2 * 2 * 2 = 8 patterns

.

.

.

n bits: **2^n patterns**

Common sizes

nibble: 4 bits [2^4 or 16]

byte: 8 bits [2^8 or 256]

word: depends on computer

8 bits [2^8 or 256]

16 bits [2^{16} or 65536]

32 bits [2^{32} or 4,294,967,296]

64 bits [2^{64} or 18,446,744,073,709,551,616]

Binary
or
Base-2

bit 4	bit 3	bit 2	bit 1	
0	0	0	0	**0**
0	0	0	1	**1**
0	0	1	0	**2**
0	0	1	1	**3**
0	1	0	0	**4**
0	1	0	1	**5**
0	1	1	0	**6**
0	1	1	1	**7**
1	0	0	0	**8**
1	0	0	1	**9**
1	0	1	0	**10**
1	0	1	1	**11**
1	1	0	0	**12**
1	1	0	1	**13**
1	1	1	0	**14**
1	1	1	1	**15**

Base-10

5120

5 thousands
1 hundreds
2 tens
0 ones

$$5 * 1000 + 1 * 100 + 2 * 10 + 0 * 1$$

$$5 * 10^3 + 1 * 10^2 + 2 * 10^1 + 0 * 10^0$$

Base-10

$$d_3d_2d_1d_0 =$$

$$d_3 * 10^3 + d_2 * 10^2 + d_1 * 10^1 + d_0 * 10^0$$

Base-b to Base-10

$$d_3d_2d_1d_0 =$$

$$d_3 * b^3 + d_2 * b^2 + d_1 * b^1 + d_0 * b^0$$

Base-2 to Base-10

$$1101_2 = ?_{10}$$

$$1 * 2^3 + 1 * 2^2 + 0 * 2^1 + 1 * 2^0$$

$$1 * 8 + 1 * 4 + 0 * 2 + 1 * 1$$

$$8 + 4 + 1$$

$$13_{10}$$

Base-10 to Base-2?

Algorithm for converting number N to base-2

step 1: if N is 0, go to step 5

step 2: divide N by 2 to get a quotient and remainder

step 3: set N to be the quotient

step 4: return to step 1

step 5: read off the remainders in reverse order - this is the number in base-2

$N = 19_{10}$

$N / 2 = 9$ R 1, $N = 9$

$N / 2 = 4$ R 1, $N = 4$

$N / 2 = 2$ R 0, $N = 2$

$N / 2 = 1$ R 0, $N = 1$

$N / 2 = 0$ R 1, $N = 0$

$$19_{10} = 10011_2$$

Doing math…

$$\begin{array}{r} {\scriptstyle 1} \\ 1 \\ +\ 1 \\ \hline 1\ 0 \end{array}$$

$$\begin{array}{rr} {\scriptstyle 1\ 1\ 1\ 1} & \\ 1101 & 13 \\ +\ 0011 & 3 \\ \cline{1-1} 10000 & 16 \end{array}$$

$$\begin{array}{rr} 10 & 2 \\ \times\ 11 & 3 \\ \cline{1-1} 10 & \\ +\ 10 & \\ \cline{1-1} 110 & 6 \end{array}$$

Integer Representation on a computer

- **Need to represent positive and negative numbers**

  –> Use a sign bit
- **Two’s complement notation**
  - Represent an integer with fixed # of bits
  - Leftmost bit is the *sign bit* (0: positive, 1: negative)
  - “Binary odometer”:

```
 1: 0001
 0: 0000
-1: 1111
```

Two’s complement with 4 bits

Decimal	Bit Pattern
7	0111
6	0110
5	0101
4	0100
3	0011
2	0010
1	0001
0	0000
-1	1111
-2	1110
-3	1101
-4	1100
-5	1011
-6	1010
-7	1001
-8	1000

Two’s complement

- To change sign:
    - From *right to left*, copy any 0s and first 1
    - Invert the remaining bits
- Examples (using 8 bits):

```
-20: 1110 1100  +96: 0110 0000  -1: 1111 1111  0: 0000 0000
+20: 0001 0100  -96: 1010 0000  +1: 0000 0001  0: 0000 0000
```