A note on binary numbers
Bits and Bit Operations

- A *bit* is a binary (base 2) digit: 0 or 1.
- Bits may be used to represent truth values.
- By convention:
  0 represents “false”;
  1 represents “true”.
- *Boolean algebra* is like ordinary algebra except that variables stand for bits,
  + means “or”, and multiplication means “and”.
  - See module 23 (chapter 10) for more details.
Bit Strings

• A *Bit string* of *length* \( n \) is an ordered sequence (series, tuple) of \( n \geq 0 \) bits.
  • More on sequences in §3.2.

• By convention, bit strings are (sometimes) written left to right:
  • *e.g.* the “first” bit of the bit string “1001101010” is 1.
  • But a more common convention is that the rightmost bit is bit #0, the 2\(^{nd}\)-rightmost is bit #1, etc.

• When a bit string represents a base-2 number, by convention, the first (leftmost) bit is the *most significant* bit. *Ex.* \( 1101_2 = 8 + 4 + 1 = 13 \).
Counting in Binary

• Did you know that you can count to 1023 just using two hands?
  • How? Count in binary!
    • Each finger (up/down) represents 1 bit.

• To increment: Flip the rightmost (low-order) bit.
  • If it changes 1→0, then also flip the next bit to the left,
    • If that bit changes 1→0, then flip the next one, *etc.*

• 0000000000, 0000000001, 0000000010, ...
  ..., 1111111101, 1111111110, 1111111111
Bitwise Operations

- Boolean operations can be extended to operate on bit strings as well as single bits.
- E.g.:
  01 1011 0110
  11 0001 1101
  Bit-wise OR
  Bit-wise AND
  Bit-wise XOR