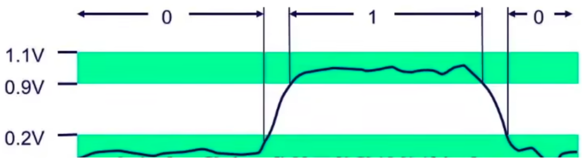
# 2_0

# Chapter 2: Representing and Manipulating Information

- Modern computers store and process information represented as two-valued signals: 0 and 1.




  - Each bit is 0 or 1, by using analog signal
  - Reliably transmitted on noisy and inaccurate wires
- In isolation, a single bit is not very useful. When we group bits together and apply some interpretation that gives meaning to the different possible bit patterns, however, we can represent the elements of any finite set, which is encoding.
- 3 most important representations of numbers:
  - Unsigned encoding (无符号编码) - the traditional way and representing numbers greater than or equal to 0.
  - Two's-complement encodings (补码编码) - the most common way to represent signed integers, that is, numbers that may be either positive or negative.
  - Floating-point encodings - a base-2 version of scientific notation for representing real numbers(实数).
    - 实数 - 有理数和无理数的集合。
- Some tips:
  - Overflow - Computer representations use a limited number of bits to encode a number, and hence some operations can overflow when the results are too large to be represented.

```
lldb: print 200*300*400*500
```

  - Integer computer arithmetic satisfies many of the familiar properties of true integer arithmetic: multiplication is associative and commutative.

```
lldb:
print 200*300*400*500
print 200*300*(400*500)
print 200*400*300*500
```

  - Floating-point arithmetic has altogether different mathematical properties.
    - The product(乘积) of a set of positive numbers will always be positive, although overflow will yield the special value + ∞.
    - Floating-point arithmetic is not associative due to the finite precision of the representation.

```
lldb:
print (3.14+1e20)-1e20
print 3.14+(1e20-1e20)
```

  - To summarize, integer representations can encode a comparatively **small range of values, but do so precisely**, while floating-point representations can encode **a wide**

**range of values, but only approximately.**