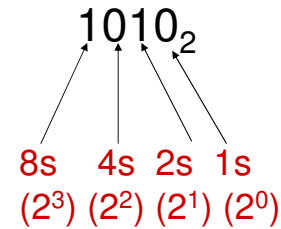
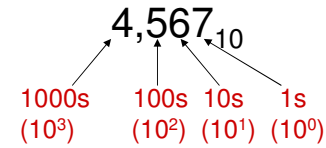
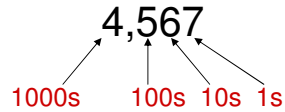


## Working in base 2

- Consider the number 4,567.
  - The value is:  $4 \cdot 1000 + 5 \cdot 100 + 6 \cdot 10 + 7 \cdot 1$ .
  - Back in grade school we'd have said that the 6 is "in the 10s place"



## Adding

$$\begin{array}{r} 1001 \\ + 0010 \\ \hline \hline \hline \end{array}$$

$$\begin{array}{r} 1001 \\ + 0011 \\ \hline \hline \hline \end{array}$$

$$\begin{array}{r} 1111 \\ + 0011 \\ \hline \hline \hline \end{array}$$

## Subtracting

$$\begin{array}{r} 1001 \\ - 0010 \\ \hline \hline \hline \end{array}$$

$$\begin{array}{r} 1001 \\ - 0011 \\ \hline \hline \hline \end{array}$$

$$\begin{array}{r} 1111 \\ - 0011 \\ \hline \hline \hline \end{array}$$

## Negative numbers

- Say I want to use 8 bits to represent positive and negative numbers.
  - One solution is to use the “Most Significant Bit” (MSB – the one on the far left) as a “sign bit”
    - If it is 1 the number is negative.
    - What would be the value of:
      - 1000 0110
      - 0000 1000
    - Now what is the smallest and largest number we could represent?
- This scheme is called “signed magnitude”

## More on negative numbers

- It turns out using signed-magnitude is pretty icky on a computer.
  - So instead they use “two’s complement”
    - If the MSB is 1 the number is negative
    - To get the magnitude of a negative flip all bits and add 1
      - 1111 1111 would be 0000 0000 +1 = 1. So 1111 1111 is negative 1.
      - 1000 0000 is an exception. That would be -128.
  - We use two’s complement because it turns out adding those numbers and adding unsigned numbers is really the same thing.
  - With 8 bits can represent  $2^7-1$  to  $-2^7$  (127 to -128)

## Consider adding

1001	<sup>c<sub>3</sub>c<sub>2</sub>c<sub>1</sub></sup> <b>abcd</b>
+ 0010	<b>efgh</b>
=====	=====
1011	<b>wxyz</b>

- Notice that
  - $z = (d \text{ and } !h) \text{ or } (!d \text{ and } h)$
  - $c_1 = d \text{ and } h$
- Who cares?
  - It turns out we can build an adder out of “and”, “or” and “not” operations.
  - We can also build subtractors, multipliers, etc.
- **We can use these simple “ands”, “ors” and “nots” to build a computer. But we pretty much are stuck with base 2.**