ARITHMETIC OPERATIONS ON BITS

So far

With $k$ bits we can represent $2^k$ numbers:

* Unsigned representation: from 0 to $2^k - 1$
* Signed representation: from $-(2^{k-1} - 1)$ to $+(2^{k-1} - 1)$
* 2's complement representation: from $-2^{k-1}$ to $+(2^{k-1} - 1)$
Addition using base 2

\[
\begin{array}{c}
0 + 0 + 1 + 1 \\
\hline
0 \quad 1 \quad 0 \quad 1
\end{array}
\]

Remember that computers have fixed width: an operation on a data type that has \( K \) bits should finish with \( K \) bits.
Addition & subtraction

* Unsigned addition:

```
  0 0 0 1₂ +
  0 1 0 1₂
```

© 2017-2018 Juan Jose Jaramillo. All rights reserved.
* Unsigned subtraction:
* Signed addition:

**Conclusion:** signed addition does not work!

* Signed subtraction: makes no sense either
* 2’s complement addition:
2's complement subtraction

**Key insight:** \[ A - B = A + (-B) \]
where \[ -B = \text{complement}(B) + 1 \]

Example: given \( A = 5_{10} = 00101_2 \) and \( B = -7_{10} = 11001_2 \), find \( A - B \)
Overflow & 2’s complement

It means the result of the addition cannot be represented in the number of bits allotted.

Example:

\[
\begin{array}{c}
0 \ 1 \ 0 \ 0 \ 1_2 + \\
0 \ 1 \ 0 \ 1 \ 0_2
\end{array}
\]

\[
\begin{array}{c}
9_{10} \\
10_{10}
\end{array}
\]
Example:

\[ \begin{array}{cccc}
\text{1 0 1 0 0}_2 & + & -12_{10} & + \\
\text{1 1 0 0 0}_2 & & -8_{10} & \\
\hline
\end{array} \]
How to detect overflow in 2's complement:

* If leading bit becomes 1 when adding two positive numbers

* If leading bit becomes 0 when adding two negative numbers
Overflow can also be detected by analyzing carry in bit into leftmost position and carry out bit:

Overflow occurs whenever \( C_n \neq C_{n-1} \)!
Sign extension

How to add two numbers of different number of bits?

Sign extension: extend the MSB as needed

Examples:

\[
\begin{array}{c}
00101_2 + 011_2 \\
\hline
10100_2
\end{array}
\]

\[
\begin{array}{c}
00101_2 + 110_2 \\
\hline
10011_2
\end{array}
\]