1.3 Signed binary numbers: Two's complement

Two's complement

Unsigned numbers involve only non-negative numbers, like 0 and 3. **Signed numbers** involve both positive and negative numbers, like 3 and -3.

In binary, a **signed-magnitude representation** uses the left bit for the sign: 0 means positive, 1 means negative. Ex: For 4-bit numbers, 0011 is 3, and 1011 is -3. Signed-magnitude representation is rarely used, because calculations involving negative numbers, such as 5 - 3, would require special circuits beyond an adder.

A more clever negative number representation exists that can use an adder for both positive and negative numbers. A **complement** of an N-digit number is another number that yields a sum of 100...00 (with N 0's), and can be used to represent the negative of that number.



1.3.1: Two's complement signed number representation.



o. . .



2x speed

Base 10:

Replace by:

Base 2:

Why?

7 + 3 = 10; 7 is the complement of 3.

Thus 5 + 7 is 10 too much, so the carry can be ignored.

Complement: invert bits, add 1.

0011 (3) has complement: (0011)' + 1 1100 + 1

So -3 is: 1101

Captions ^

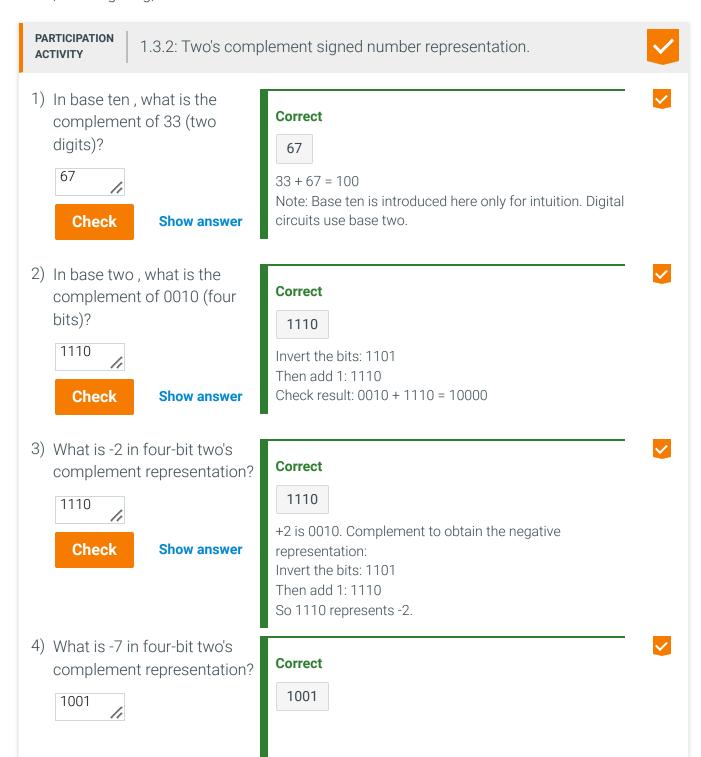
- 1. Intuition in base 10.
- 2. In base two, a complement is obtained by inverting each bit and adding 1.
- 3. Subtraction is performed by adding the complement.

4

The above is called the **two's complement representation**, which inverts every bit and adds 1. One's complement also exists, but is rarely used, and so is not discussed further. This material uses "complement" to mean two's complement.

The left bit indicates the sign. 0011 is +3. 1101 is a negative; complementing yields the positive version: 0010 + 1 = 0011, which is 3. So 1101 is -3.

Given a negative number like 1110, the value can be obtained by complementing, so 0001 + 1 = 0010, and negating, so -0010. Thus 1110 is -2.



7 is 0111. Complement to obtain the negative Check **Show answer** representation: Invert the bits: 1000 Then add 1: 1001 5) Assuming four-bit two's Correct complement representation, is 1011 positive or negative? Negative negative Left bit is 1, so negative. Check Show answer 6) Assuming two's complement Correct representation, what base ten number does 1111 -1 represent? Left bit is 1, so negative. Complement to find positive. 0000 + 1 = 0001Magnitude is 1. Negate to yield -1. Check **Show answer** 7) Assuming two's complement Correct representation, what base ten number does 1001 -7 represent? Left bit is 1, so negative. Complement to find positive. 0110 + 1 = 0111Magnitude is 7. Negate to yield -7. Check **Show answer** 8) In base two, for four bits, Correct what is the complement of 0000? 0000 0000 Invert the bits: 1111 Then add 1: 10000, which in four bits is 0000. 0000 is somewhat of an exception. Check **Show answer** 9) What is -3 in eight-bit two's Correct complement representation? 111111101 11111101 +3 is 00000011 in 8 bits. Check **Show answer** Complement to find negative:

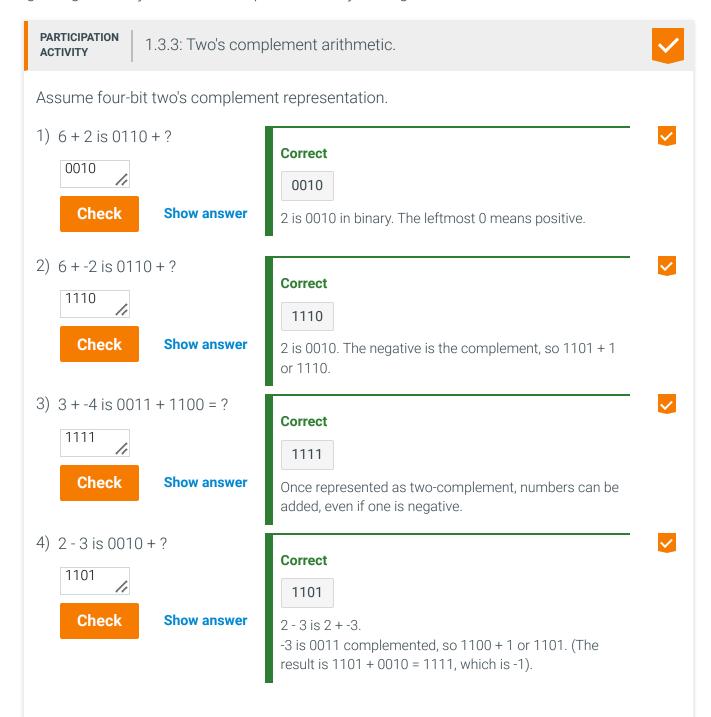
(00000011)' + 1 = 11111100 + 1 = 11111101

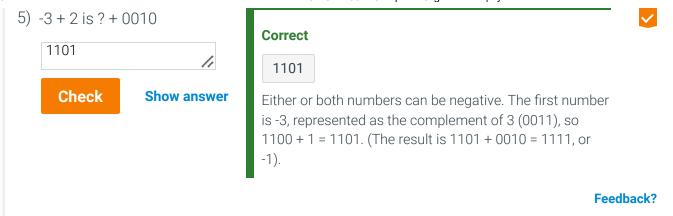
Feedback?

Note: This section uses 4-bit numbers for ease of example; wider numbers like 8 or 32 bits are more typical.

Subtracting by adding

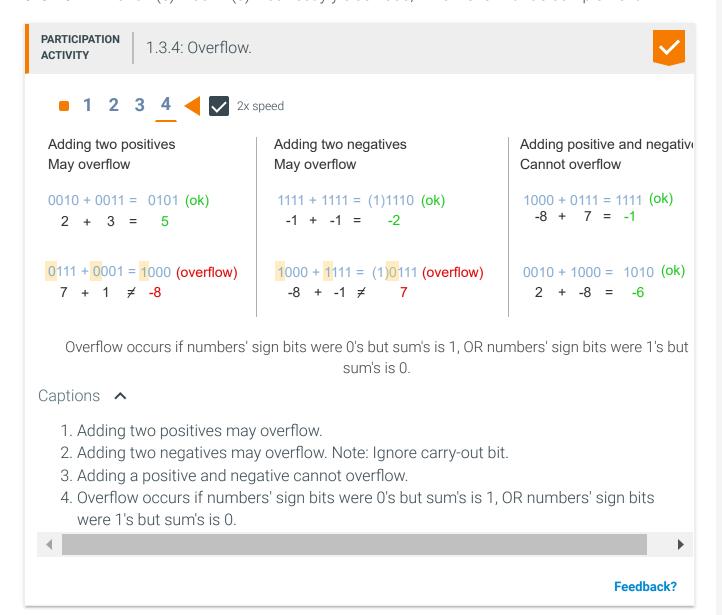
Two's complement representation has the benefit of allowing an adder to be used even when dealing with negative numbers. Ex: 5 + -3 is just 0101(5) + 1101(-3) = 10010, or 0010(2) after ignoring the carry. No extensive special circuitry for negative numbers is needed.





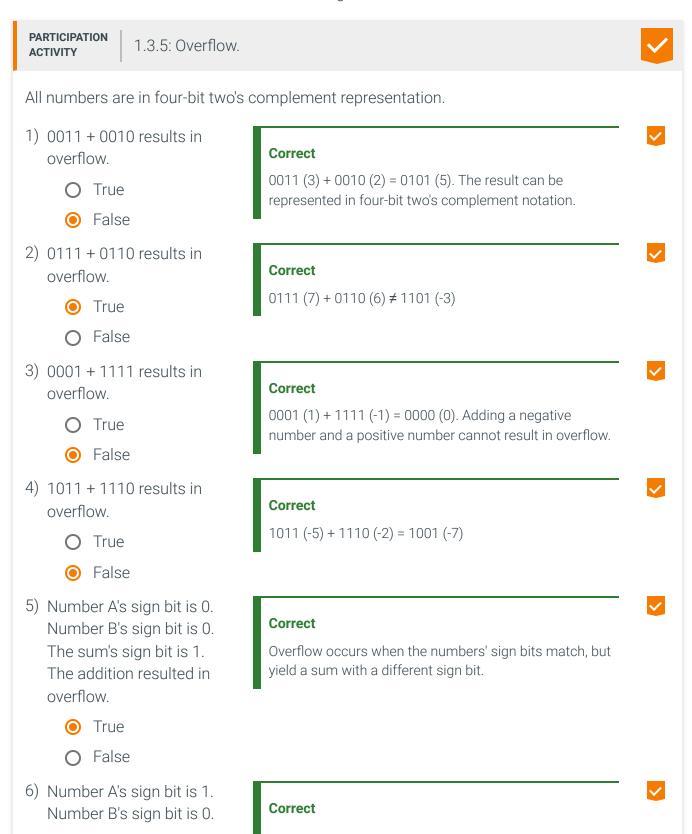
Overflow

The largest positive four-bit two's complement number is 0111, or 7. The smallest negative is 1000, or -8 (0111 + 1 = 1000, so magnitude is 8). Adding two positives, or adding two negatives, may yield a value that can't be represented in the given number of bits, a situation known as **overflow**. Ex: 0101 (5) + 0011 (3) incorrectly yields 1000, which is -8 in two's complement.



As seen above, overflow occurs if the numbers being added have the same sign bit but the sum's sign bit differs. In other words, overflow occurs if two positives sum to a negative (clearly wrong), or two negatives sum to a positive (clearly wrong).

Adding a positive number and negative number (or vice-versa) cannot result in overflow. The sum always has a smaller magnitude than one or both of the numbers, so clearly can fit in the same number of bits. Ex: 7 + -2 = 5, and 5's magnitude is smaller than 7.



The sum's sign bit is 0.

The addition resulted in overflow.

Addir resulted in a neg

Adding a negative number and a positive number cannot result in overflow. The numbers' sign bits differ, meaning a negative and positive are being added.

O True

False

Feedback?

