## 1.5 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 nur -3.

In binary, a **signed-magnitude representation** uses the left bit for the sign: 0 means positive, 1 means negative. Ex: For 4-bir is 3, and 1011 is -3. Signed-magnitude representation is rarely used, because calculations involving negative numbers, such 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 **com** 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

PARTICIPATION ACTIVITY	1 5 1. Two's complement signed number representation					
Start	2x speed					
Base 10:		Base 2:				
Replace by:						
5	5	0101 (5)	0101 (5)			
- 3	+ 7	- 0011 (3)	+ 1101 (-3)			
2	1 2 ignore carry	0010 (2)	1 0010 (2) ignore carry			
Why?	Complement: invert bits, add 1.					
7 + 3 = 10; 7 is the complement of 3.		0011 (3) has compl	lement: (0011)' + 1			
Thus 5 + 7 is 10 too much, so the			1100 + 1			
carry can be ignored.		\$	So -3 is: 1101			

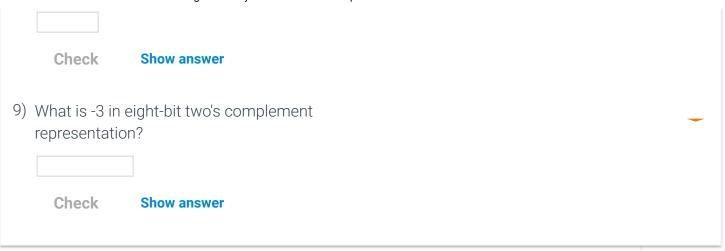
The above is called the **two's complement representation**, which inverts every bit and adds 1. One's complement also exists used, and so is not discussed further. This material uses "complement" to mean two's complement.

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

Given a negative number like 1110, the value can be obtained by complementing, so 0001 + 1 = 0010, and negating, so -00° -2.

PARTICIPATION ACTIVITY	1.5.2: Two's complement signed number representation.	
1) In base ten 33 (two dig	gits)?	_
Check	Show answer	
0010 (four		•
Check	Show answer	
3) What is -2 i representat	in four-bit two's complement tion?	•
Chaola	Show answer	

	Offeck	
4)	What is -7 in for representation	our-bit two's complement ?
	Check	Show answer
5)		r-bit two's complement , is 1011 positive or
	Check	Show answer
5)	Assuming two representation does 1111 rep	, what base ten number
	Check	Show answer
7)	Assuming two representation does 1001 rep	, what base ten number
	Check	Show answer
3)	In base two, fo complement o	r four bits, what is the f 0000?



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 number just 0.01(5) + 1.01(-3) = 1.0010, or 0.010(2) after ignoring the carry. No extensive special circuitry for negative numbers is r

PARTICIPATION ACTIVITY

1.5.3: Two's complement arithmetic.

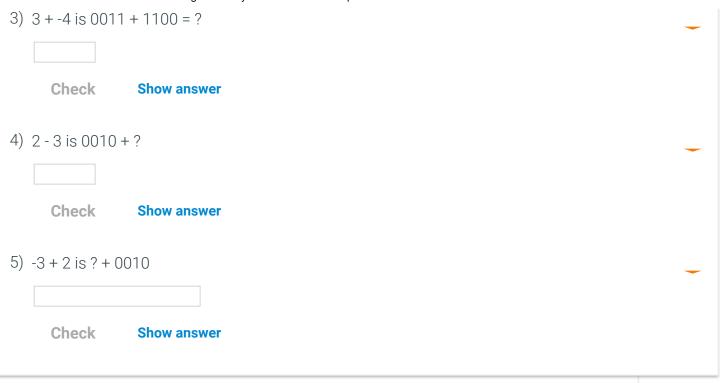
Assume four-bit two's complement representation.

1) 6 + 2 is 0110 + ?

Check Show answer

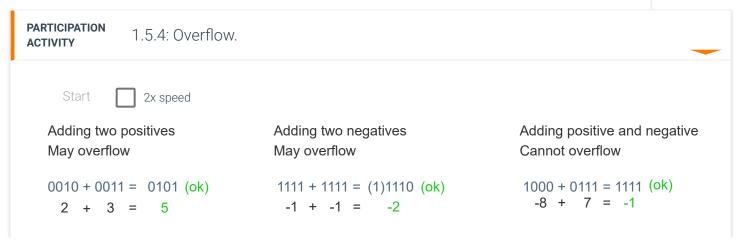
2) 6 + -2 is 0110 + ?

Check Show answer



## **Overflow**

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

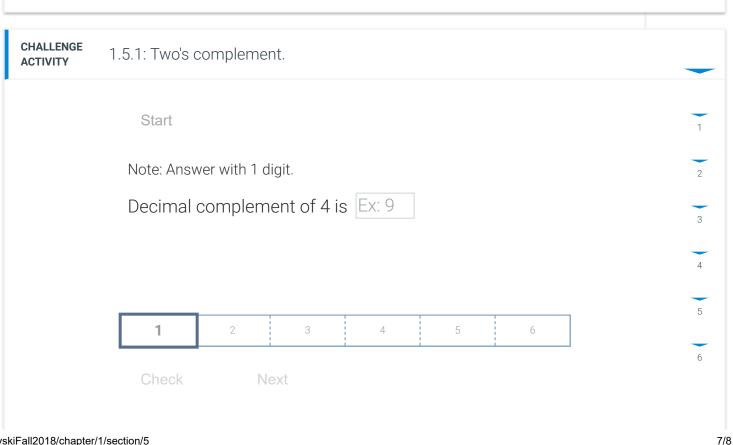


As seen above, overflow occurs if the numbers being added have the same sign bit but the sum's sign bit differs. In other w occurs if two positives sum to a negative (clearly wrong), or two negatives sum to a positive (clearly wrong).

Adding a positive number and negative number (or vice-versa) cannot result in overflow. The sum always has a smaller ma 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.

PARTICIPATION ACTIVITY 1.5.5: Overflow.	_				
All numbers are in four-bit two's complement representation.					
1) 0011 + 0010 results in overflow.					
O True					
O False					
2) 0111 + 0110 results in overflow.					
O True					
O False					
3) 0001 + 1111 results in overflow.	_				
O True					
O False					
4) 1011 + 1110 results in overflow.					
O True					







Provide feedback on this section