Signed Binary addition

Overflow occurs when the result of a binary operation is too large to fit in allowed number of bits.

- **When adding two signed numbers of different signs, ignore the carry-out bit.**
- When adding two signed numbers of the same sign, the sign bit of the sum should match the numbers.
- If the two numbers are negative and the MSB is 0, then there is an overflow with a carry-out bit of 1. The carry-out bit must become the sign bit. (if adding a bit is allowed)
- If the two numbers are positive and the MSB is 1, then there is an overflow with a carry-out bit of 0. The carry-out bit must become the sign bit. (if adding a bit is allowed)

Start at LSB

Example: add binary 01010 + 11111110 (10 + (-2) = 8)

carry:	1	1	1	1	1	1	1	0	0
num1:		0	0	0	0	1	0	1	0
num2:		1	1	1	1	1	1	1	0
+									
		0	0	0	0	1	0	0	0

Signed Binary addition

Overflow occurs when the result of a binary operation is too large to fit in allowed number of bits.

- When adding two signed numbers of different signs, ignore the carry-out bit.
- When adding two signed numbers of the same sign, the sign bit of the sum should match the numbers.
- If the two numbers are negative and the MSB is 0, then there is an overflow with a carry-out bit of 1. The carry-out bit must become the sign bit. (if adding a bit is allowed)
- If the two numbers are positive and the MSB is 1, then there is an overflow with a carry-out bit of 0. The carry-out bit must become the sign bit. (if adding a bit is allowed)

Start at LSB

Example: add binary 1111001 + 111110 (-7 + (-2)) = -9

carry:	1	1	1	1	1	0	0	0	0
num1:		1	1	1	1	1	0	0	1
num2:		1	1	1	1	1	1	1	0
+									
		1	1	1	1	0	1	1	1

Signed Binary addition

Overflow occurs when the result of a binary operation is too large to fit in allowed number of bits.

- When adding two signed numbers of different signs, ignore the carry-out bit.
- When adding two signed numbers of the same sign, the sign bit of the sum should match the numbers.
- **If the two numbers are negative and the MSB is 0, then there is an overflow with a carry-out bit of 1. The carry-out bit must become the sign bit. (if adding a bit is allowed)**
- If the two numbers are positive and the MSB is 1, then there is an overflow with a carry-out bit of 0. The carry-out bit must become the sign bit. (if adding a bit is allowed)

Example: 10000001 + 10000001 (-127 + (-127) = -254)

Start at LSB

carry:	1	0	0	0	0	0	0	1	0
num1:		1	0	0	0	0	0	0	1
num2:		1	0	0	0	0	0	0	1
+									
	1	0	0	0	0	0	0	1	0

Signed Binary addition

Overflow occurs when the result of a binary operation is too large to fit in allowed number of bits.

- When adding two signed numbers of different signs, ignore the carry-out bit.
- When adding two signed numbers of the same sign, the sign bit of the sum should match the numbers.
- If the two numbers are negative and the MSB is 0, then there is an overflow with a carry-out bit of 1. The carry-out bit must become the sign bit. (if adding a bit is allowed)
- If the two numbers are positive and the MSB is 1, then there is an overflow with a carry-out bit of 0. The carry-out bit must become the sign bit. (if adding a bit is allowed)

Start at LSB

Example: 00010001 + 00011 (17 + 3 = 20)

carry:	0	0	0	0	0	0	1	1	0
num1:		0	0	0	1	0	0	0	1
num2:		0	0	0	0	0	0	1	1
+									
		0	0	0	1	0	1	0	0

Signed Binary addition

Overflow occurs when the result of a binary operation is too large to fit in allowed number of bits.

- When adding two signed numbers of different signs, ignore the carry-out bit.
- When adding two signed numbers of the same sign, the sign bit of the sum should match the numbers.
- If the two numbers are negative and the MSB is 0, then there is an overflow with a carry-out bit of 1. The carry-out bit must become the sign bit. (if adding a bit is allowed)
- **If the two numbers are positive and the MSB is 1, then there is an overflow with a carry-out bit of 0. The carry-out bit must become the sign bit. (if adding a bit is allowed)**

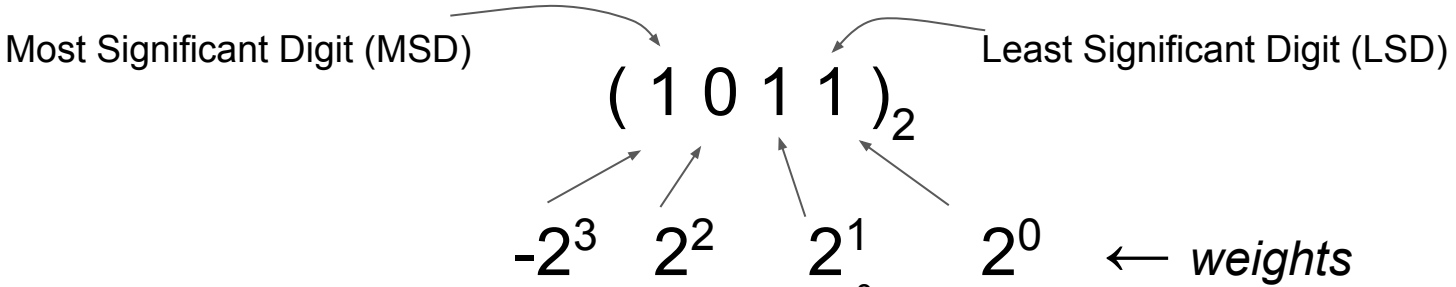
Example: 01000001 + 01000001 (65 + 65 = 130)

Start at LSB

carry:	0	1	0	0	0	0	0	1	0
num1:		0	1	0	0	0	0	0	1
num2:		0	1	0	0	0	0	0	1
+									
	0	1	0	0	0	0	0	1	0

Weighted-Positional number systems (SIGNED):

Example: Consider 1011, signed binary number (base-2 number)



- starting with lowest weight of 2^0, moving from right-to-left
- the **weight** of each bit, increases by a factor of 2
- The sign of the weight of the most significant bit is always negative

---

The value of a number is weighted sum of its digits:

$$(1\ 0\ 1\ 1)_2 = (1 \times (-2^3)) + (0 \times 2^2) + (1 \times 2^1) + (1 \times 2^0)$$
$$(1\ 0\ 1\ 1)_2 = -8 + 0 + 2 + 1 = -5$$

$$111010_2 = -(1 \times 2^5) + (1 \times 2^4) + (1 \times 2^3) + (0 \times 2^2) + (1 \times 2^1) + (0 \times 2^0)$$
$$= (1 \times 2^4) + (1 \times 2^3) + (0 \times 2^2) + (1 \times 2^1) + (0 \times 2^0) - (1 \times 2^5)$$
$$= 16 + 8 + 0 + 2 + 0 - 32$$
$$= -6$$
$$001010_2 = -(0 \times 2^5) + (0 \times 2^4) + (1 \times 2^3) + (0 \times 2^2) + (1 \times 2^1) + (0 \times 2^0)$$
$$= (0 \times 2^4) + (1 \times 2^3) + (0 \times 2^2) + (1 \times 2^1) + (0 \times 2^0) - (0 \times 2^5)$$
$$= 0 + 8 + 0 + 2 + 0 - 0$$
$$= 10$$

convert **from** decimal **to** signed binary $(-25)_{10} \rightarrow (?)_2$

Repeated Division

Divide the given decimal number by base=2.

Write the remainder after each division until a quotient of zero is obtained.

Division	Quotient	Remainder	
25/2	12	1	**LSB**
12/2	6	0	
6/2	3	0	
3/2	1	1	
1/2	0	1	
-	-	0	**MSB**

We read the answer from the remainder column, bottom-up, adding 0 as the sign bit. If the decimal number is negative. Find the two's complement to get the final answer.

$(011001)_2 = 100111_2$

answer = 100111_2

Quiz#2 next week Tue. 2025-10-07:

Prepare and Practice

Signed: weighted sum, repeated division, binary addition/subtraction, two's complement

- Use Signed Weighted Sum algorithm to convert from:
  $base_2 \rightarrow base_{10}$
- Use Signed Repeated Division algorithm to convert from:
  $base_{10}$ to $base_2$
- Properly convert to Two's Complement
- Be able to show bit-by-bit signed addition/subtraction with carry values:
  00001111 + 00000010
  11010101 + 01101011
  10001111 + 10001111