Signed numbers

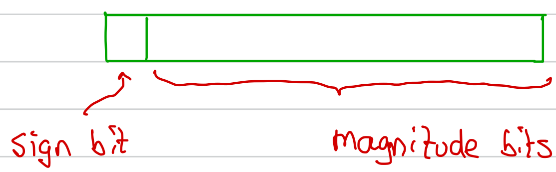
Note the numbers considered previously are all unsigned numbers (always positive or zero). However, we really need a system to represent both positive and negative.

We use the sign/magnitude format in daily life.

e.g., $+5$, -37, 44

sign magnitude implied "+"

In the binary number system, sign/magnitude format is



sign: 1 – negative
0 – positive

E.g., for 5 bits

$01010_2 = +10_{10}$

$11010_2 = -10_{10}$

For an n-bit number, there is one sign bit, n-1 magnitude bits

> For a signed number, we must always know the size n

The largest magnitude is

$$max = 2^{n-1} - 1$$

e.g.

n	max
5	15
16	32767
32	2,147,483,647

giving a total range of

$$-(2^{n-1}-1) \leq N \leq (2^{n-1}-1)$$

Sign/magnitude is the simplest representation, but:

- Difficult to add numbers of opposite sign
  e.g., suppose we simply add +3 and -5 (in 4 bits)

$$3 + -5 = 0011_2 + 1101_2 = 10000_2 \quad \text{(wrong!)}$$

- Two representations of zero (+0 and −0)

The two's-complement number system

Almost universally used for signed binary numbers. Uses the same sign bit as above, plus:

Positive numbers: identical to sign/magnitude

e.g., in 5 bits,
$$01001_2 = +9_{10}$$
$$00000_2 = 0_{10}$$

Negative numbers:

For an n-bit number, $-a$, 2's-complement representation is given by the binary representation of

$$N = 2^n - |a|$$

E.g., for the number -10_{10} and $n=5$ bits, we have

$$N = 2^5 - 10 = 22_{10} = 10110_2$$

Comparing:

number	sign/magnitude	2's-complement
10_{10}	01010_2	01010_2
-10_{10}	11010_2	10110_2

Three-step method for finding the 2's-complement representation of a negative number:

Consider again -10_5 in 5 bits:

1. Find binary value of magnitude (5 bits) $\quad +10_{10} = 01010_2$
2. Invert (i.e., complement each bit, $0 \to 1$, $1 \to 0$) $\quad 01010 \longrightarrow 10101$
3. Add 1 to the result

$$\begin{array}{r} 10101 \\ +\quad 1 \\ \hline 10110 \end{array} \longleftarrow -10_{10} \text{ in 2's-complement representation}$$

Steps 2 and 3 are called <u>2's-complement negation</u>

<u>Examples:</u> Represent the numbers -17_{10} and 23_{10} in 2's-complement 6-bit format.

First, -17_{10}. Find its magnitude in binary

$17 \div 2 =$	8,	remainder = 1,	$d_0 = 1$
$8 \div 2 =$	4,	rem = 0,	$d_1 = 0$
$4 \div 2 =$	2,	rem = 0,	$d_2 = 0$
$2 \div 2 =$	1,	rem = 0,	$d_3 = 0$
$1 \div 2 =$	0,	rem = 1,	$d_4 = 1$

$17_{10} = 010001_2$

pad to make 6 bits

Complement each bit: $\quad 101110$

And add 1:

$$\begin{array}{r} 101110 \\ +\quad 1 \\ \hline 101111_2 \end{array}$$

2's-complement negation

Hence, $-17_{10} = 101111_2$

Now $+23_{10}$. This is a positive number, so just convert it to binary.

By successive division,

$$23_{10} = 010111_2$$

Beauty of 2's-complement representation:

- only one representation of 0 (e.g., 000000_2)
- no decisions to make when adding numbers of opposite sign; regular old binary adder all you need.

Example: Add our two previous 6-bit 2's-complement numbers -17_{10} and $+23_{10}$