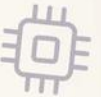


how a fixed number of bits is made to carry a sign, a fraction and a character



A fixed width forces a choice

- a machine word holds a fixed number of bits, so every scheme below is a decision about what those bits are spent on
- an unsigned 8 bit pattern counts 0 to 255, because all eight bits carry magnitude
- the moment negatives are wanted, one bit has to say which sign, and the range paid for it
- nothing in the pattern itself says which scheme is in force: 10000101 is 133 unsigned, minus 5 in sign magnitude and minus 123 in 2's complement



the same eight bits: read as unsigned they total 133, and the highlighted leading bit is what a signed scheme spends on the sign

Three ways to write a negative number

Sign magnitude

- leading bit is the sign, 0 for plus and 1 for minus, the rest is plain magnitude
- easy to read, but two patterns mean zero and adding needs the signs compared first

1's complement

- negate by flipping every bit
- still two zeros, and addition needs the end around carry

2's complement

- negate by flipping every bit and adding 1
- one zero only, and ordinary addition just works, which is why hardware uses it

all three agree on positive numbers and differ only once the leading bit is 1

What n bits can hold

unsigned

0 to $2^n - 1$

sign magnitude

$-(2^{n-1} - 1)$ to $+(2^{n-1} - 1)$, with two zeros

1's complement

$-(2^{n-1} - 1)$ to $+(2^{n-1} - 1)$, with two zeros

2's complement

-2^{n-1} to $+(2^{n-1} - 1)$, one zero

on 8 bits: 0 to 255 unsigned, minus 127 to plus 127 for the first two, and minus 128 to plus 127 for 2's complement