Método sinal e amplitude/magnitude:

Define-se esse bit para *0* para representar um número positivo, e define-se como *1* para representar um número negativo. Os bits restantes do número representam a magnitude (ou valor absoluto). Assim, em um byte com 8 bits, são utilizados 7 bits para representar o valor e um bit para representar o sinal. Neste caso, o valor pode variar de 0000000 (0) a 1111111 (127). Representando números de -127_{10} a $+127_{10}$, uma vez que você adicione o bit de sinal (o oitavo bit).

Exemplo:
$+10_{10}$ = **0**1010_2
-10_{10} = **1**1010_2

Complemento p/ 1:

Para adicionar dois números representados neste sistema, se faz uma adição binária convencional, mas é então necessário adicionar qualquer "vai-um" resultante de volta para a soma resultante. Para ver por que isso é necessário, considere o seguinte exemplo:
100_{10} = **0**1100100_2
Inverte-se...
1$0011011_2$ = -100_{10}

Contra:
O problema desta representação é que existe 2 padrões de bits para o 0, havendo assim desperdício de representação:
0_{10} = 00000000_2 = 1111111_2

Complemento p/ 2:

Em complemento para dois, há apenas um zero (00000000). Se nega um número (negativo ou positivo) invertendo-se todos os bits e, em seguida, adicionando 1 ao resultado. A adição de um par números em complemento para dois é o mesmo a adição de um par de números sem-sinal .

Exemplo:
101_{10} = 01100101_2
Inverte-se...
10011010_2
Soma-se uma unidade...
$10011010_2 + 1 = 10011011 = -101_{10}$

Prós: Numero mais exato devido a maior capacidade de arquitetura

ANDRE SILVEIRA MACHADO
INTRODUCAO A INFORMATICA