Números Inteiros em Complemento de Dois

- Como representar números negativos?
- Usar um bit para sinal?
  Possível, mas há solução melhor.
- Usar complemento de dois.

Inteiros com 4 bits

- Um conjunto de 4 bits pode representar 16 valores distintos.

0000	0001	0010	0011
0100	0101	0110	0111
1000	1001	1010	1011
1100	1101	1110	1111

Inteiro sem sinal

- Representação mais simples. Números de 0 a 15. Não tem números negativos.

0000 0	0001 1	0010 2	0011 3
0100 4	0101 5	0110 6	0111 7
1000 8	1001 9	1010 10	1011 11
1100 12	1101 13	1110 14	1111 15

Bit mais signficativo é sinal

- Bit mais significativo 0: positivo, 1: negativo. Números de -7 a 7. 0 é duplicado.

0000 0	0001 1	0010 2	0011 3
0100 4	0101 5	0110 6	0111 7
1000 0	1001 -1	1010 -2	1011 -3
1100 -4	1101 -5	1110 -6	1111 -7

Não muito prático!

Complemento de 2

- Contando de um número para o seguinte.
- Desprezamos "vai um" que não cabe nos bits disponíveis.
- Último número (`1111`) é seguido pelo primeiro (`0000`).
- Se contamos para trás, `1111` é anterior a `0000`.
- Usamos `1111` para representar **-1**, etc...

Complemento de 2

- Números de -8 a 7.

0000 0	0001 1	0010 2	0011 3
0100 4	0101 5	0110 6	0111 7
1000 -8	1001 -7	1010 -6	1011 -5
1100 -4	1101 -3	1110 -2	1111 -1

Inversão de sinal

- Inverter sinal do número com complemento de 2:
  - Trocar o valor de todos os bits.
  - Somar 1 no resultado.

```
0101  (5)
1010
1011  (-5)
```

Soma

- Soma com complemento de 2.
  - Soma os dois números.
  - Despreza vai um que não cabe.

```
0011    (3)          0110    (6)
0100+   (4)          1100+   (-4)
0111    (7)          0010    (2)
```

```
1101    (-3)
1100+   (-4)
1001    (-7)
```

Subtração

- Subtração com complemento de 2:
  - Inverte o sinal do segundo valor.
  - Soma o resultado (complemento de 2)

```
0111   (7)
0011-  (3)
```



```
0111   (7)
1101+  (-3)
0100   (4)
```