Binários: Aritmética e Representação de Negativos

PCS3115 - Sistemas Digitais 1

Departamento de Engenharia de Computação e Sistemas Digitais
Escola Politécnica
Universidade de São Paulo

São Paulo, 2018

No tópico passado vimos...

- Sistemas de numeração posicionais, inclusive o binário.

No tópico passado vimos...

- Sistemas de numeração posicionais, inclusive o binário.
- Conversão entre bases, focando na binária.

No tópico passado vimos...

- Sistemas de numeração posicionais, inclusive o binário.
- Conversão entre bases, focando na binária.
- Mas como **operar** $(+, -, \times, /)$ números binários?

No tópico passado vimos...

- Sistemas de numeração posicionais, inclusive o binário.
- Conversão entre bases, focando na binária.
- Mas como **operar** $(+, -, \times, /)$ números binários?
- E os **negativos**?

Tópicos da aula de hoje

- Soma, subtração, multiplicação e divisão em binário.

Tópicos da aula de hoje

- Soma, subtração, multiplicação e divisão em binário.
- Binários negativos via sinal-magnitude.

Tópicos da aula de hoje

- Soma, subtração, multiplicação e divisão em binário.
- Binários negativos via sinal-magnitude.
- Negativos via complemento de 2.

Tópicos da aula de hoje

- Soma, subtração, multiplicação e divisão em binário.
- Binários negativos via sinal-magnitude.
- Negativos via complemento de 2.
- Operações com complemento de 2.

Soma com 2 números decimais

- Como somar 456 com 1345?

Soma com 2 números decimais

- Como somar 456 com 1345?
- Alinhamos os operandos à direta.

Soma com 2 números decimais

- Como somar 456 com 1345?
- Alinhamos os operandos à direta.
- Da direita pra esquerda, para cada par de dígitos, calculamos a soma (considerando se "veio" um do dígito anterior) e vemos se "vai-um" para o próximo dígito.

Soma no sistema binário

- Considerando 2 bits isolados:

Soma no sistema binário

- Considerando 2 bits isolados: 0+0=0

Soma no sistema binário

- Considerando 2 bits isolados: 0+0=0, 0+1=1+0=1

Soma no sistema binário

- Considerando 2 bits isolados: 0+0=0, 0+1=1+0=1, 1+1=10.

Soma no sistema binário

- Considerando 2 bits isolados: 0+0=0, 0+1=1+0=1, 1+1=10.
- $x_n \dots x_1 x_0 + y_n \dots y_1 y_0$: operamos **bit** a **bit** $(x_j + y_j)$.

Soma no sistema binário

- Considerando 2 bits isolados: 0+0=0, 0+1=1+0=1, 1+1=10.
- $x_n \dots x_1 x_0 + y_n \dots y_1 y_0$: operamos **bit** a **bit** $(x_j + y_j)$.
- 0+0=0

Soma no sistema binário

- Considerando 2 bits isolados: 0+0=0, 0+1=1+0=1, 1+1=10.
- $x_n \dots x_1 x_0 + y_n \dots y_1 y_0$: operamos **bit** a **bit** $(x_j + y_j)$.
- 0+0=0, 0+1=1+0=1

Soma no sistema binário

- Considerando 2 bits isolados: 0+0=0, 0+1=1+0=1, 1+1=10.
- $x_n \dots x_1 x_0 + y_n \dots y_1 y_0$: operamos **bit** a **bit** $(x_j + y_j)$.
- 0+0=0, 0+1=1+0=1, 1+1=0, e **vai-um** (carry).

Soma no sistema binário

- Considerando 2 bits isolados: 0+0=0, 0+1=1+0=1, 1+1=10.
- $x_n \dots x_1 x_0 + y_n \dots y_1 y_0$: operamos **bit** a **bit** $(x_j + y_j)$.
- 0+0=0, 0+1=1+0=1, 1+1=0, e **vai-um** (carry).
- A soma no bit anterior $(x_{j-1} + y_{j-1})$ pode ter gerado um **vai-um** (c_j):

Soma no sistema binário

- Considerando 2 bits isolados: 0+0=0, 0+1=1+0=1, 1+1=10.
- $x_n \dots x_1 x_0 + y_n \dots y_1 y_0$: operamos **bit** a **bit** $(x_j + y_j)$.
- 0+0=0, 0+1=1+0=1, 1+1=0, e **vai-um** (carry).
- A soma no bit anterior $(x_{j-1} + y_{j-1})$ pode ter gerado um **vai-um** (c_j):

x_j	y_j	c_j	s_j	c_{j+1}
0	0	1	1	0
0	1	1	0	1
1	0	1	0	1
1	1	1	1	1

Soma no sistema binário

$$
\begin{array}{cccccccccc}
 & & & 1 & 1 & 1 & 1 & 1 & 1 & \\
 & 1 & 0 & 0 & 1 & 1 & 1 & 1 & 0 & 1 \\
+ & 0 & 1 & 0 & 1 & 1 & 0 & 1 & 1 & 1 \\
\hline
 & 1 & 1 & 1 & 1 & 1 & 0 & 1 & 0 & 0
\end{array}
$$

Soma no sistema binário

$$
\begin{array}{cccccccccc}
 & & & 1 & 1 & 1 & 1 & 1 & 1 & \\
 & 1 & 0 & 0 & 1 & 1 & 1 & 1 & 0 & 1 \\
+ & 0 & 1 & 0 & 1 & 1 & 0 & 1 & 1 & 1 \\
\hline
 & 1 & 1 & 1 & 1 & 1 & 0 & 1 & 0 & 0
\end{array}
$$

- **Exercício**: Somar 100111 com 10101.

Soma no sistema binário

$$
\begin{array}{cccccccccc}
 & & & 1 & 1 & 1 & 1 & 1 & 1 & \\
 & 1 & 0 & 0 & 1 & 1 & 1 & 1 & 0 & 1 \\
+ & 0 & 1 & 0 & 1 & 1 & 0 & 1 & 1 & 1 \\
\hline
 & 1 & 1 & 1 & 1 & 1 & 0 & 1 & 0 & 0
\end{array}
$$

- **Exercício**: Somar 100111 com 10101. Correspondentes em decimais?

Subtração no sistema binário

- Considerando 2 bits isolados.

Subtração no sistema binário

- Considerando 2 bits isolados.
- 0-0=0

Subtração no sistema binário

- Considerando 2 bits isolados.
- 0-0=0, 1-0=1

Subtração no sistema binário

- Considerando 2 bits isolados.
- 0-0=0, 1-0=1, 1-1=0

Subtração no sistema binário

- Considerando 2 bits isolados.
- 0-0=0, 1-0=1, 1-1=0 , 0-1=?

Subtração no sistema binário

- Considerando 2 bits isolados.
- 0-0=0, 1-0=1, 1-1=0 , 0-1=?
- $x_n \dots x_1 x_0 - y_n \dots y_1 y_0$: operamos **bit** a **bit** $(x_j - y_j)$.

Subtração no sistema binário

- Considerando 2 bits isolados.
- 0-0=0, 1-0=1, 1-1=0 , 0-1=?
- $x_n \dots x_1 x_0 - y_n \dots y_1 y_0$: operamos **bit** a **bit** $(x_j - y_j)$.
- 0-0=0

Subtração no sistema binário

- Considerando 2 bits isolados.
- 0-0=0, 1-0=1, 1-1=0 , 0-1=?
- $x_n \dots x_1 x_0 - y_n \dots y_1 y_0$: operamos **bit** a **bit** $(x_j - y_j)$.
- 0-0=0, 1-0=1

Subtração no sistema binário

- Considerando 2 bits isolados.
- 0-0=0, 1-0=1, 1-1=0 , 0-1=?
- $x_n \dots x_1 x_0 - y_n \dots y_1 y_0$: operamos **bit** a **bit** $(x_j - y_j)$.
- 0-0=0, 1-0=1, 1-1=0

Subtração no sistema binário

- Considerando 2 bits isolados.
- 0-0=0, 1-0=1, 1-1=0 , 0-1=?
- $x_n \ldots x_1 x_0 - y_n \ldots y_1 y_0$: operamos **bit** a **bit** $(x_j - y_j)$.
- 0-0=0, 1-0=1, 1-1=0, 0-1:, **pega-um-emprestado** (borrow) 10-1=1.

Subtração no sistema binário

- Considerando 2 bits isolados.
- 0-0=0, 1-0=1, 1-1=0 , 0-1=?
- $x_n \ldots x_1 x_0 - y_n \ldots y_1 y_0$: operamos **bit** a **bit** $(x_j - y_j)$.
- 0-0=0, 1-0=1, 1-1=0, 0-1:, **pega-um-emprestado** (borrow) 10-1=1.
- O bit anterior $(x_{j-1} - y_{j-1})$ pode ter "pegado um emprestado" do próximo bit $(b_j = 1)$:

Subtração no sistema binário

- Considerando 2 bits isolados.
- 0-0=0, 1-0=1, 1-1=0 , 0-1=?
- $x_n \ldots x_1 x_0 - y_n \ldots y_1 y_0$: operamos **bit** a **bit** $(x_j - y_j)$.
- 0-0=0, 1-0=1, 1-1=0, 0-1:, **pega-um-emprestado** (borrow) 10-1=1.
- O bit anterior $(x_{j-1} - y_{j-1})$ pode ter "pegado um emprestado" do próximo bit $(b_j = 1)$:

x_j	y_j	b_j	d_j	b_{j+1}
0	0	1	1	1
0	1	1	0	1
1	0	1	0	0
1	1	1	1	1

Subtração no sistema binário

Exemplo: 25-14

−	1	1	1			borrow
+	1	1	0	0	1	X
−	0	1	1	1	0	Y
	0	1	0	1	1	D

Subtração no sistema binário

Exemplo: 25-14

−	1	1	1			borrow
+	1	1	0	0	1	X
−	0	1	1	1	0	Y
	0	1	0	1	1	D

Subtração no sistema binário

Exemplo: 25-14

−	1	1	1			borrow
+	1	1	0	0	1	X
−	0	1	1	1	0	Y
	0	1	0	1	1	D

- **Exercício**: Subtrair 1111 de 11001. Correspondentes em decimais?

Multiplicação no Sistema Binário

- Como calcular 567×234 em decimal?

Multiplicação no Sistema Binário

- Como calcular 567×234 em decimal?
- $234 = 200 + 30 + 4 \;\rightarrow\; 4 \times 567 + 30 \times 567 + 200 \times 567$

Multiplicação no Sistema Binário

- Como calcular 567×234 em decimal?
- $234 = 200 + 30 + 4 \;\rightarrow\; 4 \times 567 + 30 \times 567 + 200 \times 567$
- $4 \times 567 + 3 \times 5670 + 2 \times 56700 = 132678$

Multiplicação no Sistema Binário

- Como calcular 567×234 em decimal?
- $234 = 200 + 30 + 4 \;\rightarrow\; 4 \times 567 + 30 \times 567 + 200 \times 567$
- $4 \times 567 + 3 \times 5670 + 2 \times 56700 = 132678$
- Mesma coisa em binário: Método **desloca-e-soma**

Multiplicação no Sistema Binário

- Como calcular 567×234 em decimal?
- $234 = 200 + 30 + 4 \;\rightarrow\; 4 \times 567 + 30 \times 567 + 200 \times 567$
- $4 \times 567 + 3 \times 5670 + 2 \times 56700 = 132678$
- Mesma coisa em binário: Método **desloca-e-soma**

$$
\begin{array}{r}
1\,0\,1\,0 \\
\times\;\; 1\,0\,1\,1 \\
\hline
1\,0\,1\,0 \\
1\,0\,1\,0 \\
0\,0\,0\,0 \\
1\,0\,1\,0 \\
\hline
1\,1\,0\,1\,1\,1\,0 \\
\hline
\end{array}
$$

Multiplicação - Algoritmo Formal

- Multiplicando: A, Multiplicador: $B = b_n \ldots b_0$

Multiplicação - Algoritmo Formal

- Multiplicando: A, Multiplicador: $B = b_n \ldots b_0$
- **Produto parcial** armazenado em P, inicialmente nulo.

Multiplicação - Algoritmo Formal

- Multiplicando: A, Multiplicador: $B = b_n \ldots b_0$
- **Produto parcial** armazenado em P, inicialmente nulo.
- Para i de 0 a n (para cada bit do multiplicador B)

Multiplicação - Algoritmo Formal

- Multiplicando: A, Multiplicador: $B = b_n \ldots b_0$
- **Produto parcial** armazenado em P, inicialmente nulo.
- Para i de 0 a n (para cada bit do multiplicador B)
- $P \leftarrow P + b_i * 2^i A$

Multiplicação - Algoritmo Formal

- Multiplicando: A, Multiplicador: $B = b_n \ldots b_0$
- **Produto parcial** armazenado em P, inicialmente nulo.
- Para i de 0 a n (para cada bit do multiplicador B)
- $P \leftarrow P + b_i * 2^i A$
- Ao fim, $P = \sum_{i=1}^{n} b_i * 2^i A$

Multiplicação - Algoritmo Formal

- Multiplicando: A, Multiplicador: $B = b_n \ldots b_0$
- **Produto parcial** armazenado em P, inicialmente nulo.
- Para i de 0 a n (para cada bit do multiplicador B)
- $P \leftarrow P + b_i * 2^i A$
- Ao fim, $P = \sum_{i=1}^{n} b_i * 2^i A = A \sum_{i=1}^{n} b_i * 2^i$

Multiplicação - Algoritmo Formal

- Multiplicando: A, Multiplicador: $B = b_n \dots b_0$
- **Produto parcial** armazenado em P, inicialmente nulo.
- Para i de 0 a n (para cada bit do multiplicador B)
- $P \leftarrow P + b_i * 2^i A$
- Ao fim, $P = \sum_{i=1}^{n} b_i * 2^i A = A \sum_{i=1}^{n} b_i * 2^i = AB$

Multiplicação - Algoritmo Formal

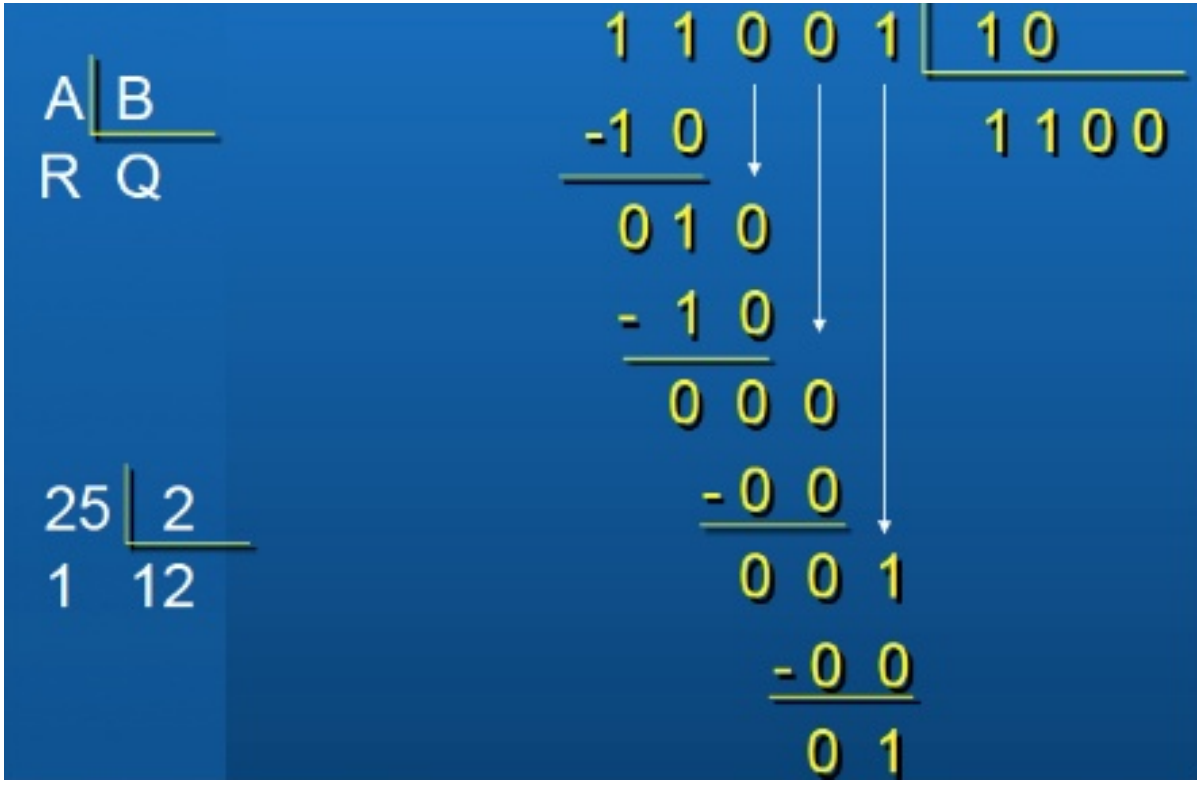
- Multiplicando: A, Multiplicador: $B = b_n \dots b_0$
- **Produto parcial** armazenado em P, inicialmente nulo.
- Para i de 0 a n (para cada bit do multiplicador B)
- $P \leftarrow P + b_i * 2^i A$
- Ao fim, $P = \sum_{i=1}^{n} b_i * 2^i A = A \sum_{i=1}^{n} b_i * 2^i = AB$
- **Exercício**: Multiplicar 10101 por 101.

Divisão no sistema binário

- Método análogo ao caso decimal, **desloca-e-subtrai**.

Divisão no sistema binário

- Método análogo ao caso decimal, **desloca-e-subtrai**.



Divisão - Algoritmo Formal

- Método análogo ao caso decimal, **desloca-e-subtrai**.

Divisão - Algoritmo Formal

- Método análogo ao caso decimal, **desloca-e-subtrai**.
- Dividendo $A = a_m, \ldots a_0$, Divisor $B = b_n \ldots b_n$.

Divisão - Algoritmo Formal

- Método análogo ao caso decimal, **desloca-e-subtrai**.
- Dividendo $A = a_m, \ldots a_0$, Divisor $B = b_n \ldots b_n$.
- Dividendo reduzido D, inicialmente igual a A, no fim será o resto.

Divisão - Algoritmo Formal

- Método análogo ao caso decimal, **desloca-e-subtrai**.
- Dividendo $A = a_m, \ldots a_0$, Divisor $B = b_n \ldots b_n$.
- Dividendo reduzido D, inicialmente igual a A, no fim será o resto.
- Quociente $Q = q_{m-n} \ldots q_0$, queremos calcular.

Divisão - Algoritmo Formal

- Método análogo ao caso decimal, **desloca-e-subtrai**.
- Dividendo $A = a_m, \ldots a_0$, Divisor $B = b_n \ldots b_n$.
- Dividendo reduzido D, inicialmente igual a A, no fim será o resto.
- Quociente $Q = q_{m-n} \ldots q_0$, queremos calcular.
- Para $i = m - n$ até $i = 0$

Divisão - Algoritmo Formal

- Método análogo ao caso decimal, **desloca-e-subtrai**.
- Dividendo $A = a_m, \ldots a_0$, Divisor $B = b_n \ldots b_n$.
- Dividendo reduzido D, inicialmente igual a A, no fim será o resto.
- Quociente $Q = q_{m-n} \ldots q_0$, queremos calcular.
- Para $i = m - n$ até $i = 0$
  - **Desloca**: Acrescenta i zeros à direita de B: $2^i B$

Divisão - Algoritmo Formal

- Método análogo ao caso decimal, **desloca-e-subtrai**.
- Dividendo $A = a_m, \ldots a_0$, Divisor $B = b_n \ldots b_n$.
- Dividendo reduzido D, inicialmente igual a A, no fim será o resto.
- Quociente $Q = q_{m-n} \ldots q_0$, queremos calcular.
- Para $i = m - n$ até $i = 0$
  - **Desloca**: Acrescenta i zeros à direita de B: $2^i B$
  - **Subtrai**: Se $2^i B \leq D$, $D \leftarrow D - 2^i B, q_i \leftarrow 1$, C.C. $q_i \leftarrow 0$

Divisão - Algoritmo Formal

- Método análogo ao caso decimal, **desloca-e-subtrai**.
- Dividendo $A = a_m, \ldots a_0$, Divisor $B = b_n \ldots b_n$.
- Dividendo reduzido D, inicialmente igual a A, no fim será o resto.
- Quociente $Q = q_{m-n} \ldots q_0$, queremos calcular.
- Para $i = m - n$ até $i = 0$
  - **Desloca**: Acrescenta i zeros à direita de B: $2^i B$
  - **Subtrai**: Se $2^i B \leq D$, $D \leftarrow D - 2^i B, q_i \leftarrow 1$, C.C. $q_i \leftarrow 0$
- **Exercício**: Dividir 10101 por 101.

Representação de Negativos - O Bug de 2038

- Sistemas baseados em Unix com 32 bits armazenam data e hora com 32 bits.

Representação de Negativos - O Bug de 2038

- Sistemas baseados em Unix com 32 bits armazenam data e hora com 32 bits.
- A data é armazenada como o número de segundos desde meia-noite de 1/jan/1970.

Representação de Negativos - O Bug de 2038

- Sistemas baseados em Unix com 32 bits armazenam data e hora com 32 bits.
- A data é armazenada como o número de segundos desde meia-noite de 1/jan/1970.
- **Problema**: Um segundo depois de 3h14m07s de 19/jan/2038, tais sistemas voltarão para 20h45m52s de 13/dez/1901.

Representação de Negativos - O Bug de 2038

- Sistemas baseados em Unix com 32 bits armazenam data e hora com 32 bits.
- A data é armazenada como o número de segundos desde meia-noite de 1/jan/1970.
- **Problema**: Um segundo depois de 3h14m07s de 19/jan/2038, tais sistemas voltarão para 20h45m52s de 13/dez/1901.
- Com a **aula de hoje**, vamos entender o por quê.

Representando números negativos

- Em decimal, sinais $+$ e $-$ à frente.

Representando números negativos

- Em decimal, sinais $+$ e $-$ à frente.
- **Ideia**: usar um **bit de sinal**, a frente um número fixo (n) de bits

Representando números negativos

- Em decimal, sinais $+$ e $-$ à frente.
- **Ideia**: usar um **bit de sinal**, a frente um número fixo (n) de bits

$$011_2 = \ 3_{10}$$

Representando números negativos

- Em decimal, sinais $+$ e $-$ à frente.
- **Ideia**: usar um **bit de sinal**, a frente um número fixo (n) de bits

$$011_2 = \ 3_{10}$$

Representando números negativos

- Em decimal, sinais $+$ e $-$ à frente.
- **Ideia**: usar um **bit de sinal**, a frente um número fixo (n) de bits
  - $0011_2 = +3_{10}$

Representando números negativos

- Em decimal, sinais $+$ e $-$ à frente.
- **Ideia**: usar um **bit de sinal**, a frente um número fixo (n) de bits
  - $0011_2 = +3_{10}$
    $011_2 = \ \ 3_{10}$

Representando números negativos

- Em decimal, sinais $+$ e $-$ à frente.
- **Ideia**: usar um **bit de sinal**, a frente um número fixo (n) de bits
  - $0011_2 = +3_{10}$
    $011_2 = \ \ 3_{10}$

Representando números negativos

- Em decimal, sinais $+$ e $-$ à frente.
- **Ideia**: usar um **bit de sinal**, a frente um número fixo (n) de bits
  - $0011_2 = +3_{10}$
  - $1011_2 = -3_{10}$

Representando números negativos

- Em decimal, sinais $+$ e $-$ à frente.
- **Ideia**: usar um **bit de sinal**, a frente um número fixo (n) de bits
  - $0011_2 = +3_{10}$
  - $1011_2 = -3_{10}$
- Sistema **sinal-magnitude**.

Representando números negativos

- Em decimal, sinais $+$ e $-$ à frente.
- **Ideia**: usar um **bit de sinal**, a frente um número fixo (n) de bits
  - $0011_2 = +3_{10}$
  - $1011_2 = -3_{10}$
- Sistema **sinal-magnitude**.
- **Problema 1**: Antes de somar (subtrair) dois números, precisamos checar os sinais...

Representando números negativos

- Em decimal, sinais $+$ e $-$ à frente.
- **Ideia**: usar um **bit de sinal**, a frente um número fixo (n) de bits
  - $0011_2 = +3_{10}$
  - $1011_2 = -3_{10}$
- Sistema **sinal-magnitude**.
- **Problema 1**: Antes de somar (subtrair) dois números, precisamos checar os sinais...
- **Problema 2**: Duas representações para o zero (com 3 bits: 000 e 100).

Complemento de 1 e a 2

- Em binário, o **complemento** de um bit d é $\bar{d} = 1 - d$: o complemento de 1 é 0, o complemento de 0 é 1.

Complemento de 1 e a 2

- Em binário, o **complemento** de um bit d é $\bar{d} = 1 - d$: o complemento de 1 é 0, o complemento de 0 é 1.
- O **complemento de 1** de $d_n \ldots d_2 d_1$ é $\overline{d_n \ldots d_1} = \bar{d}_n \ldots \bar{d}_1$.

Complemento de 1 e a 2

- Em binário, o **complemento** de um bit d é $\bar{d} = 1 - d$: o complemento de 1 é 0, o complemento de 0 é 1.
- O **complemento de 1** de $d_n \ldots d_2 d_1$ é $\overline{d_n \ldots d_1} = \bar{d}_n \ldots \bar{d}_1$.
- **Exemplo**: complemento de 1 de 101_2 é $\overline{101} = 010$

Complemento de 1 e a 2

- Em binário, o **complemento** de um bit d é $\bar{d} = 1 - d$: o complemento de 1 é 0, o complemento de 0 é 1.
- O **complemento de 1** de $d_n \ldots d_2 d_1$ é $\overline{d_n \ldots d_1} = \bar{d}_n \ldots \bar{d}_1$.
- **Exemplo**: complemento de 1 de 101_2 é $\overline{101} = 010$
- O **complemento de 2** de $d_n \ldots d_2 d_1$ é $\bar{d}_n \ldots \bar{d}_2 \bar{d}_1 + 1$.

Complemento de 1 e a 2

- Em binário, o **complemento** de um bit d é $\bar{d} = 1 - d$: o complemento de 1 é 0, o complemento de 0 é 1.
- O **complemento de 1** de $d_n \ldots d_2 d_1$ é $\overline{d_n \ldots d_1} = \bar{d}_n \ldots \bar{d}_1$.
- **Exemplo**: complemento de 1 de 101_2 é $\overline{101} = 010$
- O **complemento de 2** de $d_n \ldots d_2 d_1$ é $\bar{d}_n \ldots \bar{d}_2 \bar{d}_1 + 1$.O complemento de 2 de um número é o (complemento de 1) mais 1.

Complemento de 1 e a 2

- Em binário, o **complemento** de um bit d é $\bar{d} = 1 - d$: o complemento de 1 é 0, o complemento de 0 é 1.
- O **complemento de 1** de $d_n \ldots d_2 d_1$ é $\overline{d_n \ldots d_1} = \bar{d}_n \ldots \bar{d}_1$.
- **Exemplo**: complemento de 1 de 101_2 é $\overline{101} = 010$
- O **complemento de 2** de $d_n \ldots d_2 d_1$ é $\bar{d}_n \ldots \bar{d}_2 \bar{d}_1 + 1$.O complemento de 2 de um número é o (complemento de 1) mais 1.
- Fixamos n bits, complemento de 2 de 000 é $111 + 1 = 1000$.

Complemento de 1 e a 2

- Em binário, o **complemento** de um bit d é $\bar{d} = 1 - d$: o complemento de 1 é 0, o complemento de 0 é 1.
- O **complemento de 1** de $d_n \ldots d_2 d_1$ é $\overline{d_n \ldots d_1} = \bar{d}_n \ldots \bar{d}_1$.
- **Exemplo**: complemento de 1 de 101_2 é $\overline{101} = 010$
- O **complemento de 2** de $d_n \ldots d_2 d_1$ é $\bar{d}_n \ldots \bar{d}_2 \bar{d}_1 + 1$.O complemento de 2 de um número é o (complemento de 1) mais 1.
- Fixamos n bits, complemento de 2 de 000 é $111 + 1 = \not{1}000$.

Complemento de 1 e a 2

- Em binário, o **complemento** de um bit d é $\bar{d} = 1 - d$: o complemento de 1 é 0, o complemento de 0 é 1.
- O **complemento de 1** de $d_n \ldots d_2 d_1$ é $\overline{d_n \ldots d_1} = \bar{d}_n \ldots \bar{d}_1$.
- **Exemplo**: complemento de 1 de 101_2 é $\overline{101} = 010$
- O **complemento de 2** de $d_n \ldots d_2 d_1$ é $\bar{d}_n \ldots \bar{d}_2 \bar{d}_1 + 1$.O complemento de 2 de um número é o (complemento de 1) mais 1.
- Fixamos n bits, complemento de 2 de 000 é $111 + 1 = \not{1}000$.
- **Exemplo**: complemento de 2 de 101_2 é $\overline{101} + 1 = 011$

Complemento de 1 e a 2

- Em binário, o **complemento** de um bit d é $\bar{d} = 1 - d$: o complemento de 1 é 0, o complemento de 0 é 1.
- O **complemento de 1** de $d_n \dots d_2 d_1$ é $\overline{d_n \dots d_1} = \bar{d}_n \dots \bar{d}_1$.
- **Exemplo**: complemento de 1 de 101_2 é $\overline{101} = 010$
- O **complemento de 2** de $d_n \dots d_2 d_1$ é $\bar{d}_n \dots \bar{d}_2 \bar{d}_1 + 1$.O complemento de 2 de um número é o (complemento de 1) mais 1.
- Fixamos n bits, complemento de 2 de 000 é $111 + 1 = \not{1}000$.
- **Exemplo**: complemento de 2 de 101_2 é $\overline{101} + 1 = 011$
- Com n bits, o complemento de 2 de $x < 2^n$ é $y = 2^n - x$.

Complemento de 1 e a 2

- Em binário, o **complemento** de um bit d é $\bar{d} = 1 - d$: o complemento de 1 é 0, o complemento de 0 é 1.
- O **complemento de 1** de $d_n \dots d_2 d_1$ é $\overline{d_n \dots d_1} = \bar{d}_n \dots \bar{d}_1$.
- **Exemplo**: complemento de 1 de 101_2 é $\overline{101} = 010$
- O **complemento de 2** de $d_n \dots d_2 d_1$ é $\bar{d}_n \dots \bar{d}_2 \bar{d}_1 + 1$.O complemento de 2 de um número é o (complemento de 1) mais 1.
- Fixamos n bits, complemento de 2 de 000 é $111 + 1 = \not{1}000$.
- **Exemplo**: complemento de 2 de 101_2 é $\overline{101} + 1 = 011$
- Com n bits, o complemento de 2 de $x < 2^n$ é $y = 2^n - x$. $(1111 - x + 1 = 10000 - x)$

Complemento de 1 e a 2

- Em binário, o **complemento** de um bit d é $\bar{d} = 1 - d$: o complemento de 1 é 0, o complemento de 0 é 1.
- O **complemento de 1** de $d_n \dots d_2 d_1$ é $\overline{d_n \dots d_1} = \bar{d}_n \dots \bar{d}_1$.
- **Exemplo**: complemento de 1 de 101_2 é $\overline{101} = 010$
- O **complemento de 2** de $d_n \dots d_2 d_1$ é $\bar{d}_n \dots \bar{d}_2 \bar{d}_1 + 1$.O complemento de 2 de um número é o (complemento de 1) mais 1.
- Fixamos n bits, complemento de 2 de 000 é $111 + 1 = \not{1}000$.
- **Exemplo**: complemento de 2 de 101_2 é $\overline{101} + 1 = 011$
- Com n bits, o complemento de 2 de $x < 2^n$ é $y = 2^n - x$. $(1111 - x + 1 = 10000 - x)$
- Como $x = 2^n - y$, $y = 2^n - x$: complemento de 2 é uma **involução**.

Binários Negativos via Complemento de 2

- Fixamos **n** bits: um de sinal, $n-1$ de "magnitude".

Binários Negativos via Complemento de 2

- Fixamos **n** bits: um de sinal, $n-1$ de "magnitude".
- Qualquer não-negativo (até um limite) é representado normalmente.

Binários Negativos via Complemento de 2

- Fixamos **n** bits: um de sinal, $n-1$ de "magnitude".
- Qualquer não-negativo (até um limite) é representado normalmente.
- **Exemplo**: se $n = 4$, $0110 = 6_{10}$.

Binários Negativos via Complemento de 2

- Fixamos **n** bits: um de sinal, $n-1$ de "magnitude".
- Qualquer não-negativo (até um limite) é representado normalmente.
- **Exemplo**: se $n = 4$, $0110 = 6_{10}$.
- Um negativo $x < 0$ é representado pelo **complemento de 2** de $|x|$.

Binários Negativos via Complemento de 2

- Fixamos **n** bits: um de sinal, $n-1$ de "magnitude".
- Qualquer não-negativo (até um limite) é representado normalmente.
- **Exemplo**: se $n = 4$, $0110 = 6_{10}$.
- Um negativo $x < 0$ é representado pelo **complemento de 2** de $|x|$.
- **Exemplo**: Com 4 bits, para representar -6_{10}, complementamos $0110_2 = 6_{10}$ de 2 e obtemos 1010.

Binários Negativos via Complemento de 2

- Fixamos **n** bits: um de sinal, $n-1$ de "magnitude".
- Qualquer não-negativo (até um limite) é representado normalmente.
- **Exemplo**: se $n = 4$, $0110 = 6_{10}$.
- Um negativo $x < 0$ é representado pelo **complemento de 2** de $|x|$.
- **Exemplo**: Com 4 bits, para representar -6_{10}, complementamos $0110_2 = 6_{10}$ de 2 e obtemos 1010.
- Primeiro bit igual a 1 indica número negativo .

Binários Negativos via Complemento de 2

- Fixamos **n** bits: um de sinal, $n-1$ de "magnitude".
- Qualquer não-negativo (até um limite) é representado normalmente.
- **Exemplo**: se $n = 4$, $0110 = 6_{10}$.
- Um negativo $x < 0$ é representado pelo **complemento de 2** de $|x|$.
- **Exemplo**: Com 4 bits, para representar -6_{10}, complementamos $0110_2 = 6_{10}$ de 2 e obtemos 1010.
- Primeiro bit igual a 1 indica número negativo .
- Para "decodificar" um negativo, vemos seu complemento de 2 em binário. **Exemplo**: complemento de 2 de 1010 é $0110_2 = 6_{10}$, então $1010 = -6_{10}$.

Binários Negativos via Complemento de 2

- Fixamos **n** bits: um de sinal, $n-1$ de "magnitude".
- Qualquer não-negativo (até um limite) é representado normalmente.
- **Exemplo**: se $n = 4$, $0110 = 6_{10}$.
- Um negativo $x < 0$ é representado pelo **complemento de 2** de $|x|$.
- **Exemplo**: Com 4 bits, para representar -6_{10}, complementamos $0110_2 = 6_{10}$ de 2 e obtemos 1010.
- Primeiro bit igual a 1 indica número negativo .
- Para "decodificar" um negativo, vemos seu complemento de 2 em binário. **Exemplo**: complemento de 2 de 1010 é $0110_2 = 6_{10}$, então $1010 = -6_{10}$.
- Complemento de 2 de 1000 é $1000_2 = 8_{10}$, então $1000 = -8_{10}$

Alternativas para decodificar negativos

- Fixamos **n** bits.

Alternativas para decodificar negativos

- Fixamos **n** bits.
- O complemento de 2 de $x = -|x| < 0$ é $y = 2^n - |x| = 2^n + x$. Logo, $x = y - 2^n$.

Alternativas para decodificar negativos

- Fixamos **n** bits.
- O complemento de 2 de $x = -|x| < 0$ é $y = 2^n - |x| = 2^n + x$. Logo, $x = y - 2^n$.
- **Exemplo**: $n = 4$, $1010_2 = 10$. $10 - 2^4 = -6$.

Alternativas para decodificar negativos

- Fixamos **n** bits.
- O complemento de 2 de $x = -|x| < 0$ é $y = 2^n - |x| = 2^n + x$. Logo, $x = y - 2^n$.
- **Exemplo**: $n = 4$, $1010_2 = 10$. $10 - 2^4 = -6$.
- Como $2^4 = 2^3 + 2^3$, $10 - 2^4 = 10 - 2^3 - 2^3 = 0010_2 - 2^3$.

Alternativas para decodificar negativos

- Fixamos **n** bits.
- O complemento de 2 de $x = -|x| < 0$ é $y = 2^n - |x| = 2^n + x$. Logo, $x = y - 2^n$.
- **Exemplo**: $n = 4$, $1010_2 = 10$. $10 - 2^4 = -6$.
- Como $2^4 = 2^3 + 2^3$, $10 - 2^4 = 10 - 2^3 - 2^3 = 0010_2 - 2^3$.
- $1010 = 0 \times 2^0 + 1 \times 2^1 + 0 \times 2^2$**-1** $\times 2^3$

Alternativas para decodificar negativos

- Fixamos **n** bits.
- O complemento de 2 de $x = -|x| < 0$ é $y = 2^n - |x| = 2^n + x$. Logo, $x = y - 2^n$.
- **Exemplo**: $n = 4$, $1010_2 = 10$. $10 - 2^4 = -6$.
- Como $2^4 = 2^3 + 2^3$, $10 - 2^4 = 10 - 2^3 - 2^3 = 0010_2 - 2^3$.
- $1010 = 0 \times 2^0 + 1 \times 2^1 + 0 \times 2^2$**-1** $\times 2^3$
- **Generalizando**: $b_{n-1} b_{n-2} \ldots b_0 = \sum_{i=0}^{n-2} b_i 2^i - b_{n-1} 2^{n-1}$

Alternativas para decodificar negativos

- Fixamos **n** bits.
- O complemento de 2 de $x = -|x| < 0$ é $y = 2^n - |x| = 2^n + x$. Logo, $x = y - 2^n$.
- **Exemplo**: $n = 4$, $1010_2 = 10$. $10 - 2^4 = -6$.
- Como $2^4 = 2^3 + 2^3$, $10 - 2^4 = 10 - 2^3 - 2^3 = 0010_2 - 2^3$.
- $1010 = 0 \times 2^0 + 1 \times 2^1 + 0 \times 2^2$**-1** $\times 2^3$
- **Generalizando**: $b_{n-1}b_{n-2}\ldots b_0 = \sum_{i=0}^{n-2} b_i 2^i - b_{n-1}2^{n-1}$
- Não negativos, $b_{n-1} = 0$, nada se altera.

Limites do Sistema de Complemento de 2

- Fixamos **n** bits.

Limites do Sistema de Complemento de 2

- Fixamos **n** bits.
- **Não-negativos**: 0 seguido de n-1 bits: $0 \leq x \leq 2^{n-1} - 1$.

Limites do Sistema de Complemento de 2

- Fixamos **n** bits.
- **Não-negativos**: 0 seguido de n-1 bits: $0 \leq x \leq 2^{n-1} - 1$.
- Exemplo: se $n = 4$, $0000_2 = 0$ a $0111_2 = 7_{10}$.

Limites do Sistema de Complemento de 2

- Fixamos **n** bits.
- **Não-negativos**: 0 seguido de n-1 bits: $0 \leq x \leq 2^{n-1} - 1$.
- Exemplo: se $n = 4$, $0000_2 = 0$ a $0111_2 = 7_{10}$.
- **Negativos**: $10\ldots0 = -2^{n-1}$ a $1\ldots1 = -1$.

Limites do Sistema de Complemento de 2

- Fixamos **n** bits.
- **Não-negativos**: 0 seguido de n-1 bits: $0 \leq x \leq 2^{n-1} - 1$.
- Exemplo: se $n = 4$, $0000_2 = 0$ a $0111_2 = 7_{10}$.
- **Negativos**: $10\ldots0 = -2^{n-1}$ a $1\ldots1 = -1$.
- Exemplo: se $n = 4$, $1000 = -8_{10}$ a $1111_2 = -1_{10}$.

Limites do Sistema de Complemento de 2

- Fixamos **n** bits.
- **Não-negativos**: 0 seguido de n-1 bits: $0 \leq x \leq 2^{n-1} - 1$.
- Exemplo: se $n = 4$, $0000_2 = 0$ a $0111_2 = 7_{10}$.
- **Negativos**: $10\ldots0 = -2^{n-1}$ a $1\ldots1 = -1$.
- Exemplo: se $n = 4$, $1000 = -8_{10}$ a $1111_2 = -1_{10}$.
- A **negação** de um número x é $-x$.

Limites do Sistema de Complemento de 2

- Fixamos **n** bits.
- **Não-negativos**: 0 seguido de n-1 bits: $0 \leq x \leq 2^{n-1} - 1$.
- Exemplo: se $n = 4$, $0000_2 = 0$ a $0111_2 = 7_{10}$.
- **Negativos**: $10\ldots0 = -2^{n-1}$ a $1\ldots1 = -1$.
- Exemplo: se $n = 4$, $1000 = -8_{10}$ a $1111_2 = -1_{10}$.
- A **negação** de um número x é $-x$.
- Negar é tomar o complemento de 2.

Limites do Sistema de Complemento de 2

- Fixamos **n** bits.
- **Não-negativos**: 0 seguido de n-1 bits: $0 \leq x \leq 2^{n-1} - 1$.
- Exemplo: se $n = 4$, $0000_2 = 0$ a $0111_2 = 7_{10}$.
- **Negativos**: $10\ldots0 = -2^{n-1}$ a $1\ldots1 = -1$.
- Exemplo: se $n = 4$, $1000 = -8_{10}$ a $1111_2 = -1_{10}$.
- A **negação** de um número x é $-x$.
- Negar é tomar o complemento de 2.
- Exemplos: negação de $1010 = -6_{10}$ é $0110 = 6_{10}$.

Limites do Sistema de Complemento de 2

- Fixamos **n** bits.
- **Não-negativos**: 0 seguido de n-1 bits: $0 \leq x \leq 2^{n-1} - 1$.
- Exemplo: se $n = 4$, $0000_2 = 0$ a $0111_2 = 7_{10}$.
- **Negativos**: $10\ldots0 = -2^{n-1}$ a $1\ldots1 = -1$.
- Exemplo: se $n = 4$, $1000 = -8_{10}$ a $1111_2 = -1_{10}$.
- A **negação** de um número x é $-x$.
- Negar é tomar o complemento de 2.
- Exemplos: negação de $1010 = -6_{10}$ é $0110 = 6_{10}$.
- Negação de $001111 = 15_{10}$ é $110001 = -15_{10}$.

Adicionando bits

- Positivos (e o 0) começam com 0, Negativos, com 1.

Adicionando bits

- Positivos (e o 0) começam com 0, Negativos, com 1.
- Queremos adicionar bits e representar o mesmo número.

Adicionando bits

- Positivos (e o 0) começam com 0, Negativos, com 1.
- Queremos adicionar bits e representar o mesmo número.
- **Não-negativos**: Adicionamos 0's a esquerda.

Adicionando bits

- Positivos (e o 0) começam com 0, Negativos, com 1.
- Queremos adicionar bits e representar o mesmo número.
- **Não-negativos**: Adicionamos 0's a esquerda.
- $4 = 0100$ com 4 bits, e $4 = \mathbf{0}0100$ com 5.

Adicionando bits

- Positivos (e o 0) começam com 0, Negativos, com 1.
- Queremos adicionar bits e representar o mesmo número.
- **Não-negativos**: Adicionamos 0's a esquerda.
- $4 = 0100$ com 4 bits, e $4 = \mathbf{0}0100$ com 5.
- **Negativos**: Adicionamos 1's a esquerda.

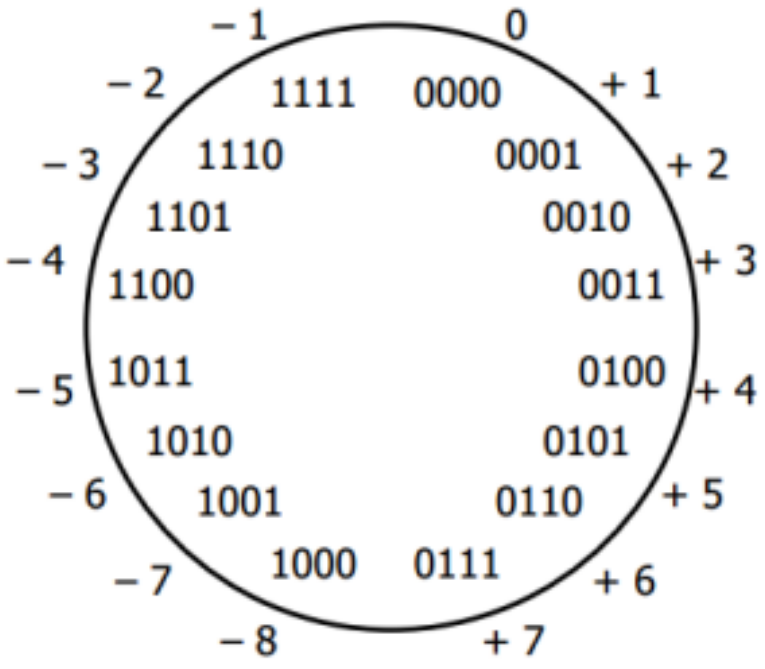
Adicionando bits

- Positivos (e o 0) começam com 0, Negativos, com 1.
- Queremos adicionar bits e representar o mesmo número.
- **Não-negativos**: Adicionamos 0's a esquerda.
- $4 = 0100$ com 4 bits, e $4 = \mathbf{0}0100$ com 5.
- **Negativos**: Adicionamos 1's a esquerda.
- $-3 = 1101$ com 4 bits, e $-3 = \mathbf{1}1101$ com 5.

Adicionando bits

- Positivos (e o 0) começam com 0, Negativos, com 1.
- Queremos adicionar bits e representar o mesmo número.
- **Não-negativos**: Adicionamos 0's a esquerda.
- $4 = 0100$ com 4 bits, e $4 = \mathbf{0}0100$ com 5.
- **Negativos**: Adicionamos 1's a esquerda.
- $-3 = 1101$ com 4 bits, e $-3 = \mathbf{1}1101$ com 5.
- **Pergunta**: Podemos tirar bits?

Aritmética modular do Complemento de 2



- Se **ignorarmos** o **vai-um** e o **empresta-um**: Somar é andar no sentido horário, e subtrair, no sentido anti-horário.

Somando no Sistema de Complemento de 2

- Parcelas positivas, nada se altera.

Somando no Sistema de Complemento de 2

- Parcelas positivas, nada se altera.

Fixando **n=3** bits.

Comp-2	100	101	110	111	000	001	010	011
Decimal	-4	-3	-2	-1	0	1	2	3

Somando no Sistema de Complemento de 2

- Parcelas positivas, nada se altera.

Fixando **n=3** bits.

Comp-2	100	101	110	111	000	001	010	011
Decimal	-4	-3	-2	-1	0	1	2	3

- Para qualquer $-4 \leq x \leq +2$, a representação de x+1 é obtida somando 1 a representação de x.

Somando no Sistema de Complemento de 2

- Parcelas positivas, nada se altera.

Fixando **n=3** bits.

Comp-2	100	101	110	111	000	001	010	011
Decimal	-4	-3	-2	-1	0	1	2	3

- Para qualquer $-4 \leq x \leq +2$, a representação de x+1 é obtida somando 1 a representação de x.
- $x = -3 = 101$, $x + 1 = -2 = 110$.

Somando no Sistema de Complemento de 2

- Parcelas positivas, nada se altera.

Fixando **n=3** bits.

Comp-2	100	101	110	111	000	001	010	011
Decimal	-4	-3	-2	-1	0	1	2	3

- Para qualquer $-4 \leq x \leq +2$, a representação de x+1 é obtida somando 1 a representação de x.
- $x = -3 = 101$, $x + 1 = -2 = 110$.
- **Só 3 bits**: x=-1=111, x+1=0=$\not{1}$000.

Somando no Sistema de Complemento de 2

- Parcelas positivas, nada se altera.

Fixando **n=3** bits.

Comp-2	100	101	110	111	000	001	010	011
Decimal	-4	-3	-2	-1	0	1	2	3

- Para qualquer $-4 \leq x \leq +2$, a representação de x+1 é obtida somando 1 a representação de x.
- $x = -3 = 101$, $x + 1 = -2 = 110$.
- **Só 3 bits**: x=-1=111, x+1=0=$\not{1}$000.
- Queremos somar $x < 0$ com $y \geq 0$.

Somando no Sistema de Complemento de 2

- Parcelas positivas, nada se altera.

Fixando **n=3** bits.

Comp-2	100	101	110	111	000	001	010	011
Decimal	-4	-3	-2	-1	0	1	2	3

- Para qualquer $-4 \leq x \leq +2$, a representação de x+1 é obtida somando 1 a representação de x.
- $x = -3 = 101$, $x + 1 = -2 = 110$.
- **Só 3 bits**: x=-1=111, x+1=0=~~1~~000.
- Queremos somar $x < 0$ com $y \geq 0$.
- $y = 1 + \cdots + 1, x + y = (\ldots((x + 1) + 1) + \ldots) + 1$

Somando no Sistema de Complemento de 2

- Parcelas positivas, nada se altera.

Fixando **n=3** bits.

Comp-2	100	101	110	111	000	001	010	011
Decimal	-4	-3	-2	-1	0	1	2	3

- Para qualquer $-4 \leq x \leq +2$, a representação de x+1 é obtida somando 1 a representação de x.
- $x = -3 = 101$, $x + 1 = -2 = 110$.
- **Só 3 bits**: x=-1=111, x+1=0=~~1~~000.
- Queremos somar $x < 0$ com $y \geq 0$.
- $y = 1 + \cdots + 1, x + y = (\ldots((x + 1) + 1) + \ldots) + 1$
- Representar de $x + y$ é somar y 1's à representação de x.

Somando no Sistema de Complemento de 2

- Parcelas positivas, nada se altera.

Fixando **n=3** bits.

Comp-2	100	101	110	111	000	001	010	011
Decimal	-4	-3	-2	-1	0	1	2	3

- Para qualquer $-4 \leq x \leq +2$, a representação de x+1 é obtida somando 1 a representação de x.
- $x = -3 = 101$, $x + 1 = -2 = 110$.
- **Só 3 bits**: x=-1=111, x+1=0=$\not{1}$000.
- Queremos somar $x < 0$ com $y \geq 0$.
- $y = 1 + \cdots + 1, x + y = (\ldots((x + 1) + 1) + \ldots) + 1$
- Representar de $x + y$ é somar y 1's à representação de x.
- Representação de $x + y$ é a soma das representações de x e y.

Somando dois negativos

Fixando **n=4** bits.

palavra	1000	1001	1010	1011	1100	1101	1110	1111
Comp-2	-8	-7	-6	-5	-4	-3	-2	-1
Binário	8	9	10	11	12	13	14	15
Binário	16-8	16-7	16-6	16-5	16-4	16-3	16-2	16-1

Somando dois negativos

Fixando **n=4** bits.

palavra	1000	1001	1010	1011	1100	1101	1110	1111
Comp-2	-8	-7	-6	-5	-4	-3	-2	-1
Binário	8	9	10	11	12	13	14	15
Binário	16-8	16-7	16-6	16-5	16-4	16-3	16-2	16-1

- Somar as representações de -2 e -5 é somar $1110_2 = 16 - 2$ e $1011_2 = 16 - 5$.

Somando dois negativos

Fixando **n=4** bits.

palavra	1000	1001	1010	1011	1100	1101	1110	1111
Comp-2	-8	-7	-6	-5	-4	-3	-2	-1
Binário	8	9	10	11	12	13	14	15
Binário	16-8	16-7	16-6	16-5	16-4	16-3	16-2	16-1

- Somar as representações de -2 e -5 é somar $1110_2 = 16 - 2$ e $1011_2 = 16 - 5$.
- Bit extra: $1110 + 1011 = 11001_2 = 25 = 16 + 16 - 7$.

Somando dois negativos

Fixando **n=4** bits.

palavra	1000	1001	1010	1011	1100	1101	1110	1111
Comp-2	-8	-7	-6	-5	-4	-3	-2	-1
Binário	8	9	10	11	12	13	14	15
Binário	16-8	16-7	16-6	16-5	16-4	16-3	16-2	16-1

- Somar as representações de -2 e -5 é somar $1110_2 = 16 - 2$ e $1011_2 = 16 - 5$.
- Bit extra: $1110 + 1011 = 11001_2 = 25 = 16 + 16 - 7$.
- **Descartando o bit extra**: $\not{1}1001_2 = 9 = 16 - 7$.

Somando dois negativos

Fixando **n=4** bits.

palavra	1000	1001	1010	1011	1100	1101	1110	1111
Comp-2	-8	-7	-6	-5	-4	-3	-2	-1
Binário	8	9	10	11	12	13	14	15
Binário	16-8	16-7	16-6	16-5	16-4	16-3	16-2	16-1

- Somar as representações de -2 e -5 é somar $1110_2 = 16 - 2$ e $1011_2 = 16 - 5$.
- Bit extra: $1110 + 1011 = 11001_2 = 25 = 16 + 16 - 7$.
- **Descartando o bit extra**: $\not{1}1001_2 = 9 = 16 - 7$.
- Representação de $-x$ mais a representação de $-y$ é $16 + 16 - x - y$.

Somando dois negativos

Fixando **n=4** bits.

palavra	1000	1001	1010	1011	1100	1101	1110	1111
Comp-2	-8	-7	-6	-5	-4	-3	-2	-1
Binário	8	9	10	11	12	13	14	15
Binário	16-8	16-7	16-6	16-5	16-4	16-3	16-2	16-1

- Somar as representações de -2 e -5 é somar $1110_2 = 16 - 2$ e $1011_2 = 16 - 5$.
- Bit extra: $1110 + 1011 = 11001_2 = 25 = 16 + 16 - 7$.
- **Descartando o bit extra**: $\not{1}1001_2 = 9 = 16 - 7$.
- Representação de $-x$ mais a representação de $-y$ é $16 + 16 - x - y$.
- **Descartando o bit extra**: $16 - x - y$ é a representação de $-x - y$

Somando quaisquer números

- **Conclusão:** Com complemento de 2, podemos somar negativos como se fossem positivos.

Somando quaisquer números

- **Conclusão:** Com complemento de 2, podemos somar negativos como se fossem positivos.
- **Importante**: Temos que ignorar o "vai-um".

Somando quaisquer números

- **Conclusão:** Com complemento de 2, podemos somar negativos como se fossem positivos.
- **Importante**: Temos que ignorar o "vai-um".
- **Exemplo**: 4 bits, calcular 6+(-7) no sistema de complemento de 2.

Somando quaisquer números

- **Conclusão:** Com complemento de 2, podemos somar negativos como se fossem positivos.
- **Importante**: Temos que ignorar o "vai-um".
- **Exemplo**: 4 bits, calcular 6+(-7) no sistema de complemento de 2.
- $7 = 0111$; tomando o complemento de 2: $-7 = 1001$

Somando quaisquer números

- **Conclusão:** Com complemento de 2, podemos somar negativos como se fossem positivos.
- **Importante**: Temos que ignorar o "vai-um".
- **Exemplo**: 4 bits, calcular 6+(-7) no sistema de complemento de 2.
- $7 = 0111$; tomando o complemento de 2: $-7 = 1001$
- Somando $6 = 0110$ com $-7 = 1001$, temos $1111 = -1$.

Somando quaisquer números

- **Conclusão:** Com complemento de 2, podemos somar negativos como se fossem positivos.
- **Importante**: Temos que ignorar o "vai-um".
- **Exemplo**: 4 bits, calcular 6+(-7) no sistema de complemento de 2.
- $7 = 0111$; tomando o complemento de 2: $-7 = 1001$
- Somando $6 = 0110$ com $-7 = 1001$, temos $1111 = -1$.
- **Exercício**: Calcular, em complemento de 2, $0010 + 1111$ e $1010 + 1110$ e conferir com os equivalentes decimais.

Subtração via Soma da Negação

- **Ideia**: $X - Y = X + (-Y)$

Subtração via Soma da Negação

- **Ideia**: $X - Y = X + (-Y)$
- $X - Y$ é a soma de X com a **negação** de Y.

Subtração via Soma da Negação

- **Ideia**: $X - Y = X + (-Y)$
- $X - Y$ é a soma de X com a **negação** de Y.
- Negação é computada como **Complemento de 2**.

Subtração via Soma da Negação

- **Ideia**: $X - Y = X + (-Y)$
- $X - Y$ é a soma de X com a **negação** de Y.
- Negação é computada como **Complemento de 2**.
- **Exemplo**: 4 bits, calcular 6-7 no sistema de complemento de 2.

Subtração via Soma da Negação

- **Ideia**: $X - Y = X + (-Y)$
- $X - Y$ é a soma de X com a **negação** de Y.
- Negação é computada como **Complemento de 2**.
- **Exemplo**: 4 bits, calcular 6-7 no sistema de complemento de 2.
- $7 = 0111$; tomando o complemento de 2: $-7 = 1001$

Subtração via Soma da Negação

- **Ideia**: $X - Y = X + (-Y)$
- $X - Y$ é a soma de X com a **negação** de Y.
- Negação é computada como **Complemento de 2**.
- **Exemplo**: 4 bits, calcular 6-7 no sistema de complemento de 2.
- $7 = 0111$; tomando o complemento de 2: $-7 = 1001$
- Somando $6 = 0110$ com $-7 = 1001$, temos $1111 = -1$.

Overflow no Sistema de Complemento de 2

- Fixamos **n** bits, n-1 para magnitude.

Overflow no Sistema de Complemento de 2

- Fixamos **n** bits, n-1 para magnitude.
- **Não-negativos**: de $00\ldots0_2 = 0$ a $01\ldots1_2 = 2^{n-1} - 1$.

Overflow no Sistema de Complemento de 2

- Fixamos **n** bits, n-1 para magnitude.
- **Não-negativos**: de $00\ldots0_2 = 0$ a $01\ldots1_2 = 2^{n-1} - 1$.
- **Negativos**: de $10\ldots0 = -2^{n-1}$ a $1\ldots1 = -1$.

Overflow no Sistema de Complemento de 2

- Fixamos **n** bits, n-1 para magnitude.
- **Não-negativos**: de $00\ldots0_2 = 0$ a $01\ldots1_2 = 2^{n-1} - 1$.
- **Negativos**: de $10\ldots0 = -2^{n-1}$ a $1\ldots1 = -1$.
- **Exemplo**: 4 bits, $0 = 0000$ a $2^3 - 1 = 7 = 0111$ e $-2^3 = -8 = 1000$ a $-1 = 1111$

Overflow no Sistema de Complemento de 2

- Fixamos **n** bits, n-1 para magnitude.
- **Não-negativos**: de $00\ldots0_2 = 0$ a $01\ldots1_2 = 2^{n-1} - 1$.
- **Negativos**: de $10\ldots0 = -2^{n-1}$ a $1\ldots1 = -1$.
- **Exemplo**: 4 bits, $0 = 0000$ a $2^3 - 1 = 7 = 0111$ e $-2^3 = -8 = 1000$ a $-1 = 1111$
- Quando o resultado esperado sai dessas faixas, temos **overflow**!

Overflow no Sistema de Complemento de 2

- Fixamos **n** bits, n-1 para magnitude.
- **Não-negativos**: de $00\ldots0_2 = 0$ a $01\ldots1_2 = 2^{n-1} - 1$.
- **Negativos**: de $10\ldots0 = -2^{n-1}$ a $1\ldots1 = -1$.
- **Exemplo**: 4 bits, $0 = 0000$ a $2^3 - 1 = 7 = 0111$ e $-2^3 = -8 = 1000$ a $-1 = 1111$
- Quando o resultado esperado sai dessas faixas, temos **overflow**!
- Exemplos: $1010 = -6_{10}$ mais $1101 = -3_{10}$ é $\not1 0111 = 7$

Overflow no Sistema de Complemento de 2

- Fixamos **n** bits, n-1 para magnitude.
- **Não-negativos**: de $00\ldots0_2 = 0$ a $01\ldots1_2 = 2^{n-1} - 1$.
- **Negativos**: de $10\ldots0 = -2^{n-1}$ a $1\ldots1 = -1$.
- **Exemplo**: 4 bits, $0 = 0000$ a $2^3 - 1 = 7 = 0111$ e $-2^3 = -8 = 1000$ a $-1 = 1111$
- Quando o resultado esperado sai dessas faixas, temos **overflow**!
- Exemplos: $1010 = -6_{10}$ mais $1101 = -3_{10}$ é ~~1~~$0111 = 7$???

Overflow no Sistema de Complemento de 2

- Fixamos **n** bits, n-1 para magnitude.
- **Não-negativos**: de $00\ldots0_2 = 0$ a $01\ldots1_2 = 2^{n-1} - 1$.
- **Negativos**: de $10\ldots0 = -2^{n-1}$ a $1\ldots1 = -1$.
- **Exemplo**: 4 bits, $0 = 0000$ a $2^3 - 1 = 7 = 0111$ e $-2^3 = -8 = 1000$ a $-1 = 1111$
- Quando o resultado esperado sai dessas faixas, temos **overflow**!
- Exemplos: $1010 = -6_{10}$ mais $1101 = -3_{10}$ é $\not{1}0111 = 7$???
- $0101 = 5_{10}$ mais $0100 = 4_{10}$ é $1001 = -7$

Overflow no Sistema de Complemento de 2

- Fixamos **n** bits, n-1 para magnitude.
- **Não-negativos**: de $00\ldots0_2 = 0$ a $01\ldots1_2 = 2^{n-1} - 1$.
- **Negativos**: de $10\ldots0 = -2^{n-1}$ a $1\ldots1 = -1$.
- **Exemplo**: 4 bits, $0 = 0000$ a $2^3 - 1 = 7 = 0111$ e $-2^3 = -8 = 1000$ a $-1 = 1111$
- Quando o resultado esperado sai dessas faixas, temos **overflow**!
- Exemplos: $1010 = -6_{10}$ mais $1101 = -3_{10}$ é $\not{1}0111 = 7$???
- $0101 = 5_{10}$ mais $0100 = 4_{10}$ é $1001 = -7$???

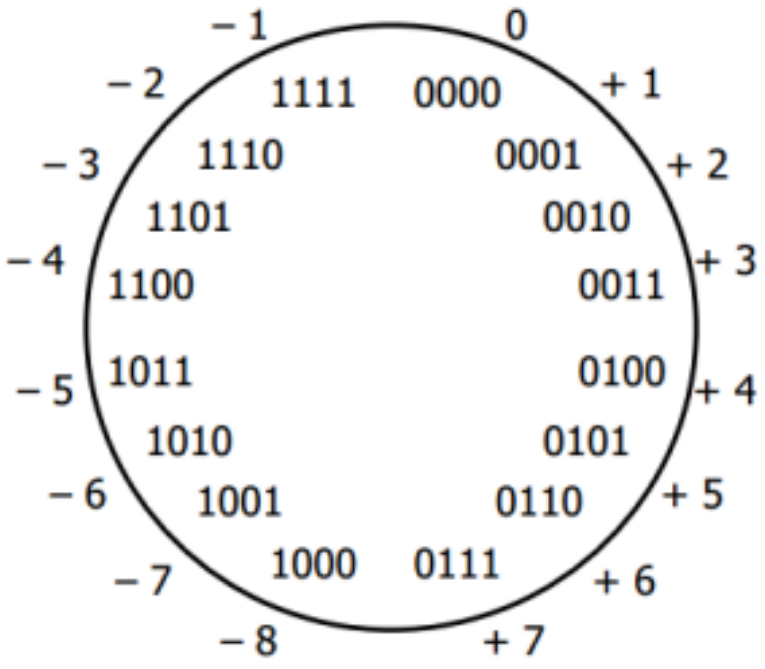
Overflow no Sistema de Complemento de 2

- Fixamos **n** bits, n-1 para magnitude.
- **Não-negativos**: de $00\ldots0_2 = 0$ a $01\ldots1_2 = 2^{n-1} - 1$.
- **Negativos**: de $10\ldots0 = -2^{n-1}$ a $1\ldots1 = -1$.
- **Exemplo**: 4 bits, $0 = 0000$ a $2^3 - 1 = 7 = 0111$ e $-2^3 = -8 = 1000$ a $-1 = 1111$
- Quando o resultado esperado sai dessas faixas, temos **overflow**!
- Exemplos: $\mathbf{1}010 = -6_{10}$ mais $\mathbf{1}101 = -3_{10}$ é $\not{1}\mathbf{0}111 = 7$???
- $0101 = 5_{10}$ mais $0100 = 4_{10}$ é $1001 = -7$???
- Para detectar overflow, basta analisar os **sinais**.

Overflow no Sistema de Complemento de 2

- Fixamos **n** bits, n-1 para magnitude.
- **Não-negativos**: de $00\ldots0_2 = 0$ a $01\ldots1_2 = 2^{n-1} - 1$.
- **Negativos**: de $10\ldots0 = -2^{n-1}$ a $1\ldots1 = -1$.
- **Exemplo**: 4 bits, $0 = 0000$ a $2^3 - 1 = 7 = 0111$ e $-2^3 = -8 = 1000$ a $-1 = 1111$
- Quando o resultado esperado sai dessas faixas, temos **overflow**!
- Exemplos: $1010 = -6_{10}$ mais $1101 = -3_{10}$ é $\not{1}0111 = 7$???
- $\mathbf{0}101 = 5_{10}$ mais $\mathbf{0}100 = 4_{10}$ é $\mathbf{1}001 = -7$???
- Para detectar overflow, basta analisar os **sinais**.

Overflow graficamente



- Overflow acontece quando passamos de $100\ldots00$ para $011\ldots11$ ou vice-versa.

De volta ao problema do ano 2038

- Sistemas baseados em Unix com 32 bits contam os segundos que se passaram desde à meia noite de 1/jan/1970.

De volta ao problema do ano 2038

- Sistemas baseados em Unix com 32 bits contam os segundos que se passaram desde à meia noite de 1/jan/1970.
- São usados 32 bits no sistema de complemento de 2.

De volta ao problema do ano 2038

- Sistemas baseados em Unix com 32 bits contam os segundos que se passaram desde à meia noite de 1/jan/1970.
- São usados 32 bits no sistema de complemento de 2.
- Às 3h14m07s de 19/jan/2038 terão se passado $2^{31} - 1$ segundos da data zero: $01111\ldots1111$

De volta ao problema do ano 2038

- Sistemas baseados em Unix com 32 bits contam os segundos que se passaram desde à meia noite de 1/jan/1970.
- São usados 32 bits no sistema de complemento de 2.
- Às 3h14m07s de 19/jan/2038 terão se passado $2^{31} - 1$ segundos da data zero: $01111\ldots1111$
- Somando um segundo: $100\ldots000 = -2^{31}$.

De volta ao problema do ano 2038

- Sistemas baseados em Unix com 32 bits contam os segundos que se passaram desde à meia noite de 1/jan/1970.
- São usados 32 bits no sistema de complemento de 2.
- Às 3h14m07s de 19/jan/2038 terão se passado $2^{31} - 1$ segundos da data zero: $01111\ldots1111$
- Somando um segundo: $100\ldots000 = -2^{31}$.
- Subtraindo (ou somando menos) 2^{31} segundos da meia-noite de 1/jan/1970, voltamos para 20h45m52s de 13/dez/1901.

Multiplicação no Sistema Complemento de 2

- Dois **positivos**: basta completar com 0's à frente para o primeiro bit ser sempre 0.

Multiplicação no Sistema Complemento de 2

- Dois **positivos**: basta completar com 0's à frente para o primeiro bit ser sempre 0.
- Jeito **fácil** para lidar com **negativos**: nega os negativos, multiplica e possivelmente nega o resultado.

Multiplicação no Sistema Complemento de 2

- Dois **positivos**: basta completar com 0's à frente para o primeiro bit ser sempre 0.
- Jeito **fácil** para lidar com **negativos**: nega os negativos, multiplica e possivelmente nega o resultado.
- **Exemplo**: multiplicar $1101 = -3$ e $0110 = 6$

Multiplicação no Sistema Complemento de 2

- Dois **positivos**: basta completar com 0's à frente para o primeiro bit ser sempre 0.
- Jeito **fácil** para lidar com **negativos**: nega os negativos, multiplica e possivelmente nega o resultado.
- **Exemplo**: multiplicar $1101 = -3$ e $0110 = 6$
- Negamos 1101: $0011 = 3$. Multiplicamos por $0110 = 6$.

Multiplicação no Sistema Complemento de 2

- Dois **positivos**: basta completar com 0's à frente para o primeiro bit ser sempre 0.
- Jeito **fácil** para lidar com **negativos**: nega os negativos, multiplica e possivelmente nega o resultado.
- **Exemplo**: multiplicar $1101 = -3$ e $0110 = 6$
- Negamos 1101: $0011 = 3$. Multiplicamos por $0110 = 6$.
- Obtemos $010010 = 18$, cuja negação é $101110 = -18$

Multiplicação no Sistema Complemento de 2

- Dois **positivos**: basta completar com 0's à frente para o primeiro bit ser sempre 0.
- Jeito **fácil** para lidar com **negativos**: nega os negativos, multiplica e possivelmente nega o resultado.
- **Exemplo**: multiplicar $1101 = -3$ e $0110 = 6$
- Negamos 1101: $0011 = 3$. Multiplicamos por $0110 = 6$.
- Obtemos $010010 = 18$, cuja negação é $101110 = -18$
- Fácil para quem? Sempre que há operando negativo, faz-se **duas negações**!

O jeito difícil...

- Caso **aparentemente difícil**: o multiplicando é negativo.

O jeito difícil...

- Caso **aparentemente difícil**: o multiplicando é negativo.
- Ambos soma e deslocamento funcionam no sistema complemento de 2!

O jeito difícil...

- Caso **aparentemente difícil**: o multiplicando é negativo.
- Ambos soma e deslocamento funcionam no sistema complemento de 2!
- Basta completar com 1's suficientes a frente das parcelas negativas (por quê?).

O jeito difícil...

- Caso **aparentemente difícil**: o multiplicando é negativo.
- Ambos soma e deslocamento funcionam no sistema complemento de 2!
- Basta completar com 1's suficientes a frente das parcelas negativas (por quê?).
- **Multiplicador negativo**: $(b_n \dots b_1 b_0) = -b_n 2^n + \sum_{i=0}^{n-1} b_i 2^i$.

O jeito difícil...

- Caso **aparentemente difícil**: o multiplicando é negativo.
- Ambos soma e deslocamento funcionam no sistema complemento de 2!
- Basta completar com 1's suficientes a frente das parcelas negativas (por quê?).
- **Multiplicador negativo**: $(b_n \dots b_1 b_0) = -b_n 2^n + \sum_{i=0}^{n-1} b_i 2^i$.
- Só se altera o último deslocamento, que é negado.

O jeito difícil...

- Caso **aparentemente difícil**: o multiplicando é negativo.
- Ambos soma e deslocamento funcionam no sistema complemento de 2!
- Basta completar com 1's suficientes a frente das parcelas negativas (por quê?).
- **Multiplicador negativo**: $(b_n \dots b_1 b_0) = -b_n 2^n + \sum_{i=0}^{n-1} b_i 2^i$.
- Só se altera o último deslocamento, que é negado.
- Fazemos no máximo **uma negação**.

O jeito difícil...

- Caso **aparentemente difícil**: o multiplicando é negativo.
- Ambos soma e deslocamento funcionam no sistema complemento de 2!
- Basta completar com 1's suficientes a frente das parcelas negativas (por quê?).
- **Multiplicador negativo**: $(b_n \dots b_1 b_0) = -b_n 2^n + \sum_{i=0}^{n-1} b_i 2^i$.
- Só se altera o último deslocamento, que é negado.
- Fazemos no máximo **uma negação**.
- **Exercício**: Multiplicar 101 por 111.

Onde estamos e para onde vamos

- Operações aritméticas com binários.

Onde estamos e para onde vamos

- Operações aritméticas com binários.
- Representação de binários negativos via sinal-magnitude.

Onde estamos e para onde vamos

- Operações aritméticas com binários.
- Representação de binários negativos via sinal-magnitude.
- Via complemento de 2.

Onde estamos e para onde vamos

- Operações aritméticas com binários.
- Representação de binários negativos via sinal-magnitude.
- Via complemento de 2.
- **Próxima aula**: Portas lógicas com transistores!

Referências e Sugestões de Leitura e Exercícios

- Wakerly, J.F..Digital Design, Pearson Prentice-Hall, 4° Ed, 2006.
  - Capítulo 2.