Binários:Representação de Negativos

PCS3115 - Sistemas Digitais 1

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

São Paulo, 2020

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 de 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 de 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 $\bar{d}_n \ldots \bar{d}_1$.

Complemento de 1 e de 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 $\bar{d}_n \ldots \bar{d}_1$.
- **Exemplo**: complemento de 1 de 101_2 é $= 010$

Complemento de 1 e de 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 $\bar{d}_n \ldots \bar{d}_1$.
- **Exemplo**: complemento de 1 de 101_2 é $= 010$
- 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 de 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 $\bar{d}_n \ldots \bar{d}_1$.
- **Exemplo**: complemento de 1 de 101_2 é $= 010$
- 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 de 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 $\bar{d}_n \dots \bar{d}_1$.
- **Exemplo**: complemento de 1 de 101_2 é $= 010$
- 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 $111 + 1 = 1000$.

Complemento de 1 e de 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 $\bar{d}_n \dots \bar{d}_1$.
- **Exemplo**: complemento de 1 de 101_2 é $= 010$
- 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 $111 + 1 = \cancel{1}000$.

Complemento de 1 e de 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 $\bar{d}_n \ldots \bar{d}_1$.
- **Exemplo**: complemento de 1 de 101_2 é $= 010$
- 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 $111 + 1 = \cancel{1}000$.
- **Exemplo**: complemento de 2 de 101_2 é $\bar{1}\bar{0}\bar{1} + 1 = 011$

Complemento de 1 e de 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 $\bar{d}_n \dots \bar{d}_1$.
- **Exemplo**: complemento de 1 de 101_2 é $= 010$
- 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 $111 + 1 = \cancel{1}000$.
- **Exemplo**: complemento de 2 de 101_2 é $\bar{1}\bar{0}\bar{1} + 1 = 011$
- Com n bits, o complemento de 2 de $0 < x$ é $y = 2^n - x$.

Complemento de 1 e de 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 $\bar{d}_n \dots \bar{d}_1$.
- **Exemplo**: complemento de 1 de 101_2 é $= 010$
- 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 $111 + 1 = \cancel{1}000$.
- **Exemplo**: complemento de 2 de 101_2 é $\bar{1}\bar{0}\bar{1} + 1 = 011$
- Com n bits, o complemento de 2 de $0 < x$ é $y = 2^n - x$. $(1111 - x + 1 = 10000 - x)$

Complemento de 1 e de 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 $\bar{d}_n \dots \bar{d}_1$.
- **Exemplo**: complemento de 1 de 101_2 é $= 010$
- 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 $111 + 1 = \cancel{1}000$.
- **Exemplo**: complemento de 2 de 101_2 é $\bar{1}\bar{0}\bar{1} + 1 = 011$
- Com n bits, o complemento de 2 de $0 < x$ 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 $|x|$, com n bits.

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 $|x|$, com n bits.
- **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 $|x|$, com n bits.
- **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 $|x|$, com n bits.
- **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 (sem sinal). **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 $|x|$, com n bits.
- **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 (sem sinal). **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 $1000 = -8_{10}$

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 $1111_2 = -1_{10}$.

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.

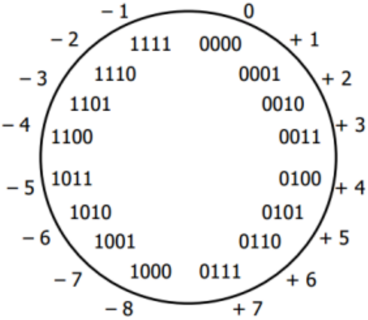
Adicionando bits

- 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 MÓDULO 2^n



- Se **ignorarmos** o **vai-um** e o **empresta-um** no bit mais significativo: 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.
- Somar 2^n é igual a somar 0 na arimética módulo 2^n (dar uma volta horária completa).

Somando no Sistema de Complemento de 2

- Parcelas positivas, nada se altera.
- Somar 2^n é igual a somar 0 na arimética módulo 2^n (dar uma volta horária completa).
- Em complemento de 2 com n bits, $-x$ é representado por $2^n - x$.

Somando no Sistema de Complemento de 2

- Parcelas positivas, nada se altera.
- Somar 2^n é igual a somar 0 na arimética módulo 2^n (dar uma volta horária completa).
- Em complemento de 2 com n bits, $-x$ é representado por $2^n - x$.
- Somar $2^n - x$ na aritmética módulo 2^n é igual a subtrair x (andar x posições no sentido anti-horário).

Somando no Sistema de Complemento de 2

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

Somando no Sistema de Complemento de 2

- **Conclusão:** Com complemento de 2, podemos somar negativos como se fossem positivos.
- **Importante**: Temos que ignorar o "vai-um" no bit mais significativo.

Somando no Sistema de Complemento de 2

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

Somando no Sistema de Complemento de 2

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

Somando no Sistema de Complemento de 2

- **Conclusão:** Com complemento de 2, podemos somar negativos como se fossem positivos.
- **Importante**: Temos que ignorar o "vai-um" no bit mais significativo.
- **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 no Sistema de Complemento de 2

- **Conclusão:** Com complemento de 2, podemos somar negativos como se fossem positivos.
- **Importante**: Temos que ignorar o "vai-um" no bit mais significativo.
- **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 $00\dots0_2 = 0$ a $01\dots1_2 = 2^{n-1} - 1$.
- **Negativos**: de $10\dots0 = -2^{n-1}$ a $1\dots1 = -1$.
- **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 $\cancel{1}0111 = 7$

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 $\cancel{1}0111 = 7$???

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 $\cancel{1}0111 = 7$???
- $0101 = 5_{10}$ 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 $\cancel{1}0111 = 7$???
- $0101 = 5_{10}$ mais $0100 = 4_{10}$ é $1001 = -7$???

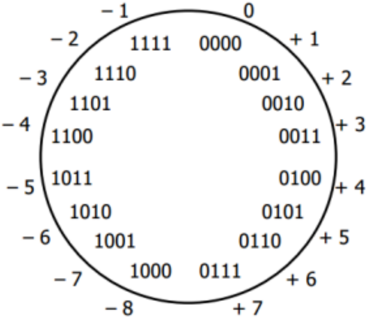
Overflow no Sistema de Complemento de 2

- 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 $\cancel{1}\mathbf{0}111 = 7$???
- $0101 = 5_{10}$ 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 $\cancel{1}0111 = 7$???
- $\mathbf{0}101 = 5_{10}$ 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.

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.
- Por que isso acontece?

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

- 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.

- 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: $01111\ldots1111$

- 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: $01111 \ldots 1111$
- Somando um segundo: $100 \ldots 000 = -2^{31}$.

- 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: $01111\dots1111$
- Somando um segundo: $100\dots000 = -2^{31}$.
- 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**: $(b_n \ldots b_1 b_0) = -b_n 2^n + \sum_{i=0}^{n-1} b_i 2^i$.

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**: $(b_n \ldots b_1 b_0) = -b_n 2^n + \sum_{i=0}^{n-1} b_i 2^i$.
- 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**: $(b_n \ldots b_1 b_0) = -b_n 2^n + \sum_{i=0}^{n-1} b_i 2^i$.
- Só se altera o último deslocamento, que é negado.
- Fazemos no máximo **uma negação**.
- **Exercício**: Multiplicar 101 por 111.

Referências e Sugestões de Leitura e Exercícios

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