Representações de Números Inteiros: Sinal e Magnitude e Representação em Excesso de k

Cristina Boeres

Instituto de Computação (UFF)

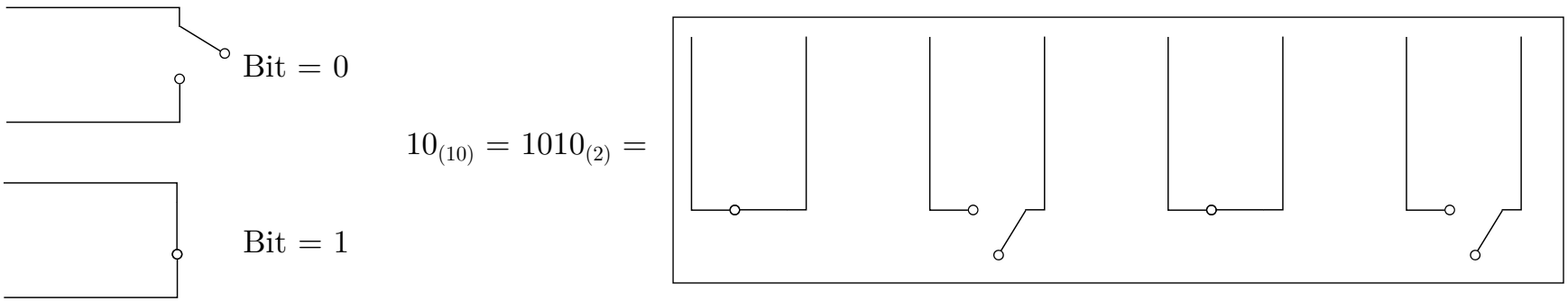
Fundamentos de Arquiteturas de Computadores

Computadores e Números

- Computadores modernos são, em sua enorme maioria, dispositivos digitais
  - Representam sinais como conjuntos de símbolos discretos
- Mais que isso, computadores são normalmente binários
  - Dois estados: tensão baixa ou tensão alta.
- Nas últimas aulas, vimos como números podem ser representados em base 2
  - Base numérica composta por dois símbolos (algarismos), os **bits**.

Computadores e Números (II)

- Com uma sequência ordenada de bits, podemos representar números quaisquer
  - Desde que o **número de bits** disponível seja **grande o suficiente**.



Computadores e Números (III)

- Note que por se tratar de circuitos eletrônicos, computadores usam um número fixo de bits para representar números:
  - 8 bits, 16 bits ou 32 bits.
- Se um número tem menos algarismos binários, ele e complementado com zeros à esquerda.
- Exemplos com 8 bits:
  - $200_{(10)} = 11001000_{(2)}$.
  - $100_{(10)} = 01100100_{(2)}$.
  - $50_{(10)} = 00110010_{(2)}$.
- Por outro lado, isso **limita o conjunto de números que podem ser representados**.

Computadores e Números (IV)

- Há necessidade de representação não somente de números inteiros positivos
  - O que fazer com os números **negativos**?
  - E com os números com parte **fracionária** não-nula?

- Devem ser definidos os **esquemas de representação**

Esquemas de Representação

- Maneira padronizada de codificar informações usando apenas bits
  - Valor que assume dois estados
- Esquemas:
  - Sinal e Magnitude
  - Representação em Excesso
  - Complemento a Um
  - **Complemento a Dois**
  - Representação em Ponto Fixo
  - **Representação em Ponto Flutuante**

Sinal e Magnitude

- A representação por Sinal e Magnitude é a mais intuitiva capaz de representar apenas números inteiros:
  - Mas tanto positivos, quanto negativos.

- Números são separados em duas componentes:
  - Sinal
    - ★ Como o sinal assume dois estados (negativo e positivo) – um bit para representá-lo
  - Magnitude (ou módulo, ou valor absoluto)
    - ★ Como se o número fosse positivo

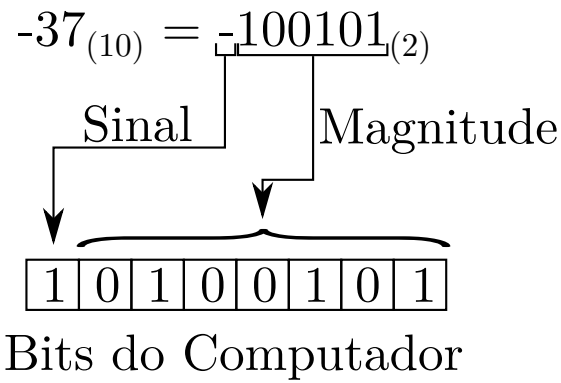
Convenções em Sinal e Magnitude

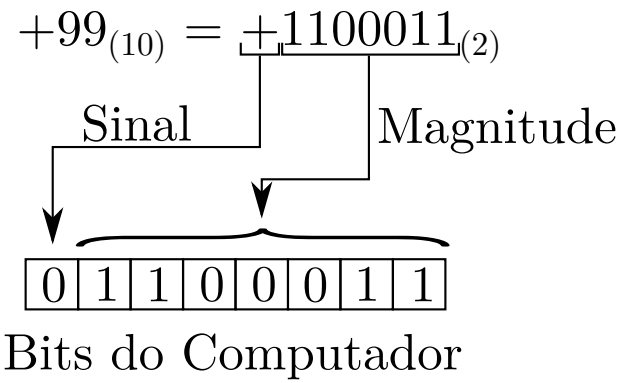
- Bit reservado para o sinal é sempre o mais significativo
  - *i.e.*, o mais à esquerda
- Em números positivos, esse bit é 0
- Em números negativos, esse bit é 1

- A magnitude é simplesmente representada em base 2 com $n - 1$ bits
  - Onde n é o número de bits usado pela máquina para representar números.

Exemplos de Representação

- Exemplos considerando 8 bits.





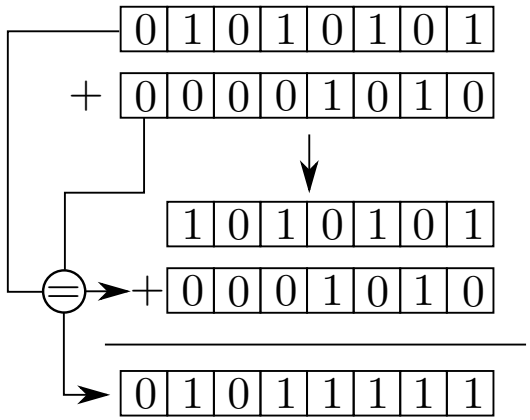
Fazendo Contas com Sinal e Magnitude

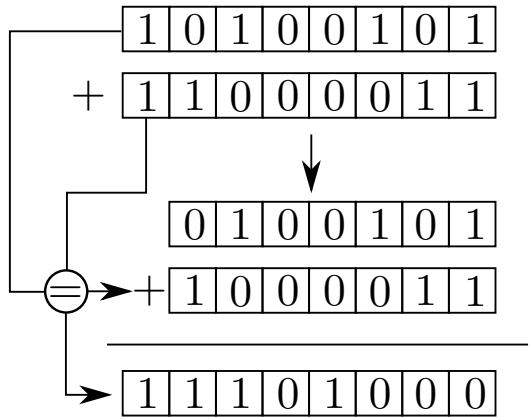
- Suponha que um computador deseje operar sobre dois números em Sinal e Magnitude.
  - Somar
  - Subtrair
  - Comparar
  - ...
- Como isso deve ser feito?
- Considere uma soma, por exemplo
  - Não podemos simplesmente somar os dois números algarismo a algarismo – um dos "algarismos" é o sinal

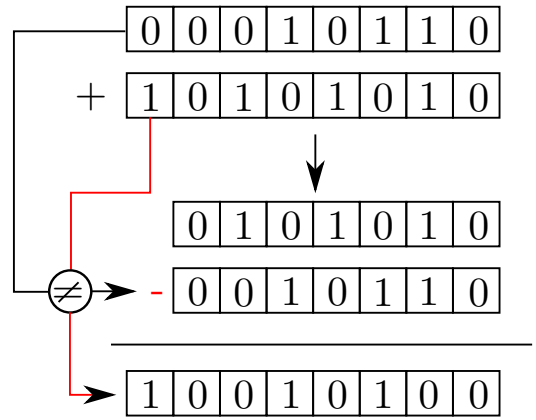
Fazendo Somas com Sinal e Magnitude

- Se os bits de sinal dos dois números são iguais
  - separe as magnitudes e as some
  - Bit de sinal do resultado é igual ao bit de sinal dos operandos.

- Caso contrário
  - isole as magnitudes e **subtraia** a menor da maior.
  - Bit de sinal do resultado é igual ao bit de sinal do operando de maior magnitude.







Fazendo Contas com Sinal e Magnitude

- Outras operações podem ser realizadas de forma similar
  - Para subtração, pode-se trocar o sinal do subtraendo (inverter bit de sinal) e executar mesmos passos da soma
  - Para divisão e multiplicação, opera-se apenas sobre a magnitude
    - ★ Se operandos têm mesmo sinal, resultado terá bit de sinal 0.
    - ★ Se operandos têm sinal diferente, resultado terá bit de sinal 1.

Sinal e Magnitude: Limites

- Suponha 8 bits para representar um número em Sinal e Magnitude
- Qual o maior número (maior positivo) que pode ser representado?
  - Em Sinal e Magnitude, primeiro bit 0 e todos os outros iguais a 1.
  - Para 8 bits: $0\ 1111111_{(2)} = 127_{(10)}$.
- Qual o menor número (negativo) que pode ser representado?
  - Em Sinal e Magnitude, o primeiro bit é 1 e todos os outros iguais a 1.
  - Para 8 bits: $-1111111_{(2)} = -127_{(10)}$.

Sinal e Magnitude: Limites (II)

- De forma mais genérica, com uma quantidade n qualquer de bits:
  - **Maior número:**

$$2^{(n-2)} + 2^{(n-3)} + \cdots + 2^0 = \boxed{2^{(n-1)} - 1}$$

  - **Menor número:**

$$-\left(2^{(n-2)} + 2^{(n-3)} + \cdots + 2^0\right) = \boxed{-\left(2^{(n-1)} - 1\right)}$$

- Exemplos para alguns valores de n:
  - Para $n = 4$: de -7 a 7.
  - Para $n = 16$: de -32767 a 32767.
  - Para $n = 32$: de -2147483647 a 2147483647

Sinal e Magnitude: Duplicidade do Valor 0

- Considere as seguintes sequências de bits:
  - 00000000 e 10000000.
- Assumindo que ambas são representações em Sinal e Magnitude com 8 bits, quais os valores representados?
- Vamos fazer a interpretação:
  - Primeiro número tem sinal positivo e magnitude 0.
  - Segundo número tem sinal negativo e magnitude 0.

- Conclusão: **ambos são 0**

Sinal e Magnitude: Usos

- Pela sua similaridade com a notação escrita, a representação em Sinal e Magnitude foi usada em alguns computadores antigos.
  - *e.g.*, IBM 7090 em 1959.
- Mas a duplicidade do valor 0 e a lógica "complicada" para certas operações matemáticas resultou em pouco popularidade.
  - Outras representações são mais simples.
  - **Note que a duplicidade do zero desperdiça bits.**
- Hoje, a maior importância desta representação é ser a base para a Representação em Ponto Flutuante.

Representação em Excesso de k

- Também chamada de *offset binário*
- Assim como a representação em Sinal e Magnitude, só permite representar números inteiros
  - Positivos, negativos e zero.
- Além do número de bits, é usado um outro parâmetro k
  - Chamado de **excesso**
- Para um dado parâmetro k, menor número representável será $-k$.

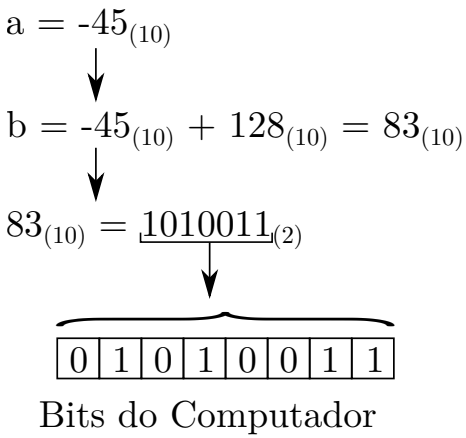
Representação em Excesso de k: como funciona

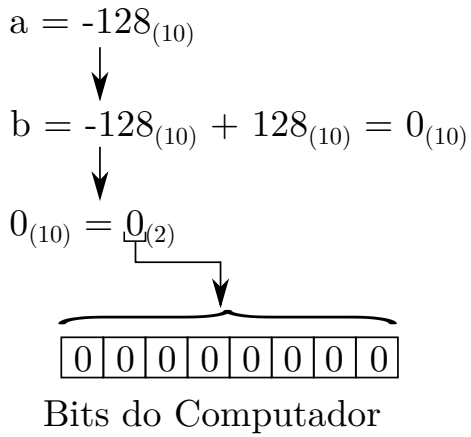
- Para representar um número a qualquer em Excesso de k com n bits:
  1. Obter um novo valor $b = a + k$.
  2. Representar b na base 2 com n bits
     - ★ Completar com zeros à esquerda, se necessário

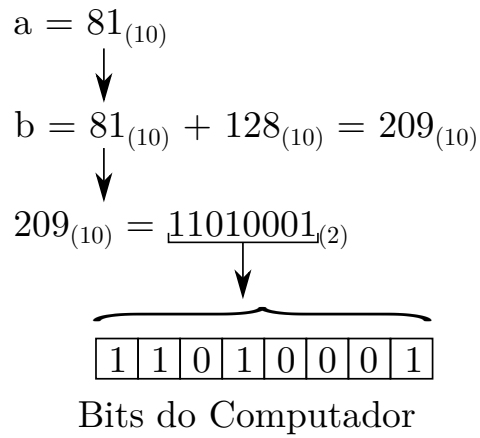
- Note que, por hipótese, $a \geq -k$.
- Logo, qualquer que seja a, $a + k \geq 0$.
  - Ou seja, b é necessariamente um **número não-negativo**

Representação em Excesso de k: exemplos

- Representação de 8 bits em excesso de 128:







Um Número na Representação em Excesso de k

- Dado um número escrito na representação em Excesso de k, podemos interpretá-lo facilmente:
  1. Converter a representação binária para decimal.
  2. Subtrair do excesso k.

- Quais os valores dos seguintes decimais representados em excesso de 64 com 8 bits?
  - 00101101.
    - ★ $00101101_{(2)} - 64_{(10)} = 45_{(10)} - 64_{(10)} = -19_{(10)}$.
  - 00010101.
    - ★ $00010101_{(2)} - 64_{(10)} = 21_{(10)} - 64_{(10)} = -43_{(10)}$.
  - 01110101.
    - ★ $01110101_{(2)} - 64_{(10)} = 117_{(10)} - 64_{(10)} = 53_{(10)}$.

Fazendo Contas com a Representação em Excesso de k

- Ao realizar contas na representação em excesso, precisamos lembrar do excesso k somado aos operandos.
- Por exemplo, se os dois operandos são a e b, na verdade a representação terá valores $a' = a + k$ e $b' = b + k$.

- Considere, por exemplo, uma soma:
  - Se somarmos diretamente os bits das representações, obteremos

$$a' + b' = a + k + b + k$$

  - O excesso fica **dobrado**
  - Logo, para corrigir o valor: **basta subtrair o excesso**

Fazendo Contas com a Representação em Excesso de k

- Podemos também inverter o sinal de um número.
  - Considere um número a com representação $a' = a + k$.
  - Queremos calcular a representação de $(-a)' = (-a) + k$.
  - Da primeira equação, temos $(-a) = k - a'$.
  - Substituindo na segunda, ficamos com $(-a)' = k - a' + k = 2k - a'$.

  - Dado um número representado em Excesso de k, basta subtraí-lo de $2k$

Fazendo Contas com a Representação em Excesso de k

- Para inverter um número representado em Excesso de k: subtraí de $2k$
- Inverter o sinal de 11010001 representado em excesso de 128 (Obs.: passando para decimal é $81_{(10)}$):
  - $2 \times 128_{(10)} = 100000000_{(2)}$.
  - $100000000_{(2)} - 11010001_{(2)} = 00101111_{(2)}$.
  - Traduzindo da representação em excesso: $00101111_{(2)} - 128_{(10)} = 47_{(10)} - 128_{(10)} = -81_{(10)}$.

Fazendo Contas com a Representação em Excesso de k

- Sabendo inverter o sinal de um número, é fácil realizar subtrações.
- Basta inverter o sinal do subtraendo e realizar a soma dos valores encontrados.

Representação em Excesso de k: Limites

- Qual o menor valor que pode ser escrito em uma representação em Excesso de k com n bits?
  - Já comentado anteriormente, o menor número que podemos representar é o $-k$. Logo:

$$-k + k = 0$$

    - ★ Tem representação com todos os bits zerados
    - ★ Independente do valor de n

Representação em Excesso de k: Limites

- E qual é o maior valor?
  - Neste caso, é o valor associado a uma representação contendo todos os bits iguais a 1.
    - ★ $11\ldots1$
  - Traduzindo:

$$11\ldots1_{(2)} - k = 2^{(n-1)} + 2^{(n-2)} + \cdots + 2^0 - k = 2^n - 1 - k$$

  - Depende tanto de k, quanto de n

Qual Valor de k Escolher?

- A princípio, pode-se escolher qualquer valor positivo de k.
- Note, no entanto, que k influencia no maior e menor valores representáveis.
  - Quanto maior o k, mais números negativos podem ser representados
  - Mas menos números positivos são viáveis
- Geralmente, opta-se por um k que balanceie a representabilidade entre números positivos e negativos.
- Para um dado número de bits n, por exemplo, pode-se escolher $k = 2^{(n-1)}$.
  - Números representáveis na faixa de $-2^{(n-1)}$ a $2^{(n-1)} - 1$.

Representação em Excesso de k: Usos

- Assim como a representação por Sinal e Magnitude, a representação em excesso não é utilizada para representar qualquer número
  - Suas operações ainda são relativamente complexas (desvantagem)
  - No entanto, **não** apresenta a duplicidade do zero (vantagem)

- Nos computadores modernos, ela é importante por fazer parte da **Representação em Ponto Flutuante**