Curso: Engenharia de Computação

Arquitetura de Computadores

Prof. Clayton J A Silva, MSc
clayton.silva@professores.ibmec.edu.br

Representação numérica

Representação de grandezas numéricas

- Notação **posicional**
- Notação **polinomial**

Representação de grandezas numéricas

- Notações **posicional e polinomial**

$$1457 = 1x10^3+4x10^2+5x10^1+7x10^0$$

+significativo

- Dígitos do conjunto *D={0,1,2,3,...,9}*, **base decimal**

Notação posicional

$$N_b = d_{m-1} d_{m-2} \ldots d_0$$

- *N* é a grandeza,
- *b* é a base,
- *m* é o número de dígitos usados na representação
- os índices *0* a *m-1* representam a posição do dígito.

Notação polinomial

$$N_b = d_{m-1} \times b^{m-1} + d_{m-2} \times b^{m-2} \ldots + d_0 \times b^0$$

- *N* é a grandeza,
- *b* é a base,
- *m* é o número de dígitos usados e
- os índices *0* a *m-1* representam a posição do dígito.

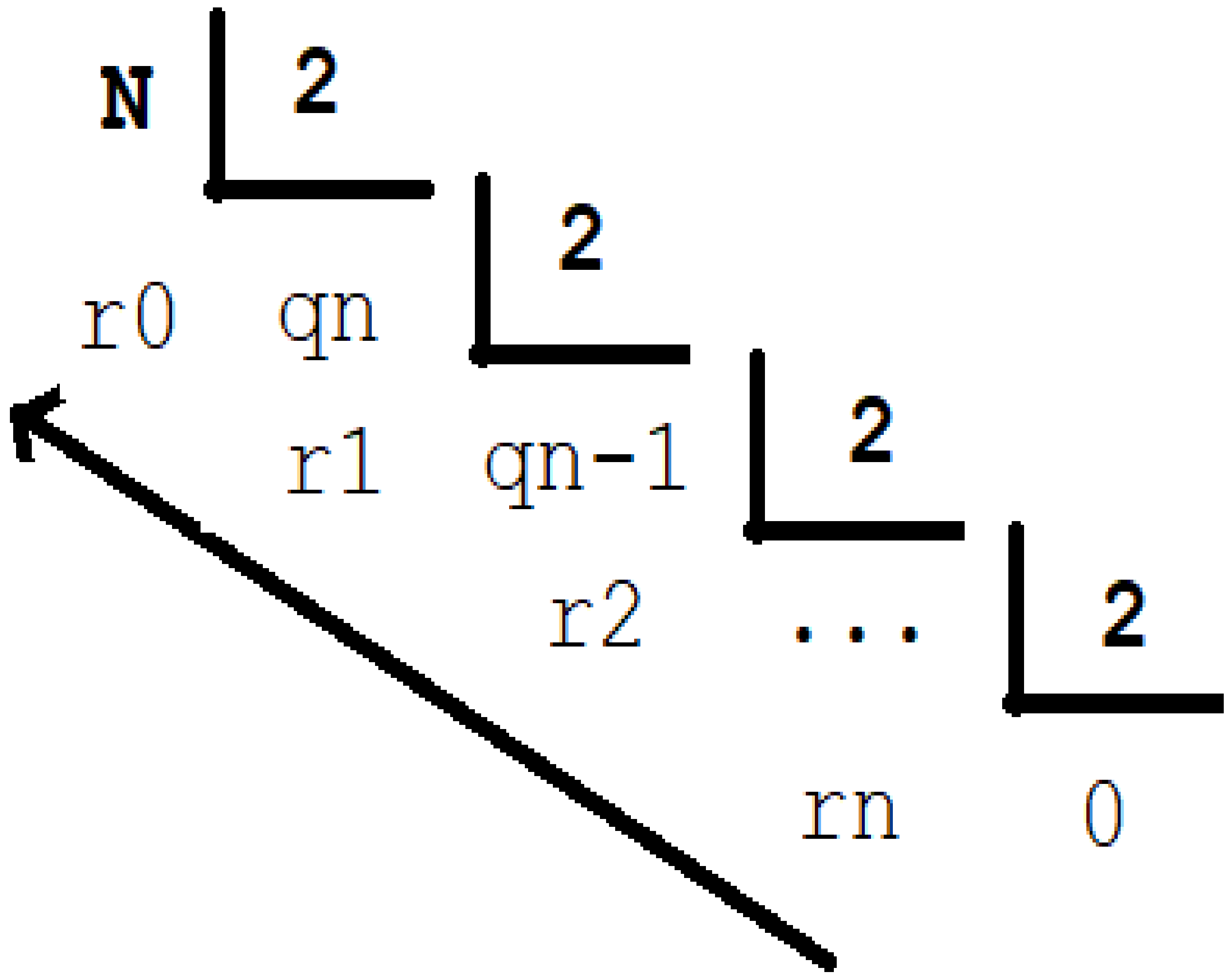
Outras bases numéricas

Base 2	Base 8	Base 10	Base 16
0	0	0	0
1	1	1	1
10	2	2	2
11	3	3	3
100	4	4	4
101	5	5	5
110	6	6	6
111	7	7	7
1000	10	8	8
1001	11	9	9
1010	12	10	A
1011	13	11	B
1100	14	12	C
1101	15	13	D
1110	16	14	E
1111	17	15	F
10000	20	16	10
10001	21	17	11

Conversão de bases numéricas

Conversão de bases

- Base <u>10 para base 2</u>: **divisões sucessivas**



$$\boxed{N = (r_n\ r_{n-1}\ r_{n-2}\ \dots r_0)_2}$$

Conversão de bases

- Base 2 para base 10: **notação polinomial**

$$b_n \times 2^n + b_{n-1} \times 2^{n-1} \ldots + b_0 \times 2^0$$

Conversão de bases

- Base <u>2 para base 16</u> e vice-versa

bn	*...*	*b6*	*b5*	*b4*	*b3*	*b2*	*b1*	*b0*	*base 2*
					dH				*base 16*

Obs. Representação das grandezas hexadecimais na forma *Nh*, *NH* ou *0xN*, por exemplo endereço *0x10A* ou *10Ah* ou *10AH*

Sistema binário

Representação binária

- O bit (*binary digit*)
- O *byte* (*B*) = 8 bits
- Múltiplos:

Kilo (k)	2^{10}
Mega (M)	2^{20}
Giga (G)	2^{30}
Tera (T)	2^{40}

Aritmética binária

- Adição binária
- Subtração binária

Aritmética binária

- Adição de dois dígitos binários: b_1+b_0

		b0		
		0	1	
b1	0	0	1	
	1	1	10	*carry ou vai um*

Aritmética binária

- Adição binária de dois números de *m* bits

1. Realizar a operação bit a bit do menos ao mais significativo (da direita para a esquerda)
2. Aplica-se a tabela anterior
3. Se houver bit 1 de *carry* transporta-se para a soma dos bits seguintes mais significativos (à esquerda)
4. Repete-se o processo até alcançar o bit mais significativo.

Aritmética binária

- Subtração de dois dígitos binários: b_1-b_0

		b0		
		0	1	
b1	0	0	11	*carry ou vai menos um*
	1	1	0	

Aritmética binária

- Subtração binária de dois números de *m* bits – Minuendo>Subtraendo

1. Operação bit a bit, do menos ao mais significativo
2. Aplica-se a tabela anterior
3. Se houver bit -1 de *carry transporta-se* para a subtração dos dígitos seguintes (à esquerda – mais significativos),
4. Repete-se o processo até alcançar o bit mais significativo.
5. Se o minuendo for menor do que o subtraendo inverter a operação e representar o número negativo

Aritmética hexadecimal

- Processo similar à aritmética binária
- Na adição, lembrar que se a soma dos dígitos exceder 16, transporta-se 1 e o valor que excedeu é o resultado da adição.

P. ex. 0x8 + 0xA = 0x12

- Na subtração, lembrar que se o minuendo for inferior ao subtraendo, transporta-se -1, soma-se 16 ao minuendo e realiza-se a subtração.

P. ex. 0x18 + 0x0A = 0x0E

Números negativos

Representação de números negativos

- Representação em bit sinal (sinal e magnitude)
- Representação em complemento de 1
- Representação em complemento de 2
- Representação em excesso 2^{m-1}

Representação em sinal e magnitude

Código para representar um número de *m* bits

1. Na posição mais significativa, utiliza-se o **bit sinal** para os números **positivos e negativos**
2. Os demais bits representam o valor absoluto da grandeza
3. Dupla representação do 0 – todos os bits iguais a 0, com bit sinal 0 ou 1

Representação em sinal e magnitude

4. Faixa de representação: $-(2^{m-1}-1)$ a $+(2^{m-1}-1)$

Código de representação

+3	0	1	1			
+2	0	1	0			
+1	0	0	1			
0	0	0	0	1	0	0
-1	1	0	1			
-2	1	1	0			
-3	1	1	1			

Representação em sinal e magnitude

- **Adição binária** e **Subtração binária**

1. Para uma adição/subtração de número com *m* bits, ...
2. aplica-se a adição/subtração da representação binária, ...
3. comparando-se os números...
4. e observando o sinal.

Complemento de números

1. O complemento de um número de *n* dígitos é a diferença entre o maior número de *n* dígitos naquela base e o número considerado.

2. Por exemplo, na base 10, o complemento de 12 é 87, pois 99 (maior número com 2 dígitos) menos 12 é igual a 87.

3. **Na base 2, para obter o complemento de 1 basta inverter os bits do número binário**

Representação em complemento de 1

Código para representar um número de *m* bits

1. Na posição mais significativa, utiliza-se o **bit sinal** para os números **positivos**
2. **Invertem-se** os bits dos números binários negativos
3. Dupla representação do 0 – todos os bits iguais a 0 ou iguais a 1

Representação em complemento de 1

4. Faixa de representação: $-(2^{m-1}-1)$ a $+(2^{m-1}-1)$
5. Exemplo:

Código de representação

+3	0	1	1			
+2	0	1	0			
+1	0	0	1			
0	0	0	0	1	0	0
-1	1	1	0			
-2	1	0	1			
-3	1	0	0			

Representação em complemento de 1

- Adição binária em complemento de 1

1. Para uma soma de número com *m* bits, ...
2. pode-se generalizar a operação do menos ao mais significativo – direita para a esquerda, ...
3. ‘carregando’ o bit 1 de *carry* para a soma dos dígitos seguintes...
4. até alcançar o bit mais significativo.
5. Se houver *carry* 1 no bit mais significativo, **transporta-se o bit 1 e o soma ao número do resultado obtido**

Representação em complemento de 1

- **Subtração binária** em complemento de 1

  Soma-se o minuendo com a **representação negativa do subtraendo** do mesmo modo anteriormente apresentado

Representação em complemento de 2

Código para representar um número de *m* bits

1. **Utiliza-se o bit sinal** para os números positivos
2. Invertem-se os bits dos números binários negativos (complemento de 1) e soma-se o resultado ao bit 1
3. O dígito mais negativo não possui inverso

Representação em complemento de 2

4. Faixa de representação: $-(2^{m-1})$ a $+(2^{m-1}-1)$
5. Exemplo

Código de representação

+3	0	1	1
+2	0	1	0
+1	0	0	1
0	0	0	0
-1	1	1	1
-2	1	1	0
-3	1	0	1
-4	1	0	0

Representação em complemento de 2

- **Adição binária** em complemento de 2

1. Para uma soma de número com *m* bits,...
2. pode-se generalizar a operação do menos ao mais significativo...
3. ‘carregando’ o bit 1 de *carry* para a soma dos dígitos seguintes...
4. até alcançar o bit mais significativo.
5. Se houver *carry* 1 no bit mais à esquerda, **despreza-se o *carry***

Representação de números negativos

- Subtração binária em complemento de 2

  Soma-se o minuendo com a representação negativa do subtraendo do mesmo modo anteriormente apresentado

Representação em excesso 2^{m-1} para números de m bits

Código para representar um número de *m* bits

1. **Não utiliza** o bit sinal
2. Soma-se 2^{m-1} a todas as grandezas a representar desde as negativas

Representação em excesso 2^{m-1} para números de m bits

3. Faixa de representação: $-(2^{m-1})$ a $+(2^{m-1}-1)$

4. Exemplo: excesso 4 de código de 3 bits

Código de representação

b2	b1	b0	
1	1	1	+3, em b10
1	1	0	+2, em b10
1	0	1	+1, em b10
1	0	0	0, em b10
0	1	1	-1, em b10
0	1	0	-2, em b10
0	0	1	-3, em b10
0	0	0	-4, em b10

Representação em excesso 2^{m-1} para números de m bits

- **Adição binária** em excesso de 2^{m-1}

1. Para uma soma de número com *m* bits,...
2. pode-se generalizar a operação do menos ao mais significativo...
3. 'carregando' o bit 1 de *carry* para a soma dos dígitos seguintes...
4. até alcançar o bit mais significativo.
5. Subtrair o resultado do excesso 2^{m-1}

Representação em excesso 2^{m-1} para números de m bits

- **Subtração binária** em excesso de 2^{m-1}

1. Soma-se o minuendo com a representação negativa do subtraendo do mesmo modo anteriormente apresentado...
2. Somar o resultado ao excesso 2^{m-1}

Exemplo: Comparação palavras de 4 bits

base 10	bit sinal				compl 1				compl 2				excesso 8			
+7	0	1	1	1	0	1	1	1	0	1	1	1	1	1	1	1
+6	0	1	1	0	0	1	1	0	0	1	1	0	1	1	1	0
+5	0	1	0	1	0	1	0	1	0	1	0	1	1	1	0	1
+4	0	1	0	0	0	1	0	0	0	1	0	0	1	1	0	0
+3	0	0	1	1	0	0	1	1	0	0	1	1	1	0	1	1
+2	0	0	1	0	0	0	1	0	0	0	1	0	1	0	1	0
+1	0	0	0	1	0	0	0	1	0	0	0	1	1	0	0	1
0	0	0	0	0	0	0	0	0	0	0	0	0	1	0	0	0
-1	1	0	0	1	1	1	1	0	1	1	1	1	1	1	0	1
-2	1	0	1	0	1	1	0	1	1	1	1	0	1	0	0	1
-3	1	0	1	1	1	1	0	0	1	1	0	1	0	1	0	1
-4	1	1	0	0	1	0	1	1	1	1	0	0	0	0	0	1
-5	1	1	0	1	1	0	1	0	1	0	1	1	1	1	0	0
-6	1	1	1	0	1	0	0	1	1	0	1	0	1	0	0	0
-7	1	1	1	1	1	0	0	0	1	0	0	1	0	1	0	0
-8	não existe				não existe				1	0	0	0	0	0	0	0

Observações

- As máquinas possuem **palavras** com tamanho definido de *m* bits
- Se a operação resultante ultrapassar a capacidade do sistema representar o número obtido...
- caracteriza-se ***overflow* = ‘estouro’**

Representação em ponto flutuante

- Notação científica: $N = f \times 10^e$

, onde *f* – fração ou mantissa; e – expoente

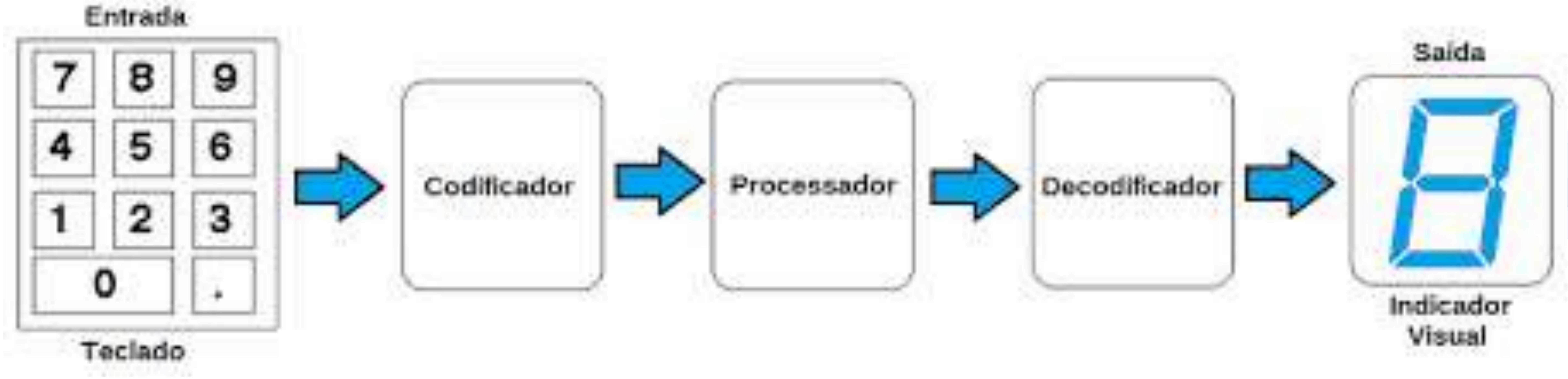
- Pela representação em ponto flutuante – equivalente computacional, quando se **convenciona o número de dígitos** para representar mantissa e expoente:

  - ❑ a faixa de representação é determinada pelo número de dígitos do expoente e

  - ❑ a precisão é determinada pelo número de dígitos da mantissa.

Representação em ponto flutuante

- A versão de ponto flutuante nos sistemas computacionais requer a representação da mantissa e do expoente no sistema binário.



Codificação binária

Códigos de detecção e correção de erros

- bit de paridade
- Distância de Hamming – número de posições de bits em que duas palavras de um código são diferentes
- Em um código com distância de Hamming igual a d+1 é possível detectar d erros de bits únicos
- Em um código com distância de Hamming igual a 2d+1 é possível corrigir d erros de bits únicos

IBMEC.BR

/IBMEC

IBMEC

@IBMEC_OFICIAL

@IBMEC

Ibmec