Introdução aos Sistemas Digitais

Introdução aos Sistemas de Numeração

Augusto Silva

Department de Electrónica, Telecomunicações e Informática
Universidade de Aveiro

augusto.silva@ua.pt

Resumo

Elementos dum sistema de numeração

- Alfabeto
  - Conjunto de símbolos
- Base
  - decimal, binária, octal, hexadecimal, outras, ...
- Regra de Representação
  - Notação posicional
- Comprimento de representação
  - Número de símbolos
  - Na prática temos um número finito de símbolos
  - Restrições à gama de representação
- Exemplos
  - 2345619_{10}
  - 00110011_2
  - 765104_8
  - $\mathtt{DACA1}_{16}$

Notação posicional

- A uma determinada quantidade X associamos uma lista ordenada de símbolos ou dígitos d_k

$$X = (d_{n-1}d_{n-2}, \ldots, d_0 d_{-1} d_{-2}, \ldots, d_{-p})_r$$

- O valor de X resulta de a cada símbolo estar associado um peso dependente da respectiva posição

$$X = \sum_{k=-p}^{n-1} d_k r^k = d_{n-1}r^{n-1} + d_{n-2}r^{n-2} + \cdots + d_0 + d_{-1}r^{-1} + \cdots d_{-p}r^{-p}$$

$d_k \in \{0, \ldots, r-1\}$, Alfabeto
$n + p$ símbolos
$r \equiv$ Base

Bases e Alfabetos

#Símbolos	Base	Alfabeto
2	2 (binária)	0,1
8	8 (octal)	0,1,2,...,6,7
10	10 (decimal	0,1,2,...,7,8,9
16	16 (hexadecimal)	0,1,2,...,8,9,A,B,C,D,E,F

- Exemplos

$$12345_{10} = 1 \times 10^4 + 2 \times 10^3 + 3 \times 10^2 + 1 \times 10^1 + 5 \times 10^0$$

$$\texttt{DACA1}_{16} = 13 \times 16^4 + 10 \times 16^3 + 12 \times 16^2 + 10 \times 16^1 + 1 \times 16^0$$

$$\begin{aligned} 1234.567_8 &= 1 \times 8^3 + 2 \times 8^2 + 3 \times 8^1 + 4 \times 8^0 \\ &+5 \times 8^{-1} + 6 \times 8^{-2} + 7 \times 8^{-3} \end{aligned}$$

As bases 8 e 16

- Relação directa como a base binária
  - Um dígito octal $\equiv$ 3 dígitos binários (bits)
  - Um dígito hexadecimal $\equiv$ 4 dígitos binários (bits) (*nibble*)

Exercício

Construa uma tabela com os primeiros 16 números inteiros representados em notação decimal, binária, octal e hexadecimal.

- Mudanças de base por substituição directa
  - $010\,111\,001\,101_2 = 2715_8$
  - $\texttt{DACA1}_{16} = 1101\,1010\,1100\,1010\,0001_2$
- Notação compacta para descrever palavras *multibyte*
  - Mais útil a notação hexadecimal (porquê?)
- Especificação compacta do espaço de endereçamento de dispositivos de memória

Outras bases

- Base 24
  - Horas do dia
- Base 60
  - Trigonometria: (g, m, s)
  - Tempo: (h, m, s)
- Bases complexas
  - $2i$ com alfabeto: $\{0, 1, 2, 3\}$
  - $i - 1$ com alfabeto: $\{0, 1\}$

Exercícios

1. Qual a quantidade decimal representada por
   a) 1101_2
   b) 37_9
   c) 0.0110_2
   d) 0.16_7
2. Qual a quantidade decimal representada por
   a) $11210.31_{(2i)}$
3. Determinar possíveis bases b e c tal que
   a) $5A_{16} = 132_b$
   b) $20_{10} = 110_c$
4. Determinar uma possível base b tal que se verifique $\sqrt{41} = 5$

Bits, Bytes e outros

- Dígito Binário $\equiv$ Binary Digit $\equiv$ Bit
- 1 Byte = 8 Bits
- 16 Bit Word = 2 Bytes
- 32 Bit Word = 4 Bytes
- 64 Bit Word = 8 Bytes

Potências de 2

2^n	n	2^{-n}
1	0	1
2	1	0.5
4	2	0.25
8	3	0.125
16	4	0.0625
32	5	0.03125
64	6	0.015625
128	7	0.0078125
256	8	0.00390625
512	9	0.001953125
1024 = 1K	10	0.0009765625

- $64\text{K} = 2^6 \times 2^{10} = 64 \times 1024 = 65536$
- $1\text{M} = 2^{20} = 2^{10} \times 2^{10} = 1024 \times 1024 = 1048576$
- $1\text{G} = 2^{30} = 2^{10} \times 2^{20} = 1024 \times 1048576 = 1073741824$
- $1\text{T} = 2^{40} = 2^{20} \times 2^{20} = ?$

Mudanças de base

- Motivações:
  - Interface entre o mundo real decimal e as instâncias de processamento binário
- Formulação do problema:
  - Passar duma representação em n dígitos e base r para uma representação em m dígitos em base s
- Condições:
  - Em geral uma quantidade resulta da justaposição duma componente inteira com uma componente fraccionária

$$0 < \mathrm{FRAC}(X) < 1,\ 0_r = 0_s,\ 1_r = 1_s$$

$$\mathrm{FRAC}(X_r) \mapsto \mathrm{FRAC}(X_s)$$

$$\mathrm{INT}(X_r) \mapsto \mathrm{INT}(X_s)$$

  - No caso de quantidades não inteiras deve adoptar-se um critério que determine o número de dígitos significativos consistente com os erros de representação nas bases inicial e final

Erros de representação

- Seja n o número de dígitos fraccionários na base r
- Seja m o número de dígitos fraccionários na base s
- Interessa determinar m de tal forma que os erros de representação sejam o mais próximos possível:

$$|E_r| \leq \left|\frac{r}{2} r^{-(n+1)}\right| \approx |E_s| \leq \left|\frac{s}{2} s^{-(m+1)}\right|$$

$$s^{-m} \approx r^{-n} \therefore m \approx n \frac{\log r}{\log s}$$

- Se considerarmos que a mudança de base não deve trazer acréscimo de precisão então

$$m = \left\lfloor n \frac{\log r}{\log s} \right\rfloor$$

Nota: $\lfloor . \rfloor$ é o operador *floor* que trunca a quantidade para o número inteiro imediatamente inferior.

Exemplos

1. Qual a representação decimal correcta para
   a) 12.345_8

$$12.345_8 = 1 \times 8 + 2 + \frac{3}{8} + \frac{4}{64} + \frac{5}{512}$$

$$\mathtt{m} = \left\lfloor 3\frac{\log 8}{\log 10} \right\rfloor = \lfloor 2.7 \rfloor = 2$$

$$12.345_8 = 10.45_{10}$$

   b) $\mathtt{DA.CA1}_{16}$

$$\mathtt{DA.CA1}_{16} = 13 \times 16 + 10 + \frac{12}{16} + \frac{10}{256} + \frac{1}{4096}$$

$$\mathtt{m} = \left\lfloor 3\frac{\log 16}{\log 10} \right\rfloor = \lfloor 3.6 \rfloor = 3$$

$$\mathtt{DA.CA1}_{16} = 218.789_{10}$$

Conversão da parte inteira

- Como o sistema aritmético humano assenta na base 10 a mudança da base r para a base s processa-se segundo um algoritmo baseado em divisões sucessivas (DIVS)
- Processo genérico:

$$N = (a_{n-1} \dots a_0)_r = \overbrace{(d_{k-1} \dots d_0)_{10}}^{\text{Conversão Intermédia}} = (b_{m-1} \dots b_0)_s$$

$$N_{10} = (d_{k-1} \dots d_0)_{10} = \sum_{i=0}^{n-1} a_i r^i$$

$$N_{10} = (b_{m-1} \dots b_0)_s = \text{DIVS}\left\lfloor \frac{N_{10}}{s} \right\rfloor$$

Divisão sucessiva

- Fundamentação:

$$
\begin{aligned}
N_{10} &= b_{m-1}s^{m-1} + b_{m-2}s^{m-2} + \cdots + b_1 s + b_0 \\
&= \left(b_{m-1}s^{m-2} + b_{m-2}s^{m-3} + \cdots + b_1\right) s + b_0 \\
&= sQ_1 + b_0 \\
Q_1 &= \left(b_{m-1}s^{m-3} + b_{m-2}s^{m-4} + \cdots + b_2\right) s + b_1 \\
&= sQ_2 + b_1 \\
Q_2 &= \left(b_{m-1}s^{m-4} + b_{m-2}s^{m-5} + \cdots + b_3\right) s + b_2 \\
&= sQ_3 + b_2 \\
\vdots &= \vdots \\
Q_{m-1} &= sb_{m-1}
\end{aligned}
$$

Exemplos

432	2							
400	216	2						
32	200	108	2					
20	16	100	54	2				
12	16	8	40	27	2			
12	0	8	14	20	13	2		
0		0	14	7	12	6	2	
			0	6	1	6	3	2
				1		0	2	1
							1	

$432 = 110110000_2$

1462	8		
800	182	8	
662	160	22	8
640	22	16	2
22	16	6	
16	6		
6			

$1462 = 2666_8$

Exemplos

$$
\begin{array}{rrrr}
432 & 5 & & \\
400 & 86 & 5 & \\
32 & 50 & 17 & 5 \\
30 & 36 & 15 & 3 \\
2 & 35 & 2 & \\
 & 1 & &
\end{array}
$$

$432 = 3212_5$

$$
\begin{array}{rrr}
1462 & 16 & \\
1440 & 91 & 16 \\
22 & 80 & 5 \\
16 & 11 & \\
6 & \downarrow & \\
 & \mathrm{B} &
\end{array}
$$

$1462 = 5B6_{16}$

Exercícios

1. Converta para base 5
   a) 10110_2
   b) 64_7
2. Converta para base 2
   a) 53_6
   b) 43_7
   c) 129_{10}

Conversão da parte fraccionária

- Fundamentação (Multiplicação Sucessiva)

$$
\begin{aligned}
F_{10} &= b_{-1}s^{-1} + b_{-2}s^{-2} + \cdots + b_p s^{-p} \\
s.F_{10} &= b_{-1} + \left(b_{-2}s^{-1} + \cdots + b_p s^{-p+1}\right) \\
&= b_{-1} + P_1 \\
s.P_1 &= b_{-2} + \left(b_{-3}s^{-1} + \cdots + b_p s^{-p+2}\right) \\
&= b_{-2} + P_2 \\
\vdots &= \vdots
\end{aligned}
$$

Exemplo

- $0.6875_{10} = (b_{-1} \dots b_{-p})_2$
- Quantos dígitos binários são adequados?

$$m = \left\lfloor 4\frac{\log 10}{\log 2} \right\rfloor = 13$$

$$\begin{array}{r} 0.6875 \\ \times \quad 2 \\ \hline 1.3750 \end{array} \qquad \begin{array}{r} 0.375 \\ \times \quad 2 \\ \hline 0.750 \end{array}$$

$$\begin{array}{r} 0.75 \\ \times \quad 2 \\ \hline 1.50 \end{array} \qquad \begin{array}{r} 0.5 \\ \times \quad 2 \\ \hline 1.0 \end{array}$$

$$\vdots$$

$0.6875_{10} = 0.1011000000000_2$

Adição e Subtracção

- Adição binária

+	0	1
0	0	1
1	1	0

Transporte

	1	1	1	0	1
+	1	1	0	0	1
1	1	0	1	1	0

- Subtracção binária

-	0	1
0	0	1
1	1	0

Transporte

	1	1	0	0	1
-			1	0	1
	1	0	1	0	0

Multiplicação

- Multiplicação binária

X	0	1
0	0	0
1	0	1

			1	0	1	1	1
			X	1	0	1	0
			0	0	0	0	0
		1	0	1	1	1	
	0	0	0	0	0		
1	0	1	1	1			
1	1	1	0	0	1	1	0

Multiplicação

- Multiplicação binária: *Shift and add*

$$\begin{array}{r} 23 \\ \times\ 13 \\ \hline 69 \\ 23 \\ \hline 299 \end{array}$$

$$\begin{array}{r} 10111 \\ \times\ \ 1101 \\ \hline 00000 \\ 10111 \\ \hline 010111 \\ 00000- \\ \hline 0010111 \\ 10111-- \\ \hline 01110011 \\ 10111--- \\ \hline 100101011 \end{array}$$

Exercícios

Aplique a técnica *shift and add* para obter em binário

a) 15×15

b) 25×4

c) 25×8

d) O que há de especial nos dois últimos casos?

Divisão

- Divisão binária: Brevemente ...

Quantidades Negativas

- Sistema de notação posicional por si só não conduz naturalmente à representação de quantidades negativas
- Metodologias mais comuns:
  - Sinal e Módulo
    - Consistência com a notação posicional
    - Necessário um símbolo adicional para representar o sinal
    - Sistema natural para os humanos mas incompatível com a fundamentação modular da aritmética dos sistemas digitais.
  - Códigos de Complemento
    - Um subconjunto da gama de representação absoluta codifica as quantidades negativas
    - Natural em sistemas computacionais digitais
    - Não necessita símbolo adicional para representação do sinal
    - Permite processamento integrado da soma algébrica

Sinal + Módulo

- Soma algébrica $A \pm B$

```
if sinal(A) = sinal(B) then
    begin
        |soma| = |A|+|B|
        sinal(soma) = sinal(A)
    end
else
    if |A|>|B| then
        begin
            |soma| = |A|-|B|
            sinal(soma) = sinal(A)
        end
    else
        begin
            |soma| = |B|-|A|
            sinal(soma) = sinal(B)
        end
```

- Muito difícil para o hardware

- Convenção: na posição mais à esquerda
  0 sinal "+"
  1 sinal "−"
- Ambiguidade para o zero

$$+0 = 0\,0000 \qquad -0 = 1\,0000$$

- $+6 = 00110$ (4 bits para módulo)
- $-6 = 10110$ (4 bits para módulo)

Exercício

Determine $4 - 7$ em binário usando notação sinal e módulo

Complementos para a base

- Com n dígitos e base r define-se o complemento para a base r duma quantidade X, como
$$\mathrm{RC}(X) = r^n - X$$
- $\mathrm{RC} \equiv$ "Radix Complement"
- A ideia é substituir o cálculo duma quantidade $M - X$ pela soma $M + \mathrm{RC}(X) = M + r^n - X$
- Este procedimento produz resultados válidos desde que

$$-\frac{r^n}{2} \leq M - X \leq \frac{r^n}{2} - 1$$

$$\therefore$$

$$M - X = (M - X)\mathrm{mod} r^n$$

Nota: a operação $\mathrm{mod} r^n$ deve ser entendida como o resto da divisão inteira por r^n

Exemplos

- No contexto decimal:
- Com 3 dígitos tem-se necessariamente $-500 \leq N \leq 499$

$$300 - 100 = (300 - 100)\mathrm{mod}1000 = (300 + (1000 - 100)\mathrm{mod}1000) = 200$$

- Repita com 5 dígitos e calcule 3000 - 2500

Complemento para a base 2

- Caso particular, relevante nos sistemas computacionais binários
- Determinação do complemento para 2 duma quantidade representada por n dígitos

$$-X = (2^n - X)$$

- Por questões de facilidade de implementação podemos recorrer ao complemento para a base diminuído ou complemento para 1

$$-X = (2^n - X - 1) + 1$$

- Importa reconhecer que $(2^n - X - 1) = (X'_{n-1} \dots X'_0)$, donde

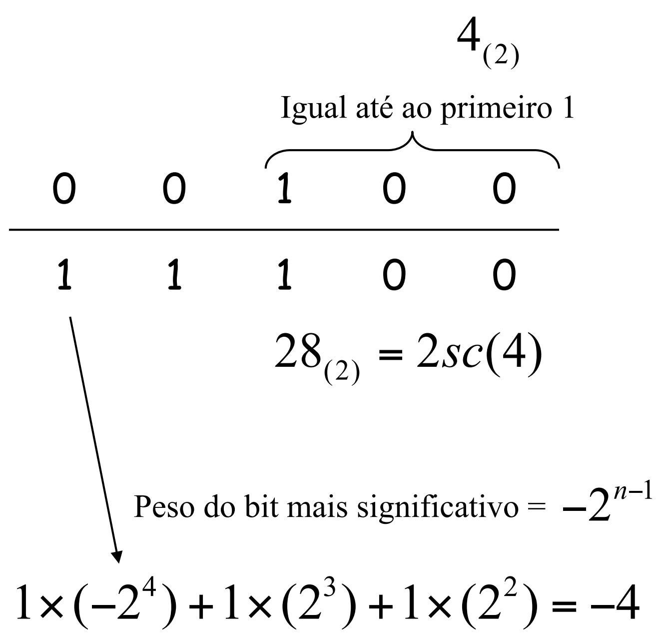
$$-X = (X'_{n-1} \dots X'_0) + 1$$

- Exemplo com 5 bits

$$X = (01101), \; -X = (10010) + 1 = 10011$$

Algoritmo sequencial

$i = 0$
while $b_i = 0$
  begin
    $2sc(b_i) = b_i$
    $i = i+1$
  end
for $k := i+1$ **to** n **do**
  begin
    $2sc(b_i) = \overline{b_i}$
  end



Nota: 2sc $\equiv$ two's complement

Overflow

- Overflow acontece quando o resultado duma operação algébrica excede a gama de representação
- Resultados são de sinal contrário aos dos operandos
- Exemplos com 4 bits

	-3	1101	
+	-6	+1010	
	-9	10111	= +7

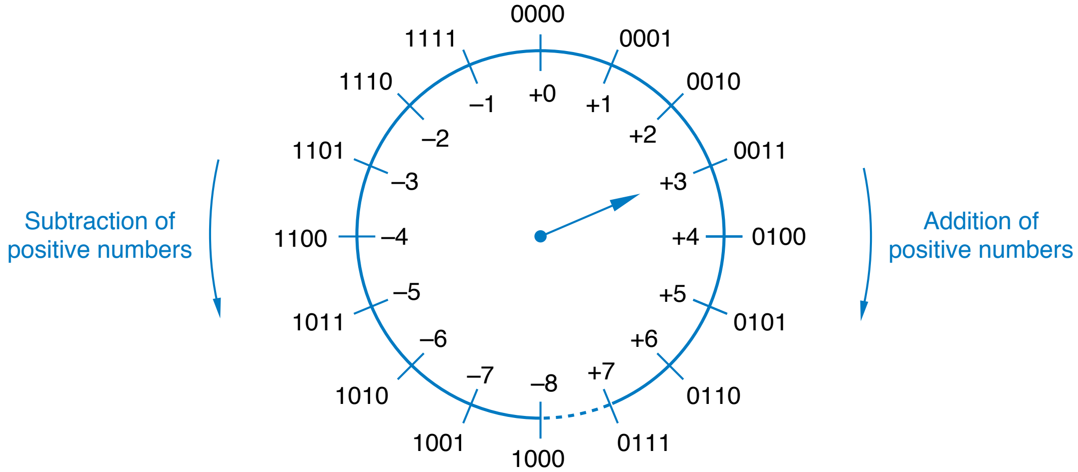
	5	0101	
+	6	+0110	
	+11	1011	= -5

	-8	1000	
+	-8	+1000	
	-16	10000	= 0

	7	0111	
+	7	+0111	
	+14	1110	= -2

Detecção de overflow

- Verifique, nestes casos o que acontece com o valor do último e penúltimo *carry-out* da adição algébrica.
- Uma perspectiva gráfica



Adaptado de J. Wakerly, "Digital Design Principles & Practices". 3rd ed.

Exercícios

Considere um comprimento de representação de 8 bits. Caso seja possível, determine em notação de complemento para 2:

a) $45 + 20$

b) $45 - 20$

c) $20 - 45$

d) $-20 - 45$

e) $90 + 50$

f) $90 - 50$

g) $-50 - 90$

Bibliografia

- J. Wakerly, "Digital Design Principles & Practices", Cap 2.