

Representação de informação

Formatos e operações

João Canas Ferreira

Setembro de 2018



Tópicos

- ① Introdução à representação de informação
- ② Representação em binário de números sem sinal
 - Representação binária
 - Conversão de base
 - O sistema de numeração hexadecimal
 - Operações aritméticas com operandos em binário
- ③ Representação de números inteiros com sinal
- ④ Representação de caracteres e símbolos gráficos
- ⑤ Representação de números em vírgula flutuante
 - Conceitos gerais
 - Formato IEEE-754 precisão simples
 - Operações aritméticas com operandos normalizados

1 Introdução à representação de informação

2 Representação em binário de números sem sinal

Representação binária

Conversão de base

O sistema de numeração hexadecimal

Operações aritméticas com operandos em binário

3 Representação de números inteiros com sinal

4 Representação de caracteres e símbolos gráficos

5 Representação de números em vírgula flutuante

Conceitos gerais

Formato IEEE-754 precisão simples

Operações aritméticas com operandos normalizados

Representação física da informação

- O computador automático pode ser definido como uma máquina de uso geral que processa informação de acordo com uma sequência de instruções.

Questão: Como representar fisicamente a informação?

- As tecnologias atuais de implementação recorrem a elementos básicos que podem estar em um de dois estados (elementos bi-estáveis).

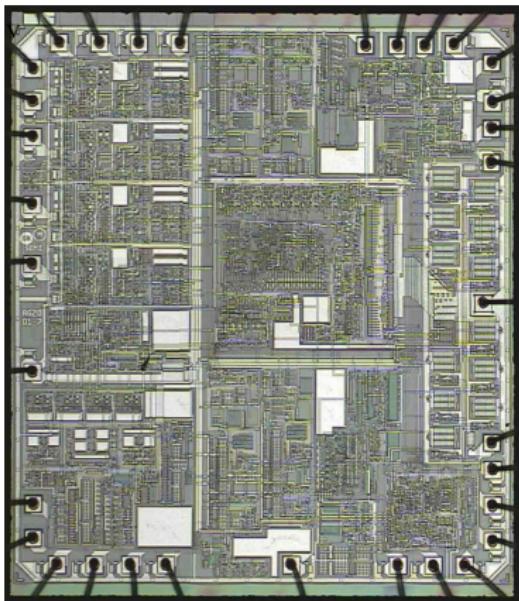
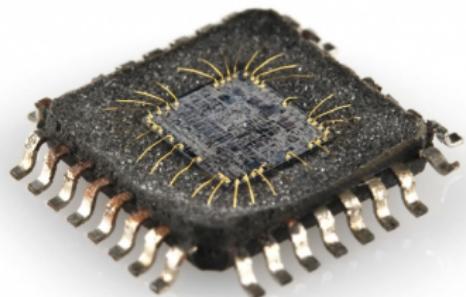
Os elementos ativos dos processadores atuais são dispositivos eletrónicos (transístores) que atuam como interruptores que podem estar **ligados** ou **desligados**.

Os dispositivos são feitos de silício, um material semicondutor, e vários compostos de silício. O óxido de silício é um dos materiais mais comuns (constituinte principal da areia).

- Os microprocessadores atuais são **circuitos integrados** que contém na ordem dos milhares de milhões (10^9) de elementos ativos, que são capazes de mudar de estado em **algumas dezenas de picosegundos** ($1 \text{ ps} = 1 \times 10^{-12} \text{ s}$).

Circuitos integrados

■ Circuitos miniaturizados em que os componentes estão todos integrados no mesmo substrato (de silício). Os circuitos **digitais** contém transistores (a funcionar como interruptores) e interligações de metal (alumínio ou cobre).



Fonte: [//learn.sparkfun.com/tutorials/integrated-circuits](http://learn.sparkfun.com/tutorials/integrated-circuits)

Representação simbólica da informação

- Para obter uma **representação abstrata**, desligada dos detalhes da realização física, os estados são representados pelos símbolos **0 e 1**.
- A unidade fundamental de informação designa-se por **bit** (símbolo bit) e constitui a quantidade mínima de informação realizável fisicamente (pode assumir apenas dois valores diferentes, 0 ou 1).
- Para representar maior quantidade de informação pode-se usar um conjunto (ordenado) de bits. Um conjunto de N bits representa-se

$$a_{N-1}a_{N-2}\dots a_1a_0$$

- Um conjunto de 8 bits designa-se por **octeto**.
- O conjunto de bits mais pequeno tratado por um processador designa-se por **byte**, símbolo B.
- Para os computadores atuais, $1\text{ B} = 8\text{ bit}$ (um byte é o mesmo que um octeto).

1 Introdução à representação de informação

2 Representação em binário de números sem sinal

Representação binária

Conversão de base

O sistema de numeração hexadecimal

Operações aritméticas com operandos em binário

3 Representação de números inteiros com sinal

4 Representação de caracteres e símbolos gráficos

5 Representação de números em vírgula flutuante

Conceitos gerais

Formato IEEE-754 precisão simples

Operações aritméticas com operandos normalizados

Sistemas de numeração posicionais

- Um sistema numérico posicional de representação é determinado por:
 - ☞ **base b :** um número inteiro superior a 1;
 - ☞ **b dígitos (símbolos individuais):** $0, 1, \dots, b-1$.
- Exemplo: O sistema decimal tem base $b=10$ e dígitos $0, 1, 2, \dots, 9$.
- Uma sequência de N dígitos (número) $d_{N-1}d_{N-2}\dots d_1d_0$ representa o valor dado pela expressão seguinte em que cada dígito aparece a multiplicar a potência da base correspondente. (Para qualquer base, $b^0 = 1$)

$$d_{N-1} \times b^{N-1} + d_{N-2} \times b^{N-2} + \dots + d_1 \times b^1 + d_0 \times b^0$$

- Por exemplo, para $b = 10$, o número **4542** representa o valor

$$4 \times 10^3 + 5 \times 10^2 + 4 \times 10^1 + 2$$

Por palavras: 4 milhares, 5 centenas, 4 dezenas e 2 unidades.

- **O peso de cada dígito depende da sua posição:** no número acima, as duas instâncias do símbolo '4' têm pesos diferentes (10^3 e 10 , respetivamente).

Sistema de numeração binário

- Num computador digital (binário) existem apenas dois símbolos que podem servir de dígitos (0 e 1). Logo, temos que usar base $b = 2$.
- Um conjunto de N bits pode ser interpretado como um número em base 2 em que $d_i \in \{0, 1\}$ (i.e., uma soma de potências de 2):

$$d_{N-1} \times 2^{N-1} + d_{N-2} \times 2^{N-2} + \dots + d_1 \times 2^1 + d_0 \times 2^0$$

- Exemplo: $10010_2 = 1 \times 2^4 + 0 \times 2^3 + 0 \times 2^2 + 1 \times 2^1 + 0 \times 2^0 = 16 + 2 = 18_{10}$
- Consequência: todos os números pares têm uma representação binária terminada por 0. [Porquê?]
- Consequência: as regras para as operações aritméticas em binário são análogas às regras usadas com o sistema decimal, porque dependem apenas das propriedades dos sistemas de numeração posicionais (com $b \in \mathbb{N}, b > 1$).
- A conversão binário → decimal pode ser feita aplicando diretamente a definição e realizando os cálculos em decimal (como no exemplo acima).

Parte fracionária

- O sistema posicional pode também representar facilmente números com parte fracionária usando potência inteiros **negativas** da base.
- A vírgula (ou o ponto, em inglês) é usado para separar as duas partes (inteira e fracionária). O número $d_{N-1}d_{N-2}\dots d_1d_0.d_{-1}d_{-2}\dots$

$$d_{N-1} \times b^{N-1} + \dots + d_1 \times b^1 + d_0 \times b^0 + d_{-1} \times b^{-1} + d_{-2} \times b^{-2} + \dots$$

- Exemplo em decimal: $17,83_{10} = 1 \times 10^1 + 7 + 8 \times \frac{1}{10} + 3 \times \frac{1}{10^2}$

Em palavras: 1 dezena, 7 unidades, 8 décimas e 3 centésimas.

- Exemplo em binário:

$$101,101_2 = 1 \times 2^2 + 1 + 1 \times \frac{1}{2} + 1 \times \frac{1}{2^3} = 4 + 1 + 0,5 + 0,125 = 5,625_{10}$$

- Em qualquer sistema posicional, pode-se considerar que existe um número infinito de zeros à esquerda e à direita da parte representada.

Exemplos: $0076,1400_{10} = 76,14_{10}$ $00101,1100_2 = 101,11_2$.

Conversão binário → decimal

- Um número em binário corresponde a uma soma de potências de 2.
- Para converter um número binário em decimal basta executar as operações implícitas na representação posicional usando base 10.

Exemplo de conversão base 2 → base 10

$$1101,011_2 = 2^3 + 2^2 + 2^0 + 2^{-2} + 2^{-3} = 8 + 4 + 1 + 0,25 + 0,125 = 13,375_{10}$$

- A posição imediatamente à *esquerda* da vírgula corresponde ao expoente 0.
- A posição imediatamente à *direita* da vírgula corresponde ao expoente -1.
- Expoentes crescem para a esquerda da vírgula, decrescem para a direita.

Conversão decimal → binário: parte inteira

- Dividir o número sucessivamente por 2 até chegar a 0.
- Guardar o resto (0 ou 1) de cada divisão.
- Os valores sucessivos do resto constituem o número em binário, sendo que o primeiro resto corresponde ao bit *menos significativo* e o último ao *mais significativo*.

Exemplo de conversão base 10 → base 2

$$19_{10} = ?_2$$

N=19	resto da divisão
9	1
4	1
2	0
1	0
0	1

→

$$19_{10} = 10011_2$$

Conversão decimal → binário: parte fracionária

- Multiplicar o número sucessivamente por 2.
- Guardar a parte inteira (0 ou 1) de cada multiplicação.
- Os valores da parte inteira constituem a parte fracionária do número em binário. O primeiro valor é o bit *mais significativo* e o último ao *menos significativo*.
- Terminar: número chega a 0 *ou* se repete *ou* atinge um número dado de bits.

Exemplo de conversão base 10 → base 2: 0,6₁₀

Número	×2	Parte inteira	
0,6	1,2	1	$0,6_{10} = 0,1001(1001)_2$ (parte fracionária infinita com repetição)
0,2	0,4	0	
0,4	0,8	0	→
0,8	1,60	1	com 8 casas binárias:
0,6	$0,10011001_2 = 2^{-1} + 2^{-4} + 2^{-5} + 2^{-8} = 0,59765625_{10}$

Sistema de base 16

- O sistema *hexadecimal* use base $b = 16$.
- O conjunto de dígitos é $0, 1, 2, 3, 4, 5, 6, 7, 8, 9, A, B, C, D, E, F$.
- Para os dígitos acima de 9, a correspondência com valores decimais é:

$$A \leftrightarrow 10 \quad B \leftrightarrow 11 \quad C \leftrightarrow 12 \quad D \leftrightarrow 13 \quad E \leftrightarrow 14 \quad F \leftrightarrow 15.$$

Conversão de hexadecimal para decimal

$$6B7_{16} = 6B7_H = 6 \times 16^2 + 11 \times 16^1 + 7 = 1719_{10}$$

$$7F,5_{16} = 7 \times 16^1 + 15 \times 16^0 + 5 \times 16^{-1} = 127,3125_{10}$$

- A conversão de decimal para hexadecimal usa o mesmo procedimento que a conversão para binário, mas dividindo ou multiplicando por 16 (em vez de 2).
- O sistema hexadecimal permite uma representação mais compacta dos números (com menos dígitos).

Relação entre sistema binário e sistema hexadecimal

► Existe uma relação direta entre as representações hexadecimal e binária!

Número binário com 8 bits: $d_7d_6d_5d_4d_3d_2d_1d_0$ com $d_i \in \{0, 1\}$ representa o valor

$$d_7 \cdot 2^7 + d_6 \cdot 2^6 + d_5 \cdot 2^5 + d_4 \cdot 2^4 + d_3 \cdot 2^3 + d_2 \cdot 2^2 + d_1 \cdot 2^1 + d_0 \cdot 2^0.$$

A expressão pode ser re-escrita como:

$$2^4 \cdot (d_7 \cdot 2^3 + d_6 \cdot 2^2 + d_5 \cdot 2^1 + d_4 \cdot 2^0) + 2^0 \cdot (d_3 \cdot 2^3 + d_2 \cdot 2^2 + d_1 \cdot 2^1 + d_0 \cdot 2^0).$$

► O valor das expressões a azul está entre 0 e 15. [Porquê?]

Portanto, a expressão pode ser escrita como

$$16^1 \cdot h_1 + 16^0 \cdot h_0 \quad \text{em que } h_i \in \{0, 1, 2, 3, 4, 5, 6, 7, 8, 9, A, B, C, D, E\}$$

que é representada em hexadecimal como h_1h_0 .

► Um número em binário pode ser convertido para hexadecimal *formando grupos de 4 dígitos binários e substituindo-os pelo correspondente dígito hexadecimal.*

Exemplos de conversão entre representações em base 2 e 16

De binário para hexadecimal

$$\boxed{1} \boxed{10} \boxed{0101}_2 = D5_H$$

$$1011101001_2 = \boxed{0010} \boxed{1110} \boxed{1001}_2 = 2E9_H$$

$$111010,0101_2 = \boxed{0011} \boxed{1010} \boxed{0,0101}_2 = 3A,5_H$$

O processo também funciona no sentido inverso.

De hexadecimal para binário

$$FF_H = \boxed{1111} \boxed{1111}_2 = 11111111_2$$

$$1AC_H = \boxed{0001} \boxed{1010} \boxed{1100}_2 = 110101100_2$$

$$5F,2A_H = \boxed{0101} \boxed{1111} \boxed{,0010} \boxed{1010}_2 = 1011111,00101010_2$$

Adição de números sem sinal

■ Números sem sinal: $\{0, 1, 2, 3, \dots\}$

■ Regras para as operações aritméticas são *análogas* às de aritmética decimal.

■ Adição: $0 + 0 = 0$ $1 + 0 = 0 + 1 = 1$

■ Adição: $1 + 1 = 10$

resultado igual a 0 e transporte de 1 unidade para a posição seguinte

■ Adição: $1 + 1 + 1 = 11$

resultado igual a 1 e transporte de 1 unidade para a posição seguinte

Adição em binário

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

$$\begin{array}{r} 1 \\ + 2 & 1 & 2 \\ \hline 3 & 0 & 5 \end{array}$$

Multiplicação de números sem sinal

► Regras: $0 \times 0 = 1 \times 0 = 0 \times 1 = 0$ $1 \times 1 = 1$

Multiplicação em binário

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

$$\begin{array}{r} & 2 & 0 \\ \times & 1 & 3 \\ \hline & 6 & 0 \\ \times & 2 & 0 \\ \hline 2 & 6 & 0 \end{array}$$

► Pode-se omitir as filas que têm apenas zeros, mas é preciso cuidado ao alinhar os produtos parciais.

Subtração de números sem sinal

■ Subtrair apenas números menores a números maiores (diminuendo maior que o subtraendo).

■ Subtração: $1 - 1 = 0 - 0 = 0 \quad 1 - 0 = 1$

■ Subtração: $10 - 1 = 01$

resultado=1 e trocar o bit da posição seguinte (do diminuendo)

■ Subtração: $100 - 1 = 011$

resultado=1 e trocar os bits das duas posições seguintes (do diminuendo)

■ As alterações do diminuendo (em binário) seguem a regra:

da direita para a esquerda a partir da posição da subtração trocar zeros por uns até ao primeiro 1; trocar também este dígito.

Subtração em binário

$$\begin{array}{r} 0 \ 0 \ 1 \ 0 \ 1 \\ 1 \ 1 \ 0 \ 1 \ 0 \ 0 \ 1 \ 1 \\ - 0 \ 1 \ 0 \ 1 \ 1 \ 1 \ 1 \ 0 \\ \hline 0 \ 1 \ 1 \ 1 \ 0 \ 1 \ 0 \ 1 \end{array}$$

$$\begin{array}{r} 1 \ 0 \\ 2 \ 1 \ 1 \\ - 9 \ 4 \\ \hline 1 \ 1 \ 7 \end{array}$$

1 Introdução à representação de informação

2 Representação em binário de números sem sinal

Representação binária

Conversão de base

O sistema de numeração hexadecimal

Operações aritméticas com operandos em binário

3 Representação de números inteiros com sinal

4 Representação de caracteres e símbolos gráficos

5 Representação de números em vírgula flutuante

Conceitos gerais

Formato IEEE-754 precisão simples

Operações aritméticas com operandos normalizados

Representações com número fixo de bits

■ Processadores usam um número fixo de bits para representar números.

Por exemplo, um processador de 32 bits tem esta designação por representar os números com 32 bit.

■ Muitos processadores suportam representações de vários tamanhos.

Ex.: microprocessadores Intel atuais suportam operandos de 8, 16, 32 ou 64 bit.

■ Apenas operações com operandos do mesmo tamanho (i.e., igual número de bits) são suportadas.

■ O uso de um tamanho fixo implica que *nem todos os resultados são representáveis*.

■ A gama de números sem sinal (binário puro) que pode ser representada com N bits é $[0; 2^N - 1]$. (Porquê?)

Saída da gama de representação: erro de *overflow*

Com N = 8, a gama da representação binária é [0; 255].

O resultado da adição $230_{10} + 72_{10}$ não pode ser representado, embora os operandos possam.

Números sem sinal

→ A gama de números sem sinal (binário puro) que pode ser representada com N bits é $[0; 2^N - 1]$. (Porquê?)

Saída da gama de representação: erro de *overflow*

Com N = 8, a gama da representação binária é [0; 255].

O resultado da adição $230_{10} + 72_{10}$ não pode ser representado, embora os operandos possam.

Adição em binário com *overflow*

Considerar N=8.

$$\begin{array}{r} 1 & 1 & 1 & 1 & 1 \\ 0 & 1 & 0 & 1 & 1 & 1 & 0 & 1 \\ + & 1 & 1 & 0 & 1 & 0 & 1 & 0 & 0 \\ \hline \text{X} & 0 & 0 & 1 & 1 & 0 & 0 & 0 & 1 \end{array} \qquad \begin{array}{r} & & 9 & 3 \\ & + & 2 & 1 & 2 \\ \hline & & 4 & 9 \end{array}$$

Erro!
Overflow
305 seria correto, mas tem-se
 $305 - 256 = 49$

Representação em sinal e grandeza

- A representação em sinal e grandeza com N bits usa o bit mais significativo para indicar o sinal e os N-1 bits para representar o valor absoluto (a grandeza).
- Bit de sinal: 0 positivo, 1 negativo.

Representação em sinal e grandeza

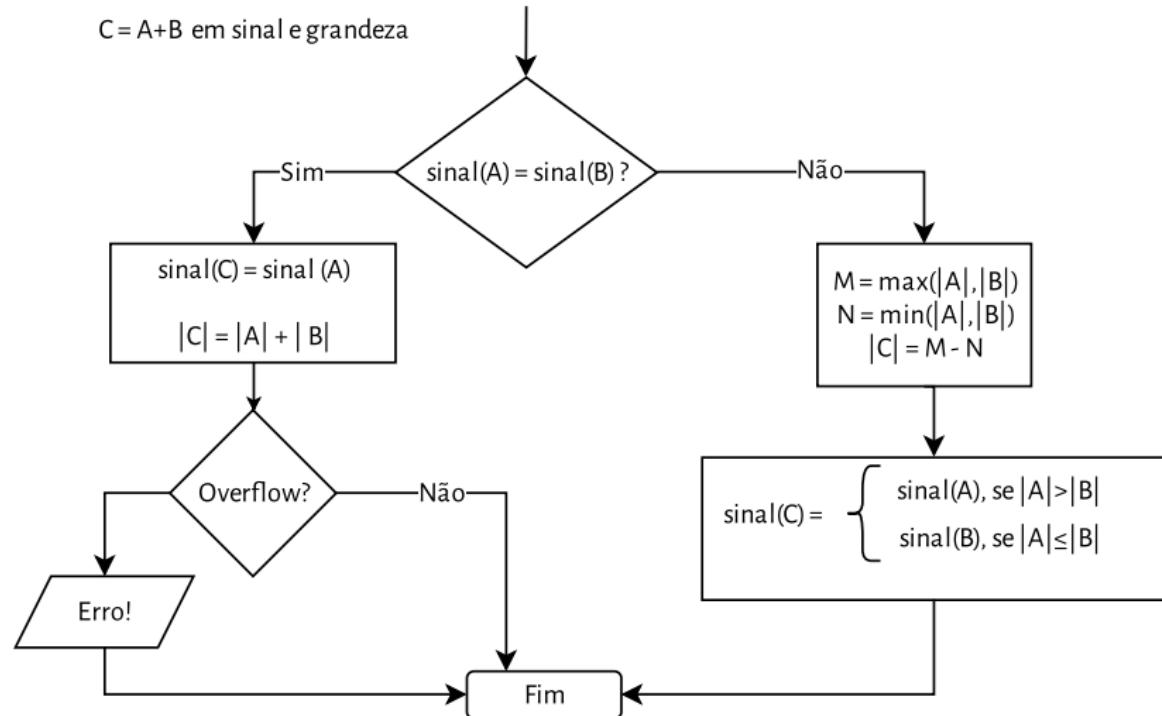
Considerar N=6.

$$17_{10} = 010001_{SG} \quad -17_{10} = 110001_{SG}$$

A gama de valores representáveis é [−31; +31].

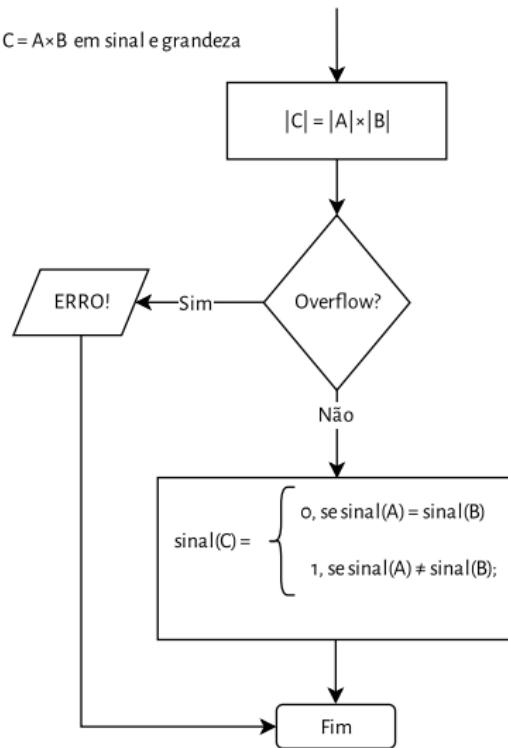
- A gama de valores representáveis em N bits é $[-(2^{N-1} - 1); (2^{N-1} - 1)]$.
(Porquê?)
- Existem duas representações para 0. (Quais?)
- É uma representação muito pouco usada para representar números inteiros.

Regras para operações aritméticas: Adição e subtração



Subtração: $A - B = A + (-B)$ e usar as regras da adição

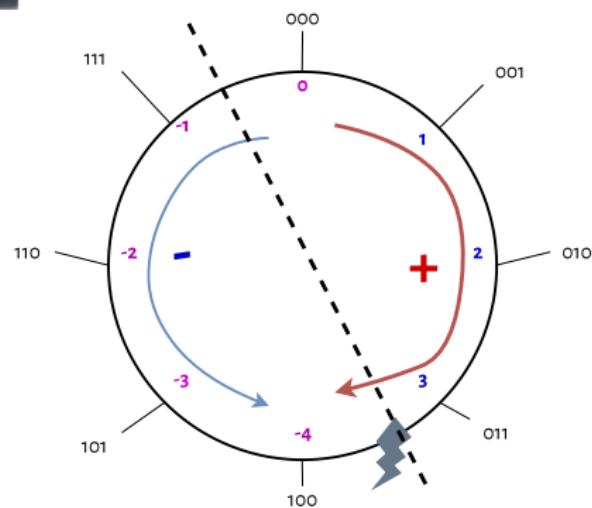
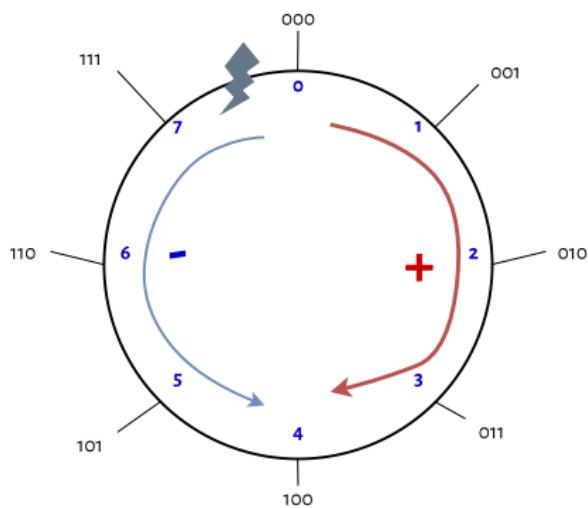
Regras para operações aritméticas: Multiplicação



Representação em complemento para 2: conceito

- Objetivo: encontrar uma representação com regras mais simples.
- O que acontece a um contador binário quando atinge o limite?

Volta a zero e continua a contar.



- Intervalo de valores da representação em complemento para 2: [-4; 3]

Representação em complemento para 2: definição

- Representação em *complemento para 2* com **N** bits: $[-2^{(N-1)}; 2^{(N-1)} - 1]$.
- Números não-negativos são representados como habitualmente \rightarrow *bit mais significativo é zero*.
- Números negativos são representados de forma a satisfazer a fórmula:

$$-d_{N-1} \times 2^{N-1} + d_{N-2} \times 2^{N-2} + \dots + d_1 \times 2^1 + d_0 \times 2^0$$

\rightarrow *bit mais significativo é um.* [Porquê?]

Conversão de números com sinal (CPL2)

Usando $N=5$, o intervalo de valores representáveis é $[-16;15]$.

$$13_{10} = 01101_{\text{cpl2}}$$

$$-13_{10} = -16 + 3 = 10011_{\text{cpl2}}$$

Representação em complemento para 2: operações

- Adição e subtração: procedimento é igual ao usado com números sem sinal, mas a deteção de *overflow* é diferente.
- Alternativa para subtração: $A - B = A + (-B)$
- Ocorre *overflow* na adição de dois números em CPL2 **se e só se:**
 - ① os dois operandos tiverem o mesmo sinal **e**
 - ② o sinal do resultado for diferente do sinal dos operandos
- Importante: ignora-se sempre o transporte para lá do dígito mais significativo.
- Determinar o simétrico (trocar o sinal) de x ;
 - ① calcular em binário $(2^N - x)$ e ficar com os N bits menos significativos;
 - ② equivalente: trocar todos os bits de x e somar 1;
 - ③ regra prática: da direita para a esquerda, copiar os dígitos de x até ao 1º um; depois inverter os dígitos restantes.
- Multiplicação: tem regras próprias que não trataremos nesta UC.

Representação em complemento para 2: exemplos

■ N = 6, intervalo [-32; 31].

Cálculo do simétrico

Seja $x = -21_{10}$. Então, $-x = 21_{10} = 010101_{\text{cpl2}}$.

1. Como $2^6 = 1000000_2$,

$$\begin{array}{r} 1 & 0 & 0 & 0 & 0 & 0 & 0 \\ - & 0 & 1 & 0 & 1 & 0 & 1 \\ \hline \cancel{X} & 1 & 0 & 1 & 0 & 1 & 1 \end{array}$$

2. $010101 \rightarrow 101010 + 1 = 101011$

3. $010101 \rightarrow 101011$

Logo $x = -21_{10} = 101011_{\text{cpl2}}$

Adição e subtração

Seja $y = -13_{10}$. Logo $-y = 13_{10} = 001101_{\text{cpl2}} \rightarrow y = 110011_{\text{cpl2}}$.

$x + y = 101011 + 110011 = \cancel{X}011110 \rightarrow \text{overflow!}$

$x - y = x + (-y) = 101011 + 001101 = 111000 \rightarrow \text{OK!}$

Extensão do formato de representação

- Com dados representados em N bits, pode surgir a necessidade de obter a representação do mesmo valor em M bits.
- Em geral, é necessário ter $M > N$ para não haver perda de informação.
- Em cada caso particular, depende da gama dos valores realmente processados.
- As regras são as seguintes:

Formato	Regra	Exemplo ($N=4, M=6$)
sem sinal	acrescentar zeros à esquerda	$1010 \rightarrow 001010$
sinal e grandeza	acrescentar zeros à direita do bit de sinal	$1010 \rightarrow 100010$
complemento para 2	replicar o bit de sinal	$1010 \rightarrow 111010$ $0110 \rightarrow 000110$

1 Introdução à representação de informação

2 Representação em binário de números sem sinal

Representação binária

Conversão de base

O sistema de numeração hexadecimal

Operações aritméticas com operandos em binário

3 Representação de números inteiros com sinal

4 Representação de caracteres e símbolos gráficos

5 Representação de números em vírgula flutuante

Conceitos gerais

Formato IEEE-754 precisão simples

Operações aritméticas com operandos normalizados

Código ASCII

American Standard Code for Information Interchange

Dec	Hex	Oct	Chr	Dec	Hex	Oct	HTML	Chr	Dec	Hex	Oct	HTML	Chr	Dec	Hex	Oct	HTML	Chr
0 0	000	NULL		32 20	040	 	Space		64 40	100	@	@		96 60	140	`	`	
1 1	001	SoH		33 21	041	!	!		65 41	101	A	A		97 61	141	a	a	
2 2	002	SoTxt		34 22	042	"	"		66 42	102	B	B		98 62	142	b	b	
3 3	003	EoTxt		35 23	043	#	#		67 43	103	C	C		99 63	143	c	c	
4 4	004	EoT		36 24	044	$	\$		68 44	104	D	D		100 64	144	d	d	
5 5	005	Enq		37 25	045	%	%		69 45	105	E	E		101 65	145	e	e	
6 6	006	Ack		38 26	046	&	&		70 46	106	F	F		102 66	146	f	f	
7 7	007	Bell		39 27	047	'	'		71 47	107	G	G		103 67	147	g	g	
8 8	010	Bsp		40 28	050	((72 48	110	H	H		104 68	150	h	h	
9 9	011	HTab		41 29	051))		73 49	111	I	I		105 69	151	i	i	
10 A	012	LFeed		42 2A	052	*	*		74 4A	112	J	J		106 6A	152	j	j	
11 B	013	VTab		43 2B	053	+	+		75 4B	113	K	K		107 6B	153	k	k	
12 C	014	FFeed		44 2C	054	,	,		76 4C	114	L	L		108 6C	154	l	l	
13 D	015	CR		45 2D	055	-	-		77 4D	115	M	M		109 6D	155	m	m	
14 E	016	SOut		46 2E	056	.	.		78 4E	116	N	N		110 6E	156	n	n	
15 F	017	SIn		47 2F	057	/	/		79 4F	117	O	O		111 6F	157	o	o	
16 10	020	DLE		48 30	060	0	0		80 50	120	P	P		112 70	160	p	p	
17 11	021	DC1		49 31	061	1	1		81 51	121	Q	Q		113 71	161	q	q	
18 12	022	DC2		50 32	062	2	2		82 52	122	R	R		114 72	162	r	r	
19 13	023	DC3		51 33	063	3	3		83 53	123	S	S		115 73	163	s	s	
20 14	024	DC4		52 34	064	4	4		84 54	124	T	T		116 74	164	t	t	
21 15	025	NAck		53 35	065	5	5		85 55	125	U	U		117 75	165	u	u	
22 16	026	Syn		54 36	066	6	6		86 56	126	V	V		118 76	166	v	v	
23 17	027	EoTB		55 37	067	7	7		87 57	127	W	W		119 77	167	w	w	
24 18	030	Can		56 38	070	8	8		88 58	130	X	X		120 78	170	x	x	
25 19	031	EoM		57 39	071	9	9		89 59	131	Y	Y		121 79	171	y	y	
26 1A	032	Sub		58 3A	072	:	:		90 5A	132	Z	Z		122 7A	172	z	z	
27 1B	033	Esc		59 3B	073	;	;		91 5B	133	[[123 7B	173	{	{	
28 1C	034	FSep		60 3C	074	<	<		92 5C	134	\	\		124 7C	174	|		
29 1D	035	GSep		61 3D	075	=	=		93 5D	135]]		125 7D	175	}	}	
30 1E	036	RSep		62 3E	076	>	>		94 5E	136	^	^		126 7E	176	~	~	
31 1F	037	USep		63 3F	077	?	?		95 5F	137	_	_		127 7F	177		Delete	

charstable.com

Máquina de escrever

Inicialmente, impressoras eram máquinas de escrever sem teclado (*typewriter*)



Fonte: https://commons.wikimedia.org/wiki/File:Erika_9_typewriter.jpg

Código ISO-8859-1 (Latin-1)

ISO/IEC 8859-1:1998, Information technology—Part 1: Latin alphabet No. 1

ISO-8859-1																
	x0	x1	x2	x3	x4	x5	x6	x7	x8	x9	xA	xB	xC	xD	xE	xF
0x	NUL	SOH	STX	ETX	EOT	ENQ	ACK	BEL	BS	HT	LF	VT	FF	CR	SO	SI
1x	DLE	DC1	DC2	DC3	DC4	NAK	SYN	ETB	CAN	EM	SUB	ESC	FS	GS	RS	US
2x	SP	!	"	#	\$	%	&	'	()	*	+	,	-	.	/
3x	0	1	2	3	4	5	6	7	8	9	:	;	<	=	>	?
4x	@	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O
5x	P	Q	R	S	T	U	V	W	X	Y	Z	[\]	^	_
6x	`	a	b	c	d	e	f	g	h	i	j	k	l	m	n	o
7x	p	q	r	s	t	u	v	w	x	y	z	{		}	~	DEL
8x	PAD	HOP	BPH	NBH	IND	NEL	SSA	ESA	HTS	HTJ	VTS	PLD	PLU	RI	SS2	SS3
9x	DCS	PU1	PU2	STS	CCH	MW	SPA	EPA	SOS	SGCI	SCI	CSI	ST	OSC	PM	APC
Ax	NBSP	í	¢	£	¤	¥	₩	₪	„	©	™	«	»	–	®	–
Bx	°	±	²	³	‘	μ	¶	·	¸	¹	º	»	¼	½	¾	ܶ
Cx	À	Á	Â	Ã	Ä	Å	Æ	Ç	È	É	Ê	Ï	Í	Ӯ	ӻ	Ӽ
Dx	Đ	Ñ	Ò	Ó	Ô	Ö	Ö	×	Ø	Ù	Ú	Û	Ü	Ý	Þ	Ը
Ex	à	á	â	ã	ä	å	æ	ç	è	é	ê	ë	ì	í	î	ĩ
Fx	ö	ñ	ò	ó	ô	ö	ö	÷	ø	ù	ú	û	ü	ý	þ	ÿ

Fonte: <https://www.alsacreations.com/>

1 Introdução à representação de informação

2 Representação em binário de números sem sinal

Representação binária

Conversão de base

O sistema de numeração hexadecimal

Operações aritméticas com operandos em binário

3 Representação de números inteiros com sinal

4 Representação de caracteres e símbolos gráficos

5 Representação de números em vírgula flutuante

Conceitos gerais

Formato IEEE-754 precisão simples

Operações aritméticas com operandos normalizados

Notação científica

- Em muitas situações existe a necessidade de representar uma gama alargada de números, desde números muito próximos de zero até números de grande magnitude: cálculo científico, contabilidade, computação gráfica, etc.
- A chamada *notação científica* é o formato mais usado.
- Nesta notação em decimal, os números têm o formato $d_0.d_1d_2d_3\dots \times 10^i$ em que i é um número inteiro qualquer.

Exemplos

$$17,987 = 1,7987 \times 10^1$$

$$0,0004567 = 4,567 \times 10^{-4}$$

$$1,756 = 175,6 \times 10^{-2} = 0,001756 \times 10^3 = \dots$$

- Para se ter uma única representação, impõe-se uma condição de *normalização*. Exemplo: os números devem ter apenas 1 dígito ($\neq 0$) à esquerda da vírgula.

$$4,567 \times 10^{-4} \checkmark \quad 0,0004567 \times 10^0 \text{ } \textcolor{red}{X} \quad 456,7 \times 10^{-6} \text{ } \textcolor{red}{X}$$

Norma para representação em vírgula flutuante

- Notação científica é designada por representação em *vírgula flutuante* (“floating point”).
- A parte $d_0.d_1d_2d_3\dots$ é designada por *mantissa*.
- Existem muitas variantes possíveis, mas os formatos usados atualmente são definidos pela norma IEEE-754 (versões em 1985 e 2008) e pela norma internacional ISO/IEC/IEEE 60559:2011 (conteúdo idêntico).
- A norma define várias representações de precisão diferente.
- O formato de precisão simples ocupa 32 bits; o de precisão dupla ocupa 64 bits.
(Existem mais)
- Existe uma representação não normalizada para números próximos de zero.
- A norma define vários modos de arredondamento.
- A norma também define representações para infinito, erro, etc.
(NaN = Not a Number)

Definição do formato VF de precisão simples

→ O formato IEEE-754 codifica números representados em binário na seguinte forma *normalizada*:

$$\pm 1, F \times 2^K$$

- **F** parte fracionária da mantissa normalizada $1, \dots$ (em binário)
- **K** expoente inteiro na gama $[-126; +127]$

→ A organização do formato é a seguinte:

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
S	K*																											parte fracionária (F)			
1	8 bits																											23 bits			

- **S** 0 se positivo, 1 se negativo (sinal)
- **K*** $K^* = K + 127$ (em binário) — codificação excesso-127
Portanto, $K^* \in [1; 254]$

→ Generalização: Para um expoente de E bits usa-se a notação excesso $2^{(E-1)} - 1$

Conversão decimal ↔ VF

Conversão decimal → VF

$X = -12,25$ em binário $|X| = 1100,01_2 = 1,10001_2 \times 2^3$

$$K = 3 \rightarrow K^* = 127 + 3 = 130 = 10000010_2$$

31 30 29 28 27 26 25 24 23 22 21 20 19 18 17 16 15 14 13 12 11 10 9 8 7 6 5 4 3 2 1 0

1	10000010	10001000000000000000000000000000
---	----------	----------------------------------

1100 0001 0100 0100 0000 0000 0000 0000 → C1440000_H

Conversão VF → decimal

Y é representado em VF pelo padrão 41B90000_H. Valor decimal?

41B90000_H → 0100 0001 1011 1001 0000 0000 0000

0	10000011	01110010000000000000000000000000
---	----------	----------------------------------

$$K^* = 10000011_2 = 131 \rightarrow K = 131 - 127 = 4$$

$$|Y| = 1,0111001_2 \times 2^4 = 10111,001_2 = 23,125 \rightarrow Y = 23,125 \quad \text{porque } S = 0$$

Casos especiais

➡ $K^* = 255$ Situações excepcionais (erro, infinito, etc) NaN

➡ $K^* = 0$ número não normalizado (*subnormal*).

Neste caso, o número representado é

$$\pm 0.F \times 2^{-126}$$

Exemplo: $2^{-127} = 0,1 \times 2^{-126}$ é um número subnormal
com $F = 10000000000000000000_2$ (são 22 zeros)

O bit da mantissa à esquerda da vírgula é (implicitamente) 0.

➡ O número zero também é subnormal: $F = 0, K^* = 0$.

➡ O uso de números subnormais permite uma aproximação mais gradual a zero, mas estes têm uma precisão inferior à dos números normalizados.

Regras para adição / subtração

- O formato VF corresponde a uma representação em *sinal e grandeza*.
- Calcular $C = A + B$ em VF

- ① Comparar $|A|$ e $|B|$: $X = \max(|A|, |B|)$ e $Y = \min(|A|, |B|)$
- ② Alterar a mantissa de Y para este ter o mesmo expoente que X .
- ③ Se $\text{sinal}(A) = \text{sinal}(B)$
 - ① $|C| = X + Y$
 - ② $\text{sinal}(C) = \text{sinal}(A)$
- ④ Se $\text{sinal}(A) \neq B$
 - ① $|C| = X - Y$
 - ② $\text{sinal}(C) = \text{sinal}$ do operando de maior valor absoluto
- ⑤ Normalizar C (se necessário)

- Subtração é transformada em adição: $C = A - B = A + (-B)$

Regras para a multiplicação

■ Calcular $C = A \times B$ em VF

- ① Se $\text{sinal}(A)=\text{sinal}(B)$ então $\text{sinal}(C)=0$ senão $\text{sinal}(C)=1$
- ② $\text{Expoente}(C)=\text{Expoente}(A)+\text{Expoente}(B)$
- ③ $\text{Mantissa}(C)=\text{Mantissa}(A) \times \text{Mantissa}(B)$
- ④ Normalizar C (se necessário)

■ Atenção: a soma de dois operandos codificados em excesso-127 faz-se somando os dois operandos normalmente e subtraindo 127. (Porquê?)

$$E_C = E_A + E_B - 127$$

em que E_A , E_B e E_C são os expoentes *codificados* dos correspondentes operandos.

Referências

- COD4** D. A. Patterson & J. L. Hennessey, Computer Organization and Design, 4 ed.
- COD3** D. A. Patterson & J. L. Hennessey, Computer Organization and Design, 3 ed.

Os tópicos tratados nesta apresentação são descritos na seguinte secção de [COD4]:

- apêndice C, secção C.9

Também são tratados na seguinte secção de [COD3]:

- apêndice B, secção B.9