Insper

Elementos de Sistemas

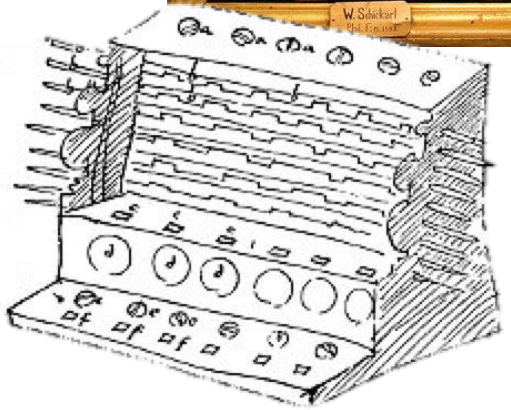
Descrição de Dados Digitais

" A música é um exercício de aritmética inconsciente da alma."
"Musica est exercitium arithmeticae occultum nescientis se numerare animi."
Gottfried Wilhelm von Leibniz (1646 - 1716) matemático e filósofo alemão

Ciência da Computação - Luciano Soares <lpsoares@insper.edu.br>

Primeiras Máquinas de Calcular

Wilhelm Schickard (1592–1635) construiu em 1623 uma calculadora para seu amigo astrônomo Johannes Kepler. Esta é uma das mais antigas calculadoras mecânicas conhecidas capazes de realizar as quatro operações. Seu projeto foi redescoberto no século XX a partir de desenhos e correspondência.



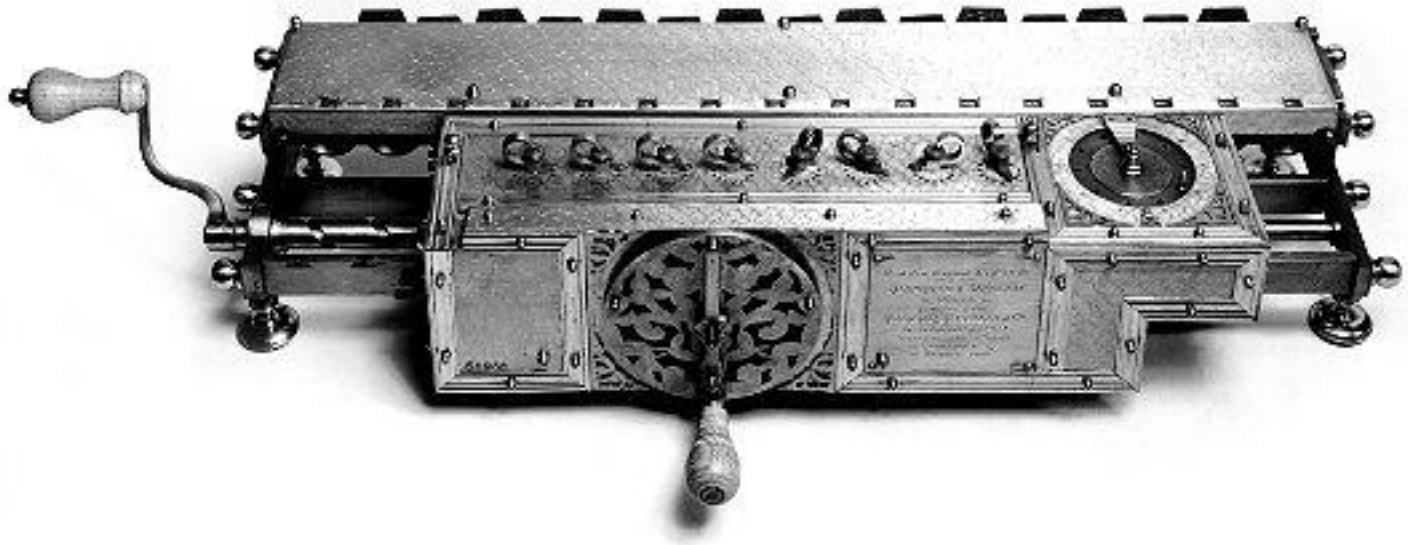
Primeiras Máquinas de Calcular

Blaise Pascal (1623-1662) inventou e produziu em 1642 a Pascaline. Ela só podia fazer adição e subtração, manipulando os números inscritos em seus mostradores. Ele construiu 50 deles ao longo de 10 anos, embora só tenha vendido 15.

Insper

Sistema Binário de Numeração

Gottfried Wilhelm von Leibniz (1646-1716) é creditado como um dos inventores do cálculo diferencial e integral. Leibniz desenvolveu e publicou uma das primeiras exposições sistemáticas modernas da aritmética binária (base 2). Em 1672 Leibniz começou a inventar uma máquina capaz de fazer as 4 operações aritméticas, o Staffelwalze.



Números

Os símbolos "1", "2", "3" levaram muitos séculos para surgir.

O sistema decimal é o mais amplamente usado pelas civilizações modernas.

Porém, muitas outras civilizações usaram outras formas de contar do que usamos hoje.

Marcações em madeira, ossos e outros materiais estão entre as formas mais antigas de registro de quantidades.

Como os algarismos chegaram a nós

No Liber Abaci (1202), Leonardo Fibonacci ajudou a difundir na Europa a numeração indo-arábica.

I, II, III, IV, V, VI, VII, VIII, IX

Fibonacci

0, 1, 2, 3, 4, 5, 6, 7, 8, 9

Base e Posicionamento dos Números

Os sistemas numéricos atuais, como o decimal indo-arábico, são chamados de sistemas posicionais, o que significa que a posição de um símbolo tem influência sobre o valor do símbolo, no caso múltiplos de 10. A prevalência histórica da base 10 é geralmente associada ao fato de termos dez dedos.

Base e Posicionamento dos Números

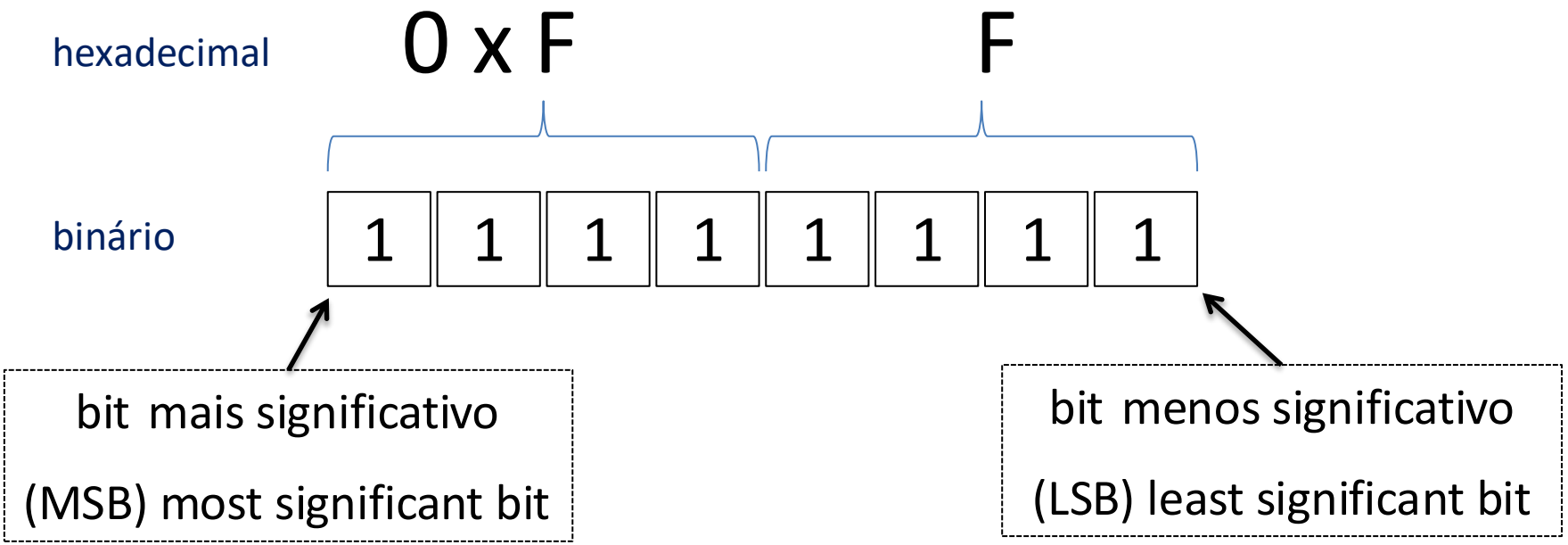
Decimal	Binário	Octal	Hexadecimal
0	0	0	0
1	1	1	1
2	10	2	2
3	11	3	3
4	100	4	4
5	101	5	5
6	110	6	6
7	111	7	7
8	1000	10	8
9	1001	11	9
10	1010	12	A
11	1011	13	B
12	1100	14	C
13	1101	15	D
14	1110	16	E
15	1111	17	F
16	10000	20	10
17	10001	21	11
18	10010	22	12

Além do radix (base) decimal, qualquer outra base pode ser usada. Por exemplo a base 12 tem a vantagem de ser divisível por 1, 2, 3, 4, e 6.

Internamente os computadores funcionam com o sistema binário, porém é mais comum usarmos base 10 no dia a dia, e eventualmente a base 16 como representação compacta de dados digitais.

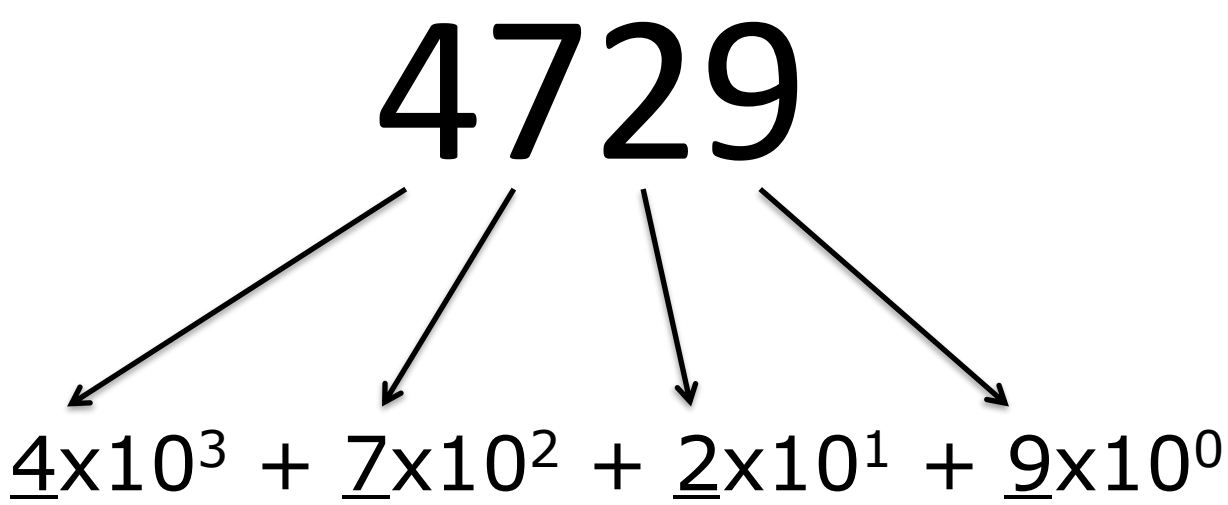
Dados Binários (e Hexadecimais)

Cada dígito binário representa um bit. Um byte possui 8 bits. Dados são frequentemente organizados em grupos de 1, 2, 4 ou 8 bytes (8, 16, 32 ou 64 bits).



Insper

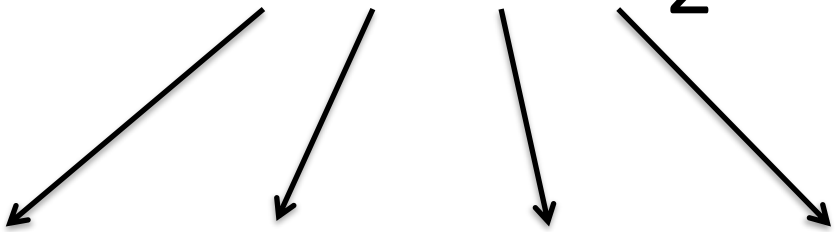
Exemplo em base 10



Base 10

Exemplo em base 2

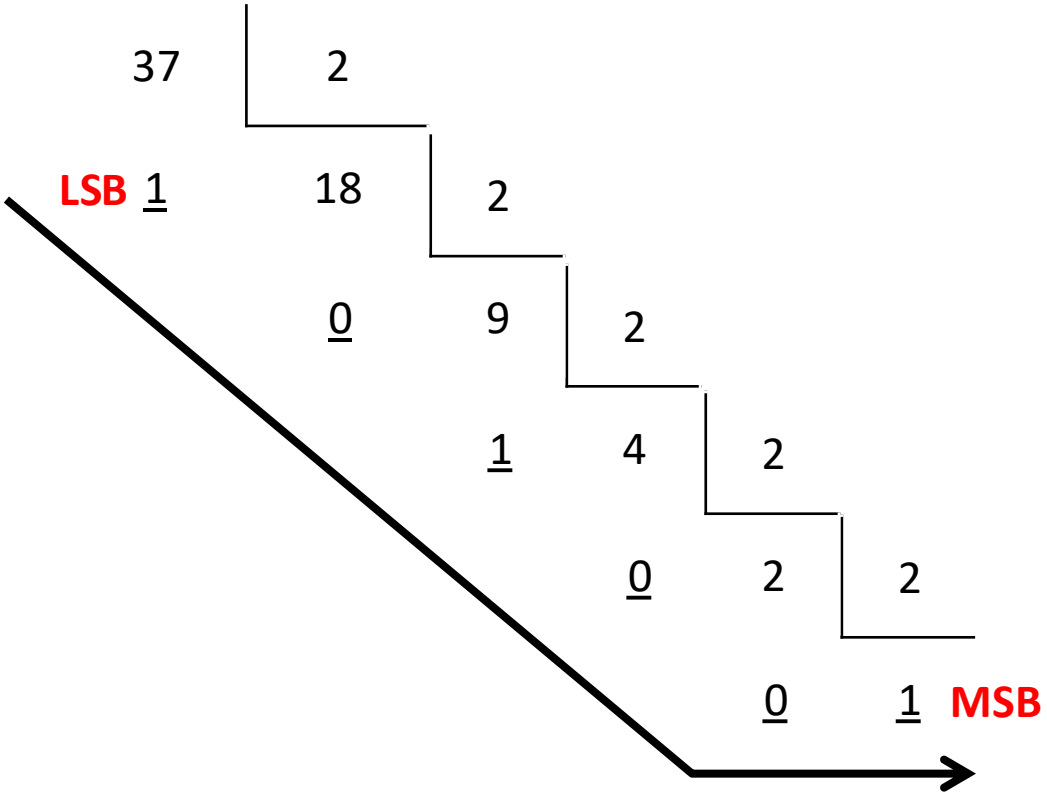
$$1001_2$$



$$\underline{1}x2^3 + \underline{0}x2^2 + \underline{0}x2^1 + \underline{1}x2^0 =$$

Base 2

Conversão para Números Binários



$$37 = 2^5+2^2+2^0 = 32 + 4 + 1 = 100101$$

MSB LSB

Insper

Conversão para Números Binários

37

37 – 32 = 5

5 – 4 = 1

1 – 1 = 0

64	32	16	8	4	2	1
0	1	0	0	1	0	1

13

Conversão para Base 10

Exemplo:

Converter 6734_8 para a base 10:

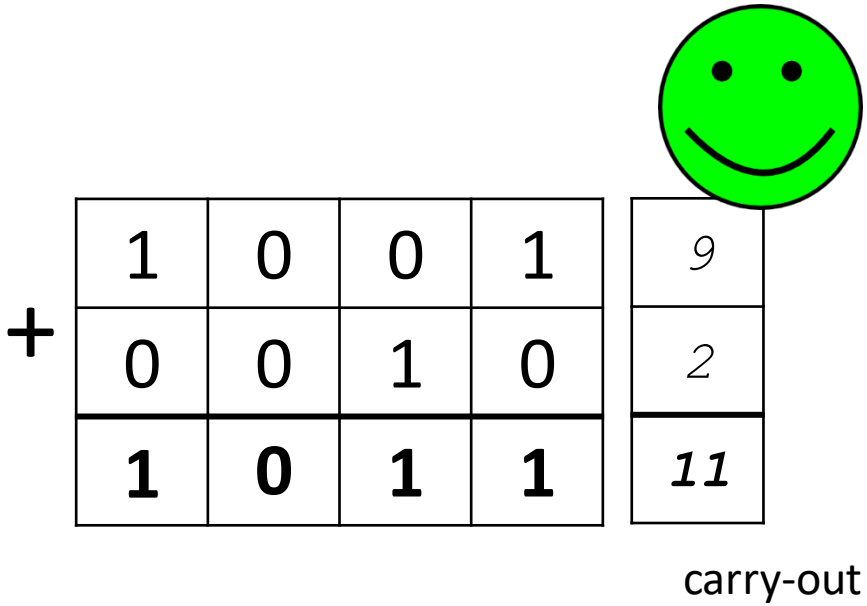
	Base 8	*Conversão*	*Base 10*
	$6 * 8^3$	$= 6 * 512$	$= 3072$
+	$7 * 8^2$	$= 7 * 64$	$= 448$
+	$3 * 8^1$	$= 3 * 8$	$= 24$
+	$4 * 8^0$	$= 4 * 1$	$= 4$
		Total	**3548**

Converter $1F8_{16}$ para a base 10:

	Base 16	*Conversão*	*Base 10*
	$1 * 16^2$	$= 1 * 256$	$= 256$
+	$F * 16^1$	$= 15 * 16$	$= 240$
+	$8 * 16^0$	$= 8 * 1$	$= 8$
		Total	**504**

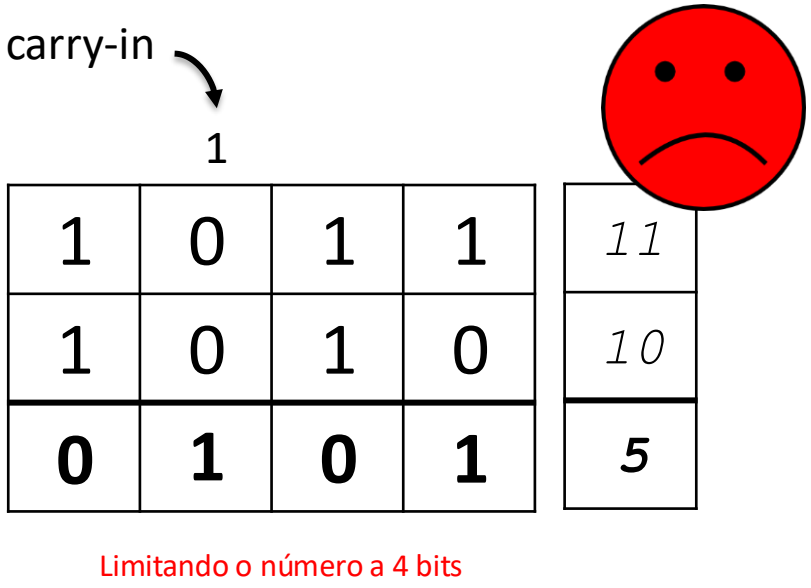
Adição Binária

Fazer operações com números binários é análogo a fazer operações com números decimais.



	1	0	0	1	9
+	0	0	1	0	2
	1	**0**	**1**	**1**	**11**

carry-out



carry-in

1

	1	0	1	1	11
+	1	0	1	0	10
1	**0**	**1**	**0**	**1**	**5**

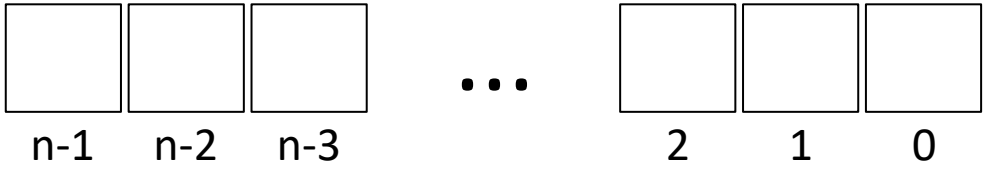
Limitando o número a 4 bits

Para o correto cálculo da soma binária, os *carries* (vai um) precisariam ser tratados. O carry-out indica que a soma unsigned excedeu a faixa representável e também precisa ser tratado.

Inteiros Positivos e Negativos

Números Inteiros em Complemento de 2 (um)

- Supondo um conjunto de números inteiros de n bits:



- valores não negativos começam com 0
- valores negativos começam com 1
- Metade dos números seriam não negativos e estariam entre:

  0 e $(2^{(n-1)} - 1)$

- A outra metade compreenderia o espaço de:

  $-2^{(n-1)}$ e -1

Insper

Complemento de 2 (dois)

Complemento de 2 é a representação mais comum de números inteiros com sinal em computadores.

Para obter a representação de −x a partir de x, em uma largura fixa de n bits:

- Inverta todos os bits
- Some 1 (um) ao resultado.

Essa representação simplifica as operações de soma e subtração.

Exemplo: 3 – 5.

Invertendo 5

5	0101
inv + 1	1010 + 1
-5	1011

Subtraindo 5 de 3

3	0011
+ (-5)	1011
3-5 = -2	**1110**

Decimal	Binário(C'2)
7	0111
6	0110
5	0101
4	0100
3	0011
2	0010
1	0001
0	0000
-1	1111
-2	1110
-3	1101
-4	1100
-5	1011
-6	1010
-7	1001
-8	1000

Insper

Complemento de Dois no Byte

2^8 valores $\rightarrow$ 256 valores

- sem sinal: 0 a 255 (unsigned)

1	1	1	1	1	1	1	1	= 255	(0xFF)

- inteiros: -128 a 127 (signed)

bit mais à esquerda (MSB) caracteriza valor negativo

1	0	0	0	0	0	0	0	= -128	(0x80)
1	0	0	0	0	0	0	1	= -127	(0x81)
1	0	0	0	0	0	1	0	= -126	(0x82)
1	1	1	1	1	1	1	1	= -1	(0xFF)
0	1	1	1	1	1	1	1	= 127	(0x7F)

Inteiros de vários bits

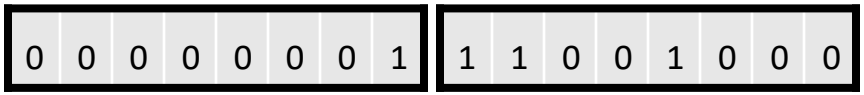
unsigned : 0 a 255
signed : –128 a 127

1 byte

67

unsigned : 0 a 65 535
signed : –32 768 a 32 767

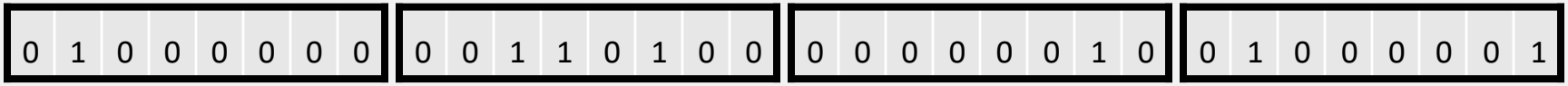
2 bytes



456

unsigned : 0 a 4 294 967 295
signed : –2 147 483 648 a 2 147 483 647

4 bytes



1.077.150.273

Insper

BCD (Binary Coded Decimal)

O BCD (Codificação binária decimal) usa uma representação binária para números decimais.

Por exemplo:

$314159_{10} = 001100010100000101011001_{BCD}$

3 1 4 1 5 9

$271823_{10} = 001001110001100000100011_{BCD}$

2 7 1 8 2 3

Decimal	BCD
0	0000
1	0001
2	0010
3	0011
4	0100
5	0101
6	0110
7	0111
8	1000
9	1001

Ponto Flutuante

$$(-1)^S \cdot (1+F_2) \cdot 2^{E-C}$$

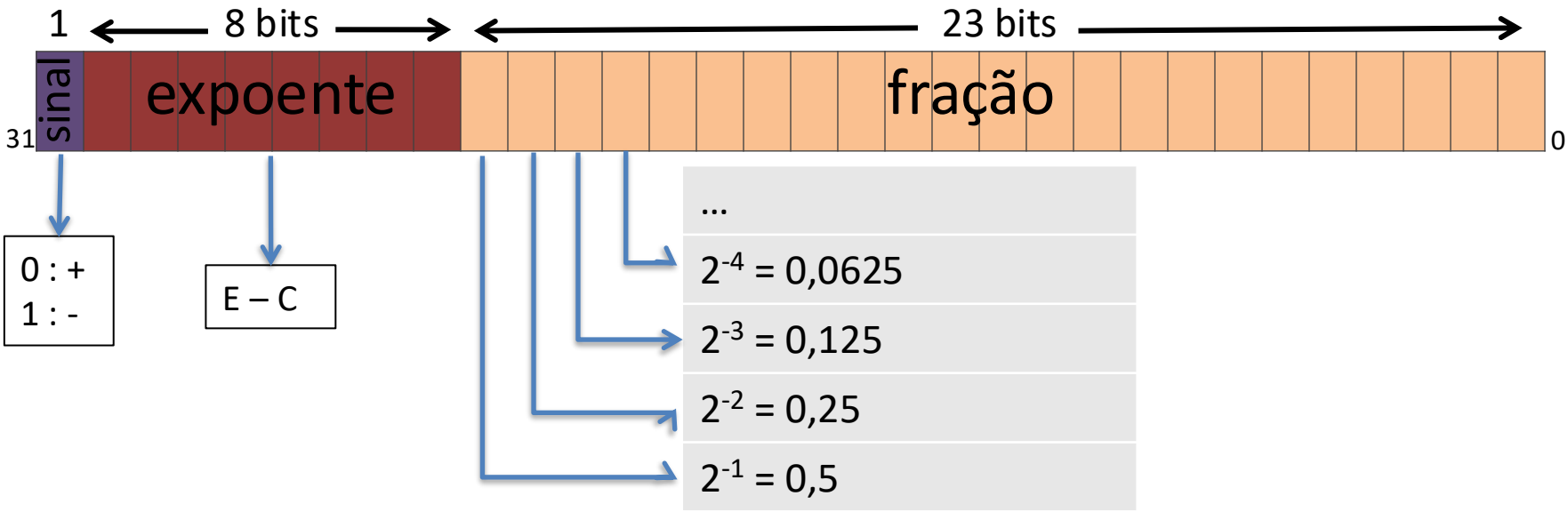
S: sinal

F: fração

E: expoente

C: 127 (single precision)

- ***IEEE 754-2008 Precisão Simples***



Obs: A especificação também trata casos especiais, como: ±∞ e NaN

Arquivos de Texto

Os arquivos podem ser divididos assim em dois sub-grupos:

- Texto;
- Binários.

Arquivos são lidos como uma sequência de bytes

– Estes bytes podem estar organizados segundo uma codificação/formato que só um programa específico entenda;
– Em arquivos de texto, sequências de bytes são interpretadas como caracteres segundo uma codificação, como ASCII.
Programas de edição de texto puro trabalham com estes arquivos.

Tabela ASCII

(American Standard Code for Information Interchange)

000	(nul)	016 ►	(dle)	032	sp	048	0	064	@	080	P	096	`	112	p
001 ☺	(soh)	017 ◄	(dc1)	033	!	049	1	065	A	081	Q	097	a	113	q
002 ☻	(stx)	018 ↕	(dc2)	034	"	050	2	066	B	082	R	098	b	114	r
003 ♥	(etx)	019 ‼	(dc3)	035	#	051	3	067	C	083	S	099	c	115	s
004 ♦	(eot)	020 ¶	(dc4)	036	$	052	4	068	D	084	T	100	d	116	t
005 ♣	(enq)	021 §	(nak)	037	%	053	5	069	E	085	U	101	e	117	u
006 ♠	(ack)	022 ▬	(syn)	038	&	054	6	070	F	086	V	102	f	118	v
007 •	(bel)	023 ↨	(etb)	039	'	055	7	071	G	087	W	103	g	119	w
008 ◘	(bs)	024 ↑	(can)	040	(	056	8	072	H	088	X	104	h	120	x
009	(tab)	025 ↓	(em)	041	)	057	9	073	I	089	Y	105	i	121	y
010	(lf)	026	(eof)	042	*	058	:	074	J	090	Z	106	j	122	z
011 ♂	(vt)	027 ←	(esc)	043	+	059	;	075	K	091	[	107	k	123	{
012 ♀	(np)	028 ∟	(fs)	044	,	060	<	076	L	092	\	108	l	124	|
013	(cr)	029 ↔	(gs)	045	-	061	=	077	M	093	]	109	m	125	}
014 ♫	(so)	030 ▲	(rs)	046	.	062	>	078	N	094	^	110	n	126	~
015 ☼	(si)	031 ▼	(us)	047	/	063	?	079	O	095	_	111	o	127	⌂

Em arquivos o Windows usa os caracteres CR e LF para terminar uma linha, enquanto sistemas Unix-like (Linux/macOS) usam somente o LF para terminar a linha.

Insper

Unicode e codificações de caracteres

Unicode → associa caracteres a *code points*

UTF-8 - codifica um *code point* usando 1 a 4 bytes
UTF-16 - utiliza uma ou duas unidades de 16 bits

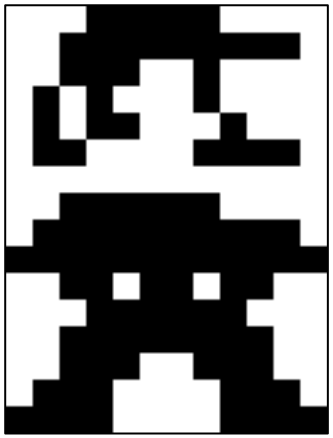
A → **U+0041** → UTF-8: **41**

é → **U+00E9** → UTF-8: **C3 A9**

😀 → **U+1F600** → UTF-8: **F0 9F 98 80**

Representando imagens em arquivos-texto

Netpbm é um formato aberto para descrever imagens em texto.

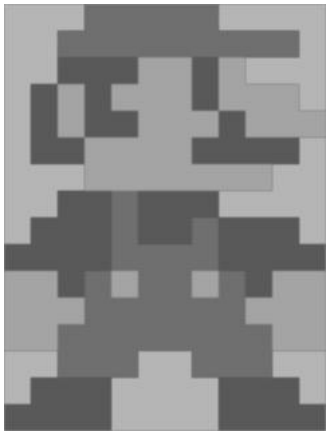


Preto: 1
Branco: 0

mario.pbm

```
P1
12 16
000111110000
001111111110
001110010000
010100010000
010110001000
011000011110
000000000000
001111110000
011111111110
111111111111
001101101100
000111111000
001111111100
001110011100
011100001110
111100001111
```

Representando imagens em arquivos-texto



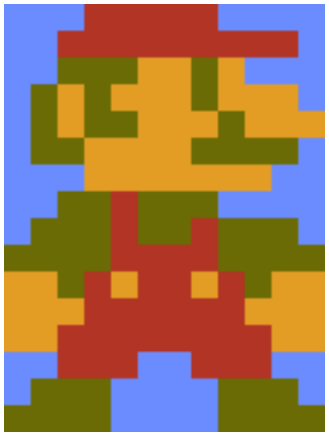
Tons de Cinza:
089
111
164
181

mario.pgm

```
P2
12 16
255
181 181 181 111 111 111 111 111 181 181 181 181
181 181 111 111 111 111 111 111 111 111 111 181
181 181 089 089 089 164 164 089 164 181 181 181
181 089 164 089 164 164 164 089 164 164 164 181
181 089 164 089 089 164 164 164 089 164 164 164
181 089 089 164 164 164 164 089 089 089 089 181
181 181 181 164 164 164 164 164 164 164 181 181
181 181 089 089 111 089 089 089 181 181 181 181
181 089 089 089 111 089 089 111 089 089 089 181
089 089 089 089 111 111 111 111 089 089 089 089
164 164 089 111 164 111 111 164 111 089 164 164
164 164 164 111 111 111 111 111 111 164 164 164
164 164 111 111 111 111 111 111 111 111 164 164
181 181 111 111 111 181 181 111 111 111 181 181
181 089 089 089 181 181 181 181 089 089 089 181
089 089 089 089 181 181 181 181 089 089 089 089
```


Insper

Representando imagens em arquivos-texto



azul: #6B8CFF
vermelho: #B13425
verde: #6A6B04
laranja: #E39D25

mario.ppm

```
P3
12 16
255
107 140 255 107 140 255 107 140 255 177 052 037 177 052 037 177 052 037 177 052 037 177 052 037 107 140 255 107 140 255 107 140 255 107 140 255
107 140 255 107 140 255 177 052 037 177 052 037 177 052 037 177 052 037 177 052 037 177 052 037 177 052 037 177 052 037 177 052 037 107 140 255
107 140 255 107 140 255 106 107 004 106 107 004 106 107 004 227 157 037 227 157 037 106 107 004 227 157 037 107 140 255 107 140 255 107 140 255
107 140 255 106 107 004 227 157 037 106 107 004 227 157 037 227 157 037 227 157 037 106 107 004 227 157 037 227 157 037 227 157 037 107 140 255
107 140 255 106 107 004 227 157 037 106 107 004 106 107 004 227 157 037 227 157 037 227 157 037 106 107 004 227 157 037 227 157 037 227 157 037
107 140 255 106 107 004 106 107 004 227 157 037 227 157 037 227 157 037 227 157 037 106 107 004 106 107 004 106 107 004 106 107 004 107 140 255
107 140 255 107 140 255 107 140 255 227 157 037 227 157 037 227 157 037 227 157 037 227 157 037 227 157 037 227 157 037 227 157 037 107 140 255
107 140 255 107 140 255 106 107 004 106 107 004 177 052 037 106 107 004 106 107 004 106 107 004 107 140 255 107 140 255 107 140 255 107 140 255
107 140 255 106 107 004 106 107 004 106 107 004 177 052 037 106 107 004 106 107 004 177 052 037 106 107 004 106 107 004 106 107 004 107 140 255
106 107 004 106 107 004 106 107 004 106 107 004 177 052 037 177 052 037 177 052 037 177 052 037 106 107 004 106 107 004 106 107 004 106 107 004
227 157 037 227 157 037 106 107 004 177 052 037 227 157 037 177 052 037 177 052 037 227 157 037 177 052 037 106 107 004 227 157 037 227 157 037
227 157 037 227 157 037 227 157 037 177 052 037 177 052 037 177 052 037 177 052 037 177 052 037 177 052 037 227 157 037 227 157 037 227 157 037
227 157 037 227 157 037 177 052 037 177 052 037 177 052 037 177 052 037 177 052 037 177 052 037 177 052 037 177 052 037 227 157 037 227 157 037
107 140 255 107 140 255 177 052 037 177 052 037 177 052 037 107 140 255 107 140 255 177 052 037 177 052 037 177 052 037 107 140 255 107 140 255
107 140 255 106 107 004 106 107 004 106 107 004 107 140 255 107 140 255 107 140 255 107 140 255 106 107 004 106 107 004 106 107 004 107 140 255
106 107 004 106 107 004 106 107 004 106 107 004 107 140 255 107 140 255 107 140 255 107 140 255 106 107 004 106 107 004 106 107 004 106 107 004
```

Insper

www.insper.edu.br