FACULDADE DE TECNOLOGIA SENAC RJ

ANÁLISE E DESENVOLVIMENTO DE SOFTWARE

ARQUITETURA DE COMPUTADORES

ARITMÉTICA E CODIFICAÇÃO BINÁRIA

SOMA BINÁRIA

$0 + 0 = 0$

$0 + 1 = 1$

$1 + 0 = 1$

$1 + 1 = 10 \quad (2_{10})$

```
 0000 1101
 0000 0111
-------------
 0001 0100
```

Exemplo na base 10:

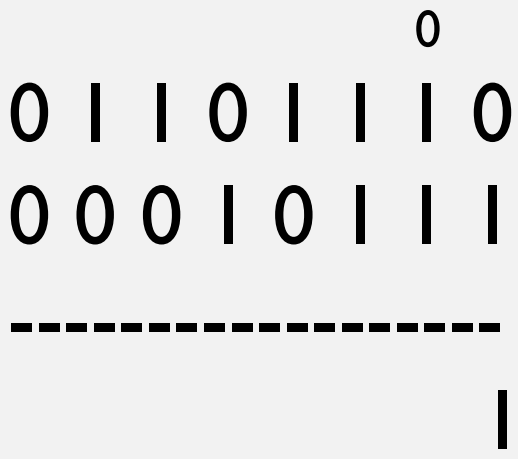
```
 1 + 0 =  1
 1 + 1 =  2
 ...
 1 + 8 =  9
 1 + 9 = 10
```

```
   1
  13
+  7
 ----
  20
```

SOMA BINÁRIA

```
0 1 1 0 1 1 1 0          110
0 0 0 1 0 1 1 1        + 23
-----------------       -----
                        133
```

SOMA BINÁRIA



SOMA BINÁRIA

```
          1 0
0 1 1 0 1 1 1 0
0 0 0 1 0 1 1 1
-----------------
          0 1
```

SOMA BINÁRIA

```
          1  1 0
0 1 1 0 1 1 1 0
0 0 0 1 0 1 1 1
-----------------
        1 0 1
```

SOMA BINÁRIA

```
      1 1 1 0
0 1 1 0 1 1 1 0
0 0 0 1 0 1 1 1
----------------
      0 1 0 1
```

SOMA BINÁRIA

```
      1 1 1 1 0
0 1 1 0 1 1 1 0
0 0 0 1 0 1 1 1
----------------
      0 0 1 0 1
```

SOMA BINÁRIA

```
   1 1 1 1 1 0
0 1 1 0 1 1 1 0
0 0 0 1 0 1 1 1
---------------
    0 0 0 1 0 1
```

SOMA BINÁRIA

```
  1 1 1 1 1 1 0
  0 1 1 0 1 1 1 0
  0 0 0 1 0 1 1 1
  ---------------
    0 0 0 0 1 0 1
```

SOMA BINÁRIA

```
  1 1 1 1 1 1 0
  0 1 1 0 1 1 1 0
  0 0 0 1 0 1 1 1
  ---------------
  1 0 0 0 0 1 0 1
```

SOMA BINÁRIA

Efetue as seguintes adições de números binários:

(a) 11 + 11
(b) 100 + 10
(c) 111 + 11
(d) 110 + 100

Solução (A soma decimal equivalente também é mostrada para referência.)

(a)			(b)			(c)			(d)		
	11	3		100	4		111	7		110	6
	11	3		10	2		11	3		100	4
	----	--		------	--		------	--		------	--
	110	6		110	6		1010	10		1010	10

Faça você: Some 1111 com 1100.

MULTIPLICAÇÃO BINÁRIA

A multiplicação é realizada com números binários da mesma maneira que com números decimais. Ela envolve a formação de produtos parciais, deslocamento de cada produto parcial sucessivo uma posição à esquerda, para então somar todos os produtos parciais.

Realize as seguintes multiplicações binárias:

(a) 11 x 11

(b) 101 x 111

Solução:

(a)

```
  11      3
 x 11    x 3
 ---     --
  11      9
 11
 -----
1001
```

(b)

```
  111      7
 x 101     5
 -----     --
  111     35
 000
111
----------
100011
```

Multiplicação de dois bits:

0 x 0 = 0

0 x 1 = 0

1 x 0 = 0

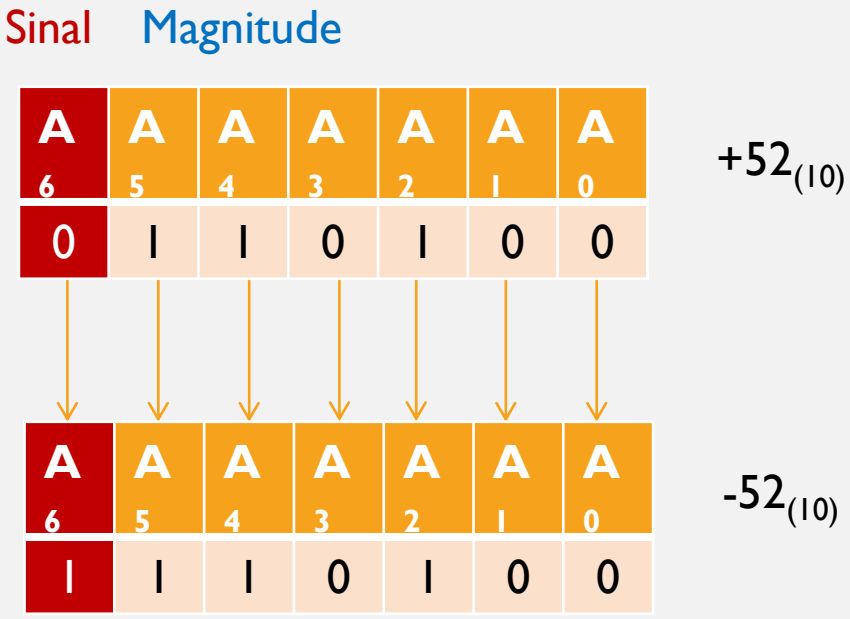
1 x 1 = 1

Faça você: Efetue a multiplicação

1011 x 1010

NÚMEROS BINÁRIOS SINALIZADOS

Sinal-Magnitude ou Sinal-Módulo: processo de representação dos números sinalizados, onde cada número é representado por um bit de sinal (o mais significativo: A_n) e o restante ($A_{n-1}...A_0$) forma sua magnitude ou módulo.



$+52_{(10)}$

$-52_{(10)}$

Binário+	Dec+	Binário-	Dec-
0 000	0	1 000	0
0 001	+1	1 001	-1
0 010	+2	1 010	-2
0 011	+3	1 011	-3
0 100	+4	1 100	-4
0 101	+5	1 101	-5
0 110	+6	1 110	-6
0 111	+7	1 111	-7

SOMA / SUBTRAÇÃO

Procedimento de Soma/Subtração:

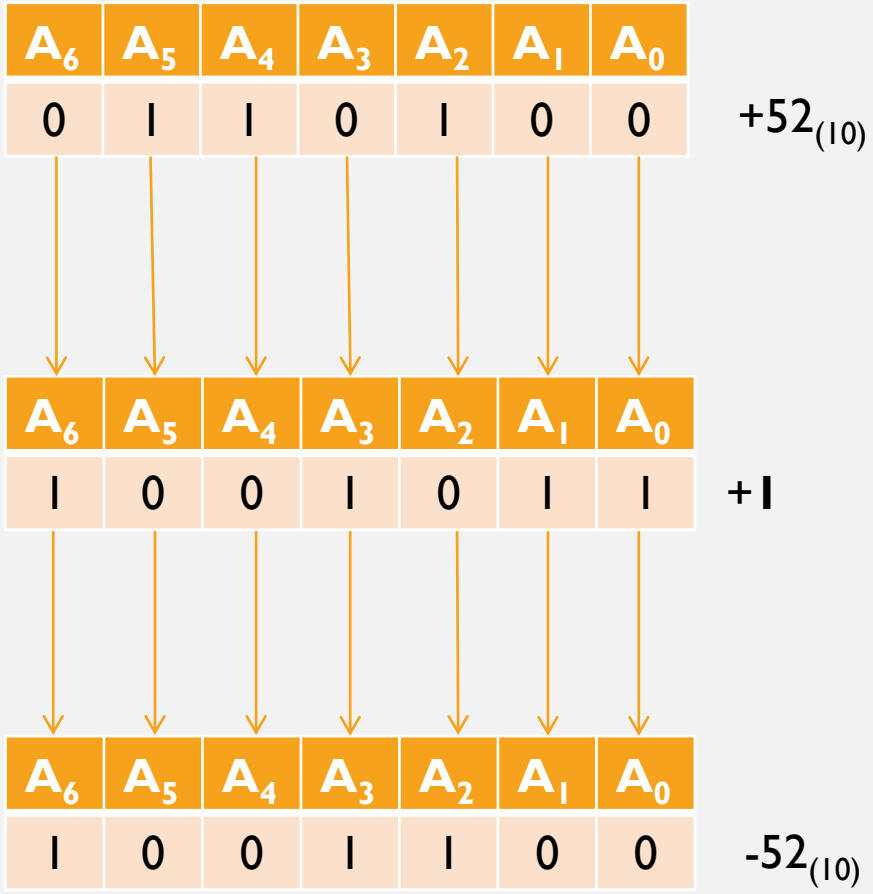
Sinais iguais: **Soma-se** os dois módulos; o sinal do resultado é o mesmo dos operandos; e pode haver estouro de módulo.

Sinais diferentes: Encontra-se o número com maior módulo; **subtrai-se** o menor do maior; e o sinal do resultado é o mesmo do operando de maior módulo.

Desvantagem: Possui lógica que requer vários testes de condições, tornando os circuitos mais complexos.

COMPLEMENTO DE 2

Complemento de Dois: processo de representação dos números sinalizados, usando complemento a base. (O número original tem seus bits invertidos e depois somado 1 ao resultado.)



Binário+	Dec+	Binário-	Dec-
0000	0	1111	-1
0001	+1	1110	-2
0010	+2	1101	-3
0011	+3	1100	-4
0100	+4	1011	-5
0101	+5	1010	-6
0110	+6	1001	-7
0111	+7	1000	-8

COMPLEMENTO DE DOIS

- Com **um byte** ou **oito bits**, podemos representar **256** números diferentes.
- Com **dois bytes** ou dezesseis bits, podemos representar **65.536** números diferentes.
- Com **quatro bytes** ou 32 bits, podemos representar **4,295 x 10^9** números diferentes.

- A fórmula para encontrar o número de combinações diferentes de **n bits**:

  Total de combinações = 2^n

- Para **números sinalizados** na forma do **complemento de 2**, a faixa de valores para números de n bits:

  Faixa = $-(2^{n-1})$ a $+(2^{n-1} - 1)$

  onde existe em cada caso um bit de sinal e n−1 bits de magnitude.
- Exemplo:
  - **quatro bits**: $-(2^3) = -8$ até $(2^3 - 1) = +7$
  - **um byte:** − 128 até + 127
  - **dois bytes:** − 32.768 até + 32.767

SOMA BINÁRIA

Determine o complemento de 2 de 10110010.

Solução:

10110010	*Número binário*
01001101	*Complemento de 1*
1	*Soma-se 1*
01001110	*Complemento de 2*

Faça você: Determine o complemento de 2 de 11001011.

COMPLEMENTO DE 2

Procedimento de Soma:

Dois números positivos: **Soma-se** os dois módulos; o sinal do resultado é o mesmo dos operandos; e pode haver estouro de módulo.

Exemplo (números de seis bits):

(vai-um)	1 1	
	0 0 0 1 1 1	7
	+ 0 1 0 0 1 0	+18
	----------	----
Soma	0 1 1 0 0 1	25

COMPLEMENTO DE 2

Procedimento de Soma:

Dois números negativos: a operação pode ser feita por meio da **soma** dos números complementados.

Exemplo (números de seis bits):

```
                    4   = 000100
(complemento de 2) = 111100

                   19  = 010011
(complemento de 2) = 101101
```

```
(vai-um)  1  1  1
          1 1 1 1 0 0        -4
        + 1 0 1 1 0 1      + -19
          ---------------      ----
Soma      1 0 1 0 0 1        -23

          0 1 0 1 1 1         23
```

COMPLEMENTO DE 2

Procedimento de Soma/Subtração:

Números com sinais diferentes: Caso o número seja positivo, deve ser mantido; caso seja negativo, deve ser complementado; e depois, deve ser feita a **soma**.

Exemplo (números de seis bits): 4 - 19

```
 4  = 000100
19  = 010011
complemento a 2
    = 101101
```

```
(vai-um)  0
          0 0 0 1 0 0          4
        + 1 0 1 1 0 1        -19
          ---------------      ----
Soma      1 1 0 0 0 1        -15

          0 0 1 1 1 1         15
```

COMPLEMENTO DE 2

A representação em complemento 2 tem as seguintes características:

• O bit mais significativo (esquerda) indica o sinal, como na representação sinal-magnitude.

•o processo serve para converter um número de positivo para negativo e de negativo para positivo;

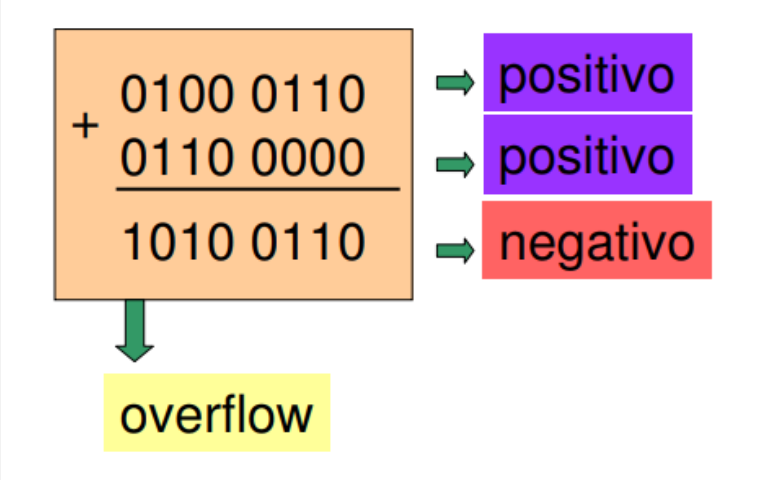
• 0 tem uma representação única: todos os bits são 0;

• A gama de valores que é possível representar com *n* bits é -2^(n-1) ... 2^(n-1) -1.

OVERFLOW

Ocorre sempre que o resultado de uma operação não pode ser representado no hardware disponível.

Overflow é um "estouro" da magnitude de um valor, no número limitado de bits de uma máquina.



Um registrador de 8 bits só pode representar valores entre -128 até +127.

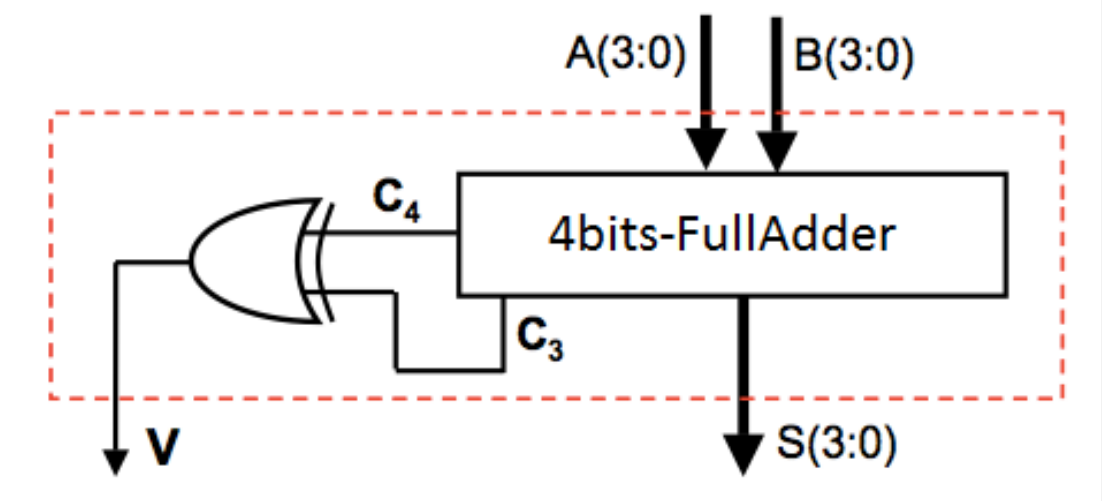
Ou seja, $0100\ 0110_2$ (70_{10}) + $0110\ 0000_2$ (96_2), que deveria resultar em $+166_d$ (usando o sinal positivo), vai ser interpretado pelo computador como: -89 / C2(01011001_2)

BIT DE OVERFLOW (V)

Bit sinalizador de condição de estouro de representação numérica em um dado conjunto de bits.

Exemplo: Soma de 4 bits

A	B	S	C_3	C_4	V
0101 (5)	0110 (6)	1011 (11)	1	0	1
0101 (5)	0010 (2)	0111 (7)	0	0	0
0101 (5)	1010 (-6)	1111 (-1)	0	0	0
0110 (6)	1011(-5)	0001 (1)	1	1	0
1011 (-5)	1010 (-6)	0101 (-11)	0	1	1



OVERFLOW

Considere as seguintes somas de valores representados em Complemento a Dois:

a) 001011 + 100001 com seis bits.

b) 101011 + 111001 com seis bits.

c) 1011 + 1101 com quatro bits.

d) 0011 + 0101 com quatro bits.

e) 100001 + 100001 com seis bits.

Sem converter os valores para a base 10, determine quais operações resultam em overflow.

CODIFICAÇÃO ASCII

- Um computador precisa ser capaz de manipular informações não numérica
- Código ASCII (*American Standard Code for Information Interchange*)
  - Possui 7 bits (portanto tem $2^7 = 128$ representações codificadas)
  - Também representa códigos de controle como o <RETURN> ou CR (Carriage Return) (`'\n'` em C) e o <LINEFEED> ou LF (`'\r'` em C)
  - Códigos ASCII representam texto puro (o chamado *plain text*) em computadores e equipamentos de comunicação
- Outros exemplos: ISO 8859 e Unicode (UTF-8, UTF-16), EBCDIC (Mainframes IBM)

CODIFICAÇÃO ASCII (I)

Hex	Name	Meaning	Hex	Name	Meaning
0	NUL	Null	10	DLE	Data Link Escape
1	SOH	Start Of Heading	11	DC1	Device Control 1
2	STX	Start Of TeXt	12	DC2	Device Control 2
3	ETX	End Of TeXt	13	DC3	Device Control 3
4	EOT	End Of Transmission	14	DC4	Device Control 4
5	ENQ	Enquiry	15	NAK	Negative AcKnowledgement
6	ACK	ACKnowledgement	16	SYN	SYNchronous idle
7	BEL	BELI	17	ETB	End of Transmission Block
8	BS	BackSpace	18	CAN	CANcel
9	HT	Horizontal Tab	19	EM	End of Medium
A	LF	Line Feed	1A	SUB	SUBstitute
B	VT	Vertical Tab	1B	ESC	ESCape
C	FF	Form Feed	1C	FS	File Separator
D	CR	Carriage Return	1D	GS	Group Separator
E	SO	Shift Out	1E	RS	Record Separator
F	SI	Shift In	1F	US	Unit Separator

CODIFICAÇÃO ASCII (2)

Hex	Char	Hex	Char	Hex	Char	Hex	Char	Hex	Char	Hex	Char
20	(Space)	30	0	40	@	50	P	60	‘	70	p
21	!	31	1	41	A	51	Q	61	a	71	q
22	"	32	2	42	B	52	R	62	b	72	r
23	#	33	3	43	C	53	S	63	c	73	s
24	$	34	4	44	D	54	T	64	d	74	t
25	%	35	5	45	E	55	U	65	e	75	u
26	&	36	6	46	F	56	V	66	f	76	v
27	’	37	7	47	G	57	W	67	g	77	w
28	(	38	8	48	H	58	X	68	h	78	x
29	)	39	9	49	I	59	Y	69	i	79	y
2A	*	3A	:	4A	J	5A	Z	6A	j	7A	z
2B	+	3B	;	4B	K	5B	[	6B	k	7B	{
2C	,	3C	<	4C	L	5C	\	6C	l	7C	|
2D	-	3D	=	4D	M	5D	]	6D	m	7D	}
2E	.	3E	>	4E	N	5E	^	6E	n	7E	~
2F	/	3F	?	4F	O	5F	_	6F	o	7F	DEL

CODIFICAÇÃO ASCII

- A seguinte sequência de bits é uma mensagem codificada em ASCII. Que mensagem é essa?

1001000 1000101 1001100 1010000

- Solução:

Converta cada código de 7 bits em seu equivalente Hexa:

48 45 4C 50

Localize na tabela:

H E L P

UNICODE (1)

Unicode é um padrão que permite aos computadores representar e manipular, de forma consistente, texto de qualquer sistema de escrita existente. Publicado no livro *The Unicode Standard* o padrão consiste de pouco mais de 107 mil caracteres, um conjunto de diagramas de códigos para referência visual, uma metodologia para codificação e um conjunto de codificações padrões de caracteres, uma enumeração de propriedades de caracteres como caixa alta e caixa baixa, um conjunto de arquivos de computador com dados de referência, além de regras para normalização, decomposição, ordenação alfabética e renderização.

Por outro lado, a UTF-8 é uma codificação de muito usada, e que maximiza a compatibilidade com ASCII. Utiliza entre um e quatro bytes por código e, sendo compacta para o sistemas latino e compatível com ASCII nos códigos até 127, fornece um padrão *de facto* de codificação para a conversão de textos para o formato Unicode. É usada pelas mais recentes distribuições Linux como uma substituta para codificações legadas na manipulação de texto. A UTF-8 representa uma forma de otimizar o espaço alocado para textos Unicode. Considerando por exemplo um texto escrito em língua inglesa, percebe-se que raramente são utilizados caracteres fora do escopo do ASCII, isto é, os primeiros 127 códigos Unicode. Isso significa que se for utilizada uma codificação de largura fixa de 16 bits, o segundo byte de cada caracter muito provavelmente estará vazio, nulo, inutilizado. Para arquivos grandes a sobrecarga desse espaço inútil alocado passa a ser relevante. Tendo uma largura variada, o UTF-8 define que caracteres ASCII são representados com somente um byte. Além de otimizar o espaço alocado no caso de caracteres ASCII, isso garante a paridade entre ASCII e UTF-8, facilitando a tradução de texto ASCII legado para o Unicode.
(https://pt.wikipedia.org/wiki/Unicode)

UNICODE (2)

Bits	Byte 1	Byte 2	Byte 3	Byte 4	Byte 5	Byte 6
7	0ddddddd					
11	110ddddd	10dddddd				
16	1110dddd	10dddddd	10dddddd			
21	11110ddd	10dddddd	10dddddd	10dddddd		
26	111110dd	10dddddd	10dddddd	10dddddd	10dddddd	
31	1111110x	10dddddd	10dddddd	10dddddd	10dddddd	10dddddd

- 0 a 127 – Código ASCII (1byte)
- Primeiro bit marcado com 1 indica que existem outros bytes.
- Bits "d" indicam bits de dados.
- Outros bytes iniciam com "10" ajudando na sincronização e correção de erro.

BIT DE PARIDADE

- Consiste em adicionar um bit extra ao conjunto de bits do código, podendo ser 0 ou 1. Existem dois métodos:

- **Paridade par**
  - O total de bits 1 do código, incluindo o bit de paridade, deve ser par.
  - Suponha que o conjunto de bits seja 1000011 (caractere C do código ASCII). Neste caso o bit de paridade será 1. A sequência transmitida será **1**1000011
  - Se o número de bits já for par, então adiciona-se o bit 0 de paridade. Ex.: Se o código for 1000001 a sequência transmitida será **0**1000001

- **Paridade ímpar**
  - Idêntico, porém o total de bits 1 do código, incluindo o bit de paridade, deve ser ímpar.

BIT DE PARIDADE

- Para se transmitir a sequência de caracteres “HELLO” usando-se a tabela ASCII e bits de paridade par, qual será a sequência de bits a ser transmitida?

H–	0	1	0	0	1	0	0	0
E–	1	1	0	0	0	1	0	1
L–	1	1	0	0	1	1	0	0
L–	1	1	0	0	1	1	0	0
O–	1	1	0	0	1	1	1	1

Bits de paridade par anexados

UNIDADES BINÁRIAS

Os computadores utilizam como unidade básica de medida dos dados o bit, tanto para guardar dados como para endereçá-los dinamicamente. Dessa forma, cada posição de memória pode ser indicada por um número que é, no máximo, uma potência de dois exata menos um (veja sistema binário) e, no mínimo zero, sendo que a quantidade total de endereços é uma potência de dois.

Devido à necessidade de indicar a quantidade de memória presente nos computadores, iniciou-se o hábito de utilizar os prefixos SI tradicionais (k,M,T) para indicar quantidades como Quilobyte (KB) e Megabyte (MB), porém, com a suposição que, nesses casos, um K equivale a 1024 (2^{10} = 1024) e 1 M equivale a 2^{20} = 1024*1024, e por assim em diante, o que é errado, dado que o prefixo mega corresponde a 1.000.000 de unidades.

Tendo em vista resolver definitivamente esse problema, outros prefixos foram criados, os prefixos binários.

Múltiplos do byte					
Prefixo binário (IEC)			Prefixo do Sistema Internacional		
Nome	Símbolo	Múltiplo	Nome	Símbolo	Múltiplo
byte	B	2^0	byte	B	10^0
kibibyte	KiB	2^{10}	kilobyte	kB	10^3
mebibyte	MiB	2^{20}	megabyte	MB	10^6
gibibyte	GiB	2^{30}	gigabyte	GB	10^9
tebibyte	TiB	2^{40}	terabyte	TB	10^{12}
pebibyte	PiB	2^{50}	petabyte	PB	10^{15}
exbibyte	EiB	2^{60}	exabyte	EB	10^{18}
zebibyte	ZiB	2^{70}	zettabyte	ZB	10^{21}
yobibyte	YiB	2^{80}	yottabyte	YB	10^{24}