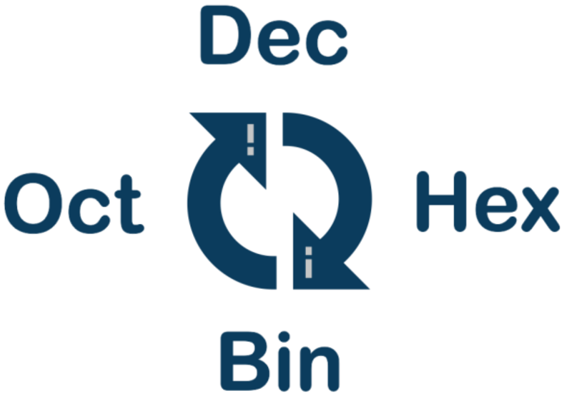
Bases Númericas

Osmar de Oliveira Braz Junior



Objetivos

- Apresentar o sistema de numeração binário;
- Como representar números binários;
- Como converter números binários para decimal, octal e hexadecimal;
- Como realizar algumas operações matemáticas com números binários considerando seus complementos.

Sistema de Numeração

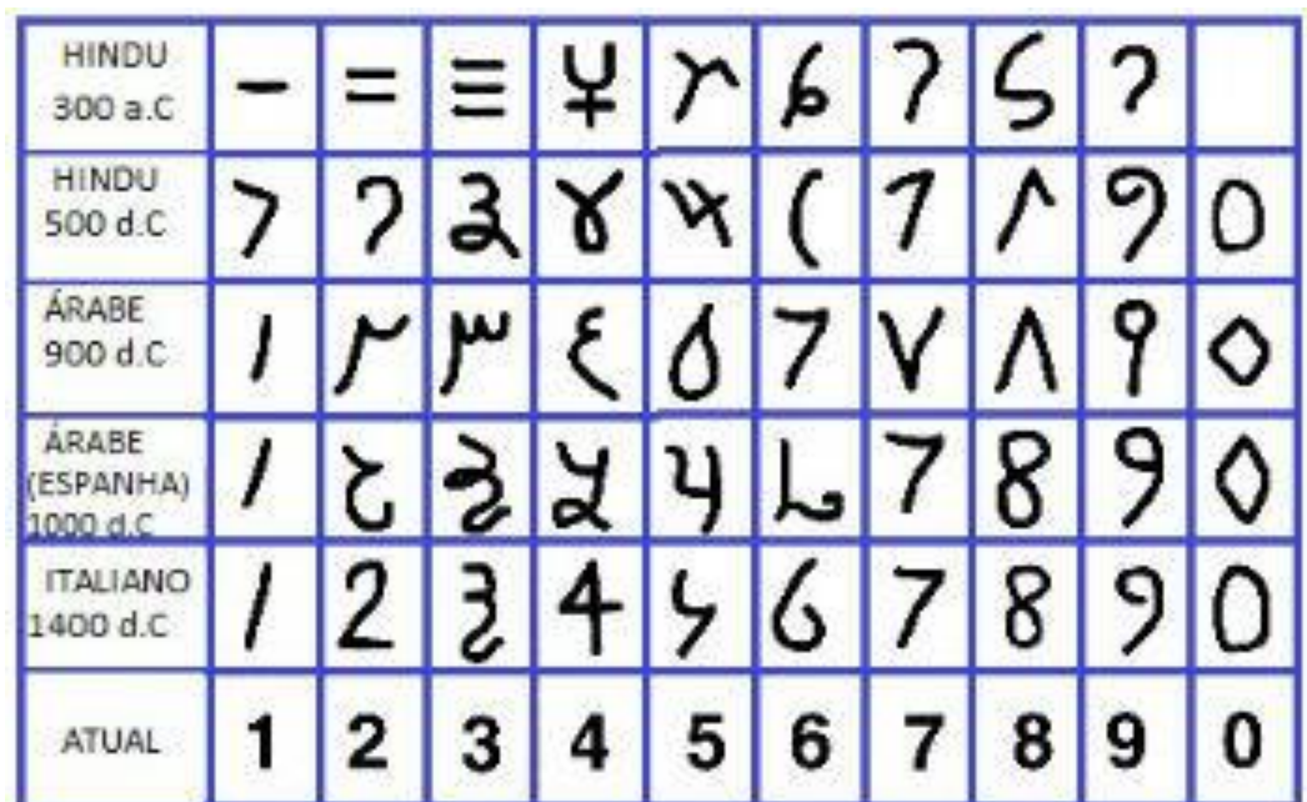
- Há milhares de anos o modo de vida era muito diferente do atual. Os homens primitivos não tinham necessidade de **contar**. Eles não compravam, não vendiam, portanto não usavam dinheiro.
- Com o passar dos anos, os costumes foram mudando e o homem passou a cultivar a terra, a criar animais, a construir casas e a comercializar. Com isso, surgiu a necessidade de **contar**.
- A vida foi tornando-se cada vez mais **complexa**. Surgiram as primeiras aldeias que, lentamente, foram crescendo, tornando-se cidades. Algumas cidades se desenvolveram, dando origem às grandes civilizações.
- Com o progresso e o alto grau de organização das antigas civilizações, a necessidade de aprimorar os processos de **contagem** e seus **registros** tornou-se fundamental.
- Foram criados, então, símbolos e regras originando assim os diferentes **sistemas de numeração**.

Sistema de Numeração

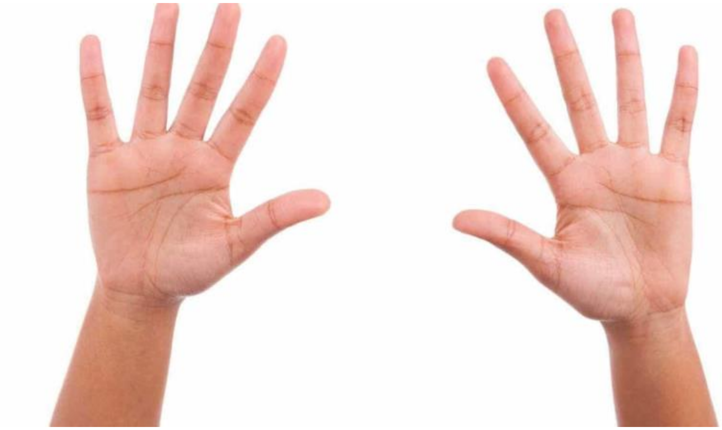
- Números são conceitos que nos são familiares desde a infância.
- Temos os números naturais, os números inteiros, os números reais, e outros.
- No dia a dia, denotamos números usando combinações de dígitos de um conjunto de 10 símbolos: 0, 1, 2, 3, 4, 5, 6, 7, 8, e 9 conhecido como sistema decimal
  - Octal : 0,1,2,3,4,5,6,7
  - Hexadecimal:0,1,2,3,4,5,6,7,8,9,A,B,C,D,E

Sistema de Numeração

- **Um sistema de numeração representa números de uma forma coerente, dando a cada um deles uma única representação.**

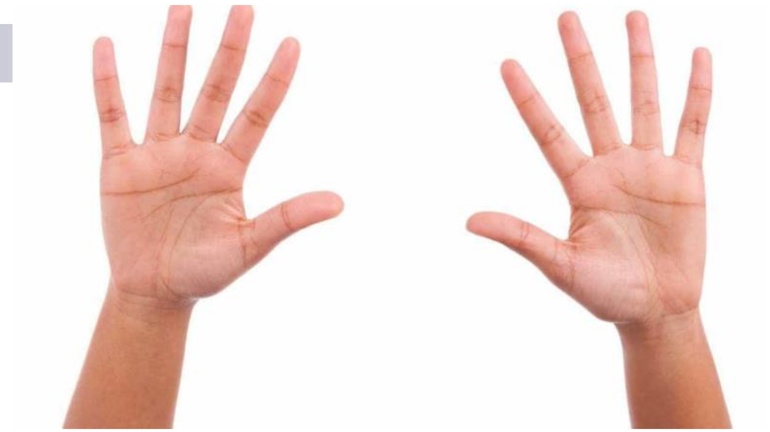


Sistema de Numeração Decimal



- **Características**
  - 1) O sistema é **decimal**, isto é, funciona com agrupamentos de dez. Esse número dez é chamado de base do sistema;
  - 2) O sistema é **posicional**, isto é, o valor de um algarismo é determinado pela posição que ocupa no numeral;
  - 3) O sistema é **multiplicativo**, isto é, em um numeral cada algarismo representa um número que é múltiplo de um potência da base dez.
  - 4) O sistema é **aditivo**, isto é, o valor do numeral é dado pela soma dos valores individuais de cada símbolo de acordo com a regra anterior

Sistema de Numeração Decimal

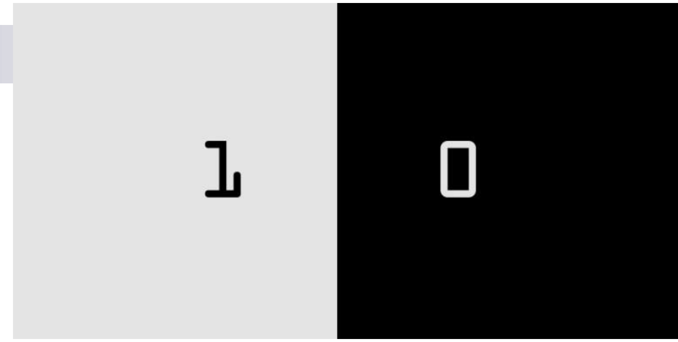


- **Por exemplo: 1234**

Classe dos milhares			Classe de unidades simples		
		1	**2**	**3**	**4**
6ª Ordem	5ª Ordem	4ª Ordem	3ª Ordem	2ª Ordem	1ª Ordem
Centenas de Milhar	Dezenas de Milhar	Unidades de Milhar	Centenas	Dezenas	Unidades
100.000	10.000	1.000	100	10	1

- **Notação polinomial**
  - $1 \times 10^3 + 2 \times 10^2 + 3 \times 10^1 + 4 \times 10^0$
  - $1 \times 1000 + 2 \times 100 + 3 \times 10 + 4 \times 1$

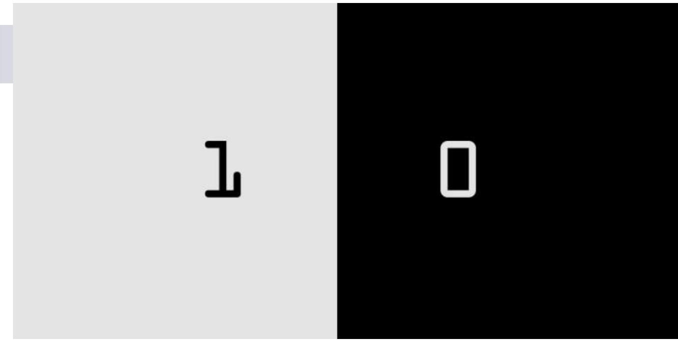
Sistema de Numeração Binário



- O computador, como ferramenta de processamento, transforma os dados de entrada em **sinais elétricos**.
- Cada sinal elétrico é chamado de **Bit** (*Binary digit*) ou **dígito binário**.

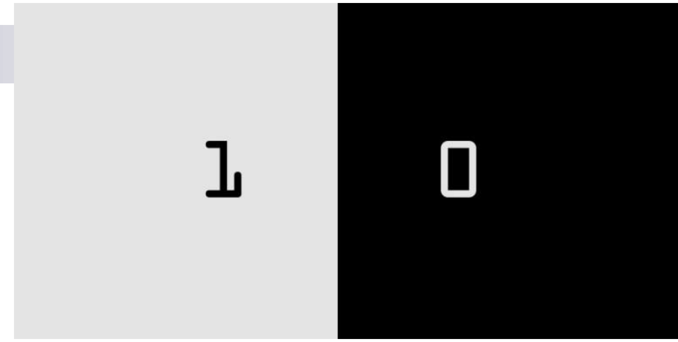
Quem inventou a palavra foi um engenheiro belga, Claude Shannon, em sua obra ***Teoria Matemática da Computação***, de 1948. Nela, Shannon descrevia um bit como sendo uma unidade de informação.

Sistema de Numeração Binário



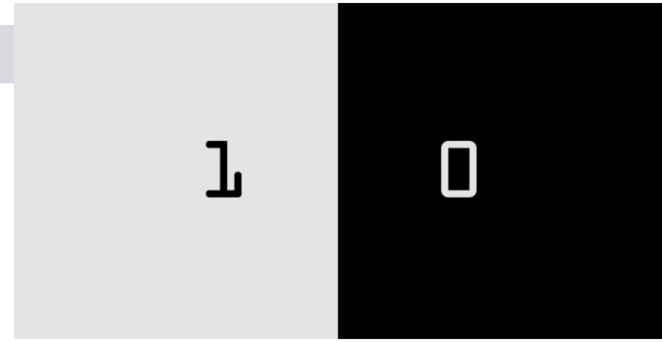
- O sistema **binário** ou de **base 2** é um sistema de numeração posicional como o decimal em que todas as quantidades se representam com base em dois números, ou seja, zero e um (**0** e **1**).
- Os computadores trabalham internamente com **dois níveis de tensão**, sendo o seu sistema de numeração natural.

Sistema de Numeração Binário



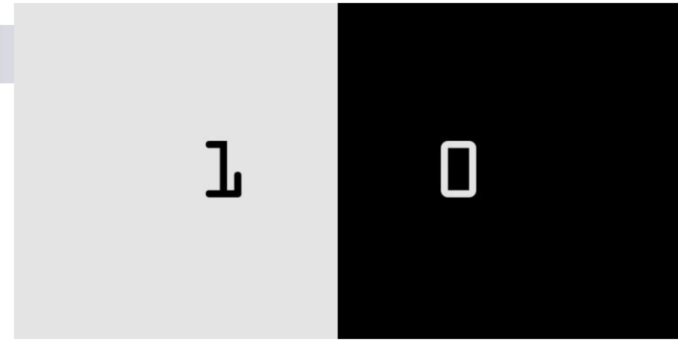
- Como dentro da máquina houvesse um **sistema de tráfego** com duas lâmpadas: a informação entra e encontra a lâmpada 1, segue em frente ate a lâmpada seguinte.
- Se encontra com a lâmpada 0, muda de direção.
- São **bilhões** de informações repetindo essas manobras a cada milésimo de segundo sequencialmente.

Sistema de Numeração Binário

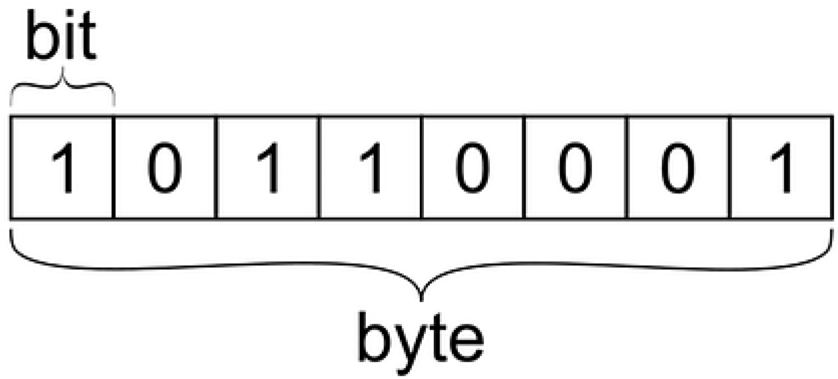


- Computadores são máquinas finitas no sentido de serem capazes de armazenar apenas um número finito de bits.
- Por esta razão, não conseguimos representar nos computadores uma quantidade infinita de dígitos.
- Computadores tipicamente utilizam blocos de ***bits*** de tamanho fixo ***n***, que chamaremos de ***palavras(words)***, para armazenar uma sequência de ***n*** dígito binários.

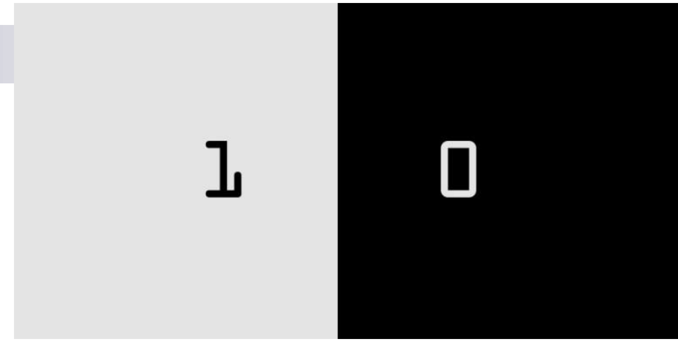
Bit e Byte



- 8 (oito) **bits** compõem um **byte**, uma unidade completa de informação.
- Os **bits** são geralmente usados como medida de velocidade na transmissão de dados, enquanto os **bytes** são normalmente associados à capacidade de armazenamento de dados (um disco rígido com memória de 500 gigabytes).



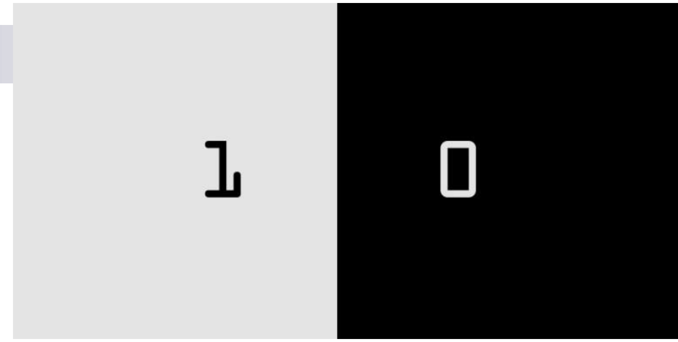
Bit e Byte



- Se um computador é de **64** bits, isso significa que ele consegue processar **64** unidades de informação ao mesmo tempo e, portanto, e mais rápido que um de **32** bits.

- 8 => 16 => 32 => 64 => 128

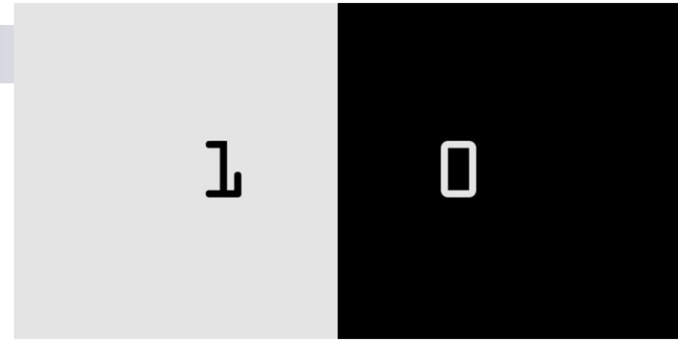
Bit e Byte



- Num sistema binário usam-se só dois dígitos (o **0** e o **1**) para representar qualquer numero.
- Comparado com o sistema de base decimal, a relação e a seguinte:

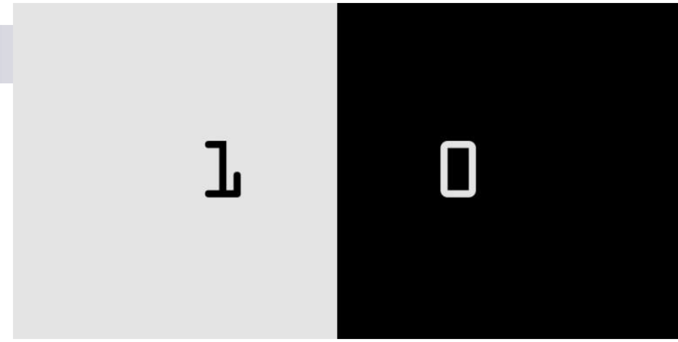
Sistema decimal									
1	2	3	4	5	6	7	8	9	10
Sistema binário									
0001	0010	0011	0100	0101	0110	0111	1000	1001	1010

Bit e Byte

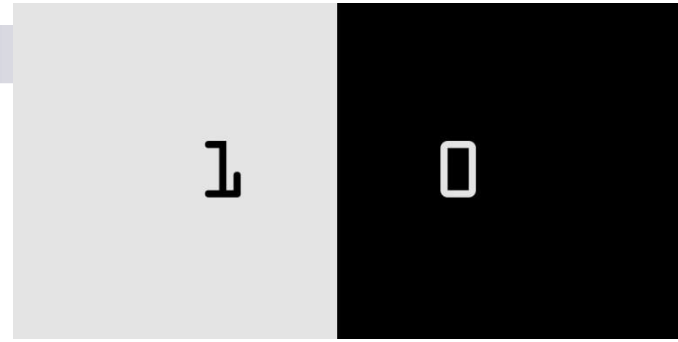


- Os **bits** não servem apenas para representar números, mas para qualquer coisa que precise ser informada a um computador.
- Podem representar uma **letra** ou uma **vírgula**, até a cor que iremos usar.
- Cada uma dessas informações é transformada em um **código binário** e interpretada pelo sistema.

Bit e Byte



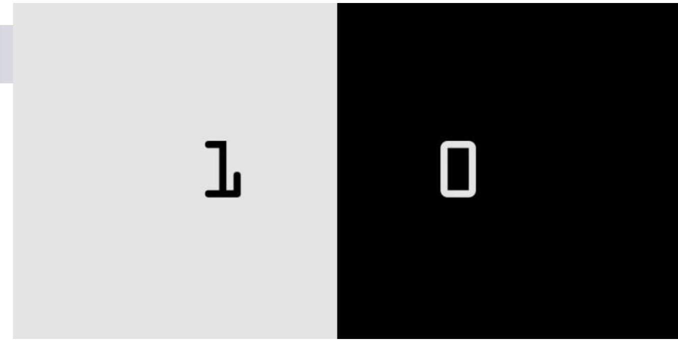
- Tal qual no sistema **numérico decimal**, cada posição de “bit” (dígito) de um número binário tem um peso particular, o qual determina a **magnitude** daquele número.
- O peso de cada posição é determinado por alguma **potência** da base do sistema numérico.
- Para calcular o valor total do número, considere os “**bits**” específicos e os pesos de suas posições.



Conversões

- Utilizando a **notação polinomial** é possível fazer cálculos de conversão de qualquer base.
  - Binária para Decimal
  - Octal para Decimal
  - Hexadecimal para Decimal
  - ...

Conversão Binário para Decimal



- Para determinar o **valor decimal** do número binário $(1111011)_2$, **multiplique** cada “bit” por seu peso posicional e **some** os resultados, ou seja utiliza a **notação polinomial**.

$$(1111011)_2$$

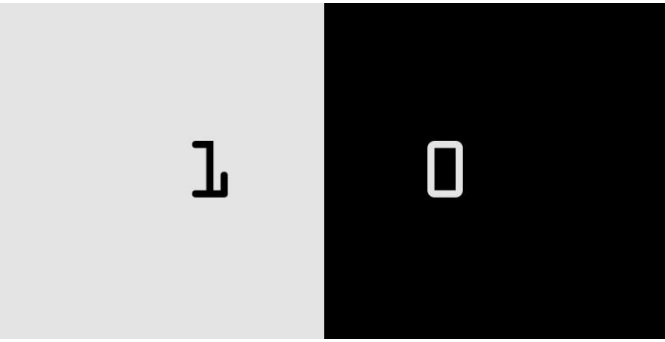
$$(\mathbf{1}x2^6)+(\mathbf{1}x2^5)+(\mathbf{1}x2^4)+(\mathbf{1}x2^3)+(0x2^2)+(\mathbf{1}x2^1)+(\mathbf{1}x2^0)$$

$$(1x64)+(1x32)+(1x16)+(1x8)+(0x4)+(1x2)+(1x1)$$

$$64 + 32 + 16 + 8 + 0 + 2 + 1$$

$$(123)_{10}$$

Conversão Octal para Decimal



- Determinar o **valor decimal** do número em **octal** $(2322)_8$.

$$(2322)_8$$

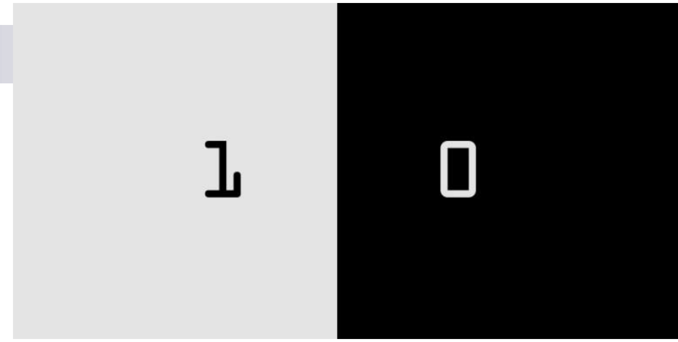
$$(2x8^3) + (3x8^2) + (2x8^1) + (2x8^0)$$

$$(2x512) + (3x64)+(2x8)+(2x1)$$

$$1024 \ + 192 + 16 + 2$$

$$(1234)_{10}$$

Conversão Hexadecimal para Decimal



- Determinar o **valor decimal** do número em **hexadecimal** $(4D2)_{16}$.

$$(4D2)_{16}$$

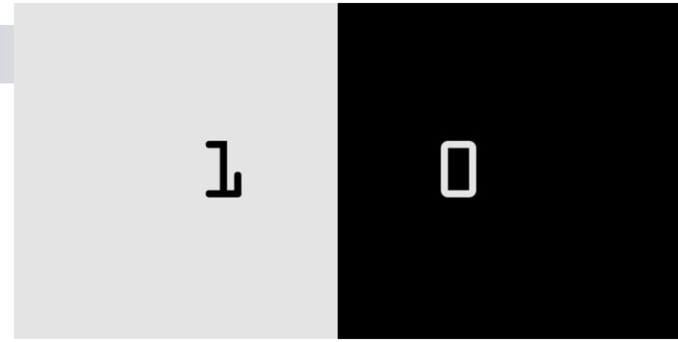
$$(4\text{x}16^2)+(13\text{x}16^1)+(2\text{x}16^0)$$

$$(4\text{x}256)+(13\text{x}16)+(2\text{x}1)$$

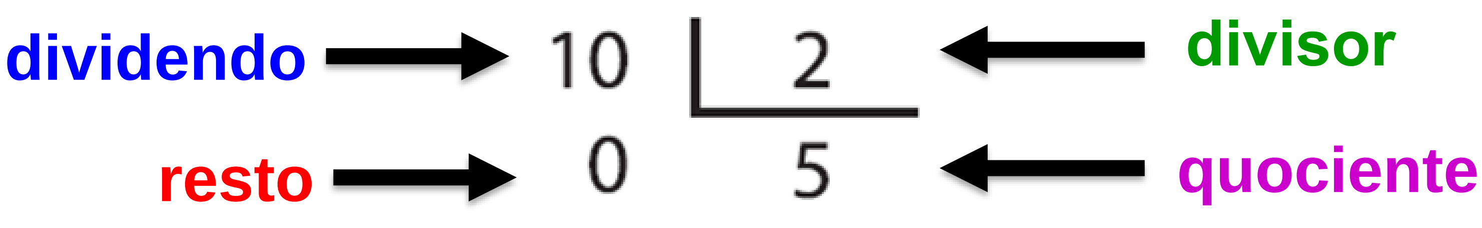
$$1024 + 208 + 2$$

$$(1234)_{10}$$

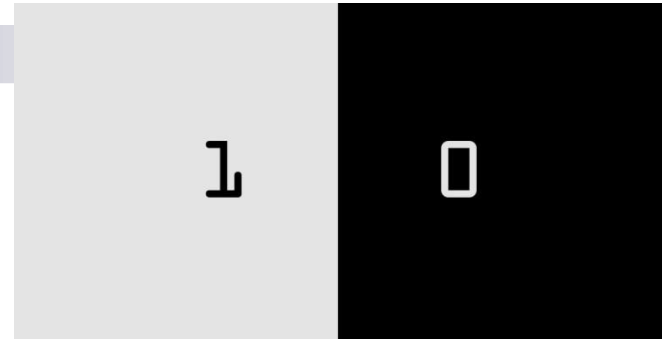
Conversão para Binário



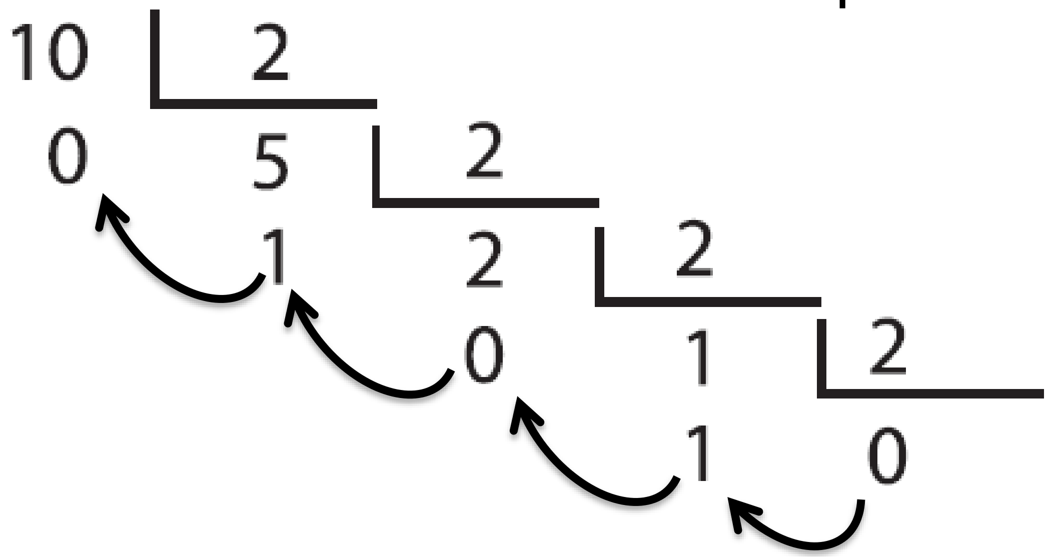
- Para converter um **número** para **binário** basta **dividir sucessivamente** pelo valor da **base(Ex. 2, 8 ou 16**.
- Se for binário divide por 2, se for octal por 8, se for hexa por 16.
- Os quocientes que vão sendo obtidos, até que o quociente de uma das divisões seja 0.



Conversão para Binário

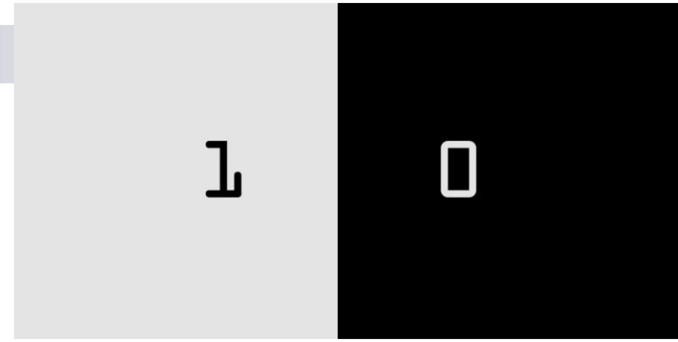


- Para converter um **número decimal** para **binário** basta **dividir sucessivamente** por 2 e os quocientes que vão sendo obtidos, até que o quociente de uma das divisões seja 0. Vamos converter o número 10 para binário.



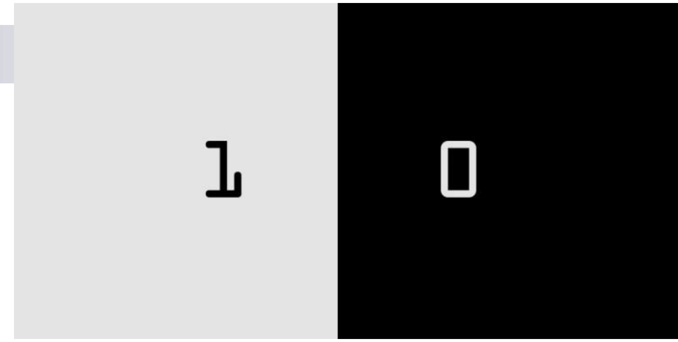
- O resultado e a sequência de baixo para cima de todos os restos obtidos $(10)_{10} = 01010 = (1010)_2$

Representação



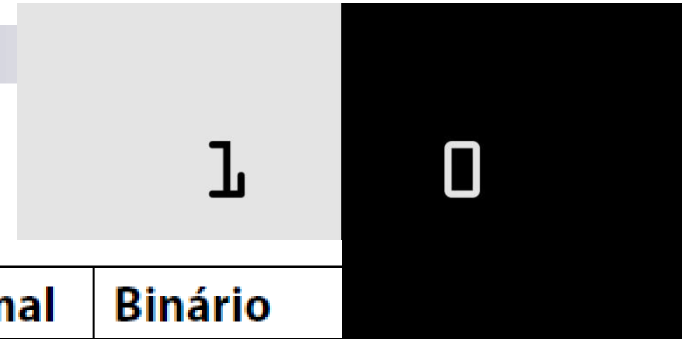
- Além de representar os **números decimais**, os **números binários** podem também representar os **números fracionários**, as **letras do alfabeto**, enfim, **todos os caracteres** que você possa encontrar no teclado do seu computador (til, acentuações, colchetes, aspas, ponto de exclamação etc.) tudo e representado através de uma **numeração binaria.**

Tabela ASCII



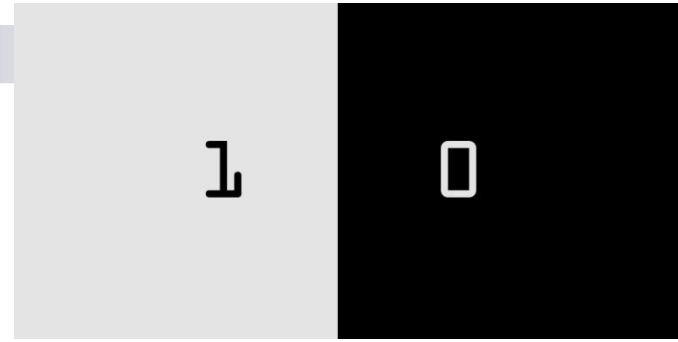
- Para organizar tais informações podemos **representar** todos os **caracteres** em uma tabela, chamada tabela **ASCII** (***American Standard Code for Information Interchange***).
- A tabela **ASCII** é usada pela maior parte da indústria de computadores para a troca de informações.
- Cada **caracter** é representado por um código de **8 bits** (um byte).
- https://desenvolvedorinteroperavel.wordpress.com/2011/09/11/tabela-ascii-completa/

Tabela ASCII



Caracter	Decimal	Hexadecimal	Binário
Espaço	32	20	0010 0000
!	33	21	0010 0001
“	34	22	0010 0010
#	35	23	0010 0011
$	36	24	0010 0100
%	37	25	0010 0101
&	38	26	0010 0110
‘	39	27	0010 0111
(	40	28	0010 1000
)	41	29	0010 1001
*	42	2A	0010 1010
+	43	2B	0010 1011
,	44	2C	0010 1100
-	45	2D	0010 1101
.	46	2E	0010 1110

Tabela ASCII



- Como seria o seu nome em binário utilizando a tabela ASCII para realizar a conversão?

- Por exemplo:

João = 0100 1010,

0110 1111,

1000 0100,

0110 1111

Exercícios

- 1) Vimos que em um sistema binário usam-se somente dois dígitos (0 e 1) para representar qualquer número. Converta os números abaixo de **decimal** para **binário**.
- a) $(20)_{10}$
- b) $(357)_{10}$

Exercícios

- 2) Agora vamos converter os números de **binário** para **decimal**.
- a) $(10100)_2$
- b) $(101100101)_2$

Exercícios

- 3) Sabemos que todos os números podem ser representados na sua forma binária, forma que o computador entende. Encontre o correspondente **binário** dos números **decimais** abaixo, mostre o cálculo realizado.
- a) $(19)_{10}$
- b) $(164)_{10}$
- c) $(129)_{10}$
- d) $(736)_{10}$
- e) $(2059)_{10}$

Exercícios

- 4) Descubra o correspondente em **decimal** dos **binários** abaixo, mostre o cálculo realizado:
- a) $(101010)_2$
- b) $(111001)_2$
- c) $(100011101)_2$
- d) $(10101010101)_2$
- e) $(1000110010)_2$

Exercícios

- 5) Você viu também que a cada 8 bits temos um byte. Para os números convertidos no exercício anterior, quantos **bytes** cada um ocupa na memória?
- a)
- b)
- c)
- d)
- e)

Operações com Números Binários

- As operações matemáticas com números binários seguem as mesmas regras da operações com números decimais.
  - Troca e destroca
    - (lembram do pegar emprestado? Do vai um)

Adição Binária

- Regras básicas:
  - `0 + 0 = 0`
  - `0 + 1 = 1`
  - `1 + 0 = 1`
  - `1 + 1 = 0 => 10` (e “troca” para o dígito de ordem superior)
  - `1 + 1 + 1 = 1 => 11` (e “troca” para o dígito de ordem superior)

$$\begin{array}{r} 0_2 \\ +\;0_2 \\ \hline 0_2 \end{array} \qquad \begin{array}{r} 0_2 \\ +\;1_2 \\ \hline 1_2 \end{array} \qquad \begin{array}{r} 1_2 \\ +\;0_2 \\ \hline 1_2 \end{array} \qquad \begin{array}{r} 1_2 \\ +\;1_2 \\ \hline 10_2 \end{array} \longrightarrow$$

troca/ vai um
carry

$$\begin{array}{rr} 1 & 1_2 \\ + & 1_2 \\ \hline 1 & 0_2 \end{array}$$

$$\begin{array}{r} 1_2 \\ 1_2 \\ +\;1_2 \\ \hline 11_2 \end{array} \longrightarrow$$

troca/ vai um
carry

$$\begin{array}{rr} & 1_2 \\ 1 & 1_2 \\ + & 1_2 \\ \hline 1 & 1_2 \end{array}$$

Adição Binária

- Ex. sem sinal:
- `101 + 011`

troca/ vai um
carry

$$
\begin{array}{rr}
 & 1\,1\,1 \\
 & (1\,0\,1)_2 \\
+ & (0\,1\,1)_2 \\
\hline
 & (1\,0\,0\,0)_2
\end{array}
$$

- 5 + 3 = 8

Multiplicação Binária

- Regras básicas:
  - **0 * 0 = 0**
  - **0 * 1 = 0**
  - **1 * 0 = 0**
  - **1 * 1 = 1**

- Mesmo método que o sistema decimal:
  - Deslocamentos e adições.
- Número maior deve ser colocado acima do menor

Multiplicação Binária

- Ex.:
- 101 * 011

$$
\begin{array}{r}
(101)_2 \\
* \quad (011)_2 \\
\hline
(101)_2 \\
(101)_2 \\
+(000)_2 \\
\hline
(01111)_2
\end{array}
$$

- 5 * 3 = 15

Subtração Binária

- Regras básicas:
  - `0 - 0 = 0`
  - `0 - 1 = 1`

    ("destroca" do dígito de ordem superior)
  - `1 - 0 = 0`
  - `1 - 1 = 0`

destroca/ pega emprestado

borrow

$$\begin{array}{r} 0_2 \\ -\,0_2 \\ \hline 0_2 \end{array} \qquad \begin{array}{r} 0_2 \\ -\,1_2 \\ \hline 1_2 \end{array} \qquad \begin{array}{r} 1_2 \\ -\,0_2 \\ \hline 1_2 \end{array} \qquad \begin{array}{r} 1_2 \\ -\,1_2 \\ \hline 0_2 \end{array} \qquad \begin{array}{r} {}^{1}0_2 \\ -\,1_2 \\ \hline 1_2 \end{array}$$

Subtração Binária

- Ex.(sem sinal):
- `101 - 011`

$$
\begin{array}{r}
0\ \ 1 \\
(101)_2 \\
-\ (011)_2 \\
\hline
(010)_2
\end{array}
$$

- 5 - 3 = 2

Divisão Binária

- Mesmo método que o sistema de numeração decimal:
  - Deslocamentos e subtrações.

Divisão Binária

- Ex.:
- `101010 / 110`

$$
\begin{array}{rl}
(101010)_2 & \underline{|\,(110)_2} \\
\underline{-110} & (111)_2 \\
1001 & \\
\underline{-110} & \\
0110 & \\
\underline{-110} & \\
000 &
\end{array}
$$

- `42 / 6 = 7`

Números com Sinal

- Representação de números com sinal usa o **Sistema sinal-magnitude.**

Bit de Sinal

$+(42)_{10}$

0	1	0	1	0	1	0

0 => +

Sinal

Magnitude

1 => -

1	1	0	1	0	1	0

$-(42)_{10}$

Números com Sinal

- No **sistema sinal-magnitude** o bit **mais significativo** é utilizado para indicar o sinal, e os demais bits representam a **magnitude** do número.
- O bit de sinal igual a **0** significa que o **valor é positivo** e o bit de sinal igual a **1** significa que o **valor é negativo**.
- Assim, se consideramos **n = 4 bits**, a representação:
  - **0101** corresponde ao número 5 (positivo)
  - **1101** corresponde ao número -5 (negativo)
- Um caso especial de representação ocorre com o zero.
  - **0000** corresponde ao número 0
  - **1000** corresponde ao número -0

Complemento de 1

- Essa interpretação, o bit mais **significativo** é também utilizado para indicar o sinal. Porém, a interpretação dos demais bits depende do bit de sinal:
  - se o bit de sinal é 0, então o número é **positivo** e a magnitude é o valor correspondente aos demais bits;
  - se o bit de sinal é 1, então o número é **negativo** e a magnitude é o valor correspondente ao **complemento** dos demais bits;
- Ex.
  - 0101 corresponde ao número 5 (positivo)
  - 1101 corresponde ao número -2 (negativo),

$$\text{pois } (\overline{101})_2 = (010)_2 = (2)_{10}$$

O complemento de 0 é 1 e o de 1 é 0.

Exercícios

- 6) Encontre as representações **binárias** para os números **decimais** abaixo. Mostre o cálculo realizado e caso o número seja negativo utilize o **complemento de 1** para gerar a representação binária.
- a) $(23)_{10}$
- b) $(-23)1_{10}$
- c) $(33)_{10}$
- d) $(-33)_{10}$

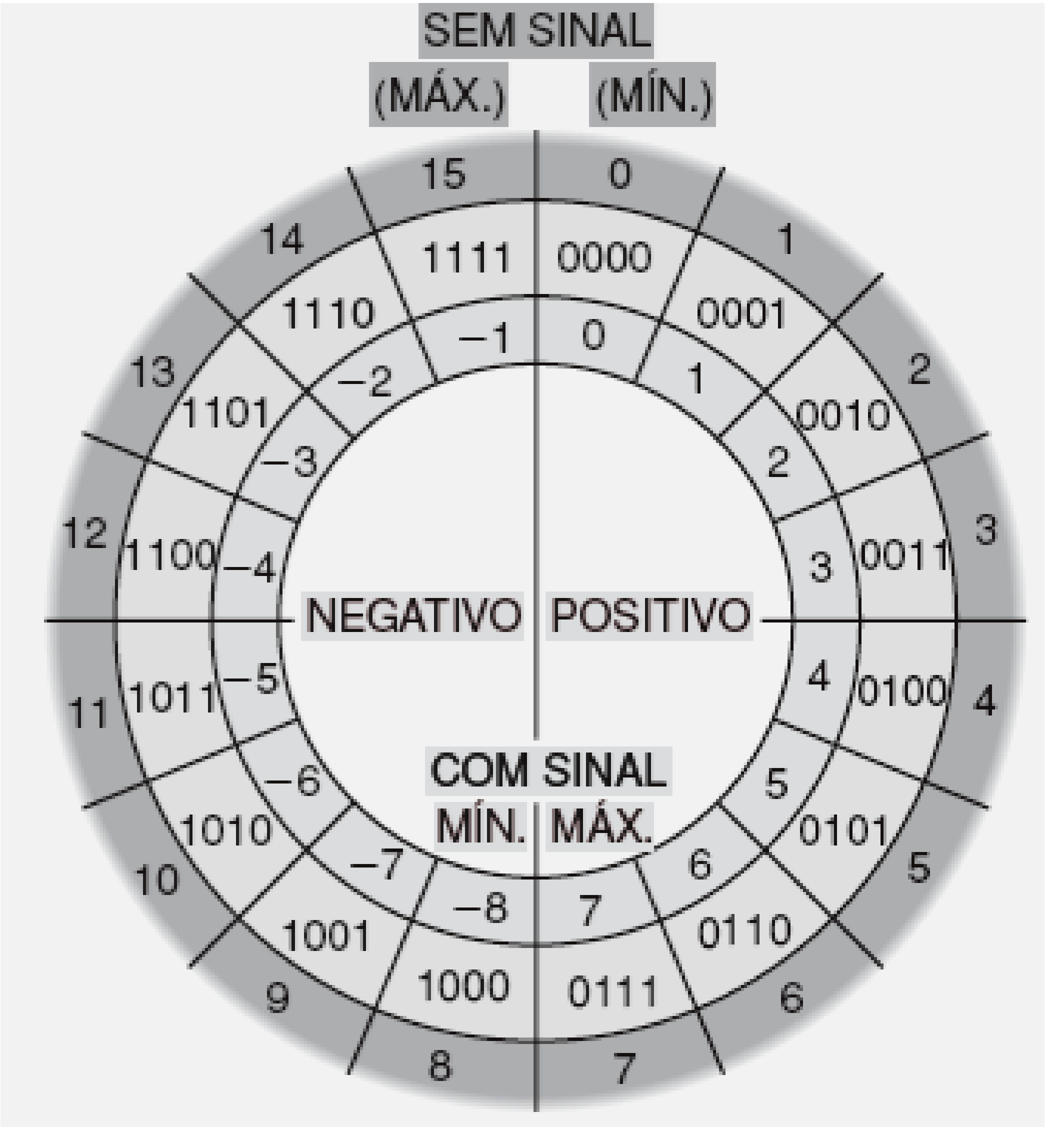
Complemento de 2

- Essa interpretação é similar ao complemento de 1, porém quando o bit de sinal é 1, a magnitude é dada pelo valor correspondente ao complemento dos demais bits, **mais 1**:
- Ex.
  - 0101 corresponde ao número 5 (positivo)
  - 1101 corresponde ao número -3 (negativo),

$$\text{pois } \overline{(101)}_2 = (010)_2 + 1 = (011)_2 = (3)_{10}$$

Representação binária	Valor em decimal de acordo com as interpretações			
$d_3\, d_2\, d_1\, d_0$	Sem sinal	Com sinal	Complemento de 1	Complemento de 2
0000	0	0	0	0
0001	1	1	1	1
0010	2	2	2	2
0011	3	4	3	3
0100	4	4	4	4
0101	5	5	5	5
...	...	...	...	...
1011	11	-3	-4	-5
1100	12	-4	-3	-4
1101	13	-5	-2	-3
1110	14	-6	-1	-2
1111	15	-7	-0	-1

Quantos números com sinal podem ser representados utilizando 4 bits?”



Exercícios

- 7) Encontre as representações binárias para os números abaixo. Mostre o cálculo realizado e caso o número seja negativo utilize o complemento de 2 para gerar a representação binária.
- a) $(23)_{10}$
- b) $(-23)_{10}$
- c) $(33)_{10}$
- d) $(-33)_{10}$

Adição Binária

- Ex. **com sinal iguais**(complemento de 2):
- $0101_2 + 0010_2$

troca/ vai um
carry

$$
\begin{array}{r}
\;\;\;\;\;\;\;\;\;1\\
(0101)_2\\
+\;(0010)_2\\
\hline
(0111)_2
\end{array}
$$

- 5 + 2 = 7

Adição Binária

- Ex. **com sinais diferentes**(complemento de 2):
- $0101_2 + 1001_2$

troca/ vai um
carry

$$
\begin{array}{r}
\ \ {}^{1} \\
(0101)_2 \\
+\ (1001)_2 \\
\hline
(1\underbrace{110}_{})_2
\end{array}
$$

- $\overline{110_2} + 1 = 001 + 1 = 010 = 2_{10} = -2_{10}$
- $5 + (-7) = -2$

Subtração Binária

- Podemos calcular subtrações entre **números binários** de forma similar ao realizado com **números decimais**.
- Por outro lado, note que uma subtração pode ser expressa na forma de uma **adição**:
- Desta forma calcular A - B é equivalente a calcular A + (-B).

Subtração em complemento de dois

- Ex: $5_{10} - 3_{10} = 2_{10}$ (utilização de 4 bits)

$5_{10} = 0101_2$

$3_{10} = 0011_2$

$-3_{10} = 1100_2 + 1$

$-3_{10} = 1101_2$

troca/ vai um *carry*

$$
\begin{array}{r}
11 \\
0101_2 \\
+1101_2 \\
\hline
(1)0010_2
\end{array}
$$

Subtração em complemento de dois

- Ex: $5_{10} - 3_{10} = 2_{10}$ (utilização de 4 bits)

$$5_{10}=0101_2$$

$$3_{10}=0011_2$$

$$-3_{10}=1100_2+1$$

$$-3_{10}=1101_2$$

$$\begin{array}{r} 0101_2 \\ +1101_2 \\ \hline (1)\,0010_2 \end{array}$$

- Notar: o bit mais significativo (decorrente do último troca/“vai um”) deve ser desprezado.

Interpretação

- A ULA (unidade lógico-aritmética responsável por por executar as operações lógicas e aritméticas**) não precisa ter “ciência"** sobre qual interpretação está sendo considerada.
- Ela simplesmente executa a adição de dois números representados em ***n*** bits, não se importando com a interpretação dos mesmos.
- Cabe às **unidades de controle(UC)** ou ao software fazer a interpretação correta.

Exercícios

- 8) Realize as operações matemáticas com os números binários. Mostre o cálculo realizado, utilize o complemento de 2 caso seja uma subtração.
- a) $(1010)_2 + (101)_2$
- b) $(1010)_2 – (101)_2$
- c) $(1010)_2 * (101)_2$
- d) $(1010)_2 / (101)_2$
- e) $(11)_{10} - (9)_{10}$
- f) $(9)_{10} - (11)_{10}$

Conclusão

- Seres humanos diferenciam os símbolos do sistema de numeração decimal, mas um computador só entende ligado e desligado, portanto somente 0 e 1.
- Apesar de parecer mais complexo as operações com números binários, ela é muito mais simples para um computador que somente reconhece 0 e 1, ou seja ligado e desligado.
- As operações matemáticas com números binários são iguais as do números decimais, mas somente operando 0 e 1.

Referências

- WEBER, Raul Fernando. Fundamentos de arquitetura de computadores. 4. ed. Porto Alegre: Bookman, 2012. E-book. Disponível em: https://integrada.minhabiblioteca.com.br/books/9788540701434

- STALLINGS, William. Arquitetura e organização de computadores. 8.ed. São Paulo: Pearson, 2010. E-book. Disponível em: https://plataforma.bvirtual.com.br/Leitor/Publicacao/459/epub/0

- HOGLUND, Greg. Como quebrar códigos: a arte de explorar (e proteger) software. São Paulo: Pearson, 2006. E-book. Disponível em: https://plataforma.bvirtual.com.br/Leitor/Publicacao/179934/epub/0

- Lógica de programação I : livro didático / Carlos Fernando Martins, Elton João Gubert, Mario Gerson Miranda Magno Júnior, Patrícia Gerent Petry ; design instrucional Daniela Erani Monteiro Will, Flavia Lumi Matuzawa, Leandro Kingeski Pacheco ; [Assistente acadêmico Luiz Henrique Queriquelli, Silvana Souza da Cruz]. – 5. ed. rev. – Palhoça : UnisulVirtual, 2008.

- https://desenvolvedorinteroperavel.wordpress.com/2011/09/11/tabela-ascii-completa/

Fim