Bases Numéricas e Sinais Digitais

Coordenadoria de Informática
Sistemas de Informação
Profs. Vitor Faiçal Campana
Francisco Rapchan
Material cedido pelo Prof. Flávio Giraldeli

2019/2

O Computador e a Informação

Troca de informações entre os seres humanos e o universo a sua volta

Vs.

Troca de informações entre os dispositivos eletrônicos (câmeras, computadores, etc...)

O Computador e a Informação

- Diferença principal: A **forma** em que a informação está representada.
- Informações para o ser humano: **analógicas**
  - Som, imagens, ...
- Informações analógicas podem teoricamente assumir qualquer valor numérico real de -∞ a +∞.
  - Ou seja, infinitos valores.
- Na verdade, lidamos sempre com valores aproximados.

O Computador e a Informação

- Computadores não podem lidar com informações analógicas!
  - Complexidade intratável.
  - Pouco confiável.
- Solução? Uma "nova" forma de representar a informação, o que implicou diretamente em um novo sistema numérico.
- Estamos falando dos ***sistemas digitais*** e a representação numérica de base 2 (***binária***).

Sinais Digitais e Números Binários

- Um sistema digital é um sistema no qual os sinais tem um ***número finito de valores discretos*** (bem definidos, enumeráveis).
  - Exemplo 1: uma balança *digital* mede o peso através de sinais discretos que indicam a massa; por outro lado, uma balança *analógica* mede o peso através de um sinal contínuo correspondente a posição de um ponteiro sobre uma escala.
  - Exemplo 2: Um sintonizador digital de rádio exibe a frequência da estação, digamos 100.1 MHz (***exatamente***). Já um sintonizador analógico, mostrará uma faixa ***contínua*** de frequências e a sintonia será, na verdade, algo aproximado.

Benefícios dos sistemas digitais

- A representação digital é bem adequada tanto para processamento numérico como não-numérico (caracteres, por exemplo) de informação.
- O processamento da informação pode usar um sistema para propósitos gerais (um computador) que seja programado para uma tarefa de processamento particular (como o de imagens), eliminando a necessidade de haver um sistema diferente para cada tarefa.

Benefícios dos sistemas digitais

- O número finito de valores num sinal digital pode ser representado por um vetor (conjunto de valores) de sinais com apenas dois valores (sinais binários). Por exemplo, os dez valores de um dígito decimal podem ser representados por um vetor de quatro sinais binários (ou bits), da seguinte maneira:

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

- Esta representação permite implementações nas quais todos os sinais são binários; em consequência, os dispositivos que processam esses sinais são muito simples (fundamentalmente, apenas chaves com dois estados: aberto e fechado).

Benefícios dos sistemas digitais

- Os sinais digitais são bastante insensíveis a variações nos valores dos parâmetros dos componentes (por exemplo, temperatura de operação, ruído), de modo que pequenas variações na representação física não mudam o valor efetivo.
- Os sistemas digitais numéricos podem se tornar mais exatos simplesmente aumentando-se o número de dígitos usados na sua representação.

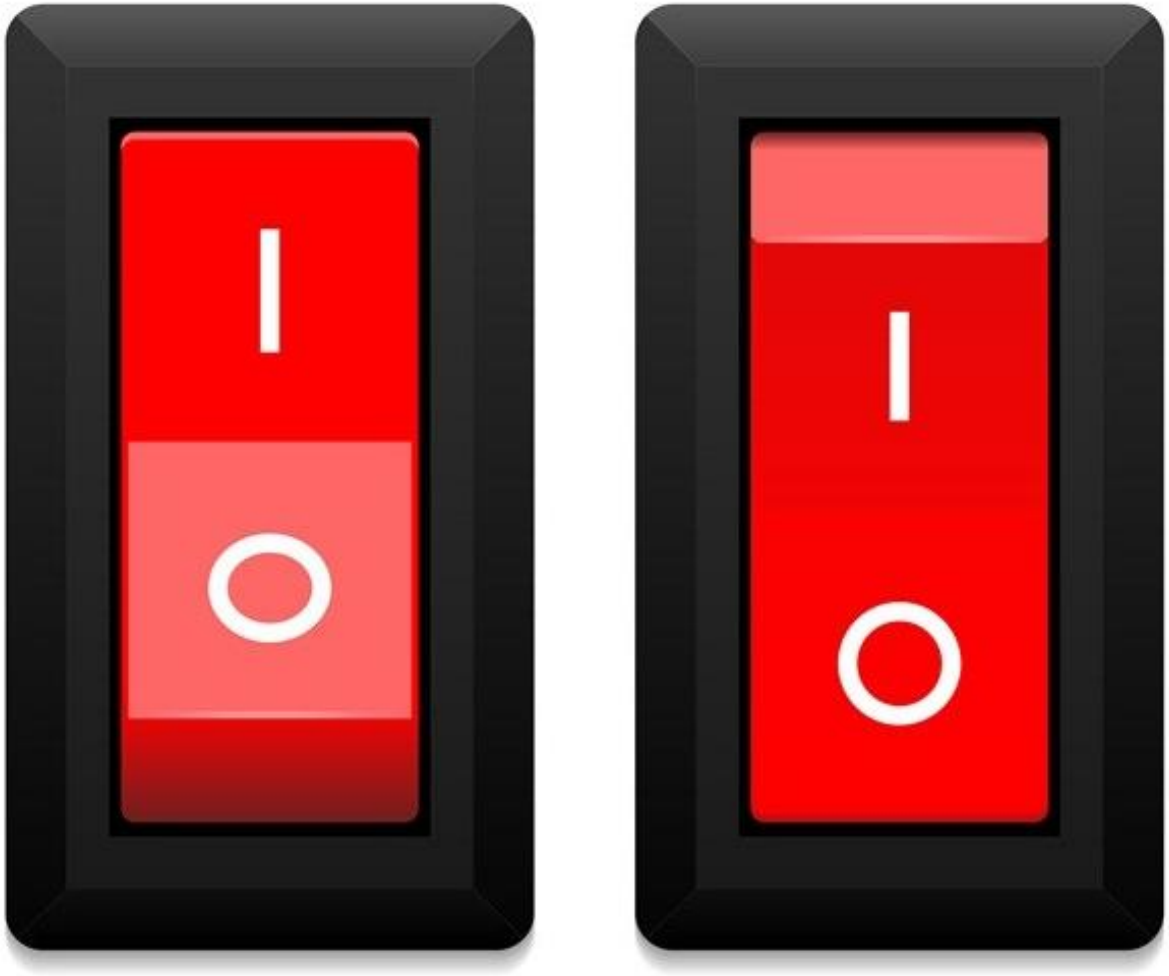
Bases Numéricas

2 (Binária)

10 (Decimal)

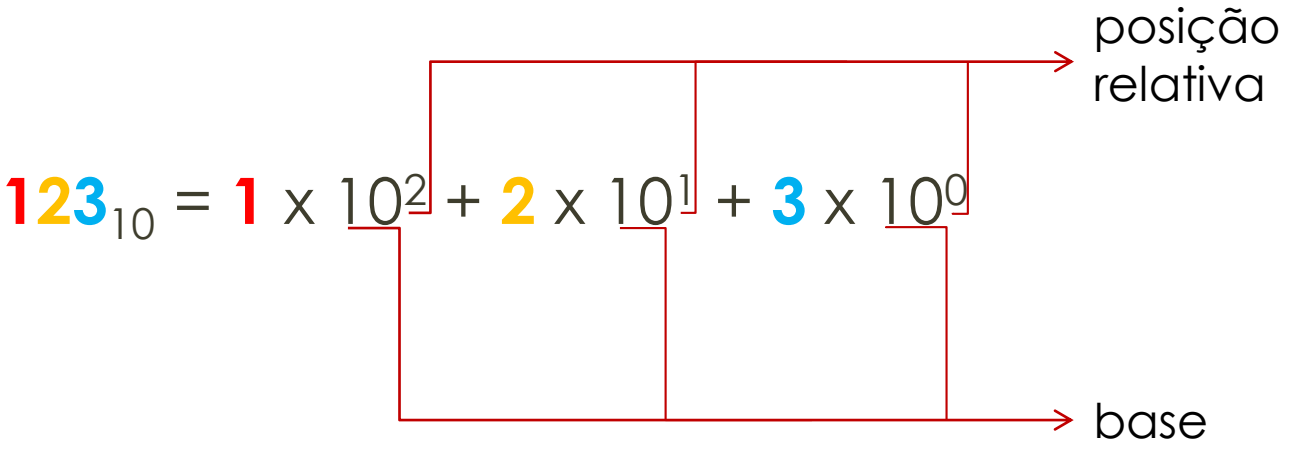
16 (Hexadecimal)

O Sistema Binário



Números Binários

- Antes de compreender melhor como funciona a representação em base binária (base 2), observe como um número qualquer é formado na base em que estamos acostumados a lidar, a base 10. Tome por exemplo o número decimal 123. Veja só como o mesmo é formado:



Números Binários

- Pense agora que ao invés de dispor de 10 algarismos diferentes, dispomos de apenas dois. Vamos chamar esses algarismos de 0 (zero) e 1 (um). De forma absolutamente análoga, pense no número 1111011 expresso na base 2. Desmembrando o mesmo temos:

$$1111011_2 = 1x2^6 + 1x2^5 + 1x2^4 + 1x2^3 + 0x2^2 + 1x2^1 + 1x2^0 = 123_{10}$$

Números Binários

- E como ficam os números não inteiros (ou seja, aqueles com "vírgula")? A ideia é acima é estendida para expoentes negativos na base 10. Observe:

$$456{,}78_{10} = 4x10^{2} + 5x10^{1} + 6x10^{0} + 7x10^{-1} + 8x10^{-2}$$

- De maneira análoga, o raciocínio acima pode ser usado em qualquer base, por exemplo, a base 2:

$$101{,}101_{2} = 1x2^{2} + 0x2^{1} + 1x2^{0} + 1x2^{-1} + 0x2^{-2} + 1x2^{-3} = 5{,}625_{10}$$

Números Binários

- Fato:
  - Na base dez, para se ***multiplicar*** um número pela base, isto é, por dez, basta deslocar a vírgula uma casa para a ***direita***. Na ***divisão*** por dez, basta deslocar a vírgula para a ***esquerda***.
  - O mesmo ocorre com qualquer base, em particular com a base dois. Para multiplicar um número por dois, basta deslocar a vírgula uma casa para a direita. Ao deslocar para a esquerda, estamos dividindo por dois.

Números Binários

- Não é muito comum descrevermos quantidades binárias em bits. Costumamos expressar os valores em bytes, que são agrupamentos formados por 8 bits.
- Outros múltiplos também existem e são expressos em relação a byte, alguns deles são:

1 Kilobyte (KB)	2^{10} bytes	1.024 bytes
1 Megabyte (MB)	2^{20} bytes	1.048.576 bytes
1 Gigabyte (GB)	2^{30} bytes	1.073.741.824 bytes
1 Terabyte (TB)	2^{40} bytes	1.099.511.627.776 bytes

Números Binários

- Importante:
  - No caso dos números binários, a faixa de valores possível de ser expressa numa dada quantidade de bits é relativamente pequena comparada aos números decimais.
    - Por exemplo, na base dez, dispondo de uma sequência formada por oito algarismos, conseguimos representar 10^8 valores diferentes, ou seja, de 0 a 99.999.999. Já na base binária, com uma sequência de oito dígitos (8 bits), conseguimos representar 2^8 valores diferentes, ou seja, 256 valores distintos (0 a 255).
  - De maneira geral, quanto menos algarismos uma base possui (***quanto menor a base***), ***menor é o intervalo representável*** para uma mesma quantidade de dígitos.

Base Hexadecimal (16)

- Uma vez compreendida a base binária e decimal, fica fácil compreender a base hexadecimal, ou base 16.
- Essa base possui não apenas 2 ou 10 algarismos, mas 16 algarismos diferentes.
- Veja a correspondência entre *decimais*, *binários* e *hexadecimais* correspondentes:

$0_{10} = 0000_2 = 0_{16}$	$1_{10} = 0001_2 = 1_{16}$	$2_{10} = 0010_2 = 2_{16}$	$3_{10} = 0011_2 = 3_{16}$
$4_{10} = 0100_2 = 4_{16}$	$5_{10} = 0101_2 = 5_{16}$	$6_{10} = 0110_2 = 6_{16}$	$7_{10} = 0111_2 = 7_{16}$
$8_{10} = 1000_2 = 8_{16}$	$9_{10} = 1001_2 = 9_{16}$	$10_{10} = 1010_2 = A_{16}$	$11_{10} = 1011_2 = B_{16}$
$12_{10} = 1100_2 = C_{16}$	$13_{10} = 1101_2 = D_{16}$	$14_{10} = 1110_2 = E_{16}$	$15_{10} = 1111_2 = F_{16}$

Base Hexadecimal (16)

- Você pode estar pensando, por que motivo eu precisaria de mais uma base?
- Resposta, com um exemplo:

Decimal	Binário	Hexadecimal
4.285.639.982	1111.1111.0111.0001.1010.1101.0010.1110	FF71AD2E

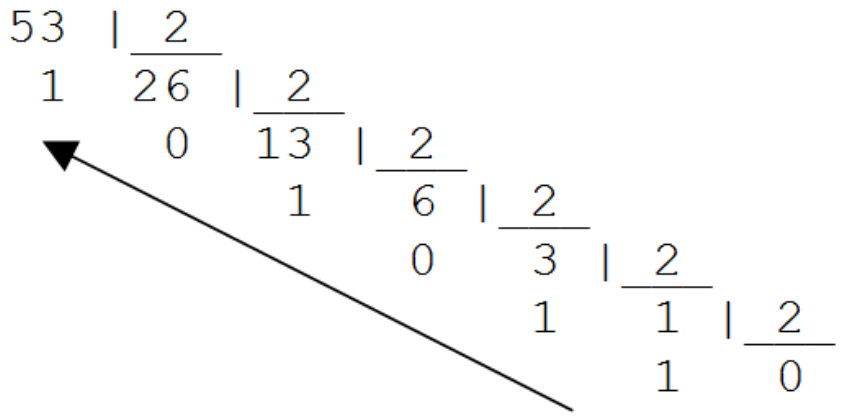
- Curiosidade:
  - Cores são muitas vezes representadas por valores numéricos de 24 bits (16.777.216 cores possíveis).
  - Imagine, perguntarmos para nosso colega que cor ele usou num determinado projeto e o ouvirmos dizer uma sequência de 24 zeros e uns!
  - Obviamente será muito mais simples dizer uma simples sequência de 6 algarismos hexadecimais, por exemplo F7FF29 que é um tom de amarelo.
  - Ou você prefere dizer: 1111 0111 1111 1111 0010 1001 ?

Conversões entre Bases Numéricas

Porque nenhuma base é ideal para todas as situações!

Decimal → Binário

- Números Inteiros
  - Forma 1: Divisões sucessivas por 2 (a base binária).



  - As divisões sucessivas acabam quando o quociente chega finalmente a zero. O número na forma binária é formado pelos restos das divisões, de baixo para cima, tal qual indicado pela seta. Assim: $53_{10} = 110101_2$

Decimal → Binário

- Números Inteiros
  - Forma 2: Método Direto.
    - Imagine um número binário qualquer, de 6 dígitos. Vamos chamá-lo de $b_5b_4b_3b_2b_1b_0$ onde cada b é um dígito, que pode ser 0 ou 1, naturalmente. O valor desse número, na base decimal é:

$$b_5b_4b_3b_2b_1b_0$$

$$=$$

$$b_5 \times \mathbf{2^5} + b_4 \times \mathbf{2^4} + b_3 \times \mathbf{2^3} + b_2 \times \mathbf{2^2} + b_1 \times \mathbf{2^1} + b_0 \times \mathbf{2^0}$$

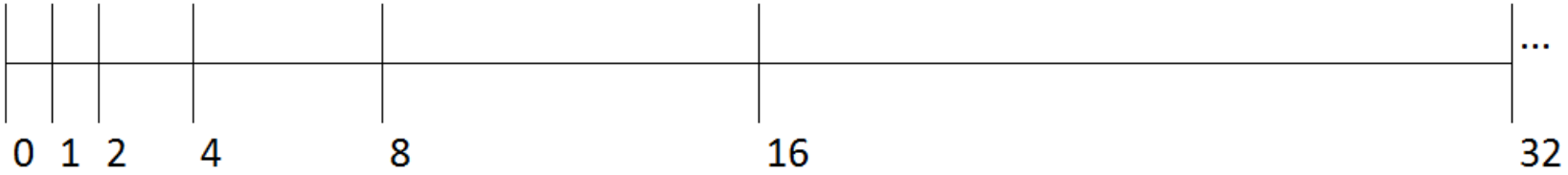
$$=$$

$$b_5 \times \mathbf{32} + b_4 \times \mathbf{16} + b_3 \times \mathbf{8} + b_2 \times \mathbf{4} + b_1 \times \mathbf{2} + b_0 \times \mathbf{1}$$

    - Que valores cada dígito binário deve assumir para que a soma dê, por exemplo, **53**? Resposta: **110101** (confira!)

Decimal → Binário

- Vamos pensar um pouco mais. Graficamente!



- Múltiplos da base 2 (números muito conhecidos na computação):
  - 1, 2, 4, 8, 16, 32, 64, 128, 256, 512, 1024, 2048, 4096...
- Dependendo da posição relativa do dígito, ele possui um valor, dentre os acima. Qualquer número inteiro pode ser representado como uma soma dessas quantidades (desde que esteja dentro do intervalo representável, claro).
- Intervalo representável, se considerados apenas os números não negativos: **0** a $\mathbf{2^{nbits} - 1}$

Decimal → Binário

- Números não inteiros:
  - Podemos tratar a parte inteira (já vista) e a parte fracionária de maneira independente. Vamos continuar com o raciocínio:

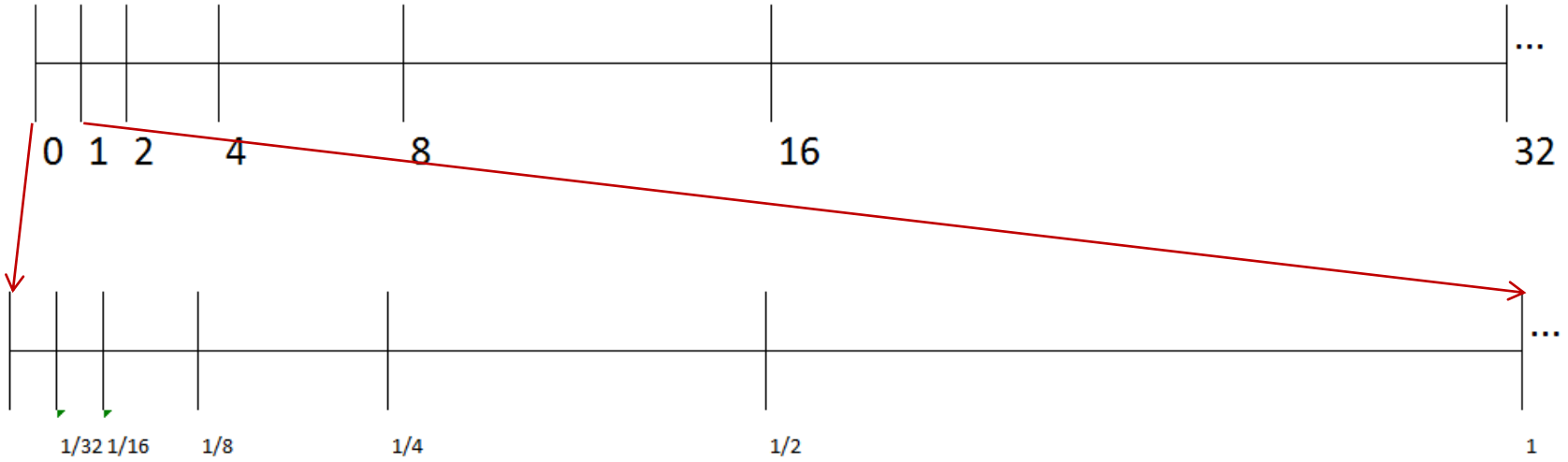
    ...
    $2^{-5} = 1/32 = 0,03125$
    $2^{-4} = 1/16 = 0,0625$
    $2^{-3} = 1/8 = 0,125$
    $2^{-2} = 1/4 = 0,25$
    $2^{-1} = 1/2 = 0,5$
    $2^{0} = 1$
    $2^{1} = 2$
    $2^{2} = 4$
    $2^{3} = 8$
    $2^{4} = 16$
    $2^{5} = 32$
    ...

Decimal → Binário

- Números não inteiros:
  - Assim, tudo o que precisamos fazer é continuar combinando os dígitos binários após a vírgula (expoentes negativos) de modo que a soma dê o valor (exato ou aproximado) desejado.
  - Ex:
    - $0{,}625_{10} = 0{,}101_2$
    - $19{,}625_{10} = 10011{,}101_2$

Decimal → Binário

- Números não inteiros:
  - E o raciocínio gráfico (reta numérica) pode ser estendido para expoentes negativos]0,1[:



Decimal → Binário

- Números não inteiros:
  - Mas, existe um método "mecânico" de fazer a conversão da parte fracionária, tal qual existe na parte inteira (divisões sucessivas por 2)? SIM!
    - Exemplo: $0{,}625_{10} = 0{,}?_2$

Residual	x 2	≥ 1 ?
0,625	1,25	Sim = 1
0,25	0,5	Não = 0
0,5	1,0	Sim = 1
0		

    - Logo, $0{,}625_{10} = 0{,}101_2$

Decimal → Binário

- Números não inteiros:
  - Desafio: Tente!
    - $0{,}8_{10} = ?_2$
    - $5{,}8_{10} = ?_2$
    - $11{,}6_{10} = ?_2$

Binário/Decimal → Hexadecimal

- A conversão de binário para hexadecimal é bastante simples. Basta agrupar os dígitos em grupos de 4, a partir da direta e completar o último grupo com zeros. O ponto, neste caso, é apenas para melhor visualização. Ex:
  - $53_{10} = 0011.0101_2 = 35_{16}$
  - $12972_{10} = 0011.0010.1010.1100_2 = 32AC_{16}$
  - $12237514_{10} = 1011.1010.1011.1010.1100.1010_2 = BABACA_{16}$
- Acima, a conversão Decimal → Hexadecimal foi feita da forma Decimal → Binário → Hexadecimal. No entanto, ela pode ser feita diretamente usando quaisquer dos dois métodos usados de Binário → Decimal (Divisões sucessivas por 16 ou de modo "Direto").

Binário/Hexadecimal → Decimal

- As conversões de hexadecimal/binário para decimal podem ser facilmente executadas a partir da própria expressão do número na base correspondente, efetuando a soma de produtos, conforme extensamente mostrado anteriormente.
- Por exemplo:
  - $110101_2 = 1x2^5 + 1x2^4 + 0x2^3 + 1x2^2 + 0x2^1 + 1x2^0 = 53_{10}$
  - $32AC_{16} = 3x16^3 + 2x16^2 + 10x16^1 + 12x16^0 = 12972_{10}$

Lógica Temporizada

Tic, Toc, Tic, Toc, Tic, ...

Lógica Temporizada

- Na comunicação digital, qualquer dado, sob a forma de bits, pode ser representado por sinais elétricos:
  - uma tensão positiva alta (“high” - geralmente no em torno de 5 volts) significando ***1*** e...
  - uma tensão baixa (“low” - próxima de zero) significando ***0***.
- O transmissor coloca o sinal no barramento, **espera um tempo** (na qual o sinal fica estável), para em seguida colocar um novo sinal.

Lógica Temporizada

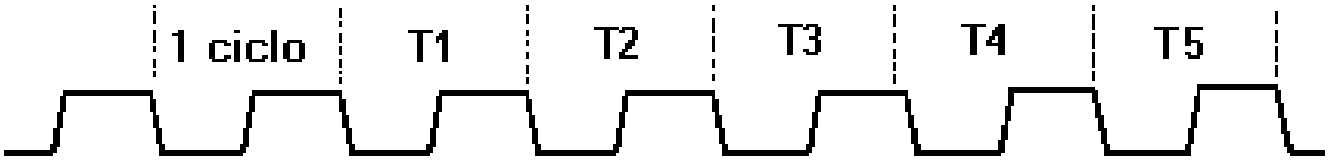
- Fica claro que, quanto menor for esse "tempo de espera", mais bits serão transmitidos por unidade de tempo (logo, maior a velocidade da transmissão).
- Essa ***base de tempo***, é um dos aspectos mais importantes do mundo da computação. Ela é dada por um sinal de sincronismo, cuja precisão é quase perfeita.
- Estamos falando do sinal de ***clock***.

Clock

- O pulso de ***clock*** nada mais é que uma *referência de tempo* para todas as atividades e permite o sincronismo das operações internas.
- O ***clock*** é um pulso alternado de sinais de tensão, gerado pelos circuitos de relógio (composto de um cristal oscilador e circuitos auxiliares).
- Cada um destes intervalos regulares de tempo é delimitado pelo início da descida do sinal, equivalendo um ciclo à excursão do sinal por um ***"low"*** e um ***"high"*** do pulso.

Clock

- O tempo do ciclo equivale ao período da oscilação. A física diz que período é o inverso da frequência. Ou seja, P = 1 / f.
- A frequência f do ***clock*** é medida em hertz. Inversamente, a duração de cada ciclo é chamada de período, definido por P=1/f (o período é o inverso da frequência).
- Por exemplo, se f = 10 Hz logo P = 1/10 = 0,1 s.



Clock

- 1 MHz (1 megahertz) = 1.000.000 ciclos/segundo.
- Sendo a frequência de um processador medida em megahertz, o período será então medido em nanosegundos, como vemos no exemplo abaixo:
  - f = 10 MHz = 10 x 10^6 Hz
  - P = 10 / 10^6 = 0,0000001 s (segundo) = 0,0001 ms (milissegundo) = 0,1 µs (microssegundo) = 100 ns (nanosegundo)

Clock

- Se comparados dois processadores de uma mesma arquitetura, o que tiver maior clock, será mais rápido, pois o período em que cada ciclo ocorre é menor.
- Por exemplo: A = x Hz | B = 5x Hz