IBGE
Instituto Brasileiro de Geografia e Estatística

Escola Nacional de Ciências Estatísticas

# Introdução à Programação

## PPT 2: Conceitos Básicos sobre Computação

**Prof. Eduardo Corrêa**

Data 06/04/2022

# O que é um Computador?

- É uma máquina capaz de **executar cálculos** e **tomar decisões lógicas** em uma velocidade da ordem de milhões ou bilhões de vezes superior aos seres humanos.

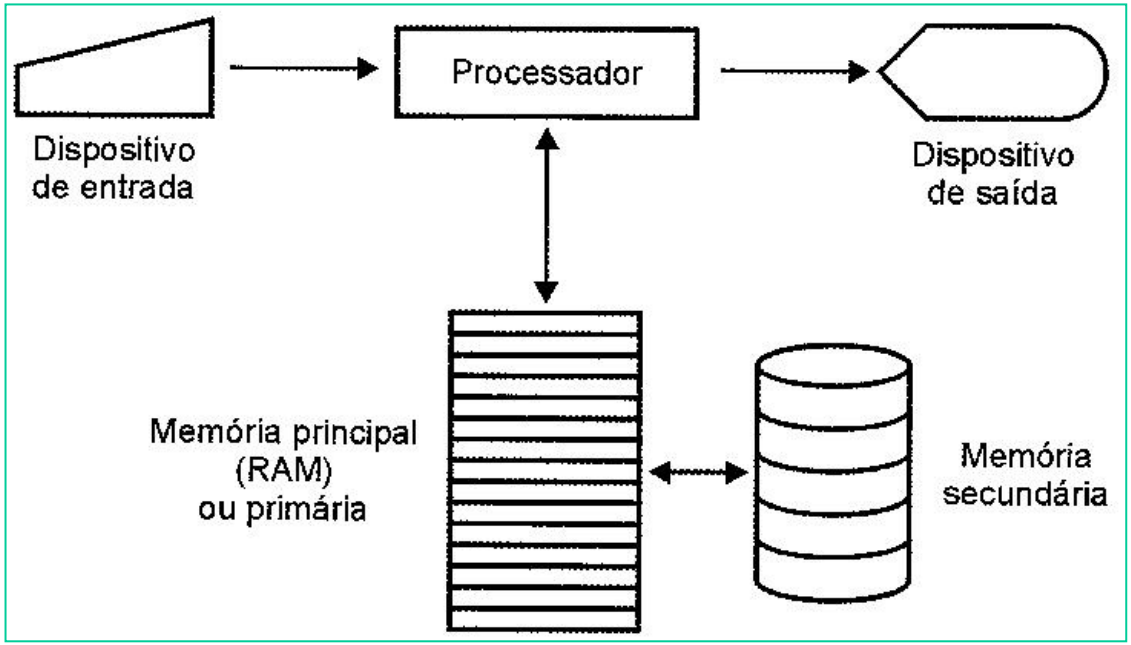
**Exemplo**: Um programa executado em um computador pessoal* leva menos de 0,5 segundo para determinar todos os números primos localizados na faixa entre 1 e 1.000.000 *(a propósito, existem 78.498 números primos menores do que 1.000.000!*).

**Pense nisto**: quanto tempo você levaria para determinar o mesmo resultado utilizando calculadora, lápis e papel?

* Experimento realizado em um PC com processador i5-8265U e 8GB de memória RAM

# Arquitetura de Computadores (1/17)

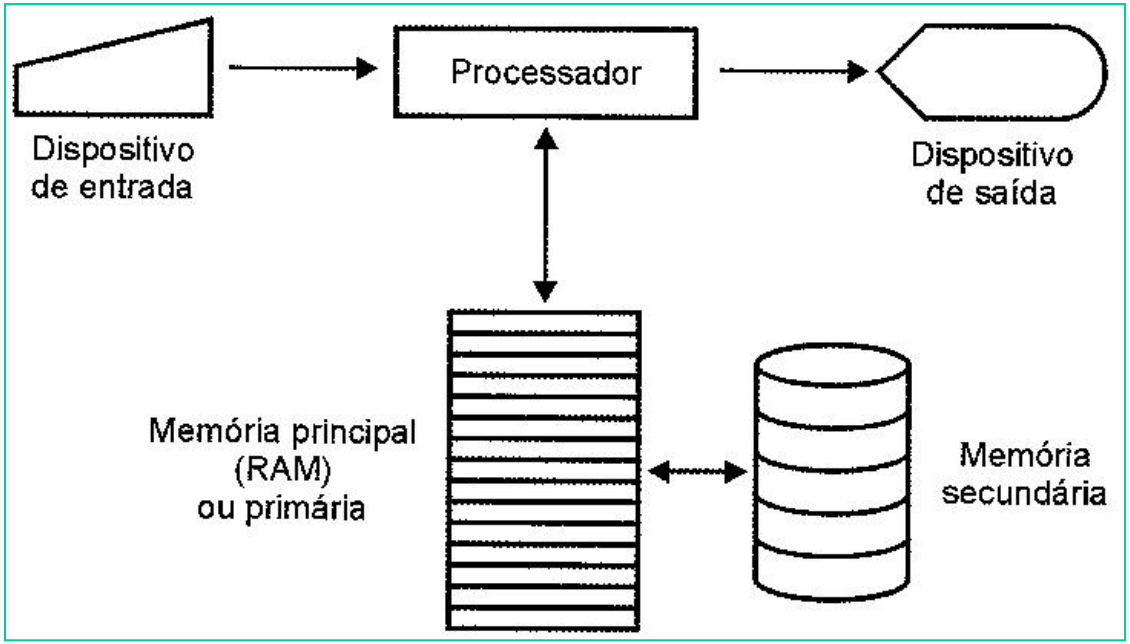
- A arquitetura básica tem-se mantido ao longo do tempo de acordo com o modelo conhecido como **Arquitetura de von Neumann** (proposto na década de 1940)




- **OBS.**: A arquitetura básica de computadores se manteve, mas ao longo do tempo ocorreram notáveis aumentos de **velocidade** e **capacidade de armazenamento**.

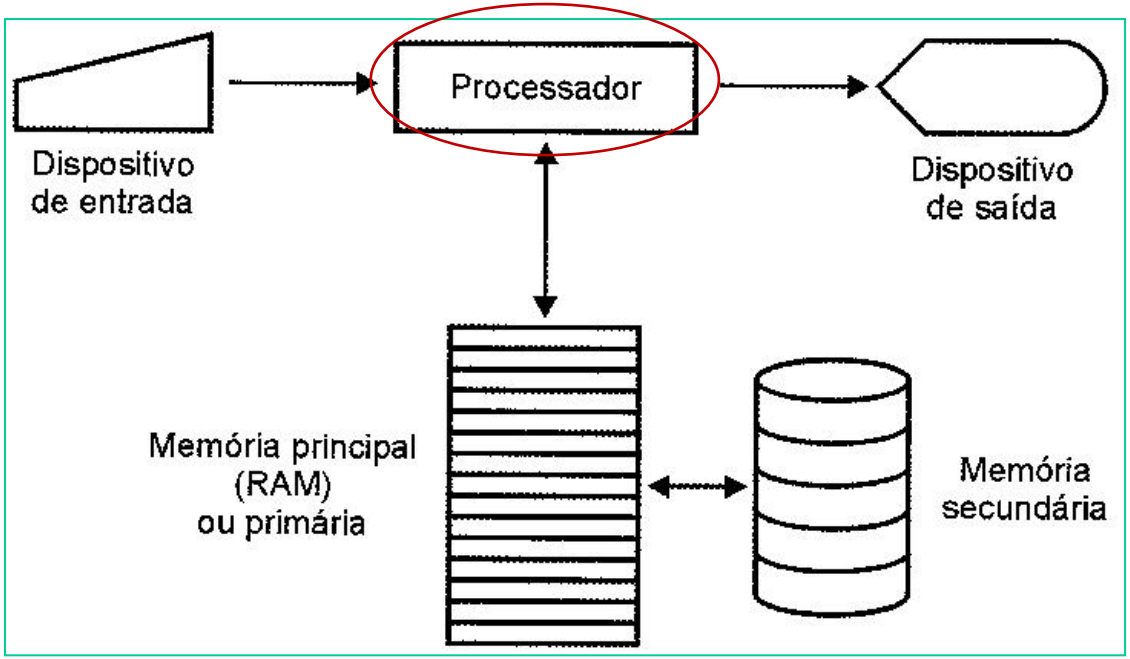
# Arquitetura de Computadores (2/17)

- 5 componentes:
  - Processador
  - Memória Principal
  - Memória Secundária
  - Entrada
  - Saída



# Arquitetura de Computadores (3/17)

- **Processador**
- Também conhecido como **UCP** (Unidade Central de Processamento) ou **CPU** (*Central Processing Unit*).
  - É responsável pela **atividade-fim do sistema**: calcular, tomar decisões lógicas e processar.



# Arquitetura de Computadores (4/17)

- **Processador**
- Para fins de estudo, costuma ser dividido em duas partes: **UAL** e **UC**.
  - **Executa as instruções que formam os programas (UAL)**.
    - É chamada Função de Processamento.
    - Quanto mais rápido o processador executar estas instruções, mais rápida será a execução dos programas.
  - **Controla todo o funcionamento do sistema (UC)**.
    - É chamada Função de Controle.
    - Em decorrência da interpretação de uma determinada instrução, ela emite os sinais de controle para os demais componentes do computador agirem e realizarem alguma tarefa. **Ex**: se programa tem instrução que manda gravar algo no pen drive, a CPU deve emitir sinais para ativar este dispositivo.

# Arquitetura de Computadores (5/17)

- **Processador**
- A CPU executa instruções dentro dos chamados **ciclos de relógio**.
- O relógio é um dispositivo gerador de pulsos cuja duração é chamada de ciclo. A unidade de medida utilizada para a frequência de relógios é o hertz (Hz), que significa 1 ciclo por segundo.
  - As frequências dos computadores atuais são muito elevadas, da ordem dos bilhões de hertz (GHz).
- A frequência representa uma importante medida de desempenho para uma CPU (embora **não** seja a única medida utilizada).
  - **Exemplo**: um computador que tem frequência igual a 1 GHz.
    - Isto significa que seu relógio vibra 1 bilhão de vezes em um segundo.
    - Se a CPU for capaz de executar uma operação de adição a cada ciclo, o número total de adições por segundo será igual a 1.000.000.000!

# Arquitetura de Computadores (6/17)

- **Exemplo:** Considere um computador cuja CPU opera a uma frequência de **2.00GHz**.
  - Suponha que esta CPU gaste **dois ciclos** para realizar uma operação de multiplicação entre 2 números inteiros.
  - Quantas multiplicações de pares de inteiros este computador é capaz de realizar em 5s?

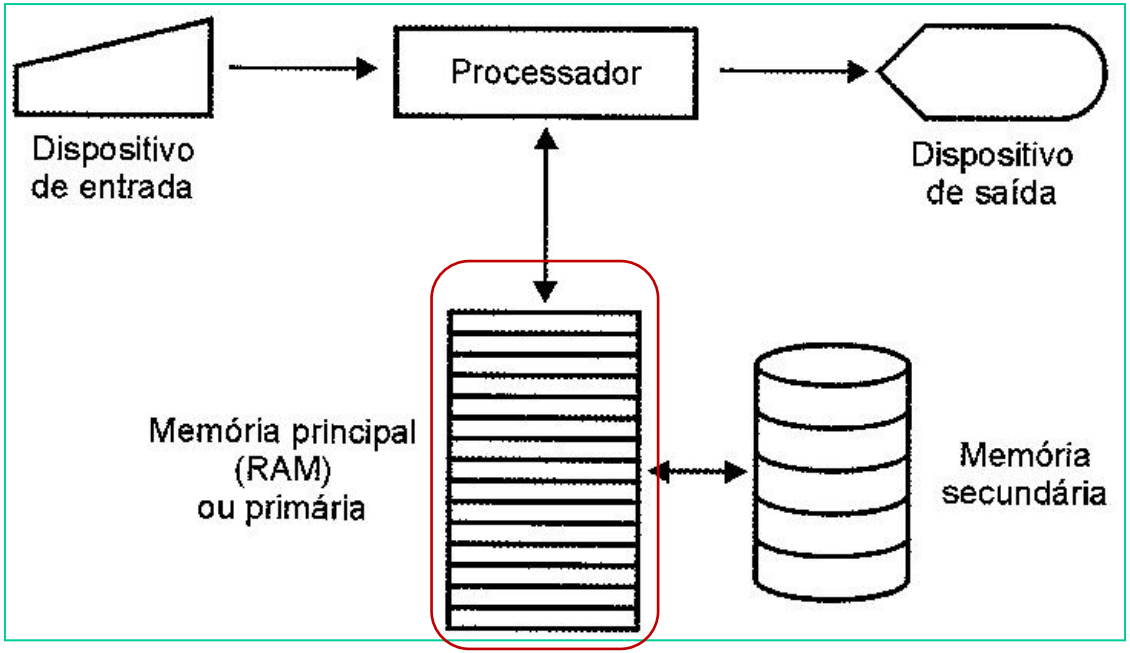
- **Resposta**:
  - 2.00GHz = 2 x $10^9$ ciclos por segundo. Se a multiplicação leva 2 ciclos, logo temos:
  - Nº mult = $\frac{(2 \times 10^9 \text{ ciclos/s}) \times (5\text{s})}{2 \text{ ciclos}}$ = 5 bilhões de multiplicações

# Arquitetura de Computadores (7/17)

- **CPUs multicore**
- São os microprocessadores com múltiplos processadores por chip que permitem o **processamento paralelo**.
  - Tarefas podem ser divididas em subtarefas e as múltiplas CPUs as podem executar em paralelo (ao mesmo tempo).
  - Porém, o programa tem que ser escrito de modo a tirar proveito disso.
- Para reduzir a confusão entre as palavras "processador" e "microprocessador", os fabricantes referem-se aos processadores como **cores** (núcleos)
  - Os microprocessadores com múltiplos processadores são chamados de **multicore**.
  - **Ex.:** família Intel i5, tem modelos dual-core (dois núcleos) e quad-core (quatro núcleos).

# Arquitetura de Computadores (8/17)

- **Memória Principal (MP)**
- Representa um **depósito** onde as informações são guardadas.
- Dividida em **ROM** (fixa) e **RAM** (volátil)

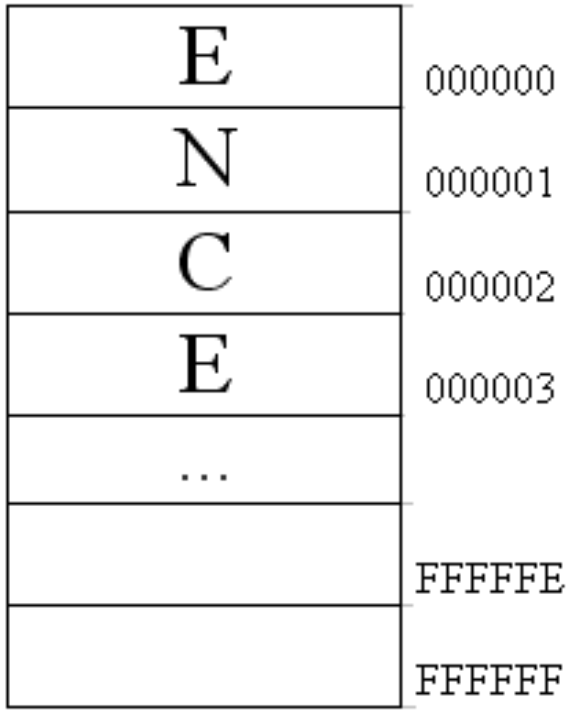


# Arquitetura de Computadores (9/17)

- **Memória RAM**
- Possui grande importância!
  - Para que um **programa** aplicativo possa ser executado (ex.: editor de texto, jogo, planilha, software estatística, browser, etc.), ele precisa inicialmente ser carregado na RAM.
  - Os **dados** que estes programas manipulam (por exemplo, textos e imagens) também precisam estar na RAM.
    - O **conteúdo de variáveis** usadas em programas é sempre armazenado na memória RAM.
    - Afinal de contas, o conteúdo de uma variável é um dado usado no programa!
    - O nome da variável é na verdade uma abstração para um endereço (local) da RAM.

# Arquitetura de Computadores (10/17)

- Na RAM, cada informação é guardada numa posição ou lugar específico da memória, chamado **endereço de memória**.
- Cada endereço possui um **número único**.
  - **Ex**: 000000, 000001, 000002, FFFFFE, FFFFFF (*representados em hexadecimal – logo veremos o que é isso!*).
- Dentro de cada endereço é normalmente possível armazenar um **byte** (byte ≅ caractere). No exemplo ao lado:
  - O endereço 000000 armazena o caractere "E"
  - 000001 armazena "N"
  - FFFFFE está com o conteúdo vazio.



# Arquitetura de Computadores (11/17)

- A quantidade de informação que pode ser armazenada na memória é chamada de **capacidade** ou **tamanho** da memória.
- Esta capacidade é usualmente expressa quantidade de bytes.
  - Mais precisamente, o tamanho da memória costuma ser apresentado em algum múltiplo de $2^{10}$ bytes (*ainda nesta aula veremos o motivo!*).
    - 1 Kbyte (1 kilobyte) = $\mathbf{2^{10}}$ bytes = 1024 bytes
    - 1 Mbyte (1 megabyte) = 1024Kbytes = $\mathbf{2^{20}}$ bytes.
    - 1 Gbyte (1 gigabyte) = 1024Mbytes = $\mathbf{2^{30}}$ bytes.
    - 1 Tbyte (1 terabyte) = $\mathbf{2^{40}}$ bytes.
    - 1Pbyte (1 petabyte) = $\mathbf{2^{50}}$ bytes.
    - 1Exbyte (1 exabyte) = $\mathbf{2^{60}}$ bytes.
    - 1Zbyte (1 zetabyte) = $\mathbf{2^{70}}$ bytes.
    - 1Ybyte (1 yottabyte) = $\mathbf{2^{80}}$ bytes.

# Arquitetura de Computadores (12/17)

- **Exemplo – Capacidade de Memória RAM:** Considere um computador antigo, que possuía memória RAM com capacidade igual a 256 KB. Qual a quantidade máxima de caracteres que esta memória podia armazenar?

- **Resposta**:

- 256 KB representa 256 x $2^{10}$ bytes = 256 x 1024 bytes = 262.144 bytes.

- Se considerarmos que 1 caractere ocupa 1 byte, então a memória do computador antigo podia armazenar 262.144 caracteres.

- **Observações**:

  - PC's atuais são equipados com memória RAM com capacidades como 4GB, 8GB, 16GB e até acima destes valores.

  - Nem sempre 1 caractere ocupa apenas 1 byte – *isto depende do esquema de codificação em uso*. Mas neste curso, vamos assumir que cada caractere ocupa 1 byte.

# Arquitetura de Computadores (13/17)

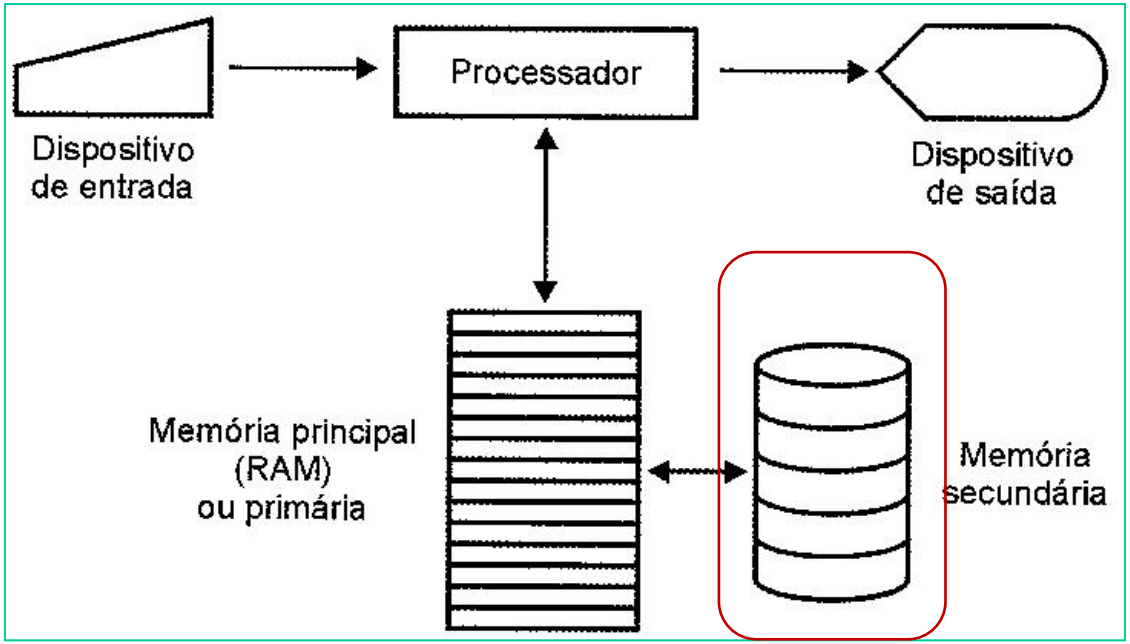
- **Memória RAM**
- As memórias aceitam duas **operações**: **READ** (leitura) e **WRITE** (escrita).
  - READ = recuperar uma informação num local (endereço) da memória.
  - WRITE = gravar uma informação num local (endereço) da memória.
  - **Exemplo** (analogia): Biblioteca
    - WRITE: guardar livro na estante/prateleira adequada (ENDEREÇO).
    - READ: pegar livro emprestado.
- **OBS**: No entanto, nos computadores a ESCRITA é uma **operação destrutiva** (o conteúdo anterior é apagado e substituído por um novo conteúdo).

# Arquitetura de Computadores (14/17)

- **Memória RAM - Tempo de Acesso**:
  - Importante propriedade relacionada ao desempenho (velocidade) da memória.
  - Indica quanto tempo a memória gasta para ler ou escrever uma informação num endereço.
  - **Melhor explicando**: período de tempo decorrido desde o instante em que foi iniciada a operação de acesso (quando a origem – em geral é a CPU – passou o endereço de acesso para o sistema de memória), até que a informação requerida (instrução ou dado) tenha sido efetivamente transferida.
  - Nos PC's atuais: da ordem de 50ns a 70ns (nanosegundos, $10^{-9}$s)

# Arquitetura de Computadores (15/17)

- **Memória Secundária**
- É uma memória de **baixa velocidade** e **grande capacidade de armazenamento**.
  - Dados não são voláteis: permanecem gravados mesmo quando desliga-se o computador.
  - Ex.: dico magnético, SSD, flash disk, DVD.



# Arquitetura de Computadores (16/17)

- **Comparação Memória Principal x Memória Secundária:**

- **Memória Principal - RAM**
  - Capacidade Típica nos PC's: 2GB a 64GB
  - Tempo de Acesso: 50-70ns ($10^{-9}$s).
  - Preço de 1GB ≈ R$ 35,00

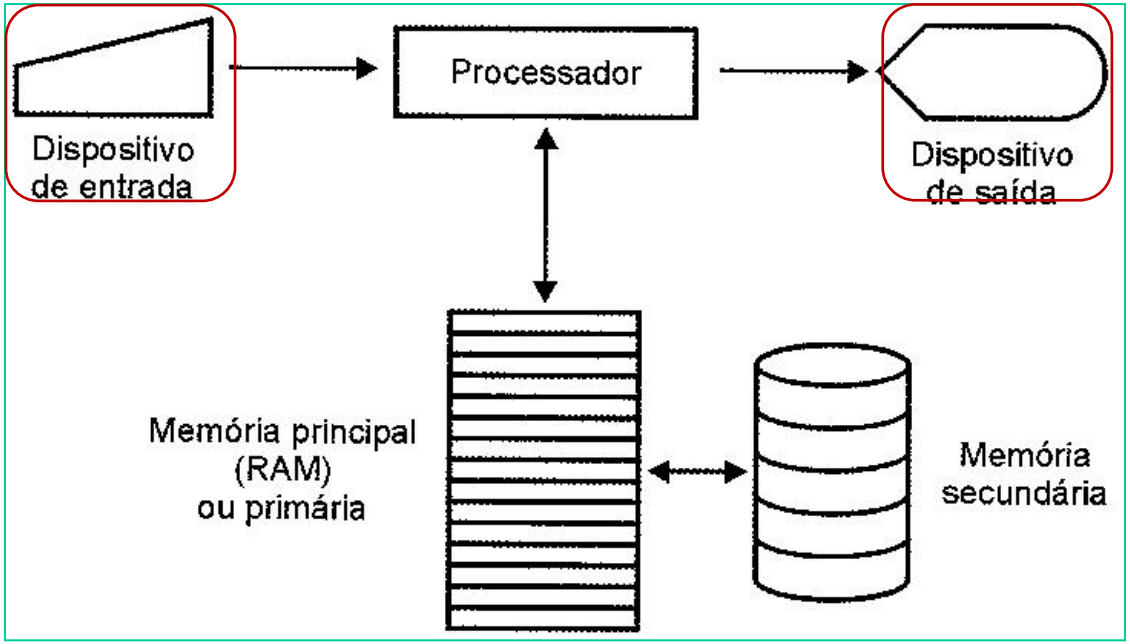
- **Memória Secundária**
  - Capacidade Típica nos PC's: 500GB ou mais
  - Principais tipos: Disco Magnético ou SSD.
  - Tempo de Acesso: 5-20ms ($10^{-3}$s) para o disco magnético e 0,1-0,2ms para SSD
  - Preço de 1GB ≈ R$ 0,40 (disco magnético) e R$ 0,80 (SSD)

- **Justificativa**: As memórias principal e secundária são construídas com tecnologias diferentes (daí a diferença de preço e velocidade).

# Arquitetura de Computadores (17/17)

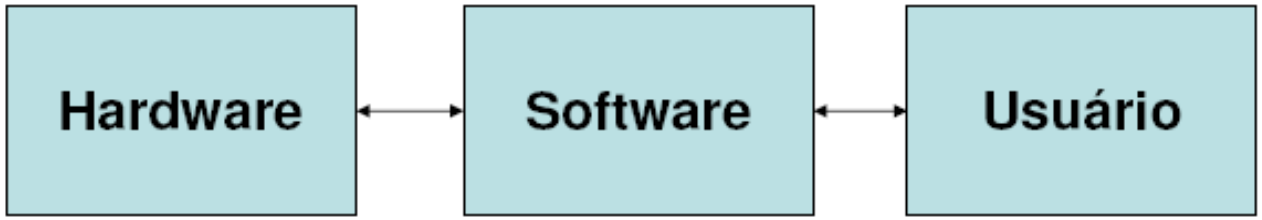
- **Dispositivo de Entrada:** "seção receptora" do computador. Obtém informações dos **dispositivos** (ou **periféricos**) de entrada e coloca estas informações disponíveis para outras unidades (como memória e CPU). **Ex**: Teclado, Mouse, Scanner, Câmera, Leitora de Código de Barras.

- **Dispositivo de Saída:** "seção de expedição" do computador. Pega informações que foram processadas por um programa, para torná-las disponíveis aos usuários. **Ex**: Monitor, Impressora, Caixa de Som.



# Software (1/3)

- Software = **Programa de Computador**.
  - É uma sequência de instruções a serem seguidas pelo processador para executar determinada tarefa.
  - Permite com que o usuário possa tirar proveito da máquina, do hardware!




- Pode ser de dois **tipos**:
  - Software de Sistema.
  - Aplicativos.

# Software (2/3)

- **Software Aplicativo**
- Software utilizado para solucionar um problema ou realizar uma tarefa específica para o usuário.
  - É o que torna o computador útil!
  - Neste curso aprenderemos a construir software aplicativo.
  - **Ex**: editor de texto, navegador Web, jogos, software para controle de estoque, software para gerenciamento de RH, planilha eletrônica, software para análise estatística, etc.

# Software (3/3)

- **Sistema Operacional**
- É um programa essencial que **gerencia todo o hardware do computador**.
- É a camada intermediária entre os chamados **softwares aplicativos** (ex: Word, Excel, Jogos, Editores Gráficos, etc) e o hardware.
- Responsável por tarefas como:
  - Controle e alocação de memória para os programas.
  - Comunicação com os periféricos de entrada e saída.
  - Gerenciamento de redes.
  - Gerenciamento de arquivos.
- Exemplos: Windows, Linux, Mac-OS, Android (celular), ...

# A Linguagem Binária (1/3)

- Quando estamos realizando a leitura de um texto em Português conseguimos entender as informações apresentadas porque **conhecemos o formato e o significado de símbolos** usados.
  - Temos 26 letras (sem contar caracteres acentuados);
  - 10 algarismos (0, 1, ... 9);
  - Diversos sinais (+, -, *, %, etc.).
  - Sabemos as formas como as letras se unem para formar as palavras.
  - Conhecemos diversas regras para a construção de frases em Português (por exemplo, regras de sintaxe).
  - Etc.

# A Linguagem Binária (2/3)

- O computador também possui a sua "linguagem natural", ou seja, a linguagem que ele consegue entender diretamente.
  - É a **linguagem de máquina** ou **linguagem binária**.
  - Diferente do Português, possui apenas dois símbolos: 0 (zero) e 1 (um).

```
00000000101000100000000100011000
00000000100000100001000000100001
10001101111000100000000000000000
10001110000100100000000000000100
10101110000100100000000000000000
10101101111000100000000000000100
00000011111000000000000000001000
```

# A Linguagem Binária (3/3)

- Toda informação armazenada em memória ou processada pela CPU, precisa que ser traduzida para a linguagem de máquina.

- Exemplos:

  - **Programas de computador**: um programa em Python, R ou qualquer linguagem de alto nível precisa ser traduzido para a linguagem de máquina para que o computador possa entende-lo. Um software especial chamado **tradutor** se encarrega dessa tarefa.

  - **Representação de Números**: por exemplo, se quero somar 5 + 8, é preciso antes traduzir 5 e 8 para binário!

  - **Representação de Caracteres**: os caracteres digitados via teclado são convertidos para o formato binário antes de serem armazenados em memória. Para isto o computador utiliza uma tabela de conversão interna. 'A' possui um código binário, 'B' possui outro código, etc.

# Base Numérica (1/2)

- A **quantidade de algarismos** disponíveis em um dado sistema de numeração é chamada de **base**.
- A notação indicada abaixo é utilizada para representar números em diferentes bases.

  $(10010101)_2$ – número na base 2

  $(304)_5$ – número na base 5

  $(1671)_8$ – número na base 8

- Em todas as bases a **notação posicional** é adotada: quanto mais à esquerda estiver um algarismo, mais ele vale.
- Os **humanos** preferem usar a **base 10** para representar e manipular números, mas os **computadores** usam a **base 2**.

# Base Numérica (2/2)

Eu posso escrever qualquer número em qualquer base!

Porém, como sou um ser humano, só consigo ter a idéia exata da quantidade que um número representa quando ele está expresso na base 10!

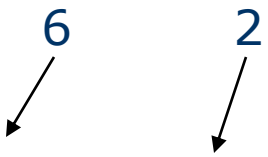
Felizmente, converter números **de uma base b para a base 10** é algo muito fácil!

# Conversão da Base b para a Base 10

- $(1101)_2 = (?)_{10}$


$1 \times 2^3 + 1 \times 2^2 + 0 \times 2^1 + 1 \times 2^0 = \mathbf{13}$

- $(62)_8 = (?)_{10}$




$6 \times 8^1 + 2 \times 8^0 = \mathbf{50}$

# Conversão da Base b para a Base 10

- $(1101)_2 = (?)_{10}$


$1 \times 2^3 + 1 \times 2^2 + 0 \times 2^1 + 1 \times 2^0 = \mathbf{13}$

- **Generalizando:** para converter um número de *n* algarismos expresso na base *b* para a base 10 use a fórmula a seguir:

$$\mathbf{A_{n-1} * b^{n-1} + A_{n-2} * b^{n-2} + \ldots + A_1 * b^1 + A_0 * b^0}$$

Onde: - $A_{n-1}$ representa o algarismo mais à esquerda (mais significativo)
- $A_{n-2}$ representa o segundo algarismo mais à esquerda

- e assim sucessivamente, até $A_0$ que representa o algarismo mais à direita.

# Base 2 (Binária)

- Estudamos esta base, pelo fato de ser usada internamente pelo computador.
- Infelizmente os números na base 2 têm o comprimento muito extenso, o que torna a sua manipulação difícil para seres humanos.

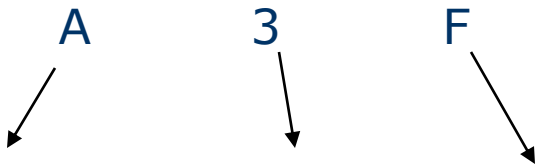
| Número na Base 10 | Número na Base 2 |
|---|---|
| 0 | 0 |
| 1 | 1 |
| 2 | 10 |
| 3 | 11 |
| 4 | 100 |
| 5 | 101 |
| 6 | 110 |
| 7 | 111 |
| 8 | 1000 |
| 9 | 1001 |
| 10 | 1010 |
| 11 | 1011 |
| 12 | 1100 |
| 13 | 1101 |
| 14 | 1110 |
| 15 | 1111 |
| 100 | 1100100 |
| 1000 | 1111101000 |

# Base 16 (Hexadecimal)

- Como os números binários têm o inconveniente citado no slide anterior, muitas vezes em textos técnicos e programas usa-se a **base 16**.
- Na base 16 existem – claro! – 16 algarismos.
  - "A", "B", "C", "D", "E" e "F" representam, respectivamente, os valores (da base 10): 10, 11, 12, 13, 14 e 15.
- Vantagens:
  - 16 é potência de 2, o que facilita a conversão entre valores expressos nas bases 2 e 16.
  - Números grandes podem ser expressos de uma forma muito compacta.

# Base 16 (Hexadecimal)

- A conversão entre base 16 e base 10 é feita do jeito já mostrado para outras bases.
- $(A3F)_{16} = (?)_{10}$




$10 \times 16^2 + 3 \times 16^1 + 15 \times 16^0 = \mathbf{2623}$

# Base 16 (Hexadecimal)

- A tabela abaixo mostra como a notação hexadecimal é mais compacta e agradável aos olhos humanos, quando comparada com  a notação binária.

| Número na Base 10 | Número na Base 16 | Número na Base 2 |
|---|---|---|
| 10 | A | 1010 |
| 11 | B | 1011 |
| 12 | C | 1100 |
| 13 | D | 1101 |
| 14 | E | 1110 |
| 15 | F | 1111 |
| 100 | 64 | 1100100 |
| 1000 | 3E8 | 1111101000 |

# Conversão da Base 10 para a Base b

- $(43)_{10} = (?)_2$

| | | | |
|---|---|---|---|
| 43 ÷ 2 | = 21 | resto = 1 | (algarismo mais à direita) |
| 21 ÷ 2 | = 10 | resto = 1 | (2º mais à direita) |
| 10 ÷ 2 | = 5 | resto = 0 | (3º mais à direita) |
| 5 ÷ 2 | = 2 | resto = 1 | (4º mais à direita) |
| 2 ÷ 2 | = 1 | resto = 0 | (5º mais à direita) |
| 1 ÷ 2 | = 0 | resto = 1 | (algarismo mais significativo) |

**$(43)_{10} = (101011)_2$**

- Divide-se o número decimal pelo valor da base desejada; o **resto** encontrado é o valor do algarismo **menos significativo** na base b.
- Em seguida divide-se o quociente encontrado pela base b; o resto é o algarismo seguinte.
- O processo é executado **até** que o **quociente da divisão** seja **igual a zero**.

# Conversão da Base 10 para a Base b

- $(490)_{10} = (?)_{16}$

490 ÷ 16 = 30 resto = 10 → A (algarismo mais à direita
10 = A em hexa!)

30 ÷ 16 = 1 resto = 14 → E (2º mais à direita
14 = E em hexa!)

1 ÷ 16 = 0 resto = 1 (algarismo mais significativo)

**$(490)_{10} = (1EA)_{16}$**

# Conversão: Base 2 para Base 10

- A conversão da base 2 para a base 10 pode ser realizada através do seguinte "**macete**"

| | | | | | | | |
|---|---|---|---|---|---|---|---|
| **(1** | **1** | **0** | **0** | **0** | **0** | **0** | **1)$_2$** |
| 128 | 64 | 32 | 16 | 8 | 4 | 2 | 1 |
| $2^7$ | $2^6$ | $2^5$ | $2^4$ | $2^3$ | $2^2$ | $2^1$ | $2^0$ |

- $(11000001)_2 = (?)_{10}$
- $(11000001)_2 = 128 + 64 + 1 = \mathbf{(193)_{10}}$

# Conversão: Base 2 para Base 10

- Descrição do "macete":

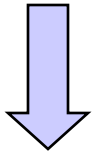
| | | | | | | | |
|---|---|---|---|---|---|---|---|
| **(1** | **1** | **0** | **0** | **0** | **0** | **0** | **1)$_2$** |
| 128 | 64 | 32 | 16 | 8 | 4 | 2 | 1 |
| $2^7$ | $2^6$ | $2^5$ | $2^4$ | $2^3$ | $2^2$ | $2^1$ | $2^0$ |

- Escreva o número 1 ($2^0$) embaixo do dígito mais à direita (dígito menos significativo) do número binário que você deseja converter.
- Embaixo do segundo dígito menos significativo, escreva 2 ($2^1$). Embaixo do terceiro dígito menos significativo, escreva 4 ($2^2$) e assim sucessivamente, até chegar ao dígito mais significativo.
- Ao terminar, some todos os números que estejam escritos embaixo dos dígitos '1'. O resultado obtido corresponderá ao valor do número na base 10.

# Conversão: Base 2 para Base 16

- **Conversão em 2 passos:**
- **PASSO 1**: Dividimos o número binário em **grupos de 4 bits** da direita para a esquerda.

$$(1\ 1\ 0\ 0\ 0\ 0\ 0\ 1)_2 = (?)_{16}$$



$$(1\ 1\ 0\ 0\ |\ 0\ 0\ 0\ 1)_2 = (?)_{16}$$

# Conversão: Base 2 para Base 16

- **PASSO 2**: Convertemos cada um dos grupos (usando o mesmo macete da conversão da base 2 para base 10!).

| | | | | | | | | |
|---|---|---|---|---|---|---|---|---|
| **(1** | **1** | **0** | **0** | **\|** | **0** | **0** | **0** | **1)$_2$** |
| 8 | 4 | 2 | 1 | | 8 | 4 | 2 | 1 |
| | | **C** | | | | | **1** | |

- $(11000001)_2 = (C1)_{16}$

# Conversão: Base 2 para Base 16

- O mesmo número em diversas bases...

| | | | | | | | | |
|---|---|---|---|---|---|---|---|---|
| **(1** | **1** | **0** | **0** | **\|** | **0** | **0** | **0** | **1)$_2$** |
| 8 | 4 | 2 | 1 | | 8 | 4 | 2 | 1 |
| | | **C** | | | | | **1** | |

- $(11000001)_2 = (C1)_{16}$
- $(C1)_{16} = 12 \times 16^1 + 1 \times 16^0 = (193)_{10}$

# Conversão: Base 2 para Base 16

- **Exemplo 2:**
  - $(10101)_2 = (?)_{16}$
  - 0001 0101

    **1** **5**
  - $(10101)_2 = \mathbf{(15)_{16}}$
  - Veja que neste exemplo, complementamos o número $(10101)_2$ com 3 zeros à esquerda, para que fosse possível dividí-lo em 2 grupos de 4.

# Conversão: Base 16 para Base 2

- Utiliza-se o mesmo "macete", no sentido inverso.
  - Cada dígito do número hexadecimal é transformado em um número binário de 4 bits
  - **Exemplo**:
  - $(A7)_{16} = (?)_2$
  - A 7

    **1010 0111**
  - $(A7)_{16} = (10100111)_2$

# Conversão: Base 16 para Base 2

- $(A7)_{16} = (10100111)_2$
- $(A7)_{16} = 10 \times 16^1 + 7 \times 16^0 = (167)_{10}$
- $(10100111)_2 = 1 + 2 + 4 + 32 + 128 = (167)_{10}$

# Resumo

| MUNDO DOS HUMANOS | MUNDO DO COMPUTADOR |
|---|---|
| Muitos símbolos: A..Z, 0..9, !, +, %, *, ... | Apenas dois símbolos: **0** e **1**. |
| 1 + 1 = 2 | 1 + 1 = 10 |
| 1K = 1000 ($10^3$) | 1K = 1024 ($2^{10}$) |
| Português, Inglês, Francês, Espanhol, Grego, Japonês, ... | Linguagem de Máquina |