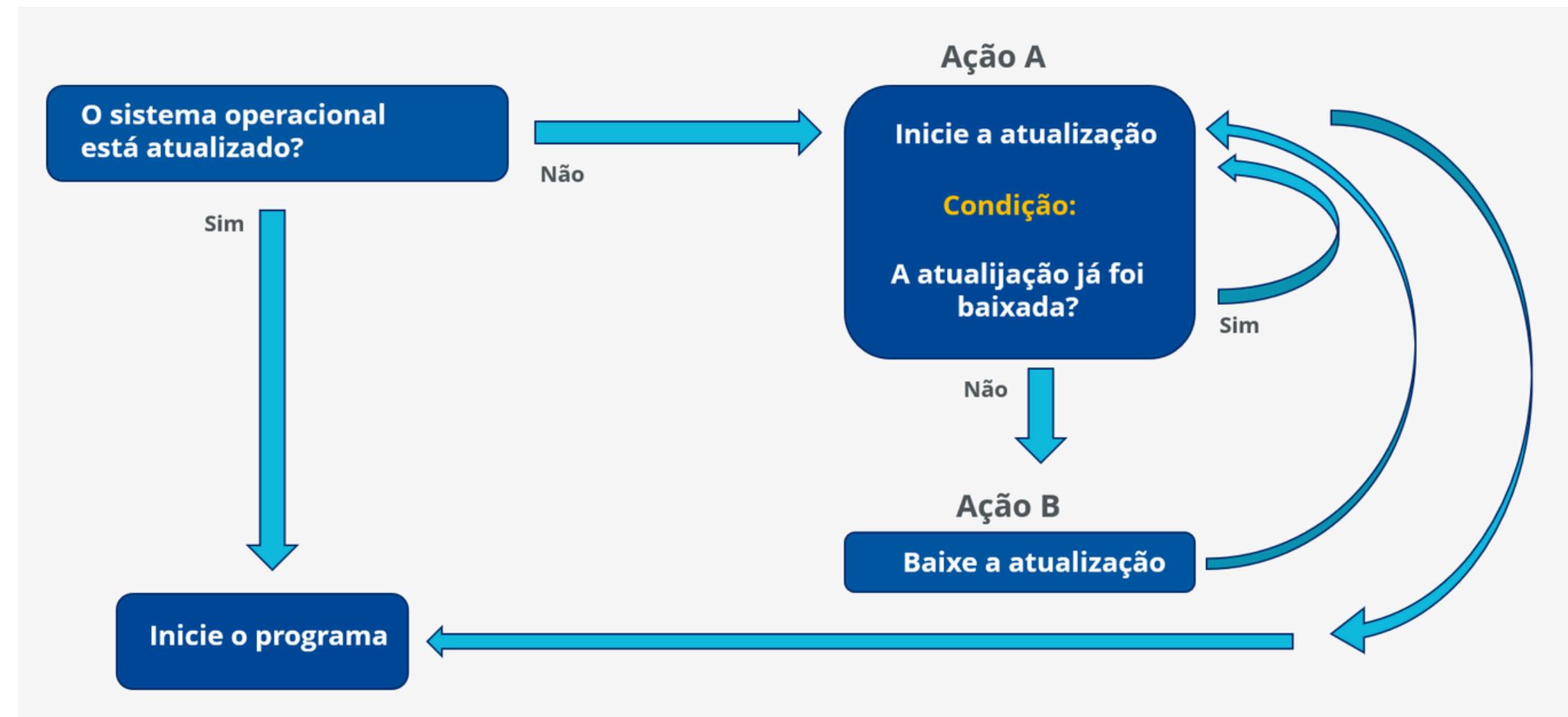
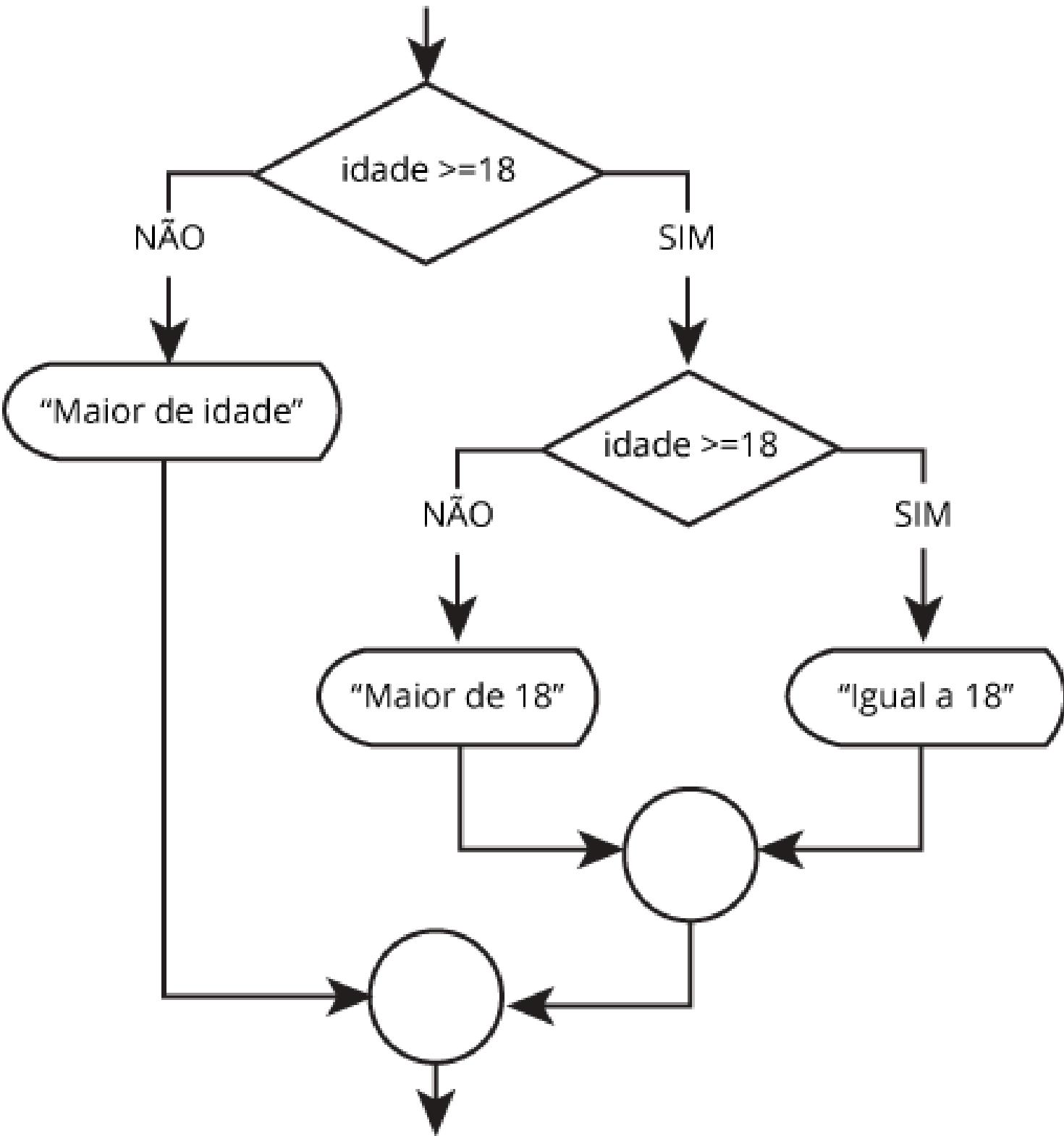


**ENTENDENDO  
PROGRAMACAO**

# O QUE É PROGRAMAÇÃO?

Programação é o processo de criar, testar e manter códigos que dão **instruções** a um computador. Essas instruções são escritas em linguagens específicas, como *Python*, *JavaScript* e *Java*, e são usadas para **resolver** problemas, **automatizar** tarefas ou **criar** aplicações.





# ÁREAS DE ATUAÇÃO DESENVOLVIMENTO WEB

**Frontend:** Focado na interface e na experiência do usuário (*HTML, CSS, JavaScript*).



**Backend:** Gerencia o que acontece nos bastidores de um sistema (*Node.js, Python, PHP*).

**Fullstack:** Combina habilidades de frontend e backend.



Cria sistemas para computadores e dispositivos, como ferramentas de edição e sistemas operacionais.



# O QUE É UM COMPUTADOR?

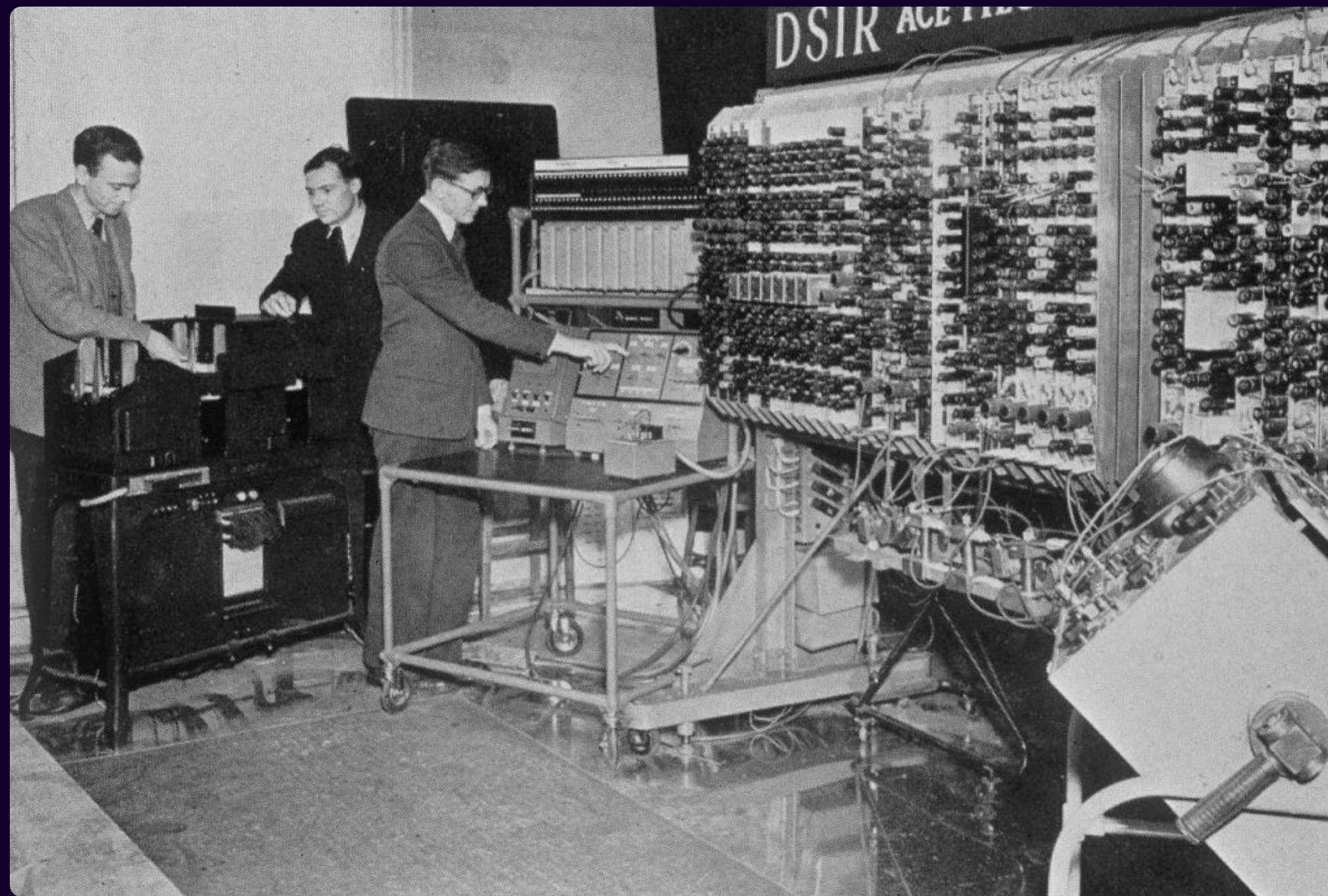
A palavra “**computador**” vem do verbo “**computar**” que, por sua vez, significa “**calcular**”. Sendo assim, podemos pensar que a criação de computadores começa na **idade antiga**, já que a relação de contar já intrigava os homens.

Dessa forma, uma das primeiras máquinas de computar foi o “**ábaco**”, instrumento mecânico de origem chinesa criado no **século V a.C.**

Assim, ele é considerado o “**primeiro computador**”, uma espécie de **calculadora** que realizava **operações algébricas**.

- No **século XVII**, o matemático escocês **John Napier** foi um dos responsáveis pela invenção da "réguia de cálculo".
- **Por volta de 1640**, o matemático francês **Pascal** inventou a primeira máquina de calcular automática.
- A primeira calculadora de bolso capaz de efetuar os **quatro principais cálculos matemáticos**, foi criada por **Gottfried Wilhelm Leibniz**.
- A primeira máquina mecânica programável foi introduzida pelo matemático francês **Joseph-Marie Jacquard**. Tratava-se de um tipo de tear capaz de controlar a confecção dos tecidos através de cartões perfurados.
- **George Boole** (1815-1864) foi um dos fundadores da lógica matemática. Essa nova área da matemática, se tornou uma poderosa ferramenta no projeto e estudo de circuitos eletrônicos e arquitetura de computadores.
- Já no século XIX, o matemático inglês **Charles Babbage** criou uma **máquina analítica** que, a grosso modo, é comparada com o computador atual com **memória** e **programas**. Através dessa invenção, alguns estudiosos o consideram o "*Pai da Informática*".

# ALAN TURING



# EVOLUÇÃO DOS COMPUTADORES

## The Bombe - Alan Turing

Os alemães usavam uma máquina conhecida como Enigma, capaz de criptografar mensagens enviadas via telégrafo de forma quase indecifrável para os seus oponentes. A máquina continha 26 chaves capazes de embaralhar as palavras, conectando pequenos cabos que trocavam uma letra pela outra nas mensagens. Os estudos para vencer a Enigma partiram da contribuição de matemáticos e criptoanalistas poloneses que já haviam conseguido decifrar algumas mensagens alemãs. A continuidade dos estudos deu frutos, e em 1940 os aliados conseguiram decodificar mensagens interceptadas do eixo, mas ainda com uma demora muito grande. Todo esse trabalho chegou ao seu ápice em 1941, quando por fim os matemáticos da Station X conseguiram criar a máquina mais poderosa no que dizia respeito a capacidade de decifrar códigos: a bomba eletromecânica, batizada como “The Bombe”.

## Primeira Geração (1951-1959)

Os computadores de primeira geração funcionavam por meio de circuitos e válvulas eletrônicas. Possuíam o uso restrito, além de serem imensos e consumirem muita energia. Um exemplo é o ENIAC (Electronic Numerical Integrator and Computer) que consumia cerca de 200 quilowatts e possuía 19.000 válvulas.

## **Segunda Geração (1959-1965)**

Ainda com dimensões muito grandes, os computadores da segunda geração funcionavam por meio de transistores, os quais substituíram as válvulas que eram maiores e mais lentas. Nesse período já começam a se espalhar o uso comercial.

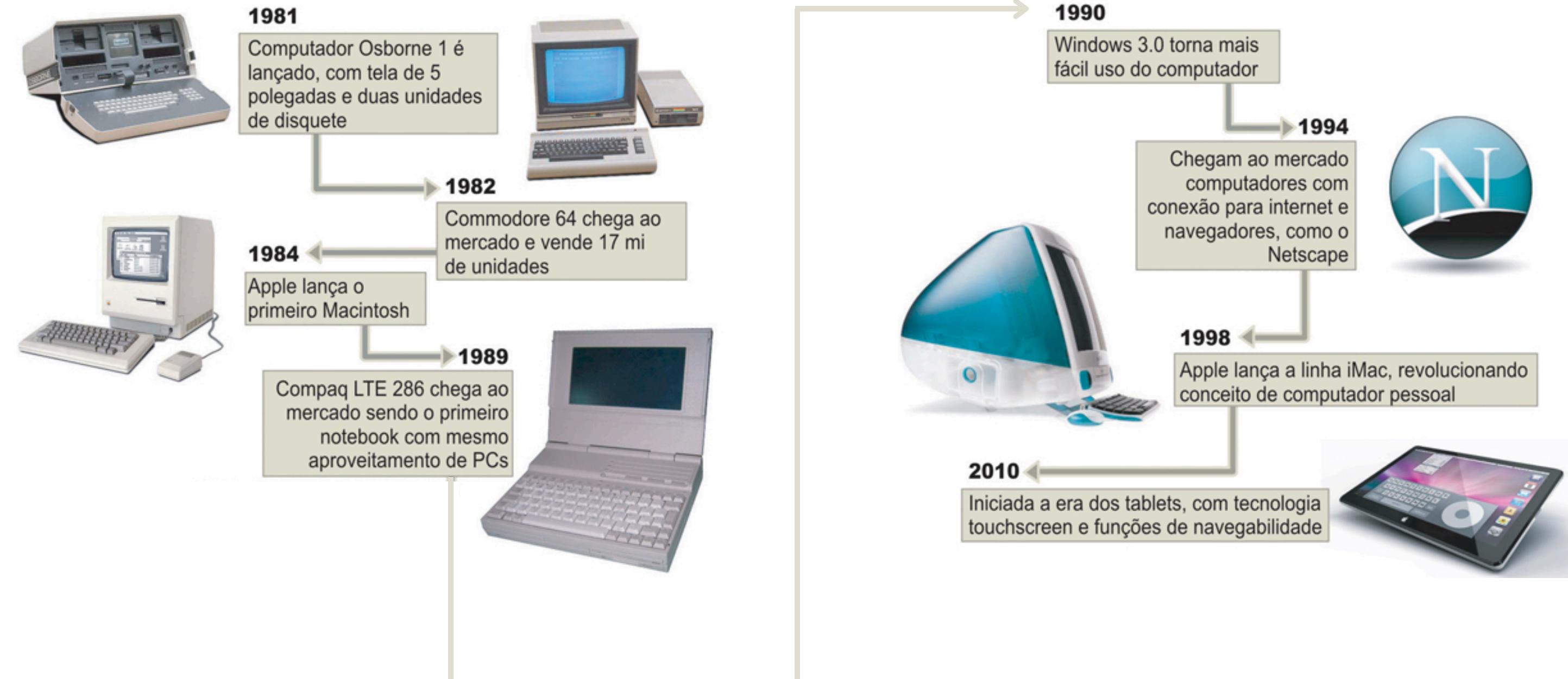
## **Terceira Geração (1965-1975)**

Os computadores da terceira geração funcionavam por circuitos integrados. Esses substituíram os transistores e já apresentavam uma dimensão menor e maior capacidade de processamento. Foi nesse período que os chips foram criados e a utilização de computadores pessoais começou.

## **Quarta Geração (1975-até os dias atuais)**

Com o desenvolvimento da tecnologia da informação, os computadores diminuem de tamanho, aumentam a velocidade e capacidade de processamento de dados. São incluídos os microprocessadores com gasto cada vez menor de energia. Nesse período, mais precisamente a partir da década de 90, há uma grande expansão dos computadores pessoais.

# O COMPUTADOR DOMÉSTICO...



**SOFTWARE**

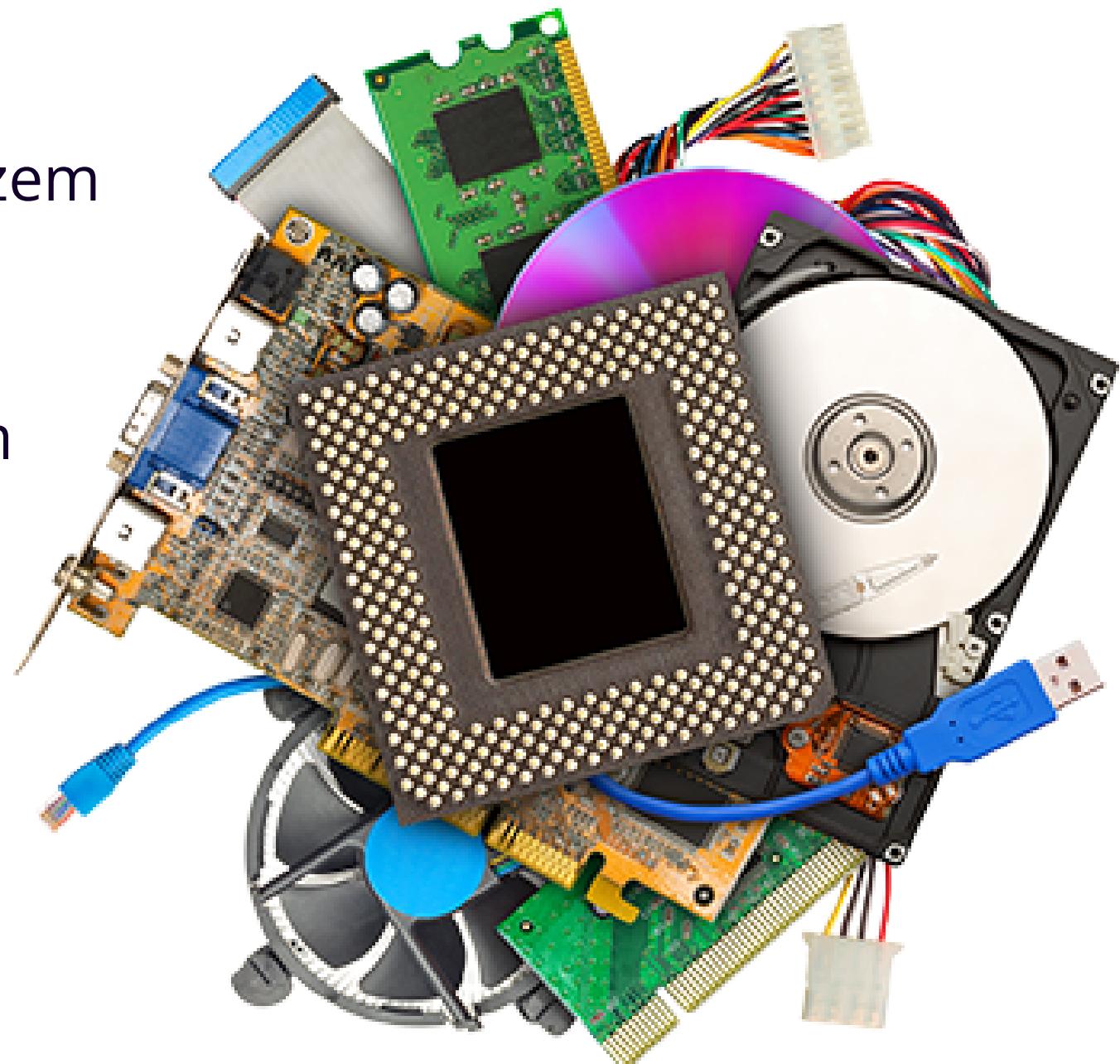
**VS**

**HARDWARE**

Os **hardwares** são as **peças físicas** que compõem um computador, como as *placas*, o *monitor*, o *teclado*, a *placa-mãe* e o *disco rígido*.

Eles são divididos em quatro elementos:

- **Dispositivos de entrada:** são os componentes que o usuário conecta, como teclado e mouse.
- **Dispositivos de saída:** são os componentes que traduzem os dados recebidos para uma linguagem acessível ao usuário, como o monitor e as caixas de som.
- **Componentes internos:** são as peças que se conectam entre si para que o computador funcione.
- **Dispositivos de armazenamento secundário:** são os componentes responsáveis por armazenar os dados permanentemente no computador.



Os **softwares** representam todas as **instruções** que o computador recebe pelo usuário para uma determinada **tarefa** ser executada. Para isso, ele utiliza códigos e **linguagem de programação**. Eles são classificados de duas formas:

- Software de sistema: são programas que permitem a interação do usuário com a máquina. Como exemplo podemos citar o Windows, que é um software pago; e o Linux, que é um software livre.

Software de aplicativo: são programas de uso cotidiano do usuário, permitindo a realização de tarefas, como o editores de texto, planilhas, navegador de internet, etc.

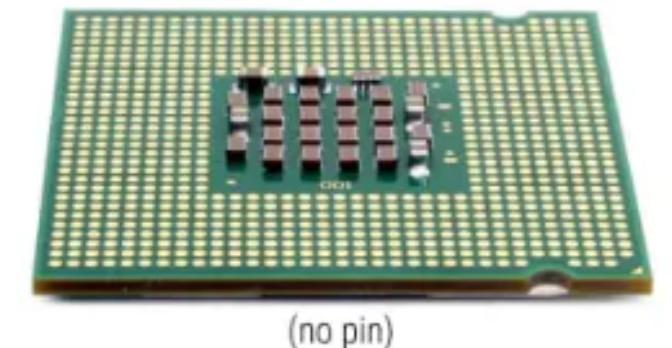


# CPUs

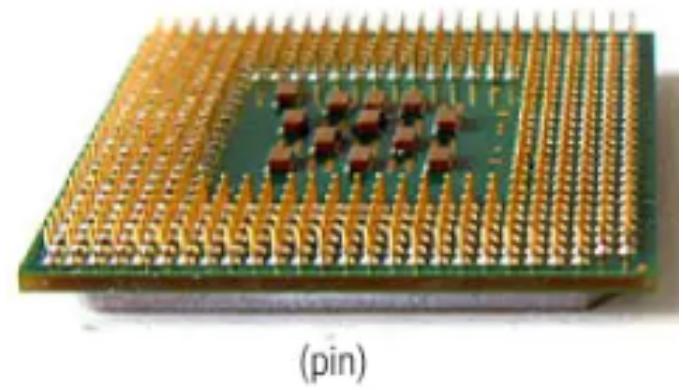
CPU é a sigla para Central Process Unit, ou **Unidade Central de Processamento**. Ele é o principal item de hardware do computador, que também é conhecido como **processador**. A CPU é responsável por **calcular** e realizar **tarefas** determinadas pelo usuário sendo considerado o **cérebro do PC**.

Uma CPU é feita de bilhões de **transistores**, que são pequenas peças microscópicas que controlam a **corrente elétrica** do componente. Existe uma teoria chamada de **Lei de Moore** que diz que a quantidade de transistores de uma **CPU** **dobra a cada dois anos**.

A CPU pode ter uma **placa de vídeo integrada** dentro do seu corpo para gerar e exibir imagens para o usuário. Por questões físicas de espaço, essas **GPUs** integradas não têm muito desempenho, mas ajudam. Nos computadores de mesa é possível trocar o processador, caso seja compatível com a placa-mãe. Já nos **notebooks**, o processador é soldado na placa-mãe e não pode ser trocado.



Intel Processor

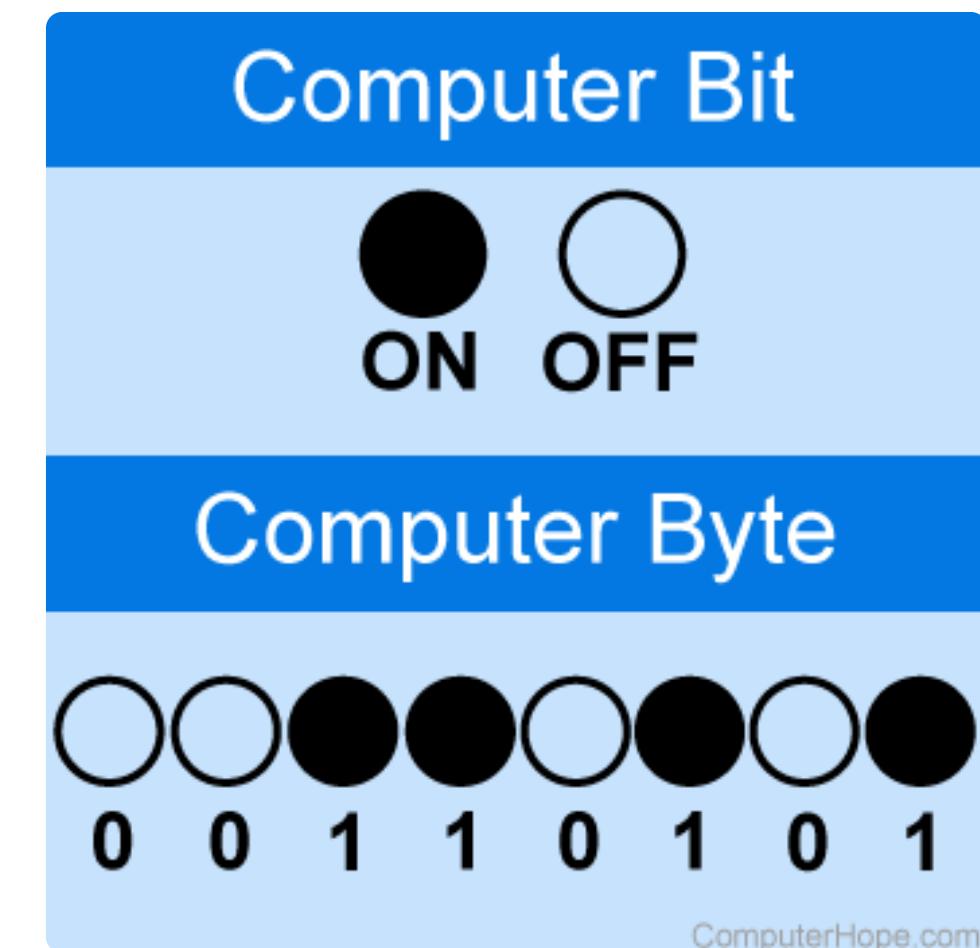
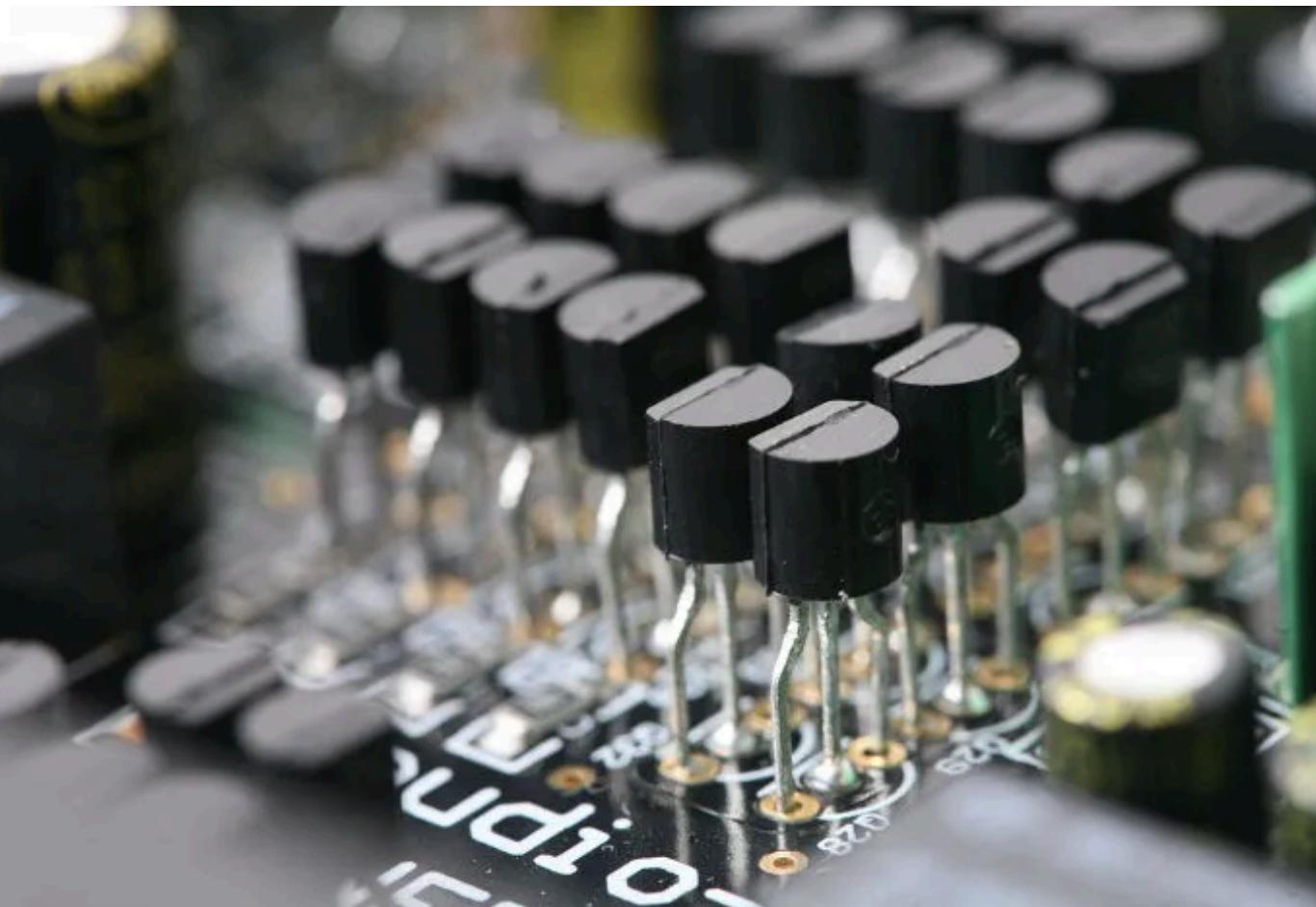


AMD Processor

# CPUs

uma máquina trabalha a partir da interpretação de **números binários**, ou seja, as CPUs compreendem apenas **0** ou **1** e **calculam** os dados para que, posteriormente, eles sejam exibidas na tela como **informações**.

De forma geral, essa peça serve para processar dados e transformá-los em uma informação. Outra função é **centralizar o gerenciamento dos processos** que acontecem no PC.



# COMO DIVIDIR A CPU?

## Unidade de Controle (UC)

Como já foi explicado, uma das funções do processador é fazer com que os demais componentes de uma máquina trabalhem em harmonia. A Unidade de Controle (UC) comanda as ações no computador. Ela garante que os processos sejam executados corretamente, utilizando sinais de controle temporizados.

## ULA

A Unidade Lógica e Aritmética, conhecida como ULA, é o "pedaço" que realiza os cálculos matemáticos e as operações lógicas na CPU. Ao Canaltech, Yuri explicou ela pode ser entendida como uma espécie de calculadora para operações de adição e subtração.

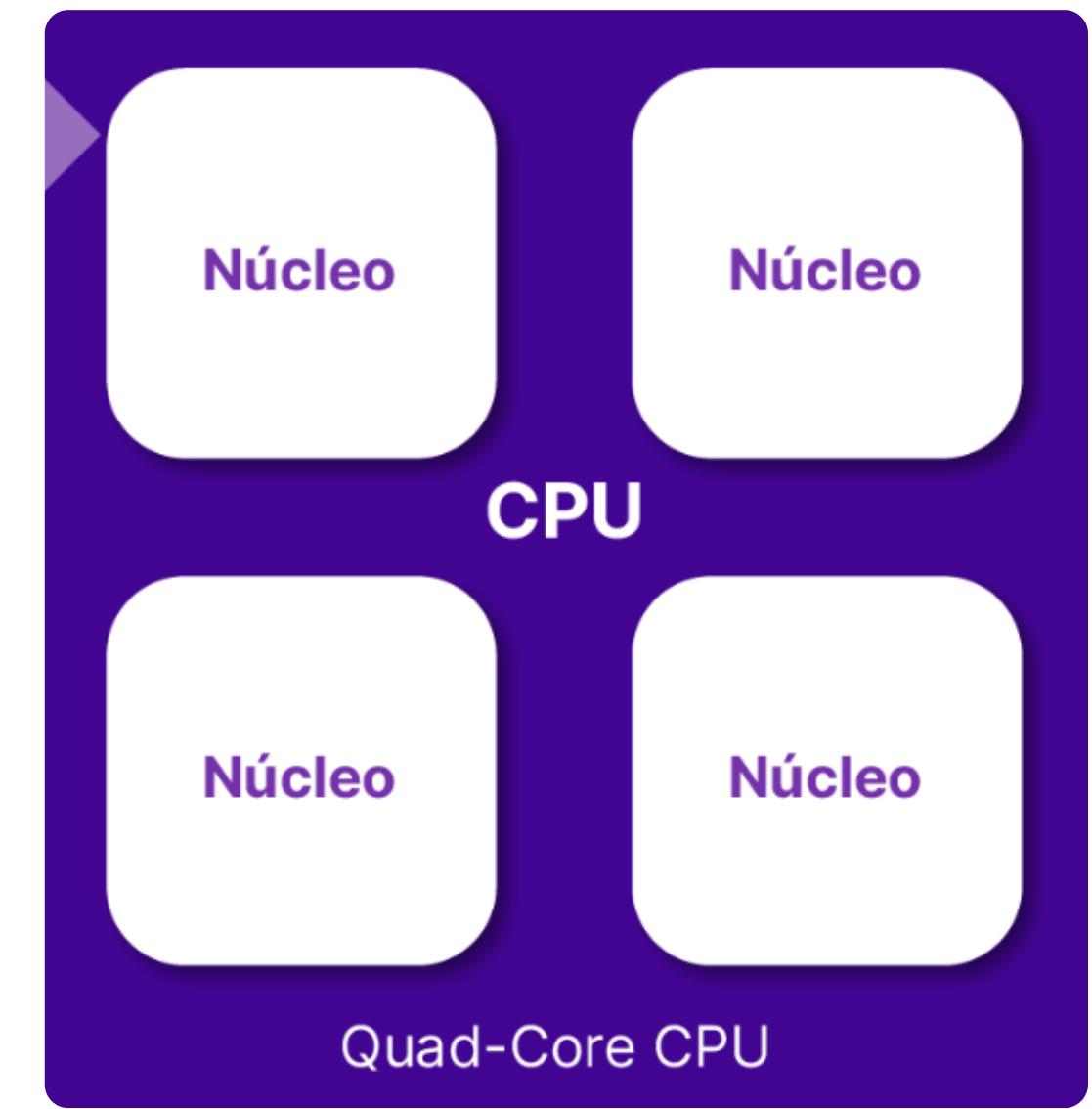
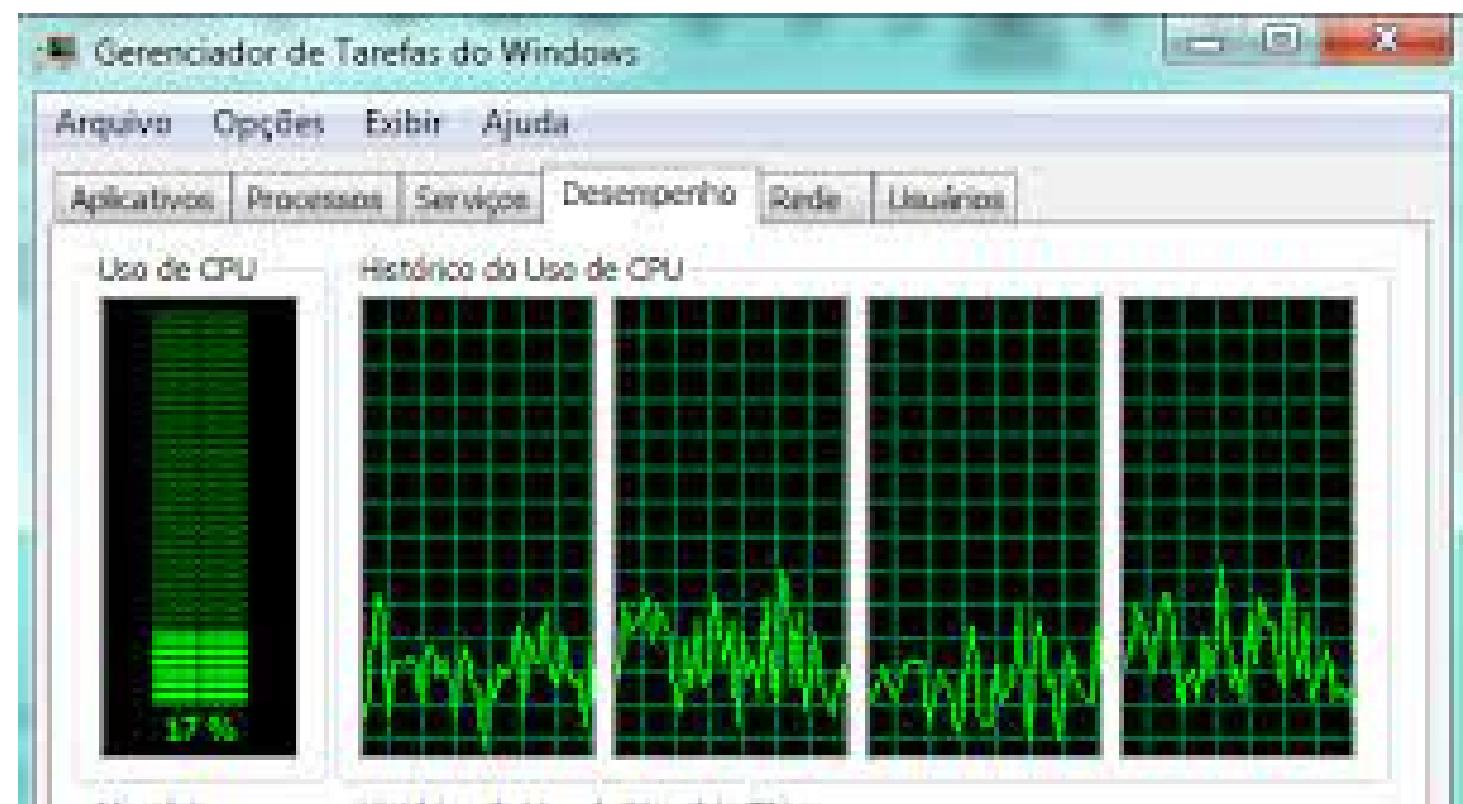
## REGISTRADORES

Assim como a Unidade de Controle, os Registradores têm um nome bem sugestivo. Essa estrutura é uma área de armazenamento que recebe e guarda informações intermediárias específicas. É uma "memória de baixo nível" que armazena poucos dados, mas todos fundamentais para a CPU.

Não confundir Registradores com memória cache, pois ambos são diferentes e têm funções específicas, segundo Daglian.

# O QUE SÃO OS NÚCLEOS DA CPU?

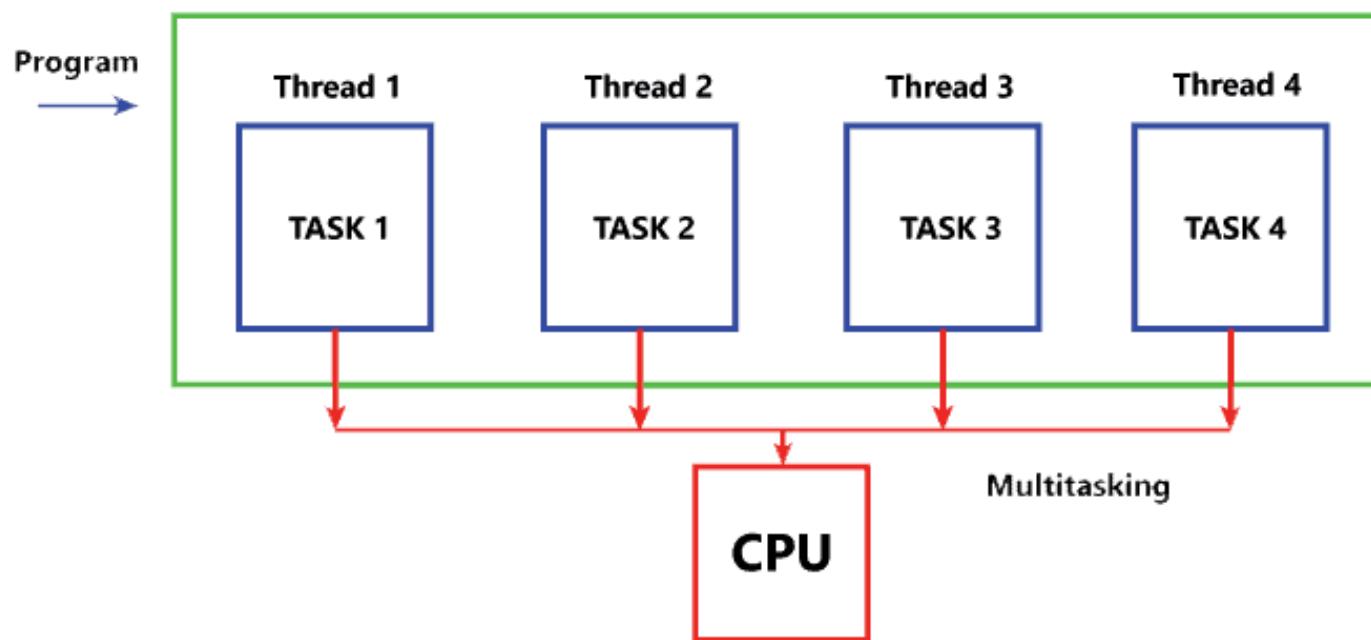
Os núcleos são **unidades de processamento** capazes de **executar instruções**, ou seja, tarefas como jogos, softwares e outras aplicações. A quantidade de cores, como também são conhecidos, influência na capacidade do processador em desempenhar atividades multitarefas.



Quanto maior for o número de núcleos, maior é a capacidade do computador em lidar com vários programas abertos ao mesmo tempo. No entanto, a potência de um processador não é medida apenas pela quantidade dessas estruturas, já que outros fatores entram nessa conta.

# O QUE SÃO OS NÚCLEOS DA CPU?

Além dos núcleos, é normal ouvir falar que uma CPU tem várias threads. Threads são **linhas de instruções** que **organizam processos** para serem executados e finalizados pelos núcleos. De forma geral, quanto mais núcleos um processador tem, mais dessas linhas de instruções existem.



# MEMÓRIAS PRIMÁRIAS VS MEMÓRIAS SECUNDÁRIAS

- As **memórias primárias** (Reais) são aquelas onde o processador endereça diretamente como: *RAM*, *ROM*, *Registradores* e o *CACHE*.
- Já as **memórias secundárias** são aquelas que não podem ser endereçadas diretamente, todas as informações deverão ser mandadas para uma memória intermediária antes (primária). Exemplos: *HD*, *SSD*, *CD*, *DVD*.



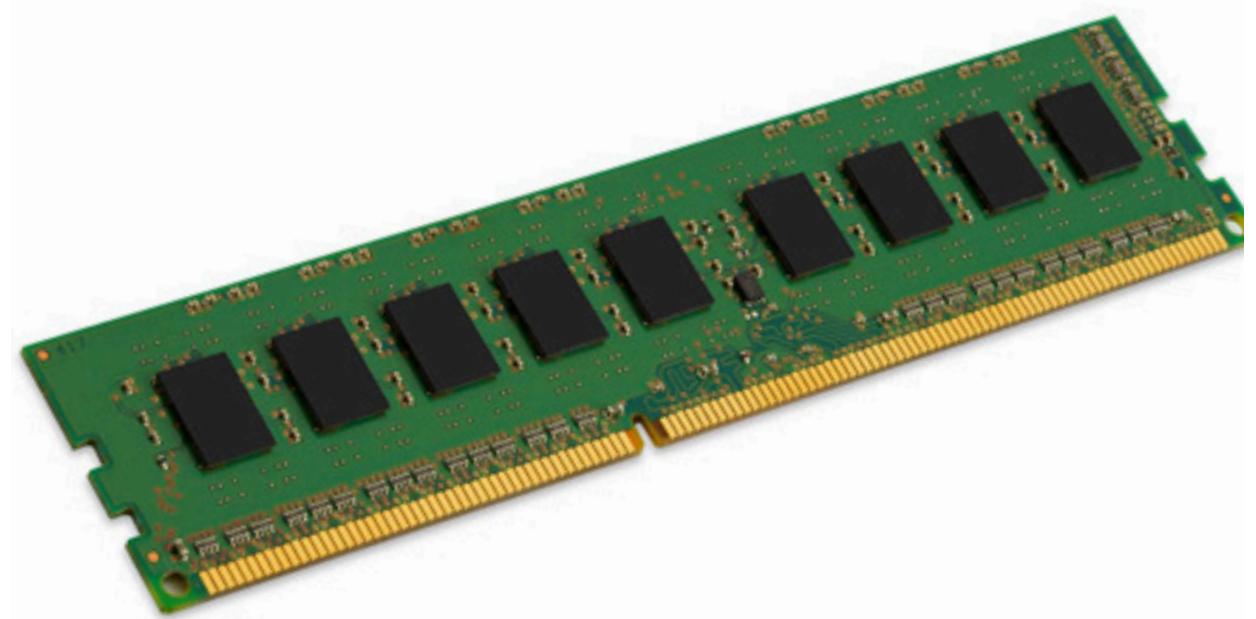
# MEMÓRIA RAM VS MEMÓRIA ROM

## Memória RAM

A Memória RAM (Memória de Acesso Aleatório) é uma memória volátil, que perde os dados ao desligar o dispositivo. Sua função principal é armazenar temporariamente os dados e programas em uso pelo sistema operacional e aplicativos.

Características:

- Velocidade e volatilidade: Permite atualização rápida dos dados, garantindo agilidade ao sistema.
  - Acesso aleatório: Qualquer dado pode ser acessado diretamente, otimizando tarefas simultâneas.
  - Capacidade variável: Dispositivos modernos possuem de alguns a vários gigabytes de RAM, atendendo às demandas atuais.
- A RAM é essencial para o desempenho e eficiência dos dispositivos e programas.



# MEMÓRIA RAM VS MEMÓRIA ROM

## Memória ROM

A Memória ROM (Memória Somente de Leitura) é uma memória não volátil, o que significa que mantém os dados armazenados mesmo quando o dispositivo é desligado. Sua função principal é armazenar o firmware e instruções essenciais para inicializar o sistema.

Características:

- Não volátil e permanente: Ideal para guardar informações críticas, como o BIOS, garantindo a inicialização correta do sistema.
- Somente leitura: Os dados gravados não podem ser alterados pelo usuário, preservando sua integridade.
- Função essencial: Fornece instruções para carregar o sistema operacional e componentes ao ligar o dispositivo.

A Memória ROM é indispensável para o funcionamento básico de qualquer computador ou dispositivo eletrônico.



**INTRODUÇÃO  
À  
LÓGICA**

# O QUE SÃO PORTAS LÓGICAS?

01

Sistema Binário

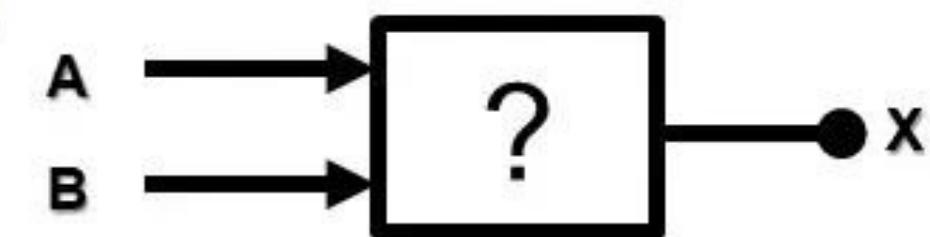
02

Álgebra Booleana

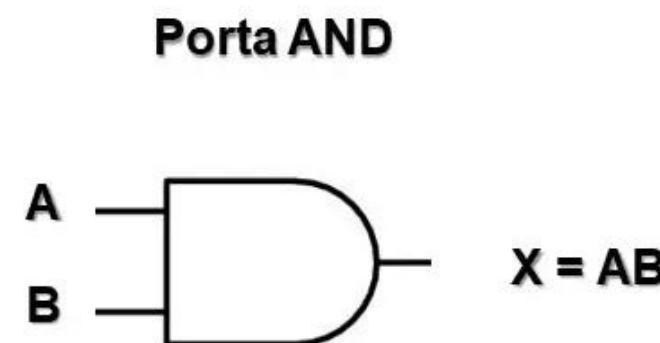
03

Tabela Verdade

Entradas		Saída
A	B	X
0	0	1
0	1	0
1	0	1
1	1	0

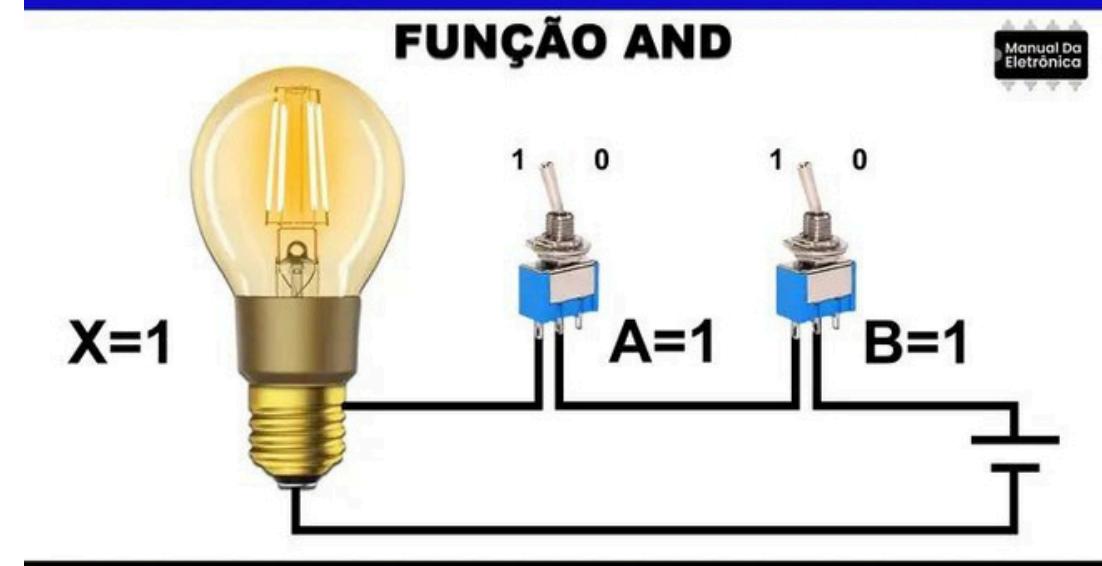


# PORTA E

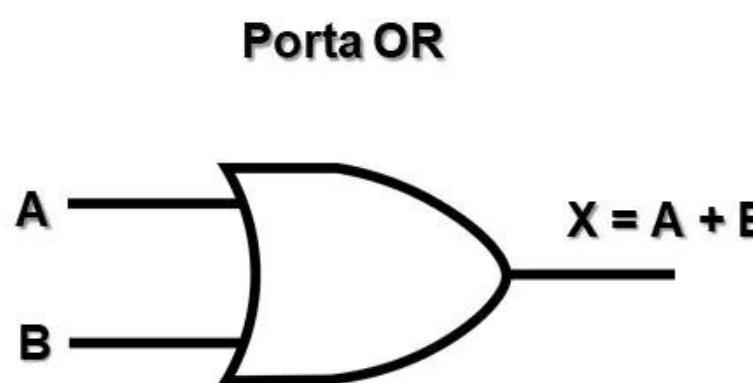


**AND**

A	B	X=AB
0	0	0
0	1	0
1	0	0
1	1	1

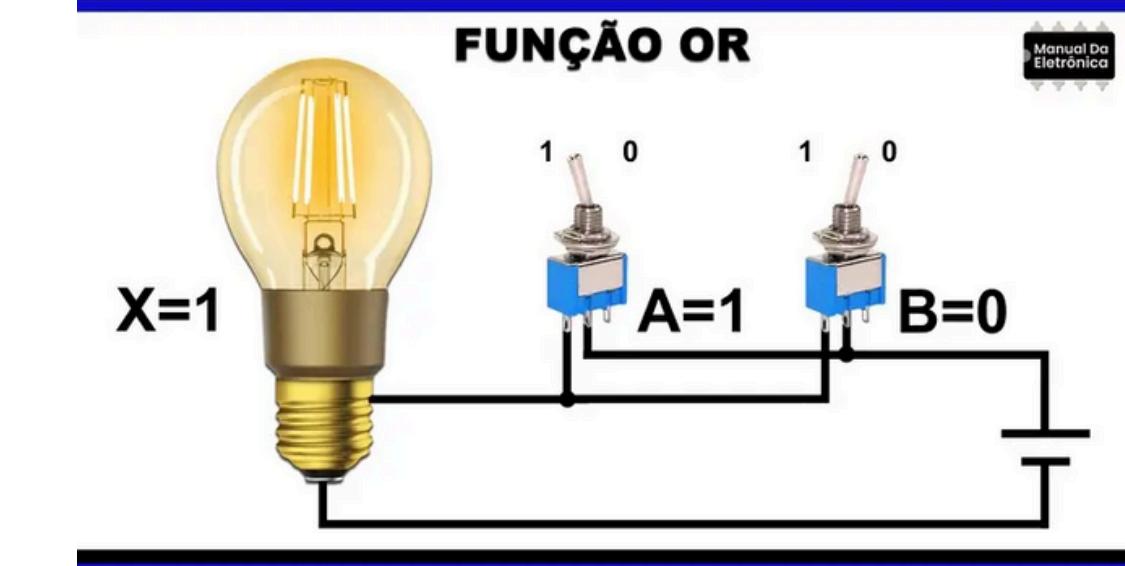


# PORTA OU

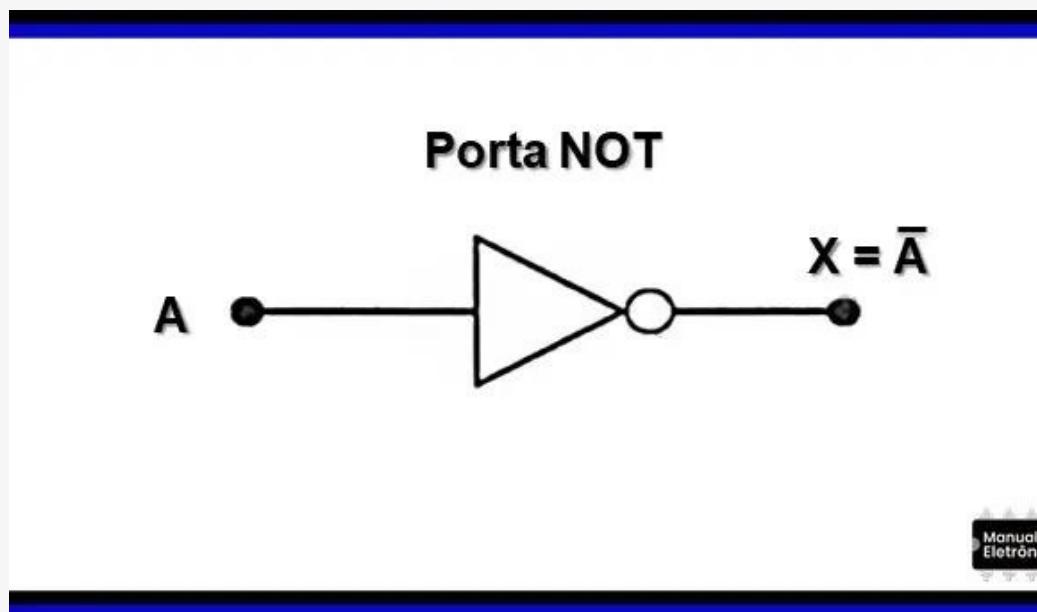


**OR**

A	B	$X=A+B$
0	0	0
0	1	1
1	0	1
1	1	1

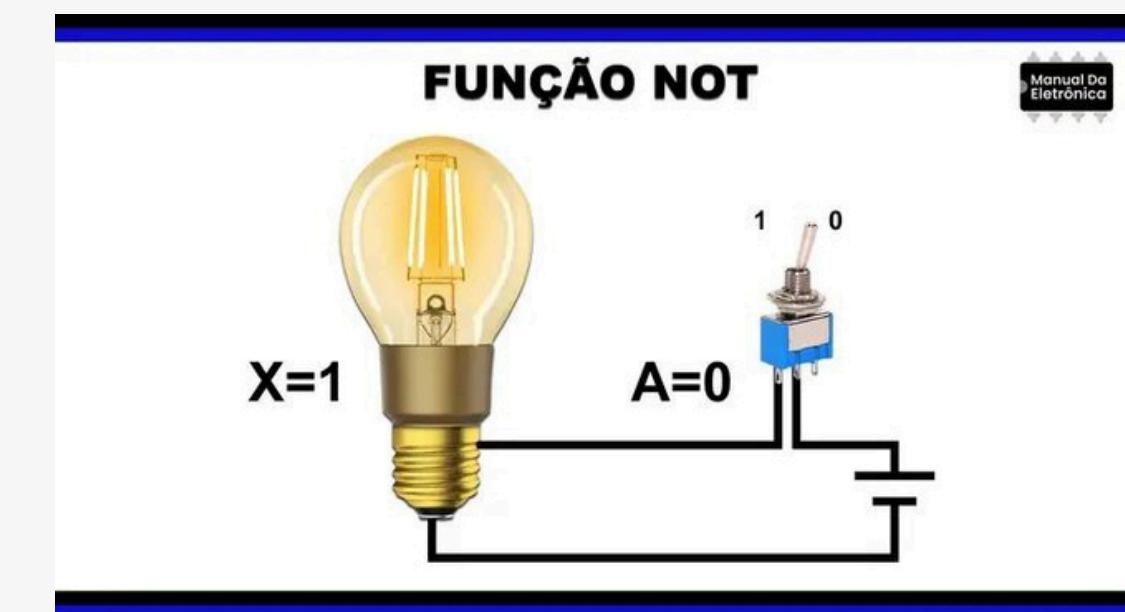


# PORTA NÃO

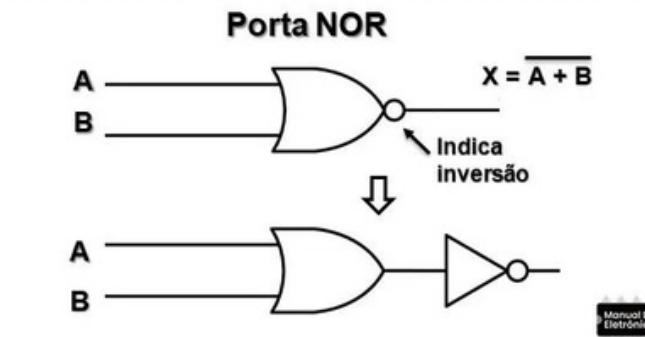
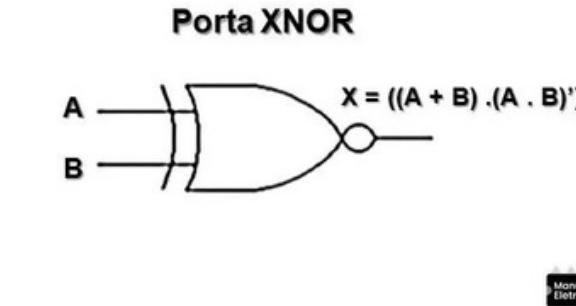
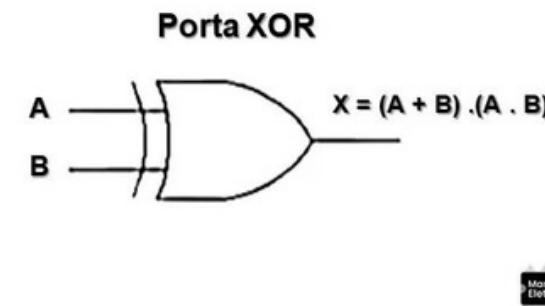
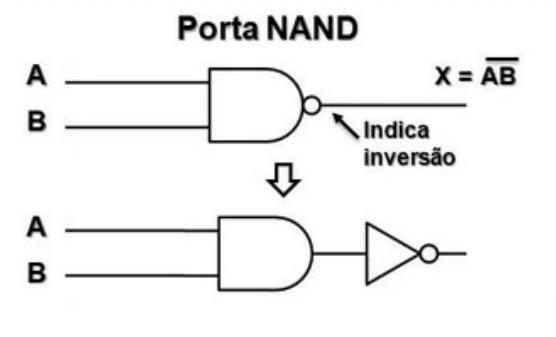


**NOT**

A	X = $\bar{A}$
0	1
1	0



# PORTAS COMPLEXAS



		AND	NAND
A	B	AB	$\overline{AB}$
0	0	0	1
0	1	0	1
1	0	0	1
1	1	1	0

**XOR**

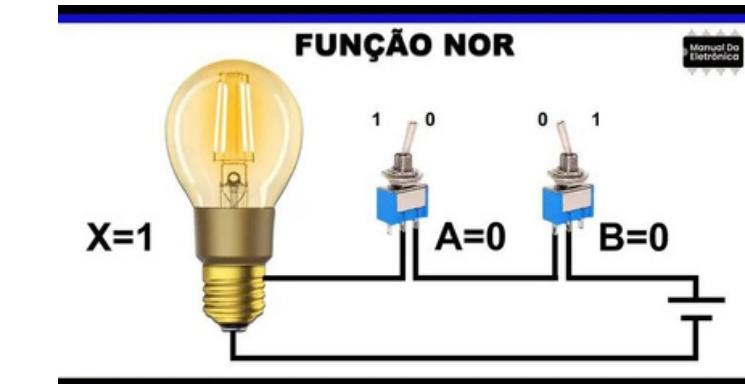
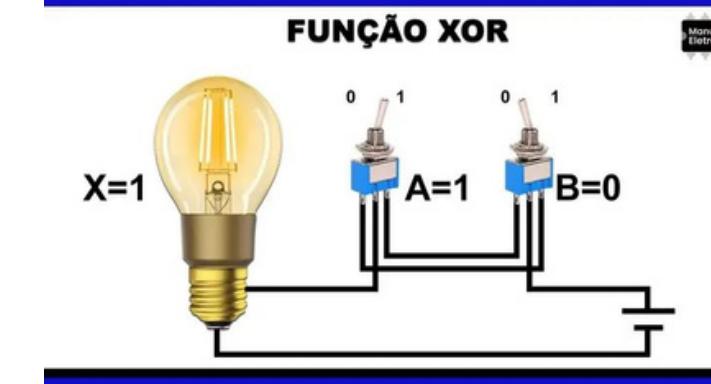
A	B	$X = (A \oplus B)$
0	0	0
0	1	1
1	0	1
1	1	0

**XNOR**

A	B	$X = \overline{(A \oplus B)}$
0	0	1
0	1	0
1	0	0
1	1	1

**OR**

A	B	$A + B$	$\overline{A + B}$
0	0	0	1
0	1	1	0
1	0	1	0
1	1	1	0



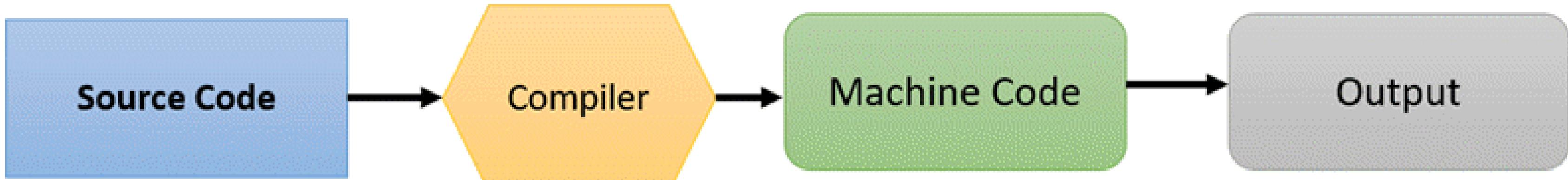
# MÉTODOS DE TRADUÇÃO: INTERPRETADOR E COMPILADOR

# O QUE É UM COMPILADOR?

Um **compilador** é um programa que traduz códigos de uma linguagem de alto nível para uma linguagem que o computador entende, como **assembly** ou **código binário**, mantendo o mesmo significado.

Ele funciona em duas etapas:

1. Análise – Examina o código, identifica sua estrutura e verifica erros como sintaxe e tipos de dados.
2. Síntese – Converte o código analisado em linguagem de máquina, gerando um novo arquivo pronto para execução.



# EXEMPLOS DE COMPILADORES

- 01** **GCC.** Ele é um pacote de programas responsáveis por fazer a compilação do seu código em C, C++, Objective-C, Fortran, Ada, Go, and D.
- 02** O **NetBeans** é uma IDE mas também compilador de códigos em diversas linguagens, como C, C++ e PHP.
- 03** Uma alternativa online e que também vai servir é o **Ideone**. Suporta mais de 60 linguagens.
- 04** **Borland Turbo C.** Turbo C é um dos mais básicos e populares compiladores para a linguagem C. Foi introduzido em 1987. Foi muito famoso pelo seu pequeno tamanho, velocidade de compilação e baixo preço.

# O QUE É UM INTERPRETADOR?

Diferente do compilador, o interpretador executa o código linha por linha, sem gerar um novo arquivo. Ele traduz e roda o programa em tempo real, permitindo detectar erros imediatamente. Isso facilita a depuração, pois o interpretador mostra onde o problema ocorreu no momento da execução.

Algumas linguagens interpretadas populares incluem:

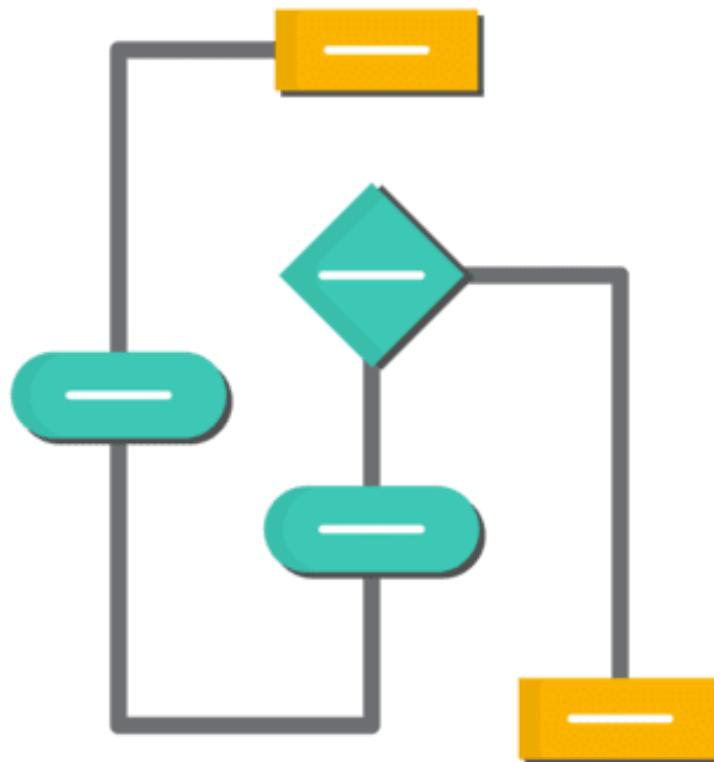
- Python 
- JavaScript 
- Ruby 
- Perl 
- Lua 
- Bash 
- R 
- PHP 



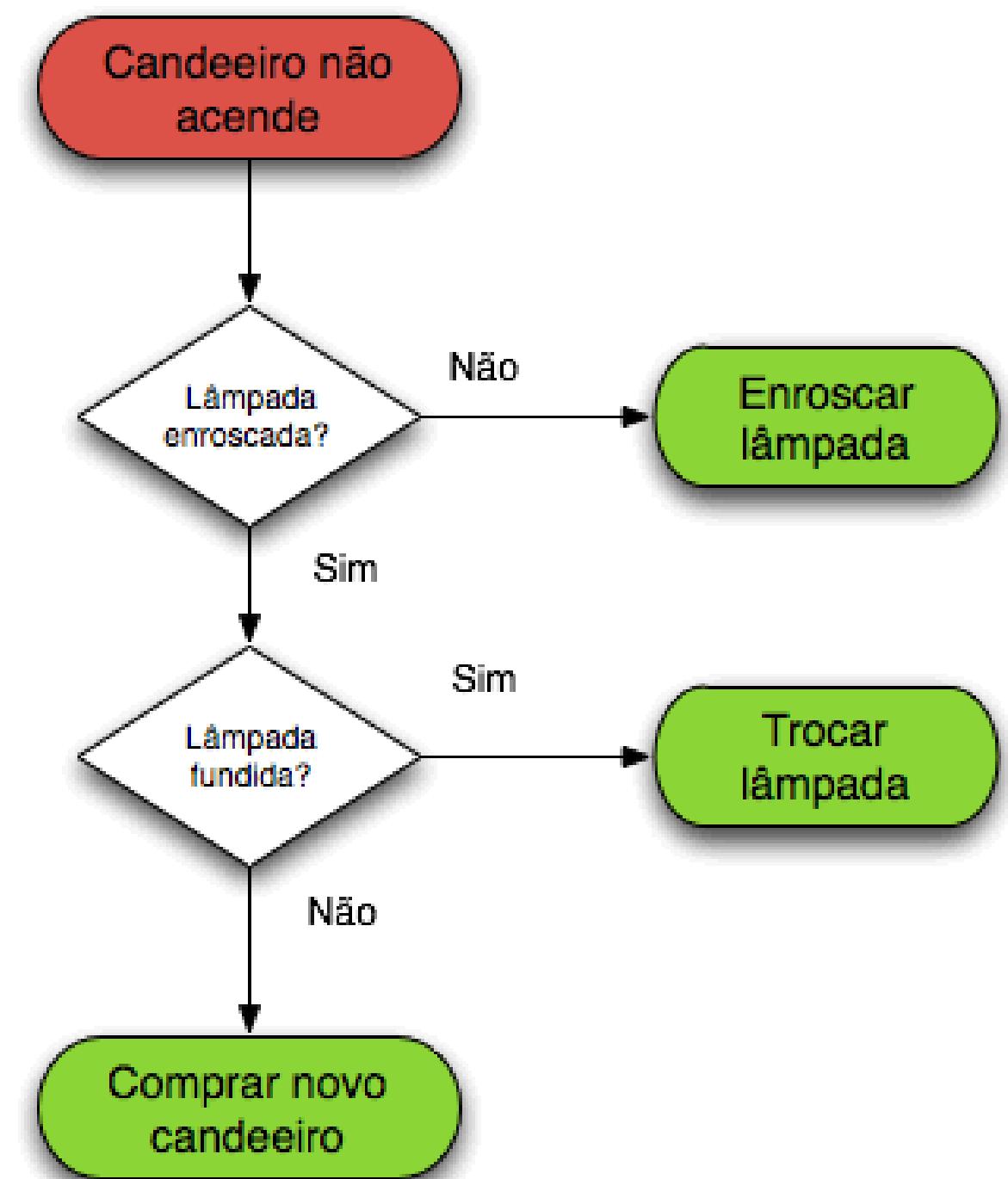
# O QUE SÃO ALGORITMOS?

Um algoritmo é uma **sequência** de raciocínios, **instruções** ou operações para alcançar um **objetivo**, sendo necessário que os passos sejam **finitos** e operados **sistematicamente**.

- variáveis: são as informações de entrada inseridas que determinam aonde o algoritmo poderá ir. As mais comuns são *texto*, *inteiro*, *lógico* e *real*;
- comandos de repetição: consiste no uso de “se” e “enquanto”, para que o algoritmo saiba o que fazer quando determinados processos ocorrerem e o que fazer se eles mudarem.



# EXEMPLO DE ALGORÍTMO



**Início**

    Declarar n, soma como Inteiro

    soma <- 0

    Escrever "Digite um número final: "

    Ler n

    Se n < 1 então

        Escrever "Digite um número maior ou igual a 1."

        Sair

    Fim Se

    Escrever "Sequência:"

    Para i de 1 até n faça

        Escrever i

        soma <- soma + i

    Fim Para

    Escrever "Soma total: ", soma

Fim

```
● ● ●
1  /*
2   PROGRESSÃO ARITMÉTICA (PA)
3
4   Uma PA é uma sequência numérica onde cada termo, a partir do segundo,
5   é obtido somando uma constante chamada RAZÃO ao termo anterior.
6 */
7
8 #include <stdio.h>
9
10 int main() {
11     int n, soma = 0;
12
13     // Entrada
14     printf("Digite um número final: ");
15     scanf("%d", &n);
16
17     // Verificação simples
18     if (n < 1) {
19         printf("Digite um número maior ou igual a 1.\n");
20         return 1;
21     }
22
23     // Mostrar sequência e somar
24     printf("\nSequência:\n");
25     for (int i = 1; i ≤ n; i++) {
26         printf("%d ", i);
27         soma += i;
28     }
29
30     // Resultado
31     printf("\n\nSoma total: %d\n", soma);
32
33     return 0;
34 }
```

```
asm

section .data
prompt db "Digite um número final: ", 0
msg_error db "Digite um número maior ou igual a 1.", 0
msg_sum db "Soma total: ", 0

section .bss
n resb 4
soma resd 1

section .text
global _start

_start:
; Exibir o prompt
mov eax, 4      ; syscall para escrever
mov ebx, 1      ; escrever para a saída padrão
mov ecx, prompt ; endereço do prompt
mov edx, 23     ; comprimento do prompt
int 0x80        ; chamada de sistema

; Ler o número (entrada do usuário)
mov eax, 3      ; syscall para ler
mov ebx, 0      ; ler da entrada padrão
mov ecx, n      ; armazenar em n
mov edx, 4      ; tamanho da entrada (4 bytes para um int)
int 0x80        ; chamada de sistema

; Converter string para número (simples)
mov eax, [n]    ; carregar o número de n
sub eax, '0'    ; converter de ASCII para valor numérico

; Verificar se o número é menor que 1
cmp eax, 1      ; comparar n com 1
jl .error        ; pular para erro se for menor
```

```
; Inicializar soma
mov ebx, 0      ; soma = 0

; Loop para somar a sequência
.loop:
cmp eax, 0      ; verificar se n chegou a 0
je .done        ; se sim, terminar
add ebx, eax    ; soma += n
dec eax         ; decrementa n
jmp .loop       ; repete o loop

.error:
; Exibir mensagem de erro
mov eax, 4      ; syscall para escrever
mov ebx, 1      ; escrever para a saída padrão
mov ecx, msg_error ; endereço da mensagem de erro
mov edx, 30     ; comprimento da mensagem
int 0x80        ; chamada de sistema

; Finalizar o programa
.done:
; Exibir a soma
mov eax, 4      ; syscall para escrever
mov ebx, 1      ; escrever para a saída padrão
mov ecx, msg_sum ; endereço da mensagem "Soma total:"
mov edx, 12     ; comprimento da mensagem
int 0x80        ; chamada de sistema

; Exibir o valor da soma (conversão simples para string)
; (Aqui seria necessário um código adicional para converter o valor numérico em string)

; Finalizar o programa
mov eax, 1      ; syscall para sair
xor ebx, ebx    ; código de saída 0
int 0x80        ; chamada de sistema
```

THANK YOU

[wallacevieira.dev](https://wallacevieira.dev)