

Photo by [iggii](#) on [Unsplash](#)

# Programação concorrente e distribuída

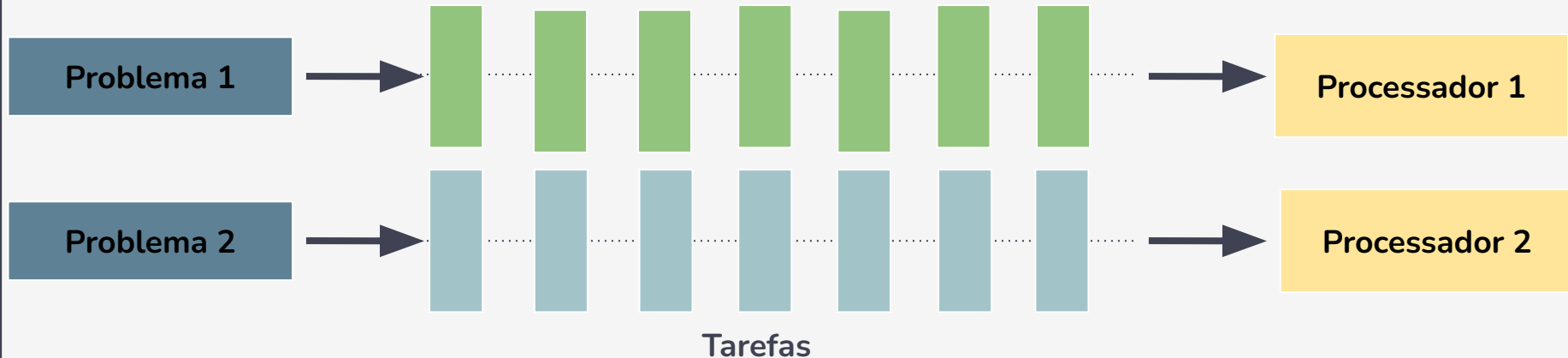
## Aula 3 - Computação paralela

---

Prof. João Robson

# Computação paralela

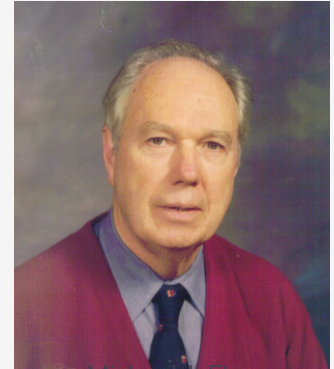
- Estratégia que envolve o uso **simultâneo** de recursos computacionais com o objetivo de resolver tarefas em menor tempo;
- Recursos computacionais podem incluir:
  - Único computador com vários processadores;
  - Vários computadores conectados via rede (computação distribuída);
  - A combinação de ambos.



# Taxonomia de Flynn

---

- Classificação de arquiteturas de computadores proposta por Michael J. Flynn em 1966 e expandida em 1972;
- Metodologia para categorizar formas de operação paralela disponíveis para um processador;
- Baseia-se em duas dimensões: **instruções** (*instructions*) e **dados** (*data*);
- Cada dimensão pode ter dois valores: **único** (*single*) ou **múltiplos** (*multiple*).



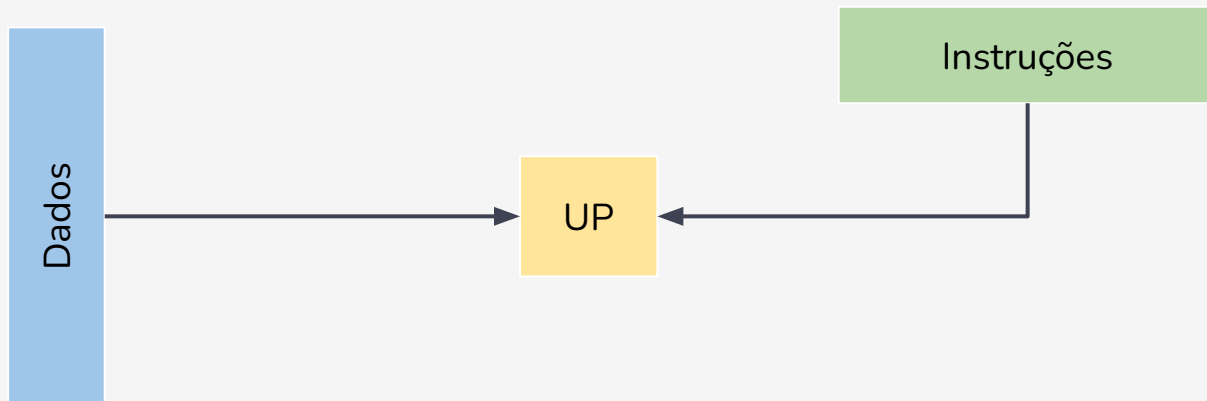
Michael J. Flynn

	Single Data	Multiple Data
Single Instruction	SISD (Single Instruction Single Data)	SIMD (Single Instruction Multiple Data)
Multiple Instruction	MISD (Multiple Instruction Single Data)	MIMD (Multiple Instruction Multiple Data)

# Single Instruction Single Data<sup>1:1</sup>

---

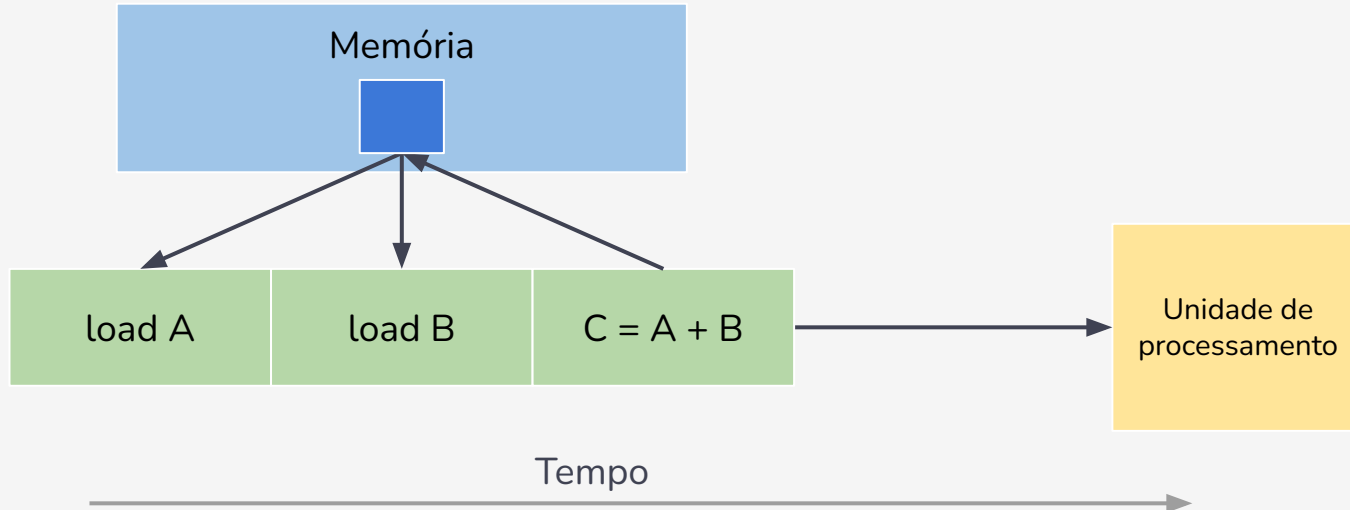
- Arquitetura de computadores com uma única unidade de processamento (núcleo), abreviada como UP;
  - Apenas **uma instrução** processada a cada momento;
  - Apenas **um fluxo de dados** processado a cada momento;
- Instruções são processadas sequencialmente.



# SISD - Exemplo

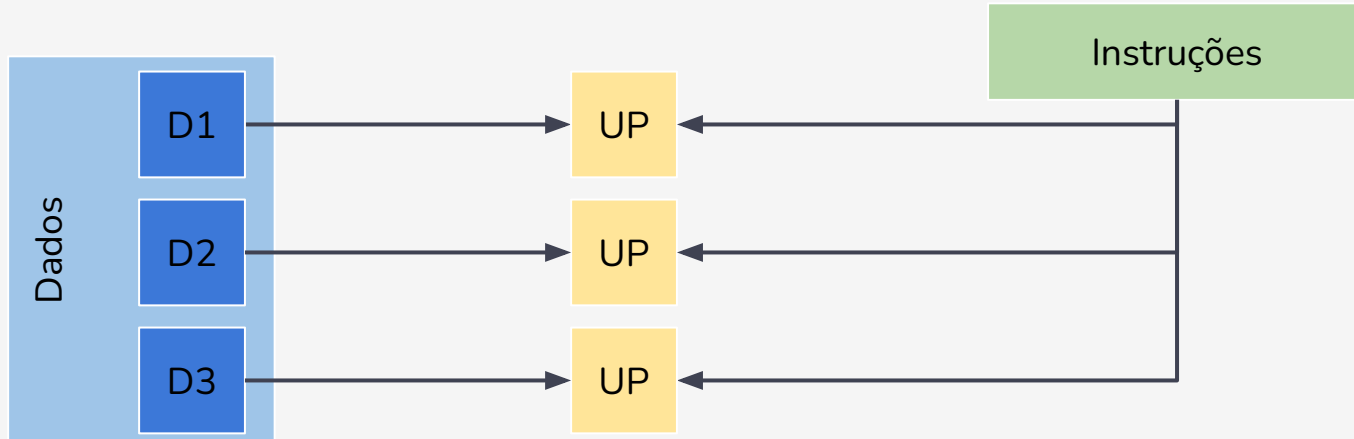
---

load A  
load B  
 $C = A + B$



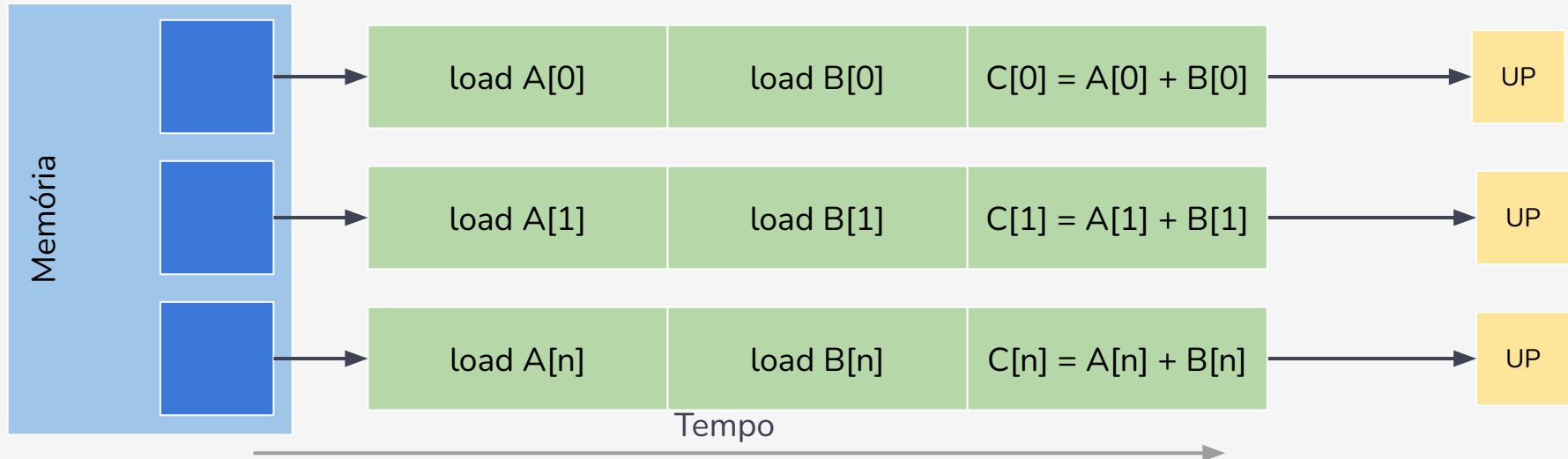
# Single Instruction Multiple Data <sup>N:1</sup>

- Arquitetura de computadores com múltiplas unidade de processamento;
- Cada UP executa a mesma instrução em um determinado momento, mas sobre dados potencialmente distintos;
- Geralmente, usada em dados com alto padrão de regularidade (ex.: processamento de imagens);
- Estratégia geralmente chamada de arquitetura vetorial ou processamento vetorial.



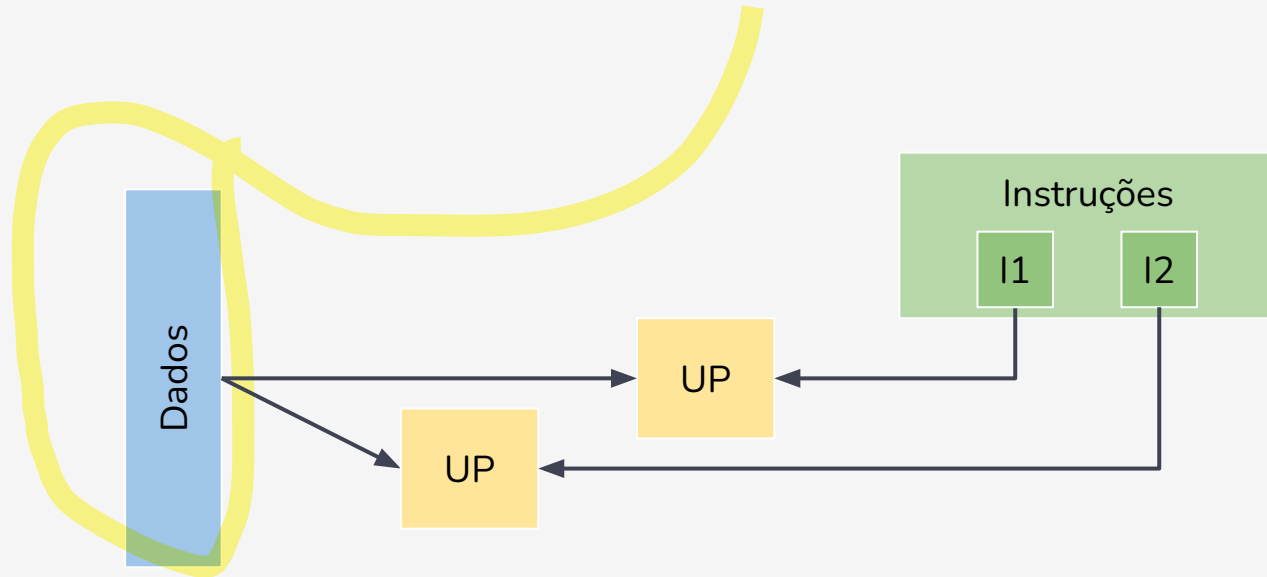
# SIMD - Exemplo

```
load A[i]  
load B[i]  
C[i] = A[i] + B[i]
```



# Multiple Instruction Single Data

- Arquitetura incomum, usada em aplicações que requerem alta **precisão** e **redundância**;
- Instruções distintas aplicadas sobre o **mesmo fluxo de dados**.

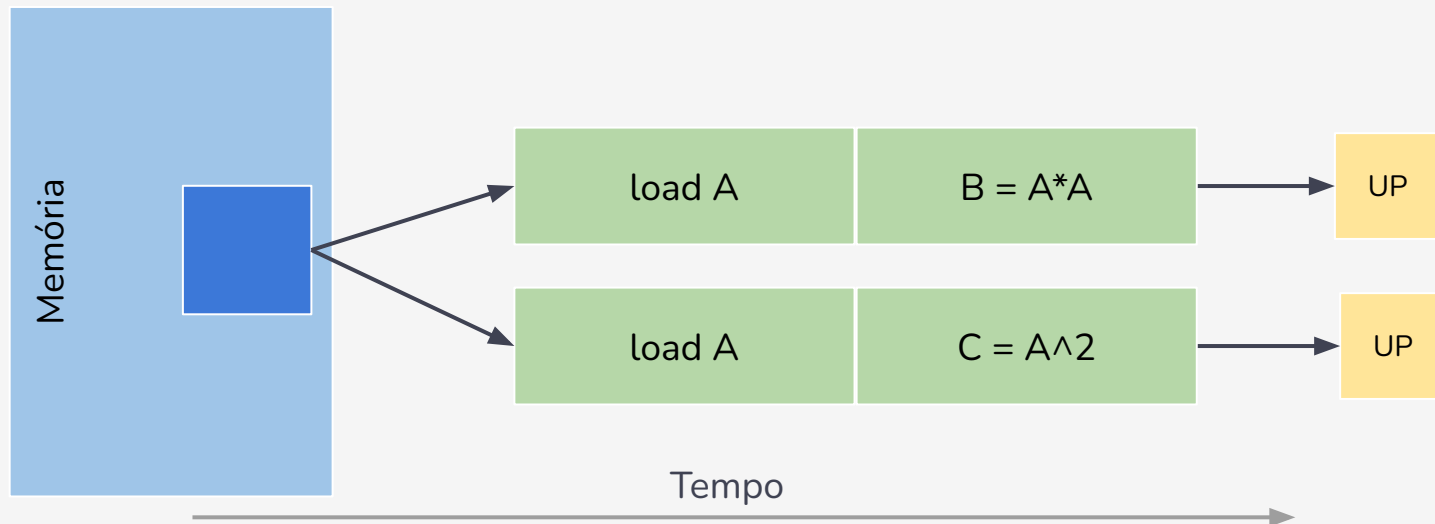




# MISD - Exemplo

load A  
 $B = A * A$

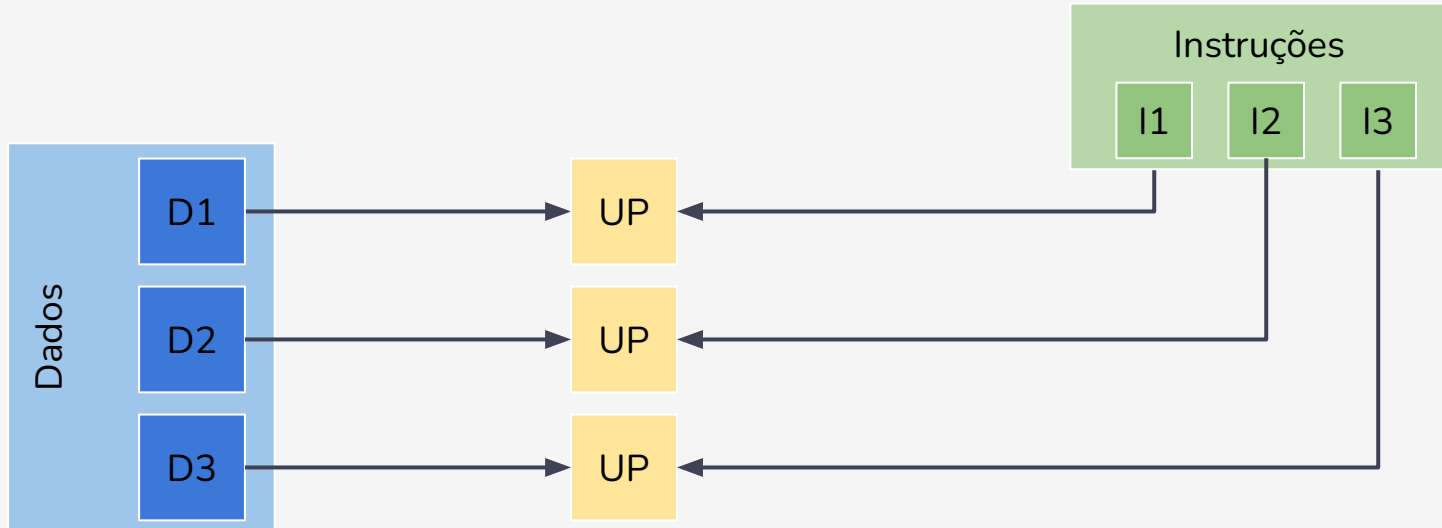
load A  
 $C = A^2$



$N:N$

# Multiple Instruction Multiple Data

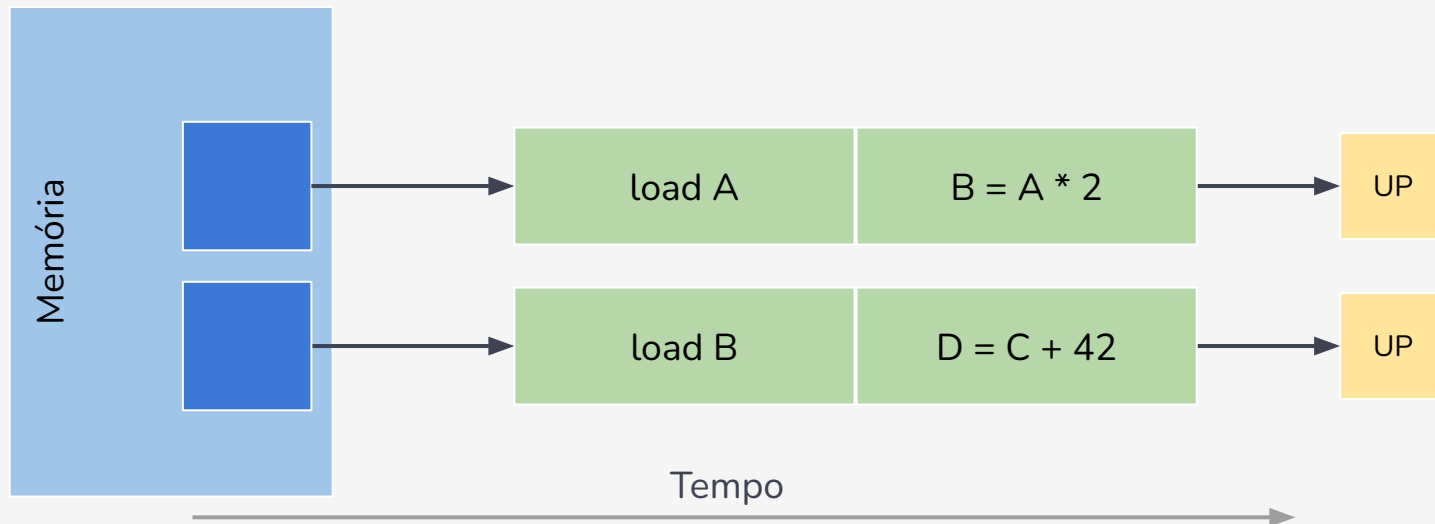
- Arquitetura predominante nos **dias atuais**;
- Cada unidade de processamento opera, no mesmo instante, sobre um fluxo de dados e um conjunto de instruções potencialmente **distintos/independentes**;
- Exemplos: **multiprocessadores e multicomputadores**.



# MIMD - Exemplo

load A  
 $B = A * 2$

load C  
 $D = C + 42$



# Multiprocessadores

---



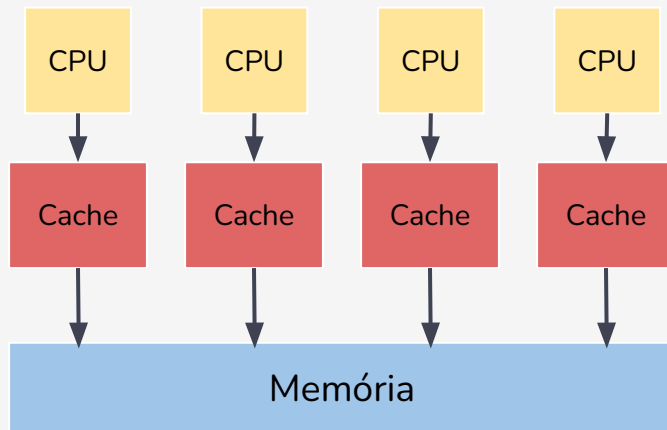
hierarquia de memória e diferença de cada uma

- Baseado em arquitetura MIMD;
- Todos processadores compartilham acesso à memória física;
- Processadores executam instruções de forma independente, mas espaço de endereçamento é compartilhado;
  - Qualquer alteração em determinada posição da memória por determinado processador é acessível para os demais.
- Duas categorias:
  - Uniform Memory Access (UMA);
  - Non-uniform Memory Access (NUMA).

# Multiprocessadores - UMA

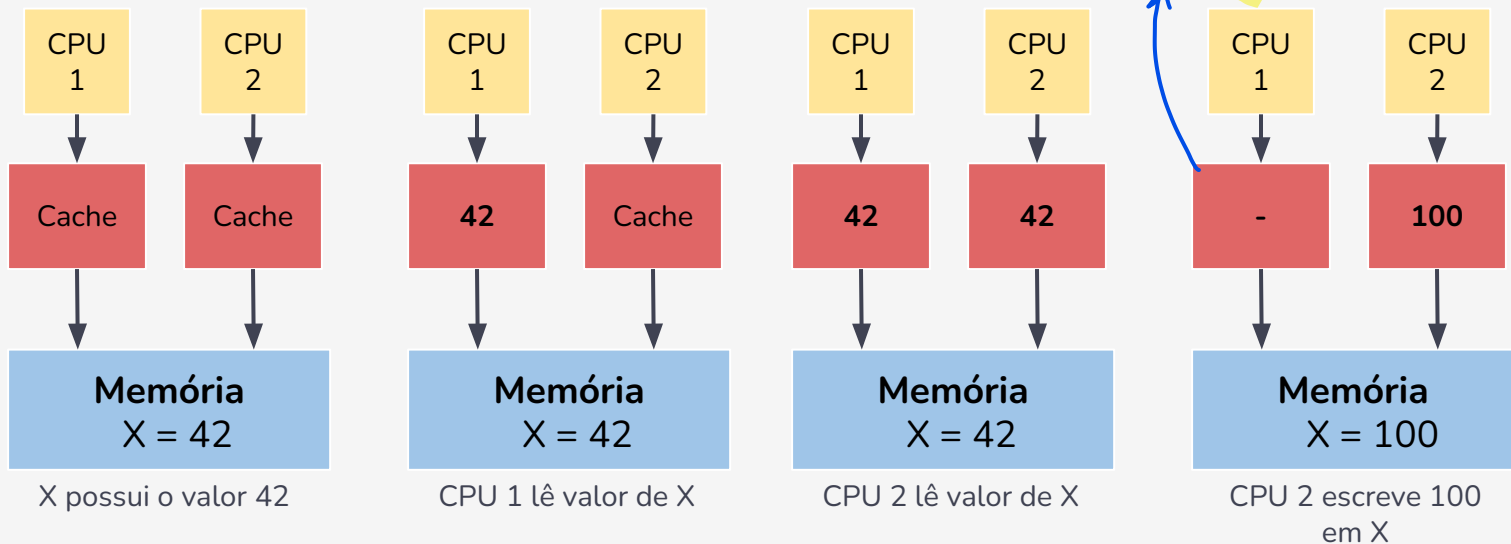
---

- Processadores com acesso idêntico/igualitário à memória;
- Coerência entre caches implementada por meio do *write invalidate protocol*.



# Write Invalidate Protocol

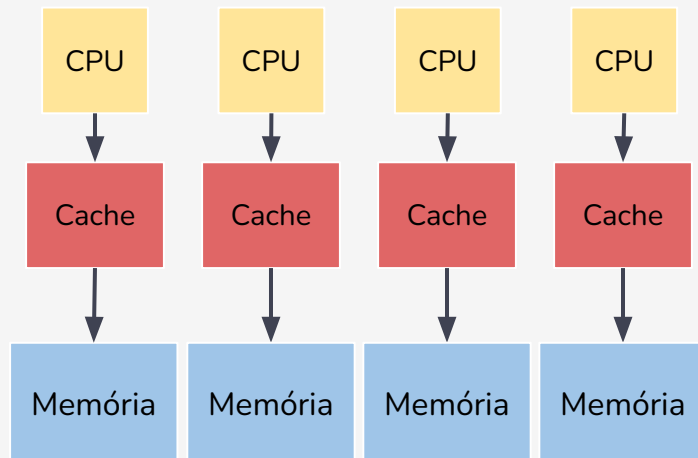
- Antes da escrita na memória, cópias nas caches de outras CPUs são invalidadas;
- Leitura futura desse bloco na memória cache:
  - *Cache miss* -> buscar dado na memória principal;
  - Buscar nos outros caches.



# Multiprocessadores - NUMA

---

- Cada processador tem sua própria memória local;
- Tempo de acesso à memória pode variar entre CPUs diferentes;
- Coerência entre caches implementada por meio do *directory-based protocol*.

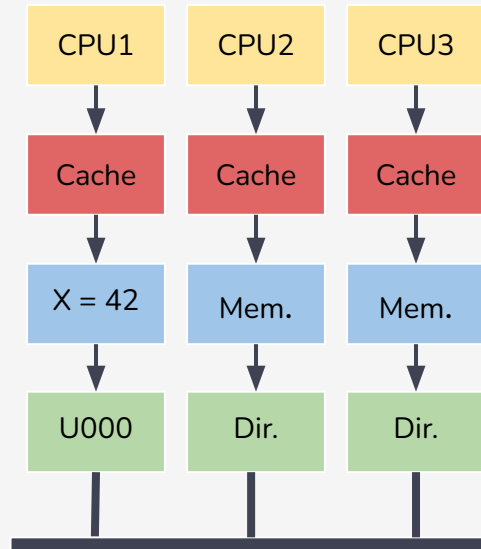


# Directory-based Protocol

compartilhamento de memória entre cpus distintas conectadas por barramento em comum.

- Cada processador possui um diretório com informações sobre o **estados** dos seus **blocos de memória**:
  - **Uncached**: não está em nenhum cache;
  - **Shared**: está na cache de um ou mais processadores e a cópia na memória está correta;
  - **Exclusive**: está na cache de um processador e cópia na memória está incorreta.

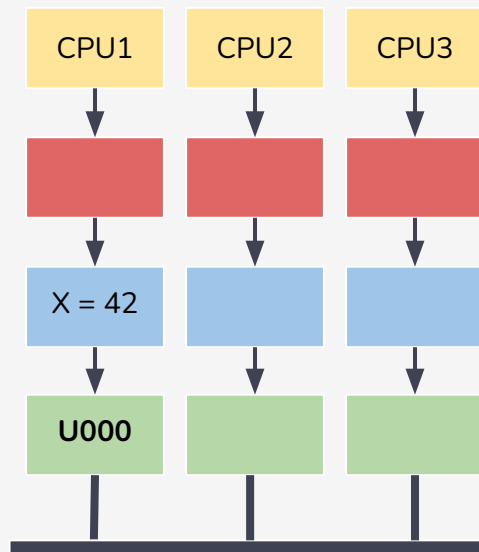
Quais caches contém o valor de X?





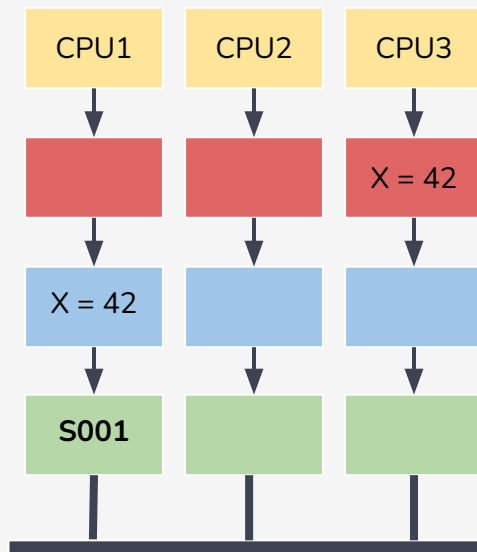
# Directory-based Protocol

X possui valor 42



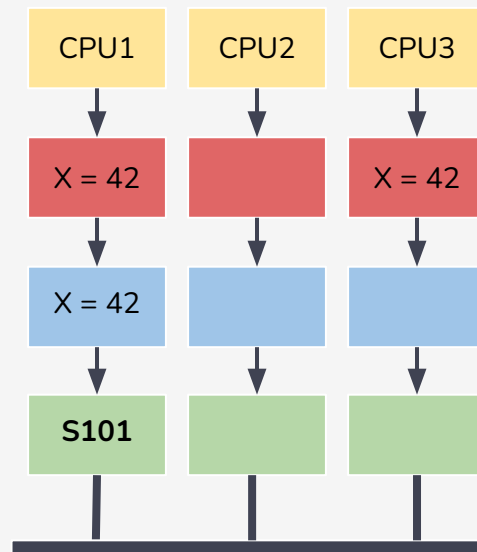
Diretório mostra que X não está em nenhuma cache

CPU3 lê valor de X



CPU3 envia *read miss* para CPU1, que por sua vez atualiza o diretório para *shared* e envia o bloco de dados com X para CPU3

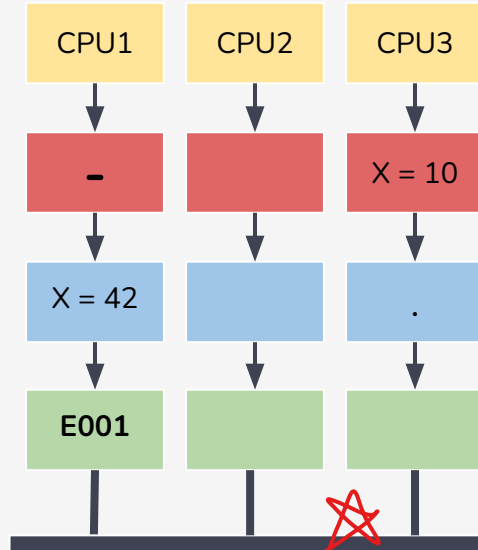
CPU1 lê valor de X



CPU1 não possui X em cache, o bit da CPU1 é atualizado no diretório e o bloco de dados com X é lido pela CPU1

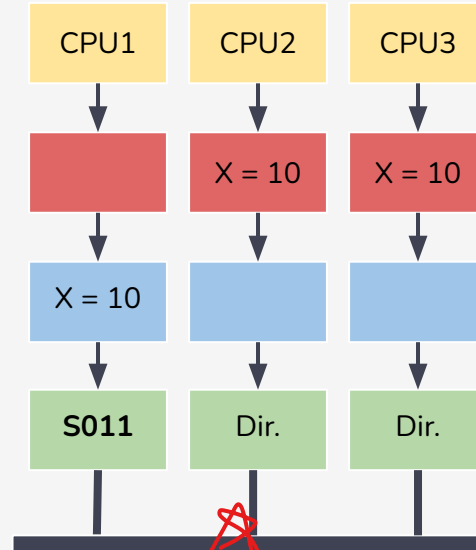
# Directory-based Protocol

CPU3 escreve 10 em X



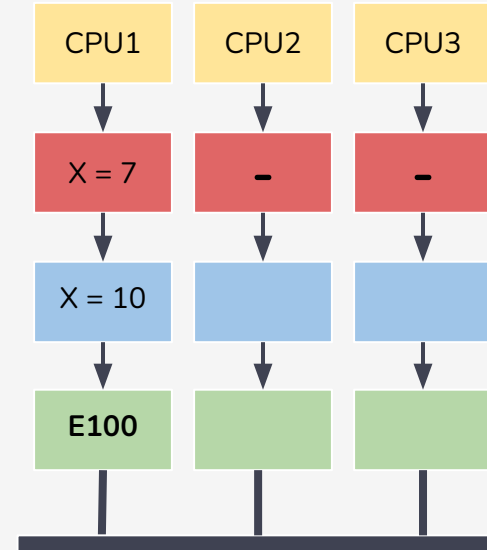
CPU3 envia um **write miss** para CPU1, que invalida sua cópia de X na cache e atualiza diretório para **exclusive**

CPU2 lê valor de X



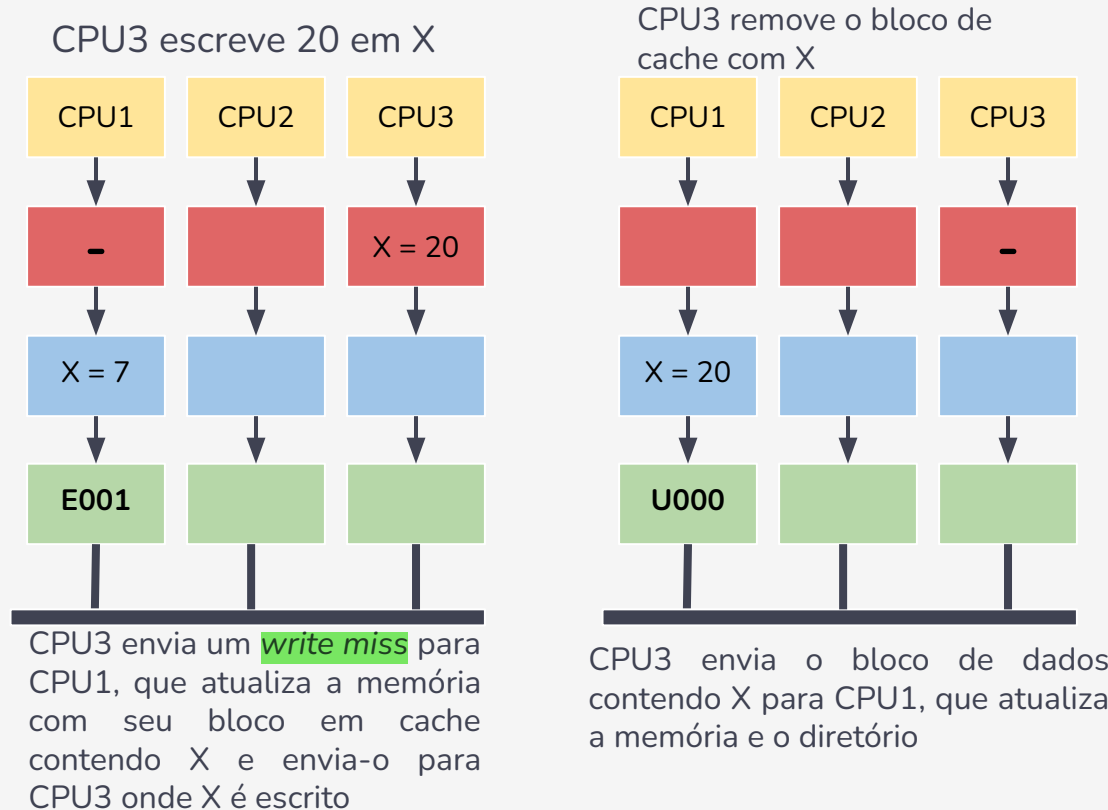
CPU2 envia **read miss** para CPU1, CPU1 pede para CPU3 bloco de dados com X, atualiza memória e o diretório e envia o X para CPU3

CPU1 escreve 7 em X



CPU1 gera um **write miss** de modo a invalidar as cópias de X em CPU2 e CPU3 e atualiza o diretório para **exclusive**

# Directory-based Protocol



# Multiprocessadores - Vantagens e desvantagens

---

- **Vantagens:**

- **Compartilhamento** de dados entre processos é alcançado de forma **simples** e **rápida**;

- **Desvantagens:**

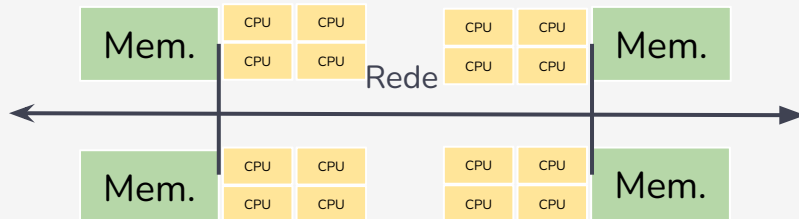
- **Necessidade** de mecanismos de **sincronização**;
- **Pouco escalável** -> quanto mais processadores, maior a contenção de acesso à memória (múltiplos processos ou *threads* acessando mesmo local na memória) -> mais difícil manter coerência entre caches;
- Custo elevado -> quanto mais processadores -> mais caro para produzir computador

# Multicomputadores

- Computadores ligados em rede com memórias físicas independentes;
- Não há coerência entre caches -> alterações em posições memória por um processador não é visível por processadores de outros computadores;
- Duas categorias:
  - *Distributed Multicomputer*



- *Distributed-shared Multicomputer*



# Multicomputadores - vantagens e desvantagens

---

- **Vantagens:**
  - Mais computadores -> mais memória disponível sem preocupação com coerência de caches;
  - Escala de forma mais simples com um custo baixo;
- **Desvantagens:**
  - Necessidade de mecanismos de comunicação entre computadores;
  - Tempo de acesso computadores diferentes não é uniforme;
  - Dificuldade em converter estruturas de dados locais para modelo distribuído.

# Limitações

---

- Computação paralela ainda tem limitações:
  - Nem todo trecho de código é paralelizável;
  - Maior parte dos problemas computacionais reais **não pode ser paralelizado** efetivamente sem aumentar custos de comunicação e coordenação entre processadores.
- Exemplo: colocação de piso por **5 pessoas** em uma casa com **5 quartos**:
  - Quartos de tamanho igual:
    - 1 quarto por pessoa
    - 1 / 5 do tempo para realizar a pintura;
  - 1 quarto com o dobro do tamanho dos quartos restantes:
    - Como dividir?
    - Quarto maior ocupará maior tempo para ter a colocação finalizada.

# Limitações - Lei de Amdahl



- Para avaliar se paralelização de uma tarefa causa alguma melhora e de quanto é esse ganho, podemos usar a **Lei de Amdahl**;
- Ela mede o **speedup (melhora)** máximo alcançado por n processos colaborando na execução de uma tarefa;
- **Considera apenas tempo de execução**;

$$S = \frac{1}{1 - p + \frac{p}{n}}$$

Onde:

$S$  = *speedup* ou melhora

$n$  = número de unidade de processamento

$p$  = porcentagem da tarefa que pode ser paralelizada



# Lei de Amdahl - Exemplo

- Exemplo da casa:
  - 5 pessoas;
  - 5 quartos, sendo que 4 tem 10 m<sup>2</sup> e 1 tem 20 m<sup>2</sup>;
  - Atribuindo 1 pessoa por quarto:
    - 50 m<sup>2</sup> dos 60 m<sup>2</sup> serão preenchidos em paralelo (5/6 da tarefa)
    - $p = 5/6$ ;
    - $1 - p = 1/6$ ;
    - $S = 1 / (1/6 + (5/6)/5) = 1 / (1/6 + 1/6) = 3$

$$S = \frac{1}{\underbrace{1-p}_{\frac{1}{6}} + \frac{p}{n} \left\{ \frac{10}{10} \right\} \frac{1}{11}}$$

Onde:  $\frac{1}{\frac{2}{11}} = \frac{11}{2}$

$S$  = speedup ou melhora

$n$  = número de unidade de processamento

$p$  = porcentagem da tarefa que pode ser paralelizada

# Lei de Amdahl - Exemplo

---

- Exemplo da casa:
  - 10 pessoas;
  - 10 quartos, sendo que 9 tem 10 m<sup>2</sup> e 1 tem 20 m<sup>2</sup>;
  - Atribuindo 1 pessoas por quarto:
    - $S = 1 / (1/11 + 1/11) = 5.5$
- Mesmo com pequeno desequilíbrio (1 quarto com o dobro do tamanho), 10 pessoas só conseguem cerca de metade do ganho que se esperaria de forma ingênua (10 vezes);
- O que isso representa em um contexto de multiprocessadores?
  - Em um contexto onde há 10 unidades de processamento e conseguimos paralelizar somente 90% do problema, conseguimos um ganho de 5 vezes e não de 10;
  - 10% não paralelizável -> corta uso da máquina pela metade;
  - Aplicar paralelismo nos 10% restante pode ser bastante complicado:
    - **Envolve coordenação e sincronia.**

Handwritten red calculation:  $\frac{100}{20} = 5.5$

# Threads - Exercícios

---

- Avalie o ganho dos exercícios abaixo usando *threads*:
  - Calcule as 4 operações aritméticas básicas com 2 números;
  - Identifique o maior valor de uma lista;
  - Calcule a soma dos números primos dentro de um intervalo determinado.

# Referências

---

- Tanenbaum, Andrew S. Sistemas operacionais modernos / Andrew S. Tanenbaum, Herbert Bos; tradução Jorge Ritter; revisão técnica Raphael Y. de Camargo. – 4. ed. – São Paulo: Pearson Education do Brasil, 2016.
- SILVA, Cleomar Pereira da. Computação de Alto Desempenho com Placas Gráficas para Acelerar o Processamento da Teoria do Funcional da Densidade. Disponível em: <https://www.maxwell.vrac.puc-rio.br/colecao.php?strSecao=resultado&nrSeq=16578@1&msg=28#>
- MENOTTI, Ricardo; DIAS, Maurício Acconcia; GUARDIA, Helio Crestana. Programação paralela: das threads aos FPGAs. Disponível em: <https://github.com/menotti/pp>
- BALANIUK, Remis. Computação paralela.