Arquiteturas Paralelas e Distribuídas

Taxonomia de arquiteturas paralelas: SISD, SIMD, MIMD e SPMD

Eduardo Furlan Miranda

Baseado em: ROCHA, R. Fundamentos de Programação Paralela e Distribuída. DCC-FCUP, 2007.

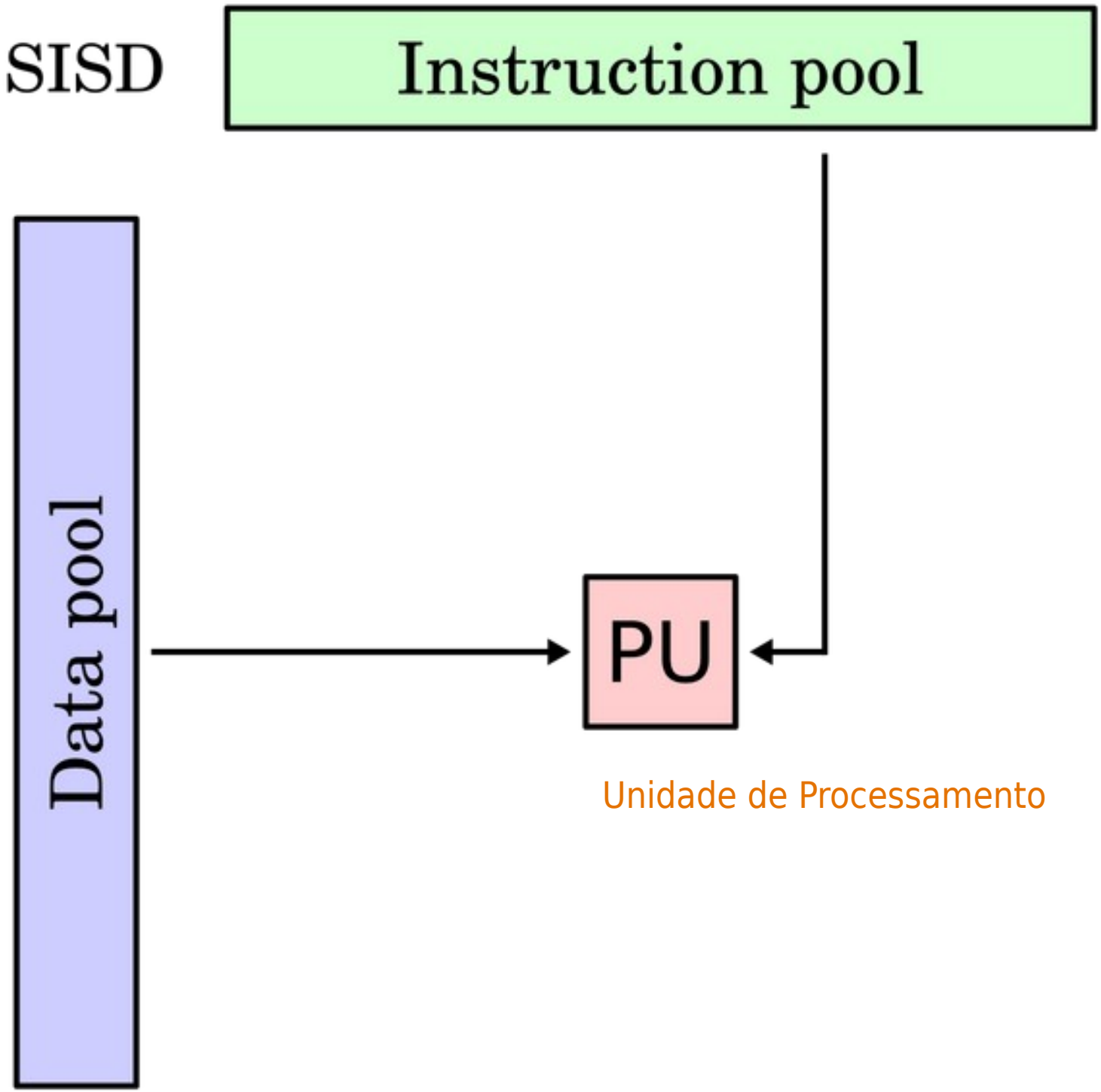
Taxonomia de Flynn (1966)

- Metodologia para classificar a arquitetura de um computador ou conjunto de computadores

	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*

SISD – Single Instruction Single Data

- Computador sequencial que não explora paralelismo em nenhum dos fluxos de instruções ou dados
  - o processador executa uma única instrução de cada vez
  - o processador opera em um único conjunto de dados por vez
  - Ex: Arduino - o Atmega328P executa uma instrução por vez operando em um dado por vez


SISD
Instruction pool
Data pool
PU
Unidade de Processamento

SIMD – Single Instruction Multiple Data

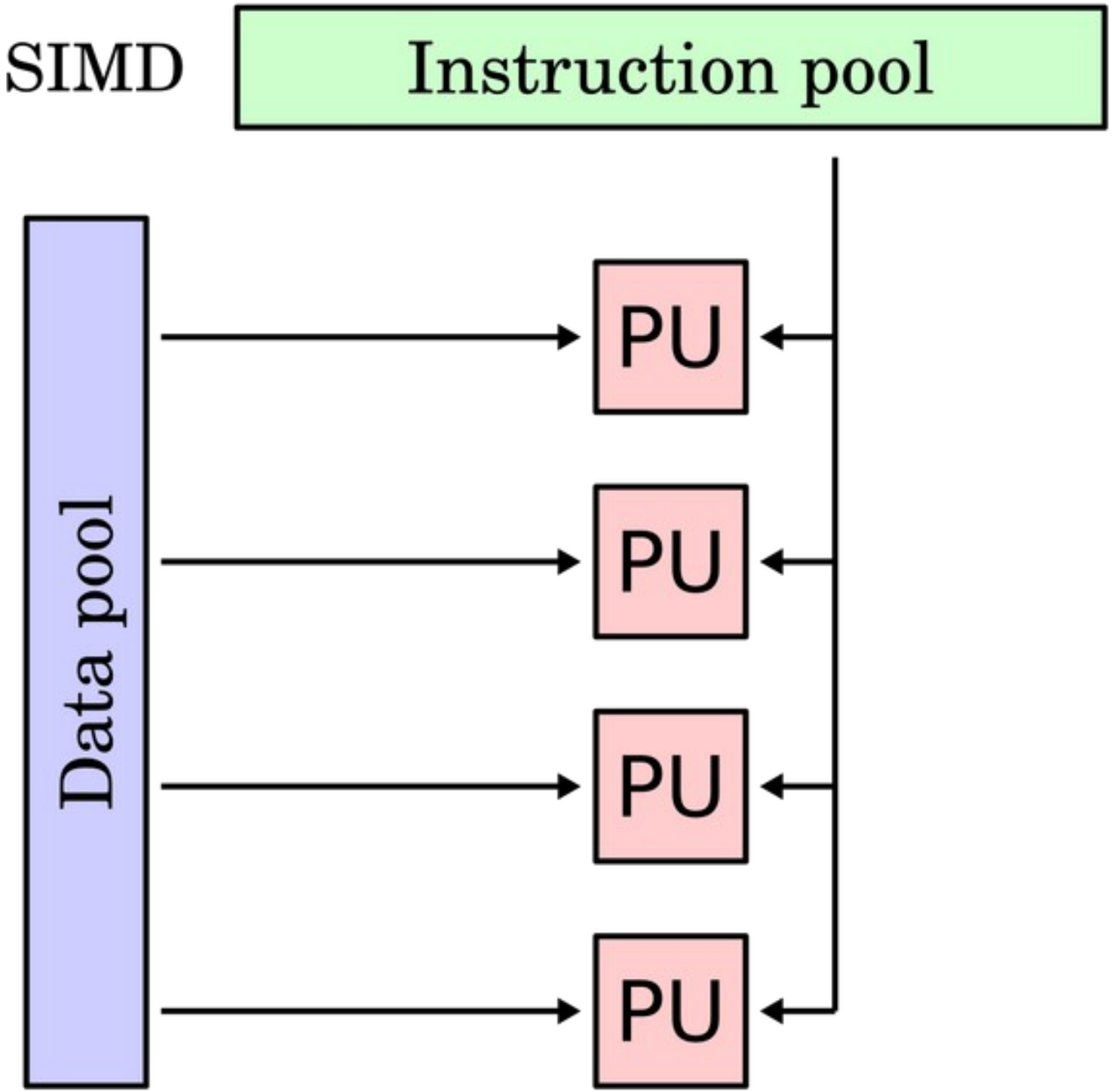
- Uma única instrução é aplicada simultaneamente a vários fluxos de dados diferentes

- As instruções podem ser executadas
  - sequencialmente
    - como por pipelining
      - o processador divide internamente uma instrução em várias etapas
        - permitindo executar várias instruções ao mesmo tempo
          - em diferentes estágios de processamento
  - em paralelo
    - por várias unidades funcionais

- O artigo de Flynn de 1972 subdividiu o SIMD em três categorias adicionais

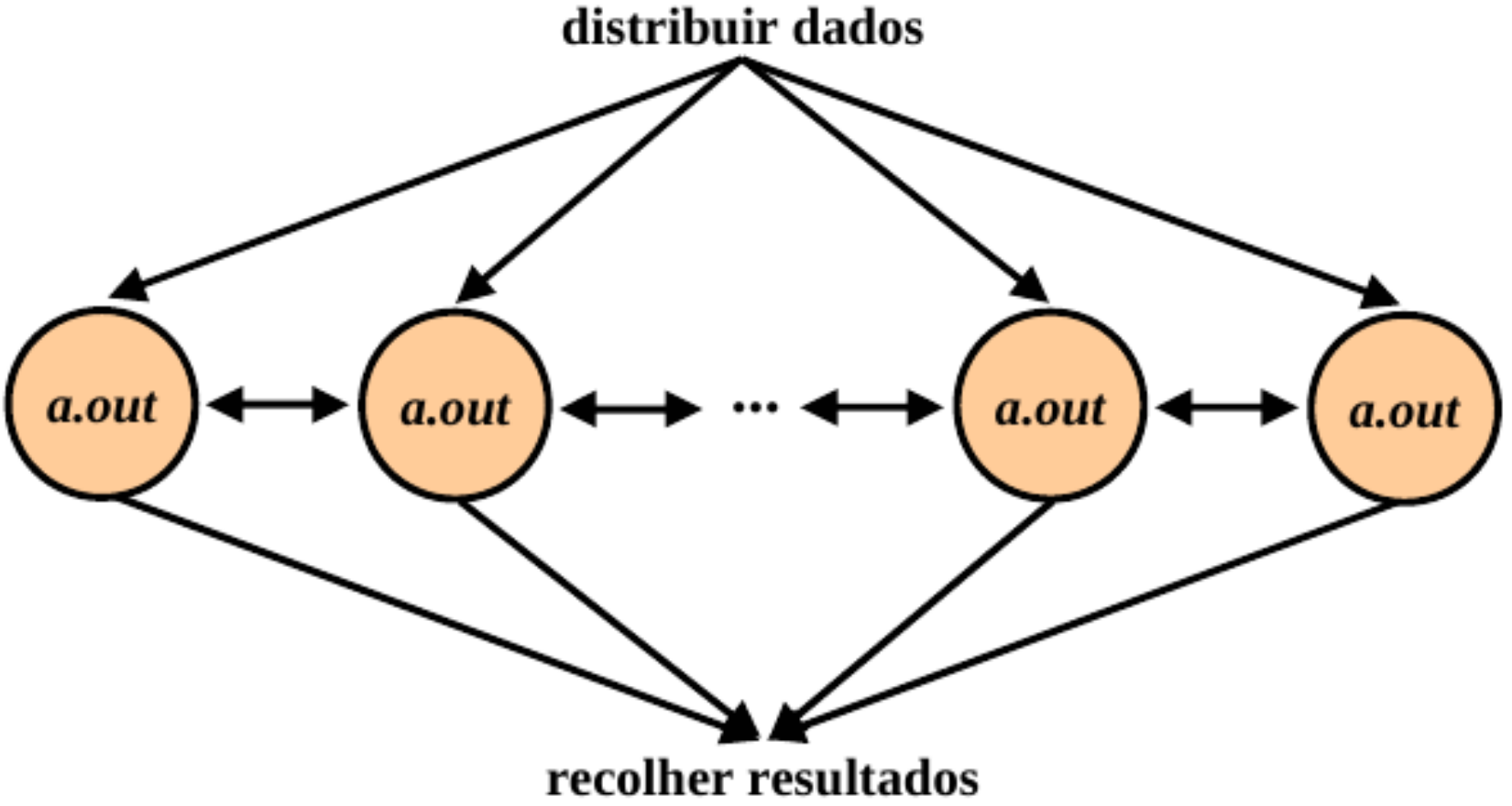
SIMD - categorias - processadores

- “Array”
  - mesma instrução, mas cada processador tem sua própria memória e arquivo de registro separados e distintos
- “Pipeline”
  - mesma instrução, mas leem dados de um recurso central, cada um processa fragmentos desses dados, e gravam os resultados no mesmo recurso central
- “Associative”
  - mesma instrução, mas cada processador executa ou ignora baseado em dados de uma memória local


SIMD
Instruction pool
Data pool
PU
PU
PU
PU

Paradigma SPMD

- Todos os processos executam o mesmo programa (executável) mas sobre diferentes partes dos dados
  - Outros nomes: *Geometric Parallelism* e *Data Parallelism*



SPMD

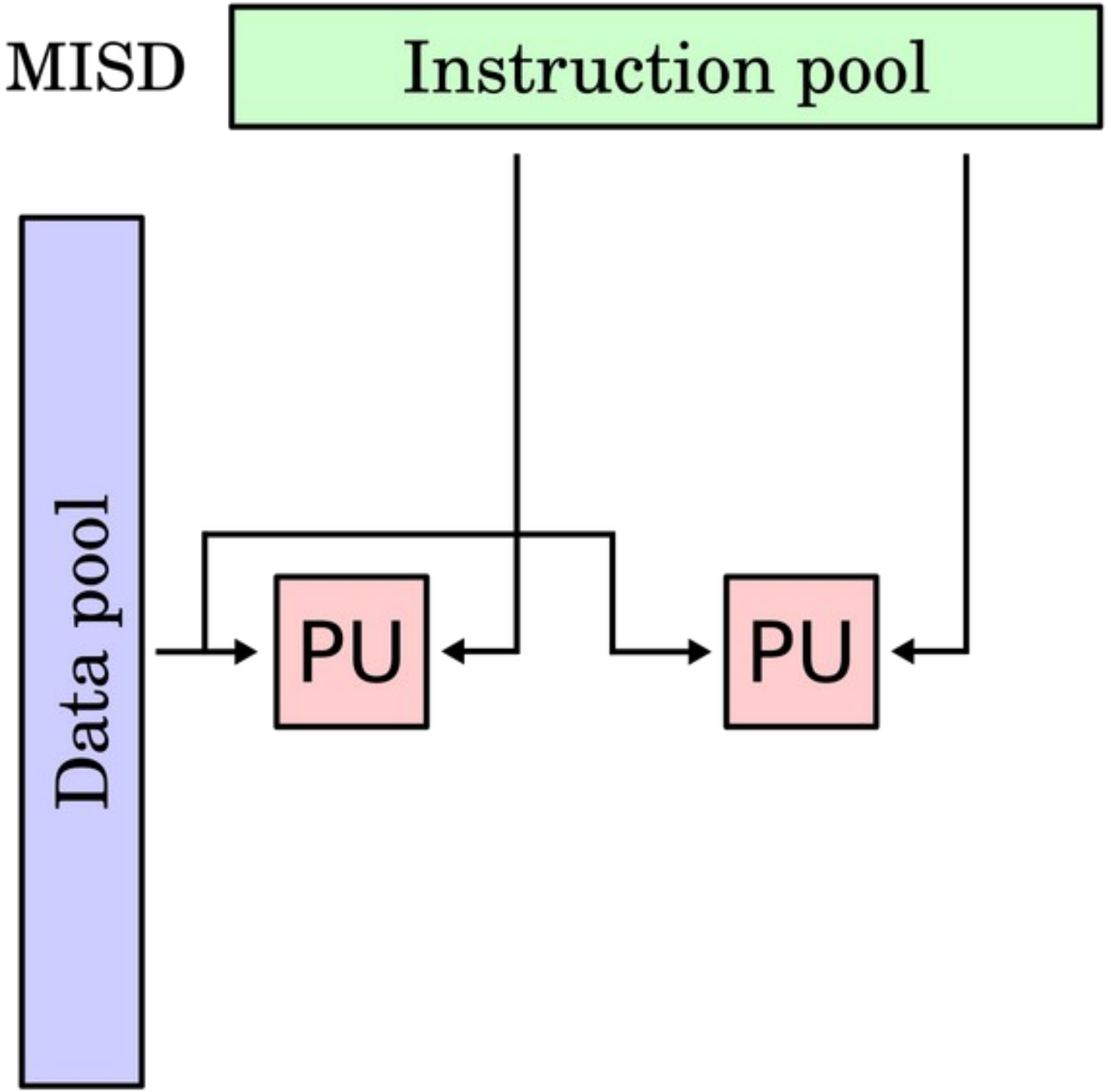
- Tipicamente, os dados são bem distribuídos (mesma quantidade e regularidade) e o padrão de comunicação é bem definido (estruturado, estático e local)
  - Dados ou são lidos individualmente por cada processo ou
    - um dos processos é o responsável por ler todos os dados e depois distribui-los pelos restantes processos
  - Os processos comunicam quase sempre com processos vizinhos e apenas esporadicamente existem pontos de sincronização global

SPMD

- Como os dados são bem distribuídos e o padrão de comunicação é bem definido,
  - Este paradigma consegue bons desempenhos e um elevado grau de escalabilidade
- Sensível a falhas
  - A perda de um processador causa parada no próximo ponto de sincronização global

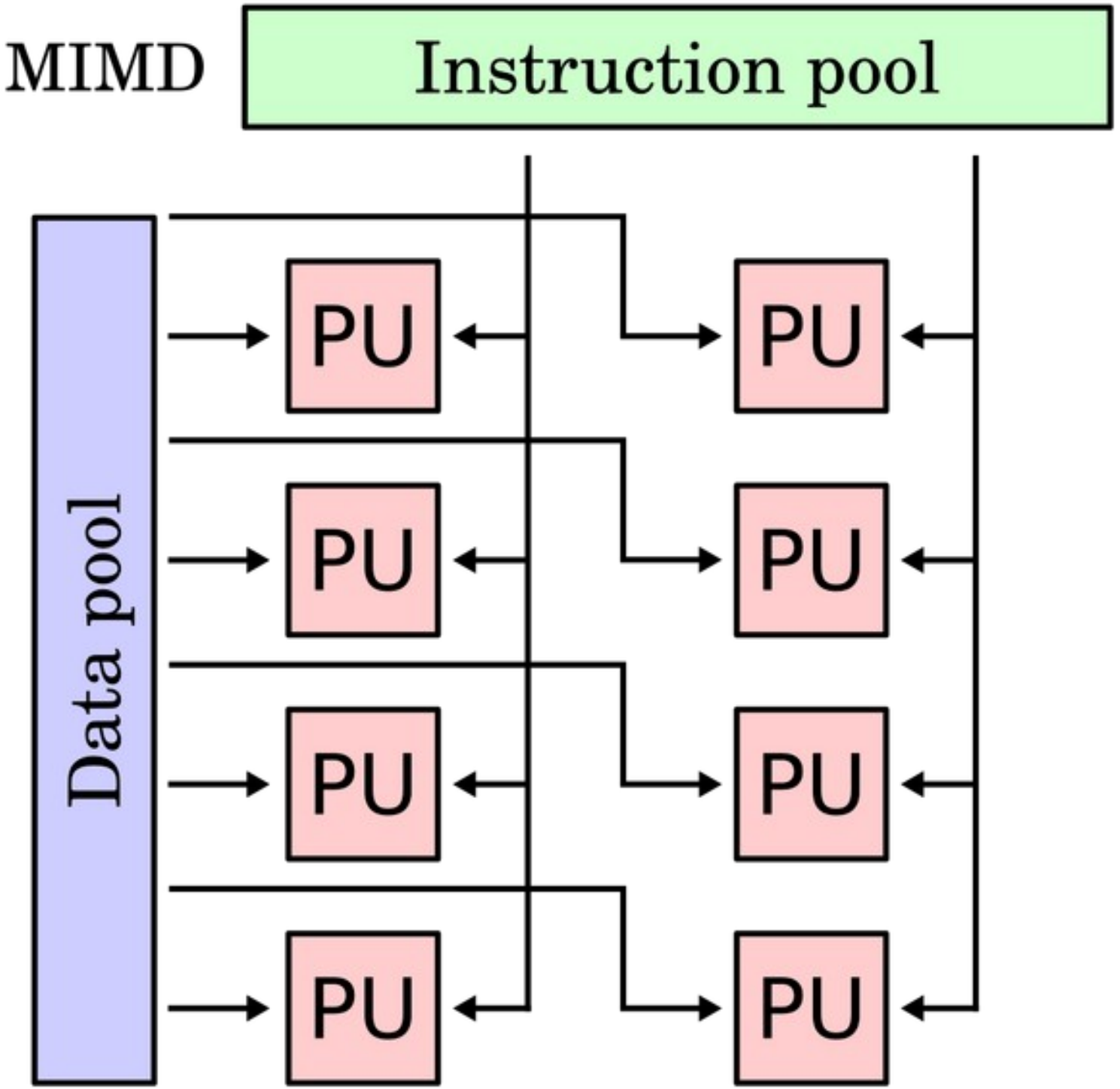
MISD – Multiple Instruction Single Data

- Arquitetura paralela para problemas específicos caracterizados por um alto padrão de regularidade funcional (ex. processamento de sinal)
  - Uma pipeline de unidades de processamento independentes
    - que operam sobre um mesmo fluxo de dados
      - enviando os resultados de uma unidade para a próxima
  - Cada unidade de processamento executa instruções diferentes a cada momento


MISD
Instruction pool
Data pool
PU
PU

MIMD – Multiple Instruction Multiple Data

- Arquitetura paralela predominante atualmente
  - Cada unidade de processamento executa instruções diferentes a cada momento
  - Cada unidade de processamento pode operar sobre um fluxo de dados diferente
  - Ex.: multiprocessors e multicomputers, usando memória compartilhada ou distribuída

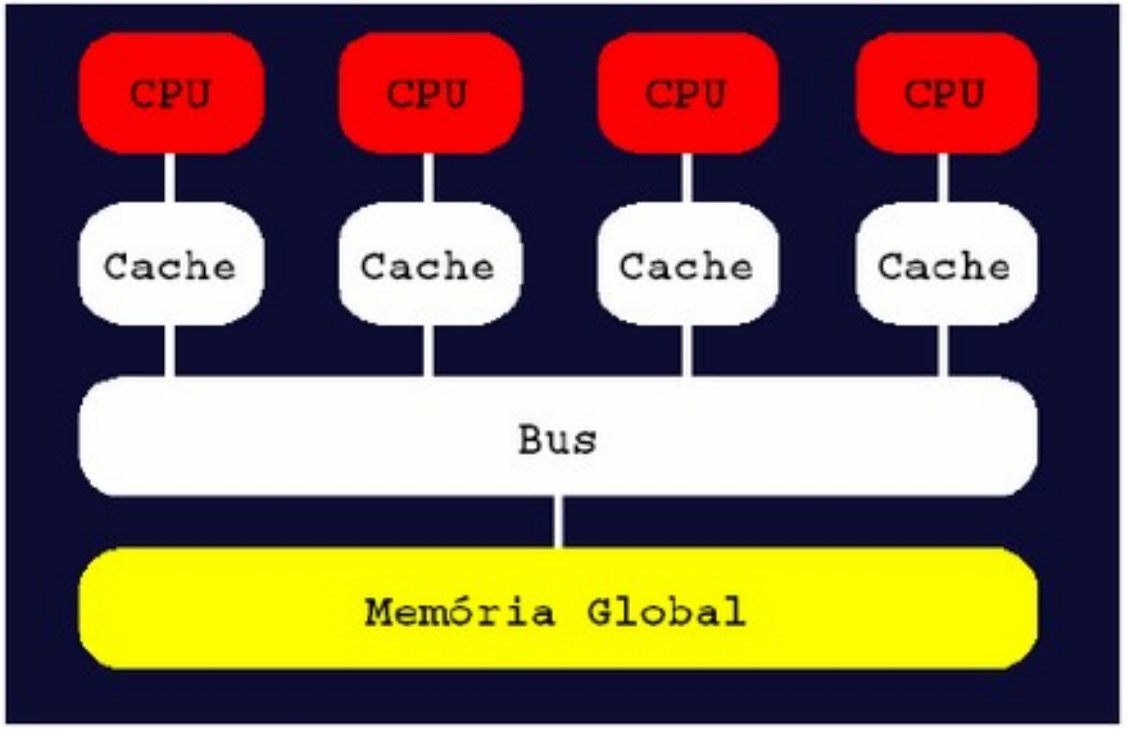

MIMD
Instruction pool
Data pool
PU
PU
PU
PU
PU
PU
PU
PU

Arquiteturas MIMD

	Multiprocessor		*Multicomputer*	
	CC-UMA	*CC-NUMA*	*Distributed*	*Distributed-Shared*
Escalabilidade	10s de CPUs	100s de CPUs	1000s de CPUs	
Comunicação	Segmentos memória partilhada *Threads* *OpenMP* (*MPI*)		*MPI*	*MPI/Threads* *MPI/OpenMP*

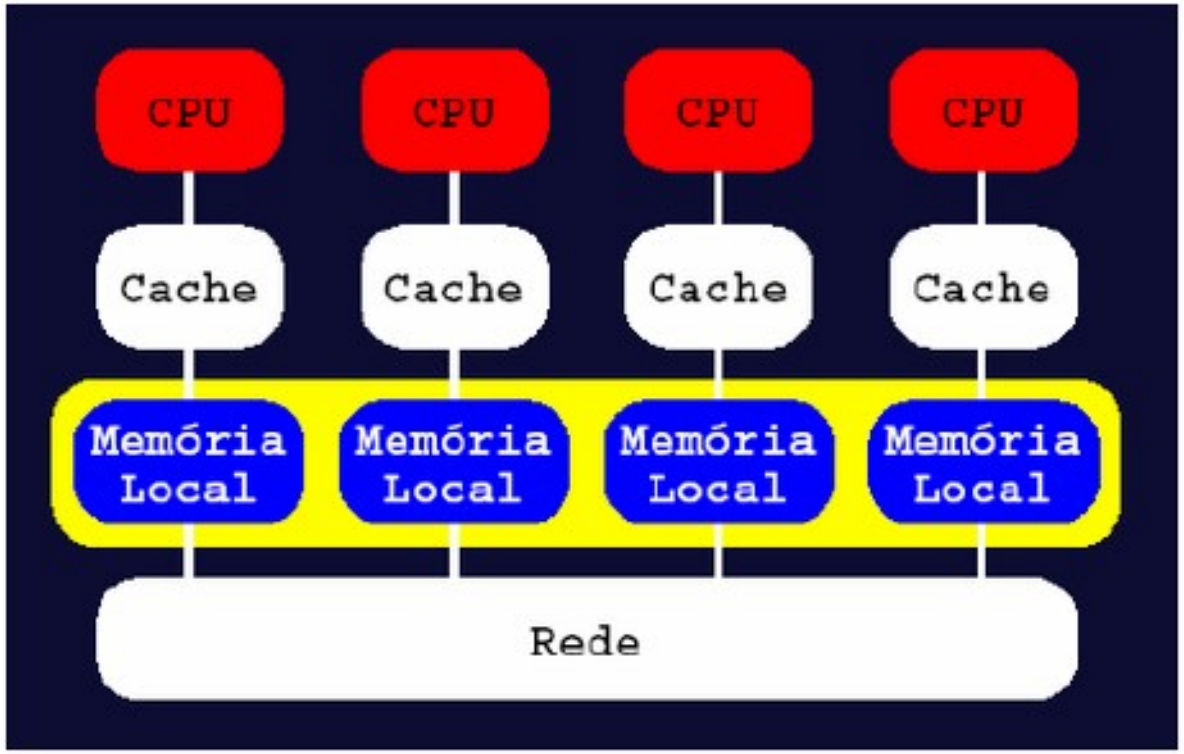
Uniform Memory Access Multiprocessor (UMA)

- Todos os processadores têm tempos de acesso idênticos a toda a memória
- A coerência das caches é implementada pelo hardware



Non-Uniform Memory Access Multiprocessor (NUMA)

- Os processadores têm tempos de acesso diferentes a diferentes áreas da memória
  - Cache Coherent NUMA (CC-NUMA)
    - Coerência das caches implementado pelo hardware

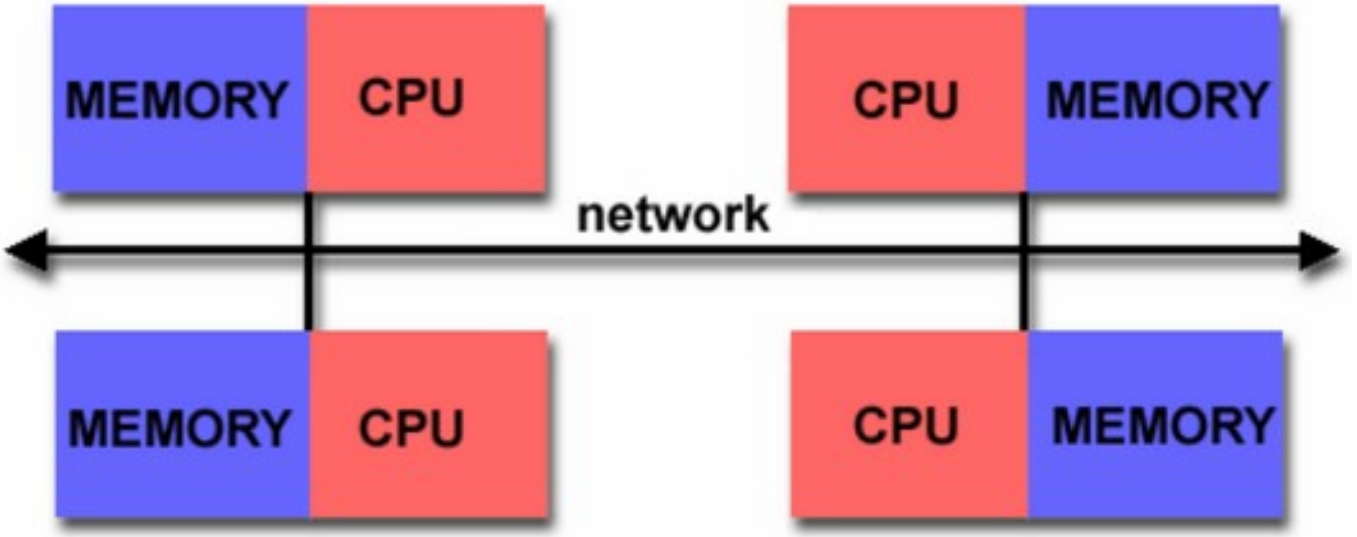


Multiprocessors

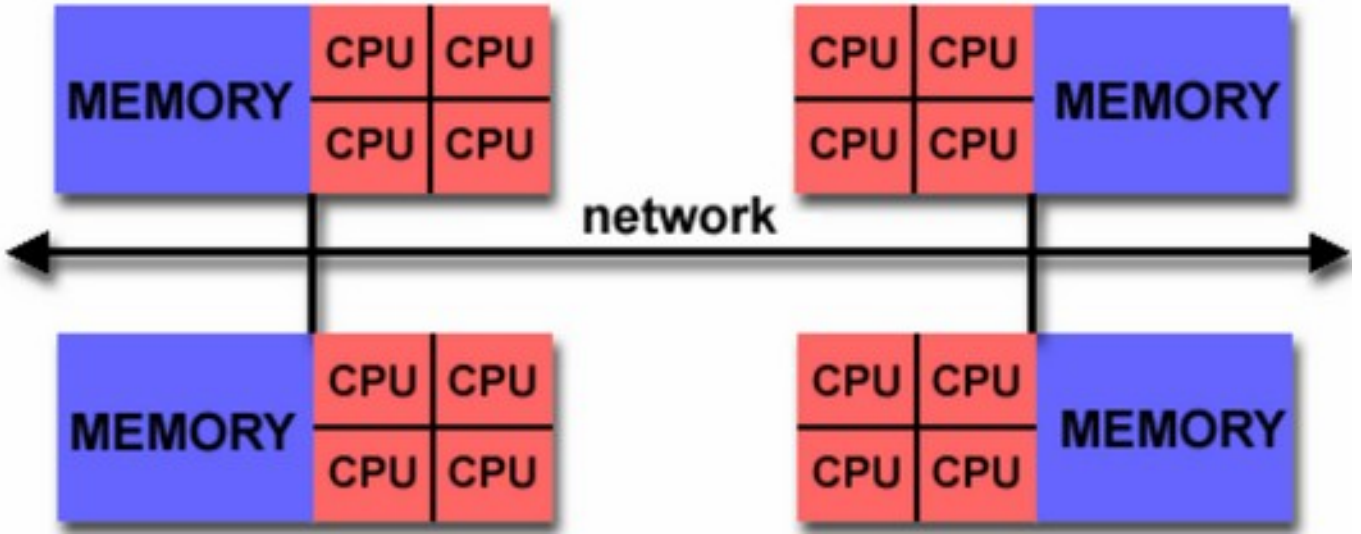
- Existem duas grandes classes de multiprocessors
  - Uniform Memory Access Multiprocessor (UMA)
    - ou Cache Coherent Uniform Memory Access Multiprocessor (CC-UMA)
    - ou Symmetrical Multiprocessor (SMP)
    - ou Centralized Multiprocessor
  - Non-Uniform Memory Access Multiprocessor (NUMA)
    - ou Distributed Multiprocessor

Multicomputers

Distributed Multicomputer



Distributed-Shared Multicomputer



Divisões adicionais

- Embora não faça parte do trabalho de Flynn, a categoria MIMD pode ser dividida:
  - Single program, multiple data streams (SPMD)
    - Vários processadores autônomos executando simultaneamente o mesmo programa em dados diferentes
  - Multiple programs, multiple data streams (MPMD)
    - Vários processadores autônomos operando simultaneamente pelo menos dois programas independentes
      - Em contextos de HPC, tais sistemas frequentemente escolhem um nó para ser o "host" que executa um programa que terceiriza dados para todos os outros nós que executam um segundo programa