

# Parallel Computing Paradigms



João Luís Ferreira Sobral  
[www.di.uminho.pt/~jls](http://www.di.uminho.pt/~jls)  
[jls@di.uminho.pt](mailto:jls@di.uminho.pt)

Web: Elearning

# Parallel Computing Paradigms

---

**At the end of the course, students should be able to:**

- **Design** and **optimise** parallel applications that can **efficiently** run on a wide range of parallel computing platforms
  - Identify/develop parallel applications using well-known **parallelism patterns**
  - Identify limitations of current **parallel programming paradigms** and languages
  - Identify limitations and workarounds to **performance scalability**

**Program (short version)**

- **Programming models (or paradigms), abstractions and languages**
  - Shared vs distributed memory models
  - Tasks, threads, processes and distributed objects
  - Mechanisms to express concurrency/parallelism
  - Tasks distribution among resources
- **Design of parallel applications**
  - Design phases: partition, communication, agglomeration, mapping
  - Typical parallelism patterns: *pipelining, farming, heartbeat e divide & conquer*
  - Measurement, analysis and optimisation of parallel applications
  - Scalability analysis: cost / benefit of parallelism and its quantification (metrics)
  - Granularity control: computation versus communication
  - Load distribution and data partition

# Parallel Computing Paradigms

---

## Final grade

- One project: development of typical parallel applications (3 phases)
  1. On shared memory (35%)
  2. On distributed memory (50%)
  3. On hybrid SM+DM or advanced language (15%)

## Requirements

- Basic programming in C (and Java);
- Knowledge of computing systems and architecture

## Bibliography (base)

- Slides
- M. Quinn. *Parallel programming in C with C and OpenMP*, McGraw Hill, 2003
- I. Foster. *Designing and Building Parallel Programs*, Addison-Wesley, 1995.

## Bibliography (additional)

- M. McCool, J. Reinders, A. Robison, *Structured Parallel Programming: Patterns for Efficient Computation*, Morgan Kaufmann, 2012
- C. Lin, L. Snyder, *Principles of parallel programming*, Pearson international, 2009
- F. Buschmann, D. Schmidt, M. Stal, H. Rohnert, *Pattern-oriented Software Architecture Vol 2: Patterns for Concurrent and Networked Objects*, John Wiley and Sons Ltd, 2000.

# Parallel Computing Paradigms

---

- Introdução
  - Evolução das arquiteturas de computadores
  - Níveis e tipos de paralelismo
  - Paradigmas vs linguagens de programação
  - Linguagens baseadas em: diretivas (OpenMP), extensões à linguagem (Java) vs bibliotecas (TBB)
- Programação baseada no paradigma de memória partilhada
  - OpenMP
  - Java threads / cilk
  - Intel *Thread Building Blocks*
  - Modelos de consistência de memória
  - Medição de otimização de desempenho em memória partilhada
- Programação baseada no paradigma de memória distribuída
  - Processos comunicantes (MPI)
  - Objetos distribuídos, cliente/servidor
  - Desenho de aplicações paralelas
  - Padrões comuns de computação
  - Medição e otimização de desempenho em memória distribuída
- Programação baseada em modelos híbridos
  - Modelo híbrido MPI + OpenMP
  - Modelos de memória virtual partilhada (*virtual shared memory*)
  - Modelo global com partições (*partitioned global address space*)
- Modelos e paradigmas de computação paralela
  - classificação das várias linguagens para computação paralela

# Parallel Computing Paradigms

## Evolução do desempenho das arquiteturas de computadores nos últimos 40 anos

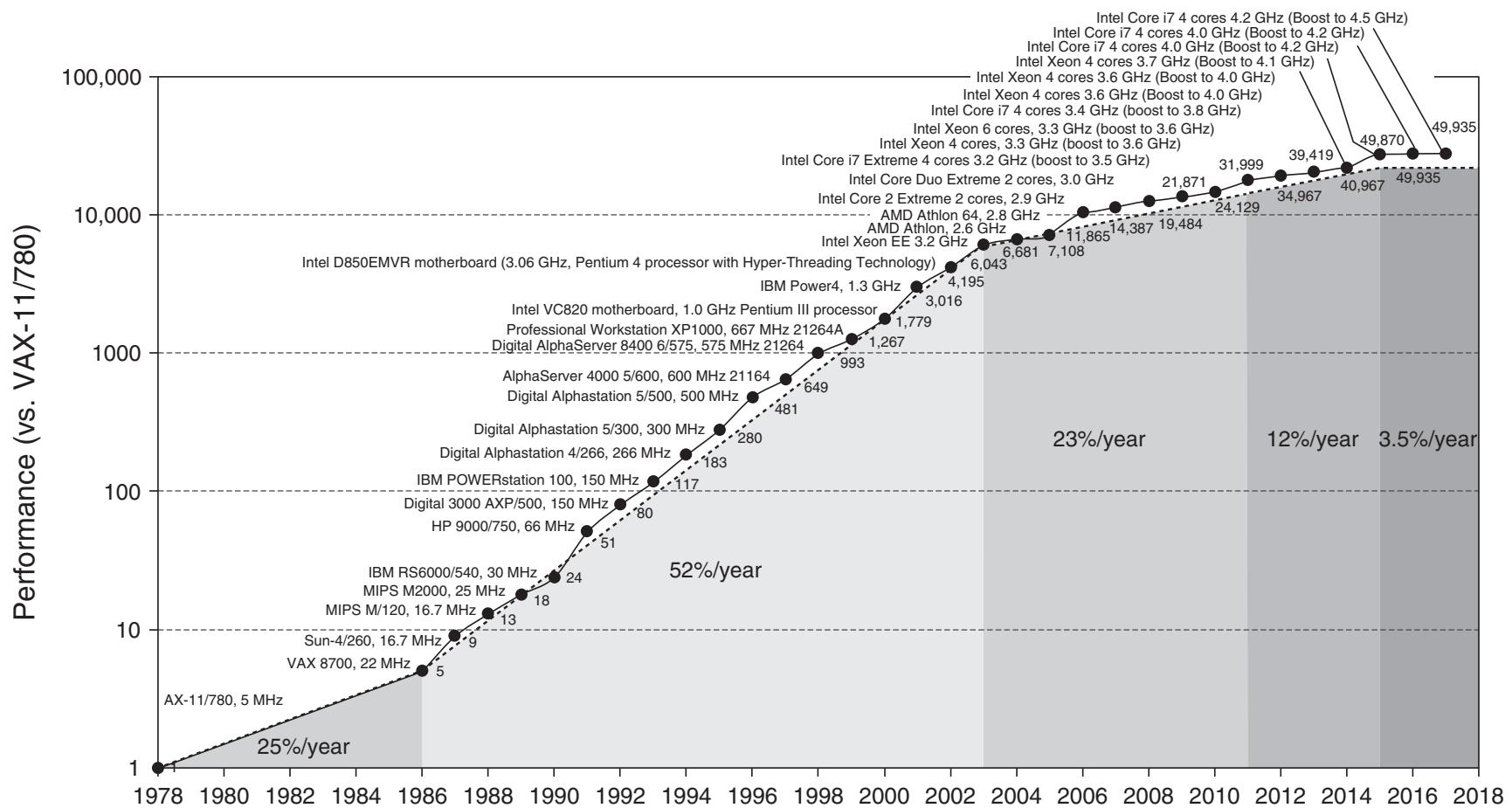
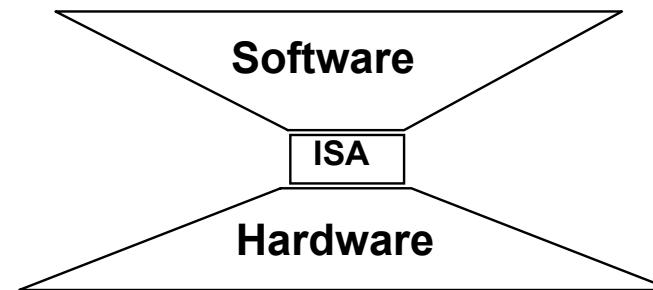


Figura de "Computer Architecture", Patterson & Hennessy 2018

# Relação entre as métricas (slide de Arq. Comp)

	Tem impacto em		
	#I	CPI	Tcc
Algoritmo	S	(S)	
Linguagem	S	(S)	
Compilador	S	(S)	
Conj. Instruções (ISA)	S	S	
Organização		S	S
Tecnologia			S

$$T_{exe} = \#I \times CPI \times T_{cc}$$



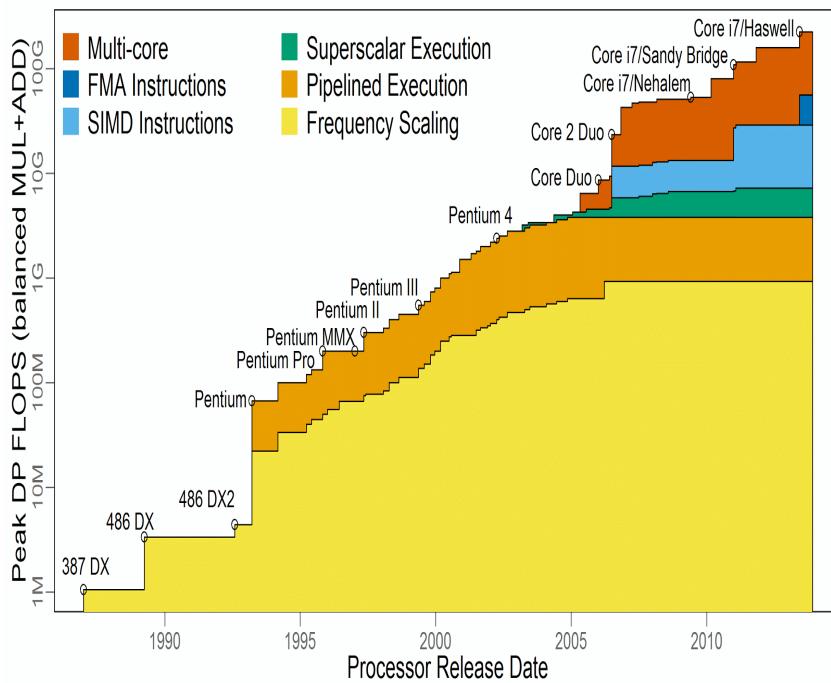
#I – depende do algoritmo, da linguagem de programação, do compilador e da arquitectura (ISA)

CPI – depende da arquitectura (ISA), da mistura de instruções efectivamente utilizadas, da organização do processador e da organização dos restantes componentes do sistema (ex., memória)

f – depende da organização do processador e da tecnologia utilizada

# Computação Paralela

## Evolução das arquiteturas de computadores (nota: escala lin-log)

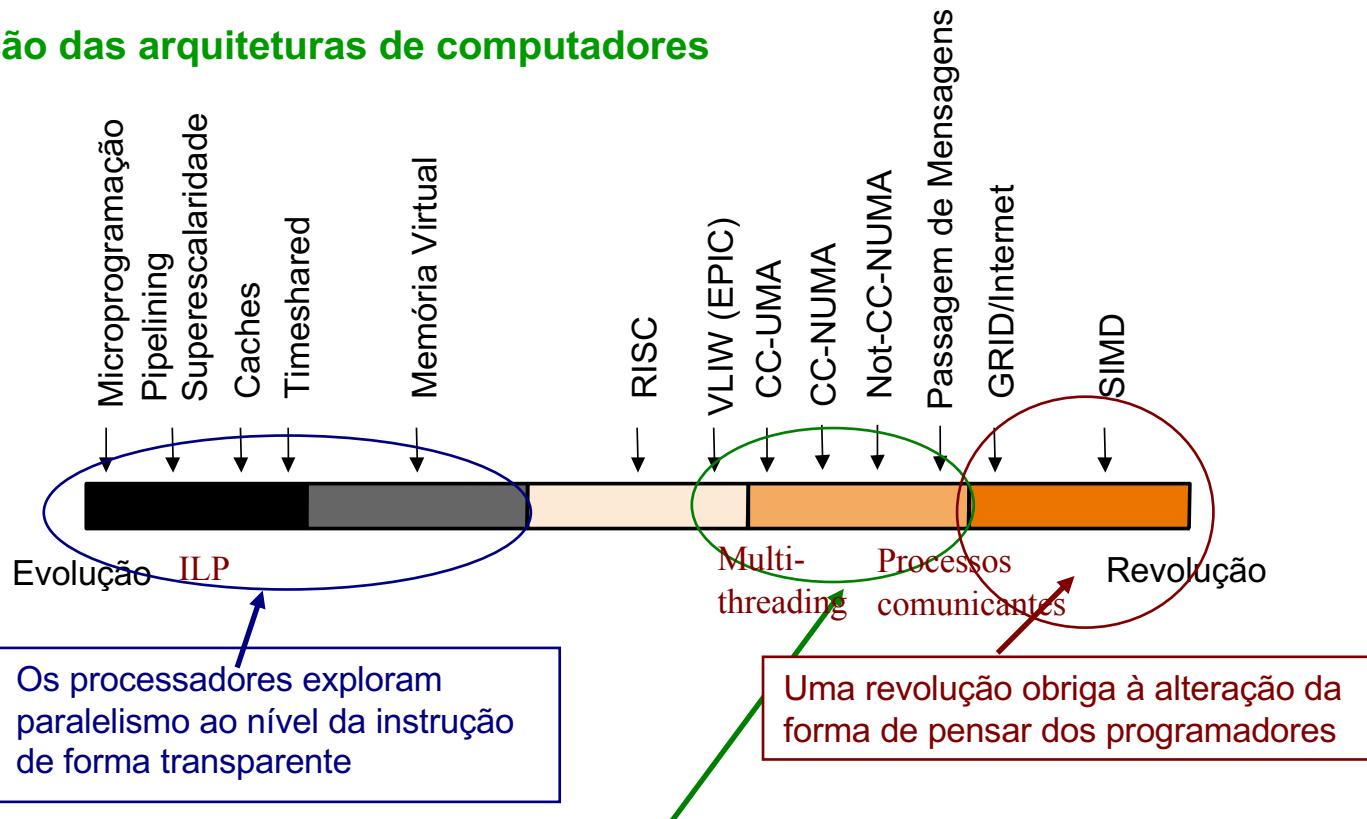


Desempenho (máx) =  
#cores  
X  
SIMD  
X  
IPC  
X  
Freq

$$\text{Texe (ideal)} = (\#I/\text{SIMD}) \times \text{CPI} \times \text{Tcc}/\#\text{cores}$$

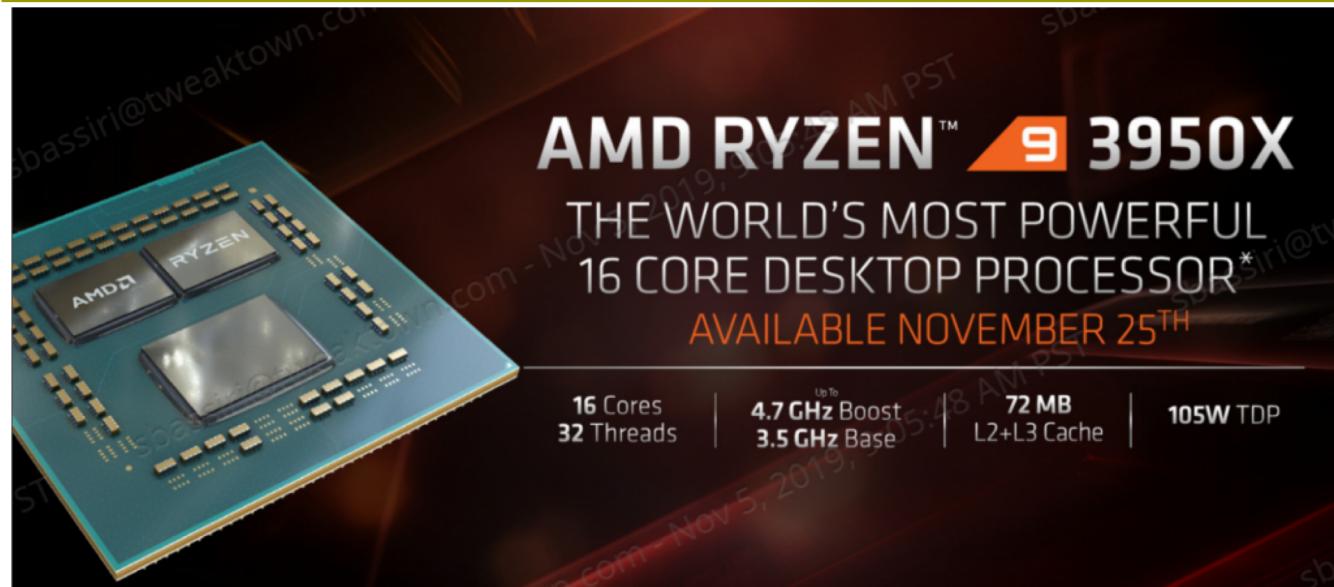
# Paradigmas de Computação Paralela

## Evolução das arquiteturas de computadores



A ênfase de computação paralela é na programação deste tipo de arquitecturas

# Computação Paralela



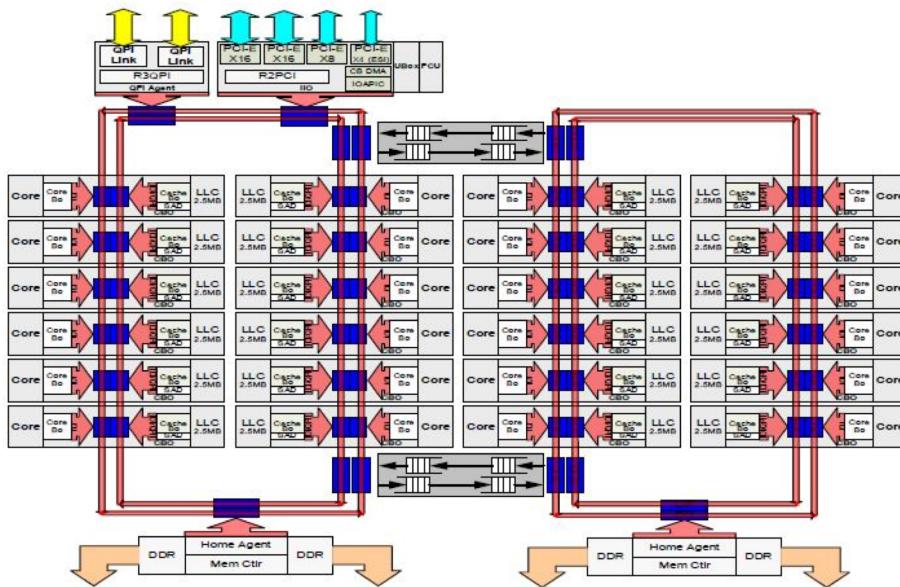
	GeForce RTX 3090	GeForce RTX 3080	GeForce RTX 3070
CUDA Cores	10,496	8,704	5,888
VRAM	24 GB GDDR6X	10 GB GDDR6X	8 GB GDDR6
Prices Starting At	\$1499	\$699	\$499
Release Date	September 24	September 17	Available October

# Paradigmas de Computação Paralela

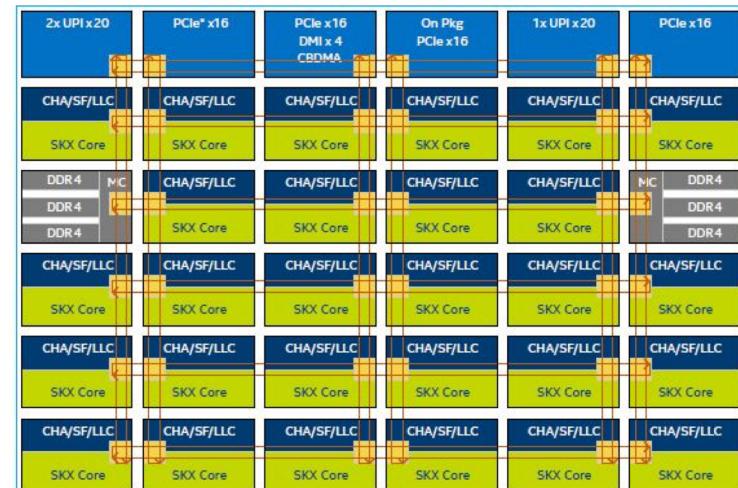
Intel Skylake (max 28 cores – 56 threads)

## New Mesh Interconnect Architecture

Broadwell EX 24-core die



Skylake-SP 28-core die



CHA – Caching and Home Agent ; SF – Snoop Filter; LLC – Last Level Cache ;  
SKX Core – Skylake Server Core; UPI – Intel® UltraPath Interconnect

MESH IMPROVES SCALABILITY WITH HIGHER BANDWIDTH AND REDUCED LATENCIES

# Paradigmas de Computação Paralela

## Níveis de paralelismo (HW+SW)

### □ Instrução (ILP)

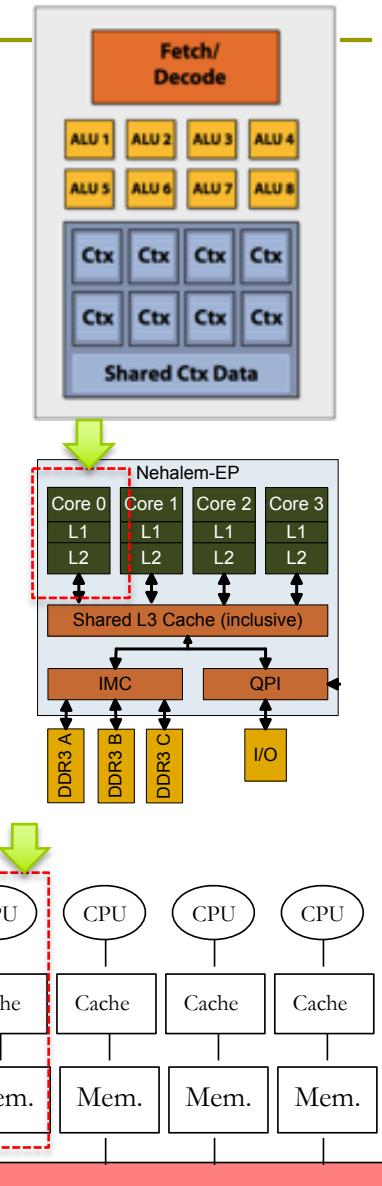
- Execução de múltiplas instruções de um programa em paralelo
- Processamento vetorial
- Explorado pelo hardware atual
- Limitado pelas dependências de dados/controlo do programa

### □ Tarefas / fios de execução

- múltiplos fluxos de instruções de um mesmo programa executam em paralelo
- Limitado pelas dependências e características do algoritmo

### □ Processos

- Múltiplos processos de um mesmo programa / ou de vários programas

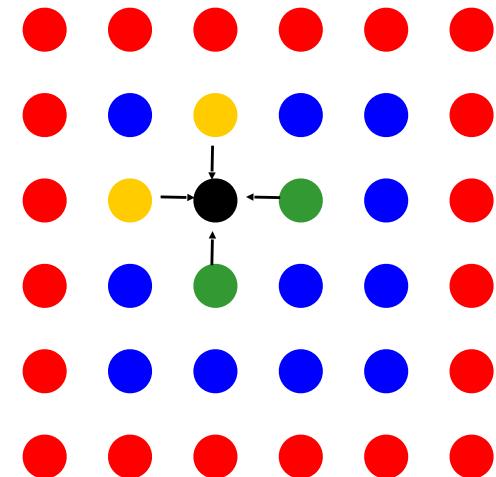


# Paradigmas de Computação Paralela

## Níveis de paralelismo: exemplo “stencil”

```
for(i, ...  
    for(j, ...
```

$$A[i,j] = 0,2 \times ( A[i-1,j] + A[i,j-1] + A[i,j] + A[i+1,j] + A[i,j+1] )$$



### □ Instrução (ILP)

- Ler valores  $A[...]$  da memória em paralelo?
- Efetuar as operações aritméticas em paralelo
- Multiplicação por 0,2 e escrita de  $A[i,j]$  só no final do cálculo
  
- Calcular valores de  $A$  em paralelo?

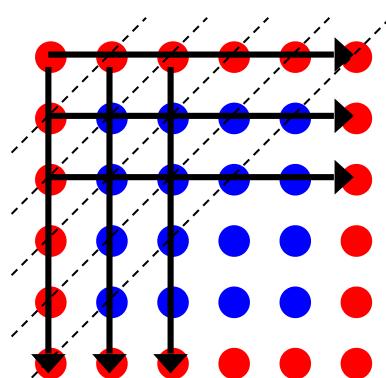
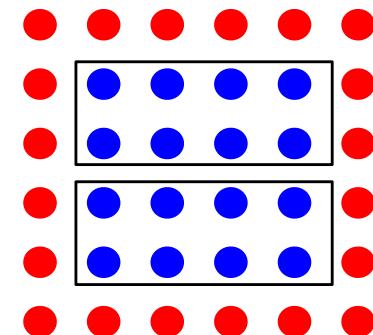
# Paradigmas de Computação Paralela

## Níveis de paralelismo: exemplo “stencil”

$$A[i,j] = 0,2 \times ( A[i-1,j] + A[i,j-1] + A[i,j] + A[i,j+1] + A[i+1,j] )$$

### □ Fios de execução

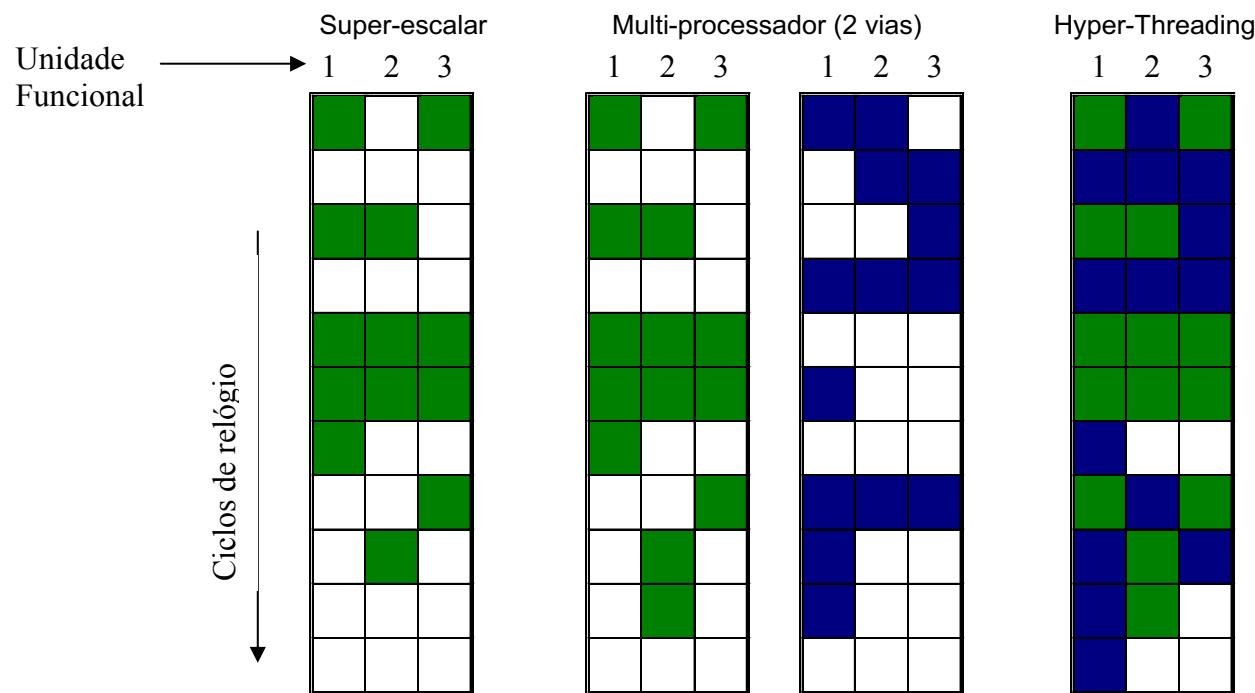
- Cada atividade calcula uma parte dos valores da matriz
- Dependências?



# Paradigmas de Computação Paralela

## Níveis de paralelismo: software VS hardware

### □ Hyper-threading: Fios de execução usados para aumentar o ILP



1 processador = 2 processadores ?

Não porque parte dos recursos do processador não são duplicados (i.e., são divididos pelos vários fios de execução)  
(caches, registos internos, buffers internos, etc.)