

MÁQUINAS PARALELAS E MULTITHREADING

Infraestrutura
Computacional Pt.3
Marco A. Z. Alves

PROFESSOR

Marco Antonio **Zanata Alves**

mazalves@inf.ufpr.br

Gabinete 86 – Departamento de Informática

PESQUISAS QUE MAIS ME INTERESSAM

Arquitetura de Computadores

Processamento de Alto Desempenho

Simulação de Sistemas Computacionais

Processamento em Memória

DISCIPLINA

Moodle

- <http://moodle.c3sl.ufpr.br>
- Infraestrutura Computacional - Parte 3 = InfraComp_Pt.3
- Senha: ICP3

PROCEDIMENTOS DIDÁTICOS

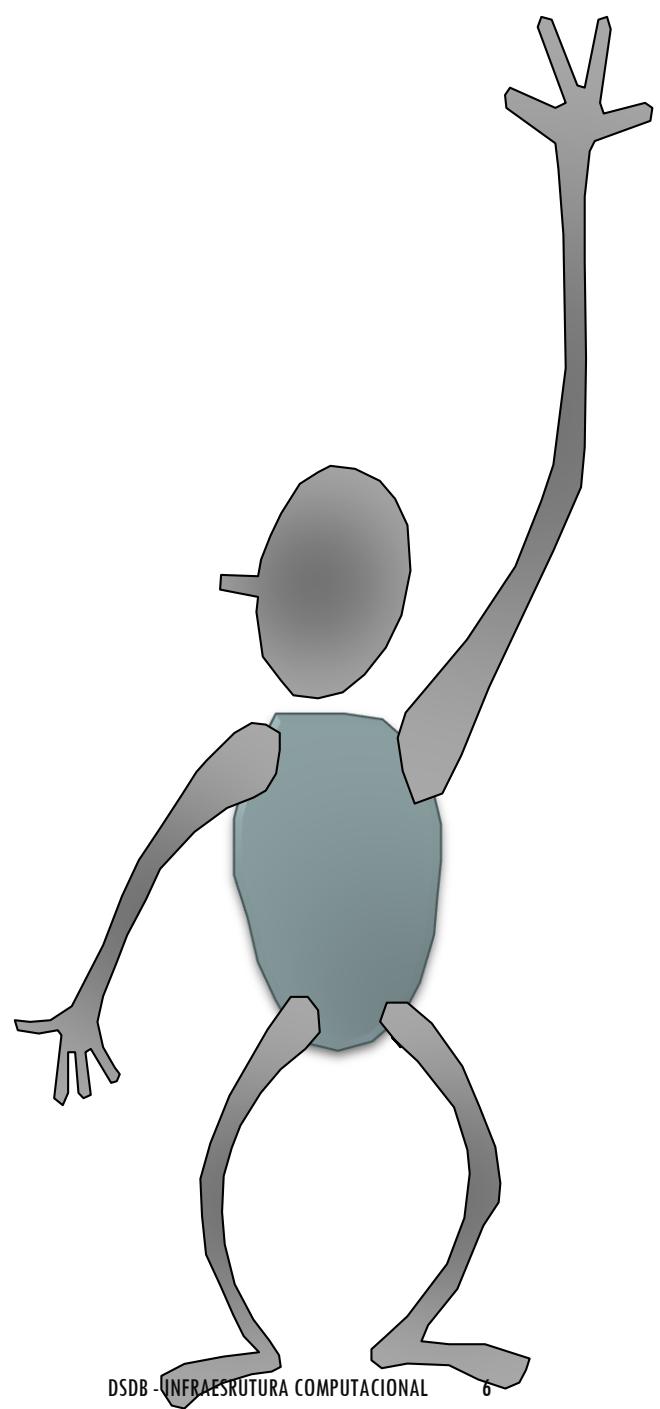
Aulas Expositivas

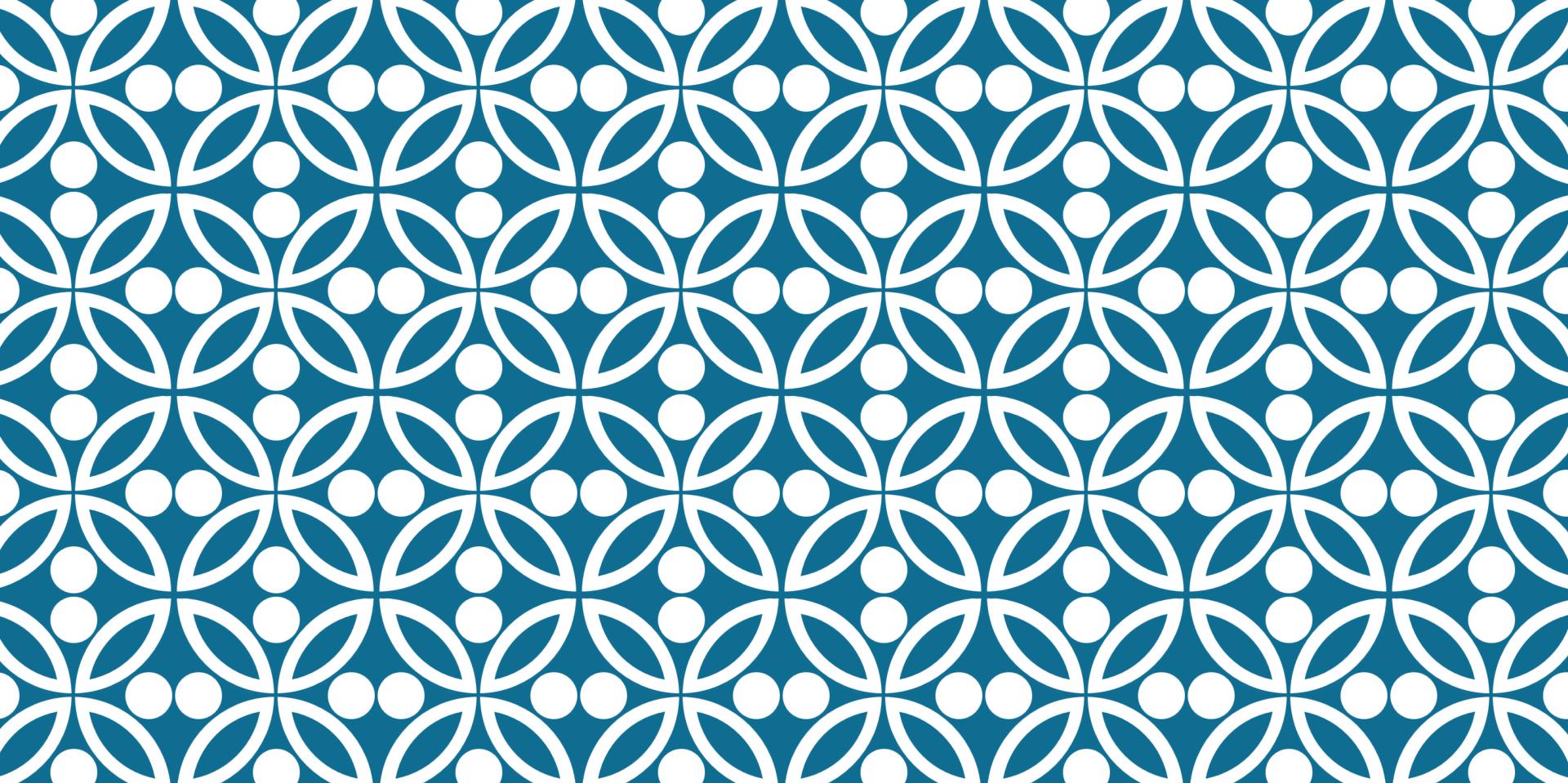
- Slides + Quadro & Giz
- Laboratório (quando disponível)

Trabalhos Práticos

Lista de Exercícios

NÃO TENHA
MEDO DE
PERGUNTAR!!!



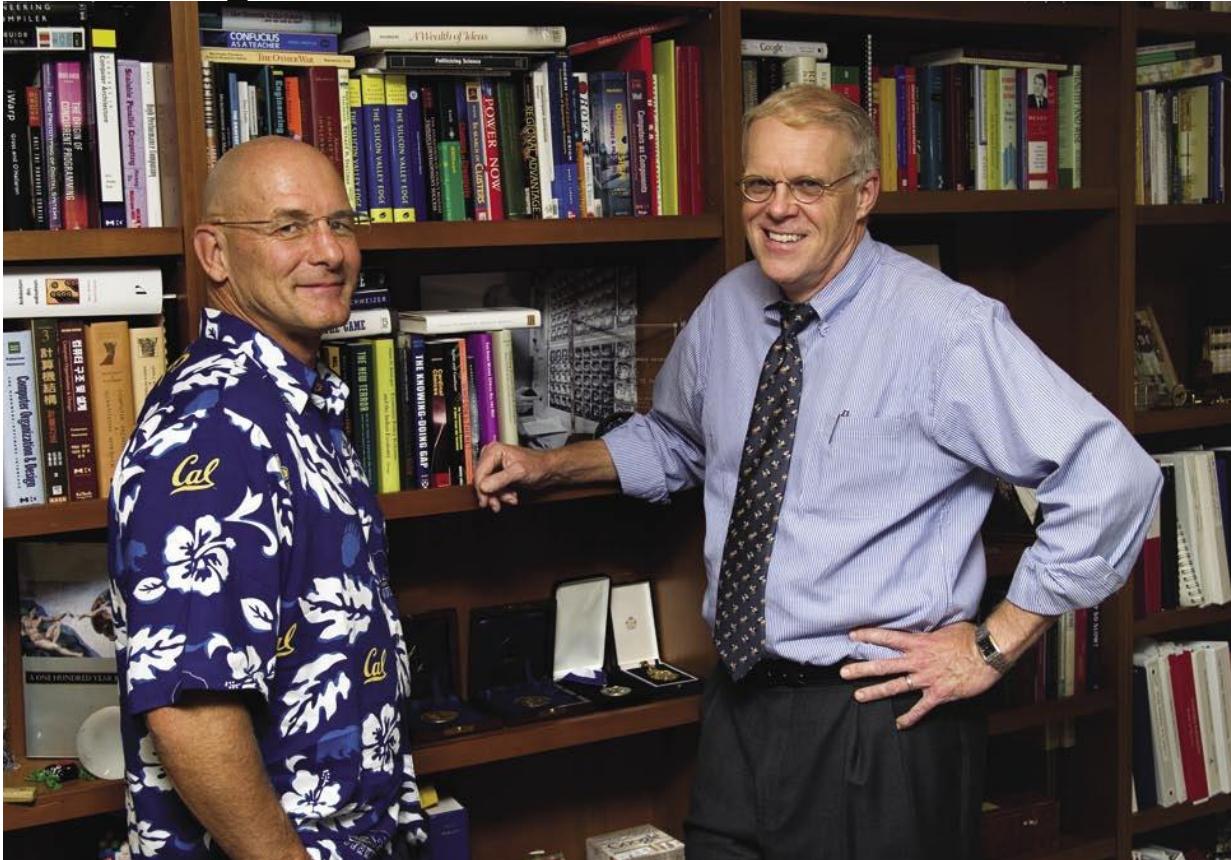


APRESENTAÇÃO DA DISCIPLINA

O QUE ESPERAR DESSA DISCIPLINA

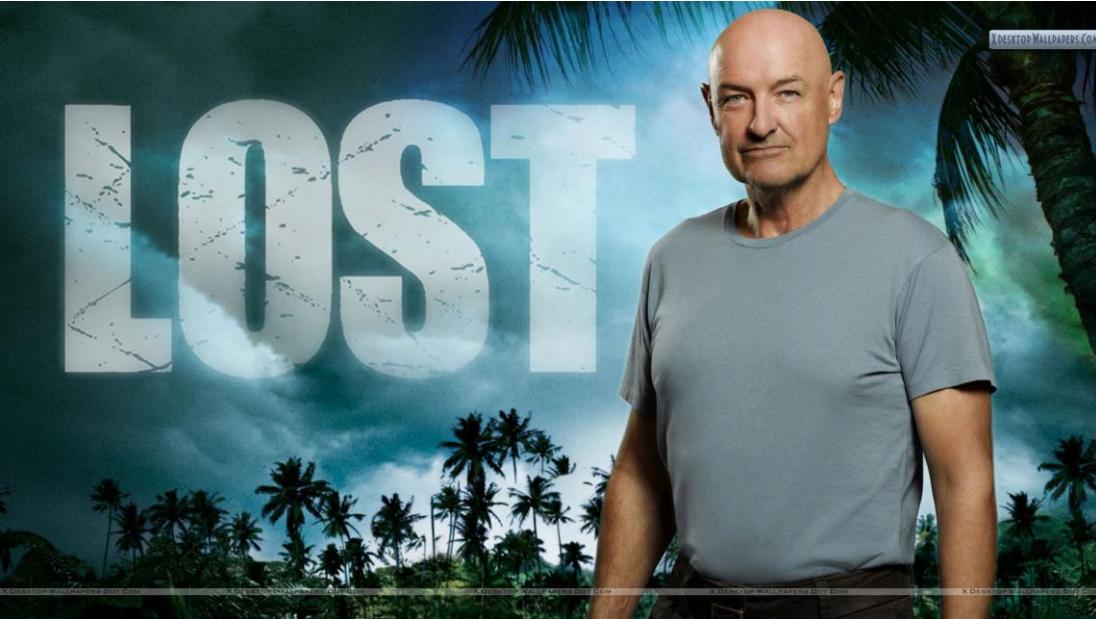
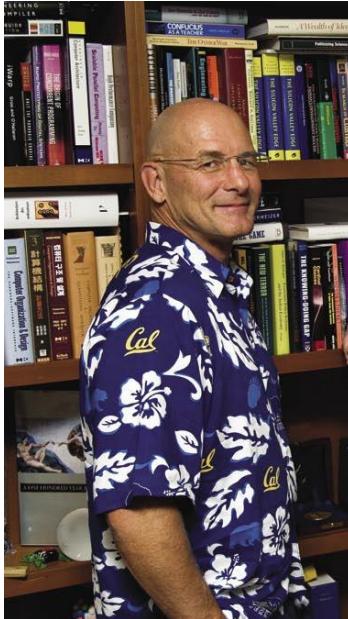


MOTIVAÇÃO



MOTIVAÇÃO

“será!?”

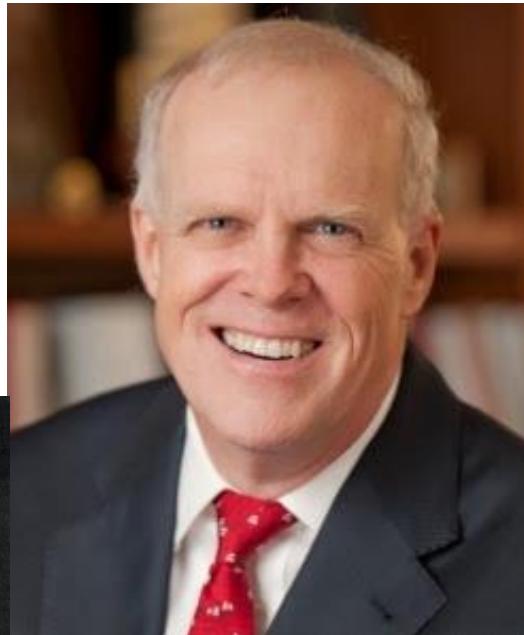


TURING AWARD 2017

“PIONEERS OF MODERN COMPUTER ARCHITECTURE”



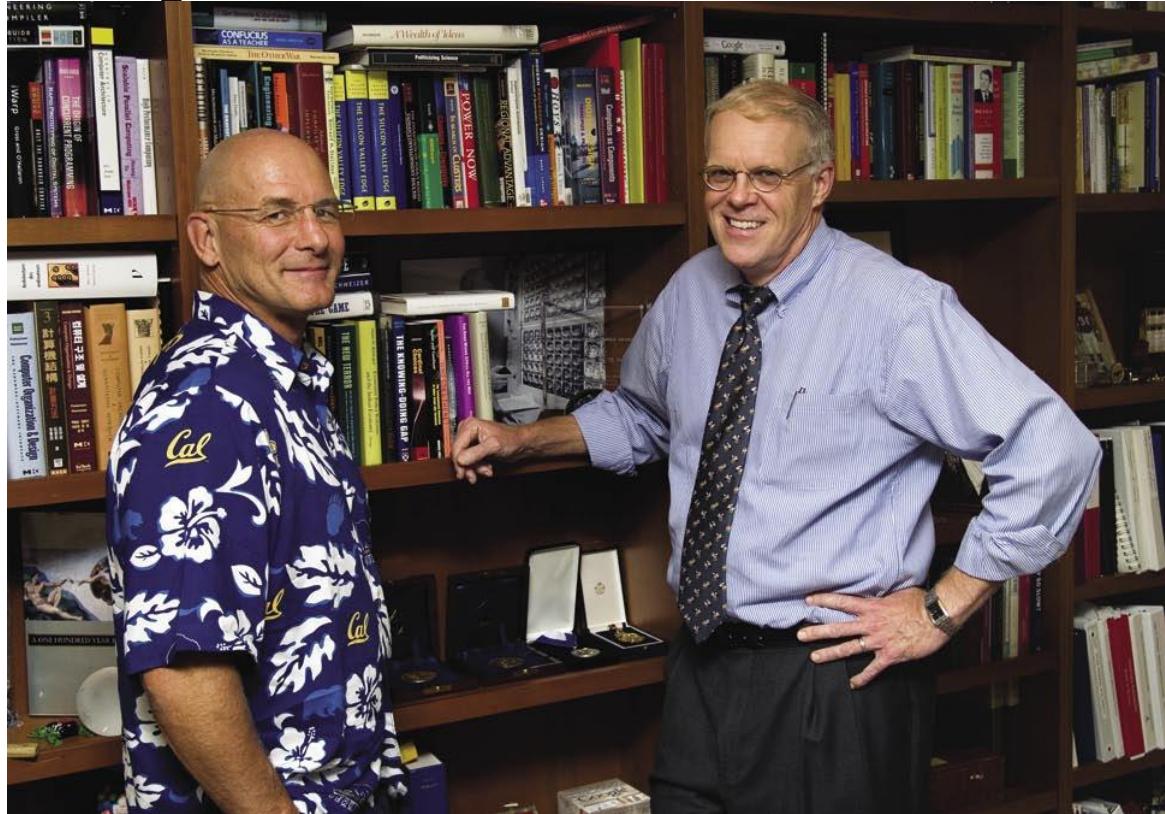
Dave Patterson (dir), e John Hennessy
início dos anos 90.



MOTIVAÇÃO

**When we talk about parallelism...
we're talking about a problem that's as hard as
any computer science has faced.**

JOHN HENNESSY

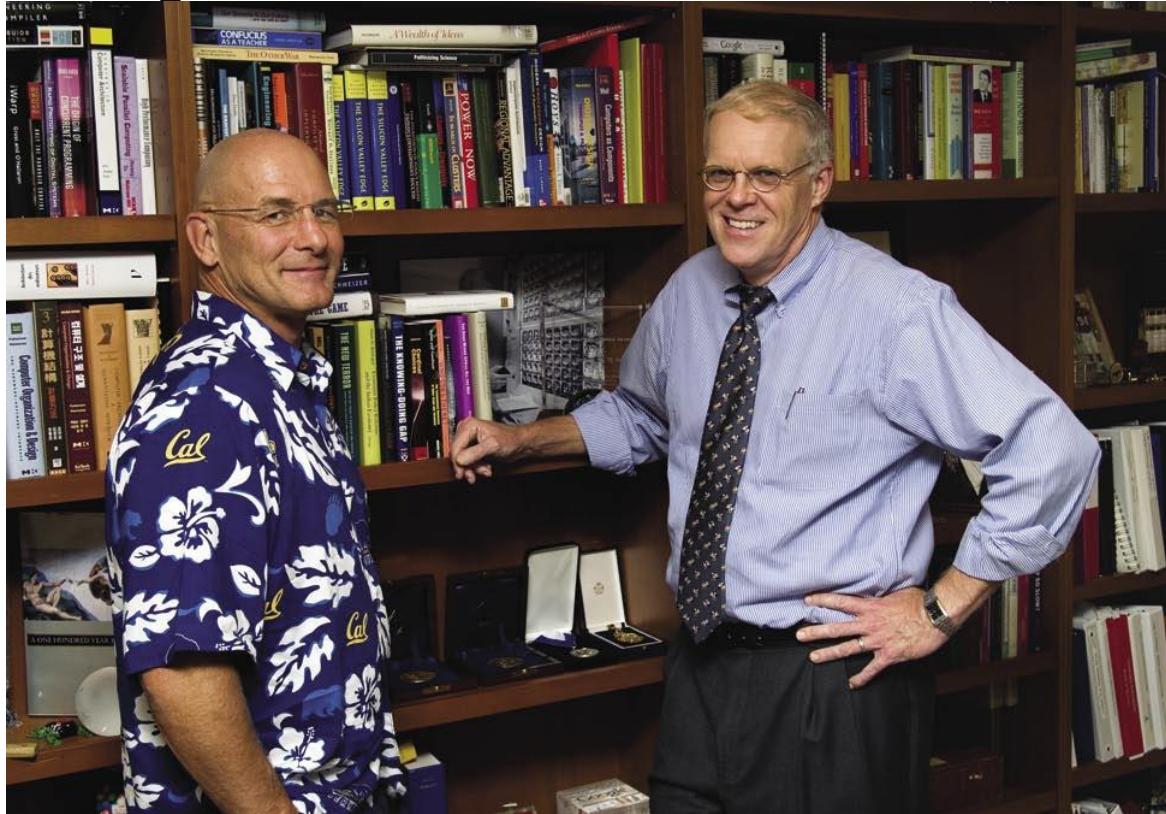


John Hennessy (dir), e Dave Patterson (esq), 2006.

MOTIVAÇÃO

When we talk about **parallelism**...
we're talking about a problem that's as **hard** as
any computer science has faced.

JOHN HENNESSY

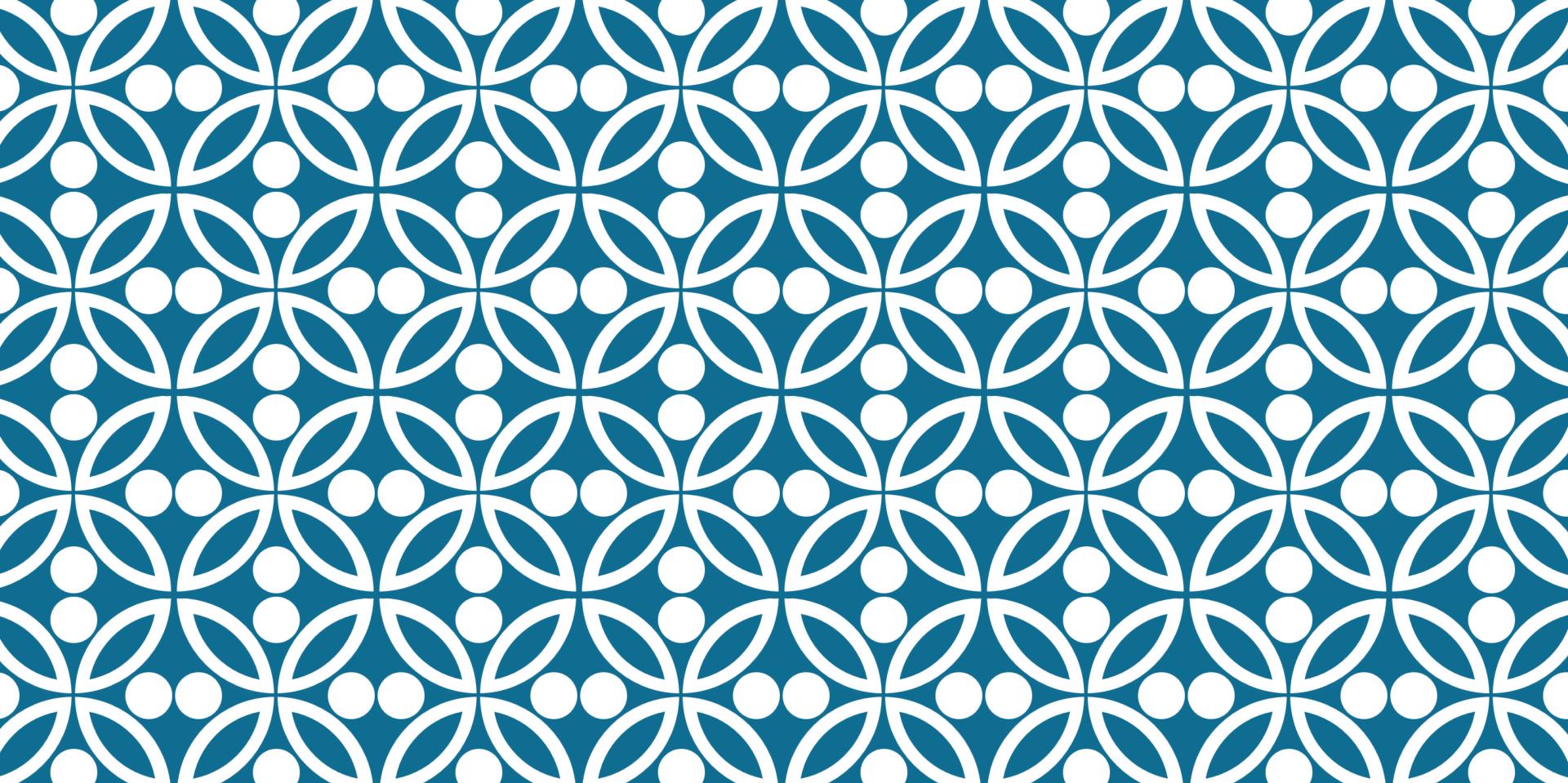


It's the biggest thing

in 50 years because industry is setting its future that
parallel programming will be useful.

DAVID PATTERSON

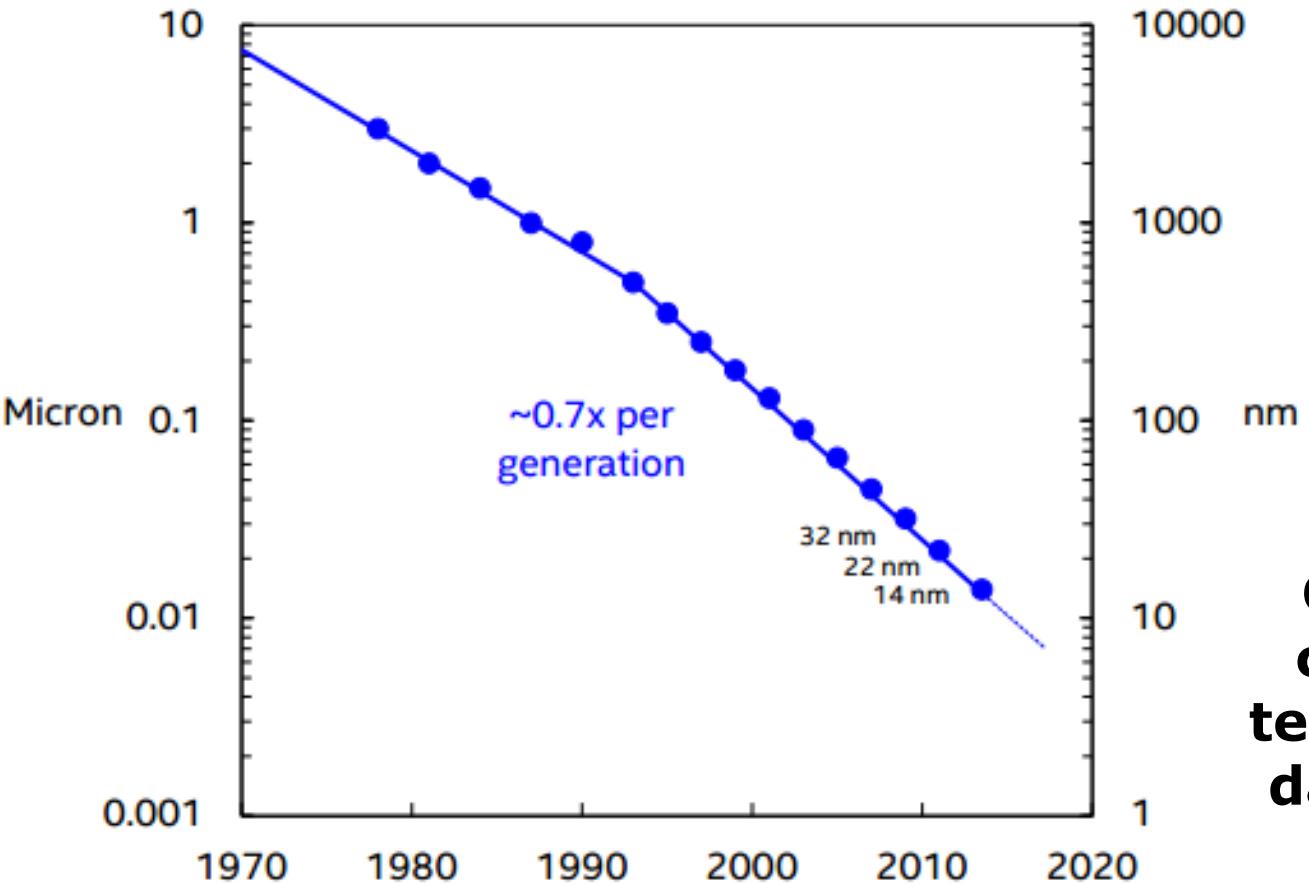
[O'Hanlon, 2006]



BREVE EVOLUÇÃO DOS PROCESSADORES

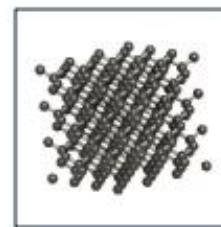
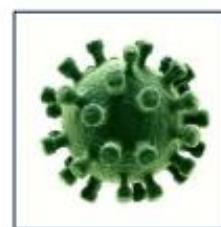
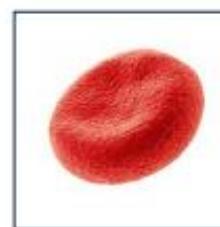
Segundo Yale Patt

INTEL SCALING TREND [IDF'14]



Os transistores de cada nova geração tem 70% do tamanho da geração passada!

QUÃO PEQUENO É 14NM? [IDF'14]



~~Mark~~ Marco
1.66 m ~~1.68 m~~

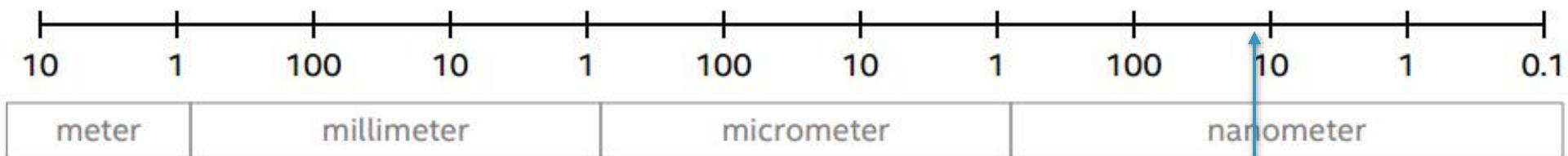
Fly
7 mm

Mite
300 um

Blood Cell
7 um

Virus
100 nm

Silicon Atom
0.24 nm



Transistores
(gate width)

AUMENTO DO DESEMPENHO EM PERSPECTIVA

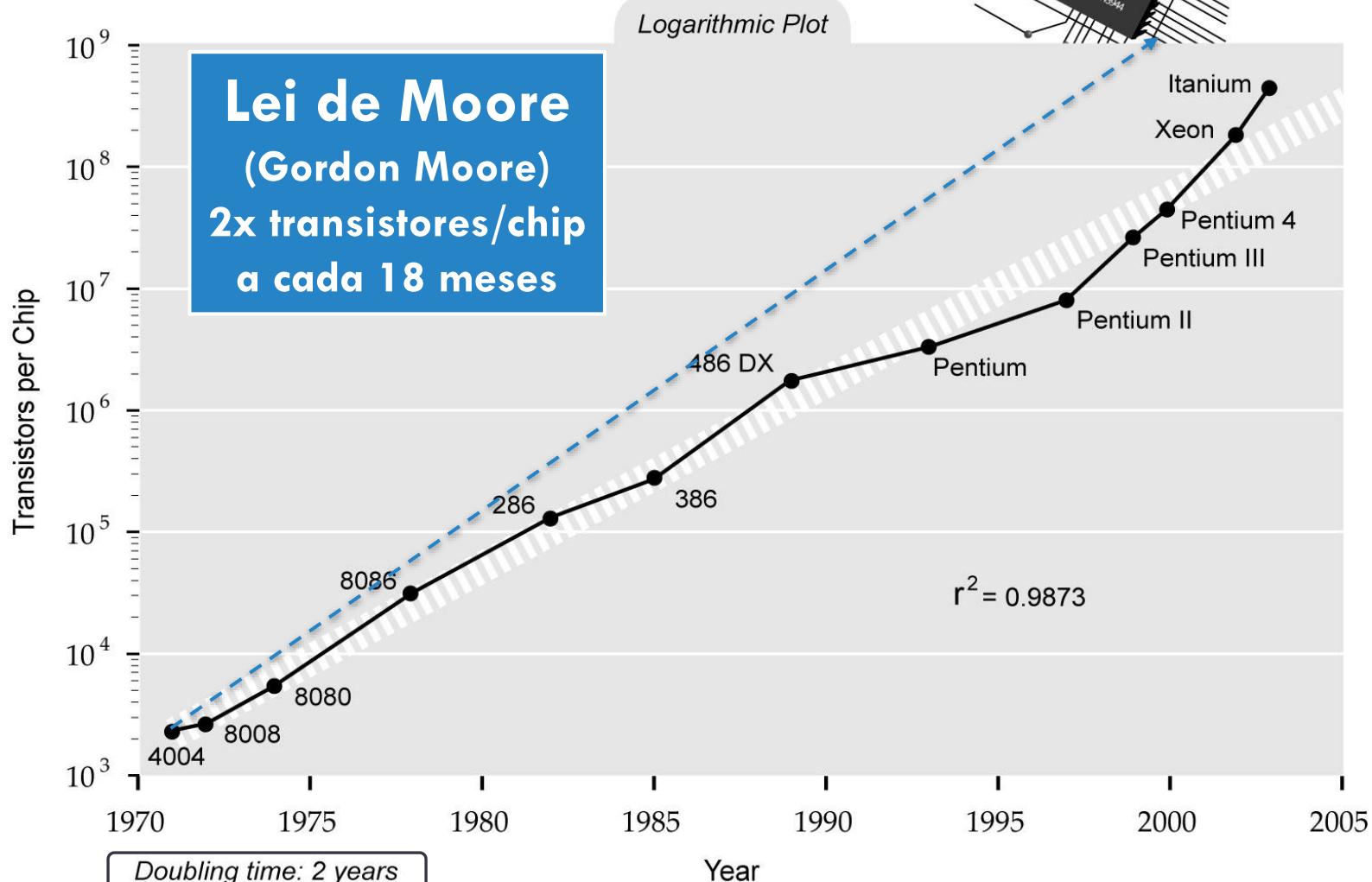
Dobrando a cada 18 meses desde 1972

Carros capazes de fazer 17.600 km/h; 3.200 km/l

Viagem LA-NY em 90 segundos (Mach 200)

CAPACIDADE DO PROCESSADOR

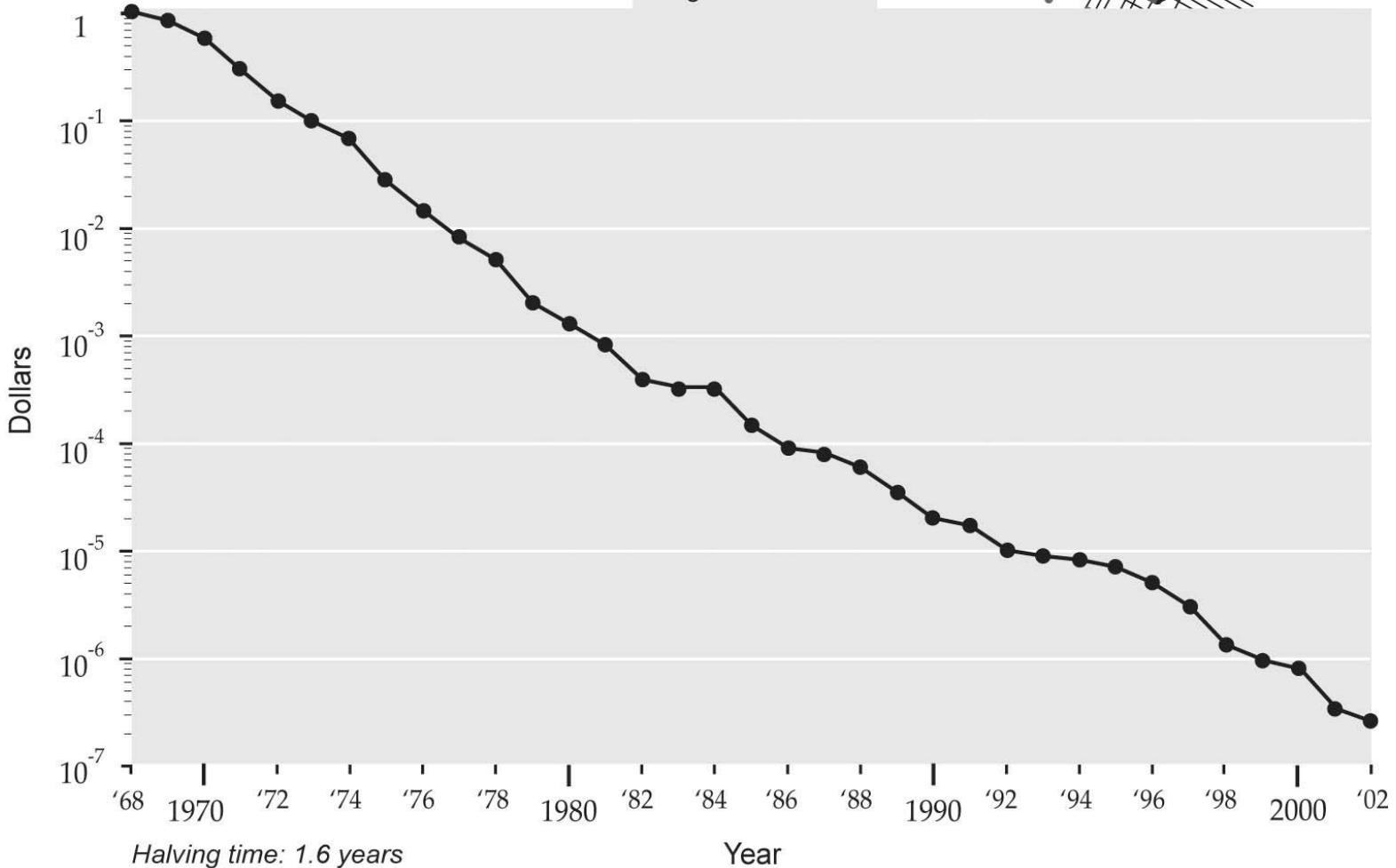
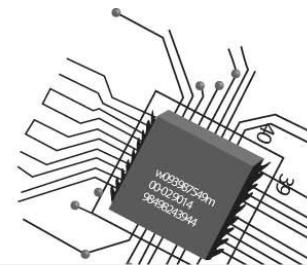
Transistors per Microprocessor



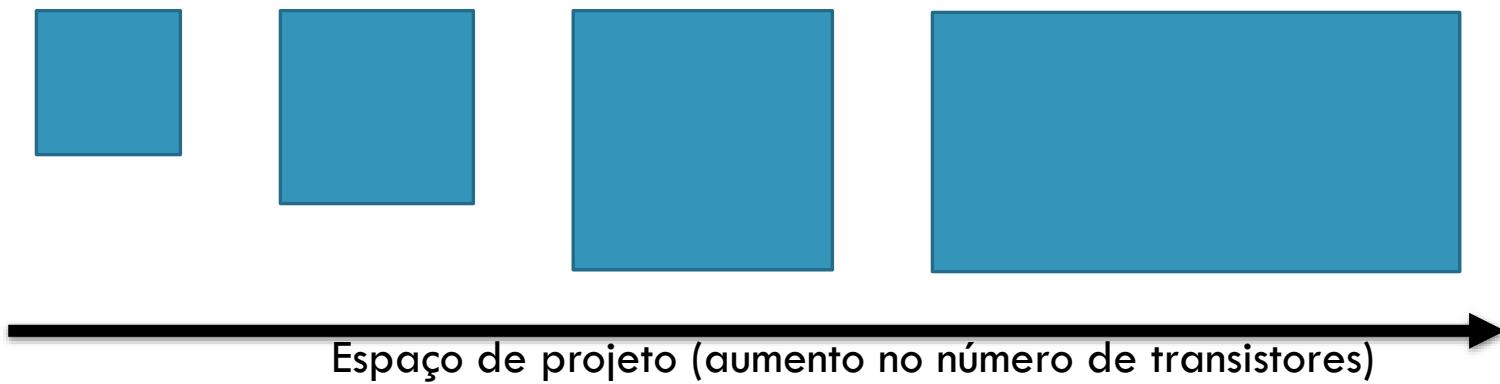
PREÇO DO TRANSISTOR

Average Transistor Price

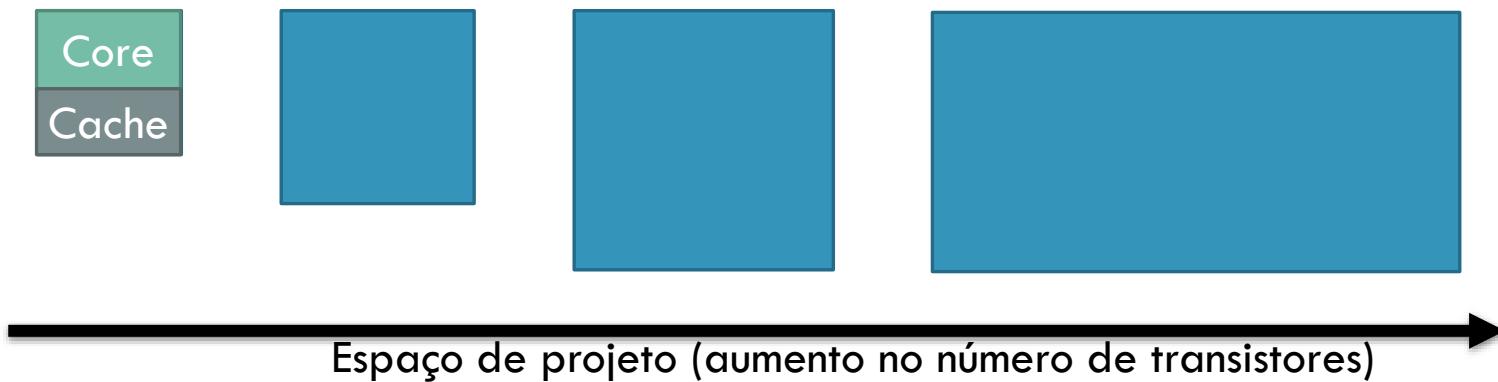
Logarithmic Plot



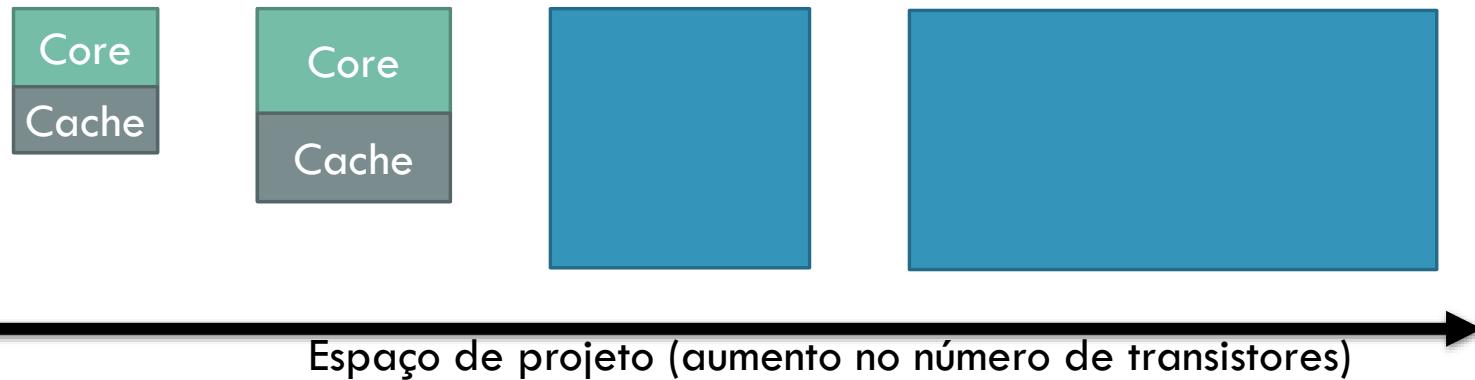
EVOLUÇÃO DOS PROCESSADORES



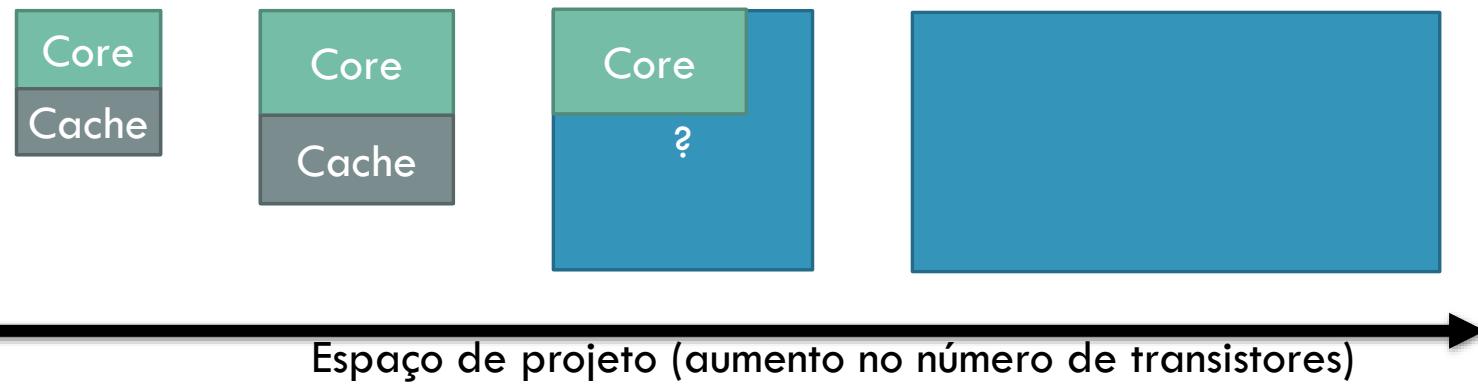
EVOLUÇÃO DOS PROCESSADORES



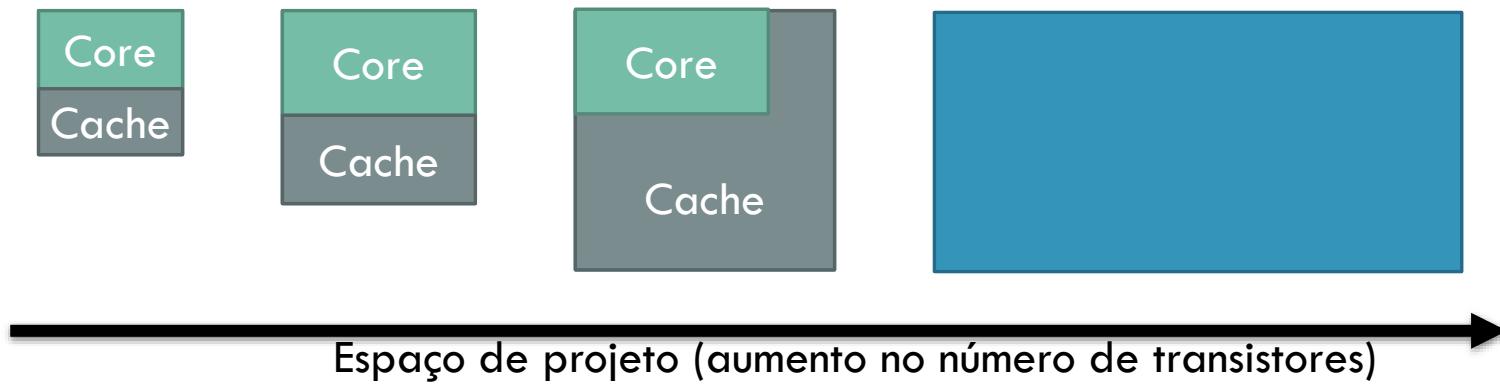
EVOLUÇÃO DOS PROCESSADORES



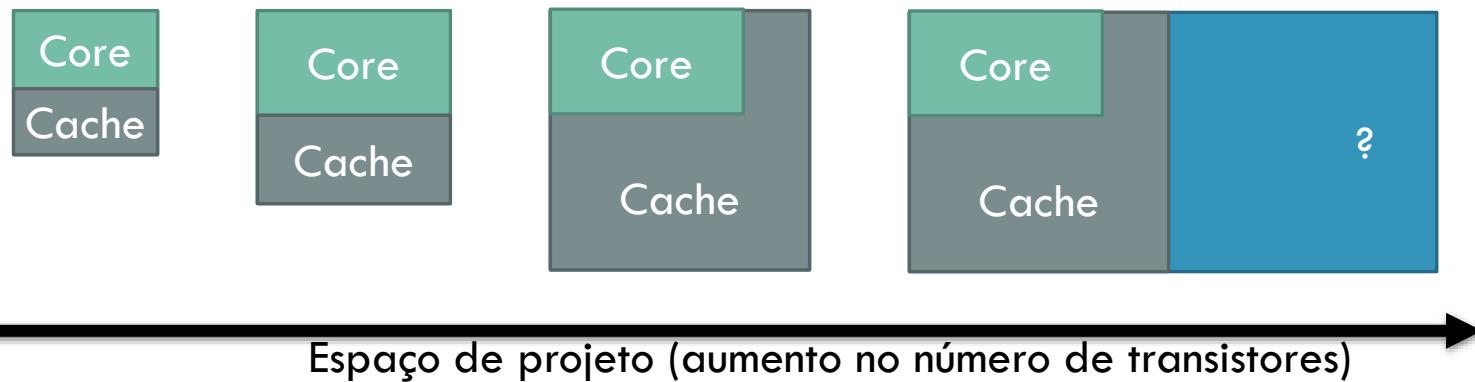
EVOLUÇÃO DOS PROCESSADORES



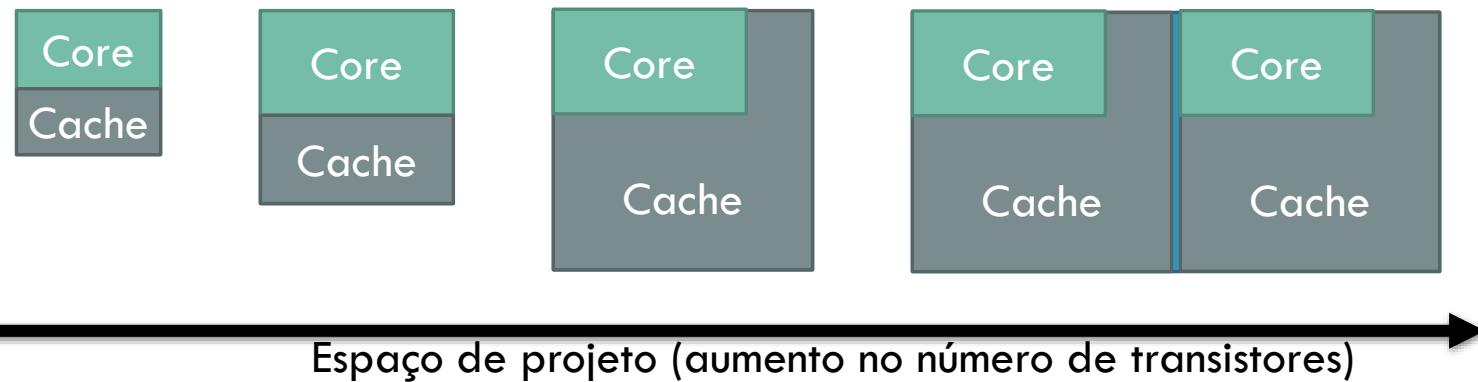
EVOLUÇÃO DOS PROCESSADORES



EVOLUÇÃO DOS PROCESSADORES

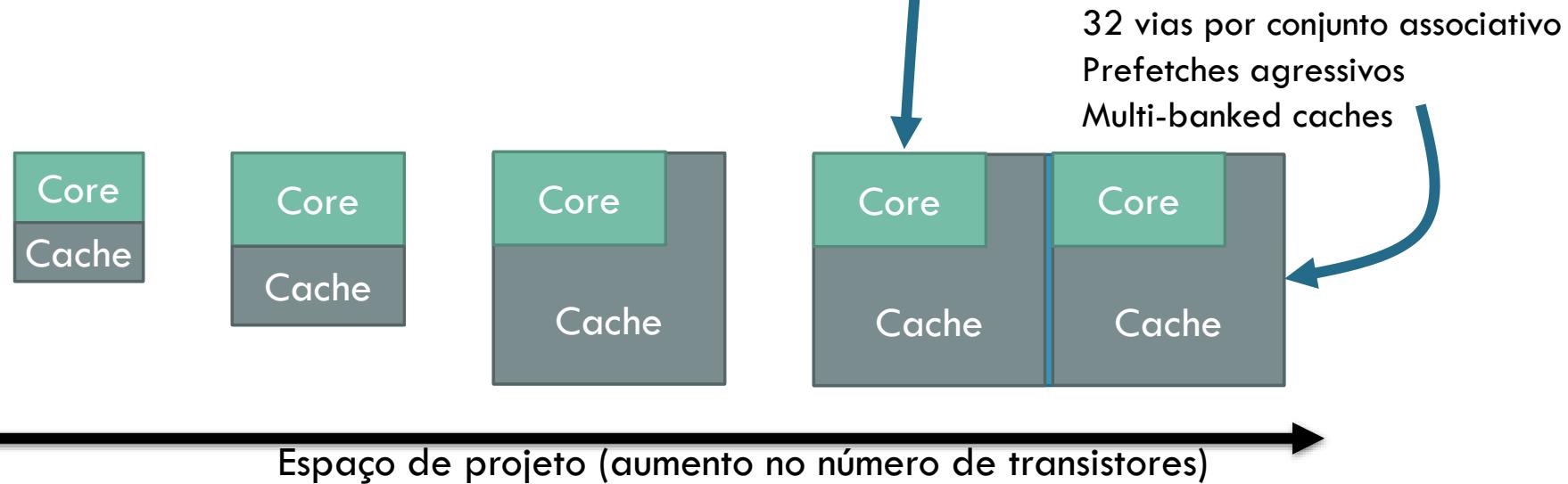


EVOLUÇÃO DOS PROCESSADORES

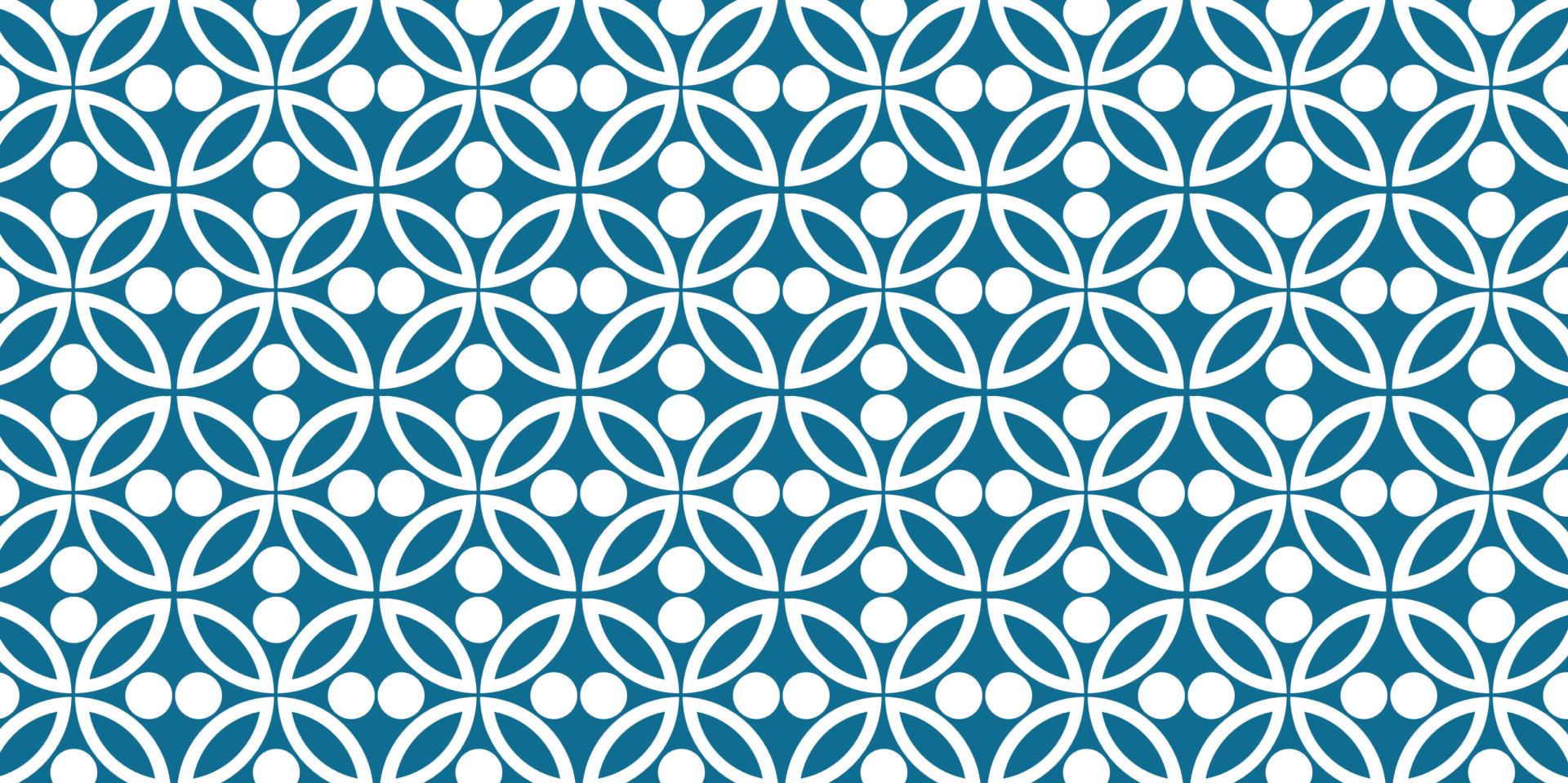


Processadores com alto ILP
Pipelines de 12 estágios
Reorder buffer de até 192 posições
Branch predictors avançados

EVOLUÇÃO DOS PROCESSADORES



Técnicas
tradicionais
continuam
importantes!



CLASSIFICANDO OS COMPUTADORES

TAXONOMIA DE FLYNN

*[Flynn, 1972]

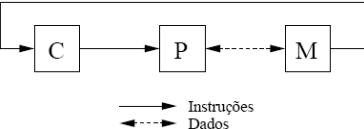
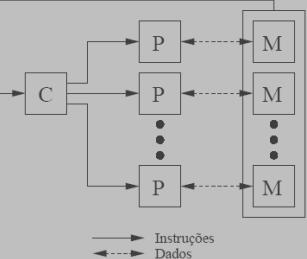
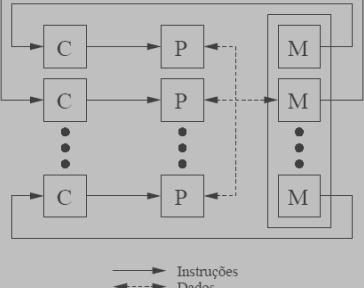
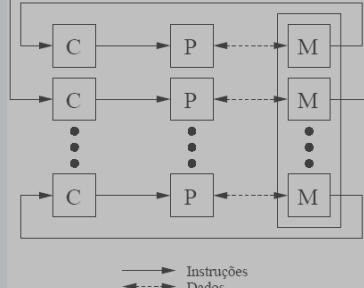
**[De Rose,
2003]

*	SD (Single Data)	MD (Multiple Data)
SI (Single Instruction)	<ul style="list-style-type: none">** <p>SISD (Máquinas von Neumann)</p>	<ul style="list-style-type: none">** <p>MISD (Sem representante / Arquiteturas Sistólicas)</p>
MI (Multiple Instruction)	<ul style="list-style-type: none">** <p> SIMD (Máquinas Vetoriais)</p>	<ul style="list-style-type: none">** <p>MIMD (Multiprocessadores e Multicomputadores)</p>

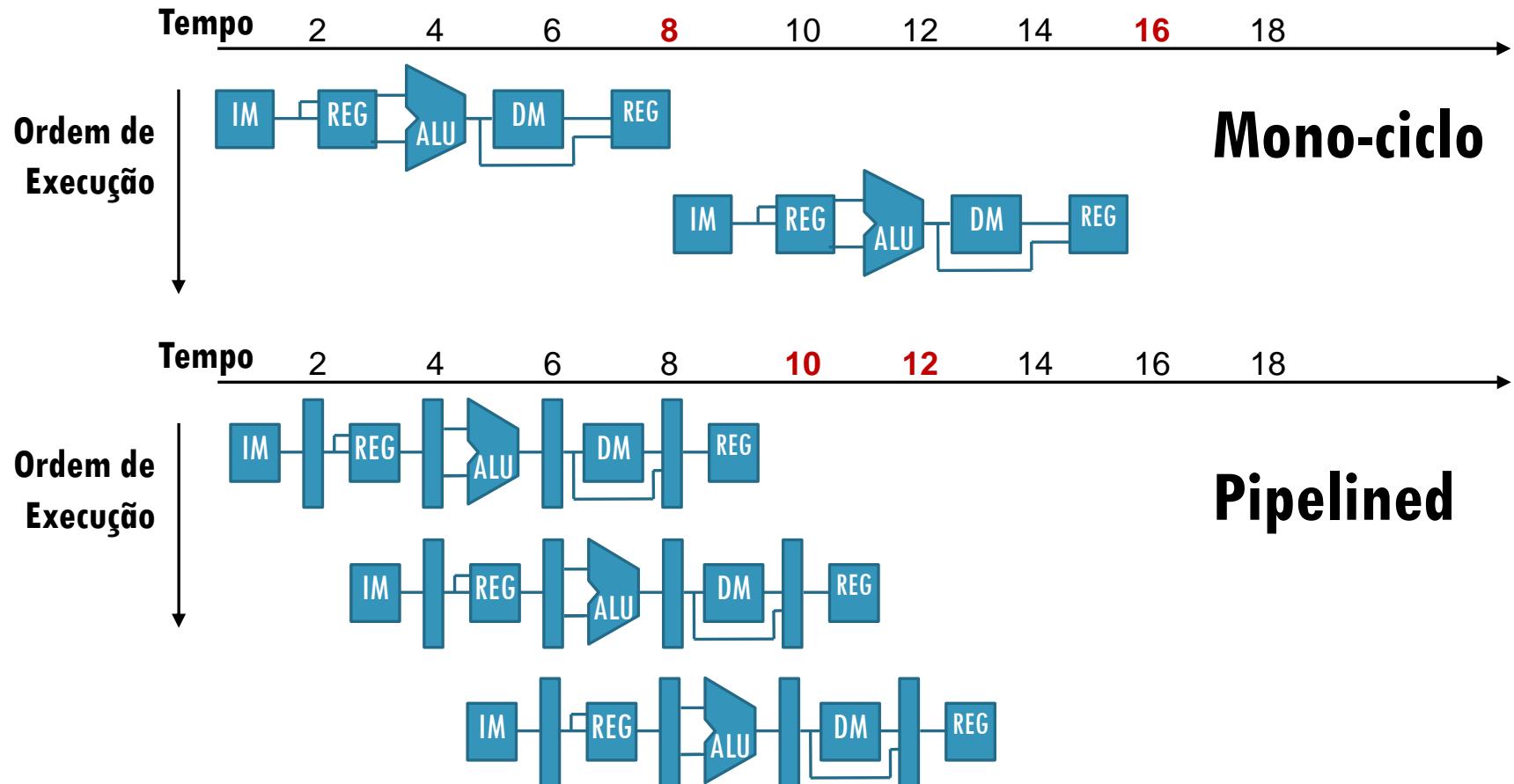
TAXONOMIA DE FLYNN

*[Flynn, 1972]

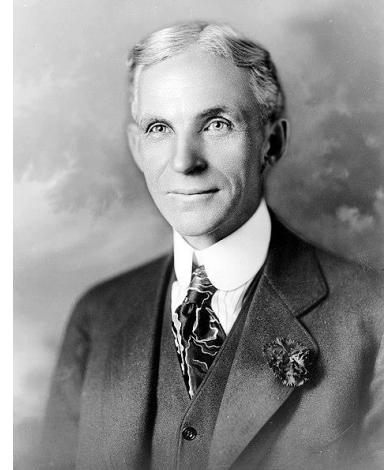
**[De Rose,
2003]

*	SD (Single Data)	MD (Multiple Data)
SI (Single Instruction)	<ul style="list-style-type: none">**  <p>SISD (Máquinas von Neumann)</p>	<ul style="list-style-type: none">**  <p>SIMD (Máquinas Vetoriais)</p>
MI (Multiple Instruction)	<ul style="list-style-type: none">**  <p>MISD (Sem representante / Arquiteturas Sistólicas)</p>	<ul style="list-style-type: none">**  <p>MIMD (Multiprocessadores e Multicomputadores)</p>

MONOCICLO VS. PIPELINE



TRANSFORMANDO O PROCESSADOR EM UMA LINHA DE MONTAGEM



Fordismo, termo criado por Henry Ford, em 1914, refere-se aos sistemas de produção em massa (linha de produção) e gestão idealizados em 1913 pelo empresário americano Henry Ford (1863-1947), fundador da Ford Motor Company.

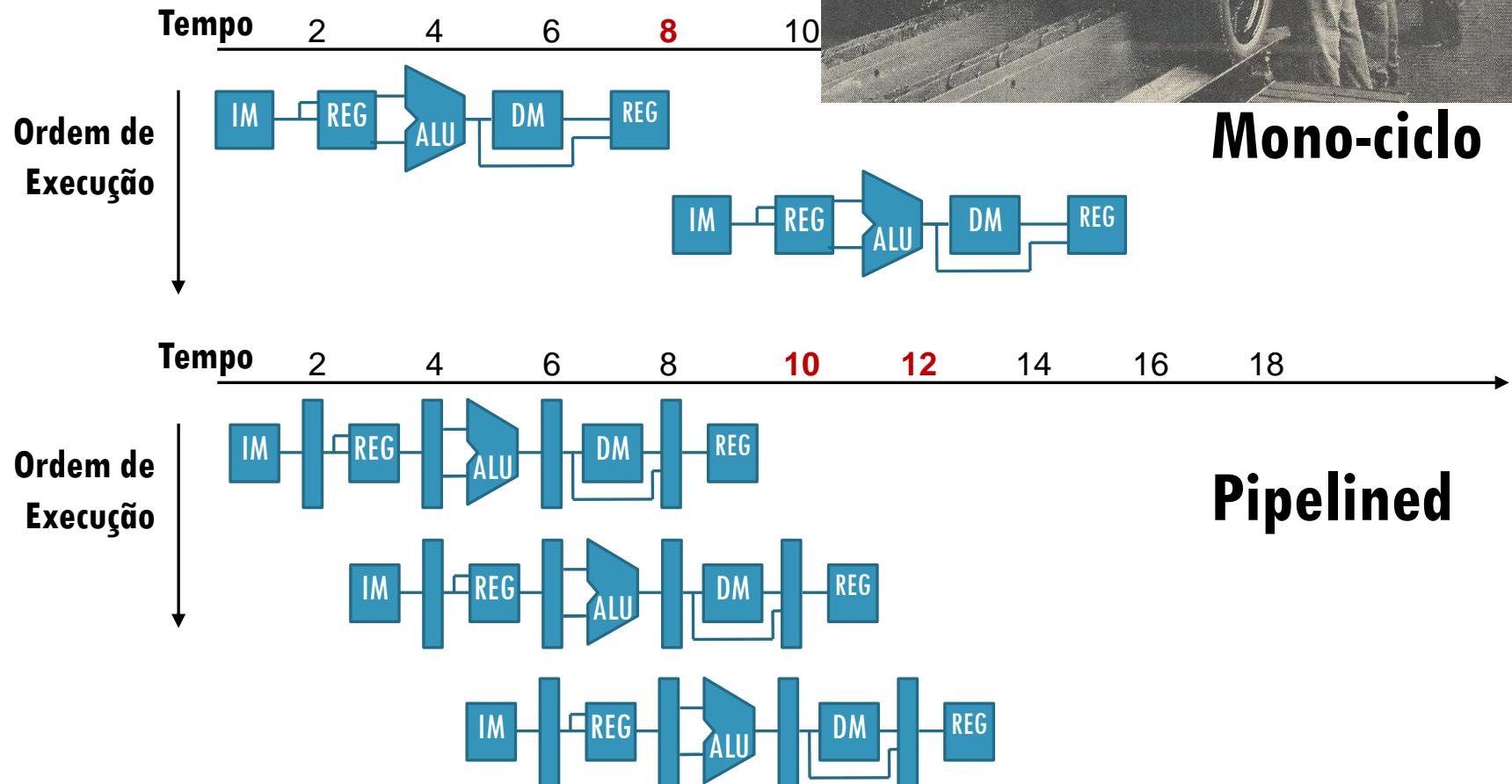
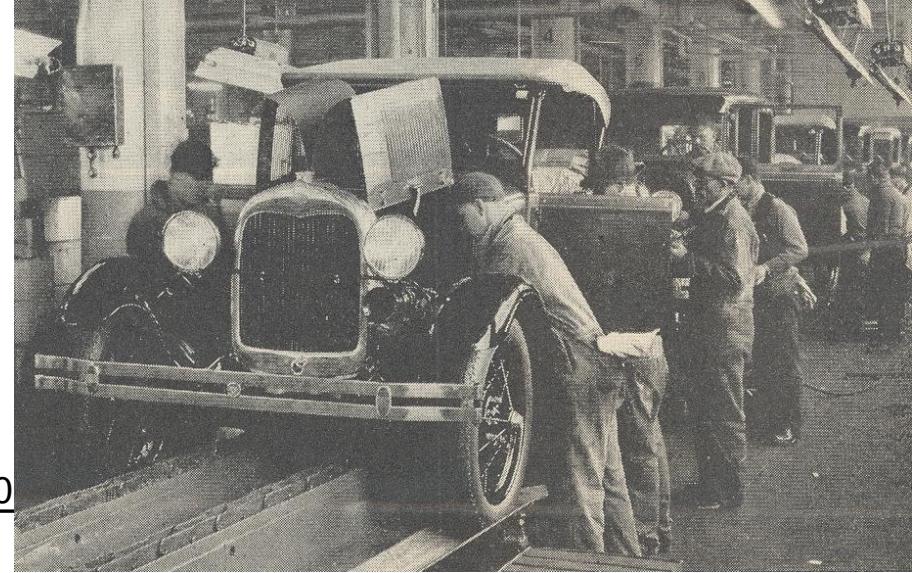
Esse modelo revolucionou a indústria automobilística a partir de 1914, quando Ford introduziu a primeira linha de montagem automatizada.

O aperfeiçoamento da linha de montagem se baseava em:

1. Os veículos eram montados em esteiras rolantes, que se movimentavam enquanto o operário ficava praticamente parado.
2. Cada operário realizava apenas uma operação simples ou uma pequena etapa da produção.
3. Desta forma não era necessária quase nenhuma qualificação dos trabalhadores.

O método de produção fordista permitiu que a Ford produzisse mais de 2 milhões de carros por ano, durante a década de 1920.

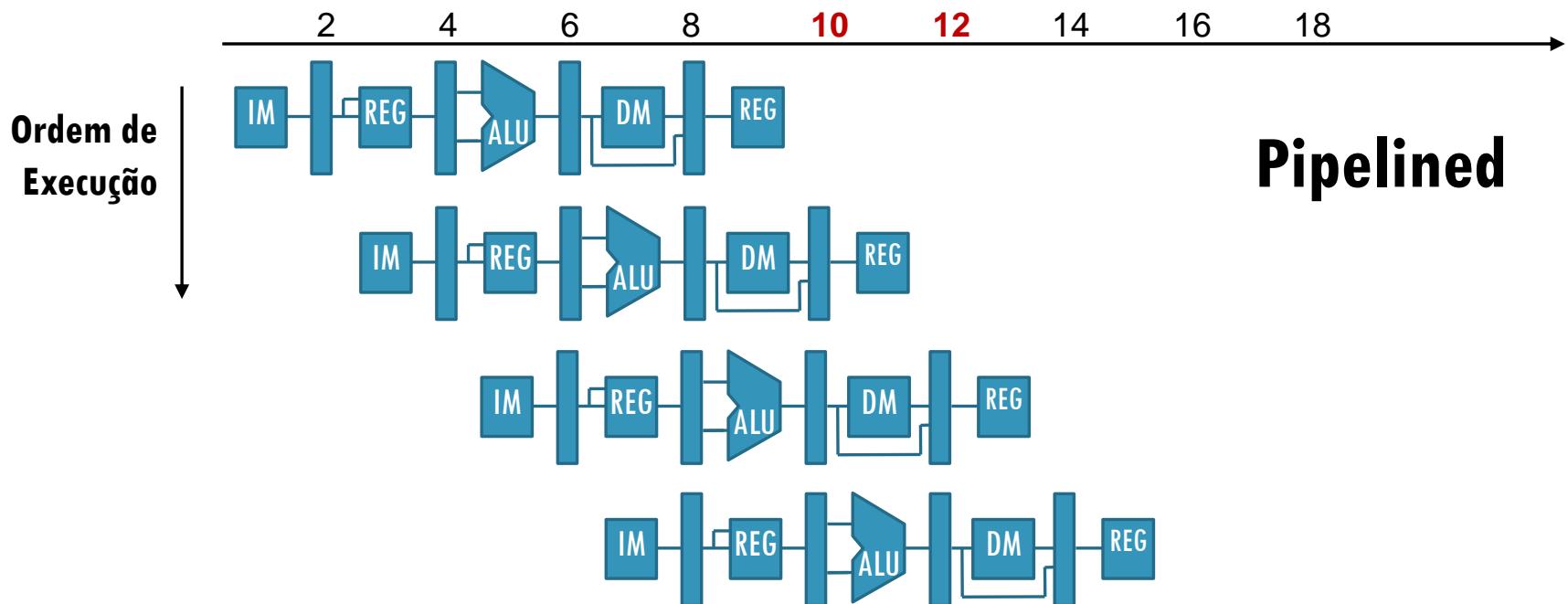
MONOCICLO VS. PIPELINE



SINGLE CORE - PIPELINING

Pipelining **não reduz a latência** de uma instrução única.

Mas **aumenta o throughput (vazão)** de todo workload



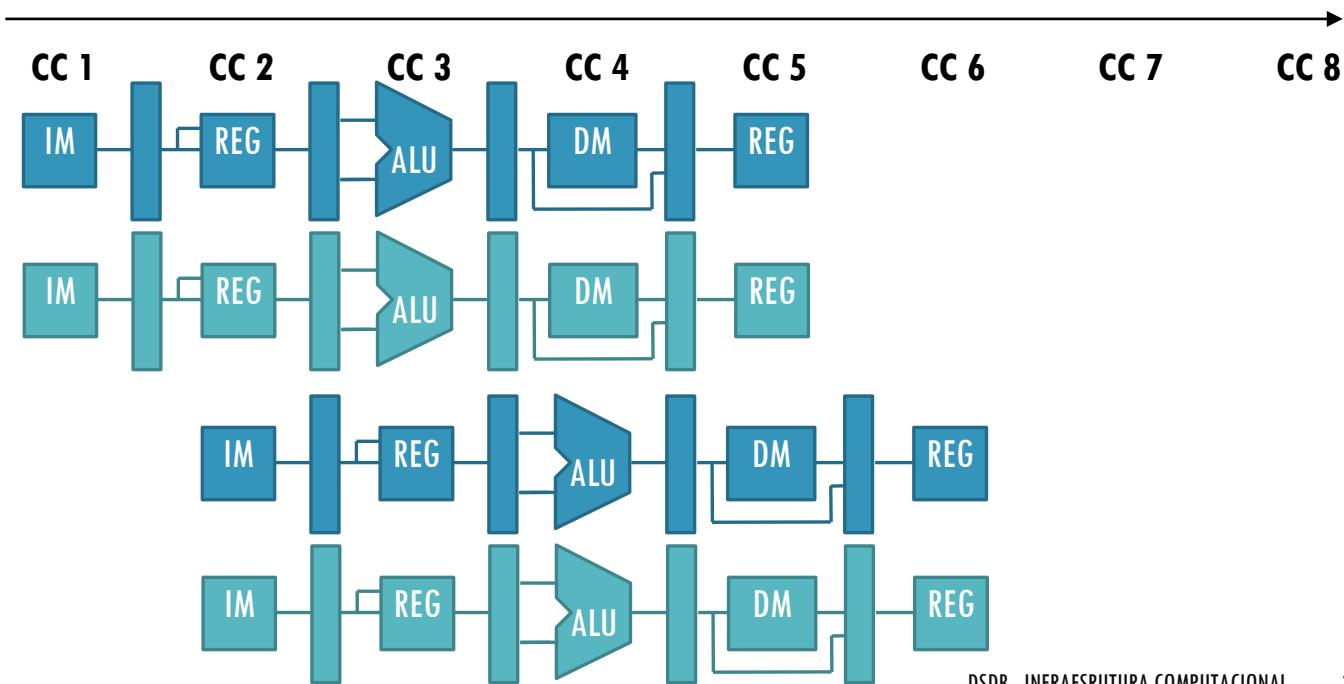
SINGLE CORE - PIPELINE SUPERSCALAR

Pipelining continua ativo

IPC ideal >1

ILP Paralelismo de instruções.

Tempo

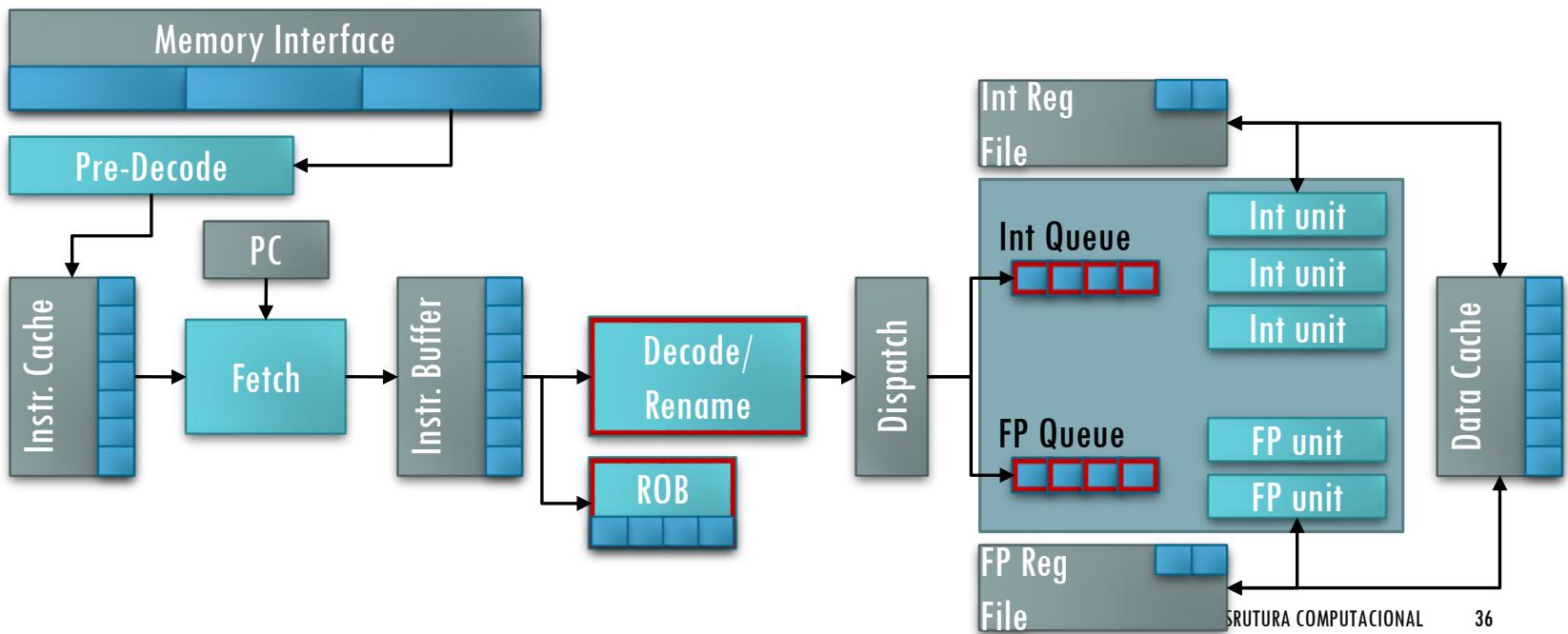


SINGLE CORE - PIPELINE SUPERSCALAR O³ (OUT-OF-ORDER)

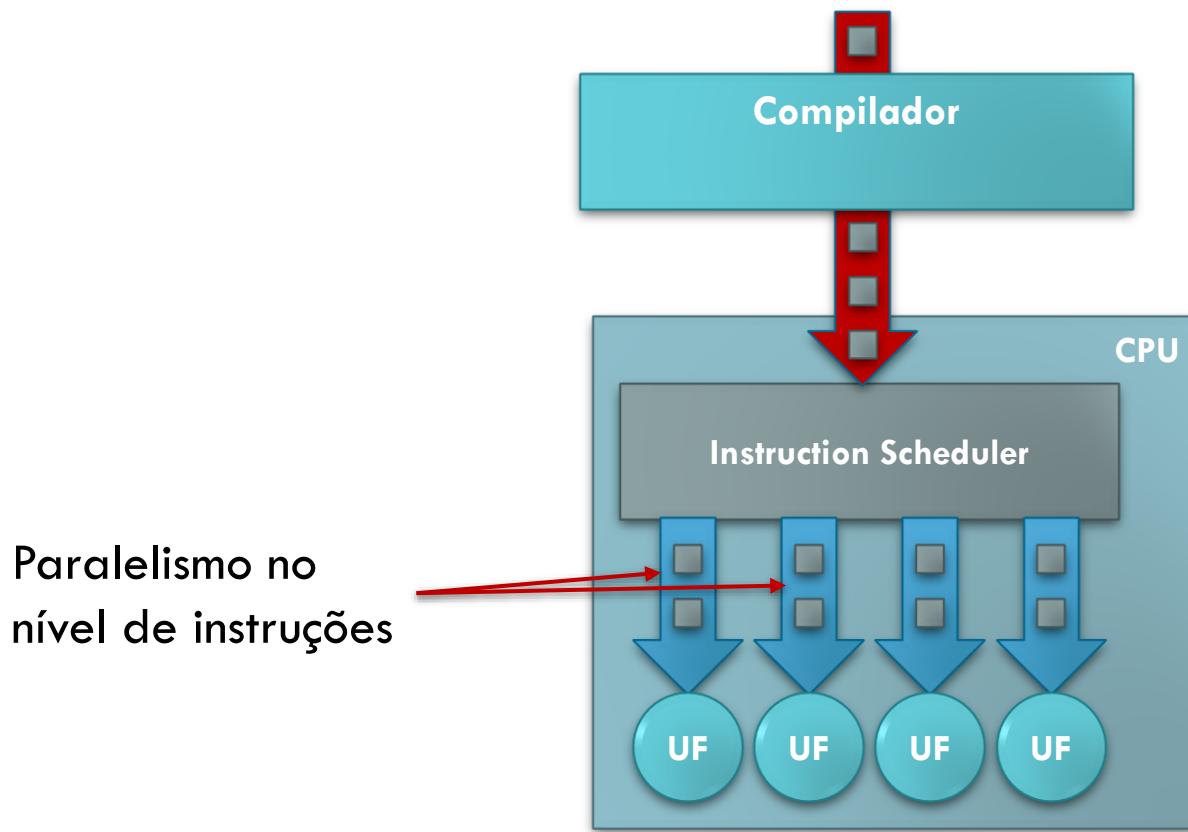
Início do Paralelismo (**ILP – Instruction Level Parallelism**)

Busca, decodificação e execução de mais de uma instrução por ciclo.

Paralelismo agressivo com despacho e execução fora-de-ordem (OOO).



SINGLE CORE - PIPELINE SUPERSCALAR O³ (OUT-OF-ORDER)



O QUE É MAIS RÁPIDO?

$A = B * 4;$
 $C = D * 4;$

$A = B + B;$
 $A = A + B;$
 $A = A + B;$
 $C = D + D;$
 $C = C + D;$
 $C = C + D;$

O QUE É MAIS RÁPIDO?

A = B * 4;
C = D * 4;

A = B + B;
A = A + B;
A = A + B;
C = D + D;
C = C + D;
C = C + D;

A = B << 2;
C = D << 2;

COMO PODEMOS MELHORAR O DESEMPENHO DE UM ÚNICO PROCESSADOR?

Reducindo a latência das instruções

- Usar variáveis de 64 bits (8bytes) apenas quando estritamente necessário
- Preferir instruções simples, sempre que possível

Reducindo a dependência de dados

- Pensar em algoritmos que não tenha muitas dependências de dados (ex. Array vs. Lista encadeada)

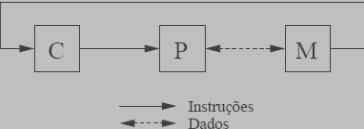
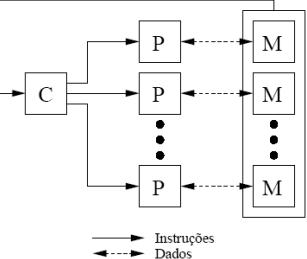
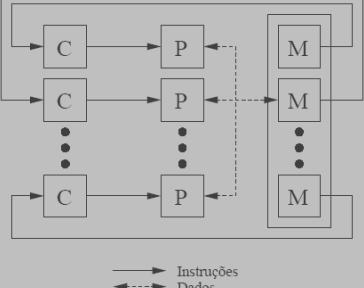
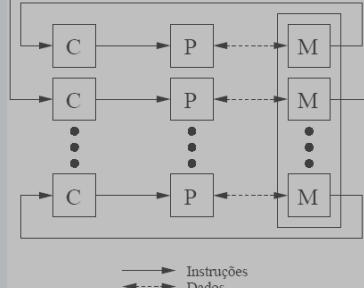
Reducindo a dependência de controle

- Evitando usar muitos IF's (ex. remover if's de dentro de laços)
- Evitando usar muitos Switch-Case
- Simplificando as condições (ex. colocar a primeira condição do IF, aquela que for mais restritiva.

TAXONOMIA DE FLYNN

*[Flynn, 1972]

**[De Rose,
2003]

*	SD (Single Data)	MD (Multiple Data)
SI (Single Instruction)	<ul style="list-style-type: none">**  <p>SISD (Máquinas von Neumann)</p>	<ul style="list-style-type: none">**  <p>SIMD (Máquinas Vetoriais)</p>
MI (Multiple Instruction)	<ul style="list-style-type: none">**  <p>MISD (Sem representante / Arquiteturas Sistólicas)</p>	<ul style="list-style-type: none">**  <p>MIMD (Multiprocessadores e Multicomputadores)</p>

MÁQUINAS SIMD

Processador opera sobre vetores de dados

Controle único

- 1 Contador de programa
- 1 Bloco de controle
- 1 Instrução sendo executada

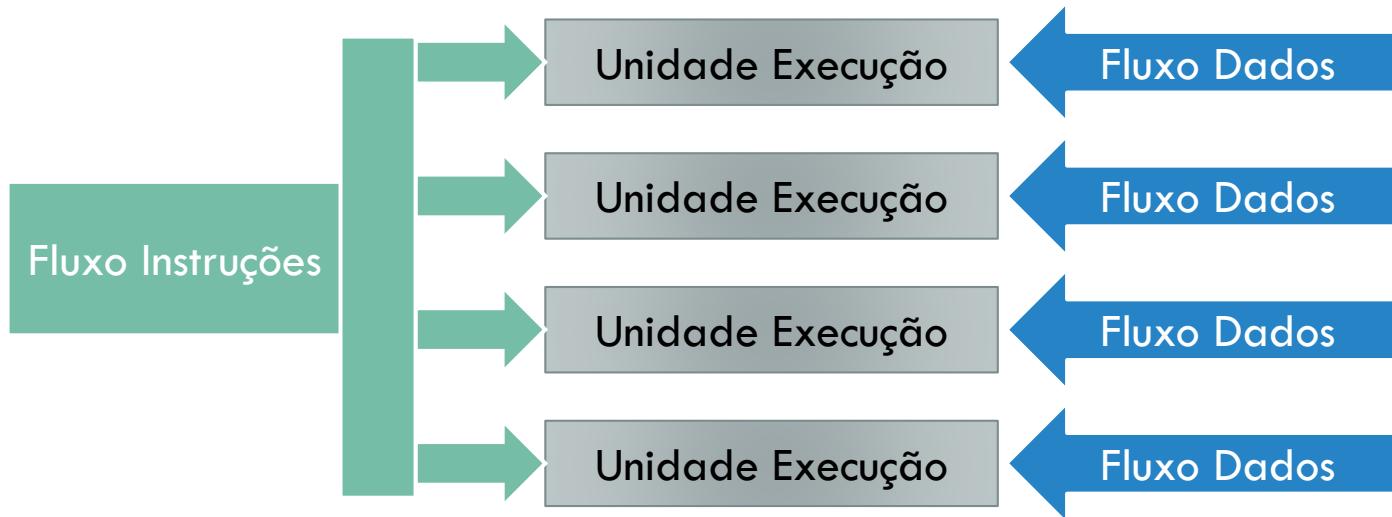
Múltiplas unidades de execução (blocos operacionais)

- ALU
- Registradores de dados
- Registradores de endereço
- Memória de dados local
- Interconexões com unidades vizinhas

MÁQUINAS SIMD (2)

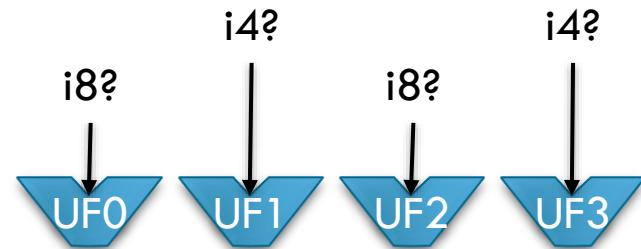
Processador hospedeiro SISD

- Executa operações sequenciais
- Calcula endereços
- Acessa memória de instruções



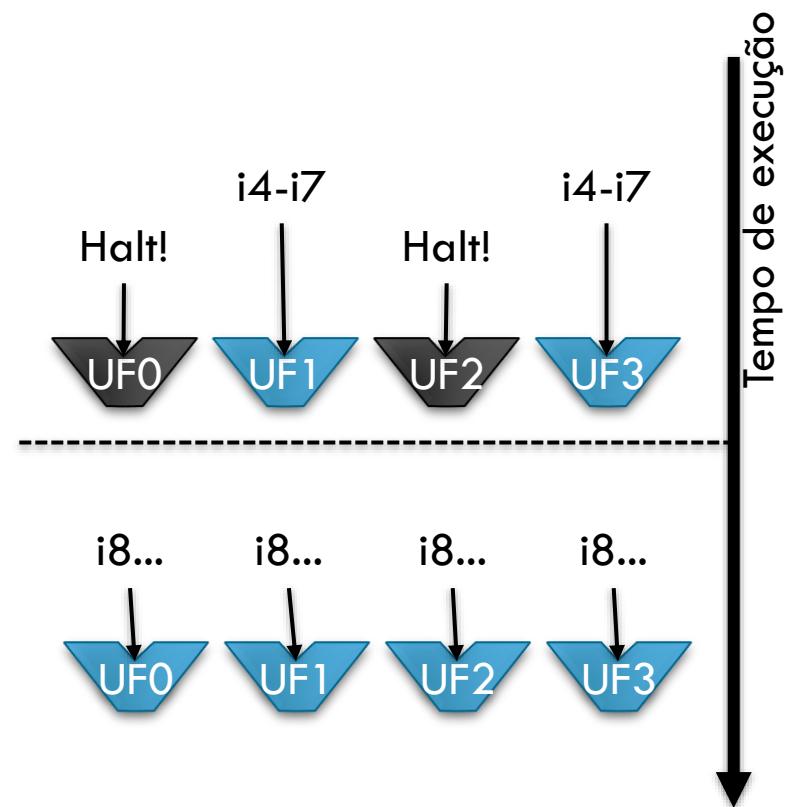
EFICIÊNCIA DE ARQUITETURAS SIMD

i0
i1
i2
i3 (br) if ($id \% 2 == 0$) goto i8
i4
i5
i6
i7
i8
i9
i10



EFICIÊNCIA DE ARQUITETURAS SIMD

i0
i1
i2
i3 (br) if ($id \% 2 == 0$) goto i8
i4
i5
i6
i7
i8
i9
i10



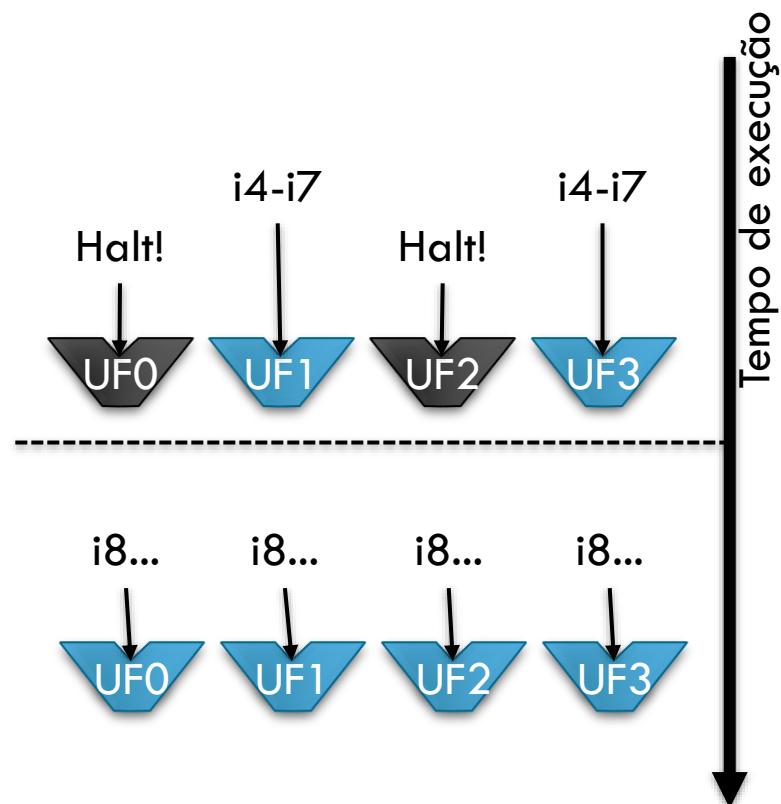
EFICIÊNCIA DE ARQUITETURAS SIMD

SIMD são eficientes quando processam arrays em laços “for”

- Eficientes quando aplicação tem paralelismo de dados massivo

SIMD são ineficientes em aplicações do tipo “case”

- Cada unidade de execução executa operação diferente, dependendo do dado



PROCESSADORES VETORIAIS

Caso aplicado de máquinas SIMD

“processador vetorial” com pipeline associado a um processador hospedeiro escalar convencional

Processador vetorial ...

- Busca dados vetoriais de “registradores vetoriais” ou diretamente da memória
- Executa operações sobre os vetores através de 1 ou mais pipelines funcionais paralelos

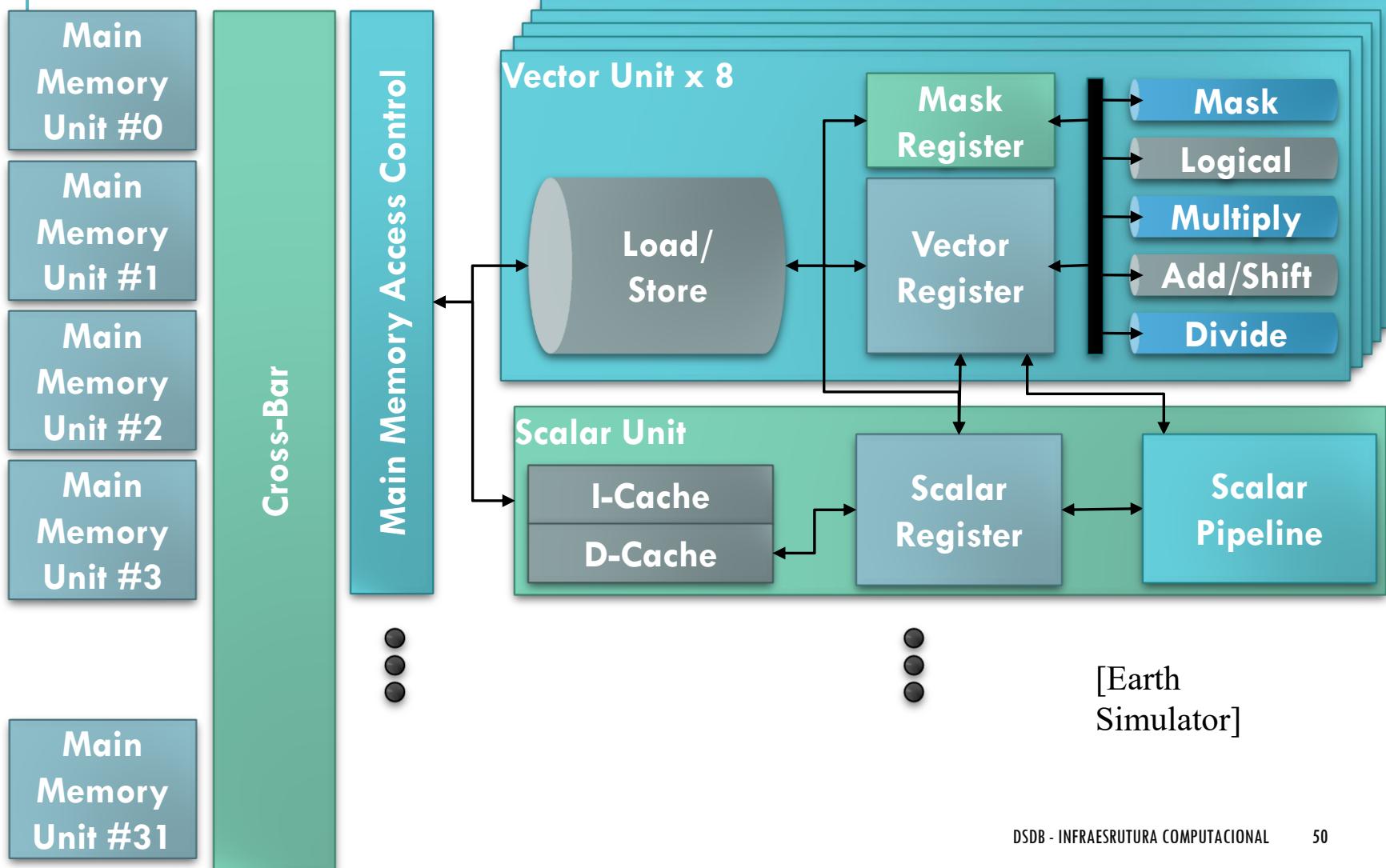
Exemplo: Cray C-90

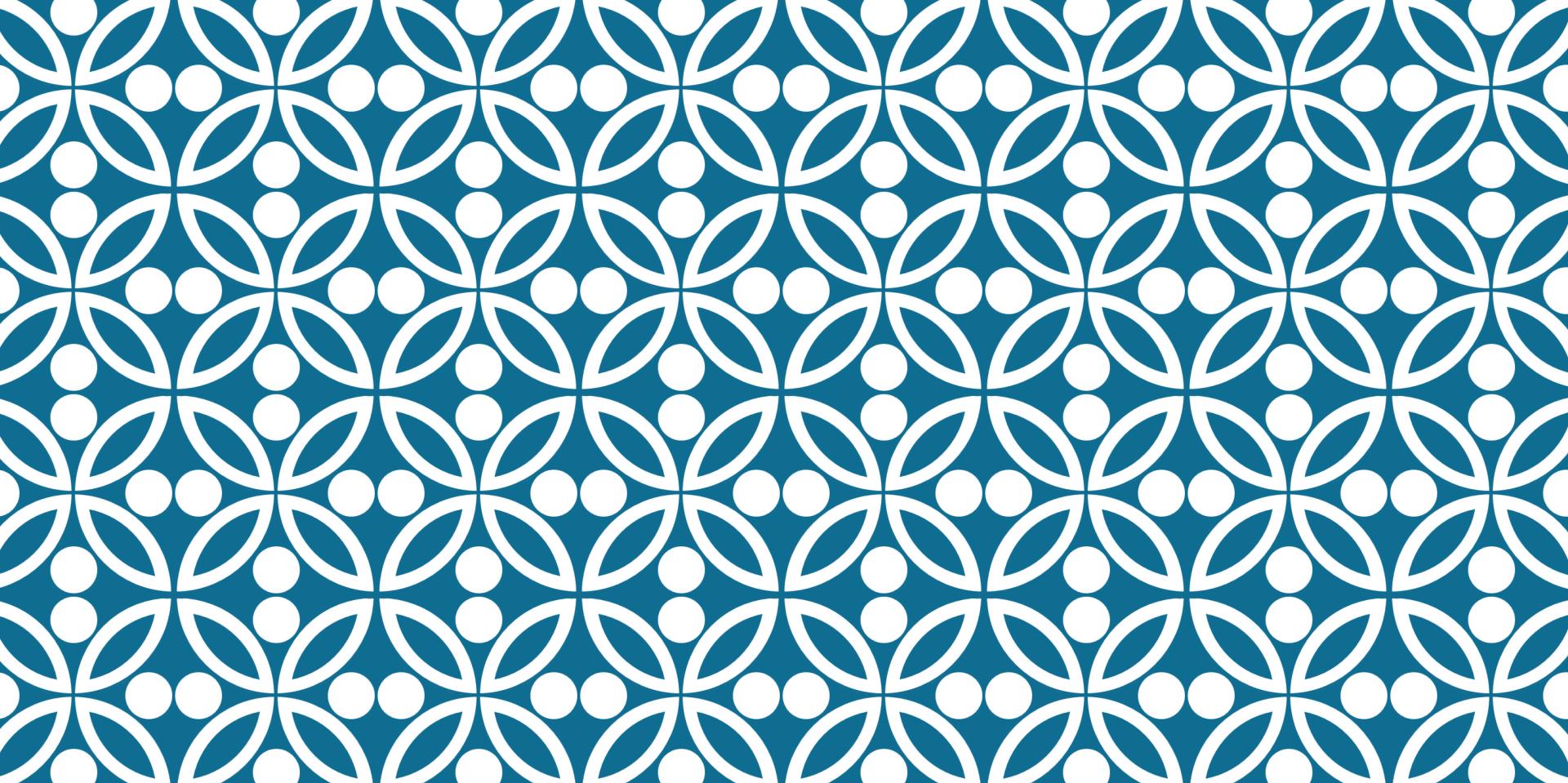
- Registrador vetorial tem 64 x 64 bits
- 2 pipelines funcionais vetoriais

Outras máquinas vetoriais

- Convex C3, DEC VAX 9000, Fujitsu VP2000, Hitachi S-810, IBM 390/VF

PROCESSADORES VETORIAIS

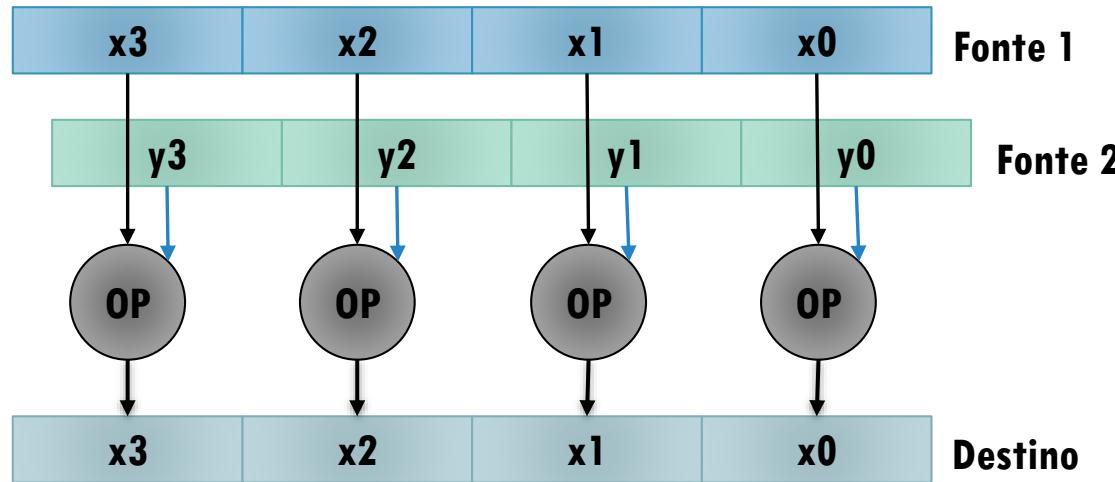




INSTRUÇÕES SIMD

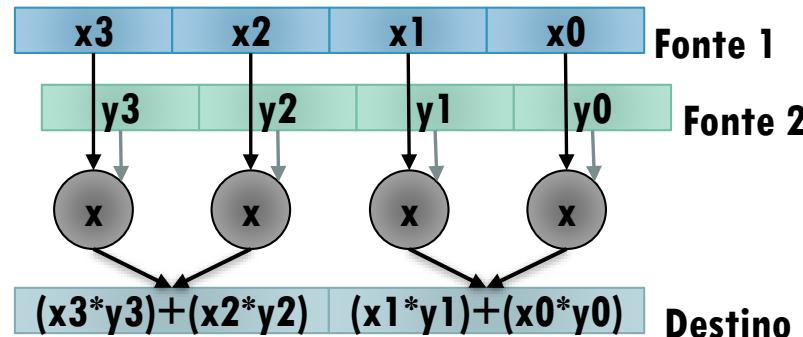
INSTRUÇÕES SIMD

Máquinas SISD/SIMD têm uma mistura de instruções SISD e SIMD

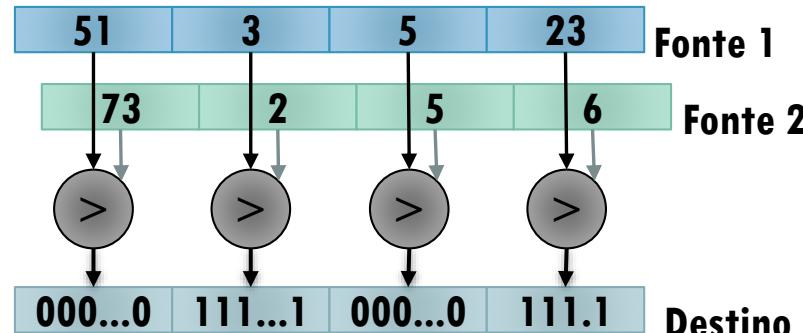


EXEMPLOS

Mult/Add



Compares

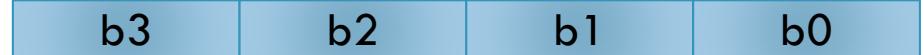


INTEL MMX – 8X64 BITS REGISTRADORES

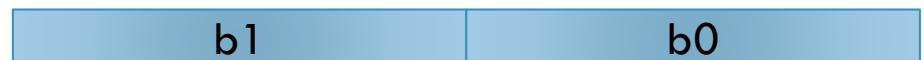
(8bits x 8) Packed Bytes



(16bits x 4) Packed Words



(32bits x 2) Packed Doublewords

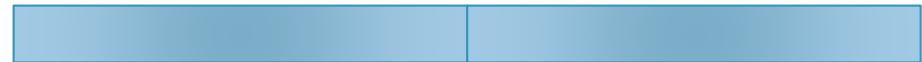


(64bits x 1) Packed Quadword



INTEL SSE - 8X128 BITS REGISTRADORES

128-Bit Packed Double-Precision FP



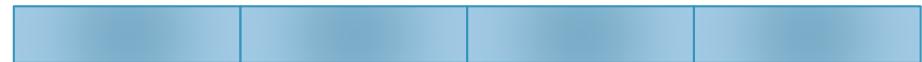
128-Bit Packed Byte Integers



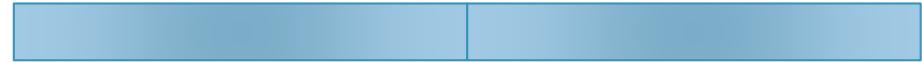
128-Bit Packed Word Integers



128-Bit Packed Doubleword Integers



128-Bit Packed Quadword Integers



REGISTRADORES X86 (INTEL/AMD)

Extensão Multimedia e
Registradores de Ponto-Flutuante

MM0/ST0
MM1/ST1
MM2/ST2
MM3/ST3
MM4/ST4
MM5/ST5
MM6/ST6
MM7/ST7

SSE/AVX-128 suportam loads de 128 bits
AVX-256 suporta load de 256 bits
AVX-512 carrega até 512 bits (64 bytes!!!)

Fluxo SIMD
Registradores de Extensão SSE

XMM0
XMM1
XMM2
XMM3
XMM4
XMM5
XMM6
XMM7
XMM8
XMM9
XMM10
XMM11
XMM12
XMM13
XMM14
XMM15

USANDO OPERAÇÕES VETORIAIS COM INTRINSICS



- Technologies**
- MMX
 - SSE
 - SSE2
 - SSE3
 - SSSE3
 - SSE4.1
 - SSE4.2
 - AVX
 - AVX2
 - FMA
 - AVX-512
 - KNC
 - SVML
 - Other

- Categories**
- Application-Targeted
 - Arithmetic
 - Bit Manipulation
 - Cast
 - Compare
 - Convert
 - Cryptography
 - Elementary Math
- Functions**
- General Support
 - Load
 - Logical
 - Mask
 - Miscellaneous

_mm_search

`__m128 _mm_add_ps (__m128 a, __m128 b)`

Synopsis

```
__m128 _mm_add_ps (__m128 a, __m128 b)
#include "xmmintrin.h"
Instruction: addps xmm, xmm
CPUID Flags: SSE
```

Description

Add packed single-precision (32-bit) floating-point elements in `a` and `b`, and store the results in `dst`.

Operation

```
FOR j := 0 to 3
    i := j*32
    dst[i+31:i] := a[i+31:i] + b[i+31:i]
ENDFOR
```

Performance

Architecture	Latency	Throughput (CPI)
Skylake	4	0.5
Broadwell	3	1
Haswell	3	1
Ivy Bridge	3	1

`__m128 _mm_add_ss (__m128 a, __m128 b)`

`__m128 _mm_div_ps (__m128 a, __m128 b)`

`__m128 _mm_div_ss (__m128 a, __m128 b)`

`__m128 _mm_mul_ps (__m128 a, __m128 b)`

`__m128 _mm_mul_ss (__m128 a, __m128 b)`

`__m64 _mm_mulhi_pu16 (__m64 a, __m64 b)`

`__m64 _m_pmulhw (__m64 a, __m64 b)`

`__m64 _m_psadbw (__m64 a, __m64 b)`

USANDO OPERAÇÕES VETORIAIS COM INTRINSICS

```
__m128 _mm_add_ps (__m128 a, __m128 b)
```

Synopsis

- `_mm_add_ps` (`_m128 a, _m128 b`)
- `#include "xmmintrin.h"`
- Instruction: `addps` `xmm, xmm`
- CPUID Flags: SSE

Description:

- Add packed single-precision (32-bit) floating-point elements in `a` and `b`, and store the results in `dst`.

Operation

- FOR `j := 0` to 3
 - `i := j*32`
 - `dst[i+31:i] := a[i+31:i] + b[i+31:i]`
- ENDFOR

Performance		
Architecture	Latency	Throughput (CPI)
Skylake	4	0.5
Broadwell	3	1
Haswell	3	1
Ivy Bridge	3	1

USANDO OPERAÇÕES VETORIAIS COM INTRINSICS

```
#include "xmmintrin.h"

void main() {
    __m128 a, b, c;
    float va[128], vb[128], vc[128];

    for (int i=0; i<128; i+=4) {
        a = _mm_load_ps(&va[i]);
        b = _mm_load_ps(&vb[i]);
        c = _mm_add_ps(a, b);
        _mm_store_ps(&vc[i], c);
    }
}
```

Performance		
Architecture	Latency	Throughput (CPI)
Skylake	4	0.5
Broadwell	3	1
Haswell	3	1
Ivy Bridge	3	1

COMO PODEMOS MELHORAR O DESEMPENHO VETORIAL DE UM PROCESSADOR?

Reducindo a latência das instruções

- Usar variáveis de 64 bits (8bytes) apenas quando estritamente necessário
- Podemos desenrolar o laço manualmente (ex. `for (i=0; i<MAX; i+=4)`)

Reducindo a dependência de dados

- Pensar em algoritmos que não tenha muitas dependências de dados (ex. Array vs. Lista encadeada)
- Removendo dependência entre iterações de um laço de repetição

Reducindo a dependência de controle

- Evitando usar muitos IF's (ex. remover if's de dentro de laços)

TAXONOMIA DE FLYNN

*[Flynn, 1972]

**[De Rose,
2003]

*	SD (Single Data)	MD (Multiple Data)
SI (Single Instruction)	<ul style="list-style-type: none">** <p>SISD (Máquinas von Neumann)</p>	<ul style="list-style-type: none">** <p>SIMD (Máquinas Vetoriais)</p>
MI (Multiple Instruction)	<ul style="list-style-type: none">** <p>MISD (Sem representante / Arquiteturas Sistólicas)</p>	<ul style="list-style-type: none">** <p>MIMD (Multiprocessadores e Multicomputadores)</p>

PROCESSADORES SISTÓLICOS

Considerado por alguns autores como máquinas MISD

Processamento “data-flow”

- Cada processador executa operação quando dados de entrada estão disponíveis

Processadores elementares

- Executam operação única, não programáveis

Aplicações em processamento digital de sinais

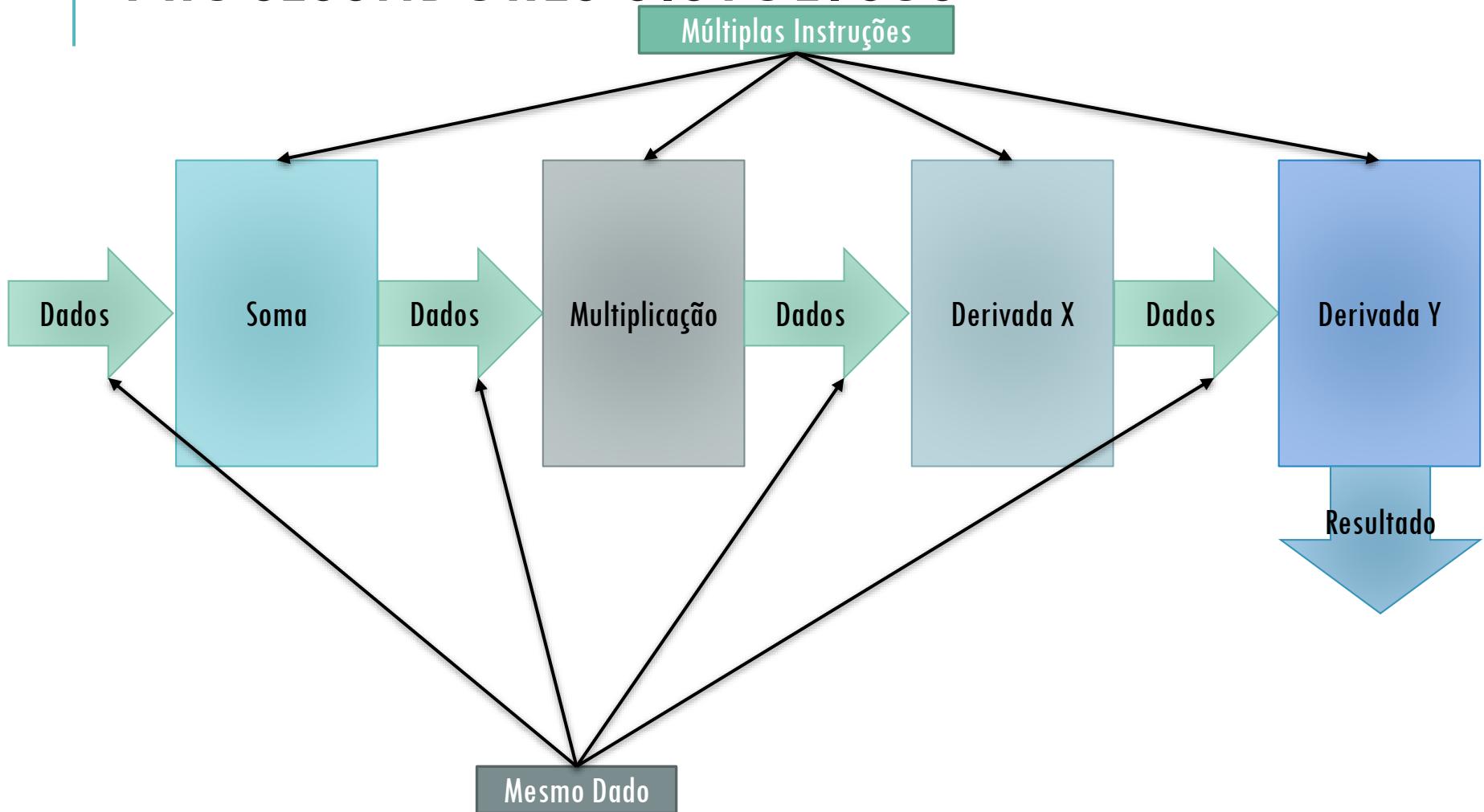
- Processamento de imagens, voz, ...

Integração num único chip

São utilizados em:

- Em sistemas eletrônicos dedicados
- Como unidade funcional especializada de um processador hospedeiro convencional

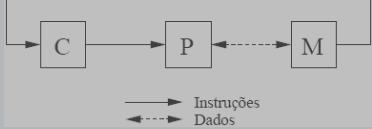
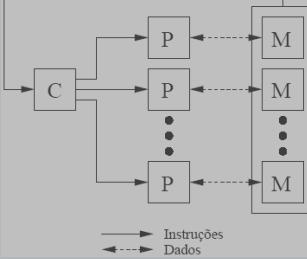
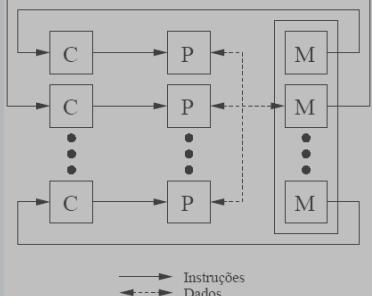
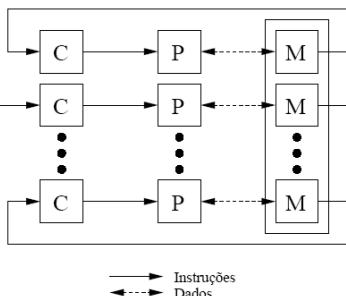
PROCESSADORES SISTÓLICOS



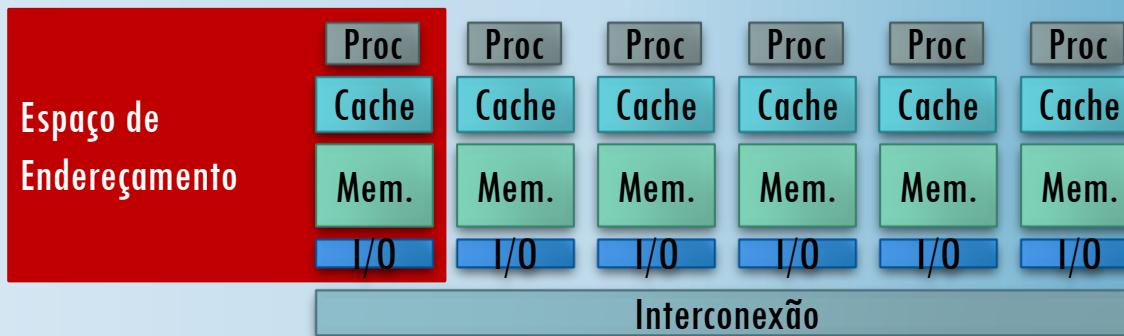
TAXONOMIA DE FLYNN – MIMD

*[Flynn, 1972]

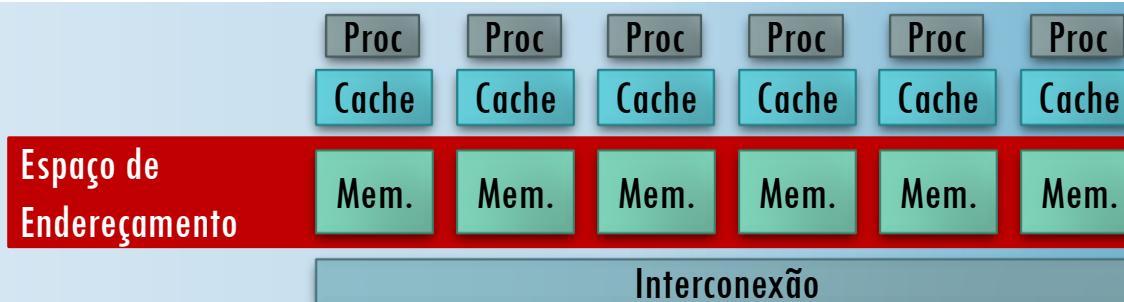
**[De Rose, 2003]

	SD (Single Data)	MD (Multiple Data)
SI (Single Instruction)	 <p>SISD (Máquinas von Neumann)</p>	 <p>SIMD (Máquinas Array)</p>
MI (Multiple Instruction)	 <p>MISD (Sem representante até agora / Arquiteturas Sistólicas)</p>	 <p>MIMD (Multiprocessadores e Multicomputadores)</p>

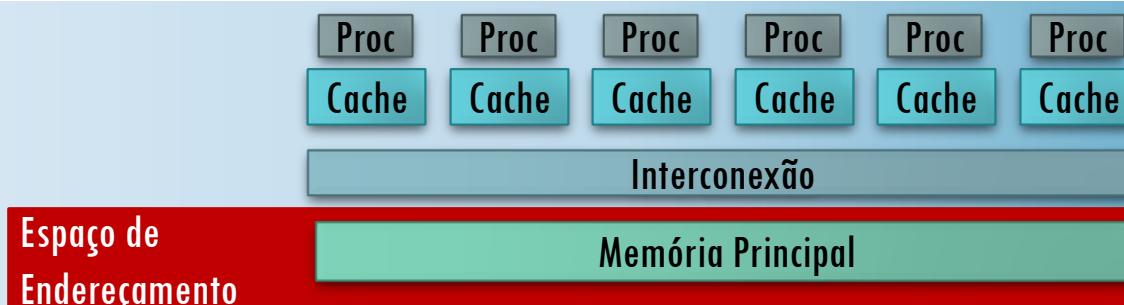
MULTI-PROCESSORS



**Multicomputadores
Cluster de Máquinas**



**Multiprocessadores
Máquina NUMA**



**Multiprocessadores
Máquina UMA**

ESPAÇO DE ENDEREÇAMENTO PRIVADO VS. COMPARTILHADO

Espaço privado

Cada processador tem sua visão da memória.

Cada um terá sua variável privada.

Comunicação através da troca de mensagens (mais lento).

Normalmente não ocorrerá condições de corrida.

Espaço compartilhado

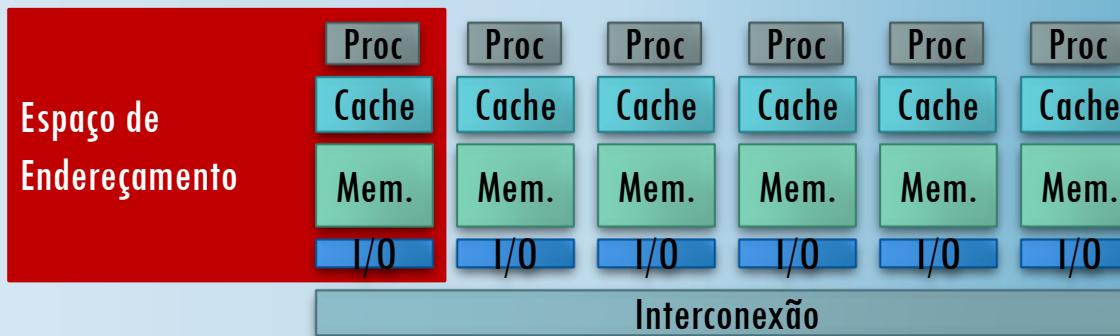
Todos os processadores tem a mesma visão da memória.

Mesma variável pode ser visível para todos.

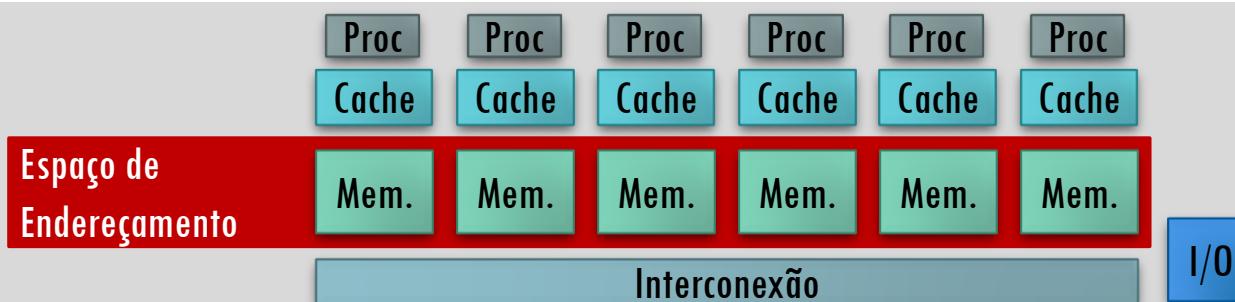
Comunicação por variável compartilhada (mais rápido).

Haverá naturalmente condições de corrida (recursos compartilhados).

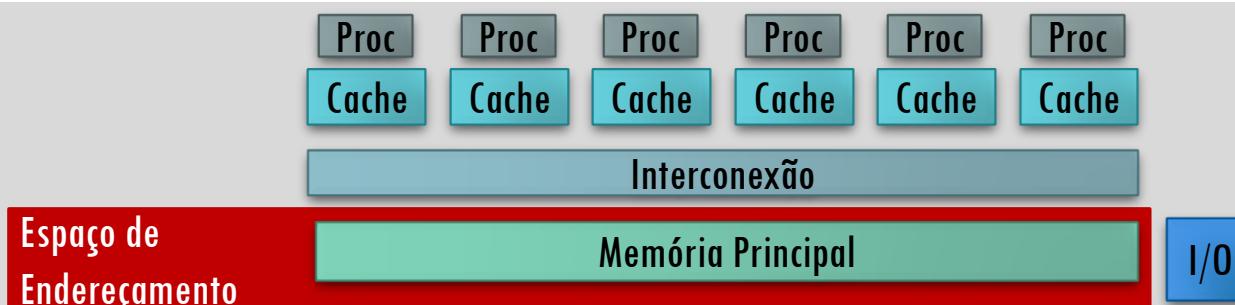
MULTI-PROCESSORS



**Multicomputadores
Cluster de Máquinas**



**Multiprocessadores
Máquina NUMA**



**Multiprocessadores
Máquina UMA**

CLUSTER DE MÁQUINAS

Conhecidos como COW/NOW: Cluster/Network of Workstations.

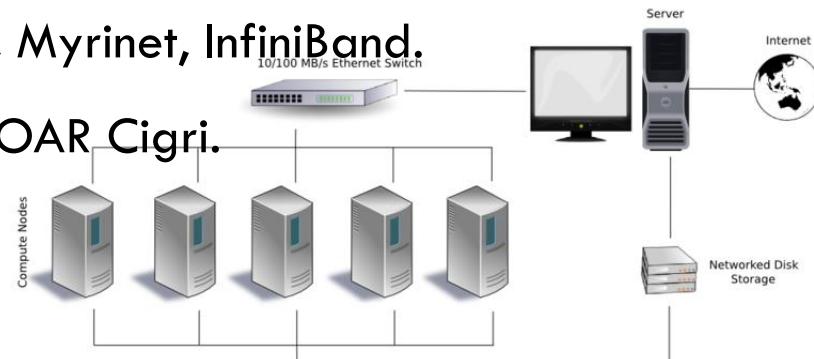
Paralelismo: Bastante claro em nível de processos.

Diversos espaços de endereçamento.

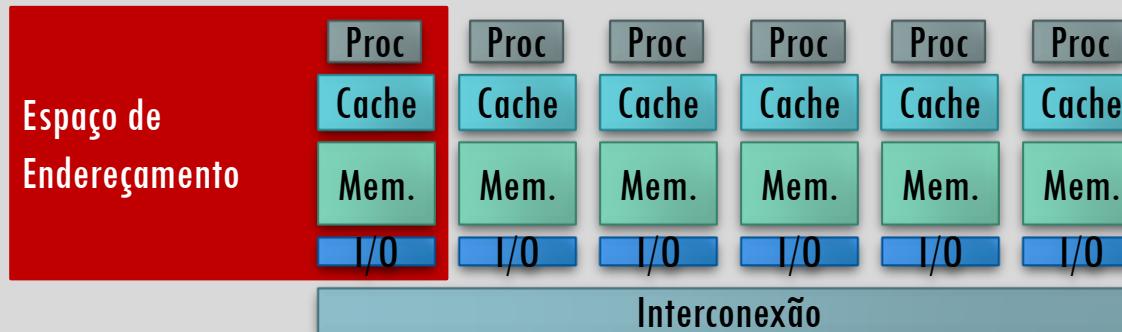
SO para cada máquina: Linux ou proprietário.

Tecnologias de Rede: Ethernet, Myrinet, InfiniBand.

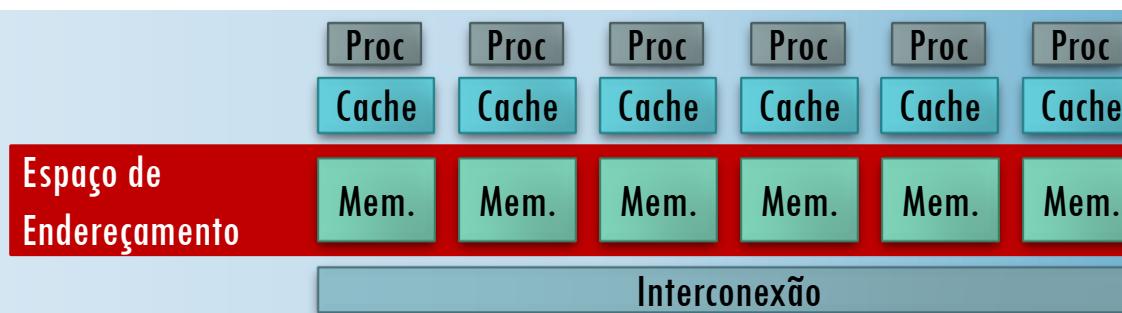
Escalonador de Jobs: Globus, OAR, Cigri.



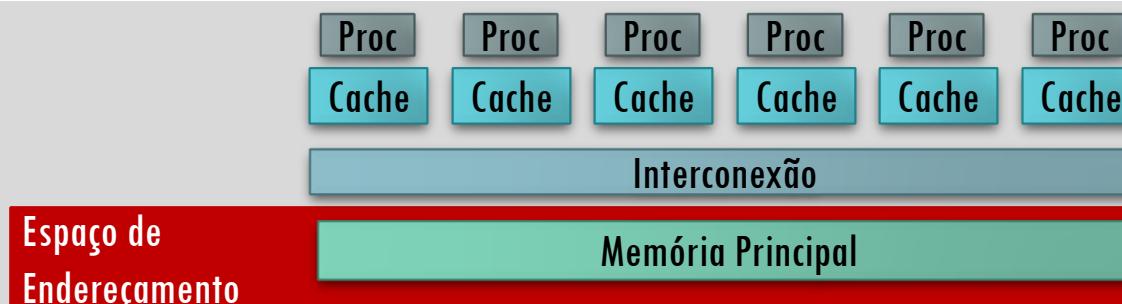
MULTI-PROCESSORS



**Multicomputadores
Cluster de Máquinas**



**Multiprocessadores
Máquina NUMA**



**Multiprocessadores
Máquina UMA**

MÁQUINAS NUMA

Na arquitetura NUMA as partições dos Cores e as memórias são agrupadas em Nós

Uma máquina pode ter vários Nós NUMA, dependendo do modelo e quantidade de memória/processadores

A latência de acesso de um processador com a memória dentro de um Nós NUMA é muito baixa, pois o barramento torna eficiente o acesso dentro do mesmo Nós

Entretanto o acesso do processador de um Nós X para a memória em um Nós Y é maior pois é necessário atravessar a interconexão que liga os diversos Nós

Pode parecer insignificante esta penalidade entretanto em acessos intensivos de memória a Nós remotos pode ter grande impacto no desempenho do sistema

A razão da diferença de latência entre o nodo local e um nodo remoto é chamado de **NUMA factor**.

MÁQUINAS NUMA MAPEAMENTO DE ENDEREÇOS

Diferentes tipos de mapeamento de endereço para os nodos NUMA podem ser adotados:

Entrelaçado			
0	1	2	3
4	5	6	7
8	9	10	11
...

Linear			
0	10	20	30
1	11	21	31
2	12	22	32
...

Bloco Entrelaçado				
0	5	10	15	
~	~	~	~	
4	9	14	19	
20

MÁQUINAS NUMA

Exemplo de NUMA com capacidade total de 400 páginas de memória

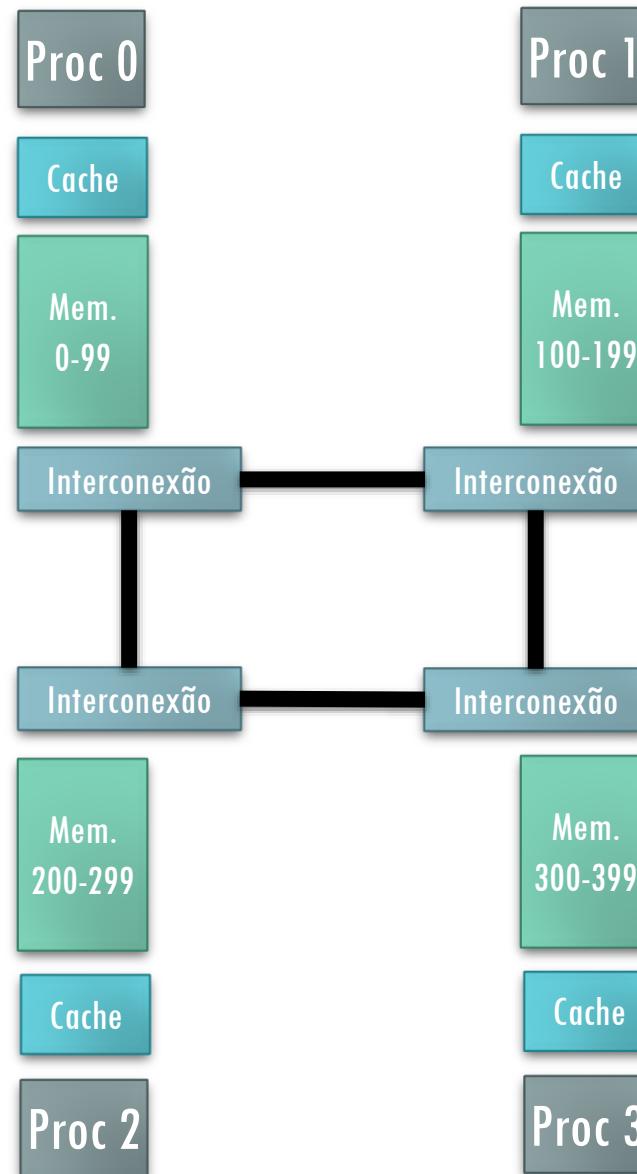
Cada nó contém um quadro de páginas contíguo

Considerando as latências:

- Caches/Memória igual a **m ciclos**
- Interconexão/link igual a **n ciclos**

O custo de acesso do Proc 0 acessar:

- Páginas(0~99) = ***m ciclos***
- Páginas(100~299) = ***n + m ciclos***
- Páginas(300~399) = ***2n + m ciclos***



MÁQUINAS NUMA NON-UNIFORM MEMORY ACCESS

COMA (Cache Only Memory Architecture)

- Formado com memórias cache de alta capacidade.
- Coerência é obtida em hardware com a atualização simultânea em múltiplos nós dos dados.

CC-NUMA (Cache Coherent NUMA)

- Coerência de cache é garantida pelo hardware.

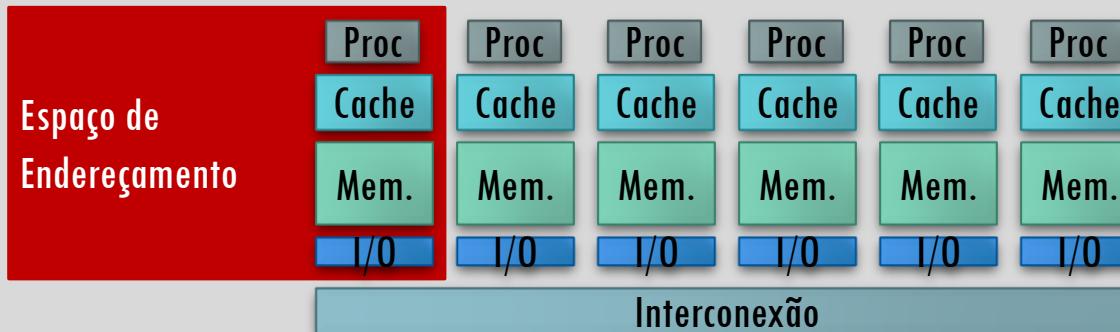
NCC-NUMA (Non-Cache Coherent NUMA)

- Não existe garantia de coerência da cache ou não existe cache.

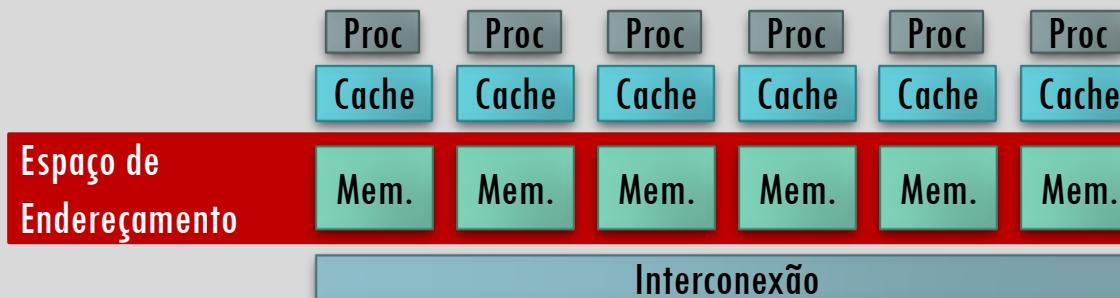
SC-NUMA (Software Coherent NUMA) /DSM.

- Coerência de cache é garantida em software.
- NORMA ou NCC-NUMA com coerência por software.

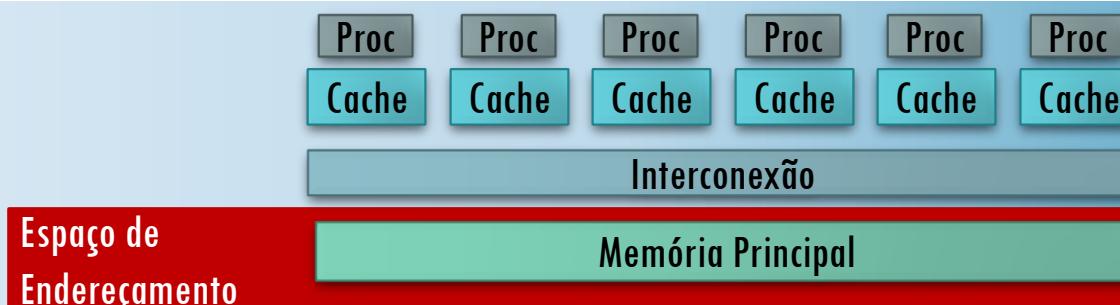
MULTI-PROCESSORS



**Multicomputadores
Cluster de Máquinas**



**Multiprocessadores
Máquina NUMA**



**Multiprocessadores
Máquina UMA**

CONTEXTO DA MUDANÇA SINGLE-CORE → MULTI-CORE

Instruction Parallelism **wall**

- Programas tem um limite para extração de ILP;

Clock **wall**

- Difícil aumentar a profundidade do Pipeline; *
- Wire-Delay – O clock muda de sinal antes de se propagar/estabilizar;

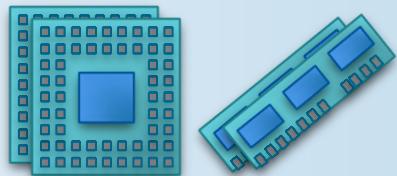
Power **wall**

- Leakage aumenta 7.5x a cada nova geração;**
- Bloco de Controle muito complexo;

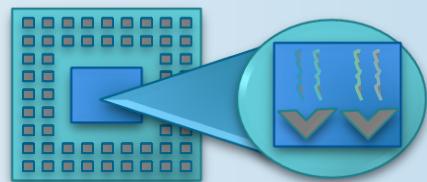
*[Agarwal, 2000]

**[Borkar, 1999]

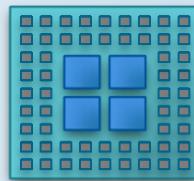
MÁQUINAS UMA UNIFORM MEMORY ACCESS



Multiprocessors



> **Chip Multithreading**
IMT, BMT, SMT



> **Chip Multiprocessor**
Multi-Core, Many-Core, GPGPU

MULTITHREADING

A troca de contexto tradicionalmente envolve

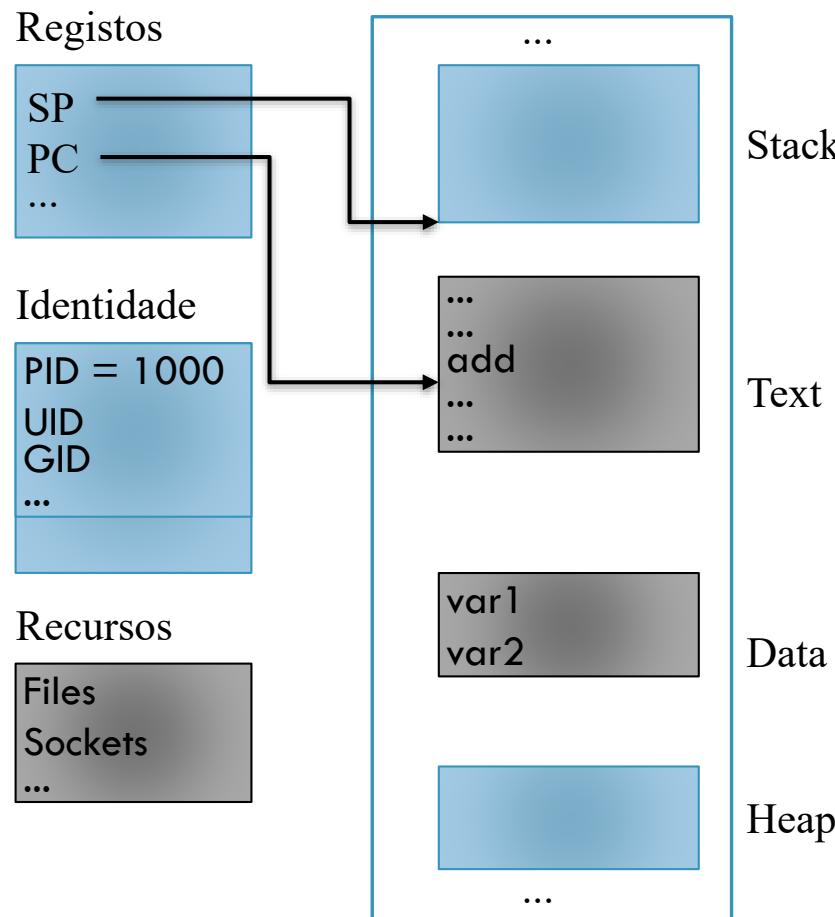
- O despejo do programa A, escrita na memória dos registradores do prog. A
- A carga do programa B, leitura da memória para carregar os registradores do prog. B

Uma das primeiras formas de paralelismo a ser implementada nos processadores comerciais foi o multithreading.

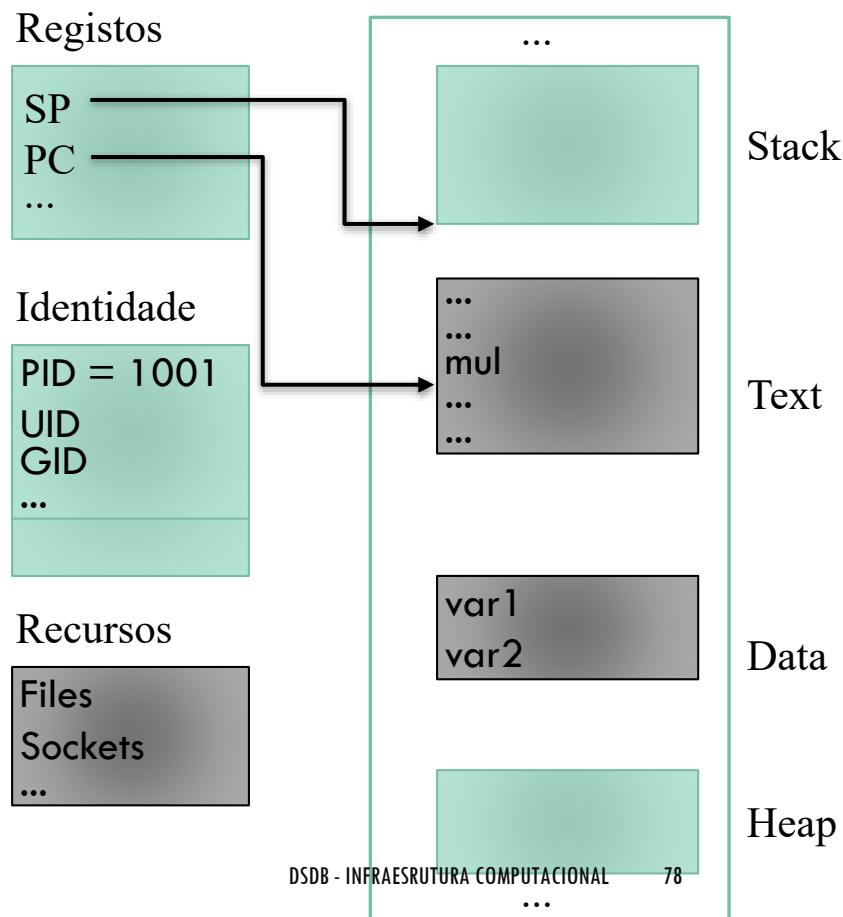
O objetivo foi modificar as arquitetura para dar **suporte a diversas threads ativas**, possibilitando a rápida troca de contexto.

ESTRUTURA DE PROCESSOS

Processo A



Processo B



O QUE É TROCA DE CONTEXTO?

PCB

O procedimento de suspender e continuar processos

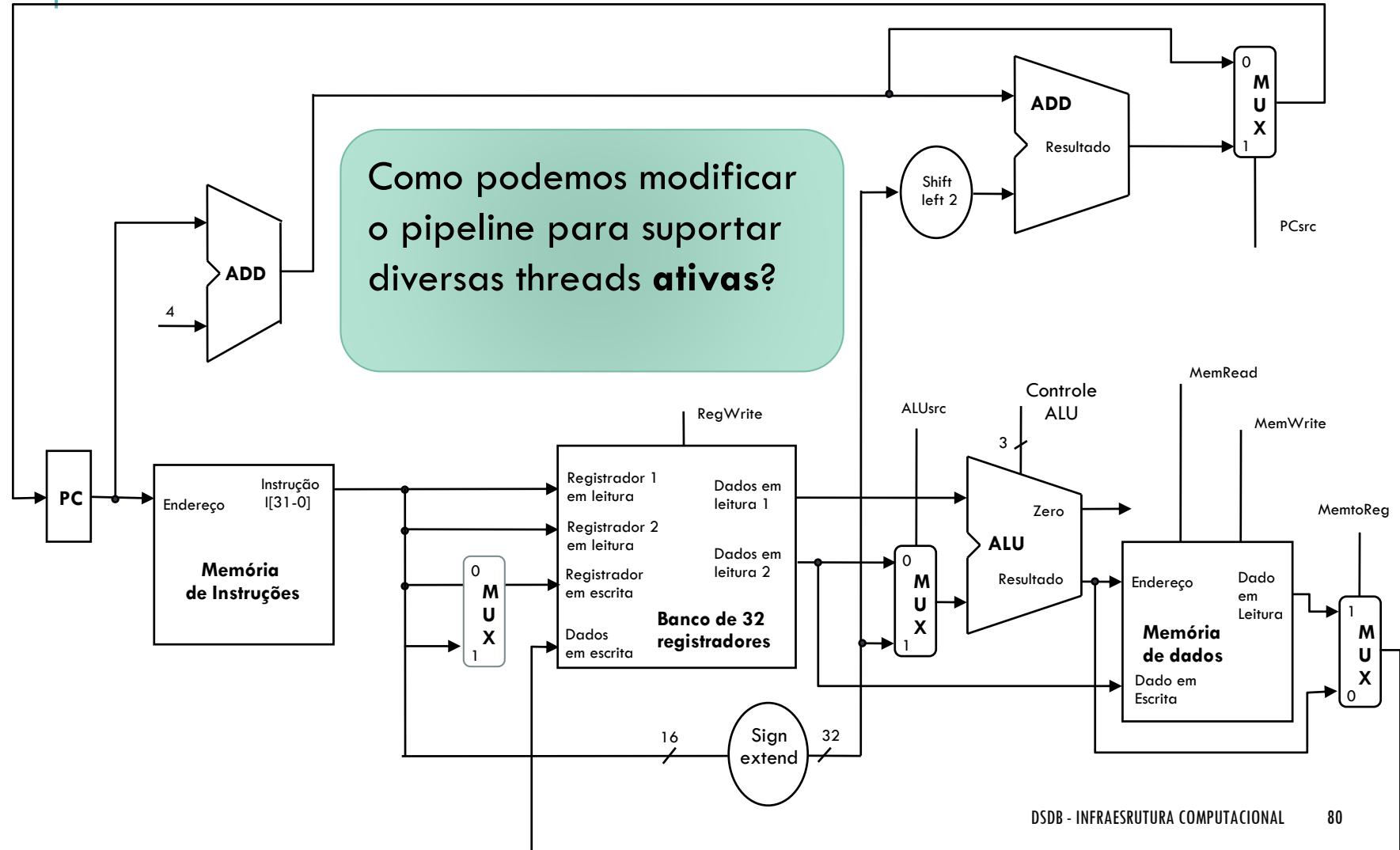
Ativado pela interrupção (ex. Interrupção de I/O), preempção de multi-tasks, ou parte do mode de troca entre kernel/usuário.

O contexto do processo é representado pelo PCB (Process Control Block)

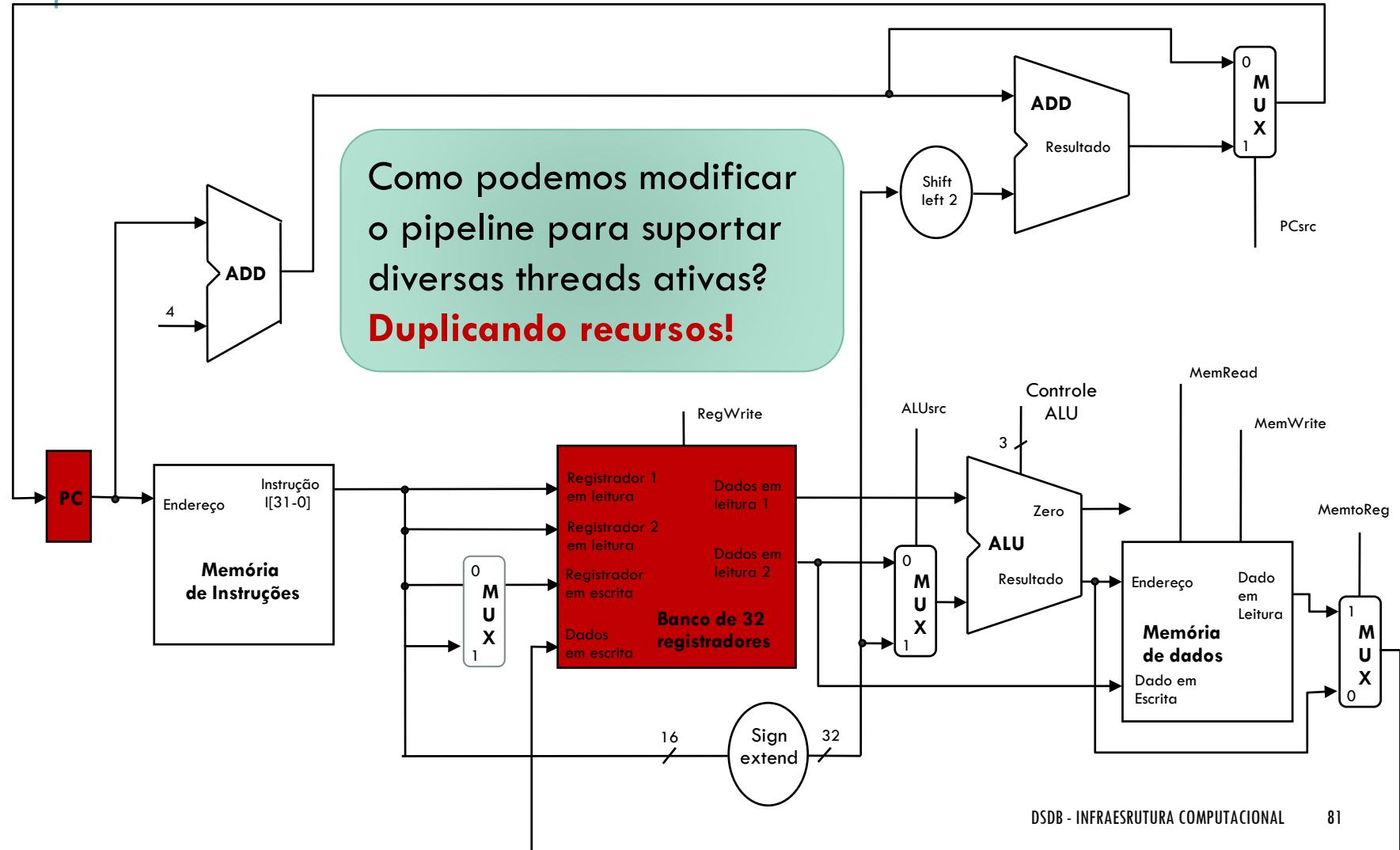
“O PCB é a manifestação de um processo dentro do SO”
[Harvey M. Deitel (1984). An introduction to operating systems]

process state
process number
program counter
registers
memory limits
list of open files
• • •

MULTITHREADING



MULTITHREADING



MULTITHREADING

Não necessariamente teremos diversas threads em execução paralela

O processador deve suportar diversas **threads ativas**

Interleaved Multithreading – IMT

- Apenas **uma thread em execução** por ciclo
- Troca de thread em execução a cada ciclo

Blocked Multithreading – BMT

- Apenas **uma thread em execução** por ciclo
- Troca de thread em execução a cada evento de alta latência (ex. interrupção, acesso memória)

Simultaneous Multithreading – SMT

- **Diversas threads em execução ao mesmo tempo**
- Hyper-threading é o nome comercial usado pela Intel

MULTITHREADING

Uma única thread ativa (single-threading)

- O processador armazena apenas os dados de uma thread
- A troca de contexto entre as threads envolve salvar todo o contexto (registradores) na memória, e depois, carregar o contexto da outra thread
- Apenas uma thread pode estar em execução ao mesmo tempo.

MULTITHREADING

Uma única thread ativa (single-threading)

- O processador armazena apenas os dados de uma thread
- A troca de contexto entre as threads envolve salvar todo o contexto (registradores) na memória, e depois, carregar o contexto da outra thread
- Apenas uma thread pode estar em execução ao mesmo tempo.

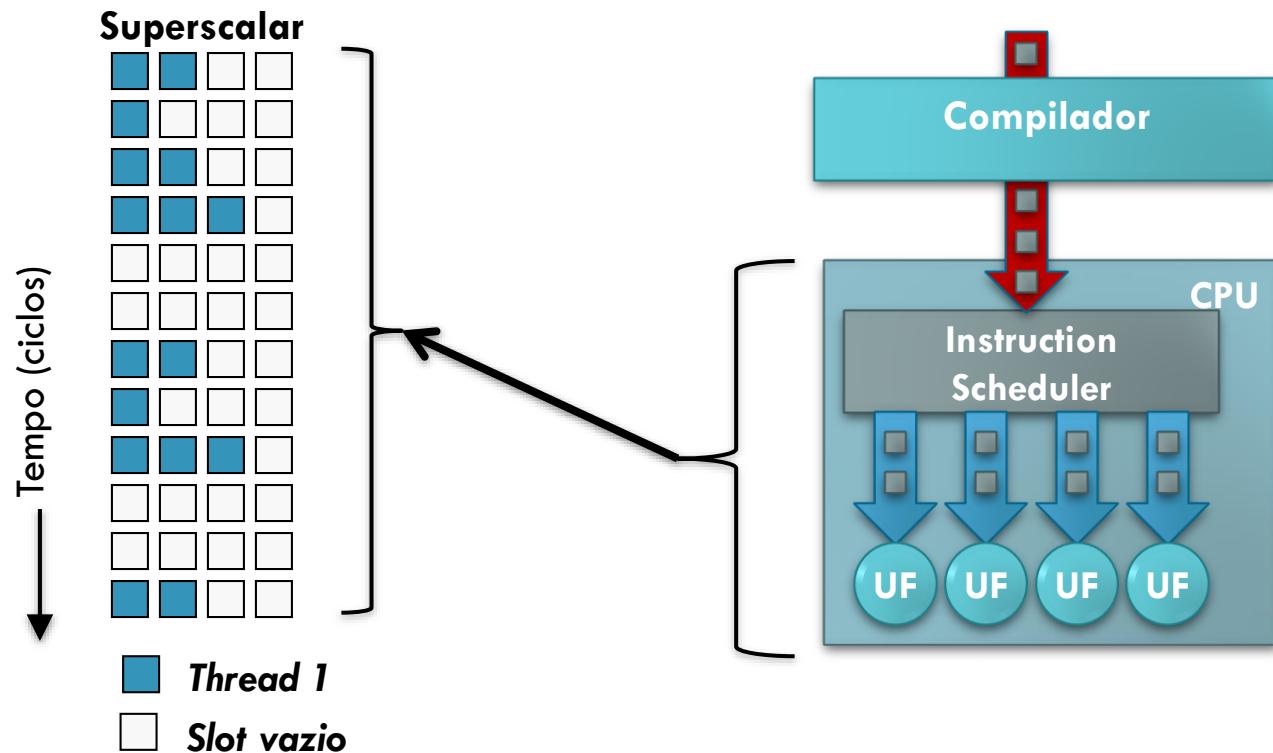
Diversas Threads Ativas

- O processador possui todas as informações sobre diversas threads
- A troca de contexto entre essas threads é feita rapidamente (temos um banco de registradores por thread ativa)
- As threads ativas **podem NÃO** estar em execução ao mesmo tempo.

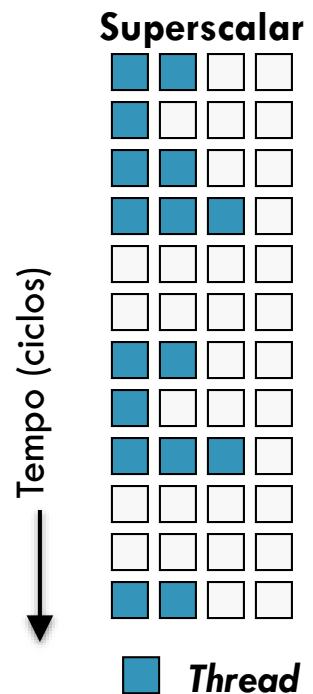
Diversas Threads em Execução

- O processador deve suportar diversas threads ativas
- Porém nesse caso, várias threads **podem** estar ao mesmo tempo no pipeline

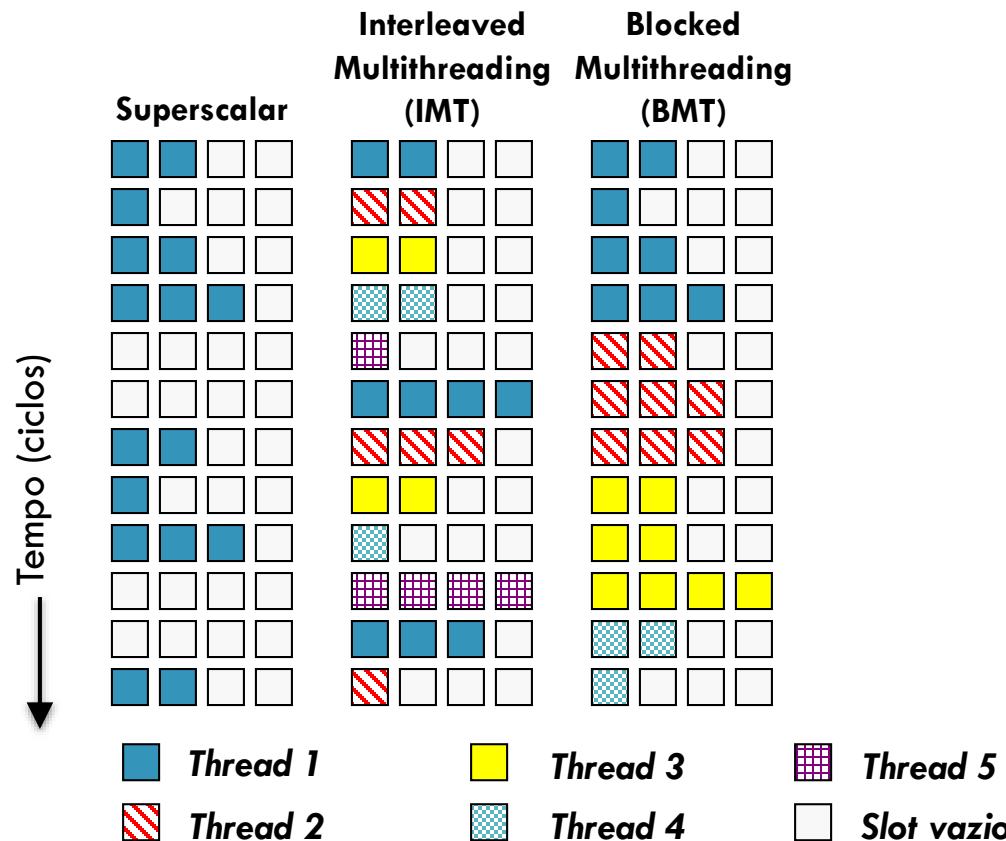
PROCESSADOR SUPERESCALAR



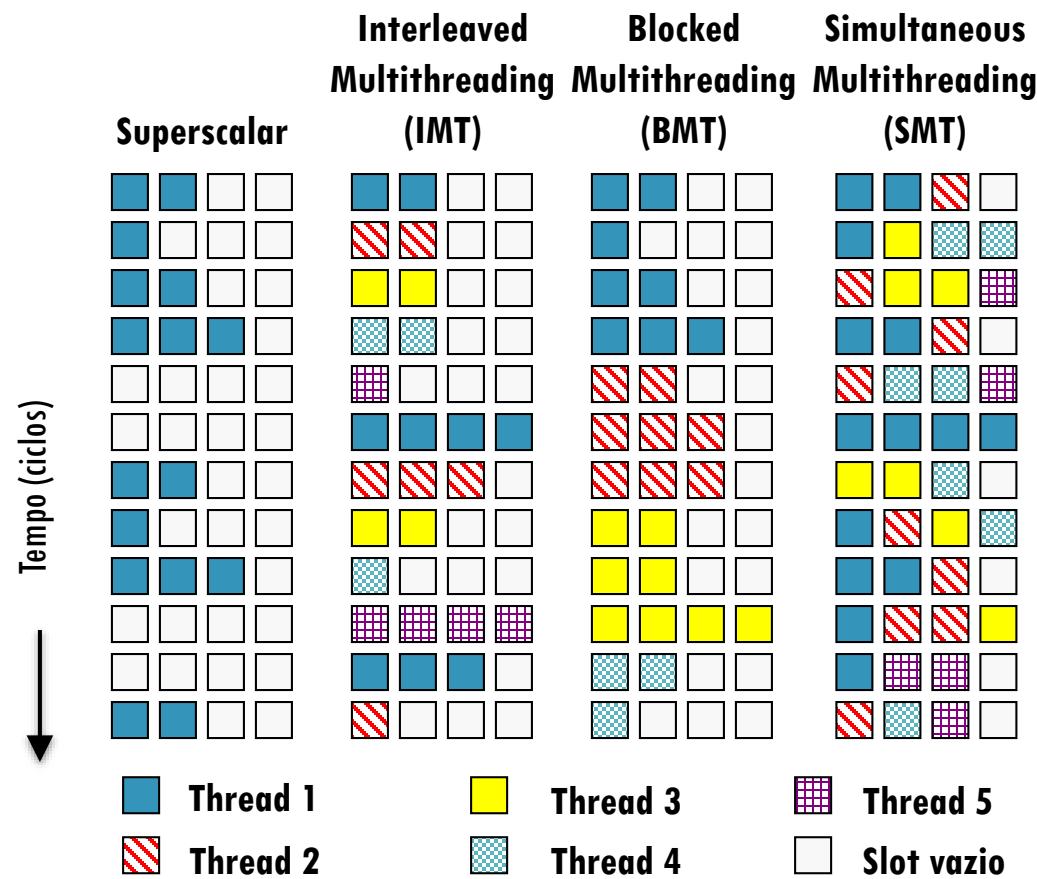
MULTITHREADING



MULTITHREADING



SMT



CHIP MULTITHREADING

IMT, BMT E SMT

Deve suportar várias **threads ativas**.

Diversas threads ativas (IMT/BMT/SMT).

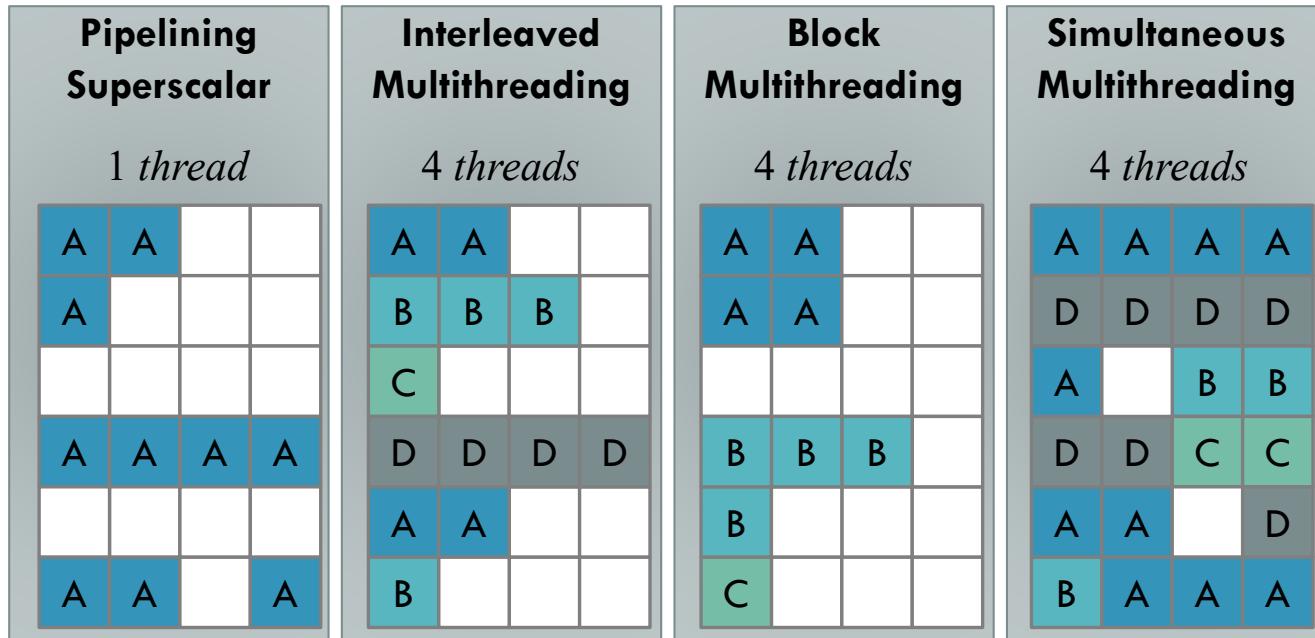
Diversas threads em execução paralela (apenas o SMT).

Único processador físico.

Vários processadores virtuais.

Usa TLP e representa uma alternativa para ILP.

CHIP MULTITHREADING COM PIPELINE SUPERSCALAR



Latência de Memória	Não esconde	Esconde	Esconde	Esconde
Troca de Contextos	Lenta	Rápida	Rápida	Rápida
Utilização UFs	Baixa	Média	Média	Alta
Uso das Caches	Baixa taxa de faltas	Alta taxa de faltas	Alta taxa de faltas	Alta taxa de faltas

CHIP MULTITHREADING

IMT, BMT E SMT

O “esqueleto” do pipeline não é mudado

Recursos duplicados

- PC, Controle de mapeamento de registradores, etc.

Recursos compartilhados

- Banco de registradores (maior tamanho)
- Fila de instruções
- TLB e Caches (L1 e L2)
- Previsor de desvios

Exemplos de implementação

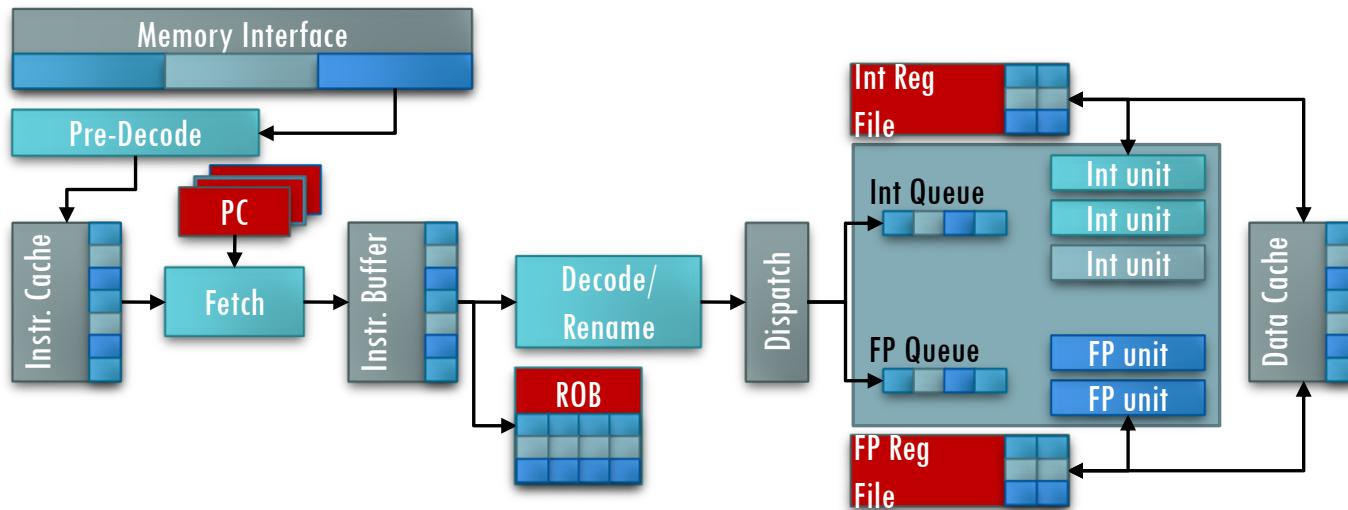
- Intel, a partir do Pentium 4 HT
- MIPS R10000 estendido
- Alpha estendido (21464)
- PowerPC 604

CHIP MULTITHREADING (IMT, BMT E SMT) COM EXECUÇÃO FORA DE ORDEM

Aumento de hardware em: PC, ROB, Reg File

Implicações: Mem. Principal, Cache Instr., Cache Dados compartilhados

Gargalos: Acesso à dados e instruções, Unidades funcionais



CHIP MULTI-CORE

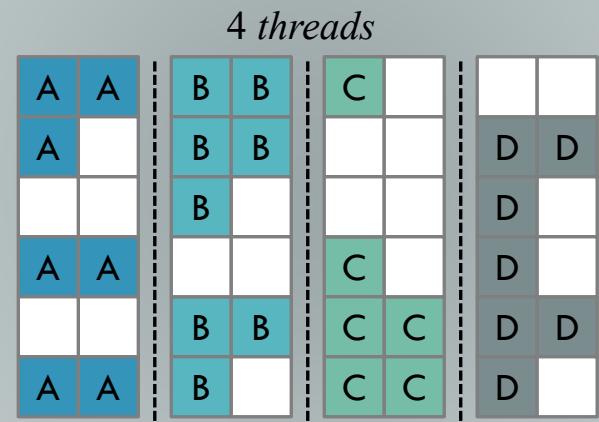
Também corresponde a um CMT.
Várias threads ativas e várias
em execução.

Ainda investe em ILP com
pipelining superscalar

Maior ênfase na mudança de
foco para TLP.

**Também pode combinar
técnicas IMT, BMT e SMT.**

CMP – Chip Multiprocessor
(pipeline superescalar em cada núcleo)



Latência de Memória	Não esconde
Troca de Contextos	Lenta
Utilização UFs	Baixa
Uso das Caches	Baixa taxa de misses

CHIP MULTI-CORE

Processadores comerciais CMP combinados com técnicas IMT, BMT e SMT.

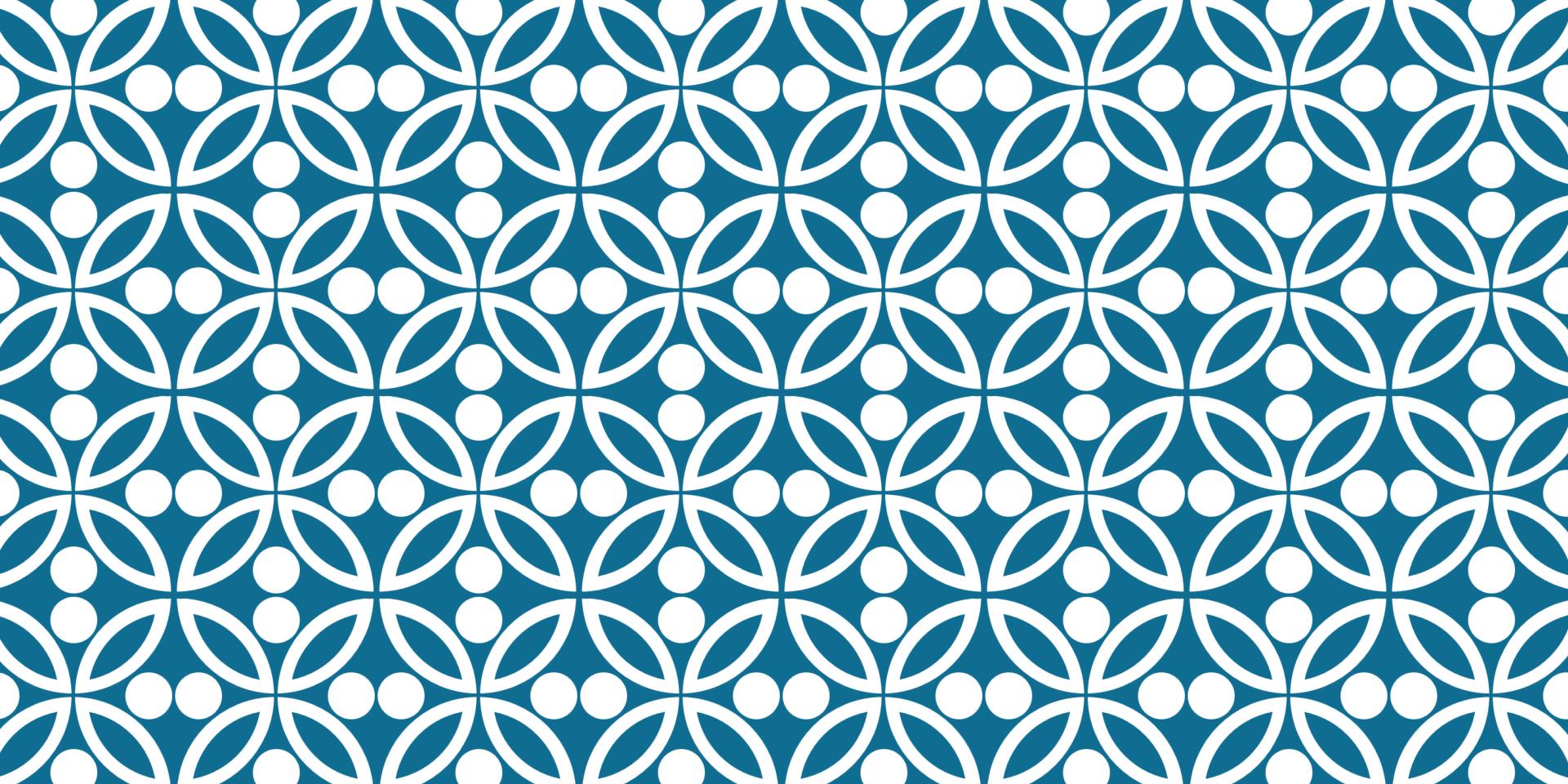
Nome	Cores	Técnica	Vias Thread/Core	Threads Ativas	Threads Paralelas
SUN Niagara I	8	IMT	4	32	8
SUN Niagara II	8	IMT	8	64	16
Sun Niagara III	16	IMT	8~16 ?	128~256 ?	32 ?
IBM Power 5	2	SMT	2	4	4
IBM Power 6	2	SMT	2	4	4
IBM Power 7	8	SMT	4	32	32
Intel Nehalem	8~12?	SMT	2	16~24?	16~24?
Intel Sandy Bridge	8?	SMT	2	16?	16?
Intel Tera-Scale	80 VLIW	-	-	80	80
AMD K10	4	-	-	4	4
AMD Bulldozer	8	-	-	8	8

COMO PODEMOS MELHORAR O DESEMPENHO DE VÁRIOS PROCESSADORES / NÚCLEOS?

Usando múltiplos programas ao mesmo tempo (ex. escutar música enquanto faço análise de dados, ou executar minha análise de dados duas vezes)

Utilizando programação paralela!!!

- Poderemos acelerar uma única tarefa, utilizando múltiplos recursos.



EXEMPLOS DE ARQUITETURA MULTI-CORE

INTEL CORE i7

Combina CMP + SMT + SIMD

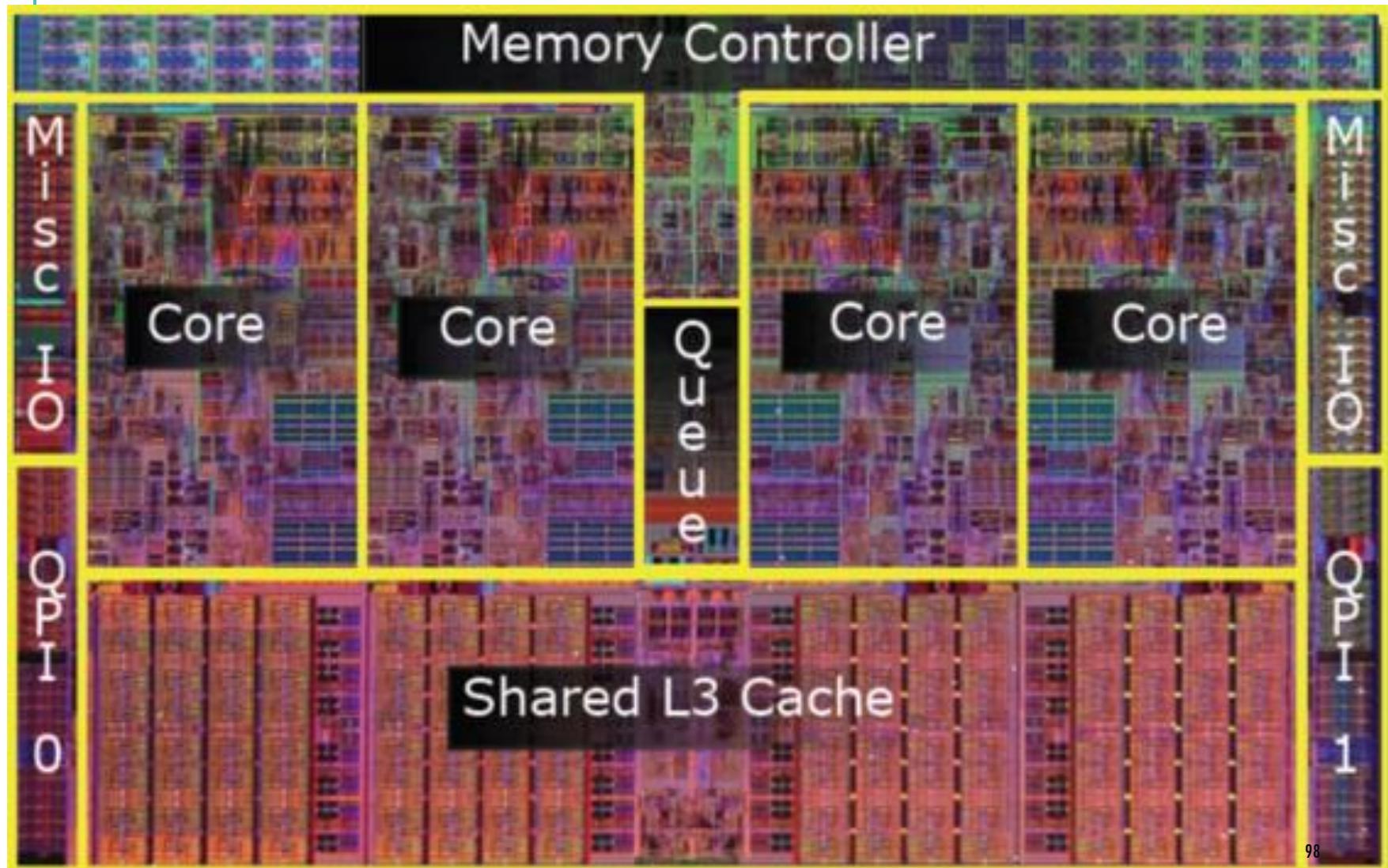
- Múltiplos cores, cada um com múltiplas threads e suporte a instruções vetoriais (MMX, SSE, AVX, AVX2)

Diversos processos de fabricação (45nm, 32nm, 22nm, 14nm, ...)

Exemplo

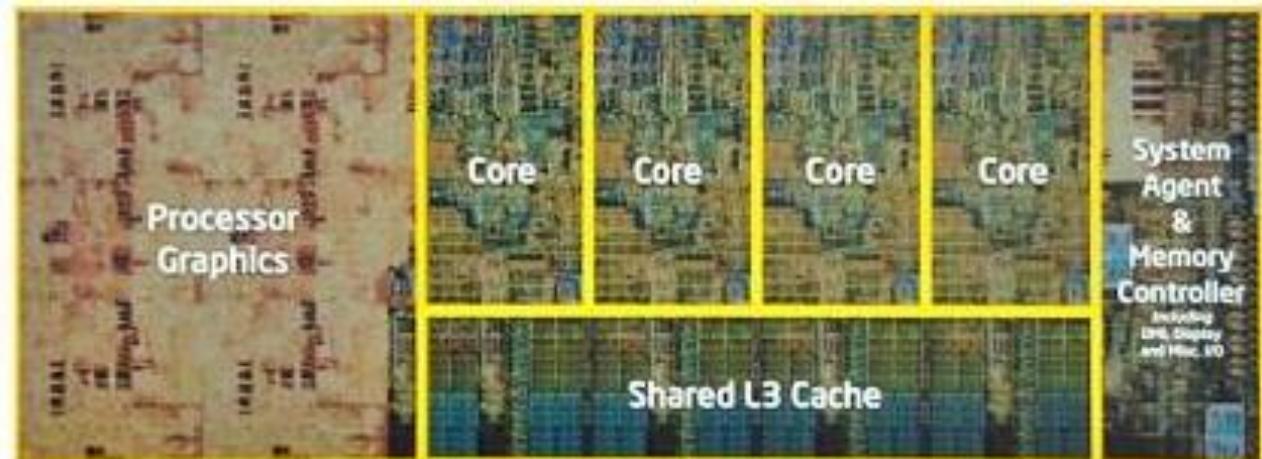
- Core i7-975 Extreme Edition
- 3.33GHz
- 4 Cores com Hyper Threading = 8 threads
- 4x256Kb L2 Cache
- 8Mb L3 Cache
- ~731 milhões de transistores

INTEL CORE i7 - NEHALEM

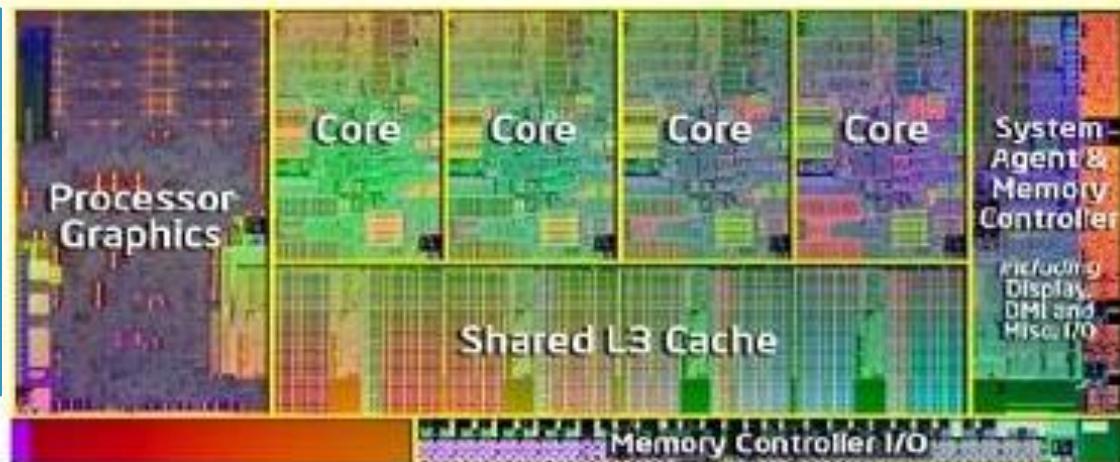


INTEL CORE I7 – SANDY-BRIDGE E IVY-BRIDGE

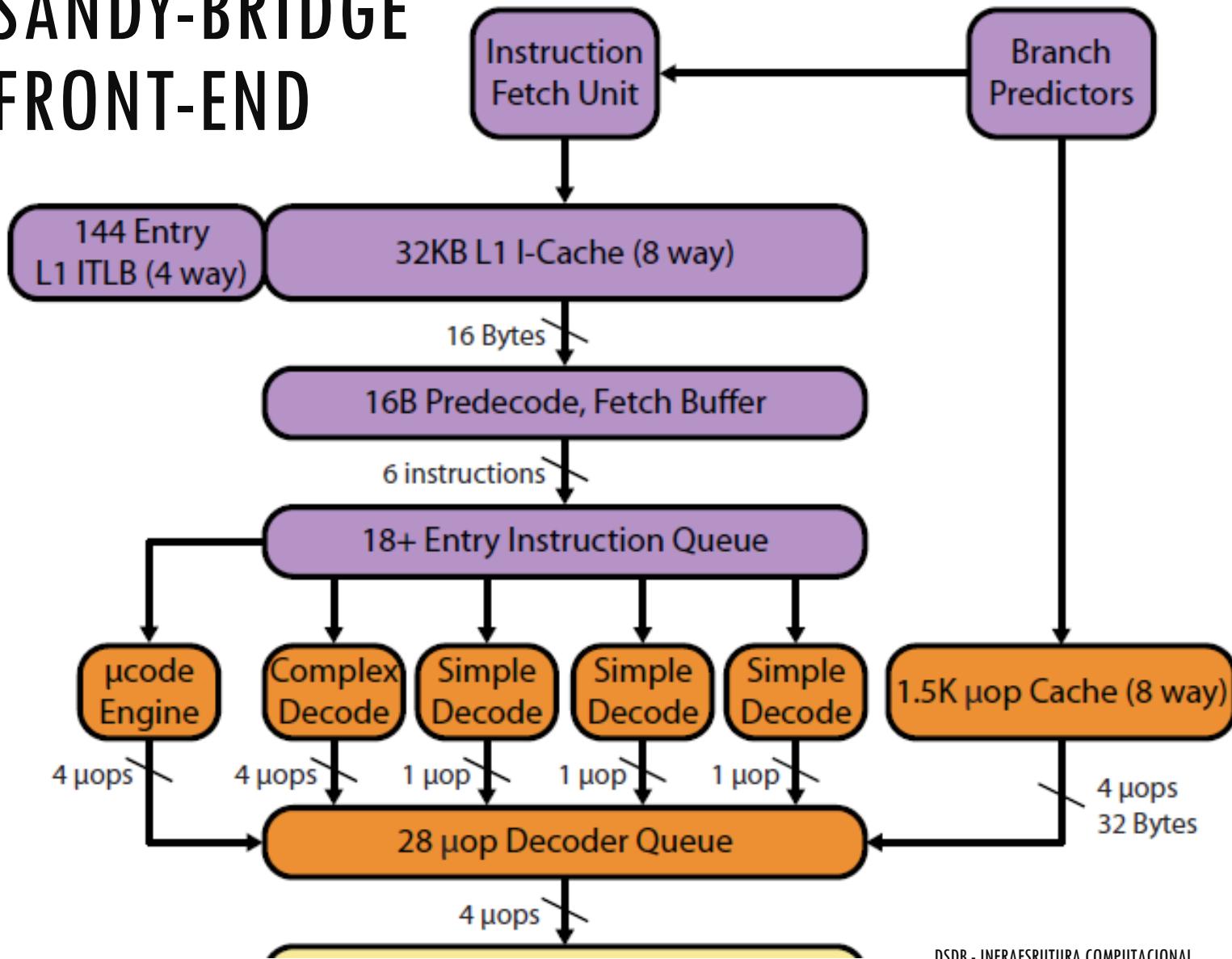
Ivy-Bridge



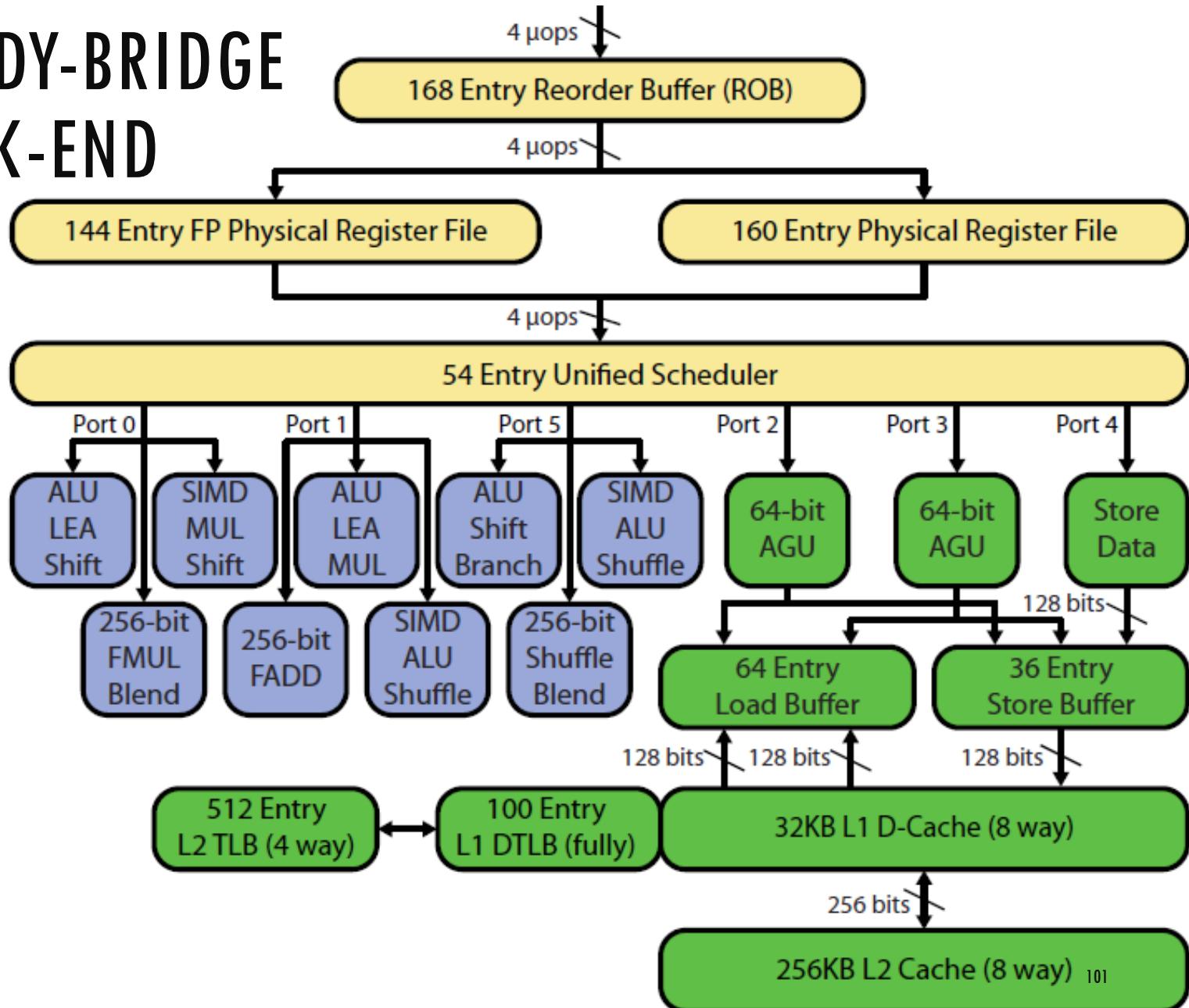
Sandy-Bridge



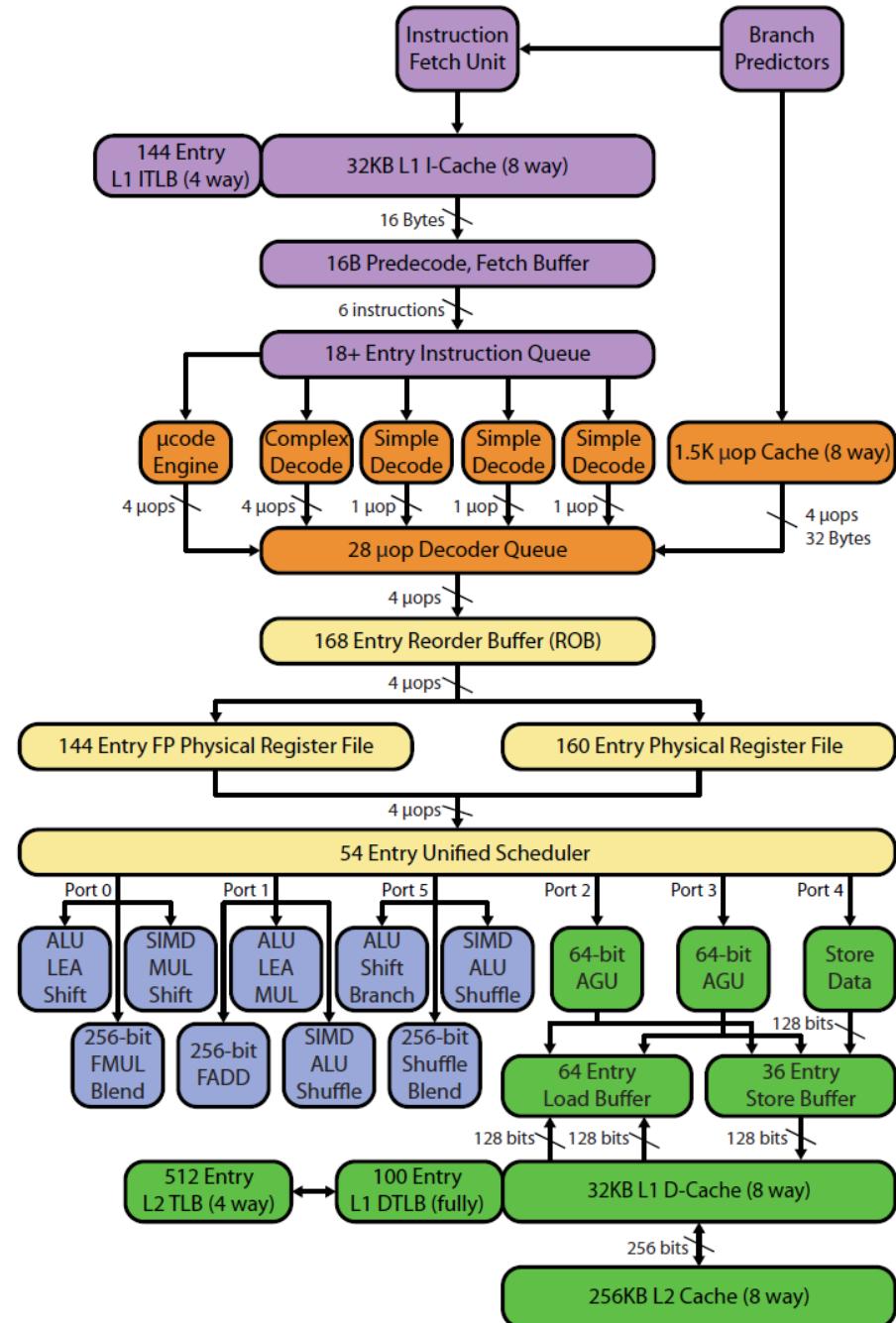
SANDY-BRIDGE FRONT-END



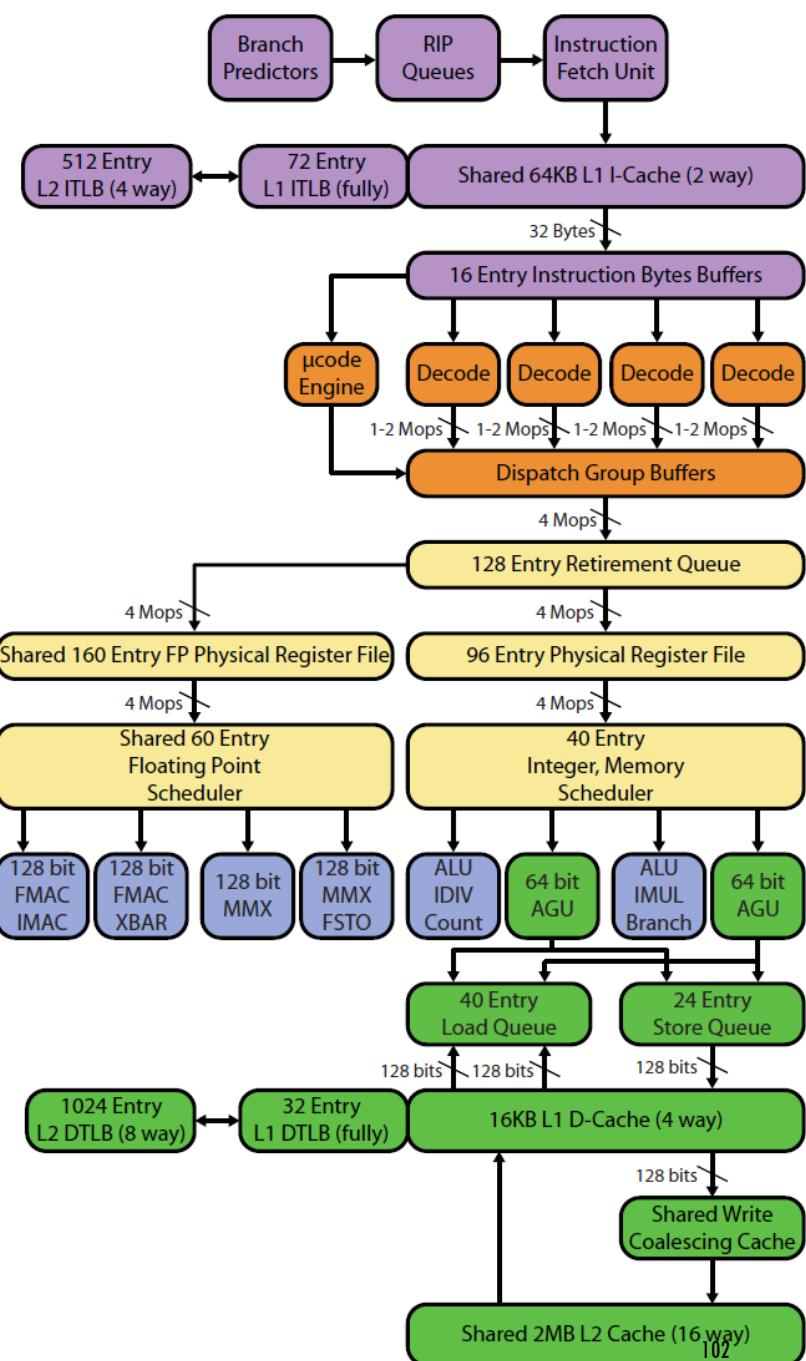
SANDY-BRIDGE BACK-END



Sandy Bridge



Bulldozer



DESAFIOS PARA MANY-CORE

Núcleos podem queimar.

Interconexão escalável: Network-on-Chip (NoC)

- Alg. Roteamento: Qual forma de trafegar dados ?

Demanda por dados RAM: Stacked 3D

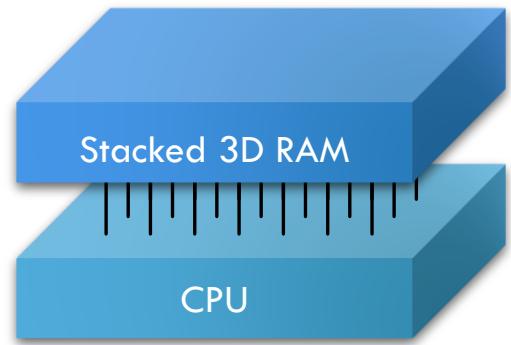
- Memórias cache: Non-Uniform Cache Architecture ?

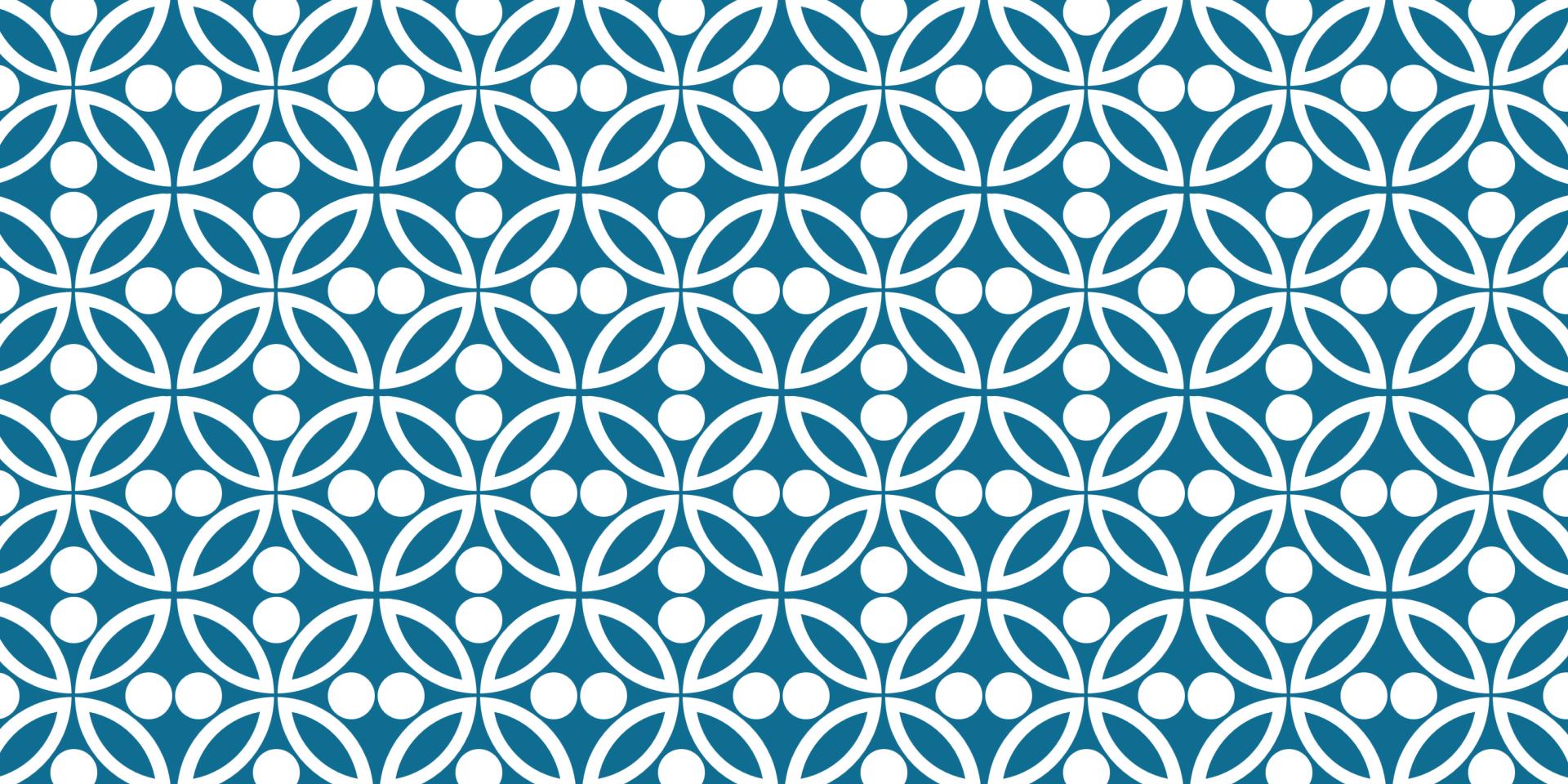
Como será a comunicação ?

- Memória Compartilhada ?
- Haverá troca de mensagens ?

Dark silicon

- Gap entre ganhos de área (tamanho dos transistores) e ganhos de potência (redução do consumo)
- Limite de quantos transistores realmente podemos ligar mantendo o orçamento de energia
- Estima-se que para tecnologias de 8nm, a quantidade de Dark Silicon atinja até 50%-80%*





INTEL XEON PHI

Intel® Many Integrated Core (Intel MIC) Architecture

Targeted at highly parallel HPC workloads

- Physics, Chemistry, Biology, Financial Services

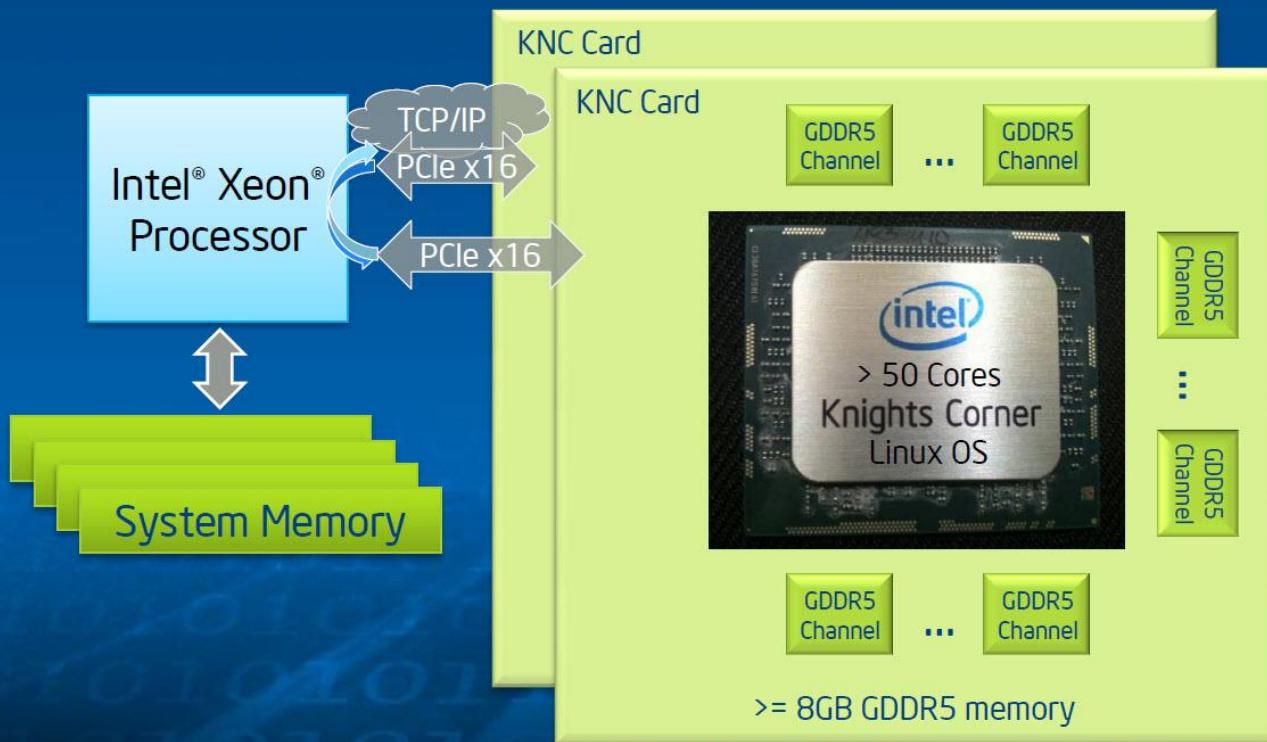
Power efficient cores, support for parallelism

- Cores: less speculation, threads, wider SIMD
- Scalability: high BW on die interconnect and memory

General Purpose Programming Environment

- Runs Linux (full service, open source OS)
- Runs applications written in Fortran, C, C++, ...
- Supports X86 memory model, IEEE 754
- x86 collateral (libraries, compilers, Intel® VTune™ debuggers, etc)

Knights Corner Coprocessor

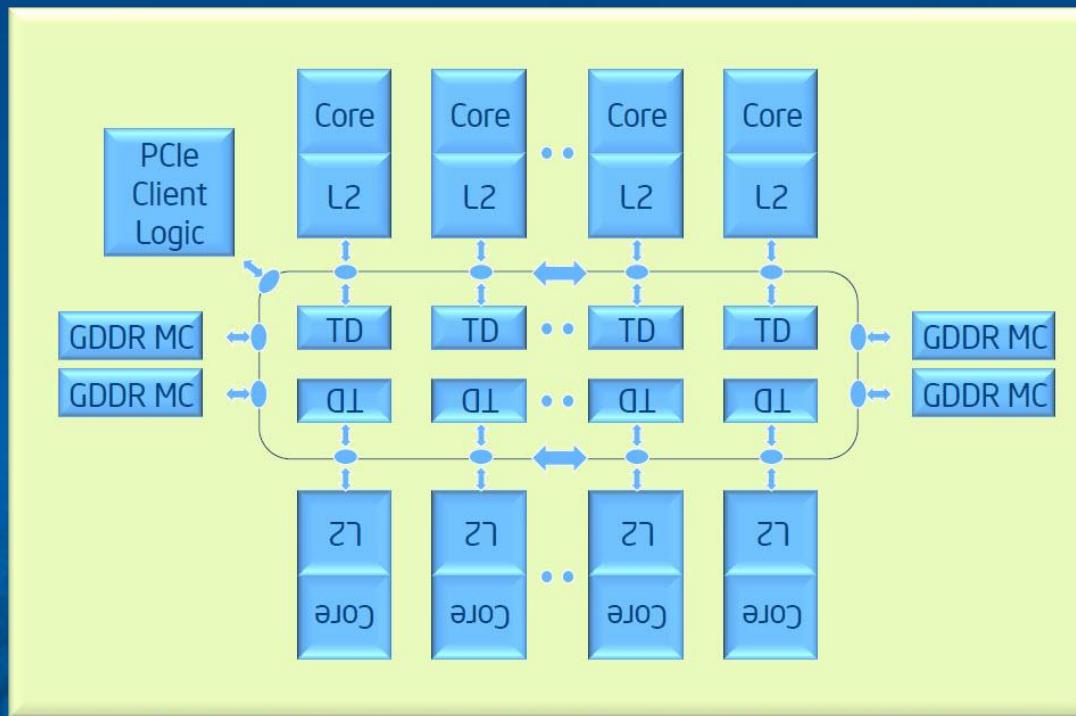


Knights Corner - Power Efficient

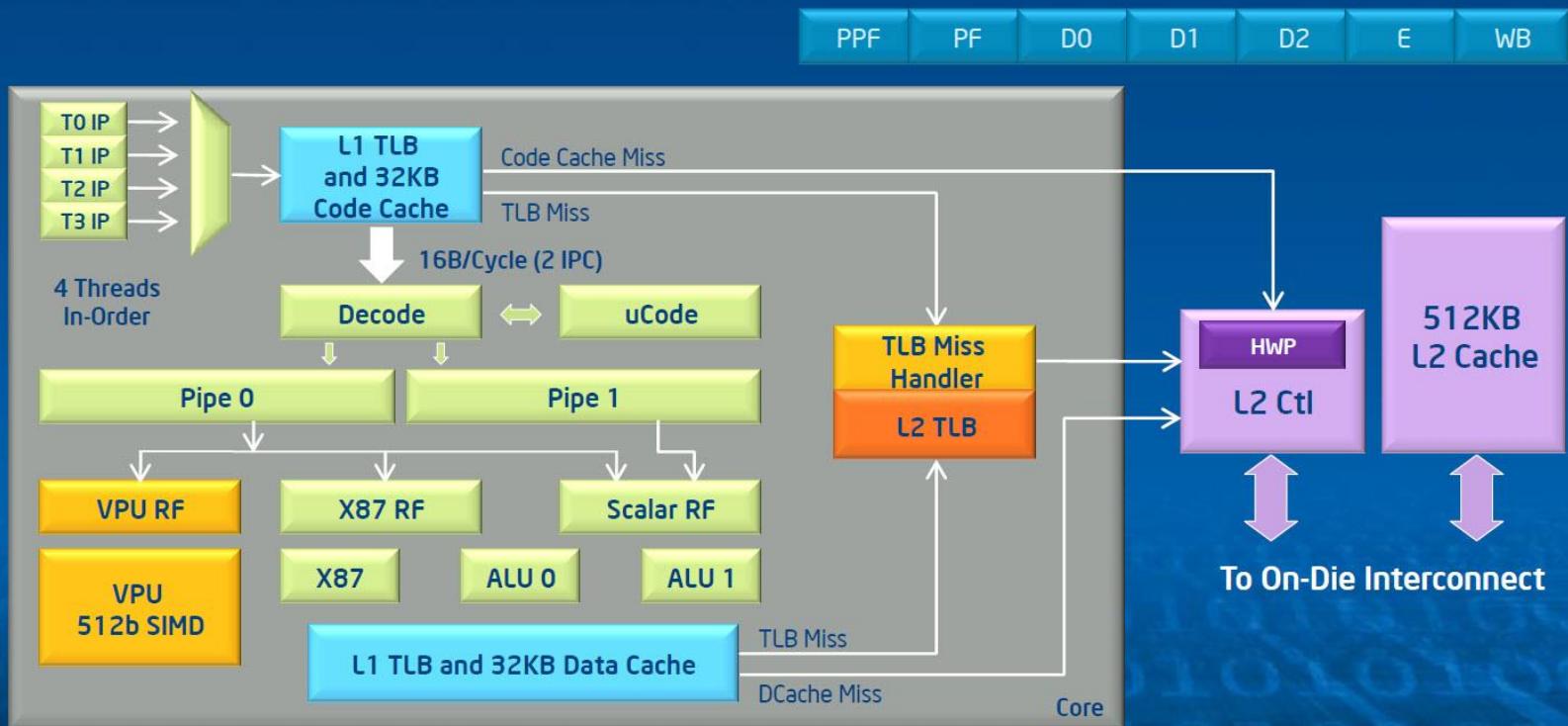
Performance per Watt of a prototype Knights Corner Cluster
compared to the 2 Top Graphics Accelerated Clusters



Knights Corner Micro-architecture

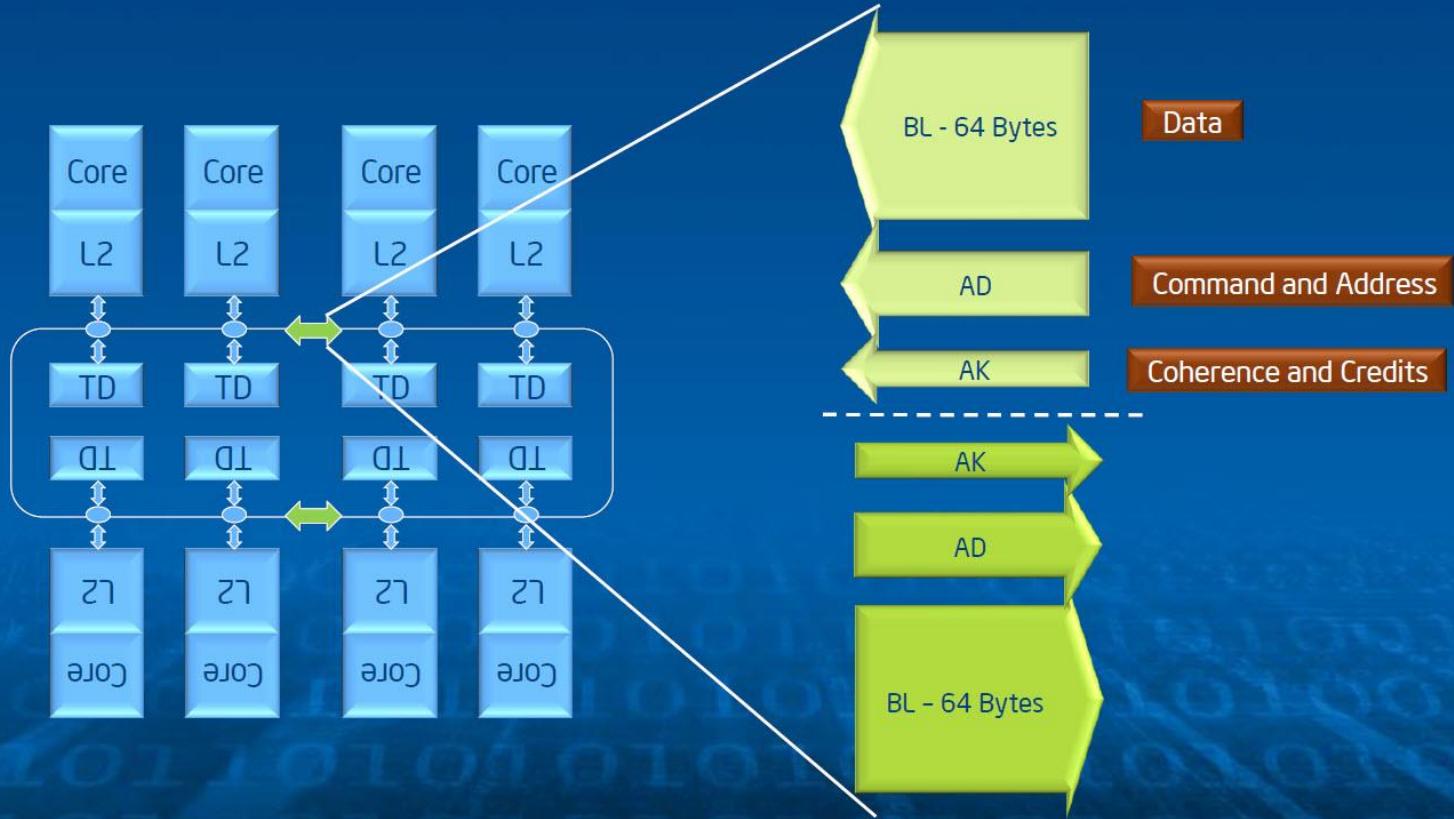


Knights Corner Core

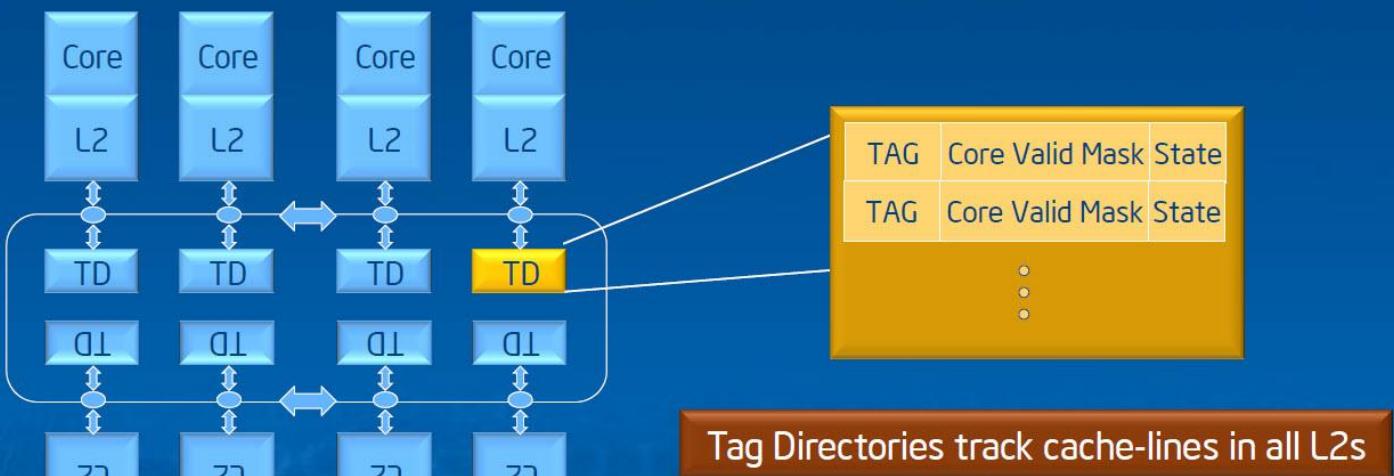


X86 specific logic < 2% of core + L2 area

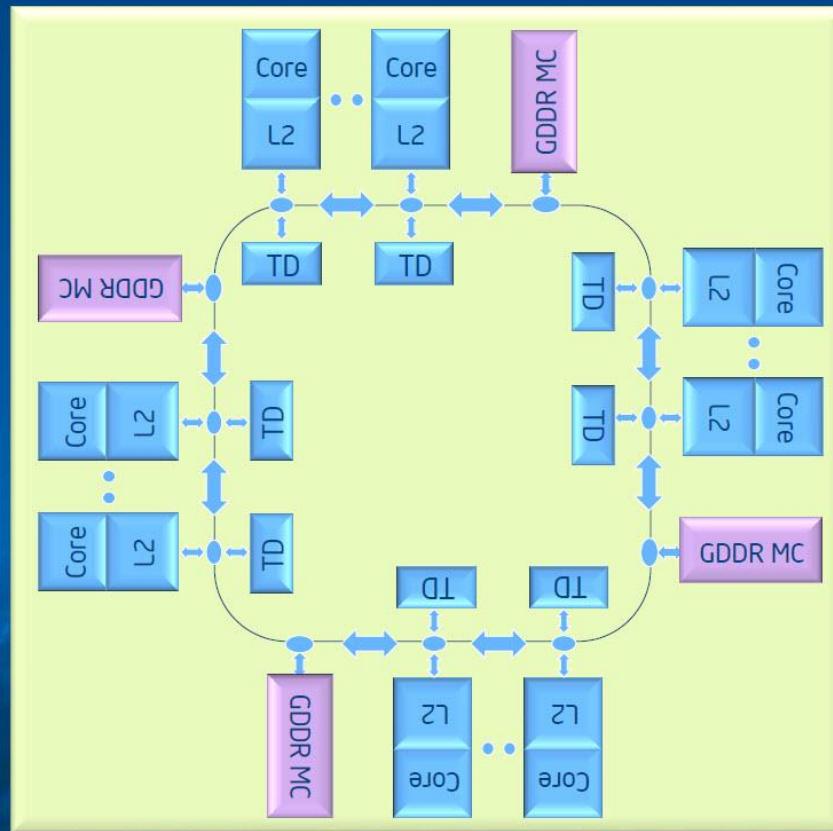
Interconnect



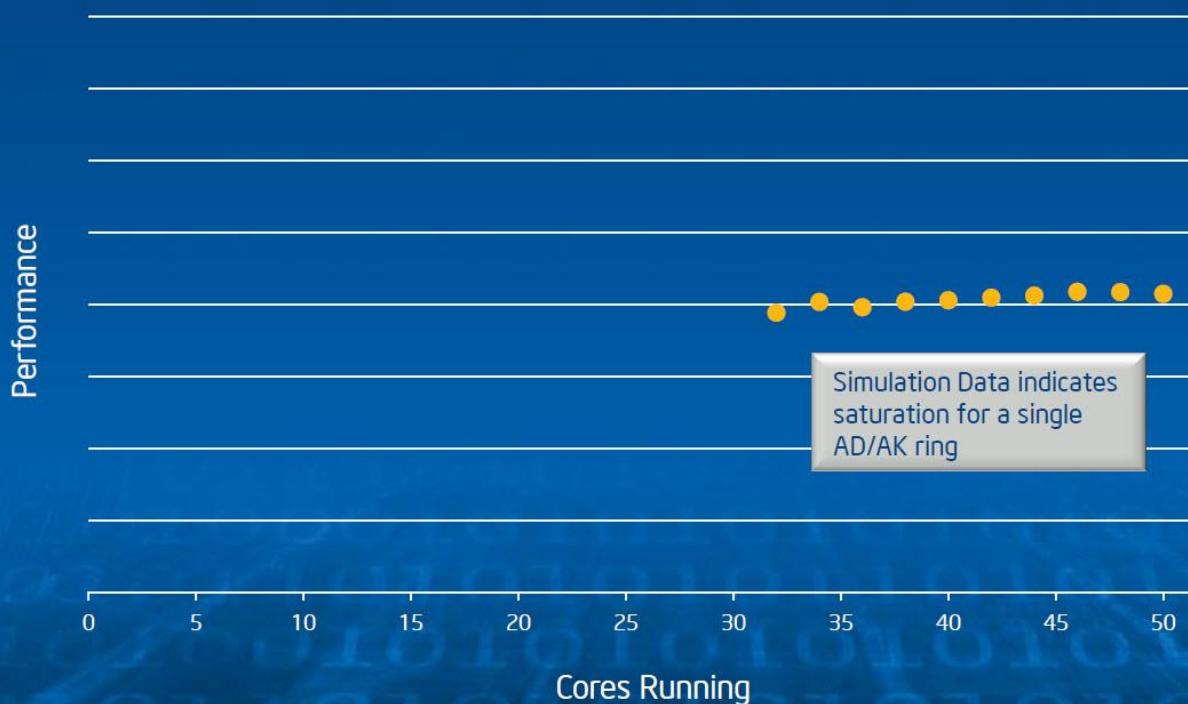
Distributed Tag Directories



Interleaved Memory Access

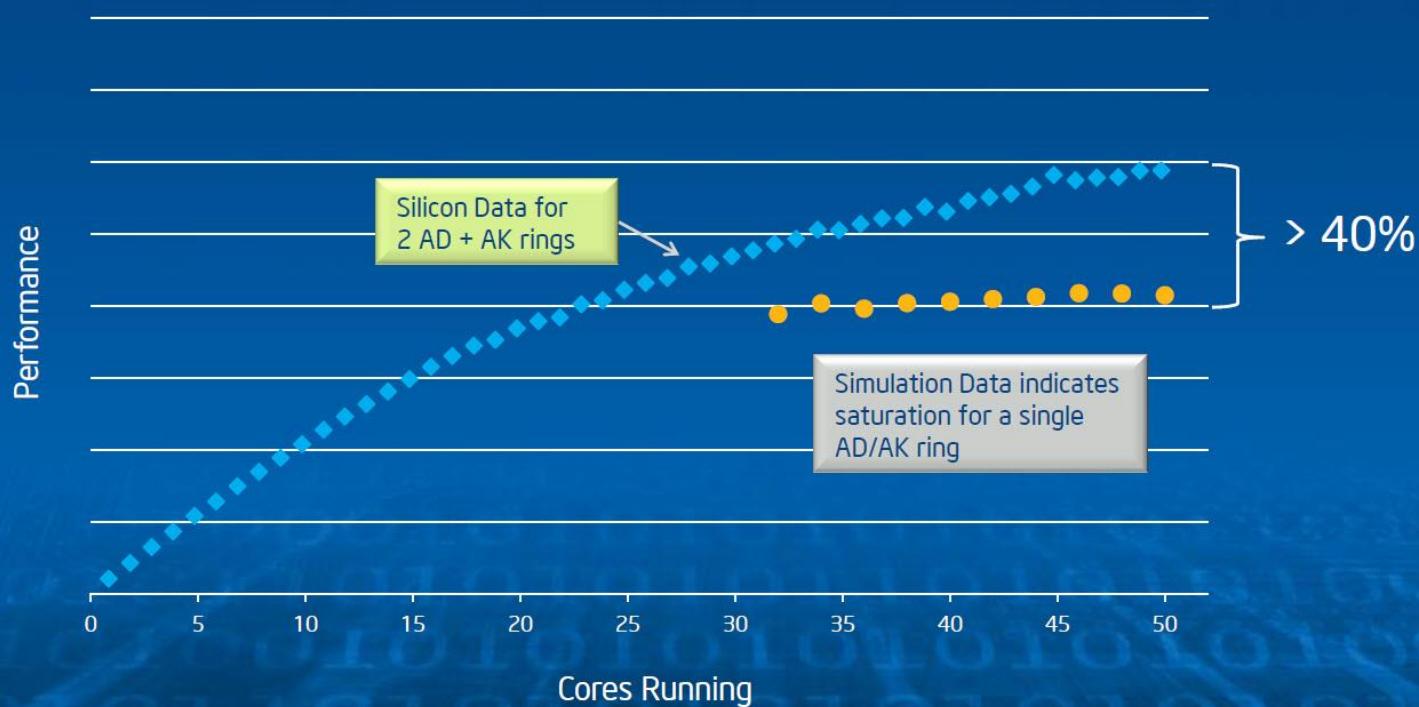


Multi-threaded Triad - Saturation for 1 AD/AK Ring



Results measured in development labs at Intel on Knights Corner prototype hardware and systems. For more information go to <http://www.intel.com/performance>

Multi-threaded Triad - Benefit of Doubling AD/AK



Results measured in development labs at Intel on Knights Corner prototype hardware and systems. For more information go to <http://www.intel.com/performance>

Streaming Stores

Streams Triad

```
for (i=0; i<HUGE; i++)  
    A[i] = k*B[i] + C[i];
```

Without Streaming Stores

Read A, B, C, Write A

256 Bytes transferred to/from memory per iteration

With Streaming Stores

Read B, C, Write A

192 Bytes transferred to/from memory per iteration

