



Estudios

ACT III

porte II

Sophia Carrazza



arquiteturas de
processadores

SUPERESCALARIDADE

- **escalar** → Não existe execução simultânea (instrução em fila, uma após a outra).
- **superescalar** → Várias unidades de execução (múltiplos UAs, por exemplo). Várias instruções completadas simultaneamente em cada ciclo de relógio.
 - ↳ limitações de paralelismo intrínseco dos problemas
 - ↳ ainda tem problemas c/ a execução simultânea de instruções e de dependências de dados / conflito de acesso a recursos comuns.
 - ↳ e controle (divisão)

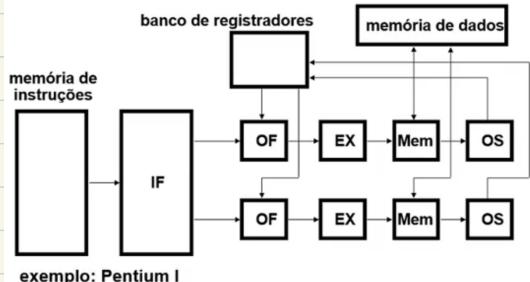
★ Pipelines ou unidades funcionais podem operar com velocidades variáveis - latências. (é um pipeline em que conseguimos executar mais de uma instrução ao mesmo tempo.)

- término das instruções pode não seguir a seqüência estabelecida no programa
- processador com capacidade de "look-ahead" → ele pode "despachar" instruções fora de ordem p/ execução
 - se há conflito que impede execução da instrução atual, processador
 - examina instruções além do ponto atual do programa
 - procura instruções que sejam independentes
 - executa estas instruções
 - possibilidade de execução fora de ordem
 - cuidado para manter a correção dos resultados do programa

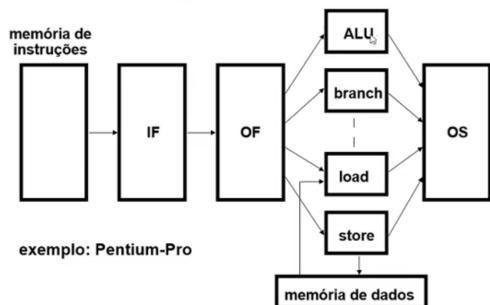
★ Cache de instruções precisa fornecer o domo (ou o mapeamento de instruções executadas ao mesmo tempo) de instruções por ciclo de processamento chamado "janela de instruções", c/ instruções futuras

★ capacidade look-ahead: analisa e examina instruções futuras na fila de execução, p/ identificar oportunidades de paralelismo e evitar possíveis conflitos em dependências de dados.

Processador com 2 pipelines



Unidades de execução especializadas



> Despacho de instruções

→ fornecimento de instruções p/ as unidades funcionais (ex: ALU, branch, load, store, etc)

com despacho em ordem, terminação em ordem.

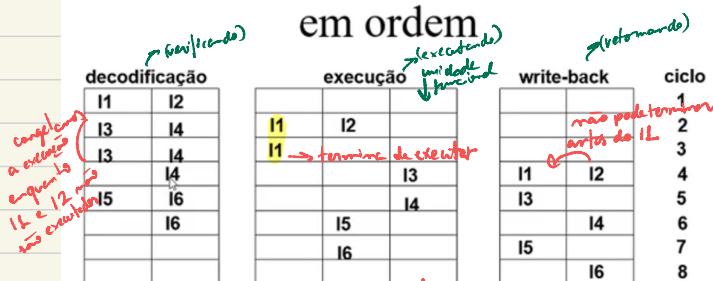
→ só é feito quando instruções anteriormente despachadas já foram executadas

→ o despacho é congelado quando existe conflito por un. funcional ou quando a un. funcional exige + de 1 ciclo p/ gerar resultado.

> Término de instruções

→ escrita de resultados das instruções

Despacho em ordem, terminação em ordem



restrições:

- fase de execução de I1 exige 2 ciclos
- I3 e I4 precisam da mesma unidade funcional
- I5 e I6 precisam da mesma unidade funcional
- I5 depende do valor produzido por I4

6 instruções em
6 ciclos
IPC = 1.0

* Despacho em Ordem, Terminação fora de ordem

- o despacho não espera que instruções anteriores já tenham sido executadas (ele não é congelado quando um funcional levava + de 1 ciclo p/ executar a instrução).
- uma un. funcional pode completar uma instrução após instruções subsequentes já terem sido completadas.
- o despacho ainda precisa ser congelado quando há conflito por um funcional ou há uma dependência de dados verdadeira.

Despacho em ordem, terminação fora de ordem

decodificação	I1	I2
	I3	I4
	I4	
	I5	I6
	I6	

execução	I1	I2	I3	I4
	I1			
	I1		I3	
			I4	
			I5	
			I6	

write-back	I1	I2	I1	I3

ciclo	1
2	
3	
4	
5	
6	
7	

6 instruções em
5 ciclos
IPC = 1.2

notar:

- I1 termina fora de ordem em relação a I2
- I3 é executada concorrentemente com último ciclo de execução de I1
- tempo total reduzido para 7 ciclos

(enviando R3) dependência de saída → I¹ e I³ concorrente em R3

→ I¹: R3 := R3 op R5

I²: R4 := R3 + 1

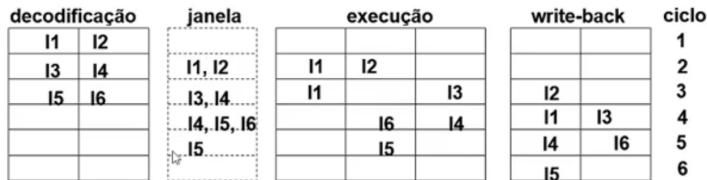
I³: R3 := R5 + 1

valor final de R3 deve ser o escrito pela 3^a instrução
outra, atribuição da 1^a não pode ser feita depois da 3^a.
* despacho da 3^a precisa ser congelado.

* Despacho Fora de Ordem, Terminação fora de ordem

- inicia o estágio de decodificação do estágio de execução
- continua buscando e decodificando instruções, mesmo que elas não possam ser executadas imediatamente
- janela de instruções → buffer entre os estágios de decodificação e execução. (é tipo um banco de questões previamente decodificados).
- instruções são buscadas da janela independentemente da sua ordem de chegada (despacho fora de ordem).
- o estágio de "congelado" de despacho não existe mais!

Despacho fora de ordem, terminação fora de ordem



6 instruções em
4 ciclos
IPC = 1.5

notar:

- estágio de decodificação opera a velocidade máxima, pois independe do estágio de execução
- I6 é independente e pode ser executada fora de ordem, concorrentemente com I4
- tempo total reduzido para 6 ciclos

* antidependência → 2^a instrução escreve em R3 e 1^a precisa ler o valor de R3

1^a: $R4 := R3 + 1$

antes que a 2^a escreva o mesmo valor

2^a: $R3 := R5 + 1$

* despacho da 2^a instrução precisa ser congelado até que a 1^a tenha lido valor de R3.

* Renomeação de Registradores:

- não é possível criar um número infinito de registradores
- pegamos nomes temporários de um banco de registradores interno p/ renomeá-los (temporariamente!)

usada somente p/
dependência falsa!!! → WAR e WAW

Renomeação de registradores

- antidependências e dependências de saída são causadas pela reutilização de registradores
- efeito destas dependências pode ser reduzido pelo aumento do número de registradores ou pela utilização de outros registradores disponíveis
- exemplo
 - ADD R1, R2, R3 ; R1 = R2 + R3
 - ADD R2, R1, 1 ; R2 = R1 + 1 antidependência em R2
 - ADD R1, R4, R5 ; R1 = R4 + R5 dependência de saída em R1
- utilizando 2 outros registradores R6 e R7 pode-se eliminar as dependências falsas
 - ADD R1, R2, R3 ; R1 = R2 + R3
 - ADD R6, R1, 1 ; R6 = R1 + 1
 - ADD R7, R4, R5 ; R7 = R4 + R5

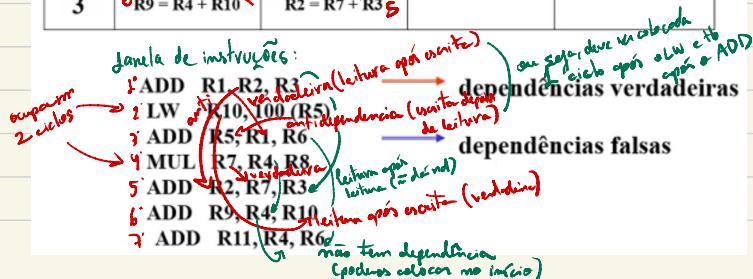
* Buffer de Rendimento

- algumas instruções não podem terminar fora de ordem, então este buffer serve p/ isso (é uma fila FIFO, em que movemos os valores no buffer antes de registrá-los no banco de registradores).

Exemplo

ciclo	somador 1	somador 2	multiplicador	load/store
1	1' R1 = R2 + R3 <i>verdadeira</i>	R11 = R4 + R6 <i>falsa</i>	R7 = R4 * R8	R10 = mem (R5+100)
2	3' R5 = R1 + R6 <i>falsa</i>		R7 = R4 * R8 <i>verdadeira</i>	R10 = mem (R5+100) <i>verdadeira</i>
3	6' R9 = R4 + R10 <i>verdadeira</i>	R2 = R7 + R3 <i>verdadeira</i>		

janela de instruções:



Antidependência = escrita após leitura

Agora, conforme o processador:

Exemplo

ciclo	somador 1	somador 2	multiplicador	load/store
1	$R1 = R2 + R3$	$R11 = R4 + R6$	$R7 = R4 * R8$	$R10 = \text{mem}(R5+100)$
2	$R1 = R1 + R6$ A		$R7 = R4 * R8$	$R10 = \text{mem}(R5+100)$
3	$R9 = R4 + R10$	$R1 = R7 + R3$ B		

O processador olha pra frente de instruções e, após as analisar, ele vai dispatchar 4 instruções no 1º ciclo, +1 no 2º ciclo e +2 no 3º ciclo.

ADD R1, R2, R3
LW R10, 100(R5)
ADD RA, R1, R6
MUL R7, R4, R8
ADD RB, R7, R3
ADD R9, R4, R10
ADD R11, R4, R6

dependências verdadeiras

dependências falsas

- * se produz p/ fazer reconstrução de registros:
- verificamos qual não as dependências falsas (os antidependências, não cole)
 - verificamos a instrução que sofre dependência (e vem depois da 1ª) p/ liberar os muros des minerais

MULTITHREADING

Objetivo: aumentar a utilização de recursos do processador executando múltiplos threads simultaneamente. → É uma parallelização + MACKO, em que executamos fluxos de instruções paralelamente.

SMT (Simultâneo): Compartilhamento de recursos físicos entre threads

IMT (Entrelaçado): Alturação rápida entre threads p/ escorrer latências de memória → Interleaved Multithreading

Supõe a múltiplas threads ≠ Execução Simultânea de Threads

→ é decorrente de uma das técnicas de suporte a multithread.

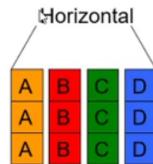
* O processador tem que ter a capacidade de gerenciar o endereço de cada uma das threads

Threads: instrução ou bloco



Entrelacamento, ciclo a ciclo, de instruções de threads diferentes! (não tem simultaneidade, mas entrelacamento de threads!).

Vertical



Horizontal

Interleaved Multithreading

Alternância rápida entre threads, dando cada uma um pedaço de tempo da CPU.

(instruções por instruções) → threads de grão fino.

IMT / BMT

Blocked Multithreading

É o entrelacamento entre threads, porém de grão grosso (bloco de instruções). Então, ao invés de entrelacar instruções, por exemplo, entrelaçamos algumas de B, depois algumas de D, etc.

SMT

Simultaneous Multithreading

Hyper-threading - Execuções das threads simultaneamente. Precisa ser da arquitetura superscalar, pq as instruções têm que ser executadas paralelamente.

* interleaving pode ser feito tanto em arquiteturas escalares quanto superscalares!

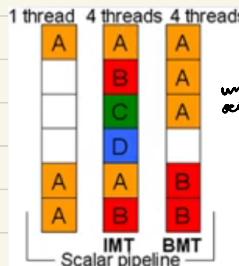
(no caso da inversa é escalar, pq não tem manutenção instruções rodando ao mesmo tempo que outra)

* (pode ser dependência verdadeira, podem ser unidades muito especiais e não tem instruções em outras unidades, etc.)

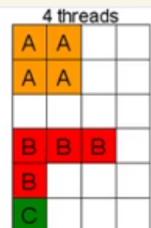
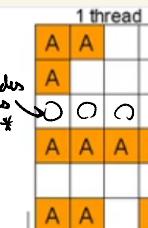
As técnicas:

IMT } referem-se ao núcleo.

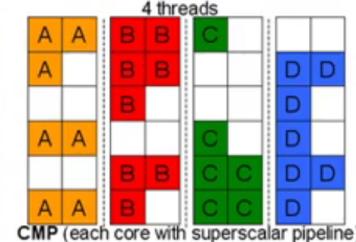
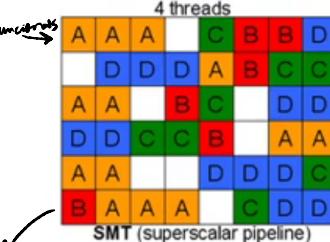
BMT } referem-se ao todo, considerando os múltiplos núcleos dentro dele



unidades operadoras *



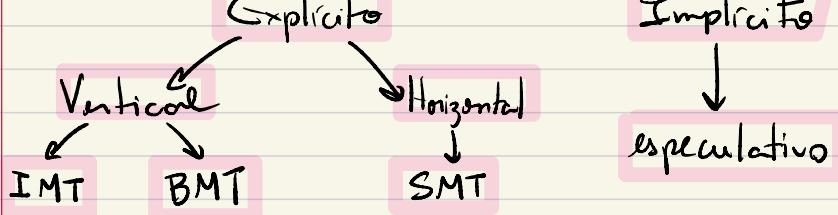
8 unidades funcionais



Conseguimos distribuir instruções de threads diferentes p/ múltiplas unidades funcionais ao mesmo tempo → redução de ociosidade (tendo a fazer um melhor uso das threads)

Chip Multiprocessor = Chip/Processor Multicore
(não tem relação c/ o suporte multithreading)
- Nesse, se encontra múltiplos processadores (no caso dessa imagem 4 núcleos superscalares)

Multithreading



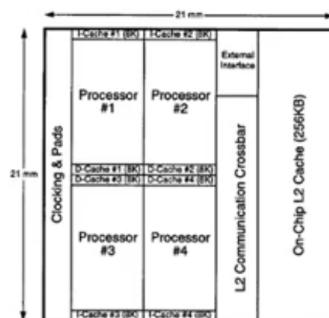
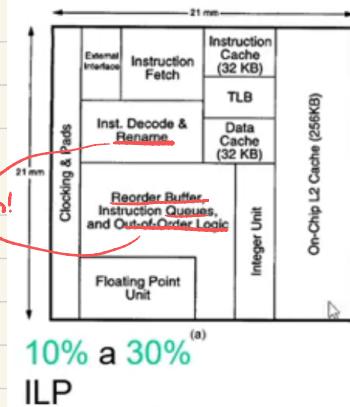
SMT: Superescalar

- Ilusão de mais de 1 núcleo de processamento
- Não existe avançamento de pipeline (BMT)
- Não existe atraso na execução de threads (IMT / BMT)
- Intel usa como "Hyperthreading"

- Banco de registradores muito grande p/ guardar vários contextos
- Conflitos de cache são degradação de desempenho

(a) Superescalaridade de seis vias de execução.

(b) Chip multi-core. Cada core superescalar com duas vias de execução.



Olukotun, 1996

→ Instruction Level Parallelism

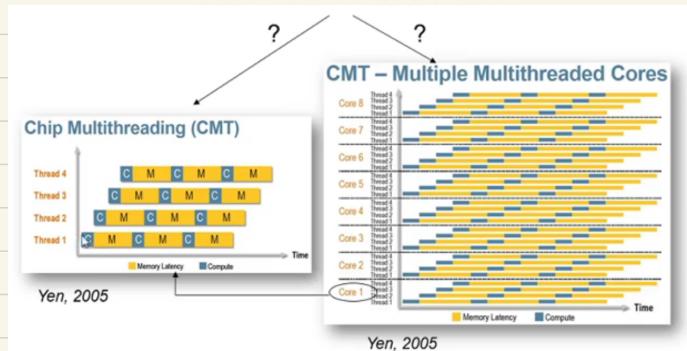
Parallelism (carga de trabalho
e alto paralelismo de instruções)

→ Thread Level Parallelism

(Ultra-Sparc TL)

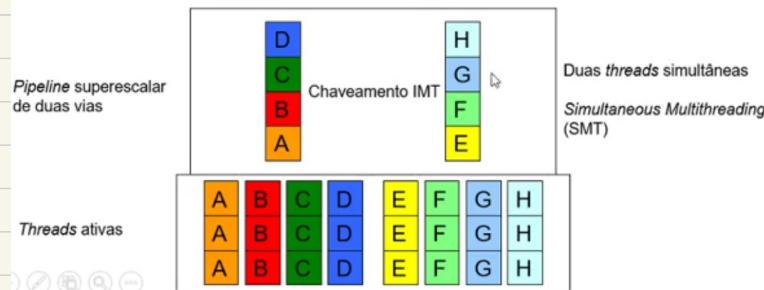
Processador NIAGARA → Produzido pela SAM

→ possuía 8 núcleos escalares que suportavam 4 threads (com IMT em cada núcleo), ou seja, 32 threads ativas dentro do processador multicore e 8 threads simultâneas, pq são 8 núcleos.



NIAGARA II → Suporta a 8 threads por núcleo (cada núcleo tem 2 pipelines com 4 threads), ou seja, 64 threads ativas e 16 threads simultâneas.

- SMT combinado com IMT em chip multi-core.
 - Superescalaridade associada a SMT com entrelaçamento de instruções.
 - Aumenta desempenho para cargas de trabalho de propósitos gerais.
 - Reduz tamanho e complexidade da arquitetura superescalar do processador.



MULTI-CORE

- Processadores em que seus núcleos são multi-core.
- Lei de Moore → relacionada à capacidade integrativa, à quantidade de transistores em um mesmo espaço.
 - Se diminuirmos o tamanho dos transistores, aumentamos a qntd de processadores dentro do chip de processador, e os chamamos de núcleos.
- Processadores Power 4 e Power 5 da IBM → possuem 2 núcleos (Acaso a cache L3 é diferente e o Power 5 suporta 2 threads SMT por núcleo)
 - em alguns casos o Power 5 não tem desempenho melhor, por causa do m^o limitado de unidades de execução e o alto consumo da largura de banda de memória pelos 2 threads.
- Intercoração dos núcleos é feita por barramento ou chave crossbar → O problema está nos fios usados p/ a conexão, pois eles têm perda de dados por causa da atenuação do sinal e núcleos distantes não escutam sinal (colisão e perda de dados).
Xede é unida p/ interconectar todo mundo
(é global)

MANYCORE (Redes em Chip)

- A ideia é eliminar a influência do fio!
 - rede em chip sem fio, óptica, reconfiguráveis ou com fio mais curto.
- Temos também um aumento no número de núcleos.
- Intercoração mais robusta (Rede em Chip) → essa é a característica principal do manycore

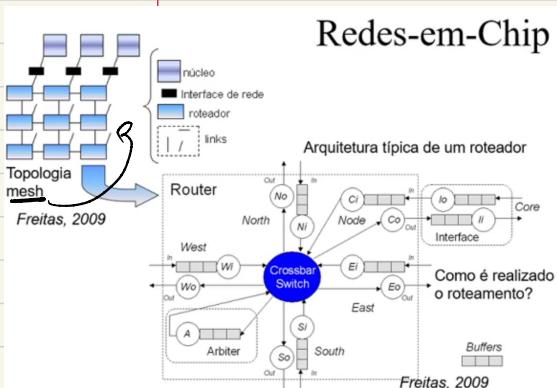
Redes-em-Chip:

- composta por roteadores
- possui pacotes de rede
- trabalha c/ protocolo de rotamento
- possui diversas topologias
- trabalha c/ Qualidade de Service (QoS)
- é tolerante a falhas
- é escalável

> Barramento e Chave Crossbar → interface não escalável
→ vários barramentos entrelacados

> Redes em Chip → interface de rede empacota o dado, entrega para o roteador e ele vai interpretar o que fazer com o pacote pelo cabeçalho. Assim, o roteamento é feito e enviado o pacote p/ o núcleo destino → Topologia Mesh

↳ Protocolo XY → estratégias de transporte de dados em uma NoC
- protocolo de roteamento em que deslocar-se pelos links em X e depois em Y p/ entregar o pacote ao núcleo destino.



Preocupações no projeto de NoC

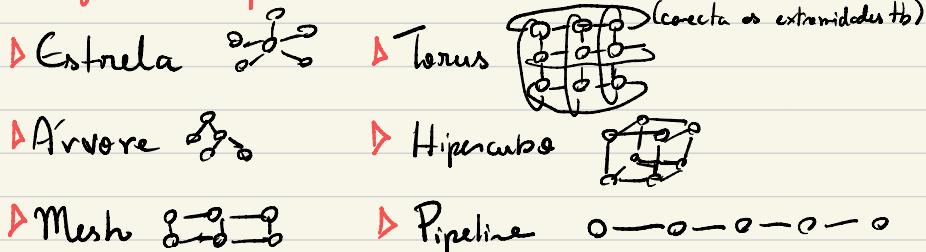
- **Deadlock:** é a representação de uma dependência cíclica. Neste caso, um pacote não consegue progredir e fica restrito a um subconjunto de estados ou roteadores.
- **Livelock:** é a representação de uma contínua retransmissão do pacote sem atingir o nó destino. Comum em protocolos de roteamento.
- **Starvation:** é a representação da não alocação de um recurso devido a postergação indefinida de acesso ao mesmo. Comum em protocolos de arbitragem.

Tipos de Buffers

- **Buffers de entrada:** As técnicas de arbitragem são relativamente simples, possui uma melhor relação de área e potência, além de proporcionar um melhor desempenho para a chave crossbar.
- **Buffers de saída:** Em função de N entradas conectadas a cada um dos buffers de saída, a chave crossbar precisa ser **N vezes mais rápida**. A adoção de buffers de saída não é a mais adequada para alto desempenho. No entanto, existem vantagens em se tratando da **eliminação do bloqueio de pacotes** que não receberam permissão de envio porque o primeiro pacote da fila ainda não teve liberação de uma determinada saída. Este problema é conhecido como **head of the line blocking** e pode acontecer nas soluções com buffers de entrada.
- **Buffers de crosspoint:** Cada ponto de conexão da chave crossbar possui um buffer. É utilizada a técnica de roteamento chamada de self-routing. Neste caso, em cada crosspoint seria necessário além do buffer um decodificador para decisão de envio ou não do pacote. Esta solução aumenta o **tamanho** e a **potência** consumida da chave crossbar.

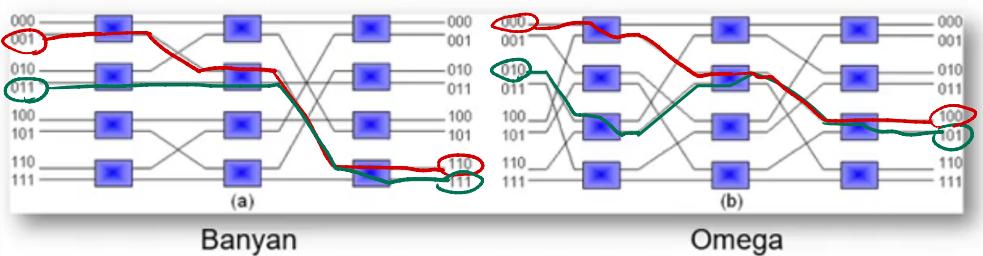
!!!

Topologias → formatos de interconexões de Redes-chip



Redes Dinâmicas (multi-nível)

único caminho entre entrada e saída faz com que o roteamento seja eficiente, podendo ser feito de forma descentralizada



- Rede direta → todos os roteadores fornecem um caminho por determinado nó
- Rede indireta → precisa passar por um roteador p/ chegar em outro

círculo lógico responsável por controlar acesso compartilhado ao barramento de dados.

Prós (+) e Contras (-)

P/fo Tratamento de dados

Tipo de Interconexão		
Barramento	Fio	O aumento do fio <u>aumenta a resistência degradando o desempenho</u> .
Chave Crossbar		O aumento do fio aumenta a resistência degradando o desempenho.
Network-on-Chip		Os fios são <u>ponto-a-ponto entre roteadores</u> e o desempenho <u>não degrada</u> em função do aumento de nós. <u>fio entre roteadores</u>
Barramento	Árbitro	O árbitro é um <u>gargalo</u> à medida que o número de nós aumenta.
Chave Crossbar		O árbitro pode ser <u>centralizado ou descentralizado</u> e não é o fator principal para degradação do desempenho em função do aumento dos nós.
Network-on-Chip		As decisões de roteamento são <u>distribuída</u> e não representam um <u>gargalo</u> .
Barramento	Largura de banda	A largura de banda é <u>limitada e compartilhada</u> por todos os nós.
Chave Crossbar		Cada interconexão é independente e a largura de banda de comunicação por conexão não é afetada pelas demais.
Network-on-Chip		A largura de banda não é afetada pelo aumento da rede.
Barramento	Latência	Latência é afetada pelo fio.
Chave Crossbar		Latência é afetada pelo fio.
Network-on-Chip		Latência é afetada pelas <u>contentções</u> em roteadores
Barramento	Compatibilidade	Em sua maioria são <u>compatíveis com qualquer IP (Intellectual Property)</u> incluindo os softwares.
Chave Crossbar		Em sua maioria são compatíveis com qualquer IP (Intellectual Property) incluindo os softwares.
Network-on-Chip		São necessários <u>adaptadores (wrappers)</u> entre os IPs e os softwares precisam de sincronização em sistemas multi-core.
Barramento	Complexidade	Conceitos simples e bem compreendidos.
Chave Crossbar		Conceitos simples e bem compreendidos.
Network-on-Chip		Projetistas precisam de uma reeducação em função dos <u>novos conceitos</u> .

Conceptos de ARQUITETURA PARALELA

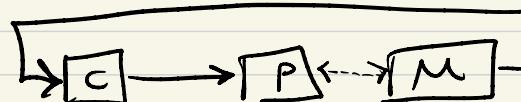
- Arquiteturas monoprocessadoras → processos compartilham o mesmo processador. Só pode ser implementado como monoprogramação ou multiprogramação → recursos do computador → processador → alternar a execução de vários processos
- 1 processo = parte de um programa em execução

► Arquiteturas multiprocessados → vários elementos de processamento.
Temos memória compartilhada e memória distribuída

	SD (Single Data)	MD (Multiple Data)
SI (Single Instruction)	SISD Máquinas von Neumann	↳ SIMD Máquinas Array
MI (Multiple Instruction)	MISD Sem representante até agora	MIMD Multiprocessadores e multicamputadores

De Rose (2003)

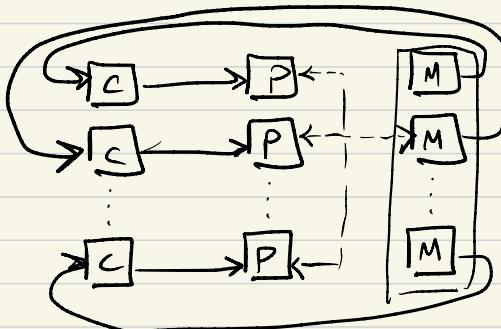
SISD →



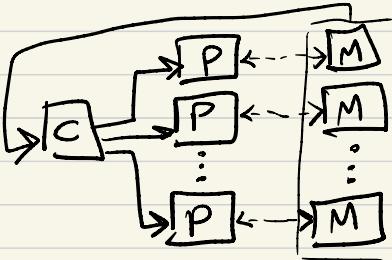
— instruções

↔ dados

MISD →

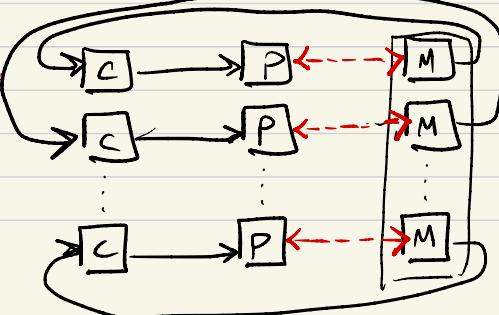


SIMD →



MIMD →

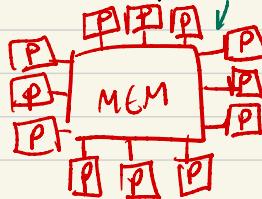
& mais usado!



MIMD

Computadores
paralelos / distribuídos

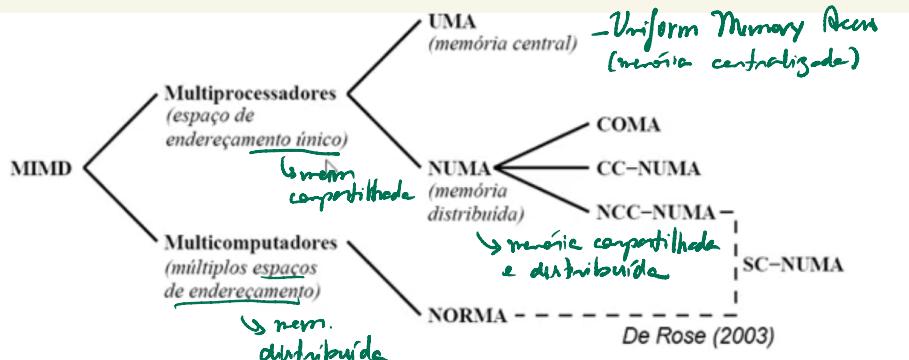
multiprocessadores
(memória compartilhada)



multicomputadores
(memória privada)

→ troca de mensagens
pela rede

- escalabilidade não é total
- programação realizada c/ threads
- desempenho maior (cache)
- problema de coerência de cache
- Dual Pentium (VMA, multicores)



A linha tracejada indica que através de um software que implemente coerência de *cache*, as máquinas NCC-NUMA e NORMA podem se transformar em máquinas SC-NUMA.

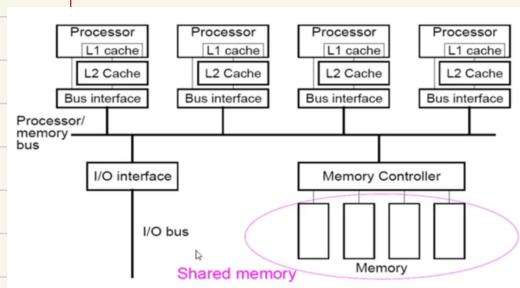
Arquiteturas

UMA

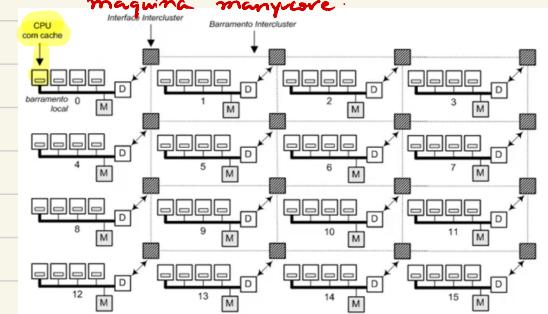
vs

NUMA

- Uniform Memory Access
- memória compartilhada centralizada
- protocolos de coerência de cache
- chega em um limite de amento de módulos centralizados que aceita
- desempenho.



maquina multicore:



NC-NUMA ↘

NUMA que não utiliza cache

CC-NUMA

Cache-Coherent NUMA -
NUMA com cache (e co-
rência de cache)

Coerência de Cache:

→ cada módulo de memória tem um diretório local que armazena info de onde as cópias dos blocos estão guardados

→ os protocolos de diretório enviam comandos de consistência p/ os caches que tem uma cópia válida do bloco de dados compartilhado.

- UMA: Uniform Memory Access
- NUMA: Non-Uniform Memory Access
- CC-NUMA: Cache-Coherent NUMA
- NCC-NUMA: Non-Cache-Coherent NUMA
- SC-NUMA: Software-Coherent NUMA
- COMA: Cache-Only Memory Architecture
- NORMA: Non-Remote Memory Access

Técnicas de Coerência de Cache

- bloquinhos que ficam "extraindo" o barramento p/ alocar um ataz
- > **Snooping**: Um método p/ manter coerências de cache, em que todos os controladores de cache monitoram o barramento p/ determinar se põem ou não uma cópia do bloco desejado.
- > **Write-validate**: Uma variação de snooping em que o processador de escrita faz com que todos os outros ^(memória) caches sejam invalidados antes de mudar sua cópia local, e que permite atualizar os dados locais até que outro processador os solicite.
- > **Protocols de Coerência de Cache = MSI, MESI, MOESI**

★ Memórias Distribuídas

- Grupo de computadores autônomos (mís) que trabalham juntos como um recurso único.
- os mís são interconectados por redes de alta disponibilidade
 - escalabilidade abs.oluta e incremental
 - alta disponibilidade
 - excelente custo benefício.
 - comunicação por pacotes de mensagens
 - MPI (Message Passing Interface)
 - PVM (Parallel Virtual Machine).
- caducidade
cada vez +
mís)

► Cluster → aglomerados de computadores

- Grids → grades computacionais = Plataforma p/ execução de aplicações paralelas
- amplamente distribuída
 - heterogênea
 - c/ múltiplos domínios administrativos
 - - - - -
 - - - - -
 - - - - -
 - - - - -

Comparação de Computadores

ESCALAR vs SUPERESCALAR



- execução sequencial (1 instrução por ciclo de clock)
- pipeline permite várias instruções em estágios diferentes do pipeline, mas só 1 instrução é concluída por ciclo.
- geram stalls no pipeline quando há dependências de dados.

SUPERESCALAR



- tem paralelismo → executa múltiplas instruções por ciclo de clock usando unidades funcionais paralelas.
- tem unidades especializadas printeiras, floats e memórias
- tem preditor de desvios
- tem despacho dinâmico (detecta e resolve dependências de dados)

Supercomputador vs Multiprocessador



- Máquina de alto desempenho
- Pode usar milhares de processadores interligados, seja em arquiteturas de multiprocessamento ou multicore.
- ✓ memória compartilhada
- ✓ computadores
- ✓ memória distribuída

Multiprocessador vs Multicomputador



- Sistema que possui 2 ou + CPUs dentro de um único computador, compartilhando 1 memória principal.
- ISO distribui as tarefas entre os processadores (que executam os processos independentemente).

Multicomputador



- Sistema paralelo construído por vários computadores independentes conectados por uma rede, cada um c/ sua memória local própria (distribuída)
- Um exemplo são os clusters de computadores.

Processadores Multicore vs Processadores Manycore

- 2 a 32 núcleos
- núcleos superscalares
- memória compartilhada
- Intel Core, AMD, Ryzen

Processadores Manycore

- Dezenas a Milhares de núcleos
- núcleos simplificados
- memória distribuída
- GPUs (NVIDIA), Sunway

↳ supercomputadores

e aceleradores de IA

ps: Leitura após Leitura não dar nenhum problema!

Dependências - Revisão

Tipo	Sigla	Causa	Exemplo	Solução Típica
Verdeceira →	Read After Write	Leitura antes da escrita concluída	ADD R1, ... → SUB ..., R1	<u>Forwarding / Stall</u>
Antidependências →	Write After Read	Escrita antes da leitura concluída	ADD ..., R3 → SUB R3, ...	<u>Renomeação de registradores</u> (evita reutilização)
Saida →	WAW	Escratas concorrentes (duas escritas no mesmo registrador)	ADD R1, ... → SUB R1, ...	Reordenação / Renomeação
De controle →	-	Desvio condicional → devios que interrompem o fluxo sequencial do pipeline	BEQ... → instrução seguinte	<u>Previsão de desvios</u>
Estrutural →	-	Conflito de recursos → Socorro simulâneo à memória	Duas LW simultâneas	<u>Interlock / Alocação dinâmica</u>

Multi-Core vs Manycore

* processadores c/
múltiplos núcleos
compartilhados

* processadores c/
muitos de núcleos
simplificados e
distribuídos

Memória Compartilhada vs Distribuída

* Compartilhada vs Distribuída
todos os processadores acervam
o mesmo espaço
de memória
multiprocessadores

cada processador
tem sua
memória
local

multiprocessadores

e como eles
se comunicam?

Redes - em - Chip

* interconexão robusta
entre múltiplos núcleos
usando redes internas no
chip (como topologia mesh) p/
melhorar a escalabilidade
e comunicação.

Multithreading

* SMT vs JMT vs BMT
(simultâneo) (interleaved) (bloqueado)

múltiplos threads no
mesmo tempo

alternância entre
threads (grão)
(entre instruções)

trabalho entre blocos
de threads quando
não latência

1 thread por vez

Parallelismo e Instruções

* escalar vs superescalar
instrução por ciclo,
sem paralelismo interno
por ciclo.

Despacho e Instruções

* despacho em ordem vs fora da ordem
instruções despachadas e int. despachadas
completadas na mesma
ordem que chegaram

(se não tiver
conflieto de recursos
ou dependência de
dados)

* dependências
(reemboração)

→ verdadeira RAW
- Read after Write
(forwarding)

→ antidependência
WAR - Write after Read
(renomeação
de registradores)

→ de saida
WAW - Write after Write
(reordenação/renomeação)

* renomeação
de registradores

usado se dependência falsa,
atribuindo nomes temporários
aos registradores p/ ter
+ paralelismo.

* Classificações Flynn

superprocessadores vs supercomputadores
memória compartilhada
memória privada