DCC-IM/NCE UFRJ
Pós-Graduação em Informática

Microarquiteturas de Alto Desempenho

Hierarquia de Memória

Gabriel P. Silva

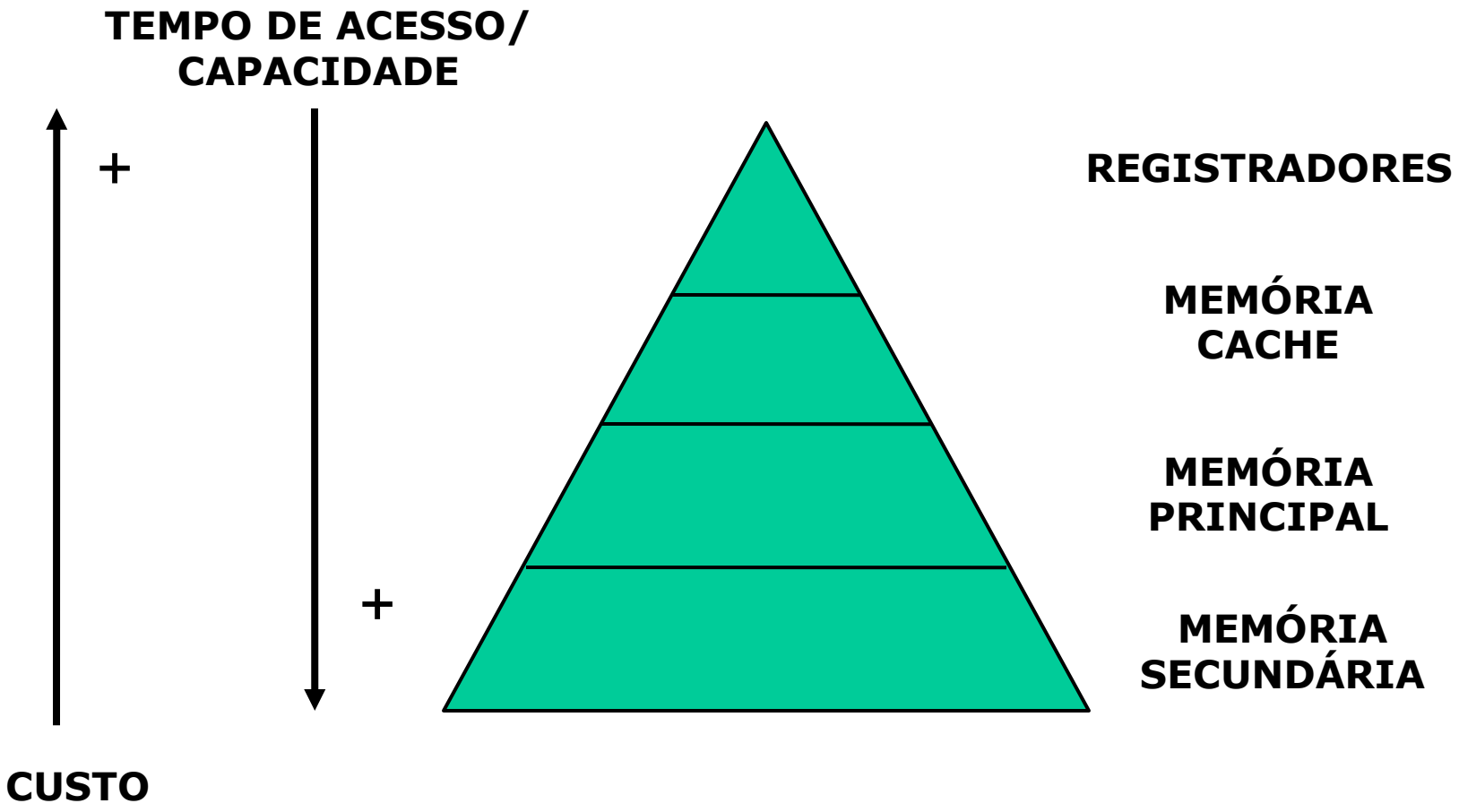
Introdução

- **Os programadores sempre ambicionaram ter quantidades ilimitadas de memória rápida.**
- **Contudo, as memórias rápidas são de alto custo e, normalmente, de pequena capacidade também.**
- **Uma solução é a organização do sistema de memória em uma hierarquia, com diversos níveis, onde memórias cada vez mais rápidas, menores e com um custo por byte maior, são colocadas nos níveis mais altos.**

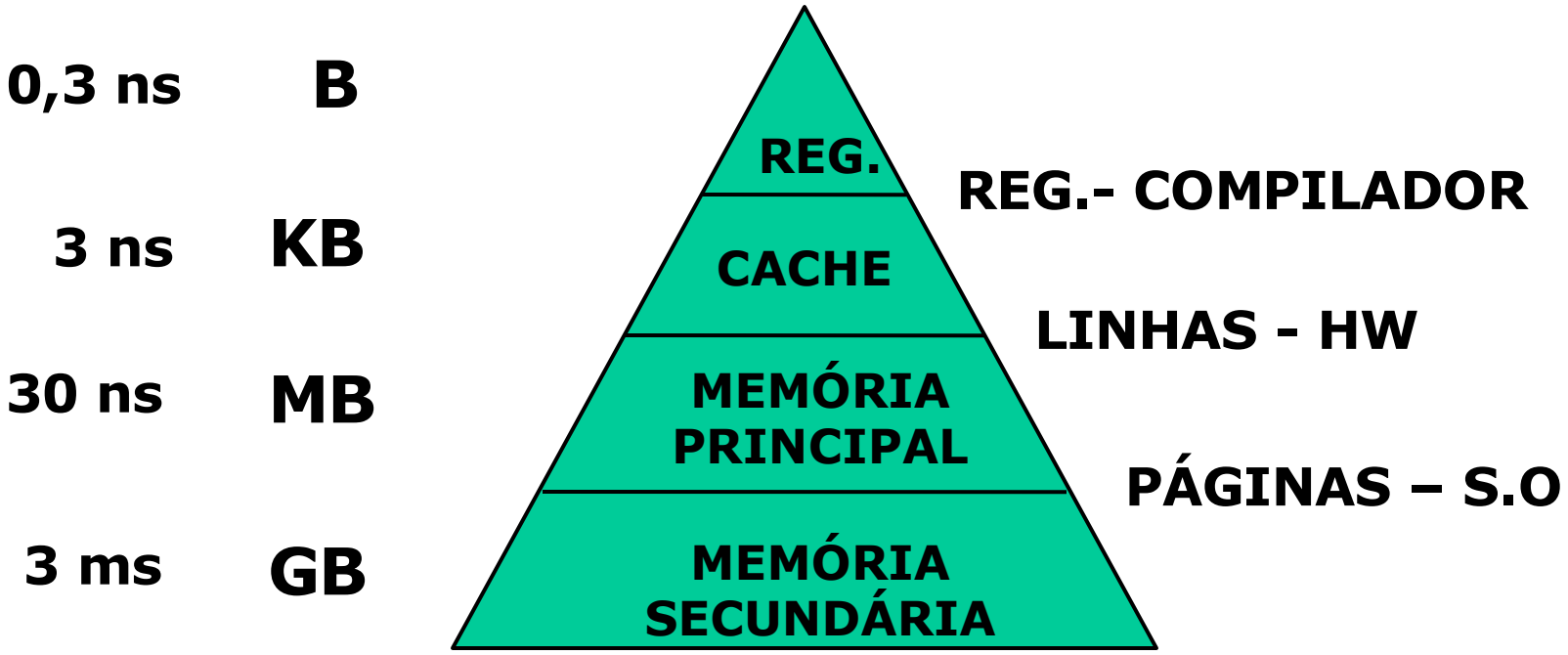
Introdução

- **O objetivo é fornecer um sistema de memória com um custo próximo daquele do nível mais baixo da hierarquia, e velocidade próxima daquela do nível mais alto.**
- **Os níveis de hierarquia mais altos normalmente são um subconjunto dos níveis mais baixos.**
- **À medida que a informação vai sendo utilizada, ela vai sendo copiada para os níveis mais altos da hierarquia de memória.**

Hierarquia de Memória



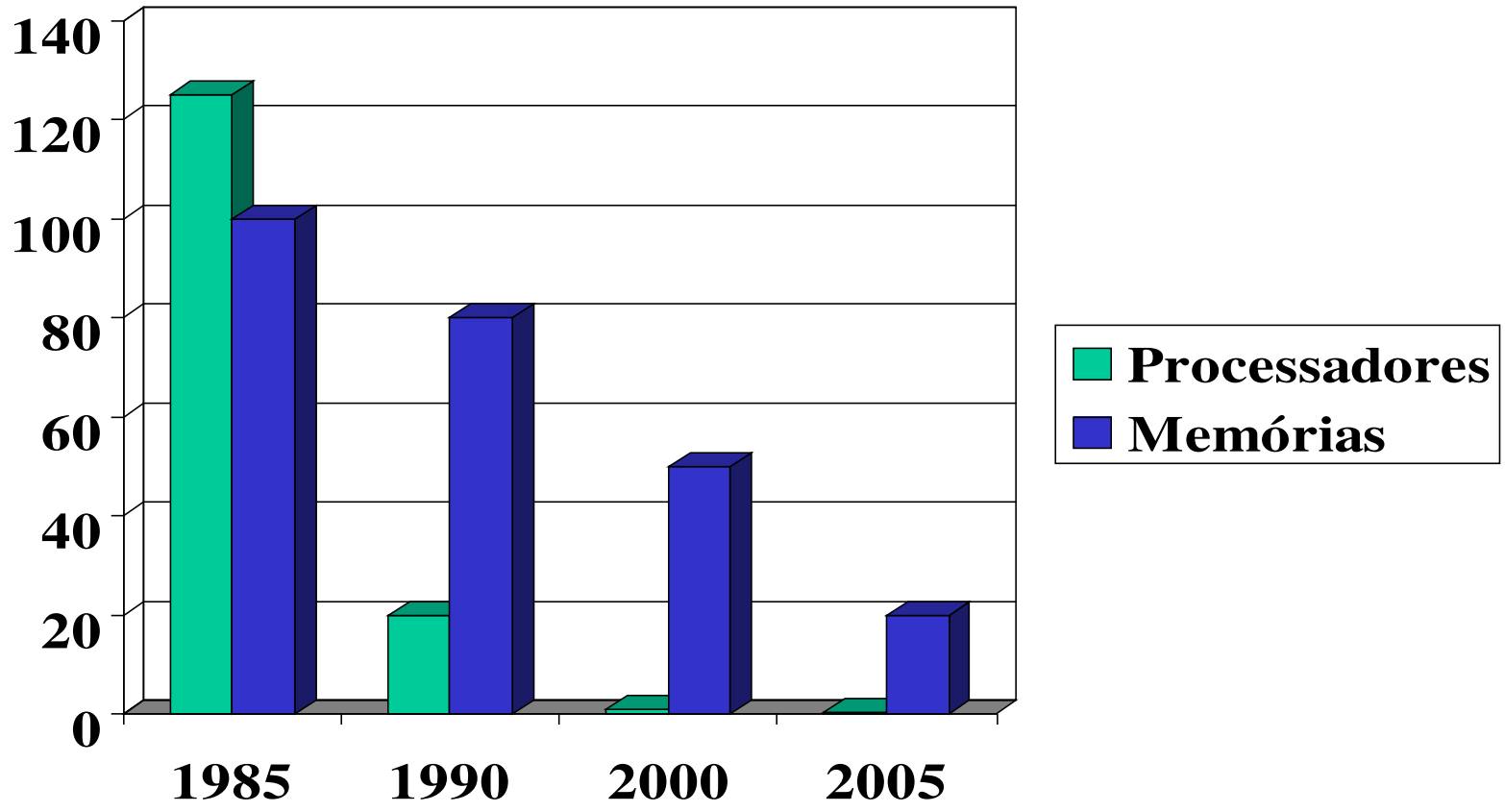
Hierarquia de Memória



Hierarquia de Memória

- **Note que a cada nível que se sobe na hierarquia, endereços de uma memória maior são mapeados para uma memória menor.**
- **Junto com esse mapeamento está associadoa uma função de proteção, evitando que dados de um usuário sejam modificados por outro.**
- **A importância da hierarquia de memória aumentou nos últimos anos devido ao aumento no desempenho dos processadores.**

Velocidade dos Processadores x Memórias



Conceitos de Localidade

- **O funcionamento da hierarquia de memória está fundamentado em duas características encontradas nos programas.**
- **Existe uma grande probabilidade de o processador executar os mesmos trechos de código e utilizar repetidamente dados próximos**
- **A essa qualidade dos programas denominamos:**
  - **Localidade Temporal**
  - **Localidade Espacial**

Conceitos de Localidade

- **Localidade temporal: posições de memória, uma vez referenciadas (lidas ou escritas) , tendem a ser referenciadas novamente dentro de um curto espaço de tempo.**
  - **Usualmente encontrada em laços de instruções e acessos a pilhas de dados e variáveis**
- **Localidade espacial: se uma posição de memória é referenciada, posições de memória cujos endereços sejam próximos da primeira tendem a ser logo referenciados.**
  - **A informação é manipulada em blocos no sistema de hierarquia de memória para fazer uso da localidade espacial.**

Hierarquia de Memória

- **No início dos tempos o único nível que possui informação válida é o mais inferior de toda a hierarquia, composto de dispositivos de armazenamento não voláteis.**
- **Na medida em que a informação vai sendo utilizada, ela é copiada para os níveis mais altos da hierarquia.**
- **Quando fazemos um acesso a um nível da hierarquia e encontramos a informação desejada, dizemos que houve um acerto, em caso contrário dizemos que houve uma falha.**

Hierarquia de Memória

- **Acessos que resultam em acertos nos níveis mais altos da hierarquia podem ser processados mais rapidamente.**
- **Os acessos que geram falhas, obrigando a buscar a informação nos níveis mais baixos da hierarquia, levam mais tempo para serem atendidos.**
- **Para um dado nível da hierarquia possa ser considerado eficiente é desejável que o número de acertos seja bem maior do que o número de falhas.**

Hierarquia de Memória

- **Define-se como taxa de acertos (h) a relação entre o número de acertos e o número total de acessos para um dado nível da hierarquia de memória.**

h = número acertos / total de acessos

- **O total de acessos inclui tanto os acessos de leitura como os de escrita.**

Hierarquia de Memória

- **O tempo de acesso com acerto é o tempo necessário para buscar a informação em um dado nível de hierarquia, que inclui o tempo necessário para determinar se o acesso à informação vai gerar um acerto ou uma falha.**

- **A penalidade por falha ou tempo acesso com falha é o tempo necessário para buscar a informação nos níveis inferiores da hierarquia, armazená-la no nível atual e enviá-la para o nível superior.**

Taxa de Acerto

- **Taxa de Acerto h: probabilidade de que uma posição referenciada seja encontrada em dado nível da hieraquia de memória.**

Ta = tempo de acesso com acerto
Tf = tempo de acesso com falha
Tma = tempo médio de acesso

Tma = h*Ta +(1 – h)*Tf

Se Ta = 10 ns
Tf = 80 ns
h = 0,85 então Tma = 20,5 ns

Se h = 1 então Tma = Ta

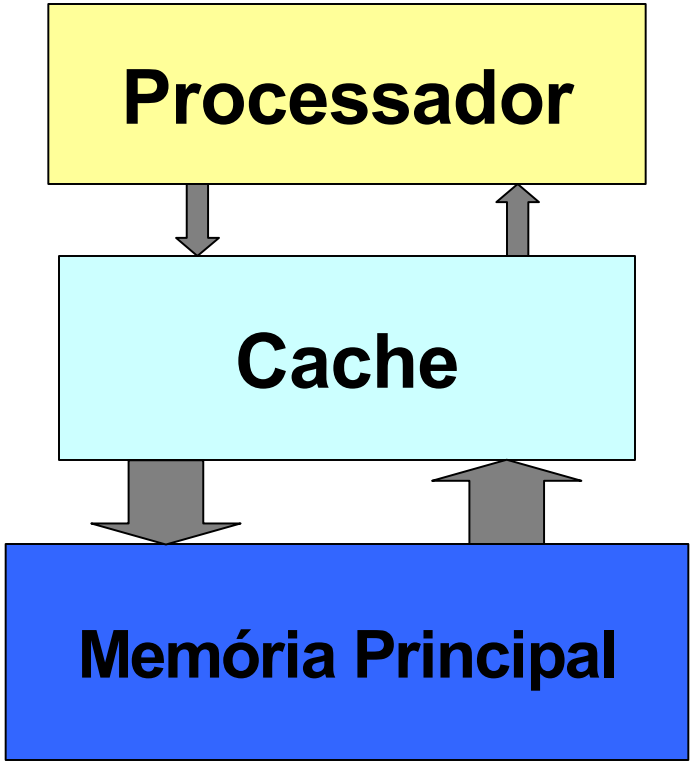
Componentes da Hierarquia

- **Os elementos mais importantes da uma hierarquia de memória são:**
  - **Registradores**
  - **Memória Cache**
  - **Memória Principal**
  - **Memória Secundária**

Registradores

- **Os registradores estão localizados no núcleo do processador. São caracterizados por um tempo de acesso menor que um ciclo de relógio e sua capacidade é da ordem de centenas de bytes.**
- **O controle de qual informação deve estar nos registradores é feita explicitamente pelo compilador, que determina quais variáveis serão colocadas no registrador.**
- **É o único nível da hierarquia que permite movimentações iguais apenas ao tamanho da informação desejada.**

Memória Cache



Memória Cache: Elemento de memória intermediário entre o Processador e a Memória Principal

Funcionamento da Cache

- **O processador inicia a busca a instrução ou dados na memória cache.**
- **Se os dados ou a instrução estiverem na cache (denomina-se acerto), a informação é transferida para o processador.**
- **Se os dados ou a instrução não estiverem na memória cache (chama-se falha), então o processador aguarda, enquanto a instrução/dados desejados são transferidos da memória principal para a cache e também para o processador.**

Funcionamento da Cache

- **Durante a busca da palavra que está faltando na cache, é trazido um bloco (ou linha) inteiro da memória principal, ao invés de apenas uma palavra.**
- **O objetivo é minimizar a taxa de falhas nos próximos acessos, seguindo o princípio da localidade espacial.**
- **O tamanho de um bloco é um parâmetro de projeto na memórias caches, tamanhos usuais são 32, 64 e 128 bytes.**

Funcionamento da Cache

- **É necessário guardar o endereço do bloco, ou parte dele, para permitir a identificação dos blocos que estão armazenados na cache.**
- **No momento em que for feita uma leitura de dados ou instrução pelo processador, os endereços armazenados na cache são comparados com o endereço fornecido pelo processador, para saber se houve um acerto ou falha.**
- **Caso haja acerto, a informação armazenada na memória cache é fornecida para o processador, evitando a ida à memória principal.**

Memória Cache

- **As células de memória da cache são elaboradas com tecnologia que permite um tempo de acesso menor que o memória principal.**
- **Normalmente são células de memória estática (SRAM), com menor capacidade, porém maior custo e maior velocidade.**
- **Associado a essas memórias está um controlador, normalmente uma máquina de estados, que faz o controle da transferência dos blocos de/para a memória principal.**

Memória Principal

- **A memória principal é constituída por células de memória dinâmica (DRAM).**
- **As memórias DRAM tem grande capacidade de armazenamento, mas são mais lentas que as memórias estáticas.**
- **As memórias DRAM possuem também tempos de acesso distintos para leitura e escrita, e necessitam de uma lógica de restauração ("refresh"), quer embutida ou fora da pastilha, que afetam o tempo médio de acesso.**

Memória Secundária

- **A memória secundária é o último nível da hierarquia de memória. É composta pelos dispositivos de armazenamento de massa, normalmente discos rígidos, de grande capacidade e menor custo por byte armazenado.**
- **Os programas e arquivos são armazenados integralmente na memória secundária, que são dispositivos de memória não volátil.**

Memória Virtual

- **O controle de qual informação deve permanecer na memória principal ou na memória secundária é feito pela Memória Virtual.**
- **A memória virtual é um conjunto de "hardware" e de rotinas do sistema operacional. Além do controle da hierarquia entre a memória principal e a memória secundária, ela realiza a proteção, evitando que um programa modifique informações que pertençam a algum outro.**

Memória Virtual

- **Finalmente, a memória virtual também faz a translação de endereços virtuais em endereços reais, já que os programas normalmente enxergam um espaço de endereçamento maior que a memória física.**
- **As translações mais freqüentes ficam armazenadas em uma pequena memória associativa chamada TLB.**
- **O método mais usual utilizado pela memória virtual é a divisão do espaço de endereçamento em páginas de tamanho fixo, que são a unidade de transferência entre o disco e a memória principal.**

Aspectos Operacionais Comuns

- **Apesar de muitos aspectos dos componentes da hierarquia de memória diferirem quantitativamente, muitas das políticas e das características são semelhantes em termos qualitativos:**
  - **Onde se colocar uma linha ou página?**
  - **Qual o tamanho de uma linha ou página?**
  - **Como encontrar uma linha ou página?**
  - **Qual das linhas/páginas deve ser substituída quando ocorrer uma falha e não houver espaço?**
  - **O que acontece em uma escrita?**

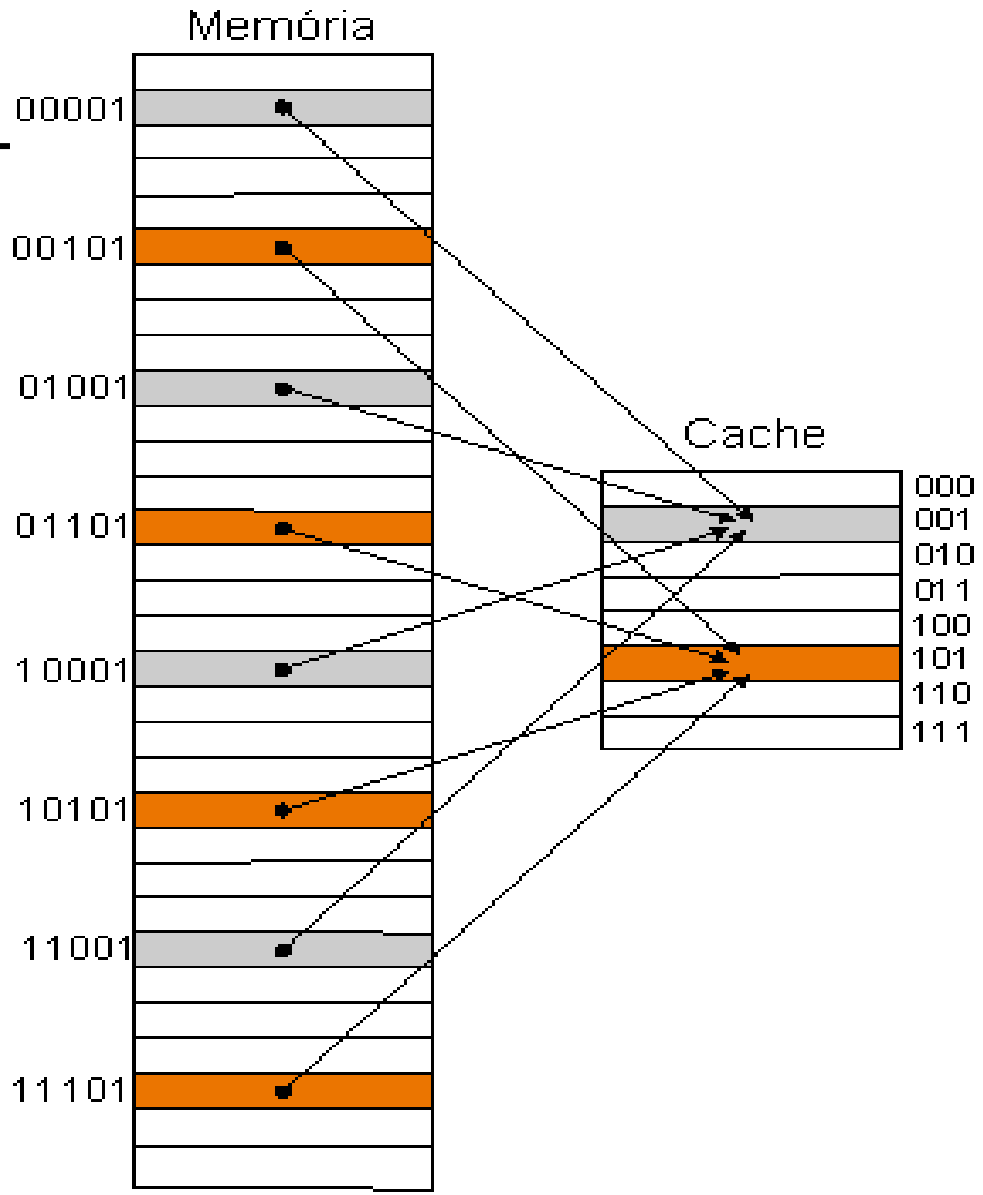
Onde Colocar um Bloco?

- **Esta questão afeta apenas a memória cache, já que o método utilizado pela memória virtual é o mapeamento totalmente associativo.**
- **Ou seja, no caso da memória virtual uma página do programa pode ser colocada em qualquer posição da memória física.**
- **Em nosso estudos, consideramos que a capacidade da memória cache será sempre dada em múltiplo inteiro de linhas / blocos.**
- **Existem três maneira possíveis de tratar este problema, chamadas de mapeamentos:**
  - **Mapeamento Totalmente Associativo**
  - **Mapeamento Direto**
  - **Mapeamento Associativo por Conjunto**

Onde Colocar um Bloco?

- Mapeamento completamente associativo
  - Um bloco da memória principal pode ser armazenado em qualquer linha da memória cache.
- Mapeamento direto
  - Cada bloco da memória principal só pode ser mapeado em uma única linha da memória cache. Normalmente utilizam-se os bits menos significativos do endereço do bloco para definir qual será esta linha.
- Mapeamento associativo por conjunto
  - Cada bloco da memória principal pode ser armazenado apenas em um determinado conjunto de linhas da cache.

Mapeamento Direto



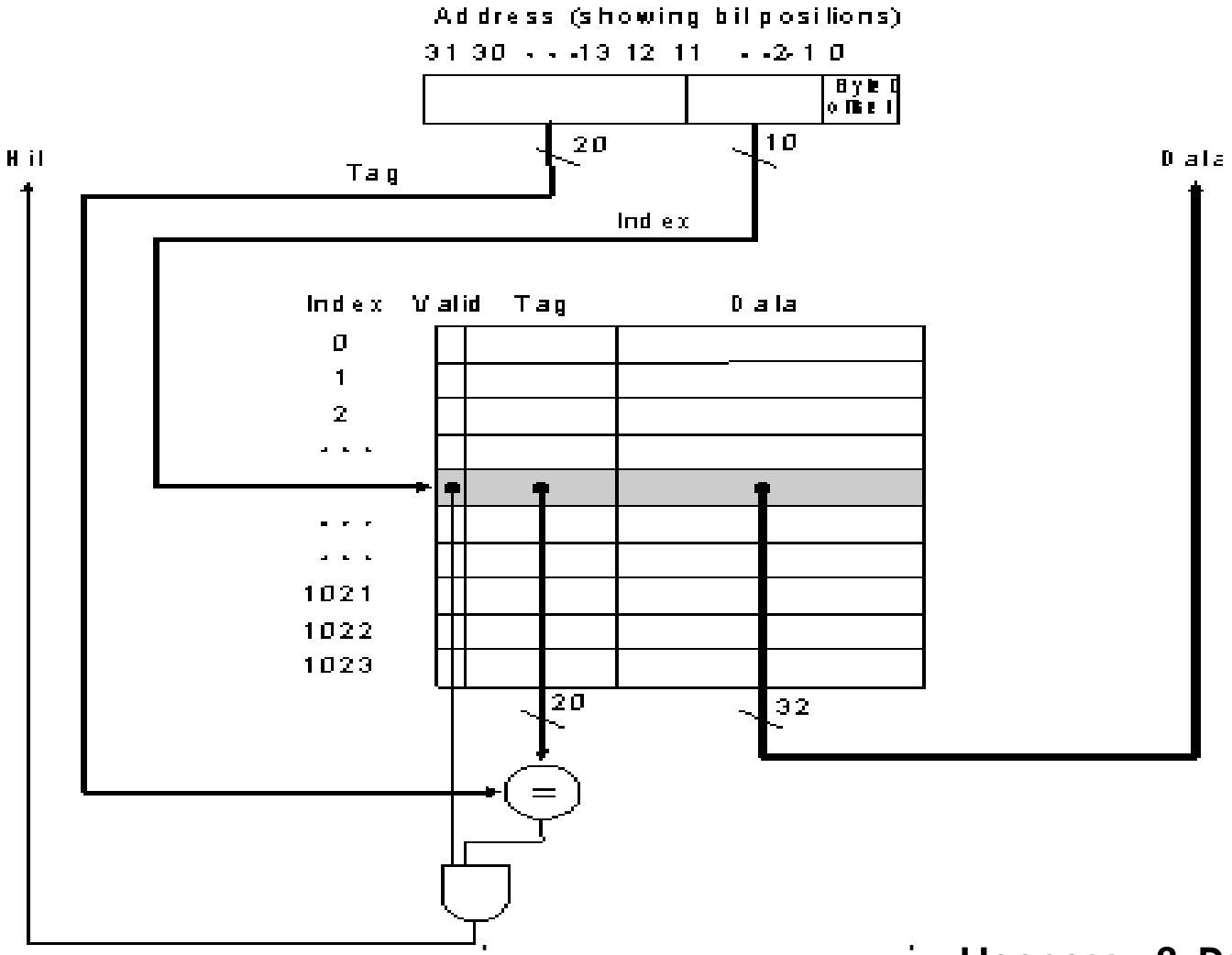
Onde Colocar um Bloco?

- **Mapeamento Totalmente Associativo**
  - **Todos os blocos da memória principal podem ser armazenados em qualquer linha na cache;**
  - **Existe 1 único conjunto na memória cache;**
  - **O número de linhas por conjunto é igual a capacidade total de linhas da cache.**
- **Características**
  - **Demorado para descobrir se um bloco reside ou não na cache;**
  - **A substituição de uma linha só é necessária quando a cache está totalmente cheia;**
  - **O endereço do bloco deve ser totalmente armazenado junto com a linha;**

Onde Colocar um Bloco?

- **Mapeamento Direto**
  - **Cada bloco só pode ocupar uma determinada linha na cache, definida pelo seu índice, que são os bits mais baixos do endereço de memória do bloco;**
  - **Existem tantos conjuntos quantos forem as linhas da cache;**
  - **Existe apenas 1 linha em cada conjunto.**
- **Características**
  - **Rápido para descobrir se um bloco está presente na cache;**
  - **Pode ocorrer que dois blocos freqüentemente referenciados estejam mapeados na mesma linha da cache;**
  - **Apenas parte do endereço precisa ser armazenada precisa ser armazenada junto com o bloco na cache.**

Mapeamento Direto



Hennessy & Patterson

Onde Colocar um Bloco?

- **Mapeamento Associativo por Conjunto**
  - **Cada bloco pode ser armazenado em um determinado conjunto de linhas da cache.**
  - **O número de linhas que existem em cada conjunto é definido como a associatividade (normalmente de 2 a 8) da cache.**
  - **O número de conjuntos é dado pela fórmula:**
    - **Número de linhas na cache / associatividade**
- **Características**
  - **Reduz as chances de conflito.**
  - **É rápido para descobrir se um bloco está armazenado na cache.**

Onde Colocar um Bloco?

- **A vantagem de se aumentar o grau de associatividade é a conseqüente queda na taxa de falhas.**
- **Isso se deve a uma menor competição pela mesma posição na memória cache.**
- **Essa melhora é tão mais significativa quanto menor for a capacidade da memória cache.**
- **As desvantagens de se aumentar a associatividade são um ligeiro aumento no custo e no tempo de acesso.**

Qual o Tamanho um Bloco?

- **Tamanhos de bloco/páginas maiores tendem a fazer uso melhor da localidade espacial, aumentando a taxa de acerto.**
- **Contudo, o uso de bloco maiores nas memórias caches tende a aumentar a penalidade por falha.**
- **Tamanhos típicos de blocos situam-se entre 16 e 128 bytes.**
- **Uso de páginas maiores tende a diminuir o tamanho da tabela de páginas, melhorando a penalidade por falha das TLBs.**
- **Tamanhos típicos de páginas situam-se entre 1K a 16 Kbytes.**

Como Encontrar um Bloco?

- **A forma de localização de um bloco depende do esquema de mapeamento utilizado.**
- **No caso do mapeamento direto é necesário apenas uma comparação.**
- **No caso do mapeamento associativo, são necessárias tantas comparações quanto forem os blocos/linhas em cada conjunto.**
- **No caso das memórias totalmente associativas, é necessário que todas as posições da memória cache sejam comparadas.**
- **Normalmente essas comparações são realizadas em paralelo por vários circuitos.**

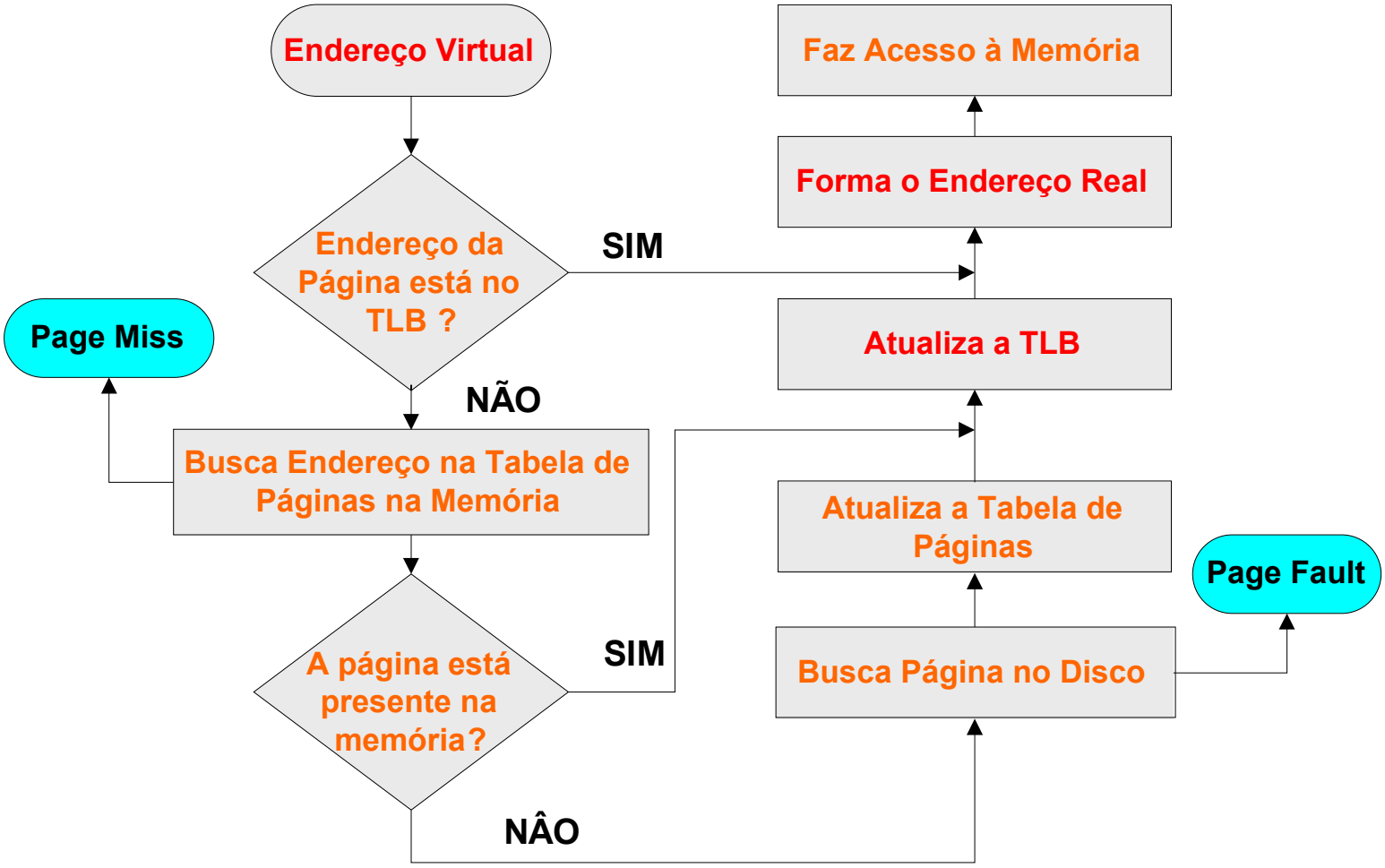
Como Encontrar um Bloco?

- **Nos sistemas de memória virtual, mantém-se uma tabela de mapeamento, chamada de tabela de páginas, para indexar a memória principal.**
- **Esta tabela contém todas as translações de endereços virtuais para endereços físicos de todas as páginas em uso.**
- **Caso a página não esteja carregada na memória, a posição correspondente na memória principal é marcada como inválida.**

Como Encontrar um Bloco?

- **Para cada acesso seria necessário um acesso extra à memória para consulta à tabela de páginas.**
- **Para solucionar este problema, utiliza-se uma pequena memória totalmente associativa (TLB), localizada junto ao processador, que guarda as translações realizadas mais freqüentemente.**
- **Uma vez feito o primeiro acesso, a translação é armazenada e todos os demais acessos a essa página podem ser feitos sem atrasos adicionais relativos à translação.**

Esquema de Memória Virtual Paginada



Qual Bloco deve ser Substituído?

- **Em uma cache de mapeamento direto há apenas 1 bloco por conjunto. Quando ocorre uma falha é este o bloco que vai ser substituído.**
- **Nas caches associativas, caso todas as posições do conjunto estejam ocupadas, é necessário escolher qual dos blocos do conjunto vai ser substituído, segundo uma das políticas:**
  - **Aleatória**
  - **LRU**
  - **FIFO**

Algoritmos de Substituição

Algoritmo de Substituição Aleatório (randômico)

- **Um bloco é escolhido aleatoriamente para ser substituído. É uma política de fácil implementação, mas gera problemas de desempenho em esquemas totalmente associativos.**

Substituição por Fila FIFO (first-in first-out)

- **O bloco que está há mais tempo na cache é removido. Menos simples de implementar e pode diminuir a taxa de acerto quando o bloco mais antigo ainda estiver sendo muito utilizado.**

Substituição LRU (Least Recently Used)

- **O bloco a ser substituído é aquele que não é referenciado há mais tempo. É o esquema de melhor desempenho, mas cuja implementação é a mais complexa.**

Qual Página deve ser Substituída?

- **No sistema de memória virtual, utilizam-se os mesmos algoritmos para decidir qual a página que deve ser removida da memória.**
- **Um bit de *modificado* na tabela de páginas indica se a página removida deve ser atualizada em disco.**
- **Nos modernos sistemas operacionais, para cada processo, apenas um subconjunto de suas páginas é mantido em memória. Ele é denominado de "conjunto de trabalho".**
- **O universo das páginas candidatas à substituição pode ser local (do mesmo conjunto de trabalho) ou global (todas as páginas de todos os processos na memória são analisadas).**

O que Acontece na Escrita?

- **Ao ser escrever em um bloco, temos duas políticas básicas em caso de acerto:**
  - **Write-through (Cache)**
    - **Escreve-se o dado no nível da hierarquia atual e no inferior**
  - **Write-back (Memória Virtual e Cache)**
    - **Escreve-se o dado apenas no nível de hierarquia atual e, quando o bloco ou página for substituído, ele é atualizado no nível inferior.**

Write-through

- As falhas são mais simples de tratar, pois nunca há necessidade de escrever-se todo o bloco no nível mais inferior da hierarquia, apenas a palavra sendo atualizada.
- Essa técnica é mais simples de se implementar.
- Quando existe mais de uma cache no mesmo barramento, permite a implementação fácil de técnicas de "snooping" para a manutenção da coerência entre as caches.

Write-back

- As palavras podem ser escritas na velocidade da cache, ao invés da memória principal;
- Várias escritas para o mesmo bloco são atualizadas em uma única operação no nível mais inferior da hierarquia;
- Uso de transferências em rajada para atualização do bloco no nível de hierarquia inferior;
- Quando houver mais de uma cache no barramento, há a necessidade de se adotar protocolos mais complexos para a manutenção da coerência das caches.

Write-back

- **Esses protocolos visam forçar a colocação no barramento de informações sobre operações de escrita sempre que necessário, pois a memória principal nem sempre possui uma cópia válida do bloco que está sendo solicitado.**
- **Esses protocolos definem a responsabilidade sobre quem deve fornecer um bloco de dados quando uma operação de leitura com falhas ocorre.**
- **No caso de memória virtual, o algoritmo utilizado para atualização das páginas em disco é sempre o "write-back".**

O que Acontece na Escrita?

- **Se houver uma falha, duas estratégias são possíveis de se utilizadas:**
  - **Write Allocate**
    - **O bloco onde vai ser feita a operação de escrita é trazido primeiramente para a memória cache e a operação de escrita é então realizada.**
  - ***No Write Allocate***
    - **O bloco a ser escrito não é trazido para a memória cache e, portanto, a operação de escrita sempre se realiza apenas na memória principal.**
  - **Usualmente a política *write allocate* é adotada com o modo *write back* e a política *no write allocate* com o modo *write through*.**

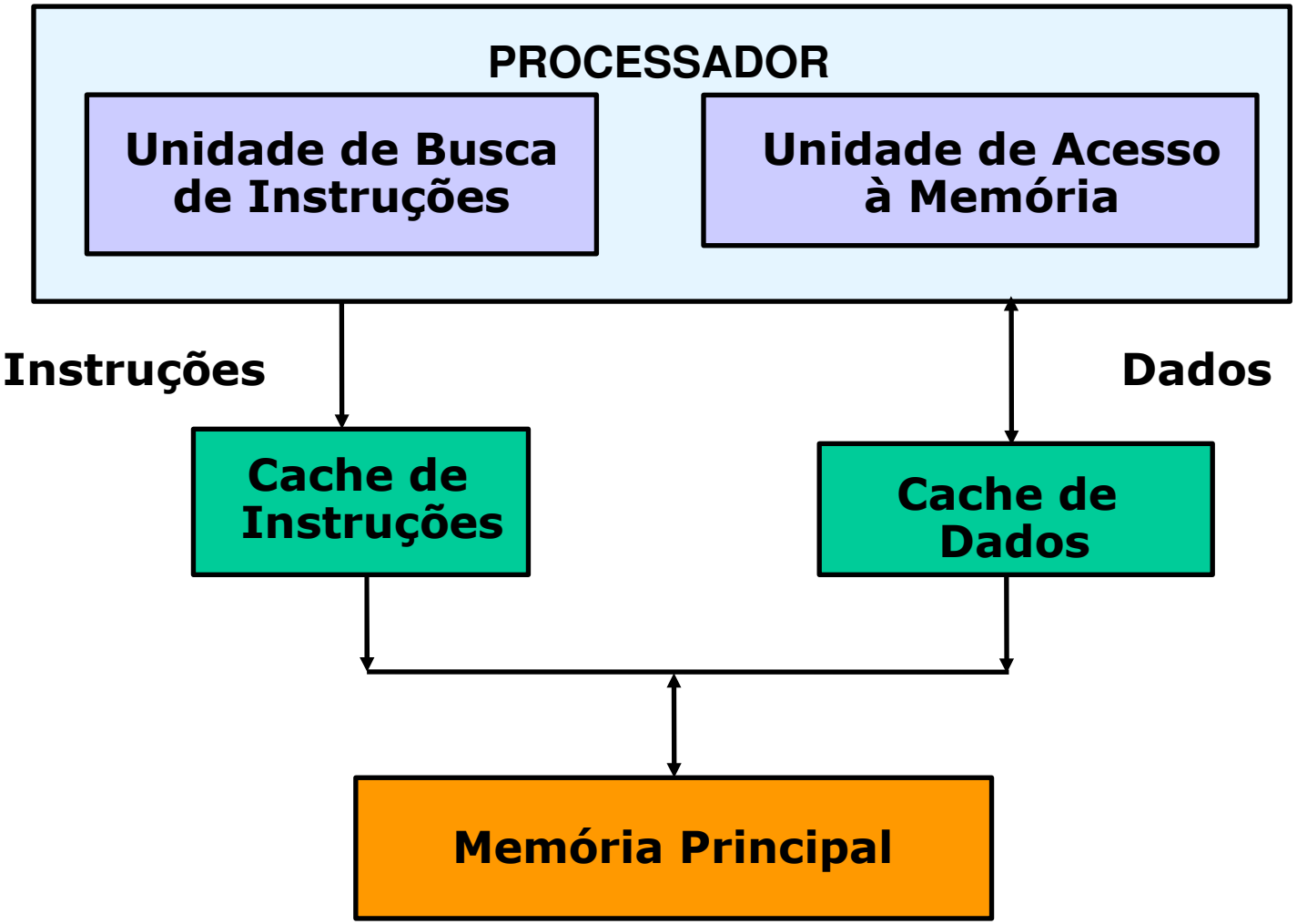
Cache Virtual

- **Endereços virtuais dos blocos de memória são armazenados na cache.**
- **A leitura do bloco na cache pode ser feita em paralelo com a conversão do endereço virtual em físico pela gerência de memória.**
- **Existe a possibilidade de ocorrência de "aliasing": dois ou mais endereços virtuais idênticos gerados por processos diferentes, que se referem a endereços físicos distintos ➔ precisa ser realizado um *flush* a cada troca de processo.**
- **De uso difícil em ambientes com múltiplos processadores.**

Cache Físico

- **Os endereços físicos dos blocos de memória são armazenados na cache.**
- **O tempo de acesso é mais lento porque o acesso à cache deve ocorrer após a conversão do endereço virtual em físico.**
- **Mais simples de implementar e usar em ambientes com múltiplos processadores.**

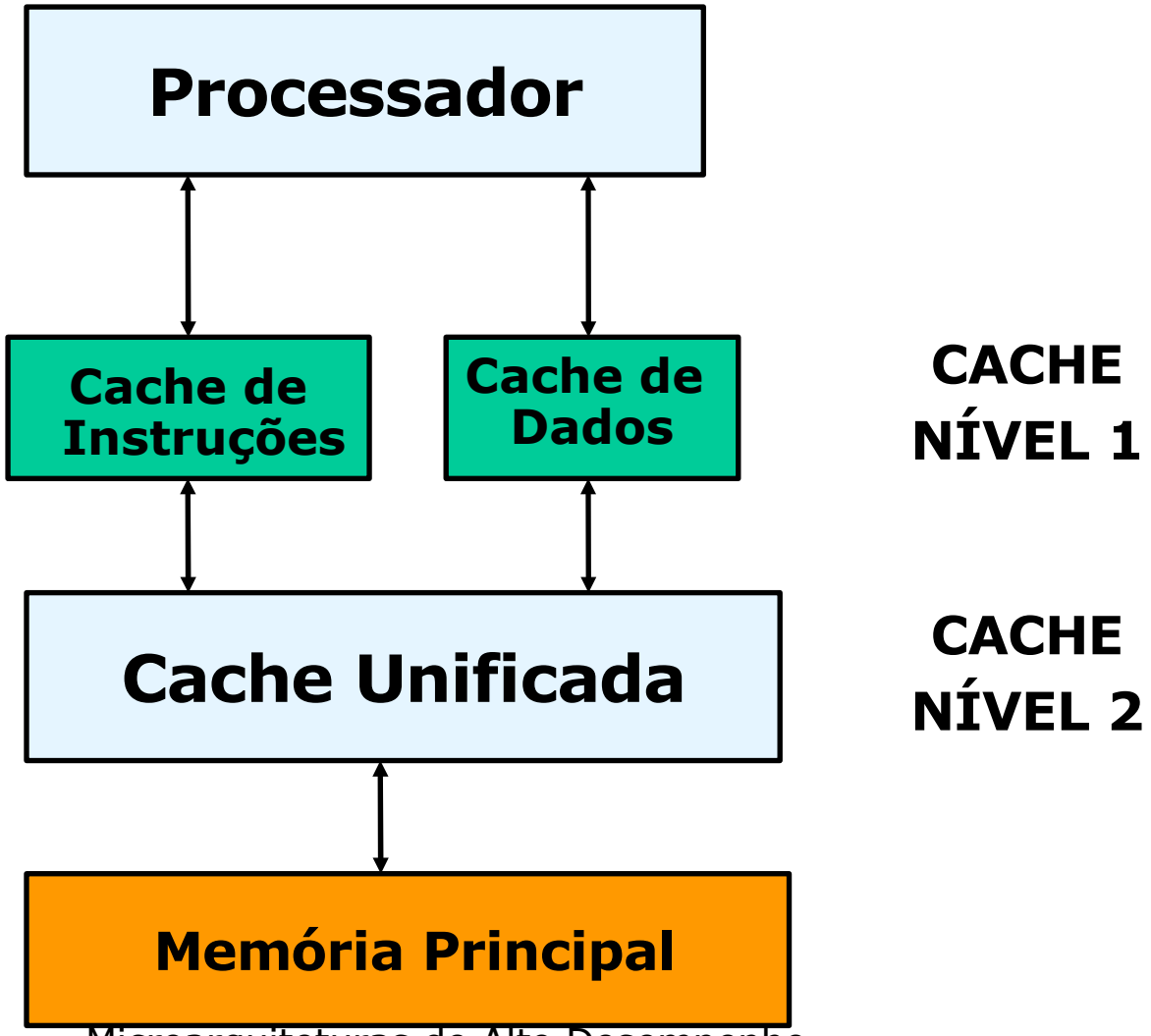
Caches de Dados e Instruções



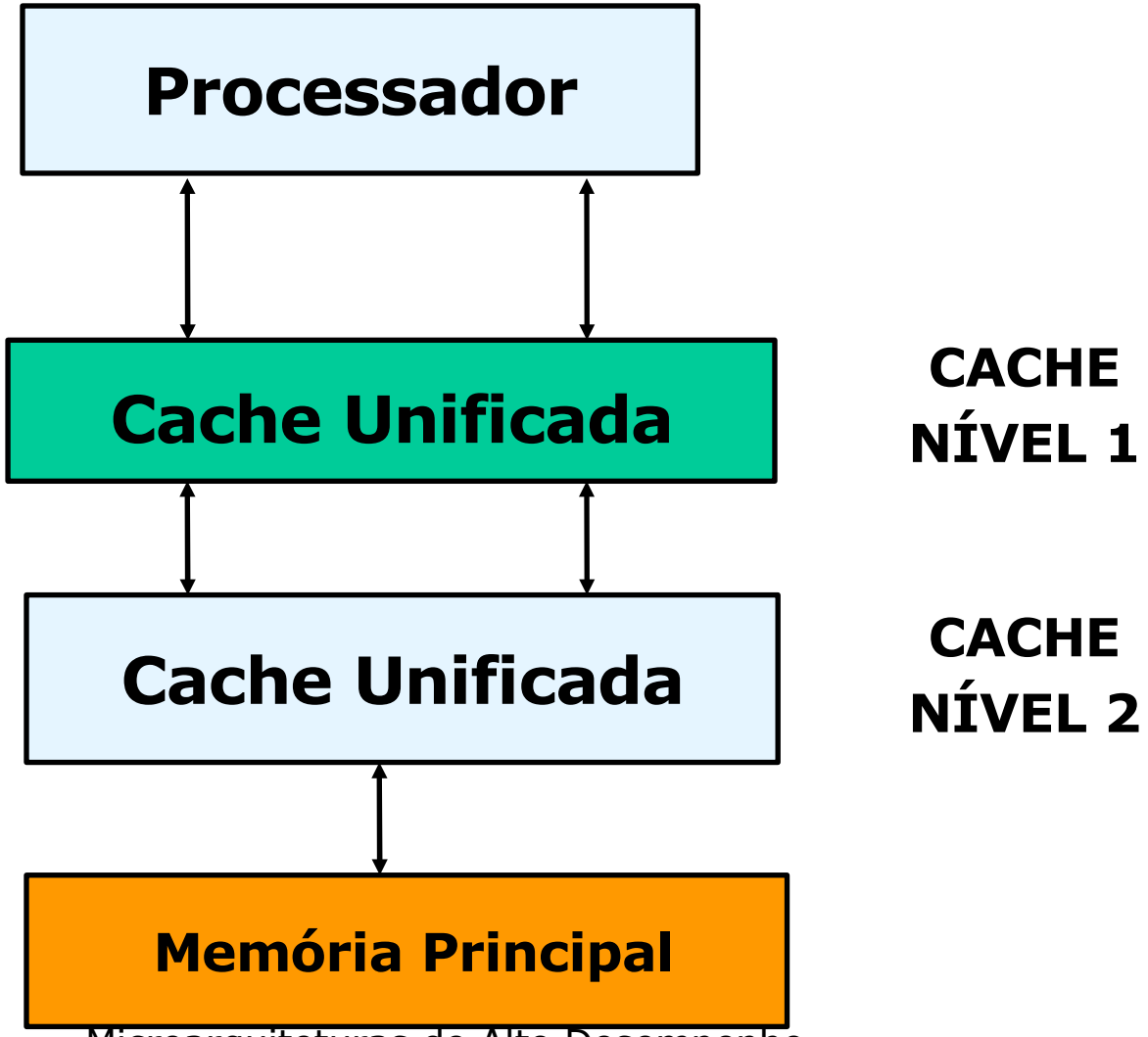
Caches de Dados e Instruções

- **Dados e instruções: cache unificada x caches separadas.**
- **Vantagens das caches separadas:**
  - **política de escrita só precisa ser aplicada à cache de dados;**
  - **caminhos separados entre memória principal e cada cache, permitindo transferências simultâneas (p.ex. quando o processador possui um *pipeline*);**
  - **Estratégias diferentes para cada cache: tamanho total, tamanho de linha, organização.**
- **Caches separadas são usadas, p.ex., no Pentium e no 68040.**

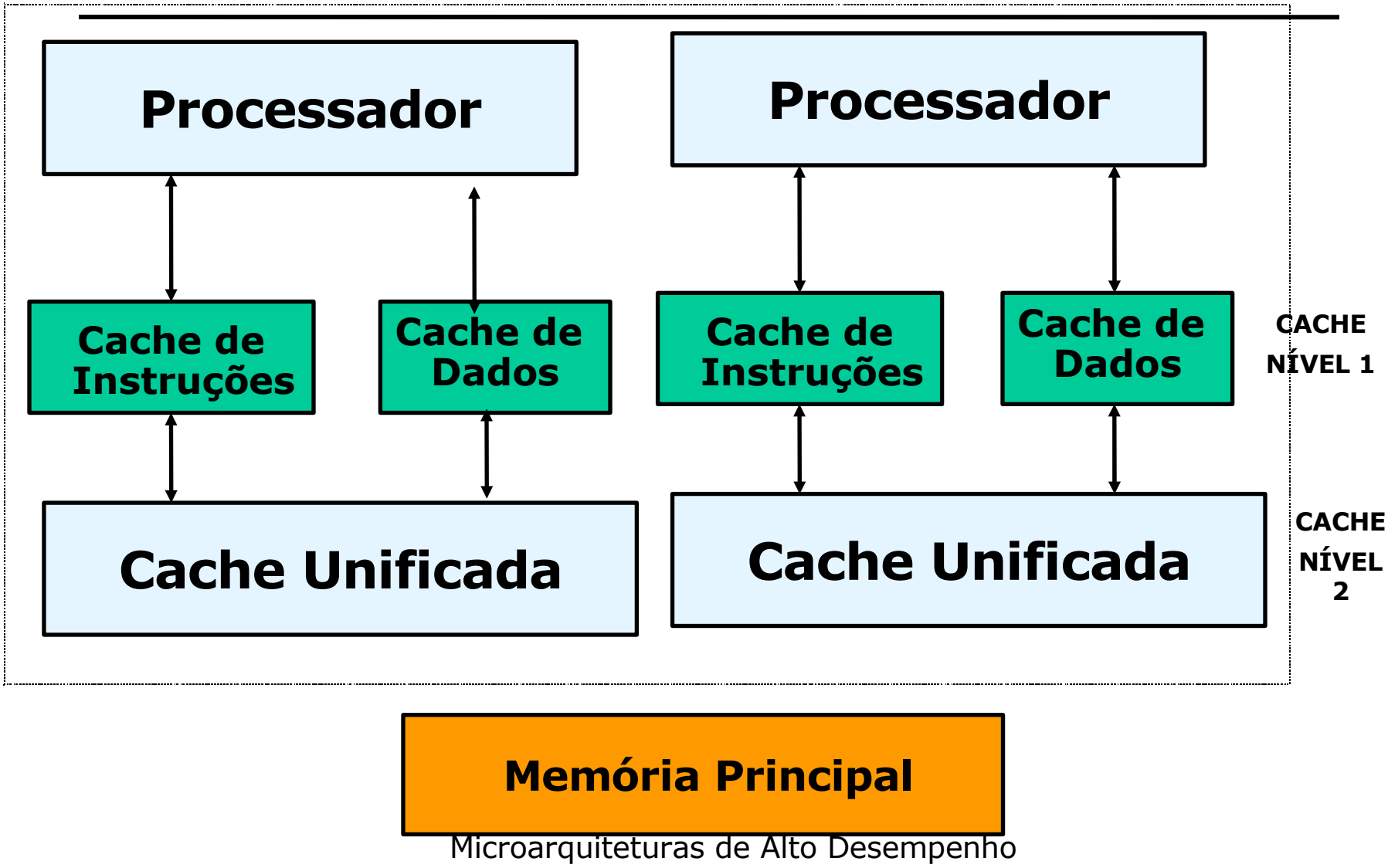
Cache Multinível



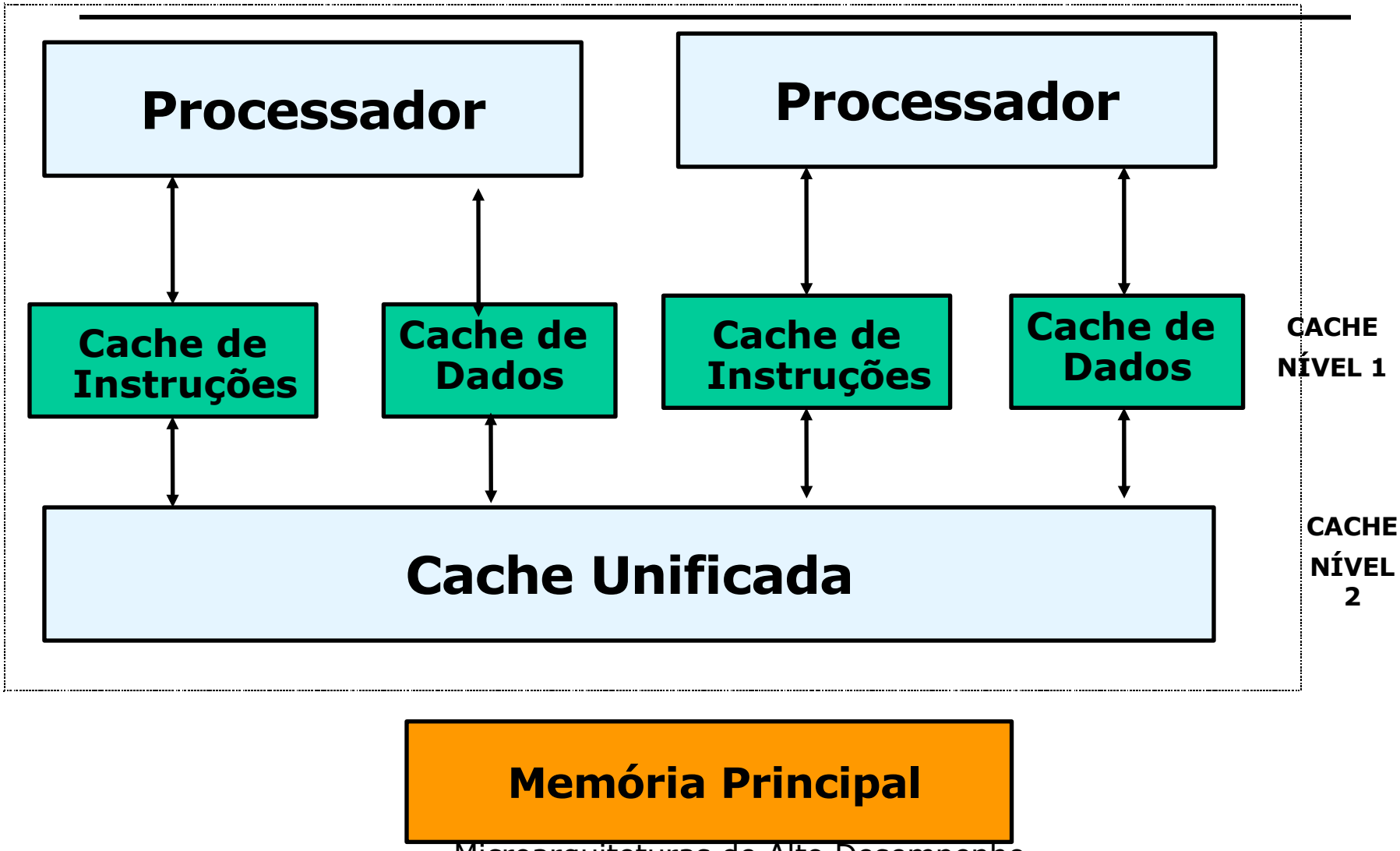
Cache Multinível



Cache Multicore



Cache Multicore



Cache Multinível

- **Manter a cache de nível 1 pequena e muito rápida, acompanhando o relógio da CPU.**
- **Utilizar um cache de nível 2 grande, mas não tão rápida, mas capaz de reduzir a penalidade das falhas.**

$$T_m = T_1^{hit} \times h_1 + (1 - h_1) \times (T_2^{hit} \times h_2 + (1 - h_2) \times T_2^{miss})$$

- **Normalmente utiliza-se a cache de nível 1 separada para dados e instruções e a cache de nível 2 unificada.**

Os Três Cs

- **Falhas Compulsórias: São faltas no acesso à cache, causadas pelo primeiro acesso que nunca esteve na cache.**
- **Falhas devido à Capacidade: São faltas que ocorrem porque a cache não pode armazenar todos os blocos necessários à execução de um programa.**
- **Falhas por Conflitos ou Colisão: São faltas que ocorrem no acesso à cache quando diversos blocos competem pelo mesmo conjunto. Não ocorrem em caches totalmente associativas.**

Comparação de Caches

PROCESSADOR	TAMANHO DA CACHE
Intel CELERON	L1 – (12 K μop + 16KB) (int.) L2 – 256 KB (int.)
Intel PENTIUM III	L1 – (16 KB + 16KB) L2 – 256 KB (int.) ou 512 KB (ext.)
Intel PENTIUM IV HT	L1 – (12 K μ op + 8KB) (int.) L2 – 512 KB (int.)
AMD DURON	L1 – (64 KB + 64 KB) (int.) L2 – 64 KB (int.) (excl.)
AMD ATHLON	L1 – (64 KB + 64 KB) (int.) L2 – 512 KB (int.)

Comparação entre os Diversos Níveis

	TLB	Cache L1	Cache L2	Mem. Virtual
Tam. Bloco (bytes)	4 – 8	4 – 32	32 – 256	4K – 16K
Tempo de Hit (Ciclos)	1	1—2	6 – 15	10 –100
Penalidade Miss (Ciclos)	10 – 30	8 – 66	30 – 200	700K – 6M
Tamanho (bytes)	32 – 8K	1 – 128 K	256K – 16M	16M – 64G
Colocação dos Blocos	Tot. Assoc./ Assoc. Conj.	Direto	Direto/ Assoc. Conj.	Totalmente Assoc.
Subst. Blocos	Aleatória	-	Aleatória	LRU
Pol. Escrita	Flush	WT ou WB	WT ou WB	WB

Exercícios

- **Utilize o simulador "sim-cache", da versão 3.0 do SimpleScalar. Escolha um programa inteiro e outro de ponto flutuante da suíte SPEC95. Faça uma variação da associatividade entre 1 e 8, e calcule a taxa de acerto para caches com capacidades de 16, 32, 64 e 128 Kbytes. Considere um tamanho de bloco com 16 bytes. Apresente um gráfico e comentários sobre os dados obtidos.**

Referências

2. **"Organização e Projeto de Computadores – A Interface Hardware/Software "** David A. Patterson e John L. Henessy**, Editora LTC, 2ed.**
3. **"Cache Memories"** Alan Jay Smith, **ACM Computing Surveys,** Volume 14 Issue 3, September 1982