Organização e Arquitetura de Computadores.

Hierarquia de Memória

Hierarquia de Memória

- Princípio de Localidade:
  - Um programa gasta 90% de seu tempo de execução com apenas 10% de seu código.
    - É a propriedade de programa mais explorada atualmente;
    - Uma aplicação:
      - Torna possível prever, com razoável precisão, instruções e dados que um programa usará no futuro próximo.
      -
  - Existem dois tipos de Localidade:
    - Localidade Temporal e Localidade Espacial

Hierarquia de Memória

- Localidade Temporal
  - Se um item é referenciado, ele tenderá a ser referenciado novamente em breve.
- Localidade Espacial
  - Se um item é referenciado, os itens cujos endereços estão próximos tenderão a ser referenciados novamente em breve.

Hierarquia de Memória

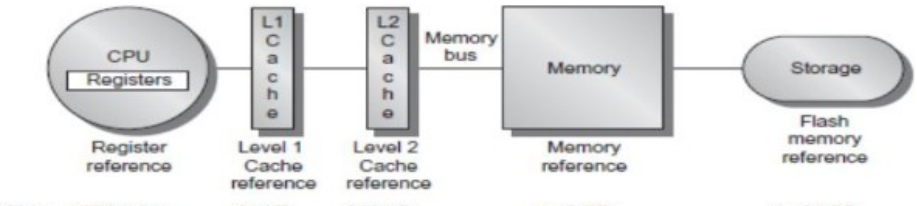
- A partir do Princípio de Localidade projetamos a hierarquia de memória tentando oferecer aos programadores uma aparente grande e rápida memória.
- Uma hierarquia de memória consiste em múltiplos níveis de memória com diferentes velocidades e tamanhos.
- As memórias mais rápidas são mais caras por bit do que as memórias mais lentas e, portanto, são menores.

Hierarquia de Memória

- Os diferentes tipos de memórias em um computador são organizados em níveis, onde aqueles mais próximos do processador estão os mais rápidos e menores e conforme a distância aumenta se tornam maiores e mais lentos.

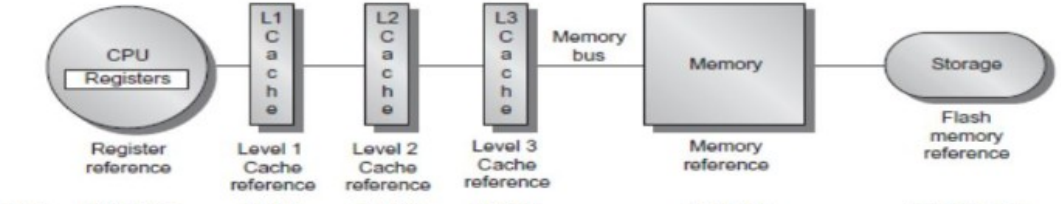
Memory Hierarchy

Mobile devices



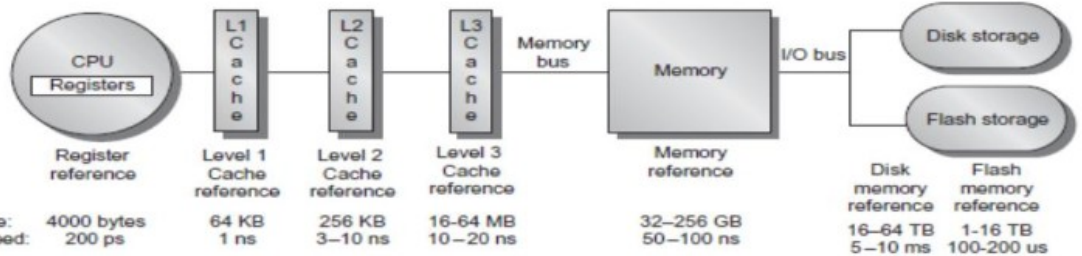
(A) Memory hierarchy for a personal mobile device

Laptops and desktops



(B) Memory hierarchy for a laptop or a desktop

Servidores



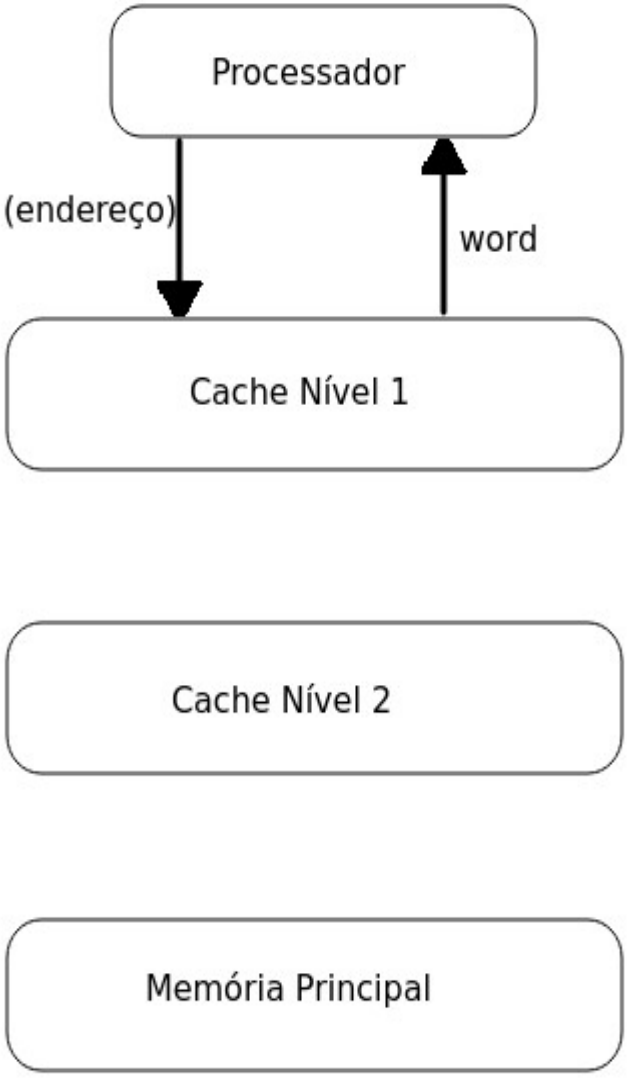
(C) Memory hierarchy for server

Hierarquia de Memória

- Memória Cache
- Cache é o nome dado ao nível mais alto ou primeiro nível de hierarquia de memória encontrada quando o endereço sai do processador.
- Quando um processador encontra um item de dado solicitado na cache, isso é chamado de **acerto de cache** – *cache hit* ;
- Quado não encontra, é chamado de **perda de cache** – *cache miss*.
- Uma coleção de dados de tamanho fixo contendo a palavra requisitada, chamada **bloco (**ou **linha)**, é apanhada da memoria principal e colocada na cache.

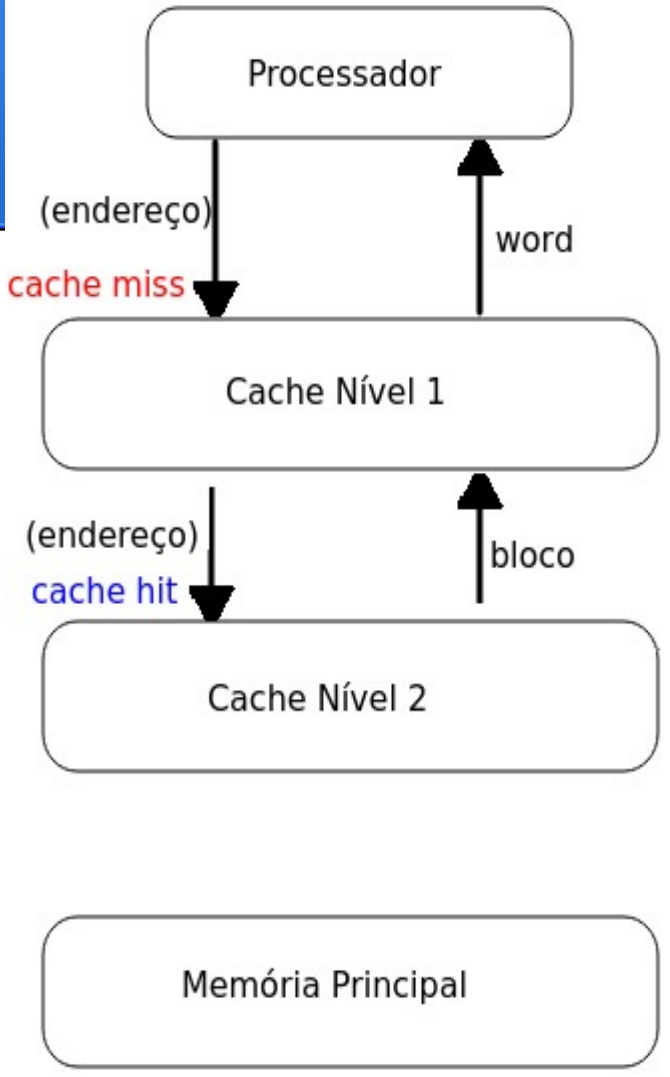
Hierarquia de Memória

- O processador faz requisição de leitura passando o endereço;
- E, se a cache L1 conter a informação desejada – cache hit, a informação (uma word) é retornada.



Hierarquia de Memória

- Se a informação não está na cache L1 ocorre um cache miss;
- E o endereço é enviado à cache L2;
- E da mesma forma ocorre caso a informação não esteja na cache L2 também.

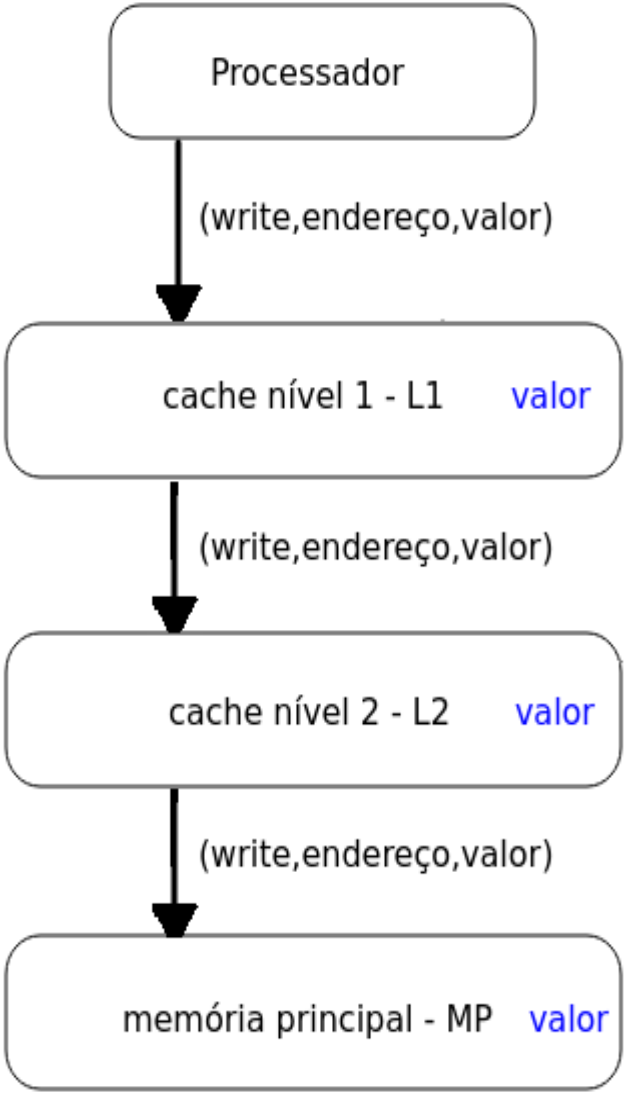


Hierarquia de Memória

- Políticas de Escrita
  - Descreve como a atualização das memórias ocorrem quando um programa faz um armazenamento. (Ambas as estratégias usam um buffer de gravação para tornar as gravações assíncronas).
- WRITE-THROUGH
  - Atualiza imediatamente os níveis inferiores de hierarquia.
  - Mais seguro, mais simples, menos eficiente.
  - Ex: Escrita em L1 provoca escrita em L2 e MP.
- WRITE-BACK
  - Atualiza apenas os níveis inferiores da hierarquia quando um bloco atualizado for substituído.
  - O mesmo bloco pode ter dados diferentes em níveis de cache diferentes (isso é um problema?).
  - Mais eficiente, mas requer controle extra (custo).

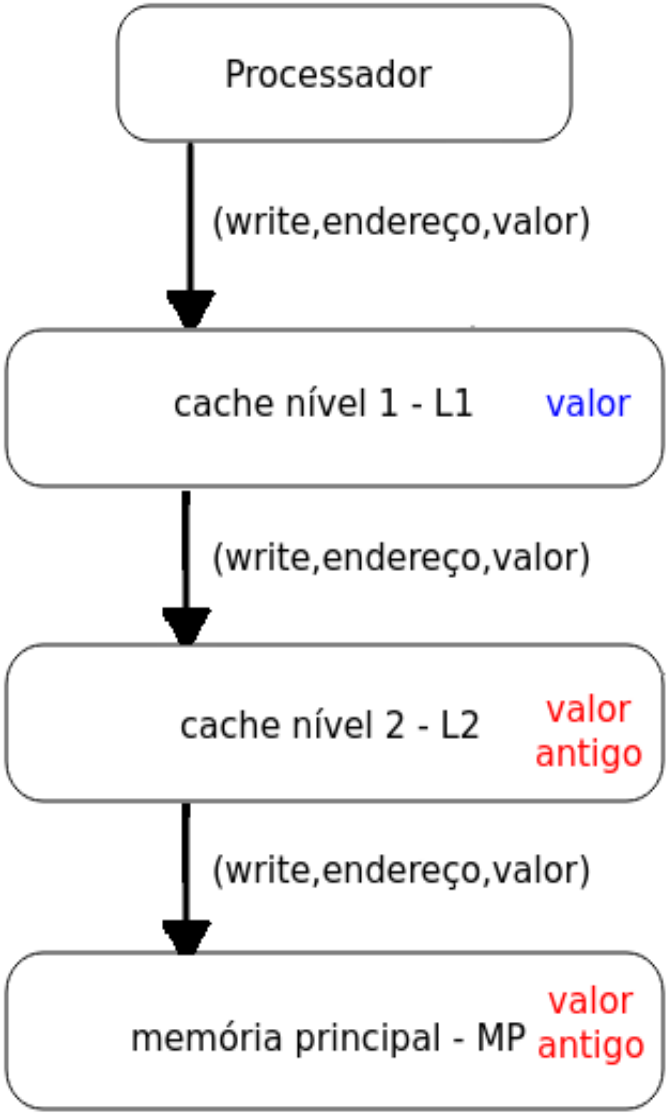
Hierarquia de Memória

- Exemplo – Write-through
  - O processador solicita a escrita de um valor em um determinado endereço,
  - Ocorrendo um cache hit, em um determinado bloco B,
  - Imediatamente ocorre a escrita
    - No bloco B da L1,
    - No bloco B da L2,
    - E no bloco B da MP.



Hierarquia de Memória

- Exemplo – Write-back.
  - O processador solicita a escrita de um valor em um determinado endereço,
  - Ocorrendo um cache hit, em um determinado bloco B,
  - Imediatamente ocorre a escrita no bloco B da L1, e um bit (dirty-bit) é marcado informando a alteração.
  - Porém, os outros níveis, L2 e MP, ficam desatualizados até que uma substituição na L1 ocorra.



Hierarquia de Memória

- Cache Associativa
  - Uma decisão importante do projeto é onde os blocos (ou linhas) podem ser colocados.
  - O esquema mais popular é o conjunto associativo, onde :
    - Um conjunto é um grupo de blocos no cache.
    - Um bloco é primeiro mapeado em um conjunto e então o bloco pode ser colocado em qualquer lugar desse conjunto.

Hierarquia de Memória

- Encontrar um bloco consiste primeiro em mapear o endereço do bloco para o conjunto e depois pesquisar o conjunto – geralmente em paralelo – para encontrar os blocos. O conjunto é escolhido pelo endereço dos dados:

(Endereço do bloco) MOD (Número de conjuntos em cache)

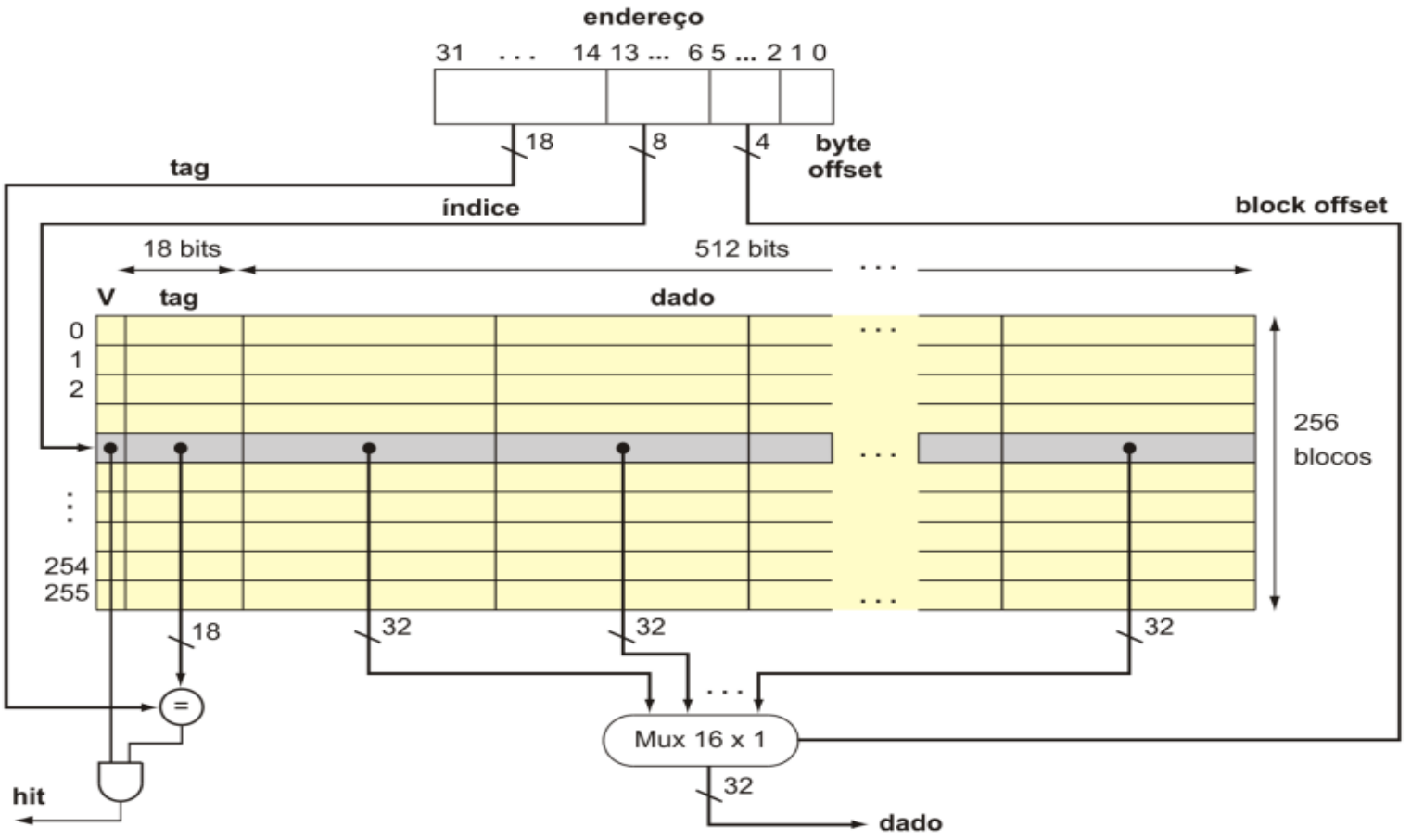
Hierarquia de Memória

- Se houver n blocos em um conjunto, o posicionamento do cache é chamado de **conjunto associativo de n vias**.
- Um **cache mapeado diretamente** possui apenas um bloco por conjunto (portanto, um bloco é sempre colocado no mesmo local),
- e um **cache totalmente associativo** possui apenas um conjunto (portanto, um bloco pode ser colocado em qualquer lugar).

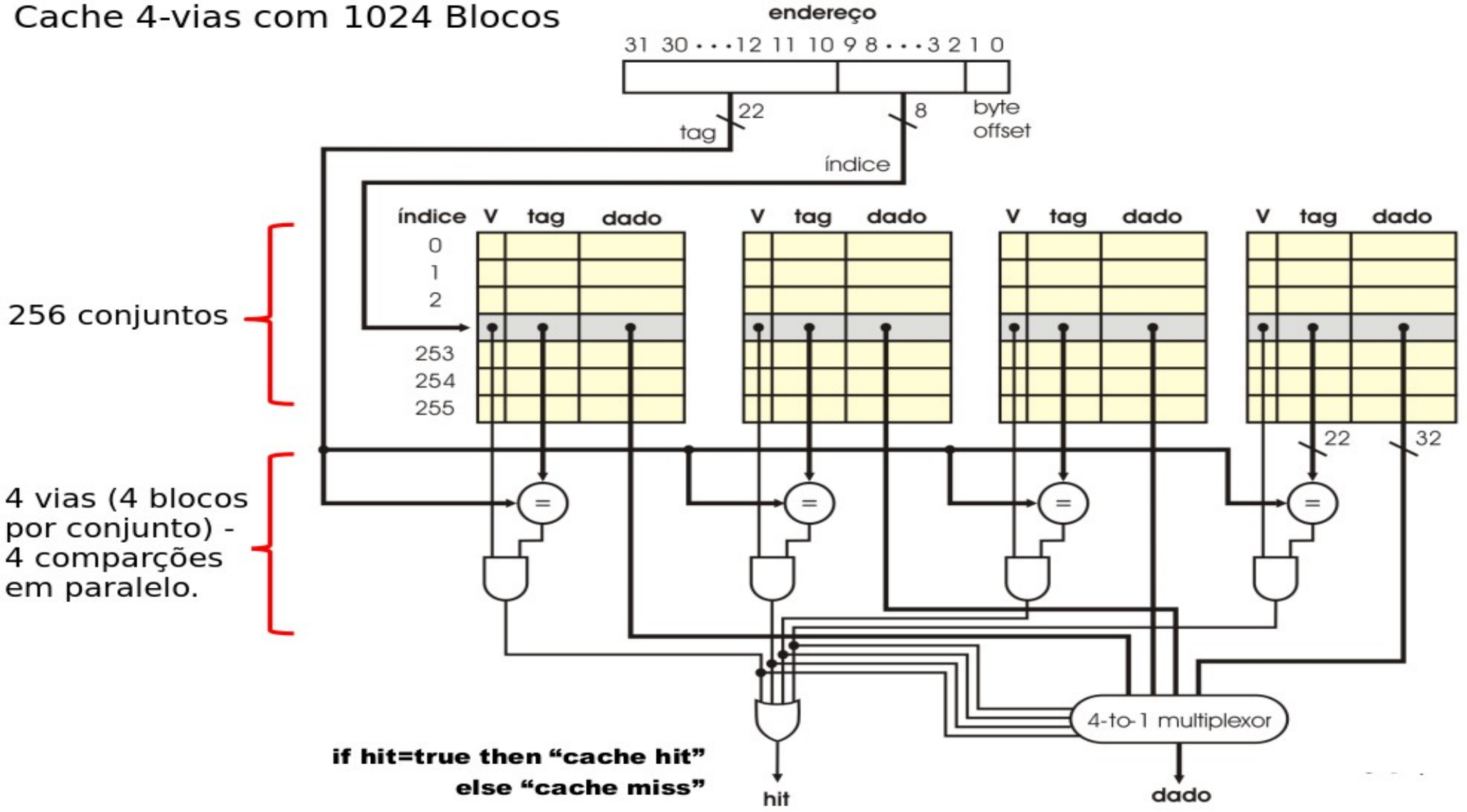
Hierarquia de Memória

Cache Diretamente Mapeada

- 256 blocos
- 16 words por bloco

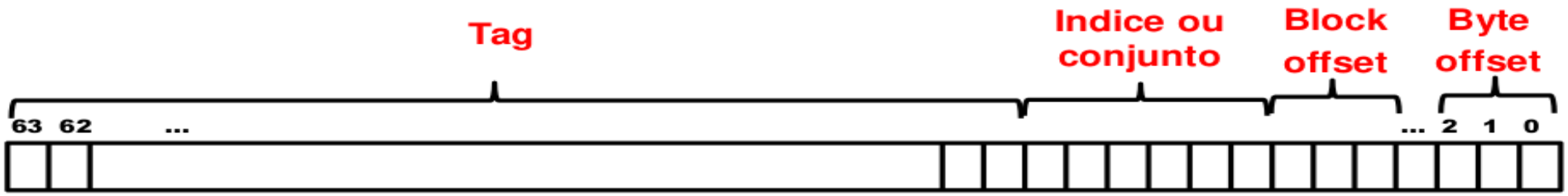


Cache 4-vias com 1024 Blocos



Cache Associativa de N-vias

Como podemos encontrar um dado na cache ?



Suponha uma cache associativa n-vias com:
- m blocos
- w palavras(words) por bloco
- b bytes por paralavra.

O endereço então precisa ter:

$\log_2 m/n$ bits - para o índice/conjunto

$\log_2 w$ bits - para o bloco (offset)

$\log_2 b$ bits - para o byte (offset)

$8*b - (\log_2 m/n + \log_2 w + \log_2 b)$ bits - tag

Exemplo:

cache 8-vias com
1024 blocos com
4 palavras por bloco de
8 bytes

índice = $\log_2(1024/8)$
= $\log_2 128 = 7$ bits

bloco = $\log_2 4 = 2$ bits
bytes = $\log_2 8 = 3$ bits
tag=8*8-(7+2+3) = 52

Hierarquia de Memória

- Exercícios

  1) Projete um cache associativo de conjunto com um total de 512 blocos, blocos com 4 palavras e tamanho de palavra de 64 bits.

  2) Suponha que um processador solicite dados de um endereço de 64 bits. Como podemos usar esse endereço para colocar ou encontrar essa palavra de dados em um cache associativo de conjunto com 512 blocos, 4 palavras por bloco?

Desempenho do sistema de memória

- **Taxa de acertos** (hit_{rate}): percentual de acerto por acesso à memória.
- **Taxa de falhas** ($miss_{rate}$): é simplesmente a fração de acessos ao cache que resultam em falhas**.**
- **Tempo de acerto** (hit_{time}): é o tempo para acessar e decidir se ocorreu um *hit* ou um *miss*.
- **Penalidade do Erro** ($miss_{pen}$): é o tempo para tratar o *cache miss*, incluindo o tempo para substituir o bloco.

Taxa de Falhas – miss rate.

- Conceitualmente podem ser classificadas em falhas :
  - ■ Compulsória — O primeiro acesso a um bloco não pode estar no cache, portanto o bloco deve ser trazido para o cache. Faltas compulsórias são aquelas que ocorrem mesmo se você tiver um cache de tamanho infinito.
  - ■ de Capacidade — Se o cache não puder conter todos os blocos necessários durante a execução de um programa, ocorrerão perdas de capacidade (além das falhas obrigatórias) devido aos blocos serem descartados e posteriormente recuperados.
  - ■ de Conflito — Se a estratégia de colocação de bloco não for totalmente associativa, ocorrerão faltas de conflito (além das faltas compulsórias e de capacidade) porque um bloco pode ser descartado e posteriormente recuperado se vários blocos forem mapeados para seu conjunto e os acessos aos diferentes blocos estiverem misturados .

Erros por instrução

- No entanto, a taxa de erros pode ser uma medida enganosa por vários motivos. Portanto, alguns projetistas preferem medir erros por instrução em vez de erros por referência de memória (taxa de erros). Esses dois estão relacionados:

  Erros/Instruções = (taxa de erros x Nº Acessos)/Instruções

  = taxa de erro x (Acessos/Instruções)

Tempo médio de acesso à memória

Cache com um nível: $t_{men} = hit_{time} + (miss_{rate} * miss_{pen})$

Cache com dois niveis:

$t_{men} = hit_{timeL1} + (miss_{rateL1} * miss_{penL1})$

$t_{men} = hit_{timeL1} + (miss_{rateL1} * (miss_{localrateL2} * miss_{penL2}))$

$t_{men} = hit_{timeL1} + miss_{rateL1} * hit_{timeL2} + miss_{globalrateL2} * miss_{penL2}$

Exercício

3) As seguintes características foram observadas através do benchmarking de um sistema contendo dois níveis de cache (L1 e L2):

- 100 milhões de solicitações de memória (ou referências)
- 0,8 acesso à memória por instrução em média
- 15 milhões de erros L1 e 2 milhões de erros L2
- O tempo de acerto L1 é 2ns e o tempo de acerto L2 é 4ns
- L2 - A penalidade de erro é de 100ns;

(a) $miss_{rateL1}$ = ?

(b) $miss_{globalL2}$ = ? $miss_{localrateL2}$ = ?

(c) erros por instrução em L1 e L2 ?

(d) tempo médio de acesso à memória ?

Estimativas de tempo de cpu

- CPU_{time} : tempo estimado para executar um conjunto de instruções (não considera E/S e nenhuma outra exceção de interferência externa).
  - Em um sistema com cache ideal (sem falhas de cache)

    $t_{cpu} = ciclos_{cpu} * t_{ciclo}$

  - Em um sistema com caches reais (considerando paralisações para lidar com falhas de cache)

    $t_{cpu} = (ciclos_{cpu} + ciclos_{stall}) * t_{ciclo}$

Estimativas de tempo de cpu

- Média de Ciclos por Instrução – CPI
  - Corresponde à média de ciclos para executar um conjunto de instruções.
  - Estimativas de tempo de CPU comumente usadas

$$t_{cpu} = instr_{exec} * (CPI + ciclos_{stall} / instr) * t_{ciclo}$$

$$ciclos_{stall} = instr_{exec} * miss_{instr\_rate} * miss_{pen\_ciclos}$$

$$ciclos_{stall} = instr_{exec} * mem_{acess\_instr} * miss_{rate} * miss_{pen\_ciclos}$$

Exercícios

4) Suponha que 100 mil instruções foram executadas por um processador de 500Mhz, com 3 ciclos de CPI médio. Foram 60 mil ciclos de tratamento de stall com cache miss. Como podemos estimar o tempo de CPU?

Exercícios

5) Obtivemos algumas estatísticas através da execução de 1 milhão de instruções em uma CPU na frequência de 1Ghz: em média, foram necessários 2 ciclos para executar cada instrução do programa, ignorando os travamentos de memória; Havia um único cache com tempo de acerto de 3 ciclos e penalidade de falta de 60 ciclos; Em média, ocorreu uma falha a cada 20 instruções; Para cada 100 instruções, 25 acessam a memória. Responda as questões abaixo:

(a) taxa de falhas por instrução ?

(b) taxa de erros por acesso à memória ?

(c) tempo médio de acesso à memória ?

(d) tempo de cpu ?

Otimizações Básicas de Cache

- Trabalho

Tecnologia de Memória

- Trabalho