SSC0611
Arquitetura de Computadores

5ª e 6ª Aulas – Revisão de Hierarquia de Memória

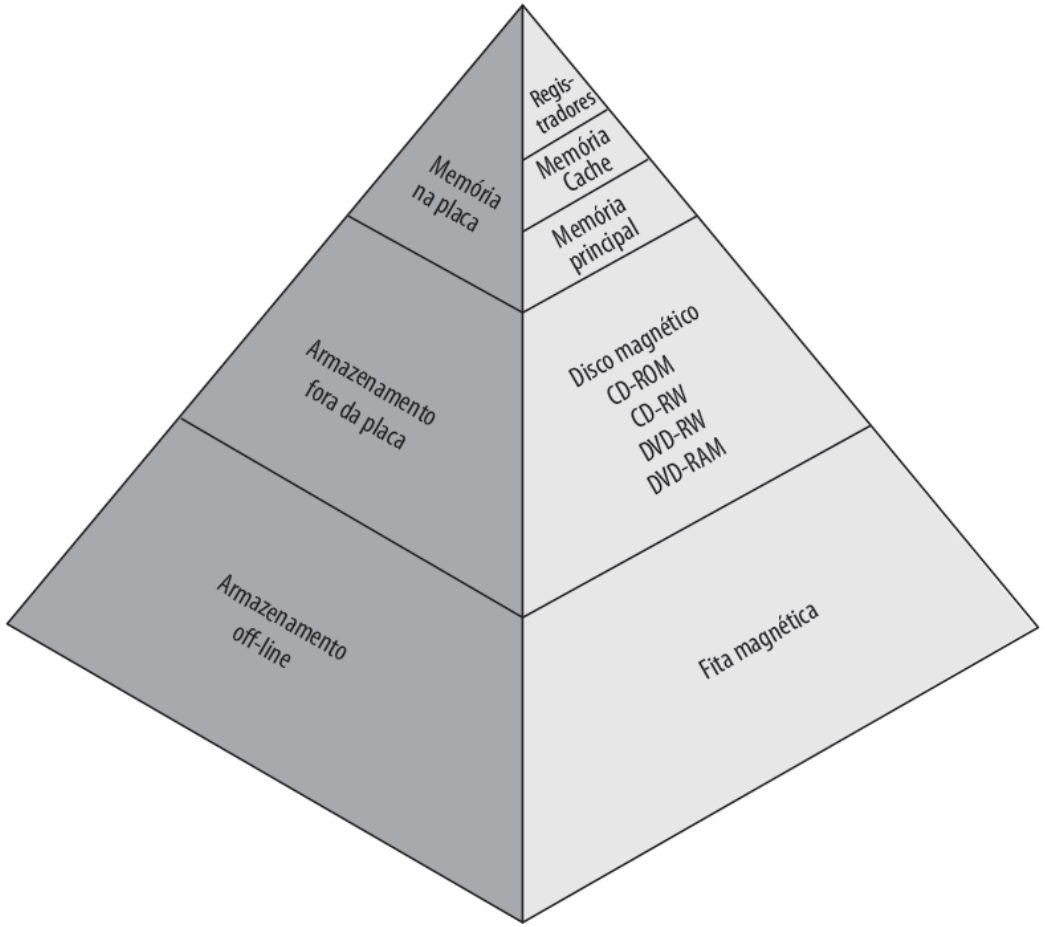
Profa. Sarita Mazzini Bruschi
sarita@icmc.usp.br

Memória

- Memória
  - Todo componente capaz de armazenar bits de informação
- Características conflitantes
  - Grande capacidade
  - Rápida acesso
  - Custo baixo
- Solução:
  - Hierarquia de Memória

Hierarquia de Memória

Figura 4.1 A hierarquia de memória



Fonte: Stallings

Hierarquia de Memória

Tabela 4.1 Principais características dos sistemas de memória do computador

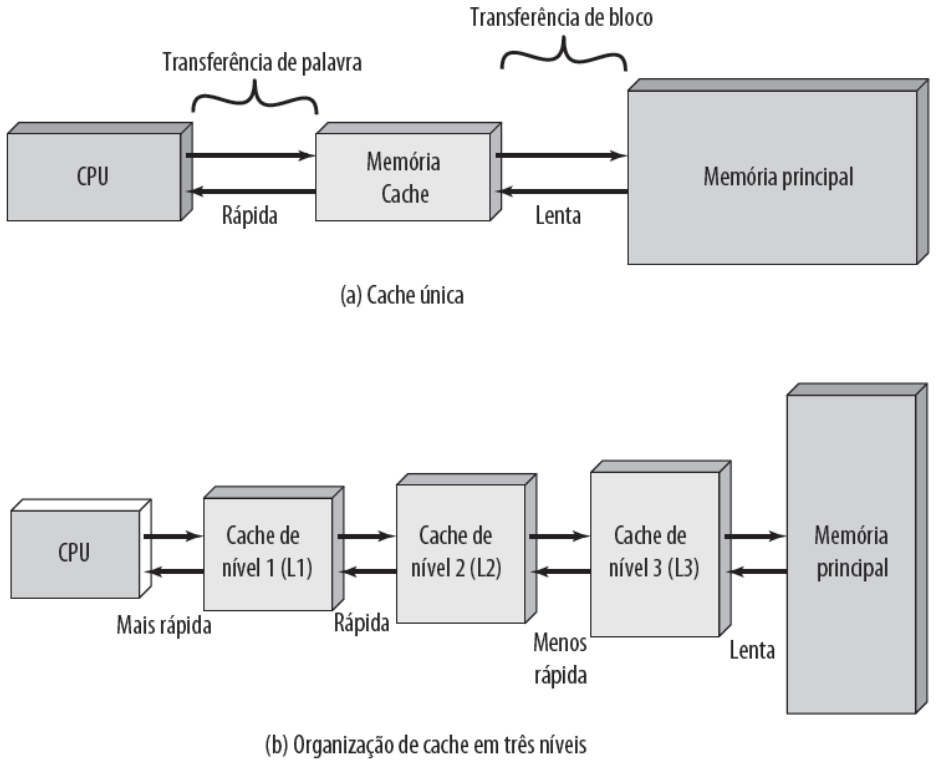
Localização	**Desempenho**
Interna (por exemplo, registradores do processador, memória principal, cache)	Tempo de acesso
	Tempo de ciclo
Externa (por exemplo, discos ópticos, discos magnéticos, fitas)	Taxa de transferência
Capacidade	**Tipo físico**
Número de palavras	Semicondutor
Número de bytes	Magnético
Unidade de transferência	Óptico
Palavra	Magneto-óptico
Bloco	**Características físicas**
Método de acesso	Volátil/não volátil
Sequencial	Apagável/não apagável
Direto	**Organização**
Aleatório	Módulos de memória
Associativo	

Memória Cache

- Método de Acesso: Associativo
  - Localização de dados na memória pelo conteúdo e não pelo endereço
- Hierarquia do acesso à memória principal
  - Um nível mais próximo do processador é normalmente um subconjunto de qualquer nível mais distante
  - Todos os dados são armazenados no nível mais baixo

Memória Cache

Figura 4.3 Cache e memória principal



Fonte: Stallings

Memória Cache

- Princípio da Localidade Temporal e Espacial
  - **Temporal**: se um dado na memória foi utilizado em um instante, existe a probabilidade dele ser utilizado novamente num instante futuro
  - **Espacial**: se um dado na memória foi utilizado, existe a probabilidade das posições que estão perto serem utilizadas também

Memória Cache

- Como manter a cache?
  - Existem políticas que determinam quais elementos devem ser mantidos na cache e que controlam a consistência da informação
- Caches bem implementados atingem taxas de acerto (*cache hit*) superior a 95%
- Além de ter políticas que determinam quais elementos devem ser mantidos na cache, as memórias devem também ser estruturadas

Memória Cache

- **Bloco**: menor unidade que pode estar presente ou ausente na hierarquia de dois níveis
- **Acerto**: a informação está em algum bloco do nível superior (mais perto do processador)
- **Falha**: a informação não está em algum bloco do nível superior (mais perto do processador)

Princípios básicos

- Definição:
  - Tamanho da cache
  - Tamanho do bloco
  - Função de mapeamento
  - Algoritmo de substituição
  - Política de escrita
  - Número de caches

Tamanho da cache

- Compromisso entre quantidade de dados armazenados, custo e desempenho
  - Tamanho suficientemente pequeno – custo/complexidade
  - Tamanho suficientemente grande – taxa de acerto alta - desempenho
- Depende da localização da cache
- Quanto menor, mais rápida
  - Maior -> maior número de portas envolvidas no endereçamento -> mais lenta

Tamanho da cache

Processador / ano	L1	L2	L3
Pentium / 1993	16 KB	-	-
Pentium Pró / 1995	16 KB	256 / 512 / 1024 KB	-
Celeron / 1998	8 – 64 KB por núcleo	0 – 1 MB	0 – 2 MB
Xeon / 1998	8 – 64 KB por núcleo	256 KB – 12 MB	4 – 16 MB
Atom / 2008	56 KB por núcleo	512 KB / 1 MB	-
Core i3 / 2010	64 KB por núcleo	256 KB	3 – 4 MB
Core i5 / 2009	64 KB por núcleo	256 KB	6 – 10 MB
Core i7 / 2011	64 KB por núcleo	256 KB por núcleo	12 – 20 MB

Tamanho do bloco

- Princípio de localidade:
  - Qto maior o bloco $\Rightarrow$ maior taxa de acerto
- Bloco grande é bom até que a probabilidade de utilizar os dados buscados recentemente se torne menor do que a probabilidade de reutilizar os dados que foram substituídos

Função de mapeamento

- Número de blocos na memória cache é menor do que o número de blocos na memória principal
- Portanto, é necessário:
  - algoritmos para mapear blocos de memória principal em blocos da memória cache
  - mecanismo para determinar o bloco da memória principal que ocupa certo bloco da memória cache

Função de mapeamento

- Tipos de mapeamento:
  - Mapeamento direto
  - Mapeamento associativo
  - Mapeamento associativo por conjunto

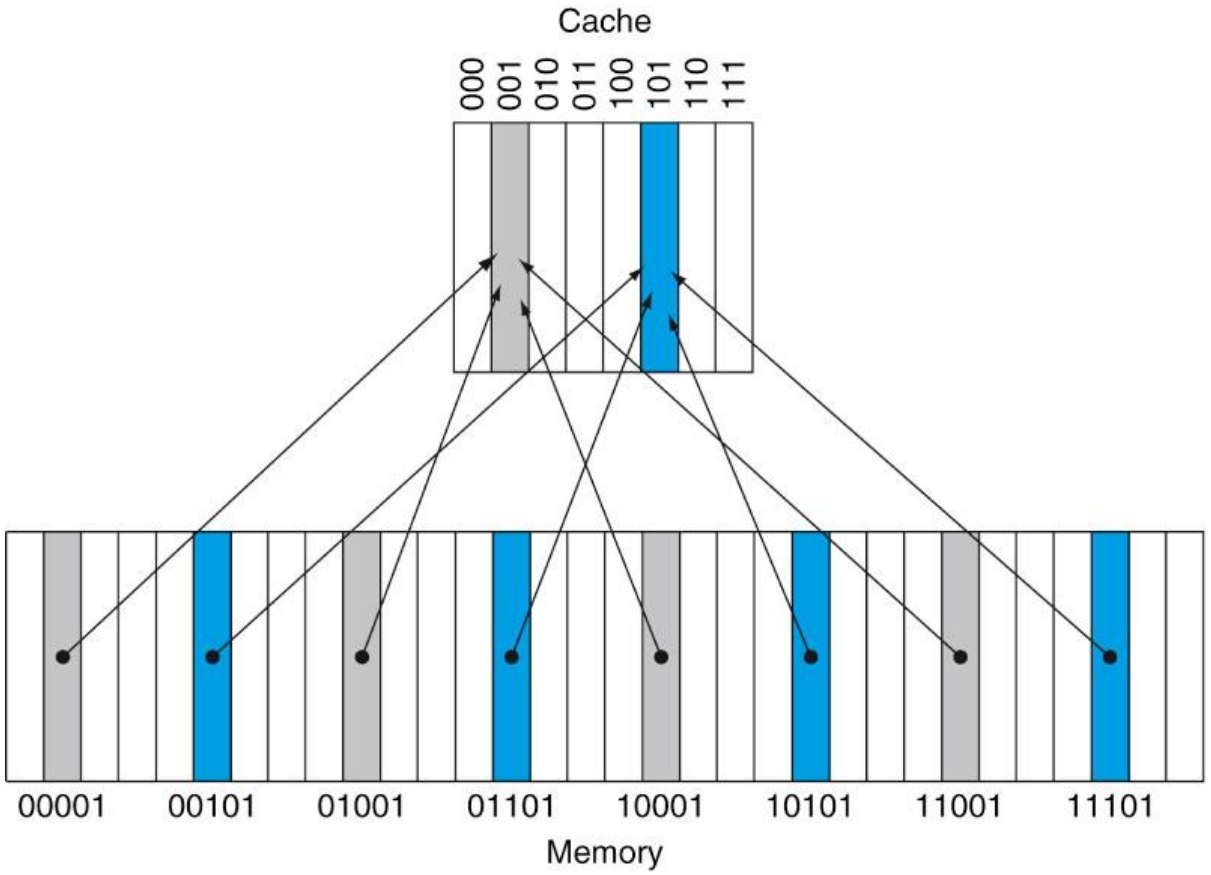
Mapeamento Direto

- Cada bloco na memória principal é mapeado em um único bloco da cache
- Expresso pela equação:

$$i = j \text{ módulo } m$$

***i* = número do bloco da memória cache**
***j* = número do bloco da memória principal**
***m* = número de blocos na memória cache**

Mapeamento Direto



Fonte: Patterson & Henessy – 5ª edição

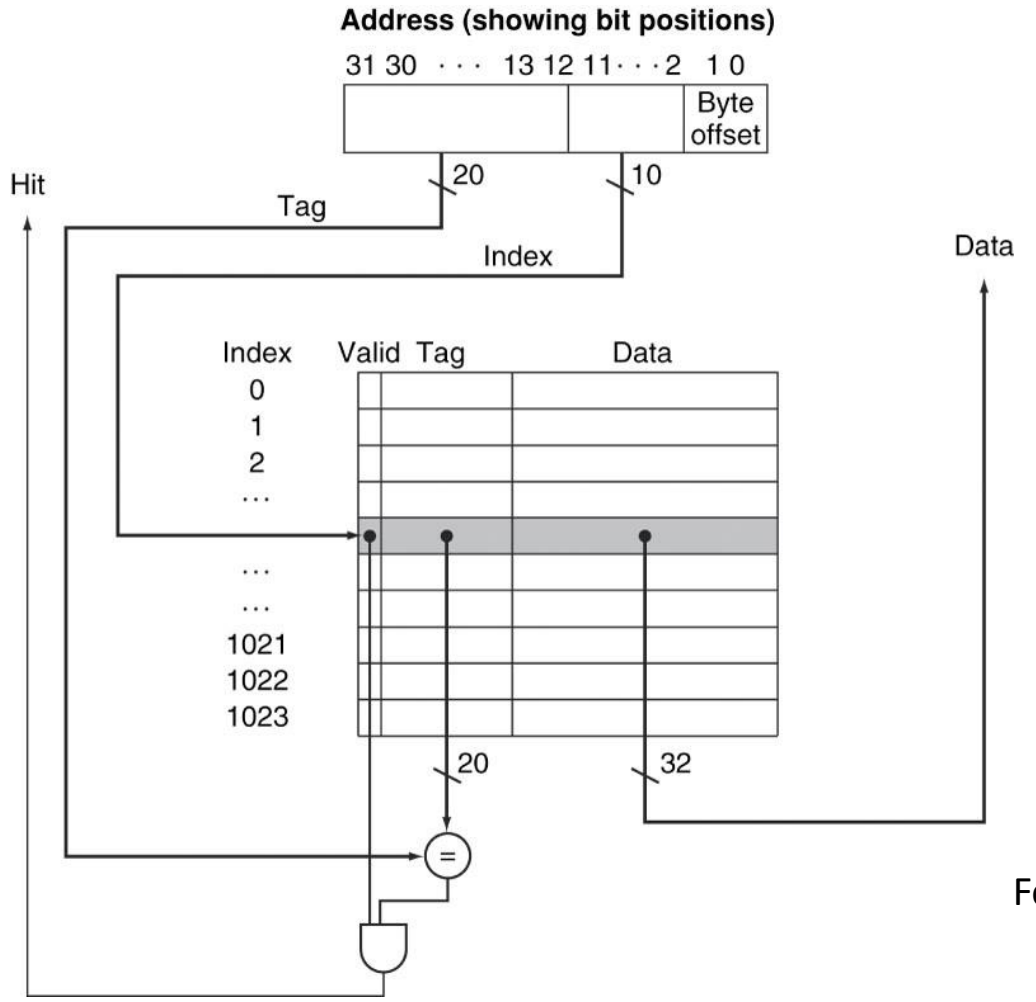
Mapeamento Direto

- Divisão do endereço em campos:
  - Byte offset: para arquiteturas endereçadas a byte
  - Word offset: quando um bloco possui mais do que uma palavra
  - Index: resultado da operação módulo
  - Tag: restante do endereço
- Como saber se a entrada na cache contém um endereço válido ou não?
  - Incluir um *bit de validade* (V)

Bits de Controle

- As caches possuem alguns bits de controle para cada bloco:
  - **V** (validade): indica se a entrada da cache é válida ou não
  - **M** (modificação): indica se os dados que estão armazenados naquele bloco da cache foram modificados ou não

Mapeamento Direto



Cache com:
- 1024 Blocos
- 1 Palavra por Bloco
- Palavras de 4 Bytes

Fonte: Patterson & Henessy – 5ª edição

Mapeamento Direto

- Vantagens
  - Técnica simples
  - Baixo custo de implementação
- Desvantagem
  - Cada bloco é mapeado em uma **posição fixa** na memória cache
    - se houver referência a blocos distintos mapeadas na mesma linha $\Rightarrow$ blocos trocados continuamente, taxa de acertos baixa

Função de mapeamento

- Tipos de mapeamento:
  - Mapeamento direto
  - Mapeamento associativo
  - Mapeamento associativo por conjunto

Mapeamento Associativo

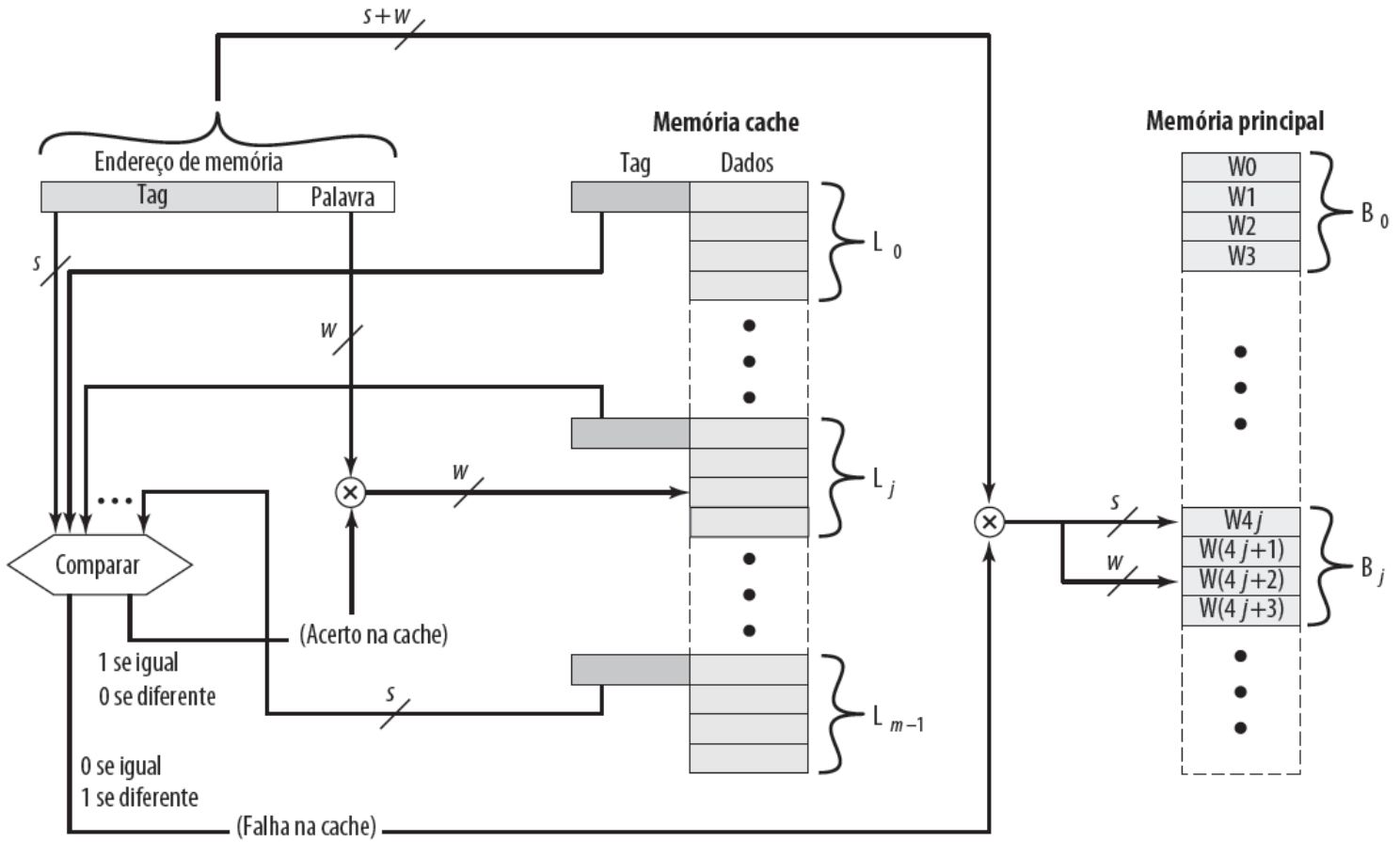
- Evita as desvantagens do mapeamento direto
- Um bloco da memória principal pode ser carregado em qualquer bloco da memória cache
- Divisão do endereço em campos:
  - Byte offset: para arquiteturas endereçadas a byte
  - Word offset: quando um bloco possui mais do que uma palavra
  - Tag ou Rótulo: restante do endereço

Mapeamento Associativo

- Rótulo: identifica um bloco da memória principal de modo unívoco
- Para determinar se um bloco está na memória cache:
  - Lógica de controle da memória cache compara simultaneamente o campo de rótulo do endereço do bloco acessado com o rótulo de todos os blocos da cache

Mapeamento Associativo
Organização

Figura 4.11 Organização da memória cache totalmente associativa



Mapeamento Associativo

- Vantagem:
  - Maior flexibilidade para a escolha do bloco a ser substituído quando um novo bloco é trazido para a memória cache
- Desvantagem:
  - Complexidade do conjunto de circuitos necessários para a comparação dos rótulos de todos os blocos da memória cache em paralelo

Função de mapeamento

- Tipos de mapeamento:
  - Mapeamento direto
  - Mapeamento associativo
  - Mapeamento associativo por conjunto

Mapeamento Associativo por Conjunto

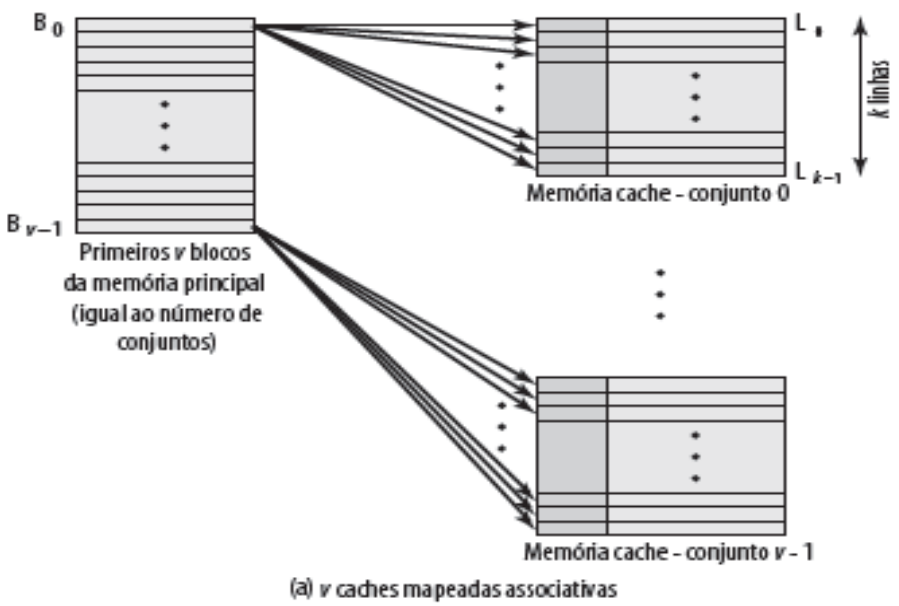
- Combina vantagens do mapeamento direto e do mapeamento associativo, diminuindo suas desvantagens
- Memória cache dividida em v conjuntos de k blocos cada

Mapeamento Associativo por Conjunto

- Divisão do endereço em campos:
  - Byte offset: para arquiteturas endereçadas a byte
  - Word offset: quando um bloco possui mais do que uma palavra
  - Set ou Conjunto: para identificar o conjunto
  - Tag ou Rótulo: restante do endereço

Mapeamento Associativo por Conjunto

- Memória cache dividida em v conjuntos de k blocos cada:



(a) *v* caches mapeadas associativas

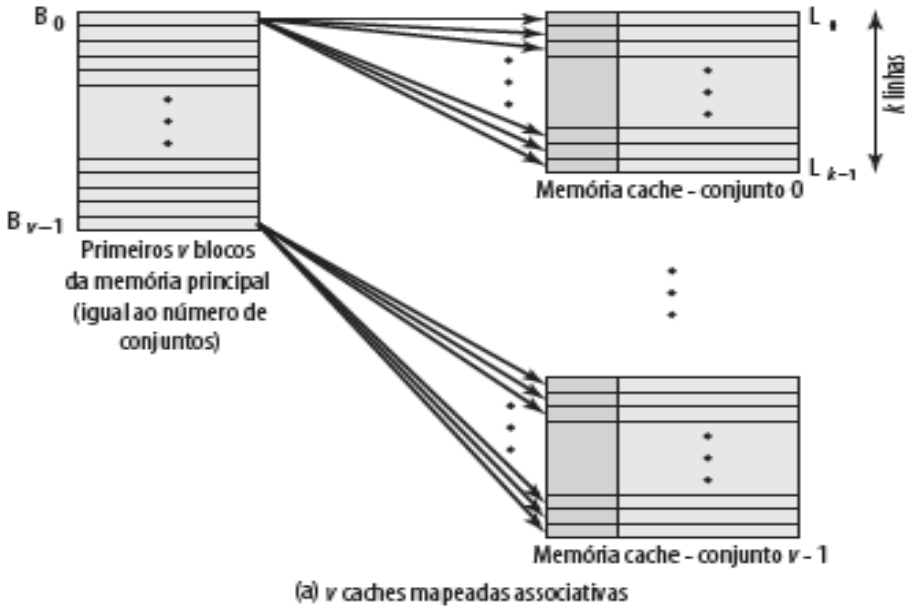
m* = *v* x *k
i* = *j* módulo *v

mapeamento associativo por conjuntos de *k* blocos
Conhecido como *k-way set*

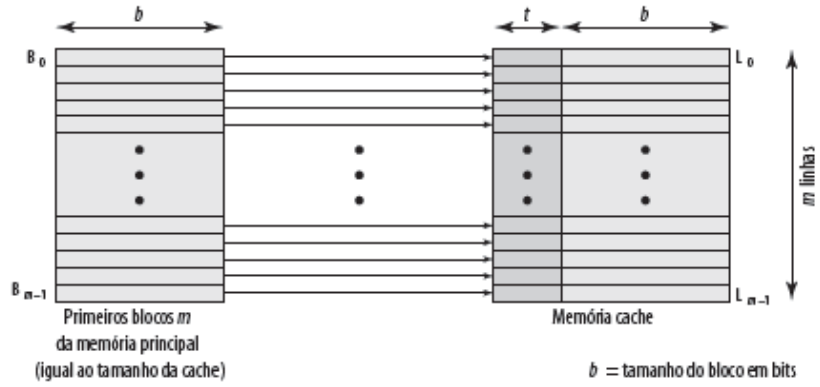
***i* = número do bloco da memória cache**
***j* = número do bloco da memória principal**
***m* = número de blocos na memória cache**

Mapeamento Associativo por Conjunto

- Se v = m (conjuntos) e k = 1 (blocos)
  - Reduz-se para o mapeamento direto



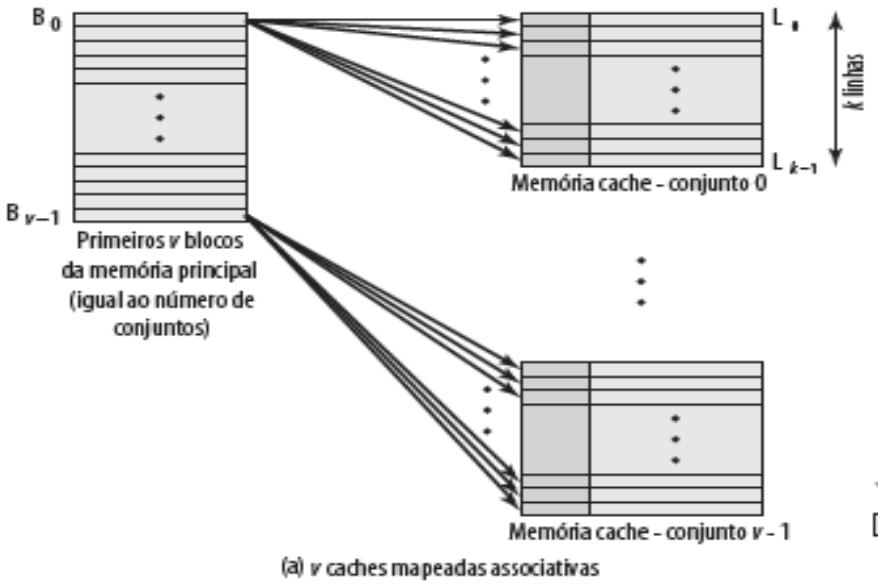
(a) v caches mapeadas associativas



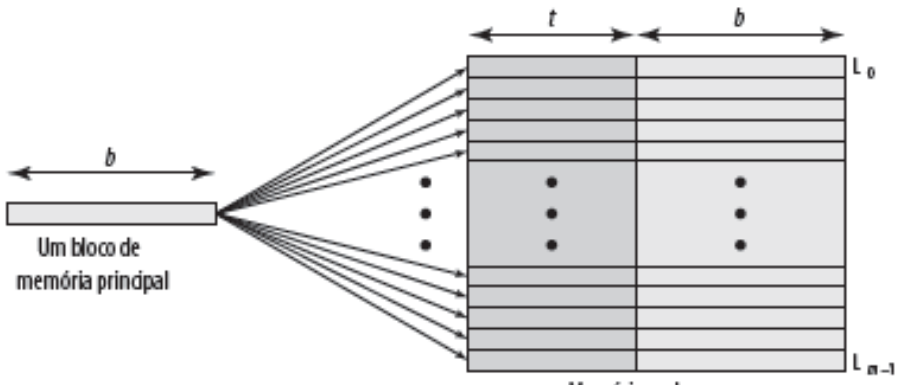
(a) Mapeamento direto

Mapeamento Associativo por Conjunto

- Se v = 1 (conjunto) e k = m (blocos)
  - Reduz-se para o mapeamento associativo total



(a) v caches mapeadas associativas



(b) Mapeamento associativo

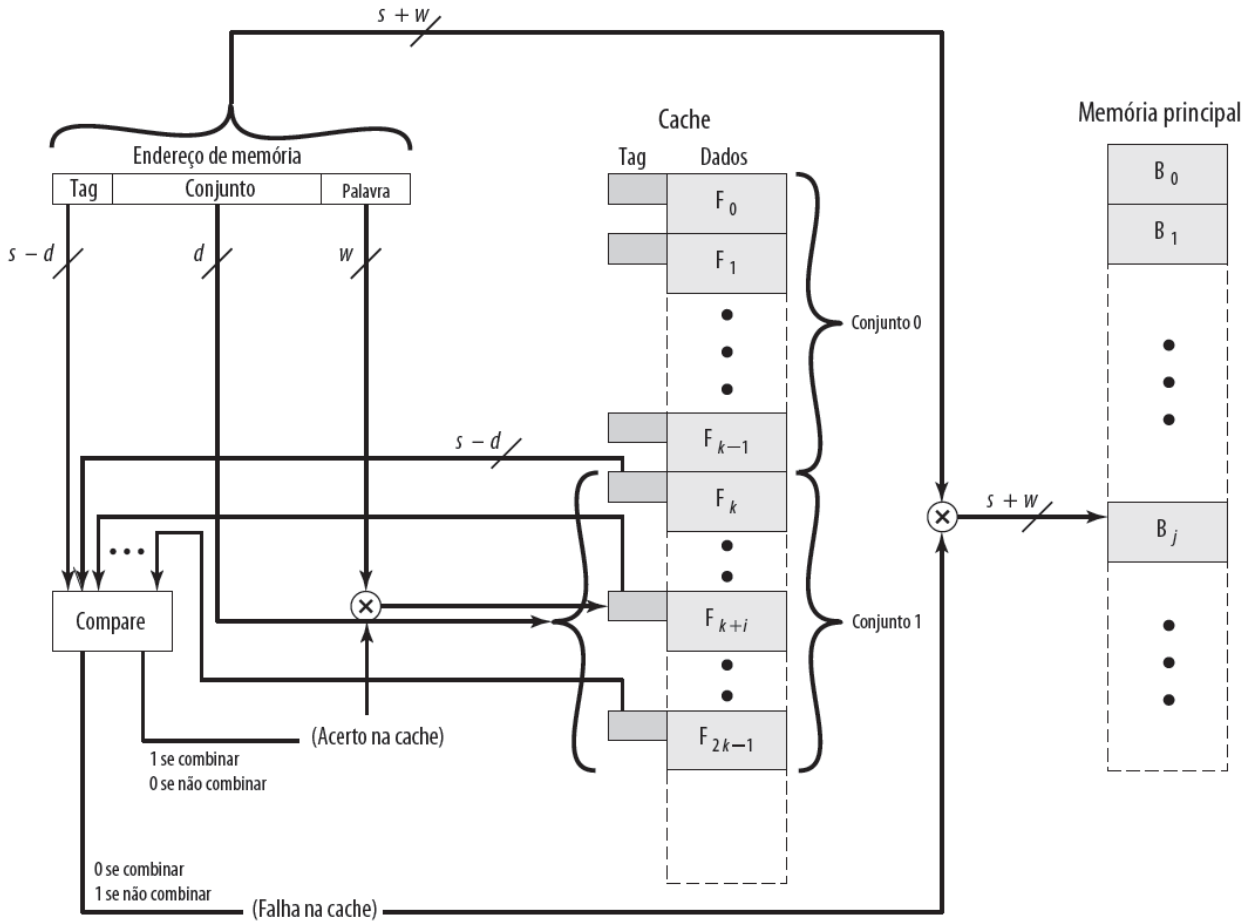
Mapeamento Associativo por Conjunto

- Na memória cache, endereço interpretado como constituído dos campos: *rótulo* + *conjunto* + *palavra*
- Mapeamento totalmente associativo:
  - Rótulo muito grande e é comparado ao rótulo de cada bloco de memória cache
- Mapeamento associativo por conjunto de *k* blocos:
  - Rótulo bem menor e é comparado apenas com *k* rótulos de um mesmo conjunto

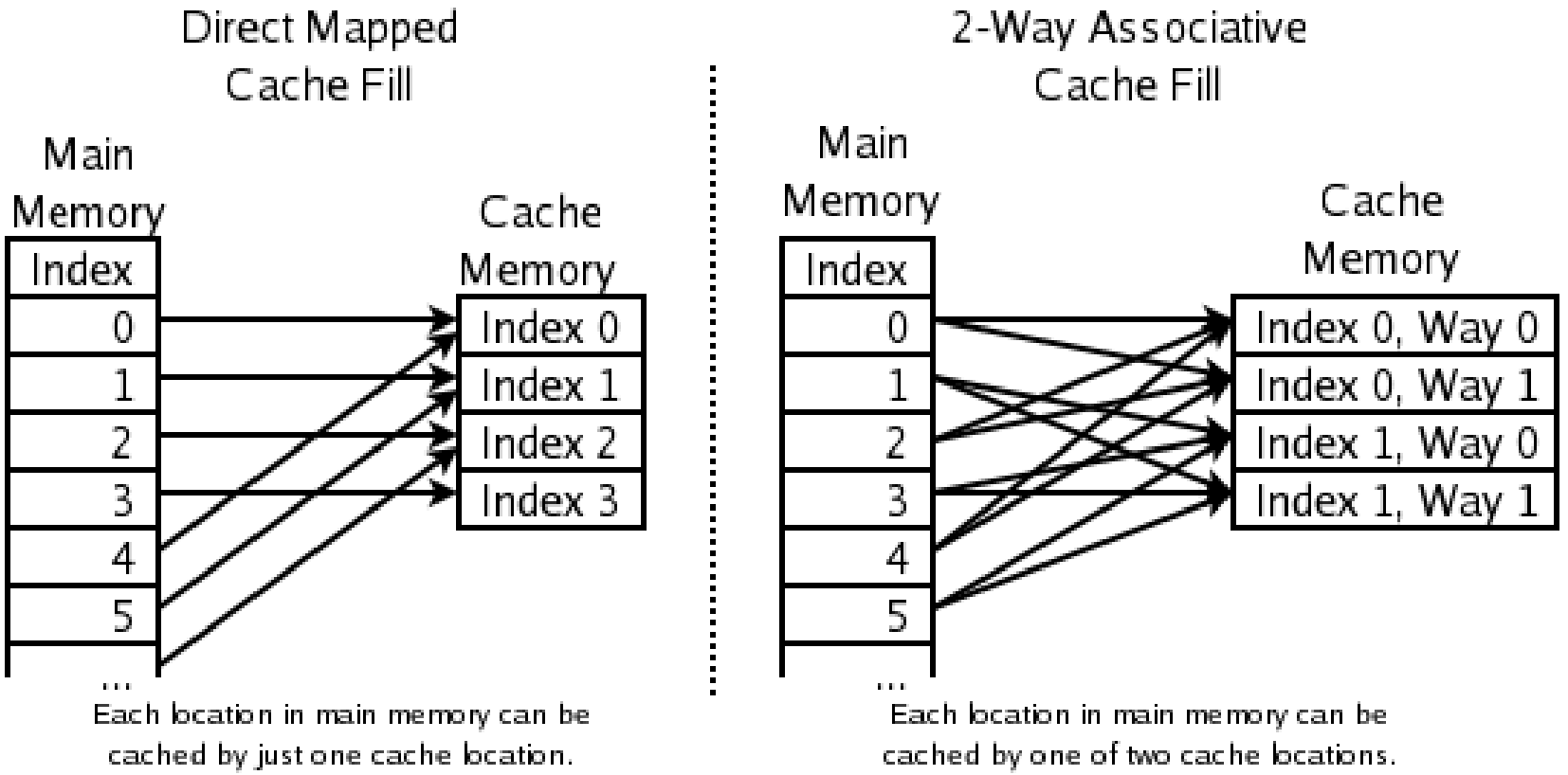
Mapeamento Associativo por Conjuntos

Organização – *K way set*

Figura 4.14 Organização da memória cache associativa em conjunto com *k* linhas por conjunto



Mapeamento Associativo por Conjunto



Princípios básicos

- Definição:
  - Tamanho da cache
  - Tamanho do bloco
  - Função de mapeamento
  - Algoritmo de substituição
  - Política de escrita
  - Número de caches

Algoritmos de substituição

- Toda vez que um novo bloco é trazido para a cache, se não existe espaço, ou seja, existe um conflito, um outro bloco deve sair da cache
  - Mapeamento direto: o mapeamento determina o bloco a ser substituído
  - Mapeamento associativo e associativo por conjuntos: requer algoritmo de substituição

Algoritmos de substituição

- Quatro possibilidades:
  - LRU – menos recentemente utilizado
  - FIFO – primeiro a entrar, primeiro a sair
  - LFU – menos frequentemente utilizado
  - Aleatório

Algoritmos de substituição

- LRU – *Least Recently Used*
  - O bloco a ser substituído é aquele que há mais tempo não é referenciado
  - Implementação mais simples em hardware: contador associado a cada bloco
    - Quando ocorre um acerto, o contador do bloco correspondente é zerado
    - Demais contadores são incrementados
    - Bloco com contador mais alto é substituído
  - Problema: quando número de blocos torna-se grande

Algoritmos de substituição

- FIFO – *First In, First Out*
  - Remove o bloco que está há mais tempo na cache
- LFU – *Least Frequently Used*
  - Remove o bloco que foi utilizado menos vezes em um determinado tempo
  - Frequência = utilização / tempo
  - Contador

Algoritmos de substituição

- Aleatório
  - Escolha de um bloco ao acaso para ser substituído
  - Implementação em hardware: contador
    - Contador é incrementado a cada ciclo de relógio
    - Quando a substituição é necessária, escolhe-se o bloco cujo endereço é igual ao valor atual do contador

Princípios básicos

- Definição:
  - Tamanho da cache
  - Tamanho do bloco
  - Função de mapeamento
  - Algoritmo de substituição
  - Política de escrita
  - Número de caches

Políticas de escrita

- Leitura na cache não afeta conteúdo
  - não há inconsistência entre memória principal e memória cache
- Escrita na cache
  - valores diferentes na memória cache e na memória principal
- Os valores deveriam ficar iguais em razão de:
  - Acessos de E/S feitos através da memória principal
  - Acessos à memória principal por múltiplos processadores
- Mecanismos para *write-hit*:
  - Write-through
  - Write-back
- Mecanismos para *write-miss*:
  - Write allocate
  - No allocate

Políticas de escrita
Write-through

- Cada escrita na cache é repetida imediatamente na memória principal
- Escrita adicional na memória principal reduz tempo médio de acesso à cache
- Estatisticamente apenas 5% a 34% dos acessos à memória são escritas
- Desvantagem: tráfego considerável de memória (gargalo)

Políticas de escrita
Write-back

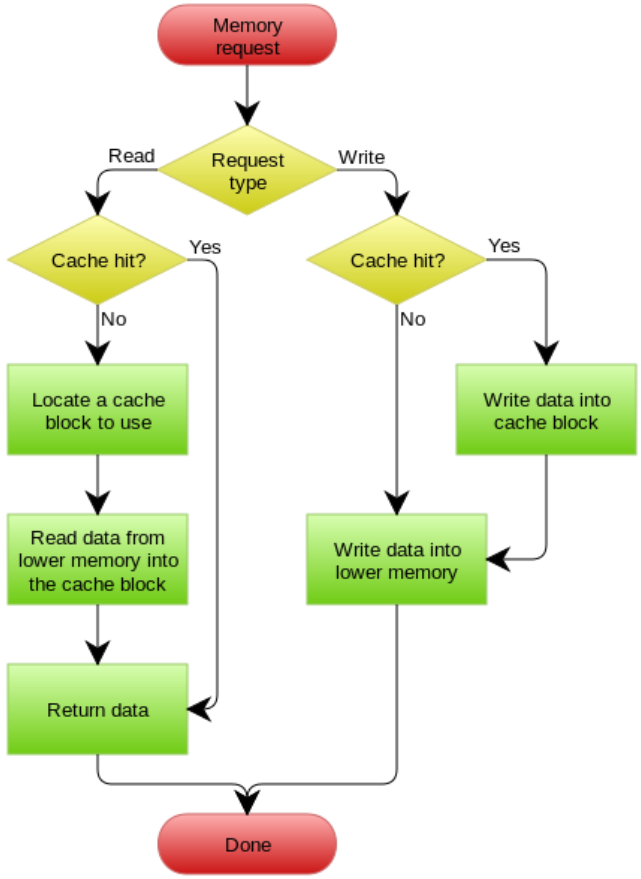
- O bloco da cache só é escrito de volta na memória principal quando precisa ser substituído
- Estratégia mais simples: a escrita é feita mesmo que o bloco não tenha sido alterado
- Estratégia mais complexa: a escrita só é feita se o bit MODIFICADO tiver com valor 1, significando que aquele bloco da cache foi alterado
- Problema: o acesso à memória principal pelos módulos de E/S deve ser feito através da memória cache
  - Circuitos tornam-se mais complexos
  - Potencial gargalo

Política de escrita
Write-miss

- *Write allocate*
  - Quando um dado deve ser escrito mas ainda não está na cache (*write-miss*), essa política faz uma cópia do bloco na cache (*read-miss*) e depois escreve na cache (*write-hit*)
- *No-write allocate*
  - Quando um dado deve ser escrito mas ainda não está na cache (*write-miss*), essa política escreve diretamente na memória principal, e o bloco não é carregado para a cache

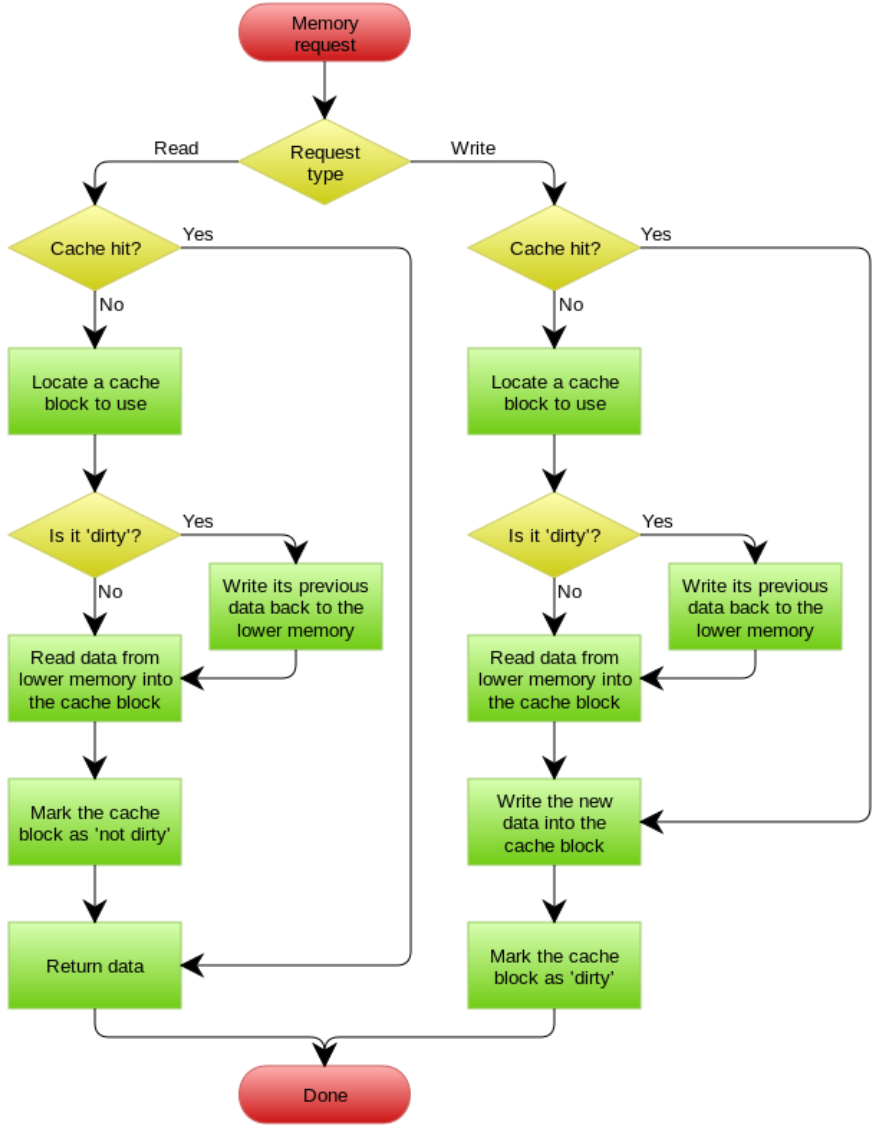
Composição de políticas de escrita

- *Write-through* (*write-hit*) e *No-write allocate* (*write-miss*)
  - Quanto se tem *write-hit* atualiza a cache e a MP
  - Quanto se tem *write-miss*, escreve só na MP



Composição de políticas de escrita

- *Write-back* (*write-hit*) e *Write allocate* (*write-miss*)
  - Quanto se tem *write-hit*, atualiza só na cache e marca o bloco como *modificado*
  - Quando se tem *write-miss*, traz o bloco para a cache, atualiza e marca como *modificado*



Algumas questões de escrita

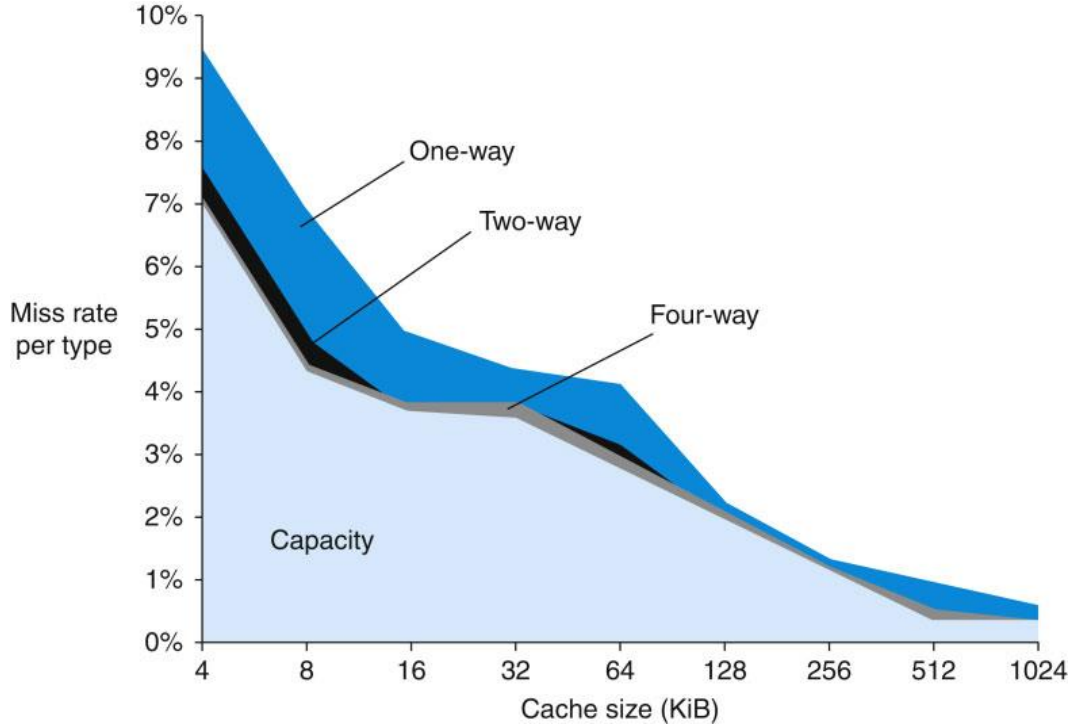
- O bit de modificação pode também ser chamado de *dirty bit*
- Quando é necessário que os conteúdos da cache e da memória principal estejam sincronizados (o mesmo conteúdo na mesma), pode-se realizar o procedimento chamado *flushing*, que atualiza todas os blocos modificados (ou *dirty blocks*) na memória principal

Origem das falhas (*misses*) nas caches

- Três possíveis falhas:
  - **Compulsórias**: falhas causadas pelo primeiro acesso a um bloco que nunca esteve na cache
  - **Capacidade**: falhas causas quando a cache não consegue conter todos os blocos necessários durante a execução de um programa, ou seja, elas acontecem quando os blocos são substituídos e depois recuperados
  - **Conflito**: ocorrem quando vários blocos disputam o mesmo conjunto, ou seja, quando se tem mapeamento direto ou mapeamento associativo por conjunto. Também chamadas de falhas de **colisão**

Origem das falhas (*misses*) nas caches

- Falhas
  - Compulsórias
    - Não podem ser vistas no gráfico
  - Capacidade
    - Diminuem conforme aumenta-se a capacidade
  - Conflito
    - Dependem do mapeamento



Princípios básicos

- Definição:
  - Tamanho da cache
  - Tamanho do bloco
  - Função de mapeamento
  - Algoritmo de substituição
  - Política de escrita
  - Número de caches

Número de Caches

- Aspectos de projeto:
  - número de níveis
  - memórias cache unificadas/separadas
- Níveis de memória cache:
  - Nível 1 (L1): interna à pastilha do processador
  - Nível 2 (L2): externa/interna à pastilha do processador
  - Nível 3 (L3): processadores mais recentes, externa à pastilha do processador, transferindo a L2 para dentro da pastilha do processador.
  - As transferências entre níveis de cache apresentam os mesmos problemas e possíveis soluções já discutidas

Número de Caches
Cache on-chip

- Memória cache na mesma pastilha do processador (*on-chip*)
- Reduz a atividade do processador no barramento externo $\Rightarrow$ diminui o tempo de execução e aumenta o desempenho global do sistema
- Enquanto faz acessos à cache interna, o barramento fica livre para efetuar outras transferências

Número de Caches
Cache off-chip

- Dados que ocasionaram falha em cache on-chip freqüentemente podem ser obtidos em cache off-chip

- Sem memória cache off-chip:
  - falha em cache $\rightarrow$ acesso à MP por meio do barramento
  - velocidade baixa do barramento + alto tempo de acesso à MP = baixo desempenho

- Utilizando memória cache off-chip:
  - Economia de tempo de acesso depende das taxas de acerto nas caches on-chip
  - Em geral, uso de cache off-chip melhora o desempenho

Número de caches

- Cache inclusiva
  - O dado que está na L1 está na L2 também
  - Exemplo: Intel Pentium 2, 3 e 4 e muitos processadores RISC
- Cache exclusiva
  - O dado é garantido estar em pelo menos uma das caches (L1 ou L2)
  - Quando tem um *miss* na L1 e o dado é encontrado na L2, esse dado é trocado com um bloco da L1
  - Exemplo: AMD Athlon

Número de caches

- Vantagem cache exclusiva
  - Maior capacidade (L1 + L2)
- Vantagem cache inclusiva
  - Quando os dispositivos de E/S desejam acessar a cache, só precisam ver a L2

Memórias cache unificadas/separadas

- Memória cache unificada (*Unified*): uma única memória cache para armazenar as referências a dados e instruções
- Memórias cache separadas (*Splited*): uma cache dedicada para dados e outra cache para instruções

Vantagens da memória cache unificada

- Em geral, taxa de acerto de memória cache unificada é maior do que de memórias cache separadas $\rightarrow$ fornece um balanceamento automático entre buscas de dados e instruções
- Apenas **uma** memória cache precisa ser projetada e implementada
- Utilizada nos níveis L2 e L3

Vantagens de memórias cache separadas

- Política de escrita só precisa ser aplicada à cache de dados
- Estratégias diferentes para cada cache: tamanho total, tamanho do bloco, organização
- Utilizada no nível L1

Exemplo i7

64 KB de cache L1 por núcleo: 8-way

256 KB de cache L2 por núcleo: 8-way

8 MB de cache L3 compartilhado entre todos os núcleos: 16-way

Desempenho de caches

Design change	Effect on miss rate	Possible negative performance effect
Increases cache size	Decreases capacity misses	May increase access time
Increases associativity	Decreases miss rate due to conflict misses	May increase access time
Increases block size	Decreases miss rate for a wide range of block sizes due to spatial locality	Increases miss penalty. Very large block could increase miss rate

Resumindo Caches

1. Onde um bloco pode ser colocado?

Scheme name	Number of sets	Blocks per set
Direct mapped	Number of blocks in cache	1
Set associative	$\frac{\text{Number of blocks in the cache}}{\text{Associativity}}$	Associativity (typically 2–16)
Fully associative	1	Number of blocks in the cache

Resumindo Caches

2. Como um bloco é encontrado?

Associativity	Location method	Comparisons required
Direct mapped	Index	1
Set associative	Index the set, search among elements	Degree of associativity
Full	Search all cache entries	Size of the cache
	Separate lookup table	0

Resumindo Caches

3. Que bloco deve ser substituído caso ocorra uma falha na cache? (Algoritmos de substituição)
   a) FIFO
   b) LRU
   c) LRU
   d) Aleatório

Deve-se considerar somente os mapeamentos totalmente associativo e associativo por conjunto, pois no mapeamento direto a própria maneira de achar o bloco já determina quem vai ser substituído

Resumindo Caches

4. O que acontece em uma escrita?
   a) Write-hit
      - Write-through
      - Write-back
   b) Write-miss
      - Write allocate
      - No-write allocate