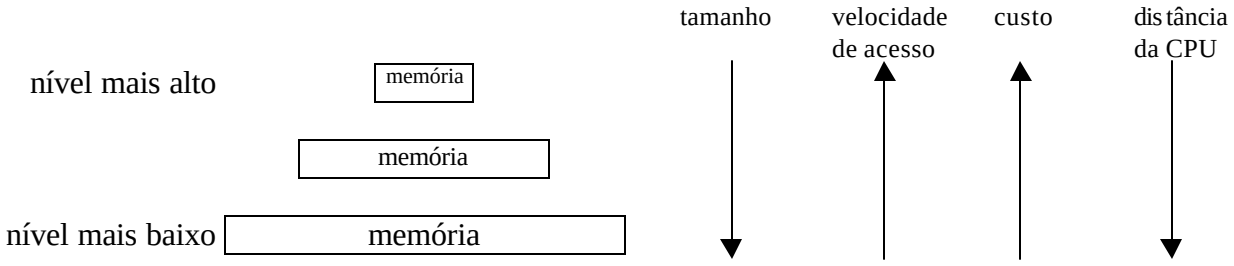
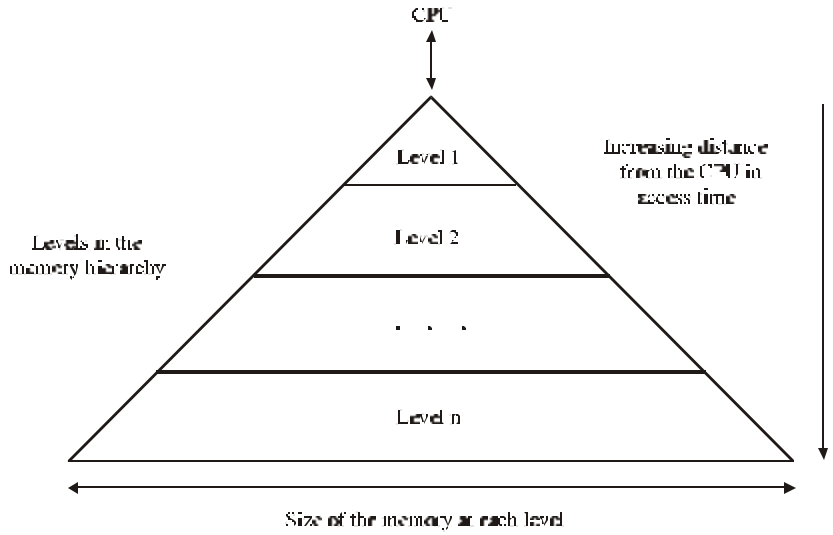
Hierarquia de Memórias

- Memória ideal:
  - Capacidade de armazenamento (tamanho) muito grande.
  - Tempo de acesso muito baixo (velocidade de acesso muito alta).
  - Custo baixo.

- Princípio de localidade:
  - Em um instante de tempo qualquer, um programa acessa uma porção relativamente pequena do seu espaço de endereçamento.
  - Localidade temporal: se um item (instrução ou dado) do programa é acessado, ele provavelmente logo será acessado novamente.
  - Localidade espacial: se um item do programa é acessado, itens cujos endereços são próximos provavelmente logo serão acessados.
  - Programas em geral exibem localidade temporal e espacial devido às estruturas de dados e de controle utilizadas (laços, execução seqüencial, subrotinas, vetores, ...).

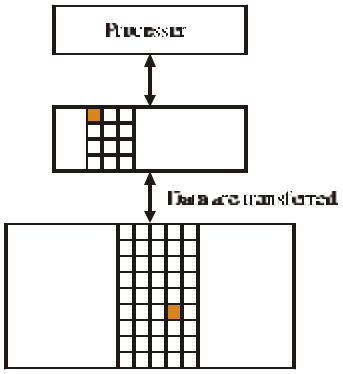
- Hierarquia de memórias:
  - Idéia: tirar vantagem do princípio de localidade.
  - Múltiplos níveis de memória com diferentes tamanhos e velocidades de acesso.
  - Memória mais rápida colocada mais perto do processador, memória mais lenta colocada mais distante do processador.
  - Objetivo:
    - Dar ilusão de que computador possui memória ideal.
    - Oferecer para programas memória com grande capacidade (nível mais baixo) e com tempo de acesso pequeno (nível mais alto) ⇒ obter bom desempenho dos programas.





- Sistema de memória organizado como uma hierarquia de memórias:
  - Existem vários níveis.
  - Em geral, um nível mais alto é um subconjunto de qualquer nível mais baixo. Isto é, dados contidos em um nível, estão contidos em todos os níveis abaixo.
  - Dados são copiados apenas entre níveis adjacentes.

- Entre 2 níveis adjacentes: nível mais alto e nível mais baixo.
  - Bloco: unidade mínima de informação que pode estar presente ou não no nível mais alto.
  - Em geral, um bloco inteiro é transferido entre níveis.



- Entre 2 níveis adjacentes: nível mais alto e nível mais baixo.
  - Se dado requisitado pelo processador está em algum bloco do nível mais alto: acerto (hit).
  - Se não está: falha (miss).
    - Nível mais baixo é acessado para obter bloco contendo dado. Bloco é copiado para nível mais alto.
  - Taxa de acertos (hit ratio): % dos acessos à memória que causam acertos.
  - Taxa de falhas (miss ratio): % dos acessos à memória que causam falhas.
    Taxa de falhas = 1 – taxa de acertos.
  - Tempo de acerto (hit time): tempo de acesso ao nível mais alto (incluindo tempo para determinar se acesso é acerto ou falha).
  - Tempo de tratamento da falha (miss penalty): tempo para substituir um bloco do nível mais alto pelo bloco desejado do nível mais baixo e para entregar este bloco ao processador.

- Desempenho:
  - Tempo de acerto << tempo de tratamento de falha
    $\Rightarrow$ acessos que causam acertos são mais rápidos que acessos que causam falhas.
  - Se taxa de acertos é alta o suficiente, hierarquia de memórias oferece memória com:
    - Tempo efetivo de acesso $\cong$ tempo de acesso do nível mais alto.
    - Tamanho = tamanho do nível mais baixo.

    $\Rightarrow$ objetivo é alcançado.
  - Taxa de acertos é uma medida de desempenho da hierarquia de memórias (mas não é a única medida).

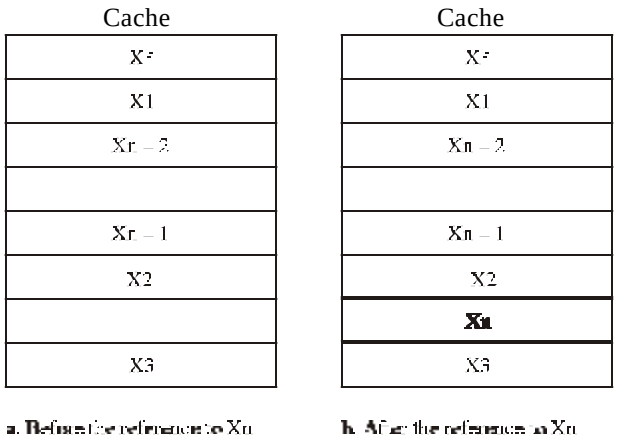
- Hierarquia de memórias tira vantagem da localidade:
  - Temporal: mantendo itens acessados mais recentemente nos níveis mais altos.
  - Espacial: copiando blocos que contêm múltiplas palavras contíguas na memória para níveis mais altos.

- Construção do sistema de memória afeta:
  - Como SO gerencia memória e entrada e saída.
  - Como compiladores geram código.
  - Como aplicações usam computador.

Memórias Cache

- Memória cache: nível da hierarquia de memórias entre processador e memória principal (MP).

- Exemplo:
  - Cache simples.
  - Bloco é uma palavra.
  - Cache contém palavras X_1, X_2, ..., X_{n-1}.
  - Processador acessa palavra X_n que não está na cache ⇒ ocorre falha ⇒ palavra X_n é trazida da MP para cache.



a. Before the reference to Xn b. After the reference to Xn

- Questões:
  - Como saber se item está na cache?
  - Se item está na cache, como encontrá-lo na cache?

Cache Mapeada Diretamente

- Solução simples: associar uma palavra da MP com uma posição da cache, com base no endereço da palavra na MP.

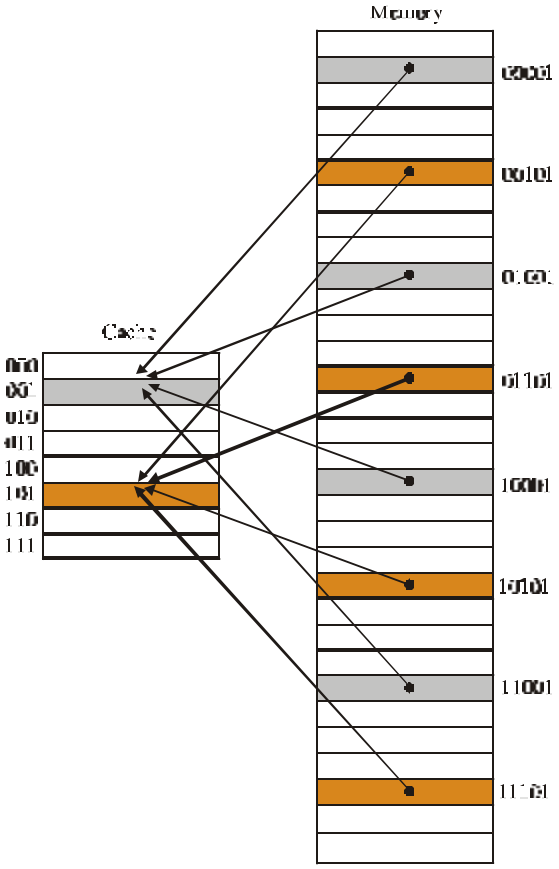
- Exemplo de mapeamento do endereço na MP na posição da cache:
  Posição do bloco na cache = endereço do bloco na MP mod nº de blocos da cache

- Em geral:
  - nº de blocos da cache = 2^k ⇒
  - Posição do bloco na cache = *k* bits menos significativos do endereço do bloco na MP.

- Várias posições da MP são mapeadas em uma mesma posição da cache.

- Exemplo:
  - Cache mapeada diretamente com 2^3 blocos de uma palavra de 32 bits.
  - MP com 32 palavras.

- Posição do bloco na cache = 3 bits menos significativos do endereço do bloco na MP.
- Endereços 00**001**, 01**001**, 10**001** e 11**001** da MP são mapeados na posição 001 da cache. Endereços 00**101**, 01**101**, 10**101** e 11**101** da MP são mapeados na posição 101 da cache.



- Questões:
  - Se cada posição da cache pode conter o conteúdo de diferentes posições da MP, como saber se o dado na cache corresponde à palavra requisitada?
  - Como saber se posição da cache contém uma informação válida?

- Solução: adicionar tag e bit de validade à cache.
  - Cada posição da cache contém um campo tag e um bit de validade, além do campo dado.
  - Campo dado: contém item armazenado (instrução ou dado).
  - Campo tag: contém informação necessária para identificar se dado naquela posição da cache corresponde ao dado procurado.
  - Bit de validade: indica se posição da cache contém informação válida ou não. Se bit é 0, posição não contém dado válido.
  - Quando computador é ligado, cache está vazia ⇒ todas as posições possuem bit de validade 0.

- Na cache mapeada diretamente:
  - Cache com 2^k posições.
  - Posição do bloco na cache = *k* bits menos significativos do endereço do bloco na MP.
  - tag = bits mais significativos restantes do endereço do bloco na MP.
  - Quando processador tenta acessar um bloco:
    - Usando *k* bits menos significativos do endereço do bloco na MP, determina posição da cache onde bloco pode estar.

- Se bit de validade da posição da cache é 0:
  - Posição não possui dado válido ⇒ falha na cache.
- Senão:
  - Compara campo tag da posição da cache com bits mais significativos restantes do endereço do bloco na MP.
  - Se são diferentes:
    - Posição contém outro dado ⇒ falha na cache.
  - Senão:
    - Campo dado da posição contém dado requisitado pelo processador ⇒ acerto na cache.

- Exemplo:
  - Cache mapeada diretamente com 2^3 blocos de uma palavra de 32 bits.
  - Posição do bloco na cache = 3 bits menos significativos do endereço do bloco na MP.
  - tag = 2 bits mais significativos do endereço do bloco na MP.
  - Estado inicial da cache: vazia.

Cache

	Validade	Tag	Dado
000	0		
001	0		
010	0		
011	0		
100	0		
101	0		
110	0		
111	0		

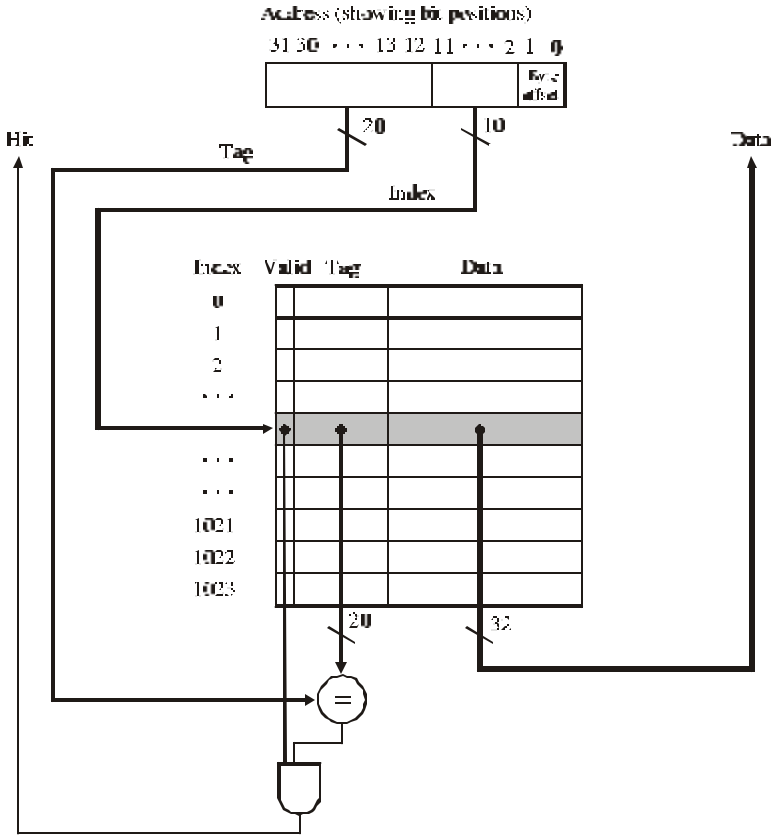
  - Acessos feitos pelo processador:

Endereço acessado	Bloco da cache	Acerto ou falha na cache
$22_{10} = 10110_2$	110_2	Falha
$26_{10} = 11010_2$	010_2	Falha
$22_{10} = 10110_2$	110_2	Acerto
$26_{10} = 11010_2$	010_2	Acerto
$16_{10} = 10000_2$	000_2	Falha
$3_{10} = 00011_2$	011_2	Falha
$16_{10} = 10000_2$	000_2	Acerto
$18_{10} = 10010_2$	010_2	Falha

- Acesso à palavra de endereço 10010_2:
  - Palavra de endereço 11010_2 que ocupava bloco 010_2 da cache é substituída pela palavra de endereço 10010_2.
  - Na cache mapeada diretamente há uma única posição onde colocar uma nova palavra ⇒ há uma única opção de qual palavra substituir.
  - Cache não estava cheia e foi necessário fazer substituição.
  - Palavra acessada mais recentemente substitui palavra acessada menos recentemente ⇒ explora localidade temporal.

- Exemplo:
  - Cache mapeada diretamente com 2^{10} blocos de uma palavra de 32 bits.
  - Endereço da MP com 32 bits.

- Cada posição da MP armazena um byte.
- 2 bits mais baixos do endereço da MP: deslocamento dos bytes dentro da palavra.
- 10 bits mais baixos seguintes do endereço da MP: posição na cache.
- 20 bits mais altos do endereço da MP: tag.



- Número total de bits da cache depende de:
  - Tamanho da cache (número de posições da cache).
  - Número de bits do endereço da MP.

- Exemplo:
  - Cache mapeada diretamente com 2^n palavras de 32 bits.
  - Endereço da MP com 32 bits.
  - Cada posição da MP armazena um byte.
  - 2 bits mais baixos do endereço da MP: deslocamento dos bytes dentro da palavra.
  - *n* bits mais baixos seguintes do endereço da MP: posição na cache.
  - $32 - n - 2$ bits mais altos do endereço da MP: tag.
  - Número total de bits da cache =
    $2^n \times$ (tamanho do bloco + tamanho da tag + tamanho do bit de validade) =
    $2^n \times (32 + (32 - n - 2) + 1) = 2^n \times (63 - n)$.

Tratamento de Falhas na Cache

- Memórias de instruções e de dados vistas na via de dados com pipeline são memórias cache:
  - Tempo de acesso à cache compatível com operação da via de dados. Acesso à cache é feito em um estágio do pipeline ⇒ leva 1 ciclo.
  - Cache de instruções e cache de dados separadas ⇒ evita conflito estrutural no pipeline.

- Quando processador tenta acessar instrução ou dado:
  - Se encontra na cache, processador continua executando normalmente.
  - Senão, trata falha na cache:
    - Busca instrução ou dado na MP e copia-o para cache.
    - Quando instrução ou dado já está presente na cache, execução é reiniciada no passo que causou falha na cache.

- Tratamento de falha na cache:
  - Realizado pela unidade de controle e por um controlador separado, que inicia acesso à MP e copia bloco para cache.
  - Cria bolha(s) no pipeline.

- Tratamento de falha no acesso de leitura (de instrução):
  1. No 1º passo da via de dados, tenta ler instrução da cache de instruções e tem falha.
     Conteúdo de IR está inválido.
     PC foi incrementado de 4 no 1º passo.
     Calcula endereço da instrução que causou falha (PC – 4).
  2. Inicia leitura na MP no endereço obtido.
     Espera leitura ser realizada.
  3. Escreve bloco lido da MP na cache, preenchendo tag e bit de validade.
  4. Reinicia execução da instrução no 1º passo da via de dados (busca instrução novamente na cache ⇒ será acerto).

- Tratamento de falha no acesso de leitura de dado é similar.

- Para reduzir tempo de tratamento da falha na cache: técnica stall on use.
  - Permitir que processador continue executando instruções enquanto falha é tratada.
  - Objetivo: reduzir número de ciclos que processador "atrasa" quando ocorre falha.
  - Apenas para falha no acesso a dado. Não serve para falha no acesso a instrução.
  - Exemplo: instrução `lw` busca dado na cache de dados e tem falha.
    - Processador continua buscando e executando instruções enquanto falha é tratada. Quando dado lido pela instrução `lw` é necessário para uma instrução posterior, execução pára.
  - Reduz atraso quando ocorre falha apenas um pouco, pois dado provavelmente será necessário logo.

- Passos do acesso de leitura (de instrução ou dado):
  1. Endereço do item a ser acessado é passado para cache apropriada (de instruções ou de dados).
     Para leitura de instrução, endereço está em PC.
     Para leitura de dado, endereço vem da ALU.
  2. Se há acerto (acerto de leitura):
     Palavra é lida da cache.
     Senão (falha de leitura):
     Endereço é passado para MP.
     Quando leitura na MP termina de ser realizada, escreve dado na cache.

- Acesso de escrita (apenas de dado):
  - Exemplo: instrução `sw`.
  - Se ocorre falha, não precisa ler dado da MP, pois vai alterar seu valor.

- Independente de ser acerto ou falha, escreve dado na cache e atualiza tag.
- MP possui valor antigo do dado e cache valor novo ⇒ MP e cache estão inconsistentes.

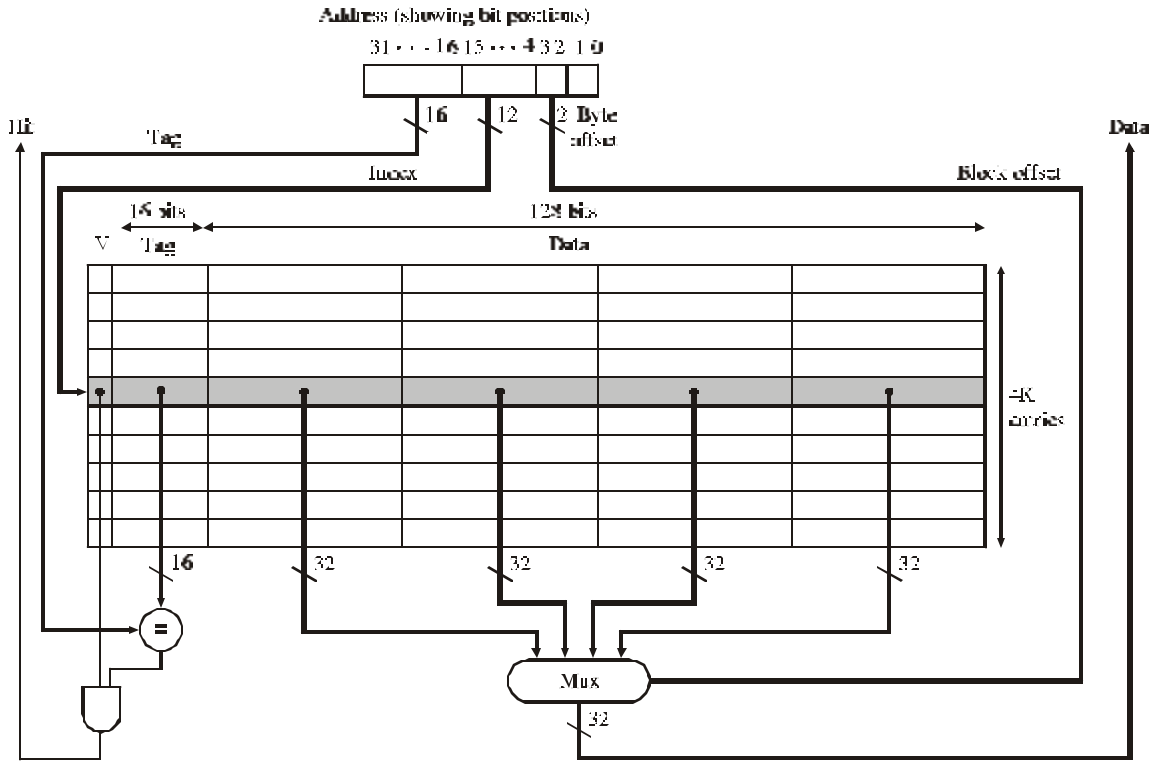
- Manutenção da consistência:
  - 1ª solução: esquema write-through.
    - Sempre que realiza escrita, escreve novo valor do dado na cache e na MP.
    - Pode não ter bom desempenho.
  - 2ª solução: esquema write-through, usando buffer de escrita.
    - Sempre que realiza escrita, escreve novo valor do dado na cache e no buffer de escrita.
    - Processador não precisa esperar escrita na MP para continuar execução. Continua executando instruções enquanto escrita na MP é feita a partir do buffer.
    - Buffer pode ter várias posições, para permitir escritas quase consecutivas.
  - 3ª solução: esquema write-back.
    - Quando realiza escrita, escreve novo valor do dado apenas na cache.
    - Dado só é atualizado na MP quando retirado (substituído) da cache.
    - Melhor desempenho e maior complexidade.

- Passos do acesso de escrita (de dado): usando write-through.
  1. Endereço do item a ser acessado é passado para cache de dados (endereço vem da ALU).
  2. Escreve tag e dado e seta bit de validade na posição da cache.
  3. Escreve dado na MP (write-through).

Explorando Localidade Espacial

- Para explorar localidade espacial:
  - Bloco da cache com várias palavras da MP.
  - Quando ocorre falha, busca na MP várias palavras adjacentes e copia-as para um bloco da cache.

- Exemplo:
  - Cache mapeada diretamente com 2^{12} blocos com 4 palavras de 32 bits.
  - Endereço da MP com 32 bits.
  - Cada posição da MP armazena um byte.
  - 2 bits mais baixos do endereço da MP: deslocamento dos bytes dentro da palavra.
  - 2 bits mais baixos seguintes do endereço da MP: deslocamento das palavras dentro do bloco. Controla multiplexador que seleciona palavra desejada dentro do bloco.
  - 12 bits mais baixos seguintes do endereço da MP: posição na cache.
  - 16 bits mais altos do endereço da MP: tag.



- Tratamento da falha de leitura é o mesmo: bloco inteiro é copiado da MP para cache.

- Acesso de escrita precisa ser realizado diferentemente.
  - Processador pede para escrever em uma palavra.
  - Palavras adjacentes a esta devem estar no mesmo bloco da cache.
  - Falha de escrita requer leitura na MP.

- Passos do acesso de escrita (de dado): usando write-through.
  1. Endereço do item a ser acessado é passado para cache (endereço vem da ALU).
  2. Compara tag.
  3. Se há acerto (acerto de escrita):
     Escreve palavra na posição da cache.
     Senão (falha de escrita):
     Endereço é passado para MP, para ler bloco inteiro.
     Quando leitura na MP termina de ser realizada, escreve bloco na cache, atualizando tag e bit de validade.
     Escreve palavra na posição da cache.
  4. Escreve dado na MP (write-through).

- Exemplo:
  - Acessos às palavras de endereço 16, 24, 20.
  - Cache inicialmente vazia.
  - Cache com blocos de 4 palavras.
  - Acesso ao endereço 16:
    - Falha na cache.
    - Bloco contendo palavras de endereços 16, 20, 24 e 28 é copiado para cache.
  - Acessos aos endereços 24 e 20: acertos.
  - Para cache com bloco de 1 palavra, acessos aos endereços 24 e 20: falhas.

- Vantagem de aumentar tamanho do bloco da cache:
  - Explora localidade espacial ⇒ reduz taxa de falhas.
  - Espaço gasto com tags e bits de validade é proporcionalmente menor, pois cada tag e bit de validade são usados para um grupo de palavras e não para apenas uma palavra.

- Desvantagem de aumentar tamanho do bloco da cache:
  - Tempo de tratamento da falha aumenta, pois busca várias palavras na MP.
  - Se aumenta demais tamanho do bloco:
    - Localidade espacial entre palavras de um bloco diminui.
    - Número de blocos da cache diminui ⇒ blocos serão substituídos antes de terminarem de ser utilizados.

    ⇒ pode aumentar taxa de falhas.

Explorando Associatividade na Cache para Reduzir Taxa de Falhas

- Diferentes esquemas de colocação de blocos na cache.
  - Um extremo: cache mapeada diretamente (*direct mapped cache*).
  - Outro extremo: cache totalmente associativa (*fully associative*).
  - Intermediário: cache set-associativa (*set-associative cache*).

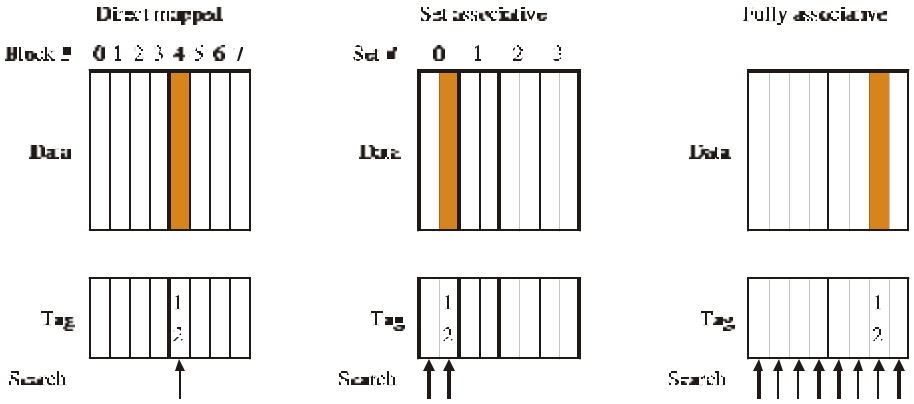
- Cache mapeada diretamente:
  - Mapeamento direto do endereço da MP em uma única posição da cache.
  - Um bloco da MP pode ser colocado em uma única posição da cache.

- Cache totalmente associativa:
  - Um bloco da MP pode ser colocado em qualquer posição da cache.
  - Fully associative: um bloco da MP pode ser associado a qualquer posição da cache.
  - Para achar um bloco na cache:
    - Busca em todos os blocos da cache.
    - Busca em paralelo: um comparador para cada bloco da cache.
    - Custo alto ⇒ viável para caches com poucos blocos.

- Cache set-associativa:
  - Um bloco da MP pode ser colocado em um número fixo de posições da cache.
  - Cache set-associativa com *n* posições para um bloco da MP: cache *n*-way set-associativa.
  - cache *n*-way set-associativa:
    - Possui um número de conjuntos (sets).
    - Cada conjunto possui *n* blocos.
    - Cada bloco da MP é associado à um único conjunto da cache.
    - Bloco da MP pode ser colocado em qualquer dos *n* blocos daquele conjunto.
  - Cache set-associativa: combina colocação de blocos mapeados diretamente com colocação totalmente associativa.
    - Um bloco da MP é mapeado diretamente em um conjunto:

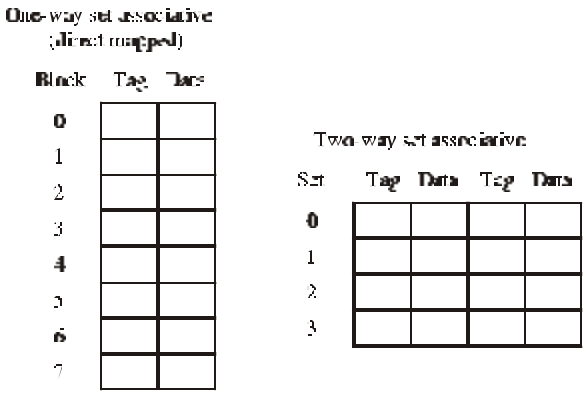
      Nº do conjunto do bloco na cache = endereço do bloco na MP mod nº de conjuntos
    - Para achar um bloco na cache busca em todos os blocos do conjunto.
    - Busca em paralelo: um comparador para cada bloco do conjunto (*n* comparadores).

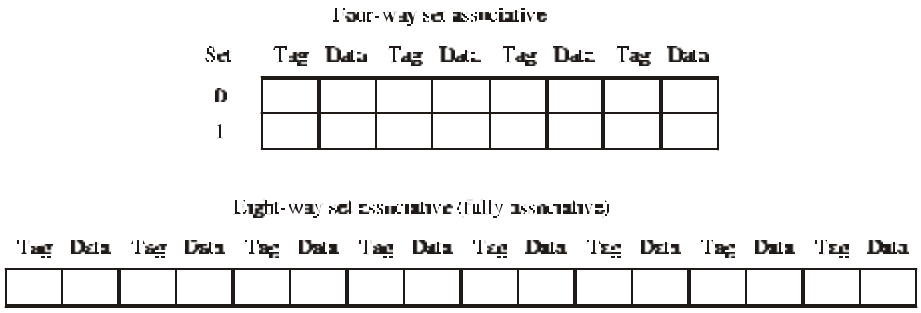
- Exemplo:
  - Cache com 8 blocos.
  - Colocação do bloco da MP de endereço 12:
    - Cache mapeada diretamente:
      - Posição do bloco na cache = 12 mod 8 = 4.
      - Busca neste único bloco da cache.
    - Cache 2-way set-associativa:
      - 4 conjuntos de 2 blocos.
      - Nº do conjunto do bloco na cache = 12 mod 4 = 0.
      - Busca nos 2 blocos deste conjunto da cache.
    - Cache totalmente associativa:
      - Busca nos 8 blocos da cache.



- Diferentes esquemas de colocação de blocos vistos como variação da cache set-associativa:
  - Cache mapeada diretamente: 1-way set-associative cache.
  - Cache totalmente associativa: cache *m*-way set-associativa, *m* = nº de blocos da cache.

- Exemplo:
  - Cache com 8 blocos.
  - Nº de blocos da cache = nº de conjuntos × associatividade.
  - Mantendo tamanho da cache fixo: aumento na associatividade ⇒ aumenta nº de blocos por conjunto ⇒ diminui nº de conjuntos.





- Aumento da associatividade:
  - Vantagem: reduz taxa de falhas.
  - Desvantagens: aumenta custo e pode aumentar tempo de acerto.

- Exemplo:
  - Caches de 4 blocos, inicialmente vazias.
  - Acessos aos blocos da MP de endereços 0, 8, 0, 6, 8.
  - Cache mapeada diretamente:
    - 4 conjuntos de 1 bloco.
    - Nº de falhas =

nº do bloco da cache	Cache
0	
1	
2	
3	

Bloco da MP acessado	Bloco da cache	Acerto ou falha
0		
8		
0		
6		
8		

  - Cache 2-way set-associativa:
    - 2 conjuntos de 2 blocos.
    - Supondo que substitui bloco menos recentemente acessado do conjunto (LRU – least recently used).
    - Nº de falhas =

nº do conjunto da cache	Cache	
0		
1		

Bloco da MP acessado	Conjunto da cache	Acerto ou falha
0		
8		
0		
6		
8		

  - Cache totalmente associativa:
    - 1 conjunto de 4 blocos.
    - Nº de falhas =

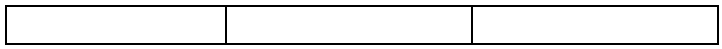
Cache

Bloco da MP acessado	Acerto ou falha
0	
8	
0	
6	
8	

Localizando um Bloco em uma Cache Set-Associativa

- Cada bloco da cache possui uma tag.

- Endereço da MP:
  - Índice: seleciona conjunto da cache que pode conter o bloco acessado.
  - Tag: comparada com tag de todos os blocos do conjunto selecionado.



tag | índice (nº do conjunto) | deslocamento dentro do bloco

- Tempo de acerto deve ser pequeno:
  - Comparação com tag de cada bloco do conjunto é feita em paralelo.
  - Nº de comparadores necessários = associatividade da cache.

- Mantendo tamanho total da cache constante e duplicando associatividade:
  - Duplica nº de blocos por conjunto ⇒ aumenta em 1 bit tamanho da tag.
  - Reduz pela metade nº de conjuntos ⇒ reduz de 1 bit tamanho do índice.
  - Duplica nº de comparações feitas em paralelo.

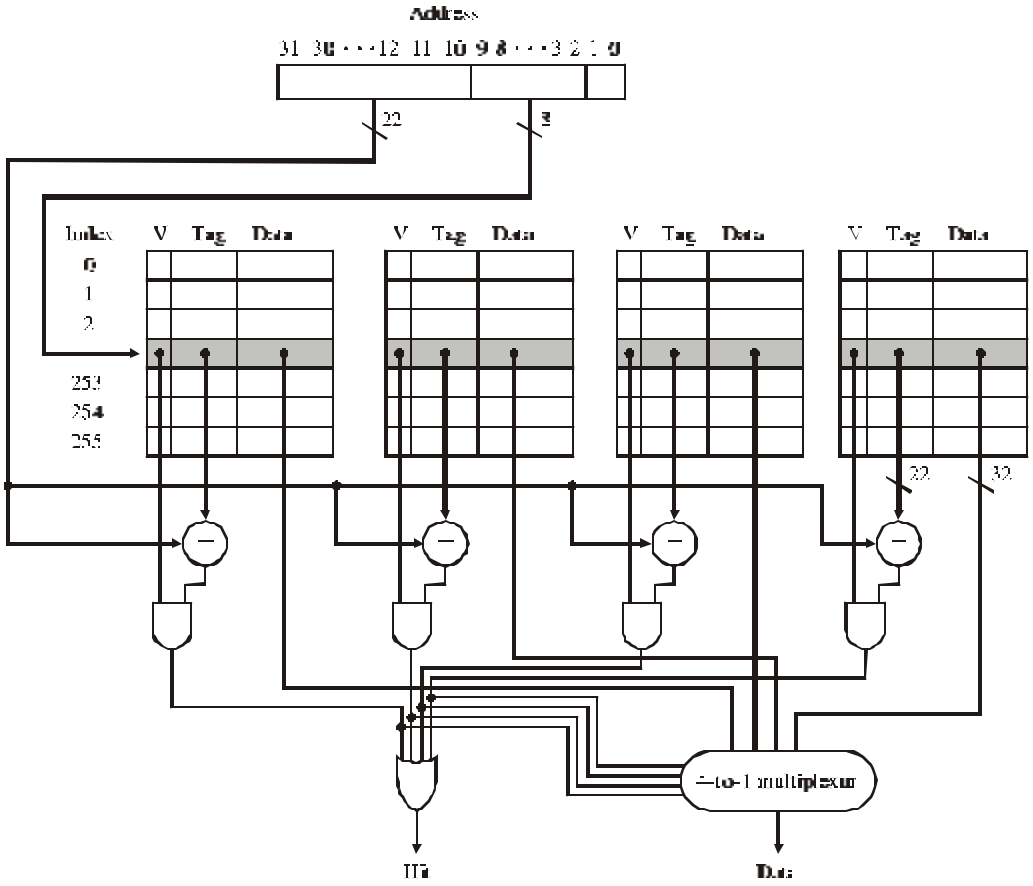
- Cache totalmente associativa:
  - Há um único conjunto ⇒ não há índice.
  - Tag deve ser comparada com tag de todos os blocos da cache em paralelo.
  - Nº de comparadores necessários = nº de blocos da cache.

- Cache mapeada diretamente:
  - Um único bloco por conjunto.
  - Tag deve ser comparada com tag do único bloco da cache que pode conter bloco acessado.
  - Nº de comparadores necessários = 1.

- Vantagem da cache set-associativa:
  - Menor taxa de falhas.

- Desvantagens da cache set-associativa:
  - Maior custo de hardware: mais comparadores.
  - Tempo de acerto mais longo: comparar tags e selecionar entre blocos do conjunto.

- Exemplo:
  - Cache 4-way set-associativa.
  - 4 blocos por conjunto.
  - 4 comparadores.
  - Um multiplexador 4 × 1 para selecionar um (ou nenhum) bloco do conjunto. (Sinal de entrada de controle do multiplexador está decodificado).



Escolha do Bloco a Substituir

- Quando ocorre falha:
  - Na cache mapeada diretamente:
    - Bloco requisitado pode ser colocado em exatamente uma posição específica.
    - Bloco que ocupava aquela posição deve ser substituído.
  - Na cache set-associativa:
    - Bloco requisitado pode ser colocado em qualquer bloco de um conjunto específico.
    - Pode escolher qual bloco do conjunto deve ser substituído.
  - Na cache totalmente associativa:
    - Bloco requisitado pode ser colocado em qualquer bloco da cache.
    - Pode escolher qual bloco da cache deve ser substituído.
- LRU (*least recently used* – menos recentemente utilizado):
  - Esquema mais utilizado para escolha do bloco a substituir (a retirar da cache).
  - Escolhe para substituir bloco menos recentemente utilizado do conjunto (isto é, bloco que está a mais tempo sem ser acessado).
  - Implementação: controlar quando (em que ordem) cada bloco do conjunto foi acessado pela última vez em relação aos outros blocos do conjunto.

Memória Virtual

- Nível da hierarquia de memórias abaixo da MP: dispositivo de armazenamento secundário (disco magnético).

- Mecanismo de memória virtual: faz MP atuar como uma "cache" do dispositivo de armazenamento secundário.

- Hierarquia de memórias:
  - MP contém apenas as partes ativas (localidade atual) de vários programas. Programas inteiros ficam no disco.
  - Cache contém apenas partes mais ativas de um programa.

- Objetivos do mecanismo de memória virtual:
  - Permitir que vários programas compartilhem a MP de maneira eficiente e protegida.
  - Permitir que programas sejam maiores que a MP.

- Cada programa possui o seu espaço de endereçamento virtual.
  - Mecanismo de memória virtual implementa o mapeamento do espaço de endereçamento virtual de um programa nos endereços físicos da MP.

- Memória virtual com paginação:
  - Bloco (de tamanho fixo) da MP: página.
  - Falha na MP: falta de página.
  - Programa possui uma seqüência de páginas virtuais.
  - MP é vista como um conjunto de páginas físicas.

- Processador acessa um endereço virtual (lógico) de um programa:
  - Endereço virtual é mapeado em um endereço real (físico) da MP (pelo hardware e software – sistema operacional):
    - Determina se página virtual que contém endereço virtual está presente em alguma página física da MP e em qual página física.
    - Determina endereço real correspondente ao endereço virtual acessado.
  - Se página que contém endereço físico está presente na MP então:
    - Endereço físico é acessado na MP.
  - Senão:
    - Houve falta de página.
    - Realiza tratamento da falta de página:
      - Se há página física livre na MP então:
        - Carrega página faltosa do disco para página física da MP.
        - Endereço físico é acessado na MP.
      - Senão:
        - Escolhe página presente na MP para ser retirada (substituída), de acordo com alguma política de substituição (por exemplo LRU).
        - Se página escolhida foi modificada então:
          - Copia página escolhida de volta da MP para o disco (write-back).
        - Carrega página faltosa do disco para página física liberada da MP.
        - Endereço físico é acessado na MP.

- Mecanismo de memória virtual realiza relocação dinâmica:
  - Relocação dinâmica: mapeamento dos endereços virtuais usados pelo programa em endereços fisicos da MP, em tempo de execução.
  - Permite que programa seja carregado em qualquer posição da MP ⇒ simplifica carga do programa na MP.

- Número de páginas virtuais pode ser maior que número de páginas físicas ⇒ mecanismo de memória virtual dá a ilusão de haver uma memória principal enorme.

- Características da memória virtual:
  - Blocos (páginas) grandes.
  - Colocação de páginas na MP completamente associativa.
    Objetivo: reduzir taxa de falta de páginas.
  - Tratamento da falta de página envolve acesso ao disco ⇒ tempo de tratamento da falta de página é muito longo.
  - Tratamento da falta de página é realizado por software (sistema operacional).
  - Utiliza algoritmo de substituição de página LRU, aproximação de LRU ou outro.
  - Mantém consistência do disco com MP usando esquema write-back.

- Mapeamento do endereço virtual em endereço físico:

  - Usa tabela de páginas do programa.
  - Tabela de páginas de um programa informa, para cada página virtual do programa, se ela está presente em alguma página física da MP e em qual página física.
  - Tabela de páginas fica na MP.

  - Para acessar um endereço virtual do programa:
    - Acessar tabela de páginas (para mapear endereço virtual em endereço físico).
    - Acessa endereço físico na MP.

    ⇒ Acesso fica mais lento (requer 2 acessos à MP).

  - Solução: usar TLB (translation lookaside buffer).
    - Memória cache que mantém mapeamentos mais recentes do programa.
    - Contém número da página virtual (tag) e número da página física correspondente (dado) dos últimos endereços acessados pelo programa.
    - Objetivo: evitar acesso à tabela de páginas (na maioria das vezes).