UNIVERSIDADE FEDERAL DO PIAUÍ – UFPI
CAMPUS SENADOR HELVÍDIO NUNES DE BARROS - PICOS
Disciplina: Arquitetura e Organização de Computadores
Acadêmico: Weliton de Sousa Araujo

Memória cache

4.1 Visão geral do sistema de memória do computador
Características dos sistemas de memória

O assunto complexo da memória de computador pode ser melhor compreendido se classificarmos os sistemas de memória de acordo com suas principais características:

Localização: indica se a memória é interna ou externa ao computador. A memória interna normalmente significa a memória principal, mas existem outras formas de memória interna. O processador requer sua própria memória local, na forma de registradores. A cache é outra forma de memória interna. A memória externa consiste em dispositivos de armazenamento periféricos, como disco e fita, que são acessíveis ao processador por meio de controladores de E/S.

Capacidade: Para a memória interna, isso normalmente é expresso em termos de bytes ou palavras. Os tamanhos comuns de palavra são 8, 16 e 32 bits. A capacidade da memória externa normalmente é expressa em termos de bytes.

Unidade de transferência: Para a memória interna, a unidade de transferência é igual ao número de linhas elétricas para dentro e para fora do módulo de memória. Isso pode ser igual ao tamanho da palavra, mas normalmente é maior.

Método de acesso: método de acesso das unidades de dados, que inclui os seguintes:

Acesso sequencial: a memória é organizada em unidades de dados chamadas registros. O acesso é feito em uma sequência linear específica.

Acesso direto: envolve um mecanismo compartilhado de leitura-escrita. Porém, os blocos ou registros individuais têm um endereço exclusivo, baseado no local físico.

Acesso aleatório: cada local endereçável na memória tem um mecanismo de endereçamento exclusivo, fisicamente interligado. A memória principal e alguns sistemas de cache são de acesso aleatório.

Associativo: esse é o tipo de memória de acesso aleatório que permite fazer uma comparação de um certo número de bit desejados dentro de uma palavra para uma combinação especificada, e faz isso para todas as palavras simultaneamente. Assim, uma palavra é recuperada com base em uma parte de seu conteúdo,

em vez do seu endereço. As memórias cache podem empregar o acesso associativo.

Desempenho. Três parâmetros de desempenho são usados:

Tempo de acesso (latência): para a memória de acesso aleatório, esse é o tempo gasto para realizar uma operação de leitura ou escrita, ou seja, o tempo desde o instante em que um endereço é apresentado à memória até o instante em que os dados foram armazenados ou se tornaram disponíveis para uso.

Tempo de ciclo de memória: Consiste no tempo de acesso mais qualquer tempo adicional antes que um segundo acesso possa iniciar. Esse tempo adicional pode ser exigido para a extinção de transientes nas linhas de sinal ou para a regeneração de dados, se eles forem lidos destrutivamente.

Taxa de transferência: essa é a taxa em que os dados podem ser transferidos para dentro ou fora de uma unidade de memória.

Tecnologias de memória: diversos foram empregados. As mais comuns hoje são memória semicondutora, memória de superfície magnética, usada para disco e fita, e óptica e magneto-óptica.

Características físicas: Em uma memória volátil, a informação se deteriora naturalmente ou se perde quando a energia elétrica é desligada. Em uma memória não volátil, a informação uma vez gravada permanece sem deterioração até que seja deliberadamente mudada; nenhuma energia elétrica é necessária para reter a informação.

Organização: Com organização, queremos dizer o arranjo físico de bits para formar palavras.

A hierarquia de memória

As restrições de projeto sobre a memória de um computador podem ser resumidas por três questões: Quanto? Com que velocidade? Com que custo?

A questão da quantidade é, de certa forma, livre. Se houver capacidade, as aplicações provavelmente serão desenvolvidas para utilizá-la.

A questão da velocidade, de certa forma, é mais fácil de responder. Para conseguir maior desempenho, a memória precisa ser capaz de acompanhar a velocidade do processador. Ou seja, enquanto o processador está executando instruções, não gostaríamos que ele tivesse que parar, aguardando por instruções ou operandos.

A questão final também precisa ser considerada. Para um sistema prático, o custo da memória deve ser razoável em relação a outros componentes.

Como se pode esperar, existe uma relação entre as três características principais da memória, a saber: capacidade, tempo de acesso e custo. Diversas tecnologias são usadas para implementar sistemas de memória e, por meio desse espectro de tecnologias, existem as seguintes relações:

- Tempo de acesso mais rápido, maior custo por bit.
- Maior capacidade, menor custo por bit.
- Maior capacidade, tempo de acesso mais lento.

O dilema que o projetista enfrenta é claro. O projetista gostaria de usar tecnologias de memória que oferecessem grande capacidade de memória, tanto porque a capacidade é necessária quanto porque o custo por bit é baixo. Porém, para atender os requisitos de desempenho, ele precisa usar memórias caras, relativamente com menor capacidade e com menores tempos de acesso. Para sair desse dilema, é preciso não contar com um único componente ou tecnologia de memória, mas empregar uma hierarquia de memória. A memória do computador é organizada em uma hierarquia.

No nível mais alto (mais perto do processador), estão os registradores do processador. Em seguida, vêm um ou mais níveis de cache. Quando são usados múltiplos níveis, eles são indicados por L1, L2 e assim por diante. Em seguida, vem a memória principal, que normalmente é uma memória dinâmica de acesso aleatório e dinâmico (DRAM). Todos estes são considerados internos ao sistema de computação. A hierarquia continua com a memória externa, com o próximo nível geralmente sendo um disco rígido fixo, e um ou mais níveis abaixo disso consistindo em mídia removível, como discos ópticos e fita.

Enquanto se desce na hierarquia, ocorre o seguinte:

- a. Diminuição do custo por bit.
- b. Aumento da capacidade.
- c. Aumento do tempo de acesso.
- d. Diminuição na Frequência de acesso à memória pelo computador.

A chave para o sucesso dessa organização é o item “diminuição na frequência de acesso”.

É possível organizar dados pela hierarquia de modo que a percentagem de acessos a cada nível sucessivamente inferior é muito menor que para o nível acima. Considere o exemplo de dois níveis, já apresentado. Suponha que a memória de nível 2 contenha todas as instruções e dados do programa. Os conjuntos atuais podem ser temporariamente colocados no nível 1. De vez em quando, um dos conjuntos no nível 1 terá que ser passado para o nível 2, para dar espaço para um novo conjunto chegando ao nível 1. Porém, na média, a maioria das referências será para instruções e dados contidos no nível 1.

O uso de três níveis explora o fato de que existem diversos tipos de memória semicondutora em diversos tipos, que diferem em velocidade e custo. Os dados são armazenados de forma mais permanente em dispositivos externos, de armazenamento em massa, sendo os mais comuns o disco rígido e a mídia removível, como disco magnético removível, fita e armazenamento óptico. A memória externa, não volátil, também é chamada de memória secundária ou memória auxiliar. Estas são usadas para armazenar arquivos de programa e dados e, normalmente, são visíveis ao programador apenas em termos de arquivos e registros, ao contrário de bytes ou palavras individuais. O disco também é usado para oferecer uma extensão à memória principal, conhecida como memória
virtual.

4.2 Princípios da memória cache

O uso da memória cache visa obter velocidade de memória próxima das memórias mais rápidas que existem e, ao mesmo tempo, disponibilizar uma memória de grande capacidade ao preço de memórias semicondutoras mais baratas. Existe uma memória

principal relativamente grande e lenta junto com a memória cache, menor e mais rápida. A cache contém uma cópia de partes da memória principal.

Quando o processador tenta ler uma palavra da memória, é feita uma verificação para determinar se a palavra está na cache. Se estiver, ela é entregue ao processador. Se não, um bloco da memória principal, consistindo em algum número fixo de palavras, é lido para a cache e depois a palavra é fornecida ao processador. Devido ao fenômeno de localidade de referência, quando um bloco de dados é levado para a cache para satisfazer uma única referência de memória, é provável que haja referências futuras a esse mesmo local da memória ou a outras palavras no mesmo bloco.

4.3 Elementos do projeto da memória cache

Embora haja um grande número de implementações de memória cache, existem alguns elementos básicos de projeto que servem para classificar e diferenciar as arquiteturas de memórias cache.

Endereços de cache

Quase todos os processadores não embutidos, e muitos processadores embutidos, admitem memória virtual. Basicamente, a memória virtual é uma facilidade que permite que os programas endereçarem a memória a partir de um ponto de vista lógico, sem considerar a quantidade de memória principal disponível fisicamente. Quando a memória virtual é usada, os campos de endereço das instruções de máquina contêm endereços virtuais. Para leituras e escritas da memória principal, uma unidade de gerenciamento da memória (MMU, do inglês memory management unit) física traduz cada endereço virtual para um endereço físico na memória principal.

Quando são usados endereços virtuais, o projetista do sistema pode escolher colocar a cache entre o processador e a MMU ou entre a MMU e a memória principal. Uma cache lógica, também conhecida como cache virtual, armazena dados usando endereços virtuais. O processador acessa a cache diretamente, sem passar pela MMU. Uma cache física armazena dados usando endereços físicos da memória principal.

Uma vantagem óbvia da cache lógica é que a velocidade de acesso a ela é maior do que para uma cache física, pois a cache pode responder antes que a MMU realize uma tradução de endereço. A desvantagem é que a maioria dos sistemas de memória virtual fornece, a cada aplicação, o mesmo espaço de endereços de memória virtual. A memória cache, portanto, precisa ser completamente esvaziada a cada troca de contexto de aplicação, ou então bits extras precisam ser adicionados a cada linha da cache para identificar a que espaço de endereço virtual esse endereço se refere.

Tamanho da memória cache

Gostaríamos que o tamanho da cache fosse pequeno o sufi ciente para que o custo médio geral por bit fosse próximo do custo médio da memória principal isolada e grande o sufi ciente para que o tempo de acesso médio geral fosse próximo do tempo de acesso médio da cache isolada.

Função de mapeamento

Como existem menos linhas de cache do que blocos da memória principal, é necessário haver um algoritmo para mapear os blocos da memória principal às linhas de cache. Além do mais, é preciso haver um meio para determinar qual bloco da memória principal atualmente ocupa uma linha da cache. A escolha da função de mapeamento dita como a cache é organizada. Três técnicas podem ser utilizadas: direta, associativa e associativa em conjunto.

MAPEAMENTO DIRETO

A técnica mais simples, conhecida como mapeamento direto, mapeia cada bloco da memória principal a apenas uma linha de cache possível. O efeito desse mapeamento é que os blocos da memória principal são alocados nas linhas da cache. Assim, o uso de uma parte do endereço como o número da linha oferece um mapeamento exclusivo de cada bloco da memória principal à cache. Quando um bloco é armazenado na sua respectiva linha, é necessário marcar os dados para distingui-los de outros blocos que podem ser alocados nessa linha. A técnica de mapeamento direto é simples e pouco dispendiosa para se implementar. Sua principal desvantagem é que existe um local de cache fixo para cada bloco. Assim, se um programa referenciar palavras repetidamente de dois blocos diferentes, mapeados para a mesma linha, então os blocos serão continuamente trocados na cache, e a razão de acerto será baixa (um fenômeno conhecido como thrashing).

MAPEAMENTO ASSOCIATIVO

Permite que cada bloco da memória principal seja carregado em qualquer linha da cache. Nesse caso, a lógica de controle da cache interpreta um endereço de memória simplesmente como um campo Tag e um campo palavra. O campo Tag identifica o bloco da memória principal. Para determinar se um bloco está na cache, a lógica de controle da cache precisa comparar simultaneamente a tag de cada linha. Observe que nenhum campo no endereço corresponde ao número de linhas, de modo que o número de linhas na cache não é determinado pelo formato do endereço. Com o mapeamento associativo, existe flexibilidade em relação a qual bloco substituir quando um novo bloco for lido para a cache.

Os algoritmos de substituição, discutidos mais adiante nesta seção, são projetados para maximizar a razão de acerto. A principal desvantagem do mapeamento associativo é a complexidade do circuito necessário para comparar as tags de todas as linhas da cache em paralelo.

MAPEAMENTO ASSOCIATIVO EM CONJUNTO:

O mapeamento associativo em conjunto é um meio-termo que realça os pontos fortes das técnicas direta e associativa, enquanto reduz suas desvantagens. Neste caso, a cache é uma série de conjuntos, cada um consistindo em uma série de linhas.

Algoritmos de substituição

Uma vez que a cache estiver cheia, e um novo bloco for trazido para a cache, um dos blocos existentes precisa ser substituído. Para o mapeamento direto, existe apenas uma linha possível para qualquer bloco em particular e nenhuma escolha é possível. Para as técnicas associativa e associativa em conjunto, um algoritmo de substituição é

necessário. Diversos algoritmos foram experimentados. Mencionamos quatro dos mais comuns:

- O mais eficaz seja o usado menos recentemente (LRU, do inglês least recently used): substitua aquele bloco no conjunto que permaneceu na cache por mais tempo sem qualquer referência a ele. Devido à sua simplicidade de implementação, LRU é o algoritmo de substituição mais popular.
- Primeiro a entrar, primeiro a sair (FIFO, do inglês fi rst-in-first-out): substitua o bloco no conjunto que esteve na cache por mais tempo.
- Usado menos frequentemente (LFU, do inglês least frequently used): substitua aquele bloco no conjunto que teve menos referências.
- Uma técnica não baseada no uso (ou seja, não LRU, LFU, FIFO ou alguma variante) é escolher uma linha aleatória dentre as linhas candidatas. Estudos de simulação têm mostrado que a substituição aleatória oferece um desempenho apenas ligeiramente inferior a um algoritmo baseado no uso.

Política de escrita

Quando um bloco que está residente na cache estiver para ser substituído, existem dois casos a considerar. Se o bloco antigo na cache não tiver sido alterado, então ele pode ser substituído por um novo bloco sem primeiro atualizar o bloco antigo. Se pelo menos uma operação de escrita tiver sido realizada em uma palavra nessa linha da cache, então a memória principal precisa ser atualizada escrevendo a linha de cache no bloco de memória antes de trazer o novo bloco. Diversas políticas de escrita são possíveis, com escolhas econômicas de desempenho.

write-through: Usando essa técnica, todas as operações de escrita são feita na memória principal e também na cache, garantindo que a memória principal sempre seja válida. Qualquer outro módulo processador-cache pode monitorar o tráfego para a memória principal para manter a consistência dentro de sua própria cache. A principal desvantagem dessa técnica é que ela gera um tráfego de memória considerável e podendo vir a ser um gargalo.

write-back: minimiza as escritas na memória. Com write-back, as atualizações são feitas apenas na cache. Quando ocorre uma atualização, um bit de modificação, ou bit de uso, associado à linha, é marcado. Depois, quando um bloco é substituído, ele é escrito de volta na memória principal se, e somente se, o bit de modificação estiver marcado. O problema com write-back é que partes da memória principal podem ficar inválidas, e daí os acessos pelos módulos de E/S só podem ser permitidos pela cache. Isso exige circuitos complexos e gera um gargalo em potencial.

Observação do barramento com write-through: cada controlador de cache monitora as linhas de endereço para detectar as operações de escrita para a memória por outros mestres de barramento. Se outro mestre escrever em um local na memória compartilhada que também reside na memória cache, o controlador de cache invalida essa entrada da cache.

Transparência do hardware: um hardware adicional é usado para garantir que todas as atualizações na memória principal por meio da cache sejam refletidas em todas as caches. Assim, se um processador modificar uma palavra em sua cache, essa atualização é

escrita na memória principal. Além disso, quaisquer palavras correspondentes em outras caches são semelhantemente atualizadas.

Memória não chacheável: somente uma parte da memória principal é compartilhada por mais de um processador, e esta é designada como não mantida em cache. Nesse tipo de sistema, todos os acessos à memória compartilhada são falhas de cache, pois a memória compartilhada nunca é copiada para a cache.

Tamanho da linha

Outro elemento do projeto é o tamanho da linha. Quando um bloco de dados é recuperado e colocado na cache, não apenas a palavra desejada, mas também algumas palavras adjacentes são armazenadas. À medida que o tamanho do bloco aumenta de tamanhos muito pequenos para maiores, a razão de acerto a princípio aumentará devido ao princípio da localidade, que diz que os dados nas vizinhanças de uma palavra referenciada provavelmente serão referenciados no futuro próximo. À medida que o tamanho do bloco aumenta, dados mais úteis são trazidos para a cache.

Contudo, dois efeitos específicos entram em cena: Blocos maiores reduzem o número de blocos que cabem em uma cache. Como cada busca de bloco escreve sobre o conteúdo antigo da cache, um número pequeno de blocos resulta em dados sendo modificados pouco depois de serem buscados. À medida que o bloco se torna maior, cada palavra adicional fica mais distante da palavra solicitada e, portanto, têm menos probabilidade de ser necessária no futuro próximo.

O relacionamento entre o tamanho do bloco e a razão de acerto é complexo, dependendo das características de localidade de um programa em particular, e nenhum valor ideal definitivo foi encontrado. Um tamanho de 8 a 64 bytes parece ser razoavelmente próximo do ideal.

Número de memórias caches

Quando as memórias caches foram introduzidas originalmente, o sistema de memória típico tinha uma única cache. Mais recentemente, o uso de múltiplas caches tem se tornado comum. Dois aspectos do projeto dizem respeito ao número de níveis de memórias caches e ao uso de caches unificadas ou separadas.

CACHES MULTINÍVEL

À medida que a densidade lógica aumenta, torna-se possível ter uma cache no mesmo chip que o processador: a cache no chip (on chip). Em comparação com uma cache conectada por meio de um barramento externo, a cache no chip reduz a atividade do barramento externo do processador e, portanto, agiliza o tempo de execução e aumenta o desempenho geral do sistema. Quando a instrução e a leitura dos dados são feitos na cache no chip, não existe o acesso ao barramento. Com os caminhos de dados internos ao processador são curtos, em comparação com o tamanho do barramento, os acessos à cache no chip serão feitos mais rápido que os ciclos de barramento com estado zero-wait (tempo de espera nulo).

Dois recursos de projeto moderno de cache para caches multinível merecem ser citados. Primeiro, para uma cache L2 fora do chip, muitos projetos não usam o barramento do sistema como caminho para transferência entre a cache L2 e o processador, mas usam um caminho de dados separado, a fim de reduzir a carga sobre o barramento do sistema. Segundo, com o encolhimento contínuo dos componentes do processador, diversos processadores agora incorporam a cache L2 no chip do processador, melhorando o desempenho.

Com a disponibilidade cada vez maior de área no chip, a maior parte dos microprocessadores modernos passou a cache L2 para dentro do chip processador e acrescentou uma cache L3. Originalmente, a cache L3 era acessível pelo barramento

externo. Mais recentemente, a maioria dos microprocessadores incorporou uma cache L3 no chip. De qualquer forma, parece haver uma vantagem no desempenho para acrescentar um terceiro nível.

CACHES UNIFICADAS VERSUS SEPARADAS

Quando a cache no chip apareceu inicialmente, muitos dos projetos consistiam em uma única cache usada para armazenar referências a dados e instruções. Mais recentemente, tornou-se comum dividir a cache em duas: uma dedicada a instruções e uma dedicada a dados. Essas duas caches existem no mesmo nível, normalmente como duas caches L1. Quando o processador tenta buscar uma instrução da memória principal, ele primeiro consulta a cache L1 de instrução, e quando o processador tenta buscar dados da memória principal, ele primeiro consulta a cache L1 de dados.

Existem duas vantagens em potencial de uma cache unificada:

- Para determinado tamanho de cache, uma cache unificada tem uma taxa de acerto mais alta que as caches divididas, pois ela equilibra a carga entre buscas de instrução e dados automaticamente.
- Somente uma cache precisa ser projetada e implementada.

Apesar dessas vantagens, a tendência é em direção a caches separadas, particularmente para máquinas superescalares, como o Pentium e o PowerPC, que enfatizam a execução de instrução paralela e a pré-busca de instruções futuras. A principal vantagem do projeto de cache separado é que isso elimina a disputa pela cache entre a unidade de busca/decodificação de instrução e a unidade de execução. Isso é importante em qualquer projeto que conta com o pipeline de instruções. Quando a unidade de execução realiza um acesso à memória para carregar e armazenar dados, a solicitação é submetida à cache unificado. Essa disputa pela cache pode diminuir o desempenho, interferindo com o uso eficiente da pipeline de instruções. A estrutura de cache separada contornar essa dificuldade.