**CAPÍTULO 7 – PATTERSON**

7.1 – Introdução

* Criar a ilusão de memória rápida.
* Um programa não acessa todo o seu código ou dados ao mesmo tempo com igual probabilidade.
* Princípio da localidade: Os programas acessam uma parte relativamente pequena de seu espaço de endereçamento em qualquer instante de tempo.
* Localidade temporal: Se um item é referenciado, ele tenderá a ser referenciado novamente em breve.
* Localidade espacial: Se um item é referenciado, os itens cujo endereço estão próximos tenderão a ser referenciados em breve.
* Hierarquia de memória (múltiplos níveis de memória com velocidades e tamanhos distintos) tira proveito do principio da localidade.
* Memória mais rápida = Memória mais cara por bit.
* 3 tecnologias principais para fabricação de memórias: DRAM (memória principal), SRAM(caches) e discos magnéticos.
* Dados são copiados apenas entre 2 níveis adjacentes.
* Bloco: Unidade mínima de informação que pode estar presente ou ausente na hierarquia de dois níveis.
* Acerto, ou hit, os dados requisitados pelo processador aparecem em um bloco de nível superior. O contrário se chama falha, ou miss.
* Quando ocorre um miss, os dados são buscados no nível inferior.
* Taxa de acertos é a fração dos acessos à memória encontrados no nível superior. A taxa de falhas é a porcentagem que sobra, ou seja se de 100 acessos a memória 70 terminaram em hit, os 30 restantes ou 30% é a taxa de falhas.
* Tempo de acerto é o tempo para acessar um item na memória e encontra-lo, o tempo de acesso + a verificação se é acerto.
* Penalidade de falha é o tempo para substituir um item do nível inferior no nível superior mais o tempo de buscar este item no nível inferior.
* O sistema de memória é um fator determinante de desempenho.

7.2 – Princípios básicos de cache

* Mapeamento direto, uma estrutura de cache em que cada local da memória é mapeado extamete para um local na cache, baseando-se no endereço.
* Endereço módulo Número de blocos
* Tag é um campo em uma tabela usado para uma hierarquia de memória que contém informações de endereço necessárias para identificar se o bloco associado na hierarquia corresponde a uma word requisitada.
* A tag contém os bits q n são utilizados como índice.
* Bit de validade em cada linha da cache.

Acessando uma cache

* Words mais recentemente utilizadas substituem words menos recentemente referenciadas.
* O endereço referenciado é divido em : índice de cache, campo de tag.
* Blocos maiores exploram a localidade espacial.
* Aumentar o número de blocos demais causa um aumento nas falhas porque o número de blocos que podem ser armazenados na cache aumentará aumentando a competição pelo bloco.
* Blocos muito grandes demoram mais tempo para serem substituídos, ou seja, aumenta a penalidade de falha.

Tratando as falhas de cache

* A unidade de controle é quem lida com as falhas de cache.
* Etapas quando ocorre uma falha na cache de instruções: Envia o valo do PC original a memória, ler da memória, escreve o resultado na cache, reinicia a execução da instrução que gerou a falha.

Tratando escritas

* Write-through – Quando os dados sempre são escritos na cache e na memória principal, para manter a consistência.
* Usa-se o buffer de escrita para armazenar os dados enquanto estão esperando para serem escritos na memória. Se o buffer estiver cheio o processador para(stall)
* Write back, quando ocorre uma escrita o novo valor é escrito apenas na cache. O bloco modificado é escrito no nível superior apenas quanto for substituído.
* Esquema write back é mais complexo de ser implementado que um write through, mas melhora o desempenho para momentos de rajada.

Uma cache de exemplo: o processador Instrinsity FastMath

* Cache dividida – Separação de caches em dados e instruções. Funcionam em paralelo.

Projetando o sistema de memória para suportar caches

* A velocidade do barramento afeta a penalidade de falha.
* Aumentar a largura de banda da memória: 1) Aumentar a largura da memória e do barramento e da cache. 2) Aumenta a largura da memória, mas não do barramento. Intercalação.
* SDRAMS fornecem uma vantagem sobre as DRAMS, uso de clock elimina necessidade de sincronização, e uso de rajadas elimina a necessidade de especificar endereços seguidos.

7.3 – Medindo e melhorando o desempenho da cache.

* Tempo de CPU =( (ciclos de execução da CPU) +(ciclos de clock de stall de memória))X(Tempo de ciclo de clock)
* Os acertos na cache são classificados no ciclo de execução da CPU.
* Os ciclos de clock de stall de memória provém principalmente das falhas da cahce.
* Ciclos de stall de memória = Ciclos de stall de leituras + de escrita.
* Ciclos de leitura = (# de leituras)/(programa)X(Taxa de falhas de leitura) X Penalidade de falha de leitura.
* Ciclos de escrita = (#escritas)/programa)XTaxa de falha de escritas X Penalidade de falha de escrita + stalls do buffer de escrita.
* Cache write back possuem potenciais stalls extra.
* Ciclos de clock de stall de memória WB = (Acessos)/programaXTaxa de falhasXPenalidade de falhas
* Ciclos de clock de stall WB = Instruções/Programa X falhas/instrução X penalidade de falha
* As penalidades de cache relativas aumentam a medida que o processador se torna mais rápido.
* Menor o CPI, maior o impacto nos stalls.

Reduzindo as falhas de cache com um posicionamento de blocos mais flexível

* Mapeamento direto: Um bloco só pode entrar em um lugar na cache.
* Totalmente associativo, um bloco pode ser posicionado em qualquer lugar na cache. Um bloco de memória pode ser associado a qualquer entrada da cache. Todas as entradas precisam ser pesquisadas para encontrar algo na cache. Para tornar a pesquisa mais rápida é feita com um comparador associado a cada entrada da cache. Os comparadores aumentam o custo do hardware. Só vale a pena para cache muito pequeno número de blocos.
* Associativa por conjunto, existe um número fixo de locais onde cada bloco pode ser colocado. Cada bloco da memória é mapeado para um único conjunto, determinado pelo índice. Um bloco pode ser colocado em qualquer elemento desse conjunto.
* Cache diretamente mapeada = Associativa por conjunto de 1 via.
* Cacha completamente associativa = Associativa por conjunto de n vias.
* Aumentar a associatividade diminui a taxa de falhas, mas aumenta o tempo de acerto.

Localizando um bloco na cache

* O índice na associativa por conjunto serve para selecionar o conjunto.
* Tags são pesquisadas em paralelo num conjunto.
* Cada aumento na associatividade diminui o índice e aumenta a tag.

Escolhendo que bloco substituir

* Cache diretamente mapeada , só há um lugar para guardar um novo bloco.
* Cache associativa por conjunto, é necessário decidir onde colocar o novo bloco.
* Usa-se o LRU.

Reduzindo a penalidade de falha usando cache multiníveis.

* Maioria das vezes a cache vem no mesmo die do processador.
* Usa-se mais de um nível de cache de forma que se ocorrer uma falha na cache primária o tempo para acessar as outras caches será bem menor do que aquele para acessar a memória principal.
* Cache de nível mais alto se concentram em taxa de acertos e cache de nível mais baixo se concentram em recuperação das falhas, penalidade de falhas.
* Cache primária = Tamanho de bloco menor, tamanho menor.
* Cache secundária = Tamanho de bloco maior, tamanho maior.

Entendendo o desempenho dos programas

* Associatividade diminui com a aproximação do processador.