Relatório 1º Laboratório OC

Grupo: 4

Alunos: Francisca Almeida (105901), Patrícia Gameiro (107245), José Frazão (106943)

Responsável: Diogo Miguel Bispo Dinis

---

1 Introdução

Neste laboratório foi proposto simular uma hierarquia de memória cache. O projeto focou-se na implementação de uma hierarquia com até dois níveis de cache, explorando três configurações: uma cache L1 de mapeamento direto, L1 e L2 de mapeamento direto, e uma com L1 de mapeamento direto e L2 associativa de 2 vias.

2 Detalhes de Implementação

2.1 Indexação

Na nossa implementação, utilizámos uma abordagem consistente para as caches L1 e L2 de mapeamento direto, calculando o número de bits necessários para o índice com base no tamanho da cache e do bloco. Para a cache L2 associativa por conjuntos (2 vias), foi necessário ajustar o cálculo dividindo o número de linhas por 2, devido ao seu caráter associativo.
Em ambas as abordagens, utilizamos uma função auxiliar que calcula o logaritmo de base 2 do número de linhas na cache, o que corresponde ao número de bits necessários para o índice.

2.2 Offset

A implementação do offset é uniforme em todas as versões da nossa cache. O processo inicia-se calculando o número de bits necessários para o representar, baseado no tamanho dos blocos de cache. Utilizamos uma função auxiliar para determinar este valor, criando em seguida uma máscara bit a bit que é aplicada ao endereço de memória completo. Esta operação isola eficazmente os bits que representam o offset dentro do bloco. Para garantir o alinhamento adequado com o tamanho das palavras da cache, deslocamos os bits do offset para a direita e depois para a esquerda, utilizando o tamanho da palavra como referência.

2.3 Blocos em Conflito

Para assegurar o bom funcionamento da cache, é essencial gerir os blocos em conflito. A abordagem para a resolução desses conflitos é bastante semelhante nas distintas implementações. Quando ocorre uma falha na cache *(miss)*, o sistema verifica se o bloco está presente, analisando a validade da linha e a *Tag*. Caso o bloco não esteja na cache, o sistema recupera o correto da memória principal. Se a linha da cache for marcada como *dirty*, indicando que foi modificada, seguindo a política de *write-back*, o bloco é escrito de volta na memória antes de ser substituído.
No caso da cache L2 associativa, se o bloco puder ser armazenado em mais de uma via, o sistema precisa decidir qual bloco substituir, nestes casos, utilizamos o *Least Recently Used*.

2.3.1 Write-Back

Nesta abordagem, quando ocorre uma escrita, os dados são atualizados apenas na cache, e o bloco é marcado como *dirty*. A escrita para o nível mais baixo da hierarquia de memória (DRAM) só ocorre quando um bloco *dirty* precisa de ser substituído. Esta estratégia reduz significativamente o tráfego de escrita para os níveis inferiores da hierarquia de memória, melhorando o desempenho global do sistema.

2.3.2 Least Recently Used

O LRU é implementado apenas na cache associativa. Utilizamos o tempo de acesso para determinar qual dos blocos foi usado menos recentemente, atualizando esse valor cada vez que um bloco é acedido. Desta forma, garantimos que os blocos com acesso mais recente permanecem na cache.

2.4 Inicialização e validade da Cache

No início, a cache é sinalizada como não iniciada. A inicialização só ocorre no primeiro acesso a cada cache, onde todas as linhas são marcadas como inválidas. Após isso, a cache é considerada inicializada. No caso da cache L2 associativa, os tempos de acesso das vias vão ser reiniciados.
A validação é contínua e ocorre em cada acesso à cache, verificando se a linha é válida ou não através da flag. Só quando um novo bloco for colocado na cache, é que a linha irá a passar a ser considerada válida. Sempre que a cache é acedida, verifica-se a validade da linha e a tag do bloco. Se houver uma cache *miss*, os dados serão obtidos da memória inferior.

2.5 Vias de Cache

A nossa implementação utiliza diferentes organizações de vias para as caches. As caches de mapeamento direto mapeiam cada endereço de memória para uma única linha na cache, sendo implementadas com um array simples de linhas de cache. Já a cache L2, quando é associativa de 2 vias, é implementada através de um array de estruturas, onde cada estrutura representa um conjunto com duas vias.

2.6 Hierarquia da Memória

A nossa implementação segue uma hierarquia de memória de três níveis: cache L1, cache L2 e DRAM. A cache L1 é o ponto de acesso para todas as operações de leitura e escrita, oferecendo a resposta mais rápida. A cache L2, associativa de 2 vias, atua como intermediária, com maior capacidade que a L1, mas acesso ligeiramente mais lento. A DRAM, no último nível, proporciona a maior capacidade de armazenamento, mas com o tempo de acesso mais elevado.

2.7 Tempo de Acesso à Memória

Utilizamos uma variável global time para simular e contabilizar o tempo de acesso aos diferentes níveis da hierarquia de memória. Os tempos de acesso são definidos como constantes para cada nível, L1_READ_TIME e L1_WRITE_TIME para a cache L1, L2_READ_TIME e L2_WRITE_TIME para a cache L2, e DRAM_READ_TIME e DRAM_WRITE_TIME para a memória principal. A cada operação de leitura ou escrita, o tempo correspondente é adicionado à variável time. Esta abordagem permite-nos simular de forma realista os diferentes tempos de latência associados a cada nível da hierarquia.

3 Testar o Código

Para garantir a correta funcionalidade do nosso simulador, desenvolvemos novos testes, para além dos fornecidos. Estes testes incluem verificações específicas para avaliar aspectos críticos do funcionamento da cache. Um dos testes foca-se na verificação de problemas com o alinhamento de dados e se um bloco é colocado na cache em toda a sua extensão. Outro teste verifica se o programa está protegido contra acessos à memória fora dos limites estabelecidos.

4 Conclusão

A implementação progressiva deste simulador de cache, desde uma cache L1 de mapeamento direto até uma hierarquia com L2 associativa de 2 vias, permitiu-nos explorar os desafios e benefícios de diferentes arquiteturas. A nossa solução, incorporando políticas como write-back e LRU, demonstrou a eficácia de uma hierarquia de memória bem estruturada na redução da latência média de acesso aos dados (AMAT).