Relatório do Segundo Lab:

'Cache Simulator'

Realizado por:
102948 – Alexandre Duarte | 103131 – Joana Marques | 103145 – Nathaniel Prazeres

INTRODUÇÃO

O objetivo deste lab é desenvolver uma hierarquia de memória completa. Para isso, implementamos a memória cache L1 e L2, usando um simulador em C, com acesso a dependências permitidas pela biblioteca glibc (como stdio.h e stdlib.h).

CÓDIGO

4.1. Memória Cache L1 Diretamente Mapeada

Iniciamos o projeto desenvolvendo uma memória cache L1 diretamente mapeada com múltiplas linhas, conforme especificado nas constantes fornecidas. Para isso, copiámos os arquivos SimpleCache.c e SimpleCache.h para L1Cache.c e L1Cache.h, respectivamente, e modificamos o código para acomodar várias linhas em vez de apenas uma.

Começámos por inicializar a cache, e cada linha, individualmente na função *initcache*. Sabendo que é uma cache diretamente mapeada, tivemos que descobrir o *índice*, a *tag* e o *offset* de acordo com os valores da Cache.h. Pois, uma vez tendo o *index*, facilmente descobrimos a que linha temos de aceder. Uma vez feitas estas alterações, não foi necessário alterar mais nada do SimpleCache.c para resolver esta tarefa.

4.2. Memória Cache L2 Diretamente Mapeada

Nesta tarefa, desenvolvemos uma memória cache L2 diretamente mapeada com base nos parâmetros especificados no arquivo de constantes. Integramos essa memória Cache L2 com a Cache L1 desenvolvida na tarefa anterior (4.1).

Visto que a L2 apenas difere da Cache L1 nos valores, não foram necessárias implementações novas face às que realizamos na tarefa anterior. Já no que toca à Cache L1 em si, foi necessário mudar a sua implementação para que em caso de miss, em vez de aceder diretamente à memória aceder à Cache L2.

4.3. Memória Cache L2 de Duas Vias

Na terceira tarefa, alterámos a memória cache L2 desenvolvida anteriormente e transformá-la numa memória cache L2 de 2 vias, mantendo os parâmetros como o valor de L2Size parâmetros inalterados. Nesta configuração, continuamos a utilizar a Cache L1 desenvolvida na tarefa 4.1.

Para esta tarefa, foi necessário modificar a L2 anterior. Os valores index, tag e offset tiveram de ser calculados de maneira diferente, tendo em conta o número de vias definido na constante *WAYS* do ficheiro header *L2Cache2W.h,* sendo que este valor pode ser alterado e o programa mantém a funcionalidade. De seguida acedemos a cada via à procura de um hit. Caso ocorra um cache miss, utilizamos a abordagem de substituição de bloco 'LRU' para descobrir a que via aceder. Esta abordagem utiliza a função *getTime* para conseguir descobrir a linha mais antiga na Cache.

Concluindo, ao longo deste laboratório, realizamos a implementação de uma hierarquia de memória eficaz, que consiste em duas caches, L1 e L2. Modificamos o código fonte fornecido, para adicionar novas funcionalidades e garantir a coerência entre as caches, resultando numa hierarquia de memória que otimiza o desempenho.