Report / Relatório - Grupo 2

Miguel Teixeira 103449

João Ferreira 103680

Rodrigo Alves 103299

4.1 - Directly-Mapped L1 Cache

- Criámos um ficheiro L1Cache.c e L1Cache.h para a Cache L1.
- Na inicialização do Cache L1, usámos um loop para percorrer as linhas de cache (L1_LINES) e definimos o bit “valid” como 0 para cada linha de cache.
Assim, ao iniciar o sistema ou sempre que uma linha de cache for considerada inválida, garante que a cache funciona corretamente, mantém a consistência dos dados e gere eficazmente os dados armazenados. Quando os dados são carregados na cache, o bit "valid" é alterado para 1 para indicar que os dados são válidos e podem ser usados.
- Mudámos a maneira de calcular a tag, index e offset. Na primeira versão, os cálculos de endereço para o cache L1 são baseados em operações de deslocamento. Na nossa versão,
Tag = address / L1_SIZE;
index = (address / BLOCK_SIZE) % (L1_LINES);
offset = address % BLOCK_SIZE;
para funcionar de acordo com o size, block size e o número de linhas de L1.
- Se a linha de cache é válida e se a tag corresponde, é hit e utilizamos memcpy para copiar os dados entre a cache e a memória.
- Ao adicionar várias linhas de cache e associá-las diretamente a blocos de memória específicos, melhoramos o desempenho e a capacidade de armazenamento da cache em comparação com um sistema de mapeamento direto tradicional, que possui apenas uma linha de cache.

4.2 - Directly-Mapped L2 Cache

- Criámos um ficheiro L2Cache.c e L2Cache.h para a Cache L2.
- Com semelhança à Cache L1, porém agora como temos duas camadas de cache, inicializamos as duas, metendo os seus bits "valid" definidos para 0, indicando que nenhuma linha de cache contém dados válidos no início.
- A principal diferença é que, como são duas caches, quando ocorre uma falta de cache (miss) no L1, a função accessL1 agora acessa a cache L2 (accessL2) para buscar os dados da memória principal.

4.3 - 2-Way L2 Cache

- Criámos um ficheiro L2_Associative_Cache.c e L2_Associative_Cache.h.
- Adicionámos novas estruturas de dados ao ficheiro .h, chamadas CacheSet, Cache_L2, para lidar com uma cache L2 associativa, e uma nova variável global associada.
- Para a L2 Cache Associativa, o processo de inicialização é mais complexo devido à natureza associativa da cache. Percorremos todas as linhas da cache L2 (L2_LINES) e, para cada linha, percorremos os conjuntos (sets) (L2_ASSOCIATIVITY) e definimos a validade de cada conjunto como 0. Isso garante que, no início, nenhum dos conjuntos da L2 Cache Associativa contem dados válidos.
- Mudámos os cálculos de endereços do L2 Cache para o L2 Cache Associativo:

```
Tag = address / (L2_SIZE / (BLOCK_SIZE * L2_ASSOCIATIVITY) * BLOCK_SIZE);
index = (address / BLOCK_SIZE) % (L2_SIZE / (BLOCK_SIZE * L2_ASSOCIATIVITY));
offset = address % BLOCK_SIZE;
```

- A principal diferença está na consideração da associatividade da cache, o que envolve dividir o espaço da cache em conjuntos (sets), onde cada conjunto contém várias linhas da cache. O cálculo da Tag e do índice é ajustado de acordo para garantir que o endereço seja mapeado corretamente na cache associativa.
- Em resumo, o que difere da cache L2 é que a cache L2 associativa é mais flexível, pois permite que um bloco de memória seja armazenado em várias localizações na cache, reduzindo as colisões.