Grupo 29

Pedro Sousa: 102664
Fábio Mata: 102802
Nuno Gonçalves: 103392

4.1 - Cache L1 de mapeamento direto

Utilizámos como base o código fornecido para uma cache L1 de mapeamento direto com apenas uma linha.

Relativamente aos acessos à *RAM* (função `accessDRAM`), fizemos uma pequena alteração na validação do endereço recebido como argumento (`if (address >= DRAM_SIZE - WORD_SIZE + 1)`), alterando a constante `WORD_SIZE` para `BLOCK_SIZE`. Sendo que os acessos são feitos por blocos, é necessário ter em consideração o último bloco, e não a última *word*, de forma a garantir que não se ultrapassa o último endereço da memória.

Na função `accessL1`, começamos por percorrer a cache e inicializar todos os *bits* de validade a zero. De seguida determinamos as várias "partes" do endereço: *offset*, *index* e *tag*, assim como o endereço “base” do bloco correspondente (visto que os acessos à *RAM* são feitos por blocos e não *words*). Após obter a linha a que queremos aceder (através do *index*), verificamos se estamos perante um *miss*. Neste caso, traz-se o bloco correspondente da *RAM* para a L1 (para um bloco temporário) e, caso o bloco a ser substituído esteja *dirty*, efetua-se um *write-back* deste último para *RAM* antes de se escrever o novo bloco na cache. Para finalizar o tratamento de um *miss*, alteramos os bits de validade e *dirty* para um e zero, respectivamente, assim como a sua *tag*. Por fim, é efectuado o acesso desejado (*read* ou *write*), tendo este um comportamento genérico visto que, tanto em caso de *hit* como de *miss*, o bloco correspondente já se encontra na cache.

4.2 - Cache L2 de mapeamento direto

Criámos uma struct para cada tipo de cache, ou seja, uma para a L1 e outra para a L2, devido às suas diferenças de dimensão.

Os acessos à L1 comportam-se da mesma forma que na tarefa anterior, excepto quando existe um *miss*. Anteriormente, os *misses* eram resolvidos acedendo à *RAM* (isto é, efetuava-se *fetch* do bloco desejado e *write-back* do bloco substituído se necessário) enquanto que nesta tarefa passam a ser resolvidos na L2, que é o nível imediatamente acima na hierarquia de memória.

Os acessos à L2 são bastante semelhantes ao que é feito na L1. Como a L2 tem uma maior capacidade (e, consequentemente, um maior número de blocos), existe uma diferença no número de *bits* correspondentes à *tag* e ao *index*. Nesta cache, a resolução de *misses* é feita acedendo à *RAM*.

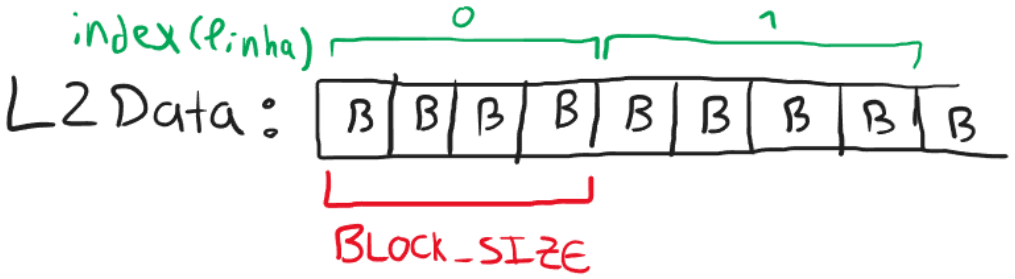
4.3 - Cache L2 *2-way associative* (LRU)

Para representar os *sets* associativos utilizamos uma *struct* que conta com dois campos:

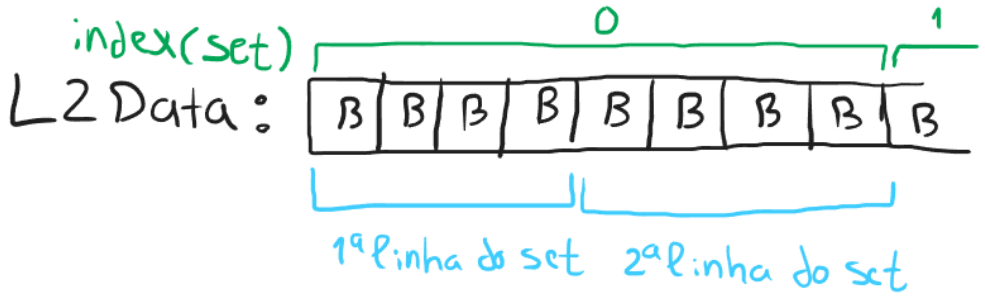
- `CacheLine lines[L2_SET_ASSOCIATIVITY]`: as duas linhas de cache que fazem parte do set (partilham o mesmo índice);
- `uint8_t last_referred`: bit que indica qual das linhas do set foi mais recentemente acedida (0 → 1ª linha, 1 → 2ª linha).

O uso desta nova *struct* implica uma alteração na estrutura da cache L2, que passa agora a guardar *sets* em vez de linhas. O armazenamento dos dados desta cache também sofreu modificações: (nas ilustrações seguintes é utilizado um valor de `BLOCK_SIZE` genérico)

- Enquanto que anteriormente os dados eram organizados da seguinte forma:



- Agora o índice corresponde ao *set* e não à linha, estando as linhas de cada *set* dispostas de forma contígua:



Quanto aos acessos à L2, visto que tanto a capacidade da cache como o tamanho de cada bloco permanece inalterado e passam a existir duas vias de associatividade, o número de *indexes* reduz-se para metade. De forma a lidar com a associatividade, têm que ser feitas verificações adicionais para decidir qual linha do *set* será utilizada (tanto para verificar se é um *hit* como para guardar o novo bloco no caso de um *miss*): **(as condições são verificadas nesta ordem)**

- 1ª linha válida e *tags* iguais? → *hit*, 1ª linha;
- 2ª linha válida e *tags* iguais? → *hit*, 2ª linha;
- 1ª linha inválida? → *miss*, 1ª linha;
- 2ª linha inválida? → *miss*, 2ª linha;
- Decidir com base no *LRU* → *miss*, linha que não é acedida há mais tempo.

Após realizar o acesso, é atualizado o *bit* `last_referred` para referenciar a linha acedida. Todo o restante funcionamento da hierarquia da memória mantém-se igual.