Relatório

4.1 Directly-Mapped L1 Cache

L1Cache.h

Para alterar o código previamente fornecido, foi criado um array de cachelines para ser possível suportar todas as linhas da cache necessárias. Dentro das cachelines criamos um bloco, em que consiste num array com as words do respectivo bloco. Por fim, foi adicionada a constante LINES que representa o número de linhas, dividindo o tamanho de L1 pelo tamanho dos blocos.

L1Cache.c

Inicializamos todas as linhas da cache, calculamos a tag e o MemAddress dando um shift de 6 bits, retirando o offset, como um bloco contém 16 words existirão 64 bytes no total, que é 2^6, logo o offset são 6 bits. O index é obtido através do resto da divisão do MemAddress pelo número de linhas e o offset pelo resto da divisão do address pelo tamanho do bloco.

Por fim ajustamos os antigos memcpy para ser possível passar a word pretendida dos respectivos bloco e cache line.

No final o funcionamento esperado seria a pesquisa do address na cache L1, e caso houvesse um miss o bloco iria ser encontrado na memória RAM e retornado a cache L1 para esta ficar com o bloco correto.

4.2 Directly-Mapped L2 Cache

L2Cache.h

Foi adicionada a constante que guarda o número de linhas da cache L2 e criada uma struct para cada cache, com o respetivo número de linhas. A struct simpleCache foi modificada (nova struct Cache) permitindo o funcionamento das caches L1 e L2 em simultâneo.

L2Cache.c

Criamos a função accessL2 para permitir lidar com os misses em L1 e poder procurar o bloco em L2. Se o bloco não estiver presente em L2, aplicamos a mesma lógica aplicada no ficheiro L1Cache.c e vamos buscar o bloco em falta à memória RAM.

Alterámos também a função initCache para podermos iniciar logo as duas caches.

4.3 2-Way L2 Cache:

2-WayCache.h

Modificámos a struct L2Cache para permitir uma 2 Way Associative Cache criando um array bidimensional. O novo array possui 2 sets, cada um com metade das linhas totais da cache L2.

2-WayCache.c

Ajustamos as chamadas à cache.L2 e a função initCache de forma a integrarem os sets. Para decidir em qual dos sets será colocado o bloco utilizámos a política de LRU, comparando o time das linhas previamente usadas.