RELATÓRIO TRABALHO 1
Organização de Computadores

Grupo: 27 Turno: OC11L10
Integrantes do Grupo: Matheus Alves istid1104683
Rafael Pedro Augusto Garcia istid1106379
André Baltazar dos Santos istid1106919

4.1 Direct-Mapped L1 Cache

A realização do primeiro exercício envolve a implementação de uma Cache com mapeamento direto, seguindo os parâmetros fornecidos no ficheiro Cache.h. Assim, foi preciso alterar os ficheiros fornecidos, que forneciam apenas suporte para uma cache de 1 bloco com 2 words. Para a implementação da cache com mapeamento direto, foi necessário descobrir os bits necessários para o endereçamento na Cache. Dessa maneira, partindo da constante L1 SIZE definida como 256*BLOCK_SIZE, infere-se a necessidade de 8 bits para o endereçamento do index da cache (2^8 posições) e partindo da quantidade de palavras por bloco (BLOCK_SIZE) determina-se 6 bits para o offset (4 bits para a palavra + 2 bits para o byte), com os restantes bits tendo a função de reconhecimento Tag. A partir destes valores, foi alterada a estrutura Cache para suportar múltiplas Cache Lines (Blocos) e foi alterada a estrutura CacheLine para suportar múltiplas words (16 por linha), ambas estruturas no ficheiro L1Cache.h. Na função accessL1 executamos operações sobre o address fornecido pela instrução para isolar os bits do correspondentes ao index, offset e tag, através de operações de shift para “livrar” o address de certos bits e masking para manter apenas aqueles relevantes, utilizando variáveis que podem ser alteradas, fazendo com que o código funcione mesmo com diferentes configurações de cache (Diferentes números de blocos ou palavras por blocos). Então, é realizada a verificação da presença do bloco na cache (Por meio do Valid bit e Tag), e se for um miss, também a verificação do Dirty Bit, para verificar a necessidade de escrever o conteúdo do bloco na memória. No caso de miss, vamos a memória através da função accessDRAM (Já presente no código fornecido) acessar a informação relevante e não presente na cache. Após isso, utilizamos as variáveis que guardam o index e o offset (apenas do word, tendo sido os bits do byte transformados em 0, para realizar a leitura de toda a palavra) para aceder a palavra relevante tanto no MODE_READ quanto no MODE_WRITE(Neste último caso indicando o Dirty Bit a 1, mostrando que a informação neste bloco é assíncrona à memória principal).

4.2 Direct-Mapped L2 Cache

A implementação da cache L2 com mapeamento direto seguiu os mesmos passos da implementação da cache L1. Criou-se a estrutura Cache2, semelhante à Cache1 do exercício anterior, com a única mudança sendo o número de blocos suportados pela cache, que passou a suportar 512 blocos, com cada bloco, igualmente à cache L1, suportando 16 words (cada word tem 4 bytes). De notar, que com a mudança no número de blocos da cache L2, foi necessário aumentar o número de bits do index para 9. Após criar-se a nova estrutura, alterou-se o funcionamento de cache L1, para que quando houvesse um miss na cache L1, em vez de ser ir à memória principal buscar o bloco em falta, a cache L1 procura-se na cache L2 pelo bloco (usando o valid bit e o index e a tag do endereço dado). No código esta implementação fez-se na função accessL1 com a troca da função que acede à memória principal no caso de miss (accessDRAM) para a função que acede à cache L2 (accessL2). Caso o bloco não se encontre na cache L2, a mesma acede à DRAM para obter o bloco em falta , através da função accessDRAM. Com o bloco na cache L2, esta carrega o mesma para a cache L1, implementação dada pela função memcpy do C, copiando para o endereço enviado pela cache L1, o bloco da cache L2. Na mesma nota, alterou-se o funcionamento da escrita de blocos dirty na cache L1 para a DRAM, escrevendo agora o conteúdo bloco na cache L2 e atualizando o bit dirty da cache L2.

4.3 2-Way Set Associative L2 Cache

Para implementação da cache L2 com duas vias de associatividade, criou-se a estrutura CacheSet, que guarda dois blocos (com 16 words cada um) e um número indicando o bloco menos recentemente usado, o LRU. De seguida, alterou-se a estrutura da Cache2 trocando-se o array de 512 blocos de cache para um array de 256 sets de cache. Em relação à função accessL2 e na mesma nota, como o index agora referência 256 sets, em vez de, 512 blocos, diminui-se o número de bits do mesmo para 8. Além disso, como agora existem dois blocos por linha, depois de encontrar o set relativo ao endereço, usando o novo index, verificamos se algum dos blocos é válido e se partilha a mesma tag do endereço. Caso contrário estamos perante um miss, acedemos à memória principal para obter o bloco em falta através do LRU, encontramos o bloco menos recentemente usado pelo bloco em falta, verificamos o dirty bit (caso o dirty bit seja 1 atualizamos na DRAM), e atualizamos tanto o bloco com os novos dados como o LRU. Após atualizar, repete-se o processo da tarefa anterior e transfere-se o novo bloco para a cache L1.