TLB Cache Simulator

Daniel Borges – ist1109474
Duarte Cruz – ist1110181
João Tiago Gama - ist1110728

O sistema operativo utiliza uma Page Table de forma a “traduzir” virtual addresses (VA) em physical addresses (PA). Este processo de tradução é demorado, de modo a que é criada a necessidade de introduzir um Translation Lookaside Buffer (TLB), onde guardamos as traduções que utilizámos recentemente, de forma a ter acesso a um método mais rápido de obter a tradução de páginas recentemente acedidas.

A TLB armazena, para cada entrada, o virtual page number (VPN), o physical page number (PPN) correspondente, entre outros dados que serão posteriormente referidos. Isto permite traduzir um VPN dado, ao compará-lo com o VPN de cada entrada da TLB, retornando o PPN correspondente dessa entrada. É de notar que apenas os page numbers são traduzidos: ao receber um virtual address, é necessário manter o valor e a posição dos bits finais correspondentes ao page offset, e apenas substituir o VPN pelo PPN de modo a obter o physical address.

A TLB criada contém dois níveis de cache (L1 e L2), tendo o primeiro um tempo de acesso e um espaço para armazenamento menor, comparativamente ao segundo, sendo este mais lento e espaçoso. Ambas as caches serão fully associative, não havendo assim index presente no endereço, apenas o page offset e a tag (VPN) para comparar. Considerámos um política inclusiva, ou seja, todos os elementos da cache L1 estão também presentes na L2.

Ao receber um VA, procuramos o VPN na cache L1. No melhor caso, há um hit na cache L1, ou seja, encontramos a tradução correspondente nesta cache. Caso contrário, procuramos na cache L2, e acrescentamos a entrada encontrada na cache L1. Se a tradução não estiver presente em nenhuma das caches, é necessário recorrer à Page Table para obter a tradução, sendo este valor acrescentado à cache L2 e L1, por esta ordem.

É necessário acrescentar as entradas com o VPN e PPN encontrados aos níveis superiores, de forma a que, caso procuremos o mesmo VPN no futuro (fenómeno provável segundo a localidade temporal), este valor já estar presente na TLB, com um tempo de acesso tanto menor como possível.

Ambas as caches são inicializadas com entradas vazias (inválidas). Para acrescentar um valor à cache, é necessário verificar a existência de uma entrada vazia, sendo o valor acrescentado a esta entrada caso exista. Se não houver nenhuma entrada vazia, substituímos pela entrada “Least Recently Used” (LRU), sendo esta a entrada cujo valor não é acedido há mais tempo. Utilizámos o número do acesso atual da respetiva cache para este efeito, pelo que a entrada LRU é a que tem o menor deste valor.

Ao remover um elemento da cache, temos de verificar se o dirty bit correspondente está marcado. Uma entrada é marcada como dirty quando é efetuado uma escrita na página correspondente, visto que, como o valor será escrito na cache e não na memória, o valor na segunda estará desatualizado. Assim sendo, aplicamos a write-back policy, onde, ao remover um valor da TLB, o valor correspondente na memória é atualizado. Isto permite múltiplas modificações na mesma página, enquanto esta estiver na cache, apenas gastando o custo associado a escrever na memória quando esta entrada é removida da TLB. Note-se que o dirty bit é único para cada entrada e para cada cache. Um mesmo valor pode ter o dirty bit marcado na L1, mas não marcado na L2 (se houver uma operação read, seguida de um write, por exemplo).

De modo a invalidar uma página na TLB (processo necessário quando removemos uma página da memória, visto que a tradução armazenada não tem mais significado), basta marcar a entrada correspondente na L1 e na L2 como inválidas (se existirem), seguindo a write-back policy caso o dirty bit da entrada esteja marcado, em qualquer uma das caches.

De modo a desenvolver o sistema descrito anteriormente, fizemos uso das seguintes funções auxiliares, para além das funções já presentes no código base:

- tlb_entry_t* get_entry(tlb_entry[], uint64_t, va_t): obtém entrada da cache dada que tem o VPN fornecido;
- void set_tlb_entry(tlb_entry_t*, va_t, pa_dram_t, uint64_t, bool): coloca os valores dados na entrada fornecida;
- void add_entry_to_tlb(bool, tlb_entry_t*, tlb_entry_t*, va_t, pa_dram_t, uint64_t, bool): adiciona valores novos de uma entrada à TLB. Os valores são adicionados a uma entrada vazia, se houver, ou à LRU, caso contrário. No caso de substituição da LRU, é chamada a função de write back na entrada substituída.
- pa_dram_t search_tlb_l1(va_t, va_t, va_t, op_t, uint64_t, tlb_entry_t**, tlb_entry_t**, bool*): procura entrada com o VPN dado, devolvendo-a se existir. Caso contrário encontra uma entrada vazia (se existir) e a entrada LRU, de forma a se adicionar o valor não encontrado à L1 posteriormente;
- pa_dram_t search_tlb_l2(va_t, va_t, va_t, op_t, uint64_t, tlb_entry_t**, tlb_entry_t**, bool*, bool*): função equivalente para a L2. Também verifica se a entrada encontrada é dirty para se sinalizar na L1.

Também fizemos o seguinte esquema como resumo do funcionamento da nossa solução: