Programming Project #2
Part 4. Extra Credit

Please read the Project2 description file first before reading this document.

For earning a 40% extra credit, you are asked to implement a *Pseudo Least Recently Used (PLRU)* replacement algorithm in your n-way set associative cache simulator, besides the random replacement policy.

Requirements:

1. Implement the PLRU algorithm (as described below) in the n-way set associative cache simulator.
2. Similar as in Part 3, please conduct evaluation and analysis with the PLRU replacement algorithm, including: 1) Given a fixed cache size of 32KB, test the 8-way, 4-way and 2-way set associative cache with cache line size of 16 bytes, 32 bytes, and 128 bytes, respectively. Please report the hit and miss rate in each case.; 2) Given a fixed cache line size of 64 bytes, test the 8-way, 4-way and 2-way set associative cache with the cache size of 16KB, 32KB and 64KB, respectively. Please report the hit and miss rate in each case.

Pseudo LRU (PLRU) implementation [1]:

PLRU is a binary tree-based approximation of the true LRU (Least Recently Used) policy in that the block reference information is maintained in a binary tree, thus reducing the hardware overhead of a true LRU. For an n-way set associative cache, PLRU policy arranges the cache blocks at the leaves of a tree with (n-1) tree nodes pointing to the block to be replaced next. Each node of the tree has a one-bit flag denoting "go left to find a PLRU candidate" (flag bit = 0), or "go right to find a PLRU candidate" (flag bit = 1).

Overall, the binary tree structure should be used to implement the PLRU algorithm in your simulator. In every cache hit, the flag bit should be updated with 1 or 0, corresponding to the position of the hit block. If the cache miss occurs, starting at the root node, the flag values are compared respectively to find the block that should be evicted.

Example: Suppose there is a 4-way set associative cache and block A, B, C and D are 4 blocks of one set. In this manner, 3 tree nodes (flag[0], flag[1] and flag[3]) are needed in the binary tree structure of PLRU, as shown in Figure 1.

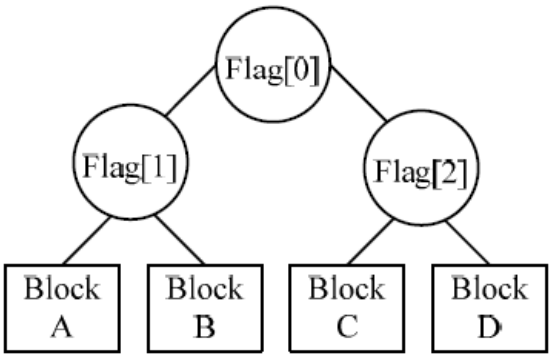

Figure 1. Binary Tree-based PLRU replacement policy for one cache set in a 4-way set associative cache

During the cache access, if block A is hit then the flag[0] and flag[1] are both set to 1. It guides the PLRU algorithm to pick the PLRU candidate from the right subtree of flag[0], as the block A in the left subtree is just referenced. In the case of cache hits, the truth table for updating flag bits in the binary tree is shown in Table 1. If the cache miss occurs, then the PLRU algorithm starts from the root nodes to check the binary value. If flag[0] is 1, it will traverse to flag[2]. Then if flag[2] is 1, it means compared with block C, block D is least recently referenced. Therefore, the block D is evicted. The truth table for selecting replacement candidates of PLRU is shown in the Table 2.

Table 1. Truth table for updating flag bits in the decision binary tree at cache hits.

Which cache block is hit?	Flag[0]	Flag[1]	Flag[2]
Cache Block A	1	1	no change
Cache Block B	1	0	no change
Cache Block C	0	no change	1
Cache Block D	0	no change	0

Table 2. Truth table for selecting replacement candidates based on flag bits in the decision binary tree at cache misses.

Flag[0]	Flag[1]	Flag[2]	Replacement candidate
0	0	0	Cache Block A
0	0	1	Cache Block A
0	1	0	Cache Block B
0	1	1	Cache Block B
1	0	0	Cache Block C
1	0	1	Cache Block D
1	1	0	Cache Block C
1	1	1	Cache Block D

Reference

[1] K. Zhang, Z. Wang, Y. Chen, H. Zhu and X.-H. Sun. PAC-PLRU: A Cache Replacement Policy to Salvage Discarded Predictions from Hardware Prefetchers. In the Proc. of the 11th IEEE/ACM International Symposium on Cluster, Cloud and Grid Computing (CCGrid'11), 2011. http://discl.cs.ttu.edu/lib/exe/fetch.php?media=wiki:docs:ccgrid11.pdf