Programming Project
CS3375 Computer Architecture
Fall 2017

Instructor: Dr. Yong Chen **Office:** EC315 **email:** yong.chen@ttu.edu
Instructor Office Hours: 11 a.m. – 12 p.m., TR, or by appointment
TA: Mr. Dian Li **Office**: EC201A **email**: dian.li@ttu.edu
TA Office Hours: Mon. and Wed., 2 p.m. - 3:30 p.m.

Please read the Programming Project description file first before reading this document.

For earning a 20% extra credit, you are asked to implement a *Pseudo Least Recently Used (PLRU)* and *Random Replacement (RR)* algorithm in simulating the n-way set associative cache, besides the NRU replacement policy.

1. Implement the PLRU algorithm (as described below) in the cache simulator. Test it on n-way set associative cache, where "n" is 4, 8, and 16, and report the cache hit and miss rate.
2. Implement the RR algorithm in the cache simulator. Test it on n-way set associative cache, where "n" is 4, 8, and 16, and report the cache hit and miss rate.
3. Compare the hit and miss rate obtained in the step 1 and 2 with the results of NRU cache replacement algorithm and describe your findings.

Hints for 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 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.

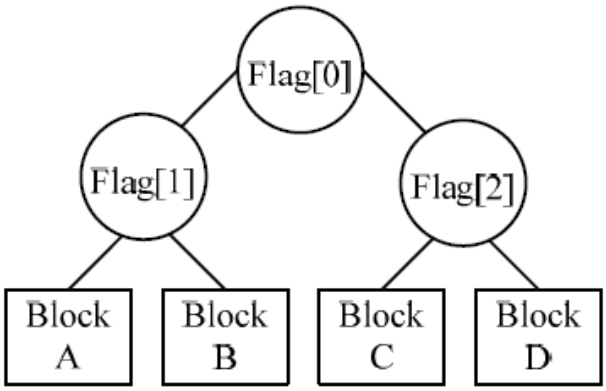

Figure 1. Binary Tree-based PLRU replacement policy for one cache set in a 4-way set associative cache

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

Hints for Random Replacement (RR) algorithm implementation:

When the cache miss occurs and no cache line is available, RR algorithm just randomly selects a candidate block and evict it to make space for the requested block. This algorithm does not require keeping any information about the access history. Due to its simplicity, it has been used in numerous processors, e.g. ARM processors.

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