Tracy Qiu

Professor Tiwari

EECE3324: Computer Architecture Sec 01

22 November 2022

Homework #3

1. Cache block size = 16 bytes
   I. What is the total number of misses (read misses plus write misses) for a direct- mapped cache of size 64 bytes?
   Cache block size = 16 bytes

   # sets = size / (block size * assoc) = 64 / 16 * 1 = 4

   # index bits = log2(# sets) = 2

   # block offset bits = log2(block size) = 4

Address (hex)	Address (binary)	Tag (hex)	Index & Offset bits (binary)	Index (decimal)	Comment
0x04CF	0000010011 001111	0x013	00 1111	0	Miss
0xF3C7	1111001111 000111	0x3CF	00 0111	0	Miss / Repl
0x0423	0000010000 100011	0x010	10 0011	2	Miss
0x0433	0000010000 110011	0x010	11 0011	3	Miss
0x2BC4	0010101111 000100	0x0AF	00 0100	0	Miss / Repl
0x0423	0000010000 100011	0x010	10 0011	2	Hit
0xF3C3	1111001111 000011	0x3CF	00 0011	0	Miss / Repl
0x04BF	0000010010 111111	0x012	11 1111	3	Miss / Repl

   There is a total of 7 misses for a direct mapped cache of size 64 bytes

   II. Total number of “writebacks” of direct blocks for the same cache in (I)? Dirty blocks are the cache blocks which have been updated in the cache, but the updated value has not been propagated to the memory.

   There are 4 writebacks of direct blocks.

   III. Find the smallest size for a direct-mapped cache such that there are no misses besides compulsory misses (i.e., misses that happen on the first access to block).

   #sets = cache size / 16 * 1

#sets * 2^4 = cache size

Total bits = index bits + tag bits + block offset bits

16 = index bits + tag bits + 4

In order to have no misses, there needs to be unique addresses for the address bits.

2^(8 + 4) = 4096.

IV. Find the smallest size for a 2-way set associative cache such that there are no misses besides compulsory misses.
#sets = cache size / 16 * 2
#sets * 2^5 = cache size
5 bit index for unique address so that there are no compulsory misses
2^5 * 2^5 = 1024

2. ISCA 1990: "Improving Direct Mapped Cache Performance by the Addition of a Small Fully Associative Cache and Prefetch Buffers"
https://www.hpl.hp.com/techreports/Compaq-DEC/WRL-TN-14.pdf
   a. What is the basic idea behind victim caches?

   Victim caches are used to improve the performance of caches by removing mapping conflict misses and reducing miss penalty. Victim cache loads a small full-associative cache with the victim of a miss. Victim caching is an improvement from miss caching because in miss caching, when a miss occurs, the data is loaded into the miss cache and the direct mapped cache. Victim caching loads the fully associative cache with the victim line from the direct mapped cache.

   b. Why does this idea work?

   This works because is a miss occurs in the upper cache but hits in the victim cache, instead of a long miss penalty of accessing the off-chip memory, a short one cycle on-chip miss occurs. This is very effective compared to miss cache when there is a lot of duplication in the miss cache. In victim caching, one set of conflicting instructions is stored in the direct mapped cache, and the other is in the victim cache. Throughout the execution of the program, these items trade places and reduces the duplication stored in the cache.

3. Explain what these two very popular program optimizations to reduce cache miss rate are.
   a. Loop interchange
      Loop interchange is changing the order of executing two nested loops. This can be useful to take advantage of the cache when accessing array elements. When the loop

with the greater number of accesses to memory is the inner loop, this takes advantage of the block more likely to have many more consecutive elements so they can all be brought directly to the cache. This optimization would not provide significant speed up in the case that looping interchange would lead to the inner loop to accessing significantly less than the outer loop. Loop interchange can also lead to worse performance in the case of repeatedly accessing and updating a value that is dependent on the value of the outer loop. For example, if there were two nested for loops that looped through values i and j and within the two loops an array indexed to element i was being set to a certain value, looping interchange would rid of the code reuse of i being used to index an array.

b. Loop blocking

   Loop blocking is a loop transformation that takes advantage of spatial and temporal locality of data in nested loops. Loop blocking breaks the memory accessing into smaller chunks instead of traversing large memory domains all at once. This approach reduces redundant cache misses and reduce the miss penalty from the TLB. A case when loop blocking is not optimizing a program is when the chunk is not small enough to fit all the data for the calculation in the cache.

4. “Adaptive Insertion Policies for High Performance Caching” (ISCA 2007)
   https://people.csail.mit.edu/emer/papers/2007.06.isca.dip.pdf
   a. Identify three major insights from this paper.
      The LRU insertion policy (LIP) places the incoming line in the LRU position instead of the MRU position. Placing it in the MRU can be useful to give the line a chance to get a hit when it traverses from MRU to LRU but causes thrashing for programs with working set greater than the cache size. This is shown to protect against thrashing and yields that are close to the optimal rate for applications with a cyclic reference pattern.
      The Bimodal Insertion Policy is an extension and improvement of LIP by adding to the changes of the set. The BIP inserts an incoming line in the MRU position less frequently and this in turn lets it respond to changes and aging of the working set.
      The Dynamic Insertion Policy dynamically chooses between the BIP and LIP. DIP compares the number of misses across the different sets and they derived analytical bounds for DIP as a function of the number of dedicated sets and the per-set variation in misses of the component policies to determine which policy has better performance.

   b. What is the role Set Dueling?
      The Set Dueling is used to implement cost effective dynamic selection between different competing policies. Set Dueling puts asides a certain number of sets in the cache to each of the two replacement policies and chooses the one with fewer misses.