Project 1 Clarification

(Please ignore the FAQ document that was posted in T-square. Follow instructions in the project description and this clarification document.)

0. How do L1 and L2 cache work?

Both L1 and L2 cache act as WBWA. Specifically, each cache (L1 and L2) follows the common logic described below:

-. Upon receiving an access (either read or write), try to find the block within a cache. If found, provide the block and count as a hit.
-. If the block is not found, count it as a cache miss and issue READ access to the lower memory. L1's lower memory is L2, and L2's lower memory is the main memory. Then find an LRU block and replace the LRU block with the new block from the lower memory.
-. If the LRU block is dirty, we write back the LRU block to the lower memory. This means L1 writes back to the L2, and L2 writes back to the main memory.
-. Remember that L2 has to perform prefetch after above steps. First L2 has to see if prefetch condition is met (refer the project spec). If the condition is met, check if prefetch candidate blocks are already in the cache. If some candidates are not in the cache, bring the blocks to the cache, evict and write back LRU blocks if necessary, then set the prefetched blocks LRU.

1. If L1 block writes back to L2, should we make the block in L2 MRU?

Yes, it is just a write operation for L2, so we make the block MRU.

2. How can we implement main memory and data store in the cache?

You don't need to implement main memory and data store in the cache for actual data. You only need to simulate certain bits (e.g., tag and dirty bits) of the cache to see which memory block is stored in the cache. Figuring out the right data structures for this is your job.

3. I see some small numbers in the input trace files. Aren't we supposed to deal with 64 bit addresses in this project?

Yes, they are all 64 bit addresses and the given template code already give you the 64 bit address values in "cache_access()" function.

4. How do we calculate the miss rate of L1 and L2?

MR1 = (L1_WRITE_MISS + L1_READ_MISS) / L1_ACCESSES
MR2 = (L2_READ_MISS) / (L1_READ_MISS + L1_WRITE_MISS)

Note that L2 write miss does not affect to AAT since it only happens when L1 evicts a dirty block and it does not make CPU to wait. For the same reason, we only use the number of L2 read access for MR2 calculation, which is equal to the sum of L1 read miss and L1 write miss.

5. How can I generate a log with the same format with the sample log?

As you can imagine, it is tricky to provide a code that can generate the log since different students use different variables and logic in their implementation. However, here's the set of printf statements that I used in the reference implementation, and I hope this helps you to get the same format of logs. **You should ignore the variable names and the way I calculate actual numbers since they are specific to my implementation. Only refer the formats in printf statements.**

```
fprintf(stderr, "%llu|%c|%llx", sim->m_timestamp, rw, address);
fprintf(stderr, "|%sReadMiss(%llx)", m_name.c_str(), (address >> m_offset_size) << m_offset_size);
fprintf(stderr, "|%sWriteMiss(%llx)", m_name.c_str(), (address >> m_offset_size) << m_offset_size);
fprintf(stderr, "|%sPut[tag=%llx, dirty=%d]", m_name.c_str(), block->m_tag, block->m_dirty);
fprintf(stderr, "|%sEvict(%llx)", m_name.c_str(), victim.GetBlockAddr());
fprintf(stderr, "|%sWB(%llx)", m_name.c_str(), victim.GetBlockAddr());
fprintf(stderr, "|dist=%lld,last_miss_block_addr=%llx,pending_stride=%lld", dist << m_offset_size,
m_last_miss_block_addr << m_offset_size, m_pending_stride << m_offset_size);
fprintf(stderr, "|Prefetch(%llx)", prefetch_addr);
fprintf(stderr, "|%sEvict(%llx)", m_name.c_str(), victim.GetBlockAddr());
fprintf(stderr, "|%sWB(%llx)", m_name.c_str(), victim.GetBlockAddr());
fprintf(stderr, "|%sPut[tag=%llx, dirty=%d]", m_name.c_str(), block->m_tag, block->m_dirty);
fprintf(stderr, "|%sWriteHit(%llx)", m_name.c_str(), block->GetBlockAddr());
fprintf(stderr, "|%sReadHit(%llx)", m_name.c_str(), block->GetBlockAddr());
fprintf(stderr, "|PrefetchSuccess(%llx)", block->GetBlockAddr());
```