SAMUEL CHEUNG

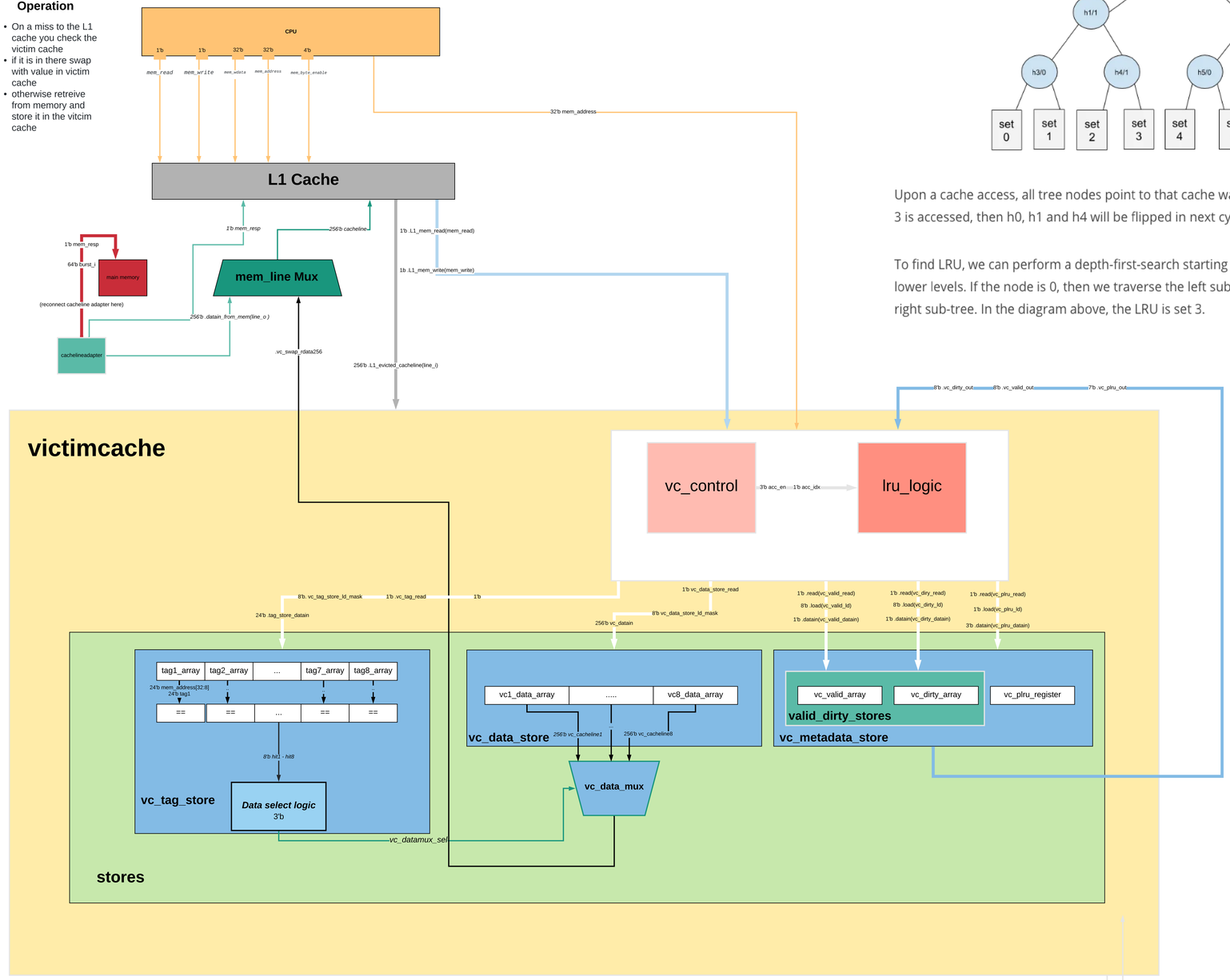
Victim CACHE

Description

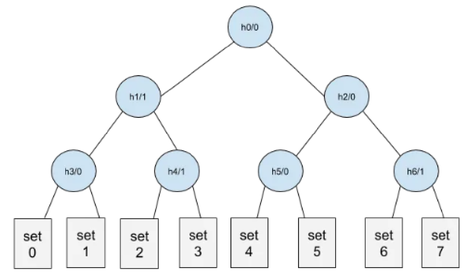
- fully associative cache betweeen L1 and memory meant to reduce conflict misses by storing evicted lines that may be accessed later in the future (temporal locality)
- Write-back with a write allocate policy
- Pseudo-LRU replacement policy
- Read/Write hits must take exactly two clock cycles to complete
- Set/index bits must come from the address bits directly adjacent to the offset bits (I chose to use the format tag/index/word
- Psudo lru uses #ways -1 bits
- "note, the numeric programs used in this study used unit stride access patterns. Numeric programs with non-unit stride and muted stride access patterns also need to be simulated."
- this assumes that all evicteed l1 lines are writeten back and to the vitctim cache

Operation

- On a miss to the L1 cache you check the victim cache
- if it is in there swap with value in victim cache
- otherwise retreive from memory and store it in the vitcim cache



An example is shown below. The data structure is similar to a binary tree. For N ways in a cache set, we need to keep (N – 1) bits. If there are 8 sets in a way, then we will need 7 bits.



Upon a cache access, all tree nodes point to that cache way will be flipped. For example, if set 3 is accessed, then h0, h1 and h4 will be flipped in next cycle.

To find LRU, we can perform a depth-first-search starting from h0, and traverse nodes in lower levels. If the node is 0, then we traverse the left sub-tree; otherwise, we traverse the right sub-tree. In the diagram above, the LRU is set 3.

References

1. (Figure3-4) Jouppi Improving Direct-mapped Cache Performance by the Addition of a Small Fully-Associative Cache and Prefetch Buffers

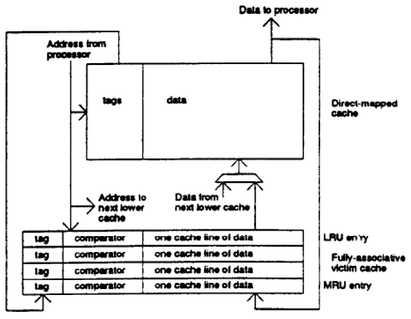


Figure 3-4: Victim cache organization