Lecture 17 (Cache Optimizations)

1 Pipelined Cache

1.1 Tasks Involved - Read

1. Tag array access || Data array access
2. Tag comparison
3. Data Block selection

1.2 Tasks Involved - Write

1. Access tag array
2. Tag comparison
3. Data write

2 Non-Blocking Cache

1. Cache miss should not stall later accesses
2. Miss Status Holding Register is used
3. Maintains read/write bit, word address, tag, store value
4. Miss queue is maintained in FIFO order
5. Can minimize the number of miss requests sent to lower levels

3 Skewed Associative Cache

4 Cuckoo hashes are used

3.1 Cuckoo Hashing

1. Have two hash functions
2. Check either of the two locations for lookup or deletion
3. For insertion, if collision, remove old entry and insert it in the other one

4 Way Prediction

1. Reading all tags and comparing is inefficient
2. Predict the way
3. Compare that first
4. Else search all

4.1 Technique

1. Steps: read registers, compute address, check for LSQ forwarding, access d-cache
2. Meanwhile compute $r1 \oplus 12$ (instruction = ld r2, 12[r1]) and get way via the way predictor
3. Prediction available when instruction at d-cache

5 Loop Tiling

pappu code for matrix multiplication; this is used 99% of the time

1. Issue with row-major vs column-major - row better for first matrix, column for second matrix
2. Can instead have tiling in the loop
3. Smaller blocks are multiplied
4. Can store these rectangles in cache

6 Virtually Indexed Physically Tagged (VIPT) Caches

1. We have been ignoring address translation
2. Can use the offset for index and byte offset
3. Page ID and frame ID are like the tag, so this translation can be done in parallel
4. This limits the number of sets in cache