**Caches (Advanced):**

🡪Cache Performance measured using AMAT (Average Memory Access Time)

AMAT = (Hit Rate x Hit Time) + (Miss Rate x Miss Time)

1. Hit – desired item is in the cache
2. Miss – desired item is not in the cache; need to request from next level (e.g.: DRAM)
3. Hit Rate – percentage of all cache accesses that are hits
4. Miss Rate – percentage of all cache accesses that are misses (1 - Hit Rate)
5. Hit Time – Time taken to determine item is in the cache & return cache data
6. Miss Time/Miss Penalty – Time taken to get item from next level of memory hierarchy (e.g.: DRAM)

**AMAT = (Hit Rate x Hit Time) + (Miss Rate x Miss Time)**

Here the assumption is that the Miss time accounts for both the time to determine item was not in the cache (a miss) plus the time taken to get the desired item from the next level of memory hierarchy.

**Cache Replacement Policies:**

🡪Set is full

🡪Miss🡪need to put new block in set

🡪Which block do we kick out?

a)Random

b)FIFO (Older one kicked out – age bits?)

c)LRU – Least Recently Used

**Implementing LRU:**

🡪Counter-based LRU implementation

🡪E.g.: lets say we have 1 set & a 4-way associative cache which means each set has 4 ways or blocks.

🡪Each block has an individual counter. That means 4 blocks will have 4 counters. We will need four 2-bit counters to cover each block in a set.

a)Way0(Block0) requires one 2-bit counter [Set0]

b)Way1(Block1) requires one 2-bit counter [Set0]

c)Way2(Block2) requires one 2-bit counter [Set 0]

d)Way3(Block3) requires one 2-bit counter [Set 0]

So, to conclude on decoding of counters needed for a n-way set associative cache= log(N) – bit counters/way => if n=4 => four 2-bit counters/way.