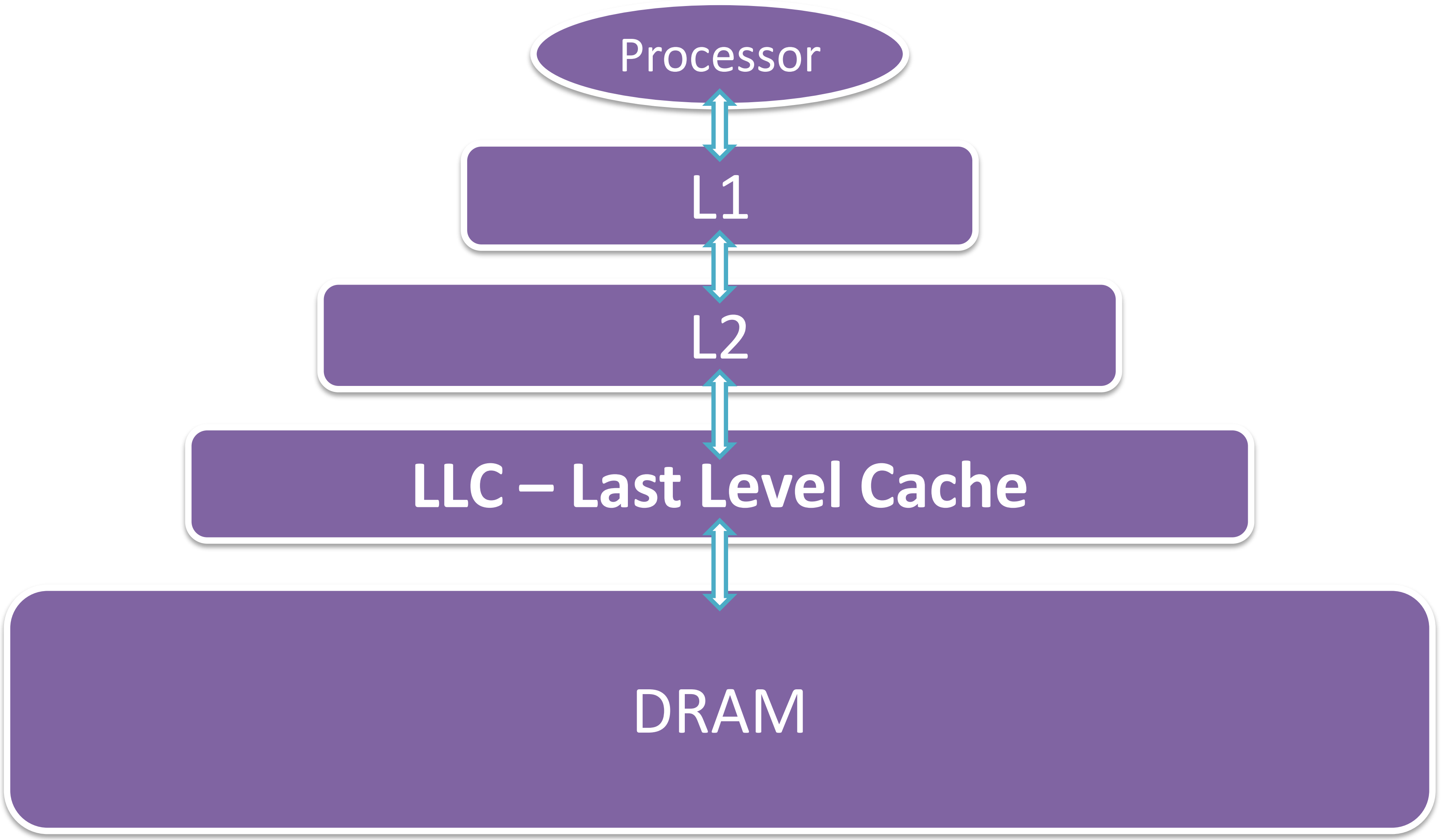
Perceptron Learning Driven Coherence Aware Reuse Prediction for Last-level Caches

Snehil Verma

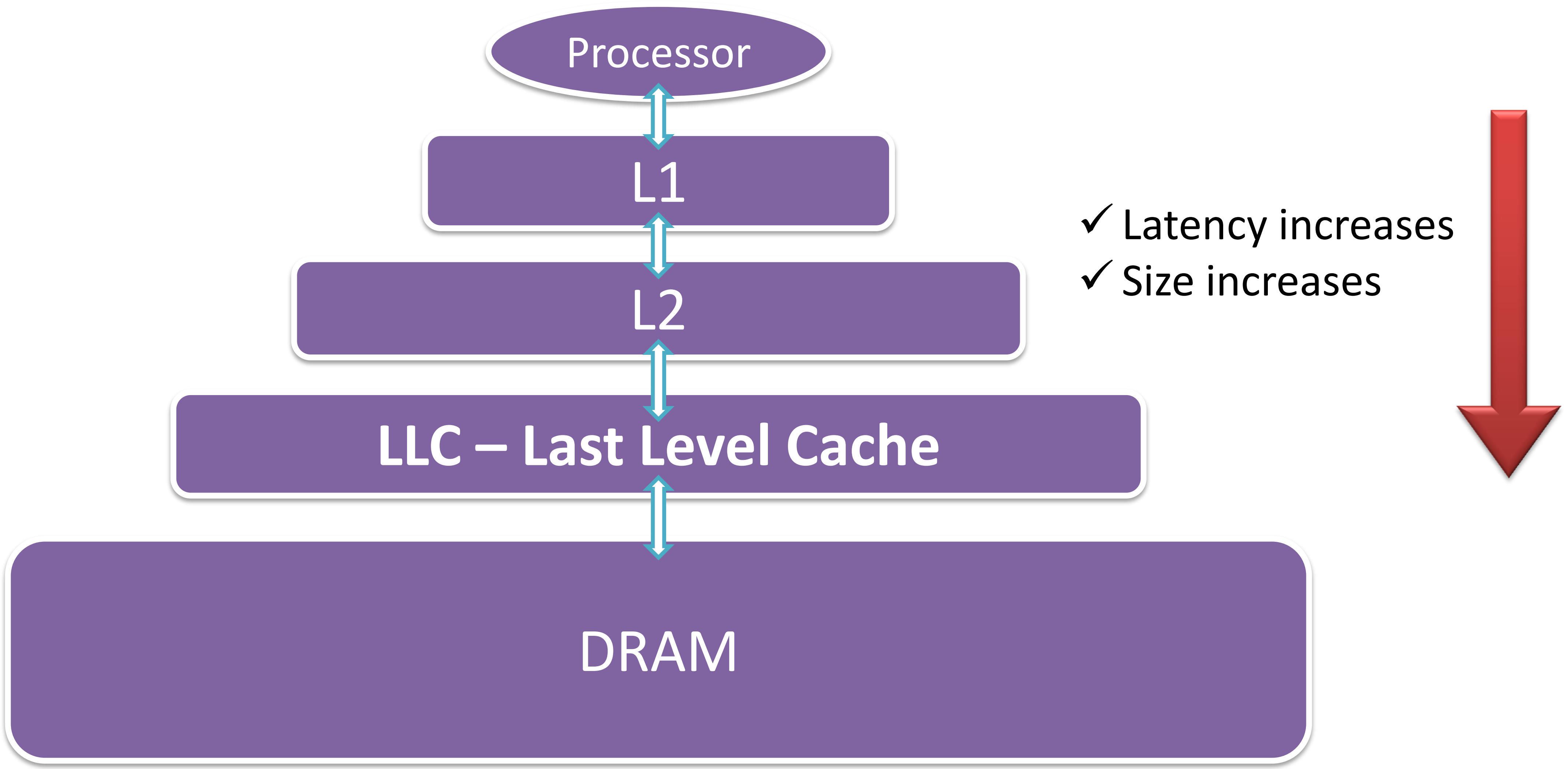
Advisor: **Dr. Biswabandan Panda**

Perceptron Learning Driven Coherence Aware Reuse Prediction for **Last-level Caches**

Cache Hierarchy with DRAM



Cache Hierarchy with DRAM



- ✓ Latency increases
- ✓ Size increases

Replacement in Last Level Cache (LLC)

Goal: To minimize off-chip DRAM accesses !!

Replacement in Last Level Cache (LLC)

Goal: To minimize off-chip DRAM accesses !!

✓ Preserve important blocks

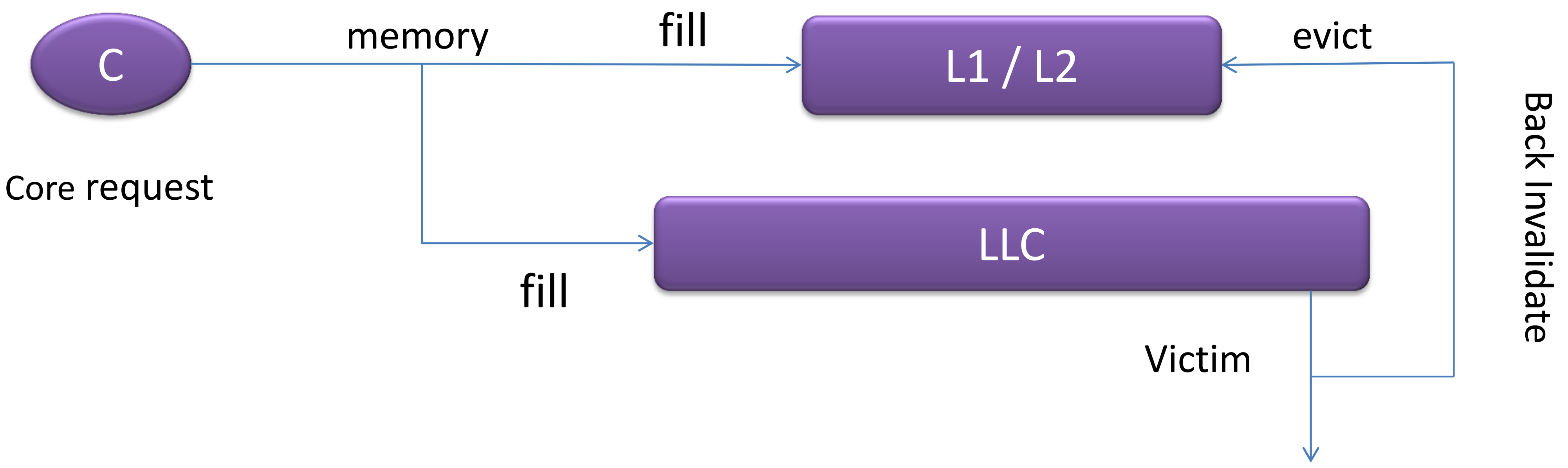
Replacement in Last Level Cache (LLC)

Goal: To minimize off-chip DRAM accesses !!

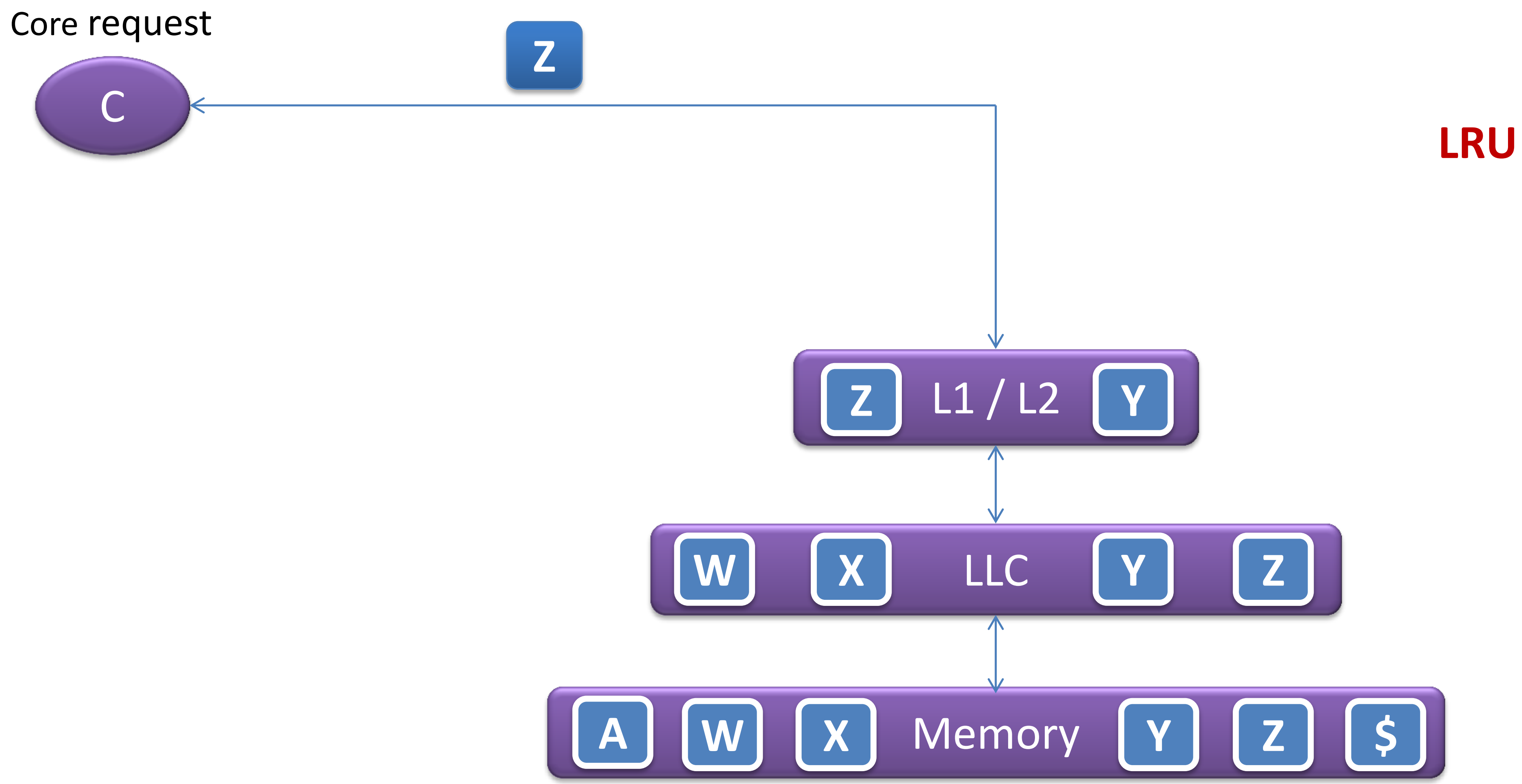
- ✓ Preserve important blocks
- ✓ Important blocks → More reused blocks
  (enjoy more hits)

Example: Inclusive Cache

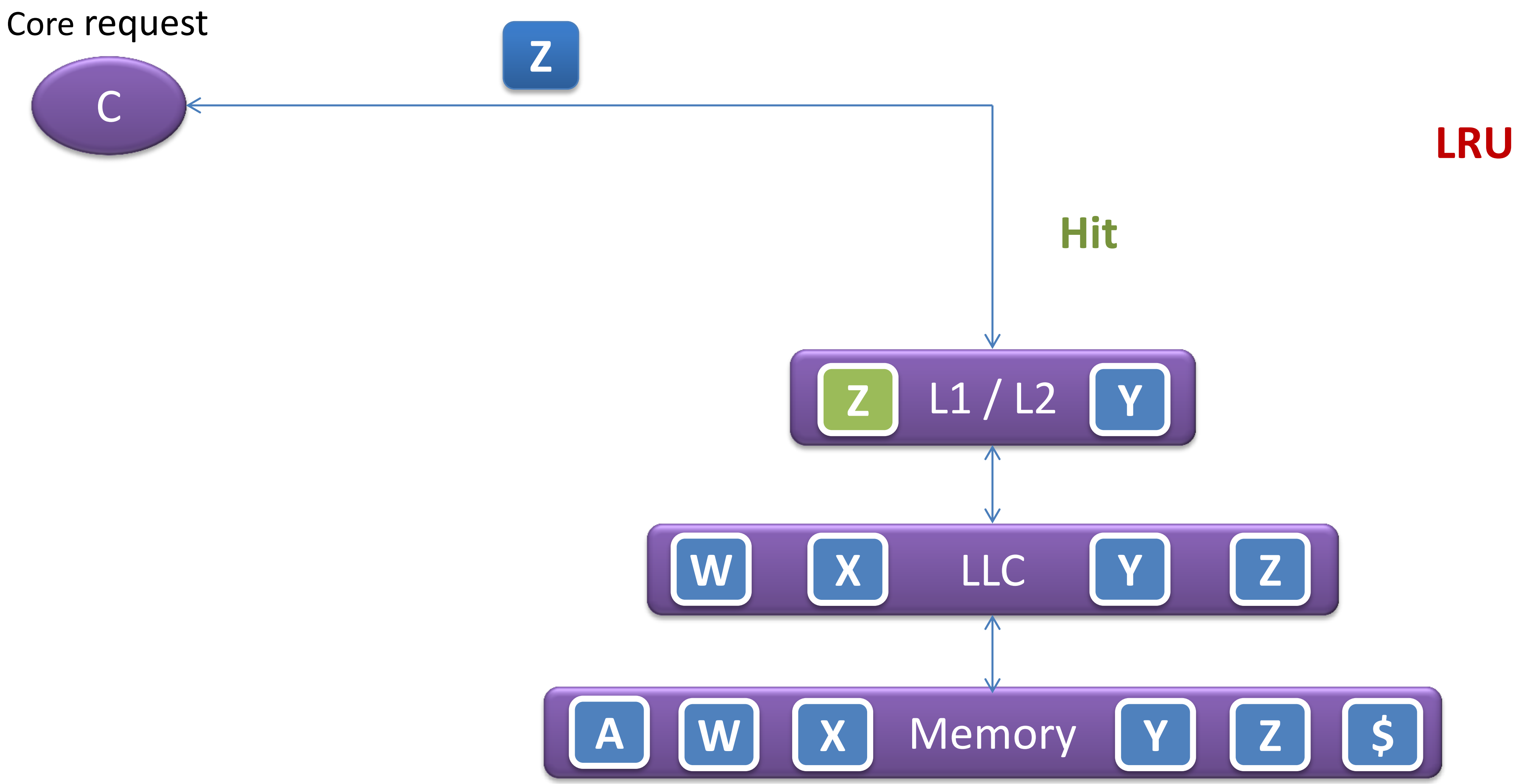
Requested data not in cache



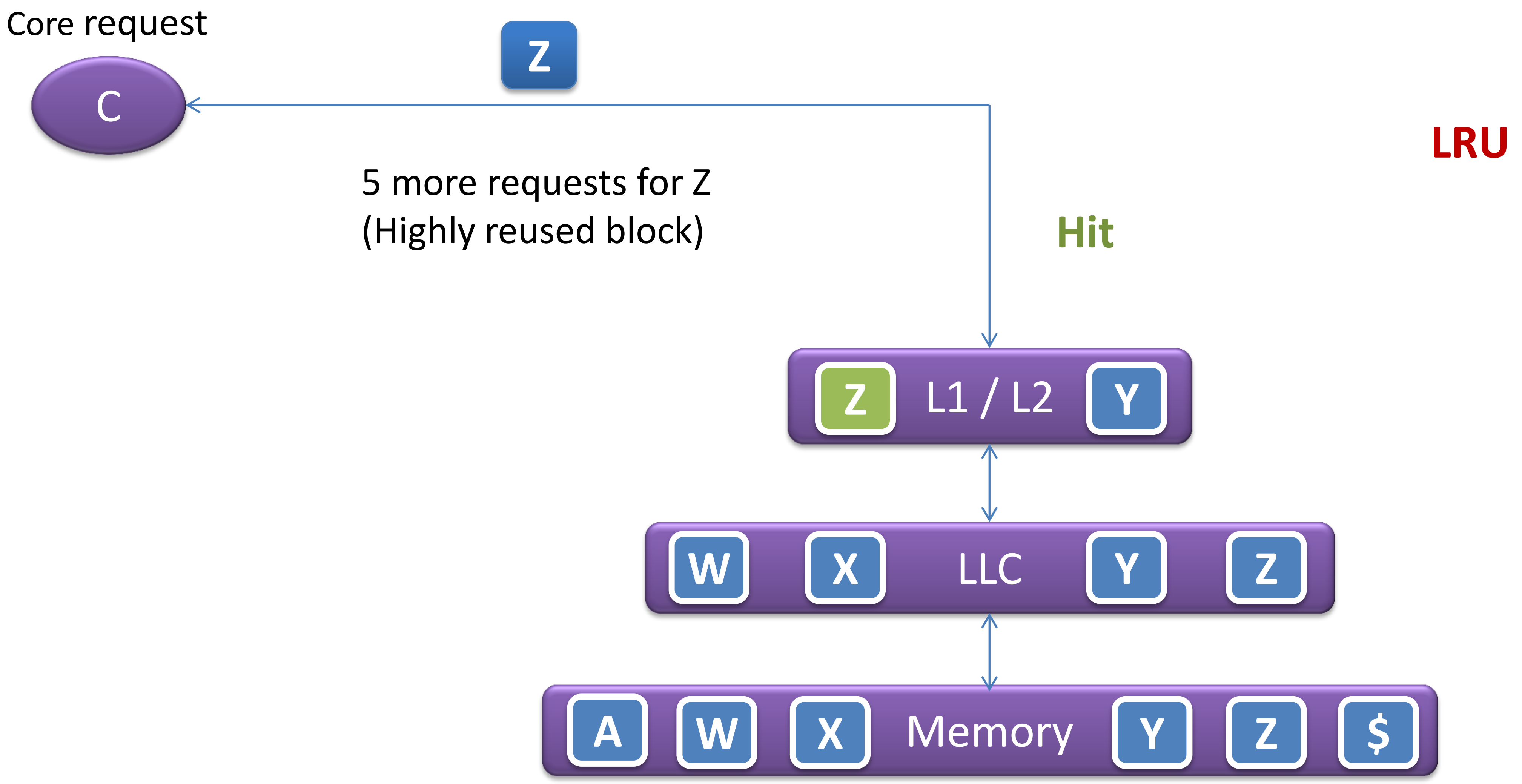
Example: Inclusive Cache



Example: Inclusive Cache

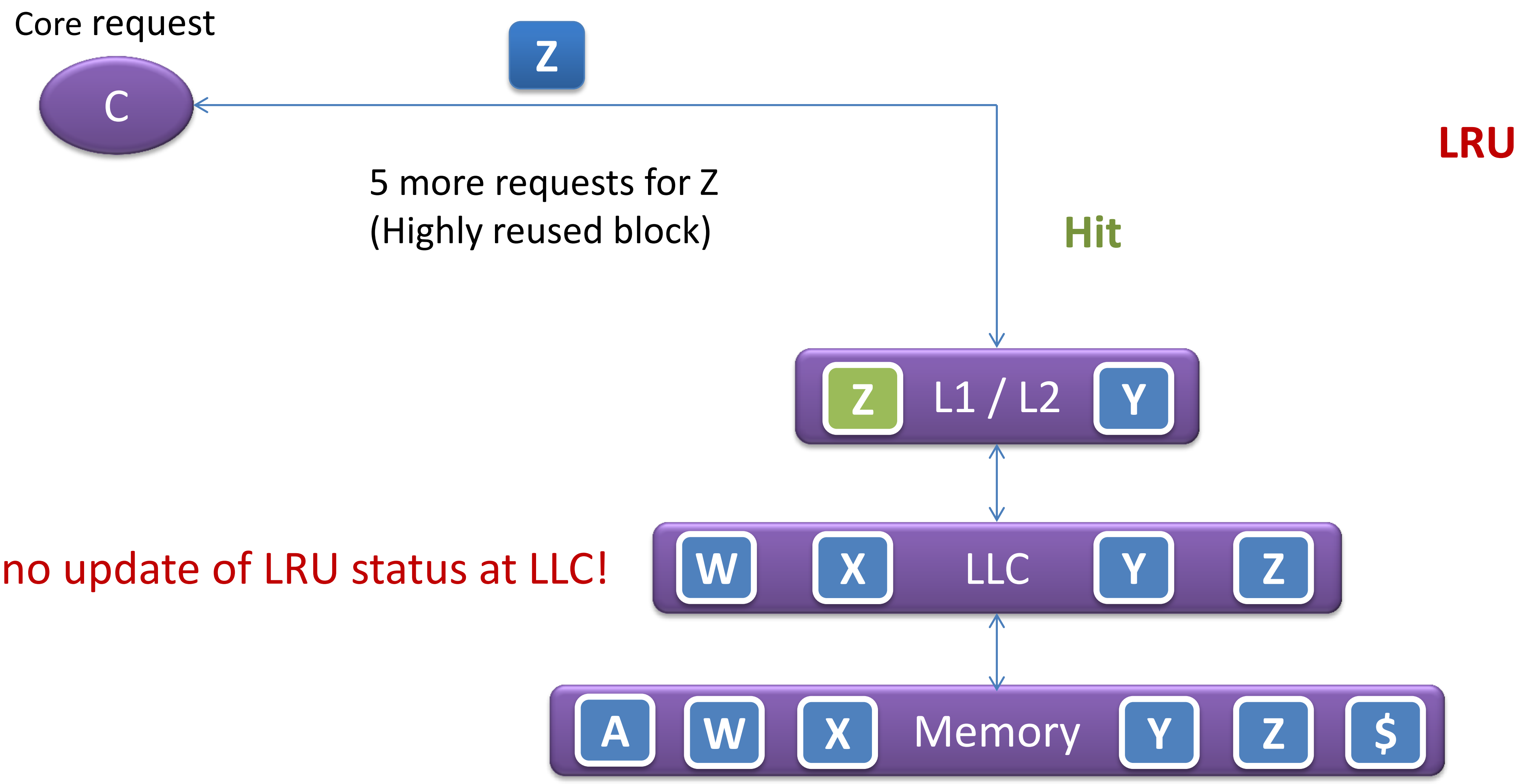


Example: Inclusive Cache



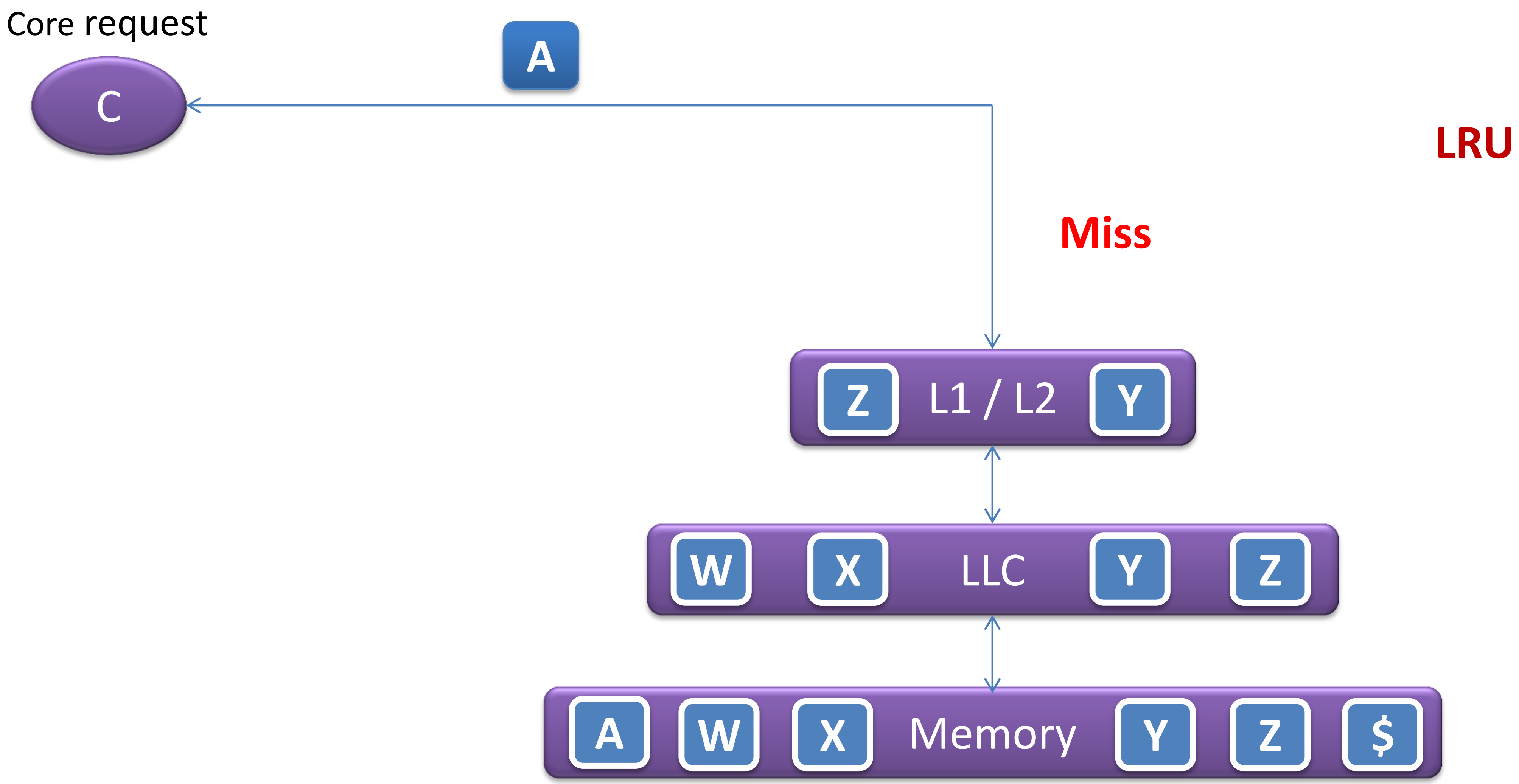
5 more requests for Z
(Highly reused block)

Example: Inclusive Cache

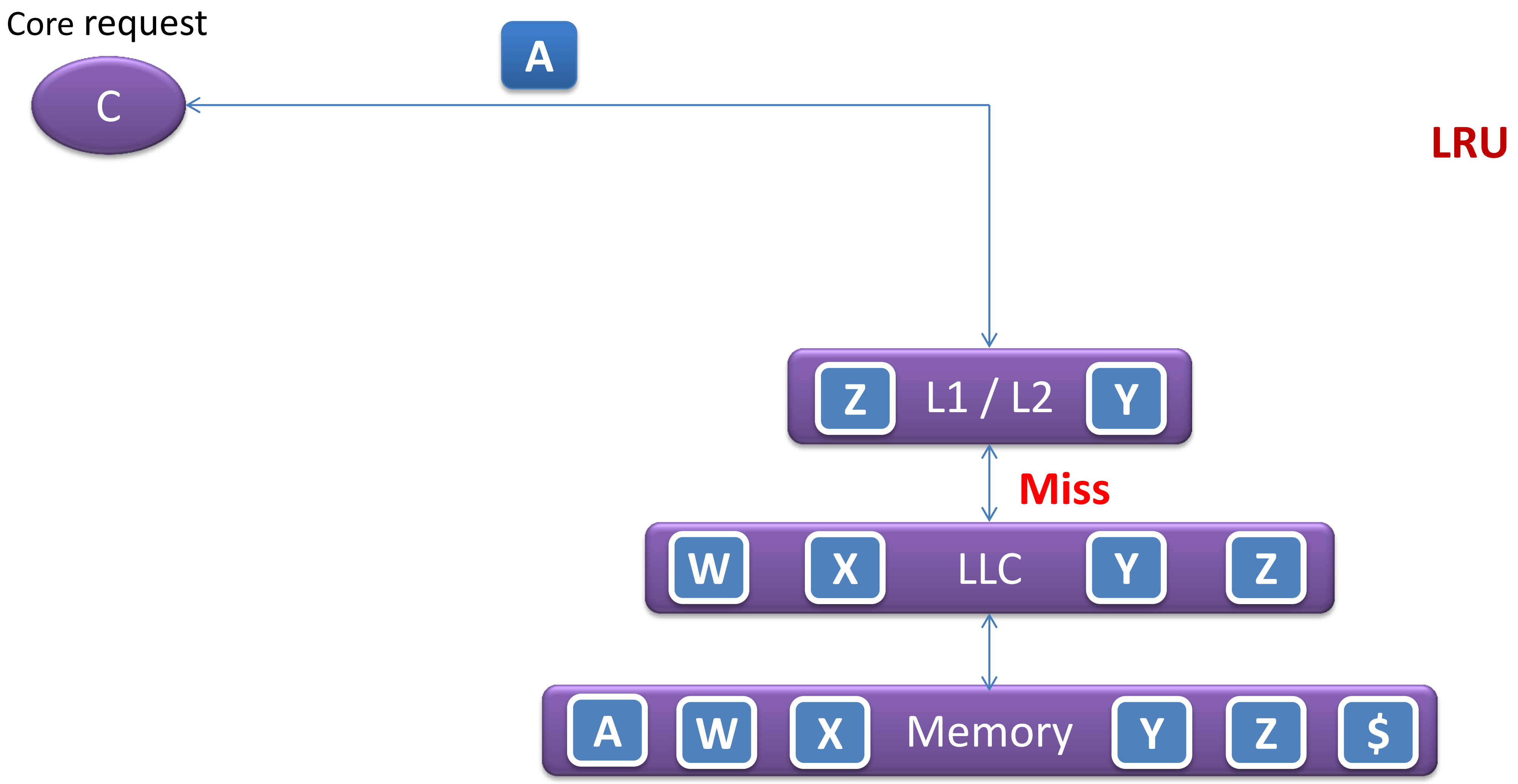


But no update of LRU status at LLC!

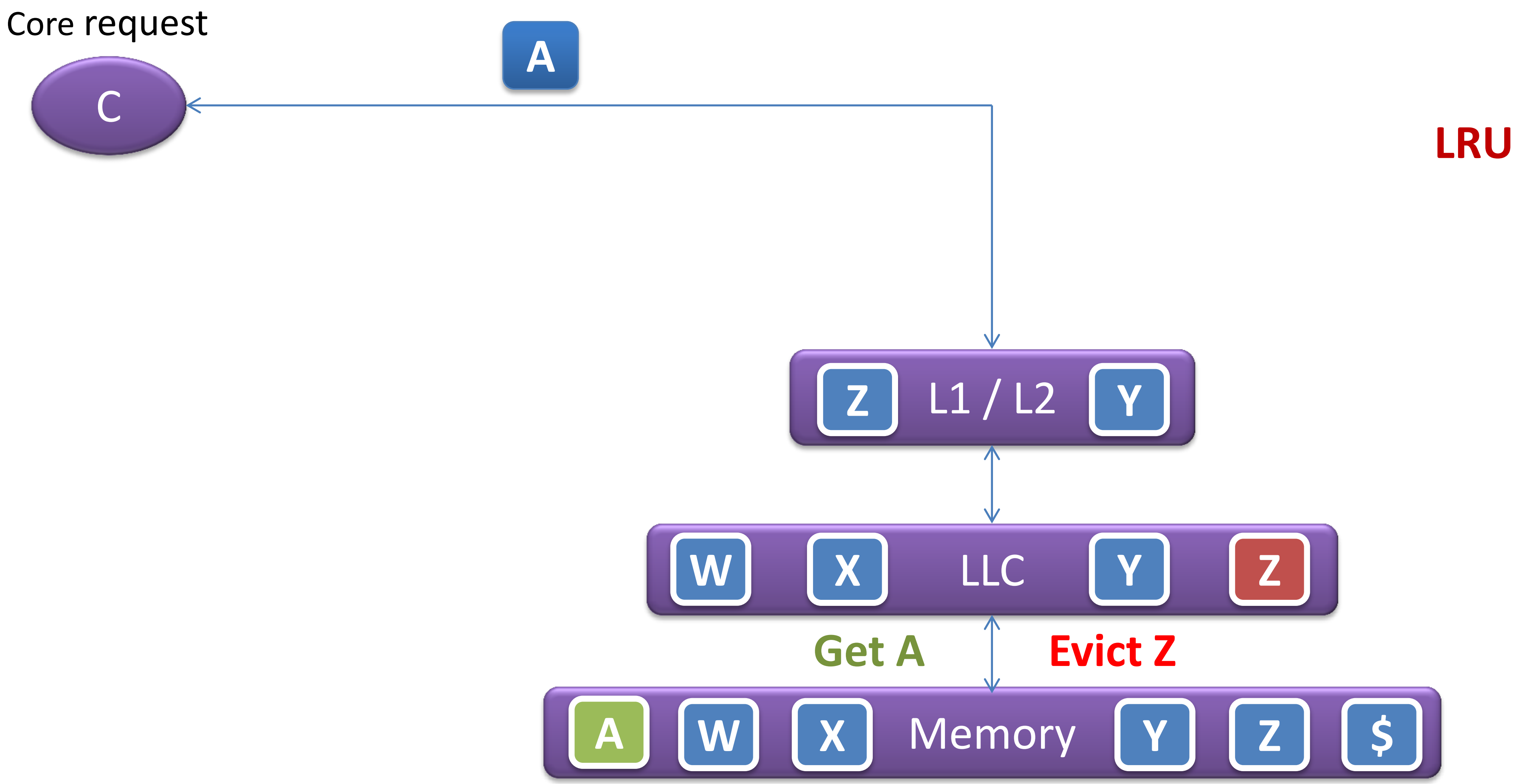
Example: Inclusive Cache



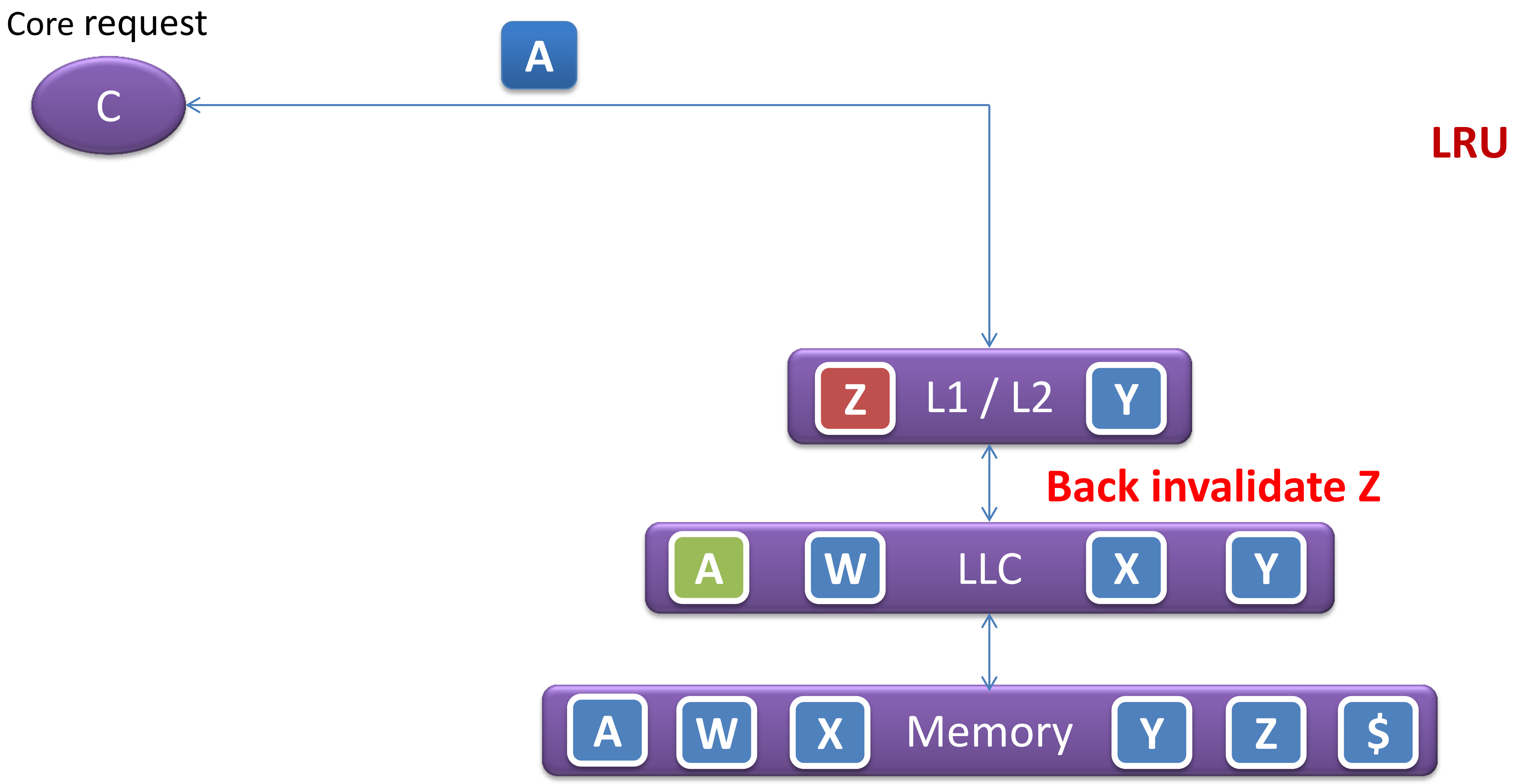
Example: Inclusive Cache



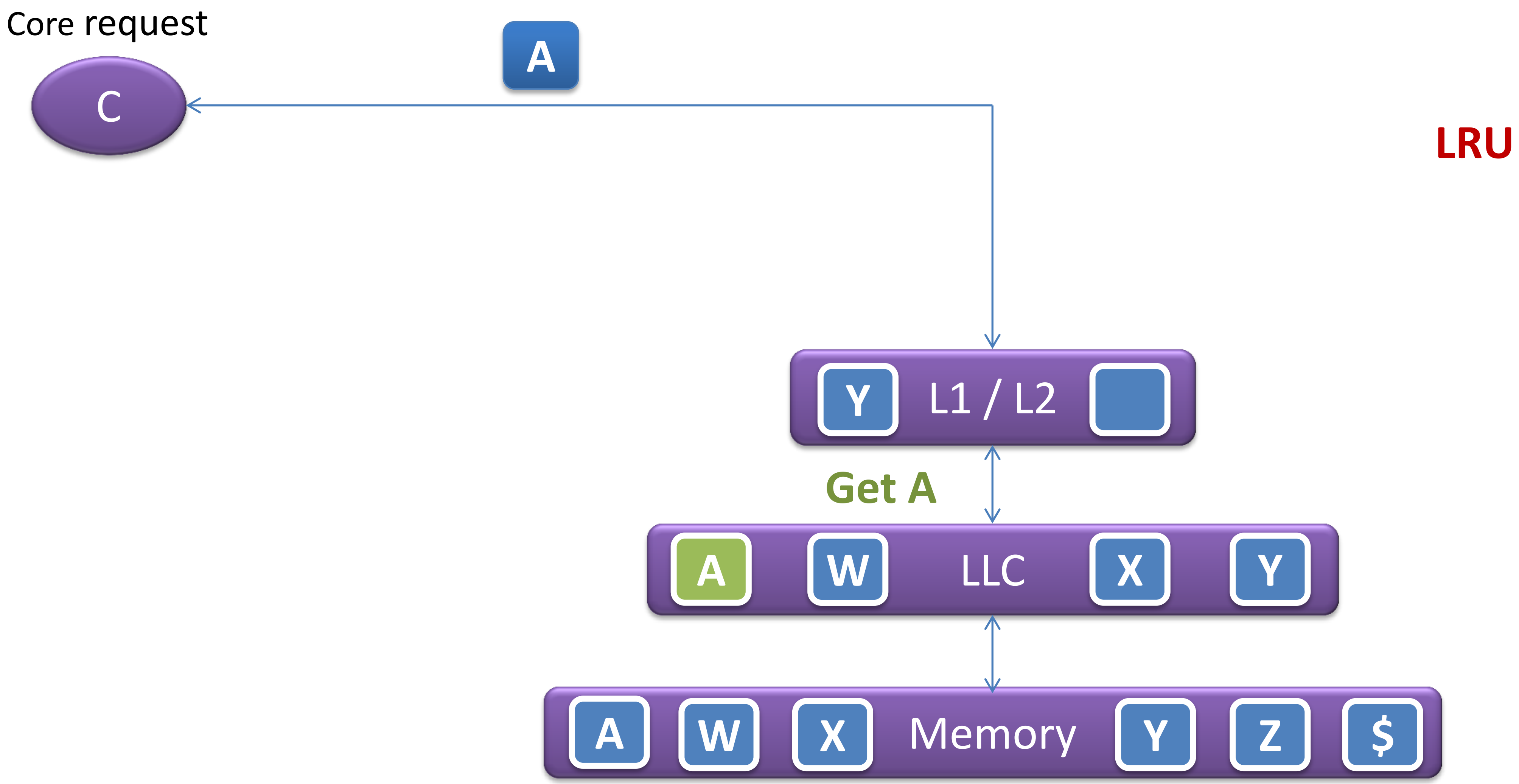
Example: Inclusive Cache



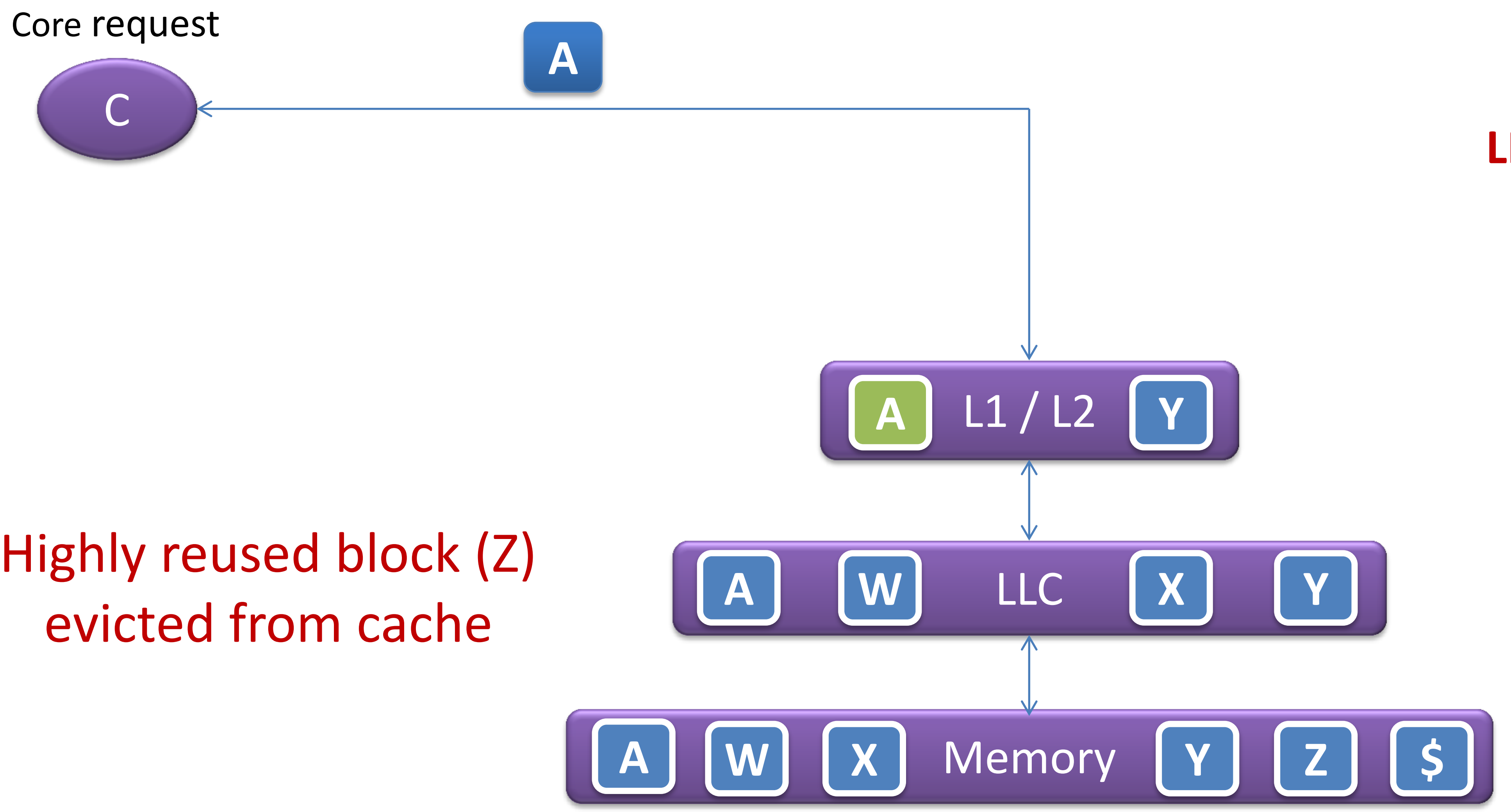
Example: Inclusive Cache



Example: Inclusive Cache



Example: Inclusive Cache



LRU

Highly reused block (Z) evicted from cache

So, what can be done?

Perceptron Learning Driven Coherence Aware **Reuse Prediction** for Last-level Caches

Which blocks have high reuse ?

Parallel Processing

Parallel Processing

Multi-threaded Applications

Parallel Processing

Multi-threaded Applications

Sharing data b/w multiple Cores

Parallel Processing

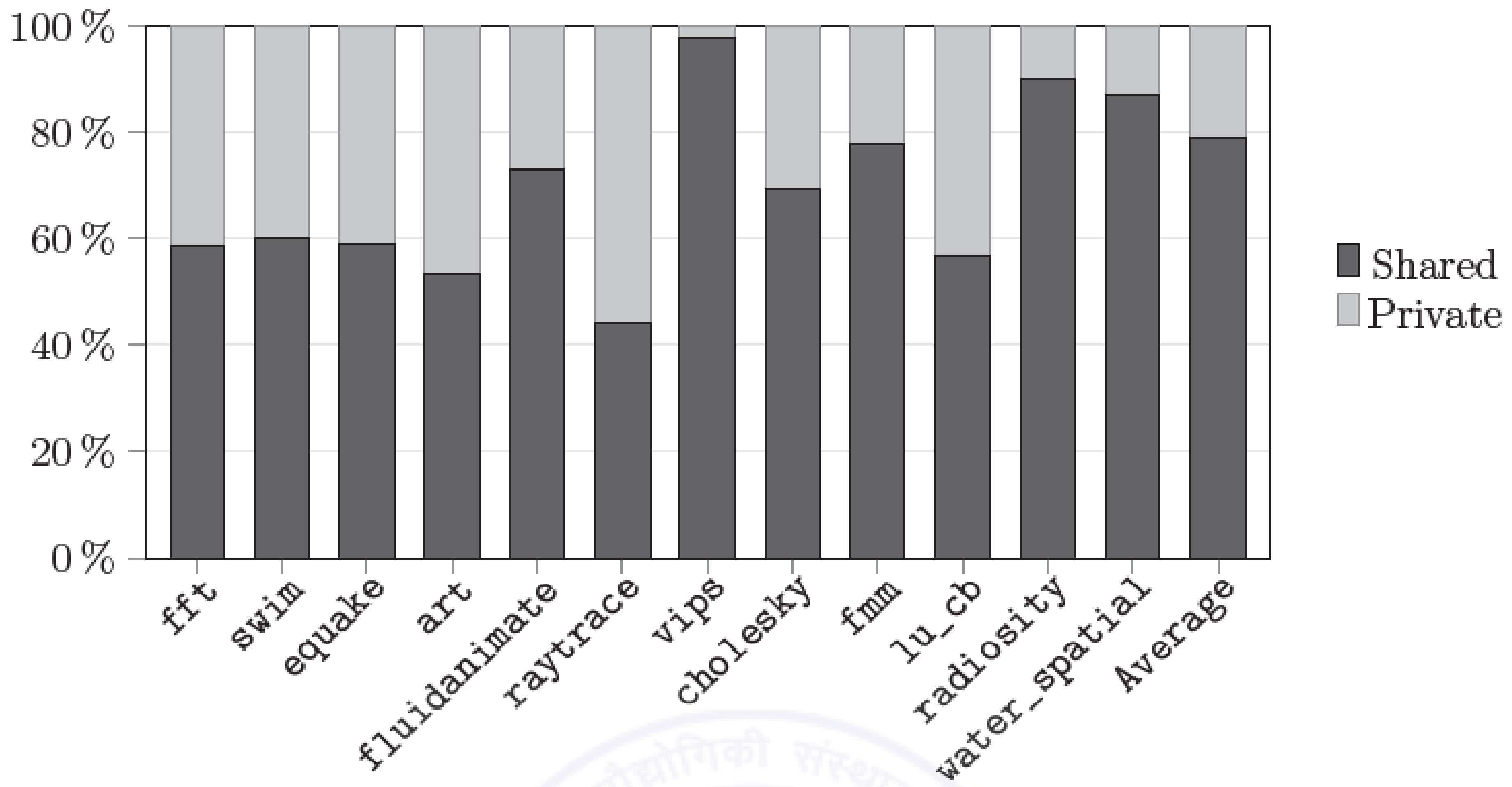
Multi-threaded Applications

Sharing data b/w multiple Cores

Shared LLC

Private L1/L2

Shared Blocks in LLC

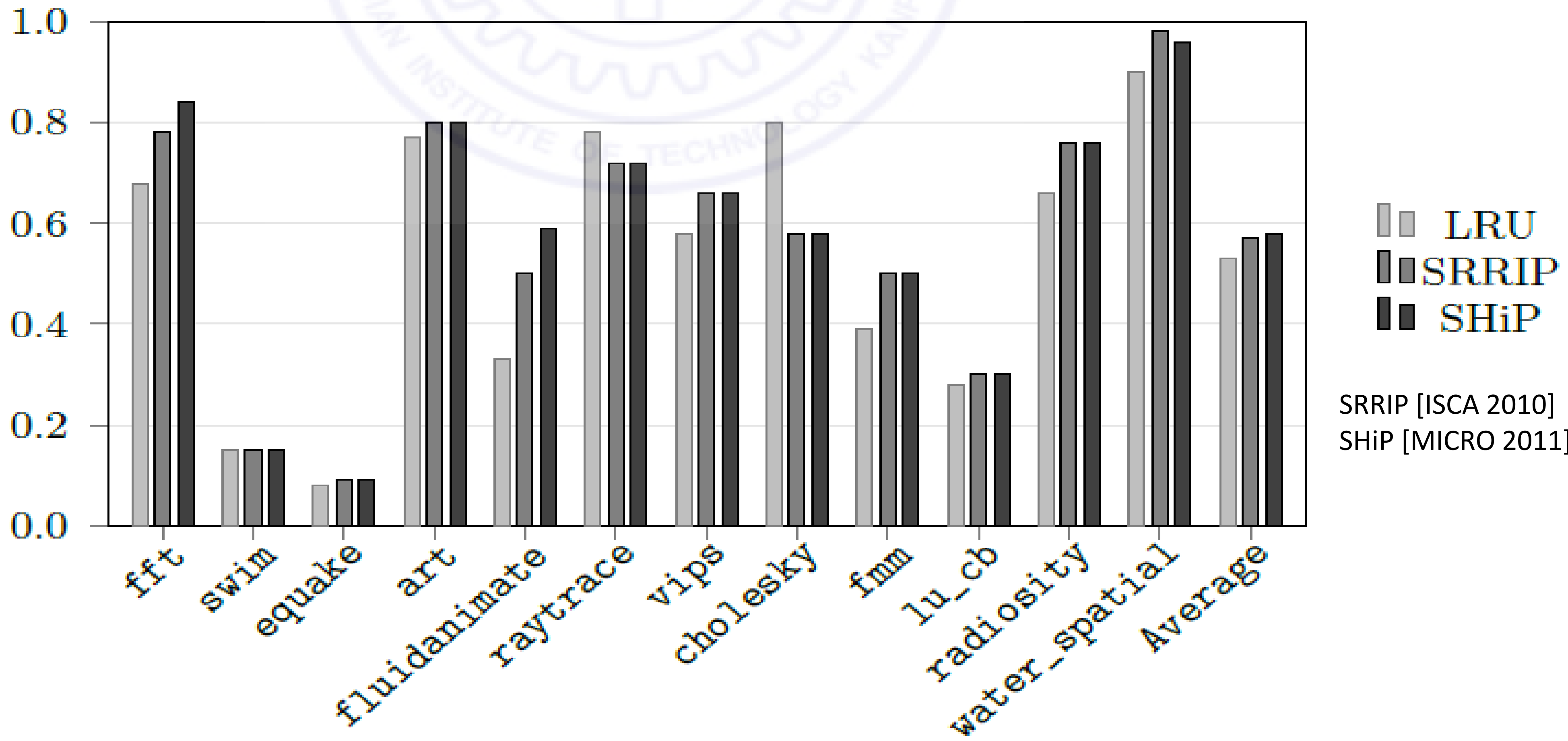


3.8 times more hits

Figure 3.5: Average number of hits enjoyed per block at the LLC with optimal policy

Banerjee, Subarno (supervisor Chaudhuri, Mainak). "Mining Signatures of Inter-core Sharing Behavior in the Last-Level Caches of Multi-core Processors." M.Tech thesis, IIT Kanpur. 2015, June

Problem? - Sharing in the Existing LLC Management Policies



SRRIP [ISCA 2010]
SHiP [MICRO 2011]

SOTA policies:

- **Only 60 %** of cache fills are shared
- They **do not** approximate the optimal sharing behavior well enough

Figure 3.8: Data sharing in baseline policies normalized to optimal sharing

Banerjee, Subarno (supervisor Chaudhuri, Mainak). "Mining Signatures of Inter-core Sharing Behavior in the Last-Level Caches of Multi-core Processors." M.Tech thesis, IIT Kanpur. 2015, June

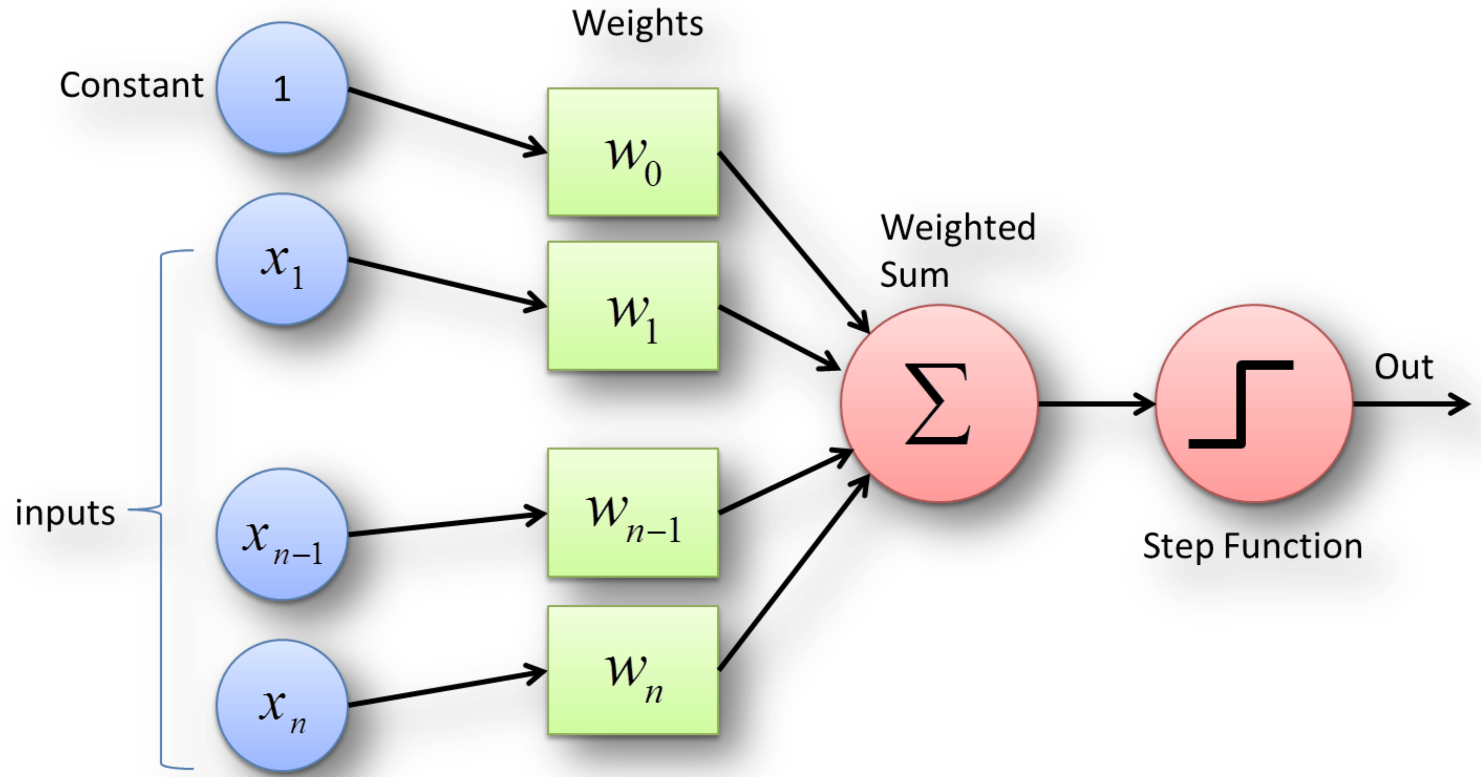
Can we do better ?

How do we predict reused blocks ?

Perceptron Learning Driven Coherence Aware Reuse Prediction for Last-level Caches

Perceptron Learning

- **Prediction** of true or false
- Correctness? → **Update weights**

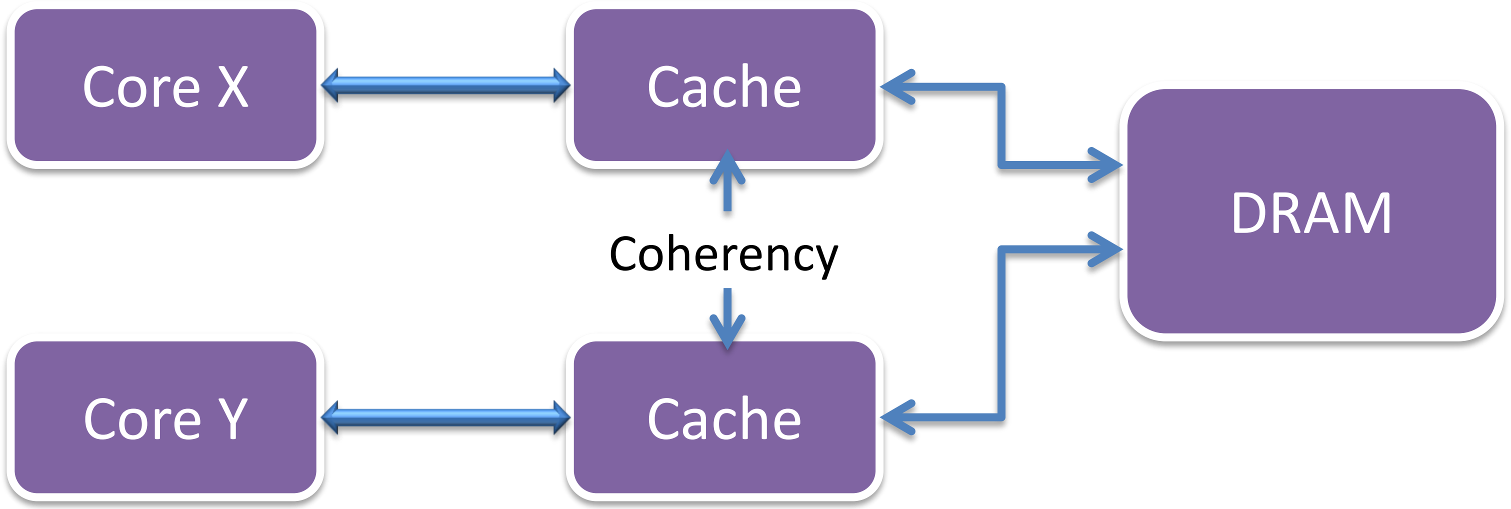


Perceptron Learning Driven **Coherence Aware** Reuse Prediction for Last-level Caches

With parallelism comes some problems !

Cache Coherence

- **Uniformity** of shared resource data present in multiple local caches
- **Coherency Features**: Sharers ID, Number of sharers, etc.



Main idea

Perceptron based reuse prediction to learn the correlation among the coherence features and reuse to guide the LLC replacement policy

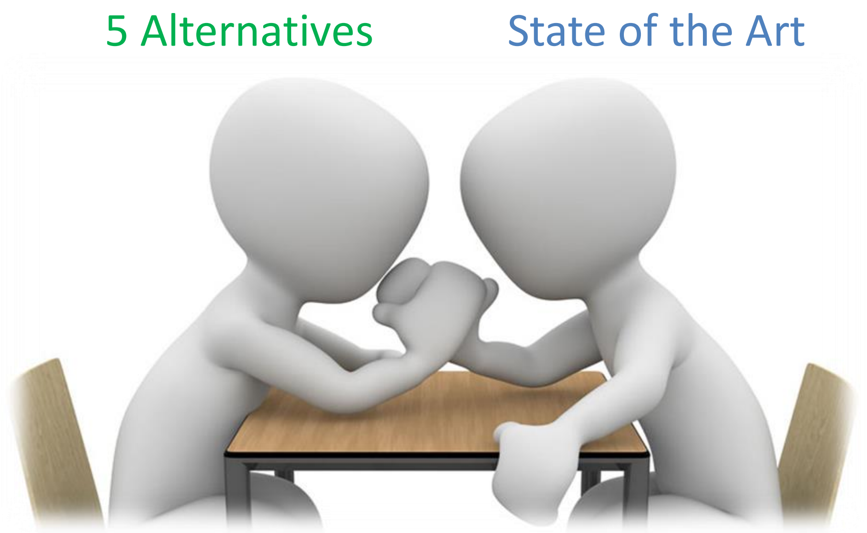
Five alternatives

- **Bias**: The number of sharers of the requested line is used as bias
- **NumSharers** and **NumSharersHash**: Query the number of sharers
- **OneHot** and **OneHotHash**: One hot encoding of the sharers of a cache block.

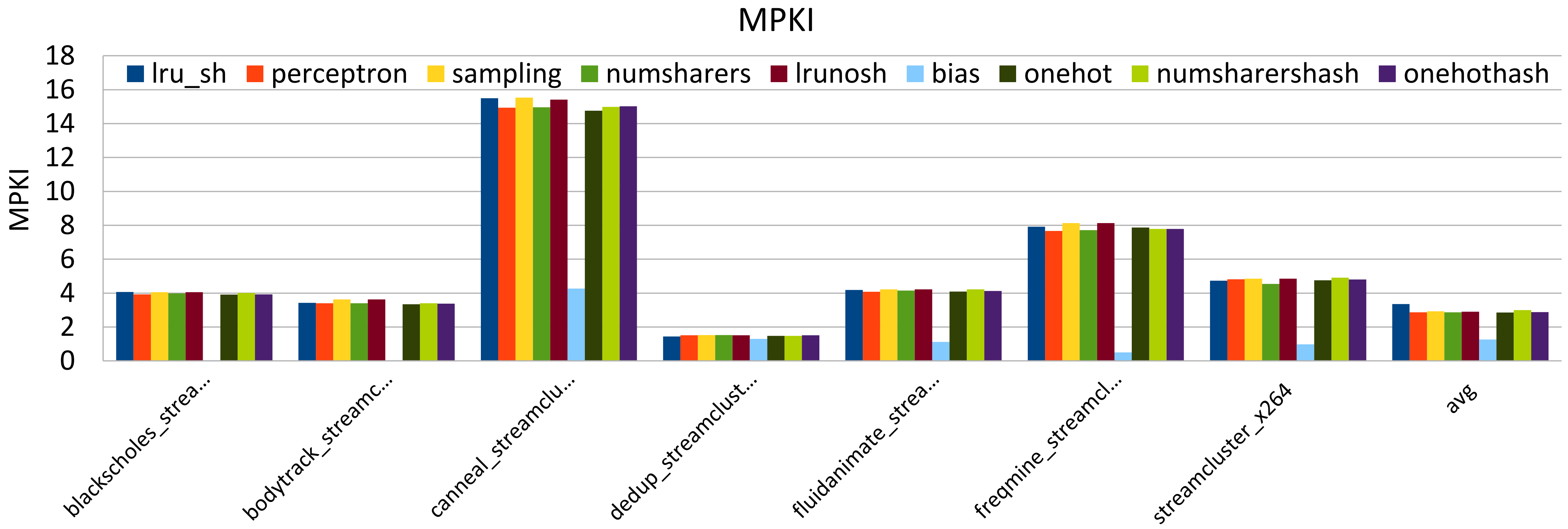
Workloads and Replacement Policies

- PARSEC benchmark suite → 8 multi-threaded applications and kernels.
- Combination of two with 4 threads each, using large input data set
- Performance comparison with:
  - ➢ SDBP
  - ➢ Perceptron Reuse Prediction
  - ➢ LRU (Sharers Aware)
  - ➢ LRU (no sharers aware)

PARSEC [PACT 2008]
SDBP [MICRO 2010]
Perceptron [MICRO 2016]

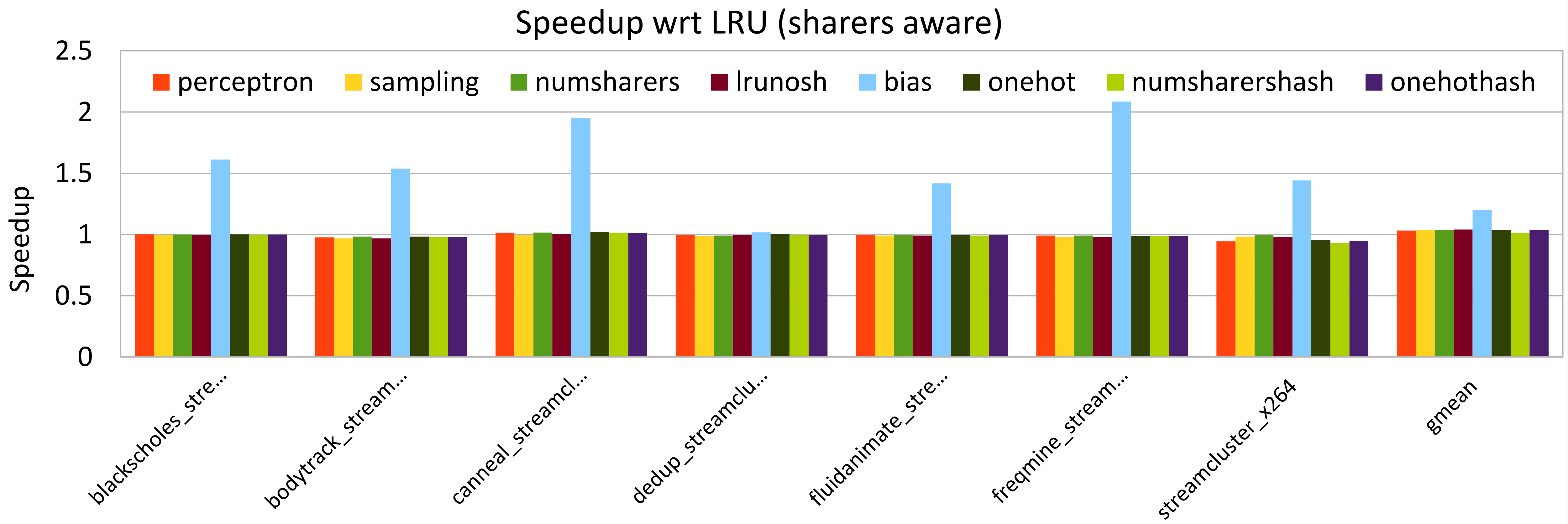


Results (4MB LLC) (MPKI)



- *streamcluster* ↔ other benchmarks, except *dedup,* bias → MPKI dropped to at least 25% wrt LRU
- On average, bias → MPKI dropped to 40%, w.r.t LRU
- Other variations of perceptron → lower MPKI w.r.t LRU.

Results (4MB) (Speedup)



- *streamcluster* ↔ other benchmarks, except *dedup,* bias → speedup of at least 40% wrt LRU.
- Bias → geometric mean speedup of 20% over LRU.
- Other variations of perceptron → marginal improvement over LRU.

Conclusion

- We derive five perceptron alternatives and for 4MB & 8MB LLC they all show improvement, on average.
- Specially, perceptron implemented with **bias** outperforms every other replacement policy and shows a major improvement.

Acknowledgement

- This work is the continuation of that initiated at **Texas A&M University**.
- It was performed along with **Jiayi Huang** and **Pritam Majumder**, supervised by **Dr. EJ Kim**.

Thank You!