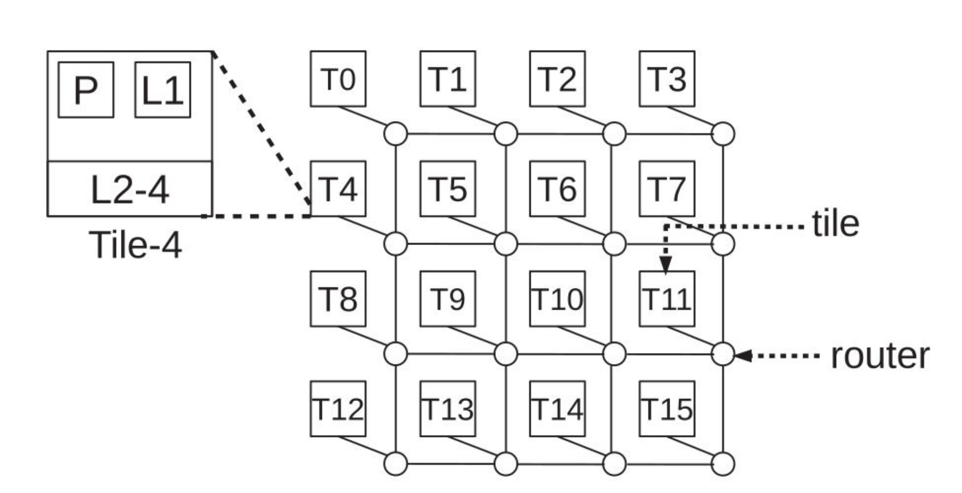
CS531: Memory Systems and Architecture

Course Instructor:

Dr. Shirshendu Das
Assistant Professor,
Department of CSE,
IIT Ropar.

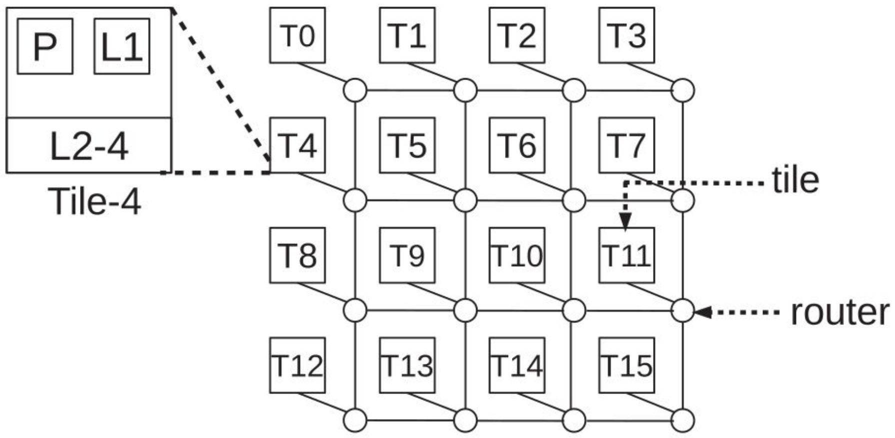
shirshendu@iitrpr.ac.in
http://cse.iitrpr.ac.in/shirshendu/shirshendu.html



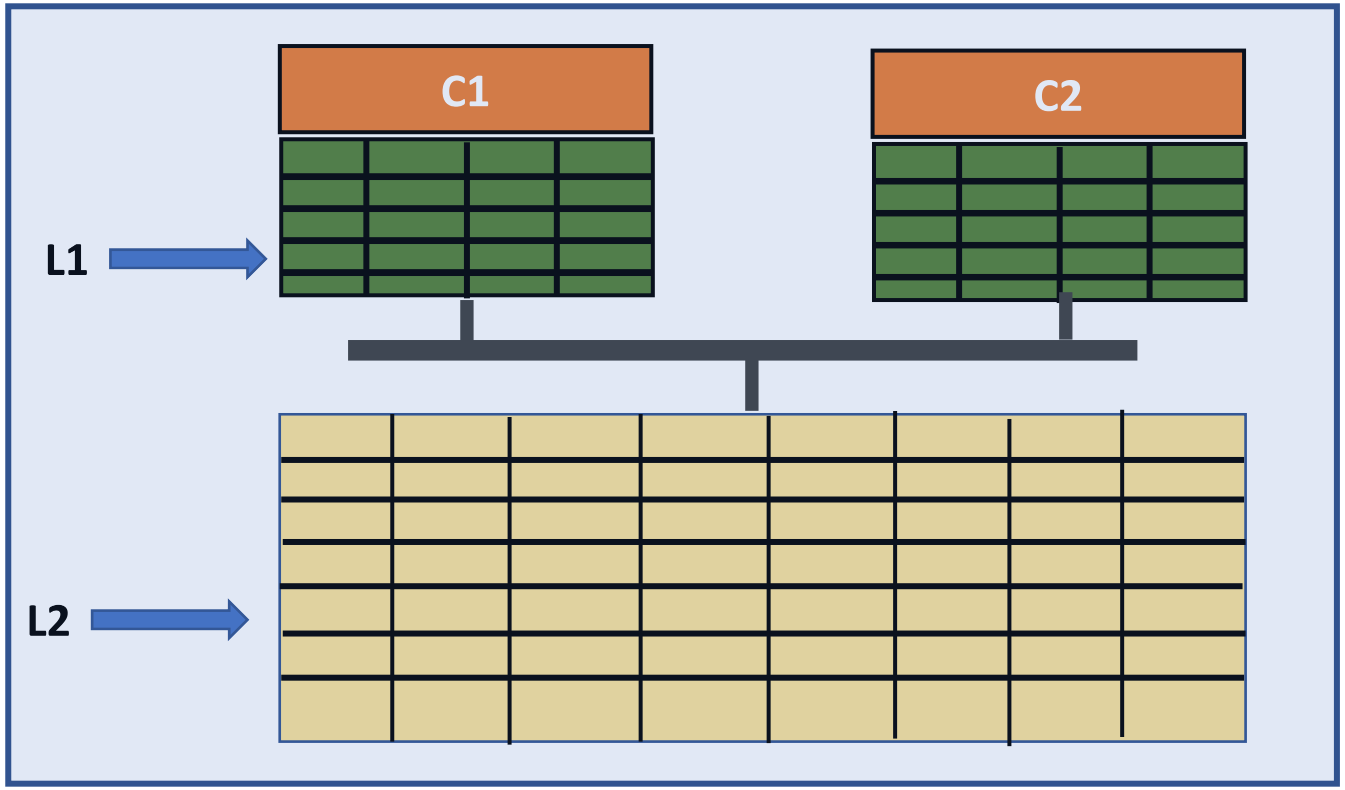
TCMP

Topic: Advancement in Replacement Policy – Part 1

Introduction 1



TCMP



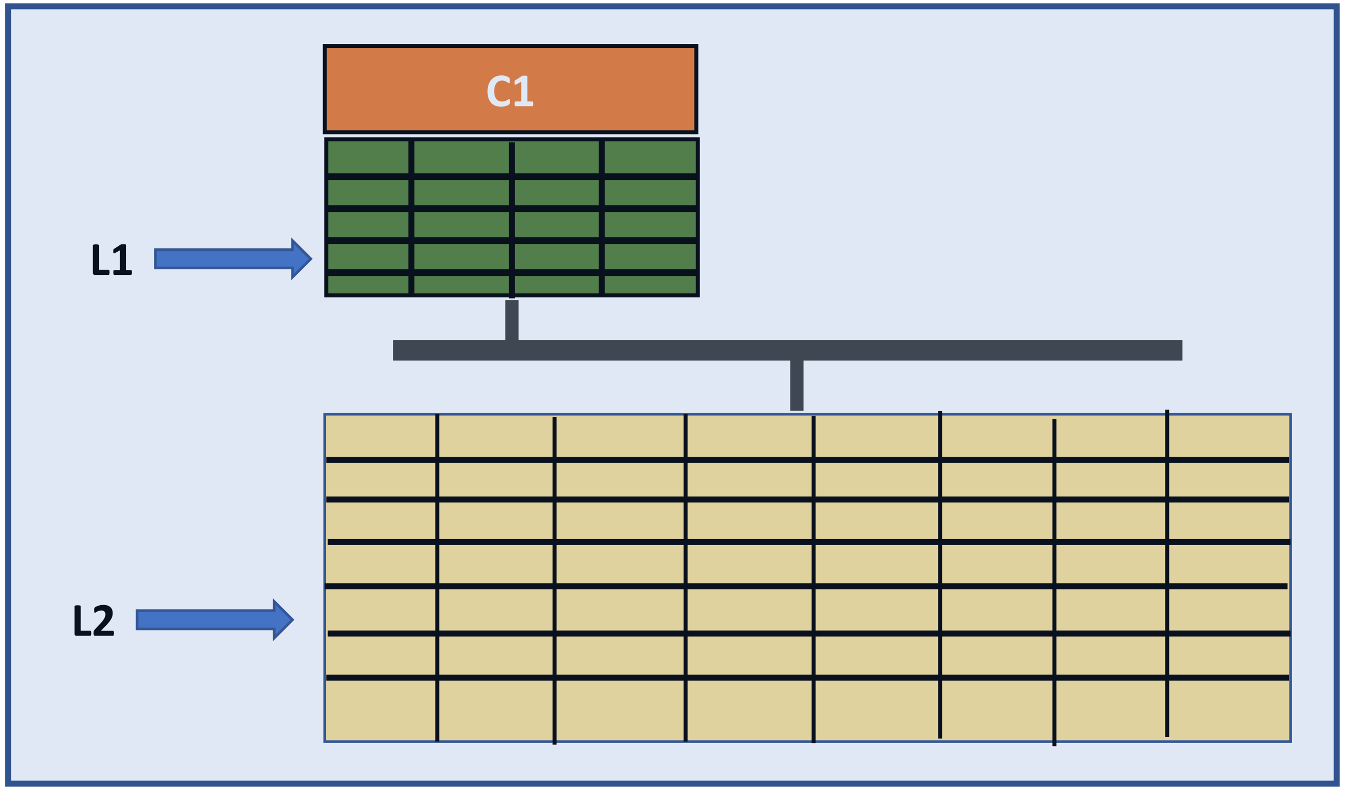
CMP

- **Important parameters:**
  - Performance.
  - Energy consumption
  - Hardware overheads.

Introduction 2

- **Important parameters:**
  - Performance.
  - Energy consumption
  - Hardware overheads.

- **Important parameters:**
  - Scanning/Streaming.
  - Thrashing.

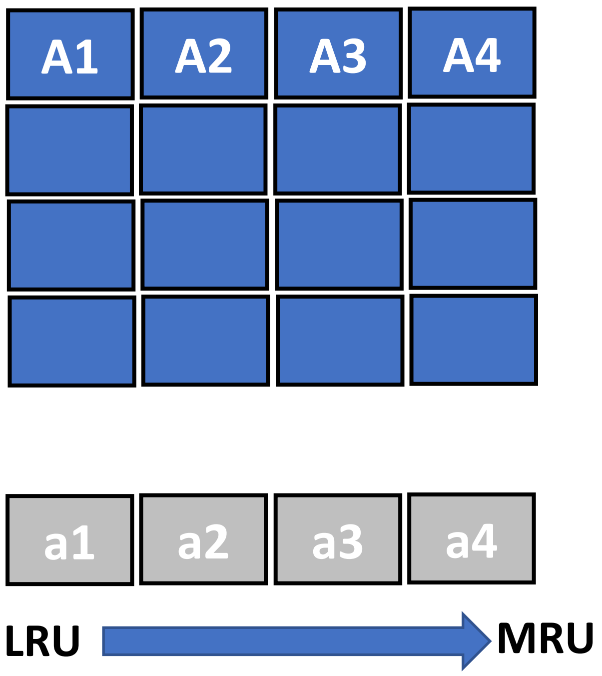


CMP

Trace-1 :	A1	A2	A3	A4	A5	A6	A7	A8	A9	A10	A11	A12	A13	A14	A15	A16	………………………
Trace-2:	A1	A2	A3	A4	A5	A6	A7	A8	A9	A10	A11	A12	A13	A14	A15	A16	

Introduction 3

- **Modules of any Replacement Policy:**
  - Insertion
  - Promotion
  - Eviction



1. *From now onwards we will assume that the hardware unit of replacement is merged with the set and the replacement bits required are stored along with the cache blocks.*
2. *To explain replacement policy we will use examples mostly for one set.*
3. *Most of the replacement policy that we will discuss are for Last Level Cache (LLC).*

Adaptive Insertion Policies for High Performance Caching

Moinuddin K. Qureshi, Aamer Jaleel, Yale N. Patt, Simon C. Steely, and Joel Emer.

In Proceedings of the 34th annual international symposium on Computer architecture (ISCA '07), 2007

Short name: DIP

I have used some figures, tables and texts from the paper in this presentation to explain you the paper. The use is completely for academic purpose.

DIP: Motivation 1

- The LRU replacement policy and its approximations have remained as the de-facto standard for replacement policy in on-chip caches over the last several decades.
- While the LRU policy has the advantage of good performance for high-locality workloads, it can have a pathological behavior for memory-intensive workloads that have a working set greater than the available cache size.
- There have been numerous proposals to improve the performance of LRU, however, many of these proposals incur a huge storage overhead.
- **A miss in the L2 cache (LLC) stalls the processor for hundreds of cycles, therefore, this work is focused on reducing L2 misses by managing the L2 cache efficiently.**
- The access stream visible to the L2 cache has filtered temporal locality due to the hits in the first-level cache. The loss of temporal locality causes a significant percentage of L2 cache lines to remain unused.
- We refer to cache lines that are not referenced between insertion and eviction as ***dead blocks*** or ***zero reuse lines***.

DIP: Motivation 2

- The LRU replacement policy and its approximations have remained as the de-facto standard for replacement policy in on-chip caches over the last several decades.
- While the LRU policy has the advantage of good performance for high-locality workloads, it can have a pathological behavior for memory-intensive workloads that have a working set greater than the available cache size.
- There have been numerous proposals to improve the performance of LRU, however, many of these proposals incur a huge storage overhead.

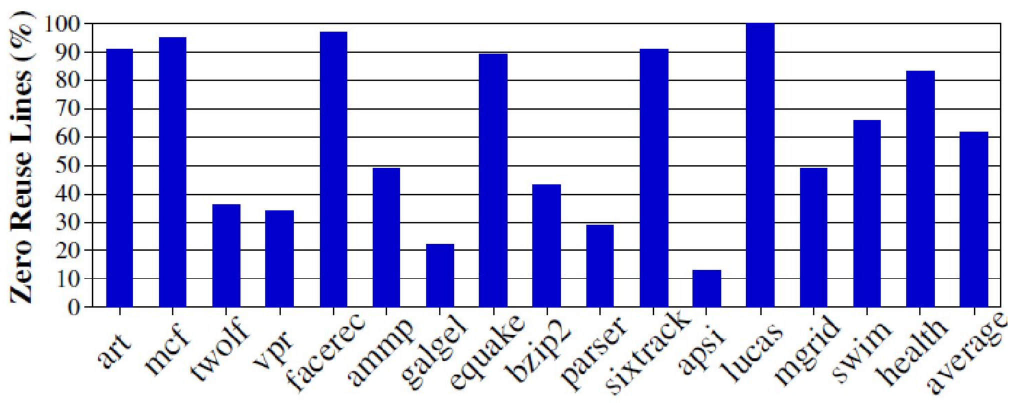
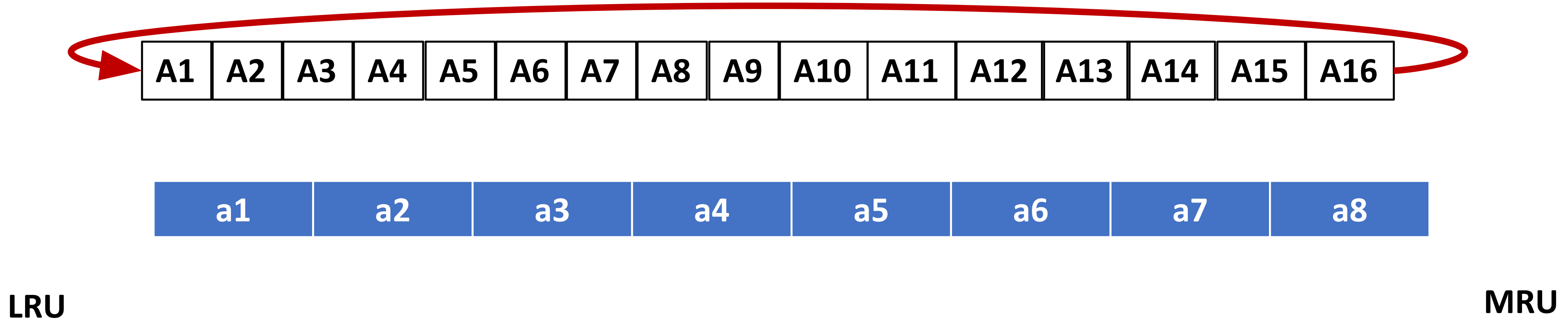


Figure 1: Zero Reuse Lines for 1MB 16-way L2 cache

- Zero reuse lines occur because of two reasons:
  - First, the line has no temporal locality which means that the line is never re-referenced.
  - Second, the line is re-referenced at a distance greater than the cache size, which causes the LRU policy to evict the line before it gets reused.

DIP: Motivation 3

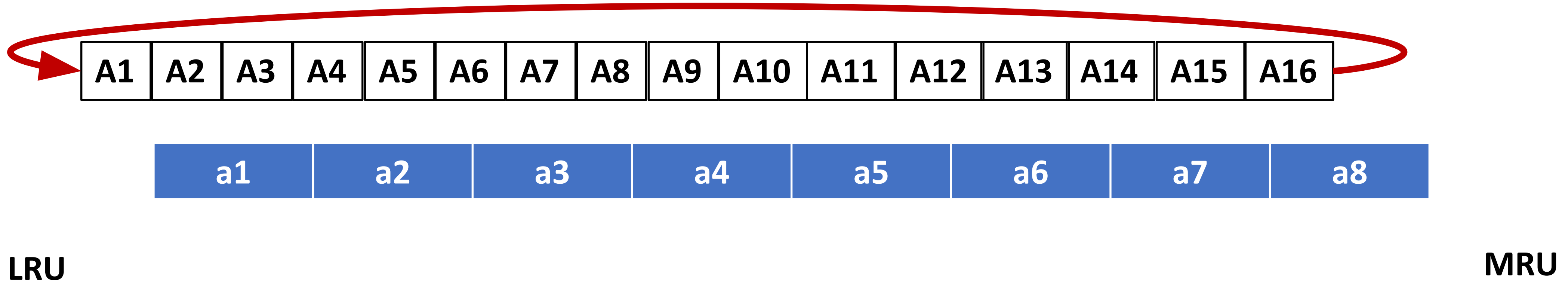


- For workloads with a working set greater than the available cache size, cache performance can be significantly improved if the cache can retain some fraction of the working set.

- To achieve this, DIP separates the replacement policy into two parts: ***victim selection policy*** and ***insertion policy***.
  - The victim selection policy decides which line gets evicted for storing an incoming line.
  - The insertion policy decides where in the replacement list the incoming line is placed.

DIP: Static Insertion Policies

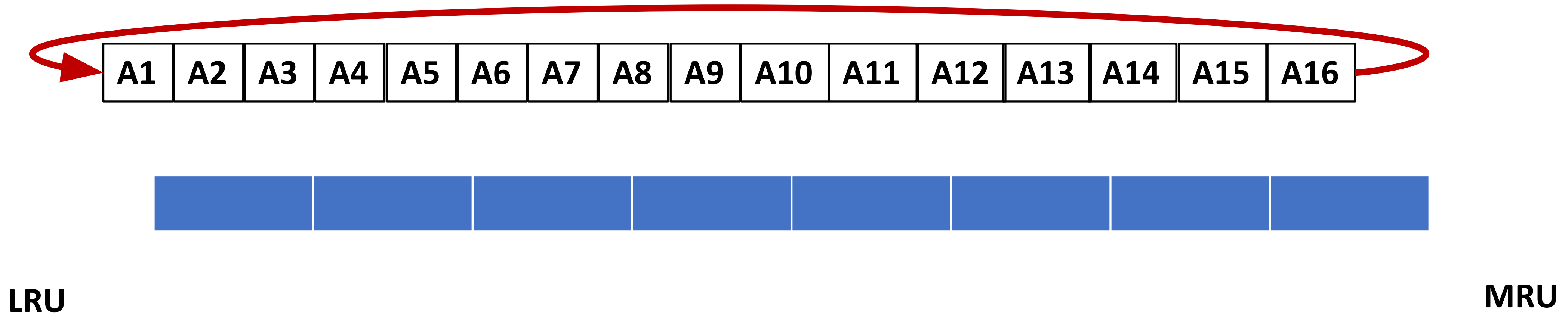
- The traditional LRU replacement policy inserts all incoming lines in the MRU position.
- Inserting the line in the MRU position gives the line a chance to obtain a hit while it traverses all the way from the MRU position to the LRU position.
- When the working set is greater than the available cache size, cache performance can be improved by retaining some fraction of the working set long enough.
- DIP propose the *LRU Insertion Policy (LIP)*, which places *all* incoming lines in the LRU position.



DIP: Static Insertion Policies

- The traditional LRU replacement policy inserts all incoming lines in the MRU position.
- Inserting the line in the MRU position gives the line a chance to obtain a hit while it traverses all the way from the MRU position to the LRU position.
- When the working set is greater than the available cache size, cache performance can be improved by retaining some fraction of the working set long enough.
- DIP propose the *LRU Insertion Policy (LIP)*, which places *all* incoming lines in the LRU position.
- LIP prevents thrashing for workloads that reuse a working set greater than the available cache size.
- LIP may retain the lines in the non-LRU position of the recency stack even if they cease to be re-referenced.
- Better option is *Bimodal Insertion Policy (BIP)*.

DIP: Bimodal Insertion Policies (BIP)



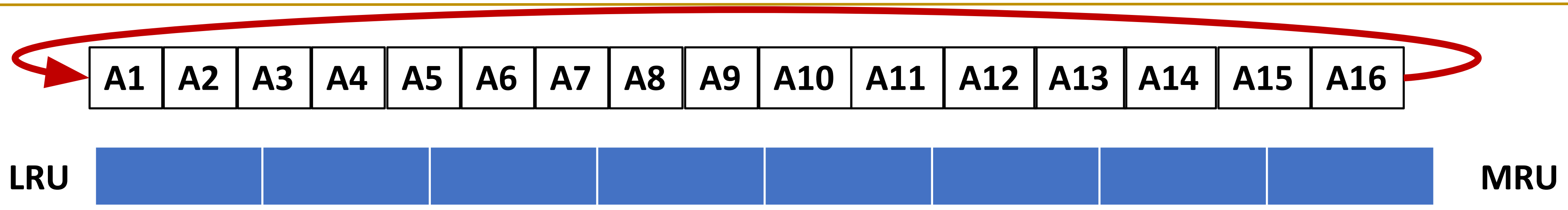
- Insert into LRU position with high probability.
- Insert into MRU position with low probability.

Table 3: Hit Rate for LRU, OPT, LIP, and BIP

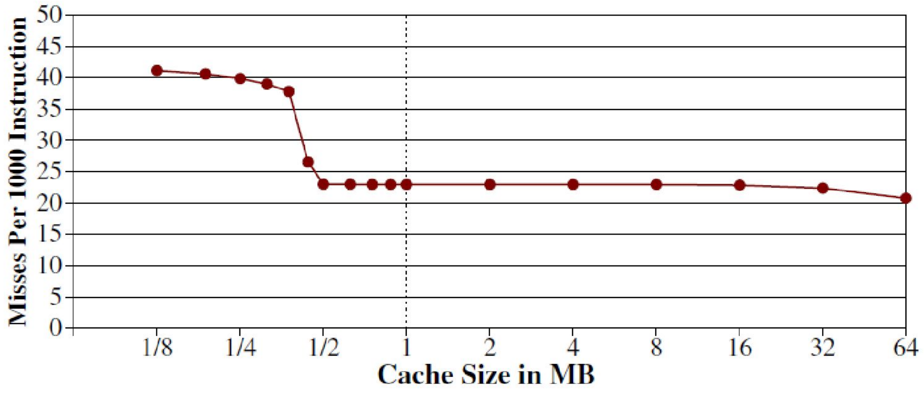
	$(a_1 \cdots a_T)^N$	$(b_1 \cdots b_T)^N$
LRU	0	0
OPT	$(K-1)/T$	$(K-1)/T$
LIP	$(K-1)/T$	0
BIP	$(K-1-\epsilon\cdot[T-K])/T$ $\approx (K-1)/T$	$\approx (K-1-\epsilon\cdot[T-K])/T$ $\approx (K-1)/T$

N: Repetition times.
K: Associativity of a fully associative cache.
T: Stream size.

DIP: LRU-Friendly Workload

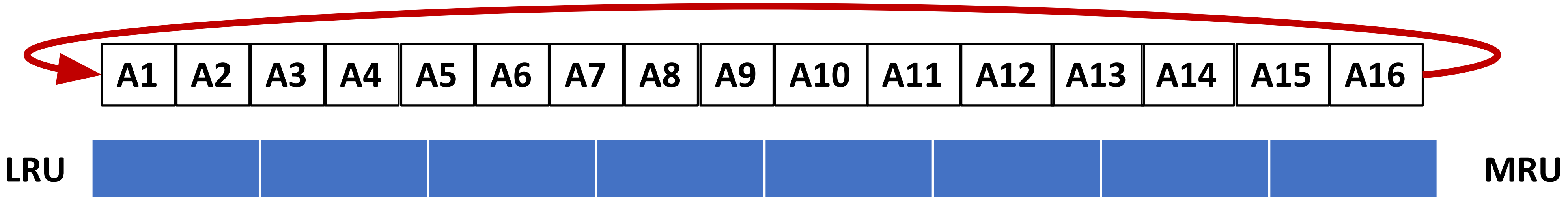


- For workloads that cause thrashing with LRU, both LIP and BIP reduce cache misses significantly.
- However, some workloads inherently favor the traditional policy of inserting the incoming line at the MRU position.



- The MPKI with both LRU and OPT are similar indicating that there is no scope for reducing misses over the LRU policy.
- In fact, changes to the insertion policy can only reduce the hits obtained from the middle of the LRU stack

DIP: Dynamic Insertion Policy (DIP)



- For some applications BIP has fewer misses than LRU and for some LRU has fewer misses than BIP.
- This work, proposes a mechanism that dynamically estimates the number of misses incurred by the two competing insertion policies and selects the policy that incurs the fewest misses.
- The mechanism is called *Dynamic Insertion Policy (DIP)*.

DIP = LRU + BIP

- **Implementation:**
  - DIP Global.
  - Dynamic Set Sampling (DSS).
  - Implementing DIP via Set Dueling

DIP: Dynamic Insertion Policy (DIP)

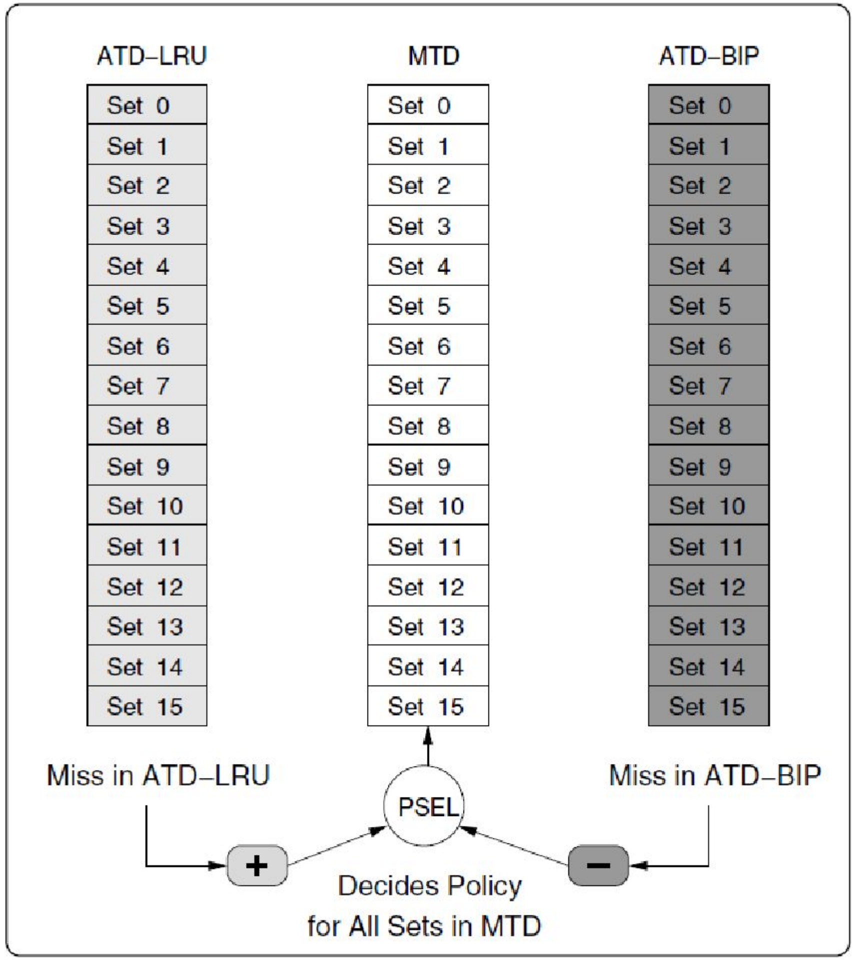


Figure 9: DIP-Global

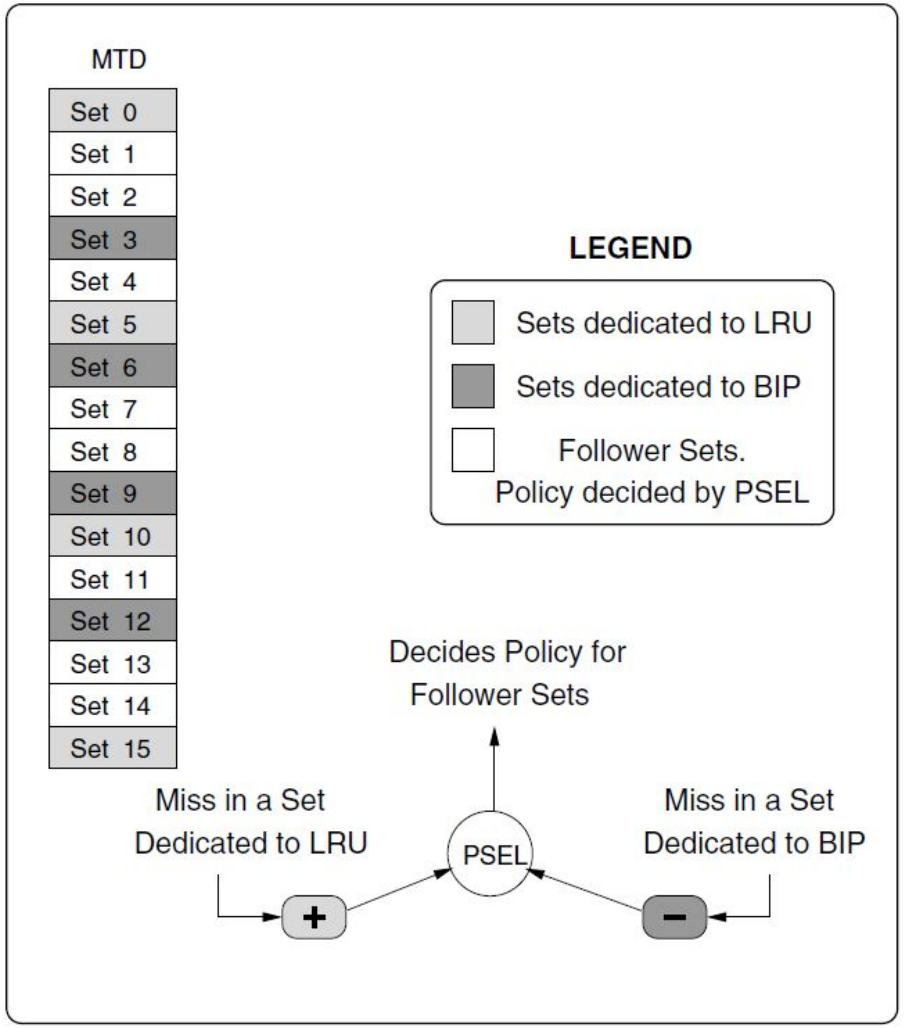


Figure 10: DIP via Set Dueling

Thank You.