



# CS230: Digital Logic Design and Computer Architecture

## Lecture 20: Caches-III

<https://www.cse.iitb.ac.in/~biswa/courses/CS230/autumn23/main.html>

<https://www.cse.iitb.ac.in/~biswa/>

# Cache Performance

How good is the cache for a given application?

Hit rate

Miss rate

Misses per kilo instructions  
(MPKI)

But  
Why?

# Average Access Time

On average, how much time it takes for a LOAD to complete

*Average memory access time (AMAT) =*

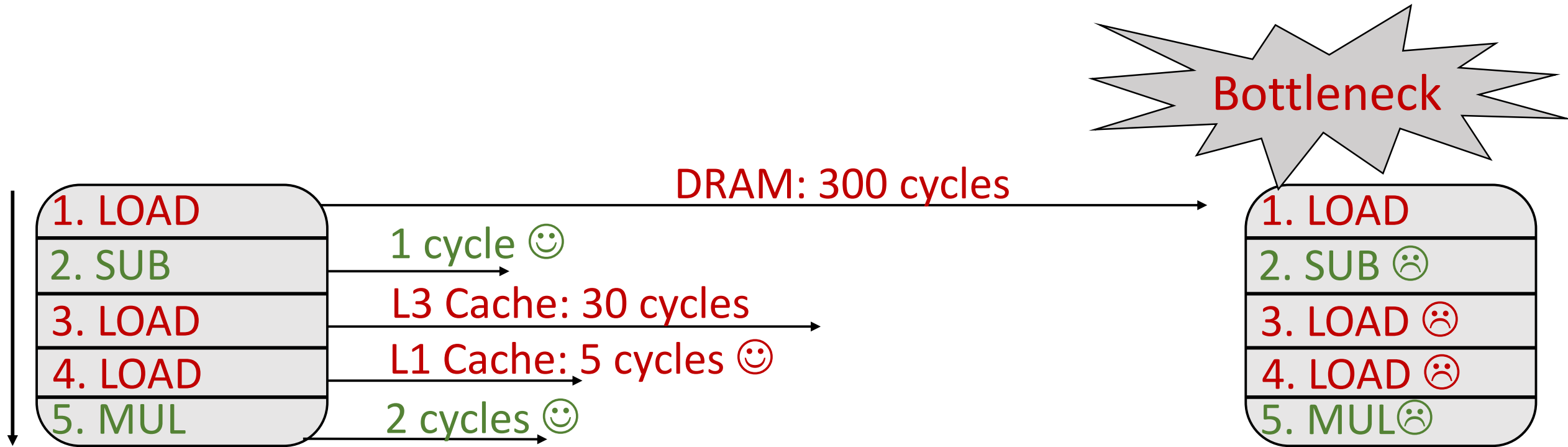
*Hit time + Miss rate x Miss penalty*  
*Hit time-L1 + Miss rate-L1 x Miss penalty-L1*

Miss penalty-L1 = Hit time-L2 + Miss rate-L2 x Miss penalty-L2

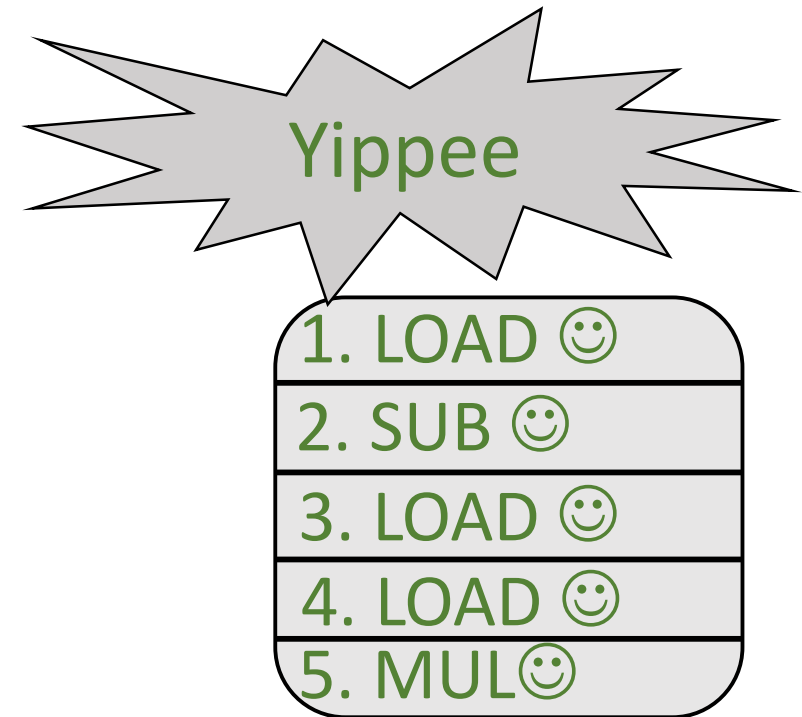
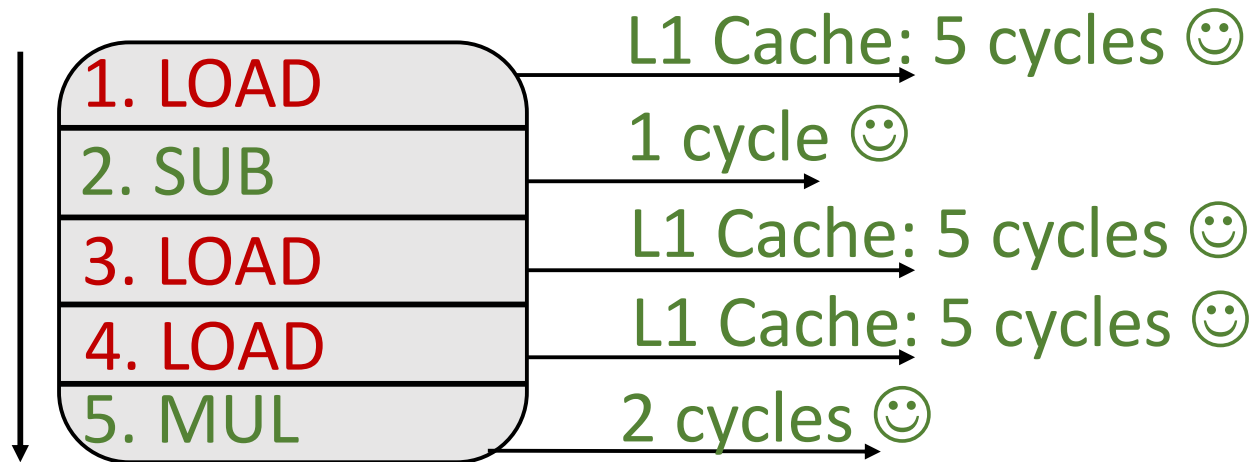
*Hit time: Low, Miss rate: Low , Miss penalty: Low*

*Ideally, miss rate = 0.00% and hit time should be one cycle, so all LOADs will take just one cycle 😊*

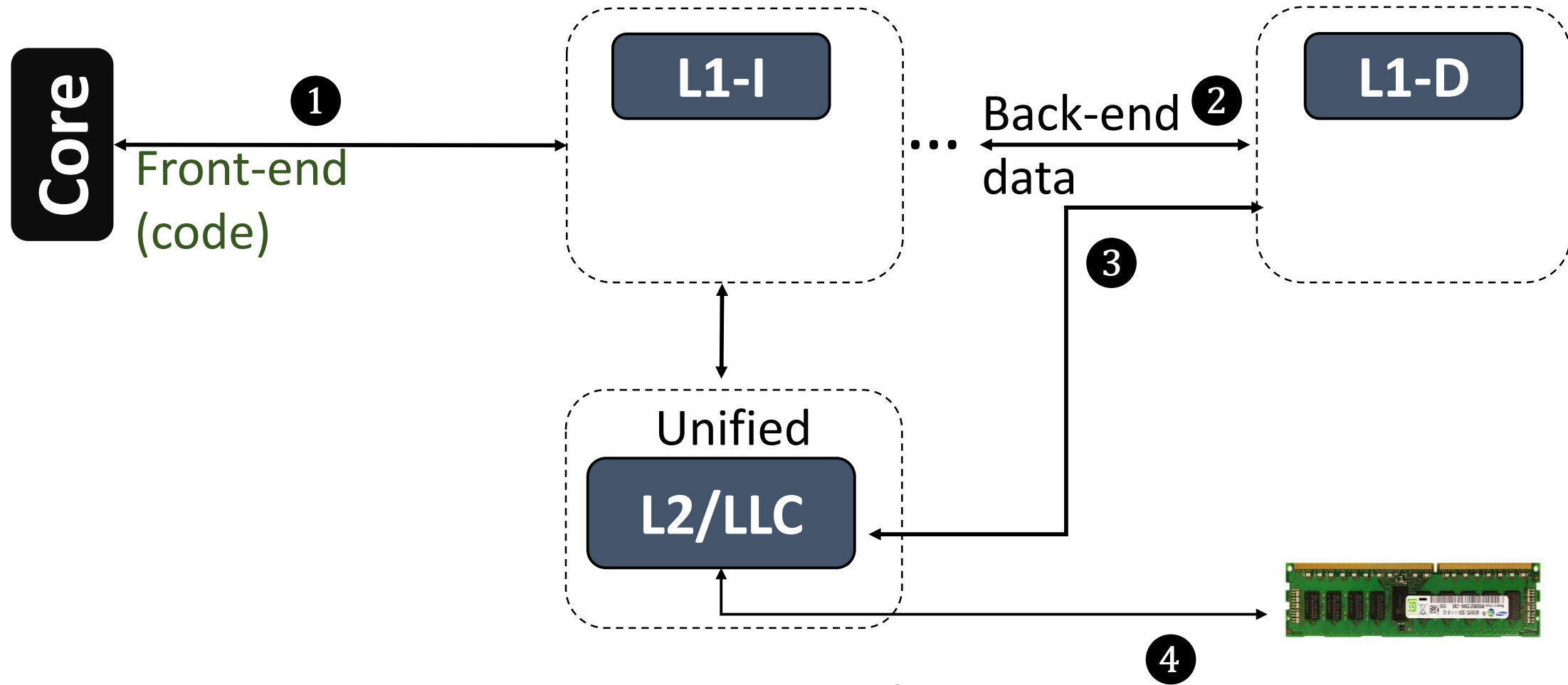
# The Implications of L1-D Hit rate of 100%



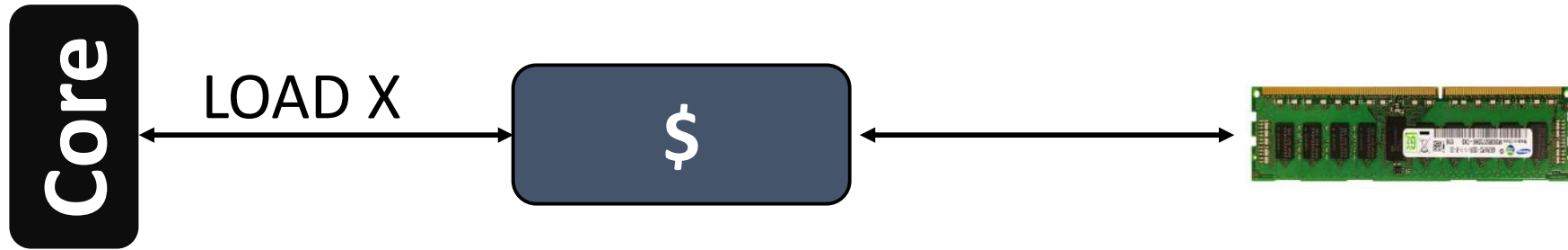
# A Dreamy World



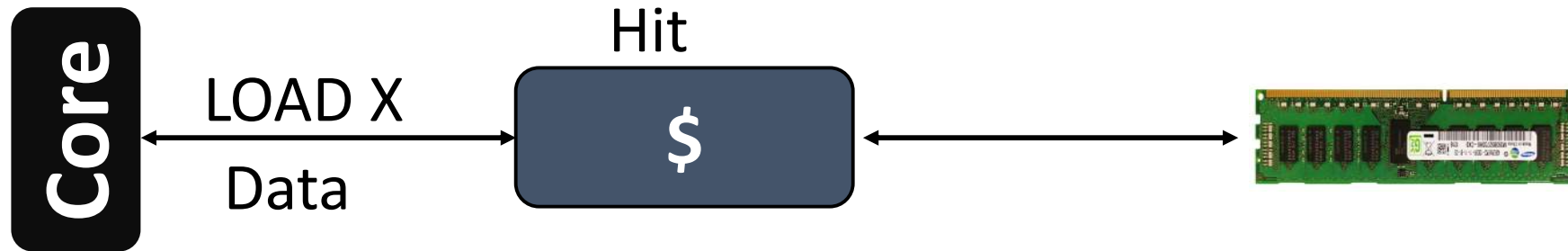
# Core, cache, DRAM interaction



# Core, cache, DRAM interaction



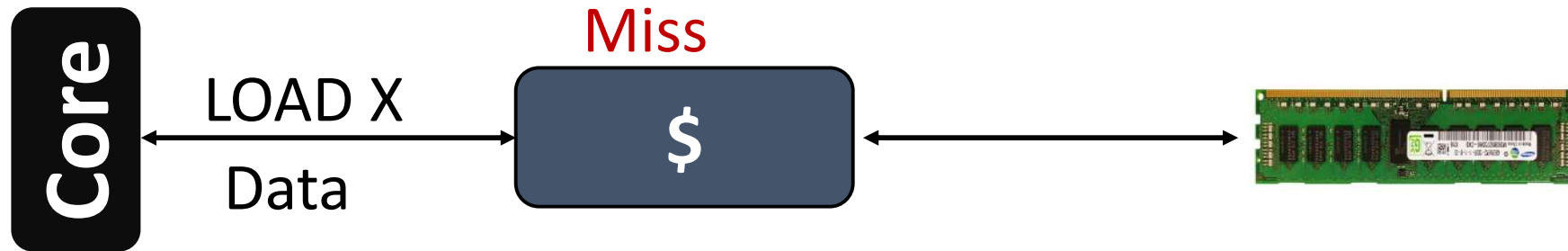
# Core, cache, DRAM interaction



Few  
cycles

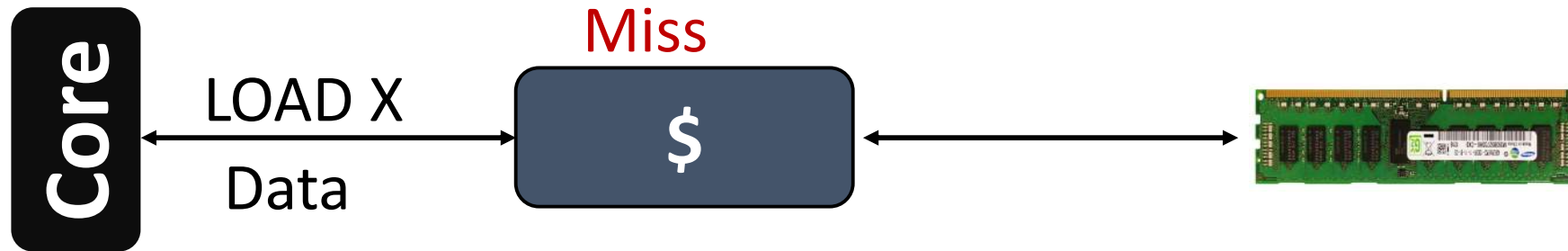


# On a miss: Critical Word first



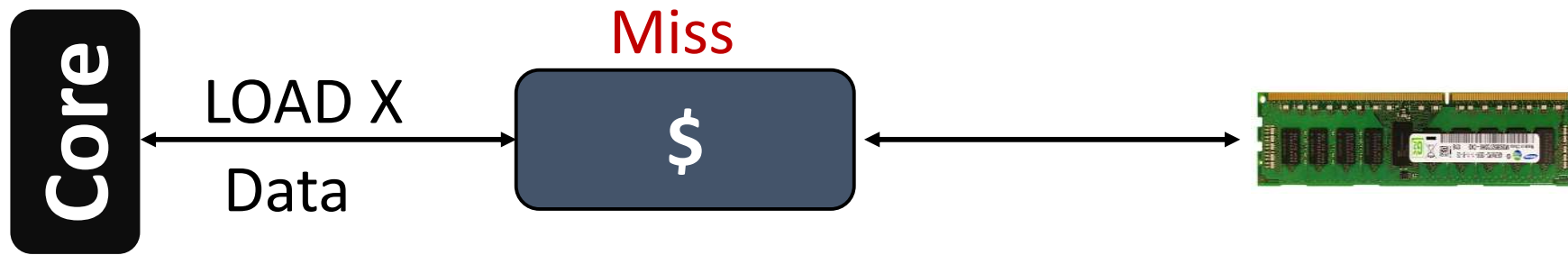
On a miss, respond **with the word/byte requested** to the core so that core can continue while fetching the rest of the block

# On a miss: Early Restart



On a miss, fetch the words/bytes in normal order, **but as soon as the requested word/byte** of the block arrives, send it to the core.

# Core, cache, DRAM interaction



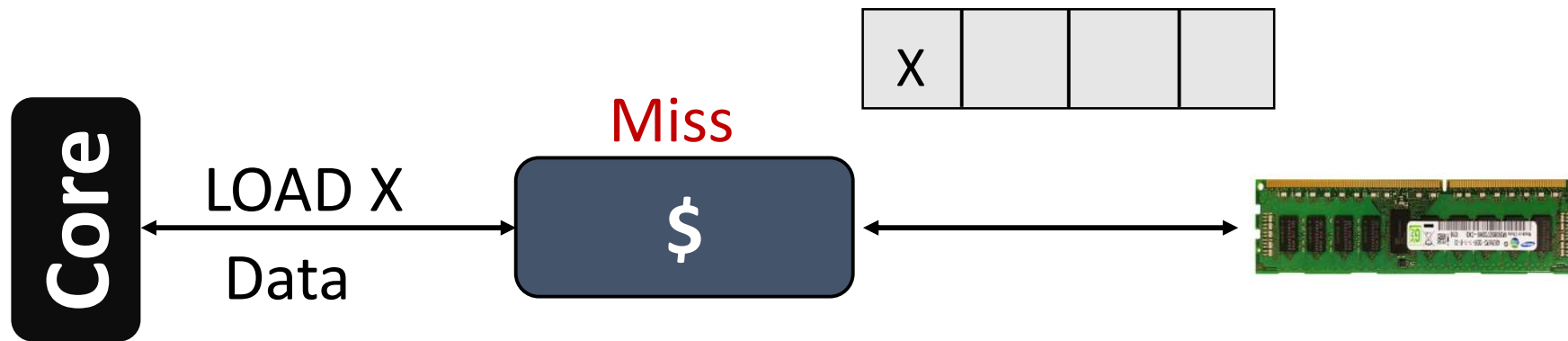
100s of cycles

I am an out-of-order  
core



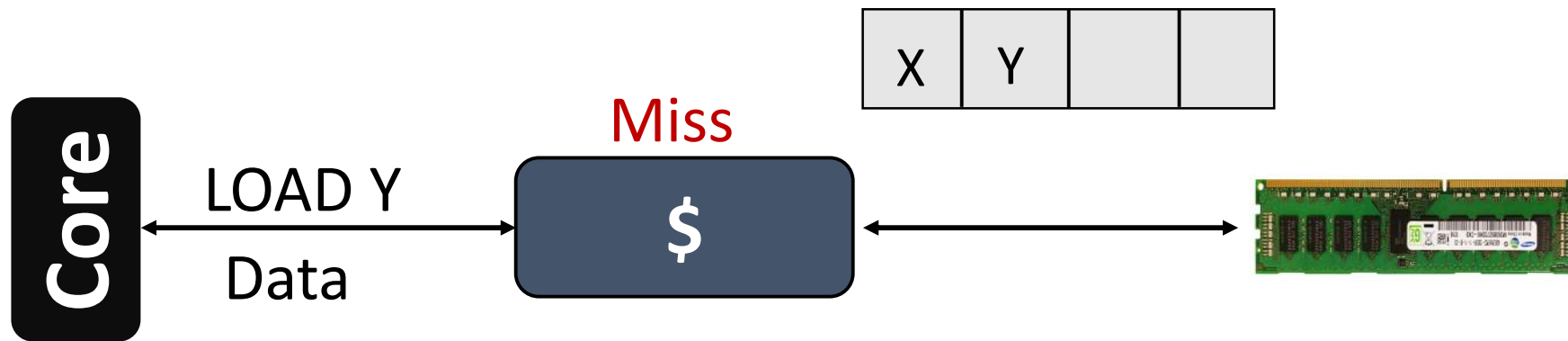
One cache miss and  
can't handle anymore  
misses

# MSHRS (Miss-status holding registers)

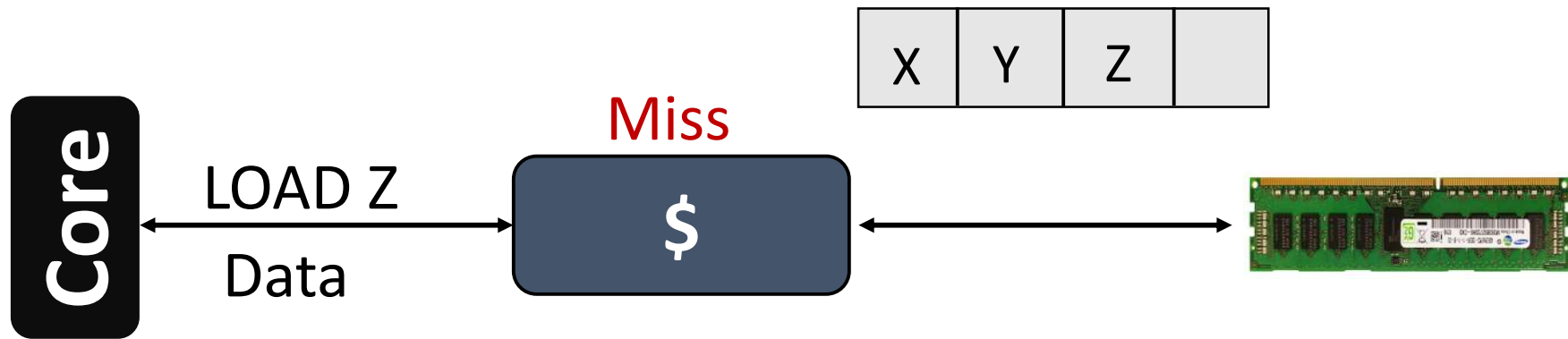


Non-blocking cache

# MSHRS (Miss-status holding registers)

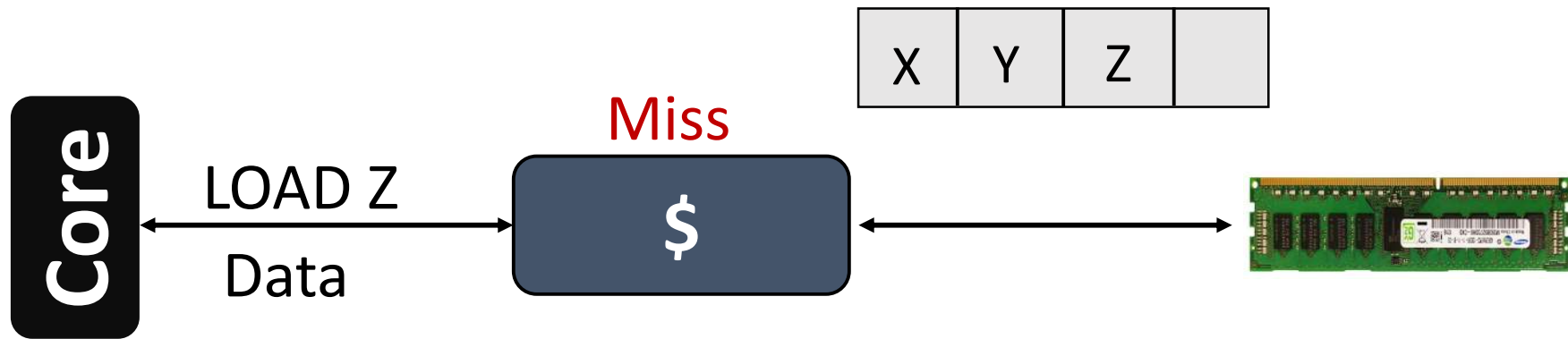


# MSHRS (Miss-status holding registers)



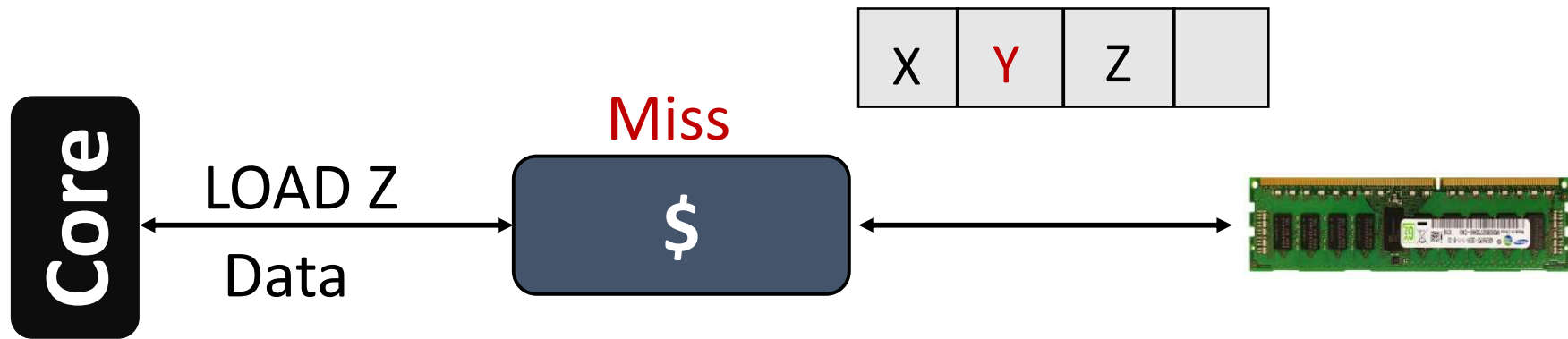
K-entry MSHR allows K outstanding misses: provides memory-level parallelism

# MSHRS (Miss-status holding registers)



DRAM response time is not constant: can take from 60 cycles to 1000s of cycles (on a multi-core system).

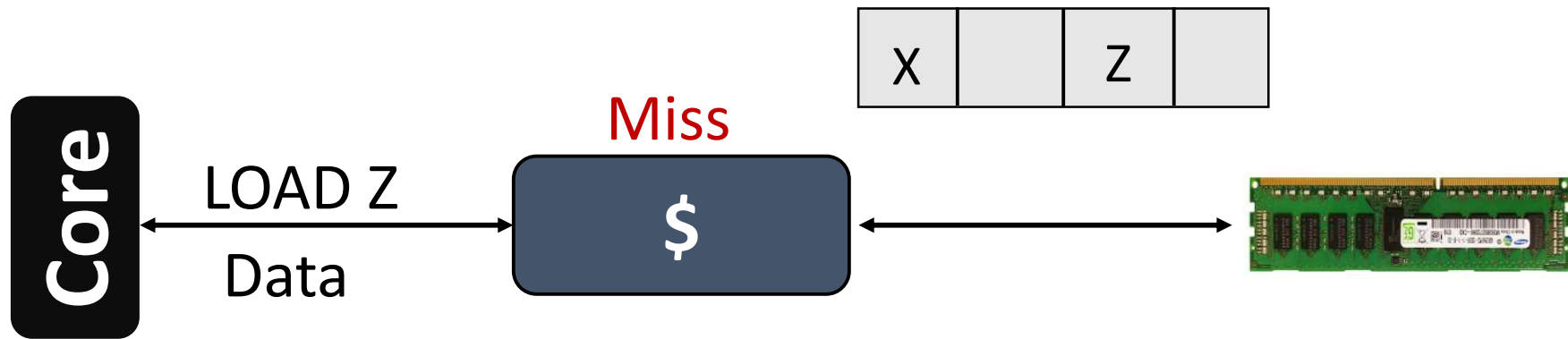
# MSHRS (Miss-status holding registers)



DRAM response time is not constant: can take from 60 cycles to 1000s of cycles (on a multi-core system).



# MSHRS (Miss-status holding registers)



DRAM response time is not constant: can take from 60 cycles to 1000s of cycles (on a multi-core system).

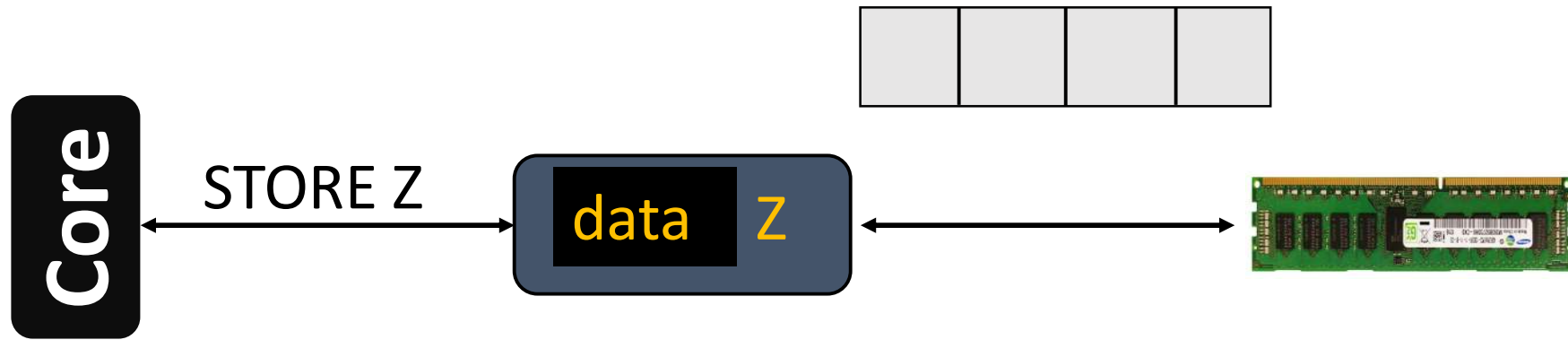
# What about writes (stores)

On a hit: Update the cache block. We need an additional bit in tag-store, named *dirty bit along with the valid bit and replacement priority bits*.

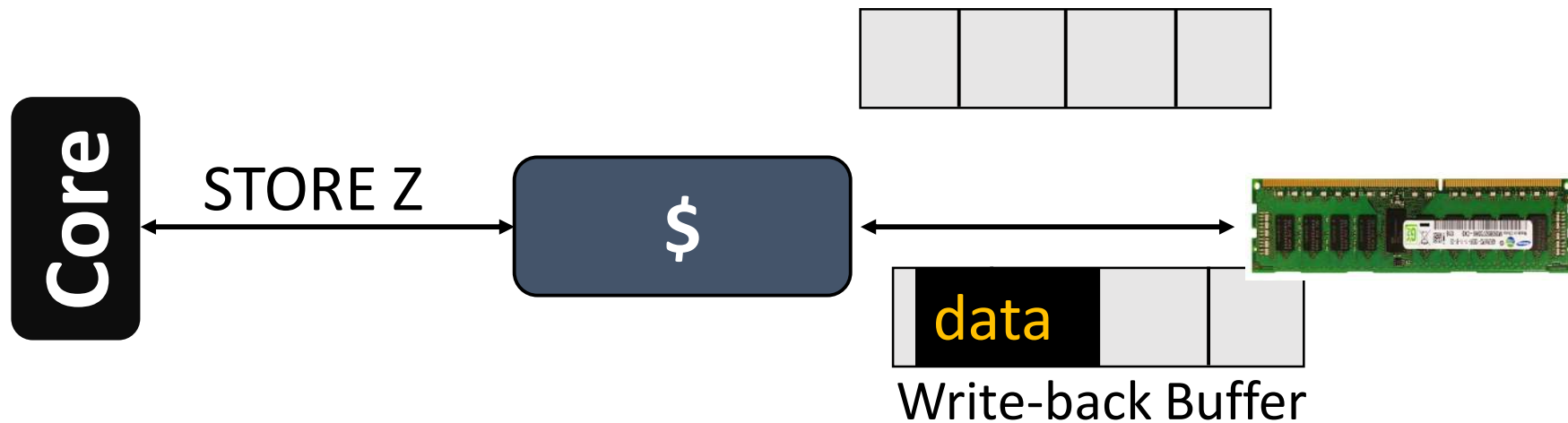
**Write-through** cache: On a hit, write into a cache block and also into the next-level in the memory hierarchy

**Write-back** cache: On a hit, write into a cache block only, and during replacement update the next-level

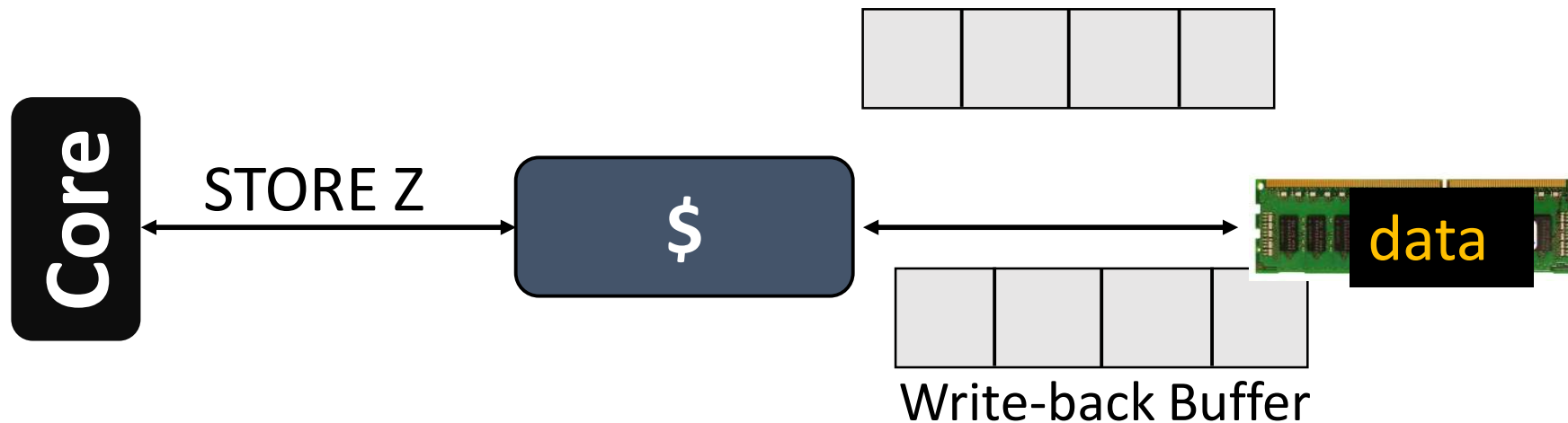
# What about writes: Writeback cache



# What about writes: On replacement



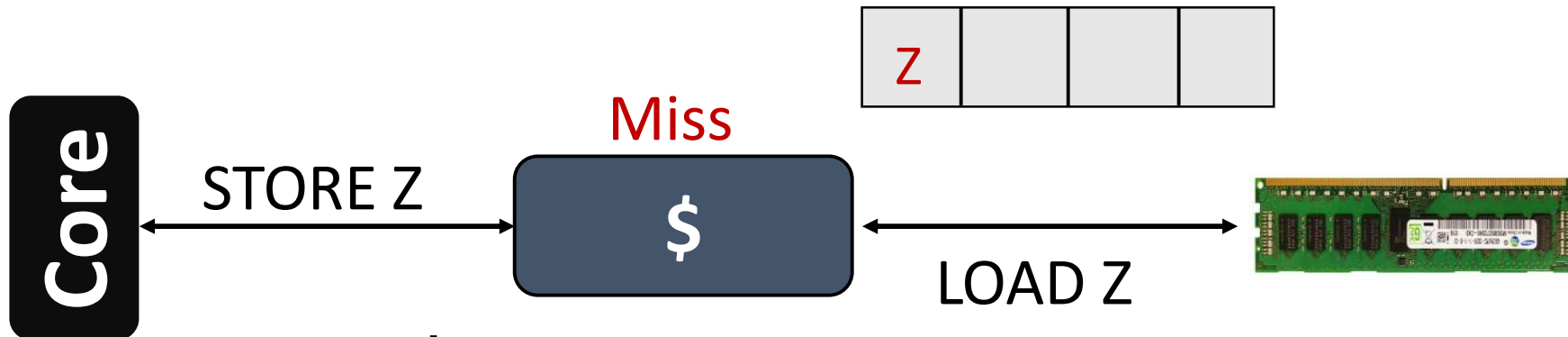
# Write-back cache



Write-back  
cache

In general, STOREs are not critical for performance. But why?

# What about a write miss?



STORE gets converted to a LOAD, and data is allocated into the cache (write-allocate policy).

Usually write-allocate is used with the write-back caches.

# The Bigger Picture

CPU time = CPU execution cycles + Memory-stall cycles

*Clock cycle time may be different*

Memory-stall cycles = Read-stalls + Write-stalls

*Read/Write stalls =*

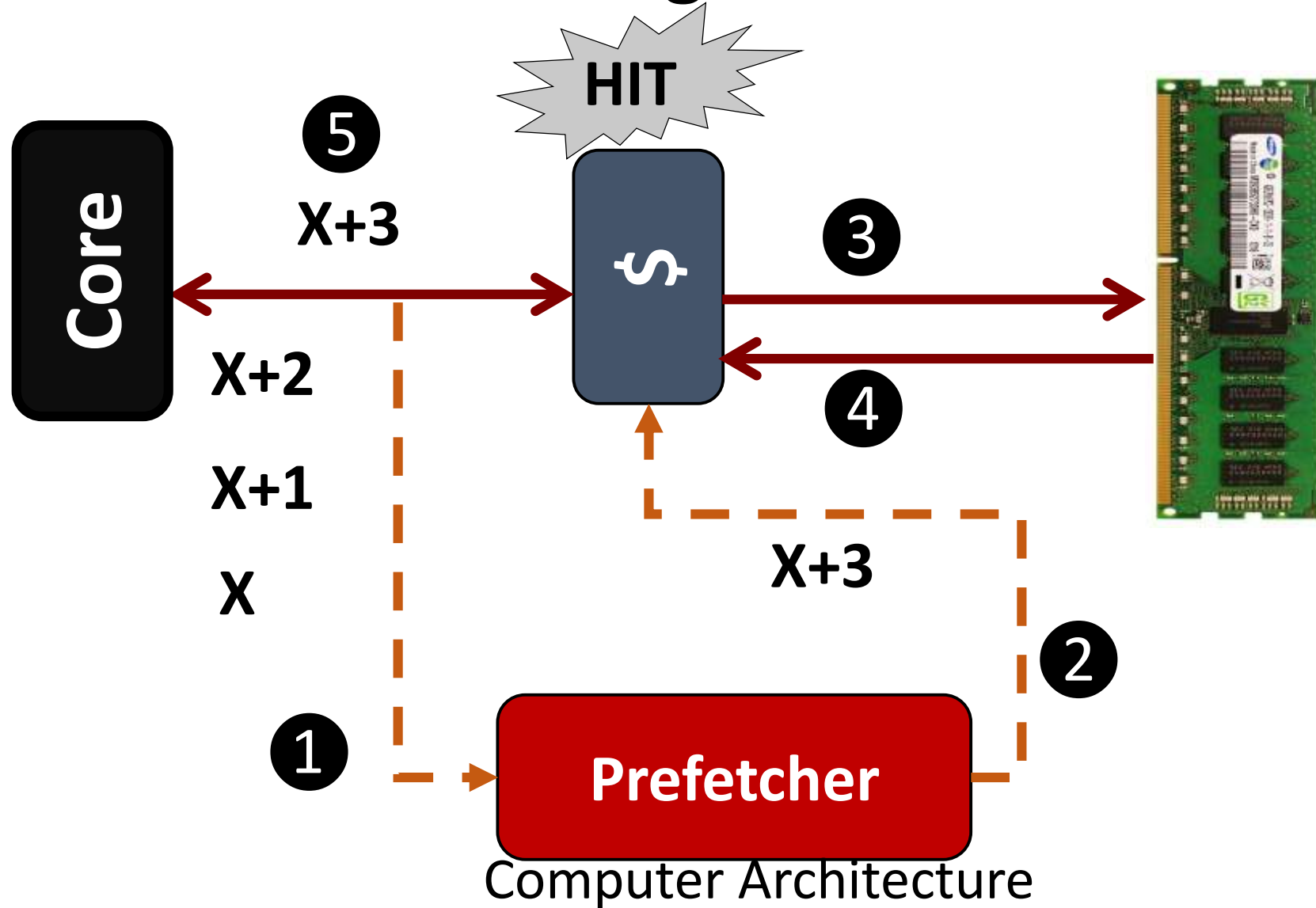
*#Reads/writes X Read/Write miss-rate X Read/write miss-penalty*



# Latency Hiding Technique



# Hardware Prefetching



# 10K Feet View

*What?*

**Latency-hiding technique** - Fetches data before the core demands.

*Why?*

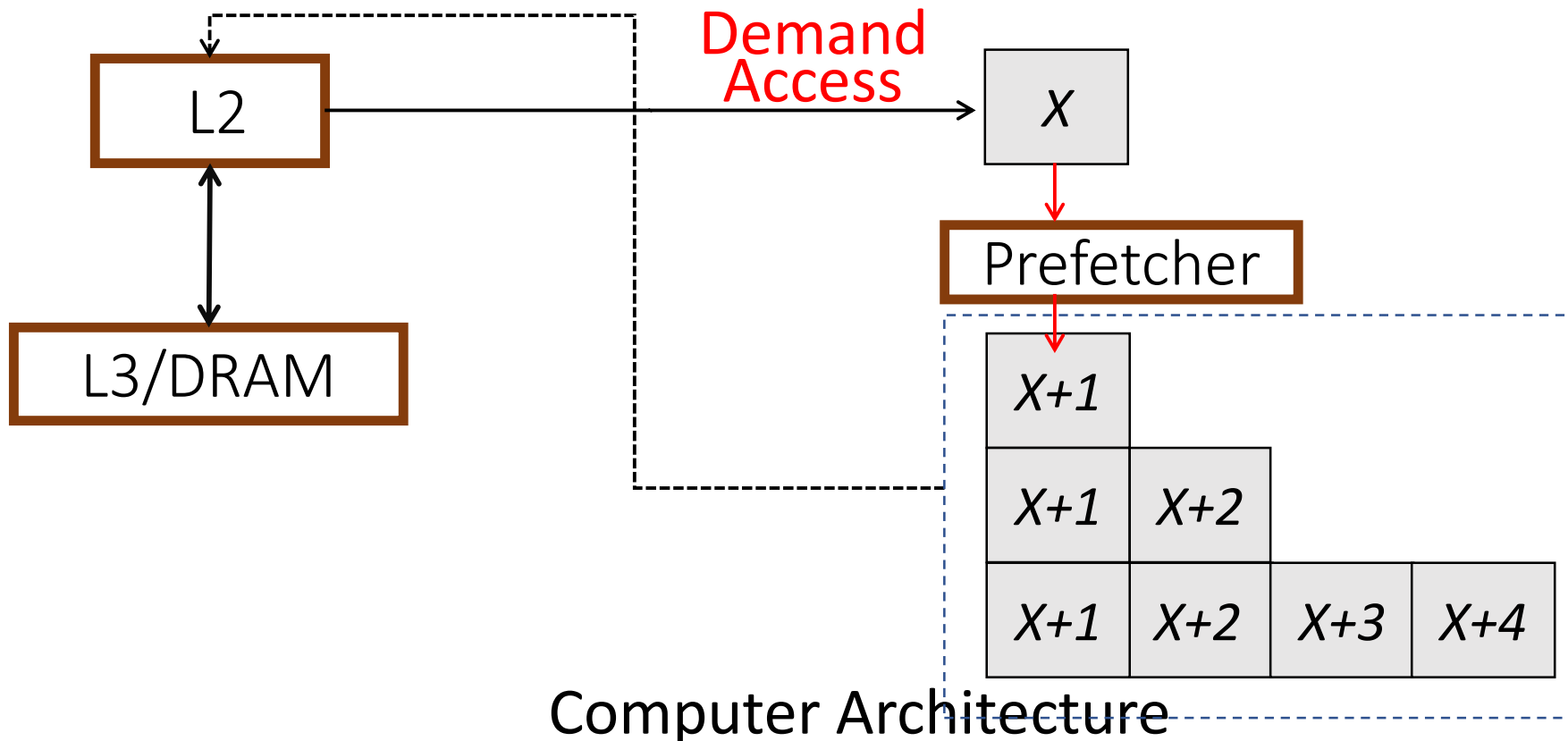
Off-chip DRAM latency has grown up to 400 to 800 cycles.

*How?*

By observing/predicting the demand access (LOAD/STORE) patterns.

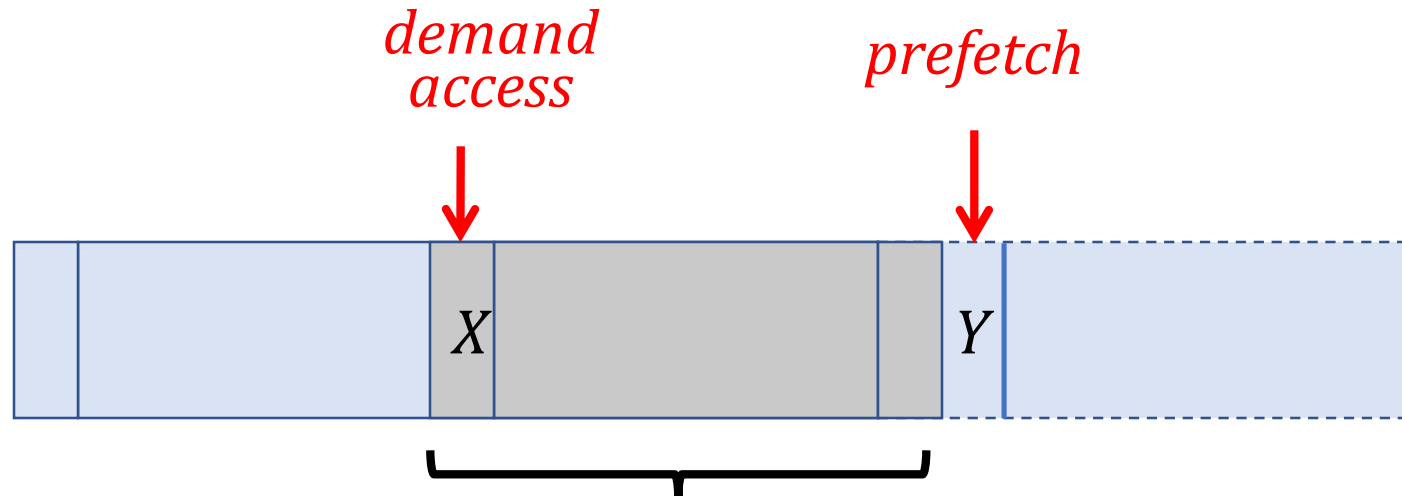
# Prefetch Degree

**Prefetch Degree:** Number of prefetch requests to issue at a given time.



# Prefetch Distance

**Prefetch Distance:** How far ahead of the demand access stream are the prefetch requests issued?



*Prefetch-distance*

$$Y = X + 4$$

$$Y = X + 8$$

$$Y = X + 16$$

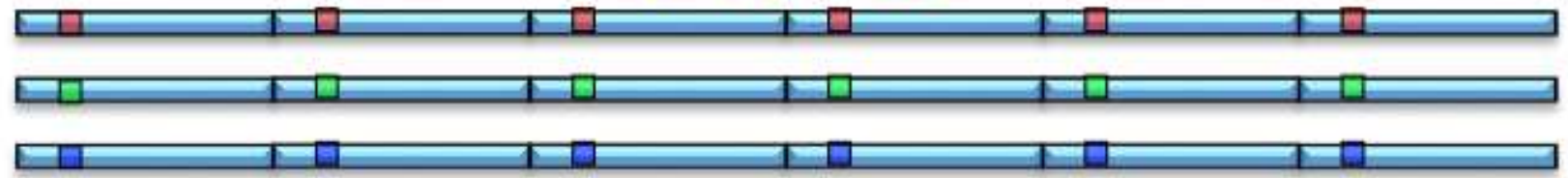
# Next-line prefetcher

**Next Line:** Miss to cache block  $X$ , prefetch  $X+1$ . Degree=1, Distance=1

Works well for L1 Icache and L1 Dcache.

# What About this?

$$Y = A + X?$$

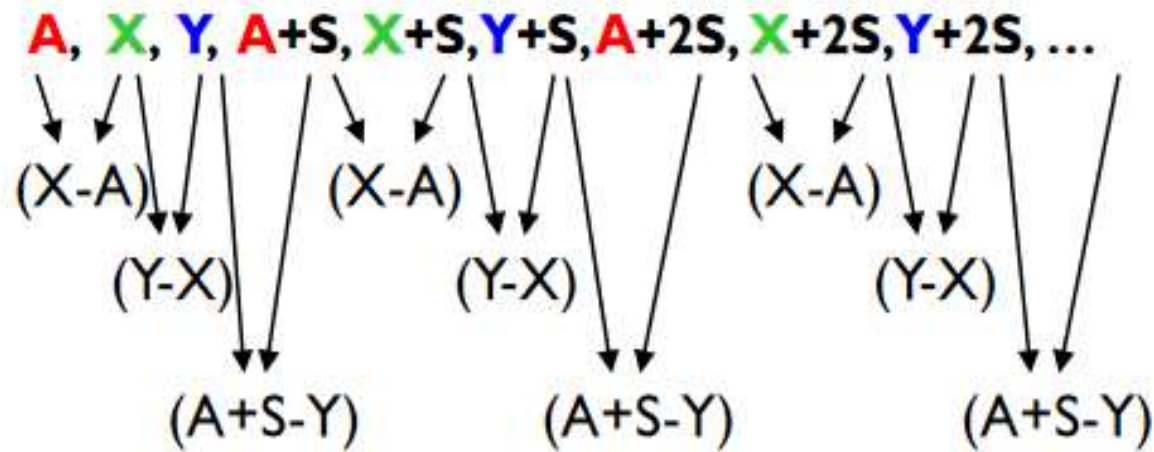


Load R1 = [R2]

Load R3 = [R4]

Add R5, R1, R3


Store [R6] = R5



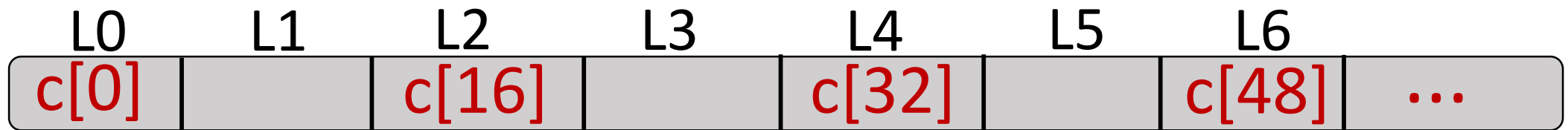
# Example

```
for (i = 0; i < N; i=i+16)
```

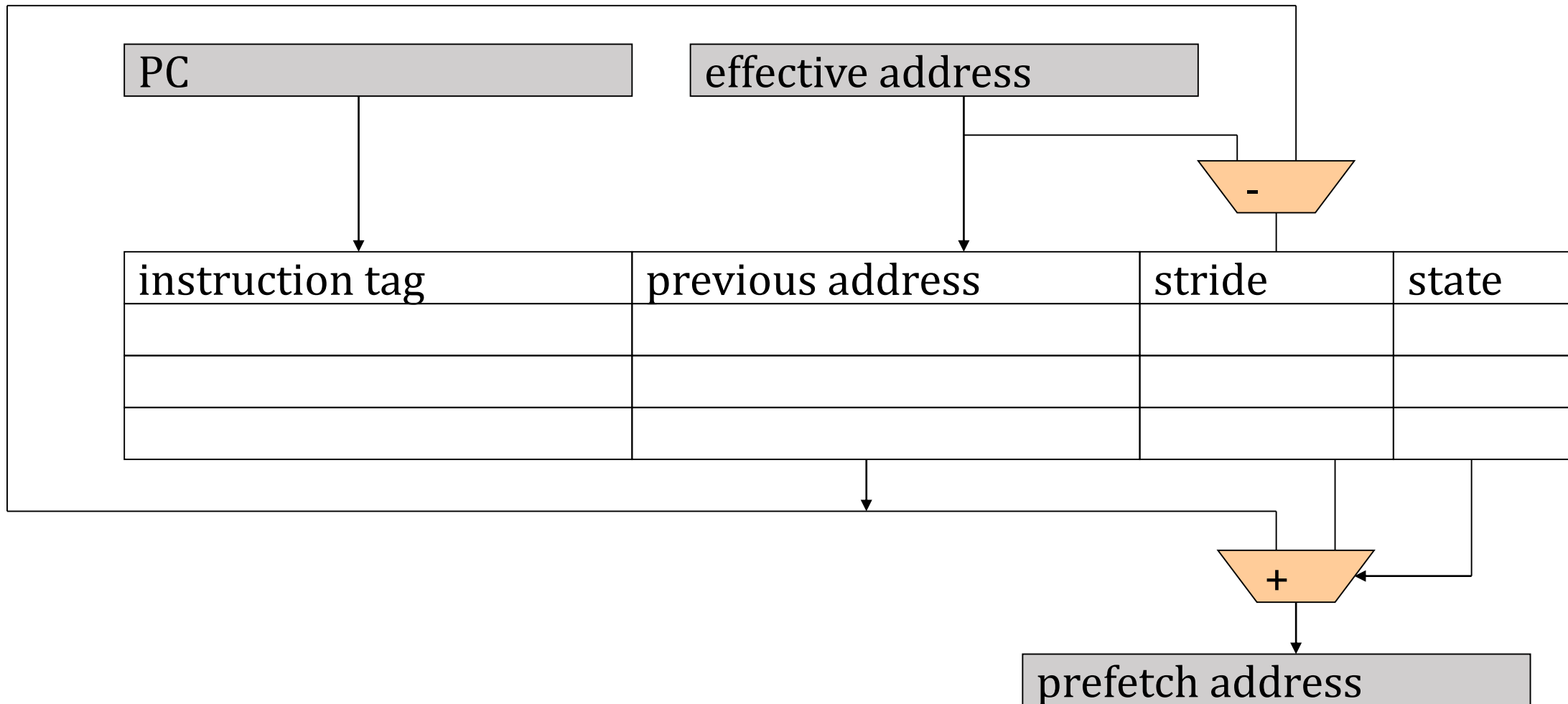
```
  sum += c[i];
```



Cache line size of 64B



# IP-stride prefetcher





# Metrics of interest

Accuracy

Coverage

Timeliness