CS223 : Computer Architecture & Organization

Lecture 21 [30.03.2022]

Optimizing Cache Memory Access Time

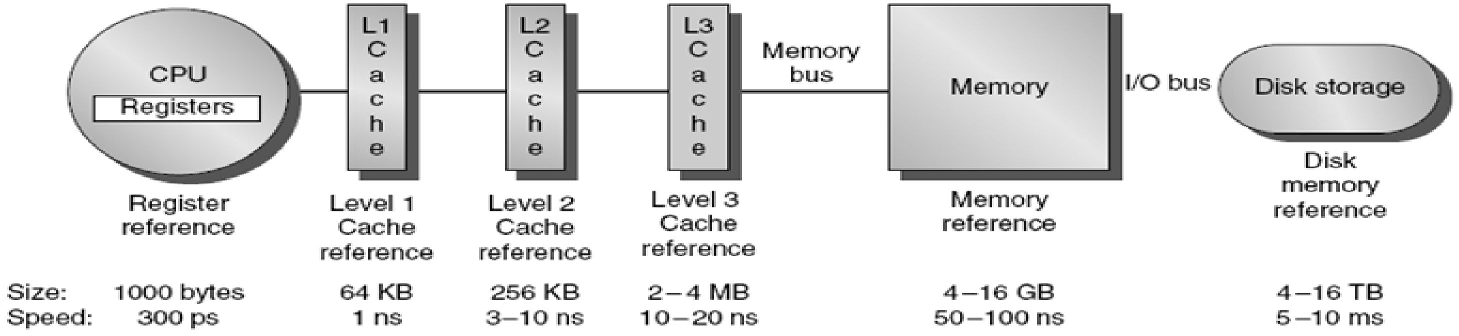
Dr. John Jose

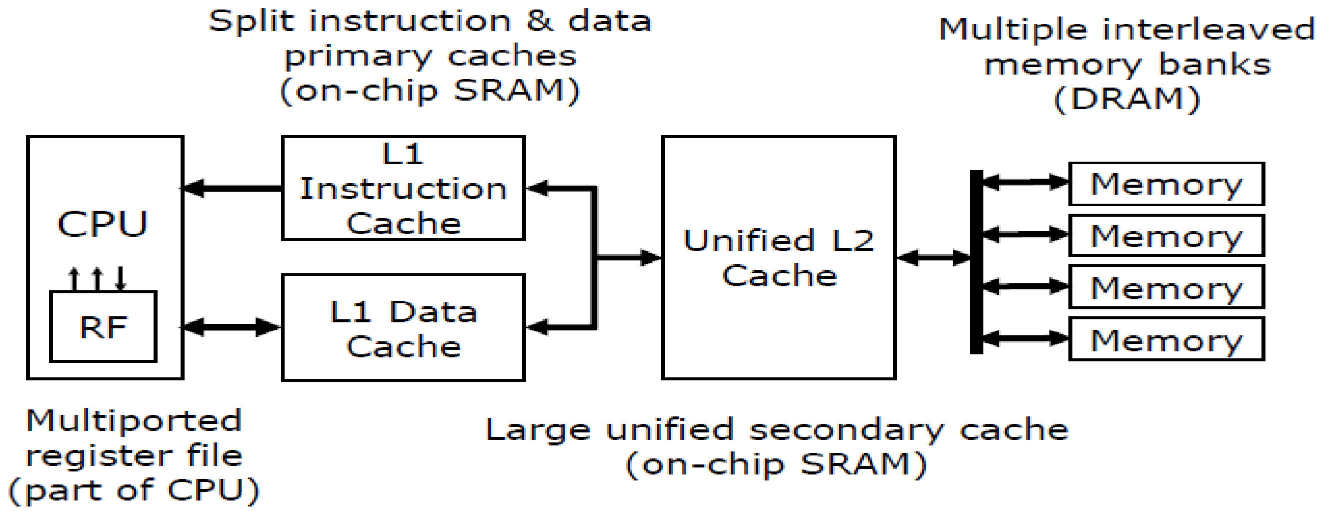
Associate Professor

Department of Computer Science & Engineering

Indian Institute of Technology Guwahati, Assam.

Memory Hierarchy



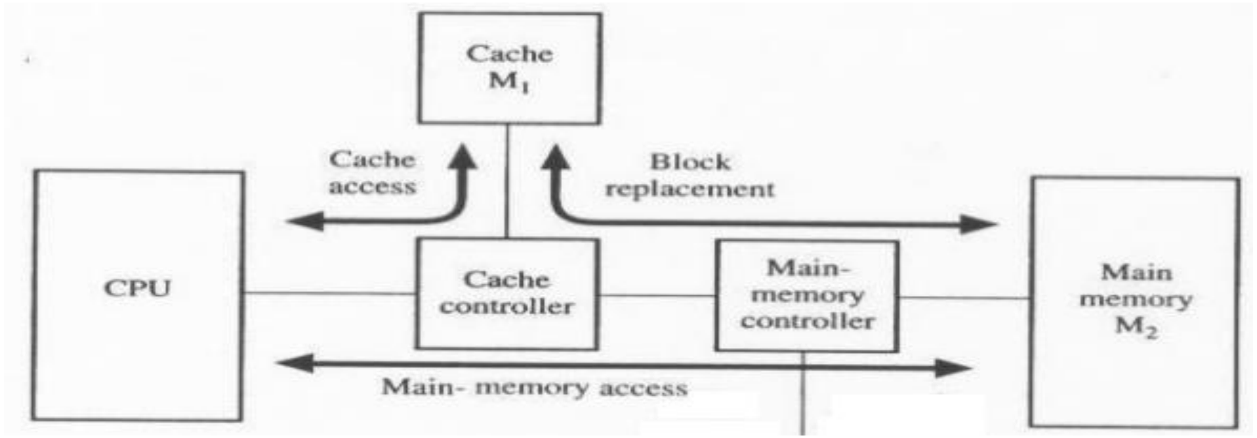

Split instruction & data primary caches (on-chip SRAM)
Multiple interleaved memory banks (DRAM)
CPU
RF
L1 Instruction Cache
L1 Data Cache
Unified L2 Cache
Memory
Memory
Memory
Memory
Multiported register file (part of CPU)
Large unified secondary cache (on-chip SRAM)

Four cache memory design choices

- ❖ Where can a block be placed in the cache?
  - – **Block Placement**
- ❖ How is a block found if it is in the upper level?
  - – **Block Identification**
- ❖ Which block should be replaced on a miss?
  - – **Block Replacement**
- ❖ What happens on a write?
  - – **Write Strategy**

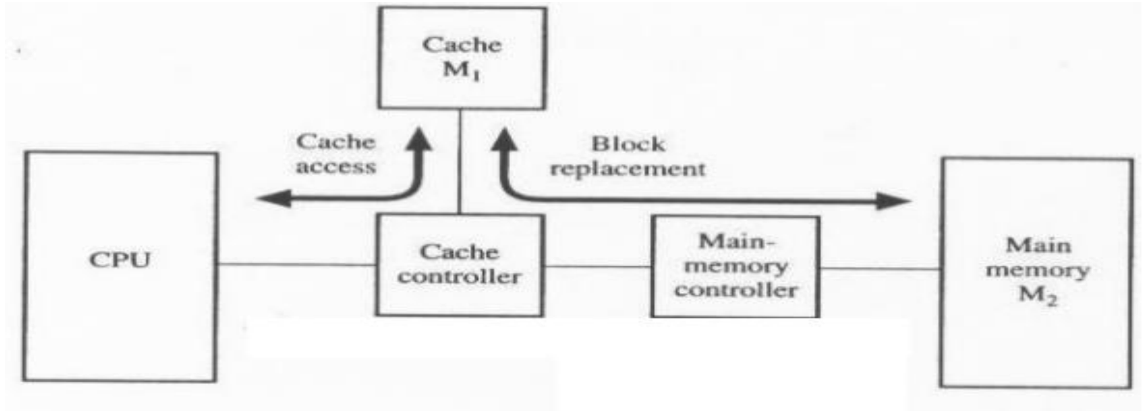
Look-aside vs Look through caches

- **Look-aside cache:** Request from processor goes to cache and main memory in parallel
- Cache and main memory both see the bus cycle
- On cache hit→ processor loaded from cache, bus cycle terminates; On cache miss: processor & cache loaded from memory in parallel



Look-aside vs Look through caches

- **Look-through cache:** Cache checked first when processor requests data from memory
- On hit→ data loaded from cache: On miss→ cache loaded from memory, then processor loaded from cache

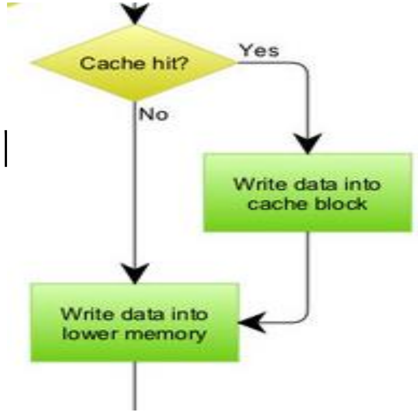


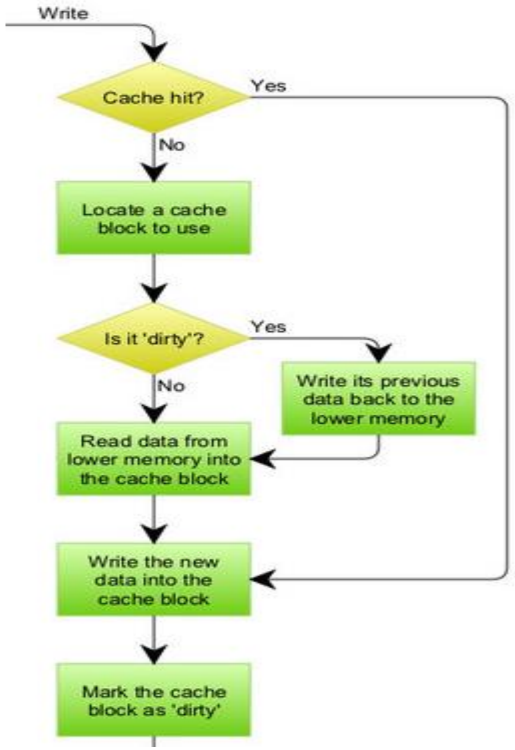
Write strategy

- **Write Hits → Write through vs Write back**
- **Write Miss→ Write allocate vs No-Write allocate**
- **Write through:** The information is written to both the block in the cache and to the main memory
- Read misses do not need to write back evicted line contents
- **Write back:** The information is written only to the block in the cache. The modified cache block is written to main memory only when it is replaced.
- Have to maintain whether block clean or dirty. No extra work on repeated writes; only the latest value on eviction gets updated in main memory.

Write strategy

- **Write allocate:** The block is loaded into cache on a write miss.
- Used along with write back caches
- **No-Write allocate:** The block is modified in the main memory but not in cache
- Used along with write th caches

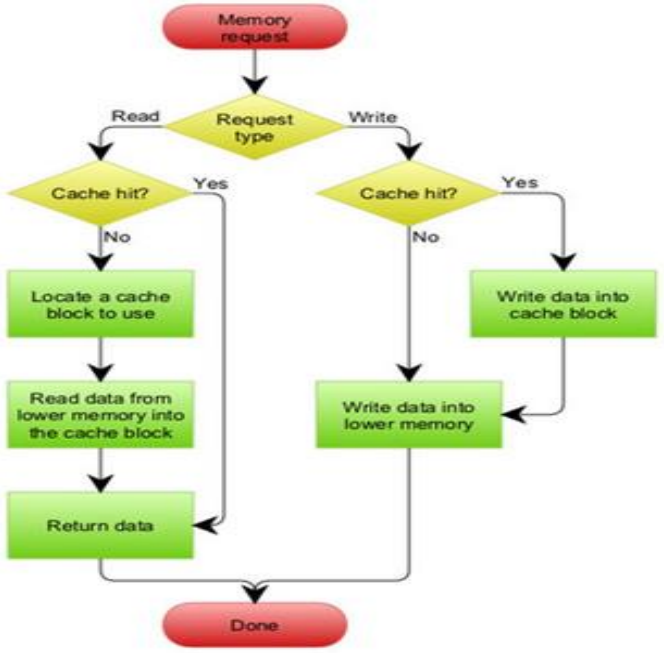




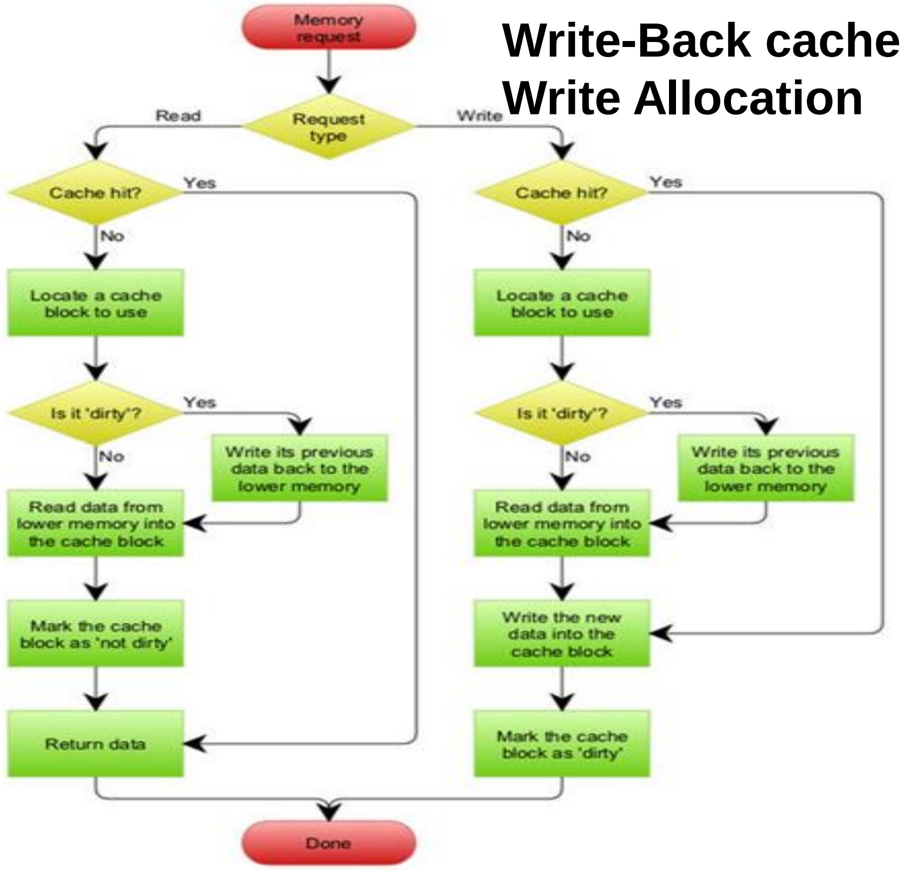
Write
Cache hit?
Yes
No
Locate a cache block to use
Is it 'dirty'?
Yes
No
Write its previous data back to the lower memory
Read data from lower memory into the cache block
Write the new data into the cache block
Mark the cache block as 'dirty'

Write strategy

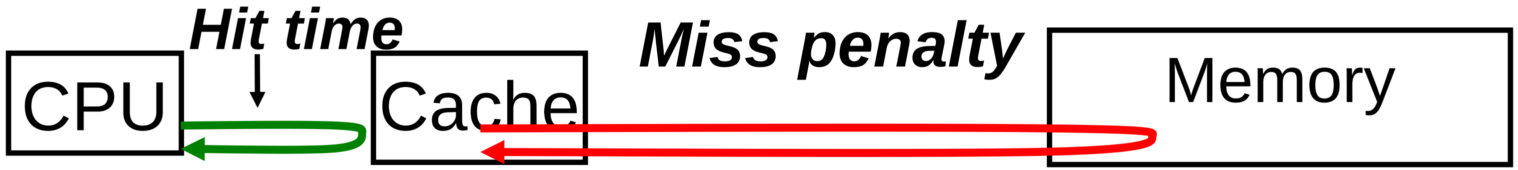
Write-Through cache with No-Write Allocation


Memory request
Request type
Read
Write
Cache hit?
Yes
No
Cache hit?
Yes
No
Locate a cache block to use
Write data into cache block
Read data from lower memory into the cache block
Write data into lower memory
Return data
Done

Write-Back cache with Write Allocation


Memory request
Request type
Read
Write
Cache hit?
Yes
No
Cache hit?
Yes
No
Locate a cache block to use
Locate a cache block to use
Is it 'dirty'?
Yes
No
Is it 'dirty'?
Yes
No
Write its previous data back to the lower memory
Write its previous data back to the lower memory
Read data from lower memory into the cache block
Read data from lower memory into the cache block
Mark the cache block as 'not dirty'
Write the new data into the cache block
Return data
Mark the cache block as 'dirty'
Done

Accessing Cache Memory



Average memory access time = Hit time + (Miss rate × Miss penalty)

- **Hit Time:** Time to find the block in the cache and return it to processor *[indexing, tag comparison, transfer].*
- **Miss Rate:** Fraction of cache access that result in a miss.
- **Miss Penalty:** Number of additional cycles required upon encountering a miss to fetch a block from the next level of memory hierarchy.

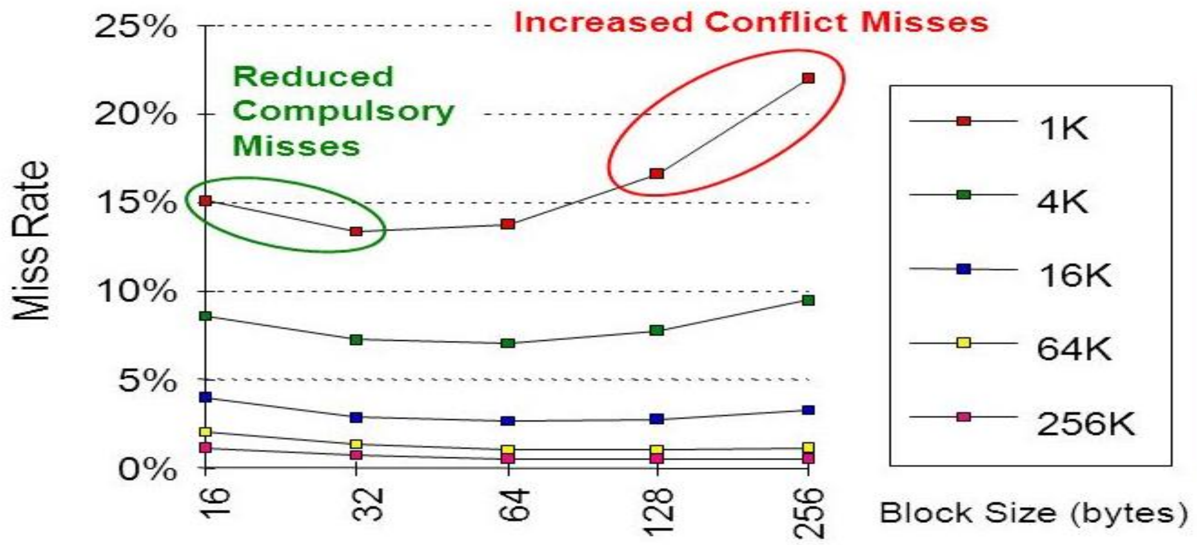
How to optimize cache ?

- Reduce Average Memory Access Time
- AMAT= Hit Time + Miss Rate x Miss Penalty
- Motives
    - Reducing the miss rate
    - Reducing the miss penalty
    - Reducing the hit time

Larger Block Size

- Larger block size to reduce miss rate
- Advantages
  - Utilize spatial locality
  - Reduces compulsory misses
- Disadvantages
  - Increases miss penalty
  - More time to fetch a block to the cache [bus width issue]
  - Increases conflict misses
  - More number of blocks will be mapped to the same location
  - May bring useless data and evict useful data [pollution]

Larger Block Size

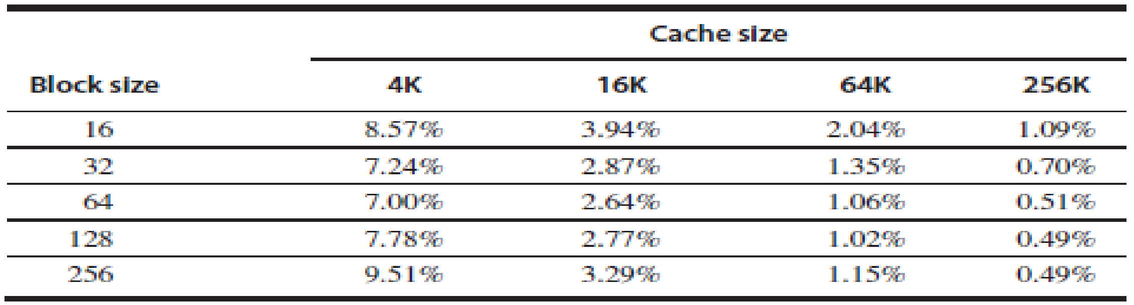


64-byte blocks are common in L1 caches

128-byte block are common in L2 caches

Larger Caches

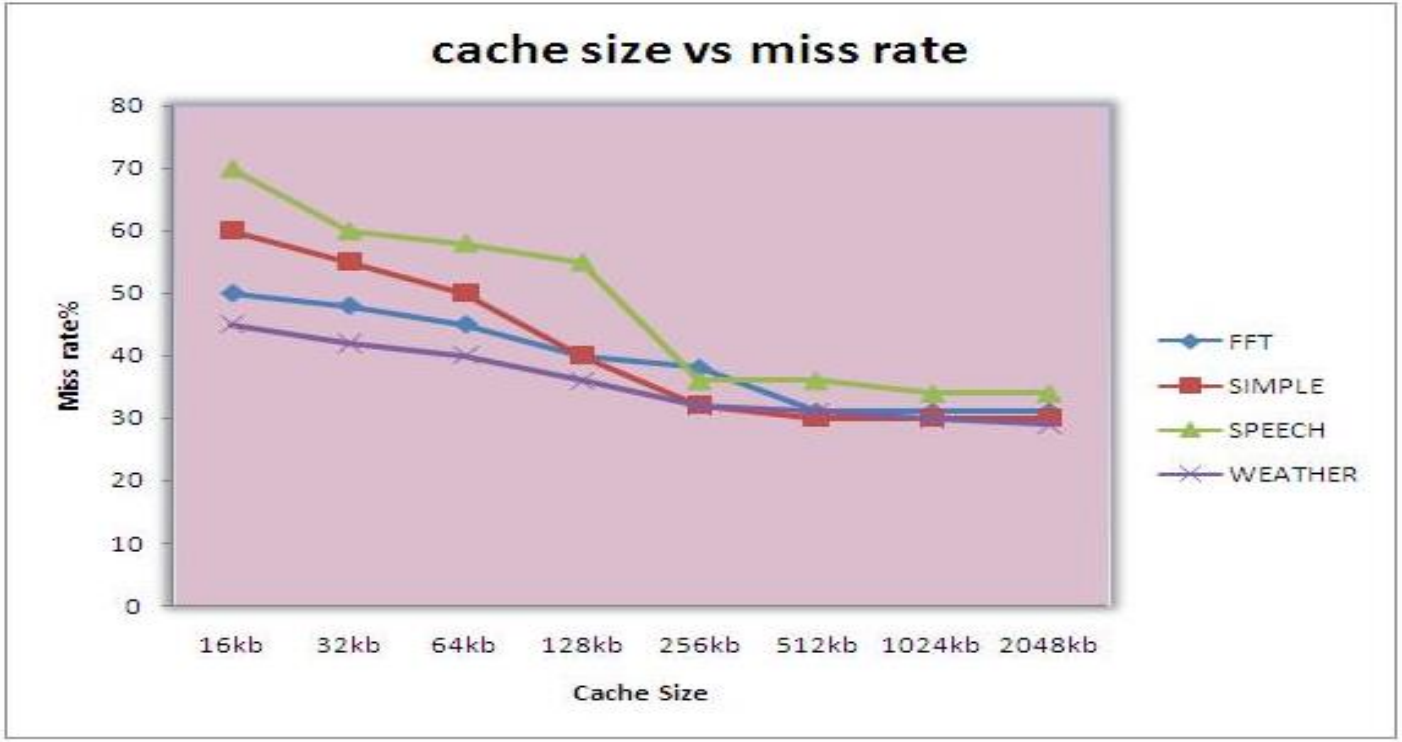
- **Larger cache to reduce miss rate**
- **Advantages**
  - Reduces capacity misses
  - Can accommodate larger memory footprint



	Cache size			
Block size	4K	16K	64K	256K
16	8.57%	3.94%	2.04%	1.09%
32	7.24%	2.87%	1.35%	0.70%
64	7.00%	2.64%	1.06%	0.51%
128	7.78%	2.77%	1.02%	0.49%
256	9.51%	3.29%	1.15%	0.49%

- **Drawbacks**
  - Longer hit time
  - Higher cost, area and power

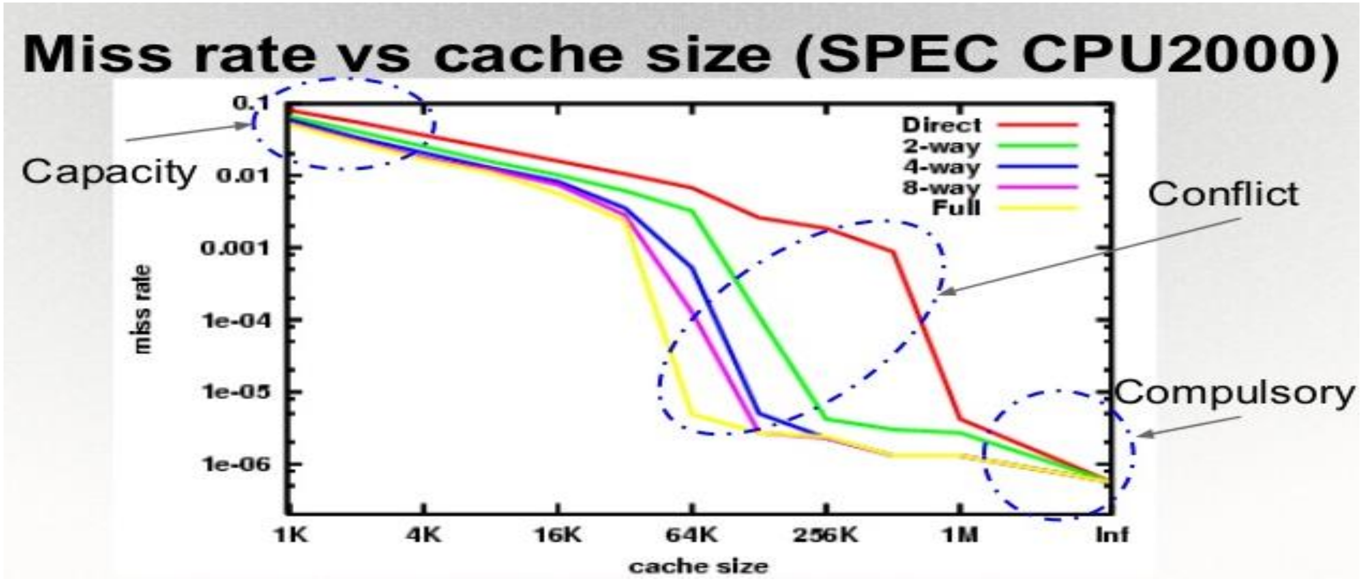
Larger Caches



Higher Associativity

- **Higher associativity to reduce miss rate**
  - Fully associative caches are the best, but high hit time.
  - So increase the associativity to an optimal possible level
- **Advantages**
  - Reduce conflict miss
  - Reduce miss rate and eviction rate
- **Drawbacks**
  - Increase in the hit time
  - Complex design than direct mapped
  - More time to search in the set (tag comparison time)

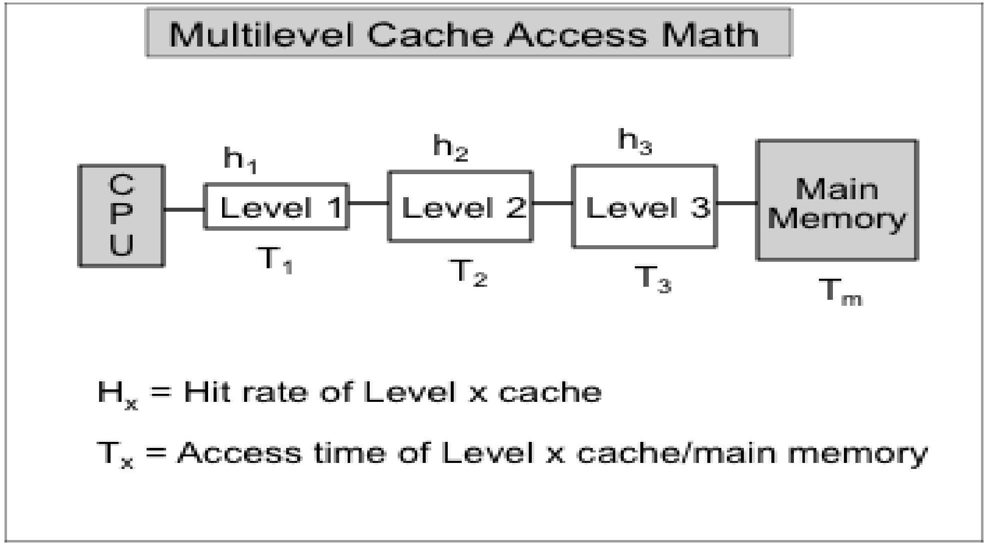
AMAT vs cache associativity



Multilevel caches

- **Multilevel caches to reduce miss penalty**
- **Caches should be faster** to keep pace with the speed of processors, **AND cache should be larger** to overcome the widening gap between the processor and main memory
- Add another level of cache between the cache and memory.
- The first-level cache (L1) can be small enough to match the clock cycle time of the fast processor. [Low hit time]
- The second-level cache (L2) can be large enough to capture many accesses that would go to main memory, thereby lessening the effective miss penalty. [Low miss rate]

Multilevel caches


Multilevel Cache Access Math
h_1
h_2
h_3
CPU
Level 1
Level 2
Level 3
Main Memory
T_1
T_2
T_3
T_m
H_x = Hit rate of Level x cache
T_x = Access time of Level x cache/main memory

$$\text{Average memory access time} = \text{Hit time}_{L1} + \text{Miss rate}_{L1} \times \text{Miss penalty}_{L1}$$

$$\text{Miss penalty}_{L1} = \text{Hit time}_{L2} + \text{Miss rate}_{L2} \times \text{Miss penalty}_{L2}$$

$$\text{Average memory access time} = \text{Hit time}_{L1} + \text{Miss rate}_{L1} \times (\text{Hit time}_{L2} + \text{Miss rate}_{L2} \times \text{Miss penalty}_{L2})$$

Reference

- **Computer Architecture-A Quantitative Approach** (5th edition), John L. Hennessy, David A. Patterson, Morgan Kaufman.
- Chapter 2: Memory Hierarchy Design
  - Section 2.1: Introduction
- Appendix B: Review of Memory Hierarchy
  - Section B.2: Cache Performance
  - Section B.3: Basic Cache Optimizations
- **NPTEL Video Link:** **https://tinyurl.com/yf94dnby**

johnjose@iitg.ac.in

http://www.iitg.ac.in/johnjose/