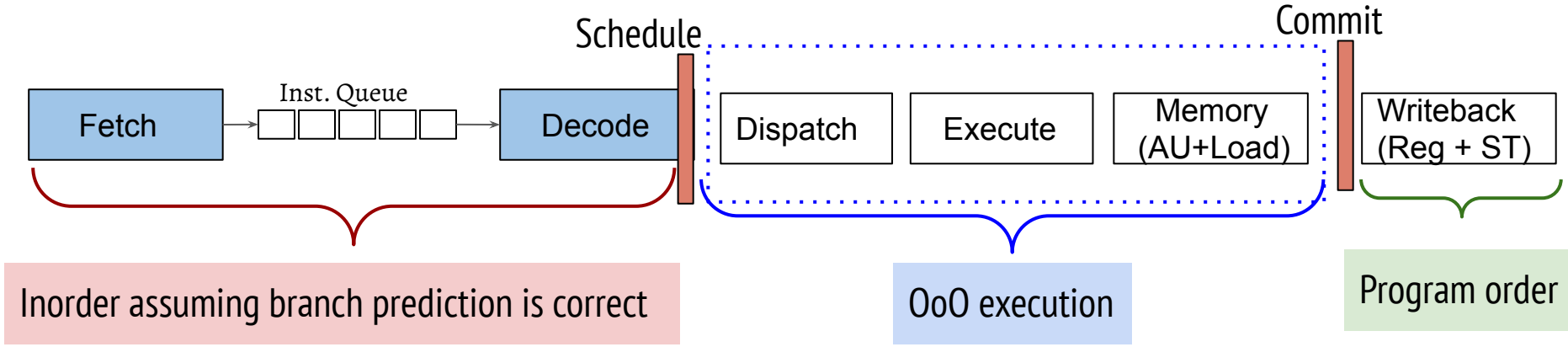
Computer Architecture

Memory Hierarchy

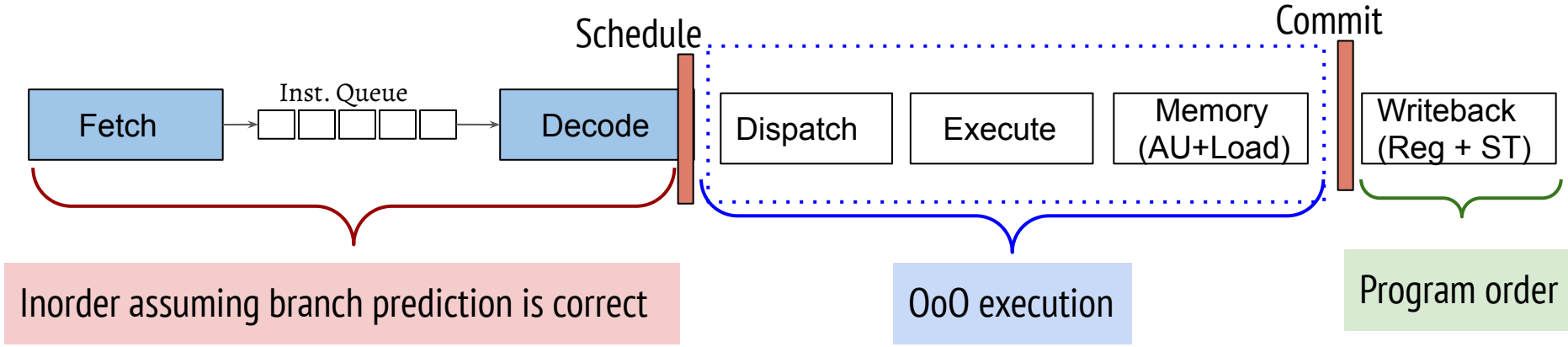
Debadatta Mishra, CSE, IITK

Superscalar Frontend and Memory Interactions



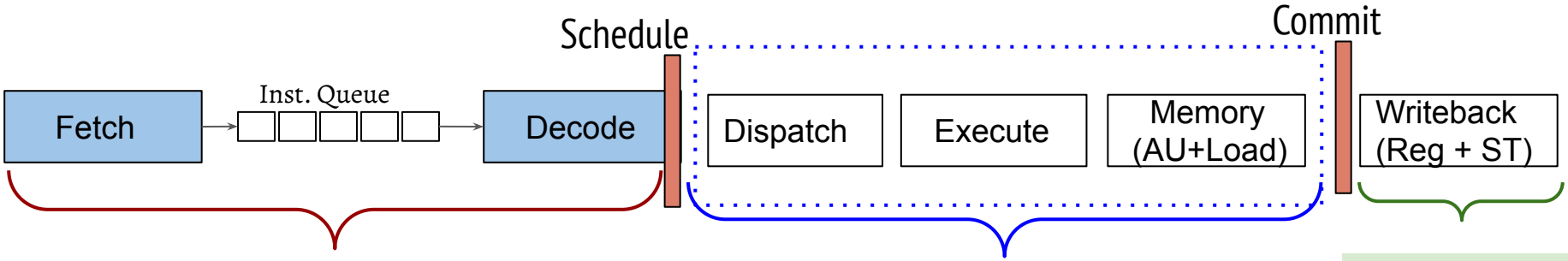
- What are the memory operations in each phase?
- How much delay can be tolerated?
- Implications of memory access bottleneck?

Superscalar Frontend and Memory Interactions



- Operation: Instruction (load)
- At any point of time #of instructions >= issue width
- Resource underutilization, pipeline stall ⟶ reduced IPC

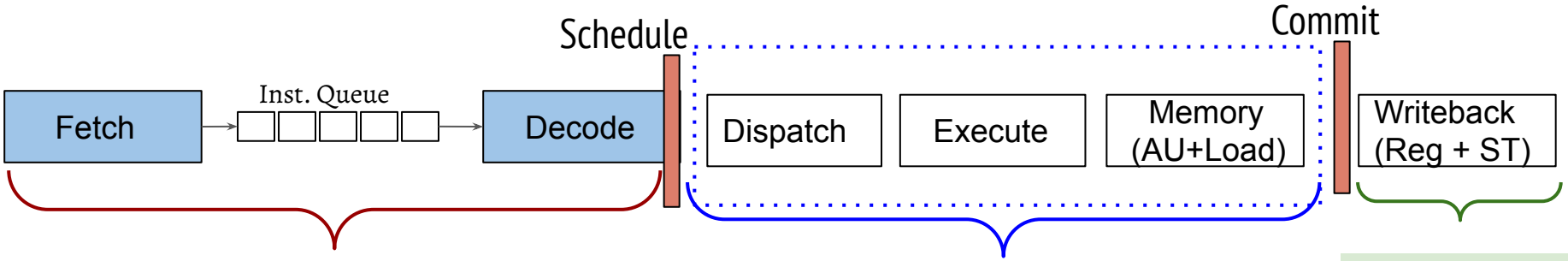
Superscalar Frontend and Memory Interactions



- Operation: Instruction (load)
- At any point of time #of instructions >= issue width
- Resource underutilization, pipeline stall ⟶ reduced IPC

- Operation: Data (load)
- No FU is idle when there are instructions in the corresponding reservation station
- RS and/or ROB full ⟶ Stall ⟶ Reduced IPC

Superscalar Frontend and Memory Interactions



- Operation: Instruction (load)
- At any point of time #of instructions >= issue width
- Resource underutilization, pipeline stall ⟶ reduced IPC

- Operation: Data (load)
- No FU is idle when there are instructions in the corresponding reservation station
- RS and/or ROB full ⟶ Stall ⟶ Reduced IPC

- Operation: Data (store)
- No instructions are waiting to be committed because of a leading store
- ROB full ⟶ Stall ⟶ Reduced IPC

Why not use only registers?

Memory Type	Approx. # of transistors required to store one bit	Access time	Usage
Flip-flop	20	Fast	Registers
SRAM	6	Medium	Cache and Registers
DRAM	1 (+1 capacitor)	slow	Primary Memory

- Faster is more costly because more #of transistors, less storage density, more power
- If money is not an issue, why not design my storage entirely using SRAM?

Why not use only registers?

Memory Type	Approx. # of transistors required to store one bit	Access time	Usage
Flip-flop	20	Fast	Registers
SRAM	6	Medium	Cache and Registers
DRAM	1 (+1 capacitor)	slow	Primary Memory

- Faster is more costly because more #of transistors, less storage density, more power
- If money is not an issue, why not design my storage entirely using SRAM?
  - Access latency increases with size

Typical Memory Hierarchy

Memory Element	Typical Capacity	Access time
Register File	~1KB	0.3 ns
L1 Cache	64KB	1ns
L2 Cache	256KB	3ns - 10ns
L3 Cache	2MB - 4MB	10ns - 20ns

- As much as possible, serve the data from the levels closer to the processor

Common Terminologies (Recap)

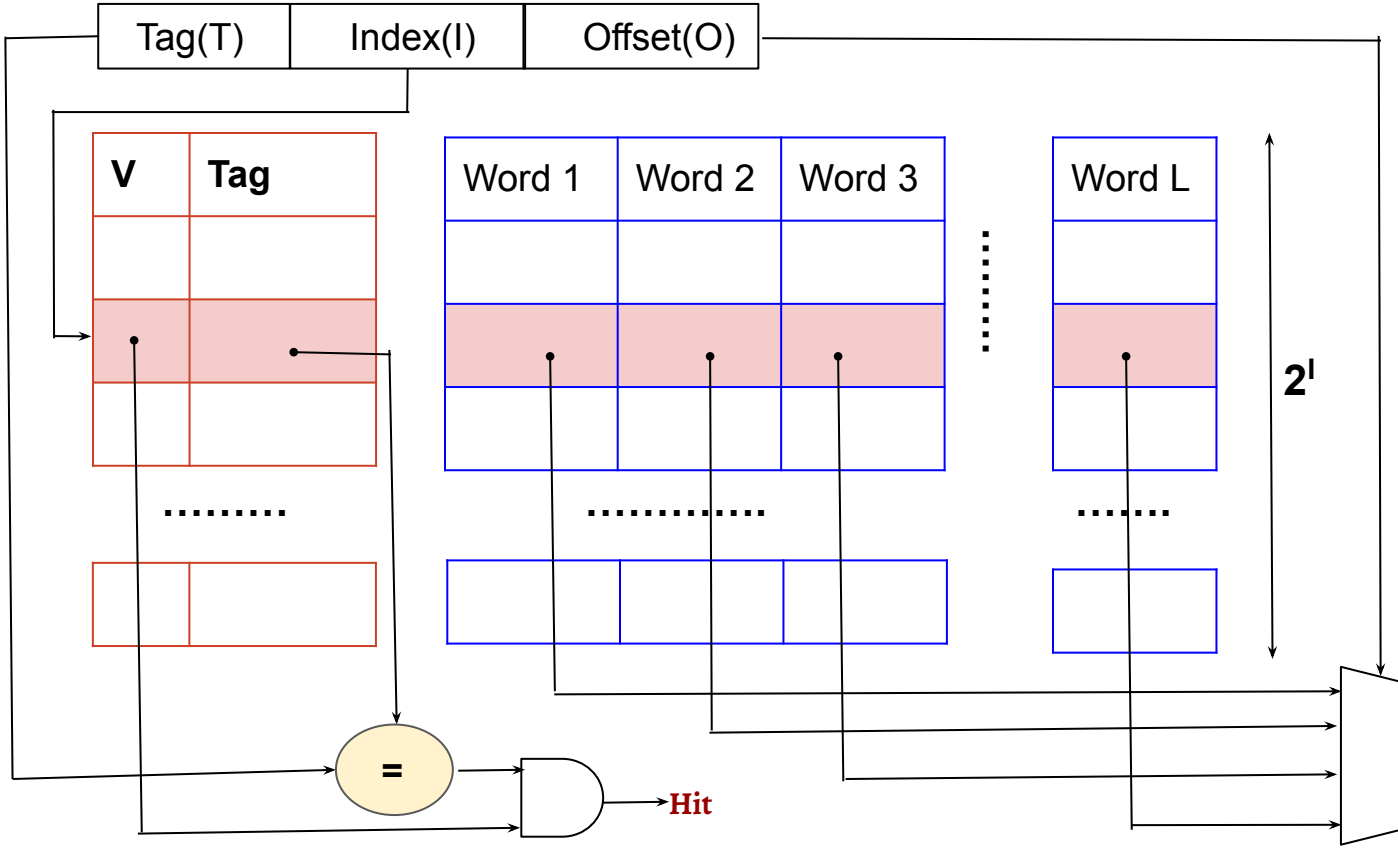
Program execution

- Access locality
- Working set
- Types of access
- Addressing
- Cache hit
- Cache miss

Mechanisms and Policies

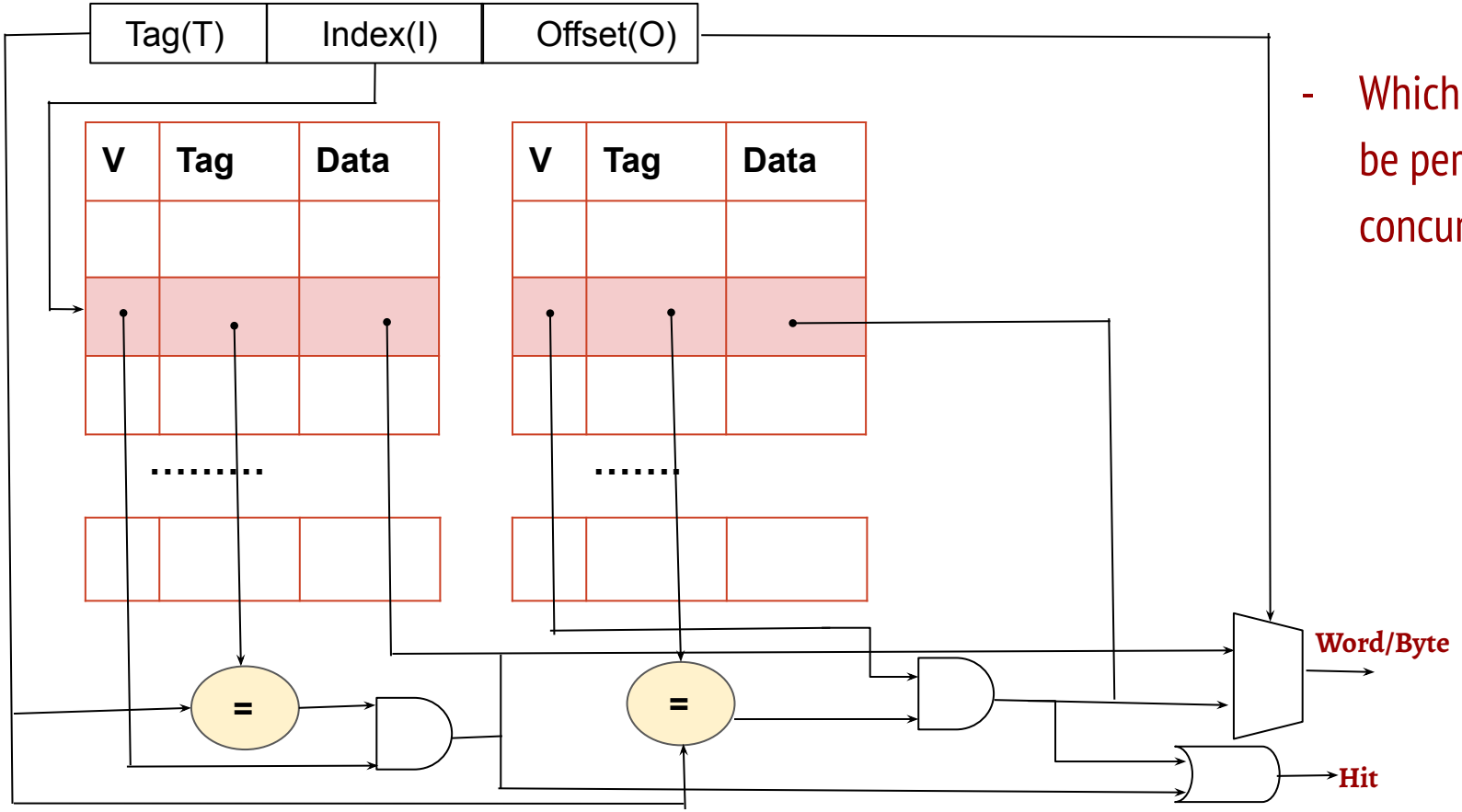
- Cache blocks
- Block placement: direct mapped, set-associative, fully associative
- Block Identification: index, tag, valid bit,
- Block replacement policies
- Write strategy: write-through, write-back, dirty bit, write allocate

Direct-mapped Cache



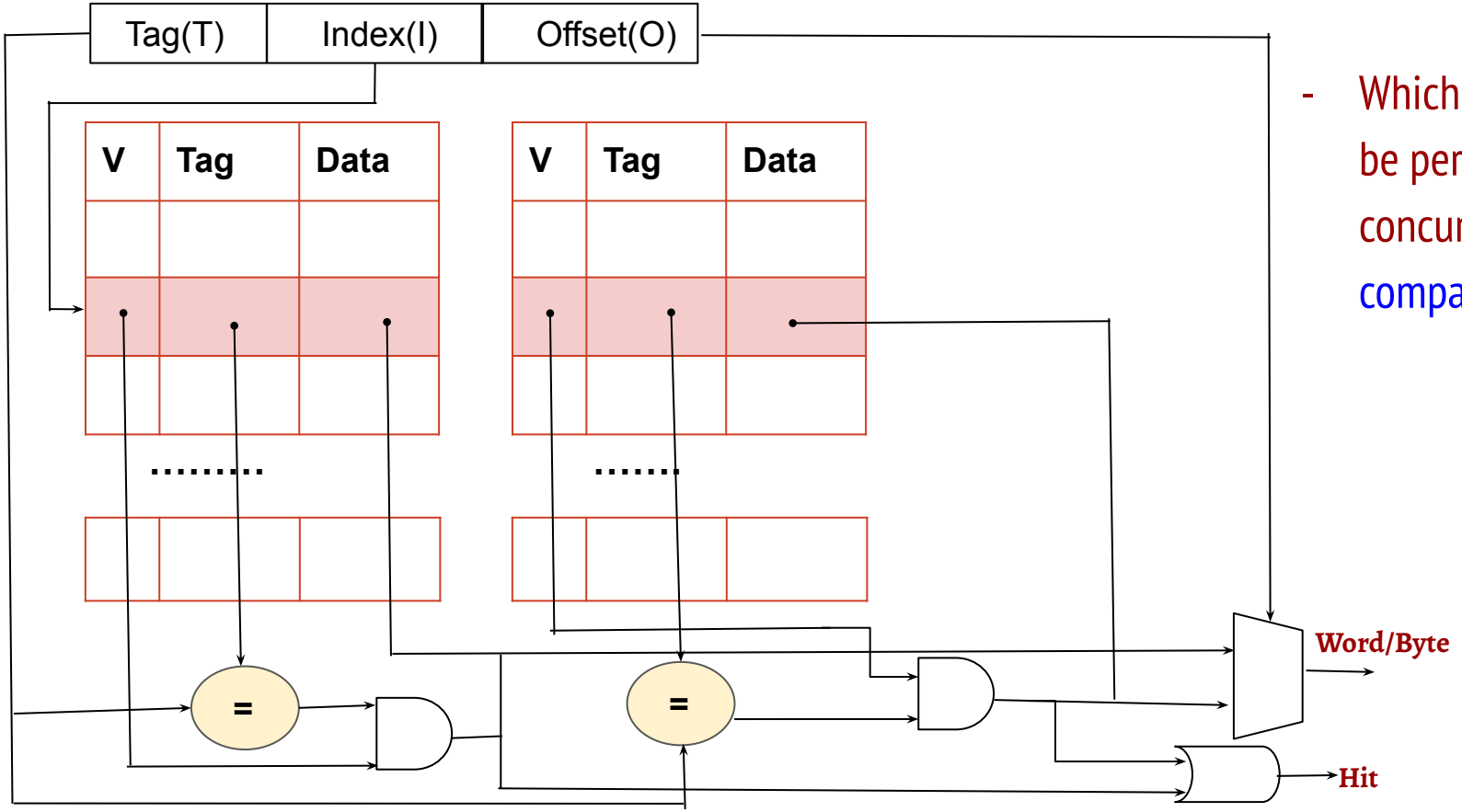
- Valid bit
- Dirty bit?
- Design parameters
  - Block size
  - Cache size
  - Mapping function
  - Index bits
  - Tag bits

2-way Set Associative Cache



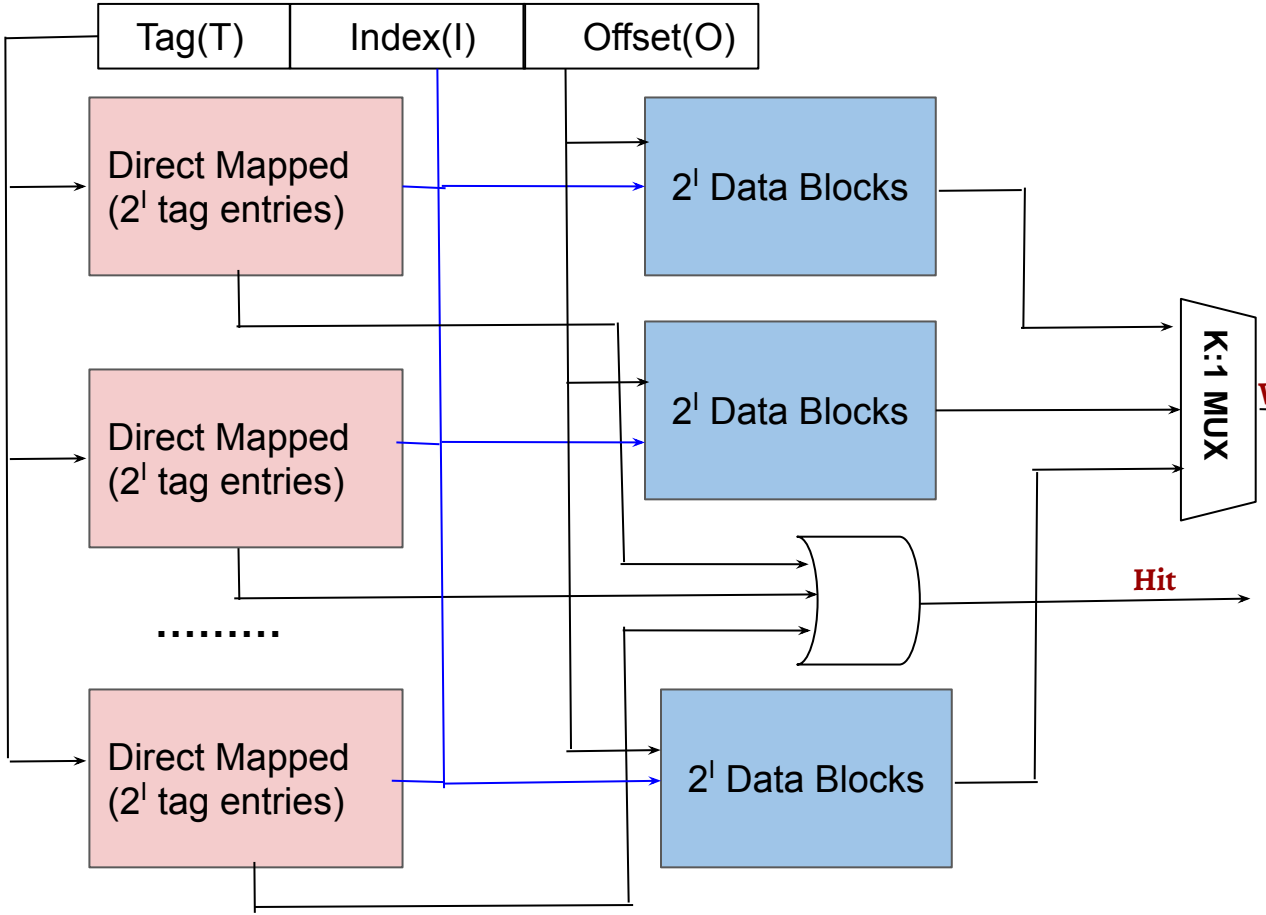
- Which operations can be performed concurrently?

2-way Set Associative Cache



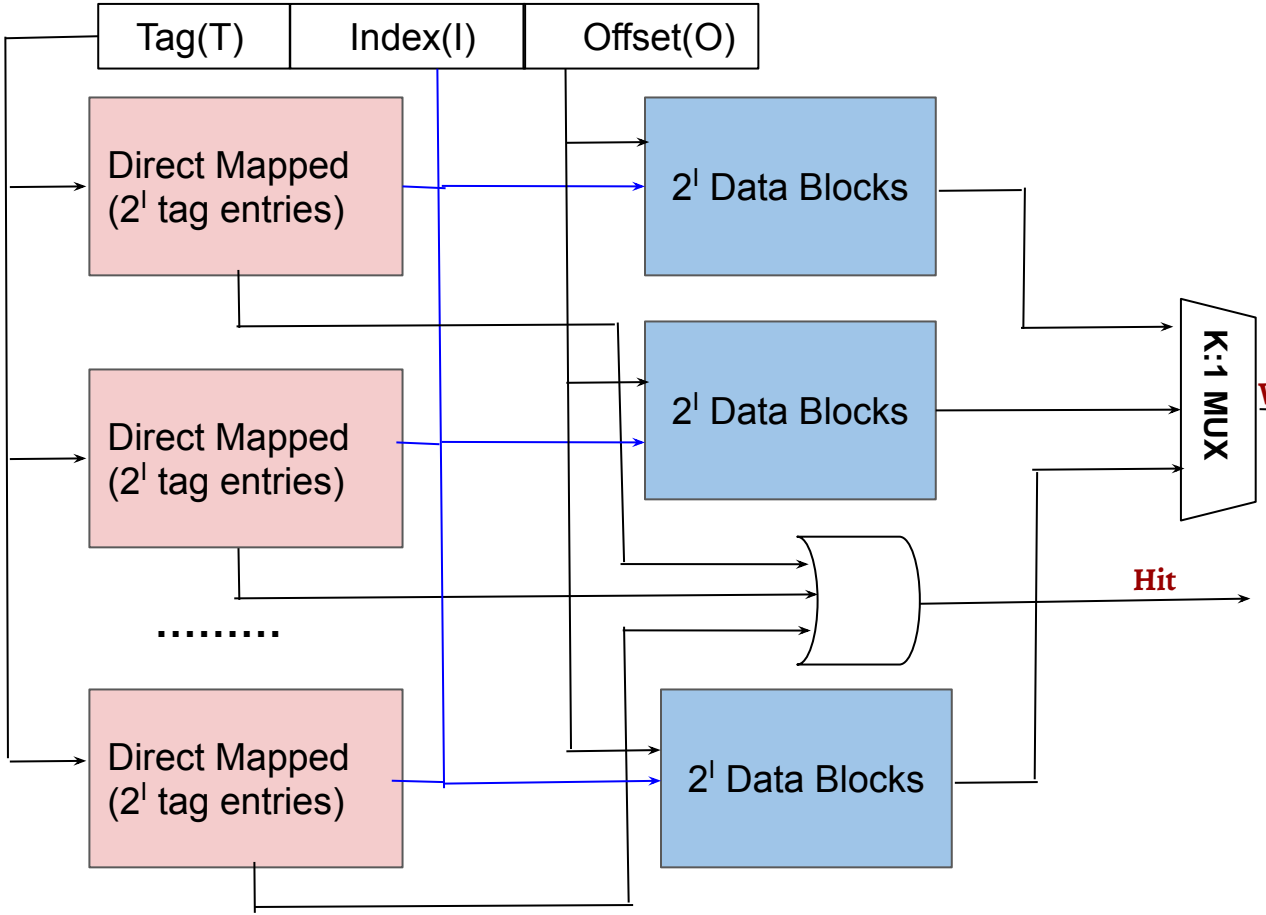
- Which operations can be performed concurrently? Tag comparison, data read

K-way Set Associative Cache



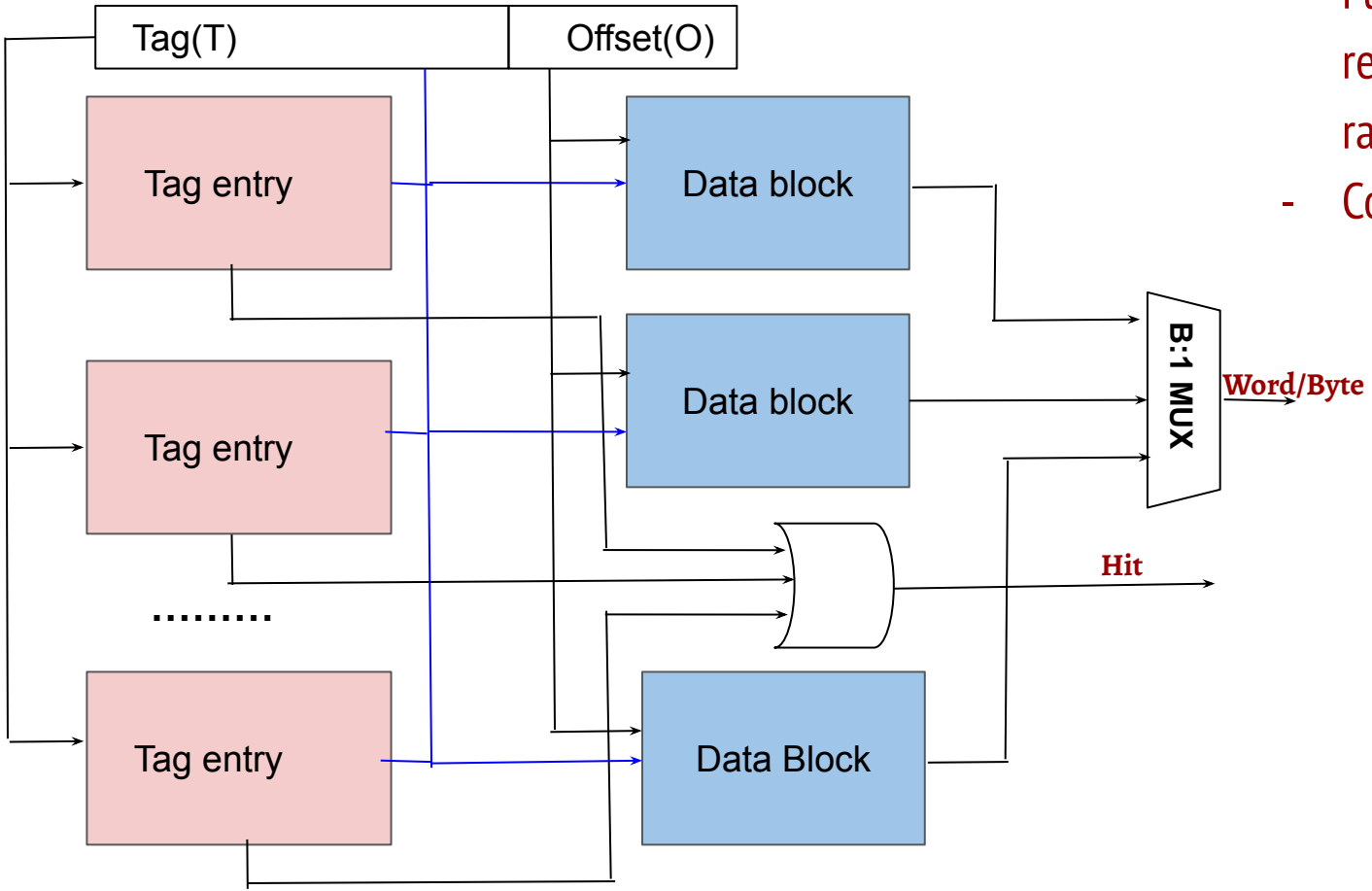
- What should be the value of K?
- What factors determine the limits on K?
- Cache replacement policies (LRU, FIFO ...)
- RRIP, Jaleel et al. (paper review due next week)

K-way Set Associative Cache



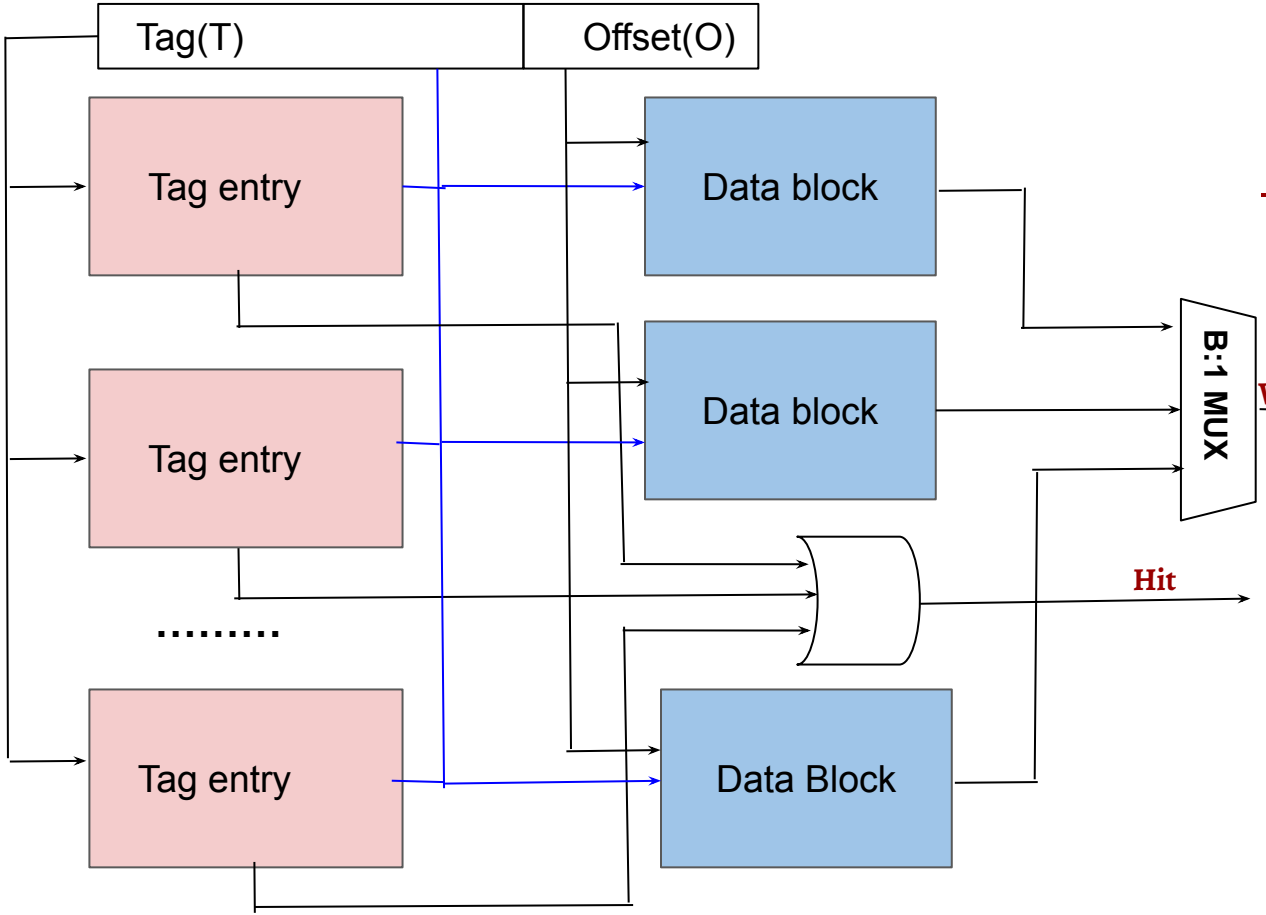
- What should be the value of K? Larger the K, more is the hit latency
- What factors determine the limits on K?
- Cache replacement policies (LRU, FIFO ...)
- RRIP, Jaleel et al. (paper review due next week)

Fully Associative Cache



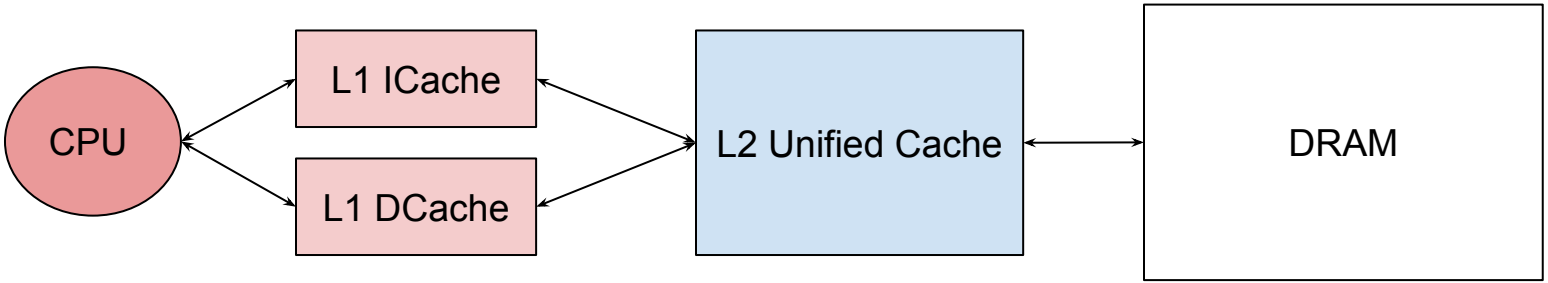
- Fully associative cache results in lowest miss rate (True/False)
- Cost vs. Benefit?

Fully Associative Cache



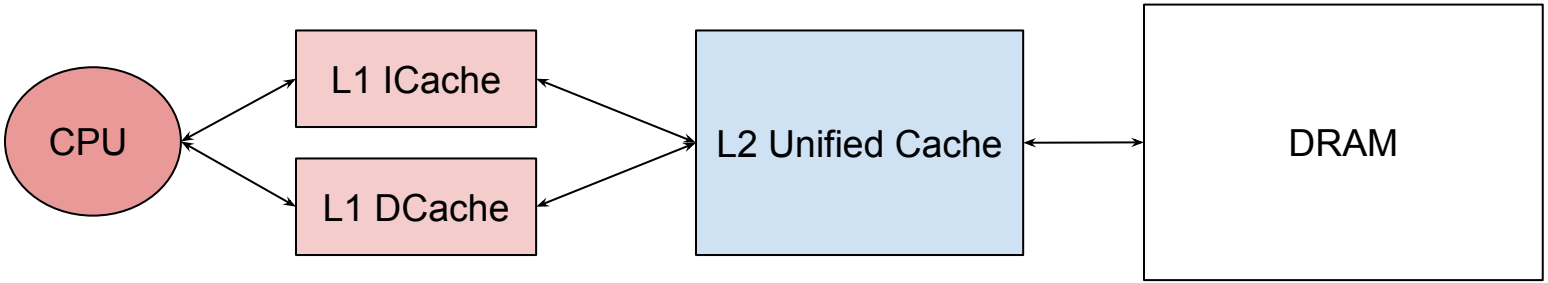
- Fully associative cache results in lowest miss rate (True/False)
- Cost vs. Benefit? Increasing associativity after a point return on cost is diminished

Multi-level Caches



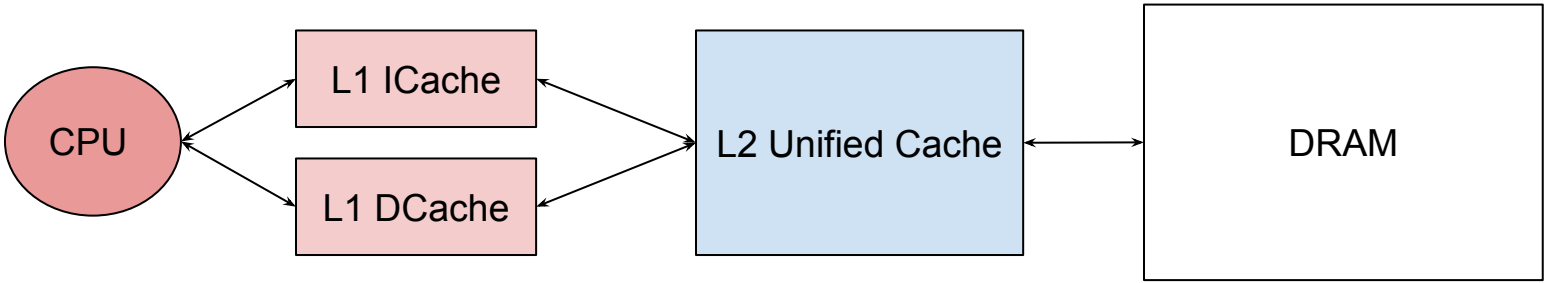
- Why design a multi-level cache, why not a single big cache?
- Typically, L1 instruction and data caches are separate, why?
- How to handle write operations?
- What are the metric of interest?
- Is there any need for bypassing certain levels of caches?

Multi-level Caches



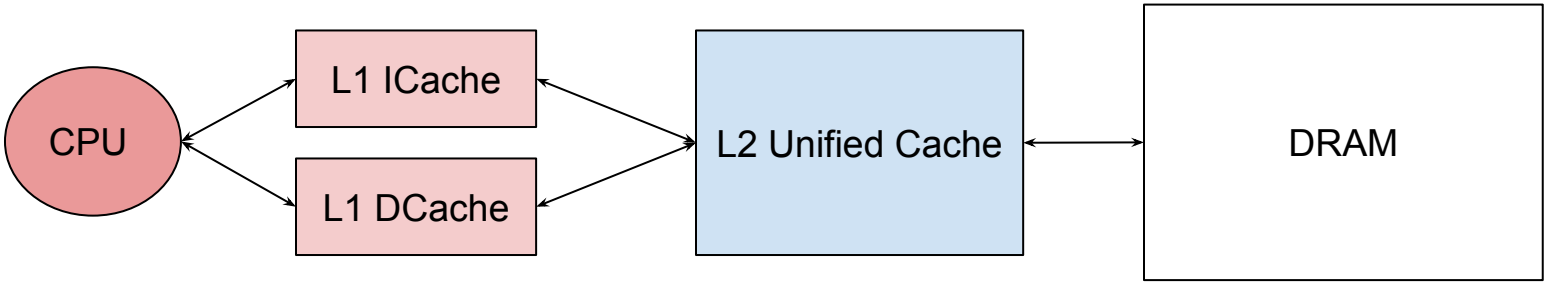
- Why design a multi-level cache, why not a single big cache? Reduce cache hit time (smaller caches closer to CPU) and cache miss penalty (increased capacity down the hierarchy)
- Typically, L1 instruction and data caches are separate, why?
- How to handle write operations?
- Is there any need for bypassing certain levels of caches?
- What are the metric of interest?

Multi-level Caches



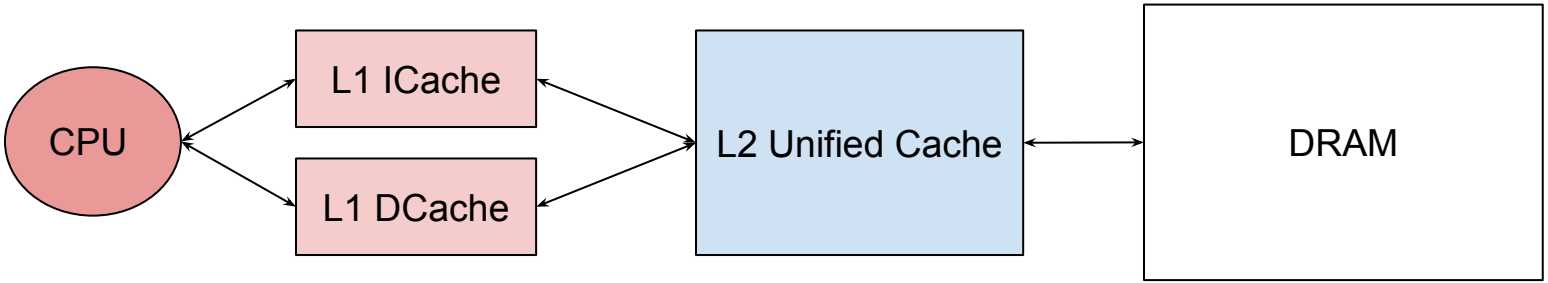
- Why design a multi-level cache, why not a single big cache? Reduce cache hit time (smaller caches closer to CPU) and cache miss penalty (increased capacity down the hierarchy)
- Typically, L1 instruction and data caches are separate, why? IF and data access become independent, avoid thrashing of data and instructions
- How to handle write operations?
- Is there any need for bypassing certain levels of caches?
- What are the metric of interest?

Multi-level Caches



- Why design a multi-level cache, why not a single big cache? Reduce cache hit time (smaller caches closer to CPU) and cache miss penalty (increased capacity down the hierarchy)
- Typically, L1 instruction and data caches are separate, why? IF and data access become independent, avoid thrashing of data and instructions
- How to handle write operations? Cache hit: write through vs. write back, Miss: Write allocate
- Is there any need for bypassing certain levels of caches?
- What are the metric of interest?

Multi-level Caches



- Why design a multi-level cache, why not a single big cache? Reduce cache hit time (smaller caches closer to CPU) and cache miss penalty (increased capacity down the hierarchy)
- Typically, L1 instruction and data caches are separate, why? IF and data access become independent, avoid thrashing of data and instructions
- How to handle write operations? Cache hit: write through vs. write back, Miss: Write allocate
- Is there any need for bypassing certain levels of caches? Page table updates, I/O (MMIO)
- What are the metric of interest?

Metrics related to cache

- Local miss rate = # of misses / # of accesses to the cache
- Global miss rate = # of misses / Total # of memory accesses
- Misses per instruction = # misses / # of instructions (misses per Kilo instruction MPKI)

- Avg. memory access time (AMAT) = Hit time + Miss Rate * Miss Penalty
- AMAT (for two levels) = Hit time + Miss Rate $_1$ * Miss Penalty $_1$ + Miss Rate $_2$ * Miss Penalty $_2$

Classifying cache misses

- Compulsory: First reference misses or cold misses
- How to measure?
- How to reduce?

Classifying cache misses

- Compulsory: First reference misses or cold misses
- How to measure? # of cache misses with infinite cache size
- How to reduce? Hardware/Software Prefetching

- Capacity misses: Non-compulsory misses occurring when the cache is full
- How to measure?
- How to reduce?

Classifying cache misses

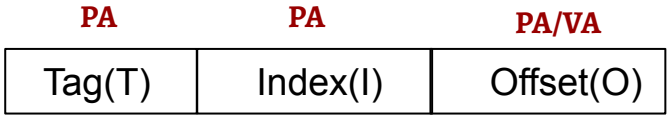
- Compulsory: First reference misses or cold misses
- How to measure? # of cache misses with infinite cache size
- How to reduce? Hardware/Software Prefetching

- Capacity misses: Non-compulsory misses occurring when the cache is full
- How to measure? Cache misses in fully associative cache (with Belady) - Compulsory misses
- How to reduce? Increased cache size, better cache replacement, prefetching
- Conflict misses: In set-associative and direct mapped caches, # of cache misses because of conflict in the set i.e., cache miss on access to a previously evicted block where the eviction was because of the tag array for the set was full
- How to measure?
- How to reduce?

Classifying cache misses

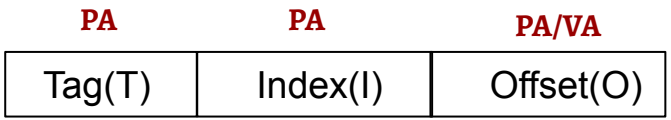
- Compulsory: First reference misses or cold misses
- How to measure? # of cache misses with infinite cache size
- How to reduce? Hardware/Software Prefetching
- Capacity misses: Non-compulsory misses occurring when the cache is full
- How to measure? Cache misses in fully associative cache (with Belady) - Compulsory misses
- How to reduce? Increased cache size, better cache replacement, prefetching
- Conflict misses: In set-associative or direct mapped caches, # of cache misses because of conflict in the set i.e., cache miss on access to a previously evicted block where the eviction was because of a full set
- How to measure? Total cache misses in k-way set associative cache - # of misses in fully associative cache
- How to reduce? Increased associativity, cache prefetching?

Cache Types (Mapping)

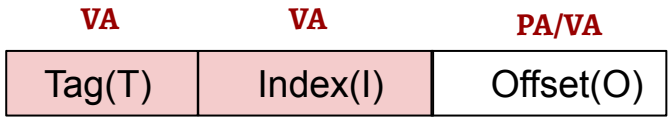


- Physically indexed and physically tagged (PIPT)
- Advantages/disadvantages?

Cache Types (Mapping)

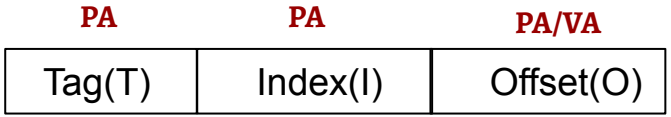


- Physically indexed and physically tagged (PIPT)
- Advantages/disadvantages? Independent of virtual address space. Issue: cache OP possible only after address translation

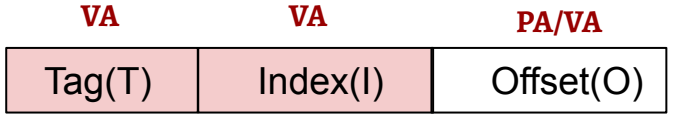


- Virtually indexed and virtually tagged (VIVT)
- Advantages/disadvantages?

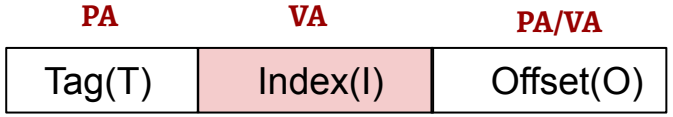
Cache Types (Mapping)



- Physically indexed and physically tagged (PIPT)
- Advantages/disadvantages? Independent of virtual address space. Issue: cache OP possible only after address translation

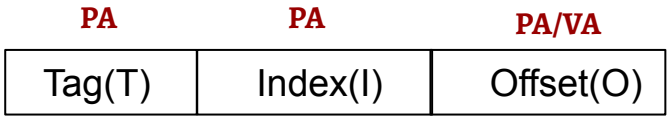


- Virtually indexed and virtually tagged (VIVT)
- Advantages/disadvantages? Cache OP independent of address translation. Issues: protection enforcement, context switches, synonyms or aliases

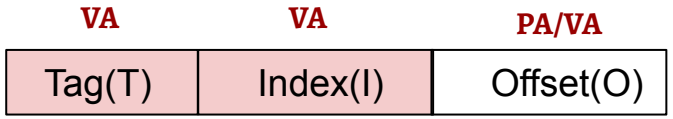


- Virtually indexed and physically tagged (VIPT)

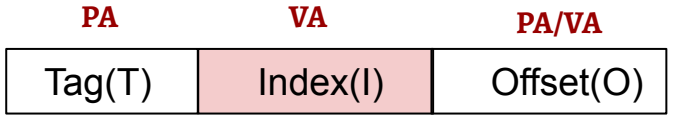
Cache Types (Mapping)



- Physically indexed and physically tagged (PIPT)
- Advantages/disadvantages? Independent of virtual address space. Issue: cache OP possible only after address translation



- Virtually indexed and virtually tagged (VIVT)
- Advantages/disadvantages? Cache OP independent of address translation. Issues: protection enforcement, context switches, synonyms or aliases



- Virtually indexed and physically tagged (VIPT)
- Data read can start after index mapping. Page size depends on block size and # of index bits