Cache Memories (Aside)

CSC 343, Operating Systems

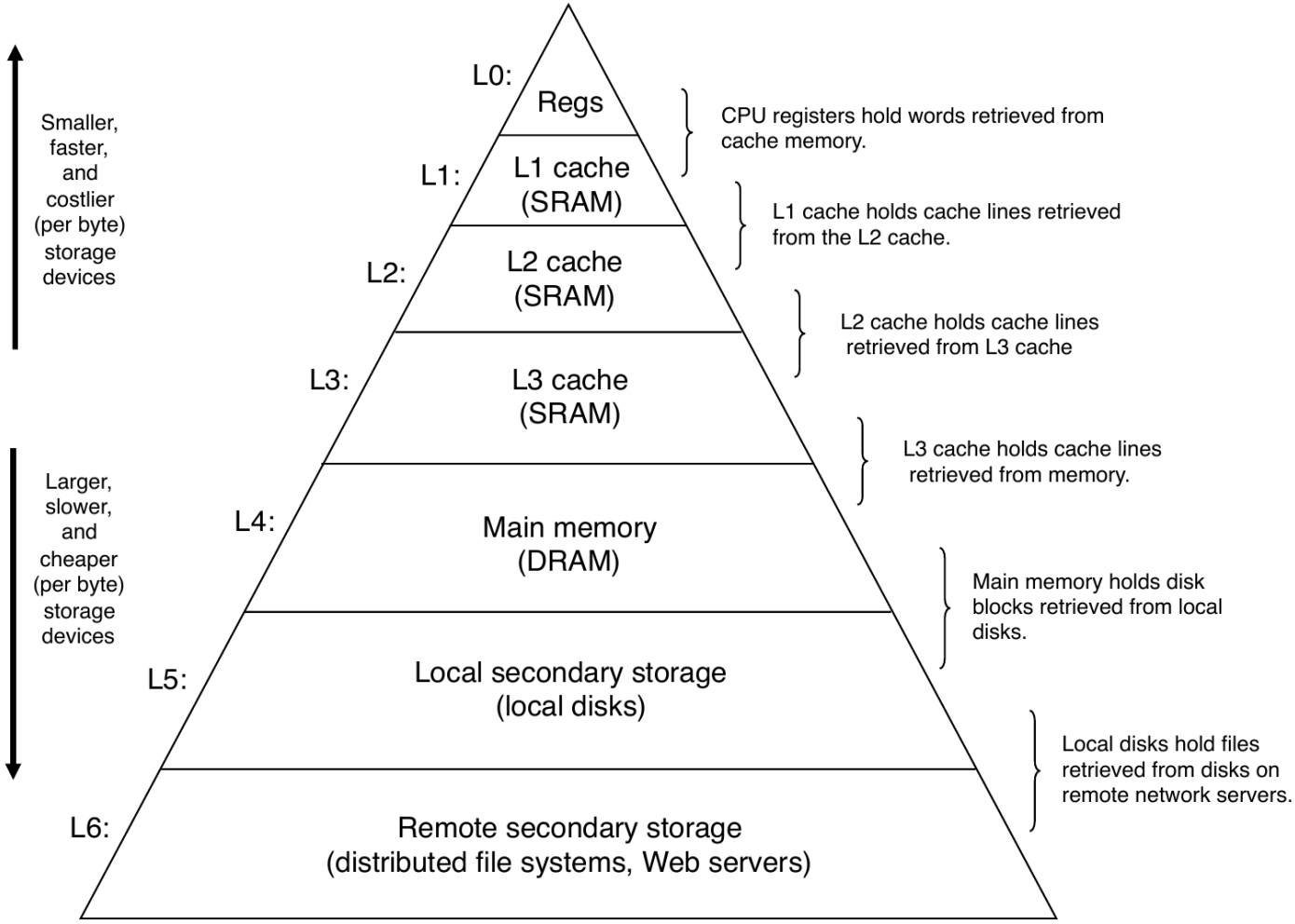
Memory Hierarchies

- Some fundamental and enduring properties of hardware and software:
  - Fast storage technologies cost more per byte, have less capacity, and require more power (heat!).
  - The gap between CPU and main memory speed is widening
  - Well-written programs tend to exhibit good locality
- These fundamental properties complement each other beautifully
- The suggest an approach for organizing memory and storage systems known as a memory hierarchy

Locality

- Principle of Locality: Programs tend to use data and instructions with addresses near or equal to those they have used recently
- Temporal locality:
  - Recently referenced items are likely to be referenced again in the near future
- Spatial locality:
  - Items with nearby addresses tend to be referenced close together in time

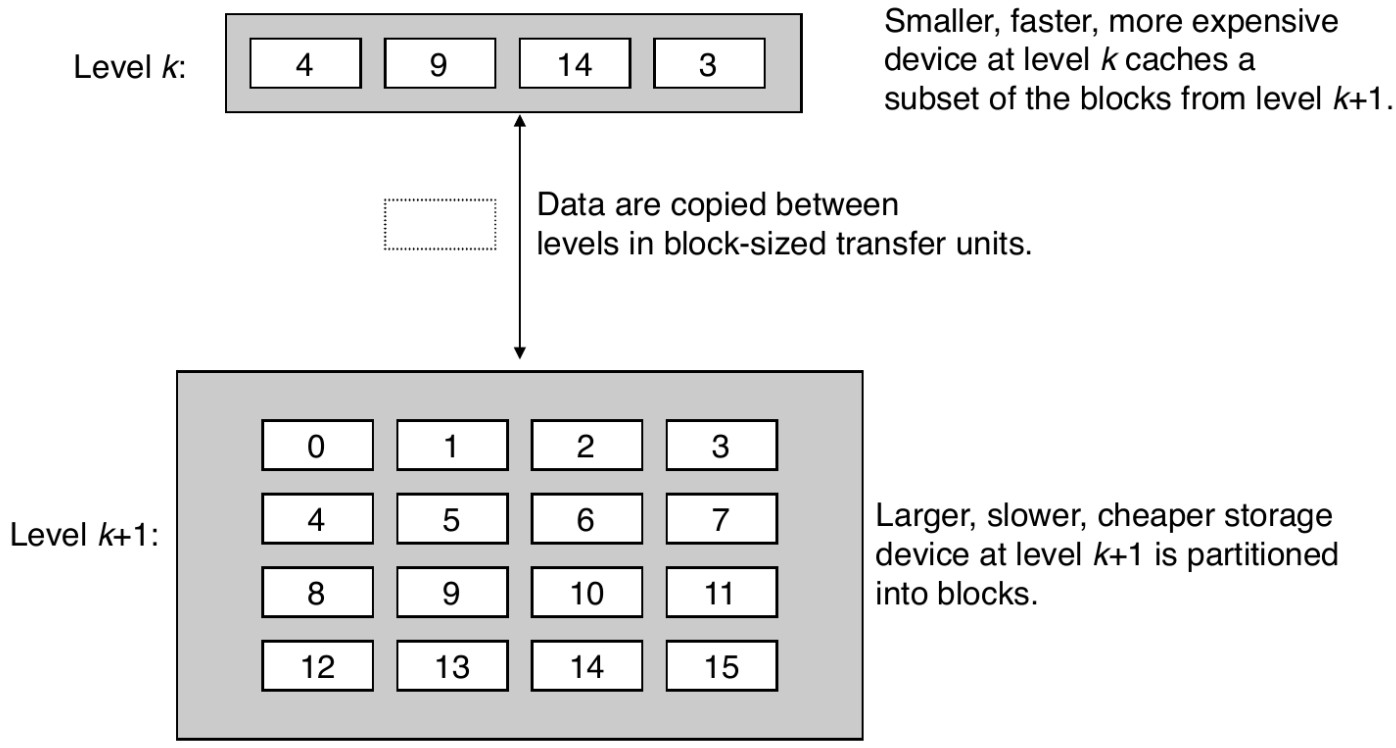
Example Memory Hierarchy



Caches

- Cache: A smaller, faster storage device that acts as a staging area for a subset of the data in a larger, slower device
- Fundamental idea of a memory hierarchy:
  - For each k, the faster, smaller device at level k serves as a cache for the larger, slower device at level $k + 1$
- Why do memory hierarchies work?
  - Because of locality, programs tend to access data at level k more often than they access the data at level $k + 1$
  - Thus, the storage at level $k + 1$ can be slower, larger, and cheaper per bit
- Big Idea (Ideal): The memory hierarchy creates a large pool of storage that costs as much as the cheap storage near the bottom, but serves data to programs at the rate of the fast storage near the top

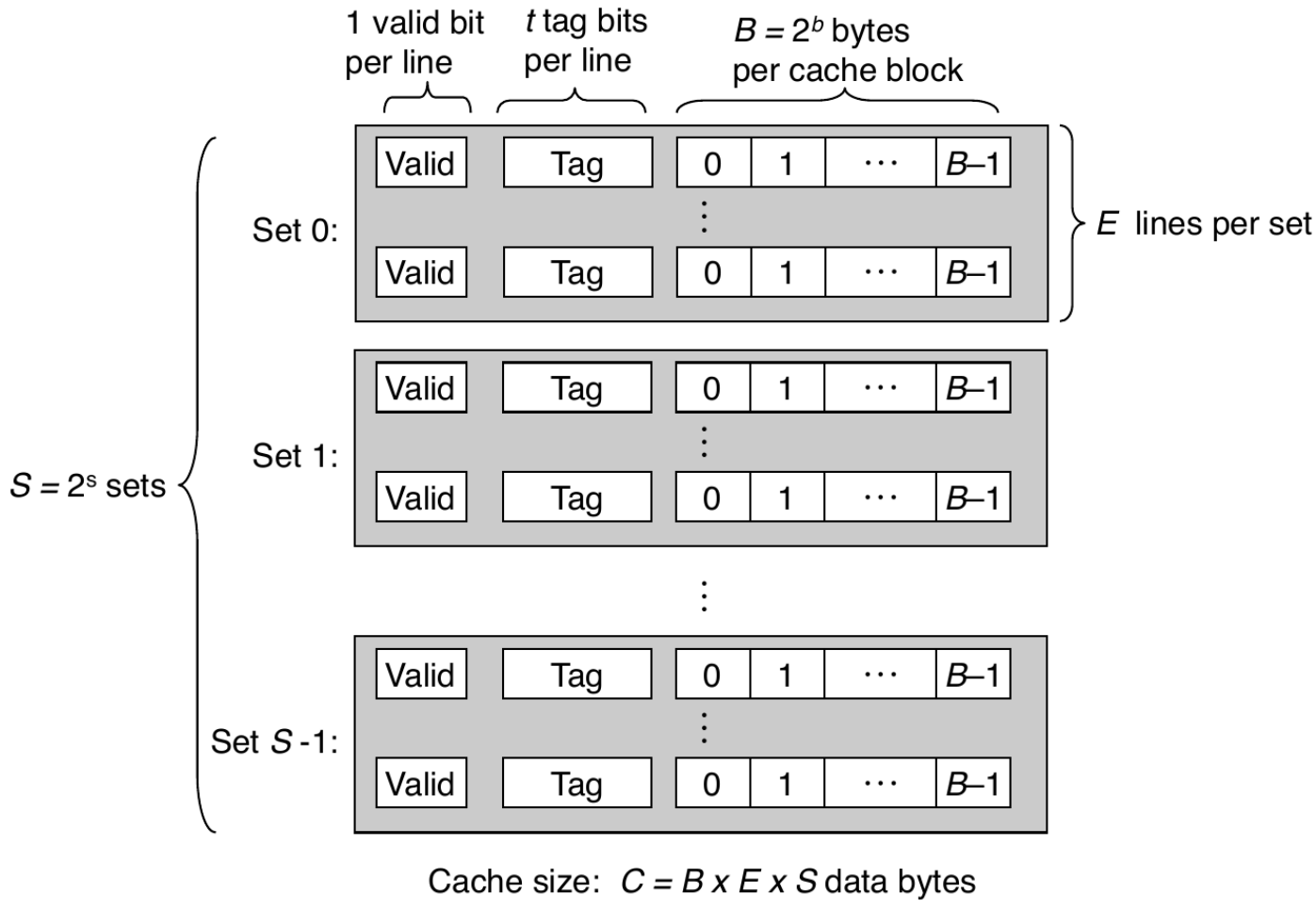
General Cache Concepts



General Cache Concepts

- A cache hit is when the data in block b is needed and is in the cache
- A cache miss is when the data in block b is needed and is in not the cache
- Types of cache misses:
  - Cold (compulsory) miss: occur because the cache starts empty and this is the first reference to the block
  - Capacity miss: occur when the set of active cache blocks (working set) is larger than the cache
  - Conflict miss: occur when the level k cache is large enough, but multiple data objects all map to the same level k block where a block is a small subset of the block positions at level $k-1$

General Cache Organization (S, E, B)

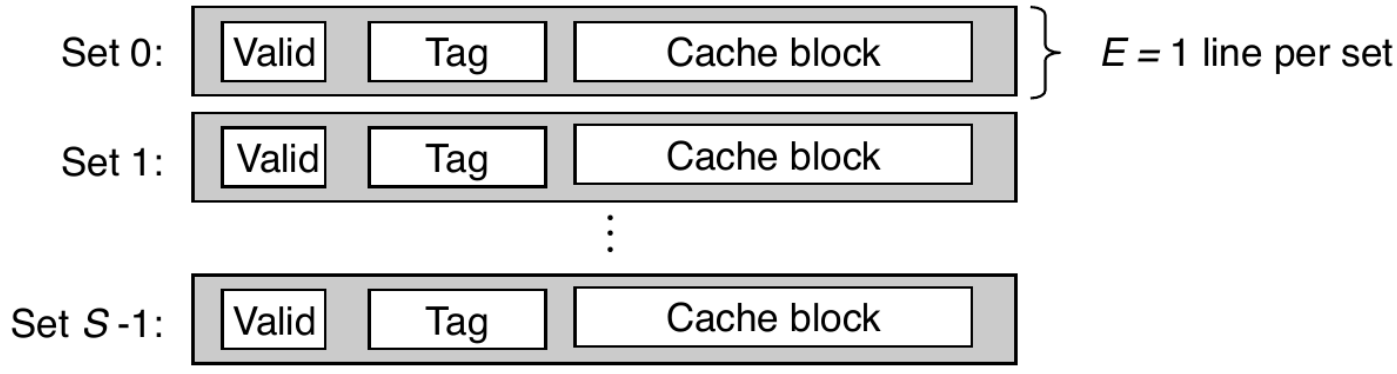


Cache Read

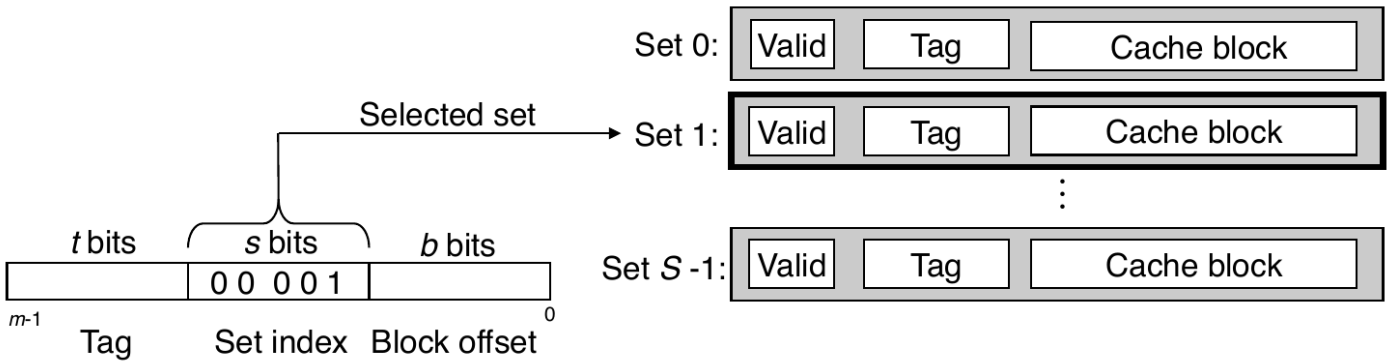
- Locate set
- Check if any line in set has matching tag
- Yes and the line is valid: hit
- Locate data starting at offset

Example: Direct-Mapped Cache

- Direct mapped: one line per set (E = 1)

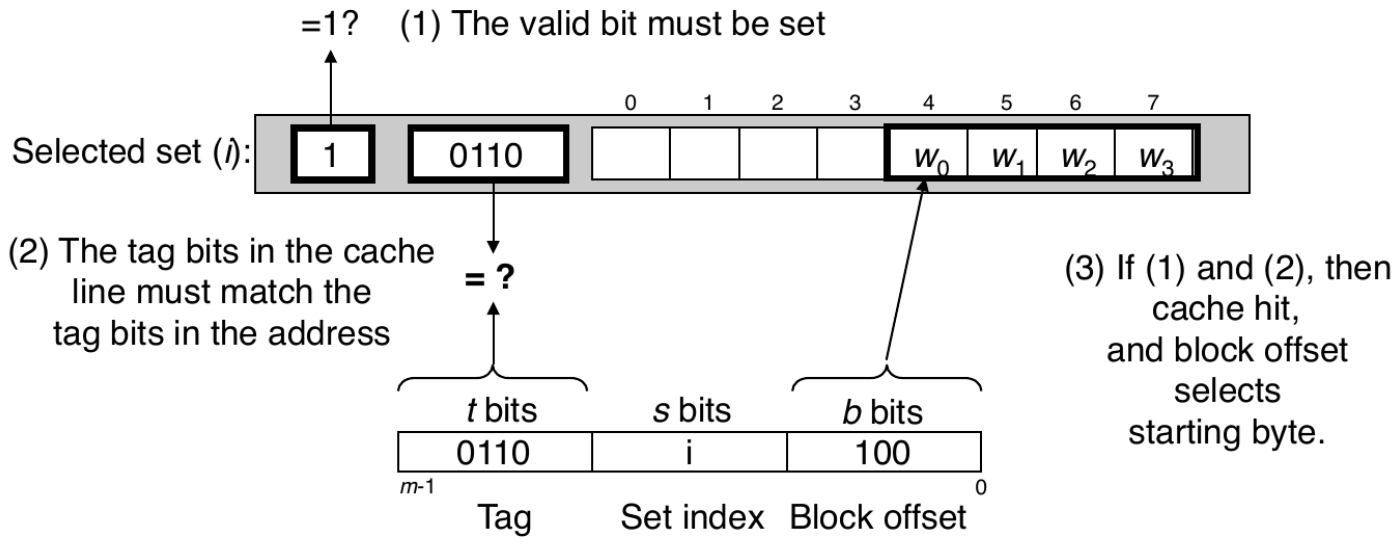


Example: Direct-Mapped Cache



- Note: the middle bits are used for indexing due to better locality

Example: Direct-Mapped Cache



- Note: if tag does not match, then old line is evicted and replaced

Direct-Mapped Cache Simulation

- Parameters: 4-bit addresses (address space size M = 16 bytes), S = 4 sets, E = 1 Block per set, B = 2 bytes per block
- Address trace (reads, one byte per read)

Address	t	s	b	Type
0	0	00	0	miss (cold)
1	0	00	1	hit
7	0	11	1	miss (cold)
8	1	00	0	miss (cold)
0	0	00	0	miss (conflict)

Direct-Mapped Cache Simulation

- Cache after trace

Set	Valid	Tag	Block
0	1	0	M[0-1]
1	0		
2	0		
3	1	0	M[6-7]

Example: E-way Set Associative Cache

- There are E lines per set
- Procedure
  - Find the set with the s-bits
  - Compare the tag for all E lines to the t-bits
  - If any of the tags match, then there is a hit
  - Otherwise, select a line for eviction and replacement from within the set
- There are many ways to select a replacement: random, least recently used (LRU), etc.

2-way Set Associative Cache Simulation

- Parameters: 4-bit addresses (address space size M = 16 bytes), S = 2 sets, E = 2 blocks per set, B = 2 bytes per block
- Address trace (reads, one byte per read)

Address	t	s	b	Type
0	00	0	0	miss
1	00	0	1	hit
7	01	1	1	miss
8	10	0	0	miss
0	00	0	0	hit

2-way Set Associative Cache Simulation

- Cache after trace

Set	Line	Valid	Tag	Block
0	1	1	00	M[0-1]
0	2	1	10	M[8-9]
1	1	1	01	M[6-7]
1	2	0		

Cache Writes

- Multiple copies of data exist:
  - L1, L2, L3, Main Memory, Disk
- What to do on a write-hit?
  - Write-through (write immediately to memory)
  - Write-back (defer write to memory until replacement of line)
    - Each cache line needs a dirty bit (set if data differs from memory)
- What to do on a write-miss?
  - Write-allocate (load into cache, update line in cache)
    - Good if more writes to the location will follow
  - No-write-allocate (writes straight to memory, does not load into cache)
- Typical combinations
  - Write-through and No-write allocate
  - Write-back and Write-allocate

Cache Performance Metrics

- Miss Rate
  - Fraction of memory accesses not found in cache (misses / access)
  - Typical numbers:
    - 3-10% for L1
    - can be quite small for L2, depending on size, etc.
- Hit Time
  - Time to deliver a cached block to the processor
    - includes time to determine whether line is in cache
  - Typical numbers:
    - 4 clock cycles for L1
    - 10 clock cycles for L2
- Miss Penalty
  - Additional time required because of a miss
    - typically 50-200 cycles for main memory (trend: increasing)

How Bad Can a Few Cache Misses Be?

- Huge difference between a hit and a miss
  - Could be 100x if just L1 and main memory
- Would you believe 99% hits is twice as good as 97%?
  - Consider this simplified example:
    - cache hit time of 1 cycle
    - cache miss penalty of 100 cycles
  - Average access time:
    - 97% hits: 1 cycle + 0.03×100 cycles = 4 cycles
    - 99% hits: 1 cycle + 0.01×100 cycles = 2 cycles
- This is why "miss rate" is used instead of "hit rate"