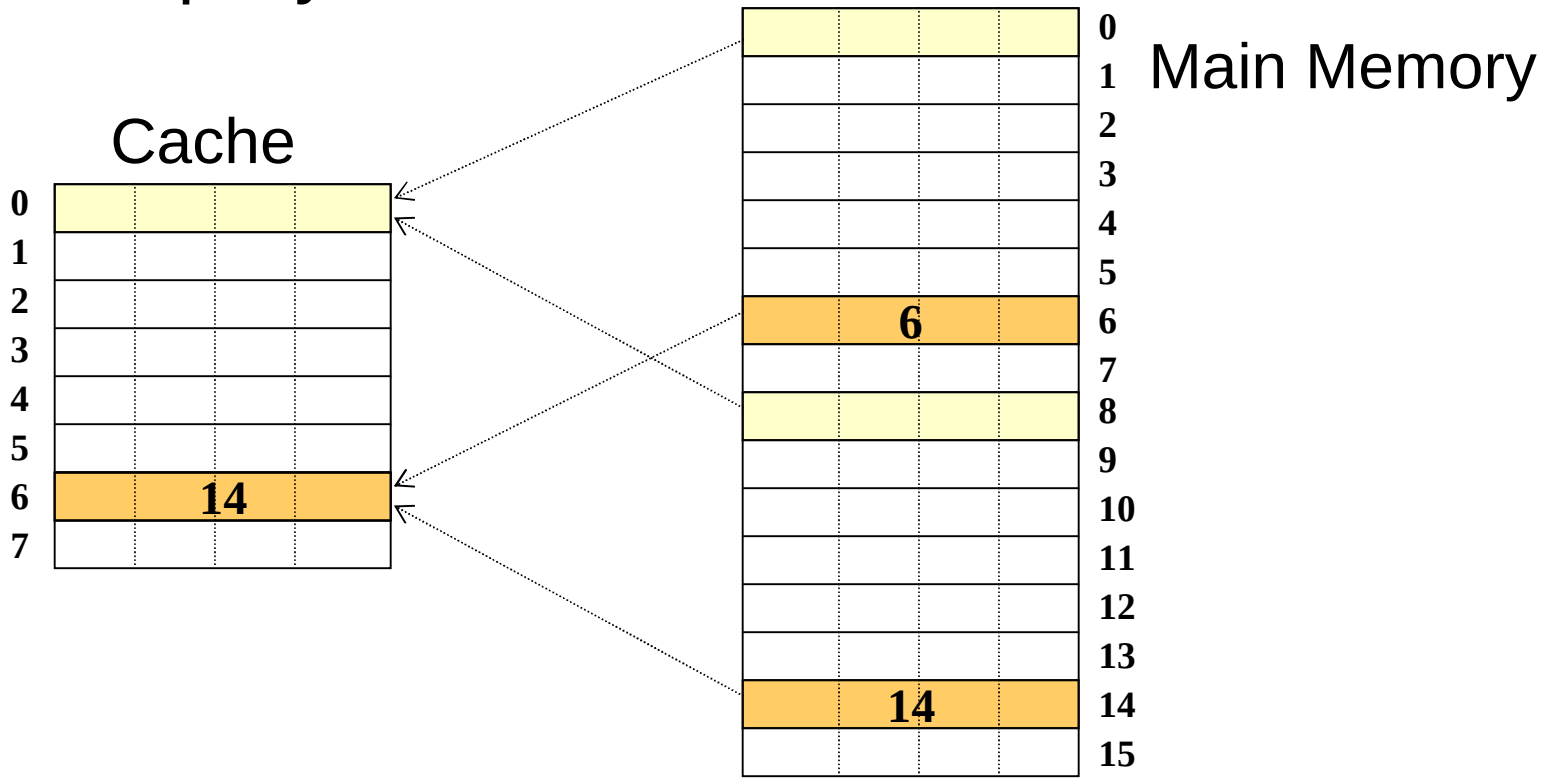
High Performance Computing
Lecture 28

Matthew Jacob

Indian Institute of Science

Block Placement: Direct Mapping (DM)

- Suppose that the cache is large enough to hold N blocks from main memory
  - i.e., Cache size = N blocks
- Direct Mapping: Memory block M is placed uniquely in cache block M mod N



Identifying Block in DM Cache

Assume 32 bit address space, 16 KB cache, 32byte cache block size.

Number of cache blocks = 16KB / 32B = 512

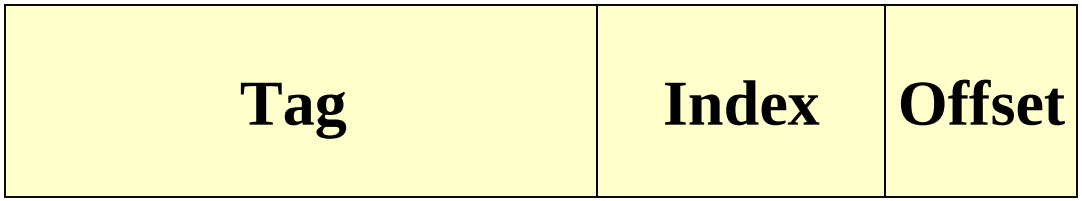
Index field: to identify the unique cache block

$\log_2 512 = 9$ bits

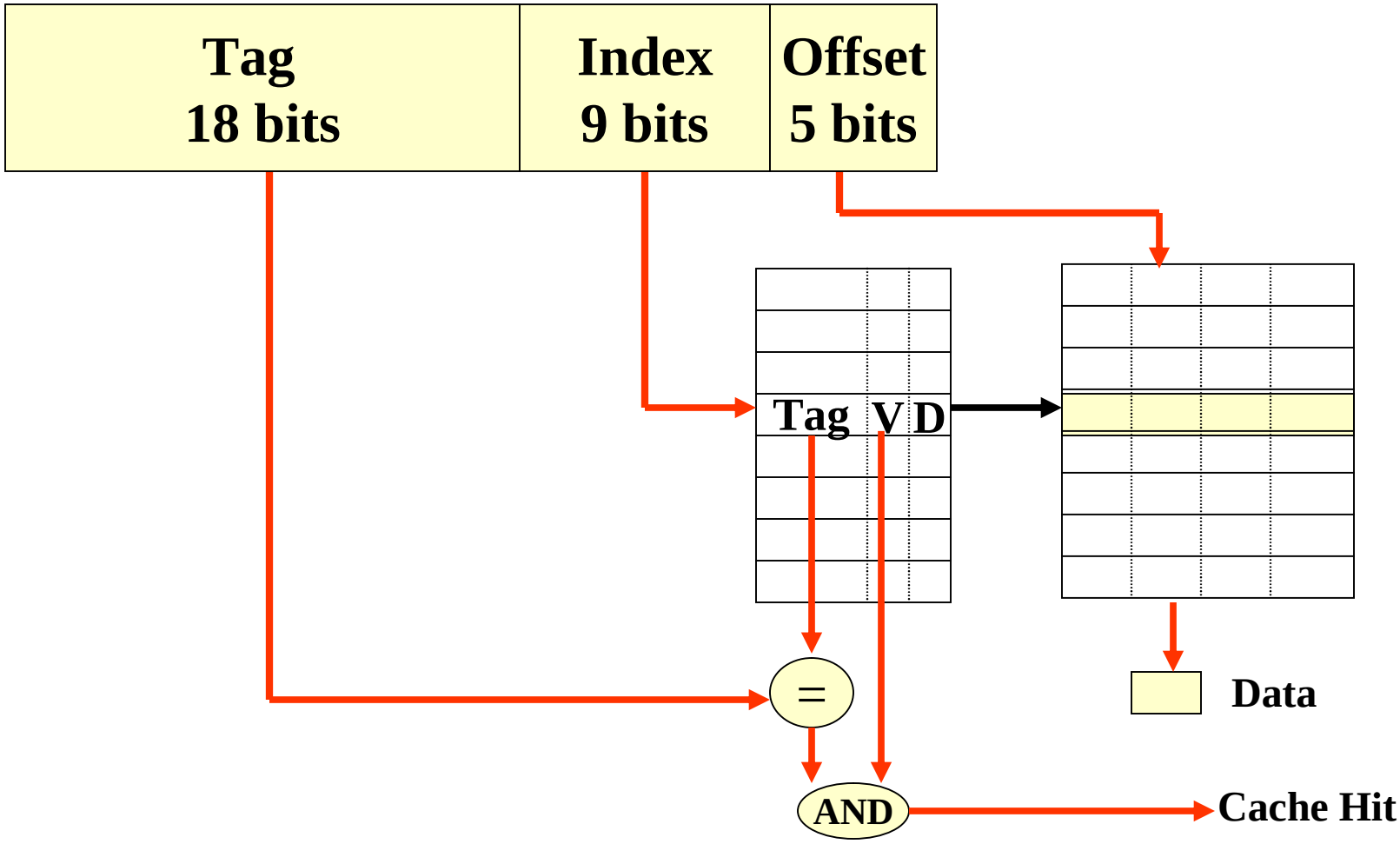
Offset field: to identify the desired byte in cache block

$\log_2 32 = 5$ bits

Tag field: to identify which memory block is currently in this cache block (remaining 18 bits)



Accessing Block in DM Cache



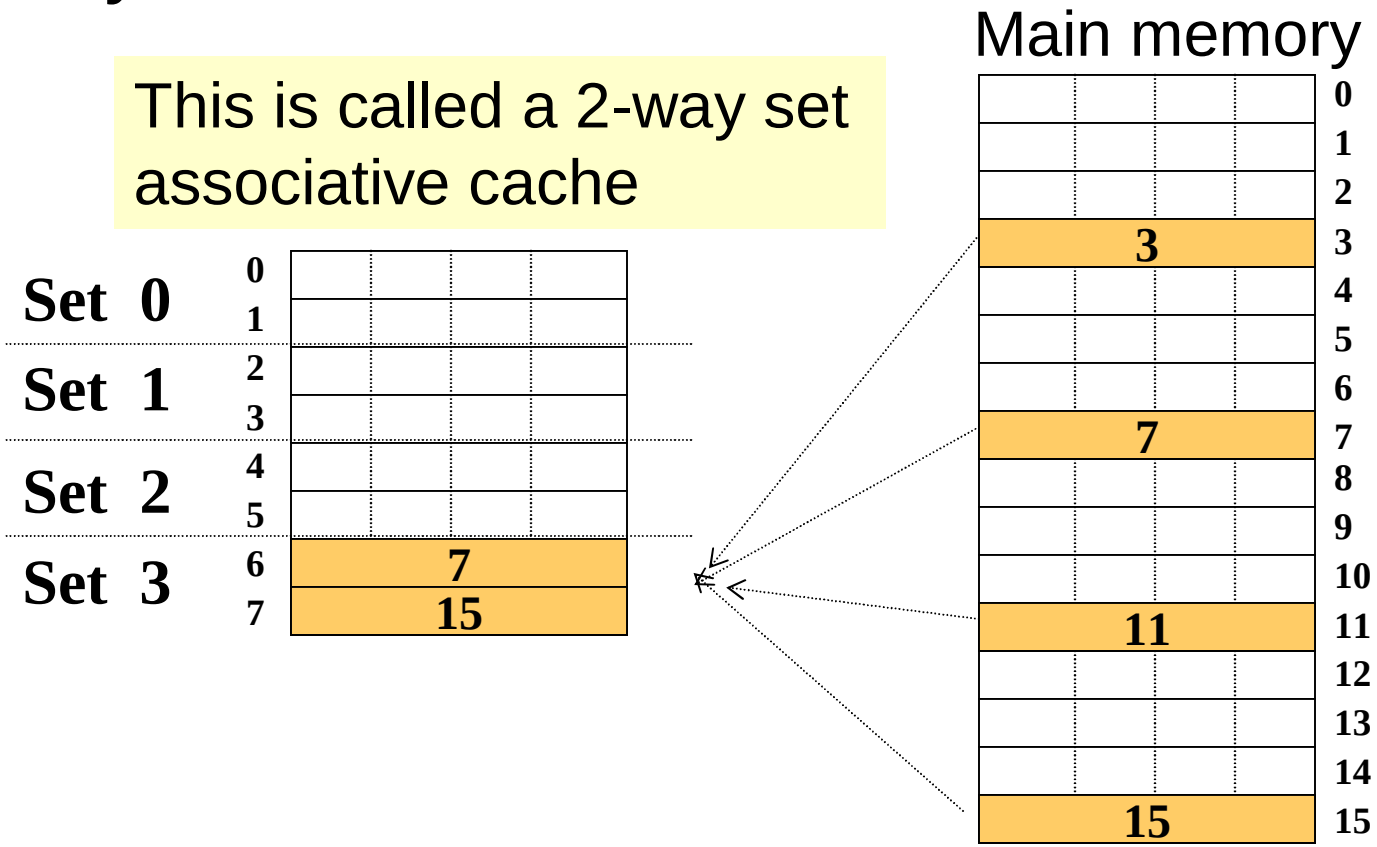
Problem with Direct Mapping

- Main memory block M is uniquely mapped to cache block M mod N
- Consider a program which accesses two memory locations alternately
  - A, B, A, B, A, B, A, B…
- Where both A and B map to the same cache block
  - Example: N = 8, A is in memory block 6 and B is in memory block 14
- Every access will result in a cache miss

Block Placement: Set Associative

- The cache is divided into S sets of blocks
- Memory block number M is placed in set M mod S; specifically, it could be placed in any block of that set

This is called a 2-way set associative cache



Identifying Block in Set Associative

Assume 32 bit address space, 16 KB cache, 32B cache block size, 4-way set associative cache

Number of Sets = Cache blocks / 4 = 512/4 = 128

Index field: to identify unique cache set

$\log_2$ (128) = 7 bits

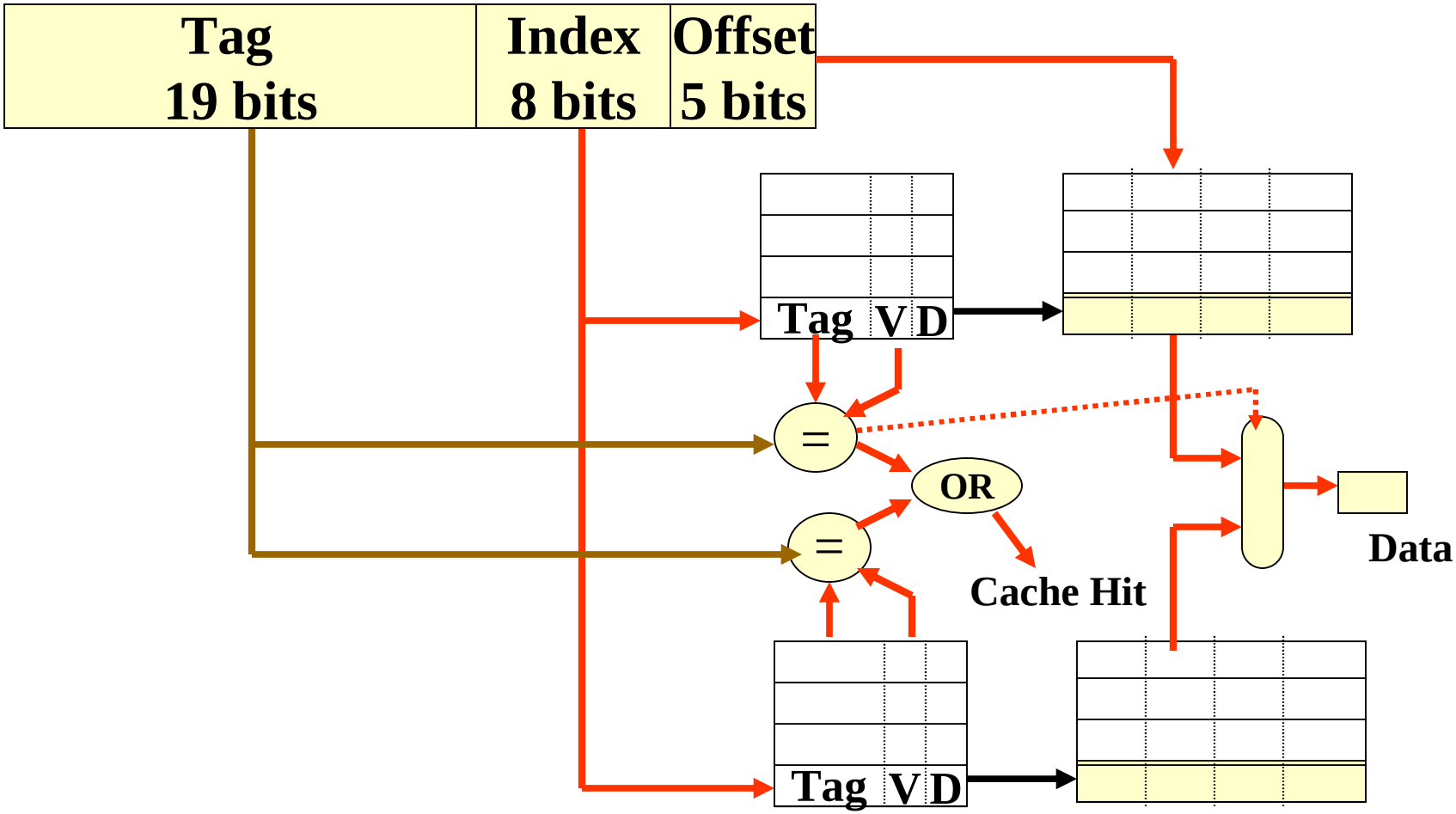
Offset field: to identify desired byte in cache block

$\log_2$ (32) = 5 bits

Tag field: to identify which memory block is currently in this cache block (remaining 20 bits)

Tag	Index	Offset

Accessing Block (2-way Set Associative)



The 4 Qs of Cache Organization

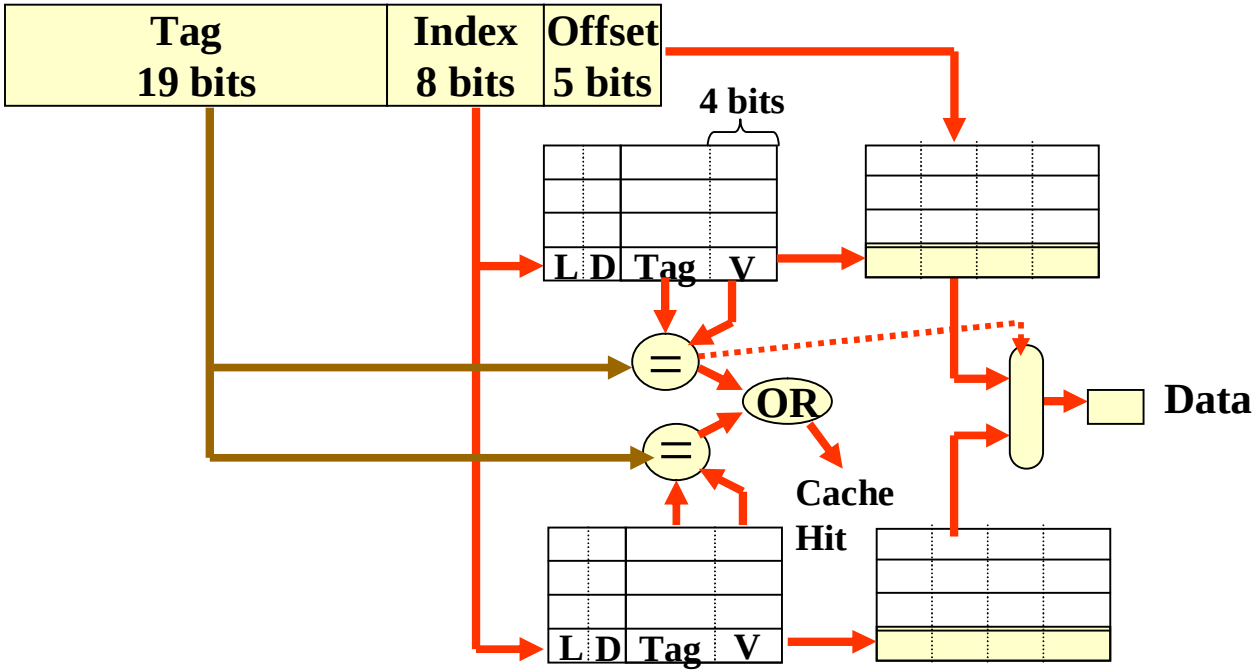
1. Where can a memory block be placed in the cache? (Block Placement)
   - Direct mapped, Set Associative
2. How is a block identified in the cache?
   - Tag, valid bit, tag checking hardware
3. What is the replacement policy used?
   - LRU, FIFO, Random …
4. What happens on writes to the cache?
   - Cache Hit: When is main memory updated?
   - Cache Miss: What happens on a write miss?

Block Replacement

- Under direct mapped placement
  - No choice is possible: unique placement
- Under set associative placement
  - One of the blocks within the unique set must be selected for replacement
  - First-In-First-Out (FIFO)
  - Least Recently Used (LRU)
  - Random

LRU Block Replacement…

- Hardware must keep track of LRU information
  - Within the cache directory
- Recall that LRU replacement was not considered feasible for Virtual Memory



The 4 Qs of Cache Organization

1. Where can a memory block be placed in the cache? (Block Placement)
   - Direct mapped, Set Associative
2. How is a block identified in the cache?
   - Tag, valid bit, tag checking hardware
3. What is the replacement policy used?
   - LRU, FIFO, Random …
4. What happens on writes to the cache?
   A. Cache Hit: When is main memory updated?
   B. Cache Miss: What happens on a write miss?

Write Policies

4A: When is Main Memory Updated on Write Hit?

- Write through: Writes are performed both in cache and in Main Memory
  - \+ Cache and memory copies are kept consistent
  - -- Multiple writes to the same location/block cause higher memory traffic
  - -- Writes must wait for longer time (memory write)
  - Solution: Use a Write Buffer to hold these write requests and allow processor to proceed immediately

Write Policies…

- Write back: writes are performed only on cache
  - Modified blocks are written back to memory only on replacement from the cache
    - o Need for dirty bit for each cache block
    - \+ Writes will happen faster than with write through
    - \+ Reduced traffic to memory
    - -- Cache & main memory copies are not always the same

Write Policies…

4B: What happens on a Write Miss?

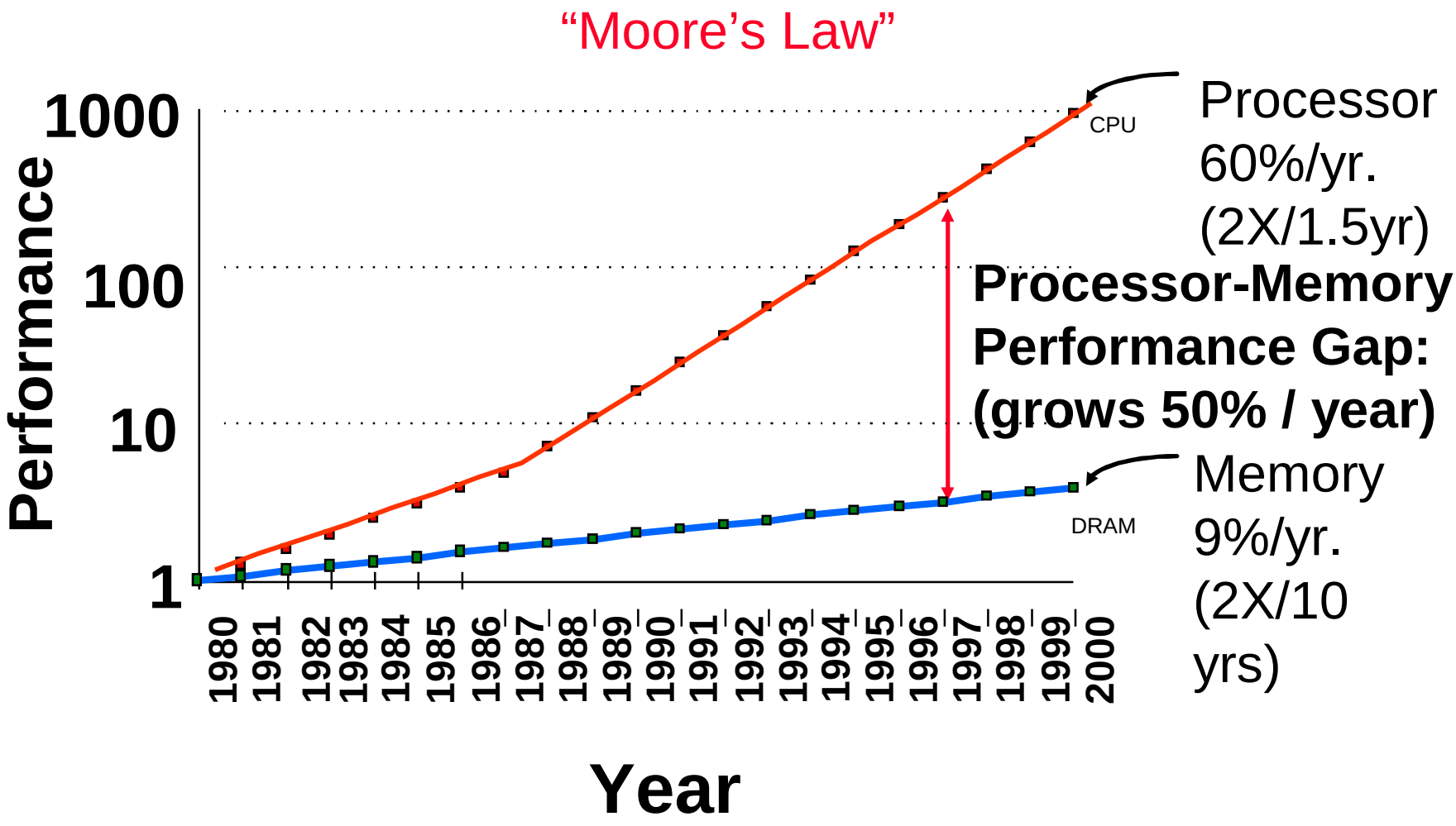
- Write-Allocate: allocate a block in the cache and load the block from memory to cache
- Write-No-Allocate: write directly to main memory

- Write allocate/no-allocate is orthogonal to write-through/write-back policy
  - Write-allocate with write-back
  - Write-no-allocate with write-through: ideal if mostly-reads-few-writes on data

Putting it all together

- The computer you are using to run your programs contains cache memory
- Cache organization and operation are described by the terms we have seen
- Example: 32 KB 2-way set associative write-back write-allocate LRU replacement cache with block size of 32B

What Drives Computer Development?



Memory Hierarchy Progression