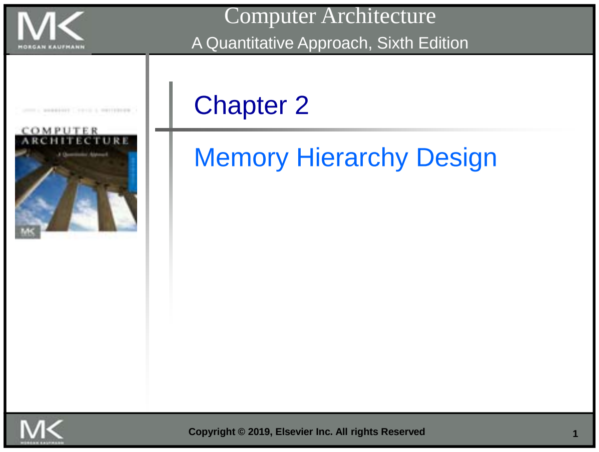
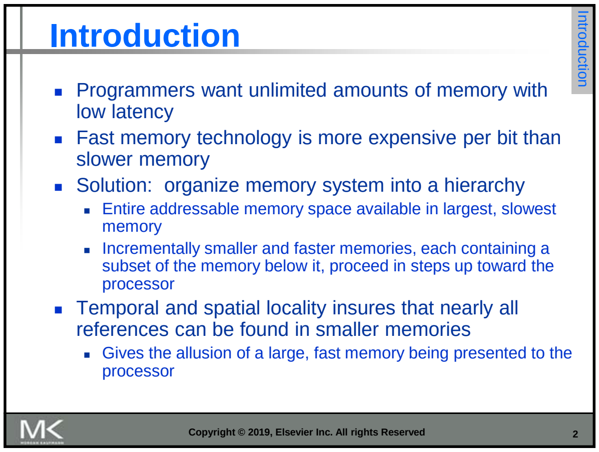


Computer Architecture

A Quantitative Approach, Sixth Edition

Chapter 2

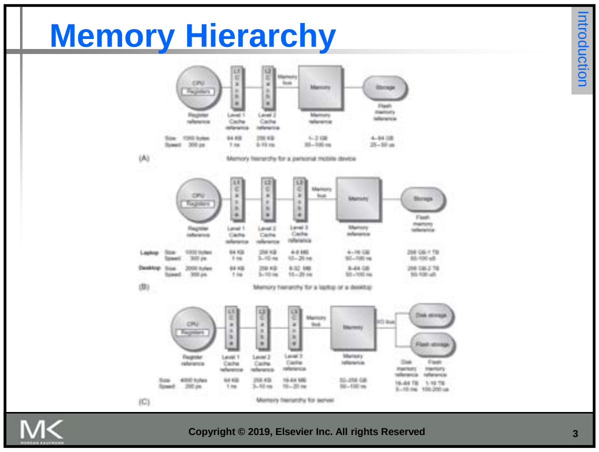
Memory Hierarchy Design



Introduction

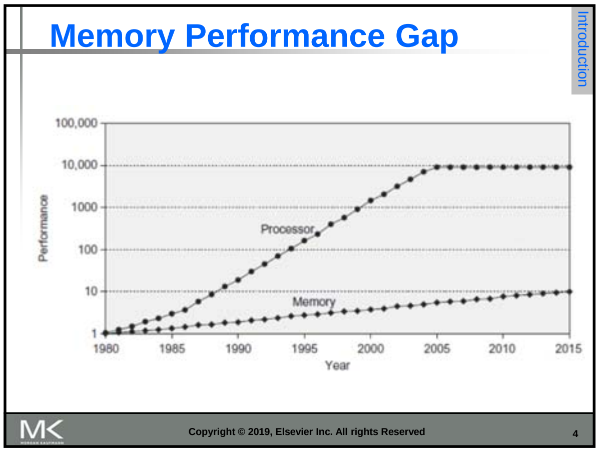
Introduction

- Programmers want unlimited amounts of memory with low latency
- Fast memory technology is more expensive per bit than slower memory
- Solution: organize memory system into a hierarchy
  - Entire addressable memory space available in largest, slowest memory
  - Incrementally smaller and faster memories, each containing a subset of the memory below it, proceed in steps up toward the processor
- Temporal and spatial locality insures that nearly all references can be found in smaller memories
  - Gives the allusion of a large, fast memory being presented to the processor

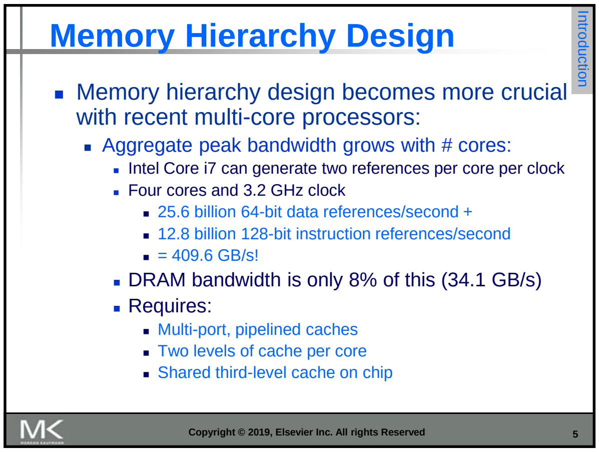


Memory Hierarchy

Introduction



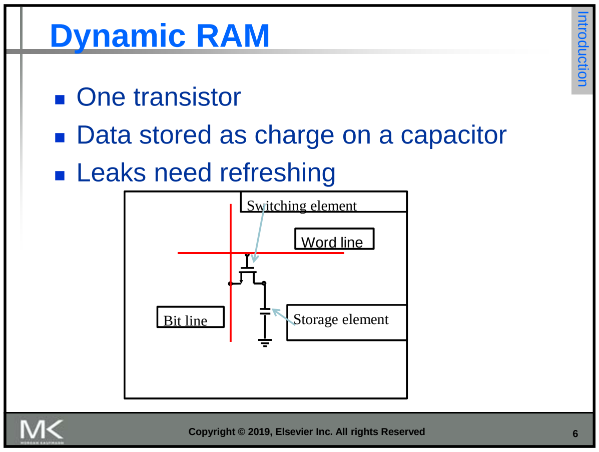
Memory Performance Gap



Memory Hierarchy Design

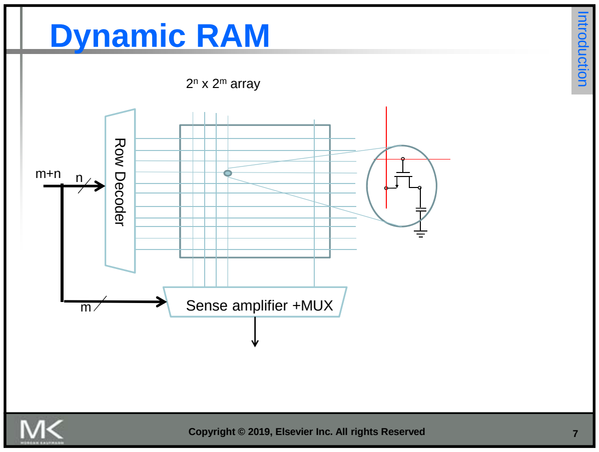
Introduction

- Memory hierarchy design becomes more crucial with recent multi-core processors:
  - Aggregate peak bandwidth grows with # cores:
    - Intel Core i7 can generate two references per core per clock
    - Four cores and 3.2 GHz clock
      - 25.6 billion 64-bit data references/second +
      - 12.8 billion 128-bit instruction references/second
      - = 409.6 GB/s!
    - DRAM bandwidth is only 8% of this (34.1 GB/s)
    - Requires:
      - Multi-port, pipelined caches
      - Two levels of cache per core
      - Shared third-level cache on chip

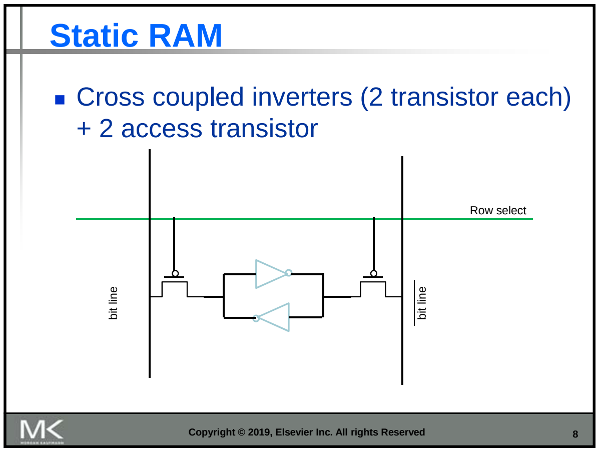


Dynamic RAM

- One transistor
- Data stored as charge on a capacitor
- Leaks need refreshing

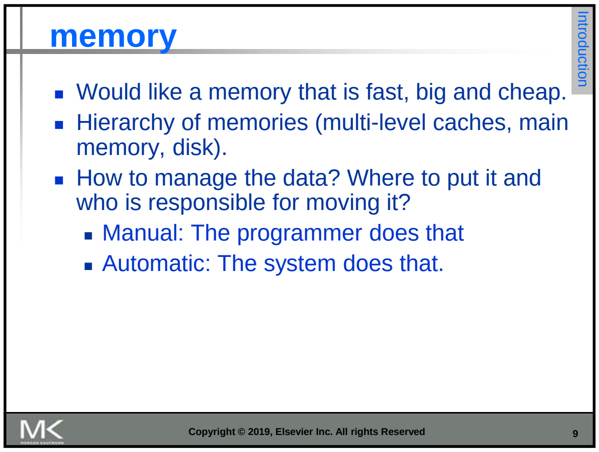


Dynamic RAM



Static RAM

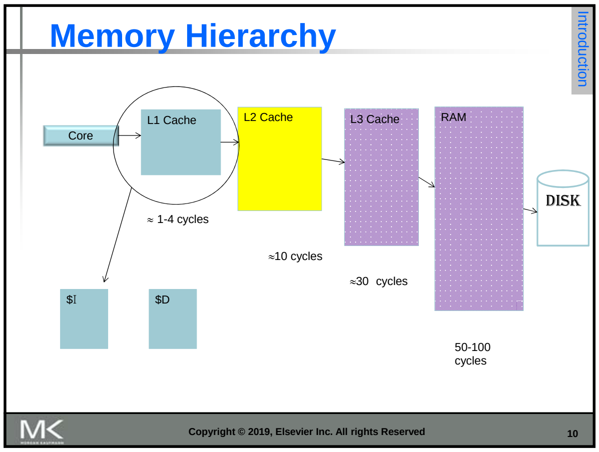
- Cross coupled inverters (2 transistor each) + 2 access transistor



memory

Introduction

- Would like a memory that is fast, big and cheap.
- Hierarchy of memories (multi-level caches, main memory, disk).
- How to manage the data? Where to put it and who is responsible for moving it?
  - Manual: The programmer does that
  - Automatic: The system does that.



Memory Hierarchy

Introduction

Core

L1 Cache

L2 Cache

L3 Cache

RAM

DISK

≈ 1-4 cycles

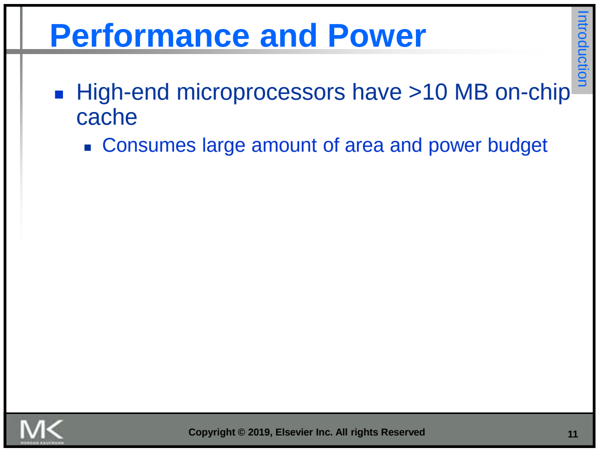
≈10 cycles

≈30 cycles

50-100 cycles

$I

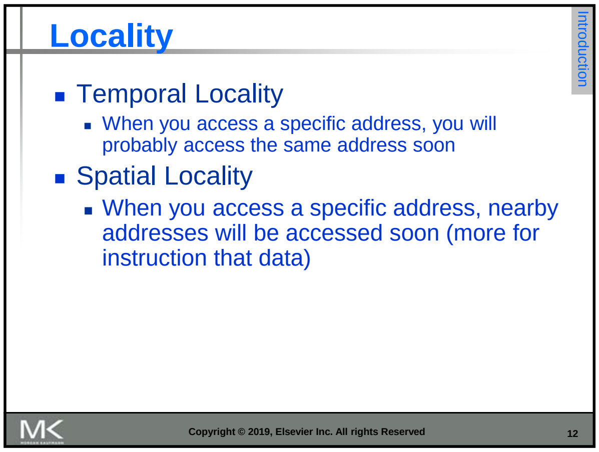
$D



Performance and Power

Introduction

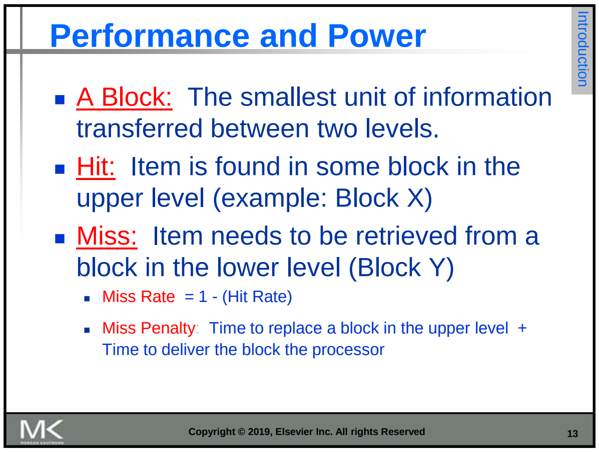
- High-end microprocessors have >10 MB on-chip cache
  - Consumes large amount of area and power budget



Locality

Introduction

- Temporal Locality
  - When you access a specific address, you will probably access the same address soon
- Spatial Locality
  - When you access a specific address, nearby addresses will be accessed soon (more for instruction that data)

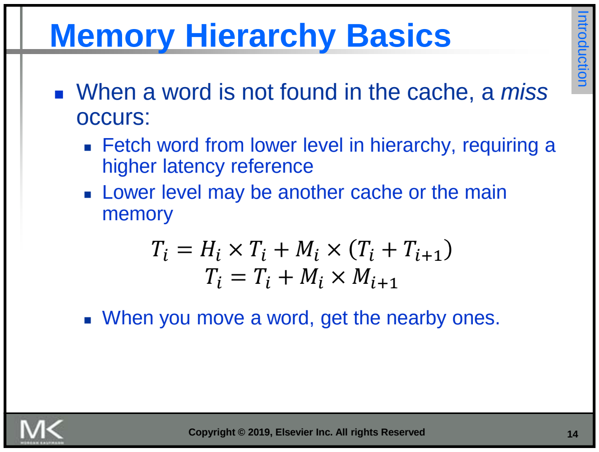


Performance and Power

Introduction

- A Block: The smallest unit of information transferred between two levels.
- Hit: Item is found in some block in the upper level (example: Block X)
- Miss: Item needs to be retrieved from a block in the lower level (Block Y)
  - Miss Rate = 1 - (Hit Rate)
  - Miss Penalty: Time to replace a block in the upper level + Time to deliver the block the processor

13



Memory Hierarchy Basics

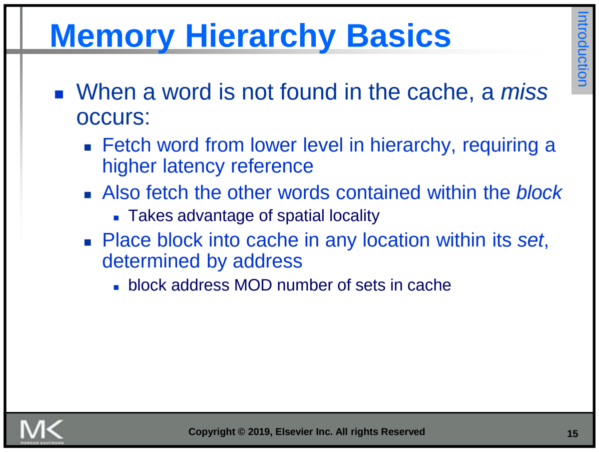
Introduction

- When a word is not found in the cache, a *miss* occurs:
  - Fetch word from lower level in hierarchy, requiring a higher latency reference
  - Lower level may be another cache or the main memory

$$T_i = H_i \times T_i + M_i \times (T_i + T_{i+1})$$
$$T_i = T_i + M_i \times M_{i+1}$$

  - When you move a word, get the nearby ones.

14

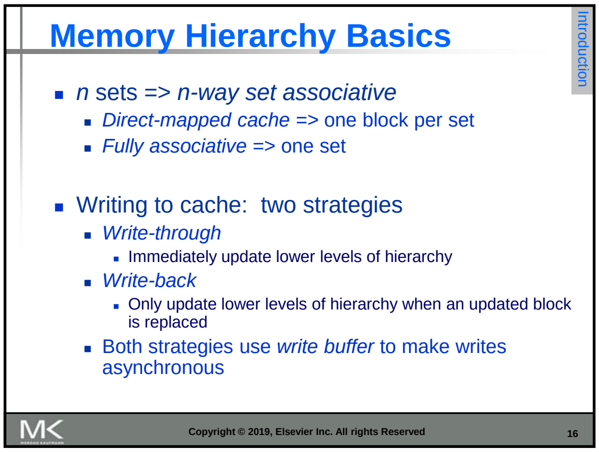


Memory Hierarchy Basics

Introduction

- When a word is not found in the cache, a *miss* occurs:
  - Fetch word from lower level in hierarchy, requiring a higher latency reference
  - Also fetch the other words contained within the *block*
    - Takes advantage of spatial locality
  - Place block into cache in any location within its *set*, determined by address
    - block address MOD number of sets in cache

15

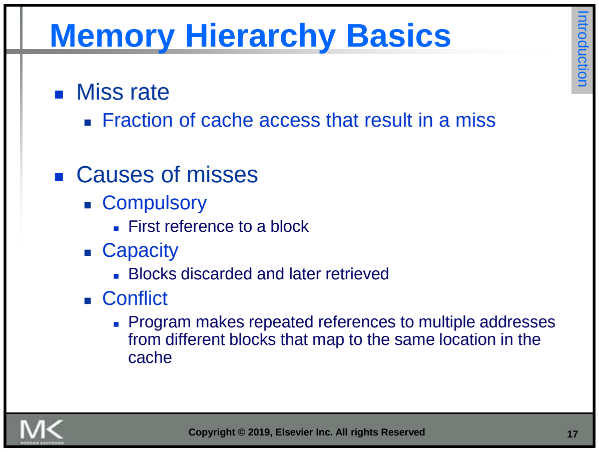


Memory Hierarchy Basics

Introduction

- *n* sets => *n-way set associative*
  - *Direct-mapped cache* => one block per set
  - *Fully associative* => one set
- Writing to cache: two strategies
  - *Write-through*
    - Immediately update lower levels of hierarchy
  - *Write-back*
    - Only update lower levels of hierarchy when an updated block is replaced
  - Both strategies use *write buffer* to make writes asynchronous

16

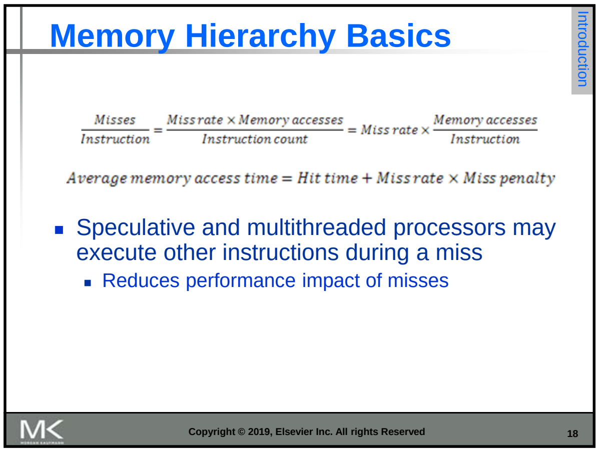


Memory Hierarchy Basics

Introduction

- Miss rate
  - Fraction of cache access that result in a miss
- Causes of misses
  - Compulsory
    - First reference to a block
  - Capacity
    - Blocks discarded and later retrieved
  - Conflict
    - Program makes repeated references to multiple addresses from different blocks that map to the same location in the cache

17



Memory Hierarchy Basics

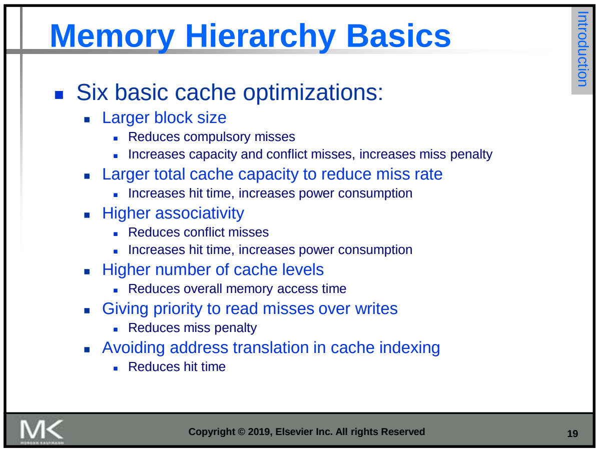
Introduction

$$\frac{Misses}{Instruction} = \frac{Miss\,rate \times Memory\,accesses}{Instruction\,count} = Miss\,rate \times \frac{Memory\,accesses}{Instruction}$$

$$Average\,memory\,access\,time = Hit\,time + Miss\,rate \times Miss\,penalty$$

- Speculative and multithreaded processors may execute other instructions during a miss
  - Reduces performance impact of misses

18

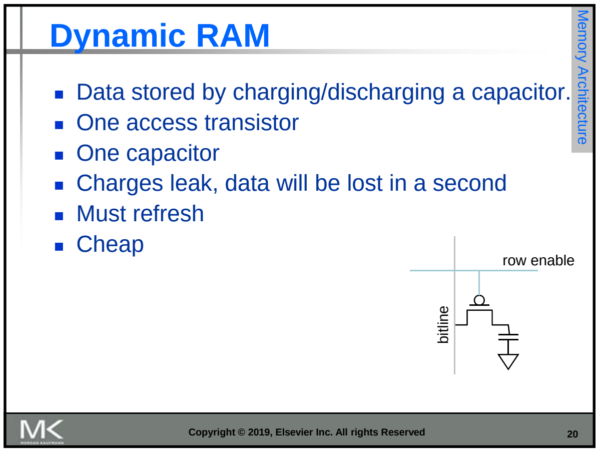


Memory Hierarchy Basics

Introduction

- Six basic cache optimizations:
  - Larger block size
    - Reduces compulsory misses
    - Increases capacity and conflict misses, increases miss penalty
  - Larger total cache capacity to reduce miss rate
    - Increases hit time, increases power consumption
  - Higher associativity
    - Reduces conflict misses
    - Increases hit time, increases power consumption
  - Higher number of cache levels
    - Reduces overall memory access time
  - Giving priority to read misses over writes
    - Reduces miss penalty
  - Avoiding address translation in cache indexing
    - Reduces hit time

19



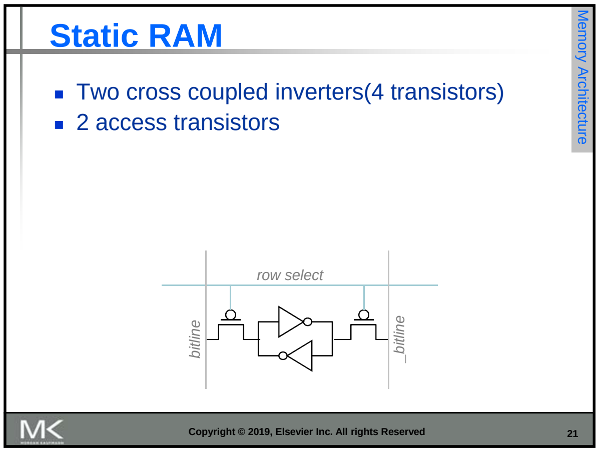
Dynamic RAM

Memory Architecture

- Data stored by charging/discharging a capacitor.
- One access transistor
- One capacitor
- Charges leak, data will be lost in a second
- Must refresh
- Cheap

row enable
bitline

20



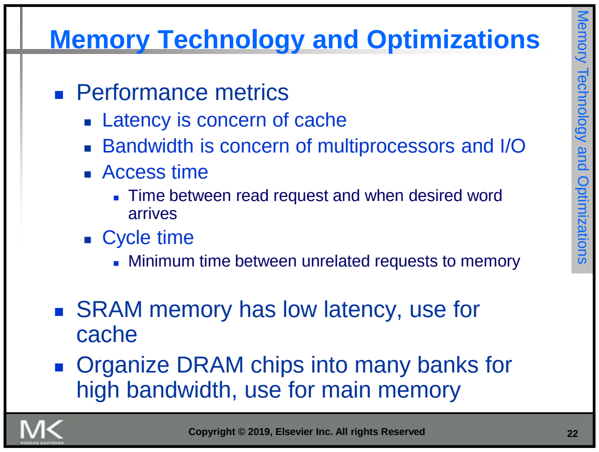
Static RAM

Memory Architecture

- Two cross coupled inverters(4 transistors)
- 2 access transistors

row select
bitline
_bitline

21

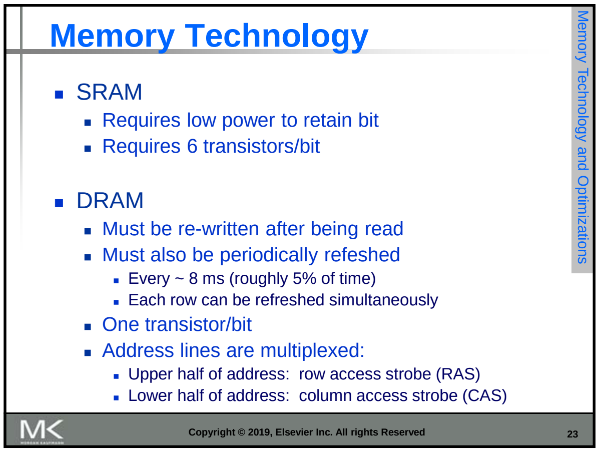


Memory Technology and Optimizations

Memory Technology and Optimizations

- Performance metrics
  - Latency is concern of cache
  - Bandwidth is concern of multiprocessors and I/O
  - Access time
    - Time between read request and when desired word arrives
  - Cycle time
    - Minimum time between unrelated requests to memory
- SRAM memory has low latency, use for cache
- Organize DRAM chips into many banks for high bandwidth, use for main memory

22

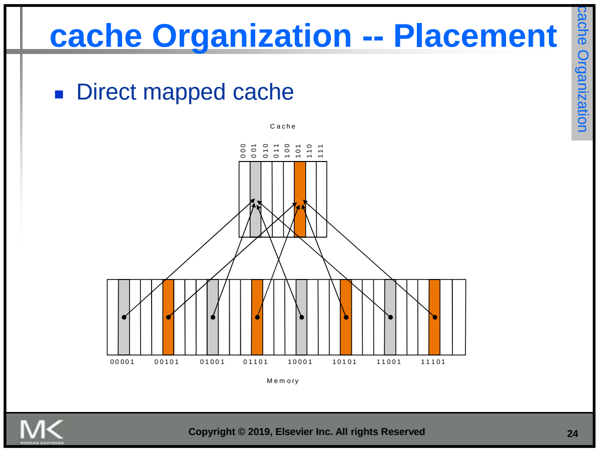


Memory Technology

Memory Technology and Optimizations

- SRAM
  - Requires low power to retain bit
  - Requires 6 transistors/bit
- DRAM
  - Must be re-written after being read
  - Must also be periodically refeshed
    - Every ~ 8 ms (roughly 5% of time)
    - Each row can be refreshed simultaneously
  - One transistor/bit
  - Address lines are multiplexed:
    - Upper half of address: row access strobe (RAS)
    - Lower half of address: column access strobe (CAS)

23

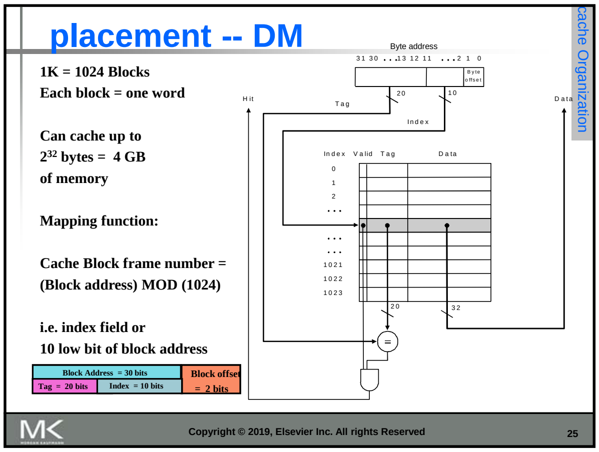


cache Organization -- Placement

cache Organization

- Direct mapped cache

24



placement -- DM

1K = 1024 Blocks
Each block = one word

Can cache up to
2^{32} bytes = 4 GB
of memory

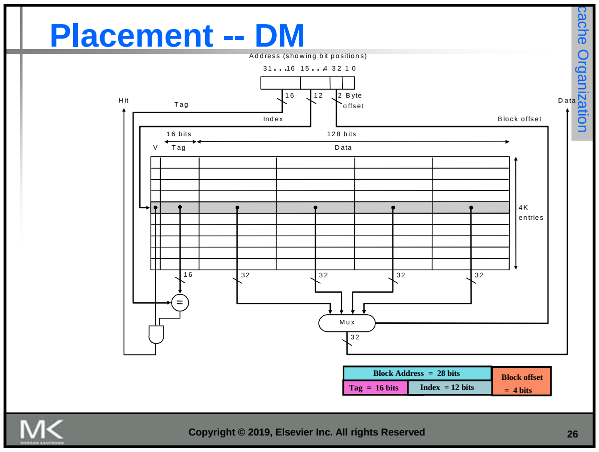
Mapping function:

Cache Block frame number =
(Block address) MOD (1024)

i.e. index field or
10 low bit of block address

Block Address = 30 bits		Block offset = 2 bits
Tag = 20 bits	Index = 10 bits	

25



Placement -- DM

Block Address = 28 bits		Block offset = 4 bits
Tag = 16 bits	Index = 12 bits	

26

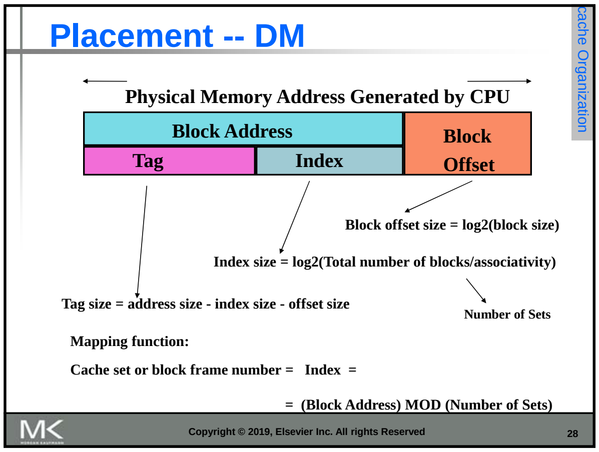
Placement -- DM

- Each block frame in cache has an address tag.
- The tags of every cache block that might contain the required data are checked in parallel.
- A valid bit is added to the tag to indicate whether this entry contains a valid address.
- The address from the CPU to cache is divided into:
  - A block address, further divided into:
    - An index field to choose a block set in cache.
    - (no index field when fully associative).
    - A tag field to search and match addresses in the selected set.
  - A block offset to select the data from the block.

Block Address		Block Offset
Tag	Index	

cache Organization

27



Placement -- DM

cache Organization

Physical Memory Address Generated by CPU

Block offset size = log2(block size)

Index size = log2(Total number of blocks/associativity)

Number of Sets

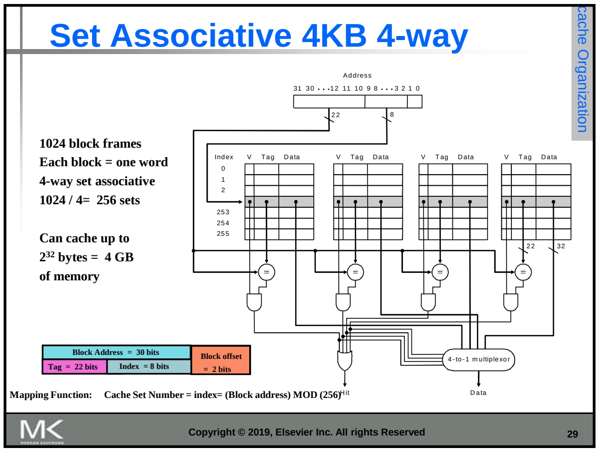
Tag size = address size - index size - offset size

Mapping function:

Cache set or block frame number = Index =

= (Block Address) MOD (Number of Sets)

28



Set Associative 4KB 4-way

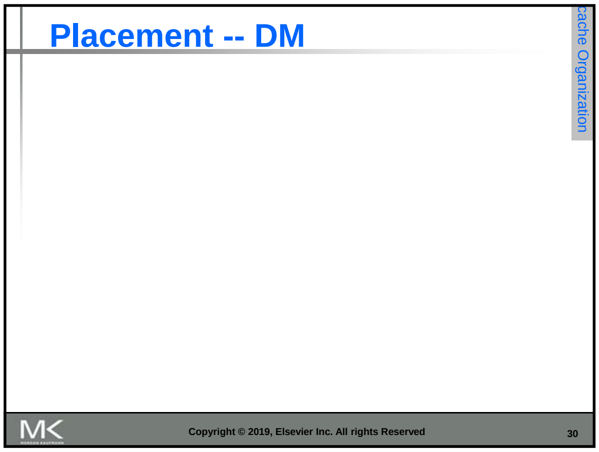
cache Organization

1024 block frames
Each block = one word
4-way set associative
1024 / 4= 256 sets

Can cache up to
2^{32} bytes = 4 GB
of memory

Mapping Function: Cache Set Number = index= (Block address) MOD (256)

29



Placement -- DM

cache Organization

30