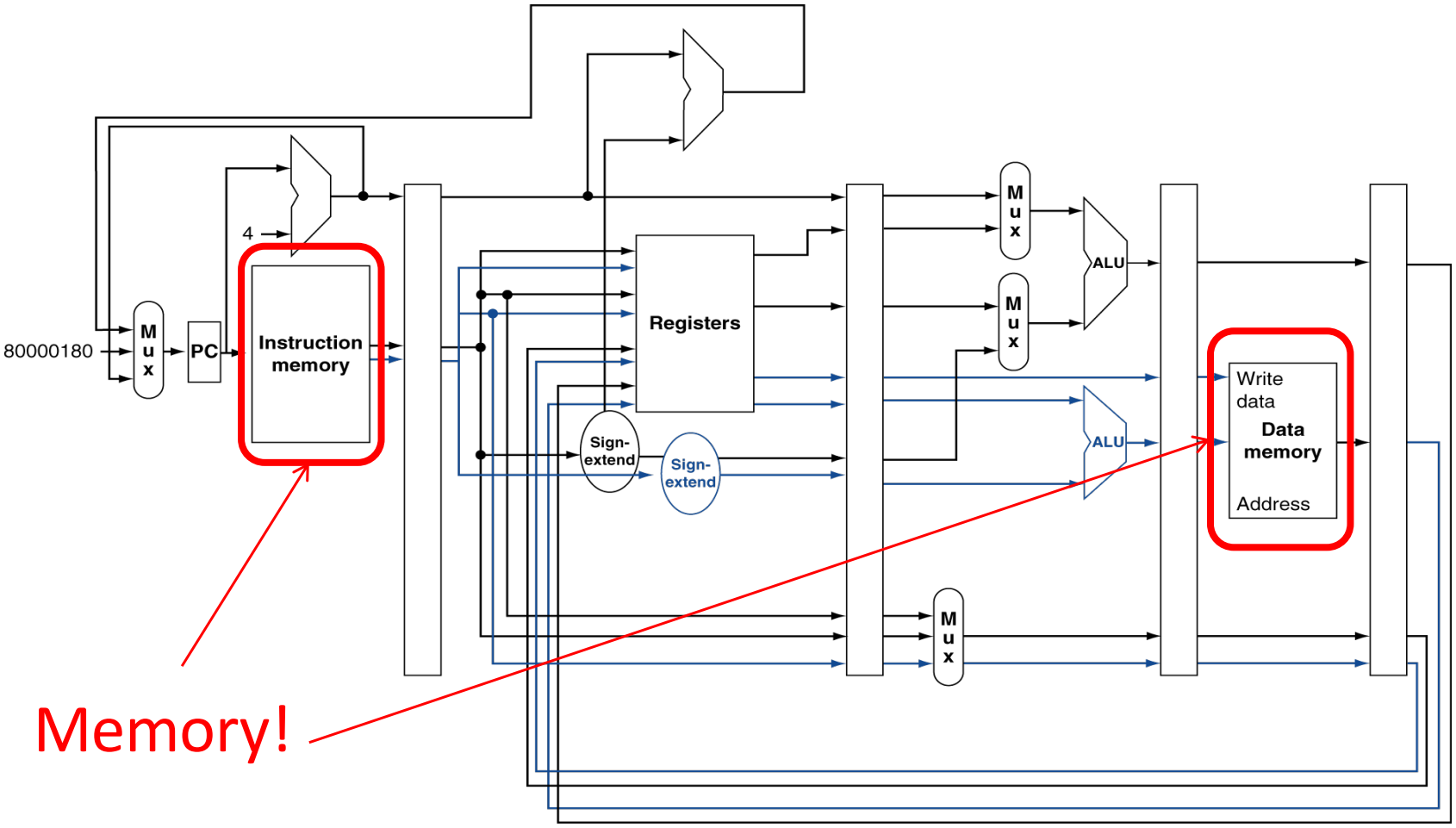
Cache and Memory Hierarchy (1)

Lecture 08
November 1st, 2023

Jae W. Lee (jaewlee@snu.ac.kr)
Computer Science and Engineering
Seoul National University

***Slide credits**: [COD:RV2e] slides from Elsevier Inc.*

The Main Topic of Chapter 5 – Memory!



Where are these memories physically located?

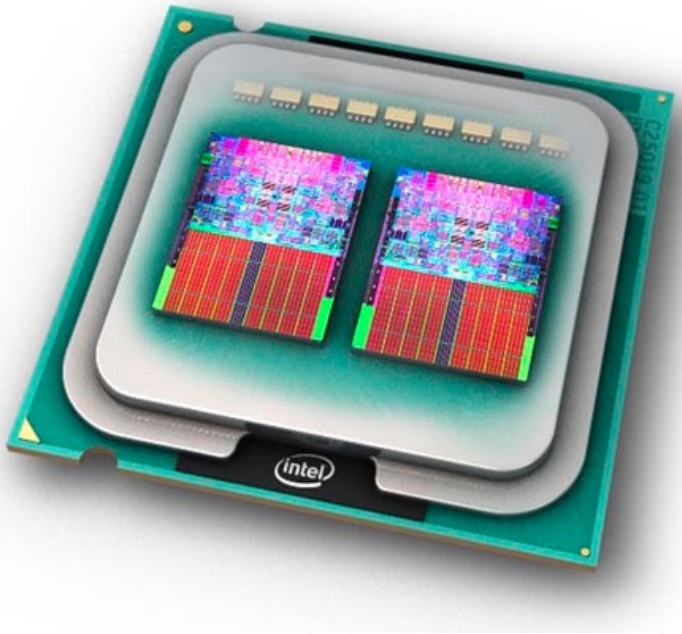
Outline

Textbook: COD 5.1 – 5.3

- **Introduction**
- **Memory Technologies**
- **The Basics of Caches**

Main Idea – Memory Hierarchy with Multiple Levels!

- How to make cheap and fast memory?



On-Chip (CPU) Caches:
Small, expensive, but fast

Off-Chip DRAM:
Large, cheap, but slow

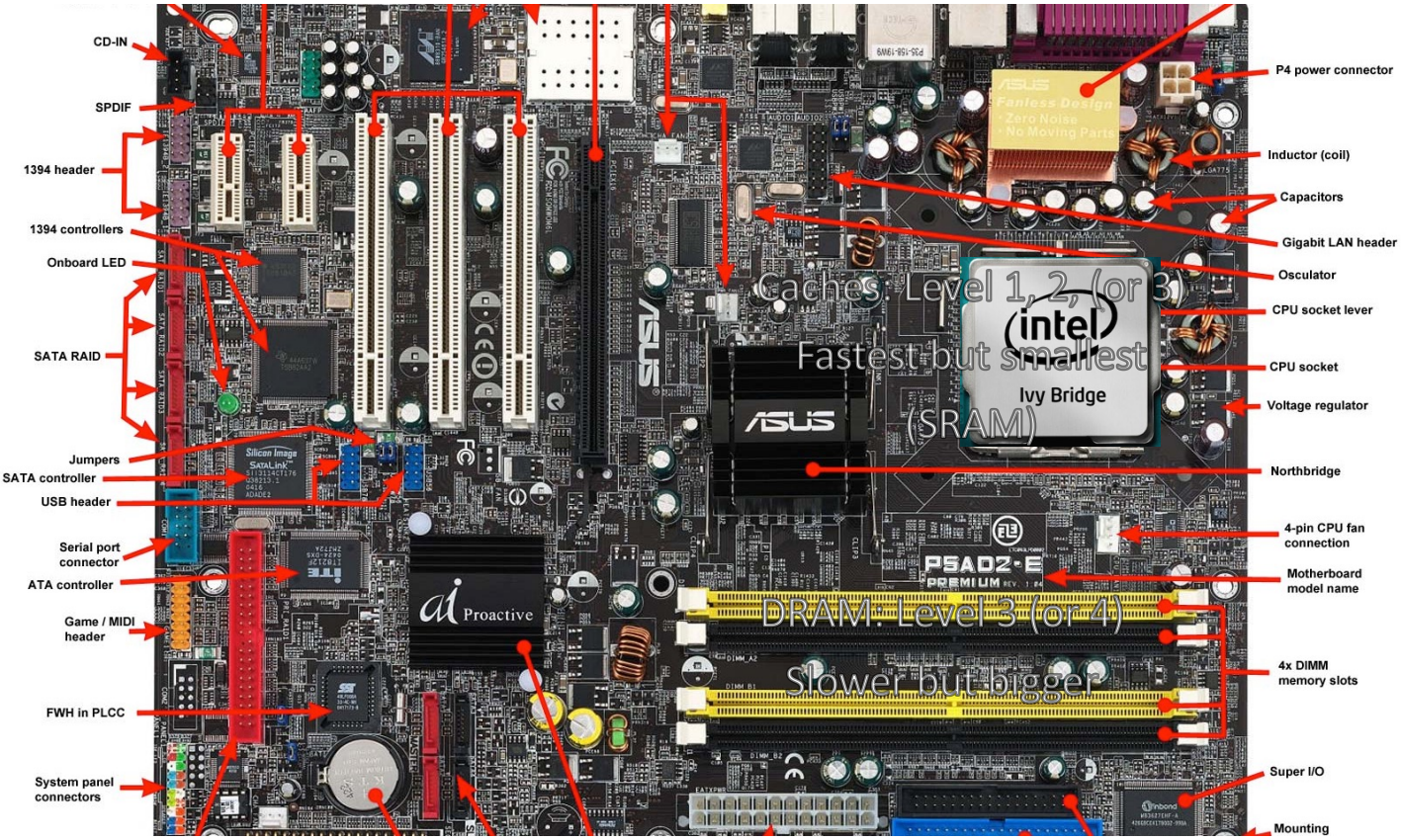
Exploiting Hierarchy Example 1: Book Storage

How about Level 3?

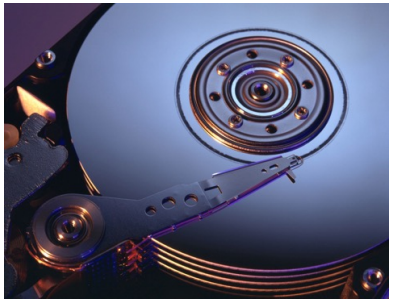
Exploiting Hierarchy Example 2: Cookers

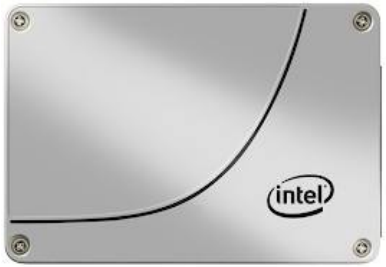
- **The concept of hierarchy is everywhere. ☺**

How about Computer System?



How about Level 4 (or 5)?





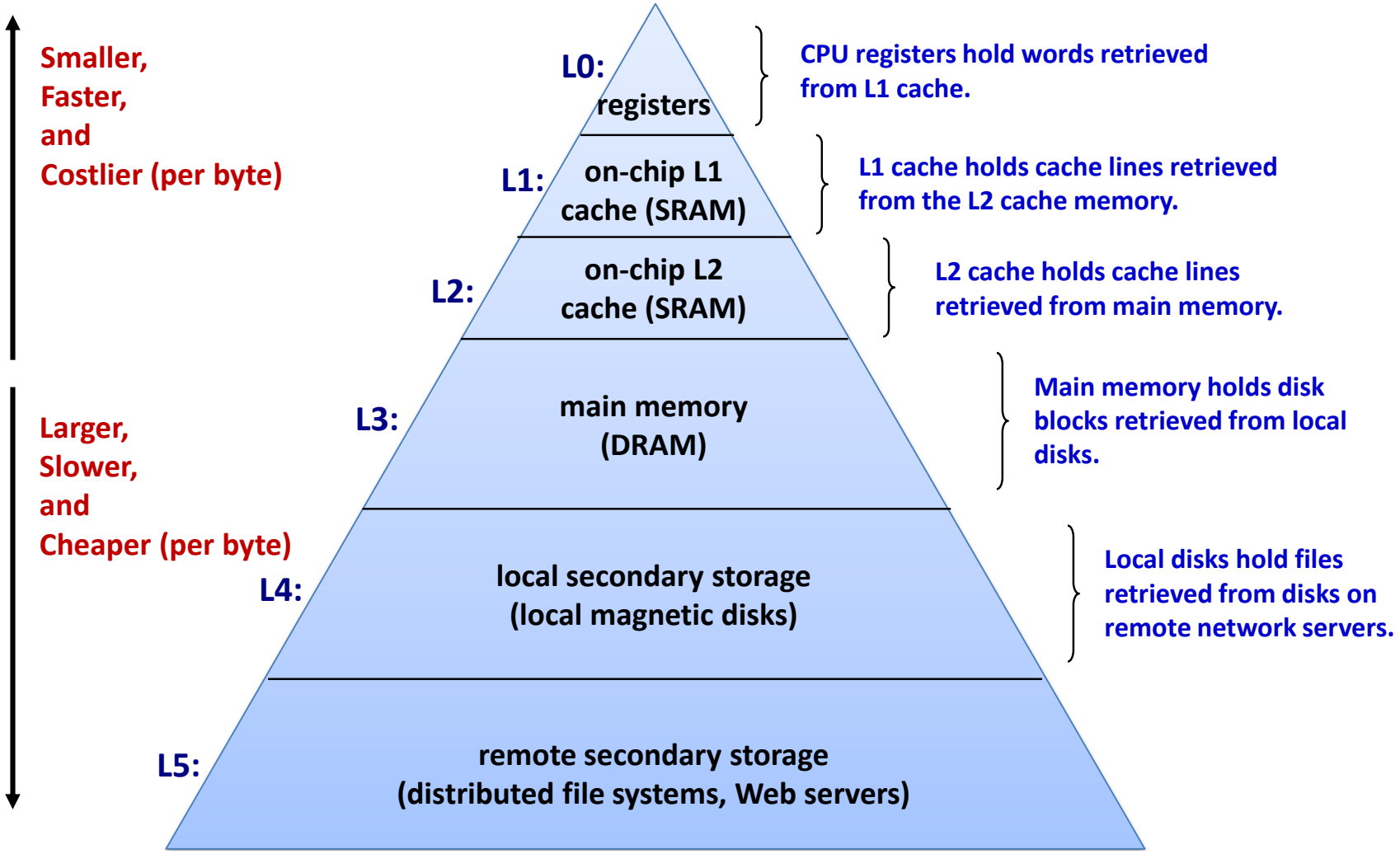
Principle of Locality

- **Programs access a small proportion of their address space at any time**
- **Temporal locality**
  - Items accessed recently are likely to be accessed again soon
  - e.g., instructions in a loop, induction variables
- **Spatial locality**
  - Items near those accessed recently are likely to be accessed soon
  - E.g., sequential instruction access, array data

Taking Advantage of Locality

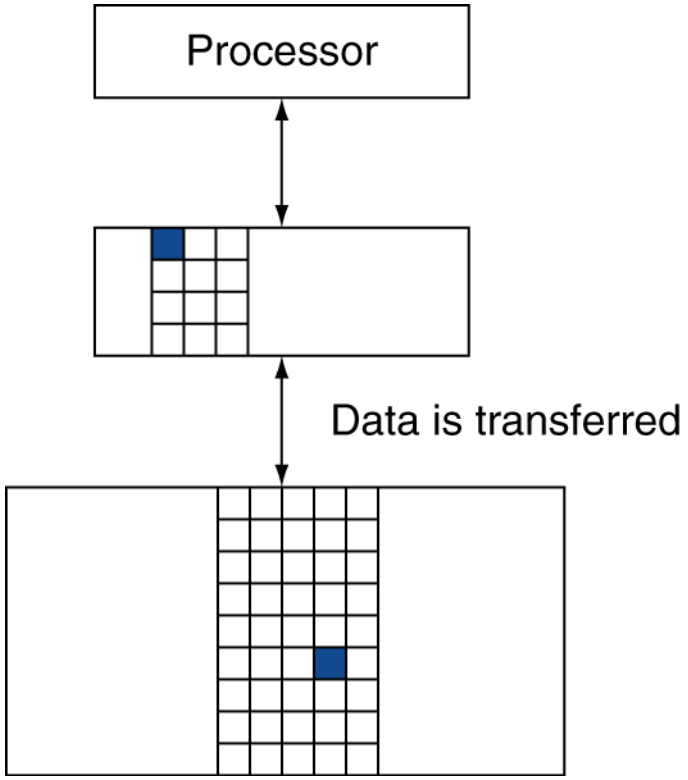
- **Memory hierarchy**
- **Store everything on disk**
- **Copy recently accessed (and nearby) items from disk to smaller DRAM memory**
  - Main memory
- **Copy more recently accessed (and nearby) items from DRAM to smaller SRAM memory**
  - Cache memory attached to CPU

Memory Hierarchy



[Source: ICE 3003 Lecture Notes by Prof. Jin-Soo Kim @ SKKU]

Memory Hierarchy Levels



- **Block (aka line): unit of copying**
  - May be multiple words
- **If accessed data is present in upper level**
  - Hit: access satisfied by upper level
    - Hit ratio: hits/accesses
- **If accessed data is absent**
  - Miss: block copied from lower level
    - Time taken: miss penalty
    - Miss ratio: misses/accesses
      = 1 – hit ratio
  - Then accessed data supplied from upper level

Outline

Textbook: COD 5.1 – 5.3

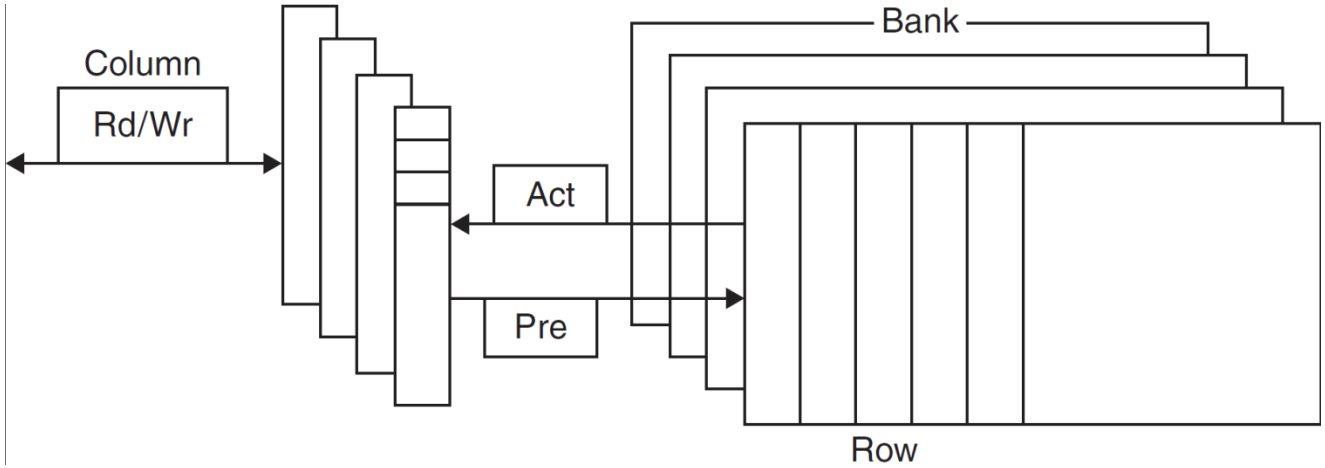
- Introduction
- **Memory Technologies**
- The Basics of Caches

Memory Technology

- **Static RAM (SRAM)**
  - 0.5ns – 2.5ns, $500 – $1000 per GB
- **Dynamic RAM (DRAM)**
  - 50ns – 70ns, $3 – $6 per GB
- **Magnetic disk**
  - 5ms – 20ms, $0.01 – $0.02 per GB
- **Ideal memory**
  - Access time of SRAM
  - Capacity and cost/GB of disk

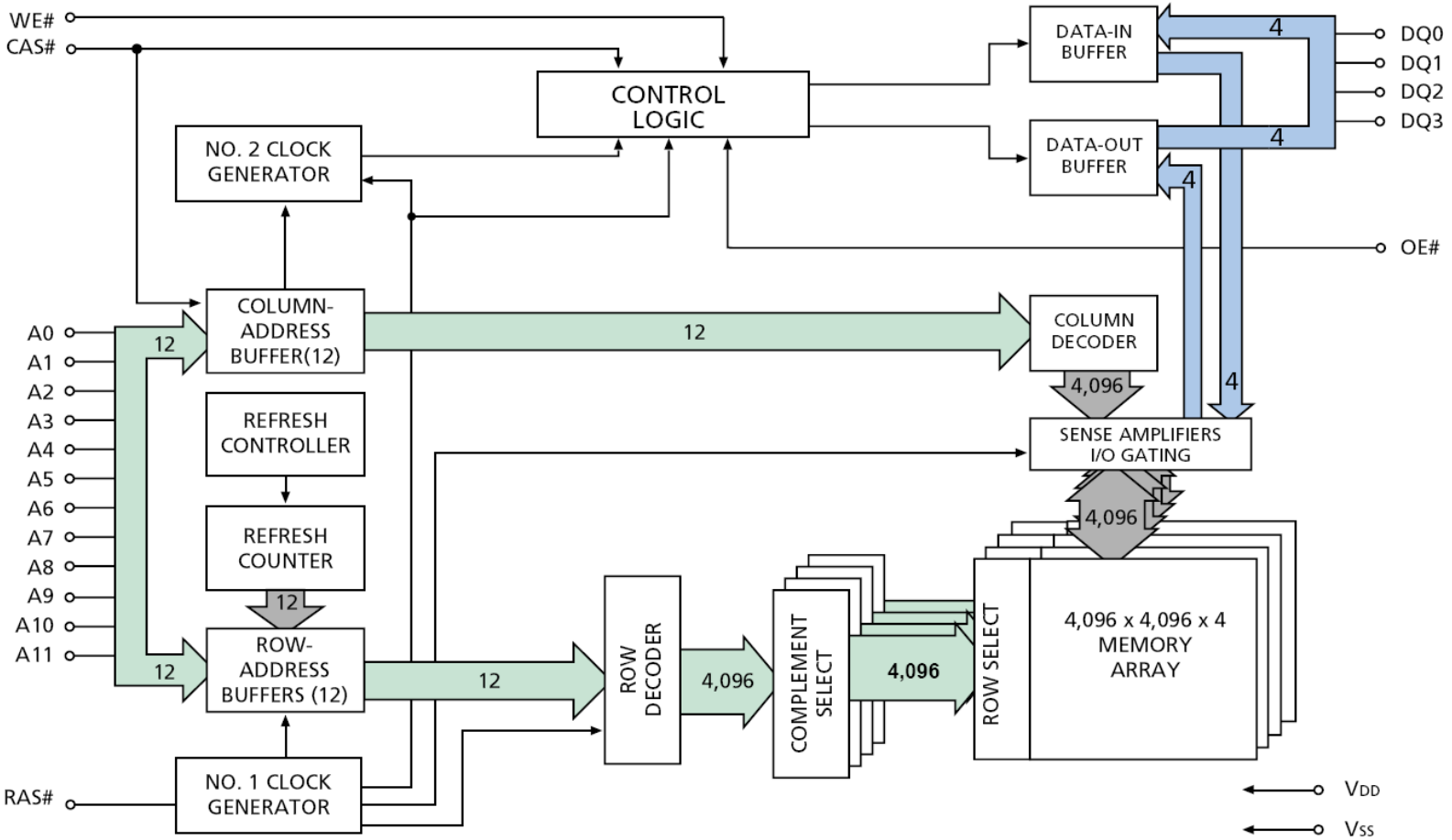
DRAM Technology

- **Data stored as a charge in a capacitor**
  - Single transistor used to access the charge
  - Must periodically be refreshed
    - Read contents and write back
    - Performed on a DRAM “row”



DRAM Organization

- **Micron MT4LC16M4T8 (16M x 4bit)**



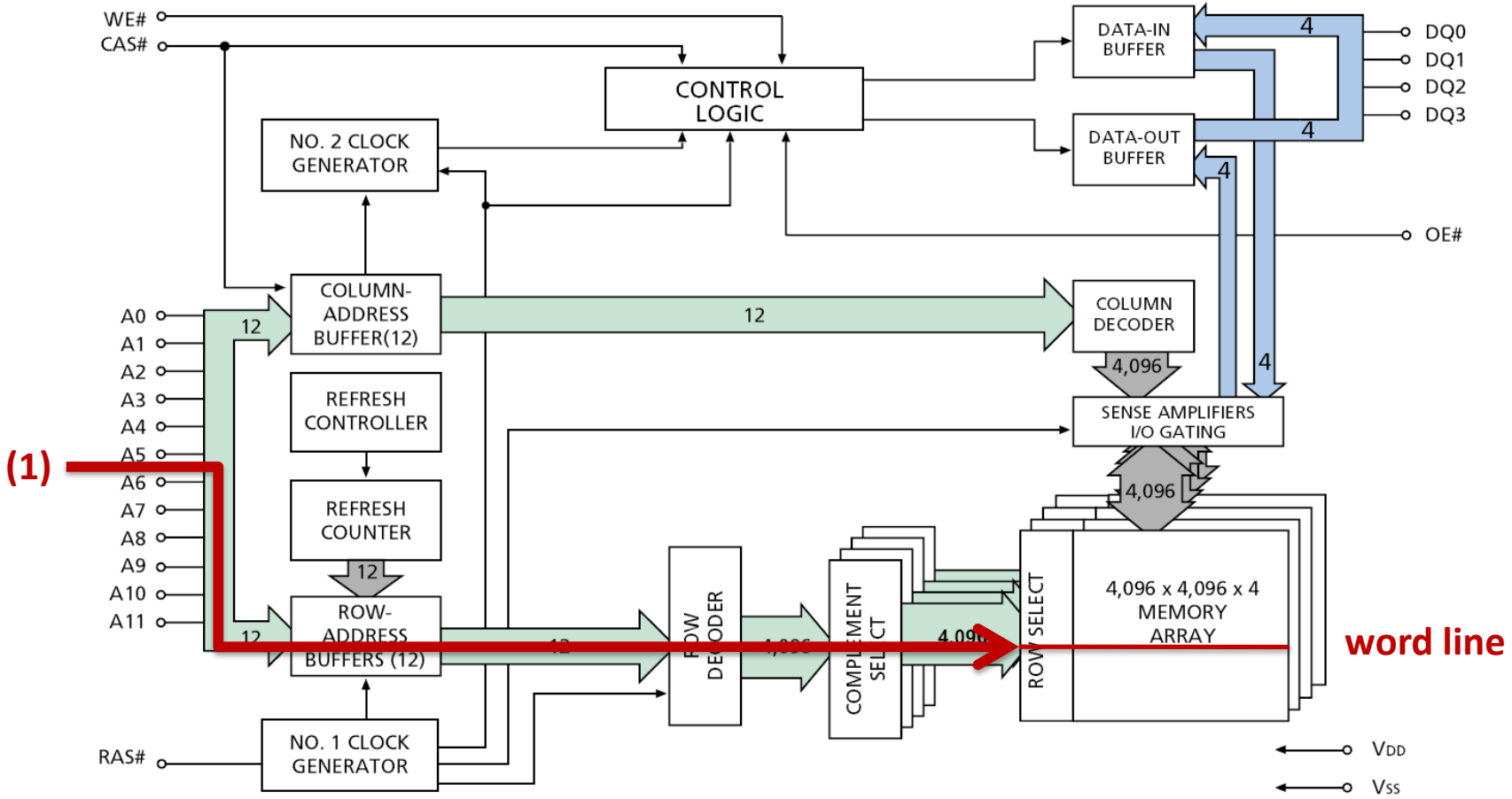
[Source: ICE 3003 Lecture Notes by Prof. Jin-Soo Kim @ SKKU]

DRAM Organization

- **DRAM configuration**
  - Large capacity: 1 – 4Gb
    - Arranged as 2D matrix
    - Minimizes wire length
    - Maximizes refresh efficiency
  - Narrow data interface: 1 – 16 bits (x1, x4, x8, x16)
    - Cheap packages $\Rightarrow$ few bus pins
    - Pins are expensive
  - Narrow address interface:
    - Multiplexed address lines: row and column address
    - Signaled by RAS# and CAS# respectively

DRAM Operation

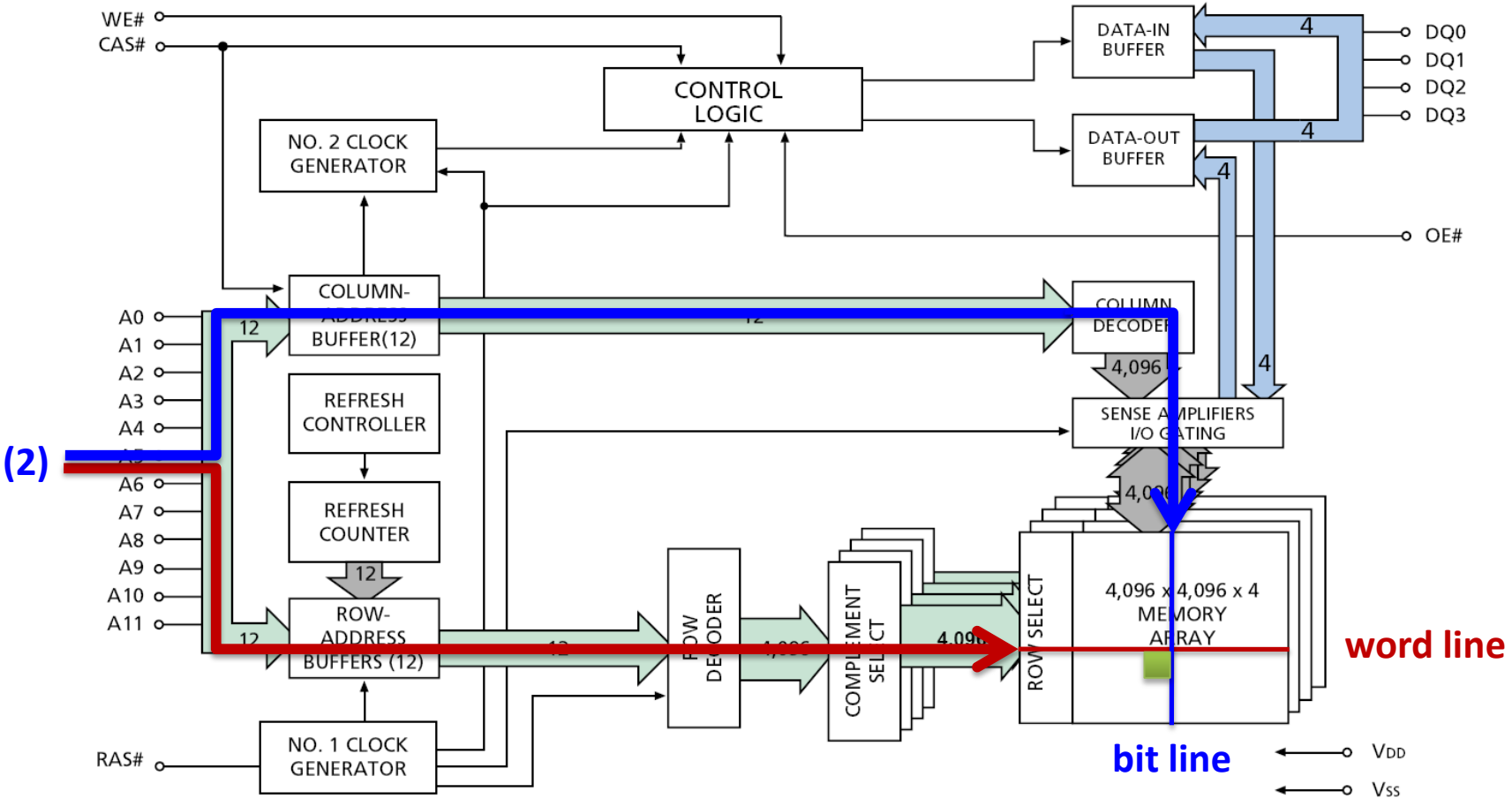
- **Read operation (1)**



[Source: ICE 3003 Lecture Notes by Prof. Jin-Soo Kim @ SKKU]

DRAM Operation

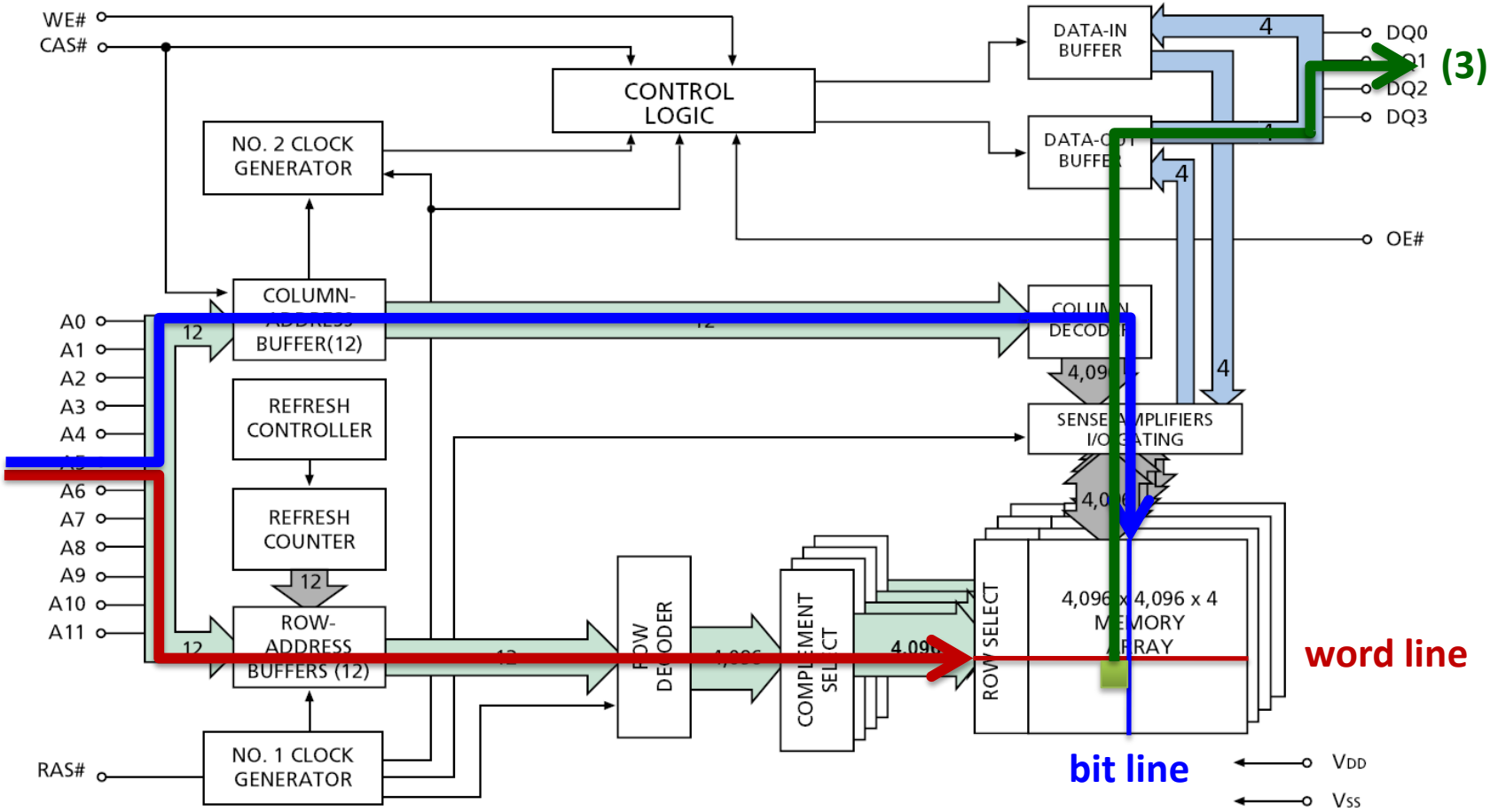
- **Read operation (2)**



[Source: ICE 3003 Lecture Notes by Prof. Jin-Soo Kim @ SKKU]

DRAM Operation

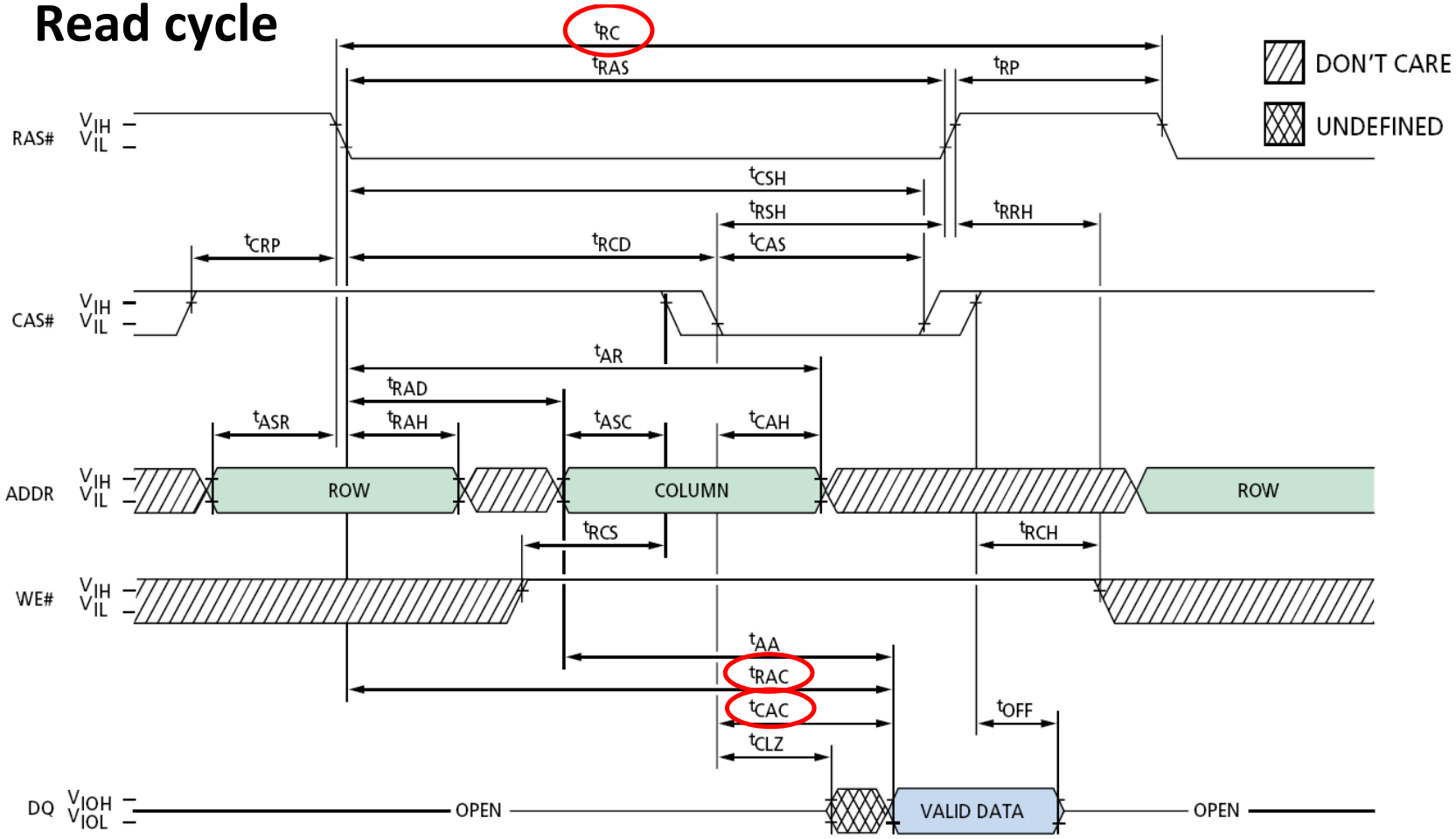
- **Read operation (3)**



[Source: ICE 3003 Lecture Notes by Prof. Jin-Soo Kim @ SKKU]

DRAM Operation

- **Read cycle**



[Source: ICE 3003 Lecture Notes by Prof. Jin-Soo Kim @ SKKU]

DRAM Operation

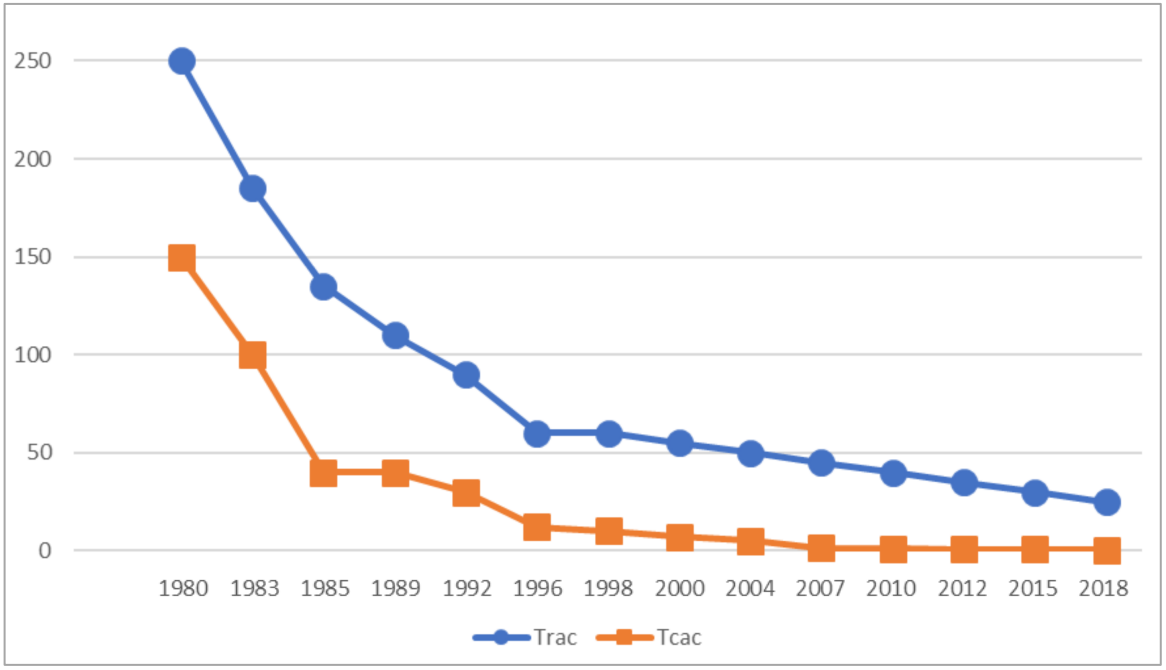
- **Timing parameters**
  - t_{RC}: minimum time from the start of one row access to the start of the next (cycle time)
  - t_{RAC}: minimum time from RAS# line falling to the valid data output (access time)
    - Used to be quoted as the nominal speed of a DRAM chip
  - t_{CAC}: minimum time from CAS# line falling to valid data output

Model	t_{RC}	t_{RAC}	t_{CAC}
MT4LC16M4T8-5	90 ns	50 ns	13 ns
MT4LC16M4T8-6	110 ns	60 ns	15 ns

[Source: ICE 3003 Lecture Notes by Prof. Jin-Soo Kim @ SKKU]

DRAM Generations

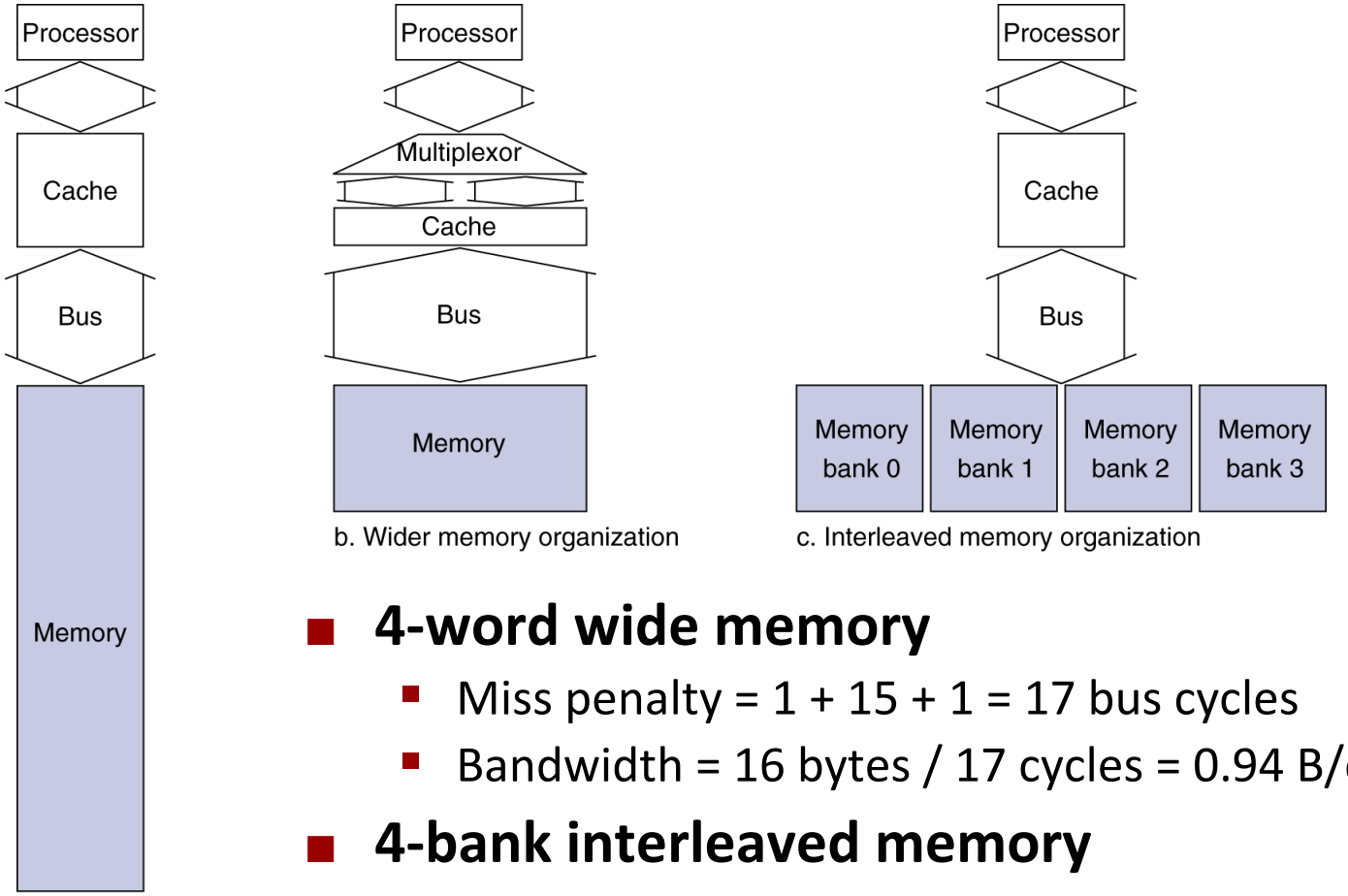
Year	Capacity	$/GB
1980	64 Kbit	$6,480,000
1983	256 Kbit	$1,980,000
1985	1 Mbit	$720,000
1989	4 Mbit	$128,000
1992	16 Mbit	$30,000
1996	64 Mbit	$9,000
1998	128 Mbit	$900
2000	256 Mbit	$840
2004	512 Mbit	$150
2007	1 Gbit	$40
2010	2 Gbit	$13
2012	4 Gbit	$5
2015	8 Gbit	$7
2018	16 Gbit	$6



DRAM Performance Factors

- **Row buffer**
  - Allows several words to be read and refreshed in parallel
- **Synchronous DRAM**
  - Allows for consecutive accesses in bursts without needing to send each address
  - Improves bandwidth
- **DRAM banking**
  - Allows simultaneous access to multiple DRAMs
  - Improves bandwidth

Increasing Memory Bandwidth



a. One-word-wide memory organization

b. Wider memory organization

c. Interleaved memory organization

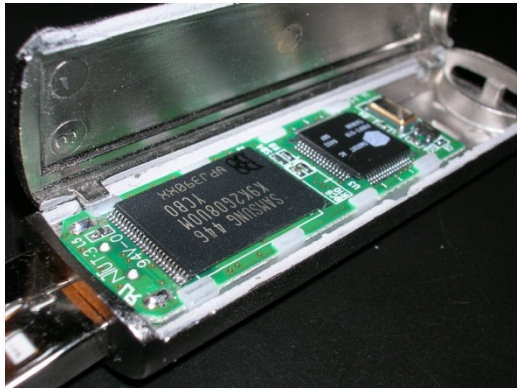
- **4-word wide memory**
  - Miss penalty = 1 + 15 + 1 = 17 bus cycles
  - Bandwidth = 16 bytes / 17 cycles = 0.94 B/cycle
- **4-bank interleaved memory**
  - Miss penalty = 1 + 15 + 4×1 = 20 bus cycles
  - Bandwidth = 16 bytes / 20 cycles = 0.8 B/cycle

Advanced DRAM Organizations

- **Bits in a DRAM are organized as a rectangular array**
  - DRAM accesses an entire row
  - Burst mode: supply successive words from a row with reduced latency
- **Double data rate (DDR) DRAM**
  - Transfer on rising and falling clock edges
- **Quad data rate (QDR) DRAM**
  - Separate DDR inputs and outputs

Flash Storage

- **Nonvolatile semiconductor storage**
  - 100× – 1000× faster than disk
  - Smaller, lower power, more robust
  - But more $/GB (between disk and DRAM)

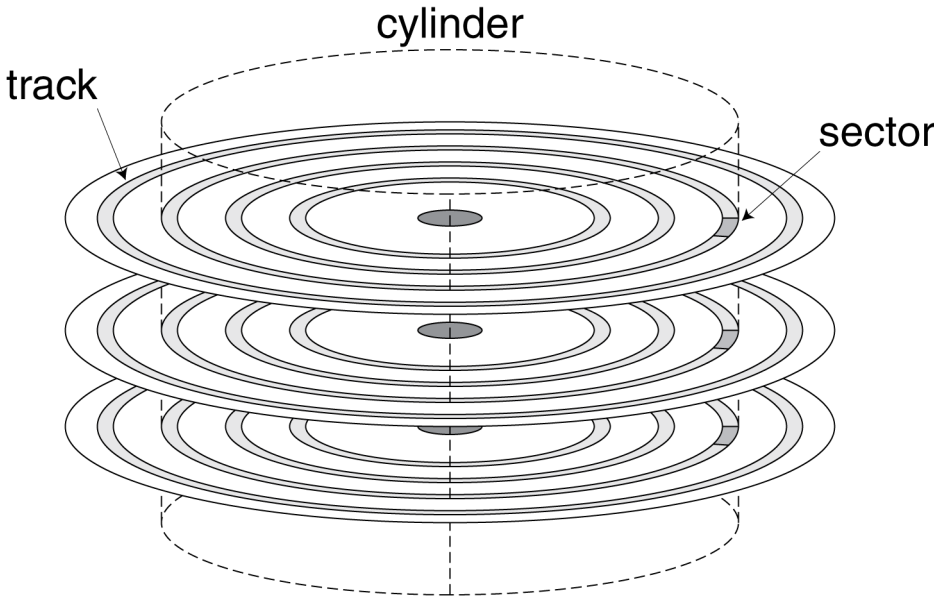


Flash Types

- **NOR flash: bit cell like a NOR gate**
  - Random read/write access
  - Used for instruction memory in embedded systems
- **NAND flash: bit cell like a NAND gate**
  - Denser (bits/area), but block-at-a-time access
  - Cheaper per GB
  - Used for USB keys, media storage, …
- **Flash bits wears out after 1000's of accesses**
  - Not suitable for direct RAM or disk replacement
  - Wear leveling: remap data to less used blocks

Disk Storage

- **Nonvolatile, rotating magnetic storage**



Disk Sectors and Access

- **Each sector records**
  - Sector ID
  - Data (512 bytes, 4096 bytes proposed)
  - Error correcting code (ECC)
    - Used to hide defects and recording errors
  - Synchronization fields and gaps
- **Access to a sector involves**
  - Queuing delay if other accesses are pending
  - Seek: move the heads
  - Rotational latency
  - Data transfer
  - Controller overhead

Disk Access Example

- **Given**
  - 512B sector, 15,000rpm, 4ms average seek time, 100MB/s transfer rate, 0.2ms controller overhead, idle disk
- **Average read time**
  - 4ms seek time
    + ½ / (15,000/60) = 2ms rotational latency
    + 512 / 100MB/s = 0.005ms transfer time
    + 0.2ms controller delay
    = 6.2ms
- **If actual average seek time is 1ms**
  - Average read time = 3.2ms

Disk Performance Issues

- **Manufacturers quote average seek time**
  - Based on all possible seeks
  - Locality and OS scheduling lead to smaller actual average seek times
- **Smart disk controller allocate physical sectors on disk**
  - Present logical sector interface to host
  - SCSI, ATA, SATA
- **Disk drives include caches**
  - Prefetch sectors in anticipation of access
  - Avoid seek and rotational delay

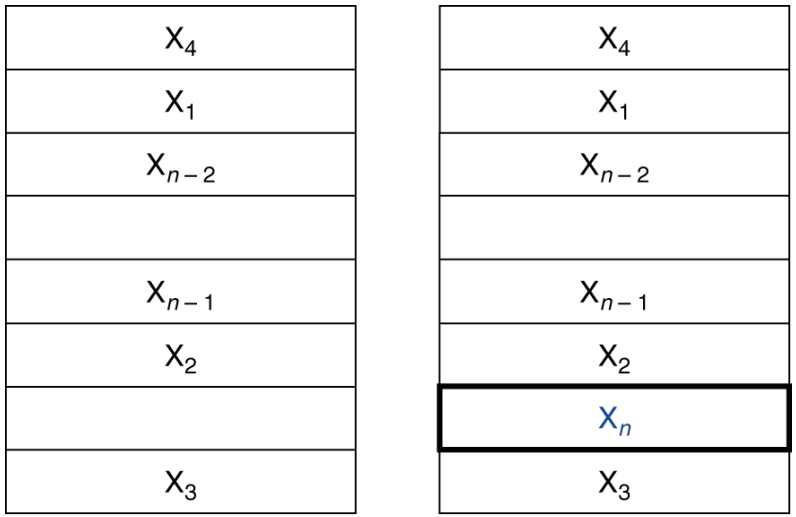
Outline

Textbook: COD 5.1 – 5.3

- Introduction
- Memory Technologies
- **The Basics of Caches**

Cache Memory

- **Cache memory**
  - The level of the memory hierarchy closest to the CPU
- **Given accesses X_1, …, X_{n-1}, X_n**

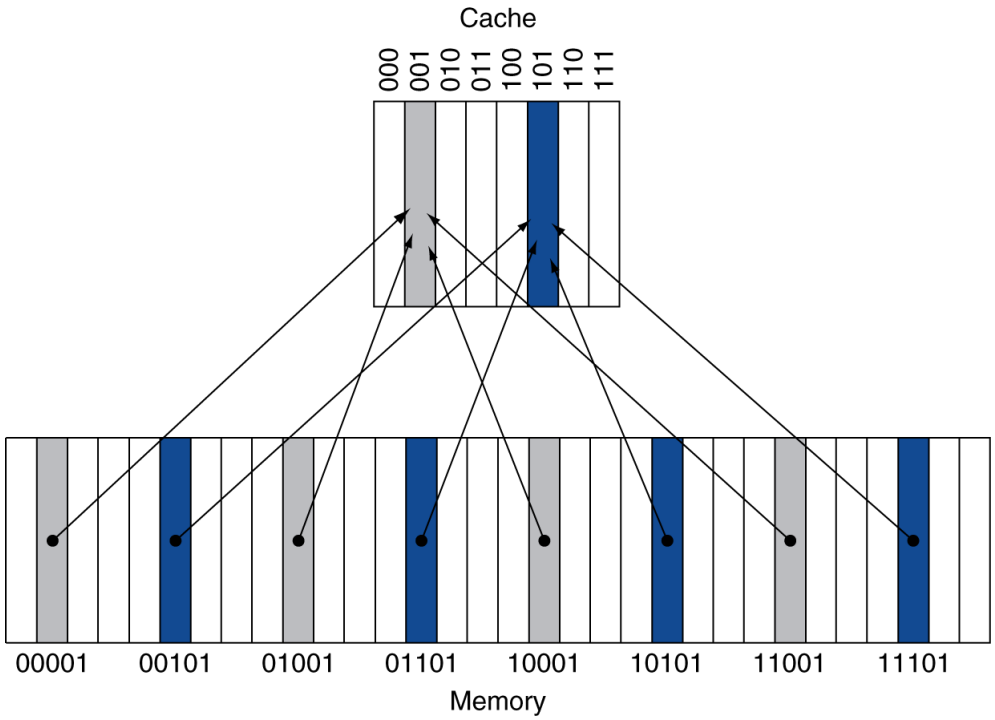


a. Before the reference to X_n b. After the reference to X_n

- **How do we know if the data is present?**
- **Where do we look?**

Direct Mapped Cache

- **Location determined by address**
- **Direct mapped: only one choice**
  - (Block address) modulo (#Blocks in cache)



- **#Blocks is a power of 2**
- **Use low-order address bits**

Tags and Valid Bits

- **How do we know which particular block is stored in a cache location?**
  - Store block address as well as the data
  - Actually, only need the high-order bits
  - Called the tag
- **What if there is no data in a location?**
  - Valid bit: 1 = present, 0 = not present
  - Initially 0

Cache Example

- 8-blocks, 1 word/block, direct mapped
- Initial state

Index	V	Tag	Data
000	N		
001	N		
010	N		
011	N		
100	N		
101	N		
110	N		
111	N		

Cache Example

Word addr	Binary addr	Hit/miss	Cache block
22	10 110	Miss	110

Index	V	Tag	Data
000	N		
001	N		
010	N		
011	N		
100	N		
101	N		
110	**Y**	**10**	**Mem[10110]**
111	N		

Cache Example

Word addr	Binary addr	Hit/miss	Cache block
26	11 010	Miss	010

Index	V	Tag	Data
000	N		
001	N		
010	**Y**	**11**	**Mem[11010]**
011	N		
100	N		
101	N		
110	Y	10	Mem[10110]
111	N		

Cache Example

Word addr	Binary addr	Hit/miss	Cache block
22	10 110	Hit	110
26	11 010	Hit	010

Index	V	Tag	Data
000	N		
001	N		
010	Y	11	Mem[11010]
011	N		
100	N		
101	N		
110	Y	10	Mem[10110]
111	N		

Cache Example

Word addr	Binary addr	Hit/miss	Cache block
16	10 000	Miss	000
3	00 011	Miss	011
16	10 000	Hit	000

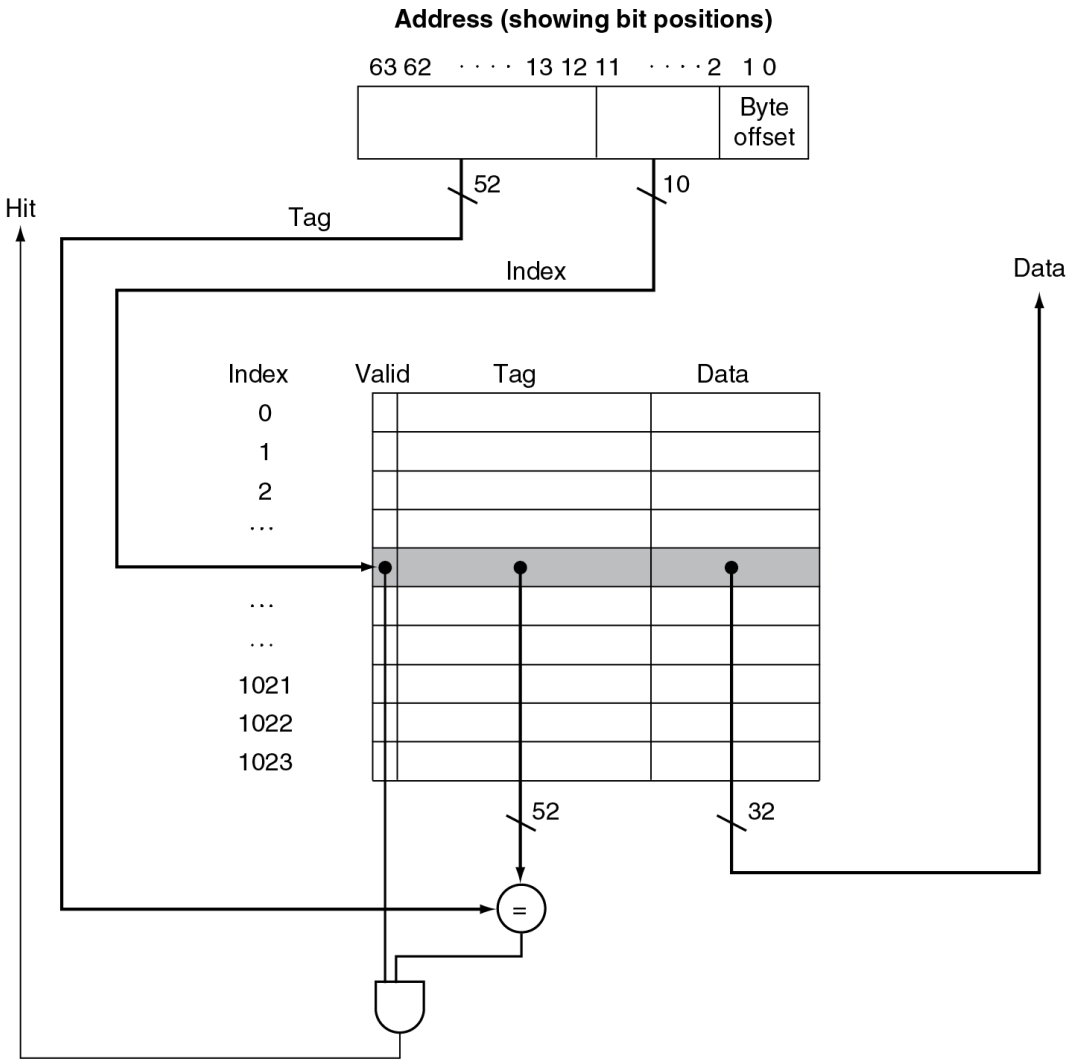
Index	V	Tag	Data
000	**Y**	**10**	**Mem[10000]**
001	N		
010	Y	11	Mem[11010]
011	**Y**	**00**	**Mem[00011]**
100	N		
101	N		
110	Y	10	Mem[10110]
111	N		

Cache Example

Word addr	Binary addr	Hit/miss	Cache block
18	10 010	Miss	010

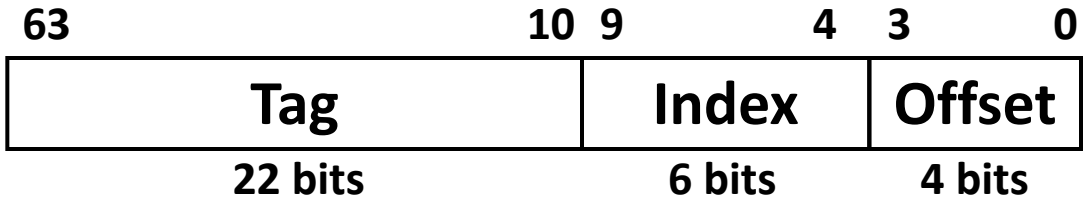
Index	V	Tag	Data
000	Y	10	Mem[10000]
001	N		
010	**Y**	**10**	**Mem[10010]**
011	Y	00	Mem[00011]
100	N		
101	N		
110	Y	10	Mem[10110]
111	N		

Address Subdivision



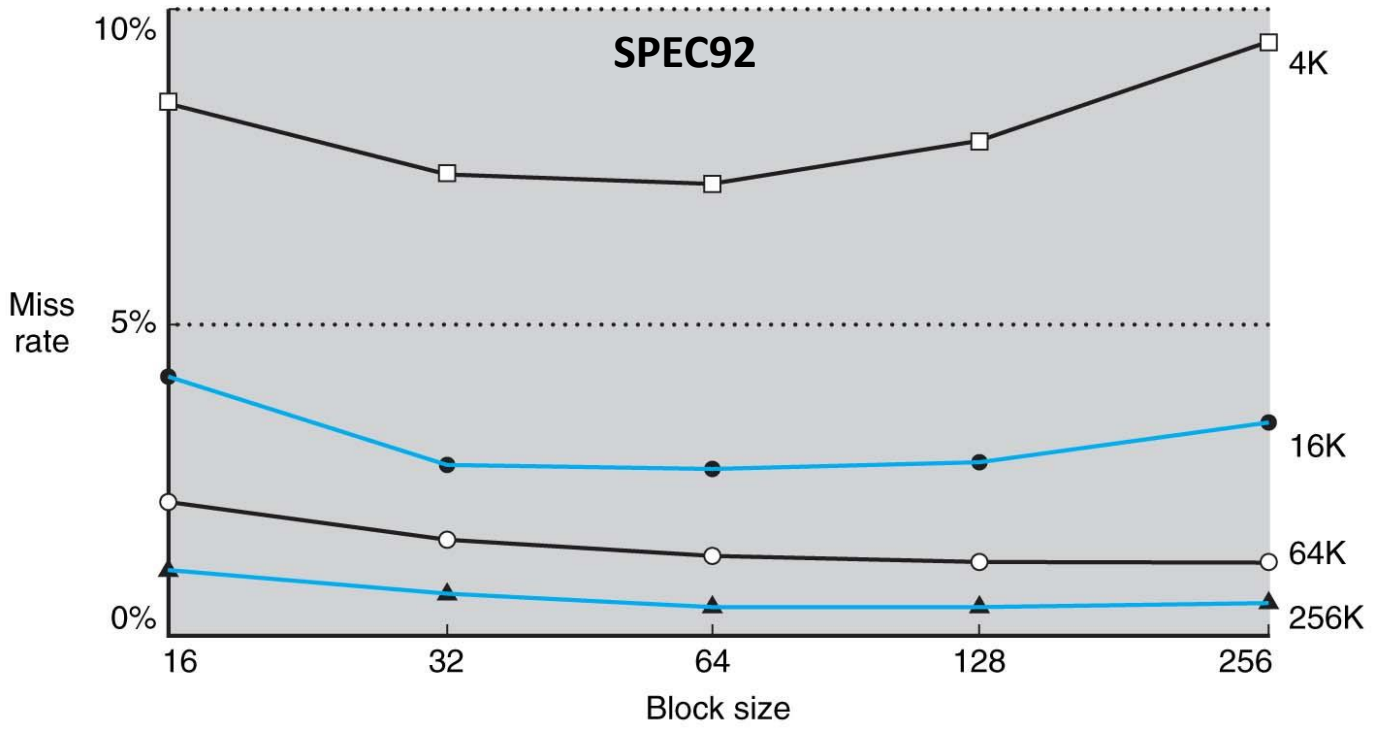
Example: Larger Block Size

- **64 blocks, 16 bytes/block**
  - To what block number does address 1200 map?
- **Block address = $\lfloor 1200/16 \rfloor$ = 75**
- **Block number = 75 modulo 64 = 11**



Block Size: Considerations

- **Considerations**
  - Larger blocks should reduce miss rate
    - Due to spatial locality



Block Size: Considerations

- **Considerations (cont'd)**
  - But in a fixed-sized cache
    - Larger blocks $\Rightarrow$ fewer of them
      - More competition $\Rightarrow$ increased miss rate
    - Larger blocks $\Rightarrow$ pollution
  - Larger miss penalty
    - Can override benefit of reduced miss rate
    - Early restart and critical-word-first can help

Cache Misses

- **On cache hit, CPU proceeds normally**
- **On cache miss**
  - Stall the CPU pipeline
  - Fetch block from next level of hierarchy
  - Instruction cache miss
    - Restart instruction fetch
  - Data cache miss
    - Complete data access

Write-Through

- **On data-write hit, could just update the block in cache**
  - But then cache and memory would be inconsistent
- **Write through: also update memory**
- **But makes writes take longer**
  - e.g., if base CPI = 1, 10% of instructions are stores, write to memory takes 100 cycles
    - Effective CPI = 1 + 0.1×100 = 11
- **Solution: write buffer**
  - Holds data waiting to be written to memory
  - CPU continues immediately
    - Only stalls on write if write buffer is already full

Write-Back

- **Alternative: On data-write hit, just update the block in cache**
  - Keep track of whether each block is dirty
- **When a dirty block is replaced**
  - Write it back to memory
  - Can use a write buffer to allow replacing block to be read first

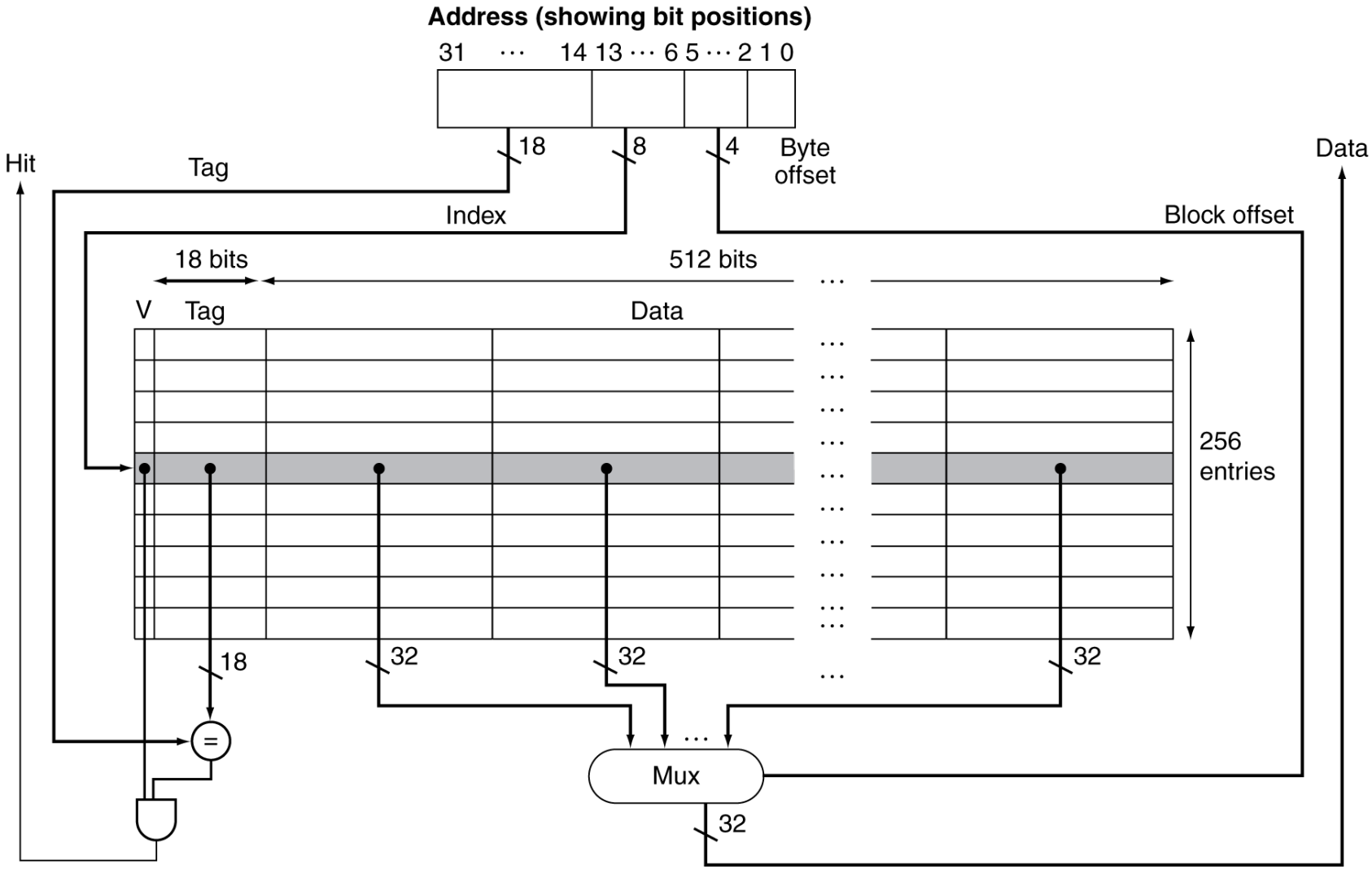
Write Allocation

- **What should happen on a write miss?**
- **Alternatives for write-through**
  - Allocate on miss: fetch the block
  - Write around: don't fetch the block
    - Since programs often write a whole block before reading it (e.g., initialization)
- **For write-back**
  - Usually fetch the block

Example: Intrinsity FastMATH

- **Embedded MIPS processor**
  - 12-stage pipeline
  - Instruction and data access on each cycle
- **Split cache: separate I-cache and D-cache**
  - Each 16KB: 256 blocks × 16 words/block
  - D-cache: write-through or write-back
- **SPEC2000 miss rates**
  - I-cache: 0.4%
  - D-cache: 11.4%
  - Weighted average: 3.2%

Example: Intrinsity FastMATH



Main Memory Supporting Caches

- **Use DRAMs for main memory**
  - Fixed width (e.g., 1 word)
  - Connected by fixed-width clocked bus
    - Bus clock is typically slower than CPU clock
- **Example cache block read**
  - 1 bus cycle for address transfer
  - 15 bus cycles per DRAM access
  - 1 bus cycle per data transfer
- **For 4-word block, 1-word-wide DRAM**
  - Miss penalty = 1 + 4×15 + 4×1 = 65 bus cycles
  - Bandwidth = 16 bytes / 65 cycles = 0.25 B/cycle