

## Topic 5

### Memories I

**Xinfei Guo**  
**[xinfei.guo@sjtu.edu.cn](mailto:xinfei.guo@sjtu.edu.cn)**

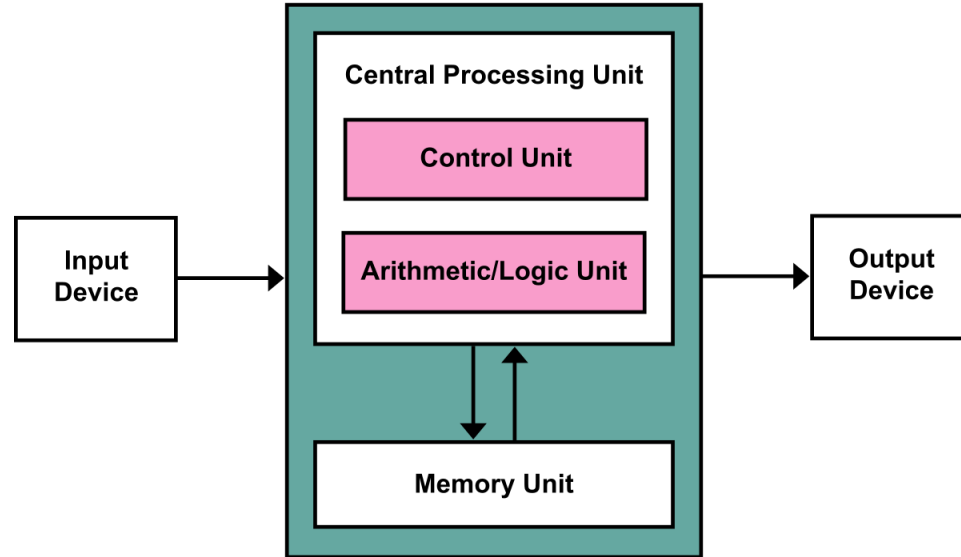
**June 13<sup>th</sup>, 2022**



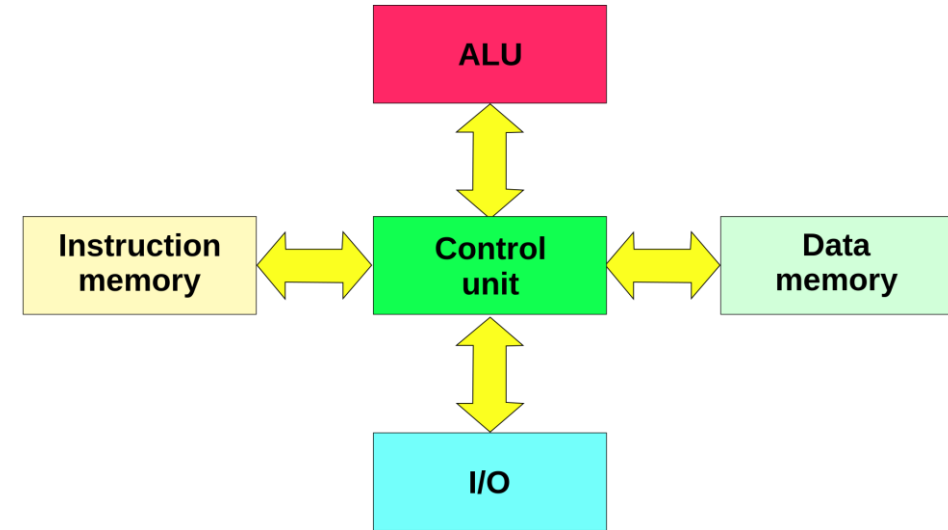
# T5 learning goals

- Memories
  - Memories I: Memory Overview
  - Memories II: Cache
  - Memories III: Memory Management

# Motivation (1/2)



Von Neumann architecture



Harvard architecture



## SSD VS HDD



- FASTER PERFORMANCE
- NO VIBRATIONS OR NOISE
- MORE ENERGY EFFICIENT

- CHEAPER PER GB
- AVAILABLE IN LARGE VERSIONS



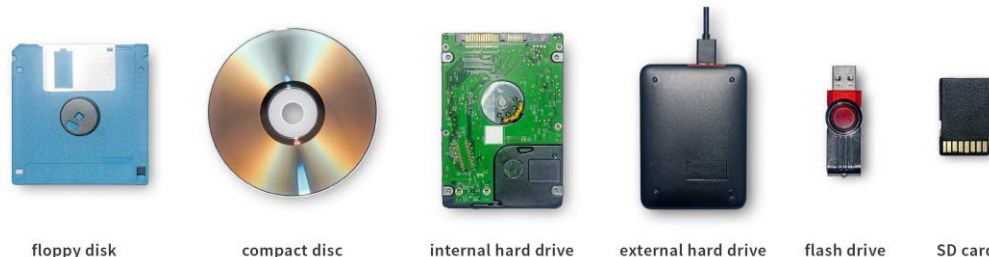
# Motivation (2/2)

- When a processor runs an application, it needs access to its instructions and data.
- These have to be placed in physical storage locations for the processor to easily access.
  - Instructions and data for one program also need isolating from other programs.
- Understanding their designs gives insight into their trade-offs.
- Understanding how they are used gives insight into how hardware can help.

# Volatile vs Non-volatile Memory

- Volatile memory
  - Requires power to retain the data information
  - Usually, faster access speed and less costly
  - Used for temporary data storage, such as CPU cache, internal memory
  - Also known as Random Access Memory (RAM)
- Non-volatile memory
  - No power is required to retain the data information
  - Usually, slower access speed and more costly
  - Used for secondary storage, or long-term persistent storage

Examples of Non-Volatile Memory



floppy disk

compact disc

internal hard drive

external hard drive

flash drive

SD card

# Quick Quiz

- Which of the following memory is the cheapest?

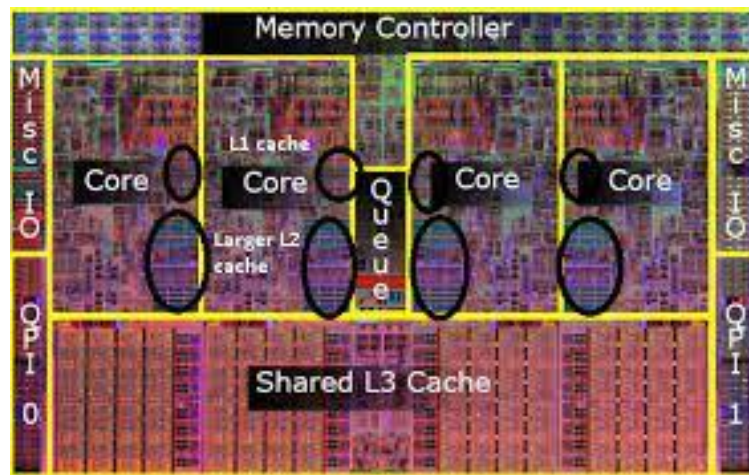
A



B



C



D



Figures: Internet

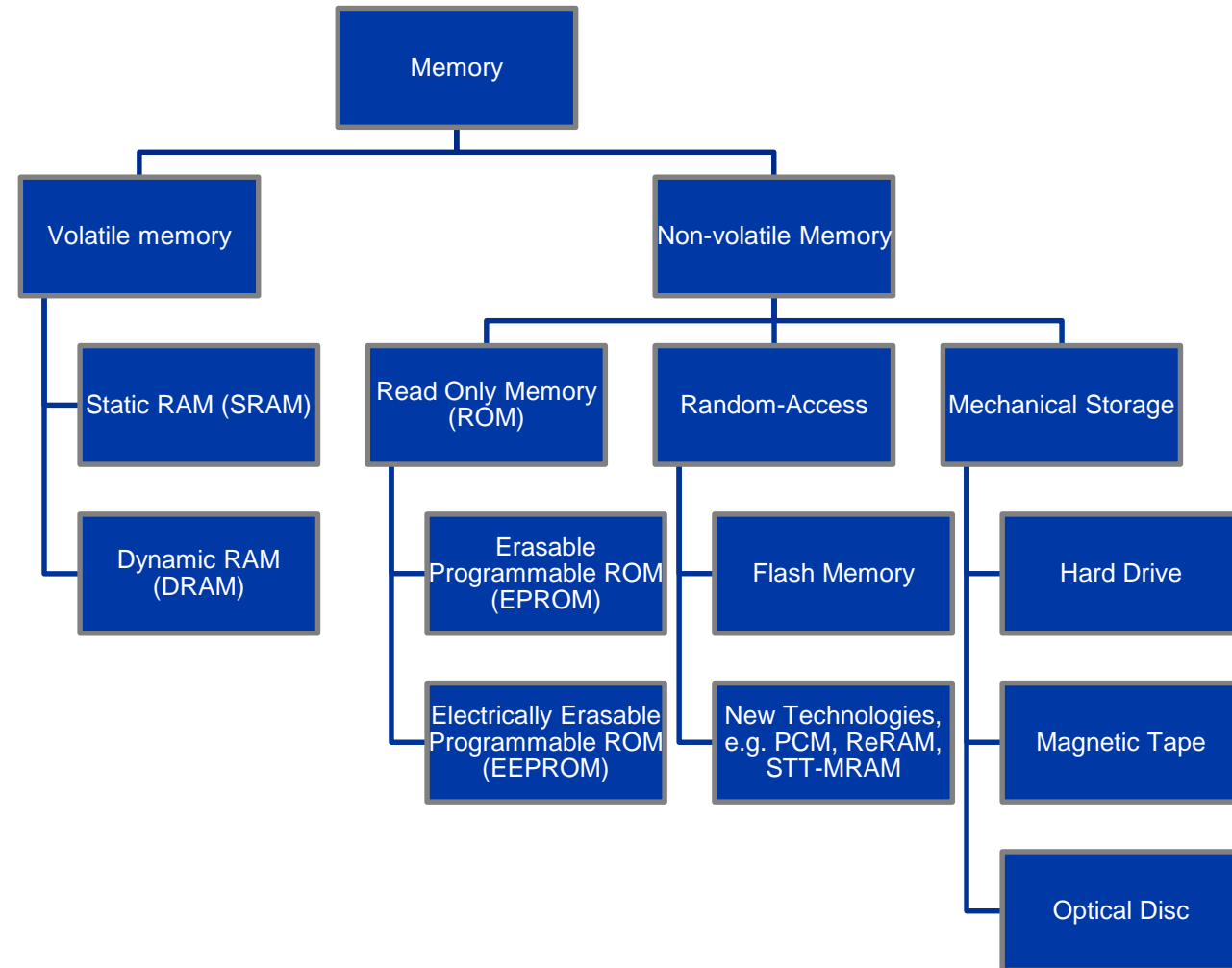
# Types of Memory

## Volatile memory

- Static RAM (SRAM)
- Dynamic RAM (DRAM)

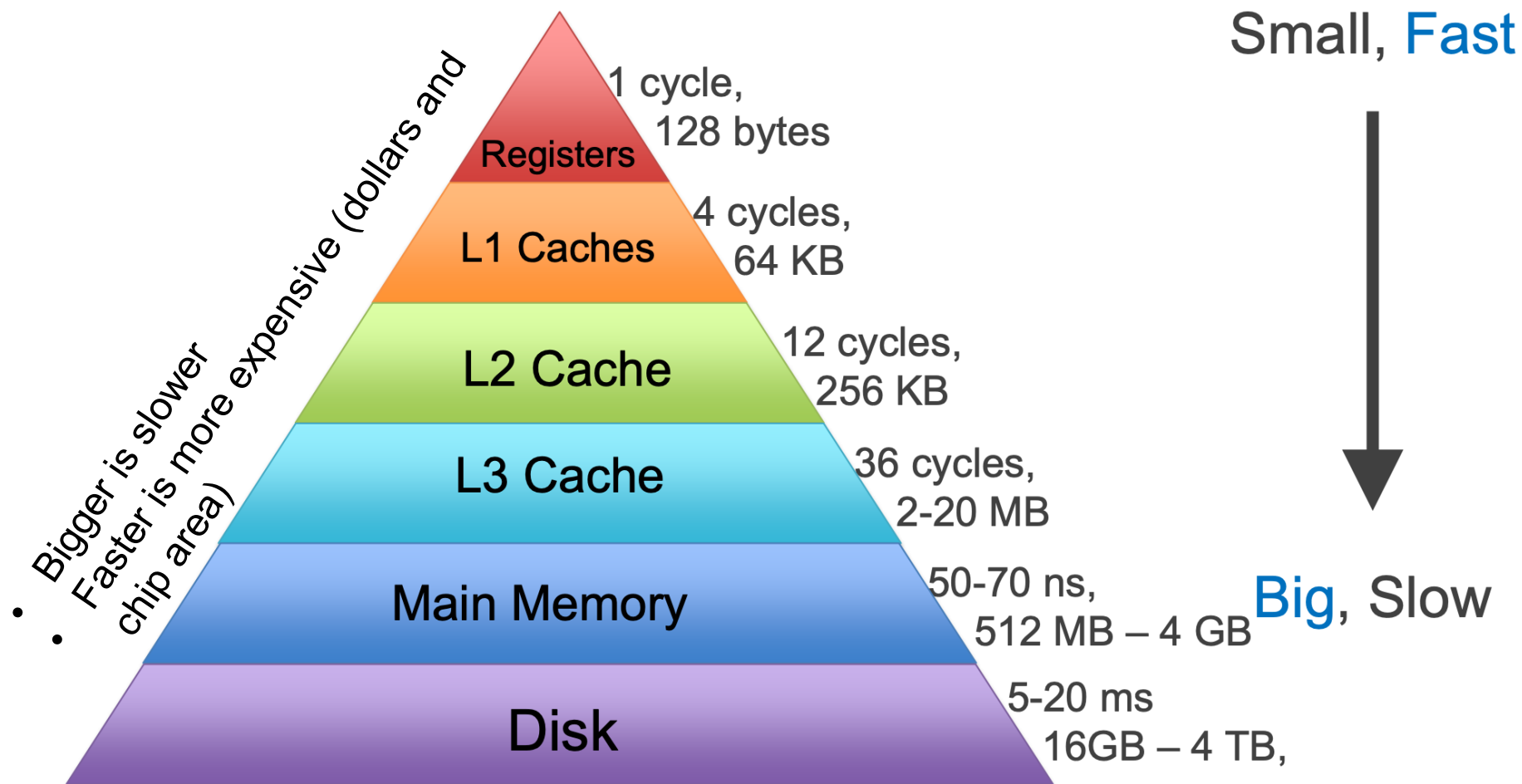
## Non-volatile memory

- Read only memory (ROM)
  - Erasable programmable ROM (EPROM)
  - Electrically erasable programmable ROM (EEPROM)
- Non-volatile random-access memory (NVRAM)
  - Flash memory
- Mechanical storage
  - Hard drive, magnetic tape





# Memory Hierarchy Levels

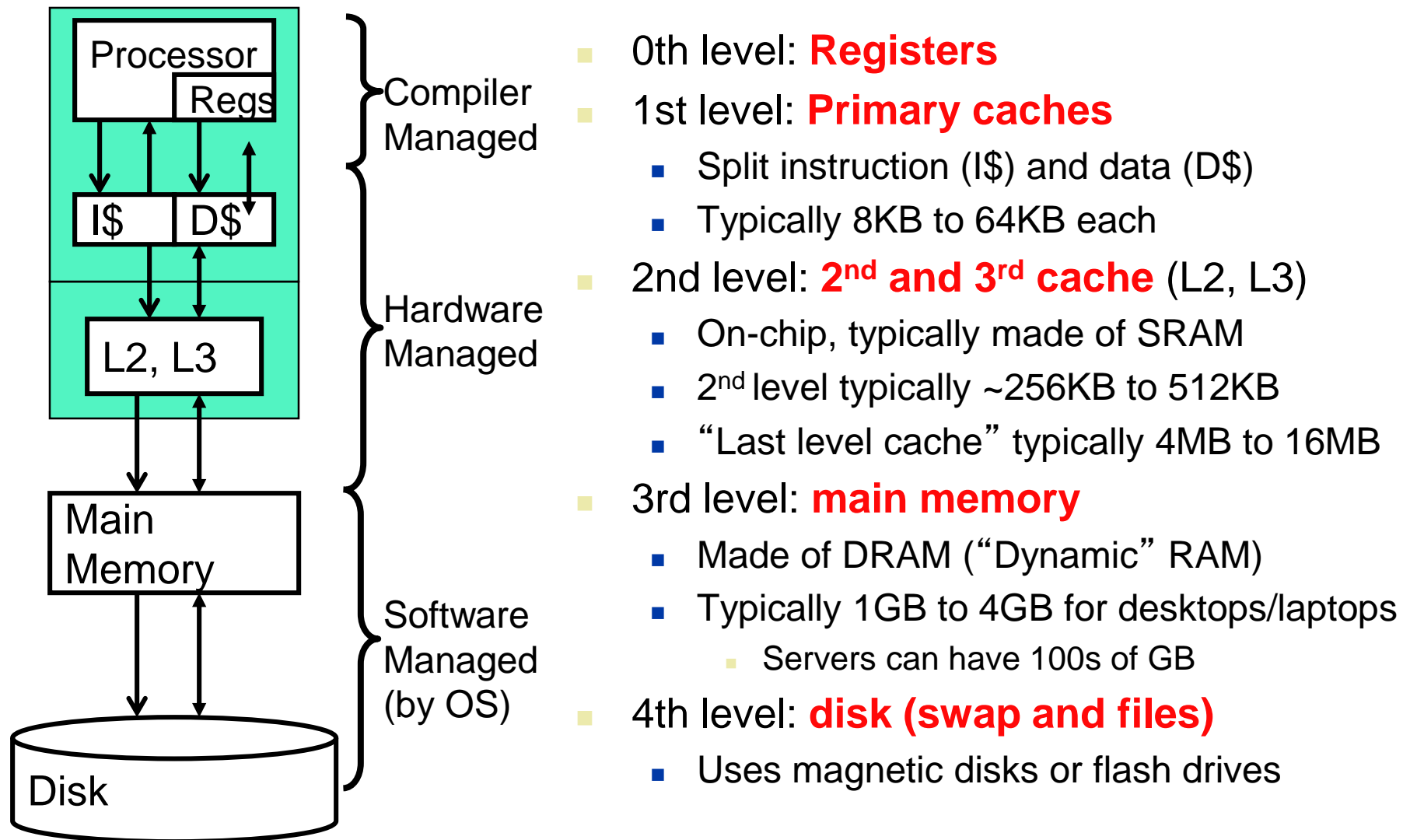


Intel Haswell Processor, 2013

Image: [cs.cornell.edu/courses/cs3410/](http://cs.cornell.edu/courses/cs3410/)



# Concrete Memory Hierarchy



source: Prof. Joe Devietti @ U Penn

# Non-Volatile Memory

## Flash Memory

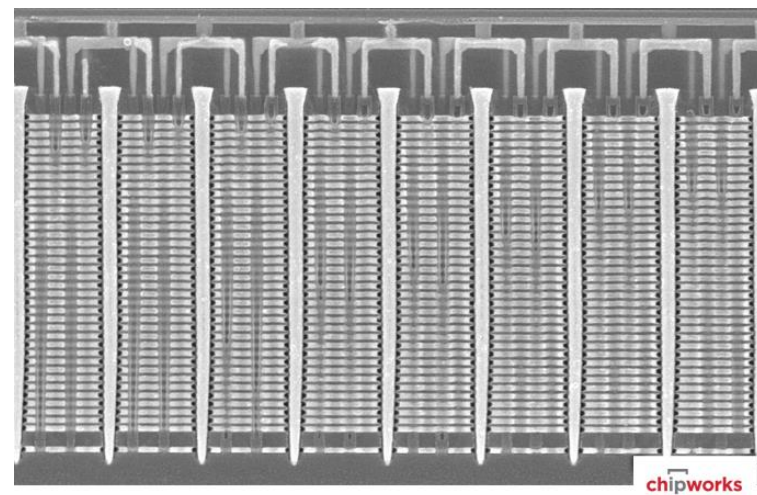
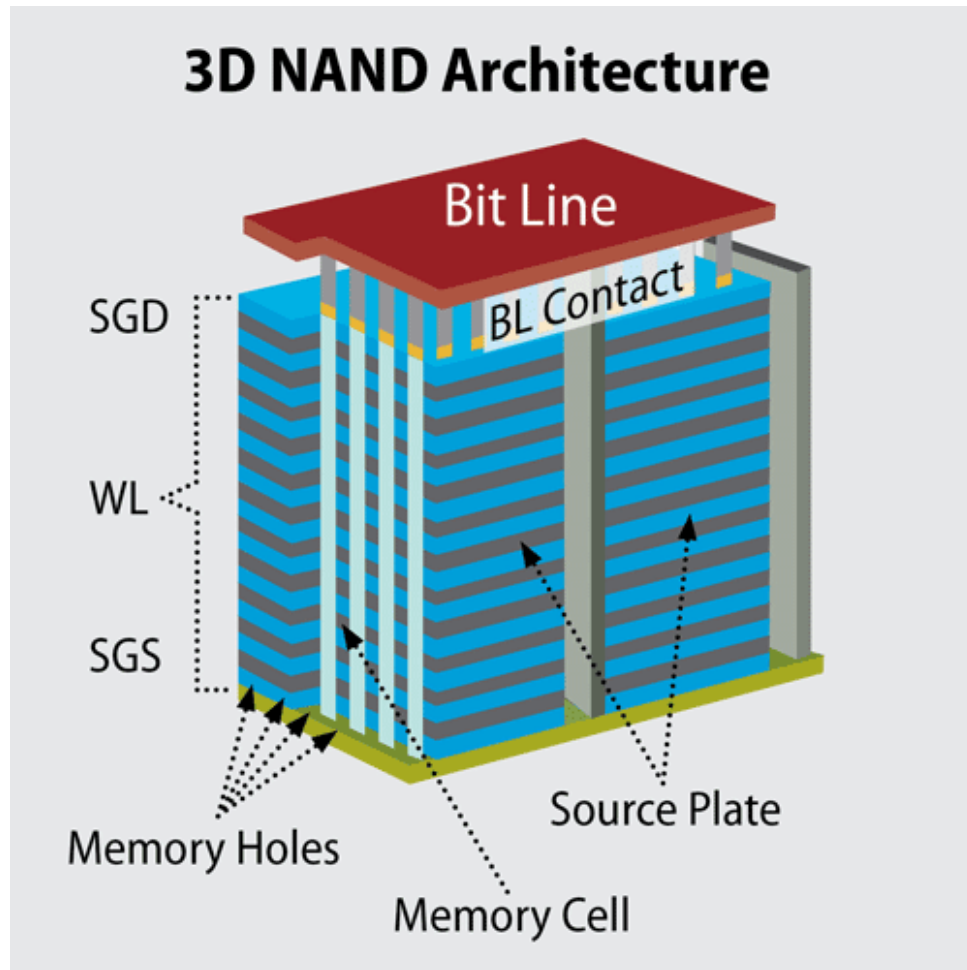
- NOR flash: bit cell like a NOR gate
  - Random read/write access
  - Used for instruction memory in embedded systems
- NAND flash: bit cell like a NAND gate
  - Denser (bits/area), but block-at-a-time access
  - Cheaper per GB
  - Used for USB keys, media storage, ...
- Flash bits wears out after 1000's of accesses
  - Wear leveling: remap data to less used blocks

# Flash Memory

	NAND	AND	NOR
Cell Array			
Layout			
Cross-section			
Cell size	$4F^2$	$8F^2$	$10F^2$

**Multi-level cell (MLC): Stores multiple bits per cell.**

# 3D NAND Flash

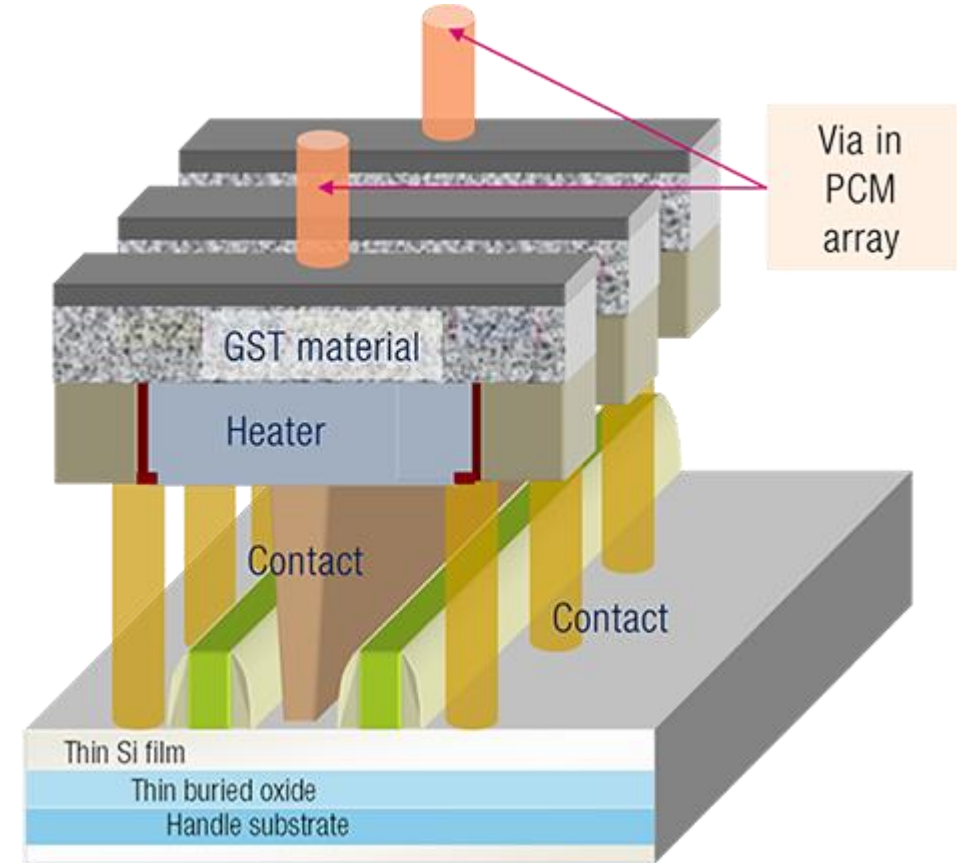


Source: Samsung

Image: <https://www.enterprisestorageforum.com/hardware/3d-nand/>

# Phase-Change Memory Technology

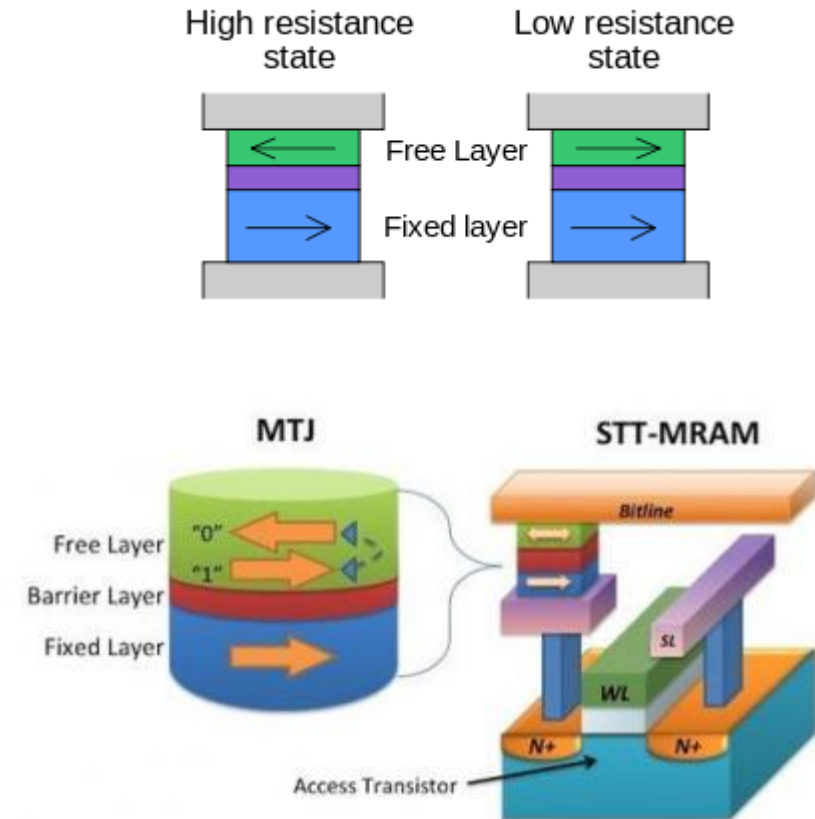
- Uses a small heating element to change the state of a bulk substrate between its crystalline form and an amorphous form, which have different resistive properties.
- Possibly 10X improvement in write performance and 2X improvement in read performance compared to NAND.



source: [https://www.st.com/content/st\\_com/en/about/innovation---technology/PCM.html](https://www.st.com/content/st_com/en/about/innovation---technology/PCM.html)

# STT-MRAM

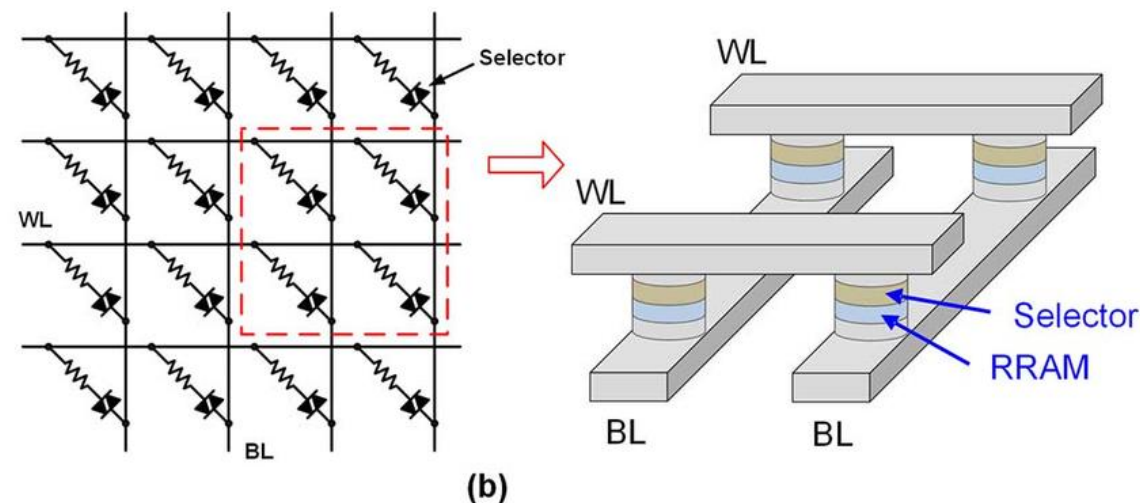
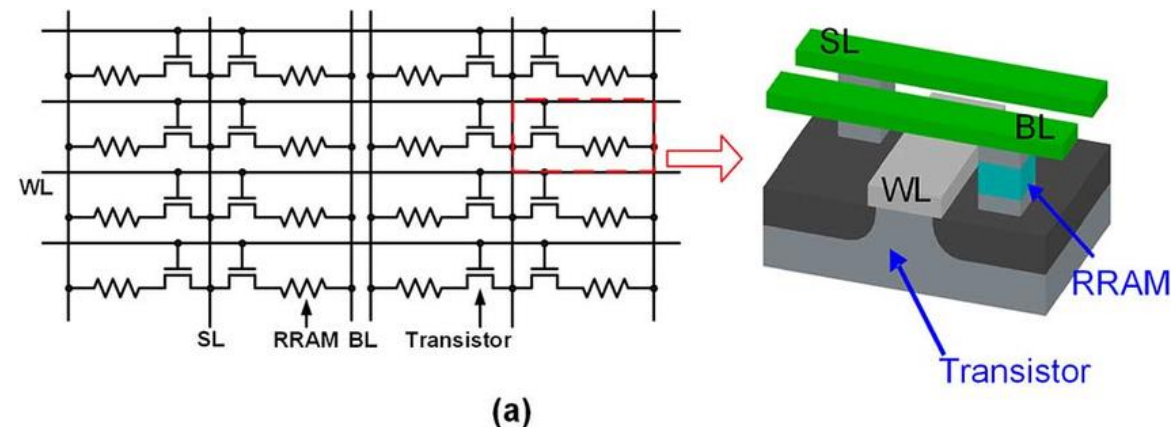
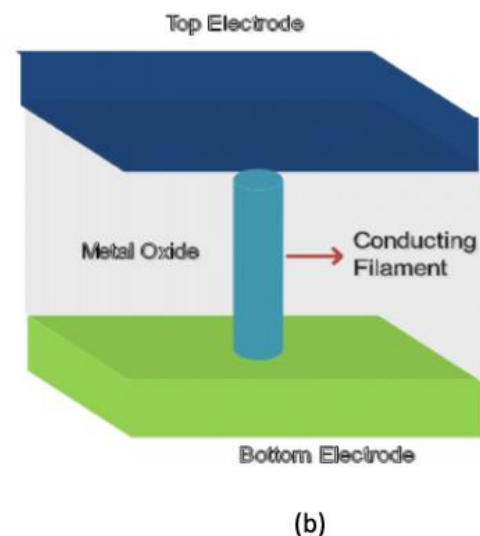
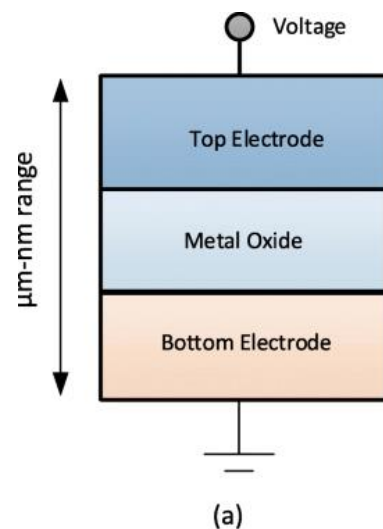
- Spin-transfer torque magnetic random-access memory
- Spin-transfer torque (STT) is an effect in which the orientation of a magnetic layer in a magnetic tunnel junction or spin valve can be modified using a spin-polarized current.
- near zero leakage



source: Wiki

# Resistive RAM (RRAM)











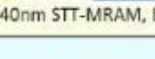















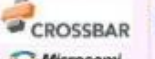










- works by changing the resistance across a dielectric solid-state material, often referred to as a memristor.



R. Liu et al. Investigation of Single-Bit and Multiple-Bit Upsets in Oxide RRAM-Based 1T1R and Crossbar Memory Arrays



# Emerging Memory Mass-Products: Major Players

MRAM STT-MRAM	 (180nm, MR2A)   (180nm, MR4A)	 (Aeroflex, UT8MR)	 (90nm, EMD3D64)	 (150nm, HXNV)	   (90nm, CT32)		 <b>SONY</b> (55nm, AS008MA)   (28nm, STT-MRAM)   (22FDX, eMRAM)  (1Gb, 28nm STT-MRAM)   (256Mb, 40nm STT-MRAM, EMD3D256)	 (22nm, eMRAM)   (22FDX, eMRAM)	 <b>UMC</b> (2Xnm, eMRAM)
PCRAM XPoint	 (90nm, NP8P)	 (65nm, K571229)	 (1Gb PCM+LPDDR2)			 (128Gb, Optane SSD)	 (Optane DC, NVDIMM)	 (XPoint: QuantX)	
ReRAM Memristor OxRAM CBRAM			 (180nm, MN101 MCU)	 (130nm, RM24)	 <b>Panasonic</b> (4Mb, MB85AS4MT)	 <b>SMIC</b> (40nm, 8Mb)	 (130nm, RM331x)   <b>Microsemi</b> (28nm)	 (22nm, eReRAM)	 (1X nm, eReRAM)   <b>UMC</b> (40 nm, OxRAM)
FeRAM & Others	 (130nm, XMS430)		 (180nm, MB89R)   (2Mb, MR45V200B)	 (130nm, CY15B)		 (LP, MR45V100A)	 (4/8Mb, MB85R)	 (DDR4, NRAM)	
	~ 2012	2013	2014	2015	2016	2017	2018	2019	2020

TechnInsights CONFIDENTIAL. All content © 2018. TechnInsights Inc. All rights reserved.

**Tech  
Insights**

# A quick comparison

Parameters	Typical memory technology			New memory technology				
	SRAM	DRAM	Flash (NAND)	FeRAM	ReRAM	PCRAM	STT-MRAM	SOT-MRAM
Non-volatility	No	No	Yes	Yes	Yes	Yes	Yes	Yes
Cell size (F <sup>2</sup> )	50-120	6-10	5	15-34	6-10	4-19	6-20	6-20
Read time (ns)	≤2	30	10 <sup>3</sup>	≈5	1-20	≈2	1-20	≤10
Write time (ns)	≤2	50	10 <sup>6</sup>	≈10	50	10 <sup>2</sup>	≈10	≤10
Write power	Low	Low	High	Low	Medium	Low	Low	Low
Endurance (cycles)	10 <sup>16</sup>	10 <sup>16</sup>	10 <sup>5</sup>	10 <sup>12</sup>	10 <sup>6</sup>	10 <sup>10</sup>	10 <sup>15</sup>	10 <sup>15</sup>
Future scalability	Good	Limited	Limited	Limited	Medium	Limited	Good	Good

source: Enlong Liu, Materials and designs of magnetic tunnel junctions with perpendicular magnetic anisotropy for high-density memory applications

# Volatile Memory

- Although the system contains many types of memory, we'll focus on the following memory for now.
  - SRAM, for caches
  - DRAM, for main memory
- These are the types of memory most commonly found in microprocessors.
  - And since they are close to the CPU, their interactions most commonly need to be considered by microarchitects.

# Memory Technology Basics

## SRAM vs DRAM

### SRAM – Static Random Access Memory

- Static – holds data as long as power is maintained
- Requires multiple transistors to retain one bit and has low density compared to DRAM, thus more expensive
- Faster than DRAM
- Used for caches (next module)

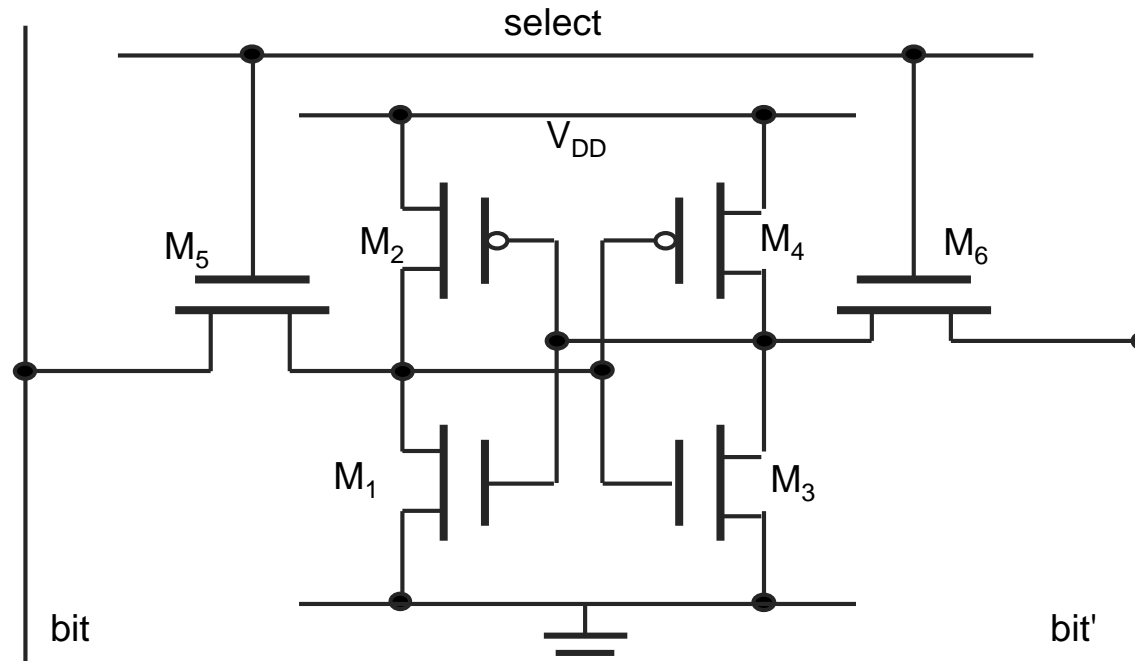
### DRAM – Dynamic Random Access Memory

- Dynamic – must be refreshed periodically to hold data
- Requires only one transistor (and one capacitor) to retain one bit of data
- High density, thus cheaper than SRAM
- Used for main memory and sometimes for larger caches

# SRAM cell

An SRAM cell is typically made up of six transistors (MOSFETs).

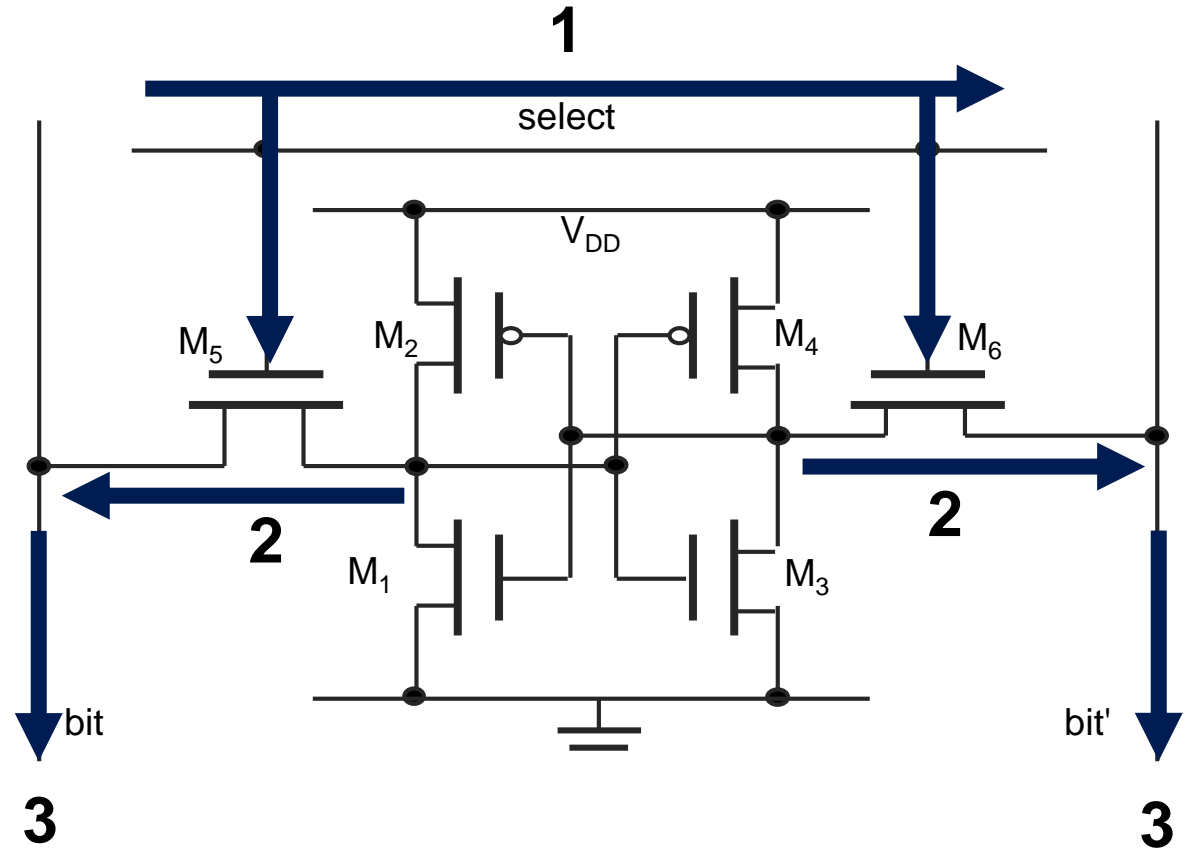
- A single bit is stored on 4 transistors (M1-M4), which form two inverters that are cross-coupled.
- Access to the bit is controlled by two access transistors (M5 and M6), which are gated by the word line (select).
- Data are read in and out through the bit lines.



# Accessing SRAM

## Read operation

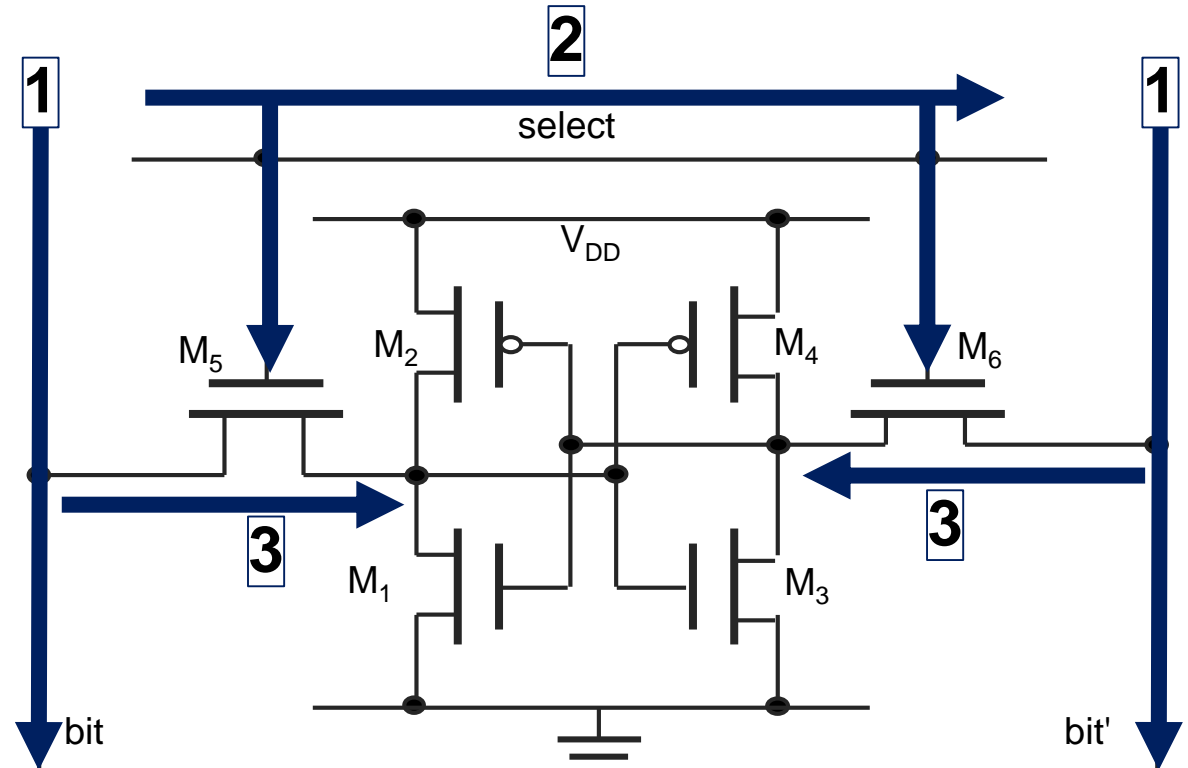
- The address is decoded and the desired cell is then selected, in which case the select line is set to one.
- Depending on the value of the 4 transistors (M1-M4), one of the bit lines (bit or bit') will be charged to 1 and the other will be drained to 0.
- The states of the two bit lines are then read out as 1-bit data.



# Accessing SRAM

## Write operation

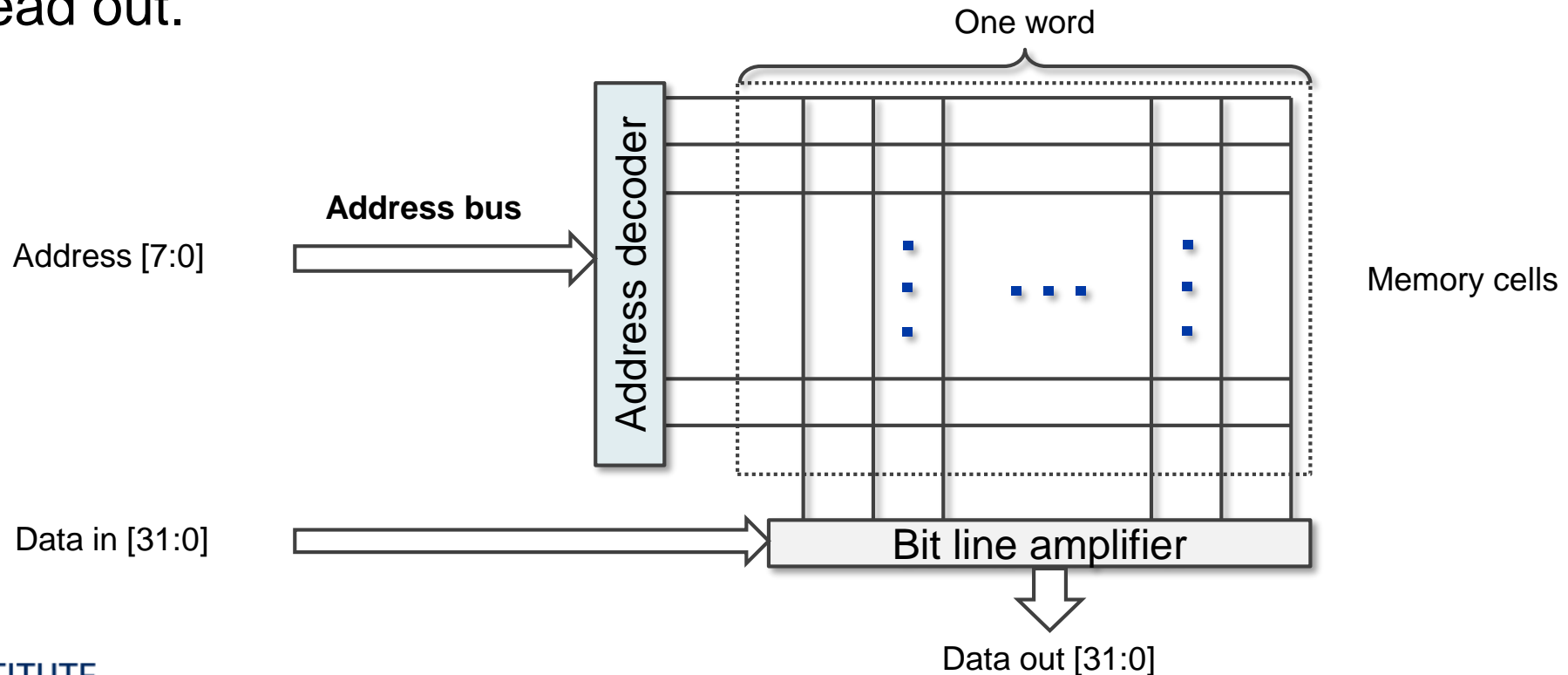
- The two bit lines are pre-charged to the desired value (e.g., bit = VDD, bit' = VSS).
- The address is decoded and the desired cell is then selected, in which case the select line is set to one.
- The 4 transistors (M1-M4) are then forced to flip their states (either charged or discharged) since the bit lines normally have much higher capacitance than the 4 transistors.



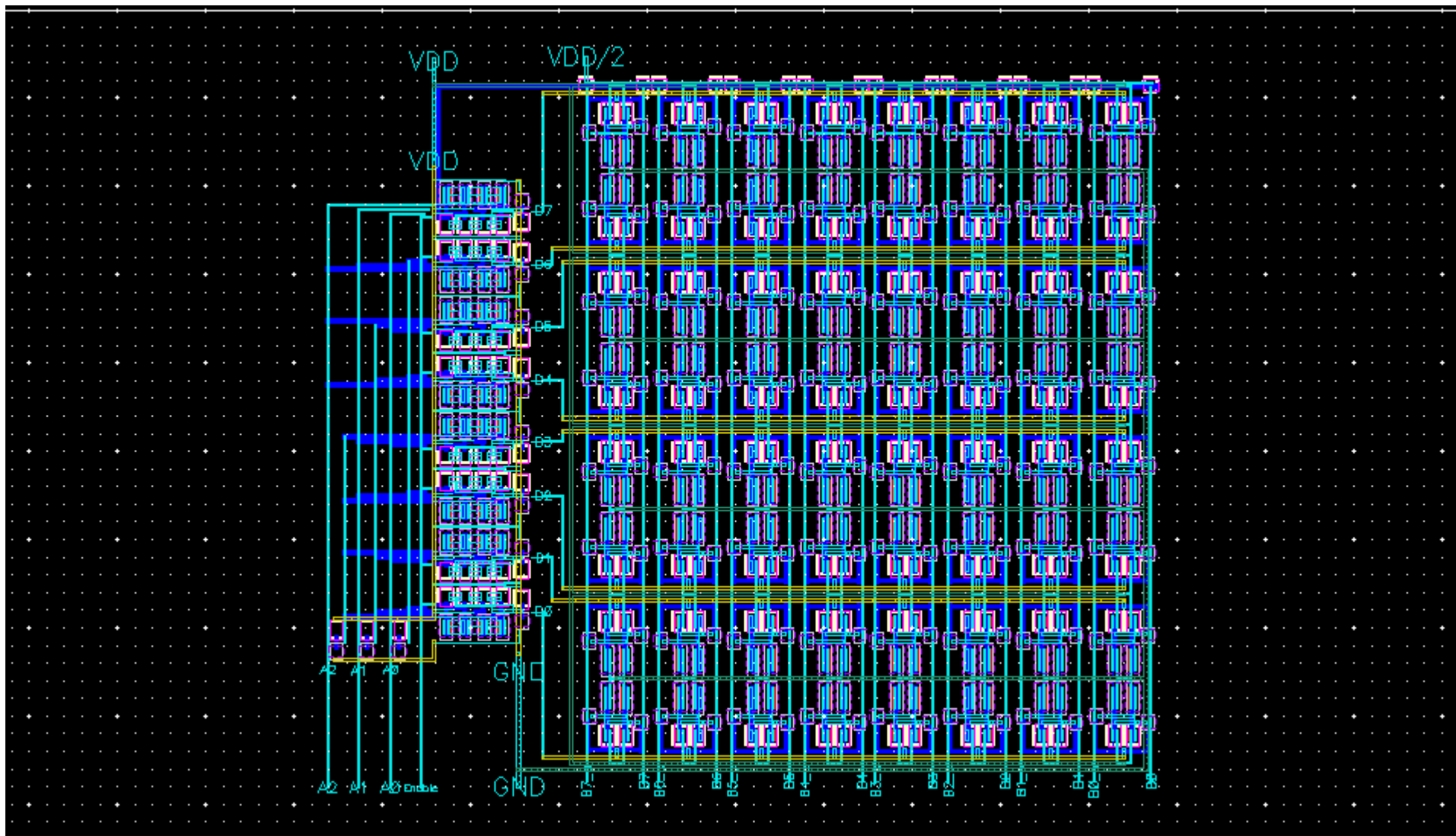


# Accessing SRAM

- The SRAM cells are organized into rows, with a whole row accessed at once.
  - For example, a memory architecture with an 8-bit address and 32-bit data is shown below.
- The address decoder uses the address to select a single row, and all its data are read out.

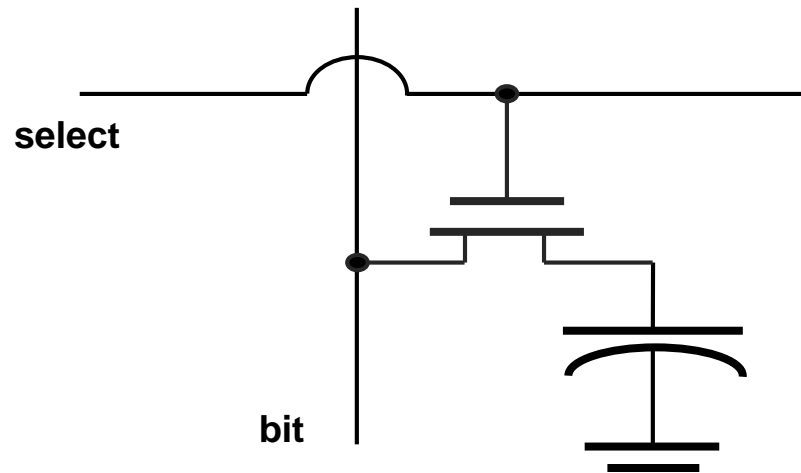


# SRAM Layout



# DRAM

- A DRAM cell is typically made up of three or even one transistor.
  - A single bit is stored in one capacitor.
  - Access to the bit is controlled by a single access transistor, which is gated by the word line (select).
  - As in SRAM, data are read in and out through the bit line.

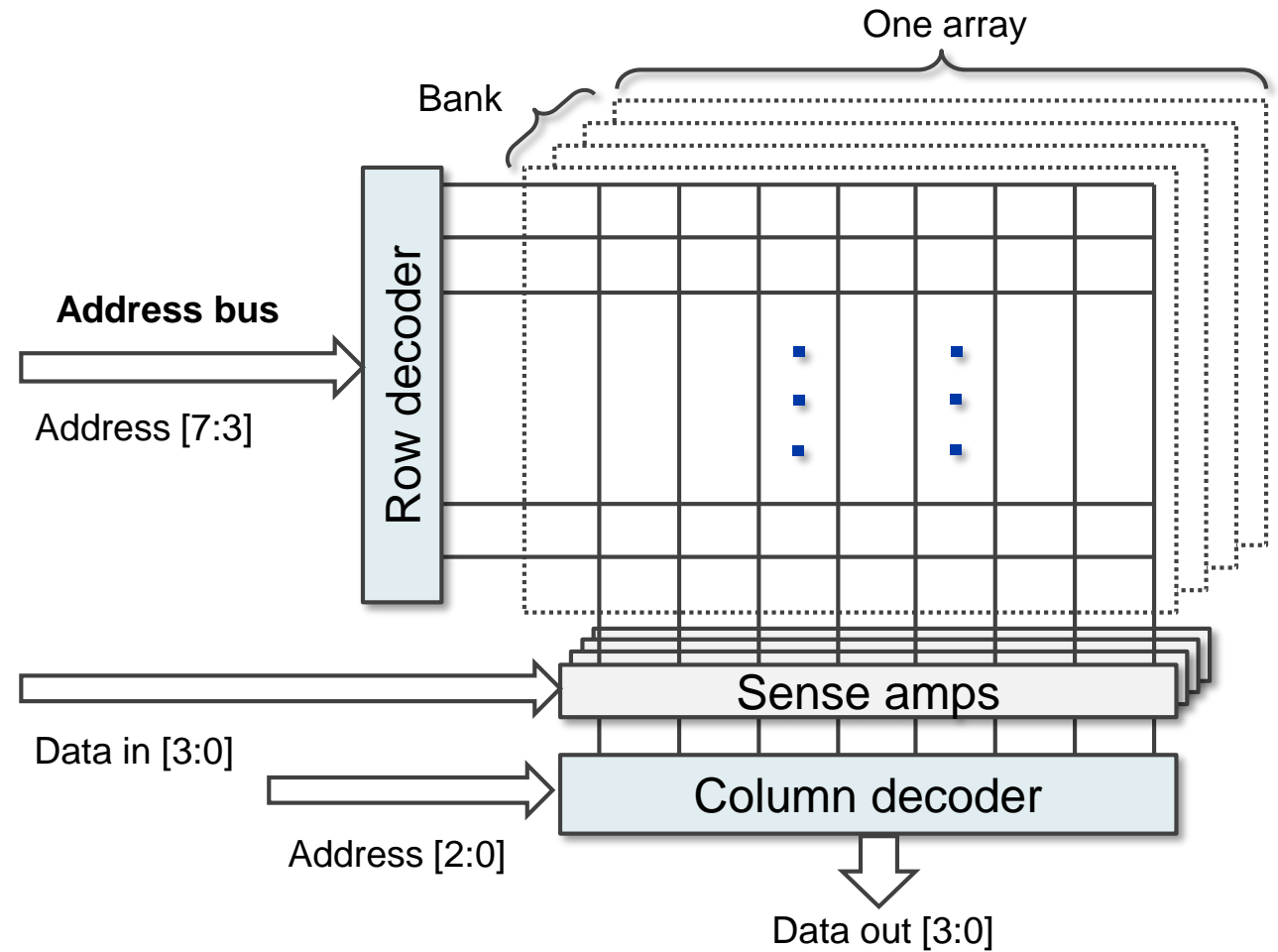


# DRAM

- The status of the capacitor (charged or uncharged) indicates the bit state (1 or 0).
- Access is similar to SRAM but.
  - The capacitor is drained on a read and charged (if storing 1) on a write.
  - The cell needs to be refreshed (or recharged) periodically since the capacitor leaks its charge.
    - For example, every 7.8 ms
- DRAM is higher density than SRAM.
  - Therefore less expensive
- DRAM can be categorized according to its synchronization and data rate.
  - Most DRAM is now synchronous (SDRAM), so it has a clock, rather than asynchronous.
  - Double data rate (DDR) DRAM transfers data on both the rising and falling clock edges.

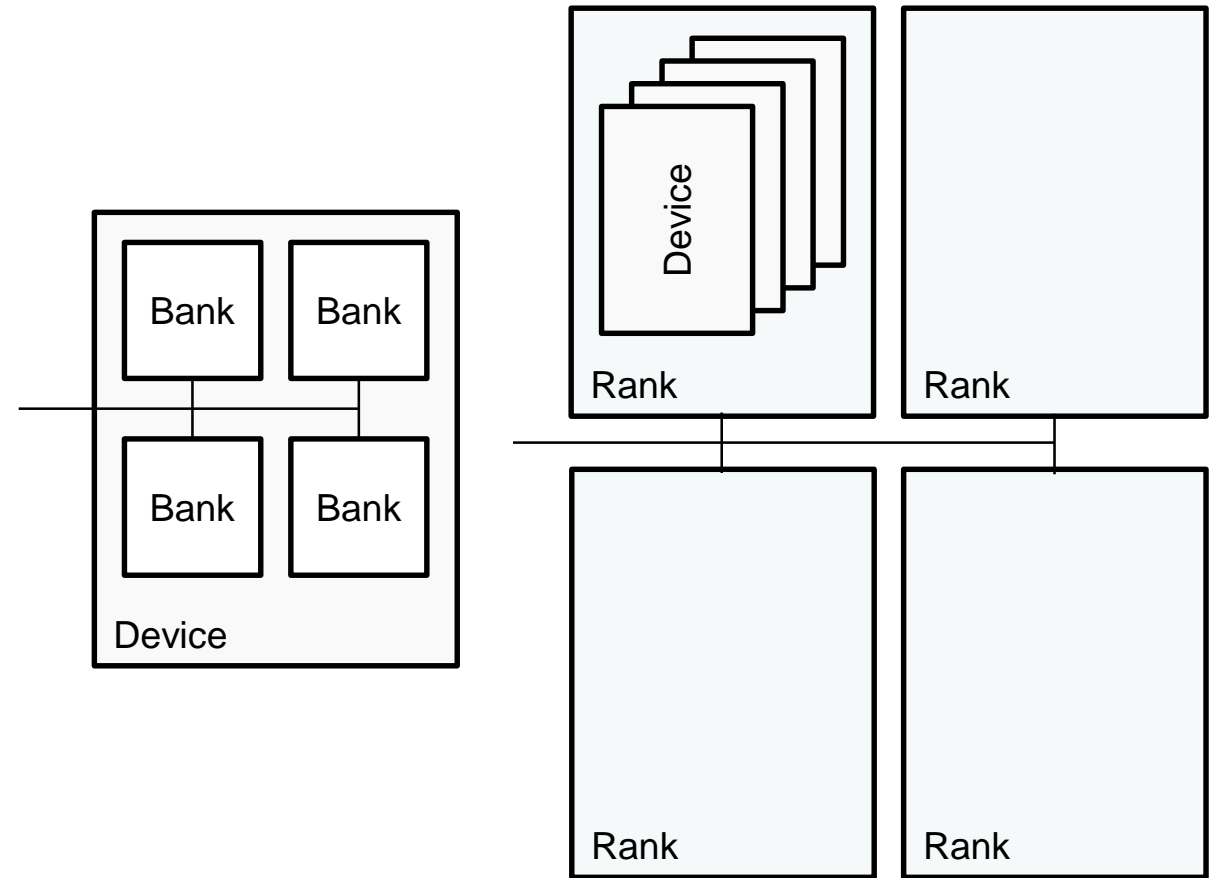
# DRAM Organization

- DRAM cells are organized into arrays.
  - A whole row is accessed at once.
  - But only one bit is read out of the row.
- Multiple arrays are grouped into banks.
  - All arrays in a bank are accessed simultaneously.
  - A bank with N arrays provides N bits per access.



# DRAM Organization

- DRAM banks are grouped into devices.
- Each device bank operates independently.
  - This allows multiple accesses to occur concurrently.
- Devices may be grouped into ranks.
  - All devices in a rank are accessed together.
  - This provides bandwidth.

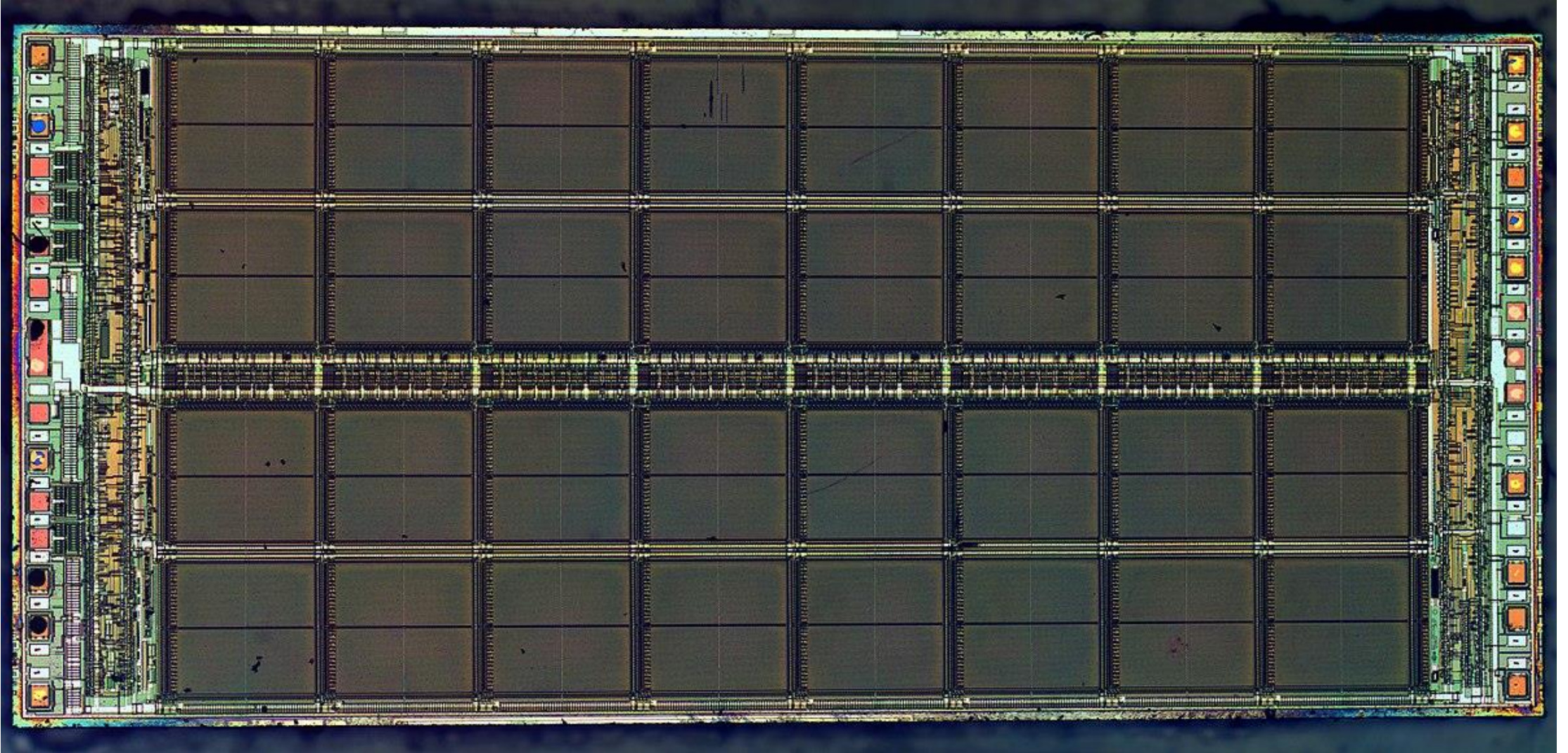


# DRAM Operation

- Three steps in read/write access to a given bank
  - Row access (RAS)
    - decode row address, enable addressed row (often multiple Kb in row)
  - Column access (CAS)
    - decode column address to select small number of sense amplifier latches (4, 8, 16, or 32 bits depending on DRAM package)
  - Precharge
    - charges bit lines to known value, required before next row access
- Each step has a latency of around 15-20ns in modern DRAMs
- Various DRAM standards (DDR, RDRAM) have different ways of encoding the signals for transmission to the DRAM, but all share same core architecture



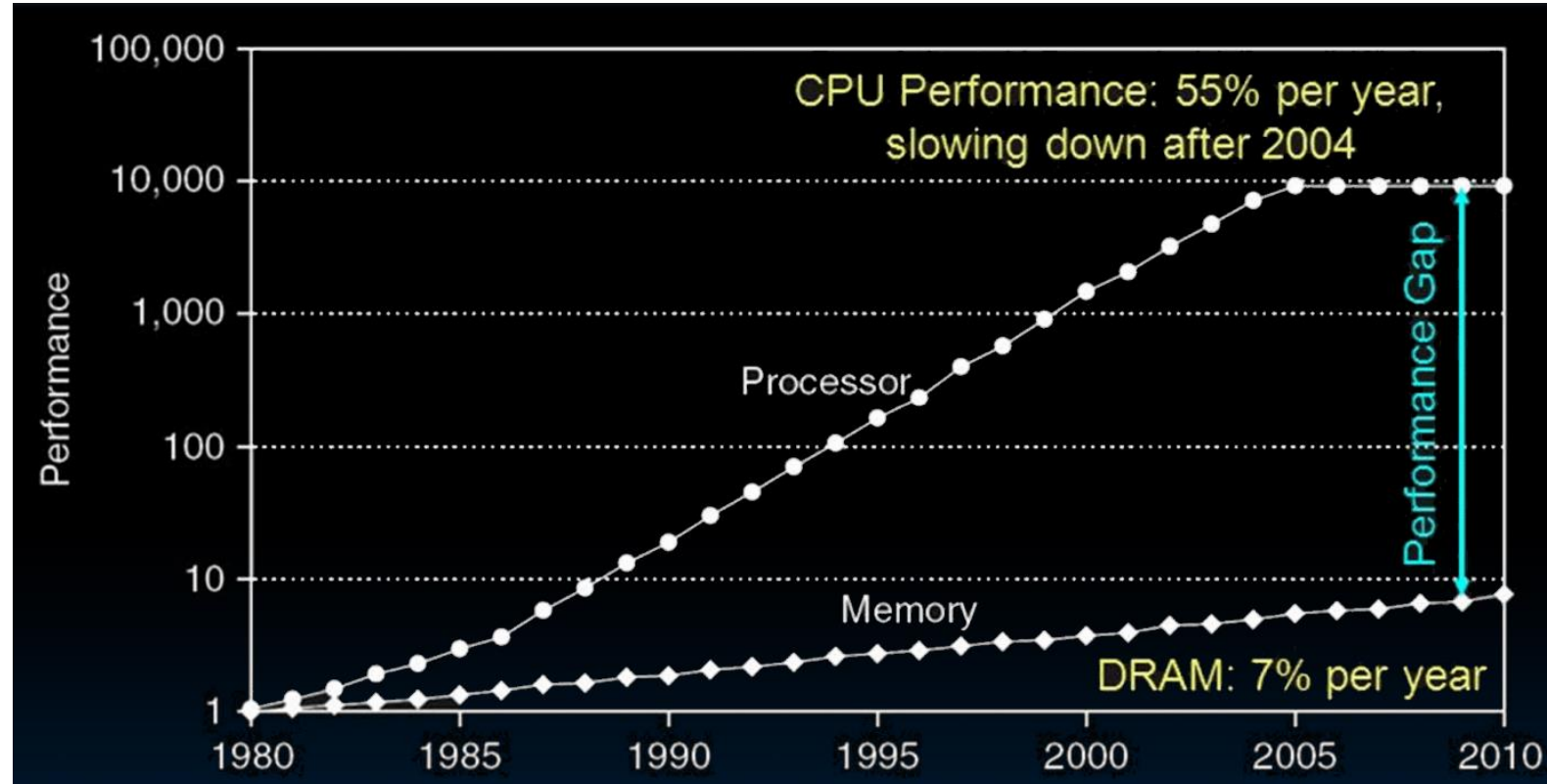
# DRAM Layout



source: Wiki



# Processor-DRAM Gap (Memory Wall)



- 1980 microprocessor executes ~one instruction in same time as DRAM access
- 2020 microprocessor executes ~1000 instructions in same time as DRAM access
- Slow DRAM access has disastrous impact on CPU performance!

Image: <https://inst.eecs.berkeley.edu/~cs61c>

# Memory Wall

- Originally theorized in 1994 by Wulf and McKee (U of Virginia)
- Processors are always waiting on memory, and CPU performance is therefore entirely limited by memory performance.

## Hitting the Memory Wall: Implications of the Obvious

Wm. A. Wulf  
Sally A. McKee

Department of Computer Science  
University of Virginia  
{wulf|mckee}@virginia.edu

December 1994

This brief note points out something obvious — something the authors “knew” without really understanding. With apologies to those who did understand, we offer it to those others who, like us, missed the point.

We all know that the rate of improvement in microprocessor speed exceeds the rate of improvement in DRAM memory speed — each is improving exponentially, but the exponent for microprocessors is substantially larger than that for DRAMs. The difference between diverging exponentials also grows exponentially; so, although the disparity between processor and memory speed is already an issue, downstream someplace it will be a much bigger one. How big and how soon? The answers to these questions are what the authors had failed to appreciate.

To get a handle on the answers, consider an old friend — the equation for the average time to access memory, where  $t_c$  and  $t_m$  are the cache and DRAM access times and  $p$  is the probability of a cache hit:

$$t_{avg} = p \times t_c + (1 - p) \times t_m$$

*"We all know that the rate of improvement in microprocessor speed exceeds the rate of improvement in DRAM memory speed — each is improving exponentially, but the exponent for microprocessors is substantially larger than that for DRAMs.*

*The difference between diverging exponentials also grows exponentially; so, although the disparity between processor and memory speed is already an issue, downstream someplace it will be a much bigger one."*

# Techniques for Improving DRAM Performance

## Row buffer

- To buffer recently accessed data without having to make another access
- Read and refresh several words and in parallel.
- The row buffer is essentially the sense amps at the bottom of each array.

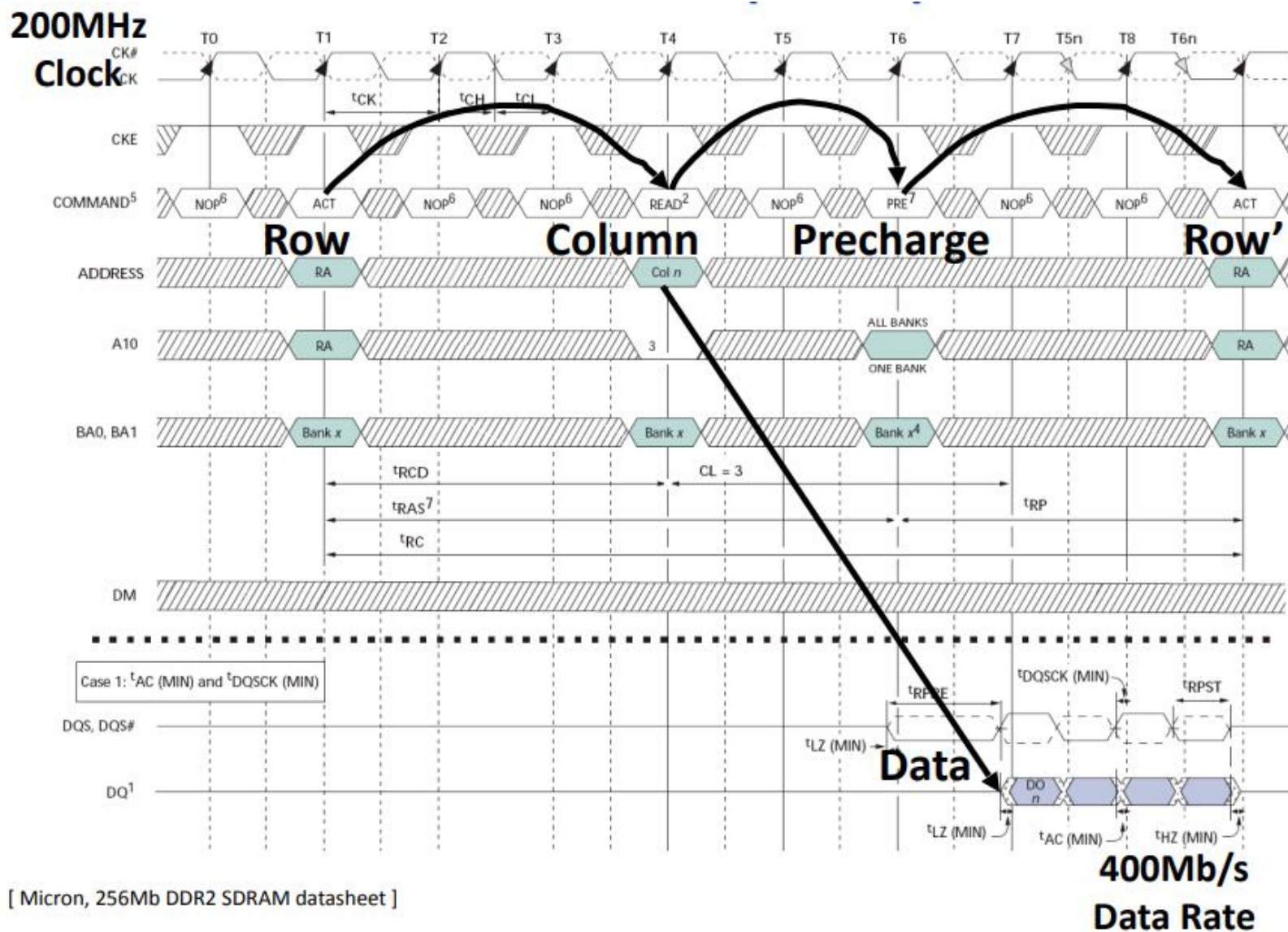
## Double data rate (DDR)

- Transfer data on the rising and falling clock edges to double the bandwidth.

## DRAM banking

- Increase the number of parallel banks to improve bandwidth with simultaneous accesses.

# Double-Data Rate (DDR2) DRAM



[ Micron, 256Mb DDR2 SDRAM datasheet ]

# DDR1 – DDR5

DDR Version	DDR1	DDR2	DDR3	DDR 4	DDR5
Released date	2000	2003	2007	2012	Under Progress
Operating voltage	2.5V	1.8V	1.5V	1.2V	1.1V
Prefetch buffer size	2	4	8	8	16
Chip densities	128Mb-1Gb	128Mb-4Gb	512Mb-8Gb	2Gb-16Gb	8Gb-64Gb
Data rate (MT/s)	200-400	400-800	800-2133	1600-3200	3200-6400
Bank groups	0	0	0	4	8
Termination/ODT	$\Omega$ on board	ODT added	Nominal, Dynamic Modes	Park Modes	Nominal Wr/Rd

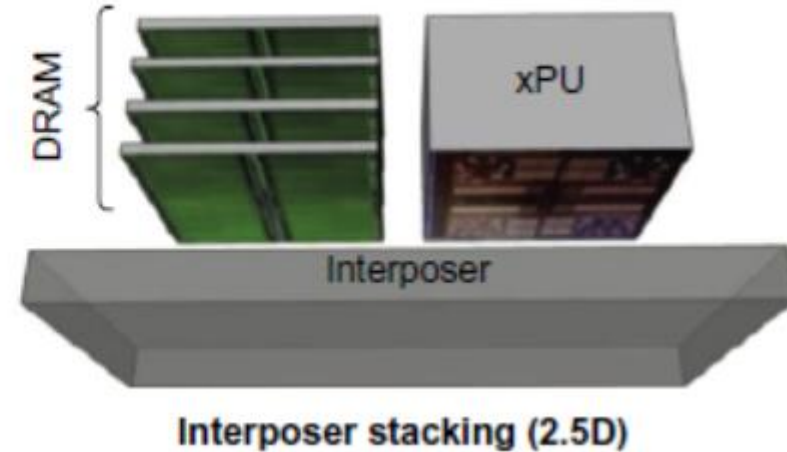
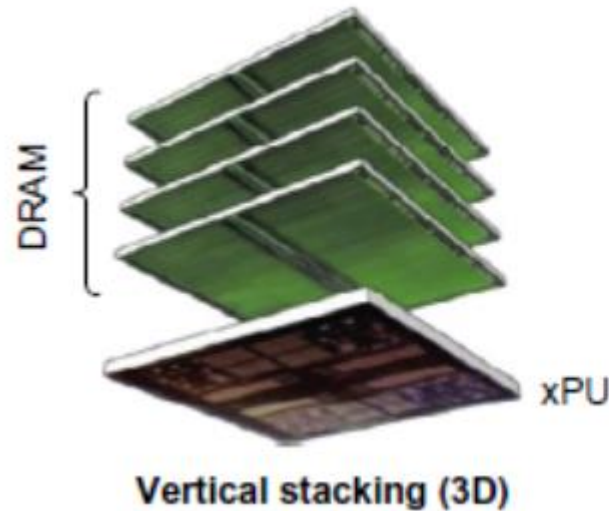
Comparison of DDR Generations

source: Synopsys



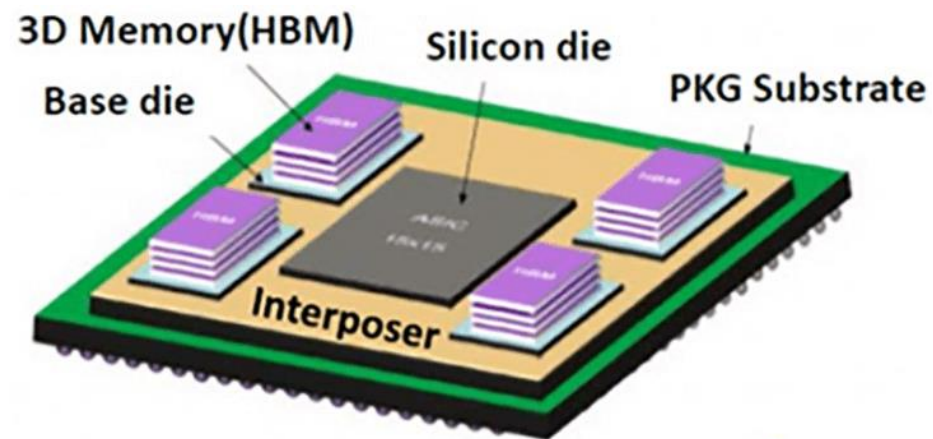
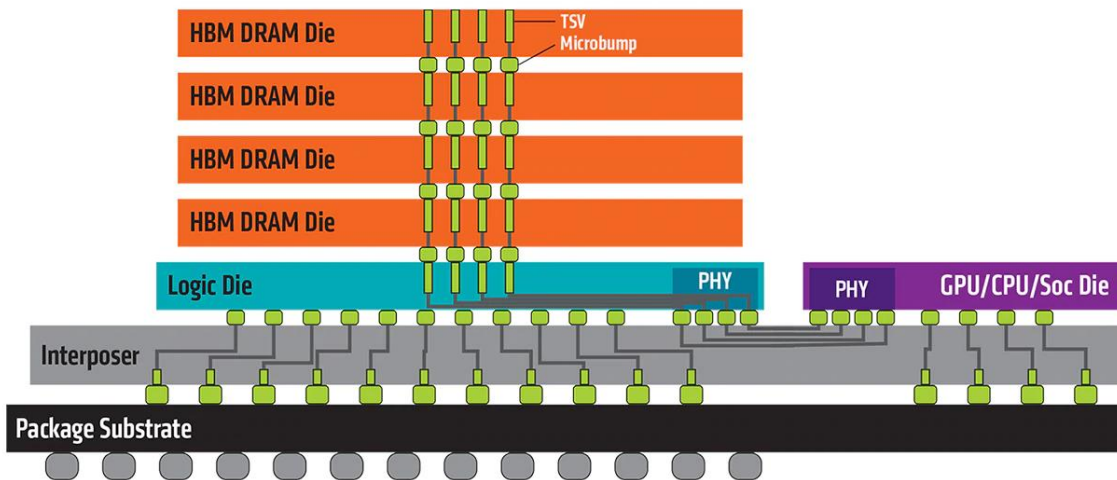
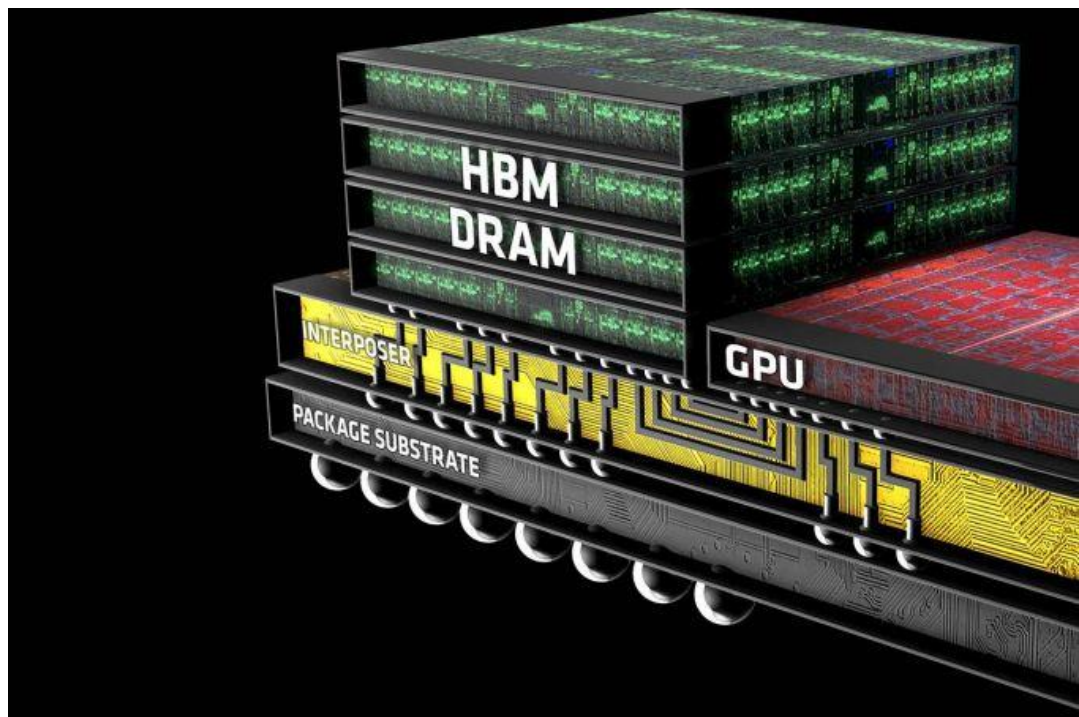
# Stacked/Embedded DRAMs

- Stacked DRAMs in same package as processor
  - High Bandwidth Memory (HBM)





# AMD HBM



Source: AMD

# DRAM Packaging, Apple M1



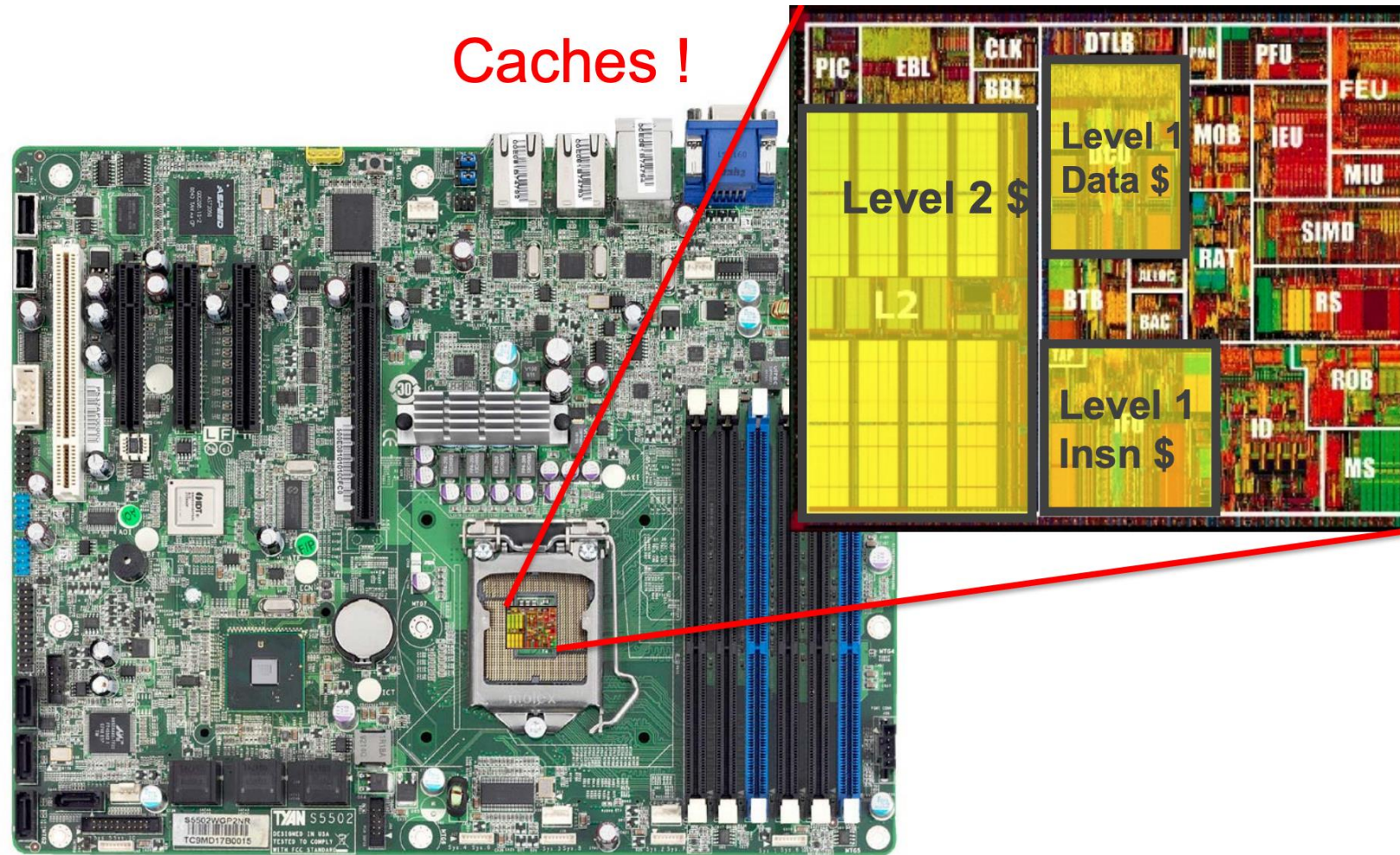
Two DRAM chips  
on same package  
as system SoC

- 128b databus,  
running at 4.2Gb/s
- 68GB/s bandwidth

source: Apple



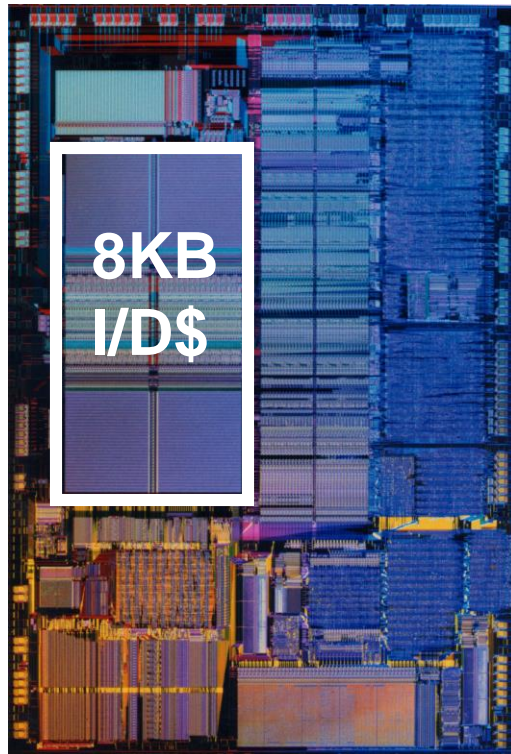
# We still need fast on-chip memories!



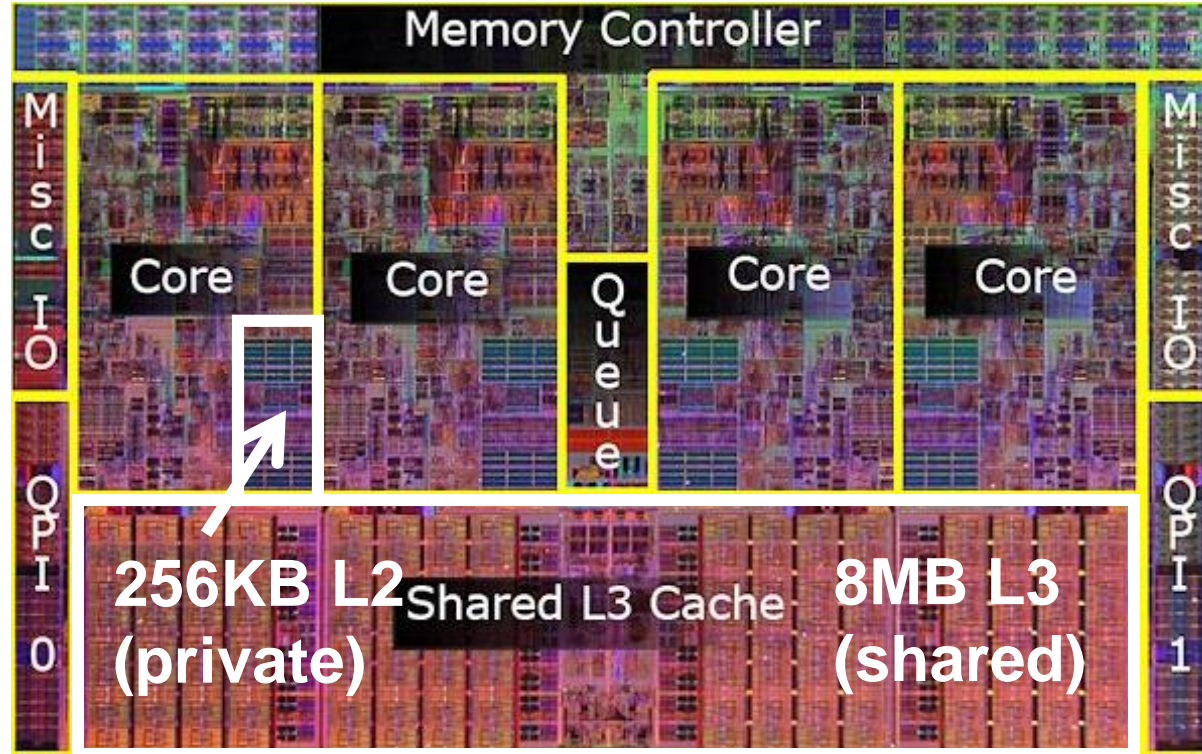
Intel Pentium 3, 1999



# Evolution of Cache Hierarchies



Intel 486



Intel Core i7 (quad core)

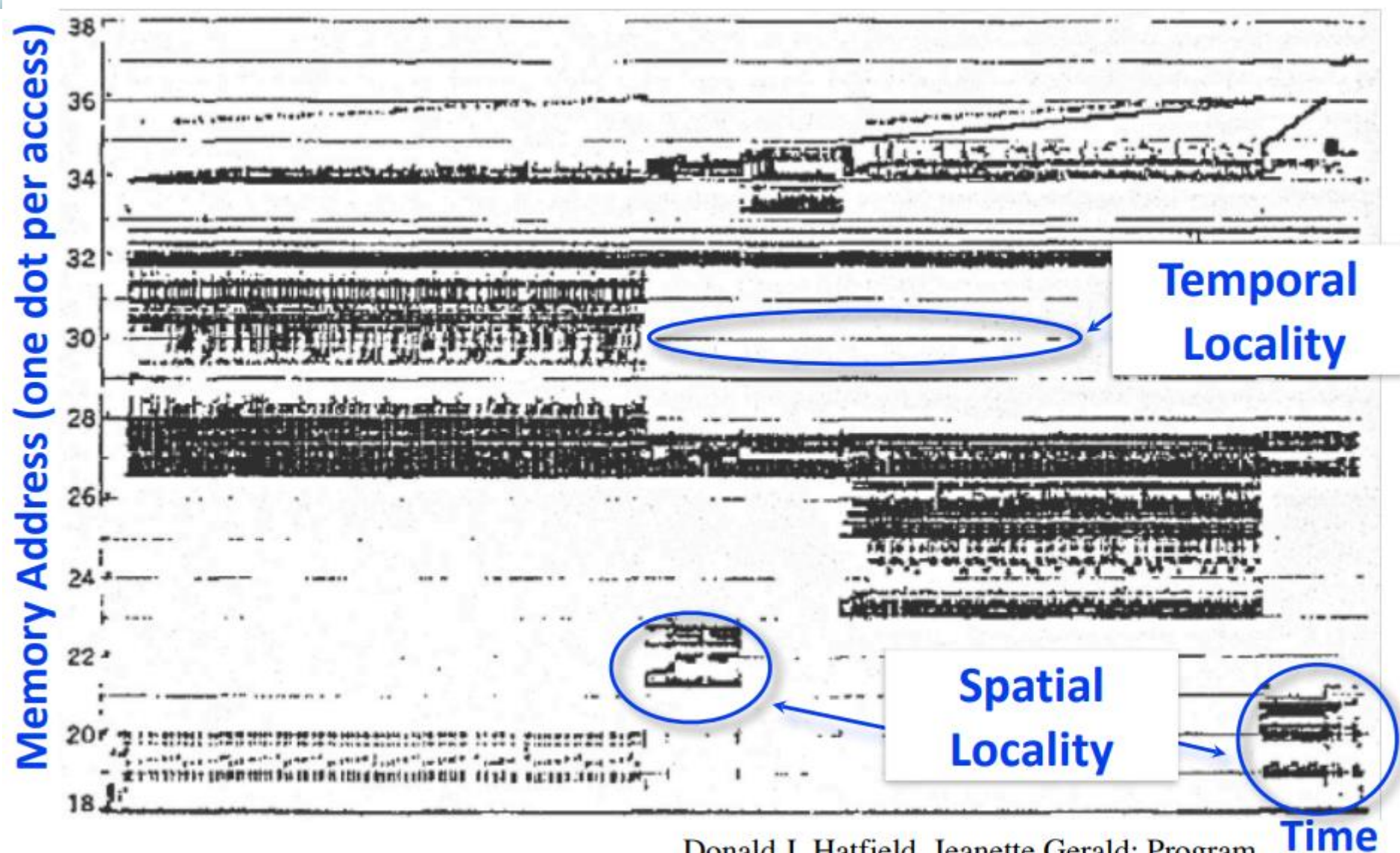
- Chips today are 30–70% cache by area

# Cache motivation

- Fortunately, most programs don't need access to all memory all of the time.
  - Accesses tend to exhibit locality of reference.
  - Temporal locality – if an address is accessed, it is likely to be accessed again soon.
  - Spatial locality – if an address is accessed, its neighbors are likely to be accessed soon.
  - Therefore, only a small number of addresses are likely to be accessed in the near future.
- Small memories are quick to access and can be placed near to the CPU.
  - If we can identify these locations likely to be accessed soon, then we can keep them in these memories.
- A cache stores copies of some memory locations for fast access when required.



# Memory Reference Patterns



Donald J. Hatfield, Jeanette Gerald: Program Restructuring for Virtual Memory. IBM Systems Journal

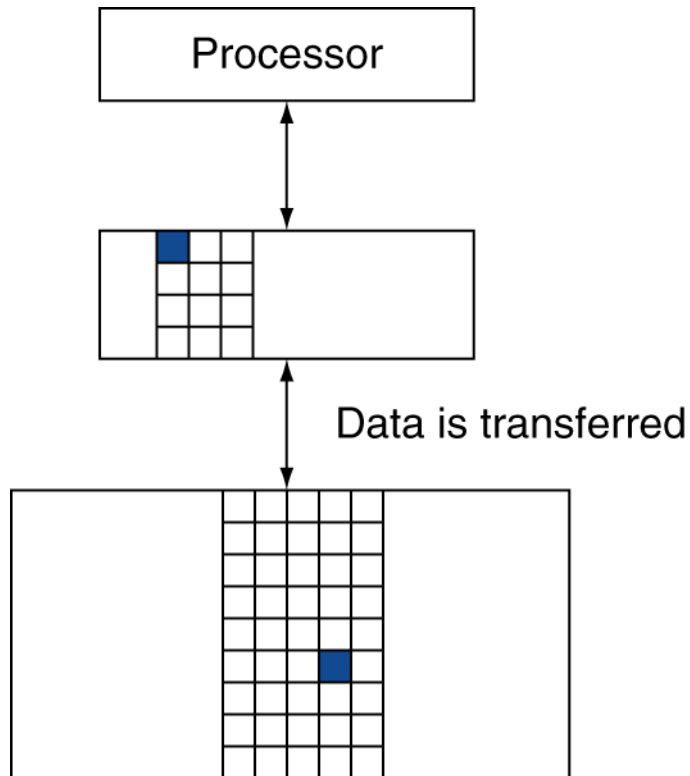
# Principle of Memory Access

- Taking advantage of Memory Locality
- 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



# Review: A few terminology

- **Block** (aka line): unit of copying
  - Minimum unit of info that is present/or not in the cache
  - May be multiple words



# Review: Block, Word, Byte Addresses

- Byte address vs. word address (assume 16-bit address)

- Example: one word has 4 ( $2^2$ ) bytes

Word number (word address)

Byte offset

Byte Address	Contents
1111_1111_0000_00 00	Byte 1
1111_1111_0000_00 01	Byte 2
1111_1111_0000_00 10	Byte 3
1111_1111_0000_00 11	Byte 4

14 bits 2 bits

- Word address vs. block address (block number)

- Example: one block has 2 words (8 bytes)

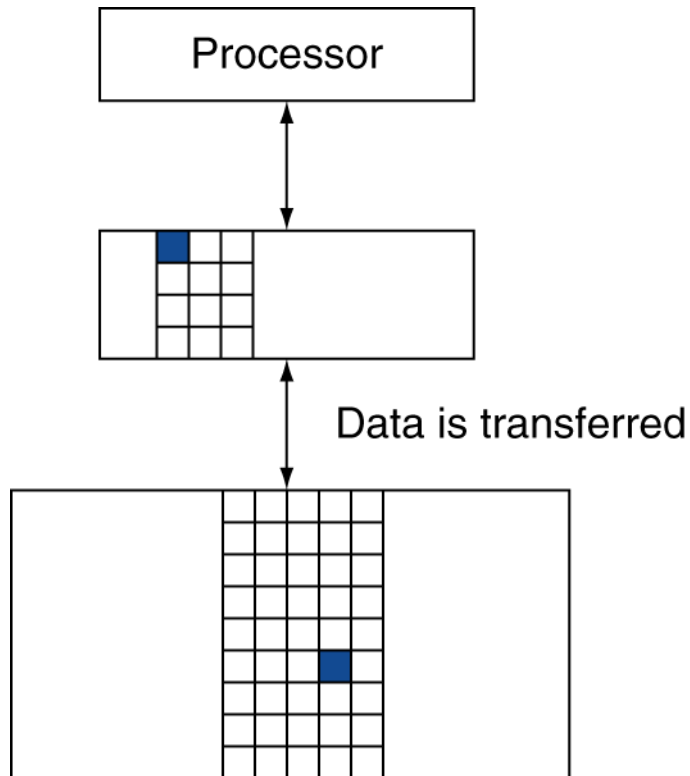
Block number (Block address)

Word offset

Word Number	Contents
1111_1111_0000_0 0	Word 1
1111_1111_0000_0 1	Word 2

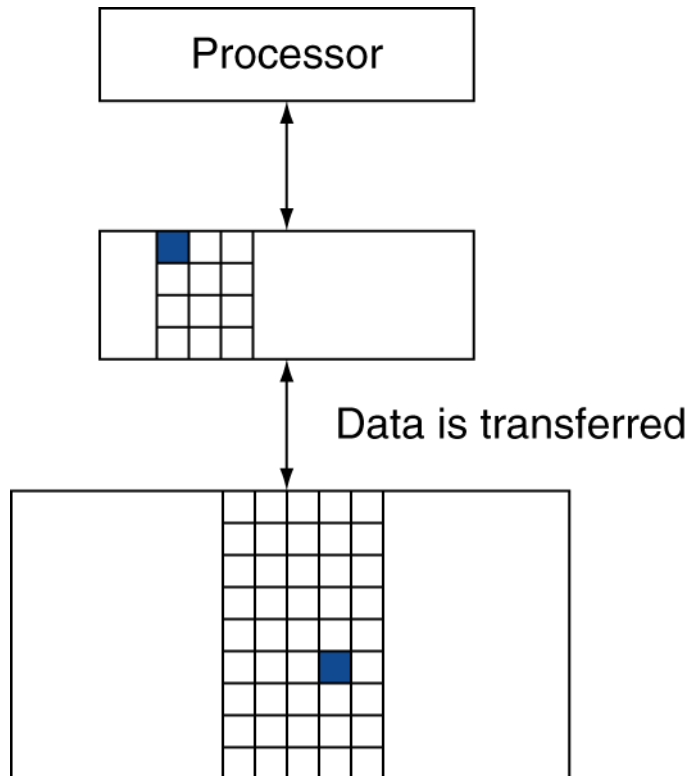
13 bits 1 bit

# Review: A few terminology



- If accessed data is present in upper level
  - **Hit**: access satisfied by upper level
    - **Hit ratio: hits/accesses**
- Hit time: time to access a memory including
  - Time to determine whether a hit or miss
  - Time to pass block to requestor

# Review: Access the Memory Hierarchy



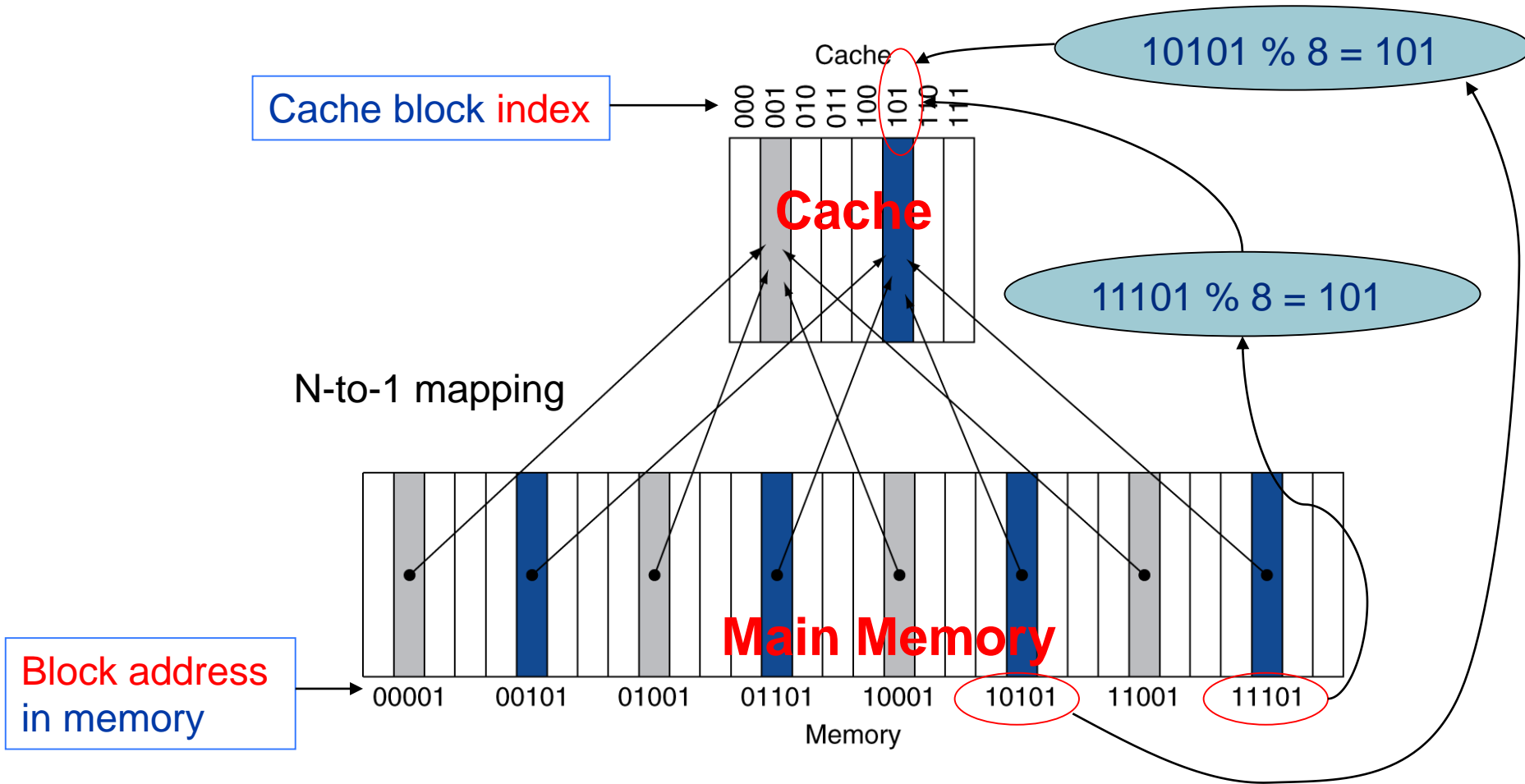
- Concepts:
  - Miss
    - If accessed data is absent
    - Block copied from lower to higher level
    - Then accessed data supplied from upper level
    - Time taken: miss penalty
    - Miss rate:  $\text{misses/accesses} = 1 - \text{hit rate}$
- **Cache controller:** finite state machine
  - Remembers miss address
  - Accesses next level of memory
  - Waits for response
  - Writes data/tag into proper locations

# Review: Direct-mapped Cache

- A simple design because each memory location only maps to one cache block
- However, this often leads to contention.
  - When several locations with the same cache index are repeatedly accessed
- Pros:
  - Simple design, therefore inexpensive
  - Quick to search
- Cons:
  - Low hit rate when there is contention

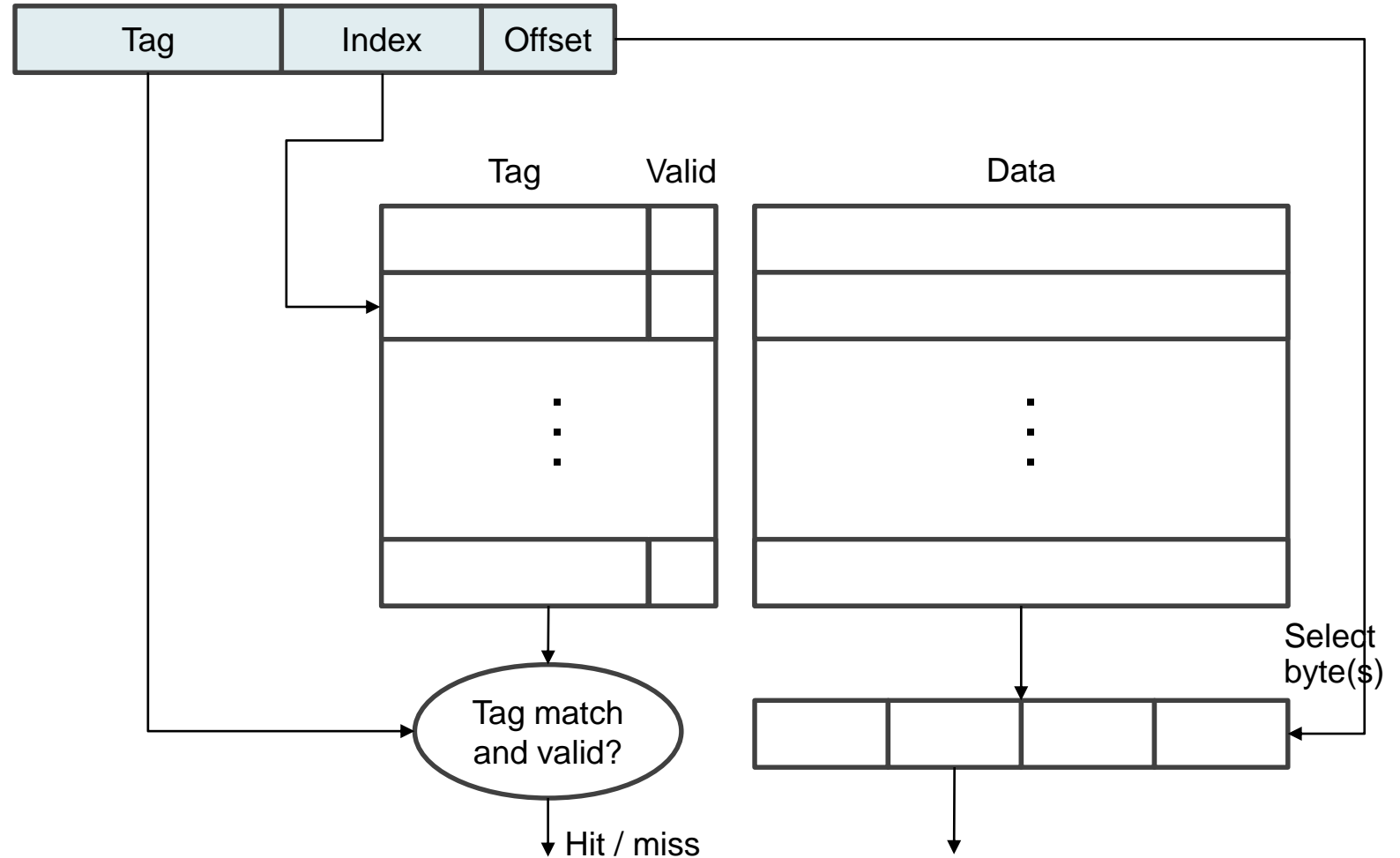
# Direct Mapped Cache

- Number of blocks in a cache is a power of **2**
- Low-order bits of block address in memory = cache index



# Direct-mapped Cache

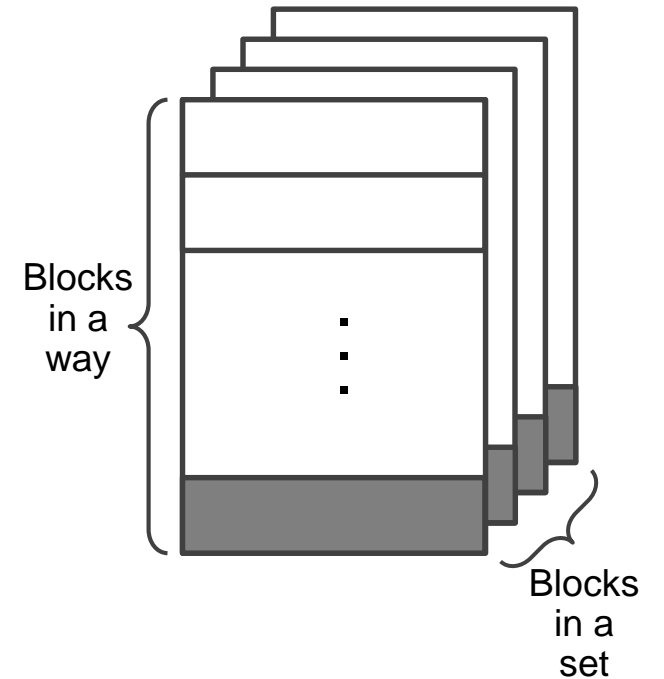
- Index used to select a single cache block
- Tags compared
  - If valid and tags match, then hit
- On hit, offset chooses starting byte from data array.





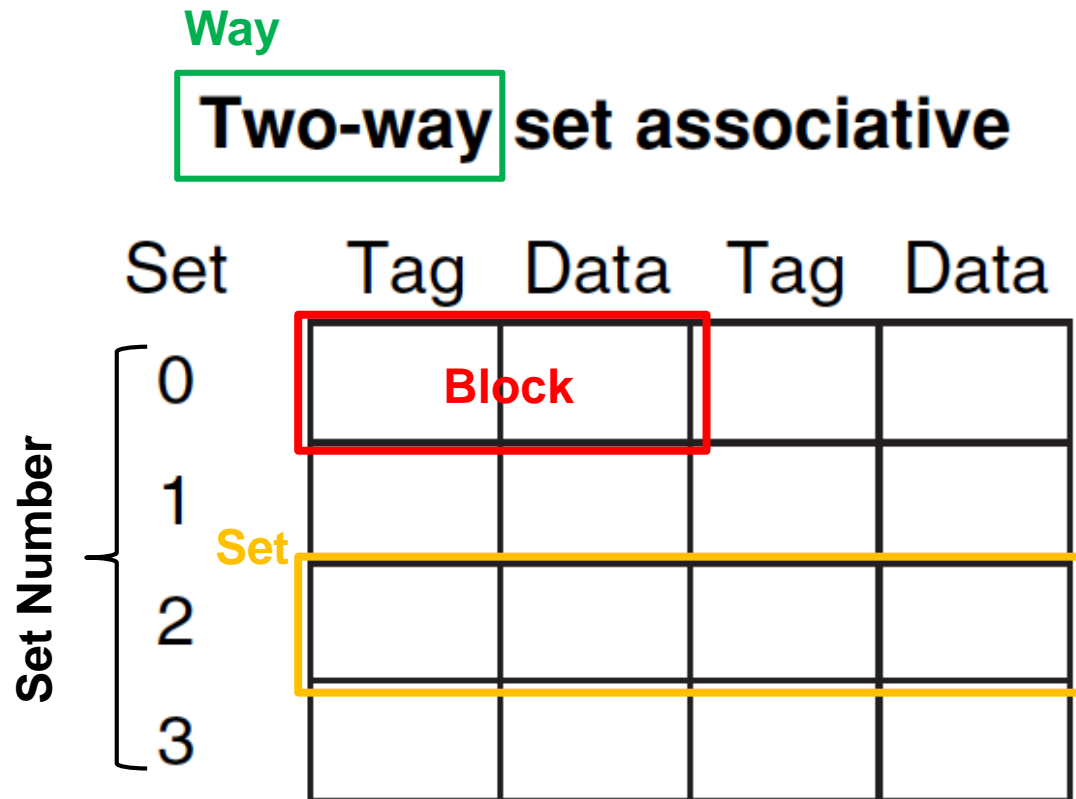
# Set-associative Cache

- Each memory location maps to N cache blocks.
  - Each group of N cache blocks is called a set – hence, N-way set associative. (Next page)
  - A group of cache blocks in the same array but with different indices is called a way.
- Improved hit rate compared to a direct-mapped cache
  - Less contention when there are several memory locations with the same cache index
- Pros:
  - Combines the speed of a direct-mapped cache with improved flexibility in placement
- Cons:
  - Finding blocks within a set requires more complex hardware.



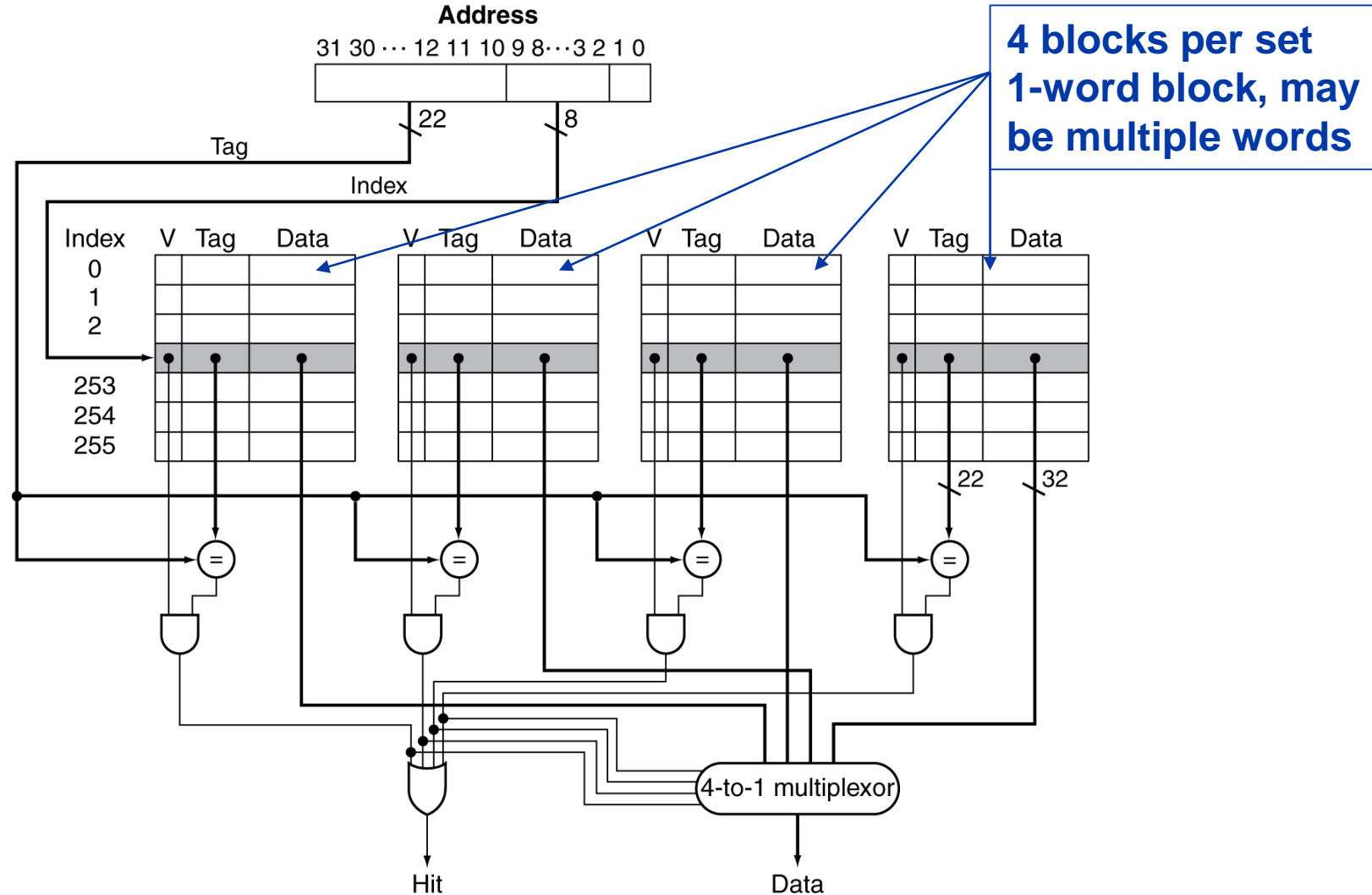
# Terminology

- Block
- Set
- Way
- # of Set



Each block can have multiple words.

# Set Associative Cache Organization



# Fully Associative Cache

- In a fully associative cache, a block can be placed in any available location in the cache.
- This makes the cache more flexible, increasing the hit rate.
- But it is also more complex, as searching for a block involves comparing against all blocks.
- Pros:
  - Good flexibility: greater hit rate
- Cons:
  - Searching for a match can be expensive, power-hungry, and slow.

# Memory Performance

## Memory

- Memory cycle time
  - Minimum delay between two memory accesses
- Memory access time
  - Elapsed time between the start and finish of one memory access

## Bandwidth

- The data throughput
  - Bit rate \* number of bits

## Cache

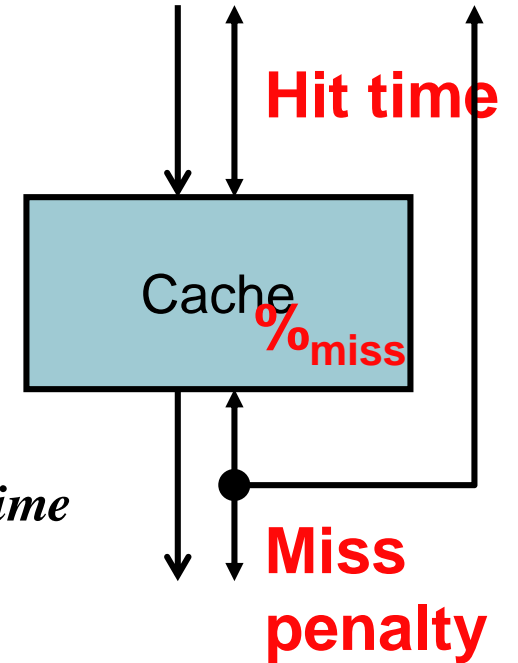
- Cache hit time
  - Elapsed time between starting access to the cache and retrieving the data
- Cache miss penalty
  - Time required to retrieve data from memory on a cache miss

# Review: Cache performance

- $CPI_{ALUOps}$  does not include memory instructions
- $AMAT$  = Average Memory Access Time

$$AMAT = HitTime + MissRate \times MissPenalty$$
$$= (HitTime_{Inst} + MissRate_{Inst} \times MissPenalty_{Inst}) + (HitTime_{Data} + MissRate_{Data} \times MissPenalty_{Data})$$

$$CPUtime = IC \times \left( \frac{AluOps}{Inst} \times CPI_{AluOps} + \frac{MemAccess}{Inst} \times AMAT \right) \times CycleTime$$

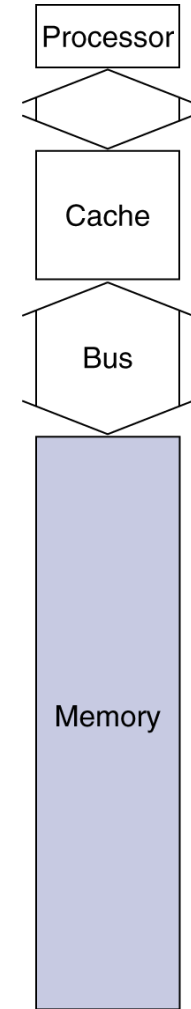


- For example:
  - An L1 cache with 1 ns hit latency and 5% miss rate
  - Combined with an L2 cache with a 10 ns hit latency and 1% miss rate
  - And 100 ns main memory latency

$$AMAT = 1 + 0.05 * (10 + 0.01 * 100) = 6.5 \text{ ns}$$

# Reducing Miss Penalty by Main Memory organization

- Use DRAMs for main memory
  - Fixed width (e.g., 1 word)
  - Connected by fixed-width 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 =  $1 + 4 \times 15 + 4 \times 1 = 65$  bus cycles
  - Bandwidth = 16 bytes / 65 cycles = 0.25 B/cycle

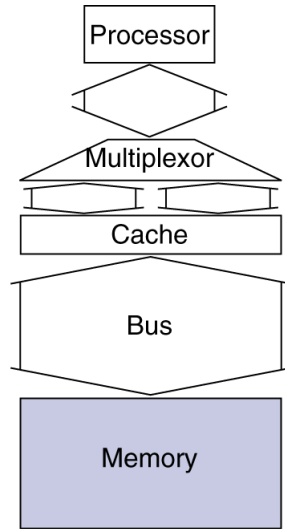


a. One-word-wide  
memory organization



# Reducing Miss Penalty

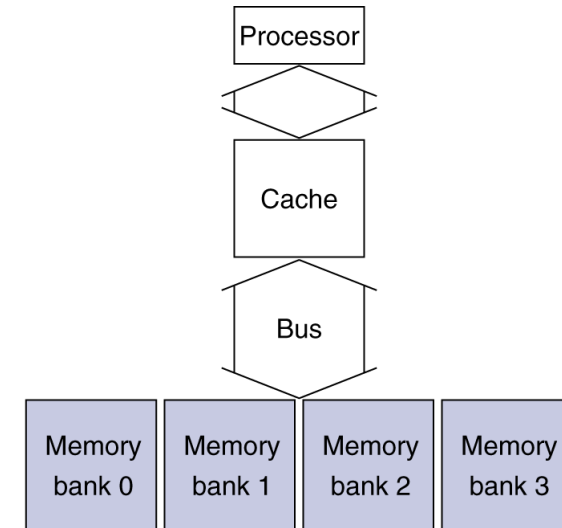
by Increasing Memory Bandwidth



b. Wider memory organization

- 2-word wide memory
  - Miss penalty =  $1 + 2 \times 15 + 2 \times 1 = 33$  bus cycles
  - Bandwidth =  $16 \text{ bytes} / 33 \text{ cycles} = 0.48 \text{ B/cycle}$

- 4-bank *interleaved memory*
  - Miss penalty =  $1 + 15 + 4 \times 1 = 20$  bus cycles
  - Bandwidth =  $16 \text{ bytes} / 20 \text{ cycles} = 0.8 \text{ B/cycle}$



c. Interleaved memory organization

# Review: “Three Cs” Model of Misses

- 1<sup>st</sup> C - Compulsory (cold):
  - occur when a program is first started
  - cache does not contain any of that program’s data yet, so misses are bound to occur
  - can’t be avoided easily
  - Every block of memory will have one compulsory miss (NOT only every block of the cache)

# Review: “Three Cs” Model of Misses

- 2<sup>nd</sup> C - Conflict:
  - miss that occurs because two distinct memory addresses map to the same cache location
  - two blocks (which happen to map to the same location) can keep overwriting each other
  - big problem in direct-mapped caches
  - how do we lessen the effect of these?
    - Set associativity

# Review: “Three Cs” Model of Misses

- 3<sup>rd</sup> C - Capacity:
  - cache cannot contain all blocks needed for program execution
  - occur when all blocks of cache are filled
  - Main design factor is capacity

# Quick Quiz

■ Does the conflict misses occur in fully associative caches?

A. Yes

B. No

# Effect of Cache Parameters on Performance

## Bigger caches

- **Pros:**
  - Reduces capacity misses
- **Cons:**
  - Increases hit time
  - More expensive
  - Consumes more power

## Larger cache blocks

- **Pros:**
  - Better spatial locality
  - Reduces number of tags
- **Cons:**
  - Increases miss penalty
  - Increases capacity misses
  - Increases conflict misses

## Higher associativity

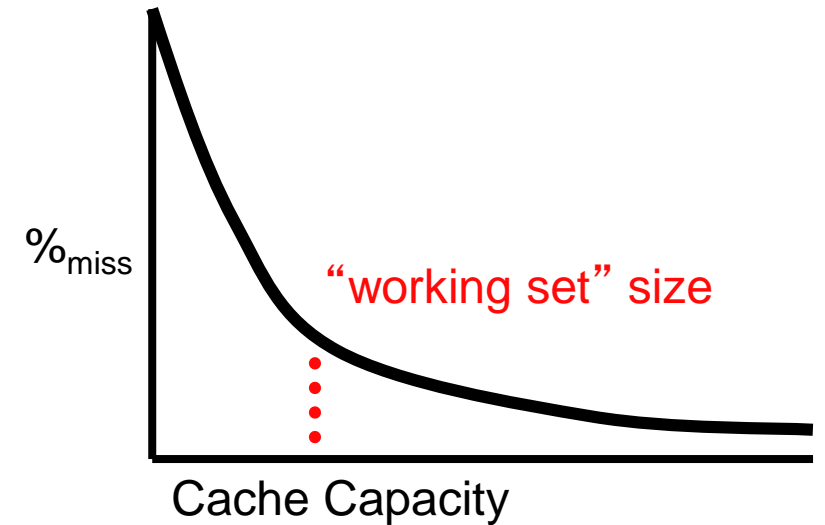
- **Pros:**
  - Reduces conflict misses
- **Cons:**
  - Increases hit time
  - Consumes more power

Miss Rate ABC: **A**ssociativity, **B**lock size, **C**apacity



# Capacity and Performance

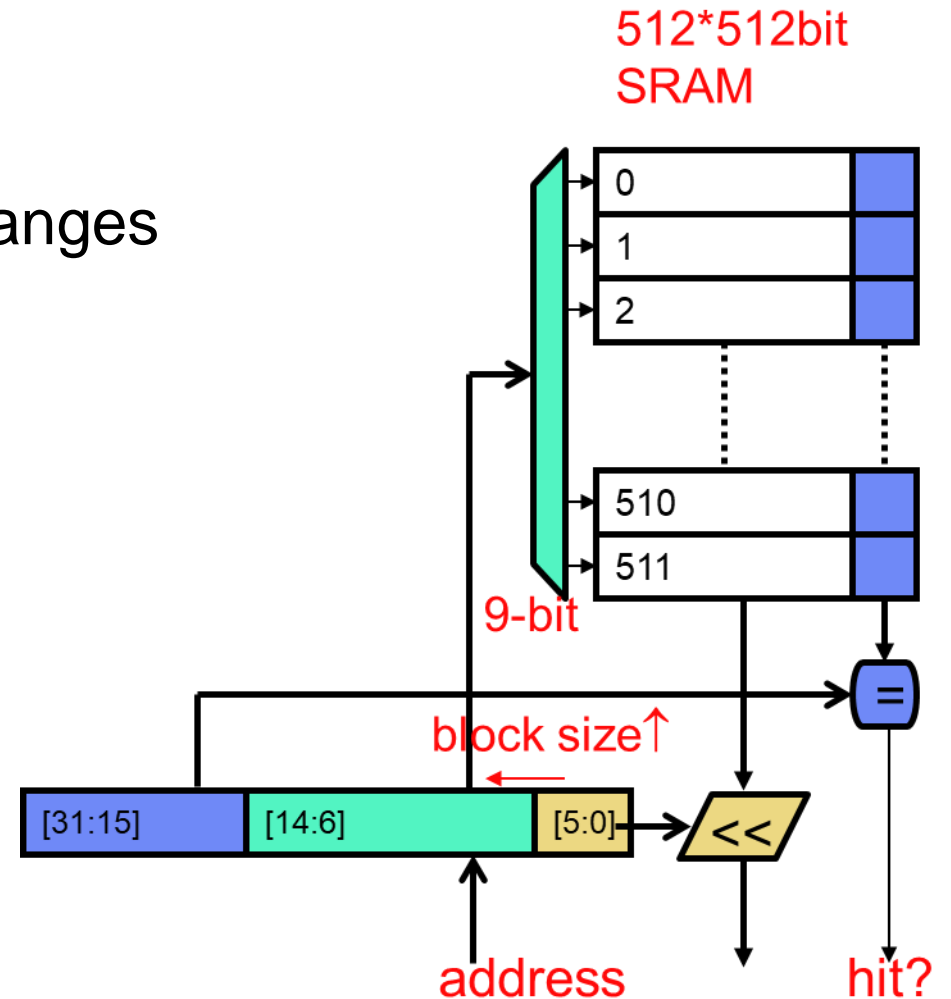
- Simplest way to reduce  $\%_{\text{miss}}$ : increase capacity
  - + Miss rate decreases monotonically
    - “**Working set**”: insns/data program is actively using
    - Diminishing returns
  - However  $t_{\text{access}}$  increases
    - Latency proportional to  $\text{sqrt}(\text{capacity})$



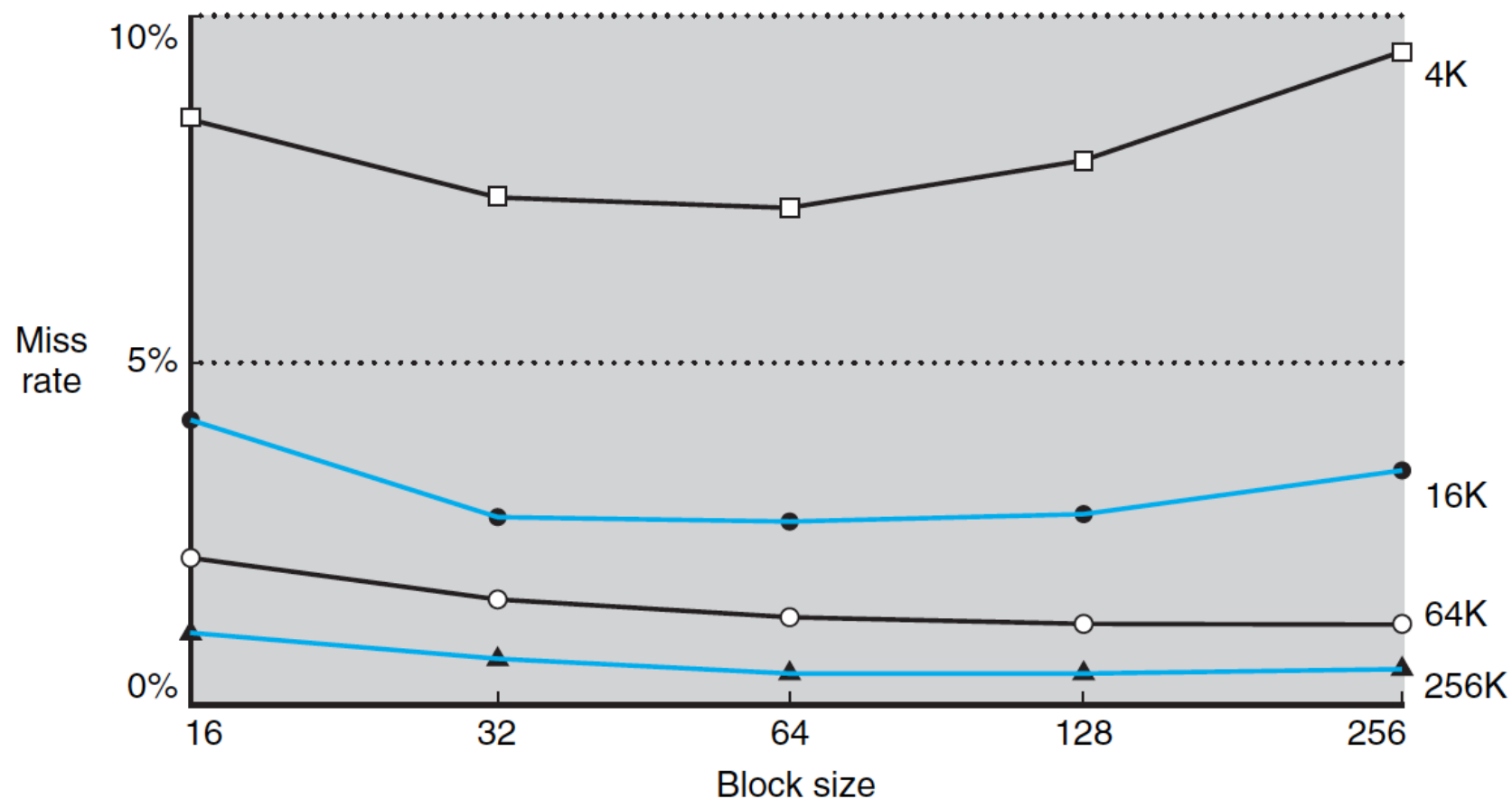
- Given capacity, manipulate  $\%_{\text{miss}}$  by changing **organization**

# Block Size and Performance

- Given capacity, manipulate  $\%_{\text{miss}}$  by changing organization
- One option: increase **block size**
  - Exploit **spatial locality**
  - Boundary between index and offset changes
  - Tag remains the same
- Ramifications
  - + Reduce  $\%_{\text{miss}}$  (up to a point)
  - + Reduce tag overhead (why?)
  - Potentially useless data transfer
  - Premature replacement of useful data



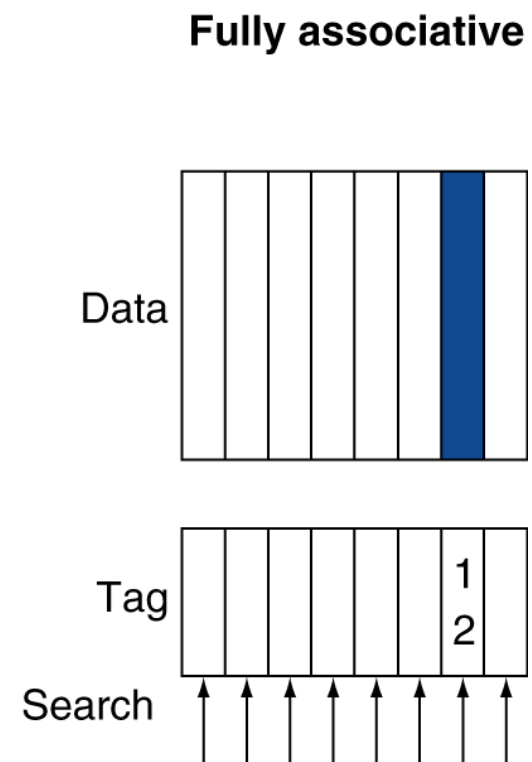
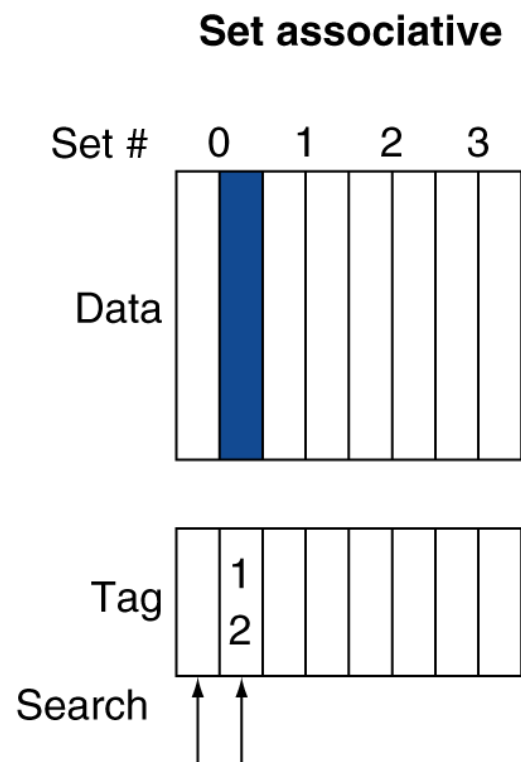
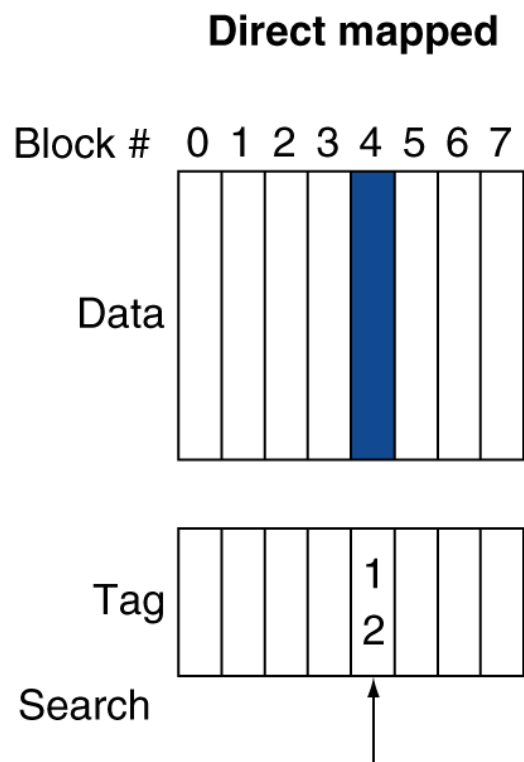
# Block Size and Performance



- The cost of miss (Primarily result of longer time to fetch block) increases
- Larger blocks  $\Rightarrow$  fewer of them

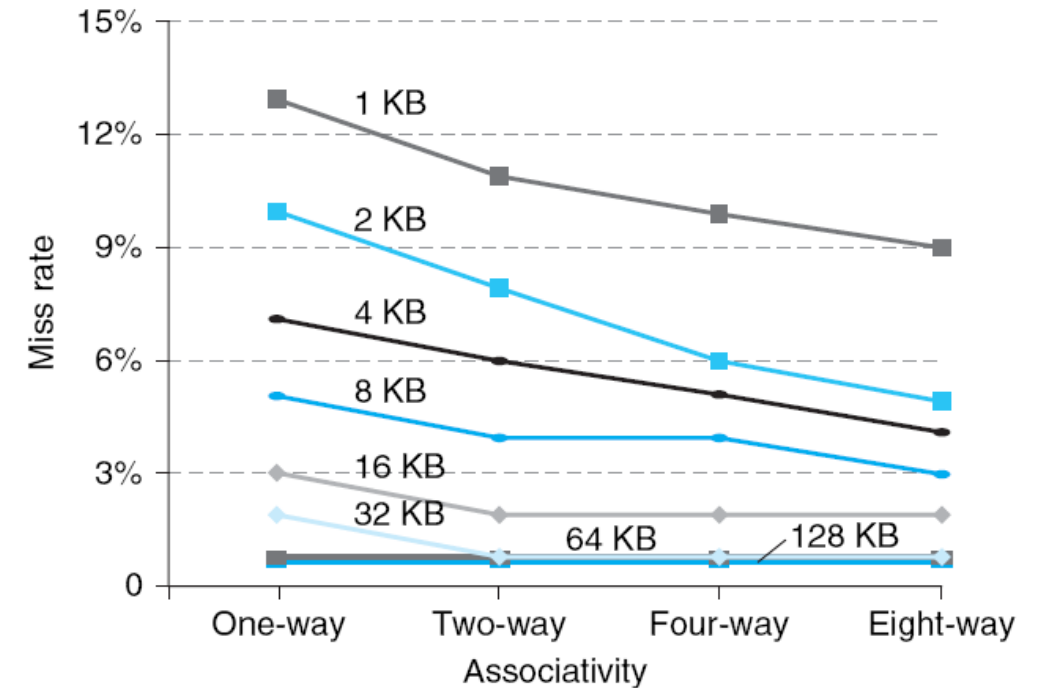
Based on SPEC92 Benchmark

# Associativity and Performance



# Associativity and Performance

- Higher associative caches
  - + Have better (lower) %<sub>miss</sub>
    - Diminishing returns
  - However  $t_{\text{access}}$  increases
    - The more associative, the slower
    - Why? The costs of an associative cache are the extra comparators and any delay imposed by having to do the compare and select from among the elements of the set.
    - The choice will depend on the cost of a miss versus the cost of implementing associativity, both in time and in extra hardware.
    - Recall CAM



# Cache Performance Improvement Techniques (So far)

$$AMAT = HitTime + MissRate \times MissPenalty$$

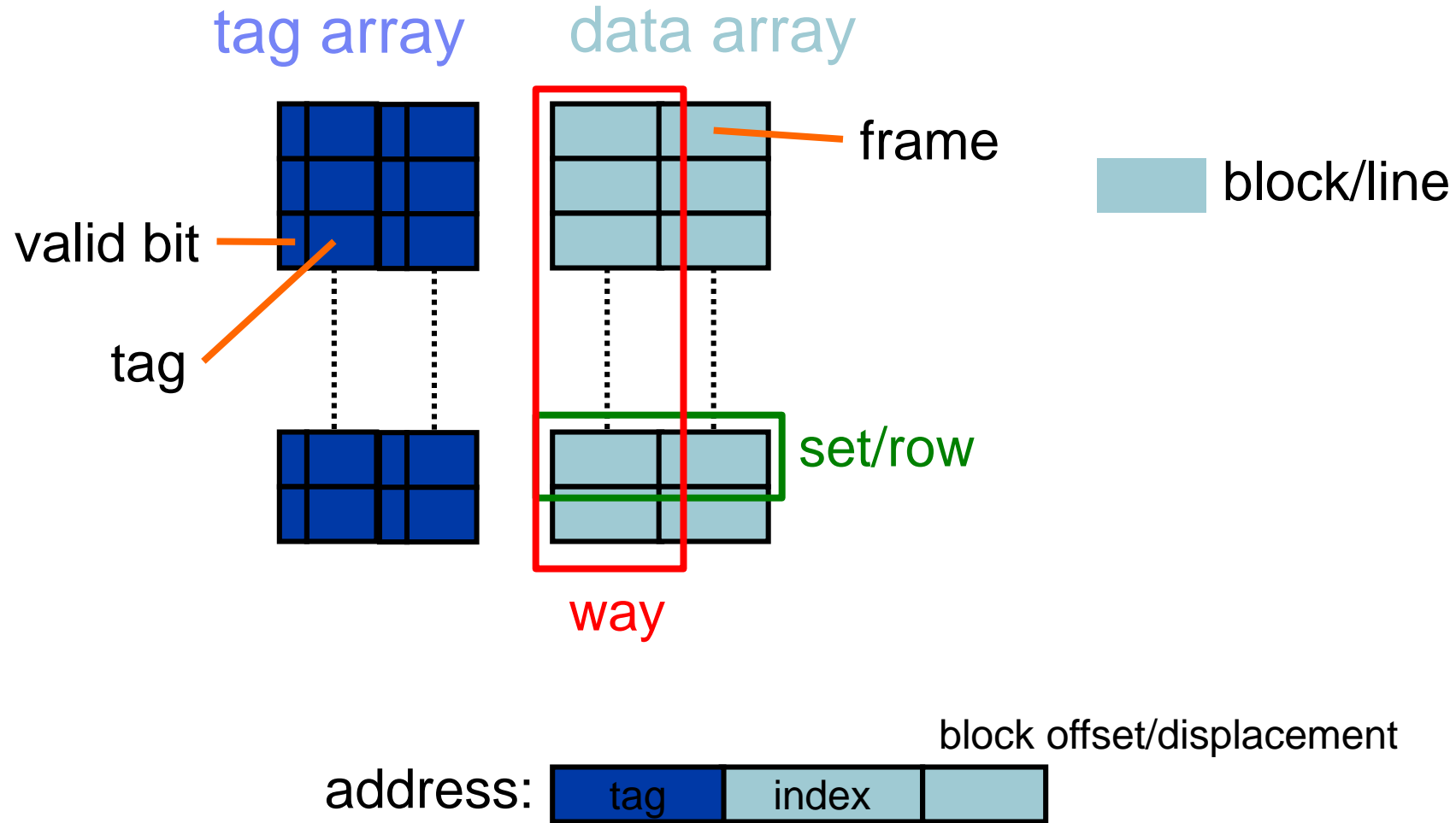
- 3 Cs: Compulsory, Capacity, Conflict
  - Reduce Misses via Larger Block Size
  - Reduce Misses via Higher Associativity
  - Reduce Misses via Bigger Cache
  - Reduce Misses via Multi-level caches
    - $CPI = \text{base CPI} + L-1 \text{ miss } L-2 \text{ hit (cycles per instruction)} + L-1 \text{ miss } L-2 \text{ miss (cycles per instruction)}$   
vs.  $CPI = \text{base CPI} + L-1 \text{ miss Memory hit}$

# Cache Quick Summary

- The cache operates as follows:
  - Whenever the CPU needs to read from a location residing in the main memory, it first checks the cache for any matching entries.
  - If the location exists in the cache, it is simply returned directly to the CPU; this is known as a cache hit.
  - If the location doesn't exist in the cache, also known as a cache miss, the cache allocates a new entry for the location, copies the contents from the main memory, and then fulfills the request from the contents in the cache.
- If the CPU needs to write some data, then it also checks the cache first and writes on a hit.
  - What happens on a miss is governed by the cache's policies, described in later lectures.
- The proportion of accesses that result in hits, as opposed to misses, is known as the hit rate and is a useful measure of the effectiveness of the cache.
- Cache Miss ABC
- More advanced cache performance optimization techniques will be discussed in the next lecture



# Appendix: Cache Glossary



# Where are we Heading?

- T5: Memories II

# Acknowledgement

Slides in this topic are inspired in part by material developed and copyright by:

- ARM Courseware
- Prof. Shuai Wang @ Nanjing U
- Prof. Joe Devietti @ Upenn, CIS 571
- Prof. Ron Dreslinski @ UMich, EECS 470
- Prof. Hakim Weatherspoon @ Cornell, CS 3410
- Prof. Krste Asanovic @ UCB, CS252
- Xinfei Guo @ JI, VE370 2021 SU

# Action Items

- HW #3 is out
- Lab #4 is out
- Reading Materials
  - Ch. 2.1 – 2.3
  - Ch. Appendix B.1-B.3