Computer Architecture
计 算 机 体 系 结 构

Lecture 5. Cache and Memory
第五讲、缓存和主存系统

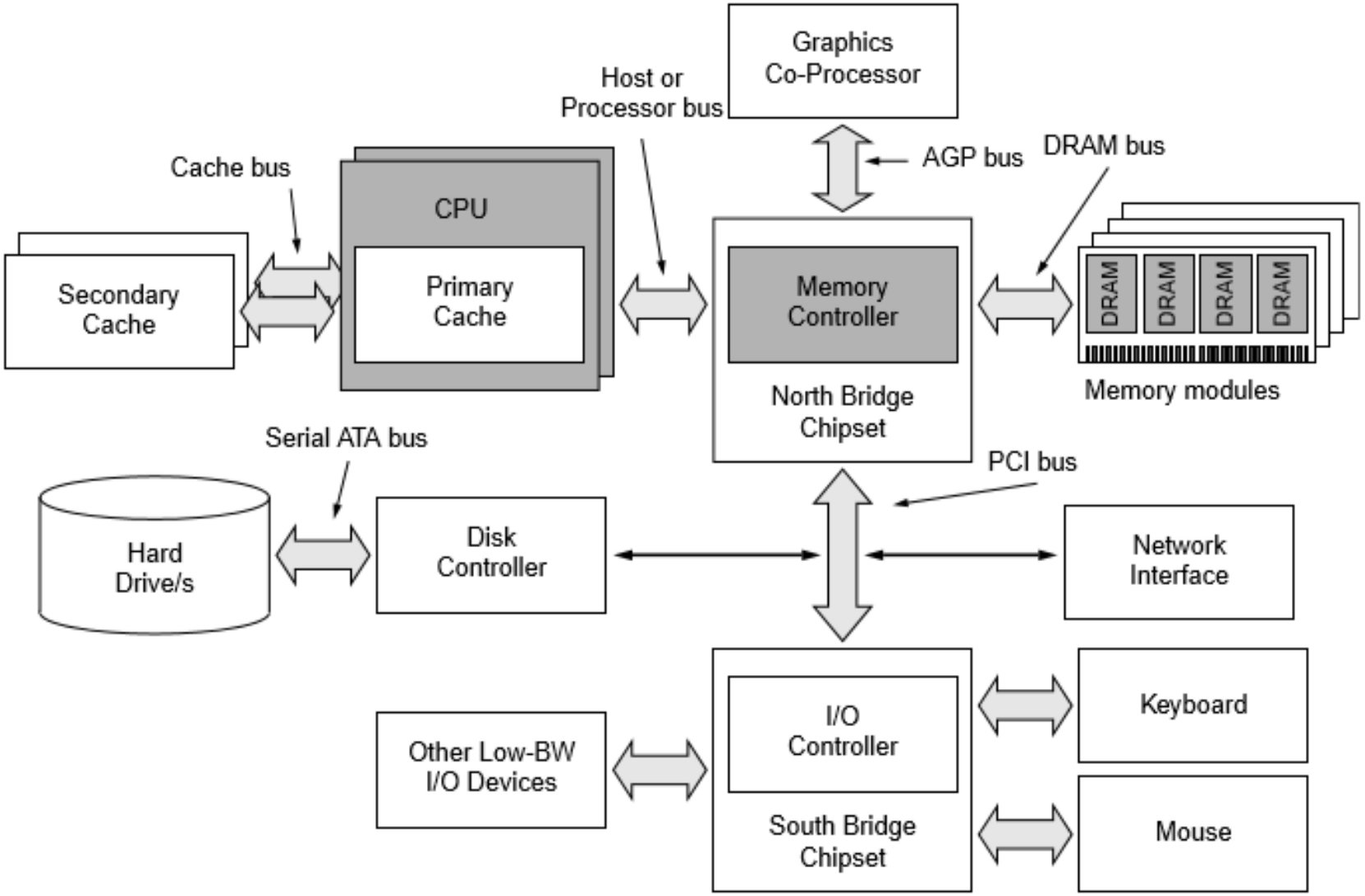
Chao Li, PhD.
李超 博士

SJTU-SE346, Spring 2019

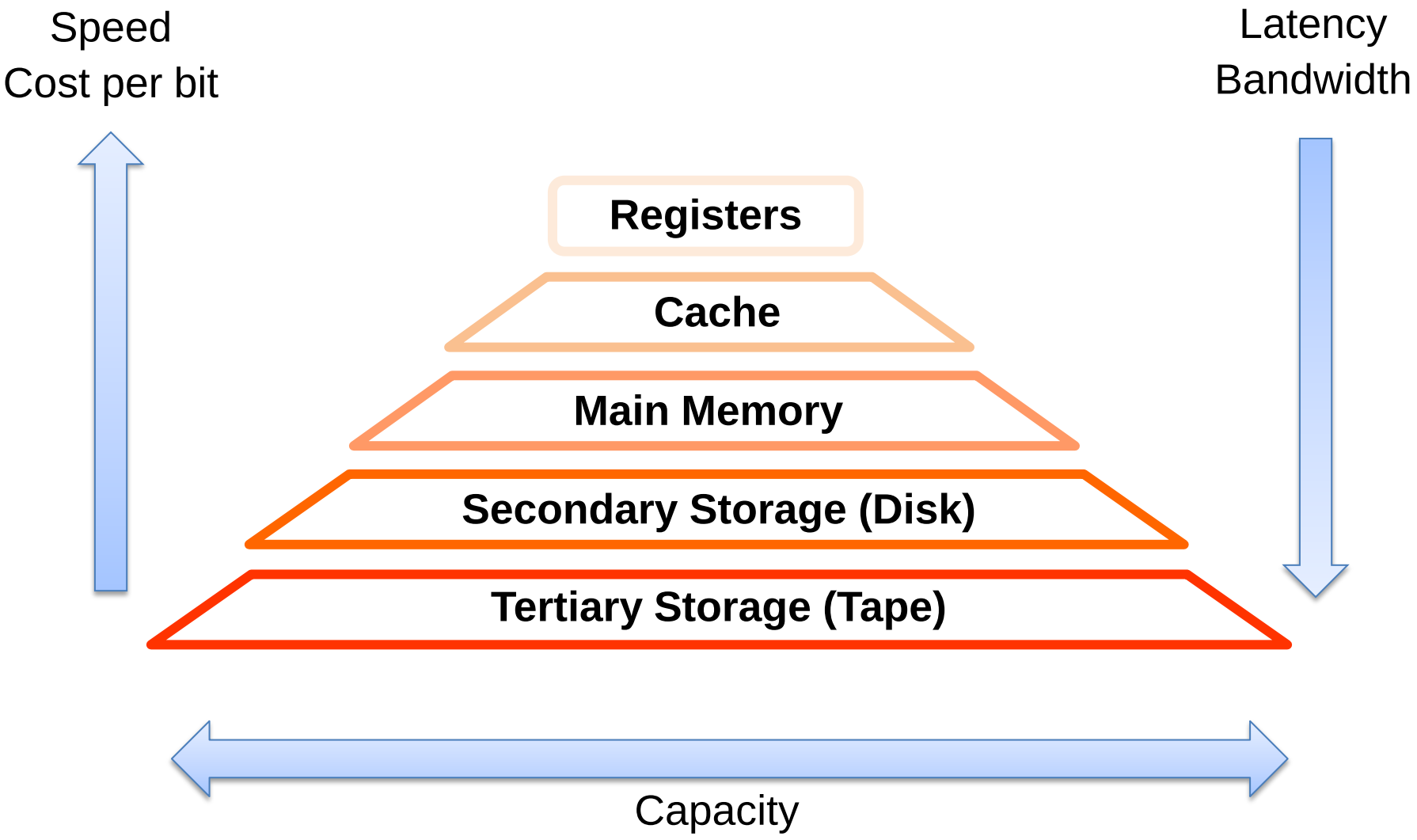
Review

- Three limitations of simple pipeline
- Superscalar pipeline and multiple issue
- Classification of ILP
- Branch prediction and precise exception
- Reorder buffer
- Tomasulo algorithm with ROB
- VLIW and loop unrolling

Typical PC Organization



Memory Hierarchy: An Overview



Memory Hierarchy: Parameters

Level	1	2	3	4
Called	Registers	Cache	Main memory	Disk storage
Typical size	< 1 KB	< 16 MB	< 16 GB	> 100 GB
Implementation technology	Custom memory with multiple ports, CMOS	On-chip or off-chip CMOS SRAM	CMOS DRAM	Magnetic disk
Access time (in ns)	0.25 -0.5	0.5 to 25	80-250	5,000,000
Bandwidth (in MB/sec)	20,000-100,000	5,000-10,000	1000-5000	20-150
Managed by	Compiler	Hardware	Operating system	Operating system/operator
Backed by	Cache	Main memory	Disk	CD or Tape

[Hennessy and Patterson]

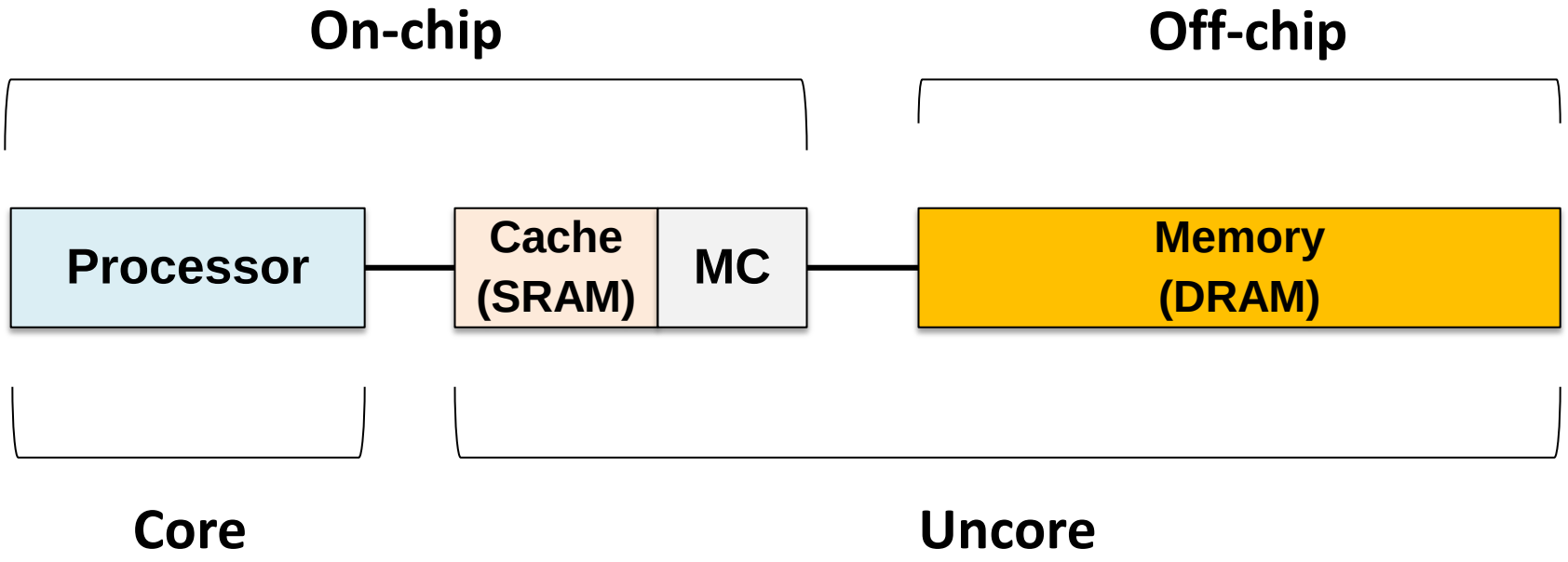
	Access type	Capacity	Latency	Bandwidth	Cost/MB
CPU registers	Random	64–1024 bytes	1–10 ns	System clock rate	High
Cache memory	Random	8–512 KB	15–20 ns	10–20 MB/s	$500
Main memory	Random	16–512 MB	30–50 ns	1–2 MB/s	$20–50
Disk memory	Direct	1–20 GB	10–30 ms	1–2 MB/s	$0.25
Tape memory	Sequential	1–20 TB	30–10,000 ms	1–2 MB/s	$0.025

[Abd-El-Barr and El-Rewini]

Outlines

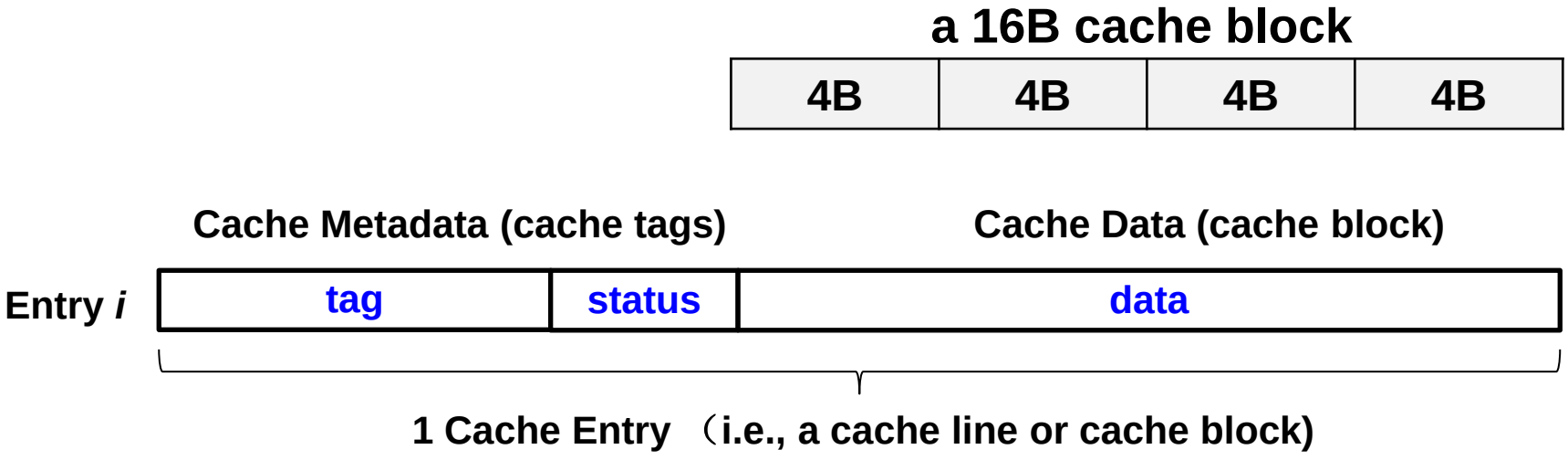
- Cache Basics
  - Concepts
  - Optimizations
- DRAM Basics

Basic Concepts

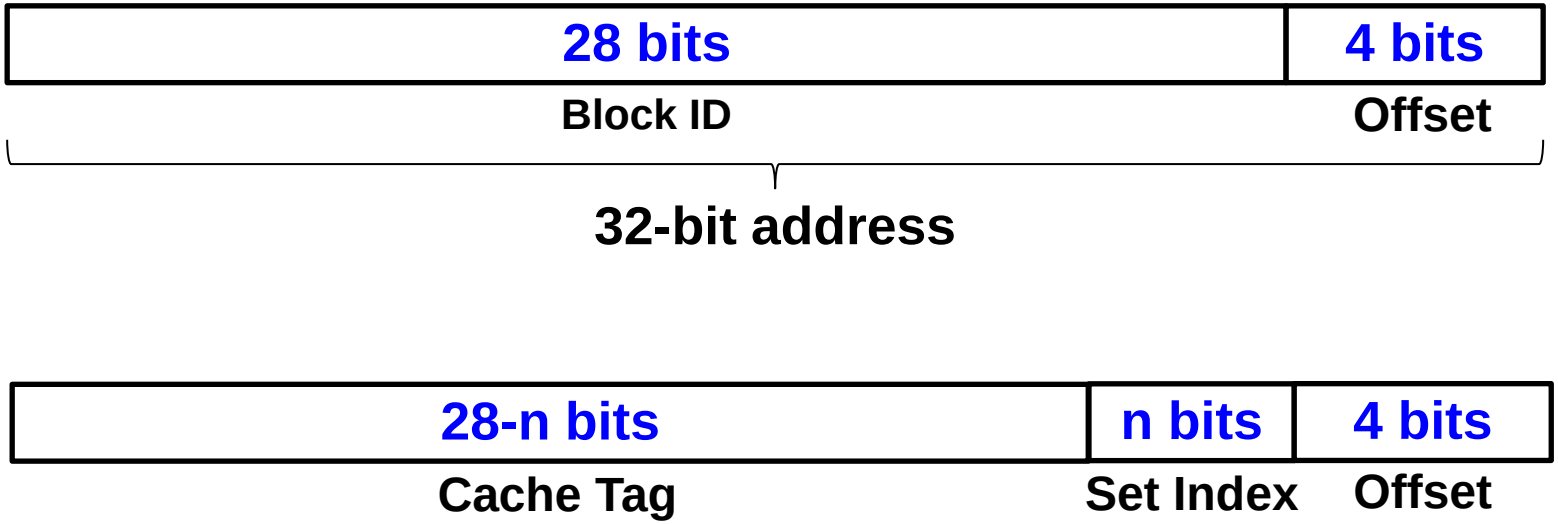


Cache and Data Access Granularity

- A cache stores chunks of data (called *cache blocks* or *cache lines*) that come from the backing store
- Cache entry fields
  - **tag**: gives the address of the block
  - **status**: typically a valid bit
  - **data**: the cache data

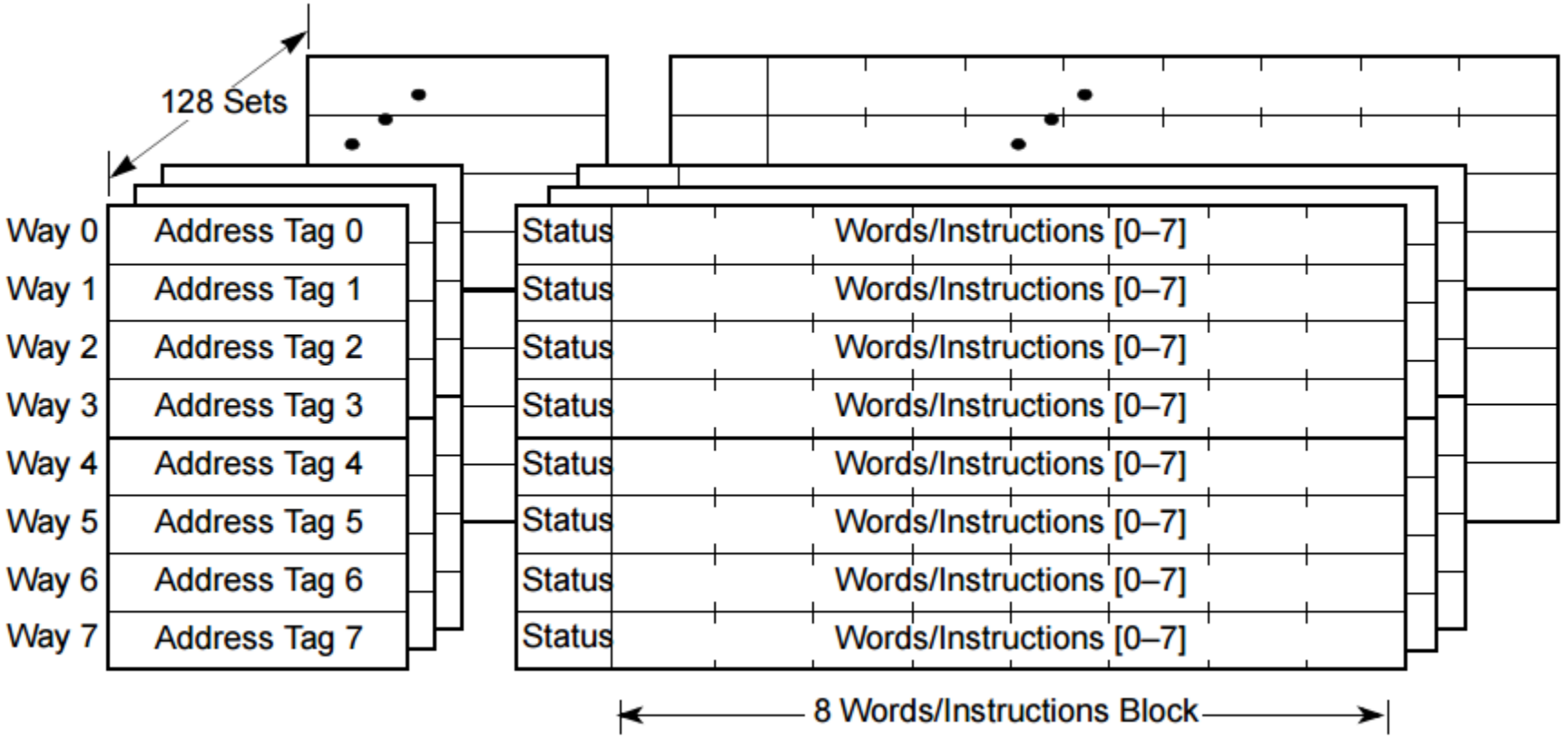


How is a Block Found - The Address



- Offset: the byte's offset within a cache block
  - 4-bit offset => cache block is 16B (4 words)
- Set Index: for selecting a set
  - n-bit Index => 2^n cache set

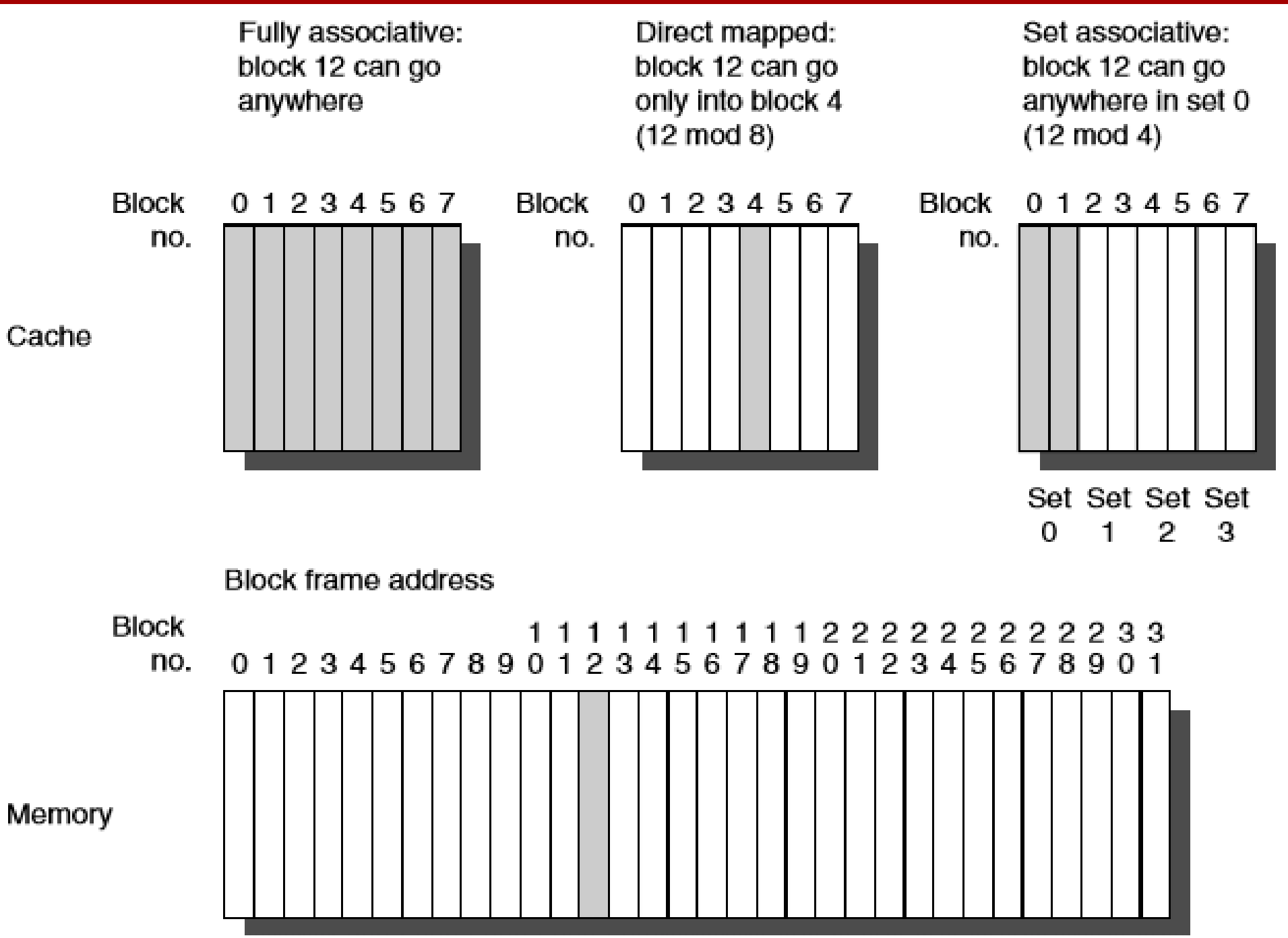
How is a Block Found - The Address



L1 Cache Organization

Where to Put Data: Logical Organization of Cache

- The cache is said to be *direct mapped*, if each block has only one place it can appear in the cache
  - e.g., each cache line is a “set”
- The cache is said to be *fully associative*, if a block can be placed anywhere in the cache
  - e.g., all the cache lines belong to one “set”
- The cache is said to be *set associative*, if a block can be placed in a restricted set of places in the cache
  - e.g., 4-way set associative: 4 lines per set

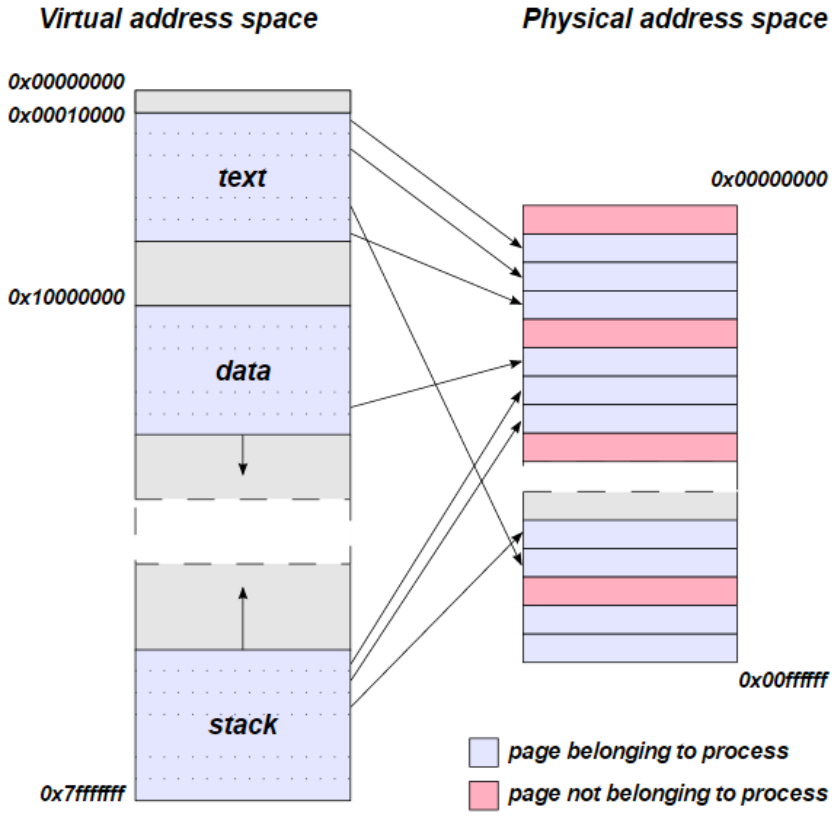

Fully associative: block 12 can go anywhere
Direct mapped: block 12 can go only into block 4 (12 mod 8)
Set associative: block 12 can go anywhere in set 0 (12 mod 4)
Block no.
0 1 2 3 4 5 6 7
Cache
Set 0
Set 1
Set 2
Set 3
Block frame address
Block no.
0 1 2 3 4 5 6 7 8 9 10 11 12 13 14 15 16 17 18 19 20 21 22 23 24 25 26 27 28 29 30 31
Memory

Virtual Memory

- How can one support programs that require more memory than is physically available?
- Program addresses are virtual addresses
- The basic idea is that the entire code need not be loaded into memory for the CPU to begin execution
- A page, or virtual page is a fixed-length contiguous block of virtual memory

Virtual Memory and Address Translation (I)

A page table is an array of page table entries (PTEs) that maps virtual pages to physical pages



Assuming 32-bit address, 4KB page size and 4-byte PTE:

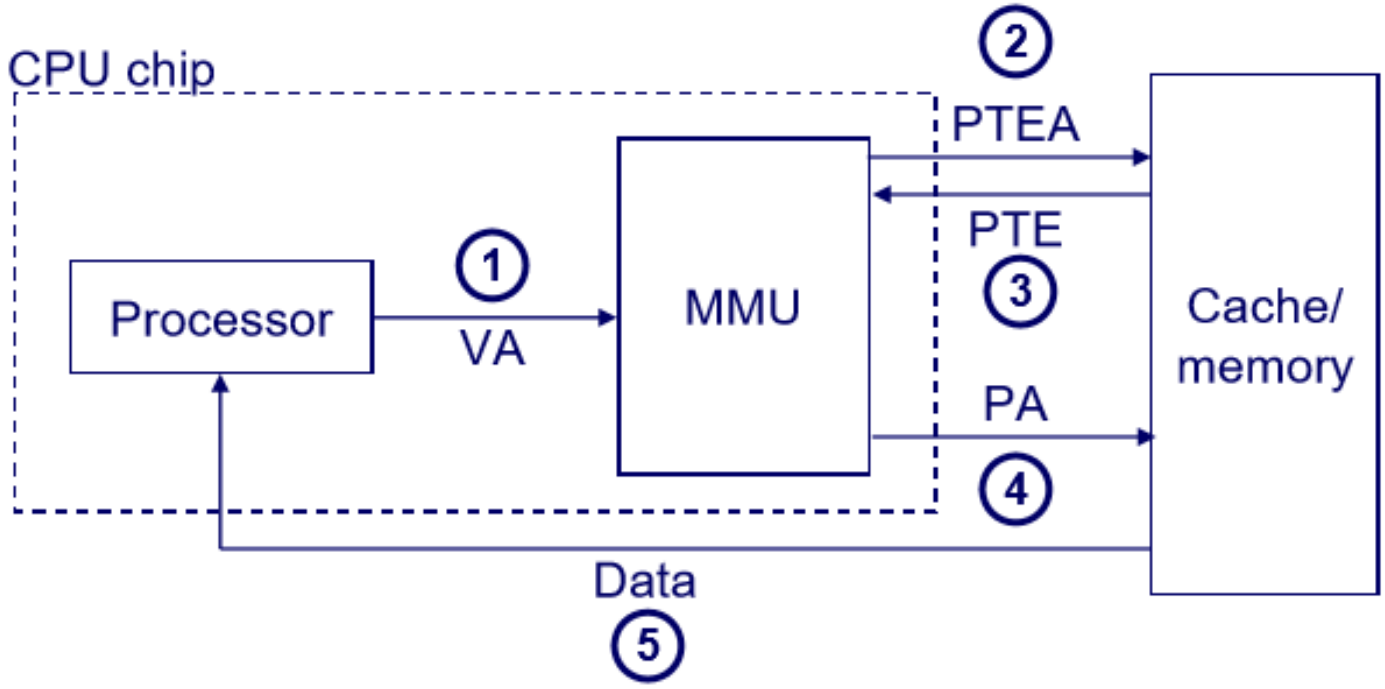
How many PTEs?

virtual space size / page size = 2^20 (1 M) entries

The size of page table for each user/process?

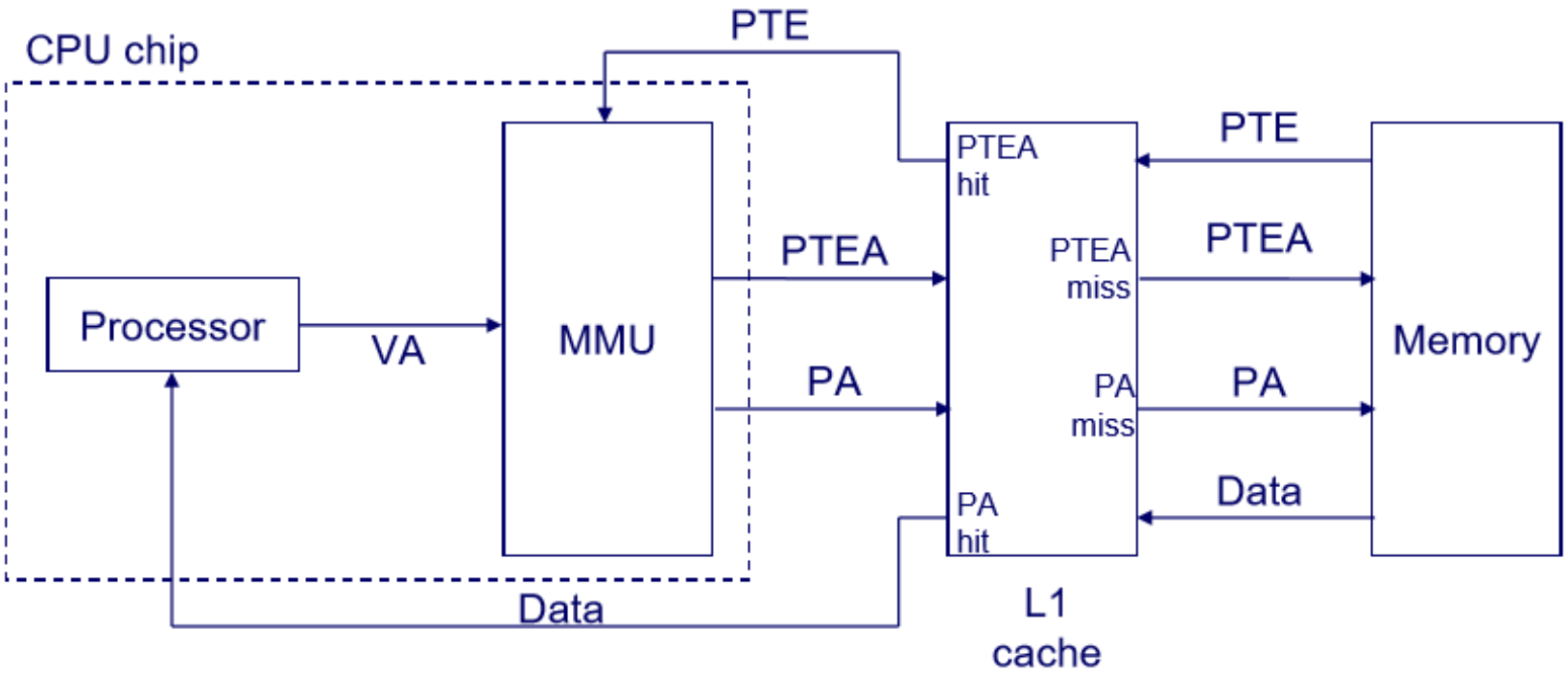
1M entries at 4-byte each: 4MB memory space

Virtual Memory and Address Translation (II)



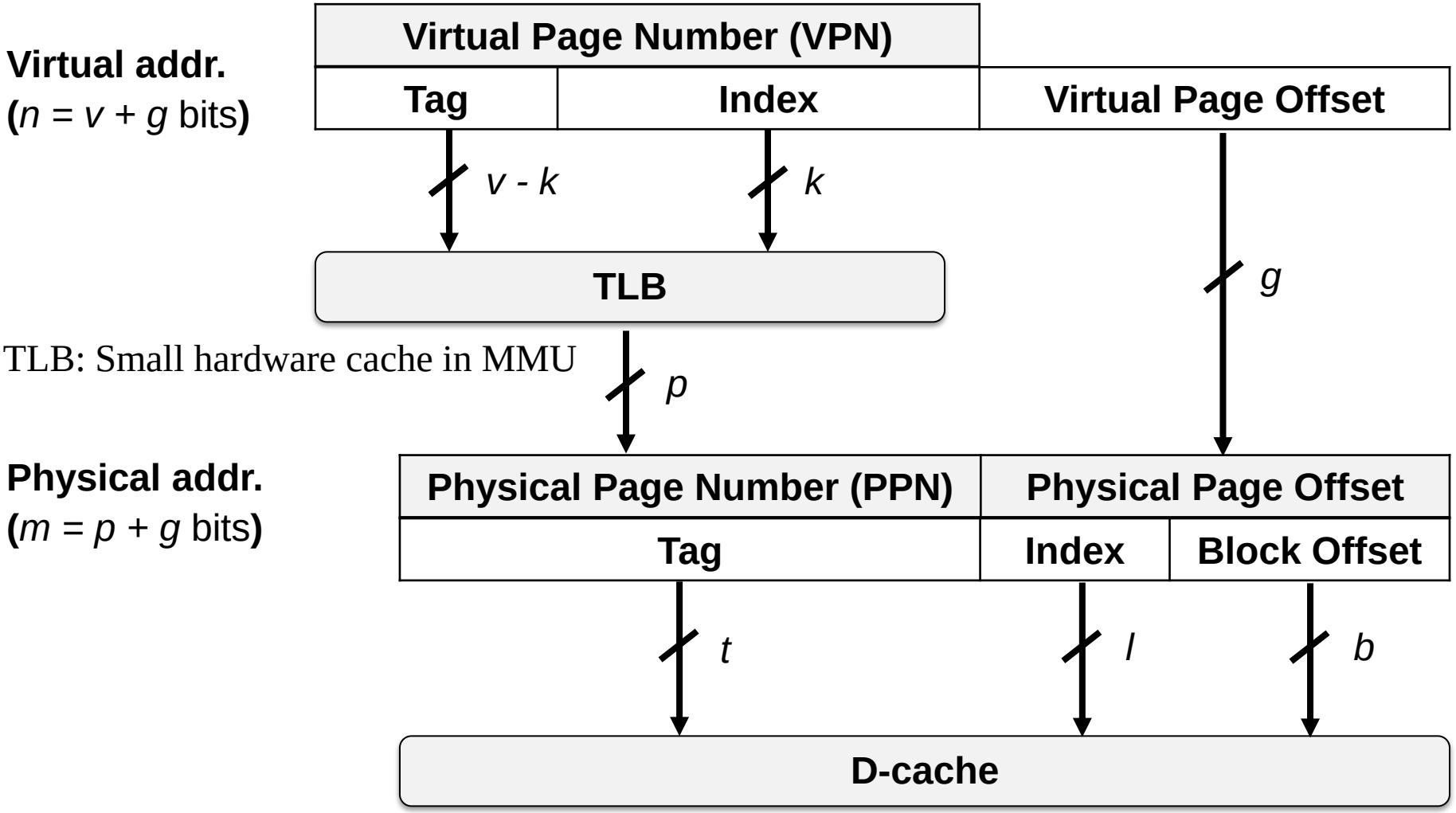
- Processor sends virtual address to MMU (1)
- MMU fetches PTE from page table in memory (2-3)
- MMU sends physical address to L1 cache
- L1 cache sends data word to processor

Virtual Memory and Address Translation (III)



- PTEs are cached in L1 like other memory word
  - Can be evicted by other data references
  - PTE hit still requires a 1-cycle delay
- Solution: cache PTEs in a small fast memory in the MMU

Virtual Memory and Address Translation (IV)



Outlines

- Cache Basics
  - Concepts
  - Optimizations

- DRAM Basics

Locality Principles

- Temporal locality is the tendency of programs to use data items over and again during the course of their execution
- Spatial locality arises because of the tendency of programmers and compilers to cluster related objects together in the memory space
- Algorithmic locality arises when a program repeatedly accesses data items or executes code blocks that are distributed widely across the memory space.

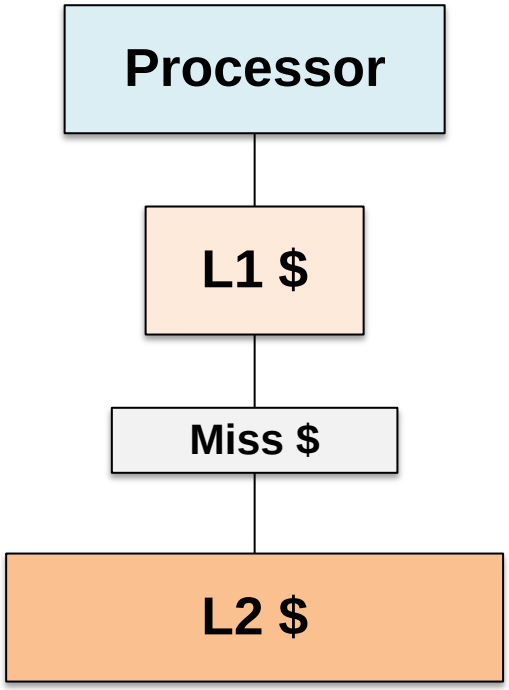
Reducing Miss Rate

Mark D. Hill
Prof. of UW-Madison
IEEE/ACM Fellow

Inventor of the 3C cache model

- "3C model" of cache behavior
  - *Compulsory miss*: The very first access to a block cannot be in the cache, a.k.a. cold start misses
  - *Capacity miss*: If the cache cannot contain all the blocks needed during execution of a program.
  - *Conflict miss*: A block may be discarded and later retrieved if too many blocks map to its set.

Discussion: Miss Caching vs Victim Caching

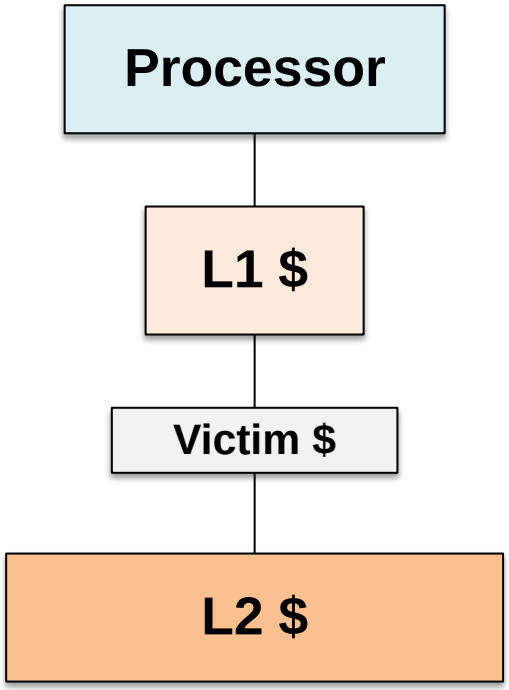


Miss Caching

1. Check the miss cache on a miss in the L1 cache; If the block is there, bring it into the L1 cache
2. Otherwise, fetch from L2 and store the retrieved value in both the miss cache and the L1 cache

- Fully-associative cache inserted between L1 and L2
- Contains between 2 and 5 cache lines of data
- Aims to reduce conflict misses

Discussion: Miss Caching vs Victim Caching



Victim Caching

1. Check the victim cache on a miss in the L1 cache. If the block is there, bring it into L1 and swap the ejected value.
2. Otherwise, fetch the block from L2

- Modified replacement policy
- Outperform miss cache: smaller, better performance
- Even a single line can be effective

Prefetching

- Miss cache and victim cache are most helpful when temporal locality can be exploited

- Fetching memory blocks ahead of time can be attractive
  - Fill request in anticipation of data request
  - Prefetching instructions, i.e., branch prediction
  - Prefetching sequential data accesses

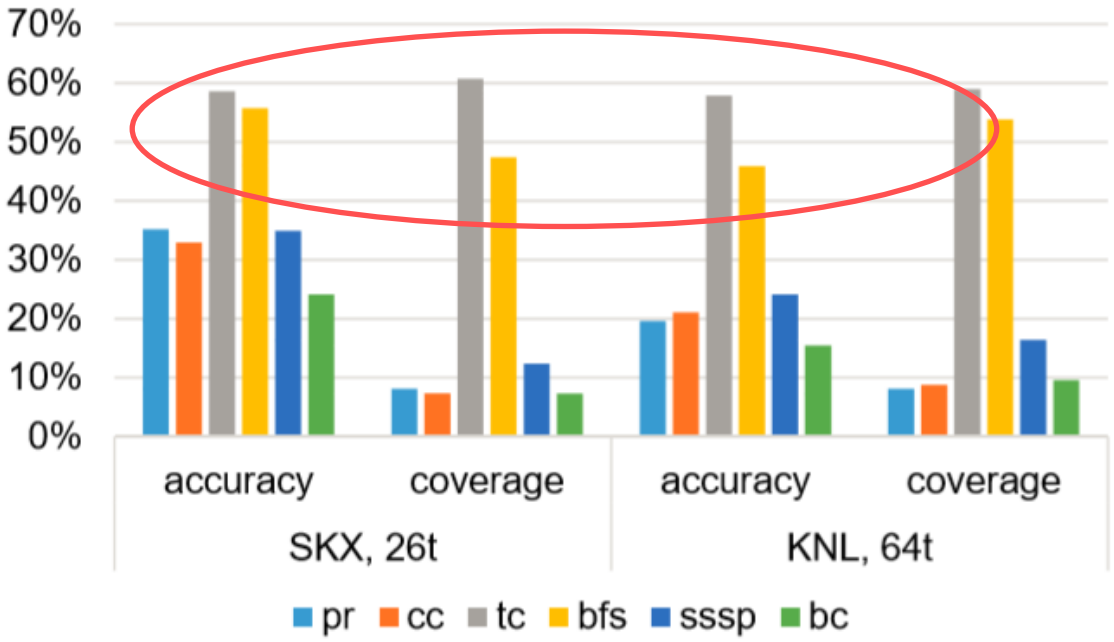
- Instead of prefetching directly into the L1 cache, one can also place prefetched data in a special buffer
  - Prefetched blocks that are never referenced by the processor do not evict potentially useful blocks from the L1 cache

Content Management

- Partitioning heuristics
  - Decide whether to cache an object or not and/or in which cache an object should be placed
- Prefetching heuristics
  - Decide whether and when to bring an item into a cache (the least interesting of which is demand-fetch, i.e., waiting until the item is actually requested)
- Locality optimizations
  - Mechanisms that (re-)organize code, data, and their references to best exploit cache behavior

Discussion: Impact of Data Prefetch

- L2 prefetcher accuracy and coverage
  - Accuracy is defined as the number of useful prefetches relative to the total number of prefetches
  - Coverage is the number of avoided misses (i.e., useful prefetches) relative to the total number of misses in absence of a prefetcher



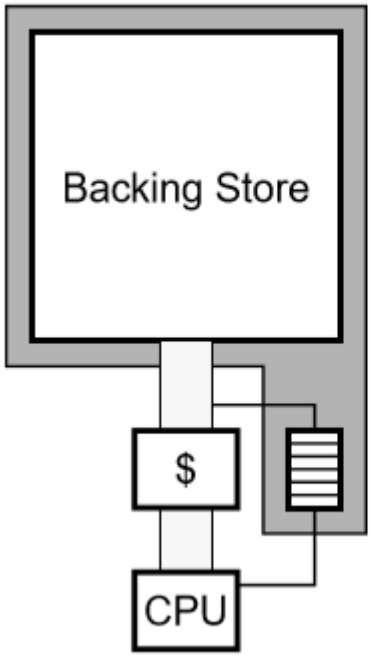
Cache Write Policies

- **Write operation upon a cache hit**
  - write-back: The value is only written to the cache. The modified ("dirty") block is written to memory at replacement time.
    - Advantage：save bandwidth and energy
  - write-through: Every write operation to the cache is repeated to the main memory at the same time
    - Advantage：easy to implement, keeps cache clean

- **Write operation upon a cache miss**
  - write allocate: The main memory block is brought to the cache and then updated.
  - no-write allocate: Write misses do not affect the cache. Instead, the block is modified only in the lower level memory

Cache Write Policies

- Sending every write to the backing store can be expensive
  - A typical solution is to insert a new memory structure into the system that uses write-through policy



- A *write buffer*
  - if it is a tag-less FIFO organization
- A *write cache*
  - If it has tags can be probed like a cache

Handing a request to the backing store should be careful

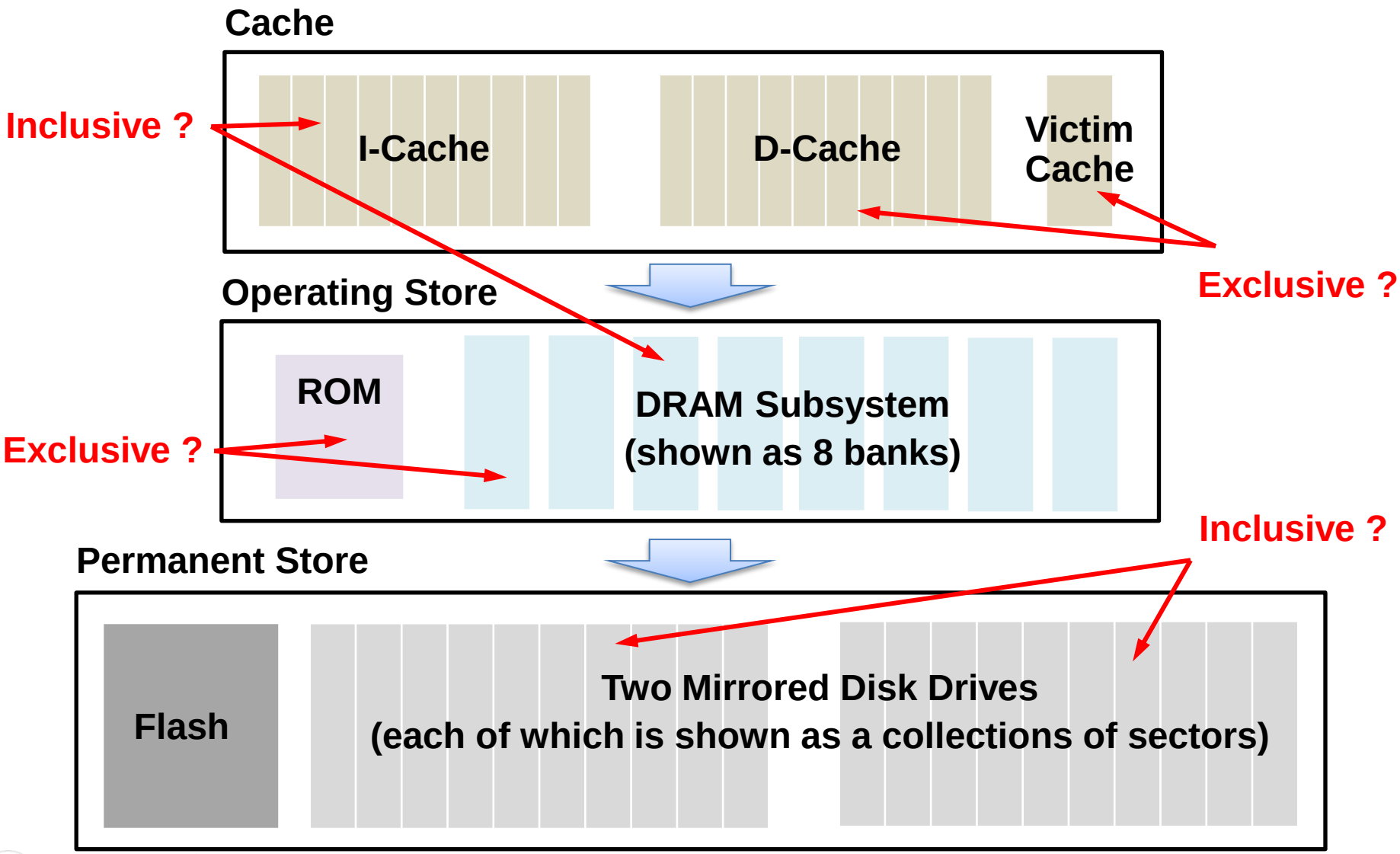
Write buffer: have to dump its entire contents to the backing store

Write cache: have to be probed as part of the backing store

Consistency Management

- The memory hierarchy is vertically partitioned into separate levels, and each level can be horizontally partitioned further
- The principles of inclusion and exclusion define a particular class of relationship that exists between any two partitions.
- An inclusive relationship:
  - Every cached item found in one has a copy found in the other
- An exclusive relationship:
  - The expected intersection between two units is the null set

A Specific Example of a Memory System

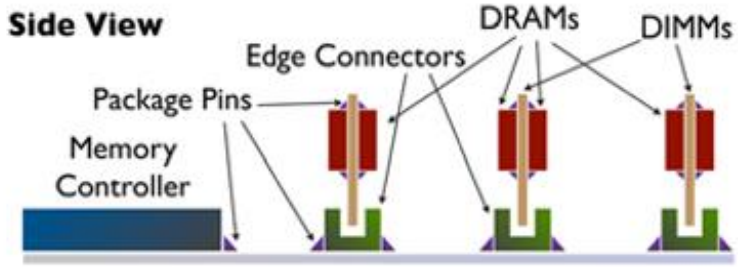


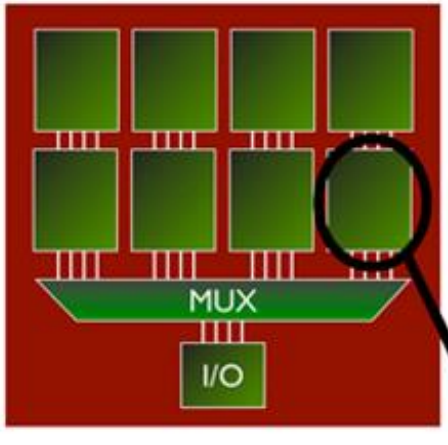
Outlines

- Cache Basics
  - Concepts
  - Optimizations

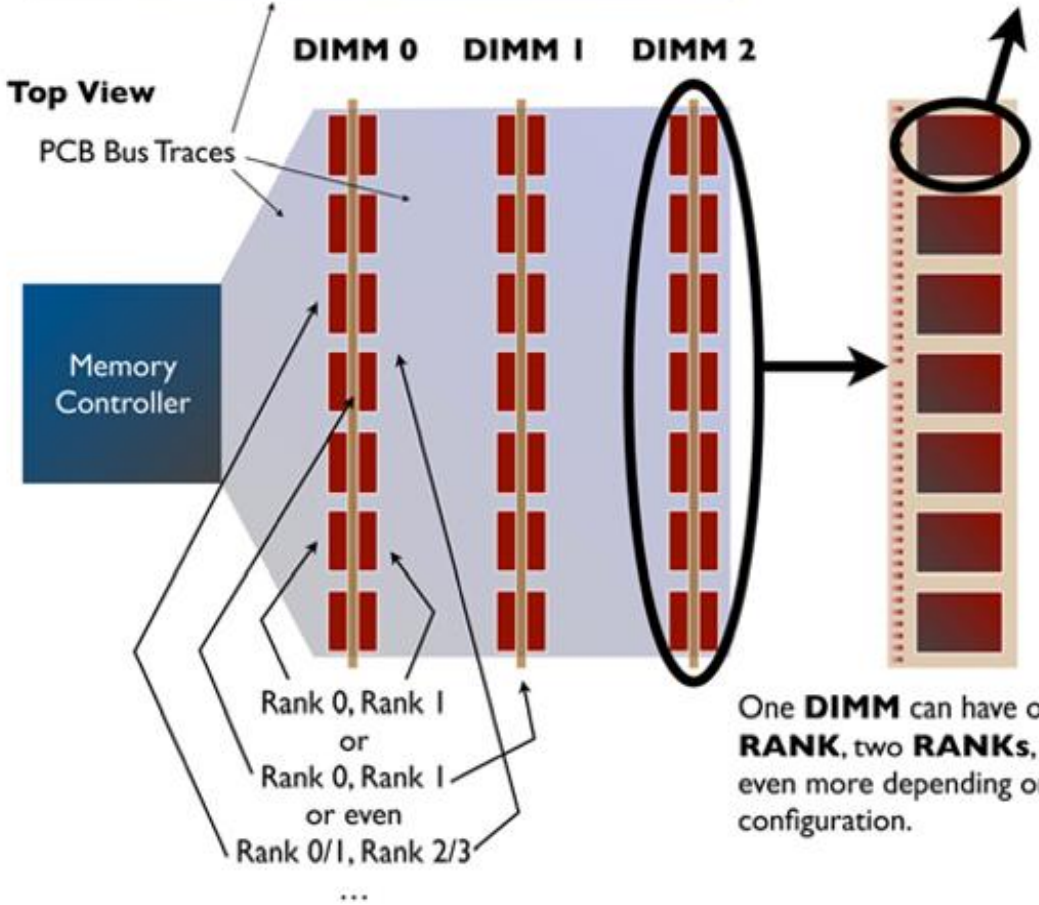
- DRAM Basics

Overview of DRAM System





One **DRAM device** with eight internal **BANKS**, each of which connects to the shared I/O bus.



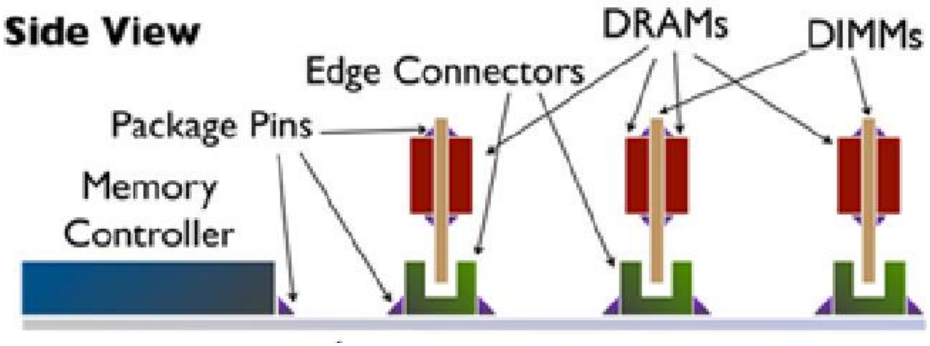
One **DIMM** can have one **RANK**, two **RANKs**, or even more depending on its configuration.

One **BANK**, four **ARRAYS**

One **DRAM bank** is comprised of many **DRAM ARRAYS**, depending on the part's configuration. This example shows four arrays, indicating a x4 part (4 data pins).

Physical Architecture of Main Memory

DIMM (dual in-line memory module)



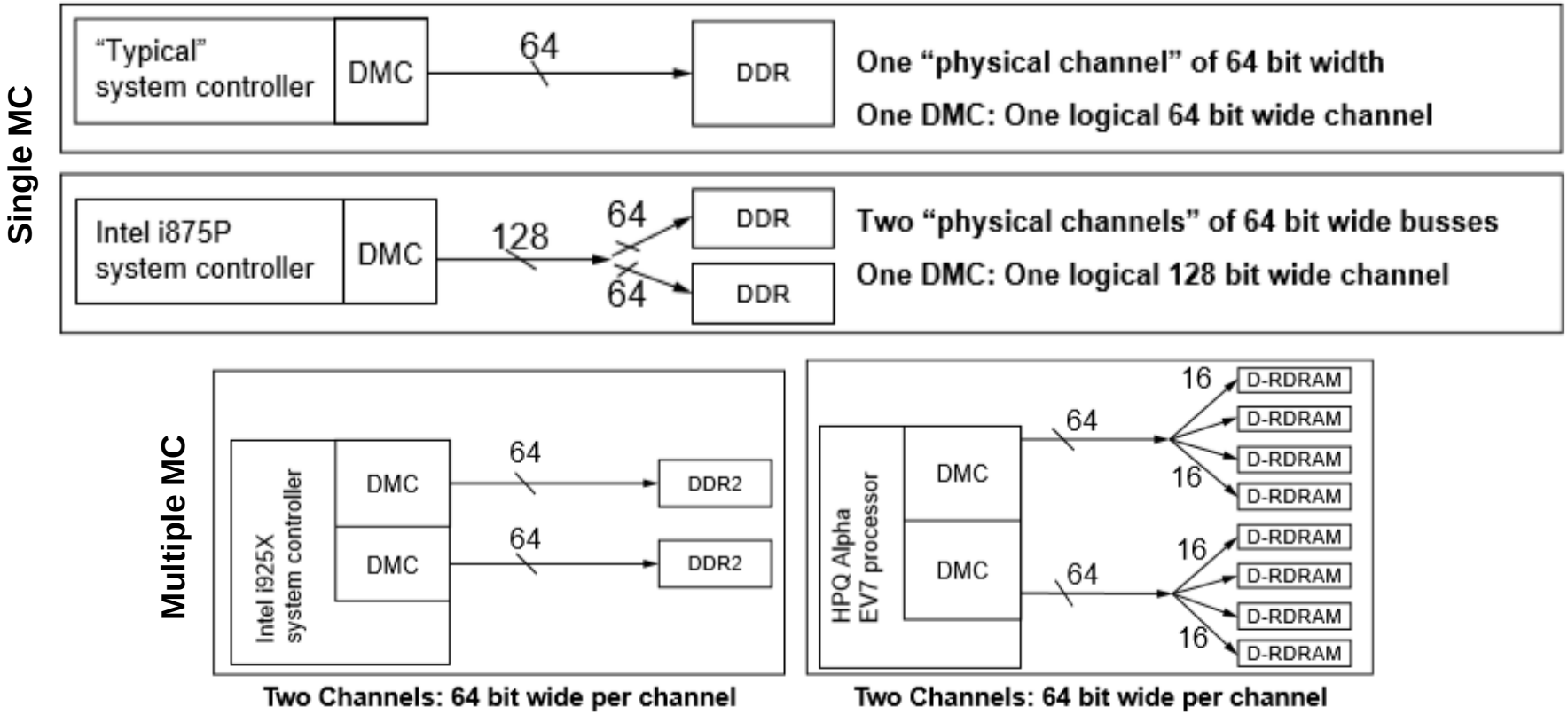
Memory modules are essentially miniature system boards that hold a number of DRAM devices

Oftentimes, a system is comprised of many independent DIMMs

- -Each DIMM may contain one or more independent ranks
- -Each rank is a set of DRAM devices that operate in unison
- -Each DRAM device has one or more independent banks
- -Finally, each bank is comprised of slaved memory arrays

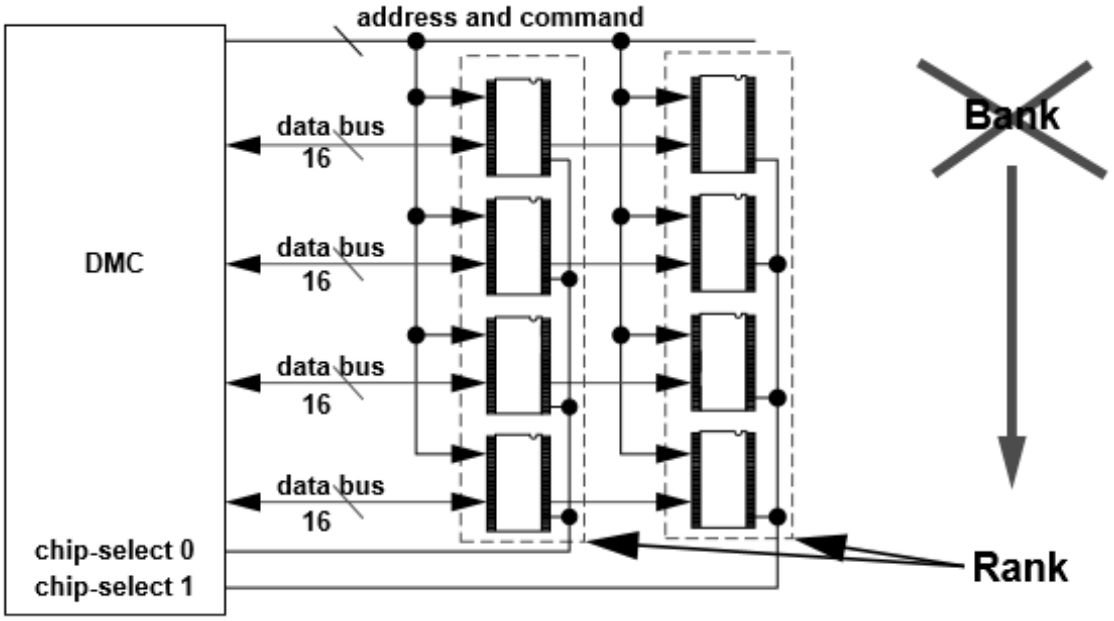
Revisit the Definition of Memory System

- **Channel:** The path between MC and DRAM module



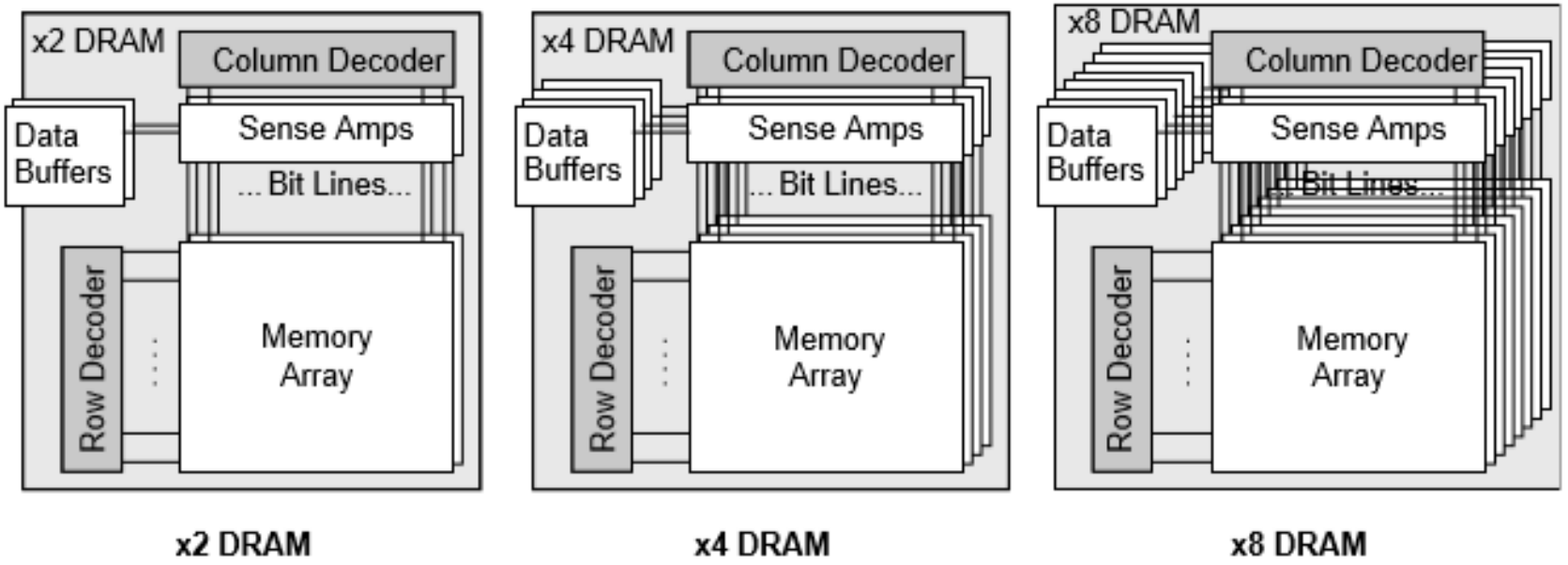
Revisit the Definition of Memory System

- **Rank:** is a set of one or more DRAM devices that operate in lockstep in response to a given command
- **Bank:** describe a set of independent memory arrays inside of a DRAM devices



Memory system with 2 ranks of DRAM devices

Revisit the Definition of Memory System

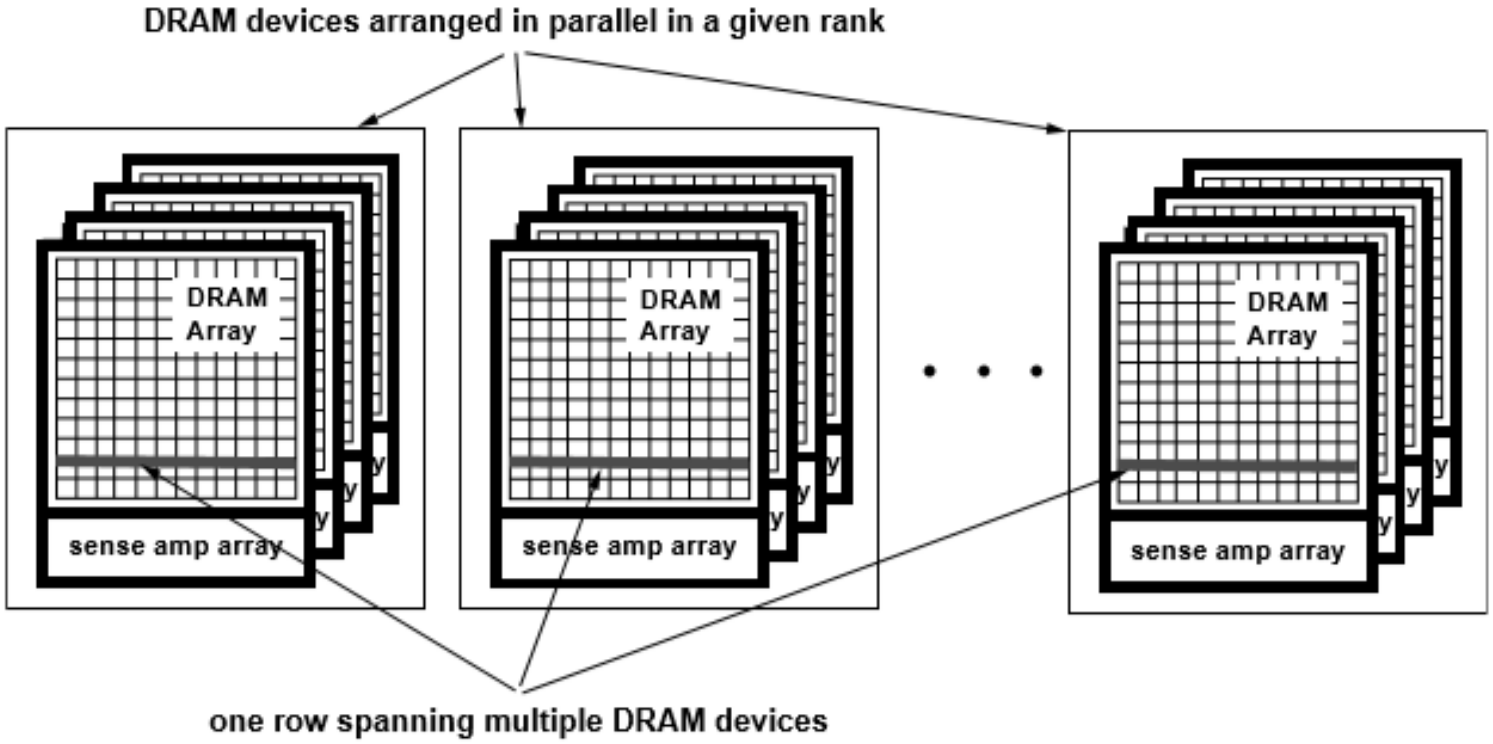


Wide Data-out DRAMs

- DRAM chips are described as xN, where N refers to the number of output pins; A x4 DRAM (pronounced “by four”) indicates that the DRAM has at least four memory arrays (in a single bank) and that a column width is 4 bits (each column read or write transmits 4 bits of data).

Revisit the Definition of Memory System

- A row is also referred to as a DRAM page.

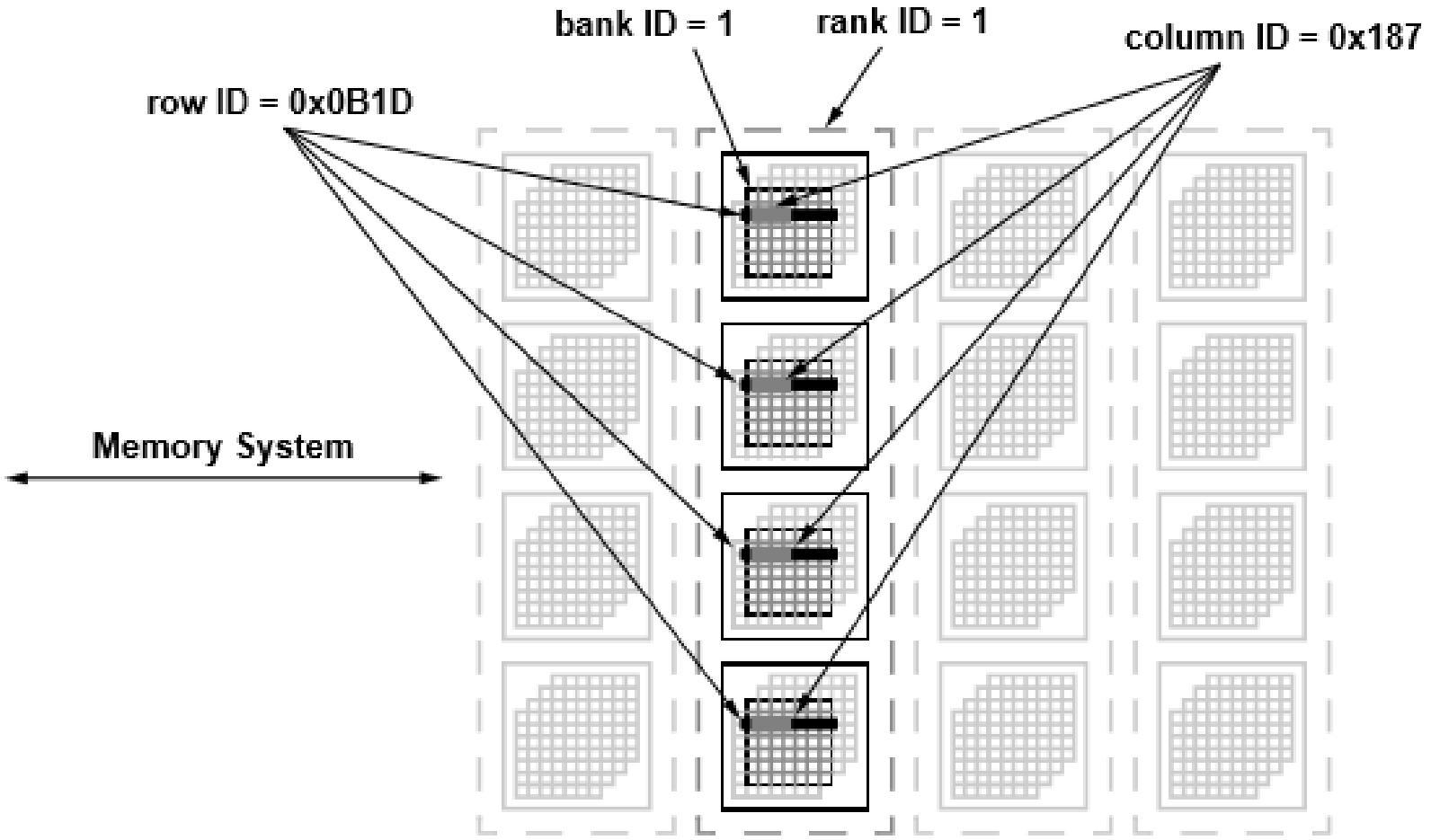


A row activation command will activate the same addressed row in all DRAM devices in a given rank of memory

Parallelism in DRAM

- Rank is a separately addressable set of DRAM devices. It allows for DIMM-level independent operation.
- Each set of memory arrays (inside of a DRAM device) that operates independently is referred to as a bank.
- Multiple banks are provided so we can be simultaneously working on different requests（*interleaving*）
- Having concurrency at the rank and bank levels provides bandwidth through the ability to pipeline requests

Location of Data in a DRAM

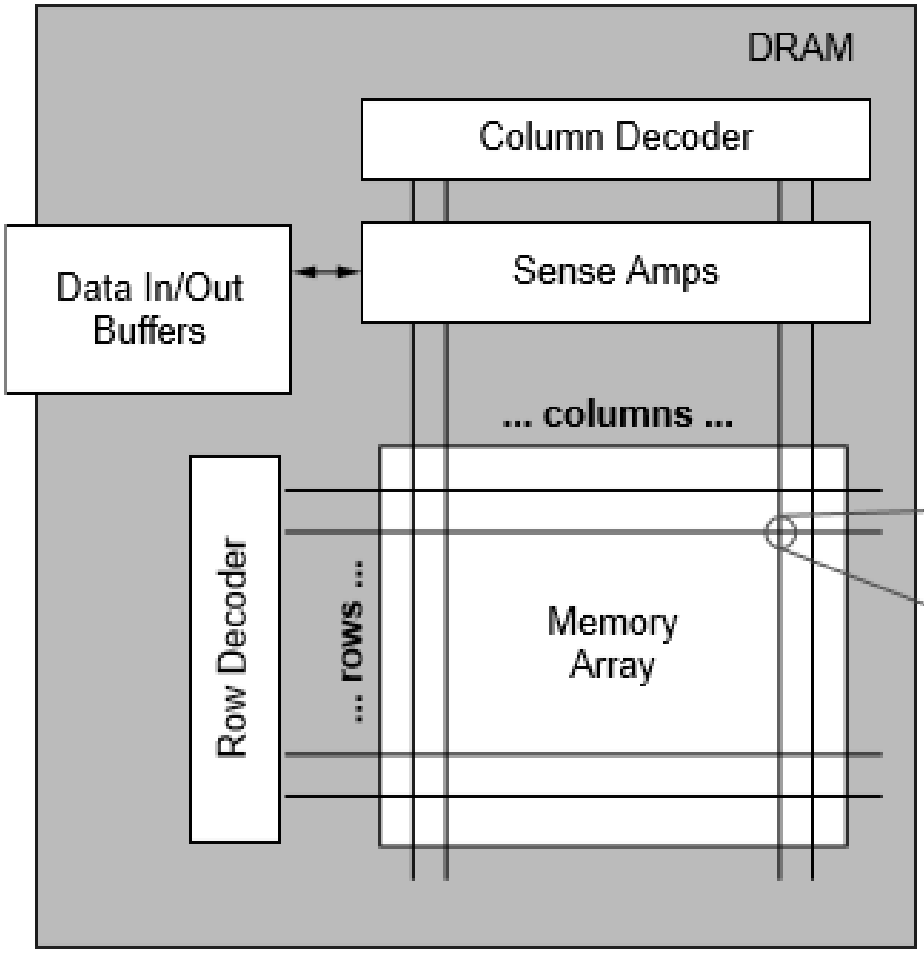


A column of data is the smallest addressable unit of DRAM

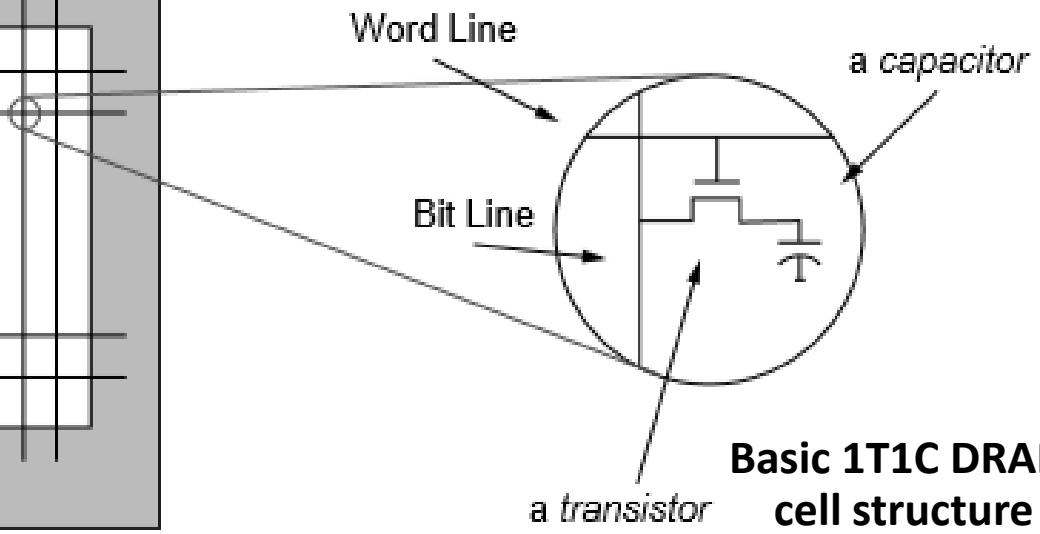
Address Mapping (Translation)

- Physical address is resolved into indices:
  - Channel ID, rank ID, bank ID, row ID, column ID
- Example address mapping policies:
  - row:rank:bank:channel:column:blkoffset
    row:column:rank:bank:channel:blkoffset
- Consecutive cache lines can be placed in the same row to boost row buffer hit rates
- Consecutive cache lines can be placed in different ranks to boost parallelism

1T1C DRAM Cell

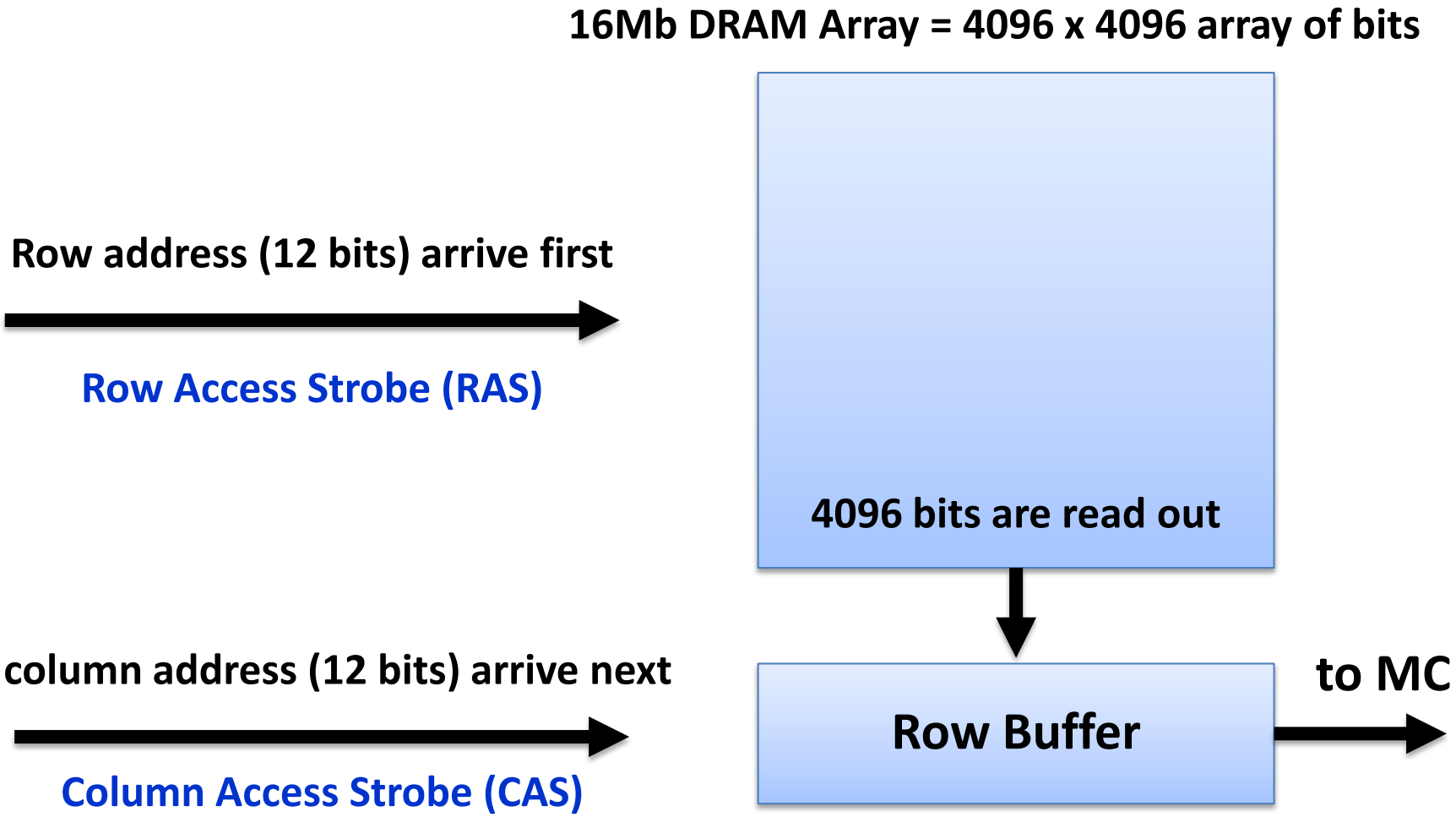


- A single transistor-capacitor pair for a bit in DRAM
- Each capacitor must be periodically ***refreshed***
- A control bus is composed of the row and column strobes, clock, and other signals.



Basic 1T1C DRAM cell structure

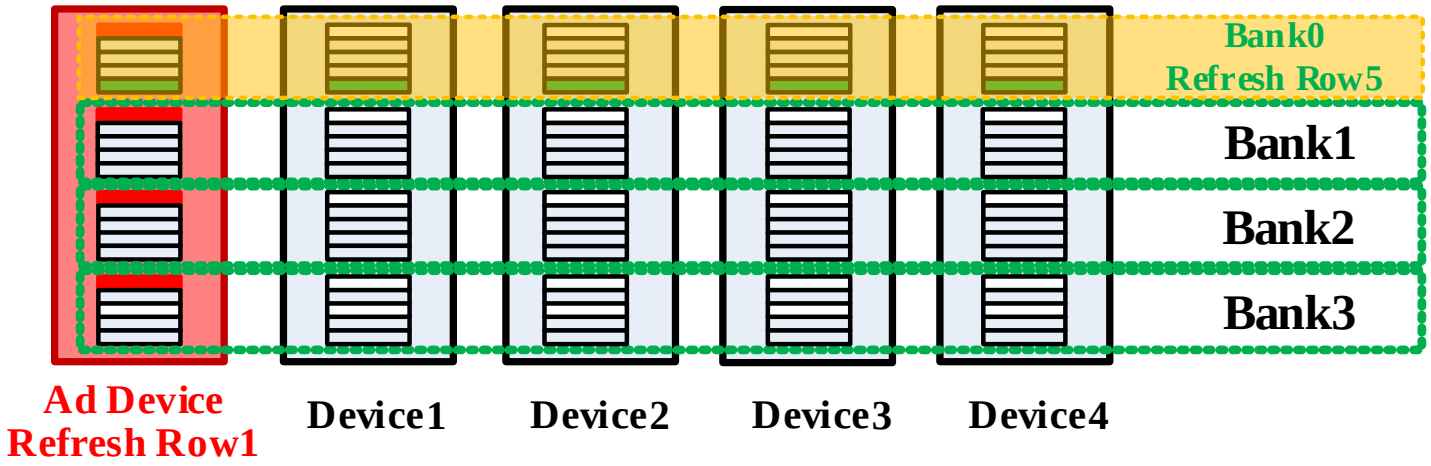
Example: DRAM Array Access



Classical DRAM systems: width of data bus equals column size. Send N bits for xN DRAM, one per cycle

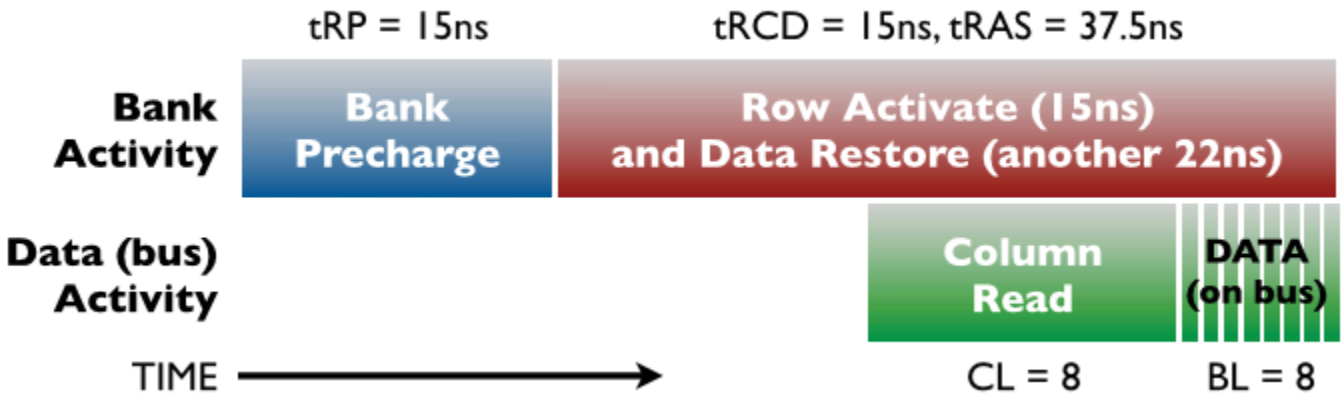
Refresh Mechanism

- A Row is the smallest refresh unit in bank.
- Typically, the retention time of data in DRAM cell is 64ms if the ambient temperature is less than 85 degree Celsius.
- Refresh operation can implement at rank level (per-rank refresh) or at bank level (per-bank refresh)



In per-rank refresh, all banks are locked and inaccessible during refresh cycle.

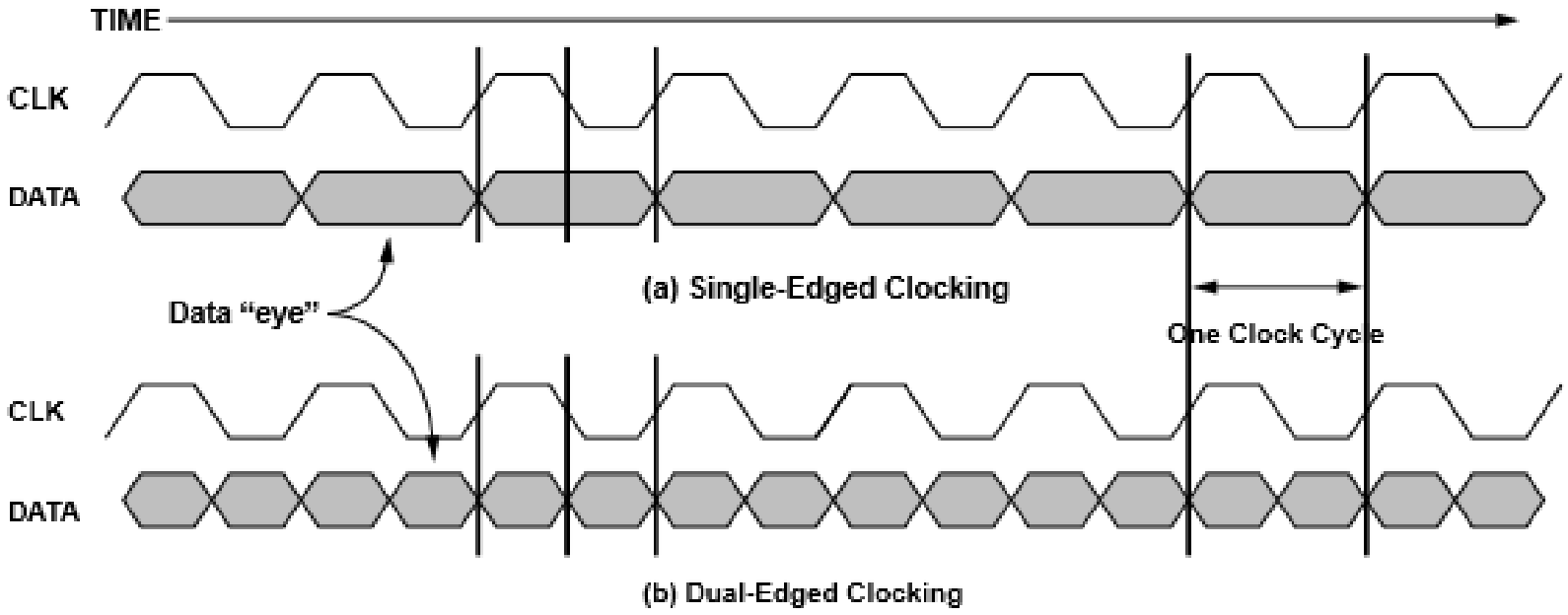
Cost of Accessing DRAM



- Row buffers act as a cache within DRAM
  - Row buffer hit: ~20 ns access time
    - must only move data from row buffer to pins
  - Empty row buffer access: ~40 ns
    - must first read arrays, then move data from row buffer to pins
  - Row buffer conflict: ~60 ns
    - must first write back, then read new row, then move data

Synchronous and Asynchronous Devices

- Synchronous DRAMs (SDRAM):
  - If they are driven directly by the system clock signal
  - 用时钟取代RAS和CAS，从而获得更高数据传输率
- Double Data Rate (DDR) SDRAM
  - Data is transmitted at both edges of the clock

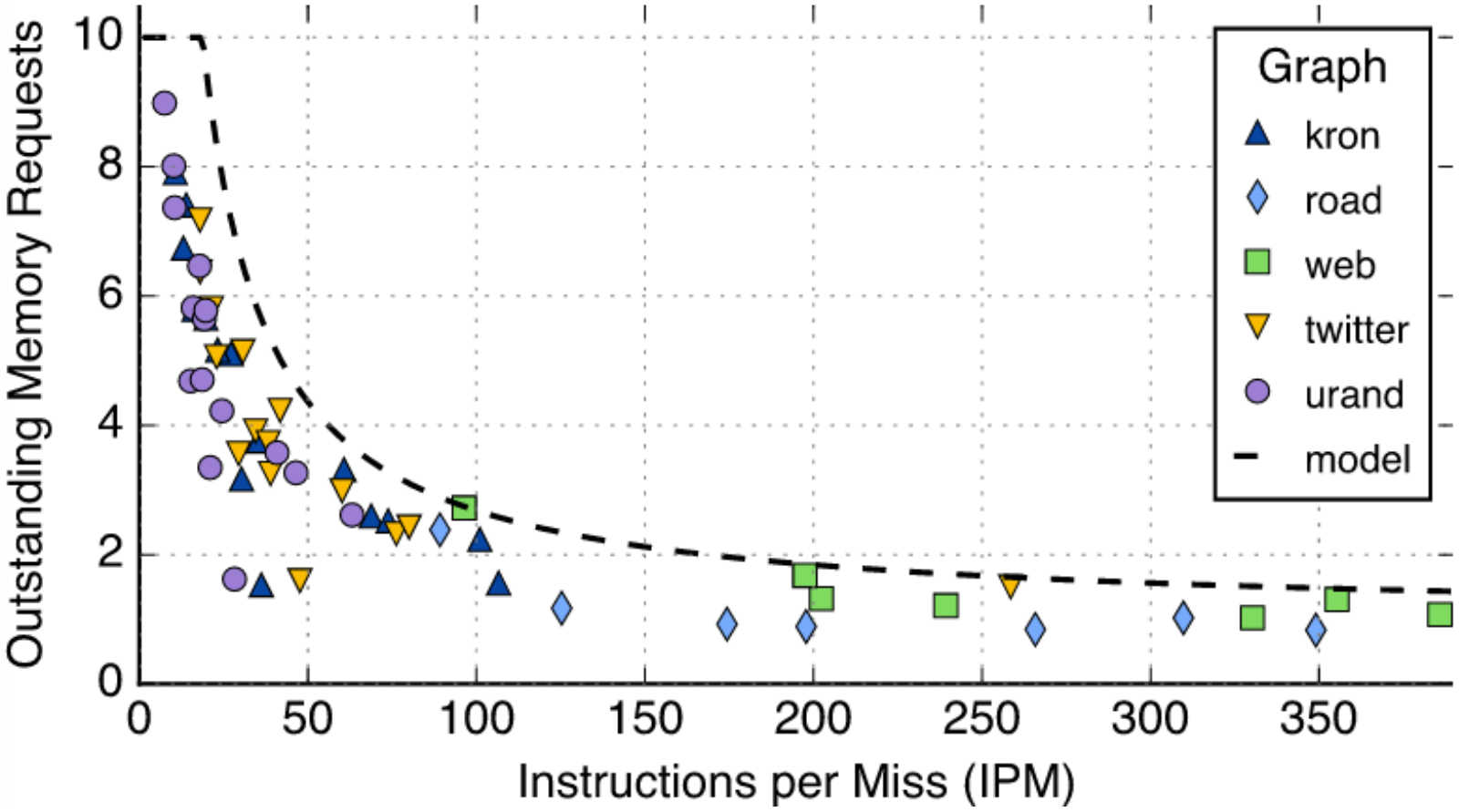


Memory Wall and Memory Bandwidth Wall

- Since the pins on a processor chip are expected to not increase much, we will hit a memory bandwidth wall
- Improvements in technology => DRAM capacities double every two years, but latency does not change much
- Power wall: 25-40% of datacenter power can be attributed to the DRAM system
- Latency and power can be both improved by increasing the row buffer hit rate; requires intelligent mapping of data to rows, clever scheduling of requests, etc.

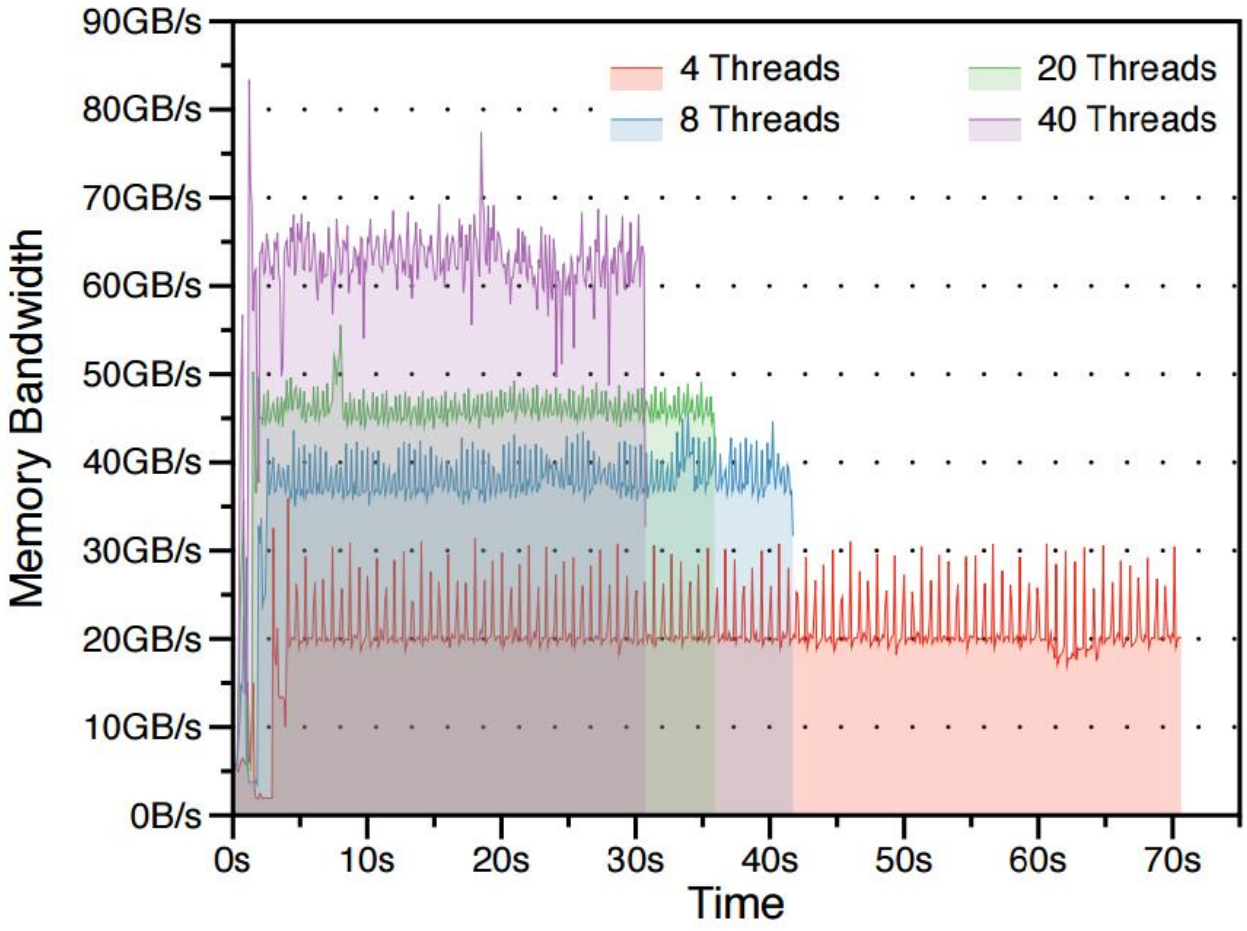
Discussion: MLP of Graph Workload

- Memory bandwidth utilization limited by high IPM.



Discussion: MLP of Graph Workload

- Growing memory utilization with more CPU cores.



Summary

- Memory hierarchy, uncore and off-chip
- Cache line, block, address
- Virtual memory, page table, PTE, TLB
- Locality principle, Inclusive and exclusive relationship
- Miss caching, victim caching, prefetching
- Cache write policies, write buffer/cache
- rank, bank, array, channel, MC, parallelism in DRAM
- 1T1C DRAM cell, data access, DRAM refresh
- DRAM access cost, synchronous/asynchronous design
- Memory design challenges, memory wall, MLP