CMPE110 Lecture 21
DRAM II

Heiner Litz

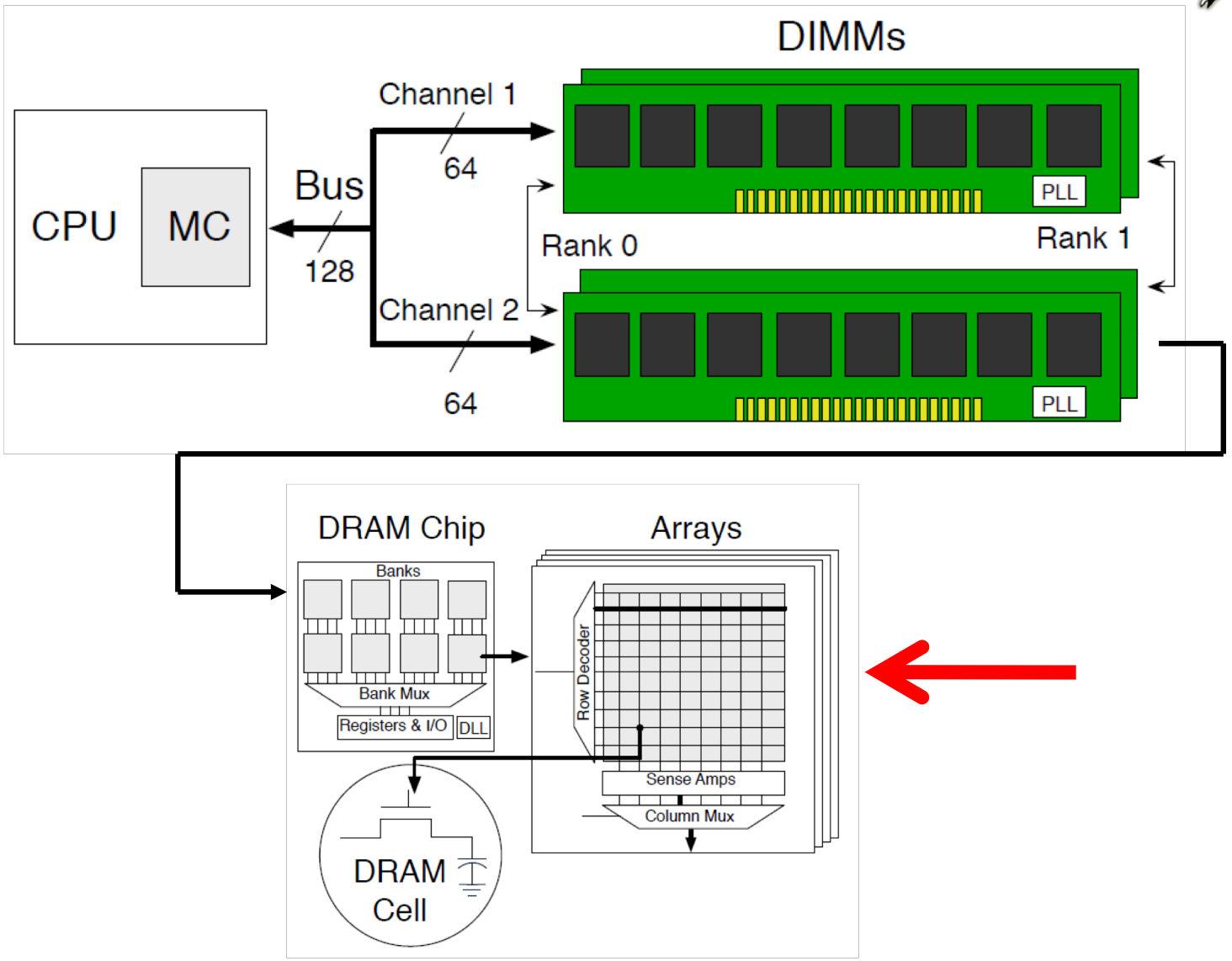
https://canvas.ucsc.edu/courses/19290

Announcements

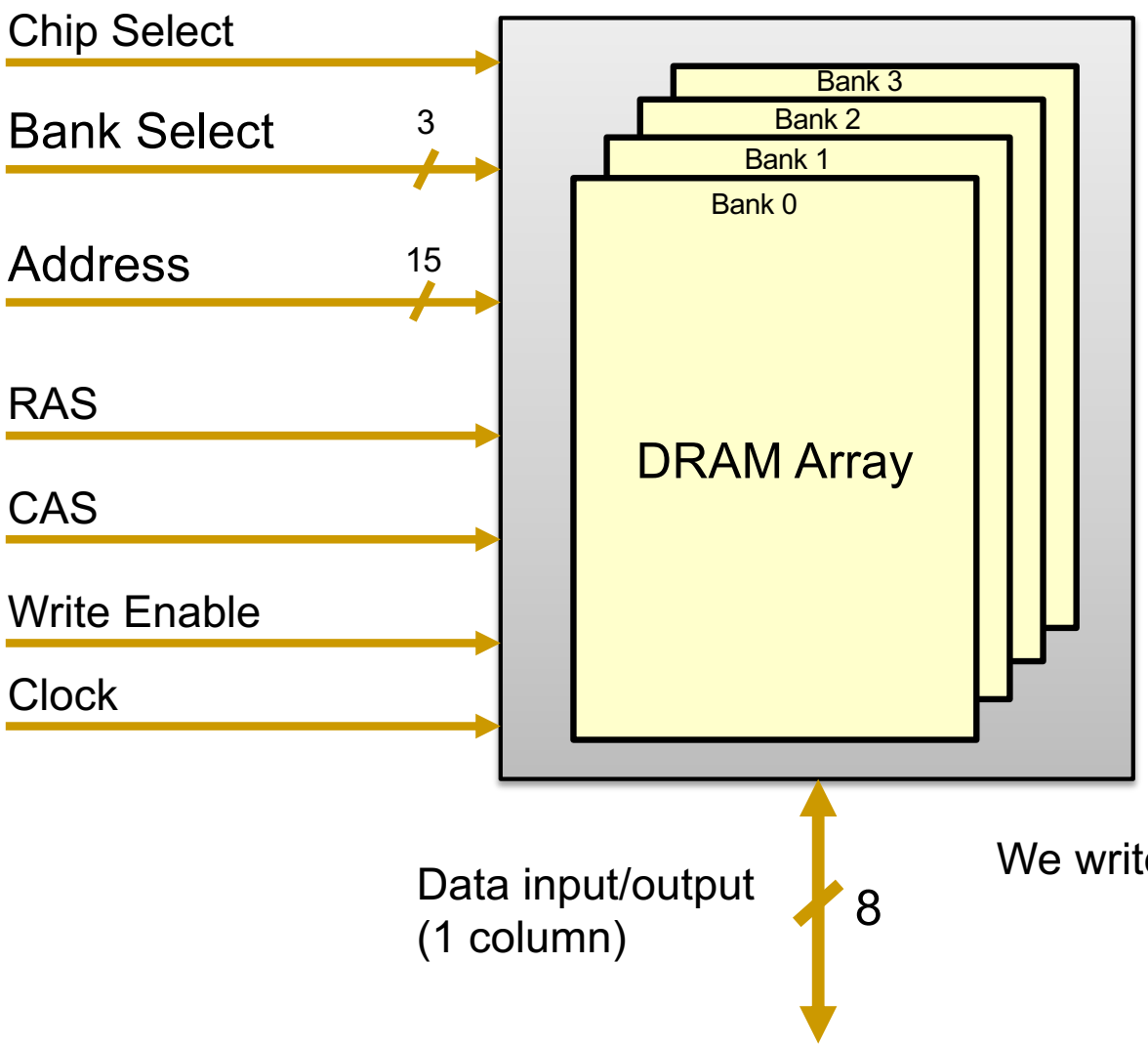
- Review Lecture on Monday March 11th
  - Given by Peter, Minghao, Saba
  - Propose Topics on Piazza now
  - If we have lots of topics, well poll and select on Friday

Review

Main Memory Overview

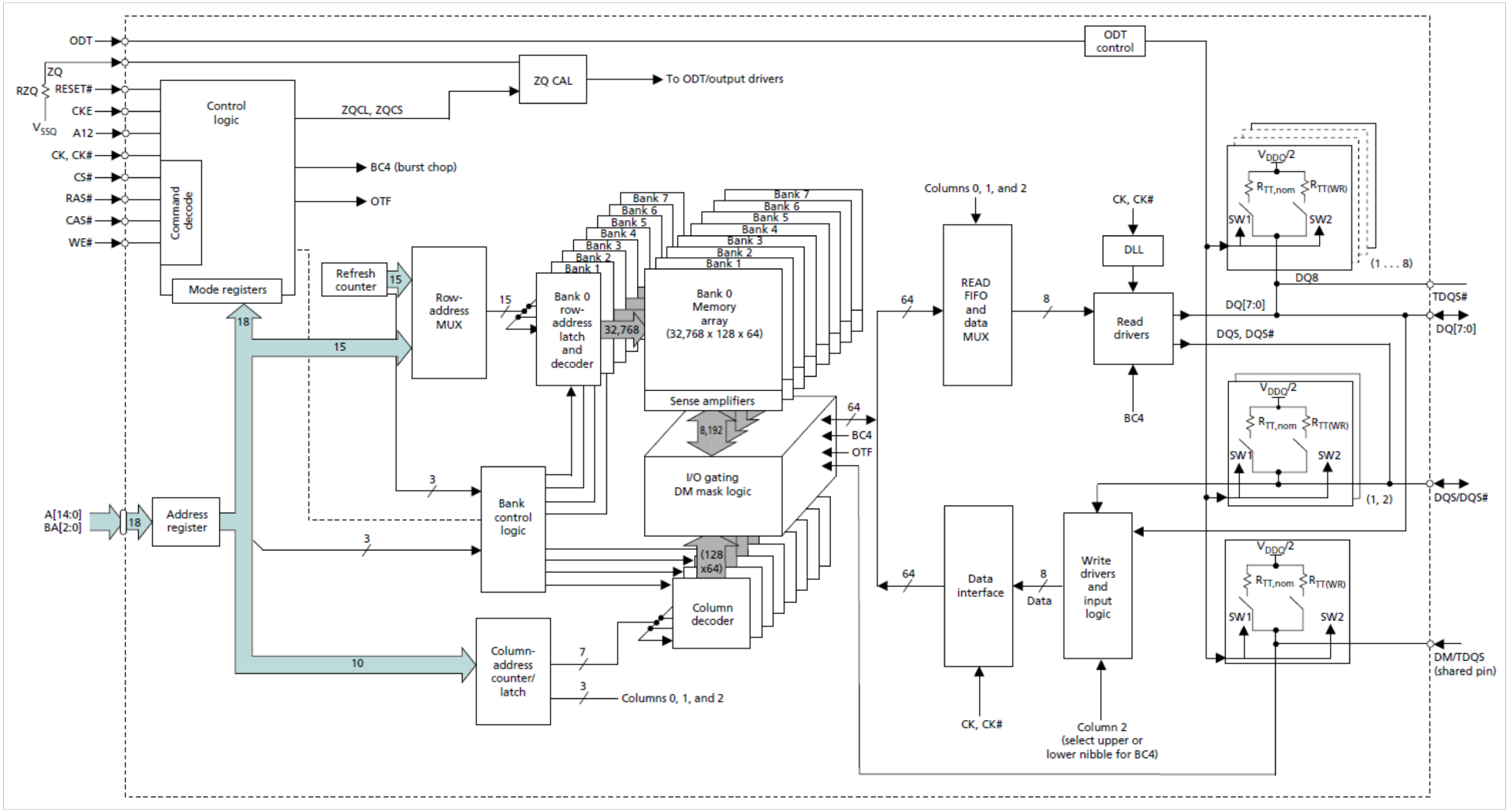


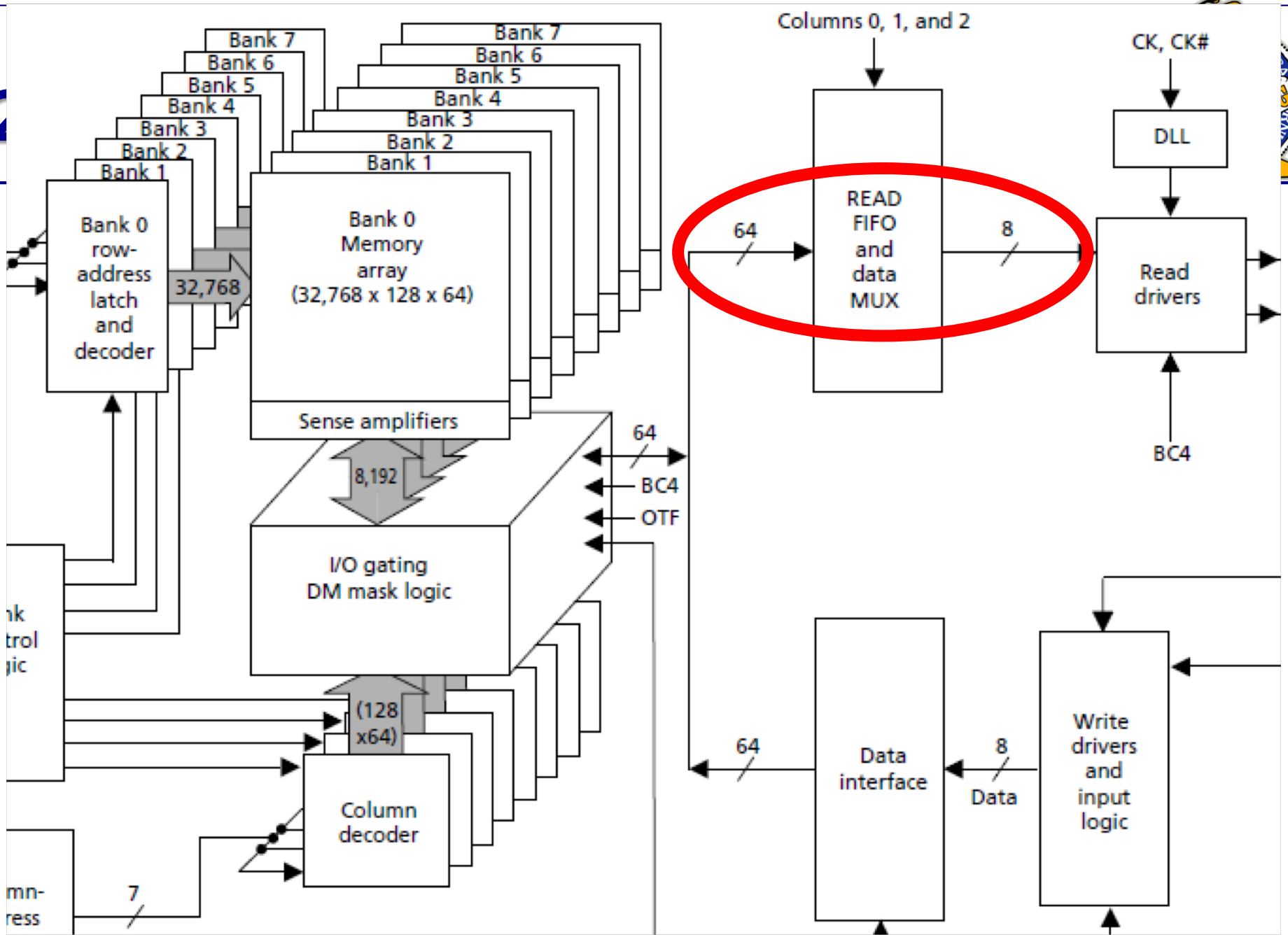
DRAM Chip: High-level View



We write “x8” to mean an 8 data bits,
x16 = 16 data bits, etc.

2Gb x8 DDR3 Chip [Micron]

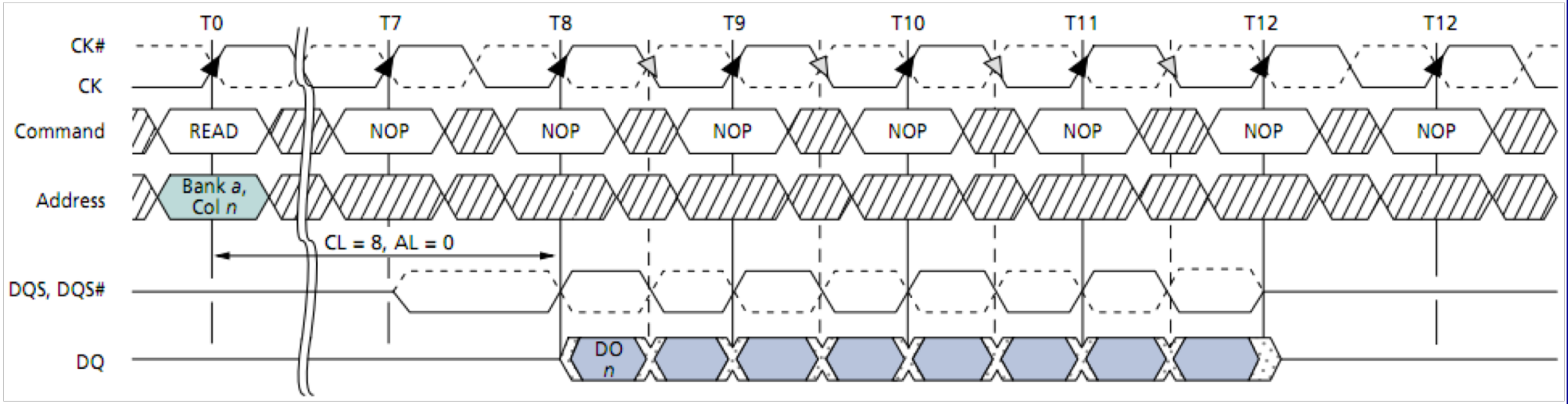




Observe: row width, 64 → 8 bit datapath

DRAM: Burst

- Each READ/WRITE command transfers multiple columns (8 in DDR3)
- DRAM channel clocked faster than DRAM core



- Critical word first?

DDR4 SDRAM: Current industry standard

- Introduced in 2014
- SDRAM = Synchronous DRAM = **Clocked**
- DDR = Double Data Rate
  - Data transferred on both clock edges
  - min 800 Mhz = 1600 MT/s
  - max 2133 MHz = 4266 MT/s
- x4, x8, x16 datapath widths
- Minimum burst length of 8
- 8 banks
- 8Gb, 16Gb, 32Gb, 64Gb capacity common
- Relative to SDR/DDR/DDR2/DDR3: + bandwidth, ~ latency

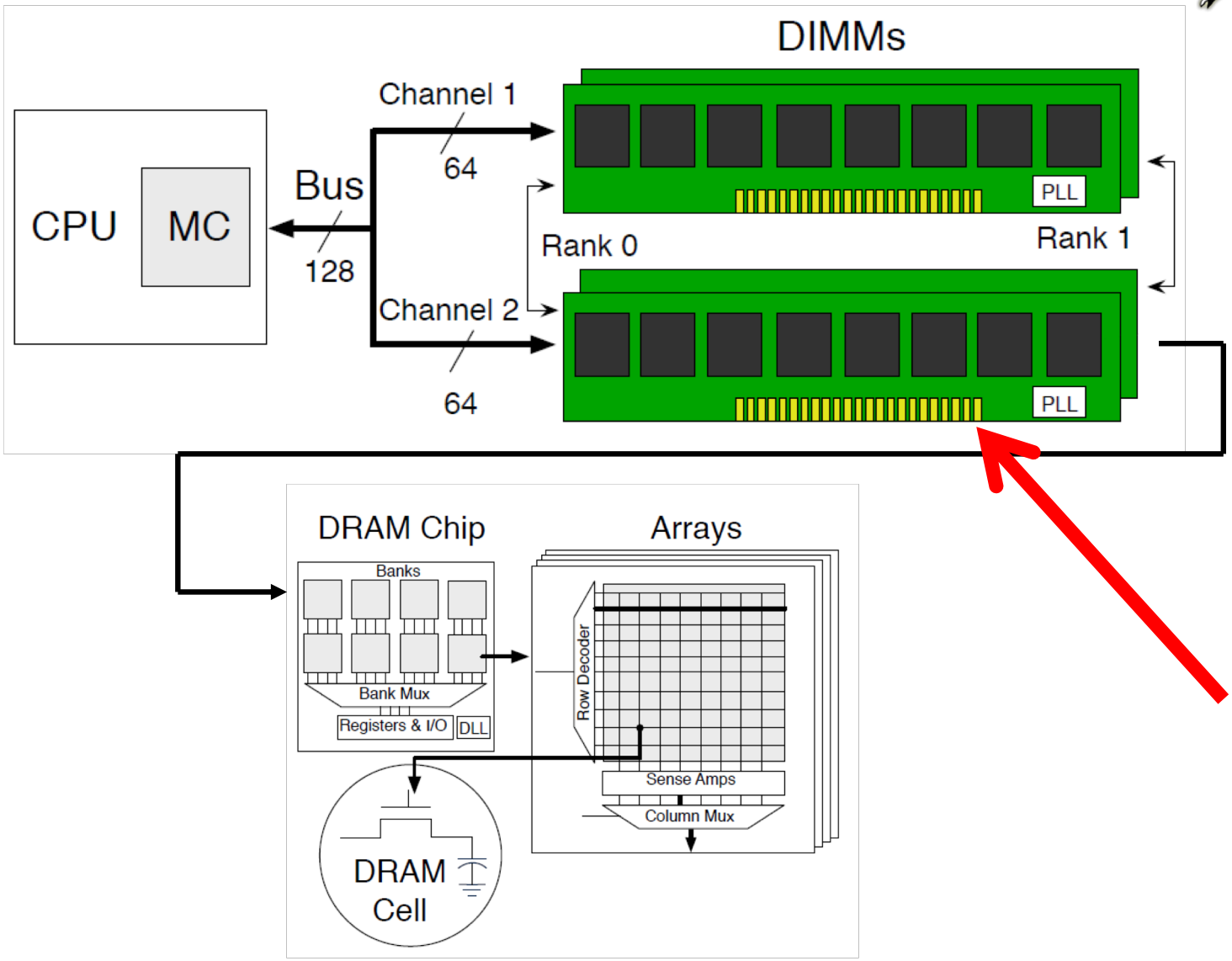
DRAM: Timing Constraints

- Memory controller must respect physical device characteristics
  - **tRCD** = Row to Column command delay
    - How long it takes row to get to sense amps
  - **tCAS** = Time between column command and data out
  - **tCCD** = Time between column commands
    - Rate that you can pipeline column commands
  - **tRP** = Time to precharge DRAM array
  - **tRAS** = Time between RAS and data restoration in DRAM array (minimum time a row must be open)
  - **tRC** = tRAS + tRP = Row "cycle" time
    - Minimum time between accesses to different rows

Latency Components: Basic DRAM Operation

- CPU → memory controller transfer time
- Controller latency
  - Queuing & scheduling delay at the controller
  - Access converted to basic commands
- DRAM bank latency
  - tCAS is row is "open" OR
  - tRCD + tCAS if array precharged OR
  - tRP + tRCD + tCAS (worst case: tRC + tRCD + tCAS)
- DRAM data transfer time
  - BurstLen / (MT/s)
- Memory controller → CPU transfer time

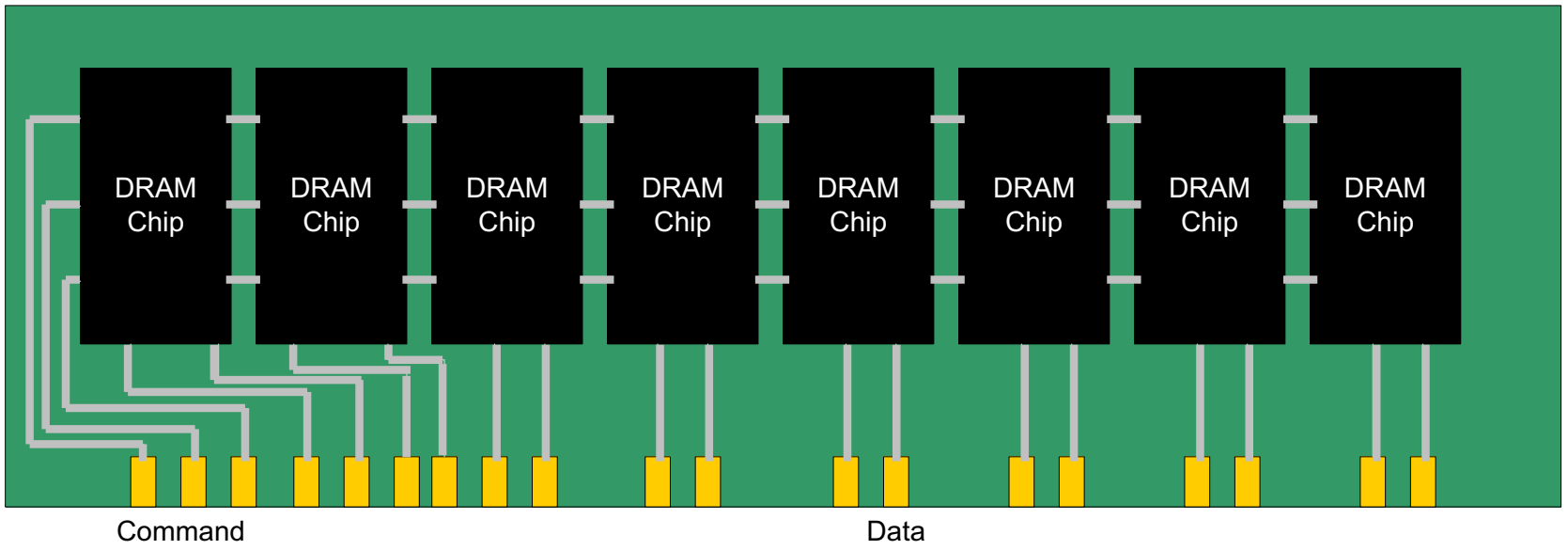
Main Memory Overview



DRAM Modules

- DRAM chips have narrow interface (typically x4, x8, x16)
- Multiple chips are put together to form a wide interface
  - DIMM: Dual Inline Memory Module
  - To get a 64-bit DIMM with x8 DRAM chips, we need to access 8 chips
  - Share command/address lines, but not data
- Advantages
  - Acts like a high-capacity DRAM chip with a wide interface
    - 8x capacity, 8x bandwidth, same latency
- Disadvantages
  - Granularity: Accesses cannot be smaller than the interface width
  - 8x power

A 64-bit Wide DIMM
(physical view)

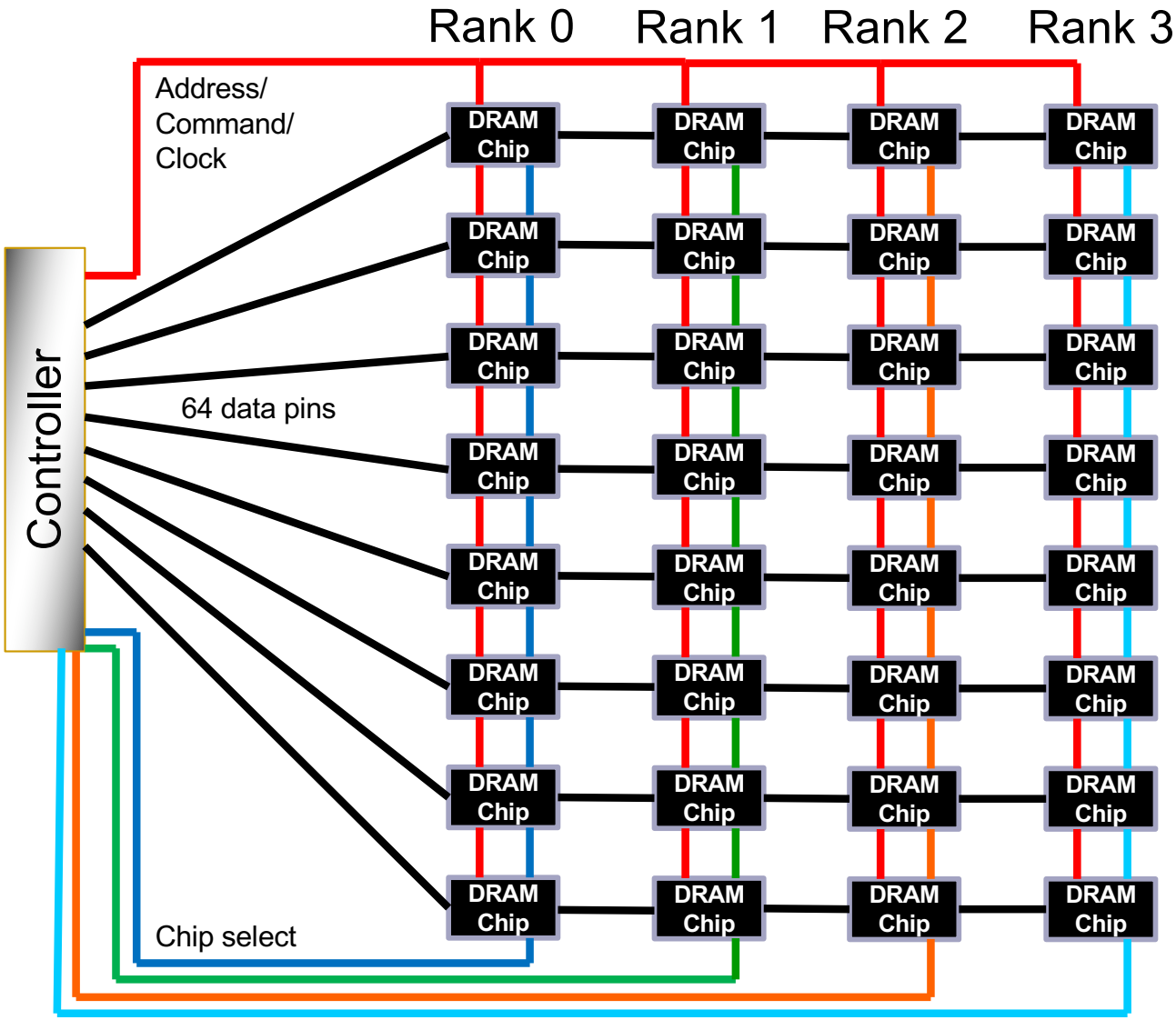


DIMM Capacity

- 64-bit interface
- 2Gb x8 chips
- 64/8 = 8 chips
- 2Gb x 8 = 2GB

- What if we want more capacity on a channel?
  - Option 1: Use narrower chips (i.e. x4)
  - Option 2: Ranks

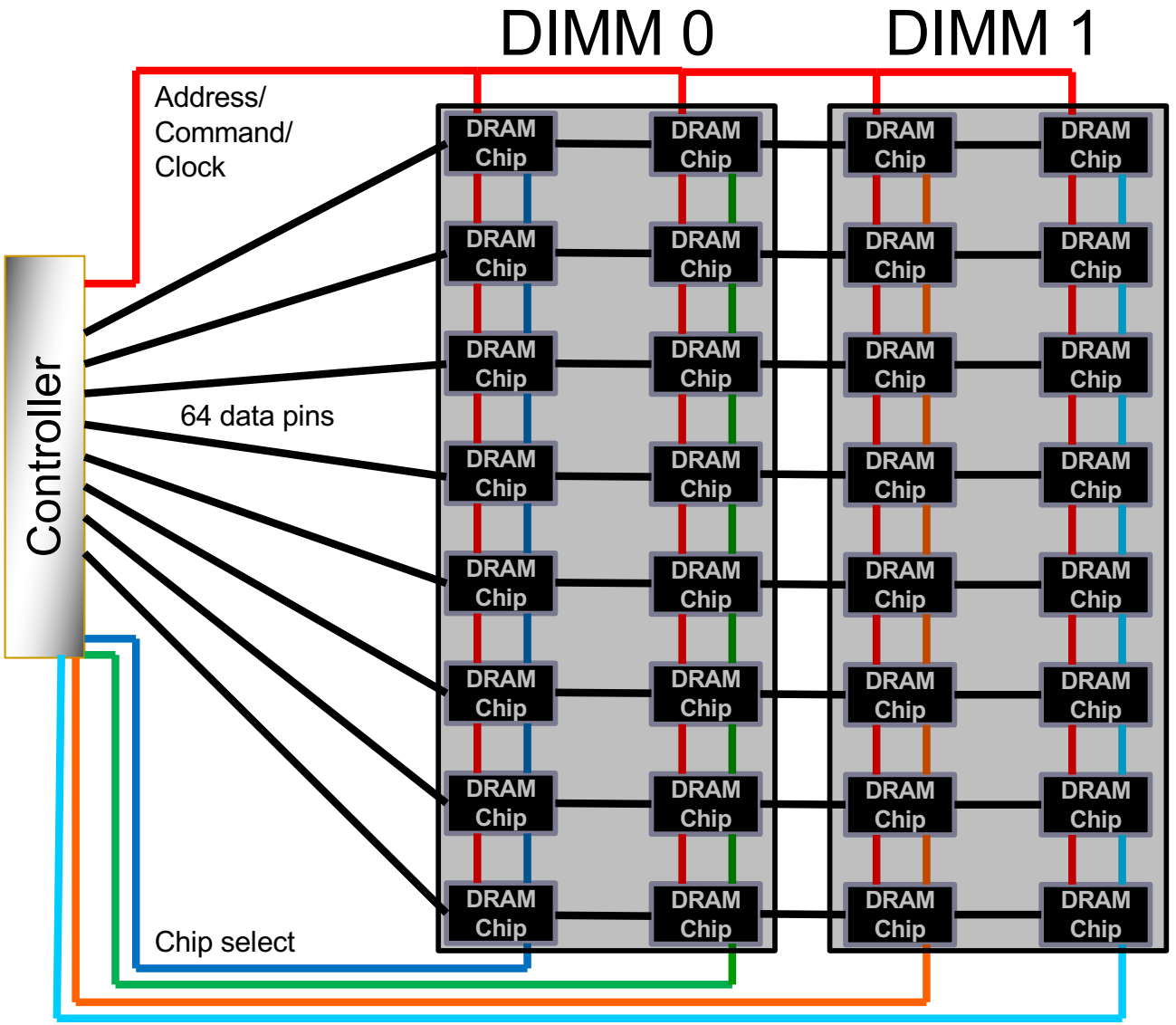
Ranks


Rank 0
Rank 1
Rank 2
Rank 3
Address/
Command/
Clock
Controller
64 data pins
Chip select
DRAM Chip

Ranks

- A DIMM may include multiple Ranks
  - A 64-bit DIMM with 16 chips with x8 interfaces has 2 ranks
- Each 64-bit group of chips is called a rank
  - All chips in a rank respond to a single command
  - Different ranks share command/address/data lines
    - Select between ranks with “Chip Select” signal
  - Ranks provide more “banks” across multiple chips (but don’t confuse rank and bank!)

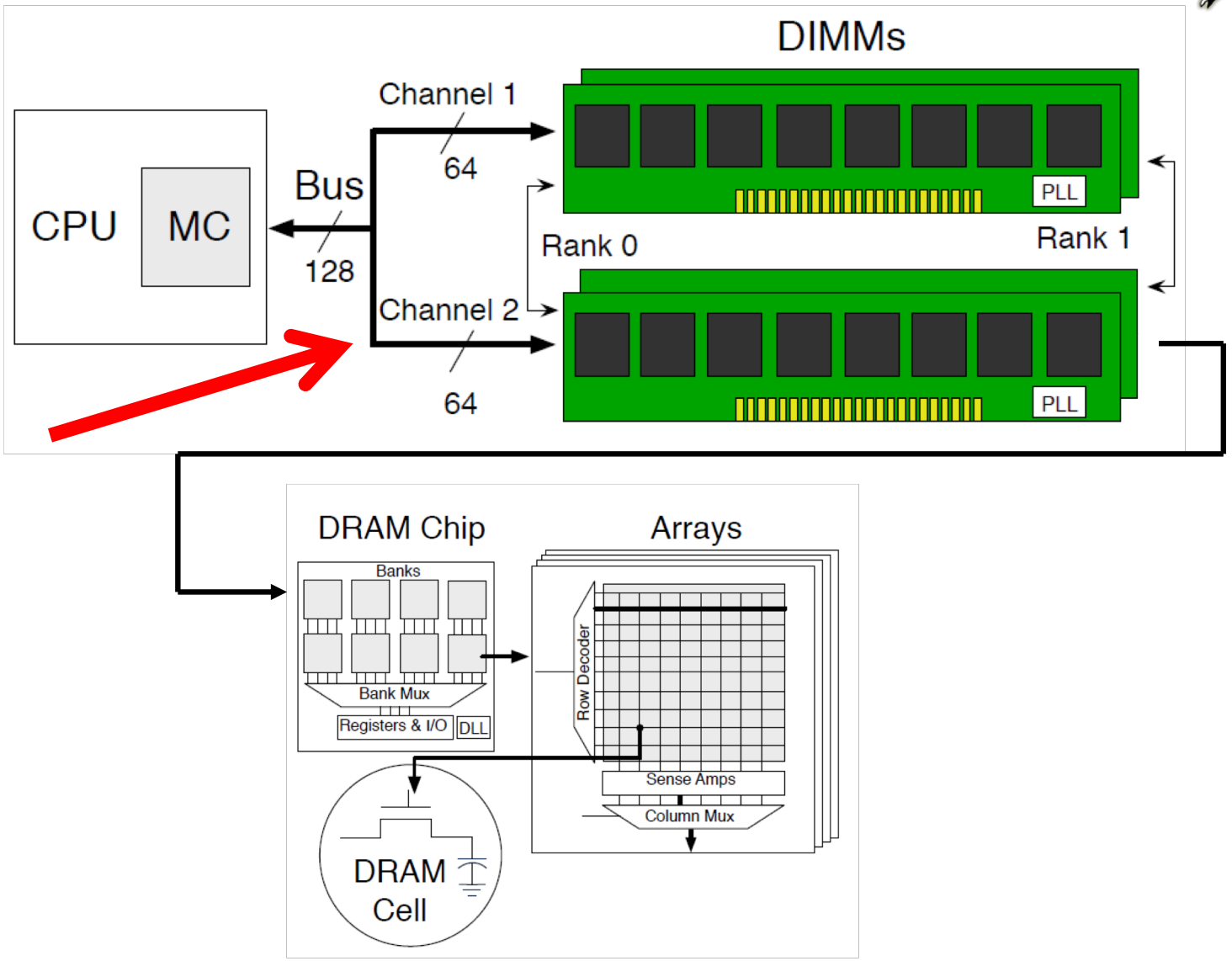
Multiple DIMMs on a Channel



DRAM Capacity

- What if we want even more capacity?
  - Can only put limited number of ranks on channel (signal and driver limitations)
- x2 chips? x1 chips?
  - What happens to power consumption?
- More channels?

Main Memory Overview

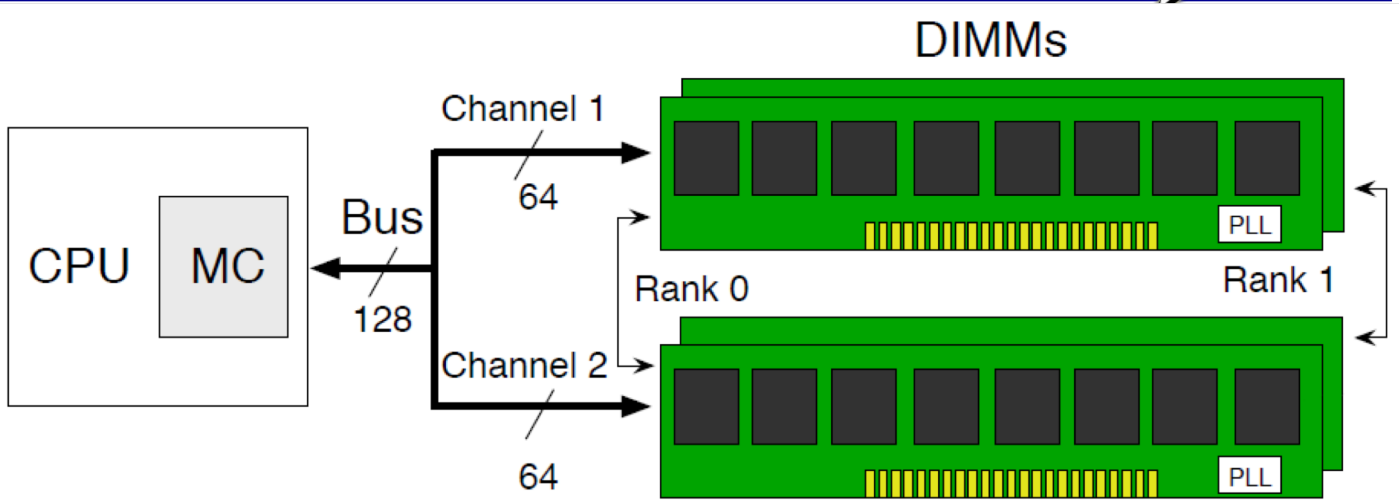


DRAM Channels

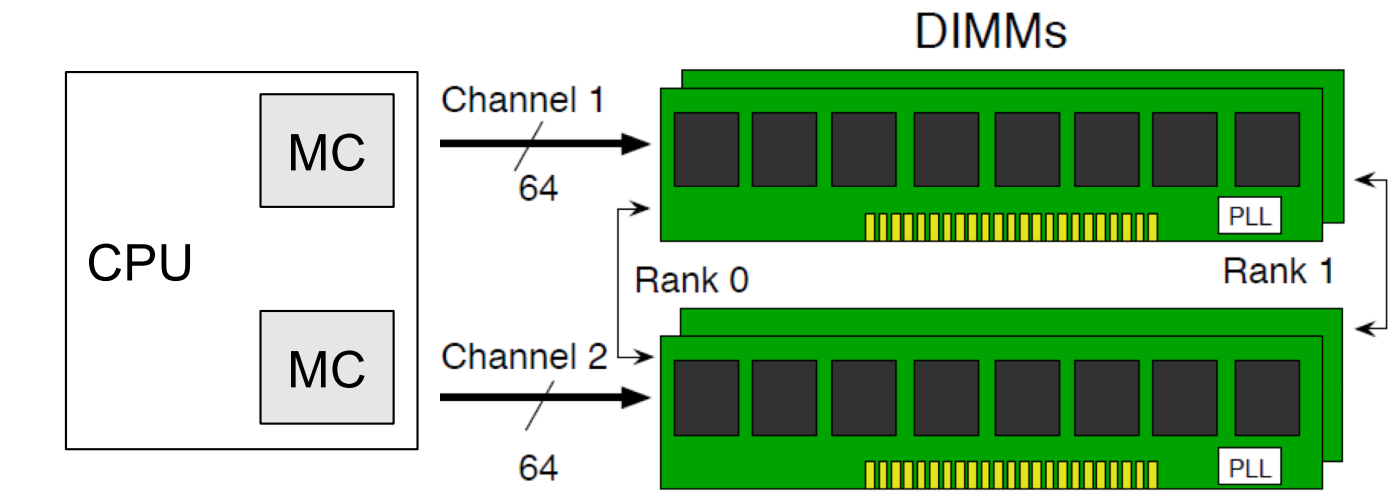
- Channel: a set of DIMMs in series
  - All DIMMs get the same command, one of the ranks replies
- Multiple-channel options
  - Multiple lock-step channels
    - Single “channel” with wider interface (faster cache line refill!)
    - Sometimes called “Gang Mode”
    - Only works if DIMMs are identical (organization, timing)
  - Multiple independent channels
    - Requires multiple controllers
- Tradeoffs in having multiple channels
  - Cost: pins, wires, controllers
  - Benefit: higher bandwidth, capacity

DRAM Channel Options

Lock-step



Independent

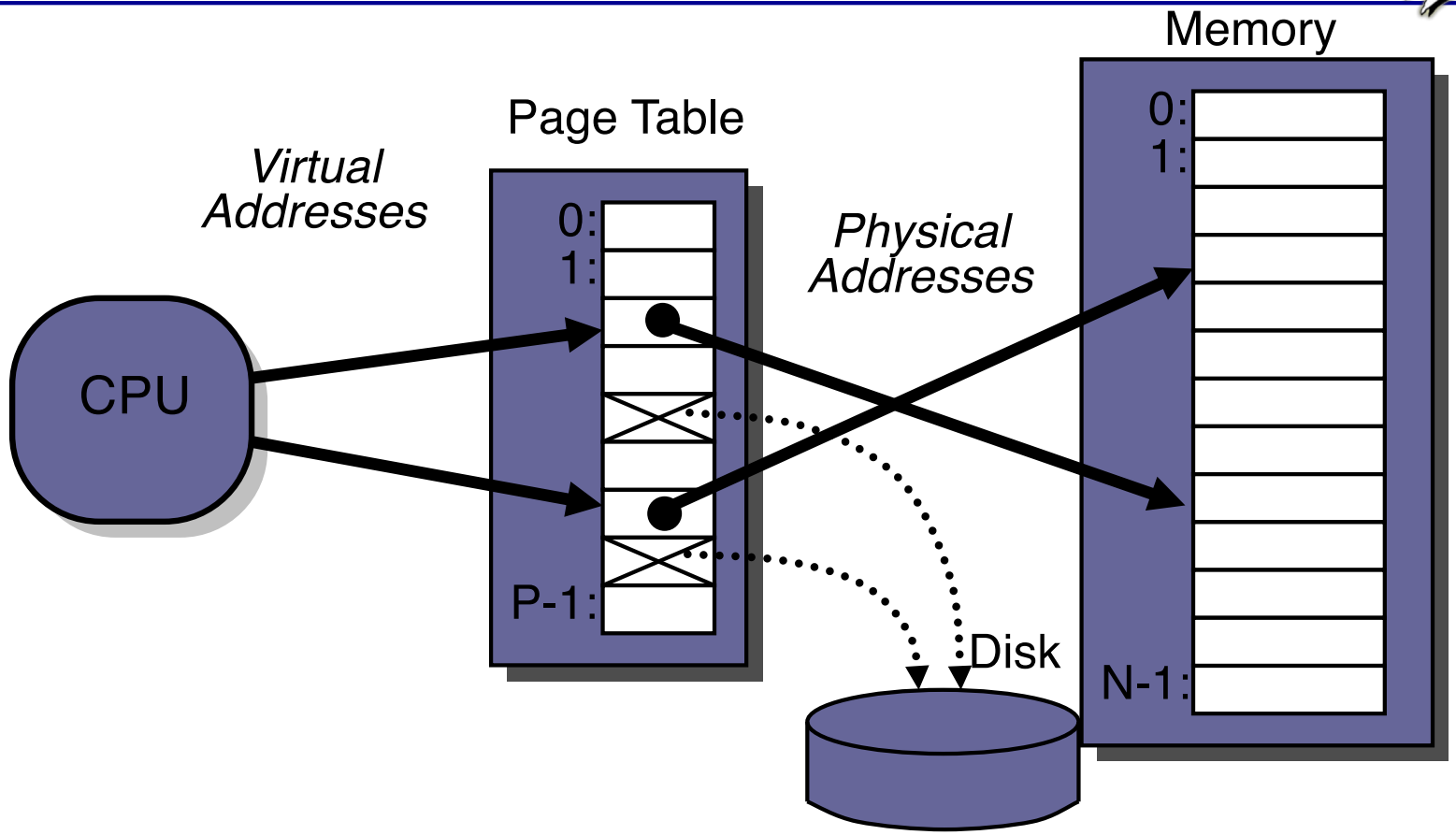


Virtual Memory

Virtual Memory

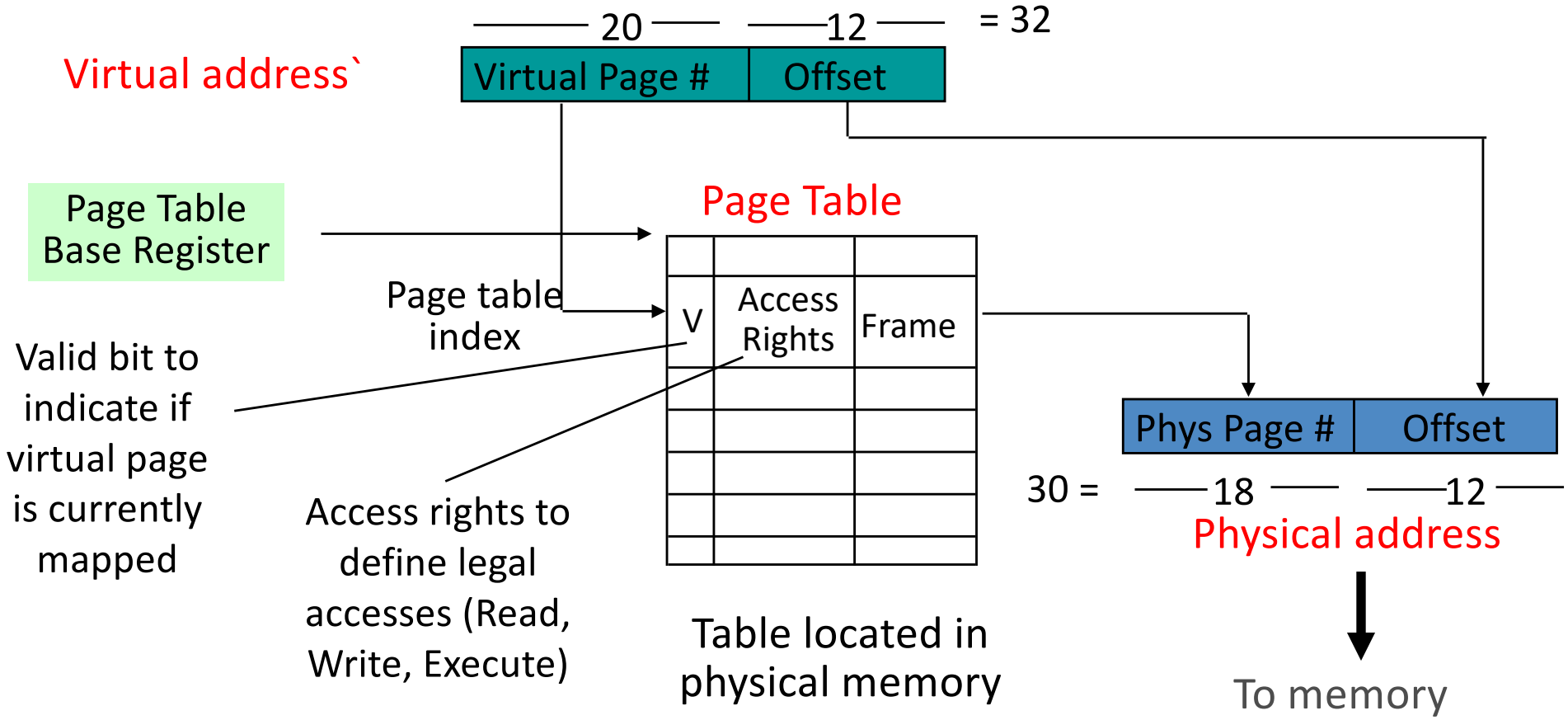
- How to share/use physical memory by multiple applications?
- How to guarantee that applications can only access “their” memory (security)
- How to expose virtually unlimited amount of main memory to applications?

Virtual Memory



- HW translates addresses using an OS-managed lookup table
- Indirection allows redirection and checks!

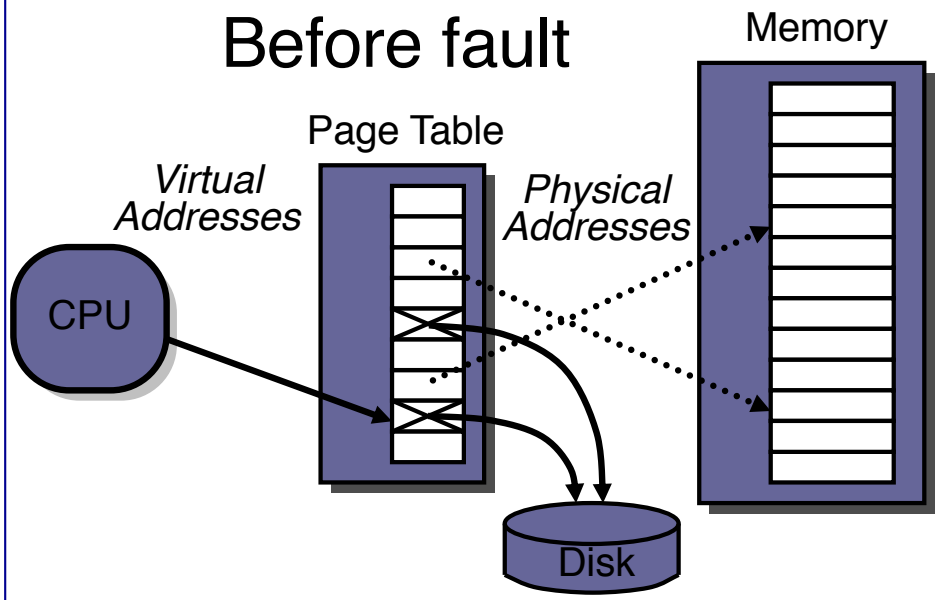
Page Table

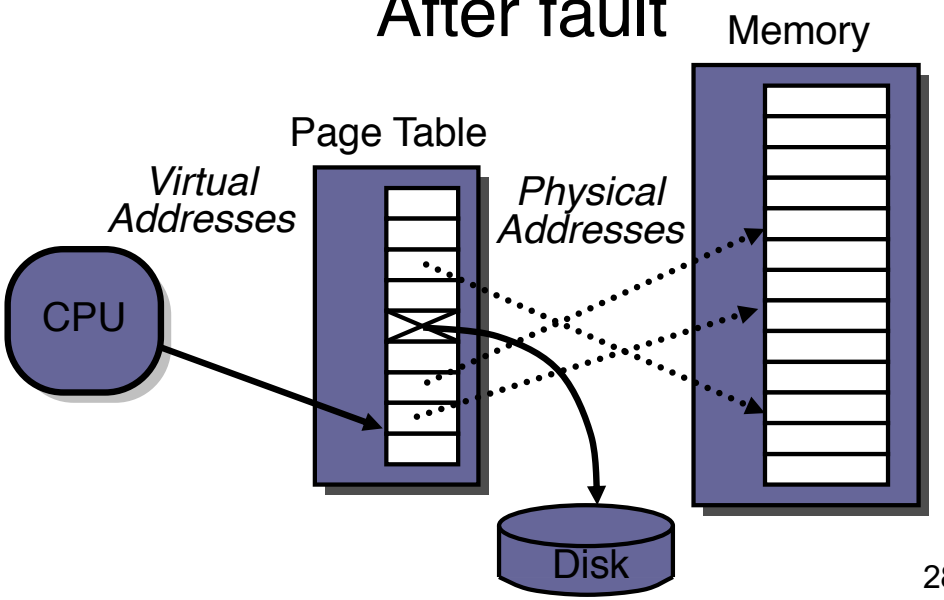


- One page table per process stored in memory
  - Process == instance of program
- One entry per virtual page number

Page Faults

- Page fault → an exception case
  - HW indicates exception cause and problematic address
  - OS handler invoked to move data from disk into memory
    - Current process suspends, others can resume
    - OS has full control over placement
  - When process resumes, repeat load or store





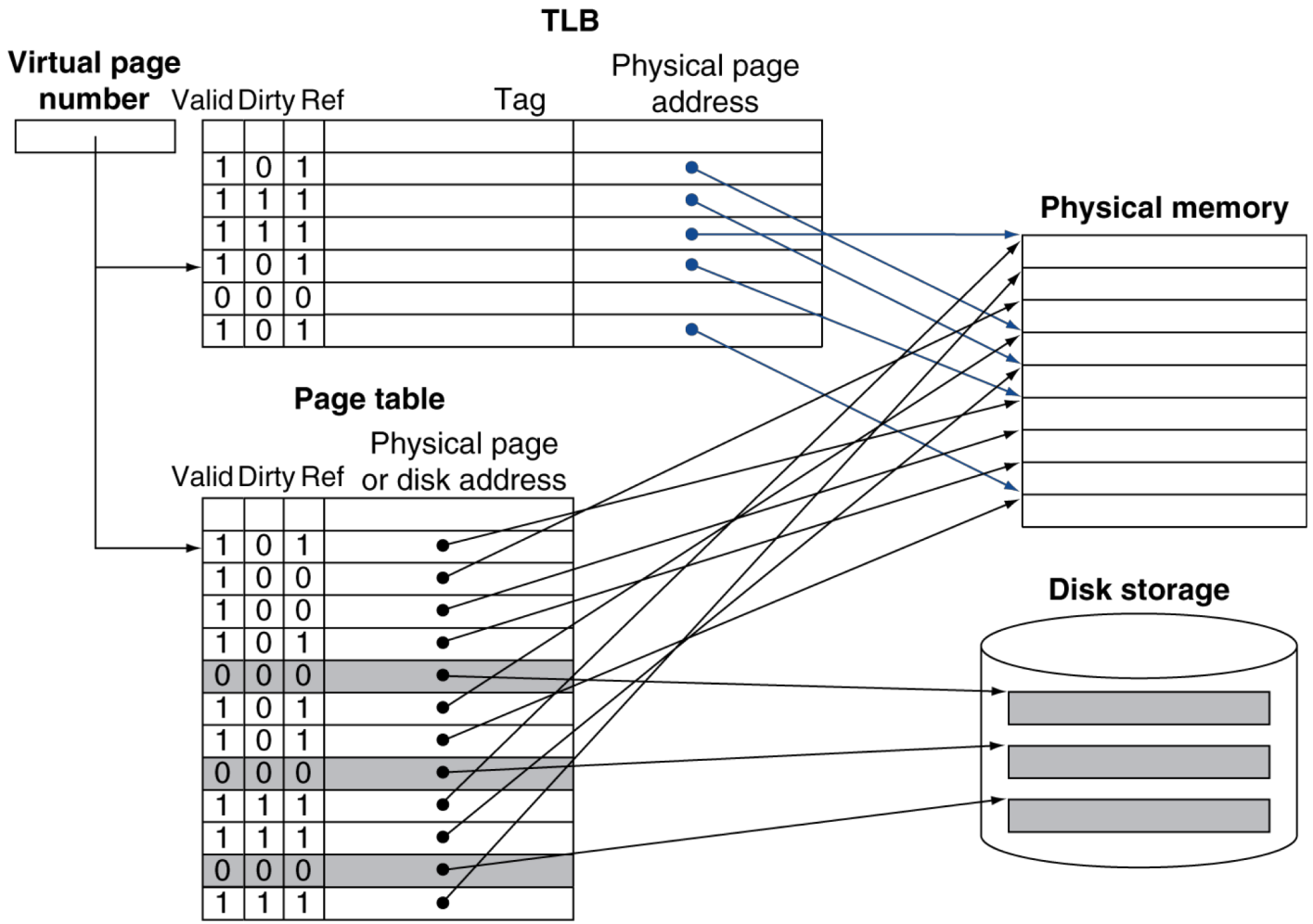
Wait, How About Performance?

- Virtual memory is great but
- We just doubled the memory accesses
  - A load requires an access to the page table first
  - Then an access to the actual data
- How can we do translation fast?
  - Without an additional memory access?

Translation Look-aside Buffer

- TLB = a hardware cache just for translation entries
  - A hardware cache specializing in page table entries
- Key idea: locality in accesses → locality in translations
- TLB design: similar issues to all caches
  - Basic parameters: capacity, associativity replacement policy
  - Basic optimizations: instruction/data TLBs, multi-level TLBs, …
  - Misses may be handled by HW or SW
    - x86: hardware services misses
    - MIPS: software services misses through exception

TLB Organization

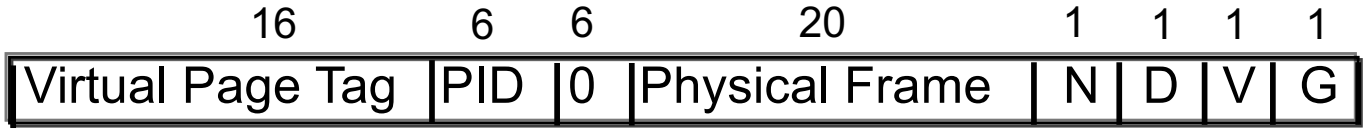


TLB Entries

- Each TLB entry stores a page table entry (PTE)
- TLB entry data → PTE entry fields
  - Physical page numbers
  - Permission bits (RXW)
  - Other PTE information (dirty bit, etc)
- The TLB entry metadata
  - Tag: portion of virtual page # not used to index the TLB
    - Depends on the TLB associativity
  - Valid bit
  - LRU bits
    - If TLB is associative and LRU replacement is used

Example TLB

- TLB entry design
  - Addresses are 32 bits with 4 KB pages (12 bit offset)
  - TLB has 64 entries, 4-way set associative
  - Each entry is 42 bits wide:



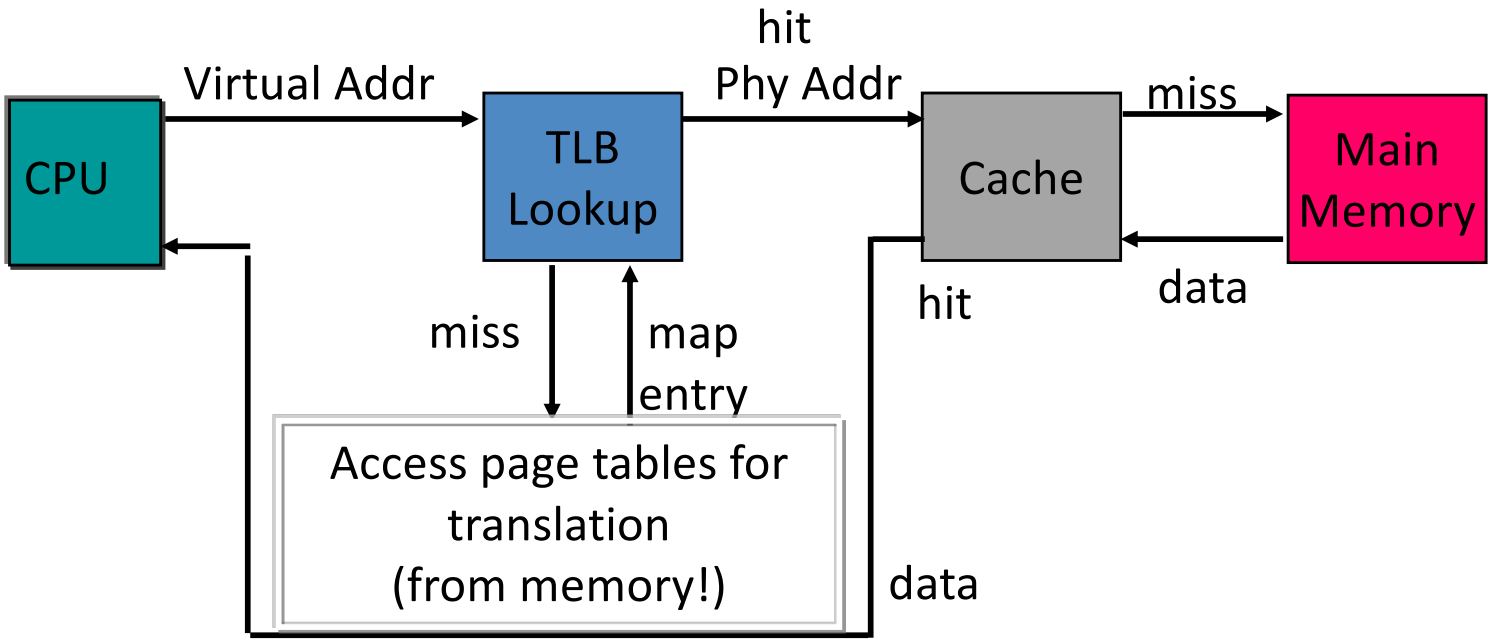
PID	Process ID
N	Do not cache memory address (optional)
D	Dirty bit
V	Valid bit
G	Global (valid regardless of PID)

TLB Caveats: Context Switching

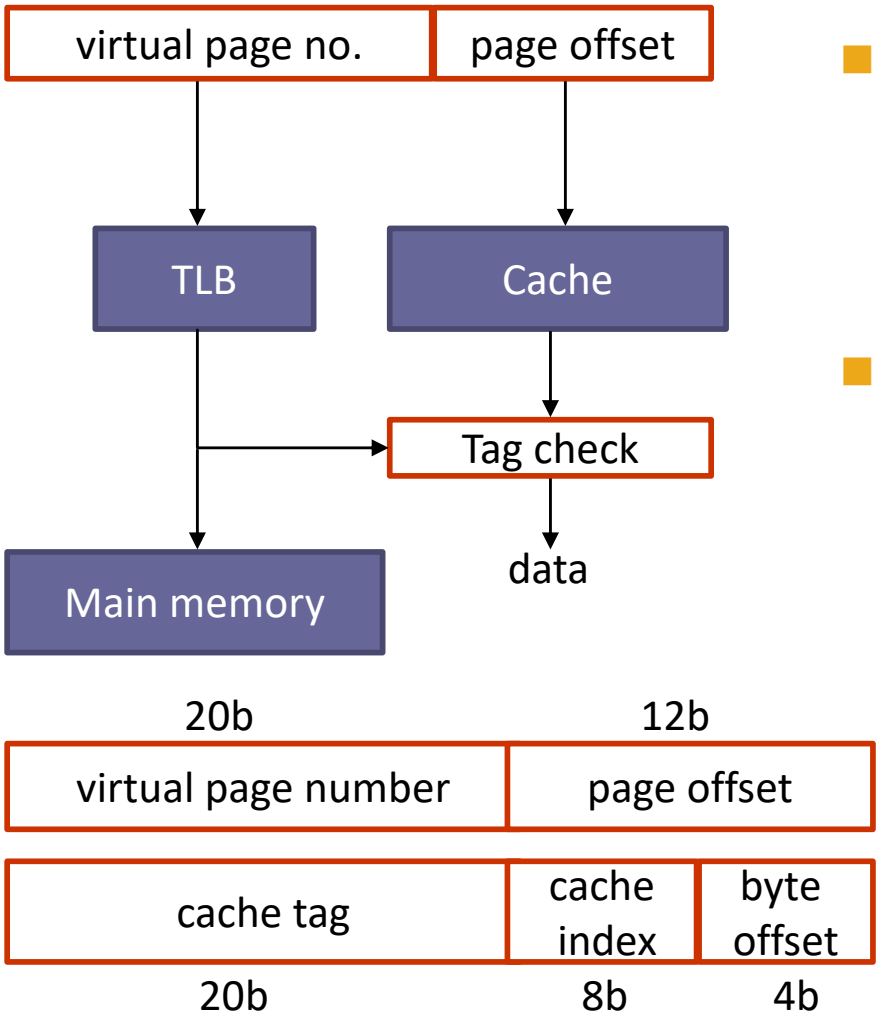
- What happens to TLB when switching between processes?
- The OS must flush the entries in the TLB
  - Large number of TLB misses after every switch
- Alternatively, use a process ID each TLB entry
  - PID or ASID
  - Allows entries from multiple programs to co-exist
  - Gradual replacement

TLB & Memory Hierarchies

- Basic process
  - Use TLB to get VA → PA
  - Use PA to access caches and DRAM
- Question: can you ever access the TLB and the cache in parallel?



Virtually Indexed, Physically Tagged Caches



- Translation & cache access in parallel
  - Start access to cache with page offset
  - Tag check used physical address
- Only works when
  - VPN bits not needed for cache lookup
  - Cache Size ≤ Page Size * Associativity
    - I.e. Set Size ≤ Page Size
  - Ok, we want L1 to be small anyway