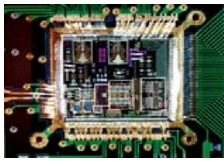
Lecture 7: Memory

CSCE5610 Computer System Architecture
CSCE4610 Computer Architecture

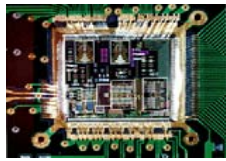
Instructor: Saraju P. Mohanty, Ph. D.

NOTE: The figures, text etc included in slides are borrowed from various books, websites, authors pages, and other sources for academic purpose only. The instructor does not claim any originality.



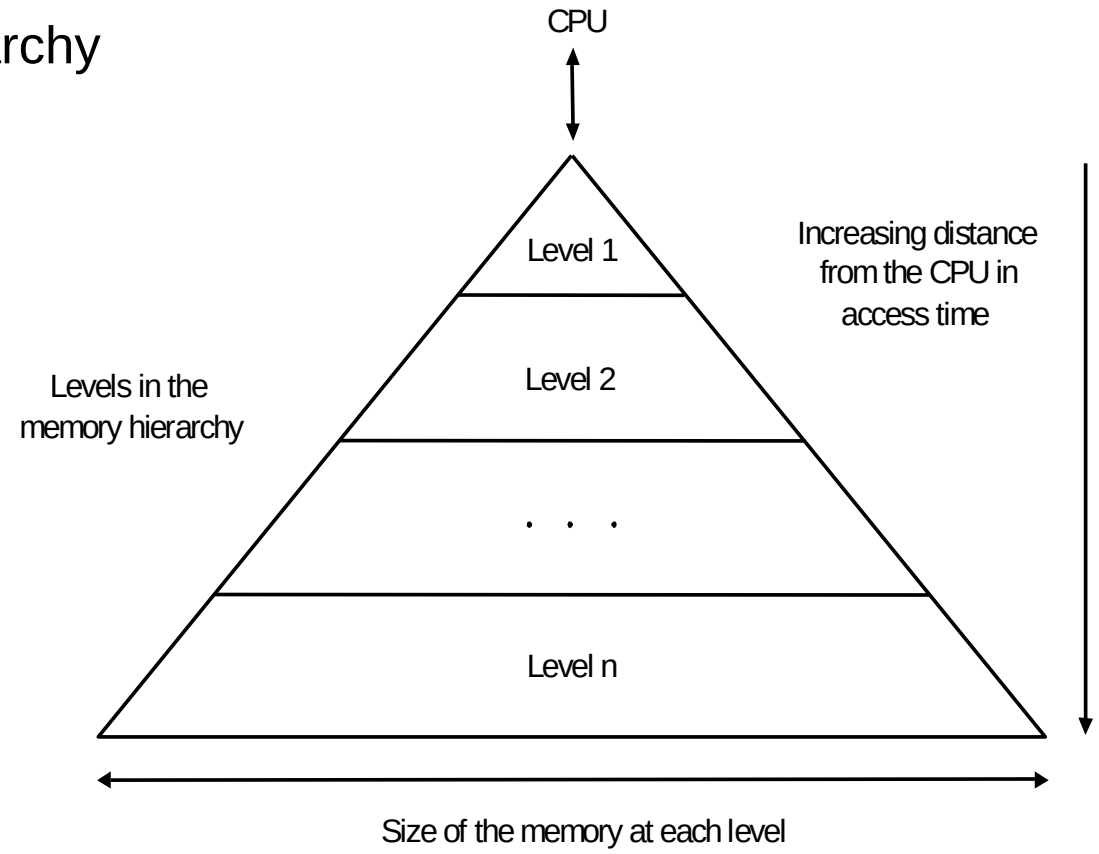
So far …

- We have..
  - Designed Instruction set – MIPS subset
  - Understood what performance means
  - Designed Datapath
    - Single cycle implementation
    - Multi-cycle implementation
  - Designed Controllers for single/multi-cycle implementations
  - In the relentless quest for performance, we explored pipelining
    - Hazards (Data/Control/Structural) need to be handled
- Now, we are ready to study the third component of computer: MEMORY!

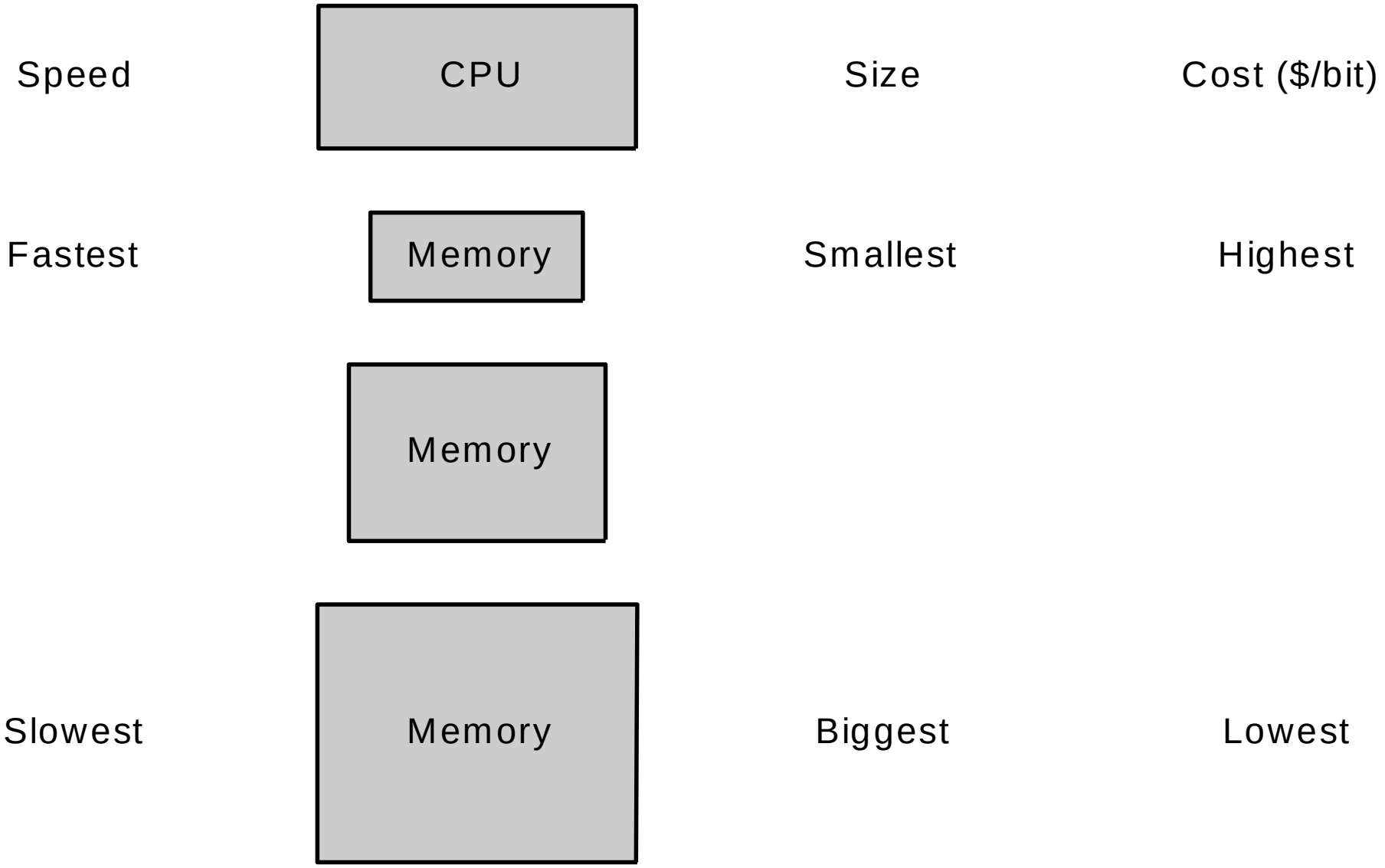


Exploiting Memory Hierarchy

- Users want large and fast memories! **1997**
  - SRAM access times are 2 - 25ns at cost of $100 to $250 per Mbyte.
  - DRAM access times are 60-120ns at cost of $5 to $10 per Mbyte.
  - Disk access times are 10 to 20 million ns at cost of $.10 to $.20 per Mbyte.
- Try and give it to them anyway
  - build a memory hierarchy



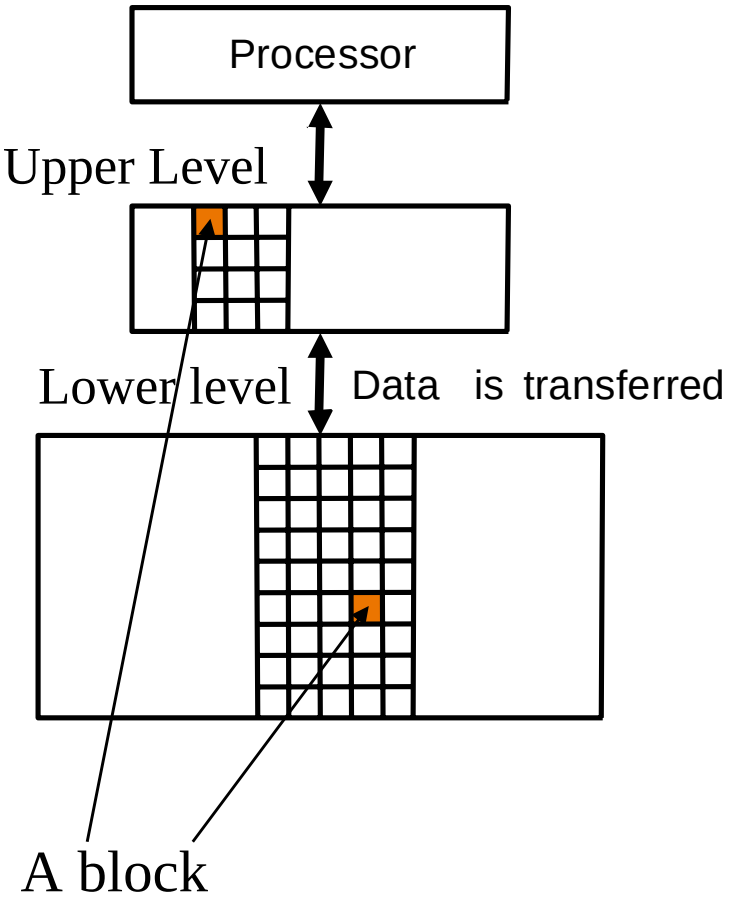
Basic Structure of Memory Hierarchy



Locality Principles

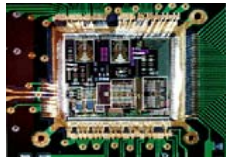
- A principle that makes having memory hierarchy a good idea.
- If an item is referenced, *temporal locality:* it will tend to be referenced again soon.

  spatial locality: nearby items will tend to be referenced soon.
- Our initial focus: two levels (upper, lower)
  - block: minimum unit of data
  - hit: data requested is in upper level
  - miss: data requested is not in the upper level



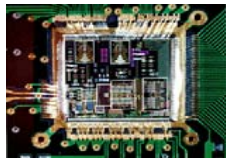
Memory Hierarchy - Buzzwords

- block: minimum unit of data that is transferred.
- hit: data requested is in the upper level.
- miss: data requested is not in the upper level.
- Hit rate (or hit ratio): is the fraction of memory accesses found in the upper level.
- Miss rate: is the fraction of memory accesses NOT found in upper level.
- Miss rate = (1 – hit rate).
- Hit time: time to access the upper level of memory hierarchy which includes the time needed to determine whether the access is a hit or a miss.
- Miss penalty: time to replace the block in the upper level with the corresponding block at the lower level plus the time to deliver this block to the processor.

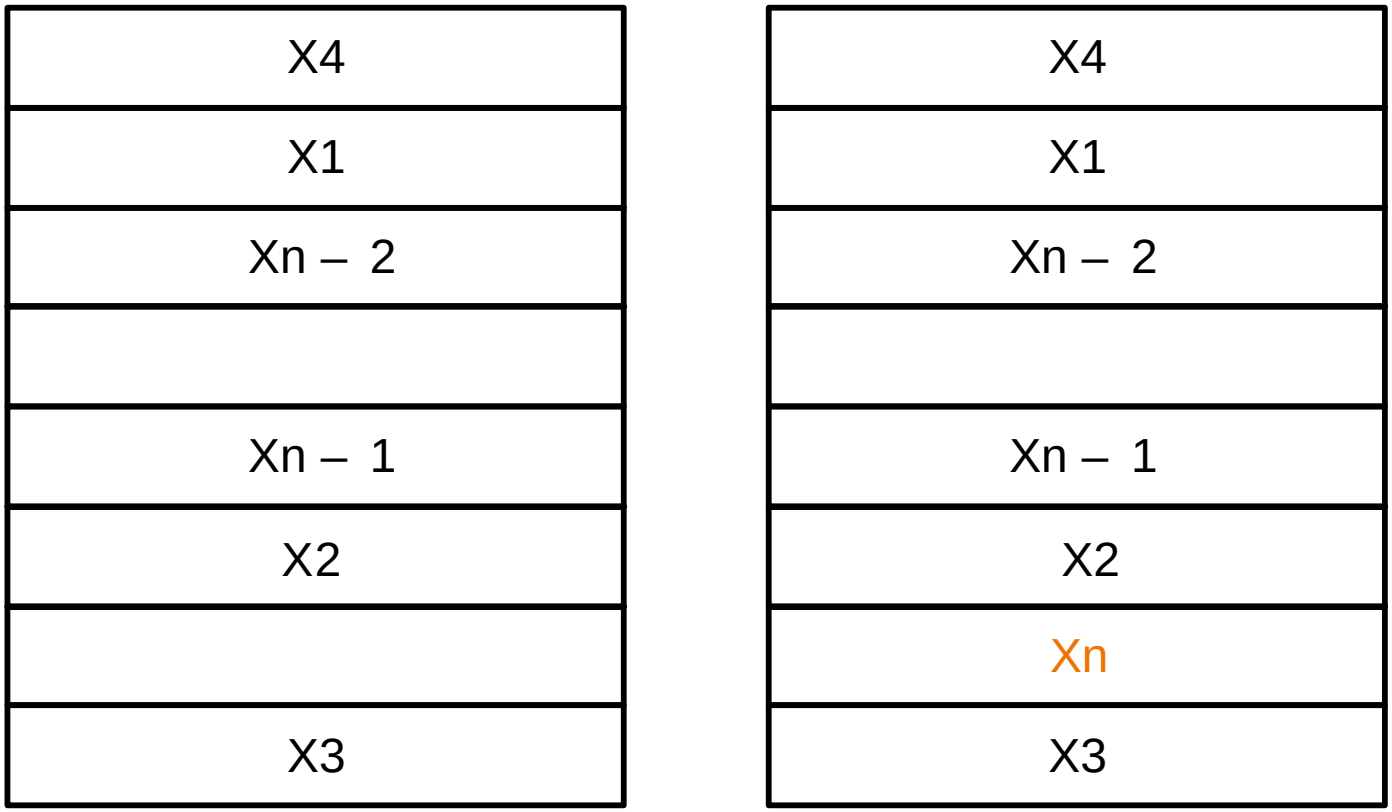


Cache

- Cache: a safe place for hiding or storing things.
- Cache – name chosen to represent the level of the memory hierarchy between the CPU and main memory.
- Two issues:
  - How do we know if a data item is in the cache?
  - If it is, how do we find it?
- Our first example:
  - block size is one word of data
  - "direct mapped"



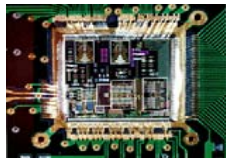
Reference to a missing item..



a. Before the reference to Xn

b. After the reference to Xn

The cache just before and just after a reference to a word X_n that is not initially in the cache.



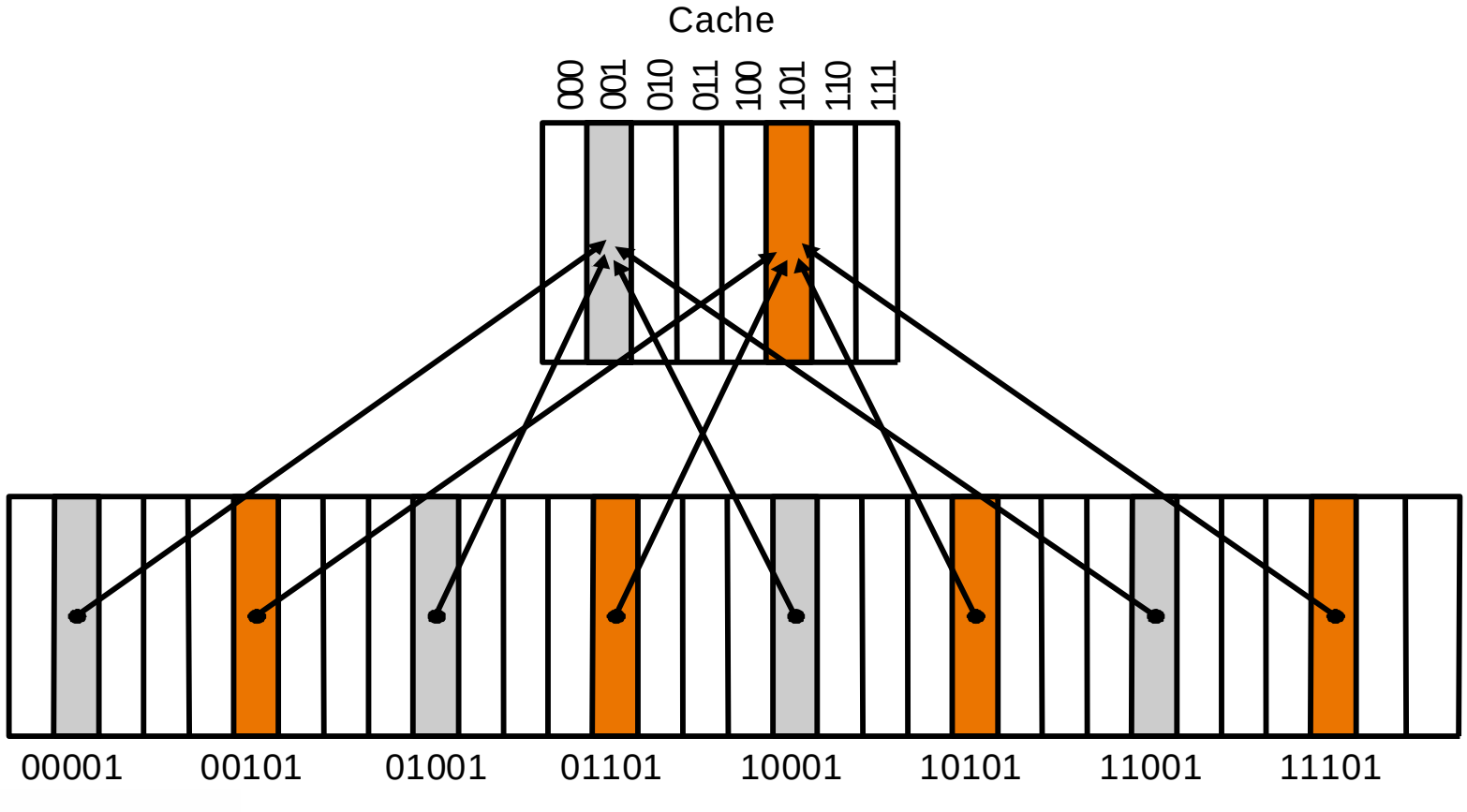
Direct Mapped Cache

- For each item of data at the lower level, there is exactly one location in the cache where it might be.

  e.g., lots of items at the lower level share locations in the upper level.

- Mapping: (Block address) modulo (Number of cache blocks in the cache. e.g. block address (01101) = 13 maps to

  (13 mod 8) = cache block 5 = (101)



Accessing a Cache: An example (page – 477 Interface Book)

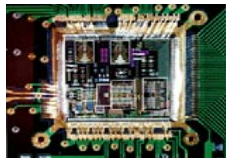
Initial state of the cache after power ON

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

After handling a miss of address (10110)

Index	V	Tag	Data
000			
001			
010			
011			
100			
101			
110	Y	10	Mem(10110)
111			

- Cache is empty
- Valid bits (V) are OFF (N)



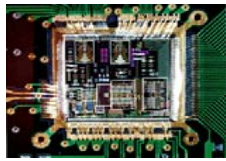
Example Contd.,

Index	V	Tag	Data
000			
001			
010	Y	11	Mem(11010)
011			
100			
101			
110	Y	10	Mem(10110)
111			

- After handling a miss of address (11010)
- After handling a hit of address (10110)
- After handling a hit of address (11010)

Index	V	Tag	Data
000	Y	10	Mem(10000)
001			
010	Y	11	Mem(11010)
011			
100			
101			
110	Y	10	Mem(10110)
111			

- After handling a miss of address (10000)



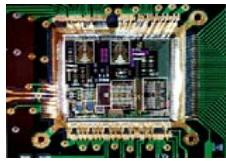
Example Contd.,

Index	V	Tag	Data
000	Y	10	Mem(10000)
001			
010	Y	11	Mem(11010)
011	Y	00	Mem(00011)
100			
101			
110	Y	10	Mem(10110)
111			

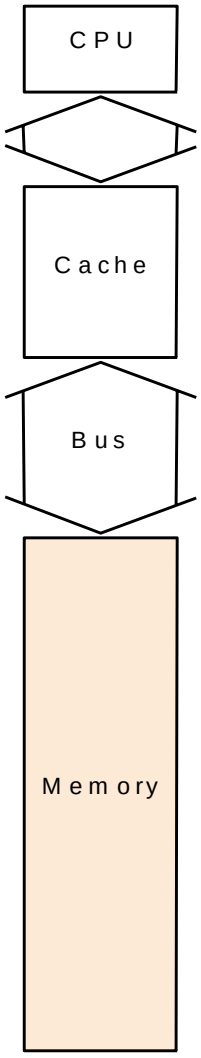
- After handling a miss of address (00011)
- After handling a hit of address (10000)

Index	V	Tag	Data
000	Y	10	Mem(10000)
001			
010	Y	10	Mem(10010)
011	Y	00	Mem(00011)
100			
101			
110	Y	10	Mem(10110)
111			

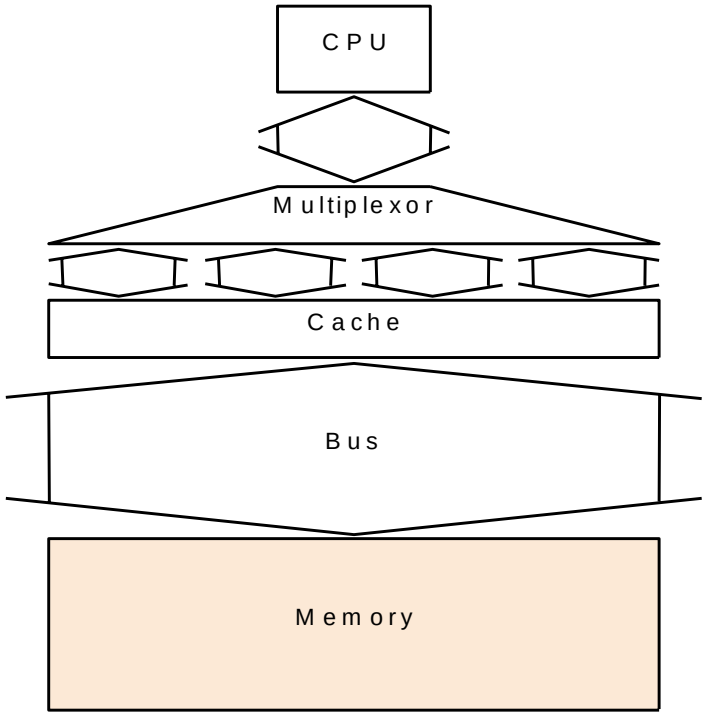
- After handling a miss of address (10010)
- (overwrites previous contents, Mem(11010) at location (010))



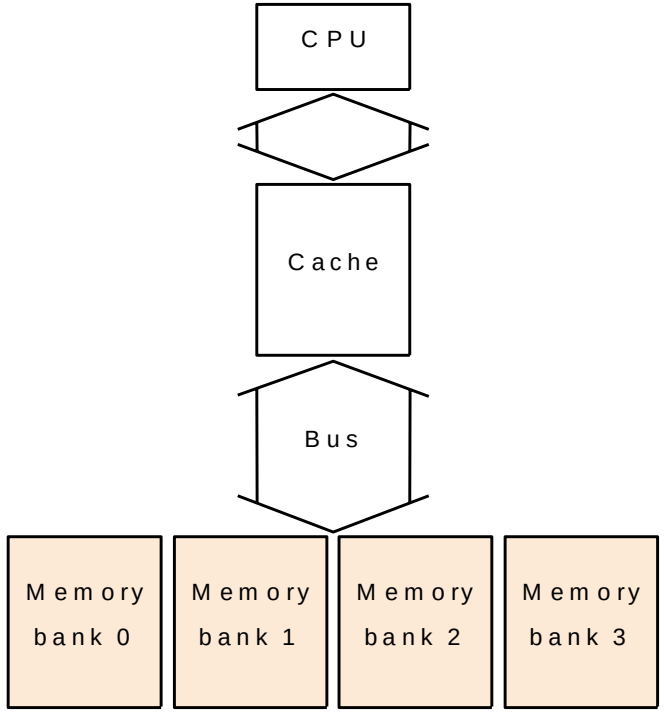
Memory Organization – 3 Options



a. One-word-wide memory organization

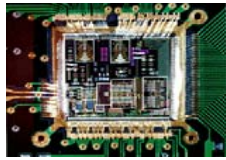


b. Wide memory organization

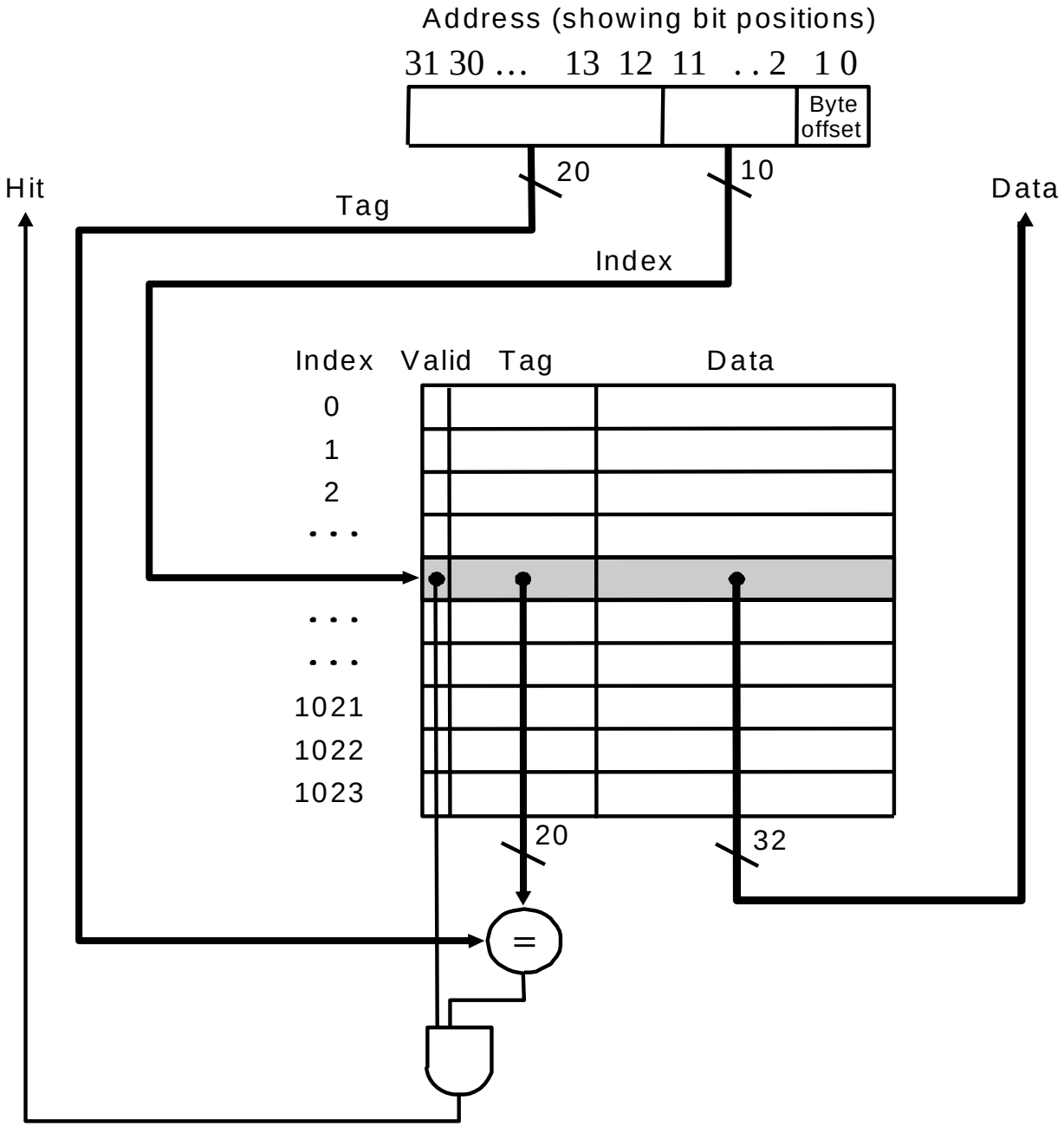


c. Interleaved memory organization

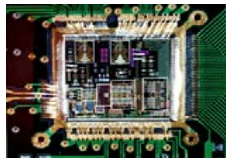
- The primary way of achieving higher memory bandwidth is to increase the physical or logical width of the memory system.
  - Wider memory, bus, and cache
  - Interleaved memory



Direct Mapped Cache



- Tag of the cache is compared with the upper portion of the address to determine whether the entry in the cache corresponds to the requested address.
- Cache has 2^{10} i.e. 1024 words and a block size of 1 word, 10-bits are used to index the cache, and 32-10-2=20bits are compared against tag.
- Cache Hit: If the tag and upper 20bits of the address are equal and the valid bit is ON and the word is supplied to the processor.
- Cache Miss: Otherwise.



An Example: Page-479 Interface Book

How many total bits are required for a direct-mapped cache with 16KB of data and 4-word blocks, assuming a 32-bit address?

Solution:

Word address size = 32 bits

We have 16KB = 4K words = 2^{12} words

As block size = 4 (2^2) words, therefore, 2^{10} blocks i.e. cache address size = 10 bits

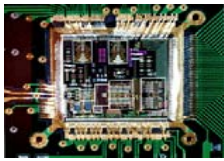
Tag width = (word address size) – (cache address size) – (block address size) – (byte offset)

Each block has 32 bits of data plus a tag, which is 32 –10–2-2 bits, plus a valid bit. Thus the cache size is:

2^{10} x (128 + (32 – 10-2 – 2) + 1) = 2^{10} x 147 = 147 Kbits.

or 18.4KB for a 16 KB cache.

For this cache, the total number of bits in the cache is over 1.15 times as many as needed just for the storage of the data.



Multi-word Cache Block

Consider a cache with 64 blocks and a block size of 16 bytes. What block number does byte address 1200 map to?

Solution:

The block is given by (Block address) modulo (Number of cache blocks)

where the address of the block is

(Byte address) / (Bytes per block)

This block address is the block containing all addresses

between

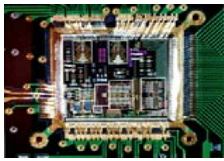
floor (Byte address/ Bytes per block) x (Bytes per block)

and

floor (Byte address/ Bytes per block) x (Bytes per block) + (Bytes per block – 1)

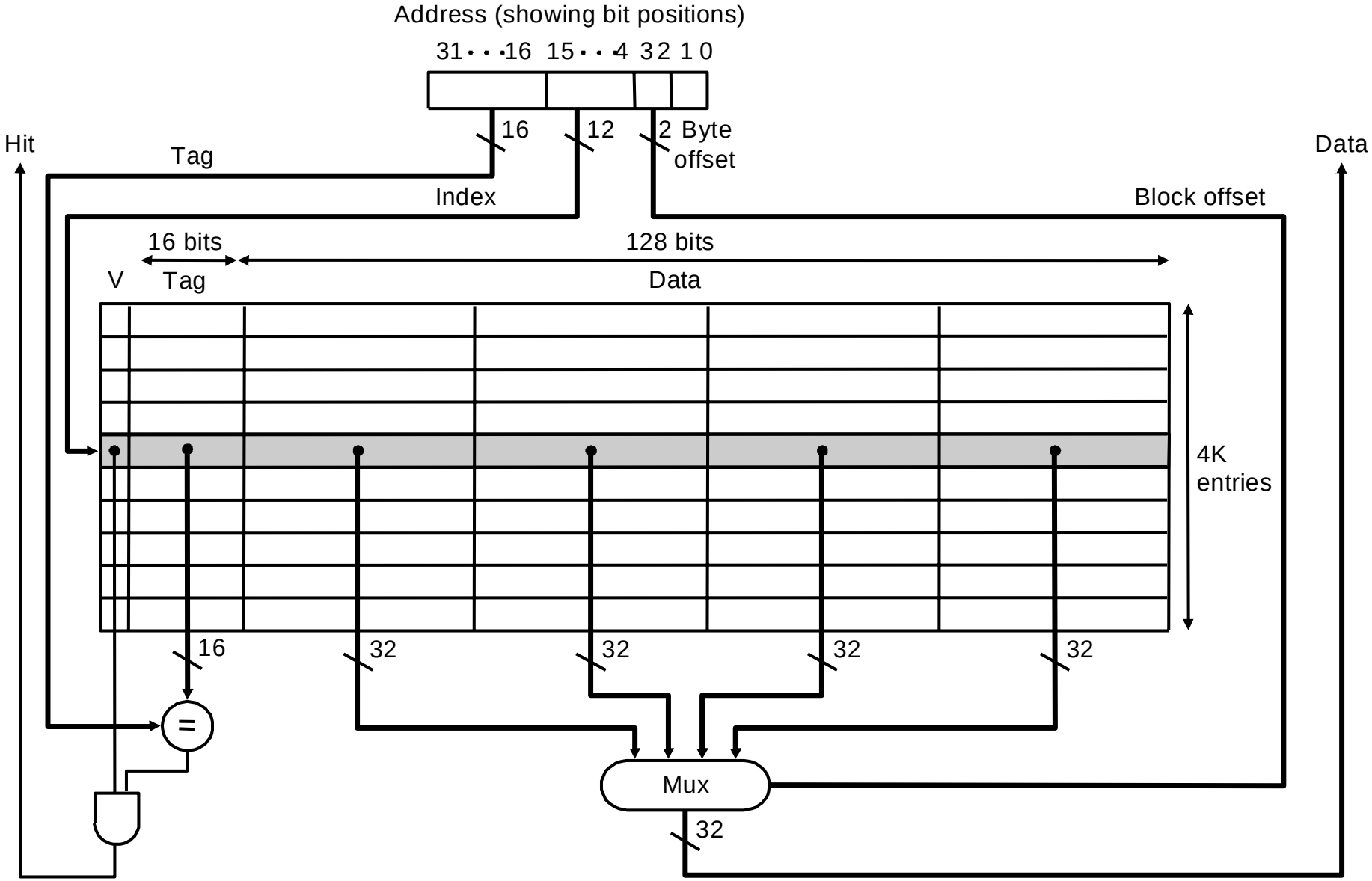
Thus, with 16 bytes per block, byte address 1200 is block address floor(1200/16) = 75

which maps to cache block number (75 module 64) = 11.



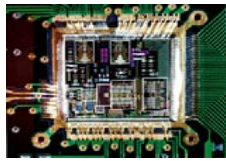
Direct Mapped Cache : 4 Words per Block

- Taking advantage of spatial locality:



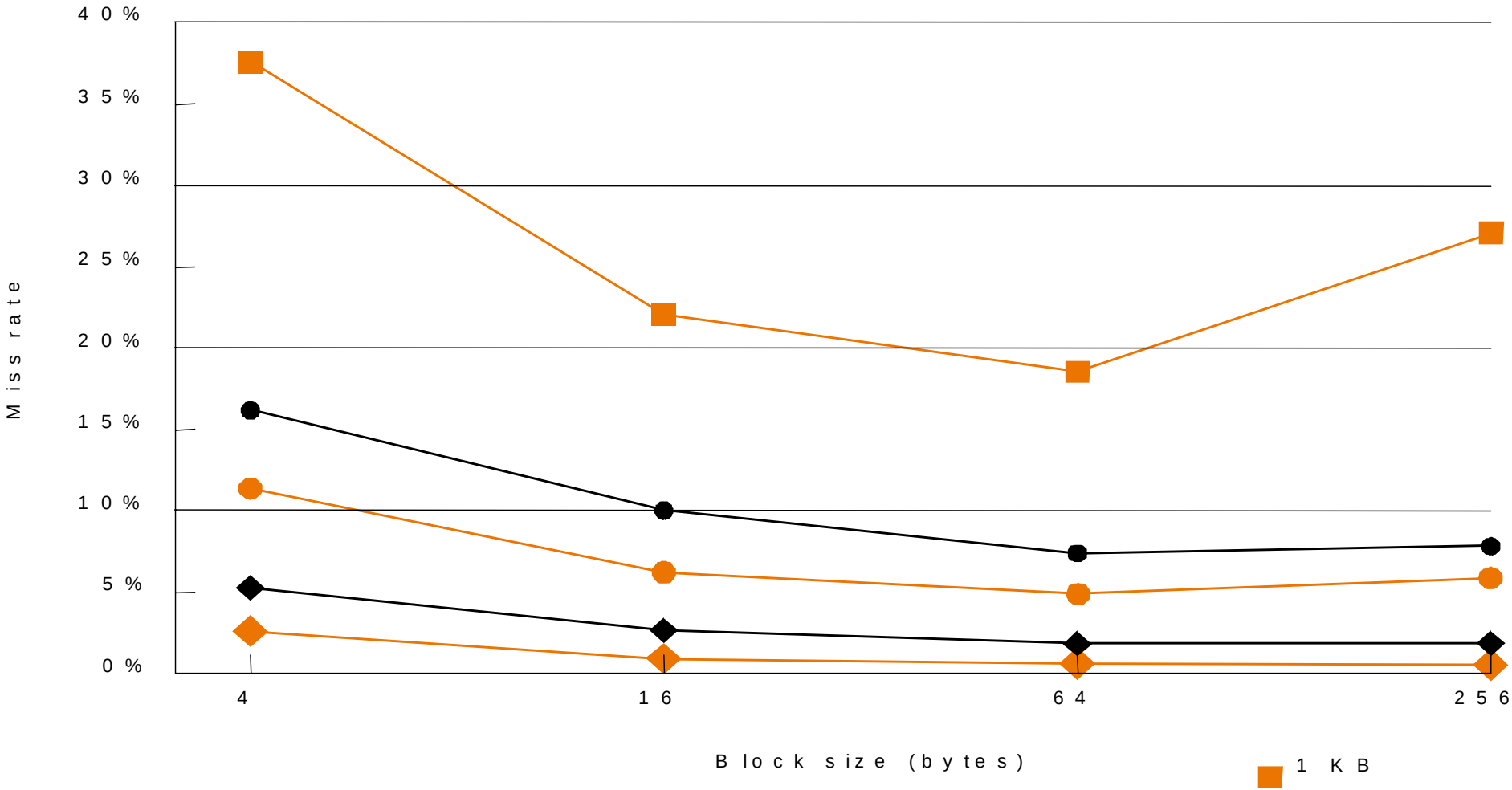
Hits vs. Misses

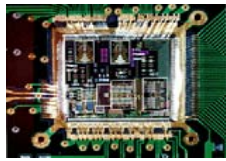
- Read hits
  - this is what we want!
- Read misses
  - stall the CPU, fetch block from memory, deliver to cache, restart
- Write hits:
  - can replace data in cache and memory (write-through)
  - write the data only into the cache (write-back the cache later)
- Write misses:
  - read the entire block into the cache, then write the word



Performance

- Increasing the block size tends to decrease miss rate:





Performance: Measurement

- CPU execution time

  = (CPU clock cycles + Memory stall cycles) × Clock cycle time
- Memory stall cycles

  = Number of misses x Miss Penalty

  = # of instructions × Miss ratio × Miss penalty
- # of instructions

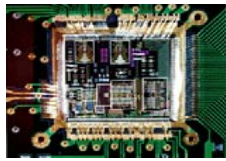
  = Instruction Count (IC)
- Miss Ratio

  = Misses / Instruction = (Memory Accesses / Instruction) x Miss rate
- Memory stall cycles

  = IC x Reads per instruction x Read miss rate x Read miss penalty

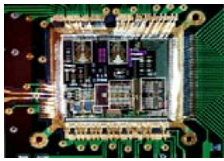
  = IC x Write per instruction x Write miss rate x Write miss penalty

NOTE: Refer page C-4 Quantitative Approach for more details.



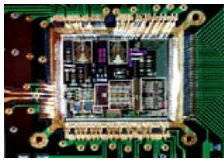
Performance: Measurement

- Example: Page-C-5 Quantitative book
  - Misses per memory reference
- Example: Page-C-6 Quantitative book
  - Misses per instruction
- Example: page-493 Interface book
  - Cache performance with separate instruction and data cache miss
- Example: page-495 Interface book
  - Cache performance with increased clock rate



Performance: Improvement

- Two ways of improving performance:
  - Decreasing the miss ratio: Improve chances of having data at higher level.
  - Decreasing the miss penalty: Improve time of transferring data from lower to higher level.



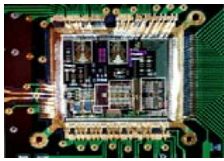
Decreasing Miss Ratio with Associativity

- In a direct mapped cache a memory block maps to exactly one cache block.

- At the other extreme, could allow a memory block to be mapped to any cache block – fully associative cache.

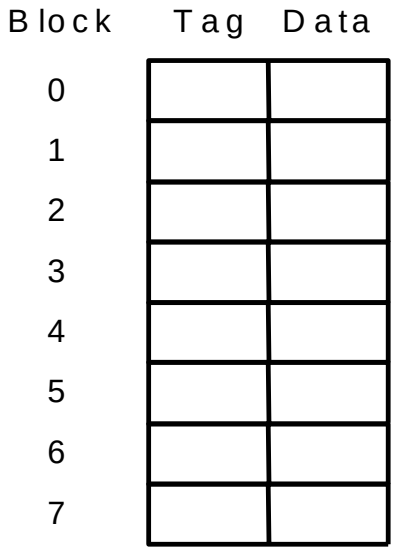
- A compromise is to divide the cache into sets each of which consists of n “ways” (n-way set associative). A memory block maps to a unique set (specified by the index field) and can be placed in any way of that set (so there are n choices)

  (block address) modulo (# sets in the cache)

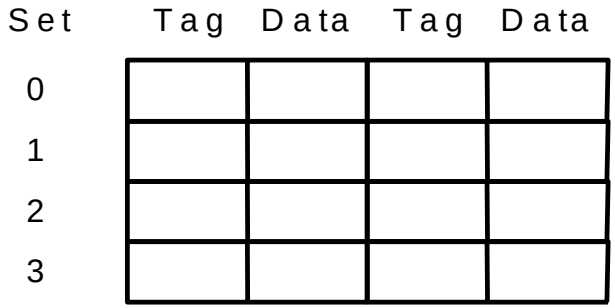


Different Configuration of a 8-Block Cache

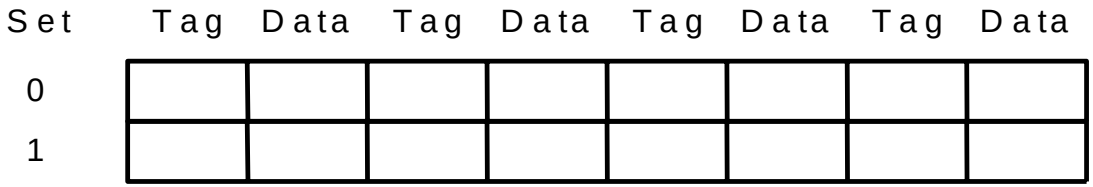
One-way set associative
(direct mapped)



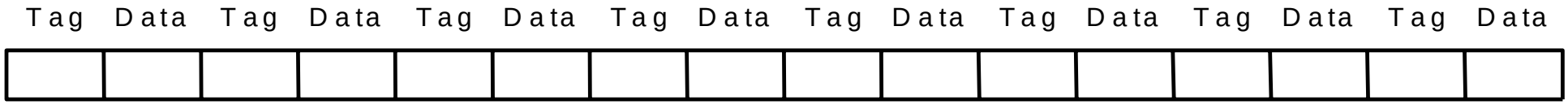
Two-way set associative



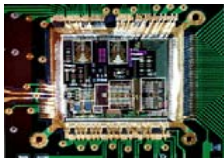
Four-way set associative



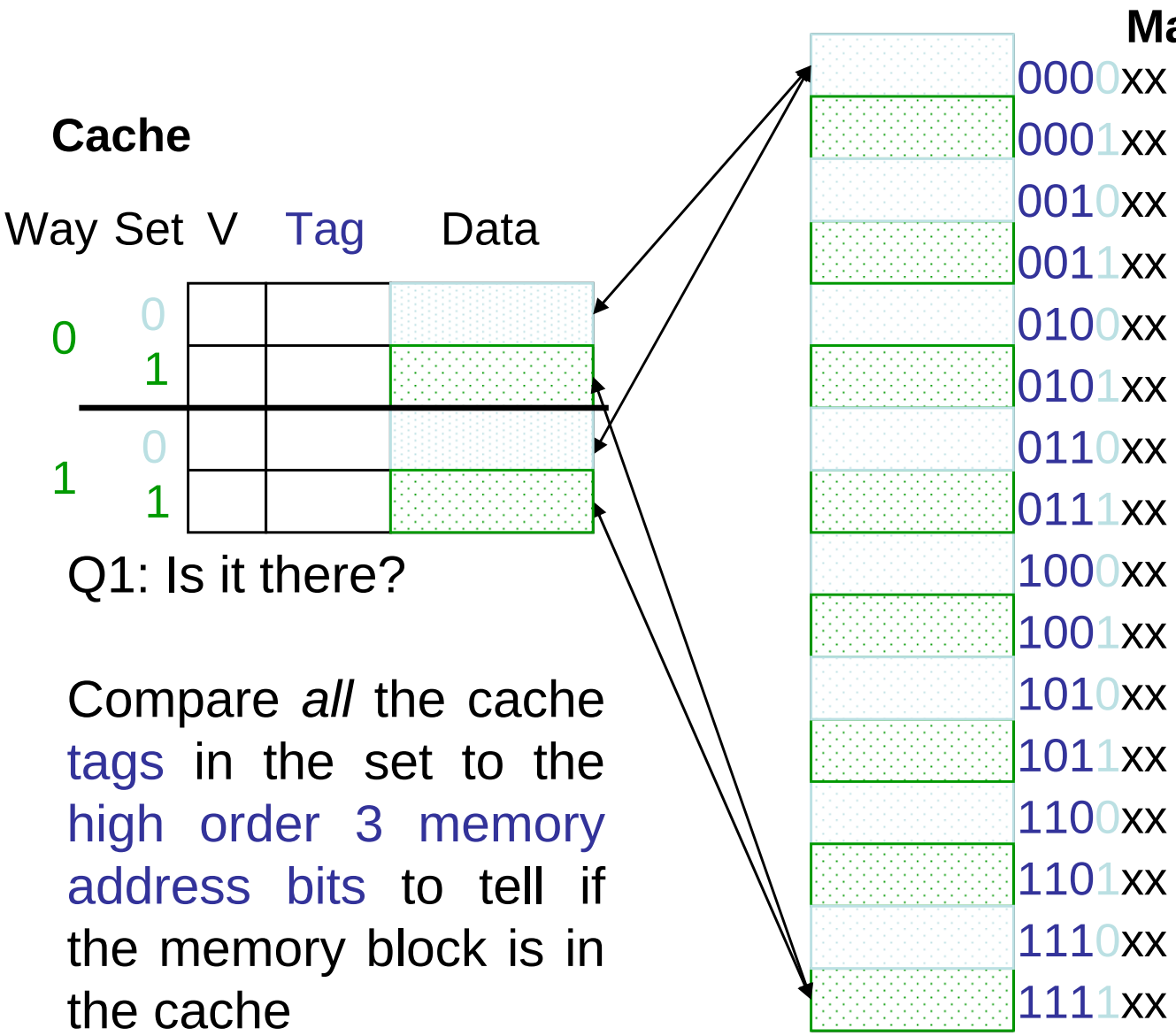
Eight-way set associative (fully associative)



Solve example page-499 of Interface book.



Set Associative Cache Example



Two low order bits define the byte in the word (32-b words)
One word blocks

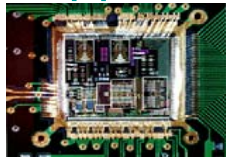
Q1: Is it there?

Compare *all* the cache tags in the set to the high order 3 memory address bits to tell if the memory block is in the cache

Q2: How do we find it?

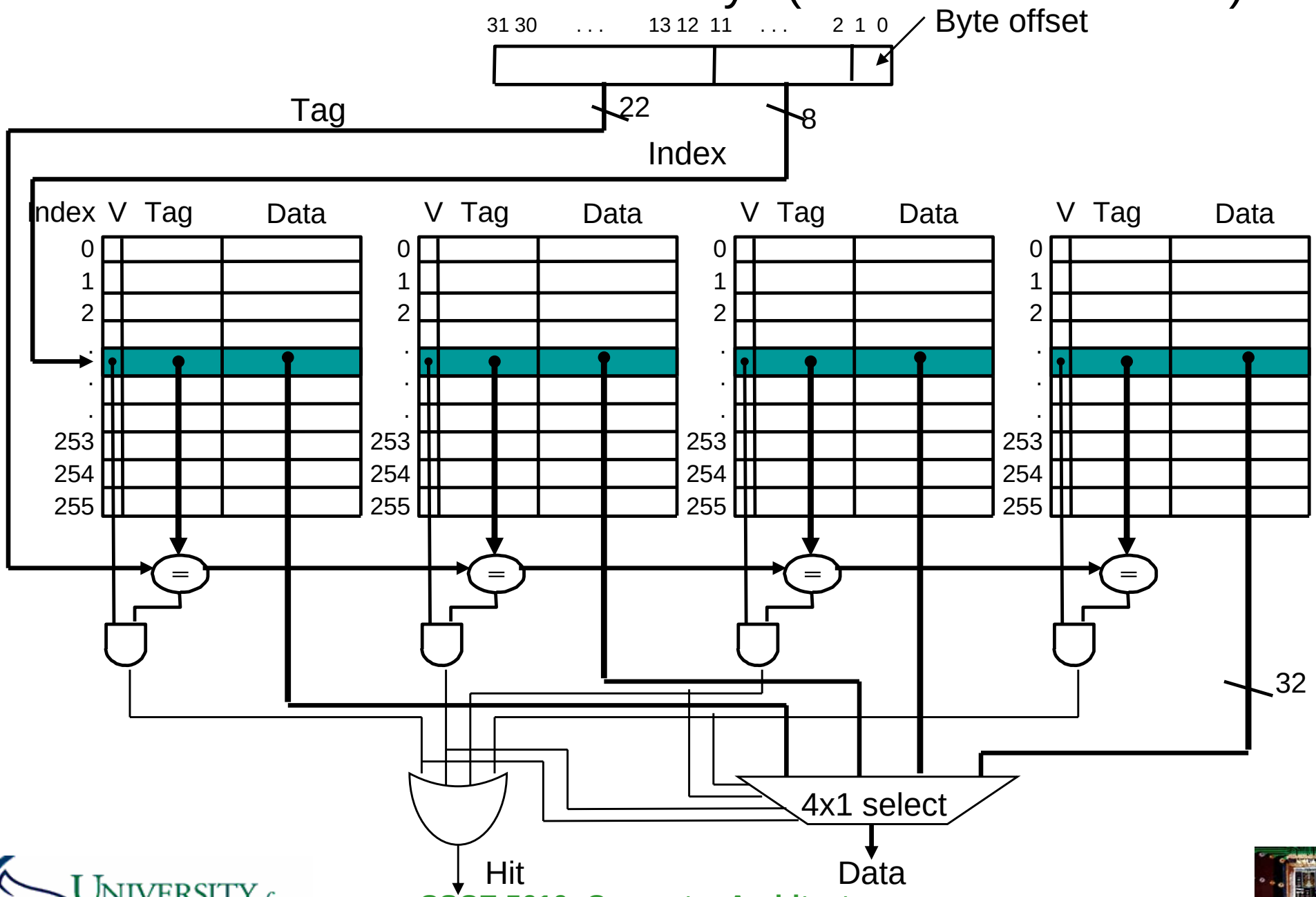
Use next 1 low order memory address bit to determine which cache set (i.e., modulo the number of sets in the cache)

NOTE: http://mdlwiki.cse.psu.edu/twiki/pub/MDL/MJI431/cse431-2021cacheperf.ppt



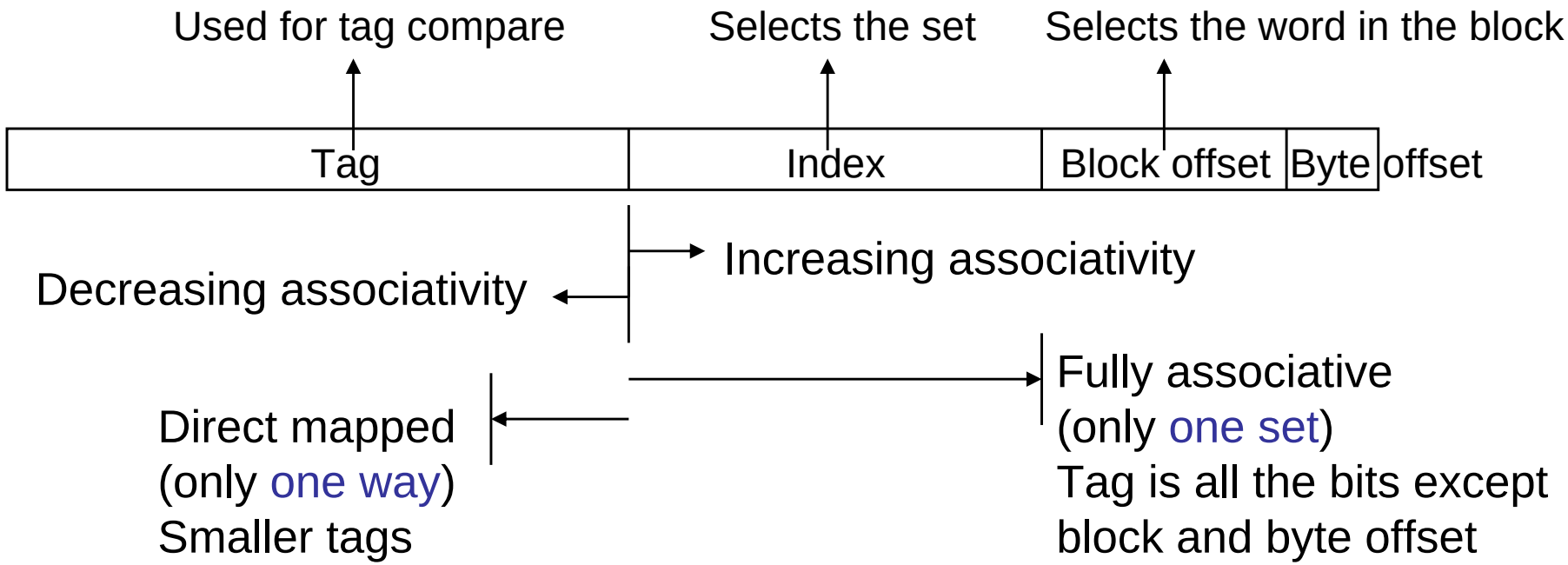
Four-Way Set Associative Cache

- 2^8 = 256 sets each with four ways (each with one block)



Range of Set Associative Caches

- For a fixed size cache, each increase by a factor of two in associativity doubles the number of blocks per set (i.e., the number or ways) and halves the number of sets – decreases the size of the index by 1 bit and increases the size of the tag by 1 bit.

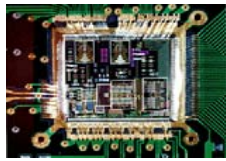


Solve example page-504 of Interface book.

NOTE: http://mdlwiki.cse.psu.edu/twiki/pub/MDL/MJI431/cse431-2021cacheperf.ppt

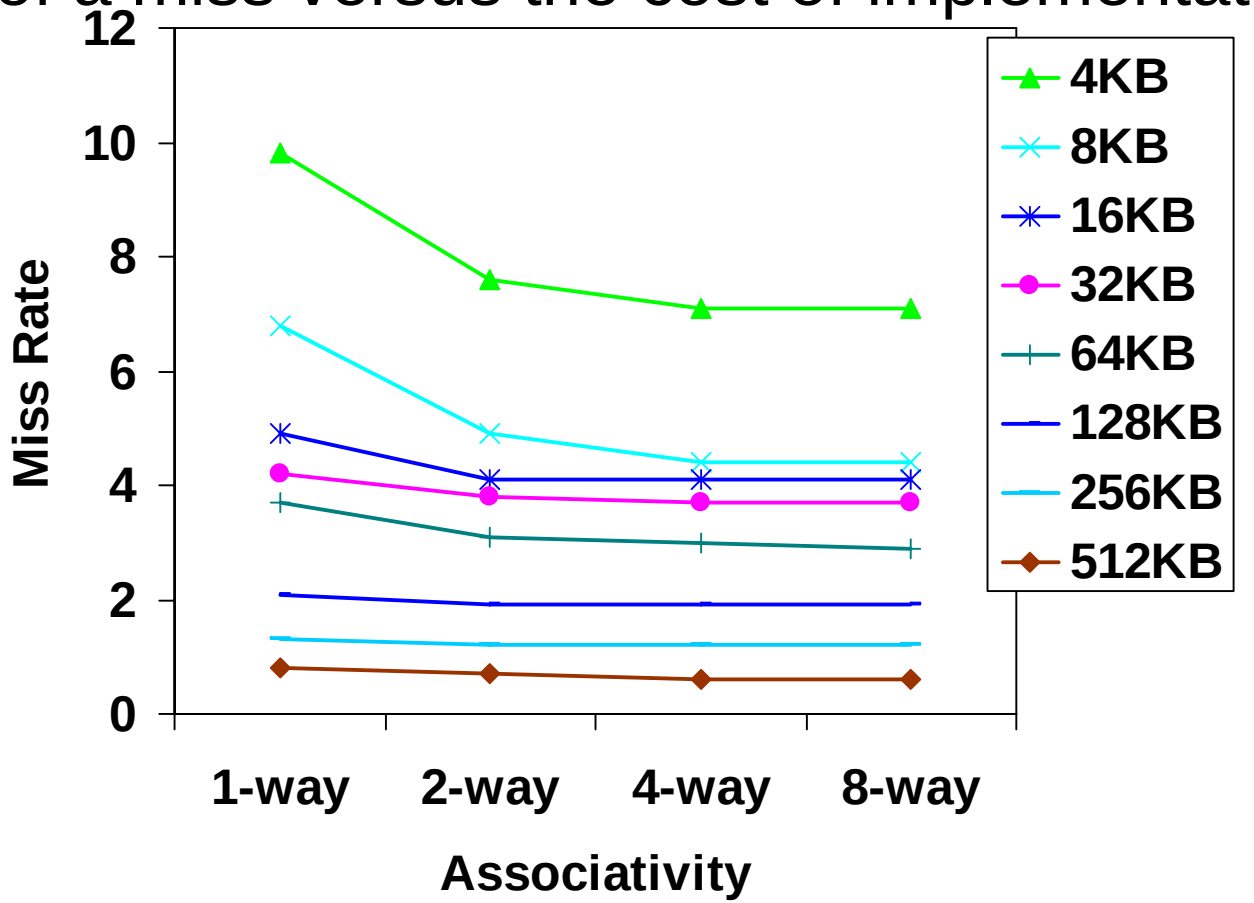
Costs of Set Associative Caches

- When a miss occurs, which way's block do we pick for replacement?
  - Least Recently Used (LRU): the block replaced is the one that has been unused for the longest time.
    - Must have hardware to keep track of when each way's block was used relative to the other blocks in the set.
    - For 2-way set associative, takes one bit per set → set the bit when a block is referenced (and reset the other way's bit).
- N-way set associative cache costs
  - N comparators (delay and area)
  - MUX delay (set selection) before data is available
  - Data available after set selection (and Hit/Miss decision). In a direct mapped cache, the cache block is available before the Hit/Miss decision.
    - So its not possible to just assume a hit and continue and recover later if it was a miss.



Benefits of Set Associative Caches

- The choice of direct mapped or set associative depends on the cost of a miss versus the cost of implementation.



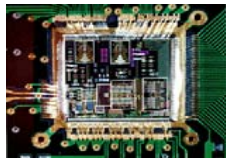
- Largest gains are in going from direct mapped to 2-way (20%+ reduction in miss rate).

NOTE: http://mdlwiki.cse.psu.edu/twiki/pub/MDL/MJI431/cse431-2021cacheperf.ppt

Decreasing Miss Penalty with Multilevel Caches

- Add a second level cache:
  - often primary cache is on the same chip as the processor.
  - use SRAMs to add another cache above primary memory (DRAM).
  - miss penalty goes down if data is in 2nd level cache.
- Example:
  - CPI of 1.0 on a 500MHz machine with a 5% miss rate, 200ns DRAM access.
  - Adding 2nd level cache with 20ns access time decreases miss rate to 2%.
- Using multilevel caches:
  - try and optimize the hit time on the 1st level cache.
  - try and optimize the miss rate on the 2nd level cache.

Solve example page-505 Interface Book.



Split Caches: Separate Instruction and Data

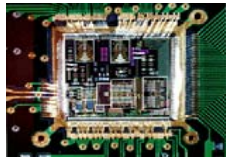
- Use split caches because there is more spatial locality in code:

Program	Block size in words	Instruction miss rate	Data miss rate	Effective combined miss rate
gcc	1	6.1%	2.1%	5.4%
	4	2.0%	1.7%	1.9%
spice	1	1.2%	1.3%	1.2%
	4	0.3%	0.6%	0.4%

- Modern processors have a split cache architecture, at least on the first cache level (L1).

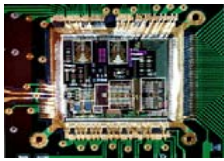
Solve example page-C-15 of Quantitative book.

NOTE: http://www.cs.colostate.edu/TechReports/Reports/1997/tr97-105.pdf



Performance: Measurement …

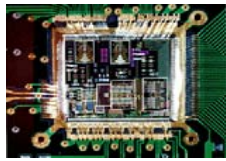
- Example: Page-C-17 Quantitative book
  - In-order execution computer: misses per instruction vs miss rate.
- Example: Page-C-18 Quantitative book
  - In-order execution computer: Direct mapped vs 2-way set associative.



Memory Hierarchy: 4 Questions

- Block placement:
  - Where can a block be placed in the upper level?
- Block identification:
  - How is a block found if it is in the upper level?
- Block replacement:
  - Which block should be replaced on a miss?
- Write strategy:
  - What happens on a write?

NOTE: Page-C-6 Quantitative book.

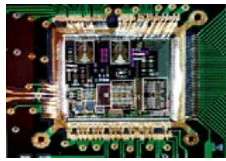


Q1&Q2: Where can a block be placed?

	# of sets	Blocks per set
Direct mapped	# of blocks in cache	1
Set associative	(# of blocks in cache)/ associativity	Associativity (typically 2 to 16)
Fully associative	1	# of blocks in cache

	Location method	# of comparisons
Direct mapped	Index	1
Set associative	Index the set; compare set's tags	Degree of associativity
Fully associative	Compare all blocks tags	# of blocks

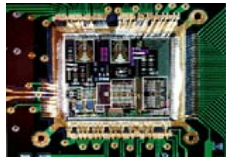
NOTE: http://mdlwiki.cse.psu.edu/twiki/pub/MDL/MJI431/cse431-2021cacheperf.ppt



Q3: Which block should be replaced on a miss?

- Easy for direct mapped – only one choice.
- Set associative or fully associative:
  - Random
  - LRU (Least Recently Used)
- For a 2-way set associative cache, random replacement has a miss rate about 1.1 times higher than LRU.
- LRU is too costly to implement for high levels of associativity (> 4-way) since tracking the usage information is costly.

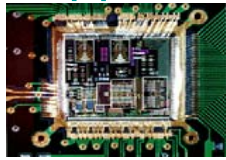
NOTE: http://mdlwiki.cse.psu.edu/twiki/pub/MDL/MJI431/cse431-2021cacheperf.ppt



Q4: What happens on a write?

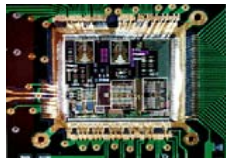
- Write-through – The information is written to both the block in the cache and to the block in the next lower level of the memory hierarchy.
  - Write-through is always combined with a write buffer so write waits to lower level memory can be eliminated (as long as the write buffer doesn't fill).
- Write-back – The information is written only to the block in the cache. The modified cache block is written to main memory only when it is replaced.
  - Need a dirty bit to keep track of whether the block is clean or dirty.
- Pros and cons of each?
  - Write-through: read misses don't result in writes (so are simpler and cheaper).
  - Write-back: repeated writes require only one write to lower level.

NOTE: http://mdlwiki.cse.psu.edu/twiki/pub/MDL/MJI431/cse431-2021cacheperf.ppt



Improving Cache Performance

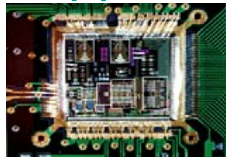
- Reduce the time to hit in the cache
  - smaller cache
  - direct mapped cache
  - smaller blocks
  - for writes
    - no write allocate – no “hit” on cache, just write to write buffer.
    - write allocate – to avoid two cycles (first check for hit, then write) pipeline writes via a delayed write buffer to cache.
- Reduce the miss rate
  - bigger cache.
  - more flexible placement (increase associativity).
  - larger blocks (16 to 64 bytes typical).
  - victim cache – small buffer holding most recently discarded blocks.



Improving Cache Performance

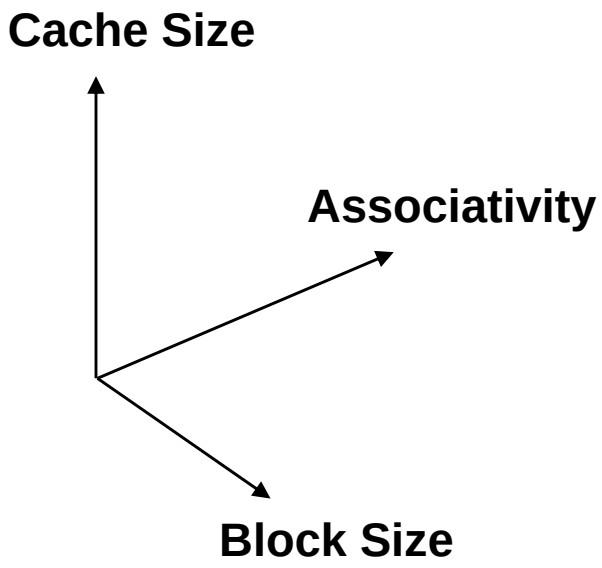
- Reduce the miss penalty
  - smaller blocks
  - use a write buffer to hold dirty blocks being replaced so don't have to wait for the write to complete before reading.
  - check write buffer (and/or victim cache) on read miss – may get lucky.
  - for large blocks fetch critical word first.
  - use multiple cache levels – L2 cache not tied to CPU clock rate.
  - faster backing store/improved memory bandwidth.
    - wider buses
    - memory interleaving, page mode DRAMs

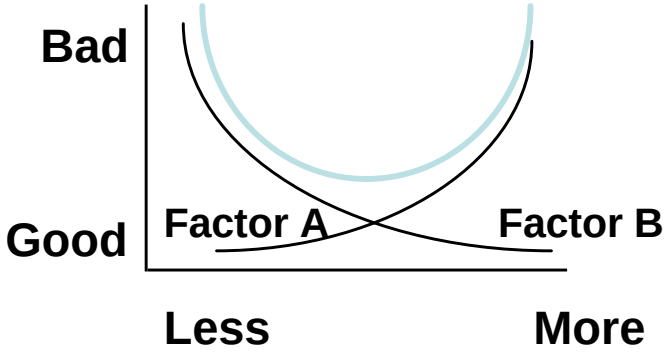
NOTE: http://mdlwiki.cse.psu.edu/twiki/pub/MDL/MJI431/cse431-2021cacheperf.ppt



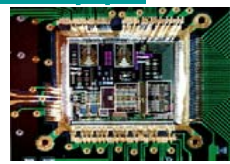
Summary: The Cache Design Space

- Several interacting dimensions
  - cache size
  - block size
  - associativity
  - replacement policy
  - write-through vs write-back
  - write allocation
- The optimal choice is a compromise
  - depends on access characteristics
    - workload
    - use (I-cache, D-cache, TLB)
  - depends on technology / cost
- Simplicity often wins



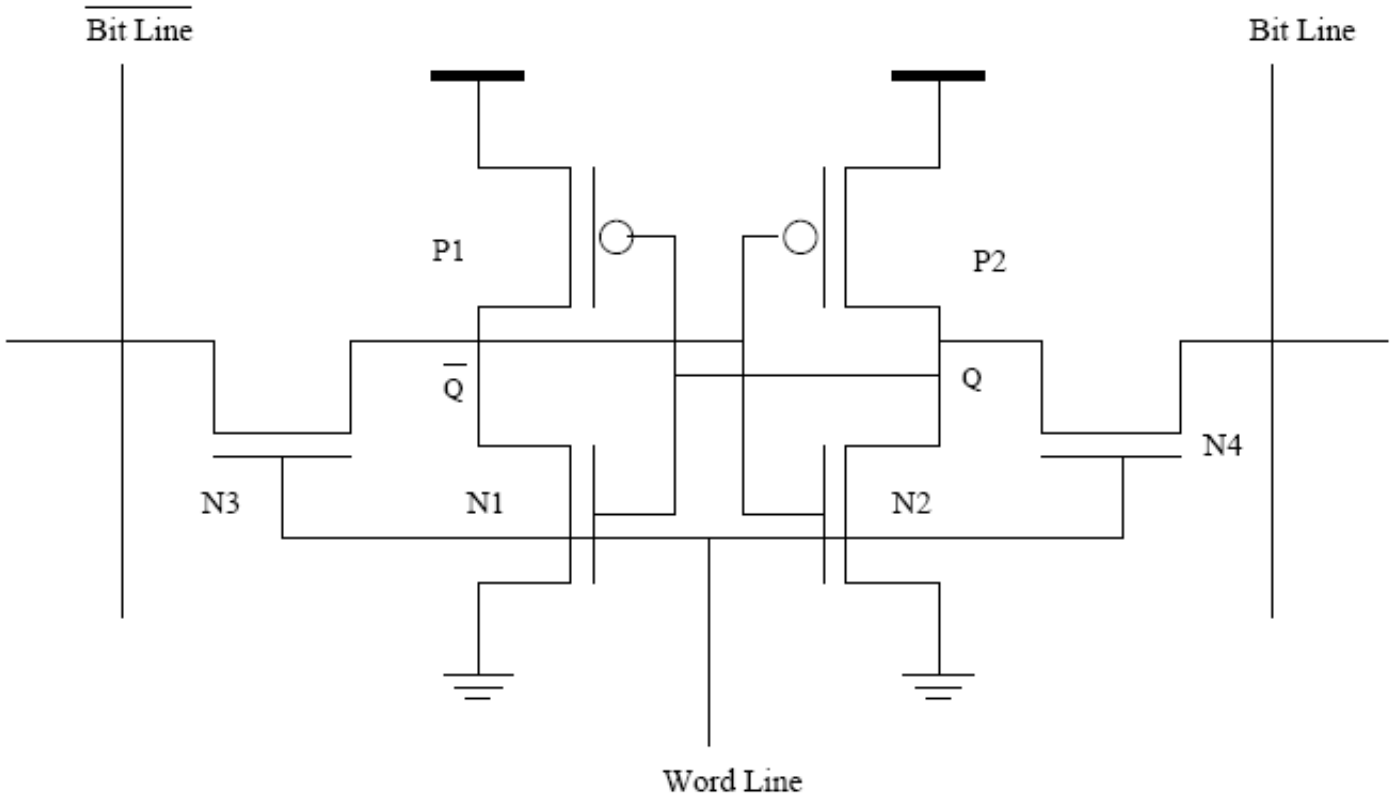


NOTE: http://mdlwiki.cse.psu.edu/twiki/pub/MDL/MJI431/cse431-2021cacheperf.ppt



Memory Technology: SRAM

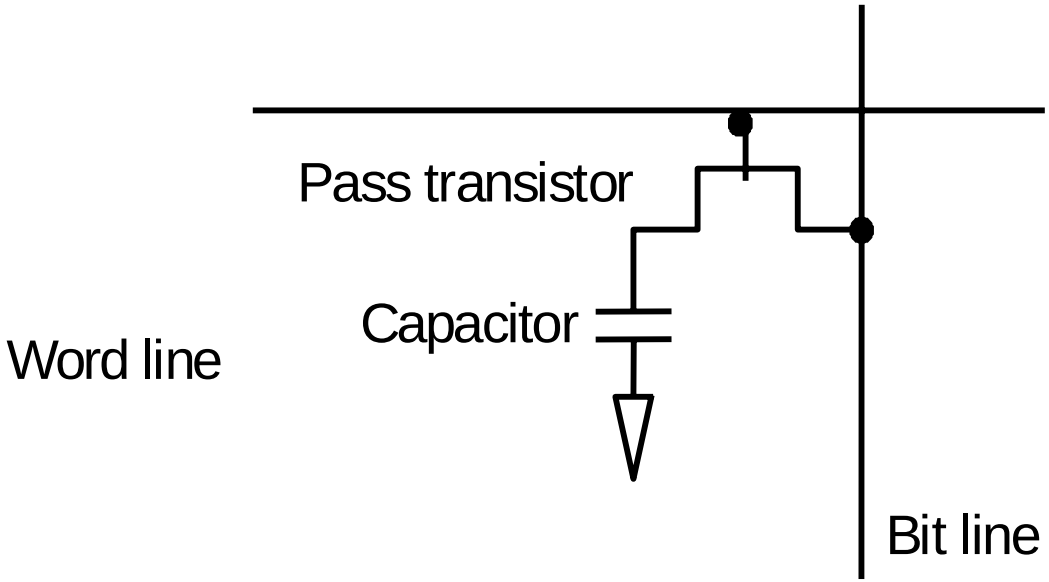
- SRAM:
  - value is stored on a pair of inverting gates
  - very fast but takes up more space than DRAM (4 to 6 transistors)



NOTE: Page-311 Quantitative book.

Memory Technology: DRAM

- DRAM:
  - value is stored as a charge on capacitor (must be refreshed)
  - very small but slower than SRAM (factor of 5 to 10)

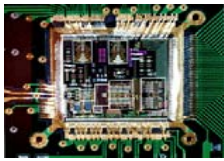


NOTE: Page-311 Quantitative book.

Virtual Memory: Motivations

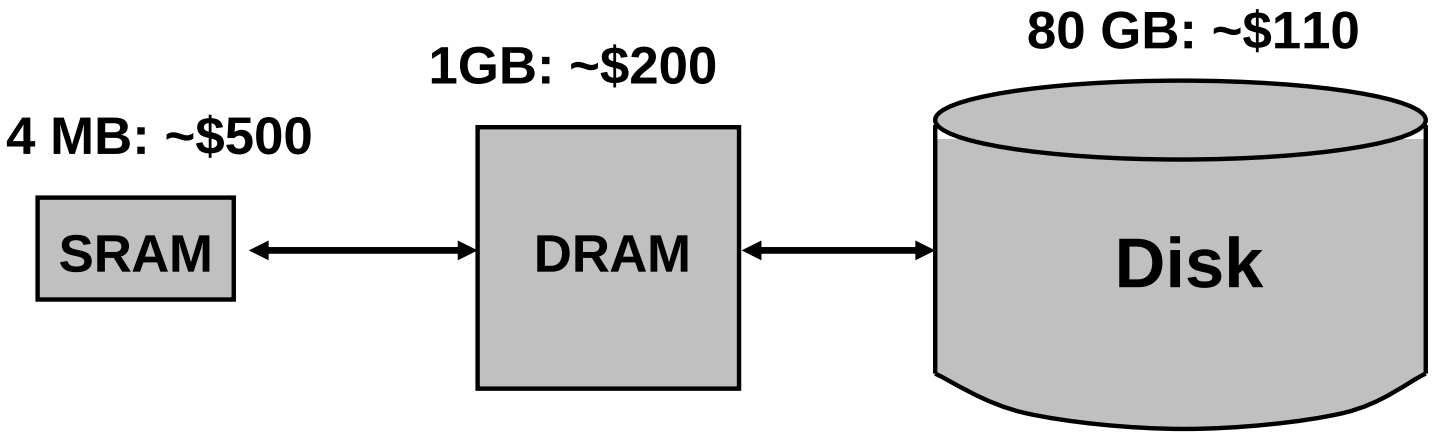
- Use Physical DRAM as a Cache for the Disk
  - Address space of a process can exceed physical memory size.
  - Sum of address spaces of multiple processes can exceed physical memory.
- Simplify Memory Management
  - Multiple processes resident in main memory.
    - Each process with its own address space.
  - Only “active” code and data is actually in memory.
    - Allocate more memory to process as needed.
- Provide Protection
  - One process can’t interfere with another.
    - because they operate in different address spaces.
  - User process cannot access privileged information
    - different sections of address spaces have different permissions.

NOTE: http://csapp.cs.cmu.edu/public/lectures/class19.ppt



Motivation #1: DRAM a "Cache" for Disk

- Full address space is quite large:
  - –32-bit addresses: ~4,000,000,000 (4 billion) bytes
  - –64-bit addresses: ~16,000,000,000,000,000,000 (16 quintillion) bytes
- Disk storage is ~300X cheaper than DRAM storage.
- To access large amounts of data in a cost-effective manner, the bulk of the data must be stored on disk.

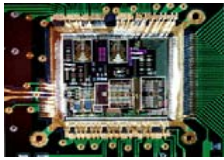


NOTE: http://csapp.cs.cmu.edu/public/lectures/class19.ppt

DRAM vs. SRAM as a "Cache"

- DRAM vs. disk is more extreme than SRAM vs. DRAM.
  - Access latencies:
    - DRAM ~10X slower than SRAM.
    - Disk ~**100,000X** slower than DRAM.
  - Importance of exploiting spatial locality:
    - First byte is ~**100,000X** slower than successive bytes on disk.
      - vs. ~4X improvement for page-mode vs. regular accesses to DRAM.
  - Bottom line:
    - Design decisions made for DRAM caches driven by enormous cost of misses.

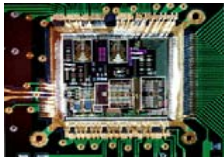
NOTE: http://csapp.cs.cmu.edu/public/lectures/class19.ppt



Impact of Properties on Design

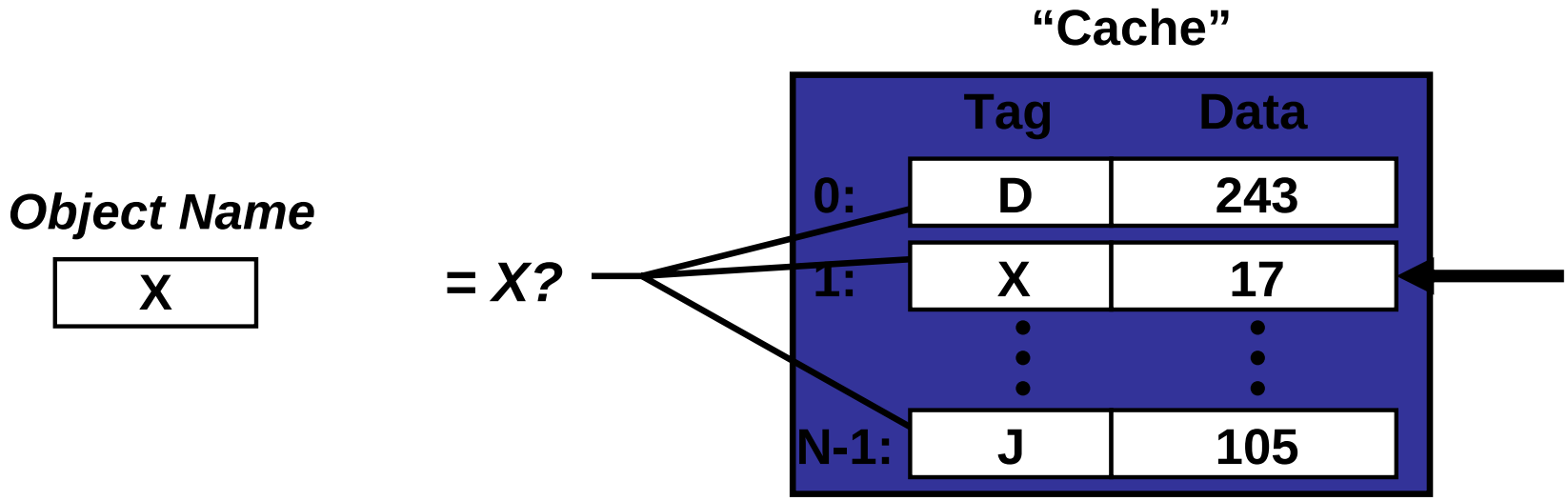
- If DRAM was to be organized similar to an SRAM cache, how would we set the following design parameters?
  - Line size: Large, since disk better at transferring large blocks
  - Associativity: High, to mimimize miss rate
  - Write through or write back?
    - Write back, since can't afford to perform small writes to disk
- What would the impact of these choices be on:
  - miss rate: Extremely low. << 1%
  - hit time: Must match cache/DRAM performance
  - miss latency: Very high. ~20ms
  - tag storage overhead: Low, relative to block size

NOTE: http://csapp.cs.cmu.edu/public/lectures/class19.ppt



Locating an Object in a "Cache"

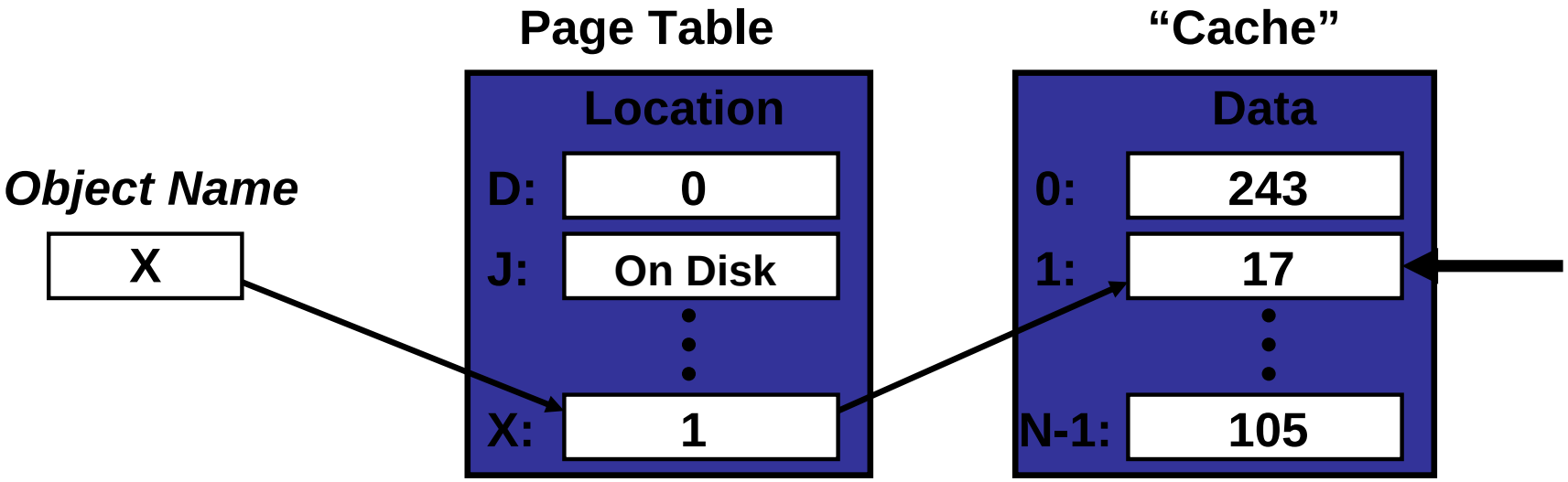
- SRAM Cache
  - Tag stored with cache line
  - Maps from cache block to memory blocks
    - From cached to uncached form
    - Save a few bits by only storing tag
  - No tag for block not in cache
  - Hardware retrieves information
    - can quickly match against multiple tags



NOTE: http://csapp.cs.cmu.edu/public/lectures/class19.ppt

Locating an Object in "Cache" (cont.)

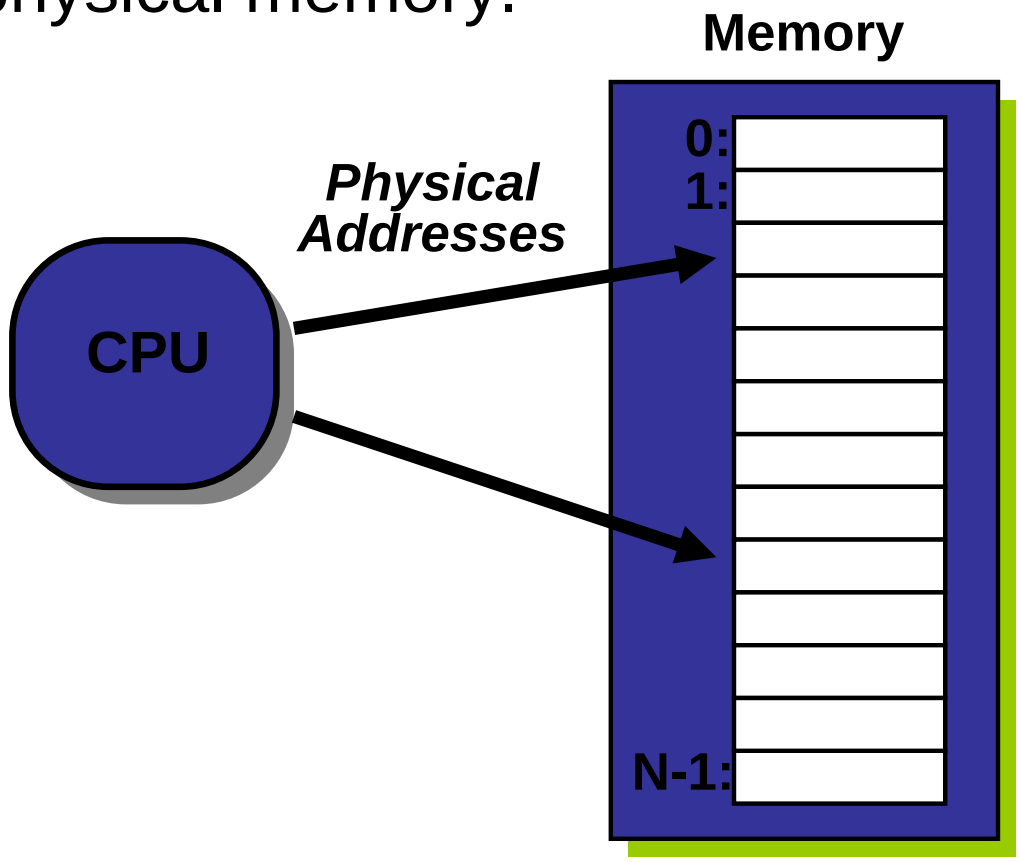
- DRAM Cache
  - Each allocated page of virtual memory has entry in *page table.*
  - Mapping from virtual pages to physical pages.
    - From uncached form to cached form.
  - Page table entry even if page not in memory.
    - Specifies disk address.
    - Only way to indicate where to find page.
  - OS retrieves information.



NOTE: http://csapp.cs.cmu.edu/public/lectures/class19.ppt

A System with Physical Memory Only

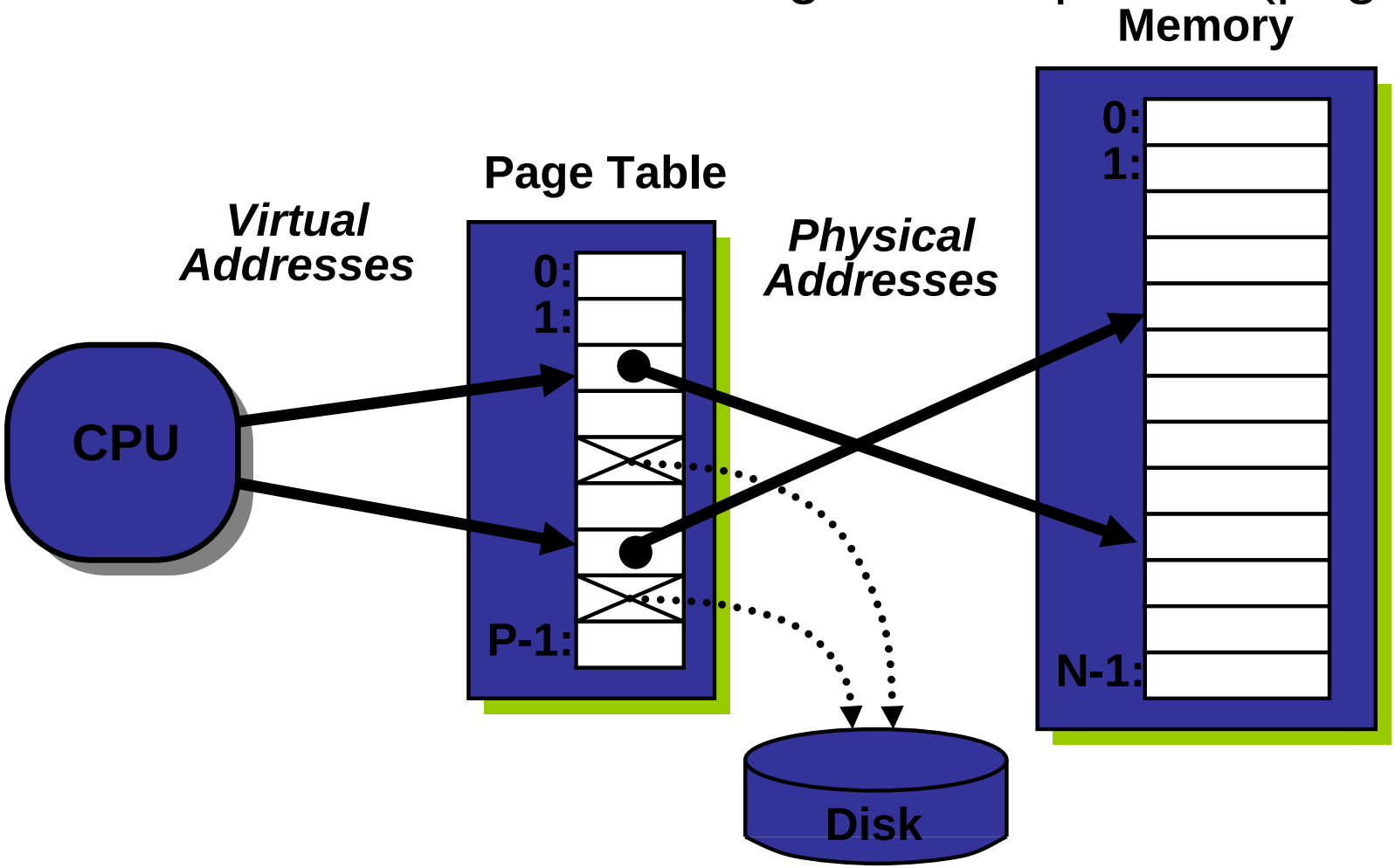
- Examples: Early PCs, nearly all embedded systems, etc.
- Addresses generated by the CPU correspond directly to bytes in physical memory.



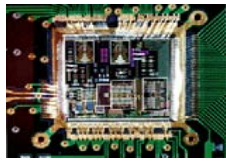
NOTE: http://csapp.cs.cmu.edu/public/lectures/class19.ppt

A System with Virtual Memory

- Examples: workstations, servers, modern PCs, etc.
- Address Translation: Hardware converts virtual addresses to physical addresses via OS-managed lookup table (page table).

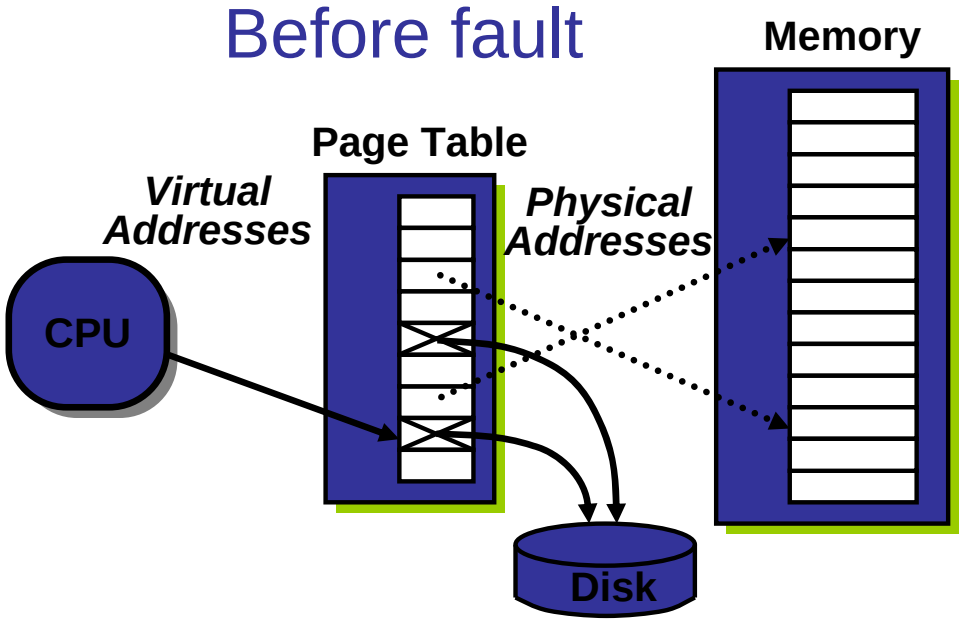


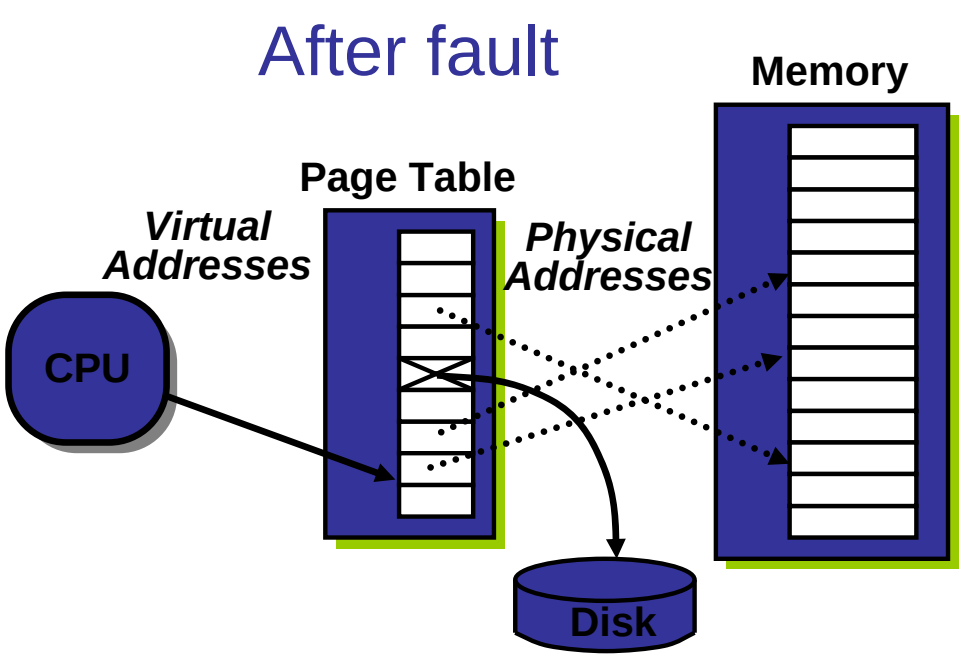
NOTE: http://csapp.cs.cmu.edu/public/lectures/class19.ppt



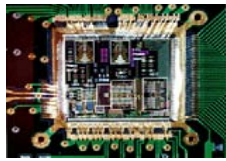
Page Faults (like "Cache Misses")

- What if an object is on disk rather than in memory?
  - Page table entry indicates virtual address not in memory.
  - OS exception handler invoked to move data from disk into memory.
    - current process suspends, others can resume
    - OS has full control over placement, etc.



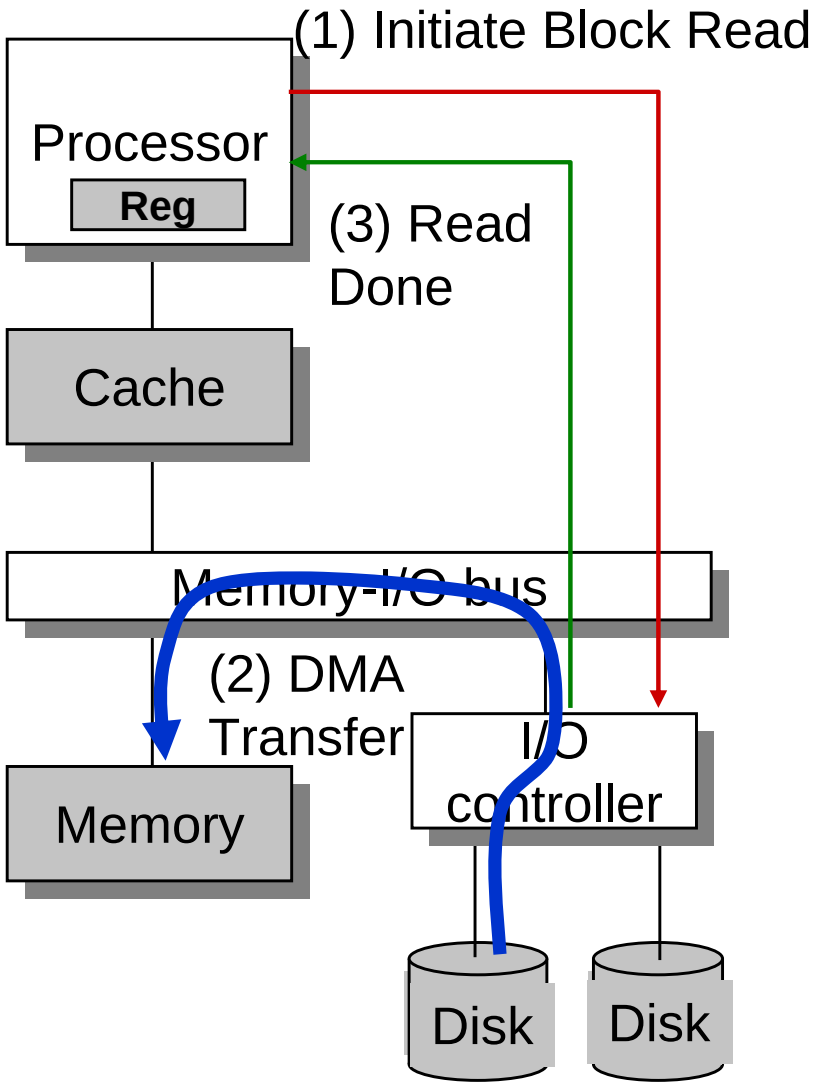


NOTE: http://csapp.cs.cmu.edu/public/lectures/class19.ppt



Servicing a Page Fault

- Processor Signals Controller
  - –Read block of length P starting at disk address X and store starting at memory address Y.
- Read Occurs
  - –Direct Memory Access (DMA)
  - –Under control of I/O controller
- I / O Controller Signals Completion
  - –Interrupt processor
  - –OS resumes suspended process

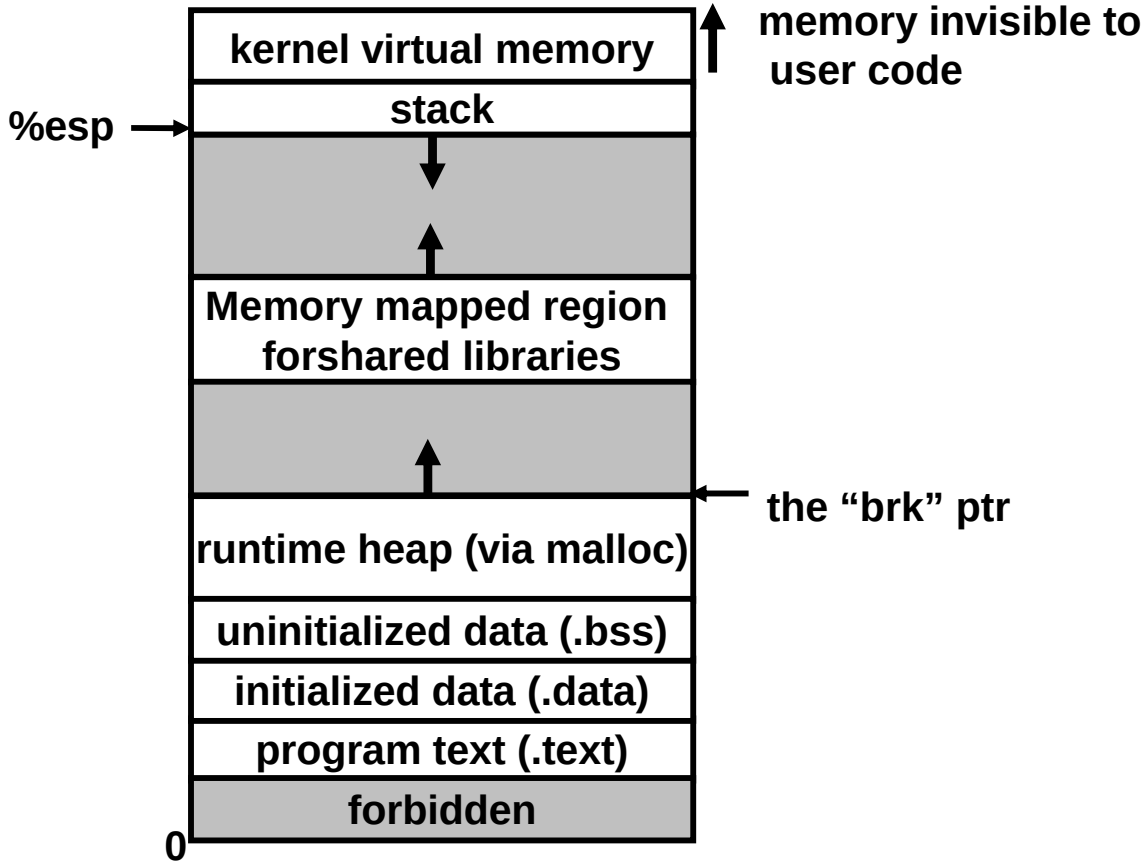


NOTE: http://csapp.cs.cmu.edu/public/lectures/class19.ppt

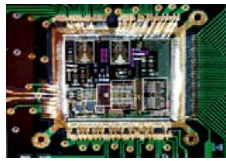
Motivation #2: Memory Management

- Multiple processes can reside in physical memory.
- How do we resolve address conflicts?
  - What if two processes access something at the same address?

Linux/x86 process memory image

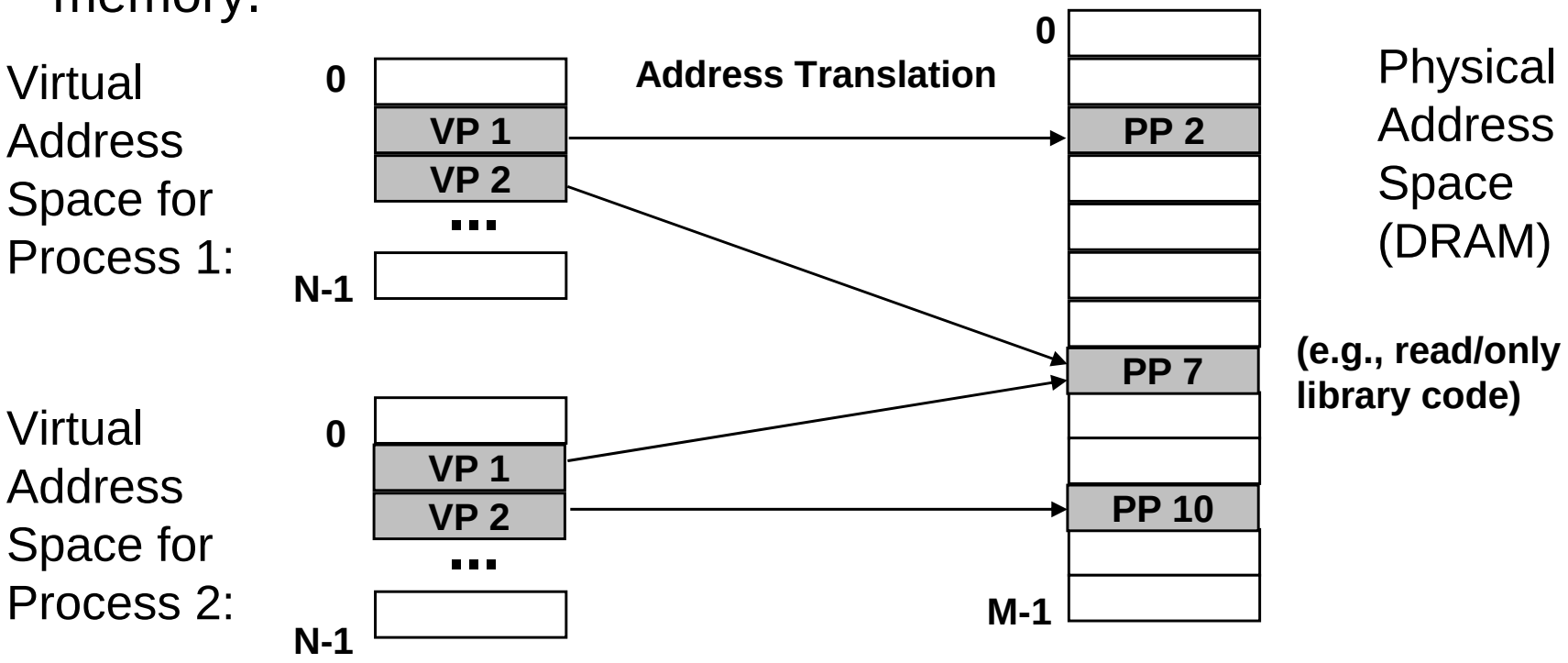


NOTE: http://csapp.cs.cmu.edu/public/lectures/class19.ppt



Solution: Separate Virtual Address Spaces

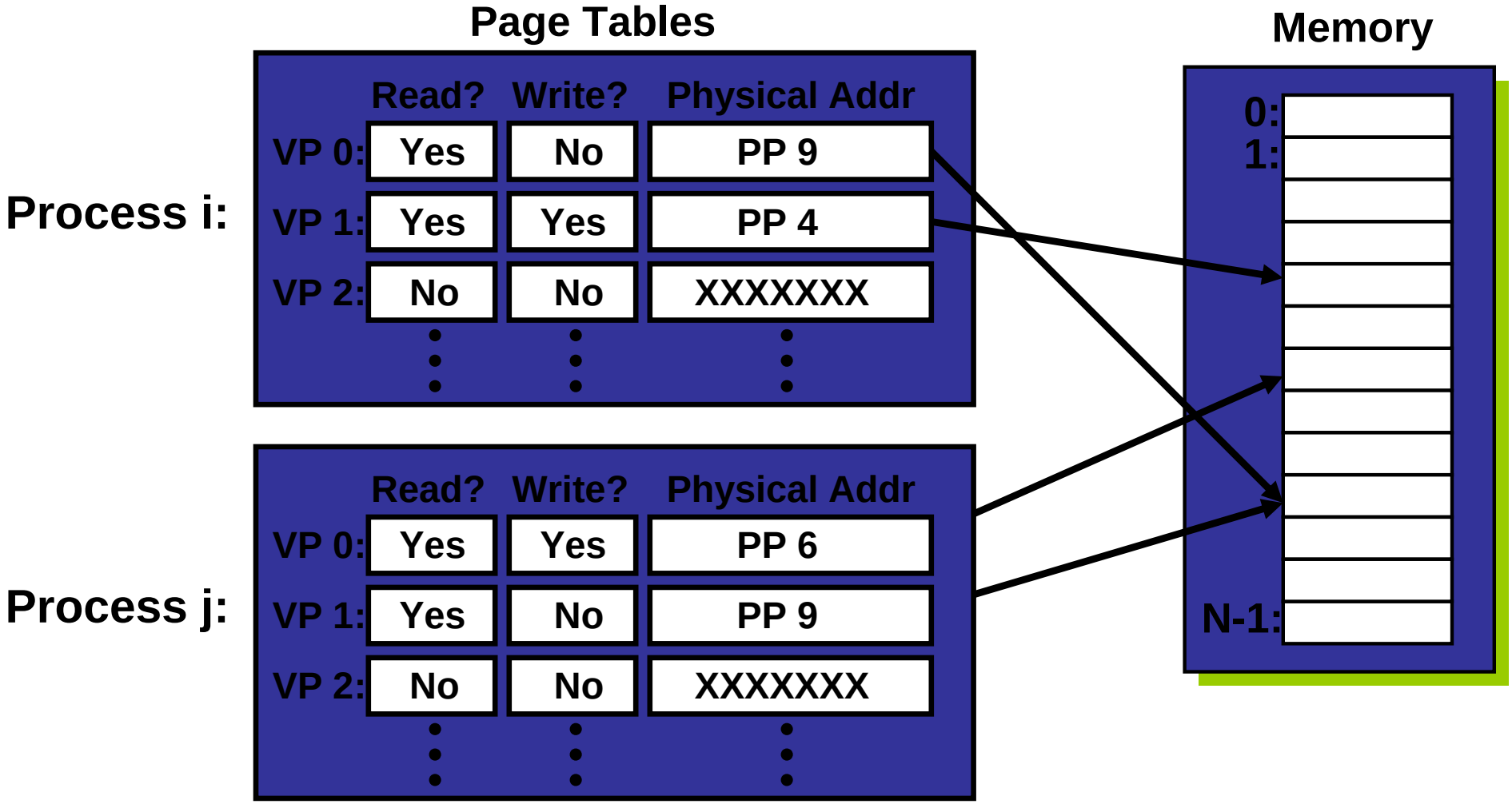
- Virtual and physical address spaces divided into equal-sized blocks.
  - blocks are called “pages” (both virtual and physical).
- Each process has its own virtual address space
  - operating system controls how virtual pages as assigned to physical memory.



NOTE: http://csapp.cs.cmu.edu/public/lectures/class19.ppt

Motivation #3: Protection

- Page table entry contains access rights information.
  - hardware enforces this protection (trap into OS if violation occurs)

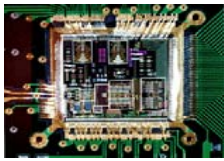


NOTE: http://csapp.cs.cmu.edu/public/lectures/class19.ppt

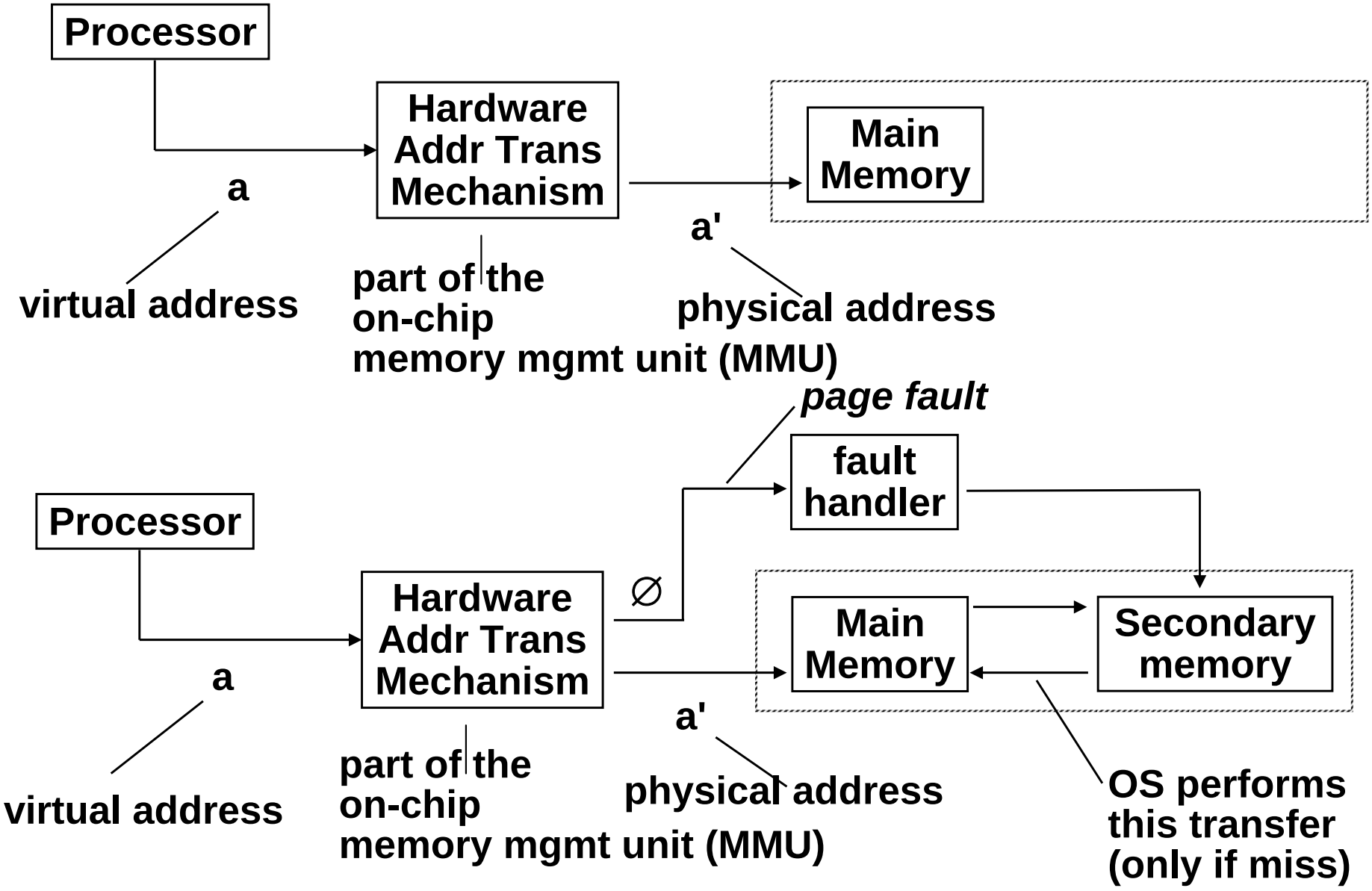
VM Address Translation

- Virtual Address Space
  - V = {0, 1, …, N–1}
- Physical Address Space
  - P = {0, 1, …, M–1}
  - M < N
- Address Translation
  - MAP: V $\rightarrow$ P U {$\varnothing$}
  - For virtual address a:
    - MAP(a) = a' if data at virtual address a at physical address a' in P
    - MAP(a) = $\varnothing$ if data at virtual address a not in physical memory
      - Either invalid or stored on disk

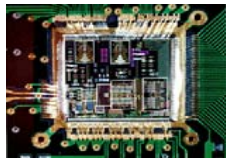
NOTE: http://csapp.cs.cmu.edu/public/lectures/class19.ppt



VM Address Translation: Hit and Miss

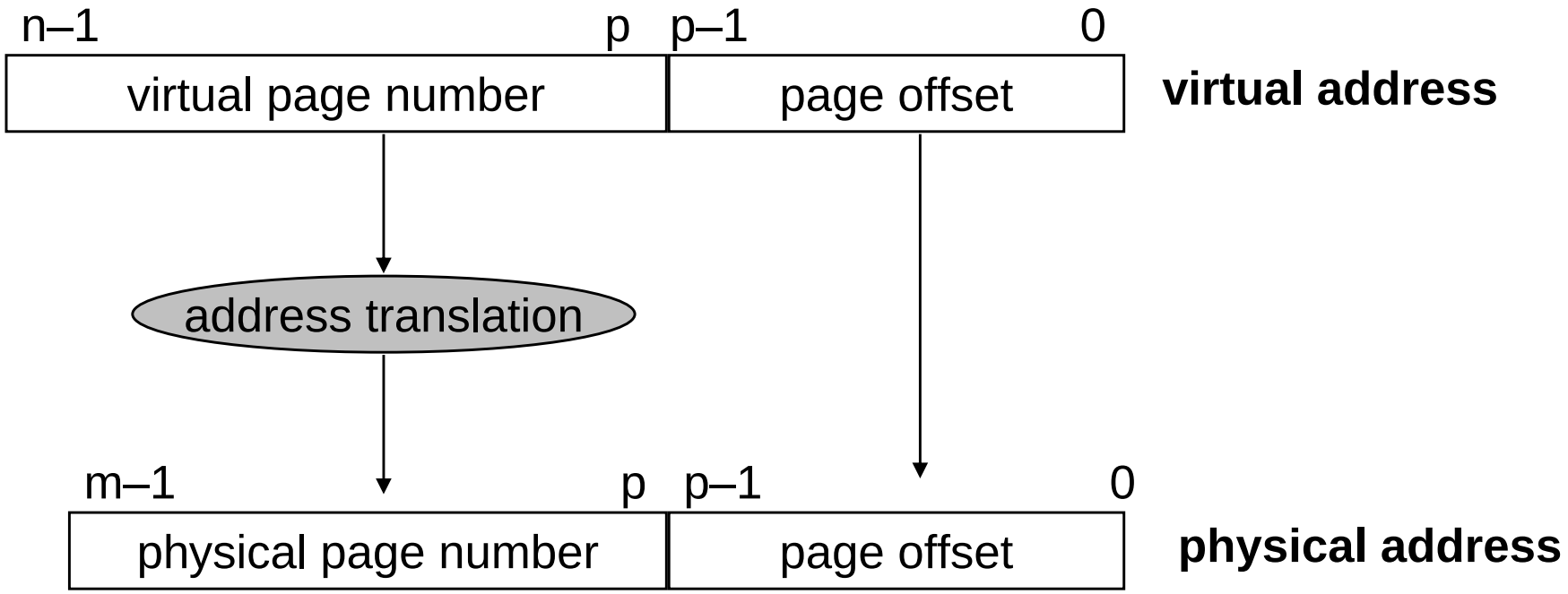


NOTE: http://csapp.cs.cmu.edu/public/lectures/class19.ppt

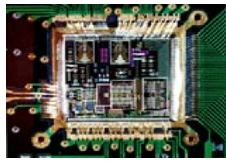


VM Address Translation

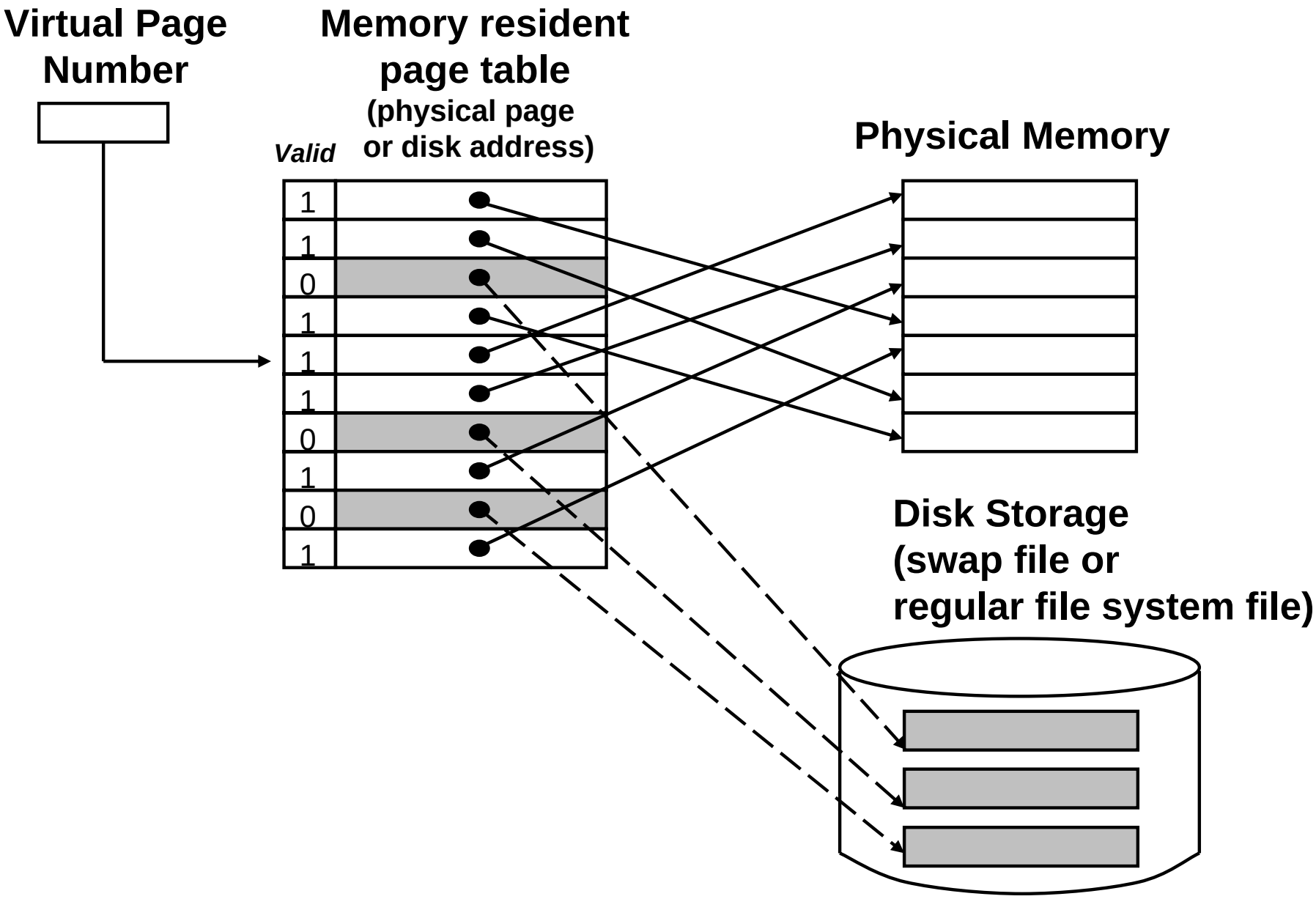
- Parameters
  - $P = 2^p$ = page size (bytes).
  - $N = 2^n$ = Virtual address limit
  - $M = 2^m$ = Physical address limit
- Page offset bits don't change as a result of translation.



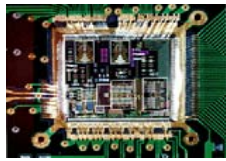
NOTE: http://csapp.cs.cmu.edu/public/lectures/class19.ppt



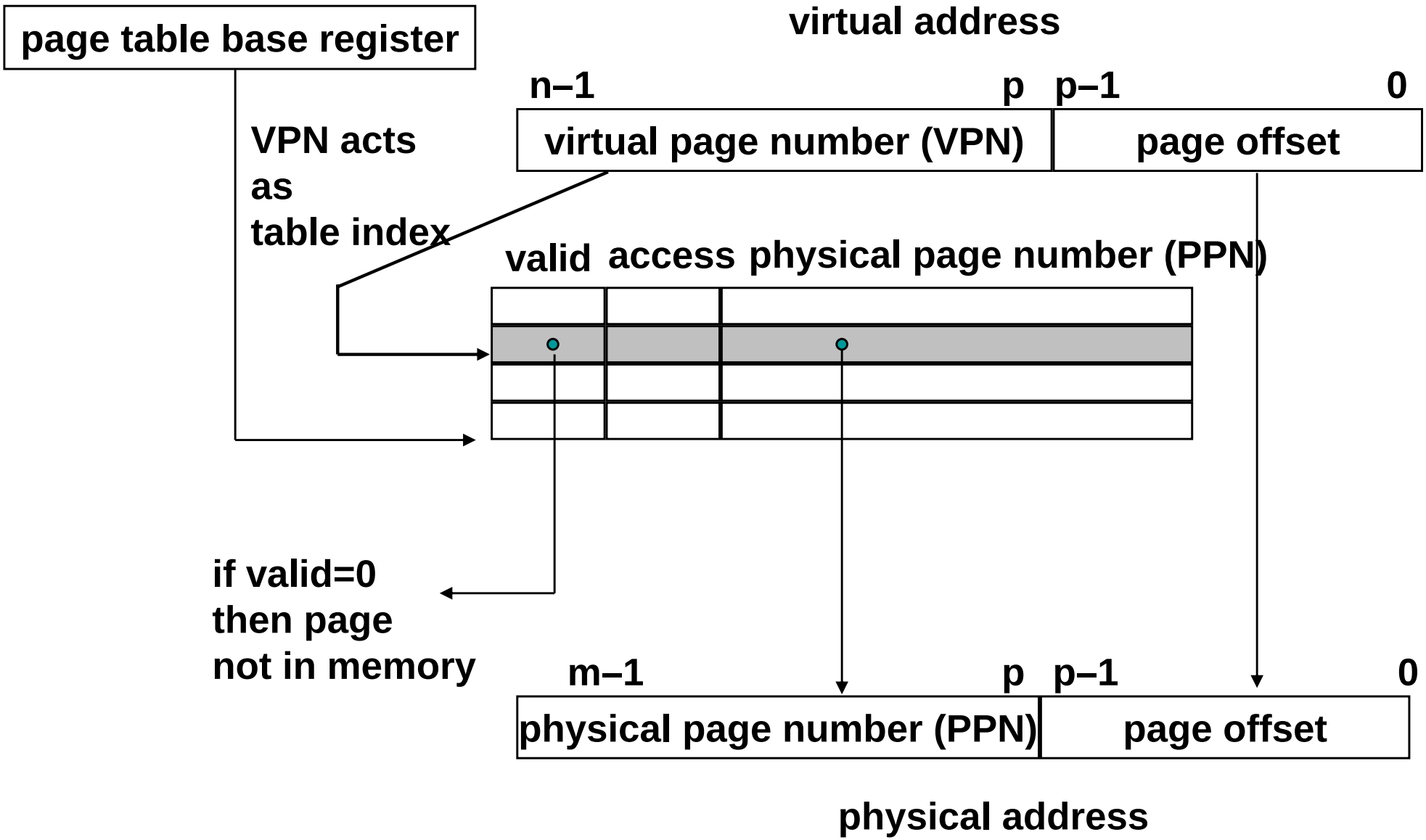
Page Table



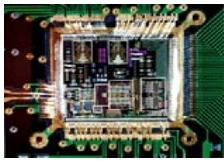
NOTE: http://csapp.cs.cmu.edu/public/lectures/class19.ppt



Address Translation via Page Table

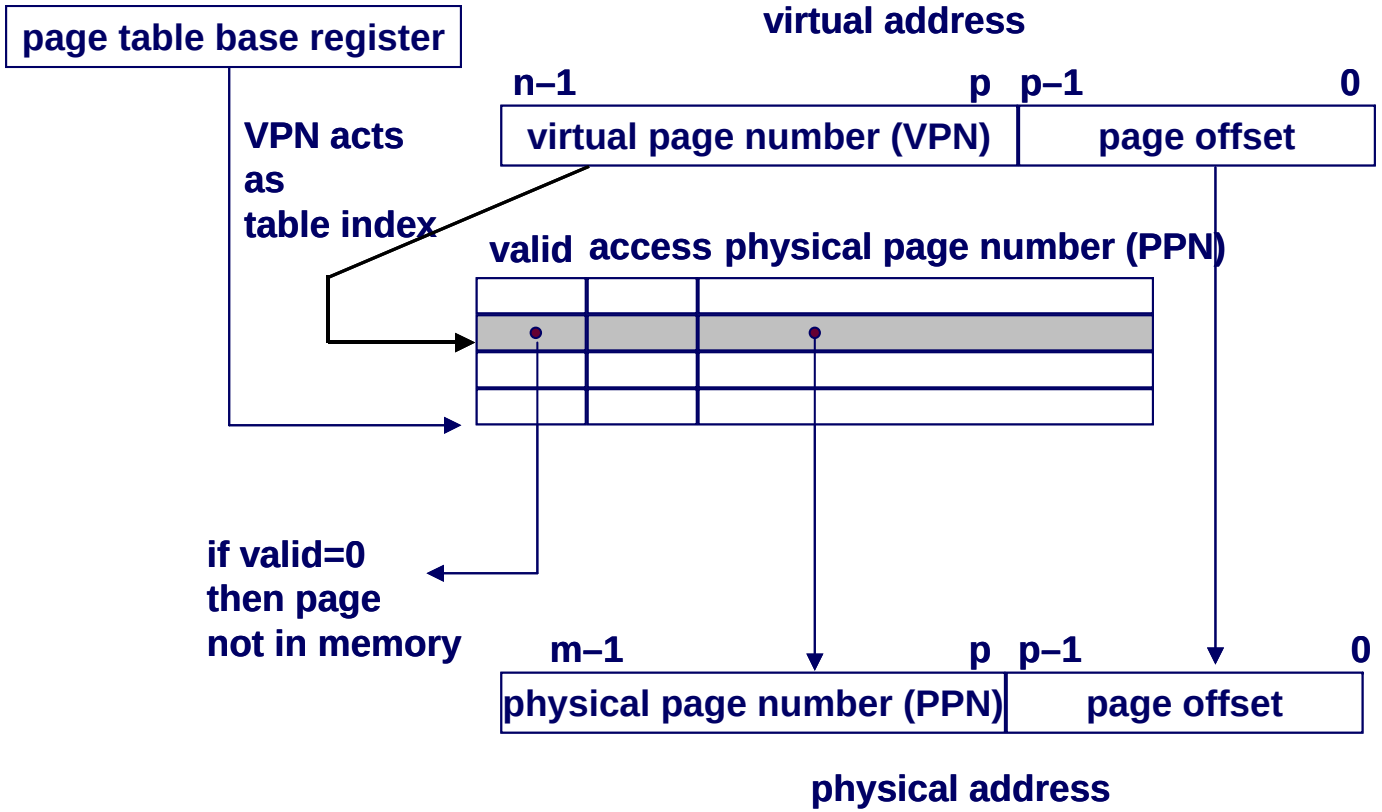


NOTE: http://csapp.cs.cmu.edu/public/lectures/class19.ppt

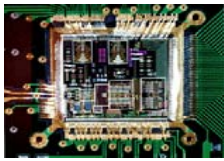


Page Table Operation

- Translation
  - Separate (set of) page table(s) per process.
  - VPN forms index into page table (points to a page table entry).

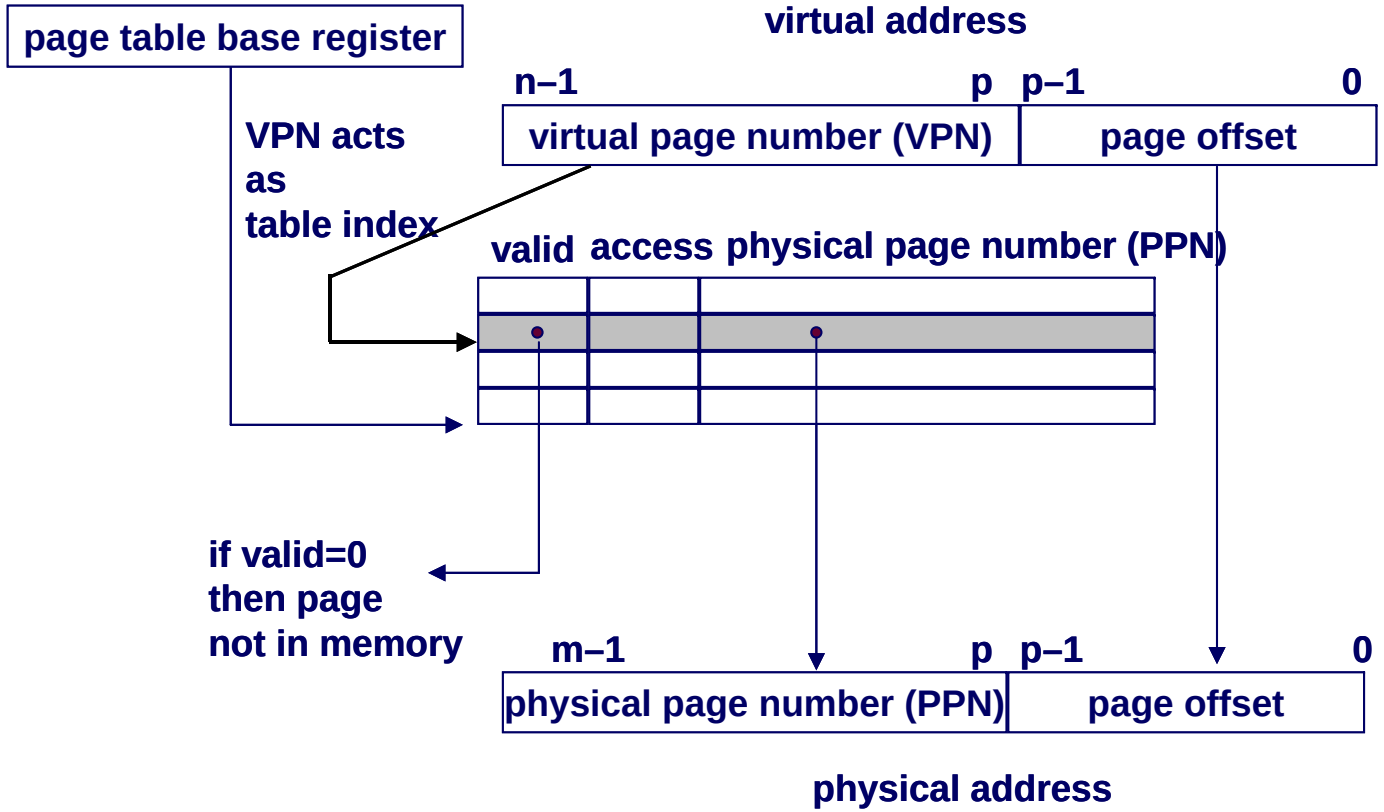


NOTE: http://csapp.cs.cmu.edu/public/lectures/class19.ppt

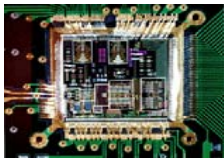


Page Table Operation

- Computing Physical Address
  - Page Table Entry (PTE) provides information about page
    - if (valid bit = 1) then the page is in memory.
      - Use physical page number (PPN) to construct address
    - if (valid bit = 0) then the page is on disk
      - Page fault

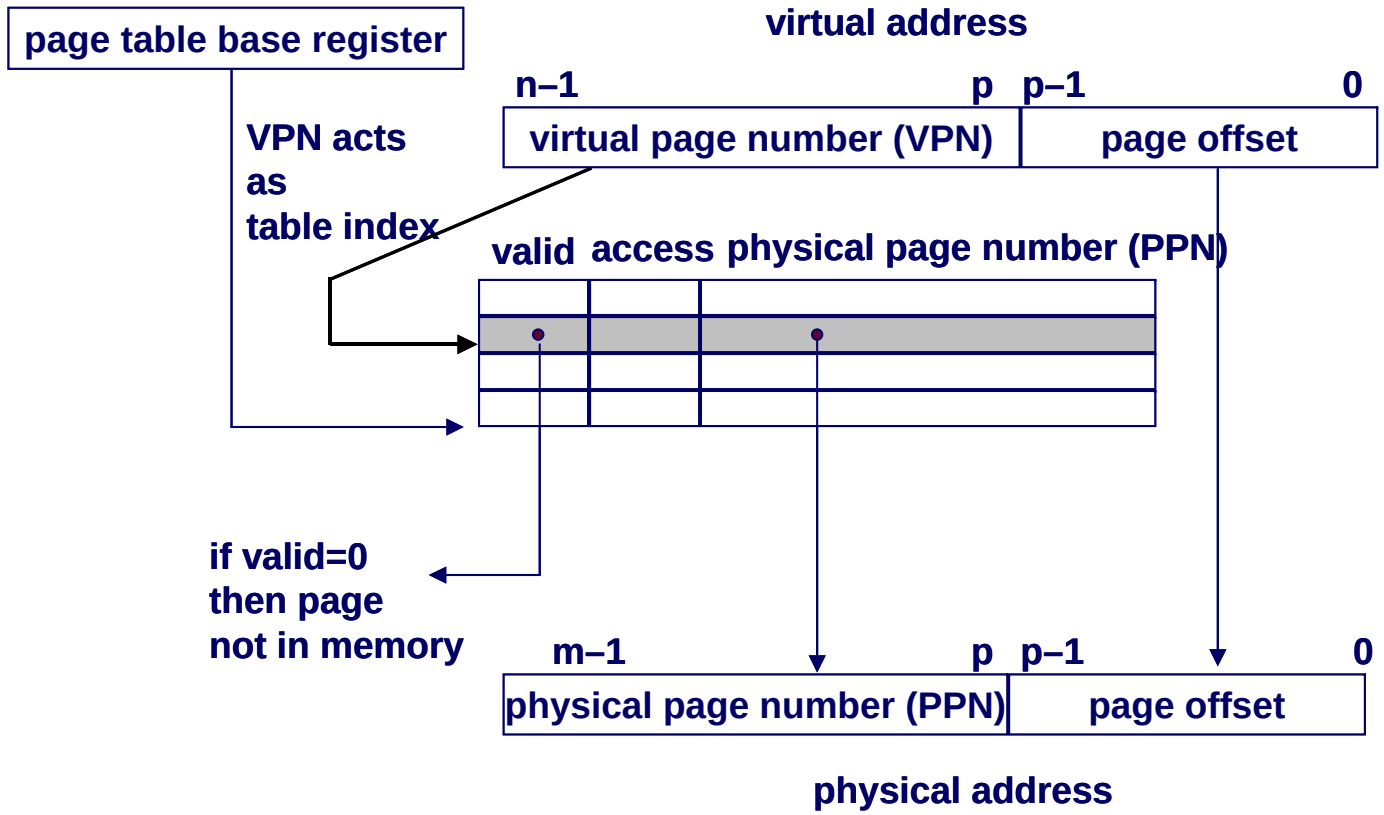


NOTE: http://csapp.cs.cmu.edu/public/lectures/class19.ppt

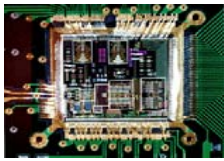


Page Table Operation

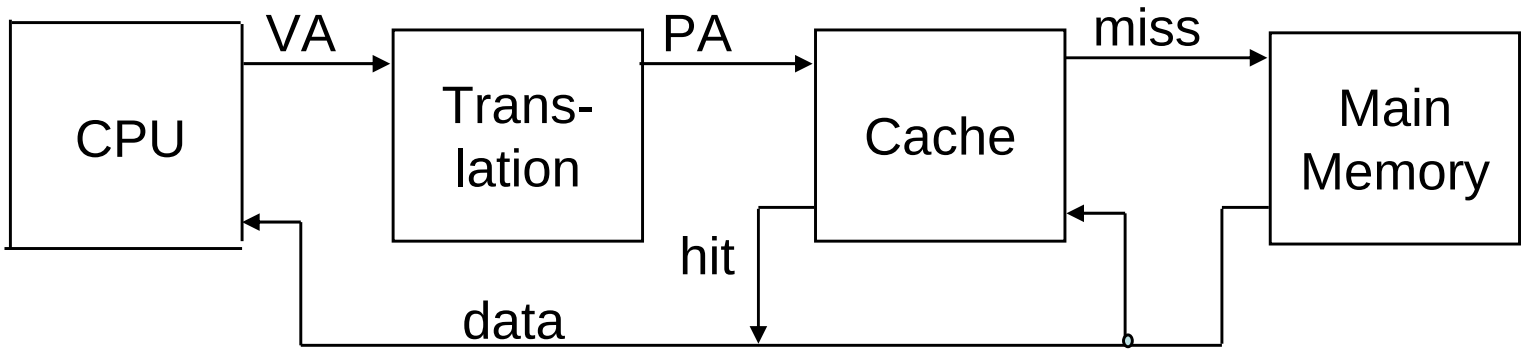
- Checking Protection
  - Access rights field indicate allowable access
    - e.g., read-only, read-write, execute-only
    - typically support multiple protection modes (e.g., kernel vs. user)
  - Protection violation fault if user doesn't have necessary permission



NOTE: http://csapp.cs.cmu.edu/public/lectures/class19.ppt

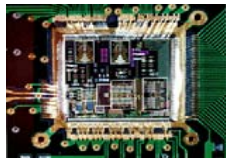


Integrating VM and Cache



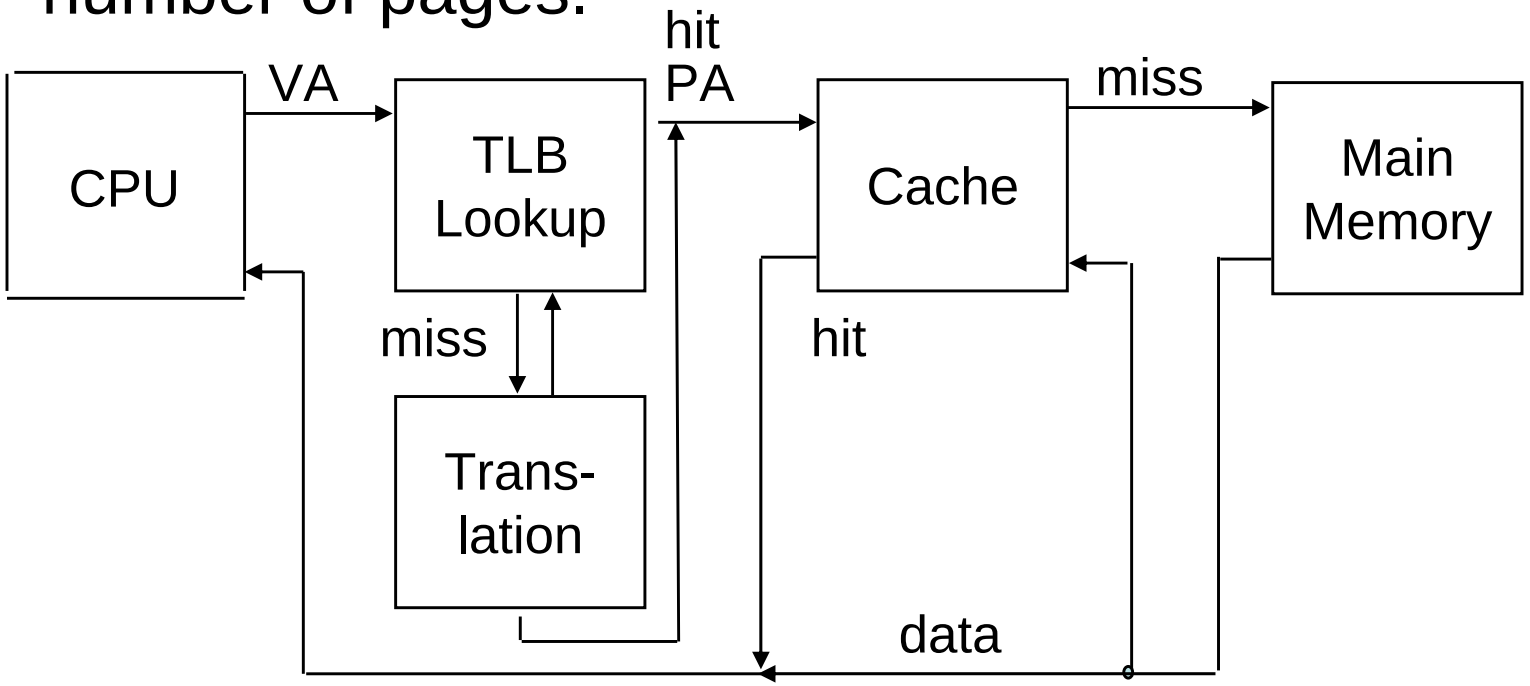
- Most Caches "Physically Addressed"
  - Accessed by physical addresses.
  - Allows multiple processes to have blocks in cache at same time.
  - Allows multiple processes to share pages.
  - Cache doesn't need to be concerned with protection issues.
    - Access rights checked as part of address translation.
- Perform Address Translation Before Cache Lookup.
  - But this could involve a memory access itself (of the PTE).
  - Of course, page table entries can also become cached.

NOTE: http://csapp.cs.cmu.edu/public/lectures/class19.ppt

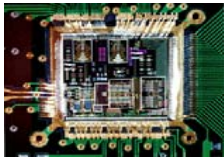


Speeding up Translation with a TLB

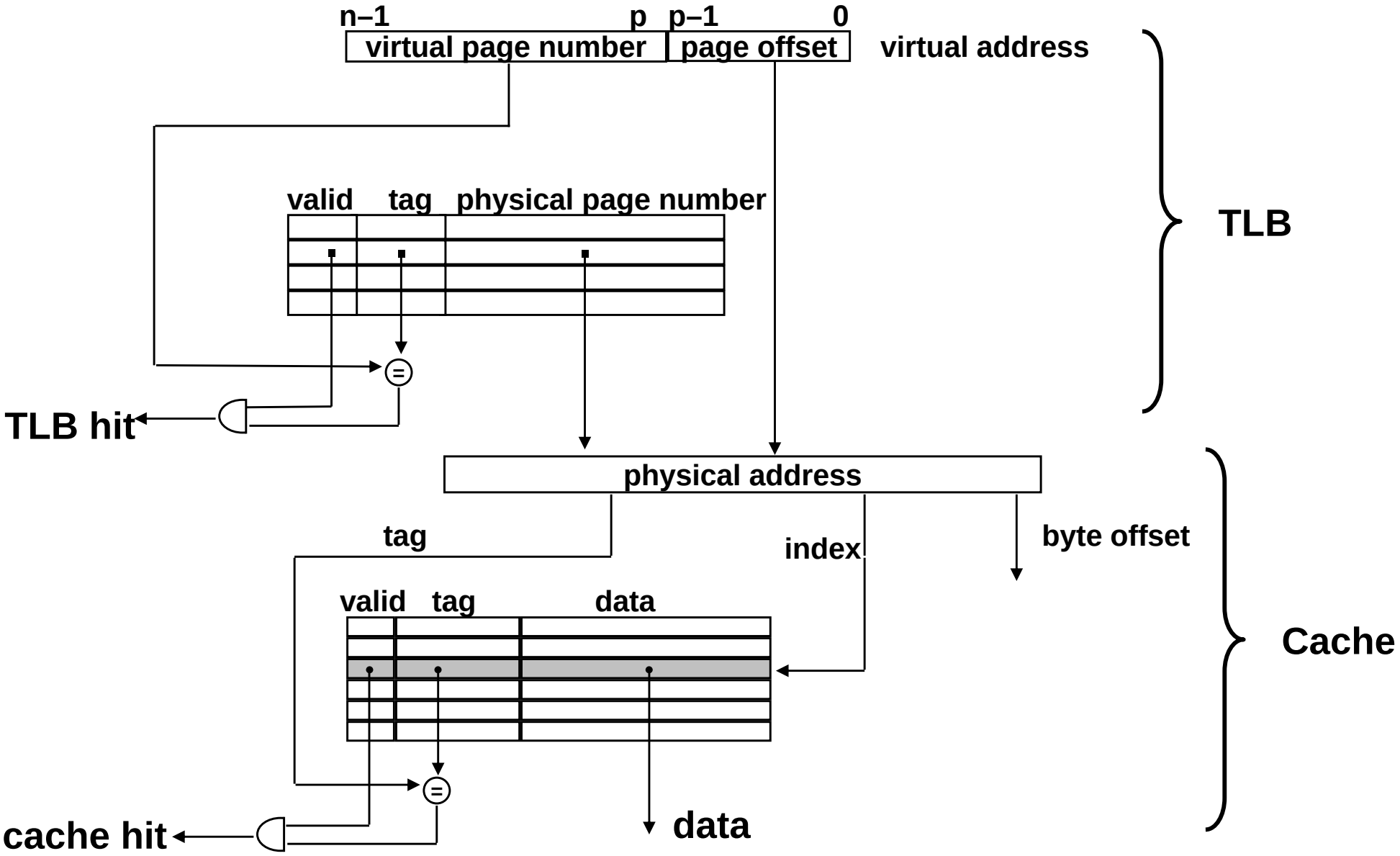
- "Translation Look Aside Buffer" (TLB)
  - Small hardware cache in MMU.
  - Maps virtual page numbers to physical page numbers.
  - Contains complete page table entries for small number of pages.



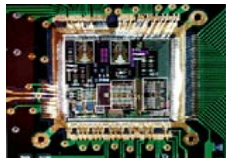
NOTE: http://csapp.cs.cmu.edu/public/lectures/class19.ppt



Address Translation with a TLB

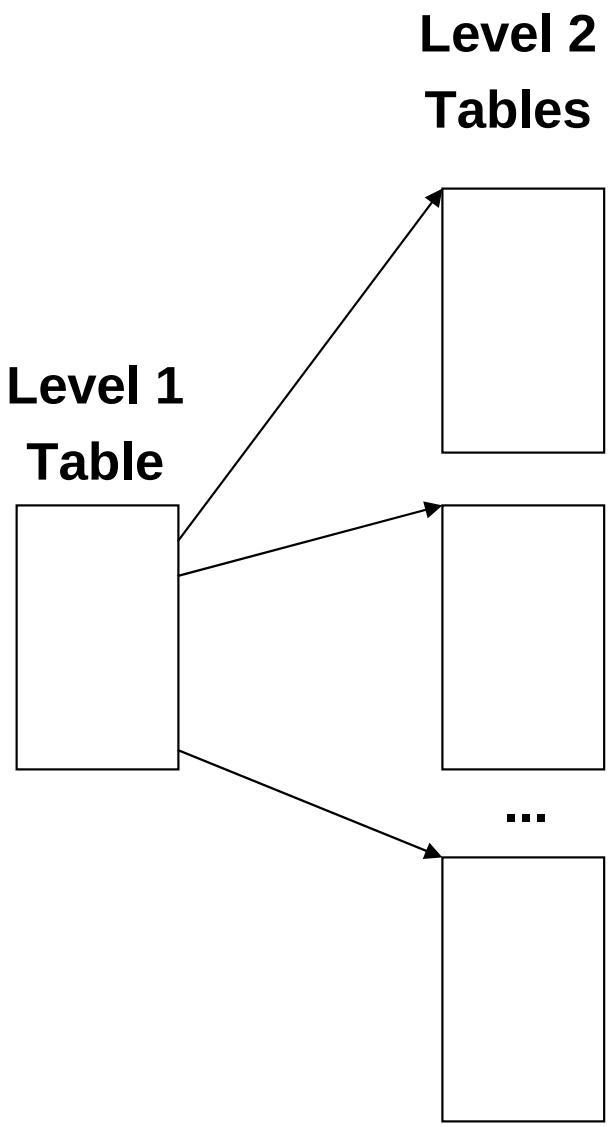


NOTE: http://csapp.cs.cmu.edu/public/lectures/class19.ppt



Multi-Level Page Tables

- Given:
  - 4KB (2^{12}) page size
  - 32-bit address space
  - 4-byte PTE
- Problem:
  - Would need a 4 MB page table!
    - 2^{20} *4 bytes
- Common solution
  - multi-level page tables
  - e.g., 2-level table (P6)
    - Level 1 table: 1024 entries, each of which points to a Level 2 page table.
    - Level 2 table: 1024 entries, each of which points to a page.

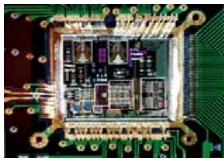


NOTE: http://csapp.cs.cmu.edu/public/lectures/class19.ppt

Virtual Memory: Main Themes

- Programmer's View
  - Large "flat" address space
    - Can allocate large blocks of contiguous addresses.
  - Processor "owns" machine
    - Has private address space.
    - Unaffected by behavior of other processes.
- System View
  - User virtual address space created by mapping to set of pages.
    - Need not be contiguous
    - Allocated dynamically
    - Enforce protection during address translation
  - OS manages many processes simultaneously.
    - Continually switching among processes
    - Especially when one must wait for resource
      - E.g., disk I/O to handle page fault

NOTE: http://csapp.cs.cmu.edu/public/lectures/class19.ppt



Summary

- Memory hierarchy concept exploits spatial and temporal properties exhibited by the code.
- Levels of memory are organized such that:
  - The average cost/bit approaches the cost of the cheapest memory technology.

AND

  - The average word access time approaches the access time of the fastest memory technology.
- Direct-mapped cache organization is the simplest.
- Miss rate can be reduced by resorting to set-associativity (set-associative caches).
- As the block size increases, typically the miss rate goes down!
- Multi-level caches can be used to further improve the performance of the memory system.
- Remember: Motto of Memory Hierarchy

Right amount of data at the right place all the time!!