
CS152 – Computer Architecture and Engineering

Lecture 13 – Cache Part I

2003-10-07

Dave Patterson

(www.cs.berkeley.edu/~patterson)

www-inst.eecs.berkeley.edu/~cs152/



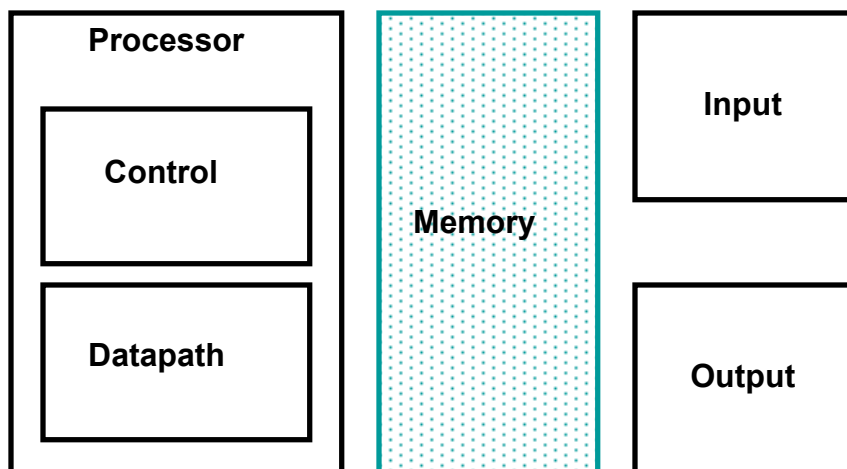
Review

- Exceptions, Interrupts handled as unplanned procedure calls
- Control adds arcs to check for exceptions, new states to adjust PC, set CPU status
- OS implements interrupt/exception policy (priority levels) using Interrupt Mask
- For pipelining, interrupts need to be precise (like multicycle)
- Control design can reduce to Microprogramming
- Control is more complicated with:
 - complex instruction sets
 - restricted datapaths



The Big Picture: Where are We Now?

- The Five Classic Components of a Computer



Technology Trends

	Capacity	Speed (latency)
Logic:	2x in 3 years	2x in 3 years
DRAM:	4x in 4 years	2x in 10 years
Disk:	4x in 2 years	2x in 10 years

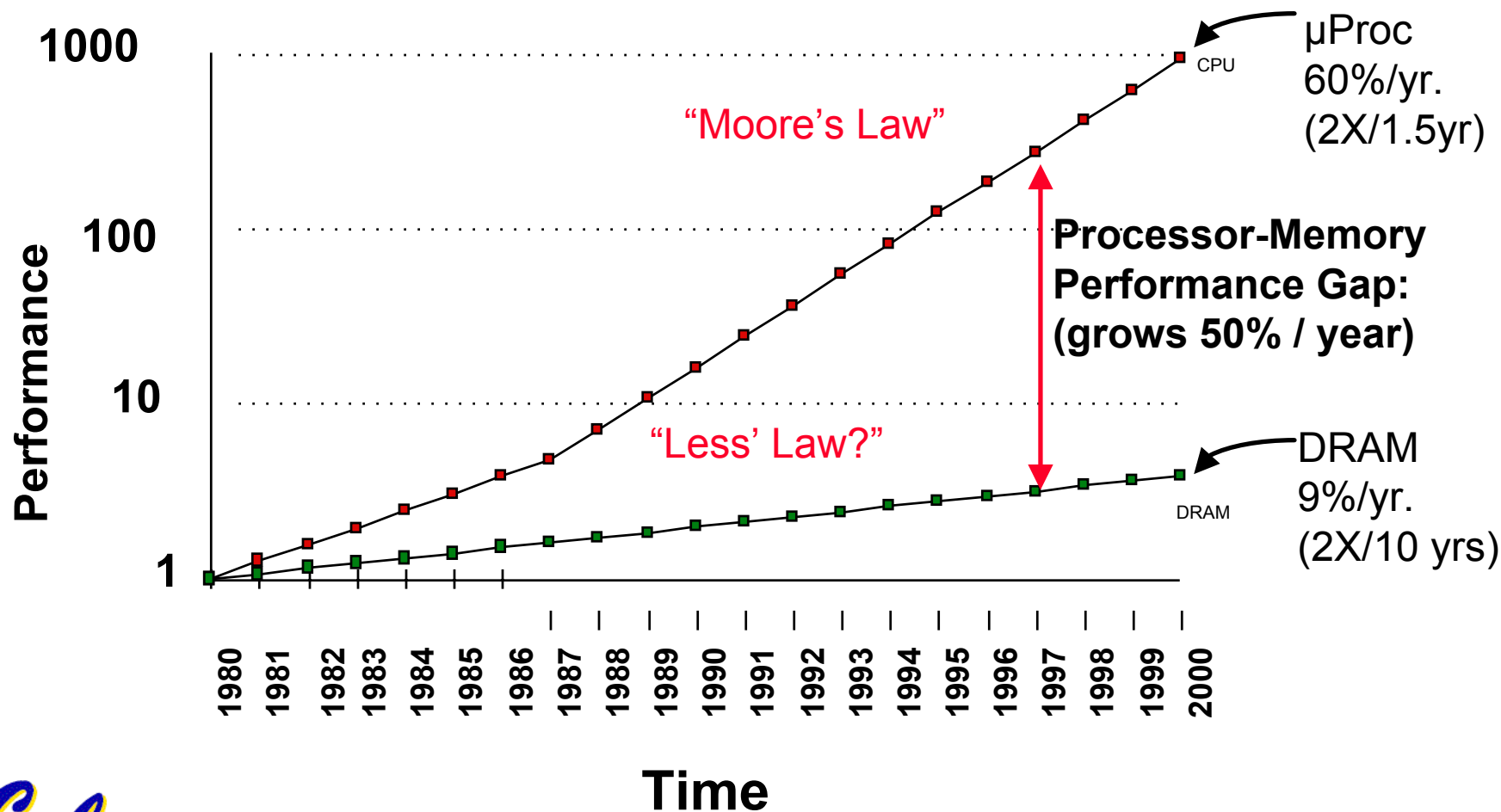
DRAM		
Year	Size	Cycle Time
1980	64 Kb	250 ns
1983	256 Kb	220 ns
1986	1 Mb	190 ns
1989	4 Mb	165 ns
1992	16 Mb	145 ns
1996	64 Mb	120 ns
1998	128 Mb	100 ns
2000	256 Mb	80 ns
2002	512 Mb	60 ns

1000:1! (Size growth from 1980 to 2000)
3:1! (Cycle Time reduction from 1980 to 2000)



Who Cares About the Memory Hierarchy?

Processor-DRAM Memory Gap (latency)



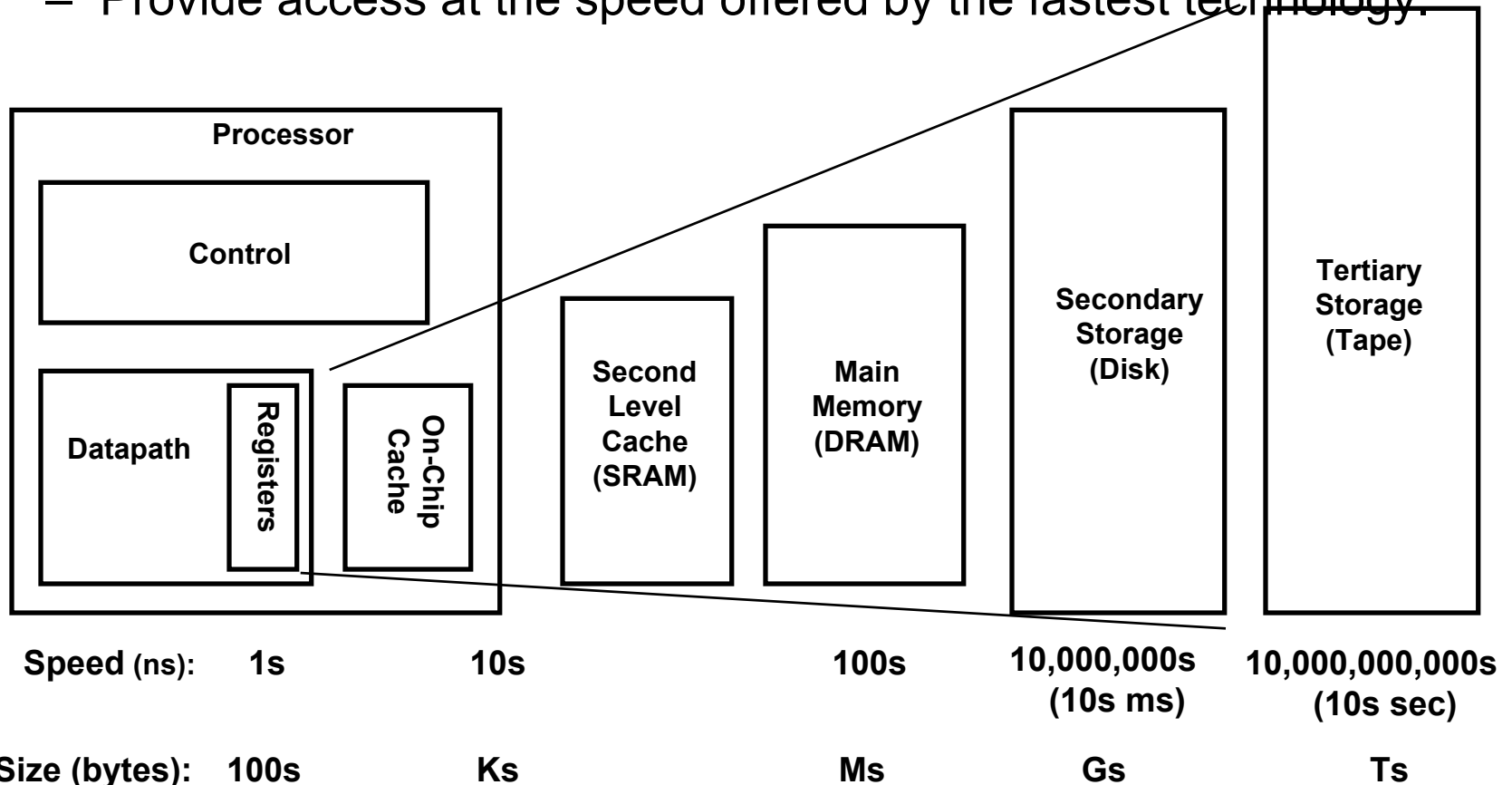
The Goal: illusion of large, fast, cheap memory

- Fact:
 - Large memories are slow
 - Fast memories are small
- How do we create a memory that is large, cheap and fast (most of the time)?
 - Hierarchy
 - Parallelism



Memory Hierarchy of a Modern Computer System

- By taking advantage of the principle of locality:
 - Present the user with as much memory as is available in the cheapest technology.
 - Provide access at the speed offered by the fastest technology.



Recap: Memory Hierarchy Technology

- Random Access:

- “Random” is good: access time is the same for all locations

- **DRAM**: Dynamic Random Access Memory

- High density, low power, cheap, slow
 - Dynamic: need to be “refreshed” regularly
 - 1 Transistor per memory cell

- **SRAM**: Static Random Access Memory

- Low density, high power, expensive, fast
 - Static: content will last “forever”(until lose power)
 - 6 transistors per memory cell
 - SRAM cell about 10X larger than DRAM cell

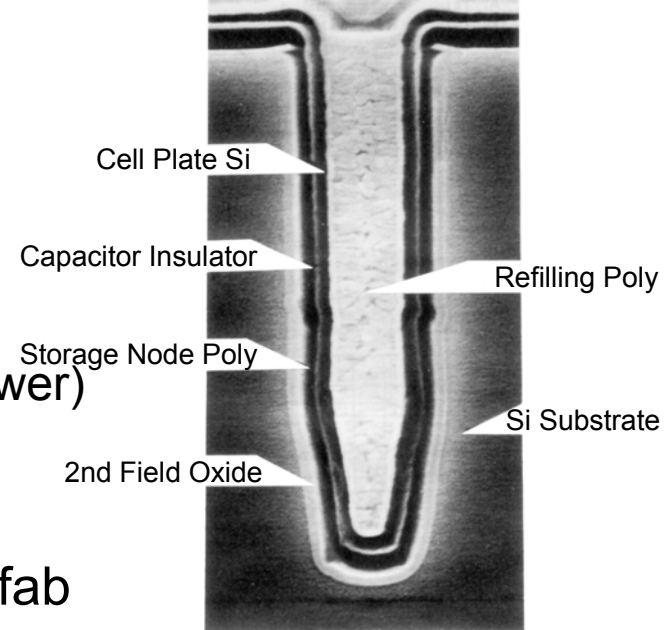
- SRAM in logic fab process, DRAM has own fab

- “Non-so-random” Access Technology:

- Access time varies from location to location and from time to time
 - Examples: Disk, CDROM, DRAM page-mode access

- Sequential Access Technology: access time linear in location (e.g., Tape)

DRAM trench cell



DRAM Design Goals

- Die size, testing time, yield => profit
 - Yield >> 60%
(redundant rows/columns to repair flaws)
- 3 phases: engineering samples, first customer ship(FCS), mass production
 - Fastest to FCS, mass production wins share
- Sell 10% of a single DRAM generation



What does “Synchronous” RAM mean?

- Take basic RAMs (SRAM and DRAM) and add clock:
 - Gives SSRAM or SDRAM (Synchronous SRAM/DRAM)
 - Addresses and Control set up ahead of time, clock edges activate
- More complicated, on-chip controller
 - Operations synchronized to clock
 - So, give row address one cycle
 - Column address some number of cycles later (say 2)
 - Data comes out later (say 2 cycles later)
 - Burst modes
 - Typical might be 1,2,4,8, or 256 length burst
 - Thus, only give RAS and CAS once for all of these accesses
 - Multi-bank operation (on-chip interleaving)
 - Lets you overlap startup latency (5 cycles above) of two banks
- Careful of timing specs!
 - 10ns (100 MHz) SDRAM may still require 60ns to get first data!
 - 60ns DRAM means first data out in 60ns

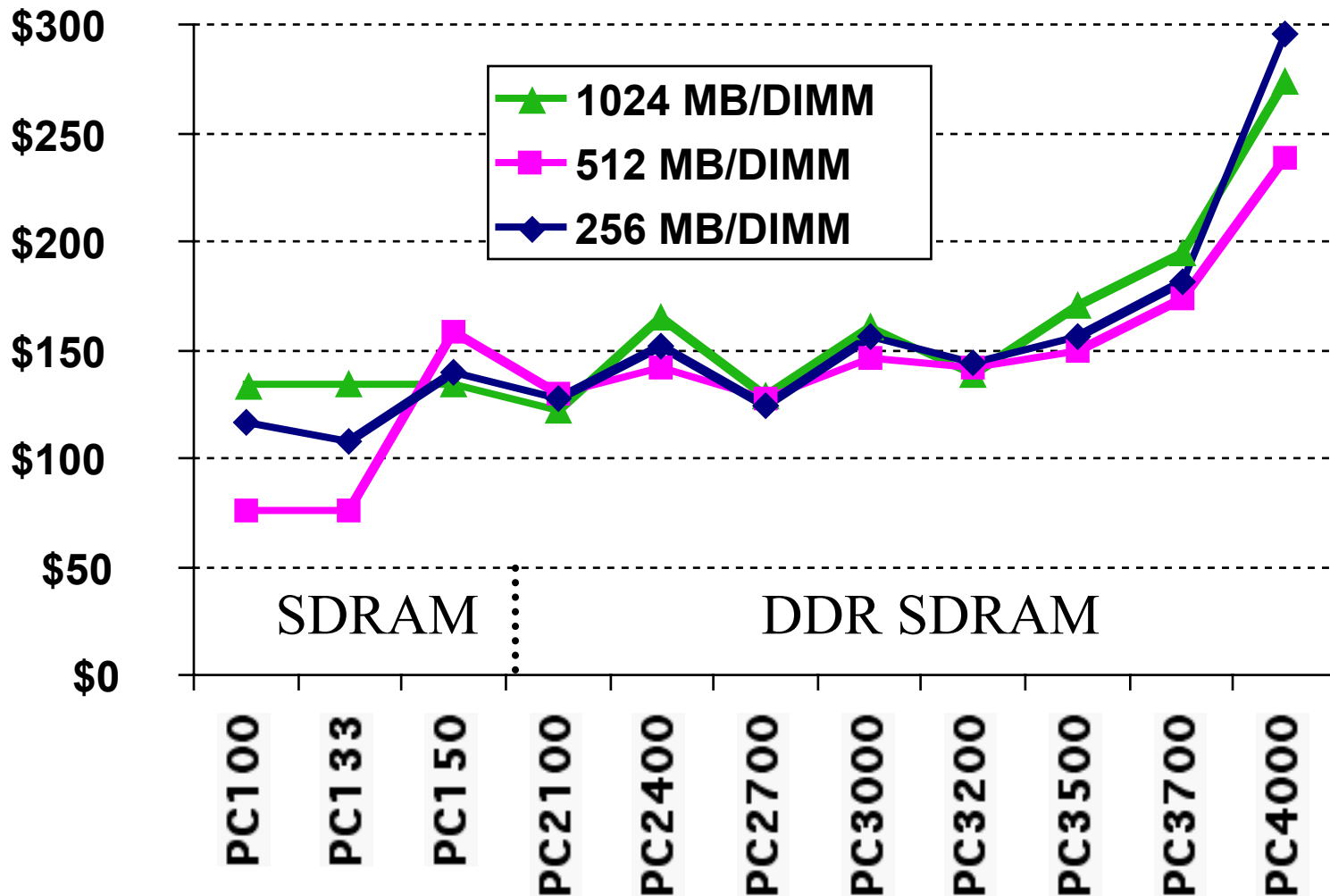


What about DDR, PC3200, ...?

- SDRAM
 - bus clock 100 MHz, 150 MHz, ...
 - DIMM width = 8 bytes
 - BW: clock x 8 bytes = 800 MB/sec, 1200 MB/s,...
 - SDRAM DIMMs called PC100, PC 150,... (clock)
- Double Data Rate (DDR) SDRAM
 - transfer on both rising edge and falling edge
 - bus clock 100 MHz, 150 MHz, 200 MHz, ...
 - BW: clock x 2 x 8 bytes = 1600, 2400, 3200 MB/s
 - DDR SDRAM DIMMs called PC1600, PC2400, PC3200 (peak BW)



\$/GB: Pay for Bandwidth, Capacity/DIMM



- 2X to 4X for highest vs. lowest BW/DIMM
- ~1.1X to ~1.2X for biggest capacity

DRAM History

- DRAMs: capacity +60%/yr, cost –30%/yr
 - Before: 2.5X cells/area, 1.5X die size in -3 years
- DRAM fab line costs \$2B to \$3B
 - DRAM only: density, leakage v. speed
- Commodity, second source industry
=> high volume, low profit, conservative
 - Little organization innovation in 20 years
page mode, EDO, Synch DRAM, DDR SDRAM
- Order of importance: 1) Cost/bit 1a) Capacity
 - RAMBUS: 10X BW, +30% cost => little impact

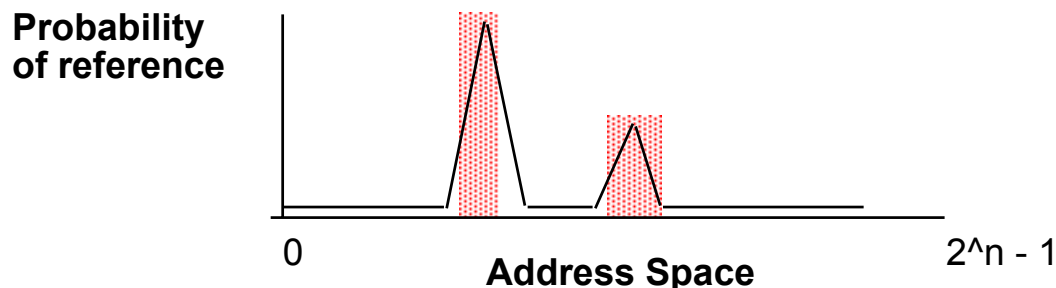


Administrivia

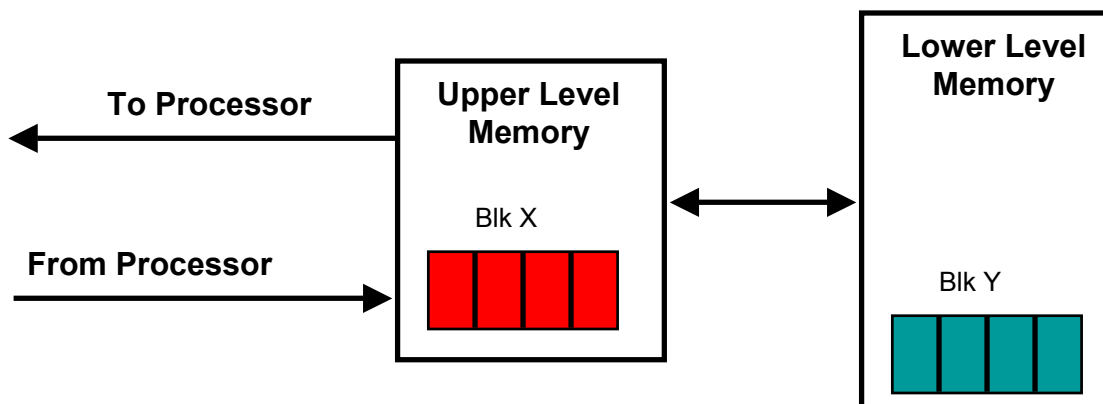
- Lab 4 demo Mon 10/13, write up Tue 10/14
- Midterm Wed Oct 8 5:30 - 8:30 in 1 LeConte
 - Bring 1 page (both sides), handwritten notes
 - Meet at LaVal's Northside afterwards for Pizza
 - No lecture Thursday Oct 9
 - Through last week lecture (Chapters 1 to 6)
- Office hours
 - Mon 4 – 5:30 Jack, Tue 3:30-5 Kurt,
Wed 3 – 4:30 John, Thu 3:30-5 Ben
 - Dave's office hours Tue 3:30 – 5



Memory Hierarchy: Why Does it Work? Locality!



- **Temporal Locality** (Locality in Time):
=> Keep most recently accessed data items closer to the processor
- **Spatial Locality** (Locality in Space):
=> Move blocks consists of contiguous words to the upper levels



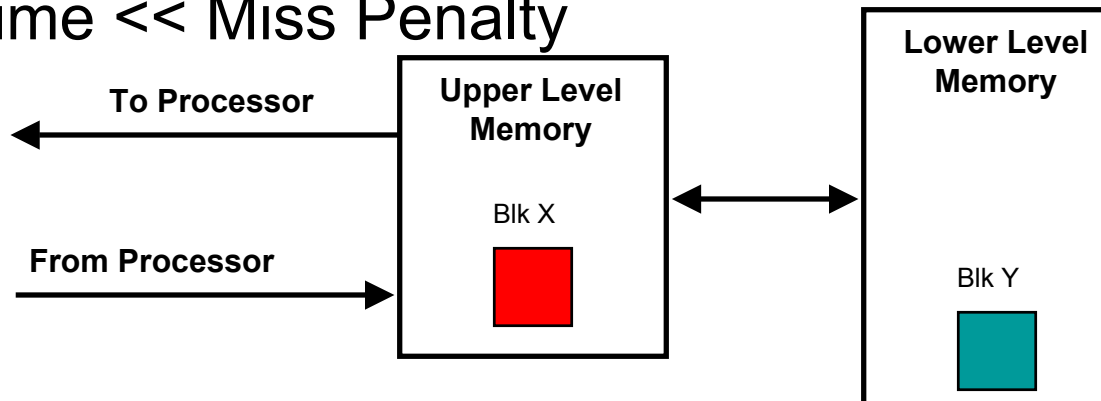
How is the hierarchy managed?

- Registers \leftrightarrow Memory
 - by compiler
- cache \leftrightarrow memory
 - by the hardware
- memory \leftrightarrow disks
 - by the hardware and operating system (virtual memory)
 - by the programmer (files)



Memory Hierarchy: Terminology

- **Hit**: data appears in some block in the upper level (example: Block X)
 - **Hit Rate**: the fraction of memory access found in the upper level
 - **Hit Time**: Time to access the upper level which consists of
RAM access time + Time to determine hit/miss
- **Miss**: data needs to be retrieve from a block in the lower level (Block Y)
 - **Miss Rate** = $1 - (\text{Hit Rate})$
 - **Miss Penalty**: Time to replace a block in the upper level +
Time to deliver the block the processor
- Hit Time \ll Miss Penalty



Recap: Cache Performance

- CPU time = (CPU execution clock cycles + Memory stall clock cycles) x clock cycle time
- Memory stall clock cycles =
(Reads x Read miss rate x Read miss penalty +
Writes x Write miss rate x Write miss penalty)
- Memory stall clock cycles =
Memory accesses x Miss rate x Miss penalty
- Different measure: AMAT

Average Memory Access time (AMAT) =
Hit Time + (Miss Rate x Miss Penalty)

Note: *memory hit time is included in execution cycles*



Recap: Impact on Performance

- Suppose a processor executes at
 - Clock Rate = 2 GHz (0.5 ns per cycle)
 - Base CPI = 1.1
 - 50% arith/logic, 30% ld/st, 20% control
- Suppose that 10% of memory operations get 50 cycle miss penalty (very optimistic)
- Suppose that 1% of instructions get same miss penalty
- $$\begin{aligned}\text{CPI} &= \text{Base CPI} + \text{average stalls per instruction} \\ &= 1.1(\text{cycles/ins}) + \\ &\quad [0.30 (\text{DataMops/ins}) \\ &\quad \quad \times 0.10 (\text{miss/DataMop}) \times 50 (\text{cycle/miss})] + \\ &\quad [1 (\text{InstMop/ins}) \\ &\quad \quad \times 0.01 (\text{miss/InstMop}) \times 50 (\text{cycle/miss})] \\ &= (1.1 + 1.5 + .5) \text{ cycle/ins} = 3.1\end{aligned}$$
- 58% time (2.0/3.1) proc is stalled waiting for memory!
- $$\text{AMAT} = (1/1.3) \times [1 + 0.01 \times 50] + (0.3/1.3) \times [1 + 0.10 \times 50] = 2.54$$

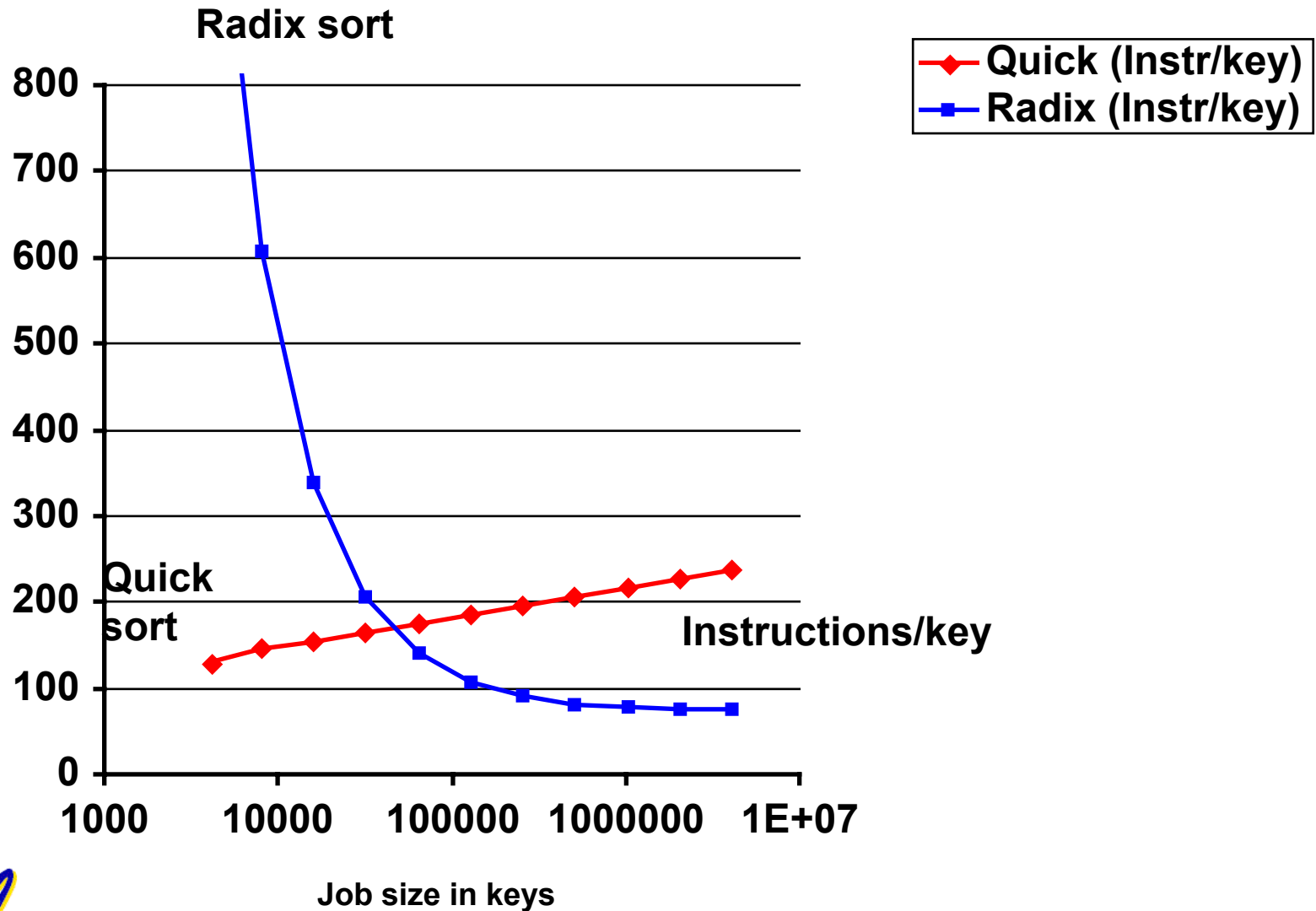


Impact of Memory Hierarchy on Algorithms

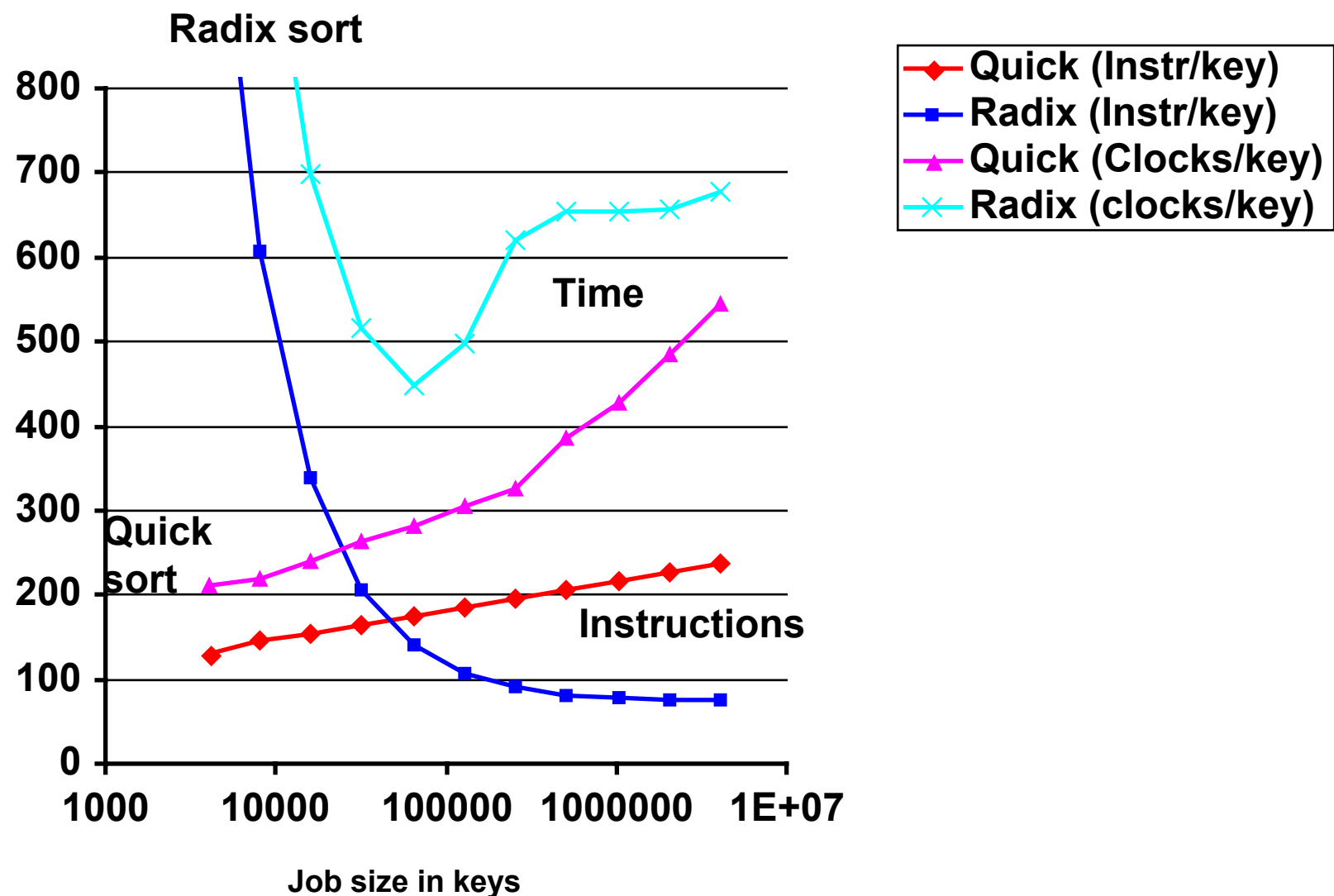
- Today CPU time is a function of (ops, cache misses)
- What does this mean to Compilers, Data structures, Algorithms?
 - Quicksort:
fastest comparison based sorting algorithm when keys fit in memory
 - Radix sort: also called “linear time” sort
For keys of fixed length and fixed radix a constant number of passes over the data is sufficient independent of the number of keys
- “The Influence of Caches on the Performance of Sorting” by A. LaMarca and R.E. Ladner. *Proceedings of the Eighth Annual ACM-SIAM Symposium on Discrete Algorithms*, January, 1997, 370-379.
 - 32 byte blocks, direct mapped L2 2MB cache, 8 byte keys, from 4000 to 4,000,000 elements (keys)



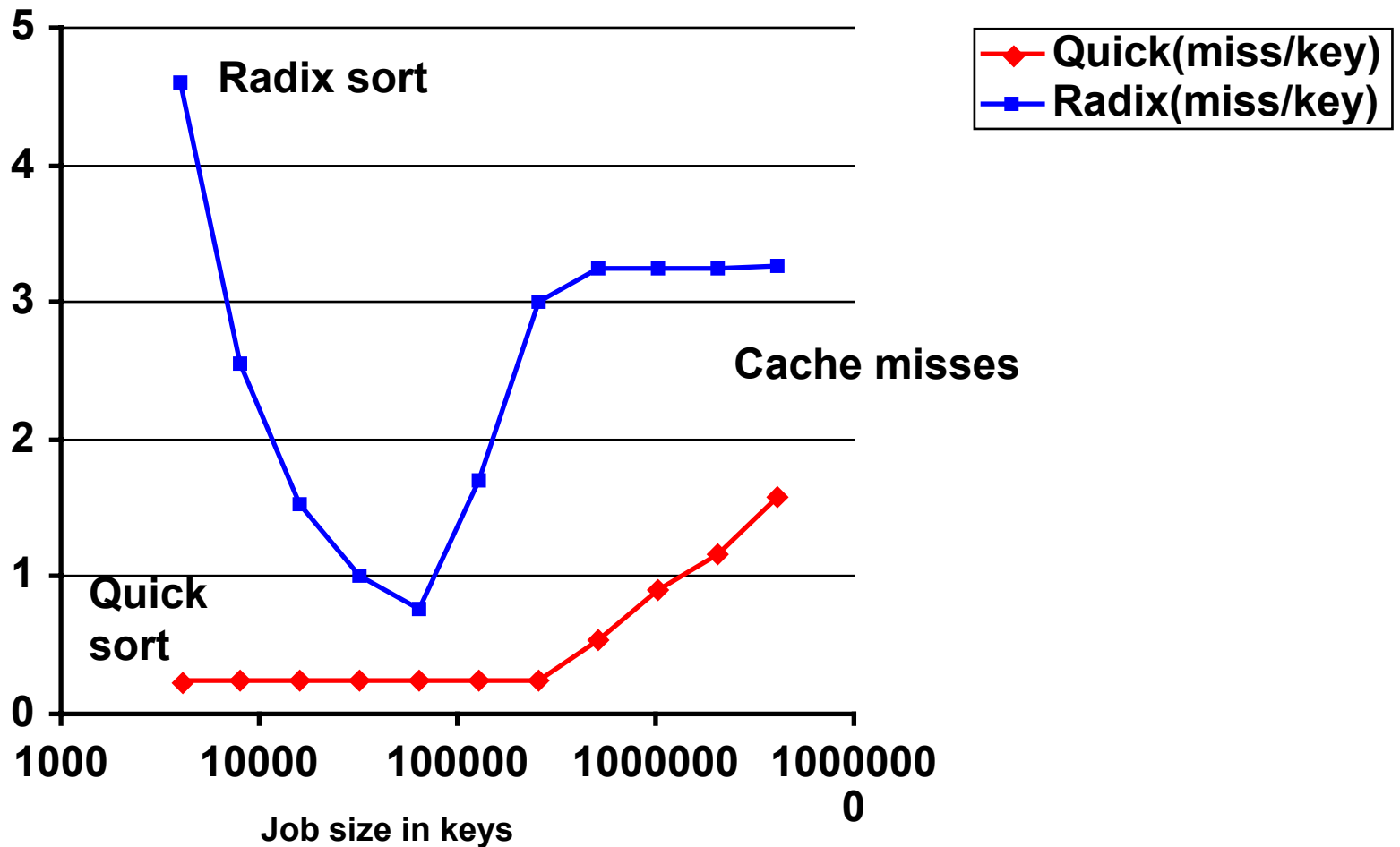
Quicksort vs. Radix as vary number keys: Instructions



Quicksort vs. Radix as vary number keys: Instrs & Time



Quicksort vs. Radix as vary number keys: Cache misses

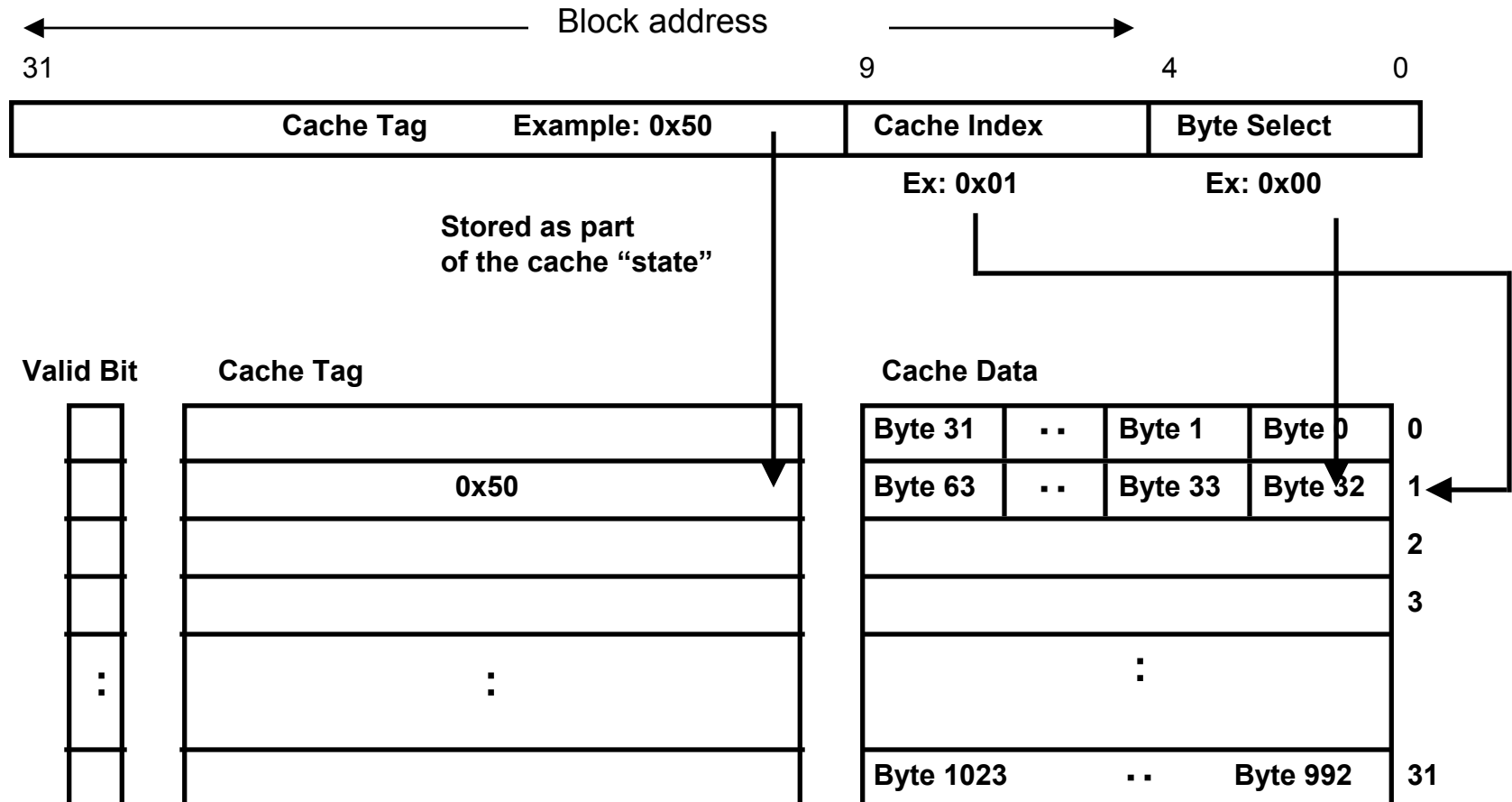


What is proper approach to fast algorithms?



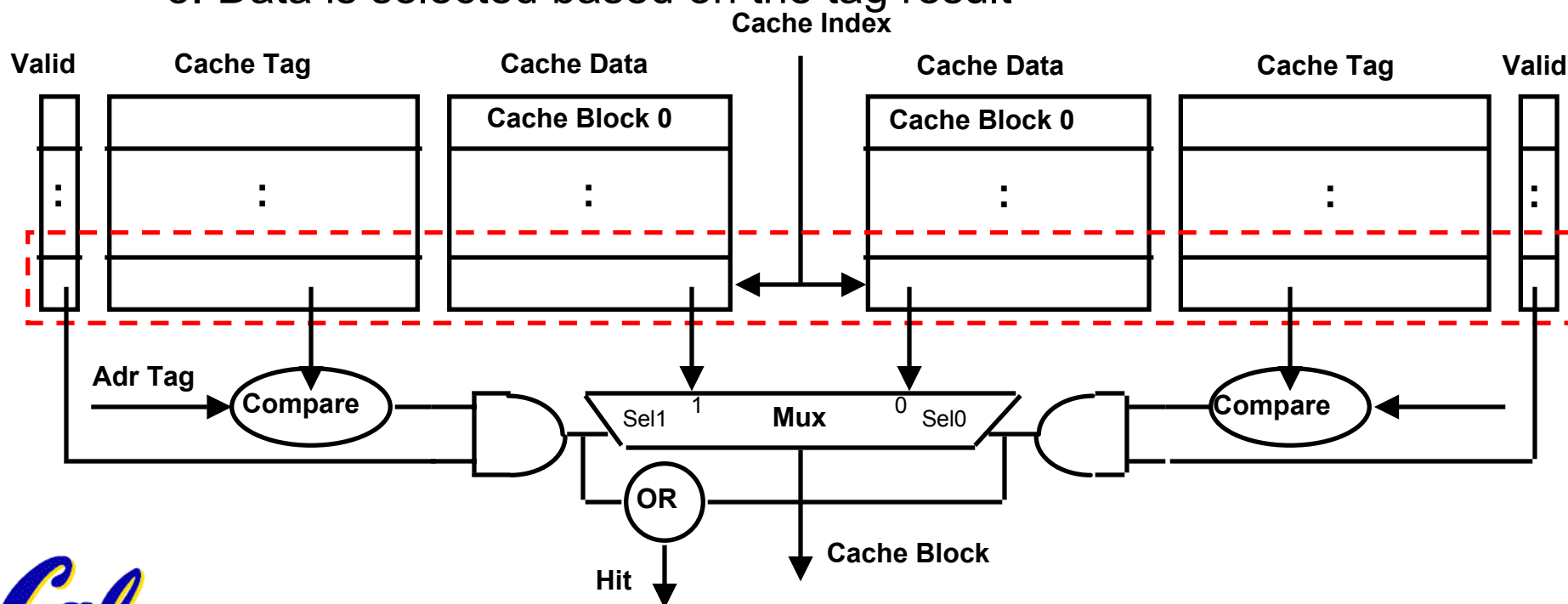
Example: 1 KB Direct Mapped Cache with 32 B Blocks

- For a 2^N byte cache:
 - The uppermost $(32 - N)$ bits are always the Cache Tag
 - The lowest M bits are the Byte Select (Block Size = 2^M)
 - One cache miss, pull in complete “Cache Block” (or “Cache Line”)



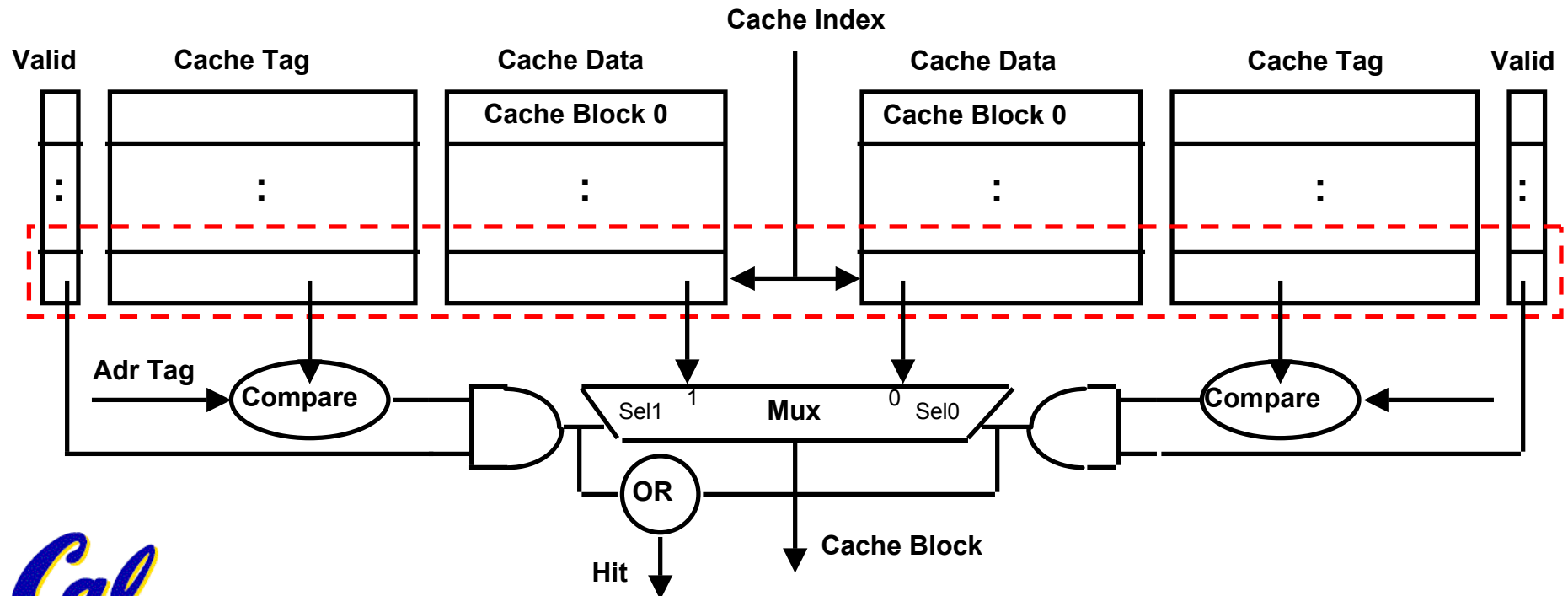
Example: Set Associative Cache

- **N-way set associative**: N entries for each Cache Index
 - N direct mapped caches operates in parallel
- Example: Two-way set associative cache
 1. Cache Index selects a “set” from the cache
 2. The two tags in the set are compared to the input in parallel
 3. Data is selected based on the tag result



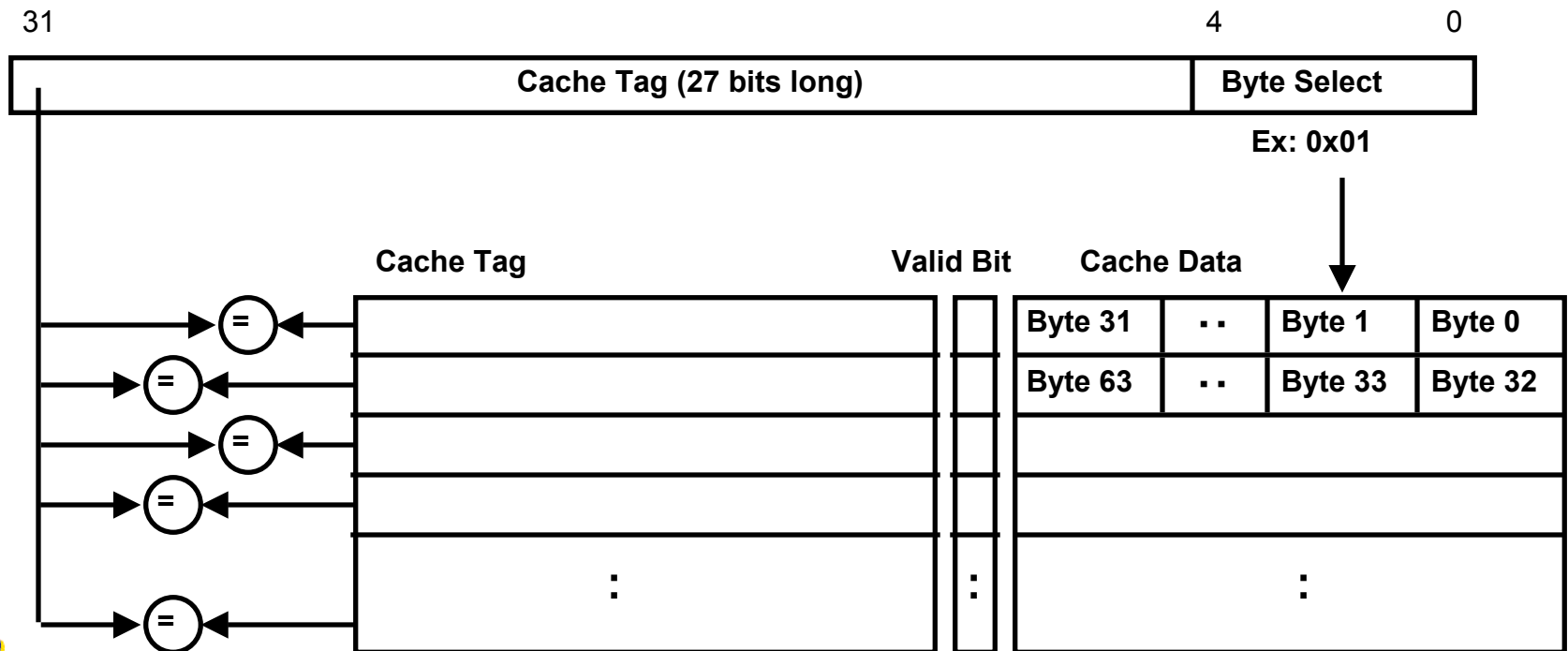
Disadvantage of Set Associative Cache

- N-way Set Associative Cache versus Direct Mapped Cache:
 - N comparators vs. 1
 - Extra MUX delay for the data
 - Data comes **AFTER** Hit/Miss decision and set selection
- In a direct mapped cache, Cache Block is available **BEFORE** Hit/Miss:
 - Possible to assume a hit and continue. Recover later if miss.



Example: Fully Associative

- **Fully Associative Cache**
 - Forget about the Cache Index
 - Compare the Cache Tags of all cache entries in parallel
 - Example: Block Size = 32 B blocks, we need N 27-bit comparators
- (By definition: Conflict Miss = 0 for a fully associative cache)



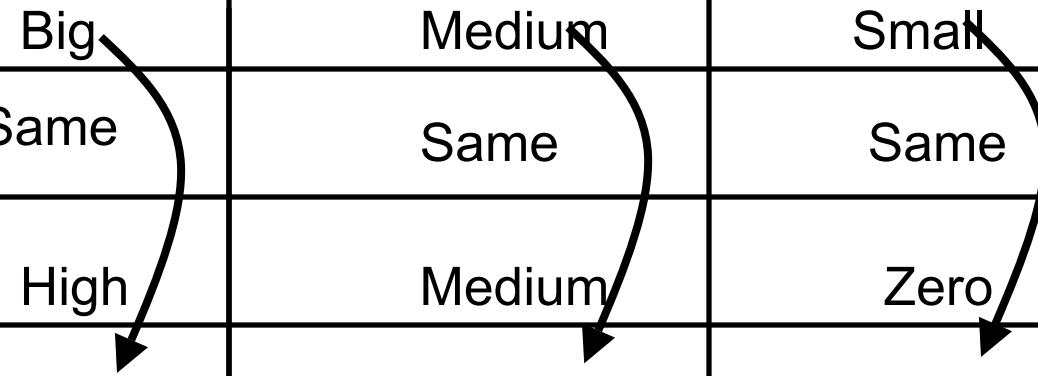
A Summary on Sources of Cache Misses

- **Compulsory** (cold start or process migration, first reference): first access to a block
 - “Cold” fact of life: not a whole lot you can do about it
 - Note: If you are going to run “billions” of instruction, Compulsory Misses are insignificant
- **Capacity**:
 - Cache cannot contain all blocks access by the program
 - Main Solution: increase cache size
- **Conflict** (collision):
 - Multiple memory locations mapped to the same cache location
 - Main Solution: increase associativity
- **Coherence** (Invalidation): other process (e.g., I/O) updates memory



Design options at constant cost

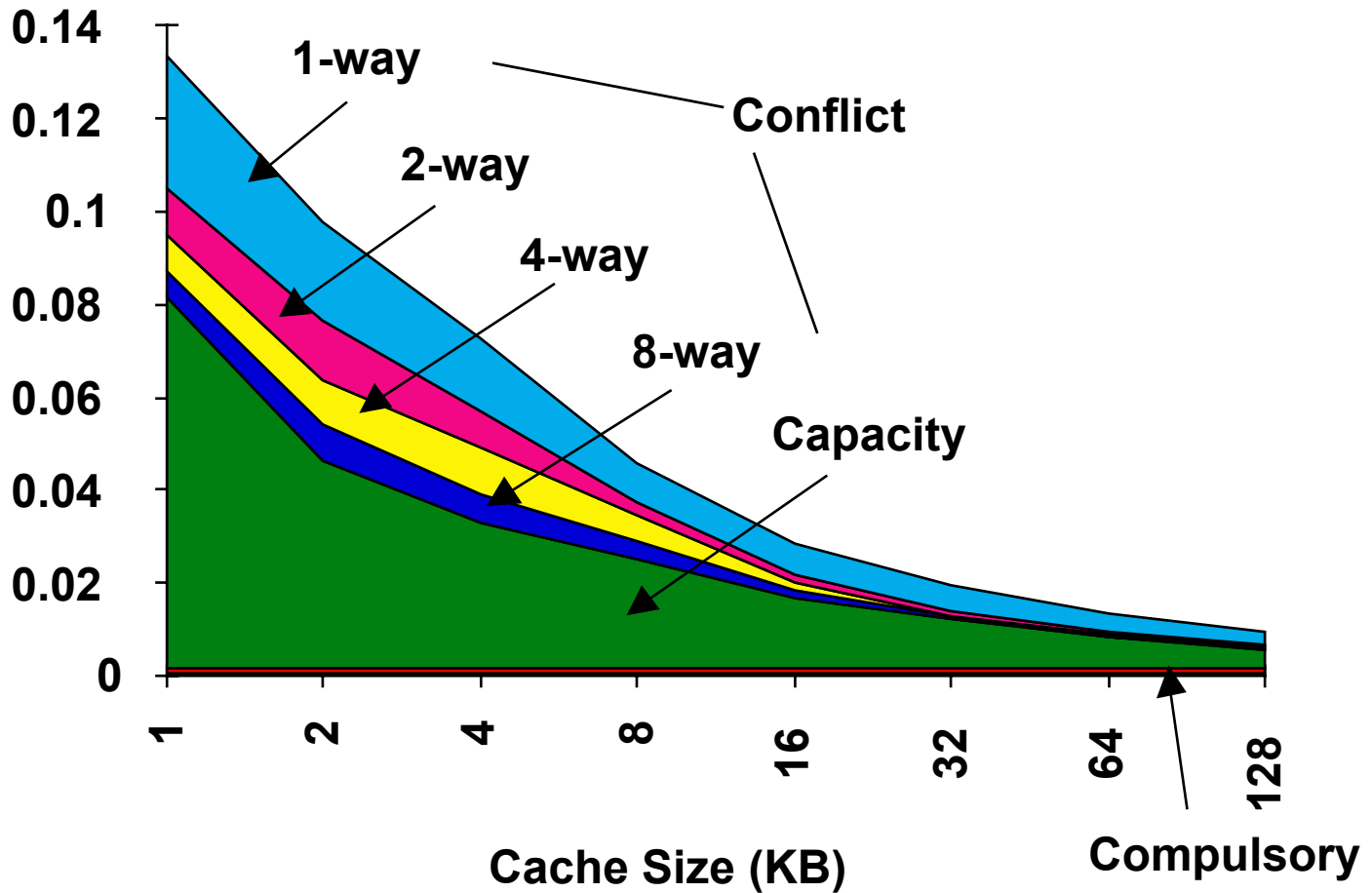
	Direct Mapped	N-way Set Associative	Fully Associative
Cache Size	Big	Medium	Small
Compulsory Miss	Same	Same	Same
Conflict Miss	High	Medium	Zero
Capacity Miss	Low	Medium	High
Coherence Miss	Same	Same	Same



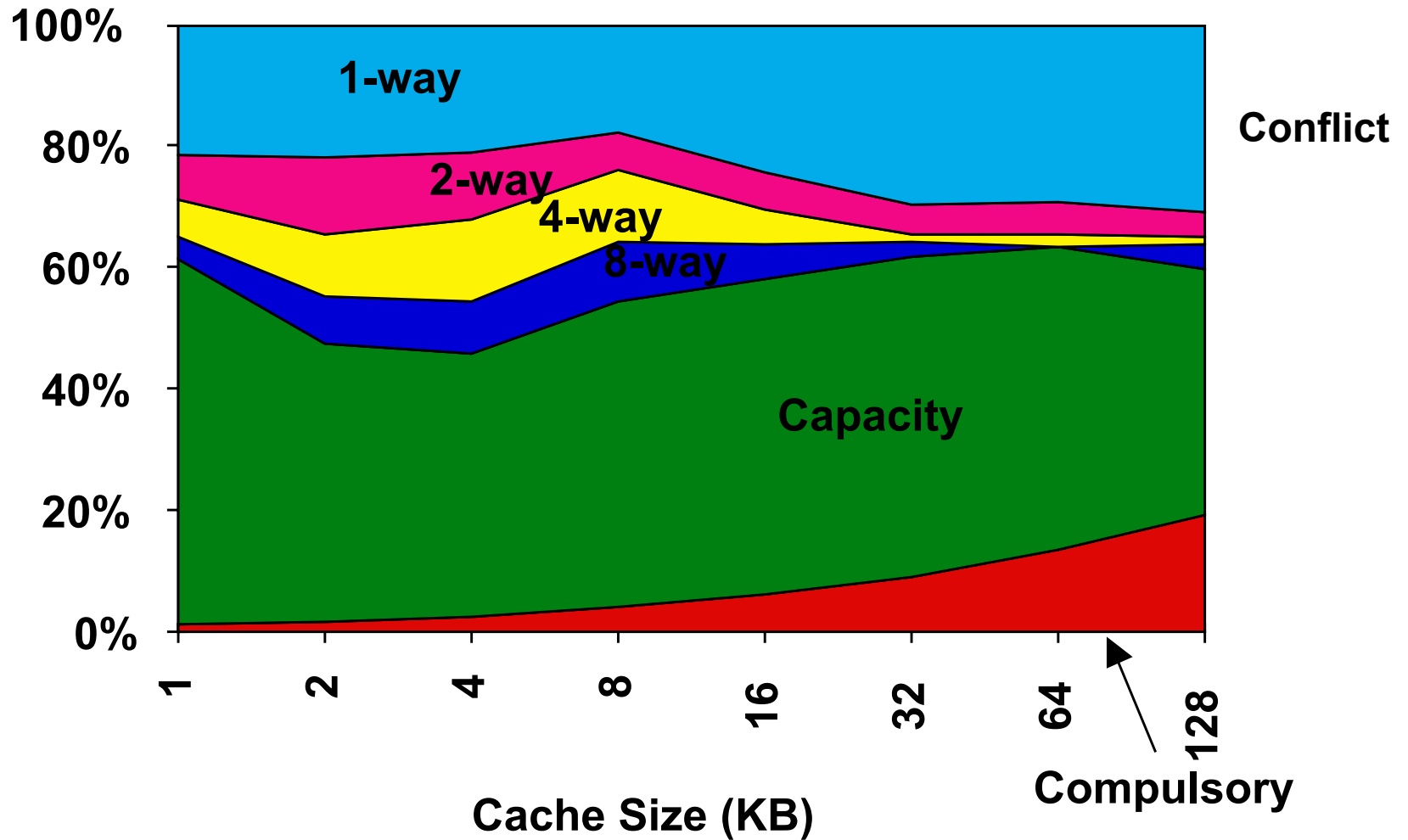
Note:

If you are going to run “billions” of instruction, Compulsory Misses are insignificant (except for streaming media types of programs).

3Cs Absolute Miss Rate (SPEC92)



3Cs Relative Miss Rate



Peer Review: 4Cs model

- Which are true?

A: You cannot reduce compulsory misses
(hence the name)

B: To get rid of a particular conflict miss,
you must increase cache associativity

C: To reduce capacity misses, you must
increase cache size

- | | |
|-------------|-------------|
| 1. ABC: FFF | 5. ABC: TFF |
| 2. ABC: FFT | 6. ABC: TFT |
| 3. ABC: FTF | 7. ABC: TTF |
| 4. ABC: FTT | 8. ABC: TTT |



Recap: Four Questions for Caches and Memory Hierarch

- Q1: Where can a block be placed in the upper level? (*Block placement*)
- Q2: How is a block found if it is in the upper level? (*Block identification*)
- Q3: Which block should be replaced on a miss? (*Block replacement*)
- Q4: What happens on a write? (*Write strategy*)



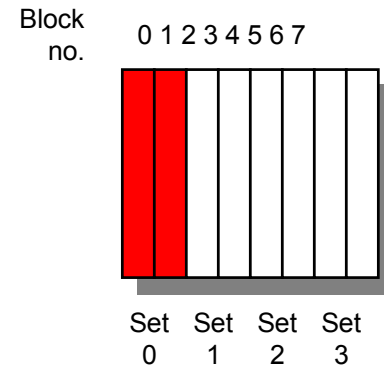
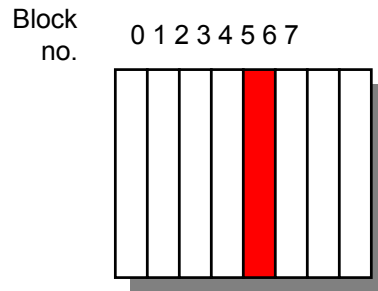
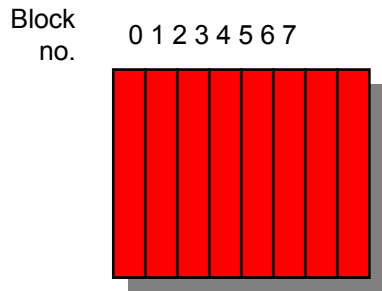
Q1: Where can a block be placed in the upper level?

- Block 12 placed in 8 block cache:
 - Fully associative, direct mapped, 2-way set associative
 - S.A. Mapping = Block Number Modulo Number Sets

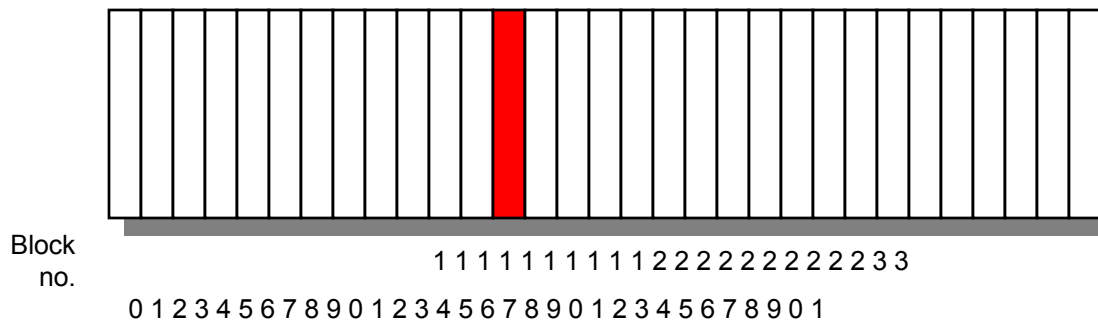
Fully associative:
block 12 can go anywhere

Direct mapped:
block 12 can go only into
block 4 ($12 \bmod 8$)

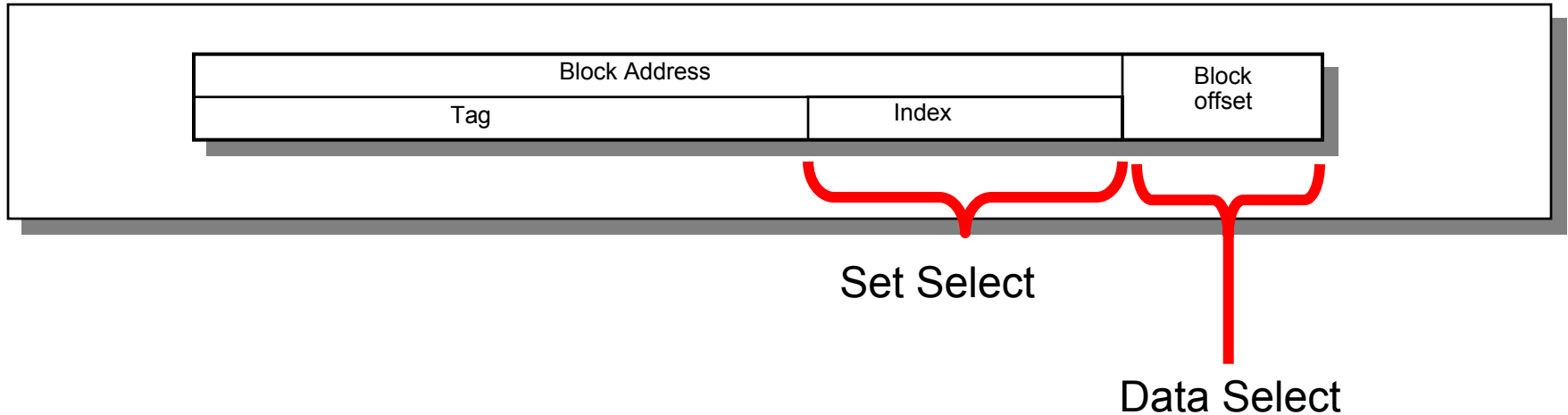
Set associative:
block 12 can go anywhere
in set 0 ($12 \bmod 4$)



Block-frame address



Q2: How is a block found if it is in the upper level?



- Direct indexing (using index and block offset), tag compares, or combination
- Increasing associativity shrinks index, expands tag

Q3: Which block should be replaced on a miss?

- Easy for Direct Mapped
- Set Associative or Fully Associative:
 - Random
 - LRU (Least Recently Used)

Associativity:	2-way		4-way		8-way	
Size	LRU	Random	LRU	Random	LRU	Random
16 KB	5.2%	5.7%	4.7%	5.3%	4.4%	5.0%
64 KB	1.9%	2.0%	1.5%	1.7%	1.4%	1.5%
256 KB	1.15%	1.17%	1.13%	1.13%	1.12%	1.12%

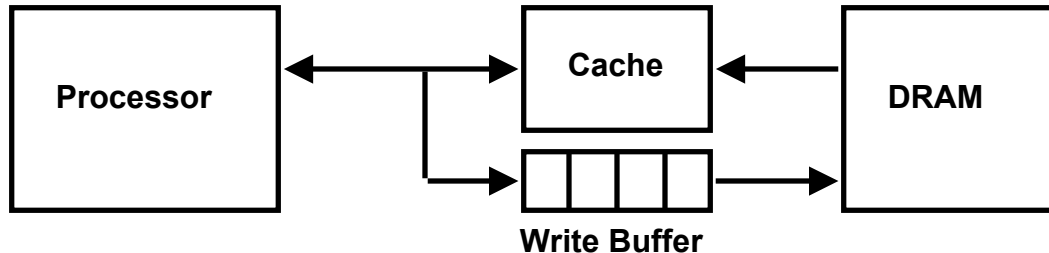


Q4: What happens on a write?

- Write through—The information is written to both the block in the cache and to the block in the lower-level memory.
- 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.
 - is block clean or dirty?
- Pros and Cons of each?
 - WT: read misses cannot result in writes
 - WB: no writes of repeated writes
- **WT much easier to debug, design** (hence lab 5/6)
- WT always combined with write buffers so that don't wait for lower level memory



Write Buffer for Write Through



- A Write Buffer is needed between the Cache and Memory
 - Processor: writes data into the cache and the write buffer
 - Memory controller: write contents of the buffer to memory
- Write buffer is just a FIFO:
 - Typical number of entries: 4
 - **Must handle bursts of writes**
 - Works fine if: Store frequency (w.r.t. time) $\ll 1 / \text{DRAM write cycle}$

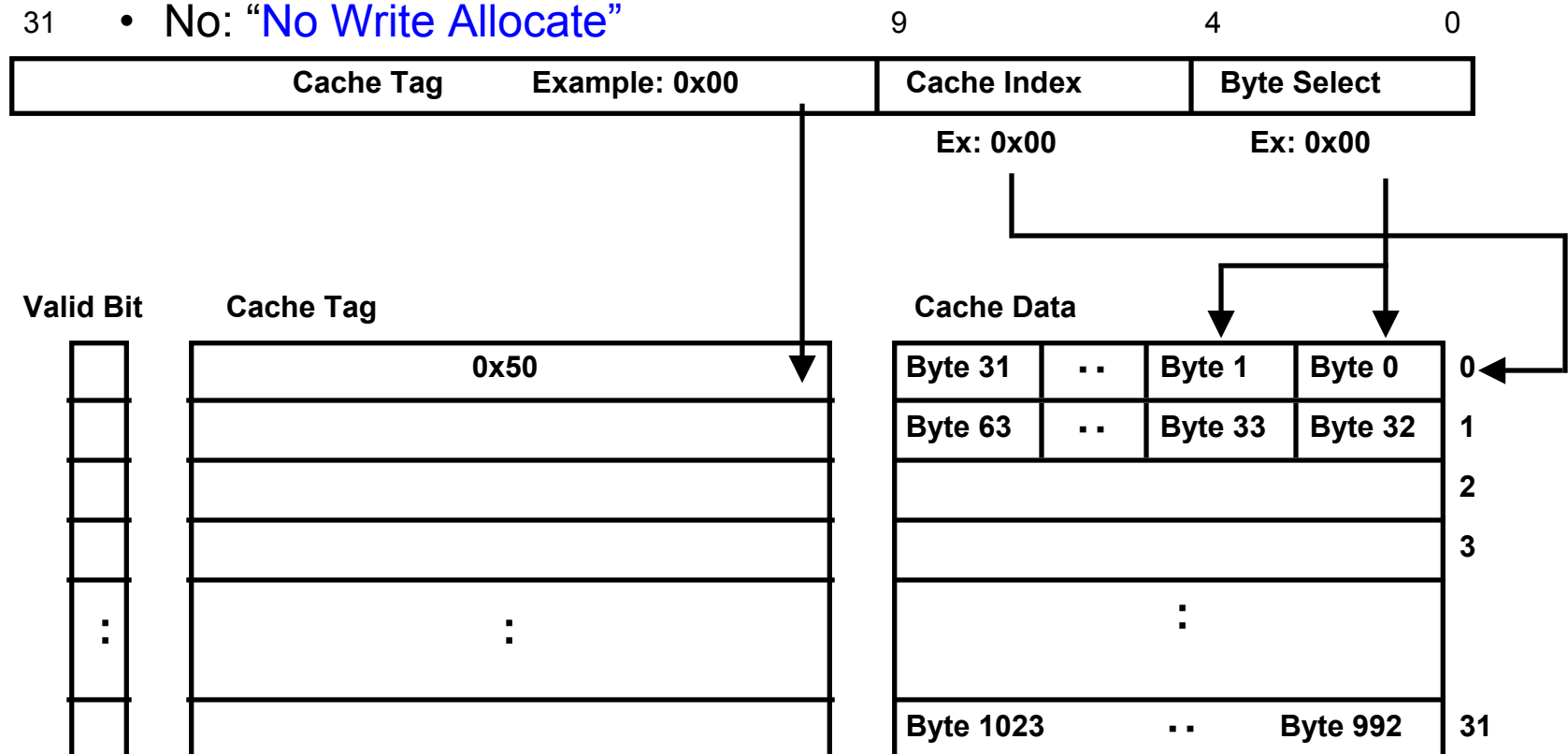
RAW Hazards from Write Buffer!

- Write-Buffer Issues: Could introduce RAW (Read After Write) Hazard with memory!
 - Write buffer may contain *only* copy of valid data \Rightarrow
Reads to memory may get wrong result if we ignore write buffer
- Solutions:
 - Simply wait for write buffer to empty before servicing reads:
 - Might increase read miss penalty (old MIPS 1000 by 50%)
 - Check write buffer contents before read (“fully associative”);
 - If no conflicts, let the memory access continue
 - Else grab data from buffer
- Can Write Buffer help with Write Back?
 - Read miss replacing dirty block
 - Copy dirty block to write buffer *while* starting read to memory
 - CPU stall less since restarts as soon as do read



Write-miss Policy: Write Allocate versus Not Allocate

- Assume: a 16-bit write to memory location 0x0 and causes a miss
 - Do we allocate space in cache and possibly read in the block?
 - Yes: “Write Allocate”
 - No: “No Write Allocate”



Peer Review: Oxymorons and Caches

- Which combinations are oxymorons (e.g., military intelligence)?

A: LRU replacement, direct mapped cache

B: Write back cache with a write buffer

C: Write through cache with write allocate

1. ABC: NNN

5. ABC: YNN

2. ABC: NNY

6. ABC: YNY

3. ABC: NYN

7. ABC: YYN

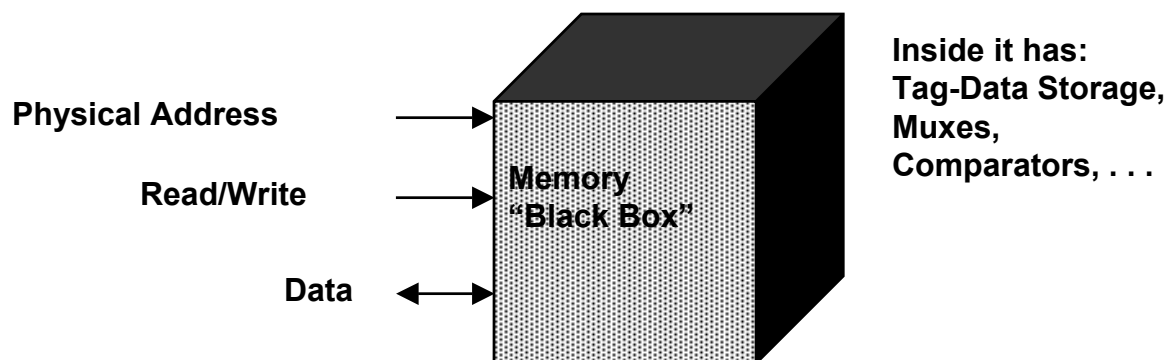
4. ABC: NYY

8. ABC: YYY

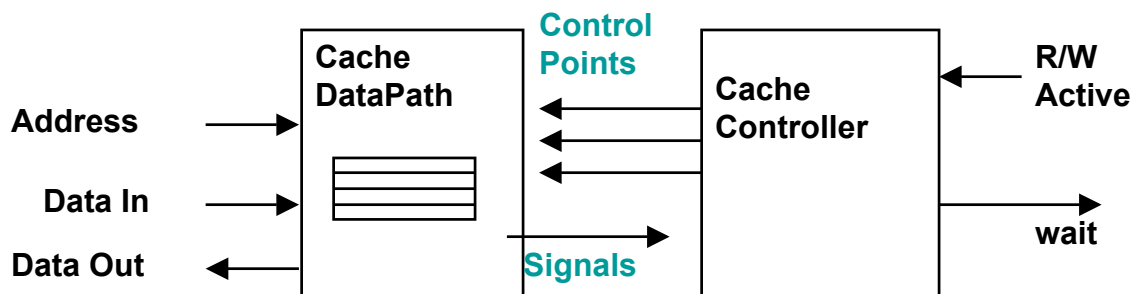


How Do you Design a Memory System?

- Set of Operations that must be supported
 - read: $\text{data} \leq \text{Mem}[\text{Physical Address}]$
 - write: $\text{Mem}[\text{Physical Address}] \leq \text{Data}$



- Determine the internal register transfers
- Design the Datapath
- Design the Cache Controller



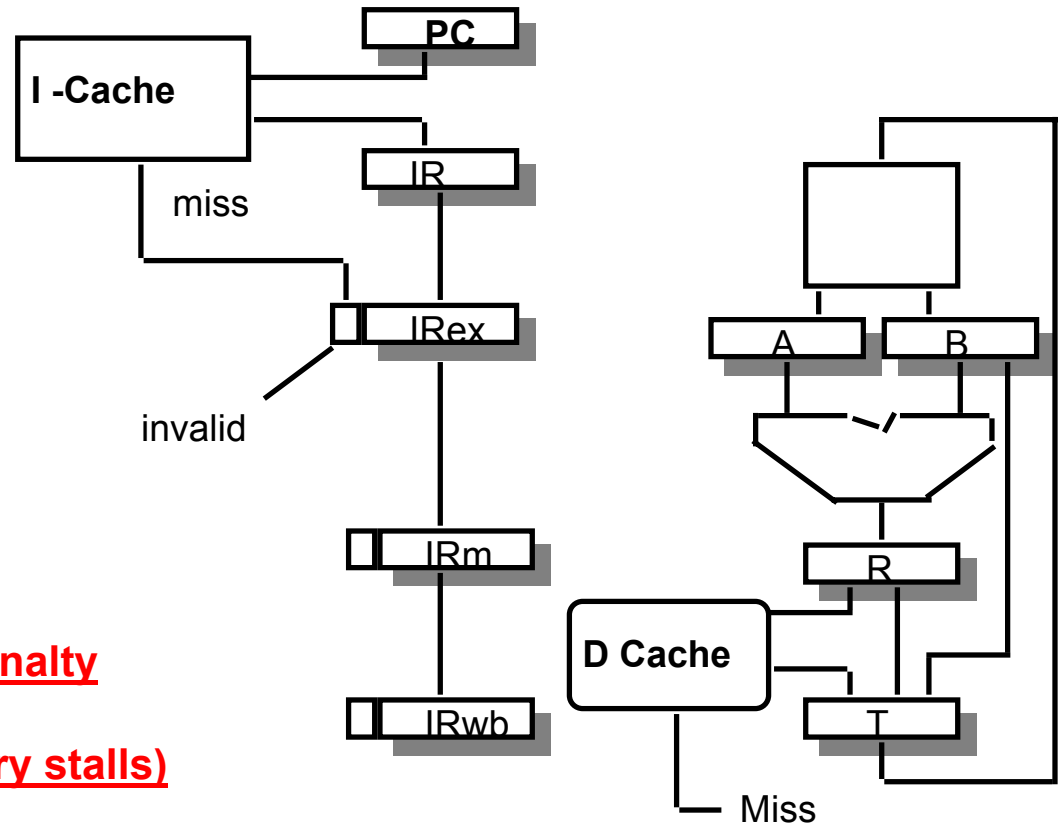
Impact on Cycle Time

Cache Hit Time:

directly tied to clock rate
increases with cache size
increases with associativity

Average Memory Access time =
Hit Time + Miss Rate x Miss Penalty

Time = IC x CT x (ideal CPI + memory stalls)



Improving Cache Performance: 3 general options

Time = IC x CT x (ideal CPI + memory stalls)

**Average Memory Access time =
Hit Time + (Miss Rate x Miss Penalty) =
(Hit Rate x Hit Time) + (Miss Rate x Miss Time)**

Options to reduce AMAT:

1. Reduce the miss rate,
2. Reduce the miss penalty, or
3. Reduce the time to hit in the cache.

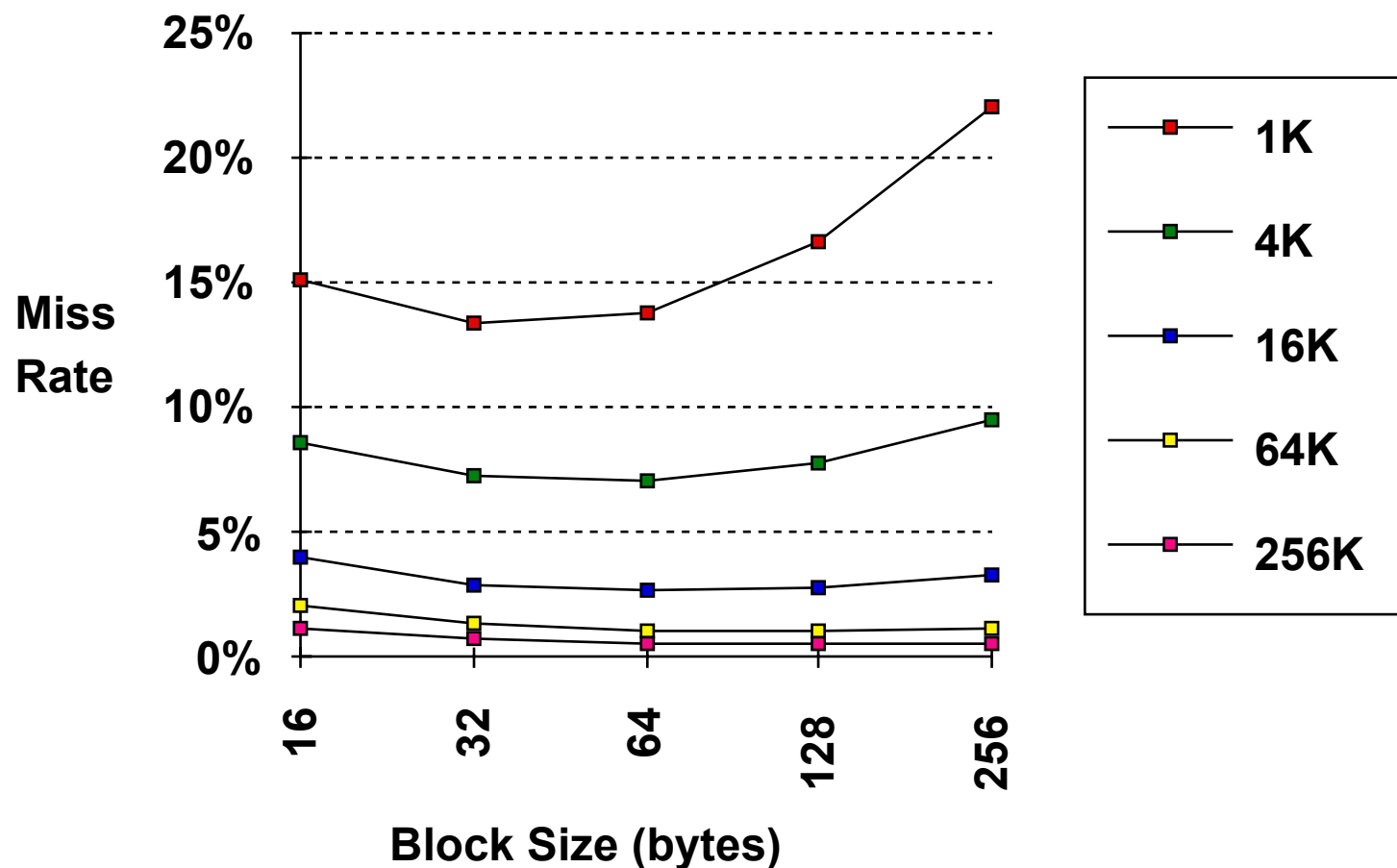


Improving Cache Performance

1. Reduce the miss rate,
2. Reduce the miss penalty, or
3. Reduce the time to hit in the cache.



1. Reduce Misses via Larger Block Size



2. Reduce Misses via Higher Associativity

- 2:1 Cache Rule:
 - Miss Rate DM cache size N \sim Miss Rate 2-way cache size $N/2$
- Beware: Execution time is only final measure!
 - Will Clock Cycle time increase?
 - Hill [1988] suggested hit time for 2-way vs. 1-way external cache +10%, internal + 2%



Example: Avg. Memory Access Time vs. Miss Rate

- Assume CCT = 1.10 for 2-way, 1.12 for 4-way, 1.14 for 8-way vs. CCT direct mapped

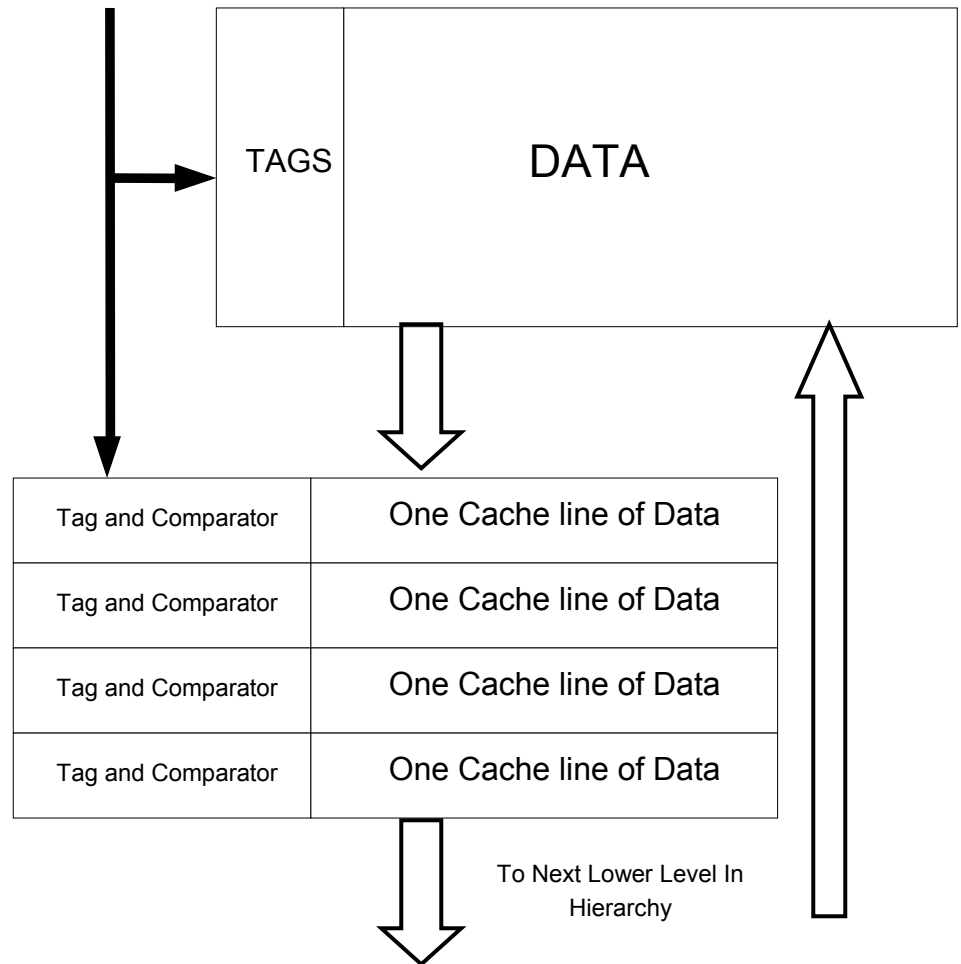
Cache Size (KB)	Associativity			
	1-way	2-way	4-way	8-way
1	2.33	2.15	2.07	2.01
2	1.98	1.86	1.76	1.68
4	1.72	1.67	1.61	1.53
8	1.46	1.48	1.47	1.43
16	1.29	1.32	1.32	1.32
32	1.20	1.24	1.25	1.27
64	1.14	1.20	1.21	1.23
128	1.10	1.17	1.18	1.20

(Red means A.M.A.T. not improved by more associativity)



3. Reducing Misses via a “Victim Cache”

- How to combine fast hit time of direct mapped yet still avoid conflict misses?
- Add buffer to place data discarded from cache
- Jouppi [1990]: 4-entry victim cache removed 20% to 95% of conflicts for a 4 KB direct mapped data cache
- Used in Alpha, HP machines



4. Reducing Misses by Hardware Prefetching

- E.g., Instruction Prefetching
 - Alpha 21064 fetches 2 blocks on a miss
 - Extra block placed in “stream buffer”
 - On miss check stream buffer
- Works with data blocks too:
 - Jouppi [1990] 1 data stream buffer got 25% misses from 4KB cache; 4 streams got 43%
 - Palacharla & Kessler [1994] for scientific programs for 8 streams got 50% to 70% of misses from 2 64KB, 4-way set associative caches
- Prefetching relies on having extra memory bandwidth that can be used without penalty
 - Could reduce performance if done indiscriminantly!!!



Improving Cache Performance (Continued)

1. Reduce the miss rate,
- 2. Reduce the miss penalty,* or
3. Reduce the time to hit in the cache.

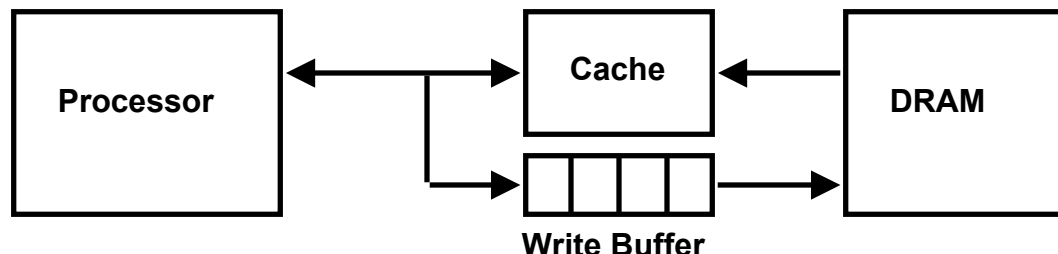


0. Reducing Penalty: Faster DRAM / Interface

- New DRAM Technologies
 - RAMBUS - same initial latency, but much higher bandwidth
 - Synchronous DRAM
 - Startup companies
 - Merged DRAM/Logic - IRAM project here at Berkeley
- Better BUS interfaces
- CRAY Technique: only use SRAM



1. Reducing Penalty: Read Priority over Write on Miss



- A Write Buffer is needed between the Cache and Memory
 - Processor: writes data into the cache and the write buffer
 - Memory controller: write contents of the buffer to memory
- Write buffer is just a FIFO:
 - Typical number of entries: 4
 - Works fine if: Store frequency (w.r.t. time) $\ll 1 / \text{DRAM write cycle}$
 - **Must handle burst behavior as well!**

2. Reduce Penalty: Early Restart and Critical Word First

- Don't wait for full block to be loaded before restarting CPU
 - *Early restart*—As soon as the requested word of the block arrives, send it to the CPU and let the CPU continue execution
 - *Critical Word First*—Request the missed word first from memory and send it to the CPU as soon as it arrives; let the CPU continue execution while filling the rest of the words in the block. Also called *wrapped fetch* and *requested word first*
 - *DRAM FOR LAB 5 can do this in burst mode! (Check out sequential timing)*
- Generally useful only in large blocks,
- Spatial locality a problem; tend to want next sequential word, so not clear if benefit by early restart



3. Reduce Penalty: Non-blocking Caches

- *Non-blocking cache* or *lockup-free cache* allow data cache to continue to supply cache hits during a miss
 - requires F/E bits on registers or out-of-order execution
 - requires multi-bank memories
- “*hit under miss*” reduces the effective miss penalty by working during miss vs. ignoring CPU requests
- “*hit under multiple miss*” or “*miss under miss*” may further lower the effective miss penalty by overlapping multiple misses
 - Significantly increases the complexity of the cache controller as there can be multiple outstanding memory accesses
 - Requires multiple memory banks (otherwise cannot support)
 - Pentium Pro allows 4 outstanding memory misses



What happens on a Cache miss?

- For in-order pipeline, 2 options:
 - Freeze pipeline in Mem stage (popular early on: Sparc, R4000)

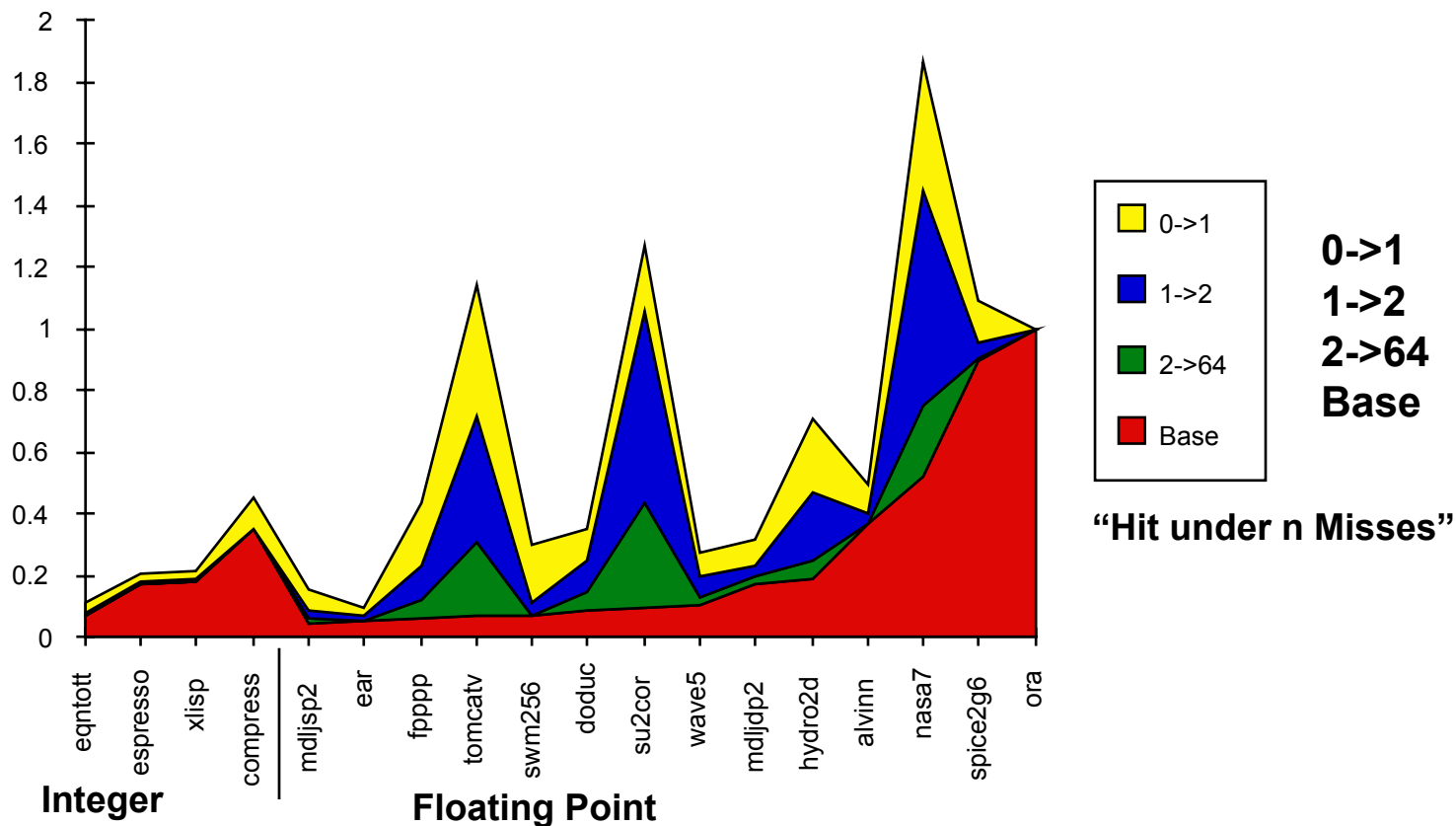
IF	ID	EX	Mem	stall	stall	stall	...	stall	Mem	Wr
	IF	ID	EX	stall	stall	stall	...	stall	stall	Ex Wr

- Use Full/Empty bits in registers + MSHR queue
 - MSHR = “Miss Status/Handler Registers” (Kroft)
Each entry in this queue keeps track of status of outstanding memory requests to one complete memory line.
 - Per cache-line: keep info about memory address.
 - For each word: register (if any) that is waiting for result.
 - Used to “merge” multiple requests to one memory line
 - New load creates MSHR entry and sets destination register to “Empty”.
Load is “released” from pipeline.
 - Attempt to use register before result returns causes instruction to block in decode stage.
 - Limited “out-of-order” execution with respect to loads.

Popular with in-order superscalar architectures.

- Out-of-order pipelines already have this functionality built in... (load queues, etc)

Value of Hit Under Miss for SPEC



- FP programs on average: AMAT= 0.68 -> 0.52 -> 0.34 -> 0.26
- Int programs on average: AMAT= 0.24 -> 0.20 -> 0.19 -> 0.19
- 8 KB Data Cache, Direct Mapped, 32B block, 16 cycle miss



Summary (1/3)

- Two Different Types of Locality:
 - Temporal Locality (Locality in Time): If an item is referenced, it will tend to be referenced again soon.
 - Spatial Locality (Locality in Space): If an item is referenced, items whose addresses are close by tend to be referenced soon.
- SRAM is fast but expensive and not very dense:
 - 6-Transistor cell (no static current) or 4-Transistor cell (static current)
 - Does not need to be refreshed
 - Good choice for providing the user FAST access time.
 - Typically used for CACHE
- DRAM is slow but cheap and dense:
 - 1-Transistor cell (+ trench capacitor)
 - Must be refreshed
 - Good choice for presenting the user with a BIG memory system
 - Both asynchronous and synchronous versions
 - Limited signal requires “sense-amplifiers” to recover



Summary 2/ 3:

- The Principle of Locality:
 - Program likely to access a relatively small portion of the address space at any instant of time.
 - **Temporal Locality**: Locality in Time
 - **Spatial Locality**: Locality in Space
- Three (+1) Major Categories of Cache Misses:
 - **Compulsory Misses**: sad facts of life. Example: cold start misses.
 - **Conflict Misses**: increase cache size and/or associativity.
Nightmare Scenario: ping pong effect!
 - **Capacity Misses**: increase cache size
 - **Coherence Misses**: Caused by external processors or I/O devices
- Cache Design Space
 - total size, block size, associativity
 - replacement policy
 - write-hit policy (write-through, write-back)
 - write-miss policy



Summary 3 / 4: 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**

