Memory Hierarchies

Instructor: Dmitri A. Gusev

Fall 2007

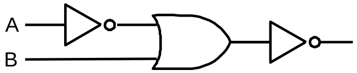
CS 502: Computers and Communications Technology

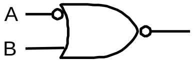
Lecture 10, October 8, 2007

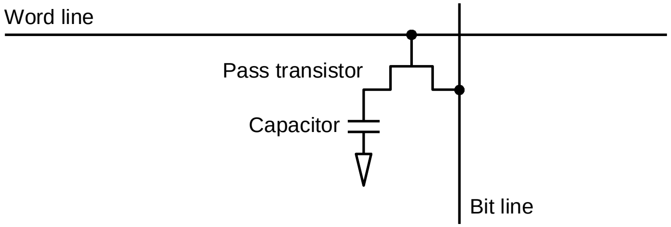
Memories

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

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







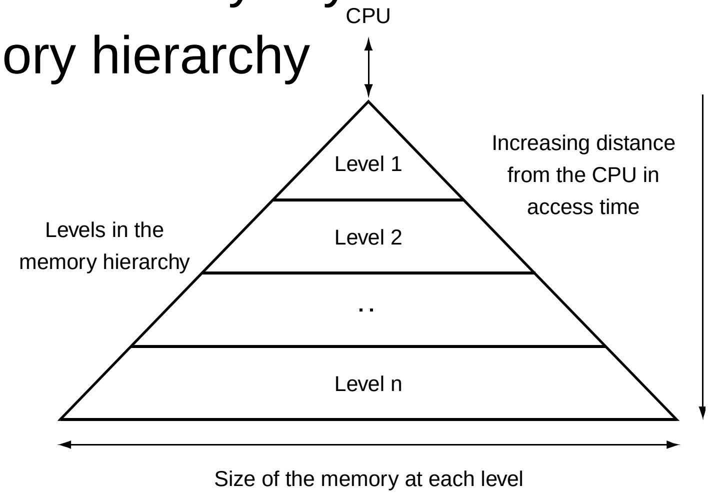
Exploiting Memory Hierarchy

- Users want large and fast memories!

2004

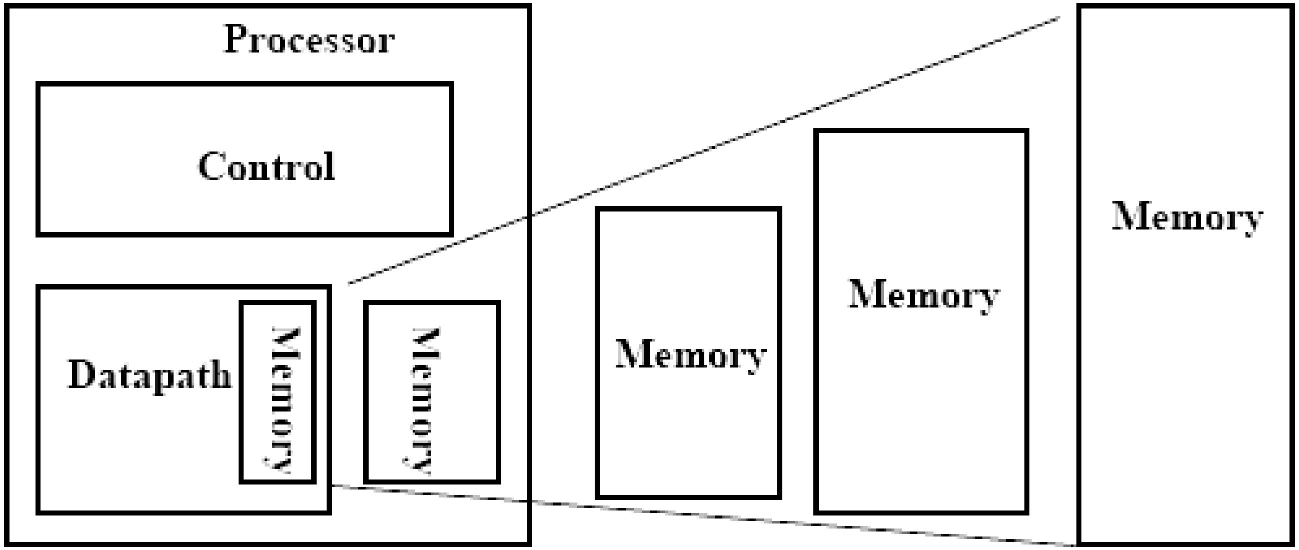
SRAM access times are .5 – 5ns at cost of $4000 to $10,000 per GB.
DRAM access times are 50-70ns at cost of $100 to $200 per GB.
Disk access times are 5 to 20 million ns at cost of $.50 to $2 per GB.

- Try and give it to them anyway
  - build a memory hierarchy


CPU
Increasing distance from the CPU in access time
Level 1
Level 2
Levels in the memory hierarchy
. .
Level n
Size of the memory at each level

An Expanded View of the Memory System

•



Speed:	Fastest	Slowest
Size:	Smallest	Biggest
Cost:	Highest	Lowest

Locality

- A principle that makes having a 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.

Why does code have locality?

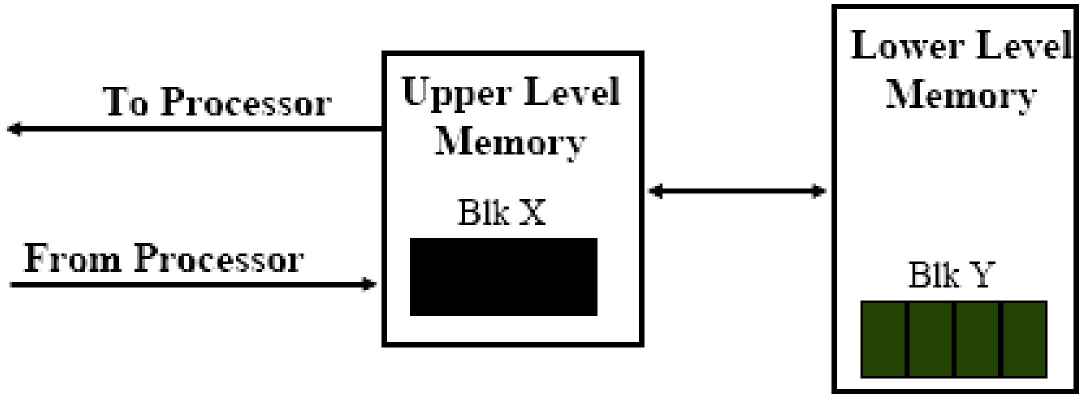
- Our initial focus: two levels (upper, lower)
  - block: minimum unit of data
  - hit: data requested is in the upper level
  - miss: data requested is not in the upper level

Memory Hierarchy: How Does it Work?

- 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

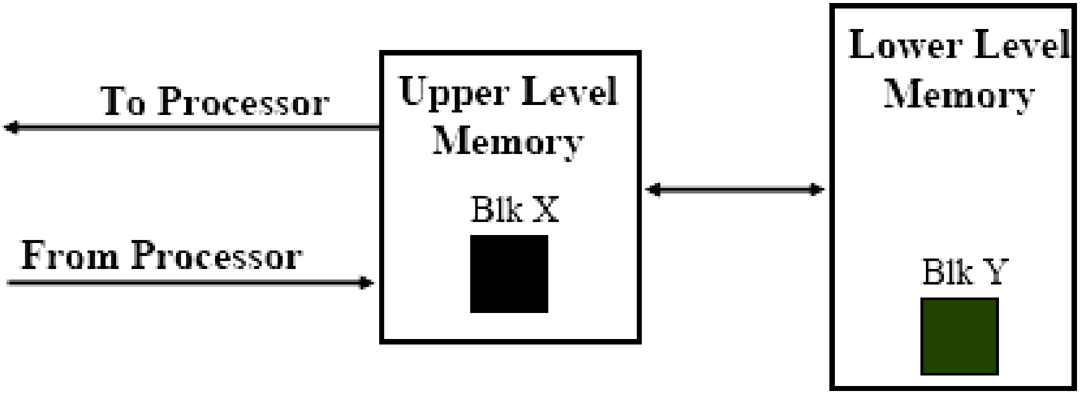


Memory Hierarchy: Terminology

- 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 - (Hit Rate)
  - Miss Penalty: Time to replace a block in the upper level +

    Time to deliver the block the processor
- Hit Time << Miss Penalty



How is the hierarchy managed?

- Registers <-> Memory
  - by compiler (programmer?)
- cache <-> memory
  - by the hardware
- memory <-> disks
  - by the hardware and operating system (virtual memory)
  - by the programmer (files)

Cache

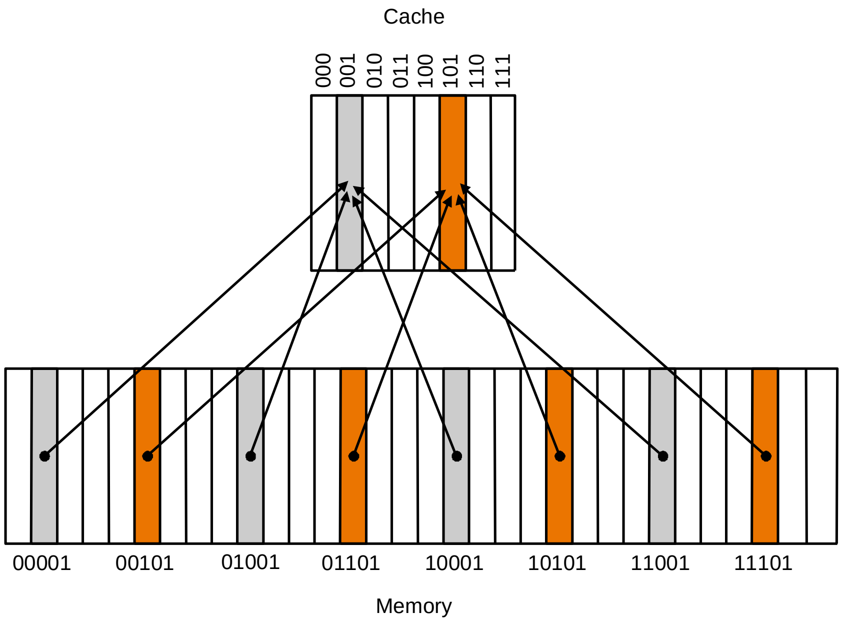
- Two issues:
  - How do we know if a data item is in the cache?
  - If it is, how do we find it?
- Our first example:
  - block size is one word of data
  - "direct mapped"

**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

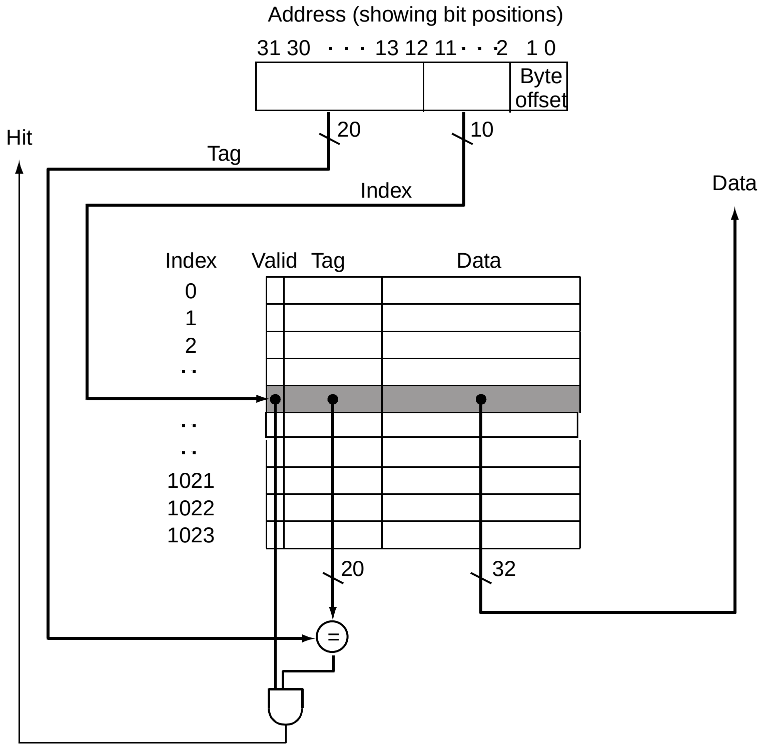
Direct Mapped Cache

- Mapping: address is modulo the number of blocks in the cache



Direct Mapped Cache

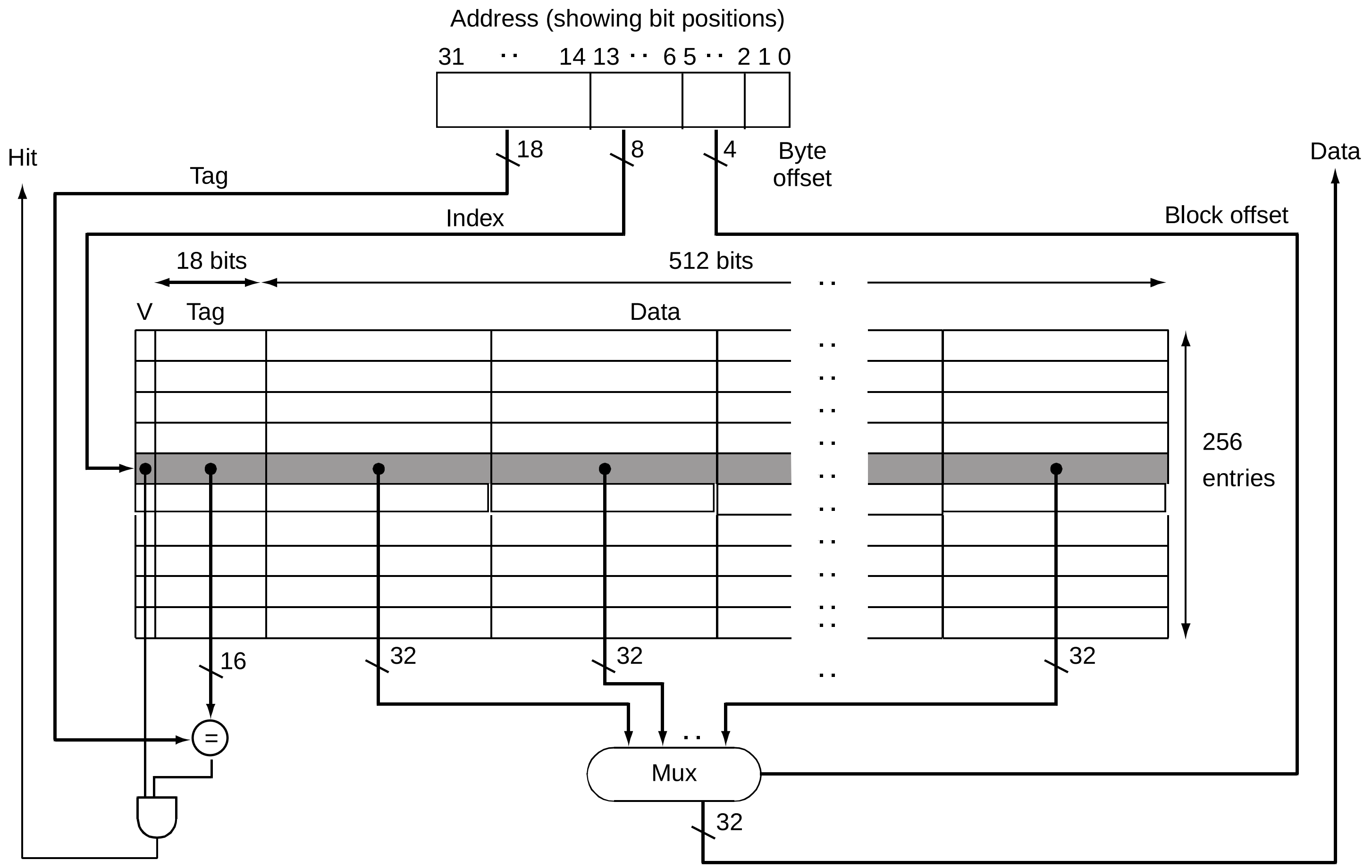
- For MIPS:



What kind of locality are we taking advantage of?

Direct Mapped Cache

- Taking advantage of spatial locality:

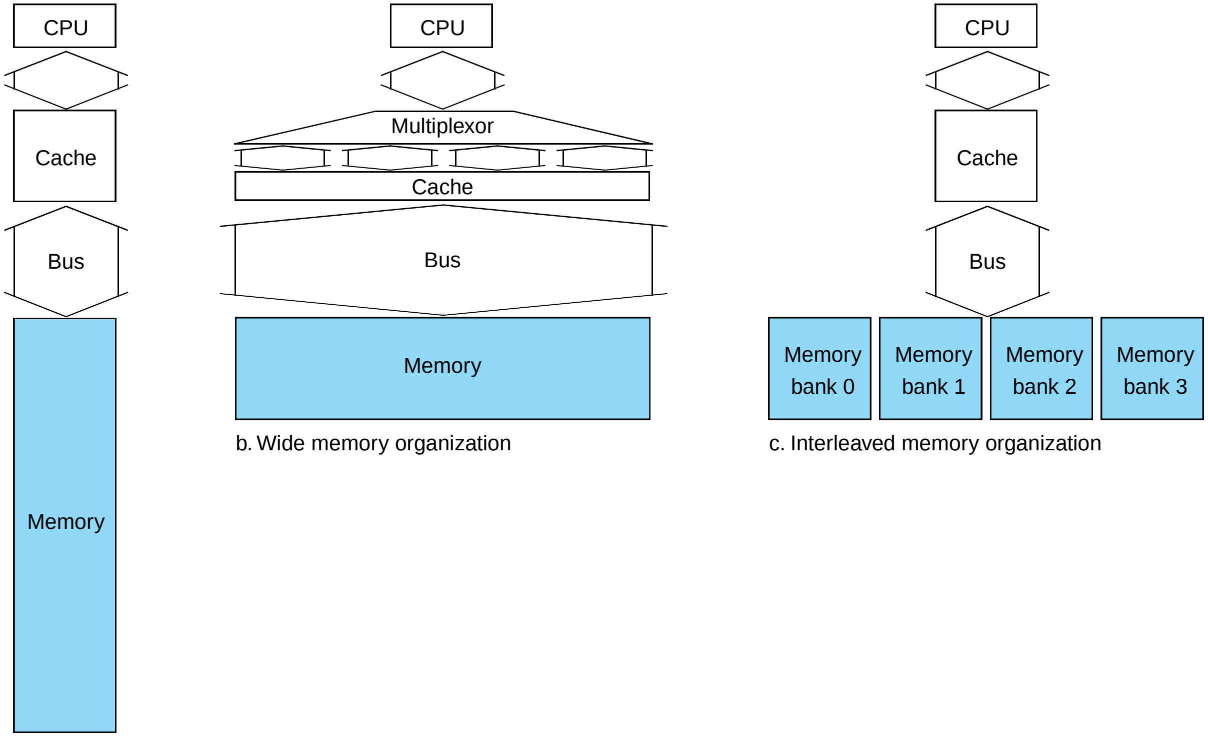


Hits vs. Misses

- 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

Hardware Issues

- Make reading multiple words easier by using banks of memory



a. One-word-wide memory organization

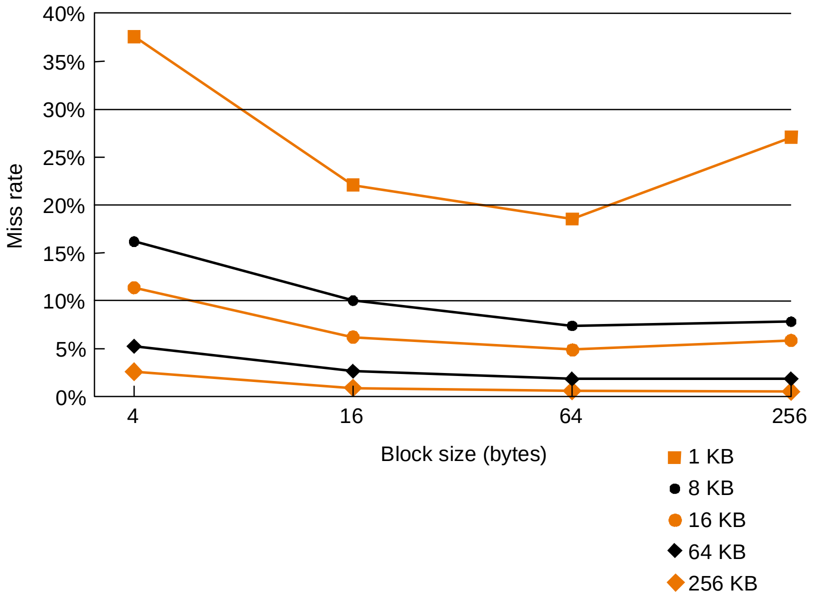
b. Wide memory organization

c. Interleaved memory organization

- It can get a lot more complicated...

Performance

- Increasing the block size tends to decrease miss rate:



- 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%

Performance

- Simplified model:

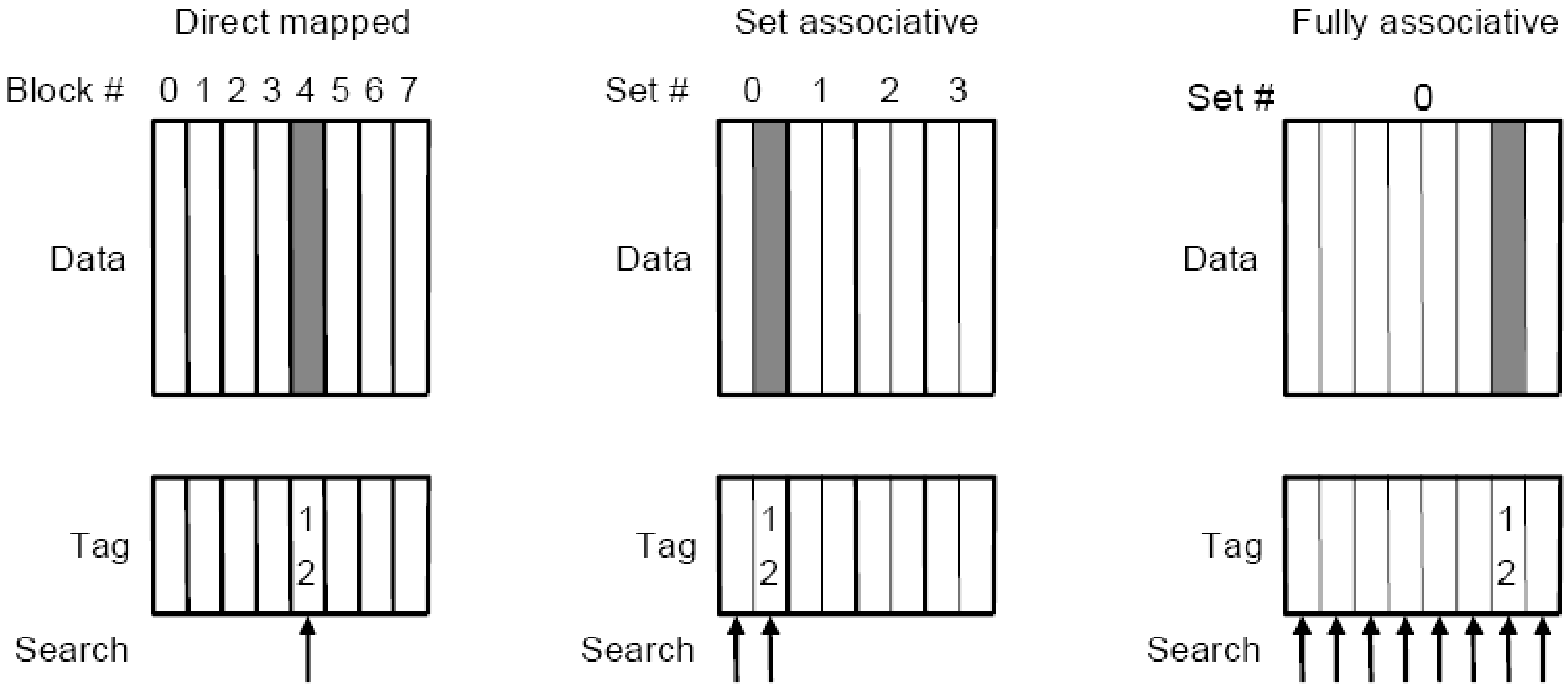
  execution time = (execution cycles + stall cycles) × cycle time

  stall cycles = # of instructions × miss ratio × miss penalty

- Two ways of improving performance:
  - decreasing the miss ratio
  - decreasing the miss penalty

What happens if we increase block size?

Improving cache performance

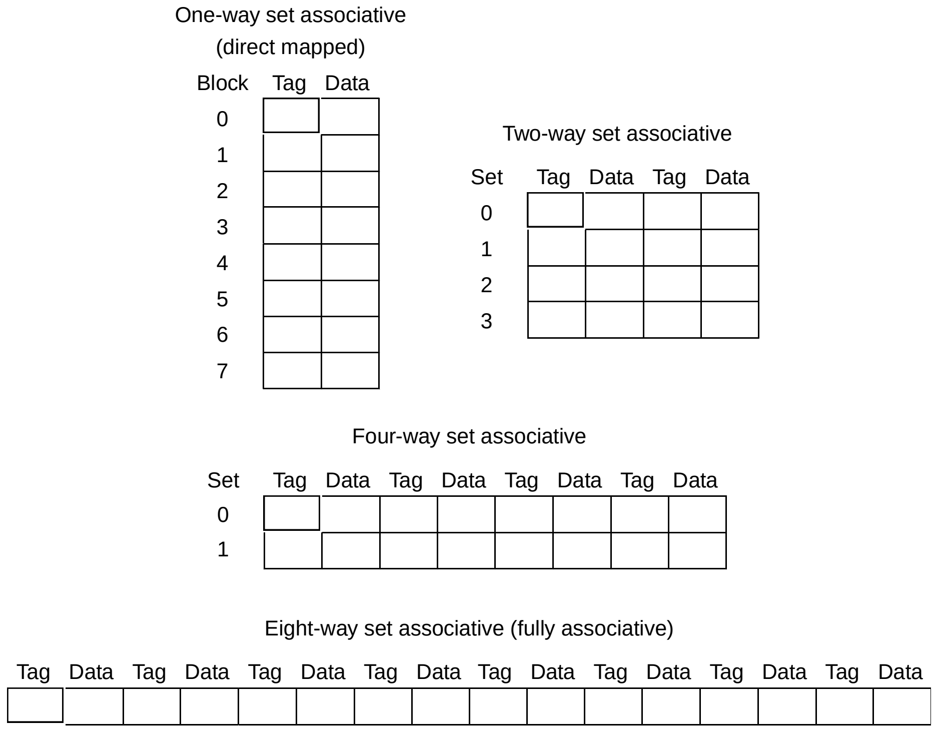


12 mod 8 = 4

12 mod 4 = 0

12 mod 1 = 0

Decreasing miss ratio with associativity



Compared to direct mapped, give a series of references that:

- *results in a lower miss ratio using a 2-way set associative cache*
- *results in a higher miss ratio using a 2-way set associative cache*

assuming we use the "least recently used" replacement strategy

*Cache size (blocks) = Number of sets * Associativity*

Tag size increases as the associativity increases

Example of associativity

Direct mapped cache

Block address	Cache index (block)
0	(0 mod 4) = 0
6	(6 mod 4) = 2
8	(8 mod 4) = 0

Ref.#	Block address	Hit or Miss	Cache index 0	Cache index 1	Cache index 2	Cache index 3
1	0	Miss	Memory[0]			
2	8	Miss	Memory[8]			
3	0	Miss	Memory[0]			
4	6	Miss	Memory[0]		Memory[6]	
5	8	Miss	Memory[8]		Memory[6]	

2-way associative

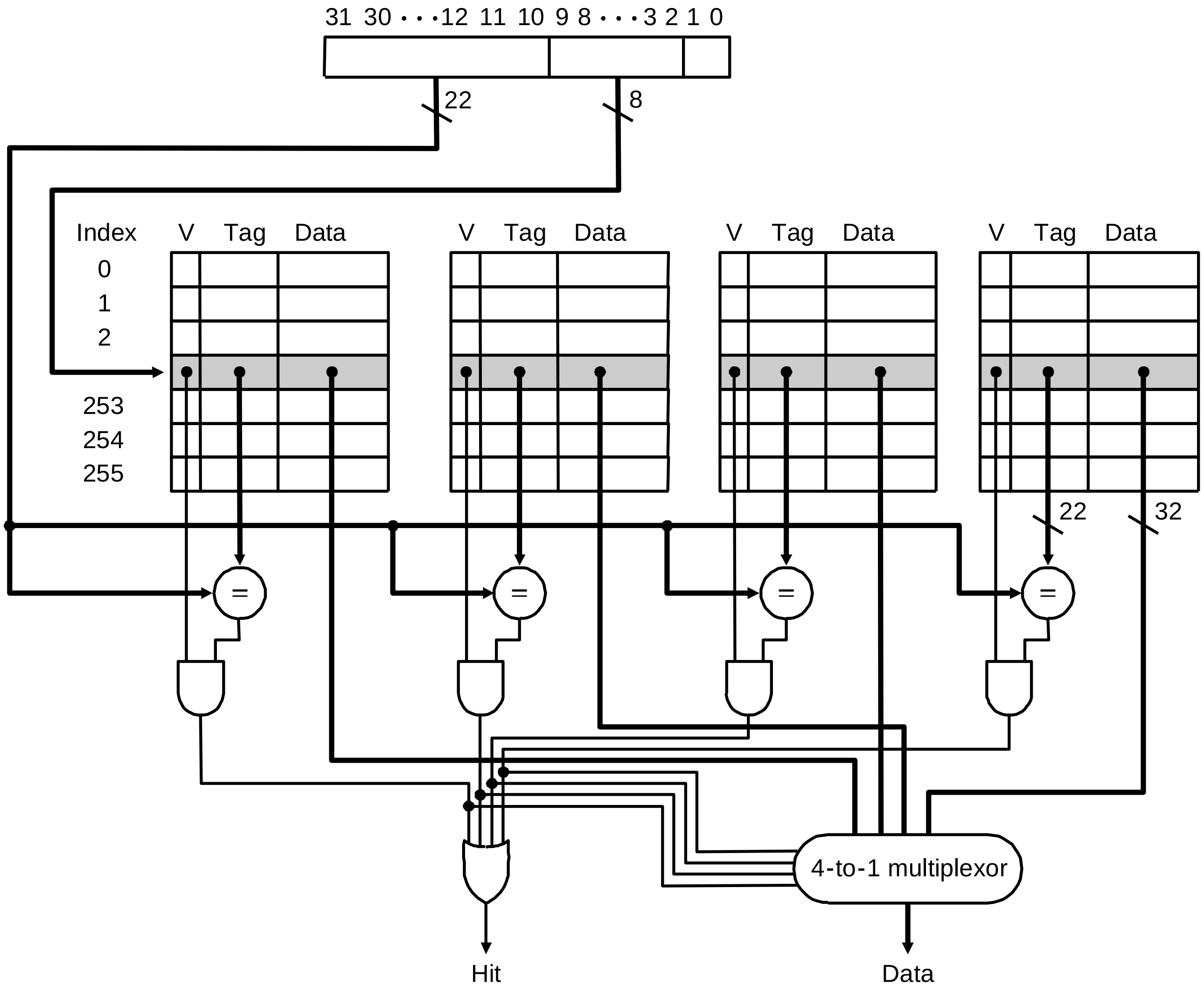
Block address	Cache index (set)
0	(0 mod 2) = 0
6	(6 mod 2) = 0
8	(8 mod 2) = 0

Ref.#	Block address	Hit or Miss	Cache set 0	Cache set 0	Cache set 1	Cache set 1
1	0	Miss	Memory[0]			
2	8	Miss	Memory[0]	Memory[8]		
3	0	Hit	Memory[0]	Memory[8]		
4	6	Miss	Memory[0]	Memory[6]		
5	8	Miss	Memory[8]	Memory[6]		

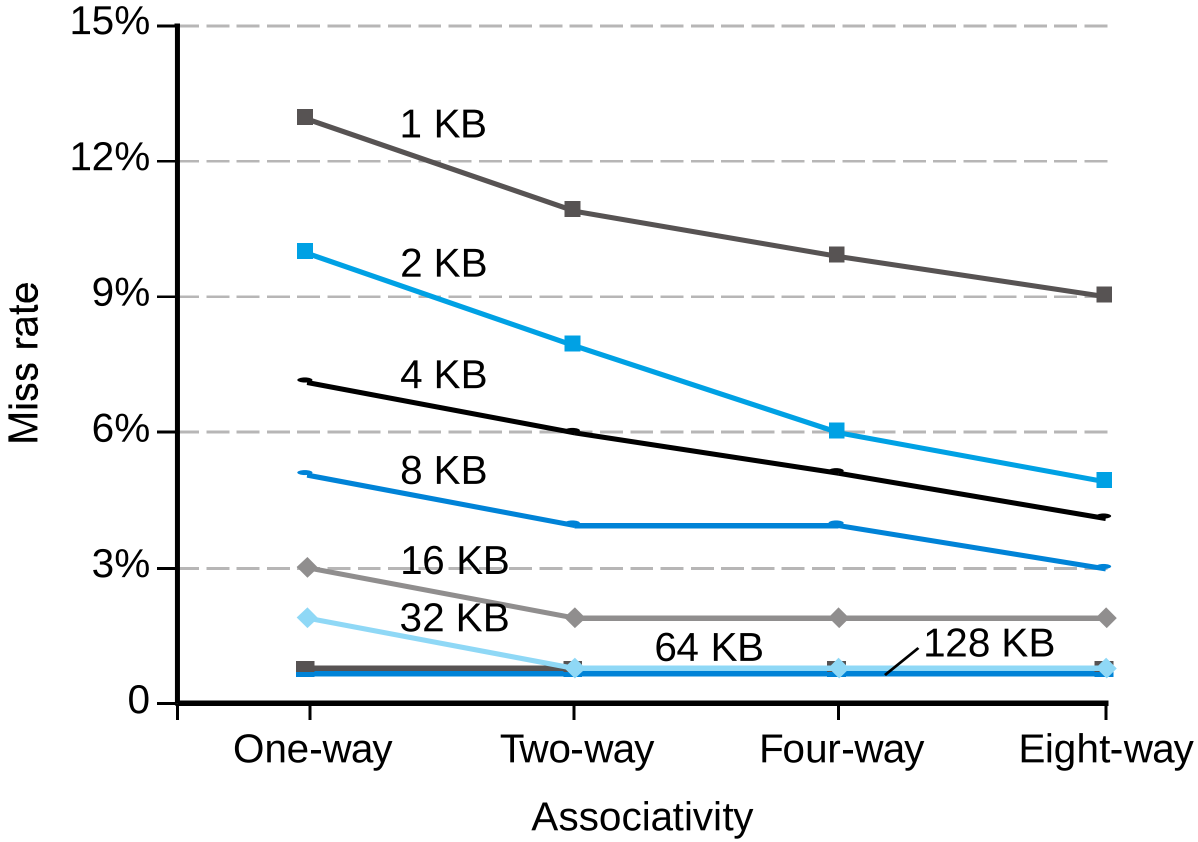
Fully associative

Ref.#	Block address	Hit or Miss	Cache block 0	Cache block 1	Cache block 2	Cache block 3
1	0	Miss	Memory[0]			
2	8	Miss	Memory[0]	Memory[8]		
3	0	Hit	Memory[0]	Memory[8]		
4	6	Miss	Memory[0]	Memory[8]	Memory[6]	
5	8	Hit	Memory[0]	Memory[8]	Memory[6]	

An implementation



Performance



Decreasing miss penalty with multilevel caches

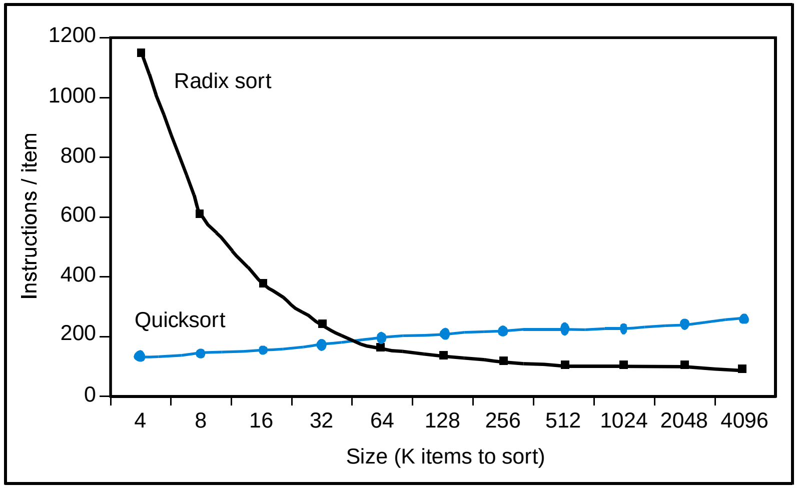
- Add a second level cache:
  - often primary cache is on the same chip as the processor
  - use SRAMs to add another cache above primary memory (DRAM)
  - miss penalty goes down if data is in 2nd level cache

- Example:
  - CPI of 1.0 on a 5 Ghz machine with a 5% miss rate, 100ns DRAM access
  - Adding 2nd level cache with 5ns access time decreases miss rate to .5%

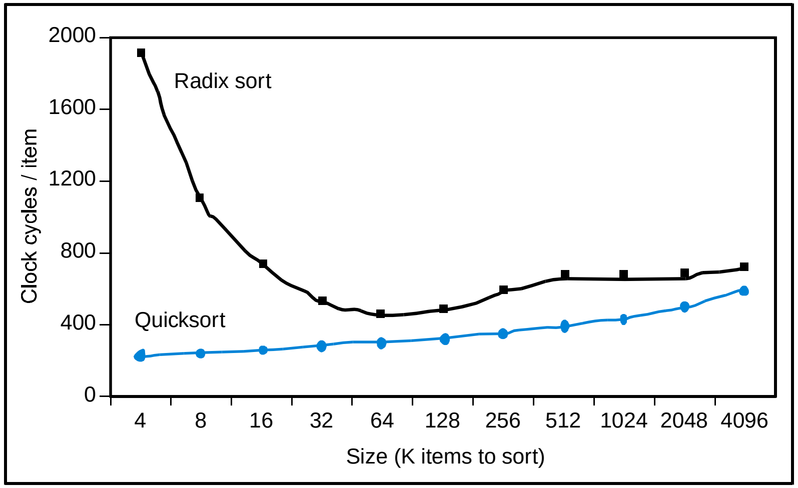
- Using multilevel caches:
  - try and optimize the hit time on the 1st level cache
  - try and optimize the miss rate on the 2nd level cache

Cache Complexities

- Not always easy to understand implications of caches:



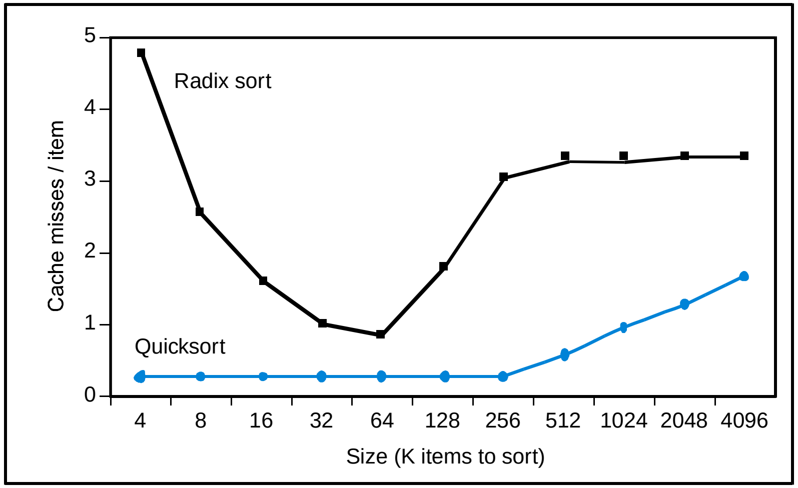
Theoretical behavior of Radix sort vs. Quicksort



Observed behavior of Radix sort vs. Quicksort

Cache Complexities

- Here is why:



- Memory system performance is often critical factor
  - multilevel caches, pipelined processors, make it harder to predict outcomes
  - Compiler optimizations to increase locality sometimes hurt ILP

- Difficult to predict best algorithm: need experimental data