

---

## Slide Set #16: Exploiting Memory Hierarchy

1

### Memory, Cost, and Performance

---

- **Ideal World:** we want a memory that is
  - Fast,
  - Big, &
  - Cheap!
- **Real World:**  
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.
- **Solution?**

3

### ADMIN

---

- Chapter 7 Reading
  - 7.1-7.3

2

### Locality

---

- A principle that makes caching work
- If an item is referenced,
  1. it will tend to be referenced again soon  
*why?*
  2. nearby items will tend to be referenced soon.  
*why?*

4

## Caching Basics

- Definitions

1. Minimum unit of data: "block" or "cache line"

For now assume, block is 1 byte

2. Data requested is in the cache:
3. Data requested is not in the cache:

- Cache has a given number of blocks (N)

- Challenge: How to locate an item in the cache?

- Simplest way:

Cache index = (Data address) mod N

e.g., N = 10, Address = 1024, Index =

e.g., N = 16, Address = 33, Index =

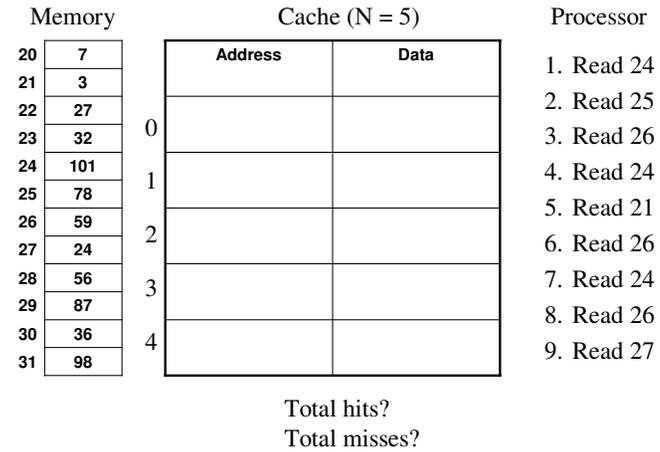
- Implications

For a given data address, there is \_\_\_\_\_ possible cache index

But for a given cache index there are \_\_\_\_\_ possible data items that could go there

5

## Example – (Simplified) Direct Mapped Cache



6

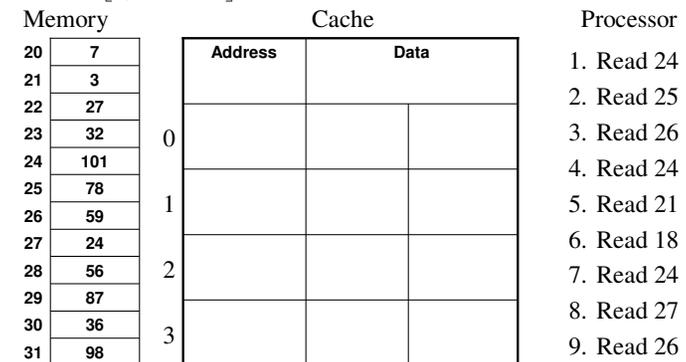
## Improving our basic cache

- Why did we miss? How can we fix it?

7

## Approach #1 – Increase Block Size

$$\text{Index} = \left\lfloor \frac{\text{ByteAddress}}{\text{BytesPerBlock}} \right\rfloor \bmod N$$



8

## Approach #2 – Add Associativity

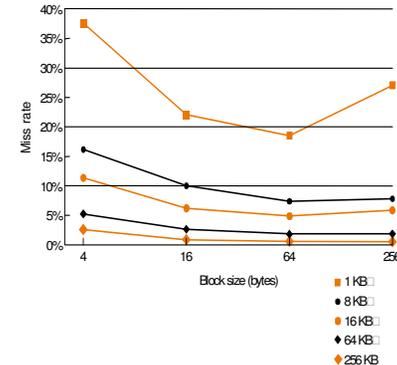
$$\text{Index} = \left\lfloor \frac{\text{ByteAddress}}{\text{BytesPerBlock}} \right\rfloor \bmod \frac{N}{\text{Associativity}}$$

Memory		Cache		Processor
		Address	Data	
20	7			1. Read 24
21	3			2. Read 25
22	27			3. Read 26
23	32			4. Read 24
24	101			5. Read 21
25	78			6. Read 18
26	59			7. Read 24
27	24			8. Read 27
28	56			9. Read 26
29	87			
30	36			
31	98			

9

## Performance Impact – Part 1

- To be fair, want to compare cache organizations with same data size
  - E.g., increasing block size must decrease number blocks (N)
- Overall, increasing block size tends to decrease miss rate:



10

## Performance Impact – Part 2

- Increasing block size...
  - May help by exploiting \_\_\_\_\_ locality
  - But, may hurt by increasing \_\_\_\_\_ (due to smaller \_\_\_\_\_)
  - Lesson – want block size > 1, but not too large
- Increasing associativity
  - Overall N stays the same, but smaller number of sets
  - May help by exploiting \_\_\_\_\_ locality (due to fewer \_\_\_\_\_)
  - May hurt because cache gets slower
  - Do we want associativity?

11

## How to handle a miss?

- Things we need to do:
  - \_\_\_\_\_ the CPU until miss completes
  - \_\_\_\_\_ old data from the cache  
Which data?
  - \_\_\_\_\_ the needed data from memory  
Pay the \_\_\_\_\_  
How long does this take?
  - \_\_\_\_\_ the CPU

What about a write miss?

12