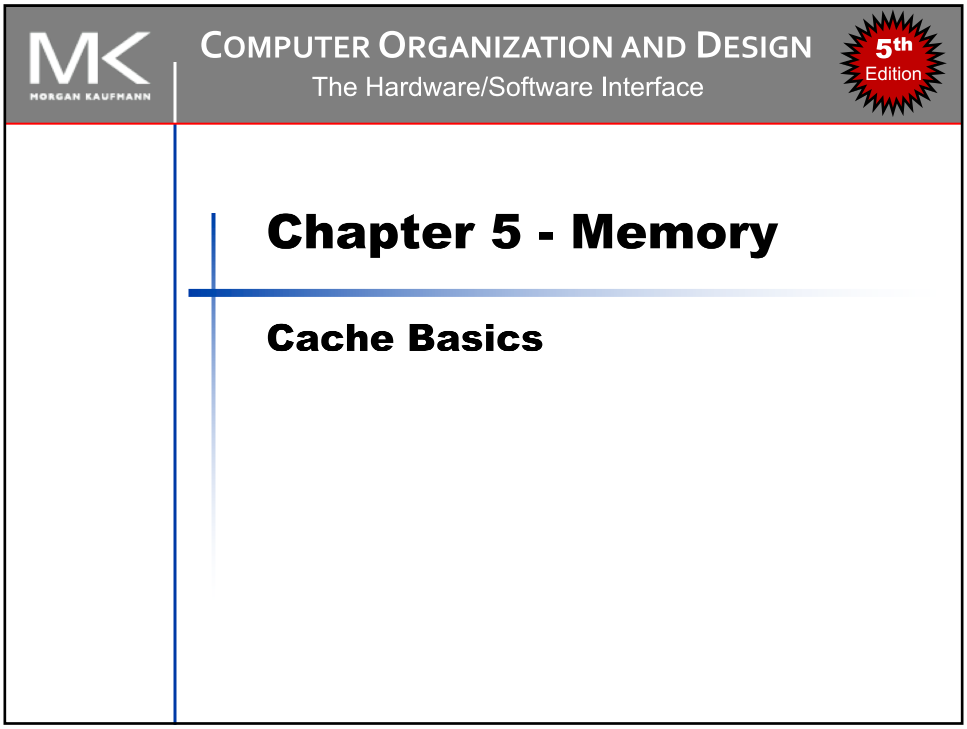


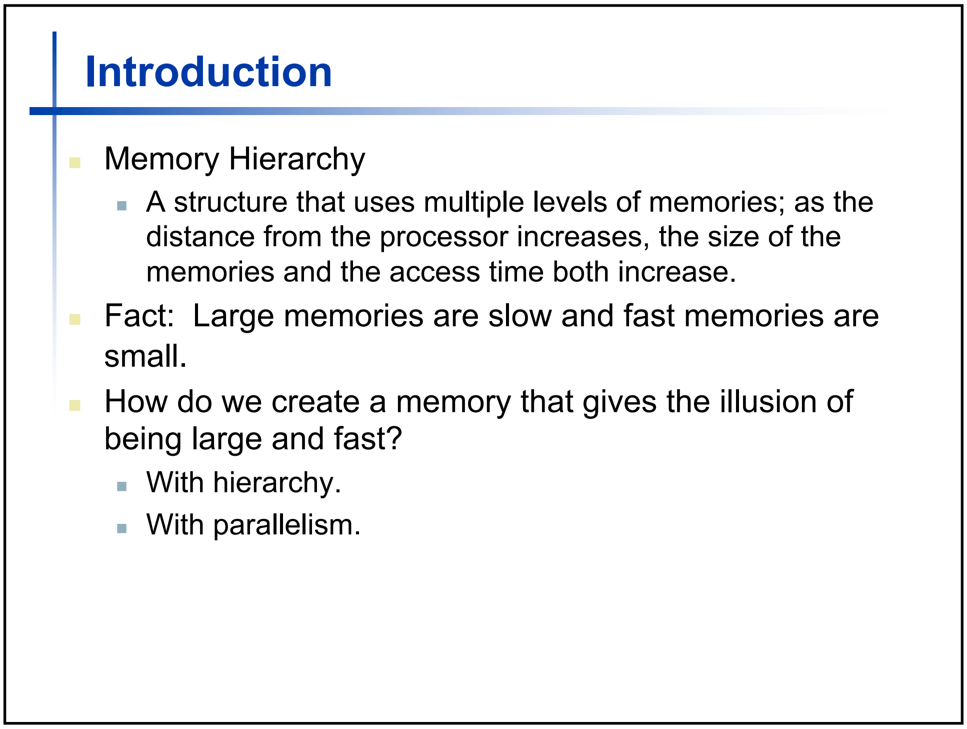
COMPUTER ORGANIZATION AND DESIGN

The Hardware/Software Interface

5th Edition

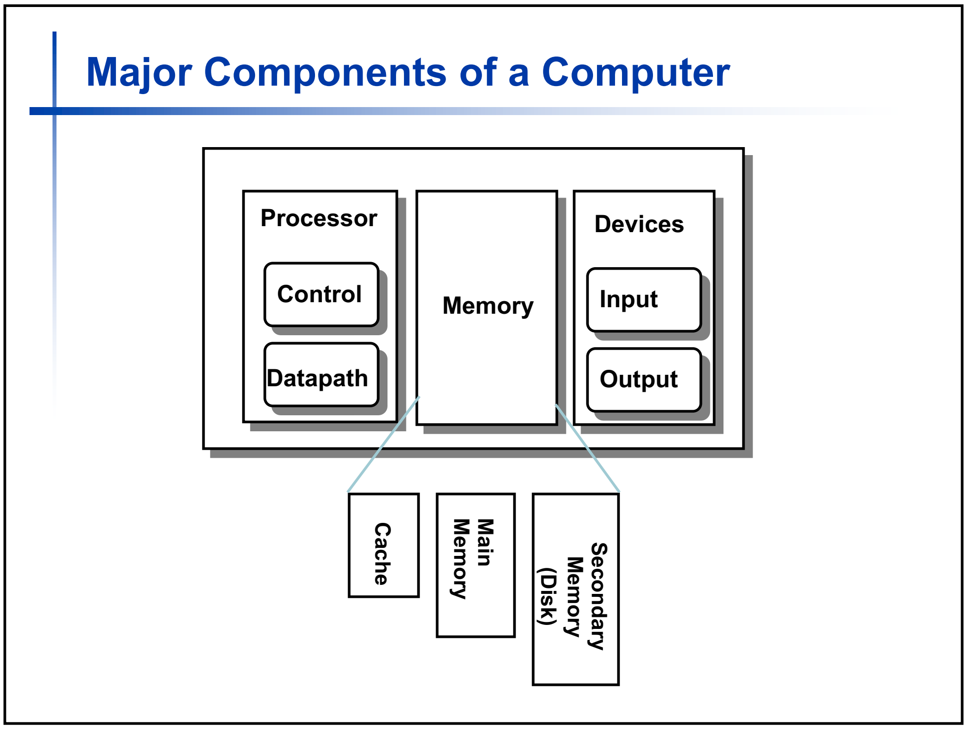
Chapter 5 - Memory

Cache Basics



Introduction

- Memory Hierarchy
  - A structure that uses multiple levels of memories; as the distance from the processor increases, the size of the memories and the access time both increase.
- Fact: Large memories are slow and fast memories are small.
- How do we create a memory that gives the illusion of being large and fast?
  - With hierarchy.
  - With parallelism.



Major Components of a Computer

Processor

Control

Datapath

Memory

Devices

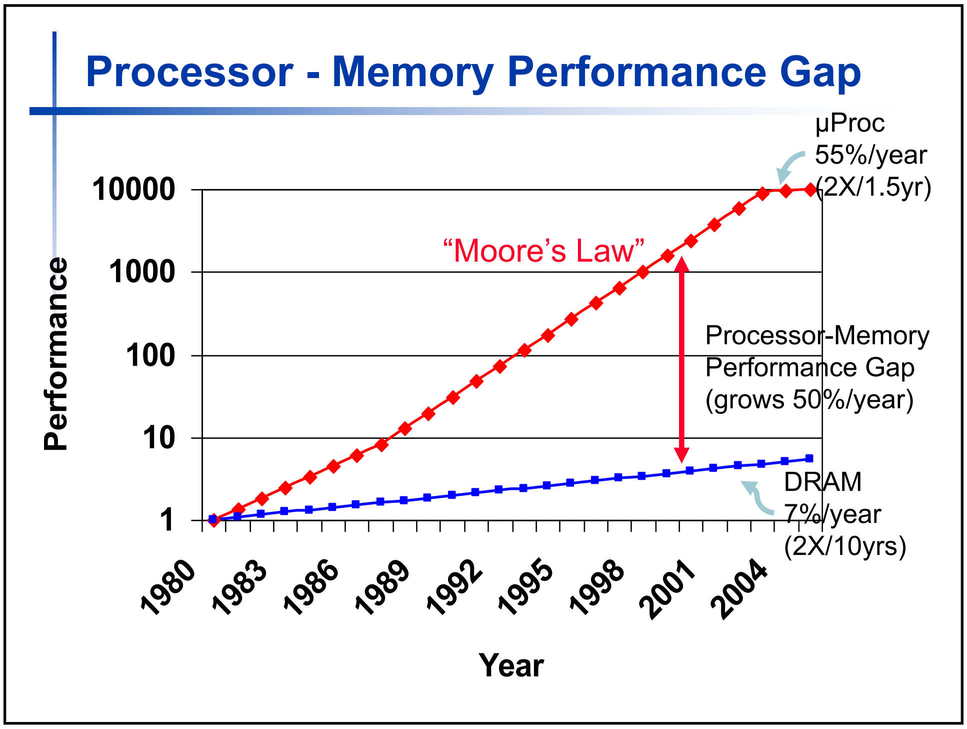
Input

Output

Cache

Main Memory

Secondary Memory (Disk)



Processor - Memory Performance Gap

µProc
55%/year
(2X/1.5yr)

“Moore’s Law”

Processor-Memory
Performance Gap
(grows 50%/year)

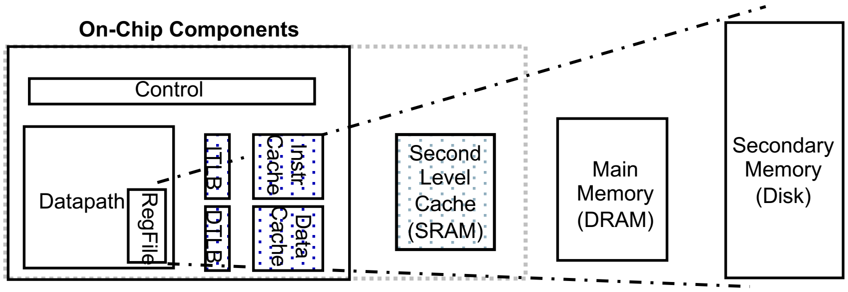
DRAM
7%/year
(2X/10yrs)

Cache Hierarchies

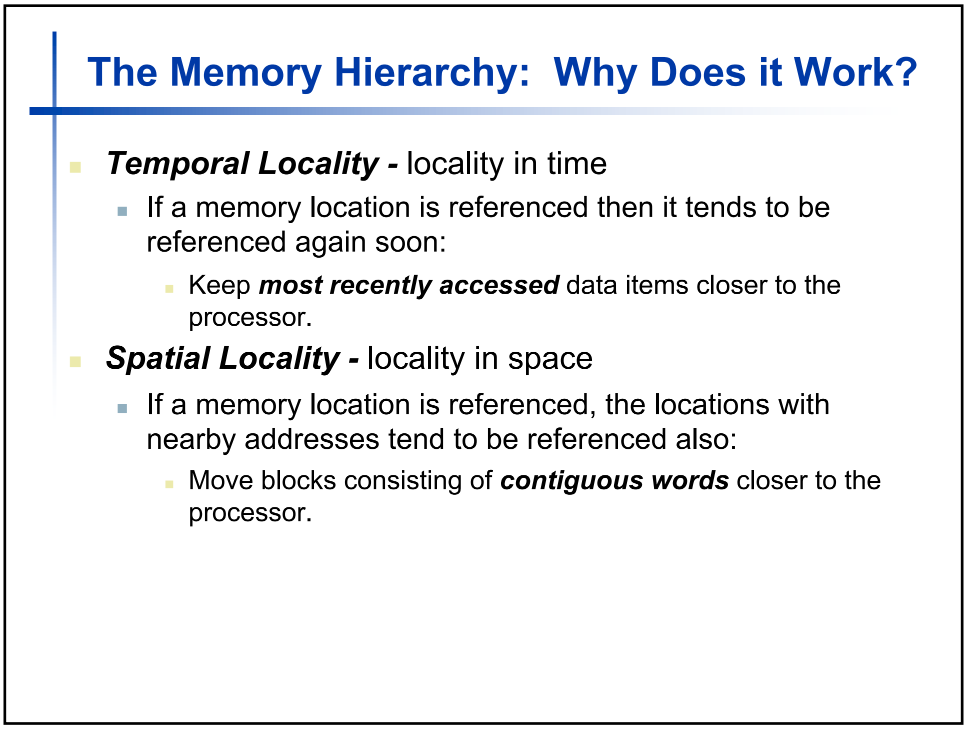
- Internet browsers cache web pages – Why?
- Data and instructions are stored in **Main Memory** composed of dynamic memory (**DRAM**). DRAM is a technology that has high bit density, but relatively poor latency – an access to data in memory can take as many as 300 cycles today.
- Hence, some data is stored on the processor in a structure called the cache – caches employ Static Ram (**SRAM)** technology, which is faster, but has lower bit density.

A Typical Memory Hierarchy

- Takes advantage of the ***principle of locality*** to present the user with as *large* a memory as possible in the *cheapest* technology at the *fastest* speed.

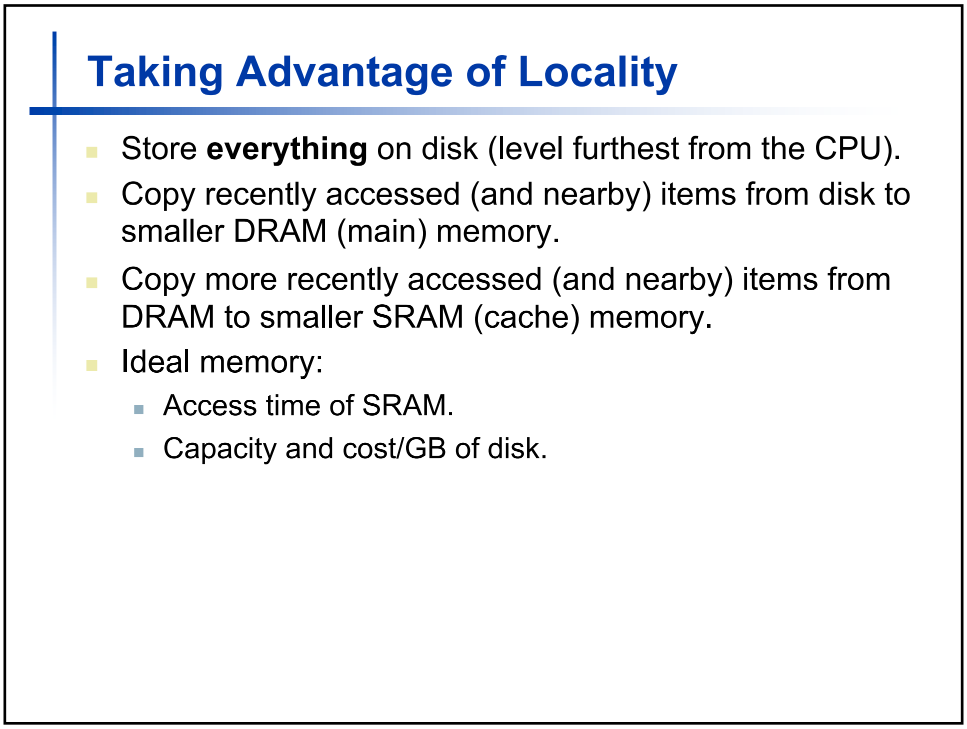


Speed (%cycles):	½'s	1's	10's	100's	10,000's
Size (bytes):	100's	10K's	M's	G's	T's
Cost:	highest				lowest



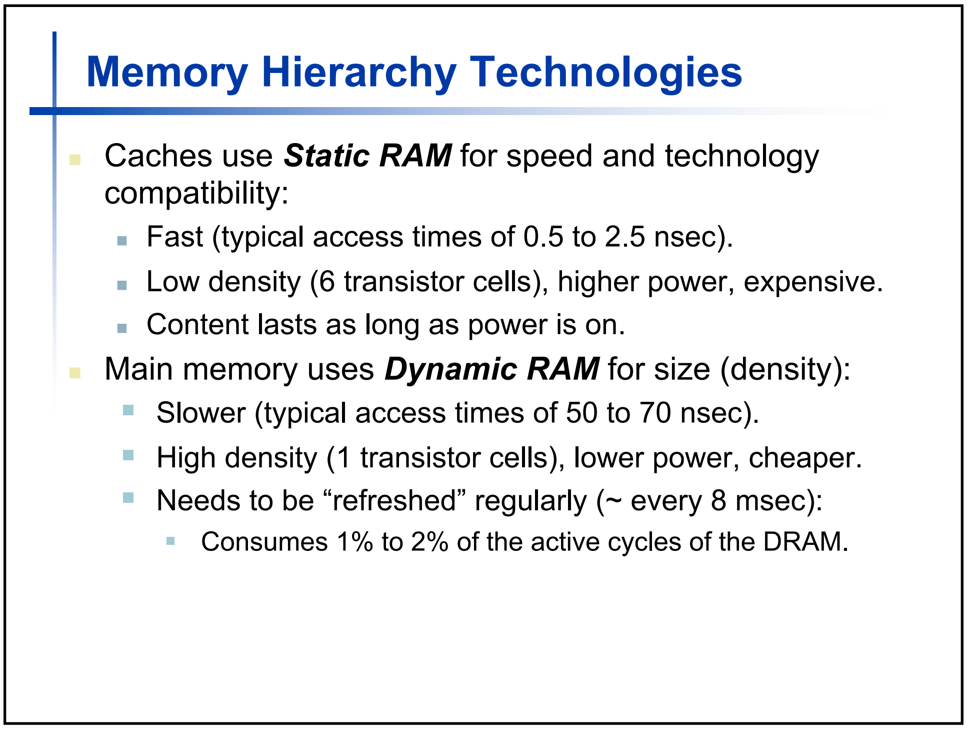
The Memory Hierarchy: Why Does it Work?

- ***Temporal Locality -*** locality in time
  - If a memory location is referenced then it tends to be referenced again soon:
    - Keep ***most recently accessed*** data items closer to the processor.
- ***Spatial Locality -*** locality in space
  - If a memory location is referenced, the locations with nearby addresses tend to be referenced also:
    - Move blocks consisting of ***contiguous words*** closer to the processor.



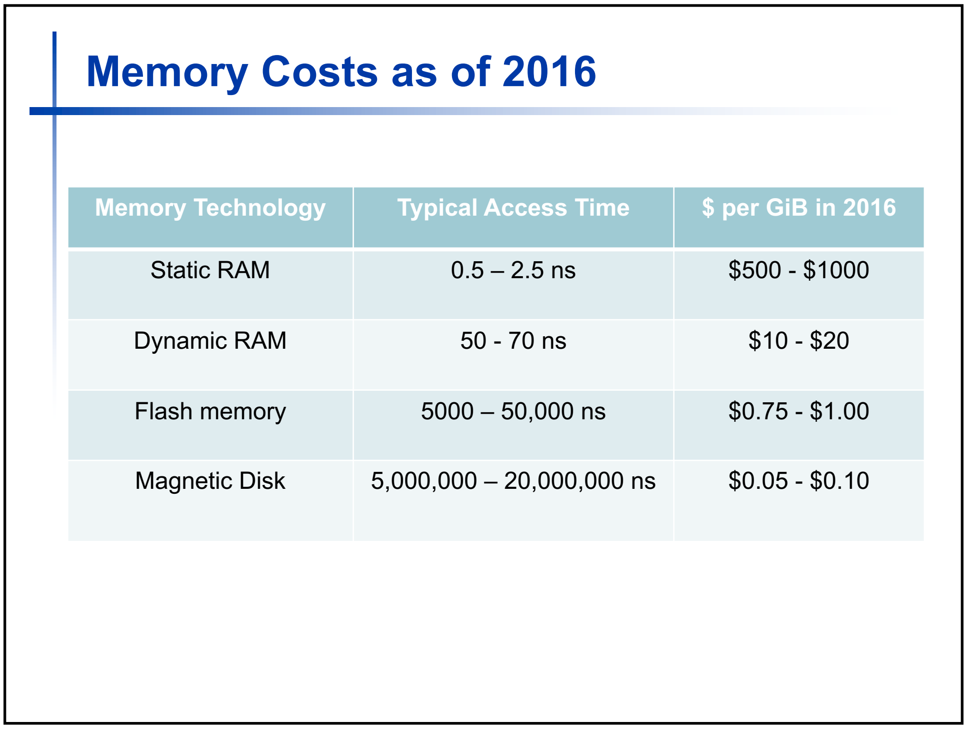
Taking Advantage of Locality

- Store **everything** on disk (level furthest from the CPU).
- Copy recently accessed (and nearby) items from disk to smaller DRAM (main) memory.
- Copy more recently accessed (and nearby) items from DRAM to smaller SRAM (cache) memory.
- Ideal memory:
  - Access time of SRAM.
  - Capacity and cost/GB of disk.



Memory Hierarchy Technologies

- Caches use ***Static RAM*** for speed and technology compatibility:
  - Fast (typical access times of 0.5 to 2.5 nsec).
  - Low density (6 transistor cells), higher power, expensive.
  - Content lasts as long as power is on.
- Main memory uses ***Dynamic RAM*** for size (density):
  - Slower (typical access times of 50 to 70 nsec).
  - High density (1 transistor cells), lower power, cheaper.
  - Needs to be "refreshed" regularly (~ every 8 msec):
    - Consumes 1% to 2% of the active cycles of the DRAM.



Memory Costs as of 2016

Memory Technology	Typical Access Time	$ per GiB in 2016
Static RAM	0.5 – 2.5 ns	$500 - $1000
Dynamic RAM	50 - 70 ns	$10 - $20
Flash memory	5000 – 50,000 ns	$0.75 - $1.00
Magnetic Disk	5,000,000 – 20,000,000 ns	$0.05 - $0.10

The Memory Hierarchy: Terminology

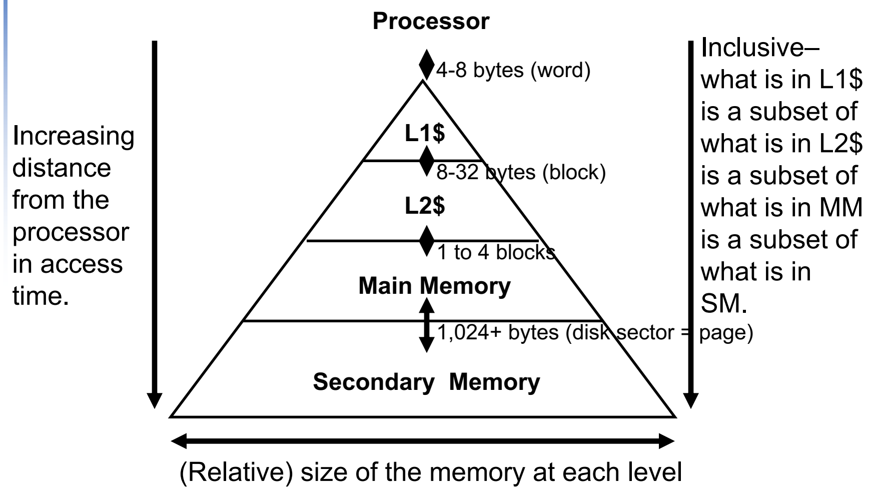
- ***Block*** or **Line** - the minimum unit of information that is accessed in a cache.
- ***Hit Rate*** - the fraction of memory accesses found in a given level of the memory hierarchy:
  - ***Hit Time*** – the time to access that level which consists of:

    Time to access the block + Time to determine hit/miss.
- ***Miss Rate*** - the fraction of memory accesses *not* found in a level of the memory hierarchy ⇒ 1 - (Hit Rate)
  - ***Miss Penalty*** – the time to replace a block in that level with the corresponding block from a lower level which consists of:

    Time to access the block in the lower level + Time to transmit that block to the level that experienced the miss + Time to insert the block in that level + Time to pass the block to the requester.

Hit Time << Miss Penalty

Characteristics of the Memory Hierarchy

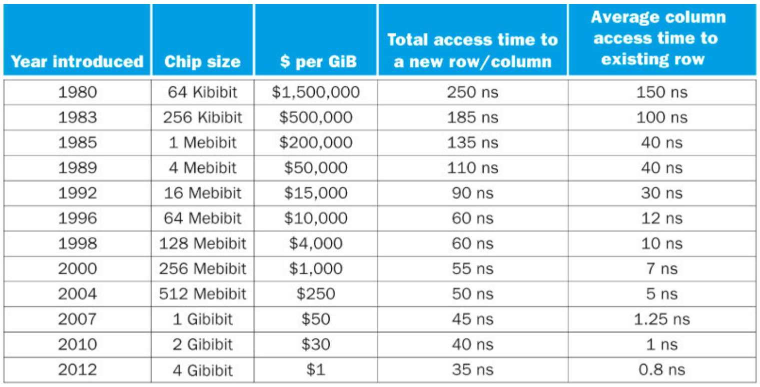


Increasing distance from the processor in access time.

Inclusive– what is in L1$ is a subset of what is in L2$ is a subset of what is in MM is a subset of what is in SM.

(Relative) size of the memory at each level

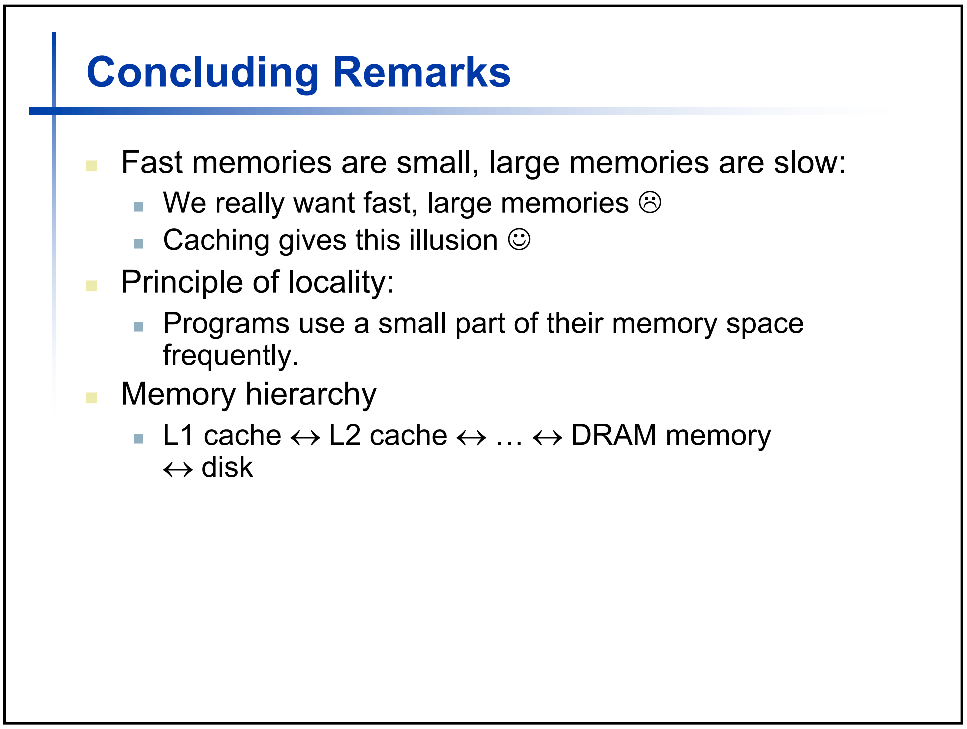
DRAM Generations



Year introduced	Chip size	$ per GiB	Total access time to a new row/column	Average column access time to existing row
1980	64 Kibibit	$1,500,000	250 ns	150 ns
1983	256 Kibibit	$500,000	185 ns	100 ns
1985	1 Mebibit	$200,000	135 ns	40 ns
1989	4 Mebibit	$50,000	110 ns	40 ns
1992	16 Mebibit	$15,000	90 ns	30 ns
1996	64 Mebibit	$10,000	60 ns	12 ns
1998	128 Mebibit	$4,000	60 ns	10 ns
2000	256 Mebibit	$1,000	55 ns	7 ns
2004	512 Mebibit	$250	50 ns	5 ns
2007	1 Gibibit	$50	45 ns	1.25 ns
2010	2 Gibibit	$30	40 ns	1 ns
2012	4 Gibibit	$1	35 ns	0.8 ns

Cache Basics

- Two questions to answer in hardware:
  - Q1: How do we know if a data item is in the cache?
  - Q2: If it is in the cache, how do we find it?
- Direct mapped method
  - Each memory block is mapped to exactly one block in the cache
    - Lots of lower level blocks must ***share*** blocks in the cache.
  - How do we know which particular block is stored in a cache location?
    - Store a portion of the block address as well as the data.
    - The higher-order address bits stored with the data are called the ***tag.***
  - What if there is no data in a location?
    - Valid bit: 1 = present, 0 = not present
    - Initially set to 0.



Concluding Remarks

- Fast memories are small, large memories are slow:
  - We really want fast, large memories ☹
  - Caching gives this illusion ☺
- Principle of locality:
  - Programs use a small part of their memory space frequently.
- Memory hierarchy
  - L1 cache ↔ L2 cache ↔ … ↔ DRAM memory ↔ disk