Memory Hierarchy

BITS Pilani
Pilani Campus

K Hari Babu
Department of Computer Science & Information Systems

innovate achieve lead

BITS Pilani

Pilani Campus

Memory Hierarchy

Memory Hierarchies

- Some fundamental and enduring properties of hardware and software:
  - Fast storage technologies cost more per byte, have less capacity, and require more power (heat!).
  - The gap between CPU and main memory speed is widening.
  - Well-written programs tend to exhibit good locality.
- These fundamental properties complement each other beautifully.
- They suggest an approach for organizing memory and storage systems known as a memory hierarchy.

Caching: Multi-Level Inclusion Principle

- Multi-Level Inclusion Principle
  - All the data in level h is included in level h+1
- Reasons?
  - Level h+1 is typically more persistent than level h.
  - Level h+1 is order(s) of magnitude larger.
  - When level h data has to be replaced (Why?),
    - only written data needs to be copied.
    - Why is this good savings?
    - typically 15% of all memory activity is made of “write”s i.e. about 85% of data can be discarded

From Prof. Shan’s slides

Memory Hierarchy – Access Time

BITS Pilani

- Performance Issue:
  - What is the amortized access time?
- For a 2-level memory hierarchy
  - Effective Access time at level h

$T_{eff} = (1-P_{miss}) * T_h + P_{miss} * T_{h+1}$

$T_{eff} = T_h + P_{miss} (T_{h+1} - T_h)$

  - where P_{miss} is "probability of not finding the data in level h"

From Prof. Shan's slides

Memory Hierarchy – Access Time

- Memory efficiency (M.E)
  - Highest if we can access the hierarchy as fast as the fastest level
    - i.e. at level h in a hierarchy with levels h, h+1, ...
- Define $M.E = 100 * (T_h / T_{eff})$

From Prof. Shan's slides

Memory Hierarchy – Access Time

- Memory efficiency (M.E)
  - Highest if we can access the hierarchy as fast as the fastest level
    - i.e. at level h in a hierarchy with levels h, h+1, ...
- Define $M.E = 100 * (T_h / T_{eff})$
- → $M.E = 100 / (1 + P_{miss} (R-1))$
  - Where $R = T_{h+1} / T_h$
  - M.E. = 100 when R=1 or Pmiss=0
    - When will R be 1?
  - Consider
    - R=10 (CPU/SRAM),
    - R= 50(CPU/DRAM)
    - R= 10000 (CPU/disk)
  - Calculate the P_{miss} for good M.E (say 95) for each level

From Prof. Shan's slides

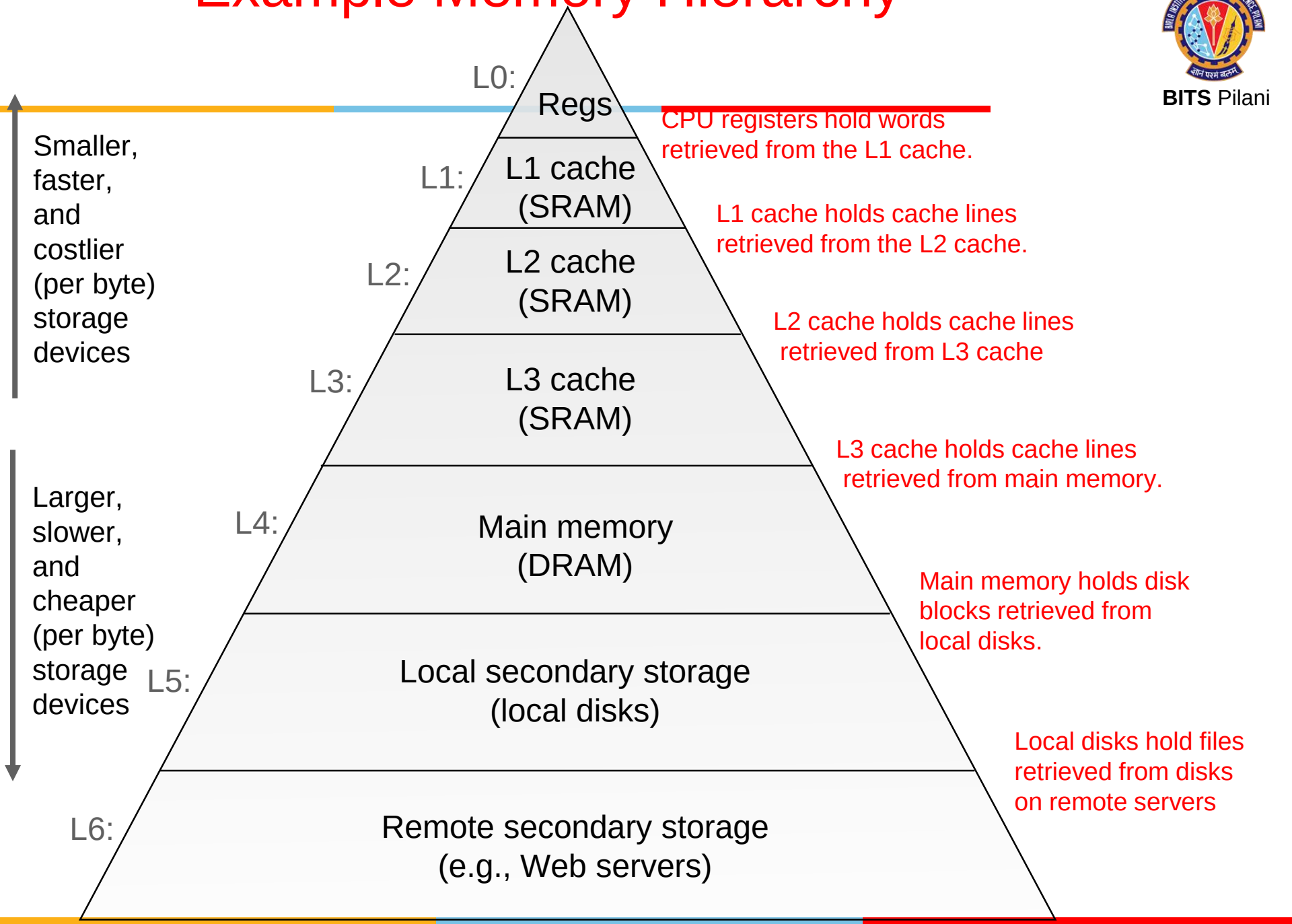
Memory Technologies – Access Time

BITS Pilani

- Cache between CPU registers and main memory
  - Static RAM (6 transistors per cell)
  - Typical Access Time ~10ns (or less)
- Main Memory
  - Dynamic RAM (1 transistor + 1 capacitor)
  - Capacitive leakage results in loss of data
    - Needs to be refreshed periodically – hence the term “dynamic”
  - Typical Access Time ~50ns
  - Typical Refresh Cycle ~100ms.
- Flash Memory
  - Typical access time ~20-100 microseconds

From Prof. Shan’s slides

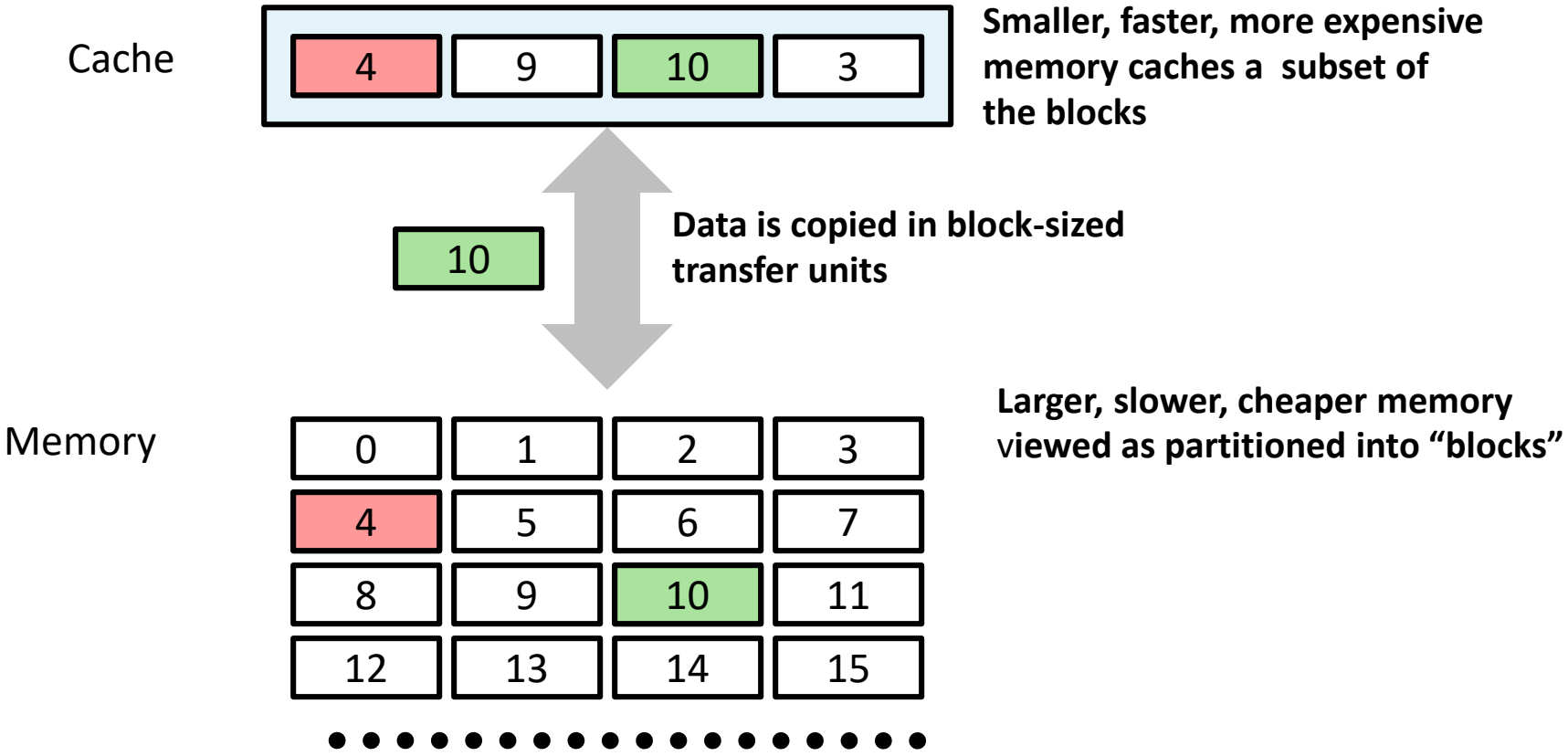
Example Memory Hierarchy



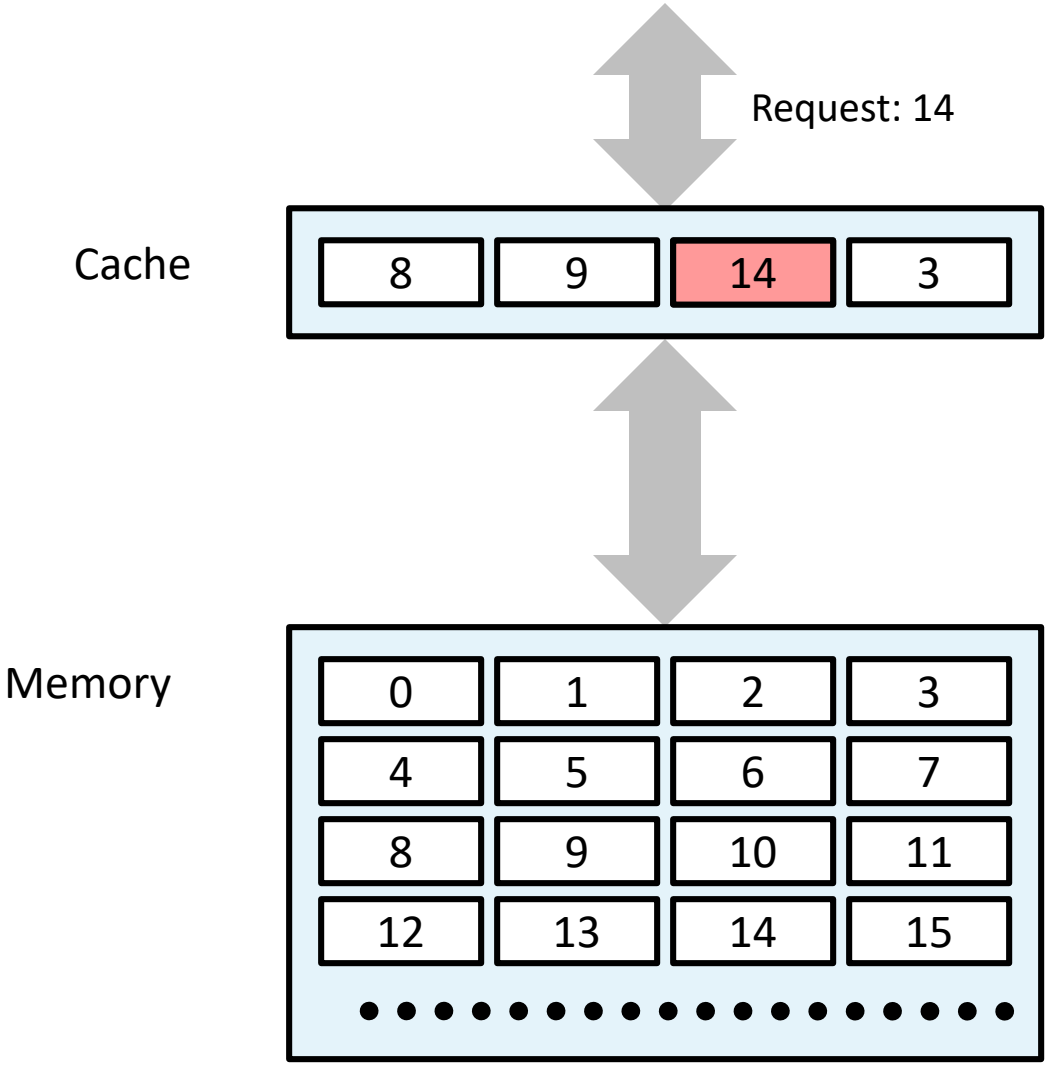
Caches

- *Cache:* A smaller, faster storage device that acts as a staging area for a subset of the data in a larger, slower device.
- Fundamental idea of a memory hierarchy:
  - For each k, the faster, smaller device at level k serves as a cache for the larger, slower device at level k+1.
- Why do memory hierarchies work?
  - Because of locality, programs tend to access the data at level k more often than they access the data at level k+1.
  - Thus, the storage at level k+1 can be slower, and thus larger and cheaper per bit.
- *Big Idea:* The memory hierarchy creates a large pool of storage that costs as much as the cheap storage near the bottom, but that serves data to programs at the rate of the fast storage near the top.

General Cache Concepts



General Cache Concepts: Hit

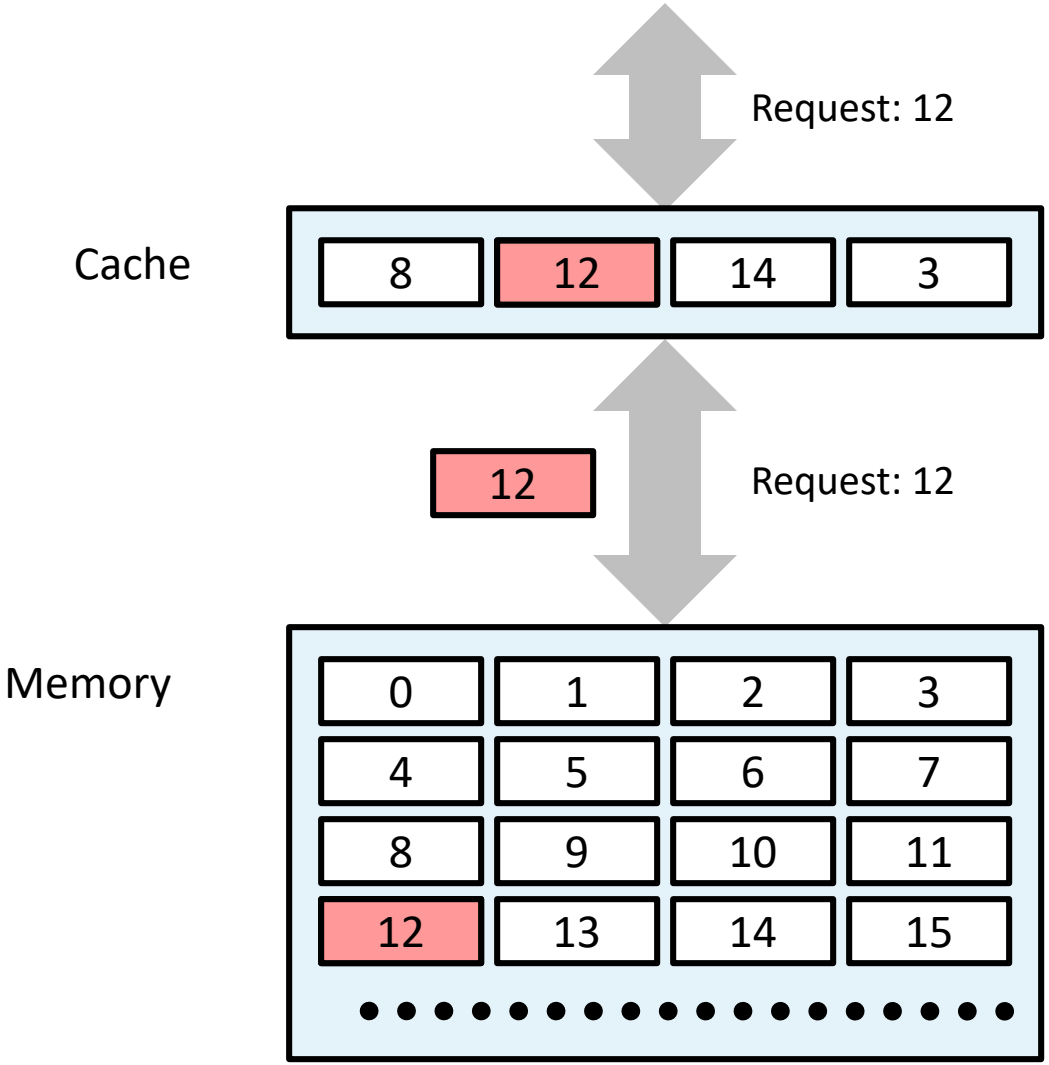


Data in block b is needed

Block b is in cache:
Hit!

When a program needs a particular data object *d* from level *k* + *1*, it first looks for *d* in one of the blocks currently stored at level *k*. If d happens to be cached at level *k*, then we have what is called a cache hit.

General Cache Concepts: Miss



Data in block b is needed

Block b is not in cache:
Miss!

Block b is fetched from *memory*

Block b is stored in cache

- Placement policy: determines where b goes
- Replacement policy: determines which block gets evicted (victim)

General Caching Concepts: Types of Cache Misses

- Cold (compulsory) miss
  - Cold misses occur because the cache is empty.
- Conflict miss
  - Most caches limit blocks at level k+1 to a small subset (sometimes a singleton) of the block positions at level k.
    - E.g. Block i at level k+1 must be placed in block (i mod 4) at level k.
  - Conflict misses occur when the level k cache is large enough, but multiple data objects all map to the same level k block.
    - E.g. Referencing blocks 0, 8, 0, 8, 0, 8, ... would miss every time.
- Capacity miss
  - Occurs when the set of active cache blocks (working set) is larger than the cache.

Cache Management

- The essence of the memory hierarchy is that the storage device at each level is a cache for the next lower level
  - At each level, some form of logic must manage the cache.
    - something has to partition the cache storage into blocks, transfer blocks between different levels, decide when there are hits and misses, and then deal with them
  - The logic that manages the cache can be hardware, software, or a combination of the two
  - For example, the compiler manages the register file, the highest level of the cache hierarchy
    - It decides when to issue loads and determines which register to store the data in
  - The caches at levels L1, L2, and L3 are managed entirely by hardware logic built into the caches
  - In a system with virtual memory, the DRAM main memory serves as a cache for data blocks stored on disk

Examples of Caching in the Mem. Hierarchy

Cache Type	What is Cached?	Where is it Cached?	Latency (cycles)	Managed By
Registers	4-8 bytes words	CPU core	0	Compiler
TLB	Address translations	On-Chip TLB	0	Hardware MMU
L1 cache	64-byte blocks	On-Chip L1	4	Hardware
L2 cache	64-byte blocks	On-Chip L2	10	Hardware
Virtual Memory	4-KB pages	Main memory	100	Hardware + OS
Buffer cache	Parts of files	Main memory	100	OS
Disk cache	Disk sectors	Disk controller	100,000	Disk firmware
Network buffer cache	Parts of files	Local disk	10,000,000	NFS client
Browser cache	Web pages	Local disk	10,000,000	Web browser
Web cache	Web pages	Remote server disks	1,000,000,000	Web proxy server

References

- Section 6.3 from Computer Systems: A Programmer's Perspective, 3rd Edition by Randal E. Bryant and David R. O'Hallaron, Pearson, 2016

Q&A

innovate achieve lead

BITS Pilani

Pilani Campus

Thank You