Lecture 5.3
Cache Hierarchy

EN 600.320/420

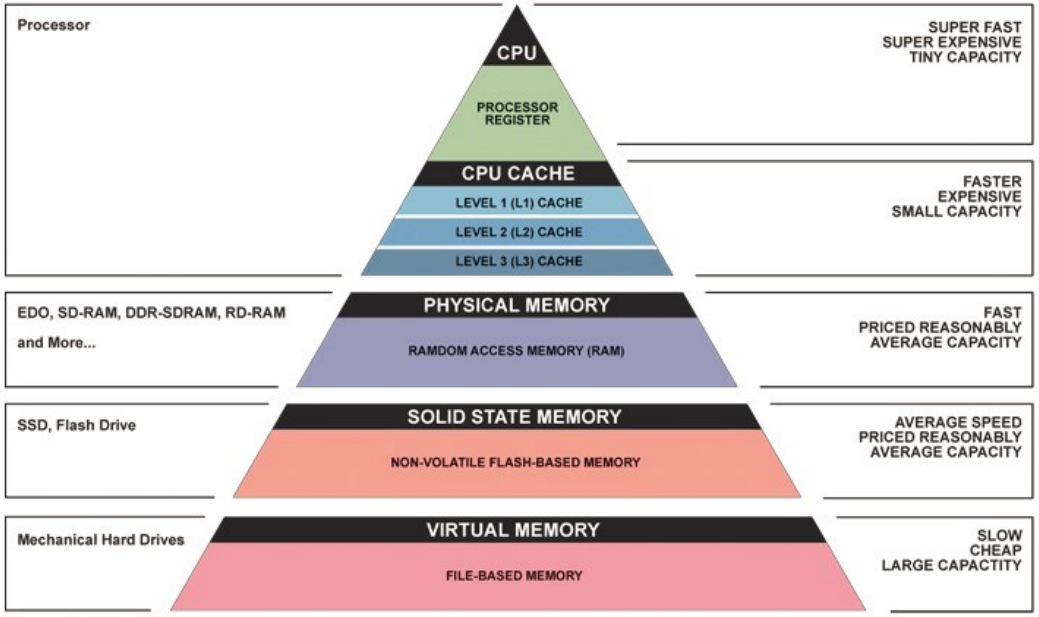
Instructor: Randal Burns

12 February 2018

Department of Computer Science, *Johns Hopkins University*

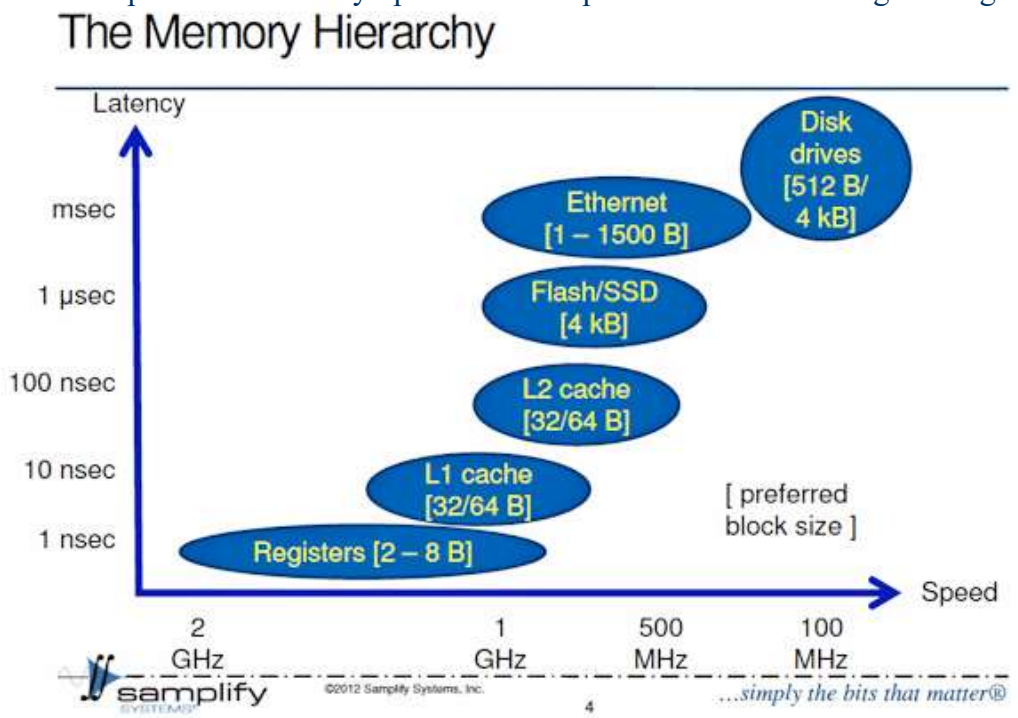
The Cache Hierarchy

- Memory is an abstraction
  - Actually a *steep, hiearchy* of caches
- Different levels have:
  - Performance
  - Capacity
  - Sharing
  - Semantics?



▲ Simplified Computer Memory Hierarchy
Illustration: Ryan J. Leng

https://denalimemoryreport.files.wordpress.com/2012/04/wegener-1.gif



The Cache Hierarchy

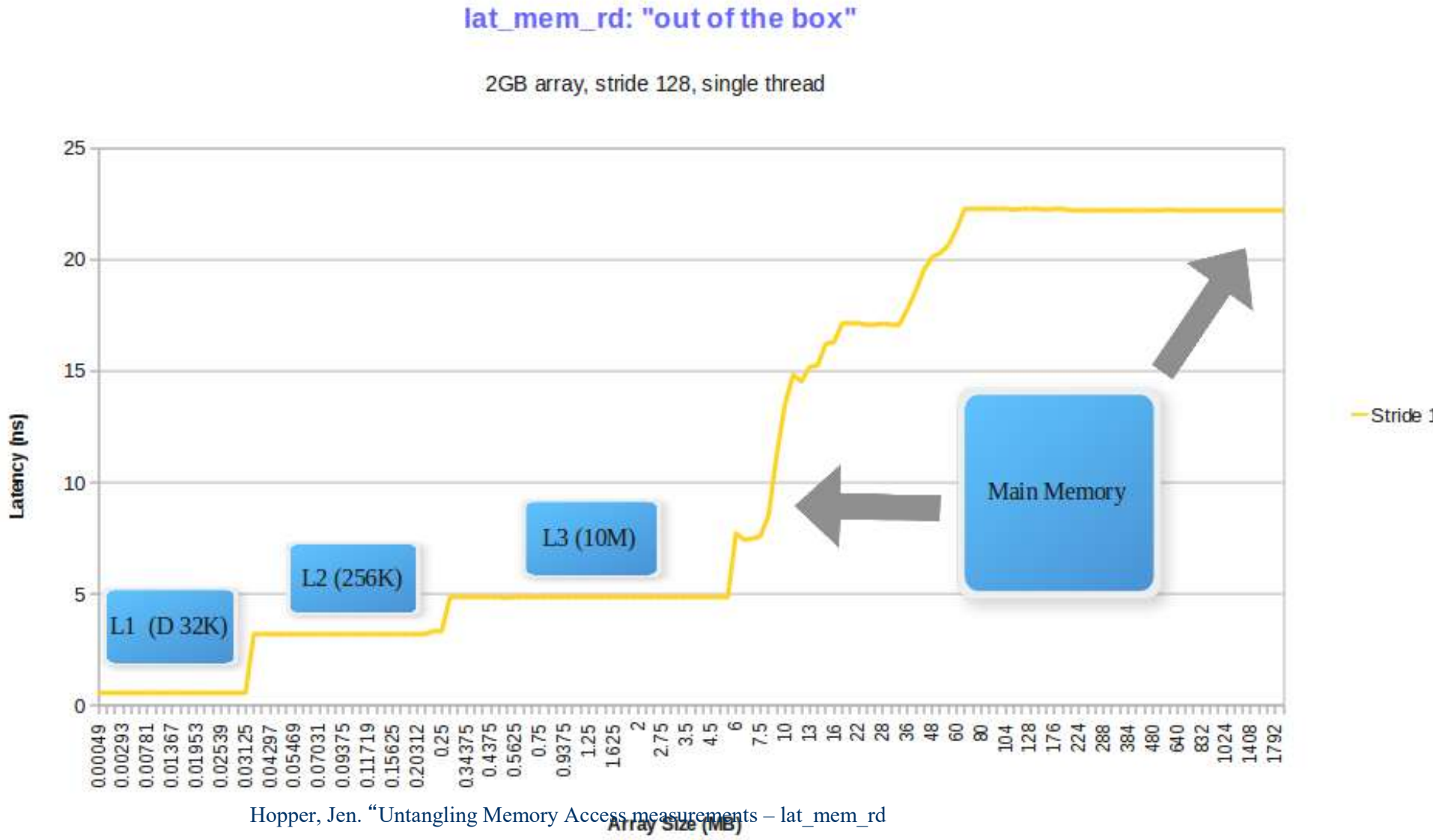
- L1 cache (on chip)
  - Actually 3 L1 caches (Instruction, data, TLB)
  - Typically per processor
- L2 cache (on chip)
  - ~10x slower, 10x bigger, 10x less $$ than L1
  - Typically shared among cores
- L3 (most modern processors)
- Main memory
  - ~10x slower, ~2000x bigger, ~20x less $$ than L2
  - Shared among mutliple processors
- SSDs
  - ~10x slower, 100x bigger, ~20x less $$ than memory
- And disk drives
  - 10x slower, 5x bigger, ~5x less $$ than NVRAM

Cache Performance

- Delays (in clock cycles) to different levels in the cache hierarchy for an i7 (Nehalem)
  - 1 cycle to registers (private to each core)
  - 1 cycle to L1 (private to each core)
  - 4 cycles to L2 (private to each core)
  - 35 cycles to L3 (shared by cores)
  - 145 cycles to memory (shared by processors)
  - 10^5 cycles to NVRAM
  - 10^7 cycles to disk

Cache Latency

- Access every 128th byte in a array of varying size

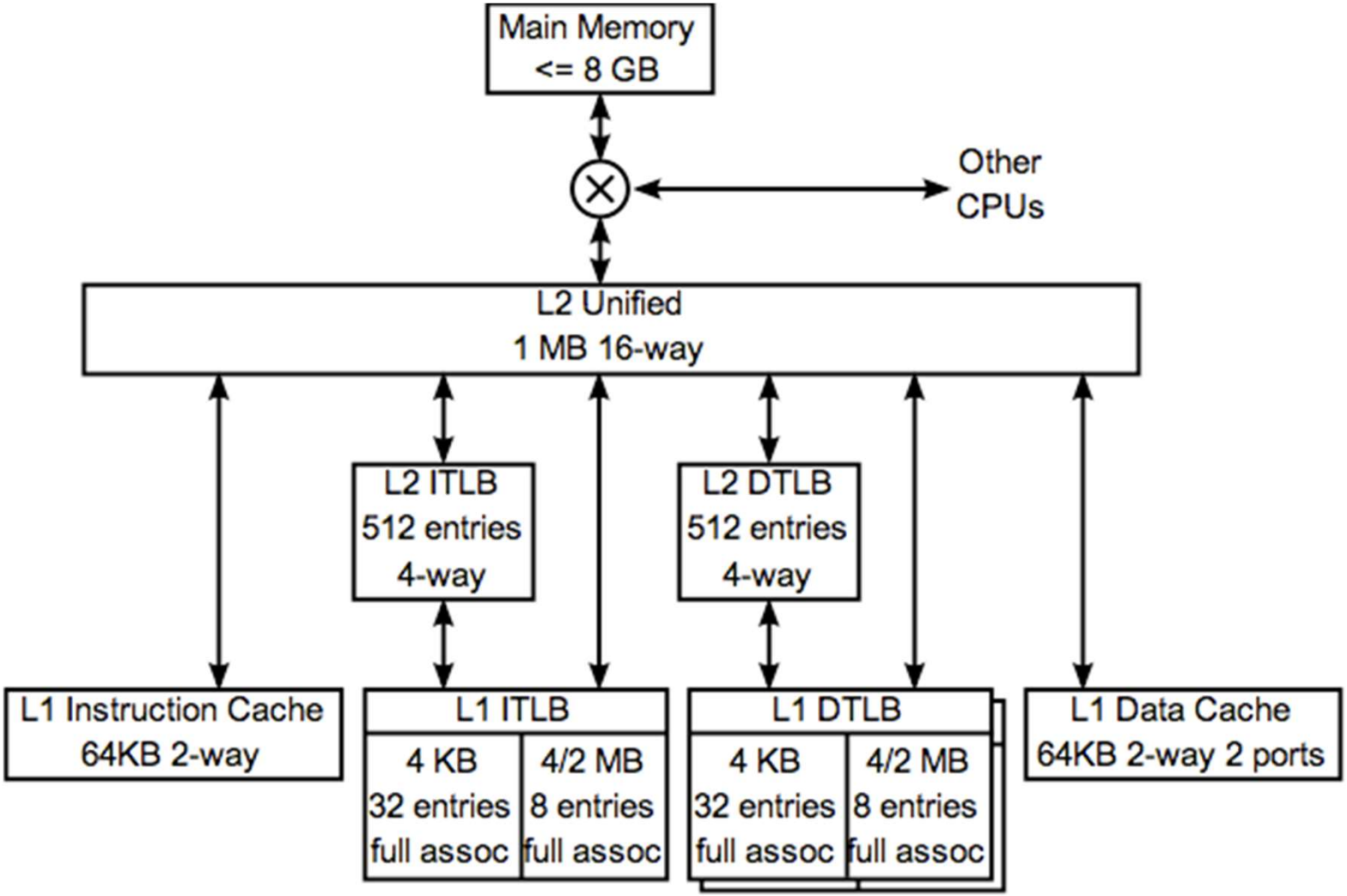


Hopper, Jen. "Untangling Memory Access measurements – lat_mem_rd

Caching Concepts

- Cache line
  - Fundamental unit of caching
  - Line size (typically power of 2)
- Inclusive/exclusive
  - Strictly incl. = all data in L1 must be in L2
  - Exclusive = data in L1 cannot be in L2
- Associativity
  - Number of locations a cache block can go into
- Unified
  - Shared among multiple cores
- All of these features relate to complexity of hardware implementation, density, and utility

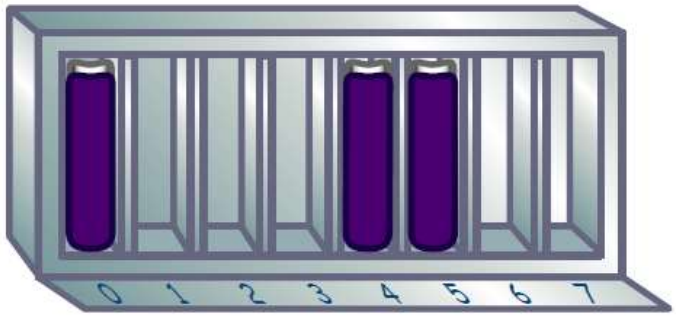
Example: Athlon 64 K8



Associativity Illustrated

Just as bookshelves come in different shapes and sizes, caches can also take on a variety of forms and capacities. But no matter how large or small they are, caches fall into one of three categories: direct mapped, n-way set associative, and fully associative.

Direct Mapped



Tag	Index	Offset

A cache block can only go in one spot in the cache. It makes a cache block very easy to find, but it's not very flexible about where to put the blocks.

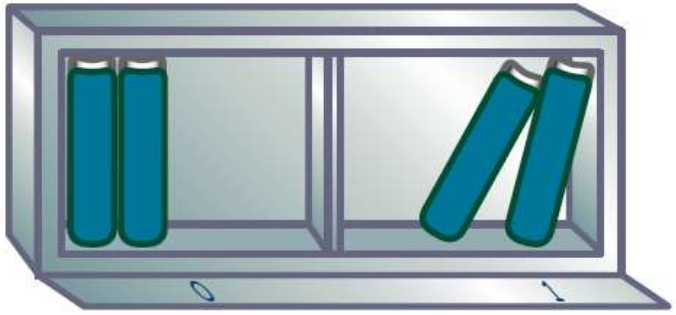
2-Way Set Associative

Tag	Index	Offset

This cache is made up of sets that can fit two blocks each. The index is now used to find the set, and the tag helps find the block within the set.

http://csillustrated.berkeley.edu/PDFs/cache-types.pdf

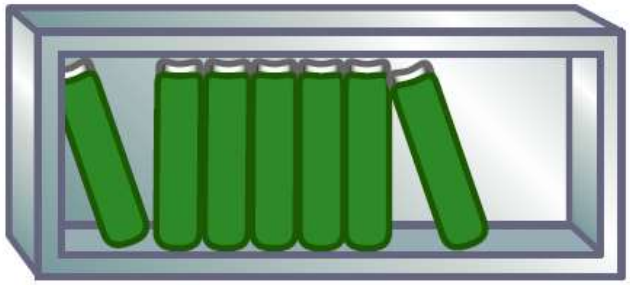
Associativity Illustrated



4-Way Set Associative

Tag	Index	Offset

Each set here fits four blocks, so there are fewer sets. As such, fewer index bits are needed.



Fully Associative

Tag	Offset

No index is needed, since a cache block can go anywhere in the cache. Every tag must be compared when finding a block in the cache, but block placement is very flexible!

http://csillustrated.berkeley.edu/PDFs/cache-types.pdf

So What

- You can't just access memory
  - Different memory access patterns result in 50x performance differences for the same computation
- Worry about:
  - Parallelism: am I using all the data in a cache line
    - To access a single byte, one must load a whole line
    - Sequential access to memory is always parallel!
  - Sharing/reuse: is my program referencing data in the cache more than once? At what levels?
- Memory terms:
  - Aligned – access range starts/ends on cache line boundaries
  - Sequential – a continuous range of bytes
  - Coalesced – uses all "banks" (associative regions) of memory