Lecture 20: Multi-Cache Designs

Spring 2024
Jason Tang

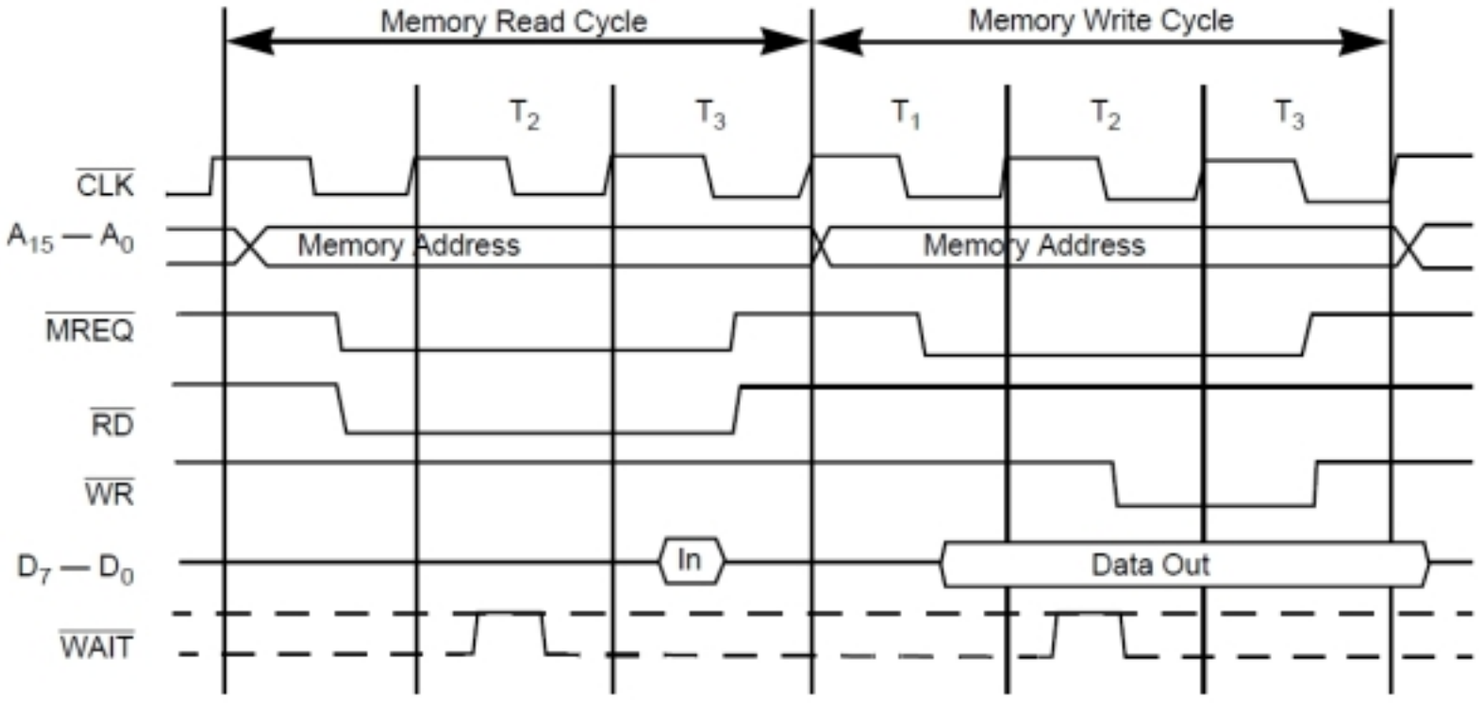
Topics

- Split caches
- Multi-level caches
- Multiprocessor caches

3 Cs of Memory Behaviors

- Classify all cache misses as:
    - Compulsory Miss (also cold-start miss): caused by first access to a block that has never been in cache
    - Capacity Miss: caused when cache cannot contain all blocks needed during execution
    - Conflict Miss (also collision miss): multiple blocks compete to be stored within same set

Memory Control Lines



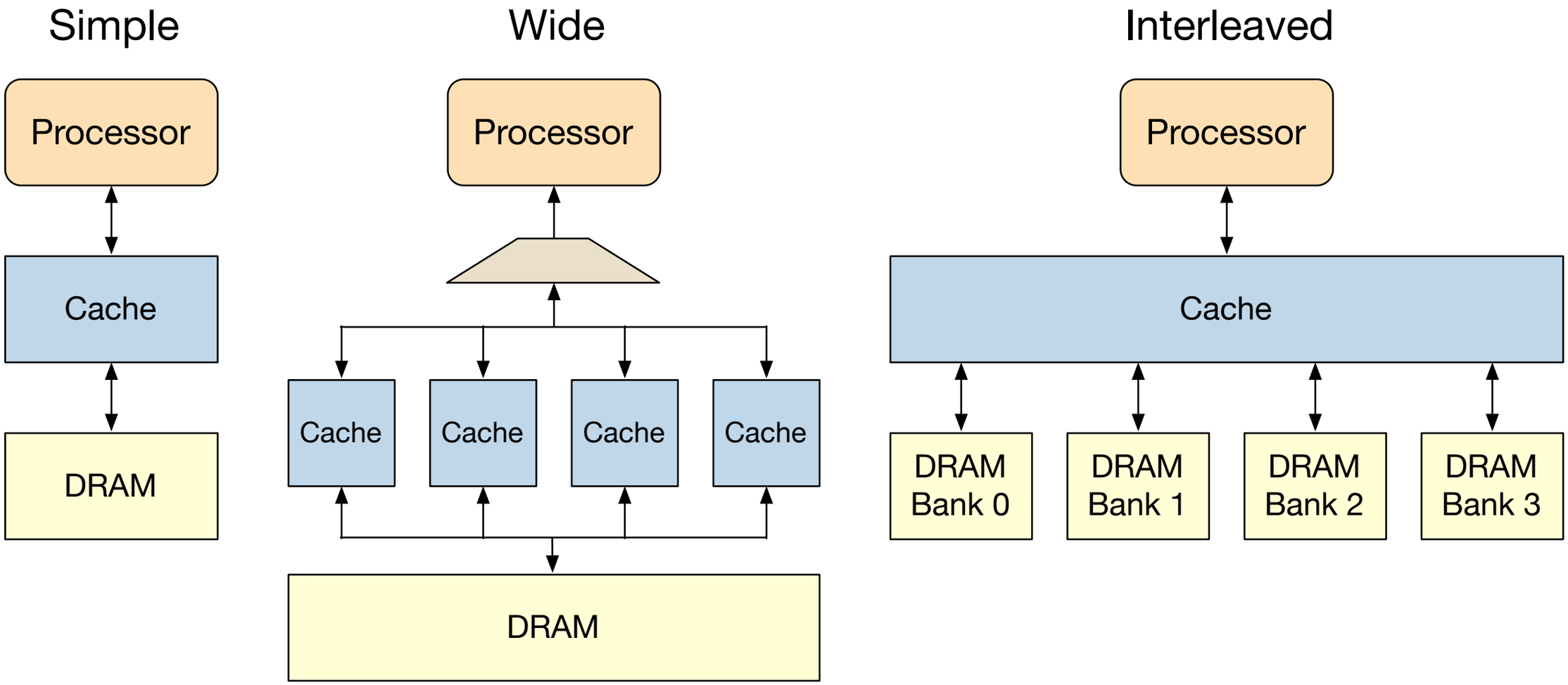
- Main memory takes a while to access, as compared to caches

- CPU is stalled while waiting for data to be read

z80 Line	Meaning
Clk	Inverted clock (falling edge)
A_{15} - A_0	Address bus
MREQ	Memory request: active low when address line is valid
RD	Active low to request a read
WR	Active low to request a write
D_t - D_0	Data bus
Wait	Active high to stall CPU

http://www.primrosebank.net/computers/mtx/components/memory/dram/dram_mtx1.htm

Memory Organization



- Wide: DRAM's data blocks larger than CPU's data size

- Interleaved: Concurrent, overlapping DRAM accesses, via chip select lines

Split Caches

- Unified Cache: single cache holds both instructions and data
    - Single set of control logic, but can have multiple cache misses in a single clock cycle
- Split Cache: two independent caches, operating in parallel (“Modified Harvard” architecture)
    - Instruction Cache: optimized for instructions
    - Data Cache: optimized for data

Instruction Cache

- Every instruction has a data access (at the address of the `PC`)
    - But not every instruction is a data load/store
- Usually implemented to assume only reads (no self-modifying code)
- Uses prefetching to anticipate next *sequential* instruction to execute
- May have a branch target cache to prefetch instruction at branch target

Data Cache

- Supports reads and writes
    - In addition to cache tag, each cache entry has a dirty bit indicating that contents have been modified, but not yet committed to main memory
- Self-modifying code requires maintaining coherency between memory systems
    - Write new instructions to data cache
    - Flush data cache
    - Then invalidate instruction cache
- Need not have same characteristics as instruction cache

Cache Replacement Policies

- Upon a conflict miss, need to decide which victim cache block to evict
  - Random: victim block randomly selected
  - Round-Robin: victim block is the first one that was loaded
  - Least Recently Used (LRU): victim block is one that has been unused the longest
- LRU is generally better, but harder to implement in hardware

Cache Locking

- Some cache designs allow for a given cache block to be locked
  - Cache block will not be evicted upon a conflict miss
- Important for hard real-time software to reduce cache miss penalty
- Reduces amount of cache available to other software

https://www.ece.umd.edu/~barua/anand-TECS-2014.pdf

Cache Timing Attacks

- Because CPI is higher for cache hits than misses, computer systems are vulnerable to cache timing attacks

- By carefully measuring how long certain operations take, an attacker can deduce the contents of memory

- Evict+Time: execution time changes by evicting specific cache set (such as a crucial branch instruction)

- Prime+Probe: fill cache with data, execute victim program, then observe which cache sets are still filled

- Flush+Reload: flush a *shared* cache line, execute victim program, then time how long it takes to read from that cache line

https://www.usenix.org/system/files/conference/usenixsecurity16/sec16_paper_lipp.pdf

Cache Timing Attack Example

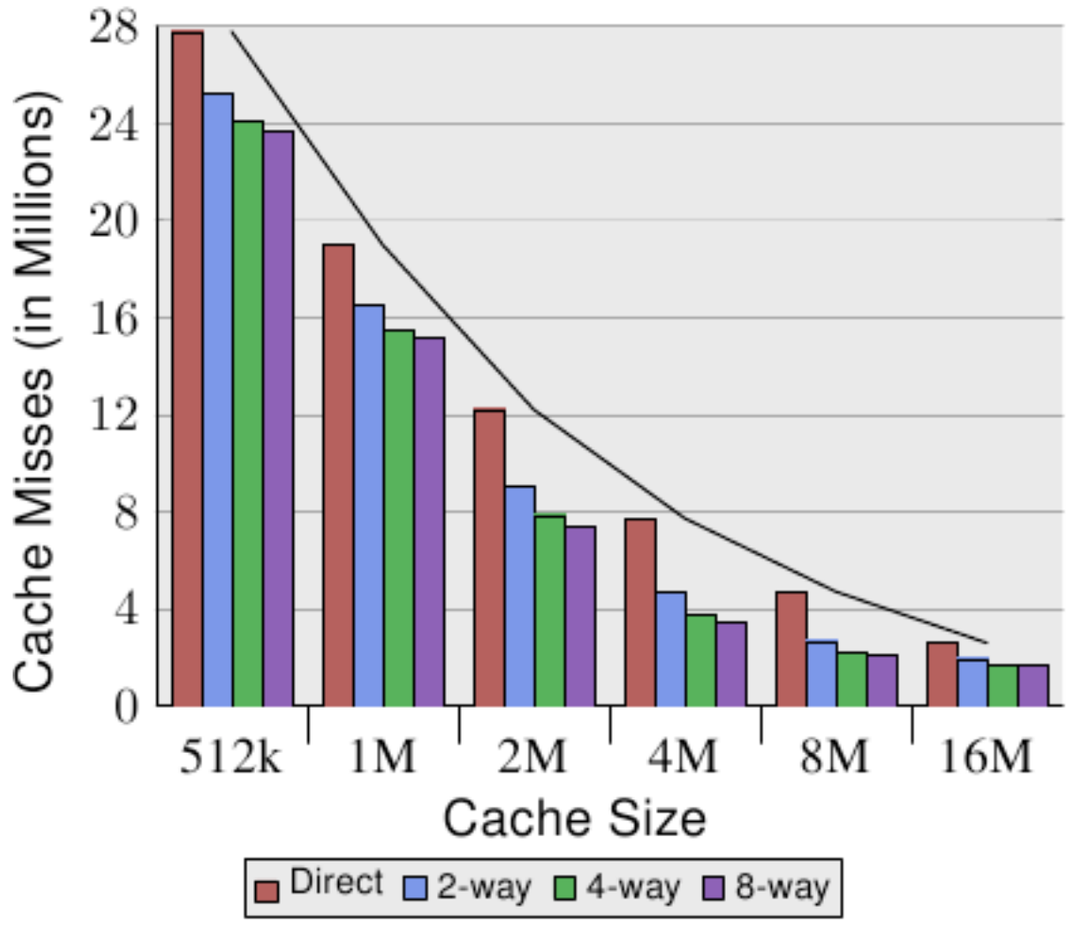
- A cached value is faster to access than an uncached value
    - Suppose that a user allocates a large amount of memory (via `malloc()`). Cache miss occurs the first time a byte is read from it. The CPU determines which cache block is associated with that byte's address. It must then evict a cache block from the set-associative data cache, and then load from main memory the cache block. This takes many tens to hundreds of nanoseconds.
- Attacker ensures that a range of addresses are uncached (by either flushing data cache, or by reading other addresses that map to same cache set)
    - Attacker performs some operation, then carefully measures the time it takes to read each byte from the memory range. A faster read means that byte was cached.

Multi-Level Caches

- Modern systems have multiple levels of cache
  - Level 1 (L1) (closest to processor) is smaller and faster
  - Other levels are bigger, but slower
- Different characteristics between cache levels
  - L1 cache has smaller blocks and fewer associativity, to be faster
  - L2 cache has larger blocks and more associativity, to reduce miss rate

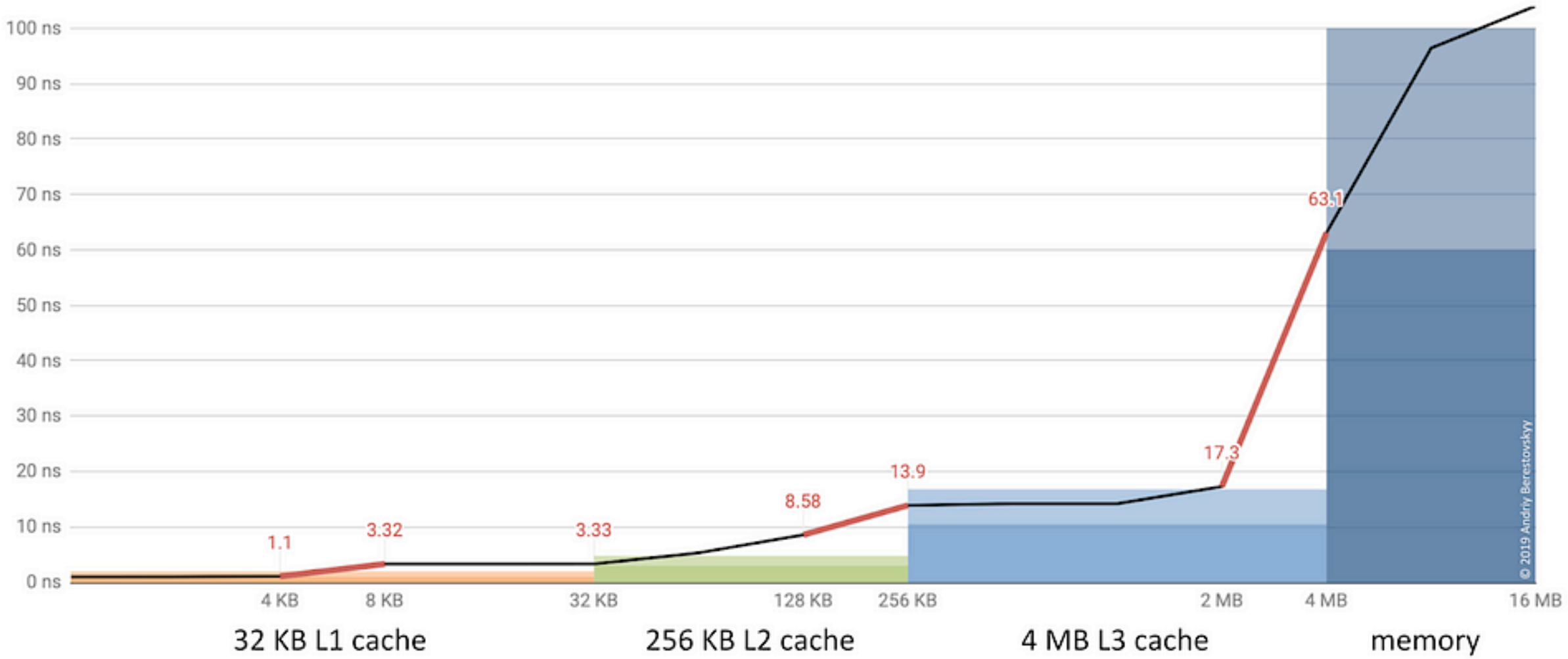
L1 vs. L2 Cache

- Infeasible to have an infinite sized cache
- L2 cache is larger, cheaper, and requires less power than L1, but a bit slower
- Newer systems have even more levels of caching
  - Intel's Haswell architecture added a 128 MiB L4 cache



https://lwn.net/Articles/252125/

Intel Kaby Lake Access Latency



https://www.nexthink.com/blog/smarter-cpu-testing-kaby-lake-haswell-memory

ARM Cortex-A53 Cache Systems

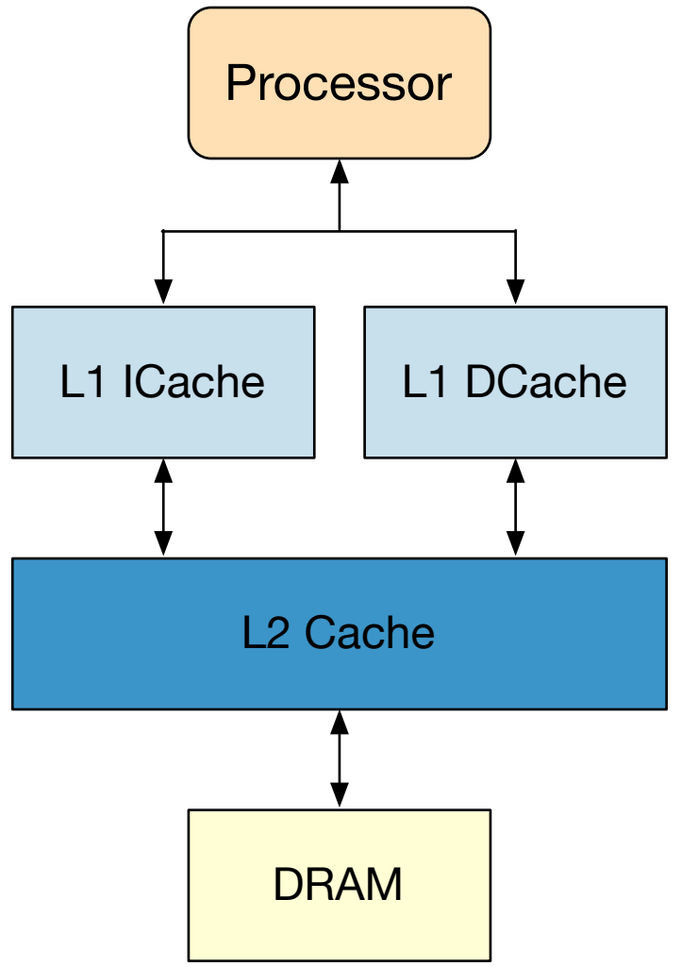
- L1 ICache is up to 64 KiB, has 64-bit cache lines, is 2-way set associative, and has a 128-bit read interface to L2

- L1 DCache is up to 64 KiB, has 64-bit cache lines, is 4-way set associative, has a 128-bit read interface to L2, and a 256-bit write interface to L2

- L1 ICache and DCache have random replacement

- L2 Cache is up to 2 MiB, has 64-bit cache lines, and is 16-way set associative

- L2 Cache has LRU replacement



ARM Cortex-A53 Processor Technical Manual, §6.1 and §7.1

Intel Golden Cove Cache Systems

- L1 ICache is set associative (8 ways of 64 sets), 32 KiB total, ? cycle latency
- L1 DCache is set-associative (12 ways of 64 sets), 48 KiB total, 5 cycle latency
- L2 Unified Cache is set-associative (10 ways of 2048 sets), 1280 KiB total, 15 cycle latency
- L3 Unified Cache is set-associative (12 ways of 40960 sets), 30 MiB total, 67 cycle latency

https://chipsandcheese.com/2022/02/11/going-armchair-quarterback-on-golden-coves-caches/

Cache Inclusion Policy

- Multi-level caches are designed depending upon if data in one cache level are also in other cache levels
  - Inclusive Policy: Same data in both L1 and L2 caches
  - Exclusive Policy: Data in only one cache
- Exclusive policy increases effective amount of caching, but:
  - If data in L2 but not L1, then block is moved from L2 to L1
  - If this causes an eviction from L1, then victim cache block moved to L2
- Non-inclusive Non-exclusive (NINE) policy is a blend of inclusive and exclusive policies

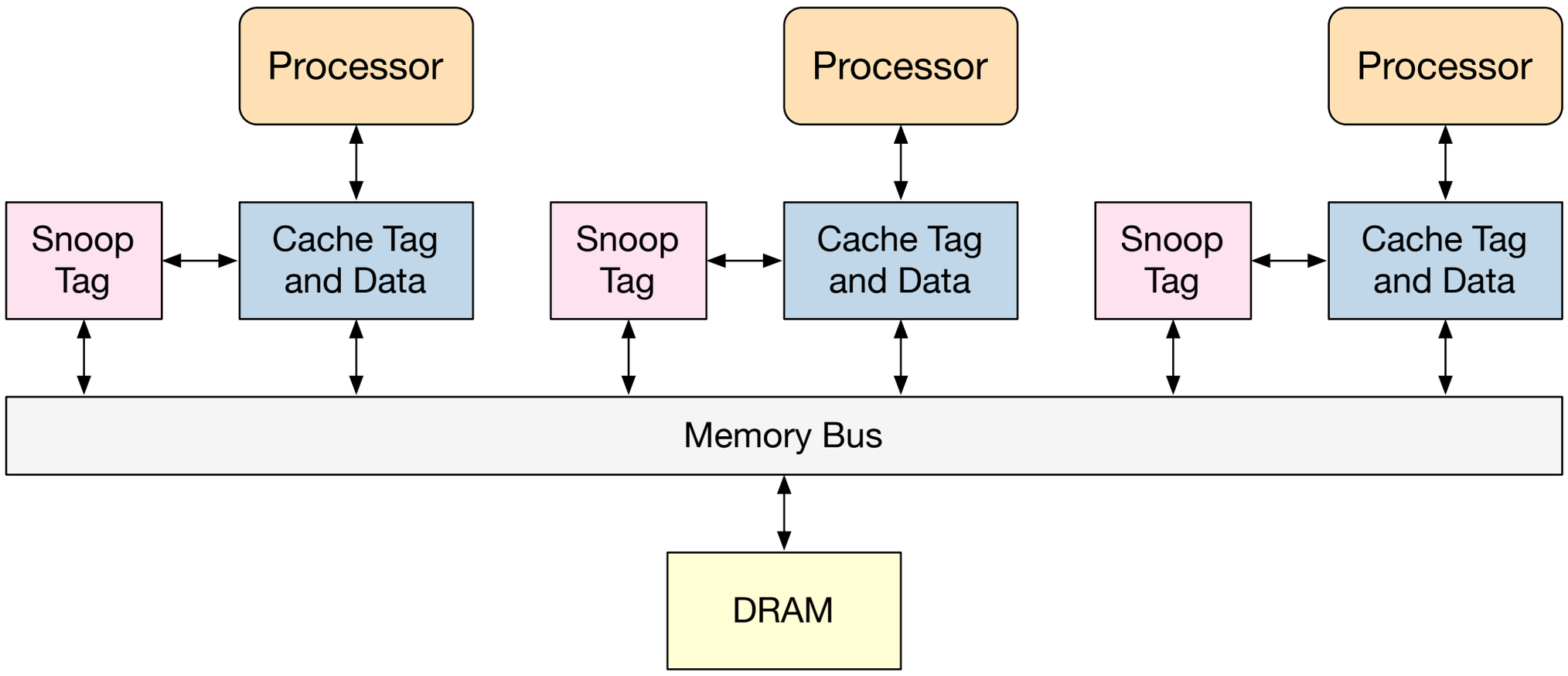
Comparison of Cache Inclusion Policies

Cache Policy	Data Not in L1, but in L2	Data Neither in L1 nor L2	Data evicted from L2
Inclusive	Cache block fetched from L2; evicted L1 cache entry is not saved	Read from main memory to both L1 and L2	Back Invalidation: matching L1 cache block also invalidated
Exclusive	Cache block copied from L2 to L1; evicted L1 cache entry moved to L2	Victim Cache: Read from main memory directly to L1; evicted L1 cache entry moved to L2	No change to L1
NINE	Cache block fetched from L2; evicted L1 cache entry is not saved (same as inclusive)	Read from main memory to both L1 and L2 (same as inclusive)	No change to L1 (same as exclusive)

Multiprocessor Caches

- On a symmetric multiprocessing (SMP) system, some caches are exclusive to each processor and others are shared
  - Often L1 and L2 caches are exclusive, while L3 and higher are shared
- Cache coherence problem: when different processors' caches store different copies of the same data
  - Write Propagation: If processor **A** writes to address **X**, processor **B** should read new value
  - Write Serialization: If processor **A** and then **B** write to address **X**, a later read from **X** should return **B**'s data, not **A**'s

Cache Snooping



- All cache controllers monitor (snoop) on a shared bus for memory transactions
  - Whenever a processor writes to a cache block, it broadcasts a message

Snooping Protocols

- Write Invalidate Protocol: All other caches containing that block are invalidated, similar to a write-back policy
  - Most common snooping protocol

Processor Activity	Bus Activity	CPU A's Cache	CPU B's Cache	Contents at Address X
		uncached	uncached	0
CPU A reads X	Cache miss for X	0		0
CPU B reads X	Cache miss for X	0	0	0
CPU A writes a 1 to X	Invalidate X	1	invalid	1
CPU B reads X	Cache miss for X	1	1	1

- Write Update Protocol: All other caches are updated to contain the new cache block, similar to a write-through policy

MESI protocol

- Common write invalidate protocol
- In addition to invalid (I) and modified (M) bits, a cache block also has these status bits:
  - Exclusive (E): cache block present only in current cache and is clean
  - Shared (S): cache block may be stored in other caches and is clean