计算机组成与系统结构 Computer Organization & System Architecture

Huang Kejie (黄科杰) 百人计划研究员

Office: 玉泉校区老生仪楼 304

Email address: huangkejie@zju.edu.cn

HP: 17706443800

Sources of Cache Misses (3 C's)

- *Compulsory* (cold start, first reference):
  - 1st access to a block, not a lot you can do about it
    - If running billions of instructions, compulsory misses are insignificant
  - misses that would occur even with infinite cache

- *Capacity*:
  - cache is too small to hold all data needed by the program
    - Misses that would not occur with infinite cache
    - misses that would occur even under perfect replacement policy

- *Conflict* (collision):
  - Misses that occur because of collisions due to line-placement strategy (multiple memory locations mapped to same cache set)
    - Misses that would not occur with ideal fully associative cache

How to Calculate 3C's Using Cache Simulator

- *Compulsory:* set cache size to infinity and fully associative, and count number of misses

- *Capacity:* Change cache size from infinity, usually in powers of 2, and count misses for each reduction in size
  - 16 MB, 8 MB, 4 MB, … 128 KB, 64 KB, 16 KB

- *Conflict:* Change from fully associative to n-way set associative while counting misses
  - Fully associative, 16-way, 8-way, 4-way, 2-way, 1-way

Effect of Cache Parameters on Performance

- Larger cache size
  - + reduces capacity and conflict misses
  - – hit time will increase

- Higher associativity
  - + reduces conflict misses
  - – may increase hit time

- Larger line size
  - + reduces compulsory and capacity (reload) misses
  - – increases conflict misses and miss penalty

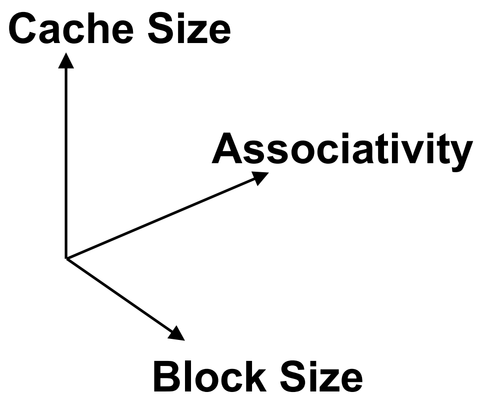
Primary Cache Parameters

- Block size
  - How many bytes of data in each cache entry?
- Associativity
  - How many ways in each set?
  - Direct-mapped => Associativity = 1
  - Set-associative => 1 < Associativity < #Entries
  - Fully associative => Associativity = #Entries
- Capacity (bytes) = Total #Entries * Block size
- #Entries = #Sets * Associativity

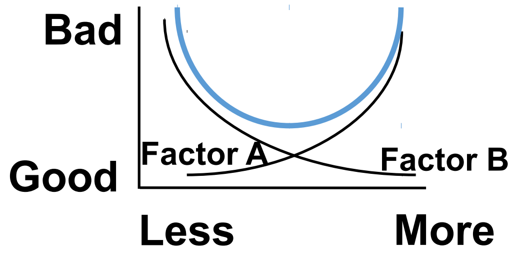
Cache Design Space

Computer architects expend considerable effort optimizing organization of cache hierarchy – big impact on performance and power!

- Several interacting dimensions
  - Cache size
  - Block size
  - Associativity
  - Replacement policy
  - Write-through vs. write-back
  - Write allocation



- Optimal choice is a compromise
  - Depends on access characteristics
    - Workload
    - Use (I-cache, D-cache)
  - Depends on technology / cost



- Simplicity often wins

Six Basic Cache Optimizations

Larger block size

- Reduces compulsory misses
- Increases capacity and conflict misses, increases miss penalty

Larger total cache capacity to reduce miss rate

- Increases hit time, increases power consumption

Higher associativity

- Reduces conflict misses
- Increases hit time, increases power consumption

Higher number of cache levels

- Reduces overall memory access time

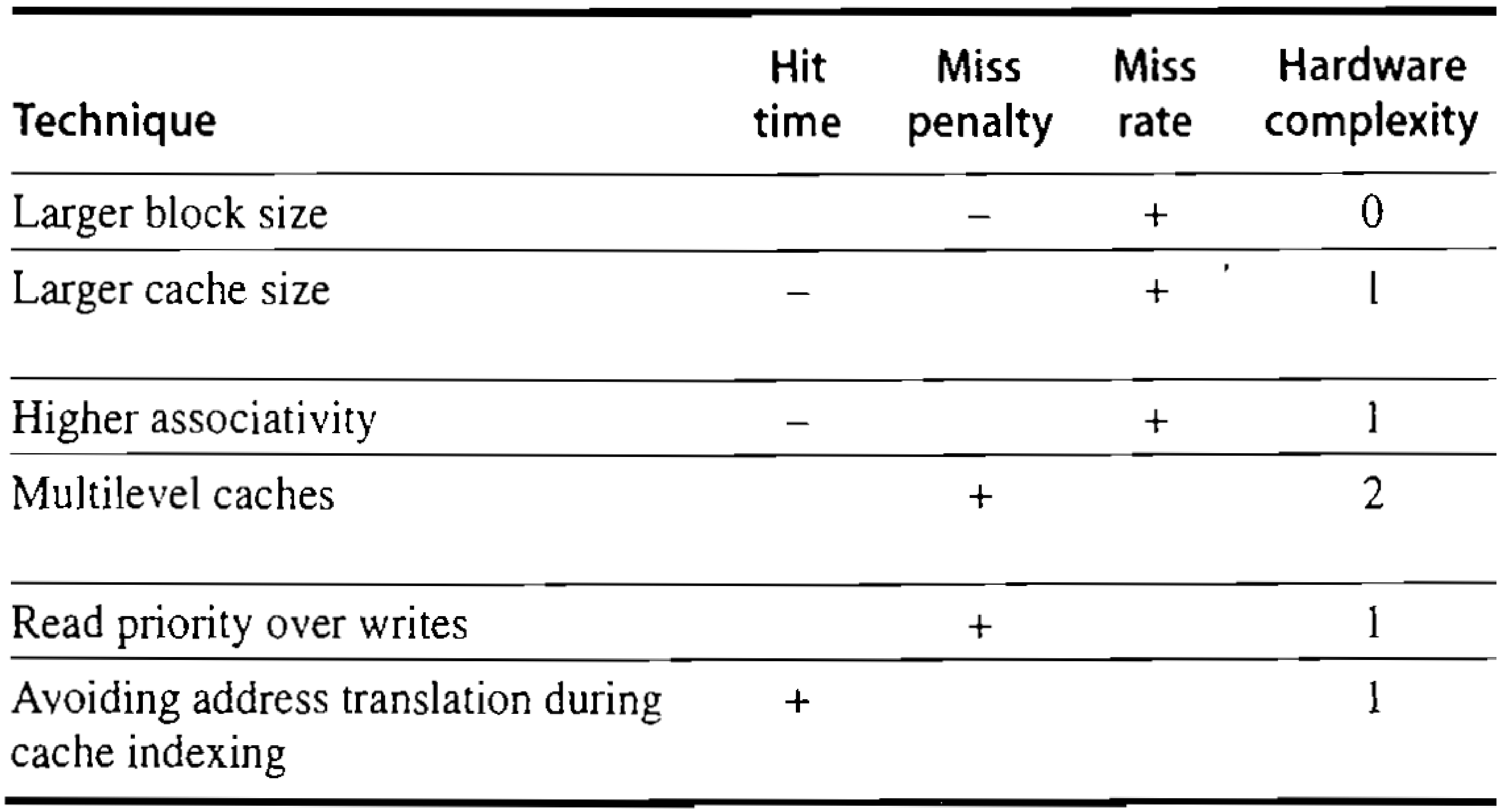
Giving priority to read misses over writes

- Reduces miss penalty

Avoiding address translation in cache indexing

- Reduces hit time

Impact of Cache Performance and Complexity



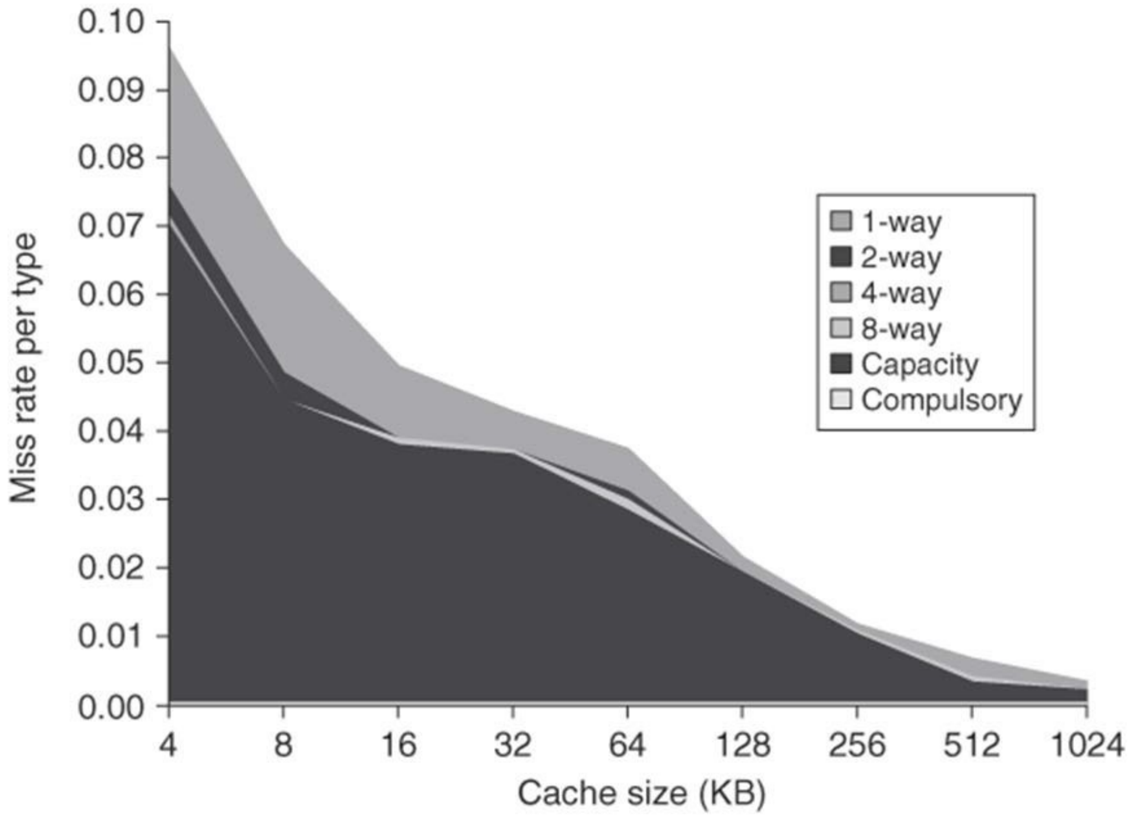
Technique	Hit time	Miss penalty	Miss rate	Hardware complexity
Larger block size		–	+	0
Larger cache size	–		+	1
Higher associativity	–		+	1
Multilevel caches		+		2
Read priority over writes		+		1
Avoiding address translation during cache indexing	+			1

Impact of Larger Cache on AMAT?

- 1) Reduces misses (what kind(s)?)
- 2) Longer Access time (Hit time): smaller is faster
  - Increase in hit time will likely add another stage to the pipeline
- At some point, increase in hit time for a larger cache may overcome improvement in hit rate, yielding a decrease in performance
- Computer architects expend considerable effort optimizing organization of cache hierarchy – big impact on performance and power!

Larger Caches (Miss Rate↓)

- Good for reducing capacity misses
  - Drawback: potentially longer hit time, and
  - higher cost and power
  - Popular in off-chip caches

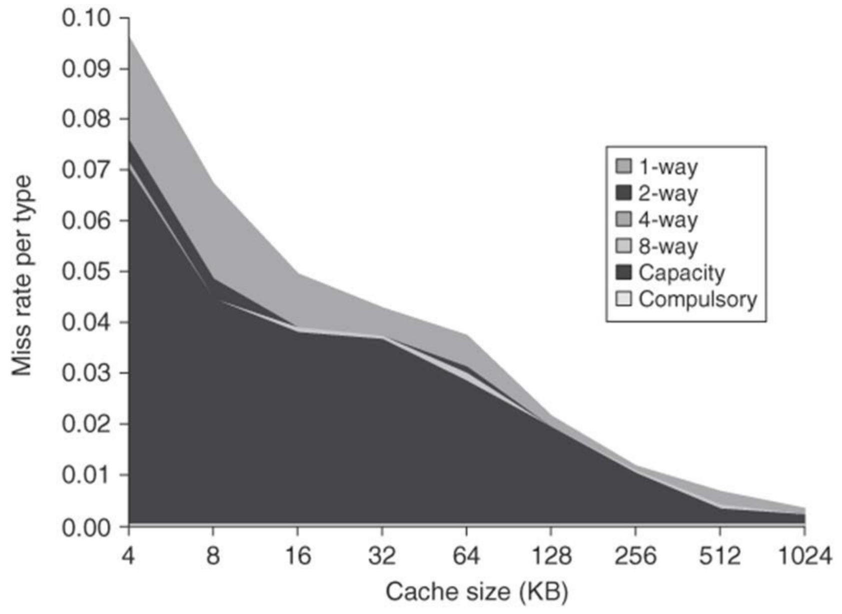


Increasing Associativity?

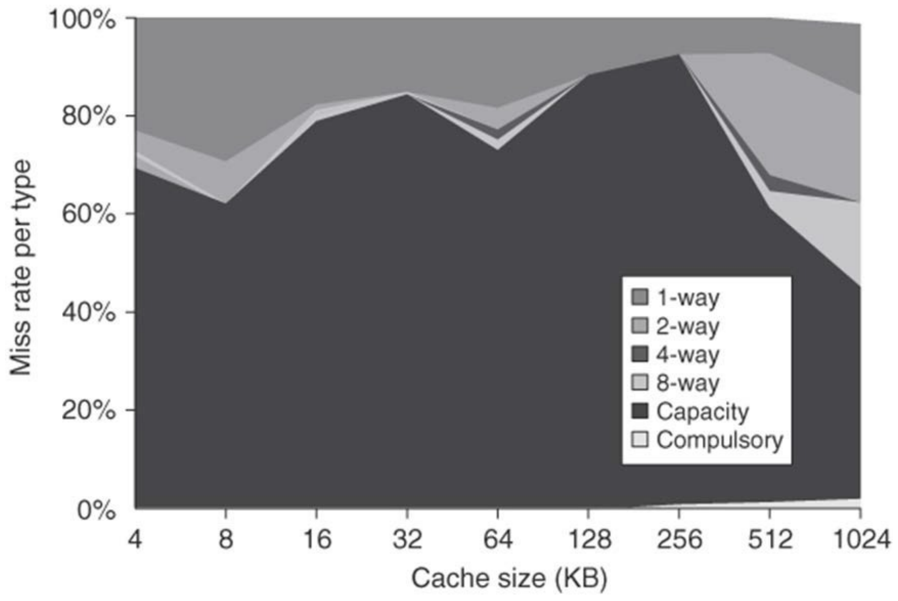
- Hit time as associativity increases?
  - Increases, with large step from direct-mapped to >=2 ways, as now need to mux correct way to processor
  - Smaller increases in hit time for further increases in associativity
- Miss rate as associativity increases?
  - Goes down due to reduced conflict misses, but most gain is from 1->2->4-way with limited benefit from higher associativities
- Miss penalty as associativity increases?
  - Unchanged, replacement policy runs in parallel with fetching missing line from memory

Higher Associativity (Miss Rate↓)

- n-way cache, we increase n with a higher associativity, the cache has a smaller number of sets.

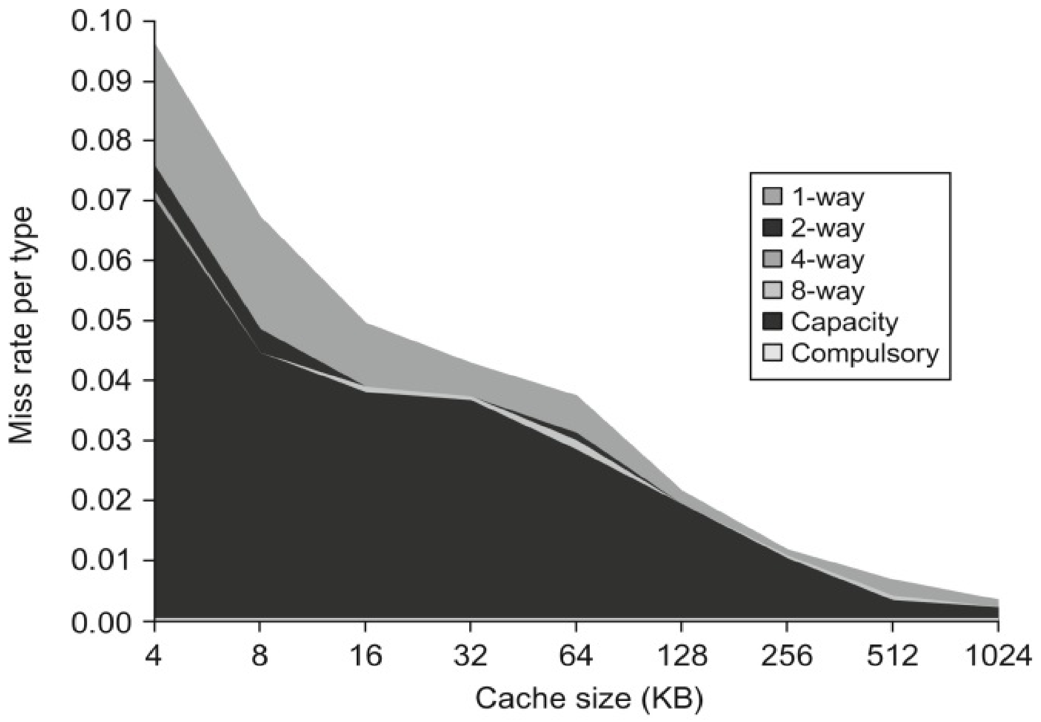


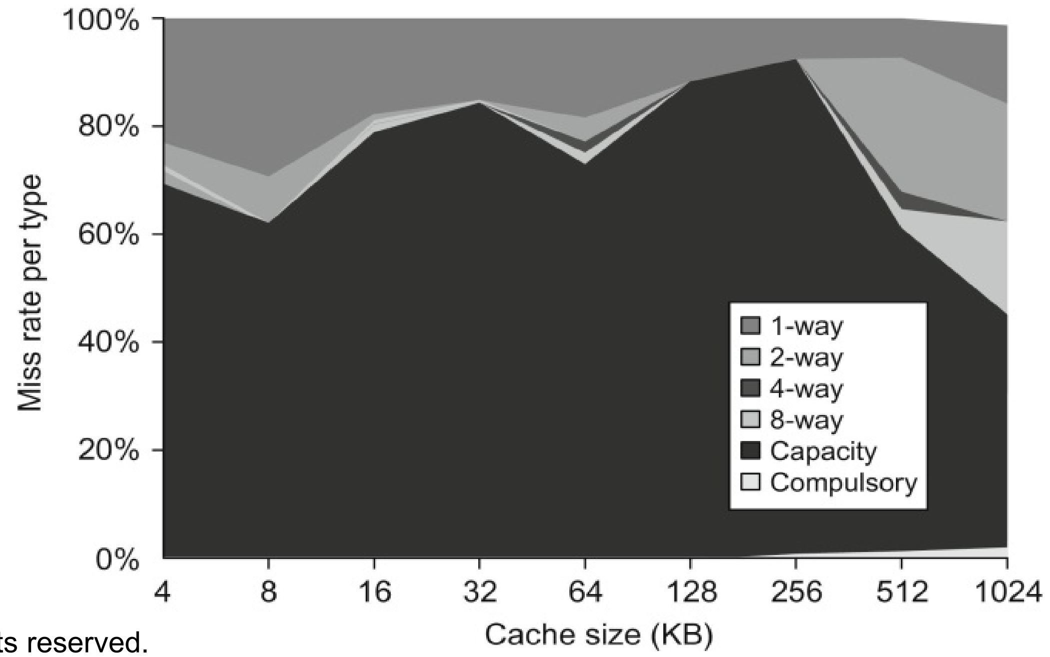
Total miss rate



Distribution of miss rate

Figure B.9 Total miss rate (top) and distribution of miss rate (bottom) for each size cache according to the three C's for the data in Figure B.8. The top diagram shows the actual data cache miss rates, while the bottom diagram shows the percentage in each category. (*Space allows* the graphs to show one extra cache size than can fit in Figure B.8.)





Higher Associativity (cont')

- Two general rules of thumb
  - Eight-way set associative is for practical purposes as effective in reducing misses for these sized caches as fully associative
  - 2:1 cache rule of thumb. A direct-mapped cache of size N has about the same miss rate as a two-way set associate cache of size N/2

Peer Instruction

- For a cache of fixed capacity and block size, what is the impact of increasing associativity on AMAT:

  A [✓ Increases hit time, decreases miss rate]

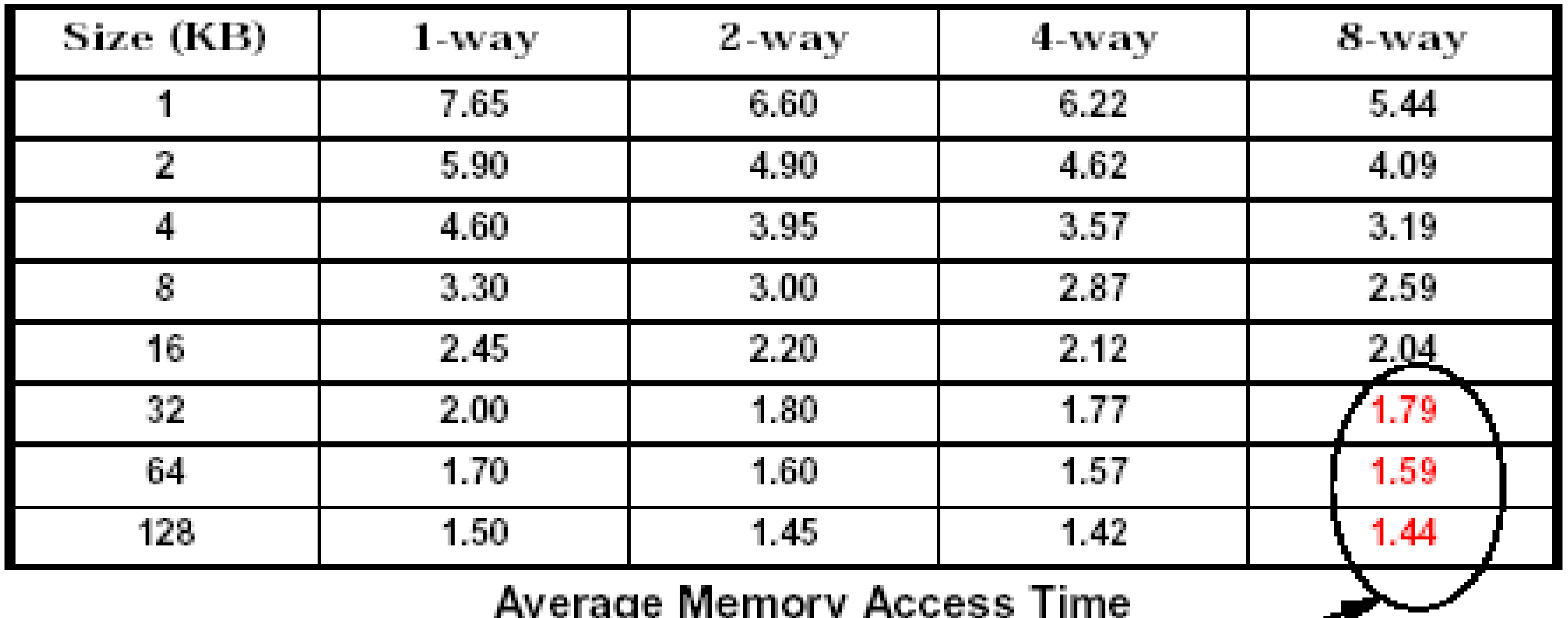
  B : Decreases hit time, decreases miss rate

  C : Increases hit time, increases miss rate

  D : Decreases hit time, increases miss rate

Average Memory Access Time

- Higher associativity can reduce cache miss rates
- Note that an 8-way set associative cache is essentially a fully-associative cache!
- But, Greater associativity can cause increased hit time



Size (KB)	1-way	2-way	4-way	8-way
1	7.65	6.60	6.22	5.44
2	5.90	4.90	4.62	4.09
4	4.60	3.95	3.57	3.19
8	3.30	3.00	2.87	2.59
16	2.45	2.20	2.12	2.04
32	2.00	1.80	1.77	1.79
64	1.70	1.60	1.57	1.59
128	1.50	1.45	1.42	1.44

Average Memory Access Time

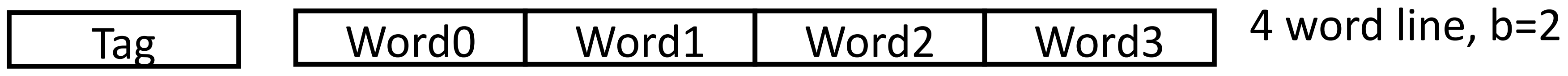
OOPS!

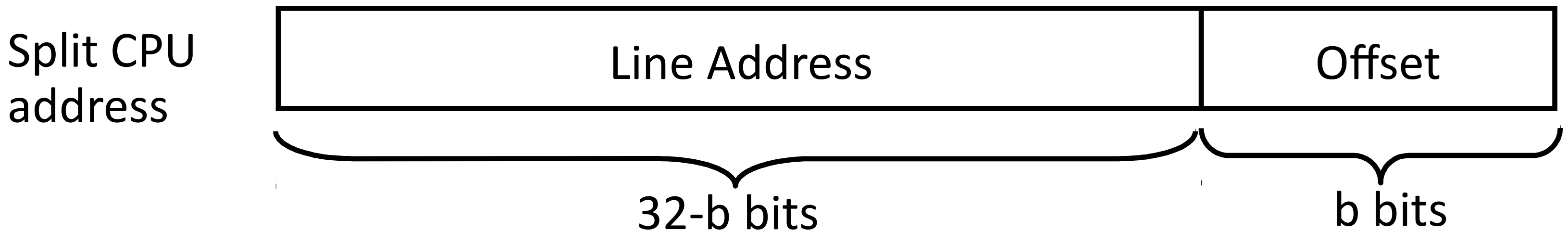
Increasing Block Size?

- Hit time as block size increases?
  - Hit time unchanged, but might be slight hit-time reduction as number of tags is reduced, so faster to access memory holding tags
- Miss rate as block size increases?
  - Goes down at first due to spatial locality, then increases due to increased conflict misses due to fewer blocks in cache
- Miss penalty as block size increases?
  - Rises with longer block size, but with fixed constant initial latency that is amortized over whole block

Recap: Line Size and Spatial Locality

A line is unit of transfer between the cache and memory





2^b = line size *a.k.a* line size (in bytes)

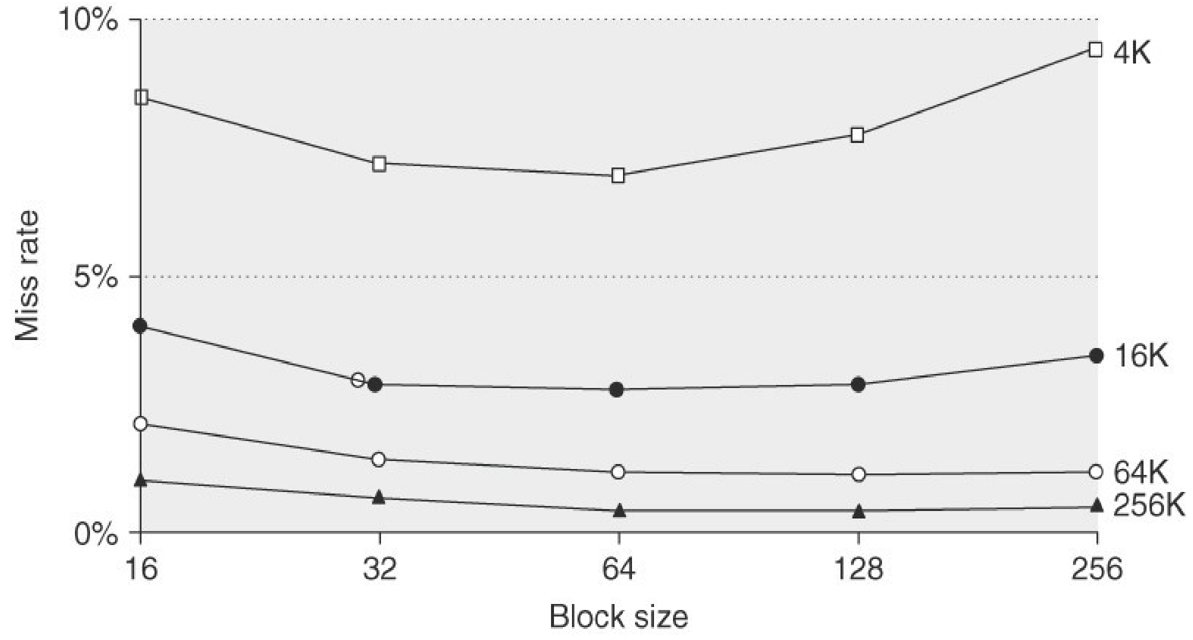
Larger line size has distinct hardware advantages

- less tag overhead
- exploit fast burst transfers from DRAM
- exploit fast burst transfers over wide busses

What are the disadvantages of increasing line size?

Fewer lines => more conflicts. Can waste bandwidth.

Larger Block Size (Miss Rate↓)



- Reduce compulsory misses -> Principle of spatial locality
- At the same time, may increase miss penalty, increase conflict misses

Peer Instruction

Impact of Larger Blocks on **AMAT**:

- For fixed total cache capacity and associativity, what is effect of larger blocks on each component of AMAT:

  A : Decrease

  B : Unchanged

  C : Increase

Shorter tags +, mux at edge -

Hit Time?	C: Unchanged (but slight increase possible)
Miss Rate?	A: Decrease (spatial locality; conflict???)
Miss Penalty?	C: Increase (longer time to load block)

Write Allocation? It depends!

Peer Instruction

Impact of Larger Blocks on **Misses**:

- For fixed total cache capacity and associativity, what is effect of larger blocks on each component of miss:

  A : Decrease
  B : Unchanged
  C : Increase

Compulsory?	A: Decrease (if good Spatial Locality)
Capacity?	C: Increase (smaller blocks fit better)
Conflict?	C: Increase (more ways better!)

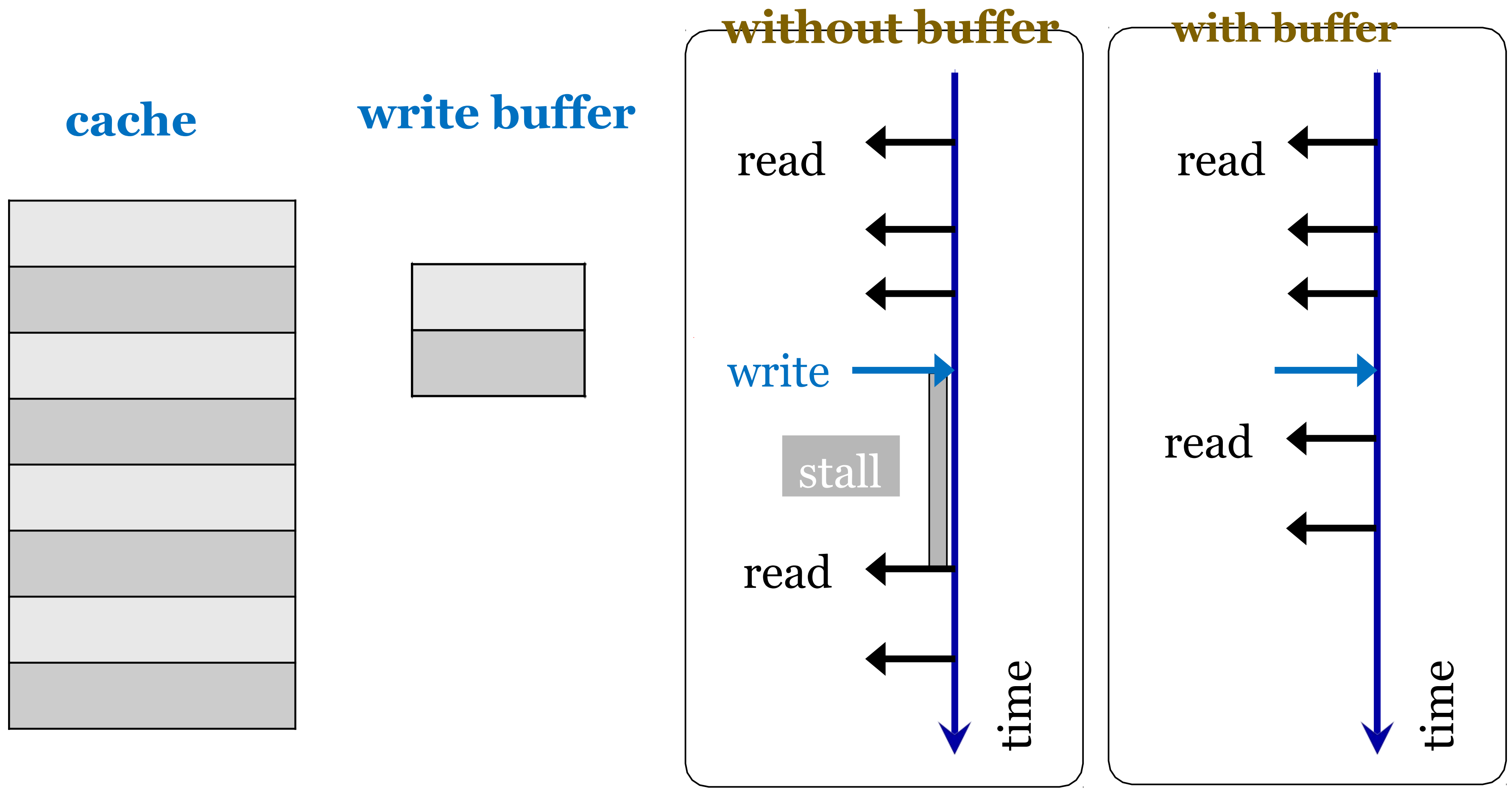
Less effect for large caches

Increasing #Entries?

- Hit time as #entries increases?
  - Increases, since reading tags and data from larger memory structures
- Miss rate as #entries increases?
  - Goes down due to reduced capacity and conflict misses
  - Architects rule of thumb: miss rate drops ~2x for every ~4x increase in capacity (only a gross approximation)
- Miss penalty as #entries increases?
  - Unchanged

At some point, increase in hit time for a larger cache may overcome the improvement in hit rate, yielding a decrease in performance

Giving Priority to Read misses (Penalty ↓)

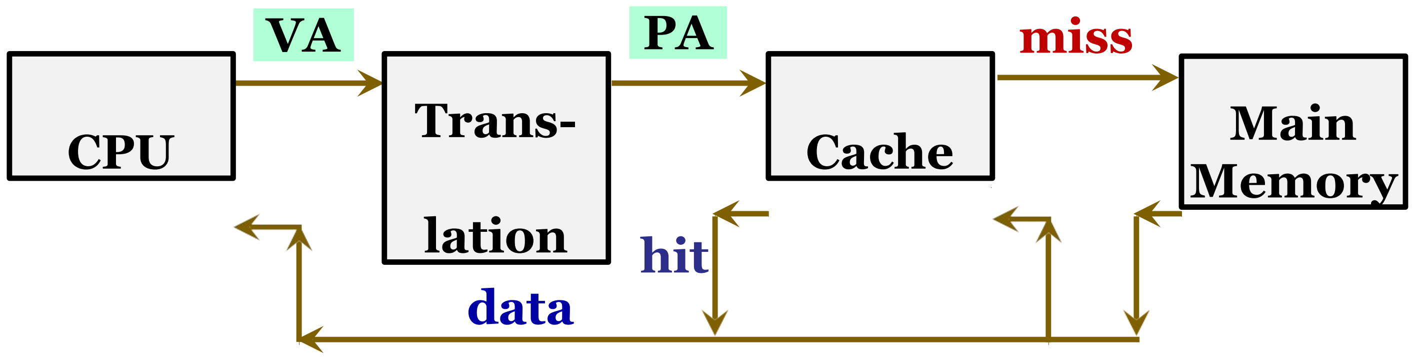


- To serve reads before writes have been completed!

Avoiding Address Translations (Hit Time↓)

- For each memory access, we need to check whether the data is in cache

- To do the checking, we need to do two tasks,
  - Locating the possible data in cache (by using the index)
  - Comparing the tags

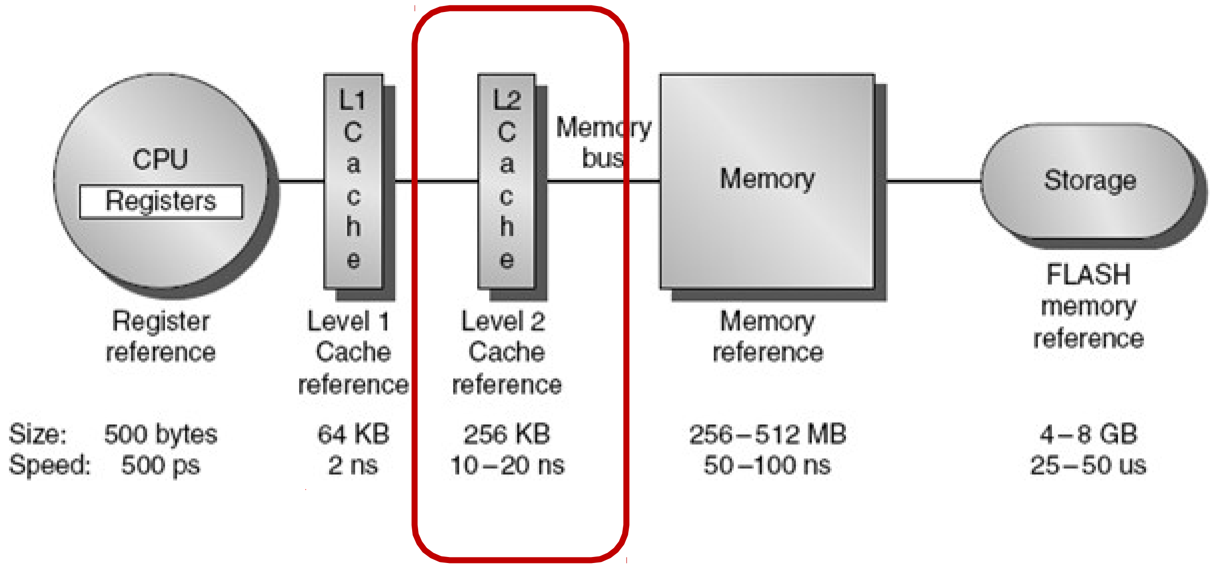


Cache with Virtual Address

How to Reduce Miss Penalty?

- Could there be locality on misses from a cache?
  - Use multiple cache levels!
  - With Moore's Law, more room on die for bigger L1$ and for second-level L2$
  - And in some cases even an L3$!

- Mainframes have ~1GB L4 cache off-chip



Local vs. Global Miss Rates

- *Local miss rate*: fraction of references to a given level of a cache that miss
  - Local Miss rate L2$ = L2$ Misses / L1$ Misses
    = L2$ Misses / total_L2_accesses
- *Global miss rate*: the fraction of references that miss in all levels of a multilevel cache and go all the way to memory
  - L2$ local miss rate >> than the global miss rate
- Global Miss rate = L2$ Misses / Total Accesses
  = (L2$ Misses / L1$ Misses) × (L1$ Misses / Total Accesses)
  = Local Miss rate L2$ × Local Miss rate L1$
- AMAT = Time for a hit + Miss rate × Miss penalty
- AMAT = Time for a L1$ hit + (local) Miss rate L1$ × (Time for a L2$ hit + (local) Miss rate L2$ × L2$ Miss penalty)

Multilevel Cache AMAT Example

AMAT = $T_{hit}(L1)$ + miss%(L1)×($T_{hit}(L2)$ + miss%(L2)×($T_{hit}(L3)$ + miss %(L3) ×T(memory)))

- Example:
  - ✓ miss rate L1=10%, $T_{hit}(L1)$ = 1 cycle
  - ✓ miss rate L2=5%, $T_{hit}(L2)$ = 10 cycles
  - ✓ miss rate L3=1%, $T_{hit}(L3)$ = 20 cycles
  - ✓ T(memory) = 300 cycles
- AMAT = ?
  - ✓ 2.115 (compare to 31 with no multi-levels)
  - ✓ 14.7× speed-up!

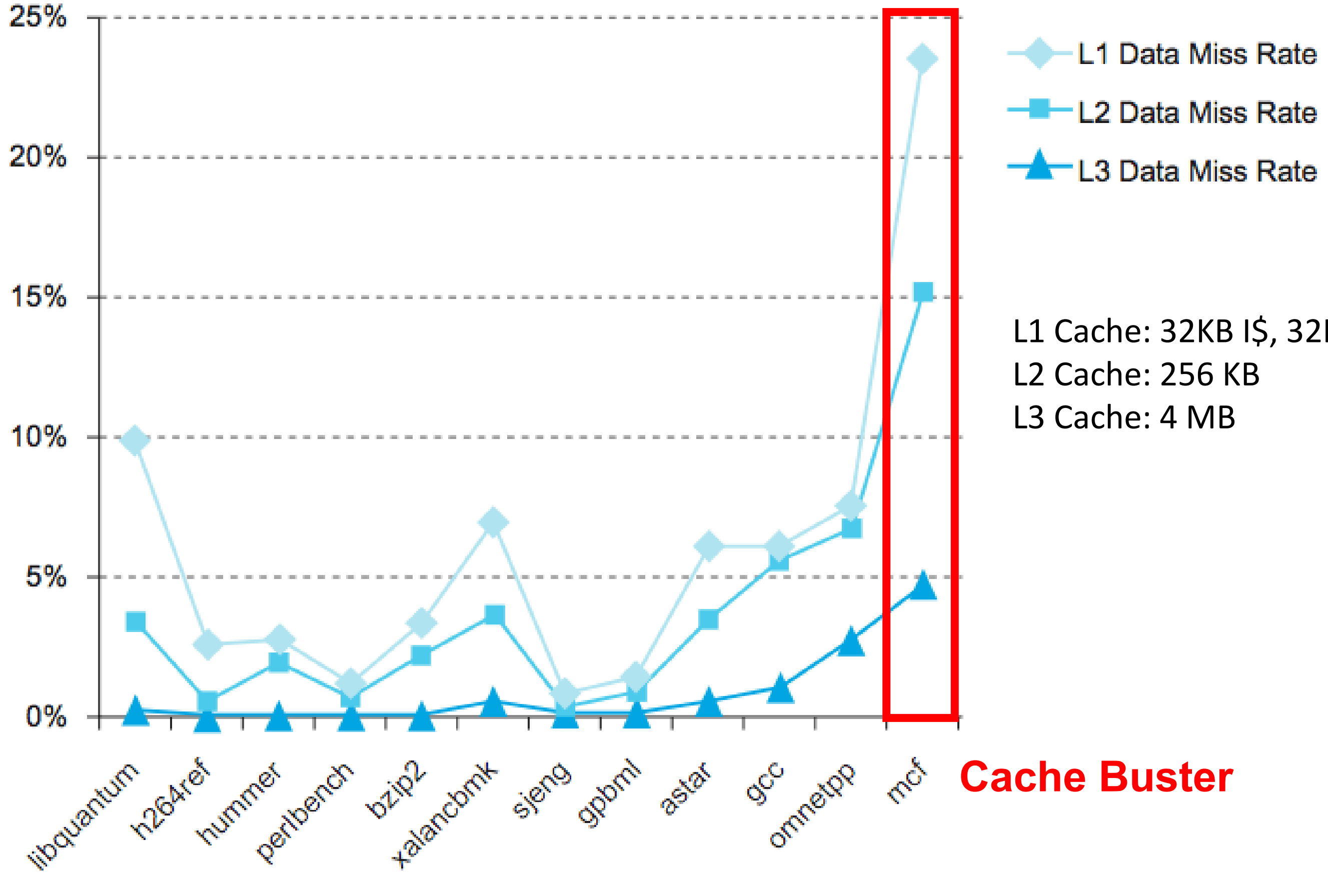
Multilevel Cache Considerations

- Different design considerations for L1$ and L2$
  - L1$ focuses on fast access: minimize hit time to achieve shorter clock cycle, e.g., smaller $
  - L2$, L3$ focus on low miss rate: reduce penalty of long main memory access times: e.g., Larger $ with larger block sizes/higher levels of associativity

- Miss penalty of L1$ is significantly reduced by presence of L2$, so can be smaller/faster even with higher miss rate

- For the L2$, fast hit time is less important than low miss rate
  - L2$ hit time determines L1$'s miss penalty
  - L2$ local miss rate >> than the global miss rate

Multilevel Cache Considerations

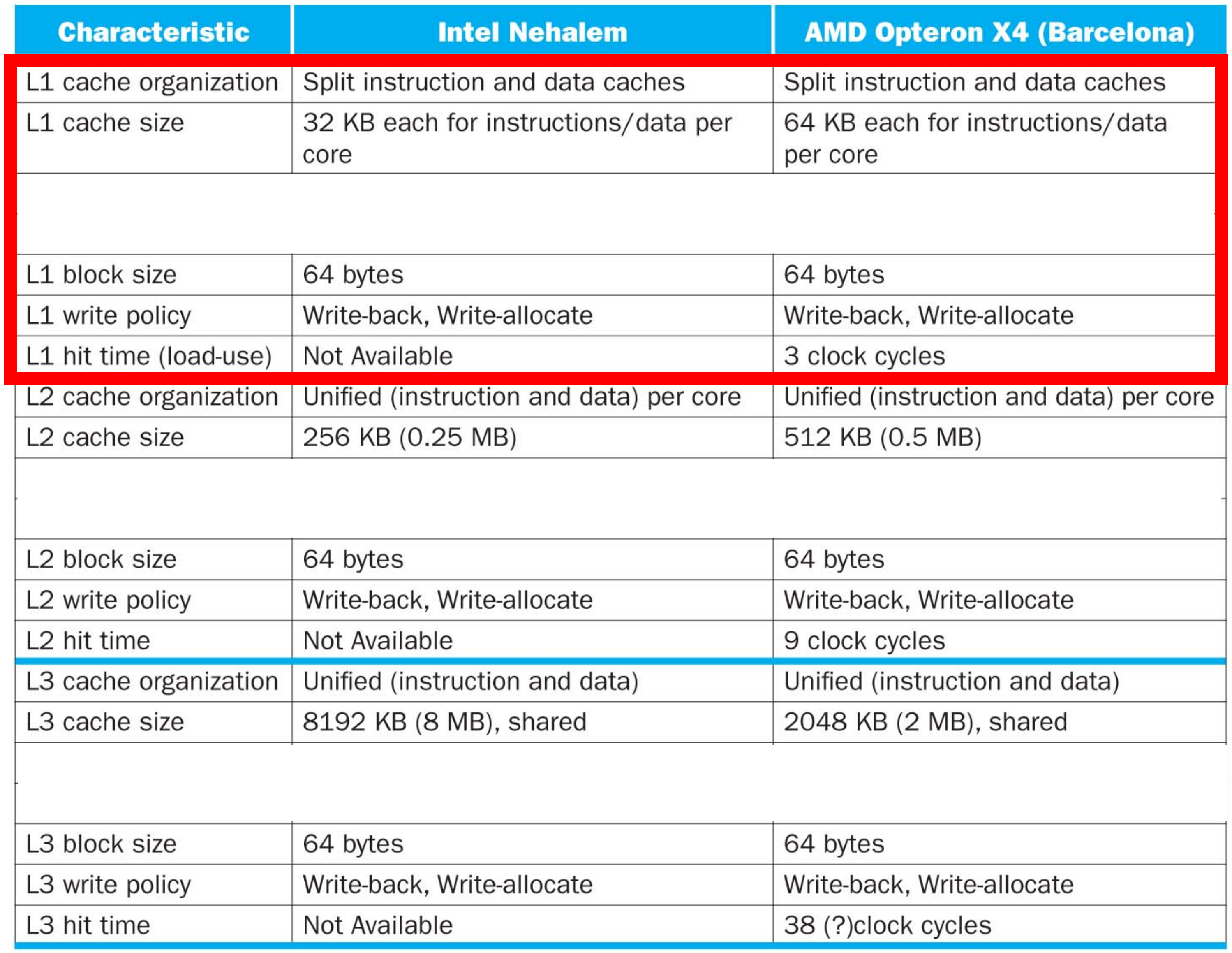


L1 Cache: 32KB I$, 32KB D$
L2 Cache: 256 KB
L3 Cache: 4 MB

Cache Buster

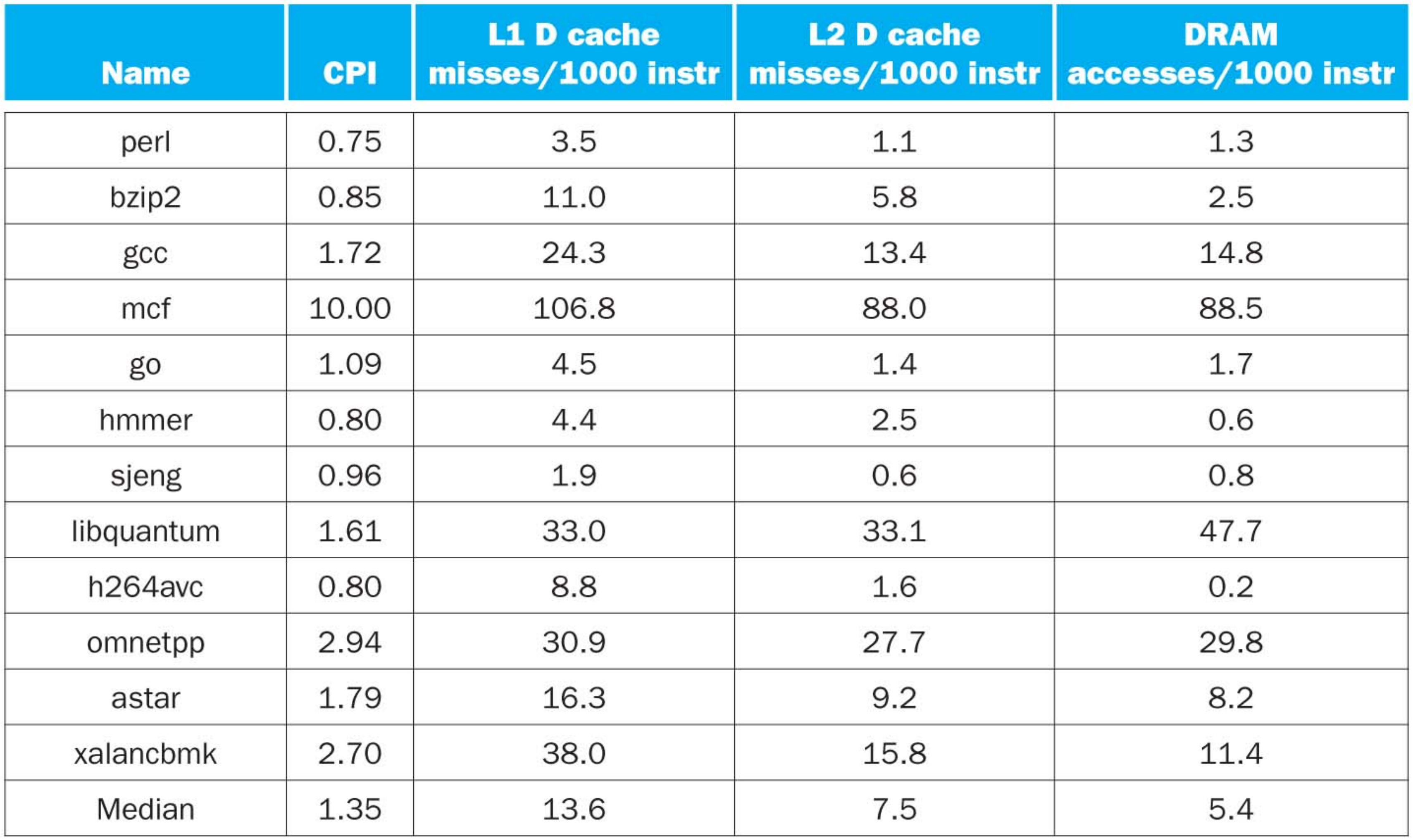
FIGURE 5.47 The L1, L2, and L3 data cache miss rates for the Intel Core i7 920 running the full integer SPECCPU2006 benchmarks.

Multilevel Cache Considerations



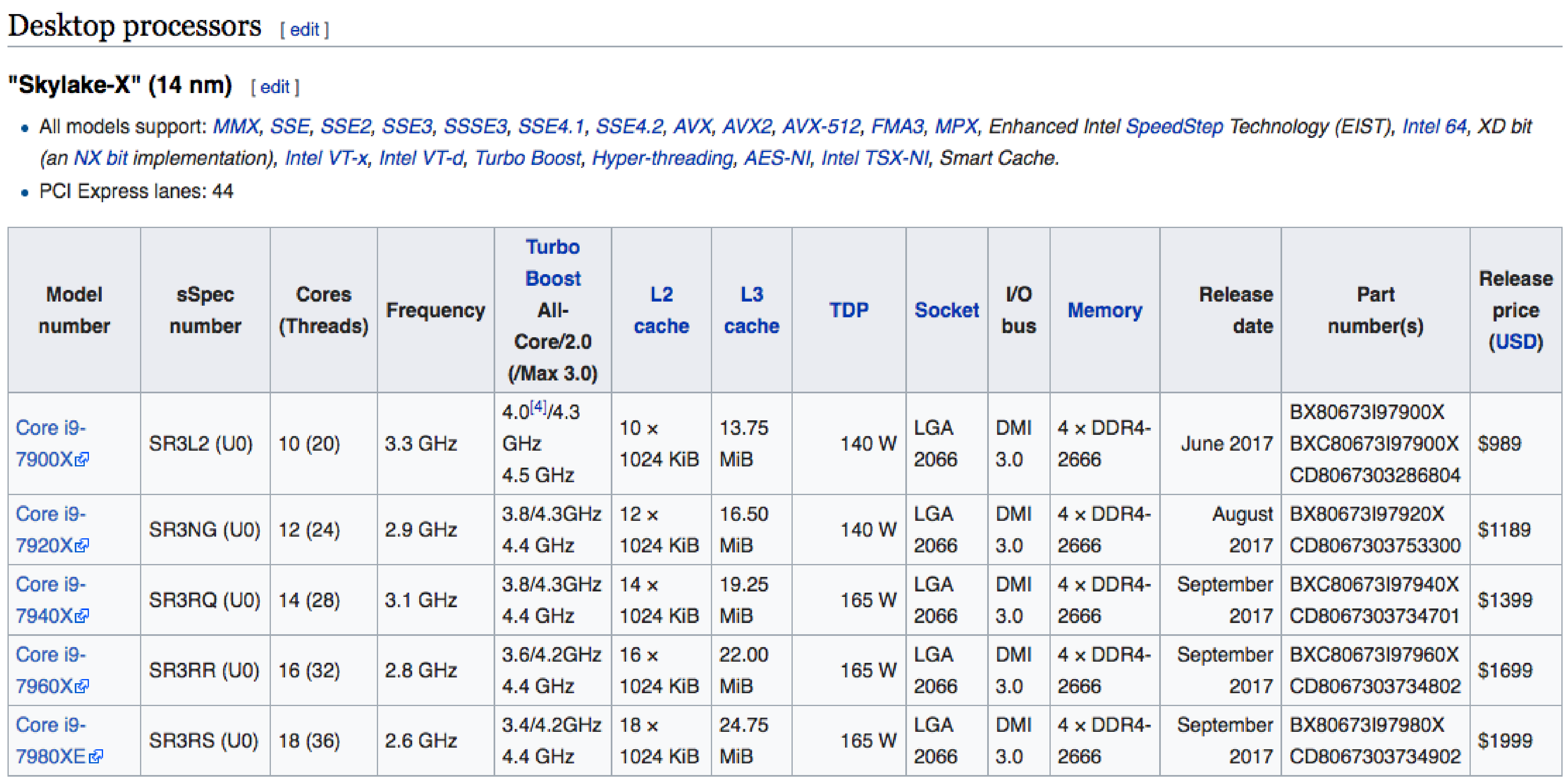
Characteristic	Intel Nehalem	AMD Opteron X4 (Barcelona)
L1 cache organization	Split instruction and data caches	Split instruction and data caches
L1 cache size	32 KB each for instructions/data per core	64 KB each for instructions/data per core
L1 block size	64 bytes	64 bytes
L1 write policy	Write-back, Write-allocate	Write-back, Write-allocate
L1 hit time (load-use)	Not Available	3 clock cycles
L2 cache organization	Unified (instruction and data) per core	Unified (instruction and data) per core
L2 cache size	256 KB (0.25 MB)	512 KB (0.5 MB)
L2 block size	64 bytes	64 bytes
L2 write policy	Write-back, Write-allocate	Write-back, Write-allocate
L2 hit time	Not Available	9 clock cycles
L3 cache organization	Unified (instruction and data)	Unified (instruction and data)
L3 cache size	8192 KB (8 MB), shared	2048 KB (2 MB), shared
L3 block size	64 bytes	64 bytes
L3 write policy	Write-back, Write-allocate	Write-back, Write-allocate
L3 hit time	Not Available	38 (?)clock cycles

CPI/Miss Rates/DRAM Access SpecInt2006



		Data Only	Data Only	Instructions and Data
Name	**CPI**	**L1 D cache misses/1000 instr**	**L2 D cache misses/1000 instr**	**DRAM accesses/1000 instr**
perl	0.75	3.5	1.1	1.3
bzip2	0.85	11.0	5.8	2.5
gcc	1.72	24.3	13.4	14.8
mcf	10.00	106.8	88.0	88.5
go	1.09	4.5	1.4	1.7
hmmer	0.80	4.4	2.5	0.6
sjeng	0.96	1.9	0.6	0.8
libquantum	1.61	33.0	33.1	47.7
h264avc	0.80	8.8	1.6	0.2
omnetpp	2.94	30.9	27.7	29.8
astar	1.79	16.3	9.2	8.2
xalancbmk	2.70	38.0	15.8	11.4
Median	1.35	13.6	7.5	5.4

Skylark: Intel's Recent Generation Laptop/Tablet Class CPUs



Desktop processors [edit]

"Skylake-X" (14 nm) [edit]

- All models support: *MMX, SSE, SSE2, SSE3, SSSE3, SSE4.1, SSE4.2, AVX, AVX2, AVX-512, FMA3, MPX, Enhanced Intel SpeedStep Technology (EIST), Intel 64, XD bit (an NX bit implementation), Intel VT-x, Intel VT-d, Turbo Boost, Hyper-threading, AES-NI, Intel TSX-NI, Smart Cache.*
- PCI Express lanes: 44

Model number	sSpec number	Cores (Threads)	Frequency	Turbo Boost All-Core/2.0 (/Max 3.0)	L2 cache	L3 cache	TDP	Socket	I/O bus	Memory	Release date	Part number(s)	Release price (USD)
Core i9-7900X	SR3L2 (U0)	10 (20)	3.3 GHz	4.0[4]/4.3 GHz 4.5 GHz	10 × 1024 KiB	13.75 MiB	140 W	LGA 2066	DMI 3.0	4 × DDR4-2666	June 2017	BX80673I97900X BXC80673I97900X CD8067303286804	$989
Core i9-7920X	SR3NG (U0)	12 (24)	2.9 GHz	3.8/4.3GHz 4.4 GHz	12 × 1024 KiB	16.50 MiB	140 W	LGA 2066	DMI 3.0	4 × DDR4-2666	August 2017	BX80673I97920X CD8067303753300	$1189
Core i9-7940X	SR3RQ (U0)	14 (28)	3.1 GHz	3.8/4.3GHz 4.4 GHz	14 × 1024 KiB	19.25 MiB	165 W	LGA 2066	DMI 3.0	4 × DDR4-2666	September 2017	BXC80673I97940X CD8067303734701	$1399
Core i9-7960X	SR3RR (U0)	16 (32)	2.8 GHz	3.6/4.2GHz 4.4 GHz	16 × 1024 KiB	22.00 MiB	165 W	LGA 2066	DMI 3.0	4 × DDR4-2666	September 2017	BXC80673I97960X CD8067303734802	$1699
Core i9-7980XE	SR3RS (U0)	18 (36)	2.6 GHz	3.4/4.2GHz 4.4 GHz	18 × 1024 KiB	24.75 MiB	165 W	LGA 2066	DMI 3.0	4 × DDR4-2666	September 2017	BX80673I97980X CD8067303734902	$1999