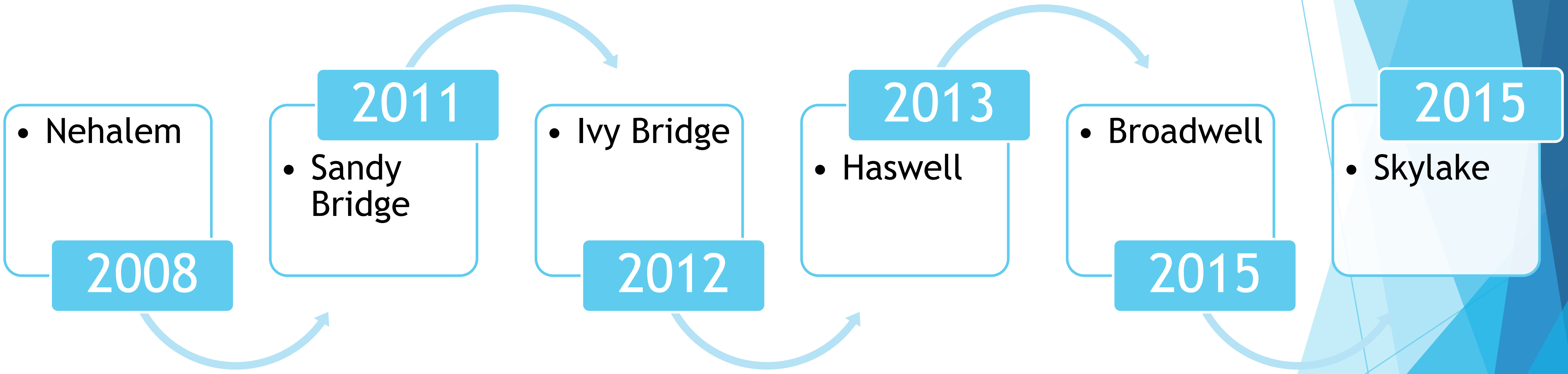
Intel Core i7 Memory Hierarchy

Amanda Adkins, Brett Ammeson, James Anouna,
Tony Garside, Lukas Hunker, Sam Mailand

Intel i7 Timeline



2008
- Nehalem

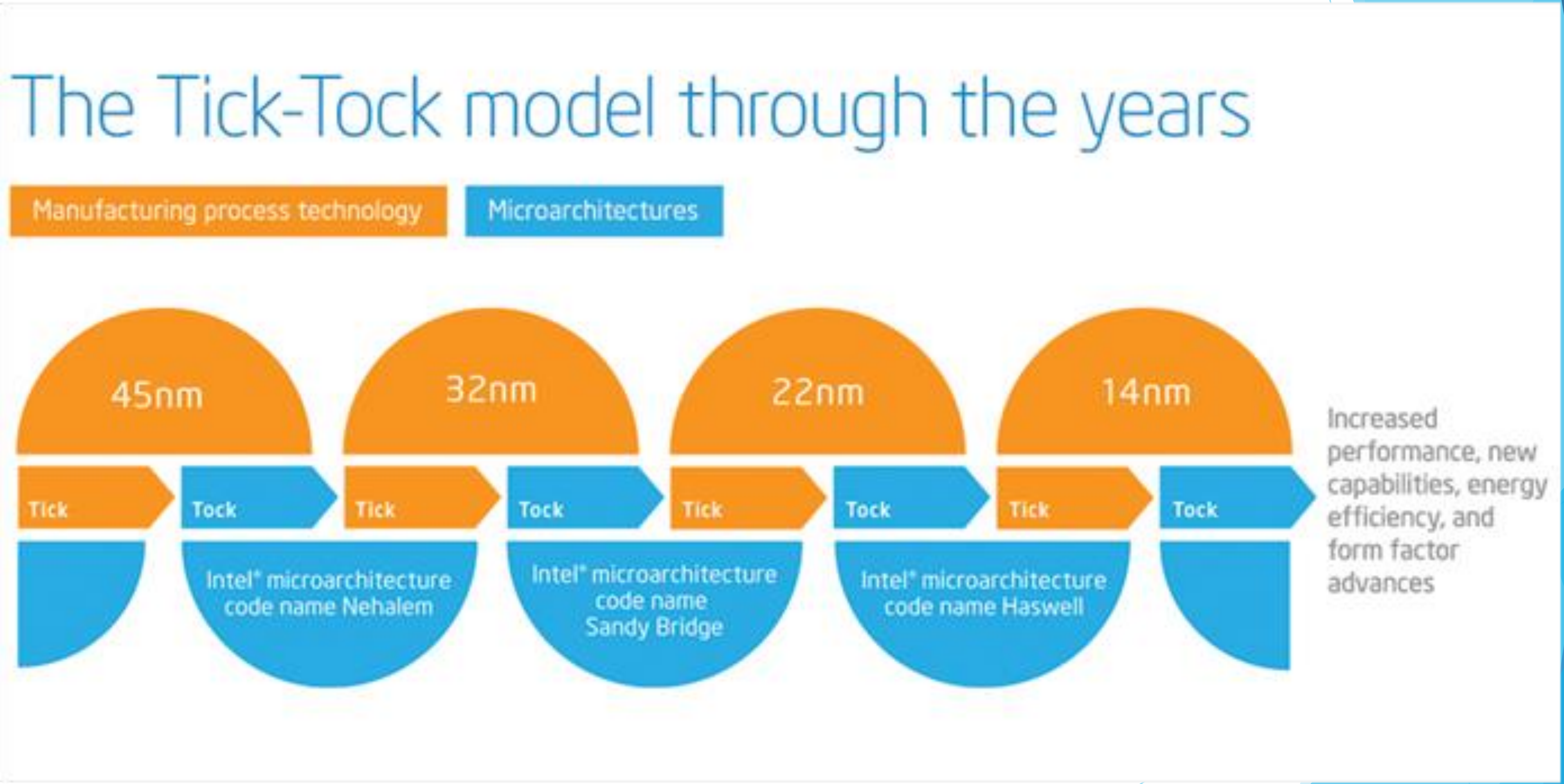
2011
- Sandy Bridge

2012
- Ivy Bridge

2013
- Haswell

2015
- Broadwell

2015
- Skylake



The Tick-Tock model through the years

Manufacturing process technology

Microarchitectures

45nm

32nm

22nm

14nm

Tick

Tock

Tick

Tock

Tick

Tock

Tick

Tock

Intel* microarchitecture code name Nehalem

Intel* microarchitecture code name Sandy Bridge

Intel* microarchitecture code name Haswell

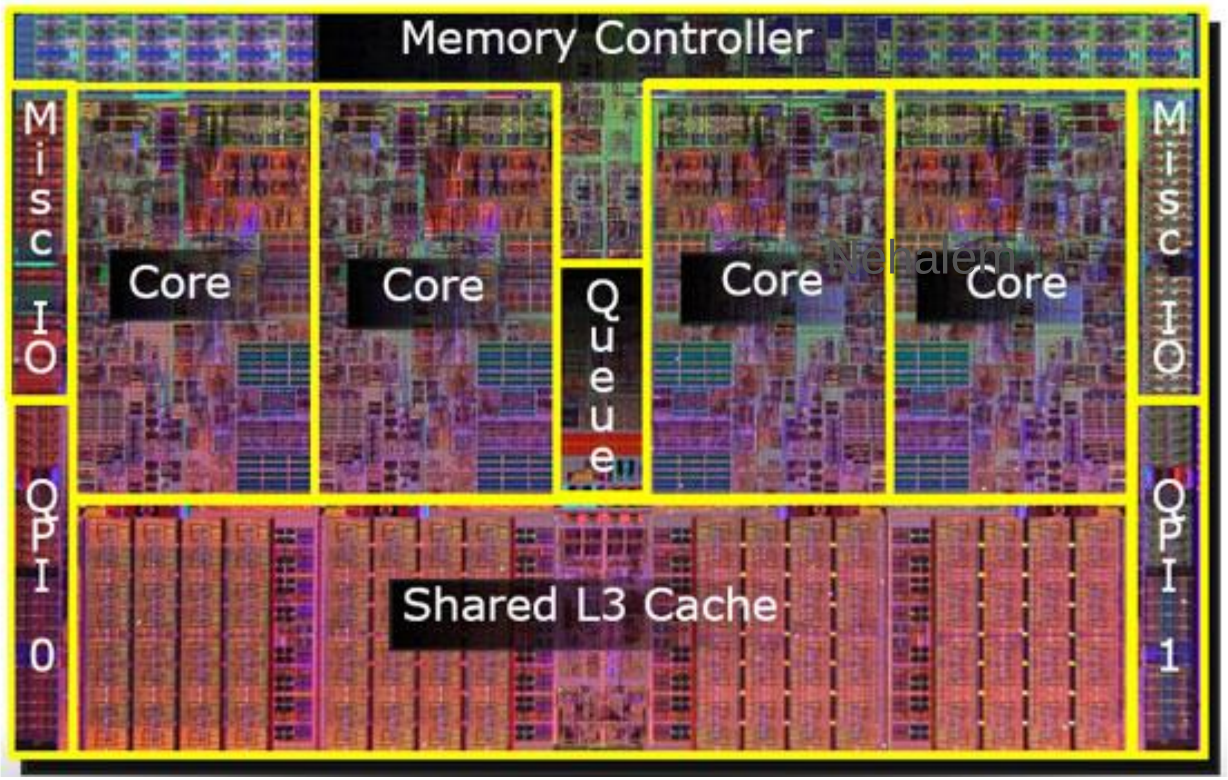
Increased performance, new capabilities, energy efficiency, and form factor advances

Core i7 Basic Structure

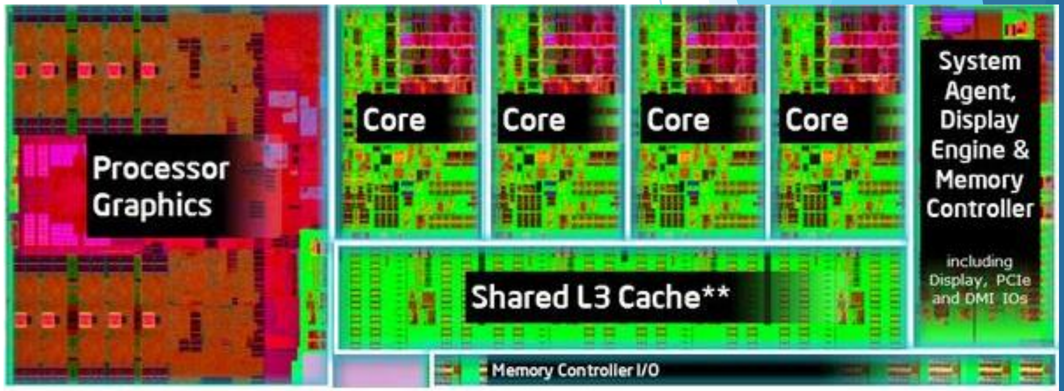
- 4 cores
- Hyper threaded – 8 threads
- Pipelined with 16 stages

DESKTOP
INTEL® CORE™ i7 PROCESSOR
THE ULTIMATE IN VISIBLY SMART PERFORMANCE
intel® inside™
CORE™ i7
•4 Cores / 8-Way Multitask Processing
•Intel® Turbo Boost Technology 2.0
•Intel® Hyper-Threading Technology
•Intel® Smart Cache Technology
•Integrated Memory Controller
•Supports Two-Channel DDR3 Memory
•Three-Year Limited Warranty
Intel® HD Graphics 4000
For information about Intel's system integrator program, visit www.intel.com/reseller
DESKTOP
INTEL® CORE™ i7 PROCESSOR
THE ULTIMATE IN VISIBLY SMART PERFORMANCE

Footprint



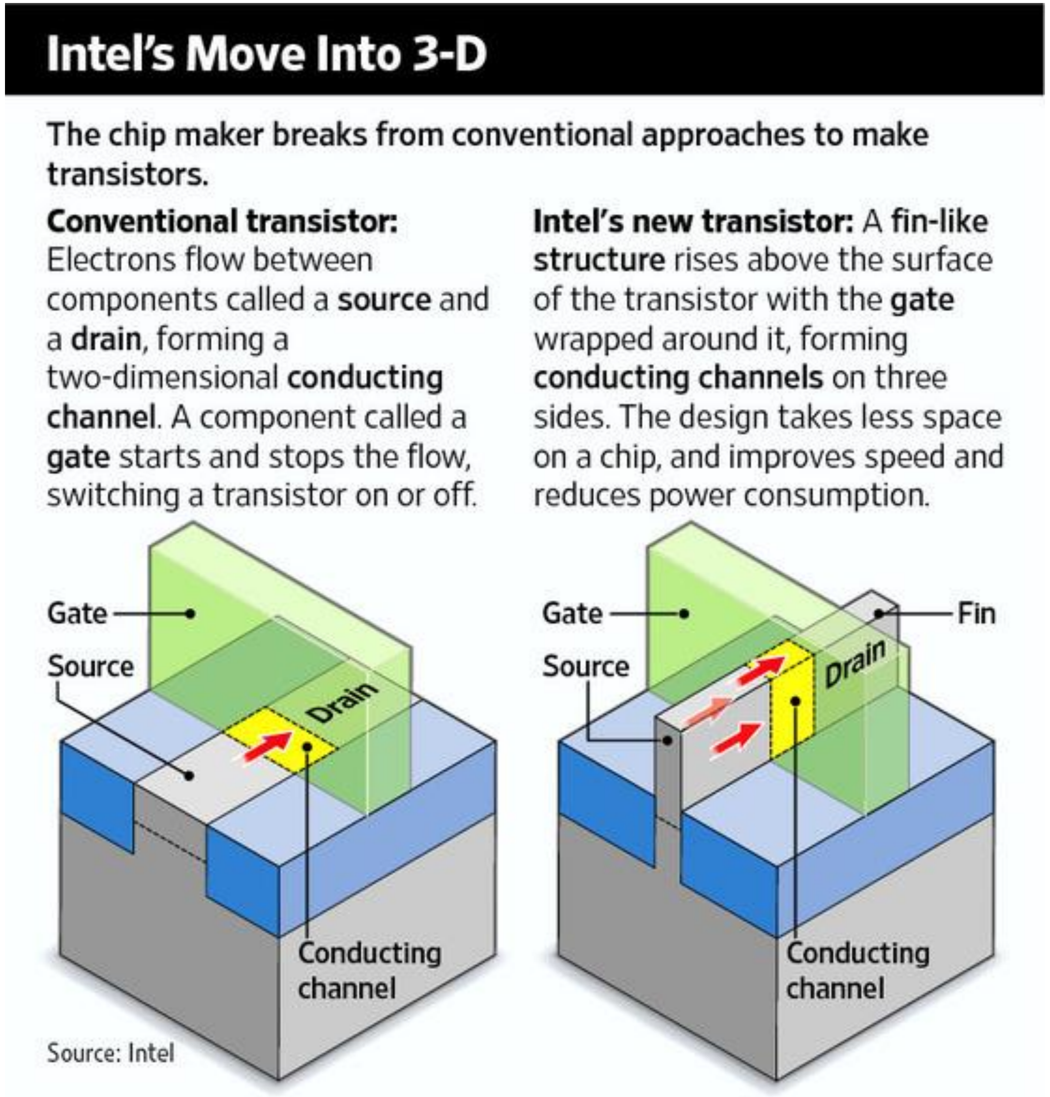
Nehalem (First Gen)



Processor Graphics
Core
Core
Core
Core
System Agent, Display Engine & Memory Controller
including Display, PCIe and DMI IOs
Shared L3 Cache**
Memory Controller I/O

Haswell (Fourth Gen)

Major Developments



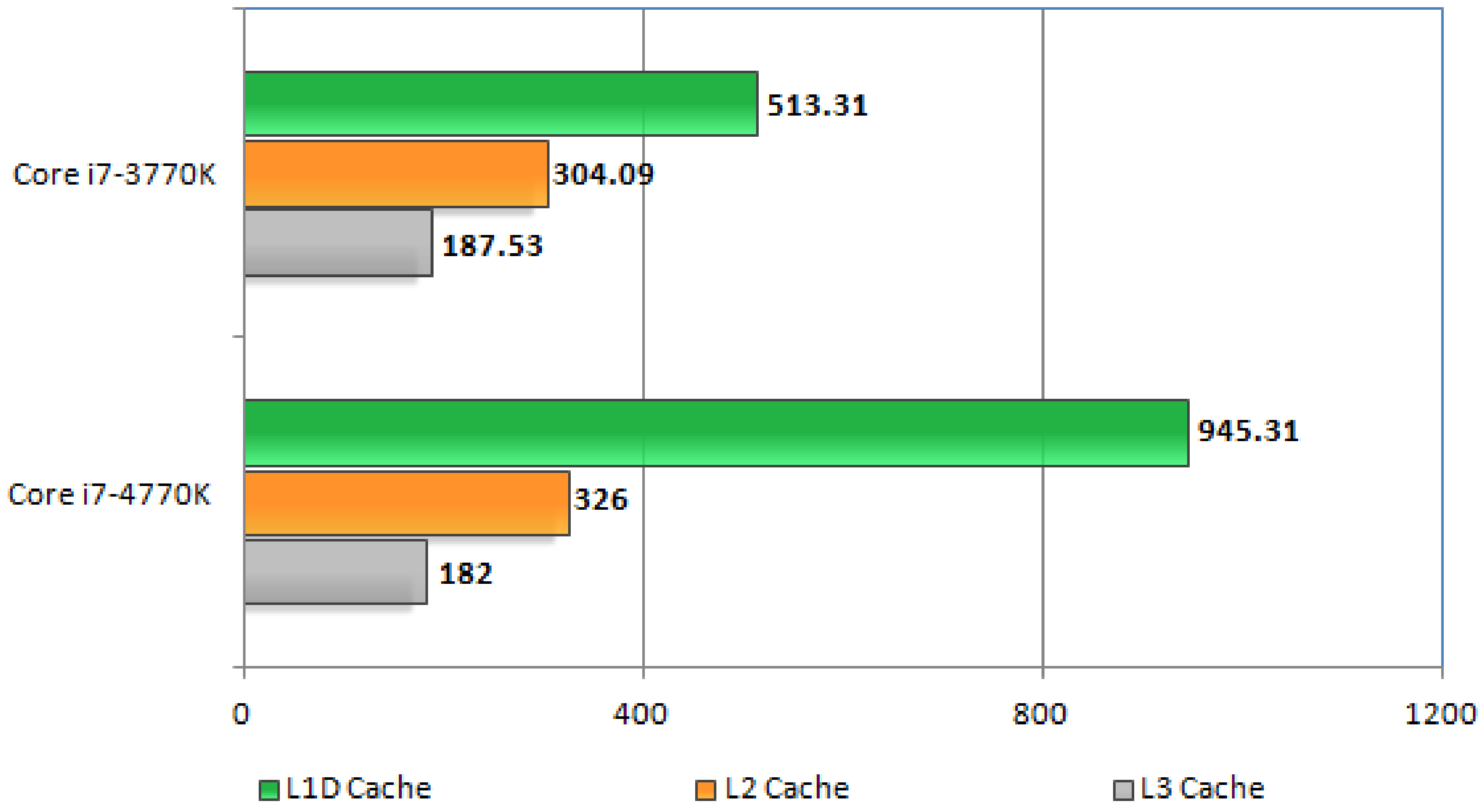
Intel's Move Into 3-D

The chip maker breaks from conventional approaches to make transistors.

Conventional transistor: Electrons flow between components called a **source** and a **drain**, forming a two-dimensional **conducting channel**. A component called a **gate** starts and stops the flow, switching a transistor on or off.

Intel's new transistor: A **fin-like structure** rises above the surface of the transistor with the **gate** wrapped around it, forming **conducting channels** on three sides. The design takes less space on a chip, and improves speed and reduces power consumption.

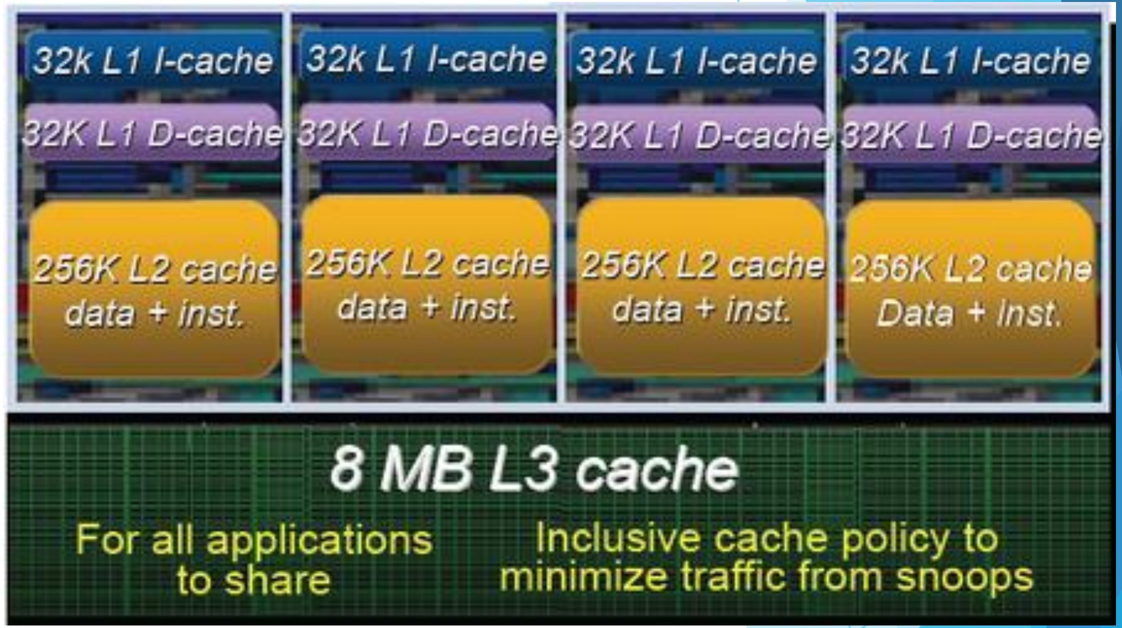
Source: Intel



Increased Cache Bandwidth

Intel Core i7 Caching Basics

- Intel core i7 processors feature three levels of caching.
  - Separate L1 and L2 cache for each core.
  - L1 cache broken up into to halves, instruction/data.
  - L3 cache shared among all cores and is inclusive.


32k L1 I-cache
32k L1 I-cache
32k L1 I-cache
32k L1 I-cache
32K L1 D-cache
32K L1 D-cache
32K L1 D-cache
32K L1 D-cache
256K L2 cache data + inst.
256K L2 cache data + inst.
256K L2 cache data + inst.
256K L2 cache Data + Inst.
8 MB L3 cache
For all applications to share
Inclusive cache policy to minimize traffic from snoops

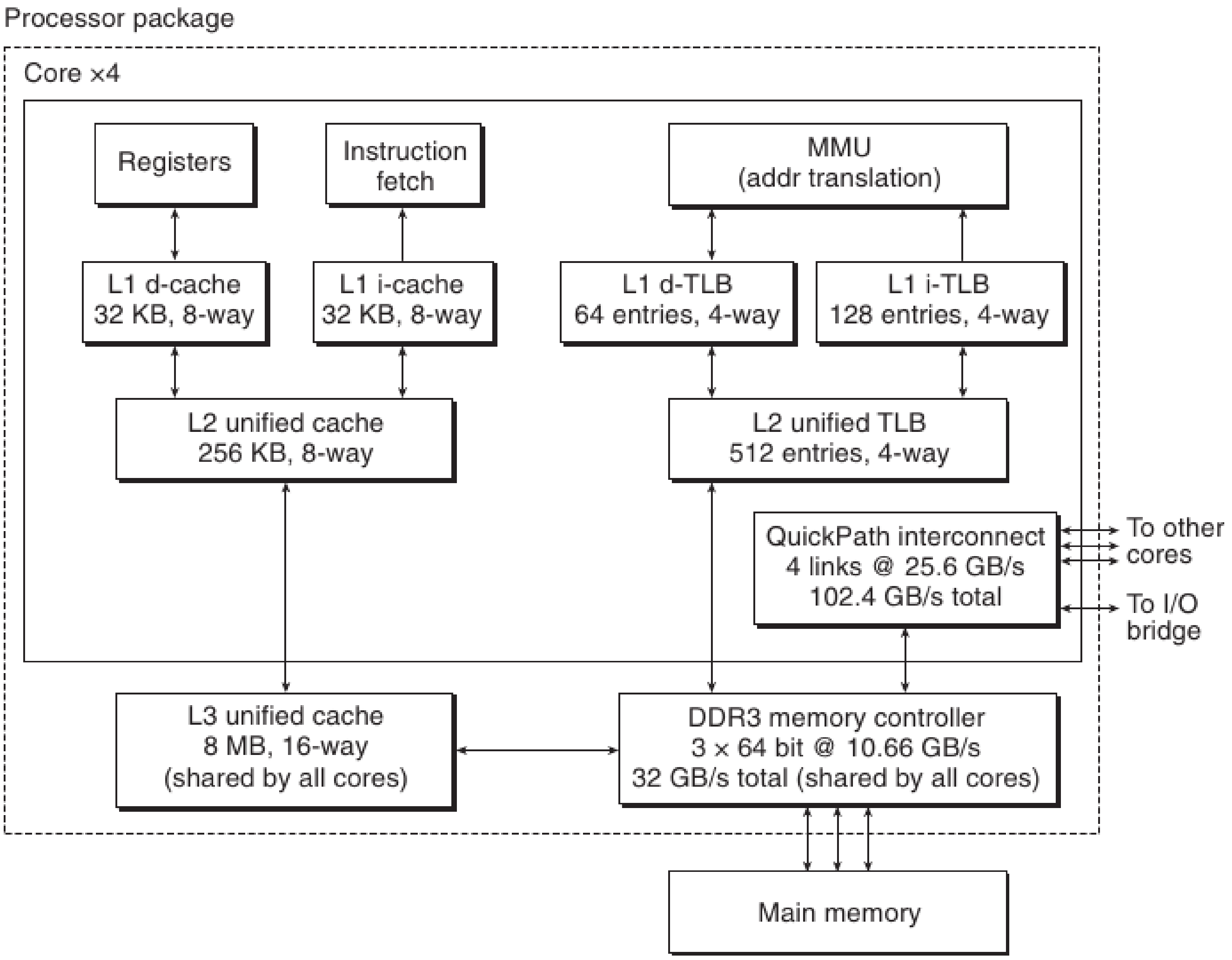
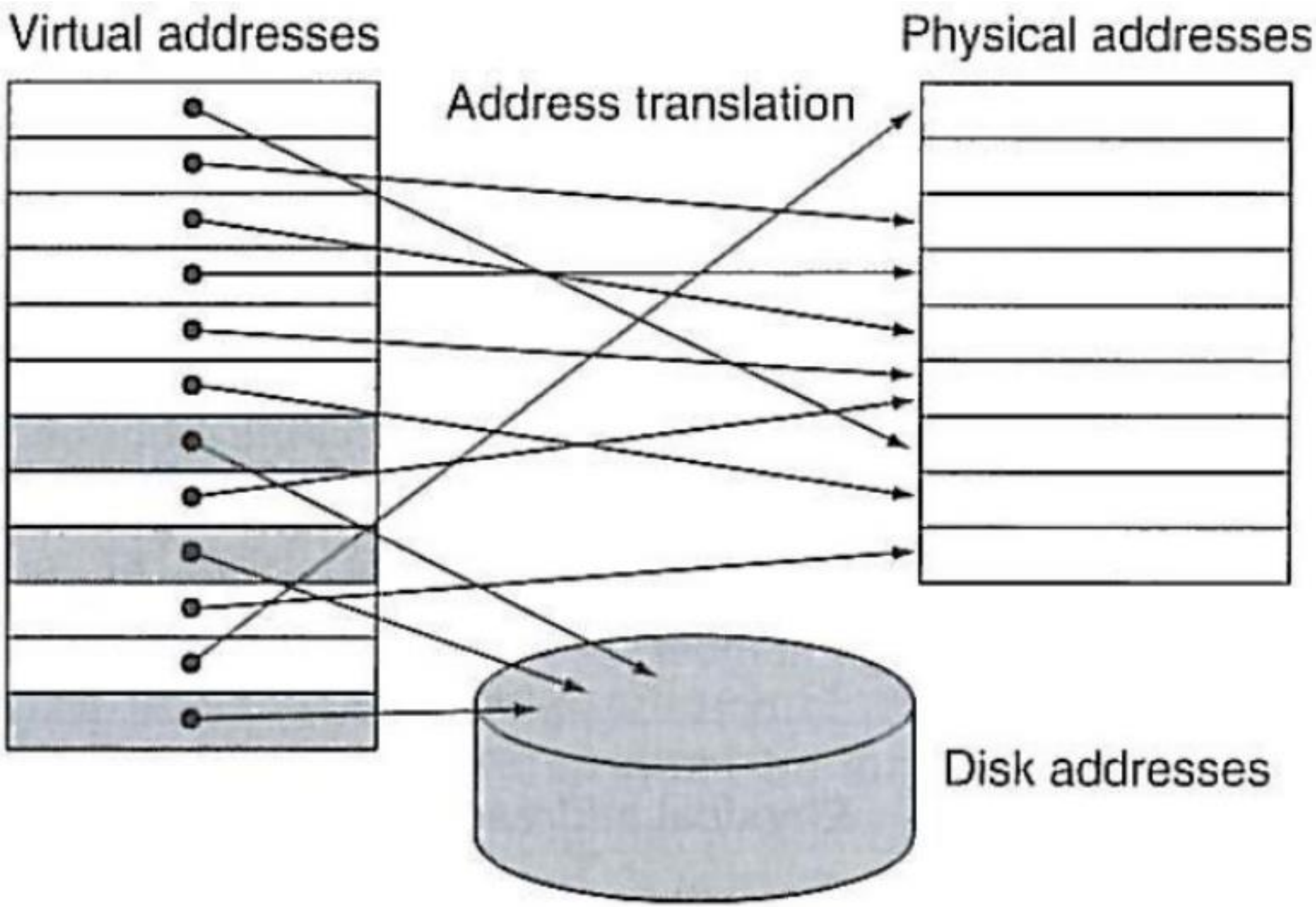
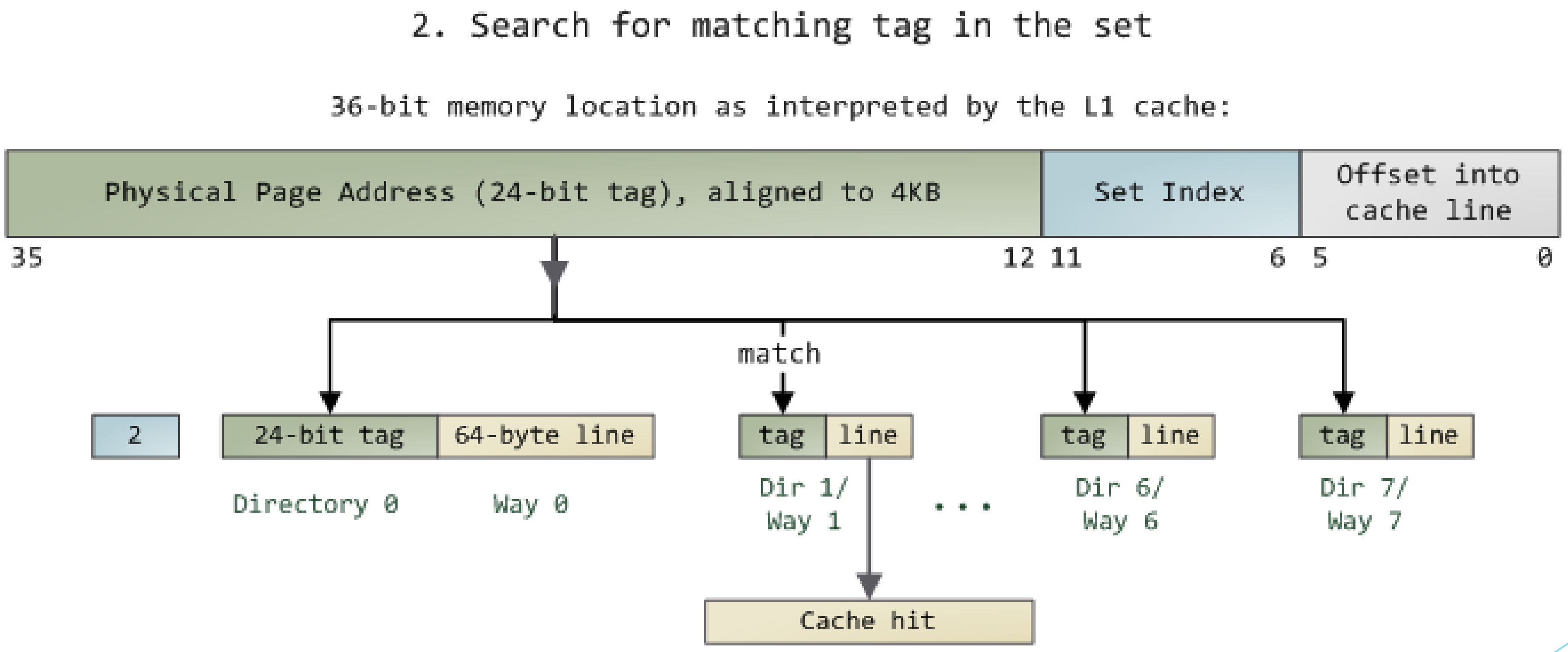

Processor package
Core ×4
Registers
Instruction fetch
MMU (addr translation)
L1 d-cache 32 KB, 8-way
L1 i-cache 32 KB, 8-way
L1 d-TLB 64 entries, 4-way
L1 i-TLB 128 entries, 4-way
L2 unified cache 256 KB, 8-way
L2 unified TLB 512 entries, 4-way
QuickPath interconnect 4 links @ 25.6 GB/s 102.4 GB/s total
To other cores
To I/O bridge
L3 unified cache 8 MB, 16-way (shared by all cores)
DDR3 memory controller 3 × 64 bit @ 10.66 GB/s 32 GB/s total (shared by all cores)
Main memory

Figure 9.21 **The Core i7 memory system.**

Virtual Addressing



Physical Addressing



N-way set associativity (Review)

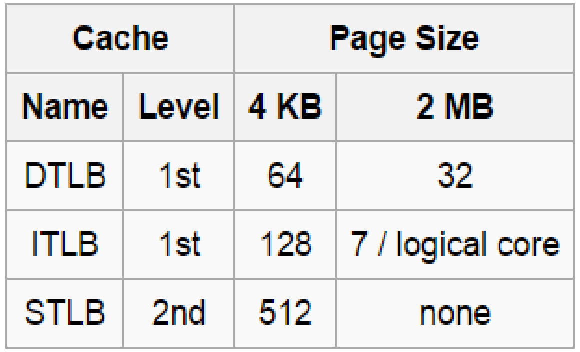
- Multiple entries per index
- Narrows search area needed to find unused slot
- i7 4790
  - L1 4x32 KB 8-way
  - L2 4/256 KB 8-way
  - L3 shared 8 MB 16-way

Intel's core i7 TLB design

- Memory cache that stores recent translations of virtual memory to physical addresses for faster retrieval.
- Uses a 2 level cache system
- L1 TLB
  - Divided into 2 parts
  - Data TLB: 64 4KB entries
  - Instruction TLB: 128 4KB entries
- L2 TLB (Services misses in L1 DTLB)
  - Can hold translations for 4KB and 2 MB pages (vs. only 4KB)
  - 1024 entries (vs. 512)
  - 8-way associative (vs. 4-way)

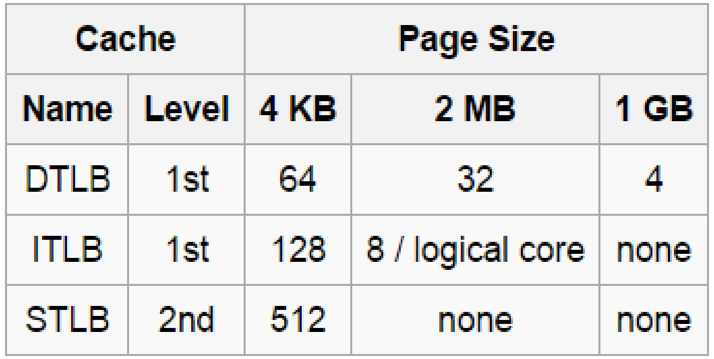
TLB Comparisons between generations

Nehalem



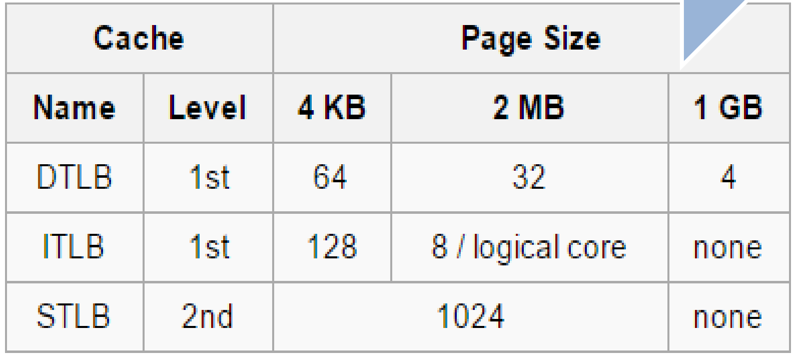
Cache		Page Size	
Name	Level	4 KB	2 MB
DTLB	1st	64	32
ITLB	1st	128	7 / logical core
STLB	2nd	512	none

Sandy Bridge and Ivy Bridge



Cache		Page Size		
Name	Level	4 KB	2 MB	1 GB
DTLB	1st	64	32	4
ITLB	1st	128	8 / logical core	none
STLB	2nd	512	none	none

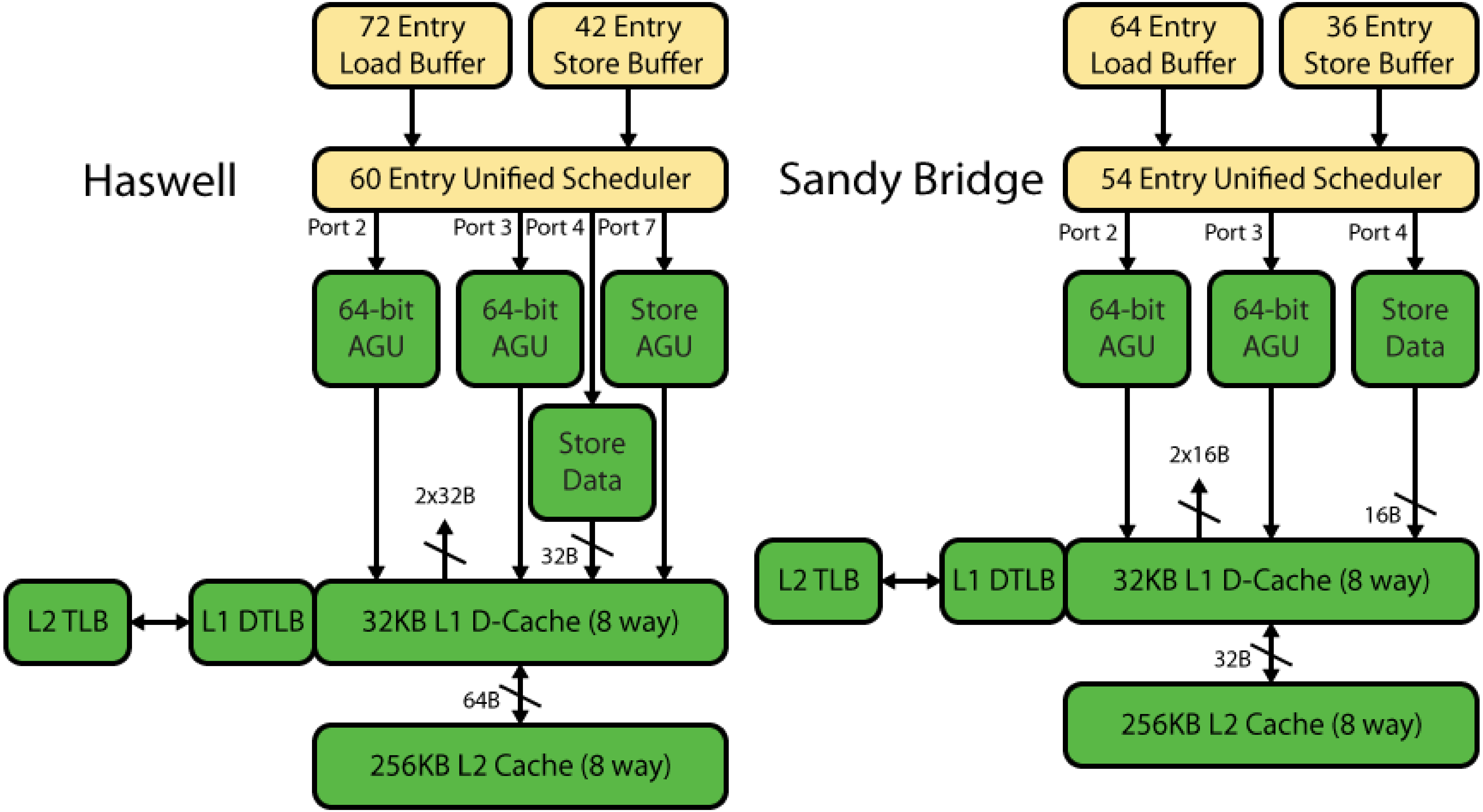
Haswell



Cache		Page Size		
Name	Level	4 KB	2 MB	1 GB
DTLB	1st	64	32	4
ITLB	1st	128	8 / logical core	none
STLB	2nd	1024		none

Pseudo-LRU (Intel's core i7 caching algorithm)

- One bit per cache line
- Resets after all lines' bit is set
- Lowest line index with a '0' replaced



Haswell

72 Entry Load Buffer

42 Entry Store Buffer

60 Entry Unified Scheduler

Port 2

Port 3

Port 4

Port 7

64-bit AGU

64-bit AGU

Store AGU

Store Data

2x32B

32B

L2 TLB

L1 DTLB

32KB L1 D-Cache (8 way)

64B

256KB L2 Cache (8 way)

Sandy Bridge

64 Entry Load Buffer

36 Entry Store Buffer

54 Entry Unified Scheduler

Port 2

Port 3

Port 4

64-bit AGU

64-bit AGU

Store Data

2x16B

16B

L2 TLB

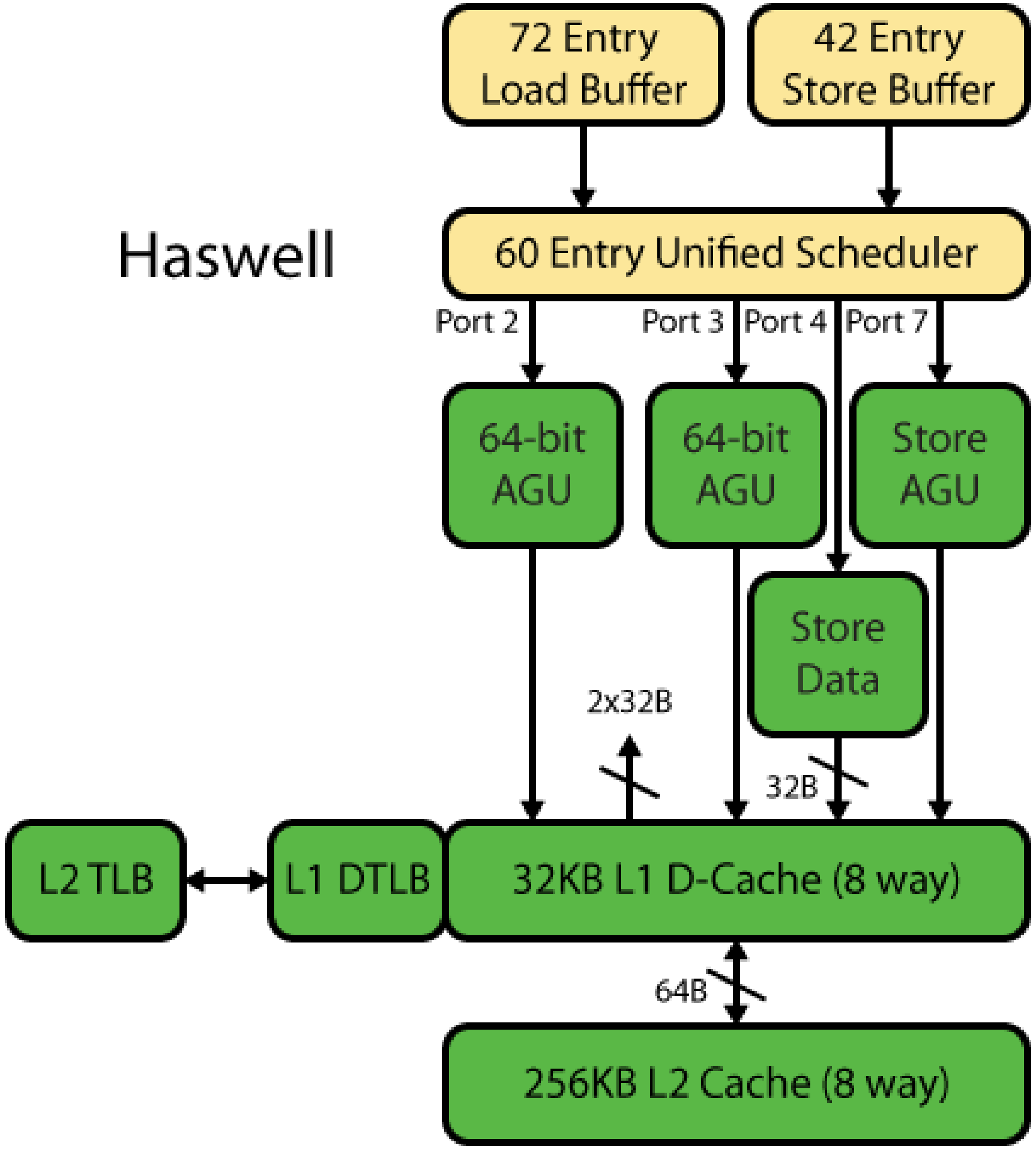
L1 DTLB

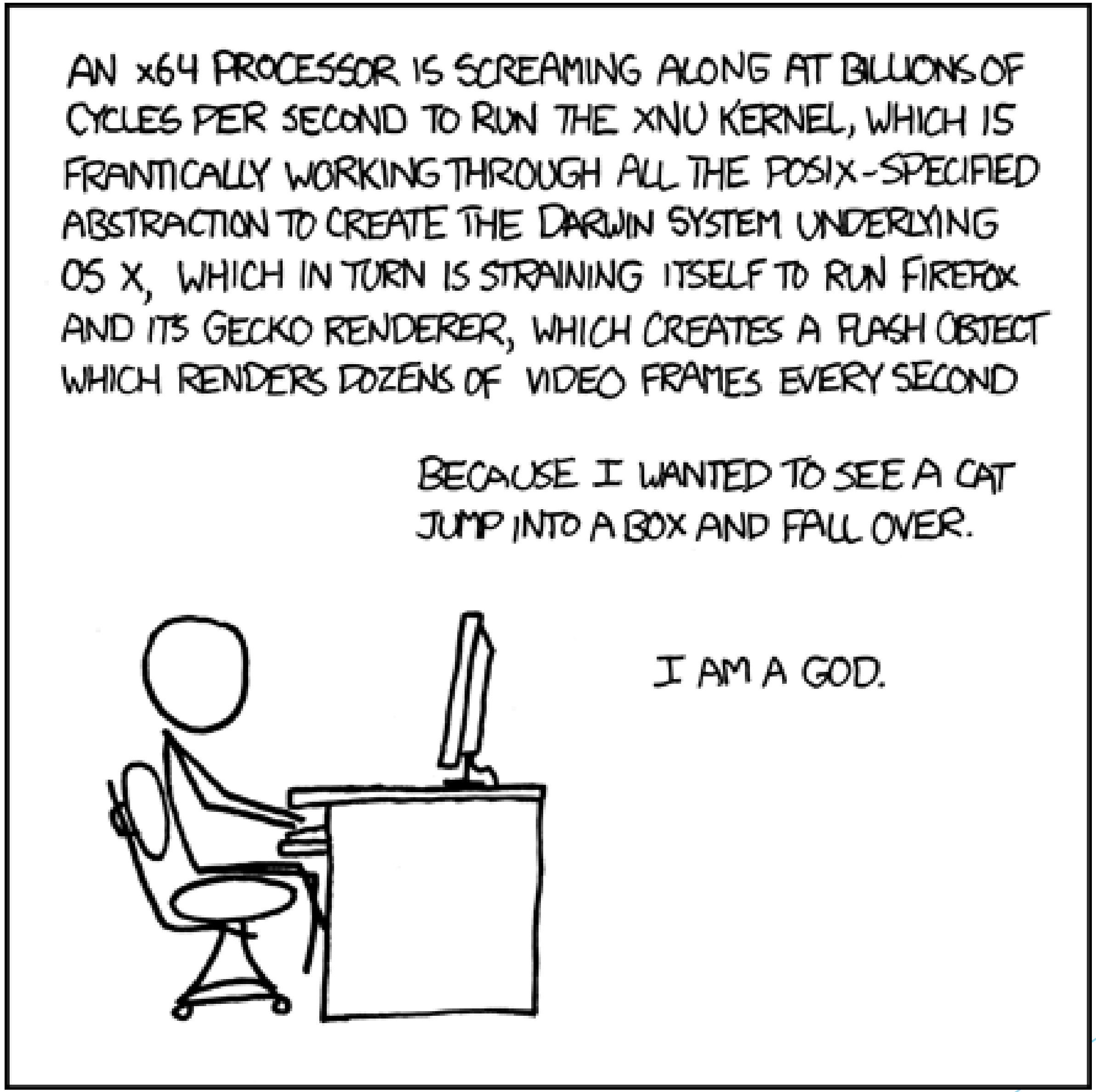
32KB L1 D-Cache (8 way)

32B

256KB L2 Cache (8 way)

- Port 2 and 3 are the Address Generation Units
- Port 4 for writing data from the core to the L1 Cache
- Additional port added to Haswell
- Haswell can sustain 2 loads and 1 store per cycle "under nearly any circumstances"
- Forwarding latency for AVX loads decreased from 2 to 1 cycle
- AVX: Set of instructions for doing SIMD operations on Intel CPUs
- 4 Split line buffers to resolve unaligned loads (vs 2 in Sandy-bridge)
- Decrease impact of unaligned access




AN x64 PROCESSOR IS SCREAMING ALONG AT BILLIONS OF CYCLES PER SECOND TO RUN THE XNU KERNEL, WHICH IS FRANTICALLY WORKING THROUGH ALL THE POSIX-SPECIFIED ABSTRACTION TO CREATE THE DARWIN SYSTEM UNDERLYING OS X, WHICH IN TURN IS STRAINING ITSELF TO RUN FIREFOX AND ITS GECKO RENDERER, WHICH CREATES A FLASH OBJECT WHICH RENDERS DOZENS OF VIDEO FRAMES EVERY SECOND
BECAUSE I WANTED TO SEE A CAT JUMP INTO A BOX AND FALL OVER.
I AM A GOD.

Haswell L1 Cache

- 32 kb
- 8 way associative
- Writeback
- TLB access & cache tag can occur in parallel
- Does not suffer from bank conflicts (unlike Sandy Bridge)
- Minimum latency: 4 cycles (same as Sandy-Bridge)
- Minimum lock latency of haswell is 12 cycles (sandy-bridge was 16)

Haswell L2 Cache

- Bandwidth doubled
- Can deliver 64 bit line to data or instruction cache every cycle
- 11 cycle latency
- 256 KB for each cache

Haswell L3 Cache

- Shared between all cores
- Size varies between models and generations between 6MB and 15MB
- Most Haswell models have an 8MB cache
- Size reduced for power efficiency

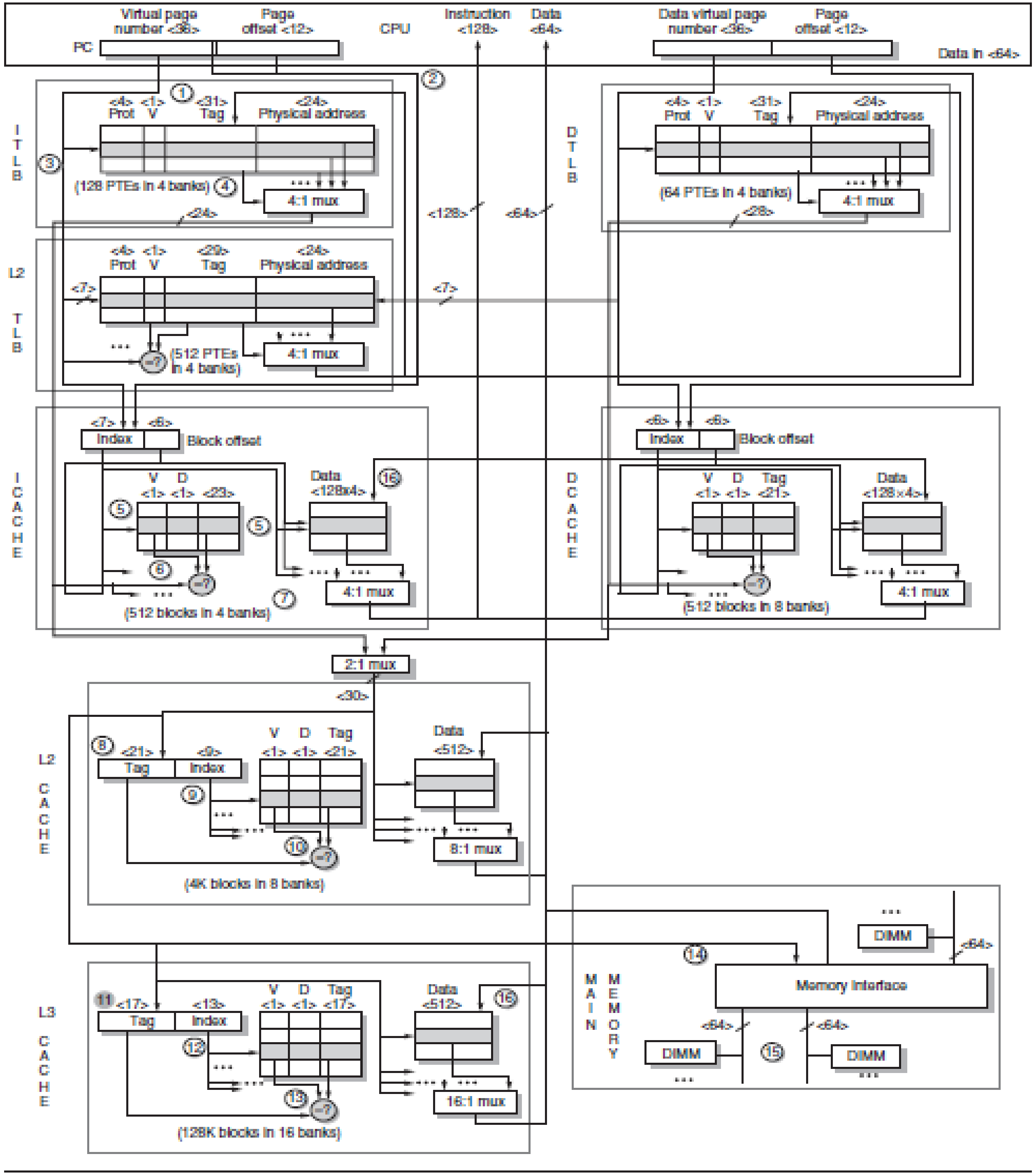
Shared Data

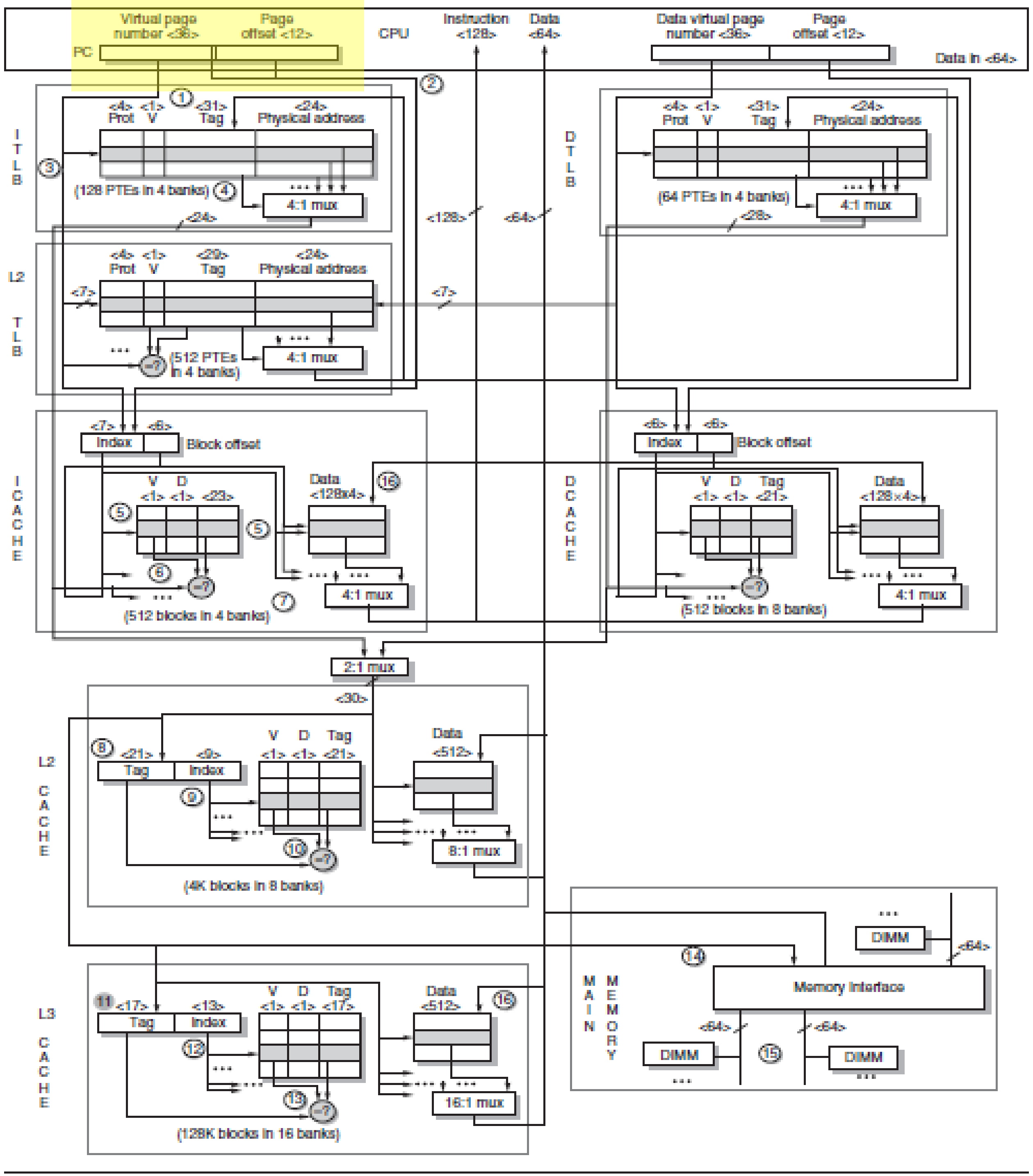
- Transactional Synchronization Extensions
  - Transactional memory
- Hardware Lock Elision
  - Backwards Compatible, Windows only
  - Uses instruction prefixes to lock and release
- Restricted Transactional Memory
  - Newer, more flexible
  - Fallback code in case of failure

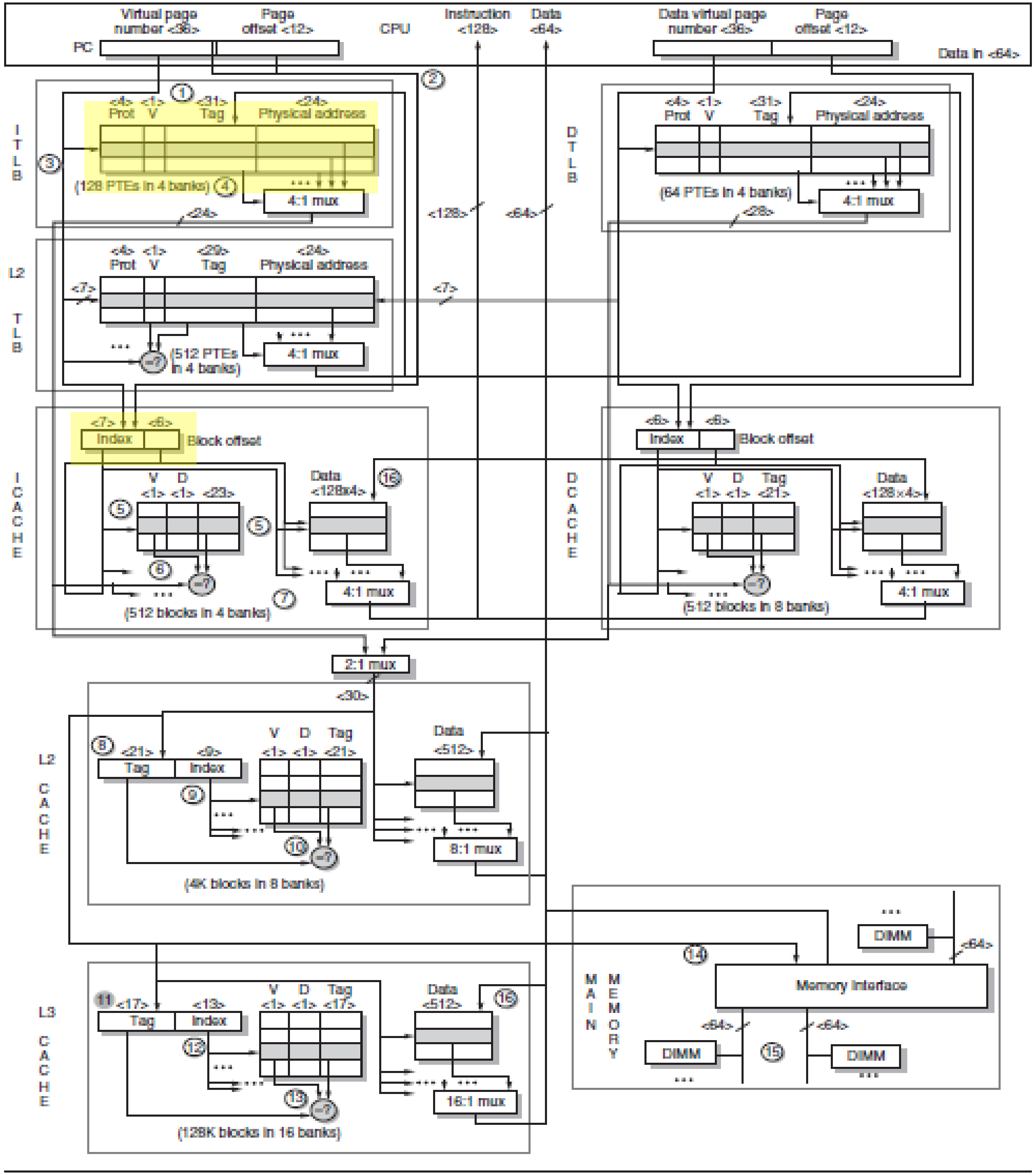
Pre-fetching

- Fetch Instructions/Data before needed
  - On a miss 2 blocks are fetched
- If successful, miss will grab from buffer, and pre-fetch next block

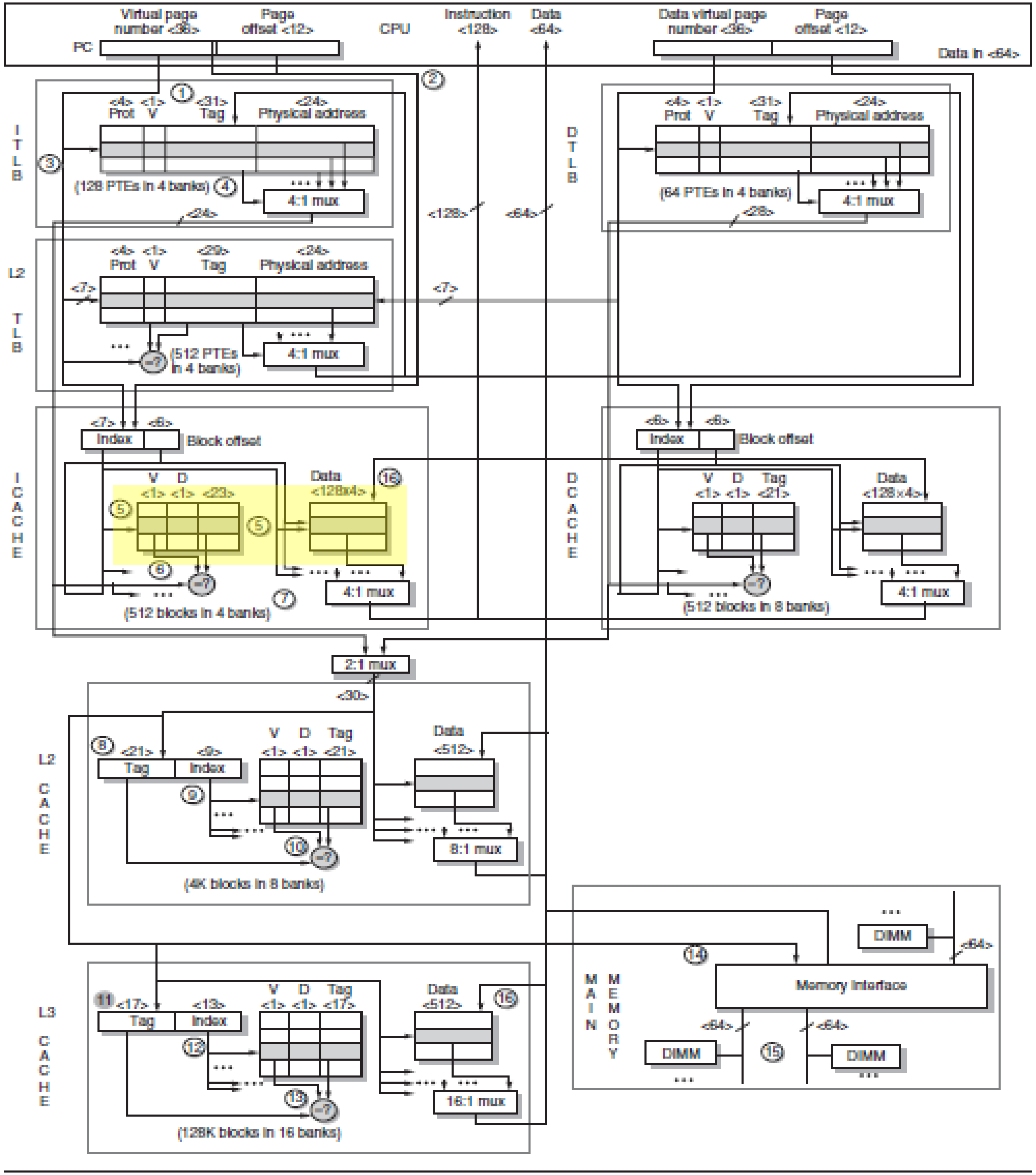
Memory Hierarchy Access Steps

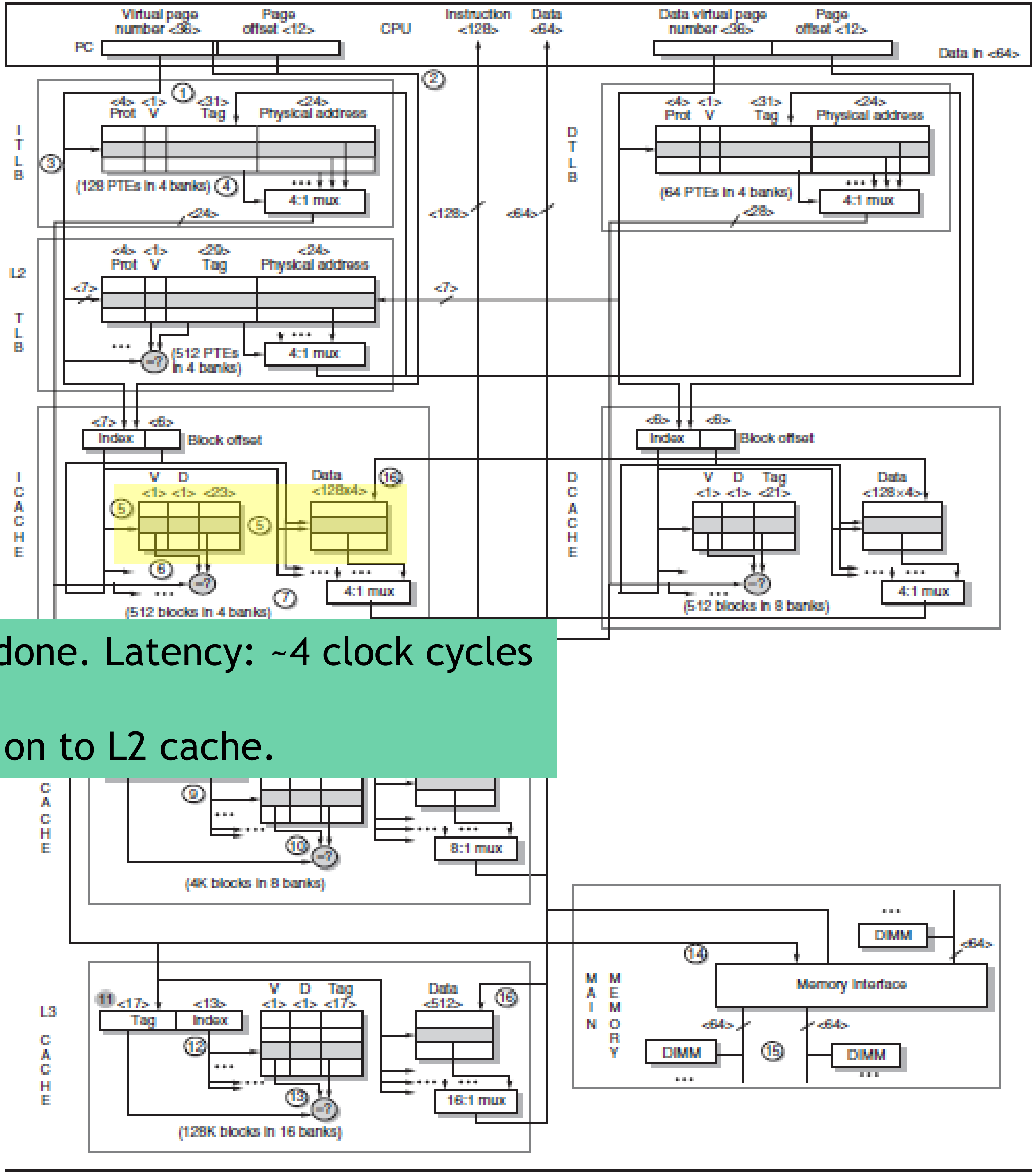



Virtual page number <36>
Page offset <12>
PC
CPU
Instruction <128>
Data <64>
Data virtual page number <36>
Page offset <12>
Data in <64>
1
2
3
4
I TLB
<4> Prot
<1> V
<31> Tag
<24> Physical address
(128 PTEs in 4 banks)
4:1 mux
<24>
<128>
<64>
D TLB
<4> Prot
<1> V
<31> Tag
<24> Physical address
(64 PTEs in 4 banks)
4:1 mux
<28>
L2 TLB
<4> Prot
<1> V
<29> Tag
<24> Physical address
<7>
=?
(512 PTEs in 4 banks)
4:1 mux
<7>
I CACHE
<7> Index
<6> Block offset
V <1>
D <1>
<23>
Data <128x4>
5
6
7
16
=?
(512 blocks in 4 banks)
4:1 mux
D CACHE
<6> Index
<6> Block offset
V <1>
D <1>
Tag <21>
Data <128×4>
=?
(512 blocks in 8 banks)
4:1 mux
2:1 mux
<30>
L2 CACHE
8
<21> Tag
<9> Index
V <1>
D <1>
Tag <21>
Data <512>
9
10
=?
8:1 mux
(4K blocks in 8 banks)
L3 CACHE
11
<17> Tag
<13> Index
V <1>
D <1>
Tag <17>
Data <512>
12
13
16
=?
16:1 mux
(128K blocks in 16 banks)
MAIN MEMORY
14
15
Memory interface
DIMM
DIMM
DIMM
<64>
<64>
<64>



Virtual page number <36>
Page offset <12>
PC
CPU
Instruction <128>
Data <64>
Data virtual page number <36>
Page offset <12>
Data in <64>
ITLB
<4> Prot
<1> V
<31> Tag
<24> Physical address
(128 PTEs in 4 banks)
4:1 mux
<24>
<128>
<64>
DTLB
(64 PTEs in 4 banks)
<28>
L2 TLB
<29> Tag
<7>
(512 PTEs in 4 banks)
=?
ICACHE
<7> Index
<6> Block offset
V <1>
D <1>
<23>
Data <128x4>
(512 blocks in 4 banks)
DCACHE
<6> Index
<6> Block offset
Tag <21>
Data <128×4>
(512 blocks in 8 banks)
2:1 mux
<30>
L2 CACHE
<21> Tag
<9> Index
Tag <21>
Data <512>
8:1 mux
(4K blocks in 8 banks)
L3 CACHE
<17> Tag
<13> Index
Tag <17>
Data <512>
16:1 mux
(128K blocks in 16 banks)
MAIN MEMORY
DIMM
Memory interface
<64>
1
2
3
4
5
6
7
8
9
10
11
12
13
14
15
16

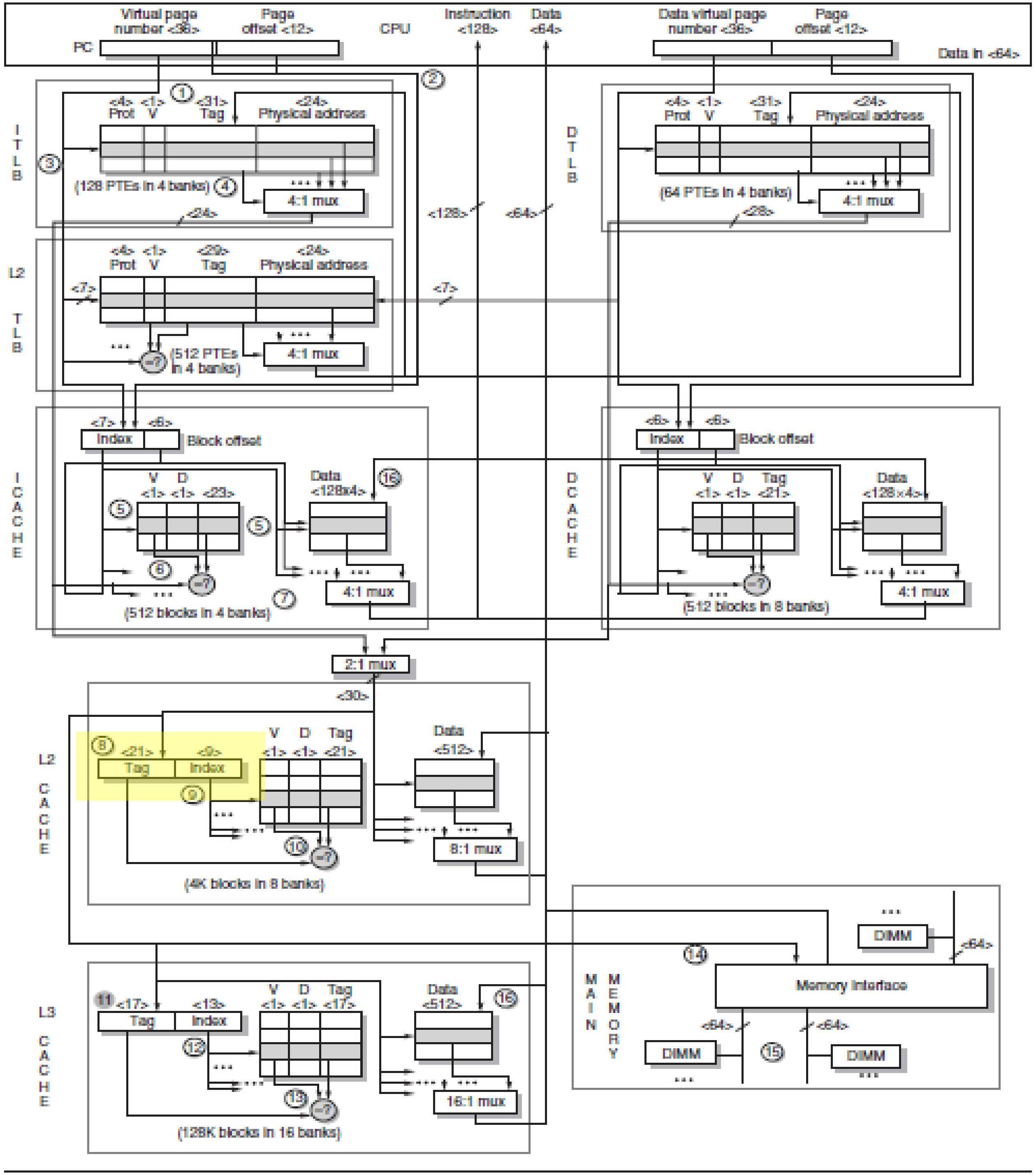

Virtual page number <36>
Page offset <12>
PC
CPU
Instruction <128>
Data <64>
Data virtual page number <36>
Page offset <12>
Data in <64>
1
2
3
4
I TLB
<4> Prot
<1> V
<31> Tag
<24> Physical address
(128 PTEs in 4 banks)
4:1 mux
<24>
<128>
<64>
D TLB
<4> Prot
<1> V
<31> Tag
<24> Physical address
(64 PTEs in 4 banks)
4:1 mux
<28>
L2 TLB
<4> Prot
<1> V
<29> Tag
<24> Physical address
<7>
=?
(512 PTEs in 4 banks)
4:1 mux
<7>
I CACHE
<7> Index
<6> Block offset
V <1>
D <1>
<23>
Data <128x4>
5
5
6
=?
7
16
(512 blocks in 4 banks)
4:1 mux
D CACHE
<6> Index
<6> Block offset
V <1>
D <1>
Tag <21>
Data <128×4>
=?
(512 blocks in 8 banks)
4:1 mux
2:1 mux
<30>
L2 CACHE
8
<21> Tag
<9> Index
9
V <1>
D <1>
Tag <21>
Data <512>
10
=?
8:1 mux
(4K blocks in 8 banks)
L3 CACHE
11
<17> Tag
<13> Index
12
V <1>
D <1>
Tag <17>
Data <512>
16
13
=?
16:1 mux
(128K blocks in 16 banks)
MAIN MEMORY
14
DIMM
<64>
Memory interface
<64>
<64>
DIMM
15
DIMM

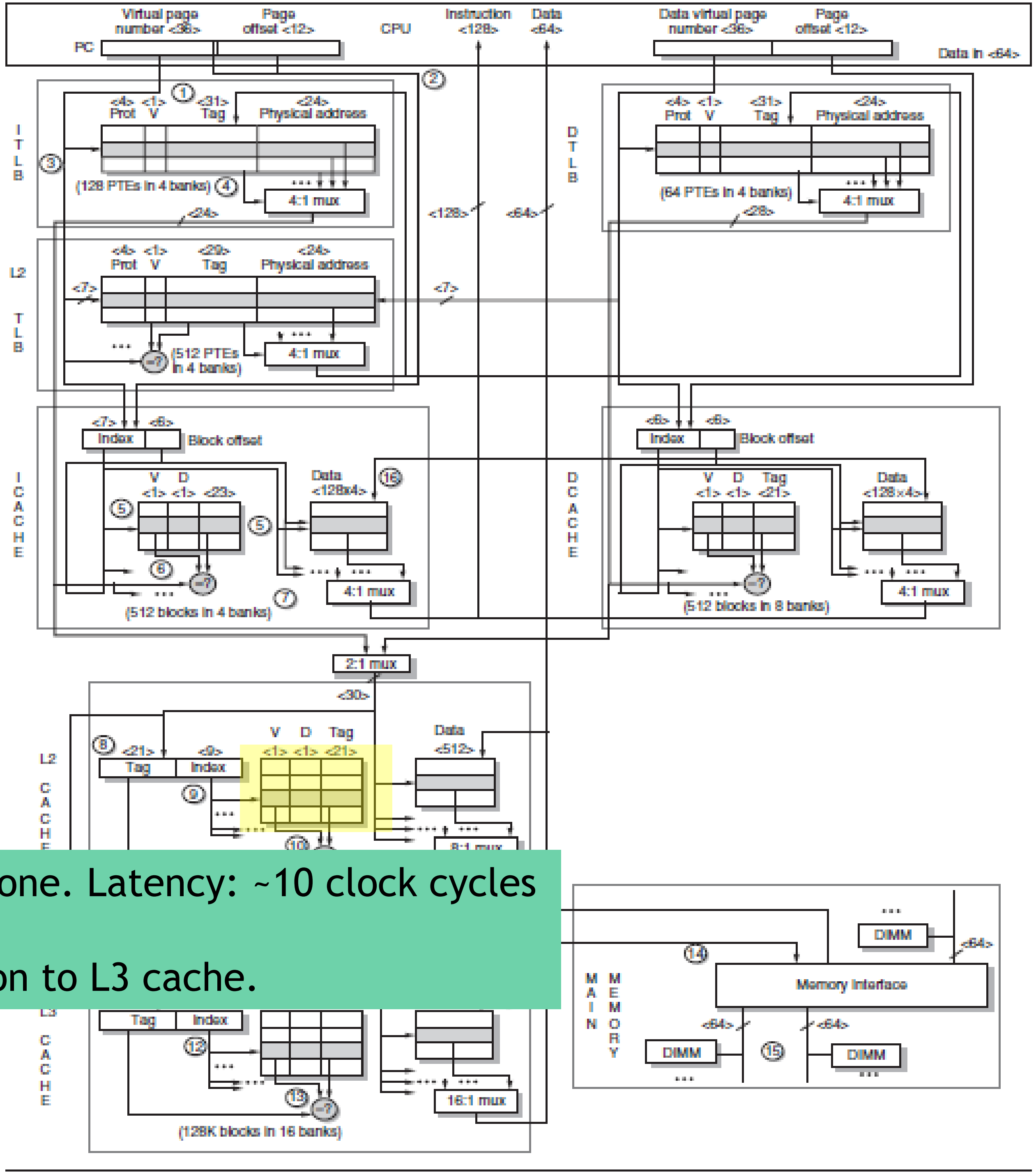


Cache hit! We’re done. Latency: ~4 clock cycles

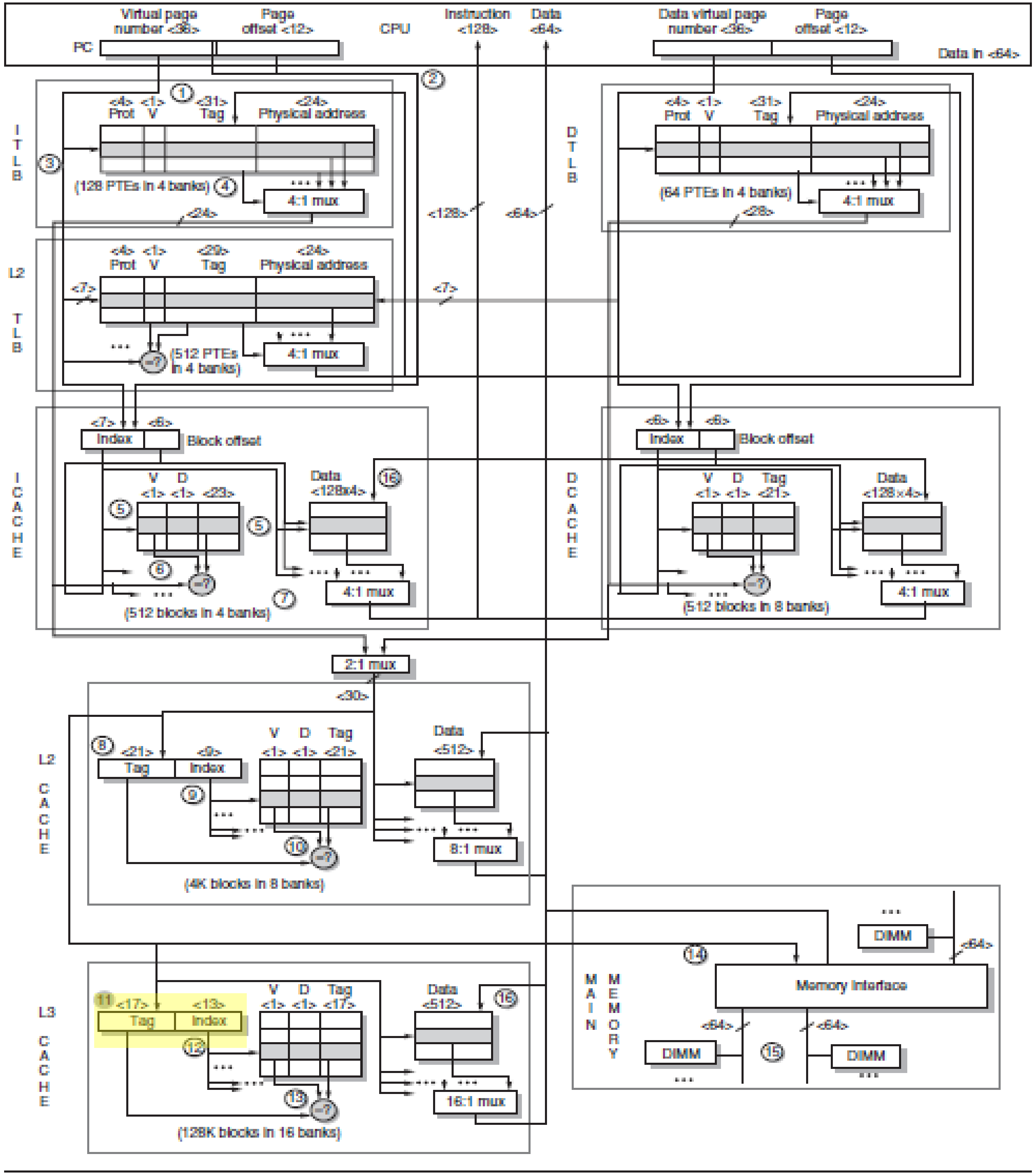
OR

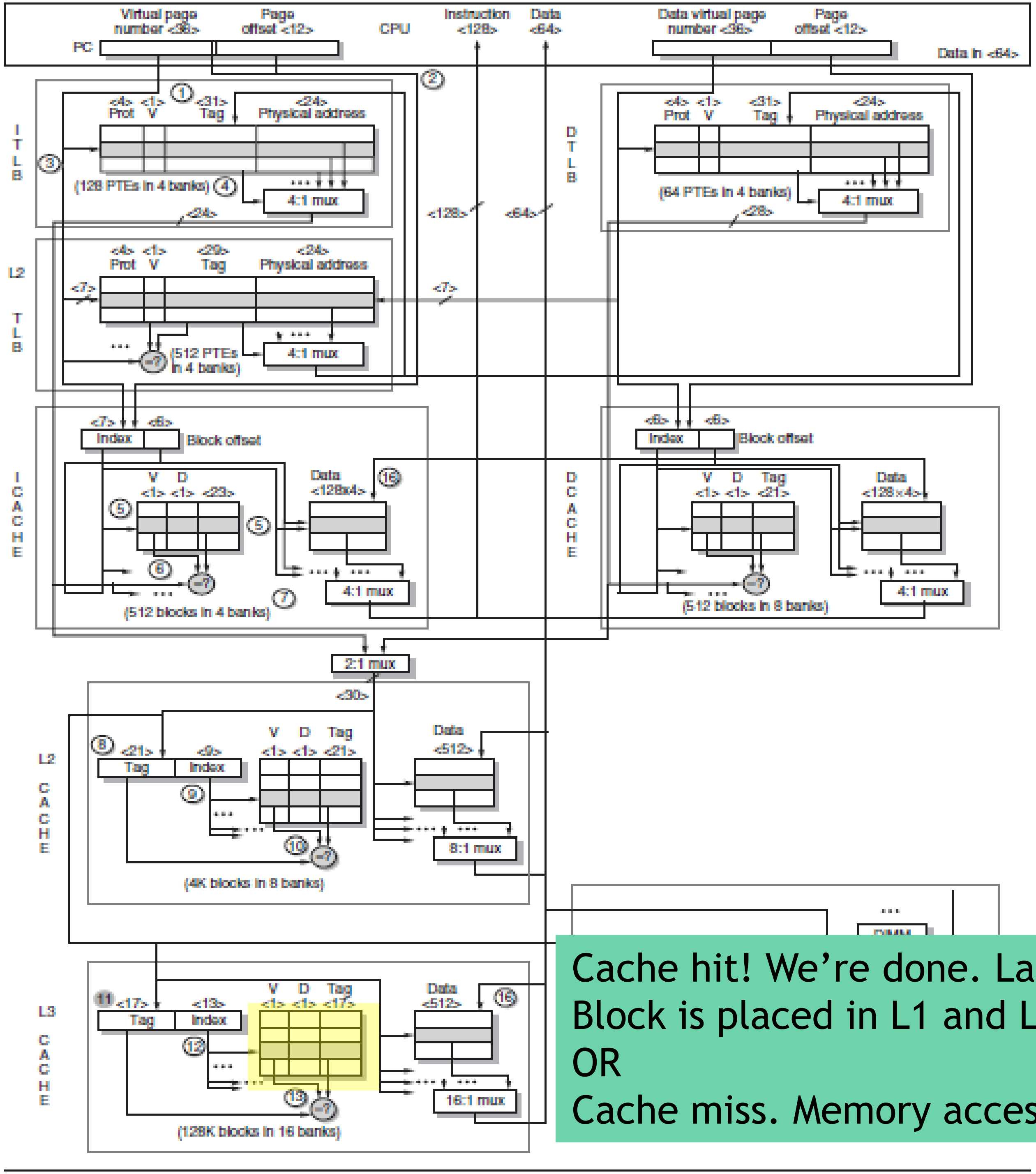
Cache miss. Move on to L2 cache.


Virtual page number <36>
Page offset <12>
PC
CPU
Instruction <128>
Data <64>
Data virtual page number <36>
Page offset <12>
Data in <64>
1
2
3
4
I TLB
<4> Prot
<1> V
<31> Tag
<24> Physical address
(128 PTEs in 4 banks)
4:1 mux
<24>
<128>
<64>
D TLB
<4> Prot
<1> V
<31> Tag
<24> Physical address
(64 PTEs in 4 banks)
4:1 mux
<28>
L2 TLB
<4> Prot
<1> V
<29> Tag
<24> Physical address
<7>
=?
(512 PTEs in 4 banks)
4:1 mux
<7>
I CACHE
<7> Index
<6> Block offset
V <1>
D <1>
<23>
Data <128x4>
5
5
6
7
16
=?
(512 blocks in 4 banks)
4:1 mux
D CACHE
<6> Index
<6> Block offset
V <1>
D <1>
Tag <21>
Data <128×4>
=?
(512 blocks in 8 banks)
4:1 mux
2:1 mux
<30>
L2 CACHE
8
<21> Tag
<9> Index
9
V <1>
D <1>
Tag <21>
Data <512>
10
=?
8:1 mux
(4K blocks in 8 banks)
MAIN MEMORY
DIMM
<64>
14
Memory interface
<64>
<64>
DIMM
15
DIMM
L3 CACHE
11
<17> Tag
<13> Index
12
V <1>
D <1>
Tag <17>
Data <512>
16
13
=?
16:1 mux
(128K blocks in 16 banks)

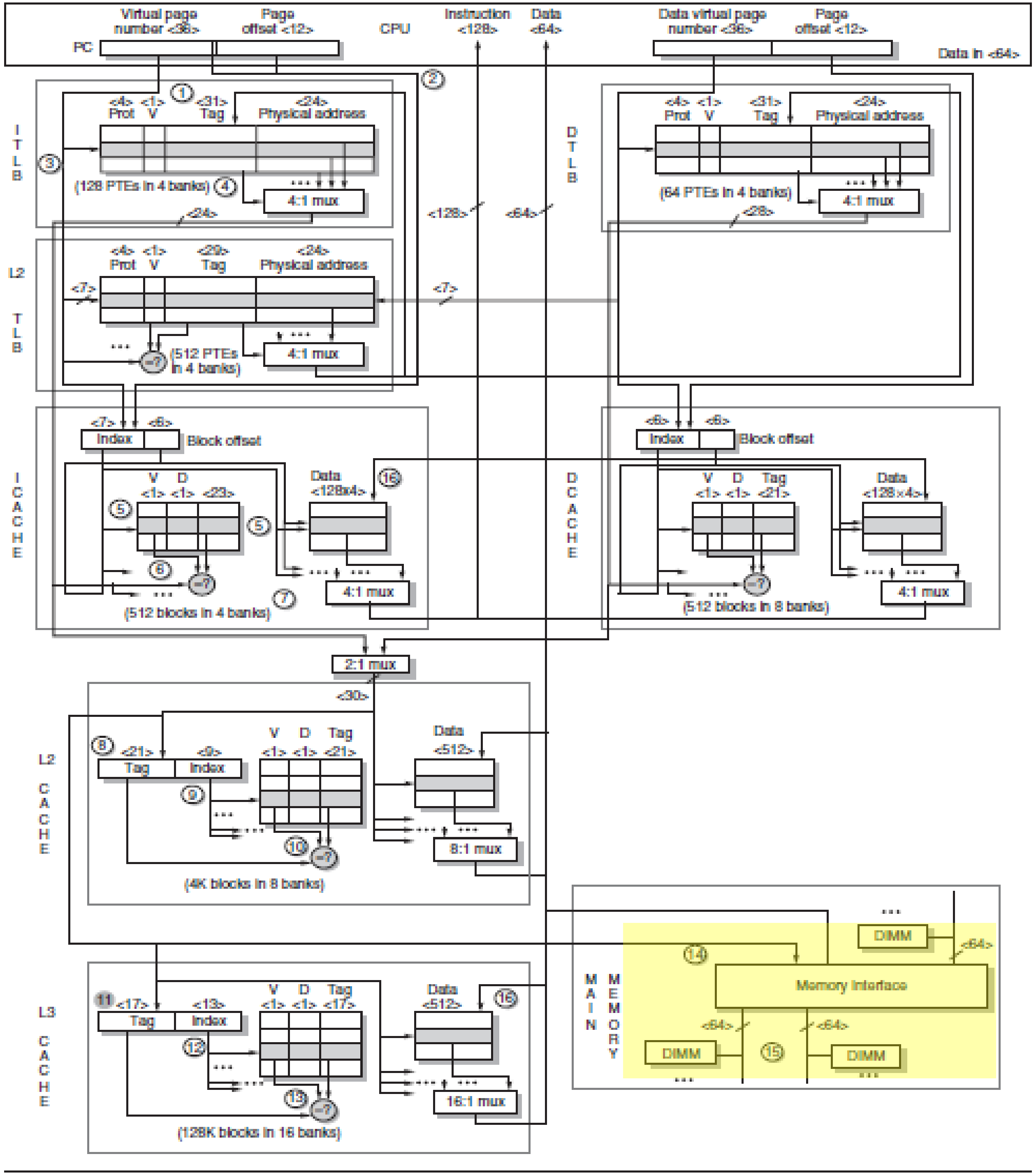


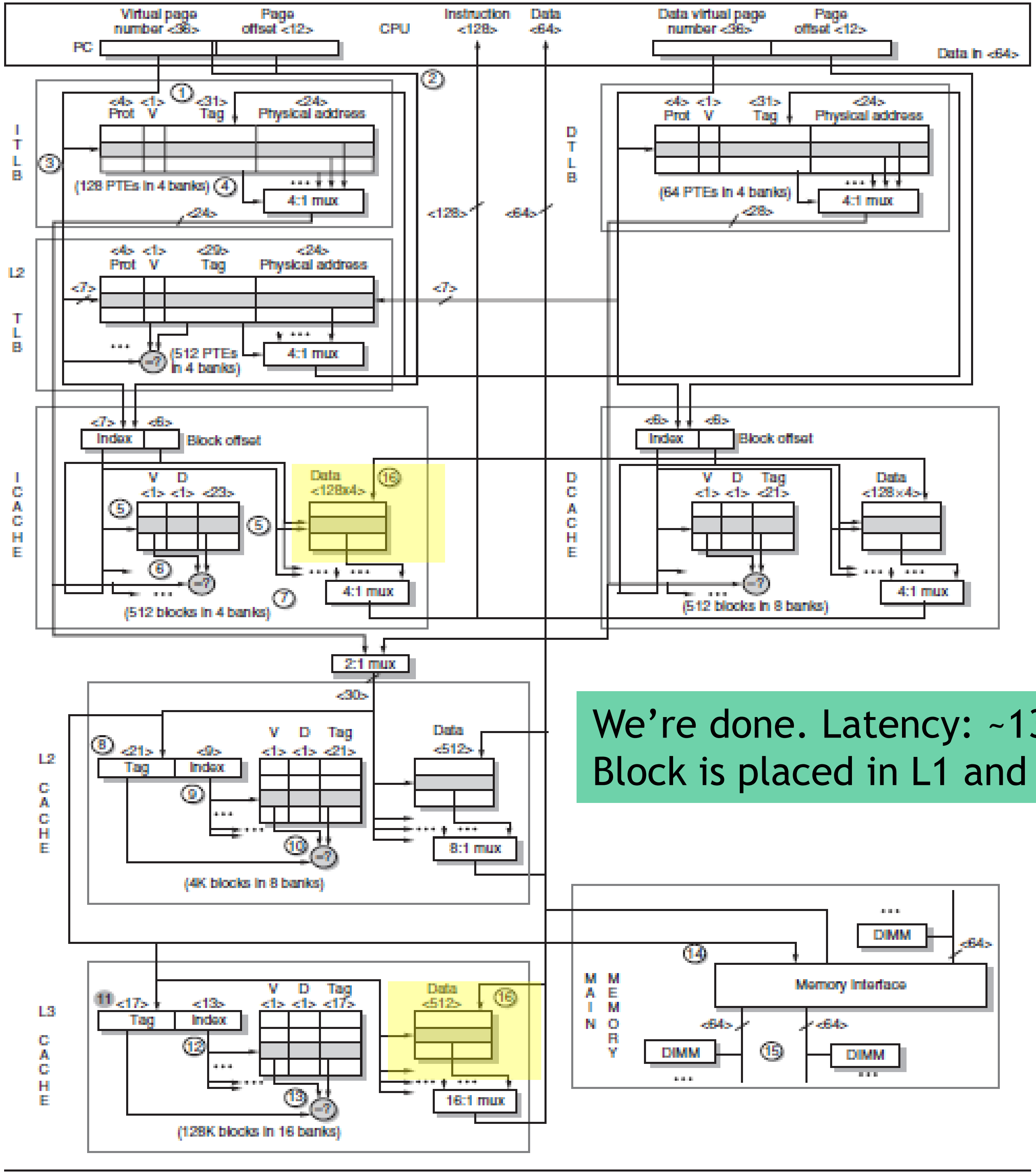
Cache hit! We’re done. Latency: ~10 clock cycles
OR
Cache miss. Move on to L3 cache.


Virtual page number <36>
Page offset <12>
PC
CPU
Instruction <128>
Data <64>
Data virtual page number <36>
Page offset <12>
Data in <64>
1
2
3
4
I TLB
<4> Prot
<1> V
<31> Tag
<24> Physical address
(128 PTEs in 4 banks)
4:1 mux
<24>
<128>
<64>
D TLB
<4> Prot
<1> V
<31> Tag
<24> Physical address
(64 PTEs in 4 banks)
4:1 mux
<28>
L2 TLB
<4> Prot
<1> V
<29> Tag
<24> Physical address
<7>
<7>
=?
(512 PTEs in 4 banks)
4:1 mux
I CACHE
<7> Index
<6> Block offset
V <1>
D <1>
<23>
Data <128x4>
5
5
6
7
16
=?
(512 blocks in 4 banks)
4:1 mux
D CACHE
<6> Index
<6> Block offset
V <1>
D <1>
Tag <21>
Data <128×4>
=?
(512 blocks in 8 banks)
4:1 mux
2:1 mux
<30>
L2 CACHE
8
<21> Tag
<9> Index
9
10
V <1>
D <1>
Tag <21>
Data <512>
=?
8:1 mux
(4K blocks in 8 banks)
L3 CACHE
11
<17> Tag
<13> Index
12
13
V <1>
D <1>
Tag <17>
Data <512>
16
=?
16:1 mux
(128K blocks in 16 banks)
MAIN MEMORY
14
Memory Interface
DIMM
<64>
<64>
<64>
DIMM
15
DIMM



Cache hit! We’re done. Latency: ~35 clock cycles
Block is placed in L1 and L3 cache
OR
Cache miss. Memory access is initiated.

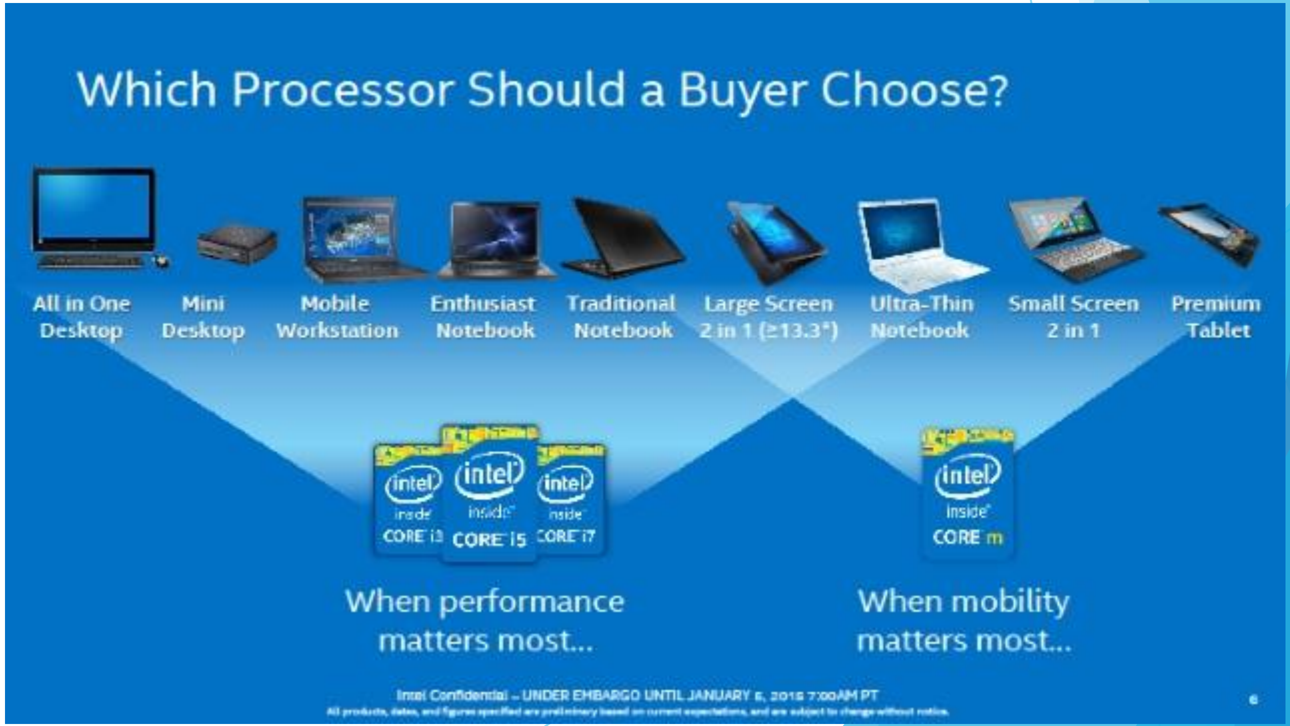

Virtual page number <36>
Page offset <12>
PC
CPU
Instruction <128>
Data <64>
Data virtual page number <36>
Page offset <12>
Data in <64>
1
2
3
4
I T L B
<4> Prot
<1> V
<31> Tag
<24> Physical address
(128 PTEs in 4 banks)
4:1 mux
<24>
<128>
<64>
D T L B
<4> Prot
<1> V
<31> Tag
<24> Physical address
(64 PTEs in 4 banks)
4:1 mux
<28>
L2 T L B
<4> Prot
<1> V
<29> Tag
<24> Physical address
<7>
<7>
=?
(512 PTEs in 4 banks)
4:1 mux
I C A C H E
<7> Index
<6> Block offset
V <1>
D <1>
<23>
Data <128x4>
5
5
6
7
16
=?
(512 blocks in 4 banks)
4:1 mux
D C A C H E
<6> Index
<6> Block offset
V <1>
D <1>
Tag <21>
Data <128×4>
=?
(512 blocks in 8 banks)
4:1 mux
2:1 mux
<30>
L2 C A C H E
8
<21> Tag
<9> Index
9
10
V <1>
D <1>
Tag <21>
Data <512>
=?
8:1 mux
(4K blocks in 8 banks)
L3 C A C H E
11
<17> Tag
<13> Index
12
13
V <1>
D <1>
Tag <17>
Data <512>
16
=?
16:1 mux
(128K blocks in 16 banks)
M A I N M E M O R Y
14
15
Memory interface
DIMM
DIMM
DIMM
<64>
<64>
<64>



We’re done. Latency: ~135 clock cycles
Block is placed in L1 and L3 cache.

Generation 5 (Broadwell)

- Currently mobile only (Lower power systems)
  - Two cores
- Shrunk to 14 nm
- Power Consumption down to 15 w
- No low-end desktop processors
- Extended instruction set



Which Processor Should a Buyer Choose?

When performance matters most...

When mobility matters most...

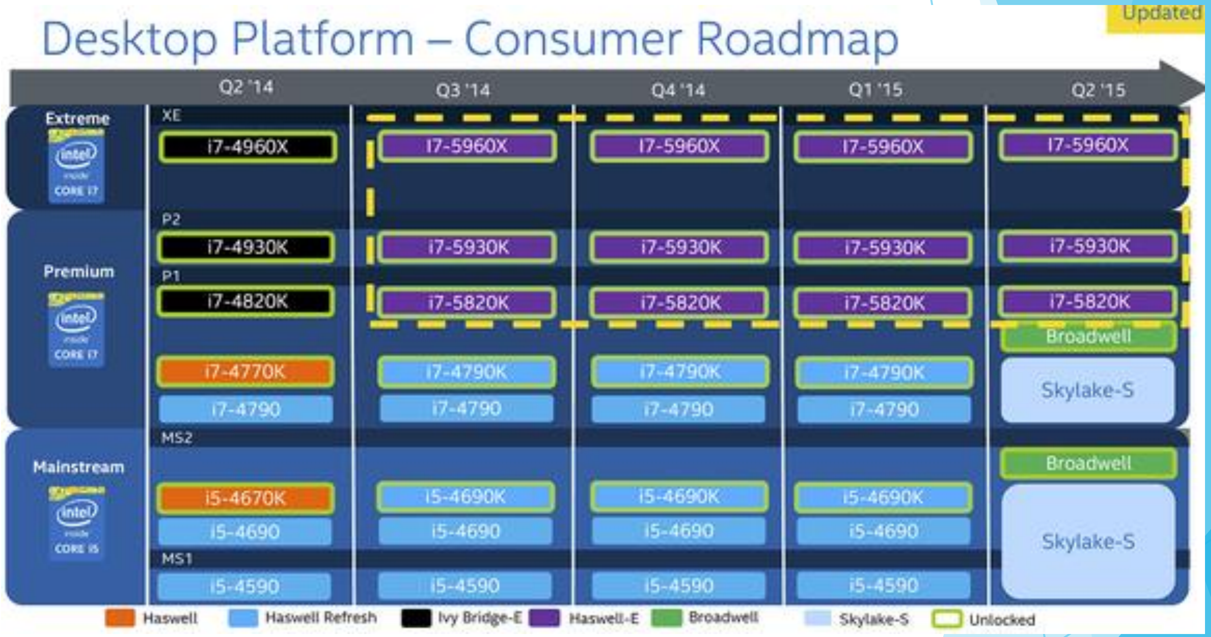
Intel Confidential – UNDER EMBARGO UNTIL JANUARY 5, 2015 7:00AM PT

All products, dates, and figures specified are preliminary based on current expectations, and are subject to change without notice.

6

Future Releases

- Broadwell Desktop
  - Many manufacturers plan to skip
  - Possibly due to lack of low-end offerings
- Skylake
  - Second half of 2015



Updated

Desktop Platform – Consumer Roadmap

		Q2 '14	Q3 '14	Q4 '14	Q1 '15	Q2 '15
Extreme	XE	i7-4960X	i7-5960X	i7-5960X	i7-5960X	i7-5960X
Premium	P2	i7-4930K	i7-5930K	i7-5930K	i7-5930K	i7-5930K
	P1	i7-4820K	i7-5820K	i7-5820K	i7-5820K	i7-5820K
						Broadwell
		i7-4770K	i7-4790K	i7-4790K	i7-4790K	Skylake-S
		i7-4790	i7-4790	i7-4790	i7-4790	
Mainstream	MS2					Broadwell
		i5-4670K	i5-4690K	i5-4690K	i5-4690K	Skylake-S
		i5-4690	i5-4690	i5-4690	i5-4690	
	MS1	i5-4590	i5-4590	i5-4590	i5-4590	

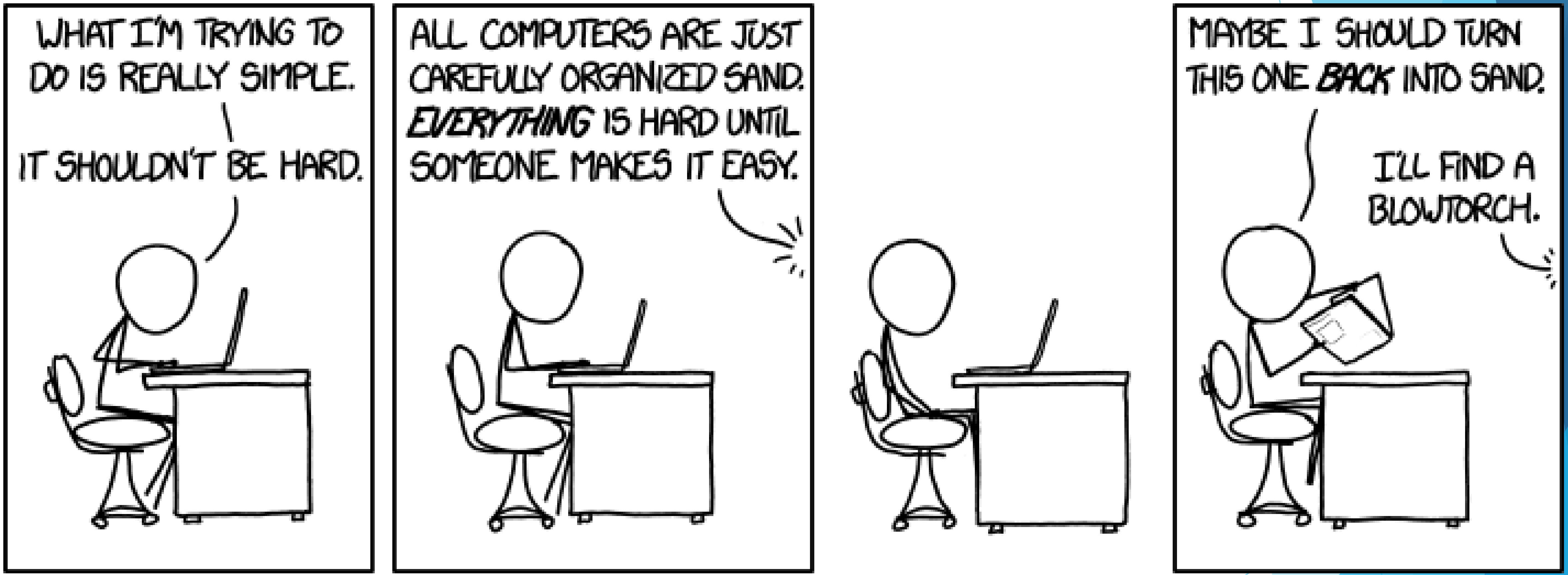
🟧 Haswell 🟦 Haswell Refresh ⬛ Ivy Bridge-E 🟪 Haswell-E 🟩 Broadwell ▢ Skylake-S □ Unlocked

Metric	Nehalem	Sandy Bridge	Haswell
L1 Instruction Cache	32K, 4-way	32K, 8-way	32K, 8-way
L1 Data Cache	32K, 8-way	32K, 8-way	32K, 8-way
Fastest Load-to-use	4 cycles	4 cycles	4 cycles
Load bandwidth	16 Bytes/cycle	32 Bytes/cycle (banked)	**64 Bytes/cycle**
Store bandwidth	16 Bytes/cycle	16 Bytes/cycle	**32 Bytes/cycle**
L2 Unified Cache	256K, 8-way	256K, 8-way	256K, 8-way
Fastest load-to-use	10 cycles	11 cycles	11 cycles
Bandwidth to L1	32 Bytes/cycle	32 Bytes/cycle	**64 Bytes/cycle**
L1 Instruction TLB	4K: 128, 4-way 2M/4M: 7/thread	4K: 128, 4-way 2M/4M: 8/thread	4K: 128, 4-way 2M/4M: 8/thread
L1 Data TLB	4K: 64, 4-way 2M/4M: 32, 4-way 1G: fractured	4K: 64, 4-way 2M/4M: 32, 4-way 1G: 4, 4-way	4K: 64, 4-way 2M/4M: 32, 4-way 1G: 4, 4-way
L2 Unified TLB	4K: 512, 4-way	4K: 512, 4-way	**4K+2M shared: 1024, 8-way**

All caches use 64-byte lines

Conclusion

- Why is it faster?
  - Increased Bandwidth
  - Doubled the associativity in L2 TLB
  - Tri Gate Transistors
- Smaller chip size
- Lower power requirements
  - Decreased L3 Cache Size


WHAT I'M TRYING TO DO IS REALLY SIMPLE.
IT SHOULDN'T BE HARD.
ALL COMPUTERS ARE JUST CAREFULLY ORGANIZED SAND. *EVERYTHING* IS HARD UNTIL SOMEONE MAKES IT EASY.
MAYBE I SHOULD TURN THIS ONE *BACK* INTO SAND.
I'LL FIND A BLOWTORCH.

Questions?