Intel Core i7 Memory Hierarchy

Amanda Adkins, Brett Ammeson, James Anouna, Tony Garside, Lukas Hunker, Sam Mailand
Intel i7 Timeline

- Nehalem 2008
- Sandy Bridge 2011
- Ivy Bridge 2012
- Haswell 2013
- Broadwell 2015
- Skylake 2015
The Tick-Tock model through the years

- **45nm**: Intel® microarchitecture code name Nehalem
- **32nm**: Intel® microarchitecture code name Sandy Bridge
- **22nm**: Intel® microarchitecture code name Haswell
- **14nm**: Increased performance, new capabilities, energy efficiency, and form factor advances
Core i7 Basic Structure

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

Nehalem (First Gen)

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

![Graph showing cache bandwidth for Core i7-3770K and Core i7-4770K with higher L1D Cache, L2 Cache, and L3 Cache bandwidths.](image-url)
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 halves, instruction/data.
  - L3 cache shared among all cores and is inclusive.
Figure 9.21  The Core i7 memory system.
Virtual Addressing
Physical Addressing

2. Search for matching tag in the set

36-bit memory location as interpreted by the L1 cache:

Physical Page Address (24-bit tag), aligned to 4KB

Set Index

Offset into cache line

24-bit tag 64-byte line

Directory 0 Way 0

match

tag line

tag line

tag line

Dir 1/ Way 1

... Dir 6/ Way 6

Dir 7/ Way 7

Cache hit
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

<table>
<thead>
<tr>
<th>Cache</th>
<th>4 KB</th>
<th>2 MB</th>
</tr>
</thead>
<tbody>
<tr>
<td>DTLB</td>
<td>64</td>
<td>32</td>
</tr>
<tr>
<td>ITLB</td>
<td>128</td>
<td>7 / logical core</td>
</tr>
<tr>
<td>STLB</td>
<td>512</td>
<td>none</td>
</tr>
</tbody>
</table>

Sandy Bridge and Ivy Bridge

<table>
<thead>
<tr>
<th>Cache</th>
<th>Level</th>
<th>4 KB</th>
<th>2 MB</th>
<th>1 GB</th>
</tr>
</thead>
<tbody>
<tr>
<td>DTLB</td>
<td>1st</td>
<td>64</td>
<td>32</td>
<td>4</td>
</tr>
<tr>
<td>ITLB</td>
<td>1st</td>
<td>128</td>
<td>8 / logical core</td>
<td>none</td>
</tr>
<tr>
<td>STLB</td>
<td>2nd</td>
<td>512</td>
<td>none</td>
<td>none</td>
</tr>
</tbody>
</table>

Haswell

<table>
<thead>
<tr>
<th>Cache</th>
<th>Level</th>
<th>4 KB</th>
<th>2 MB</th>
<th>1 GB</th>
</tr>
</thead>
<tbody>
<tr>
<td>DTLB</td>
<td>1st</td>
<td>64</td>
<td>32</td>
<td>4</td>
</tr>
<tr>
<td>ITLB</td>
<td>1st</td>
<td>128</td>
<td>8 / logical core</td>
<td>none</td>
</tr>
<tr>
<td>STLB</td>
<td>2nd</td>
<td>1024</td>
<td>none</td>
<td>none</td>
</tr>
</tbody>
</table>
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
- 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
Cache hit! We’re done. Latency: ~4 clock cycles
OR
Cache miss. Move on to L2 cache.
Cache hit! We’re done. Latency: ~10 clock cycles
OR
Cache miss. Move on to L3 cache.
Cache hit! We’re done. Latency: ~35 clock cycles
Block is placed in L1 and L3 cache
OR
Cache miss. Memory access is initiated.
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
- Shrank to 14 nm
- Power Consumption down to 15 w
- No low-end desktop processors
- Extended instruction set

![Which Processor Should a Buyer Choose?](image-url)
Future Releases

- **Broadwell Desktop**
  - Many manufacturers plan to skip
  - Possibly due to lack of low-end offerings
- **Skylake**
  - Second half of 2015
<table>
<thead>
<tr>
<th>Metric</th>
<th>Nehalem</th>
<th>Sandy Bridge</th>
<th>Haswell</th>
</tr>
</thead>
<tbody>
<tr>
<td>L1 Instruction Cache</td>
<td>32K, 4-way</td>
<td>32K, 8-way</td>
<td>32K, 8-way</td>
</tr>
<tr>
<td>L1 Data Cache</td>
<td>32K, 8-way</td>
<td>32K, 8-way</td>
<td>32K, 8-way</td>
</tr>
<tr>
<td>Fastest Load-to-use</td>
<td>4 cycles</td>
<td>4 cycles</td>
<td>4 cycles</td>
</tr>
<tr>
<td>Load bandwidth</td>
<td>16 Bytes/cycle</td>
<td>32 Bytes/cycle (banked)</td>
<td>64 Bytes/cycle</td>
</tr>
<tr>
<td>Store bandwidth</td>
<td>16 Bytes/cycle</td>
<td>16 Bytes/cycle</td>
<td>32 Bytes/cycle</td>
</tr>
<tr>
<td>L2 Unified Cache</td>
<td>256K, 8-way</td>
<td>256K, 8-way</td>
<td>256K, 8-way</td>
</tr>
<tr>
<td>Fastest load-to-use</td>
<td>10 cycles</td>
<td>11 cycles</td>
<td>11 cycles</td>
</tr>
<tr>
<td>Bandwidth to L1</td>
<td>32 Bytes/cycle</td>
<td>32 Bytes/cycle</td>
<td>64 Bytes/cycle</td>
</tr>
<tr>
<td>L1 Instruction TLB</td>
<td>4K: 128, 4-way</td>
<td>4K: 128, 4-way</td>
<td>4K: 128, 4-way</td>
</tr>
<tr>
<td></td>
<td>2M/4M: 7/thread</td>
<td>2M/4M: 8/thread</td>
<td>2M/4M: 8/thread</td>
</tr>
<tr>
<td>L1 Data TLB</td>
<td>4K: 64, 4-way</td>
<td>4K: 64, 4-way</td>
<td>4K: 64, 4-way</td>
</tr>
<tr>
<td></td>
<td>2M/4M: 32, 4-way</td>
<td>2M/4M: 32, 4-way</td>
<td>2M/4M: 32, 4-way</td>
</tr>
<tr>
<td></td>
<td>1G: fractured</td>
<td>1G: 4, 4-way</td>
<td>1G: 4, 4-way</td>
</tr>
<tr>
<td>L2 Unified TLB</td>
<td>4K: 512, 4-way</td>
<td>4K: 512, 4-way</td>
<td>4K+2M shared:</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>1024, 8-way</td>
</tr>
</tbody>
</table>

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?