Nehalem - Everything You Need to Know about Intel's New Architecture
by Anand Lal Shimpi on 11/3/2008 1:00:00 PM
Posted in CPUs

It's called the Core i7, but we knew it as Nehalem. We go through the entire micro-architecture and explain the new developments from IDF.

Page 1

This is Ronak Singhal, he works for Intel, say hello:

He's kind of focused on the person speaking in this situation, but trust me, he's a nice guy. He also happens to be the lead architect on Nehalem.

Nehalem of course is the latest microarchitecture from Intel, it’s a “tock” if you’re going by Intel’s tick-tock cadence:

It's a new architecture, at least newer than Penryn, but still built on the same 45nm process that debuted with Penryn. Next year we’ll have the 32nm version of Nehalem called Westmere and then Sandy Bridge, a brand new architecture also built on 32nm. But today is all about Nehalem.

Recently Intel announced Nehalem’s branding: the Intel Core i7 microprocessor. I've asked Intel why it’s called this and so far the best response I can get is that the naming will make sense once the rest of the lineup is announced. Intel wouldn’t even let me know what the model numbers are going to look like, so for now all we’ve got is that it’s called the Core i7. I’ll use that and Nehalem interchangeably throughout the course of this article.

Page 2

Looking at Nehalem

Let’s start with this diagram:
What we have above is a single Nehalem core, note that you won’t actually be able to buy one of these as it doesn’t include the memory controller, L3 cache and all of what Intel considers to be the un-core. The diagram is drawn correctly, the execution engine isn’t even 1/3 of the core and nearly as much die area is dedicated to the out-of-order scheduling and retirement logic. Now you can understand why Atom is an in-order core.

A single Nehalem core isn’t made up of a majority of cache. Approximately 1/3 of the core is L1/L2 cache, 1/3 is the out of order execution engine and the remaining 1/3 is decode, the branch prediction logic, memory ordering and paging. Obviously this is a bit deceiving since you’re only looking at the core here, the un-core includes a massive 8MB L3 cache which changes the balance of things considerably:

Here we have a full quad-core Nehalem. What Intel calls the un-core is the L3 cache, the I/O, the memory controller logic and the Quick Path Interconnects (QPI). Desktop Nehalem processors will only have one QPI link (QPI 0) while server/workstation chips will have two (QPI 0 and QPI 1).

The Nehalem architecture is designed to be scalable and modular, you will see dual-core, quad-core and eight-core versions in 2009:

Some versions of Nehalem will also include a graphics core, it will be located in the “un-core” of Nehalem as you’ll soon see. The graphics won’t be Larrabee based, it will simply be a derivative of the current G45 architecture.

Page 3

Not Another Conroe
Comparing Conroe to Pentium 4 was night-and-day, the former was such a radical departure from the NetBurst micro-architecture that seemingly everything was done differently. The Pentium 4 needed a tremendous amount of software optimization to actually extract performance from that chip, Intel has since learned its lesson and no longer expects the software community to re-compile and re-optimize code for every new architecture. Nehalem had to be fast out of the box, so it was designed that way.

Conroe was the first Intel processor to introduce this 4-issue front end. The processor could decode, rename and retire up to four micro-ops at the same time. Conroe’s width actually went under utilized a great deal of the time, something that Nehalem did address, but fundamentally there was no reason to go wider.

Intel introduced macro-ops fusion in Conroe, a feature where two coupled x86 instructions could be "fused" and treated as one. They would decode, execute and retire as a single instruction instead of two, effectively widening the hardware in certain situations.

Nehalem added additional instructions that could be fused together, in addition to all of the cases supported in existing Core 2 chips:

- Support all the cases in Intel Core 2 microarchitecture PLUS
  - CMP+Jcc macrofusion added for the following branch conditions
    - JL/JNGE
    - JGE/JNL
    - JLE/JNG
    - JG/JNLE

The other macro-ops fusion enhancement is that now 64-bit instructions can be fused together, whereas in the past only 32-bit instructions could be. It’s a slight performance improvement but 64-bit code could see a performance improvement on Nehalem.

**Page 4**

**Improved Loop Stream Detection**

Core 2 featured a Loop Stream Detector (LSD), the point of this logic was to detect when the CPU was executing a loop in software, stop predicting branches (and potentially incorrectly predicting the last branch of the loop) and simply stream instructions out of the LSD:

![Diagram of Loop Stream Detector](http://www.anandtech.com/print/2594)
The branch prediction and fetch hardware could be disabled and in current Core 2 CPUs you could hold up to 18 instructions in the LSD and simply stream them over and over again into the decode engine until the loop was completed or you ran out of instructions in the LSD.

In Nehalem, the LSD is moved behind the decoder and now caches decoded micro-ops. If a loop is detected, the branch prediction, fetch and decode hardware can now all be powered down and the LSD can stream directly into the re-order buffer. Nehalem can cache 28 micro-ops in its LSD, which actually works out to more “instructions” than what Core 2 could do.

Page 5
Understanding Nehalem’s Server Focus (and Branch Predictors)

I’ve talked about these improvements before so I won’t go into such great detail here, but Nehalem made some moderate improvements on Intel’s already very strong branch predictors.

The processor now has a second level branch predictor that is slower, but looks at a much larger history of branches and whether or not they were taken. The inclusion of the L2 branch predictor enables applications with very large code sizes (Intel gave the example of database applications), to enjoy improved branch prediction accuracy.

The renamed return stack buffer is also a very important enhancement to Nehalem. Mispredicts in the pipeline can result in incorrect data being populated into Penryn’s return stack (a data structure that keeps track of where in memory the CPU should begin executing after working on a function). A return stack with renaming support prevents corruption in the stack, so as long as the calls/returns are properly paired you’ll always get the right data out of Nehalem’s stack even in the event of a mispredict.

The targeted applications here are very important: Nehalem is designed to fix Intel’s remaining shortcomings in the server space. Our own Johan de Gelas has been talking about Intel not being as competitive in the server market as on the desktop for quite some time now. He even published a very telling article on Nehalem’s server focus before IDF started. While many of Nehalem’s improvements directly impact the desktop market, motivating its design were servers.

This is an important thing to realize because this whole architecture, where Nehalem and its predecessors came from started on the mobile side of the business with Banias/Pentium-M and Centrino. We may have just come full circle with Nehalem, where we once again have the server market driving the microprocessor design for the desktop and mobile chips as well.

The key distinction here and what will hopefully prevent Nehalem’s successors from turning into Pentium 4 redux is Intel’s performance/power ratio golden rule. Nehalem and Atom were both designed, for the first time ever in Intel history, with one major rule on power/performance. For every feature proposed for Nehalem (and Atom), for each 1% increase in power consumption that feature needed to provide a corresponding 2% or greater increase in performance. If the feature couldn’t equal or beat this ratio, it wasn’t added, regardless of how desirable.

Page 6
Finish Him!

The execution engine of Nehalem is largely unchanged from Penryn; just like the front end was already wide enough, so was the execution end of the architecture:
Intel did however increase the size of many data structures on the chip and increase the size of the out of order scheduling window. Nehalem can now keep 128 µops in flight, up from 96 in Conroe/Merom/Penryn.

Increased Parallelism

- Goal: Keep powerful execution engine fed
- Nehalem increases size of out of order window by 33%
- Must also increase other corresponding structures

<table>
<thead>
<tr>
<th>Structure</th>
<th>Intel® Core™ microarchitecture (formerly Merom)</th>
<th>Intel® Core™ microarchitecture (Nehalem)</th>
<th>Comment</th>
</tr>
</thead>
<tbody>
<tr>
<td>Reservation Station</td>
<td>32</td>
<td>36</td>
<td>Dispatches operations to execution units</td>
</tr>
<tr>
<td>Load Buffers</td>
<td>32</td>
<td>48</td>
<td>Tracks all load operations allocated</td>
</tr>
<tr>
<td>Store Buffers</td>
<td>20</td>
<td>32</td>
<td>Tracks all store operations allocated</td>
</tr>
</tbody>
</table>

The reservation station can now hold 36 µops up from 32; both load and store buffers have increased from 32 and 20 to 48 and 32 entries, respectively.

Nehalem may not be any wider than Conroe/Penryn, but it should make better usage of its architecture than any of its predecessors.

Page 7

New TLBs, Faster Unaligned Cache Accesses

Historically the applications that push a microprocessor's limits on TLB size and performance are server applications, once again with things like databases. Nehalem not only increases the sizes of its TLBs but also adds a 2nd level unified TLB for both code and data.

| # of Entries | 
|---------------|---------|
| 1st Level Instruction TLBs                   |         |
| Small Page (4k)                          | 128     |
| Large Page (2M/4M)                        | 7 per thread |
| 1st Level Data TLBs                       |         |
| Small Page (4k)                           | 64      |
| Large Page (2M/4M)                        | 32      |
| New 2nd Level Unified TLB                  |         |
| Small Page Only                           | 512     |

Another potentially significant fix that I've already talked about before is Nehalem's faster unaligned cache accesses. The largest size of a SSE memory operation is 16-bytes (128-bits). For any of these load/store operations there are two versions: an op that is aligned on a 16-byte boundary, and an op that is unaligned.

Compilers will use the unaligned operation if they can't guarantee that a memory access won't fall on a 16-byte boundary. In all Core 2 processors, the unaligned op was significantly slower than the aligned op, even on aligned data.

The problem is that most compilers couldn't guarantee that data would be aligned properly and would default to using the unaligned op, even though it would sometimes be used on aligned data.
With Nehalem, Intel has not only reduced the performance penalty of the unaligned op but also made it so that if you use the unaligned op on aligned data, there’s absolutely no performance degradation. Compilers can now always use the unaligned op without any fear of a reduction in performance.

To get around the unaligned data access penalty in previous Core 2 architectures, developers wrote additional code specifically targeted at this problem. Here’s one area where a re-optimization/re-compile would help since on Nehalem they can just go back to using the unaligned op.

Faster Synchronization Primitives

- Multi-threaded software becoming more prevalent
- **Scalability** of multi-thread applications can be limited by synchronization
- Synchronization primitives: LOCK prefix, XCHG
- Reduce synchronization latency for legacy software

Thread synchronization performance is also improved on Nehalem, which leads us in to the next point.

Page 8

...and Now We Understand Why: Hyper Threading

Years ago I asked Pat Gelsinger what he was most excited about in the microprocessor industry, he answered: threading. This was during the early days of Hyper Threading but since the demise of the Pentium 4, we never really got another look at HT on any of the subsequent microprocessors based on Core.

Hyper Threading was Intel’s marketing name for simultaneous multi-threading (SMT), the ability to fetch instructions from two threads at the same time instead of just a single one. The OS sees a HT enabled processor as multiple processors, in this case two, and send two threads of instructions to the CPU.

Nehalem sees the return of Hyper Threading and it is met with much, much better performance than we ever saw with Pentium 4 for a number of reasons:

1) Nehalem has much more memory bandwidth and larger caches than Pentium 4, giving it the ability to get data to the core faster and more predictably
2) Nehalem is a much wider architecture than Pentium 4, taking advantage of it demands the use of multiple threads per core.

Just as the first Pentium 4 didn’t ship with Hyper Threading support, the architectures that Nehalem was derived from didn’t have HT either. The time is right with Nehalem for the reasons mentioned above and because there are simply many more applications that can take advantage of HT today than ever before.

The chart below shows the performance gain from enabling HT on a Nehalem processor, no other variables were changed:

Just like on Atom, enabling HT on Nehalem required very little die area. Only the register state, renamed return stack buffer and large page instruction TLBs are duplicated. The rest of the data structures are either partitioned in half when HT is enabled or are “competitively” shared, in that they dynamically determine how much of each resource is allocated per thread:

http://www.anandtech.com/print/2594
Enabling HT was a very power efficient move for Nehalem, and you can expect that it will improve performance in a number of applications - far more regularly than it ever did with the Pentium 4.

Now it should also make sense why Intel increased buffer sizes in Nehalem, because many of those buffers now have to deal with ops coming from two threads instead of just one. That’s more instructions to decode, better utilization of the front end, more micro-ops in the pipeline, more micro-ops in the re-order engine and more micro-ops executing concurrently.

Page 9
Cache Hierarchy

Once again, I’ve already talked about the Nehalem cache hierarchy in great detail so I’ll keep this to a quick overview.

Nehalem, like AMD’s Phenom, features a 3-level cache hierarchy. There’s a 64KB L1 cache (32KB I + 32KB D), a 256KB L2 cache (per core, unshared) and up to an 8MB L3 cache (shared among all cores).

The L1 cache is the same size as what we have in Penryn, but it’s actually slower (4 cycles vs. 3 cycles). Intel slowed down the L1 cache as it was gating clock speed, especially as the chip grew in size and complexity. Intel estimated a 2 - 3% performance hit due to the higher latency L1 cache in Nehalem.

The L2 cache also gets a hit - while in Penryn we had a 6MB L2 cache shared between two cores, Nehalem moves the L2 cache next to each individual core and reduces the size to a meager 256KB. We haven’t had a high performance Intel CPU with such a small L2 cache since the first Pentium 4. The smaller L2 is quicker, it only takes 10 cycles from load to get data out of the L2 cache.

The L2 cache acts as a buffer to the L3 cache so you don’t have all of the cores banging on the L3 cache, requiring tons of bandwidth.

The L3 cache is shared by all cores and in the initial Core i7 processors will be 8MB large, although its size will vary depending on the number of cores. Multi-threaded applications that are being worked on by all cores will enjoy the large, shared L3 cache.

Intel defended its reasoning for using an inclusive cache architecture with Nehalem, something it has always done in the past. Nehalem’s L3 cache is inclusive in that it contains all data stored in the L1 and L2 caches as well. The benefit is that if the CPU looks for data in L3 and doesn’t find it, it knows that the data doesn’t exist in any core’s L1 or L2 caches - thereby saving core snoop traffic, which not only improves performance but reduces power consumption as well.

An inclusive cache also prevents core snoop traffic from getting out of hand as you increase the number of cores, something that Nehalem has to worry about given its aspirations of extending beyond 4 cores.

Page 10
Further Power Managed Cache?

One new thing we learned about Nehalem at IDF was that Intel actually moved to a 8T (8 transistor) SRAM cell design for all of the core cache memory (L1 and L2 caches, but not L3 cache). By moving to a 8T design Intel was able to actually reduce the operating voltage of Nehalem, which in turn reduces power consumption. You may remember that Intel’s Atom team did something similar with its L1 cache:

"Instead of bumping up the voltage and sticking with a small signal array, Intel switched to a register file (1 read/1 write port). The cache now had a larger cell size (8 transistors per cell) which increased the area and footprint of the L1 instruction and data caches. The Atom floorplan had issues accommodating the larger sizes so the data cache had to be cut down from 32KB to 24KB in favor of the power benefits. We wondered why Atom had an asymmetrical L1 data and instruction cache (24KB and 32KB respectively, instead of 32KB/32KB) and it turns out that the cause was voltage.

http://www.anandtech.com/print/2594
A small signal array design based on a 6T cell has a certain minimum operating voltage, in other words it can retain state until a certain \( V_{min} \). In the L2 cache, Intel was able to use a 6T signal array design since it had inline ECC. There were other design decisions at work that prevented Intel from equipping the L1 cache with inline ECC, so the architects needed to go to a larger cell size in order to keep the operating voltage low.

Intel states that Nehalem’s “Core memory converted from 6-T traditional SRAM to 8-T SRAM”. The only memory in the “core” of Nehalem is its L1 and L2 caches, which may help explain why the L2 cache per core is a very small 256KB. It would be costly to increase the transistor count of Nehalem’s 8MB L3 cache by 33%, but it’s also most likely not as necessary since the L3 cache and the rest of the uncore runs on its own voltage plane as I’ll get to shortly.

---

**Page 11**

**Integrated Memory Controller**

In Nehalem’s un-core lies a number of DDR3 memory controllers, on-die and off of the motherboard - finally. The first incarnation of Nehalem will ship with a triple-channel DDR3 memory controller, meaning that DDR3 DIMMs will have to be installed in sets of three in order to get peak bandwidth. Memory vendors will begin selling Nehalem memory kits with three DIMMs just for this reason. Future versions of Nehalem will ship with only two active controllers, but at the high end and for the server market we’ll have three.

With three DDR3 memory channels, Nehalem will obviously have tons of memory bandwidth, which will help feed its wider and hungrier cores. A side effect of a tremendous increase in memory bandwidth is that Nehalem’s prefetchers can work much more aggressively.

I haven’t talked about Nehalem’s server focus in a couple of pages so here we go again. With Xeon and some server workloads, Core 2’s prefetchers were a bit too aggressive so for many enterprise applications the prefetchers were actually disabled. This mostly happened with applications that had very high bandwidth utilization, where the prefetchers would kick in and actually rob the system of useful memory bandwidth.

With Nehalem the prefetcher aggressiveness can be throttled back if there’s not enough available bandwidth.

**QPI**

When Intel made the move to an on-die memory controller it needed a high speed interconnect between chips, thus the Quick Path Interconnect (QPI) was born. I’m not sure whether or not QPI or Hyper Transport is a better name for this.

Each QPI link is bi-directional supporting 6.4 GT/s per link. Each link is 2-bytes wide so you get 12.8GB/s of bandwidth per link in each direction, for a total of 25.6GB/s of bandwidth on a single QPI link.
The high end Nehalem processors will have two QPI links while mainstream Nehalem chips will only have one.

The QPI aspect of Nehalem is much like HT with AMD’s processors, now developers need to worry about Intel systems being a NUMA platform. In a multi-socket Nehalem system, each socket will have its own local memory and applications need to ensure that the processor has its data in the memory attached to it rather than memory attached to an adjacent socket.

Here’s one area where AMD having being so much earlier with an IMC and HT really helps Intel. Much of the software work that has been done to take advantage of AMD’s architecture in the server world will now benefit Nehalem.

**New Instructions**

With Penryn, Intel extended the SSE4 instruction set to SSE4.1 and in Nehalem Intel added a few more instructions which Intel is calling SSE4.2.

The future of Intel’s architectural extensions beyond Nehalem lie in the Advanced Vector Extensions (AVX), which add support for 256-bit vector operations. AVX is an intermediate step between where SSE is today and where Larrabee is going with its instruction set. At some point I suspect we may see some sort of merger between these two ISAs.

**New Stuff: Power Management**

At this year’s IDF the biggest Nehalem disclosures had to do with power management.

Nehalem’s design was actually changed on a fairly fundamental level compared to previous microprocessors. Dynamic domino logic was used extensively in microprocessors like the Pentium 4 and IBM’s Cell processor in order to drive clock speeds up. With Nehalem, Intel has removed all domino logic and moved back to an entirely static CMOS design.
Nehalem’s architects spent over 1 million transistors on including a microcontroller on-die called the Power Control Unit (PCU). That’s around the transistor budget of Intel’s 486 microprocessor, just spent on managing power. The PCU has its own embedded firmware and takes inputs on temperature, current, power and OS requests.

**Power Control Unit**

Each Nehalem core gets its own PLL, so each core can be clocked independently - much like AMD’s Phenom processor. Also like Phenom, each core runs off of the same core voltage - the difference between Nehalem and Phenom however is Intel’s use of integrated power gates.

Through close cooperation between Nehalem’s architects and Intel’s manufacturing engineers, Intel managed to manufacture a very particular material that could act as a power gate between the voltage source being fed to a core, and the core itself.
The benefit is that while still using a single power plane/core voltage, individual Nehalem cores can be completely (nearly) shut off when they are in deep sleep states. Currently in a multi-core CPU (AMD or Intel), all cores need to run at the same voltage, which means that leakage power on idle cores is still high just because there's one or more active cores in the CPU.

Nehalem's power gates allow one or more cores to be operating in an active state at a nominal voltage, while remaining idle cores can have power completely shut off to them - without resorting to multiple power planes, which would drive up motherboard costs and complexity.

The other benefit of doing this power management on-die is that the voltage ramp up/down time is significantly faster than conventional, off-die, methods. Fast voltage switching allows for more efficient power management.

I mentioned earlier that the PCU monitors OS performance state requests, so it can actually make intelligent decisions about what power/performance state to go into, despite what the OS is telling it. There are some situations where Vista (or any other OS) running an application with a high level of interrupts will keep telling the CPU to go into a low power idle state, only to wake it up very shortly thereafter. Nehalem’s PCU can monitor for these sorts of situations and attempt to more intelligently decide what power/performance states it will instruct the CPU to go into, regardless of what the OS thinks it wants.

**Turbo Mode**

This last feature Intel actually introduced with mobile Penryn. The idea was that if you have a dual-core mobile Penryn only running a single threaded application, one core is completely idle, and the total chip TDP is lower than what it was designed for. Intel sought to use these conditions to actually increase the clock speed of the active core by a single speed bump. Unfortunately on mobile Penryn the performance benefit of Turbo mode just wasn’t utilized, for one, Vista always bumps a single thread around on multiple cores so the idle core always alternated between the cores on a chip.

The other issue was that it’s rare that you only had a single thread running on your machine, Vista would always spawn additional threads that would keep your mobile Penryn from entering Turbo mode.

Nehalem does things a little better. Not only can it enable Turbo mode if all cores are idle but it can also enable Turbo mode if only some of the cores are idle, or if all cores are active but not at full utilization.

All Nehalem processors will at least be able to go up a single clock step (133MHz) in Turbo mode, even if all cores are active, just as long as the PCU detects that the TDP hasn’t been exceeded. If the TDP levels are low enough, or the cores idle enough, Nehalem can actually increase clock speeds by more than one clock step. Right now it looks like the only bump you’ll see is 266MHz, which is still quite mild, but Intel appears to have lofty goals for Turbo mode with Nehalem.

Future versions could increase the amount of “turbo” you could get out of Nehalem, and you can imagine situations where it would increase its clock speed more than just 266MHz. The idea is that Intel could actually capitalize on how overclockable its CPUs are and safely increase performance for those who aren’t avid overclockers.

Don’t worry though, Turbo mode can be disabled completely if you’d like.

**Launch Speeds and Performance**

http://www.anandtech.com/print/2594
Intel hasn’t said anything about what speeds or prices Nehalem will launch at, but here’s what I’m expecting.

Three Core i7 branded Nehalem parts, one at 2.66GHz, one at 2.93GHz and one at 3.2GHz - all with the same 8MB L3 cache and all quad-core. There is no conventional FSB, but all of these chips run off of a 133MHz source clock.

With Turbo mode each chip can go up a maximum of two clock steps, (266MHz) and worst case scenario they’ll go up a single clock bump (133MHz) - conditions permitting.

I expect pricing to be pretty reasonable, at least on the 2.66GHz part but it’s unclear exactly what that will be.

I’ve already done a bit of work on expected Nehalem performance. Nehalem’s largest impact will be on servers, without a doubt, but there are many threaded desktop applications where you’ll see significant improvements thanks to Nehalem. Video encoding, 3D rendering, etc... were the biggest areas where we saw a performance boost with Nehalem in our earlier article.

If your apps aren’t well threaded, the Nehalem benefit will be limited to the 0 - 15% range compared to Penryn depending on the app.

Page 15

Final Words

Nehalem was a key focus at this year’s IDF, and in the coming months we should see its availability in the market. You’ll need a new motherboard and CPU, and maybe even new memory, but if you’re running well threaded applications Nehalem won’t disappoint.

Designed from the start to really tackle Intel’s weaknesses in the server market, it makes sense why the first Nehalem scheduled to launch is the high end, quad-core, dual QPI, triple-channel version. The overall architecture is very reminiscent of Barcelona, which does lend credibility to AMD’s direction with its latest core - at least in the enterprise space.

Thanks to the past few years of multi-core development on the desktop, Nehalem should see some impressive gains there as well - again from threaded applications.

The biggest changes with Nehalem are actually the ones that took place behind the scenes. The fundamental changes in design decisions requiring that each expenditure of additional power come at dramatic corresponding increase in performance, is a huge step for Intel. It’s the same design mentality used on the lowest power Intel cores (Atom), which is impressive for Intel’s highest performing Nehalem cores.

We can’t help but be excited about Nehalem as the first tock since the Core 2 processor arrived, but we do wonder what’s next. Much of the performance gains with Nehalem are due to increases in bandwidth and HT, we’ll have to wait two more years to find out what Intel can do to surprise us once more. As Pat Gelsinger told me when AMD integrated the memory controller, he said you can only do that once - what do you do to improve performance next?

While Larrabee will be the focus of Intel’s attention in 2009, Sandy Bridge in 2010 is the next tock to look forward to. Until then, Nehalem should do wonders for Intel’s competitiveness in the enterprise market, and actually be a worthy successor to Conroe on the desktop.

The only other concern I have about Nehalem is how things will play out with the two-channel DDR3 versions of the chip. They will require a different socket and as we saw in the days of Socket-940/939/754 with AMD’s K8, it can easily be a painful process. I do hope that Intel has learned from AMD’s early issues with platforms and K8, it would be a shame if initial Nehalem adopters were eventually left out in the cold. If Intel does launch an affordable 2.66GHz quad-core part, I don’t expect that the enthusiast market will be left out but I’m not much of a fortune teller.

So there you have it, a more complete look at Nehalem - the only thing we’re missing is a full performance review. Intel launches in Q4 of this year, so you know when to expect one...