

Intel's Haswell CPU Microarchitecture

Jyoti Yadav¹, Amandeep Singh², Asif³

Abstract- This paper focuses mainly on the architecture of Intel's 4th generation processors Haswell. The main aim is the extensive study of the architecture, instruction set, memory hierarchy and performance of the Haswell architecture. It also presents a comparison between Intel's previous processors and Haswell processor architecture. This research paper talks about Haswell architecture's battery life, graphics and security.

Index Terms- processors, instruction set, memory hierarchy, graphics.

I. INTRODUCTION

Haswell is the first family of SoCs that have been tailored to take advantage of Intel's 22nm FinFET process technology. While Ivy Bridge is also 22nm, Intel's circuit design team sacrificed power and performance in favour of a swift migration to a process with a radically new transistor architecture.

The Haswell family features a *new CPU core, new graphics and substantial changes to the platform in terms of memory and power delivery and power management*. All of these areas are significant from a technical and economic perspective and interact in various ways. However, the Haswell family represents a menu of options that are available for SoCs tailored to certain markets. Not every product requires graphics (e.g. servers), nor is a new power architecture desirable for cost optimized products (e.g. desktops). Architects will pick and choose from the menu of options, based on a variety of technical and business factors.

The heart of the Haswell family is the eponymous CPU. The Haswell CPU core pushes beyond the PC market into new areas, such as the high-end of the emerging tablet market. Haswell SoCs are aimed at 10W, potentially with further power reductions in the future. The 22nm node enables this wider range, but Haswell's design and architecture play critical roles in fully exploiting the benefits of the new process technology.

The Haswell CPU boasts a huge number of architectural enhancements, with four extensions that touch every aspect of the x86 instruction set architecture (ISA). AVX2 brings integer SIMD to 256-bit vectors, and adds a gather instruction for sparse memory accesses. The fused multiply-add extensions improve performance for floating point

(FP) workloads, such as scientific computing, and nicely synergize with the new gather instructions. A small number of bit manipulation instructions aid cryptography, networking and certain search operations. Last, Intel has introduced TSX, or transactional memory, an incredibly powerful programming model for concurrency and multi-threaded programming. TSX improves performance and efficiency of software by better utilizing the underlying multi-core hardware.

Intel's design philosophy emphasizes superb single core performance with low power. The new Haswell core achieves even higher performance than Sandy Bridge. The improvements in Haswell are concentrated in the out-of-order scheduling, execution units and especially the memory hierarchy. It is a testament to the excellent front-end in Sandy Bridge that relatively few changes were necessary. The Haswell microarchitecture is a dual-threaded, out-of-order microprocessor that is capable of decoding 5 instructions, issuing 4 fused uops (micro operations) and dispatching 8 uops each cycle. The Haswell core is the basis of Intel's upcoming generation of SoCs and will be used from tablets to servers, competing with AMD and a variety of ARM-based SoC vendors.

II. HASWELL INSTRUCTION SET AND FRONT-END

Haswell introduces a huge number of new instructions for the x86 ISA, that fall into four general families. The first is AVX2, which promotes integer SIMD instructions from 128-bits wide in SSE to 256-bits wide. The original AVX was a 256-bit extension using the YMM registers, but largely for floating point instructions. AVX2 is the complement and brings integer SIMD to the full YMM registers, along with some enhancements for 128-bit operation. AVX2 also adds more robust and generalized support for vector permutes and shifts. Perhaps more importantly, AVX2 includes 16 new gather instructions, loads that can fetch 4 or 8 non-contiguous data elements using special vector addressing for both integer and floating point (FP) SIMD. Gather is crucial for wider SIMD and substantially simplifies vectorizing code. Note that AVX2 does not include scatter instructions (i.e., vector addressed stores), because of complications

with the x86 memory ordering model and the load/store buffers.

While AVX2 emphasizes integer SIMD, Haswell has huge benefits for floating point code. In addition to gather, Intel's Fused Multiply Add (FMA) includes 36 FP instructions for performing 256-bit computations and 60 instructions for 128-bit vectors. As announced in early 2008, Intel's FMA was originally architected for 4-operand instructions. However, the 22nm Ivy Bridge can perform register move instructions in the front-end through register renaming tricks, without issuing any uops. Intel's architects determined that MOV elimination with FMA3 provides about the same performance as FMA4, but using denser and easier to decode instructions; hence the abrupt about face in late 2008.

The third extension is 15 scalar bit manipulation instructions (known as BMI) that operate on general integer registers. These instructions fall into three general areas: bit field manipulations such as insert, shift and extract; bit counting such as leading zero count; and arbitrary precision integer multiply and rotation. The latter is particularly useful for cryptography. As an aside, Haswell also adds a big-endian move instruction (MOVBE) that can convert to and from traditional x86 little-endian format. MOVBE was introduced for Intel's Atom, and is quite useful for embedded applications that deal with storage and networking, since TCP/IP is big-endian.

The last and most powerful of Intel's ISA extensions is TSX, which has been extensively discussed in a previous article on Haswell's transactional memory. In short, TSX enables programmers to write parallel code that focuses on using synchronization for correctness, while the hardware optimizes the execution for performance and concurrency. Hardware Lock Elision (HLE) transparently provides the performance and throughput of fine-grained locking, even when programmers use coarse-grained locks. Most importantly, the hint prefixes are compatible with older processors.

Restricted Transactional Memory (RTM) is an entirely new programmer interface that provides transactional memory to x86 developers. TM is far more useful than traditional lock-based synchronization, because transactions can protect more complex data structures and be composed across functions, modules and even applications. However, it does require linking new libraries

using RTM and possibly rewriting software to get the full benefits.

Both variants of TSX are tracked at 64B cache line granularity. Excessive conflicts due to transaction limits, false sharing, or data races can actually harm performance, so developers must judiciously adopt TSX. However, future implementations will most likely have fewer conflicts and be more flexible.

Of these new instructions, the vast majority are simple instructions that decode into a single uop. However, the more complex ones such as gather and the TSX commit and abort are micro coded.

Fundamentally, x86 is quite similar to RISC architectures in a number of dimensions. ALU operations are largely the same; there are only so many ways to do addition, subtraction and multiplication. However, the front-end is quite different and one of the most challenging aspects of modern x86 CPUs. The instruction caches are kept coherent with data caches, and the variable length instructions make decoding quite complex. x86 instructions range in size from 1-15 bytes, with length-changing prefixes, inconsistent operand positions and complex micro coded instructions. Since the P6, these instructions have been transformed into more tractable fixed length uops that can be tracked by an out-of-order core. As with all architectures, the instruction stream is frequently interrupted by control flow such as conditional branches, jumps, calls and returns, which potentially redirect the instruction fetching and introduce bubbles into the pipeline.

Sandy Bridge made tremendous strides in improving the front-end and ensuring the smooth delivery of uops to the rest of the pipeline. The biggest improvement was a uop cache that essentially acts as an L0 instruction cache, but contains fixed length decoded uops. The uop cache is virtually addressed and included in the L1 instruction cache. Hitting in the uop cache has several benefits, including reducing the pipeline length by eliminating power hungry instruction decoding stages and enabling an effective throughput of 32B of instructions per cycle. For newer SIMD instructions, the 16B fetch limit was problematic, so the uop cache synergizes nicely with extensions such as AVX.

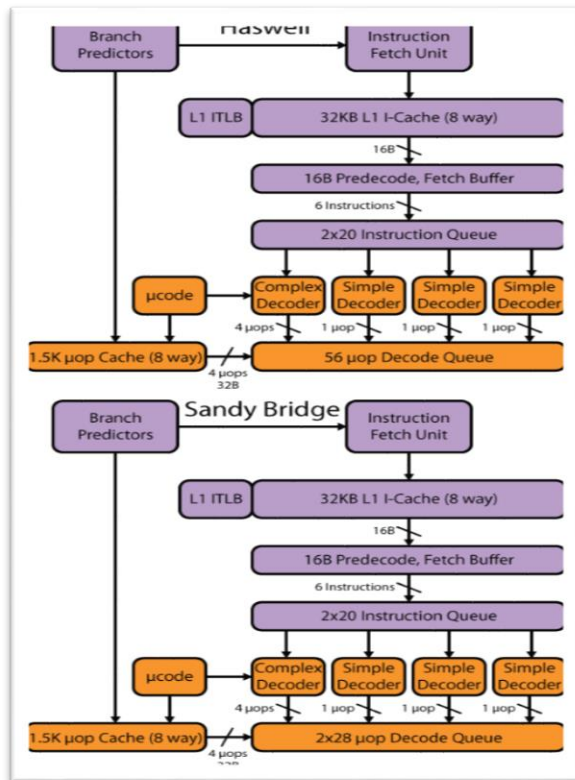


Fig 1. Haswell and Sandy Bridge Front-end

The most significant difference in Haswell's front-end is undoubtedly support for the various instruction set extensions outlined above. At a high level, instruction fetch and decode microarchitecture is largely similar to Sandy Bridge, as shown in Figure 1. There are a number of subtle enhancements, but the concepts and many of the details are the same.

As to be expected, the branch prediction for Haswell has improved. Unfortunately, Intel was unwilling to share the details or results of these optimizations. The instruction cache is still 8-way associative, 32KB and dynamically shared by the two threads. The instruction TLBs are also unchanged, with 128 entries for 4KB page translations; the 4KB translations are statically partitioned between the two threads and 4-way associative. The 2MB page array is 8 entries, fully associative and replicated for each thread. Instruction fetching from the instruction cache continues to be 16B per cycle. The fetched instructions are deposited into a 20 entry instruction queue that is replicated for each thread, in both Sandy Bridge and Haswell.

The Haswell instruction cache was optimized for handling misses faster. Speculative ITLB and cache accesses are supported with better timing to

improve the benefits of prefetching, and the cache controller is much more efficient about handling instruction cache misses in parallel.

As with previous generations, the decoding is performed by a complex decoder that emits 1-4 fused uops and three simple decoders that emit a single fused uop in parallel. Alternatively, instructions requiring more than 4 uops are handled by microcode and block the conventional decoders. Macro-fusion can combine adjacent compare and branch instructions into a single uop, improving the potential throughput to 5 instructions per cycle. The decoding also contains the stack engine, which resolves push/pop and call/return pairs without sending uops further down the pipeline.

The Haswell uop cache is the same size and organization as in Sandy Bridge. The uop cache lines hold up to 6 uops, and the cache is organized into 32 sets of 8 cache lines (i.e., 8 way associative). A 32B window of fetched x86 instructions can map to 3 lines within a single way. Hits in the uop cache can deliver 4 uops/cycle and those 4 uops can correspond to 32B of instructions, whereas the traditional front-end cannot process more than 16B/cycle. For performance, the uop cache can hold micro coded instructions as a pointer to microcode, but partial hits are not supported. As with the instruction cache, the decoded uop cache is shared by the active threads. One difference in Haswell's decoding path is the uop queue, which receives uops from the decoders or uop cache and also functions as a loop cache. In Sandy Bridge, the 28 entry uop queue was replicated for each thread. However, in Ivy Bridge the uop queue was combined into a single 56 entry structure that is statically partitioned when two threads are active. The distinction is that when a single thread is executing on Ivy Bridge or Haswell, the entire 56 entry uop buffer is available for loop caching and queuing, making better use of the available resources.

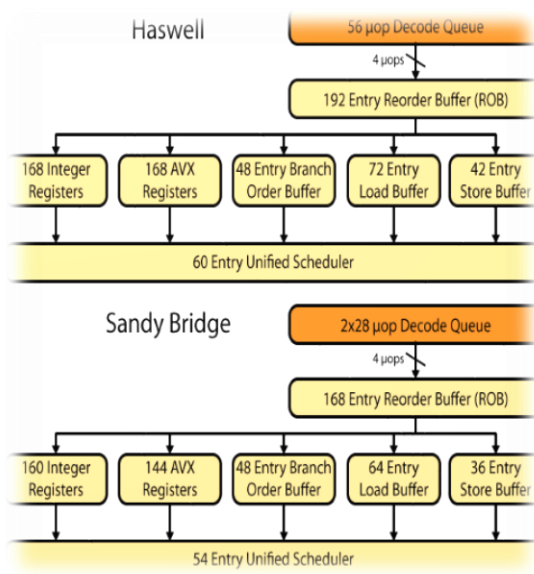
III. OUT-OF-ORDER SCHEDULING

Haswell's out-of-order execution is where the microarchitecture becomes quite interesting and many changes are visible. Haswell is substantially wider than Sandy Bridge with more resources for dynamic scheduling.

The first part of out-of-order execution is renaming. The renamer will map architectural source and destination x86 registers onto the underlying physical register files (PRFs) and allocates other

resources, such as load, store and branch buffer entries and scheduler entries. Lastly, uops are bound to particular ports for downstream execution.

The renamer can take 4 fused uops out of the uop queue for a single thread, allocating the appropriate resources and renaming registers to eliminate false dependencies. Crucially, these 4 fused uops can map to more than just 4 execution pipelines. For example, 4 fused load+execute uops might map to 8 actual uops, 4 loads and 4 dependent ALU operations.



Haswell and Sandy Bridge Out-of-Order Scheduling

Unlike Sandy Bridge, the renamer in Haswell and Ivy Bridge does not have to handle all register to register move uops. The front-end was enhanced to handle certain register move uops, which saves resources in the actual out-of-order execution by removing these uops altogether.

The Haswell and Sandy Bridge core has unified integer and vector renaming, scheduling and execution resources. Most out-of-order resources are partitioned between two active threads, so that if one thread stalls, the other can continue to make substantial forward progress. In contrast, AMD's Bulldozer splits the vector and integer pipelines. Each Bulldozer module includes two complete integer cores, but shares a single large vector core between the two. Conceptually, Bulldozer is dynamically sharing the floating point and vector unit, while having dedicated integer cores.

The most performance critical resources in Haswell have all been expanded. The ROB contains status information about uops and has grown from 168

uops to 192, increasing the out-of-order window by around 15%. Each fused uop occupies a single ROB entry, so the Haswell scheduling window is effectively over 300 operations considering fused loads and stores. The ROB is statically split between two threads, whereas other structures are dynamically shared.

The physical register files hold the actual input and output operands for uops. The integer PRF added a modest 8 registers, bringing the total to 168. Given that AVX2 was a major change to Haswell, it should be no surprise that the number of 256-bit AVX registers grew substantially to accommodate the new integer SIMD instructions. Haswell features 24 extra physical registers for renaming YMM and XMM architectural registers. The branch order buffer, which is used to rollback to known good architectural state in the case of a misprediction is still 48 entries, as with Sandy Bridge. The load and store buffers, which are necessary for any memory accesses have grown by 8 and 6 entries respectively, bringing the total to 72 loads and 42 stores in-flight.

Unlike AMD's Bulldozer, Haswell continues to use a unified scheduler that holds all different types of uops. The scheduler in Haswell is now 60 entries, up from 54 in Sandy Bridge, those entries are dynamically shared between the active threads. The scheduler holds uops that are waiting to execute due to resource or operand constraints. Once ready, uops are issued to the execution units through dispatch ports. While fused uops occupy a single entry in the ROB, execution ports can handle a single un-fused uop. So a fused load+ALU uop will occupy two ports to execute.

Haswell and Sandy Bridge both retire upto 4 fused uops/cycle, once all the constituent uops have been successfully executed. Retirement occurs in-order and clears out resources such as the ROB, physical register files and branch order buffer.

Two of the new instruction set extensions place new burdens on the out-of-order machine. TSX creates a new class of potential pipeline flushes. As we predicted in the earlier article on Haswell's TM, when a transaction aborts, the pipeline is cleared and the architectural state is rolled back. This looks largely similar to a branch misprediction.

TSX also supports multiple nested transactions, which requires hardware resources to track and disambiguate different levels of nesting. Specifically, Haswell can have 7 nested transactions in flight. Any subsequent transactions

will abort as soon as they begin. It is important to note that this limit is micro architectural and may increase in future generations. Presumably this is linked to some hardware structure with 7 speculative entries and the last entry is for the rollback state.

Gather instructions are micro coded and introduce additional complexity to the microarchitecture. As to be expected, Intel’s approach to gather emphasizes simplicity and correctness. This leaves performance on the table relative to a more sophisticated implementation, but is less likely to result in project delays due to risky design choices. The number of uops executed by gather instruction depend on the number of elements. Each element is fetched by a load uop that consumes a load buffer entry, while ALU uops calculate the vector addressing and merge the gathered data into a single register. The uops that execute a gather have full access to the hardware and semantics that are quite different from conventional x86 load and ALU instructions, so the implementation of gather is more efficient than what a programmer could create.

IV. HASWELL EXECUTION UNITS

The execution units in Haswell are tremendously improved over Sandy Bridge, particularly to support AVX2 and the new FMA. Haswell adds an integer dispatch port and a new memory port, bringing the execution to eight uops/cycle. But the biggest changes are to the vector execution units. On the integer SIMD side, the hardware has been extended to single cycle 256-bit execution. For floating point vectors, the big change is 256-bit fused multiply add units for two of the execution ports. As a result, the theoretical peak performance for Haswell is more than double that of Sandy Bridge.

Every cycle, up to eight uops are sent from the unified scheduler to the dispatch ports. As shown in Figure 3, computational uops are dispatched to ports 0, 1, 5, and 6 and executed on the associated execution units. The execution units include three types: integer, SIMD integer, and FP (both scalar and SIMD).

Port 6 on Haswell is a new scalar integer port. It only accesses the integer registers and handles standard ALU (Arithmetic Logic Unit) operations, including shifts and branches that were previously on port 5 (in Sandy Bridge). One of the advantages of the new integer port is that it can handle many

instructions while the SIMD dispatch ports are fully utilized.

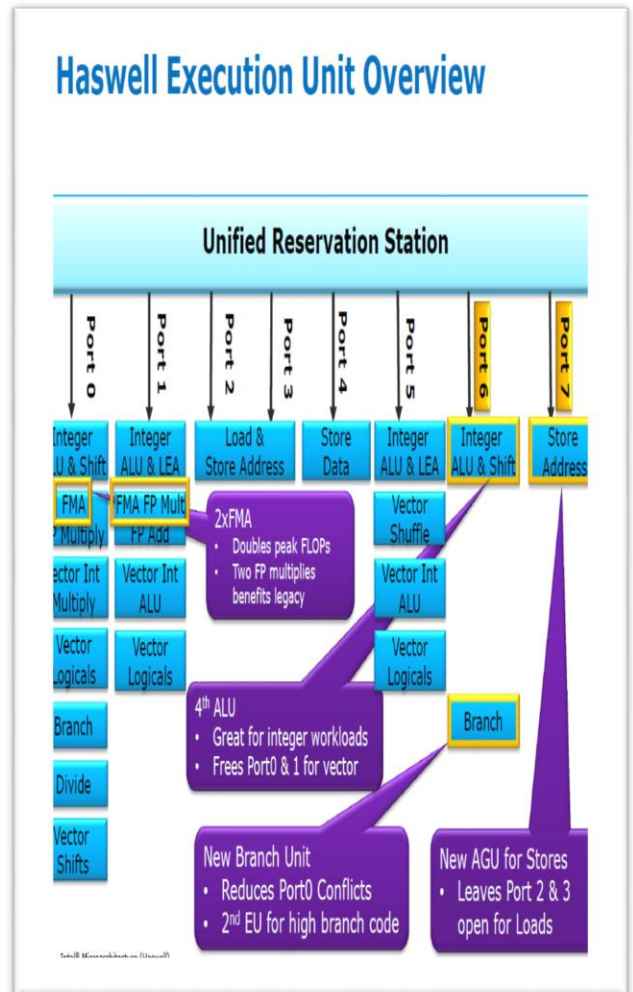


Figure 3

V. HASWELL MEMORY HIERARCHY

The memory hierarchy for Haswell is probably the biggest departure from the previous generation. The cache bandwidth doubled in tandem with an increase in FLOP/s from the new FMA units. Moreover, the whole memory system has been enhanced to support gather instructions and transactional memory.

Memory accesses start by allocating entries in the load and store buffers, which can track more than 100 uops, statically split between two threads. For Sandy Bridge, ports 2 and 3 calculated addresses, with port 4 for writing data into the L1 data cache. The new port 7 on Haswell handles address generation for stores. As a result, Haswell can now sustain two loads and one store per cycle under nearly any circumstances.

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			

Comparison between Intel Sandy Bridge and Intel Haswell Memory Hierarchy

Once an address has been calculated by the Address Generation Unit (AGU), the uop will probe the translation look-aside buffers (TLBs). The L1 DTLB in Haswell is the same organization as in Sandy Bridge. However, there is a third port on the DTLB to accommodate the new store AGU on port 7. Misses in the L1 DTLB are serviced by the unified L2 TLB, which has been substantially improved with support for 2MB pages and twice the number of entries.

Similarly, the L1 data cache in Haswell is the same size and latency (minimum of four cycles), but with a third more bandwidth. The data cache can sustain two 256-bit loads and a 256-bit store every cycle, for 96B/cycle compared with 48B/cycle for Sandy Bridge. Moreover, the data cache in Sandy Bridge was banked, meaning that conflicts could potentially reduce the actual bandwidth. Turning to Haswell's L2 cache, the capacity, organization, and latency is the same, but the bandwidth has also doubled. A full 64B cache line can be read each cycle.

While the organization of the caches was largely unchanged, the capabilities are substantially greater in Haswell since the caches have been designed for

TSX. As speculated, Haswell's transactional memory uses the L1 to store transaction data (either for Hardware Lock Elision or Restricted Transactional Memory). Transactions where the data fits in the L1D cache should be able to execute successfully. From a practical standpoint, this means that the L1D cache contains a bit of extra meta-data to track whether cache lines have been read or written to detect any conflicts.

While the closest levels of the memory hierarchy have been significantly improved, Haswell's system architecture has also been enhanced. The tags for the Last Level Cache (LLC) have been replicated, with one copy for reading data (at the same 32B/cycle) and another for prefetching and coherency requests. The write throughput for the memory controller is also significantly better due to larger write buffers for DRAM accesses and better scheduling algorithms.

To reduce power, the ring and LLC are on a separate frequency domain from the CPU cores. This means that the CPUs can enter in a low-power state, while the ring and LLC run at full throttle to feed the GPU. For many graphically intense workloads, this can reduce the power consumption substantially.

While Intel has demonstrated substantial improvements in idle and active power for Haswell, there was not sufficient detail for a comprehensive discussion. It's likely that this information will only be available when products come to market, since power management is implementation-specific. The most interesting innovation is the new S0ix states for Haswell, which reportedly brings tablet-like power characteristics (e.g., always-on) to the PC through undisclosed mechanisms.

So, in short here are the key 4th generation Intel processor (based on haswell architecture) features:

The new processor builds on the processor graphics architecture first introduced in 2nd gen Intel® Core™ processors. While they were built with the 32 nm manufacturing process, both 3rd and 4th generation processors are based on the 22 nm technology. The following paragraphs describe the key differences between the 3rd and 4th gen processors.

1. First ever System on Chip (SoC) for a PC :

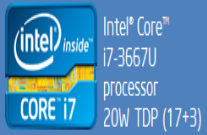

The 4th gen Intel® Core™ processor is the first ever SoC for a PC. System on Chip, or SoC, integrates all the major building blocks for a system onto a single chip. With CPU, Graphics, Memory, and connectivity in one package, this innovative

modular design provides the flexibility to package a compelling processor graphics solution for multiple form factors.

2.Enhanced battery life:

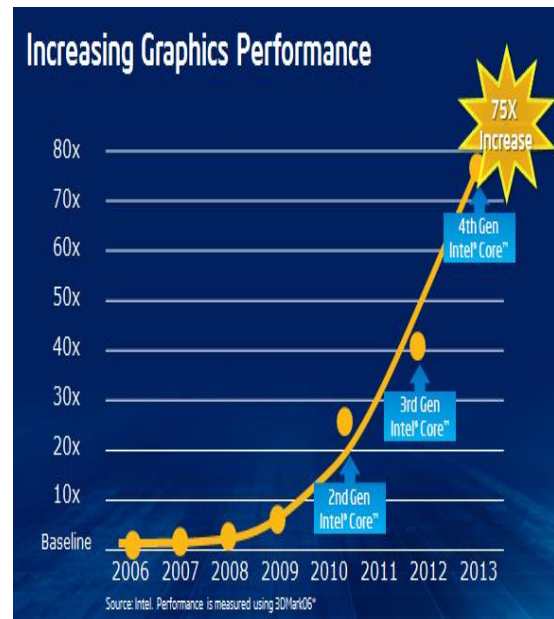
The 4th gen processor provides up to 9.1 hours of HD video viewing compared to 6 hours on the 3rd gen one. The latest processor also provides 10-13 days of standby power (with refreshed email and social media notifications) compared to 4.5 days of standby power on 3rd generation processors.

Here is a Battery life comparison between 3rd Generation and 4th generation Intel® Core™ Processors

	3rd Generation Core™ Processor	4th Generation Core™ Processor
	 <p>Intel® Core™ i7-3667U processor 20W TDP (17+3)</p>	 <p>Intel® Core™ i7-4650U processor 15W TDP</p>
HD Video Playback	6.0 hours	9.1 hours
MobileMark* 2012	6.1 hours	8.3 hours
Standby with fresh data	4.5 days	10-13 days
Performance-MobileMark* 2012	138	155

Battery life is calculated based on measured platform power assuming a 50 Watt-Hour battery capacity.

Intel® Iris™ Graphics allows you to play the most graphic intensive games without the need for an additional graphics card. The graphics performance on the 4th gen processor nearly doubles the performance relative to the previous generation of Intel® HD Graphics.



Intel® AVX 2.0

Intel® Advanced Vector Extensions (Intel® AVX) 2.0 is a 256-bit instruction set extension to Intel® Streaming SIMD Extensions (Intel® SSE). Intel AVX 2.0 build on version 1.0 and provides features like Fully Pipelined Fused Multiply Add on two ports thus providing twice the floating point performance for multiply-add workloads, 256-bit integer SIMD operations compared to older 128-bit gather operations and bit manipulation instructions. These capabilities enhance usages such as face detection, pro-imaging, high performance computing, consumer video and imaging, increased vectorization, and other advanced video processing capabilities.

3. Intel Iris Graphics Extensions to DirectX API
An added feature with 4th generation processor graphics is the API set for DirectX extensions. Two APIs are available that provide for pixel synchronization and instant access. Pixel synchronization lets you effectively read/modify/write per-pixel data, which makes the tasks of programmable blending and order independent transparency (OIT) more efficient. Instant access lets both CPU and GPU access the same memory for mapping and rendering. These APIs work on DirectX 11 and above.

4. Security:
Ultrabook systems with 4th gen processors come with enhanced security features like Intel® Platform Trust Technology, Intel® Insider, and Intel® Anti-Theft technology, the processors also feature Intel® Identity Protection Technology,

which provides identity protection and fraud deterrence.

VI. CONCLUSIONS AND ANALYSIS

There is much to like in Haswell: a bevy of new instructions, a more powerful microarchitecture, and lower power. For vector friendly workloads, AVX2 brings integer SIMD to 256-bit width—dead even with FP vectors—while doubling the number of FP operations through fused multiply-add (FMA). Crucially, Haswell also includes gather instructions to fetch non-contiguous data from memory, which makes it easier for compilers and programmers to use the x86 SIMD extensions.

Intel's transactional memory extensions have a smaller impact on performance but have more potential in the long run. The Hardware Lock Elision separates correctness from performance. Software can use simpler techniques such as coarse-grained locks, and the hardware will automatically optimize for performance. Alternatively, Restricted Transactional Memory is an entirely new way to write concurrent code that is far easier and more intuitive, with higher performance to boot.

The Haswell microarchitecture has a modestly larger out-of-order window, with a 33 percent increase in dispatch ports and execution resources. Compared to previous generations, the theoretical FLOPs and integer operations have doubled for each core, primarily due to wider vectors. More significantly, the cache hierarchy can sustain twice the bandwidth, and it has fewer utilization bottlenecks.

As tablets have become more robust and capable, in many respects Intel's competitive focus has shifted away from AMD to the ARM ecosystem: Qualcomm, Samsung, NVidia, and others. Haswell will be the first high performance x86 core that can really fit in a tablet, albeit in high-powered models around the 10W mark rather than the 4W devices. For consumers, Haswell will offer a heady combination of Windows 8 (and compatibility with the x86 software base) with excellent performance and tablet-like power characteristics. Compared to AMD or ARM-based solutions, the performance will be dramatically higher, so the biggest question will be power efficiency. There, Intel is claiming some impressive advances while keeping the details close, particularly about *actual* products.

In summary, Haswell is a superb new architecture that will carry Intel into new markets and a new era

of competition, not only from AMD, but also the ARM ecosystem. Ultimately, products will reveal the performance and efficiency advantages of the Haswell family, but the architecture looks quite promising, a testament to Intel's design team.

REFERENCES

- [1] Haswell (Microarchitecture), "en.wikipedia.org/wiki/Haswell_(microarchitecture)"
- [2] David Kanter Principal Analyst and Editor-in-Chief, Real World Tech, "http://www.realworldtech.com/haswell-cpu/"
- [3] A look at Haswell, "http://arstechnica.com/gadgets/2013/05/a-look-at-haswell/1/"
- [4] Intel key features , "https://software.intel.com/en-us/articles/an-introduction-to-the-intel-4th-generation-core-processor "
- [5] Intel new generation Haswell processors : "What you need to know", "http://www.cnet.com/news/intels-new-fourth-gen-haswell-processors-what-you-need-to-know-faq/"
- [6] The Intel Haswell Review, "http://www.anandtech.com/show/8426/the-intel-haswell-e-cpu-review-core-i7-5960x-i7-5930k-i7-5820k-tested"
- [7] "http://www.realworldtech.com/haswell-cpu/"
- [8] "http://www.pcworld.com/article/2600325/intel-turns-its-attention-to-desktop-performance-unveils-8-core-haswell-e-processor.html"
- [9] Intel Haswell PC, "https://www.pcspecialist.co.uk/computers/intel-haswell-pc/"