From IA-32 to Core: Processor Innovations at Intel

Introduction

The personal computer, laptop, workstation and server market is dominated by computers running central processing units (CPUs) designed and manufactured by Intel Corp. ARM RISC processors dominate the embedded, mobile and tablet markets. Beginning with the release of its Athlon 64 and Operton processors, Advanced Micro Devices (AMD) began to offer serious alternatives for consumers across all platforms; AMD's technical and competitive lead continued into 2005 with the release of its dual core Athlon 64 X2 and Operton dual core processors. However, all that changed in the middle of 2006 when Intel released its Core architecture which has dominated to this day. This paper provides an overview of the key technical innovations of the Core architecture; in so doing, it introduces the student to concepts in computer science and technical design features that underpin the Core architecture. First, we briefly examine the historical context and innovations that laid the foundation for Intel's Core processors.

A Short History of Intel Processors Leading to the Pentium

The 8086, Intel's first generation 16-bit processor was introduced in the late 1970s. Its initial business uses were limited to dedicated word processing machines and low-end mini-computers. However, in 1981 IBM used the Intel 8086 as the CPU of its personal computer. In 1982, Compaq adopted the CPU for use in its new Desk Pro model, which consisted of cloned versions of IBM/XT range of personal computers. Intel enhanced the 8086 with the introduction of the 80186 and 80188 CPUs, and added a math coprocessor called a floating point unit (FPU) in the 8087 Intel coprocessor.

A floating point unit carries out operations on floating point numbers (i.e. where the numbers appearing after the decimal point are not fixed, these leads to more precision in mathematical computations) when conducting high precision arithmetic operations (such as addition and multiplication), and also exponential or trigonometric calculations as well (such as square roots or cosines).

Intel's second generation 80286 appeared in 1982 and formed the core of the first powerful PC, the IBM PC /AT. The Intel 80287 provided FPU co-processing functionality in high-end platforms. While the i286 provided protected mode operation and up to 16 MB of RAM, Intel's third generation CPU, the i386 was its first 32-bit CPU. This came with 16-33 MHz core processor and system bus speeds and had an accompanying 80387 FPU. A variant of the 386, the SX was the first Intel chip to have an internal L1 cache. Intel's forth generation family appeared in 1989 with the release of the i486. This 32-bit CPU had 8KB of L1 cache and a built-in FPU. While initially running at speeds of between 20-50 MHz internally and externally (i.e. core CPU and bus speeds), the release of i486 DX/2 in 1992 saw the bus speeds multiplied by 2 for internal core CPU operation (25/50, 33/66 and 40/80MHz). These speeds were tripled in 1994 with the release of the i486 DX/4 (a slight of hand by Intel, as the CPUs ran at 25/75, 33/100 and 40/120 MHz): importantly, the 486's L1 cache was doubled to 16 KB in and 8KB instruction + 8KB data configuration.

The Pentium P5 series was first shipped in 1993 and had 3.1 million transistors. It used a 5 Volt to power its core and I/O logic, had a 2x8kb L1 cache, and operated at 50, 60 and 66 MHz. The system bus also operated at these speeds. The Pentium (P54C) was released in 1994 with a 3.3 Volts supply for core and I/O logic. It was also the first to use a multiplier to give processor speeds of 75, 90,100,120,133, 150, 166 and 200 MHz. The last version of the P5 generation was the Pentium MMX (P55C). This had 4.1 million transistors, fit Socket 7, and had a 2 x 16 KB L1 cache with improved branch prediction logic. It operated at 2.8 V for its core logic and 3.3V for I/O logic. Its 60 and 66 MHz system clock speed was multiplied on board the CPU to give between 120-300MHz CPU clock speeds.

A Primer on the Technical Innovations that Help Processors Execute More Instructions per Clock Cycle

The traditional CPU architecture uses a single pipeline to execute instructions; this is called a Von Neuman pipeline and consists of the following operations on the instruction: Fetch (Load), Decode, Operand Fetch (Load), Execute, and Retire (Store)—this is called the instruction cycle. The instruction to be fetched and decoded etc. is presented to the processor in a language it understands, i.e. its instruction set—different processor architectures have different instruction sets. E.g. the Intel x86, which includes the P series (586) and by extension Intel Core, is based on the Intel Architecture 32 bit model (IA-32), while the Itanium is based on a 64 bit Architecture or IA-64, also called the EPIC (Explicitly Parallel Instruction Computing) instruction set. Please note that AMD’s K series is also based on the IA-32, although it has additional instructions for certain functions, nevertheless, it is fully compatible. Also note that the Core and Athlon 64 architectures are simply an extension of the IA-32 and can execute 16-bit 32-bit and 64-bit instructions (although there are minor difference between the instruction sets: e.g. Game developers optimize the performance of their code by taking advantages of AMD’s IA-32 instruction set extensions for its processors). Please also note that Intel’s Core processors have what’s known as the Intel 64 Bit architectures. This is not the same as the IA-64.

A compiler is a computer program that translates text-based source code written C or C++ etc. into the target machine language (i.e. the instruction set): this is also called object code. Several object code modules, including those in libraries, are then linked or merged together by a linker (another computer program) to form the executable binary image.

Instructions are ***fetched*** based on the instruction pointer register, and appear to the CPU as 32 or 64-bit patterns. 32-bit instructions typically contain *opcode,* i.e. a machine specific instruction such *add*, *load* (from RAM or L1 or L2 cache) or *store* (into RAM) and the *operand*, that is address of the data to be processed. The ***decoder*** separates the opcode and operand and passes the operand address to the ***operand fetch*** logic to fetch or load the data from memory i.e. the L1, L2 cache. The operands are then placed into the CPU’s registers. The arithmetic and logic unit (ALU) uses the opcode to perform the desired calculations on the operands in the CPU’s registers—i.e. ***execute*** them. The results of the operations must also be written (or stored) to the caches and then RAM, so that these are also up to date—this is the ***retirement*** stage when the results of the computations are loaded into memory.

Clearly this sequential mode of operation is inefficient as one instruction cycle must finish before another begins. But this need no be so: one obvious answer is to begin the steps of instruction fetching and decoding before the current instruction finishes executing. This is the simplest form of a technique known as instruction pipelining—pipelining makes **Instruction Level Parallelism** (ILP) a reality. Pipelining allows more than one instruction to be executed (i.e. be in the pipeline) at any given time by breaking down the instruction cycle into discrete stages, very much like an assembly line. What this means is that the CPU subunits responsible for processing instructions will always be doing something. In the example in the following table, the result of instruction A is being retired while instruction E is just entering the pipeline.

Time	Stage	Instruction Cycle	Instruction.
	1	Fetch	E
	2	Decode	D
	3	Operand Fetch	C
	4	Execute	B
T5	5	Retire	A

Pipelining does, however, introduce the possibility of situations where the result of the one instruction (i.e. A) is required is to complete the execution of the next (instruction B). Thus the result of instruction A's computation must be in L1, L2 or RAM, but clearly it won't be when the ALU tries to execute B, as A is in the retirement stage. In fact, B should not be able to advance beyond the Operand Fetch stage until A's result is retired—this is termed data dependency conflict. Hence, additional logic circuits must be in place in a processor to check for these dependency conditions and stop an instruction from advancing in the pipeline if this occurs. Clearly this is undesirable, as in an ideal scenario, each stage in the pipeline needs to be occupied for every clock cycle giving the optimum result of having 1 instruction being retired per CPU clock cycle.

The design of Intel CPUs has therefore focused on (1) having as many instructions in the pipeline as possible; (2) having several pipelines operating in parallel, thereby executing 2 or more instructions per clock cycle; and (3) finding a way to execute instructions and opcode out-of-order to overcome the pipeline blockages caused by data dependency conflicts; (4) introducing L1 (and later L2) cache to store executing code and data in the CPU; and (5) increasing the number of cycles per second (i.e. clock frequency measure in MHz or GHz) of the CPU. Of course this presupposes that there is no delay in fetching instructions, opcode and operands, from RAM; however, the challenges of reducing memory latencies in L1, L2 and RAM adds further complexity and ever more innovative solutions, aside from merely clocking the Front Side Bus (FSB) to run at higher speeds, as we shall see.

Memory latency refers to the time it takes RAM or Cache memory to respond to a read request for data from memory. While memory is byte-addressable and the bus to memory relatively slower than the CPU speed, RAM transmits a block of data, typically 32 bytes or more, to the CPU for every memory read.

One major problem facing CPU designers is the nature of the code being executed. Most programmes execute in loops. This makes life easy for the CPU, as instructions can usually be executed sequentially and predictably. There is, however, the not insignificant issue of the tendency of programmes to make changes in their execution paths dynamically and unexpectedly: that is they branch under certain conditions and execute different functions, sub-routines etc., and then, perhaps, return to their original path, or they may enter a whole different area of execution. There are several ramifications here: these concern the need to stop execution of the existing routine, this means abandoning any fetch operation(s), decoding of instructions after the branch, and then fetching the branched code (the time or clock cycles spent fetching this code and data is therefore wasted). In the days before the use of L1 cache, such occurrences did not have any measurable impact on overall performance, if the branched code was in RAM along with the rest of the program. The use of L1 cache by processes or threads executing in a loop helps the CPU run that code faster, as fetches to RAM are avoided, by having the CPU load in instructions from the cache. When there is a branch to another area within the program, that code has to be loaded into L1 and existing code replaced, if there is no free memory available. Thus, making L1 cache bigger helps overcome any degradation in performance by being able to store the branched code along with the previously executing code (to which the branch may return to). Likewise having a relatively large L2 cache with low latency avoids the need to go to RAM. However, programmes were, and still are, typically much larger than available cache, to say nothing of available RAM. These issues are further compounded by the need to cache data also. So how did CPU designers overcome these limitations? They used computer logic to make branch predictions—more about that later.

In an ideal world CPUs would use Reduced Instruction Set Computing (RISC) architectures, where all instructions are the same size, with no need for decoders, and where there are several pipelines in which instructions are executed in parallel. The RISC compiler would flag any data dependencies between instructions for the CPU, allowing it to make appropriate provisions; furthermore, the compiler would also flag potential branches, taking some of the load off of the logic in making branch predictions. Intel moved somewhat in this direction with the Itanium's EPIC architecture. In the IA-32 world, however, complex instructions are decoded into simple instructions called micro-operations (i.e. micro-ops or micro code). However, many instructions are more complex than others and lead to several micro-operations on decode, while others are relatively simple and lead to a single micro-operation on decode. The former instructions are therefore slow to decode while the latter require less clock cycles. It would make sense therefore to have a fast pipeline and a slow pipeline in the CPU, to speed up the execution of simple instructions by preventing them from being blocked in the pipeline by slower more complex instructions. This brings the problem of ensuring that data dependencies and other dependent relationships between micro-ops are dealt with, while also allowing the execution of simple, non-dependent instruction micro-ops out of the order and ahead of others (i.e. bringing in the concept of speculative execution). This minimizes the effects dependency-related blockages in the pipeline by allowing the non-dependent micro-ops to jump ahead of the stalled dependent micro-ops. How did Intel and AMD solve this? They placed a circular memory buffer with an input from the decoder and an output to the execution units (ALUs, inc. FPUs). However, this buffer also has an input from the output of the execution units and another output to the retirement unit. This permits all micro-ops to be retired in the order of the original instructions. In addition to this, and to help speed up the resolution of dependencies, the registers nominated for use by institutions can be renamed (reallocated) to avoid dependency conflicts.

Pentium P5 Innovations

The previous section outlined several possible solutions to the problem of making processors more powerful: this section presents a number of innovations Intel introduced in the P5.

- Superscalar architecture (e.g. instruction-level parallelism): The first Pentium had two integer (U (slow) and V (fast)) and one floating point pipeline. The U and V pipelines contained five stages of instruction/micro-op execution, in addition to the floating point pipeline which had 8 stages. The U and V pipelines were served by two 32 byte prefetch buffers. This allowed overlapping execution of instructions in the pipelines.
- Dynamic branch prediction used the Branch Target Buffer. The Pentium's branch prediction logic helped speed up program execution by anticipating branches and ensuring that branched-to code was available in cache
- An Instruction and a Data Cache each of 8 Kbyte capacity
- A 64 bit system data bus and 32 bit address bus
- Dual processing capability
- On-board Advanced Programmable Interrupt Controller (e.g. managing hardware interrupts)
- The Pentium MMX version contained an additional MMX unit that sped up multimedia and 3D applications. Processing multimedia data involved instructions operating on large volumes of packetized data. Intel proposed a new vector-based approach: *single instruction multiple data*, which could operate on video pixels or Internet audio streams. The MMX unit contained eight new 64 bit registers and executed 57 'simple' hardwired MMX instructions that operated on 4 new data types. To leverage the features of the MMX unit, applications were programmed to include the new instructions.

Scalar processors process one data item per instruction. Vector processors operate on multiple data items simultaneously using only a single instruction. The difference is analogous to the difference between scalar and vector arithmetic. A superscalar processor takes the best of both worlds, as each instruction processes one data item, but there are multiple execution units so that multiple instructions can be processing separate data items at the same time. Other units such as MMX, have a single instruction operate on multiple data.

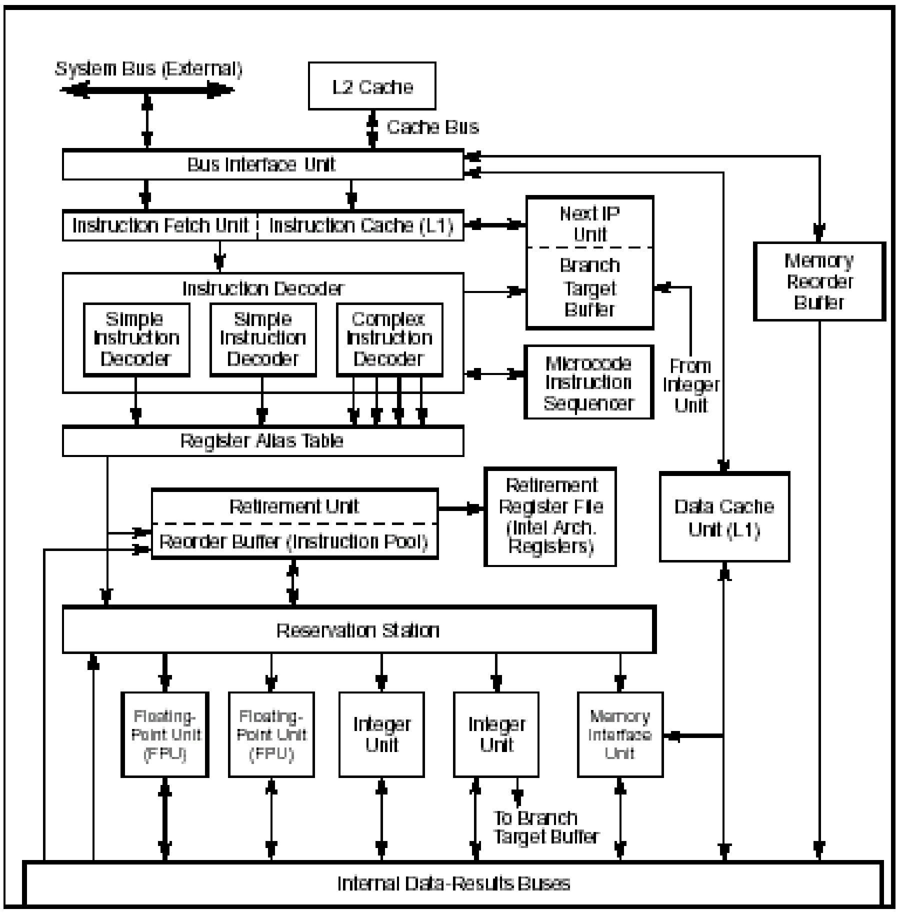
P6 and P7 Processors

Intel's Core architecture is clearly a descendant of the P6 architecture. It contains several of its innovations simply because the Intel's Israel Development Center based the Core architecture on the Pentium M architecture, on which is, in turn, based on the Pentium III, which was the last of the P6 generation processors. This team did not employ innovations in the P7 generation architecture, such as HyperThreading, as used in the Pentium IV.

P6 Processor Architecture

The Pentium Pro was the first processor to incorporate the P6 architecture. The platform was optimized for 32-bit instructions and to accommodate 32-bit operating systems such as Windows NT and Linux. Other members of the P6 family were the Pentium II, the Celeron variants, and the Pentium III. However, aimed as it was at the server market, the Pentium Pro did not incorporate MMX technology, which was present in the other processors in the P6 family. It was expensive to produce, as it included an L2 cache with over 8 million on its substrate (but on a separate die which was piggy-backed on the core); it also had 5.5 million transistors at its core. Its core logic operated at 3.3Volts. The microprocessor was still, however, chiefly CISC in design.

Figure 1 Functional Block Diagram of the Pentium Pro Processor Micro-architecture



The chief features of the Pentium Pro were:

- A partly integrated L2 cache of up to 512 KB (on a specially manufactured SRAM separate die) that was connected via a dedicated ‘backside’ bus that ran at full CPU speed.
- Three 12 staged pipelines
- Speculative execution of instructions

- Out-of-order completion of instructions
- 40 renamed registers
- Dynamic branch prediction
- Multiprocessing with up to 4 Pentium Pros
- New PAE (Physical Address Extension) which featured an increased bus size up from 32 to 36 bits to enable up to 64 Gb of memory to be used. (Please note that the 4 extra bits can address up to 16 memory locations; this gives 4 Gb x 16 = 64 Gb of memory.)

The P6 is three-way superscalar in that its uses parallel processing techniques, which enables the processor, to decode, dispatch, and complete execution of (or retire) three instructions per clock cycle. To handle this level of instruction throughput, the Pentium Pro processor used a decoupled, 12-stage Superpipeline. The key innovations to support this superscalar architecture involved logic circuit support for micro-data flow analysis, out-of-order execution, superior branch prediction, and speculative execution.

Dynamic Execution

In the P6, three instruction decode units worked in parallel to decode object code into smaller operations called “micro-ops” (microcode). These went into an instruction pool, and (when interdependencies don’t prevent) were executed out of order by the five parallel execution units (two integer, two FPU and one memory interface unit). The Retirement Unit retired completed micro-ops in their original program order, taking account of any branches.

The centerpiece of the Pentium Pro processor architecture was an innovative out-of-order execution mechanism called “dynamic execution.” Dynamic execution incorporates three data-processing concepts:

- Deep branch prediction.
- Dynamic data flow analysis.
- Speculative execution.

Branch prediction is a concept found in most mainframe and high-speed RISC microprocessor architectures. It allows the processor to decode instructions beyond branches to keep the instruction pipeline full. In the Pentium Pro processor, the instruction fetch/decode unit used a highly optimized branch prediction algorithm to predict the direction of the instruction stream through multiple levels of branches, procedure calls, and returns

Dynamic data flow analysis involves real-time analysis of the flow of data through the processor to determine data and register dependencies and to detect opportunities for out-of-order instruction execution. The Pentium Pro processor dispatch/execute unit can simultaneously monitor many instructions and execute these instructions in the order that optimizes the use of the processor’s multiple execution units, while maintaining the integrity of the data being operated on. This out-of-order execution keeps the execution units busy even when cache misses and data dependencies among instructions occur.

Speculative execution refers to the processor's ability to execute instructions ahead of the program counter, but ultimately to commit the results in the order of the original instruction stream. To make speculative execution possible, the Pentium Pro processor microarchitecture decoupled the dispatching and executing of instructions from the commitment of results. The processor's dispatch/execute unit used data-flow analysis to execute all available instructions in the instruction pool and store the results in temporary registers. The retirement unit then linearly searched the instruction pool for completed instructions that no longer had data dependencies with other instructions or unresolved branch predictions. When completed instructions were found, the retirement unit committed the results of these instructions to memory and/or the Intel Architecture registers (the processor's eight general-purpose registers and eight floating-point unit data registers) in the order they were originally issued and retired the instructions from the instruction pool.

Through deep branch prediction, dynamic data-flow analysis, and speculative execution, dynamic execution removed the constraint of linear instruction sequencing between the traditional fetch and execute phases of instruction execution. It allowed instructions to be decoded deep into multi-level branches to keep the instruction pipeline full. It promoted out-of-order instruction execution to keep the processor's six instruction execution units running at full capacity. And finally it committed the results of executed instructions in original program order to maintain data integrity and program coherency.

2-, 4- or 8-way set associativity in cache memory, with increasing speed of access and lower latency the higher the value, is similar to saying that if you send 8 people to access and retrieve a book in a library, they will be faster in retrieving the book than by sending 4 or 2 people.

The power of the Pentium Pro processor was further enhanced by its caches: it had the same two on-chip 8-KByte L1 caches as did the Pentium processor, and also had a 256-512 KByte L2 cache that was in the same package as, and closely coupled to, the CPU, using a dedicated 64-bit ("backside") full clock speed bus. The L1 cache was dual ported (2-way set associative), the L2 cache supported up to 4 concurrent accesses (4-way set associative), and the 64-bit external data bus was transaction-oriented, meaning that each access was handled as a separate request and response, with numerous requests allowed while awaiting a response. These parallel features for data access worked with the parallel execution capabilities to provide a "non-blocking" architecture in which the processor was more fully utilized and performance is enhanced.

Pentium III: The Precursor of the Pentium M

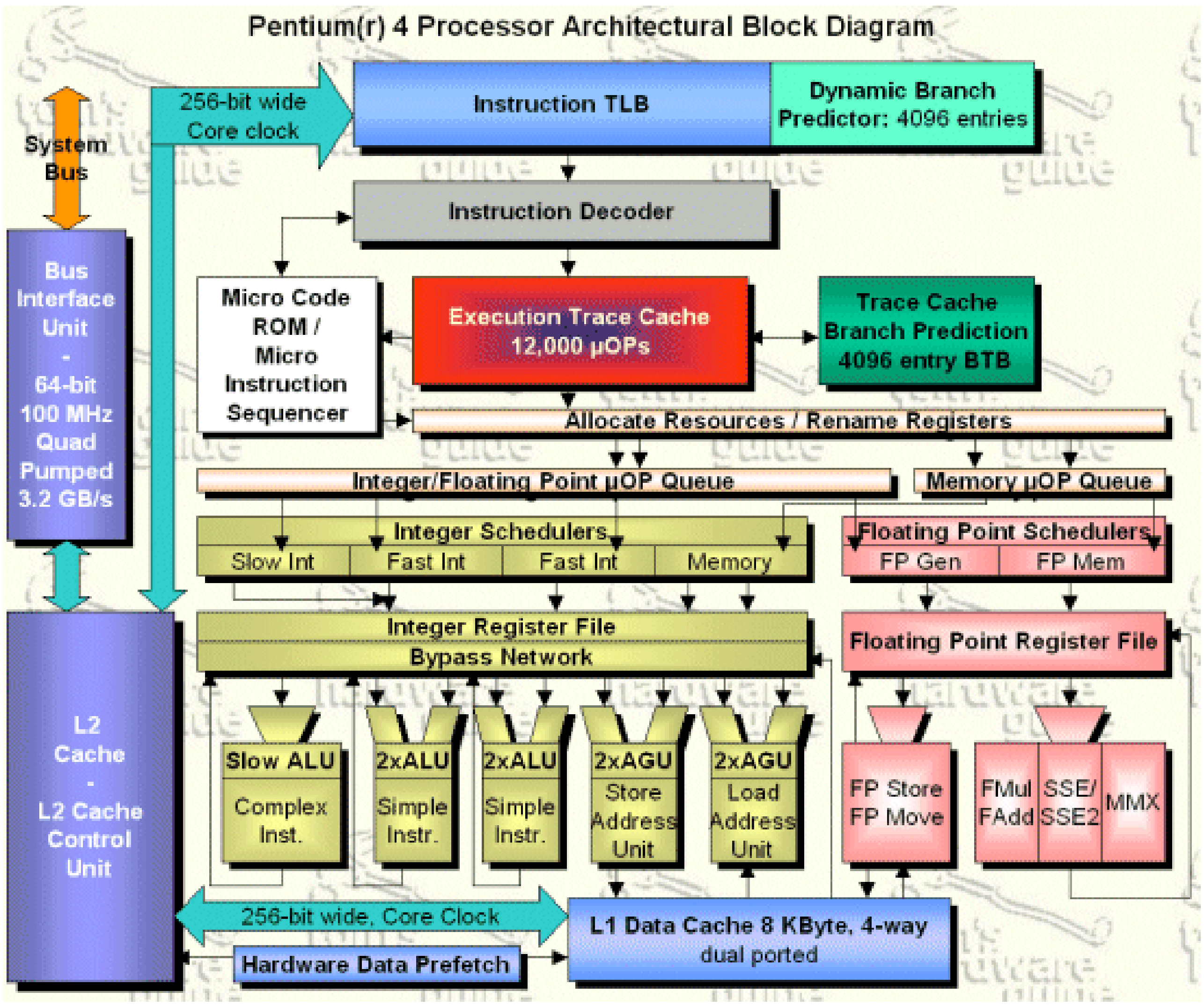
The only significant difference between the Pentium III and its predecessors was the inclusion of 72 MMX instructions, known as the Internet Streaming Single Instruction Multiple Data Extensions (ISSE); these include RISC-based integer and floating point operations. However, like the original MMX instructions, application programmers must include the corresponding extensions if any use is to be made of these instructions. The most controversial and short-lived addition was the CPU ID number which could be used for software licensing and e-commerce. After protest from various sources, Intel disabled it as default, but did not remove it. Depending on the BIOS and motherboard manufacturer, it may remain as such but it can be enabled via the BIOS.

The three variants of Pentium III were the Katami, Coppermine, and Tualatin. Katami first introduced the ISSE (MMX2) as described, with a FSB of 100 MHZ. The Coppermine also introduced Advanced Transfer Cache (ATC) technology in the L2 cache which reduced cache capacity to 256 KB, but saw the cache run at full processor speed: ATC incorporated Data Prefetch Logic that predicted the data required by the cache and loaded it into the L2 cache in advance of it being required by the CPU. Also, the 64-bit Katami cache bus was quadrupled to 256 bits. Coppermine used an 8-way set associative cache, rather than the 4-way set associative cache in the Katami and older Pentiums. Bringing the cache on-die also increased the transistor count to 30 million, from the 10 million on the Katami. Another advance in the Coppermine was Advanced System Buffering (ASB), which simply increased the number of buffers to account for the increased FSB speed of 133 MHz. The Pentium III Tualatin had a reduced die size that allowed it to run at higher speeds, had a 133MHz FSB and had ATC and ASB.

Pentium IV: A Brilliant Technological Dead End?

The release of the Pentium IV in 2000 heralded the seventh generation of Intel microprocessors. The release was brought forward, however, due to the out performance of the Pentium III Coppermine, with its 1 GHz top CPU speed, by AMD Athlon processors. At this time, Intel was not ready to answer the competition through the early release the Pentium III Tualatin, which were designed to break the 1 GHz barrier. (Previous attempts to do so with the Pentium III Coppermine 1.13 GHz met with failure due to design flaws.) Paradoxically, however, Intel was in a position to release the first of the Pentium IV family the Willamette, which ran at 1.3, 1.4 and 1.5 MHz. Worse still, the only Intel chipset available for the Pentium IV could only house the highly expensive RAMBus DRAM. In addition, the early versions of Pentium IV CPU were outperformed by slower AMD Athlons. Nevertheless, the core capability of Intel’s P7 generation processors is that they had the capability to run at ever-higher CPU speeds. Compare, for example, the fact Intel’s P6s began at 120 MHz with the Pentium Pro and ended at over 1.2 GHz, a tenfold increase. The bottom line here is that Intel’s P7 CPUs were forecast to run at speeds of 10 GHz or more. How did Intel achieve this? Through a radical redesign of the Pentium’s core architecture (which by 2006 became a dead design end…for now).

The most ‘visible’ improvement seen on the Pentium IV concerned the Front Side Bus (FSB) speed, which initially operated at 400 MHz as compared to 100 MHz on the Pentium III. The Pentium III had a 64-bit data bus that delivered a data throughput of 1.066 GB (8 bytes* 133= 1.066). The Pentium IV FSB bus is also 64 bits wide, however, in the earlier model the 100 MHz bus speed was ‘quad-pumped’ giving an effective bus speed of 400 MHz and a data transfer rate of 3.2 Gbps. In late 2002, Pentium IV and associated chipsets operated at 133 MHz core bus speed, which was ‘quad-pumped’ to 533 MHz, delivering a throughput of 4.2 Gbps. Thus the first Pentium IV versions exchanged data with the i845 and i850 chipsets faster than any other processor, thereby removing the Pentium III’s most significant bottleneck.


Pentium(r) 4 Processor Architectural Block Diagram
System Bus
256-bit wide Core clock
Instruction TLB
Dynamic Branch Predictor: 4096 entries
Instruction Decoder
Bus Interface Unit - 64-bit 100 MHz Quad Pumped 3.2 GB/s
Micro Code ROM / Micro Instruction Sequencer
Execution Trace Cache 12,000 µOPs
Trace Cache Branch Prediction 4096 entry BTB
Allocate Resources / Rename Registers
Integer/Floating Point µOP Queue
Memory µOP Queue
Integer Schedulers
Slow Int
Fast Int
Fast Int
Memory
Floating Point Schedulers
FP Gen
FP Mem
Integer Register File
Bypass Network
Floating Point Register File
L2 Cache - L2 Cache Control Unit
Slow ALU Complex Inst.
2xALU Simple Instr.
2xALU Simple Instr.
2xAGU Store Address Unit
2xAGU Load Address Unit
FP Store FP Move
FMul FAdd
SSE/ SSE2
MMX
256-bit wide, Core Clock
L1 Data Cache 8 KByte, 4-way dual ported
Hardware Data Prefetch

Advanced Transfer Cache L2

The first major improvement of the Pentium was the integration of the L2 cache and the evolution of the Advanced Transfer Cache introduced in the Pentium III Coppermine, which had just 256 KB of L2 Cache. The first Pentium IV, the Willamette, had a similar sized cache, but could transfer data at **48 GB/s** at a CPU clock speed of 1.5 GHz into the CPU's core logic. In comparison, the Pentium III Coppermine could only transfer **16 GB/s** at 1 GHz to its L1 Instruction Cache. In addition, the Pentium IV L2 cache had 128-byte cache lines, which were divided in two 64-byte segments. Significantly, when the Pentium IV fetched data from the RAM, it does so in 64 byte burst transfers. However, if just four bytes (32 bits) are required, this block transfer becomes inefficient. Consequently, the cache has Advanced Data Prefetch Logic that predicts the data required by the cache and loads it into the L2 cache in advance. The Pentium IV's hardware prefetch

The Translation Lookaside Buffer stores virtual address (0-64 GB) to real, physical memory addresses (0-4GB) in the CPU. When a the virtual address first used to locate an instruction or data it is translated to the physical address in RAM at which the instruction or data actually resides. As programs operate in loops, this would happen repeatedly, wasting clock cycles, unless the translated addresses were stored: the TLB fulfills this role.

logic significantly accelerates the execution of processes that operate on large data arrays. The read latency (the time it takes the cache to transfer data into the pipeline) of Pentium IV's L2-cache is 7 clock pulses. However, its connection to the core logic (the Translation Lookaside Buffer (TLB) in this case, there is no I-Cache in the Pentium IV) is 256-bit wide and clocked the full processor speed. The second member of the Pentium IV family was the Northwood, which had a 512 KB L2 Cache running at the processor's clock speed.

The 90nm process-based Pentium IV processor featured 1-MB L2 Advanced Transfer Cache (ATC) compared to 512-KB on the 130nm micron process-based Pentium IV processor. The Level 2 ATC delivers a much higher data throughput channel between the Level 2 cache and the processor core. The Advanced Transfer Cache consists of a 256-bit (32-byte) interface that transfers data on each core clock. As a result, the Pentium IV processor at 3.60 GHz can deliver a data transfer rate of **108 GB/s**. Features of the ATC include:

- Non-Blocking, full speed, on-die level 2 cache
- 8-way set associativity
- 256-bit data bus to the level 2 cache
- Data clocked into and out of the cache every clock cycle

Integrated 2-MB Level 3 Cache on Intel® Pentium® IV Processor Extreme Edition

The 2-MB L3 cache was available only with the Pentium IV processor Extreme Edition at 3.40 GHz. The additional third level of cache is located on the processor die and was designed specifically to meet the compute needs of high-end gamers and other power users. The L3 cache was coupled with the 800 MHz system bus to provide a high bandwidth path to memory. The efficient design of the integrated L3 cache provided a faster path to large data sets stored in cache on the processor. This results in reduced average memory latency and increased throughput for larger workloads.

L1 Data Cache

The second major development in cache technology was that the 130nm Pentium IV had only one L1 8 KB data cache. In place of the L1 instruction cache (I-Cache) in the 6th generation Pentiums it had a much more efficient Execution Trace Cache. The 90nm Pentium IV processor featured 16-KB data cache. On a general note, Intel reduced the size of its L1 data cache to enable a very low latency of only 2 clock cycles. This results in an overall read latency (the time it takes to read data from cache memory) of less than half of the Pentium III's L1 data cache. Clearly, advances in processor design and manufacture had permitted Intel to increase the L1 Data cache to 16-KB.

NetBurst Micro-Architecture

Intel claimed that its NetBurst Micro-Architecture provided a firm foundation for future advances in processor performance, particularly where speed of operation was concerned. That may be true, but Intel changed its design strategy and preoccupation with higher and higher GHz.

Intel's NetBurst microarchitecture delivered a number of innovative features including Hyper-Threading Technology, hyper-pipelined technology, 800, 533 or 400 MHz system bus, and Execution Trace Cache, as well as a number of enhanced features such as Advanced Transfer Cache, Advanced Dynamic Execution, enhanced floating-point and multimedia unit, and Streaming SIMD Extensions 2 (SSE2). Further enhancements in the 90 nm process-based Pentium IV processor included Streaming SIMD Extensions 3 (SSE3). Many of these innovations and advances were made possible with improvements in processor technology, process technology, and circuit design and could not previously be implemented in high-volume, manufacturable solutions. The features and resulting benefits of the microarchitecture are defined below.

Hyper Pipelined Technology

The traditional approach to increasing a CPU's clock speed was make smaller processors by shrinking the die. An alternative strategy evident in RISC processors is to make the CPU more efficient do less per clock cycle and have more of them. To do this in a CISC-based processor, Intel simply increased the number of stages in the processor's pipeline. The upshot of this is that less is accomplished per clock cycle. This is akin to a 'bucket-brigade' passing smaller buckets rapidly down a chain, rather than larger buckets at a slower rate. For example, the U and V integer pipelines in the original Pentium each had just five stages: instruction fetch, decode 1, decode 2, execute and write-back. The Pentium Pro introduced a P6 architecture with a pipeline consisting of 10-12 stages. The P7 NetBurst micro-architecture in the Pentium IV increased the number of stages to 20. This, Intel termed its Hyper Pipelined Technology. In the 90 nm Pentium IV processor, one of the key pipelines, the branch prediction/recovery pipeline, was implemented in 31 stages, compared to 20 stages on the 130 nm Pentium IV processor.

Enhanced Branch Prediction

The key to pipeline efficiency and operation is effective branch prediction, hence the much improved branch prediction logic in the Pentium IV's **Advanced Dynamic Execution Engine** (ADE). The Pentium IV's branch prediction logic delivered a 33% improvement in prediction efficiency than that of the Pentium III. The Pentium IV also contained a dedicated 4 KB Branch Transfer Buffer. When a processor's branch prediction logic predicts the flow of operation correctly no changes need to be made to the code in the pipeline. However, when an incorrect prediction is made, the contents of the pipeline must be replaced a new instruction cycle must begin at the start at the beginning of the pipeline. P6 generation processors with their 10 stage pipeline suffered a lower overhead penalty for an unpredicted branch than that of the Pentium IV with its 20 stage pipeline. The longer the pipeline, the further back in a process's instruction execution path the processor needs to go in order to correct unpredicted branches. One critical element in overcoming problems with unpredicted branches is the Execution Trace Cache.

Execution Trace Cache

The Pentium IV's sophisticated Execution Trace Cache is simply a 12 KB L1 instruction cache that sits between the decoders and the Rapid Execution Engine. The cache stores the microcode (micro-ops) of decoded complex instructions, especially those in a program loop, and minimises the wait time of the execution engine. This increases performance by removing the decoder from the main execution loop and makes more efficient usage of the cache storage space since instructions that are branched around are not stored. The result is a means to deliver a high volume of instructions to the processor's execution units and a reduction in the overall time required to recover from branches that have been mis-predicted.

Rapid Execution Engine

The major advance in the Pentium IV's execution unit is that its two Arithmetic Logic Units operated at twice the CPU clock rate. This meant that the 1.5 GHz Pentium 4 had ALU's running at 3 GHz: the ALU was effectively 'double pumped'. The Floating Point Unit had no such feature. Why the difference? Intel had to double pump the ALUs in order to deliver integer performance that was at least equal to that of a lower clocked Pentium III. Why? One reason was the length of the Pentium IV's 20 stage pipeline and the other was the need to ensure that any hit caused by poor branch prediction could be made up for by faster execution of microcode. The benefits here are that as the Pentium IV's clock speed increases, the integer performance of the processor will improve by a factor of two.

Enhanced Floating Point Processor

The Pentium IV has 128-bit floating point registers (up from the 80 bit registers in the P6 generation Pentiums) and a dedicated register for data movement. This enhanced floating point operations, which were not prone to the same type of branch prediction inefficiencies as integer-based instructions.

Streaming SIMD Extensions 2

Intel's Streaming SIMD (Single Instruction Multiple Data) Extensions (SSE) is a technology that allows a single instruction to be applied to multiple datasets at the same time. This is especially useful when processing 3 D graphics. SIMD-FP (Floating Point) extensions help speed up graphics processing by taking the multiplication, addition and reciprocal functions and applying them to the multiple datasets simultaneously. Recall, SIMD first appeared with the Pentium MMX which incorporated 57 MMX instructions. These are essentially SIMD-Int (integer) instructions. Intel first introduced SIMD-FP extensions in the Pentium III with 72 Streaming SIMD Extensions (SSE). Intel introduced 144 new instructions in the Pentium IV that enabled it to handle two 64-bit SIMD-INT operations and two double precision 64-bit SIMD-FP operations. This is contrast to the two 32-bit operations the Pentium MMX and III (under SSE) handle. The major benefit of SSE2 is enhanced greater performance, particularly with SIMD-FP instructions, as it increases the processor's ability to handle greater precision floating point calculations. As with MMX and SSE, these instructions require software support. The next generation 90 nm process-based Pentium IV processor introduced the Streaming SIMD Extensions 3 (SSE3), which includes 13 additional SIMD instructions over SSE2. The 13 new instructions in SSE3 are primarily designed to improve thread synchronization and specific application areas such as media and gaming.

Hyper-Threading Technology

Hyper-Threading Technology (HT Technology) was, and still is, a ground-breaking technology that changed the landscape of processor design by going beyond CPU power measured in GHz to improve processor performance. It allows software programs to "see" two processors and work more efficiently: it does this by incorporating two architectural states by adding a second register set to accommodate a second thread. Thus, this new technology enables the processor to execute threads of instructions at the same time, using the same core execution resources, thereby improving performance and system responsiveness—this is called **Thread Level Parallelism**. The Pentium VI processor supporting HT Technology was designed specially to deliver immediate increases in performance and system responsiveness with existing applications in multitasking environments (that is, where two or more functions are running at the same time) and with many stand-alone applications.