

Inside Intel Core Microarchitecture

By Gabriel Torres on April 12, 2006

Page 1 of 7

AMD Athlon™ Black Edition

Dual-core 5000+ Premium Performance Great Value. Do you Dare?
www.amd.com/unleash

Tsunami Flyer 6725S

Escolha a cor. Intel® Centrino® Duo. Só 1.230€

Ads by Google

Introduction

The new Intel Core microarchitecture, which was unveiled during IDF Spring 2006, will be used on all new CPUs from Intel, like Merom, Conroe and Woodcrest. It is based on Pentium M's microarchitecture, bringing some new features. In this tutorial we will give you an in-depth trip into this new microarchitecture from Intel.

The first thing to have in mind is that, besides the name, Core microarchitecture has nothing to do with Intel's Core Solo and Core Duo CPUs. Core Single is a Pentium M manufactured under 65 nm technology, while Core Duo – formerly known as Yonah – is a dual-core 65 nm CPU based on Pentium M's microarchitecture.

Pentium M is based on Intel's 6th generation architecture, a.k.a. P6, the same used by Pentium Pro, Pentium II, Pentium III and early Celeron CPUs and not on Pentium 4's as you may think, being originally targeted to mobile computers. You may think of Pentium M as an enhanced Pentium III. Thus you may think of Core microarchitecture as an enhanced Pentium M.

In order to continue reading this tutorial, however, you need to have read two other tutorials we have already posted: "[How a CPU Works](#)" and "[Inside Pentium M Architecture](#)". In the first one we explain the basics about how a CPU works, and on the second one, how Pentium M works. In the present tutorial we are assuming that you have already read them both, so if you didn't, please take a moment to read it before continuing, otherwise you may find yourself a little bit lost here. It is also a good idea to read our [Inside Pentium 4 Architecture](#) tutorial, just for understanding how Core microarchitecture differs from Pentium 4's.

Core microarchitecture uses a 14-stage pipeline. Pipeline is a list of all stages a given instruction must go thru in order to be fully executed. Intel didn't disclose Pentium M's pipeline and so far they didn't publish the description of each stage of Core microarchitecture pipeline as well, so we are unable to provide more in depth information on that. Pentium III used an 11-stage pipeline, the original Pentium 4 had a 20-stage pipeline and newer Pentium 4 CPUs based on "Prescott" core have a 31-stage one!

Of course whenever Intel publishes more details on Core microarchitecture we will update this tutorial.

Let's now talk what is different on Core microarchitecture from Pentium M's.

Originally at <http://www.hardwaresecrets.com/article/313/1>

Pages (7): [1](#) [2](#) [3](#) [4](#) [5](#) [6](#) [7](#) »

© 2004-6, Hardware Secrets, LLC. All Rights Reserved.

Total or partial reproduction of the contents of this site, as well as that of the texts available for downloading, be this in the electronic media, in print, or any other form of distribution, is expressly forbidden. Those who do not comply with these copyright laws will be indicted and punished according to the International Copyrights Law.

We do not take responsibility for material damage of any kind caused by the use of information contained in Hardware Secrets.

Inside Intel Core Microarchitecture

By Gabriel Torres on April 12, 2006

Page 2 of 7

AMD Athlon™ 64 X2 5000+

Control Processor Performance w/
a Customizable Clock Multiplier.
www.amd.com/unleash

Intel Processor Info

Get the latest Intel processor
blogs, white papers, and much
more!
Hardware.ITtoolbox.com

iMac Intel DuoCore Memory

100% Gaurant Compatible iMac
Duo Memory Upgrade. Free Same
Day Ship!

Tsunami Flyer 6725S

A mobilidade que deseja a core
Intel® Centrino® Duo. Só 1.230€
www.tsunami.pt

Ads by Google

Memory Cache and Fetch Unit

Just to remember, memory cache is a high-speed memory (static RAM or SRAM) embedded inside the CPU used to store data that the CPU may need. If the data required by the CPU isn't located in the cache, it must go all the way to the main RAM memory, which reduces its speed, as the RAM memory is accessed using the CPU external clock rate. For example, on a 3.2 GHz CPU, the memory cache is accessed at 3.2 GHz but the RAM memory is accessed only at 800 MHz.

Core microarchitecture was created having the multi-core concept in mind, i.e. more than one chip per packaging. On Pentium D, which is the dual-core version of Pentium 4, each core has its own L2 memory cache. The problem with that is that at some moment one core may run out of cache while the other may have unused parts on its own L2 memory cache. When this happens, the first core must grab data from the main RAM memory, even though there was empty space on the L2 memory cache of the second core that could be used to store data and prevent that core from accessing the main RAM memory.

On Core microarchitecture this problem was solved. The L2 memory cache is shared, meaning that both cores use the same L2 memory cache, dynamically configuring how much cache each core will take. On a CPU with 2 MB L2 cache, one core may be using 1.5 MB while the other 512 KB (0.5 MB), contrasted to the fixed 50%-50% division used on previous dual-core CPUs.

It is not only that. Prefetches are shared between the cores, i.e. if the memory cache system loaded a block of data to be used by the first core, the second core can also use the data already loaded on the cache. On the previous architecture, if the second core needed a data that was located on the cache of the first core, it had to access it thru the external bus (which works under the CPU external clock, which is far lower than the CPU internal clock) or even grab the required data directly from the system RAM.

Intel has also improved the CPU prefetch unit, which watches for patterns in the way the CPU is currently grabbing data from memory, in order to try to "guess" which data the CPU will try to load next and load it to the memory cache before the CPU requires it. For example, if the CPU has just loaded data from address 1, then asked for data located on address 3, and then asked for data located on address 5, the CPU prefetch unit will guess that the program running will load data from address 7 and will load data from this address before the CPU asks for it. Actually this idea isn't new and all CPUs since the Pentium Pro use some kind of predicting to feed the L2 memory cache. On Core microarchitecture Intel has just enhanced this feature by making the prefetch unit look for patterns in data fetching instead of just static indicators of what data the CPU would ask next.

Originally at <http://www.hardwaresecrets.com/article/313/2>

Pages (7): [1](#) [2](#) [3](#) [4](#) [5](#) [6](#) [7](#) »

© 2004-6, Hardware Secrets, LLC. All Rights Reserved.

Total or partial reproduction of the contents of this site, as well as that of the texts available for downloading, be this in the electronic media, in print, or any other form of distribution, is expressly forbidden. Those who do not comply with these copyright laws will be indicted and punished according to the International Copyrights Law.

We do not take responsibility for material damage of any kind caused by the use of information contained in Hardware Secrets.

Inside Intel Core Microarchitecture

By Gabriel Torres on April 12, 2006

Page 3 of 7

Kimberley Adventures

Walking, trekking & cruising thru out the Kimberley; small groups
AustralianWalkingHolidays.com.au

HgH.com AntiAging

120 Day Guarantee Ships same day- Buy 2 Get 1 Free
www.HumanGrowthHormone.com

Intel® Dialogic Quad Span

DMV1200A 4*E1 Board PCI & cPCI Expert

Ads by Google

Instruction Decoder: Macro-Fusion

A new concept was introduced with Core microarchitecture: macro-fusion. Macro-fusion is the ability of joining two x86 instructions together into a single micro-op. This improves the CPU performance and lowers the CPU power consumption, since it will execute only one micro-op instead of two.

This scheme, however, is limited to compare and conditional branching instructions (i.e. CMP and TEST and Jcc instructions). For example, consider this piece of a program:

```
...  
load eax, [mem1]  
cmp eax, [mem2]  
jne target  
...
```

What this does is to load the 32-bit register EAX with data contained in memory position 1, compare its value with data contained in memory position 2 and, if they are different (jne = jump if not equal), the program goes to address "target", if they are equal, the program continues on the current position.

With macro-fusion the comparison (cmp) and branching (jne) instructions will be merged into a single micro-op. So after passing thru the instruction decoder, this part of the program will something like this:

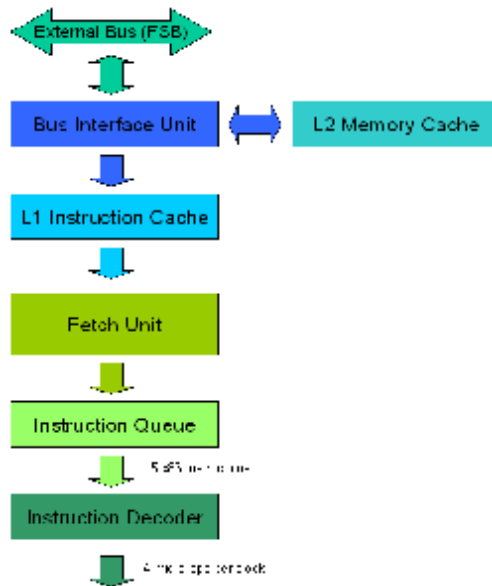
```
...  
load eax, [mem1]  
cmp eax, [mem2] + jne target  
...
```

As we can see, we saved one instruction. The less instructions there are to be executed, the faster the computer will finish the execution of the task and also less power is generated.

The instruction decoder found on Core microarchitecture can decode four instructions per clock cycle, while previous CPUs like Pentium M and Pentium 4 are able to decode only three.

Because of macro-fusion, the Core microarchitecture instruction decoder pulls five instructions per time for the instruction queue, even though it can only decode four instructions per clock cycle. This is done so if two of these five instructions are fused into one, the decoder can still decode four instructions per clock cycle. Otherwise it would be partially idle whenever a macro-fusion took place, i.e. it would deliver only three micro-ops on its output while it is capable of delivering up to four.

On Figure 1 you can see a brief summary of what we explained on this page and on the previous one.



click to enlarge

Figure 1: Fetch unit and instruction decoder on Core microarchitecture.

Originally at <http://www.hardwaresecrets.com/article/313/3>

Pages (7): [1](#) [2](#) [3](#) [4](#) [5](#) [6](#) [7](#) [»](#)

© 2004-6, Hardware Secrets, LLC. All Rights Reserved.


Total or partial reproduction of the contents of this site, as well as that of the texts available for downloading, be this in the electronic media, in print, or any other form of distribution, is expressly forbidden. Those who do not comply with these copyright laws will be indicted and punished according to the International Copyrights Law.

We do not take responsibility for material damage of any kind caused by the use of information contained in Hardware Secrets.

Inside Intel Core Microarchitecture

By Gabriel Torres on April 12, 2006

Page 4 of 7

<p>AMD Athlon™ Black Edition Dual-core 5000+ Premium Performance Great Value. Do you Dare? www.amd.com/unleash</p>	<p>Intel Processor Info Get the latest Intel processor blogs, white papers, and much more! Hardware.ITtoolbox.com</p>	
<p>Tsunami Flyer 6725S A mobilidade que deseja a core Intel®</p>		

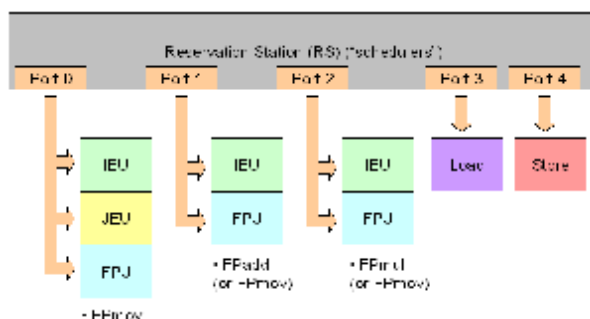
Execution Units

Pentium M has five dispatch ports located on its Reservation Station, but only two ports are used to dispatch micro-ops to execution ports. The other three are used by memory-related units (Load, Store Address and Store Data). Core microarchitecture has also five dispatch ports, however three of them are used to send micro-ops to execution units. This means that CPUs using Core microarchitecture will be able to send three micro-ops to be executed per clock cycle, contrasted to only two on Pentium M.

Core microarchitecture provides one extra FPU and one extra IEU (a.k.a. ALU) compared to Pentium M's architecture. This means Core microarchitecture can process three integer instructions per clock cycle, contrasted to only two on Pentium M.

But not all math instructions can be executed on all FPUs. As you can see on Figure 2, floating-point multiplication operations can only be executed on the third FPU and floating-point adds can only be executed on the second FPU. FPMov instructions can be executed on the first FPU or on the other two FPUs if there is no other more complex instruction (FPadd or FPMul) ready to be dispatched to them. MMX/SSE instructions are dealt by the FPU.

On Figure 2 you see a preliminary block diagram of Core microarchitecture execution units.



click to enlarge

Figure 2: Core microarchitecture execution units.

Another big difference between Pentium M and Pentium 4 architectures to Core architecture is that on Core architecture the Load and Store units have their own address generation units embedded. Pentium 4 and Pentium M have a separated address generation unit, and on Pentium 4 the first ALU is used to store data on memory.

Here is a small explanation of each execution unit found on this CPU:

- IEU: Instruction Execution Unit is where regular instructions are executed. Also known as ALU (Arithmetic and Logic Unit). "Regular" instructions are also known as "integer" instructions.
- JEU: Jump Execution Unit processes branches and is also known as Branch Unit.
- FPU: Floating-Point Unit. Is responsible for executing floating-point math operation and also MMX and SSE instructions. In this CPU the FPUs aren't "complete", as some instruction types (FPMov, FPadd and FPMul) can only be executed on certain FPUs:
 - FPadd: Only this FPU can process floating-point addition instructions, like ADDPS (which, by the way, is a SSE instruction).
 - FPMul: Only this FPU can process floating-point multiplication instructions, like MULPS (which, by the way, is a SSE instruction).
 - FPMov: Instructions for loading or copying a FPU register, like MOVAPS (which transfers

data to a SSE 128-bit XMM register). This kind of instruction can be executed on any FPU, but on the second and on the third FPUs only if FPad- or FPMul-like instructions aren't available in the Reservation Station to be dispatched.

- Load: Unit to process instructions that ask a data to be read from the RAM memory.
- Store Data: Unit to process instructions that ask a data to be written at the RAM memory.

Keep in mind that complex instructions may take several clock cycles to be processed. Let's take an example of port 2, where the FPMul unit is located. While this unit is processing a very complex instruction that takes several clock ticks to be executed, port 2 won't stall: it will keep sending simple instructions to the IEU while the FPU is busy.

Originally at <http://www.hardwaresecrets.com/article/313/4>

Pages (7): [1](#) [2](#) [3](#) [4](#) [5](#) [6](#) [7](#) [»](#)

© 2004-6, Hardware Secrets, LLC. All Rights Reserved.

Total or partial reproduction of the contents of this site, as well as that of the texts available for downloading, be this in the electronic media, in print, or any other form of distribution, is expressly forbidden. Those who do not comply with these copyright laws will be indicted and punished according to the International Copyrights Law.

We do not take responsibility for material damage of any kind caused by the use of information contained in Hardware Secrets.

Inside Intel Core Microarchitecture

By Gabriel Torres on April 12, 2006

Page 5 of 7

AMD Quad Core Opteron

Quad Core Opterons Are Here!
Custom configurations available.
www.advancedclustering.com

Intel Processor Info

Get the latest Intel processor
blogs, white papers, and much
more!
Hardware.ITtoolbox.com

AMD Athlon™ 64 X2 5000+

Control Processor Performance w/
a Customizable Clock Multiplier.
www.amd.com/unleash

Tsunami Flyer 6725S

Escolha a cor. Intel® Centrino®
Duo. Só 1.230€
www.tsunami.pt

Ads by Google

128-Bit Internal Datapath

Another new feature found on Core microarchitecture is a true 128-bit internal datapath. On previous CPUs, the internal datapath was of 64 bits only. This was a problem for SSE instructions, since SSE registers, called XMM, are 128-bit long. So, when executing an instruction that manipulated a 128-bit data, this operation had to be broke down into two 64-bit operations.

The new 128-bit datapath makes Core microarchitecture faster to process SSE instructions that manipulate 128-bit data.

Intel is calling this new feature "Advanced Digital Media Boost".

Originally at <http://www.hardwaresecrets.com/article/313/5>

Pages (7): [1](#) [2](#) [3](#) [4](#) [5](#) [6](#) [7](#) »

© 2004-6, Hardware Secrets, LLC. All Rights Reserved.

Total or partial reproduction of the contents of this site, as well as that of the texts available for downloading, be this in the electronic media, in print, or any other form of distribution, is expressly forbidden. Those who do not comply with these copyright laws will be indicted and punished according to the International Copyrights Law.

We do not take responsibility for material damage of any kind caused by the use of information contained in Hardware Secrets.

Inside Intel Core Microarchitecture

By Gabriel Torres on April 12, 2006

Page 6 of 7

<p>AMD Athlon™ 64 X2 5000+ Control Processor Performance w/ a Customizable Clock Multiplier. www.amd.com/unleash</p>	<p>Tsunami Runner 9572S Mais performance Intel® Centrino® Duo. Só 1.199€ www.tsunami.pt</p>	<p>Intel Processor Info Get the latest Intel processor blogs, white papers, and much more! Hardware.ITtoolbox.com</p>	<p>Portátil Intel - Growing Performance a excelentes preços, com os novos Intel® Centrino® Duo www.growing.pt</p>
--	--	--	--



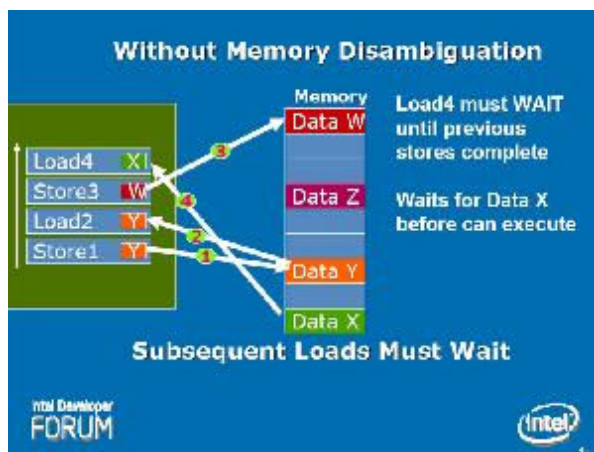
Memory Disambiguation

Memory disambiguation is a technique to accelerate the execution of memory-related instructions.

All Intel CPUs since Pentium Pro have an out-of-order engine, which allows the CPU to execute non-dependant instructions in any order. What happens is that memory-related instructions are traditionally executed in the same order they appear on the program, otherwise data inconsistency could appear. For example, if the original program has an instruction like "store 10 at address 5555" and then a "load data stored at 5555", they cannot be reversed (i.e. executed out of order) or the second instruction would get wrong data, as the data of address 5555 was changed by the first instruction.

What the memory disambiguation engine does is locate and execute memory-related instructions that can be executed out of order, accelerating the execution of the program (we will explain how this is accomplished in a minute).

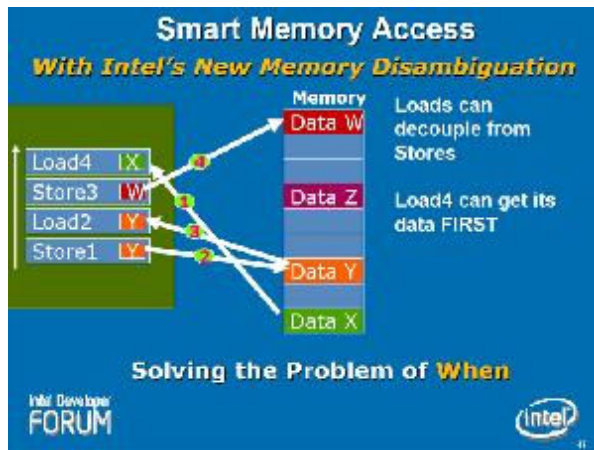
On Figure 3 you have an example of a CPU without memory disambiguation (i.e. all CPUs not based on Core microarchitecture). As you can see, the CPU has to execute the instructions as they appear on the original program. For example, "Load4" isn't related to any other memory-related instruction and could be executed first, however it has to wait all other instructions to be executed first.



click to enlarge

Figure 3: CPU without memory disambiguation.

On Figure 4 you see how the program shown on Figure 3 works on a CPU based on Core microarchitecture. It "knows" that "Load4" isn't related to the other instructions and can be executed first.



click to enlarge

Figure 4: CPU with memory disambiguation.

This improves the CPU performance because now that "Load4" is executed, the CPU has the data required for executing other instructions that need the value of "X" to be executed.

On a regular CPU, if after this "Load4" we had an "Add 50", this "Add 50" (and all other instructions that depend on that result) would have to wait all other instructions shown on Figure 3 to be executed. With memory disambiguation, these instructions can be executed early, since the CPU will now have the value of "X" early.

Originally at <http://www.hardwaresecrets.com/article/313/6>

Pages (7): [1](#) [2](#) [3](#) [4](#) [5](#) [6](#) [7](#) »

© 2004-6, Hardware Secrets, LLC. All Rights Reserved.

Total or partial reproduction of the contents of this site, as well as that of the texts available for downloading, be this in the electronic media, in print, or any other form of distribution, is expressly forbidden. Those who do not comply with these copyright laws will be indicted and punished according to the International Copyrights Law.

We do not take responsibility for material damage of any kind caused by the use of information contained in Hardware Secrets.

Inside Intel Core Microarchitecture

By Gabriel Torres on April 12, 2006

Page 7 of 7

AMD Athlon™ 64 X2

Black Edition processor 5000+ w/
Tunable Performance. Learn more!
www.amd.com/unleash

Intel Processor Info

Get the latest Intel processor
blogs, white papers, and much
more!
Hardware.ITtoolbox.com

Itanium / 64-Bit Migration

Move your application to Itanium.
Upgrade your application to 64-bit
www.s7solutions.com

Torquato PDC drill bits

PDC bits from 3-7/8" to 12-1/4"
Drill faster and deeper.
www.torquato.com/pdcrockbits.htm

Ads by Google

Advanced Power Gating

With advanced power gating, Core microarchitecture brought CPU power saving to a totally new level. This feature enables the CPU to shut down units that aren't being used at the moment. This idea goes even further, as the CPU can shut down specific parts inside each CPU unit in order to save energy, to dissipate less power and to provide a greater battery life (in the case of mobile CPUs).

Another power-saving capability of Core microarchitecture is to turn on only the necessary bits in the CPU internal busses. Many of the CPU internal busses are sized for the worst-case scenario – i.e. the largest x86 instruction that exists, which is a 15-byte wide instruction (480 bits)*. So, instead turning on all the 480 data lanes of this particular bus, the CPU can turn on only 32 of its data lanes, all that is necessary for transferring a 32-bit instruction, for example.

* You can find yourself quite lost by this statement, since you were always told that Intel architecture uses 32-bits instructions, so further explanation is necessary in order to clarify this affirmation.

Inside the CPU what is considered an instruction is the instruction opcode (the machine language equivalent of the assembly language instruction) plus all its required data. This is because in order to be executed, the instruction must enter the execution engine "completed", i.e. together with all its required data. Also, the size of each x86 instruction opcode is variable and not fixed at 32 bits, as you may think. For example, an instruction like `mov eax, (32-bit data)`, which stores the (32-bit data) into the CPU's EAX register is considered internally as a 40-bit length instruction (`mov eax` translates into a 8-bit opcode plus the 32 bits from its data). Actually, having instruction with several different lengths is what characterizes a CISC (Complex Instruction Set Computing) instruction set.

If you want to learn more about this subject, read [AMD64 Architecture Programmer's Manual Vol. 3: General Purpose and System Instructions](#) (even though Intel provides the same information on their [Intel Architecture Software Developer's Manual Vol. 2A](#), AMD explanation and diagrams are easier to understand).

Originally at <http://www.hardwaresecrets.com/article/313/7>

Pages (7): [1](#) [2](#) [3](#) [4](#) [5](#) [6](#) [7](#) »

© 2004-6, Hardware Secrets, LLC. All Rights Reserved.

Total or partial reproduction of the contents of this site, as well as that of the texts available for downloading, be this in the electronic media, in print, or any other form of distribution, is expressly forbidden. Those who do not comply with these copyright laws will be indicted and punished according to the International Copyrights Law.

We do not take responsibility for material damage of any kind caused by the use of information contained in Hardware Secrets.