Microarquiteturas Modernas x86

- Para aproveitar as otimizações existentes nos processadores x86 atuais, é preciso conhecer algumas características relevantes das microarquiteturas modernas
- Microarquiteturas Modernas x86 (Intel)
  - Pentium M
  - Sandy Bridge
  - Intel Core
  - Nehalem
  - Netburst
- Cada arquitetura tem um conjunto de objetivos específicos
  - Exemplo: Intel Core - alta performance com eficiência energética; Pentium M - eficiência energética com performance razoável

Características Relevantes para Otimização de Software

- Microarquiteturas que permitem execução com alta vazão (throughput) em clocks elevados, com hierarquia de cache de alta velocidade e barramentos muito rápidos
- Arquiteturas multicore
- Suporte a tecnologia Hyper-Threading
- Suporte a 64 bit
- Extensões SIMD
  - MMX, SSE (Streaming SIMD Extensions), SSE2, SSE3, SSSE3 (Supplemental SSE3), SSE4.1, SSE4.2
  - AVX (Advanced Vector Extensions)

Características Relevantes para Otimização de Software

- Microarquiteturas que permitem execução com alta vazão (throughput) em clocks elevados, com hierarquia de cache de alta velocidade e barramentos muito rápidos
- Arquiteturas multicore
- Suporte a tecnologia Hyper-Threading
- Suporte a 64 bit
- Extensões SIMD
  - MMX, SSE (Streaming SIMD Extensions), SSE2, SSE3, SSSE3 (Supplemental SSE3), SSE4.1, SSE4.2
  - AVX (Advanced Vector Extensions)

MicroArquitetura x86

Sandy Bridge

Microarquitetura Sandy Bridge

- Evolução das microarquiteturas Intel Core e Nehalem
  - Exemplos: processadores Core i7 2xxx
- Algumas Características:
  - Suporte a Advanced Vector Extensions (AVX)
    - Extensão 256-bit ponto flutuante às instruções SIMD 128-bits, oferecendo até 2x a performance de código baseado em SIMD 128-bit
  - Melhoras na execução de instruções
    - Exemplo: branch prediction mais eficiente

Microarquitetura Sandy Bridge

- Algumas Características (cont.):
  - Hierarquia de cache melhorada, aumentando a banda
  - Suporte a SoC (system on a chip)
    - Processador gráfico integrado
    - Controlador PCIE integrado
    - Controlador de memória integrado
  - Turbo Boost Technology melhorada
    - Gerenciamento automático do clock dos cores do processador para mantê-lo sempre a uma temperatura adequada

Sandy Bridge: Pipeline

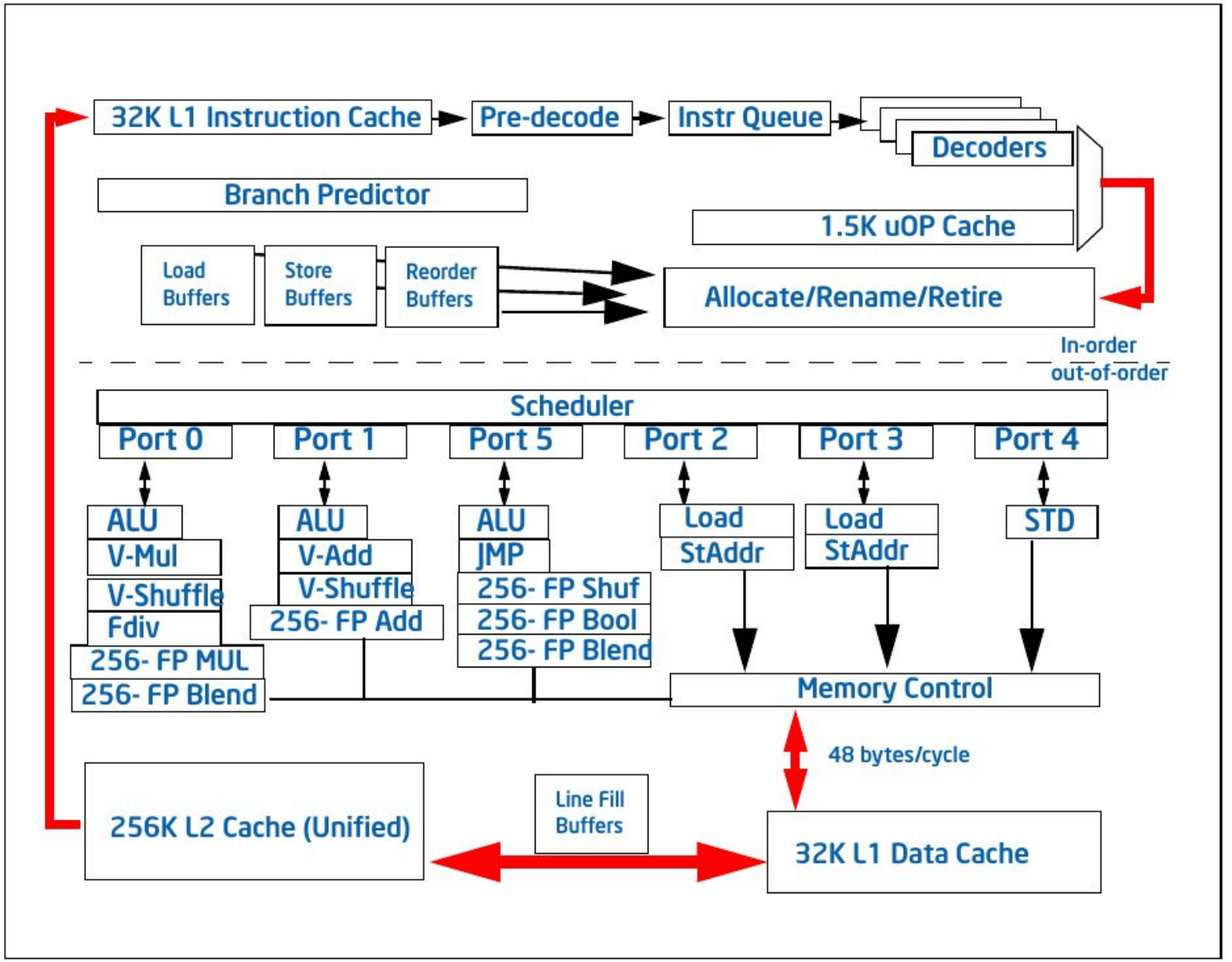
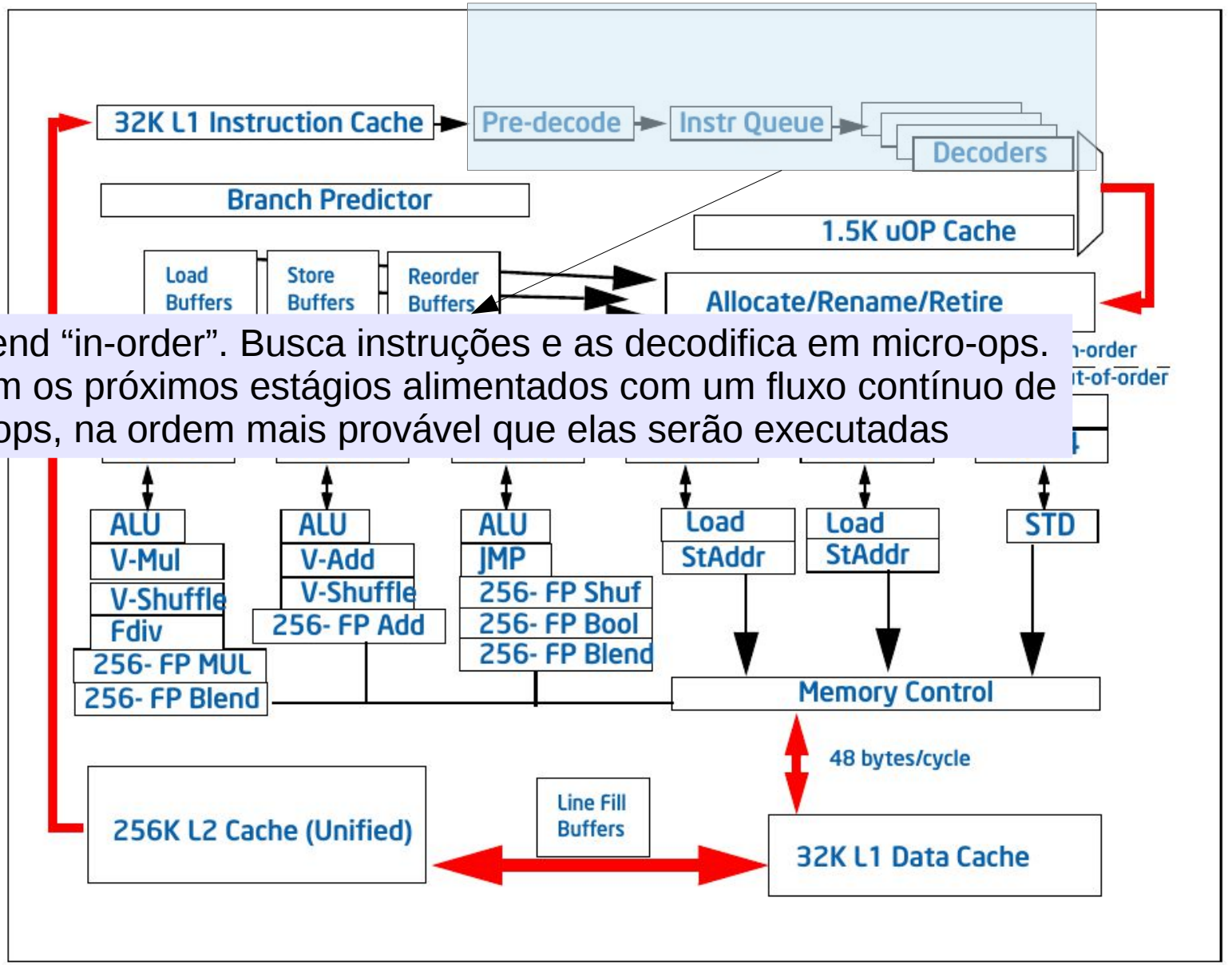


Figure 2-1. Intel microarchitecture code name Sandy Bridge Pipeline Functionality

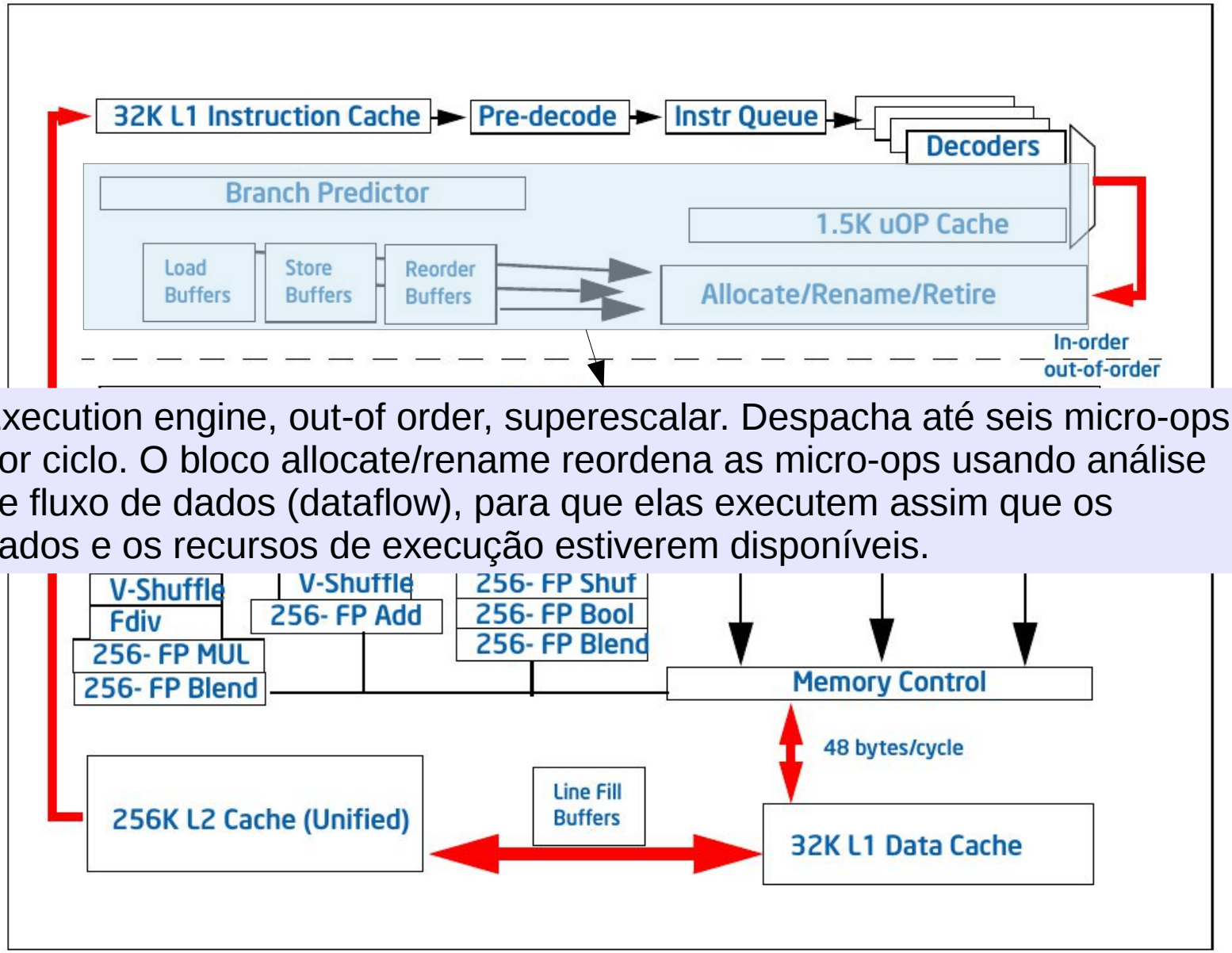
Sandy Bridge: Pipeline



Front-end “in-order”. Busca instruções e as decodifica em micro-ops. Mantém os próximos estágios alimentados com um fluxo contínuo de micro-ops, na ordem mais provável que elas serão executadas

Figure 2-1. Intel microarchitecture code name Sandy Bridge Pipeline Functionality

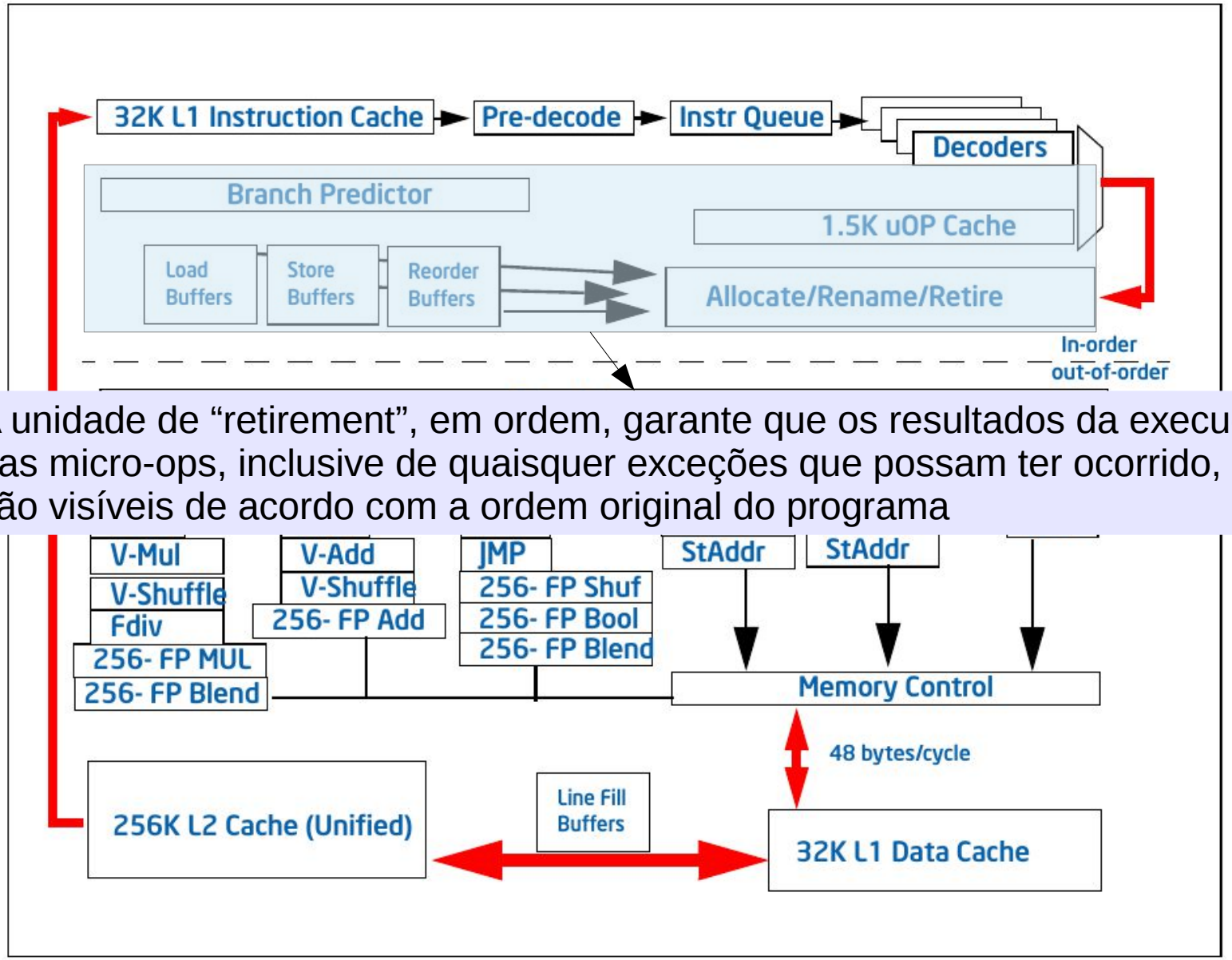
Sandy Bridge: Pipeline



Execution engine, out-of order, superescalar. Despacha até seis micro-ops por ciclo. O bloco allocate/rename reordena as micro-ops usando análise de fluxo de dados (dataflow), para que elas executem assim que os dados e os recursos de execução estiverem disponíveis.

Figure 2-1. Intel microarchitecture code name Sandy Bridge Pipeline Functionality

Sandy Bridge: Pipeline



A unidade de “retirement”, em ordem, garante que os resultados da execução das micro-ops, inclusive de quaisquer exceções que possam ter ocorrido, são visíveis de acordo com a ordem original do programa

Figure 2-1. Intel microarchitecture code name Sandy Bridge Pipeline Functionality

Sandy Bridge: fluxo de uma instrução, resumo

(1) A unidade de branch prediction escolhe o próximo bloco de código do programa a ser executado

(2) As micro-ops correspondentes a esse código são enviadas para o bloco Rename/retirement. Elas entram no Scheduler na ordem do programa, mas são executadas em ordem de dataflow. Para micro-ops que estejam prontas simultaneamente, quase sempre é respeitada a ordem FIFO.

- As micro-ops são executadas usando recursos de execução organizados em três pilhas. (detalhes no próximo slide)

(3) Operações em memória são reordenadas para alcançar paralelismo e performance máxima. Caches misses no L1 data cache vão para o L2 Cache

(4) Exceções são sinalizadas no retirement da instrução que o causou.

Observação: cada core de um processador Sandy Bridge pode suportar 2 processadores lógicos se o HyperThreading for habilitado.

Sandy Bridge: pilhas de execução de micro-ops

Pilha		***Porta 0***	***Porta 1***	***Porta 2***	***Porta 3***	***Porta 4***	***Porta 5***
Inteiros – uso geral		ALU, Shift	ALU, Fast LEA, Slow LEA, MUL	Load_Addr, Store_addr	Load_Addr, Store_addr	Store_data	ALU, Shift, Branch, Fast LEA
SIMD	Inteiro (SSE-Int, AVX-Int, MMX)	Mul, Shift, STTNI, Int-Div, 128b-Mov	ALU, Shuf, Blend, 128b-Mov			Store_data	ALU, Shuf, Shift, Blend, 128b-Mov
	Ponto-flutuante (SSE-FP, AVX-FP_low)	Mul, Div, Blend, 256b-Mov	Add, CVT			Store_data	Shuf, Blend, 256b-Mov
X87, AVX-FP_High		Mul, Div, Blend, 256b-Mov	Add, CVT			Store_data	Shuf, Blend, 256b-Mov

Sandy Bridge: hierarquias de cache

- First level **instruction** cache
- First level **data** cache (L1 Dcache)
  - Pode ser compartilhado entre dois processadores lógicos se o processador suportar HyperThreading
- Second level (unified: data/instruction) cache (L2)
- Todos os cores em um processador físico compartilham um shared last level cache (LLC)
- A coerência de dados em todos os níveis de cache é mantida usando o protocolo MESI

Protocolo MESI: Visão Geral

Protocolo de coerência de Cache MESI

Modified **–** o bloco endereçado é válido na cache e apenas nessa cache. O bloco foi modificado em relação à memória do sistema – isto é, os dados modificados do bloco não foram ainda escritos na memória.

Exclusive **–** o bloco endereçado está somente nessa cache. Os dados desse bloco estão consistentes com a memória do sistema.

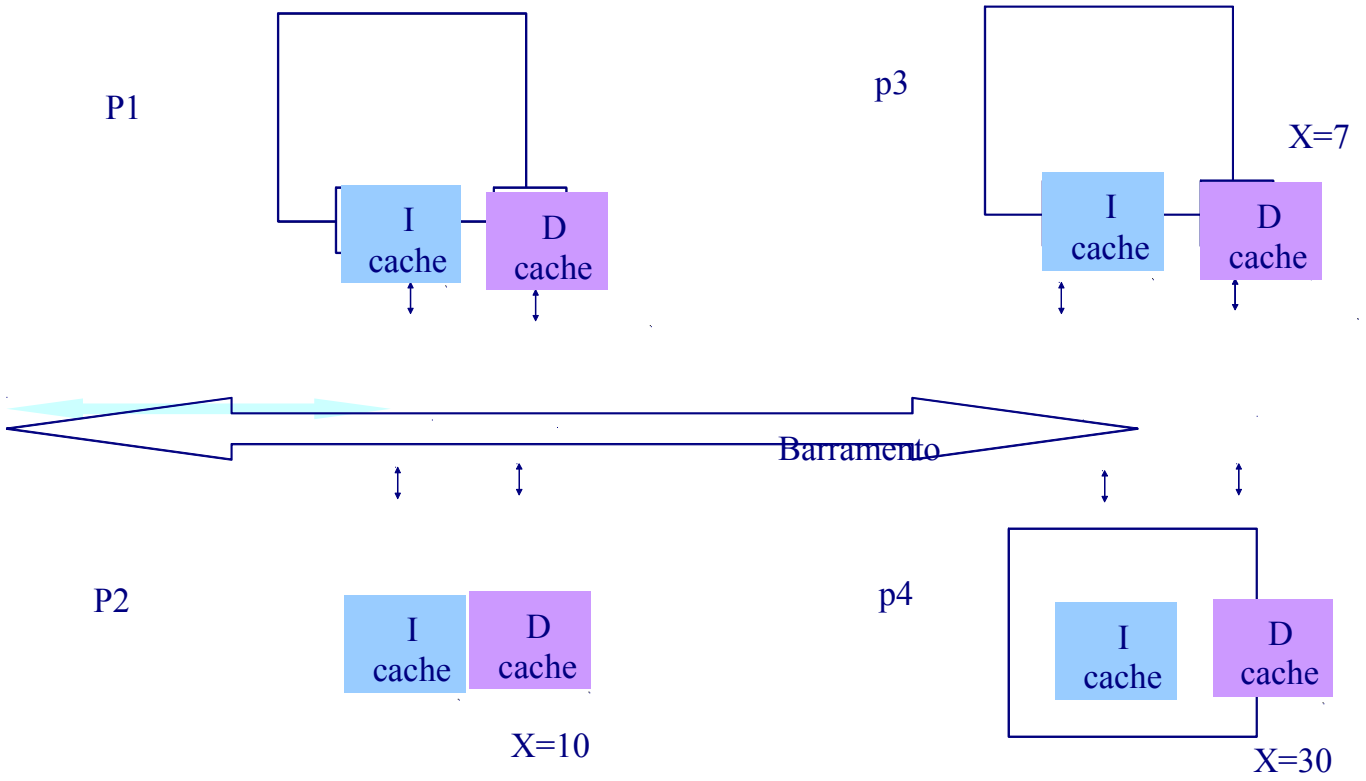
Shared **–** o bloco endereçado é válido nesta cache e em pelo menos uma outra cache.

Invalid **–** o bloco endereçado está inválido (não está presente na cache ou os dados contidos não podem ser usados).

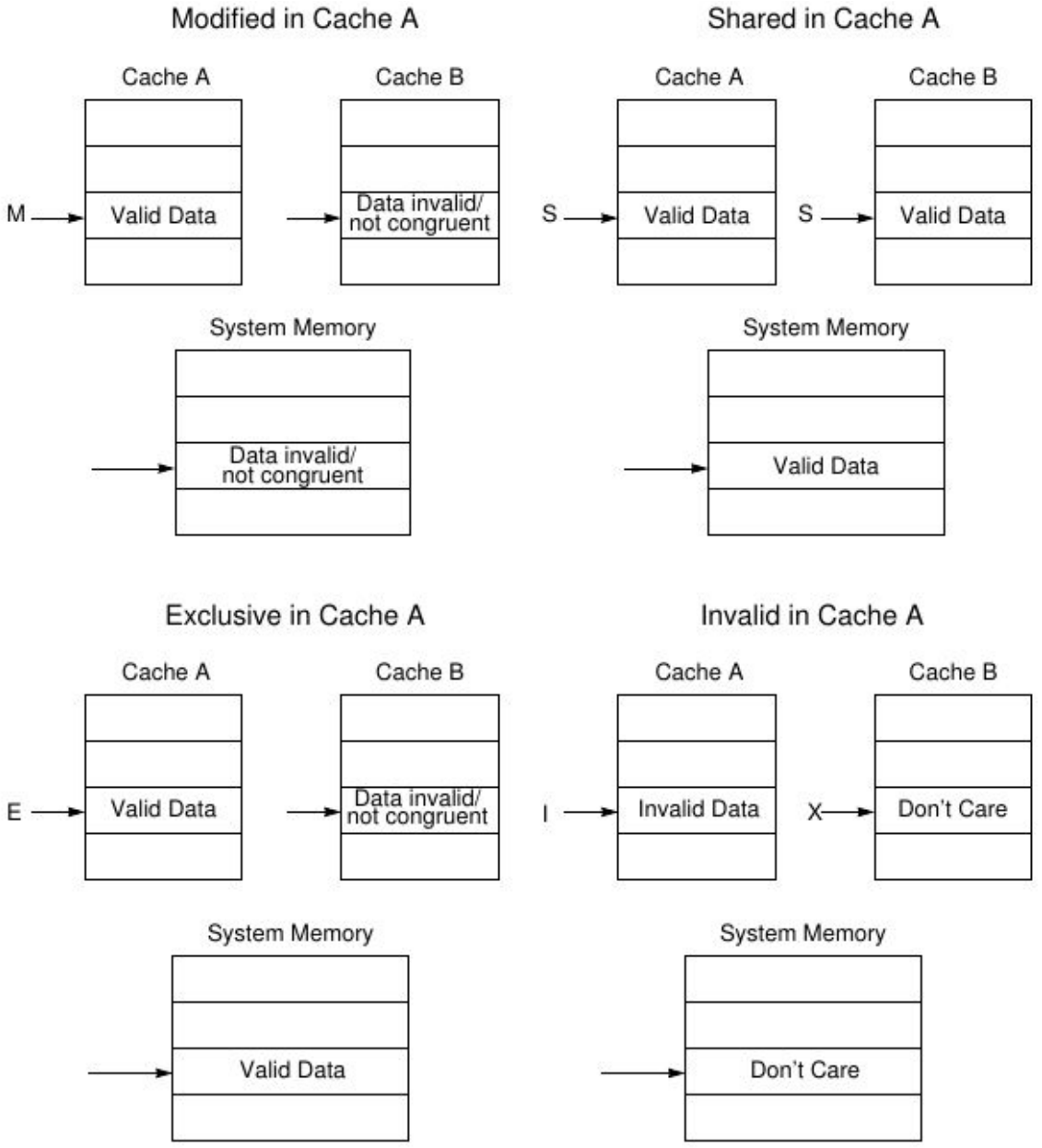
Protocolo MESI: Visão Geral

Protocolo MESI requer a monitoração de todos os acessos a memória em sistema multiprocessado

Adota bus snooping



Relação entre estados MESI



MicroArquitetura x86

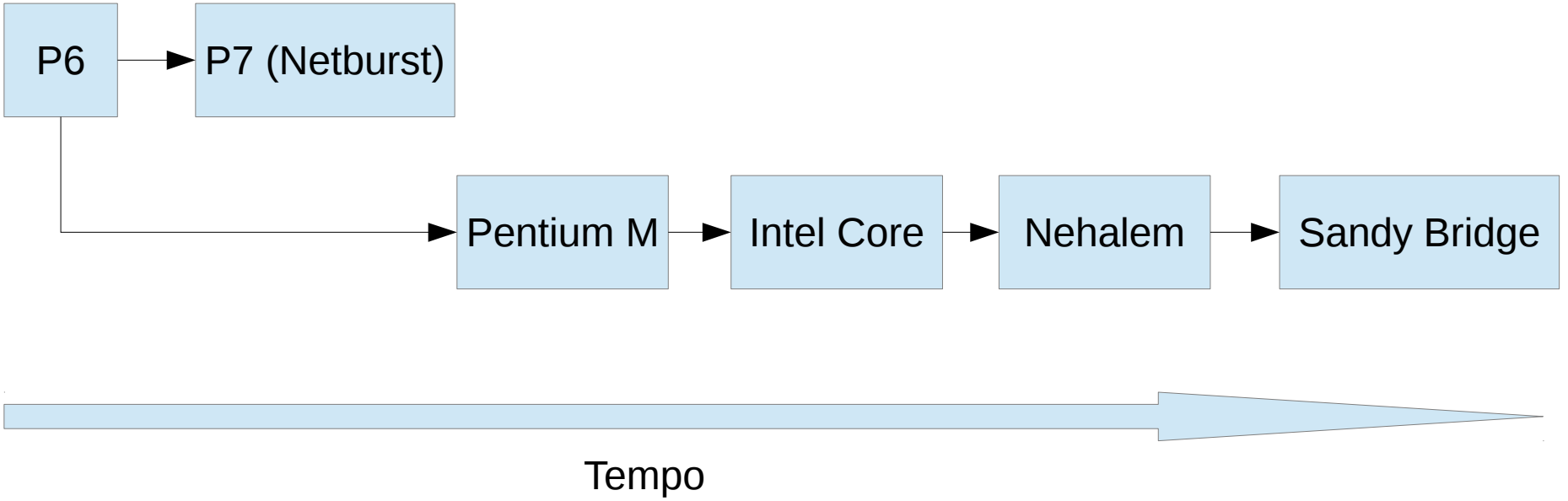
Pentium M

MicroArquitetura Pentium M

- Foco em eficiência energética
- Baseado na arquitetura de sexta geração da Intel, conhecida como P6, usada em:
  - Pentium Pro, Pentium II, Pentium III
- Pipeline “mais curto” do que outras arquiteturas
  - Como Pentium 4
  - A Intel não costuma mais divulgar o tamanho do pipeline

Evolução das MicroArquiteturas

PX – Geração X



Grosso modo:
P6: Pentium Pro, Pentium II, Pentium III
P7: Pentium 4
Pentium M: Notebooks chamados “Centrino” = Pentium M + chipsets/wi-fi Intel
Intel Core: Core 2 processor series (Core 2 Duo, Core 2 Quad etc.)
Nehalem: primeiros Core i3, i5 e i7
Sandy Bridge: Core i3, i5, i7 mais recentes

Pentium M: Características

- Four data transferred per clock cycle. This technique is called QDR (Quad Data Rate) and makes the local bus to have a performance four times its actual clock rate, see table below.

Real Clock	Performance	Trnasfer Rate
100 MHz	400 MHz	3.2 GB/s
133 MHz	533 MHz	4.2 GB/s

- L1 memory cache: two 32 KB L1 memory caches, one for data and another for instructions (Pentium III had two 16 KB L1 memory caches).
- L2 memory cache: 1 MB on 130 nm models (“Banias” core) or 2 MB on 90 nm models (“Dothan” core). Pentium III had up to 512 KB. Celeron M, which is a low-cost version of Pentium M, has a 512 KB L2 memory cache.
- Support for SSE2 instructions.
- Advanced branch prediction: branch prediction circuit was redesigned (and based on Pentium 4’s branch prediction circuit) to improve performance.
- Micro-ops fusion: The instruction decoder fuses two micro-ops into one micro-op in order to save energy and improve performance. We’ll talk more about this later.
- Enhanced SpeedStep Technology, which allows the CPU to reduce its clock while idle in order to save battery life.
- Several other battery-saving features were added to Pentium M’s microarchitecture, since this CPU was originally designed for mobile computers.

MicroArquitetura Pentium M

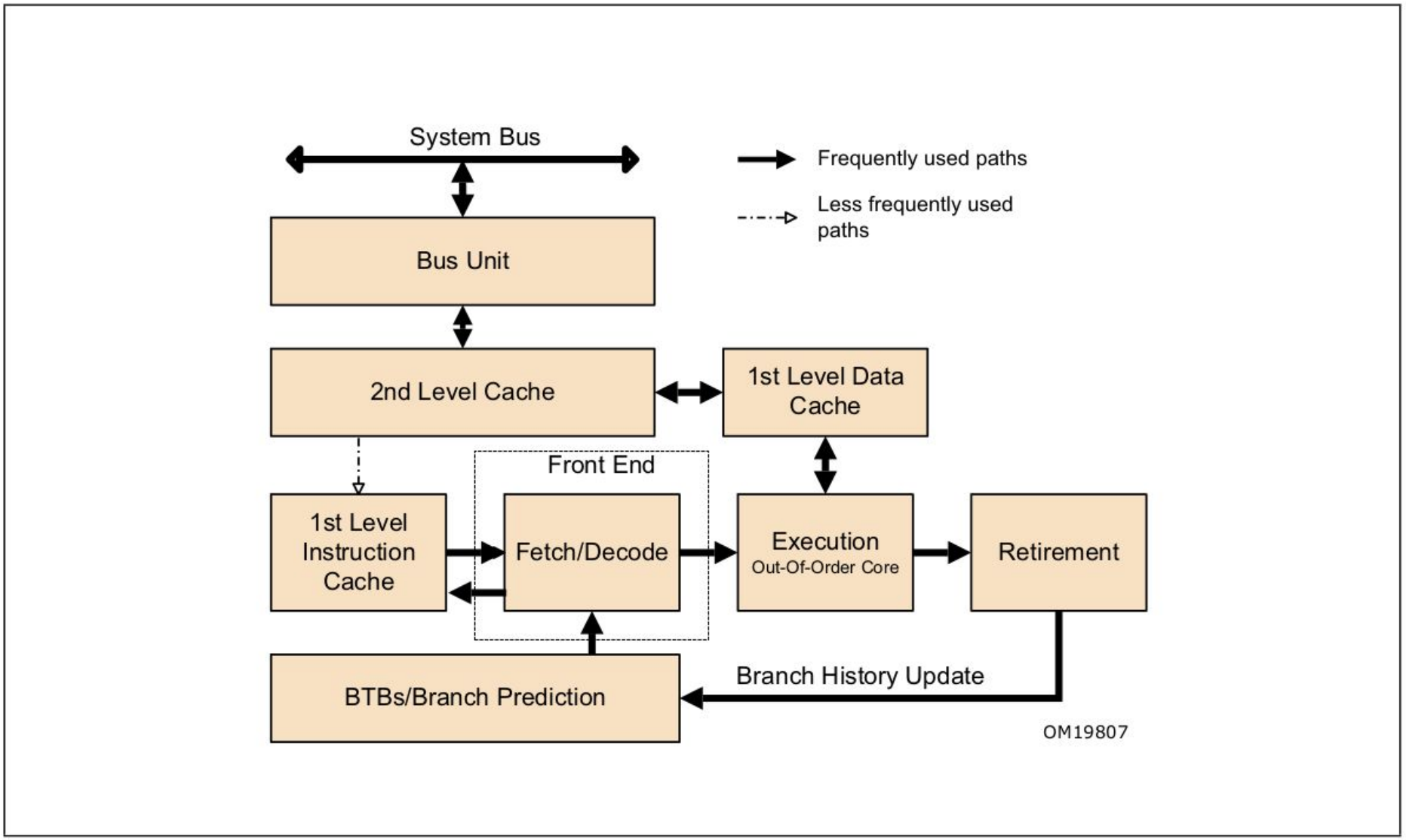
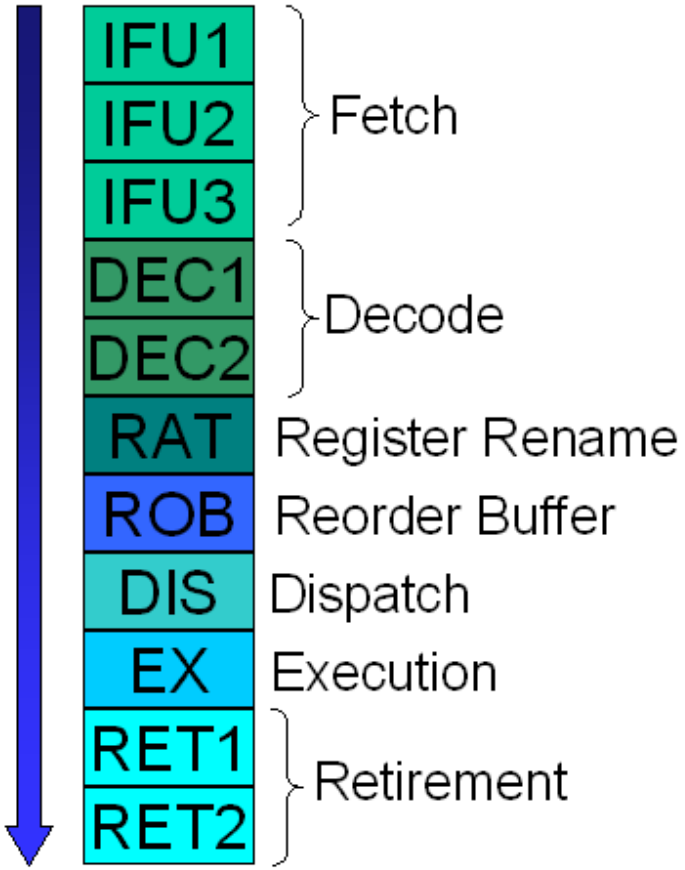


Figure 2-12. The Intel Pentium M Processor Microarchitecture

Pentium M: Pipeline

- Baseado na arquitetura P6, provavelmente com mais estágios
- Para comparação:
  - Pentium III (P6): 11 estágios
  - Pentium 4 (P7 inicial): 20 estágios
  - Pentium 4 (Prescott): 31 estágios!

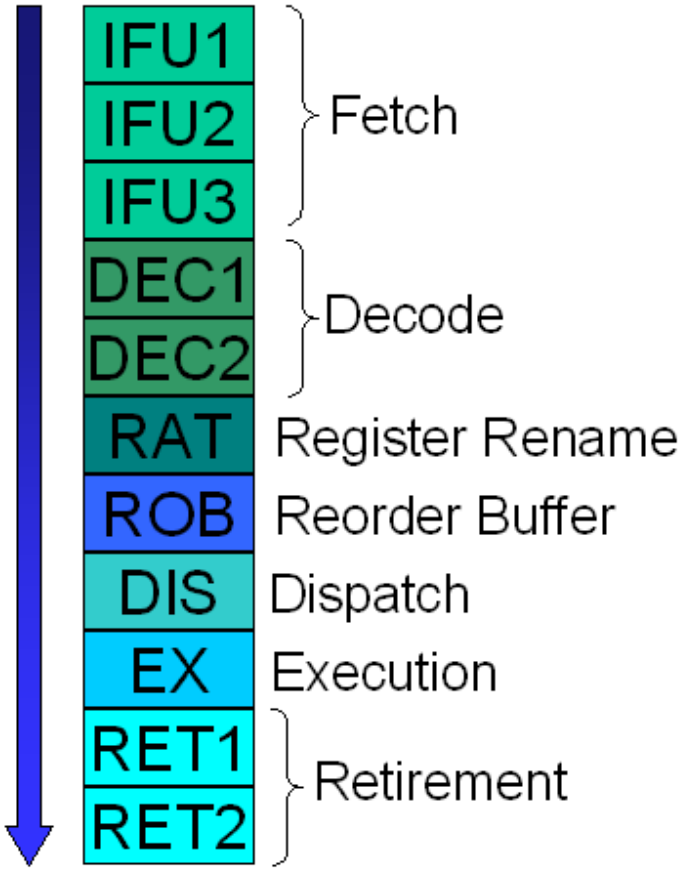
Pipeline: estágios do Pentium III



- IFU1: Loads one line (32 bytes, i.e., 256 bits) from L1 instruction cache and stores it in the Instruction Streaming Buffer.
- IFU2: Identifies the instructions boundaries within 16 bytes (128 bits). Since x86 instructions don't have a fixed length this stage marks where each instruction starts and ends within the loaded 16 bytes. If there is any branch instruction within these 16 bytes, its address is stored at the Branch Target Buffer (BTB), so the CPU can later use this information on its branch prediction circuit.
- IFU3: Marks to which instruction decoder unit each instruction must be sent. There are three different instruction decoder units, as we will explain later.
- DEC1: Decodes the x86 instruction into a RISC microinstruction (a.k.a. micro-op). Since the CPU has three instructions decode units, it is possible to decode up to three instructions at the same time.

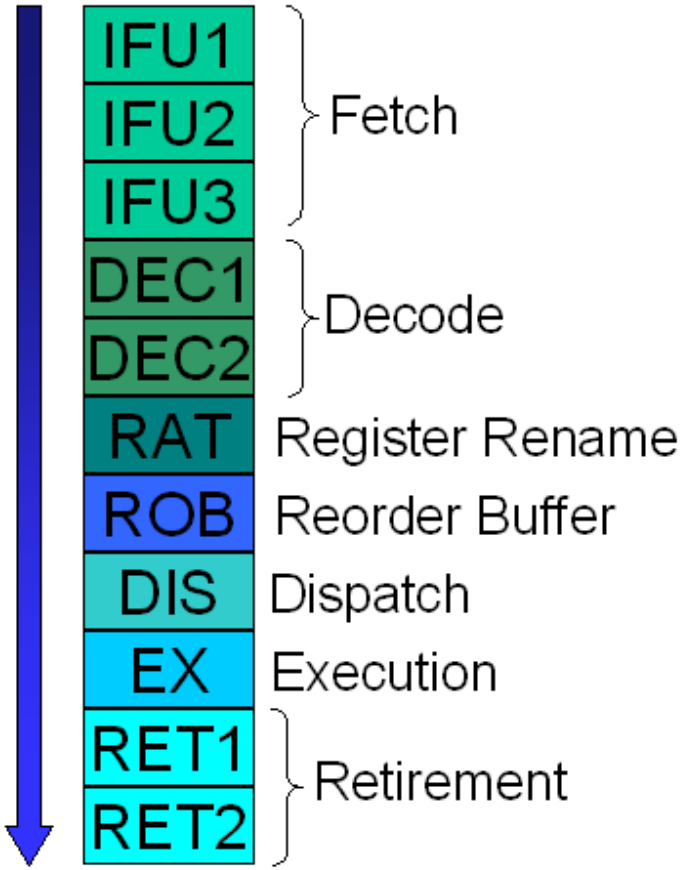
Fonte: http://www.hardwaresecrets.com/article/Inside-Pentium-M-Architecture/270

Pipeline: estágios do Pentium III



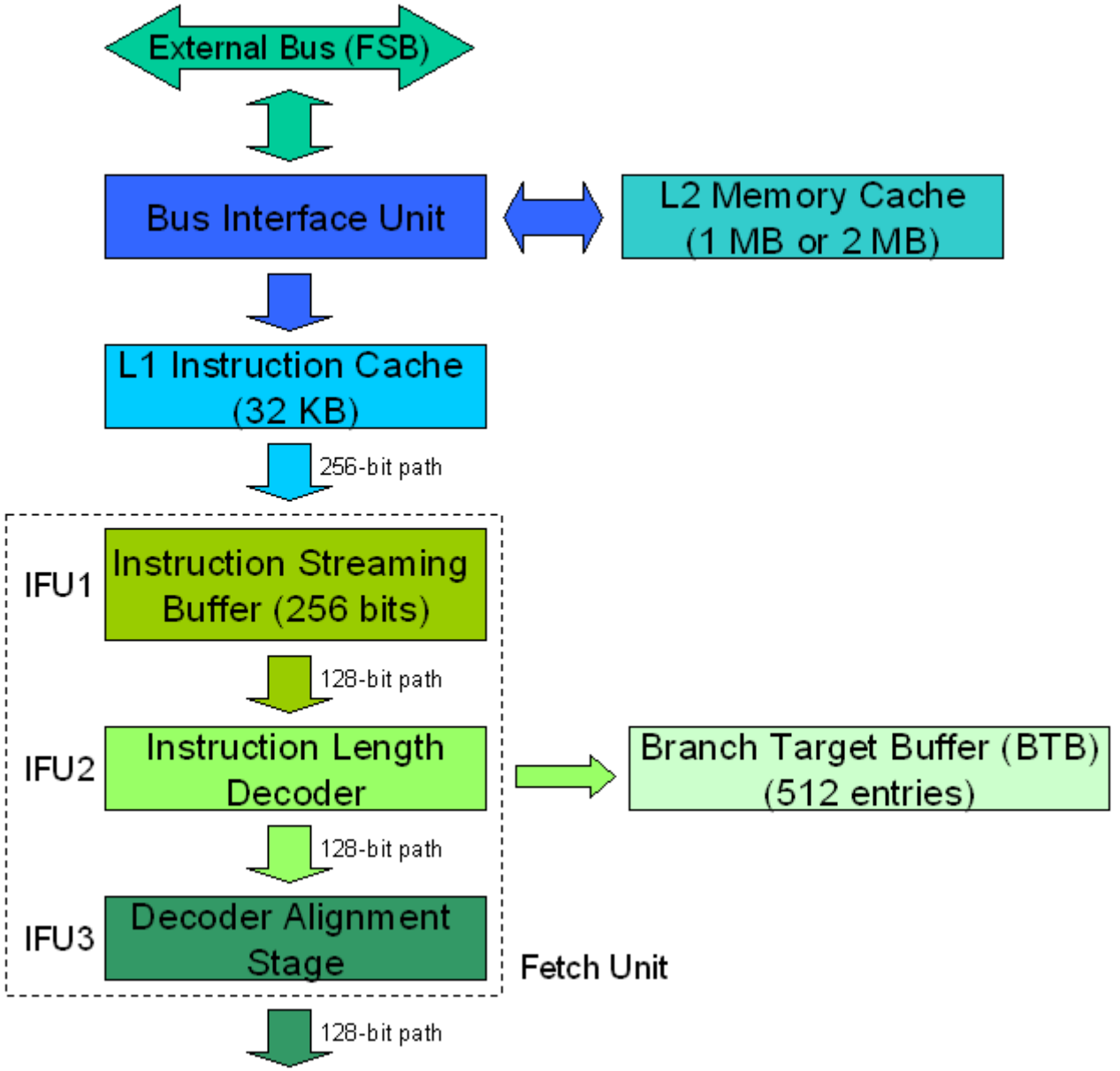
- DEC2: Sends the micro-ops to the Decoded Instruction Queue, which is capable to store up to six micro-ops. If the instruction was converted in more than six micro-ops, this stage must be repeated in order to catch the missing micro-ops.
- RAT: Since P6 microarchitecture implements out-of-order execution (OOO), the value of a given register could be altered by an instruction executed before its “correct” (i.e., original) place in the program flow, corrupting the data needed by another instruction. So, to solve this kind of conflict, at this stage the original register used by the instruction is changed to one of the 40 internal registers that P6 microarchitecture has.
- ROB: At this stage three micro-ops are loaded into the Reorder Buffer (ROB). If all data necessary for the execution of a micro-op are available and if there is an open slot at the Reservation Station micro-op queue, then the micro-op is moved to this queue.

Pipeline: estágios do Pentium III



- DIS: If the micro-op wasn't sent to the Reservation Station micro-op queue, this is done at this stage. The micro-op is sent to the proper execution unit.
- EX: The micro-op is executed at the proper execution unit. Usually each micro-op needs only one clock cycle to be executed.
- RET1: Checks at the Reorder Buffer if there is any micro-op that can be flagged as "executed".
- RET2: When all micro-ops related to the previous x86 instruction were already removed from the Reorder Buffer and all micro-ops related to the current x86 instruction were executed, these micro-ops are removed from the Reorder Buffer and the x86 registers are updated (the inverse process done at RAT stage). The retirement process must be done in order. Up to three micro-ops can be removed from the Reorder Buffer per clock cycle.

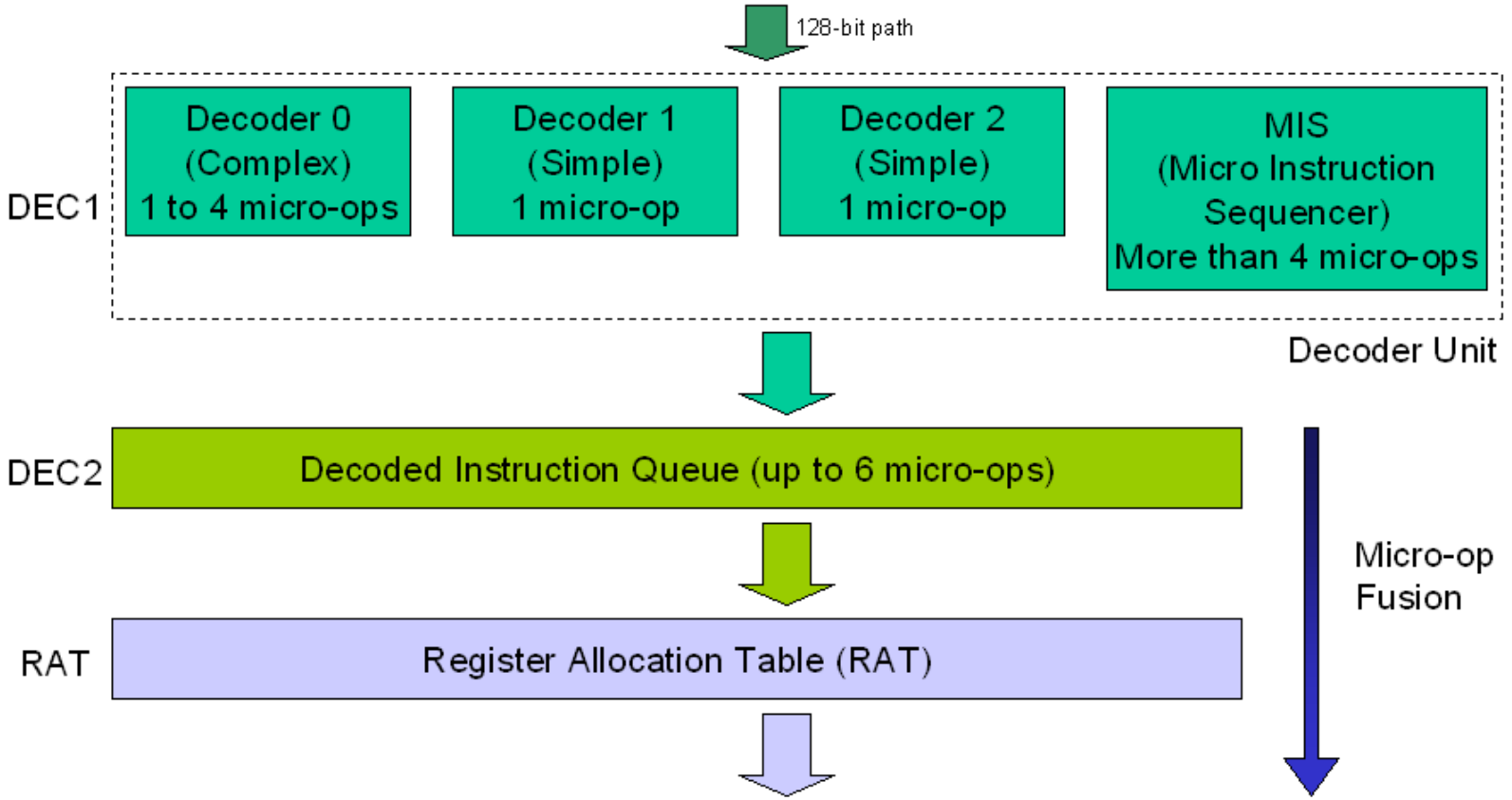
Fetch Unit: Detalhes



Micro-ops

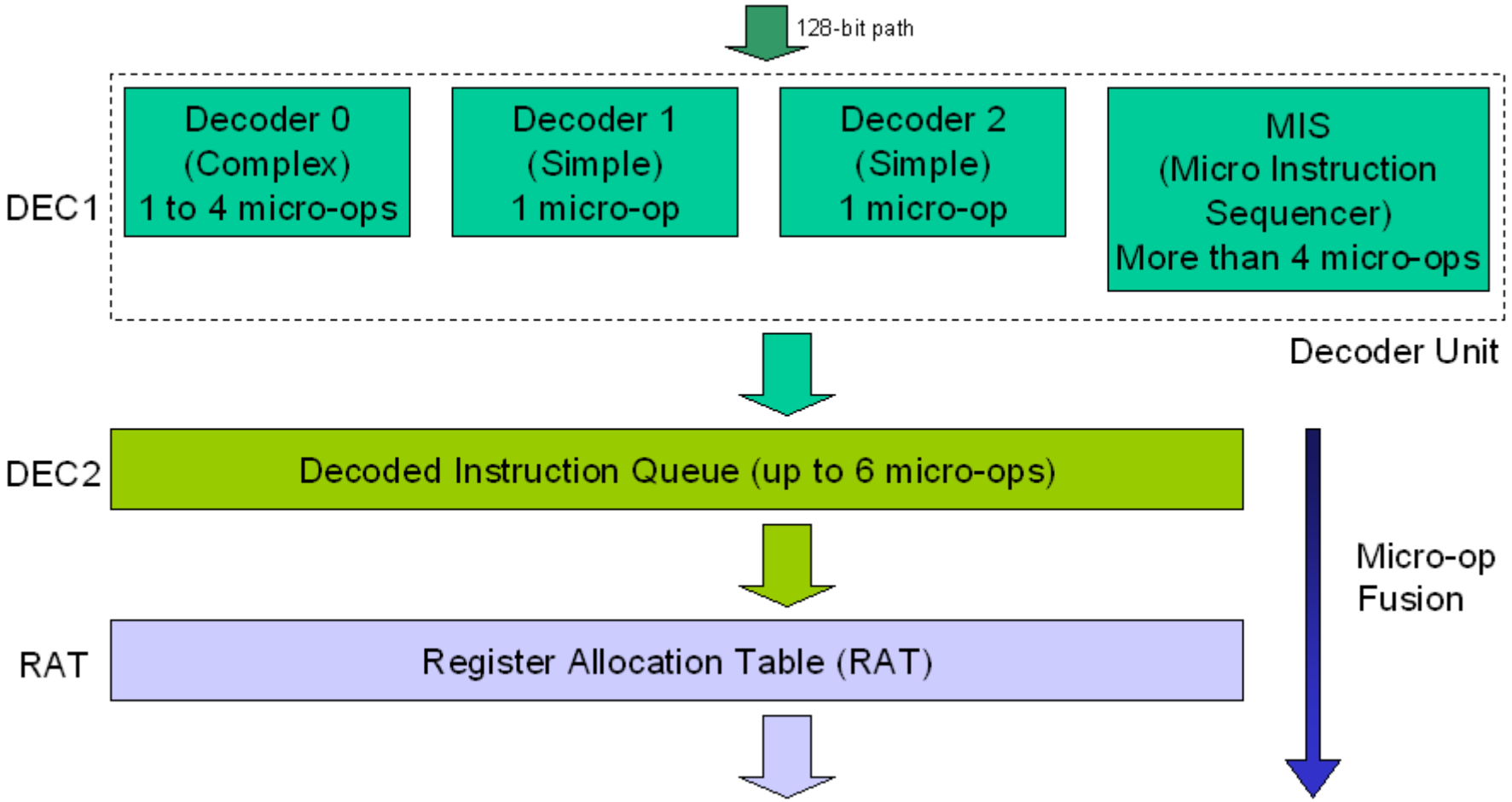
- A partir da arquitetura P6 (desde o Pentium Pro), os processadores x86 usam arquiteturas híbridas CISC/RISC
  - O processador aceita **apenas** instruções CISC (x86) no seu frontend
  - Internamente as instruções são convertidas em instruções RISC, chamadas de micro-ops por uma unidade chamada Instruction Decoder
  - Essas micro-ops não podem ser acessadas, e cada CPU usa seu próprio conjunto de micro-ops, que não são documentadas e são provavelmente incompatíveis com micro-ops de outras CPUs (ex. Micro-ops do PentiumM são diferentes das do Pentium 4, que são diferentes do Athlon64)
  - Dependendo da complexidade de uma instrução x86, ela pode ser convertida em várias micro-ops RISC

Pentium M: Instruction Decoder



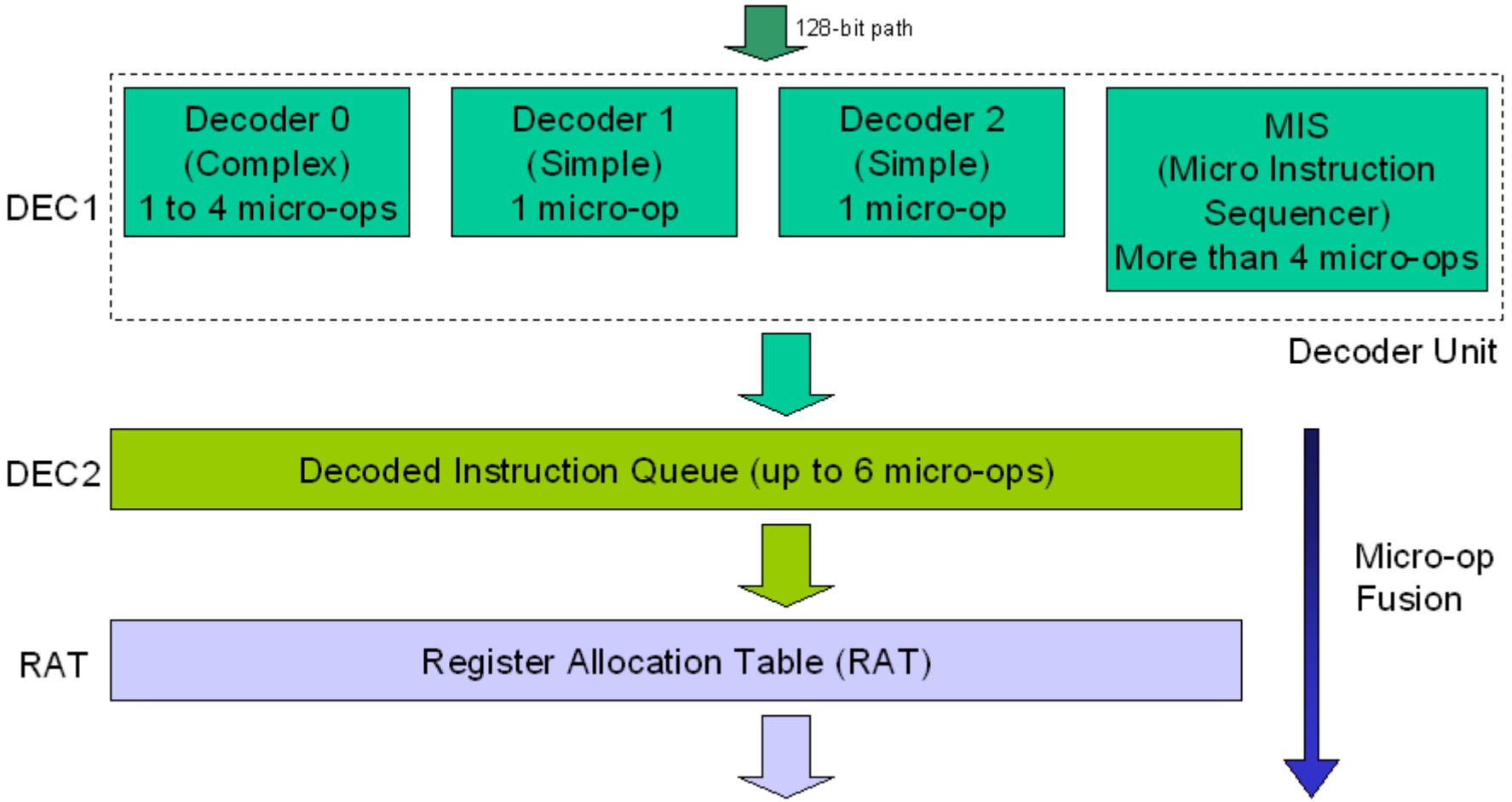
- As you can see, there are three decoders and a Micro Instruction Sequencer (MIS). Two decoders are optimized for simple instructions, which are the most used ones. This kind of instruction is converted in just one micro-op. One decoder is optimized for complex x86 instructions, which can be converted in up to four micro-ops. If the x86 instruction is too complex, i.e., it converts into more than four micro-ops, it is sent to the Micro Instruction Sequencer, which is a ROM memory containing a list of micro-ops that should replace the given x86 instruction.

Pentium M: Instruction Decoder



- The instruction decoder can convert up to three x86 instructions per clock cycle, one complex at Decoder 0 and two simple at decoders 1 and 2

Pentium M: Instruction Decoder



- The Decoded Instruction Queue can be feeded with up to six micro-ops per clock cycle, scenario that is reached when Decoder 0 sends four micro-ops and the other two decoders send one micro-op each – or when the MIS is used

Dealing with Very complex x86

- Micro Instruction Sequencer
  - Very complex x86 instructions that use the Micro Instruction Sequencer can delay several clock cycles to be decoded, depending on how many micro-ops will be generated from the conversion. Keep in mind that the Decoded Instruction Queue can hold only up to six micro-ops, so if more than six micro-ops are generated by the decoder plus MIS, another clock cycle is needed to send the current micro-ops present in the queue to the Register Allocation Table (RAT), empty the queue and accept the micro-ops that didn’t “fit” before.

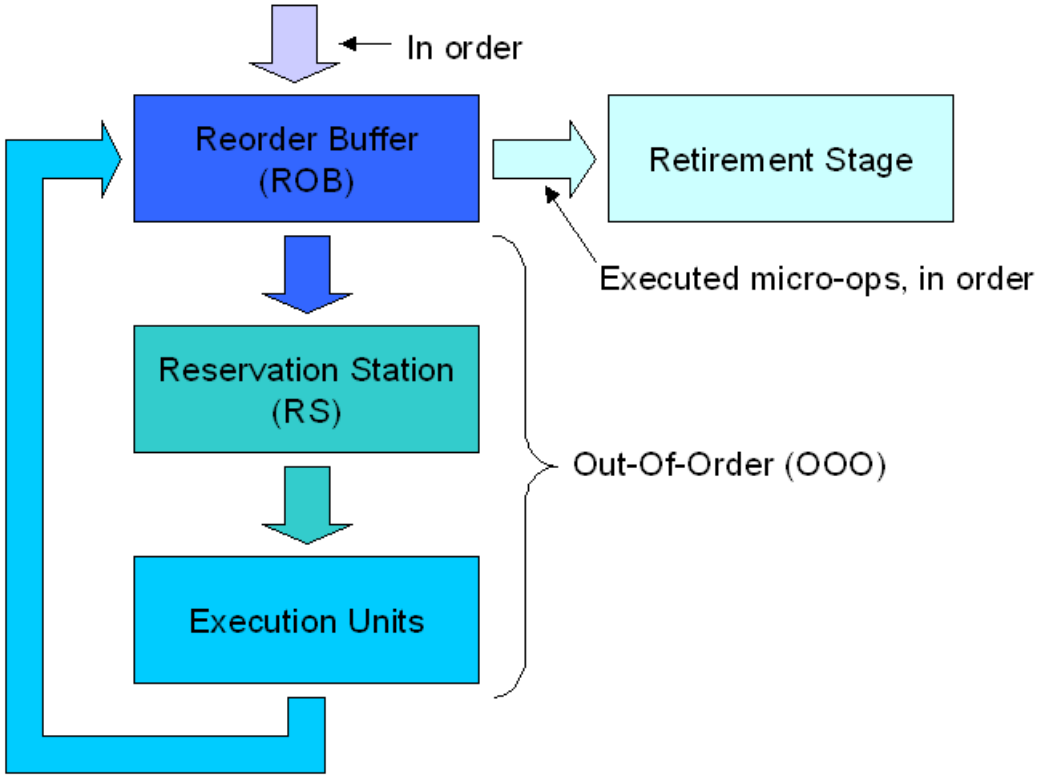
Micro-op fusion

- Pentium M uses a new concept to the P6 architecture that is called micro-op fusion. On Pentium M the decoder unit fuses two micro-ops into one. They will be de-fused only to be executed, at the execution stage.
- On P6 architecture, each microinstruction is 118-bit long. Pentium M instead of working with 118-bit micro-ops works with 236-bit long micro-ops that are in fact two 118-bit micro-ops.
- Keep in mind that the micro-ops continue to be 118-bit long; what changed is that they are transported in groups of two.
- This idea behind this approach was to save energy and increase performance. It is faster to send one 236-bit micro-op than two 118-bit micro-ops. Also the CPU will consume less power, since less micro-ops will be circulating inside of it.

Register Allocation Table (RAT)

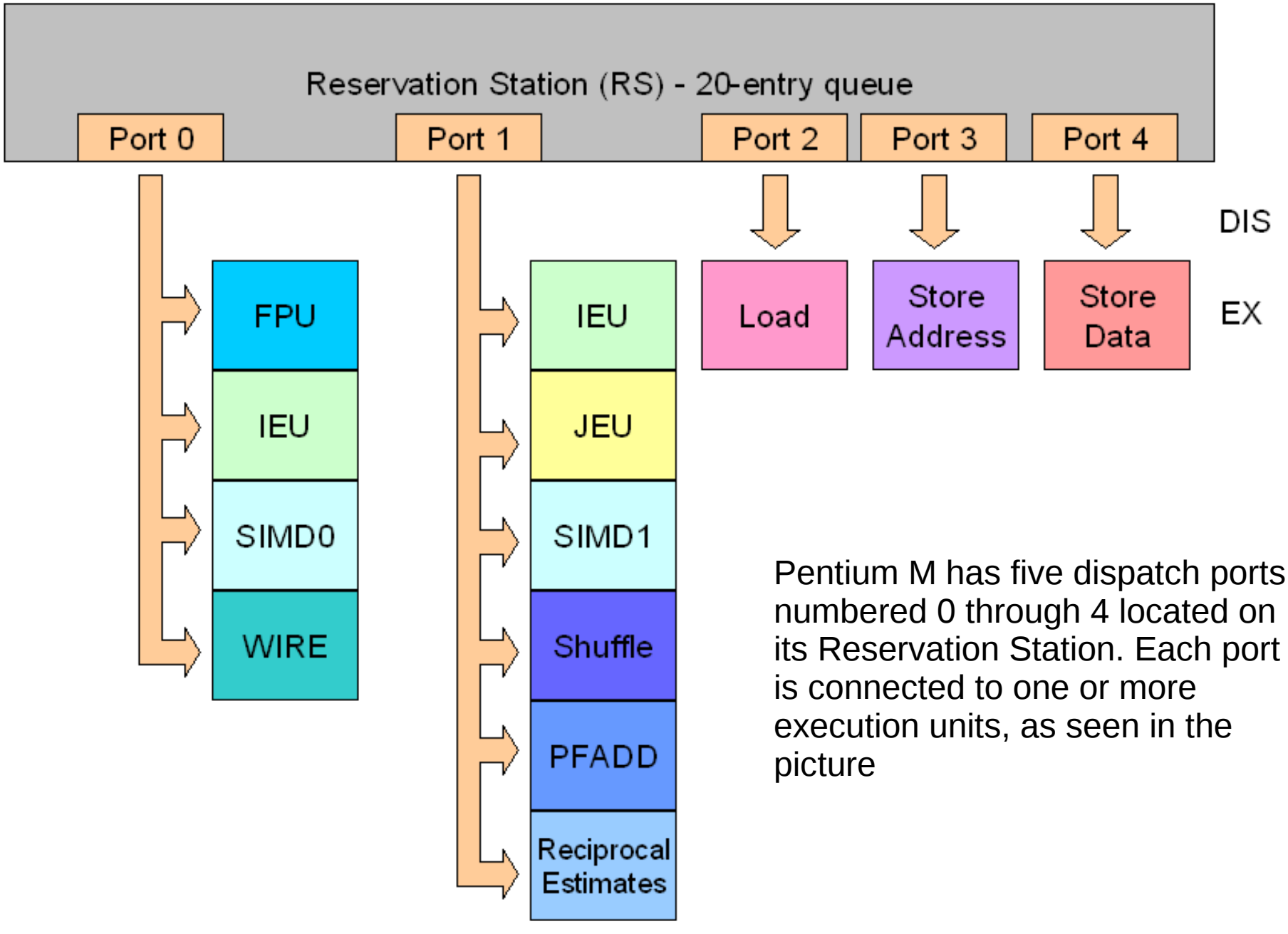
- Fused micro-ops are then sent to the Register Allocation Table (RAT).
  - CISC x86 architecture has only eight 32-bit registers (EAX, EBX, ECX, EDX, EBP, ESI, EDI and ESP). This number is simply too low, especially because modern CPUs can execute code out-of-order, what would “kill” the contents of a given register, crashing the program.
  - So, at this stage, the processor changes the name and contents of the registers used by the program into one of the 40 internal registers available (each one of them is 80-bit wide, thus accepting both integer and floating-point data), allowing the instruction to run at the same time of another instruction that uses the exact same standard register, or even out-of-order, i.e., this allows the second instruction to run before the first instruction even if they mess with the same register.

The Reorder Buffer



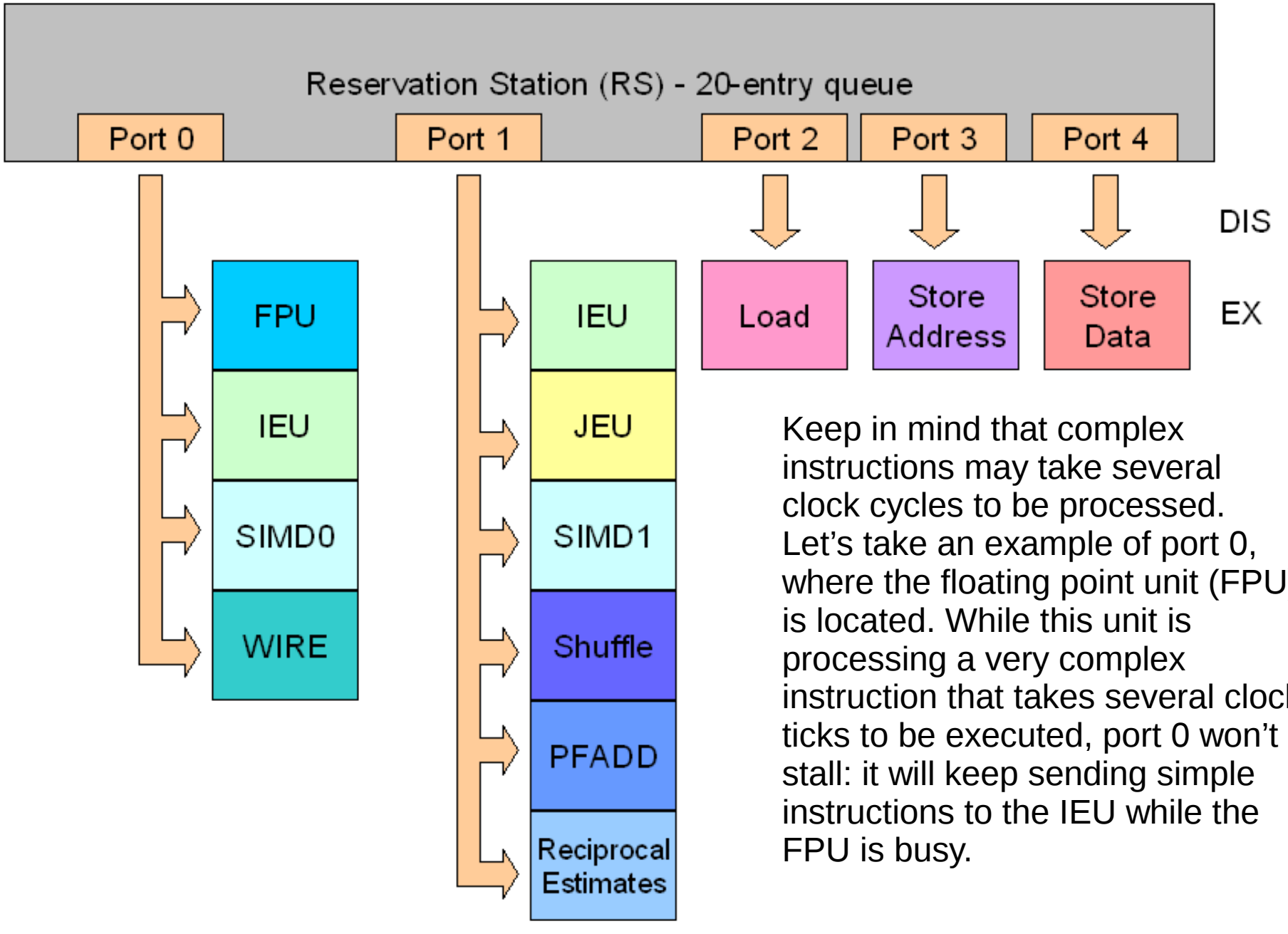
- So far the x86 instructions and the micro-ops resulted from them are transferred between the CPU stages in the same order they appear on the program being run.
- Arriving at the ROB, micro-ops can be loaded and executed out-of-order by the execution units. After being executed, the instructions are sent back to the Reorder Buffer. Then at the Retirement stage, executed micro-ops are pulled out of the Reorder Buffer at the same order they entered it, i.e., they are removed in order.
- Pentium M uses fused micro-ops (i.e., carries two micro-ops together) from the Decode Unit up to the dispatch ports located on the Reservation Station. The Reservation Station dispatches each micro-op individually (defused).

Dispatch ports and Execution Units



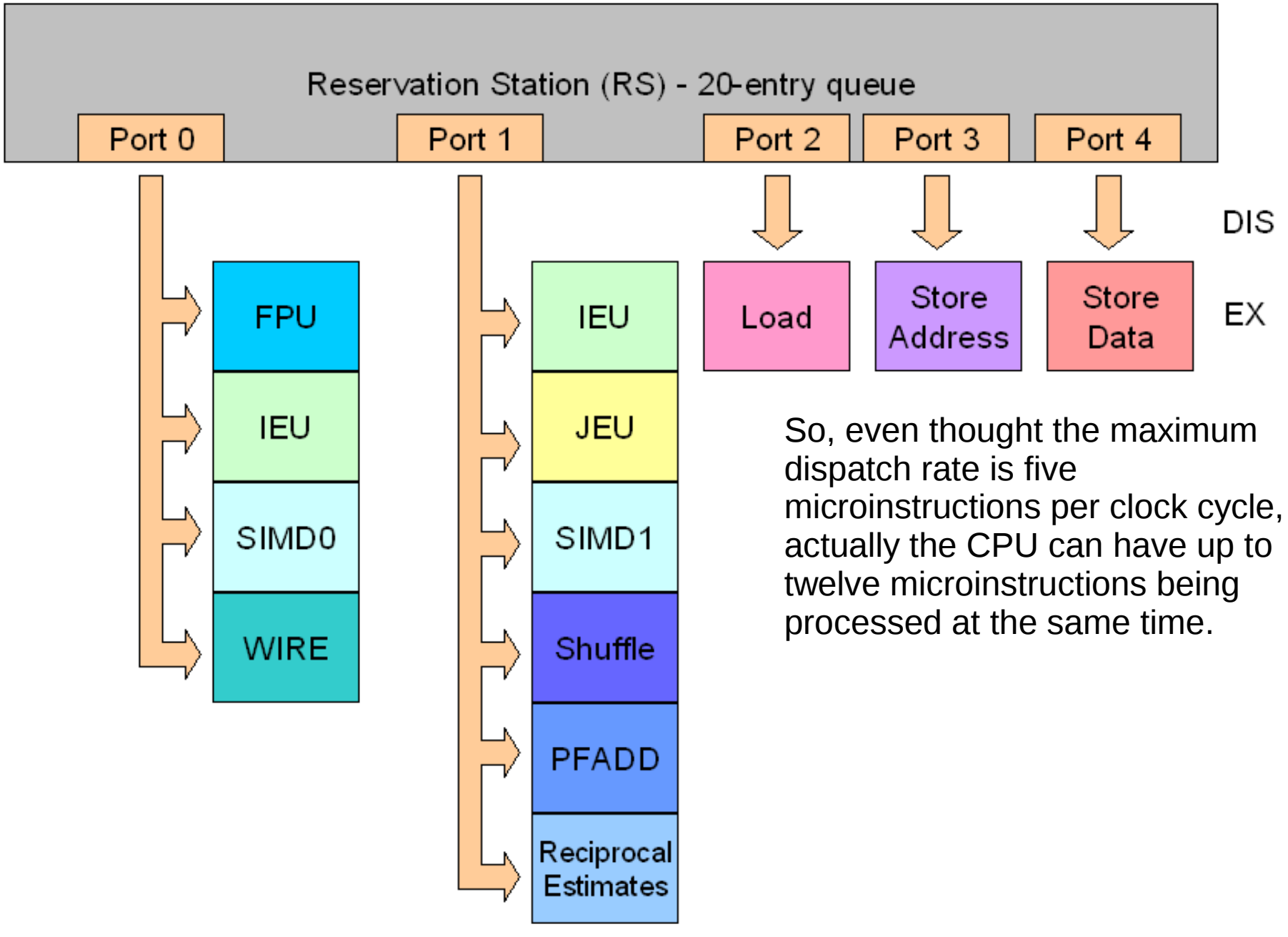
Pentium M has five dispatch ports numbered 0 through 4 located on its Reservation Station. Each port is connected to one or more execution units, as seen in the picture

Execution Units: Explanation



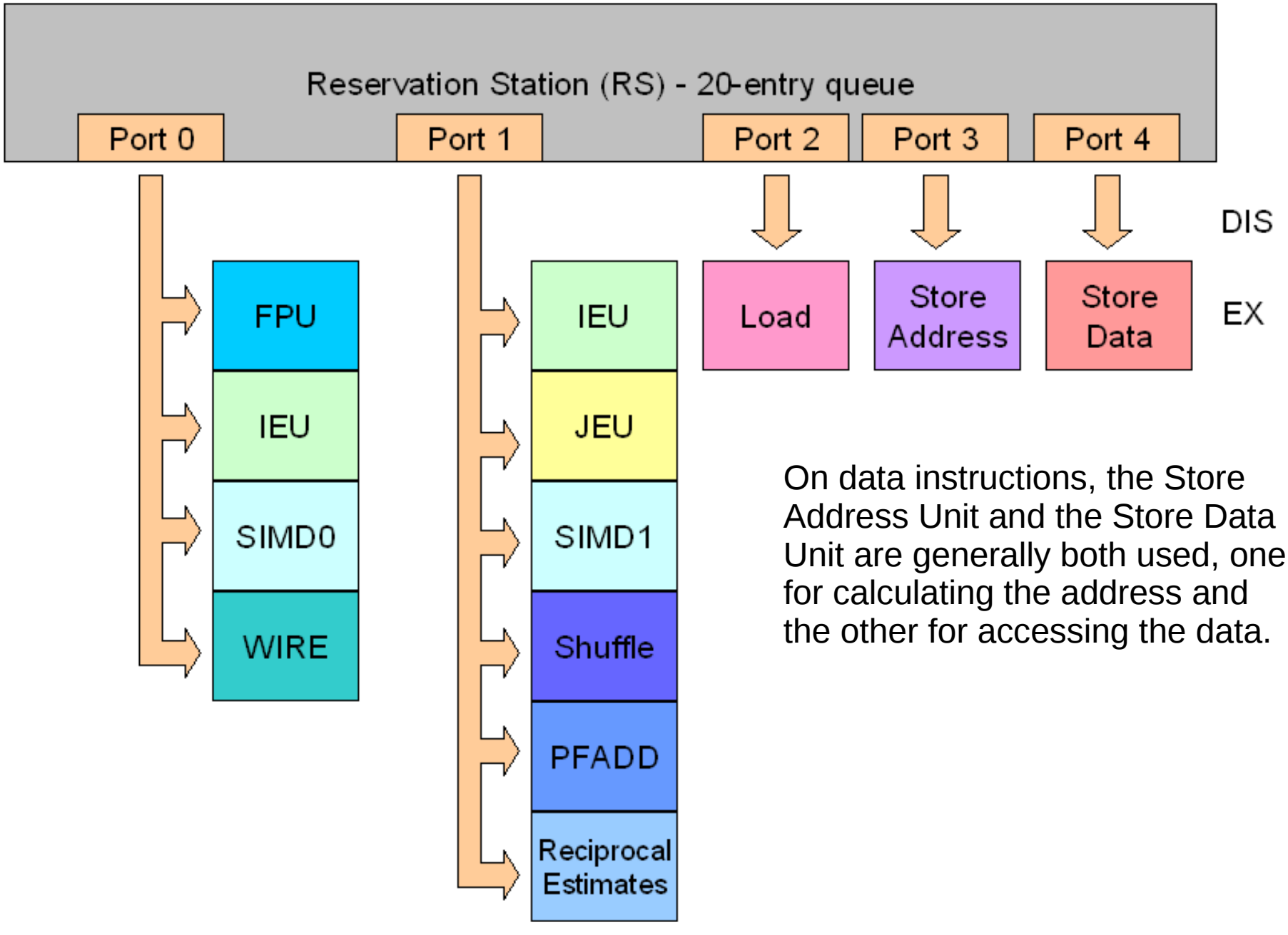
Keep in mind that complex instructions may take several clock cycles to be processed. Let's take an example of port 0, where the floating point unit (FPU) is located. While this unit is processing a very complex instruction that takes several clock ticks to be executed, port 0 won't stall: it will keep sending simple instructions to the IEU while the FPU is busy.

Execution Units: Explanation



So, even thought the maximum dispatch rate is five microinstructions per clock cycle, actually the CPU can have up to twelve microinstructions being processed at the same time.

Execution Units: Explanation



On data instructions, the Store Address Unit and the Store Data Unit are generally both used, one for calculating the address and the other for accessing the data.

Retirement

- After each micro-op is executed, it returns to the Reorder Buffer, where its flag is set to “executed”. Then at the Retirement Stage the micro-ops that have their “executed” flag on are removed from the Reorder Buffer on its original order (i.e., the order they were decoded) and then the x86 registers are updated (the inverse step of register renaming stage). Up to three micro-ops can be removed from the Reorder Buffer per clock cycle. After this the instruction was fully executed.

PentiumM: Enhanced SpeedStep

- SpeedStep Technology was created to increase battery life and was first introduced with Pentium III M processor. This first version of SpeedStep Technology allowed the CPU to switch between two clock frequencies on the fly: Low Frequency Mode (LFM), which maximized battery life, and High Frequency Mode (HFM), which allowed you to run your CPU at its maximum speed. The CPU had two clock multiplier ratios and what it did was to change the ratio it was using. The LFM ratio was factory-lock and you couldn't change it.
- Pentium M introduced Enhanced SpeedStep Technology, which goes beyond that, by having several other clock and voltage configurations between LFM (which is fixed at 600 MHz) and HFM (which is the CPU full clock).
- Example

Voltage	Clock
1.484V	1,6GHz
1,42	1,4
1,276	1,2
1,1164	1
1,036	800 MHz
0,956	600

PentiumM: Enhanced SpeedStep

- Each Pentium M model has its own voltage/clock table. It is very interesting to notice that it is not only about lowering the clock rate when you don’t need so much processing power from your laptop, but also about lowering its voltage, which helps a lot to lower battery consumption.
- Enhanced SpeedStep Technology works by monitoring specific MSRs (Model Specific Registers) from the CPU called Performance Counters. With this information, the CPU can lower or raise its clock/voltage depending on CPU usage. Simply put, if you increase CPU usage, it will increase its voltage/clock, if you lower the CPU usage, it will lower its voltage/clock.
- Enhanced SpeedStep was just one of the several enhancements done on Pentium M microarchitecture in order to increase battery life. A good example was done on the execution units. On other processors, the same power line feeds all execution units. So it is not possible to turn off an idle execution unit on Pentium 4, for example. On Pentium M execution units have different power lines, making the CPU capable of turning off idle execution units. For example, Pentium M detects in advance if a given instruction is an integer one (“regular instruction”), disabling the units and datapaths not needed to process that instruction, if they are idle, of course.

MicroArquitetura x86

Intel Core

MicroArquitetura IntelCore

- Exemplos:
  - 2 Cores:
    - Intel Core 2 Extreme X6800
    - Intel Core 2 Duo
    - Intel Xeon Séries 3000, 5100
  - 4 cores:
    - Intel Core 2 Extreme quad-core
    - Intel Core 2 Quad
    - Intel Xeon Séries 3200, 5300

Características

- Evolução da arquitetura Pentium M
- Pensada desde o início para múltiplos cores
  - Cache L2 compartilhado entre os cores:

Vantagens L2 compartilhado

The problem with separate L2 caches is 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 through 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.

Melhoras na Prefetch Unit

Intel also has 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.

Melhoras nos Instruction Decoder

- Macro-Fusion: quando micro-ops ficam “maiores” que instruções x86...
  - Macro-fusion is the ability of joining two x86 instructions together into a single micro-op
  - Improves the CPU performance and lowers the CPU power consumption, since it will execute only one micro-op instead of two.
  - Used in compare and conditional branching instructions (CMP, TEST and J*cc*)

Macro-fusion: exemplo

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

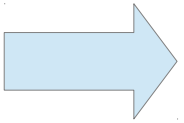
What this does is to load the 32-bit register EAX with data contained in memory position 1, compare its value with data contained in memory position 2 and, if they are different (jne = jump if not equal), the program goes to address “target”, if they are equal, the program continues on the current position.

Macro-fusion: exemplo

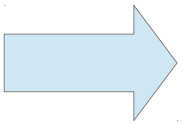
load eax, [mem1]

cmp eax, [mem2]

jne target



Micro-op1 (load eax with contents at mem1 addr)



“Micro”-op2 (compare eax with contents at mem2 addr and jump to target if equals)

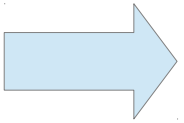
With macro-fusion the comparison (cmp) and branching (jne) instructions will be merged into a single micro-op. As we can see, we saved one instruction. The less instructions there are to be executed, the faster the computer will finish the execution of the task and also less power is generated.

Macro-fusion: exemplo

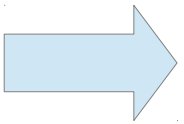
load eax, [mem1]

cmp eax, [mem2]

jne target



Micro-op1 (load eax with contents at mem1 addr)



“Micro”-op2 (compare eax with contents at mem2 addr and jump to target if equals)

With macro-fusion the comparison (cmp) and branching (jne) instructions will be merged into a single micro-op. 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.

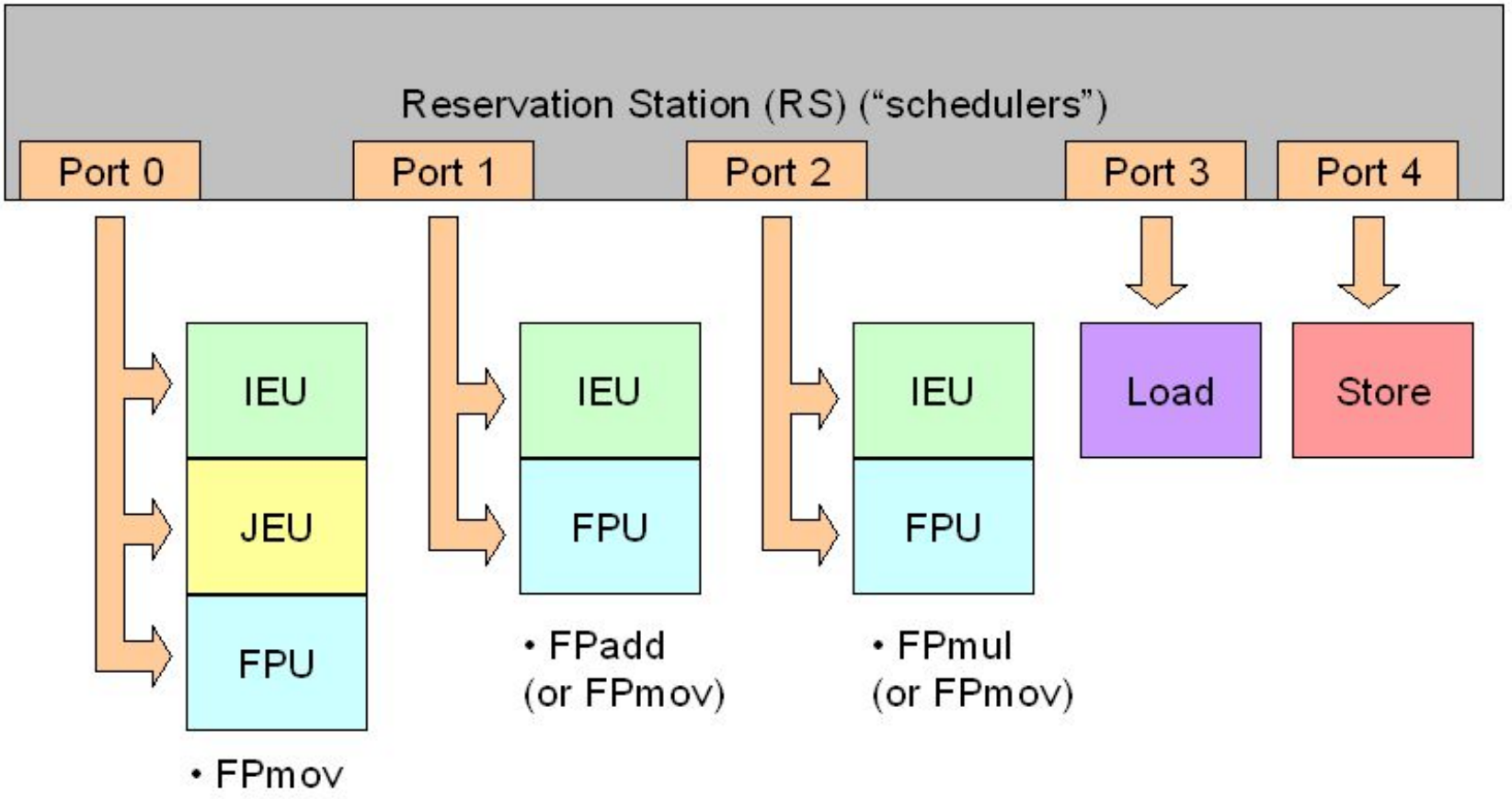
Mais instructions decoders

- 4 ao invés de 3 (Pentium M)
- 5 instruções fetched da fila de instruções por vez (por causa da macro-fusion)
  - 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.

Execution Units

- O Pentium M tinha 5 portas de execução, mas só 2 eram ligadas a unidades de execução (ver slide acima), sendo 3 delas ocupadas com unidades de operações de memória
- Arquitetura Core também tem 5 portas, mas 3 são ligadas a unidades de execução, uma vez que a unidade de geração de endereço não existe mais (funcionalidade incorporada nas unidades Load-Store)

Execution Units – Intel Core



Execution Units – Intel Core

- **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 FPadd- 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.

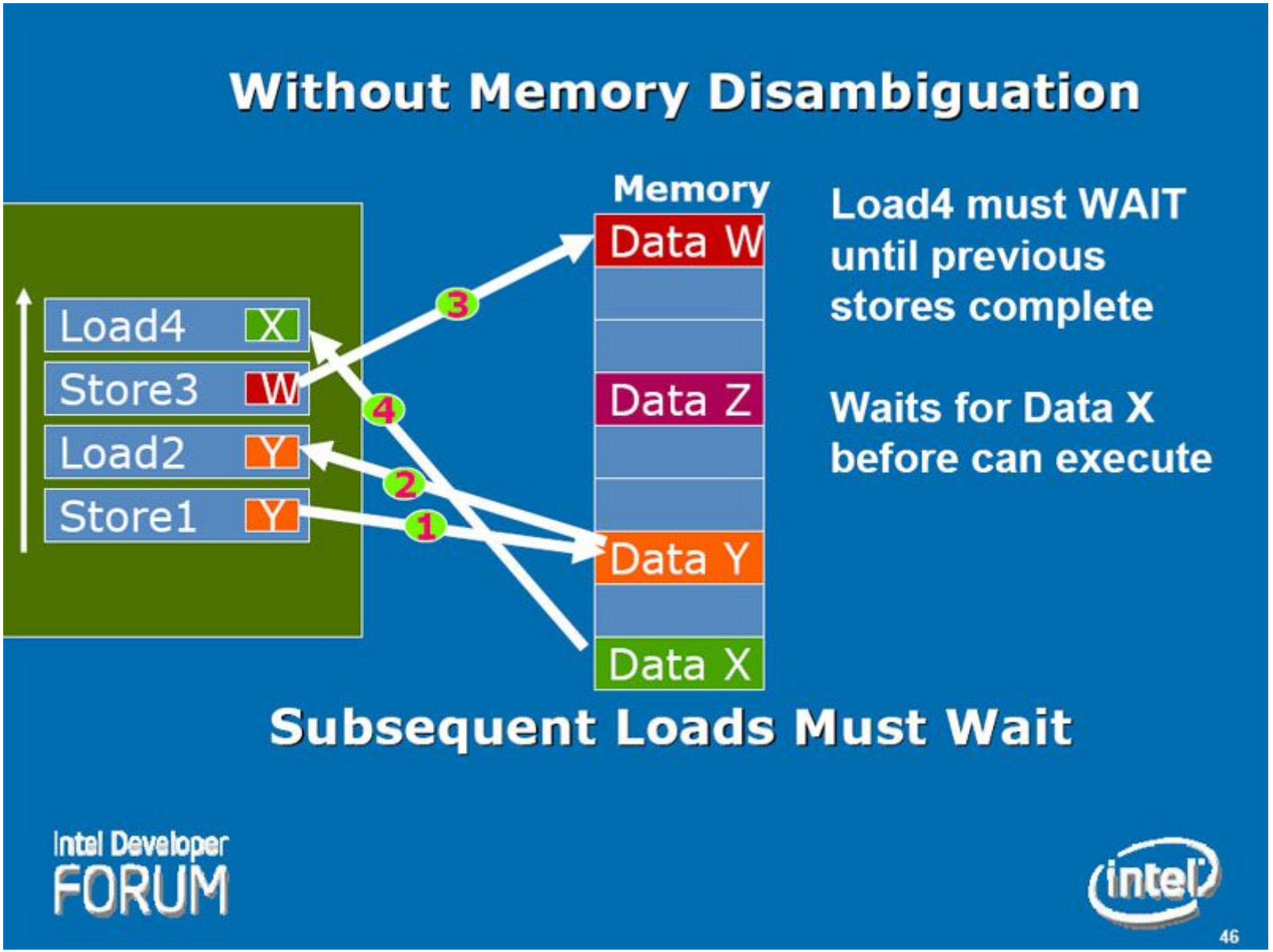
128-Bit Internal Datapath

- 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”.

Memory Disambiguation (Dataflow Analysis)

- 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).
- In the following slide 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.

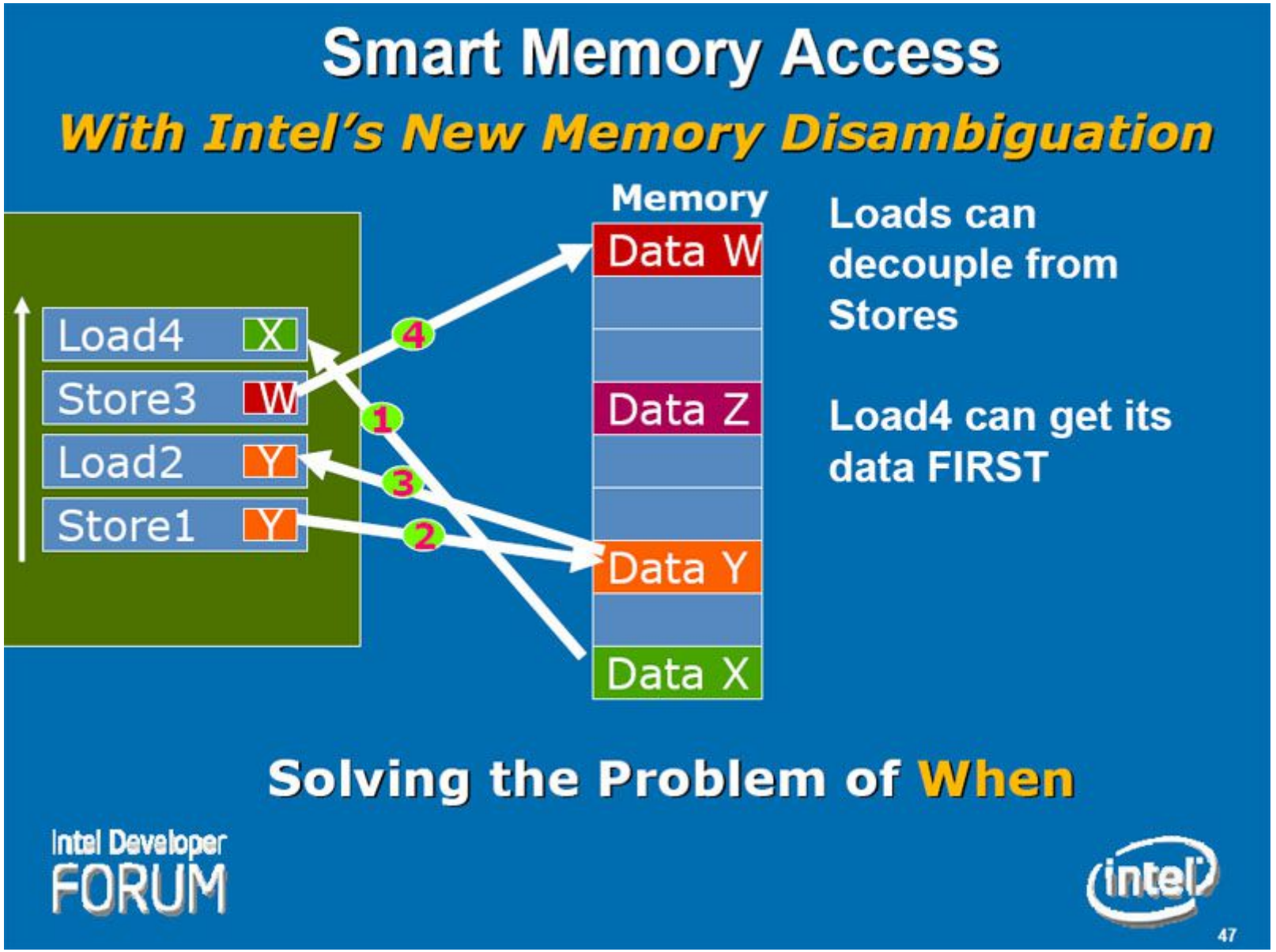
Memory Disambiguation (Dataflow Analysis)



Memory Disambiguation (Dataflow Analysis)

- In the next slide you see how the program shown in 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.
- 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 thatdepend on that result) would have to wait all other instructions shown in 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.

Memory Disambiguation (Dataflow Analysis)



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.

Intel Core: Overview

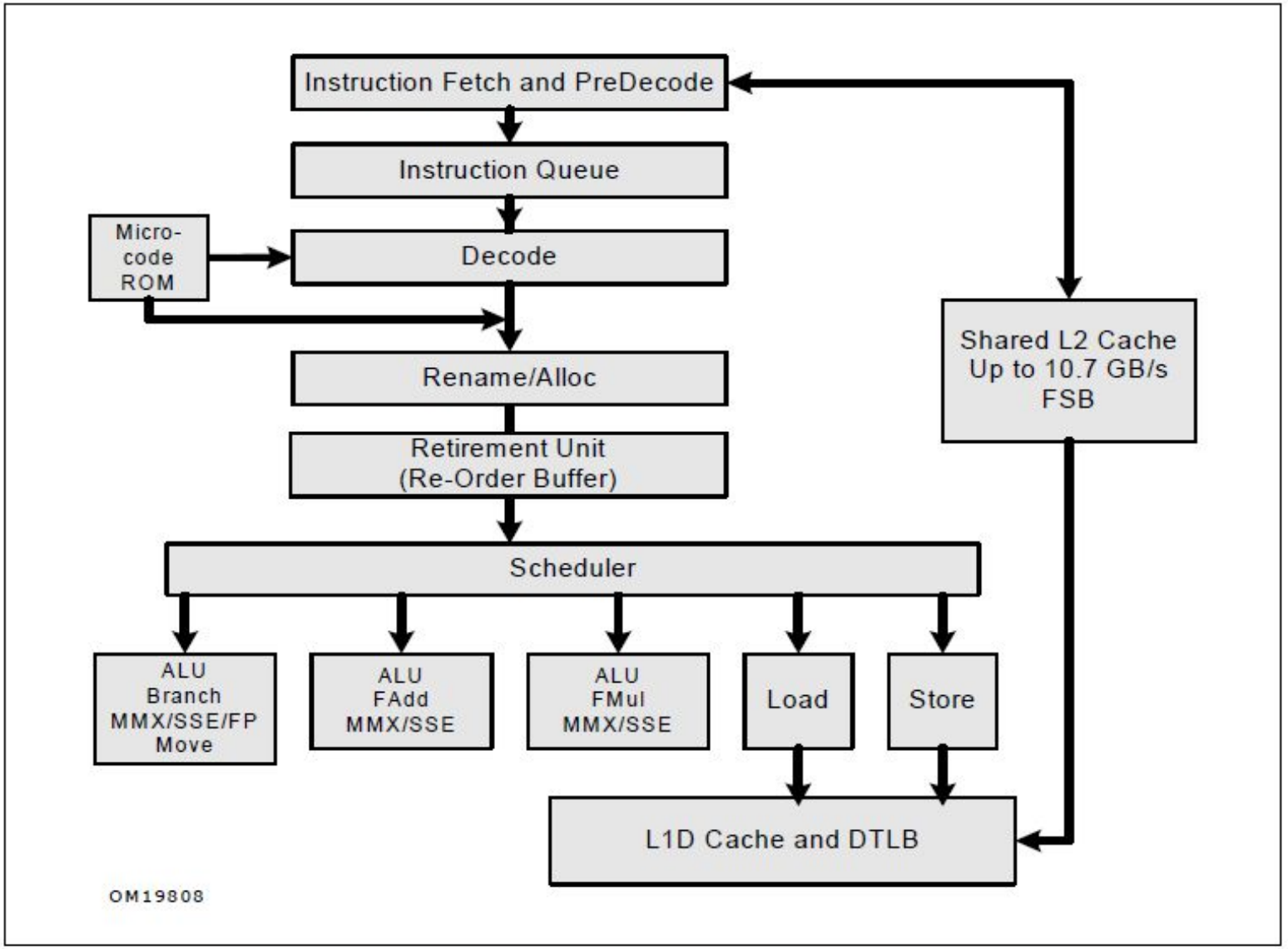
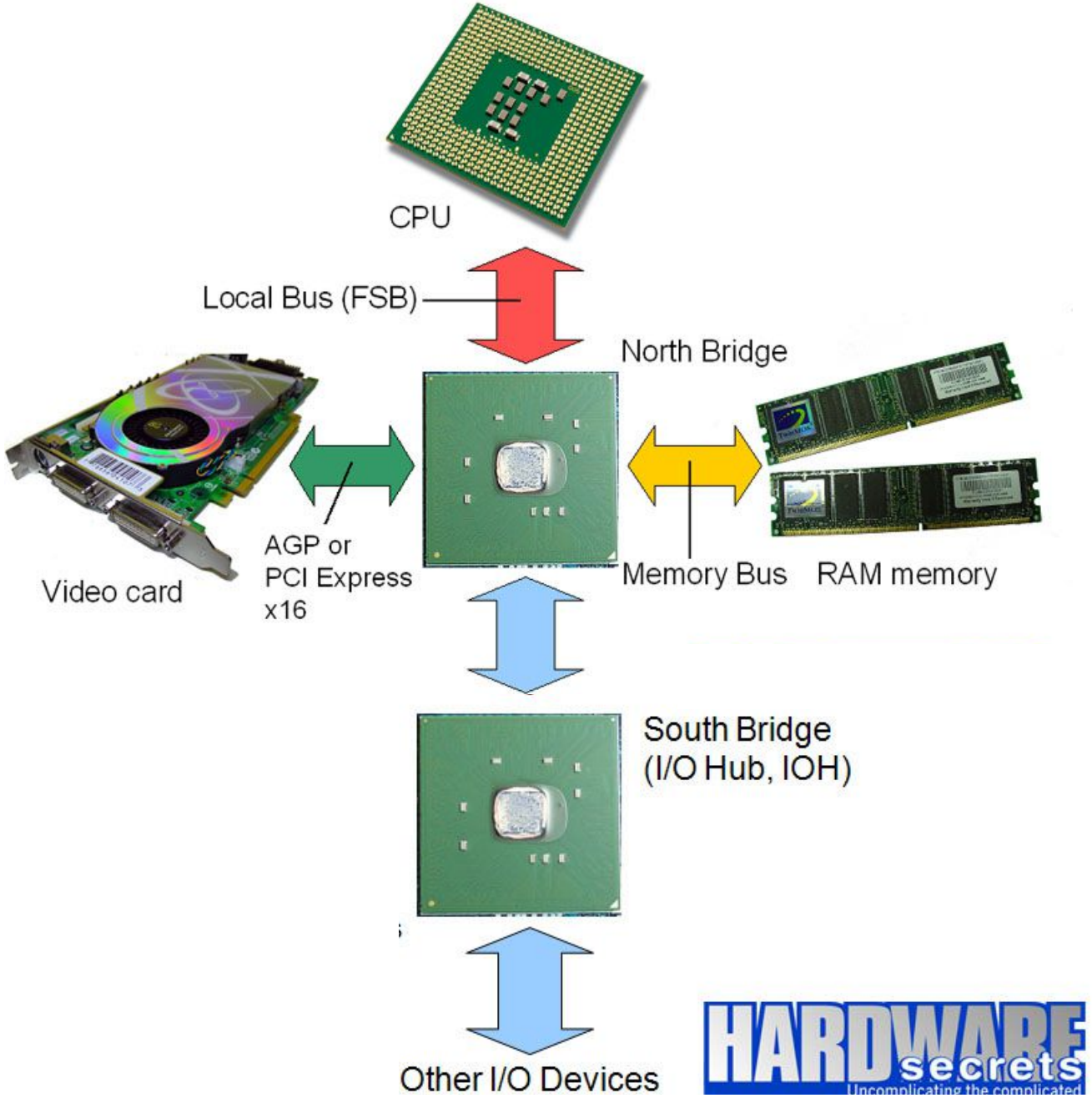


Figure 2-2. Intel Core Microarchitecture Pipeline Functionality

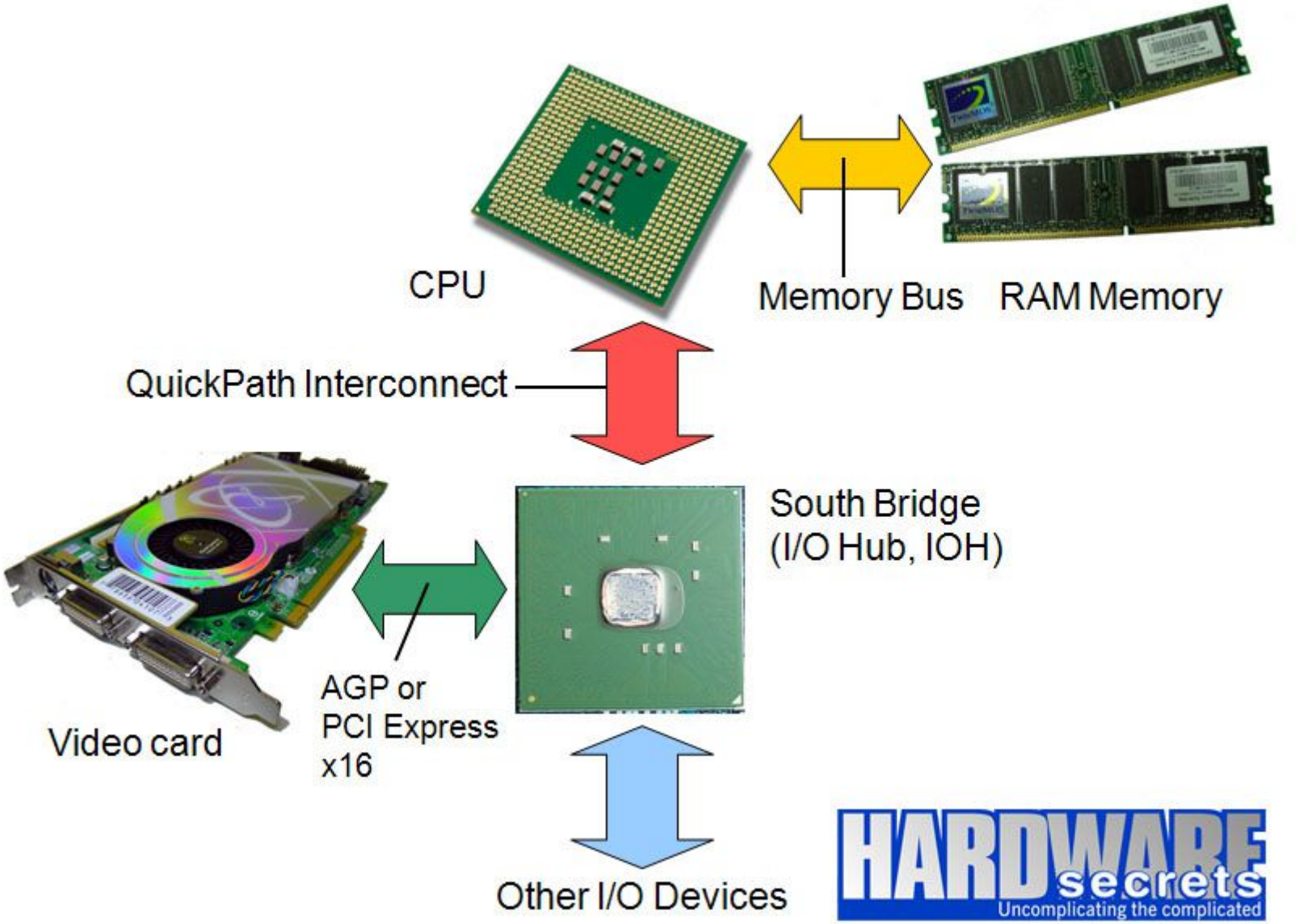
MicroArquitetura x86

Nehalem

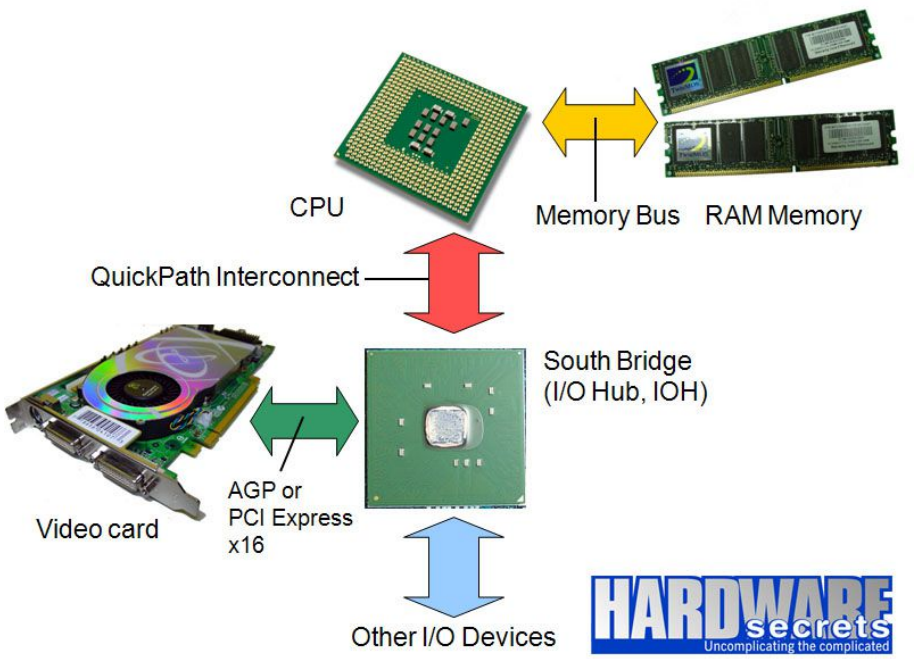
Arquitetura tradicional x86



Arquitetura x86 Nehalem em diante



Arquitetura x86 Nehalem em diante

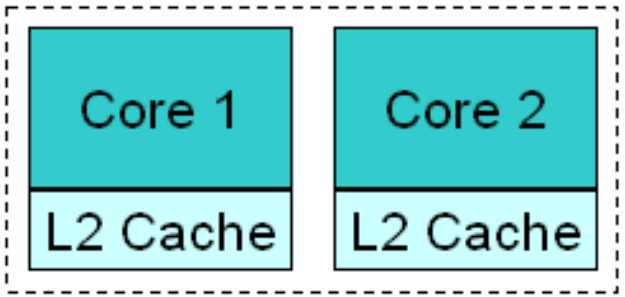


- A AMD que introduziu o controlador de memória integrado ao processador, com a arquitetura AMD64
- O barramento equivalente ao QuickPath Interconnect da AMD é o HyperTransport
- Arquitetura AMD64
  - Implementada inicialmente no Opteron, em 2003.
- Arquitetura Intel Nehalem
  - Implementada inicialmente no Core i7, em 2008.

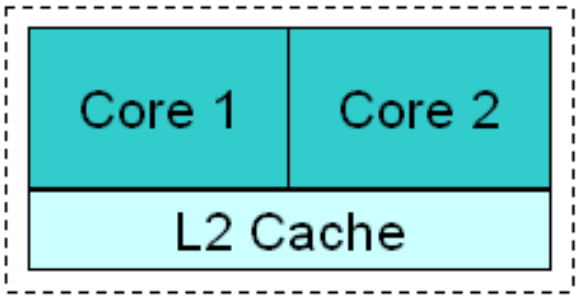
Nehalem Cache

- No Nehalem os cores voltarem a ter um cache L2 exclusivo
- Mas foi introduzido um cache L3 (maior), compartilhado
- Essa solução também já era usada pela AMD, nas suas CPUs Phenom

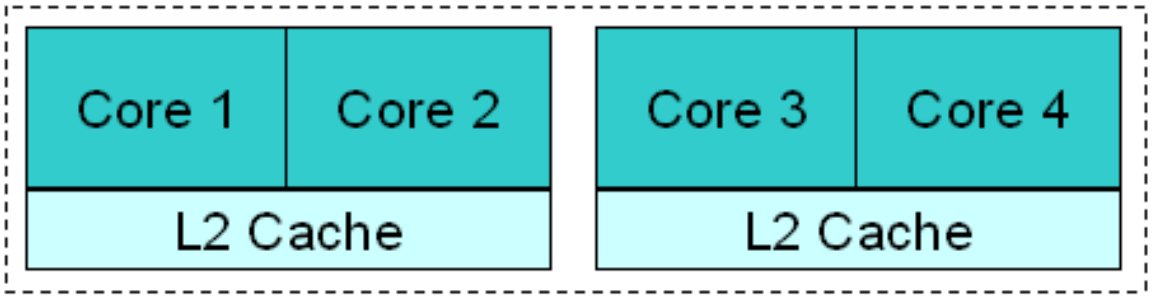
Arquiteturas de Cache



Separated Caches
- AMD CPUs
- Pentium D

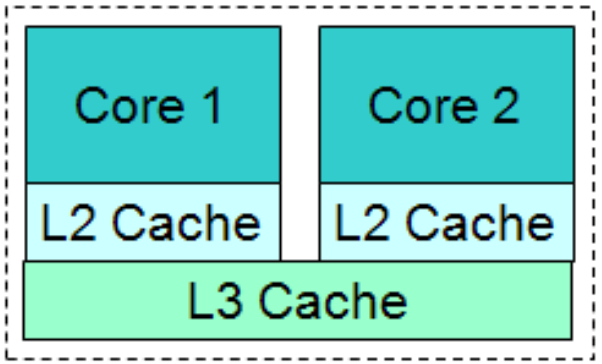


Shared Cache
- Core Duo
- Core 2 Duo

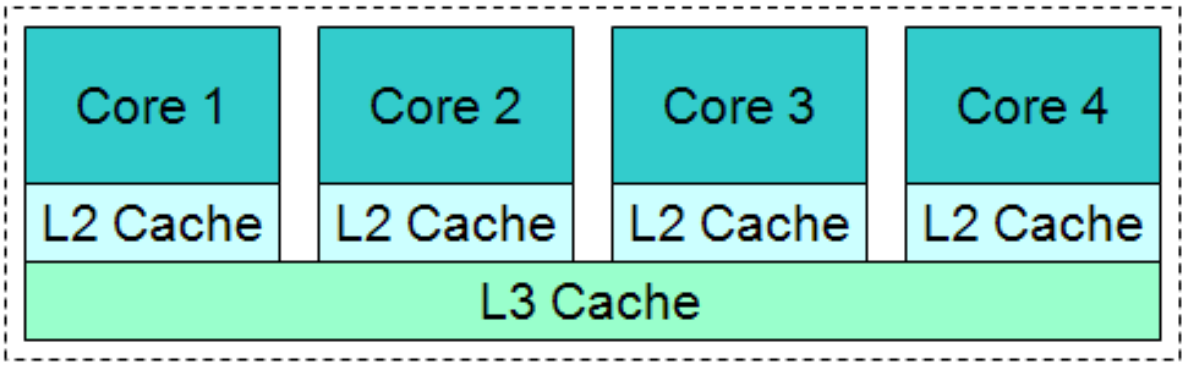


Current Intel Quad-Core CPUs
- Core 2 Quad
- Core 2 Extreme QX

Arquiteturas de Cache



- AMD K10-based dual-core CPU's
- Intel Nehalem



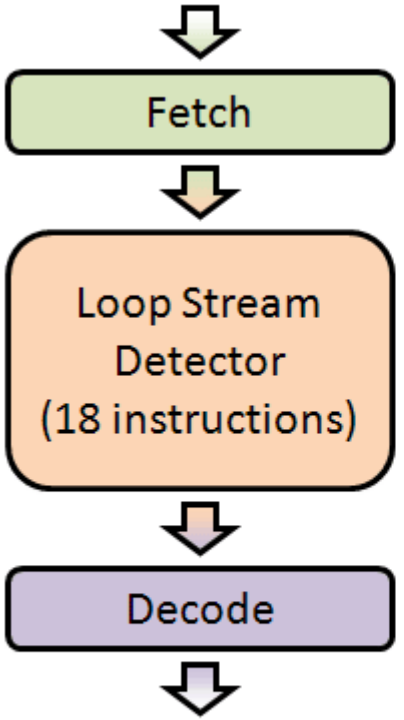
- AMD K10-based quad-core CPU's (Phenom 9 series)
- Intel Nehalem

Macro-fusion: melhorias

- Fusão de novas instruções de branch que não eram fundidas na arquitetura Core
- Macro-fusion em modo 64 bits
  - Na arquitetura Core, somente em modo 32 bits

Loop Stream Detector: Melhorias

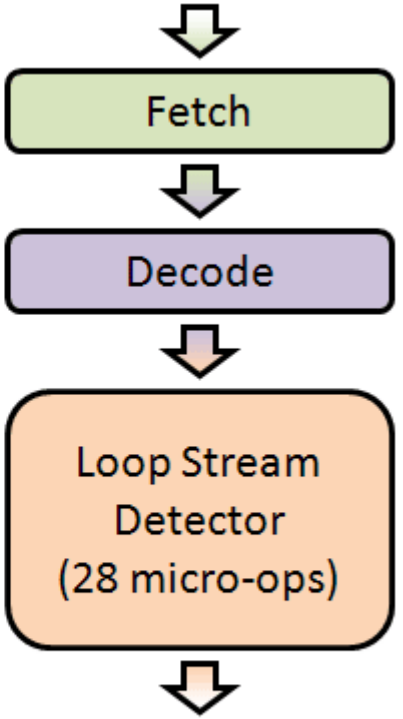
- Na arquitetura Core foi introduzido o Loop Stream Detector (LSD)
  - Um cache de 18 instruções entre as unidades de fetch e decode da CPU
  - Quando a CPU está rodando um Loop pequeno, ela não precisa buscar as instruções novamente no Cache L1, elas já estão próximas dos decodificadores de instruções no LSD
  - Além disso a CPU desliga as unidades de fetch e branch prediction enquanto está rodando um loop detectado, para economizar energia



LSD na arquitetura Intel Core

Loop Stream Detector: Melhorias

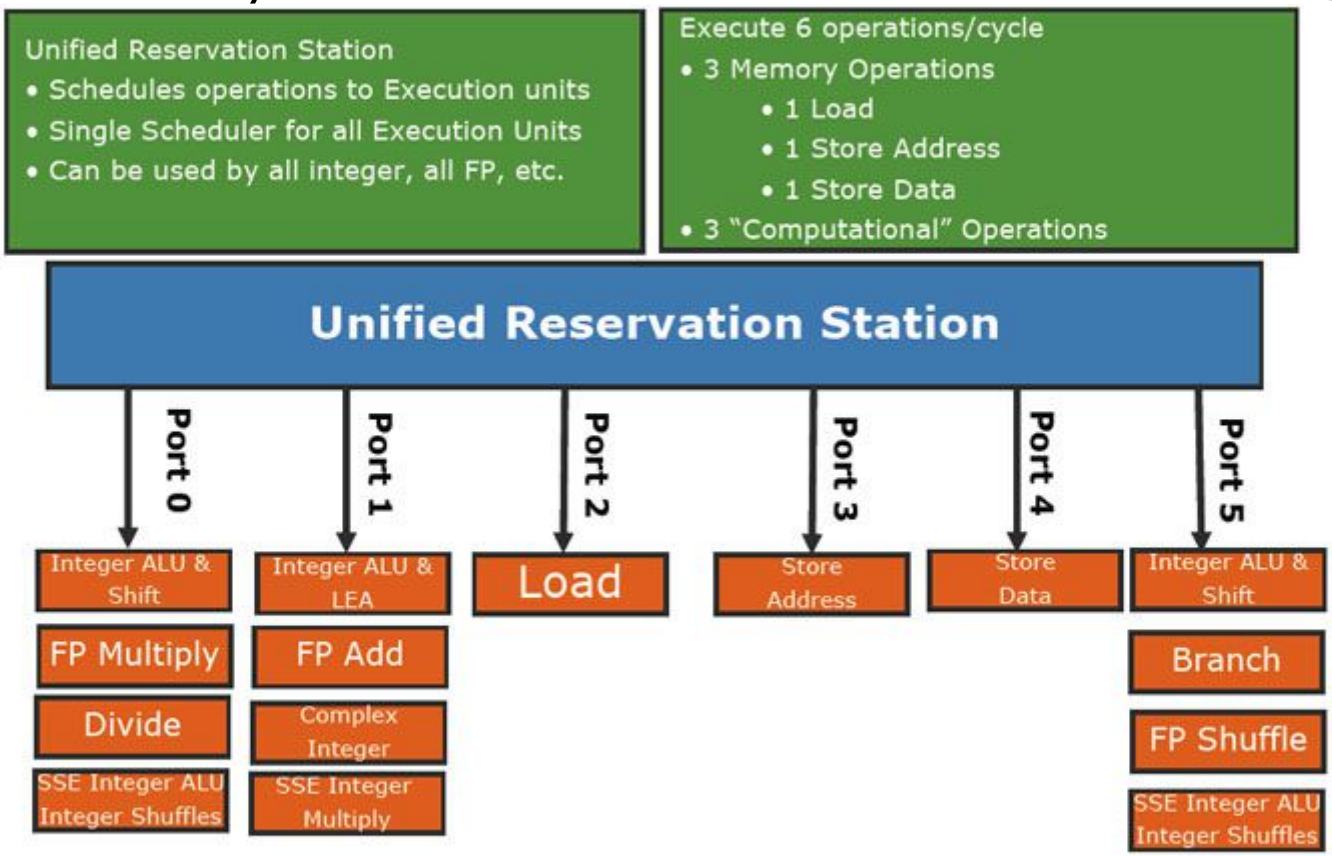
- Na arquitetura Nehalem
  - O LSD foi movido para depois das unidades de “Decode”
  - Com isso é feito “cache” de micro-ops (até 28 micro-ops)
  - Melhora de performance: não é preciso mais decodificar as instruções no loop
  - Economia de energia: a CPU também desliga as unidades de “Decode” enquanto está rodando um Loop detectado (além das unidades de branch prediction e fetch)



LSD na arquitetura Nehalem

Unidades de Execução: Melhorias

- Foi adicionada mais uma porta de execução: agora são 6, com 12 unidades de execução



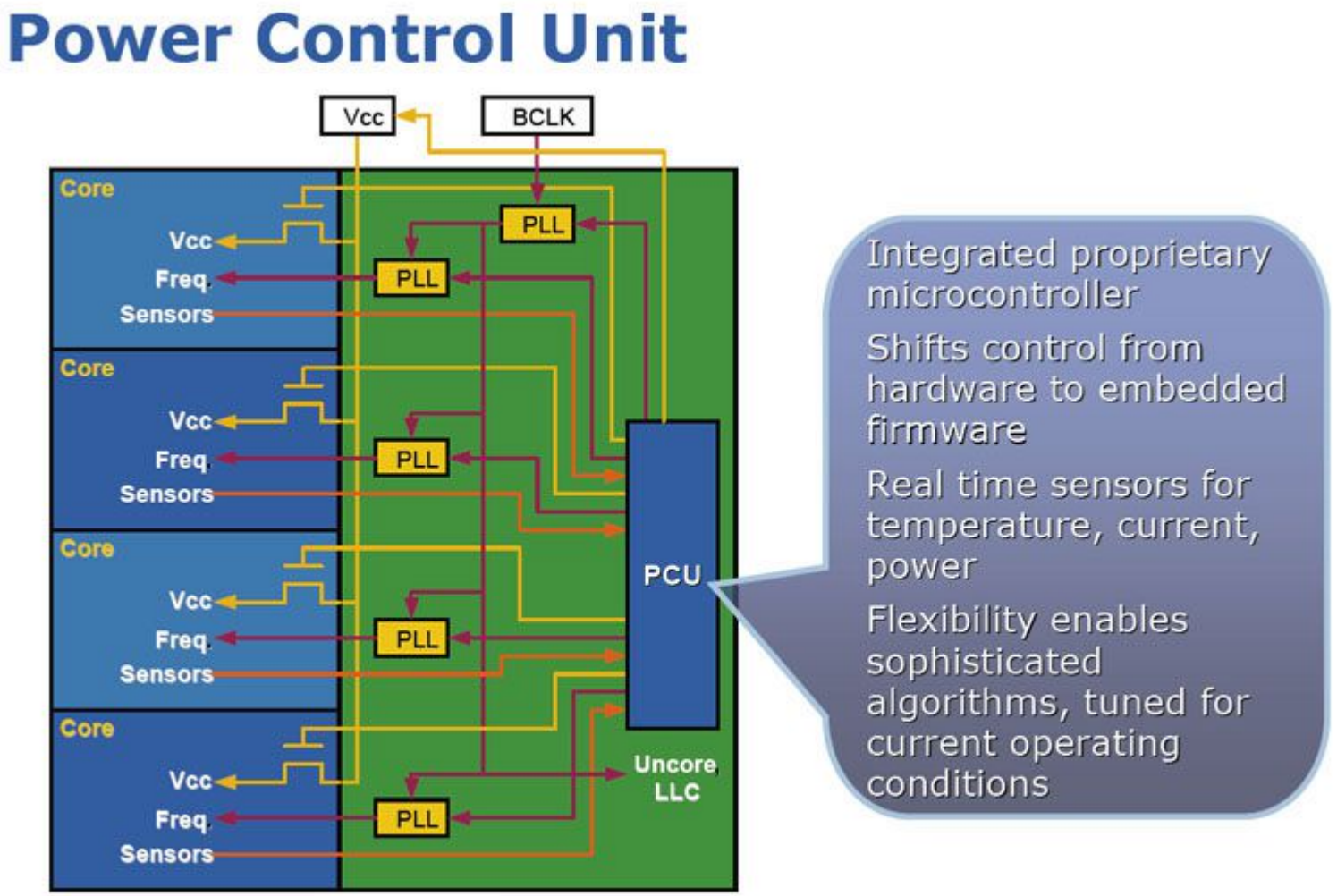
Buffers extras

- Um segundo Translation Look-aside Buffer (TLB) com 512 entradas
- Um segundo Branch target Buffer

Lembrete 1:
TLB is a table used for the conversion between physical addresses and virtual addresses by the virtual memory circuit. Virtual memory is a technique where the CPU simulates more RAM memory on a file on the hard drive (called swap file) to allow the computer to continue operating even when there is not enough RAM available (the CPU gets what is on the RAM memory, stores inside this swap file and then frees memory for using).

Lembrete 2:
Branch prediction is a circuit that tries to guess the next steps of a program in advance, loading to inside the CPU the instructions it thinks the CPU will try to load next. If it hits it right, the CPU won't waste time loading these instructions from memory, as they will be already inside the CPU. Increasing the size (or adding a second one, in the case of Nehalem-based CPUs) of the BTB allows this circuit to load even more instructions in advance, improving the CPU performance.

Melhorias no gerenciamento de Energia



Cada Core pode rodar em um determinado momento com tensões e frequências diferentes. Sensores monitoram de forma permanente o quanto cada Core está dissipando (potência), possibilitando a implementação do “Turbo Mode”

Turbo mode: “Auto overclock”

- The CPU is constantly monitoring its temperature and power consumption. The CPU will overclock the active cores until the CPU reaches its maximum allowed TDP, based on the cooling system you are using. This is configurable on the motherboard setup. For example, if you say your CPU cooler is able to dissipate 130 W, the CPU will increase (or reduce) its clock so the power currently dissipated by the CPU matches the amount of power the CPU cooler can dissipate. So if you, for example, replace the CPU stock cooler with a better cooler, you will have to enter the motherboard setup to configure the new cooler TDP (i.e., the maximum amount of thermal power it can dissipate) in order to make Turbo Mode to increase the CPU clock even more.
- Notice that the CPU doesn’t have to necessary shut down unused cores to enable Turbo Mode. But since this dynamic overclocking technique is based on how much power you can still dissipate using your current CPU cooler, shutting down unused cores will reduce the CPU consumption and power dissipation and thus will allow a higher overclocking.
- The new Turbo Mode is an extension to the SpeedStep technology, so it is viewed by the system as a SpeedStep feature.
- It only works for the CPU cores, so the memory controller and the memory cache are not affected by this technology.
- Apparently Turbo Mode will only be available on “Extreme Edition” models.

Hyperthreading

- Funcionalidade introduzida na arquitetura P7 (Pentium 4), pela primeira vez disponível na arquitetura P6 com a Nehalem
- Permite que um core da CPU seja reconhecido pelo software como 2 cores
- Internamente a CPU possui algumas unidades duplicadas (por exemplo, conjunto de registradores X86) e outras unidades são compartilhadas entre os “2 cores virtuais”
  - Exemplo: unidades de execução “não sabem” se estão executando micro-ops de um ou outro “processador”
- O front-end (instruction fetch, decode...) alterna entre os 2 processadores lógicos para garantir fairness.

Otimização para acessos não alinhados na memória

Fast Unaligned Cache Accesses

- Two flavors of 16-byte SSE loads/stores exist
  - Aligned (MOVAPS/D, MOVDQA) -- Must be aligned on a 16-byte boundary
  - Unaligned (MOVUPS/D, MOVDQU) -- No alignment requirement
- Prior to Intel® Core™ microarchitecture (Nehalem)
  - Optimized for Aligned instructions
  - Unaligned instructions slower, lower throughput -- Even for aligned accesses!
    - Required multiple uops (not energy efficient)
  - Compilers would largely avoid unaligned load
    - 2-instruction sequence (MOVSD+MOVHPD) was faster
- Intel Core microarchitecture (Nehalem) optimizes Unaligned instructions
  - Same speed/throughput as Aligned instructions on aligned accesses
  - Optimizations for making accesses that cross 64-byte boundaries fast
    - Lower latency/higher throughput than Core 2
  - Aligned instructions remain fast
- No reason to use aligned instructions on Intel Core microarchitecture (Nehalem)!
- Benefits:
  - Compiler can now use unaligned instructions without fear
  - ***Higher performance*** on key media algorithms
  - More ***energy efficient*** than prior implementations

Nehalem: Summary of New Features

- Based on Intel Core microarchitecture.
- Two to eight cores.
- Integrated DDR3 triple-channel memory controller.
- Individual 256 KB L2 memory caches for each core.
- 8 MB L3 memory cache.
- New SSE 4.2 instruction set (seven new instructions).
- Hyper-Threading technology.
- Turbo mode (auto overclocking).
- Enhancements to the microarchitecture (support for macro-fusion under 64-bit mode, improved Loop Stream Detector, six dispatch ports, etc).

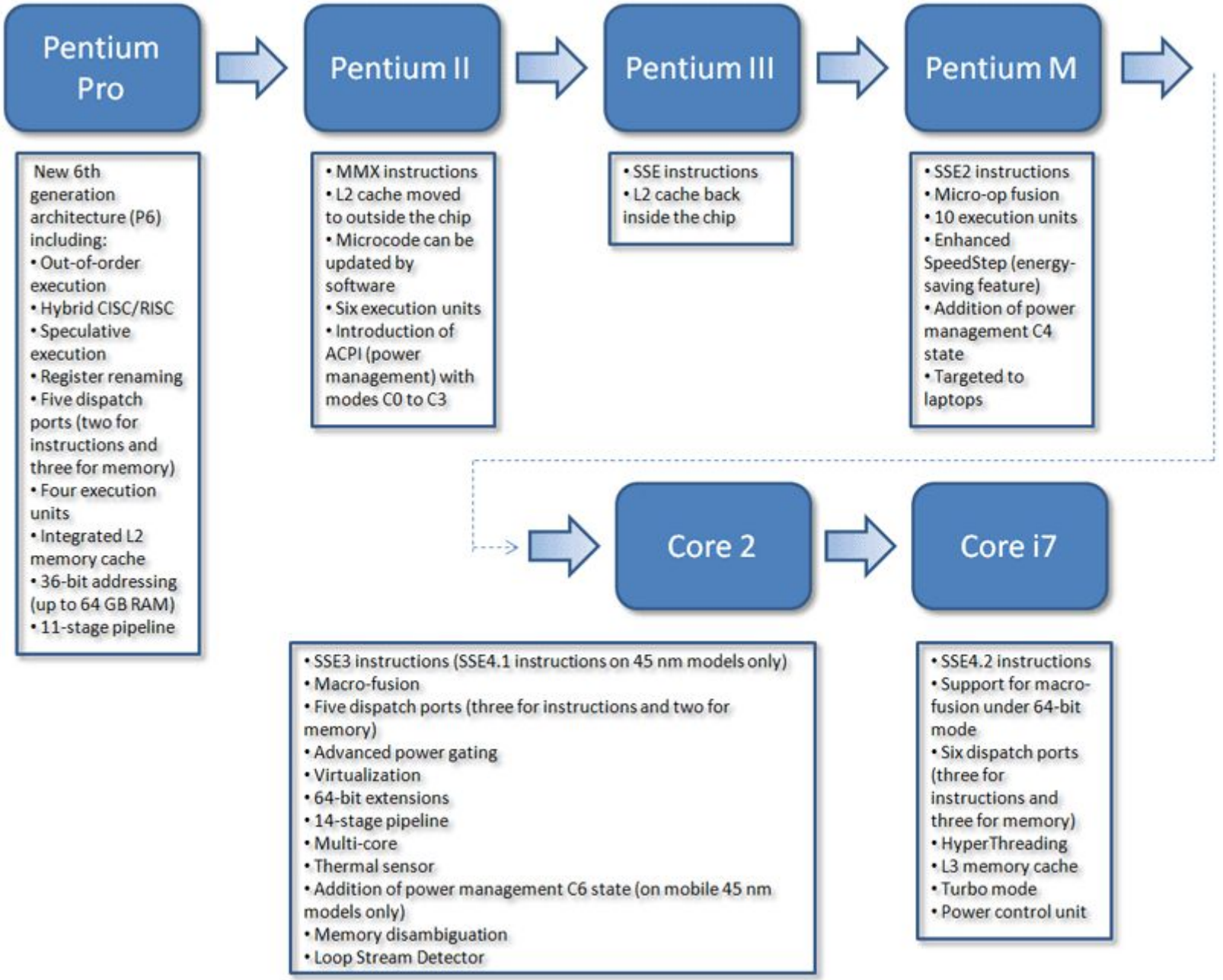
Nehalem: Summary of New Features

- Enhancements on the prediction unit, with the addition of a second Branch Target Buffer (BTB).
- A second 512-entry Translation Look-aside Buffer (TLB).
- Optimized for unaligned SSE instructions.
- Improved virtualization performance (60% improvement on round-trip virtualization latency compared to 65-nm Core 2 CPUs and 20% improvement compared to 45-nm Core 2 CPUs, according to Intel).
- New QuickPath Interconnect external bus.
- New power control unit.
- 45 nm manufacturing technology at launch, with future models at 32 nm (CPUs codenamed “Westmere”).
- New socket with 1366 pins.

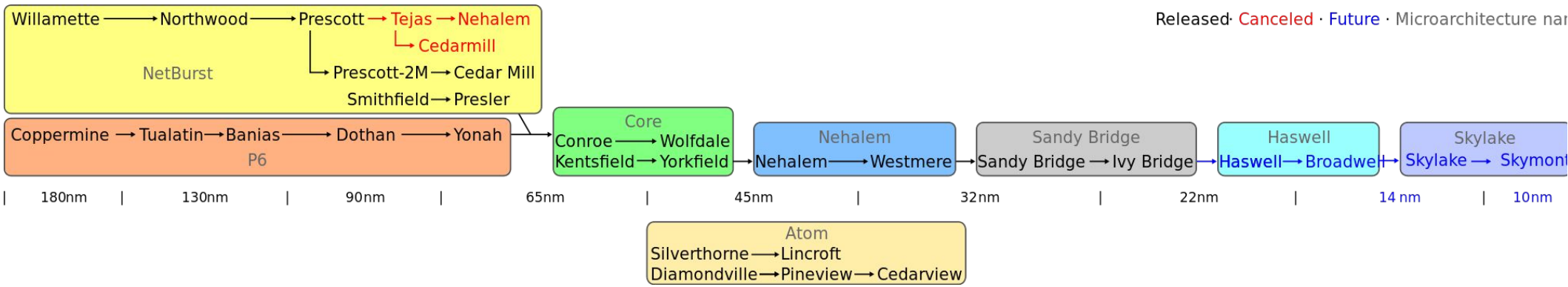
Features from 2nd generation Core architecture

- 1st generation Intel Core CPUs are based on 65-nm fabrication process
- 2nd generation Intel Core CPUs are based on 45-nm fabrication process
- 45-nm Intel Core CPUs had the following features added, they are all present in Nehalem CPUs:
  - SSE4.1 instruction set (47 new SSE instructions).
  - Deep Power Down Technology (only on mobile CPUs, also known as C6 state).
  - Enhanced Intel Dynamic Acceleration Technology (only on mobile CPUs).
  - Fast Radix-16 Divider (FPU enhancement).
  - Super Shuffle engine (FPU enhancement).
  - Enhanced Virtualization Technology (between 25% and 75% performance improvement on virtual machine transition time).

Genealogia: Features



Genealogia: Processo de fabricação



Fonte: Wikipedia http://en.wikipedia.org/wiki/Sandy_bridge

Fontes:

- O conteúdo dos slides foi baseado sobretudo nas informações disponíveis em:
  - Intel® 64 and IA-32 Architectures Optimization Reference Manual, Chapter 2: INTEL® 64 AND IA-32 PROCESSOR ARCHITECTURES
  - Artigos em www.hardwaresecrets.com
    - Inside Pentium M Architecture: http://www.hardwaresecrets.com/article/Inside-Pentium-M-Architecture/270
    - Inside Intel Core Microarchitecture: http://www.hardwaresecrets.com/article/Inside-Intel-Core-Microarchitecture/313
    - Inside Intel Nehalem Microarchitecture: http://www.hardwaresecrets.com/article/Inside-Intel-Nehalem-Microarchitecture/535
    - Inside Intel Sandy Bridge Microarchitecture: http://www.hardwaresecrets.com/article/Inside-the-Intel-Sandy-Bridge-Microarchitecture/1161