# Multi-core Architecture and Programming

Yang Quansheng(杨全胜)
http://www.njyangqs.com/

School of Computer Science & Engineering
Southeast University

# Introduce of Multi-Core

- Content
  - Why we need Multi-core?
  - What is Multi-core?
  - Multi-core Architecture
  - Intel® Dual-core introduction
  - Challenge

# Why we need Multi-core

- When computer was born, a program is a **sequence of instructions.**
- In the 1960s, the technology called **concurrency** let Multi-users could access a single mainframe computer simultaneously.
- In the early days of personal computers, the OS are called **single-user operating system** and only one program would run at a time.

# Why we need Multi-core

- Recently
  - More and more task want to execute simultaneously.
  - Each task (the applications) become more and more complex
  - The internet become more and more pervasive
  - User's high performance processor expectation

# Why we need Multi-core

- Parallel computing is the best way for the high performance expectation
  - Computation requirements are ever increasing
    - visualization, distributed databases, simulations, scientific prediction (earthquake), large scale network activities, etc.
  - Silicon based (sequential) architectures reaching physical limits in processing limits as they are constrained by:
    - the speed of light
    - Thermodynamics

# Why we need Multi-core

- Parallel computing is the best way for the high performance expectation
  - Hardware improvements like pipelining, superscalar are not scaling well and require sophisticated compiler technology to exploit performance out of them.
  - Techniques such as vector processing works well for certain kind of problems.

# Why we need Multi-core

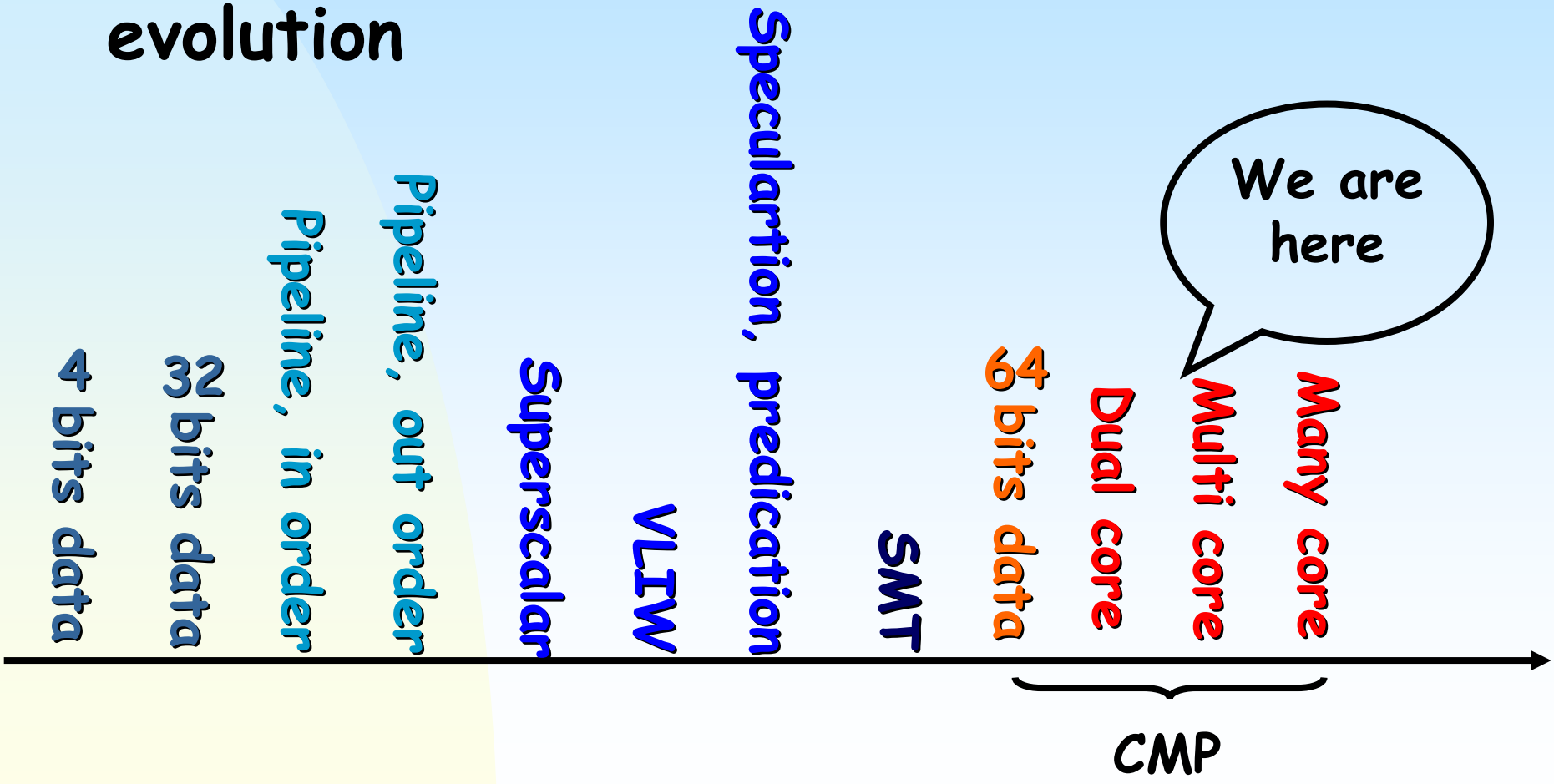
- Parallel computing is the best way for the high performance expectation
  - Significant development in networking technology is paving a way for network-based cost-effective parallel computing.
    - InifiBand
      - Today: 10,20Gb/s node-to-node
      - 30,60Gb/s switch-to-switch

# Why we need Multi-core

- Parallel computing is the best way for the high performance expectation
  - Parallel computing
    - one of the best ways to overcome the speed bottleneck of a single processor
    - good price/performance ratio of a small cluster-based parallel computer
    - Parallel processing technology is mature and is being exploited commercially.

# Why we need Multi-core

- A brief history of micro-architecture evolution



# Why we need Multi-core

- How to improve the performance of the processor
  - Delivered Performance = Frequency * Instructions Per Cycle (IPC)
  - Improving the main frequence is the traditional method, but is led huge heat.
  - Multi-core is the another way which can increase the IPC.

# Why we need Multi-core

- The benefits of Multi-core
  - Better responsiveness
  - Higher multithreaded throughput
  - Benefits of parallel computing in mainstream applications
  - enhance user experiences in multitasking environments

# Introduce of Multi-Core

- Content
  - Why we need Multi-core?
  - What is Multi-core?
  - Multi-core Architecture
  - Intel® Dual-core introduction
  - Challenge

# What is Multi-core

- a single processor package that contains two or more processor core
- Two or more processor in a single die
- Isomorphic Multi-core vs. isomerous Multi-core
- Share the Cache vs. interconnect on chip

# What is Multi-core

- Concurrency and Parallel computing
  - Concurrency: in software is a way to manage the sharing of resources used at the same time.
  - Parallel computing: involves the simultaneous use of more than one computer or processor to execute a program. Ideally, parallel processing makes a program run faster because there are more engines (CPUs) running it. Even single-core processor computers can perform parallel processing by connecting to other computers in a network.

# What is Multi-core

- Simultaneous multithreading (SMT)
  - Permits multiple independent threads to execute SIMULTANEOUSLY on the SAME core
  - Weaving together multiple "threads" on the same core
  - Example: if one thread is waiting for a floating point operation to complete, another thread can use the integer units

# What is Multi-core

- Parallelism
  - Instruction-level parallelism
    - Parallelism at the machine-instruction level
    - The processor can re-order, pipeline instructions, split them into micro-ops, do aggressive branch prediction, etc.
    - Instruction-level parallelism enabled rapid increases in processor speeds over the last 15 years

# What is Multi-core

- Parallelism
  - Thread-level parallelism (TLP)
    - This is parallelism on a more coarser scale
    - Server can serve each client in a separate thread (Web server, database server)
    - A computer game can do AI, graphics, and physics in three separate threads
    - Single-core superscalar processors cannot fully exploit TLP
    - Multi-core architectures are the next step in processor evolution: explicitly exploiting TLP

# What is Multi-core

- Multi-core processor is a special kind of a multiprocessor
  - All processors are on the same chip
  - Multi-core processors are MIMD: Different cores execute different threads (Multiple Instructions), operating on different parts of memory (Multiple Data).
  - Multi-core is a shared memory multiprocessor:

All cores share the same memory

# What is Multi-core

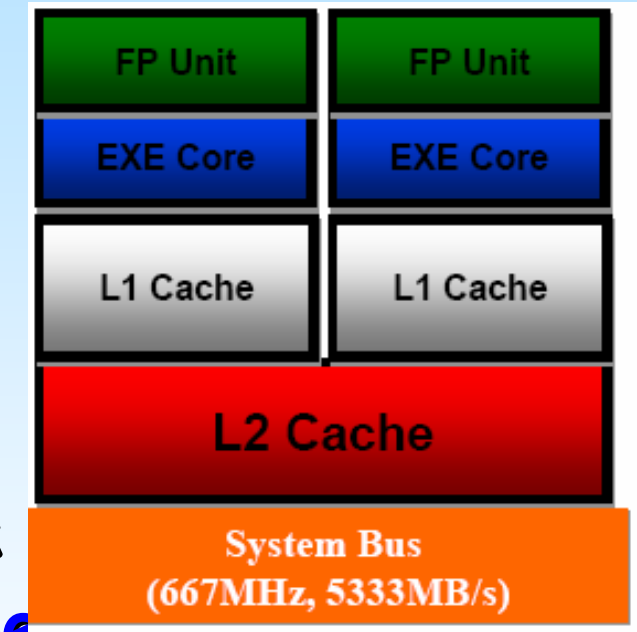
- What applications benefit from multi-core?
  - Database servers
  - Web servers (Web commerce)
  - Compilers
  - Multimedia applications
  - Scientific applications, CAD/CAM
  - In general, applications with *Thread-level parallelism* (as opposed to instruction-level parallelism)

# Introduce of Multi-Core

- Content
  - Why we need Multi-core?
  - What is Multi-core?
  - Multi-core Architecture
  - Intel® Dual-core introduction
  - Challenge

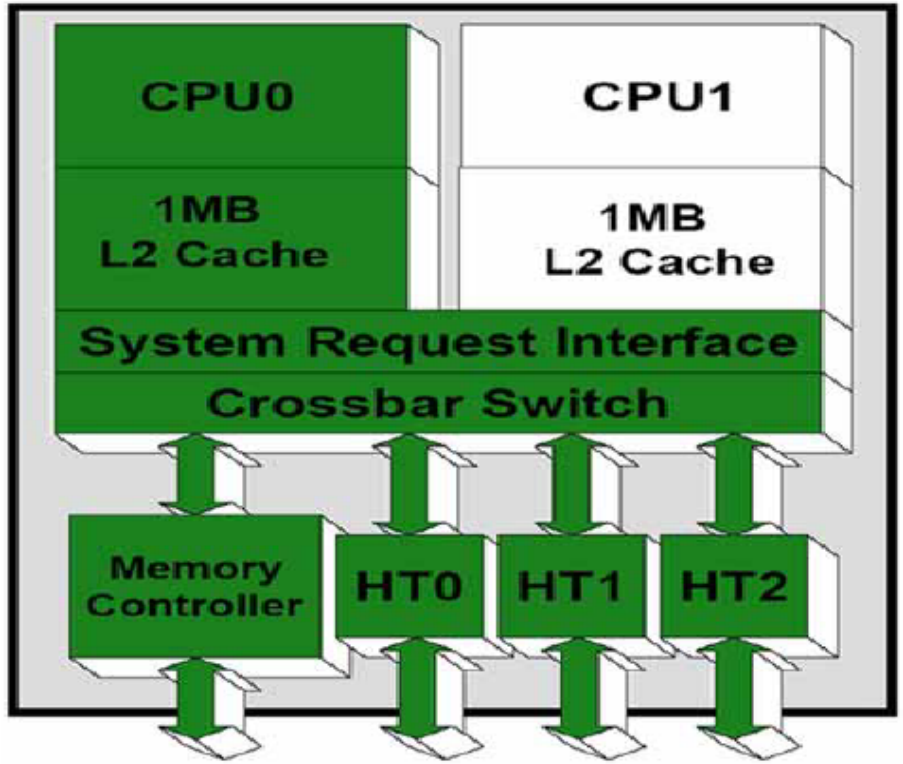
# Multi-core Architecture

- INTEL CORE DUO
  - Two physical cores in a package
  - Each with its own execution resources
  - Each with its own L1 cache
    - 32K instruction and 32K data
  - Both cores share the L2 cache
    - 2MB 8-way set associative; 64-byte line size
    - 10 clock cycles latency; Write Back update policy



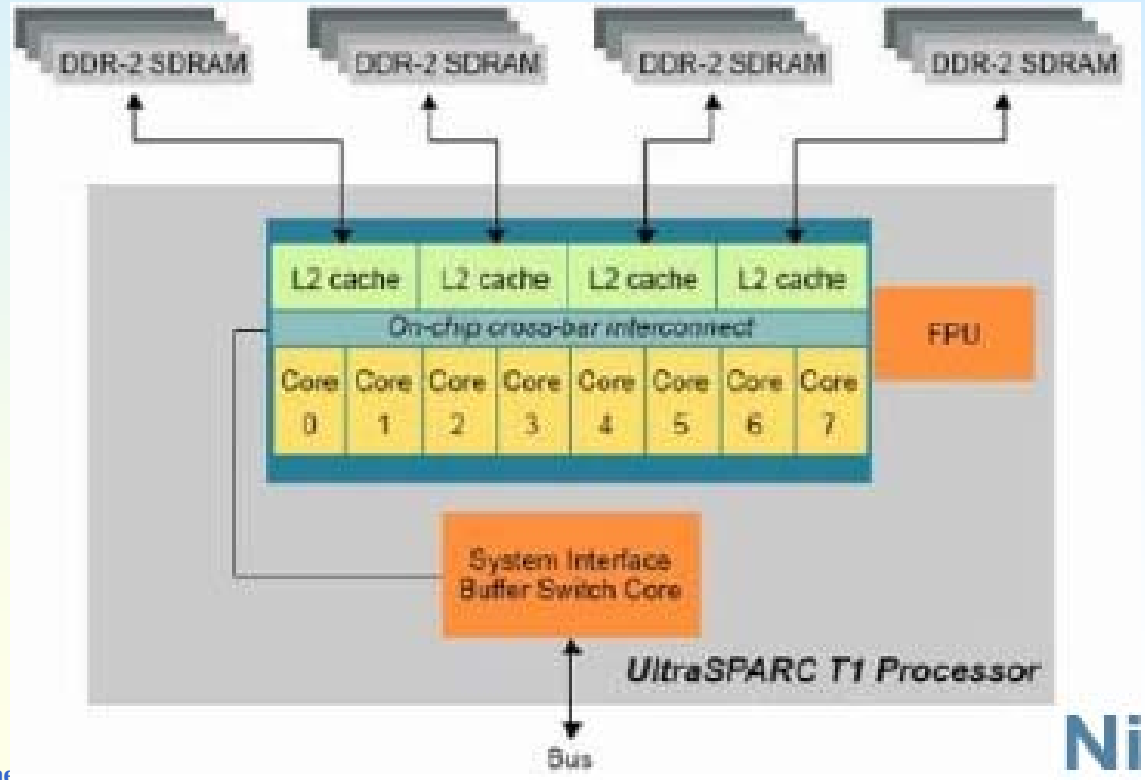
# Multi-core Architecture

- AMD Opteron
  - Separate 1 MbyteL2 caches
  - CPU0 and CPU1 communicate through the SRQ



# Multi-core Architecture

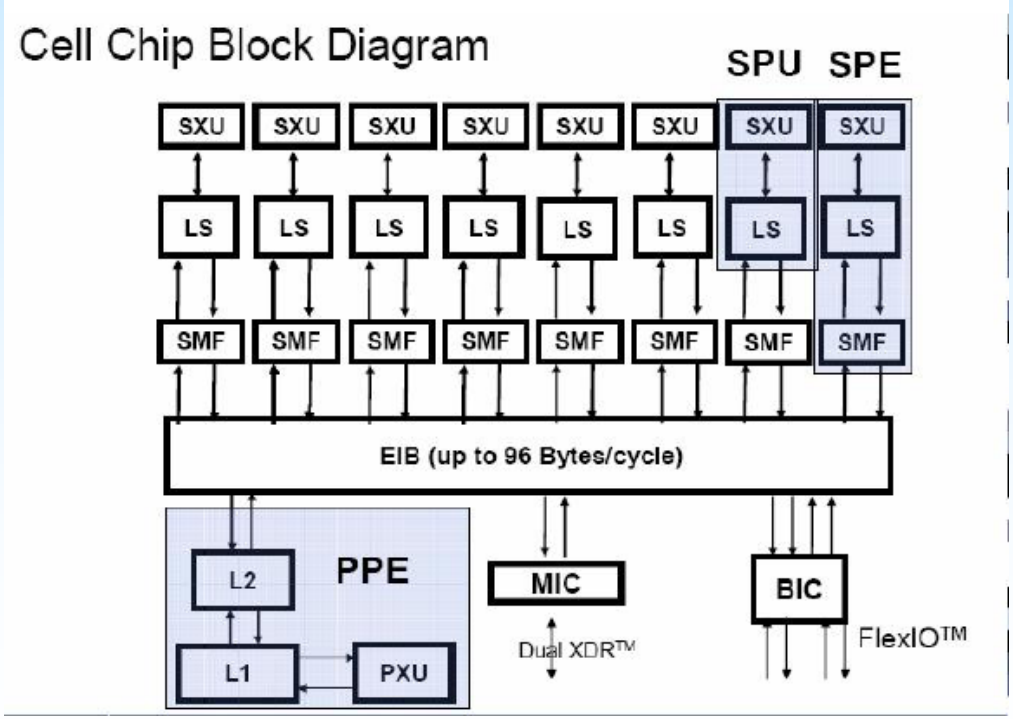
- Niagara from SUN
  - 8 CPU core
  - Each core can run 4 thread simultaneity




In Niagara2, Each core can run 8 thread simultaneity

# Multi-core Architecture

- Cell from IBM, Toshiba and Sony
  - 1 PPE (Power Processor Element)
  - 8 SPE (Synergistic Processor Element)



# Introduce of Multi-Core

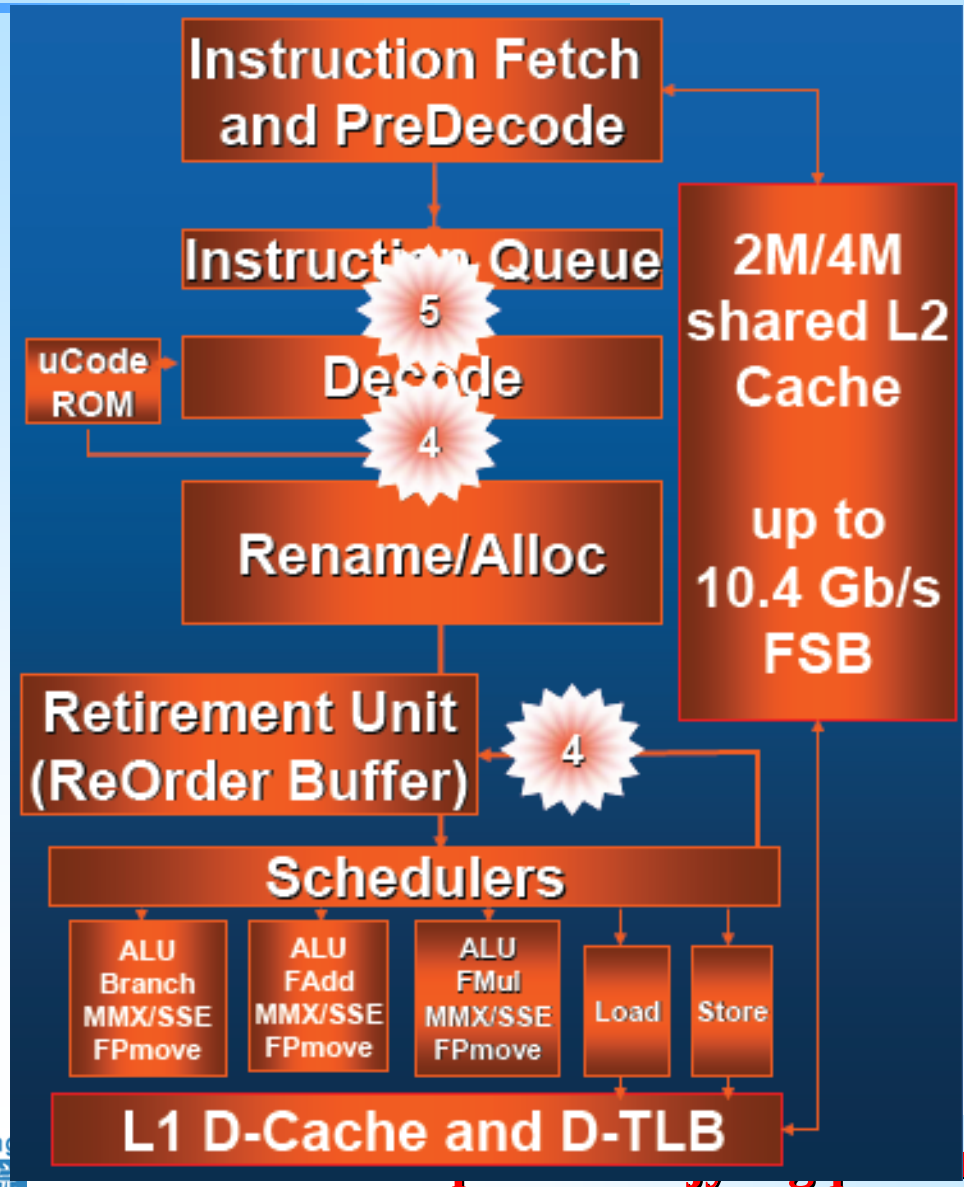
- Content
  - Why we need Multi-core?
  - What is Multi-core?
  - Multi-core Architecture
  - Intel® Dual-core introduction
  - Challenge

# Intel® Dual-core introduction

- Key Features
  - Two physical cores in a package
  - Each core with its own execution resources
  - Each core with its own L1 cache
    - 32K instruction and 32K data
  - Both cores share the L2 cache
    - 2MB
    - 10 clock cycles latency
    - Write Back update policy
    - What is write back cache policy ?

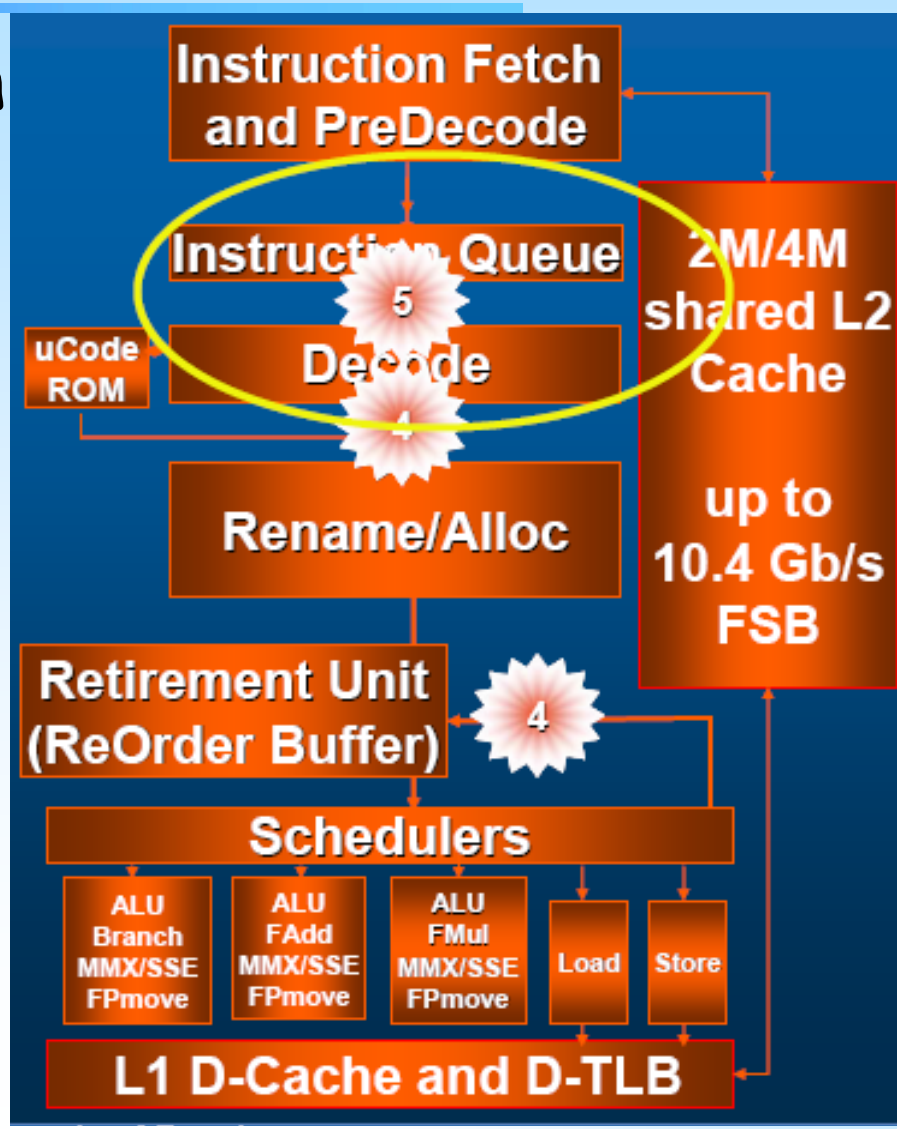
# Intel® Dual-core introduction

- New technique
  - Wide Dynamic Execution
  - Advanced Digital Media Boost
  - Smart Memory Access
  - Advanced Smart Cache
  - Intelligent Power Capability



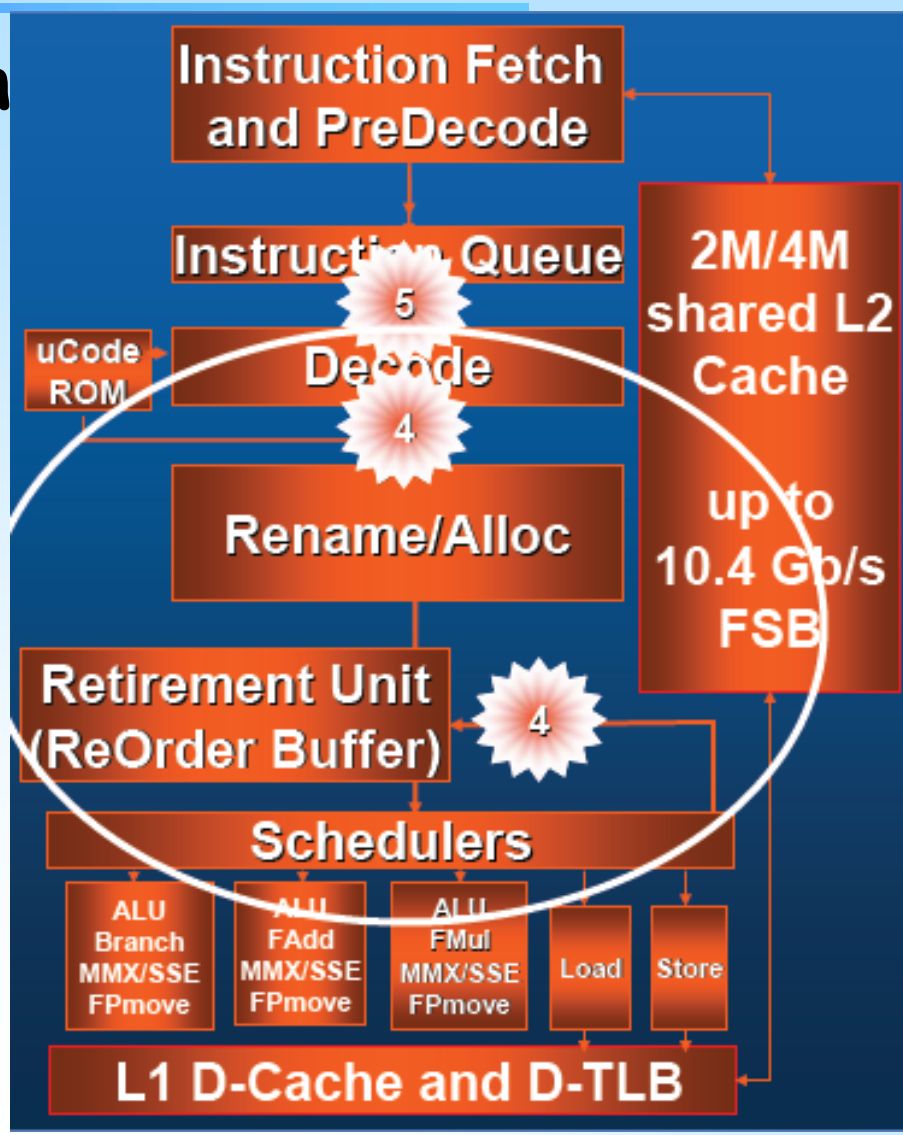
# Intel® Dual-core introduction

- Wide Dynamic Execution
  - *Start with Instruction Fetch*
  - four(+) instructions / cycle
  - >33% increase over other x86 processors
  - Instructions converted to micro-ops (uops)
  - ~1 uopper x86 instruction



# Intel® Dual-core introduction

- Wide Dynamic Execution
  - 4 wide rename
  - 4 wide micro-op execution
  - 4 wide retire
  - Deeper out of order storage
  - 32 discontiguous micro-ops considered for dispatch per cycle



# Intel® Dual-core introduction

- Wide Dynamic Execution

Delivered Performance =
Frequency * Instructions Per Cycle (IPC)

Power = $C_{dynamic}$ * V * V * Frequency

Fewer uops per instruction allows IPC to be increased while lowering C dynamic (less bits and less toggling)

# Intel® Dual-core introduction

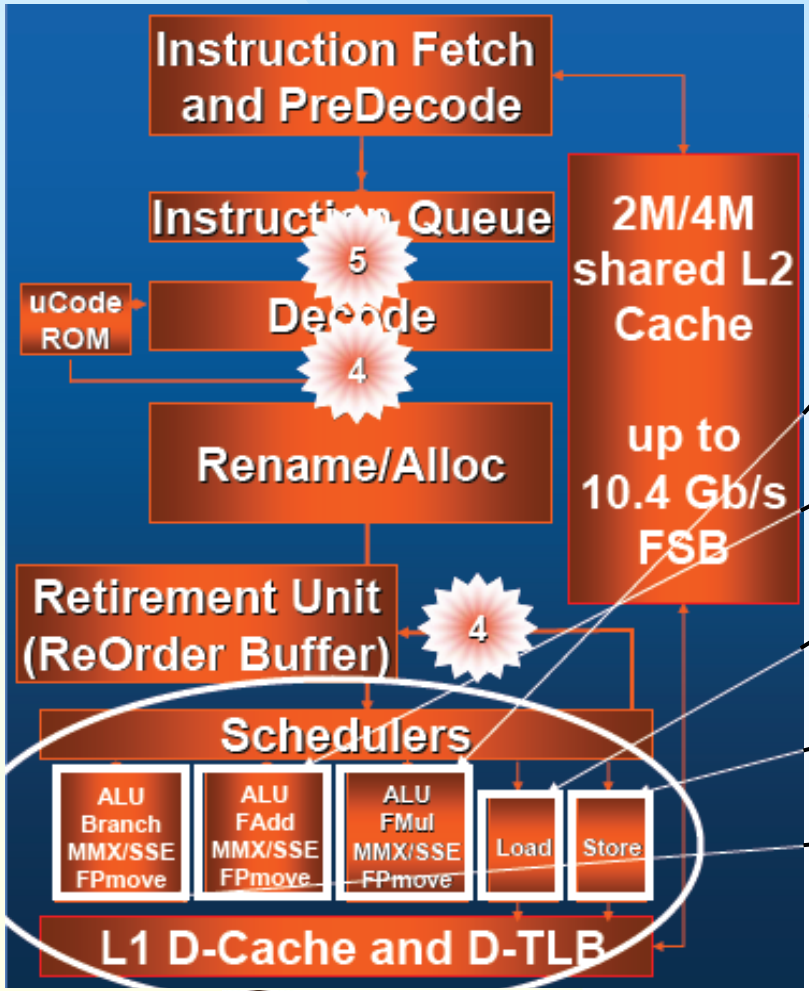
- Techniques for Micro-op Reduction
  - ESP Tracker (Extended Stack Pointer)
    - Execute Stack Pointer updates in dedicated hardware
    - *Intel Intel® Core™ microarchitecture increases Band Width 33%**
  - Micro-Op Micro-Fusion
    - Single Uop representation of "multi-uop" instruction
    - *Intel Intel® Core™ microarchitecture increase # instructions**
  - *Macro-Fusion*
    - New technique in Intel® Core™ microarchitecture

# Intel® Dual-core introduction

- Macro-Fusion
  - Represent common x86 instruction pairs in single micro-op
    - CMP or TEST + Conditional Branch (Jcc)
  - Enhanced Arithmetic Logic Unit (ALU) for macro-fusion
    - Single dispatch- efficiency
    - Single cycle execution - performance

# Intel® Dual-core introduction

- Intel® Advanced Digit Media Boost




2xCompute Throughput/Clock

128-bit packed Multiply

128-bit packed Add

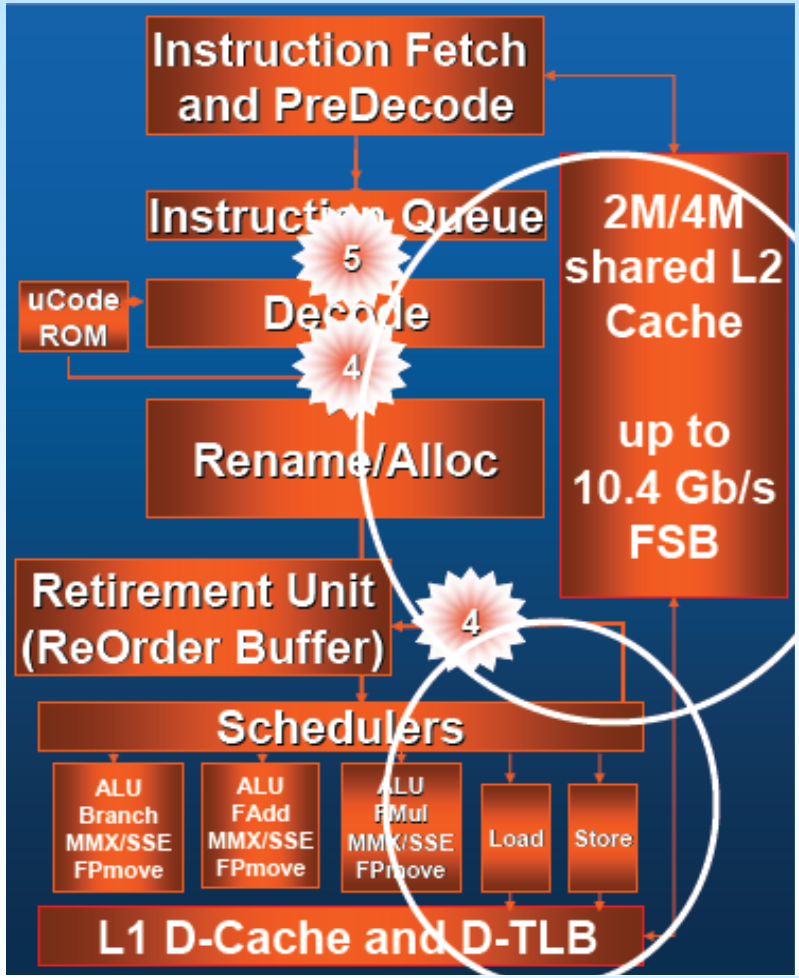
128-bit packed Load

128-bit packed Store

How about a CMPJCC

# Intel® Dual-core introduction

- Intel® Smart Memory Access
  - Memory Disambiguation
  - Improved Prefetchers
  - The Goal
    - WHEN - Ensure data can be used as early as possible
    - WHERE - Ensure user of data has it as close as possible



# Intel® Dual-core introduction

- Intel® Smart Memory Access
  - Memory Disambiguation - Solving the Problem of WHEN
    - Loads can decouple from Stores
    - Loads that are predicted NOT to forward from preceding store are allowed to schedule as early as possibleas
      - increasing the performance of OOO memory pipelines
    - Disambiguated loads checked at retirement
      - Extension to existing coherency mechanism
      - Invisible to software and system

# Intel® Dual-core introduction

- Intel® Smart Memory Access
  - Improved Prefetchers - Solving the Problem of WHERE
    - Memory is too far away
    - Caches are closer when they have the data
    - Prefetchers detect applications data reference patterns
    - And bring the data closer to data consumer

# Intel® Dual-core introduction

- Advanced Smart Cache
  - L2 can adapt to each core's load
  - Fast data sharing
  - No replicated data
  - 2X Band Width to L1 caches

# Intel® Dual-core introduction

- Intelligent Power Capability
  - Ultra Fine Grained Power Control
    - Even during periods of high performance execution, many parts of the chip core can be shut off.
  - Split Busses
    - By splitting buses to deal with varying data widths,we can gain the performance benefit of bus width while maintaining C dynamic closer to thinner buses.
  - Platformization of Power Management Architecture
    - PSI-2 Power Status Indicator (Mobile)
    - DTS Digital Thermal Sensors
    - PECI Platform Environment Control Interface

# Introduce of Multi-Core

- Content
  - Why we need Multi-core?
  - What is Multi-core?
  - Multi-core Architecture
  - Intel® Dual-core introduction
  - Challenge

# Challenge

- Challenge to the programmer
  - From a single-core programmer to a Multi-core programmer
  - Increase the program efficiency by using the Multi-core processor
- Challenge to the parallel programming
  - Synchronization
  - Communication
  - Load Balancing
  - Scalability