Intel Core 2 Architecture

Implication for software developers

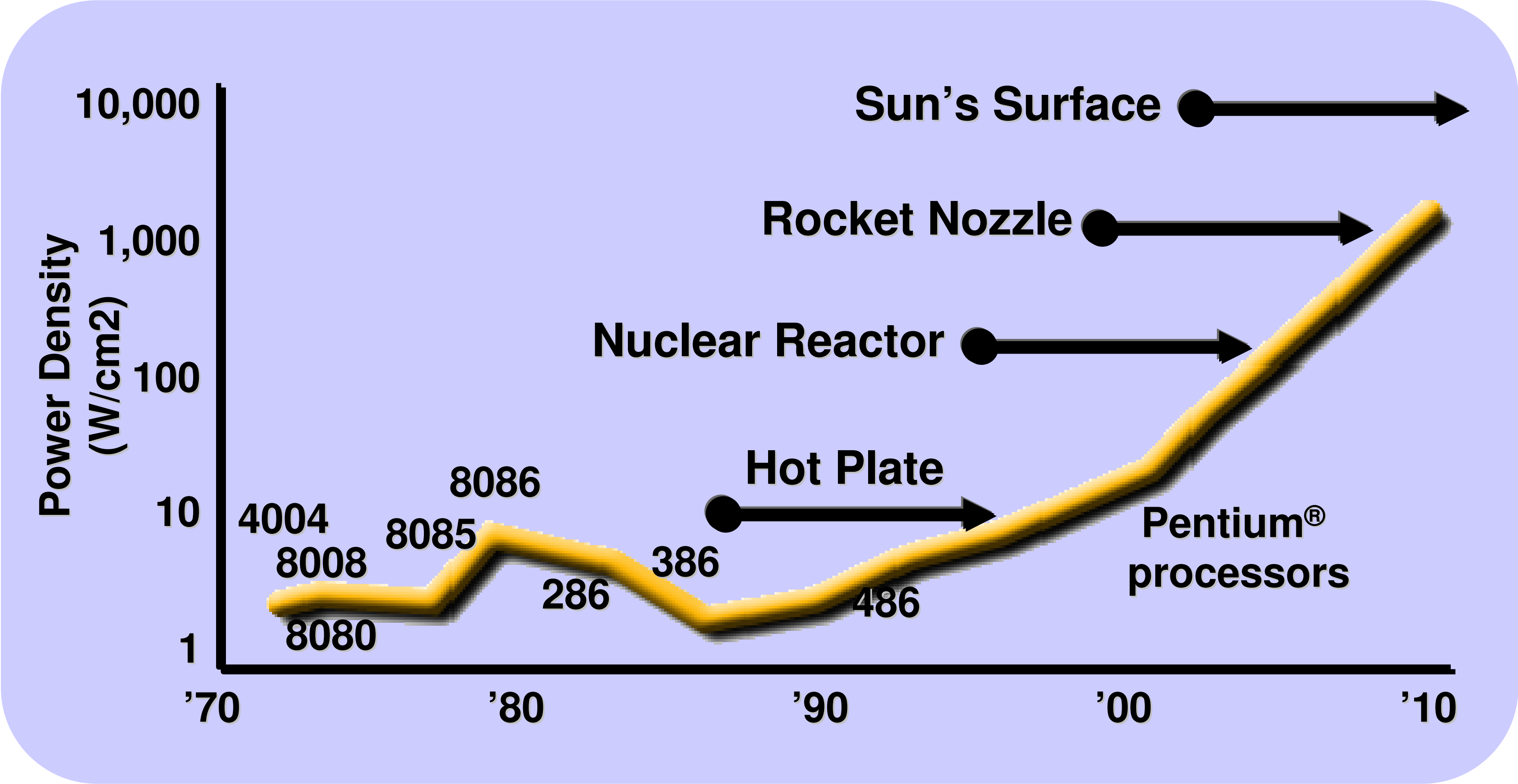
Stephen Blair-Chappell
Technical Consulting Engineer
Intel Compiler Labs

Agenda

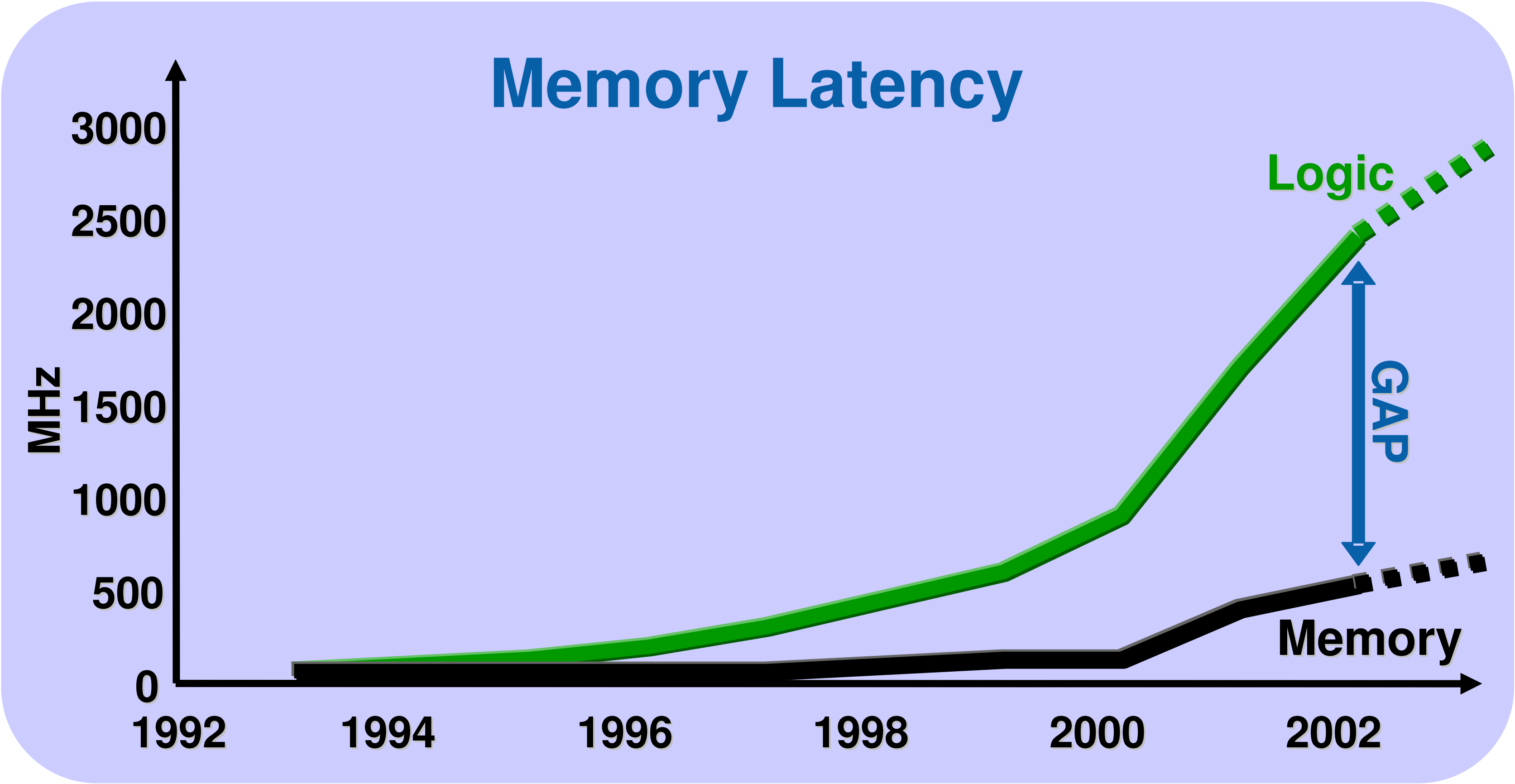
- Why Multicore
- Core 2 Microarchitecture
  - From high above
  - A closer look at the cores
  - In-depth pipeline walk-through
- Implications for Developers
  - Pipeline stalls
  - vectorization
  - Making use of dual core
- Summary / Q&A

Challenge 1: Power

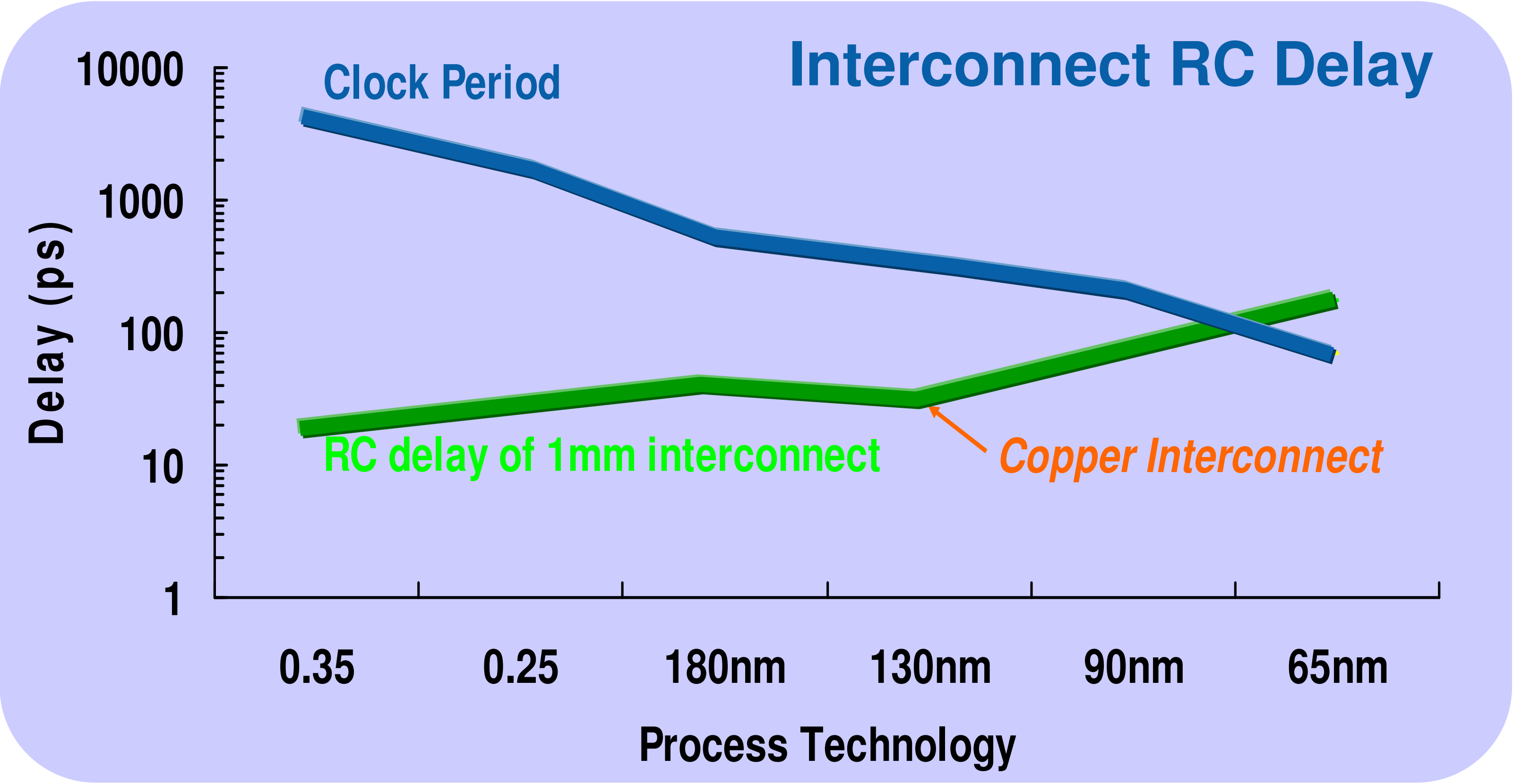
Power Density Race



Challenge 2: Memory Latency

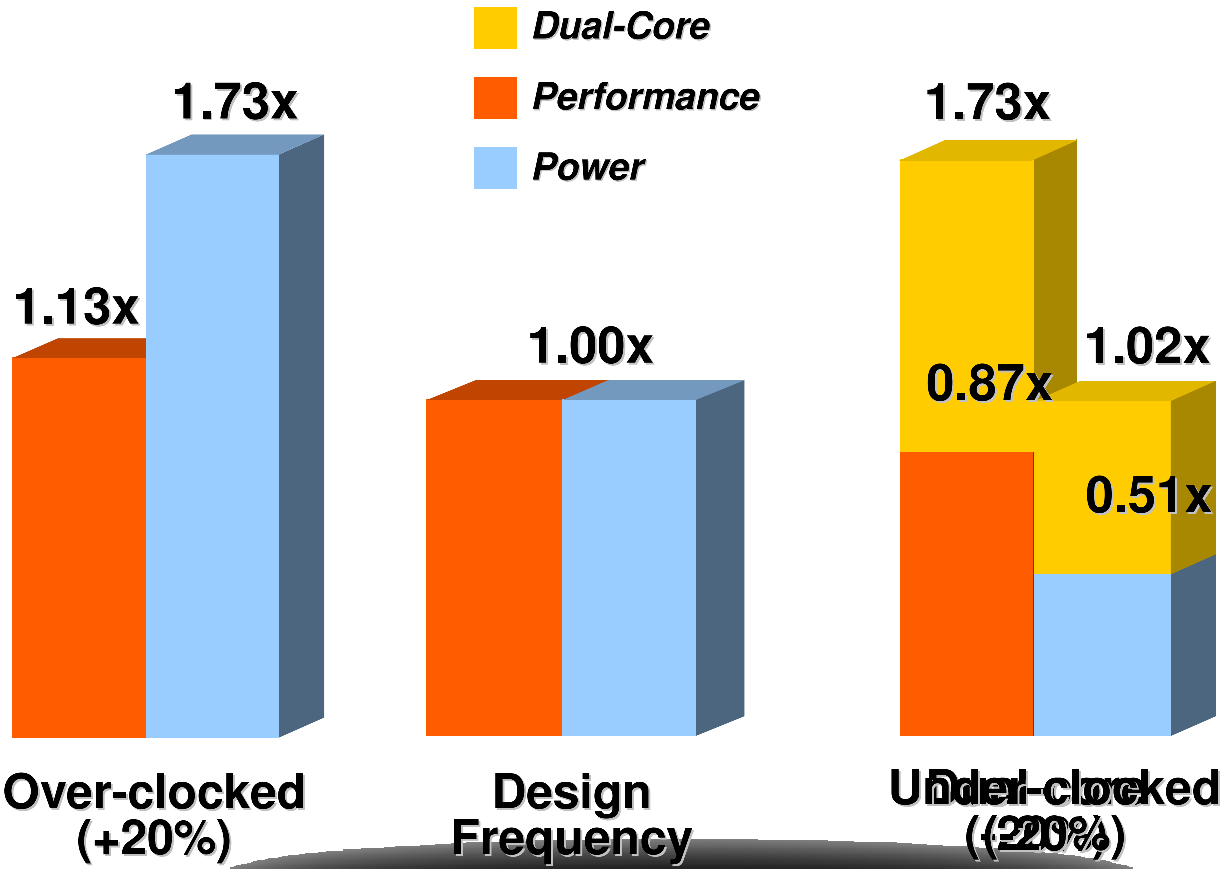

Memory Latency

Challenge 3: RC Delays



i486™: 16 clocks RAM access roundtrip
65 nm IC: 15 clocks to cross the chip

The Multi-Core Advantage



Relative single-core frequency and Vcc

Core 2 Duo – 1 Meter above

1.8 – 3.0 GHz

64Bit

291Million Transistors

65nm Process

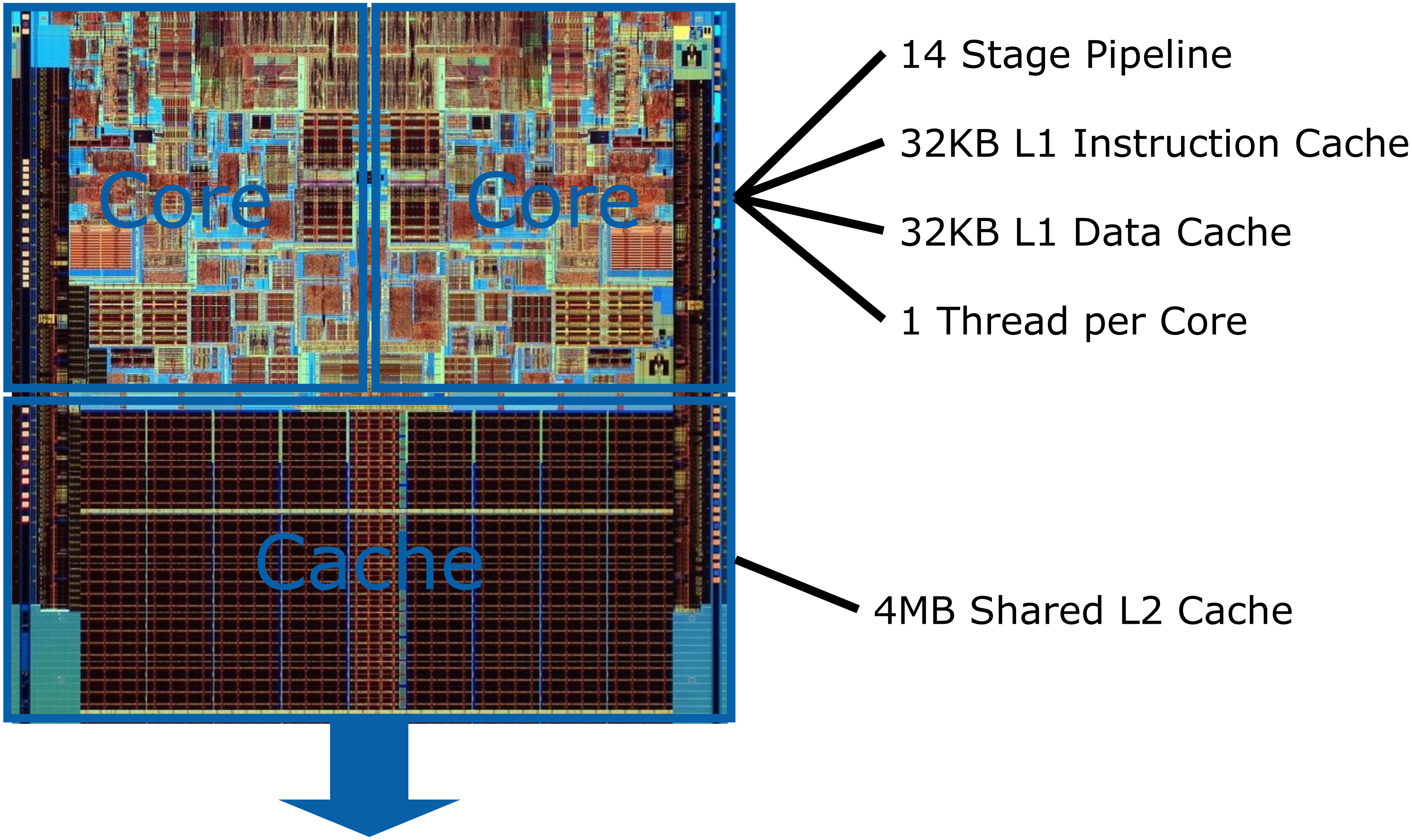
1066 – 1333 MHz FSB

10 – 65W TDP

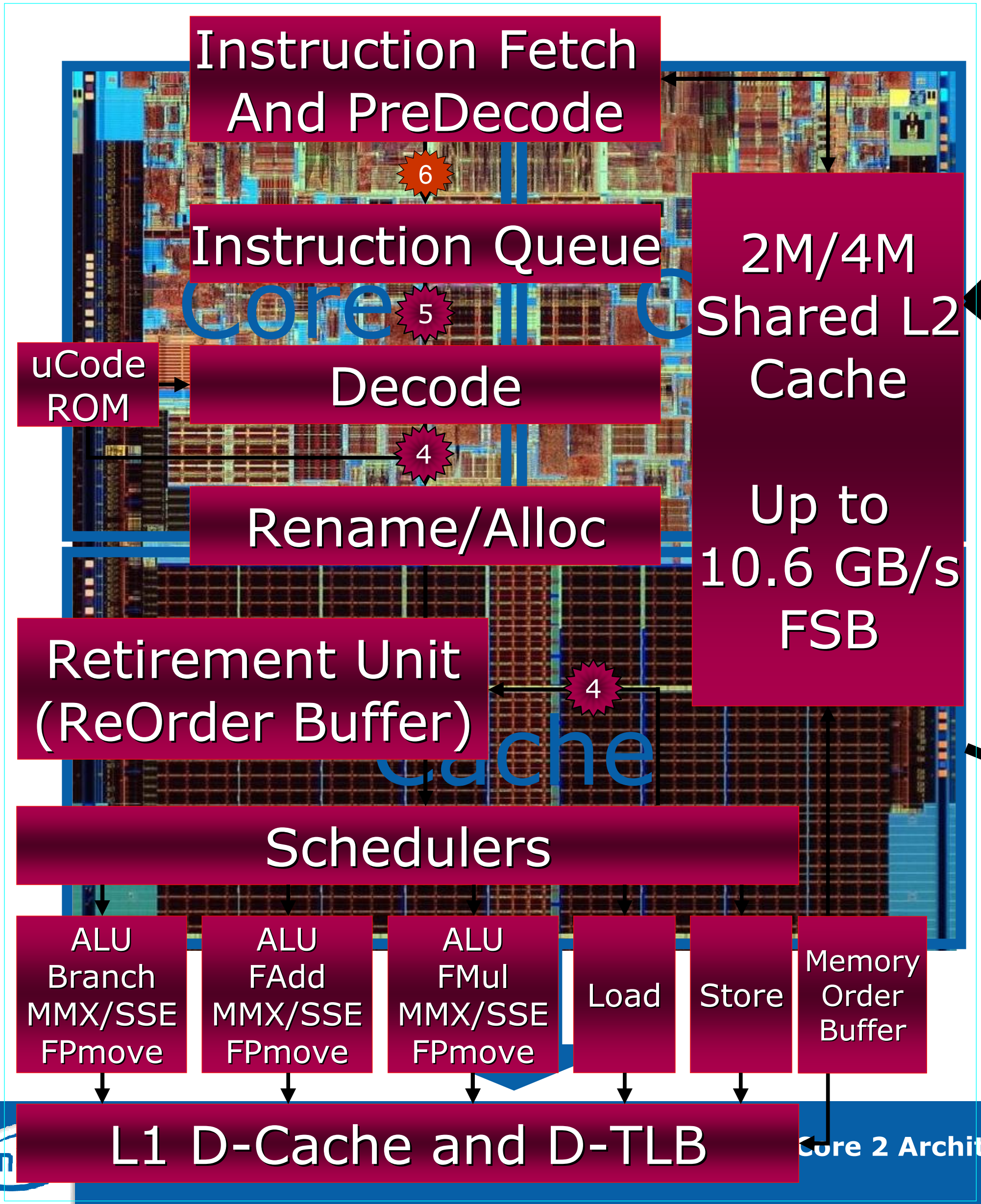
143 mm^2 Dye Size

Socket LGA 775

Core 2 – 1 cm above



Core 2 – In the Core



Instruction Fetch And PreDecode

6

Instruction Queue

5

uCode ROM

Decode

4

Rename/Alloc

Retirement Unit (ReOrder Buffer)

4

Schedulers

ALU Branch MMX/SSE FPmove

ALU FAdd MMX/SSE FPmove

ALU FMul MMX/SSE FPmove

Load

Store

Memory Order Buffer

L1 D-Cache and D-TLB

2M/4M Shared L2 Cache

Up to 10.6 GB/s FSB

14 Stage Pipeline

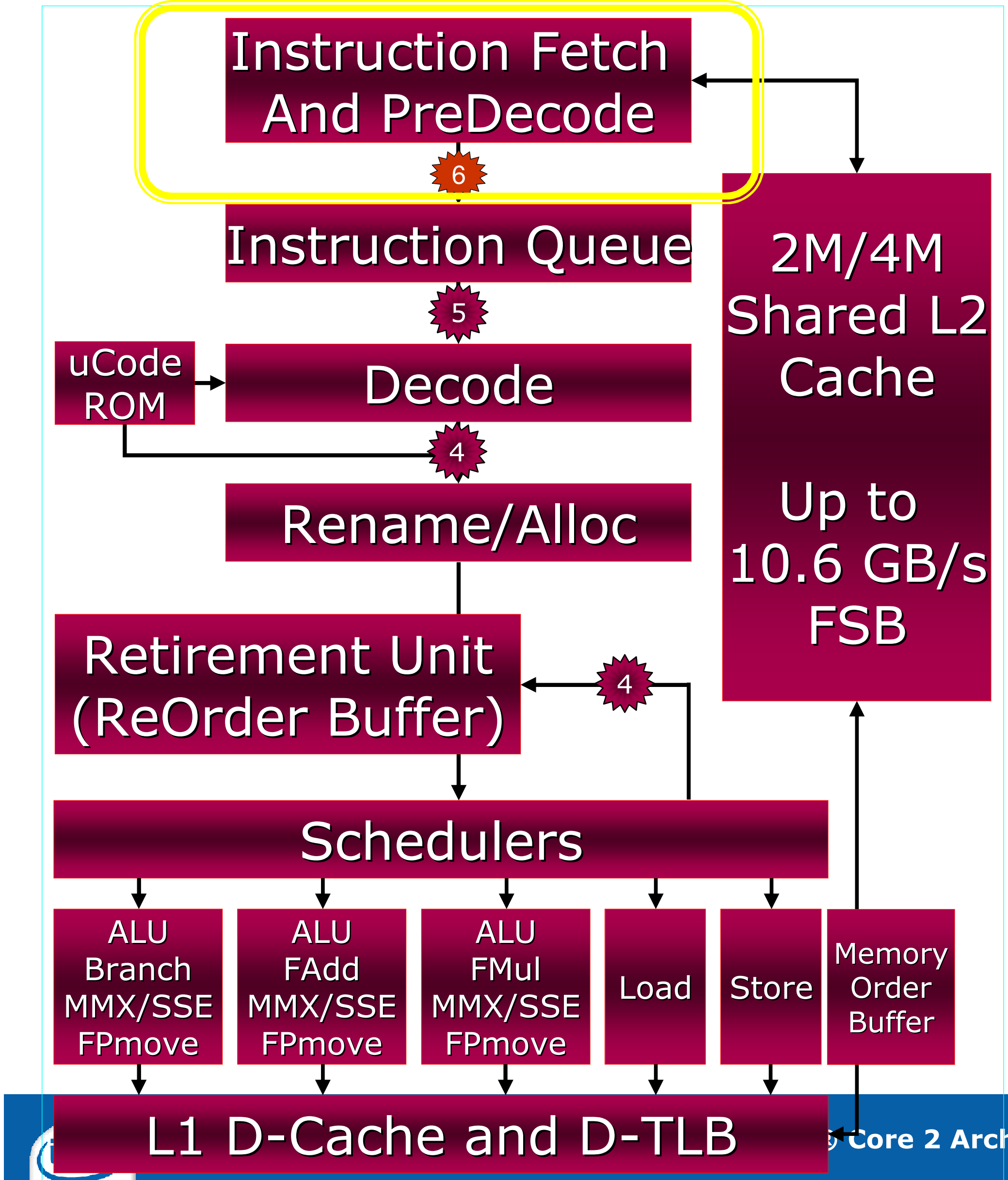
32KB L1 Instruction Cache

32KB L1 Data Cache

1 Thread per Core

4MB Shared L2 Cache

Instruction Fetch and PreDecode



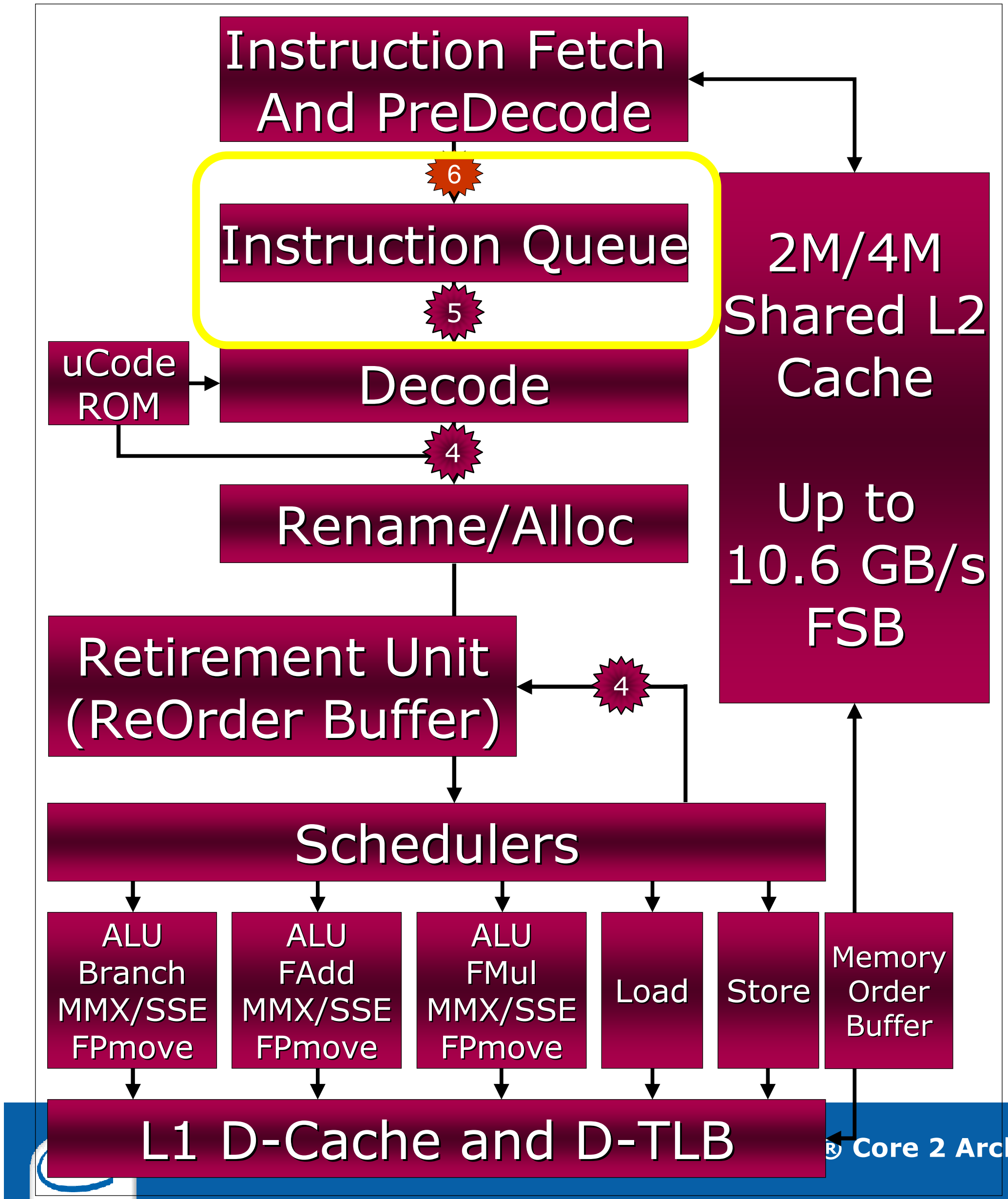
32 KBy Instruction Cache and ITLB

- One aligned 16 byte block per clock
- Branch target alignment makes the best use of this bandwidth

PreDecode

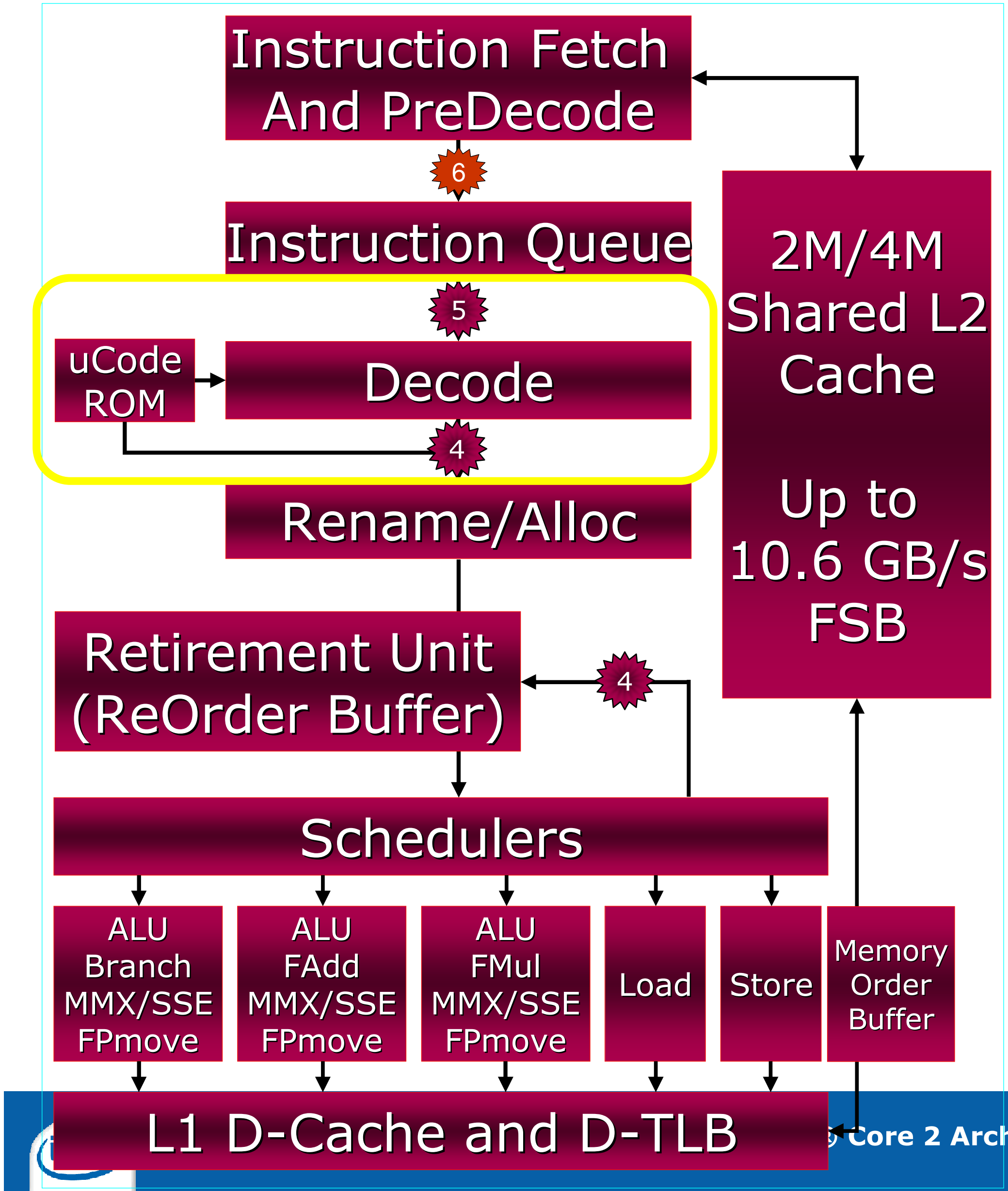
- Function: Instruction length marking to split the byte stream into discrete instructions
- As many as six instructions per cycle can be sent to the instruction queue

Instruction Queue



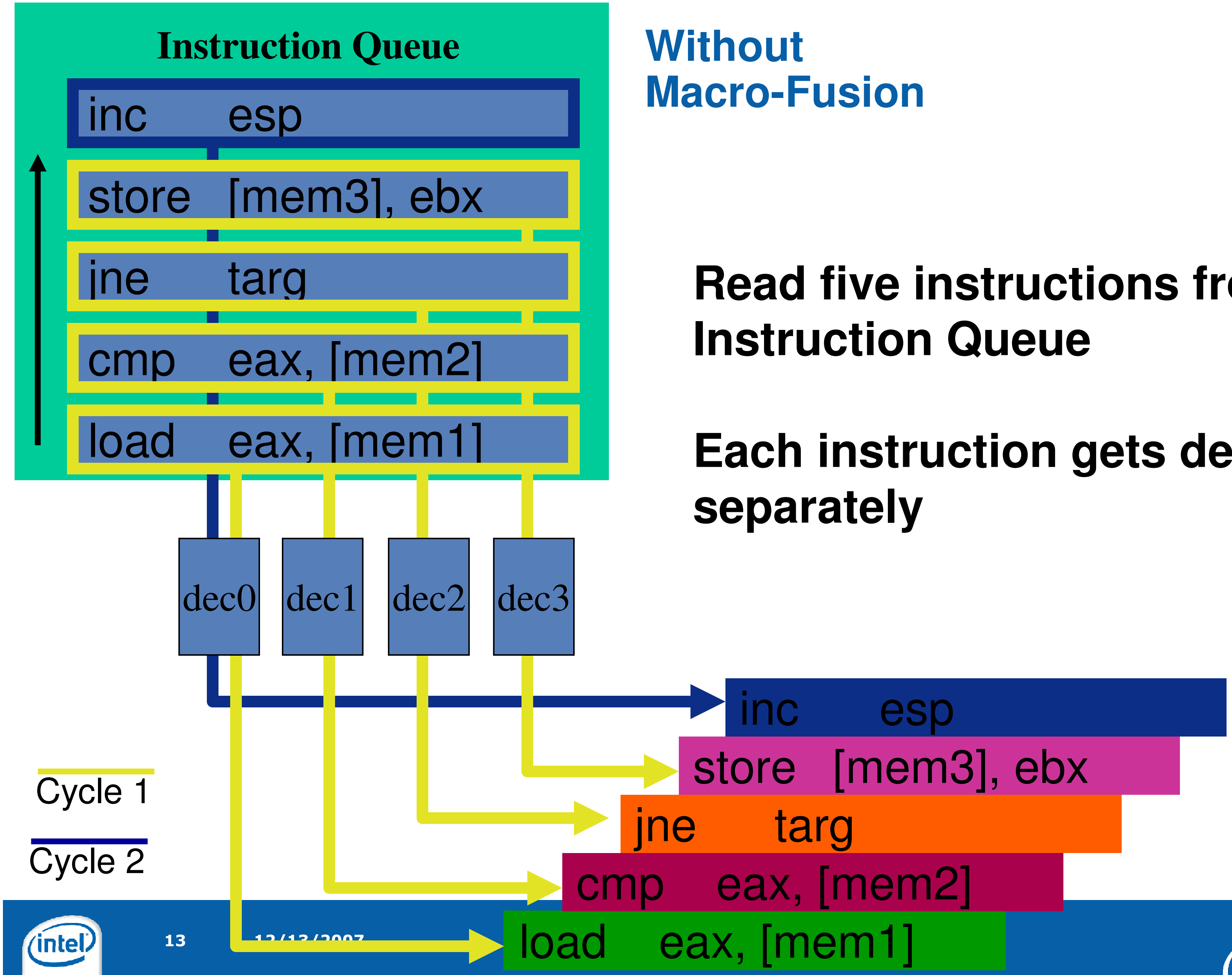
- **18 deep instruction queue**
  - 6 instructions write per cycle
  - 5 instructions read per cycle
    - One “Macro-fusion” per cycle
- **Includes a “Loop Stream Detector” (LSD)**
  - Potentially very high bandwidth instruction streaming
    - Maximum of 18 instructions in up to four 16-byte packets
    - No RET instructions (hence, little *practical* use for CALLs)
    - Up to four taken branches allowed
    - Most effective at 70+ iterations
  - Trade-off LSD with conventional loop unrolling

Instruction Decode



- **Macro Fusion**

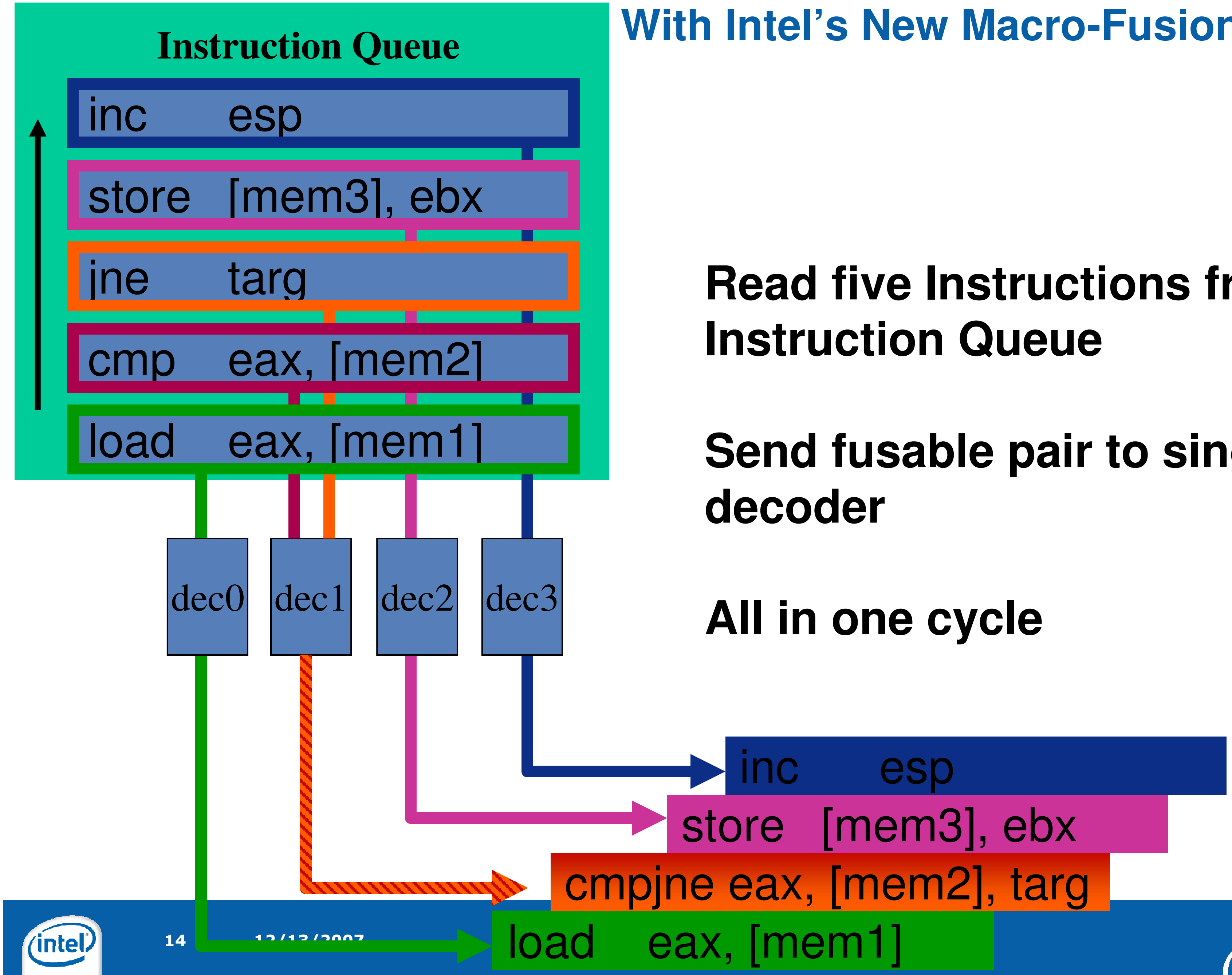
Without Macro-Fusion



Read five instructions from Instruction Queue

Each instruction gets decoded separately

With Intel’s New Macro-Fusion



Read five Instructions from Instruction Queue

Send fusable pair to single decoder

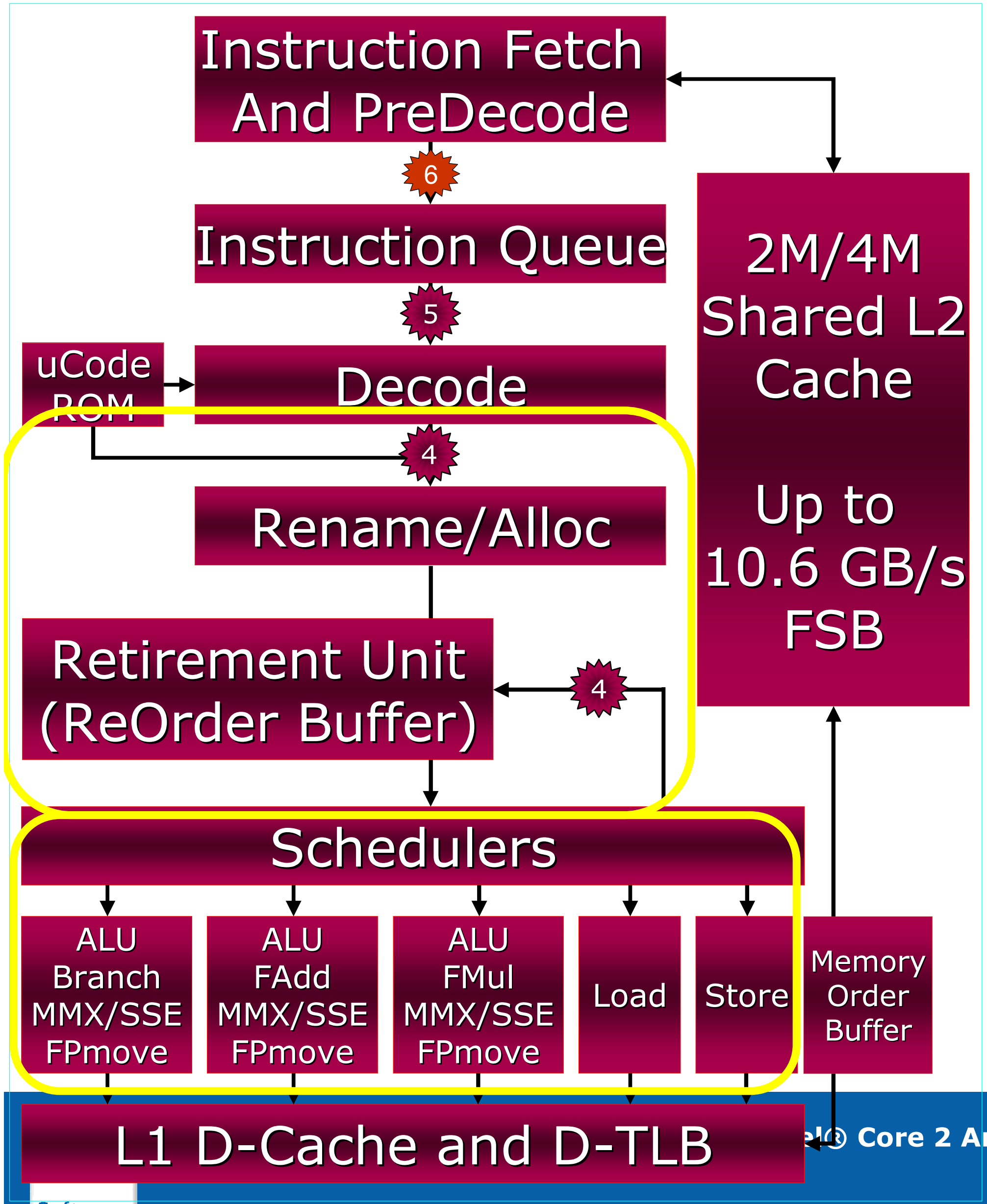
All in one cycle

Slide 14

ct1

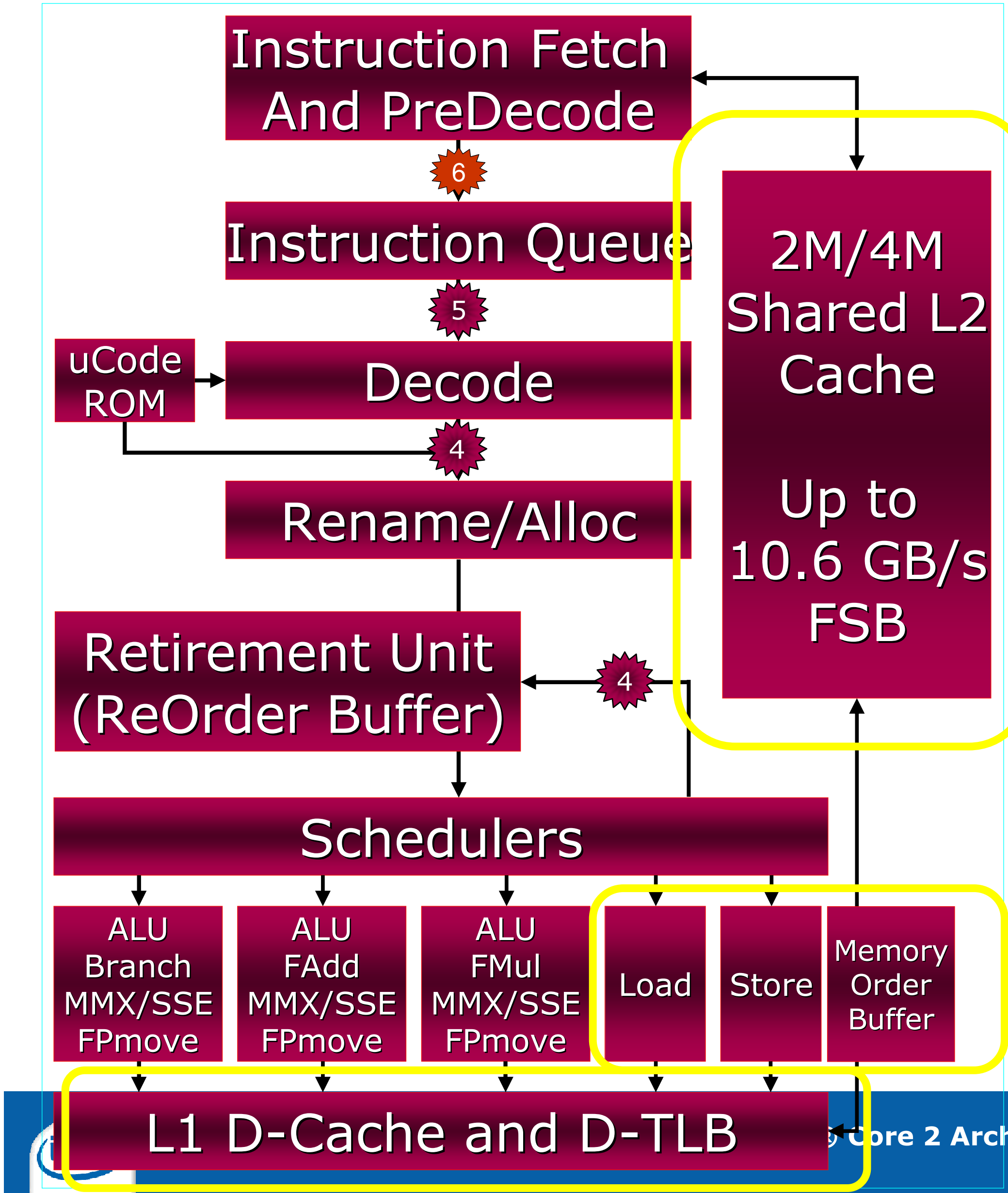
66% improvement due to
macro fusion and +1 decoder
Visually make NGMA bigger/better
ctaggard, 03/03/2006

Execution (In-order/Out-of-Order)



- Up to six uOps dispatch per clock
- Up to four results can be written back per clock

Memory

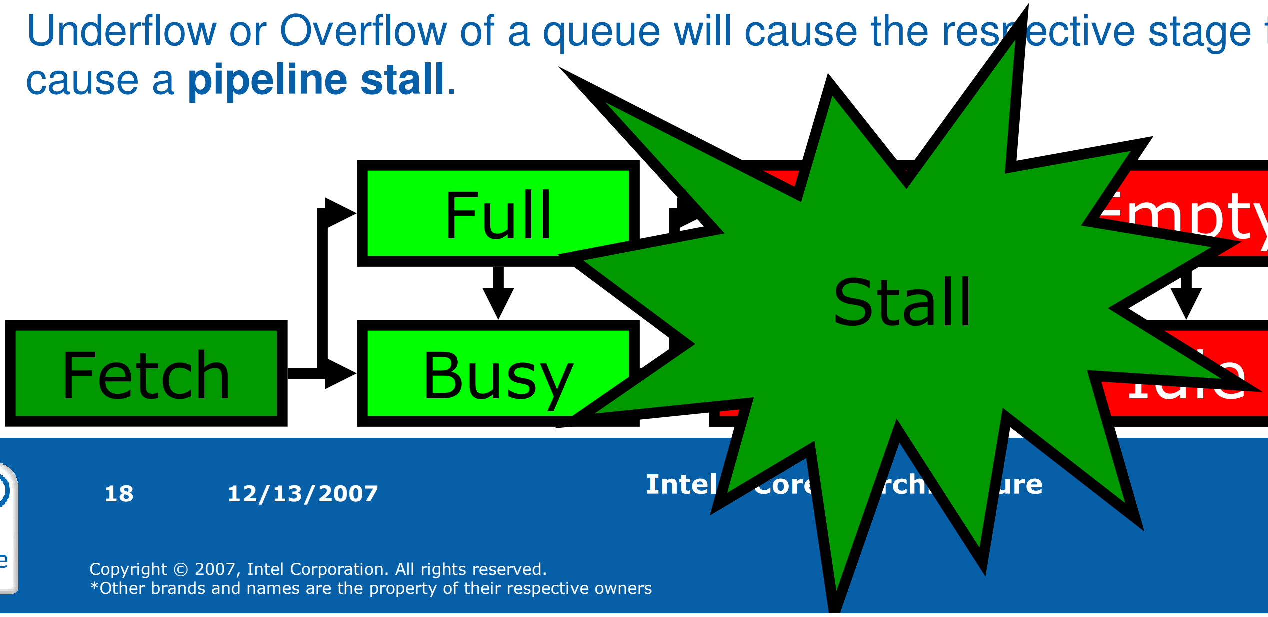


- **One load and one store per clock**
- **Load/store conflict detection uses the lower 12 bits**
  - Subject to 4 K memory address aliasing
- **Prefetchers**
  - Each core has two data prefetchers
    - One prefetches for specific instructions (“IP prefetch”)
    - One prefetcher detects streams
  - There are two prefetchers shared by the cores
    - An adjacent line prefetcher
    - “DPL” (Data prefetch logic) looks for more complex streams

Implications of Core 2 Architecture for Developers

Some Words on Pipelines (1)

- Modern CPUs may be understood by considering their basic design paradigm, the so-called **pipeline**. The pipeline is designed to break up the processing of a single instruction in independenent parts that idealy are executed in an identical time window.
- The independent parts of the processing are called **pipeline stages**.
- Since identical processing time in each stage can't be guaranteed, most pipeline stages control a **buffer or queue** that supplies instructions if the previous stage is still busy or in which instruction can be stored if the next stage is still busy.
- Underflow or Overflow of a queue will cause the respective stage to run idle and will cause a **pipeline stall**.

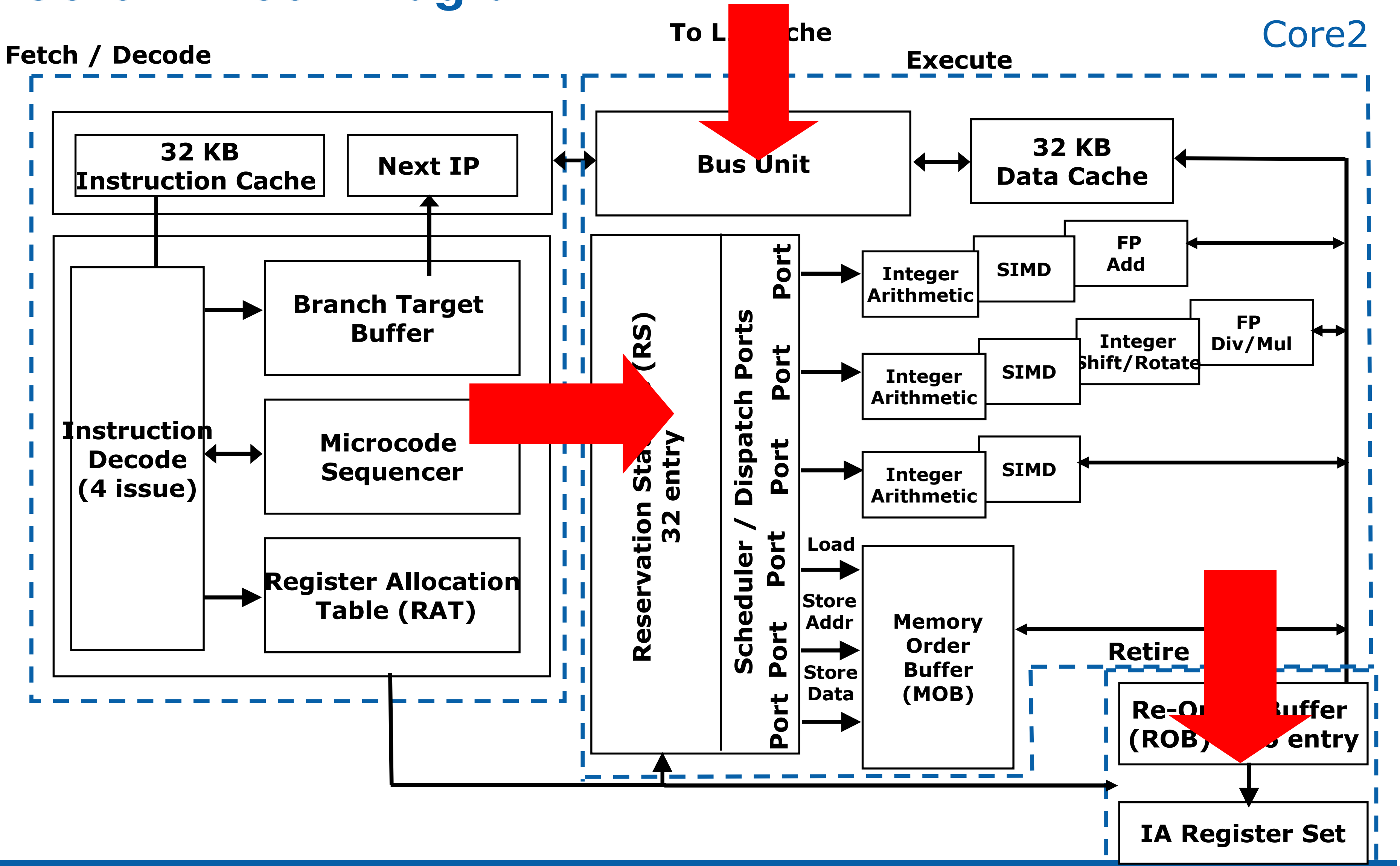


Some Words on Pipelines (2)

- In order to achieve the best performance
  - **<u>Pipeline stalls must be avoided</u>**
- Since Core 2 performance makes use of <u>speculative execution</u>, a wrongly taken branche might lead to a <u>pipeline flush</u> to keep the instructions consistent.
  - **<u>Pipeline flushes must be avoided</u>**

- Understanding the Core 2 pipeline and being able to detect pipeline problems will highly improve the performance of your software.

Core 2 Block Diagram



Vectorization – Streaming SIMD Extensions

Streaming SIMD Extensions (SSE)

SSE instructions operate on 8 128-bit wide registers

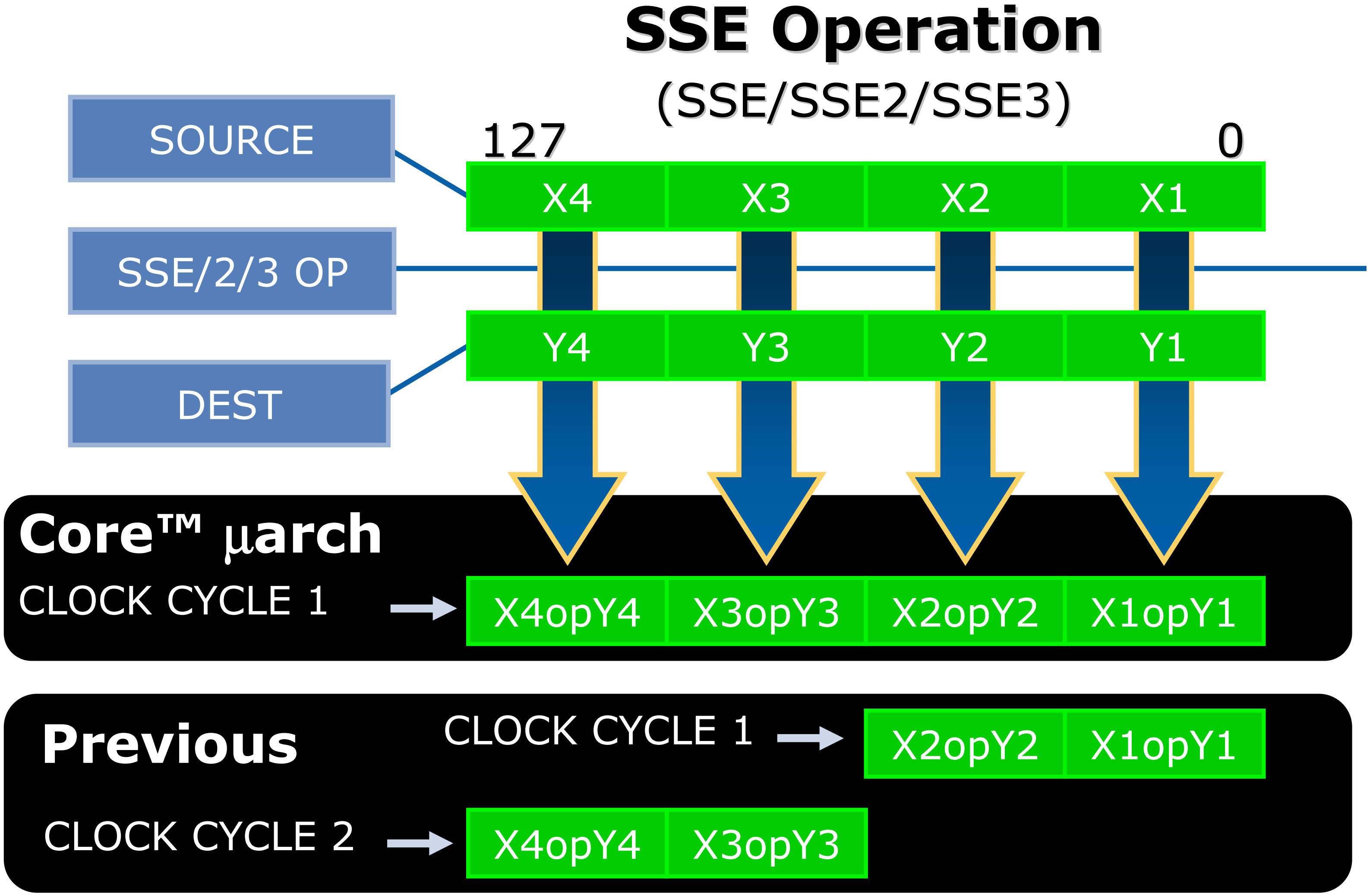
The registers can hold different data types, e.g

	127			0
2 doubles	Double 1		Double 2	
Or 4 floats	Float 1	Float 2	Float 1	Float 1

Every Core can has 3 SIMD units

Using SSE can speed up compute intensive applications extremely!

Vectorization
Advanced Digital Media Boost – Single Cycle SSE



Making use of Dual Core

- Up to now: Performance tuning = Serial performance tuning
- Core 2 is very fast even on a single core
- Most crucial advantage: Multi-core CPU!
- Parallel programming needed
    - Parallelization concept
    - Multi-threading
    - Synchronization
    - Communication
    - Load balancing

- **Multi-core means more effort for developers**

Summary

- Core 2 Microarchitecture is the state-of-the-art in x86 CPU design
- 64bit
- Dual-core
- 14 stages deep 4 instructions wide pipeline
- 128-bit SIMD registers
- Understanding the Core2 pipeline is compulsory for the best serial performance
- Understanding concurent programming and efficient parallel design are compulsory for best multi-core performance
  - **It‘s worth the time – Multi-Core is here to stay**