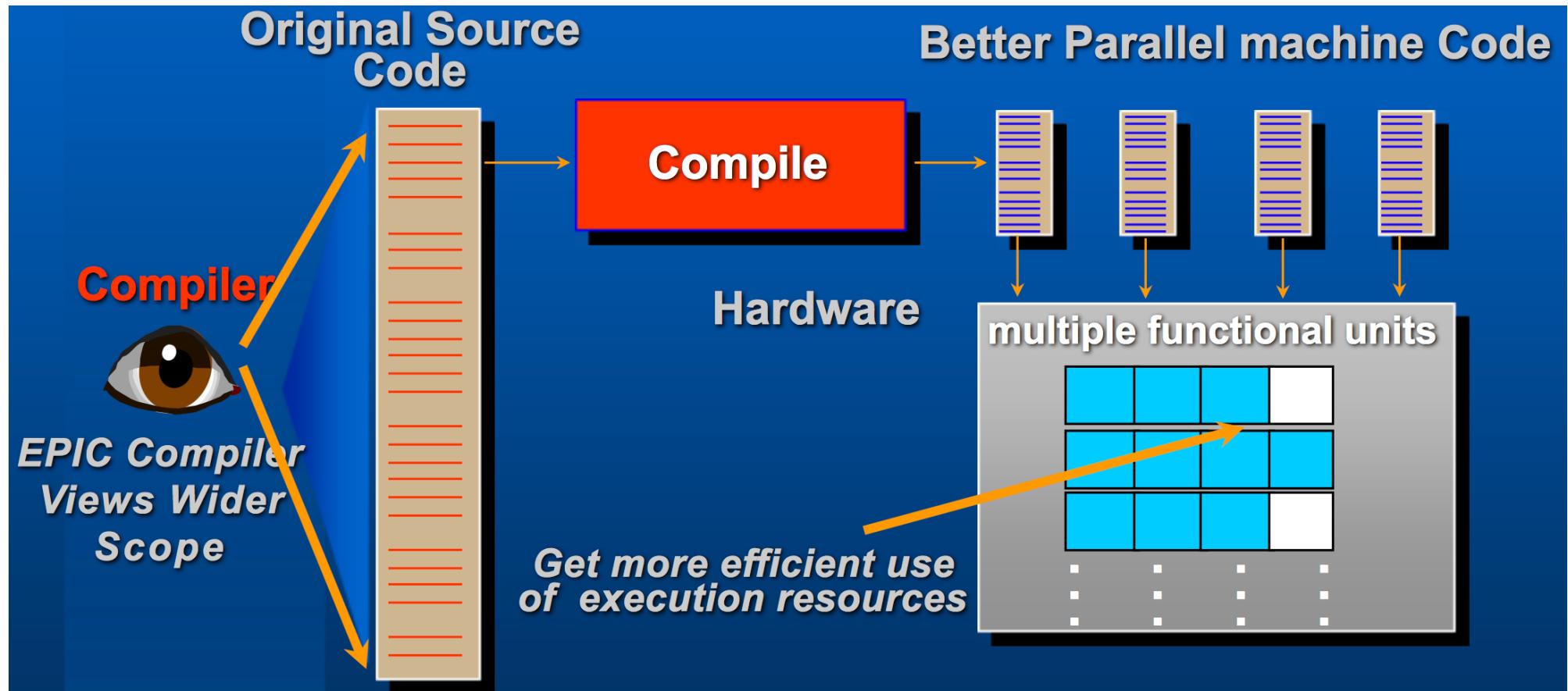


# EPIC: Explicit Parallelism



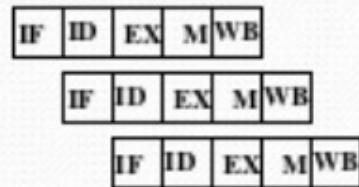
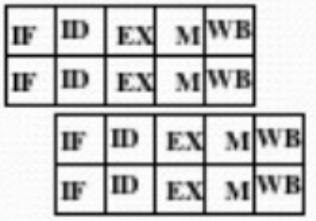
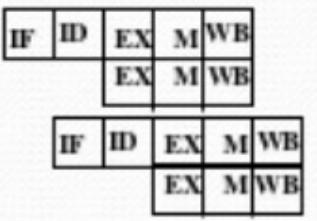
# EPIC: Integrating Superscalar and VLIW

**Processing instructions in parallel requires three major tasks:**

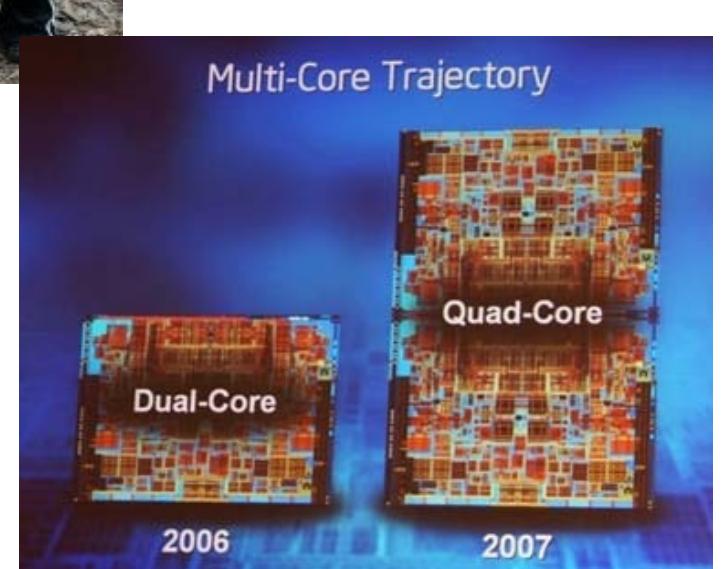
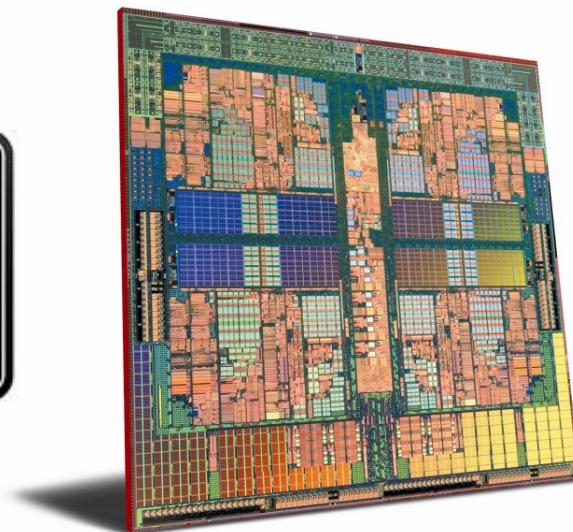
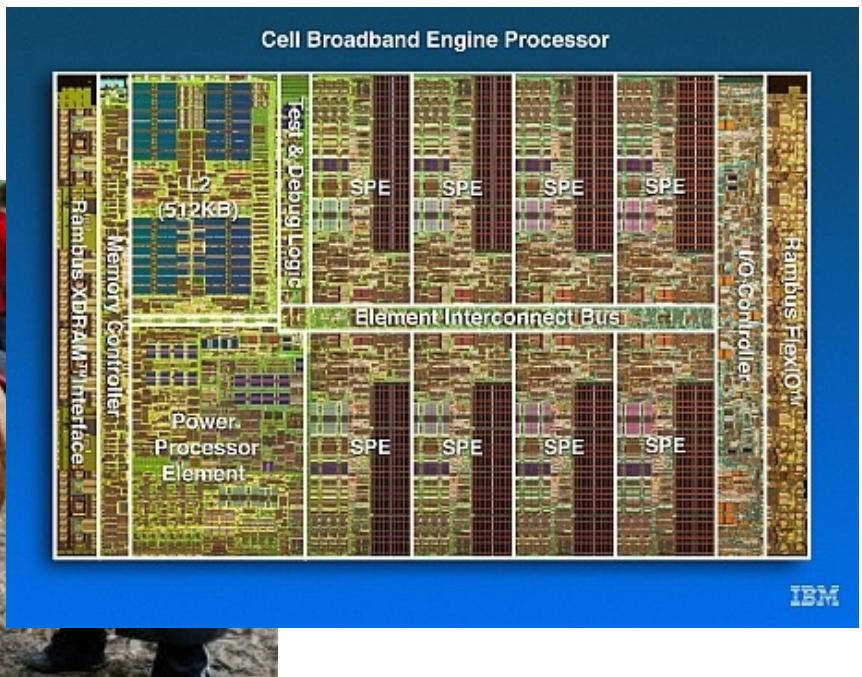
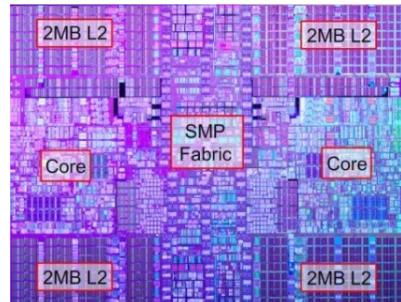
1. Checking dependencies between instructions to determine which instructions can be grouped together for parallel execution
2. Assigning instructions to the functional units on the hardware
3. Determining when instructions are initiated placed together into a single word (bundle)

	<b>Grouping</b>	<b>Fn unit asgn</b>	<b>Initiation</b>
<b>Superscalar</b>	Hardware	Hardware	Hardware
<b>EPIC</b>	Compiler	Hardware	Hardware
<b>Dynamic VLIW</b>	Compiler	Compiler	Hardware
<b>VLIW</b>	Compiler	Compiler	Compiler

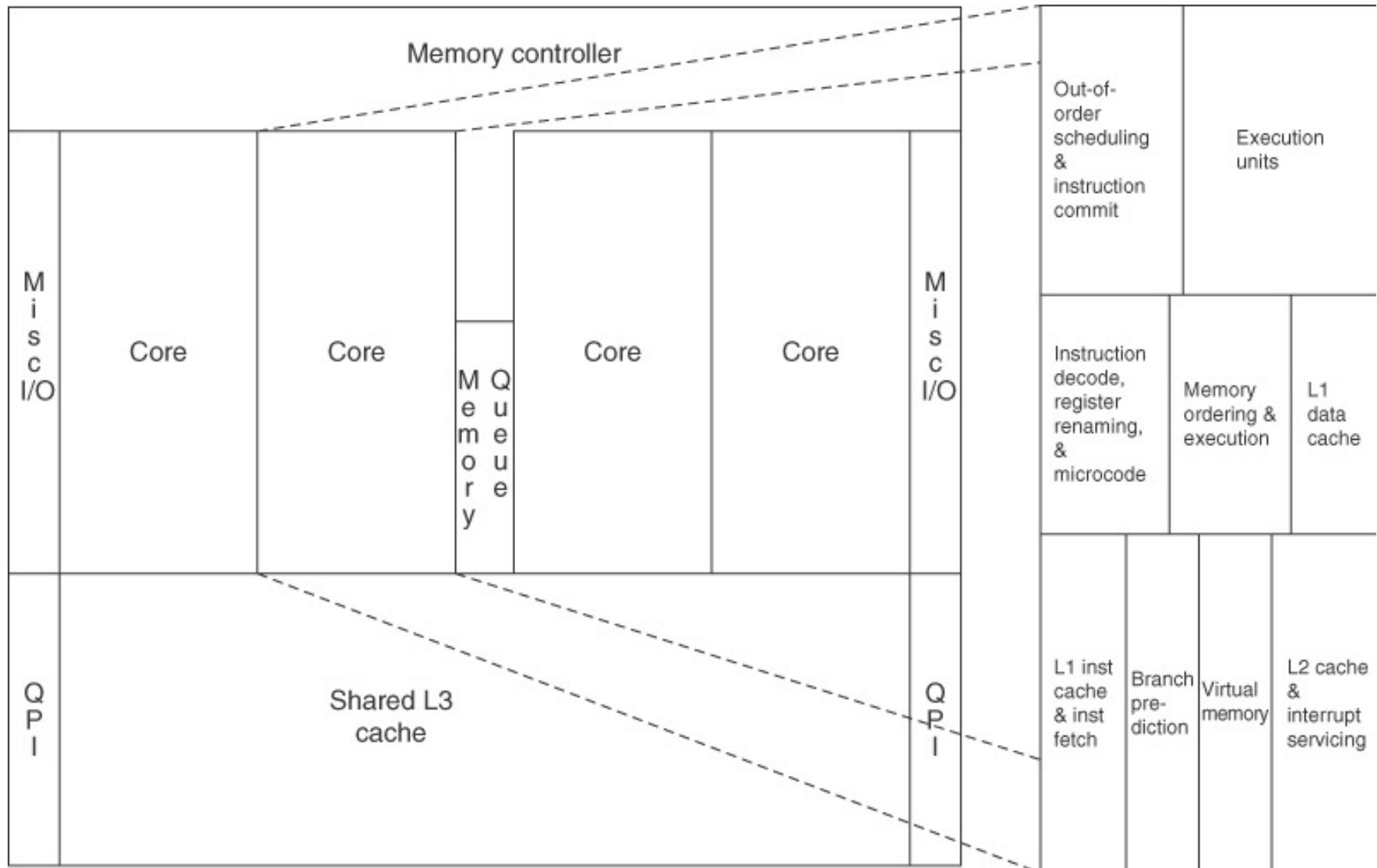
# Comparison

	<b>CISC</b>	<b>RISC</b>	<b>Superscalar</b>	<b>VLIW</b>
<b>Instruction size</b>	variable size	fixed size	fixed size	fixed size (but large)
<b>Instruction format</b>	variable format	fixed format	fixed format	fixed format
<b>Registers</b>	few, some special	many GP	GP and rename (RUU)	many, many GP
<b>Memory reference</b>	embedded in many instr's	load/store	load/store	load/store
<b>Key Issues</b>	decode complexity	data forwarding, hazards	hardware dependency resolution	code scheduling, (compiler)
<b>Instruction flow</b>		<div style="display: flex; justify-content: space-around; align-items: center;"> <div style="text-align: center;">  </div> </div>	<div style="display: flex; justify-content: space-around; align-items: center;"> <div style="text-align: center;">  </div> </div>	<div style="display: flex; justify-content: space-around; align-items: center;"> <div style="text-align: center;">  </div> </div>

# Multi-core Processor Gala

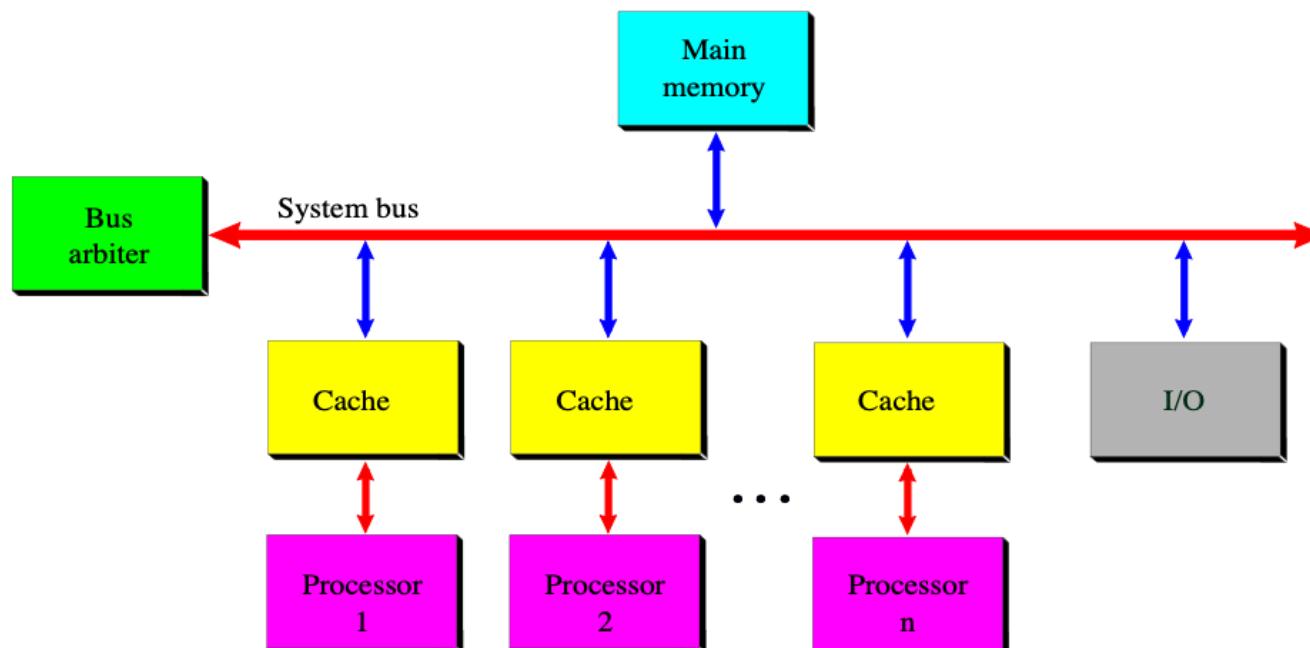


# Floorplan of Core i7 Die



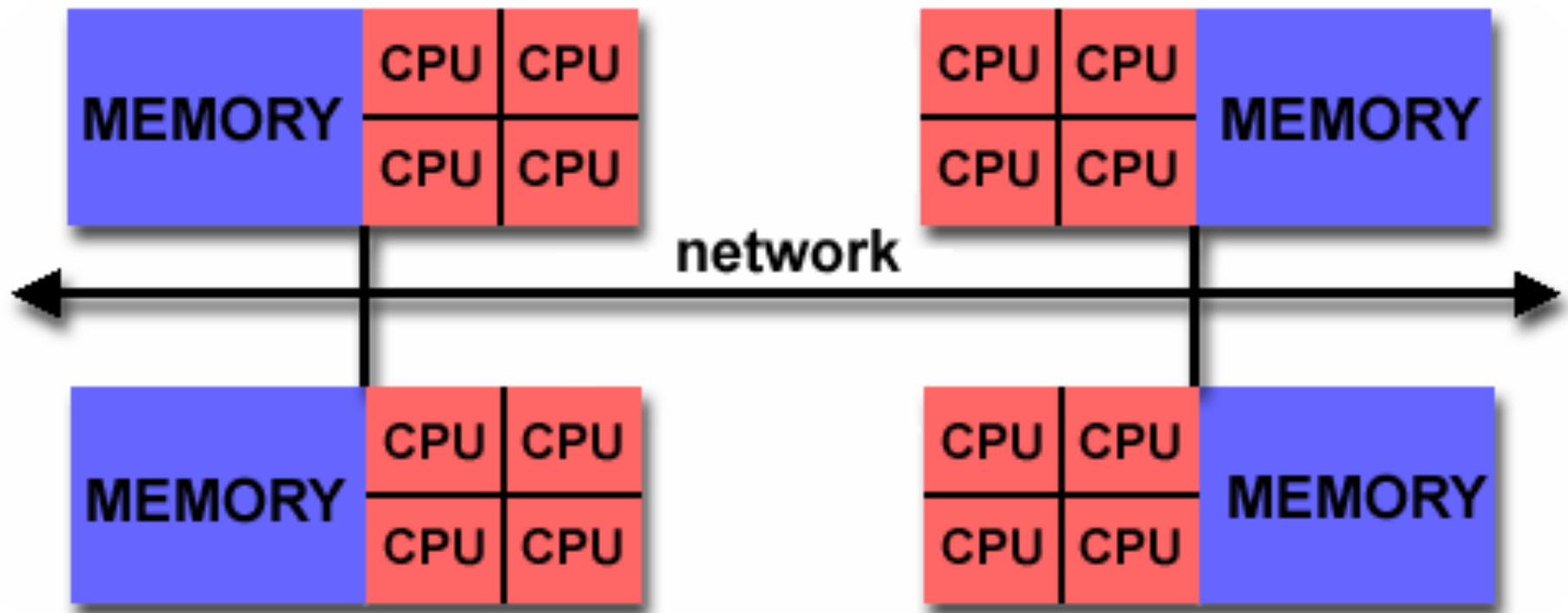
# Symmetric multiprocessing (SMP)

- **A multiprocessor computer hardware and software architecture**
  - Two or more identical processors are connected to a single, shared **main memory**, have full access to all input and output devices
  - Controlled by a single operating system instance that treats all processors equally, reserving none for special purposes
- **Most multiprocessor systems today use an SMP architecture**
  - In the case of **multi-core processors**, the SMP architecture applies to the cores, treating them as separate processors



# Current Trend: Computer Clusters

---

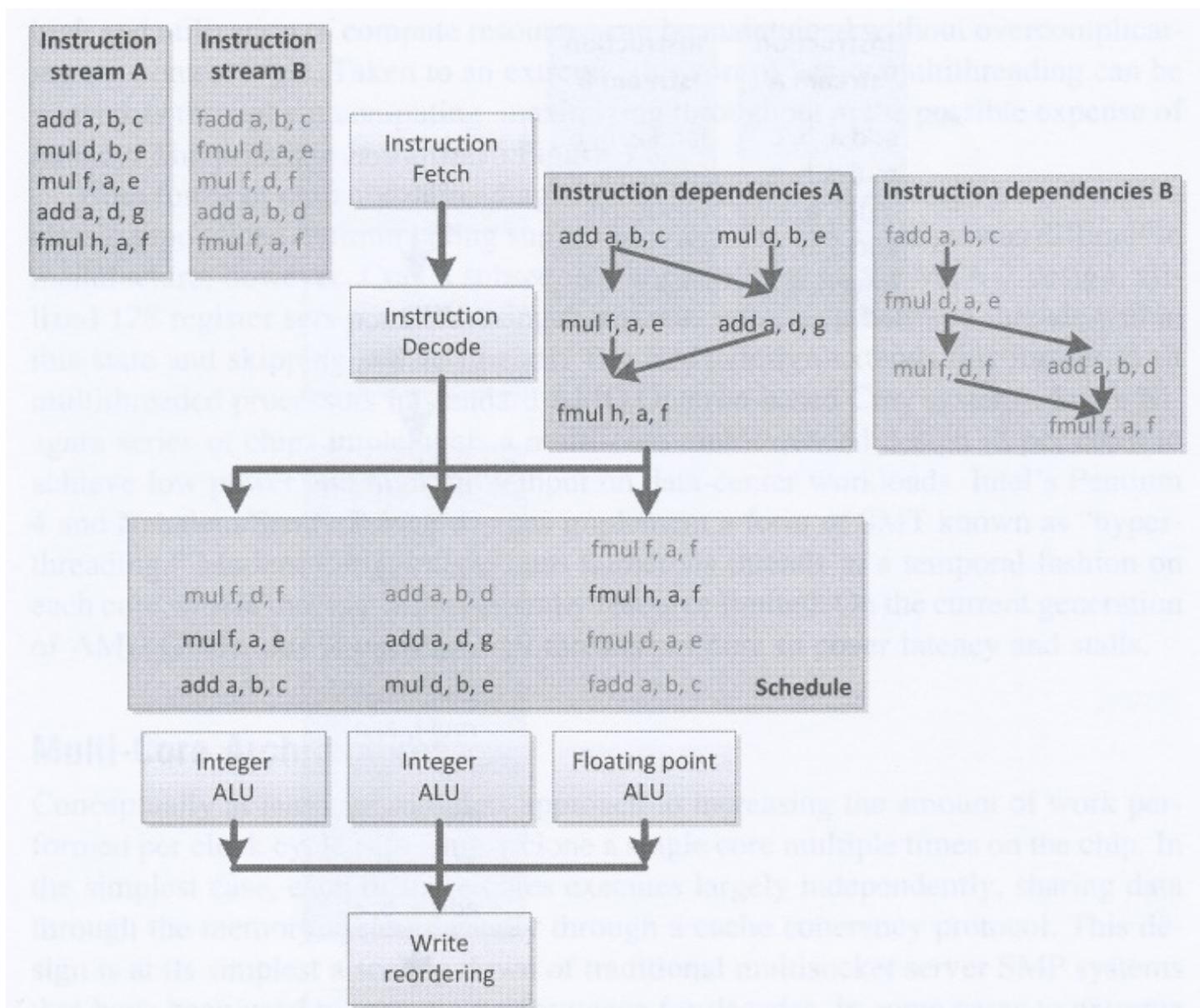


# Multithreading: Exploiting Thread-Level Parallelism

---

- **ILP has the great advantage that it is reasonably transparent to the programmer**
  - Quite limited or difficult to exploit in some applications
  - When the processor is stalled waiting on a cache miss, the utilization of the functional units drops dramatically
- **Multithreading allows multiple threads to share the functional units of a single processor in an overlapping fashion**
  - Multi-threading shares most of the processor core among a set of threads, duplicating only private state, such as the registers and program counter
  - Duplicating the per-thread state of a processor core means creating a separate register file, a separate PC, and a separate page table for each thread
  - A thread switch should be much more efficient than a process switch

# Multithreading



# Hardware Approaches to Multithreading

---

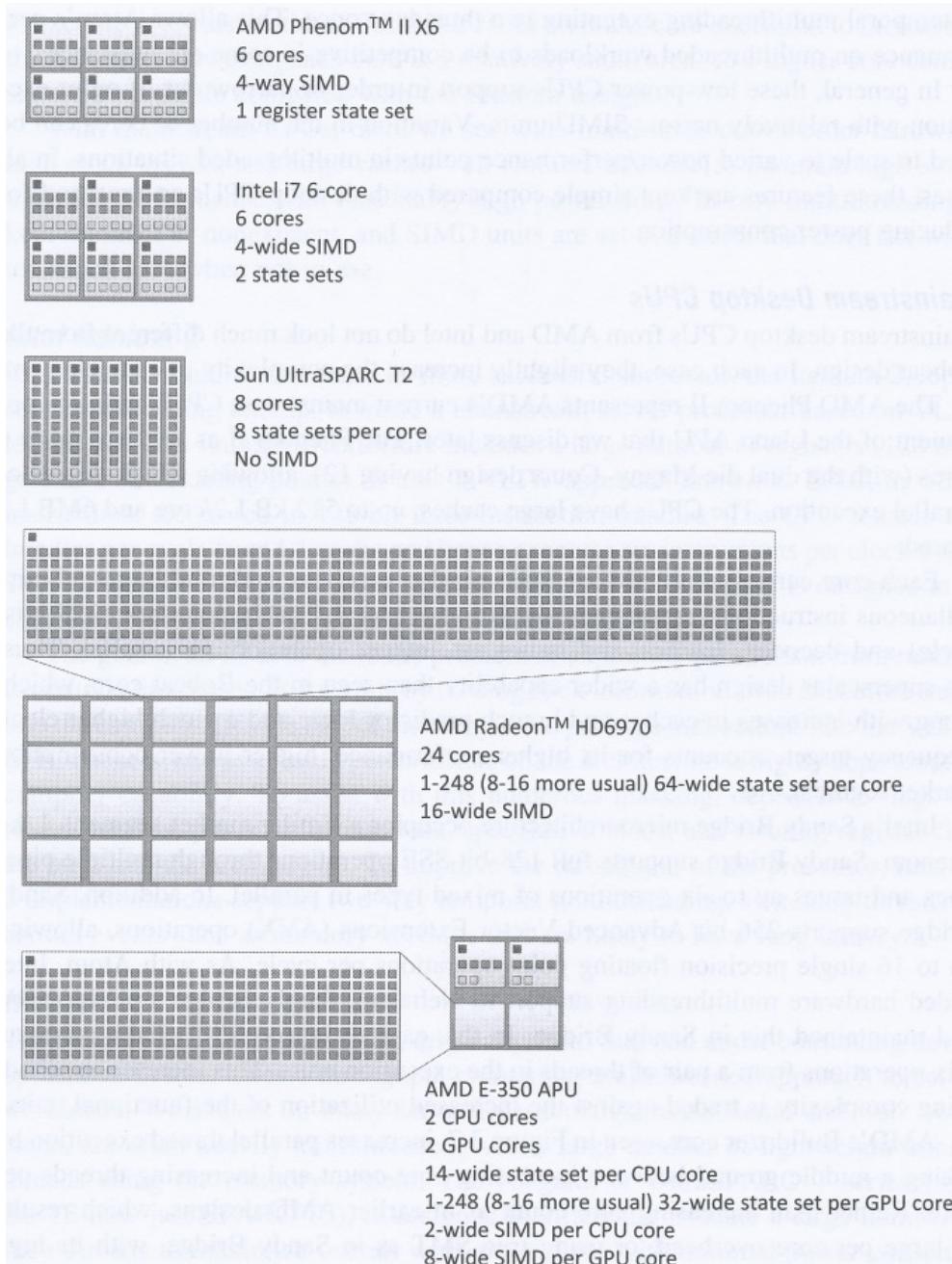
- **Fine-grained multithreading**
  - Switches between threads on each clock
    - Causing the execution of instructions from multiple threads to be interleaved
  - Hide the throughput losses that arise from both short and long stalls
    - Instructions from other threads can be executed when one thread stalls, even if the stall is only for a few cycles.
  - The primary disadvantage
    - Slows down the execution of an individual thread, since a thread that is ready to execute without stalls will be delayed by instructions from other threads
  - Sample implementation: Sun Niagara processor

# Hardware Approaches to Multithreading

---

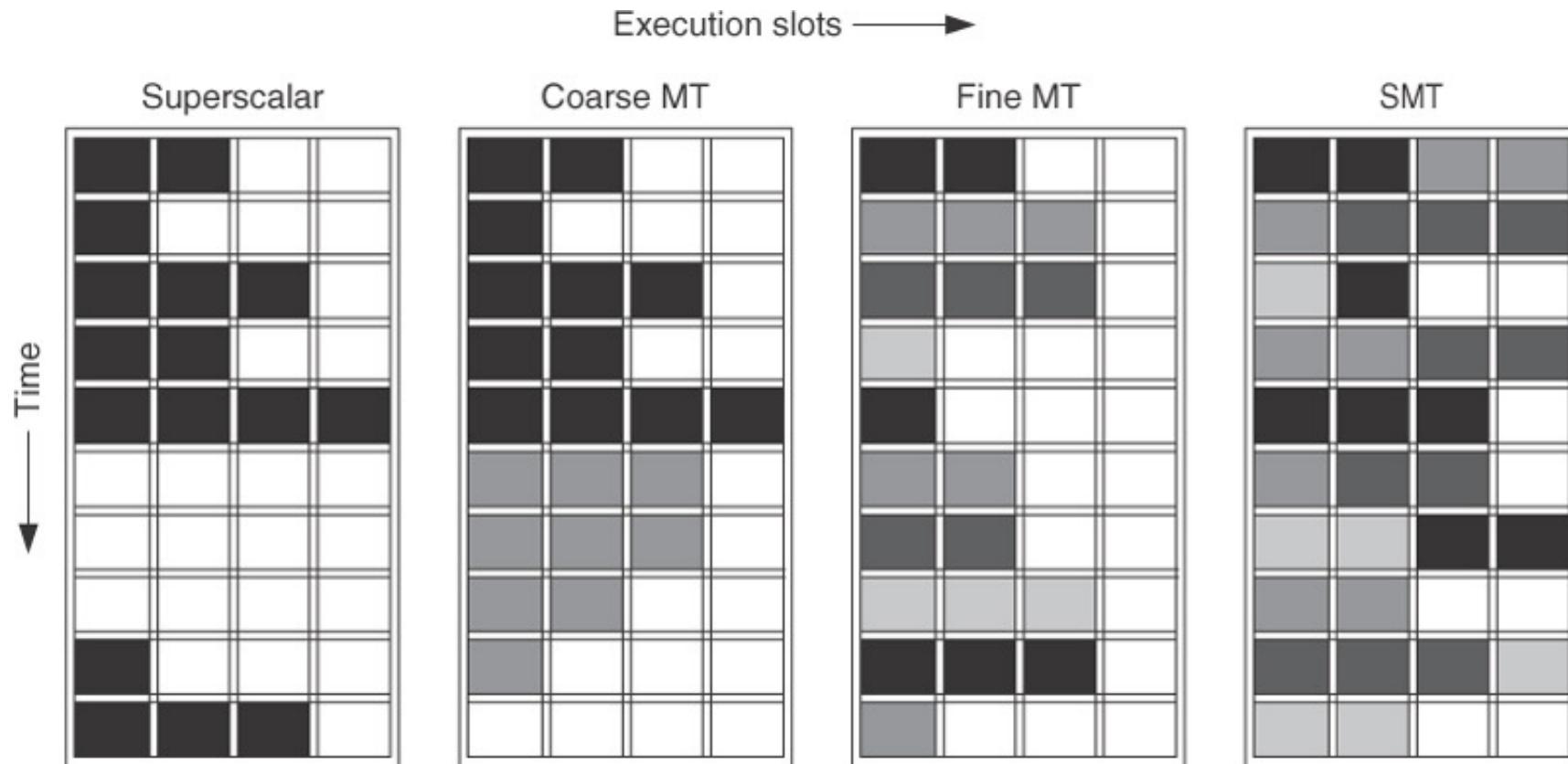
- **Coarse-grained multithreading**
  - Switches threads only on costly stalls, such as level two or three cache misses
    - High start-up overhead
  - Limited in its ability to overcome throughput losses, especially from shorter stalls
- **Simultaneous multithreading (SMT)**
  - A variation on fine-grained multithreading
  - Based on a multiple-issue, dynamically scheduled processor
  - Register renaming and dynamic scheduling allow multiple instructions from independent threads to be executed without regard to the dependences among them
    - The resolution of the dependences can be handled by the dynamic scheduling capability

# CPU and GPU Architectures

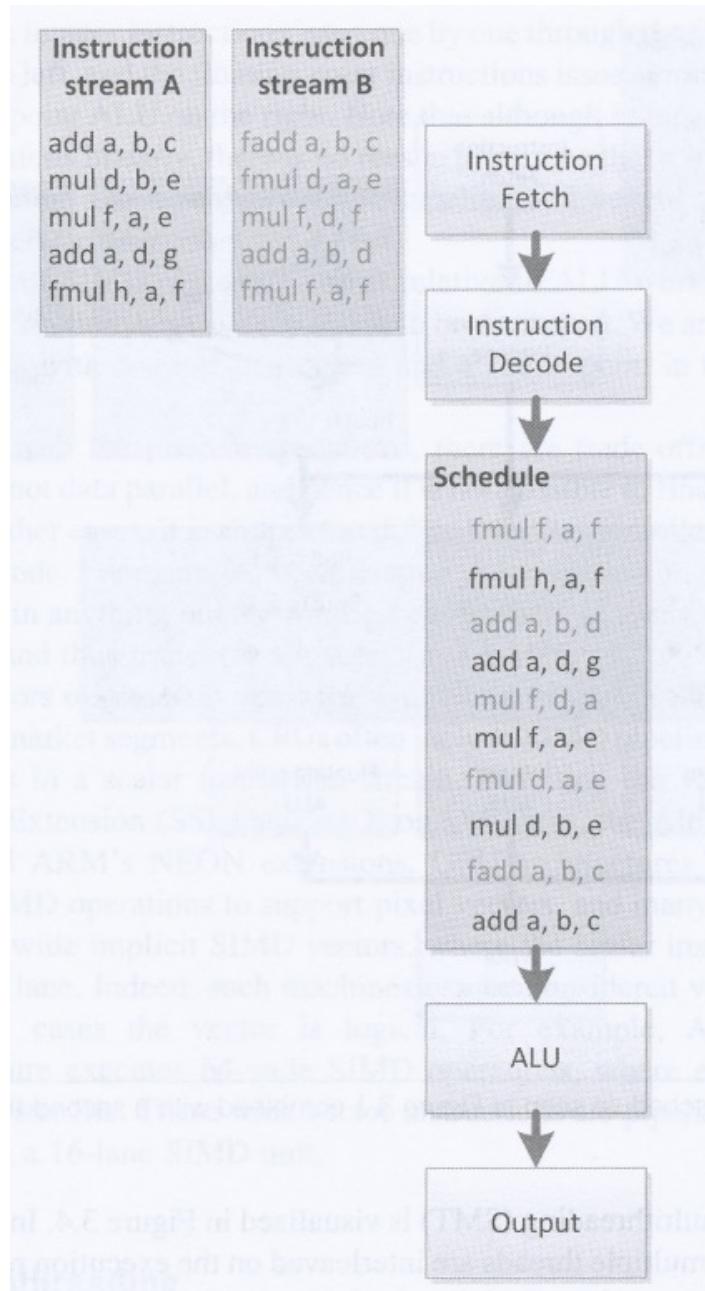


# Multithreading

- A superscalar with no multithreading support
- A superscalar with coarse-grained multithreading
- A superscalar with fine-grained multithreading
- A superscalar with simultaneous multithreading

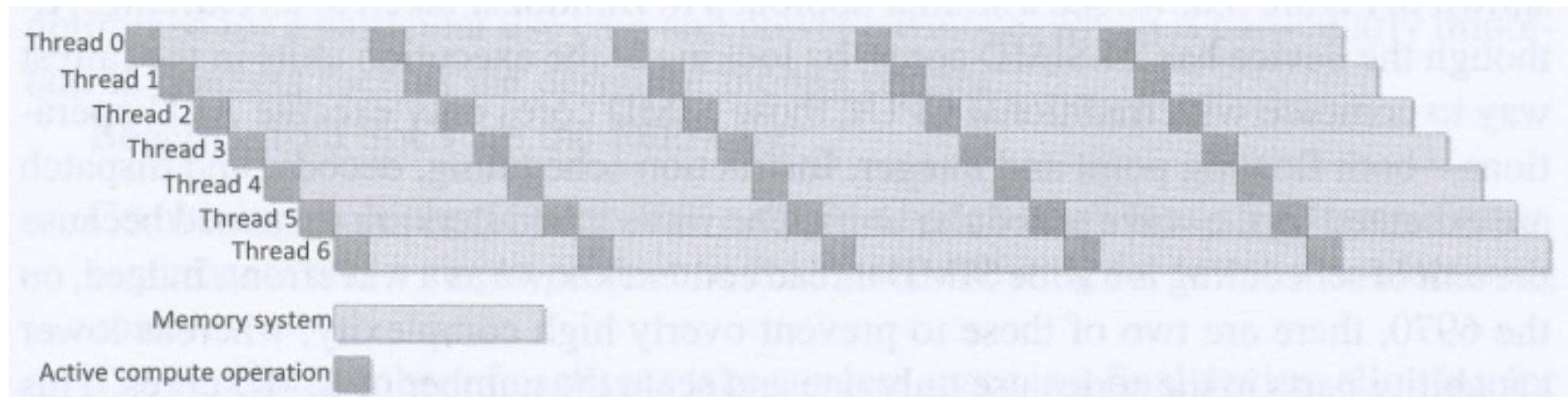


# Two Threads Scheduled in Time Slice Fashion



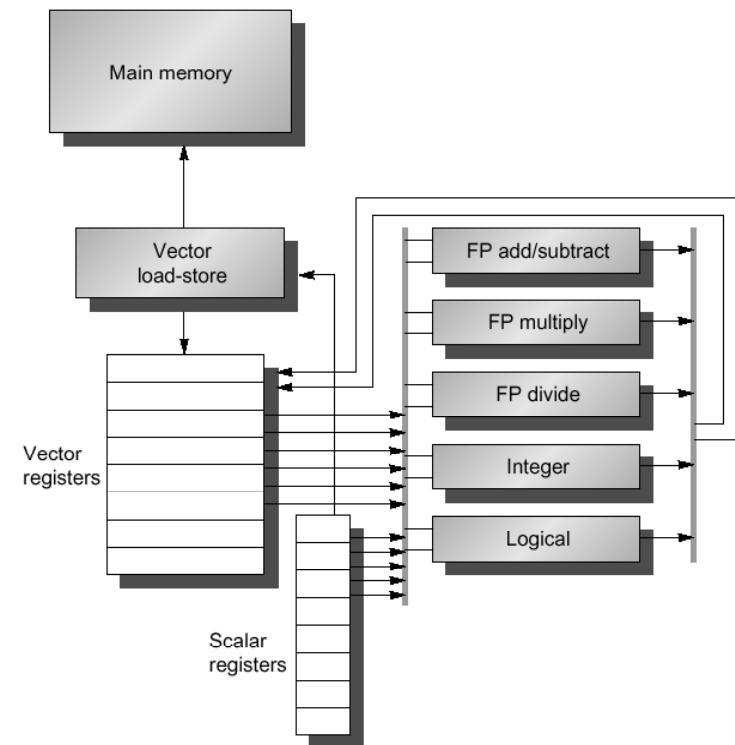
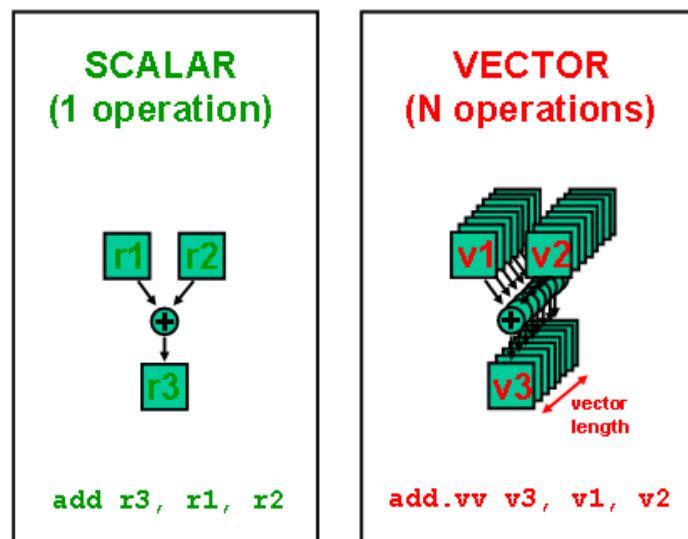
# Tolerate Latency via Hardware Threads

- A large number of threads interleave execution to keep the device busy, whereas each individual thread takes longer to execute than the theoretical minimum



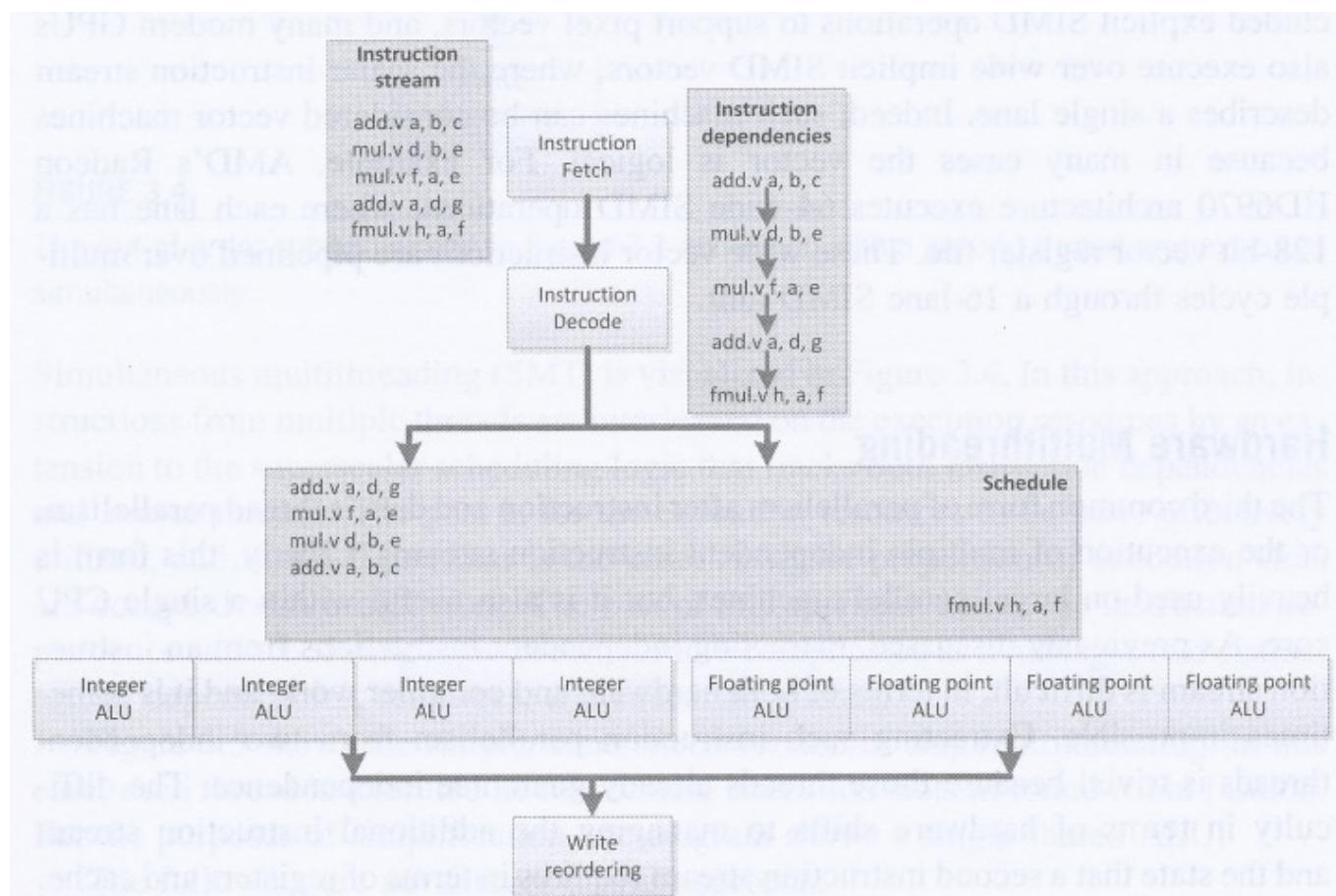
# Data Parallelism: Vector Processors

- **Provides high-level operations that work on vectors**
  - Vector is a linear array of numbers
    - Type of number can vary, but usually 64 bit floating point (IEEE 754, 2's complement)
    - Length of the array also varies depending on hardware
      - Example vectors would be 64 or 128 elements in length
      - Small vectors (e.g. MMX/SSE) are about 4 elements in length



# SIMD and Vector Processing

- SIMD and its generalization in vector parallelism approach improved efficiency by:
  - The same operation be performed on multiple data elements



# Parallelism Manners

