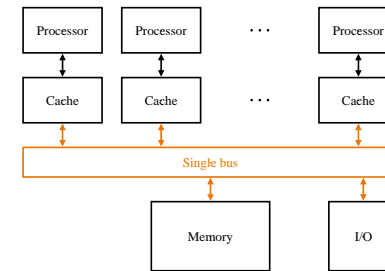


Multiprocessors and Multithreading

more is better?

Multiprocessors

- why would you want a multiprocessor?
- what things can it do well?
- What things can't it do well?

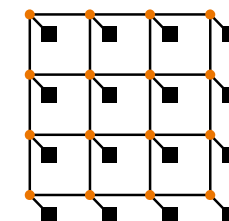
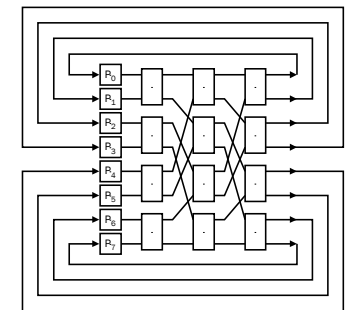
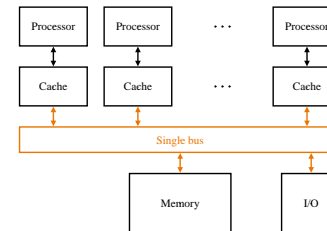


Classifying Multiprocessors

- Interconnection Network
- Memory Topology
- Programming Model

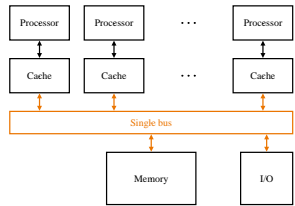
Interconnection Network

- Bus
- Network
- pros/cons?

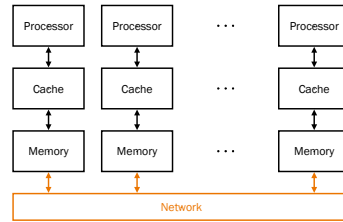
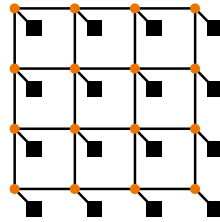


Memory Topology

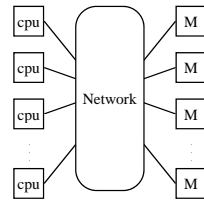
- **UMA** (Uniform Memory Access)
- **NUMA** (Non-uniform Memory Access)
- pros/cons?



CSE 141

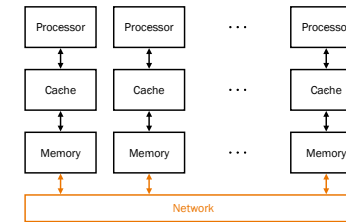


Dean Tullsen



Programming Model

- **Shared Memory** -- every processor can name every address location
- **Message Passing** -- each processor can name only its local memory. Communication is through explicit messages.
- pros/cons?



- *find the max of 100,000 integers on 10 processors.*

CSE 141

Dean Tullsen

Parallel Programming

Processor A

index = i++;

i = 47

Processor B

index = i++;

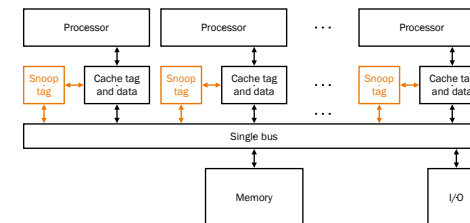
- Shared-memory programming requires synchronization to provide mutual exclusion and prevent race conditions
 - locks (semaphores)
 - barriers

CSE 141

Dean Tullsen

Cache Coherency

- **write-update**
 - on each write, each cache holding that location updates its value
- **write-invalidate** <= most common
 - on each write, each cache holding that location invalidates the cache line.



- both schemes MUCH easier on a bus-based multiprocessor
- potentially requires a LOT of messages, but...

CSE 141

Dean Tullsen

Cache Coherency

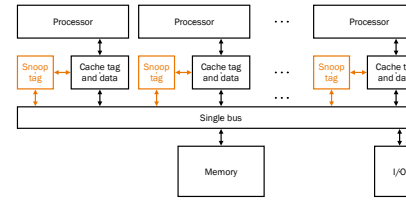
- A good cache coherency protocol can avoid sending unnecessary (and expensive) invalidate or update messages.
- Allows each cache line to be in one of several *states*.
- MESI (Illinois)
 - modified
 - exclusive
 - shared
 - Invalid
- What happens:
 - Load miss?
 - Store hit to exclusive line? Modified? Shared?

CSE 141 Store miss?

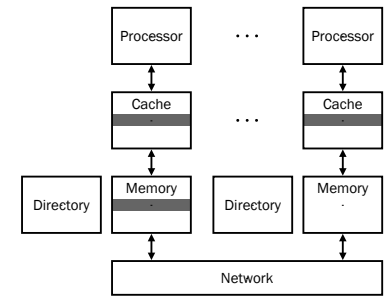
Dean Tullsen

Cache Coherency

- How do you know when an external read/write occurs?
- Snooping protocols
- Directory protocols



CSE 141



Dean Tullsen

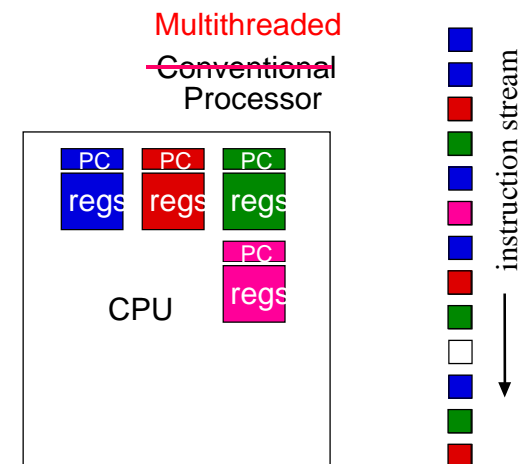
Simultaneous Multithreading

(1996 Thesis Slides)

CSE 141

Dean Tullsen

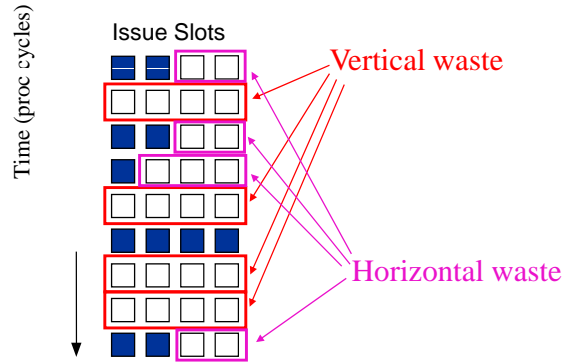
Hardware Multithreading



CSE 141

Dean Tullsen

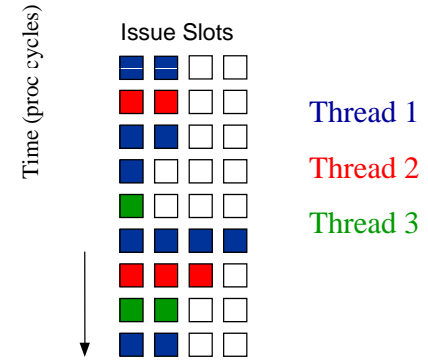
Superscalar Execution



CSE 141

Dean Tullsen

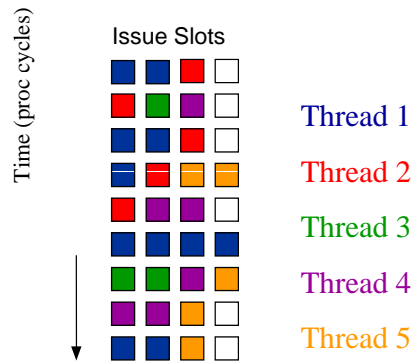
Superscalar Execution with Fine-Grain Multithreading



CSE 141

Dean Tullsen

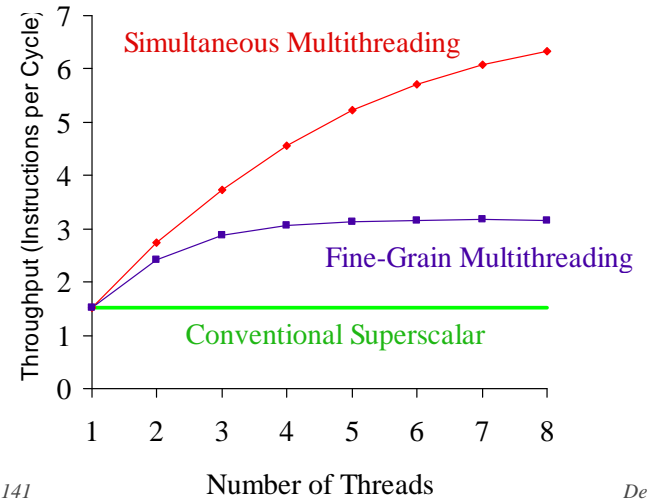
Simultaneous Multithreading



CSE 141

Dean Tullsen

SMT Performance



CSE 141

Dean Tullsen

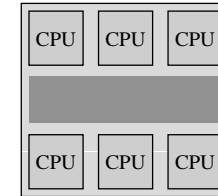
Multithreaded Processors

- Coarse-grain multithreading (Alewife-MIT)
 - context switch at long-latency operations (cache misses)
- Fine-grain multithreading (Tera Supercomputer)
 - context switch every cycle
 - Sun Niagara T1
- **Simultaneous multithreading (SMT)** (Tullsen, Eggers, Levy 1995)
 - execute instructions from multiple threads in the same cycle
 - is only different from fine-grain multithreading in the context of superscalar execution
 - requires surprisingly few changes to a conventional out-of-order superscalar processor
 - Was announced to be featured in the next Compaq Alpha processor (21464), but that processor never completed.
 - Introduced in the Intel Pentium 4 processor – announced as “Hyper-threading technology.” (HT Technology)
 - IBM Power 5, 6 has 2 cores, each 2-way SMT.
 - Nehalem Multicore has 2 threads/core

CSE 141

Dean Tullsen

Multi-Core Processors (aka Chip Multiprocessors)

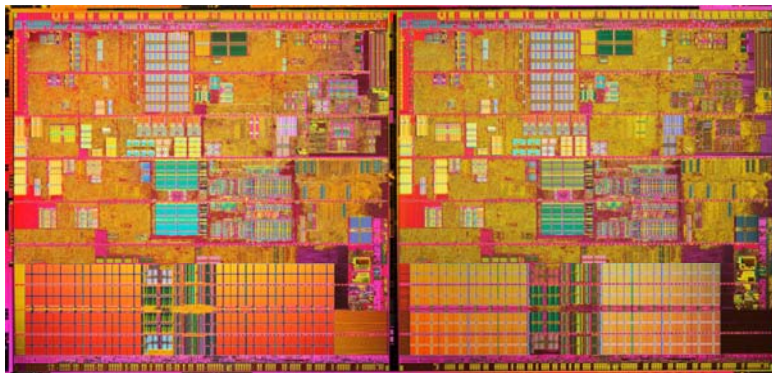


- Multiple cores on the same die, may or may not share L2 cache.
- Intel, AMD both have quad core processors. Sun Niagara T2 is 8 cores x 8 threads (64 contexts!)
- Everyone’s roadmap seems to be increasingly multi-core.

CSE 141

Dean Tullsen

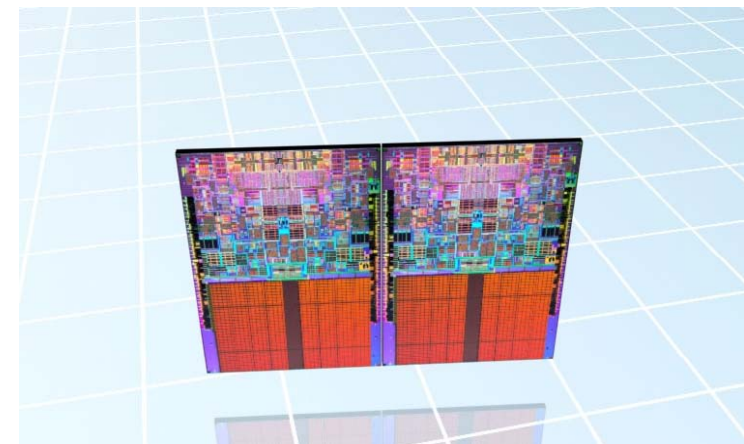
Intel Core 2 Duo



CSE 141

Dean Tullsen

Intel Quad Core



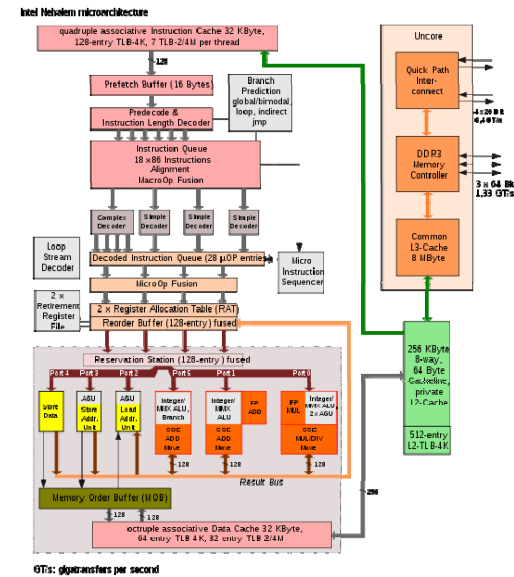
CSE 141

Dean Tullsen

Intel Nehalem

- First instantiation – Intel Core i7

Intel Nehalem Core



Intel Nehalem, summary

- Up to 8 cores (i7, 4 cores)
- 2 SMT threads per core
- 20+ stage pipeline
- x86 instructions translated to RISC-like uops
- Superscalar, 4 “instructions” (uops) per cycle (more with fusing)
- Caches (i7)
 - 32KB 4-way set-associative I cache per core
 - 32KB, 8-way set-associative D cache per core
 - 256 KB unified 8-way set-associative L2 cache per core
 - 8 MB shared 16-way set-associative L3 cache

Multiprocessors -- Key Points

- Network vs. Bus
- Message-passing vs. Shared Memory
- Shared Memory is more intuitive, but creates problems for both the programmer (memory consistency, requiring synchronization) and the architect (cache coherency).
- Multithreading gives the illusion of multiprocessing (including, in many cases, the performance) with very little additional hardware.
- When multiprocessing happens within a single die/processor, we call that a chip multiprocessor, or a multi-core architecture.