Implementing MP Machines

- One approach: add sockets to your MOBO
  - minimal changes to existing CPUs
  - power delivery, heat removal and I/O not too bad since each chip has own set of pins and cooling

Chip-Multiprocessing

- Simple SMP on the same chip
Shared Caches

- Resources can be shared between CPUs
  - ex. IBM Power 5

  L2 cache shared between both CPUs (no need to keep two copies coherent)

  L3 cache is also shared (only tags are on-chip, data are off-chip)

Benefits?

- Cheaper than mobo-based SMP
  - all/most interface logic integrated on to main chip (fewer total chips, single CPU socket, single interface to main memory)
  - less power than mobo-based SMP as well (communication on-die is more power-efficient than chip-to-chip communication)

- Performance
  - on-chip communication is faster

- Efficiency
  - potentially better use of hardware resources than trying to make wider/more OOO single-threaded CPU

Multithreaded Processors

- Single thread in superscalar execution: dependences cause most of stalls
- Idea: when one thread stalled, other can go
- Different granularities of multithreading
  - Coarse MT: can change thread every few cycles
  - Fine MT: can change thread every cycle
  - Simultaneous Multithreading (SMT)
    - Instrs from different threads even in the same cycle
    - AKA Hyperthreading
Simultaneous Multi-Threading

- Uni-Processor: 4-6 wide, lucky if you get 1-2 IPC
  - poor utilization
- SMP: 2-4 CPUs, but need independent tasks
  - else poor utilization as well
- SMT: Idea is to use a single large uni-processor as a multi-processor

SMT (2)

Overview of SMT Hardware Changes

- For an N-way (N threads) SMT, we need:
  - Fetch:
    - Ability to fetch from N threads, multiple PCs
  - Rename
    - N rename tables (RATs)
    - N ARF
  - Need to maintain interrupts, exceptions, faults on a per-thread basis
- But we don't need to replicate the entire OOO execution engine (schedulers, execution units, bypass networks, ROBs, etc.)
**SMT Cache**

- Each process has its own virtual address space
  - TLB must be thread-aware
    - translate (thread-id, virtual page) → physical page
  - Virtual portion of caches must also be thread-aware
    - VIVT cache must now be (virtual addr, thread-id)-indexed, (virtual addr, thread-id)-tagged
    - Similar for VIPT cache
    - No changes needed if using PIPT cache (like L2)

**This is all combinable**

- Can have a system that supports SMP, CMP and SMT at the same time
  - Take a dual-socket SMP motherboard...
  - Insert two chips, each with a dual-core CMP...
  - Where each core supports two-way SMT
- → Nehalem
- This example provides 8 threads worth of execution, shared on 4 actual “cores”, split across two physical packages

**OS Confusion**

- SMT/CMP is supposed to look like multiple CPUs to the software/OS
  - Schedule tasks to (virtual) CPUs
  - Say OS has n+1 idle tasks to run...
  - 2 cores (either SMP/CMP)
  - Windows

Example:
- Performance worse than if SMP was turned off and used 2-way SMP only