Parallel Computing

Bit-level parallelism

- 4-bit $\rightarrow$ 8-bit $\rightarrow \cdots \rightarrow$ 64-bit
- word-width increase will be necessary only for larger address space (not for performance increase)

Instruction-level parallelism

- pipelining (arrival of RISC)
- "wide-issue" **superscalar architecture** (fetch multiple instruction at a time and issue them in parallel to distinct function units whenever possible)
  - H/W responsible for parallelism
- **VLIW architecture**: compiler responsible for parallel scheduling
- *instruction-level parallelism worthwhile only if processor can be supplied with instructions and data fast enough to keep it busy*
  - need to avoid control/structural/data hazards
  - as a result, more sophisticated branch prediction
  - sophisticated caches to avoid the latency of cache misses
  - out-of-order instruction completion
  - larger window of instructions that are waiting to issue is maintained (issued as soon as a needed result is produced by a preceding instruction)
  - **all of these puts a heavy demand on chip resources and, also, very heavy design cost**

Thread-level parallelism

Process-level parallelism

Why parallel architecture

- performance of single processor machine is hitting the ceiling; need for a new paradigmic change
- Three types of parallel computers
  - **parallel vector processors (PVP)**
  - **massively parallel processors (MPP)**
  - **bus-based symmetric shared memory multiprocessors (SMP)**

Convergence of parallel computers

- **communication architecture**: communciation architecture (just like computer architectures) 1) defines communication/synchronization operations (i.e. H/W-S/W interface and user-system interface) and 2) defines an organizational structures that realize these operations
- different parallel programming models
  - **shared address**
  - **message passing**
  - **data parallel**

What makes parallel programming difficult

- **load balancing**: breaking into *equal-sized* subtasks
- **scheduling**: dependencies between subtasks
- **communication overhead**
- **time for synchronization**

Performance

- Performance(X) = 1/ExecutionTime(X)
- X is n-times faster than Y

$$\frac{\text{Performance(X)}}{\text{Performance(Y)}} = \frac{\text{ExecutionTime(Y)}}{\text{ExecutionTime(X)}} = n$$

Parallelsim and instructions: Synchronization

Parallelsim and computer arithmetic: Associativity

Parallelsim and advanced ILP

Parallelsim and memory hierarchies: Cache coherency

Parallelsim and I/O: Redundant arrays of inexpensive disks