Concurrent Programming

Session 3: Fundamentals

Computer Engineering Department
Iran University of Science and Technology
Tehran, Iran

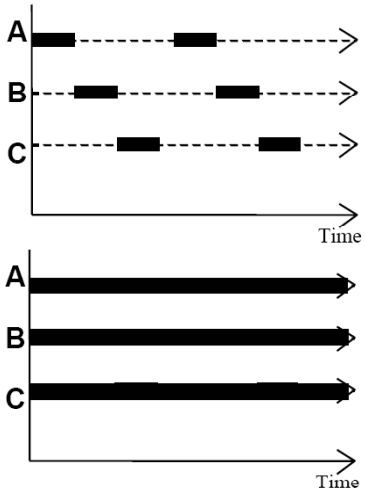
Lecturer: Nima Ghaemian
Distributed Systems Lab.
Computer Engineering Department,
Iran University of Science and Technology,
nima@comp.iust.ac.ir

What is Concurrency?

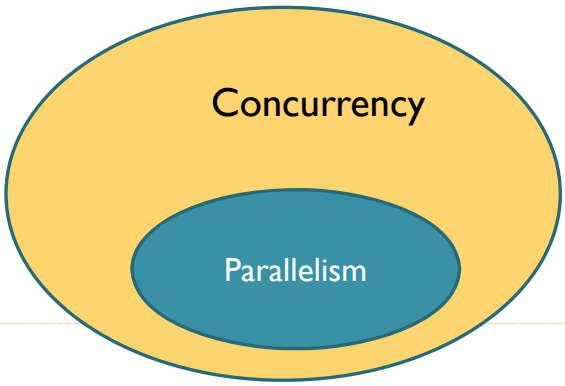
- What is a sequential program?
  - A single thread of control that executes one instruction and when it is finished execute the next logical instruction
- What is a concurrent program?
  - A collection of autonomous sequential threads, executing (logically) in parallel

Concurrency vs. Parallelism

- **Does Concurrency equal Parallelism?**
  - Concurrency is not (only) Parallelism
- **Interleaved Concurrency**
  - Logically simultaneous processing
  - Interleaved execution on a single processor
- **Simultaneous Concurrency**
  - Physically simultaneous processing
  - Requires a multiprocessors or a multicore system



So…



- Note that from another point of view, concurrency can be divided into full and pseudo parallelism.

Why Concurrency?

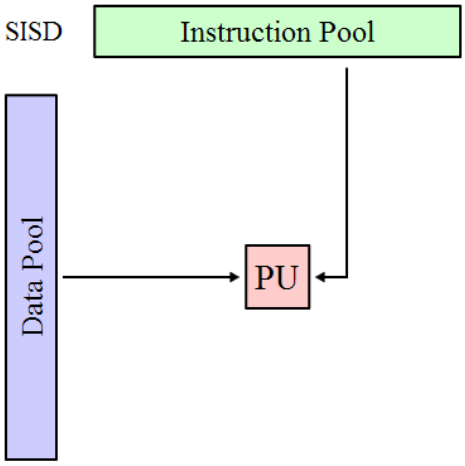
- No More Clock Cycles!
- Natural Application Structure
  - The world is not sequential! Easier to program multiple independent and concurrent activities
- Increased application throughput and responsiveness
  - Not blocking the entire application due to blocking I/O
- Performance from multiprocessor/multicore hardware
  - Parallel Execution
- Distributed Systems
  - Single application on multiple machines
  - Client/Server type or peer-to-peer systems

Flynn's Taxonomy

- Classification of Computer Architectures
  - SISD
    - Single Instruction Single Data stream
  - SIMD
    - Single Instruction Multiple Data streams
  - MISD
    - Multiple Instructions Single Data stream
  - MIMD
    - Multiple Instructions Multiple Data streams

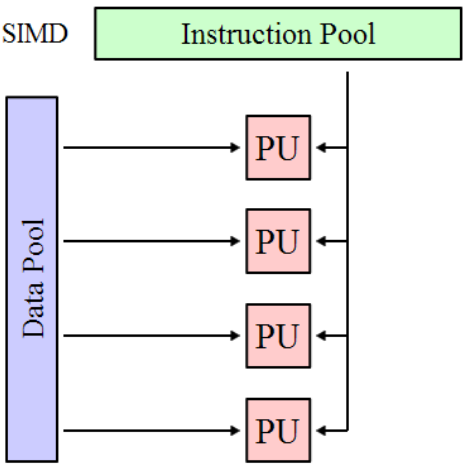
SISD

- Single Instruction Single Data stream
- Sequential Computer
- Traditional Uniprocessor Systems
  - PC
  - Old Mainframes



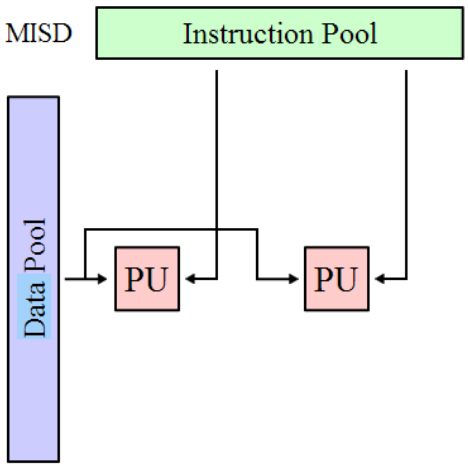
SIMD

- Single Instruction Multiple Data streams
- Intel's Streaming SIMD Extensions (SSE)
- Array Processors
- Stream Processors inside a GPU



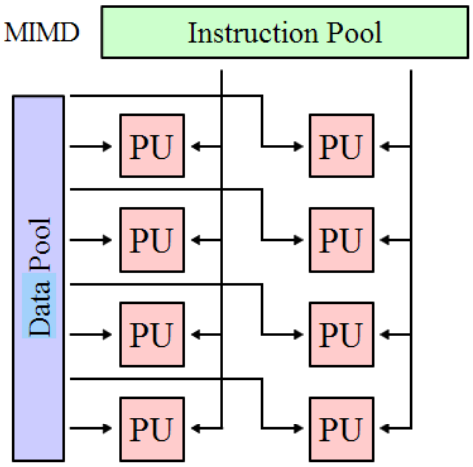
MISD

- Multiple Instructions Single Data stream
- Is not common
  - It was even thought to be N/A!!
- Where is it applicable?
  - Fault Tolerance!
  - Exploratory Decomposition
    - Decomposition?
  - An Implementation:
    - Space Shuttle Flight Control



MIMD

- Multiple Instructions Multiple Data streams
  - Multiprocessors
  - Distributed Systems
- The Most Popular as top 10 Architectures in TOP500 (from '06)
  - TOP500?
- Further Divisions…
  - SPMD
    - A Grid in GPU
  - MPMD
    - Manager/Worker Strategy



Classification of Parallel Computers

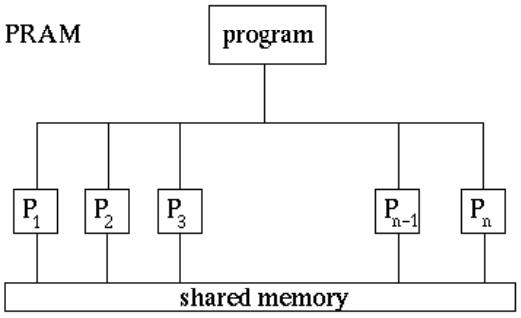
- Multicore Computing
- Symmetric Multiprocessing
- Distributed Computing
  - Cluster Computing
  - Massive Parallel Processing
  - Grid Computing
- Specialized Parallel Computers
  - Reconfigurable computing with FPGA
  - GPGPUs
  - Application-Specific Integrated Circuits
  - Vector Processors

Any Abstract Model?

- PRAM
  - Parallel Random Access Machine
- Eliminates the focus on miscellaneous issues
  - Synchronization
    - An Important Factor! We'll talk about it later!
  - Communication
    - Von Neumann Bottleneck
- Lets the designer think explicitly about the exploitation of concurrency

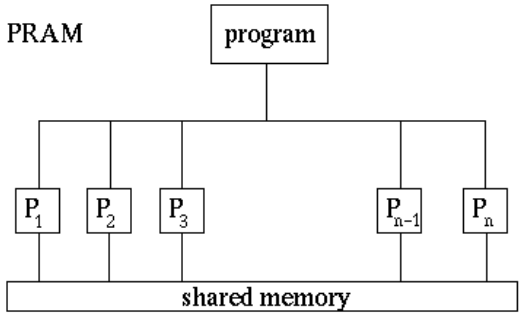
Any Abstract Model? (Cont'd)

- PRAM in Flynn's Taxonomy?
  - MIMD
- Programming Model
  - A New Structure
    - **for** *i* **pardo** *Pi*



Any Abstract Model? (Cont'd)

```
procedure Sort(modifies A: array 1..n of integer)
for i in 1..n pardo K[i]:=0
for i in 1..n pardo
  for j in 1..n pardo
    if A[i]<=A[j] then K[j]:=K[j]+1
for i in 1..n pardo A[K[i]]:=A[i]
```



How to Model More Efficiently?

- *"To write a parallel program in a machine-independent fashion requires the development of abstract models for parallel computations without sacrificing the implementability of these programs"*
- Communicating Sequential Process (CSP)
- Calculus of Communicating Systems (CCS)
- Petri Nets

Levels of Concurrency

- Levels of Parallelism
  - Task Level Parallelism
    - Focuses on distributing execution processes (threads) across different parallel computing nodes
  - Data Level Parallelism
    - distributing the data across different parallel computing nodes
  - Instruction Level Parallelism
    - fetch, decode, execute
  - Bit Level Parallelism
    - How hardware treats words

Levels of Concurrency (Cont'd)

- Granularity
  - Ratio of the compute time to communication overhead
  - Ratio of the synchronizations
- Levels of Concurrency in terms of Granularity
  - Very Coarse-Grained
  - Coarse-Grained
  - Medium-Grained
  - Fine-Grained

Models of Concurrent Programming

- Attributes of Concurrency Models
  - Level of granularity (level of atomicity)
    - Discussed before..
  - Sharing the clock
  - Sharing the memory
  - Pattern of interaction
    - Synchronization
    - Communication
  - Pattern of synchronization
    - Mutual Exclusion
    - Mutual Admission

Models of Concurrent Programming

- Attributes of Concurrency Models (Cont'd)
  - Pattern of Communication
    - Synchronous
    - Asynchronous
  - Specifying Concurrency
    - Application Concurrency
    - Implementation Concurrency
  - Implementation of Concurrency
    - Effective Concurrency
    - Simulated Concurrency
  - Interaction Protocol
    - Cooperative
    - Competitive

Development of Parallel Languages

- Sequential Language Extensions
  - UPC (Unified Parallel C)
  - CUDA C
  - Brook (Stanford's Merrimac)
  - Cilk++
  - HPF (High Performance Fortran)
- Libraries
  - MPI
- Parallelizing Compilers
  - Automatic
  - Directed (using OpenMP)
- New Parallel Languages
- Oz