High Performance Computing
Lecture 38

Matthew Jacob

Indian Institute of Science

Agenda

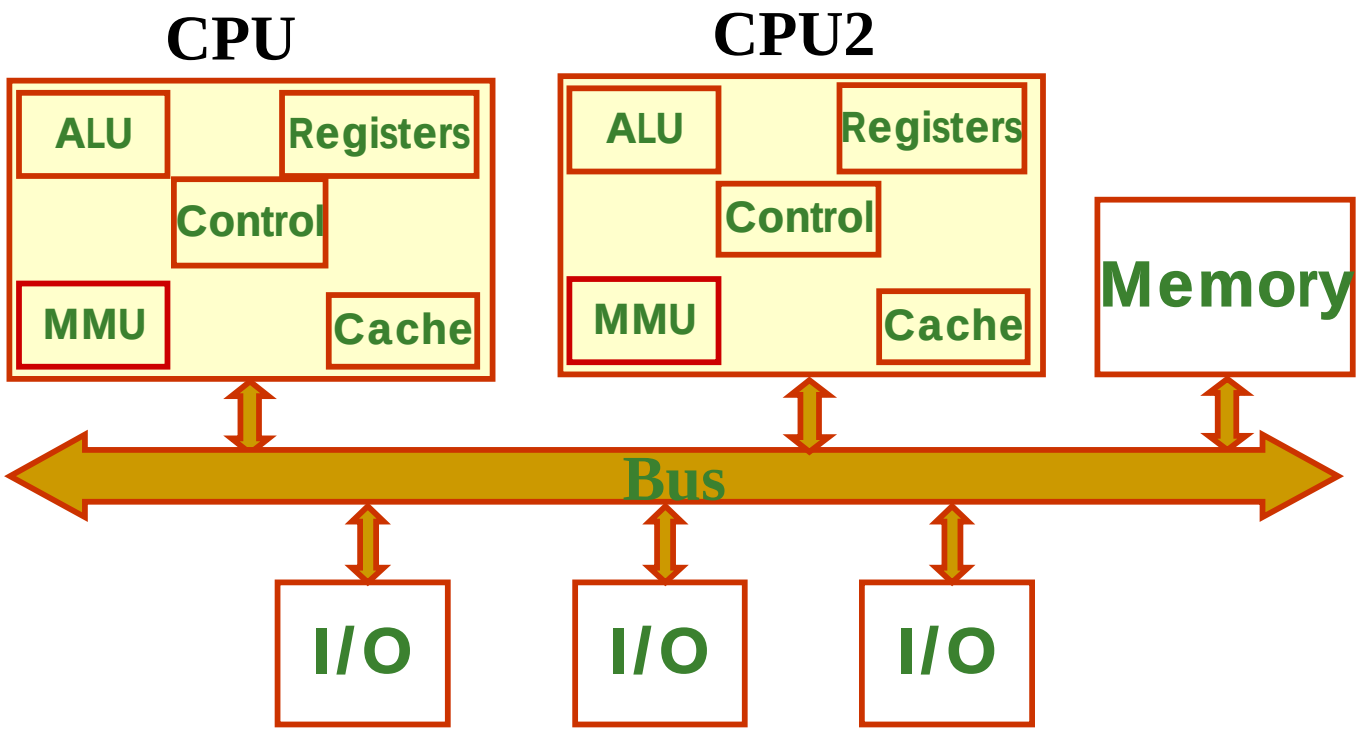
1. Program execution: Compilation, Object files, Function call and return, Address space, Data & its representation (4)
2. Computer organization: Memory, Registers, Instruction set architecture, Instruction processing (6)
3. Virtual memory: Address translation, Paging (4)
4. Operating system: Processes, System calls, Process management (6)
5. Pipelined processors: Structural, data and control hazards, impact on programming (4)
6. Cache memory: Organization, impact on programming (5)
7. Program profiling (2)
8. File systems: Disk management, Name management, Protection (4)
9. Parallel programming: Inter-process communication, Synchronization, Mutual exclusion, Parallel architecture, Programming with message passing using MPI (5)

Parallel Computing

- Parallel Architecture
- Introduction to Parallel Programming

Parallel Architecture

- Parallel computer: A computer system with more than one processor



Parallel Architecture

Question: Is a network of computers a parallel computer?

- Yes, but the time involved in interaction (communication) might be high, as the system is designed assuming that the machines are more or less independent
- Special parallel machines would be designed to make this interaction overhead less

Parallel Architecture

- Parallel computer: A computer system with more than one processor
- Question: Why use parallel computers?
  1. With two processors, twice as many instructions can be executed in a given amount of time
     - i.e., improved performance, reduced program execution time
  2. The computer can continue to operate even if one of the processors fails
     - i.e., fault tolerance

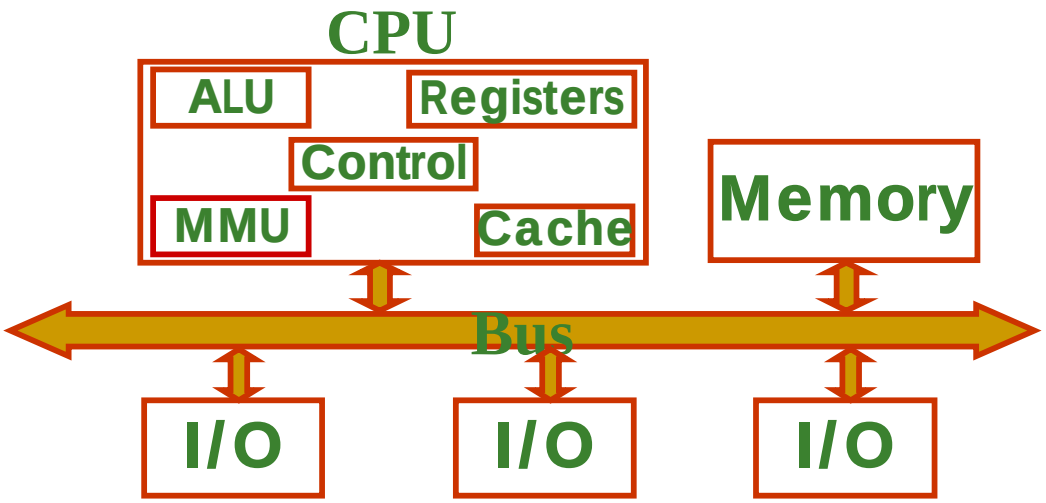
Classification of Parallel Computers

Flynn's Classification

- In terms of the number of Instruction streams and Data streams
- Instruction stream: A path to instruction memory (i.e., a program counter or PC)
- Data stream: A path to data memory
  - SISD: single instruction stream single data stream
  - SIMD: single instruction stream multiple data streams
  - MIMD: multiple instruction stream multiple data streams

Flynn's Classification: SISD

- Single Instruction Stream Single Data Stream
  - i.e., one program counter and one path to data memory
  - i.e., a computer capable of executing one instruction at a time operating on one piece of data
  - i.e., an ordinary (sequential) computer

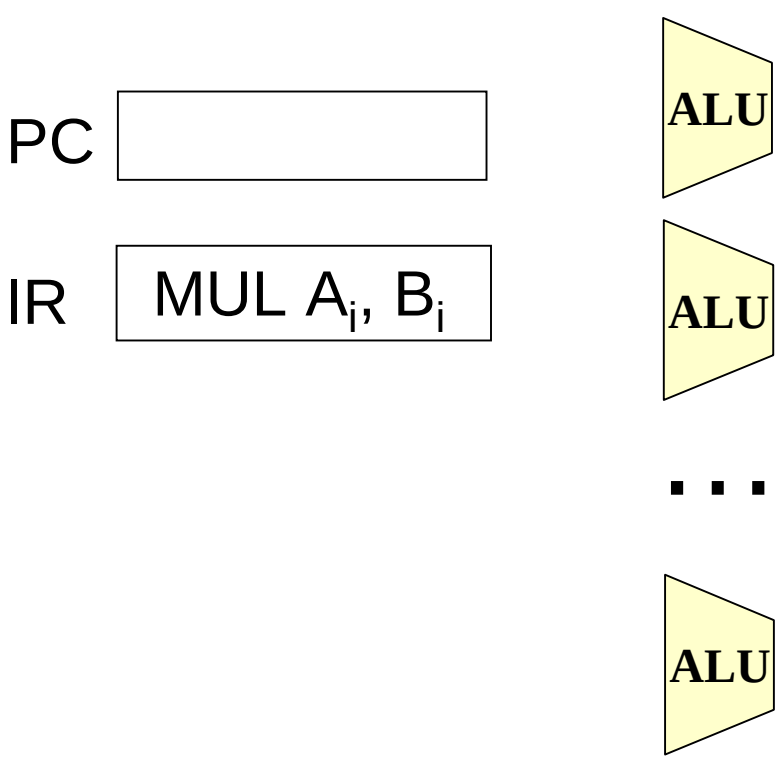


Flynn's Classification: SIMD

- Single Instruction Stream Multiple Data Streams
  - i.e., one program counter, multiple paths to data memory
  - i.e., a computer capable of executing one instruction at a time, but operating on different pieces of data

Flynn's Classification: SIMD

- Example: A computer with 1024 ALUs (each with a separate data path to memory), but only one program counter (PC and IR)



The same MUL instruction is executed on each of the ALUs, but on different pieces of data

Flynn's Classification: MIMD

- Multiple Instruction Stream Multiple Data Stream
  - i.e., a computer that can run multiple processes or threads that are cooperating towards a common objective
    - in parallel, not just concurrently
  - Alternatively, the MIMD computer could run multiple independent programs at the same time

Another Classification: Shared Memory vs Message Passing

- Shared memory machine: The n processors share physical address space
  - Communication can be done through this shared memory

- The alternative is sometimes referred to as a message passing machine or a distributed memory machine

Shared Memory Machines

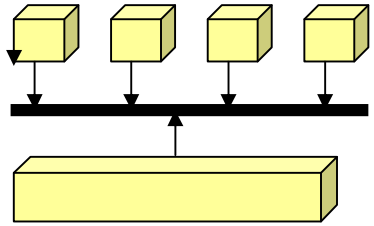
The shared memory could itself be distributed among the processor nodes

- Each processor might have some portion of the shared physical address space that is physically close to it and therefore accessible in less time

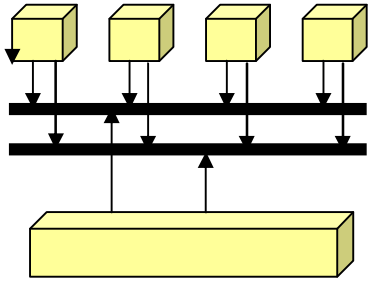
Parallel Architecture: Interconnections

- Indirect interconnects: nodes are connected to interconnection medium, not directly to each other
  - Shared bus, multiple bus, crossbar, MIN
- Direct interconnects: nodes are connected directly to each other
  - Topology: linear, ring, star, mesh, torus, hypercube
  - Routing techniques: how the route taken by the message from source to destination is decided

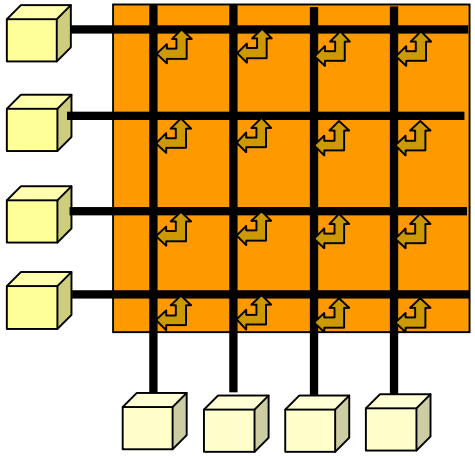
Indirect Interconnects



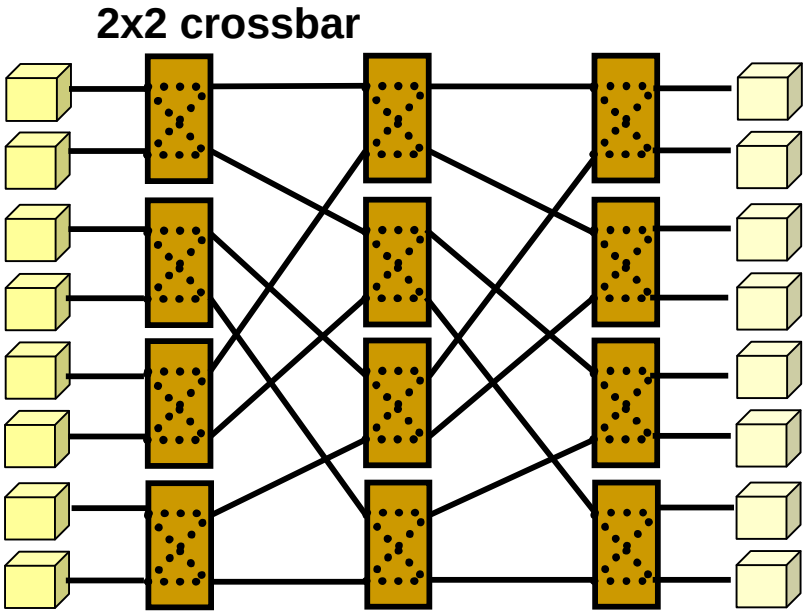
Shared bus



Multiple bus

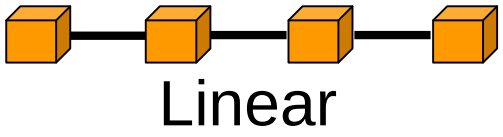


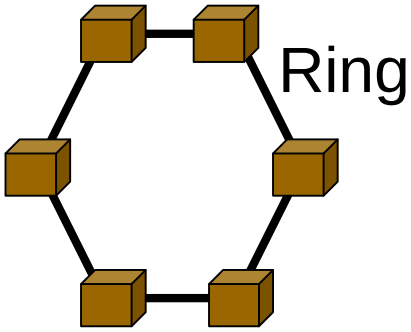
Crossbar switch

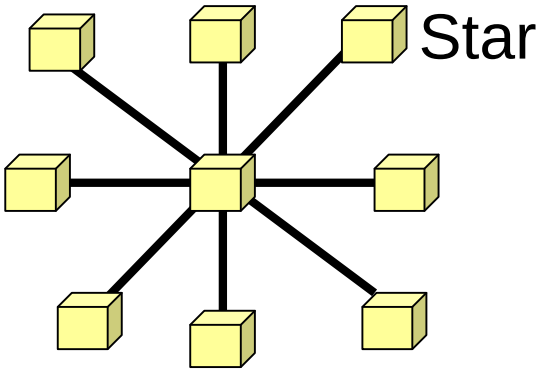


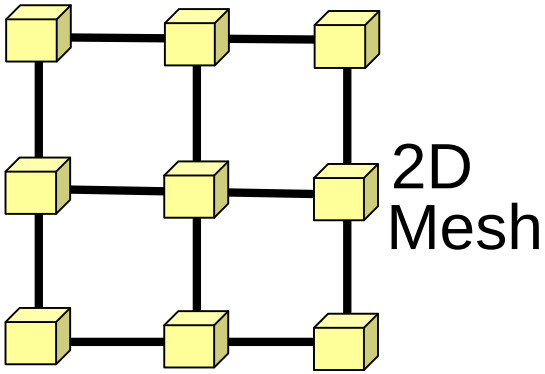
Multistage Interconnection Network

Direct Interconnect Topologies

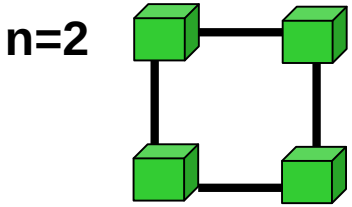


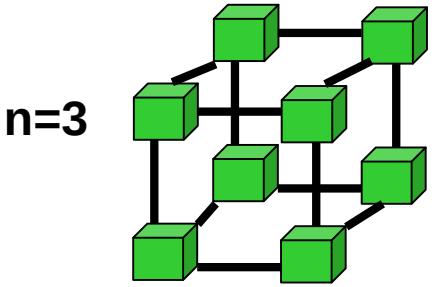


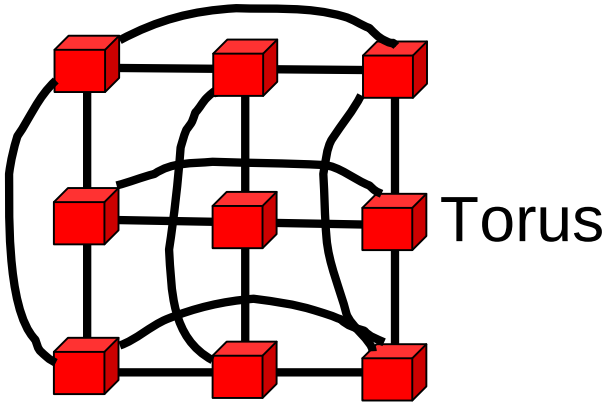




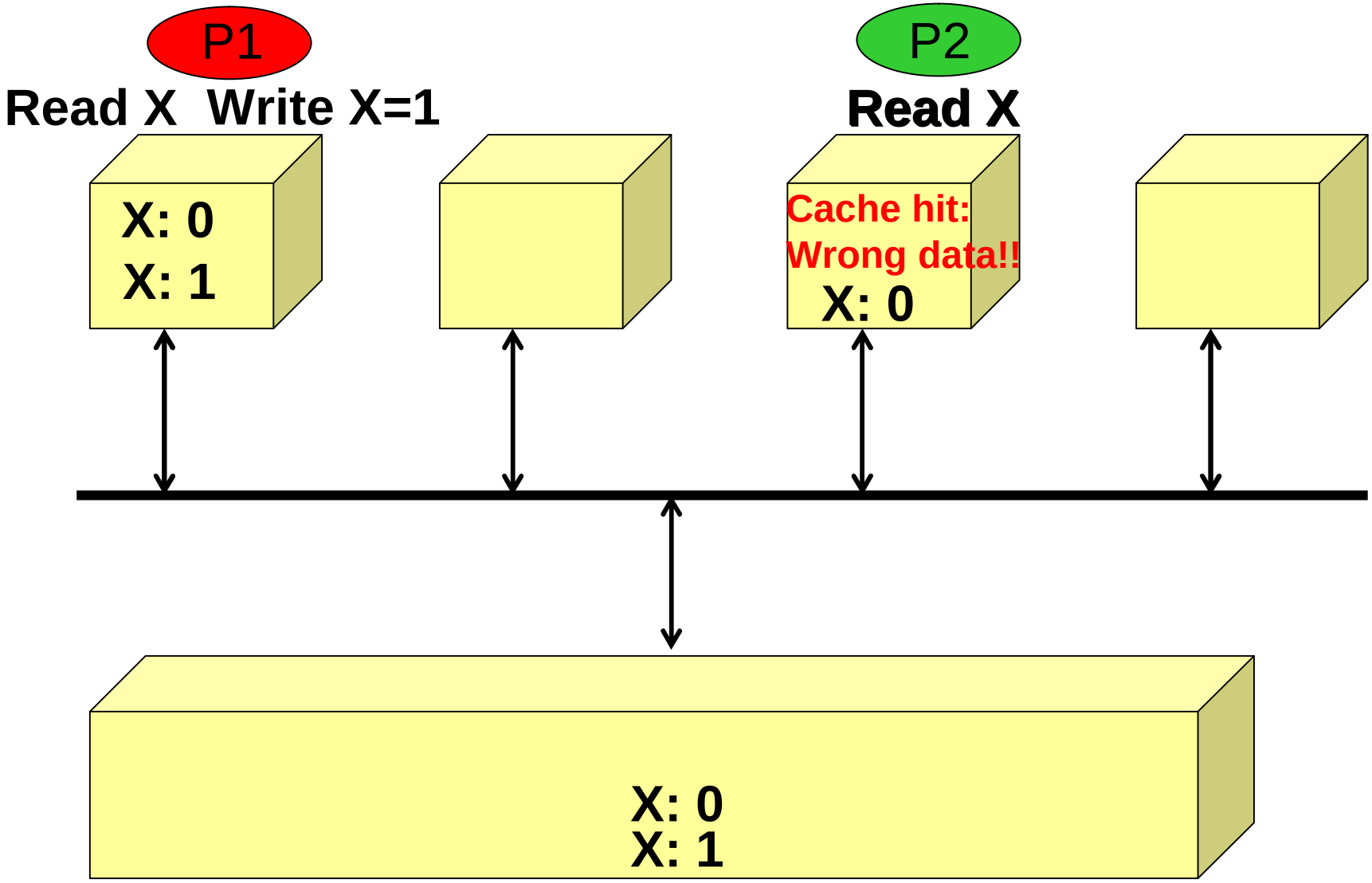
Hypercube (binary n-cube)







Shared Memory Architecture: Caches



Cache Coherence Problem

- If each processor in a shared memory multiple processor machine has a data cache
  - Potential data consistency problem: the cache coherence problem
  - Shared variable modification, private cache
- Objective: processes shouldn't read `stale' data
- Solutions
  - Hardware: cache coherence mechanisms
  - Software: compiler assisted cache coherence

Example: Write Once Protocol

- Assumption: shared bus interconnect where all cache controllers monitor all bus activity
  - Called snooping
- There is only one operation through bus at a time; cache controllers can be built to take corrective action and enforce coherence in caches
  - Corrective action could involve updating or invalidating a cache block

Example: Write Once Protocol.