

CPS 104
Computer Organization and Programming
Lecture-25: Superscalar processors,
Multiprocessors

Dec. 6, 1999

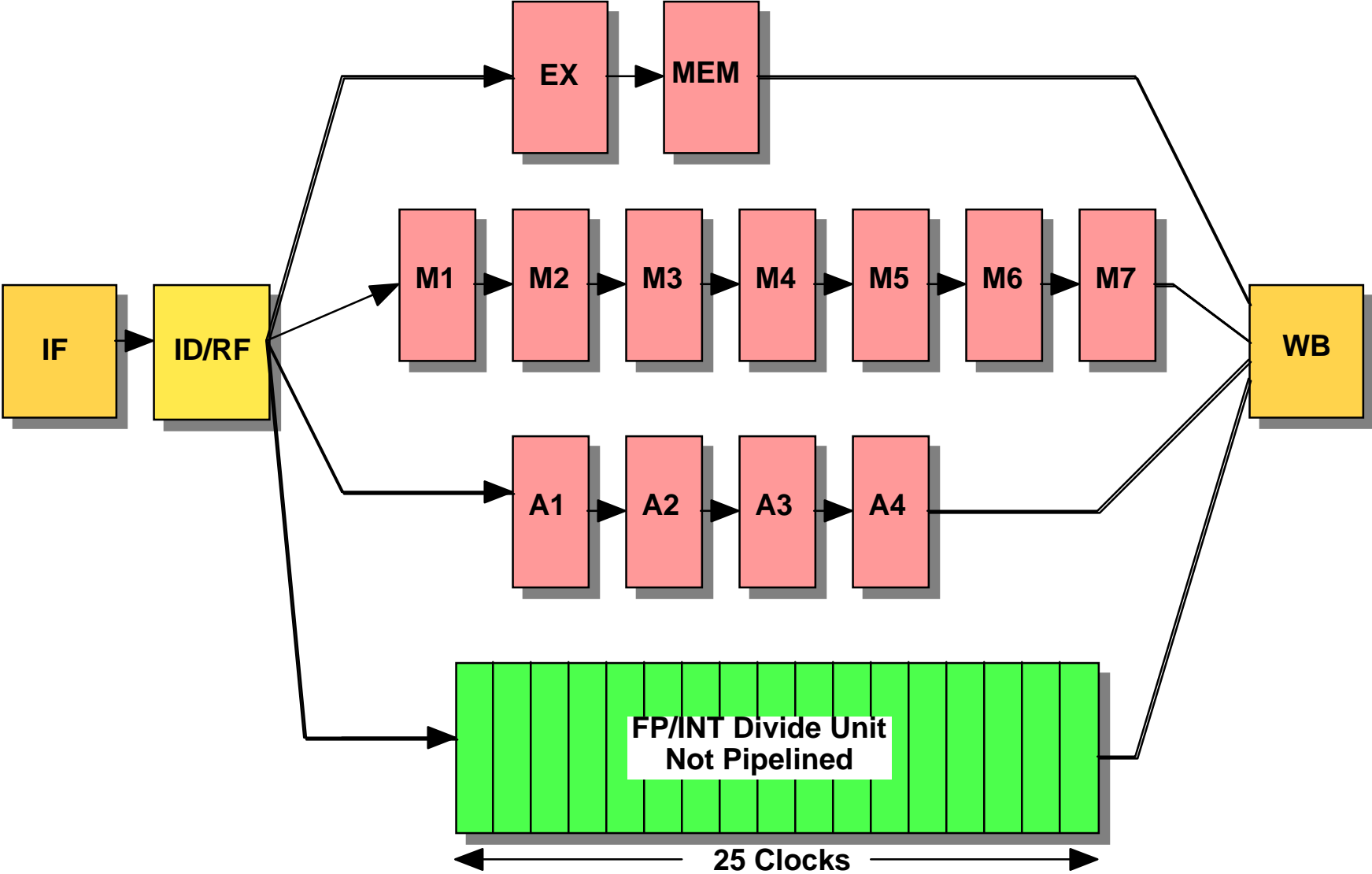
Dietolf (Dee) Ramm

<http://www.cs.duke.edu/~dr/cps104.html>

Faster and faster Processors

- So much to do, so little time. . .
- How can we make computers that execute faster?
 - ◆ Faster clock => more instructions/second. (technology constraints)
 - ◆ Pipelining: => faster clock
 - ◆ Execute more than 1 instruction per cycle, (**Superscalar processor**)
 - ◆ Use multiple processors and divide the computation (**Multiprocessors, Clustered computing, Distributed computing**)

Multiple Pipelines : Floating Point



(CPI < 1): Superscalar Design

- Pipelining can get CPI=1 and fast clock. Can we do better?
- Superscalar design: **Execute multiple instructions every clock.**
- Problems for Superscalar Design:
 - ◆ Need multiple execution units (pipelines),
 - ◆ Structural Hazards:
 - Need multiple accesses to register files.
 - Might need multiple accesses to caches
 - ◆ Data Hazards:
 - How to deal with data dependencies (keep program semantics)?
 - What to do with stalled instructions?
 - ◆ Control Hazards:
 - What to do about conditional branches?

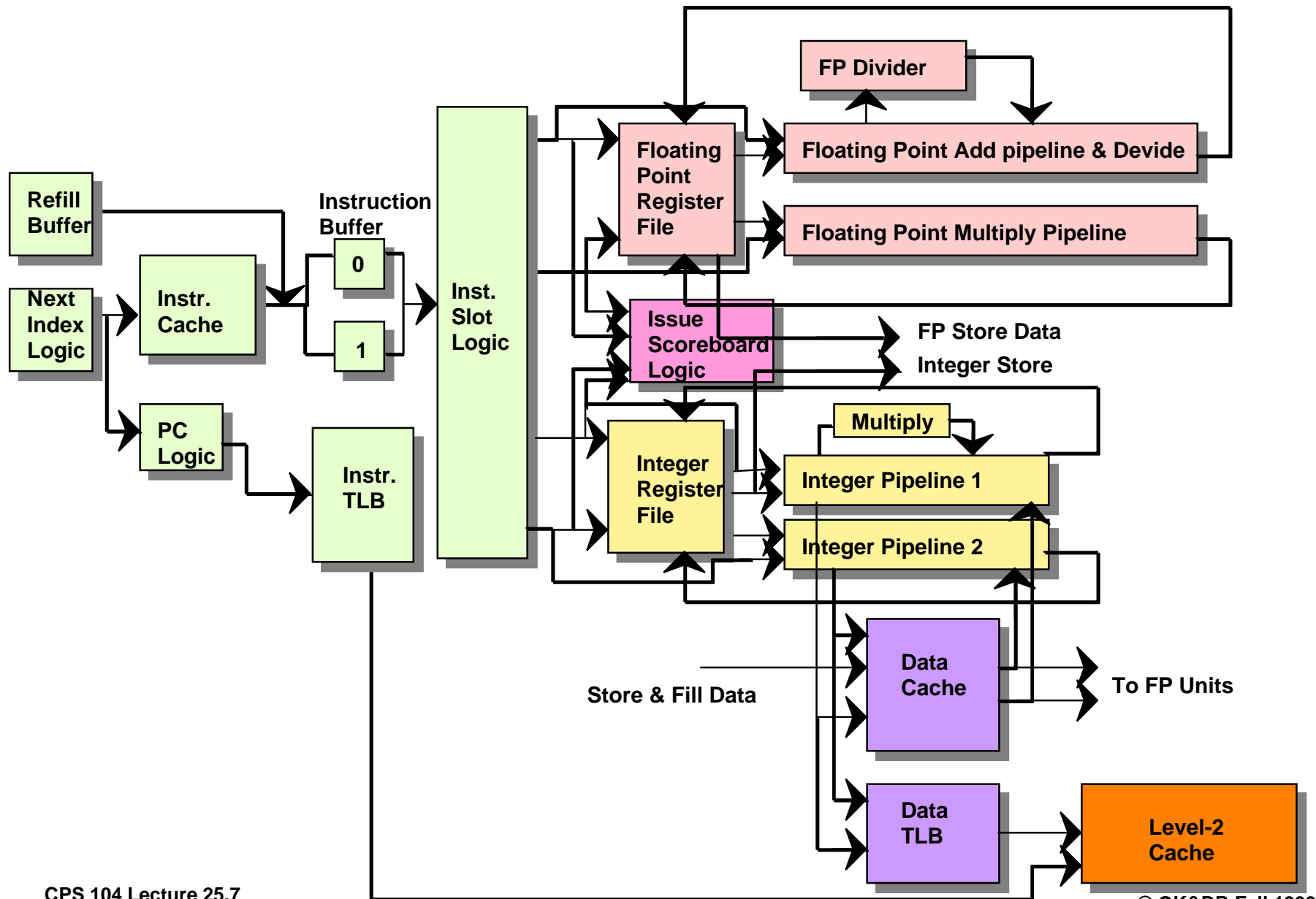
Superscalar Design Solutions

- Multiple pipelines are not a problem. We already had them in “regular” pipeline design.
- Structural Hazards: Build register files with many read and write ports: Ex; 7-read and 3-write ports. Build multi-port caches.
- Data Hazards solutions:
 - ◆ Issue instructions **in order**. Use score-board to eliminate data hazards by stalling instructions.
 - ◆ “Better Solution”: Issue instruction **out of order**, Use register renaming to avoid data hazards, Graduate instructions in order.
- Control Hazards solutions:
 - ◆ Use Branch Prediction:
 - ◆ Make sure that the branch is resolved before registers are modified.
 - ◆ OR, Use **speculative execution**, roll back results if branches were predicted wrong.

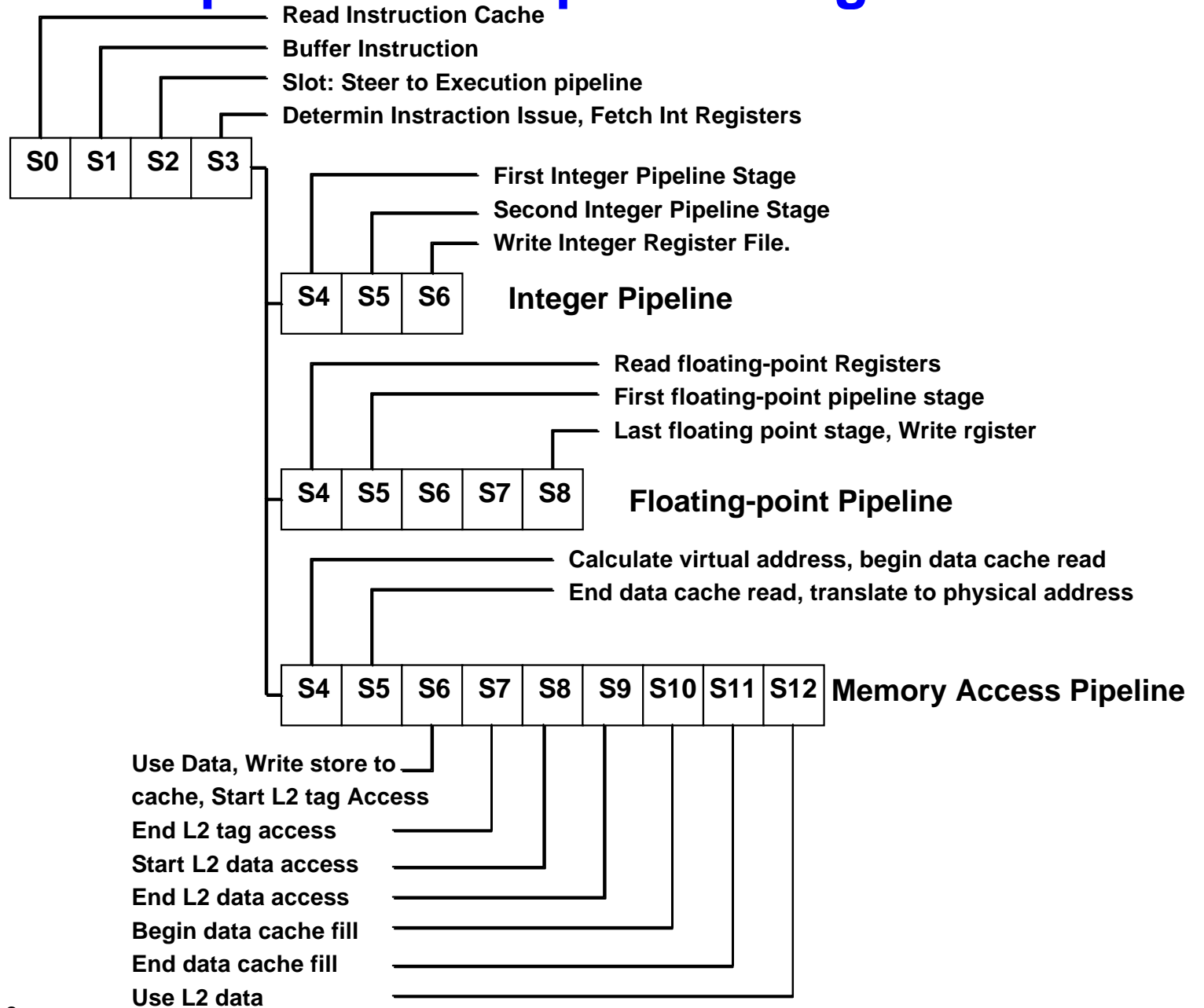
The Alpha 21164 Superscalar

- Can issue up to four instructions per clock cycle
- Deep pipeline: 7 stage integer, 9-stage floating point, up to 13 stages for on-chip load/store.
- There are two Integer and two Floating-point pipelines.
- **In order issue. In-order execution.**
- Use score-board to stall instructions with conflicts.
- Use score-board to compute all register forwarding operations.
- Integer Register File has 4 read ports and two write ports.
- Floating point Register File has 6 read ports and 3 write ports.
- Use Branch Prediction to keep the pipe full.

The Alpha 21164 Superscalar Pipeline



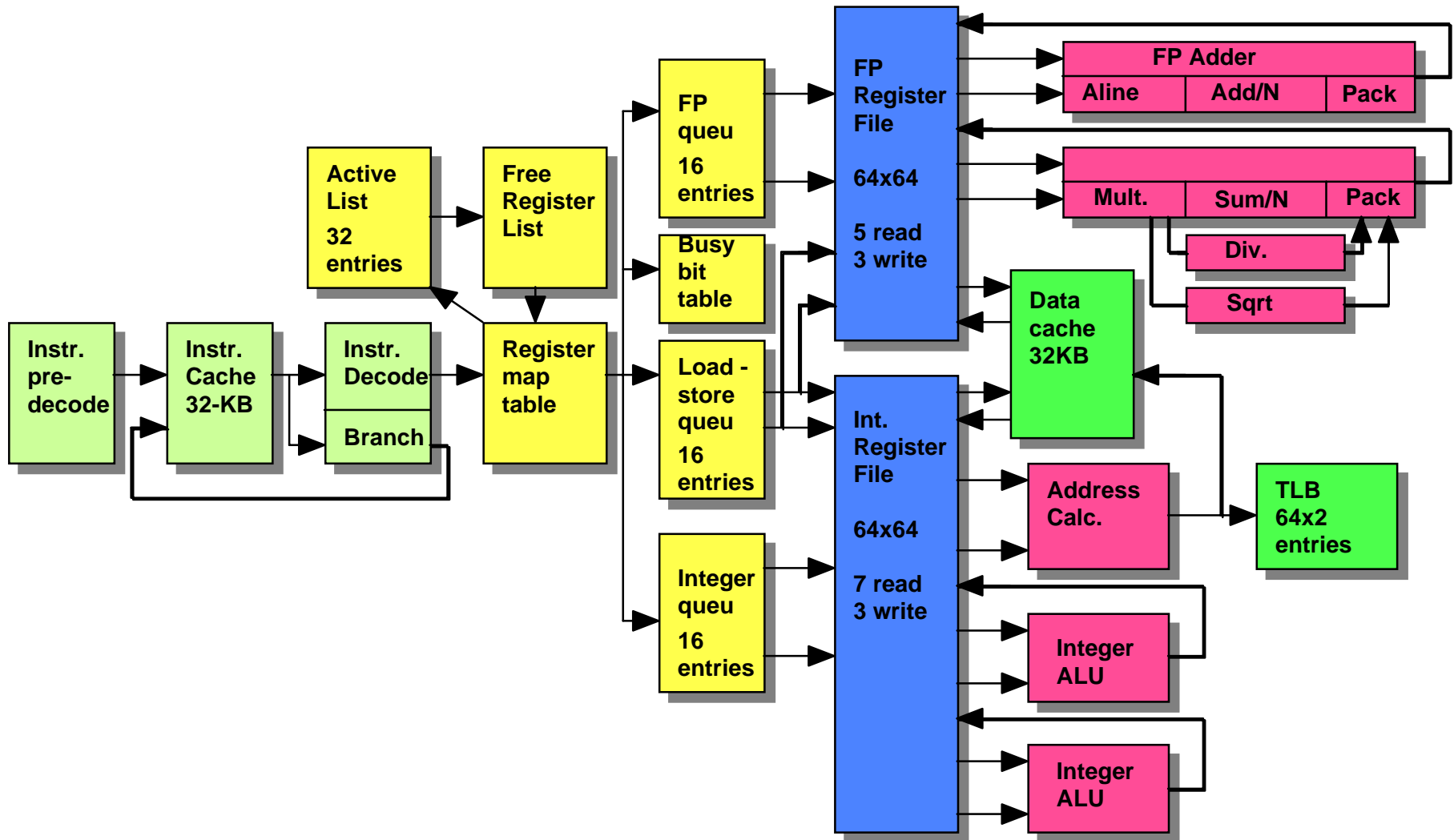
Alpha 21164 Pipeline Stages



MIPS R 10000 Superscalar

- Issues 4 instructions at a time
- Has 5 execution units: 2 FP units, 2 Integer units and load/store unit.
- Out of order execution
- Speculative execution predicts up to four branches at a time.

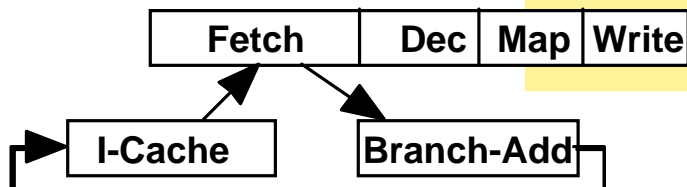
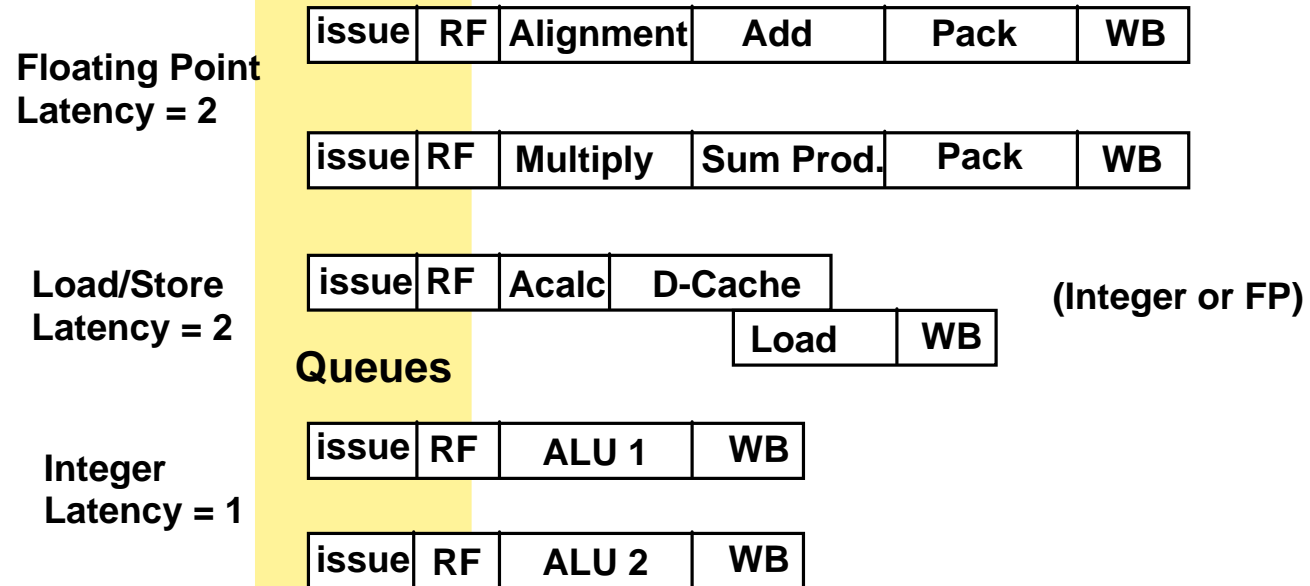
MIPS 10000 CPU



MIPS 10000 Pipelines

I-Fetch I-Decode I-Issue

Stage 1	Stage 2	Stage 3	Stage 4	Stage 5	Stage 6	Stage 7
---------	---------	---------	---------	---------	---------	---------



Instruction fetch and decode pipeline. Fills queues 4 instructions in parallel.

Up to 4 branch instructions are predicted
Speculative fetching.

What is Parallel Computer Architecture?

- **A Parallel Computer is a collection of processing elements that cooperate to solve large problems fast**
 - ◆ how large a collection?
 - ◆ how powerful are the elements?
 - ◆ how does it scale up?
 - ◆ how do they cooperate and communicate?
 - ◆ how is data transmitted between processors?
 - ◆ what are the primitive abstractions?
 - ◆ how does it all translate to performance?

Parallel Computation: Why and Why Not?

● Pros

- ◆ Performance
- ◆ Cost-effectiveness (commodity parts)
- ◆ Smooth upgrade path
- ◆ Fault Tolerance

● Cons

- ◆ Difficult to parallelize applications
- ◆ Requires automatic parallelization or parallel program development
- ◆ **Software! AAHHHH!**

Applications: Science and Engineering

- **Examples**

- ◆ Weather prediction
- ◆ Evolution of galaxies
- ◆ Oil reservoir simulation
- ◆ Automobile crash tests
- ◆ Drug development
- ◆ VLSI CAD
- ◆ Nuclear BOMBS!

- **Typically model physical systems or phenomena**
- **Problems are 2D or 3D**
- **Usually requires “number crunching”**
- **Involves “true” parallelism**

Applications: Commercial

- **Examples**
 - ◆ Transaction processing
 - ◆ Database
 - ◆ Financial models
- **Involves data movement, not much number crunching**
- **Involves throughput parallelism**

Applications: Multi-media/home

- **Examples**
 - ◆ speech recognition
 - ◆ data compression/decompression
 - ◆ 3D graphics
- **Will become ubiquitous**
- **Involves everything (crunching, data movement, true parallelism, and throughput parallelism)**

Flynn Taxonomy

- **SISD**

- ◆ Single Instruction Single Data
- ◆ Early Data parallel machines

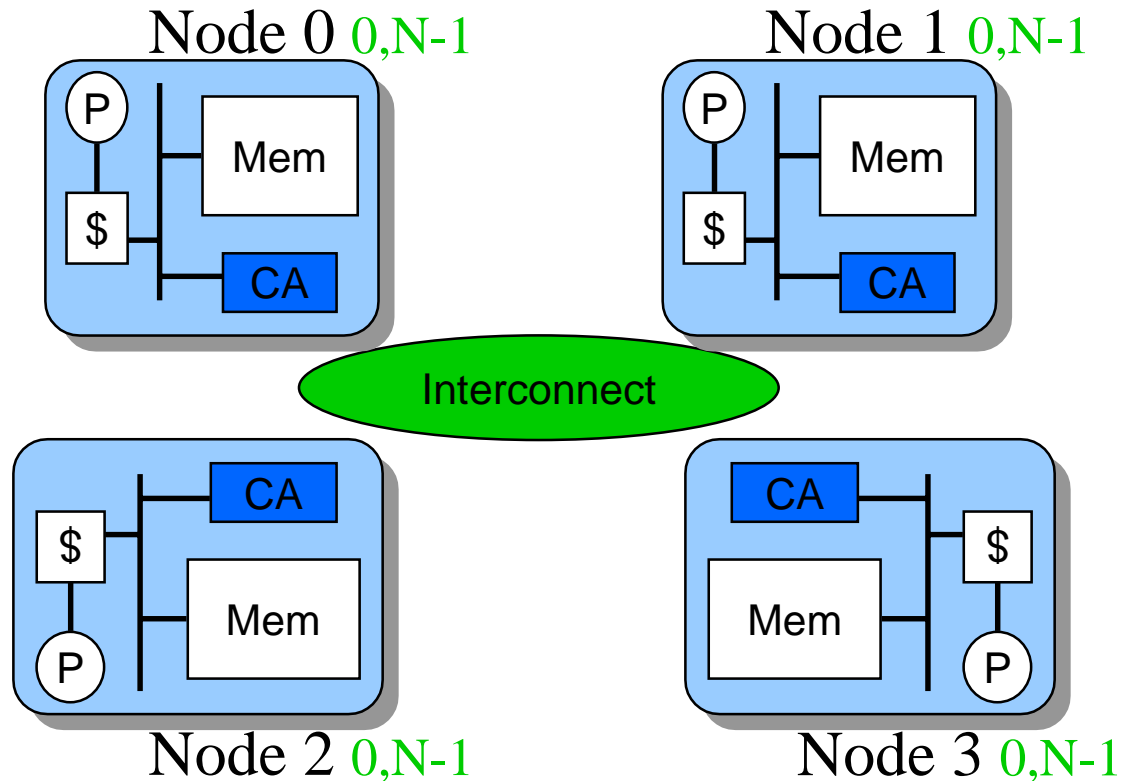
- **SIMD**

- ◆ Single Instruction Multiple Data
- ◆ on chip today, multimedia
- ◆ Special applications (graphics, image processing, cryptography)

- **MIMD**

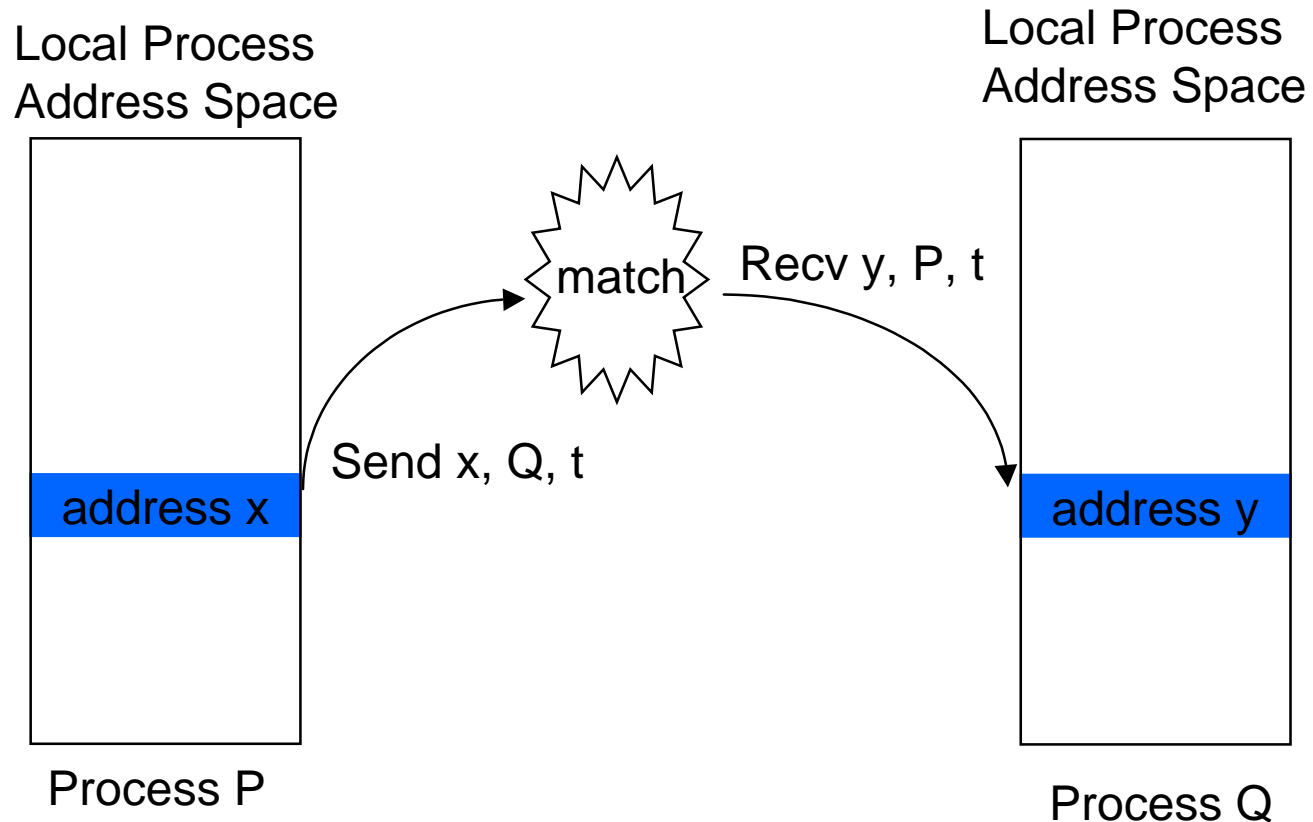
- ◆ Multiple Instruction Multiple Data
- ◆ most of today's parallel machines

Message Passing Architectures



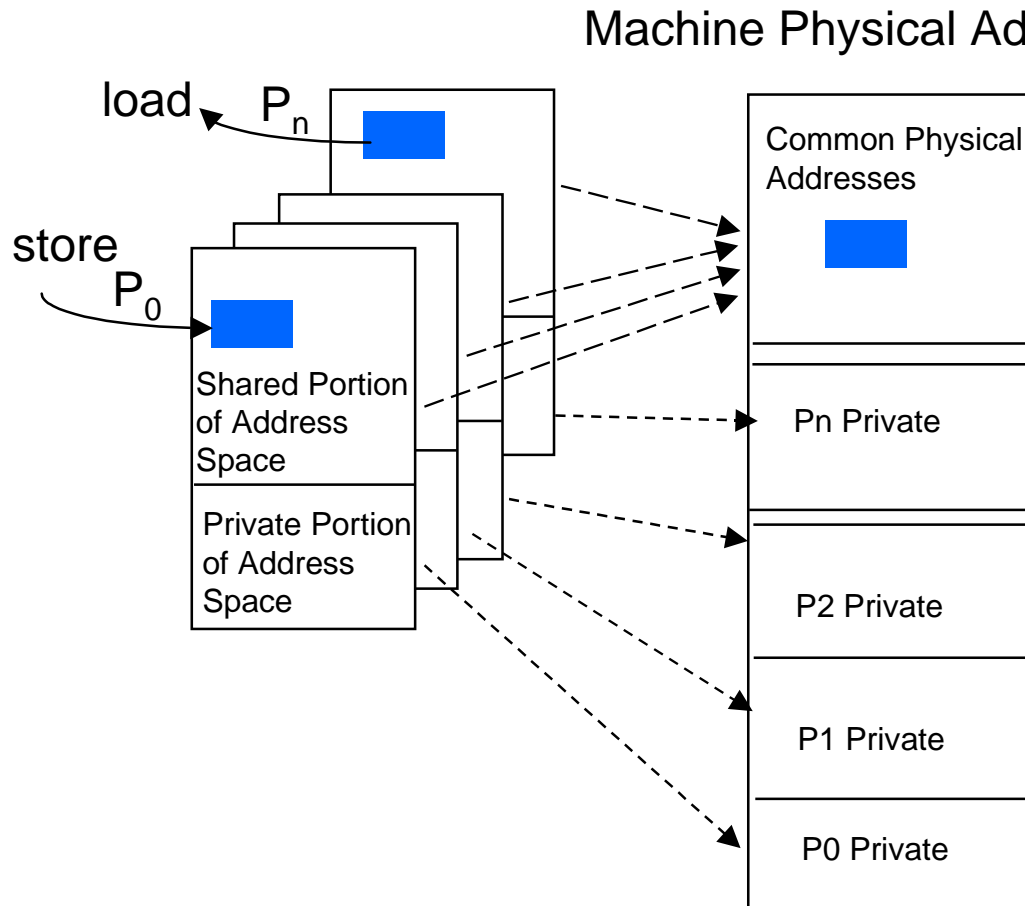
- **Cannot directly access memory on another node**
- **IBM SP-2, Intel Paragon**
- **Cluster of workstations**

Message Passing Programming Model



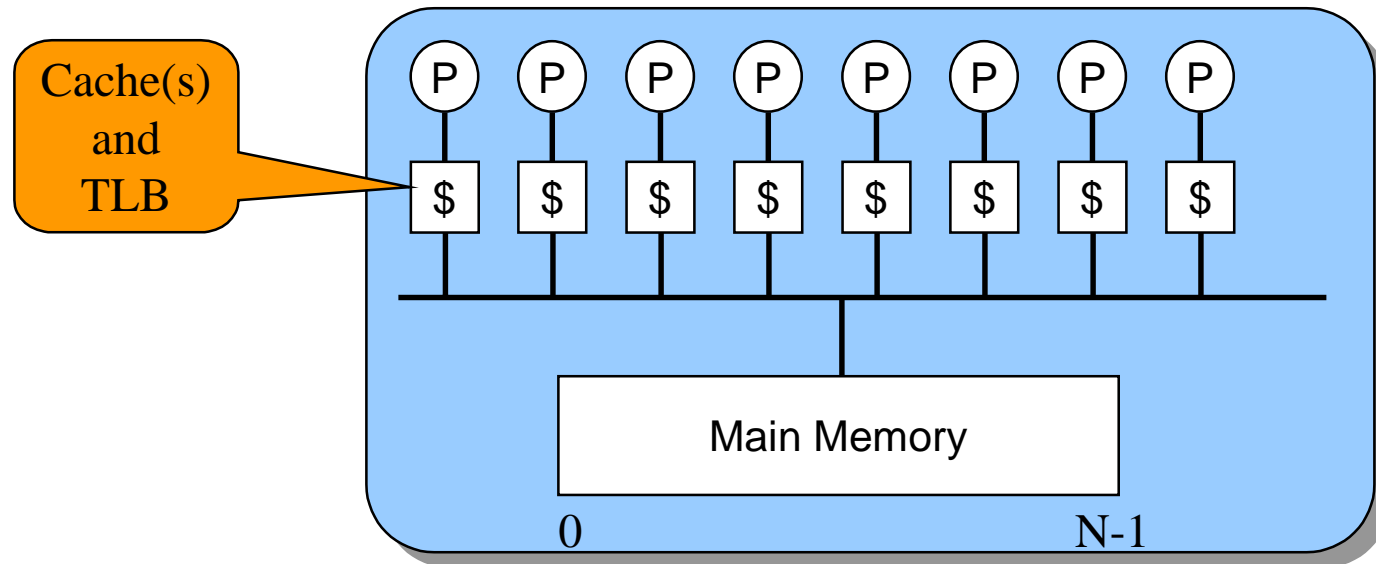
- **User level send/receive abstraction**
 - ◆ local buffer (x,y), process (Q,P) and tag (t)
 - ◆ naming and synchronization

Single Shared Address Space



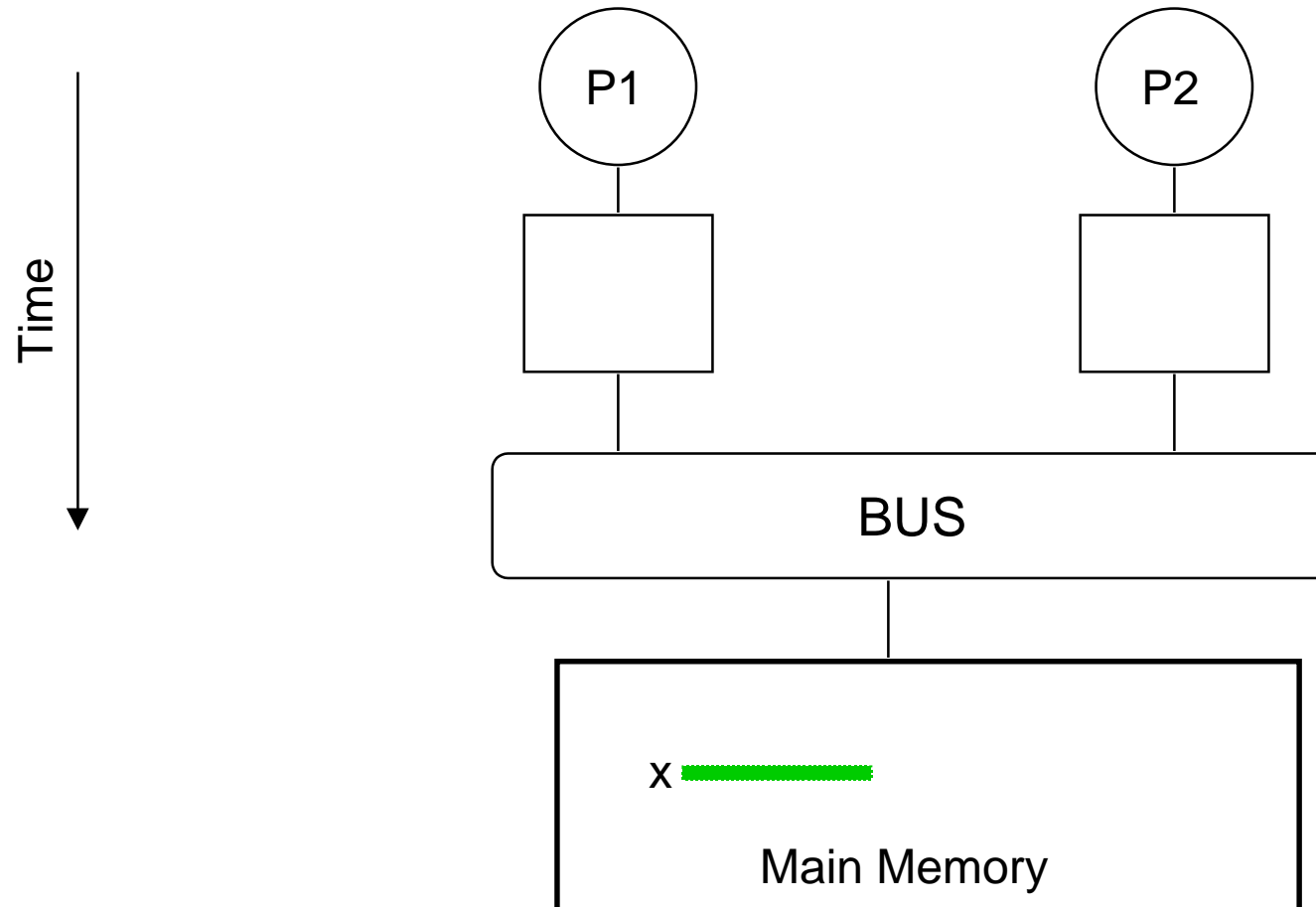
- **Communication, sharing, and synchronization with store / load on shared variables**
- **Must map virtual pages to physical page frames**
- **Consider OS support for good mapping**

Small Scale Shared Memory Multiprocessors

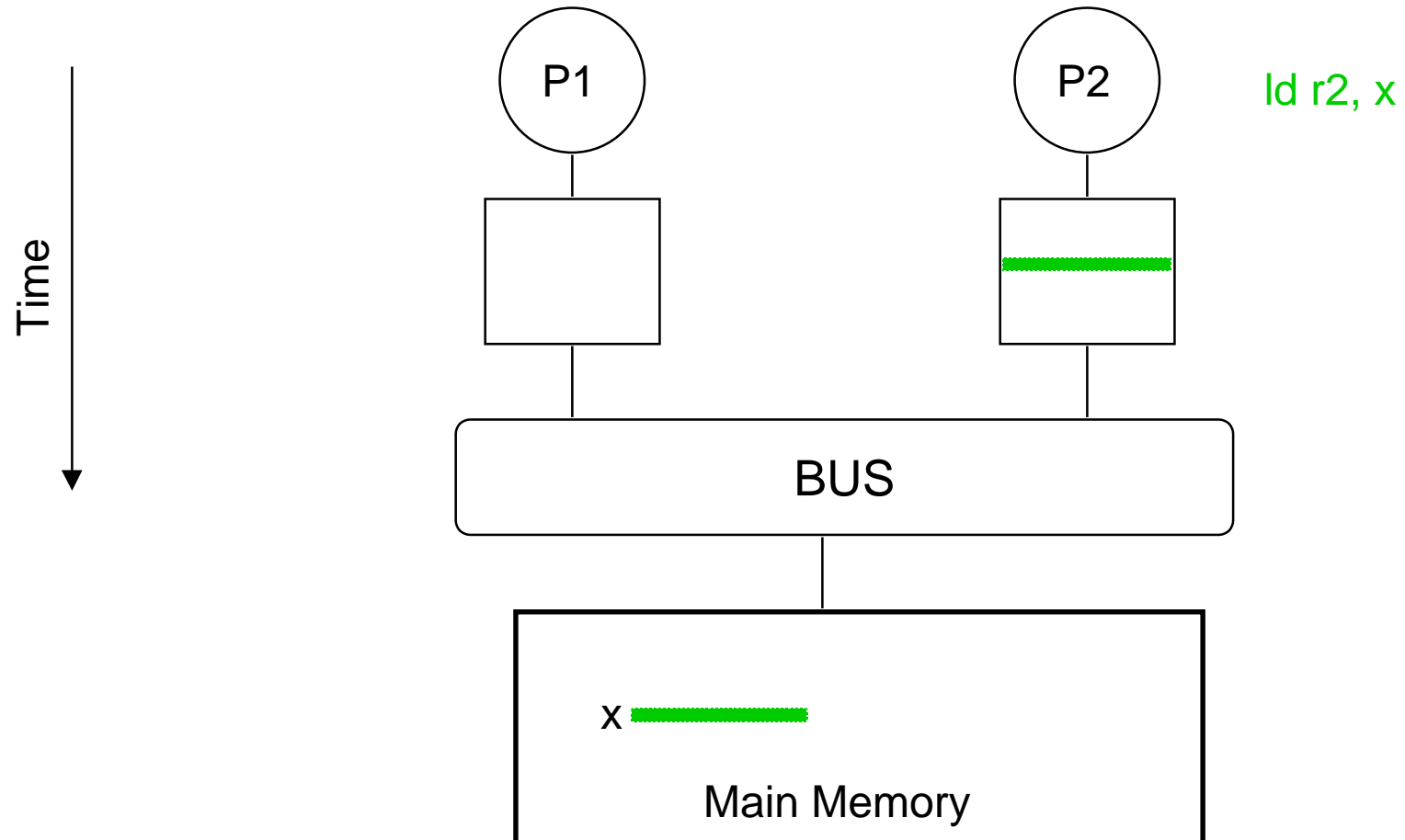


- **Small number of processors connected to one shared memory**
- **Memory is equidistant from all processors (UMA)**
- **Kernel can run on any processor (symmetric MP)**

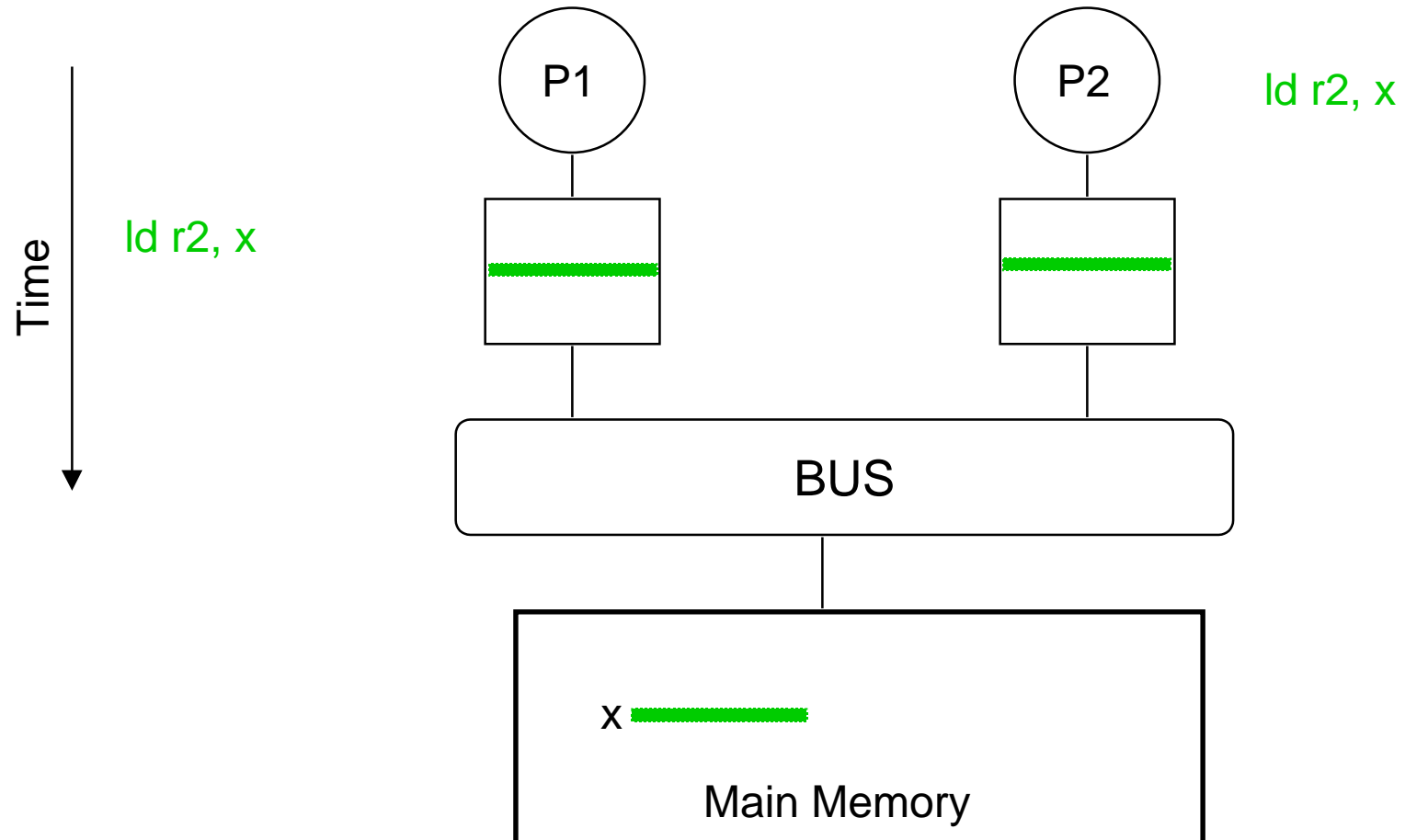
Cache Coherence Problem (Initial State)



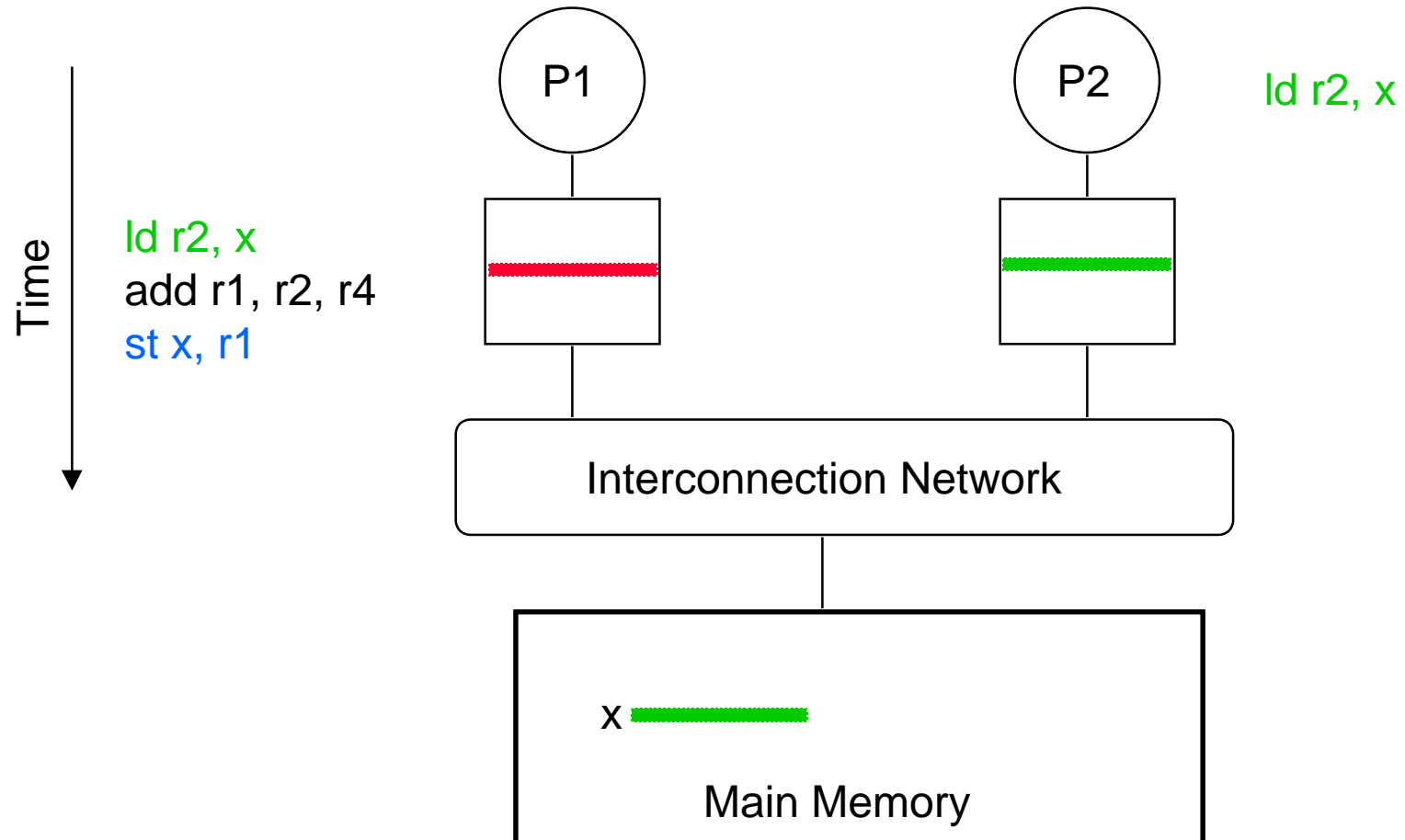
Cache Coherence Problem (Step 1)



Cache Coherence Problem (Step 2)



Cache Coherence Problem (Step 3)

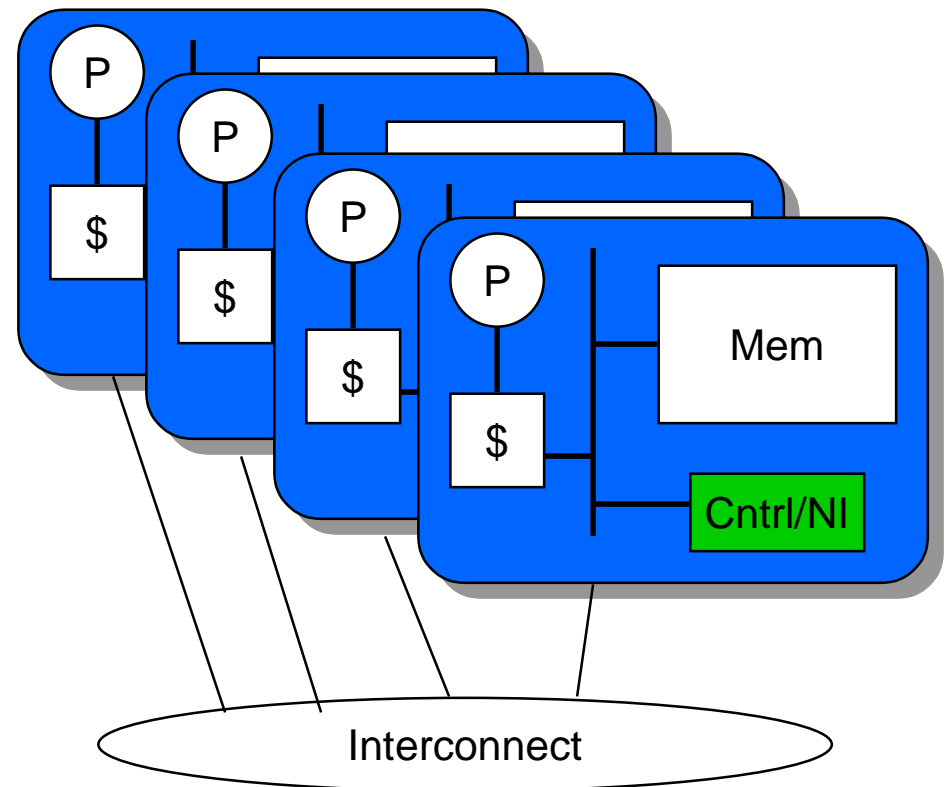


Snoopy Cache-Coherence Protocols

- **Bus provides serialization point for consistency**
- **Each cache controller “snoops” all bus transactions**
 - ◆ relevant transactions if for a block it contains
 - ◆ take action to ensure coherence
 - invalidate
 - update
 - supply value
 - ◆ depends on state of the block and the protocol
- **Simultaneous Operation of Independent Controllers**

Large Scale Shared Memory Multiprocessors

- 100s to 1000s of nodes (processors) with single shared physical address space
- Use General Purpose Interconnection Network
 - ◆ Still have cache coherence protocol
 - ◆ Use messages instead of bus transactions
 - ◆ No hardware broadcast
- Communication Assist

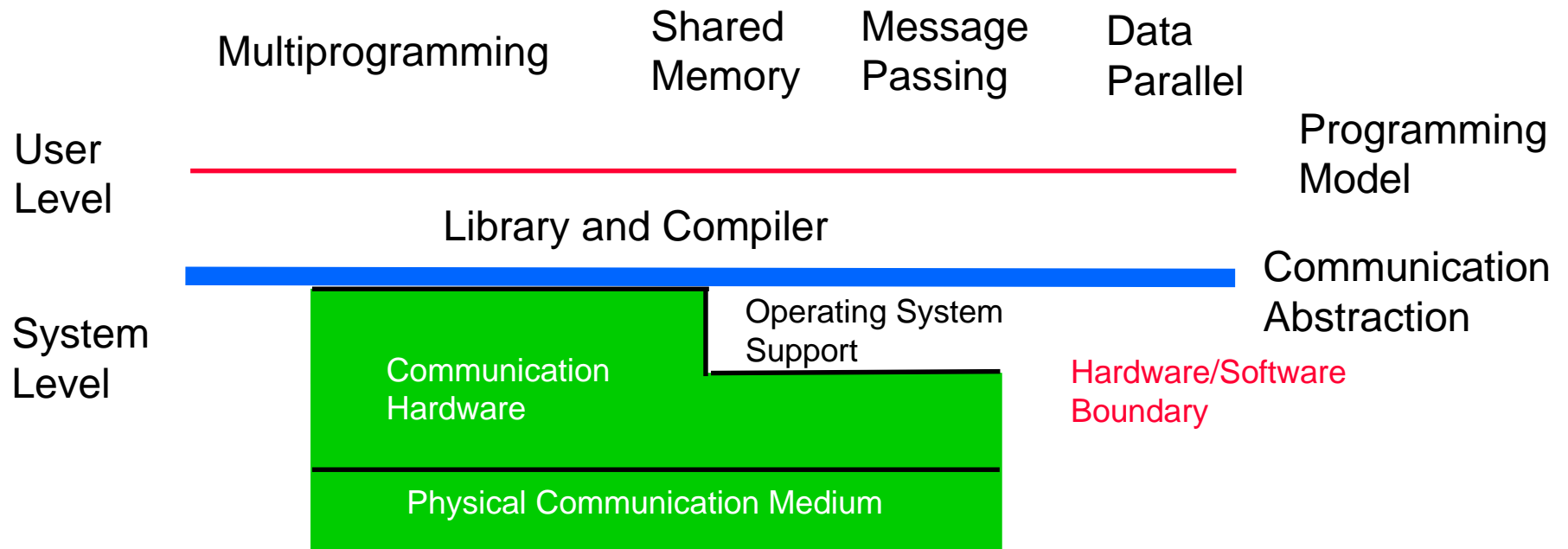


Directory Based Cache Coherence

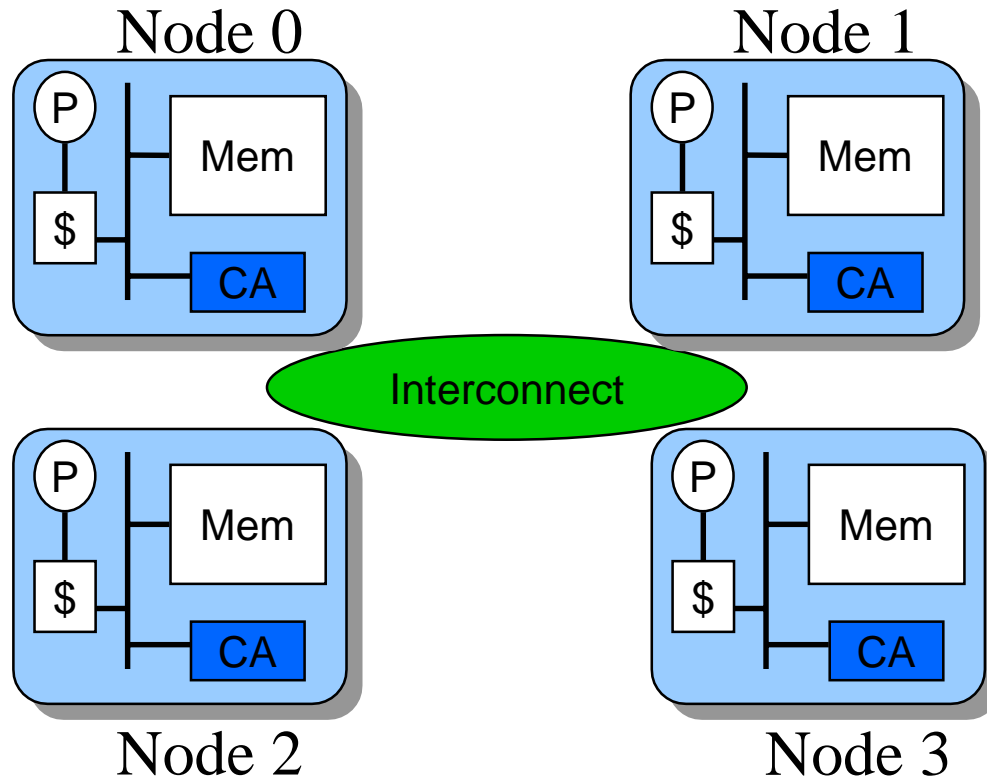
- Avoid broadcast request to all nodes on a miss
 - ◆ traffic
 - ◆ time
- Maintain **directory** of which nodes have cached copies of the block (directory **controller** + directory **state**)
- On a miss, send message to directory
- Directory determines what (if any) protocol action is required
 - ◆ e.g., invalidation
- Directory waits for protocol actions to finish and then responds to the original request

Today's Parallel Computer Architecture

- **Extension of traditional computer architecture to support communication and cooperation**
 - ◆ **Communications architecture**



Toward a Generic Parallel Machine



- Separation of programming models from architectures
- All models require communication
- Node with processor(s), memory, **communication assist**