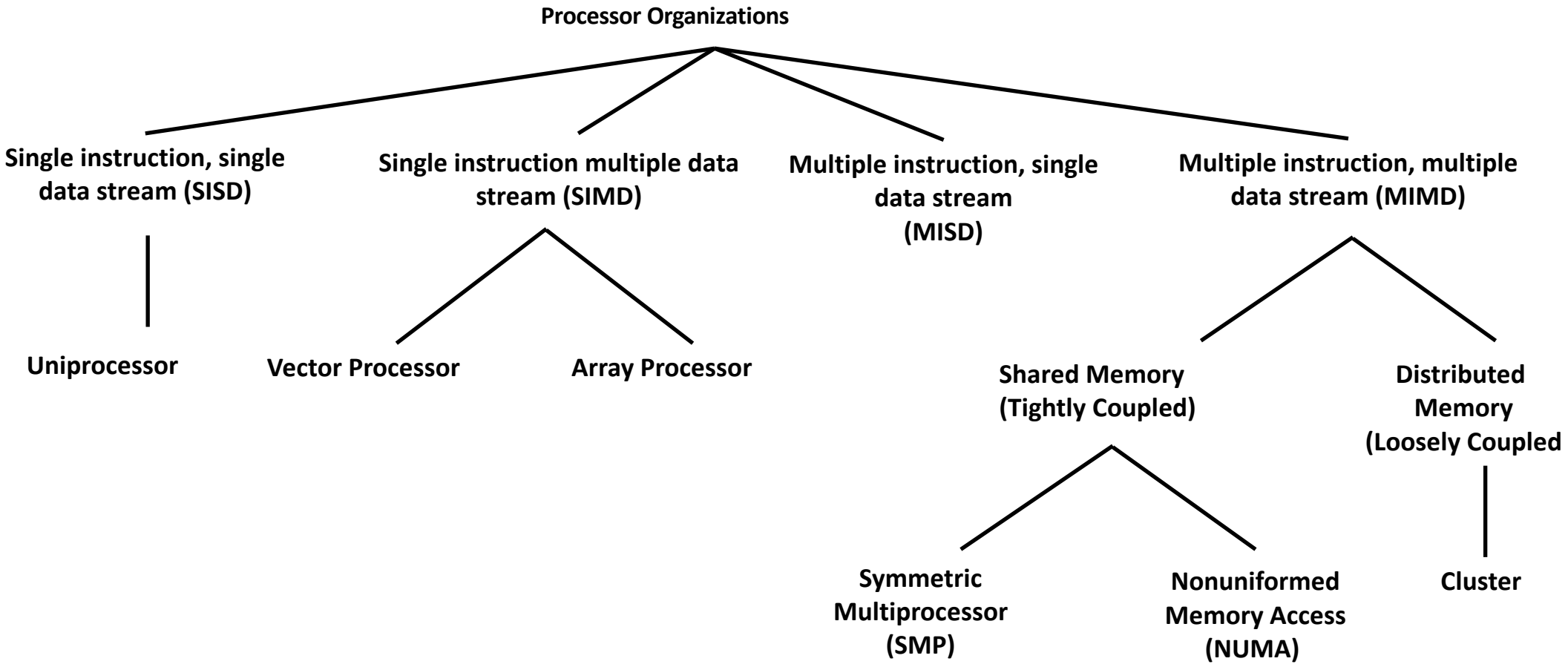
CSE 520
Computer Architecture II

Cache Coherence

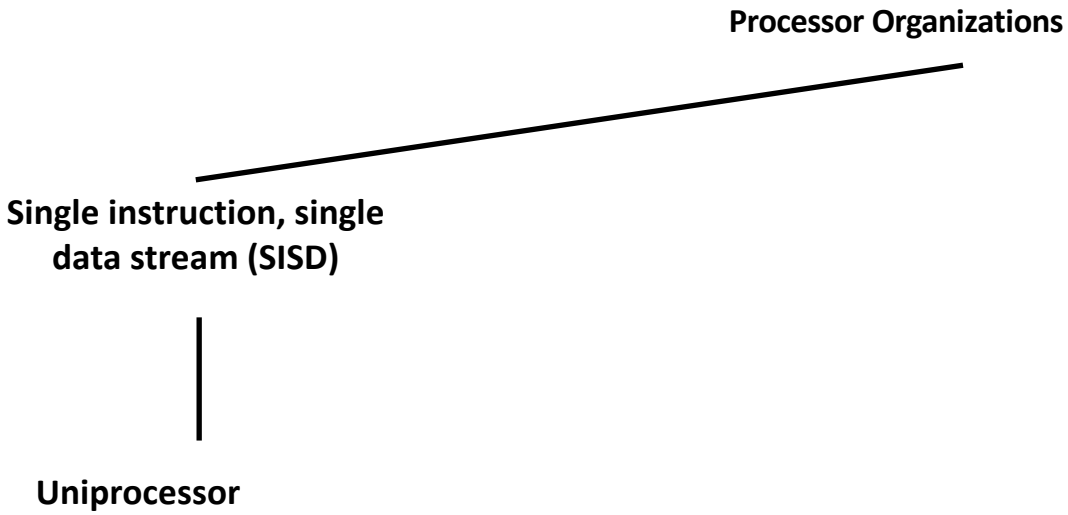
Prof. Michel A. Kinsy

Architecture Taxonomy



Parallelism Paradigms: Instruction level, Data level and Task level Parallelisms

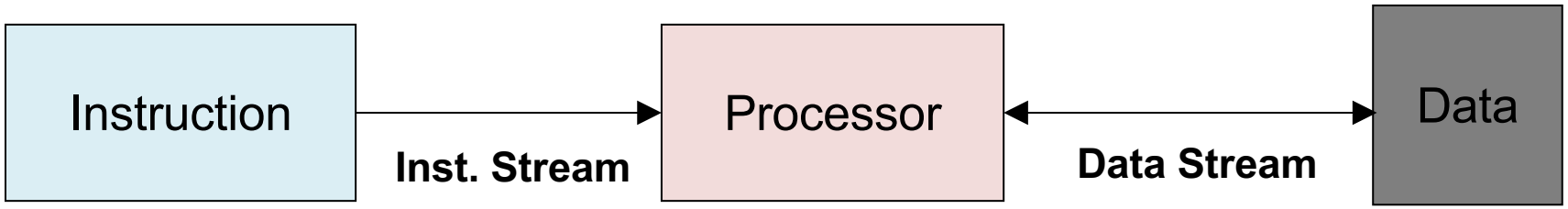
Architecture Taxonomy



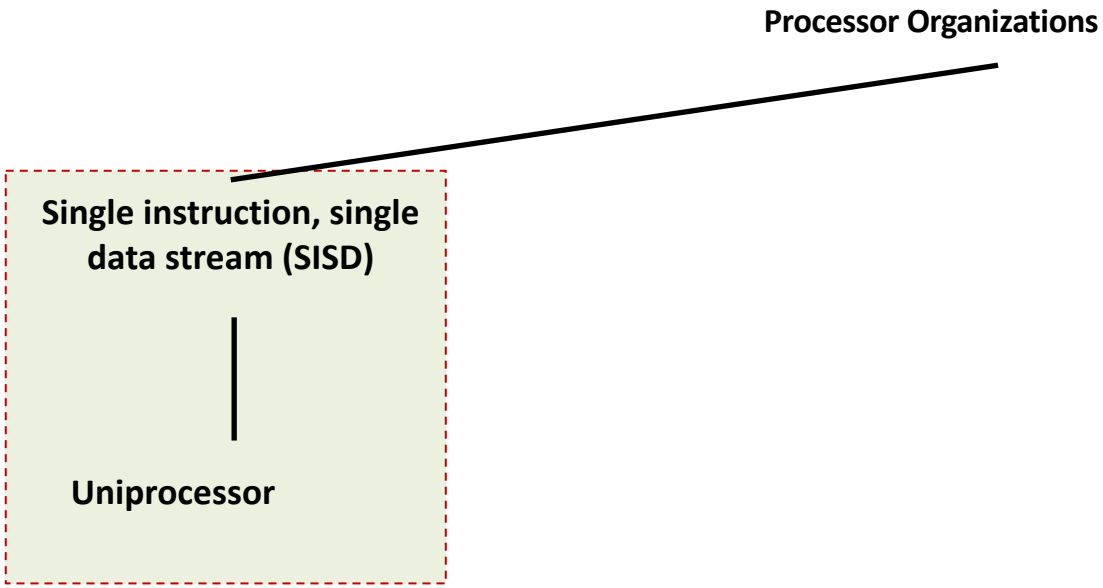
Parallelism Paradigms: Instruction level, Data level and Task level Parallelisms

Parallel Architectures

- Single instruction, single data stream – SISD
  - Single processor
  - Single instruction stream
  - Data stored in single memory



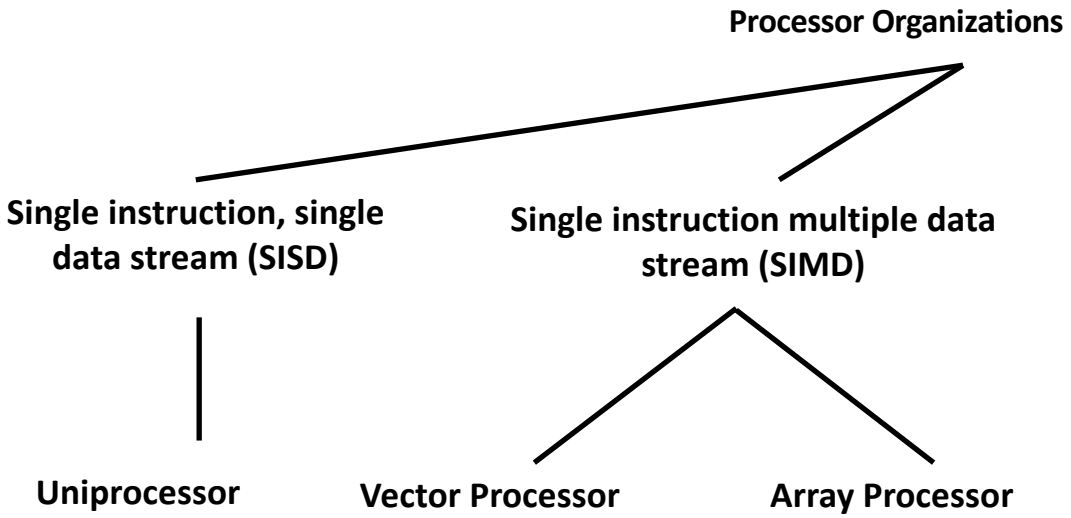
Architecture Taxonomy



Presented RISC-V Architecture

Parallelism Paradigms: Instruction level, Data level and Task level Parallelisms

Architecture Taxonomy



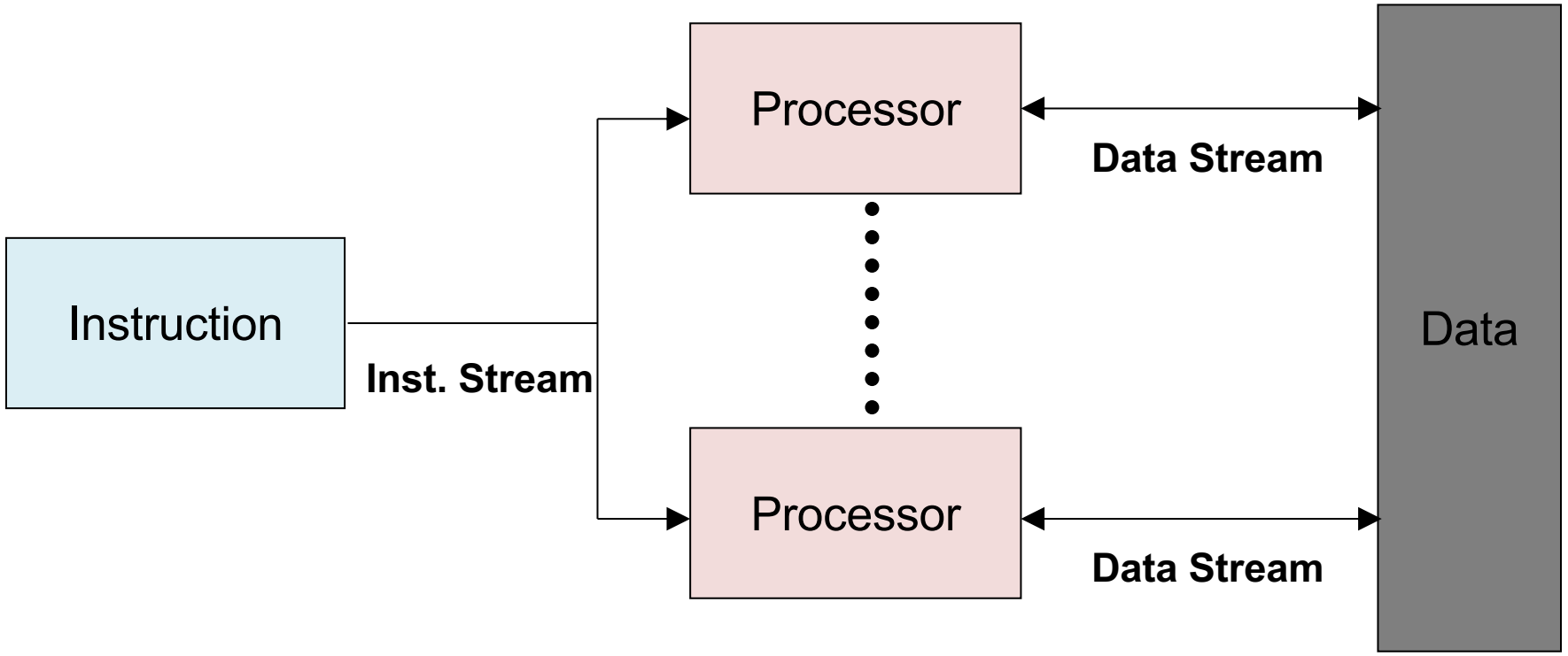
Parallelism Paradigms: Instruction level, Data level and Task level Parallelisms

Parallel Architectures

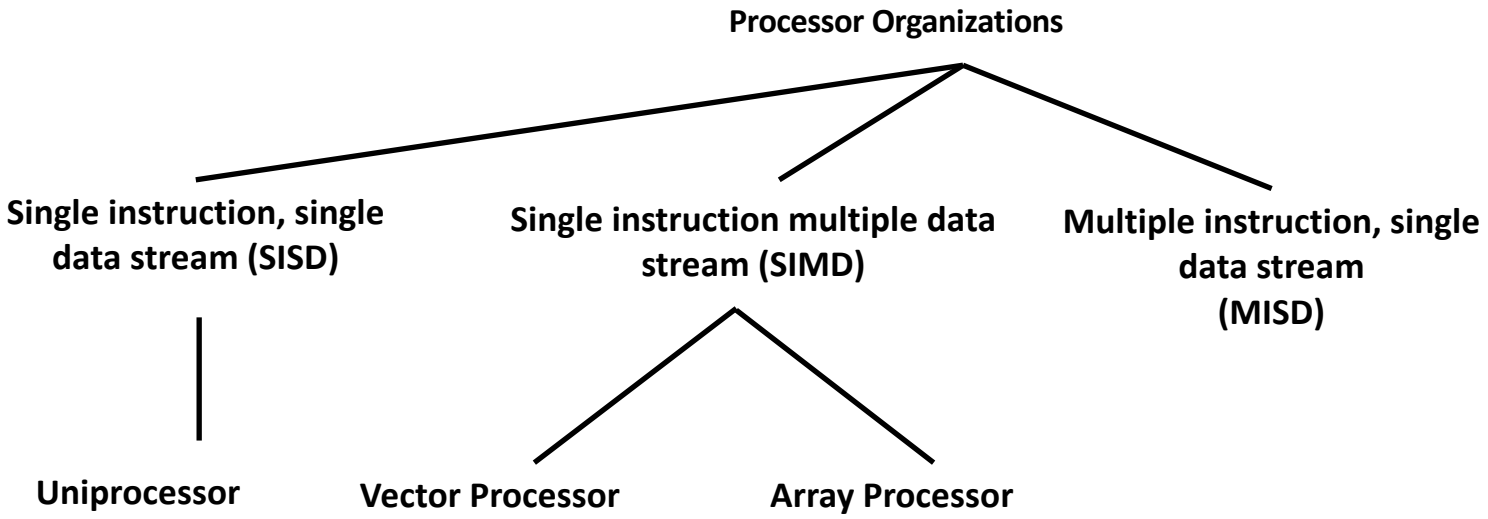
- Single instruction, multiple data stream – SIMD
- Single machine instruction
  - Each instruction executed on different set of data by different processors
- Number of processing elements
  - Machine controls simultaneous execution
    - Lockstep basis
  - Each processing element has associated data memory
- Application: Vector and array processing

Parallel Architectures

- Single instruction, multiple data stream – SIMD



Architecture Taxonomy



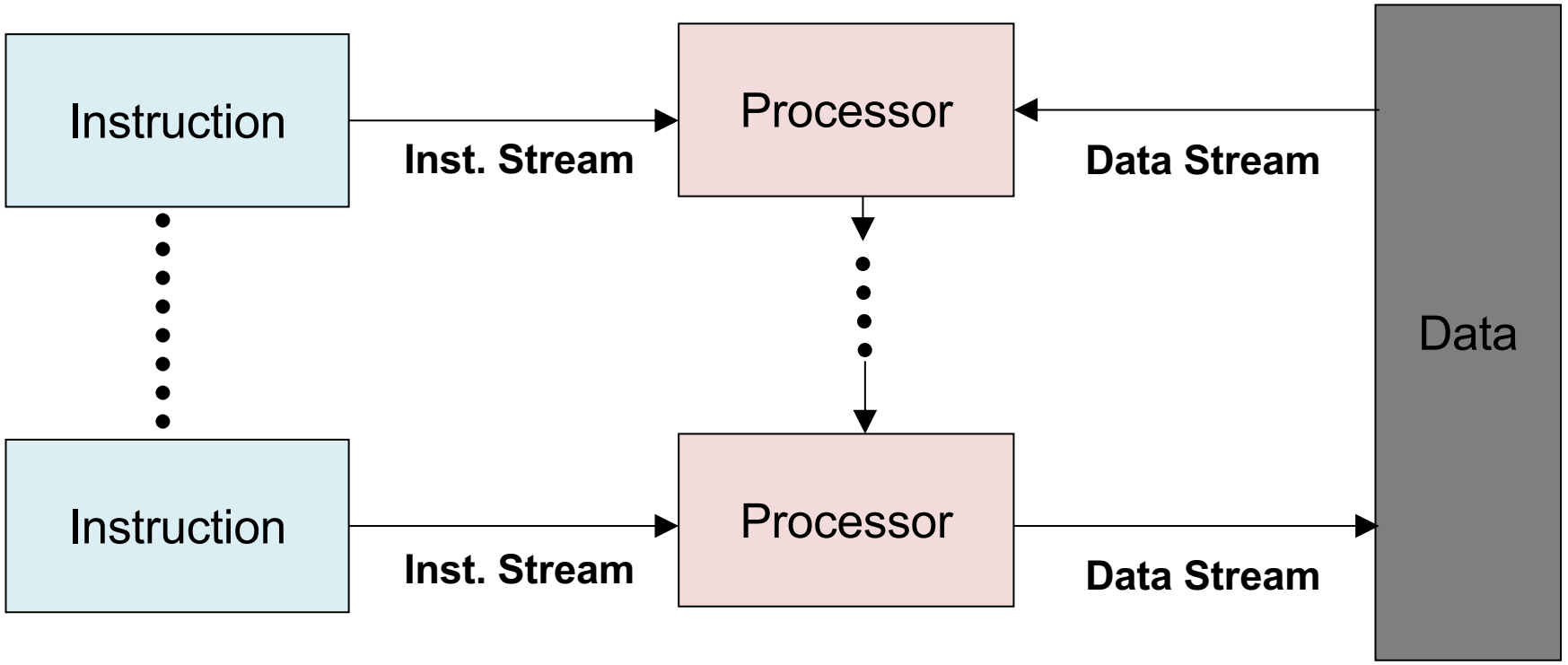
Parallelism Paradigms: Instruction level, Data level and Task level Parallelisms

Parallel Architectures

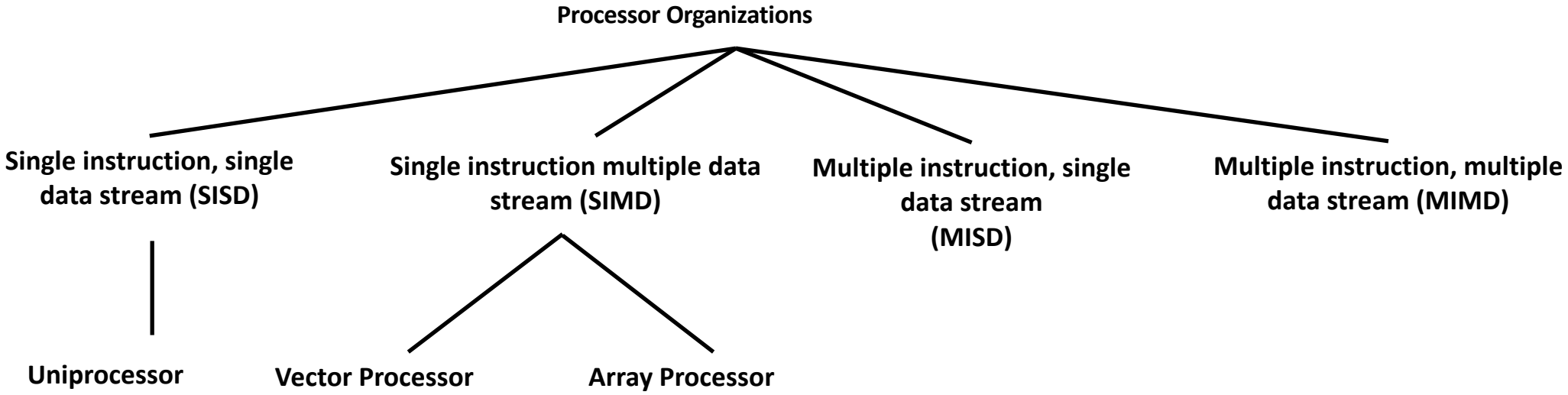
- Multiple instruction, single data stream – MISD
  - Sequence of data
  - Transmitted to set of processors
  - Each processor executes different instruction sequence
- Do not know any implemented case

Parallel Architectures

- Multiple instruction, single data stream – MISD



Architecture Taxonomy



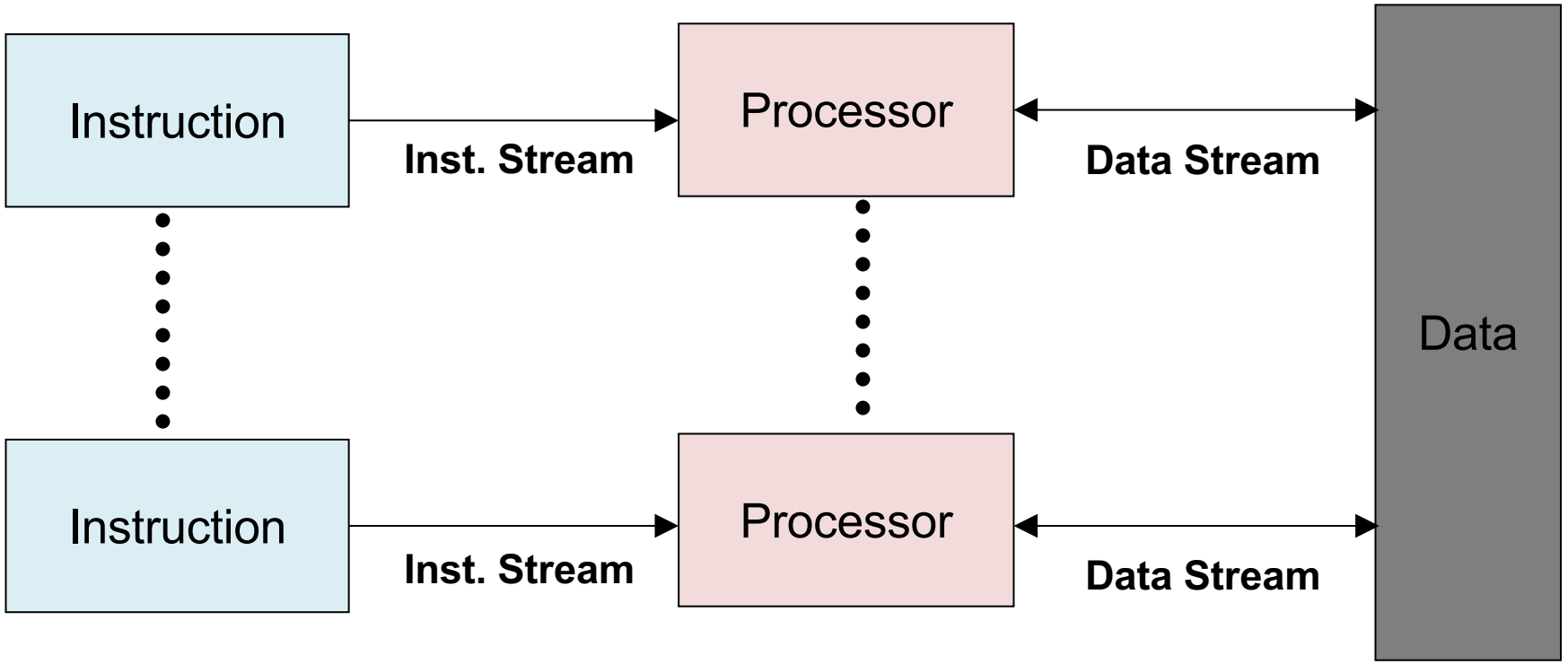
Parallelism Paradigms: Instruction level, Data level and Task level Parallelisms

Parallel Architectures

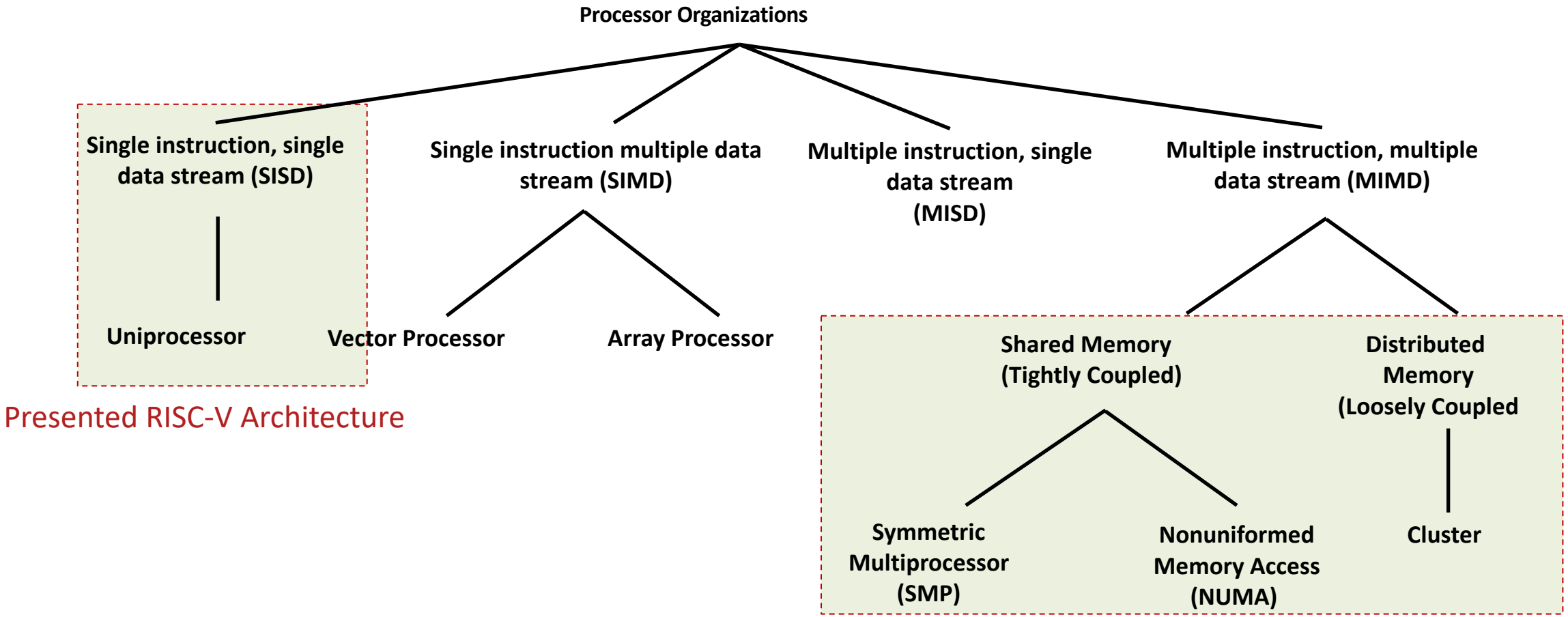
- Multiple instruction, multiple data stream- MIMD
  - Set of processors
  - Simultaneously executes different instruction sequences
  - Different sets of data
- Examples: SMPs, NUMA systems, and Clusters

Parallel Architectures

- Multiple instruction, multiple data stream- MIMD



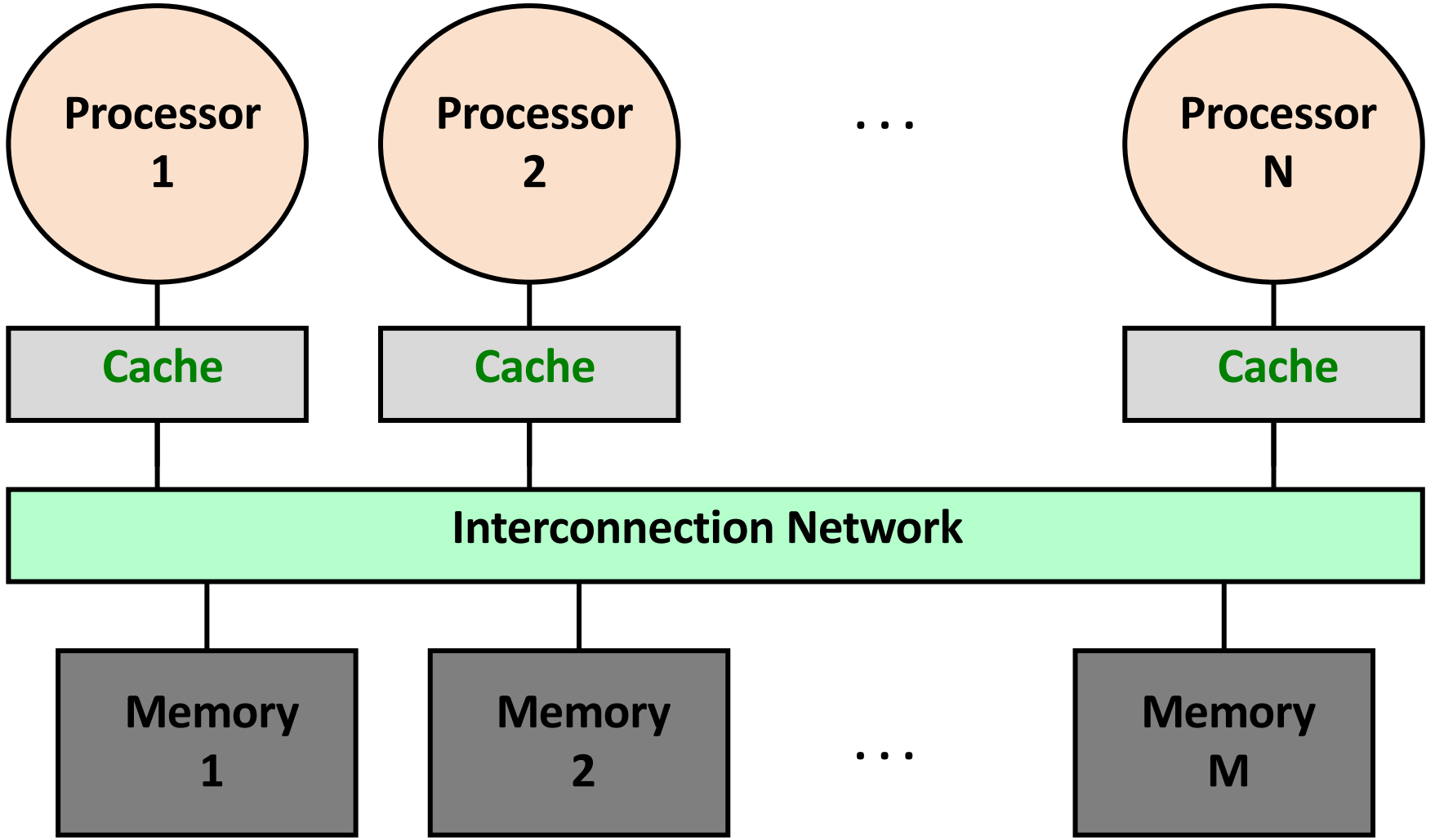
Architecture Taxonomy



Presented RISC-V Architecture

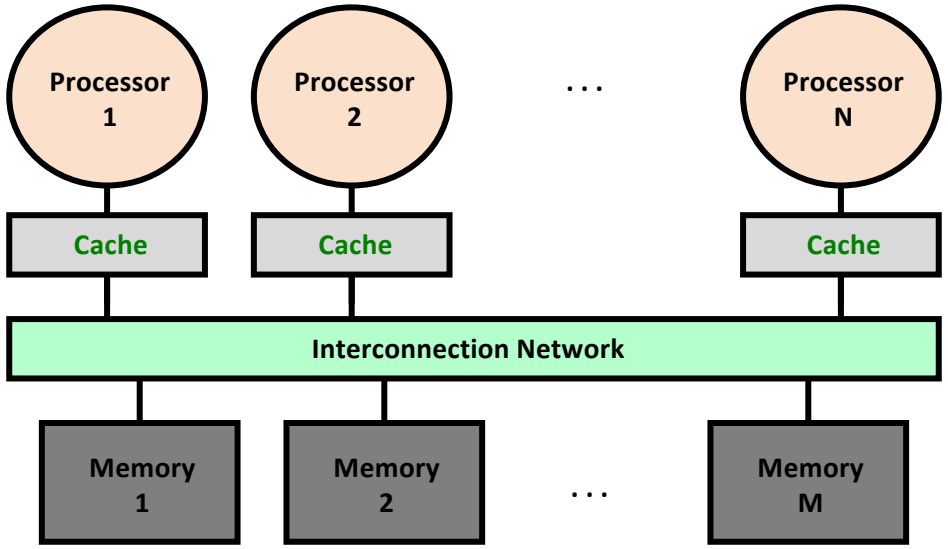
Parallelism Paradigms: Instruction level, Data level and Task level Parallelisms

Symmetric Multiprocessors (SMP)



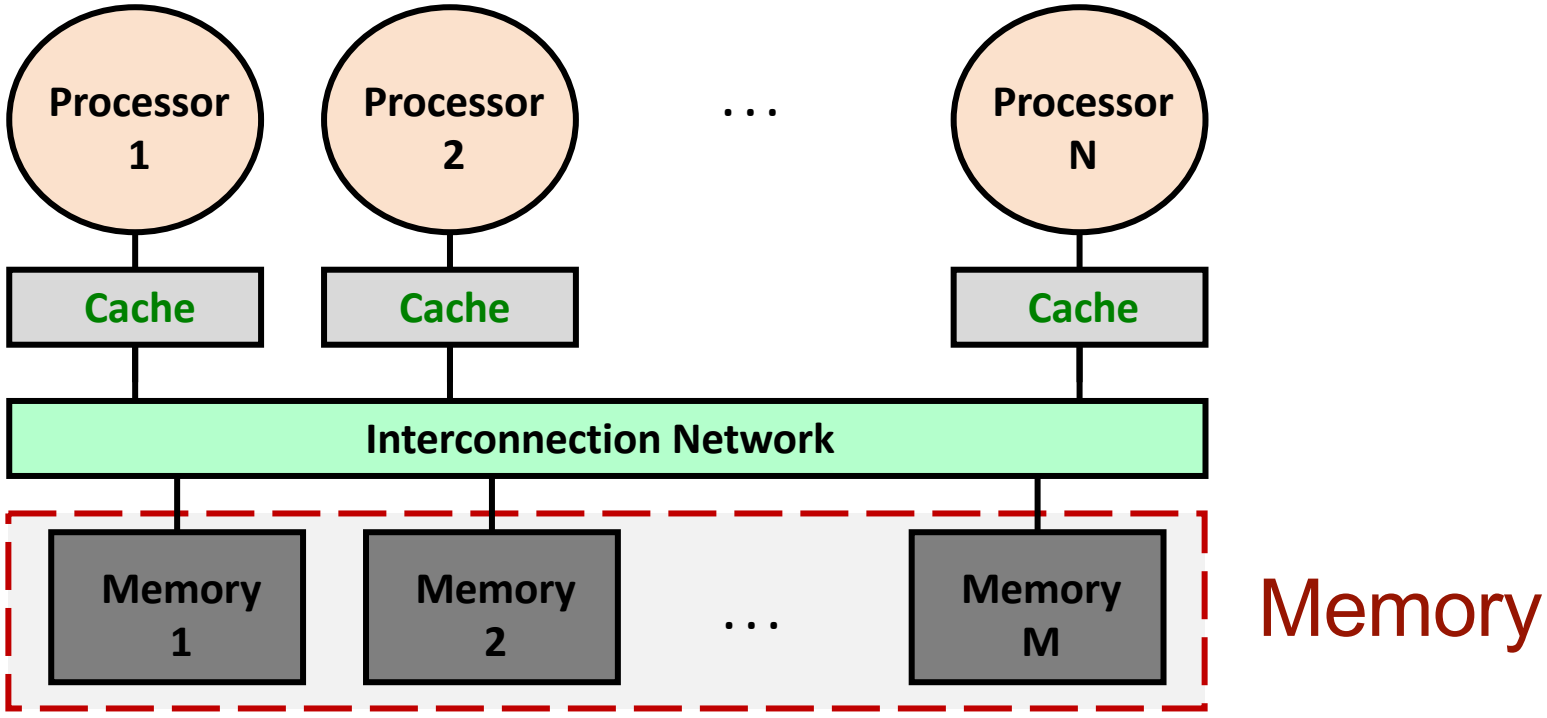
Symmetric Multiprocessors (SMP)

- A collection of processors and a collection of memory connected through some interconnect
- Symmetric because latency for any processor to access any memory is constant – uniform memory access (UMA)

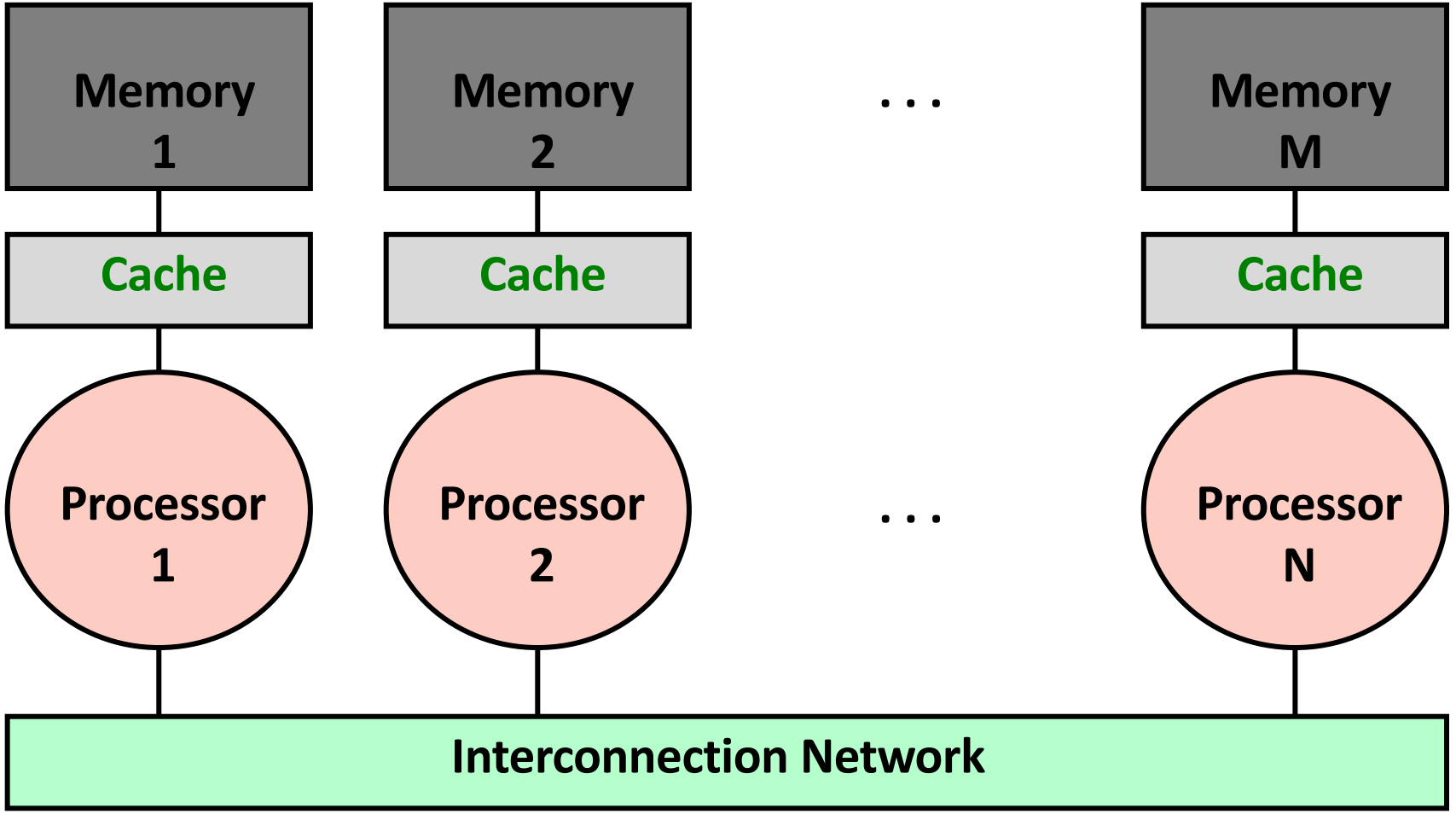


Shared Memory Architectures

- Key differentiating feature: the address space is shared, i.e., any processor can directly address any memory location and access them with load/store instructions

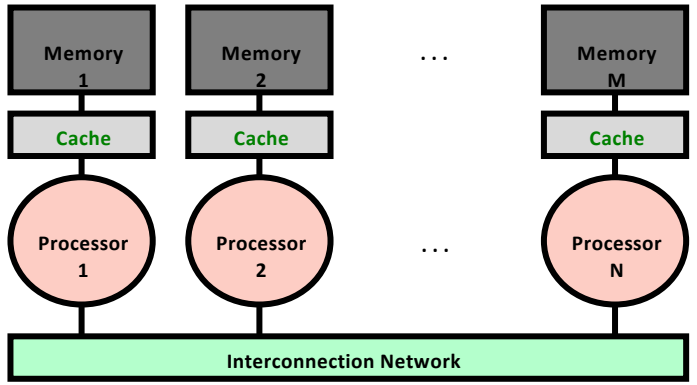


Distributed Memory Multiprocessors



Distributed Memory Multiprocessors

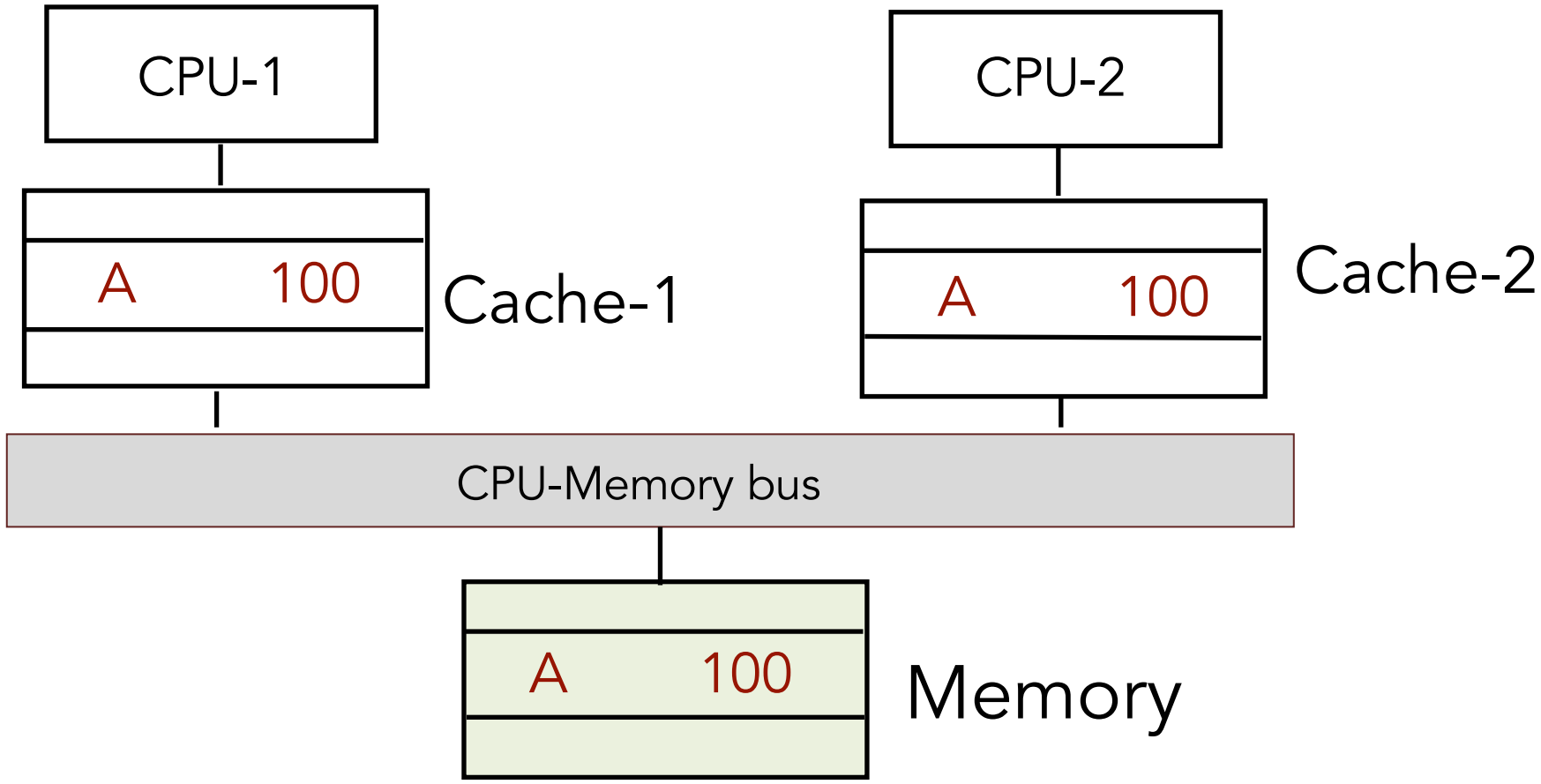
- Each processor has local memory that is accessible through a fast interconnect
- The different nodes are connected as I/O devices with (potentially) slower interconnect
- Local memory access is a lot faster than remote memory
  - Nonuniform memory access (NUMA)



Parallel Architectures

- The parallel computers are classified as
  - Shared memory
  - Distributed memory
- Both shared and distributed memory systems have:
  - Processors: now generally commodity processors
  - Memory: now general commodity DRAM/DDR
  - Network/interconnect: between the processors or memory

Memory Consistency in SMPs



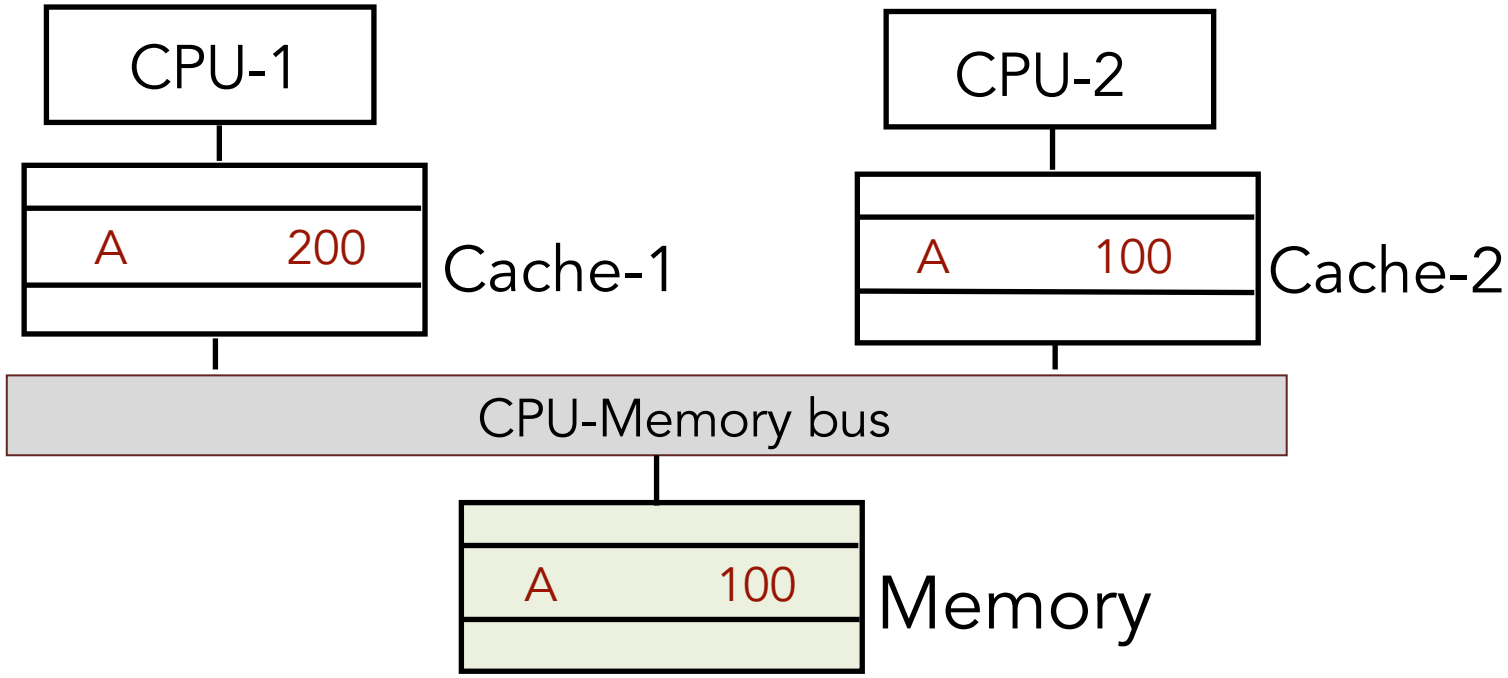
Cache Reads & Writes

- Cache reads
  - With a hit, the data can be retrieved
  - On a miss, we need to take a set of actions
- Cache writes
  - On a hit
    - Write-Through: Every time we write to a variable in cache, we also need to update that variable in memory
    - Write-Back: When we write to a variable in the cache, the modified copy is written back to memory when it gets evicted or replaced in the cache.
      - Write-back can improve performance, but is more complex to implement
        - Dirty Bit: In a write-back cache, cache blocks have a dirty bit
        - Block Replacement: Block is written back to memory if it is dirty and replaced with a new memory block

Cache Reads & Writes

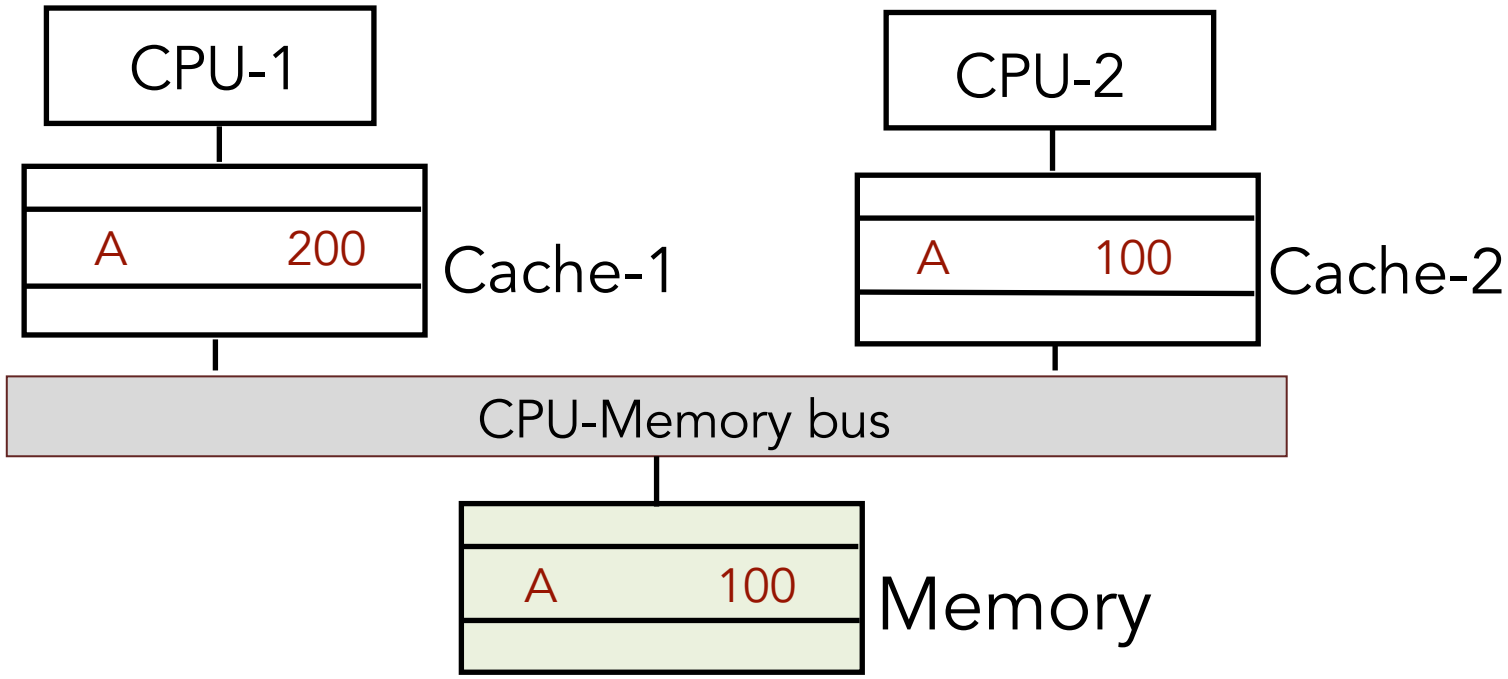
- Cache reads
  - With a hit, the data can be retrieved
  - On a miss, we need to take a set of actions
- Cache writes
  - On a miss
    - Write Allocate - i.e., Fetch on Write
      - Load block into memory as on a read miss
      - Perform Write Hit actions
    - No Write Allocate - i.e., Write Around
      - Modify block in lower memory directly without loading into cache
- Relation of write miss policies to write hit policies
  - Common pairings
    - Write Through with No Write Allocate
    - Write Back with Write Allocate

Memory Consistency in SMPs



- Suppose CPU-1 updates A to 200
  - Write-back: memory and cache-2 have stale values
  - Write-through: cache-2 has a stale value

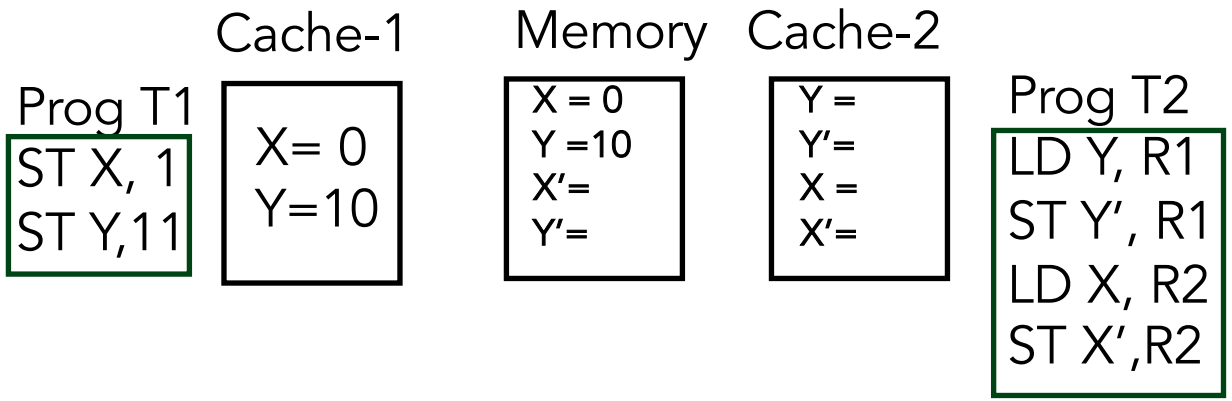
Memory Consistency in SMPs



- Suppose CPU-1 updates A to 200
  - Do these stale values matter?
  - What is the view of shared memory for programming?

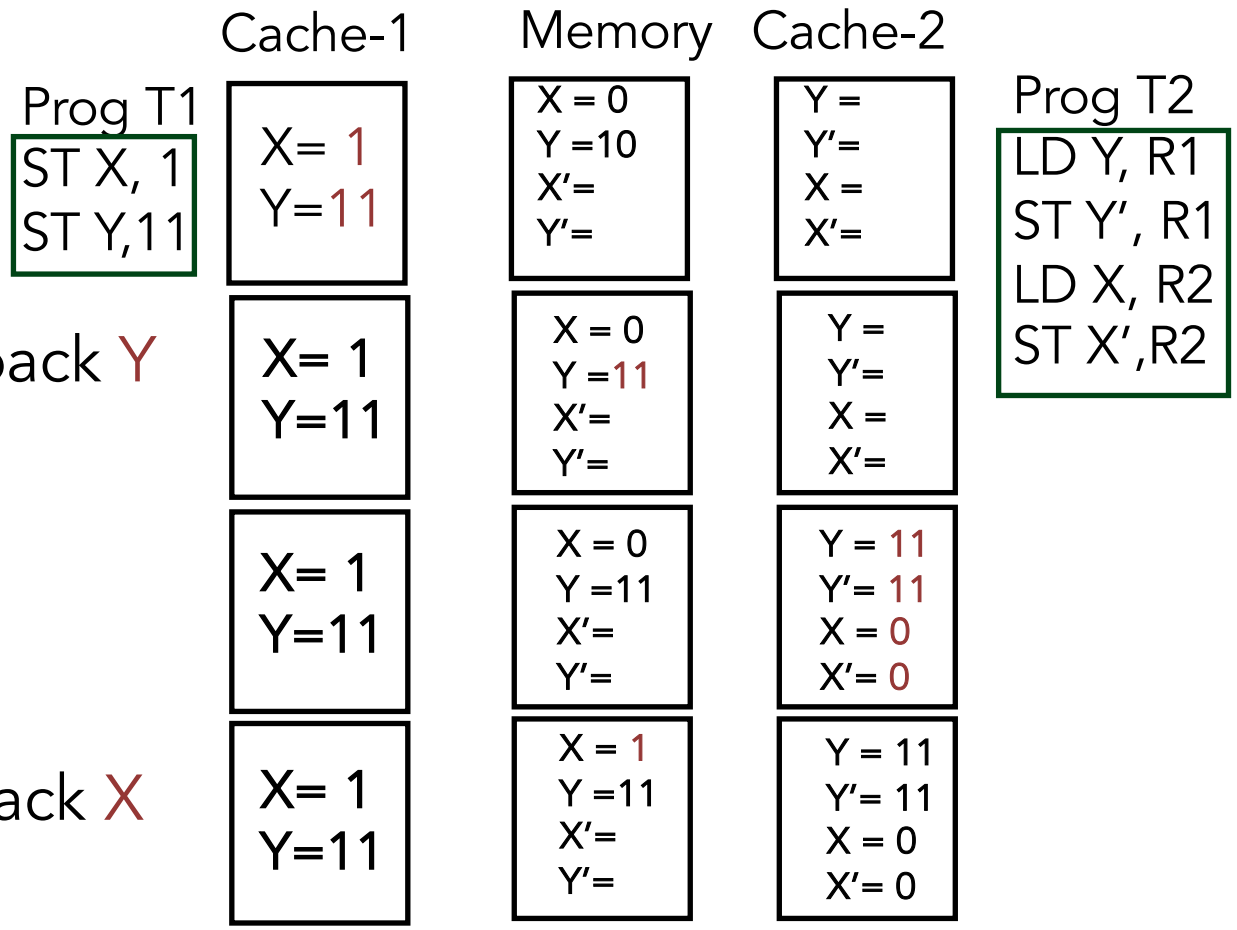
Write-through Caches & Sequential Consistency

- T1 is executed



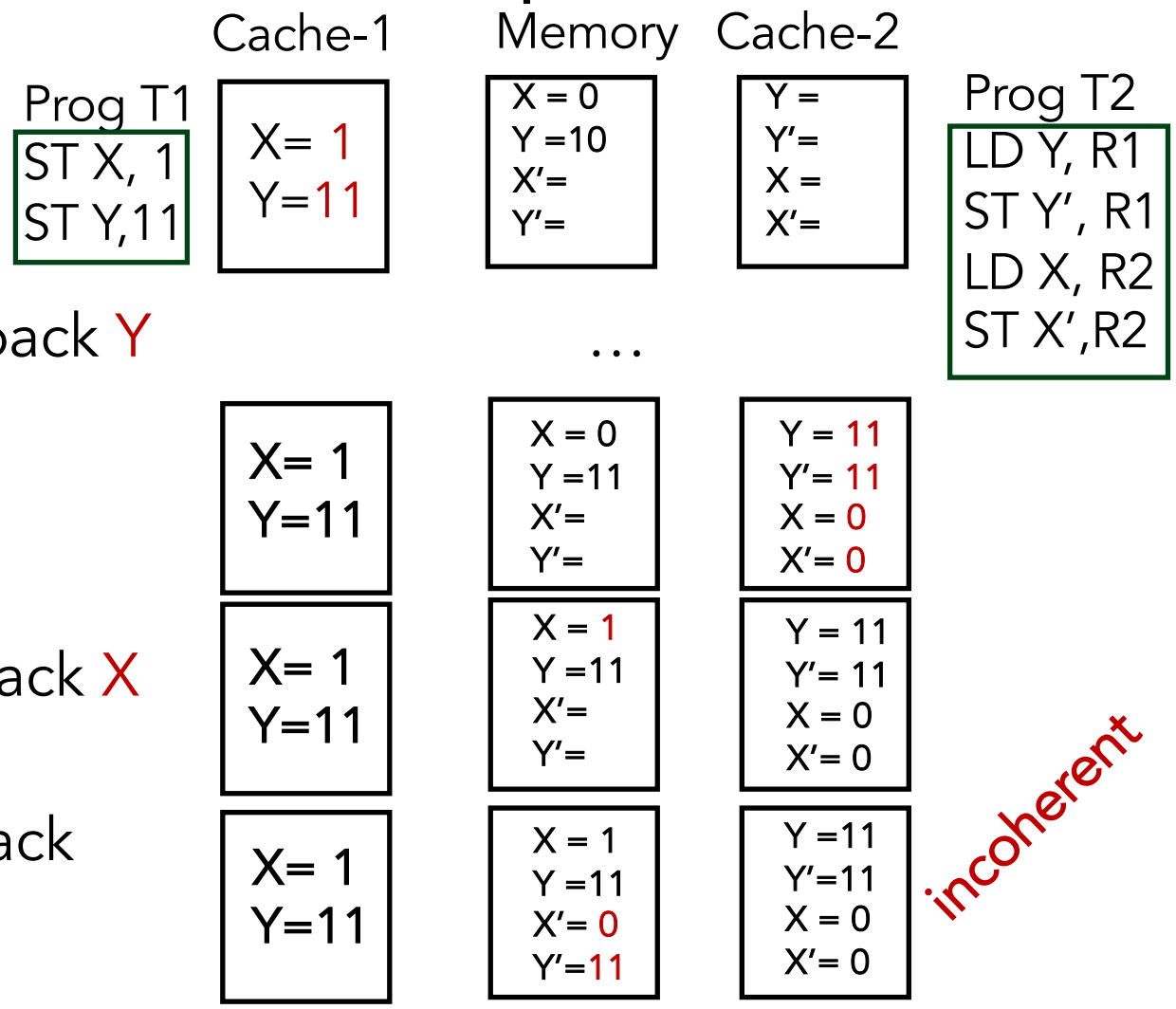
Write-through Caches & Sequential Consistency

- T1 is executed
- Cache-1 writes back Y
- T2 executed
- Cache-1 writes back X

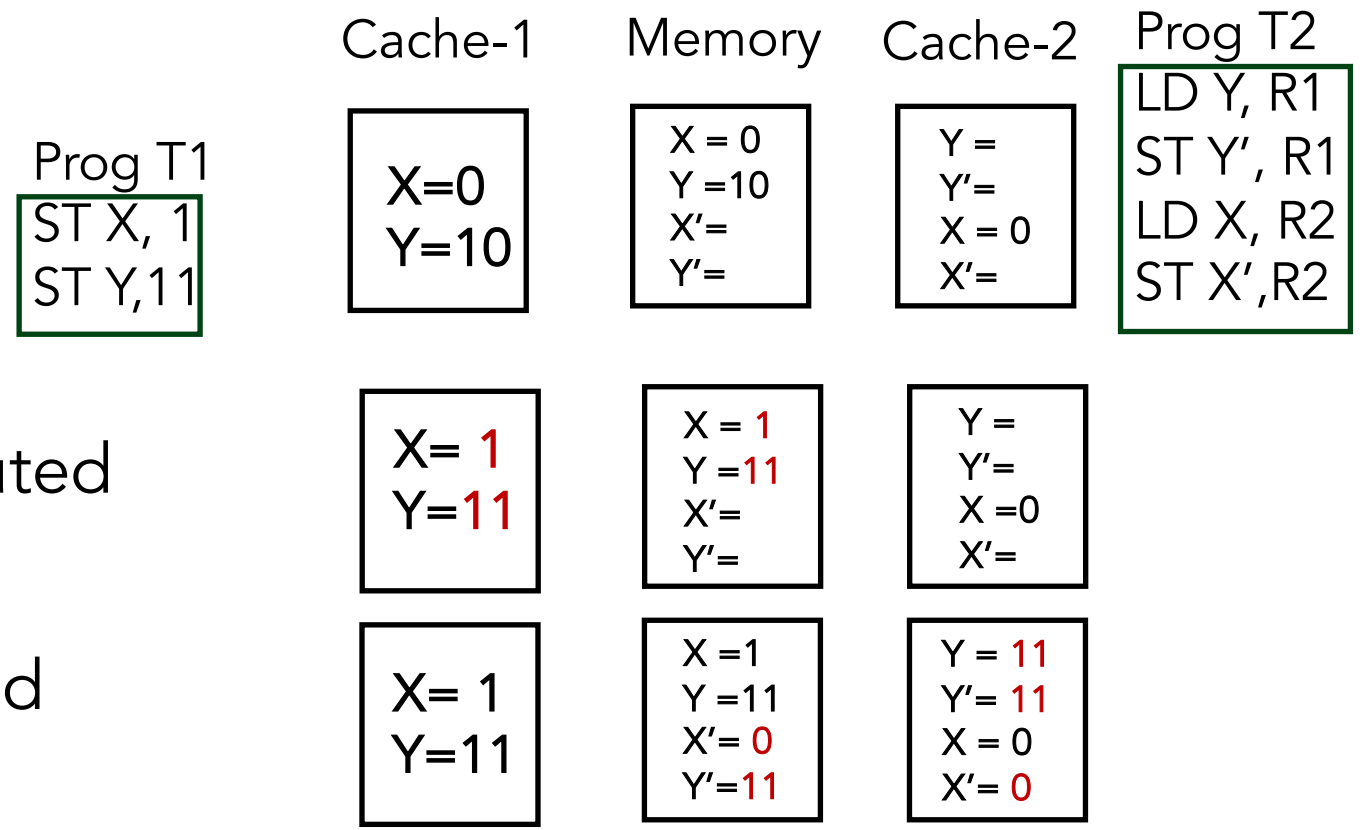


Write-through Caches & Sequential Consistency

- T1 is executed
- Cache-1 writes back Y
- T2 executed
- Cache-1 writes back X
- Cache-2 writes back X′ & Y′



Write-through Caches & Sequential Consistency

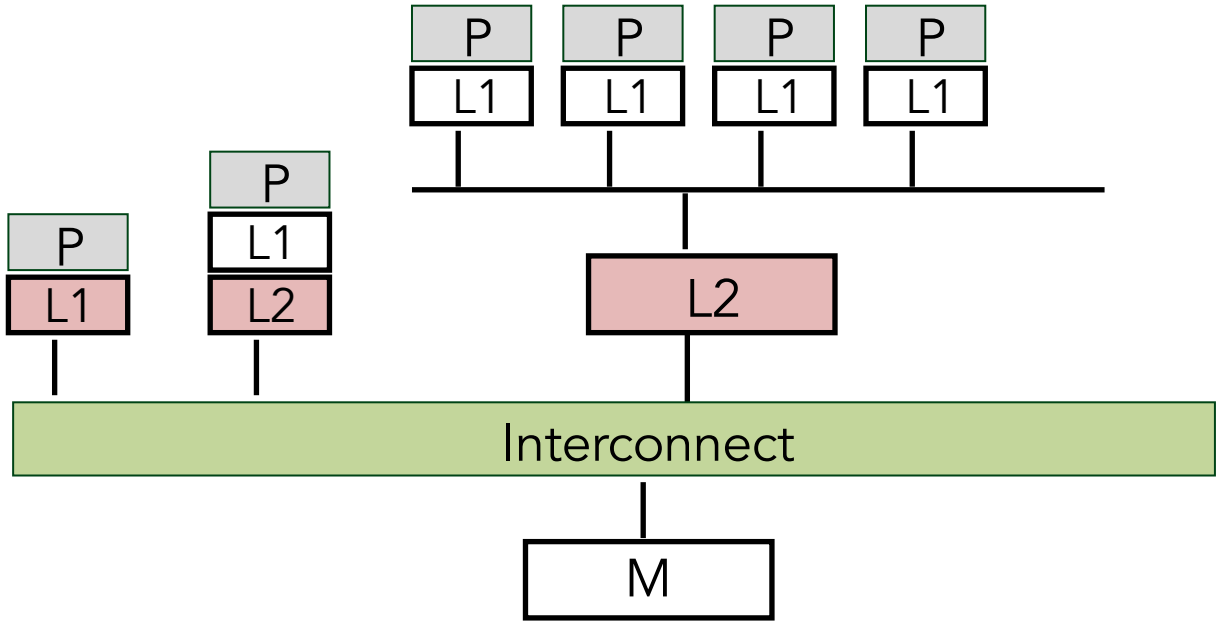


- T1 is executed
- T2 executed
- Write-through caches don't preserve sequential consistency either

Maintaining Sequential Consistency

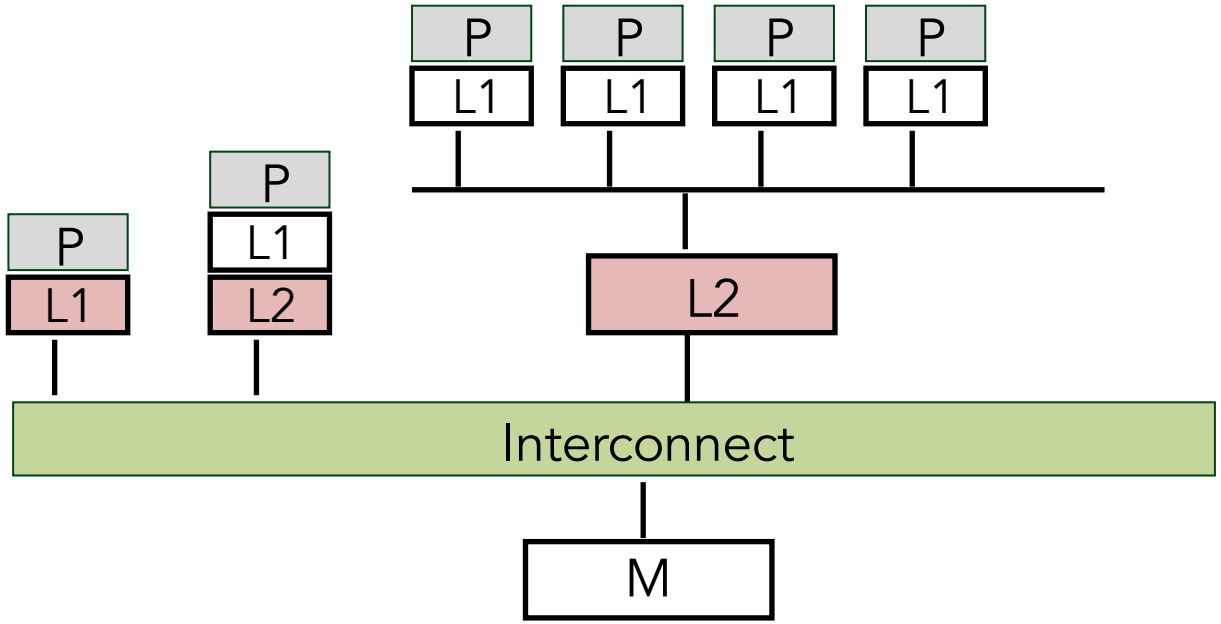
- Problem: Multiple copies of a location in various caches can cause SC to break down.
- Hardware support is required such that
  - Only one processor at a time has write permission for a location
  - No processor can load a stale copy of the location after a write
    - Cache coherence protocols

A System with Multiple Caches



- Modern systems often have hierarchical caches
- Each cache has exactly one parent but can have zero or more children
- Only a parent and its children can communicate directly

A System with Multiple Caches



- Inclusion property is maintained between a parent and its children, i.e.,
  - a in L_i implies that a in L_{i+1}

Cache Coherence Protocols for SC

- Write request:
  - The address is invalidated in all other caches before the write is performed, or
  - The address is updated in all other caches after the write is performed
- Read request:
  - If a dirty copy is found in some cache, a write-back is performed before the memory is read
  - We will focus on Invalidation protocols as opposed to Update protocols

Next Learning Module

- Cache Coherence Protocols