CS 207 D

Computer Architecture

Lecture 9: Multiprocessors

Challenges of Parallel Processing

- First challenge is % of program inherently sequential
- • Second challenge is long latency to remote memory

1. **Application parallelism** □ primarily via new algorithms that have better parallel performance

2. **Long remote latency impact** □ both by architect and by the programmer

More Challenges:

- **Parallel Programming**
- **Complex Algorithms**
- **Memory issues**
- **Communication Costs**
- **Load Balancing**

Shared-Memory Architectures

- **A Parallel Architecture is a collection of processing elements that cooperate to solve large problems fast**
- From multiple boards on a shared bus to multiple processors inside a single chip

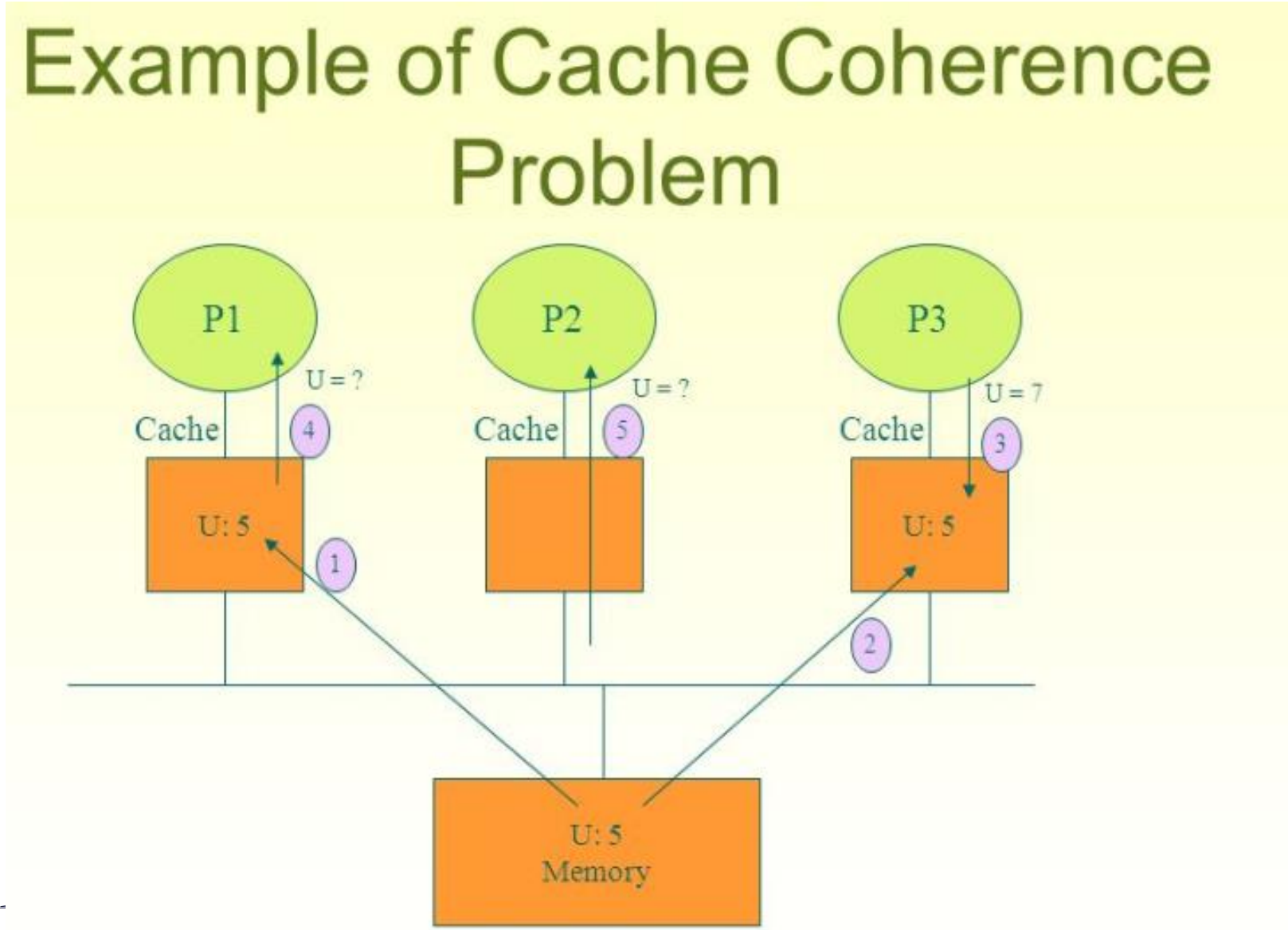
- **Caching shared data** (HW)

Caches both

- **Private data** are used by a single processor
- **Shared data** are used by multiple processors

- **Caching shared data**
- reduces latency to shared data, memory bandwidth for shared data, and interconnect bandwidth

Cache coherence problem



Amdahl's Law

- • **Amdahl's Law:**
- – performance improvements throug an enhancement is limited
- by the fraction of time the enhancement comes into play

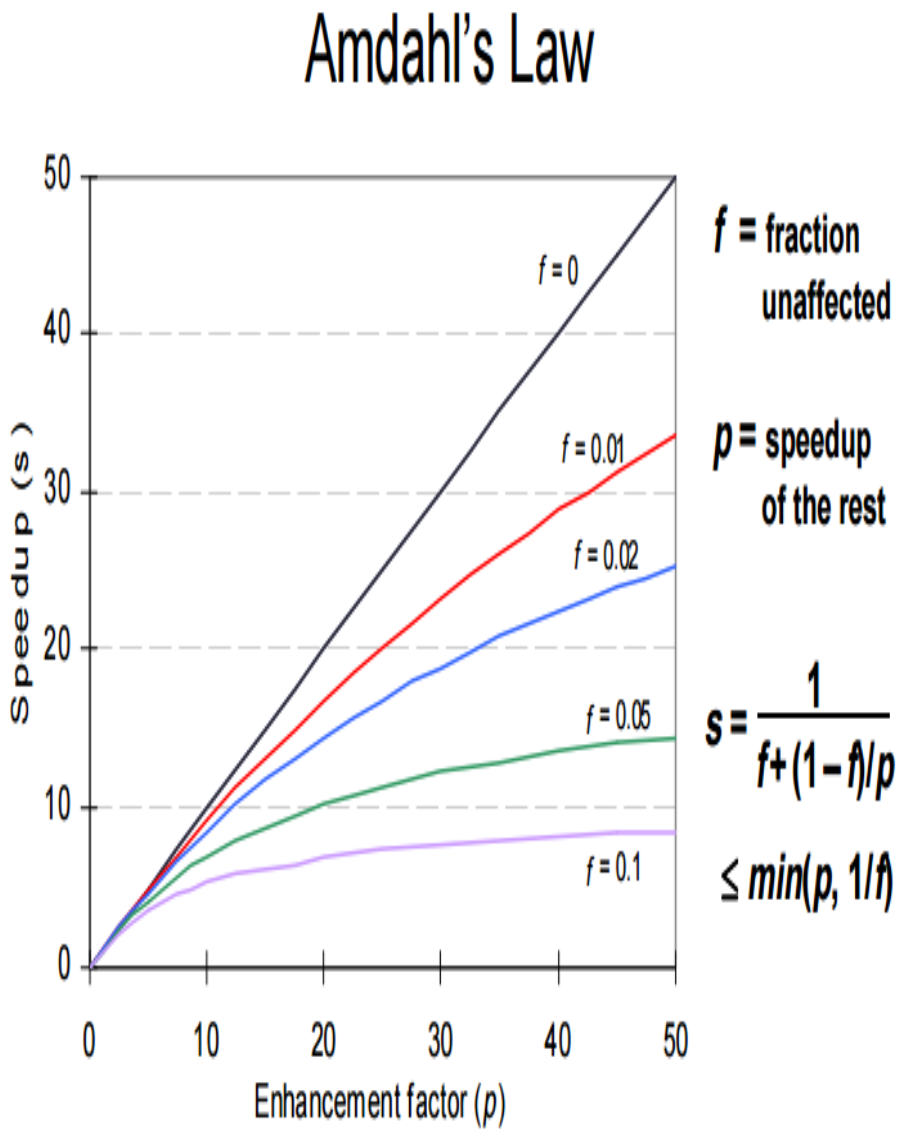
- Execution Time After Improvement =
- Execution Time Unaffected +

Execution Time Affected

Amount of Improvement

Speedup = Performance after improvement

Performance before improvement

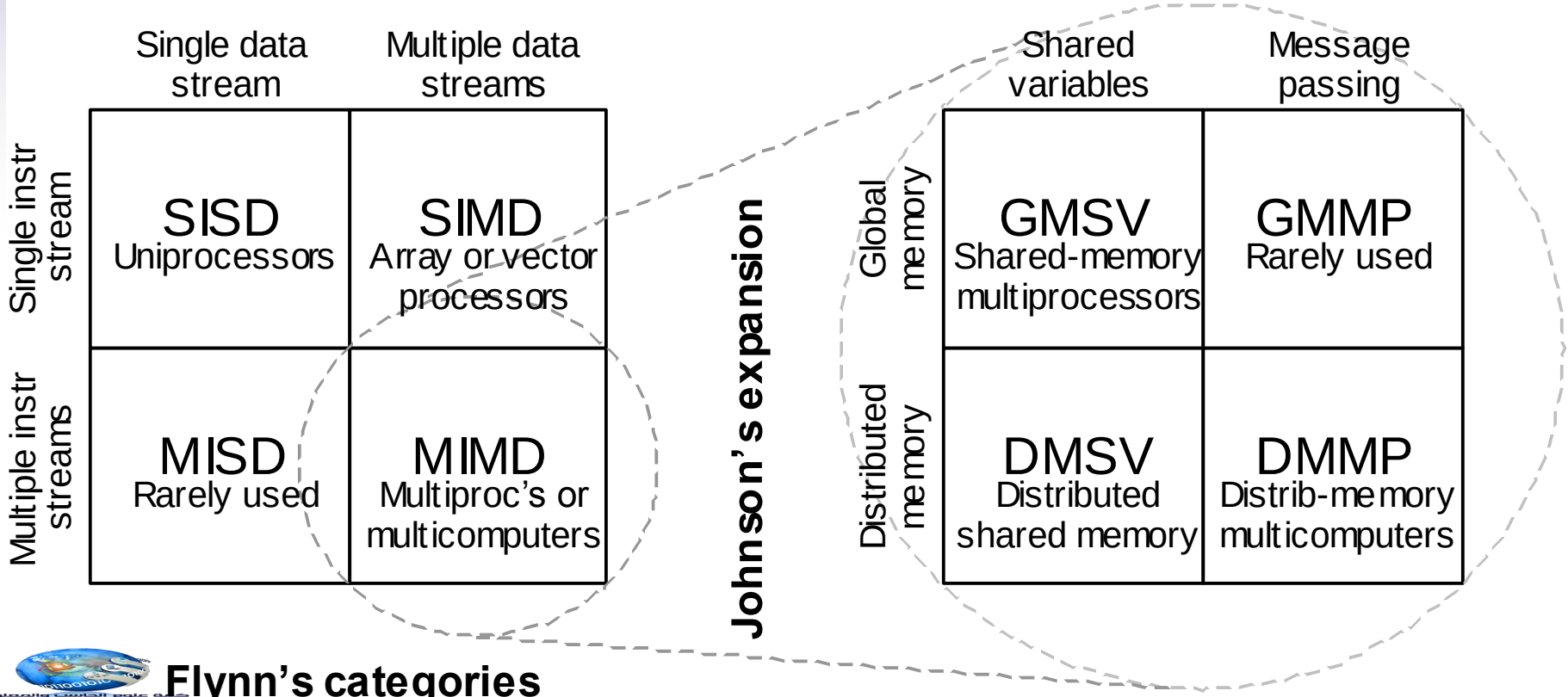


Flynn Classification of parallel processing architectures

- **Flynn classification of parallel processing architectures**

SISD, SIMD, MISD, MIMD

Basic multiprocessor architectures



Flynn's categories

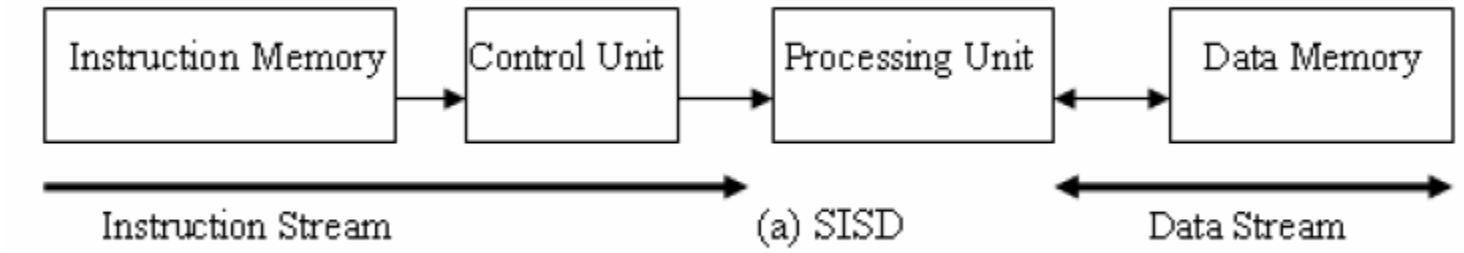
SISD

No instruction parallelism

No data parallelism

• SISD processing architecture example

– a personal computer processing instructions and data on single processor

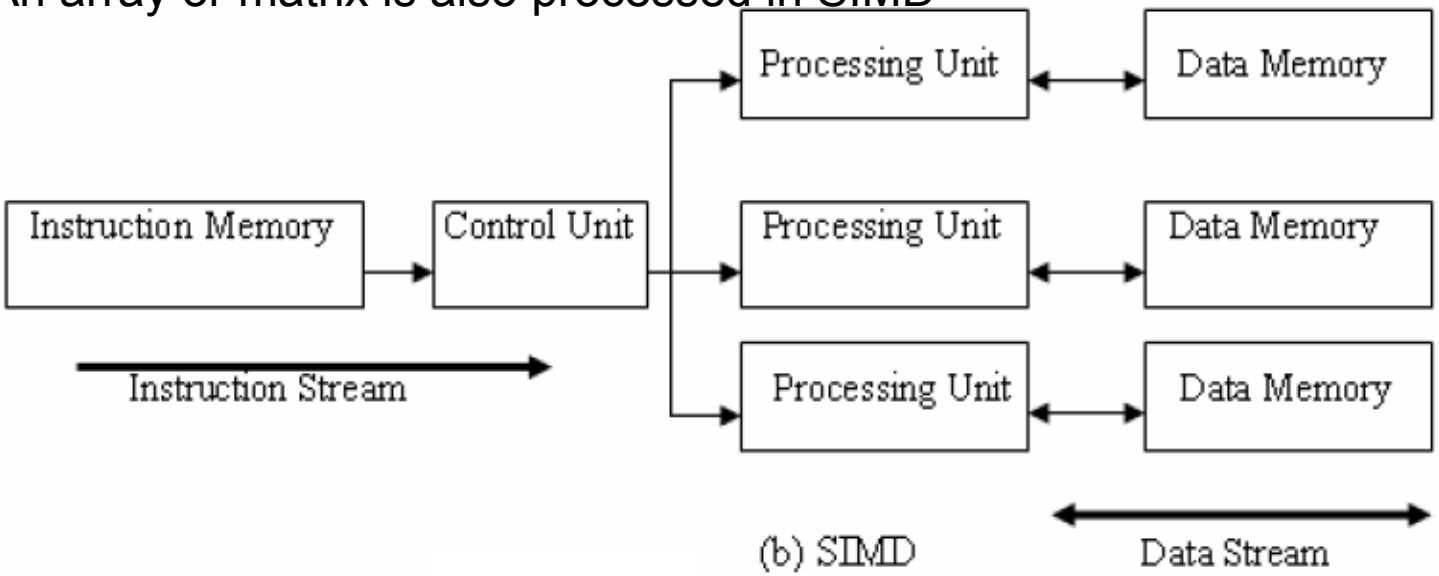


SIMD

- Multiple data streams in parallel with a single instruction stream
- At one time, one instruction operates on many data
  - Data parallel architecture
  - Vector architecture has similar characteristics, but achieve the parallelism with pipelining.
- **Array processors**

– a graphic processor processing instructions for translation or rotation or other operations are done on multiple data

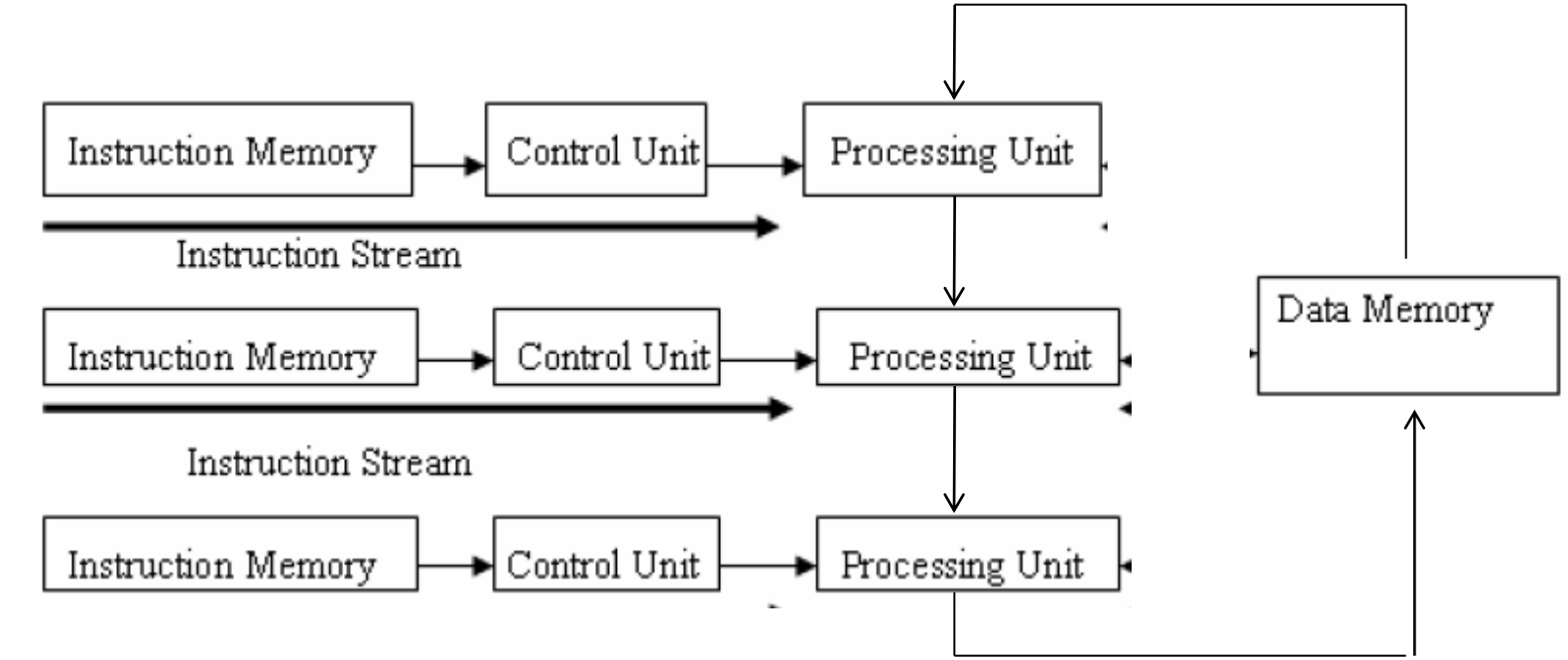
- • An array or matrix is also processed in SIMD



MISD

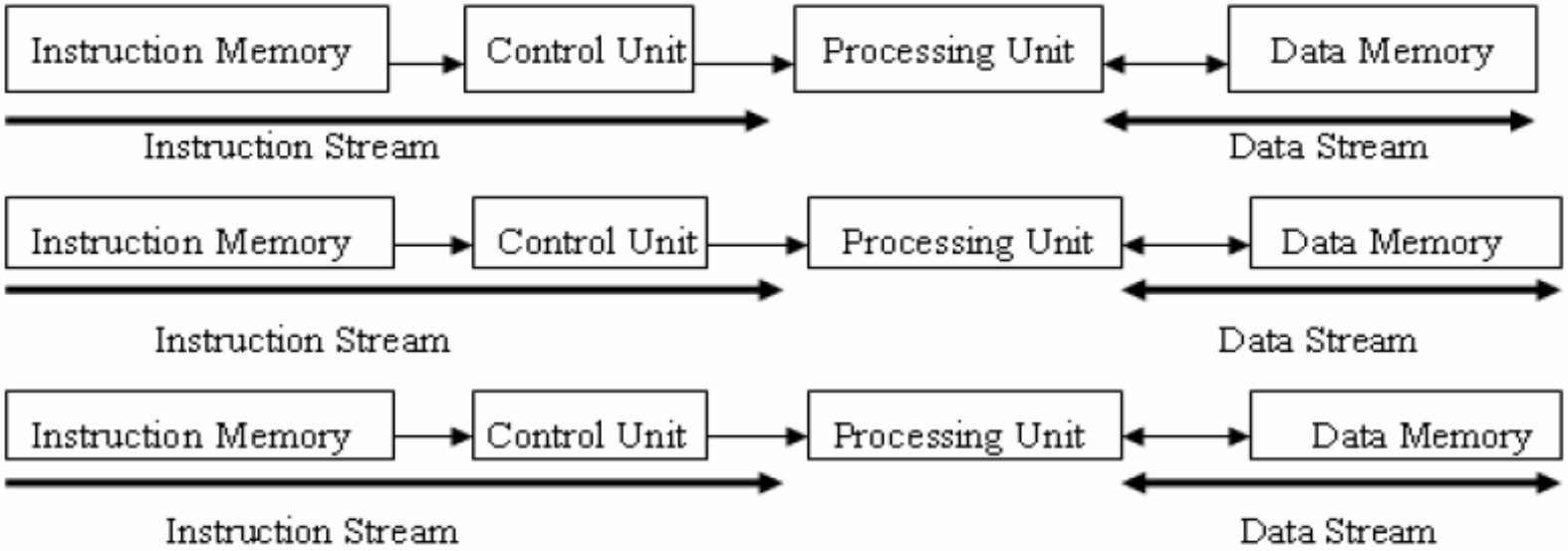
Multiple instruction streams in parallel operating on single Data stream

• Processing architecture example– processing for critical controls of missiles where single data stream processed on different processors to handle faults if any during processing



MIMD

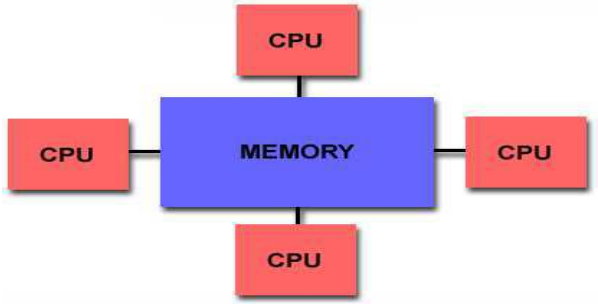
- Multiple processing streams in parallel processing on parallel data streams
- MIMD processing architecture example is super computer or distributed computing
- systems with distributed or single shared memory



Multiprocessor Basics

- **Parallel Architecture = Computer Architecture + Communication**

Architecture



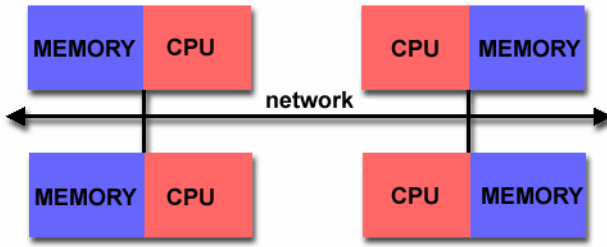
- 2 classes of multiprocessors WRT **memory**:

1. Centralized Memory Multiprocessor

- <few dozen cores
- Small enough to share single, centralized memory with **Uniform Memory Access** latency (**UMA**)

2. Physically Distributed-Memory Multiprocessor

- Larger number chips and cores than the first.
- BW demands Memory distributed among processors with **Non-Uniform Memory Access**/latency (NUMA)

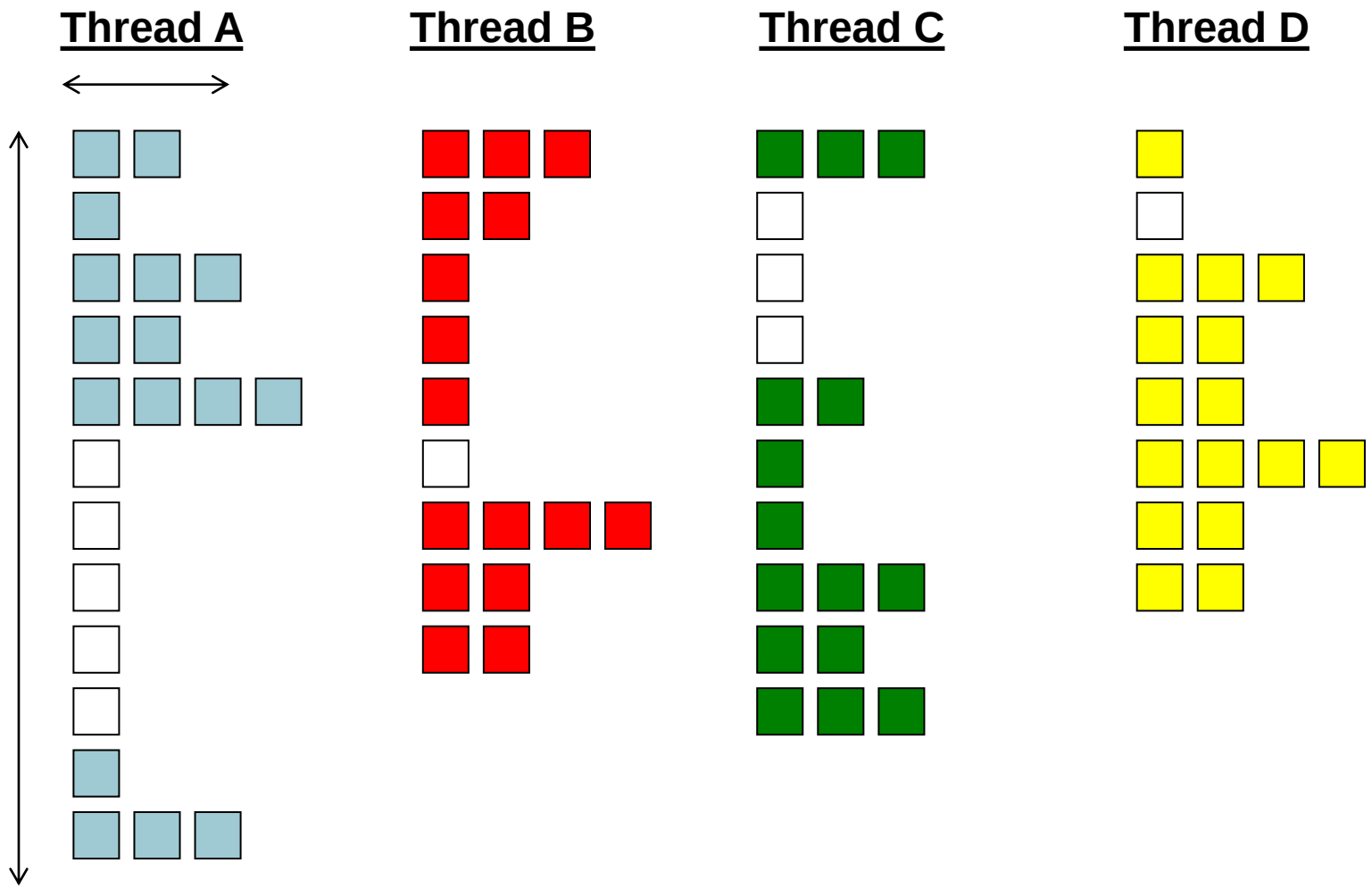


Definitions of Threads and Processes

- Instruction stream divided into smaller streams (threads)
- Parallel execution
- Thread in multithreaded processors may or may not be same as software threads
- Process:
  - An instance of program running on computer
- Thread:
  - dispatchable unit of work within process
- Thread switch
  - Switching processor between threads within same process
  - Typically less costly than process switch

a thread of execution is the smallest sequence of programmed instructions that can be managed independently

Hardware Multithreading Conceptual Diagram



Coarse Multithreading

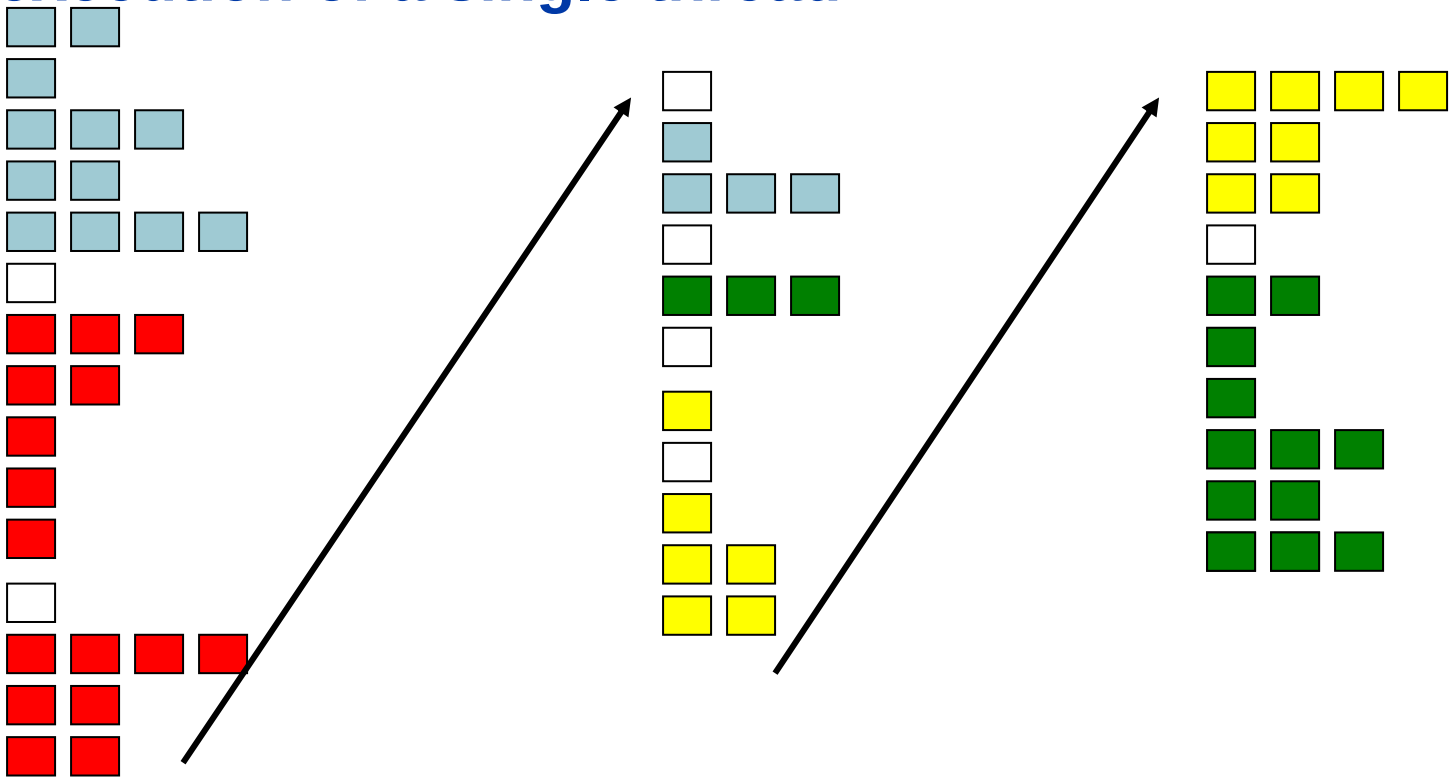
- Single thread runs until a costly stall

E.g. 2nd level cache miss

- Another thread starts during stall for first

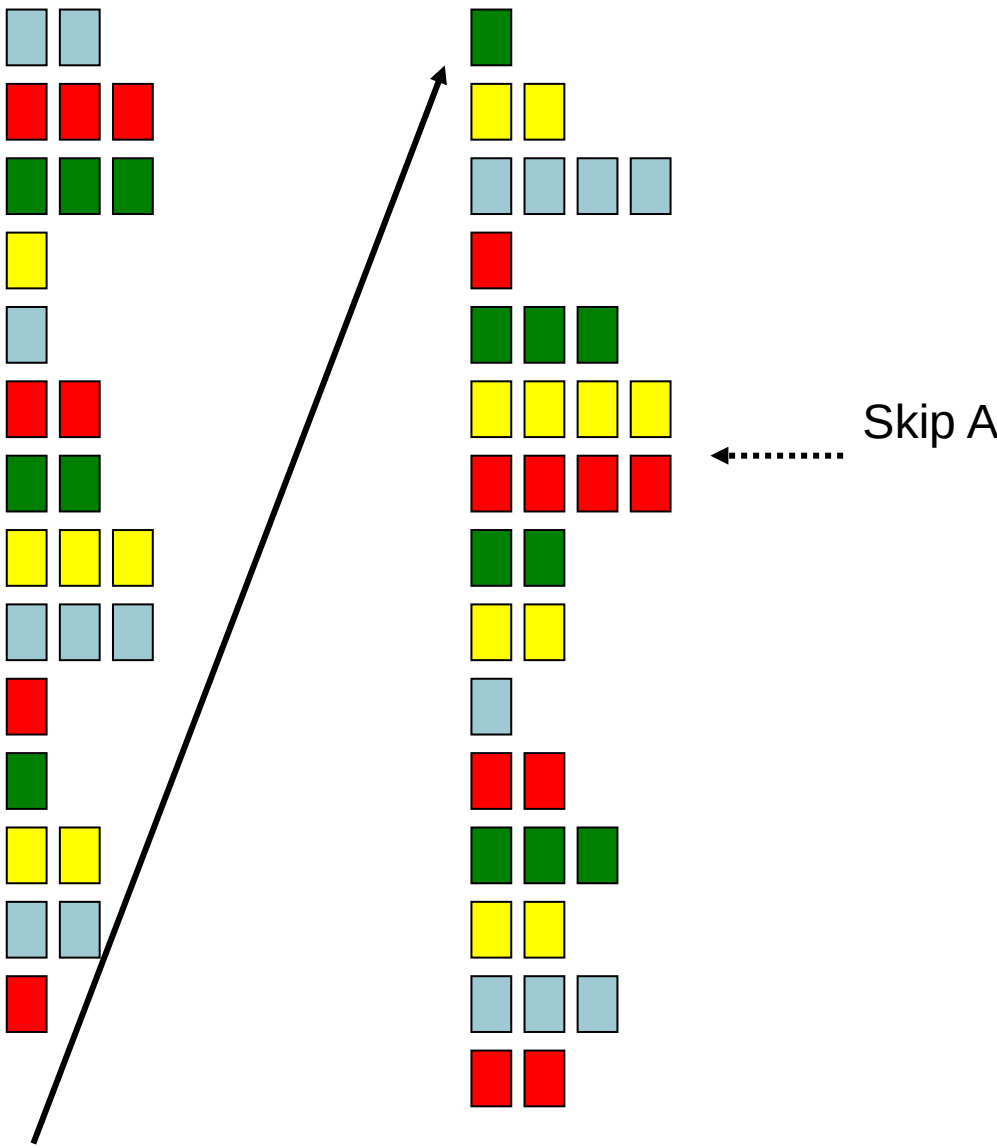
Pipeline fill time requires several cycles!

- Does not cover short stalls
- Slow execution of a single thread



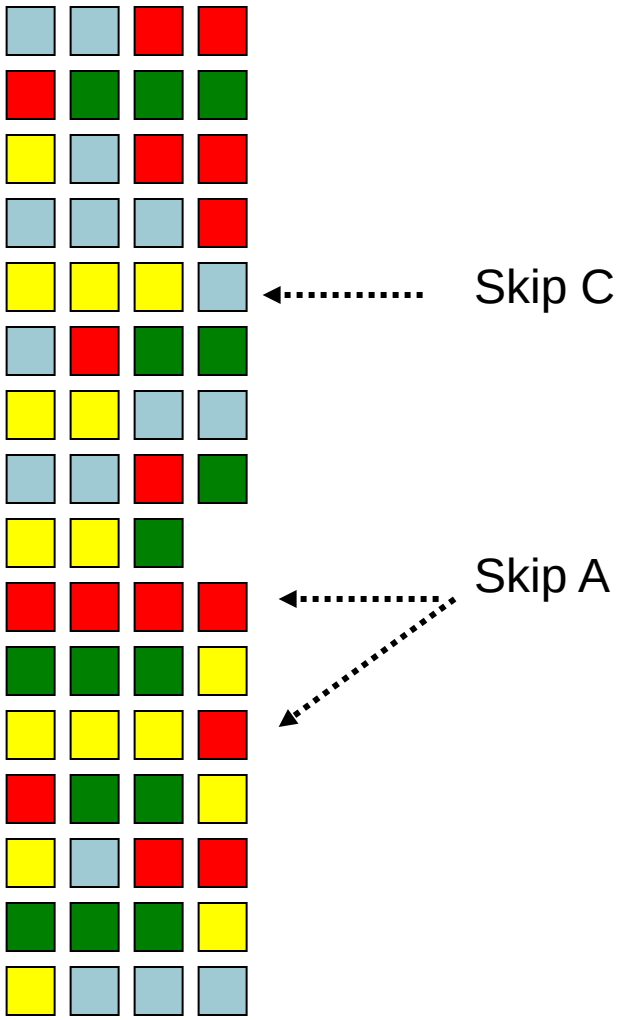
Fine Multithreading

- Two or more threads interleave instructions
  - Round-ribbon fashion and skip stalled threads
- Needs hardware support
- Hides delays: Data hazards, Cache misses
- Pipeline runs with rare stalls
- Does not make full use of multi-issue architecture

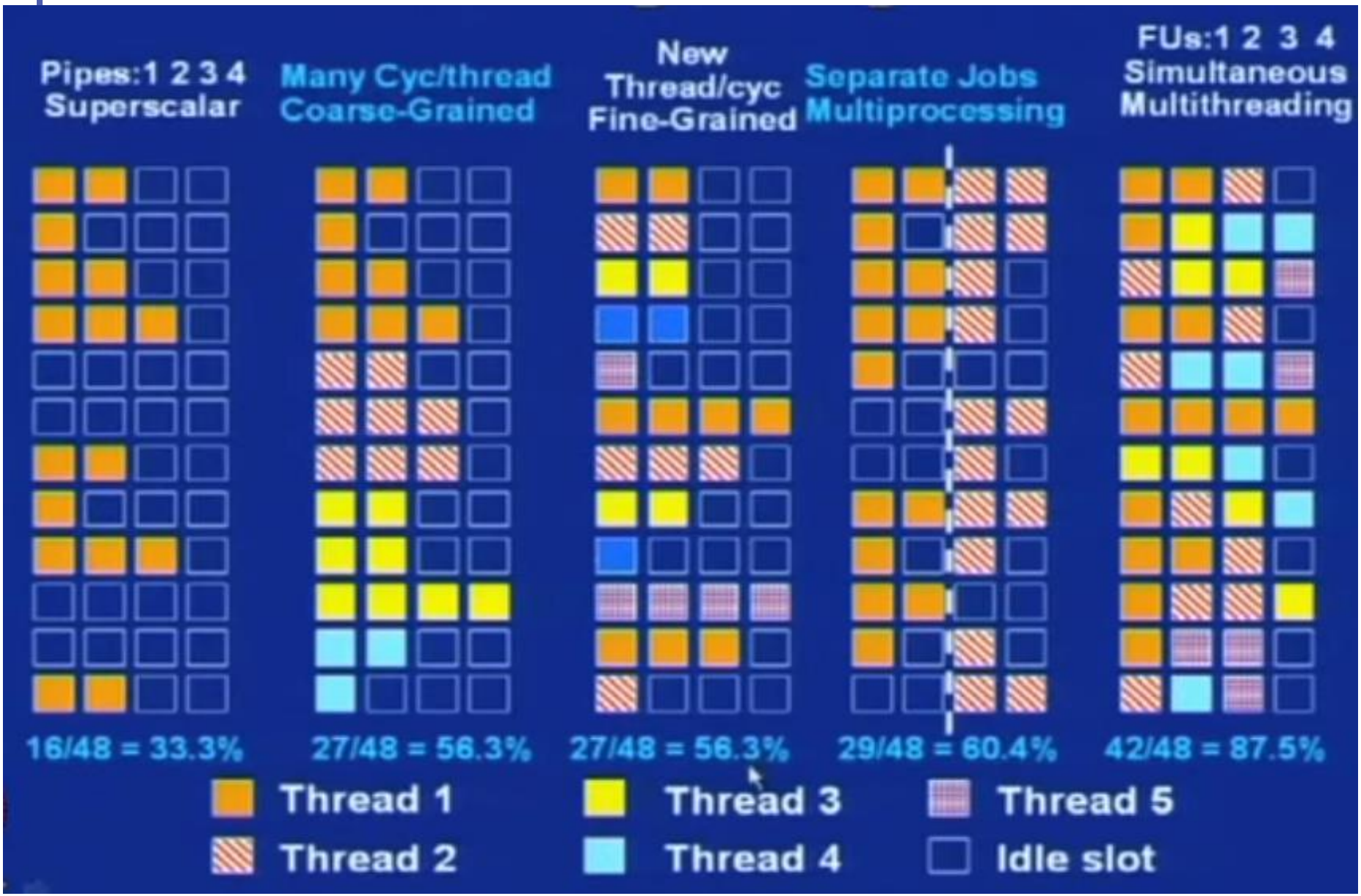


Simultaneous Multithreading (Hyper-threading)

- Instructions from multiple threads issued on same cycle
  - **Uses register renaming** and dynamic scheduling facility of **multi-issue architecture**
- Needs more hardware support
  - Register files, PC's for each thread
  - Temporary result registers before commit
  - Support to sort out which threads get results from which instructions
- Maximizes utilization of execution units



Multithreading Categories



Basic Principles

- Systems provided of reconfigurable logic are often called Reconfigurable Instruction Set Processors (**RISP**). **The reconfigurable logic includes a set of programmable processing units, which can be reconfigured in the field to implement logic operations or functions**, **and programmable interconnections between them.**

FPGAs Are Everywhere

Applications are found in virtually all industry segments:

Aerospace and defense
Medical electronics
Automotive control
Software-defined radio
Encoding and decoding

Basic steps in a reconfigurable system