معماری کامپیوتر

فصل ده

موازی‌سازی

Computer Architecture

Chapter Ten

Parallelism

Computer Architecture

Copyright Notice

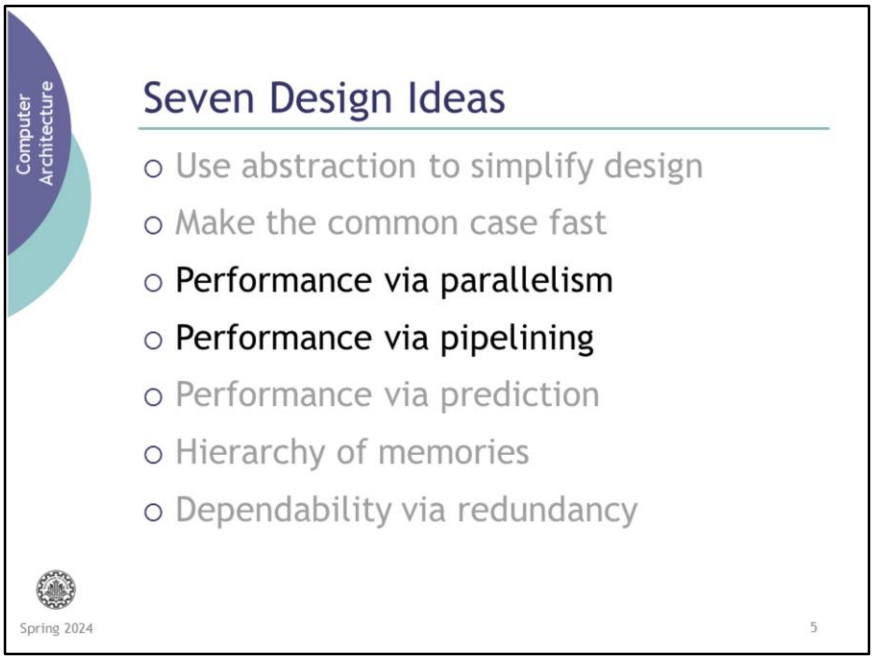
Parts (text & figures of this lecture are adopted from:

- M. Mano, "Computer System Architecture", Pearson, 1999
- D. Patterson & J. Hennessey, "Computer Organization & Design, The Hardware/Software Interface", 6th Ed., MK publishing, 2020
- W. Stallings, "Computer Organization and Architecture, Designing for Performance", 10th Ed., Pearson, 2016
- A. Tanenbaum, "Structured Computer Organization", 5th Ed., Pearson, 2006

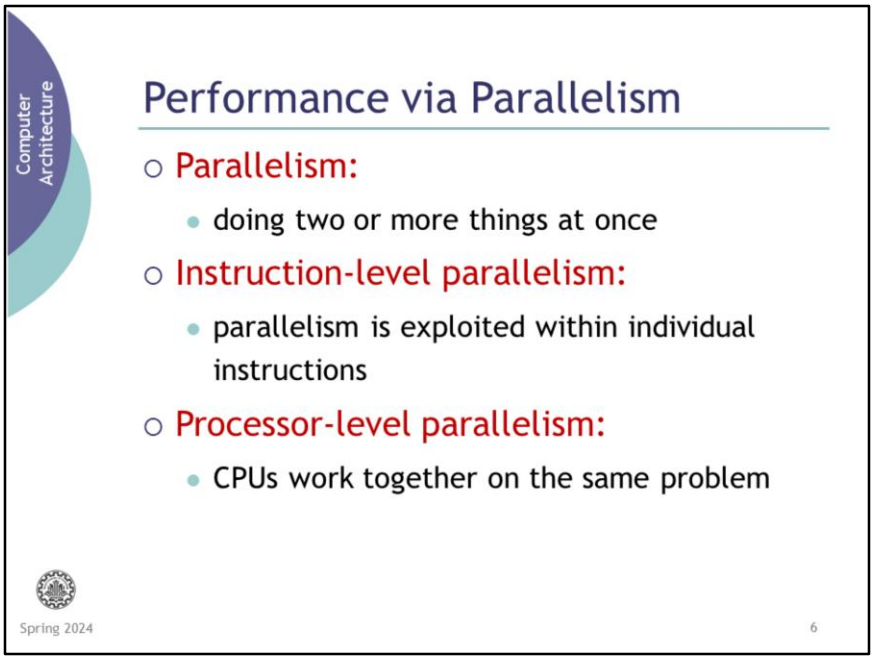
Computer Architecture

Outlines

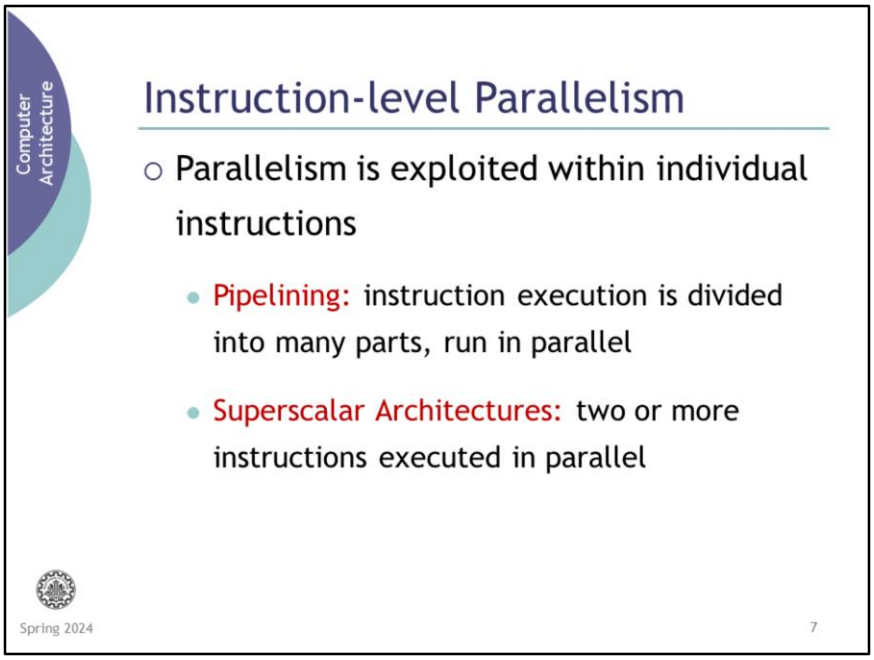
- Instruction-level Parallelism
- Processor-level Parallelism
- Flynn Classification
- HW vs. SW Parallelism



Energy efficiency has replaced die area as the most critical resource of microprocessor design. Conserving power while trying to increase performance has forced the hardware industry to switch to multicore microprocessors, thereby forcing the software industry to switch to programming parallel hardware. **Parallelism** is now required for performance. [Patterson2020]



[Tanenbaum2006]

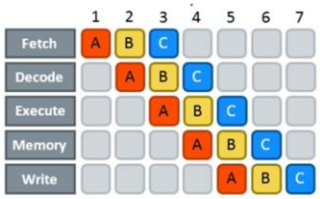


[Tanenbaum2006]

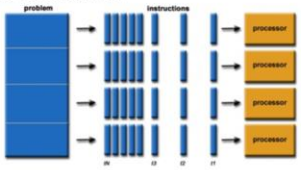
Computer Architecture

Temporal vs. Spatial Parallelism

- Temporal parallelism (Pipelining)



- Spatial parallelism



Computer Architecture

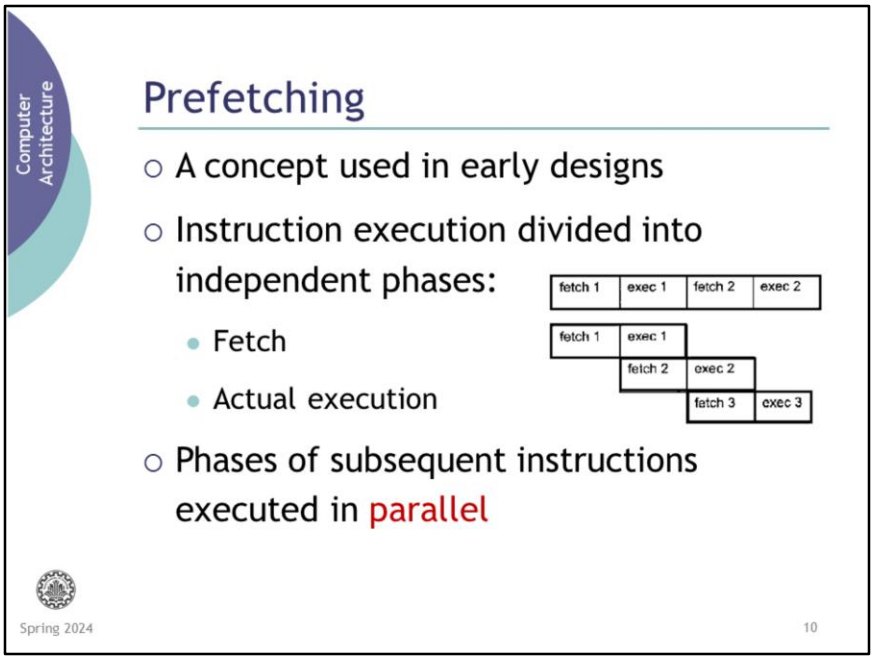
Temporal vs. Spatial Parallelism

- Temporal parallelism (Pipelining)

  - A task is broken into stages, like an assembly line
  - Multiple tasks can be spread across the stages
  - Each task must pass through all stages, but a different task will be in each stage at any given time so multiple tasks can overlap

- Spatial parallelism

  - Multiple copies of the hardware are provided so that multiple tasks can be done at the same time



It has been known for years that the actual fetching of instructions from memory is a major bottleneck in instruction execution speed. To alleviate this problem, computers going back at least as far as the IBM Stretch (1959) have had the ability to fetch instructions from memory in advance, so they would be there when they were needed.

These instructions were stored in a special set of registers called the **prefetch buffer**. This way, when an instruction was needed, it could usually be taken from the prefetch buffer rather than waiting for a memory read to complete.

In effect, prefetching divides instruction execution into two parts: fetching and actual execution. The concept of a **pipeline** carries this strategy much further. Instead of being divided into only two parts, instruction execution is often divided into many (often a dozen or more) parts, each one handled by a dedicated piece of hardware, all of which can run in parallel. [Tanenbaum2006]

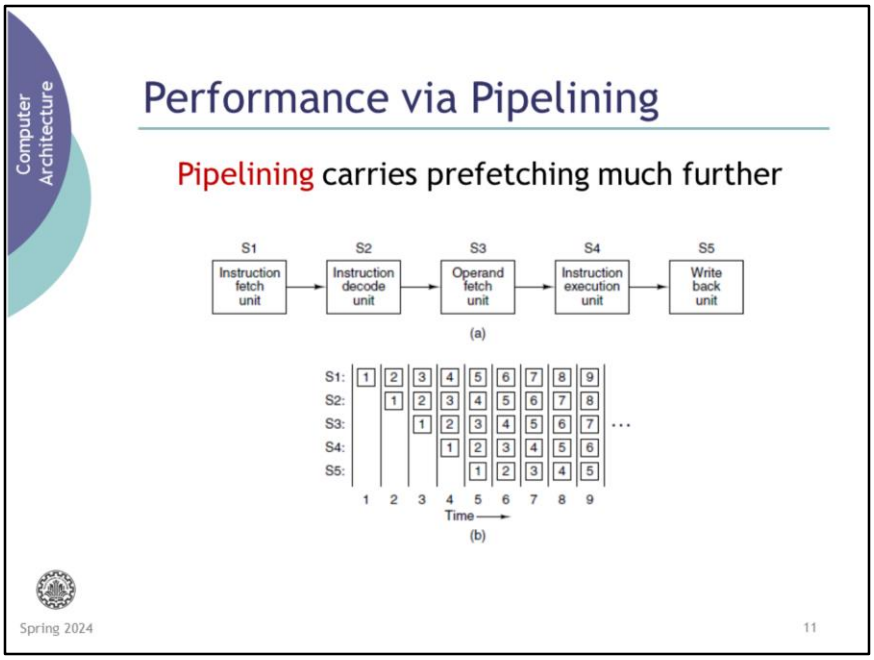
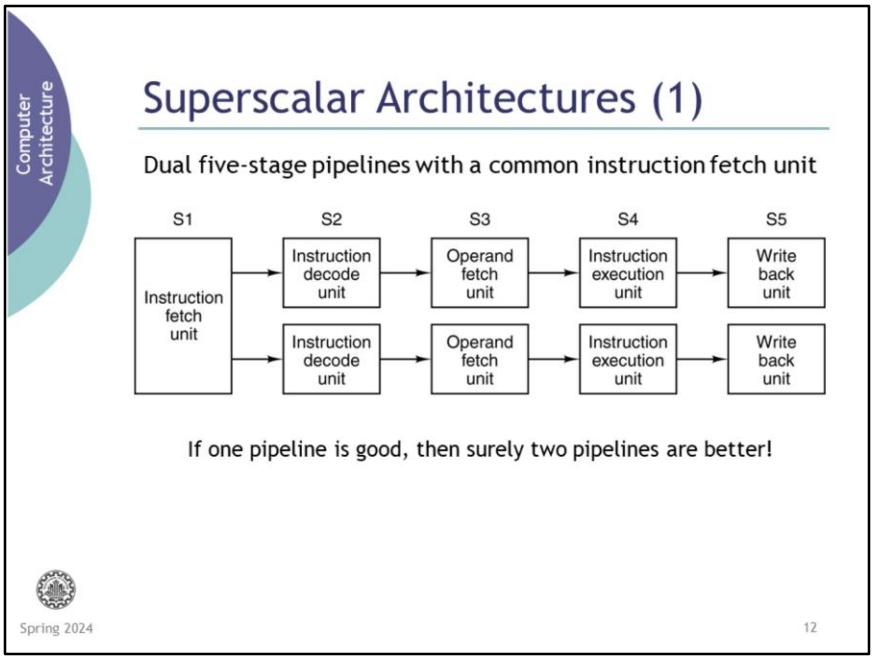
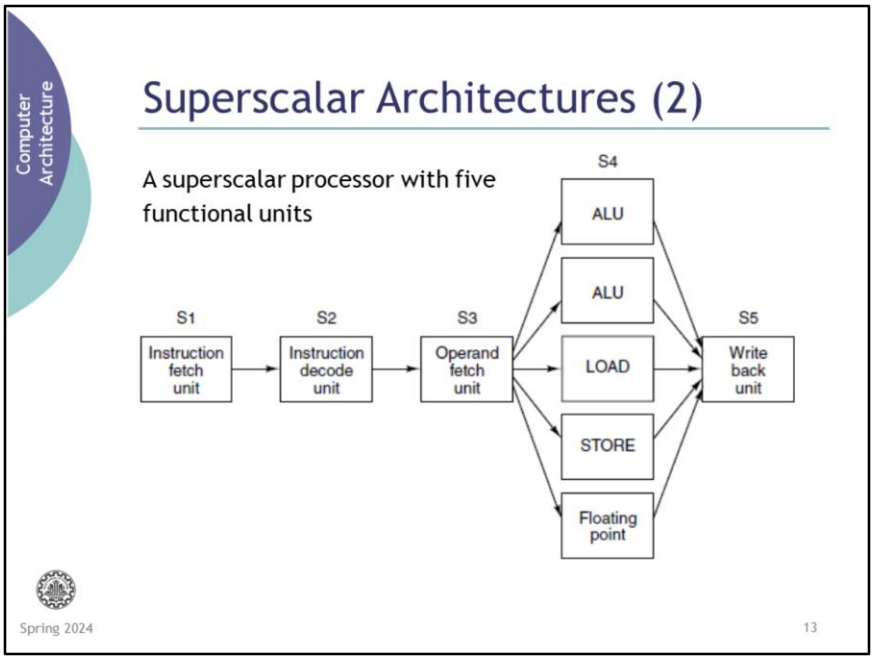


Figure 2-4. (a) A five-stage pipeline. (b) The state of each stage as a function of time. Nine clock cycles are illustrated. [Tanenbaum2006]

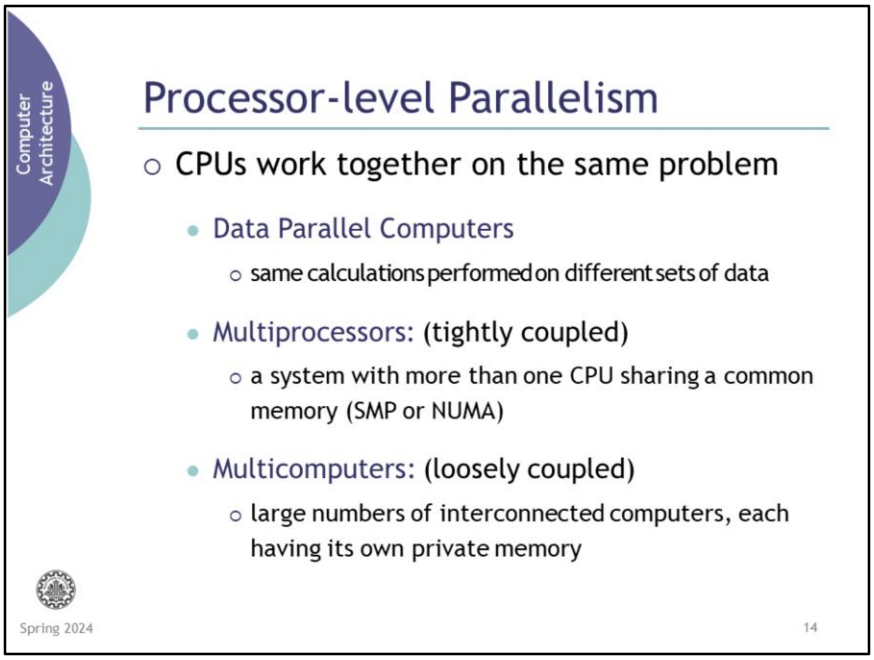


Here a single instruction fetch unit fetches pairs of instructions together and puts each one into its own pipeline, complete with its own ALU for parallel operation. To be able to run in parallel, the two instructions must not conflict over resource usage (e.g., registers), and neither must depend on the result of the other. [Tanenbaum2006]



The basic idea here is to have just a single pipeline but give it multiple functional units. For example, the Intel Core architecture has a structure similar to this figure. Implicit in the idea of a superscalar processor is that the S3 stage can issue instructions considerably faster than the S4 stage is able to execute them.

The term **superscalar architecture** was coined for this approach in 1987 (Agerwala and Cocke, 1987). Its roots, however, go back more than 40 years to the CDC 6600 computer. The 6600 fetched an instruction every 100 nsec and passed it off to one of 10 functional units for parallel execution while the CPU went off to get the next instruction. [Tanenbaum2006]

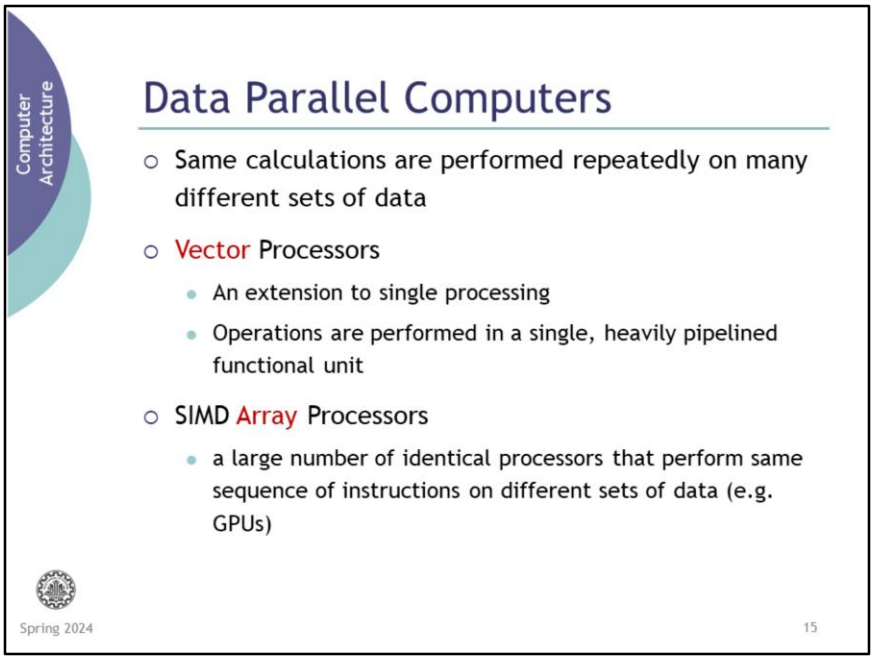


While CPUs keep getting faster, eventually they are going to run into the problems with the speed of light. Faster chips also produce more heat, whose dissipation is a huge problem.

Instruction-level parallelism helps a little, but pipelining and superscalar operation rarely win more than a factor of five or ten. To get gains of 50, 100, or more, the only way is to design computers with multiple CPUs. [Tanenbaum2006]

SMP: Symmetric MultiProcesors

NUMA: NonUniform Memory Access



Data Parallel Computers: [Tanenbaum2006]

SIMD processors: consist of a large number of identical processors that perform the same sequence of instructions on different sets of data. (e.g. GPUs)

Vector processors: unlike a SIMD processor, all of the operations are performed in a single, heavily pipelined functional unit

Computer Architecture

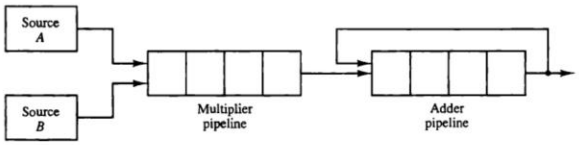
Vector Processors (example)

Figure 9-11 Instruction format for vector processor.

Operation code	Base address source 1	Base address source 2	Base address destination	Vector length

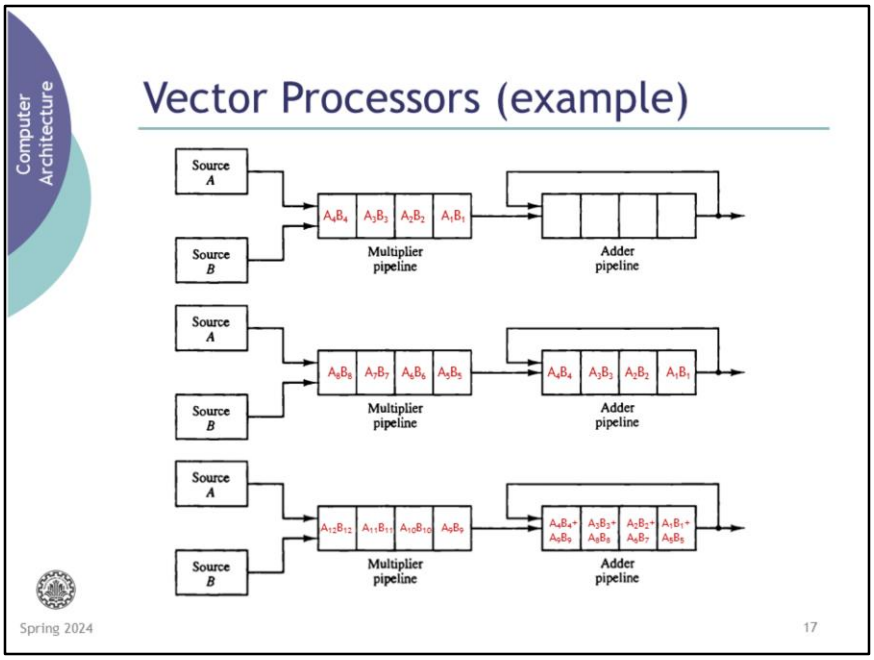
Sample vector instruction: `C(1:100)=A(1:100)+B(1:100)`

Figure 9-12 Pipeline for calculating an inner product.



Inner product calculation: $C=A_1B_1+A_2B_2+...+A_kB_K$

In a vector processor, the operation is specified with a single vector instruction and the instruction is executed in an arithmetic pipeline. [Mano1999]



In a vector processor, the operation is specified with a single vector instruction and the instruction is executed in an arithmetic pipeline. [Mano1999]

Computer Architecture

SIMD Array Processors

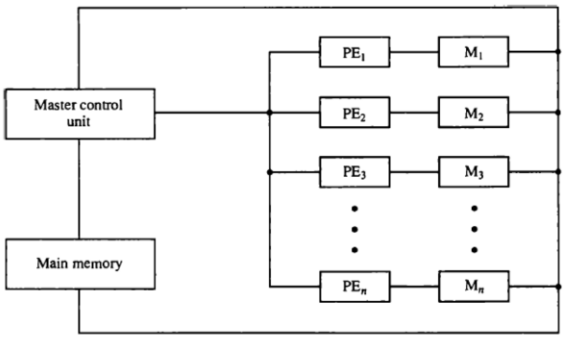


Figure 9-15 SIMD array processor organization.

An SIMD array processor is a computer with multiple processing units operating in parallel. The processing units are synchronized to perform the same operation under the control of a common control unit, thus providing a single instruction stream, multiple data stream (SIMD) organization. [Mano1999]

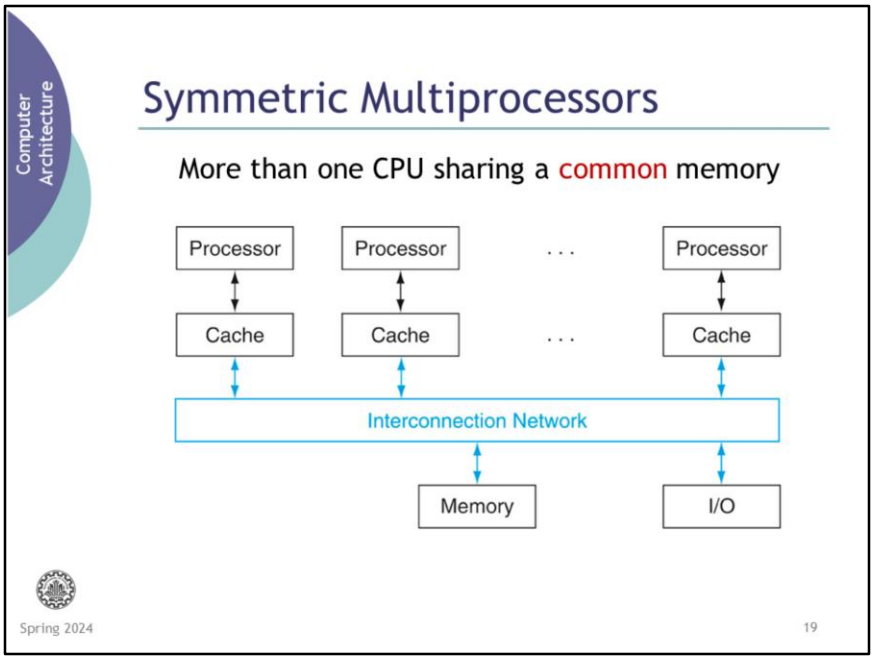


FIGURE 6.7 Classic organization of a shared memory multiprocessor. [Patterson2020]

SMP (Symmetric MultiProcessors): [Stallings2016]

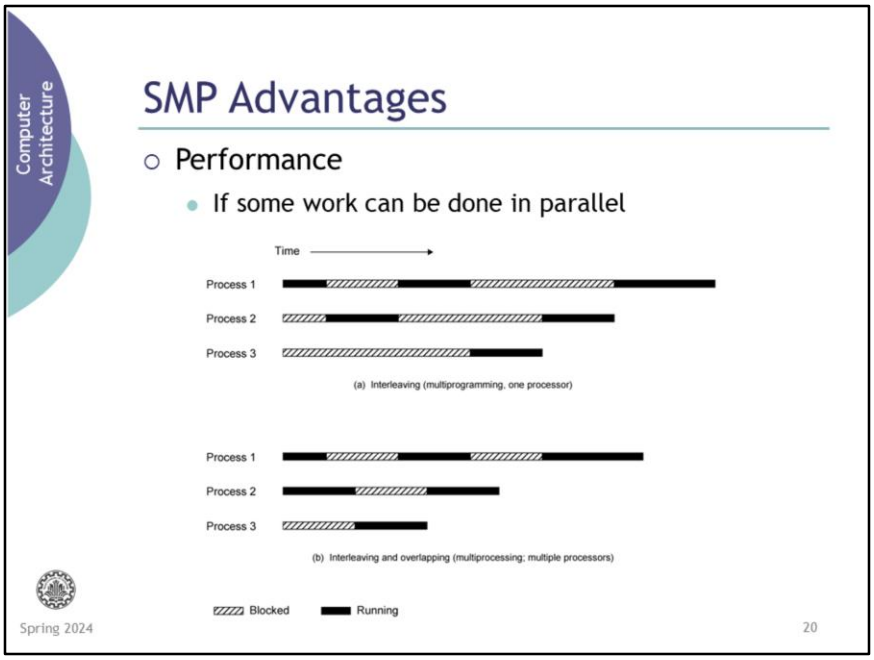
1. There are two or more similar processors of comparable capability.

2. These processors share the same main memory and I/O facilities and are interconnected by a bus or other internal connection scheme, such that memory access time is approximately the same for each processor.

3. All processors share access to I/O devices, either through the same channels or through different channels that provide paths to the same device.

4. All processors can perform the same functions (hence the term *symmetric*).

5. The system is controlled by an integrated operating system that provides interaction between processors and their programs at the job, task, file, and data element levels.



Multiprogramming & Multiprocessing:

If the work to be done by a computer can be organized so that some portions of the work can be done in parallel, then a system with multiple processors will yield greater performance than one with a single processor of the same type

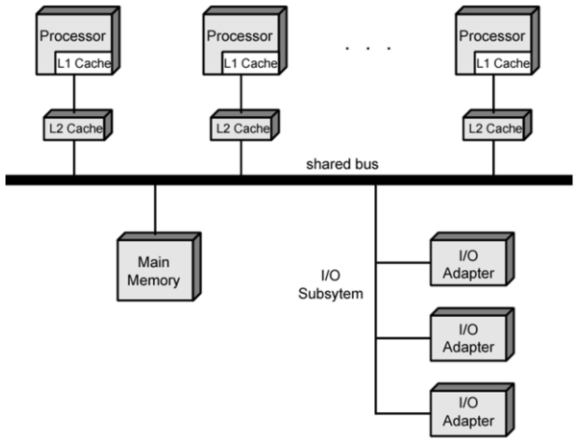
Computer Architecture

SMP Advantages

- Performance
  - If some work can be done in parallel
- Availability
  - Since all processors can perform the same functions, failure of a single processor does not halt the system
- Incremental growth
  - User can enhance performance by adding additional processors
- Scaling
  - Vendors can offer range of products based on number of processors

Computer Architecture

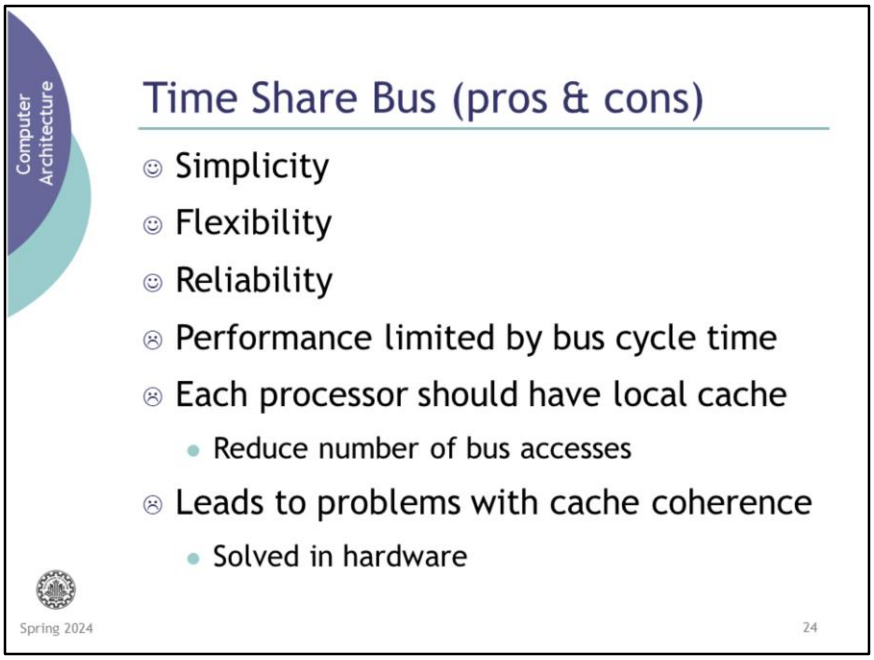
Symmetric Multiprocessor Organization



Computer Architecture

Time Shared Bus

- Simplest form
- Structure and interface similar to single processor system
- Following features provided
  - Addressing: distinguish modules on bus
  - Arbitration: any module can be temporary master
  - Time sharing: if one module has the bus, others must wait and may have to suspend
- Now have multiple processors as well as multiple I/O modules



Simplicity: This is the simplest approach to multiprocessor organization. The physical interface and the addressing, arbitration, and time-sharing logic of each processor remain the same as in a single-processor system.

Flexibility: It is generally easy to expand the system by attaching more processors to the bus.

Reliability: The bus is essentially a passive medium, and the failure of any attached device should not cause failure of the whole system.

The **cache coherence** problem: If a word is altered in one cache, it could conceivably invalidate a word in another cache. To prevent this, the other processors must be alerted that an update has taken place

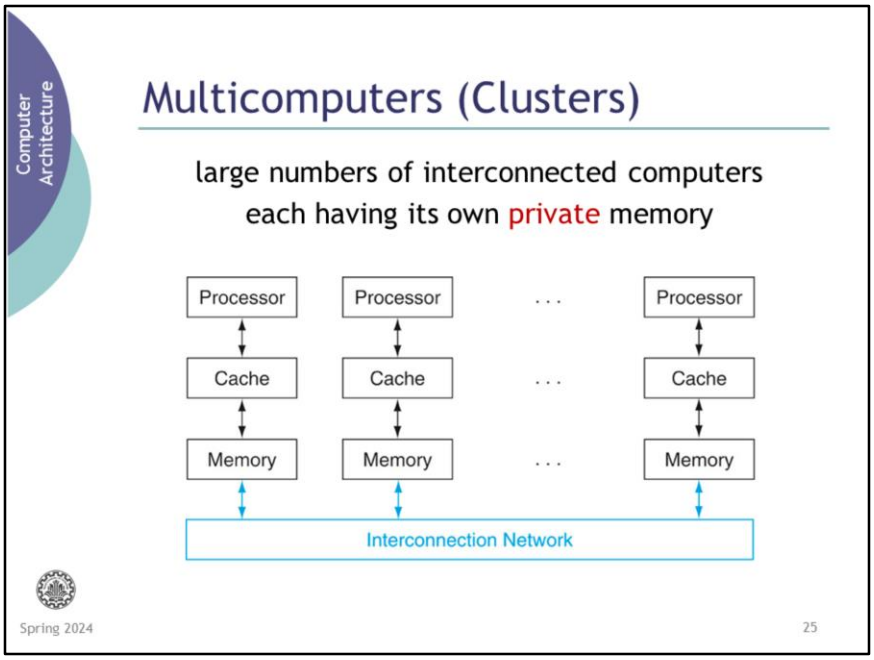


FIGURE 6.13 Classic organization of a multiprocessor with multiple private address spaces, traditionally called a message-passing multiprocessor.

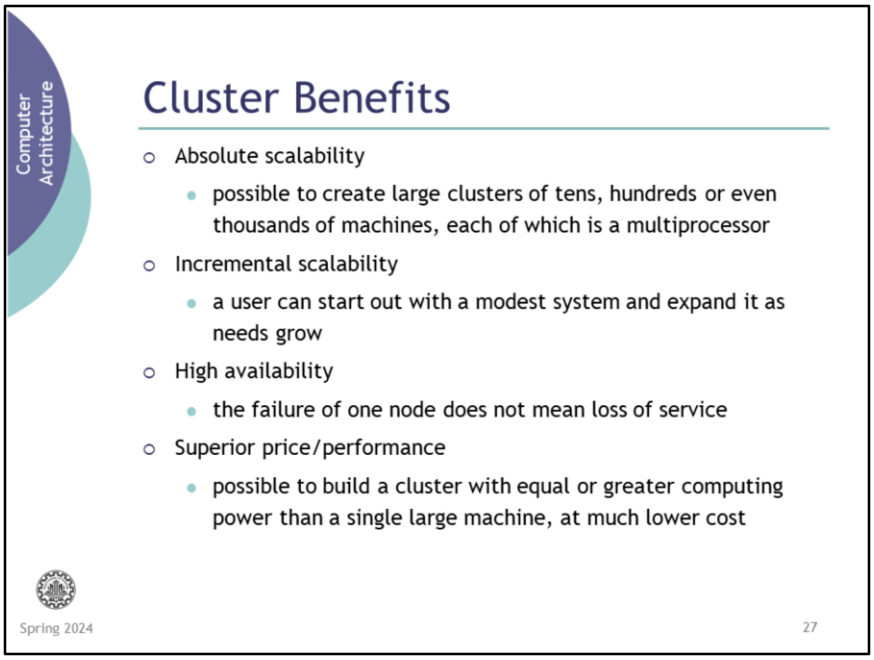
Note that unlike the SMP in Figure 6.7, the interconnection network is not between the caches and memory but is instead between processor-memory nodes. [Patterson2020]

Computer Architecture

Clusters

- Alternative to SMP
- High performance
- High availability
- Server applications

- A group of interconnected whole computers
- Working together as unified resource
- Illusion of being one machine
- Each computer called a node

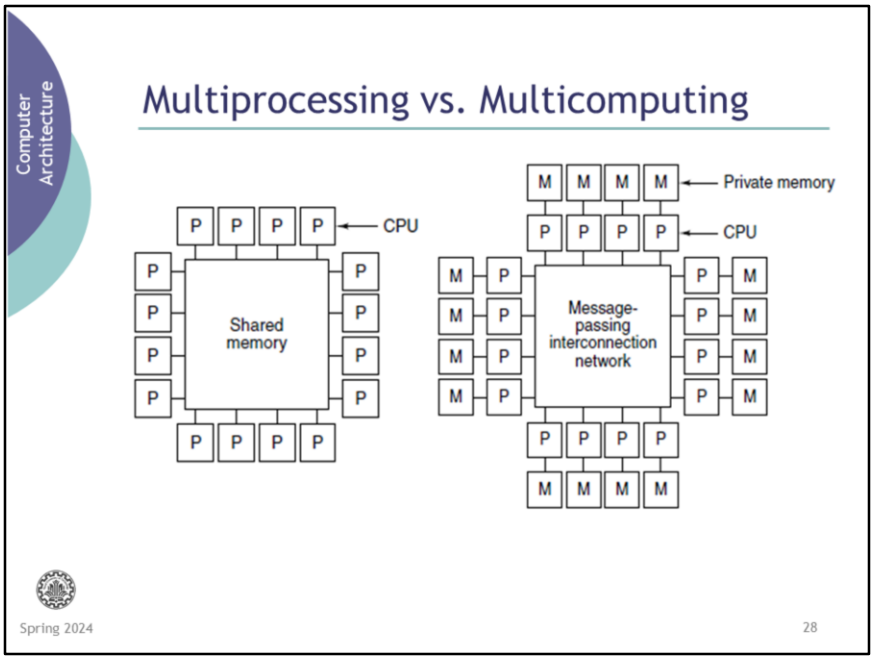


Absolute scalability: It is possible to create large clusters that far surpass the power of even the largest standalone machines. A cluster can have tens, hundreds, or even thousands of machines, each of which is a multiprocessor.

Incremental scalability: A cluster is configured in such a way that it is possible to add new systems to the cluster in small increments. Thus, a user can start out with a modest system and expand it as needs grow, without having to go through a major upgrade in which an existing small system is replaced with a larger system.

High availability: Because each node in a cluster is a standalone computer, the failure of one node does not mean loss of service. In many products, fault tolerance is handled automatically in software.

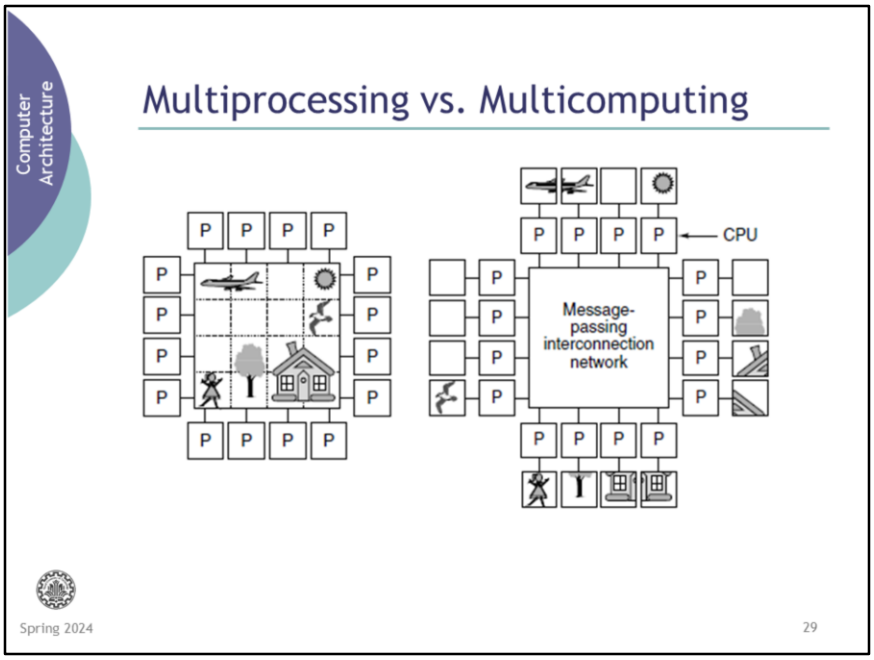
Superior price/performance: By using commodity building blocks, it is possible to put together a cluster with equal or greater computing power than a single large machine, at much lower cost.



[Tanenbaum2006]

Right: A multiprocessor with 16 CPUs sharing a common memory.

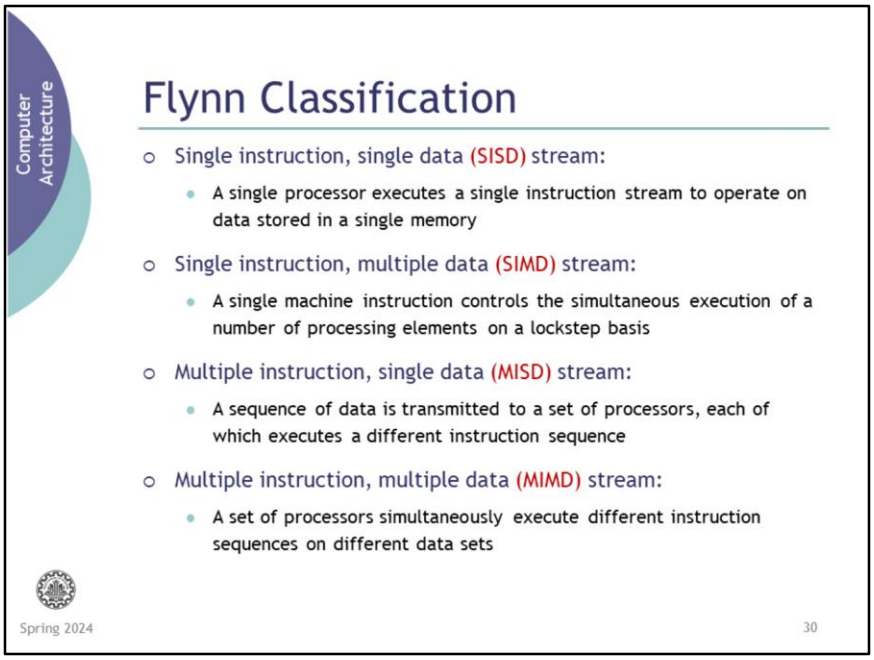
Left: A multicomputer with 16 CPUs, each with its own private memory.



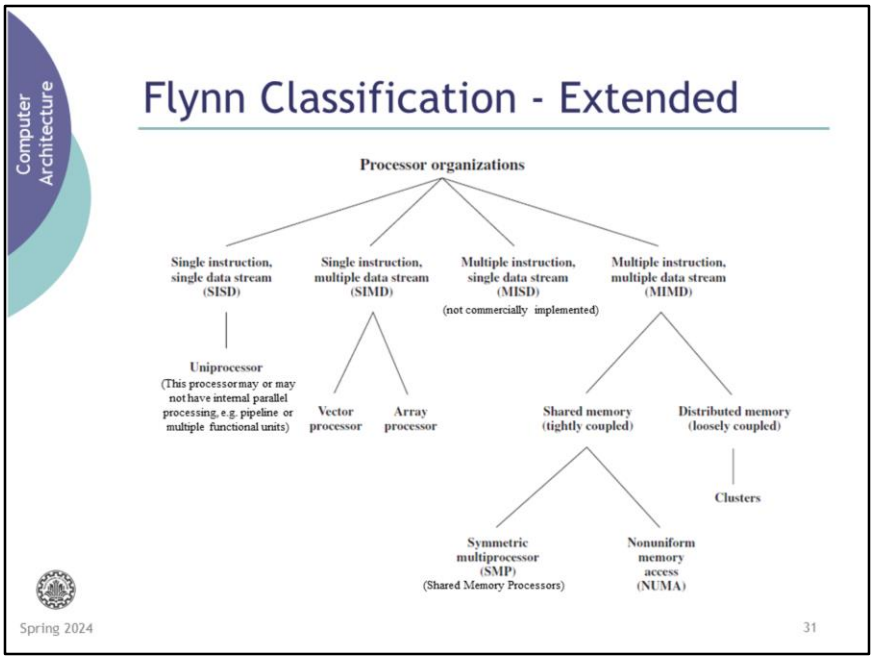
[Tanenbaum2006]

Right: An image partitioned into 16 sections, each being analyzed by a different CPU.

Left: The bit-map image split up among the 16 memories



[Stallings2016]

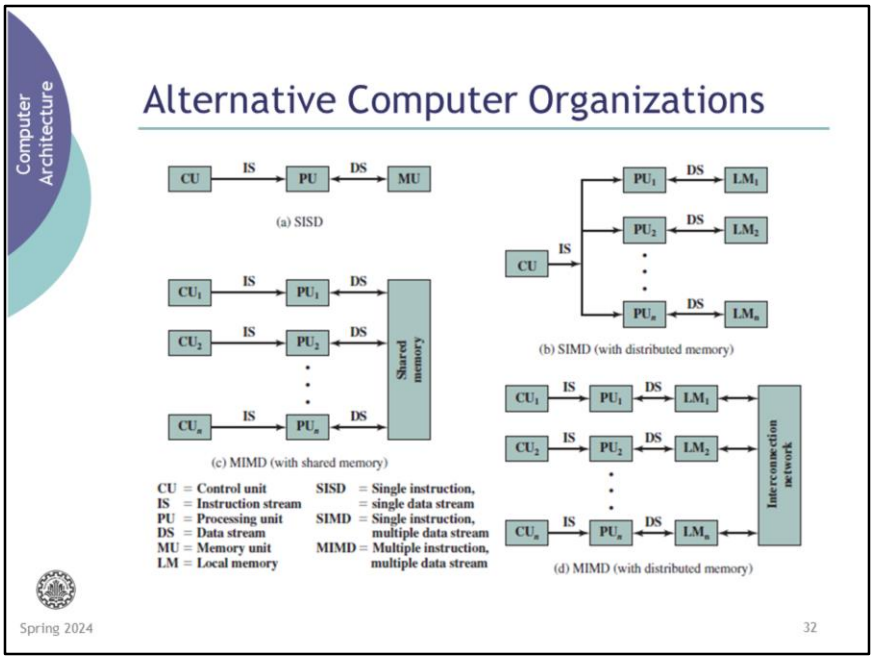


[Stallings2016]

MIMDs can be subdivided by the means in which the processors Communicate:

- Multiprocessors:
    - If the processors share a common memory, then each processor accesses programs and data stored in the shared memory, and processors communicate with each other via that memory. The most common form of such system is known as a **symmetric multiprocessor (SMP)**, in which, multiple processors share a single memory or pool of memory by means of a shared bus or other interconnection mechanism; a distinguishing feature is that the memory access time to any region of memory is approximately the same for each processor.
    - A more recent development is the **nonuniform memory access (NUMA)** organization, in which the memory access time to different regions of memory may differ.
- Multicomputers:
    - A collection of independent uniprocessors or SMPs may be interconnected to form a **cluster**. Communication among the computers is either via fixed

paths or via some network facility.

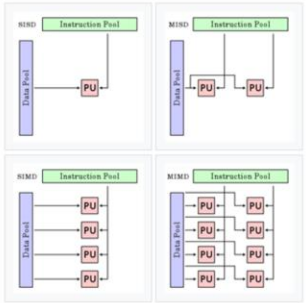


[Stallings2016]

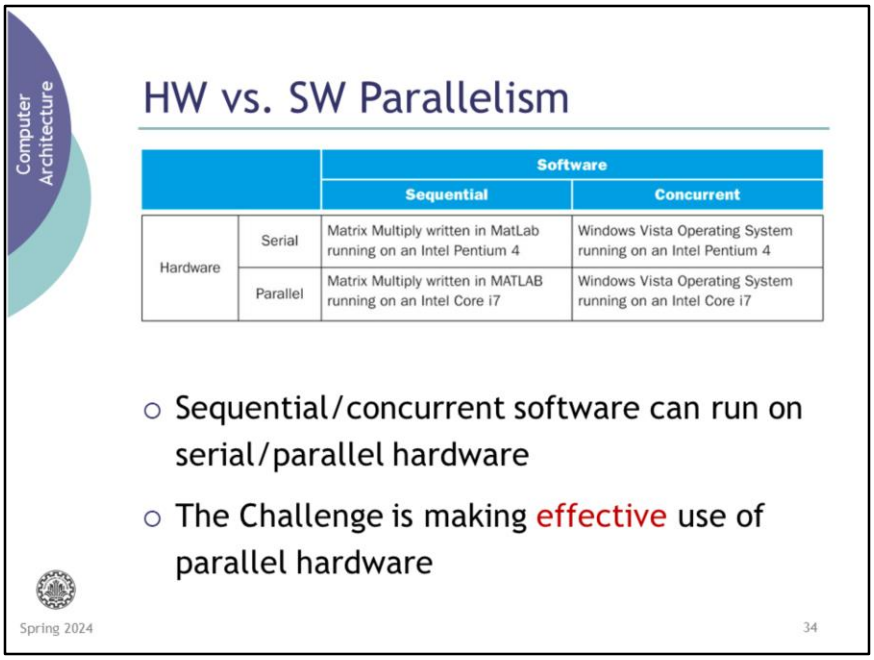
Computer Architecture

Flynn Classification Summary

		Data Streams	
		Single	**Multiple**
Instruction Streams	Single	SISD: Intel Pentium 4	SIMD: SSE instructions of x86
	Multiple	MISD: No examples today	MIMD: Intel Core i7

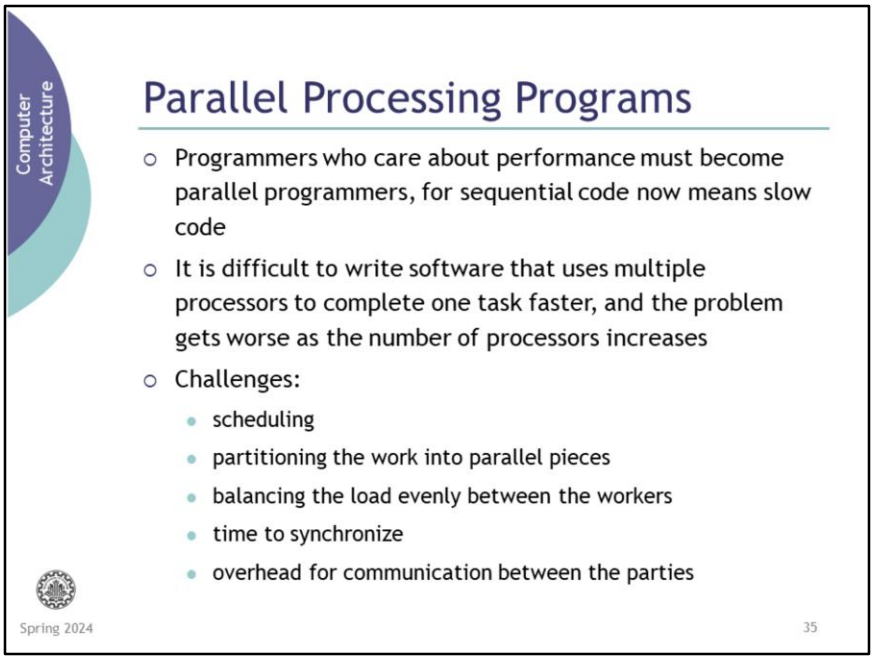


[Stallings2016]



The columns of this figure represent the software, which is either inherently sequential or concurrent.

The rows of the figure represent the hardware, which is either serial or parallel. [Patterson2020]



[Patterson2020]

Computer Architecture

Parallelism (Summary)

- Parallelism: doing two or more things at once
- Instruction-level parallelism: parallelism is exploited within individual instructions
  - Pipelining: instruction execution is divided into many parts, run in parallel
  - Superscalar Architectures: two or more instructions executed in parallel
- Processor-level parallelism: CPUs work together on the same problem
  - Data Parallel Computers:
    - same calculations performed on different sets of data
  - Multiprocessors: (tightly coupled)
    - a system with more than one CPU sharing a common memory
  - Multicomputers: (loosely coupled)
    - large numbers of interconnected computers, each having its own private memory