Parallel and Distributed Computing CS3006 (BDS-6A) Lecture 03

Instructor: Dr. Syed Mohammad Irteza

Assistant Professor, Department of Computer Science, FAST

07 February, 2023

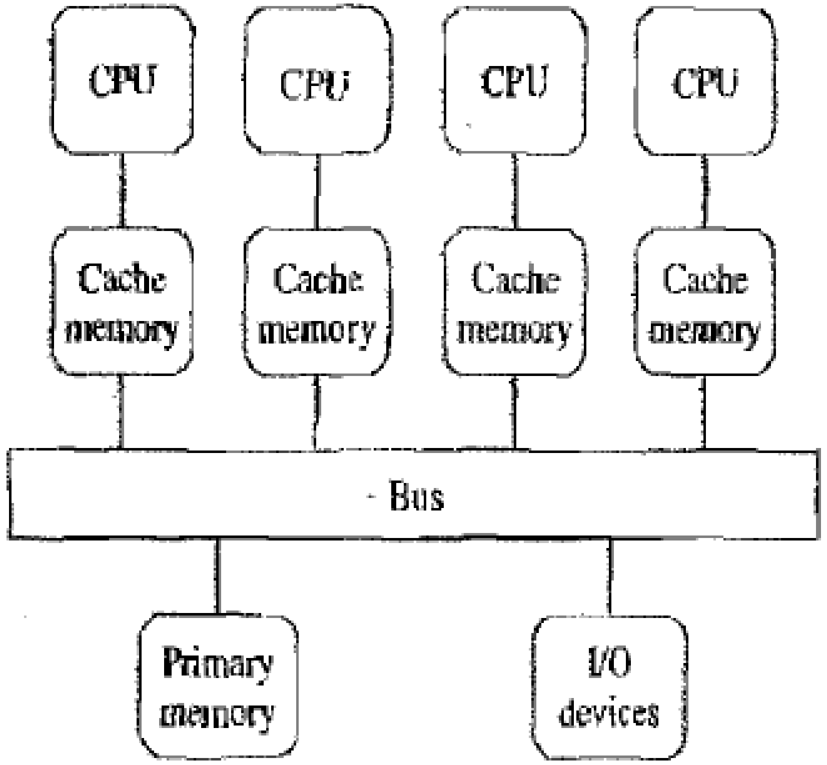
Previous Lecture

- Distributed Computing v. Distributed Systems
- Parallel Computing vs. Distributed Computing
- Practical Applications of Parallel and Distributed Computing
- Limitations
- Amdahl’s Law
- Karp-Flatt Metric
- Types of Parallelism: data, functional, pipelining
- Multi-processors (centralized and distributed)

Multi-Processor

i. **Centralized Multi-processor**

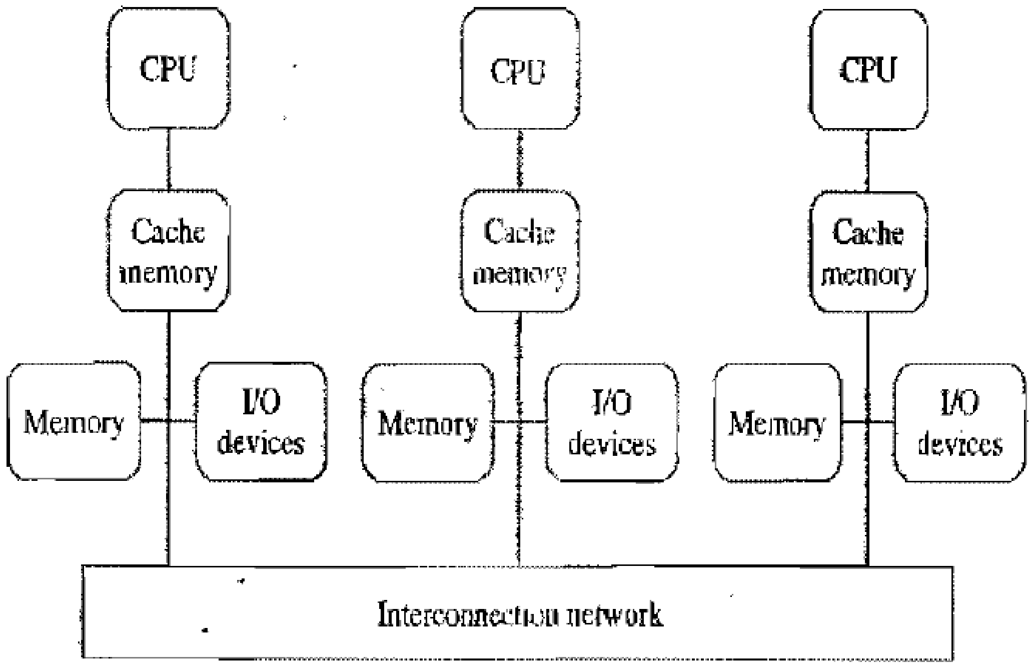
- Additional CPUs are attached to the system bus, and all the processors share the same primary memory
- All the memory is at one place and has the same access time from every processor
- Also known as **UMA** (Uniform Memory Access) multi-processor or **SMP** (symmetrical Multi-processor)



Multi-Processor

ii. Distributed Multi-processor

- Distributed collection of memories forms one logical address space
- Again, the same address on different processors refers to the same memory location.
- Also known as non-uniform memory access (**NUMA**) architecture
- Because, memory access time varies significantly, depending on the physical location of the referenced address



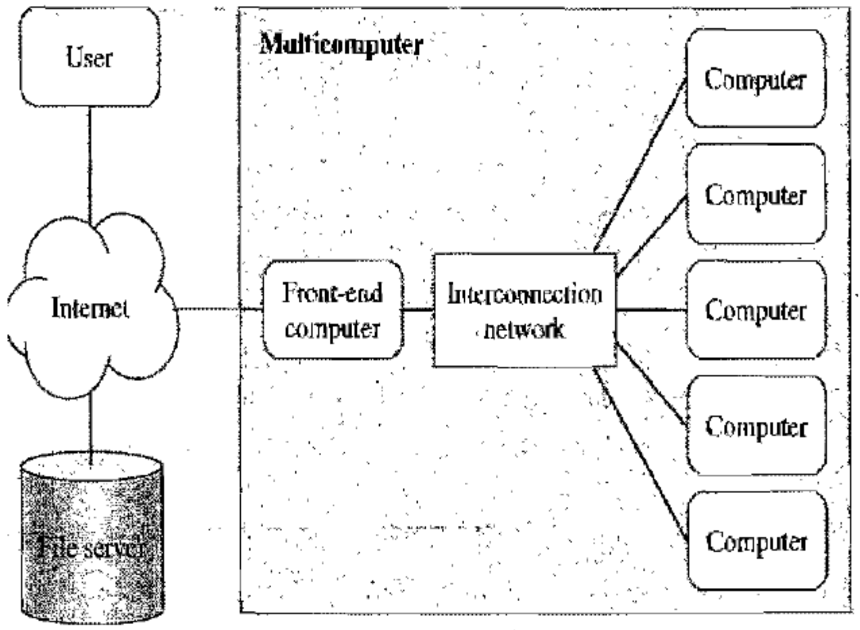
Multi-Computer

- Distributed-memory, multi-CPU computer
- Unlike **NUMA** architecture, a multicomputer has disjoint local address spaces
- Each processor has direct access to their local memory only.
- The same address on different processors refers to two different physical memory locations.
- Processors interact with each other through passing messages

Multi-Computer

Asymmetric Multi-Computers

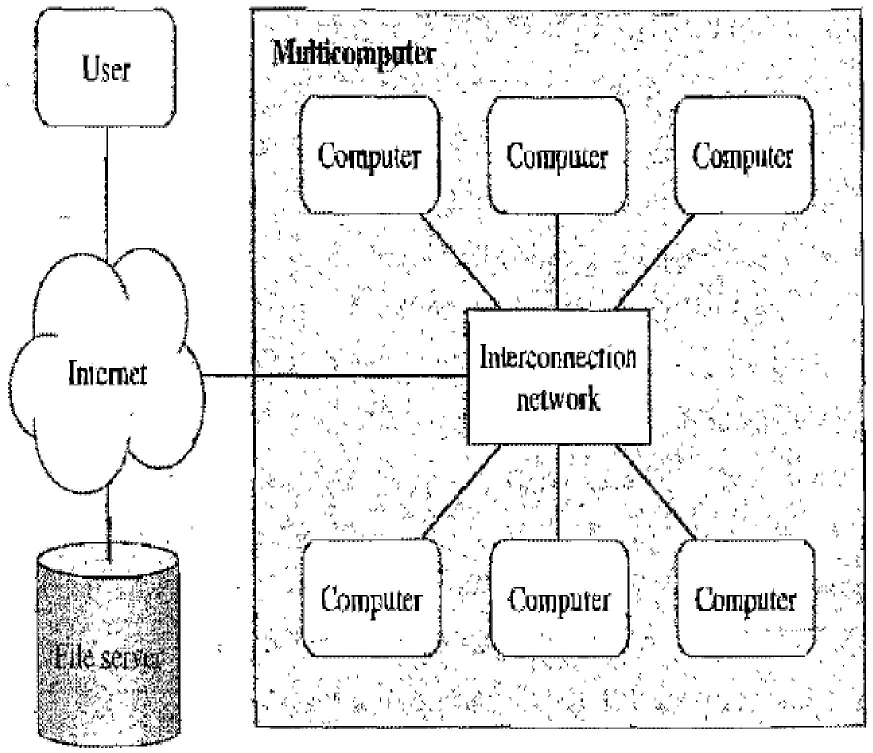
- A front-end computer that interacts with users and I/O devices
- The back-end processors are dedicatedly used for "number crunching"
- Front-end computer executes a full, multi-programmed OS and provides all functions needed for program development
- The backends are reserved for executing parallel programs



Multi-Computer

Symmetric Multi-Computers

- Every computer executes same OS
- Users may log into any of the computers
- This enables multiple users to concurrently login, edit and compile their programs.
- All the nodes can participate in execution of a parallel program



Cluster vs. Network of Workstations

Cluster	Network of workstations
Usually a co-located collection of low-cost computers and switches, dedicated to running parallel jobs. All computer run the same version of operating system.	A dispersed collection of computers. Individual workstations may have different Operating systems and executable programs
Some of the computers may not have interfaces for the users to login	User have the power to login and power off their workstations
Commodity cluster uses high speed networks for communication such as fast Ethernet@100Mbps, gigabit Ethernet@1000 Mbps and Myrinet@1920 Mbps.	Ethernet speed for this network is usually slower. Typically in the range of 10 Mbps

Parallel and Distributed Systems

- Parallel and distributed computing is going to be more and more important
  - Dual and quad core processors are very common
  - Up to six and eight cores for each CPU
  - Multithreading is growing
- Hardware structure or architecture is important for understanding how much speed up is possible beyond a single CPU
- Also, the capability of compilers to generate efficient code is very important
- It is always difficult to distinguish between HW and SW influences

Parallel Computing vs. Distributed Computing

Parallel Computing	**Distributed Computing**
Shared memory	Distributed memory
Multiprocessors	Clusters and networks of workstations
Processors share logical address spaces	Processors do not share address space
Processors share physical memory	No sharing of physical memory
Also referred to the study of parallel algorithms	Study of theoretical distributed algorithms or neural networks

Amdahl’s Law

- Scaling falls off as the number of processors increases due to lock (barriers for synchronizing masters and slaves) and memory collisions
- It is very difficult to compute (1-F)
    - serialized regions not only in our code, also in kernel and in HW
    - profiling is very important
- The private data cache of multiprocessor systems has to be kept consistent

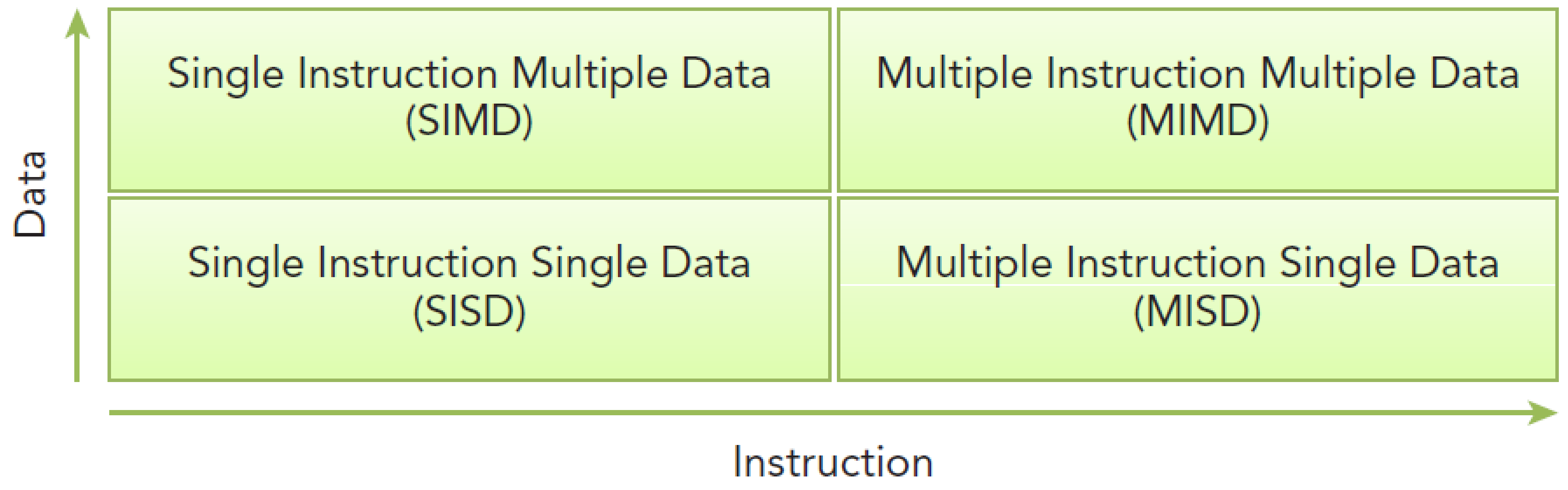
Flynn’s Taxonomy

Flynn’s Taxonomy

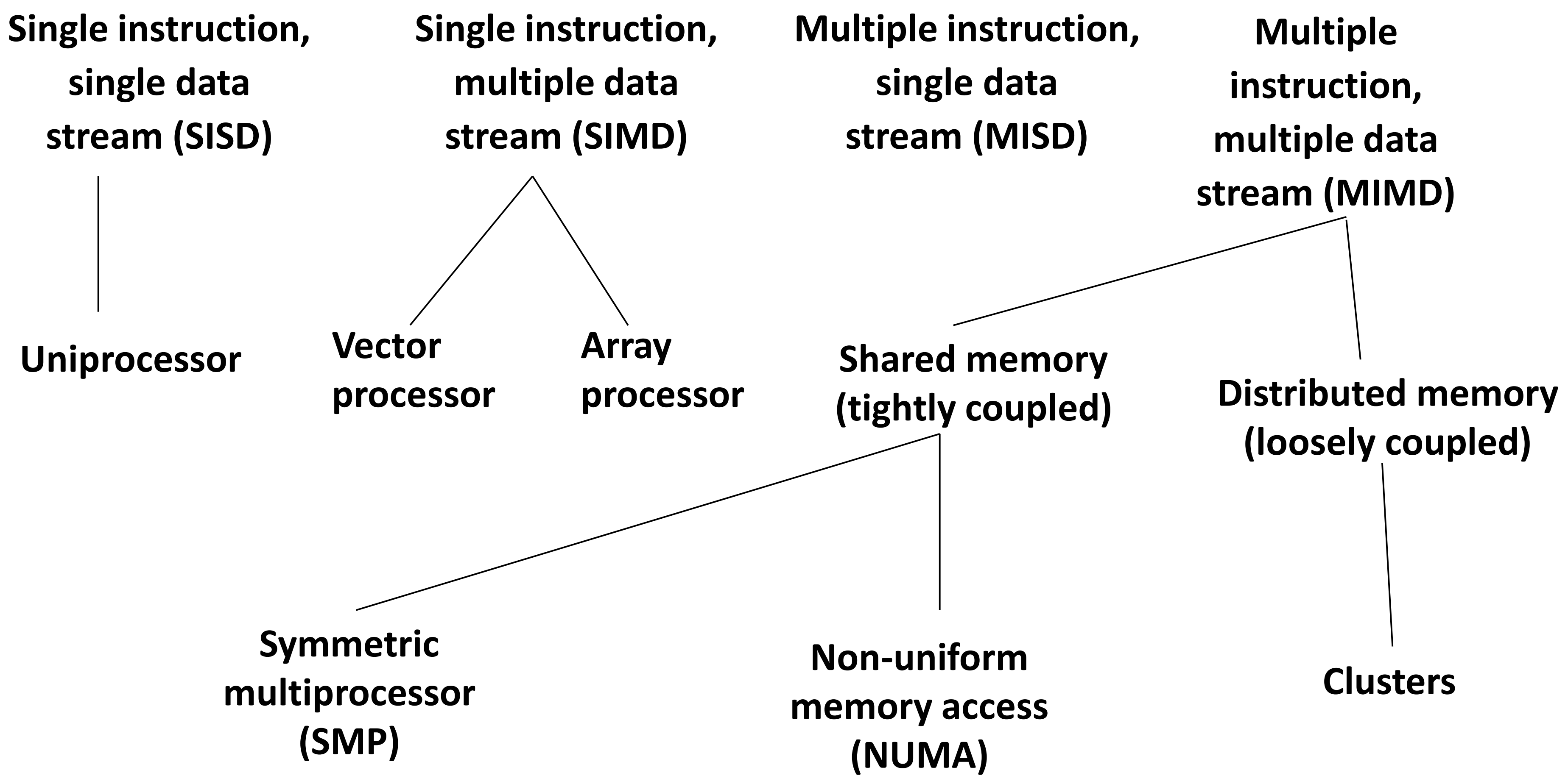
- SISD: Single instruction stream, single data stream
- SIMD: Single instruction stream, multiple data stream
- MISD: Multiple instruction stream , single data stream
- MIMD: Multiple instruction stream, multiple data stream

Flynn’s Taxonomy

- Widely used architectural classification scheme
- Classifies architectures into four types
- The classification is based on how data and instructions flow through the cores.


Single Instruction Multiple Data (SIMD)
Multiple Instruction Multiple Data (MIMD)
Single Instruction Single Data (SISD)
Multiple Instruction Single Data (MISD)
Data
Instruction

Processor Organizations

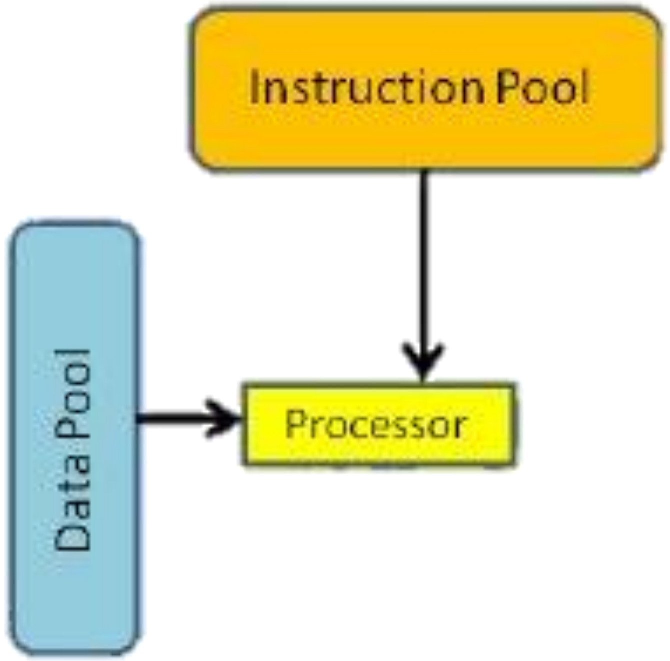


Single instruction, single data stream (SISD)
Single instruction, multiple data stream (SIMD)
Multiple instruction, single data stream (MISD)
Multiple instruction, multiple data stream (MIMD)
Uniprocessor
Vector processor
Array processor
Shared memory (tightly coupled)
Distributed memory (loosely coupled)
Symmetric multiprocessor (SMP)
Non-uniform memory access (NUMA)
Clusters

Flynn’s Taxonomy

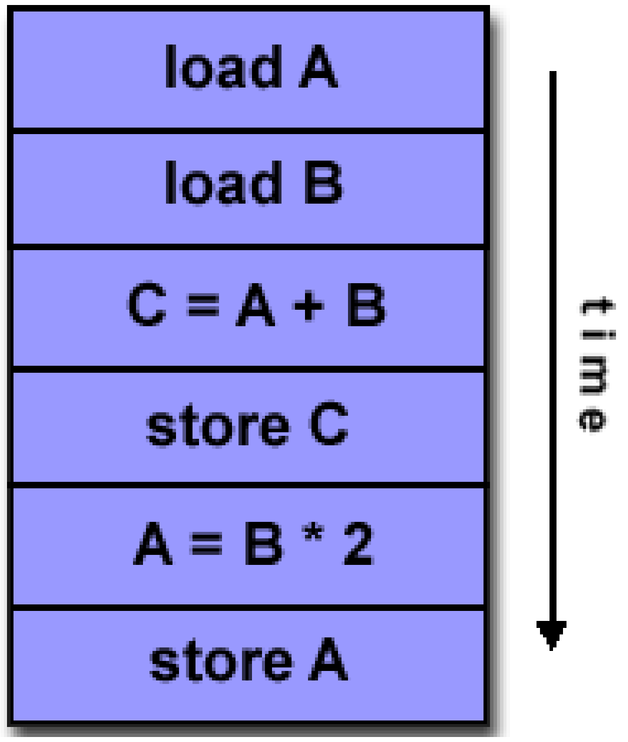
SISD (Single Instruction Single Data)

- Refers to the traditional computer: a serial architecture
- This architecture includes single core computers
- Single instruction stream is in execution at a given time
- Similarly, only one data stream is active at any time



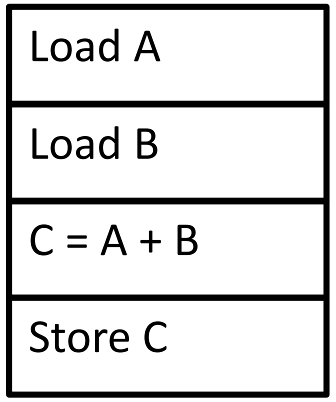
Not parallel, classical Von Neumann architecture
Parallelism can be introduced using pipelining

Example of SISD:



SISD

- A serial (non-parallel computer)
- Single instruction: one instruction per cycle
- Single data: only one data stream per cycle
- Easy and deterministic execution



Examples:

- Single CPU workstations
- Most workstations from HP, IBM and SGI are SISD machines

SISD (continued)

- Performance of a processor can be measured with:

  MIPS rate = f x IPC (instructions per cycle)
- How to increase performance:
  - increasing clock frequency
  - increasing number of instructions completed during a processor cycle (multiple pipelines in a superscalar architecture and/or out of order execution)
  - multithreading

SISD (continued)

Implicit multithreading

- concurrent execution of multiple threads extracted from a single sequential program
- Managed by processor hardware
- Improve individual application performance

Explicit multithreading

- concurrent execution of instructions from different explicit threads, either by interleaving instructions from different threads or by parallel execution on parallel pipelines

SISD – Explicit Multithreading

- Four approaches for explicit multithreading
  - Interleaved multithreading (fine-grained): switching can be at each clock cycle. In case of few active threads, performance degrades
  - Blocked multithreading (coarse-grained): events like cache miss produce switch
  - Simultaneous multithreading (SMT): execution units of a superscalar processor receive instructions from multiple threads
  - Chip multiprocessing: e.g. dual core (not SISD)
- Architectures like IA-64 Very Long Instruction Word (VLIW) allow multiple instructions (to be executed in parallel) in a single word

Intel’s Hyper-threading Technology

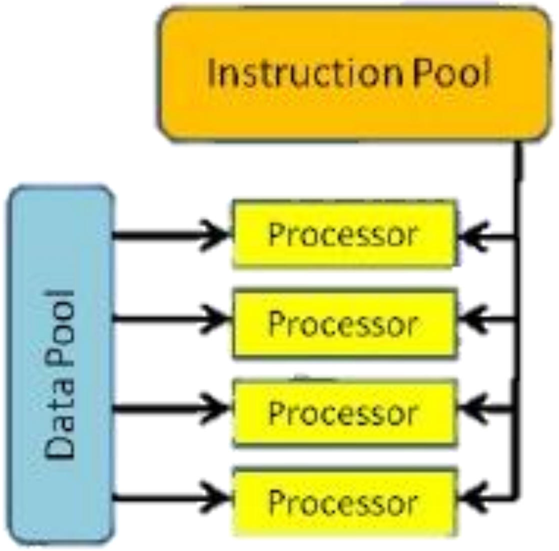
- A single physical processor appears as two logical processors by applying two-threaded SMT approach
- Each logical processor maintains a complete set of architecture state (general-purpose registers, control registers,…)
- Logical processors share nearly all other resources such as caches, execution units, branch predictors, control logic and buses
- Partitioned resources are recombined when only one thread is active
- Add less than 5% to the relative chip size
- Improve performance by 16% to 28%

SMT: https://en.wikipedia.org/wiki/Simultaneous_multithreading

Flynn’s Taxonomy

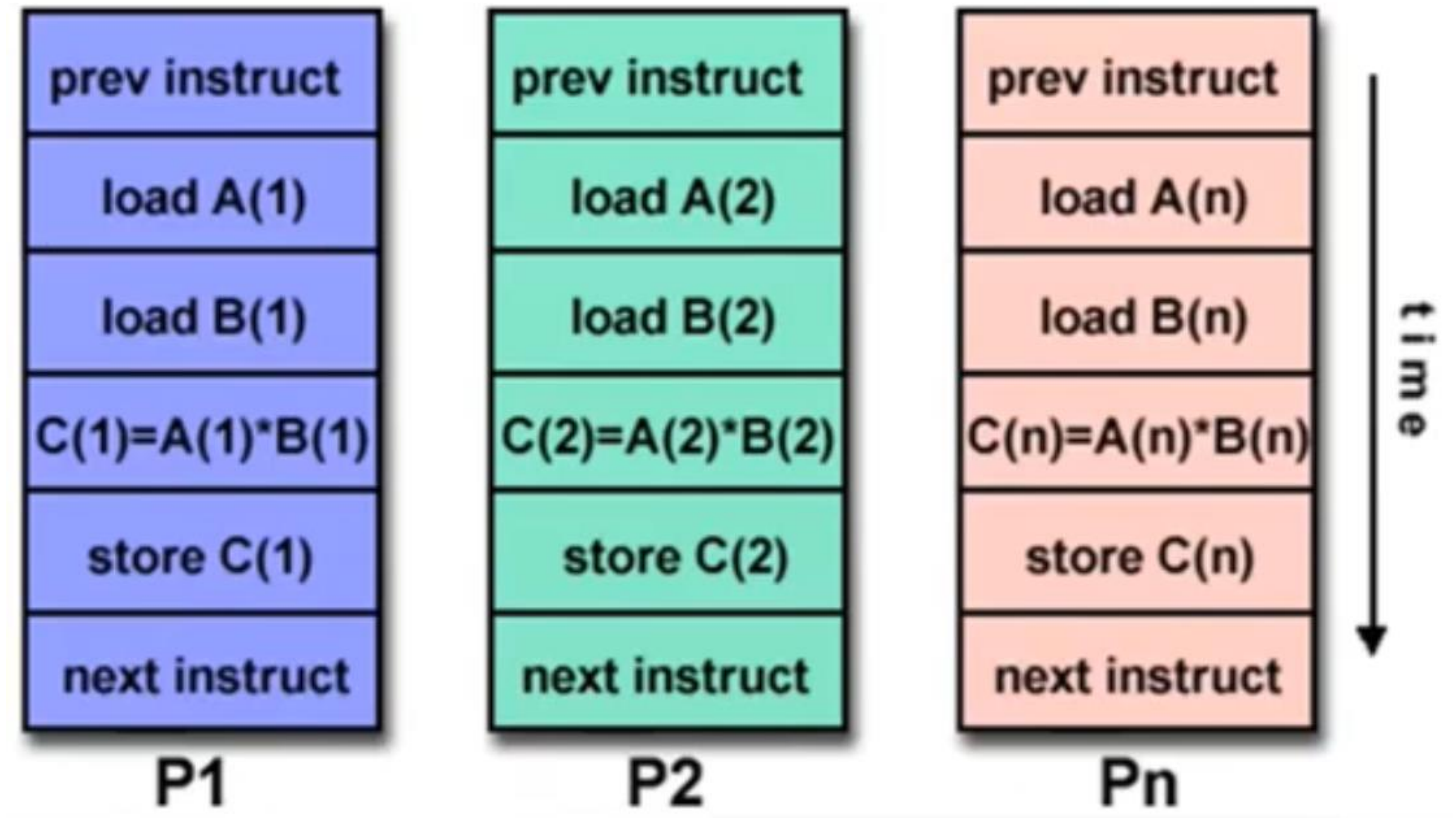
SIMD (Single Instruction Multiple Data)

- Refers to parallel architecture with multiple cores
- All the cores execute the same instruction stream at any time but, data stream is different for each.
- Well-suited for scientific operations requiring large matrix-vector operations
- Vector computers (Cray vector processing machine) and Intel co-processing unit ‘MMX’ fall under this category.
- Used with array operations, image processing and graphics

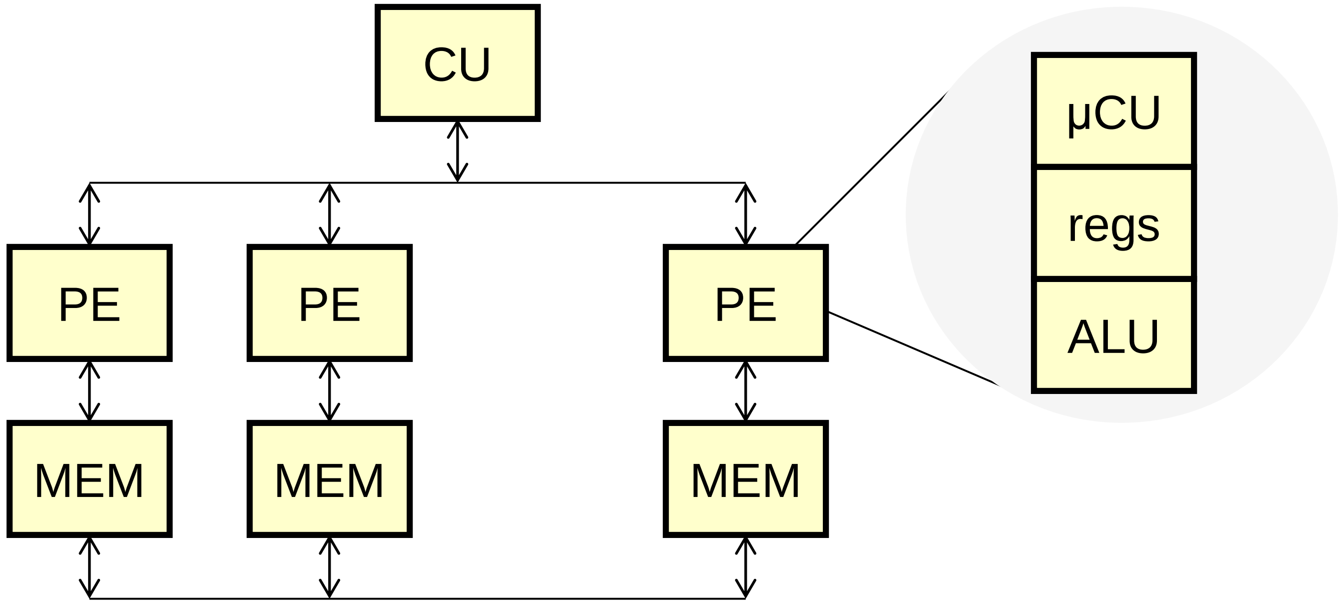


Array: same operations on different array elements.
Replaces the loops
Image: Applying same operation on different pixels

Example of SIMD:



SIMD



- Homogeneous processing units
- Single instruction: All processor units execute the same instruction at any given time
- Multiple data: Each processing unit can operate on different data set

SIMD (continued)

- Each processing element has an associated data memory, so that each instruction is executed on a different set of data by the different processors
- Used by vector and array processors
- Vector processors act on array of similar data (only when executing in vector mode) and in this case they are several times faster than when executing in scalar mode
  - Example is NEC SX-8B

SIMD – Example

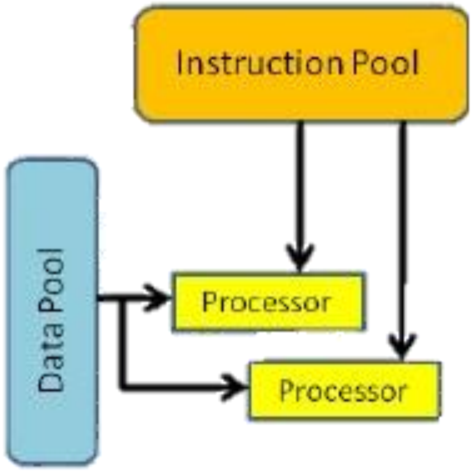
- A good example is the processing of pixels on screen
- A sequential processor would examine each pixel one at a time and apply the processing instruction
- An array or vector processor can process all the elements of an array simultaneously
- Game consoles and graphic cards make heavy use of such processors to shift those pixels
- Such designs are usually dedicated to a particular application and not commonly marketed for general purpose computing

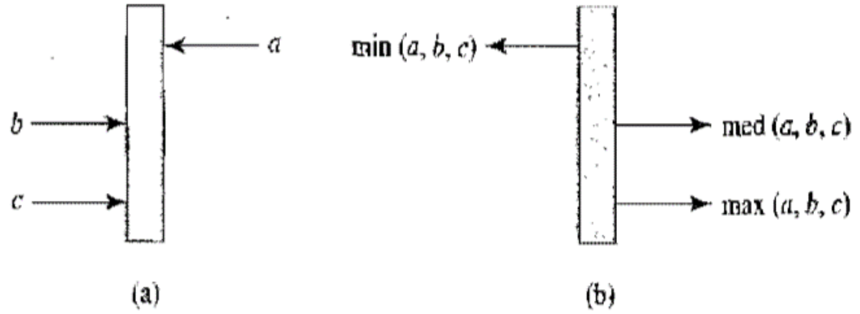
SIMD – Example

Flynn’s Taxonomy

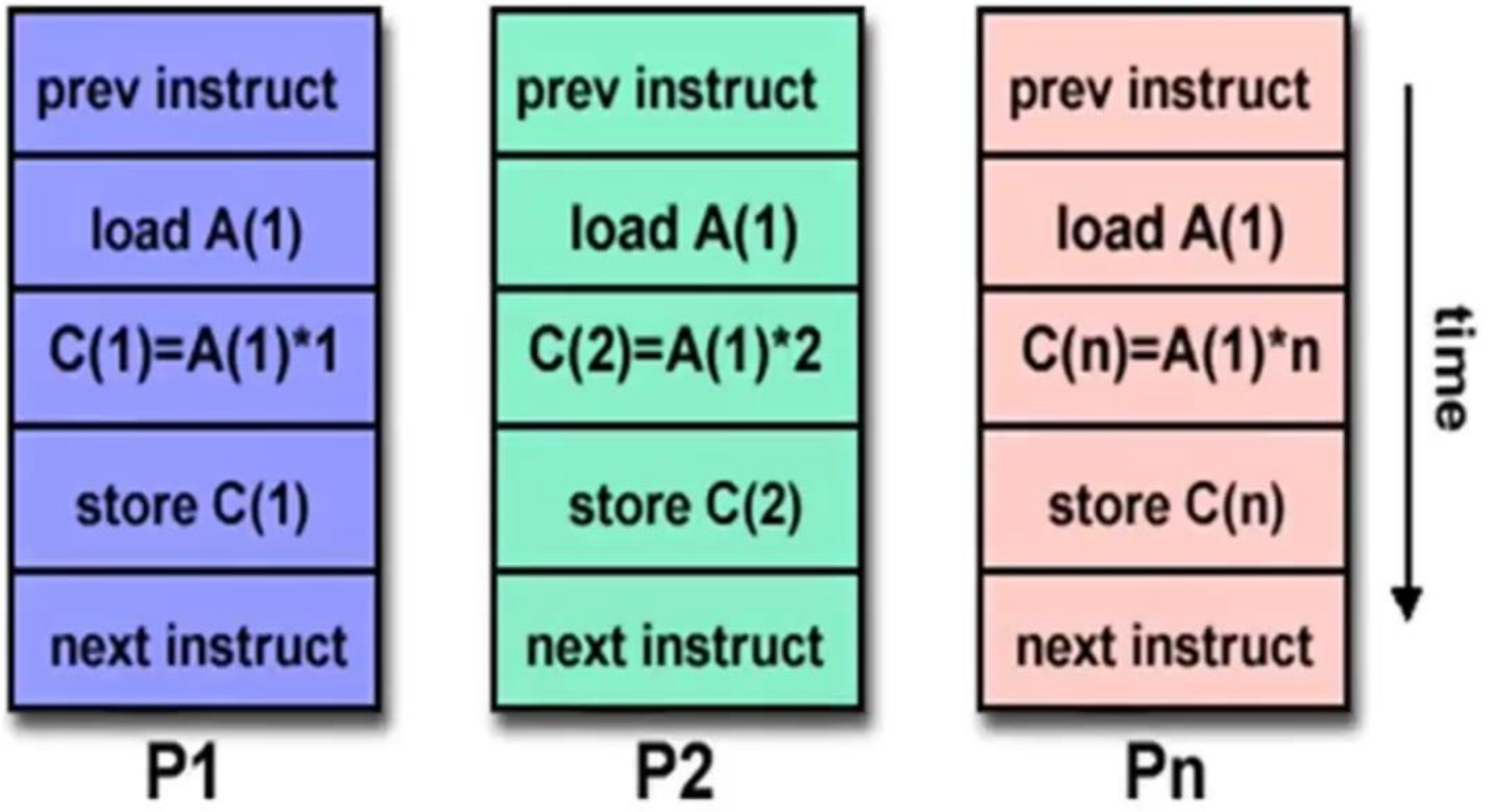
MISD (Multiple Instructions Single Data)

- Multiple instruction stream and single data stream
  - A pipeline of multiple independently executing functional units
  - Each operating on a single stream of data and forwarding results from one to the next
- Rarely used in practice
- E.g., Systolic arrays : network of primitive processing elements that pump data.





Example of MISD:



MISD

- A single data stream is transmitted to a set of processors, each of which executes a different instruction sequence
- Each processing unit operates on the data independently via independent instruction stream
- This structure is not commercially implemented
- An example of use could be multiple cryptography algorithms attempting to crack a coded message

Reading Assignment

- Cache Coherence and Snooping
- Branch prediction and issues while pipelining the problem

Sources

- Slides of Dr. Rana Asif Rahman & Dr. Haroon Mahmood, FAST