

Parallel & Distributed Computing: Lecture 5a

from Blaise N. Barney, [HPC Training Materials](#), by kind permission of
Lawrence Livermore National Laboratory's Computational Training
Center

October 9, 2018

Concepts and Terminology

1 General concepts

General concepts

von Neumann Computer Architecture

- Named after the Hungarian mathematician/genius **John von Neumann** who first authored the general requirements for an electronic computer in his 1945 papers.



Figure 1: **John von Neumann circa 1940s** (Source: LANL archives)

von Neumann Computer Architecture

- Named after the Hungarian mathematician/genius John von Neumann who first authored the general requirements for an electronic computer in his 1945 papers.
- Also known as “stored-program computer” - both program instructions and data are kept in electronic memory. Differs from earlier computers which were programmed through “hard wiring”.



Figure 1: John von Neumann circa 1940s (Source: LANL archives)

von Neumann Computer Architecture

- Named after the Hungarian mathematician/genius **John von Neumann** who first authored the general requirements for an electronic computer in his 1945 papers.
- Also known as “**stored-program computer**” - both **program instructions** and **data** are kept in electronic memory. Differs from earlier computers which were programmed through “hard wiring”.
- Since then, virtually all computers have followed this basic design



Figure 1: **John von Neumann circa 1940s** (Source: LANL archives)

von Neumann Computer Architecture

- Four main components:

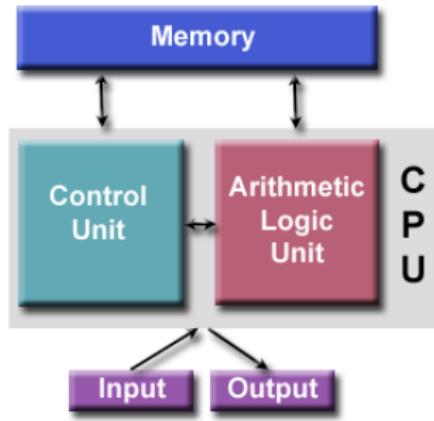


Figure 2: von Neumann
Architecture

von Neumann Computer Architecture

- Four main components:
 - Memory

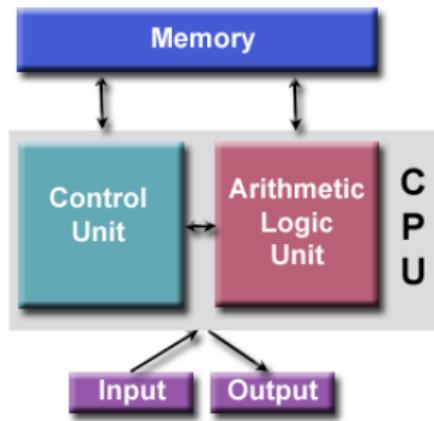


Figure 2: von Neumann
Architecture

von Neumann Computer Architecture

- Four main components:

- Memory
- Control Unit

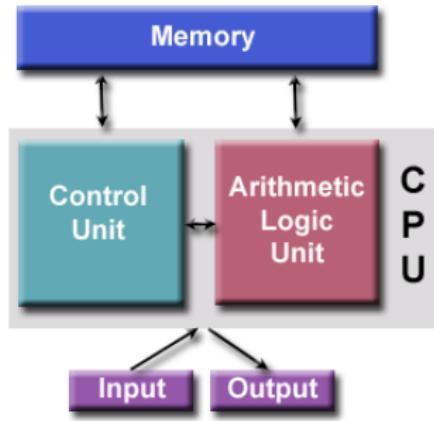


Figure 2: von Neumann
Architecture

von Neumann Computer Architecture

- Four main components:

- Memory
- Control Unit
- Arithmetic Logic Unit

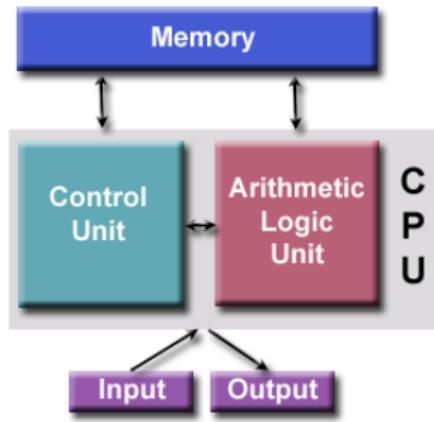


Figure 2: von Neumann
Architecture

von Neumann Computer Architecture

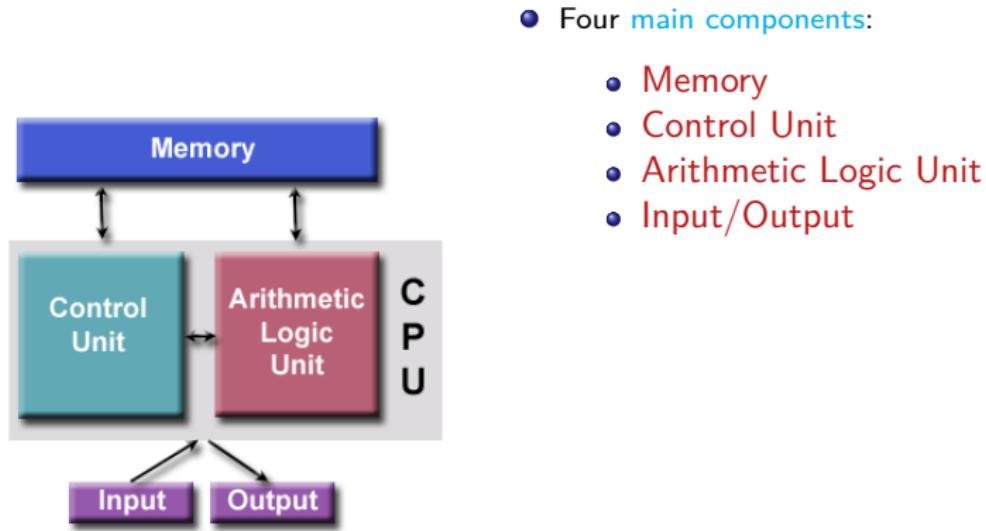
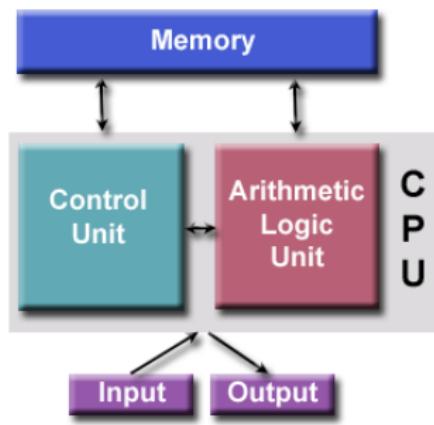


Figure 2: von Neumann
Architecture

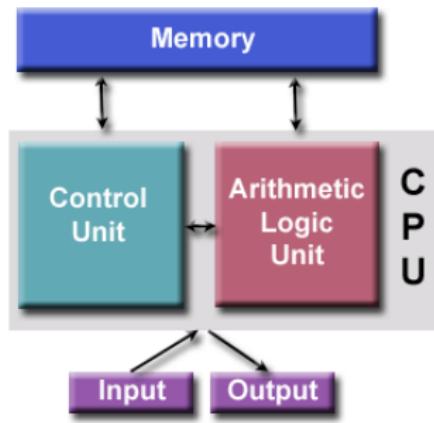
von Neumann Computer Architecture



- Four main components:
 - Memory
 - Control Unit
 - Arithmetic Logic Unit
 - Input/Output
- Read/write, random access memory to store both program instructions and data

Figure 2: von Neumann Architecture

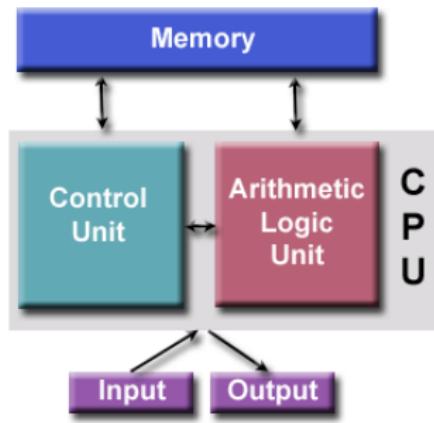
von Neumann Computer Architecture



- Four main components:
 - Memory
 - Control Unit
 - Arithmetic Logic Unit
 - Input/Output
- Read/write, random access memory to store both program instructions and data
 - Program instructions are coded data which tell the computer to do something

Figure 2: von Neumann Architecture

von Neumann Computer Architecture



- Four main components:
 - Memory
 - Control Unit
 - Arithmetic Logic Unit
 - Input/Output
- **Read/write**, random access memory to store both program instructions and data
 - Program instructions are coded data which tell the computer to do something
 - Data is information to be used by the program

Figure 2: von Neumann Architecture

von Neumann Computer Architecture

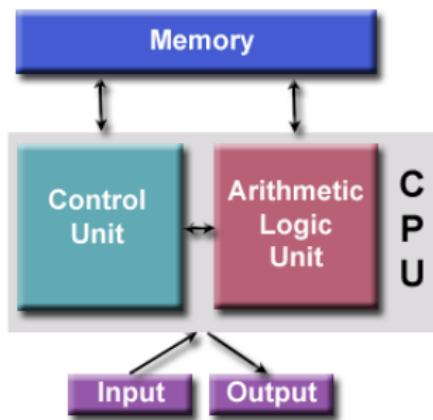


Figure 2: von Neumann Architecture

- Four main components:
 - Memory
 - Control Unit
 - Arithmetic Logic Unit
 - Input/Output
- Read/write, random access memory to store both program instructions and data
 - Program instructions are coded data which tell the computer to do something
 - Data is information to be used by the program
- Control unit fetches instructions/data from memory, decodes the instructions and then sequentially coordinates operations to accomplish the programmed task.

von Neumann Computer Architecture

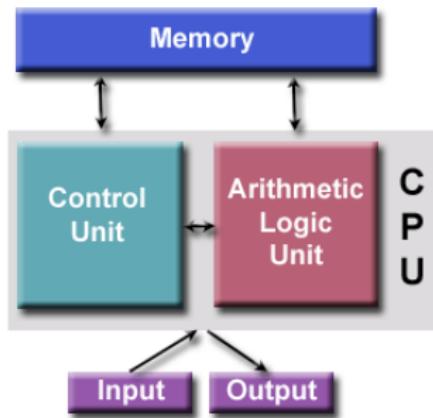


Figure 2: von Neumann Architecture

- Four main components:
 - Memory
 - Control Unit
 - Arithmetic Logic Unit
 - Input/Output
- Read/write, random access memory to store both program instructions and data
 - Program instructions are coded data which tell the computer to do something
 - Data is information to be used by the program
- Control unit fetches instructions/data from memory, decodes the instructions and then sequentially coordinates operations to accomplish the programmed task.
- Arithmetic Unit performs basic arithmetic operations

von Neumann Computer Architecture

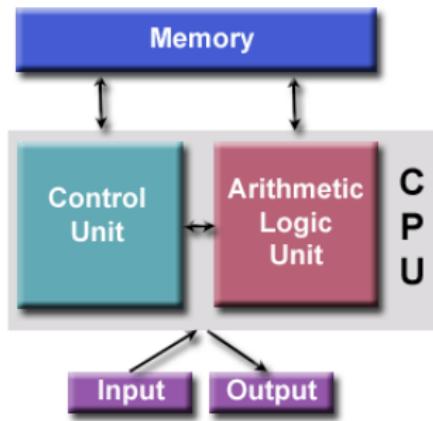


Figure 2: von Neumann Architecture

- Four main components:
 - Memory
 - Control Unit
 - Arithmetic Logic Unit
 - Input/Output
- Read/write, random access memory to store both program instructions and data
 - Program instructions are coded data which tell the computer to do something
 - Data is information to be used by the program
- Control unit fetches instructions/data from memory, decodes the instructions and then sequentially coordinates operations to accomplish the programmed task.
- Arithmetic Unit performs basic arithmetic operations
- Input/Output is the interface to the human operator

Flynn's Classical Taxonomy

- There are different ways to classify parallel computers. Examples available [HERE](#).

S I S D	S I M D
Single Instruction stream Single Data stream	Single Instruction stream Multiple Data stream
M I S D	M I M D
Multiple Instruction stream Single Data stream	Multiple Instruction stream Multiple Data stream

Figure 3: 4 possible classifications according to Flynn

Flynn's Classical Taxonomy

- There are different ways to classify parallel computers. Examples available [HERE](#).
- One of the more widely used classifications, in use since 1966, is called **Flynn's Taxonomy**.

S I S D Single Instruction stream Single Data stream	S I M D Single Instruction stream Multiple Data stream
M I S D Multiple Instruction stream Single Data stream	M I M D Multiple Instruction stream Multiple Data stream

Figure 3: 4 possible classifications according to Flynn

Flynn's Classical Taxonomy

- There are different ways to classify parallel computers. Examples available [HERE](#).
- One of the more widely used classifications, in use since 1966, is called **Flynn's Taxonomy**.
- Flynn's taxonomy distinguishes multi-processor computer architectures according to how they can be classified along the two independent dimensions of **Instruction Stream** and **Data Stream**.

S I S D Single Instruction stream Single Data stream	S I M D Single Instruction stream Multiple Data stream
M I S D Multiple Instruction stream Single Data stream	M I M D Multiple Instruction stream Multiple Data stream

Figure 3: 4 possible classifications according to Flynn

Flynn's Classical Taxonomy

- There are different ways to classify parallel computers. Examples available [HERE](#).
- One of the more widely used classifications, in use since 1966, is called **Flynn's Taxonomy**.
- Flynn's taxonomy distinguishes multi-processor computer architectures according to how they can be classified along the two independent dimensions of **Instruction Stream** and **Data Stream**.
- Each dimension can have only two possible states: **Single** or **Multiple**.

S I S D Single Instruction stream Single Data stream	S I M D Single Instruction stream Multiple Data stream
M I S D Multiple Instruction stream Single Data stream	M I M D Multiple Instruction stream Multiple Data stream

Figure 3: 4 possible classifications according to Flynn

Single Instruction, Single Data (SISD)

A serial (non-parallel) computer

- **Single Instruction:** Only one instruction stream is being acted on by the CPU during any one clock cycle

Single Instruction, Single Data (SISD)

A serial (non-parallel) computer

- **Single Instruction:** Only one instruction stream is being acted on by the CPU during any one clock cycle
- **Single Data:** Only one data stream is being used as input during any one clock cycle

Single Instruction, Single Data (SISD)

A serial (non-parallel) computer

- **Single Instruction:** Only one instruction stream is being acted on by the CPU during any one clock cycle
- **Single Data:** Only one data stream is being used as input during any one clock cycle
- Deterministic execution

Single Instruction, Single Data (SISD)

A serial (non-parallel) computer

- **Single Instruction:** Only one instruction stream is being acted on by the CPU during any one clock cycle
- **Single Data:** Only one data stream is being used as input during any one clock cycle
- Deterministic execution
- This is the **oldest type of computer**

Single Instruction, Single Data (SISD)

A serial (non-parallel) computer

- **Single Instruction:** Only one instruction stream is being acted on by the CPU during any one clock cycle
- **Single Data:** Only one data stream is being used as input during any one clock cycle
- Deterministic execution
- This is the **oldest type of computer**

Single Instruction, Single Data (SISD)

A serial (non-parallel) computer

- **Single Instruction:** Only one instruction stream is being acted on by the CPU during any one clock cycle
- **Single Data:** Only one data stream is being used as input during any one clock cycle
- Deterministic execution
- This is the **oldest type of computer**
- Examples: older generation mainframes, minicomputers, workstations and single processor/core PCs.

Single Instruction, Single Data (SISD)

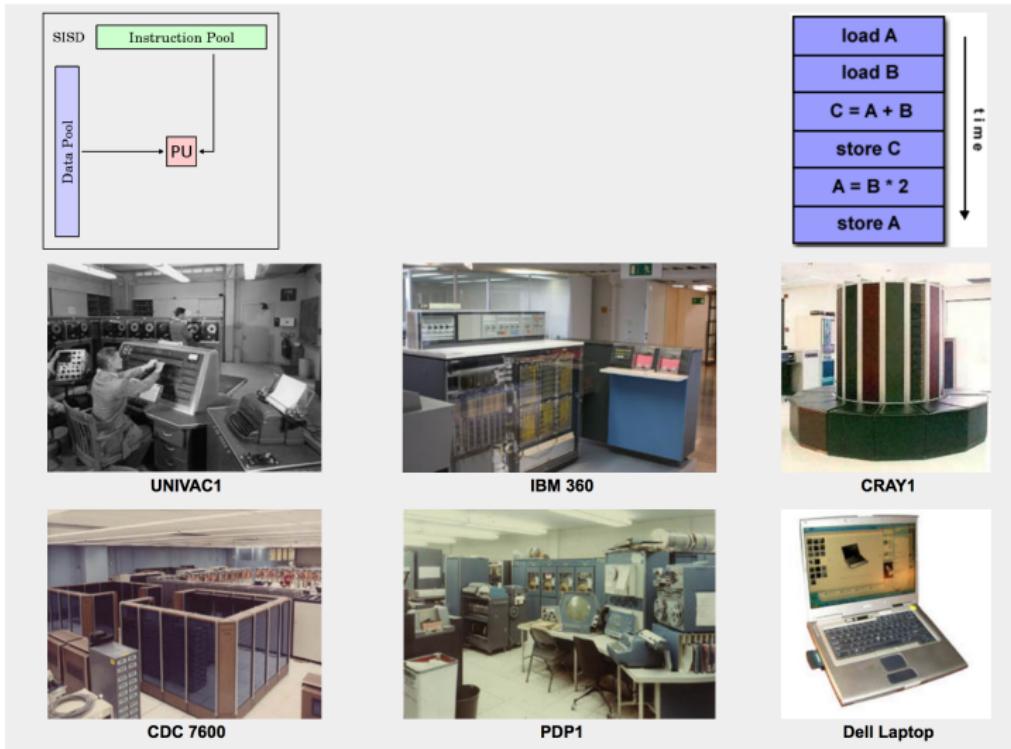


Figure 4. Single Instruction, Single Data (SISD)

Single Instruction, Multiple Data (SIMD)

A type of parallel computer

- **Single Instruction:** All processing units execute the same instruction at any given clock cycle

Single Instruction, Multiple Data (SIMD)

A type of parallel computer

- **Single Instruction:** All processing units execute the same instruction at any given clock cycle
- **Multiple Data:** Each processing unit can operate on a different data element

Single Instruction, Multiple Data (SIMD)

A type of parallel computer

- **Single Instruction:** All processing units execute the same instruction at any given clock cycle
- **Multiple Data:** Each processing unit can operate on a different data element
- Best suited for **specialized problems** characterized by a high degree of regularity, such as **graphics/image processing**.

Single Instruction, Multiple Data (SIMD)

A type of parallel computer

- **Single Instruction:** All processing units execute the same instruction at any given clock cycle
- **Multiple Data:** Each processing unit can operate on a different data element
- Best suited for **specialized problems** characterized by a high degree of regularity, such as **graphics/image processing**.
- Synchronous (lockstep) and deterministic execution

Single Instruction, Multiple Data (SIMD)

A type of parallel computer

- **Single Instruction:** All processing units execute the same instruction at any given clock cycle
- **Multiple Data:** Each processing unit can operate on a different data element
- Best suited for **specialized problems** characterized by a high degree of regularity, such as **graphics/image processing**.
- Synchronous (lockstep) and deterministic execution
- Two varieties: **Processor Arrays** and **Vector Pipelines**

Single Instruction, Multiple Data (SIMD)

A type of parallel computer

- **Single Instruction:** All processing units execute the same instruction at any given clock cycle
- **Multiple Data:** Each processing unit can operate on a different data element
- Best suited for **specialized problems** characterized by a high degree of regularity, such as **graphics/image processing**.
- Synchronous (lockstep) and deterministic execution
- Two varieties: **Processor Arrays** and **Vector Pipelines**

Single Instruction, Multiple Data (SIMD)

A type of parallel computer

- **Single Instruction:** All processing units execute the same instruction at any given clock cycle
- **Multiple Data:** Each processing unit can operate on a different data element
- Best suited for **specialized problems** characterized by a high degree of regularity, such as **graphics/image processing**.
- Synchronous (lockstep) and deterministic execution
- Two varieties: **Processor Arrays** and **Vector Pipelines**

- Examples:

Single Instruction, Multiple Data (SIMD)

A type of parallel computer

- **Single Instruction:** All processing units execute the same instruction at any given clock cycle
- **Multiple Data:** Each processing unit can operate on a different data element
- Best suited for **specialized problems** characterized by a high degree of regularity, such as **graphics/image processing**.
- Synchronous (lockstep) and deterministic execution
- Two varieties: **Processor Arrays** and **Vector Pipelines**

- Examples:
 - **Processor Arrays:** Thinking Machines CM-2, MasPar MP-1 & MP-2, ILLIAC IV

Single Instruction, Multiple Data (SIMD)

A type of parallel computer

- **Single Instruction:** All processing units execute the same instruction at any given clock cycle
- **Multiple Data:** Each processing unit can operate on a different data element
- Best suited for **specialized problems** characterized by a high degree of regularity, such as **graphics/image processing**.
- Synchronous (lockstep) and deterministic execution
- Two varieties: **Processor Arrays** and **Vector Pipelines**

- Examples:
 - **Processor Arrays:** Thinking Machines CM-2, MasPar MP-1 & MP-2, ILLIAC IV
 - **Vector Pipelines:** IBM 9000, Cray X-MP, Y-MP & C90, Fujitsu VP, NEC SX-2, Hitachi S820, ETA10

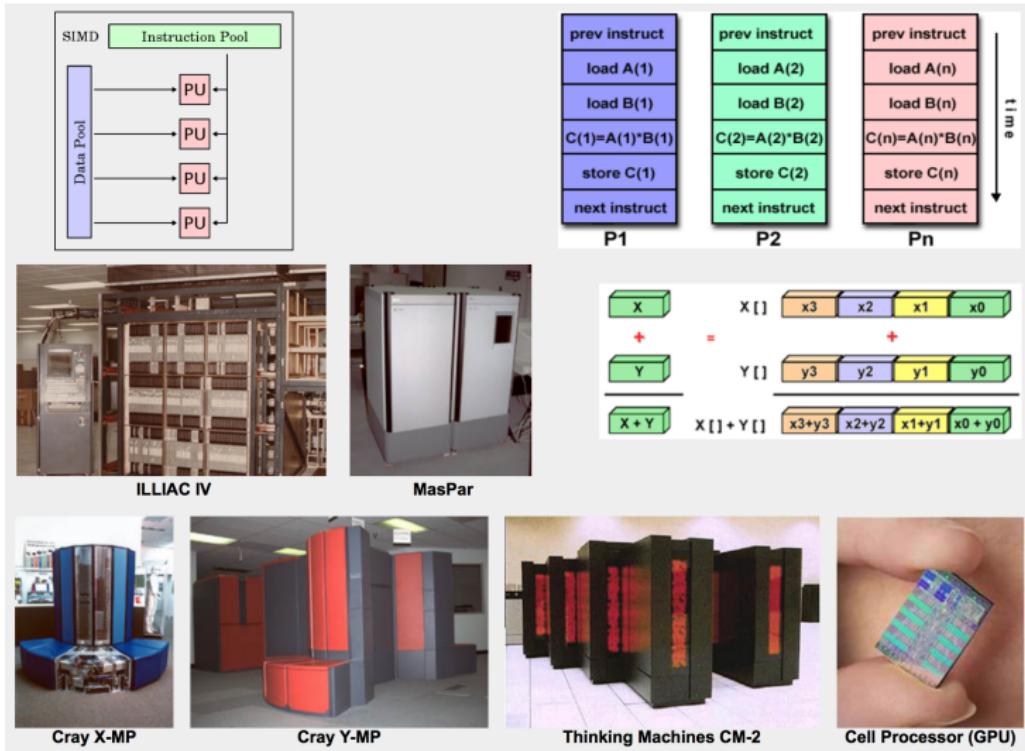
Single Instruction, Multiple Data (SIMD)

A type of parallel computer

- **Single Instruction:** All processing units execute the same instruction at any given clock cycle
- **Multiple Data:** Each processing unit can operate on a different data element
- Best suited for **specialized problems** characterized by a high degree of regularity, such as **graphics/image processing**.
- Synchronous (lockstep) and deterministic execution
- Two varieties: **Processor Arrays** and **Vector Pipelines**

- Examples:
 - **Processor Arrays:** Thinking Machines CM-2, MasPar MP-1 & MP-2, ILLIAC IV
 - **Vector Pipelines:** IBM 9000, Cray X-MP, Y-MP & C90, Fujitsu VP, NEC SX-2, Hitachi S820, ETA10
- **Most modern computers**, particularly those with **graphics processor units (GPUs)** employ SIMD instructions and execution units.

Single Instruction, Multiple Data (SIMD)



Multiple Instruction, Single Data (MISD)

A type of parallel computer

- **Multiple Instruction:** Each processing unit operates on the data independently via separate instruction streams.

Multiple Instruction, Single Data (MISD)

A type of parallel computer

- **Multiple Instruction:** Each processing unit operates on the data independently via separate instruction streams.
- **Single Data:** A single data stream is fed into multiple processing units.

Multiple Instruction, Single Data (MISD)

A type of parallel computer

- **Multiple Instruction:** Each processing unit operates on the data independently via separate instruction streams.
- **Single Data:** A single data stream is fed into multiple processing units.
- **Few (if any) actual examples** of this class of parallel computer have ever existed.

Multiple Instruction, Single Data (MISD)

A type of parallel computer

- **Multiple Instruction:** Each processing unit operates on the data independently via separate instruction streams.
- **Single Data:** A single data stream is fed into multiple processing units.
- **Few (if any) actual examples** of this class of parallel computer have ever existed.

Multiple Instruction, Single Data (MISD)

A type of parallel computer

- **Multiple Instruction:** Each processing unit operates on the data independently via separate instruction streams.
- **Single Data:** A single data stream is fed into multiple processing units.
- **Few (if any) actual examples** of this class of parallel computer have ever existed.
- Some conceivable uses might be:

Multiple Instruction, Single Data (MISD)

A type of parallel computer

- **Multiple Instruction:** Each processing unit operates on the data independently via separate instruction streams.
- **Single Data:** A single data stream is fed into multiple processing units.
- **Few (if any) actual examples** of this class of parallel computer have ever existed.
- Some conceivable uses might be:
 - multiple frequency filters operating on a single signal stream

Multiple Instruction, Single Data (MISD)

A type of parallel computer

- **Multiple Instruction:** Each processing unit operates on the data independently via separate instruction streams.
- **Single Data:** A single data stream is fed into multiple processing units.
- **Few (if any) actual examples** of this class of parallel computer have ever existed.
- Some conceivable uses might be:
 - multiple frequency filters operating on a single signal stream
 - **multiple cryptography algorithms** attempting to crack a single coded message.

Multiple Instruction, Single Data (MISD)

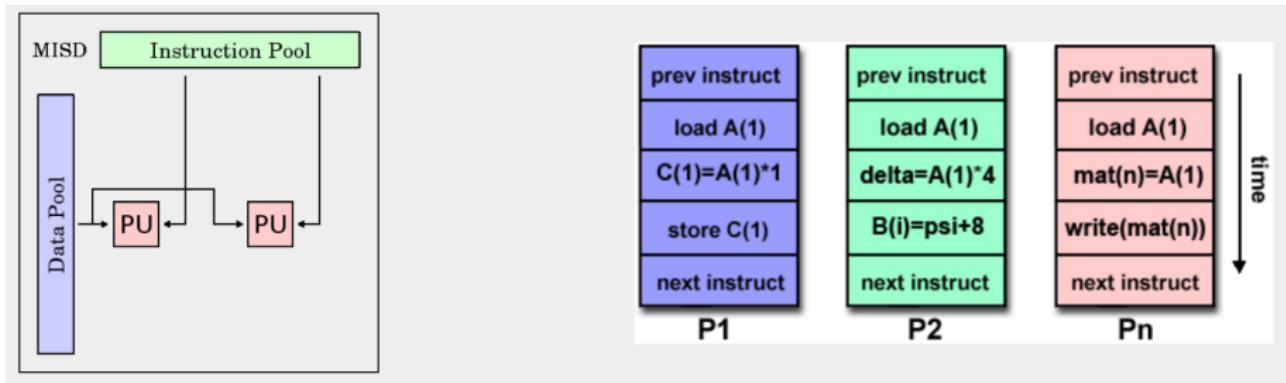


Figure 6: Multiple Instruction, Single Data (MISD)

Multiple Instruction, Multiple Data (SIMD)

A type of parallel computer

- **Multiple Instruction:** Every processor may be executing a different instruction stream

Multiple Instruction, Multiple Data (SIMD)

A type of parallel computer

- **Multiple Instruction:** Every processor may be executing a different instruction stream
- **Multiple Data:** Every processor may be working with a different data stream

Multiple Instruction, Multiple Data (SIMD)

A type of parallel computer

- **Multiple Instruction:** Every processor may be executing a different instruction stream
- **Multiple Data:** Every processor may be working with a different data stream
- Execution can be synchronous or asynchronous, deterministic or non-deterministic

Multiple Instruction, Multiple Data (SIMD)

A type of parallel computer

- **Multiple Instruction:** Every processor may be executing a different instruction stream
- **Multiple Data:** Every processor may be working with a different data stream
- Execution can be synchronous or asynchronous, deterministic or non-deterministic
- Currently, the **most common type** of parallel computer - most modern supercomputers fall into this category.

Multiple Instruction, Multiple Data (SIMD)

A type of parallel computer

- **Multiple Instruction:** Every processor may be executing a different instruction stream
- **Multiple Data:** Every processor may be working with a different data stream
- Execution can be synchronous or asynchronous, deterministic or non-deterministic
- Currently, the **most common type** of parallel computer - most modern supercomputers fall into this category.

Multiple Instruction, Multiple Data (SIMD)

A type of parallel computer

- **Multiple Instruction:** Every processor may be executing a different instruction stream
- **Multiple Data:** Every processor may be working with a different data stream
- Execution can be synchronous or asynchronous, deterministic or non-deterministic
- Currently, the **most common type** of parallel computer - most modern supercomputers fall into this category.

- Examples: most current supercomputers, networked parallel computer **clusters** and “**grids**”, multi-processor **SMP computers**, **multi-core PCs**.

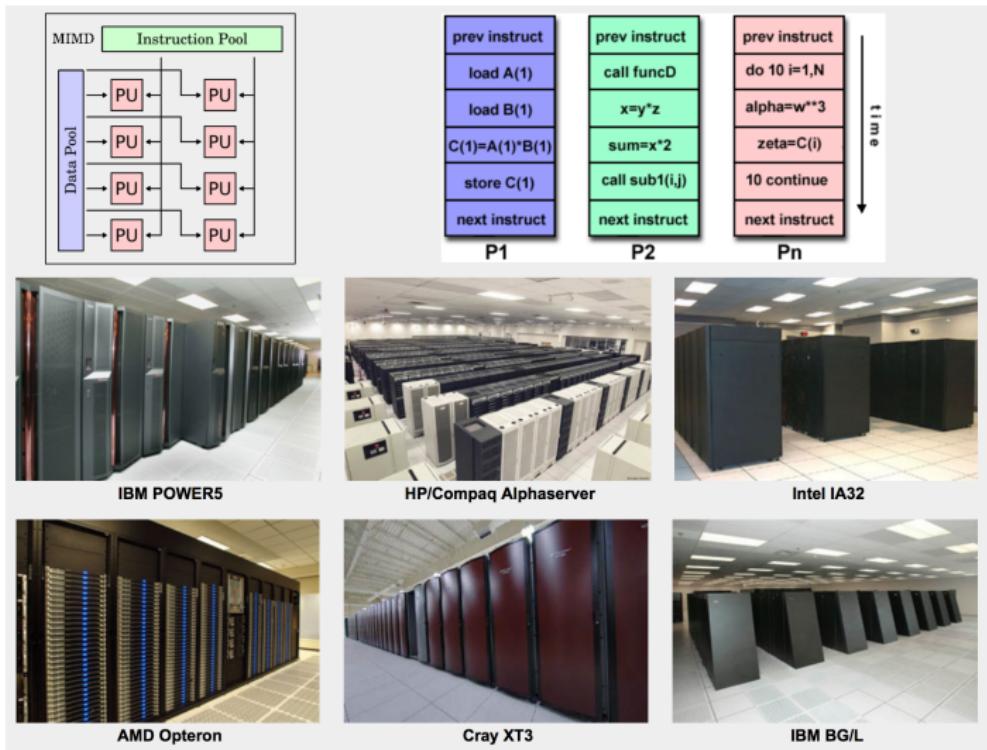
Multiple Instruction, Multiple Data (SIMD)

A type of parallel computer

- **Multiple Instruction:** Every processor may be executing a different instruction stream
- **Multiple Data:** Every processor may be working with a different data stream
- Execution can be synchronous or asynchronous, deterministic or non-deterministic
- Currently, the **most common type** of parallel computer - most modern supercomputers fall into this category.

- Examples: most current supercomputers, networked parallel computer **clusters** and “**grids**”, multi-processor **SMP computers**, multi-core PCs.
- Note: many MIMD architectures also include **SIMD execution sub-components**

Multiple Instruction, Multiple Data (SIMD)



Some General Parallel Terminology

- Like everything else, parallel computing has its own “jargon”. Some of the more commonly used terms associated with parallel computing are listed below.

Some General Parallel Terminology

- Like everything else, parallel computing has its own “jargon”. Some of the more commonly used terms associated with parallel computing are listed below.
- Most of these will be discussed in more detail later.

Some General Parallel Terminology

- Like everything else, parallel computing has its own “jargon”. Some of the more commonly used terms associated with parallel computing are listed below.
- Most of these will be discussed in more detail later.

Supercomputing / High Performance Computing (HPC) Using the world's fastest and largest computers to solve large problems.

Some General Parallel Terminology

- Like everything else, parallel computing has its own “jargon”. Some of the more commonly used terms associated with parallel computing are listed below.
- Most of these will be discussed in more detail later.

Supercomputing / High Performance Computing (HPC) Using the world's fastest and largest computers to solve large problems.

Node A standalone “computer in a box”. Usually comprised of multiple CPUs/processors/cores, memory, network interfaces, etc.

Nodes are networked together to produce a supercomputer.

Some General Parallel Terminology

CPU / Socket / Processor / Core This varies, depending upon who you talk to.

In the past, a CPU (Central Processing Unit) was a singular execution component for a computer.

Then, multiple CPUs were incorporated into a node.

Then, individual CPUs were subdivided into multiple “cores”, each being a unique execution unit. CPUs with multiple cores are sometimes called “sockets” - vendor dependent.

The result is a node with multiple CPUs, each containing multiple cores. The nomenclature is confused at times.

Wonder why?

Some General Parallel Terminology



Supercomputer - each blue light is a node

Node - standalone
Von Neumann computer

CPU / Processor / Socket - each has multiple cores / processors.

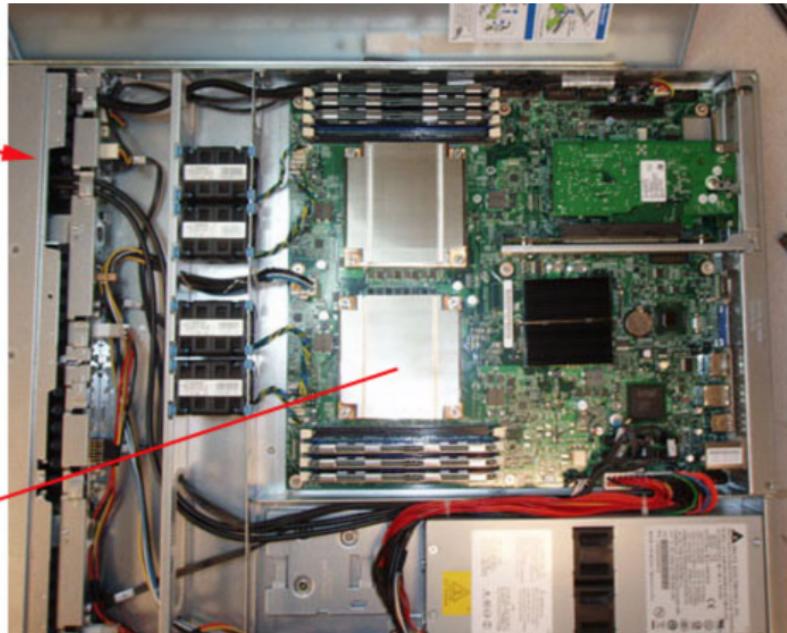
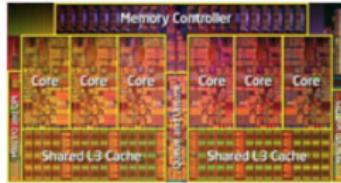


Figure 8: CPU / Socket / Processor / Core

Some General Parallel Terminology

Task A logically **discrete section** of computational work.

A task is typically a program or **program-like set of instructions** that is executed by a processor. A **parallel program** consists of **multiple tasks** running on **multiple processors**.

Some General Parallel Terminology

Task A logically **discrete section** of computational work.

A task is typically a program or **program-like set of instructions** that is executed by a processor. A **parallel program** consists of **multiple tasks** running on **multiple processors**.

Pipelining Breaking a task into steps performed by different processor units, **with inputs streaming** through, much like an assembly line; **a type of parallel computing**.

Some General Parallel Terminology

Shared Memory From a strictly hardware point of view, describes a computer architecture where **all processors** have direct (usually bus based) access to **common physical memory**. In a programming sense, it describes a model where **parallel tasks** all have the **same “picture” of memory** and can directly address and access the same logical memory locations **regardless of where** the physical memory actually exists.

Some General Parallel Terminology

Shared Memory From a strictly hardware point of view, describes a computer architecture where **all processors** have direct (usually bus based) access to **common physical memory**. In a programming sense, it describes a model where **parallel tasks** all have the **same “picture” of memory** and can directly address and access the same logical memory locations **regardless of where** the physical memory actually exists.

Symmetric Multi-Processor (SMP) Shared memory hardware architecture where **multiple processors share a single address space** and have **equal access to all resources**.

Some General Parallel Terminology

Distributed Memory In hardware, refers to network based memory access for physical memory that is not common.

As a programming model, tasks can only logically “see” local machine memory and must use communications to access memory on other machines where other tasks are executing.

Some General Parallel Terminology

Distributed Memory In hardware, refers to **network based memory access** for physical memory that is not common.

As a programming model, **tasks can only logically “see” local machine memory** and **must use communications** to access **memory on other machines** where other tasks are executing.

Communications Parallel tasks typically **need to exchange data**.

There are several ways this can be accomplished, such as through a **shared memory bus** or over a **network**, however the actual event of data exchange is commonly referred to as **communications** regardless of the method employed.