Mansoura University
Faculty of Computers and Information
Department of Computer Science
First Semester- 2020-2021

[CS131]

Computer Organization & Architecture

Grade: 2

DR: Zahraa Tarek

FUNCTIONAL ORGANIZATION

AGENDA

- Processor Organization
- Register Organization
- Instruction Cycle
- Parallel Processor Systems
  - Types of Parallel Processor Systems
  - Parallel Organizations

PROCESSOR ORGANIZATION

To understand the organization of the processor, let us consider the requirements placed on the processor, the things that it must do:

- Fetch instruction: The processor reads an instruction from memory (register, cache, main memory).
- Interpret instruction: The instruction is decoded to determine what action is required.
- Fetch data: The execution of an instruction may require reading data from memory or an I/O module.
- Process data: The execution of an instruction may require performing some arithmetic or logical operation on data.
- Write data: The results of an execution may require writing data to memory or an I/O module.

PROCESSOR ORGANIZATION

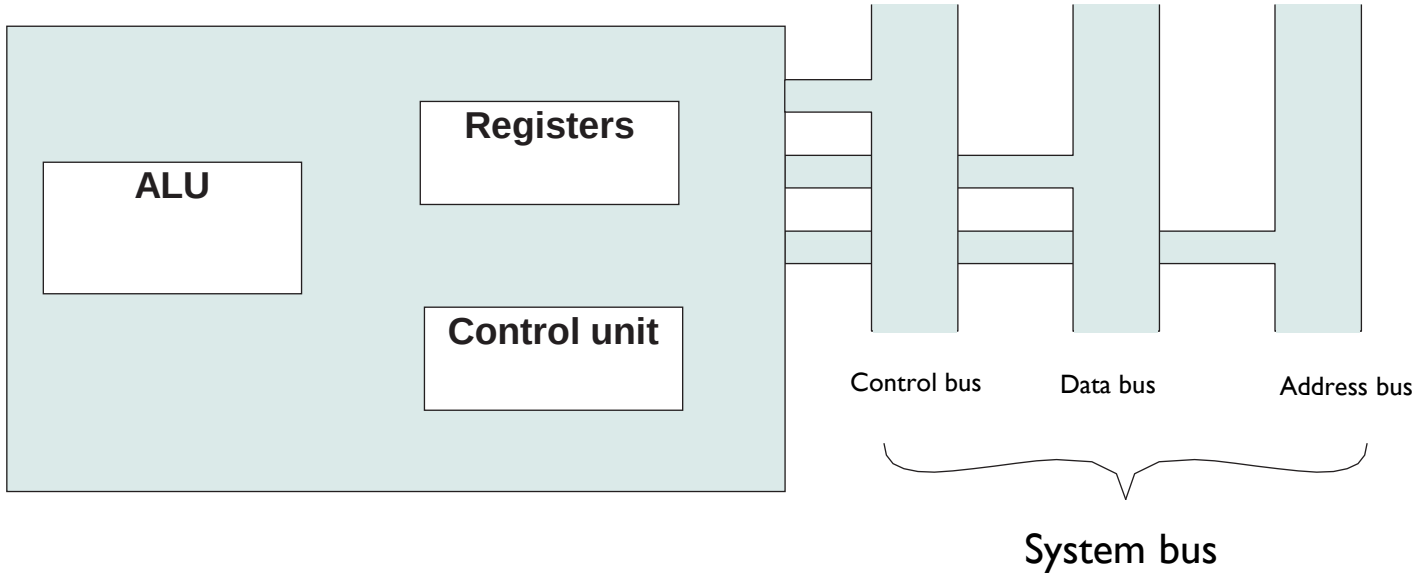
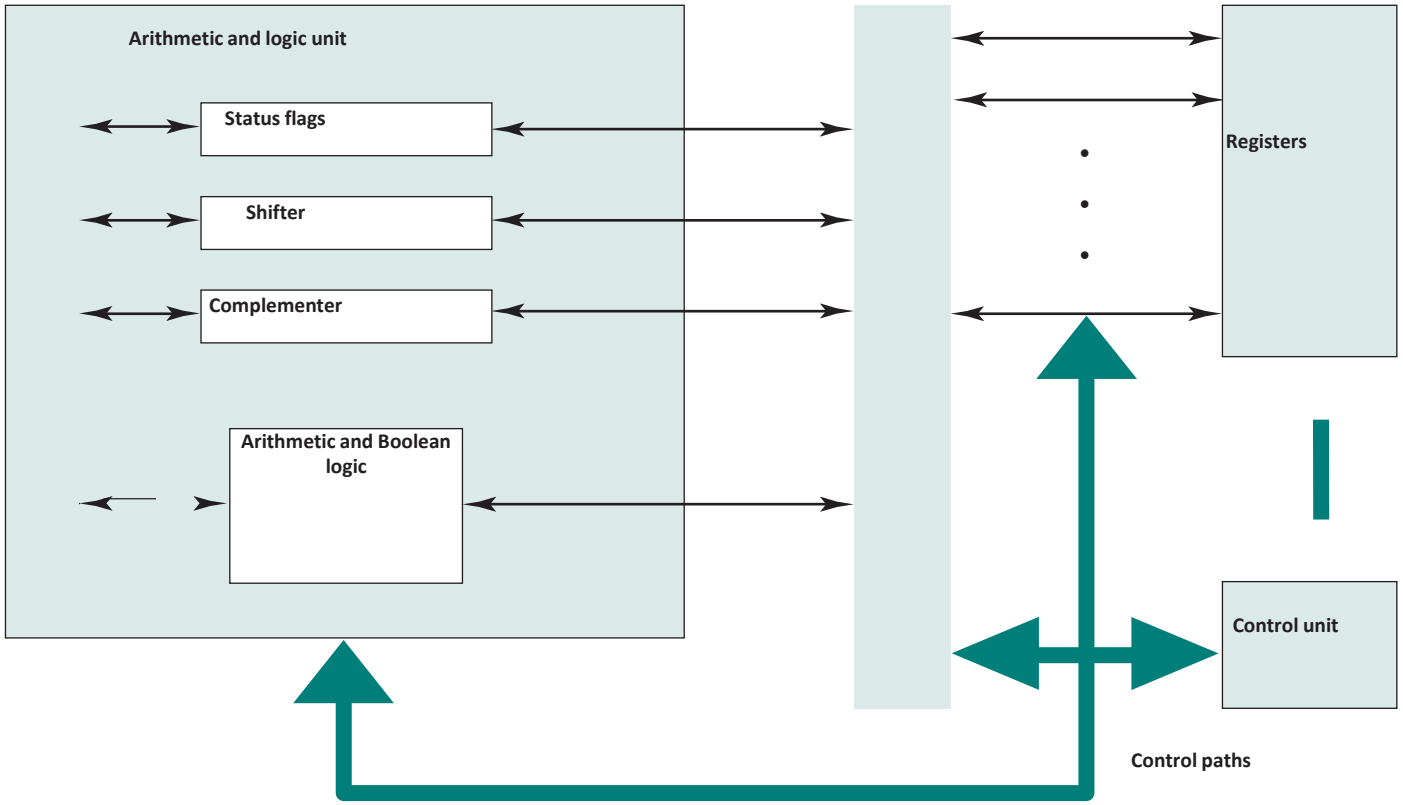


Figure 1. The CPU with the System Bus

INTERNAL STRUCTURE OF THE CPU



REGISTER ORGANIZATION

The registers in the processor perform two roles:

- **User-visible registers:** Enable the machine- or assembly language programmer to minimize main memory references by optimizing use of registers.
- **Control and status registers:** Used by the control unit to control the operation of the processor and by privileged, operating system programs to control the execution of programs.

USER-VISIBLE REGISTERS

A user-visible register is one that may be referenced by means of the machine language that the processor executes. We can characterize these in the following categories:

- General purpose
- Data
- Address
- Condition codes

GENERAL PURPOSE

- **General-purpose registers** can be assigned to a variety of functions by the programmer. Sometimes their use within the instruction set is orthogonal to the operation.
- That is, any general-purpose register can contain the operand for any opcode. This provides true general-purpose register use. Often, however, there are restrictions.
- For example, there may be dedicated registers for floating-point and stack operations.

DATA REGISTERS

- In some cases, general-purpose registers can be used for addressing functions (e.g., register indirect, displacement). In other cases, there is a partial or clean separation between data registers and address registers. Data registers may be used only to hold data and cannot be employed in the calculation of an operand address.

ADDRESS REGISTERS

Address registers may themselves be somewhat general purpose, or they may be devoted to a particular addressing mode. Examples include the following:

- **Segment pointers**
- **Index registers**
- **Stack pointer**

CONDITION CODES (FLAGS)

- Condition codes are bits set by the processor hardware as the result of operations.
- For example, an arithmetic operation may produce a positive, negative, zero, or overflow result. In addition to the result itself being stored in a register or memory, a condition code is also set. The code may be tested as part of a conditional branch.

REGISTER TYPES

- Four registers are essential to instruction execution:
  - **Program counter (PC):** Contains the address of an instruction to be fetched.
  - **Instruction register (IR):** Contains the instruction most recently fetched.
  - **Memory address register (MAR):** Contains the address of a location in memory.
  - **Memory buffer register (MBR):** Contains a word of data to be written to memory or the word most recently read.

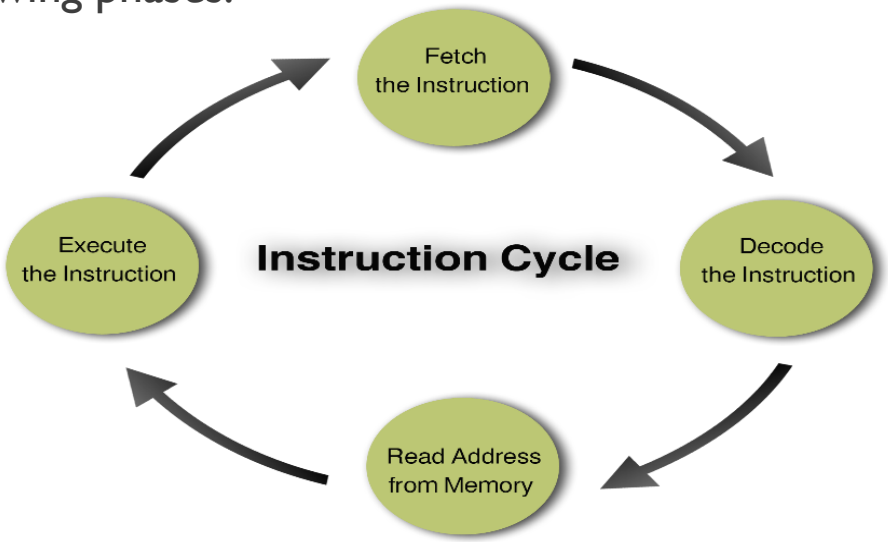
AGENDA

- Processor Organization
- Register Organization
- Instruction Cycle
- Parallel Processor Systems
  - Types of Parallel Processor Systems
  - Parallel Organizations

INSTRUCTION CYCLE

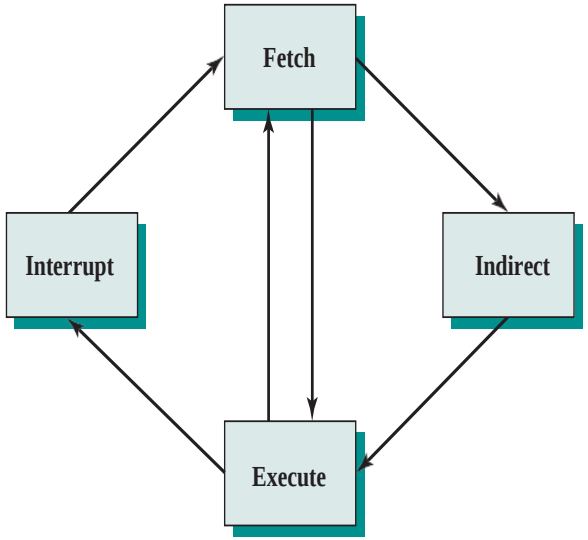
- In a basic computer, each instruction cycle consists of the following phases:

1. Fetch instruction from memory.
2. Decode the instruction.
3. Read the effective address from memory.
4. Execute the instruction



INDIRECT CYCLE

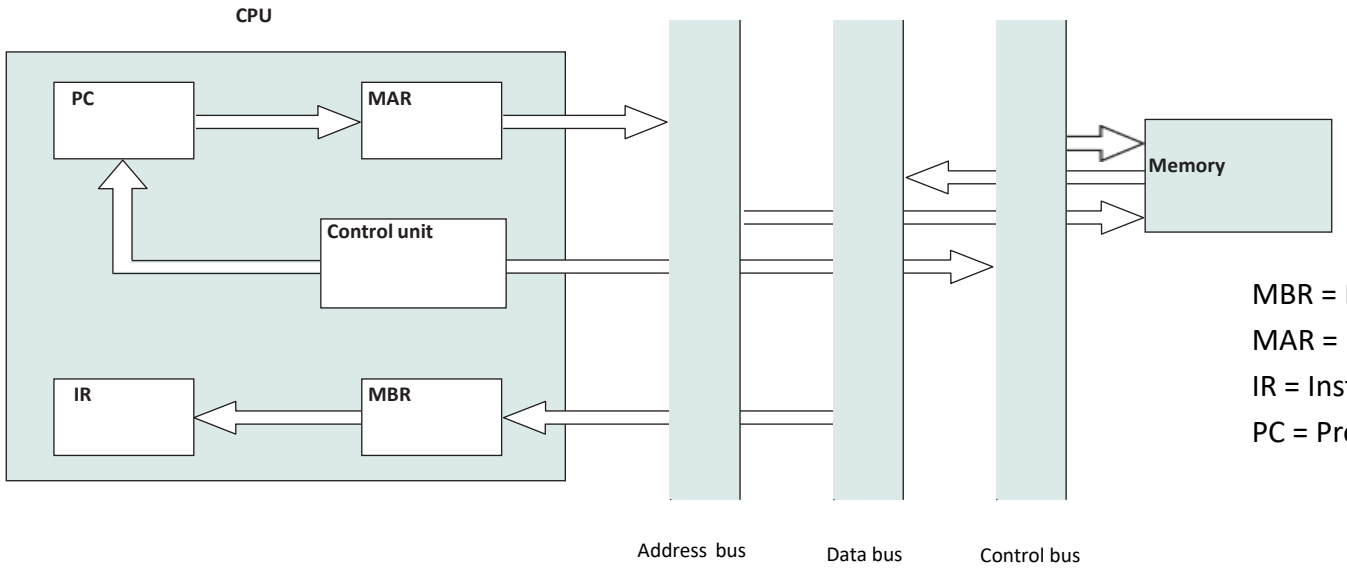
- If indirect addressing is used, then additional memory accesses are required
- After an instruction is fetched, it is examined to determine if any indirect addressing is involved. If so, the required operands are fetched using indirect addressing.
- Following execution, an interrupt may be processed before the next instruction fetch



DATA FLOW DURING FETCH CYCLE

- During the fetch cycle, an instruction is read from memory.
- The PC contains the address of the next instruction to be fetched.
- This address is moved to the MAR and placed on the address bus.
- The control unit requests a memory read, and the result is placed on the data bus and copied into the MBR and then moved to the IR.
- Meanwhile, the PC is incremented by 1, preparatory for the next fetch.

DATA FLOW, FETCH CYCLE

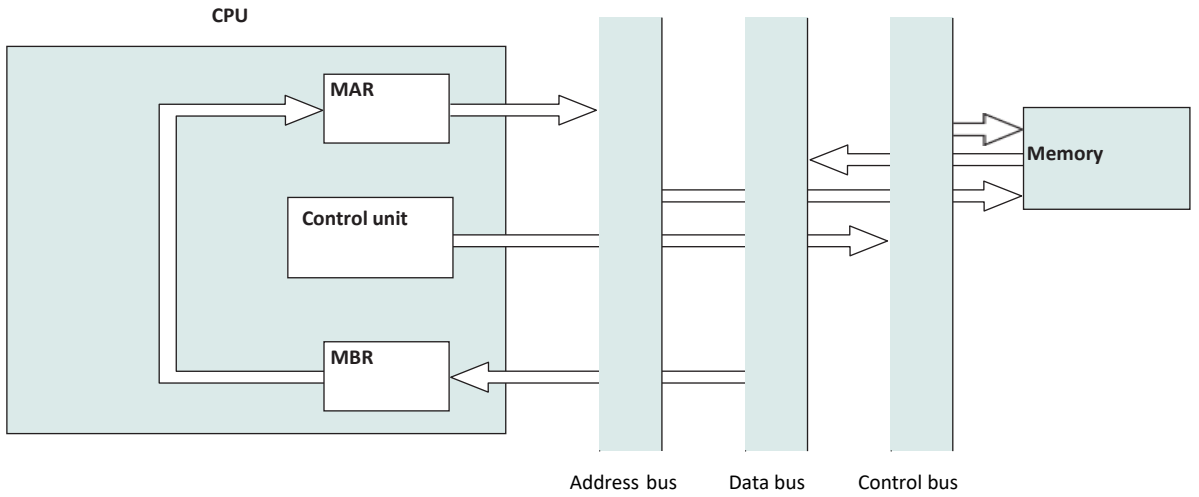


MBR = Memory buffer register
MAR = Memory address register
IR = Instruction register
PC = Program counter

DATA FLOW, INDIRECT CYCLE

- Once the fetch cycle is over, the control unit examines the contents of the IR to determine if it contains an operand specifier using indirect addressing.
- If so, an *indirect cycle* is performed.
- The right- most *N* bits of the MBR, which contain the address reference, are transferred to the MAR.
- Then the control unit requests a memory read, to get the desired address of the operand into the MBR.

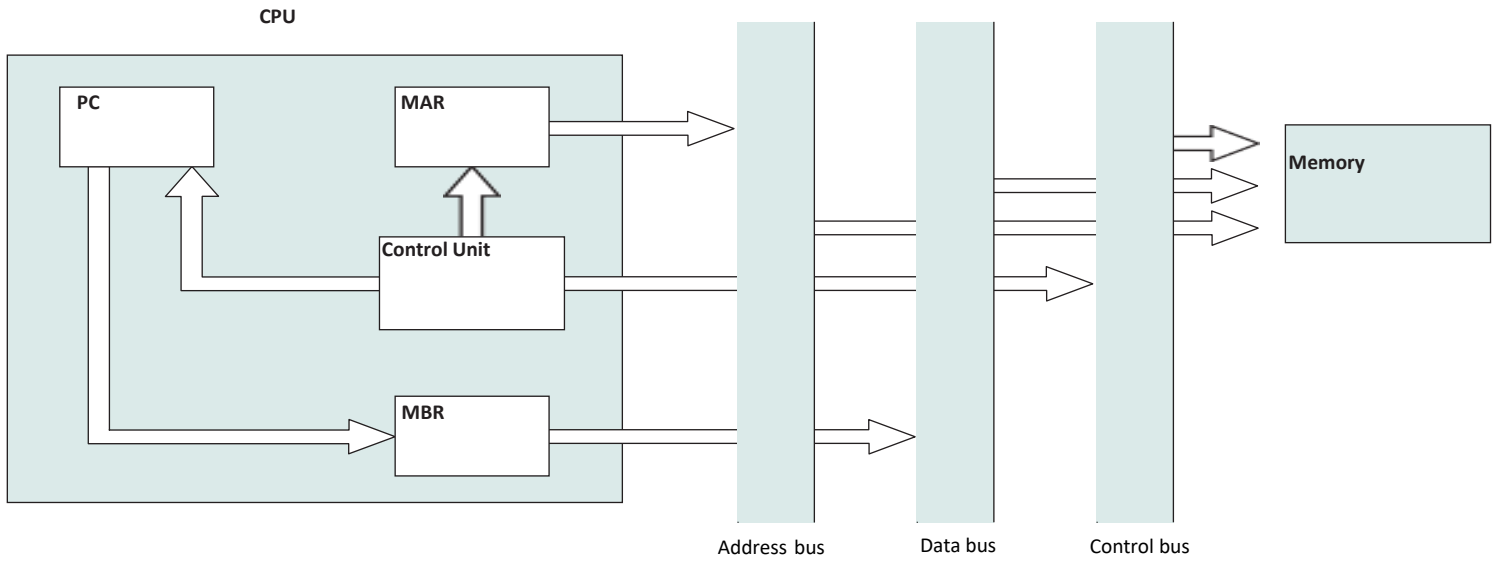
DATA FLOW, INDIRECT CYCLE



DATA FLOW, INTERRUPT CYCLE

- The current contents of the PC must be saved so that the processor can resume normal activity after the interrupt.
- Thus, the contents of the PC are transferred to the MBR to be written into memory.
- The special memory location reserved for this purpose is loaded into the MAR from the control unit. It might, for example, be a stack pointer.
- The PC is loaded with the address of the interrupt routine.
- As a result, the next instruction cycle will begin by fetching the appropriate instruction.

DATA FLOW, INTERRUPT CYCLE



AGENDA

- Processor Organization
- Register Organization
- Instruction Cycle
- Parallel Processor Systems
  - Types of Parallel Processor Systems
  - Parallel Organizations

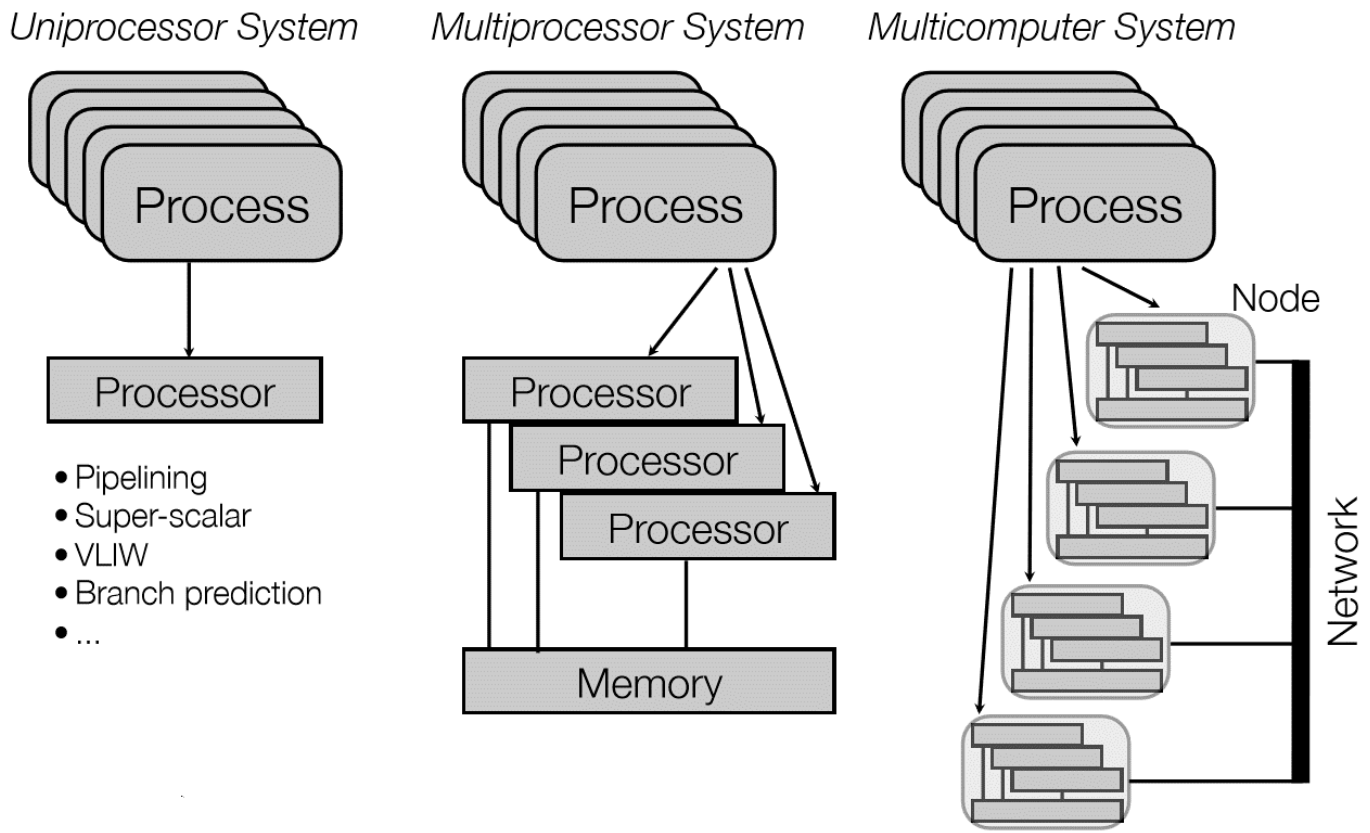
PARALLEL PROCESSOR SYSTEMS

- At the micro-operation level, multiple control signals are generated at the same time.
- Instruction pipelining, at least to the extent of overlapping fetch and execute operations, is examples of performing functions in parallel. This approach is taken further with superscalar organization, which exploits instruction- level parallelism.
- With a superscalar machine, there are multiple execution units within a single processor, and these may execute multiple instructions from the same program in parallel.

REASON FOR CHOOSING A PARALLEL ARCHITECTURE

- Performance - do it faster
- Throughput - do more of it in the same time
- Availability - do it without interruption
- Price / performance - do it as fast as possible for the given money
- Scalability - be able to do it faster with more resources
- Scavenging - do it with what I already have

PARALLEL COMPUTERS



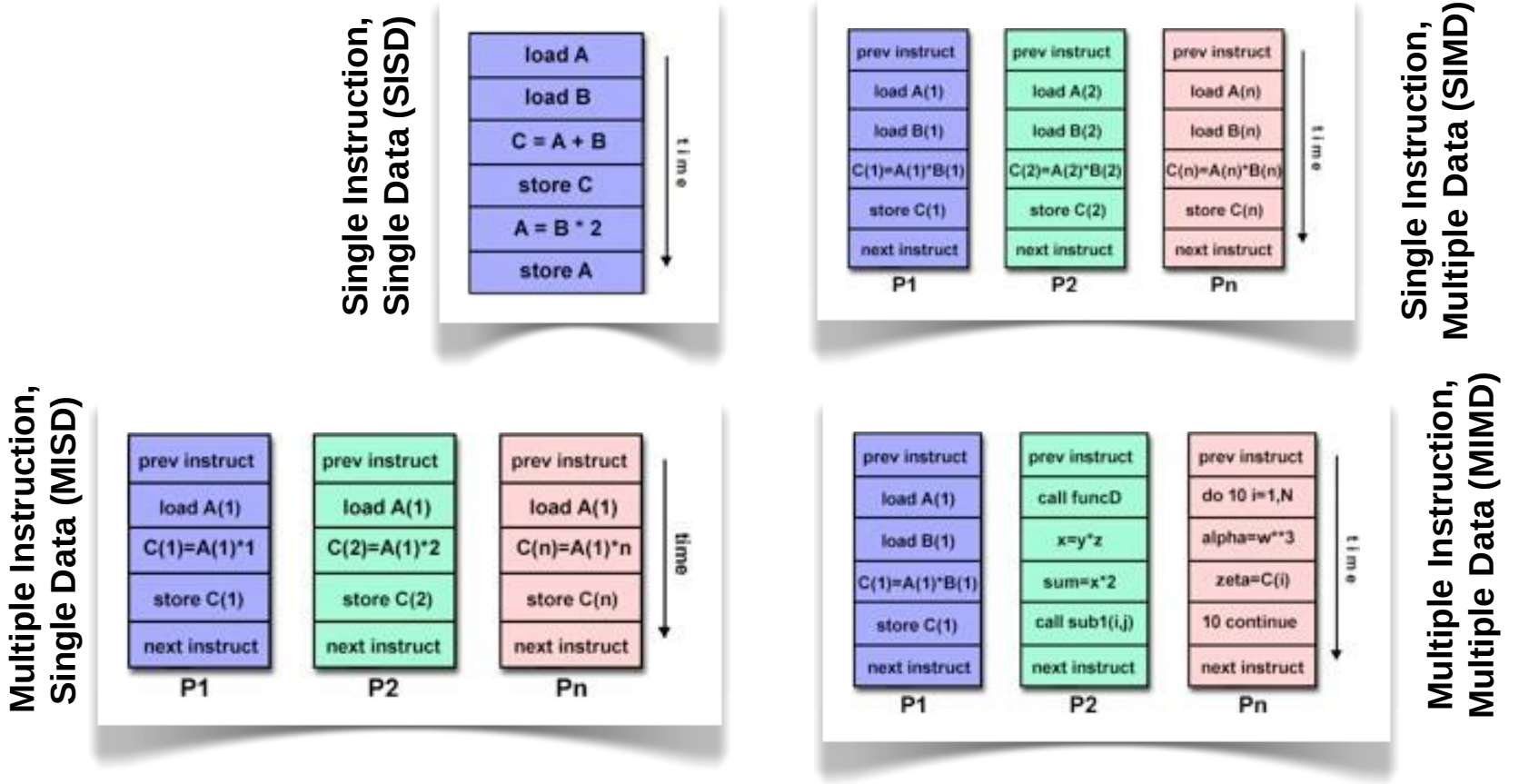
TYPES OF PARALLEL PROCESSOR SYSTEMS

Flynn proposed the following categories of computer systems:

- Single instruction, single data (**SISD**) stream: A single processor executes a single instruction stream to operate on data stored in a single memory. Uniprocessors fall into this category.
- Single instruction, multiple data (**SIMD**) stream: A single machine instruction controls the simultaneous execution of a number of processing elements on a lockstep basis. Each processing element has an associated data memory, so that instructions are executed on different sets of data by different processors. Vector and array processors fall into this category.
- Multiple instruction, single data (**MISD**) stream: A sequence of data is trans- mitted to a set of processors, each of which executes a different instruction sequence. This structure is not commercially implemented.
- Multiple instruction, multiple data (**MIMD**) stream: A set of processors simultaneously execute different instruction sequences on different data sets. SMPs, clusters, and NUMA systems fit into this category.

MULTIPROCESSOR: FLYNN'S TAXONOMY (1966)

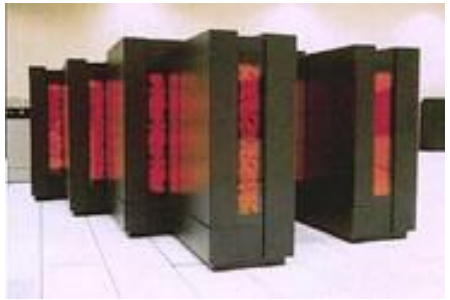
- Classify multiprocessor architectures among **instruction** and data **dimension**



SIMD EXAMPLES

ILLIAC IV (1974)

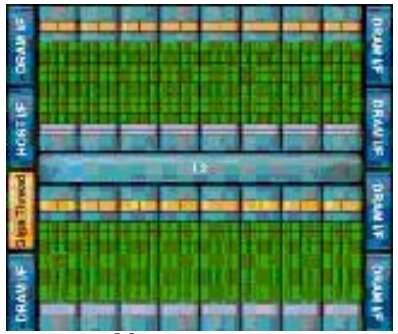
Cray Y-MP



Thinking Machines CM-2 (1985)

- Good for problems with high degree of regularity, such as graphics/image processing
- Synchronous (lockstep) and deterministic execution
- Typically exploit data parallelism
- Today: GPGPU Computing, Cell processor, SSE, AltiVec

Fermi GPU



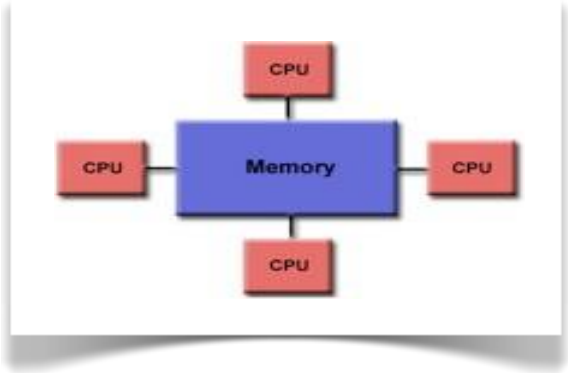
MULTIPLE INSTRUCTION MULTIPLE DATA (MIMD)

- Most common parallel hardware architecture today
  - Example: All many-core processors, clusters, distributed systems
- From software perspective [Pfister]
  - **SPMD - Single Program Multiple Data**
    - Sometimes denoted as ‚application cluster‘
    - Examples: Load-balancing cluster or failover cluster for databases, web servers, ...
  - **MPMD - Multiple Program Multiple Data**
    - Multiple implementations work together on one parallel computation
    - Example: Master / worker cluster, map / reduce framework

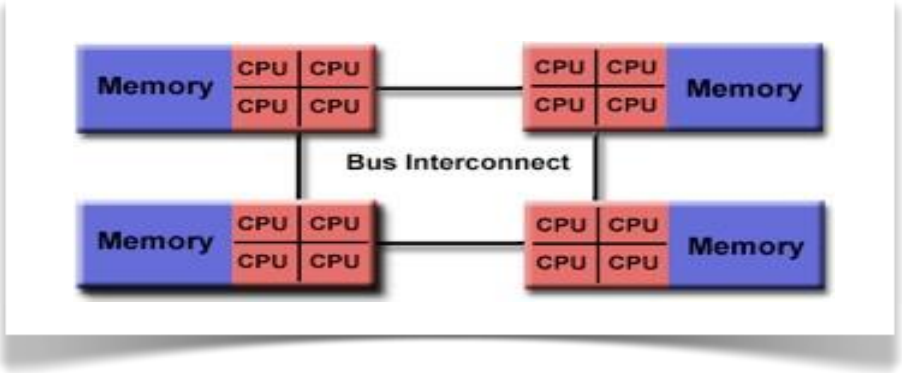
MIMD ARCHITECTURES

- Symmetric multiprocessors (SMP) and massively parallel processors (MPP) are MIMD architectures that differ in how they use memory.
- SMP systems share the same memory and MPP do not.
- An easy way to distinguish SMP from MPP is:
  - SMP $\Rightarrow$ fewer processors + shared memory + communication via memory
  - MPP $\Rightarrow$ many processors + distributed memory + communication via network (messages)

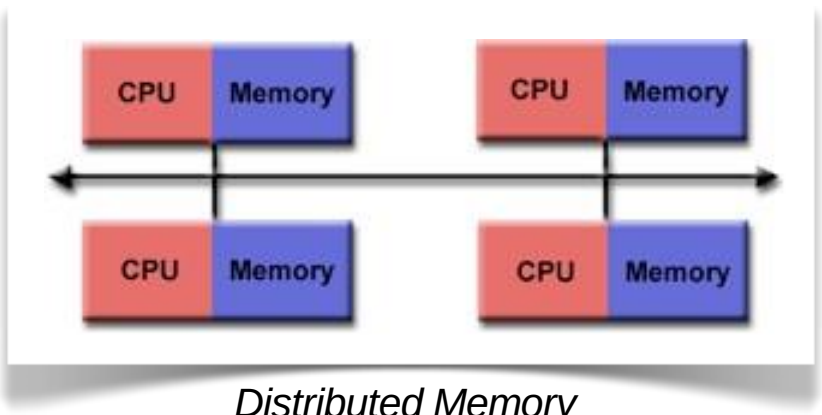
MEMORY ARCHITECTURES



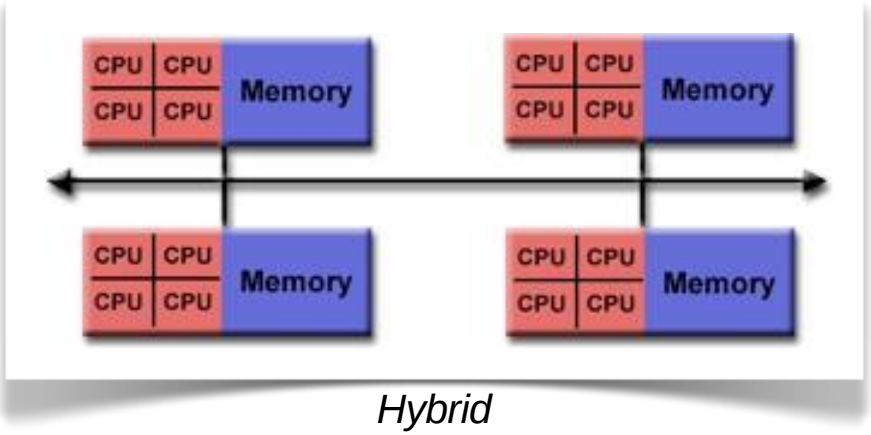
Uniform Memory Access (UMA)



Non-Uniform Memory Access (NUMA)



Distributed Memory

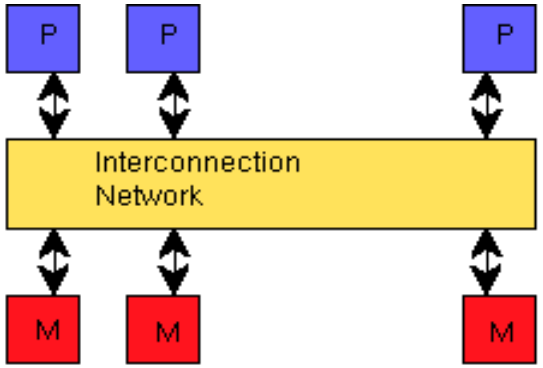


Hybrid

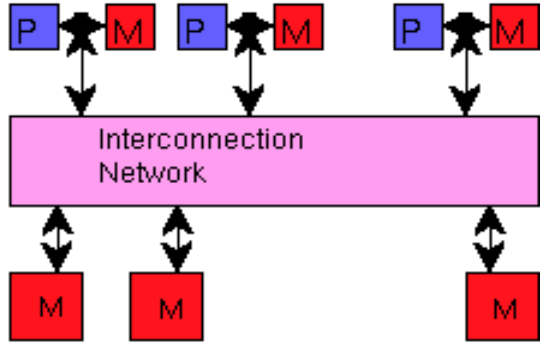
SHARED MEMORY ARCHITECTURES

- **Uniform memory access (UMA) system**
  - Equal load and store access for all processors to all memory
  - Default approach for majority of SMP systems in the past
- **Non-uniform memory access (NUMA) system**
  - Delay on memory access according to the accessed region
  - Typically realized by processor interconnection network and local memories
  - Cache-coherent NUMA (CC-NUMA), completely implemented in hardware
  - About to become standard approach with recent X86 chips

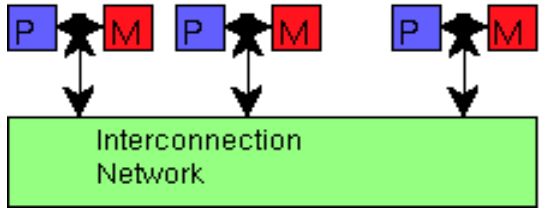
NUMA CLASSIFICATION



Uniform-memory-access
shared-address-space
computer (UMA)



Non-Uniform-memory-access
shared-address-space
computer with local and
global memories (NUMA)

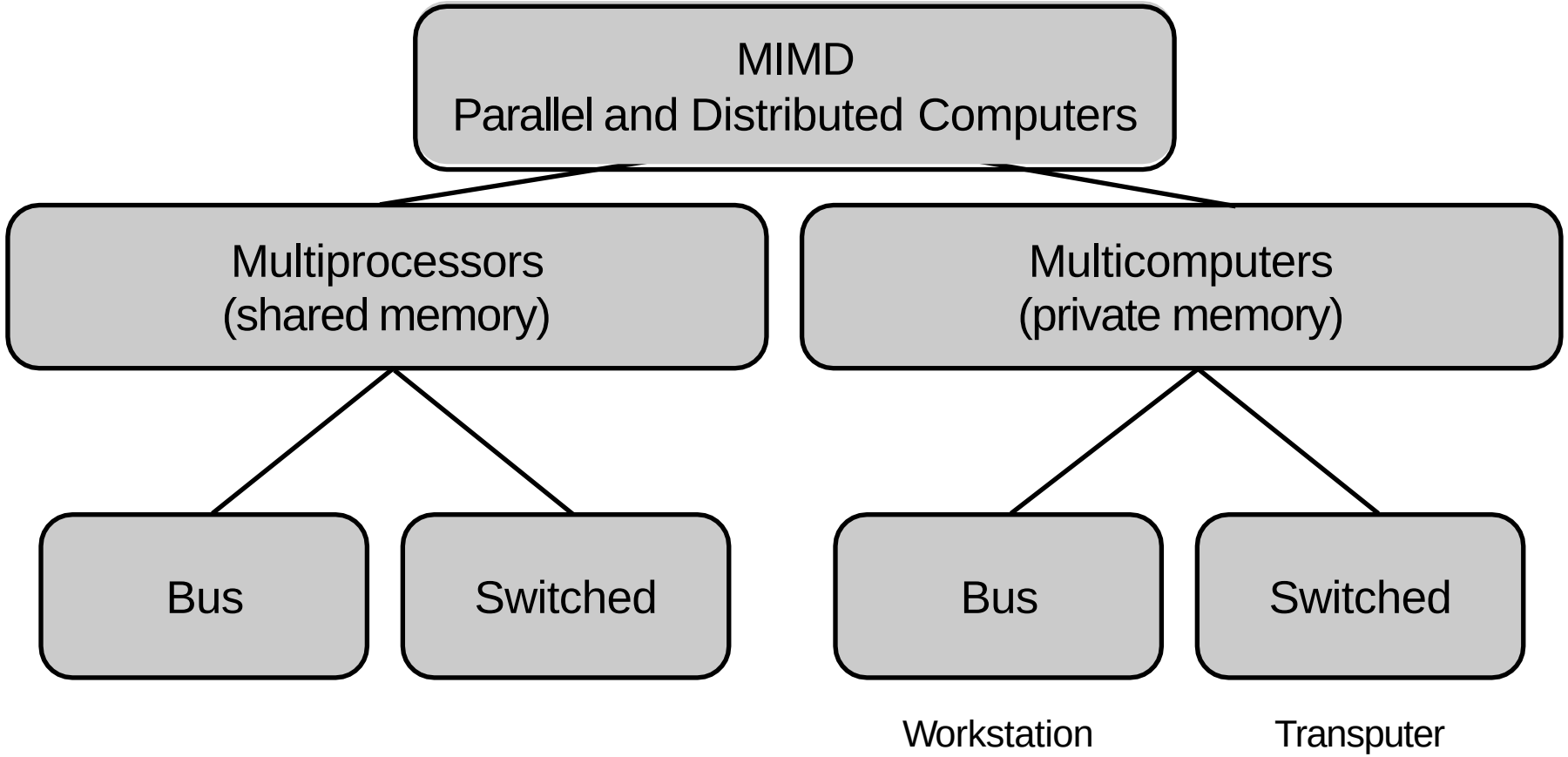


Non-uniform-memory-access
shared-address-space
computer with
local memory only (NUMA)

SHARED MEMORY VS. DISTRIBUTED MEMORY SYSTEM

- **Shared memory (SM) systems**
  - SM-SIMD: Single CPU vector processors
  - SM-MIMD: Multi-CPU vector processors, OpenMP
  - Variant: Clustered shared-memory systems (NEC SX-6, CraySV1ex)
- **Distributed memory (DM) systems**
  - DM-SIMD: processor-array machines; lock-step approach; front processor and control processor
  - DM-MIMD: large variety in interconnection networks
- **Distributed (Virtual) shared-memory systems**
  - High-Performance Fortran, TreadMarks

ANOTHER TAXONOMY (TANENBAUM)

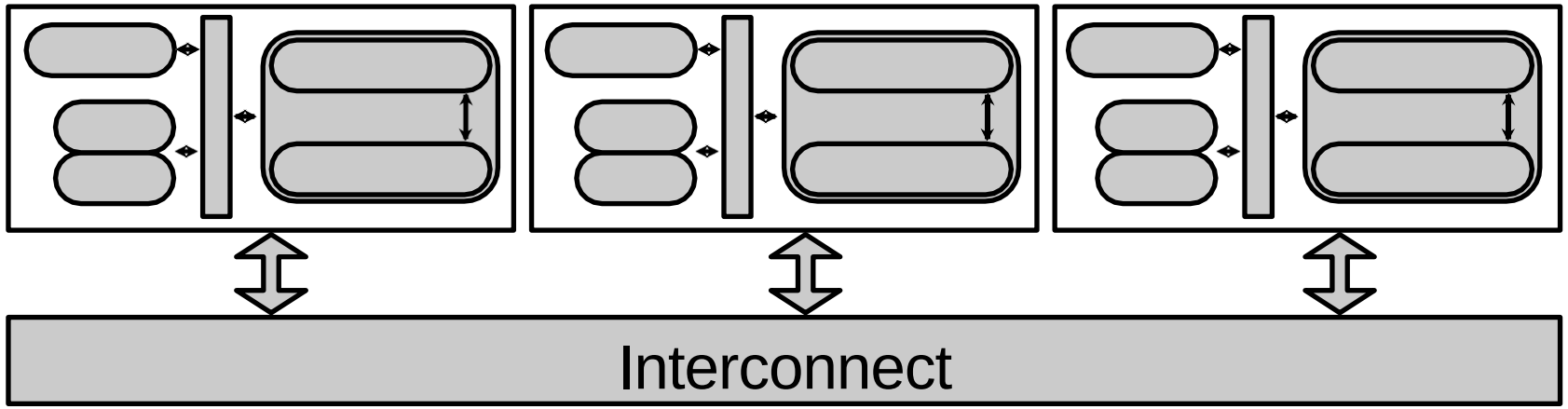


MULTIPROCESSOR SYSTEMS

- **Symmetric Multiprocessing (SMP)**
  - Set of equal processors in one system (more SM-MIMD than SIMD)
  - Processors share access to main memory over one bus
  - Today, every SMP application also works on a uniprocessor machine
- **Asymmetric multiprocessing (ASMP)**
  - Specialized processors for I/O, interrupt handling or operating system (DEC VAX 11, OS-360, IBM Cell processor)
  - Typically master processor with main memory access and slaves
- Large multiprocessor work with NUMA / COMA memory hierarchy

MULTICOMPUTER SYSTEMS

- Number of von Neumann computers, connected by a network (DM-MIMD)
- Each computer runs own program and sends / receives messages
- Local memory access is less expensive than remote memory access



MULTICOMPUTER SYSTEMS – MASSIVELY PARALLEL PROCESSING (MPP)

- Specific non-standardized interconnection network
  - Low latency, high speed
  - Distributed file system
- Specific packaging of components and nodes for cooling and upgradeability
  - Whole system provided by one vendor (IBM, HP)
  - Extensibility as major issue, in order to save investment
- Distributed processing as extension of **DM-MIMD**
- Single System View
  - Common file system, central job scheduling

MULTICOMPUTER SYSTEMS - CLUSTERS

- Collection of stand-alone workstations/PC‘s connected by a local network
  - Cost-effective technique to connect small-scale computers to a large-scale parallel computer
  - Low cost of both hardware and software
  - **Users are builders**, have control over their own system (hardware infrastructure and software), low costs as major issue
- Distributed processing as extension of **DM-MIMD**
  - Communication between processors is orders of magnitude slower
  - PVM, MPI as widely accepted programming standards
    - Used with cheap LAN hardware

DISTRIBUTED SYSTEM

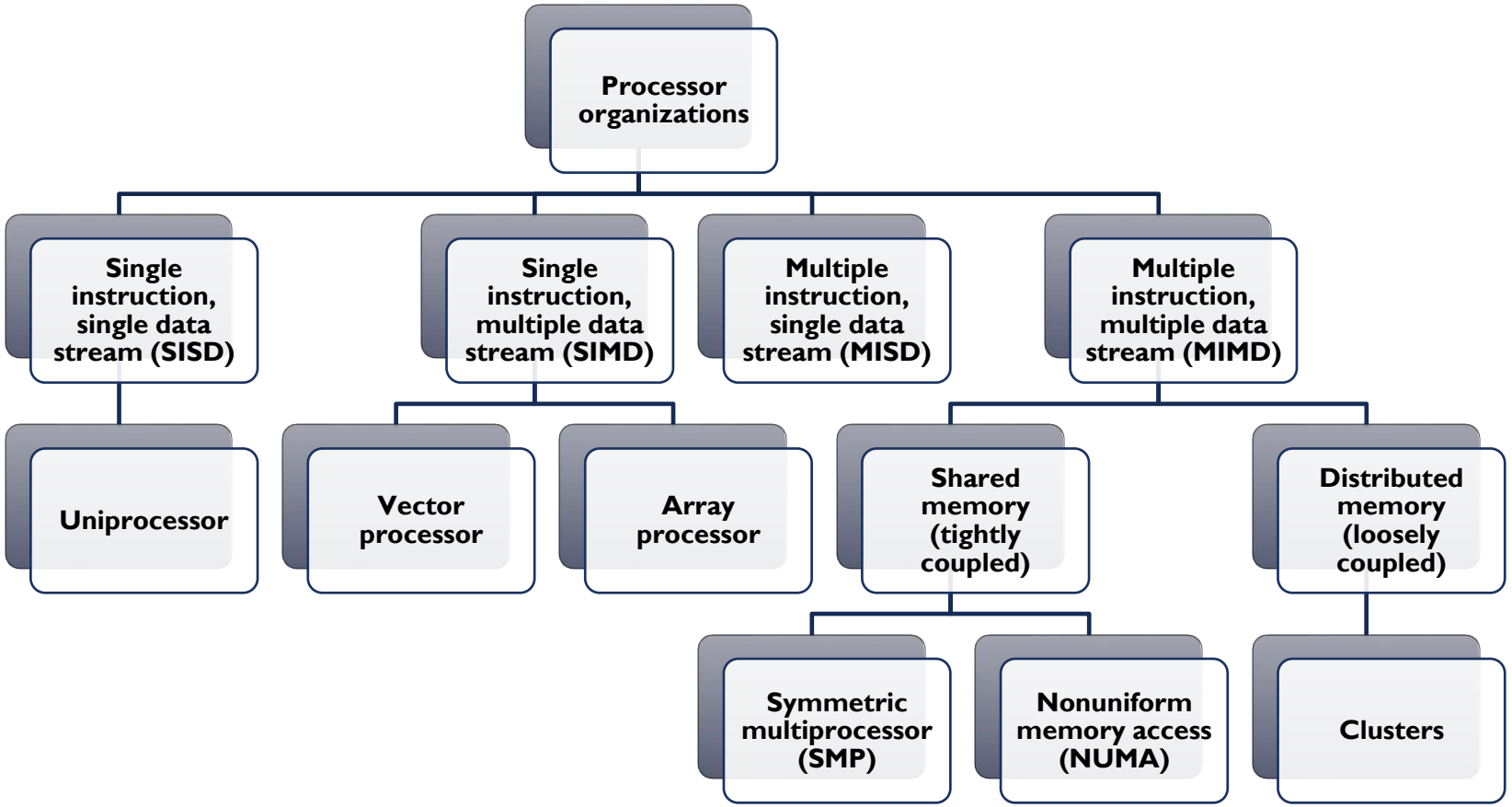
- Tanenbaum (Distributed Operating Systems):
  „A distributed system is a collection of independent computers that appear to the users of the system as a single computer.“
- Coulouris et al.:
  „… [system] in which hardware or software components located at networked computers communicate and coordinate their actions only by passing messages.“
- Lamport:
  „A distributed system is one in which the failure of a computer you didn't even know existed can render your own computer unusable.”
- Consequences: concurrency, no global clock, independent failures
- Challenges: heterogeneity, openness, security, scalability, failure handling, concurrency, need for transparency

SMP VS. CLUSTER VS. DISTRIBUTED SYSTEM

- Clusters are composed of computers, SMPs are composed of processors
  - High availability is cheaper with clusters, but demands other software
  - Scalability is easier with a cluster
  - SMPs are easier to maintain from administrators point of view
  - Software licensing becomes more expensive with a cluster
- Cluster vs. Distributed System
  - Both contain of multiple nodes for parallel processing
  - Nodes in a distributed system have their own identity
  - Physical vs. virtual organization

	MPP	SMP	Cluster	Distributed
Number of nodes	O(100)-O(1000)	O(10)-O(100)	O(100) or less	O(10)-O(1000)
Node Complexity	Fine grain	Medium or coarse grained	Medium grain	Wide range
Internode communication	Message passing / shared variables (SM)	Centralized and distributed shared memory	Message Passing	Shared files, RPC, Message Passing, IPC
Job scheduling	Single run queue on host	Single run queue mostly	Multiple queues but coordinated	Independent queues
Address Space	Multiple	Single	Multiple or single	Multiple
Ownership	One organization	One organization	One or many organizations	Many organizations

PARALLEL ORGANIZATIONS



Thank You