Advanced Architectures

- RISC versus CISC Technologies
- Flynn's taxonomy for multiple processor architecture

RISC versus CISC

- Learn the properties that often distinguish RISC from CISC architectures.
  - **RISC** = Reduced Instruction Set computer
  - **CISC** = Complex Instruction Set Computer

RISC Machines: Definition and Difference from CISC

- The underlying **philosophy of RISC** machines
  - It is better able to manage program execution when the program consists of only a **few different instructions** that are the **same length** and require the **same number of clock cycles** to decode and execute.
- In CISC systems, many different kinds of instructions access memory, making instruction length variable and fetch-decode-execute time unpredictable.
- RISC systems access memory only with explicit ***load*** and ***store*** instructions.

RISC Machines: Strategy to Enhancing Computing Performance

- The difference between CISC and RISC becomes evident through the basic computer performance equation:

$$\text{CPU Time} = \frac{\text{seconds}}{\text{program}} = \frac{\text{instructions}}{\text{program}} \times \frac{\text{avg. cycles}}{\text{instruction}} \times \frac{\text{seconds}}{\text{cycle}}$$

- RISC systems shorten execution time by reducing the clock cycles per instruction.
- CISC systems improve performance by reducing the number of instructions per program.

RISC Machines: Enable hardwired control units

- The simple instruction set of **RISC machines enables control units to be hardwired** for maximum speed.
- The more complex and variable length instruction set of **CISC machines requires microcode-based control units** that interpret instructions as they are fetched from memory. This translation takes time.
  - Only around 20% of all instructions are used most of the time in CISC
- With **fixed-length instructions, RISC lends itself to pipelining and speculative execution**.

RISC Machines: Less clocks per instruction

- Consider the the program fragments:

CISC

```
mov ax, 10
mov bx, 5
mul bx, ax
```

RISC

```
        mov ax, 0
        mov bx, 10
        mov cx, 5
Begin   add ax, bx
        loop Begin
```

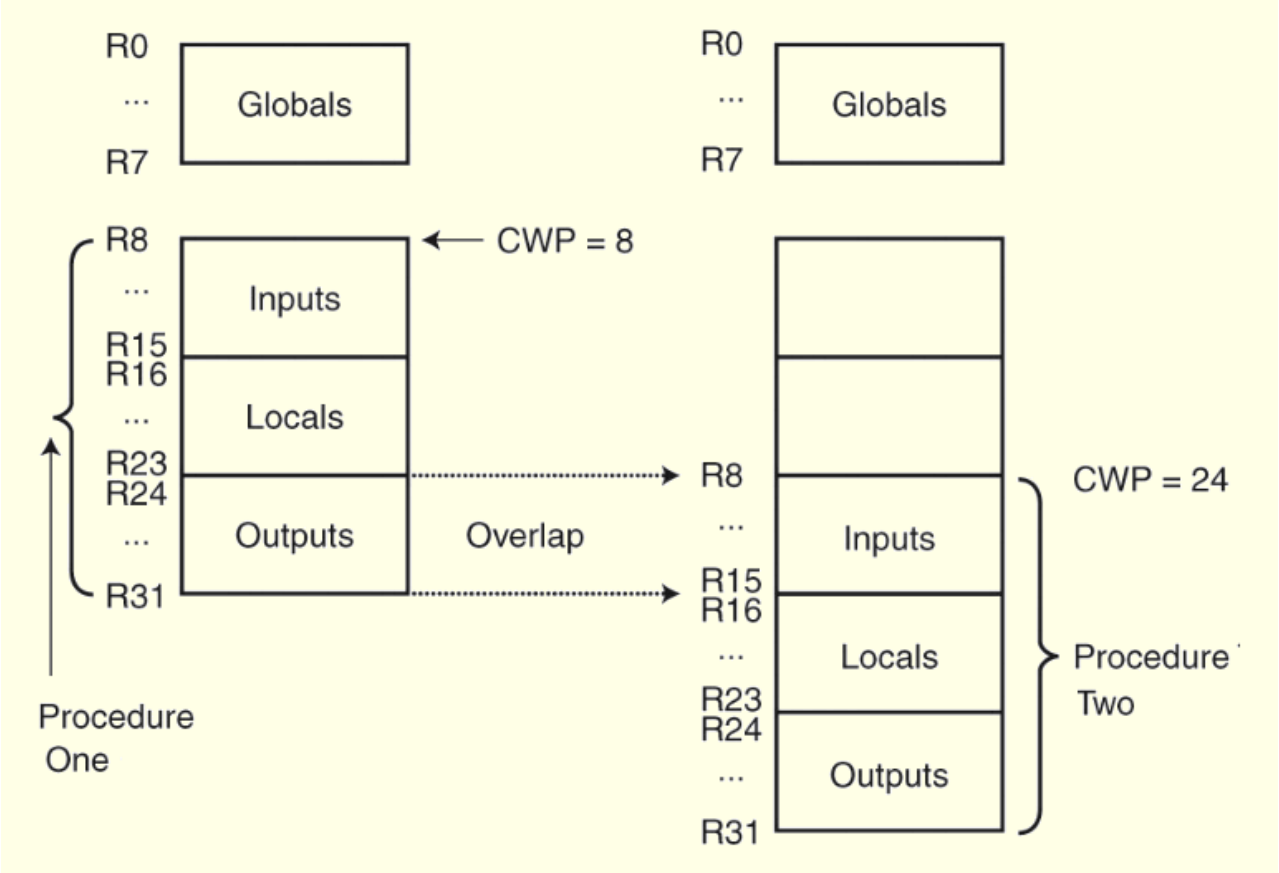
- The total clock cycles for the CISC version might be:
  `(2 movs × 1 cycle) + (1 mul × 30 cycles) = 32 cycles`
- While the clock cycles for the RISC version is:
  `(3 movs × 1 cycle) + (5 adds × 1 cycle) + (5 loops × 1 cycle) = 13 cycles`
- With RISC clock cycle being shorter, RISC gives us much faster execution speeds.

RISC Machines: LOAD-STORE ISAs

- Because of their **load-store ISAs**, RISC architectures **require a large number of CPU registers**.
- These **register provide fast access to data** during sequential program execution.
- They can also be employed to **reduce the overhead typically caused by passing parameters to subprograms**.
  - Instead of pulling parameters off of a stack, the **subprogram is directed to use a subset of registers**.

Overlapping Register Windows in RISC Machine

- This is how registers can be overlapped in a RISC system.
- The ***current window pointer*** (CWP) points to the active register window.



- There might virtually have, e.g. 24 set of 32 registers each.

RISC Machines: Converging with CISC

- It is becoming increasingly difficult to distinguish RISC architectures from CISC architectures.
- Some RISC systems provide more extravagant instruction sets than some CISC systems.
  - The RISC PowerPC has a lager ISA than the CISC Pentium
- Register usage and Load/Store architecture is more prominent
- Some systems combine both approaches.

RISC Machines: Comparisons

- **RISC**
  - Multiple register sets.
  - Three operands per instruction.
  - Parameter passing through register windows.
  - Single-cycle instructions.
  - Hardwired control.
  - Highly pipelined.
- **CISC**
  - Single register set.
  - One or two register operands per instruction.
  - Parameter passing through memory.
  - Multiple cycle instructions.
  - Microprogrammed control.
  - Less pipelined.

Continued....

RISC Machines

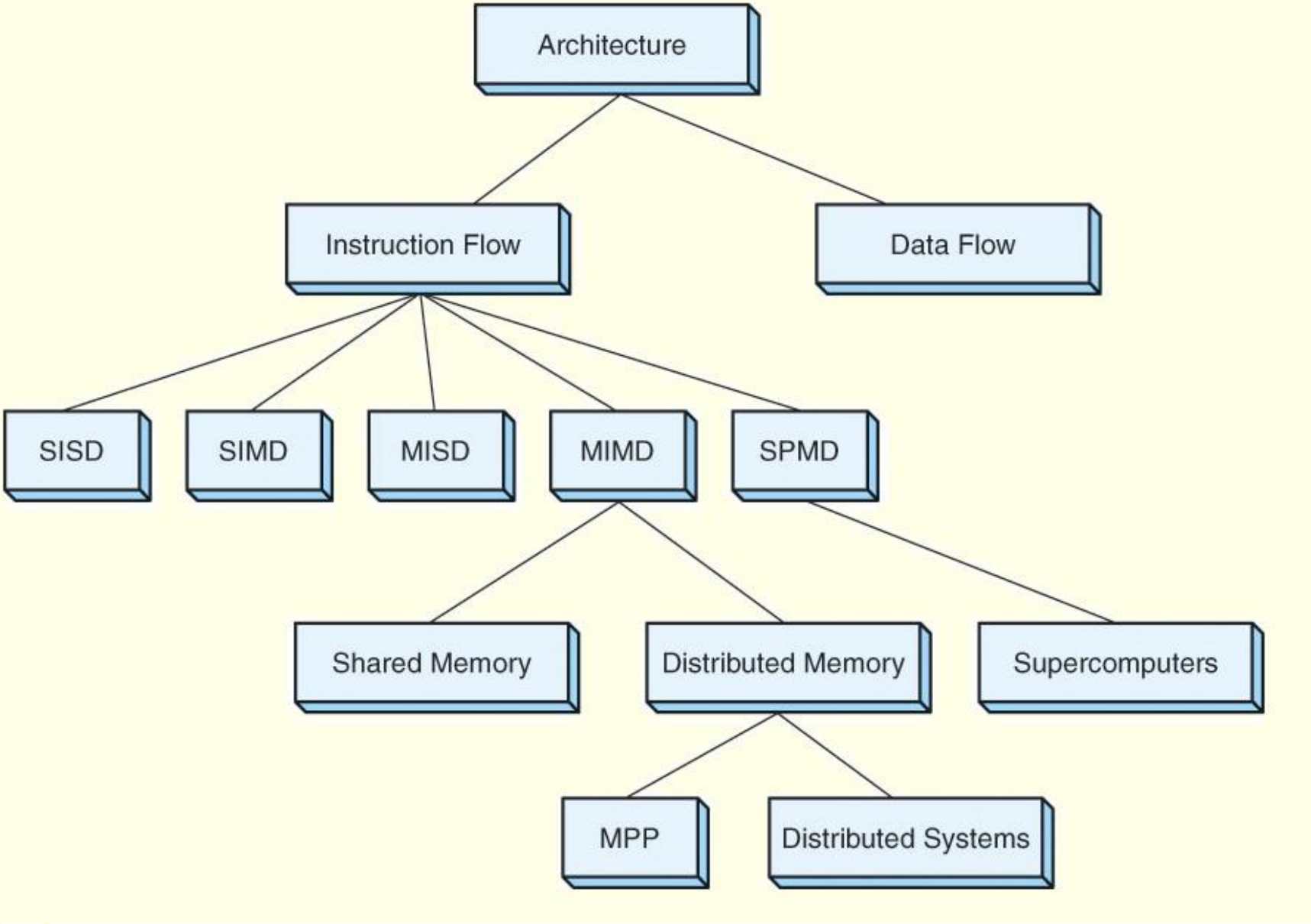
- **RISC**
  - Simple instructions, few in number.
  - Fixed length instructions.
  - Complexity in compiler.
  - Only LOAD/STORE instructions access memory.
  - Few addressing modes.
- **CISC**
  - Many complex instructions.
  - Variable length instructions.
  - Complexity in microcode.
  - Many instructions can access memory.
  - Many addressing modes.

Parallel processing

Flynn's Taxonomy

- Many attempts have been made to come up with a way to categorize computer architectures.
- *Flynn's Taxonomy* has been the most enduring of these, despite having some limitations.
- Flynn's Taxonomy **takes into consideration the number of processors and the number of data paths** incorporated into an architecture.
- A machine can have one or many processors that operate on one or many data streams.

Flynn's Taxonomy

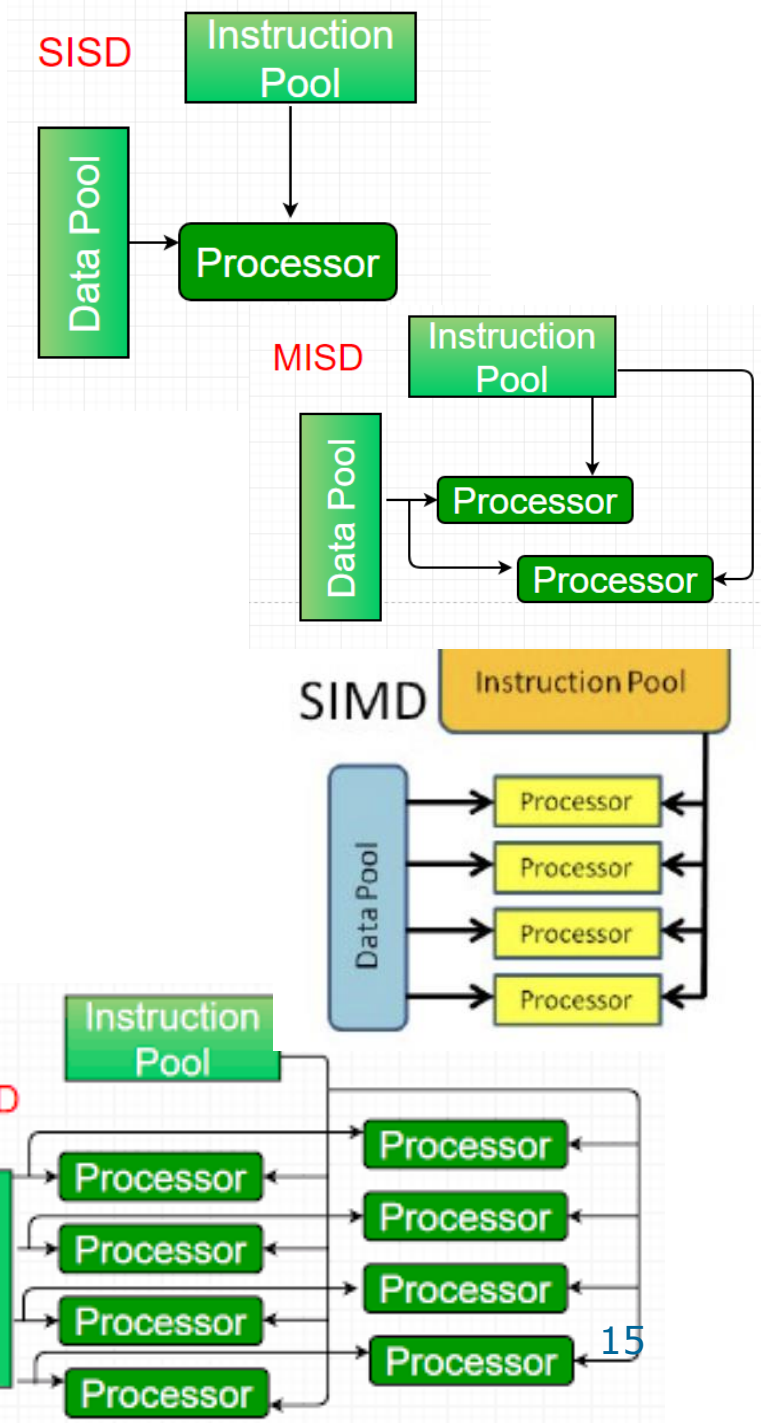


Flynn's Taxonomy

- The four combinations of multiple processors and multiple data paths are described by Flynn as:
  - **SISD:** Single instruction stream, single data stream. These are classic uniprocessor systems.
  - **MISD:** Multiple instruction streams, single data stream. → Rare architecture.

- SIMD: Single instruction stream, multiple data streams. Execute the same instruction on multiple data values, as in vector processors.

- MIMD: Multiple instruction streams, multiple data streams. These are today's parallel architectures.



Flynn's Taxonomy: The Fall Shorts

Flynn's Taxonomy falls short in a number of ways:

- **First**, there appears to be no need for MISD machines.
- **Second**, parallelism is not homogeneous. This assumption ignores the contribution of specialized processors.
  - E.g. a separate floating-point adders/multipliers, integer units, etc.
- **Third**, it provides no straightforward way to distinguish architectures of the MIMD category.
  - One idea is to divide these systems into those that share memory, and those that don't, as well as whether the interconnections are bus-based or switch-based.

Flynn's Taxonomy: Extended MIMD for SMP vs MPP

- 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:

  MPP $\Rightarrow$ many processors + distributed memory + communication via network

  SMP $\Rightarrow$ fewer processors + shared memory + communication via memory (e.g. variables)

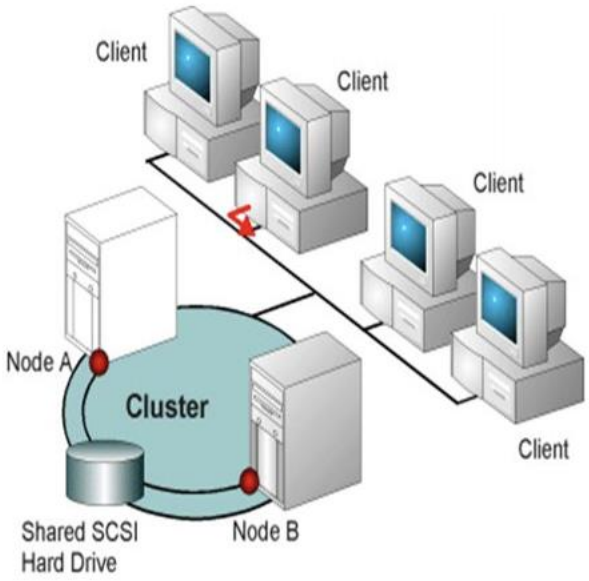
.3 Flynn's Taxonomy: MIMD-Distributed Computing

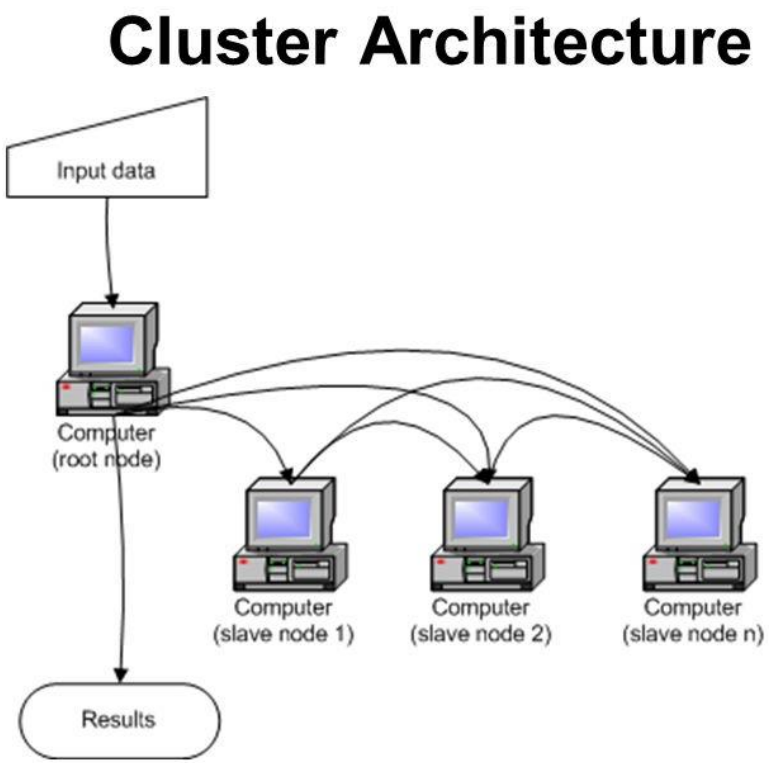
- Other examples of MIMD architectures are found in distributed computing, where **processing takes place collaboratively among networked computers**.
  - A **network of workstations** (NOW) uses otherwise idle systems to solve a problem.

- A **collection of workstations** (COW) is a NOW where one workstation coordinates the actions of the others.

- A **dedicated cluster parallel computer** (DCPC) is a group of workstations brought together to solve a specific problem.

- A **pile of PCs** (POPC) is a cluster of (usually) heterogeneous systems that form a dedicated parallel system.





Parallel and Multiprocessor Architectures

- Parallel processing is capable of:
  - Economically increasing system throughput
  - Providing better fault tolerance.
- The limiting factor is that no matter how well an algorithm is parallelized, there is always some portion that must be done sequentially.

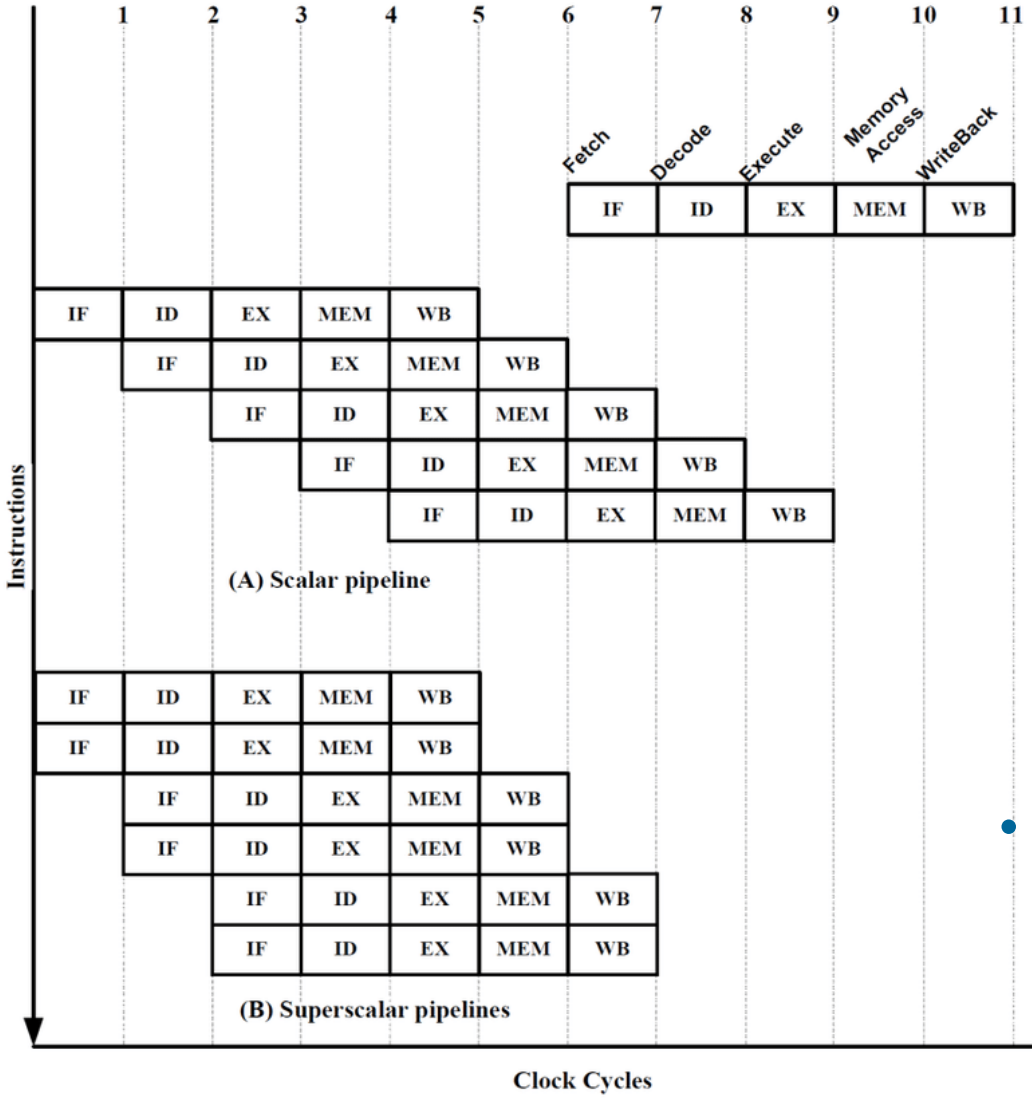
keep in mind that an n-fold increase in processing power does not necessarily result in an n-fold increase in throughput

Super-pipelining

- Recall →
  - pipelining divides the fetch-decode-execute cycle into smaller stages that carry out a small part of the process on a set of instructions.
  - an instruction exits the pipeline during **each tick of the clock**, idealistically.
- *Superpipelining* → a pipeline has stages that require less than half a clock cycle to complete.
  - We equip the pipeline with a separate clock

Superpipelining is only one aspect of superscalar design (e.g. IBM's PowerPC, Sun's UltraSparc, DEC's Alpha.)

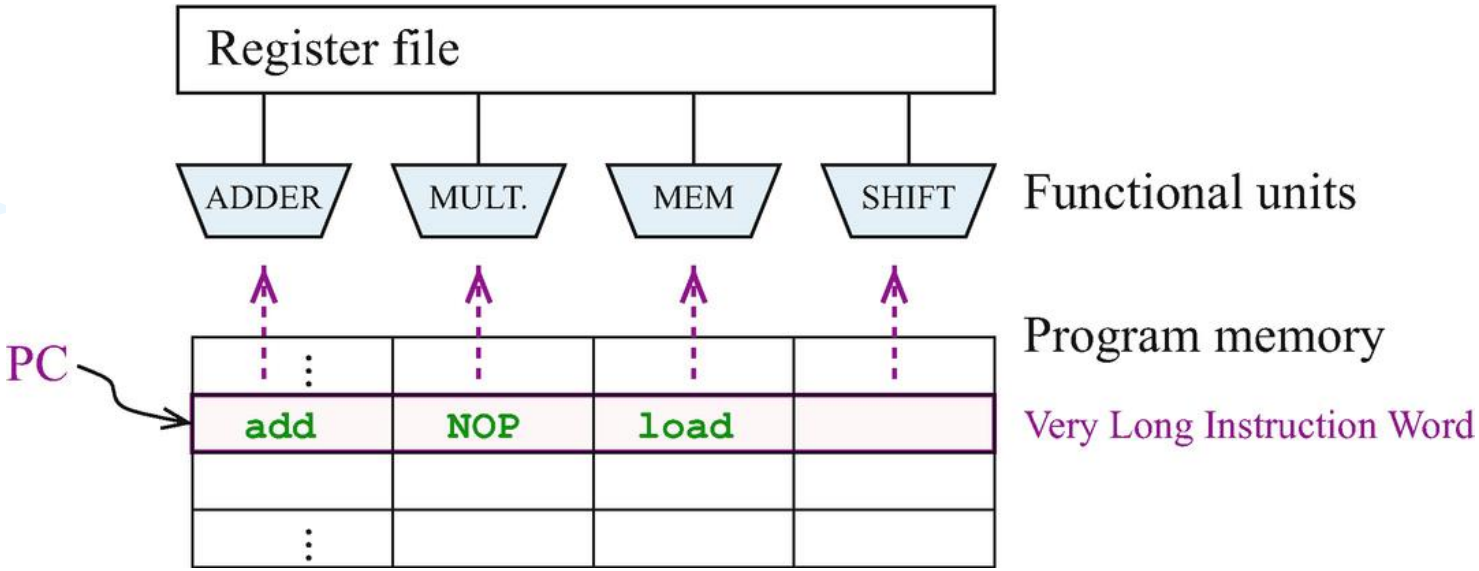
Super scalar architecture



- Integrates multiple execution units, e.g. specialized integer and floating-point adders and multipliers.
  - A critical component is the instruction fetch unit,
    - Able to simultaneously retrieve several instructions from memory.
  - A *decoding unit* determines which of these instructions can be executed in parallel and combines them accordingly.
- compilers that make optimum use of the hardware is required.

Very Long Instruction Word (VLIW)

- VLIW architecture **relies on compiler to pack independent instructions** into one long instruction that is sent down the pipeline to the execution units, instead of a hardware decoding unit as done in SuperScalar.
- Pros and Cons
  - Pro: Compiler can better identify instruction dependencies.
  - Con: Compilers tend to be conservative and cannot have a view of the run time code.



Vector computers

http://en.wikipedia.org/wiki/Vector_processor

Vector computers are processors that **operate on entire vectors or matrices at once.**

- These systems are **often called supercomputers.**

Vector computers are **highly pipelined** so that arithmetic instructions can be overlapped.

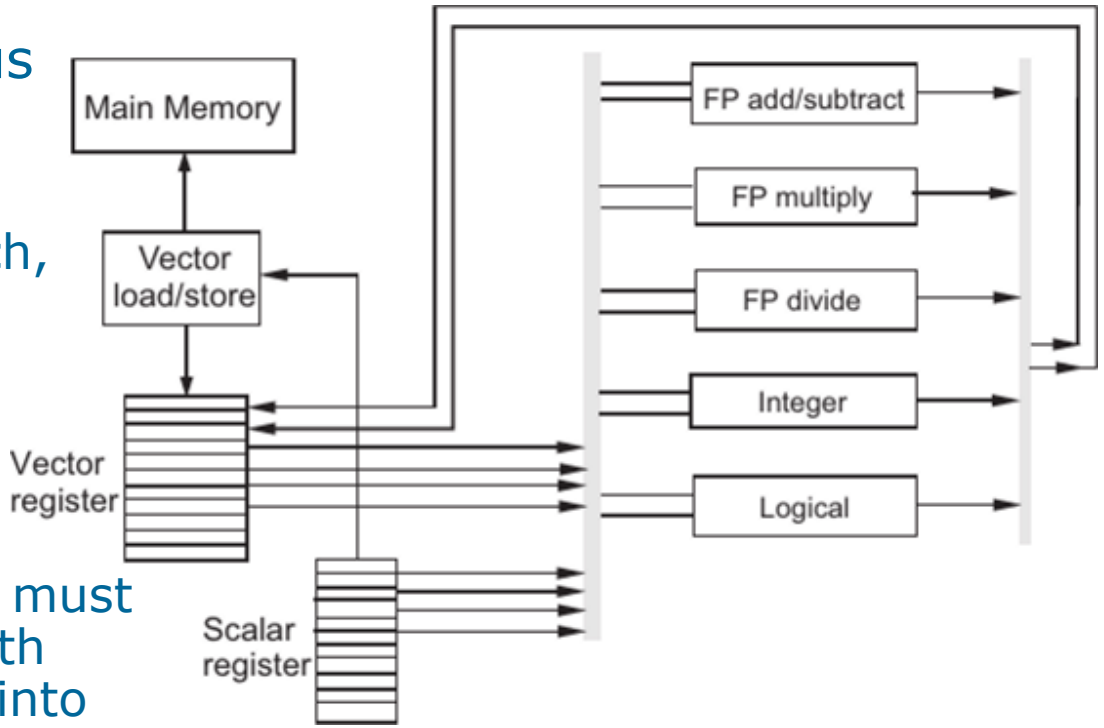
Vector processors can be categorized according to how operands are accessed.

- **Register-register** vector processors require all operands to be in registers.
- **Memory-memory** vector processors allow operands to be sent from memory directly to the arithmetic units.

https://www.geeksforgeeks.org/vector-processor-classification/

Vector computers (cont.)

- By presuming a continuous stream of data, vector machines are efficient
  - Fewer instructions to fetch,
  - Values can be prefetched
- Disadvantage of vector computer:
  - Register-register vector computers: large vectors must be broken into fixed-length segments so they will fit into the register sets.
  - Memory-memory vector computers have a longer startup time before the pipeline becomes full.



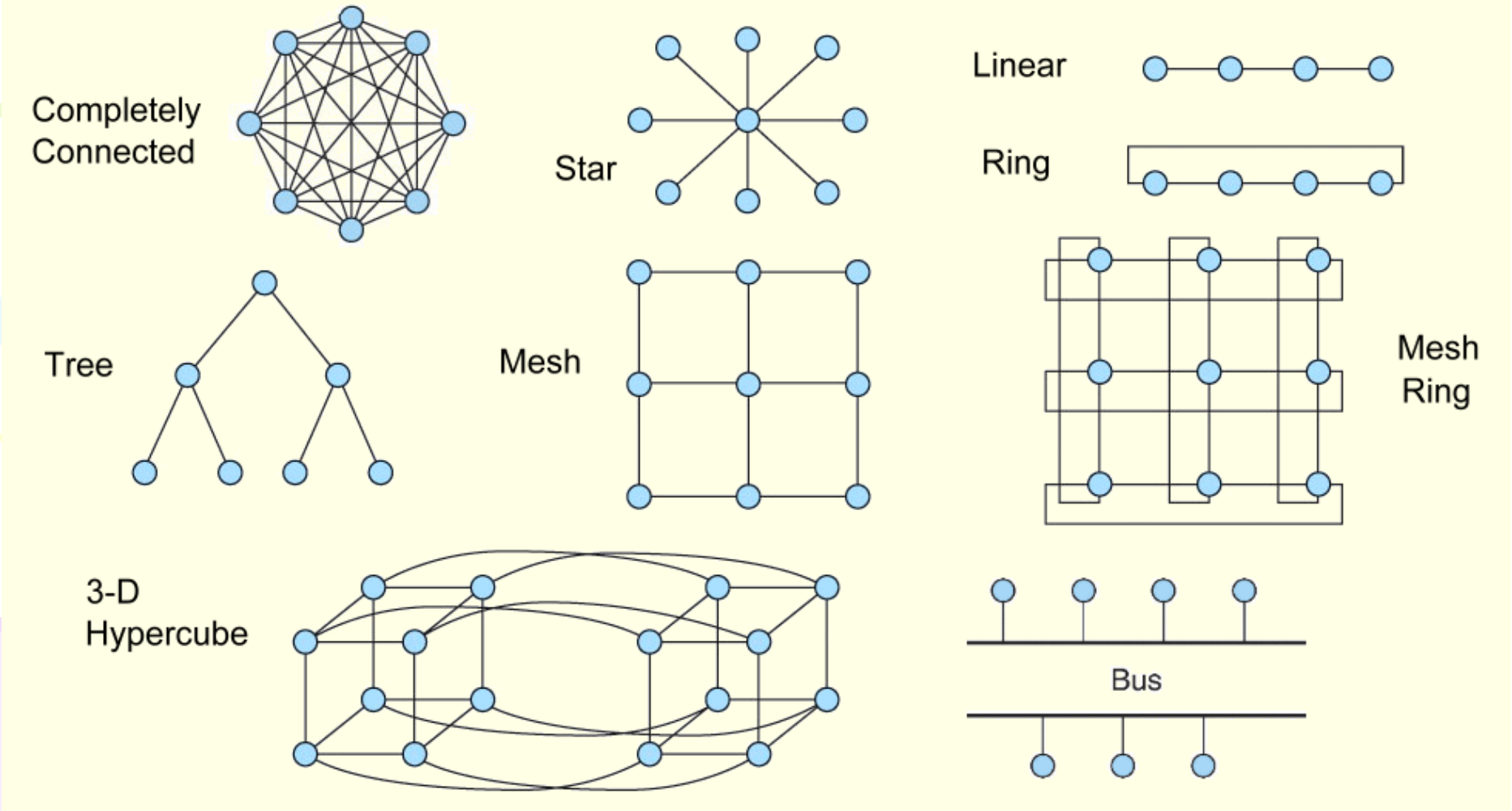
MIMD– Classification of System Interconnection

- MIMD systems can communicate through shared memory or through an interconnection network (distributed systems).
- Interconnection networks are often **classified according to their topology, routing strategy, and switching technique.**
- Of these, the **topology is a major determining factor** in the overhead cost of message passing.
  - Message passing takes time owing to network latency and incurs overhead in the processors.

MIMD– Classification of System Interconnection

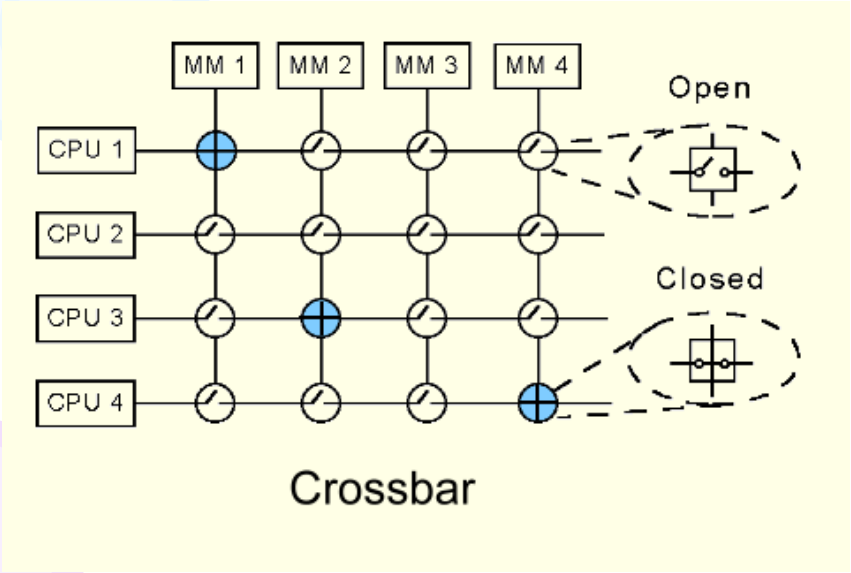
- Interconnection networks can be either **static or dynamic.**
- Processor-to-memory connections usually employ dynamic interconnections. These can be blocking or nonblocking.
  - *Nonblocking interconnections allow connections to occur simultaneously*.
    - E.g. Cross-bar architecture
  - *Example for blocking interconnection is switch*
- Processor-to-processor message-passing interconnections are usually static, and can employ any of several different topologies, as shown on the following slide.

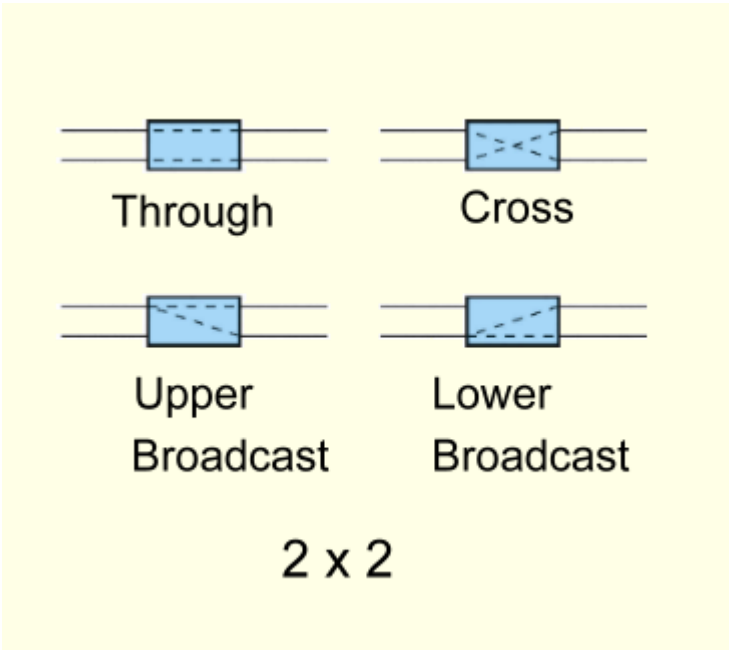
MIMD: Possible Static Interconnection



MIMD– Dynamic interconnection

- Example dynamic routing is achieved through switching networks that consist of crossbar switches or 2 × 2 switches.

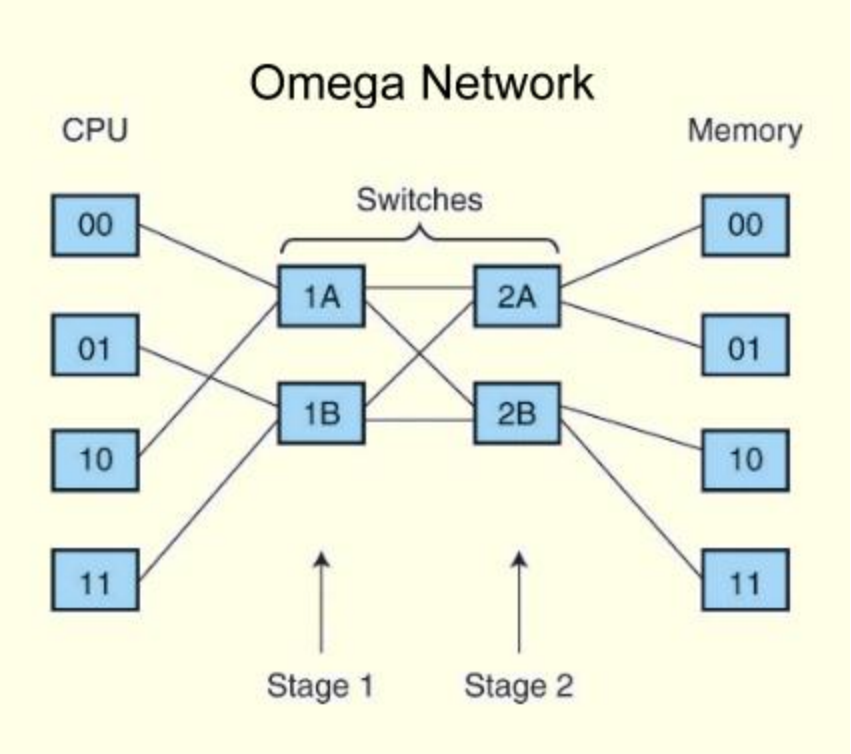




MIMD– Dynamic interconnection

- Multistage interconnection (or shuffle) networks are the most advanced class of switching networks.

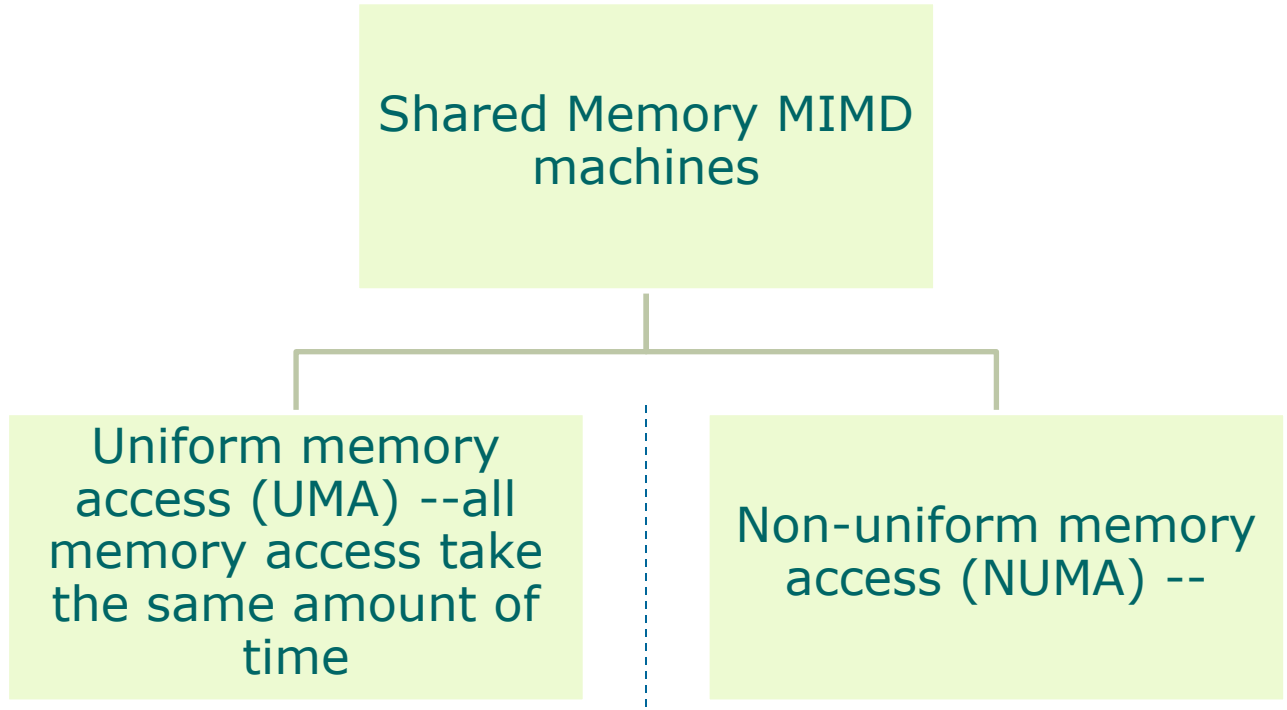
- They can be used in loosely-coupled distributed systems, or in tightly-coupled processor-to-memory configurations.



MIMD– Distributed Shared Memory Multiprocessors

- Tightly-coupled multiprocessor systems use the same memory. They are also referred to as shared memory multiprocessors.
- The processors do not necessarily have to share the same block of physical memory:
- Each processor can have its own memory, but it must share it with the other processors.
  - Configurations such as these are called *distributed shared memory multiprocessors*.

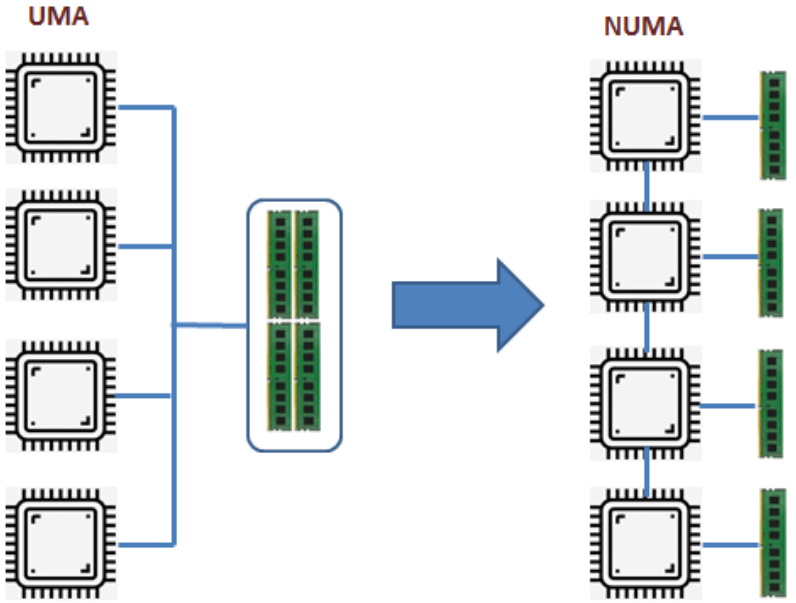
distributed shared memory multiprocessors



- The interconnection network must be fast enough to support multiple concurrent accesses to memory, or it will slow down the whole system.
- Thus, the interconnection network limits the number of processors in a UMA system.

- see memory as one contiguous addressable space, each processor gets its own piece of it
- a processor can access its own memory much more quickly.
- Not only does each processor have its own memory, it also has its own cache, a configuration that can lead to *cache coherence problems*

MIMD– Shared memory



- Cache coherence problems arise when main memory data is changed and the cached image is not. (We say that the cached value is *stale*.)

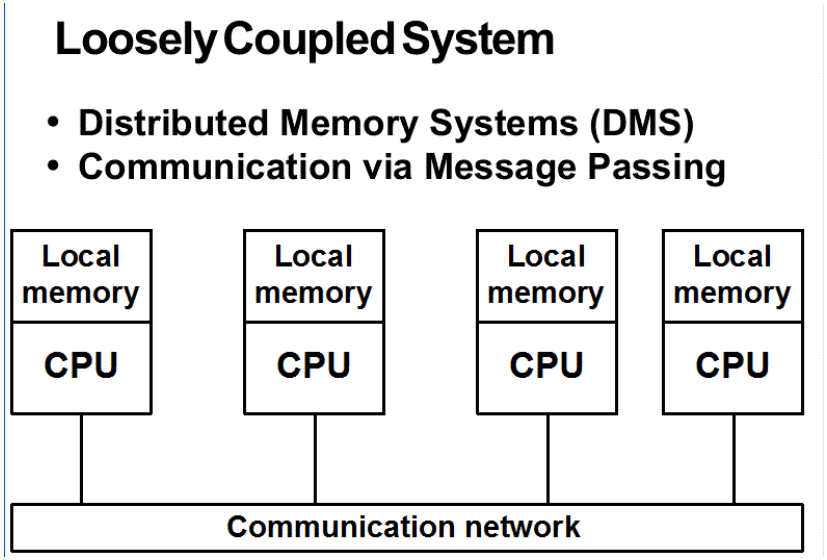
-To combat this problem,

some NUMA machines are equipped with *snoopy cache controllers* that monitor all caches on the systems. These systems are called *cache coherent NUMA* (CC-NUMA) architectures.

A simpler approach is to ask the processor having the stale value to either void the stale cached value or to update it with the new value.

Distributed Computing

- Distributed computing is another form of multiprocessing. However, the term *distributed computing* means different things to different people.
- In a sense, all multiprocessor systems are distributed systems because the processing load is distributed among processors that work collaboratively.
- **The common understanding is that a distributed system consists of very loosely-coupled processing units.**
- Recently, Networks of Workstations **(**NOWs) have been used as distributed systems to solve large, intractable problems.



Parallel in general computing

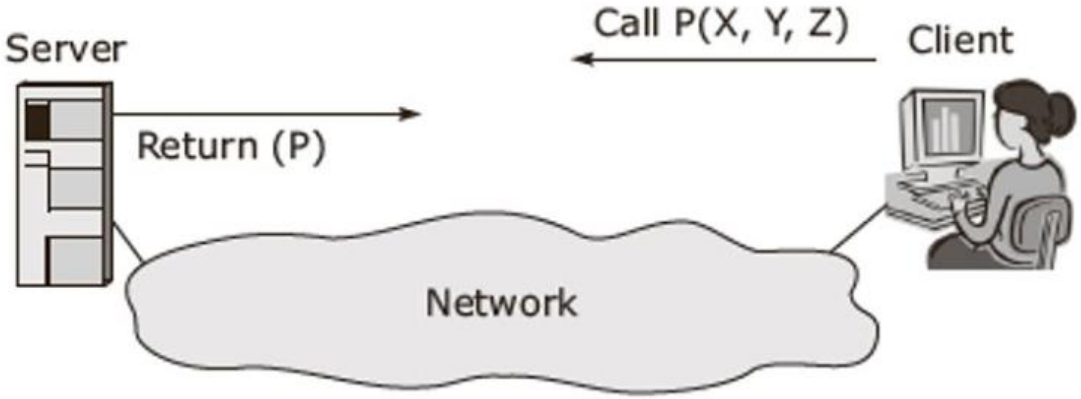


Figure 4-3 Basic RPC model

- **For general-use computing**,

  →the **details of the network and the nature of the multiplatform computing should be transparent** to the users of the system.

- **Remote procedure calls (RPCs) enable this transparency**.
  - RPCs use resources on remote machines by invoking procedures that reside and are executed on the remote machines.
- Currently, the XML-RPC, Websocket and REST (web-services) are taking place of the traditional RPCs.

9.4 Parallel and Multiprocessor Architectures

- Cloud computing is distributed computing to the extreme.
- It provides *services* over the Internet through a collection of loosely-coupled systems.
- In theory, the service consumer has no awareness of the hardware, or even its location.
  - Your services and data may even be located on the same physical system as that of your business competitor.
  - The hardware might even be located in another country.
- Security concerns are a major inhibiting factor for cloud computing.

Chapter 9 Conclusion

- The common distinctions between RISC and CISC systems include RISC's short, fixed-length instructions. RISC ISAs are load-store architectures. These things permit RISC systems to be highly pipelined.
- Flynn's Taxonomy provides a way to classify multiprocessor systems based upon the number of processors and data streams. It falls short of being an accurate depiction of today's systems.

Chapter 9 Conclusion

- Massively parallel processors have many processors, distributed memory, and computational elements communicate through a network. Symmetric multiprocessors have fewer processors and communicate through shared memory.
- Characteristics of superscalar design include superpipelining, and specialized instruction fetch and decoding units.

Chapter 9 Conclusion

- Very long instruction word (VLIW) architectures differ from superscalar architectures because the compiler, instead of a decoding unit, creates long instructions.
- Vector computers are highly-pipelined processors that operate on entire vectors or matrices at once.
- MIMD systems communicate through networks that can be blocking or nonblocking. The network topology often determines throughput.