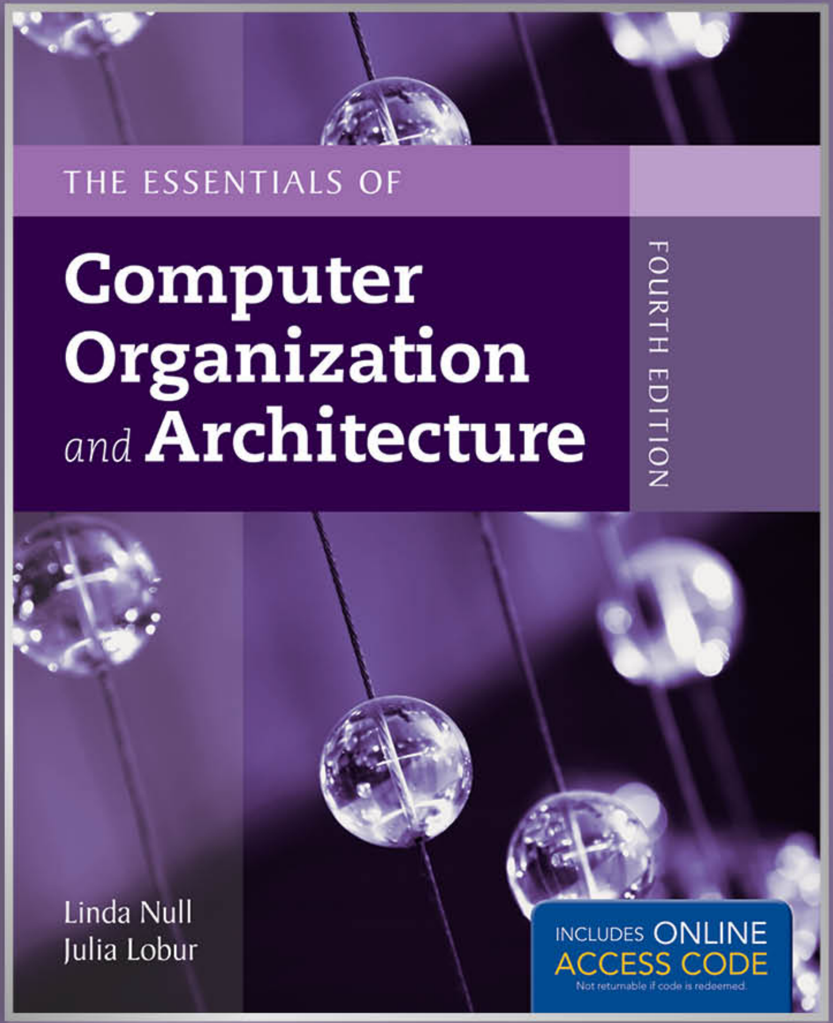
Chapter 9

Alternative Architectures


THE ESSENTIALS OF
Computer Organization and Architecture
FOURTH EDITION
Linda Null
Julia Lobur
INCLUDES ONLINE ACCESS CODE
Not returnable if code is redeemed.

Quote

“*It would appear that we have reached the limit of what is possible to achieve with computer technology, although one should be careful with such statements as they tend to sound pretty silly in 5 years*”

- John von Neumann, 1949

Chapter 9 Objectives

- Learn the properties that often distinguish RISC from CISC architectures.
- Understand how multiprocessor architectures are classified.
- Appreciate the factors that create complexity in multiprocessor systems.
- Become familiar with the ways in which some architectures transcend the traditional von Neumann paradigm.

9.1 Introduction

- We have so far studied only the simplest models of computer systems; classical single-processor von Neumann systems.
- This chapter presents a number of different approaches to computer organization and architecture.
- Some of these approaches are in place in today's commercial systems. Others may form the basis for the computers of tomorrow.

9.2 RISC Machines

- The underlying philosophy of RISC machines is that a system 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.
- RISC systems access memory only with explicit load and store instructions.
- In CISC systems, many different kinds of instructions access memory, making instruction length variable and fetch-decode-execute time unpredictable.

9.2 RISC Machines

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

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

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

9.2 RISC Machines

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

Speculative execution: The main idea is to do work before it is known whether that work will be needed at all.

9.2 RISC Machines

- 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`

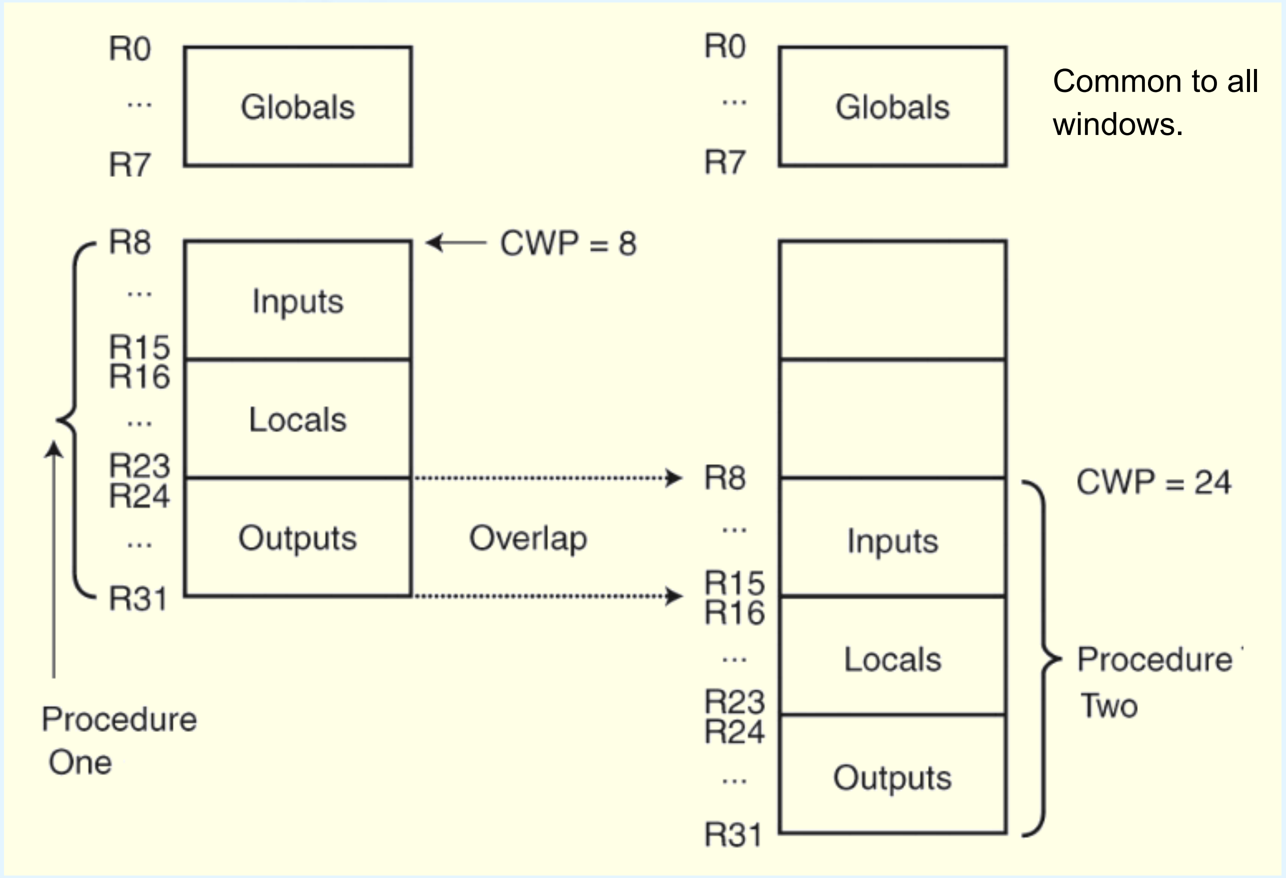
- With RISC clock cycle being shorter, RISC gives us much faster execution speeds.

9.2 RISC Machines

- 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.

9.2 RISC Machines

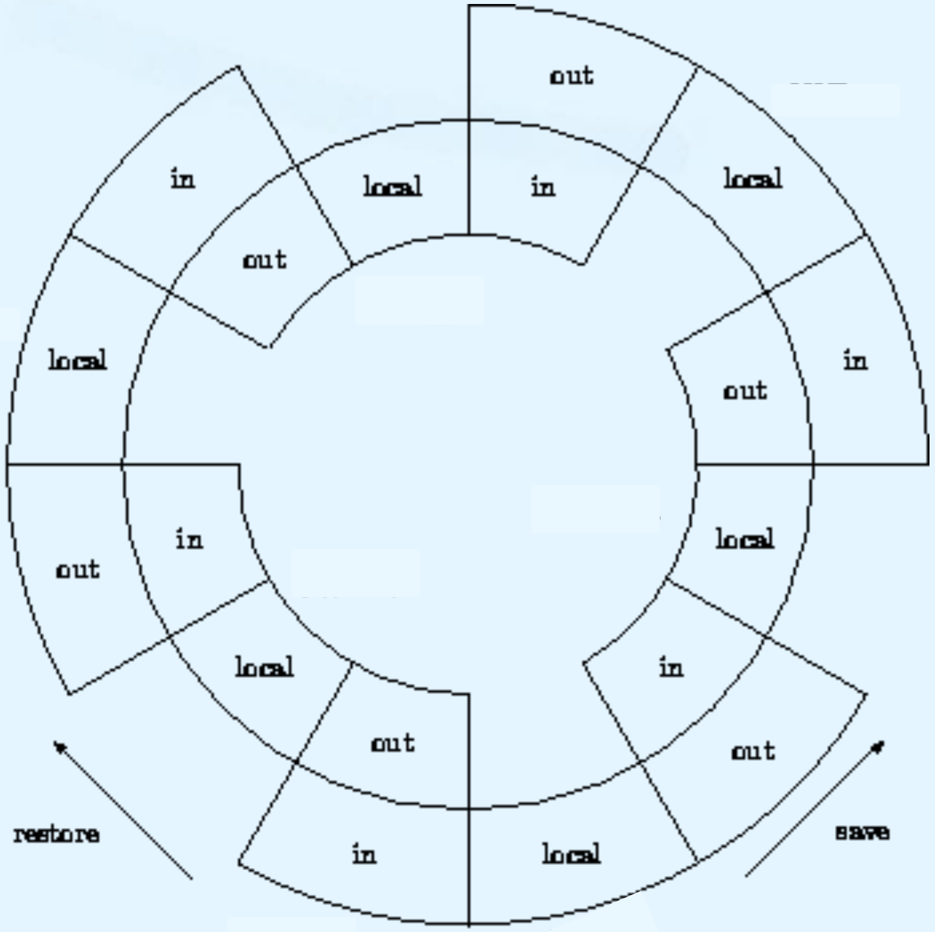
- This is how registers can be overlapped in a RISC system.
- The *current window pointer* (CWP) points to the active register window.



From the programmer's perspective, there are only 32 registers available.

9.2 RISC Machines

- The save and restore operations allocate registers in a circular fashion.
- If the supply of registers get exhausted memory takes over, storing the register windows which contain values from the oldest procedure activations.



9.2 RISC Machines

- It is becoming increasingly difficult to distinguish RISC architectures from CISC architectures.
- Some RISC systems provide more extravagant instruction sets than some CISC systems.
- Many systems combine both approaches. Many systems now employ RISC cores to implement CICS architectures.
- The following two slides summarize the characteristics that traditionally typify the differences between these two architectures.

9.2 RISC Machines

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

Continued....

9.2 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.

9.3 Flynn’s Taxonomy

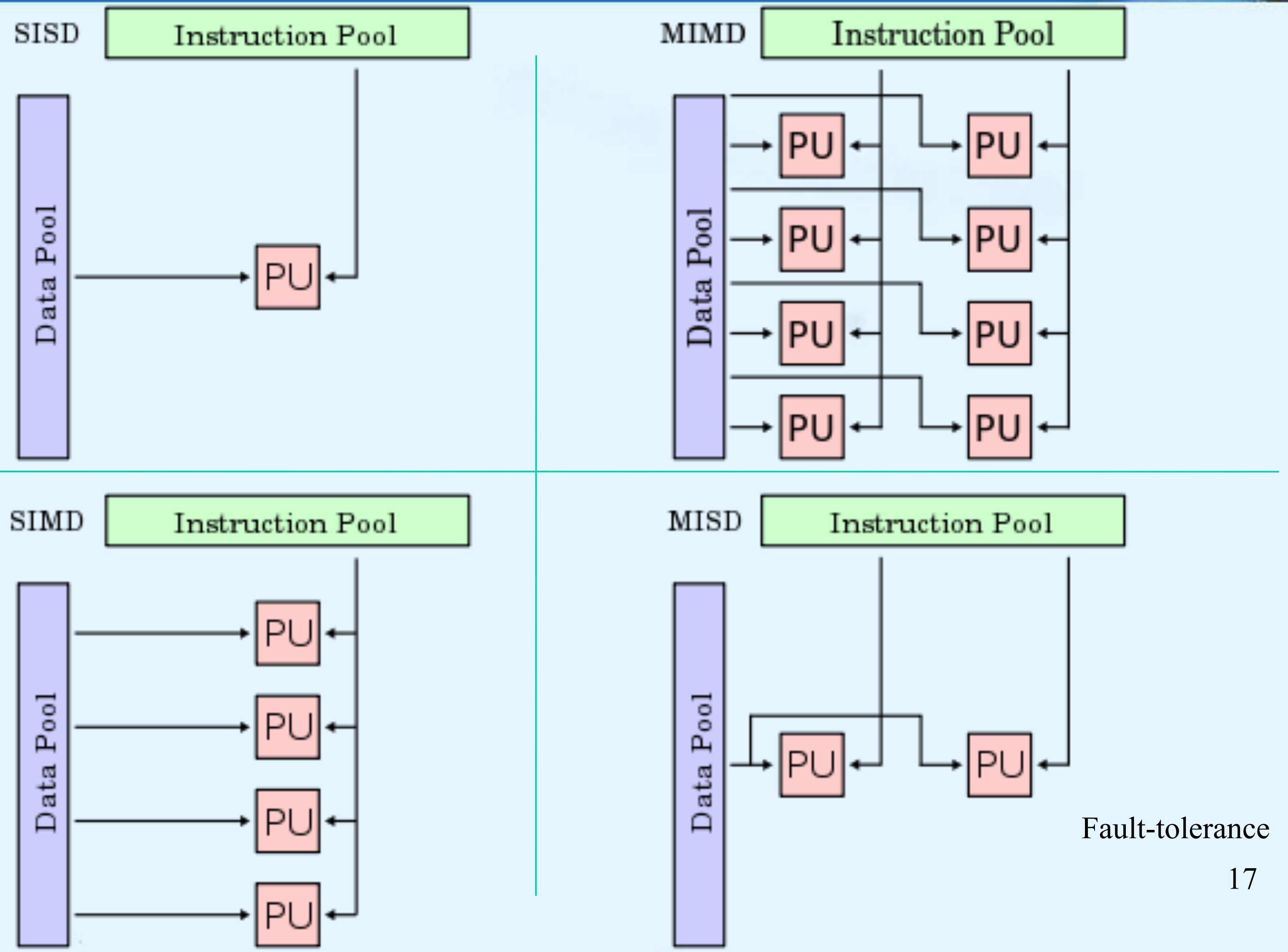
- Many attempts have been made to come up with a way to categorize computer architectures.
- *Flynn’s Taxonomy* (1972) has been the most enduring of these, despite having some limitations.
- Flynn’s Taxonomy takes into consideration the number of processors and the number of data streams incorporated into an architecture.
- A machine can have one or many processors that operate on one or many data streams.

9.3 Flynn’s Taxonomy

- The four combinations of multiple processors and multiple data streams are described by Flynn as:
  - **SISD**: **S**ingle **i**nstruction stream, **s**ingle **d**ata stream. These are classic uniprocessor systems.
  - **SIMD**: **S**ingle **i**nstruction stream, **m**ultiple **d**ata streams. Execute the same instruction on multiple data values, as in vector processors and GPUs.
  - **MIMD**: **M**ultiple **i**nstruction streams, **m**ultiple **d**ata streams. These are today’s parallel architectures.
  - **MISD**: **M**ultiple **i**nstruction streams, **s**ingle **d**ata stream.

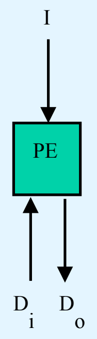
As of 2006, all the top 10 and most of the TOP500 supercomputers are based on a MIMD architecture (Wikipedia).

9.3 Flynn's Taxonomy

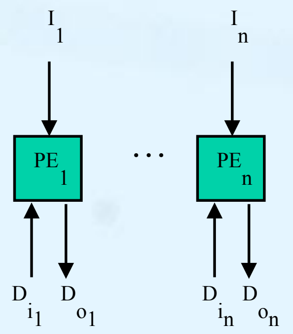


9.3 Flynn's Taxonomy

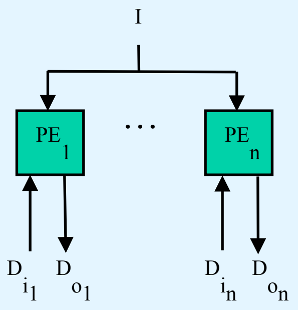
SISD (Single-Instruction Single-Data)



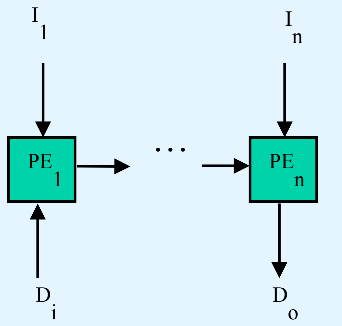
MIMD (Multiple-Instruction Multiple-Data)



SIMD (Single-Instruction Multiple-Data)



MISD (Multiple-Instruction Single-Data)



Pipeline

PE: Processing element
I: Instruction
D: Data

9.3 Flynn’s Taxonomy

- Flynn’s Taxonomy falls short in a number of ways:
- First, there appears to be very few (if any) applications for MISD machines.
- Second, parallelism is not homogeneous. This assumption ignores the contribution of specialized processors.
- 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.

9.3 Flynn’s Taxonomy

- 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)

9.3 Flynn's Taxonomy

- 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.

NOWs, COWs, DCPCs, and POPCs are all examples of **cluster computing**.

9.3 Flynn’s Taxonomy

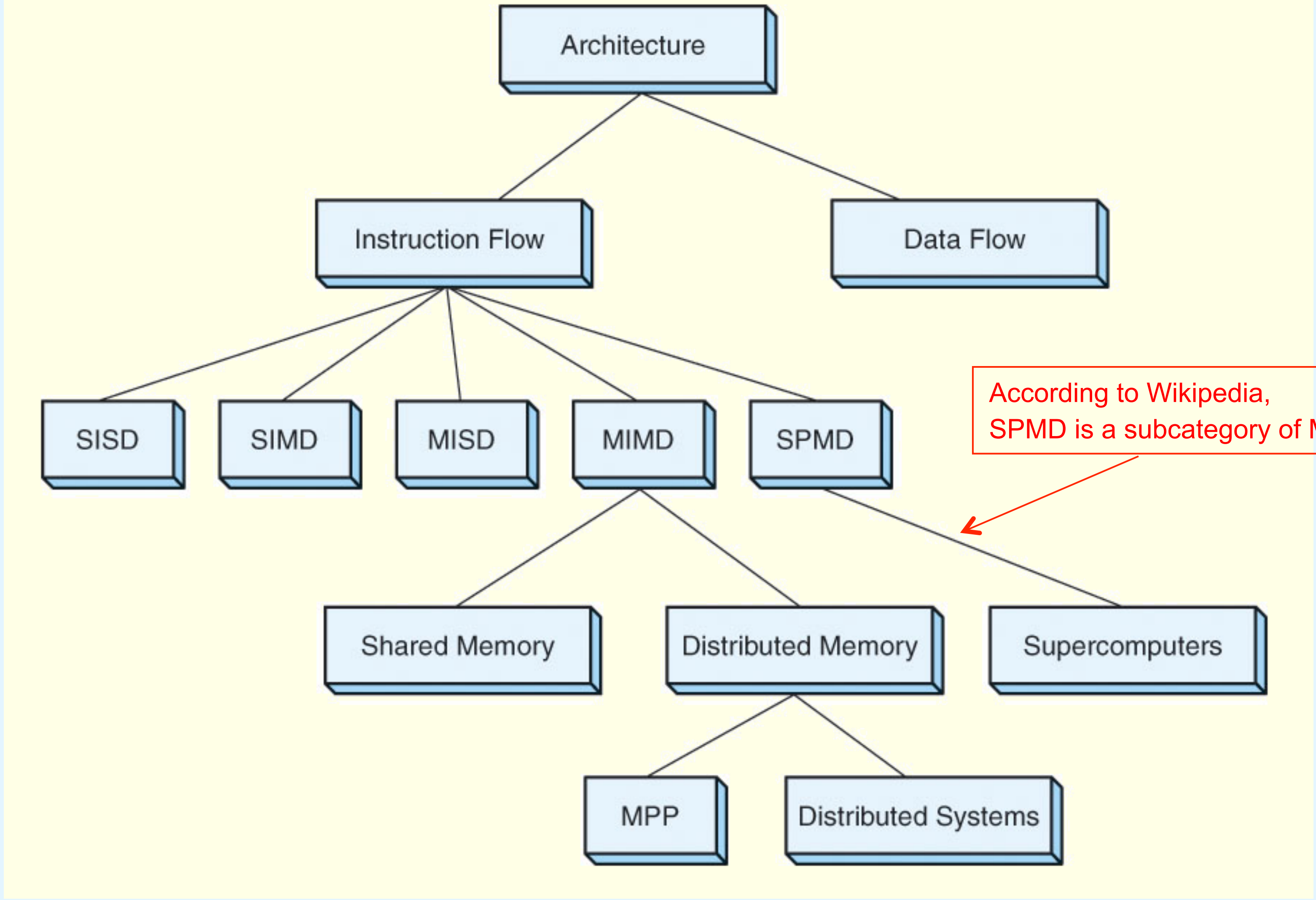
- Flynn’s Taxonomy has been expanded to include *SPMD* (**s**ingle **p**rogram, **m**ultiple **d**ata) architectures.
- Each SPMD processor has its own data set and program memory. Different nodes can execute different instructions within the same program using instructions similar to:

  If myNodeNum = 1 do this, else do that

- Yet another idea missing from Flynn’s is whether the architecture is instruction driven or data driven.

The next slide provides a revised taxonomy.

9.3 Flynn's Taxonomy



According to Wikipedia,
SPMD is a subcategory of MIMD

9.4 Parallel and Multiprocessor Architectures

- If we are using an ox to pull out a tree, and the tree is too large, we don't try to grow a bigger ox.
- In stead, we use two oxen.

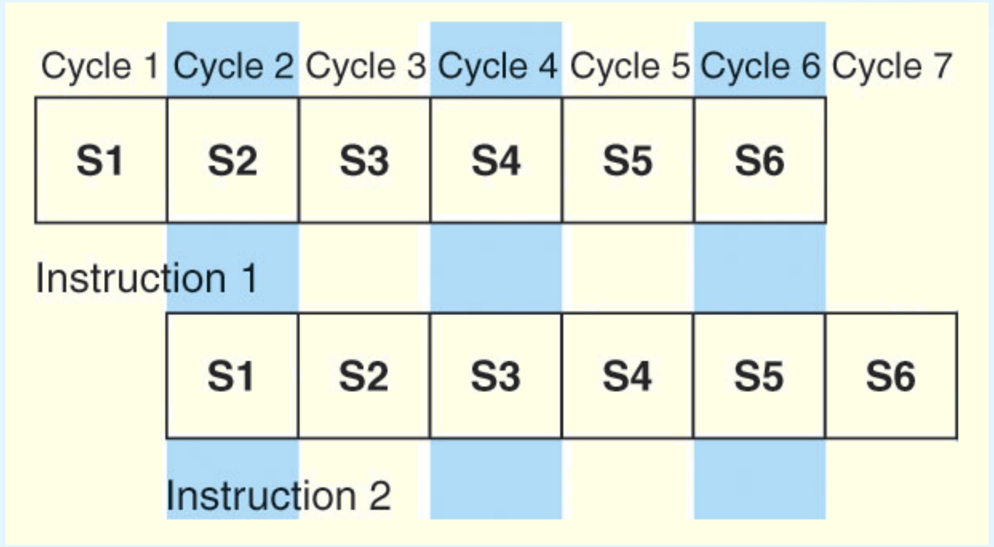
- Multiprocessor architectures are analogous to the oxen.

9.4 Parallel and Multiprocessor Architectures

- Parallel processing is capable of economically increasing system throughput.
- The limiting factor is that no matter how well an algorithm is parallelized, there is always some portion that must be done sequentially.
  - Additional processors sit idle while the sequential work is performed.
- Thus, it is important to keep in mind that an *n*-fold increase in processing power does not necessarily result in an *n*-fold increase in throughput.

5.5 Instruction-Level Pipelining

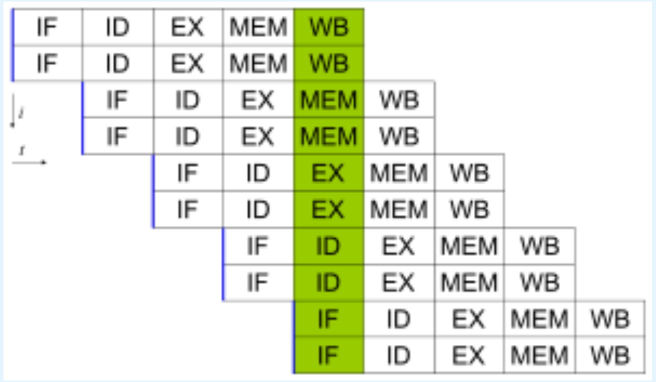
- For every clock cycle, one small step is carried out, and the stages are overlapped.



S1. Fetch instruction.
S2. Decode opcode.
S3. Calculate effective address of operands.
S4. Fetch operands.
S5. Execute.
S6. Store result.

9.4 Parallel and Multiprocessor Architectures

- Ideally, an instruction exits the pipeline during each tick of the clock.
- *Superpipelining* occurs when a pipeline has stages that require less than half a clock cycle to complete.
  - The pipeline is equipped with a separate clock running at a frequency that is at least double that of the main system clock.
- Superpipelining is only one aspect of superscalar design (envisioned having multiple parallel pipelines).



IF	ID	EX	MEM	WB				
IF	ID	EX	MEM	WB				
	IF	ID	EX	MEM	WB			
	IF	ID	EX	MEM	WB			
		IF	ID	EX	MEM	WB		
		IF	ID	EX	MEM	WB		
			IF	ID	EX	MEM	WB	
			IF	ID	EX	MEM	WB	
				IF	ID	EX	MEM	WB
				IF	ID	EX	MEM	WB

9.4 Parallel and Multiprocessor Architectures

- *Superscalar* architectures include multiple execution units such as specialized integer and floating-point adders and multipliers.
- A critical component of this architecture is the *instruction fetch unit*, which can simultaneously retrieve several instructions from memory.
- A *decoding unit* determines which of these instructions can be executed in parallel and combines them accordingly.
- This architecture also requires compilers that make optimum use of the hardware.

9.4 Parallel and Multiprocessor Architectures

- *Very long instruction word* (VLIW) architectures differ from superscalar architectures because the VLIW compiler, instead of a hardware decoding unit, packs *independent* instructions (typically 4-8 instructions) into one long instruction that is sent down the pipeline to the execution units.
- One could argue that this is the best approach because the compiler can better identify instruction dependencies.
- However, compilers tend to be conservative and cannot have a view of the run time code.

9.4 Parallel and Multiprocessor Architectures

- 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. The results are streamed back to memory.

9.4 Parallel and Multiprocessor Architectures

- A disadvantage of register-register vector computers is that 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 until the pipeline becomes full.
- In general, vector machines are efficient because there are fewer instructions to fetch, and corresponding pairs of values can be prefetched because the processor knows it will have a continuous stream of data.

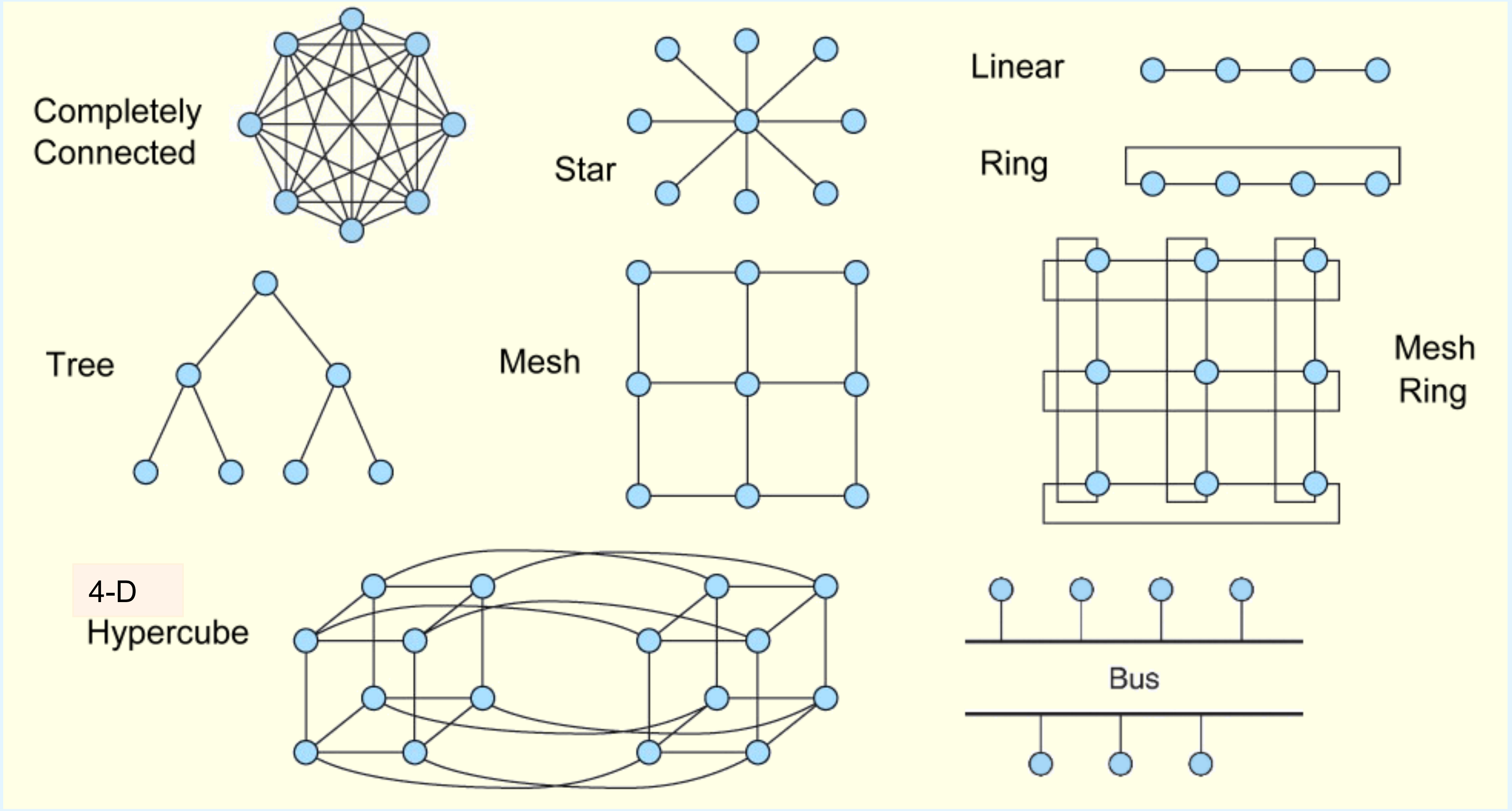
9.4 Parallel and Multiprocessor Architectures

- MIMD systems can communicate through shared memory or through an interconnection network.
- Interconnection networks are often classified according to their topology, routing strategy, and switching technique.
- Of these, the topology (the way in which the components are interconnected) 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.

9.4 Parallel and Multiprocessor Architectures

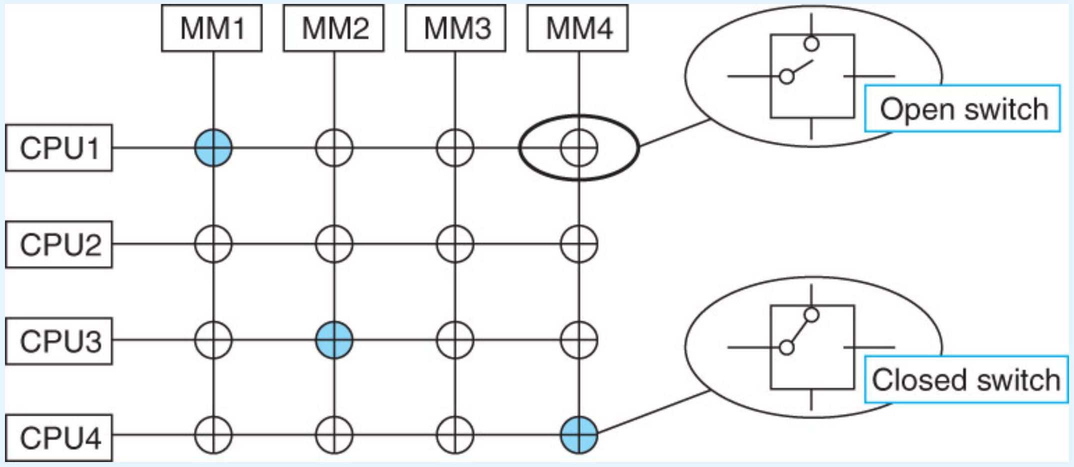
- Interconnection networks can be either static or dynamic.
- Processor-to-memory connections usually employ dynamic interconnections. These can be blocking or nonblocking.
  - Nonblocking interconnections allow connections to occur simultaneously.
- Processor-to-processor message-passing connections are usually static, and can employ any of several different topologies, as shown on the following slide.

9.4 Parallel and Multiprocessor Architectures

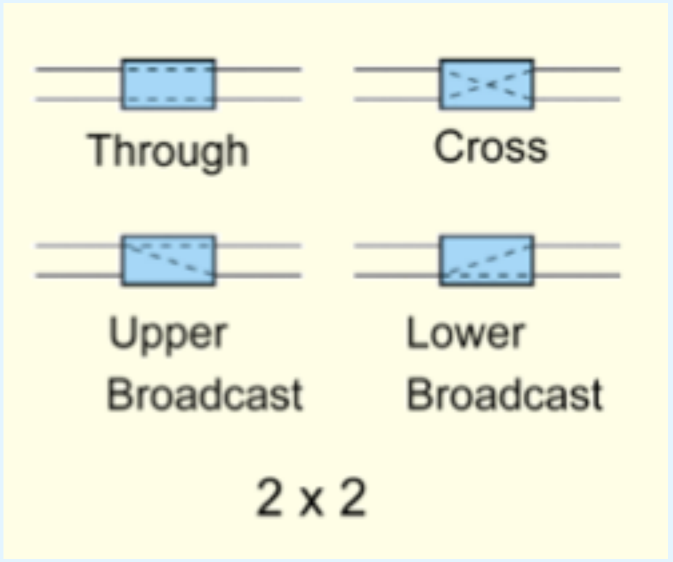


9.4 Parallel and Multiprocessor Architectures

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



Crossbar network

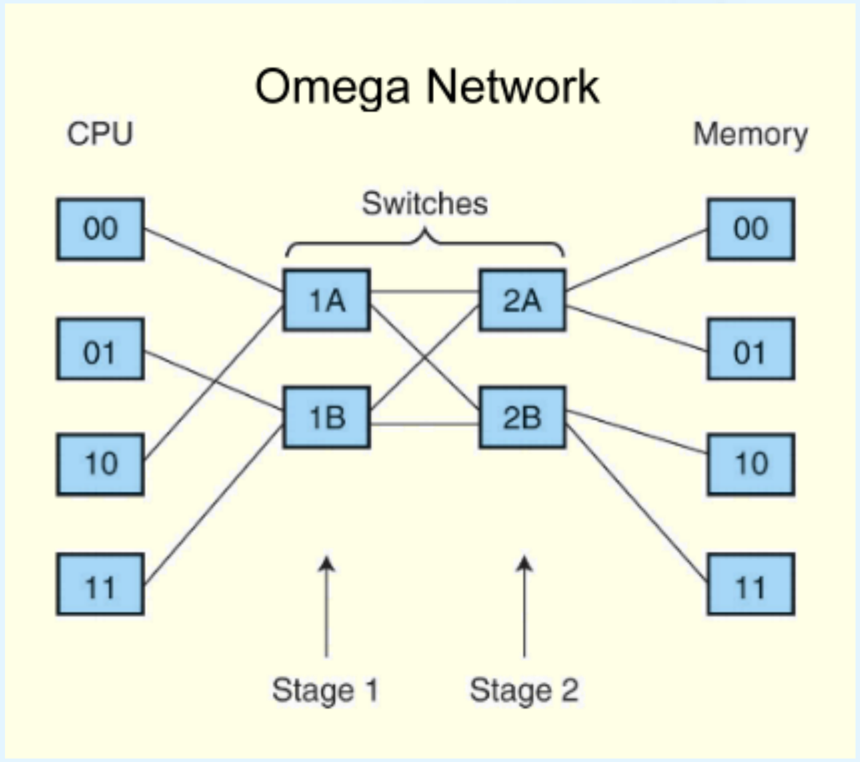


- The `through` and `cross` states are the ones relevant to interconnection networks.

9.4 Parallel and Multiprocessor Architectures

- 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.



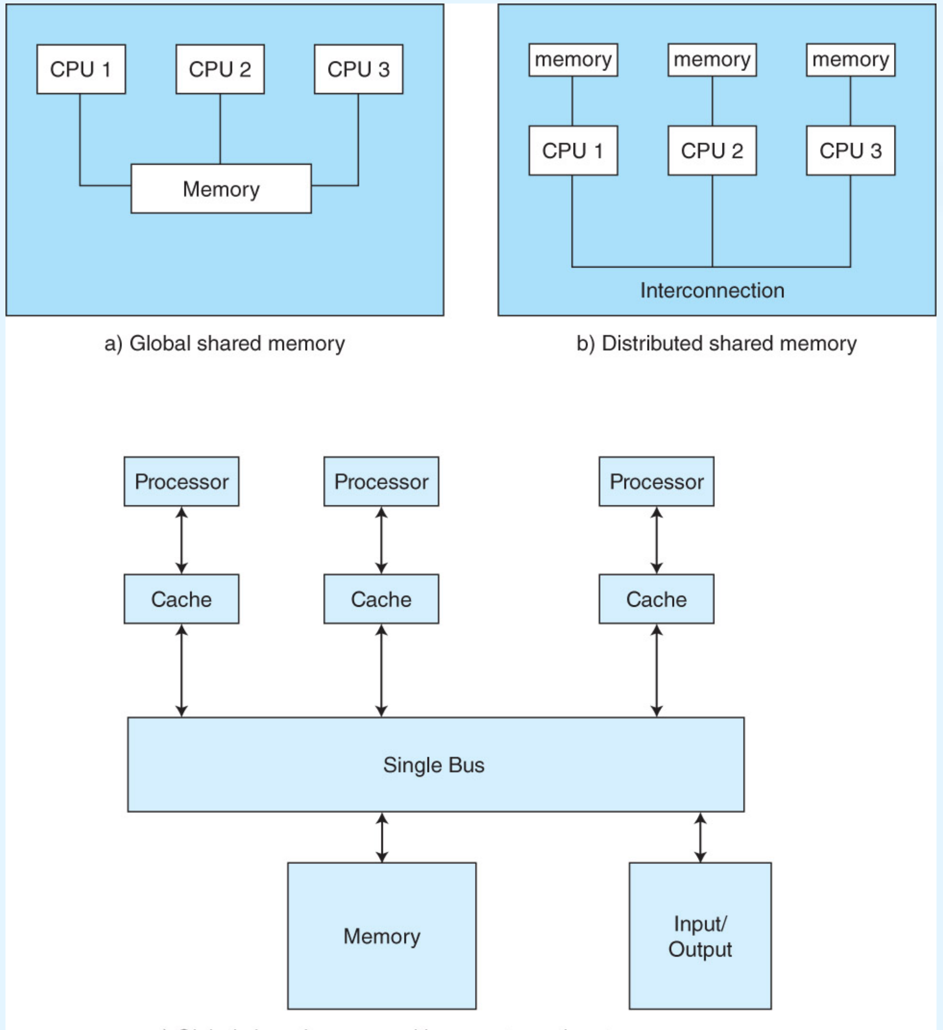
9.4 Parallel and Multiprocessor Architectures

- There are advantages and disadvantages to each switching approach.
  - Bus-based networks, while economical, can be bottlenecks. Parallel buses can alleviate bottlenecks, but are costly.
  - Crossbar networks are nonblocking, but require n^2 switches to connect n entities.
  - Omega networks are blocking networks, but exhibit less contention than bus-based networks. They are somewhat more economical than crossbar networks, n nodes needing $\log_2 n$ stages with $n / 2$ switches per stage.

9.4 Parallel and Multiprocessor Architectures

- 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*.

9.4 Parallel and Multiprocessor Architectures



a) Global shared memory

b) Distributed shared memory

c) Global shared memory with separate cache at processors

9.4 Parallel and Multiprocessor Architectures

- Shared memory MIMD machines can be divided into two categories based upon how they access memory: UMA and NUMA.
- In *uniform memory access* (UMA) systems, all memory accesses take the same amount of time.
- To realize the advantages of a multiprocessor system, 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.

9.4 Parallel and Multiprocessor Architectures

- The other category of MIMD machines are the *nonuniform memory access* (NUMA) systems.
- While NUMA machines see memory as one contiguous addressable space, each processor gets its own piece of it.
- Thus, a processor can access its own memory much more quickly than it can access memory that is elsewhere.
- Not only does each processor have its own memory, it also has its own cache, a configuration that can lead to *cache coherence problems*.

9.4 Parallel and Multiprocessor Architectures

- 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.

9.4 Parallel and Multiprocessor Architectures

- When a processor´s cached value is updated concurrently with the update to memory, we say that the system uses a *write-through* cache update protocol.
- If the *write-through with update protocol* is used, a message containing the update is broadcast to all processors so that they may update their caches.
- If the *write-through with invalidate protocol* is used, a broadcast asks all processors to invalidate the stale cached value.

9.4 Parallel and Multiprocessor Architectures

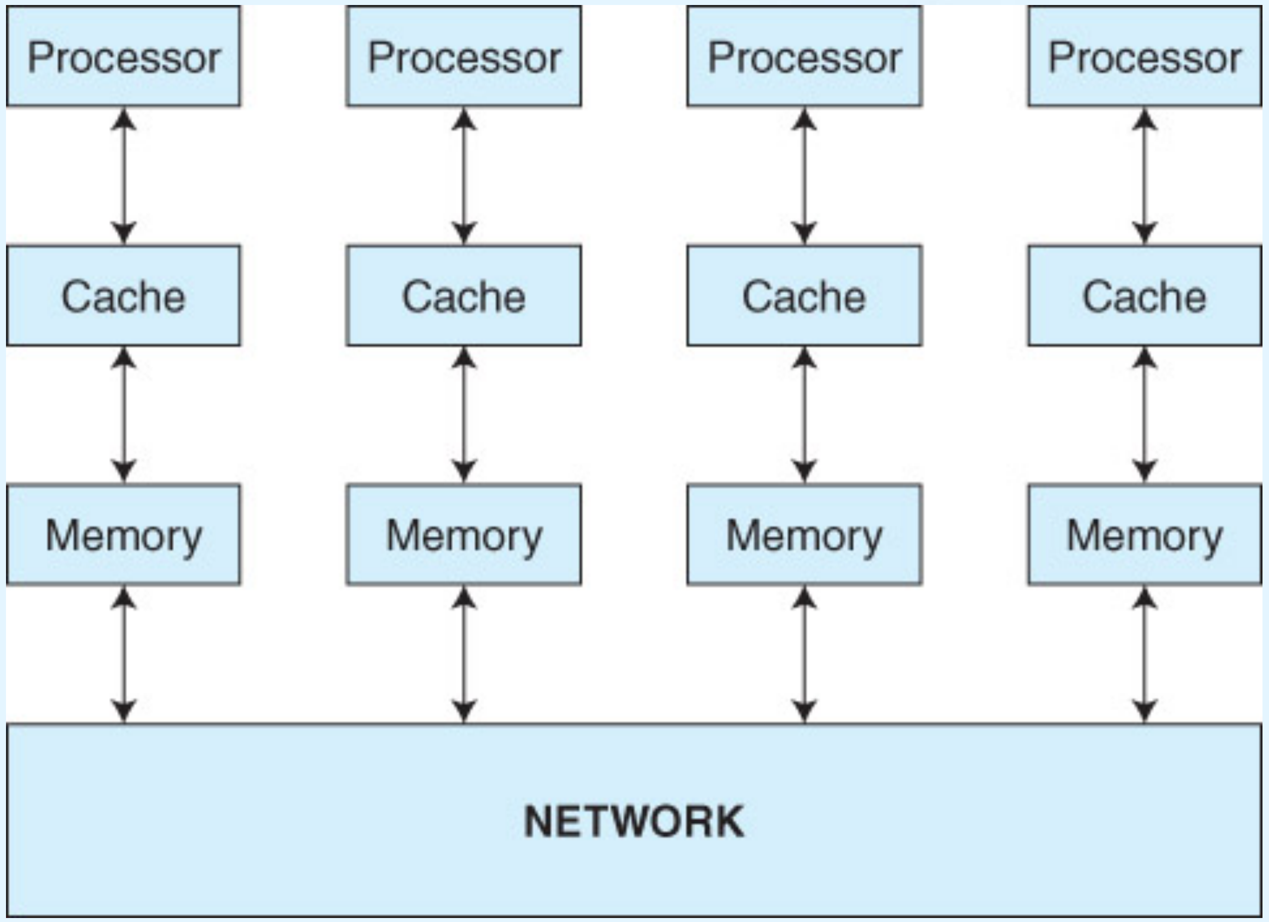
- Write-invalidate uses less bandwidth because it uses the network only the first time the data is updated, but retrieval of the fresh data takes longer.
- Write-update creates more message traffic, but all caches are kept current.
- Another approach is the *write-back* protocol that delays an update to memory until the modified cache block must be replaced.
- At replacement time, the processor writing the cached value must obtain exclusive rights to the data. When rights are granted, all other cached copies are invalidated.

9.4 Parallel and Multiprocessor Architectures

- 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, NOWs have been used as distributed systems to solve large, intractable problems.

9.4 Parallel and Multiprocessor Architectures

- Multiprocessors connected by a network



9.4 Parallel and Multiprocessor Architectures

- 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.
- RPCs are employed by numerous vendors of distributed computing architectures including the Common Object Request Broker Architecture (CORBA) and Java's Remote Method Invocation (RMI).

Computer programs and procedures that are said to be **transparent** are typically those that the user is - or could be - unaware of.

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.

9.5 Alternative Parallel Processing Approaches

- Some people argue that real breakthroughs in computational power -- breakthroughs that will enable us to solve today’s intractable problems -- will occur only by abandoning the von Neumann model.
- Numerous efforts are now underway to devise systems that could change the way that we think about computers and computation.
- In this section, we will look at three of these: dataflow computing, neural networks, and systolic processing.

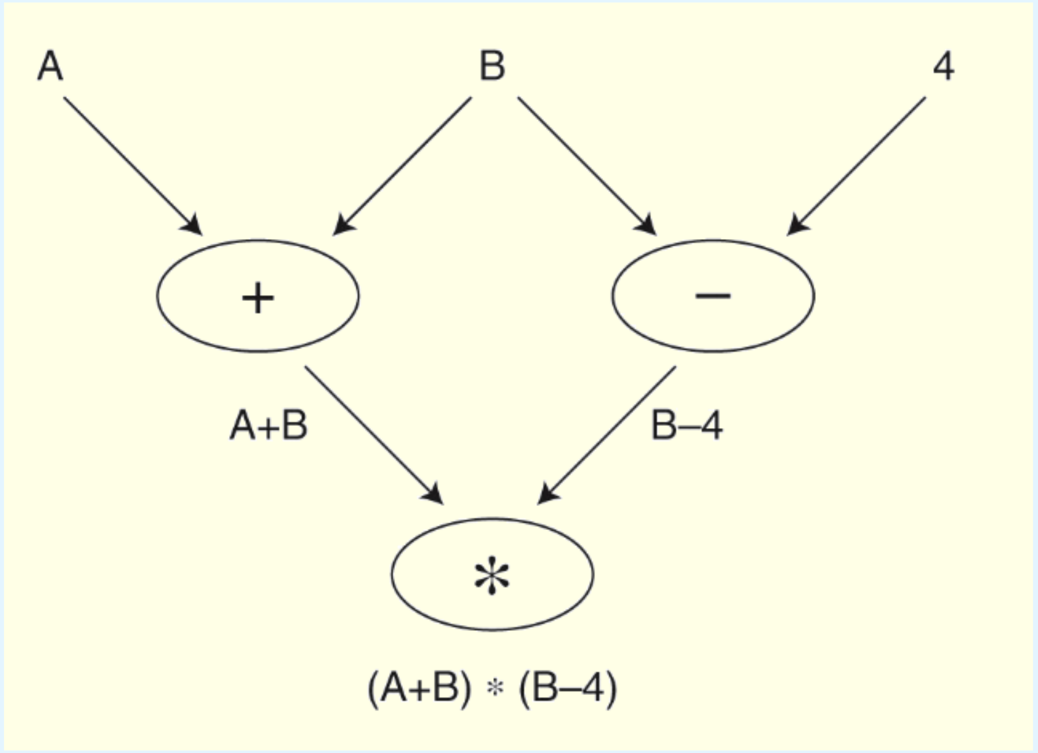
9.5 Alternative Parallel Processing Approaches

- Von Neumann machines exhibit sequential control flow: A linear stream of instructions is fetched from memory, and they act upon data.
- Program flow changes under the direction of branching instructions.
- In *dataflow* computing, program control is directly controlled by data dependencies.
- There is no program counter or shared storage.
- Data flows continuously and is available to multiple instructions simultaneously.

9.5 Alternative Parallel Processing Approaches

- A *data flow graph* represents the computation flow in a dataflow computer.

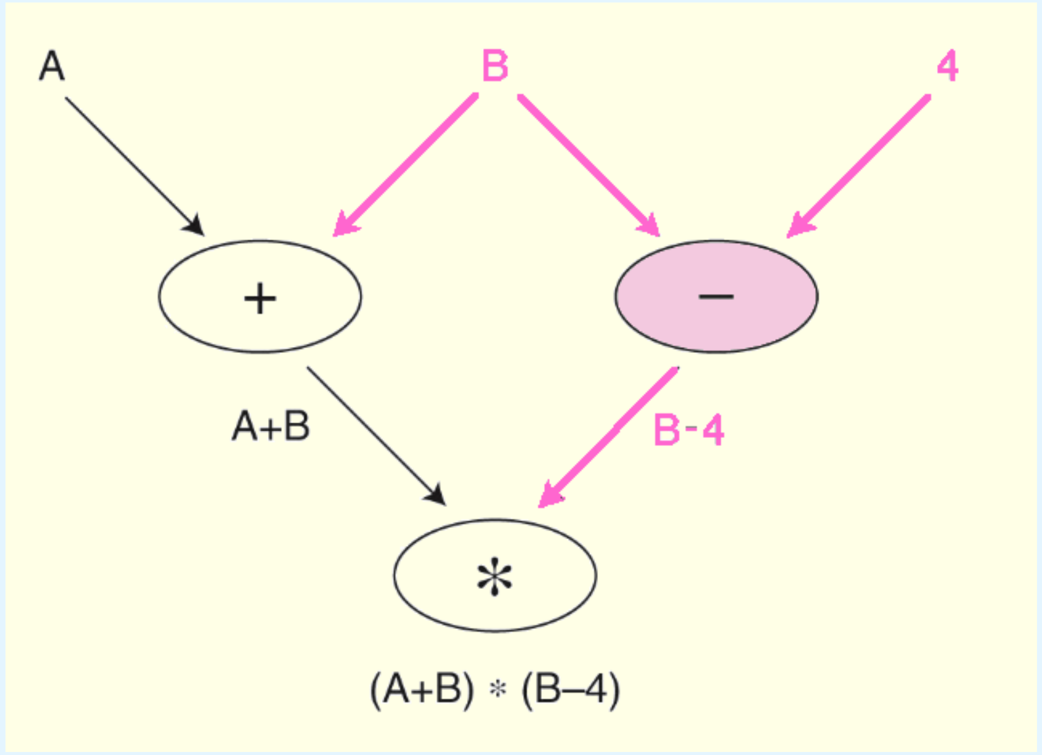
Its nodes contain the instructions and its arcs indicate the data dependencies.



9.5 Alternative Parallel Processing Approaches

- When a node has all of the data tokens it needs, it fires, performing the required operation, and consuming the token.

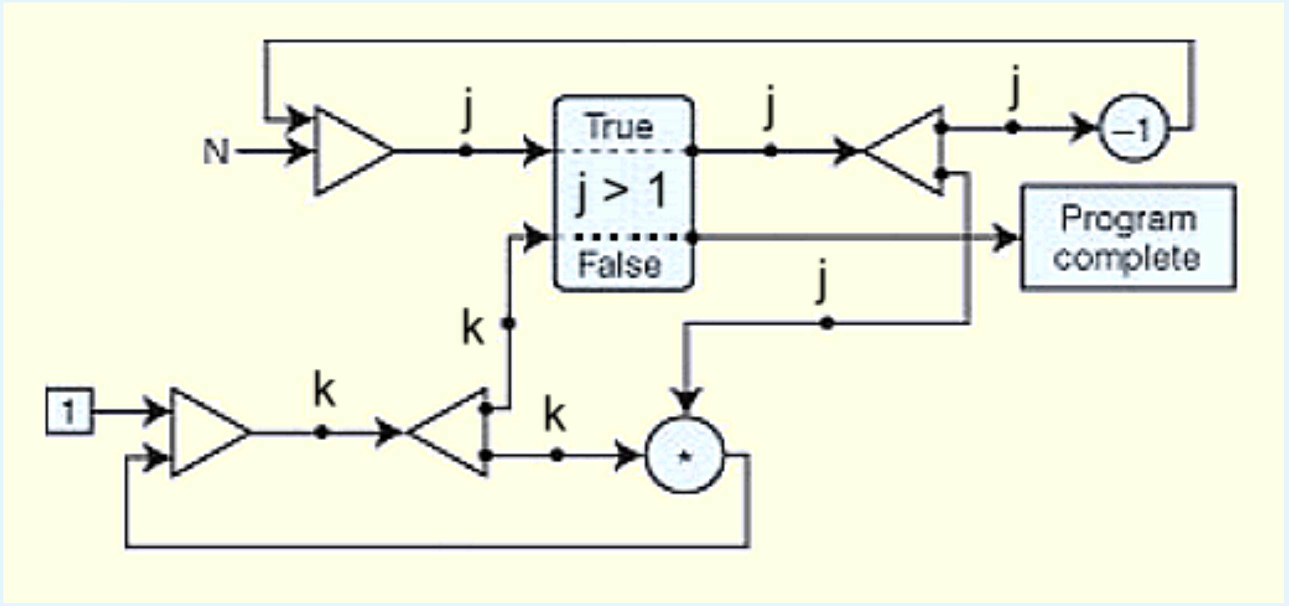
The result is placed on an output arc.



9.5 Alternative Parallel Processing Approaches

- A dataflow program to calculate *N*! and its corresponding graph are shown below.

```
(initial j <- N; k <- 1
  while j > 1 do
    new k <- k * j;
    new j <- j - 1;
 return k)
```



9.5 Alternative Parallel Processing Approaches

- The architecture of a dataflow computer consists of processing elements that communicate with one another.
- Each processing element has an *enabling unit* that sequentially accepts tokens and stores them in memory.
- If the node to which this token is addressed fires, the input tokens are extracted from memory and are combined with the node itself to form an executable packet.

9.5 Alternative Parallel Processing Approaches

- Using the executable packet, the processing element's *functional unit* computes any output values and combines them with destination addresses to form more tokens.
- The tokens are then sent back to the enabling unit, optionally enabling other nodes.
- Because dataflow machines are data driven, multiprocessor dataflow architectures are not subject to the cache coherency and contention problems that plague other multiprocessor systems.

9.5 Alternative Parallel Processing Approaches

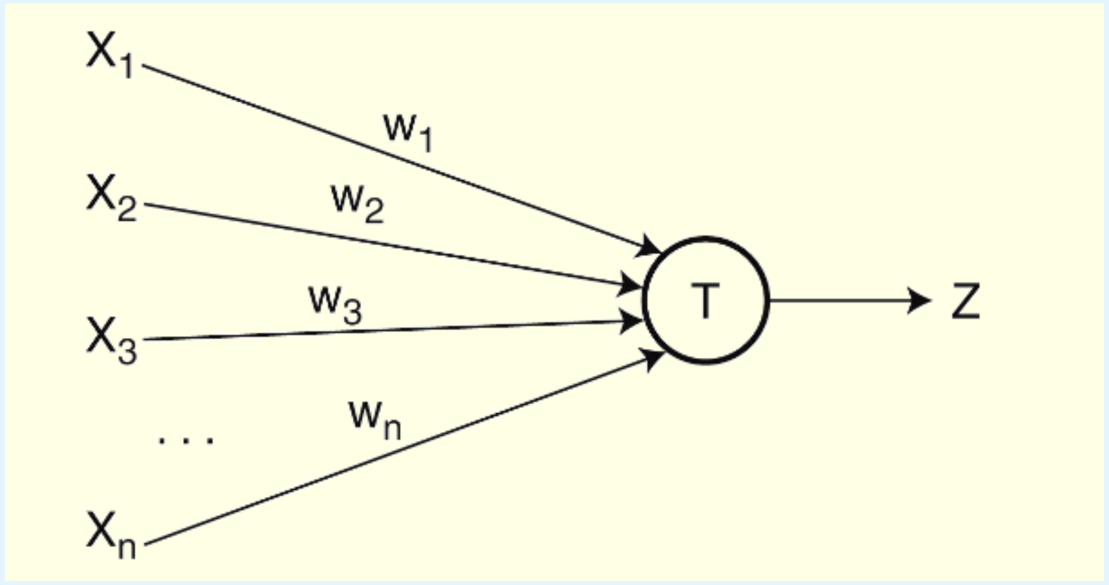
- *Neural network* computers, which are data-driven, consist of a large number of simple processing elements that individually solve a small piece of a much larger problem.
- They are particularly useful in dynamic situations that are an accumulation of previous behavior, and where an exact algorithmic solution cannot be formulated.
- Like their biological analogues, neural networks can deal with imprecise, probabilistic information, and allow for adaptive interactions.

9.5 Alternative Parallel Processing Approaches

- Neural network processing elements (PEs) multiply a set of input values by an adaptable set of weights to yield a single output value.
- The computation carried out by each PE is simplistic -- almost trivial -- when compared to a traditional microprocessor. Their power lies in their massively parallel architecture and their ability to adapt to the dynamics of the problem space.
- Neural networks learn from their environments. A built-in *learning algorithm* directs this process.

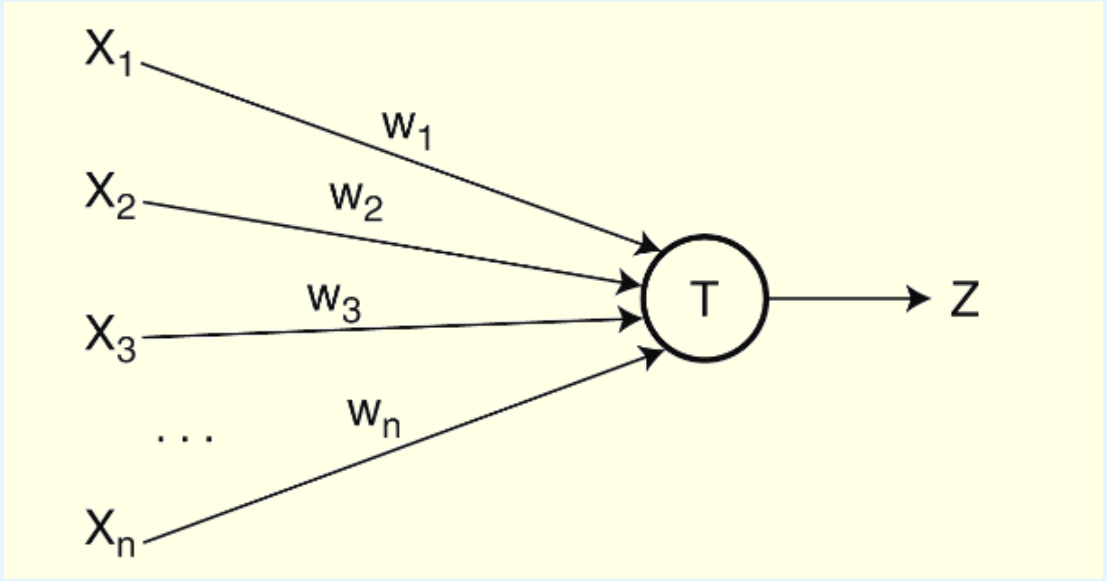
9.5 Alternative Parallel Processing Approaches

- The simplest neural net PE is the *perceptron*.
- Perceptrons are trainable neurons. A perceptron produces a Boolean output based upon the values that it receives from several inputs.



9.5 Alternative Parallel Processing Approaches

- Perceptrons are trainable because the threshold and input weights are modifiable.
- In this example, the output *Z* is true (1) if the net input, $w_1x_1 + w_2x_2 + \ldots + w_nx_n$ is greater than the threshold *T*. Otherwise, *Z* is false (0).

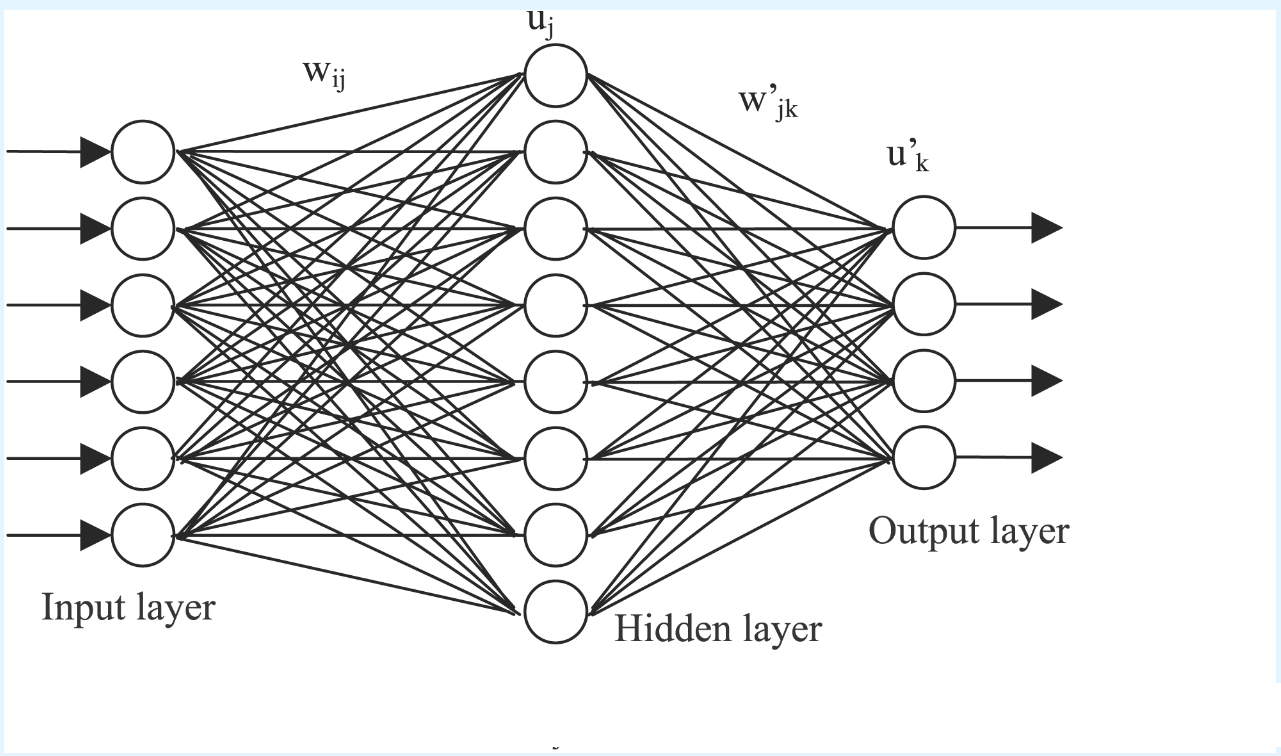


9.5 Alternative Parallel Processing Approaches

- Perceptrons are trained by use of supervised or unsupervised learning.
- Supervised learning assumes prior knowledge of correct results which are fed to the neural net during the training phase. If the output is incorrect, the network modifies the input weights to produce correct results.
- Unsupervised learning (data mining) does not provide correct results during training. The network adapts solely in response to inputs, learning to recognize patterns and structure in the input sets.

9.5 Alternative Parallel Processing Approaches

- A three layer neural network.
- This type of network may be trained with the *backpropagation* algorithm.

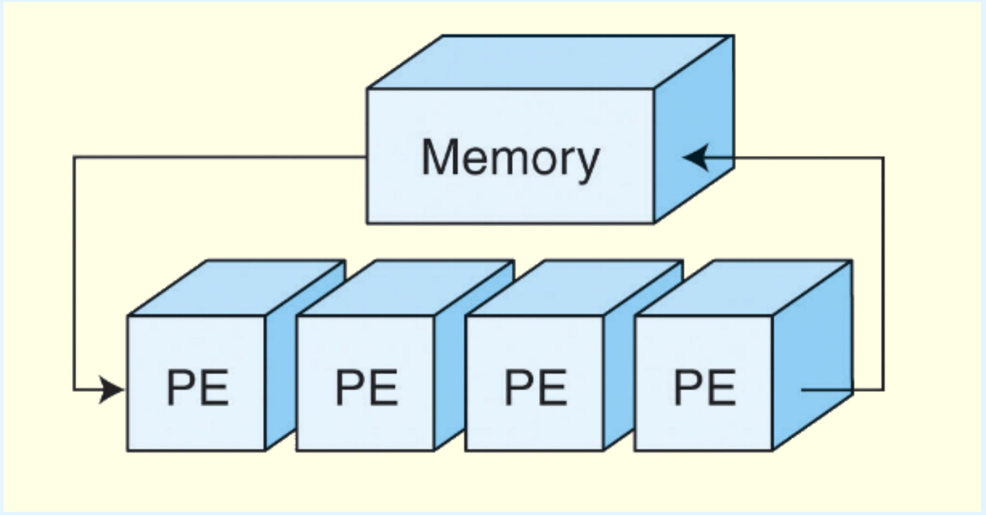


9.5 Alternative Parallel Processing Approaches

- The biggest problem with neural nets is that when they consist of more than 10 or 20 neurons, it is impossible to understand how the net is arriving at its results. They can derive meaning from data that are too complex to be analyzed by people.
  - The U.S. military once used a neural net to try to locate camouflaged tanks in a series of photographs. It turned out that the nets were basing their decisions on the cloud cover instead of the presence or absence of the tanks.
- Despite early setbacks, neural nets are gaining credibility in sales forecasting, data validation, and facial recognition.

9.5 Alternative Parallel Processing Approaches

- Where neural nets are a model of biological neurons, *systolic array* computers are a model of how blood flows through a biological heart.
- Systolic arrays, a variation of SIMD computers, have simple processors that process data by circulating it through vector pipelines.



9.5 Alternative Parallel Processing Approaches

- Systolic arrays can sustain great throughput because they employ a high degree of parallelism.
- Connections are short, and the design is simple and scalable. They are robust, efficient, and cheap to produce. They are, however, highly specialized and limited as to they types of problems they can solve.
- They are useful for solving repetitive problems that lend themselves to parallel solutions using a large number of simple processing elements.
  - Examples include sorting, image processing, and Fourier transformations.

9.6 Quantum Computing

- Computers, as we know them are binary, transistor-based systems.
- But transistor-based systems strain to keep up with our computational demands.
- We increase the number of transistors for more power, and make each transistor smaller to fit on the die.
  - Transistors are becoming so small that it is hard for them to hold electrons in the way in which we're accustomed to.
- Thus, alternatives to transistor-based systems are an active area or research.

9.6 Quantum Computing

- Computers are now being built based on:
  - Optics (photonic computing using laser light)
  - Biological neurons
  - DNA
- One of the most intriguing is quantum computers.
- Quantum computing uses quantum bits (qubits) that can be in multiple states at once.
- The "state" of a qubit is determined by the spin of an electron.

A thorough discussion of "spin" is under the domain of quantum physics.

9.6 Quantum Computing

- A qubit can be in multiple states at the same time.
  - This is called *superpositioning*.
- A 3-bit register can simultaneously hold the values 0 through 7
  - 8 operations can be performed at the same time.
- This phenomenon is called *quantum parallelism*.
  - A system with 600 qbits can superposition 2^{600} states

9.6 Quantum Computing

- D-Wave Computers is the first quantum computer manufacturer
- D-Wave computers having 512 qbits were purchased separately by University of Southern California and Google for research purposes.
- Quantum computers may be applied in the areas of cryptography, true random-number generation, and in the solution of other intractable problems.

9.6 Quantum Computing

- Making effective use of quantum computers requires rethinking our approach to problems and the development of new algorithms.
  - To break a cypher, the quantum machine simulates every possible state of the problem set (i.e., every possible key for a cipher) and it “collapses” on the correct solution.
- Examples include Schor’s algorithm for factoring products of prime numbers.
- Many others remain to be discovered.

9.6 Quantum Computing

- These systems are not constrained by a fetch-decode-execute cycle; however, quantum architectures have yet to settle on a definitive paradigm analogous to von Neumann systems.
- Rose's Law states that the number of qubits that can be assembled to successfully perform computations will double every 12 months; this has been precisely the case for the past nine years
  - This "law" is named after Geordie Rose, D-Wave's founder and chief technology officer.

9.6 Quantum Computing

- One of the largest obstacles is the tendency for qubits to decay into a state of *decoherence.*
- Decoherence causes uncorrectable errors.
- Although most scientists concur as to their potential, quantum computers have thus far been able to solve only trivial problems.
- Much research remains to be done.

9.6 Quantum Computing

- The realization of quantum computing has raised questions about *technological singularity*.
  - Technological singularity is the theoretical point when human technology has fundamentally and irreversibly altered human development.
  - This is the point when civilization changes to an extent that its technology is incomprehensible to previous generations.
- Are we there, now?

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.

Chapter 9 Conclusion

- Multiprocessor memory can be distributed or exist in a single unit. Distributed memory brings to rise problems with cache coherency that are addressed using cache coherency protocols.
- New architectures are being devised to solve intractable problems. These new architectures include dataflow computers, neural networks, and systolic arrays.

End of Chapter 9