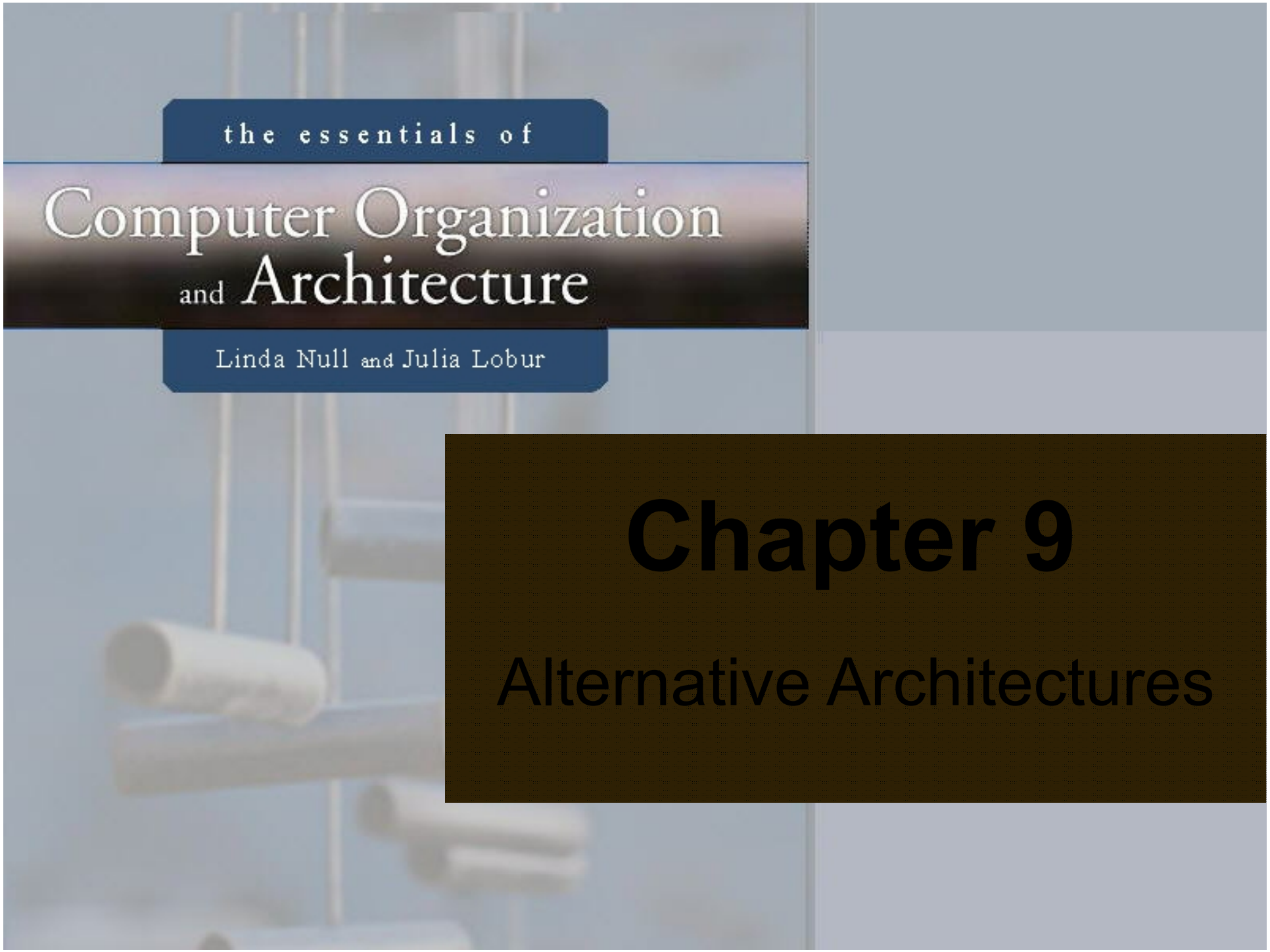


Chapter 9

Alternative Architectures

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:

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

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

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.

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
        mul 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`

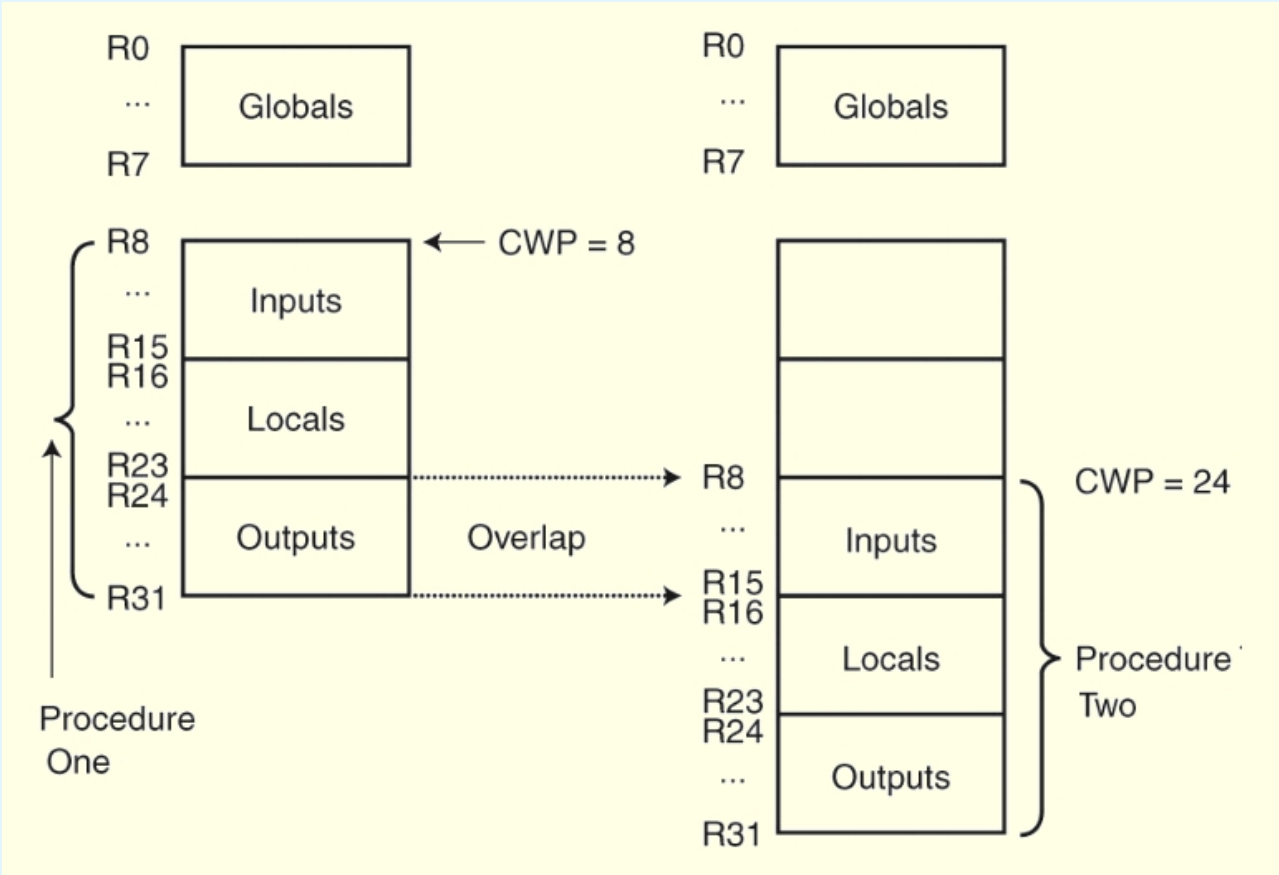
- With RISC clock cycle being shorter, RISC gives us much faster execution speeds.

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.



RISC	CISC
Multiple register sets, often consisting of more than 256 registers	Single register set, typically 6 to 16 registers total
Three register operands allowed per instruction (e.g., `add R1, R2, R3`)	One or two register operands allowed per instruction (e.g., `add R1, R2`)
Parameter passing through efficient on-chip register windows	Parameter passing through inefficient off-chip memory
Single-cycle instructions (except for `load` and `store`)	Multiple-cycle instructions
Hardwired control	Microprogrammed control
Highly pipelined	Less pipelined
Simple instructions that are few in number	Many complex instructions
Fixed length instructions	Variable length instructions
Complexity in compiler	Complexity in microcode
Only load and store instructions can access memory	Many instructions can access memory
Few addressing modes	Many addressing modes

9.3 Flynn's Taxonomy(费林分类法)

- 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.

9.3 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.
  - **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.
  - **MISD:** Multiple instruction streams, single data stream.

9.3 Flynn's Taxonomy

- Flynn's Taxonomy falls short in a number of ways:
- There appears to be no need for MISD machines.
- Flynn assumed that parallelism was homogeneous.
  - A machine could conceivably have four separate floating-point adders, two multipliers, and a single integer unit. This machine could therefore execute seven operations in parallel, but it does not readily fit into Flynn's classification system
- 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:

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

SMP $\Rightarrow$ fewer processors + shared memory + communication via memory

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

9.3 Flynn's Taxonomy

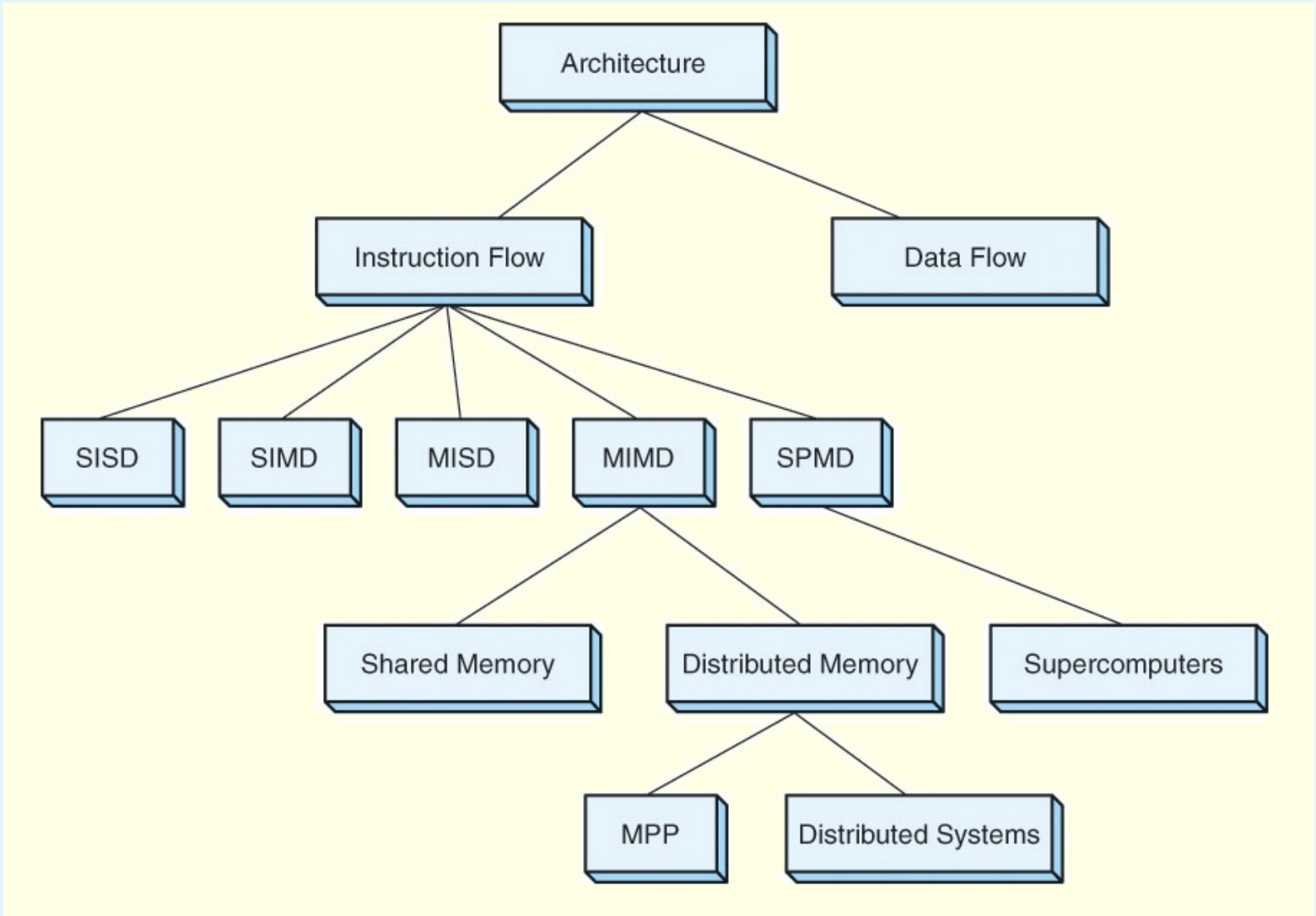
- Flynn's Taxonomy has been expanded to include SPMD (single program, multiple data) architectures.
- SIMT(Single Instruction Multiple Thread)
- Each SPMD processor has its own data set and program memory. Different nodes can execute different instructions within the same program using instructions similar 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

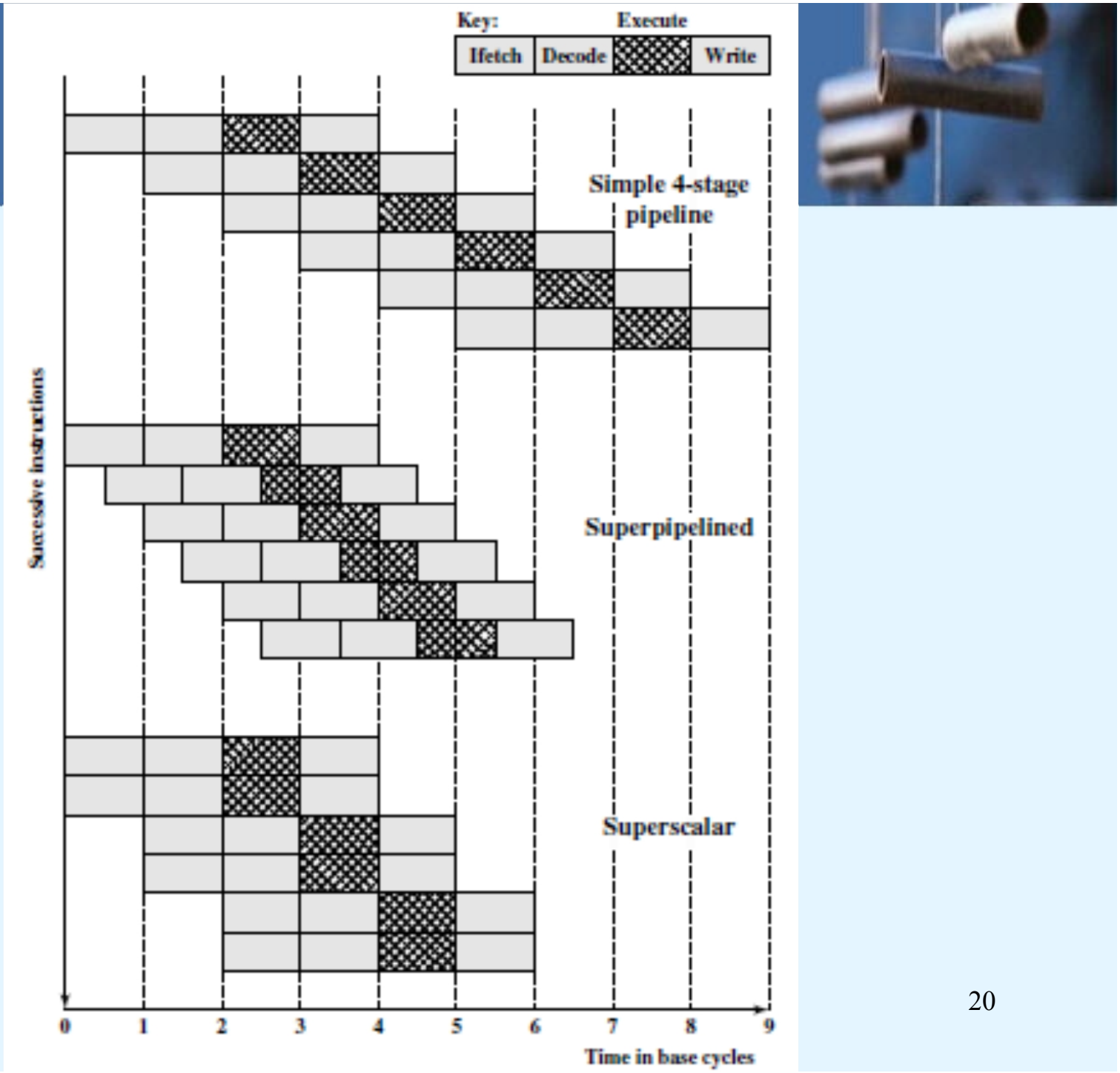


9.4 Parallel and Multiprocessor Architectures

- Parallel processing is capable of economically increasing system throughput while 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.
  - 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.

9.4 Parallel and Multiprocessor Architectures

- Recall that pipelining divides the fetch-decode-execute cycle into stages that each carry out a small part of the process on a set of instructions.
- 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.


Key:
Execute
Ifetch
Decode
Write
Simple 4-stage pipeline
Superpipelined
Superscalar
Successive instructions
0
1
2
3
4
5
6
7
8
9
Time in base cycles

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 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.

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.

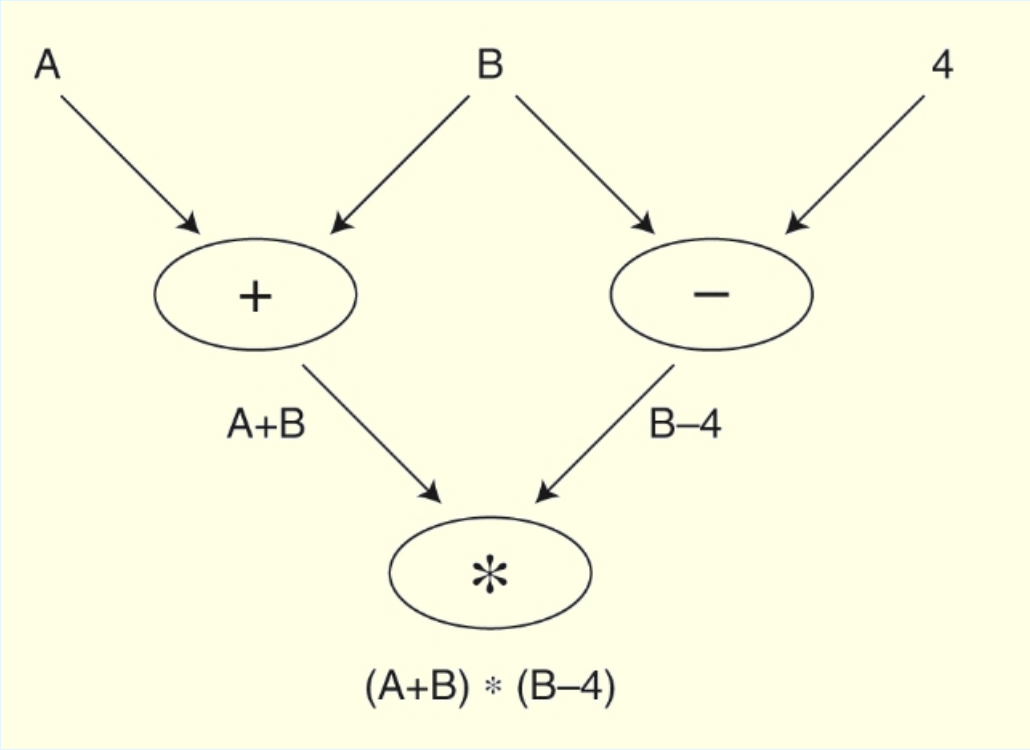
9.5 Alternative Parallel Processing Approaches

- 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.

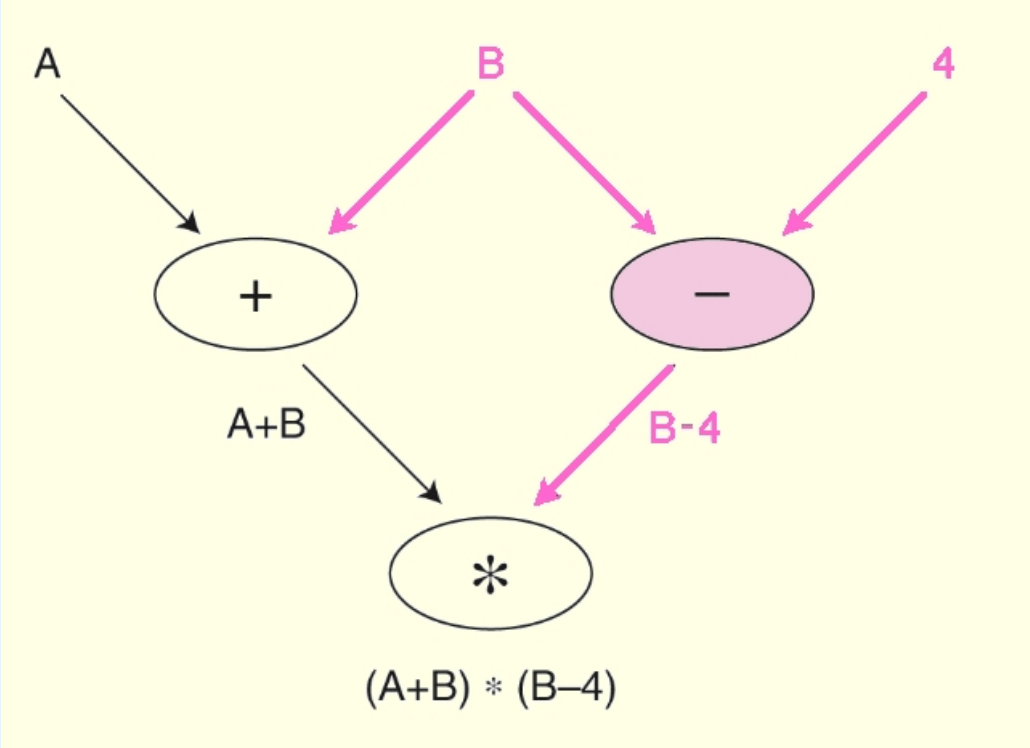
Its nodes contain the instructions and its arcs indicate the data dependencies.



9.5 Alternative Parallel Processing Approaches

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

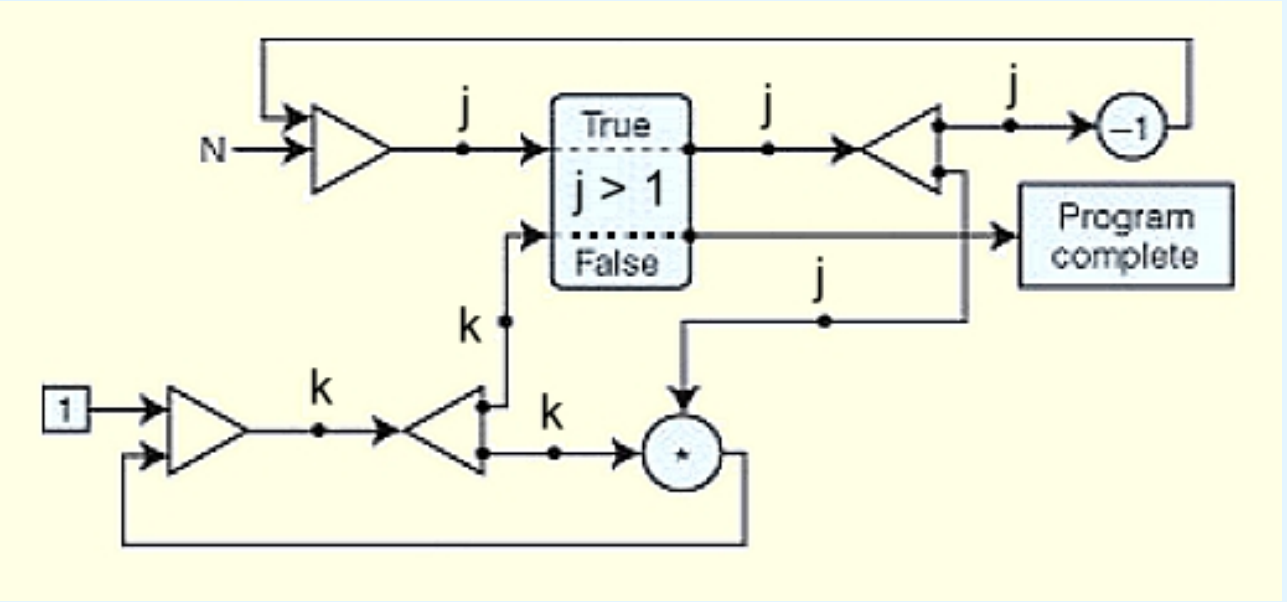
The result is placed on an output arc.



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<- * j;
    new j <- j - 1;
return k)
```



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.