The Essentials of Computer Organization and Architecture

Linda Null and Julia Lobur
Jones and Bartlett Publishers, 2003

Chapter 9 Instructor's Manual

Chapter Objectives

Chapter 9, Alternative Architectures, provides an overview of alternative architectures that have emerged in recent years. RISC, Flynn's Taxonomy, parallel processors, instruction-level parallelism, multiprocessors, interconnection networks, shared memory systems, cache coherence, memory models, superscalar machines, neural networks, systolic architectures, dataflow computers, and distributed architectures are covered. Our main objective in this chapter is to help the reader realize we are not limited to the von Neumann architecture, and to force the reader to consider performance issues, setting the stage for the next chapter. Lectures should focus on the following points:

- **RISC machines.** RISC architectures were introduced in Chapters 4 and 5. This chapter goes into more detail, covering the RISC versus CISC debate, speedup, and overlapping register windows.
- **Flynn's Taxonomy.** This is the most widely accepted method for classifying computer architectures. It is based on the number of instructions and the number of data streams, and includes SISD, SIMD, MISD, and MIMD.
- **Parallel and multiprocessor architectures.** This section covers SIMD and MIMD architectures, including superscalar, VLIW, and vector processors, with examples. Both shared memory machines (SMM, UMA, and NUMA) and interconnection networks are discussed.
- **Interconnection networks.** It is important to understand how the processors in MIMD systems communicate with the various memories and with each other. Interconnection networks are often used, and are categorized according to topology. Completely connected, star, linear, ring, mesh, tree, and hypercube are covered, and examples of each are given. In addition, dynamic networks (using 2x2 switches and stages) are also introduced.
- **Alternative parallel processing approaches.** Although the traditional von Neumann architecture is a popular platform for study, it is important to cover alternatives to this architecture to gain a better overall understanding of computer architecture. Indeed, some highly complex problems cannot be solved using our traditional model of computation . This section discusses dataflow machines (which use a data-driven architecture), neural networks, and systolic arrays.

Required Lecture Time

The important concepts in Chapter 9 can typically be covered in 3 lecture hours (this provides only an exposure to the topics). However, if a teacher wants the students to have a mastery of all topics in Chapter 9, 9 lecture hours are more reasonable.

Lecture Tips

Students often have difficulty seeing the motivation for RISC machines. It is important to emphasize *why* RISC machines are different. The English versus Chinese language example is a very good motivator for this difference. When discussing parallel architectures, it is important to note that parallel does not always mean better -- there are some applications that do not benefit from parallelism. When discussing interconnection networks, instructors should make sure the students understand why these networks are necessary (often, students grasp the concept, but don't understand the overall picture). Explaining shared memory versus non-shared memory architectures will help with this. Dataflow computing and neural networks present a totally different concept in architecture than students are used to, so if instructors opt to cover these topics, time should be spent on how these architectures differ from the traditional von Neumann approach.

Answers to Exercises

♦ 1. Why do RISC machines operate on registers?

Ans.

RISC machines limit the instructions that can access memory to load and store instructions only. This means all other instructions use registers. This requires fewer cycles and speeds up the execution of the code, and thus the performance of the hardware. The goal for RISC architectures is to achieve single-cycle instructions, which could would not be possible if instructions had to access memory instead of registers.

---

2. Which characteristics of RISC systems could be directly implemented in CISC systems? Which characteristics of RISC machines could not be implemented in CISC machines (based on the defining characteristics of both architecture as listed in Table 9.1)?

Ans.

Three register operands, hardwired control, highly pipelined, fixed-length instructions, a small number of simple instructions, complexity in compiler, and allowing few addressing modes could be implemented in CISC. Multiple register sets would require additional hardware, as would allowing parameter passing via on-chip register windows, so these would be difficult to implement on traditional CISC chips. The same is true with significant pipelining. (Note: if a designer were to implement the CISC chip with these features, then there would be no problem.) Limiting memory accesses to only load and store instructions would cause serious problems with CISC instruction sets, as would limiting the types of addressing modes.

♦ 3. What does the "reduced" in *reduced instruction set computer* really mean?

Ans.

"Reduced" originally meant providing a set of minimal instructions that could carry out all essential operations: data movement, ALU operations and branching. However, the main

objective in RISC machines today is to simplify instructions so they can execute more quickly. Each instruction performs only one operation, they are all the same size, they have only a few different layouts, and all arithmetic operations must be performed between registers (data in memory cannot be used as an operand).

---

4. Suppose a RISC machine uses overlapping register windows with:

   10 global registers
   6 input parameter registers
   10 local registers
   6 output parameter registers

   How large is each overlapping register window?

Ans.

6

♦ 5. A RISC processor has 8 global registers and 10 register windows. Each window has 4 input registers, 8 local and 4 output. How many total registers are in this CPU? (HINT: Remember, due to the circular nature of the windows, the output registers of the last window are shared as the input registers of the first window.)

Ans.

(8 + 4) * 10 = 120 + 8 = 128

---

6. A RISC processor has 152 total registers, with 12 designated as global registers. The 10 register windows each have 6 input registers and 6 output registers. How many local registers are in each register window set?

Ans.

8

---

7. A RISC processor has 186 total registers, with 18 globals. There are 12 register windows, each with 10 locals. How many input/output registers are in each register windows?

Ans.

4

---

8. Suppose a RISC machine uses overlapping register windows for passing parameters between procedures. The machine has 298 registers, and each register window has 32 registers, of which 10 are global variables and 10 are local variables. Answer the following:
   a. How many registers would be available for use by input parameters?
   b. How many registers would be available for use by output parameters?
   c. How many register window would be available for use?
   d. By how much would the current window pointer (CWP) be incremented at each procedure call?

Ans.

a. 6 b. 6 c. 18 d. 16

---

9. Recall our discussions from Chapter 8 regarding context switches. These occur when one process stops using the CPU and another process begins. In this sense, register windows could be viewed as a potential weakness of RISC. Explain why this is the case.

Ans.

During a context switch, all information about the currently executing process must be saved, including the values in the register windows. When the process is restored, the values in the register windows must be restored as well. Depending on the size of the windows, this could be a very time-consuming process.

---

10. Suppose that a RISC machine uses 5 register windows.
    a. How deep can the procedure calls go before registers must be saved in memory? (That is, what is the maximum number of "active" procedure calls that can be made before we need to save any registers in memory?)
    b. Suppose two more calls are made after the maximum value from part (a) is reached. How many register windows must be saved to memory as a result?
    c. Now suppose that the most recently called procedure returns. Explain what occurs.
    d. Now suppose one more procedure is called. How many register windows need to be stored in memory?

Ans.

    a. Due to the circular nature of the windows, the output registers of the last window are shared as the input registers of the first window. Therefore, only four procedures could be active without saving registers in memory.
    b. The input registers from the first window, on the first call (due to wrap around), and the input, local, and output registers on the second call.
    c. The previously saved registers are restored.
    d. Again, the input, local, and output values would need to be saved.

---

11. In Flynn's taxonomy:
    a. What does SIMD stand for? Give a brief description and an example.
    b. What does MIMD stand for? Give a brief description and an example.

Ans.

a. Single instruction, multiple data. One specific instruction executes on multiple pieces of data. For example, a vector processor adding arrays uses one instruction (C[i] = A[i] + B[i]), but can perform this instruction on several pieces of data (C[1] = A[1] + B[1], C[2] = A[2] + B[2], C[3] = A[3] + B[3], and so on), depending on how many ALUs the vector processor contains.

b. Multiple instruction, multiple data. Many instructions can execute in parallel on multiple pieces of data. For example, a multiprocessor has independent processing units that can execute individual instructions on specific data, resulting in multiple instructions executing on multiple pieces of data.

---

12. Flynn's taxonomy consists of four primary models of computation. Briefly describe each of the categories and give an example of a high-level problem for which each of these models might be used.

Ans.

SISD, or single instruction, single data stream, is the typical von Neumann architecture. Standard uniprocessors are examples of SISD machines. Word processing and normal, everyday activities for most users would be well suited to this model. SIMD, or single instruction, multiple data steam, executes one specific instruction on many pieces of data (see answer to 11a). This model is very useful in matrix operations. MISD, multiple

instructions, single data, is not very useful, so no examples will be given. MIMD, multiple instructions, multiple data, consists of multiprocessors and most current parallel systems.

♦13. Explain the difference between loosely coupled and tightly coupled architectures.

Ans.

Loosely coupled and tightly coupled are terms that describe how multiprocessors deal with memory. If there is one large, centralized, shared memory, we say the system is tightly coupled. If there are multiple, physically distributed memories, we say the system is loosely coupled.

---

14. Describe the characteristics of MIMD multiprocessors that distinguish them from multicomputer systems or computer networks.

Ans.

Multiprocessors typically are tightly coupled, whereas multicomputer systems and computer networks tend to be loosely coupled.

---

15. How are SIMD and MIMD similar? How are they different? Note, you are not to define the terms, but instead compare the models.

Ans.

Both SIMD and MIMD machines have multiple processors and can operate on different pieces of data in parallel. However, in SIMD, all processors must execute the same instruction at the same time, while in MIMD, each processor can execute a different instruction.

---

16. What is the difference between SIMD and SPMD?

Ans.

SIMD is single instruction, multiple data. SPMD is single program, multiple data. SPMD differs from SIMD in that the processors can do different things at the same time.

♦17. For what type of program-level parallelism (data or control) is SIMD best suited? For what type of program-level parallelism is MIMD best suited?

Ans.

SIMD is best suited for data parallelism; MIMD is best suited for control or task parallelism.

---

18. Describe briefly and compare the VLIW and superscalar models with respect to instruction level parallelism.

Ans.

In VLIW, the processor has several independent execution units. The compiler groups multiple instructions into a single long word, which is presented to the hardware.

In superscalar machines, the processor has several independent execution units as well. The hardware can dispatch multiple instructions in a single cycle, as long as the required EUs are free.

Superscalar processors rely on both the hardware (to arbitrate dependencies) and the compiler (to generate approximate schedules), whereas VLIW processor rely entirely on the compiler.

♦ 19. Which model, VLIW or Superscalar, presents the greater challenge for compilers? Why?
Ans.

Whereas superscalar processors rely on both the hardware (to arbitrate dependencies) and the compiler (to generate approximate schedules), VLIW processors rely *entirely* on the compiler (see Exercise 18). Therefore, VLIW moves the complexity completely to the compiler.

♦ 20. Compare and contrast the superscalar architecture to the VLIW architecture.
Ans.

Both architectures feature a small number of parallel pipelines for processing instructions. However, the VLIW architecture relies on the compiler for prepackaging and scheduling the instructions in a correct and efficient way. In the superscalar architecture, instruction scheduling is done by hardware.

♦ 21. Why are distributed systems desirable?
Ans.

Distributed systems allow for resource sharing (such as sharing of printers and files), and thus can reduce system cost. They also allow for redundancy, which increases reliability (if one site fails, the remaining sites can still function). These systems also speed up computation, as jobs can be distributed and run concurrently at many sites. Lastly, distributed systems run programs that, due to the nature of the system, can share data with other systems more easily via the communications network and communicate with remote sites.

---

22. What is the difference between UMA and NUMA?
Ans.

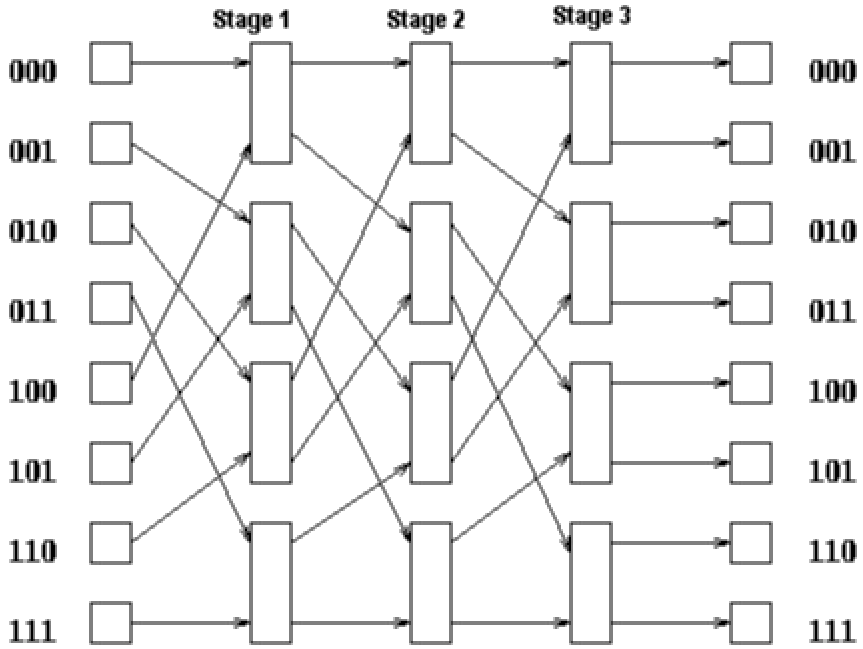
While both architectures allow hardware access to remote memory, in UMA machines, all processors can access any given memory in the same amount of time. NUMA machines allow processors to access local memories more quickly than remote memories (hence the term "non-uniform).

♦ 23. What are the main problems with using crossbars for interconnection networks? What problems do buses present in interconnection networks?
Ans.

When adding processors to a crossbar interconnection network, the number of crossbar switches grows to an unmanageable size very quickly. Bus networks suffer from potential bottlenecks and contention issues.

---

24. Given the following Omega network which allows 8 CPUs (P0 through P7) to access 8 memory modules (M0 through M7).



a. Show the following connections through the network:
   i) P0 → M2
   ii) P4 → M4
   iii) P6 → M3

b. Can these connections occur simultaneously or do they conflict? Explain.

c. List a processor-to-memory access that conflicts (is blocked) by the access P0 → M2 that is not listed in part (a).

d. List a processor-to-memory access that is not blocked by the access P0 → M2 and is not listed in part (a).

Ans.

a. No answer

b. No, there is a conflict between P0 → M2 and P6 → M3 in switch 3B.

c. Any access requiring switch 1A to be cross, 2A to be through, or 3B to be cross.

d. Any access NOT requiring 1A, 2A, or 3B, or those utilizing 1A through, 2A cross, or 3B through.

♦ 25. Describe write-through and write-back cache modification as they are used in shared memory systems, and the advantages and disadvantages of both approaches.

Ans.

With write-through cache modification, the new value is immediately flushed through to the server. This keeps the server up-to-date, but writing takes longer (losing the speed increase that caching typically provides). This is ok if the majority of accesses are reads. With write-back, the new value is flushed to the server after a given delay. This maintains the speed increase, but means that if the server crashes before the new data is flushed, some data may be lost. This is a good example to illustrate that performance improvements often come at a price.

---

26. Should the memory of a dataflow system be associative or address-based? Explain.

Ans.

Since instructions do not reference memory (they instead reference other instructions), it would make more sense for memory to be associative. Also, in dynamic dataflow systems, during every clock cycle, memory is searched for the required set of tokens, requiring associative memory.

♦ 27. Is it true that neural networks process information based on a sequential algorithm? Explain.

Ans.

Yes and no. Yes, in that individual neurons take input, process it, and provide output. This output is then used by another neuron "down the line". However, the neurons themselves work in parallel.

---

28. Compare and contrast supervised learning and unsupervised learning with regard to neural networks.

Ans.

Supervised learning assumes that for each input, the output is known a priori. In unsupervised learning, no outputs are known in advance.

♦ 29. Describe the process of supervised learning in neural networks from a mathematical perspective.

Ans.

While the network is learning, indications of incorrect output are used to make adjustments to the weights in the network. These adjustments are based on various optimization algorithms. The learning is complete when the weighted average converges to a given value.

---

30. These two questions deal with neural networks, in particular, a single perceptron.

a. The logical "not" is a little trickier than AND or OR, but can be done. In this case, there is only one Boolean input. What would the weight and threshold be for this perceptron to recognize the logical NOT operator?

b. Show that it is not possible to solve the binary XOR problem for two inputs x_1 and x_2 using a single perceptron.

Ans.

a. If the weight of the edge is -1, the input (which is either 0 or 1) becomes 0 or -1. Setting the threshold to 0 results in a 0 input reaching the threshold, with an output of 1, while an input of -1 does not reach the threshold, resulting in an output of 0.

---

31. Explain the differences between SIMD and systolic array computing, when the systolic array is one-dimensional.

Ans.

Systolic arrays are variations of SIMD computers that incorporate large arrays of simple processor using vector pipelines for data flow.

---

32. With respect to Flynn's taxonomy, where do systolic arrays fit? What about clusters of workstations?

Ans.

Systolic arrays are SIMD machines, while clusters of workstations are MIMD.

---

33. Indicate whether each of the following applies to **CISC** or **RISC** by placing either a **C** (for CISC) or an **R** (for RISC) in the blank.

_____ 1. Simple instructions averaging 1 clock cycle to execute

_____ 2. Single register set

_____ 3. Complexity is in the compiler

_____ 4. Highly pipelined

_____ 5. Any instruction can reference memory

_____ 6. Instructions are interpreted by the microprogram

_____ 7. Fixed length, easily decoded instruction format

_____ 8. Highly specialized, infrequently used instructions

_____ 9. Use of overlapping register windows

_____ 10. Relatively few addressing modes

Ans.

R, C, R, R, C, C, R, C, R, R
