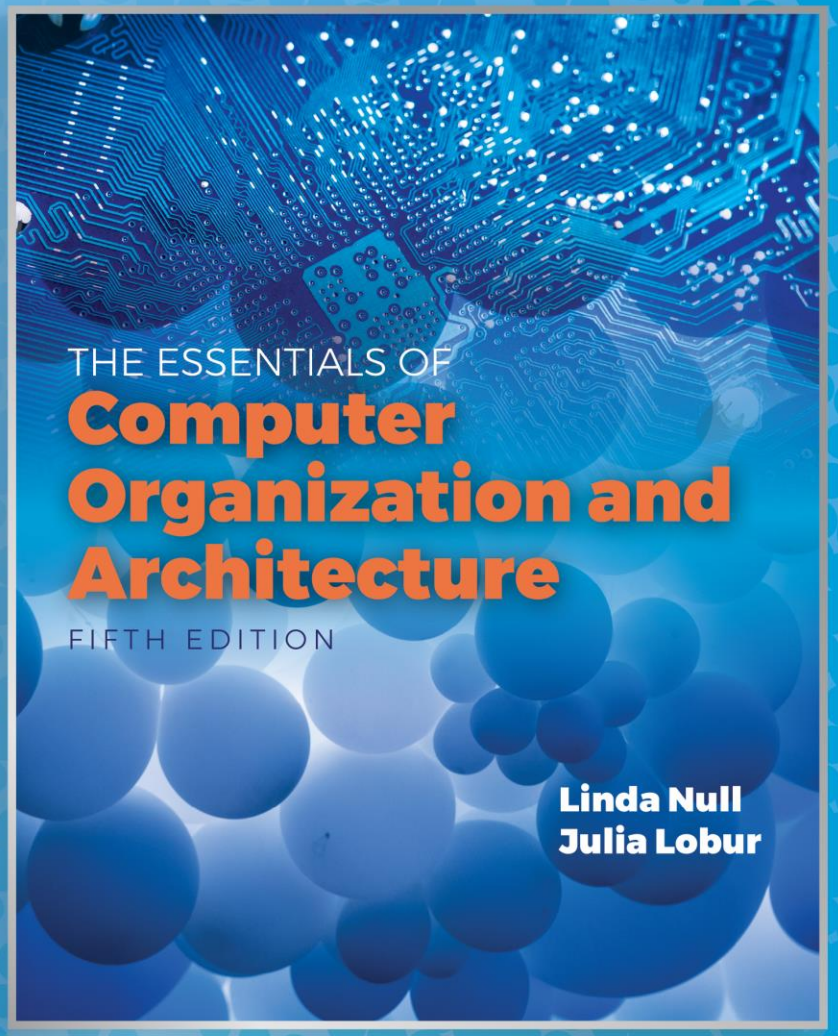
This is the first lecture of Chapter 9

Chapter 9

Alternative Architectures (A)



Quick review of last lecture

- Programming Tools:
  - Assembler,
  - linker,
  - Compilers, and
  - Interpreters
- Java: All of the Above.

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 (1 of 9)

- 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 (2 of 9)

- 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 (3 of 9)

- 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 (4 of 9)

- Consider 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
```

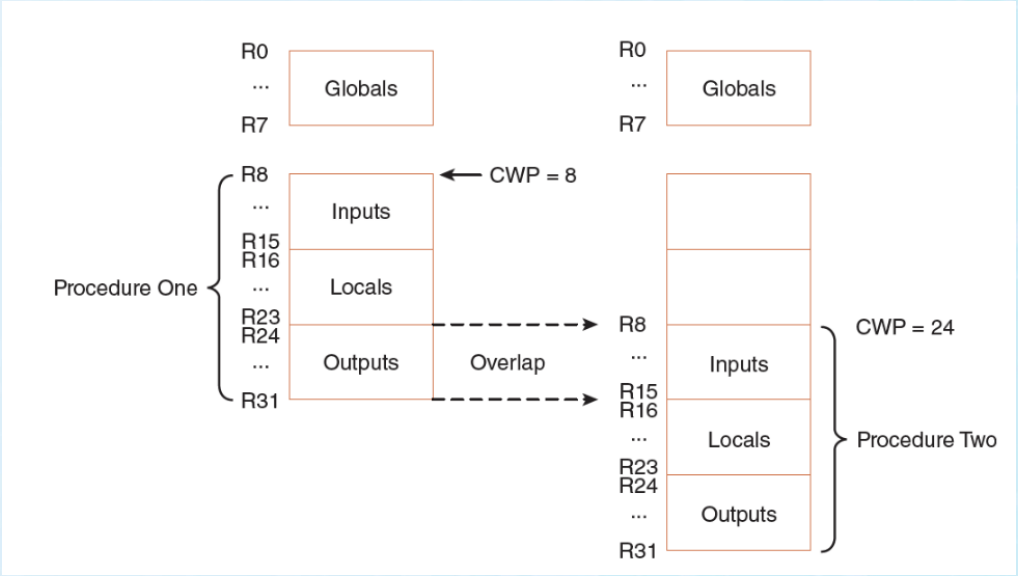
- The total clock cycles for the CISC version might be:
  - **(2 movs × 1 cycle) + (1 mul × 30 cycles) = 32 cycles**
- While the clock cycles for the RISC version is:
  - **(3 movs × 1 cycle) + (5 adds × 1 cycle) + (5 loops × 1 cycle) = 13 cycles**
- With RISC clock cycle being shorter, RISC gives us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

9.2 RISC Machines (5 of 9)

- Because of their load-store ISAs, RISC architectures require a large number of CPU registers.
- These registers 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.

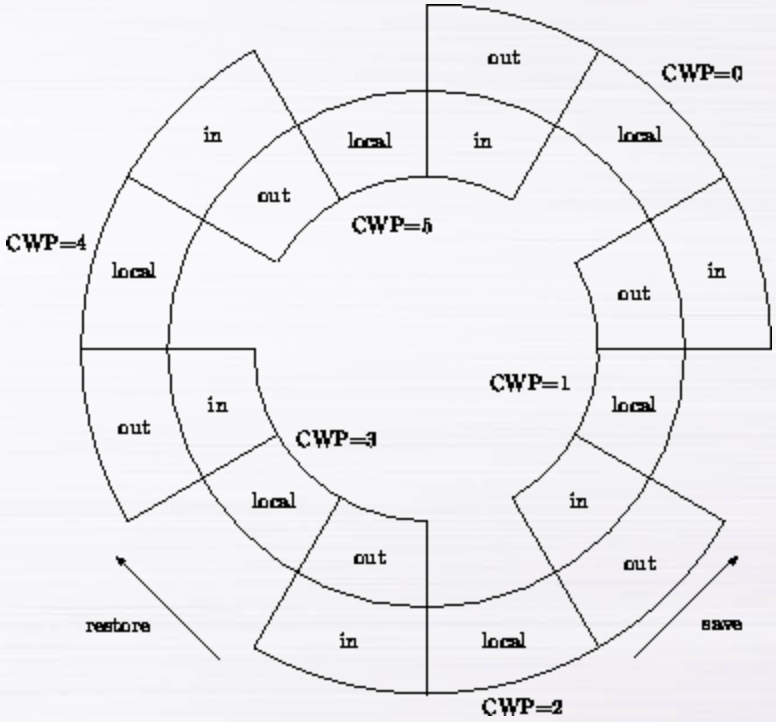
9.2 RISC Machines (6 of 9)

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



Example: Overlapping Register Windows

- A RISC processor has 16 global registers and 6 register windows. Each window has 4 input registers, 8 local and 4 output. How many total registers are in this CPU?



- (8 + 4) * 6 + 16 = 72 + 16 = 88

9.2 RISC Machines (7 of 9)

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

9.2 RISC Machines (8 of 9)

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

9.2 RISC Machines (9 of 9)

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

9.3 Flynn's Taxonomy (1 of 7)

- 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 (2 of 7)

- 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 (3 of 7)

- 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.
  - 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 (4 of 7)

- 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 (5 of 7)

- 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 cluster 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.

9.3 Flynn's Taxonomy (6 of 7)

- Flynn's Taxonomy has been expanded to include SPMD (single program, multiple data) 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 (7 of 7)