# Chapter 3: Some Real Machines

## Topics

3.1 Machine Characteristics and Performance

3.2 RISC versus CISC

3.3 A CISC Microprocessor: The Motorola MC68000

3.4 A RISC Architecture: The SPARC

# Practical Aspects of Machine Cost-Effectiveness

- Cost for useful work is fundamental issue
- Mounting, case, keyboard, etc. are dominating the cost of integrated circuits
- Upward compatibility preserves software investment
    - Binary compatibility
    - Source compatibility
    - Emulation compatibility
- Performance: strong function of application

# Performance Measures

- MIPS: Millions of Instructions Per Second
  - Same job may take more instructions on one machine than on another
- MFLOPS: Million Floating Point OPs Per Second
  - Other instructions counted as overhead for the floating point
- Whetstones: Synthetic benchmark
  - A program made up to test specific performance features
- Dhrystones: Synthetic competitor for Whetstone
  - Made up to “correct” Whetstone’s emphasis on floating point
- SPEC: Selection of “real” programs
  - Taken from the C/Unix world

# CISC Versus RISC Designs

- **CISC: Complex Instruction Set Computer**
  - **Many complex instructions and addressing modes**
  - **Some instructions take many steps to execute**
  - **Not always easy to find best instruction for a task**
- **RISC: Reduced Instruction Set Computer**
  - **Few, simple instructions, addressing modes**
  - **Usually one word per instruction**
  - **May take several instructions to accomplish what CISC can do in one**
  - **Complex address calculations may take several instructions**
  - **Usually has load-store, general register ISA**

# Design Characteristics of RISCs

- Simple instructions can be done in few clocks
  - Simplicity may even allow a shorter clock period
- A pipelined design can allow an instruction to complete in every clock period
- Fixed length instructions simplify fetch and decode
- The rules may allow starting next instruction without necessary results of the previous
  - Unconditionally executing the instruction after a branch
  - Starting next instruction before register load is complete