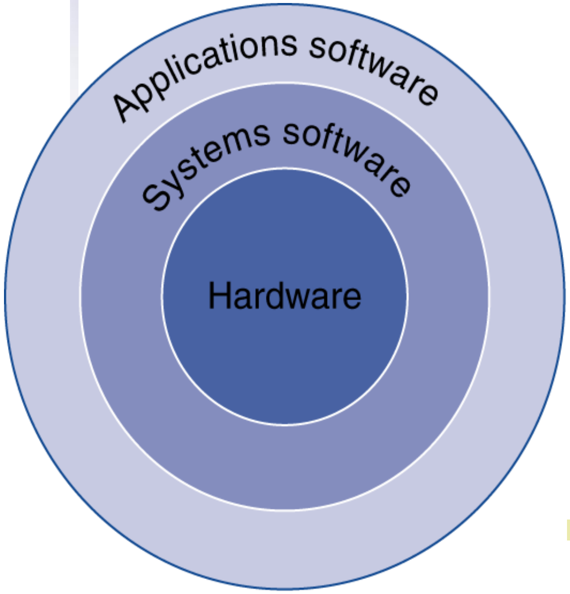
Chapter 1 (continued)

Computer Abstractions and Technology

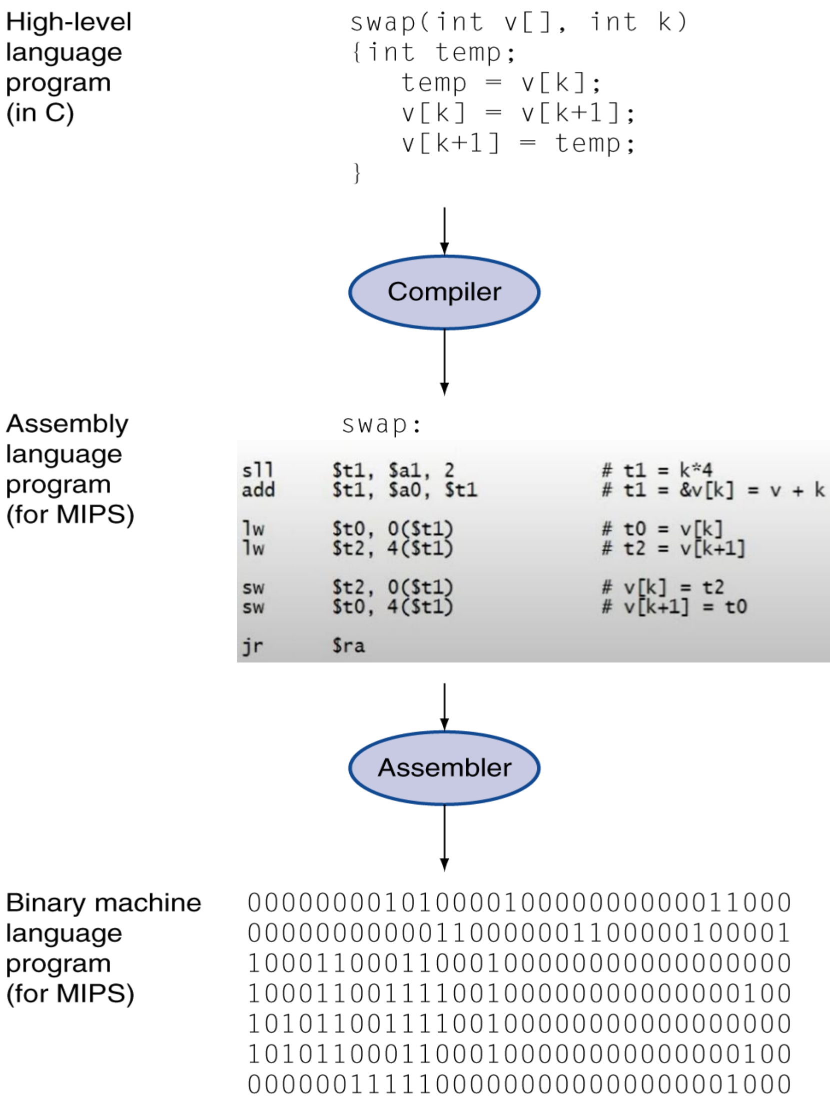
Below Your Program



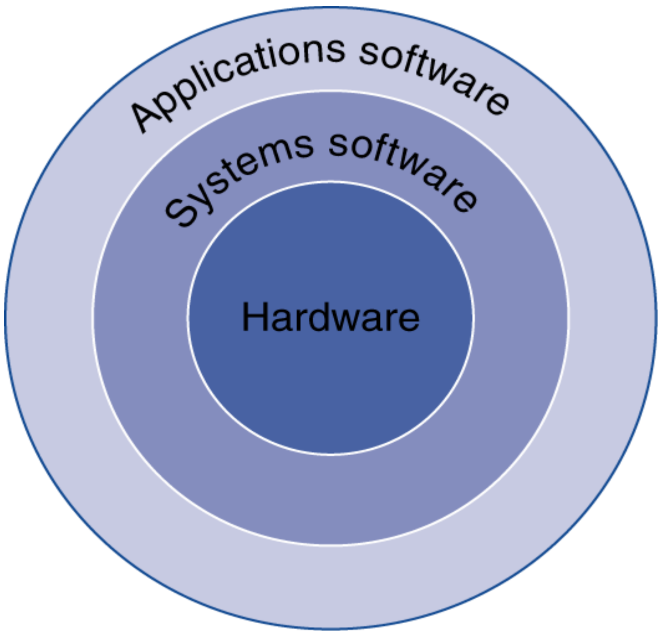
- Application software
  - Written in High-Level Language (HLL)
- System software
  - Compiler: translates HLL code to machine code
  - Operating System:
    - Handling input/output
    - Managing memory and storage
    - Scheduling tasks & sharing resources
- Hardware
  - Processor, memory, I/O controllers

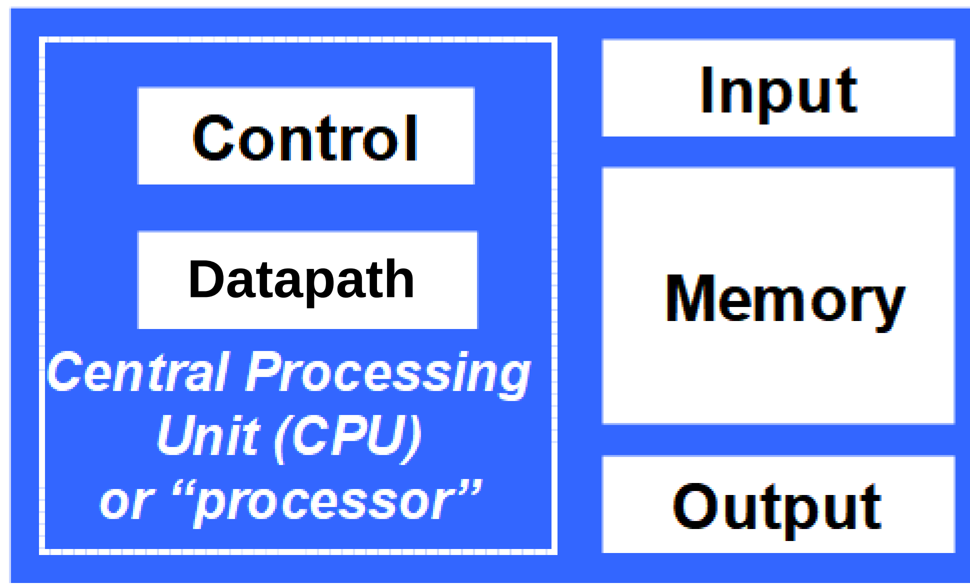
Levels of Program Code

- High-level language
  - Level of abstraction closer to problem domain
  - Provides for productivity and portability
- Assembly language
  - Textual representation of instructions
- Hardware representation
  - Binary digits (bits)
  - Encoded instructions and data



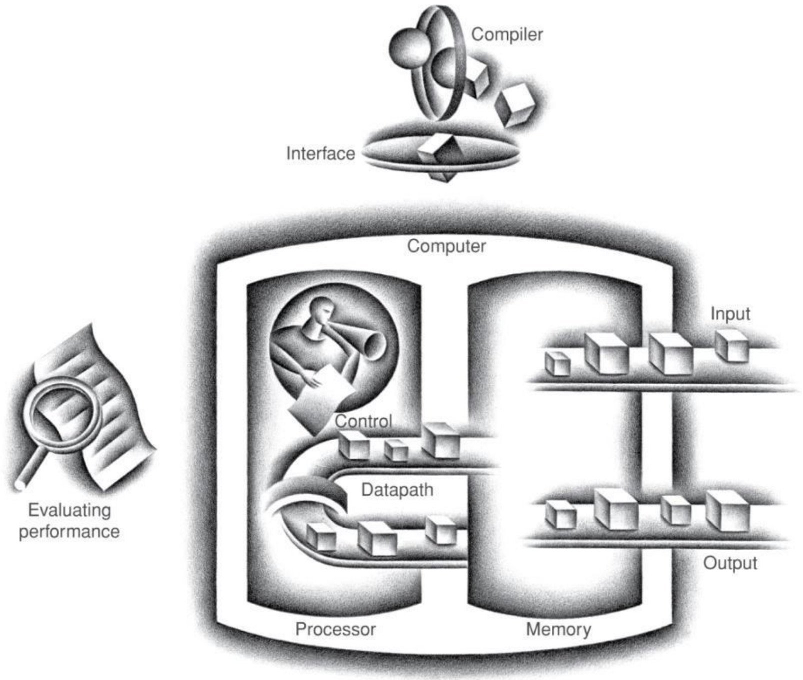
Hardware Organization of Computer





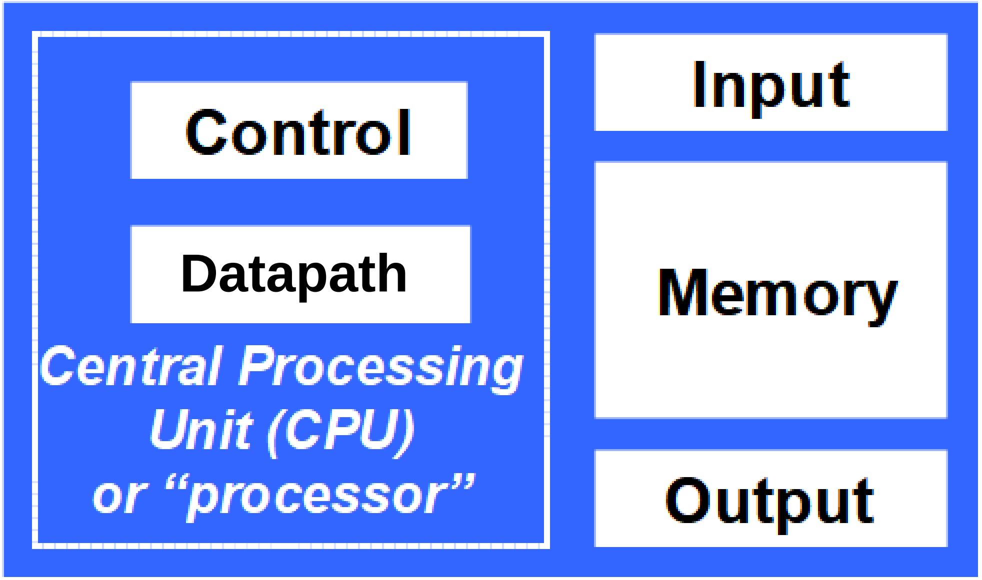
Components of a Computer

The BIG Picture



- Same components for all kinds of computer
  - Desktop, Server, Embedded
- Input/Output includes
  - User-interface devices
    - Display, keyboard, mouse
  - Storage devices
    - Hard disk, CD/DVD, flash
  - Network adapters
    - For communicating with other computers

We need a language? ISA



Instruction Set Architecture (ISA)

- A set of assembly language instructions (ISA) provides a link between software and hardware.
- Given an instruction set, software programmers and hardware engineers work more or less independently.
- ISA is designed to extract the most performance out of the available hardware technology.

Architecture and Organization

Software
Programmers

Hardware
Engineers

ISA

- Defines registers
- Defines data transfer modes between registers, memory and I/O
- Types of ISA: RISC, CISC, VLIW, Superscalar
- Examples:
  - IBM370/X86/Pentium/K6 (CISC)
  - PowerPC (Superscalar)
  - Alpha (Superscalar)
  - ARM (RISC)
  - MIPS (RISC and Superscalar)
  - Sparc (RISC), UltraSparc (Superscalar)
  - RISC V (Open Source, RISC)

Computer *Architecture*

- Architecture: System attributes that have a direct impact on the logical execution of a program
- Architecture that is visible to a programmer:
  - Instruction set
  - Data representation
  - I/O mechanisms
  - Memory addressing

Computer *Organization*

- Organization: Physical details that are transparent to a programmer, such as
  - Hardware implementation of an instruction
  - Control signals
  - Memory technology used
- Example: x86 architecture has been used by both Intel and AMD computers, which may differ in their organization.

CPU Design

- Design and implementation of a processor:
  - Define instruction set
  - Design datapath and control hardware
  - Implement hardware in FPGA (Field-programmable gate array - configurable integrated circuit)
  - Verify

Abstractions in Comp Org & Arch

The BIG Picture

- Abstraction helps us deal with complexity
  - Hides lower-level details
- Instruction Set Architecture (ISA) or Computer Architecture
  - The hardware/software interface
  - Includes instructions, registers, memory access, I/O, and so on
- Operating system hides details of doing I/O, allocating memory from programmers

Research and Developments of Continuing Interest

- Instruction level parallelism (ILP)
- Multi-core systems and chip multi-processing (CMP)
  - Processors
  - Inter-processor communication
  - Memory organization
  - Operating system
  - Programming languages
  - Computing algorithms
- Energy efficiency and low power design
- Embedded systems
- Quantum computing, biological computing, . . .

What You Will Learn: Comp Org & Arch

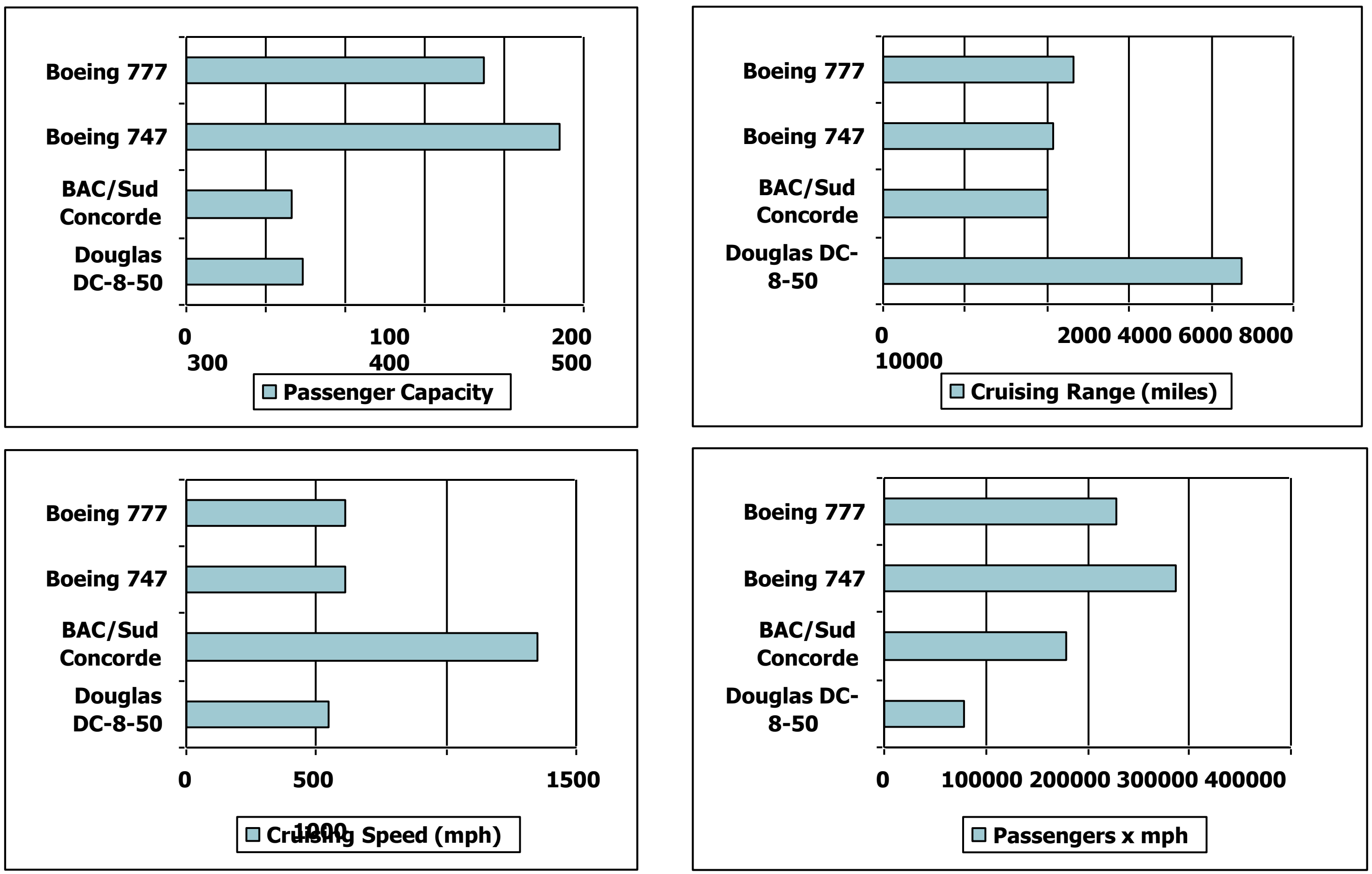
- How programs are translated into machine language
  - and how hardware executes them
- The hardware/software interface
- What determines program performance
  - and how it can be improved
- How hardware designers design computer & improve performance
- What is parallel processing

Understanding Performance

- Algorithm
  - determines number of operations executed
- Programming language, compiler, architecture
  - determine number of machine instructions executed per operation
- Processor and memory system
  - determine how fast instructions are executed
- I/O system (including OS)
  - determine how fast I/O operations are executed

Defining Performance

- Which airplane has the best performance?



Response Time and Throughput

Performance in computers

- Response time
  - How long it takes to do a task
- Throughput
  - Total work done per unit time

    e.g., tasks/transactions/… per hour
- How are response time and throughput affected by
  - Replacing the processor with a faster version?
  - Adding more processors?
- We'll focus on response time for now…

Relative Performance

- Define Performance = 1/Execution Time
- "X is n times faster than Y"

$$\text{Performance}_X / \text{Performance}_Y$$
$$= \text{Execution time}_Y / \text{Execution time}_X = n$$

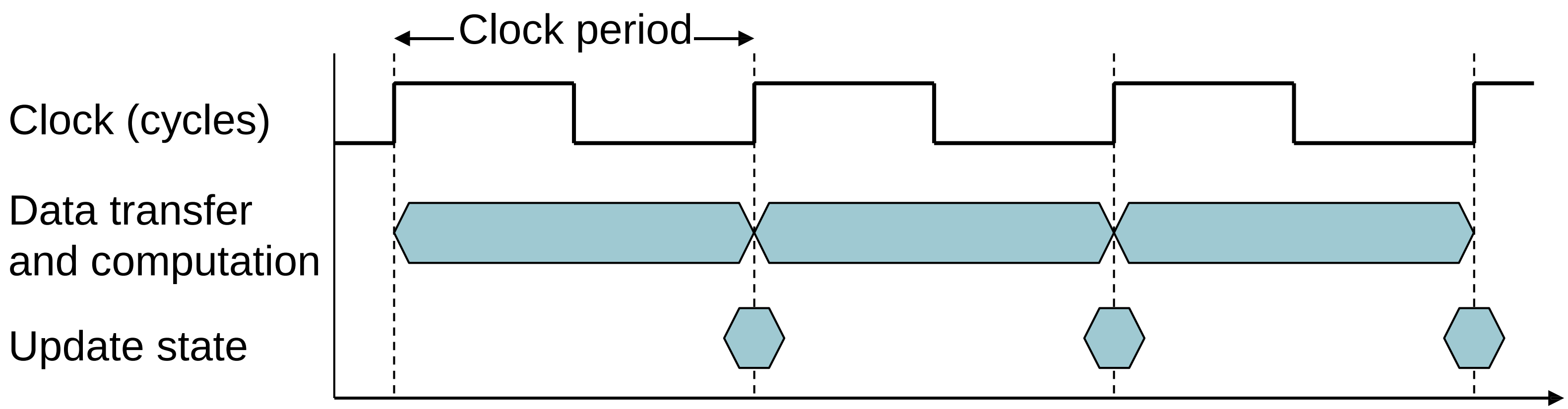
- Example: time taken to run a program
  - 10s on A, 15s on B
  - $\text{Execution Time}_B / \text{Execution Time}_A$
    = 15s / 10s = 1.5 = 1½
  - So A is 1½ times faster than B

Measuring Execution Time

- Elapsed time
  - Total response time, including all aspects
    - Processing, I/O, OS overhead, idle time
  - Determines system performance
- CPU time
  - Time spent processing a given job
    - Minus I/O time, other jobs' shares
  - Includes user CPU time and system CPU time
  - Different programs are affected differently by CPU and system performance
    - Running on servers – I/O performance – hardware and software
    - Total elapsed time is of interest
    - Define performance metric and then proceed

CPU Clocking

- Clock speed is the number of times a second that a circuit operates



- Clock period: duration of a clock cycle
  - e.g., 250ps = 0.25ns = 250×10^{-12}s (1 ps = 10^{-12}s)
- Clock frequency (rate): cycles per second
  - e.g., 4.0GHz = 4000MHz = 4.0×10^{9}Hz

CPU Time

$$\text{CPU Time} = \text{CPU Clock Cycles} \times \text{Clock Cycle Time}$$
$$= \frac{\text{CPU Clock Cycles}}{\text{Clock Rate}}$$

- Performance can be improved by
  - Reducing <u>number</u> of clock cycles
  - Increasing clock rate
  - Hardware designer must often trade off clock rate against cycle count

CPU Time Example

- Computer A: 2GHz clock, 10s CPU time
- Designing Computer B
  - Aim for 6s CPU time
  - Can do faster clock, but causes 1.2 × clock cycles
- How fast must Computer B clock be?

Let's first find the number of clock cycles required for the program on A:

$$\text{CPU time}_A = \frac{\text{CPU clock cycles}_A}{\text{Clock rate}_A}$$

$$10 \text{ seconds} = \frac{\text{CPU clock cycles}_A}{2 \times 10^9 \dfrac{\text{cycles}}{\text{second}}}$$

$$\text{CPU clock cycles}_A = 10 \text{ seconds} \times 2 \times 10^9 \frac{\text{cycles}}{\text{second}} = 20 \times 10^9 \text{ cycles}$$

CPU Time Example

CPU time for B can be found using this equation:

$$\text{CPU time}_B = \frac{1.2 \times \text{CPU clock cycles}_A}{\text{Clock rate}_B}$$

$$6 \text{ seconds} = \frac{1.2 \times 20 \times 10^9 \text{ cycles}}{\text{Clock rate}_B}$$

$$\text{Clock rate}_B = \frac{1.2 \times 20 \times 10^9 \text{ cycles}}{6 \text{ seconds}} = \frac{0.2 \times 20 \times 10^9 \text{ cycles}}{\text{second}} = \frac{4 \times 10^9 \text{ cycles}}{\text{second}} = 4 \text{ GHz}$$

To run the program in 6 seconds, B must have twice the clock rate of A.

Instruction Count and CPI

$$\text{Clock Cycles} = \text{Instruction Count} \times \text{Cycles Per Instruction}$$

$$\text{CPU Time} = \text{Instruction Count} \times \text{CPI} \times \text{Clock Cycle Time}$$

$$= \frac{\text{Instruction Count} \times \text{CPI}}{\text{Clock Rate}}$$

- Instruction Count for a program
  - Determined by program, ISA, and compiler
- Average cycles per instruction
  - Determined by CPU hardware
  - If different instructions have different CPI
    - Average CPI gets affected by instruction mix (dynamic frequency of instructions)

CPI Example

- Computer A: Cycle Time = 250ps, CPI = 2.0
- Computer B: Cycle Time = 500ps, CPI = 1.2
- Same ISA
- Which is faster? by how much?

$$\text{CPU Time}_A = \text{Instruction Count} \times \text{CPI}_A \times \text{Cycle Time}_A$$

$$= I \times 2.0 \times 250\text{ps} = I \times 500\text{ps}$$

A is faster…

$$\text{CPU Time}_B = \text{Instruction Count} \times \text{CPI}_B \times \text{Cycle Time}_B$$

$$= I \times 1.2 \times 500\text{ps} = I \times 600\text{ps}$$

$$\frac{\text{CPU Time}_B}{\text{CPU Time}_A} = \frac{I \times 600\text{ps}}{I \times 500\text{ps}} = 1.2$$

…by this much