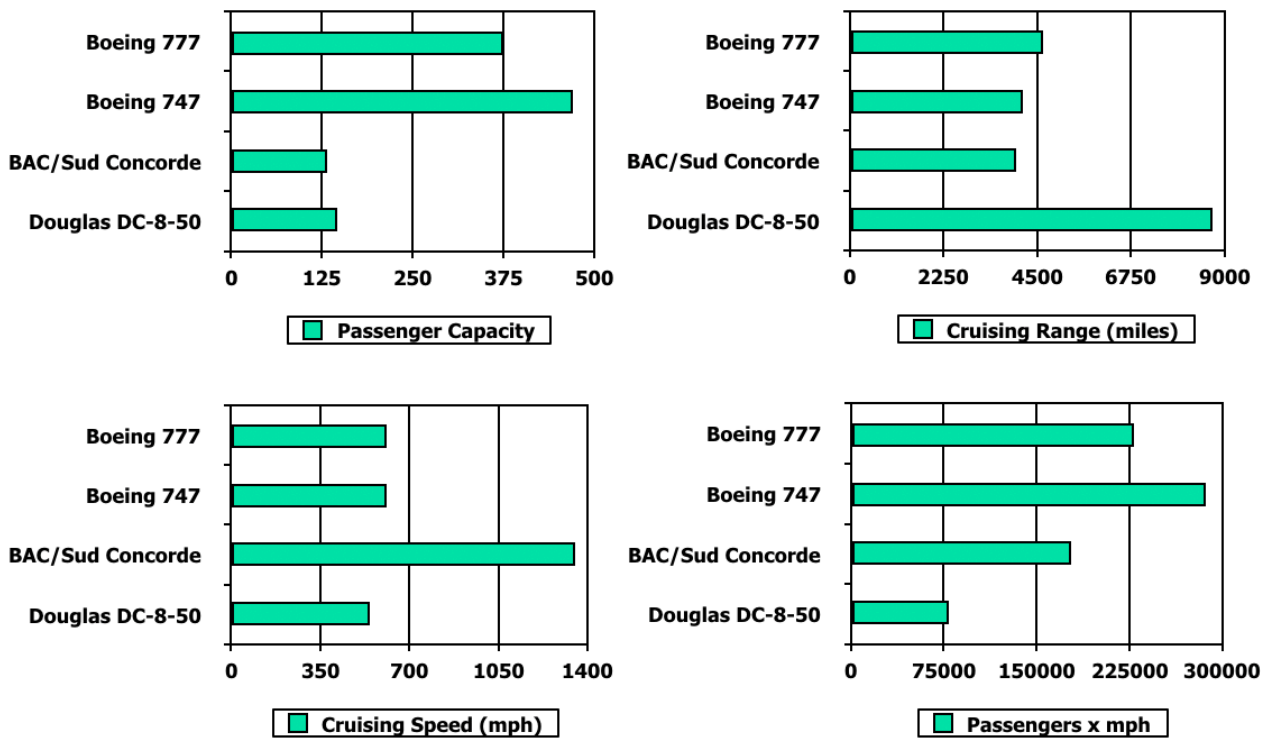
# Chapter 1

## Computer Abstractions and Technology

# Defining Performance

- Which airplane has the best performance? Performance is dependent on the measure of performance
- Measure of performance for airplanes: capacity, cruising range, cruising speed, or passenger throughput (passenger throughput= passenger capacity *speed)
- Let's suppose we define performance in terms of speed; Fastest plane? ( a single passenger: highest cruising speed) (450 passengers: highest passenger throughput)



# Response Time and Throughput

- **Response time (Execution time)**
  - How long it takes to do a single task (the time between the start and the completion of a task); *individual computer users* are interested in reducing response time
- **Throughput (Bandwidth)**
  - Total number of tasks completed per unit time; *data center managers* are interested in increasing throughput
    - e.g., tasks/transactions/… per hour
- **How are response time and throughput affected by**
  - **Replacing the processor with a faster version?** (response time and throughput are improved)
  - **Adding more processors?** (no improvement with response time, but throughout is increased)
- We'll focus on response time for now…

# Relative Performance

- **To maximize performance**, we want to **minimize the response time or execution time** for some tasks
- Performance for computer: **Performance $_x$ = 1/Execution Time $_x$**
- If **performance of computer x is greater than performance of computer y**, **Performance $_x$ > Performance $_y$**
- **If x is n times faster than y**

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

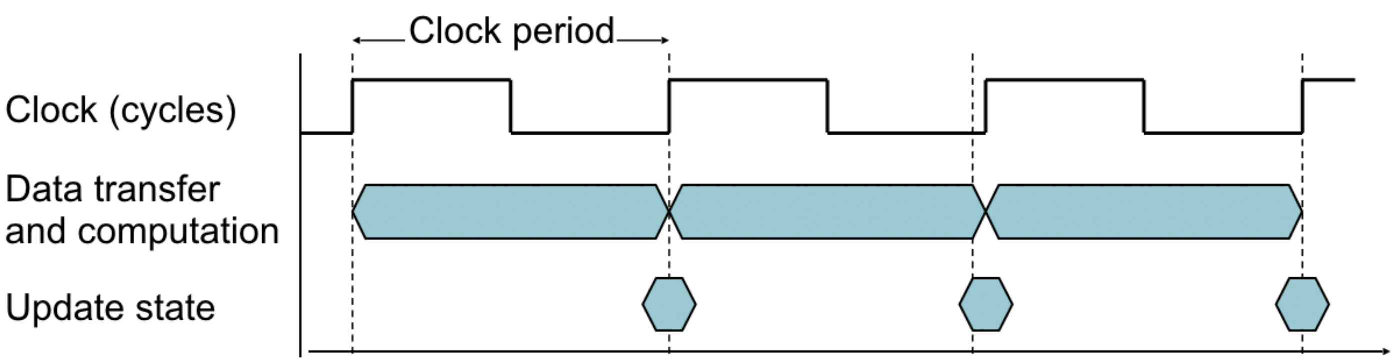
- Example: time taken to run a program
  - 10s on A, 15s on B
  - Execution Time$_B$ / Execution Time$_A$
    = 15s / 10s = 1.5
  - So A is 1.5 times faster than B

# Measuring Execution Time

- **Elapsed time ( Wall clock time or Response time)**
  - Total time to complete a task, including disk access, memory access, I/O activities, OS overhead.
  - Determines system performance
- **CPU execution time or CPU time**
  - The actual time that CPU computing for a specific task
- **CPU time= User CPU time** (the CPU time spent in a program itself) **+ System CPU time** (the CPU time spent in OS performing tasks on behalf of the program)
- We will use the term of System Performance to refer to elapsed time
- We will use the term of CPU Performance to refer to user CPU time

# CPU Clocking

- **We need a measure to show how fast the hardware can perform basic functions**
- All computers are constructed using  a clock that determines when events take place in the hardware.
- These discrete time intervals which run at a constant-rate are called **clock cycles, ticks, clock ticks, clock periods, clocks or cycles.**
- **Clock period is the length of each clock cycle.**
- **Clock frequency: 1/ clock period**




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

# CPU Execution Time for a program

**CPU execution time for a program=** CPU clock cycles for a program * clock cycle time

**CPU execution time for a program**= CPU clock cycles for a program/clock rate

- **Performance improved by**
  - Reducing number of clock cycles and clock cycle time (Increasing clock rate)
  - Many techniques that <u>decrease the number of clock cycles</u> may also <u>increase the clock cycle time</u>; Hardware designer must often trade off clock cycle time against clock cycle count

# CPU Execution Time- Example

- Computer A: 2GHz clock, run a program in 10s (CPU time)
- Designing Computer B
  - Aim for running the program in 6s (CPU time)
  - The designer has determined that a substantial increase in the clock rate is possible, but this increase will affect the rest of the CPU design, causing computer B to require 1.2 times as many clock cycles as computer A for this program
- Clock rate for Computer B ?

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

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

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

CPU time for B can be found using this equation:

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

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

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

# Instruction Count and CPI

- Previous performance equation didn't include the number of instruction needed for the program; execution time must depend on the number of instructions in a program

**CPU clock cycles= Instructions for a program* average clock cycles per instruction**

- Different instructions take different amounts of time, so we use average number of clock cycle per instruction (CPI) for a program

**CPU time= Instructions for a program* CPI* Clock cycle time**

**CPU time= (Instructions for a program* CPI)/ Clock rate**

# CPI Example

- **Computer A: Clock cycle Time = 250ps, CPI = 2.0 ;**
- **Computer B: Clock cycle Time = 500ps, CPI = 1.2; Same program**
- **Which computer is faster, and how much?**

We know that each computer executes the same number of instructions for the program; let's call this number $I$. First, find the number of processor clock cycles for each computer:

$$\text{CPU clock cycles}_A = I \times 2.0$$
$$\text{CPU clock cycles}_B = I \times 1.2$$

Now we can compute the CPU time for each computer:

$$\begin{aligned}\text{CPU time}_A &= \text{CPU clock cycles}_A \times \text{Clock cycle time} \\ &= I \times 2.0 \times 250 \text{ ps} = 500 \times I \text{ ps}\end{aligned}$$

Likewise, for B:

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

Clearly, computer A is faster. The amount faster is given by the ratio of the execution times:

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

We can conclude that computer A is 1.2 times as fast as computer B for this program.

# CPI in More Detail

- If different instruction classes take different numbers of cycles

$$\text{Clock Cycles} = \sum_{i=1}^{n} (\text{CPI}_i \times \text{Instruction Count}_i)$$

  - Weighted average CPI

$$\text{CPI} = \frac{\text{Clock Cycles}}{\text{Instruction Count}} = \sum_{i=1}^{n} \left( \text{CPI}_i \times \underbrace{\frac{\text{Instruction Count}_i}{\text{Instruction Count}}}_{\text{Relative frequency}} \right)$$

# CPI Example

- Alternative compiled code sequences using instructions in classes A, B, C

| Class | A | B | C |
|---|---|---|---|
| CPI for instruction class | 1 | 2 | 3 |
| Code sequence 1 | 2 | 1 | 2 |
| Code sequence 2 | 4 | 1 | 1 |

2+1+2=5 number of instructions  for code sequence 1

4+1+1=6 number of instructions for code sequence 2

# CPI Example

$$\text{Clock Cycles} = \sum_{i=1}^{n} (\text{CPI}_i \times \text{Instruction Count}_i)$$

$$\text{CPU clock cycles}_1 = (2 \times 1) + (1 \times 2) + (2 \times 3) = 2 + 2 + 6 = 10 \text{ cycles}$$

$$\text{CPU clock cycles}_2 = (4 \times 1) + (1 \times 2) + (1 \times 3) = 4 + 2 + 3 = 9 \text{ cycles}$$

So code sequence 2 is faster, even though it executes one extra instruction. Since code sequence 2 takes fewer overall clock cycles but has more instructions, it must have a lower CPI. The CPI values can be computed by

$$\text{CPI} = \frac{\text{CPU clock cycles}}{\text{Instruction count}}$$

$$\text{CPI}_1 = \frac{\text{CPU clock cycles}_1}{\text{Instruction count}_1} = \frac{10}{5} = 2.0$$

$$\text{CPI}_2 = \frac{\text{CPU clock cycles}_2}{\text{Instruction count}_2} = \frac{9}{6} = 1.5$$

# Performance Summary

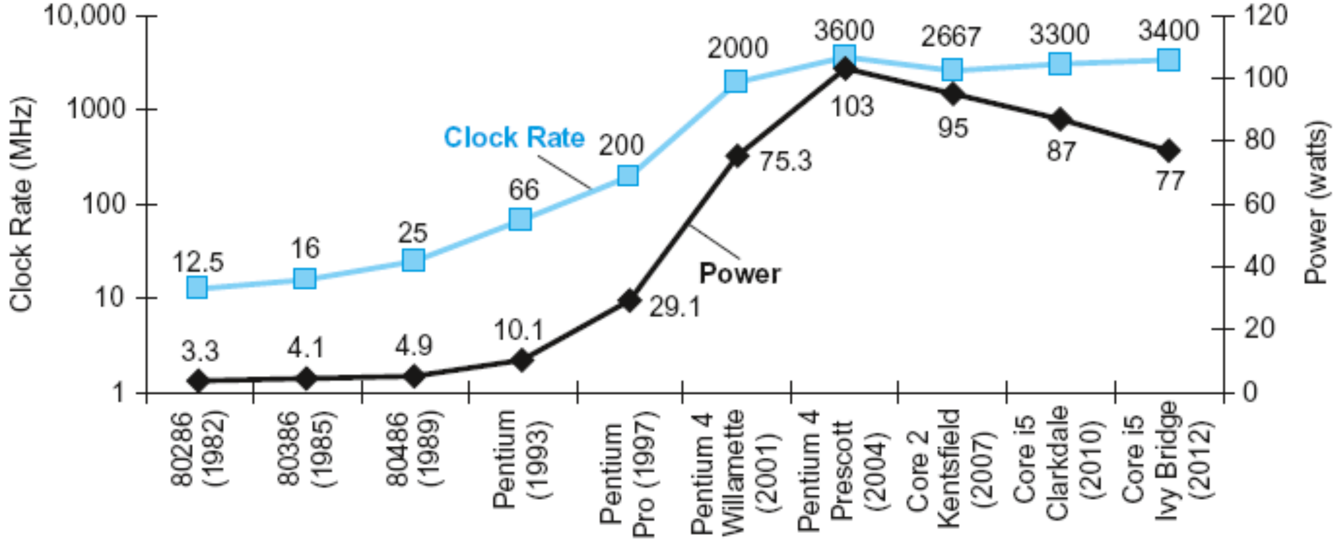
- **Measurements for cpu times at different levels;** these factors are combined to yield execution time measured in seconds per program

$$\text{CPU Time} = \frac{\text{Instructions}}{\text{Program}} \times \frac{\text{Clock cycles}}{\text{Instruction}} \times \frac{\text{Seconds}}{\text{Clock cycle}}$$

- Performance of a program depends on
  - **Algorithm:** affects <u>instruction count and possibly CPI</u> by choosing slower or faster instructions
  - **Programming language**: affects <u>instruction count</u> (statements in the PL are translated to processor instructions, which determine instruction count), and <u>CPI</u> (because of its feature; for example, abstraction in PL requires higher CPI)
  - **Compiler:** determines the translation of source instructions to the computer instructions, so compiler affects <u>instruction count and CPI</u>
  - **Instruction set architecture:** affects instruction count, CPI, clock rate

# Power Trends






- Increase in clock rate and power of eight generation of intel microprocessor over 30 years
- **Both clock rate and power increased rapidly for decades and flattened off recently**
- **They grow together as they are correlated**
- **The reason for their recent slowing that we have run into a particular <u>power limit for cooling microprocessors</u>**

# Power Trends

- **Dominant technology for integrated circuits is called CMOS** (complementary metal oxide semiconductor)
- For CMOS, the **primary source of energy consumption** is so called **dynamic energy** ( **energy that is consumed when transistors switch states from 0 to 1 and vice versa** )
- The dynamic energy **depends on** the capacitive loading of each transistor and the voltage applied
-

$$Energy \propto Capacitive\ load \times Voltage^2$$

This equation is the energy of a pulse during the logic transition of $0 \rightarrow 1 \rightarrow 0$ or $1 \rightarrow 0 \rightarrow 1$. The energy of a single transition is then

$$Energy \propto 1/2 \times Capacitive\ load \times Voltage^2$$

The power required per transistor is just the product of energy of a transition and the frequency of transitions:

$$Power \propto 1/2 \times Capacitive\ load \times Voltage^2 \times Frequency\ switched$$

# Power Trends- Example

Suppose we developed a new, **simpler processor that has 85% of the capacitive load of the more complex older processor**. **Further,** assume that it has adjustable voltage so that it can reduce voltage 15% compared to old processor, which results in a 15% shrink in frequency. What is the impact on dynamic power?

$$\frac{\text{Power}_{\text{new}}}{\text{Power}_{\text{old}}} = \frac{\langle \text{Capacitive load} \times 0.85\rangle \times \langle \text{Voltage} \times 0.85\rangle^2 \times \langle \text{Frequency switched} \times 0.85\rangle}{\text{Capacitive load} \times \text{Voltage}^2 \times \text{Frequency switched}}$$

Thus the power ratio is

$$0.85^4 = 0.52$$

Hence, the new processor uses about half the power of the old processor.

- **To reduced power consumption:** **Designers have attached device to increase cooling and turn off the parts of the chip that are not used in given cycle**

# Reading assignment

Read 1.6 and 1.7 of the text book