

# Chapter 1

Computer Abstractions and Technology

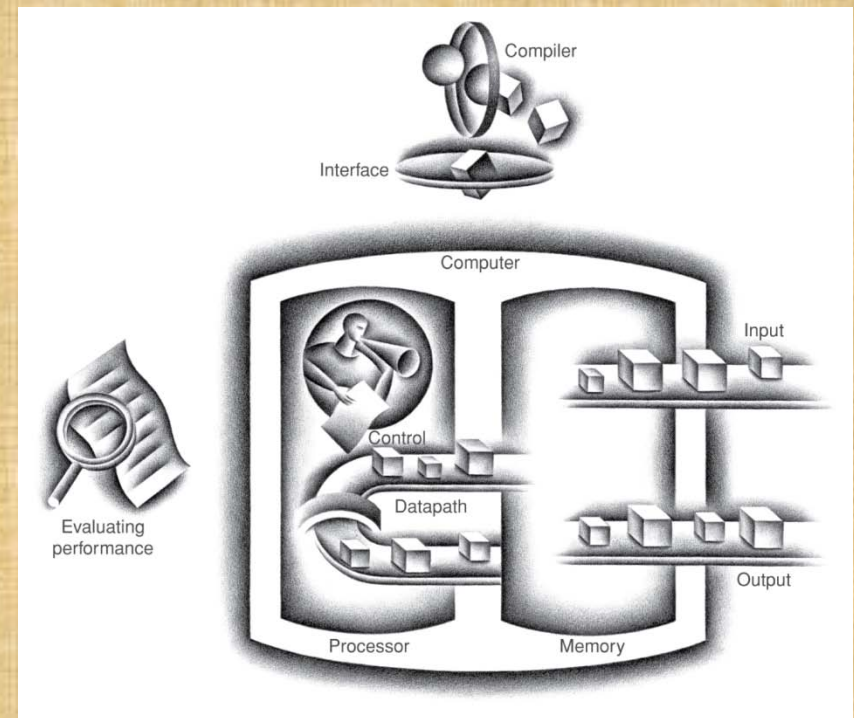
Lesson 2: **Understanding Performance**

Indeed, the cost-performance ratio of the product will depend most heavily on the implementer, just as ease of use depends most heavily on the architect.

*The Mythical Man-Month*, Brooks, pg 46

# So far...

- ◆ Progress in computer technology has been revolutionary
- ◆ The Computer Components
  - I/O
    - Mouse, Monitor, Printer, etc.
  - Processor
  - Memory
    - Volatile main memory
    - Non Volatile secondary memory





# Performance Metrics

## ◆ Purchasing perspective

- given a collection of machines, which has the
  - best performance?
  - least cost?
  - best cost/performance?

## ◆ Design perspective

- faced with design options, which has the
  - best performance improvement?
  - least cost?
  - best cost/performance?

## ◆ Both require

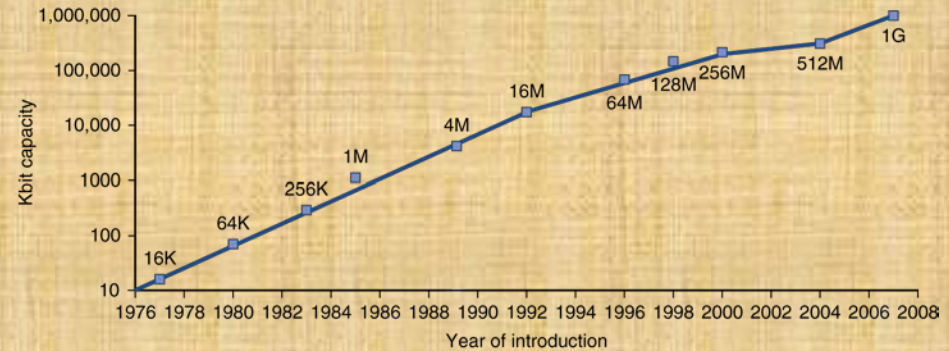
- basis for comparison
- metric for evaluation

## ◆ Our goal is to understand what factors in the architecture contribute to overall system performance and the relative importance (and cost) of these factors

# Technology Trends

◆ Electronics technology continues to evolve

- Increased capacity and performance
- Reduced cost

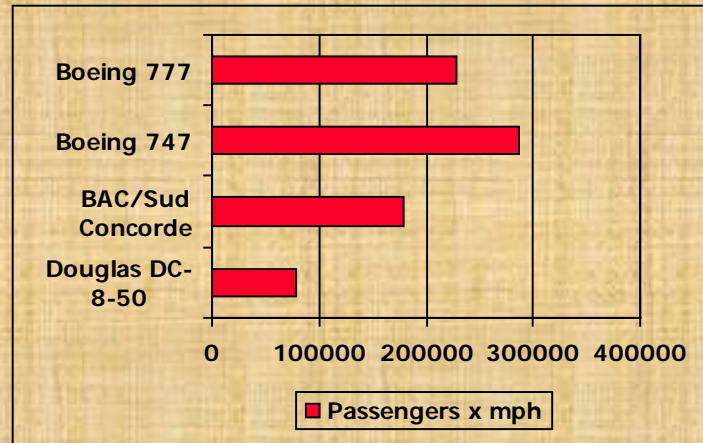
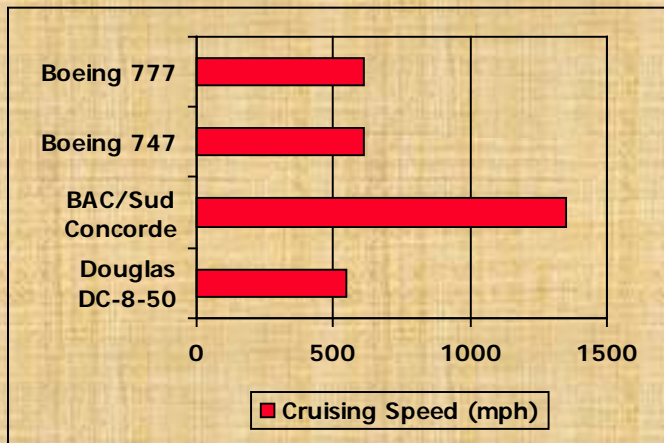
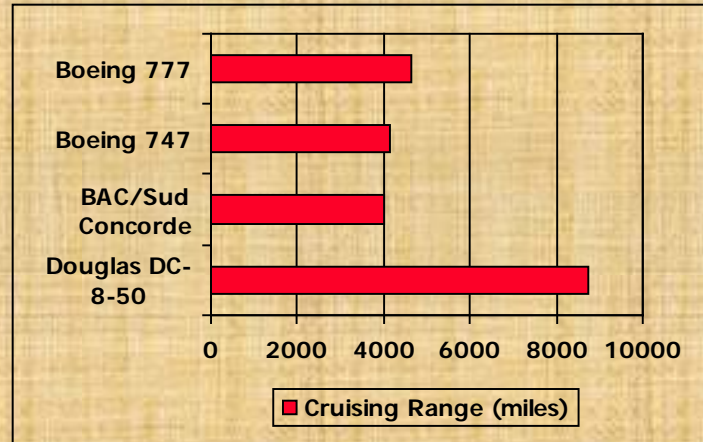
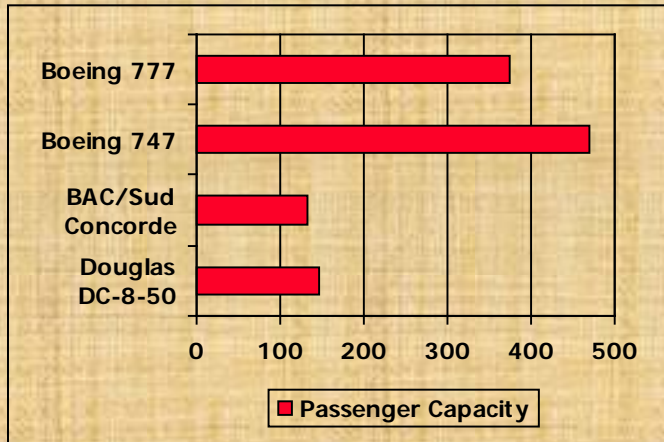


DRAM capacity

Year	Technology	Relative performance/cost
1951	Vacuum tube	1
1965	Transistor	35
1975	Integrated circuit (IC)	900
1995	Very large scale IC (VLSI)	2,400,000
2005	Ultra large scale IC	6,200,000,000

# Defining Performance

- ◆ Which airplane has the best performance?



# Response Time and Throughput

- ◆ **Response time** (aka **execution time**)
  - How long it takes to do a task
  - Important to individual users
- ◆ **Throughput**
  - Total work done per unit time
    - e.g., tasks/transactions/... per hour
    - Important to data center managers
- ◆ 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  $n$  time faster than Y”

$$\begin{aligned} & \text{Performance}_X / \text{Performance}_Y \\ &= \text{Execution time}_Y / \text{Execution time}_X = n \end{aligned}$$

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

# Measuring Execution Time

## ◆ Elapsed time

- Total response time, including all aspects
  - Processing, I/O, OS overhead, idle time
  - a useful number, but often not good for comparison purposes
- Determines system performance

## ◆ CPU time

- the time spent by the CPU to complete his task and doesn't include I/O or time spent running other programs
- Comprises user CPU time and system CPU time
  - **system time** - time spent in OS performing tasks on behalf of a program;and
  - **user time** - time spent by the CPU in a program itself

- ◆ Different programs are affected differently by CPU and system performance

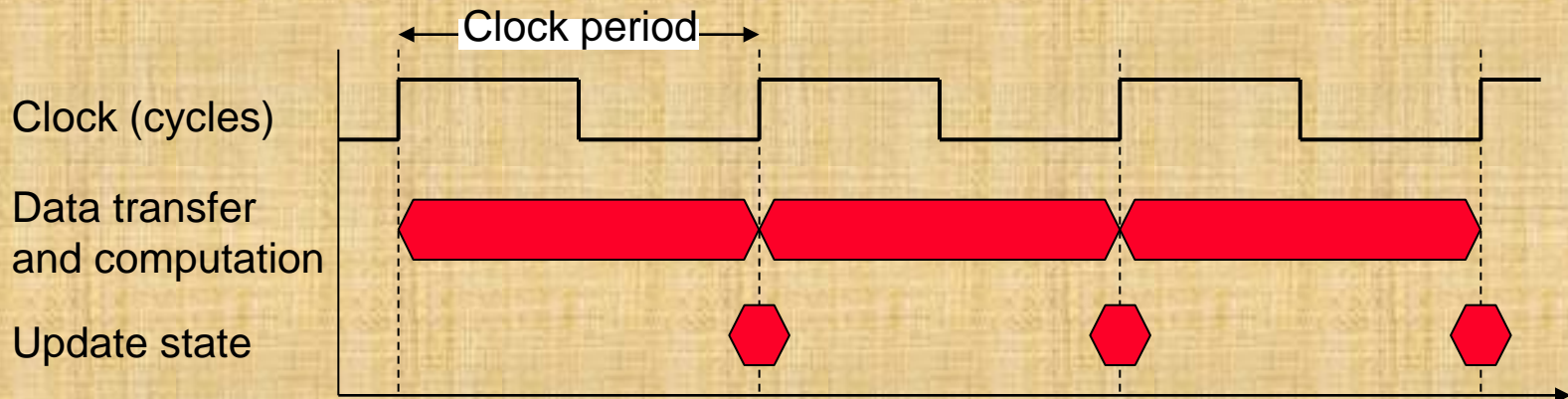
# Metric Units

Exp.	Explicit	Prefix	Exp.	Explicit	Prefix
$10^{-3}$	0.001	milli	$10^3$	1,000	Kilo
$10^{-6}$	0.000001	micro	$10^6$	1,000,000	Mega
$10^{-9}$	0.000000001	nano	$10^9$	1,000,000,000	Giga
$10^{-12}$	0.000000000001	pico	$10^{12}$	1,000,000,000,000	Tera
$10^{-15}$	0.000000000000001	femto	$10^{15}$	1,000,000,000,000,000	Peta
$10^{-18}$	0.000000000000000001	atto	$10^{18}$	1,000,000,000,000,000,000	Exa
$10^{-21}$	0.000000000000000000001	zepto	$10^{21}$	1,000,000,000,000,000,000,000	Zetta
$10^{-24}$	0.000000000000000000000001	yocto	$10^{24}$	1,000,000,000,000,000,000,000,000	Yotta

- ◆ The principal metric prefixes.

# CPU Clocking

- ◆ Operation of digital hardware governed by a constant-rate clock



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

# CPU Time

$$\begin{aligned}\text{CPU Time} &= \text{CPU Clock Cycles} \times \text{Clock Cycle Time} \\ &= \frac{\text{CPU Clock Cycles}}{\text{Clock Rate}}\end{aligned}$$

- ◆ Performance improved by
  - Reducing number 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
  - Making faster clock causes an increase of  $1.2 \times$  clock cycles wrt A
- ◆ How fast must Computer B clock be?

$$\text{Clock Rate}_B = \frac{\text{Clock Cycles}_B}{\text{CPU Time}_B} = \frac{1.2 \times \text{Clock Cycles}_A}{6\text{s}}$$

$$\begin{aligned}\text{Clock Cycles}_A &= \text{CPU Time}_A \times \text{Clock Rate}_A \\ &= 10\text{s} \times 2\text{GHz} = 20 \times 10^9\end{aligned}$$

$$\text{Clock Rate}_B = \frac{1.2 \times 20 \times 10^9}{6\text{s}} = \frac{24 \times 10^9}{6\text{s}} = 4\text{GHz}$$

# Instruction Count and Cycles Per Instruction (CPI)

Clock Cycles = Instruction Count  $\times$  Cycles Per Instruction

CPU Time = Instruction Count  $\times$  CPI  $\times$  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 affected by instruction mix

## CPI Example

- ◆ Computer A: Cycle Time = 250ps, CPI = 2.0
- ◆ Computer B: Cycle Time = 500ps, CPI = 1.2
- ◆ Same ISA (Instruction Set Architecture)
- ◆ Which is faster, and by how much?

$$\begin{aligned}\text{CPU Time}_A &= \text{Instruction Count} \times \text{CPI}_A \times \text{Cycle Time}_A \\ &= 1 \times 2.0 \times 250\text{ps} = 1 \times 500\text{ps}\end{aligned}$$

A is faster...

$$\begin{aligned}\text{CPU Time}_B &= \text{Instruction Count} \times \text{CPI}_B \times \text{Cycle Time}_B \\ &= 1 \times 1.2 \times 500\text{ps} = 1 \times 600\text{ps}\end{aligned}$$

$$\frac{\text{CPU Time}_B}{\text{CPU Time}_A} = \frac{1 \times 600\text{ps}}{1 \times 500\text{ps}} = 1.2$$

...by this much

## 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

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

Relative frequency

# CPI Example

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

Class	A	B	C
CPI for class	1	2	3
IC in sequence 1	2	1	2
IC in sequence 2	4	1	1

- Sequence 1: IC = 5

- Clock Cycles  
 $= 2 \times 1 + 1 \times 2 + 2 \times 3$   
 $= 10$
- Avg. CPI =  $10/5 = 2.0$

- Sequence 2: IC = 6

- Clock Cycles  
 $= 4 \times 1 + 1 \times 2 + 1 \times 3$   
 $= 9$
- Avg. CPI =  $9/6 = 1.5$

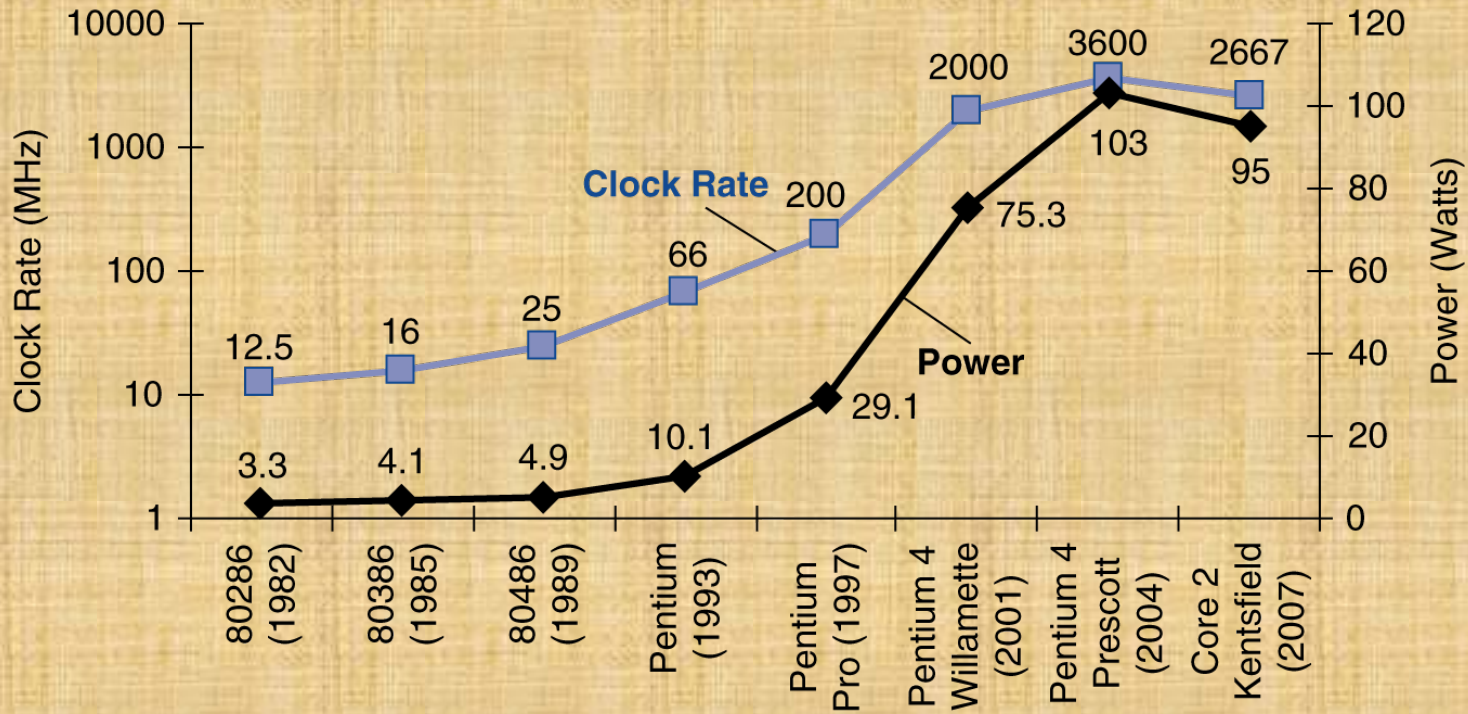
# Performance Summary

## The BIG Picture

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

- ◆ Performance depends on
  - ❑ Algorithm: affects IC, possibly CPI
  - ❑ Programming language: affects IC, CPI
  - ❑ Compiler: affects IC, CPI
  - ❑ Instruction set architecture: affects IC, CPI,  $T_c$

# Power Trends



- ◆ In CMOS Integrated Circuit (IC) technology
  - CMOS (Complementary Metal Oxide Semiconductor)

Dynamic Power Dissipation = Capacitive load of each transistor × Voltage<sup>2</sup> × Frequency Transistor Switched



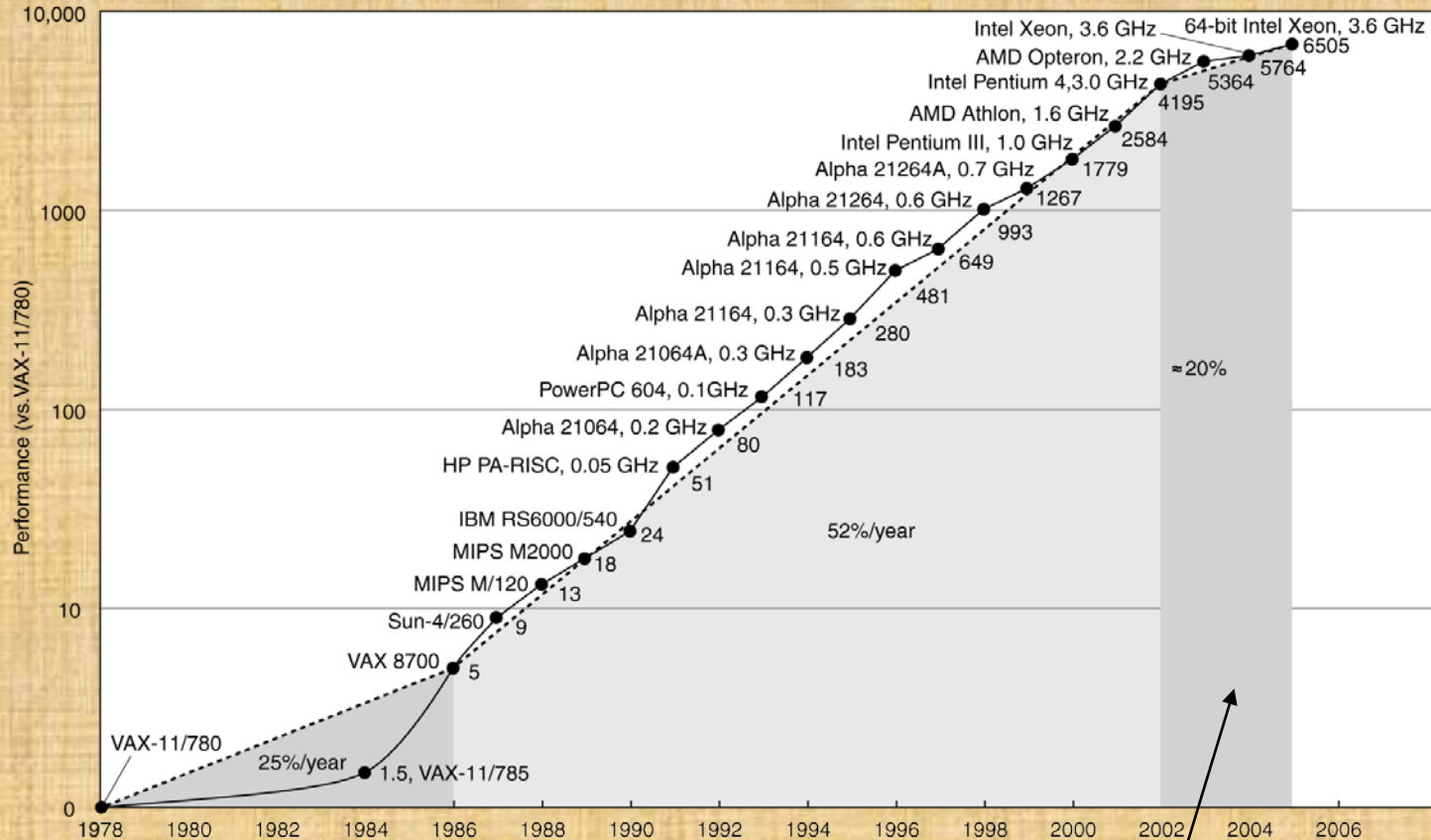
# Reducing Power

- ◆ Suppose a new CPU has
  - 85% of capacitive load of old CPU
  - 15% voltage and 15% frequency reduction

$$\frac{P_{\text{new}}}{P_{\text{old}}} = \frac{C_{\text{old}} \times 0.85 \times (V_{\text{old}} \times 0.85)^2 \times F_{\text{old}} \times 0.85}{C_{\text{old}} \times V_{\text{old}}^2 \times F_{\text{old}}} = 0.85^4 = 0.52$$

- The power wall
  - We can't reduce voltage further
  - We can't remove more heat
- How else can we improve performance?

# Uniprocessor Performance



Constrained by power, instruction-level parallelism, memory latency

# Multiprocessors

- ◆ Multicore microprocessors
  - More than one processor per chip
- ◆ Requires explicitly parallel programming
  - Compare with instruction level parallelism
    - Hardware executes multiple instructions at once
    - Hidden from the programmer
  - Hard to do
    - Programming for performance
    - Load balancing
    - Optimizing communication and synchronization

# Concluding Remarks

- ◆ Cost/performance is improving
  - Due to underlying technology development
- ◆ Hierarchical layers of abstraction
  - In both hardware and software
- ◆ Instruction set architecture
  - The hardware/software interface
- ◆ Execution time: the best performance measure
- ◆ Power is a limiting factor
  - Use parallelism to improve performance

# Next Lecture and Reminders

## ◆ Next lecture

- ❑ MIPS as millions of Instructions per second
- ❑ Amdahl's Law
- ❑ MIPS-32 and 64 instruction Set
  - Reading assignment – PH, Chapter 1

## ◆ Reminders

- ❑ HW1: Deadline – due Sept 8