UNIVERSITÉ Concordia UNIVERSITY
GINA CODY
SCHOOL OF ENGINEERING
AND COMPUTER SCIENCE

Chapter 1

Part II

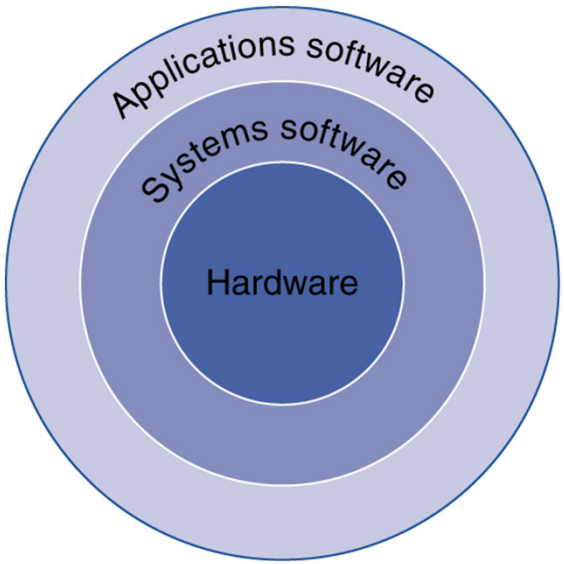
Assessing and Understanding Performance

Dr. Fadi Alzhouri
COEN: 316

[Based on Figures from *Computer Organization and Design*: *The Hardware/Software Interface* Patterson & Hennessy, 5th ed. © 2014 Elsevier Inc.
&
Computer Organization and Architecture: *Designing for Performance* William Stallings, 8th ed. © 2010 Pearson Education Inc.]

Department of Electrical & Computer Engineering (ECE)

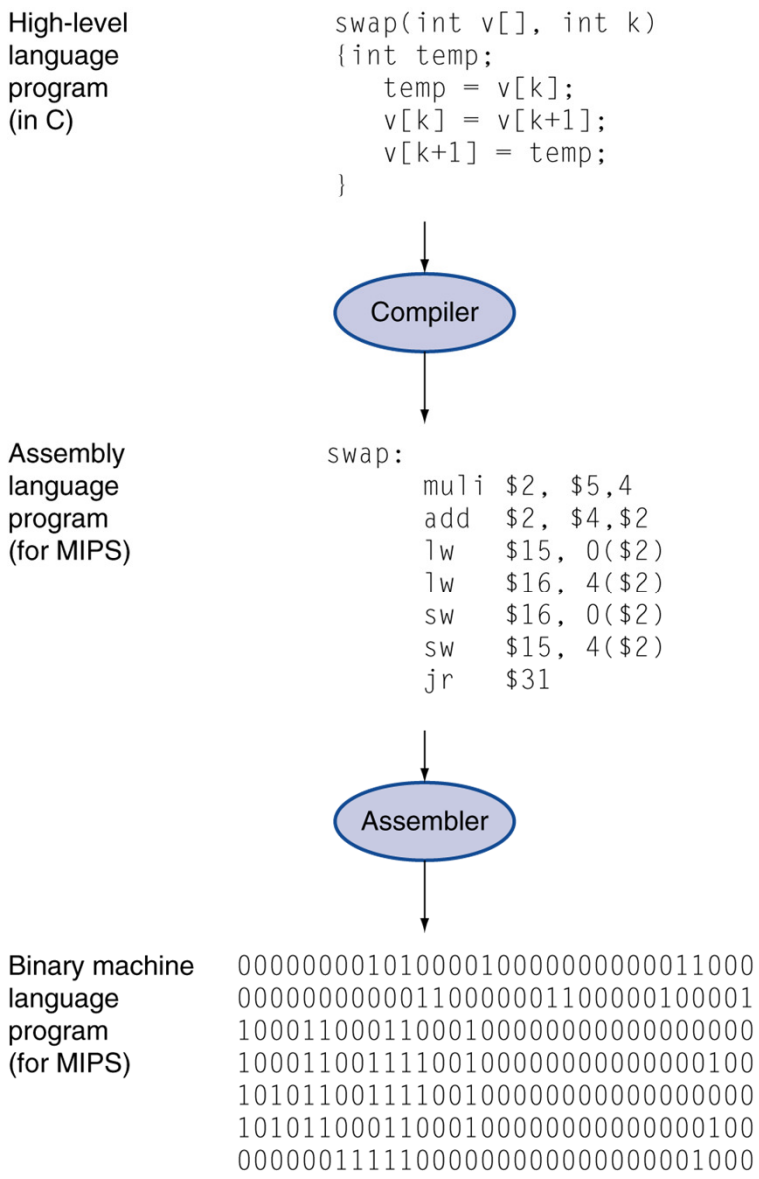
Below Your Program



- Application software
  - Written in high-level language
- System software
  - Compiler: translates HLL code to machine code
  - Operating System: service code
    - Handling input/output
    - Managing memory and storage
    - Scheduling tasks & sharing resources
- Hardware
  - Processor, memory, I/O controllers

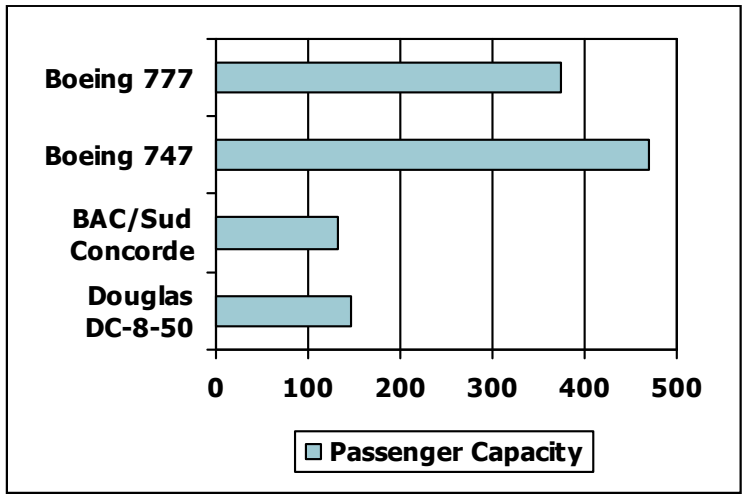
Levels of Program Code

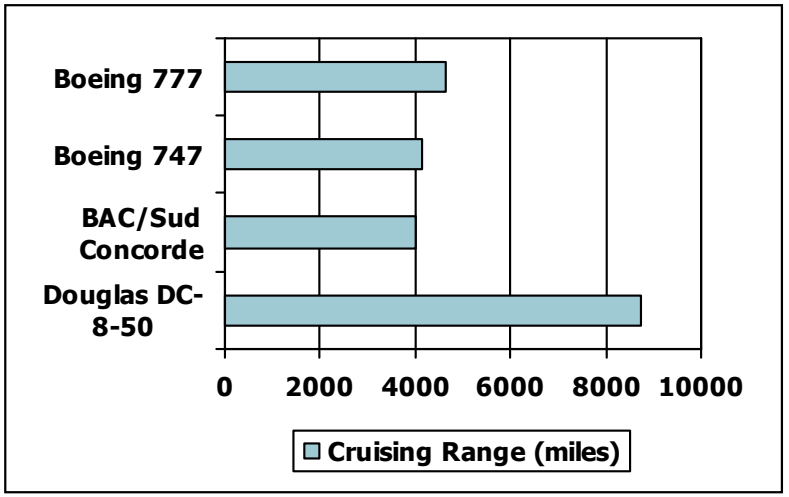
- High-level language
  - Level of abstraction closer to problem domain
  - Provides for productivity and portability
- Assembly language
  - Textual representation of instructions
- Hardware representation
  - Binary digits (bits)
  - Encoded instructions and data

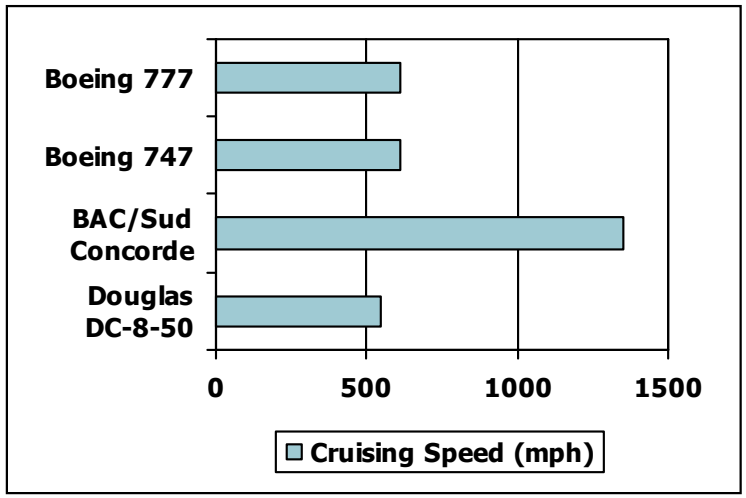


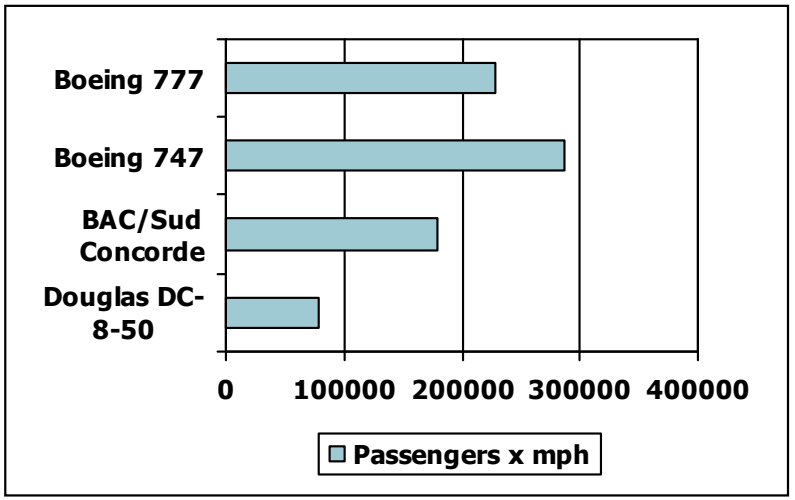
Defining Performance

- Which airplane has the best performance?









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)
  - Determines how fast I/O operations are executed

Response Time and Throughput

- 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?
- Focus on response time for now…

Relative Performance

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

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

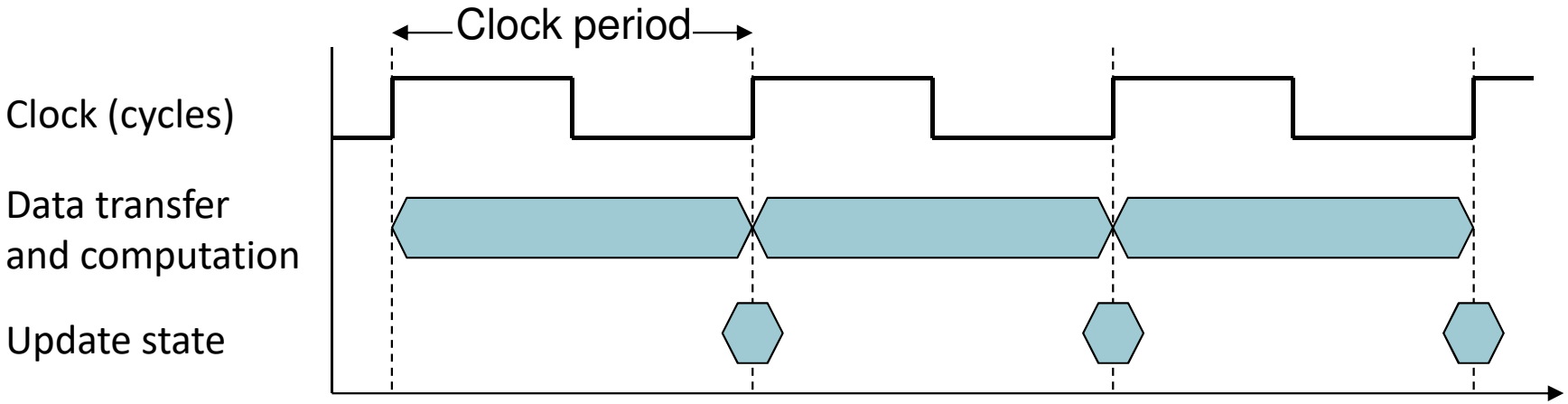
- E.g., Time taken to run a program
  - 10s on A, 15s on B
  - $\text{Execution Time}_B / \text{Execution Time}_A$ = 15s/10s = 1.5
  - A is 1.5 times faster than B

Measuring Execution Time

- Elapsed time (wall clock time, response time)
  - Total response time, including all aspects
    - Processing, I/O, OS overhead, idle time
  - Determines system performance
  - Hard to measure
- CPU time
  - Time spent processing a given job
    - Discounts I/O time, other jobs' shares
  - Comprises user CPU time and system CPU time
  - Different programs are affected differently by CPU and system performance

CPU Clocking

- Operation of digital hardware governed by a constant-rate clock (discrete time interval)



  - Clock period: duration of a clock cycle
    - E.g., 250ps = 0.25ns = 250×10^{-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 improved by
  - Reducing number of clock cycles
  - Increasing clock rate (reducing length of clock cycles)
  - Hardware designer must often trade off clock rate against cycle count

CPU Time: Example

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

CPU Time: Example

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

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

CPU Time: Example

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

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

CPU Time: Example

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

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

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

$$= 10\text{s} \times 2\text{GHz} = 20 \times 10^9$$

CPU Time: Example

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

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

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

$$= 10\text{s} \times 2\text{GHz} = 20 \times 10^9$$

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

Instruction Count and 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 (CPI)
  - Determined by CPU hardware
  - If different instructions have different CPI
    - Average CPI affected by instruction mix

CPI: Example 1

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

CPI: Example 1

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

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

CPI: Example 1

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

$$\begin{aligned}\text{CPU Time}_A &= \text{Instruction Count} \times \text{CPI}_A \times \text{Cycle Time}_A \\ &= I \times 2.0 \times 250\text{ps} = I \times 500\text{ps} \\ \text{CPU Time}_B &= \text{Instruction Count} \times \text{CPI}_B \times \text{Cycle Time}_B \\ &= I \times 1.2 \times 500\text{ps} = I \times 600\text{ps}\end{aligned}$$

CPI: Example 1

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

$$\begin{aligned}
\text{CPU Time}_A &= \text{Instruction Count} \times \text{CPI}_A \times \text{Cycle Time}_A \\
&= I \times 2.0 \times 250\text{ps} = I \times 500\text{ps} \\
\text{CPU Time}_B &= \text{Instruction Count} \times \text{CPI}_B \times \text{Cycle Time}_B \\
&= I \times 1.2 \times 500\text{ps} = I \times 600\text{ps} \\
\frac{\text{CPU Time}_B}{\text{CPU Time}_A} &= \frac{I \times 600\text{ps}}{I \times 500\text{ps}} = 1.2
\end{aligned}$$

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 2

- 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 = ??
  - Avg. CPI = ??
- Sequence 2: IC = 6
  - Clock Cycles = ??
  - Avg. CPI = ??

CPI: Example 2

- 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
    = 2×1 + 1×2 + 2×3 = 10
  - Avg. CPI = 10/5 = 2.0
- Sequence 2: IC = 6
  - Clock Cycles
    = 4×1 + 1×2 + 1×3 = 9
  - Avg. CPI = 9/6 = 1.5

CPI: Example 3

- Computer A has an overall CPI of 1.3 and can be run at a clock rate of 600MHz. Computer B has a CPI of 2.5 and can be run at a clock rate of 750 MHz. We have a particular program we wish to run. When compiled for computer A, this program has exactly 100,000 instructions. How many instructions would the program need to have when compiled for Computer B, in order for the two computers to have exactly the same execution time for this program?
- Solution
  - $\text{CPU time}_A = (\text{Instruction count})_A \times (\text{CPI})_A \times (\text{Clock cycle Time})_A = (100{,}000) \times (1.3)/(600 \times 10^6)$
  - $\text{CPU time}_B = (\text{Instruction count})_B \times (\text{CPI})_B \times (\text{Clock cycle Time})_B = (I)_B \times (2.5)/(750 \times 10^6)$
  - Since $\text{CPU time}_A = \text{CPU time}_B$
    - We have to solve for $(I)_B$ and get 65000

CPI: Example 4

- Given an instruction mix of a program on a RISC processor

$Class_i$	$Freq_i$	CPI_i
ALU	50%	1
Load	20%	5
Store	10%	3
Branch	20%	2

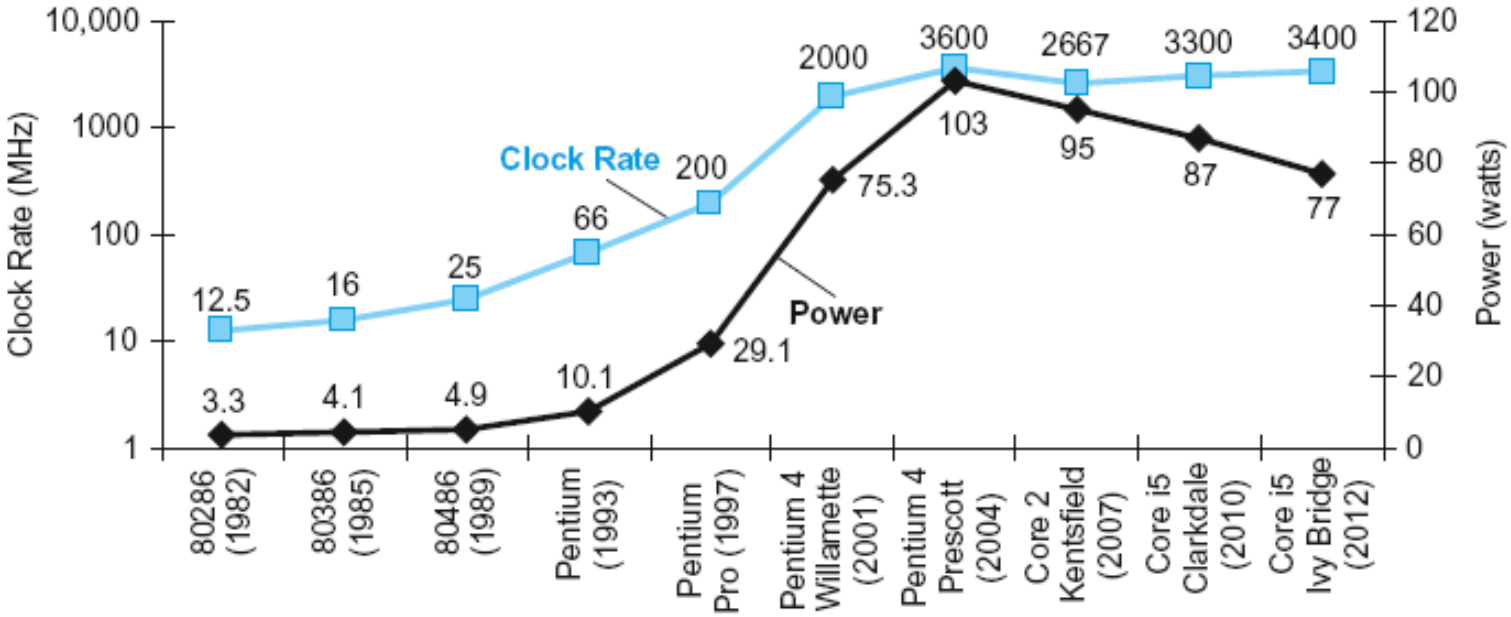
$$CPI = \sum_{i=1}^{n} (CPI_i \times Freq_i)$$

Performance Summary

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

- 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 IC technology

$$\text{Power} = \text{Capacitive load} \times \text{Voltage}^2 \times \text{Frequency}$$

×30

5V → 1V

×1000

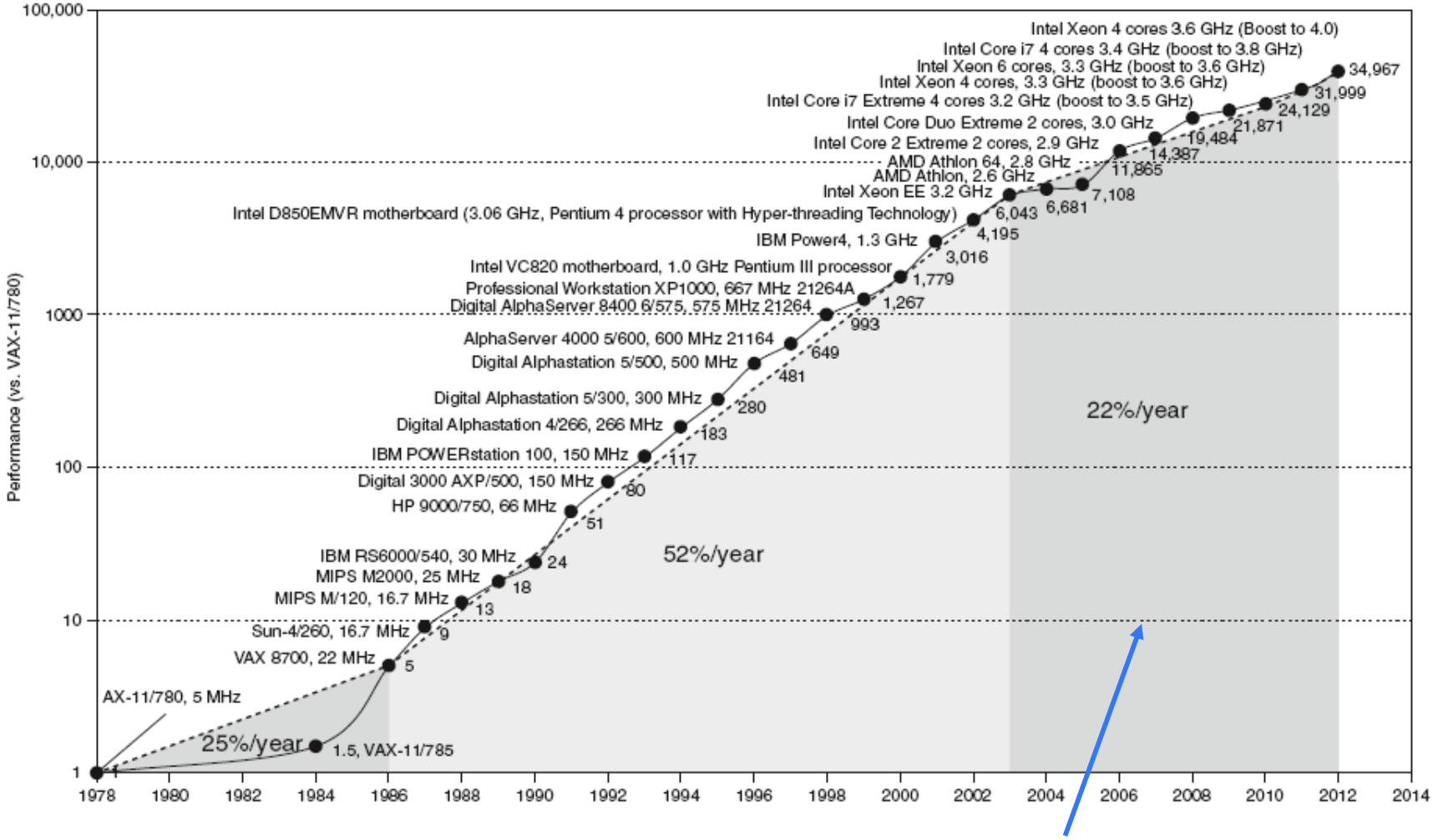
Reducing Power

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

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

- 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

Power-Workload Relationship

- Look back at i7 power benchmark
  - At 100% load: 258W
  - At 50% load: 170W (66%)
  - At 10% load: 121W (47%)
- Google data center
  - Mostly operates at 10% – 50% load
  - At 100% load less than 1% of the time
- Consider designing processors to make power proportional to workload

SPEC CPU Benchmark

- Programs used to measure performance
  - Supposedly typical of actual workload
- Standard Performance Evaluation Corp (SPEC)
  - Develops benchmarks for CPU, I/O, Web, …
- SPEC CPU2006
  - Elapsed time to execute a selection of programs
    - Negligible I/O, so focuses on CPU performance
  - Normalize relative to reference machine
  - Summarize as geometric mean of performance ratios
    - CINT2006 (integer) and CFP2006 (floating-point)

$$\sqrt[n]{\prod_{i=1}^{n} \text{Execution time ratio}_i}$$

CINT2006 for Intel Core i7 920

Description	Name	Instruction Count x 10^9	CPI	Clock cycle time (seconds x 10^{-9})	Execution Time (seconds)	Reference Time (seconds)	SPECratio
Interpreted string processing	perl	2252	0.60	0.376	508	9770	19.2
Block-sorting compression	bzip2	2390	0.70	0.376	629	9650	15.4
GNU C compiler	gcc	794	1.20	0.376	358	8050	22.5
Combinatorial optimization	mcf	221	2.66	0.376	221	9120	41.2
Go game (AI)	go	1274	1.10	0.376	527	10490	19.9
Search gene sequence	hmmer	2616	0.60	0.376	590	9330	15.8
Chess game (AI)	sjeng	1948	0.80	0.376	586	12100	20.7
Quantum computer simulation	libquantum	659	0.44	0.376	109	20720	190.0
Video compression	h264avc	3793	0.50	0.376	713	22130	31.0
Discrete event simulation library	omnetpp	367	2.10	0.376	290	6250	21.5
Games/path finding	astar	1250	1.00	0.376	470	7020	14.9
XML parsing	xalancbmk	1045	0.70	0.376	275	6900	25.1
Geometric mean	–	–	–	–	–	–	25.7

SPEC Power Benchmark

- Power is becoming important
  - Cost vs heat vs cooling vs damage
- Power consumption of server at different workload levels
  - Performance: ssj_ops/sec
  - Power: Watts (Joules/sec)

$$\text{Overall ssj\_ops per Watt} = \left(\sum_{i=0}^{10} \text{ssj\_ops}_i\right) \Big/ \left(\sum_{i=0}^{10} \text{power}_i\right)$$

SPECpower_ssj2008 for Xeon X5650

Target Load %	Performance (ssj_ops)	Average Power (Watts)
100%	865,618	258
90%	786,688	242
80%	698,051	224
70%	607,826	204
60%	521,391	185
50%	436,757	170
40%	345,919	157
30%	262,071	146
20%	176,061	135
10%	86,784	121
0%	0	80
Overall Sum	4,787,166	1,922
Σssj_ops/Σpower =		2,490

Pitfall: Amdahl's Law

- Improving an aspect of a computer and expecting a proportional improvement in overall performance

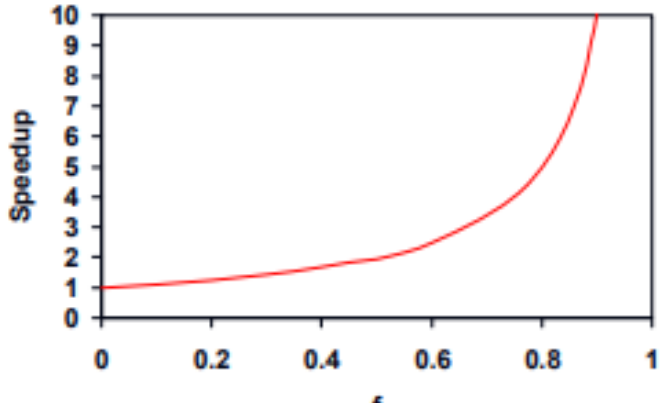
$$T_{improved} = \frac{T_{affected}}{improvement\ factor} + T_{unaffected}$$

- Corollary
  - Make the common case fast

Pitfall: Amdahl's Law

- Let f=fraction sped up and s = speedup amount for f
  - New_time = (1-f) x old_time + (f/s) x old_time
  - System Speedup = old_time / new_time
  - System Speedup = old_time / ((1-f) x old_time + (f/s) x old_time)
- Amdahl's Law:

$$Speedup = \frac{1}{1 - f + \frac{f}{s}}$$



- Your boss asks you to improve performance by either:
  - Improve the ALU used 95% of time by 10%
  - Improve memory pipeline used 5% of time by 10x
  - Which one do you choose?

Pitfall: Amdahl's Law

- Solution to the previous problem:
  - Calculate the speed up in both cases
  - Choose the one with the higher speed up

95%	1.10	1.094
5%	10	1.047
5%	∞	1.052

Pitfall: Amdahl's Law

- Suppose a program runs in 100 seconds on a machine, with multiply responsible for 80 seconds of this time. How much do we have to improve the speed of multiplication if we want the program to run 4 times faster?

- How about making the program 5 times faster?

MIPS as a Performance Metric

- MIPS: Millions of Instructions Per Second
  - Faster machine means larger MIPS
  - Doesn't account for
    - Differences in ISAs between computers
    - Differences in complexity between instructions

$$\text{MIPS} = \frac{\text{Instruction count}}{\text{Execution time} \times 10^6} = \frac{\text{Instruction count}}{\frac{\text{Instruction count} \times \text{CPI}}{\text{Clock rate}} \times 10^6} = \frac{\text{Clock rate}}{\text{CPI} \times 10^6}$$

  - CPI varies between programs on a given CPU

MIPS Example

- Two different compilers are being tested on the same program for a 4 GHz machine with three different classes of instructions: Class A, Class B, and Class C, which require 1, 2, and 3 cycles, respectively.
- The instruction count produced by the first compiler is 5 billion Class A instructions, 1 billion Class B instructions, and 1 billion Class C instructions.
- The second compiler produces 10 billion Class A instructions, 1 billion Class B instructions, and 1 billion Class C instructions.
- Which compiler produces a higher MIPS?
- Which compiler produces a better execution time?

Summary

- **Execution time**
  - Best performance measure
- **Power**
  - Limiting factor
  - Use parallelism to improve performance
- Tutorial problems
  - 1.7 page 56
  - 1.10 page 57
  - 1.14 page 59