معماری کامپیوتر

فصل چهار

کارایی

Computer Architecture

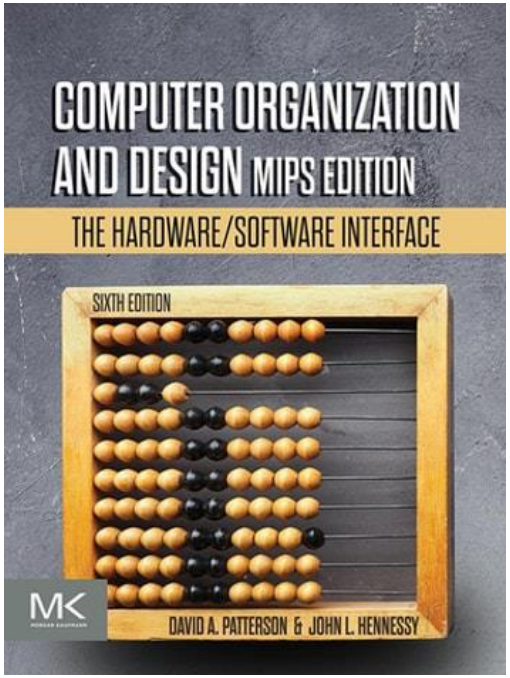
Chapter Four

Performance

Copyright Notice

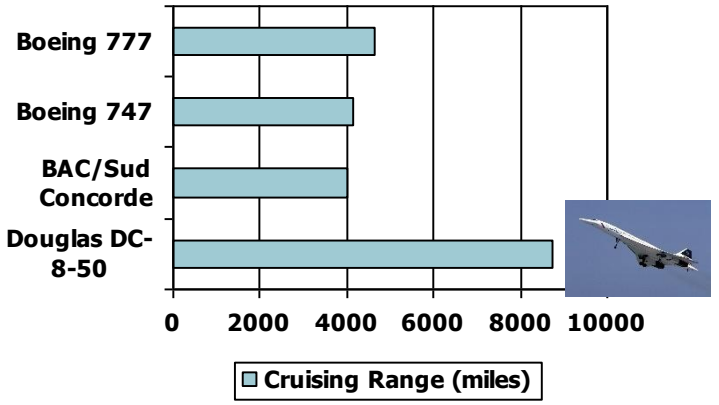
The slides of this lecture are adopted from:

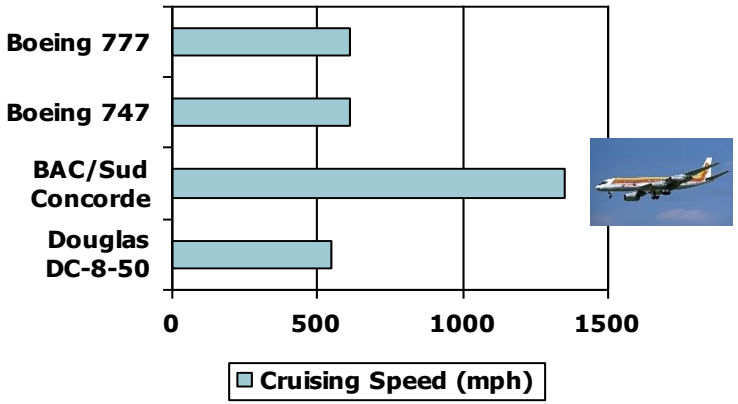
- D. Patterson & J. Hennessey, “Computer Organization & Design, The Hardware/Software Interface”, 6th Ed., MK publishing, 2020

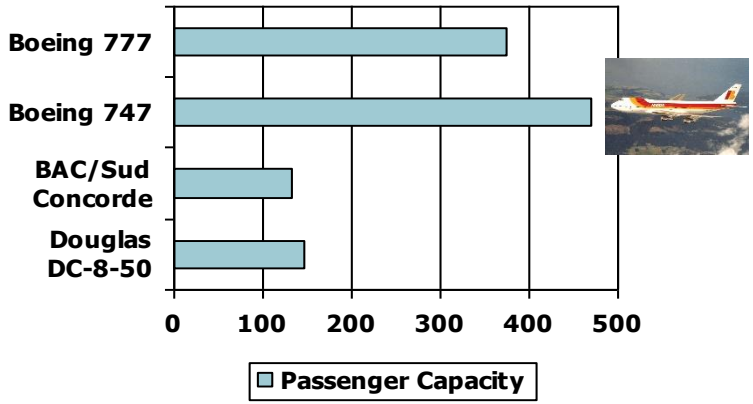


Defining Performance

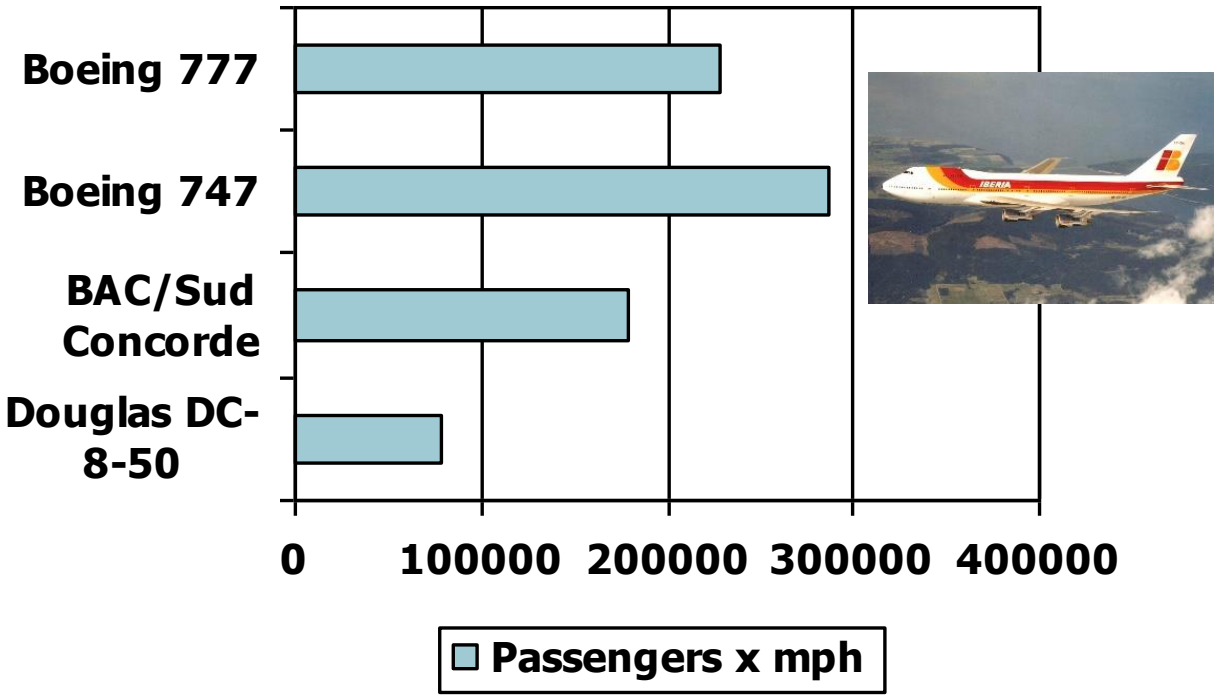
Which airplane has the best performance?







Passengers' Transports Rate



Computer Performance Measures

- Response time
  - How long it takes to do a task

- Throughput
  - Total work done per unit time
    - e.g., tasks/transactions/… per hour

- Decreasing response time almost always improves throughput

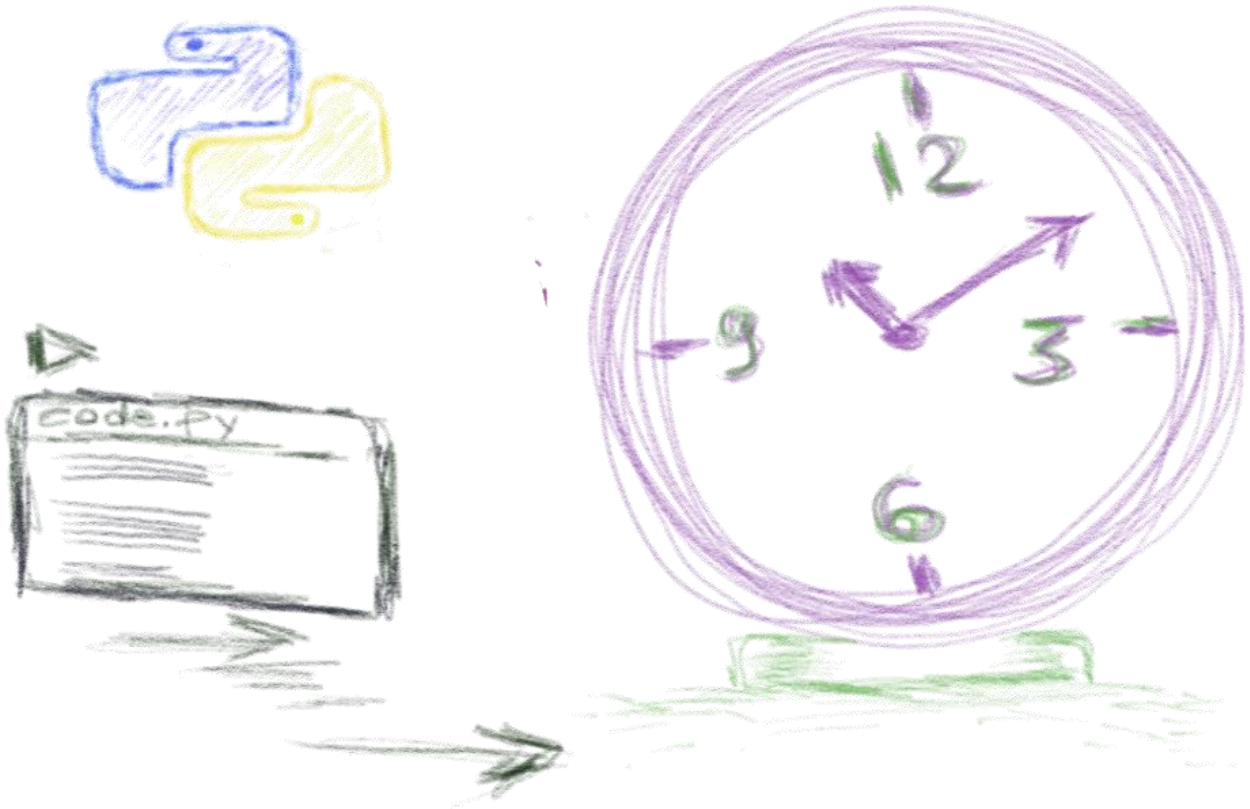
Response Time vs. Throughput

- How are response time & throughput affected by
  - Replacing a processor with a faster one?
  - Adding more processors?

- We'll focus on response time (execution time) for now

Performance

Performance = 1 / Execution time



Relative Performance

$$\frac{Performance_x}{Performance_y} = \frac{Execution\_time_y}{Execution\_time_x} = n$$

Example

- Time taken to run a program
  - 10s on A, 15s on B

$$\frac{Execution\_time_B}{Execution\_time_A} = \frac{15s}{10s} = 1.5$$

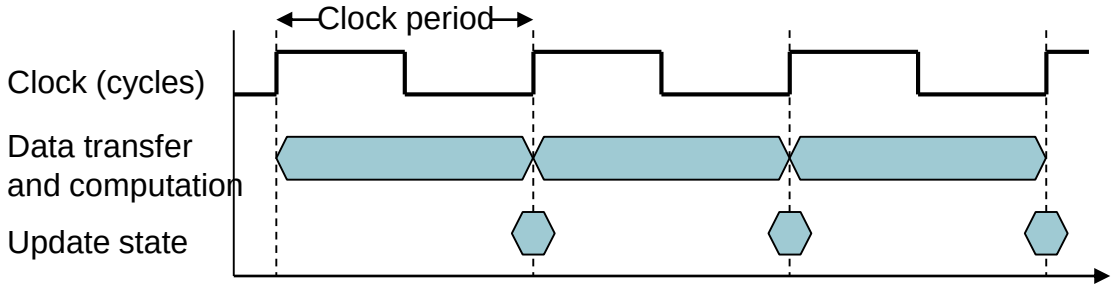
- So A is 1.5 times faster than B

Measuring Execution Time

- Elapsed time
  - Total response time, including all aspects
    - Processing, I/O, OS overhead, idle time
  - Determines system performance
- CPU time
  - Time spent processing a given job
    - Discounts I/O time, other jobs' shares
  - Comprises user CPU time & system CPU time

CPU Clocking

- Operation of digital hardware governed by a clock



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

CPU Time

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

- 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
  - Can do faster clock, but causes 1.2 × clock cycles
- How fast must Computer B clock be?

CPU Time Example (cnt.)

$$CPU\ Time_A = \frac{Clock\ Cycles_A}{Clock\ Rate_A} \Longrightarrow 10s = \frac{Clock\ Cycles_A}{2GHz}$$

$$CPU\ Time_B = \frac{Clock\ Cycles_B}{Clock\ Rate_B} \Longrightarrow 6s = \frac{1.2 \times Clock\ Cycles_A}{Clock\ Rate_B}$$

$$Clock\ Rate_B = \frac{1.2 \times 10s \times 2GHz}{6s} = 4GHz$$

Instruction Count and CPI

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

$$\text{CPU Clock Cycles} = \text{Instruction Count} \times \text{Cycles per Instruction}$$

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

Instruction Count and CPI (cont.)

- 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
    - Instruction mix is a measure of the dynamic frequency of instructions across one or many programs.

CPI Example

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

CPI Example

- Computer A: Cycle Time = 250 ps, CPI = 2.0
- Computer B: Cycle Time = 500 ps, CPI = 1.2

$$\text{CPU Time}_A = \text{Instruction Count} \times \text{CPI}_A \times \text{Cycle Time}_A$$
$$= I \times 2.0 \times 250\text{ps} = I \times 500\text{ps}$$

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

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 class	1	2	3
IC in sequence 1	2	1	2
IC in sequence 2	4	1	1

- Which sequence runs faster?

CPI Example

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

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 CPU2017
  - 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
    - SPECspeed 2017 Integer and SPECspeed 2017 Floating Point

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

SPECspeed 2017 Integer benchmarks on a 1.8 GHz Intel Xeon E5-2650L

Description	*Name*	*Instruction Count x 10^9*	*CPI*	*Clock cycle time (seconds x 10^–9)*	*Execution Time (seconds)*	*Reference Time (seconds)*	*SPECratio*
Perl interpreter	perlbench	2684	0.42	0.556	627	1774	2.83
GNU C compiler	gcc	2322	0.67	0.556	863	3976	4.61
Route planning	mcf	1786	1.22	0.556	1215	4721	3.89
Discrete Event simulation - computer network	omnetpp	1107	0.82	0.556	507	1630	3.21
XML to HTML conversion via XSLT	xalancbmk	1314	0.75	0.556	549	1417	2.58
Video compression	x264	4488	0.32	0.556	813	1763	2.17
Artificial Intelligence: alpha-beta tree search (Chess)	deepsjeng	2216	0.57	0.556	698	1432	2.05
Artificial Intelligence: Monte Carlo tree search (Go)	leela	2236	0.79	0.556	987	1703	1.73
Artificial Intelligence: recursive solution generator (Sudoku)	exchange2	6683	0.46	0.556	1718	2939	1.71
General data compression	xz	8533	1.32	0.556	6290	6182	0.98
Geometric mean							2.36

SPEC Power Benchmark

- 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 E5-2650L

Target Load %	Performance (ssj_ops)	Average Power (watts)
100%	4,864,136	347
90%	4,389,196	312
80%	3,905,724	278
70%	3,418,737	241
60%	2,925,811	212
50%	2,439,017	183
40%	1,951,394	160
30%	1,461,411	141
20%	974,045	128
10%	485,973	115
0%	0	48
Overall Sum	26,815,444	2,165
∑ssj_ops / ∑power =		12,385

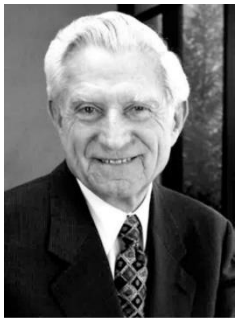
Pitfalls

- Improving an aspect of a computer & expecting a proportional improvement in overall performance
- Millions of Instructions per Second (MIPS) as a Performance Metric

Amdahl's Law

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

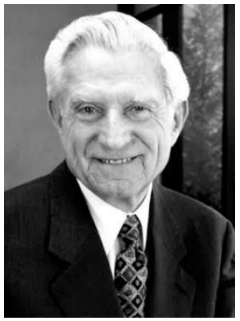
- Example 1:
  - Multiply accounts for 80s from the total 100s
  - Reduce multiply execution time to 25%
  - Overall time improvement?
    - $T_{improved}$ = 80/4 + 20 = 40s
    - Overall time reduction to 40%
- Corollary: make the common case fast



Amdahl's Law

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

- Example 2:
  - Multiply accounts for 80s from the total 100s
  - How much improvement in multiply performance to get 5× overall?
    - 20 = 80/n + 20
    - Cannot be done!



Gene M. Amdahl

MIPS as a Performance Metric

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

Millions of Instructions per Second (MIPS) specifies the instruction execution rate but does not take into account the capabilities of the instructions

MIPS Drawbacks

- Specifies the instruction execution rate but does not take into account the capabilities of the instructions
  - Doesn't account for
    - Differences in ISAs between computers
    - Differences in complexity between instructions

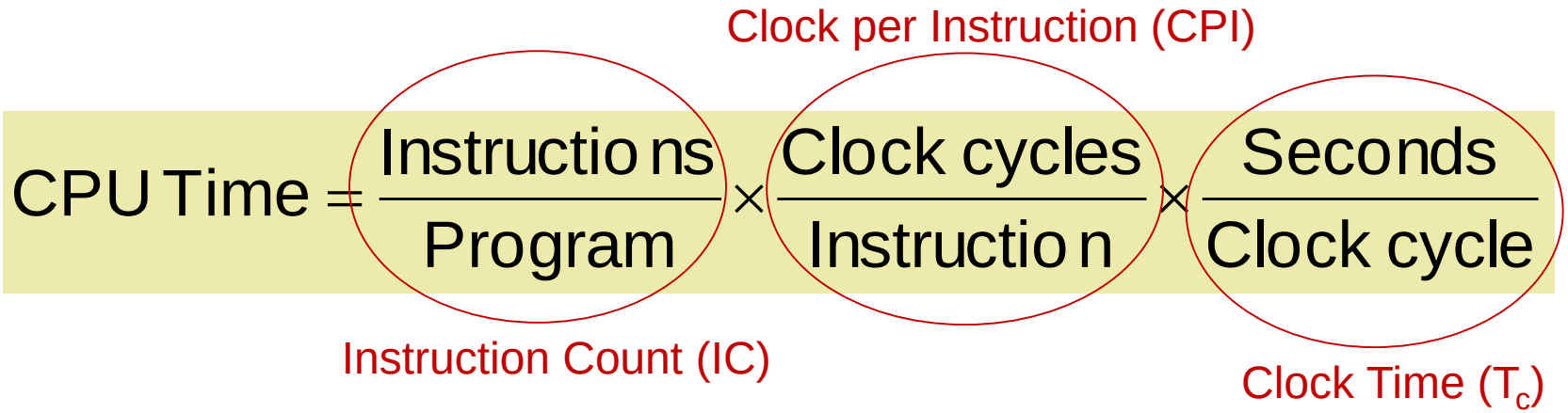
$$\text{MIPS} = \frac{\text{Clock rate}}{\text{CPI} \times 10^6}$$

- Varies between programs on the same computer
  - CPI varies between programs on a given CPU (see here)
- if a new program executes more instructions but each instruction is faster, MIPS can vary independently from performance!

Fallacy: Low Power at Idle

- Look back at Xeon E5-2650L power benchmark
  - At 100% load: 347W
  - At 50% load: 183W (53%)
  - At 10% load: 115W (33%)
- Google datacenter
  - Mostly operates at 10% – 50% load
  - At 100% load less than 1% of the time
- Consider designing processors to make power proportional to load

Performance Summary



- Performance depends on
  - Algorithm: affects IC, possibly CPI
  - Programming language: affects IC, CPI
  - Compiler: affects IC, CPI
  - ISA: affects IC, CPI, T_c