



Tikrit University
Electrical Engineering Department

EE-317
Computer Engineering
2024-2025

Introduction:
Computer Performance, Power & Multi-core CPU

Jalal Nazar Abdulbaqi, Ph.D.

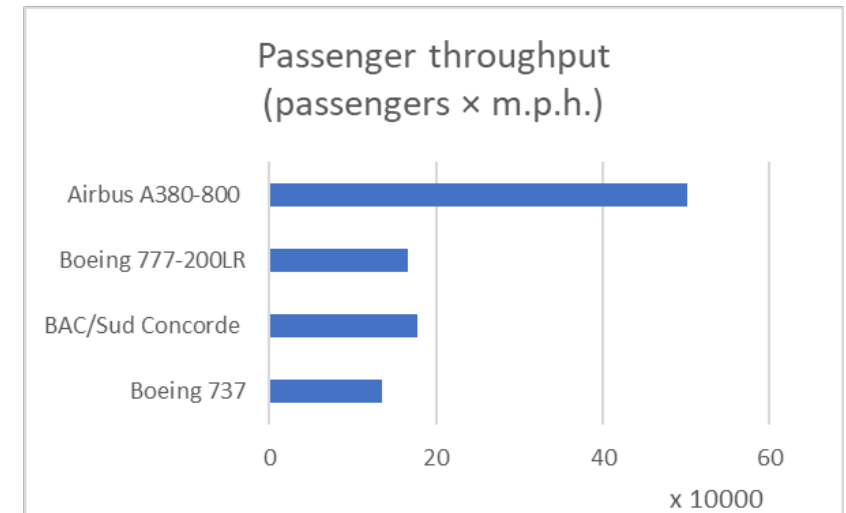
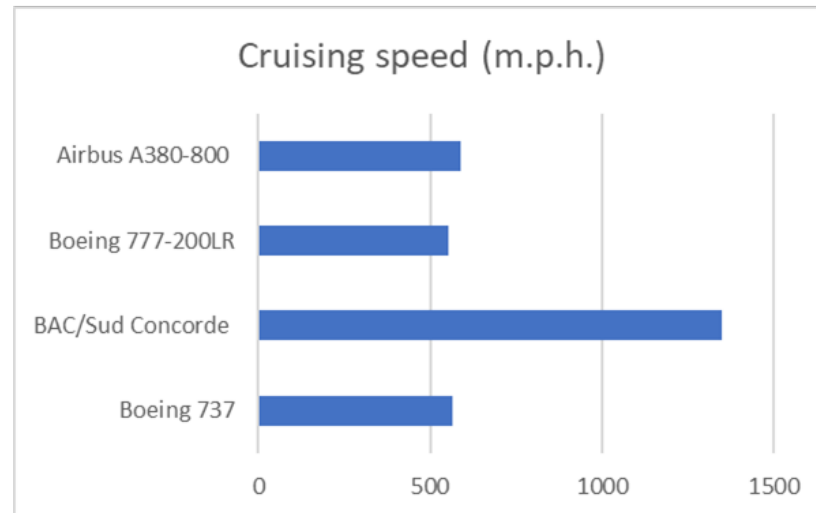
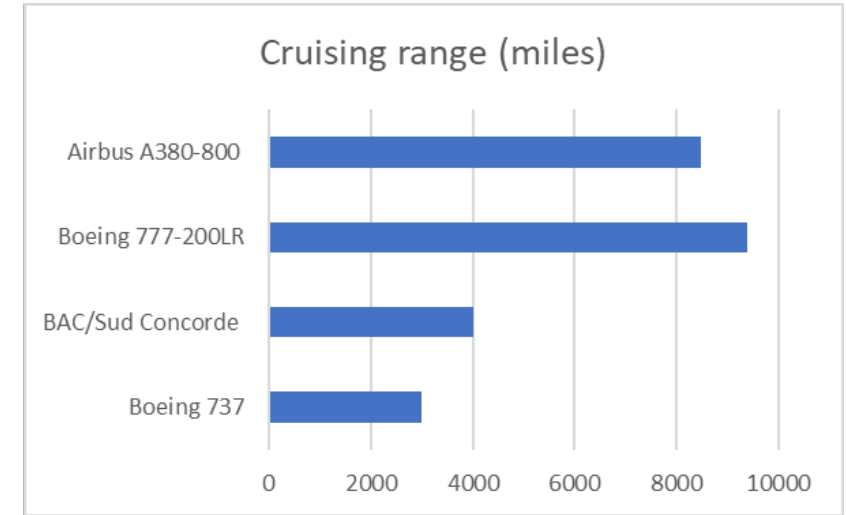
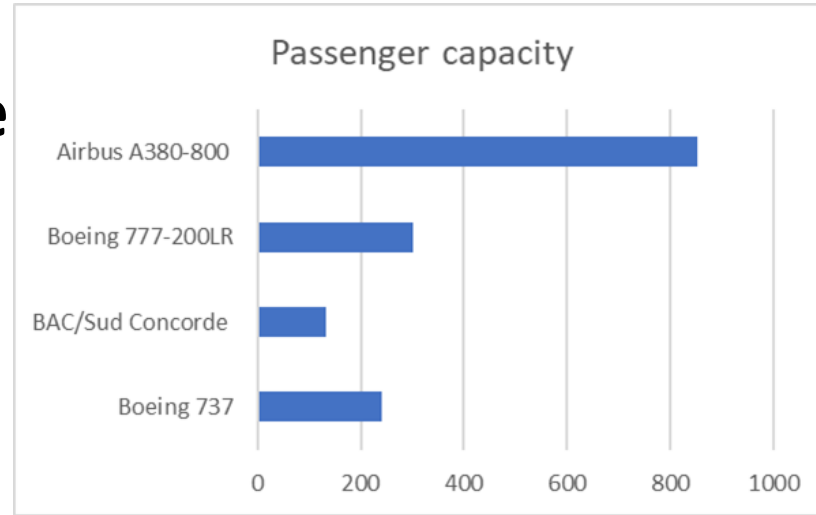
jalal.abdulbaqi@tu.edu.iq

Outline

- **Performance**
- **Power Wall**
- **Multi-processor**
- **Benchmarking**
- **Fallacies and Pitfalls**

Defining Performance

Which airplane has the best performance?



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?
- We'll focus on response time for now...

Relative Performance

- Define Performance = 1/Execution Time
- “X is n time faster than Y”

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

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

Measuring Execution Time

- **Elapsed time**

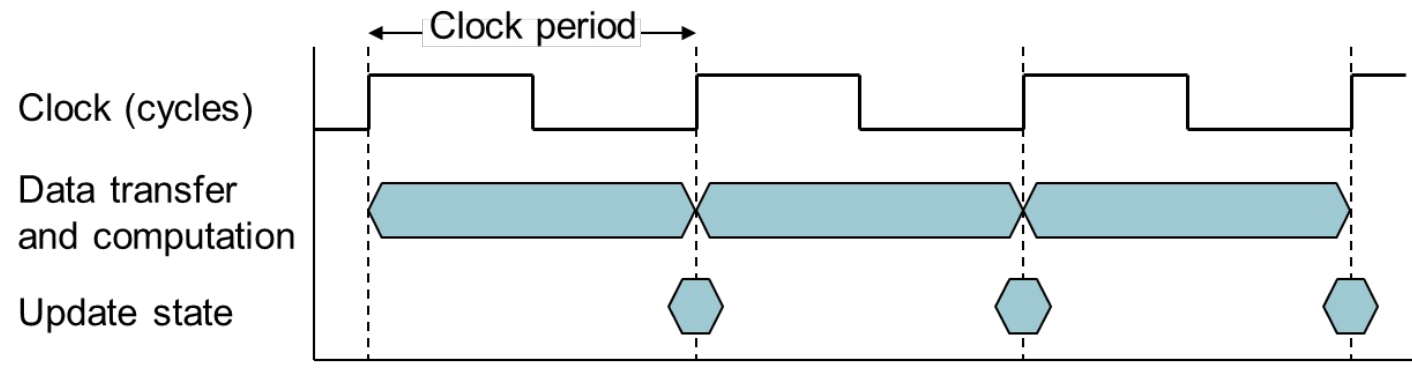
- Total response time, including all aspects
 - Processing, I/O, OS overhead, idle time
- Determines *system performance*

- **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



- **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, how fast must Computer B clock be if
 - Aim for 6s CPU time
 - Can do faster clock, but causes $1.2 \times$ clock cycles

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

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

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

Instruction Count and 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, Instruction Set Architecture (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
 - Which is faster, and by how much?
- } Same ISA

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

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

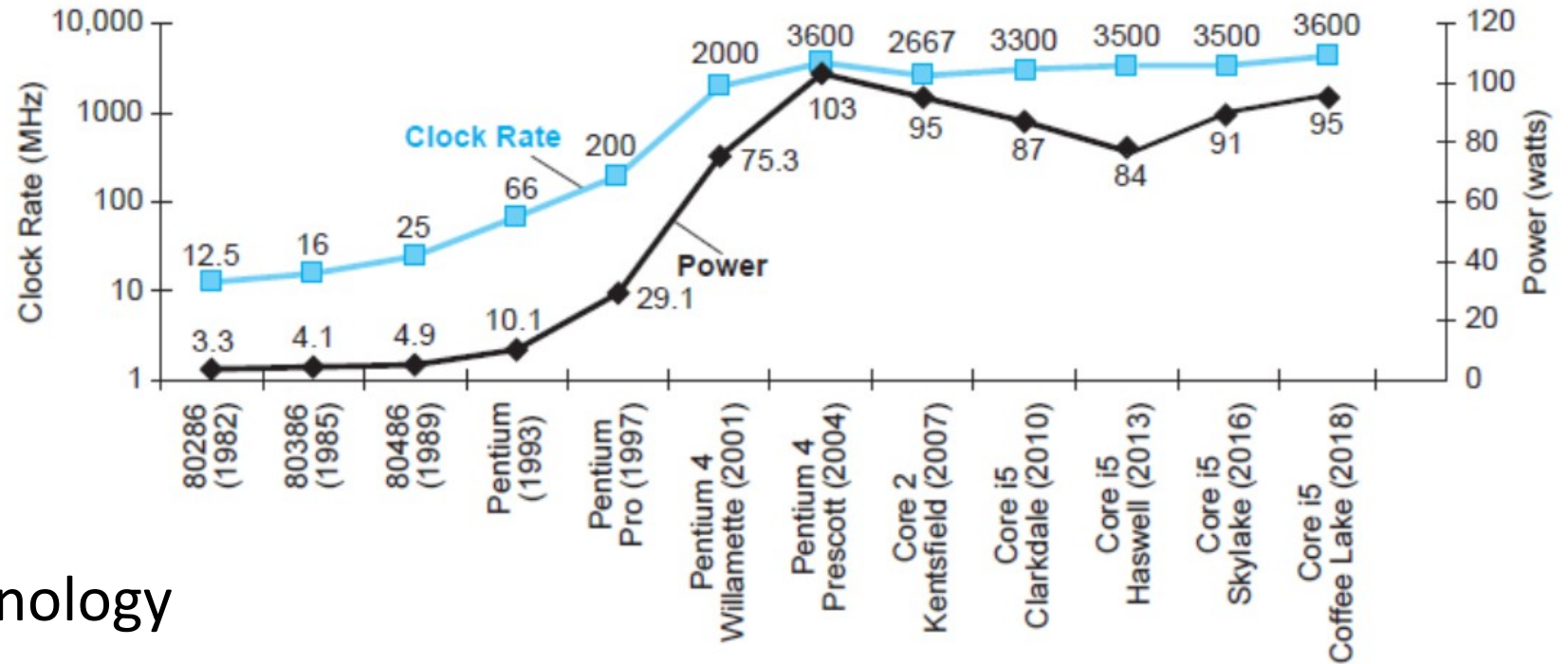
$$\text{CPU Time} = \frac{\text{Instructions}}{\text{Program}} \times \frac{\text{Clock cycles}}{\text{Instruction}} \times \frac{\text{Seconds}}{\text{Clock cycle}}$$

Performance depends on

- **Algorithm:** affects IC, possibly CPI
- **Programming language:** affects IC, CPI
- **Compiler:** affects IC, CPI
- **Instruction Set Architecture (ISA):** affects IC, CPI, T_c

Power Trends

- Why multi-core?



- In CMOS IC technology

×30

5V ---> 1V

×1000

$$\text{Dynamic Power} = \frac{1}{2} \times \text{Capacitive Load} \times \text{Voltage}^2 \times \text{Frequency}$$

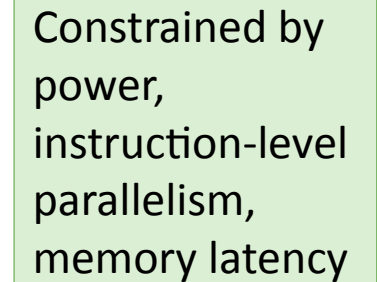
$$\text{DP} = \frac{1}{2} \times C \times V^2 \times F$$

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?



Multiprocessors

- Multi-core 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

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
 - **SPEC speed 2017 Integer #10** (integer)
 - **CFP2006 #13** (floating-point)
 - Elapsed time to execute a selection of programs
 - Negligible I/O, so focuses on CPU performance
 - Normalize relative to reference machine

SPEC CPU Benchmark

- **SPEC ratio**

$$\text{SPEC}_{\text{ratio}} = \text{Execution time (CPU}_{\text{Reference}}) / \text{Execution time (CPU}_{\text{Evaluated}})$$

- ***geometric mean* of *SPEC ratio* summarized SPECspeed 2017**

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

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

<i>Description</i>	<i>Name</i>	<i>Instruction Count x 10⁹</i>	<i>CPI</i>	<i>Clock cycle time (seconds x 10⁻⁹)</i>	<i>Execution Time (seconds)</i>	<i>Reference Time (seconds)</i>	<i>SPECratio</i>
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)

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

SPEC power_ssj 2008 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
$\Sigma \text{ssj_ops} / \Sigma \text{power} =$		12,385

Pitfall: Amdahl's Law

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

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

- Example: multiply accounts for 80s/100s
 - How much improvement in multiply performance to get 5× overall?

$$20 = \frac{80}{n} + 20$$

- Can't be done!

- Corollary: make the common case fast

Fallacy: Low Power at Idle

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

Pitfall: MIPS as a Performance Metric

- **MIPS: Million Instructions Per Second**
 - Doesn't account for
 - Differences in ISAs between computers
 - Differences in complexity between instructions

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

- CPI varies between programs on a given CPU

Pitfall: MIPS as a Performance Metric

- Example:

Measurement	Computer A	Computer B
Instruction Count	10 billion	8 Billion
Clock rate	4 GHz	4 GHz
CPI	1.0	1.1

- Which computer has the higher MIPS rating?
- Which computer faster?

Pitfall: MIPS as a Performance Metric

- $\text{MIPS} = \text{clock rate} / (\text{CPI} \times 10^6)$
 - $\text{MIPS}_A = 4 \times 10^9 / (1 \times 10^6) = \mathbf{4000}$
 - $\text{MIPS}_B = 4 \times 10^9 / (1.1 \times 10^6) = \mathbf{3636}$
- $\text{CPU}_{\text{TIME}} = \text{IC} \times \text{CPI} / \text{clock rate}$
 - $\text{CPU}_{\text{TIME}}A = 10 \times 10^9 \times 1.0 / 4 \times 10^9 = \mathbf{2.5}$ [sec.]
 - $\text{CPU}_{\text{TIME}}B = 8 \times 10^9 \times 1.1 / 4 \times 10^9 = \mathbf{2.2}$ [sec.] **Faster**
- Computer **B** is *faster* than Computer **A**, although it has *lower MIPS* than Computer **A**.

Concluding Remarks

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