# CS/ECE 3810: Computer Organization

# Lecture 3: Understanding Performance

Anton Burtsev
August, 2022

# Defining performance

| Airplane | Passenger capacity | Cruising range (miles) | Cruising speed (m.p.h.) | Passenger throughput (passengers × m.p.h.) |
|---|---|---|---|---|
| Boeing 777 | 375 | 4630 | 610 | 228,750 |
| Boeing 747 | 470 | 4150 | 610 | 286,700 |
| BAC/Sud Concorde | 132 | 4000 | 1350 | 178,200 |
| Douglas DC-8-50 | 146 | 8720 | 544 | 79,424 |

**FIGURE 1.14 The capacity, range, and speed for a number of commercial airplanes.** The last column shows the rate at which the airplane transports passengers, which is the capacity times the cruising speed (ignoring range and takeoff and landing times).

# Performance metrics

- Possible measures:
  - response time – time elapsed between start and end of a program
  - throughput – amount of work done in a fixed time
- The two measures are usually linked
  - A faster processor will improve both
  - More processors will likely only improve throughput
  - Some policies will improve throughput and worsen response time (or vice versa)
- What influences performance?

# Example

- Do the following changes to a computer system increase throughput, decrease response time, or both?
  - Replacing the processor in a computer with a faster version
  - Adding additional processors to a system that uses multiple processors for separate tasks—for example, searching the web

# Execution Time

Consider a system X executing a fixed workload W

$$\text{Performance}_X = \frac{1}{\text{Execution time}_X}$$

Execution time = response time = wall clock time

- Note that this includes time to execute the workload as well as time spent by the operating system co-ordinating various events

The UNIX “time” command breaks up the wall clock time as user and system time

# Speedup and Improvement

- Example:
  - *System X executes a program in 10 seconds, system Y executes the same program in 15 seconds*
- System X is 1.5 times faster than system Y
- The speedup of system X over system Y is 1.5 (the ratio)
  = perf X / perf Y = exectime Y / exectime X
- The performance improvement of X over Y is
  1.5 -1 = 0.5 = 50% = (perf X – perf Y) / perf Y = speedup - 1

# Factors influencing performance

# A Primer on Clocks and Cycles

# Performance Equation

$$\text{CPU execution time for a program} = \text{CPU clock cycles for a program} \times \text{Clock cycle time}$$

or alternatively

$$\text{CPU execution time for a program} = \frac{\text{CPU clock cycles for a program}}{\text{Clock rate}}$$

If a processor has a frequency of 2 GHz, the clock ticks 2 billion times in a second as we'll soon see, with each clock tick, one or more/less instructions may complete

# Performance Equation

$$\text{CPU execution time for a program} = \text{CPU clock cycles for a program} \times \text{Clock cycle time}$$

or alternatively

$$\text{CPU execution time for a program} = \frac{\text{CPU clock cycles for a program}}{\text{Clock rate}}$$

If a processor has a frequency of 2 GHz, the clock ticks 2 billion times in a second – as we'll soon see, with each clock tick, one or more/less instructions may complete

If a program runs for 10 seconds on a 2 GHz processor, how many clock cycles did it run for?

# Performance Equation (continued)

$$\text{CPU clock cycles} = \text{Instructions for a program} \times \begin{matrix}\text{Average clock cycles} \\ \text{per instruction}\end{matrix}$$

- We call *average clock cycles per instruction* = CPI
- If we substitute CPU clock cycles in

$$\begin{matrix}\text{CPU execution time} \\ \text{for a program}\end{matrix} = \begin{matrix}\text{CPU clock cycles} \\ \text{for a program}\end{matrix} \times \text{Clock cycle time}$$

- We get:

$$\text{CPU time} = \text{Instruction count} \times \text{CPI} \times \text{Clock cycle time}$$

- If a 2 GHz processor graduates an instruction every third cycle, how many instructions are there in a program that runs for 10 seconds?

# Classic performance equation

$$\text{CPU time} = \text{Instruction count} \times \text{CPI} \times \text{Clock cycle time}$$

- Three factors that influence performance
  - Instruction count
  - CPI
  - Clock cycle time
- If a 2 GHz processor graduates an instruction every third cycle, how many instructions are there in a program that runs for 10 seconds?

# Factors Influencing Performance

$$\text{CPU time} = \text{Instruction count} \times \text{CPI} \times \text{Clock cycle time}$$

| Hardware or software component |
| --- |
| Algorithm |
| Programming language |
| Compiler |
| Instruction set architecture |

| Hardware or software component | Affects what? | How? |
|---|---|---|
| Algorithm | Instruction count, possibly CPI | The algorithm determines the number of source program instructions executed and hence the number of processor instructions executed. The algorithm may also affect the CPI, by favoring slower or faster instructions. For example, if the algorithm uses more divides, it will tend to have a higher CPI. |
| Programming language | Instruction count, CPI | The programming language certainly affects the instruction count, since statements in the language are translated to processor instructions, which determine instruction count. The language may also affect the CPI because of its features; for example, a language with heavy support for data abstraction (e.g., Java) will require indirect calls, which will use higher CPI instructions. |
| Compiler | Instruction count, CPI | The efficiency of the compiler affects both the instruction count and average cycles per instruction, since the compiler determines the translation of the source language instructions into computer instructions. The compiler's role can be very complex and affect the CPI in complex ways. |
| Instruction set architecture | Instruction count, clock rate, CPI | The instruction set architecture affects all three aspects of CPU performance, since it affects the instructions needed for a function, the cost in cycles of each instruction, and the overall clock rate of the processor. |

# Benchmark Suites

- Each vendor announces a SPEC rating for their system
  - a measure of execution time for a fixed collection of programs
  - is a function of a specific CPU, memory system, IO system, operating system, compiler
  - enables easy comparison of different systems

The key is coming up with a collection of relevant programs

# Comparing performance

# SPEC CPU

- SPEC: System Performance Evaluation Corporation, an industry consortium that creates a collection of relevant programs
- SPEC 2006 includes 12 integer and 17 floating-point applications
- The SPEC rating specifies how much faster a system is, compared to a baseline machine – a system with SPEC rating 600 is 1.5 times faster than a system with SPEC rating 400
- Note that this rating incorporates the behavior of all 29 programs – this may not necessarily predict performance for your favorite program!
- Latest version: SPEC 2017

| 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 |

**FIGURE 1.18 SPECINTC2006 benchmarks running on a 2.66 GHz Intel Core i7 920.** As the equation on page 35 explains, execution time is the product of the three factors in this table: instruction count in billions, clocks per instruction (CPI), and clock cycle time in nanoseconds. SPECratio is simply the reference time, which is supplied by SPEC, divided by the measured execution time. The single number quoted as SPECINTC2006 is the geometric mean of the SPECratios.

# Deriving a Single Performance Number

How is the performance of 29 different apps compressed into a single performance number?

- SPEC uses geometric mean (GM) – the execution time of each program is multiplied and the N$^{th}$ root is derived

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

- Another popular metric is arithmetic mean (AM) – the average of each program's execution time

- Weighted arithmetic mean – the execution times of some programs are weighted to balance priorities

# Amdahl's Law

- Architecture design is very bottleneck-driven – make the common case fast, do not waste resources on a component that has little impact on overall performance/power

- Amdahl's Law: performance improvements through an enhancement is limited by the fraction of time the enhancement comes into play

- Example: a web server spends 40% of time in the CPU and 60% of time doing I/O – a new processor that is ten times faster results in a 36% reduction in execution time (speedup of 1.56) – Amdahl's Law states that maximum execution time reduction is 40% (max speedup of 1.66)

# Recap

- Knowledge of hardware improves software quality: compilers, OS, threaded programs, memory management
- Important trends: growing transistors, move to multi-core and accelerators, slowing rate of performance improvement, power/thermal constraints, long memory/disk latencies
- Reasoning about performance: clock speeds, CPI, benchmark suites, performance and power equations
- Next: assembly instructions

# The power wall

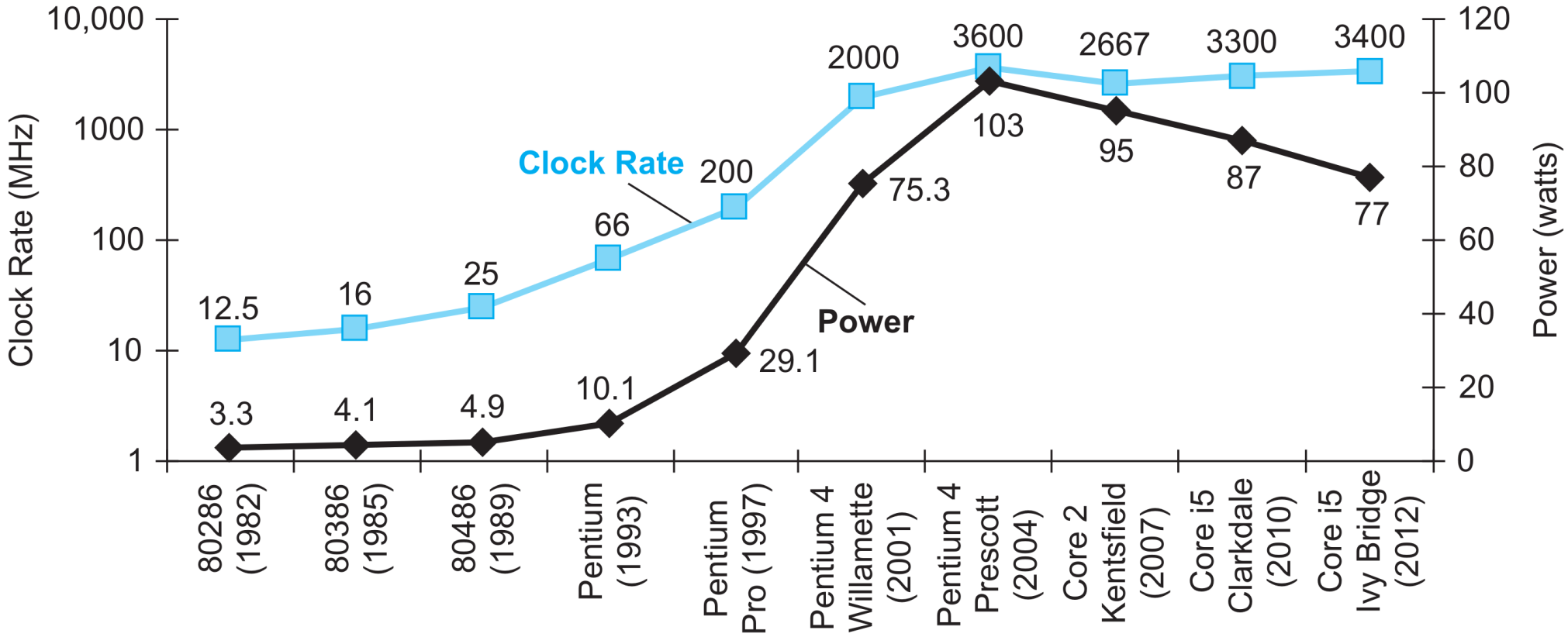



**FIGURE 1.16 Clock rate and Power for Intel x86 microprocessors over eight generations and 25 years.** The Pentium 4 made a dramatic jump in clock rate and power but less so in performance. The Prescott thermal problems led to the abandonment of the Pentium 4 line. The Core 2 line reverts to a simpler pipeline with lower clock rates and multiple processors per chip. The Core i5 pipelines follow in its footsteps.

# Power and energy

- Total energy = dynamic energy + leakage
- Dynamic energy:

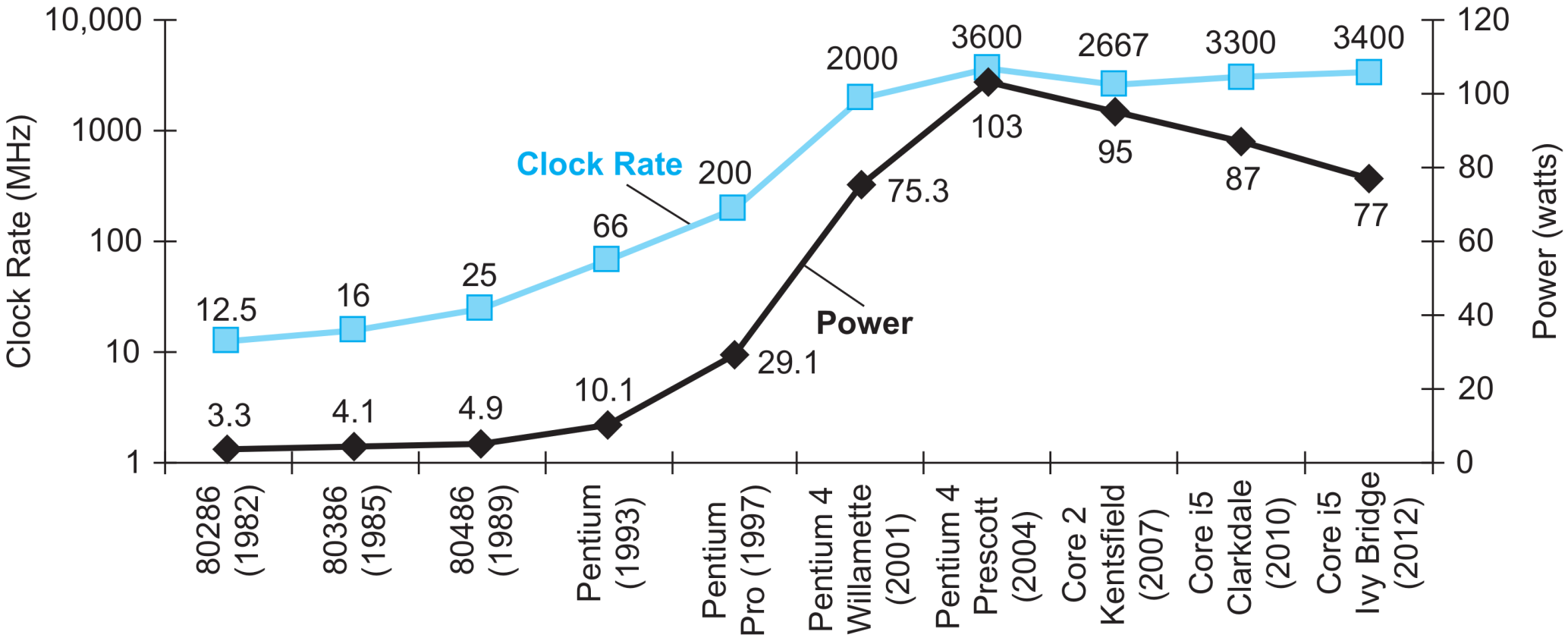
$$Energy \propto Capacitive\ load \times Voltage^2$$

- Power required per transistor is just the product of energy of a transition and the frequency of transitions:

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

- Frequency switched is a function of clock rate
- Capacitive load per transistor is a function
  - number of transistors connected to an output (called the fanout)
  - technology, which determines the capacitance of both wires and transistors

- In 20 years voltage was reduced from 5V to 1V
  - 15% per generation
- Clock rates increased 1000 times
- Power increased  30x



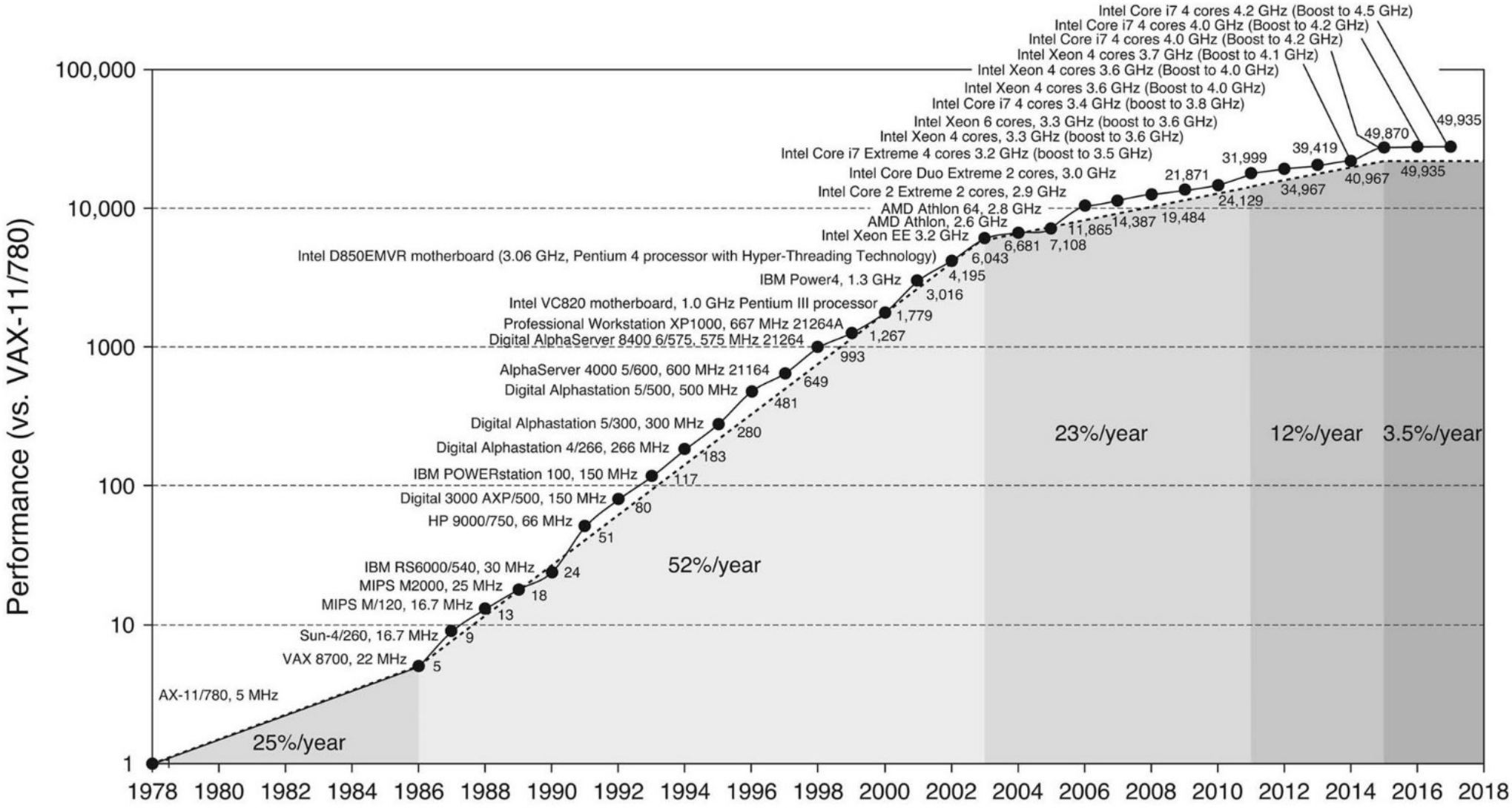



**FIGURE 1.17 Growth in processor performance since the mid-1980s.**

Thank you!