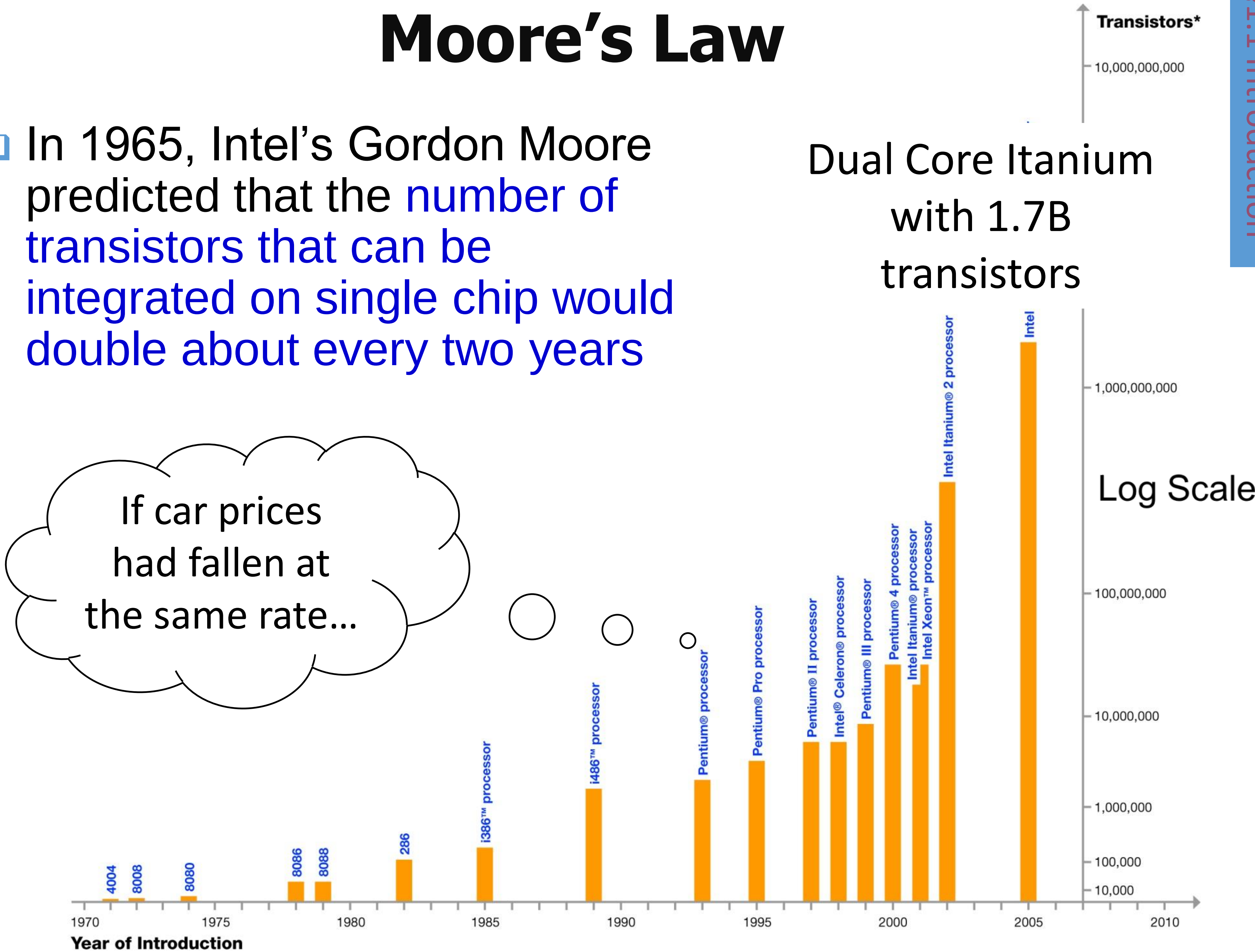
Computer Abstractions and Technology

Jongeun Lee

School of ECE, UNIST

Moore's Law

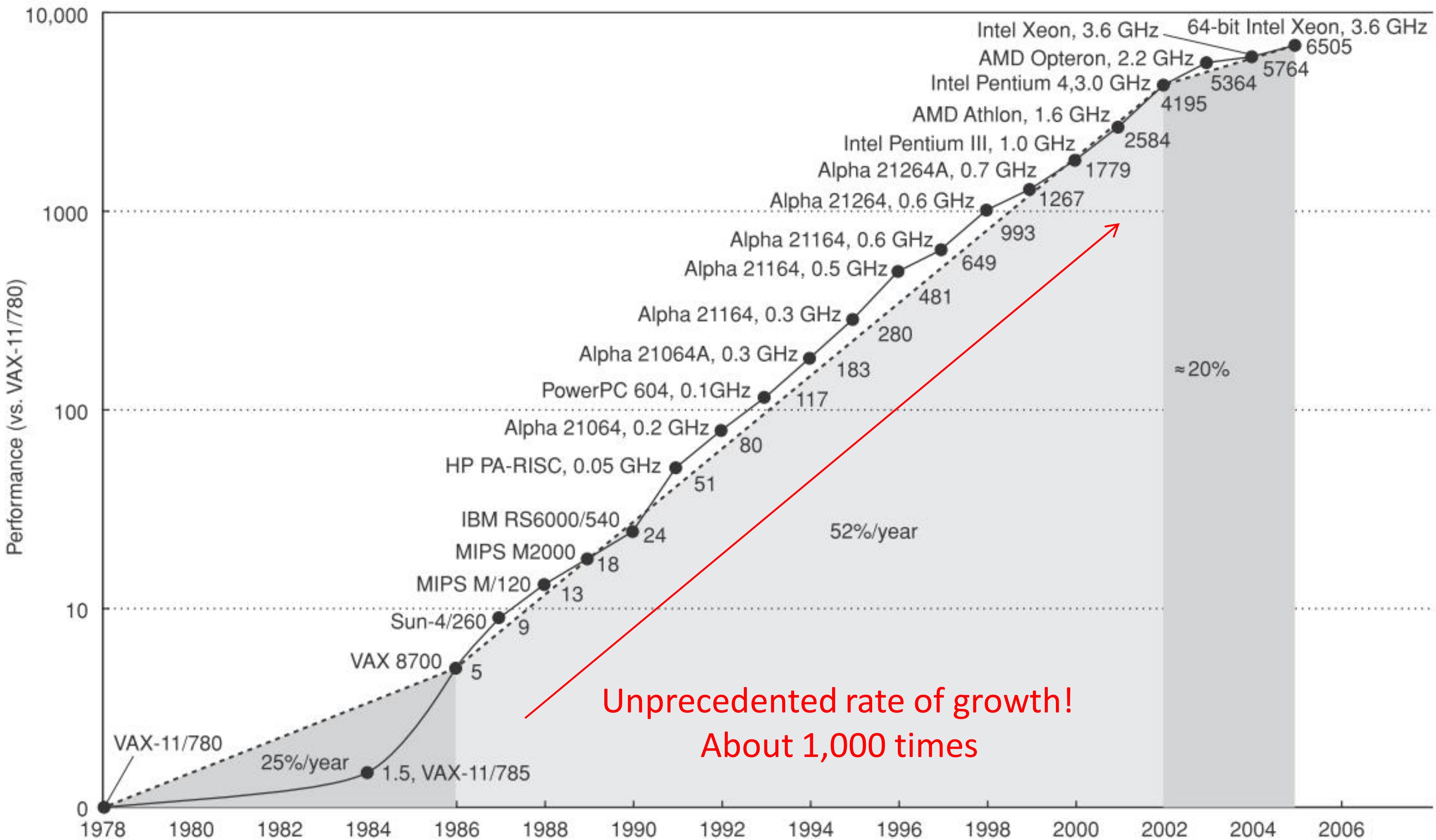
- In 1965, Intel's Gordon Moore predicted that the number of transistors that can be integrated on single chip would double about every two years



*Note: Vertical scale of chart not proportional to actual Transistor count.

Courtesy, Intel ®

Processor Performance ('78~'05)



The Computer Revolution

- **Computers are pervasive**
- **Progress in computer technology**
  - Underpinned by Moore's Law
- **Makes novel applications feasible**
  - Computers in automobiles
  - Cell phones
  - Human genome project
  - World Wide Web
  - Search Engines

Classes of Computers

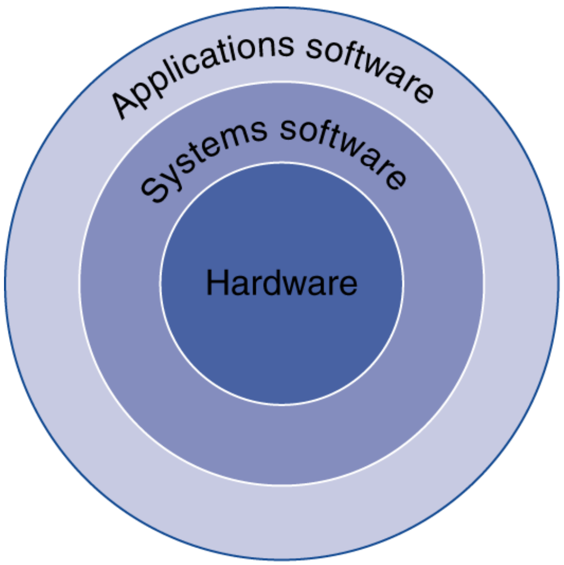
- **Desktop computers**
  - General purpose, variety of software
  - Subject to cost/performance tradeoff
- **Server computers**
  - High capacity, performance, reliability
  - Range from small servers to building sized
- **Embedded computers**
  - Hidden as components of systems
  - Stringent power/performance/cost constraints (often mobile/hand-held)
  - Largest number, and still fastest growing !!

Image: Wikipedia.org

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

Below Your Program



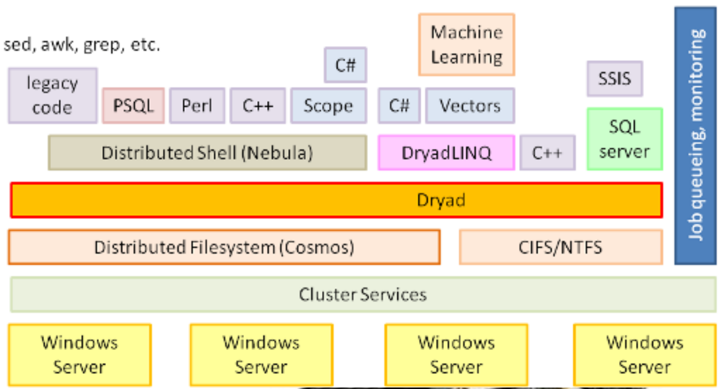
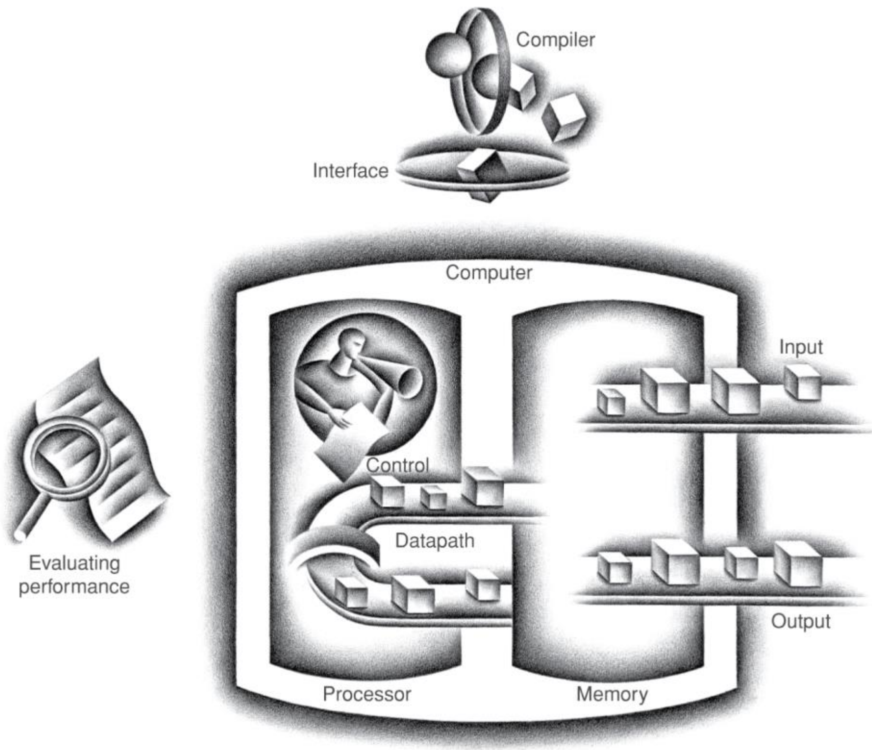


Image credit: microsoft.com, xilinx.com

- **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

Components of a Computer

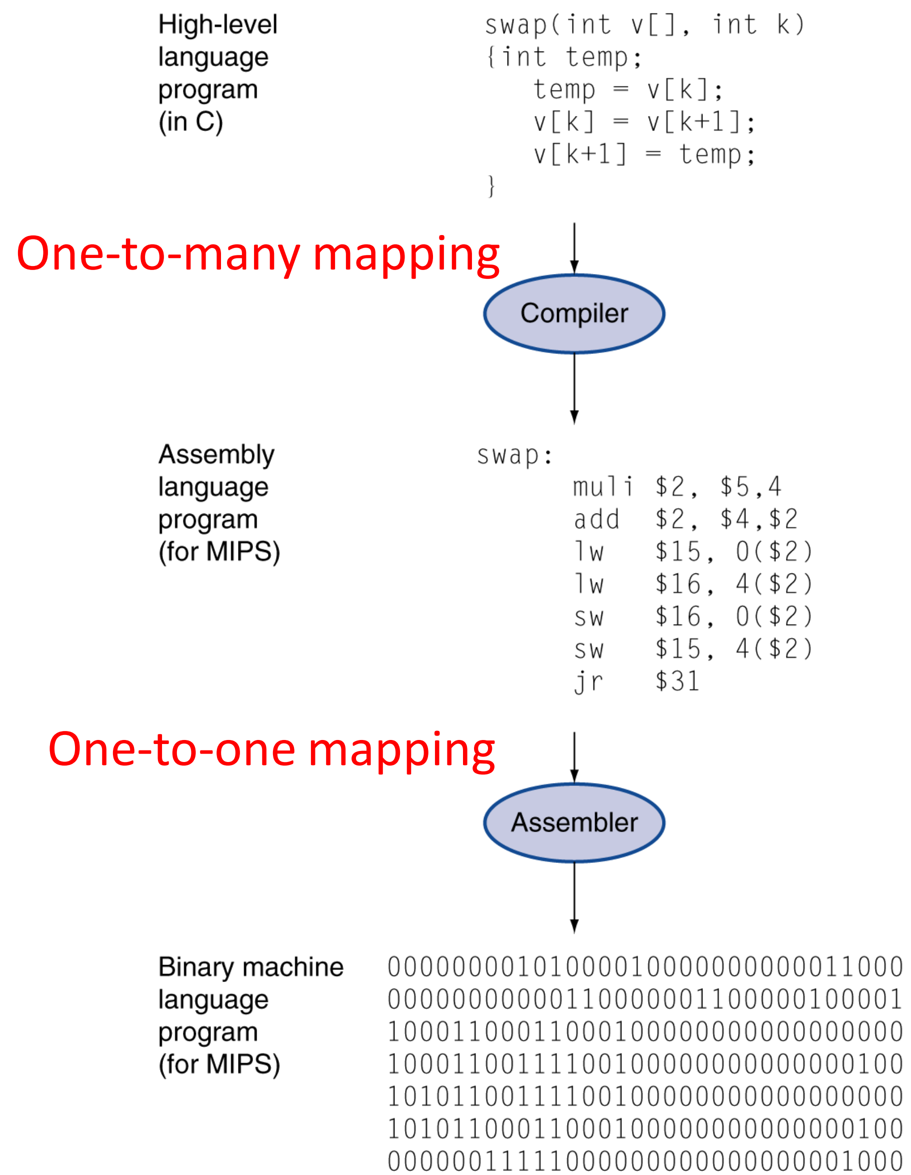
The BIG Picture



- **Same components for all kinds of computer**
  - Desktop, server, embedded
- **Input/output includes**
  - User-interface devices
    - Display, keyboard, mouse
  - Storage devices
    - Hard disk, CD/DVD, flash
  - Network adapters
    - For communicating with other computers

Levels of Program Code

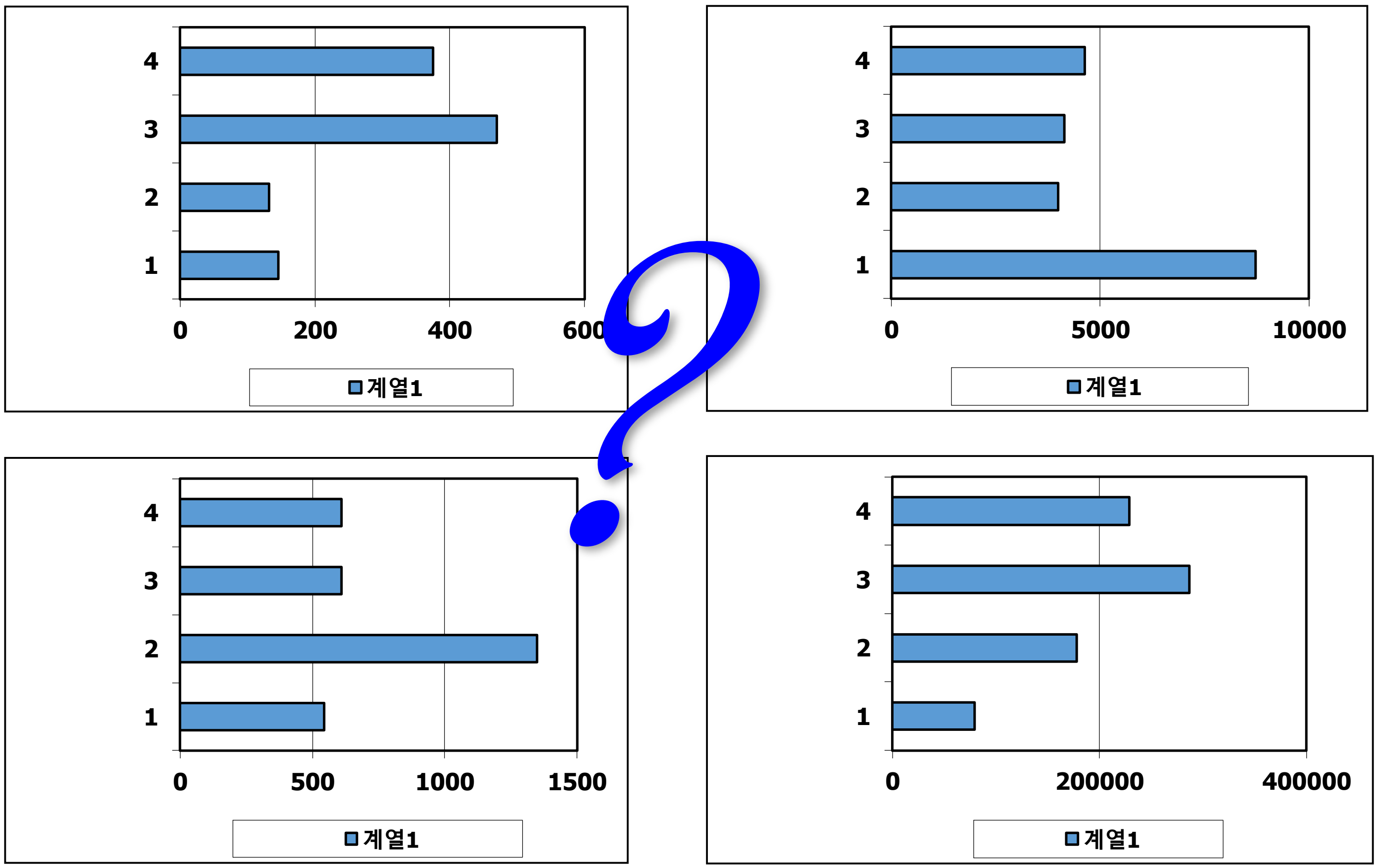
- **High-level language**
  - Level of abstraction closer to problem domain
  - Provides for productivity and portability
- **Assembly language**
  - Textual representation of instructions
- **Hardware representation**
  - Binary digits (bits)
  - Encoded instructions and data



Performance

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" means the following**

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

- Example: time taken to run a program
  - 10s on A, 15s on B
  - Execution Time_B / Execution Time_A
    = 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

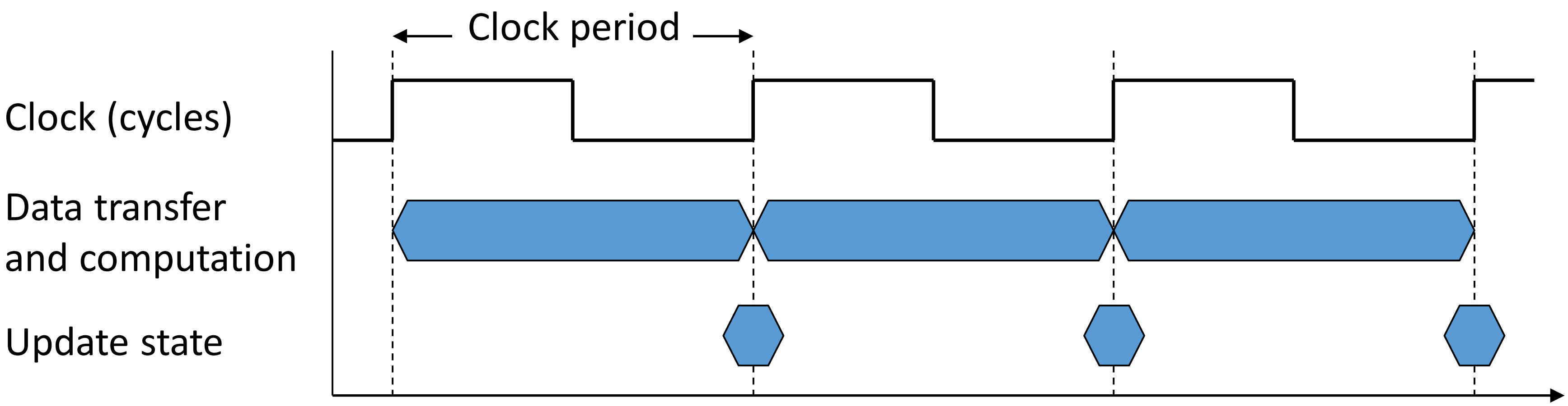
```
$ time find . -name 'xxxx'

real    0m18.102s
user    0m1.234s
sys     0m4.515s
```

- **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., 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
  - Hardware designer must often trade off clock rate against cycle count

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**
  - Determined by CPU hardware
  - If different instructions have different CPI
    - Average CPI affected by instruction mix

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

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

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

Examples

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

- **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?**

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

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

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

CPI Example

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

CPI Example

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

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

$$= \text{I} \times 2.0 \times 250\text{ps} = \text{I} \times 500\text{ps}$$

A is faster…

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

$$= \text{I} \times 1.2 \times 500\text{ps} = \text{I} \times 600\text{ps}$$

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

…by this much

Which Program is Faster?

- **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

Note: IC (Instruction Count)

Which Program is Faster?

- **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

Note: IC (Instruction Count)

- Sequence 1: IC = 5
  - Clock Cycles
    $= 2\times1 + 1\times2 + 2\times3$
    $= 10$
  - Avg. CPI = 10/5 = 2.0
- Sequence 2: IC = 6
  - Clock Cycles
    $= 4\times1 + 1\times2 + 1\times3$
    $= 9$
  - Avg. CPI = 9/6 = 1.5

Summarizing Performance

- **Computer A**
  - Program 1: 1s
  - Program 2: 1s
- **Computer B**
  - Program 1: 0.5s
  - Program 2: 1.5s
- **Now, which is faster?**

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)

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

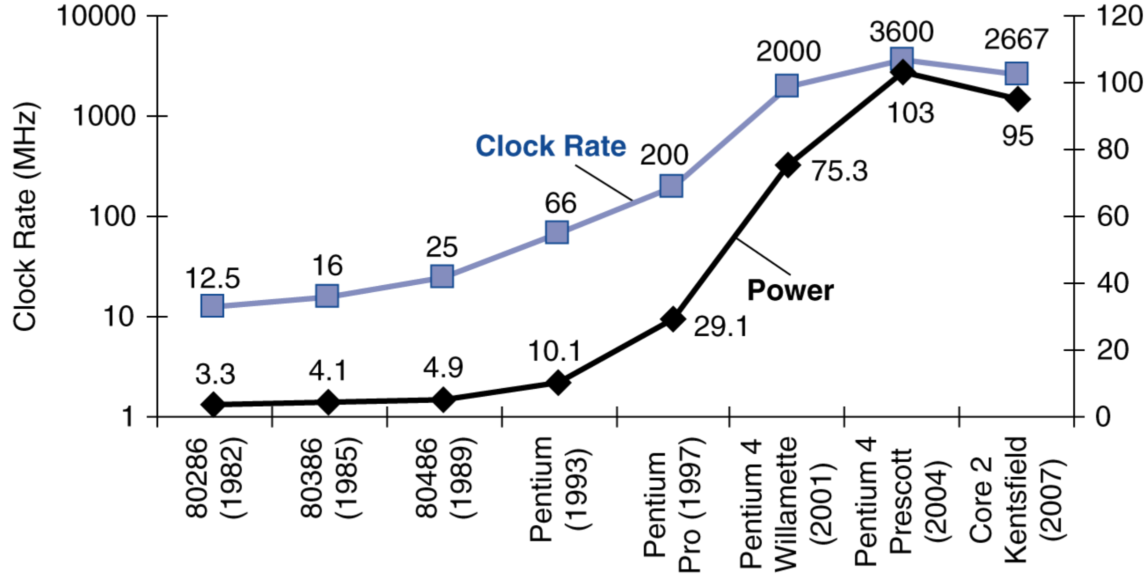
CINT2006 for Opteron X4 2356

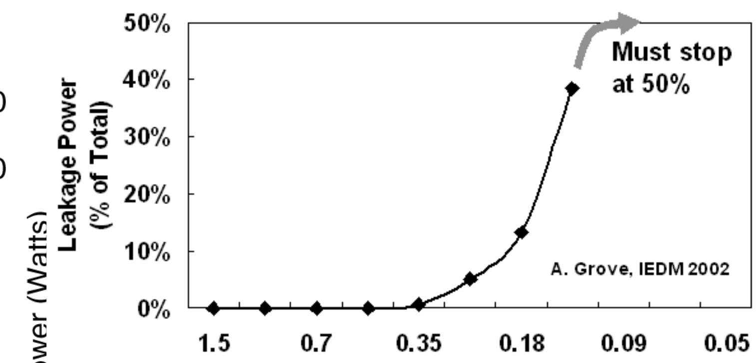
Name	Description	$IC \times 10^9$	CPI	Tc (ns)	Exec time	Ref time	SPECratio
perl	Interpreted string processing	2,118	0.75	0.40	637	9,777	15.3
bzip2	Block-sorting compression	2,389	0.85	0.40	817	9,650	11.8
gcc	GNU C Compiler	1,050	1.72	0.47	24	8,050	11.1
mcf	Combinatorial optimization	336	10.00	0.40	1,345	9,120	6.8
go	Go game (AI)	1,658	1.09	0.40	721	10,490	14.6
hmmer	Search gene sequence	2,783	0.80	0.40	890	9,330	10.5
sjeng	Chess game (AI)	2,176	0.96	0.48	37	12,100	14.5
libquantum	Quantum computer simulation	1,623	1.61	0.40	1,047	20,720	19.8
h264avc	Video compression	3,102	0.80	0.40	993	22,130	22.3
omnetpp	Discrete event simulation	587	2.94	0.40	690	6,250	9.1
astar	Games/path finding	1,082	1.79	0.40	773	7,020	9.1
xalancbmk	XML parsing	1,058	2.70	0.40	1,143	6,900	6.0
Geometric mean							11.7

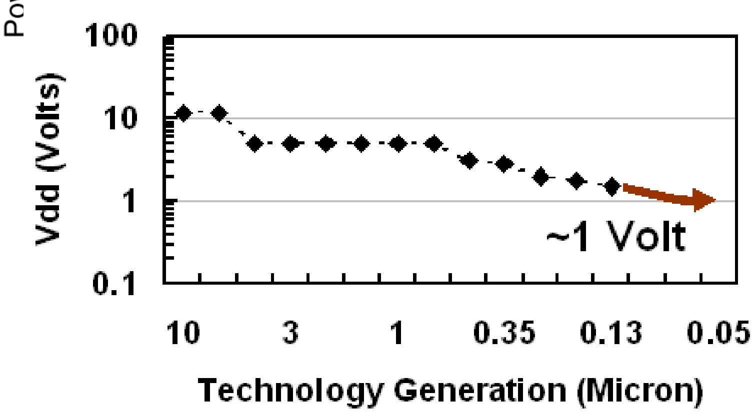
High cache miss rates

Power

Power Trends







(source: intel.com)

- **In CMOS IC technology**

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

×30

5V → 1V

×1000

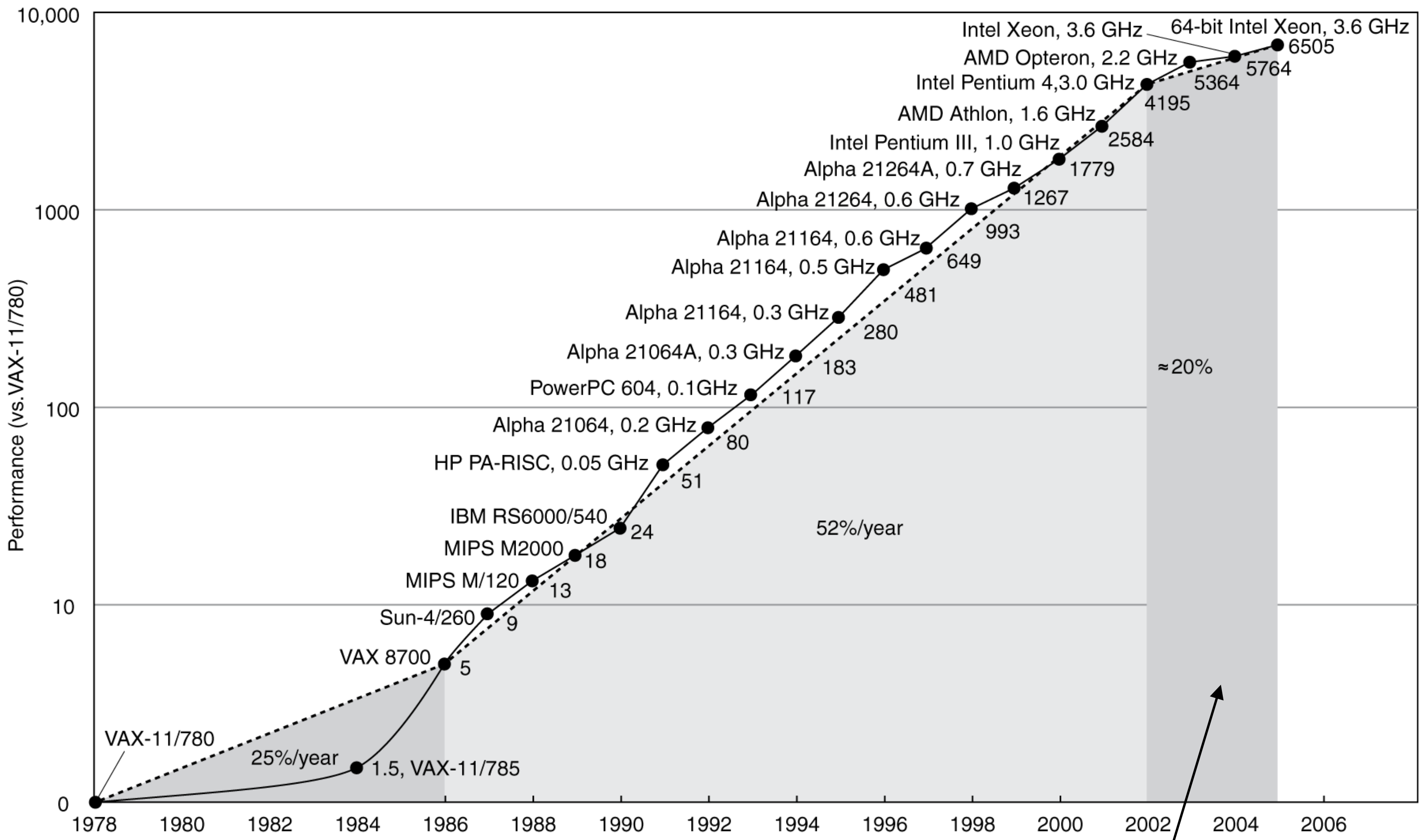
Reducing Power

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

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

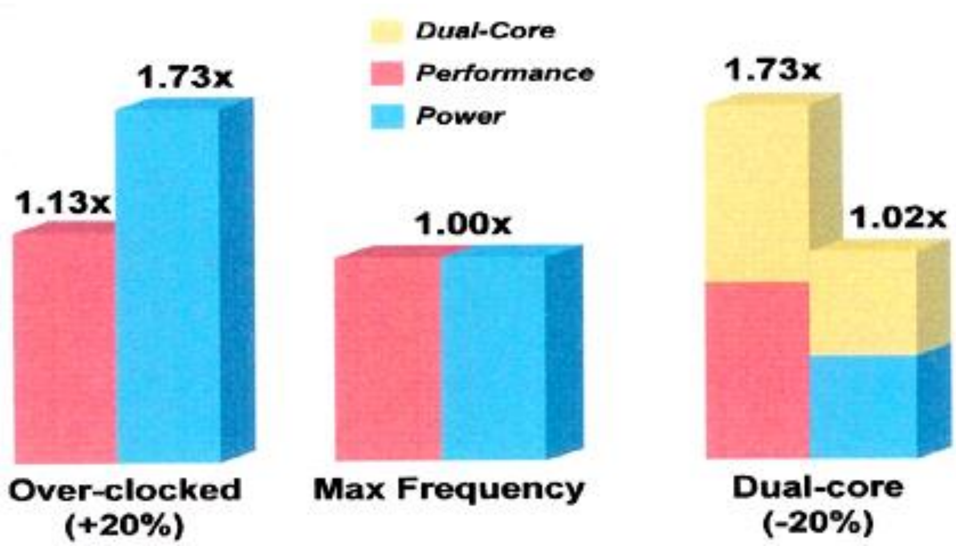
- The power wall
  - We can’t reduce voltage further
  - We can’t remove more heat
- How else can we improve performance?

Uniprocessor Performance



Constrained by power, instruction-level parallelism, memory lat ency

Rise of Multicore Processors



(source: Intel Inc. via Embedded.com)

- **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 Dissipation (in more detail)

- **Power in digital CMOS circuits**

$$P_{CMOS} = P_{dynamic} + P_{static}$$

- **Dynamic power**

$$P_{dynamic} = \tfrac{1}{2}\, C\, V_{DD}^{2}\, f_{CLK}$$

- **Static (leakage) power**

$$P_{static} = I_{leakage}\, V_{DD}$$

Pitfalls

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

Pitfall: MIPS as a Performance Metric

- **MIPS: Millions of Instructions Per Second**
  - 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