

Chapter 1

Computer Abstractions and Technology

**Instructor: Prof. Peter Lian
Department of Electrical
Engineering & Computer Science
Lassonde School of Engineering
York University**

Acknowledgement

- The slides are adapted from Computer Organization and Design, 4th Edition, by David A. Patterson and John L. Hennessy, 2008, published by MK (Elsevier)

CSE2021 Computer Organization

- Instructor: Prof. Peter Lian

email: peterlian@cse.yorku.ca

tel: 416-736-2100 ext 44647

- Course Web:

https://wiki.cse.yorku.ca/course_archive/2013-14/F/2021/

- Schedule:

- Lectures: MW 17:30 – 1900, Room R S137

- Labs: Lab-01 M 19:00 – 22:00, LAS 1006

Lab-02 T 19:00 – 22:00, LAS 1006/1002

- Office hours: MW 15:00 – 17:00 @ LAS 1012C

CSE2021 Computer Organization

- Text book:

Computer Organization and Design

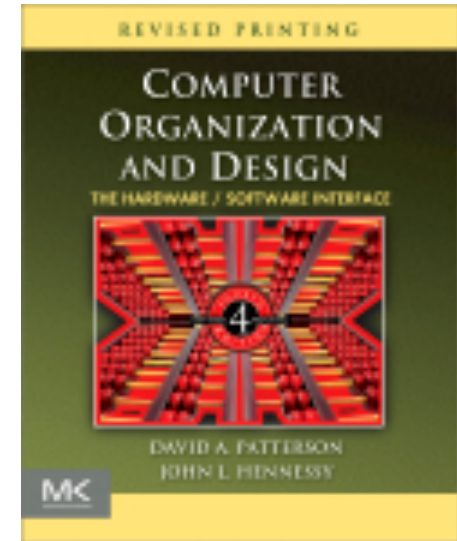
-- The Hardware/Software Interface

4th Edition

by David A. Patterson and John L. Hennessy

Morgan Kaufmann Publishers (Elsevier)

ISBN 978-0-12-374750-1



- Assessment:

- Assignments/Quizzes: 20%
- Lab projects: 25%
- Midterm test: 20%
- Final exam: 35%

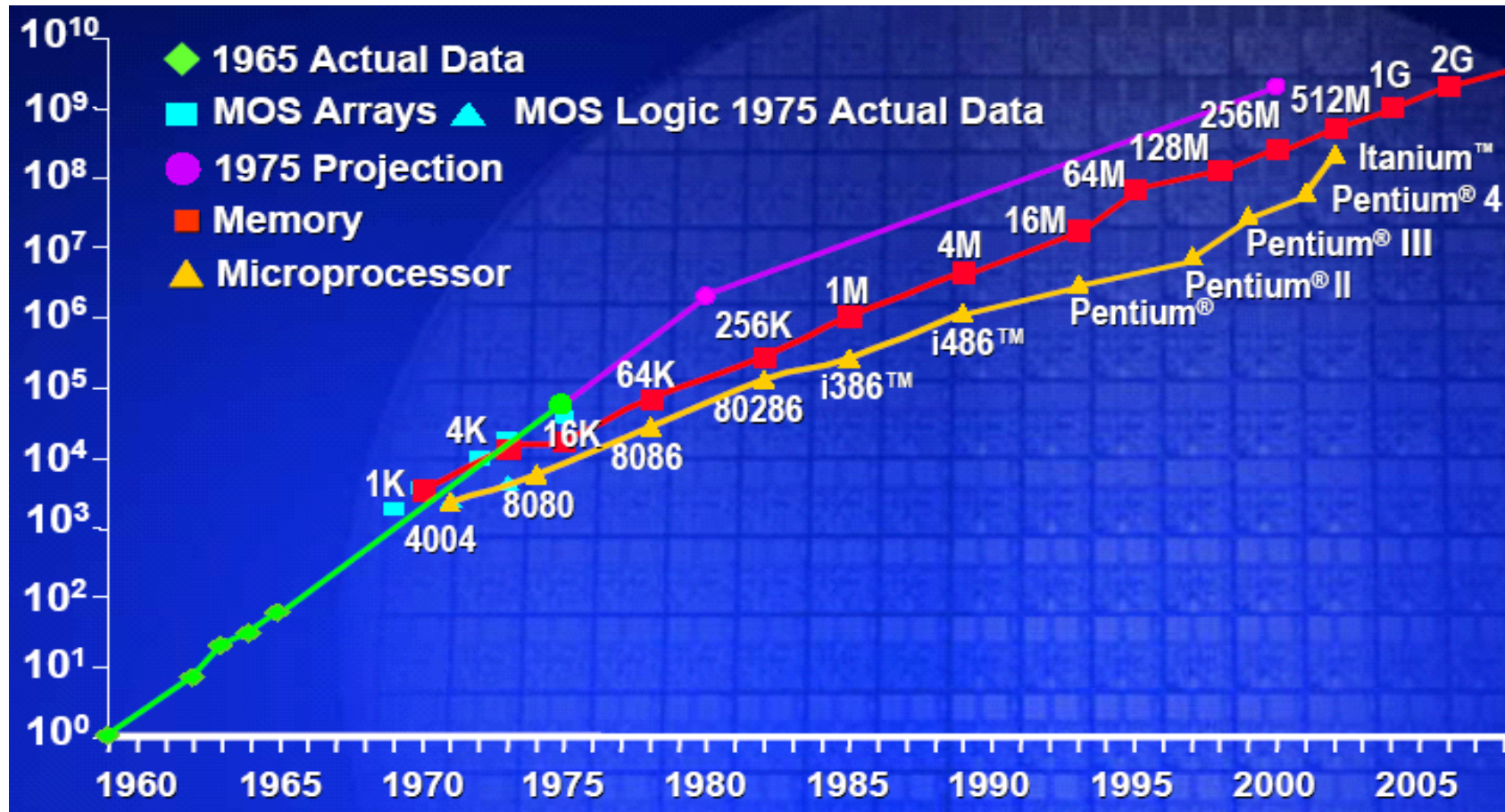
CSE2021 Computer Organization

- Topics covered:
 - Introduction
 - Computer abstractions and technology
 - Language of the computer: high level language versus assembly language versus machine language
 - Arithmetic for computers
 - The processor
 - Memory, storage, and input/output

Introduction

The Computer Revolution

■ Moore's Law

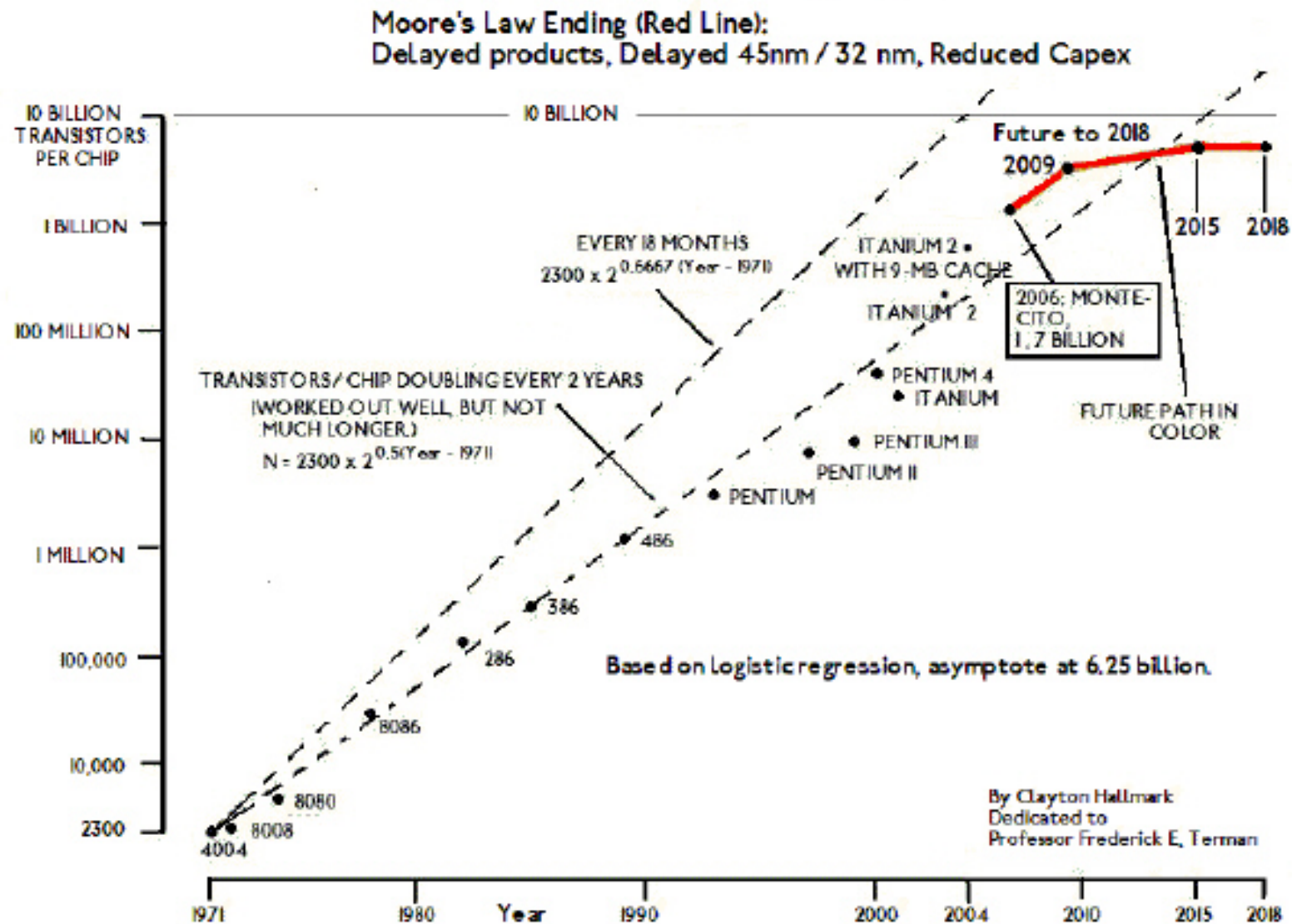


Source: ISSCC 2003 G. Moore "No exponential is forever, but 'forever' can be delayed"

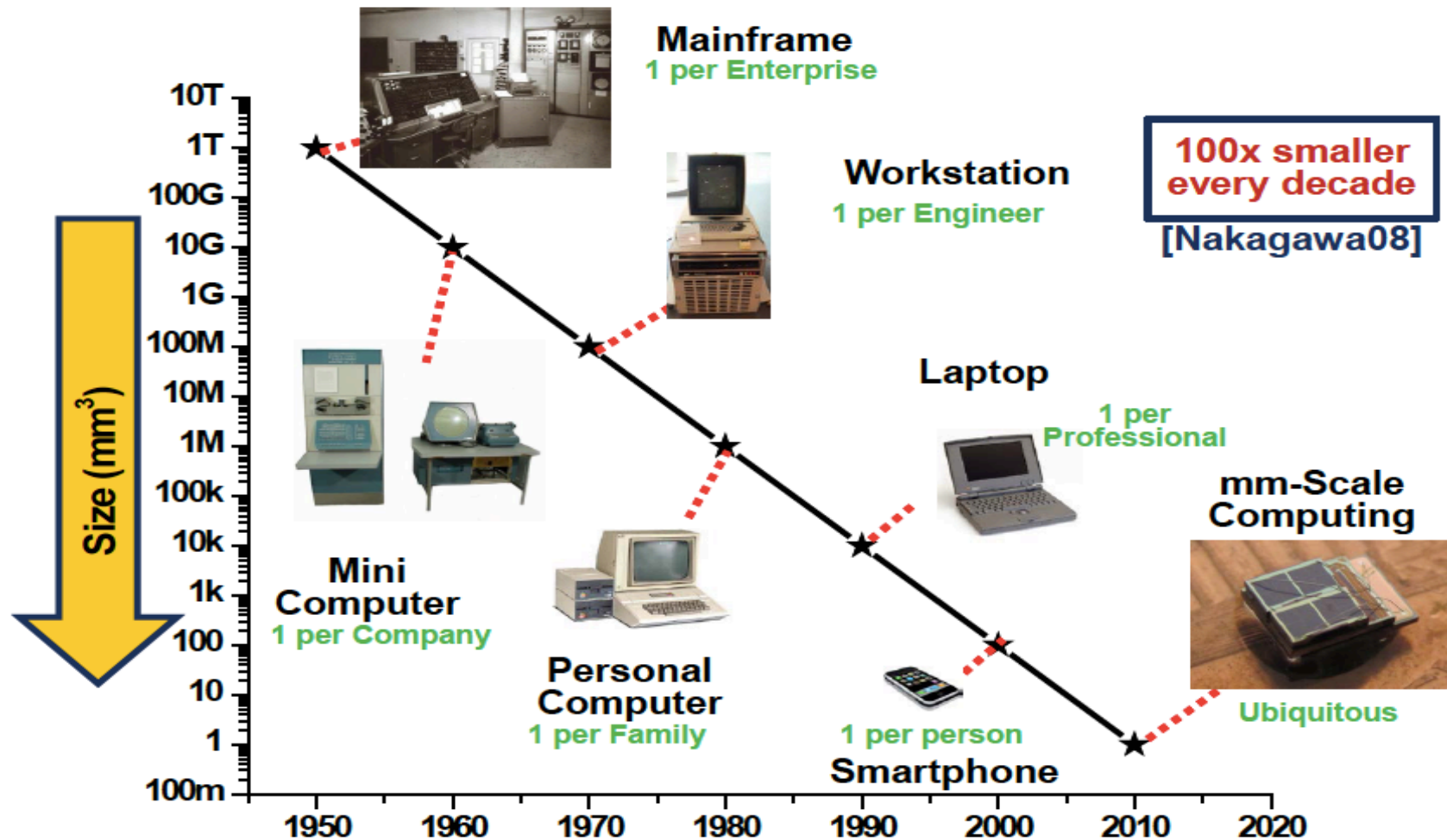
Moore's Law

	Year of introduction	Transistors
■ 4004	1971	2,250
■ 8008	1972	2,500
■ 8080	1974	5,000
■ 8086	1978	29,000
■ 286	1982	120,000
■ 386™	1985	275,000
■ 486™ DX	1989	1,180,000
■ Pentium®	1993	3,100,000
■ Pentium II	1997	7,500,000
■ Pentium III	1999	24,000,000
■ Pentium 4	2000	42,000,000

Moore's Law Ending?



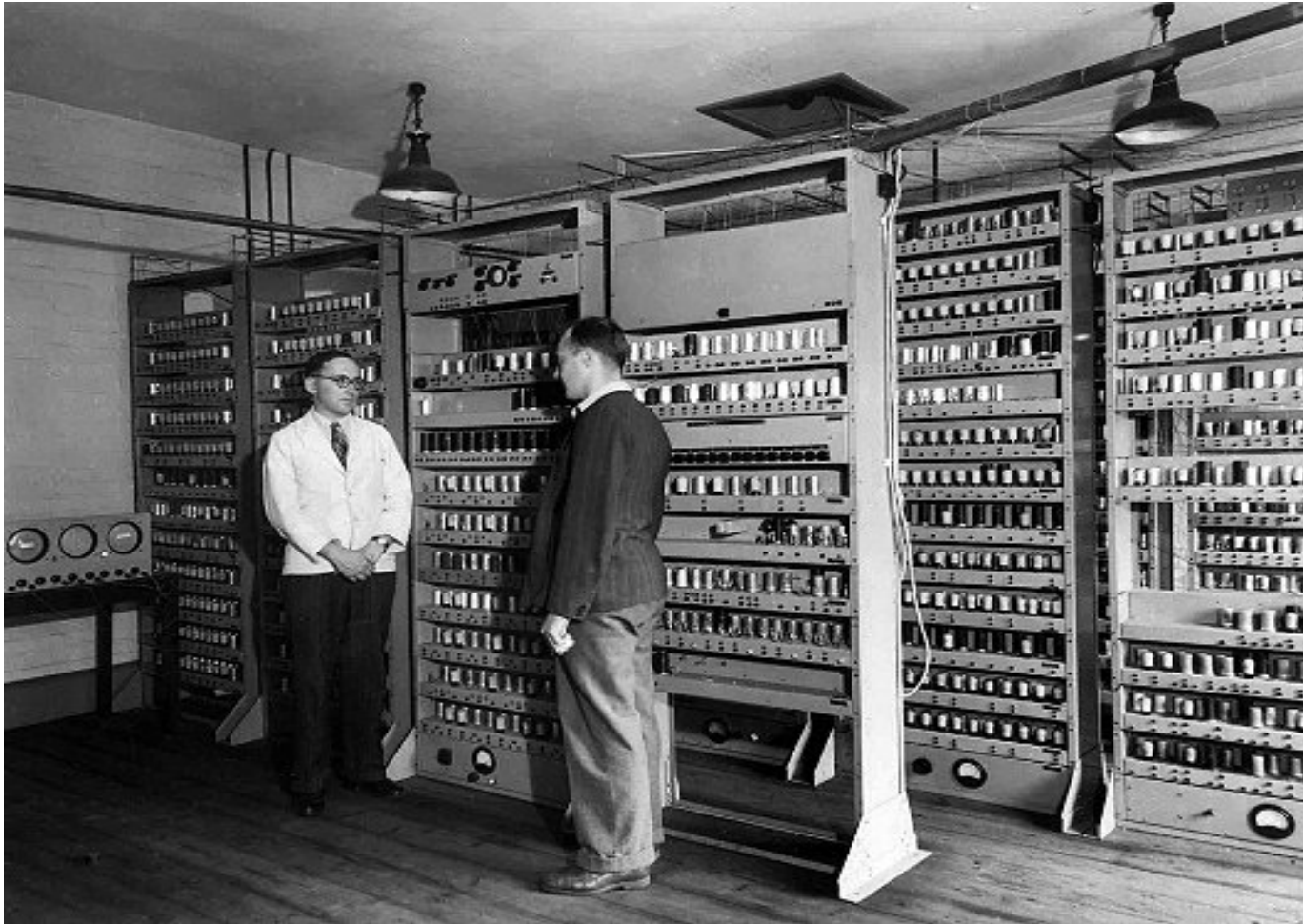
Bell's Law



Source: B Bell, "Bell's Law for the Birth and Death of Computer Classes", Comms of ACM, 2008

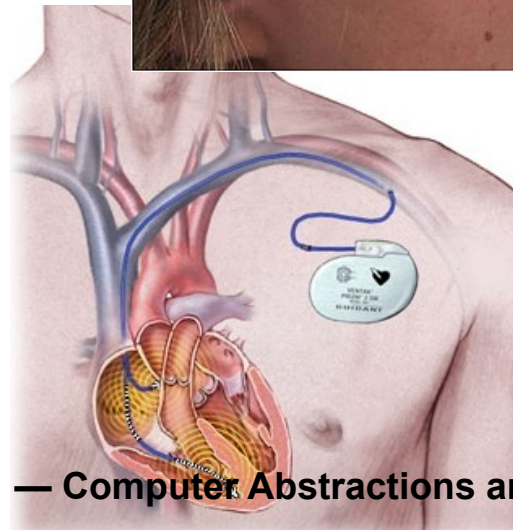
The 1st Generation Computer

- EDSAC, University of Cambridge, UK, 1949



Source: <http://www.computerhistory.org>

Computers Now





Data Center

Over three years, the power bill for a single server can be higher than the cost of the computer itself.

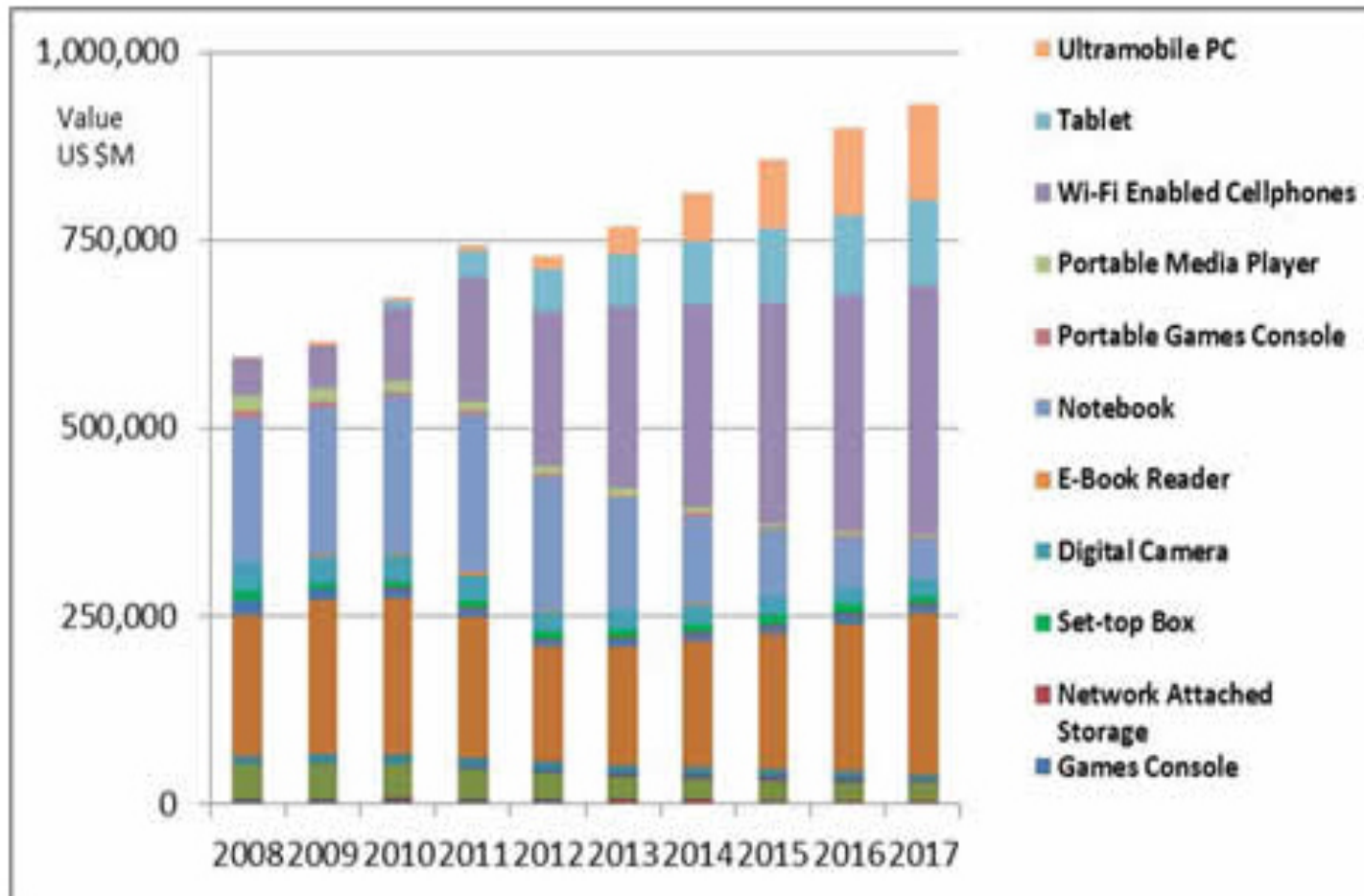
*Jeffrey W. Clarke
Vice Chairman of Operations & Technology
Sun Microsystems (now Oracle)*

One Google search consumes 0.3 watt-hours.

*Powering a Google search
The Official Google Blog*

Future Direction

GLOBAL CONSUMER ELECTRONICS DEVICE REVENUES 2008-2017



Source: <http://www.dvd-and-beyond.com/display-article.php?article=1891>

The Computer Revolution

- 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
- Computers are pervasive

Classes of Computers

- Desktop computers
 - General purpose, variety of software
 - Subject to cost/performance tradeoff
- Server computers
 - Network based
 - High capacity, performance, reliability
 - Range from small servers to building sized
- Embedded computers
 - Hidden as components of systems
 - Stringent power/performance/cost constraints

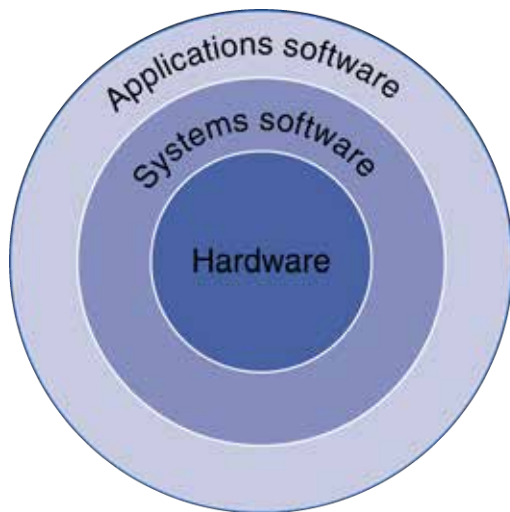
What You Will Learn

- How programs are translated into the machine language
 - And how the hardware executes them
- The hardware/software interface
- What determines program performance
 - And how it can be improved
- How hardware designers improve performance
- What is parallel processing

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



- 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

High-level
language
program
(in C)

```
swap(int v[], int k)
{int temp;
  temp = v[k];
  v[k] = v[k+1];
  v[k+1] = temp;
}
```

Compiler

Assembly
language
program
(for MIPS)

```
swap:
    muli $2, $5, 4
    add  $2, $4, $2
    lw   $15, 0($2)
    lw   $16, 4($2)
    sw   $16, 0($2)
    sw   $15, 4($2)
    jr   $31
```

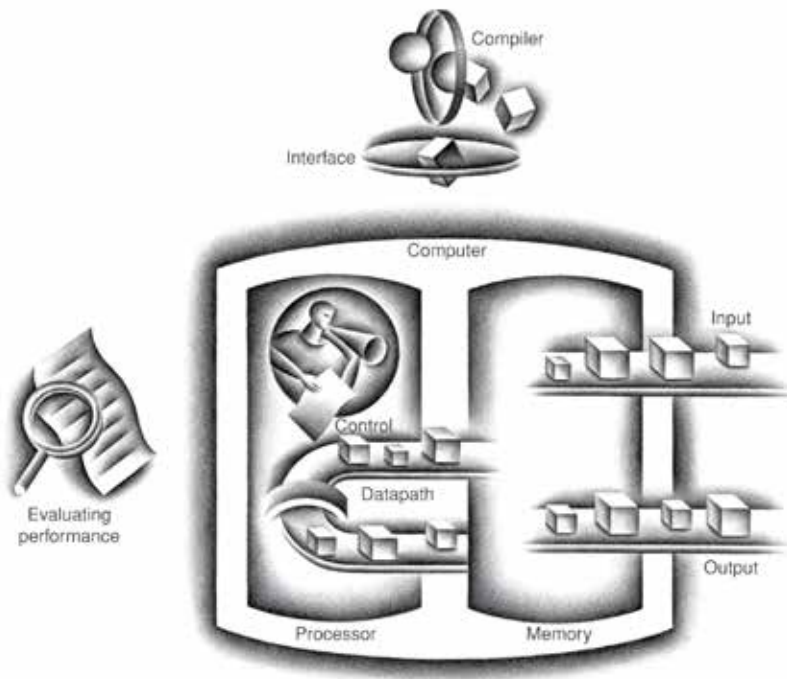
Assembler

Binary machine
language
program
(for MIPS)

```
000000001010000100000000000011000
00000000000110000001100000100001
10001100011000100000000000000000
10001100111100100000000000000100
10101100111100100000000000000000
10101100011000100000000000000100
0000001111100000000000000001000
```

Components of a Computer

The BIG Picture

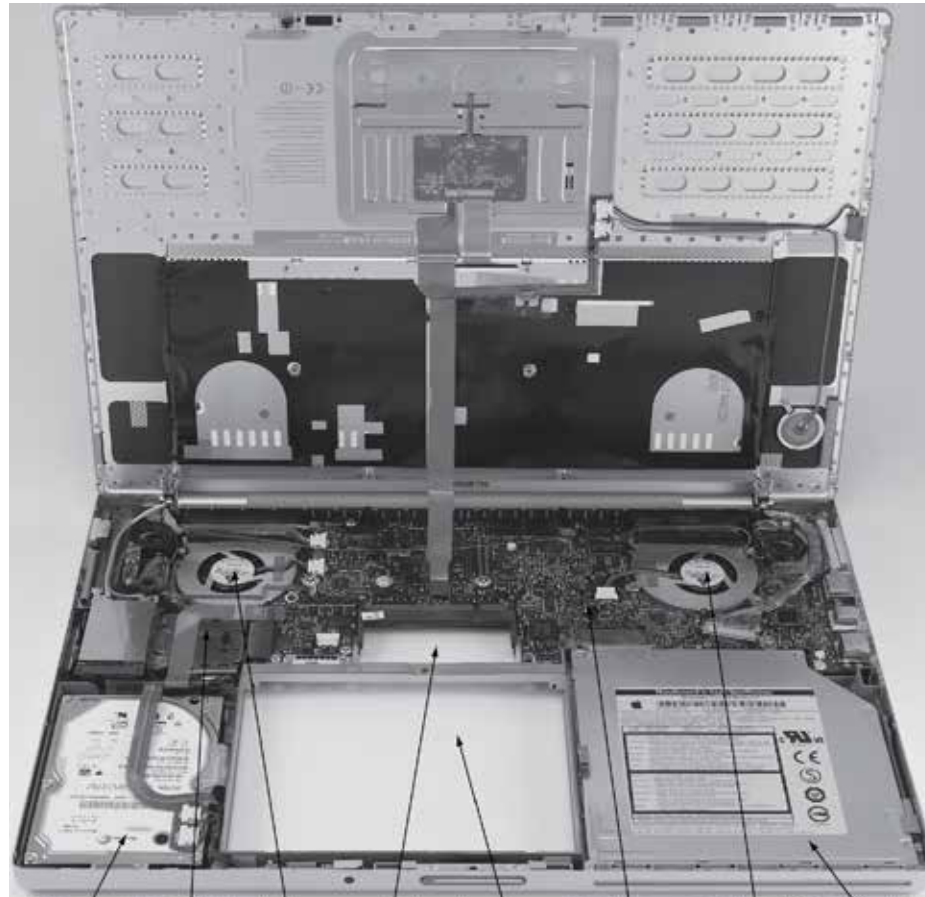


- Same components for all kinds of computer
 - Desktop, server, embedded
- Input/output includes
 - User-interface devices
 - Display, keyboard, mouse
 - Storage devices
 - Hard disk, CD/DVD, flash
 - Network adapters
 - For communicating with other computers

Anatomy of a Computer



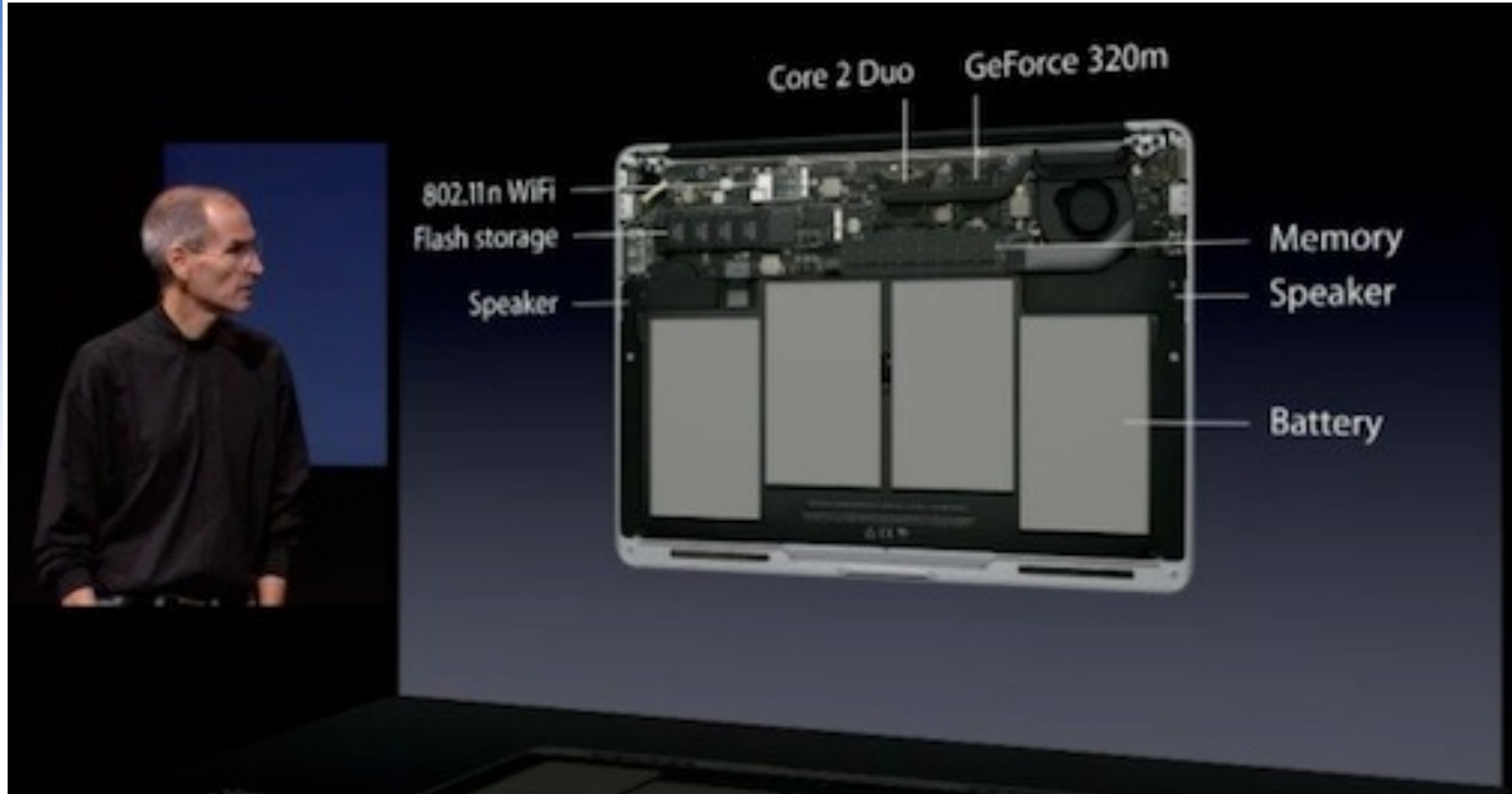
Opening the Box



Hard drive Processor Fan with cover Spot for memory DIMMs Spot for battery Motherboard Fan with cover DVD drive

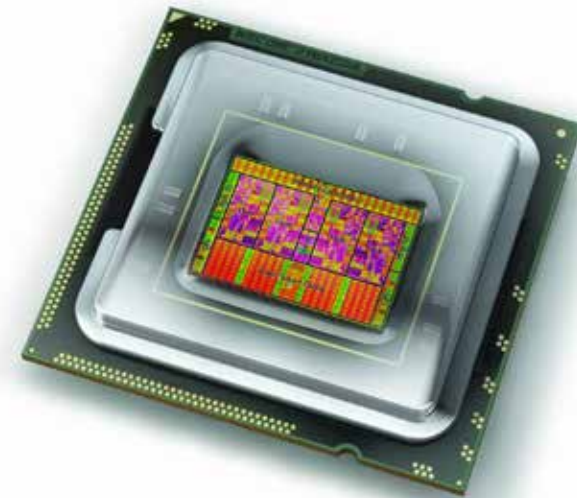
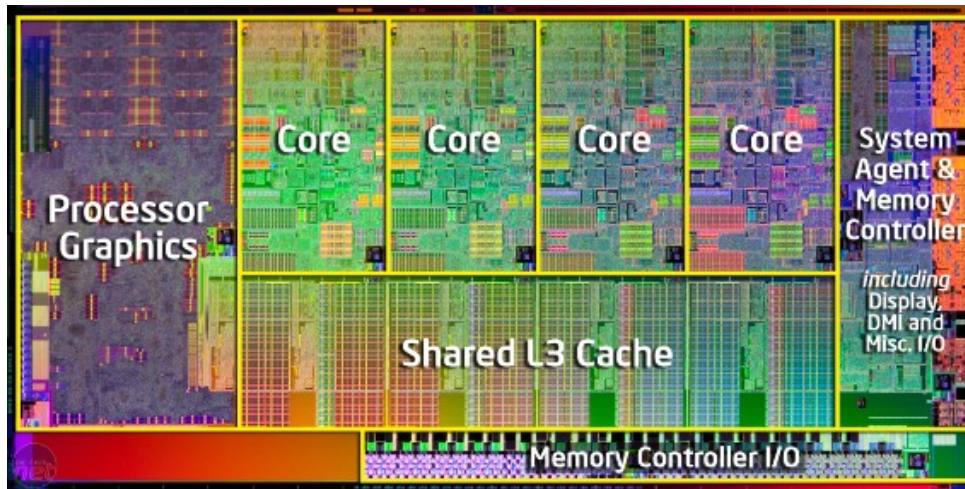


Opening the Box



Inside the Processor (CPU)

- Datapath: performs operations on data
- Control: sequences datapath, memory, ...
- Cache memory
 - Small fast SRAM memory for immediate access to data



Abstractions

The BIG Picture

- Abstraction helps us deal with complexity
 - Hide lower-level detail
- Instruction set architecture (ISA)
 - The hardware/software interface
- Application binary interface
 - The ISA plus system software interface
- Implementation
 - The details underlying and interface

A Safe Place for Data

- Volatile main memory
 - Loses instructions and data when power off
- Non-volatile secondary memory
 - Magnetic disk
 - Flash memory
 - Optical disk (CDROM, DVD)



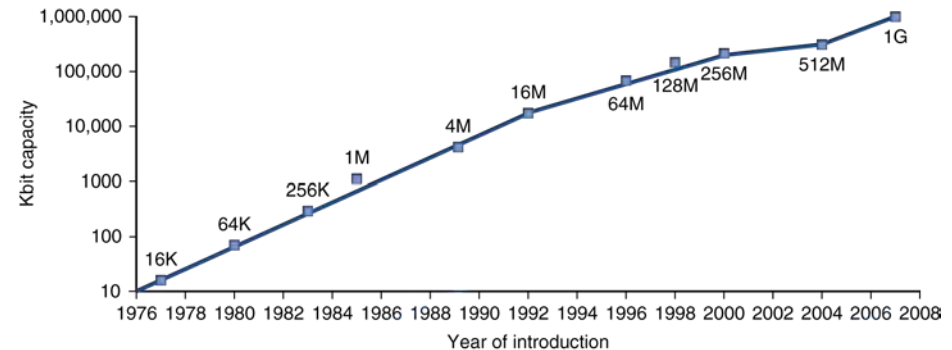
Networks

- Communication and resource sharing
- Local area network (LAN): Ethernet
 - Within a building
- Wide area network (WAN: the Internet
- Wireless network: WiFi, Bluetooth



Technology Trends

- Electronics technology continues to evolve
 - Increased capacity and performance
 - Reduced cost

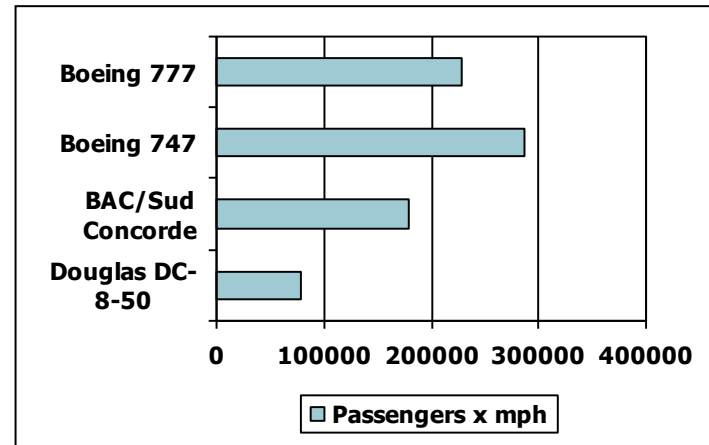
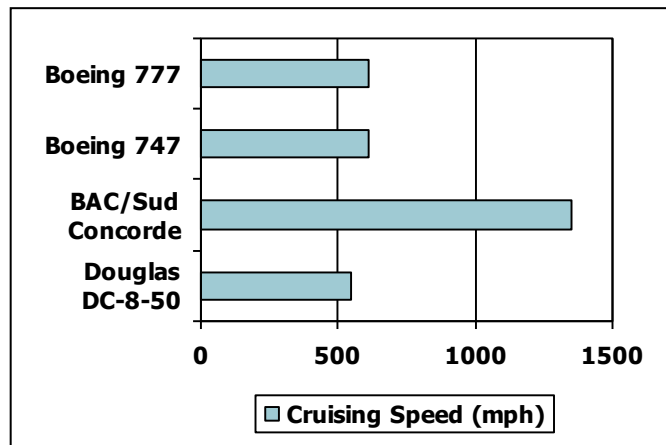
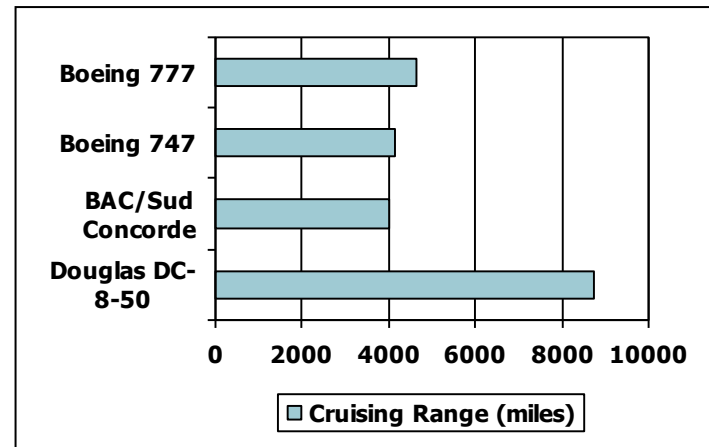
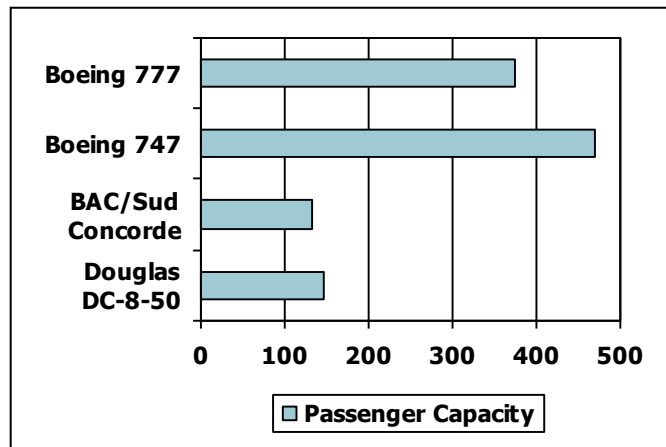


DRAM capacity

Year	Technology	Relative performance/cost
1951	Vacuum tube	1
1965	Transistor	35
1975	Integrated circuit (IC)	900
1995	Very large scale IC (VLSI)	2,400,000
2005	Ultra large scale IC	6,200,000,000

Defining Performance

- Which airplane has the best performance?



Response Time and Throughput

- Response time (execution time)
 - How long it takes to do a task
 - Important to computer users
- Throughput (bandwidth)
 - Total amount of work done per unit time
 - Important to server, data center
- Different performance metrics are needed to benchmark different systems.
- Single application is not sufficient to measure the performance of computers

Response Time vs. Throughput

- How are response time and throughput affected by
 - Replacing the processor with a faster version?
 - Adding more processors?
- We will focus on response time by now.

Relative Performance

- Define Performance = $1/(\text{Execution Time})$
- “X is n time faster than Y”

$$\begin{aligned} & \text{Performance}_X / \text{Performance}_Y \\ &= \text{Execution time}_Y / \text{Execution time}_X = n \end{aligned}$$

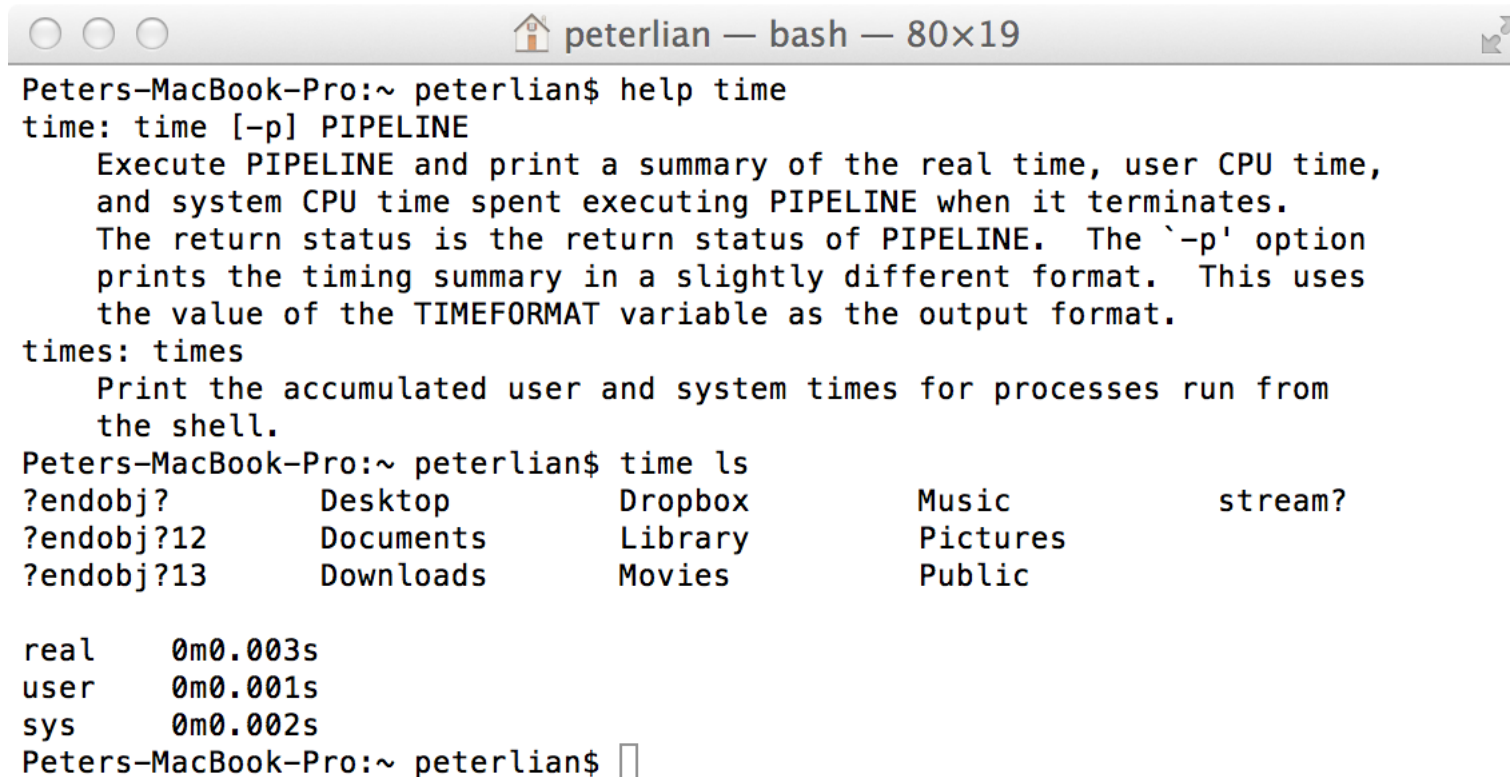
- Example: time taken to run a program
 - 10s on A, 15s on B
 - $\text{Execution Time}_B / \text{Execution Time}_A$
 $= 15\text{s} / 10\text{s} = 1.5$
 - So A is 1.5 times faster than B

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

Measuring Execution Time

- Unix command “time” can be used to determine the elapsed time and CPU time

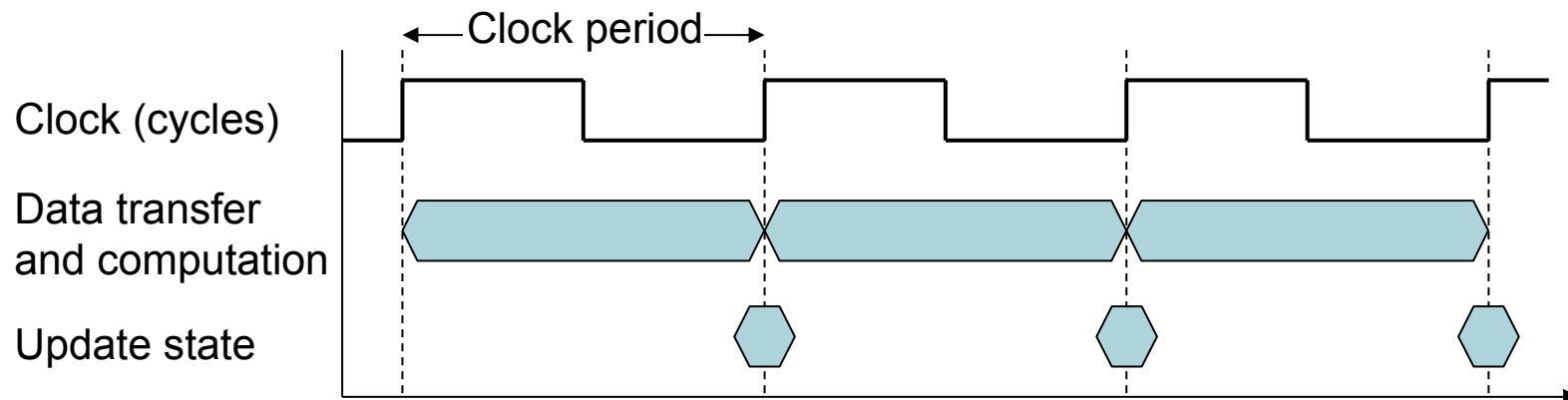


```
Peters-MacBook-Pro:~ peterlian$ help time
time: time [-p] PIPELINE
    Execute PIPELINE and print a summary of the real time, user CPU time,
    and system CPU time spent executing PIPELINE when it terminates.
    The return status is the return status of PIPELINE. The '-p' option
    prints the timing summary in a slightly different format. This uses
    the value of the TIMEFORMAT variable as the output format.
times: times
    Print the accumulated user and system times for processes run from
    the shell.
Peters-MacBook-Pro:~ peterlian$ time ls
?endobj?      Desktop      Dropbox      Music        stream?
?endobj?12    Documents   Library      Pictures
?endobj?13    Downloads   Movies       Public

real    0m0.003s
user    0m0.001s
sys     0m0.002s
Peters-MacBook-Pro:~ peterlian$
```

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
 - Aim for 6s CPU time
 - Can do faster clock, but causes $1.2 \times$ clock cycles of A
- How fast must Computer B clock be?

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

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

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

Instruction Performance

Clock Cycles = Instruction Count \times Ave Cycles per Instruction

CPU Time = Instruction Count \times CPI \times Clock Cycle Time

$$= \frac{\text{Instruction Count} \times \text{CPI}}{\text{Clock Rate}}$$

- Instruction Count: no of instruction for a program
 - Determined by program, Instruction Set Architecture (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

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

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

$$= 1 \times 2.0 \times 250\text{ps} = 1 \times 500\text{ps}$$

A is faster...

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

$$= 1 \times 1.2 \times 500\text{ps} = 1 \times 600\text{ps}$$

- By how 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 program using instructions in classes A, B, C

Class	A	B	C
CPI for class	1	2	3
IC in program 1	2	1	2
IC in program 2	4	1	1

- Program 1: IC = 5
 - Clock Cycles
 $= 2 \times 1 + 1 \times 2 + 2 \times 3$
 $= 10$
 - Avg. CPI = $10/5 = 2.0$

- Program 2: IC = 6
 - Clock Cycles
 $= 4 \times 1 + 1 \times 2 + 1 \times 3$
 $= 9$
 - Avg. CPI = $9/6 = 1.5$

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

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)

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

SPEC CPU Benchmark

- 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
 - Two benchmark suites: 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×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

SPEC Power Benchmark

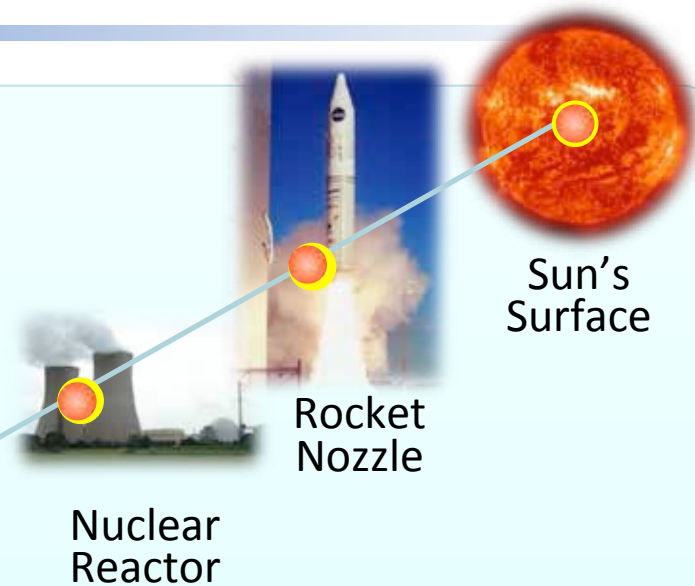
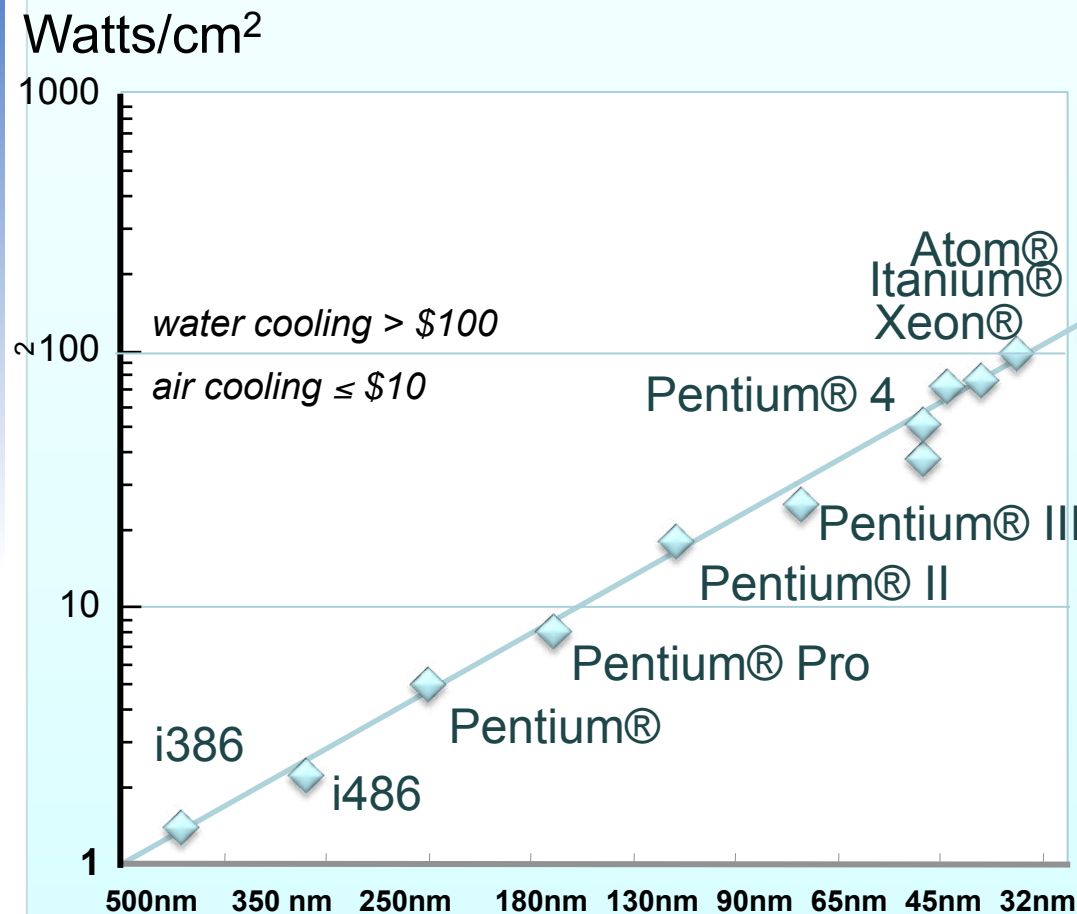
- Power consumption of server at different workload levels
 - Performance: ssj_ops/sec
 - Power: Watts (Joules/sec)

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

SPECpower_ssj2008 for X4

Target Load %	Performance (ssj_ops/sec)	Average Power (Watts)
100%	231,867	295
90%	211,282	286
80%	185,803	275
70%	163,427	265
60%	140,160	256
50%	118,324	246
40%	92,035	233
30%	70,500	222
20%	47,126	206
10%	23,066	180
0%	0	141
Overall sum	1,283,590	2,605
$\Sigma \text{ssj_ops} / \Sigma \text{power}$		493

Power Trends



190W at 65-nm \Rightarrow 135 W/cm² !

$P = VI \Rightarrow 190W$ at $1.1V = 170A$!

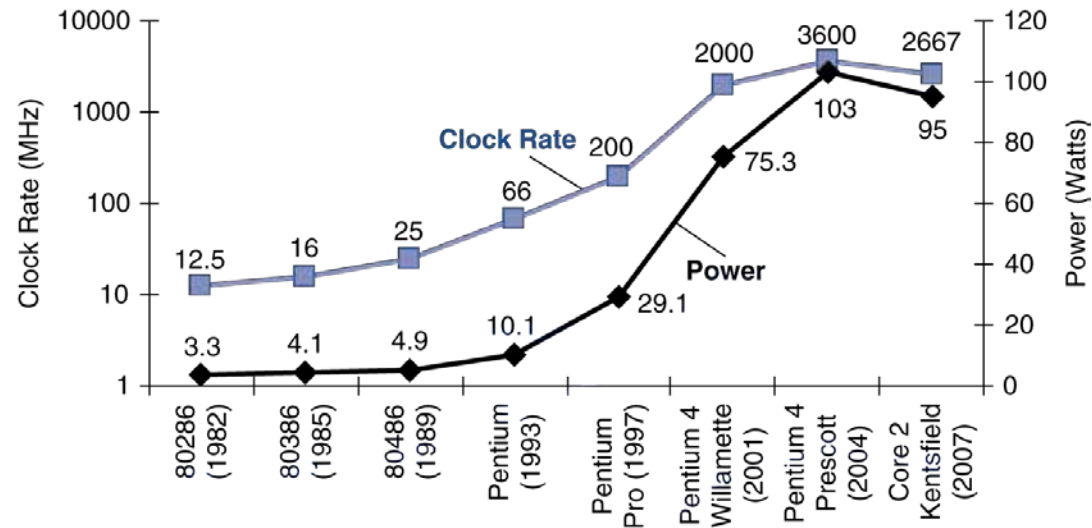
The Thermal Crisis

***What happens
when the
CPU cooler is
removed?***



www.tomshardware.de
www.tomshardware.com

Power Trends



- In CMOS IC technology

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

×30

5V → 1V

×1000

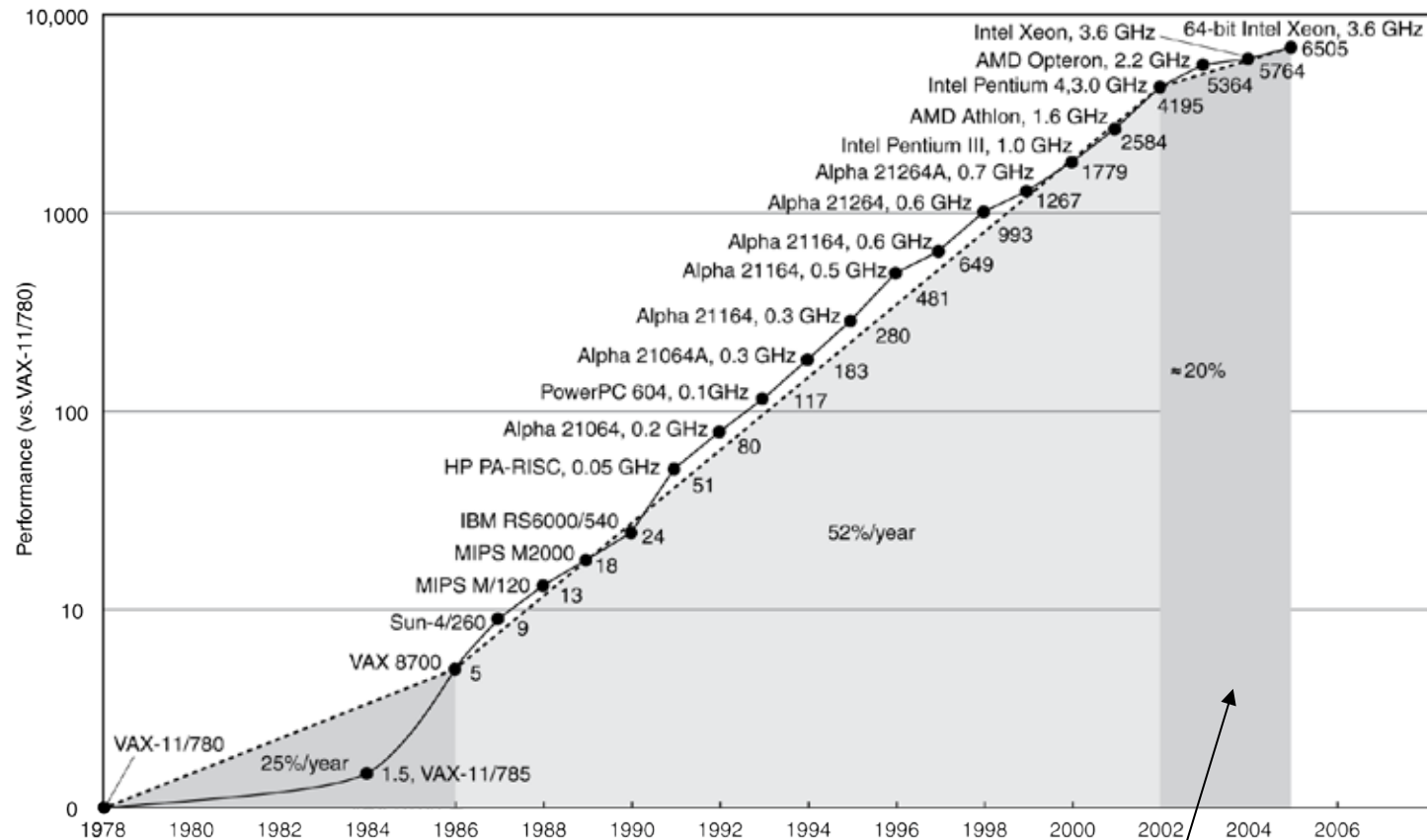
Reducing Power

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

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

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

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

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?

$$20 = \frac{80}{n} + 20 \quad \text{■ Can't be done!}$$

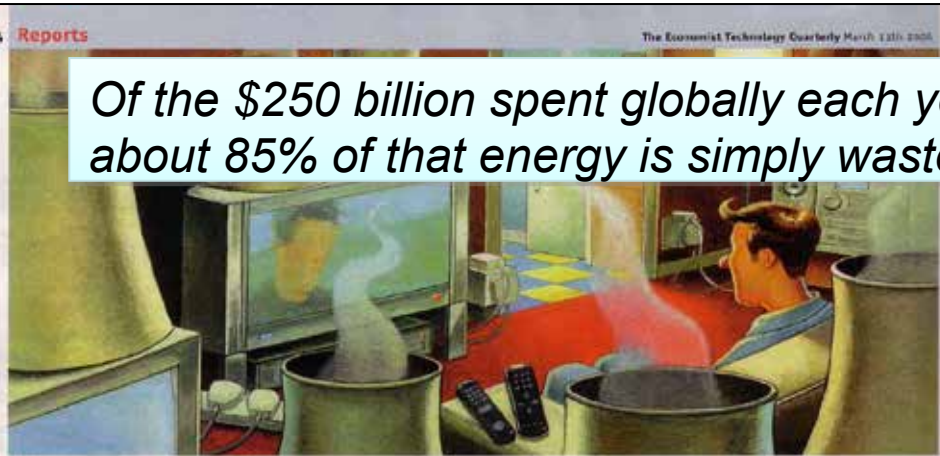
- Corollary: make the common case fast

Fallacy: Low Power at Idle

- Look back at X4 power benchmark
 - At 100% load: 295W
 - At 50% load: 246W (83%)
 - At 10% load: 180W (61%)
- 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 load

Importance of Standby Power

Of the \$250 billion spent globally each year powering computers, about 85% of that energy is simply wasted idling.

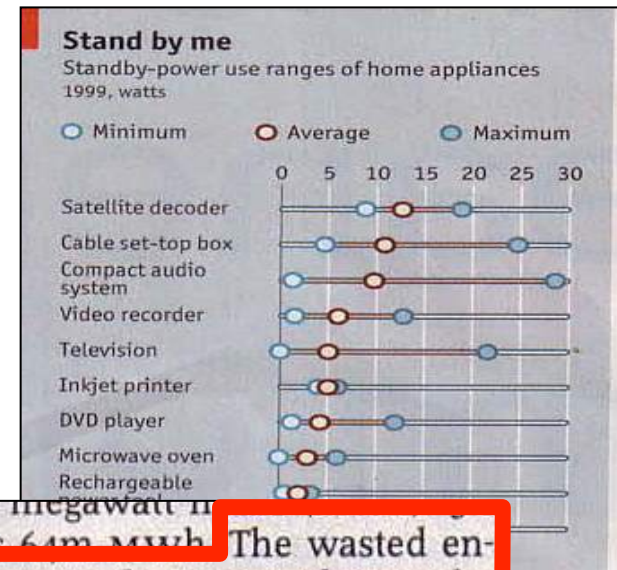


Pulling the plug on standby power

some cases. That same year, a similar study in France found that standby power accounted for 7% of total residential consumption. Further studies have since come to similar conclusions in other developed countries, including the Netherlands, Australia and Japan. Some estimates put the proportion of consumption due to standby power as high as 13%.

some cases. That same year, a similar study in France found that standby power accounted for 7% of total residential consumption. Further studies have since come to similar conclusions in other developed countries, including the Netherlands, Australia and Japan. Some estimates put the proportion of consumption due to standby power as high as 13%.

1.29 billion megawatt hours of which is 64m MWh. The wasted energy, in other words, is equivalent to the output of 18 typical power stations.



Source: *Economist*, August 11, 2010

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

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

- CPI varies between programs on a given CPU

Concluding Remarks

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