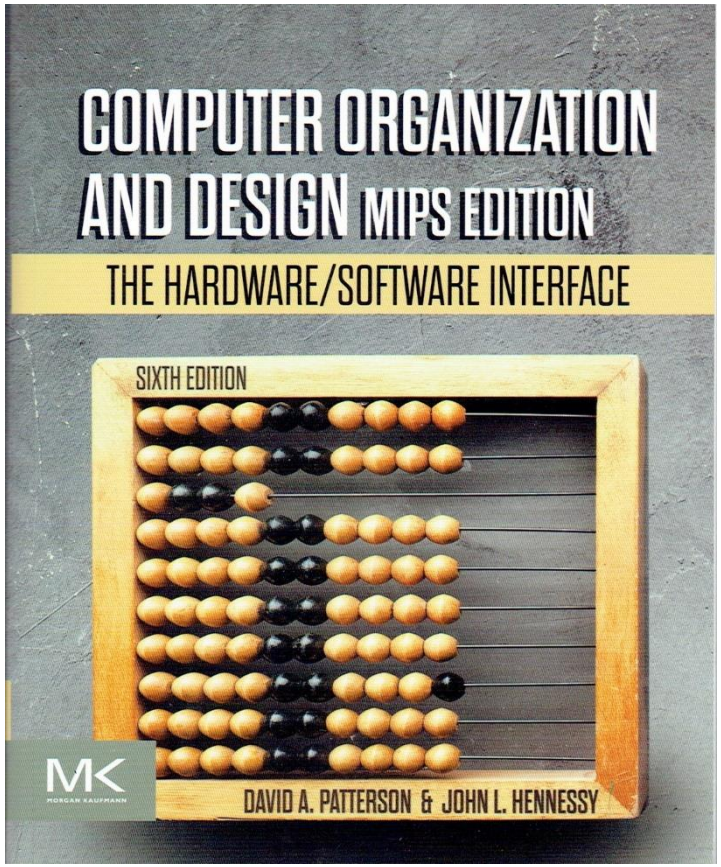
Chap. 1
Computer Abstractions and Technology

Slides

臺大電機系
吳安宇教授

2025/2/17 v1

Main textbook (6th Ed.)



(Main textbook， 算盤書、CA bible)
David A. Patterson, and John L. Hennessy, "Computer Organization and Design - The Hardware/Software Interface", 6th Edition (MIPS version, 東華書局代理**), Morgan Kaufman Publishers, Inc., 2021.**

5th edition 有 ARM、RISC-V version

Outline

- 1.1 Introduction
- 1.2 Below Your Program
- 1.3 Under the Covers
- 1.4 Performance
- 1.5 The Power Wall
- 1.6 The Sea Change: The Switch from Uniprocessors to Multiprocessors
- 1.7 Benchmarking
- 1.8 Fallacies and Pitfalls
- 1.9 Concluding Remarks

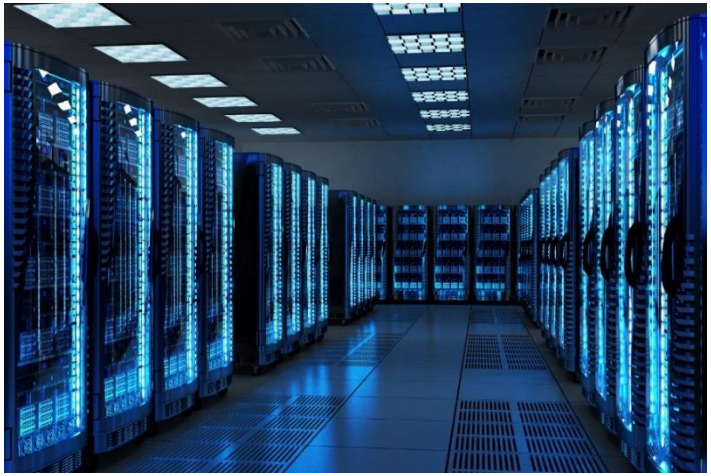
Introduction

- In the recent past, the following applications were "Computer science fictions" :
  - Automatic Teller Machines (ATM): In 1950, it costs $500,000 with size of a car
  - Computers in Automobiles: Safety, Gas efficiency, Electronics (audio, GPS, Navigation)
  - Cell/Mobile phones
  - Laptop/notebook computers: Reduction in size
  - Human genome project: Huge computing power
  - Word Wide Web: Internet + Low-cost PC
  - Search Engine

Different Types of Computers

- Desktop computer (Personal computer, PC)
  - A computer designed for use by an individual, usually incorporating a graphics display, keyboard, and mouse.
- Server (within an institute like company/school)
  - A large computer used for running larger programs for multiple users often simultaneously and typically accessed only via a network
- Supercomputer (c.f., Quantum Computer)
  - A class of computer with the highest performance and cost; they are configured as servers and typically cost millions of dollars (for scientific or military purposes)

Server (server farm) & Terminals

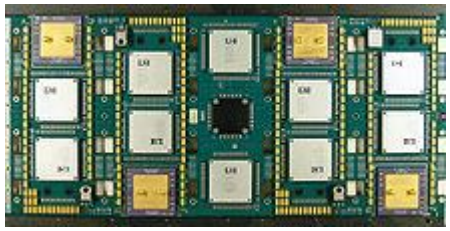


伺服器農場（Server farm）或伺服器叢集（Server cluster）是電腦伺服器的一個集合-通常被一個組織維護，所提供的伺服器功能方面遠遠超過單個伺服器的能力

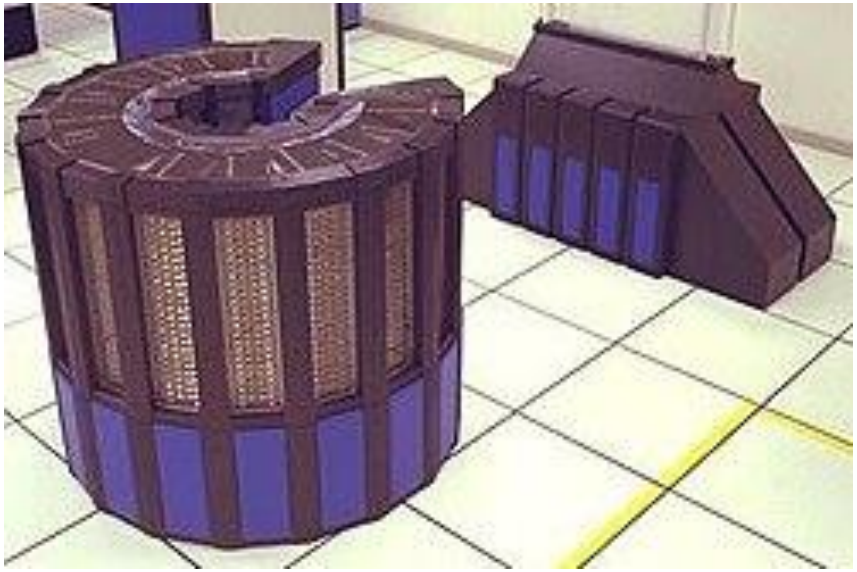
- https://zh.wikipedia.org/wiki/%E6%9C%8D%E5%8A%A1%E5%99%A8%E5%86%9C%E5%9C%BA

Cray Supercomputer

Cray-2 supercomputer



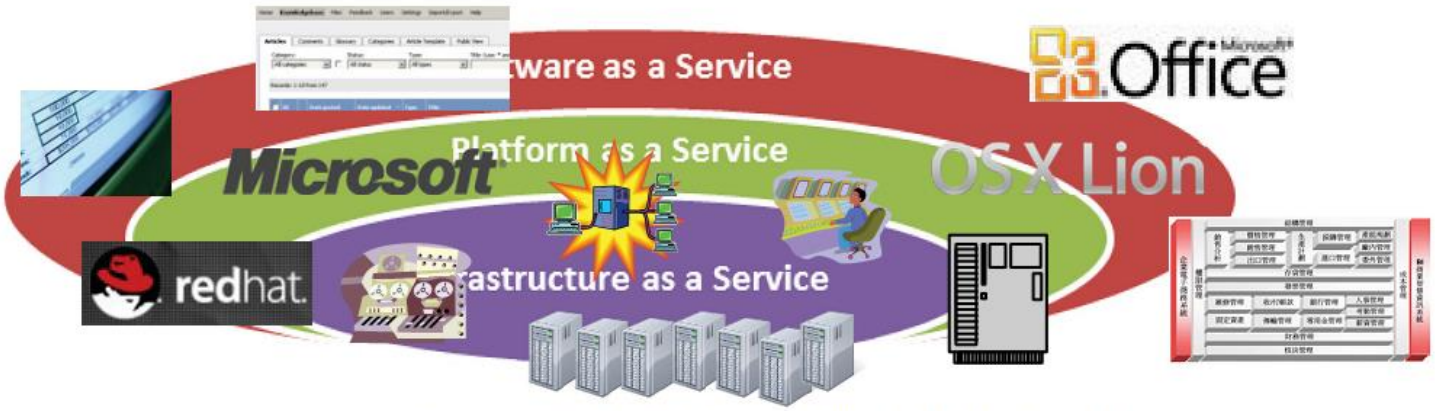
Cray T3E processor board



https://en.wikipedia.org/wiki/Cray

Cloud Computing system

- Wiki:「雲端運算」可以 在「體系架構」有三個層次：**Infrastructure as a service (IaaS)**、**Platform as a service (PaaS)**，以及 **Software as a service (SaaS)**
- IaaS 稱為「提供基礎架構的雲端服務」，將 PaaS 稱為「平台即服務」，而 SaaS 則是「軟體即服務」。
- E.g., Microsoft: Azure, Amazon: AWS, Dropbox (??)



圖一 雲端運算三個層次的簡易示意

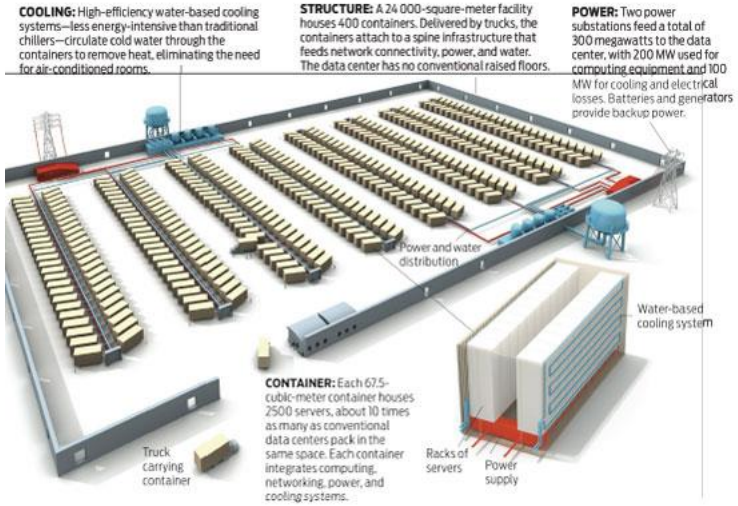
Source: https://www.dgbas.gov.tw/public/Data/112301592471.pdf

Cloud Computing system

- Container computer (in cloud center)

http://www.developertutorials.com/the-technological-underpinnings-of-the-cloud/

- Cloud center



Different Types of Computers (cont'd)

- **Embedded computer/devices**
  - A computer inside another device used for running one pre-determined application, e.g., cellular phone, video games, digital TVs, set-top boxes, personal digital assistants (PDAs), computer inside automobiles/space shuttles, Point-of-Sale (POS) machine

- **Mobile Phones**
  - A computer system with communications (3G/4G/5G), high-pixel camera, multi-core computing platform

Units in Data Size

Decimal term	Abbreviation	Value	Binary term	Abbreviation	Value	% Larger
kilobyte	KB	1000^1	kibibyte	KiB	2^{10}	2%
megabyte	MB	1000^2	mebibyte	MiB	2^{20}	5%
gigabyte	GB	1000^3	gibibyte	GiB	2^{30}	7%
terabyte	TB	1000^4	tebibyte	TiB	2^{40}	10%
petabyte	PB	1000^5	pebibyte	PiB	2^{50}	13%
exabyte	EB	1000^6	exbibyte	EiB	2^{60}	15%
zettabyte	ZB	1000^7	zebibyte	ZiB	2^{70}	18%
yottabyte	YB	1000^8	yobibyte	YiB	2^{80}	21%
ronnabyte	RB	1000^9	robibyte	RiB	2^{90}	24%
queccabyte	QB	1000^{10}	quebibyte	QiB	2^{100}	27%

FIGURE 1.1 The 2^X vs. 10^Y bytes ambiguity was resolved by adding a binary notation for all the common size terms. In the last column we note how much larger the binary term is than its corresponding decimal term, which is compounded as we head down the chart. These prefixes work for bits as well as bytes, so *gigabit* (Gb) is 10^9 bits while *gibibits* (Gib) is 2^{30} bits. The society that runs the metric system created the decimal prefixes, with the last two proposed only in 2019 in anticipation of the global capacity of storage systems. All the names are derived from the entymology in Latin of the powers of 1000 that they represent.

The Processor Market (PC and PostPC Era)

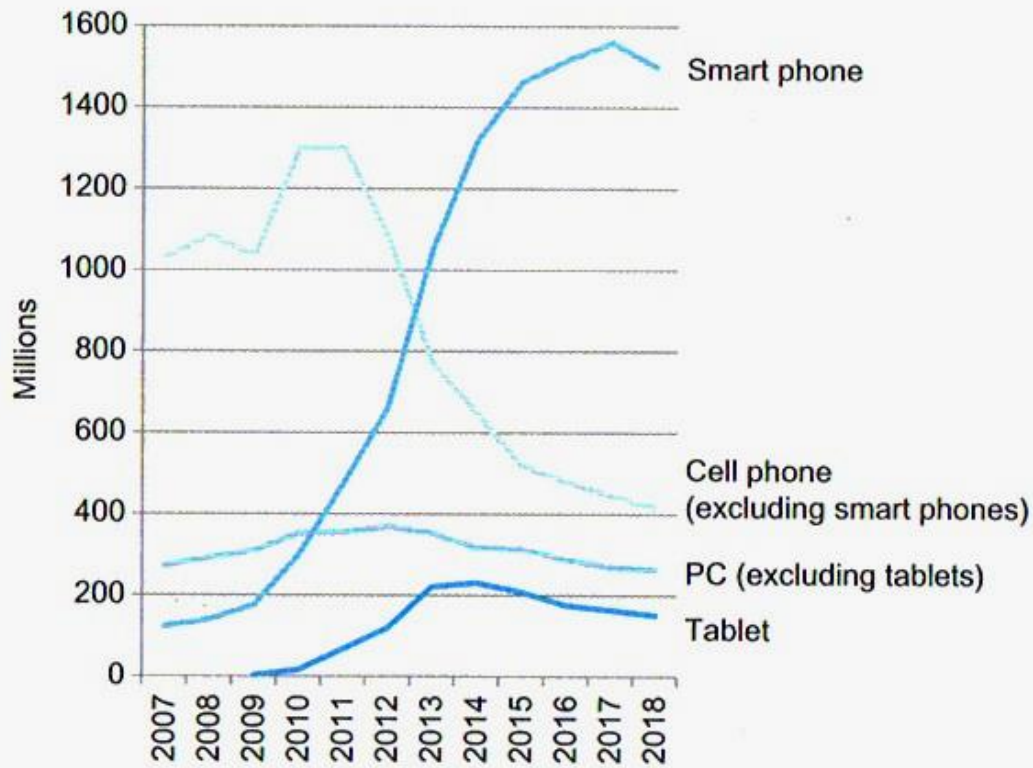
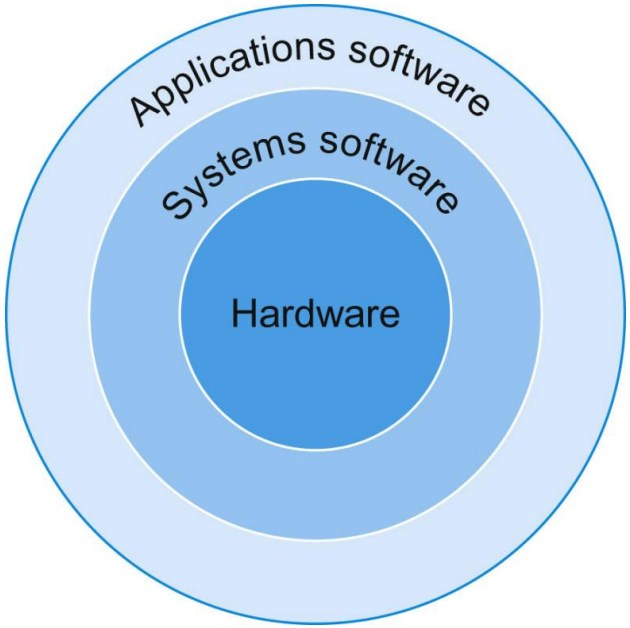


FIGURE 1.2 The number manufactured per year of tablets and smart phones, which reflect the PostPC era, versus personal computers and traditional cell phones. Smart phones represent the recent growth in the cell phone industry, and they passed PCs in 2011. PCs, tablets, and traditional cell phone categories are declining. The peak volume years text are 2011 for cell phones, 2013 for PCs, and 2014 for tablets. PCs fell from 20% of total units shipped in 2007 to 10% in 2018.

Outline

- 1.1 Introduction
- 1.2 Below Your Program
- 1.3 Under the Covers
- 1.4 Performance
- 1.5 The Power Wall
- 1.6 The Sea Change: The Switch from Uniprocessors to Multiprocessors
- 1.7 Benchmarking
- 1.8 Fallacies and Pitfalls
- 1.9 Concluding Remarks

Below your program



- A simplified view of hardware and software as hierarchical layers, shown as concentric circles with **hardware in the center** and applications software outermost
- Applications software: Matlab, Firefox browser, APPs on mobile phones
- System software: **Assembler, Compiler, OS** (Window 10, Linux, Android, iOS, etc.)
- **Hardware:** CPU + Memory + SSD/HD + I/O (Desktop PC/Notebook)

From high-level language to the language of hardware

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:
      multi $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)

```
00000000101000100000000100011000
00000000100000100001000001000001
10001101111000100000000000000000
10001110000100100000000000000100
10101110000100100000000000000000
10101101111000100000000000000100
00000011111000000000000000001000
```

High-level C/C++/Python programs

→ Compiled (by Compiler tool) into assembly language (text-based CPU instructions)

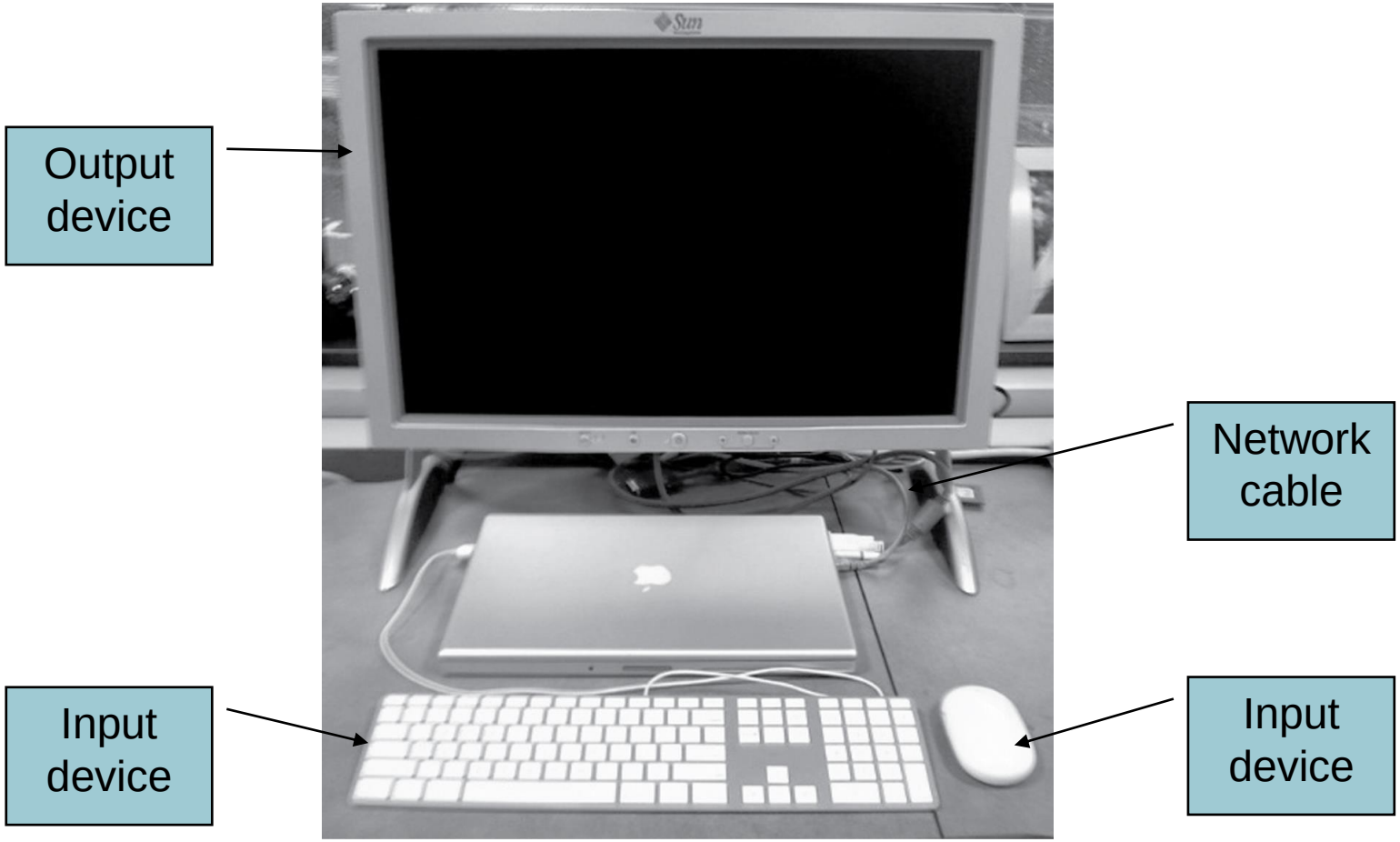
→ and then translate into (by Assembler tool)

Binary machine language (executable binary codes for CPU)

Outline

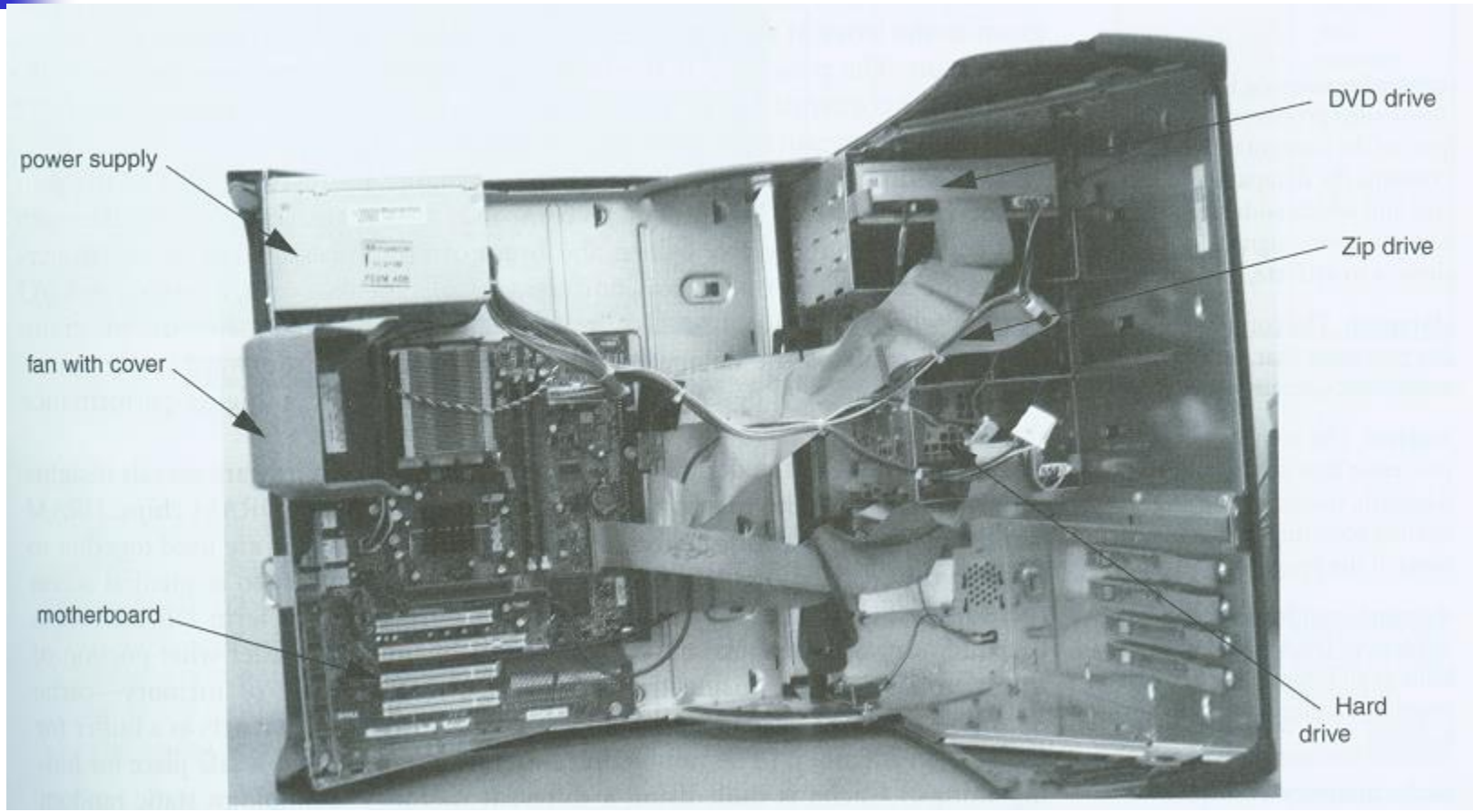
- 1.1 Introduction
- 1.2 Below Your Program
- 1.3 Under the Covers
- 1.4 Performance
- 1.5 The Power Wall
- 1.6 The Sea Change: The Switch from Uniprocessors to Multiprocessors
- 1.7 Benchmarking
- 1.8 Fallacies and Pitfalls
- 1.9 Concluding Remarks

Anatomy of a Computer



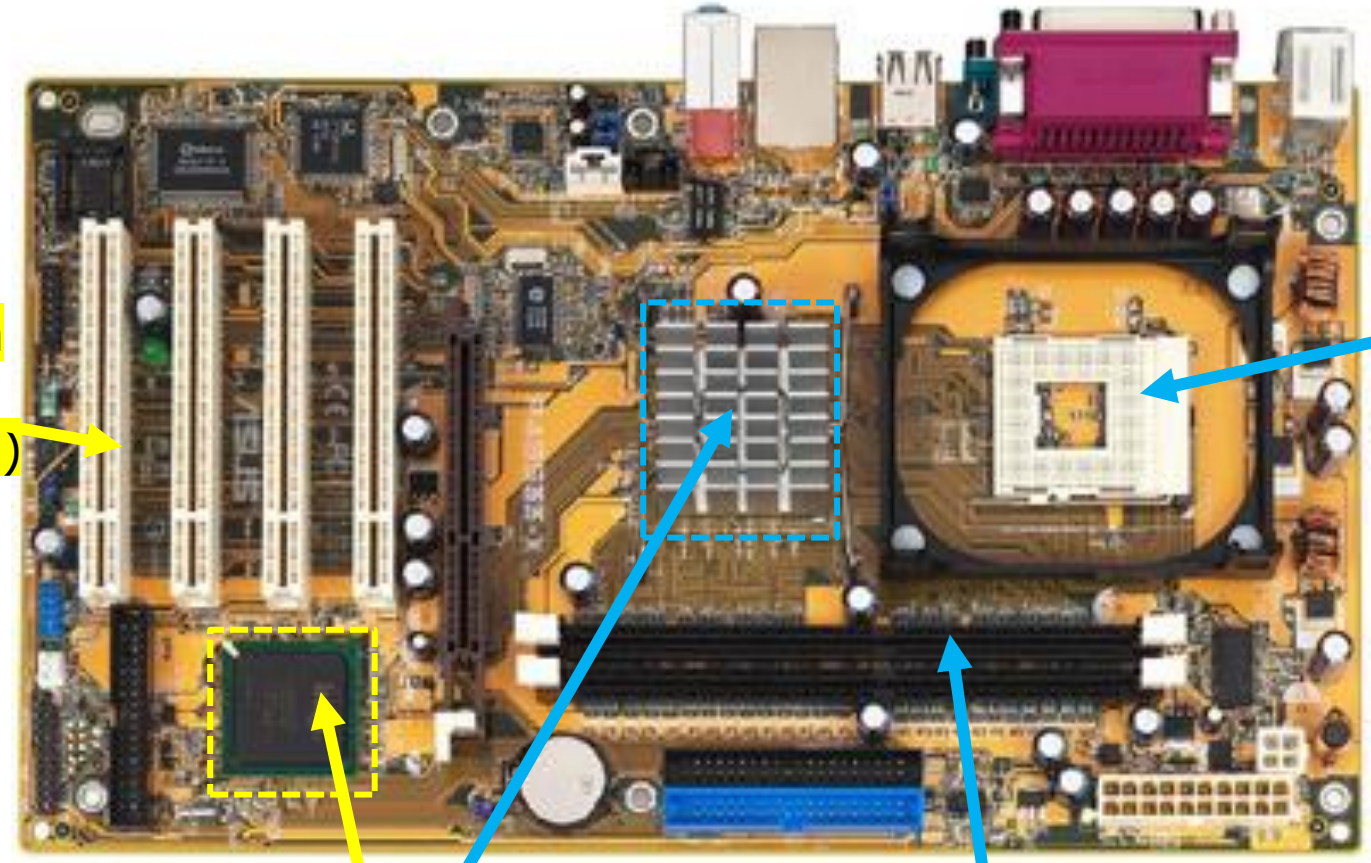
A desktop computer (PC)

Opening the box

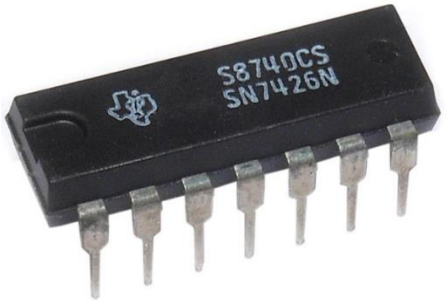


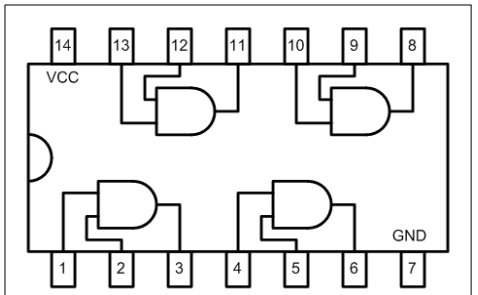
Inside the Personal Computer (PC)

Close-up of PC motherboard

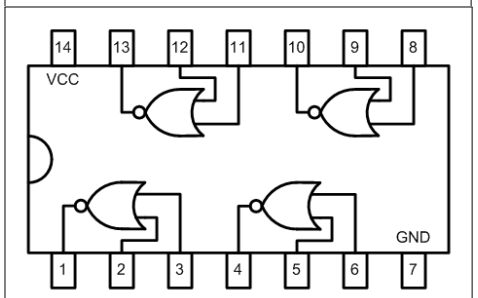


Review: Small-scale Logic ICs

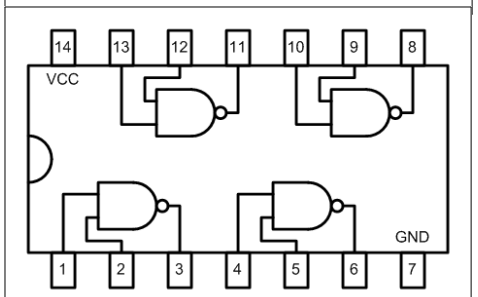




7408 Quad 2 Input AND



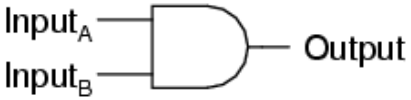
7402 Quad 2 Input NOR



7400 Quad 2 Input NAND

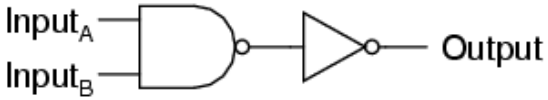
- More expensive in Positive Logic gates

AND gate



A	B	Output
0	0	0
0	1	0
1	0	0
1	1	1

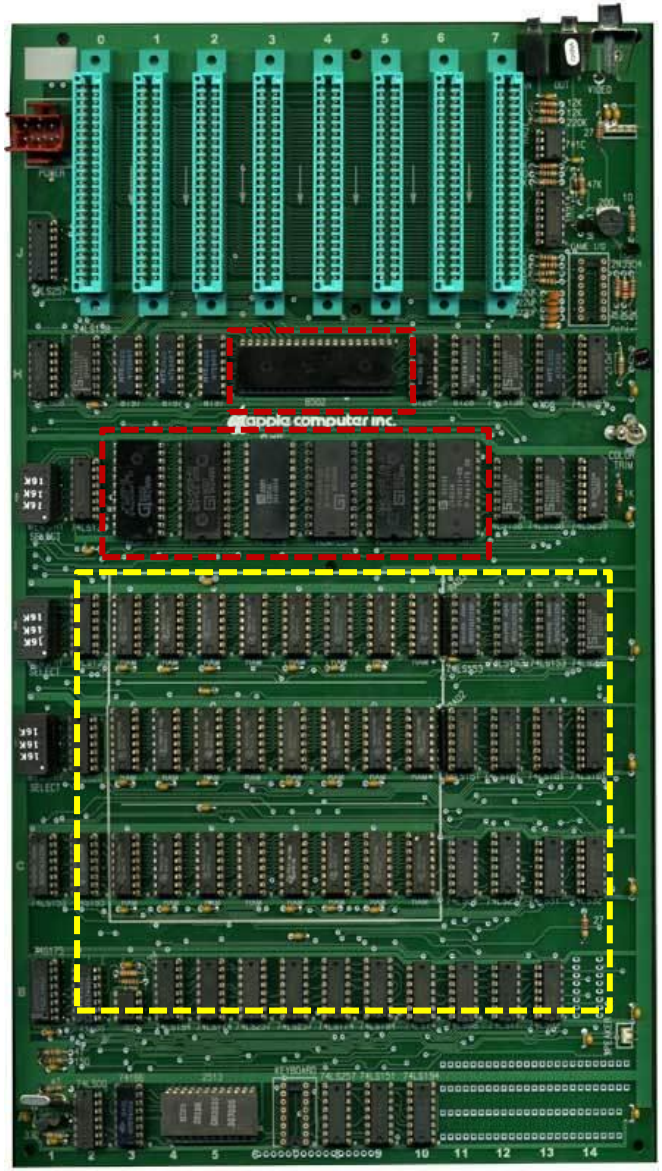
Equivalent circuit



6 TXs

TTLs on PCB board of APPLE II (v.s. Chipset)

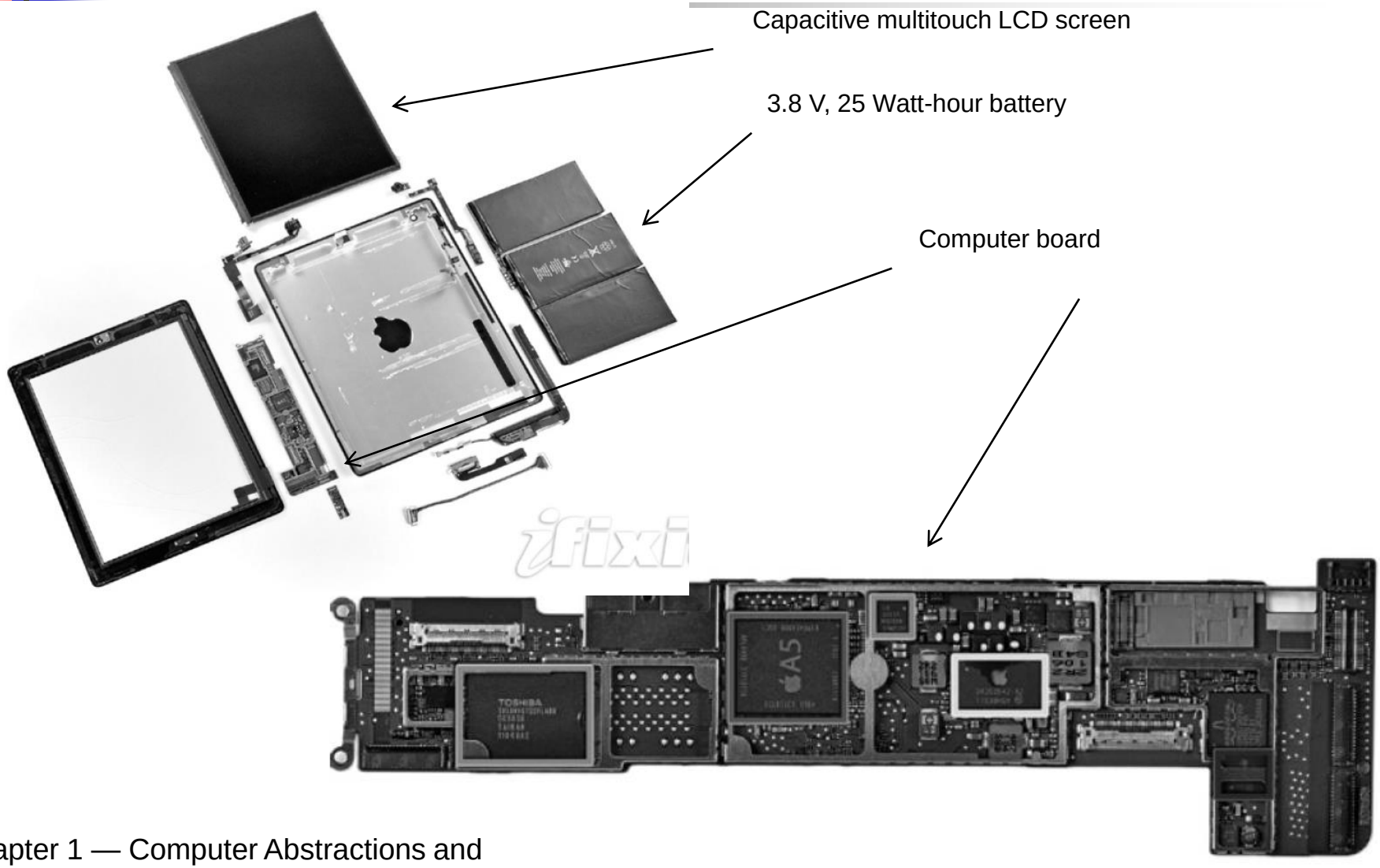
→ Convert into Chip Sets (North Bridge and South Bridge)



Computer components: Chipsets (North/South bridges chips)

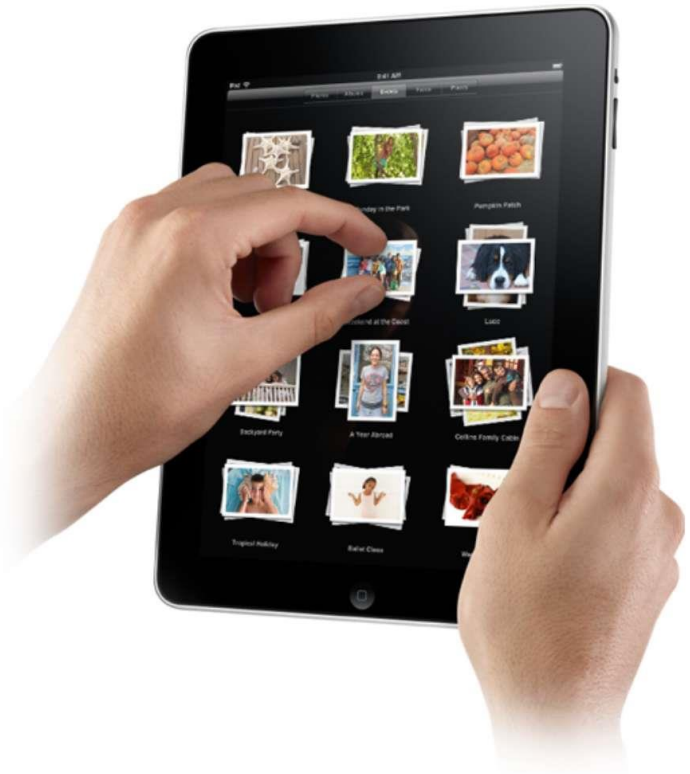
- Intel® 3 Series Chipsets
  Intel® 4 Series Chipset
  Intel® 5 Series Chipset
  Intel® 810 Chipset Family
  Intel® 815 Chipset Family
  Intel® 820 Chipset Family
  Intel® 830 Chipset
  Intel® 845 Chipset Family
  Intel® 848P Chipset
  Intel® 850 Chipset Family
  Intel® 852/855 Chipset Family
  Intel® 865 Chipset Family
  Intel® 875P Chipset
  Intel® 910 Express Chipset Family
  Intel® 915 Express Chipset Family
  Intel® 925X Express Chipset Family
  Intel® 945 Express Chipset Family
- Intel® 945 Express Chipset Family
  Intel® 946 Express Chipset Family
  Intel® 955X Express Chipset
  Intel® 965 Express Chipset Family
  Intel® 975X Express Chipset
  Intel® X58 Express Chipset
  Mobile Intel® 4 Series Express Chipset Family
  Mobile Intel® 5 Series Chipset
  Mobile Intel® 910GML Express Chipset
  Mobile Intel® 915 Express Chipset Family
  Mobile Intel® 945 Express Chipset Family
  Mobile Intel® 965 Express Chipset Family

Opening the Box of An iPAD



Touchscreen

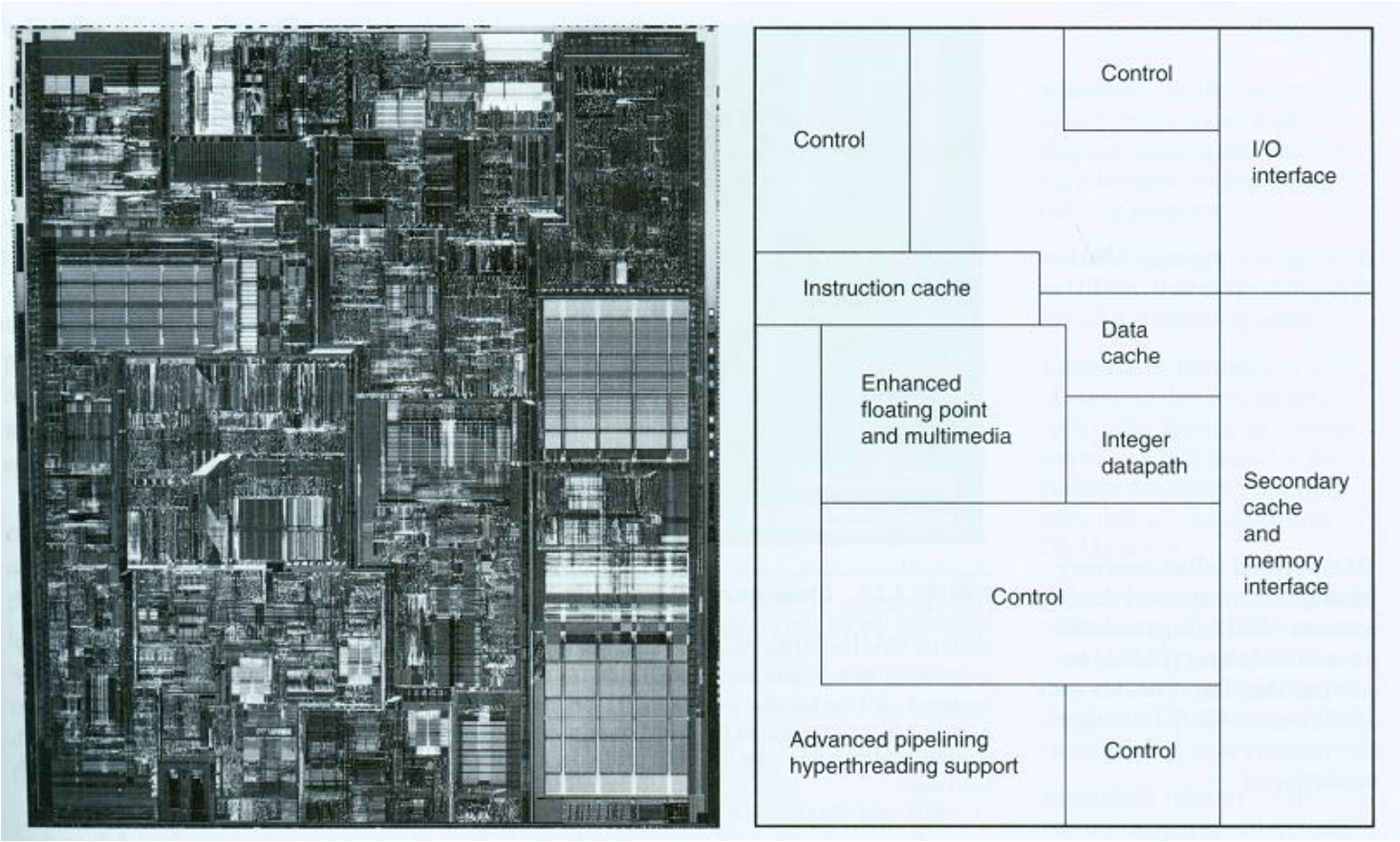
- PostPC device
- Supersedes keyboard and mouse
- Resistive and Capacitive types
  - Most tablets, smart phones use capacitive
  - Capacitive allows multiple touches simultaneously



Computer components

- Motherboard
  - A plastic board containing packages of integrated circuits or chips, including processor, cache, memory, and connectors for I/O devices such as networks and disks
- Integrated Circuit (IC) or chips → CPU & Chipsets
  - A device combining dozens to millions of transistors
- Memory (chips and modules)
  - The storage area in which programs are kept when they are running and that contains the data needed by the running programs

Opening the box - CPU



Inside the processor chip used on the board

Inside the Processor

- Apple A5

Computer components: CPU

- Central processor unit (CPU)
  - Also called processor. The active part of the computer, which contains the datapath and control and which adds numbers, tests numbers, signals I/O devices to activate, and so on.
- Datapath units
  - The component of the processor that performs **arithmetic operations (add, sub, mul, div)**
- Control unit
  - The component of the processor that commands the datapath, memory, and I/O devices according to the instructions of the program

Computer components: Memory: A place for data

- Memory: Memory built as an integrated circuit, it provides random access to any location

- Static Random Access Memory (SRAM) chips (external on PCB) – old cache designs (obsolete)
- Cache memory (L1, L2, L3, with SRAM design) – inside CPU: A small, fast memory inside CPU that acts as a data access buffer for a slower, larger memory

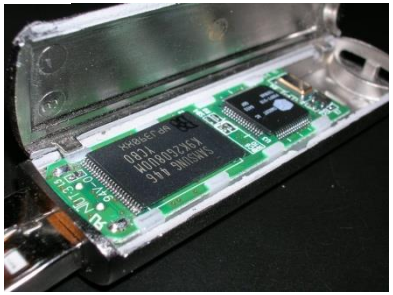
- External memory: Dynamic Random Access Memory (DRAM) module

Memory (I)

- Memory
  - The storage area in which programs are kept when they are running and that contains the data needed by the running programs
- Volatile memory
  - Storage, such as SRAM/DRAM, that only retains data only if it is receiving power
- Nonvolatile memory
  - A form of memory that retains data even in the absence of a power source and that is used to store programs between runs. Magnetic disk (hard disk, HD) and Solid-state Drive (SSD) are nonvolatile and DRAM is not.

A Safe Place for Data

- Volatile main memory
  - Loses instructions and data when power is OFF
- Non-volatile secondary memory
  - Magnetic disk (hard drive, HD)
  - Flash memory (USB)
  - Optical disk (CDROM, DVD)



Memory (II)

- Primary (Main) Memory
  - Volatile memory used to hold programs while they are running; typically consists of DRAM in today's computers with large capacity (4G, 8G, 16G)

- Secondary Memory
  - Nonvolatile memory used to store programs and data between runs; typically consists of magnetic disks (Hard drive, Hard disk) and/or Solid-state Drive (SSD) in today's computers

Communicating with Other Computers (NOT the focus of this course)

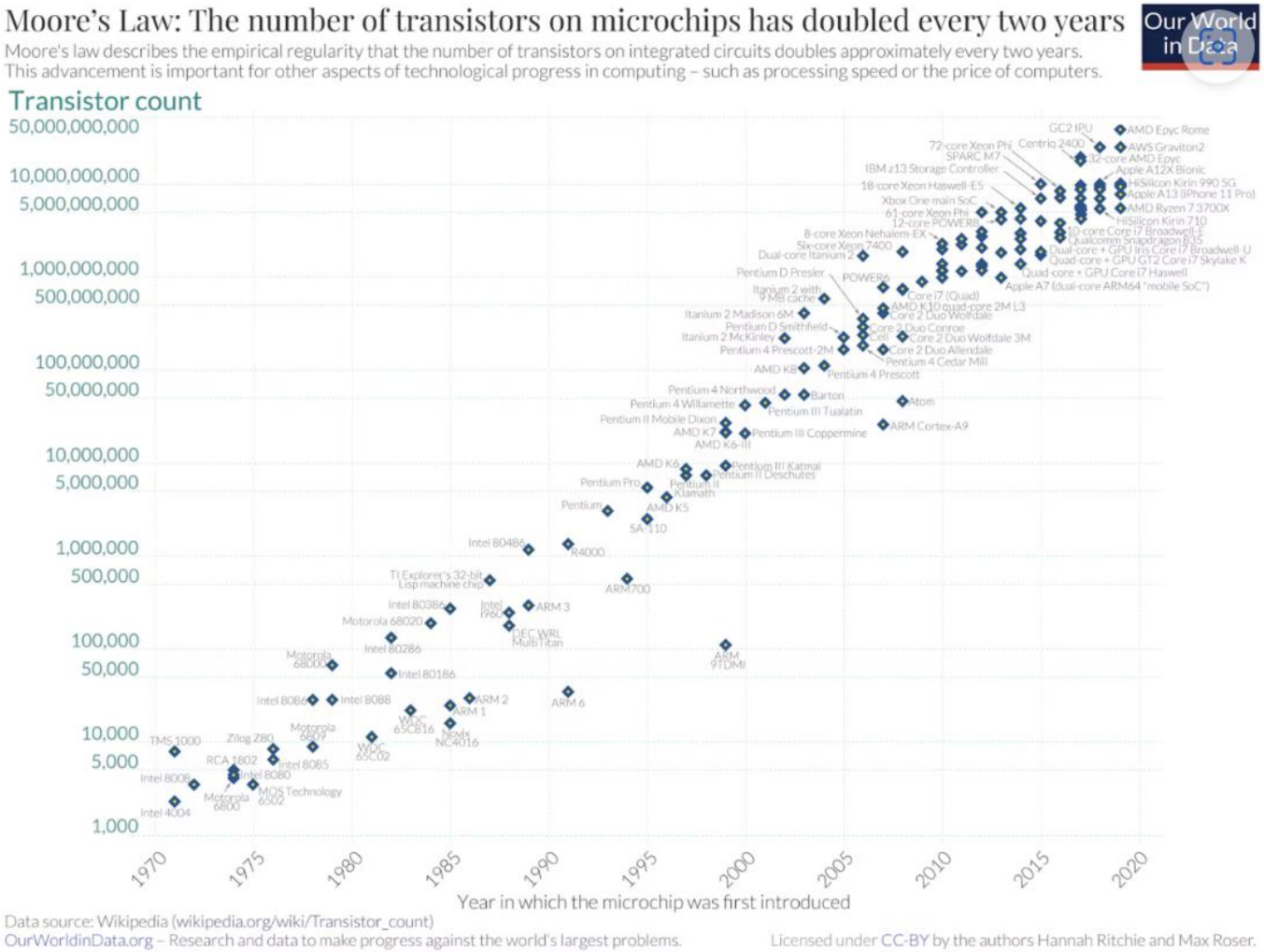
- Networked computers have several major advantages
  - *Communication*: Information is exchanged between computers at high speeds
  - *Resource sharing*: Rather than each machine having its own I/O devices, devices can be shared by computers on the *computer network*
  - *Non-local access*: By connecting computers over long distances, users need not be near the computer they are using

Communicating with Other Computers (NOT the focus of this course)

- Local area network (LAN)
  - A network designed to carry data within a geographically confined area, typically within a single building, e.g. Gigabit/10G Ethernet.
  - Wireless LAN (WLAN, WiFi – 802.11a/b/g/n), Bluetooth

- Wide area network (WAN)
  - A network extended over hundreds of kilometers which can span a continent, e.g, ADSL/VDSL/光世代 (wireline)
  - Mobile Access (3G, 4G, HSPDA, WiMAX)

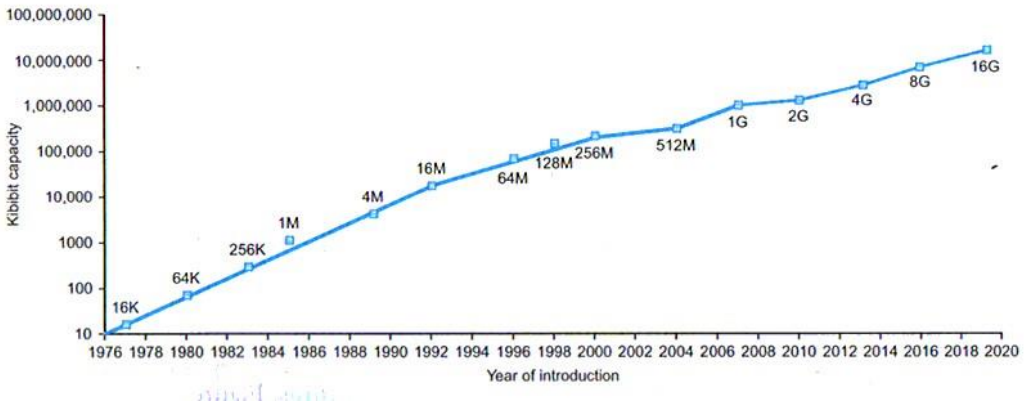
Moore's Law



https://www.splunk.com/en_us/blog/learn/moores-law.html

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
2013	Ultra large scale IC	250,000,000,000

Price of DRAM over Time

date (X)	$/Mbyte (Y)	Date		Ref:	Page	Company	Size	Cost	Speed	Memory Type
							KByte	US $	nsec	
1957.00	411,041,792	1957		Phister 366		C.C.C.	0.00098	392.00	10000	transistor Flip-Flop
1959.00	67,947,725	1959		Phister 366		E.E.Co.	0.00098	64.80	10000	vacuum tube Flip-Flop
1960.00	5,242,880	1960		Phister 367		IBM	0.00098	5.00	11500	IBM 1401 core memory
1965.00	2,642,412	1965		Phister 367		IBM	0.00098	2.52	2000	IBM 360/30 core memory
1970.00	734,003	1970		Phister 367		IBM	0.00098	0.70	770	IBM 370/135 core memory
1973.00	399,360	1973	Jan	PDP8/e User Price List		DEC	12	4680.00		Core memory 8KwordX12 bit
1974.00	314,573	1974		Phister 367		IBM	0.00098	0.30	800	IC Memory for IBM 370/125
1975.00	421,888	1975	Jan	Radio-Electronics		MITS	0.25	103.00	1000	Altair 8800 256 Byte Static Board
1975.08	180,224	1975	Feb			MITS	1	176.00		Altair 1K Static Board

https://jcmit.net/memoryprice.htm

$/Gigabyte

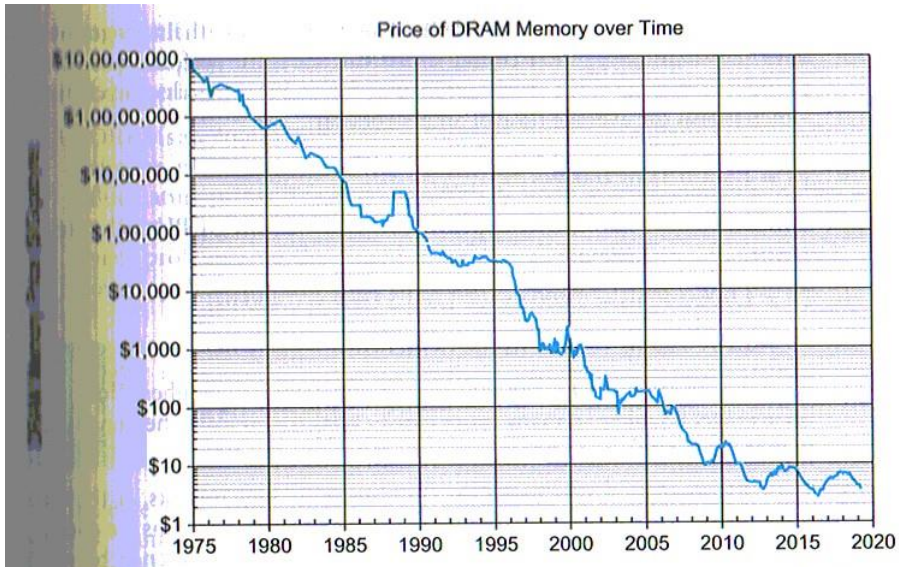
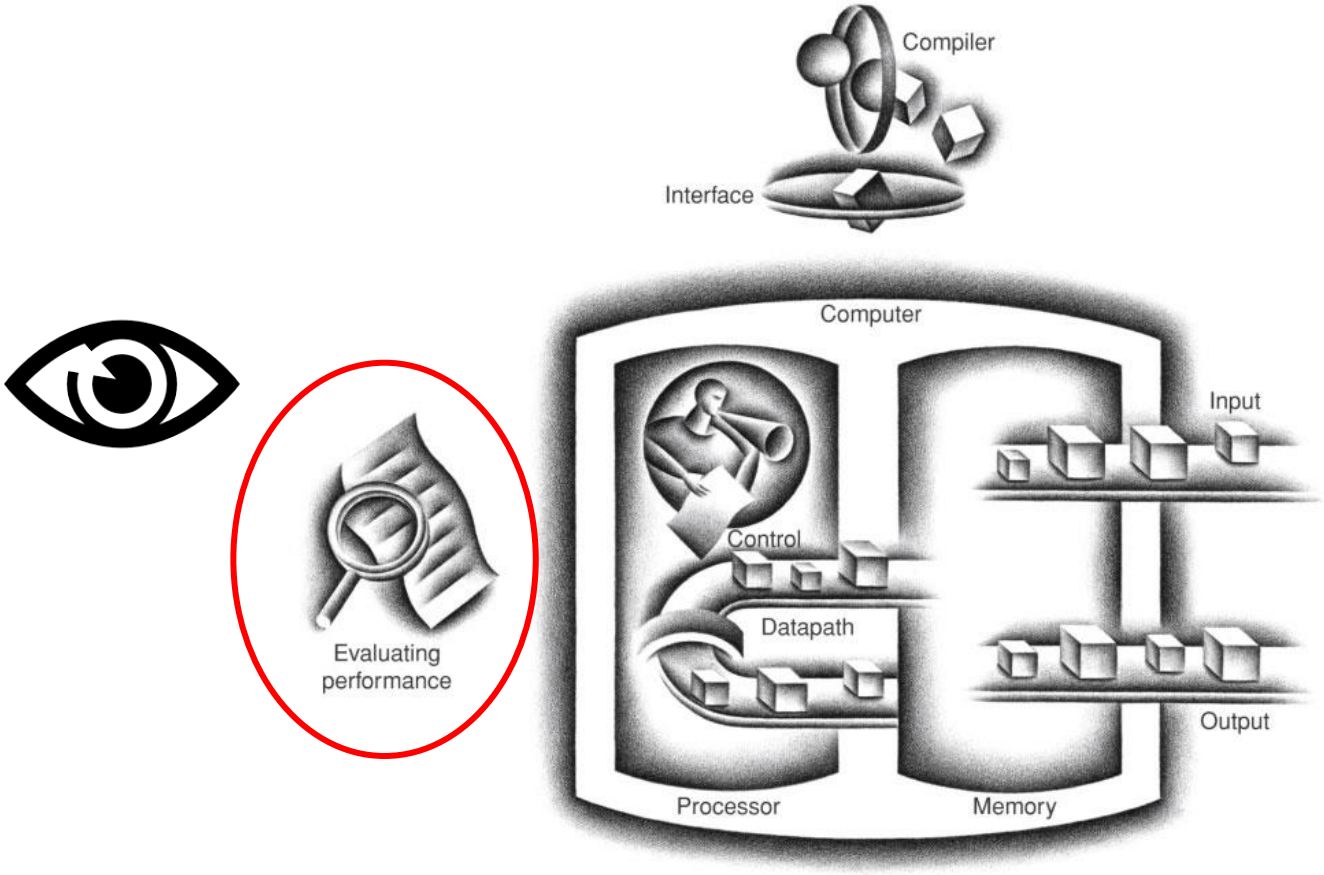


FIGURE 1.21 Price of memory per gigabyte between 1975 and 2020. (Source: https://jcmit.net/memoryprice.htm)

Outline

- 1.1 Introduction
- 1.2 Below Your Program
- 1.3 Under the Covers
- 1.4 Performance
- 1.5 The Power Wall
- 1.6 The Sea Change: The Switch from Uniprocessors to Multiprocessors
- 1.7 Benchmarking
- 1.8 Fallacies and Pitfalls
- 1.9 Concluding Remarks

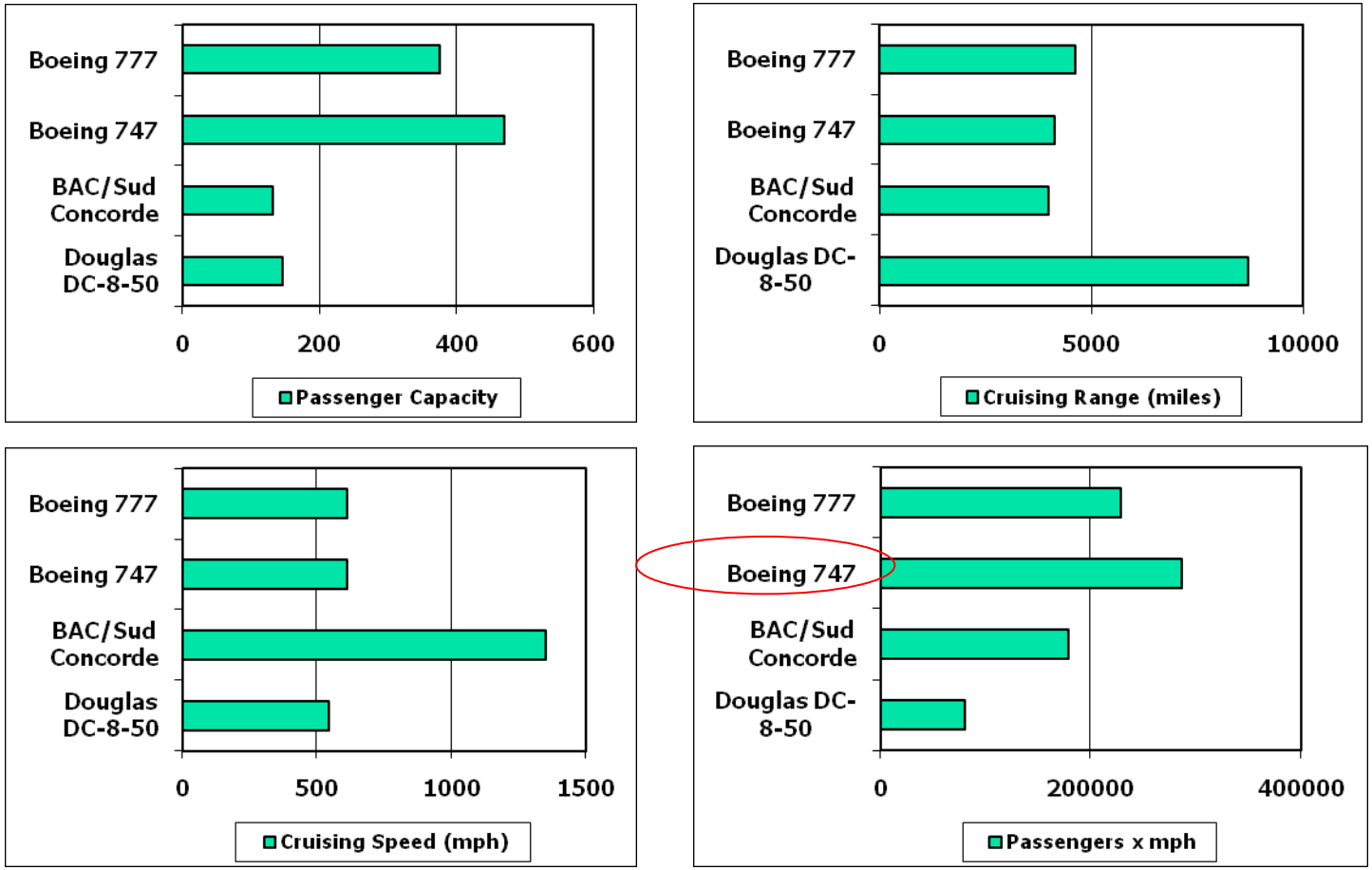
The organization of a computer



The organization of a computer, showing the five classic components

Defining Performance

- Which airplane has the best performance?



Passenger Throughput

Response Time and Throughput

- Response time
  - How long it takes to do a requested task
- Throughput
  - Total work done per unit time
    - e.g., Total tasks/transactions/etc. per hour
- How are response time and throughput affected by
  - Replacing the processor with a faster version?
  - Adding more memory modules?
- We'll focus on *response time* for now…

Relative Performance

- Define *Performance = 1/Execution Time*
- "X is n time faster than Y"

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

Relative Performance

- Example: Computer A runs a program in 10 sec and computer B runs the same program in 15 sec, how much faster is A than B?

Sol:

$$\frac{\text{Performance}_A}{\text{Performance}_B}$$

$$= \frac{\text{Execution time}_B}{\text{Execution time}_A}$$

$$= \frac{15\text{s}}{10\text{s}}$$

$$= 1.5$$

→ *A is 1.5 times faster than B*

Measuring Performance

- CPU execution time = User + System CPU time
  - Also called CPU time. The actual time the CPU spends computing for a specific task

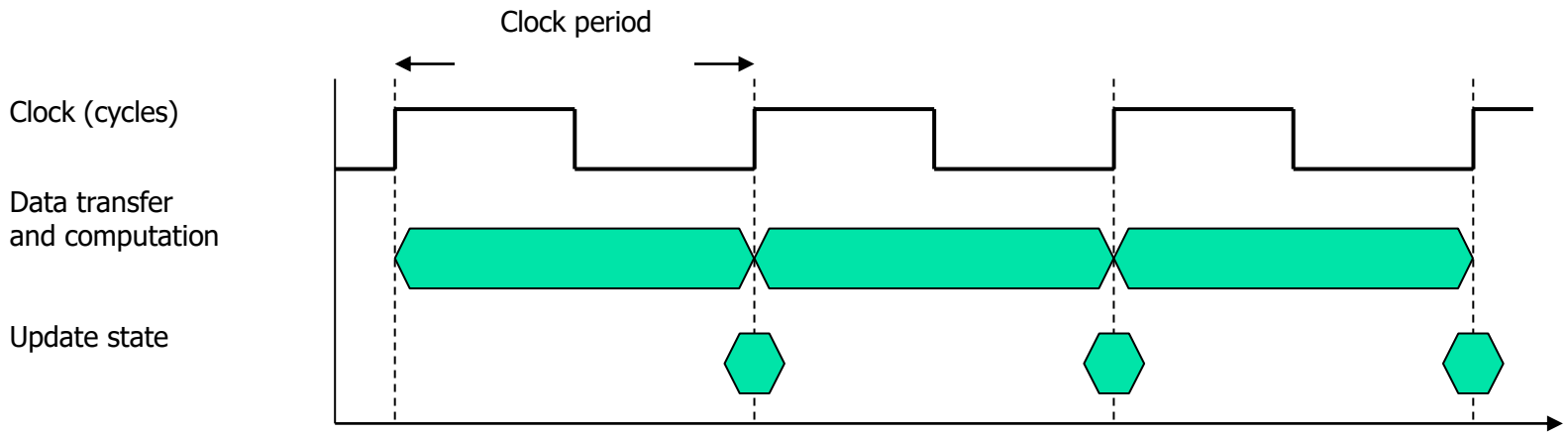
- User CPU time (focus of this chapter)
  - The CPU time spent in a program itself
- System CPU time
  - The CPU time spent in the operating system (OS) performing tasks (resource management) on behalf of the program (e.g., *fopen(), pringf(),…* in C program).

Measuring Performance

- (Total) Clock cycles of a program
  - The discrete time intervals of a program (also called *ticks, clock ticks, clock periods, clocks, cycles*). Usually measured at the processor clock which runs at a constant rate.
- Clock period (or called *clock cycle time*)
  - The length of a complete clock cycle (0.25 nanoseconds, 0.25ns; 250 picoseconds, or 250ps)
- Clock rate (or called *clock frequency*)
  - The inverse of the clock period (e.g., 4 gigahertz, 4GHz)

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

Measuring Execution Time

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

- CPU time (≠ *Elapsed* time, most of the time)
  - Time spent processing a given job
    - NOT including I/O time, other jobs' shares
  - Comprises only user CPU time and system CPU time
  - Different programs are affected differently by CPU time and system performance

CPU Time

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

- Clock Cycle Time = Clock Period
- Performance improved by
  - Reducing number of clock cycles (of a program)
  - Increasing clock rate
  - Hardware designer must often trade off clock rate against cycle count

CPU Time Example

- Computer A: 2GHz clock, 10s CPU time → calculate cycles first.
- Designing a new Computer B (with new instruction sets)
  - Aim for 6s CPU time
  - Can do faster clock, but need 1.2 × total clock cycles
- How fast must Computer B clock be?

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

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

$$= 10\text{s} \times 2\text{GHz} = 20 \times 10^9$$

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

Instruction Count and CPI (Cycles per Instruction)

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

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

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

- ***Instruction Count*** for a Program (Algorithm)
  - Determined by program, ISA (Instruction Set Architecture, 架構), and compiler (編譯器)
- ***Average Cycles per Instruction*** (Average CPI)
  - Determined by CPU hardware (addition vs multiplication inst.)
  - If different instructions have different CPI
    - Average CPI affected by **instruction mix (指令出現之比例)**

CPI Example (for same ISA)

- Computer A: Cycle Time = 250ps, CPI = 2.0
- Computer B: Cycle Time = 500ps, CPI = 1.2
- Same ISA (Instruction Set Architecture) and same Instruction count
- Which is faster, and by how much?

$$\text{CPU Time}_A = \text{Instruction Count} \times \text{CPI}_A \times \text{Cycle Time}_A$$
$$= I \times 2.0 \times 250\text{ps} = I \times 500\text{ps}$$
$$\text{CPU Time}_B = \text{Instruction Count} \times \text{CPI}_B \times \text{Cycle Time}_B$$
$$= I \times 1.2 \times 500\text{ps} = I \times 600\text{ps}$$
$$\frac{\text{CPU Time}_B}{\text{CPU Time}_A} = \frac{I \times 600\text{ps}}{I \times 500\text{ps}} = 1.2$$

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

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

CPI Example

- Alternative compiled code sequences (for same function) using instructions in classes A, B, C *(IC: Inst. Count)*

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
= 2×1 + 1×2 + 2×3
= 10

Avg. CPI = 10/5 = 2.0

Sequence 2: IC = 6

Clock Cycles
= 4×1 + 1×2 + 1×3
= 9

Avg. CPI = 9/6 = 1.5

Instruction Mix

- A measure of the dynamic frequency of instructions across one or many programs

Class	A	B	C
CPI for class	1	2	3
IC in sequence 1	2	1	2
IC in sequence 2	4	1	1
Inst. Mix for Seq. 1	2/5	1/5	2/5
Inst. Mix for Seq. 2	4/6 = 2/3	1/6	1/6
Average CPI for 1	1*(2/5) + 2*(1/5) + 3*(2/5) = 10/5 = 2		
Average CPI for 2	1*(2/3) + 2*(1/6) + 3*(1/6) = 9/6 = 1.5		

CPU Performance and Its Factors

- CPU time = Instruction count * CPI * Clock cycle time
- CPU time = (Instruction count*CPI) / Clock rate

- The basic components of performance and how each is measured

Components of performance	Units of measure
CPU execution time for a program	Seconds for the program
Instruction count	Instructions executed for the program
Clock cycles per instruction (CPI)	Average number of clock cycles per instruction
Clock cycle time	Seconds per clock cycle

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

Performance of a program

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 complicated 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. e.g., support of Multiplication

Performance Summary

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

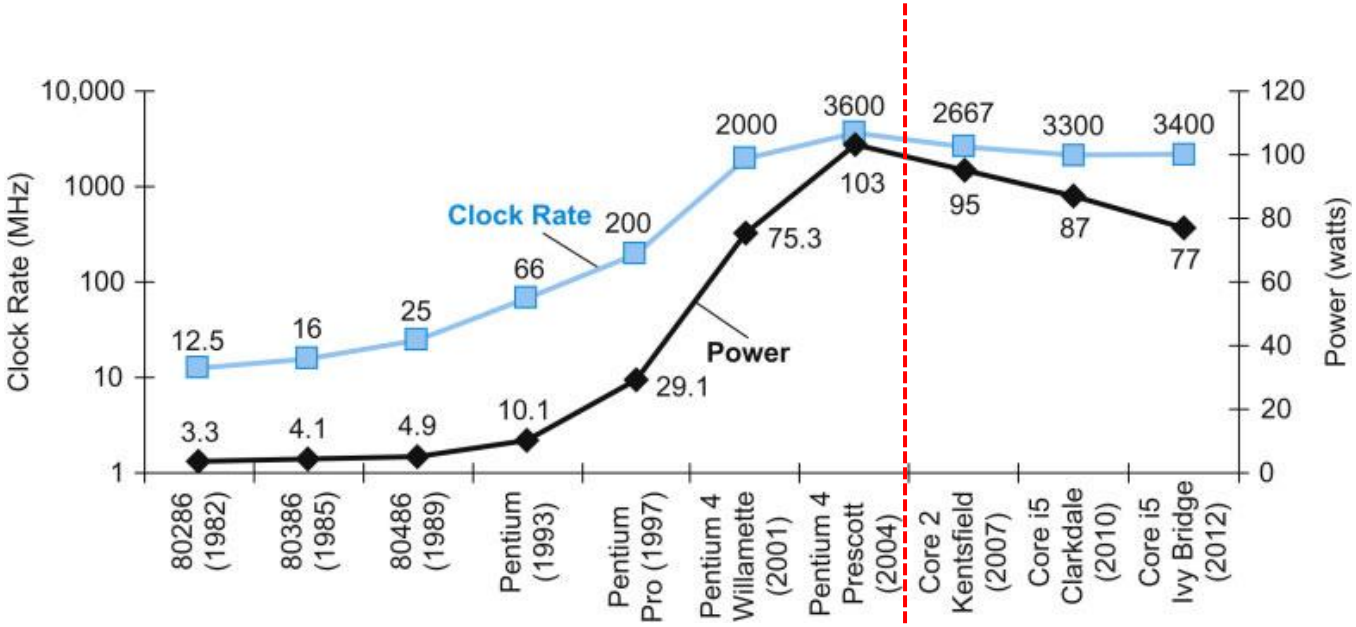
Average CPI *Clock Cycle Time (T_c)*

- Performance depends on
  - Algorithm: affects Instruction Count (IC), possibly CPI
  - Programming language: affects IC, CPI
  - Compiler: affects IC, CPI
  - Instruction set architecture and VLSI Process: affects IC, CPI, T_c

Outline

- 1.1 Introduction
- 1.2 Below Your Program
- 1.3 Under the Covers
- 1.4 Performance
- 1.5 The Power Wall
- 1.6 The Sea Change: The Switch from Uniprocessors to Multiprocessors
- 1.7 Benchmarking
- 1.8 Fallacies and Pitfalls
- 1.9 Concluding Remarks

Power Trends



- In CMOS IC technology

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

×30

5V → 1V

×1000

Power v.s. Multicore x86

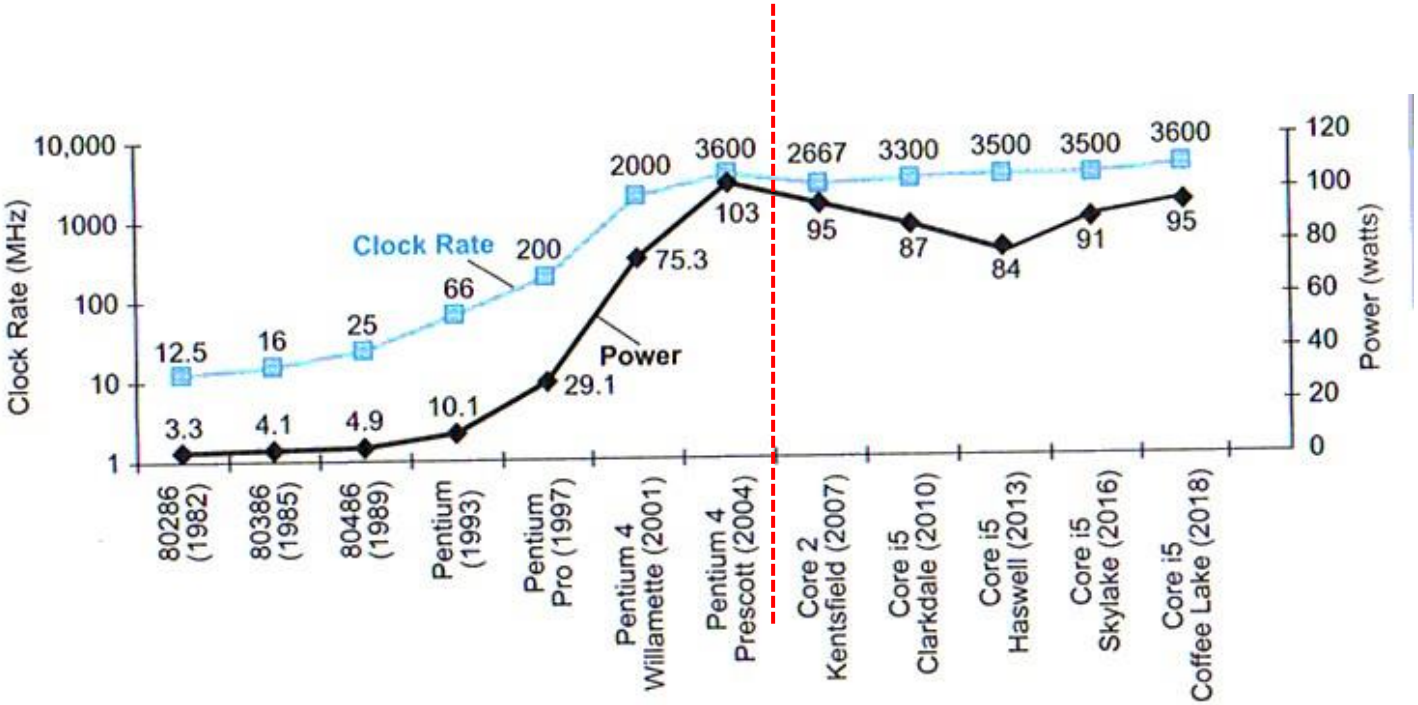
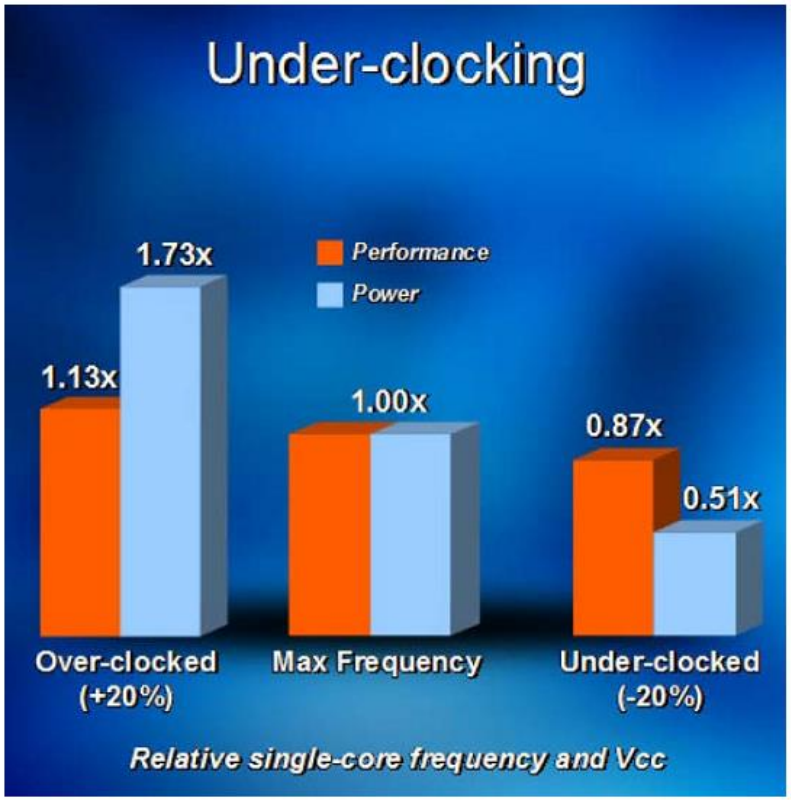
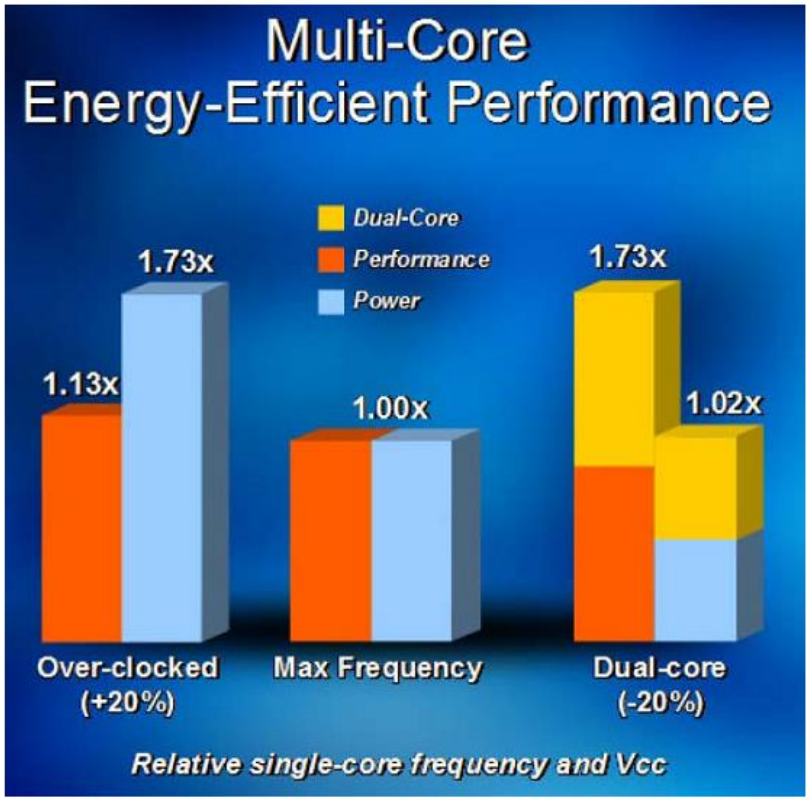


FIGURE 1.16 Clock rate and Power for Intel x86 microprocessors over nine generations and 36 years. The Pentium 4 made a dramatic jump in clock rate and power but less so in performance. The Prescott thermal problems led to the abandonment of the Pentium 4 line. The Core 2 line reverts to a simpler pipeline with lower clock rates and multiple processors per chip. The Core i5 pipelines follow in its footsteps.

Multicore for Low-Power





Source: Intel IDF 07

Reducing Power

- Suppose a new CPU has
  - 85% of capacitive load of old CPU (better technology)
  - 15% voltage and 15% frequency reduction (2 cores)

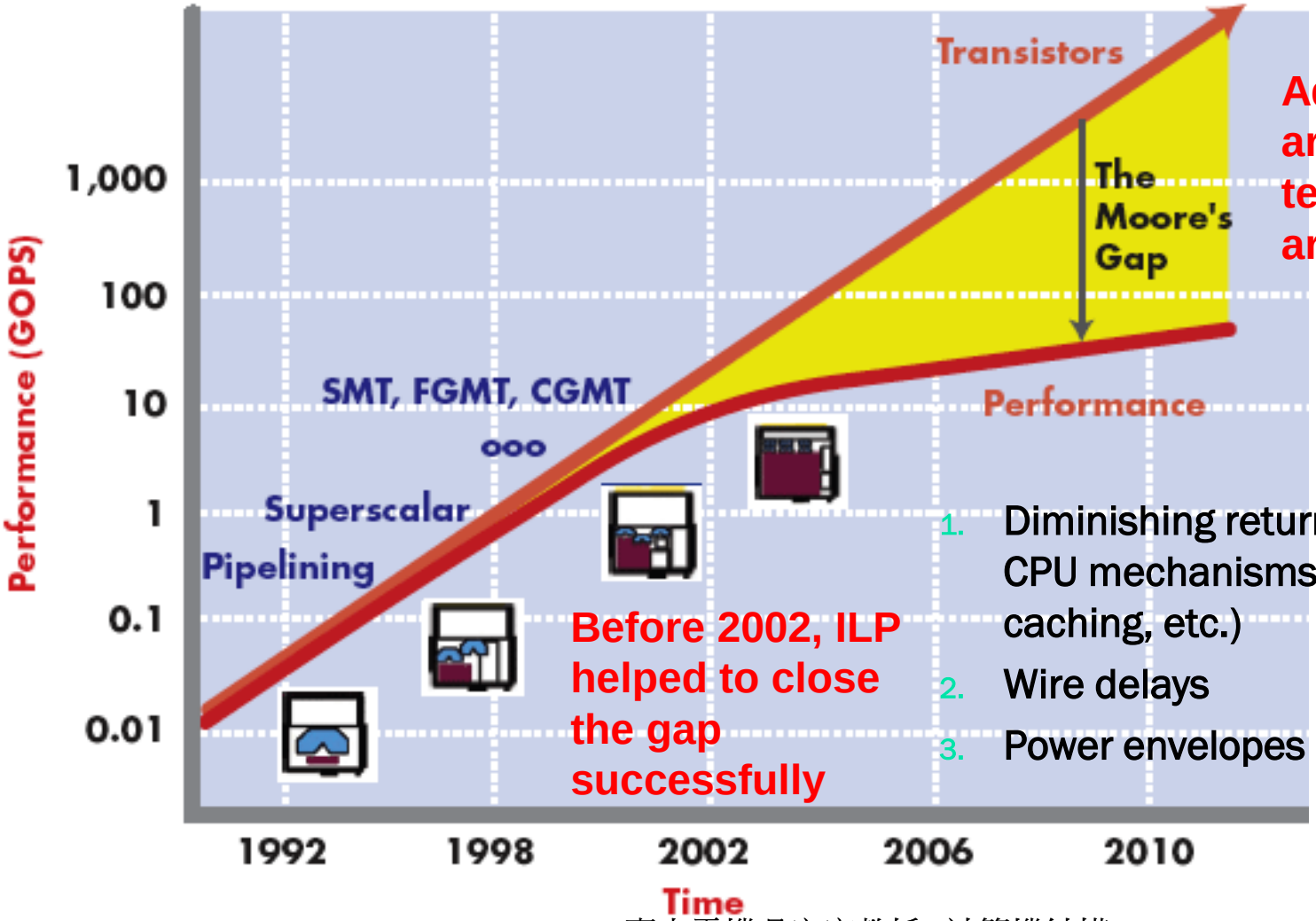
$$\frac{P_{new}}{P_{old}} = \frac{C_{old} \times 0.85 \times (V_{old} \times 0.85)^2 \times F_{old} \times 0.85}{C_{old} \times V_{old}^{\ 2} \times F_{old}} = 0.85^4 = 0.52$$

- The power wall
  - We can't reduce voltage further
  - We can't remove more heat
- How else can we improve performance?
- See Video Demo (Intel v.s. AMD)

Outline

- 1.1 Introduction
- 1.2 Below Your Program
- 1.3 Under the Covers
- 1.4 Performance
- 1.5 The Power Wall
- 1.6 The Sea Change: The Switch from Uniprocessors to Multiprocessors
- 1.7 Benchmarking
- 1.8 Fallacies and Pitfalls
- 1.9 Concluding Remarks

Driving Force (I) - Performance/Technology Gap

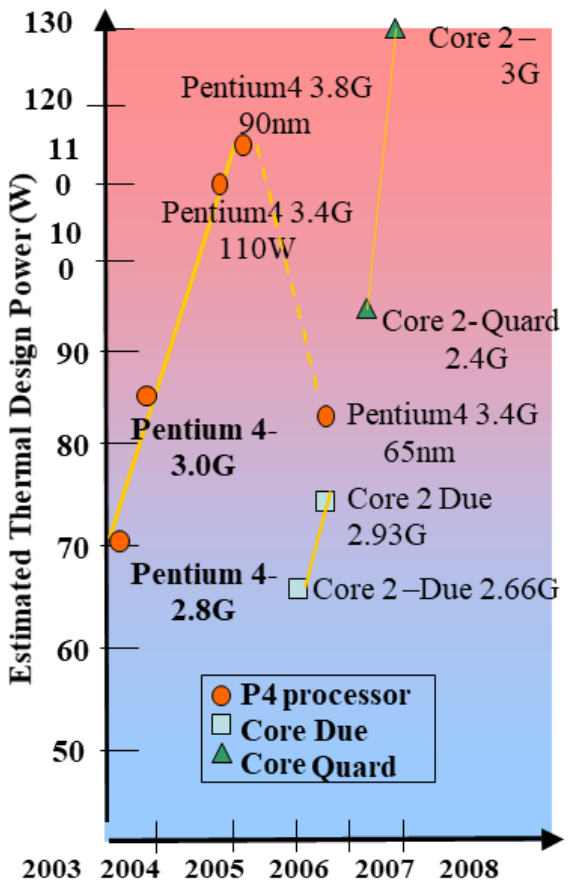


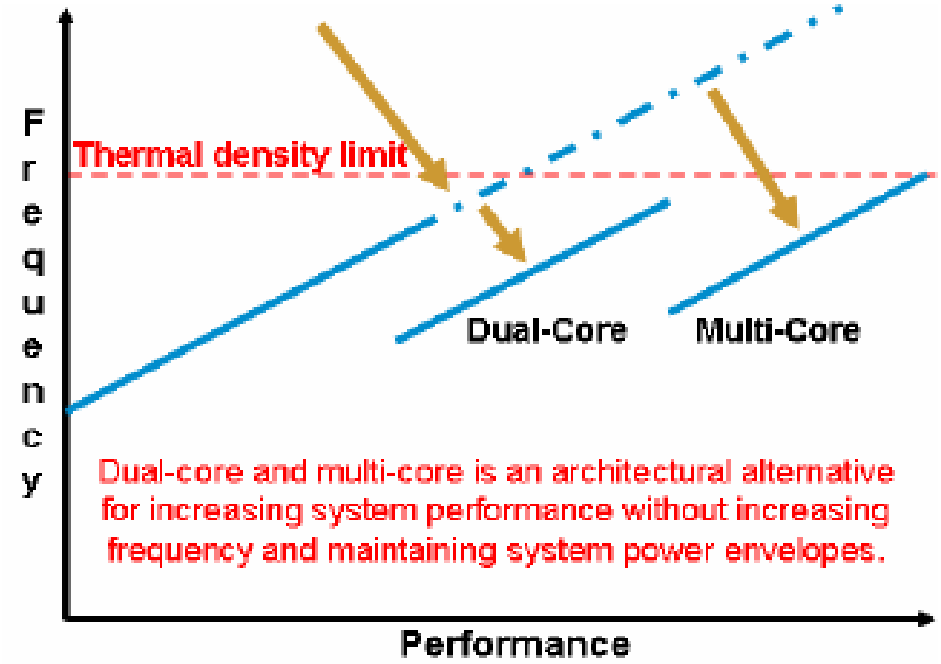
Advanced architecture techniques are required!

Before 2002, ILP helped to close the gap successfully

1. Diminishing returns from single CPU mechanisms (pipelining, caching, etc.)
2. Wire delays
3. Power envelopes

Driving Force (II)- Thermal Density Limit





Note: For more information, see "Intel® Multi-Core Processors: Making the Move to Quad-Core and Beyond," http://www.intel.com/technology/architecture/downloads/quad-core-06.pdf

Figure 2. Thermal Density Limits Frequency

Multicore v.s. Frequency

Product	AMD Opteron X4 (Barcelona)	Intel Nehalem	IBM Power 6	Sun Ultra SPARC T2 (Niagara 2)
Cores per chip	4	4	2	8
Clock rate	2.5 GHz	~ 2.5 GHz ?	4.7 GHz	1.4 GHz
Microprocessor power	120 W	~ 100 W ?	~ 100 W ?	94 W

Outline

- 1.1 Introduction
- 1.2 Below Your Program
- 1.3 Under the Covers
- 1.4 Performance
- 1.5 The Power Wall
- 1.6 The Sea Change: The Switch from Uniprocessors to Multiprocessors
- 1.7 Benchmarking
- 1.8 Fallacies and Pitfalls
- 1.9 Concluding Remarks

SPEC

- System Performance Evaluation Cooperation (SPEC) benchmark
  - A set of programs form a **"workload"**
  - A set of standard **CPU**-intensive, integer and floating point benchmarks based on "**real programs**"

- Similar to AnTuTu (安兔兔評測) for mobile phones-用戶體驗測試(多任務與虛擬機器)、CPU性能測試、RAM性能測試、2D/3D圖形性能測試以及數據儲存I/O的性能測試

System Description for "Reproducibility"

Hardware	
Hardware vendor	Dell
Model number	Precision WorkStation 360 (3.2 GHz Pentium 4 Extreme Edition)
CPU	Intel Pentium 4 (800 MHz system bus)
CPU MHz	3200
FPU	Integrated
CPU(s) enabled	1
CPU(s) orderable	1
Parallel	No
Primary cache	12K(I) micro-ops + 8KB(D) on chip
Secondary cache	512KB(I+D) on chip
L3 cache	2048KB(I+D) on chip
Other cache	N/A
Memory	4 x 512MB ECC DDR400 SDRAM CL3
Disk subsystem	1x 80GB ATA/100 7200 RPM
Other hardware	

Software	
Operating system	Windows XP Professional SP1
Compiler	Intel C++ Compiler 7.1 (20030402Z) Microsoft Visual Studio.NET (7.0.9466) MicroQuill SmartHeap Library 6.01
File system type	NTFS
System state	Default

FIGURE 4.3 System description of a desktop system using the fastest Pentium 4 available in 2003. In addition to this formatted mandatory description, there are 23 lines of notes describing special flag settings used for portability (4), optimization (2), tuning (12), base timing (2), a special library (2), and BIOS setting (1).

SPEC CPU2000 benchmarks

Integer benchmarks		FP benchmarks	
Name	**Description**	**Name**	**Type**
gzip	Compression	wupwise	Quantum chromodynamics
vpr	FPGA circuit placement and routing	swim	Shallow water model
gcc	The Gnu C compiler	mgrid	Multigrid solver in 3-D potential field
mcf	Combinatorial optimization	applu	Parabolic/elliptic partial differential equation
crafty	Chess program	mesa	Three-dimensional graphics library
parser	Word processing program	galgel	Computational fluid dynamics
eon	Computer visualization	art	Image recognition using neural networks
perlbmk	perl application	equake	Seismic wave propagation simulation
gap	Group theory, interpreter	facerec	Image recognition of faces
vortex	Object-oriented database	ammp	Computational chemistry
bzip2	Compression	lucas	Primality testing
twolf	Place and rote simulator	fma3d	Crash simulation using finite-element method
		sixtrack	High-energy nuclear physics accelerator design
		apsi	Meteorology: pollutant distribution

- Normalized by dividing the execution time on a Sun Ultra5_10 with a 300MHz processor ***(SPEC Ratio)***

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 (http://www.spec.org/cpu2006/CFP2006/)
  - 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 AMD 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

High cache miss rates

- Consists of 12 integer benchmarks (CINT2006) and 17 floating-point benchmarks (CFP2006)

CINT2006 for Intel Core i7 920

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

Benchmarking the 1.8G Intel Xeon CPU

SPEC CPU2017: 10 integer benchmark (SPECspeed2017) and 13 floating-point benchmarks (SPECspeed2017 Floating Point)

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

FIGURE 1.18 SPECspeed 2017 Integer benchmarks running on a 1.8 GHz Intel Xeon E5-2650L. 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 SPECspeed 2017 Integer is the geometric mean of the SPECratios. SPECspeed 2017 has multiple input files for perlbench, gcc, x264, and xz. For this figure, execution time and total clock cycles are the sum running times of these programs for all inputs.

SPEC Power Benchmark

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

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

Server Side Java (ssj) performance, it is one of the most comprehensive standards for energy efficiency to date giving it instant credibility.

Standardized Energy Efficiency Benchmark

- **SPECpower_ssj2008** is basically a measure of **ssj_ops/watt** (**server side Java operations per second per watt**). → can consider it as **ssj_opj** (server side Java operations per unit of energy in Joules)
- "per second per watt" is by definition "per Joule".
- SPECpower_ssj2008 factors in the fact that servers usually aren't operated at peak capacity and they're even idle at times.

- To factor for idle and peak load power consumption, average power consumption at 0, 10, 20, 30, all the way through 100% **load capacity** are measured and disclosed.
- Then server side Java operations per second are divided by the average power consumption in watts at every 10% increment and then all the scores are averaged again to produce the "overall" ssj_ops/watt metric.

Source: http://www.zdnet.com/blog/ou/spec-launches-standardized-energy-efficiency-benchmark/927

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%	920,35	233
30%	70,500	222
20%	47,126	206
10%	23,066	180
0%	0	141
Overall sum	1,283,590	2,605
∑ssj_ops/ ∑power		493

SPECpower_ssj2008

https://www.spec.org/power_ssj2008/

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

FIGURE 1.19 SPECpower_ssj2008 running on a dual socket 2.2 GHz Intel Xeon Platinum 8276L with 192 GiB of DRAM and one 80 GB SSD disk.

Server Manufacturer	Micro-processor	Total Cores/ Sockets	Clock Rate	Peak Performance (ssj_ops)	100% Load Power	50% Load Power	50% Load/ 100% Power	10% Load Power	10% Load/ 100% Power	Active Idle Power	Active Idle/ 100% Power
HP	Xeon E5440	8/2	3.0 GHz	308,022	269 W	227 W	84%	174 W	65%	160 W	59%
Dell	Xeon E5440	8/2	2.8 GHz	305,413	276 W	230 W	83%	173 W	63%	157 W	57%
Fujitsu Seimens	Xeon X3220	4/1	2.4 GHz	143,742	132 W	110 W	83%	85 W	65%	80 W	60%

FIGURE 1.22 SPECPower results for three servers with the best overall ssj_ops per watt in the fourth quarter of 2007. The overall ssj_ops per watt of the three servers are 698, 682, and 667, respectively. The memory of the top two servers is 16 GB and the bottom is 8 GB.

Matrix Multiply in Python (speedup)

```
for i in xrange(n):
    for j in xrange(n):
        for k in xrange(n):
            C[i][j] += A[i][k] * B[k][j]
```

960x960 → 5 mins
4096x4096 → 6 hours

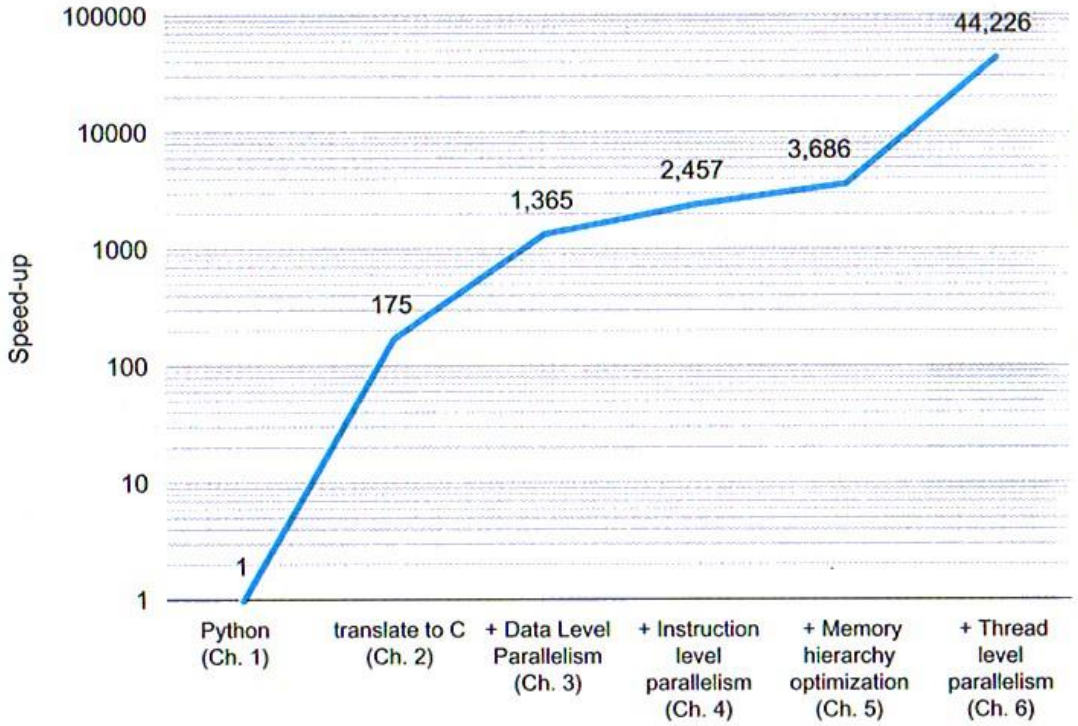


FIGURE 1.20 Optimizations of matrix multiply program in Python in the next five chapters of this book.

Outline

- 1.1 Introduction
- 1.2 Below Your Program
- 1.3 Under the Covers
- 1.4 Performance
- 1.5 The Power Wall
- 1.6 The Sea Change: The Switch from Uniprocessors to Multiprocessors
- 1.7 Benchmarking
- 1.8 Fallacies and Pitfalls
- 1.9 Concluding Remarks

Pitfall: Amdahl's Law

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

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

Make the common case fast – Corollary of Amdhl's law

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

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

- 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

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

$$= \frac{\text{Instruction count}}{\dfrac{\text{Instruction count} \times \text{CPI}}{\text{Clock rate}} \times 10^6} = \frac{\text{Clock rate}}{\text{CPI} \times 10^6}$$

- CPI varies between programs on a given CPU

>> Due to Instruction Mix & ISA (Instruction Set Architecture)

MIPS as a Performance Measure

	CPI for each instruction class		
	A	B	C
CPI	1	2	3

Code from	Instruction counts (in billions) for each instruction class		
	A	B	C
Compiler 1	5	1	1
Compiler 2	10	1	1

- Same program (e.g., Bubble sorting) from two Different compilers
- Assume that the computer's clock rate is **4 GHz**
- Which code sequence will execute faster

(a) According to MIPS?

(b) According to execution time?

MIPS as a Performance Measure (cont.)

- Execution time = CPU clock cycles/Clock rate
- CPU clock cycles= $\sum_{i=1}^{n} (CPI_i * C_i)$
  - CPU clock $cycles_1$=$(5*1+1*2+1*3)*10^9=10*10^9$
  - CPU clock $cycles_2$=$(10*1+1*2+1*3)*10^9=15*10^9$

  → Execution $time_1$= $(10*10^9)/(0.25*10^9)$=2.5 seconds

  → Execution $time_2$= $(15*10^9)/(0.25*10^9)$=3.75 seconds
  - Performance_1 > Performance_2
- MIPS = Instruction count/(Execution time*10^6)

  → $MIPS_1$= $[(5+1+1)*10^9]/(2.5*10^6)$ = 2800

  → $MIPS_2$= $[(10+1+1)*10^9]/[3.75*10^6]$ = 3200

  MIPS_1 < MIPS_2 (Sales terms → (Peak) MIPS Performance)

Fallacy:

- Computers at low utilization use little power
- Need: "*Energy-proportional computing*"

Server Manufacturer	Micro-processor	Total Cores/ Sockets	Clock Rate	Peak Performance (ssj_ops)	100% Load Power	50% Load Power	50% Load/ 100% Power	10% Load Power	10% Load/ 100% Power	Active Idle Power	Active Idle/ 100% Power
HP	Xeon E5440	8/2	3.0 GHz	308,022	269 W	227 W	84%	174 W	65%	160 W	59%
Dell	Xeon E5440	8/2	2.8 GHz	305,413	276 W	230 W	83%	173 W	63%	157 W	57%
Fujitsu Seimens	Xeon X3220	4/1	2.4 GHz	143,742	132 W	110 W	83%	85 W	65%	80 W	60%

FIGURE 1.22 SPECPower results for three servers with the best overall ssj_ops per watt in the fourth quarter of 2007. The overall ssj_ops per watt of the three servers are 698, 682, and 667, respectively. The memory of the top two servers is 16 GB and the bottom is 8 GB.

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

Concluding Remarks

- Overview of hierarchical layers of abstraction for computer
  - In both hardware and software
- Cost/performance (C/P) ratio is improving
  - Due to underlying IC technology development
- Execution time: the performance measure
- Power is a limiting factor
  - Use parallelism (multicore) to improve performance
- Instruction set architecture (ISA) (Chapter 2)
  - The hardware/software interface