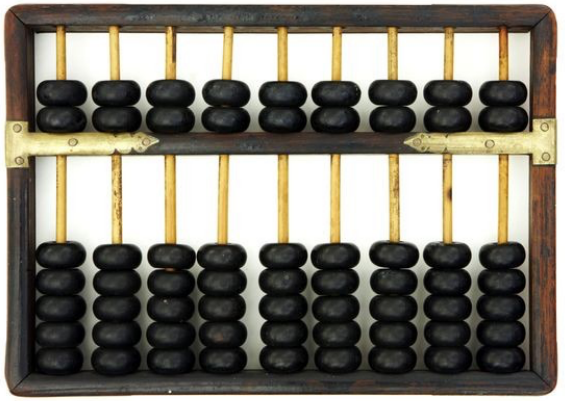
COMPUTER ARCHITECTURE

Chapter 1: Technology & Performance evaluation

TECHNOLOGY REVIEW

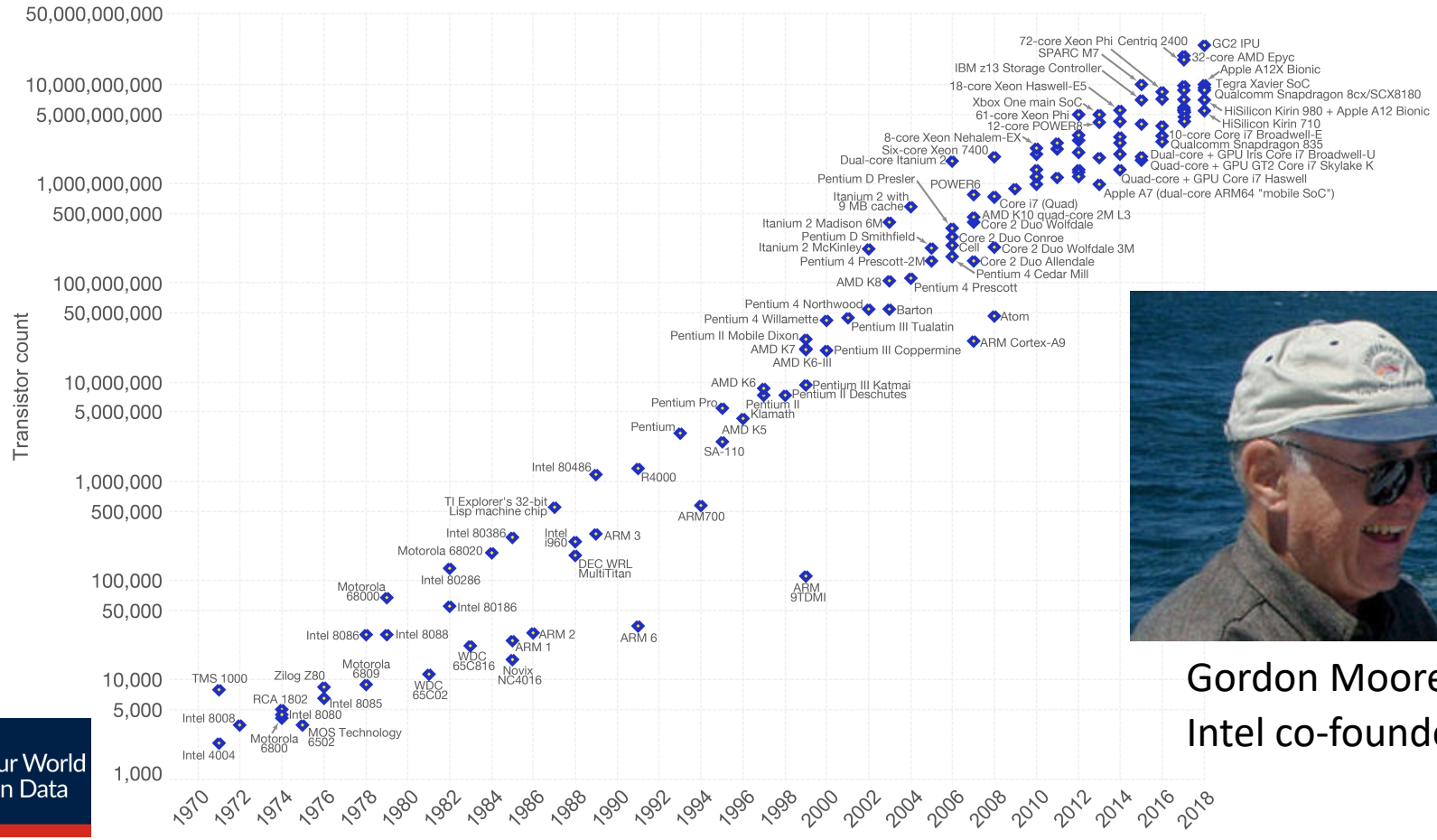
The computer revolution

- The **third revolution** along with agriculture and industry
- 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



ABACUS

The Moore's Law



Gordon Moore
Intel co-founder

The number of transistors integrated in a chip has doubled every 18-24 months (1975)

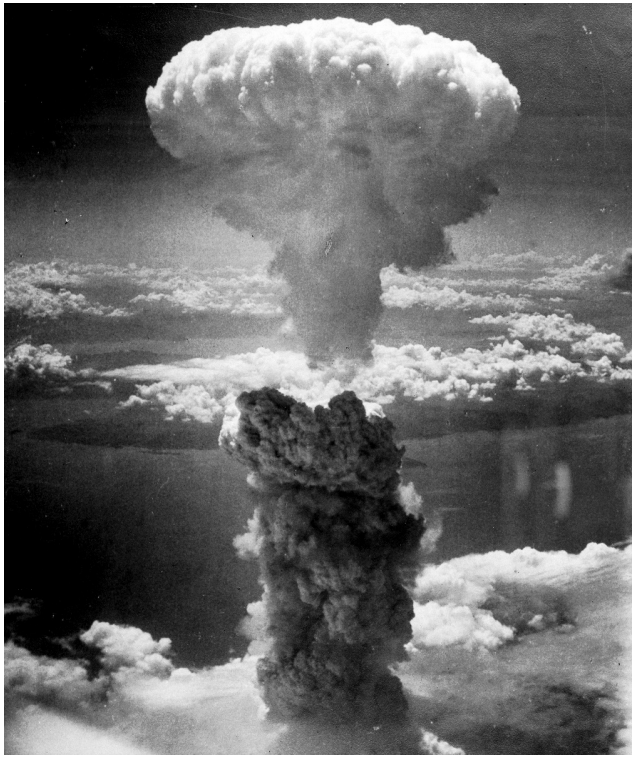
Intel processors

- As of Q1/2023
  - Raptor Lake: 13^{th} generation
  - 10 nm technology

	Intel® Core i9 Processors	Intel® Core i7 Processors	Intel® Core i5 Processors	Intel® Core i3 Processors
Max Turbo Frequency [GHz]	Up to 5.8	Up to 5.4	Up to 5.1	Up to 4.5
Intel® Turbo Boost Max Technology 3.0 Frequency [GHz]	Up to 5.7	Up to 5.4	n/a	n/a
Performance-core Max Turbo Frequency [GHz]	Up to 5.4	Up to 5.3	Up to 5.1	Up to 4.5

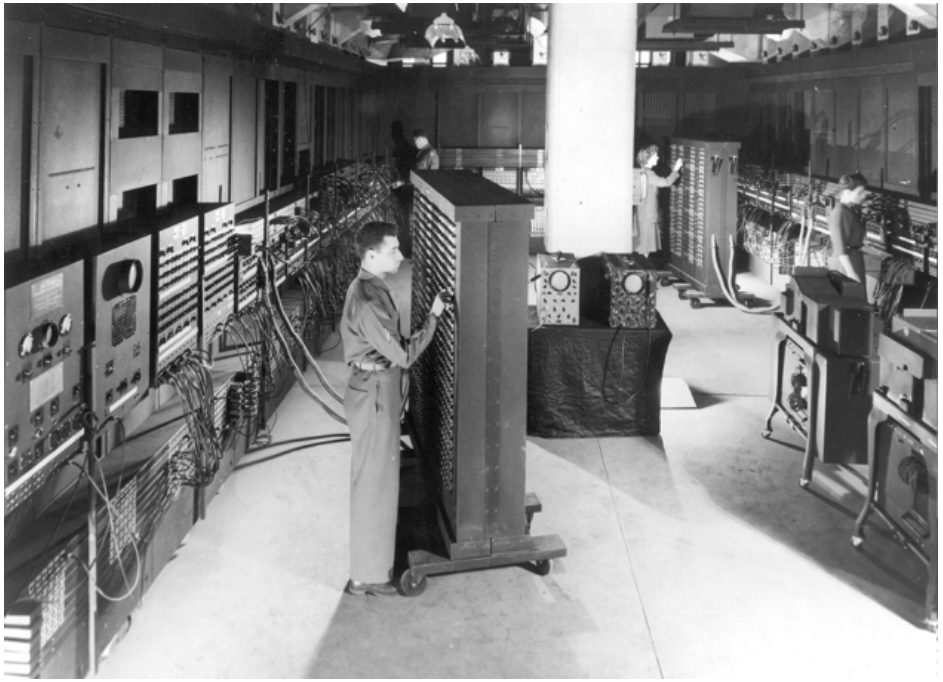
History…

- The first computer in the world



History…

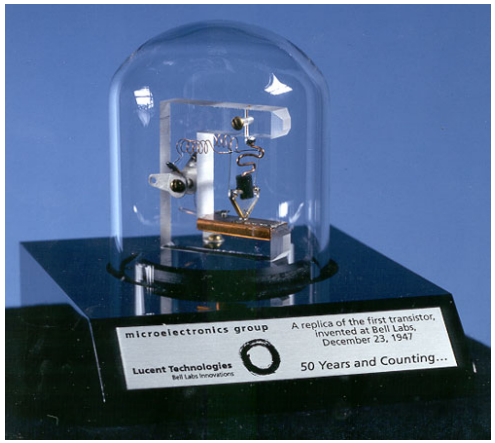
- Facts of ENIAC:
  - 30+ tons
  - 1,500+ square feet (140 square meter)
  - 18,000+ vacuum tubes
  - 140+ KW power
  - 5,000+ additions per second

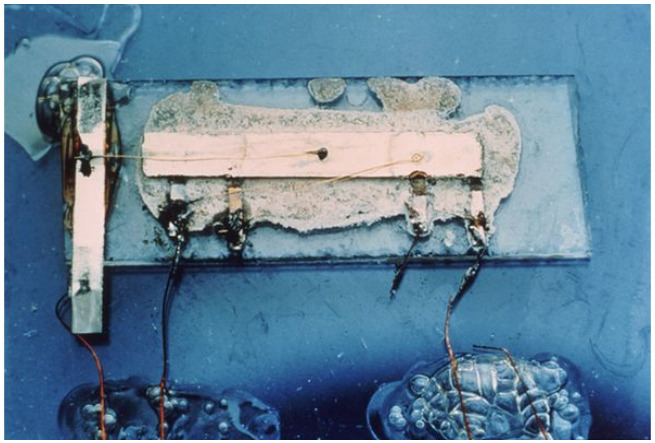


ENIAC: Electronic Numerical Integrator and Computer

A Brief History of Computers

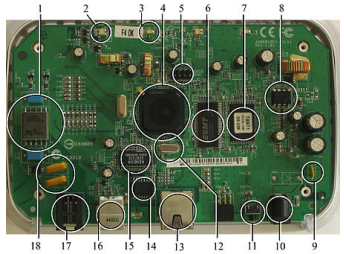
- The first generation
  - Vacuum tubes
  - 1946 – 1955
- The second generation
  - Transistors
  - 1955 – 1965
- The third generation
  - 1965 – 1980
  - Integrated circuits
- The current generation
  - 1980 - …
  - Personal computers
- What's the next?
  - *Quantum computers?*
  - *Memristor?*



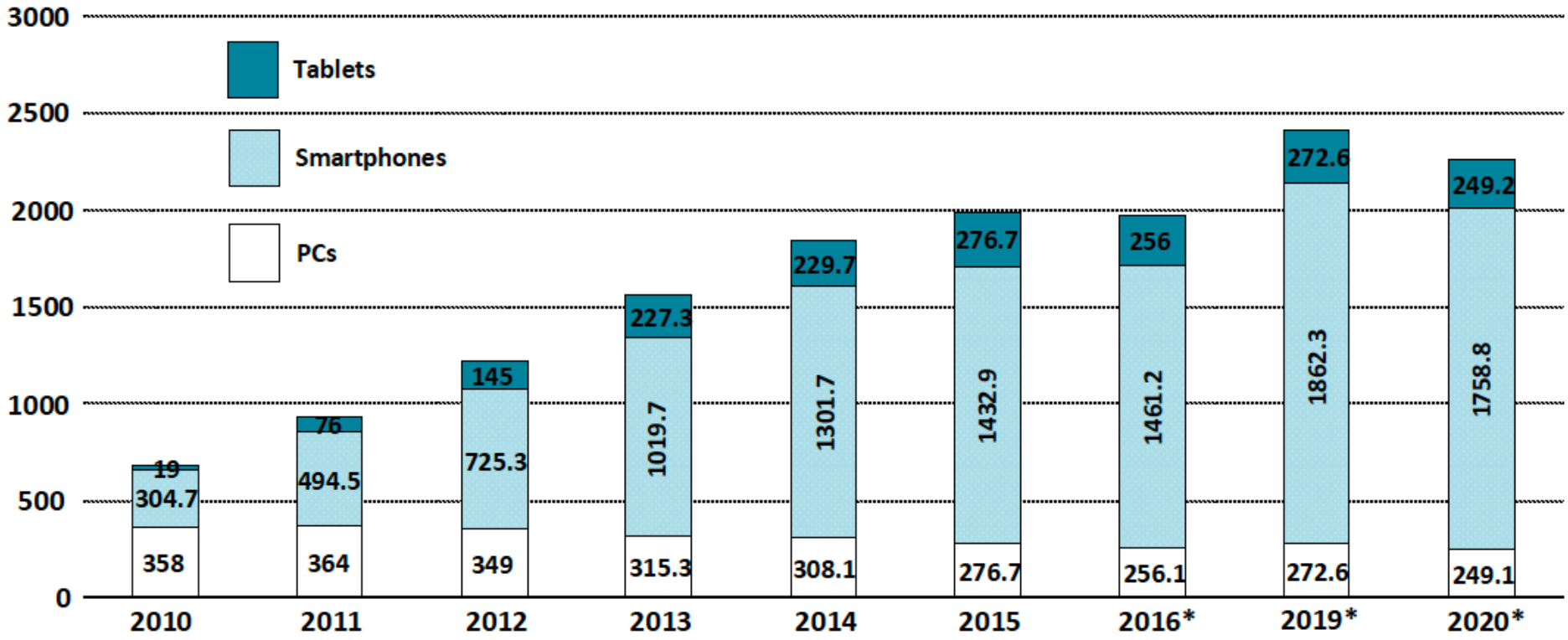


Classes of Computers

- Personal 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
- Supercomputers
  - High-end scientific and engineering calculations
  - Highest capability but represent a small fraction of the overall computer market
- Embedded computers
  - Hidden as components of systems
  - Stringent power/performance/cost constraints



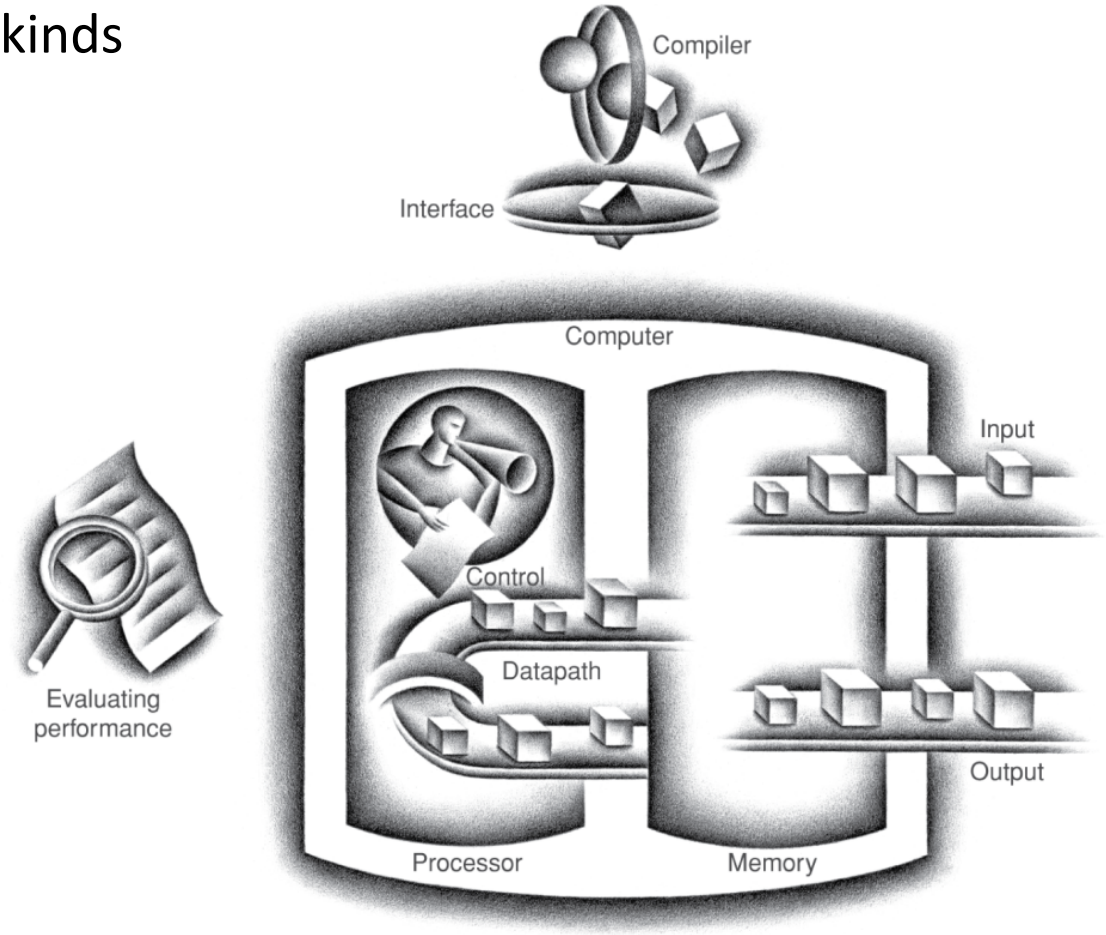
Post PC era



The number of devices (millions) shipped - source: statista.com

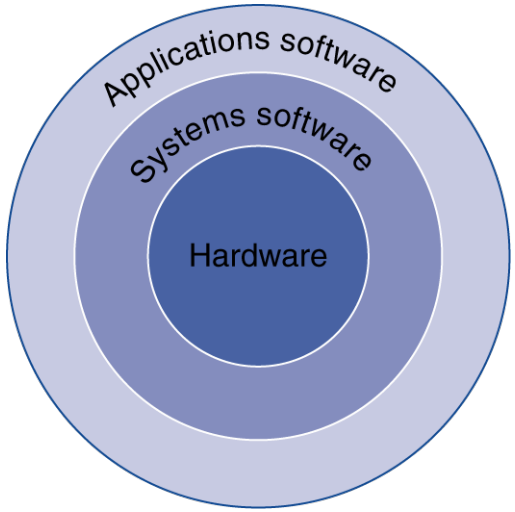
Modern computer components

- Same components for all kinds
- Components
  - Processor
    - Datapath
    - controller
  - Memory
    - Main memory
    - Cache
  - Input/Output
    - User-interface
    - Network
    - Storage



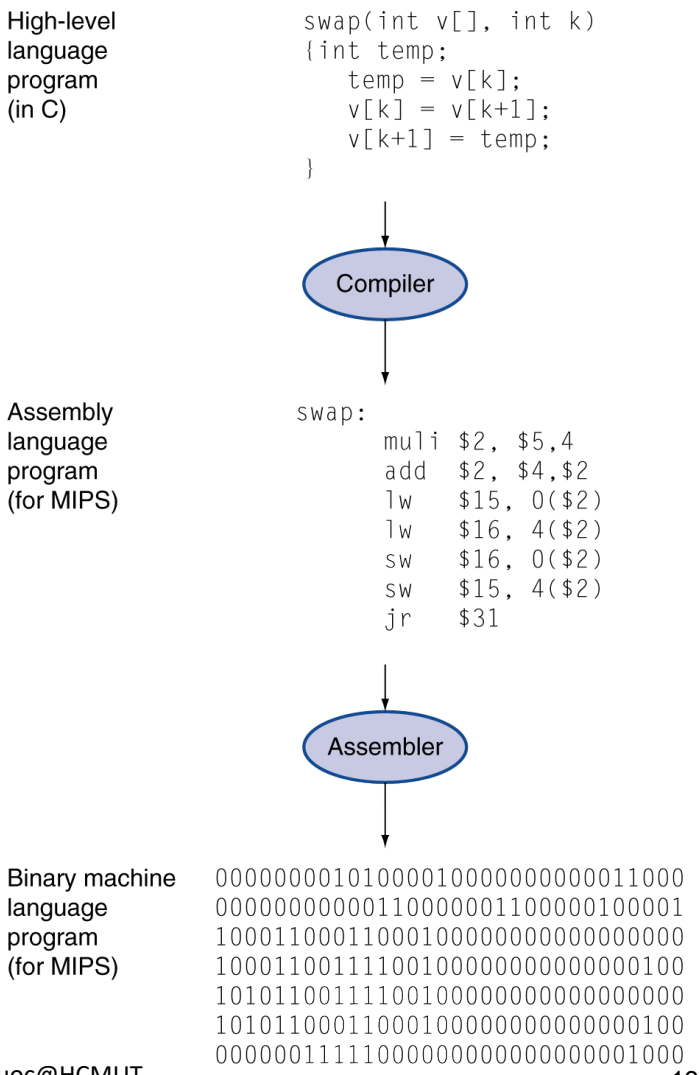
Below your program

- 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



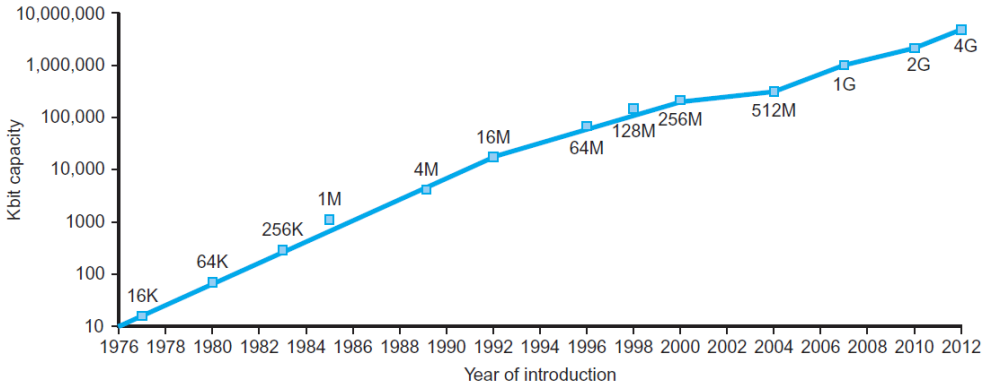
Levels of Program Code

- 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



Technology trends

- Thanks to electronics and material technologies
  - Increased capacity and performance
  - Reduced cost



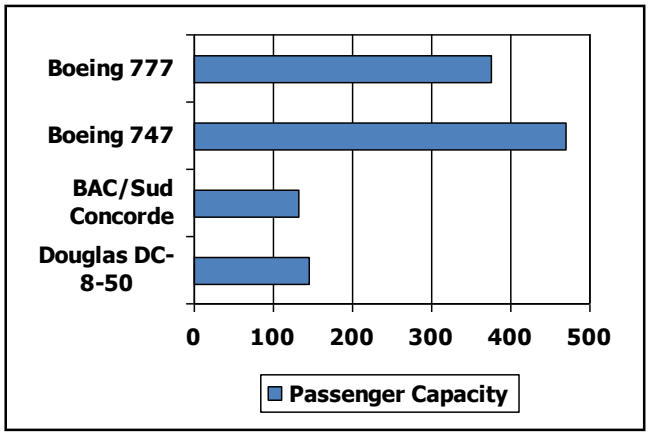
DRAM capacity

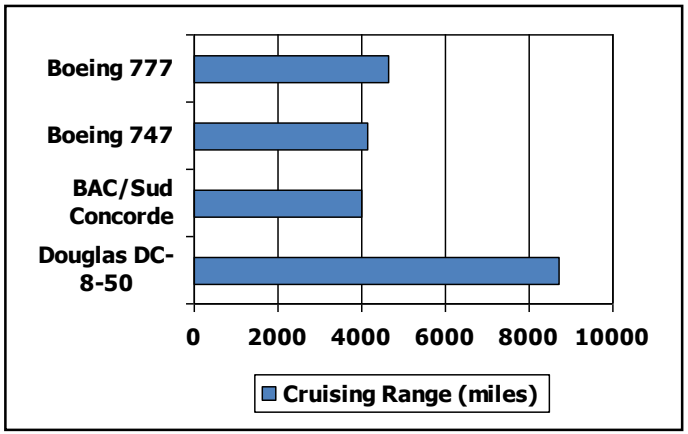
Year	Technology	Relative performance/cost
1951	Vacuum tube	1
1965	Transistor	35
1975	Integrated circuit (IC)	900
1995	Very large scale IC (VLSI)	2,400,000
2013	Ultra large scale IC	250,000,000,000

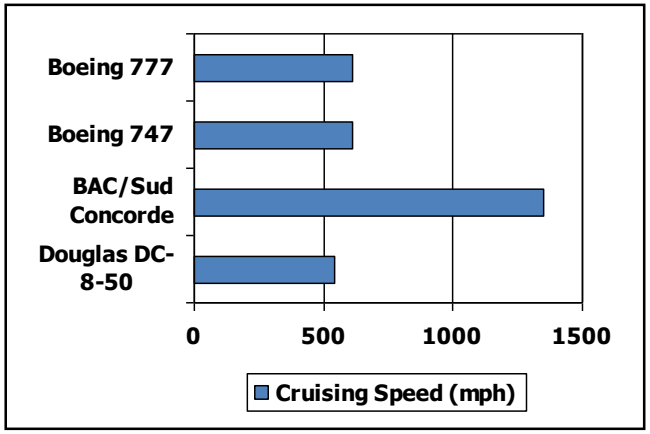
PERFORMANCE EVALUATION

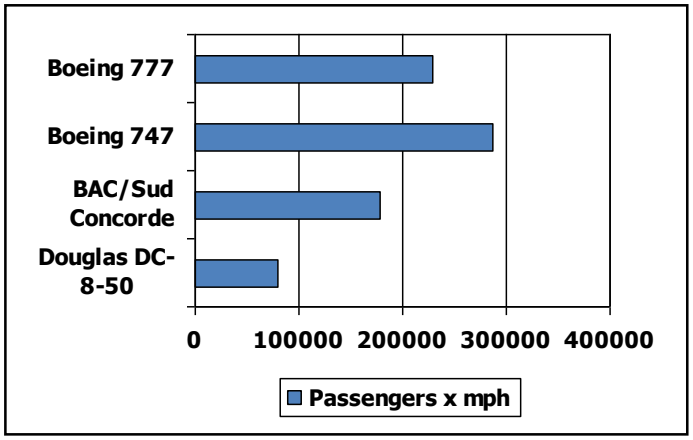
Defining performance

- 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

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

- Computer X is n times faster than Computer Y

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

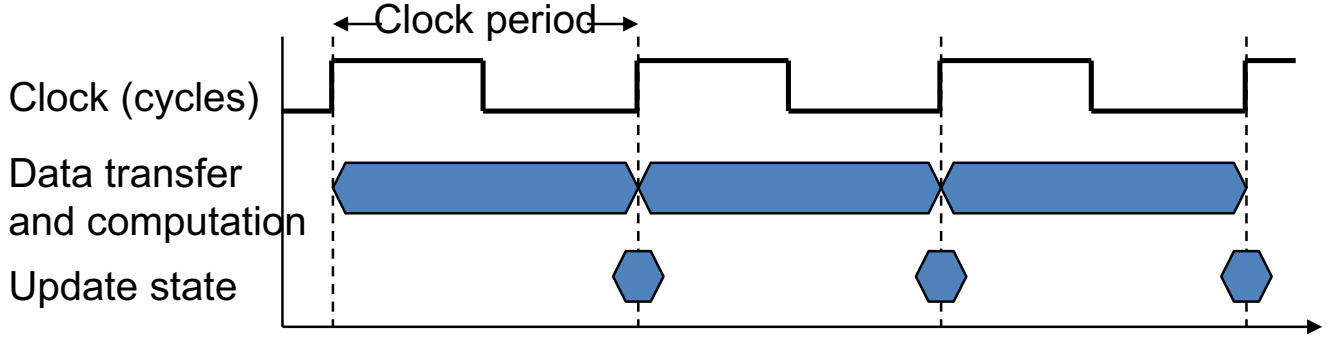
- Example: time take to run **a program**
  - 10s on A and 15s on B
  - A is $1.5\times$ faster than B because $\frac{\text{Execution}_B}{\text{Execution}_A} = \frac{15s}{10s} = 1.5\times$

Measuring 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 CPU time

- Operations of digital hardware (including CPU/processor) governed by a constant-rate clock



- Clock period (T): duration of a clock cycle
  - s, ms, μs, ns
- Clock rate/frequency ($F = \frac{1}{T}$): the number of cycles per second
  - Hz, KHz, MHz, GHz

CPU time

- Performance improved by
  - Reducing number of clock cycles
  - Increasing clock rate
  - Hardware designer must often trade off clock rate against cycle count

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

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?

$$\text{CPU Time}_A = \frac{\text{CPU Clock Cycles}_A}{\text{Clock Rate}_A} = \frac{\text{CPU Clock Cycles}_A}{2.0GHz} = 10s$$

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

$$\Rightarrow \text{Clock Rate}_B = 4.0GHz$$

Instruction count & CPI

- 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

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

$$\text{CPU Time} = \text{Instruction count} \times \text{Cycles per Instruction} \times \text{Clock Cycle Time}$$

$$= \frac{\text{Instruction count} \times \text{Cycles per Instruction}}{\text{Clock Rate}} = \frac{\text{IC} \times \text{CPI}}{\text{Clock Rate}}$$

Example

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

$$\begin{aligned}
\text{CPU Time}_A &= \text{IC}_A \times \text{CPI}_A \times \text{Cycle Time}_A \\
&= \text{IC} \times 2.0 \times 250ps \\
\text{CPU Time}_B &= \text{IC}_B \times \text{CPI}_B \times \text{Cycle Time}_B \\
&= \text{IC} \times 1.2 \times 500ps \\
\Rightarrow \frac{\text{CPU Time}_B}{\text{CPU Time}_A} &= \frac{\text{IC} \times 600ps}{\text{IC} \times 500ps} = 1.2 \times
\end{aligned}$$

Mixed instructions CPI

- CPI for instructions/operations may vary
  - e.g.,: multiplication takes more cycles than addition
- More precise CPU clock cycles should take instruction types into account

$$\text{Clock cycles} = \sum_{i=1}^{n} (\text{CPI}_i \times \text{Instruction count}_i)$$

- Weighted average CPI

$$\text{CPI} = \frac{\text{Clock cycles}}{\text{Instruction count}} = \sum_{i=1}^{n} (\text{CPI}_i \times \frac{\text{Instruction count}_i}{\text{Instruction count}})$$

Example

- **Question**: two implementations of an application that use instructions in classes A, B, and C as follows. Which one is better?
  - Implementation 1 uses 2 A, 1 B, and 2 C
  - Implementation 2 uses 4 A, 1 B, and 1 C
  - CPIs for A, B, and C are 1, 2, and 3, respectively
- **Answer**:
  - Implementation 1: clock cycles$_1 = 2 \times 1 + 1 \times 2 + 2 \times 3 = 10$
    - IC = 5, **wCPI = 2.0**
  - Implementation 2: clock cycles$_2 = 4 \times 1 + 1 \times 2 + 1 \times 3 = 9$
    - IC = 6, **wCPI = 1.5**

Exercise

- A program is executed on a 2 GHz CPU. The program consists of 1000 instructions in which:
  - 30% load/store instructions, CPI = 2.5
  - 10% jump instructions, CPI = 1
  - 20% branch instructions, CPI = 1.5
  - The rest are arithmetic instructions, CPI = 2.0

a) What is execution time (CPU time) of the program?

b) What is the weighted average CPI of the program?

c) If load/store instructions are improved so that their execution time is reduced by a factor of 2, what is the speed-up of the system?

Exercise 2

- Suppose we have the following measurement:
  - Frequency of FP instructions = 25%
  - Average CPI of FP operations = 4.0
  - Average CPI of other instructions = 1.33
  - Frequency of FPSQR= 2%
  - CPI of FPSQR = 20
- Assume that the two design alternatives are to decrease the CPI of FPSQR to 2 or to decrease the average CPI of all FP operations to 2.5.
- Compare these two design alternatives using the processor performance equation.

Exercise 3

- Assume a program requires the execution of:
  - 50 x 106 FP instructions,
  - 110 × 106 INT instructions,
  - 80 x 106 L/S instructions,
  - and 16 × 106 branch instructions.
- The CPI for each type of instruction is 1, 1, 4, and 2, respectively. Assume that the processor has a 2 GHz clock rate.

a) By how much must we improve the CPI of FP instructions if we want the program to run two times faster?

b) By how much must we improve the CPI of LS instructions if we want the program to run two times faster?

c) By how much is the execution time of the program improved if the CPI of INT and FP instructions is reduced by 40% and the CPI of L/S and Branch is reduced by 30%?

Performance summary

- **The BIG picture** (take home message)

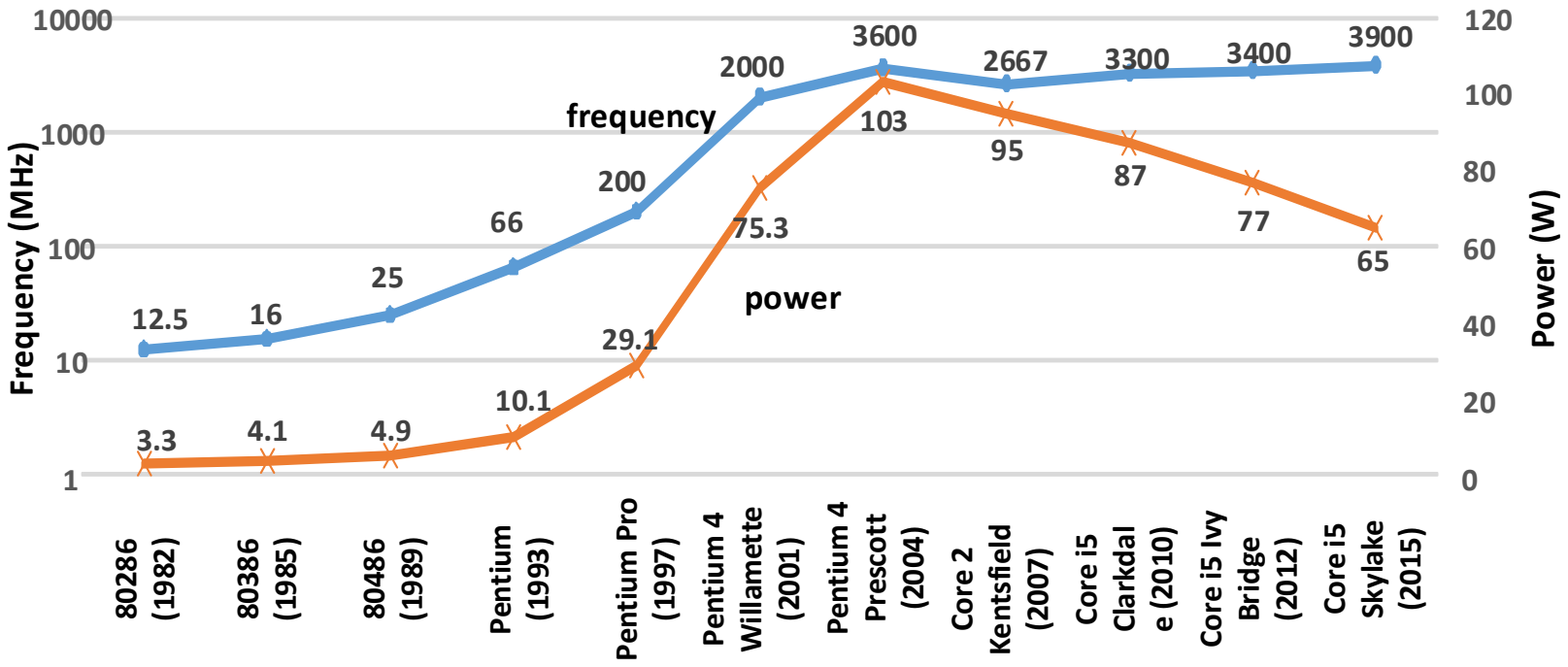
$$\text{CPU time} = \frac{\text{Instructions}}{\text{Program}} \times \frac{\text{Clock cycles}}{\text{Instruction}} \times \frac{\text{Seconds}}{\text{Clock cycle}}$$

- Performance depends on
  - Algorithm: IC, possibly CPI
  - Programming language: IC, CPI
  - Compiler: IC, CPI
  - Instruction set architecture: IC, CPI, T

Power trends

- In CMOS technology

$$\text{Power}(P) = \text{Capacitive load} \times \text{Voltage}^2 \times \text{Clock rate}$$



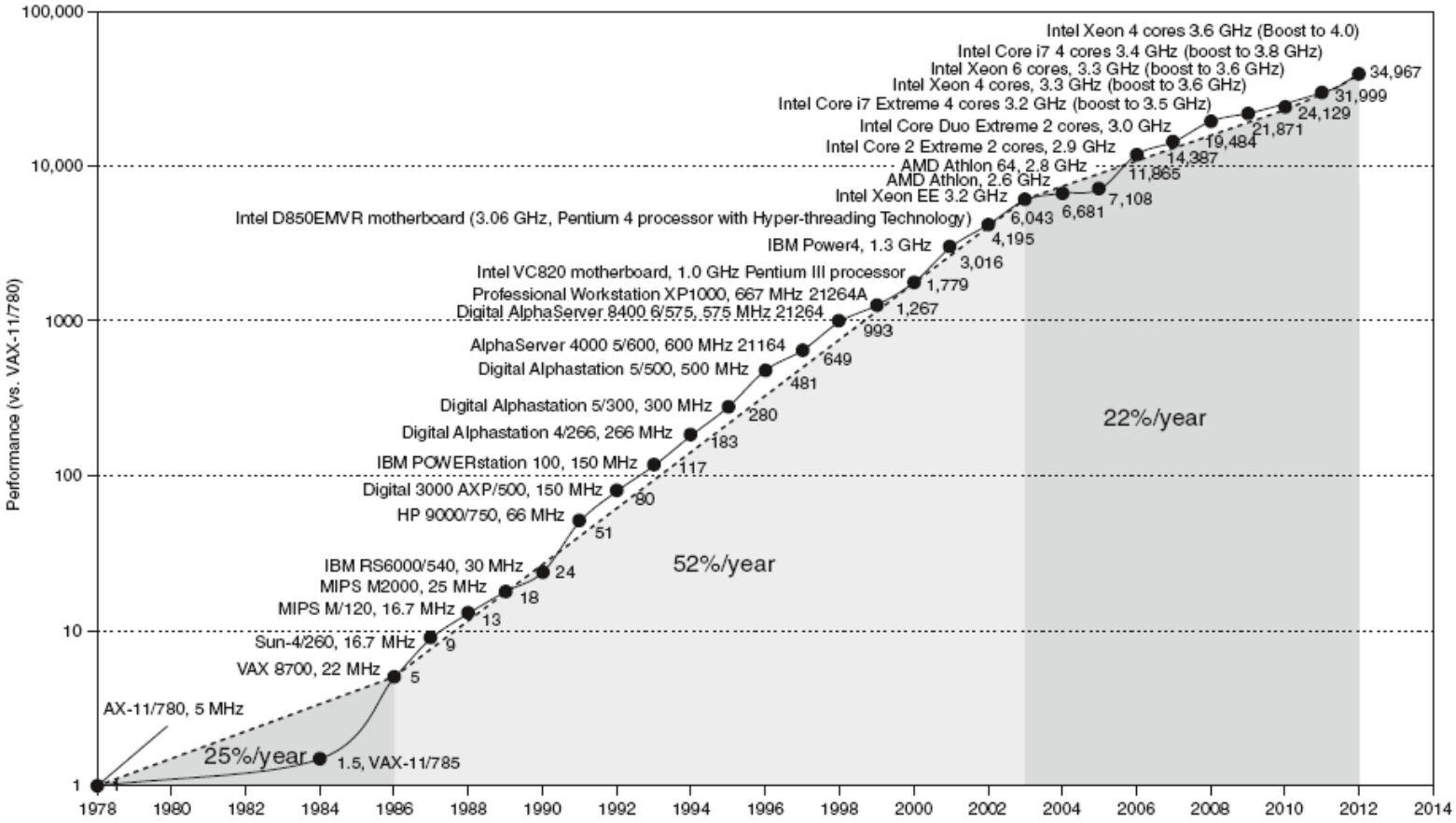
Reducing power

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

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

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

Multiprocessors



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)

Example

- Intel core i7 920 results with CINT 2006

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

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