



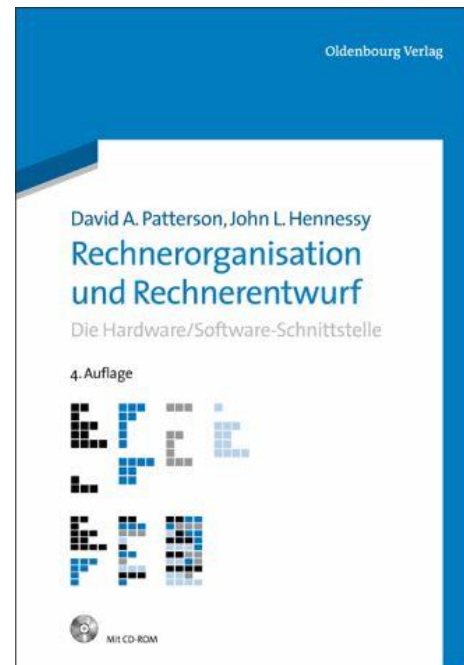
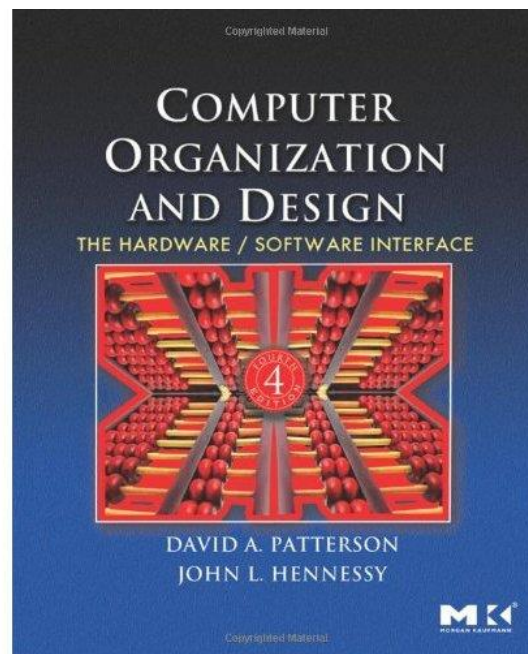
Informatik der Systeme – Chapter 4: Computer Abstractions and Technology

Prof. Dr. Michael Menth

<https://kn.inf.uni-tuebingen.de>



- ▶ This set of slides is an adaptation of Prof. Mary Jane Irwin's lecture notes, <http://www.cse.psu.edu/research/mdl/mji/>
- ▶ Adapted from Patterson & Hennessy: "Computer Organization and Design", 4th Edition, © 2008, MK
- ▶ German translation: Patterson & Hennessy: "Rechnerorganisation und Rechnerentwurf"





- ▶ Application binary interface
- ▶ Components of a computer
- ▶ Classes of computers
- ▶ Technological evolution
- ▶ Performance considerations



- ▶ Application binary interface
- ▶ Components of a computer
- ▶ Classes of computers
- ▶ Technological evolution
- ▶ Performance considerations



► 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;
}

```

► Assembly language program (for MIPS)

```

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

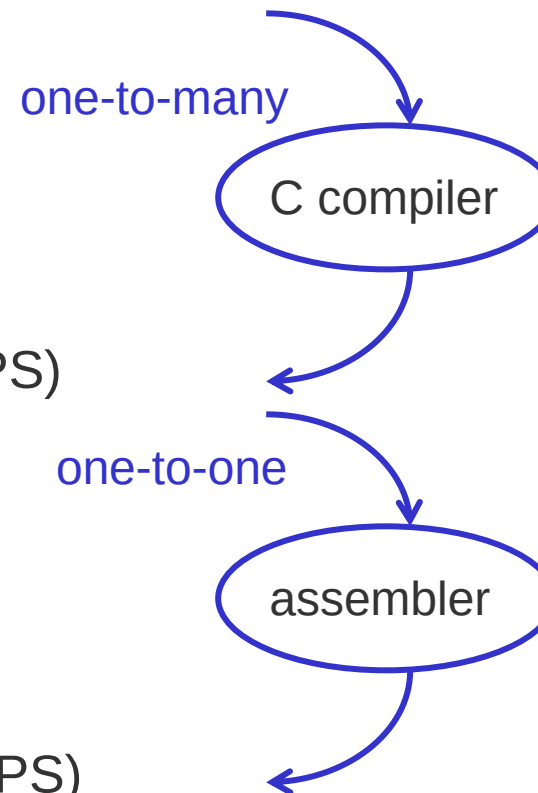
```

► Machine (object, binary) code (for MIPS)

```

000000 00000 00101 0001000010000000
000000 00100 00010 0001000000100000
. . .

```

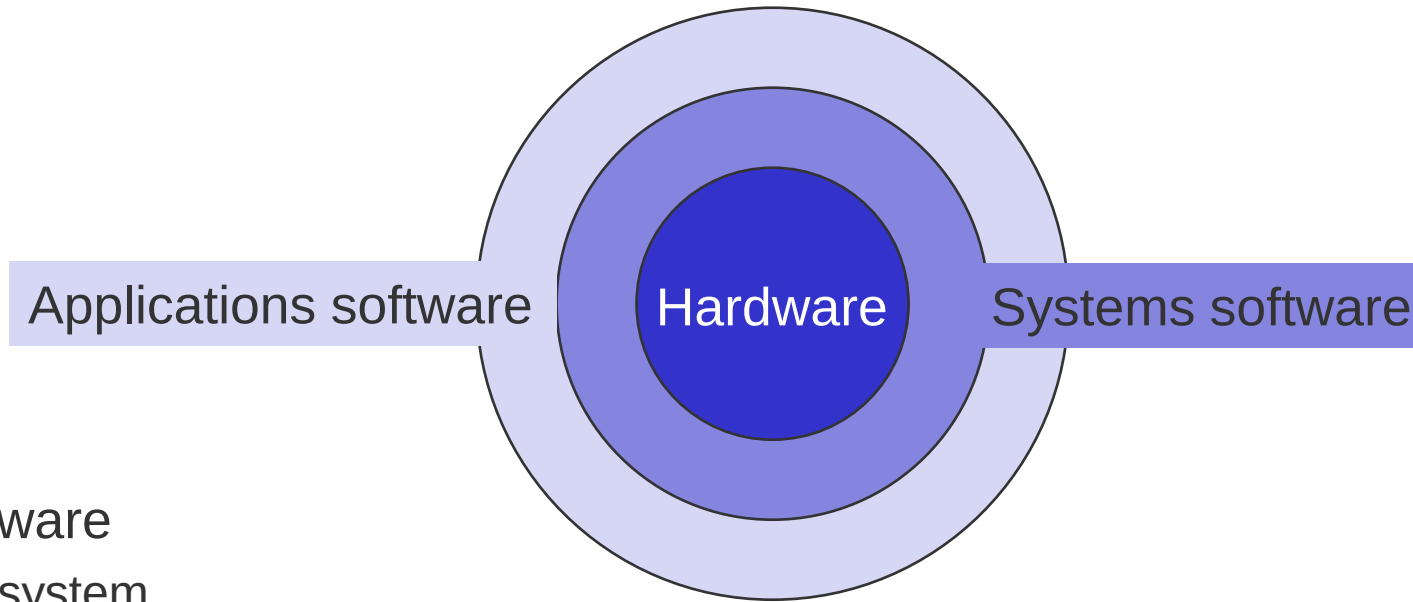




► Higher-level languages

- Allow the programmer to think in a more natural language and for their intended use (Fortran for scientific computation, Cobol for business programming, Lisp for symbol manipulation, Java for App development, ...)
- Improve programmer productivity – more understandable code that is easier to debug and validate
- Improve program maintainability
- Allow programs to be independent of the computer on which they are developed
 - Compilers and assemblers can translate high-level language programs to the binary instructions of any machine
- Compilers produce very efficient assembly code optimized for the target machine

► As a result, very little programming is done today at the assembler level



► System software

■ Operating system

- Supervising program that interfaces the user's program with the hardware (e.g., Linux, macOS, Windows)
- Handles basic input and output operations
- Allocates storage and memory
- Provides for protected sharing among multiple applications
- Today: Often also Hypervisor-Layer above OS

■ Compiler

- Translates programs written in a high-level language (e.g., C, Java) into instructions that the hardware can execute



► Instruction set architecture (ISA)

- Abstract interface between the hardware and the lowest level software (aka basic instruction set)
- Encompasses all the information necessary to write a machine language program, including instructions, registers, memory access, I/O, ...
- Enables implementations of varying cost and performance to run identical software

► Application binary interface (ABI)

- Combination of the user portion of the instruction set and the operating system interface
- Used by application programmers
- Defines a standard for binary portability across computers



- ▶ Application binary interface
- ▶ Components of a computer
- ▶ Classes of computers
- ▶ Technological evolution
- ▶ Performance considerations

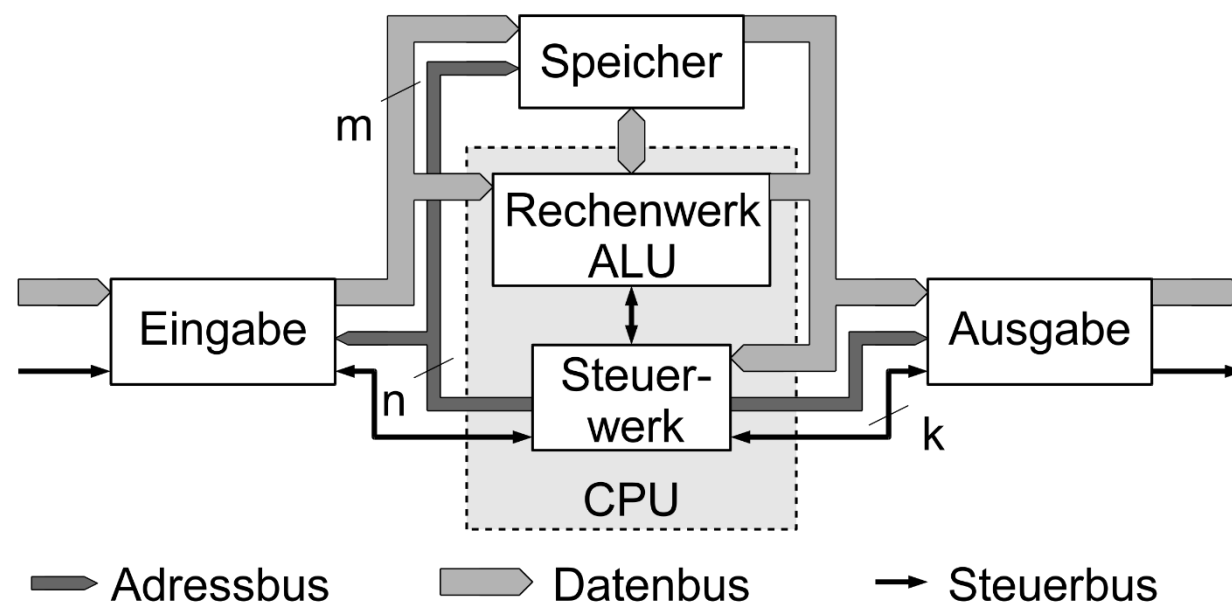
Components of a Computer

► Five classic components of a computer

- Input
- Output
- Memory
- Datapath
- Control

► Processor

- Central processing unit (CPU)
- Control
- Datapath



von Neumann architecture

https://de.wikipedia.org/wiki/Von-Neumann-Architektur#/media/Datei:%22von_Neumann%22_Architektur_de.svg

► [https://de.wikipedia.org/wiki/Prozessor#Hauptprozessor.2C CPU und Prozessorkern](https://de.wikipedia.org/wiki/Prozessor#Hauptprozessor.2C_CPU_und_Prozessorkern)



What's inside a computer?

Power supply unit



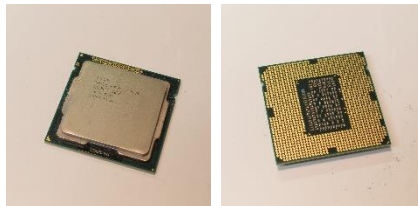
Main memory



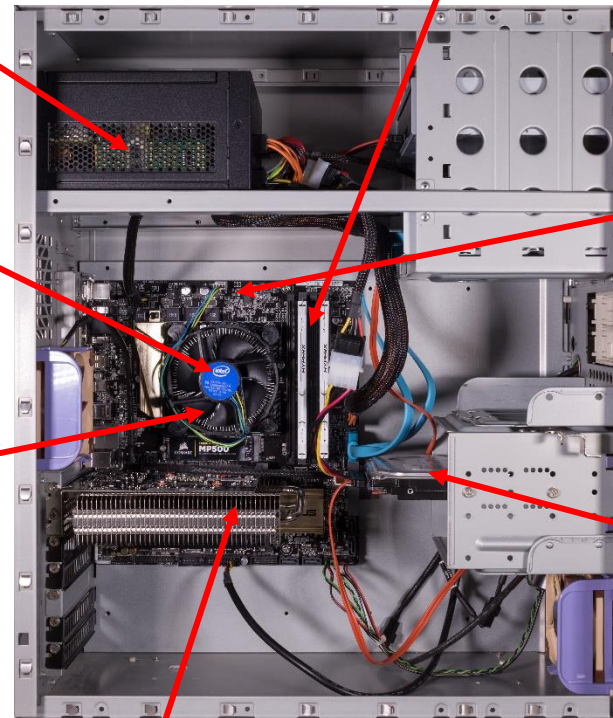
Mainboard



CPU fan



CPU



Storage

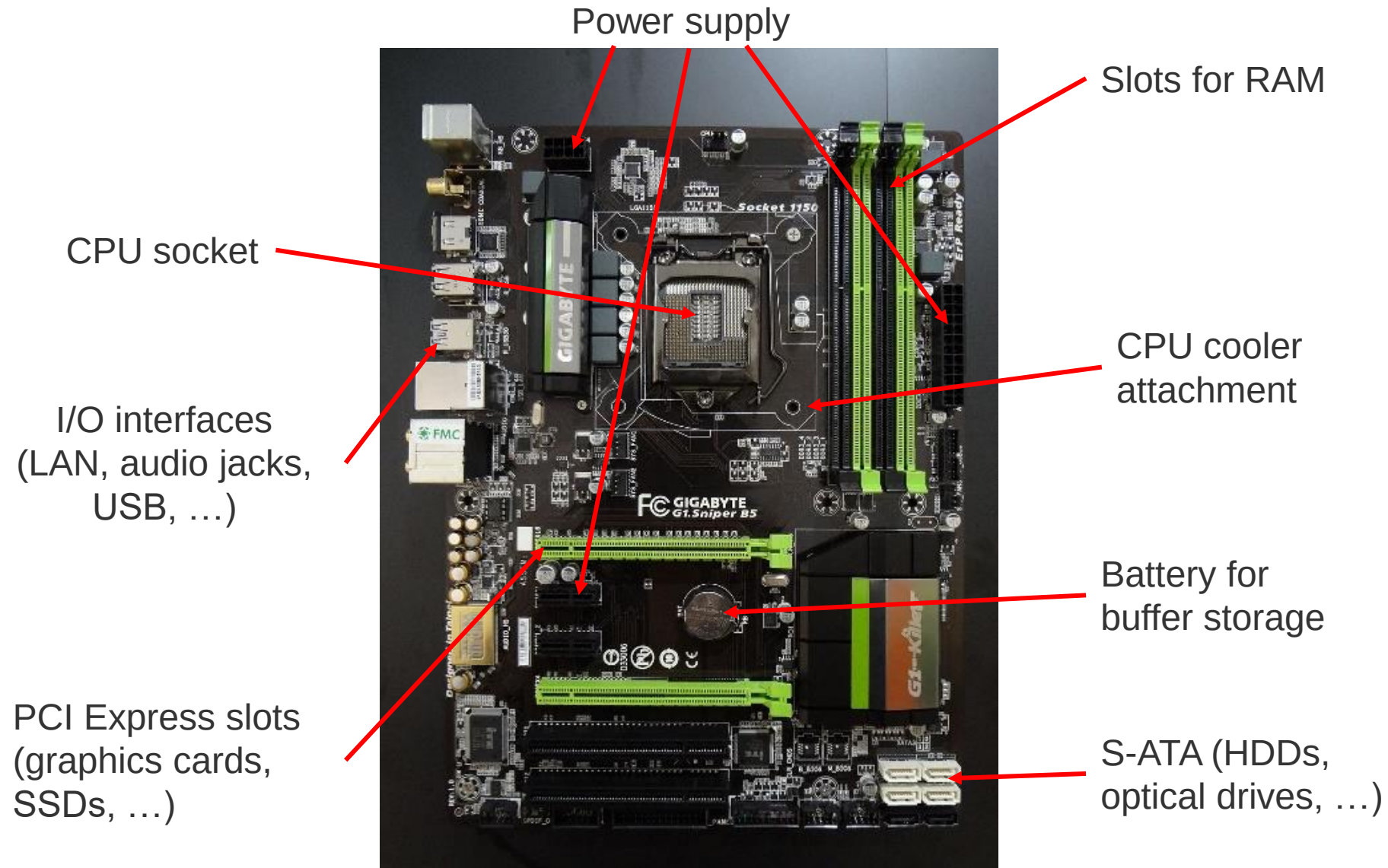
Graphics card



[https://commons.wikimedia.org/wiki/Category:DDR4_RAM#/media/File:RAM_Module_\(SDRAM-DDR4\).jpg](https://commons.wikimedia.org/wiki/Category:DDR4_RAM#/media/File:RAM_Module_(SDRAM-DDR4).jpg)
[https://commons.wikimedia.org/wiki/File:Intel_Core_i7-2600_SR00B_\(16339769307\).jpg](https://commons.wikimedia.org/wiki/File:Intel_Core_i7-2600_SR00B_(16339769307).jpg)
[https://commons.wikimedia.org/wiki/File:Pins_on_Core_i7-2600_SR00B_\(16339768957\).jpg](https://commons.wikimedia.org/wiki/File:Pins_on_Core_i7-2600_SR00B_(16339768957).jpg)
https://commons.wikimedia.org/wiki/Category:ATX_computer_motherboards#/media/File:Gigabyte_G1.Sniper_B5_20130910.jpg
https://commons.wikimedia.org/wiki/Category:ATI_Radeon_HD_5000_series_video_cards#/media/File:Sapphire_Radeon_HD_5870-front_oblique_PNr%C2%B00310.jpg
https://commons.wikimedia.org/wiki/Category:Solid_State_Drive#/media/File:Samsung_MZ-N5E500_and_Samsung_MZ-M5E250_20170314.jpg
https://commons.wikimedia.org/wiki/Category:Computer_cooling_fans_with_seven_blades#/media/File:AMD_Wraith_Spire_cooler.jpg
https://commons.wikimedia.org/wiki/Category:Desktop_computers#/media/File:2017_mid_range_pc_in_late_1990s_case.jpg

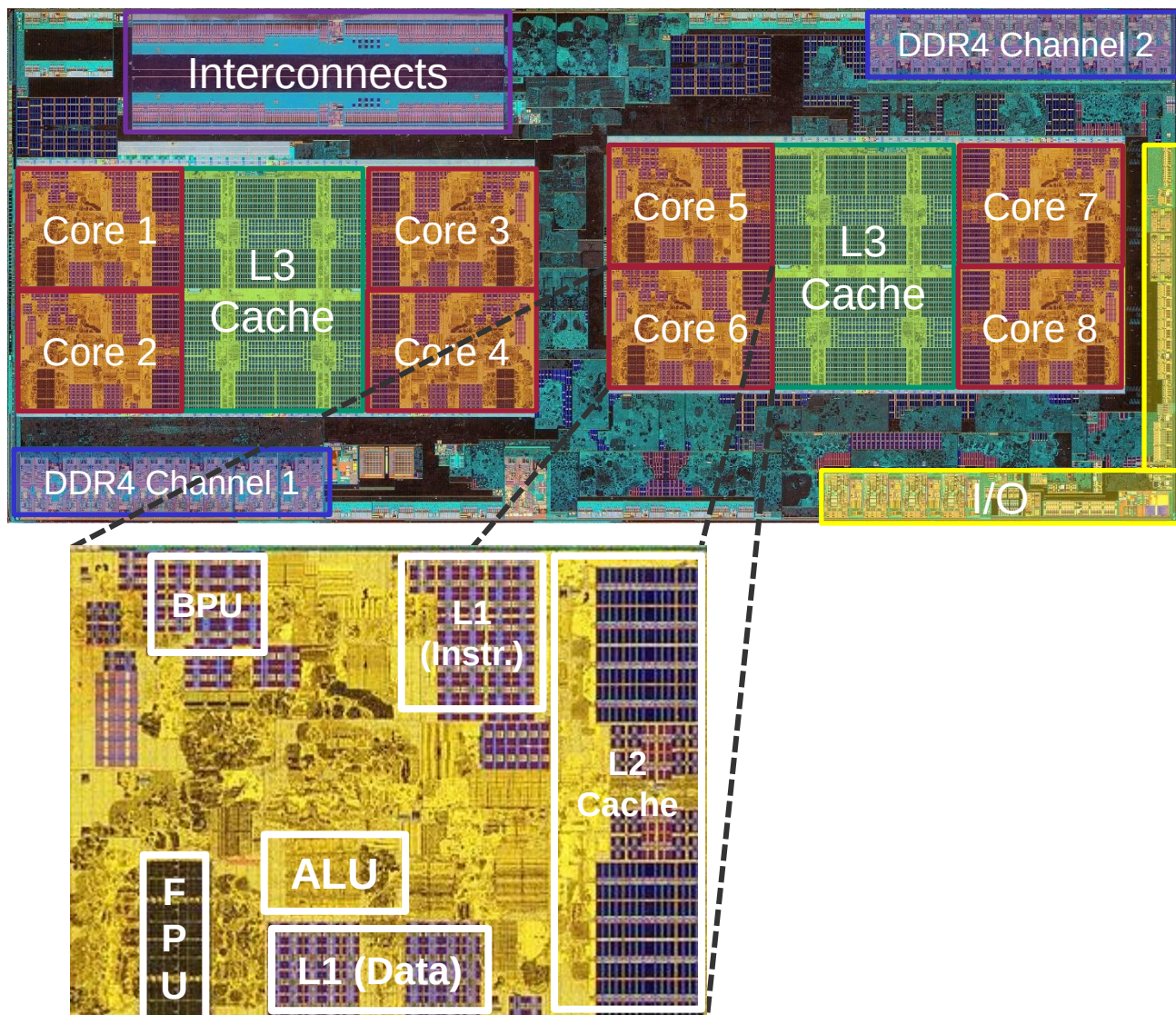


What's inside a computer?





AMD's Zen 2 Architecture



- ▶ Eight cores on two CCX
- ▶ 3.6 - 4.4 GHz clock rate
- ▶ 7 nm technology
- ▶ Three levels of caches
 - ▶ L1 & L2 per Core
 - ▶ L3 shared
- ▶ Memory controller on chip
 - ▶ Two channels for DDR4 RAM



- ▶ Application binary interface
- ▶ Components of a computer
- ▶ **Classes of computers**
- ▶ Technological evolution
- ▶ Performance considerations



► Desktop computers

- Designed to deliver good performance to a single user at low cost usually executing 3rd party software, usually incorporating a graphics display, a keyboard, and a mouse

► Servers

- Used to run larger programs for multiple, simultaneous users typically accessed only via a network and that places a greater emphasis on dependability and (often) security

► Supercomputers

- A high-performance, high-cost class of servers with hundreds to ten thousands of processors, **petabyte** of memory and **hundreds of petabytes** of storage that are used for high-end scientific and engineering applications

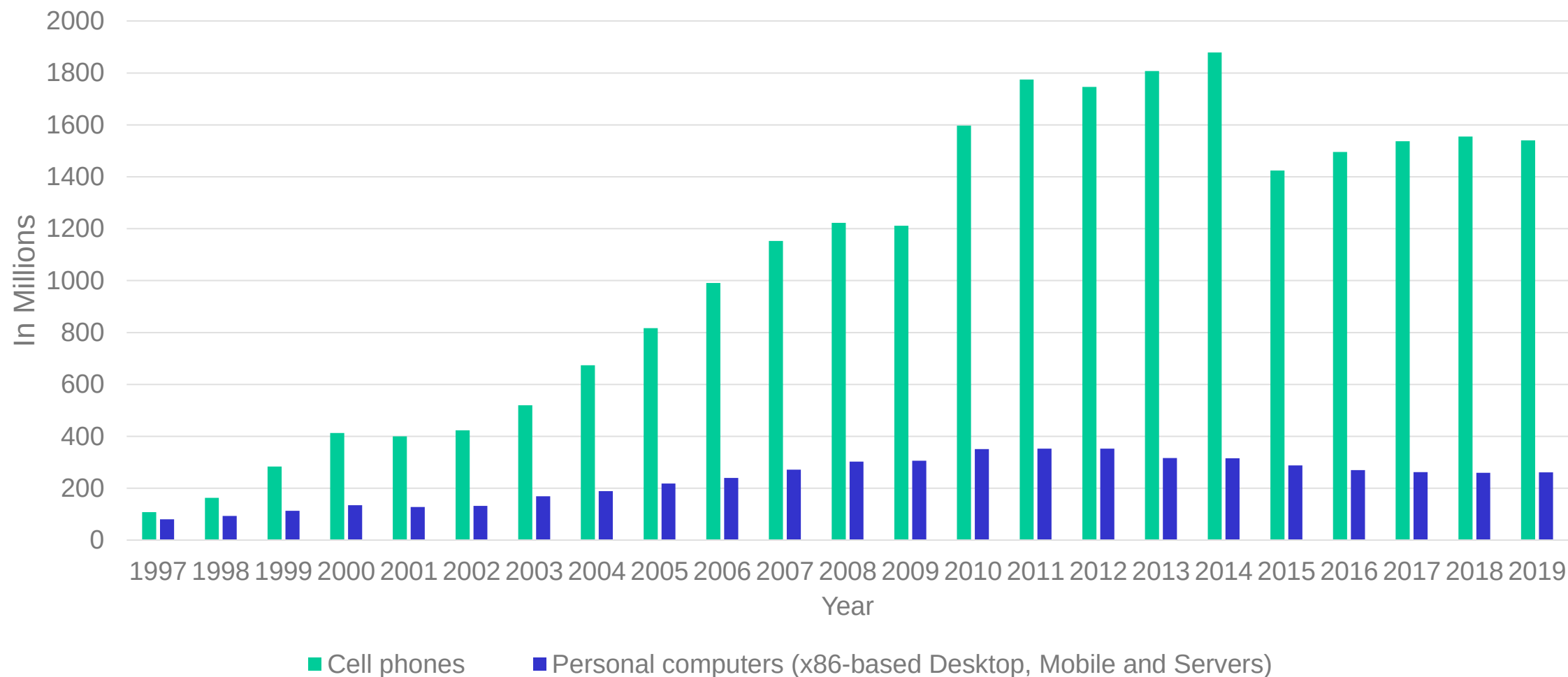
► Embedded computers (processors)

- A computer inside another device used for running one predetermined application



Growth in Cell Phone Sales (Embedded)

embedded growth >> desktop growth





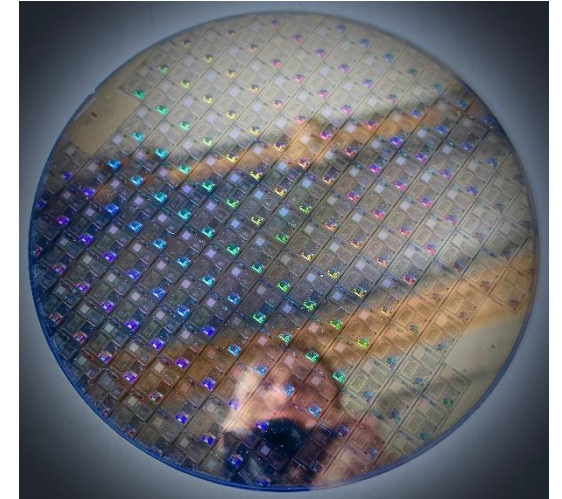
- ▶ The largest class of computers
- ▶ Span the widest range of applications and performance
- ▶ Often have
 - Minimum performance requirements
 - Stringent cost limitations
 - Stringent power consumption limitations
 - Low tolerance for failure



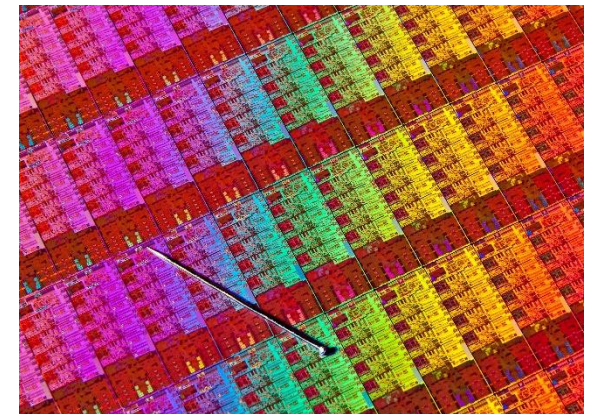
- ▶ Application binary interface
- ▶ Components of a computer
- ▶ Classes of computers
- ▶ Technological evolution
- ▶ Performance considerations



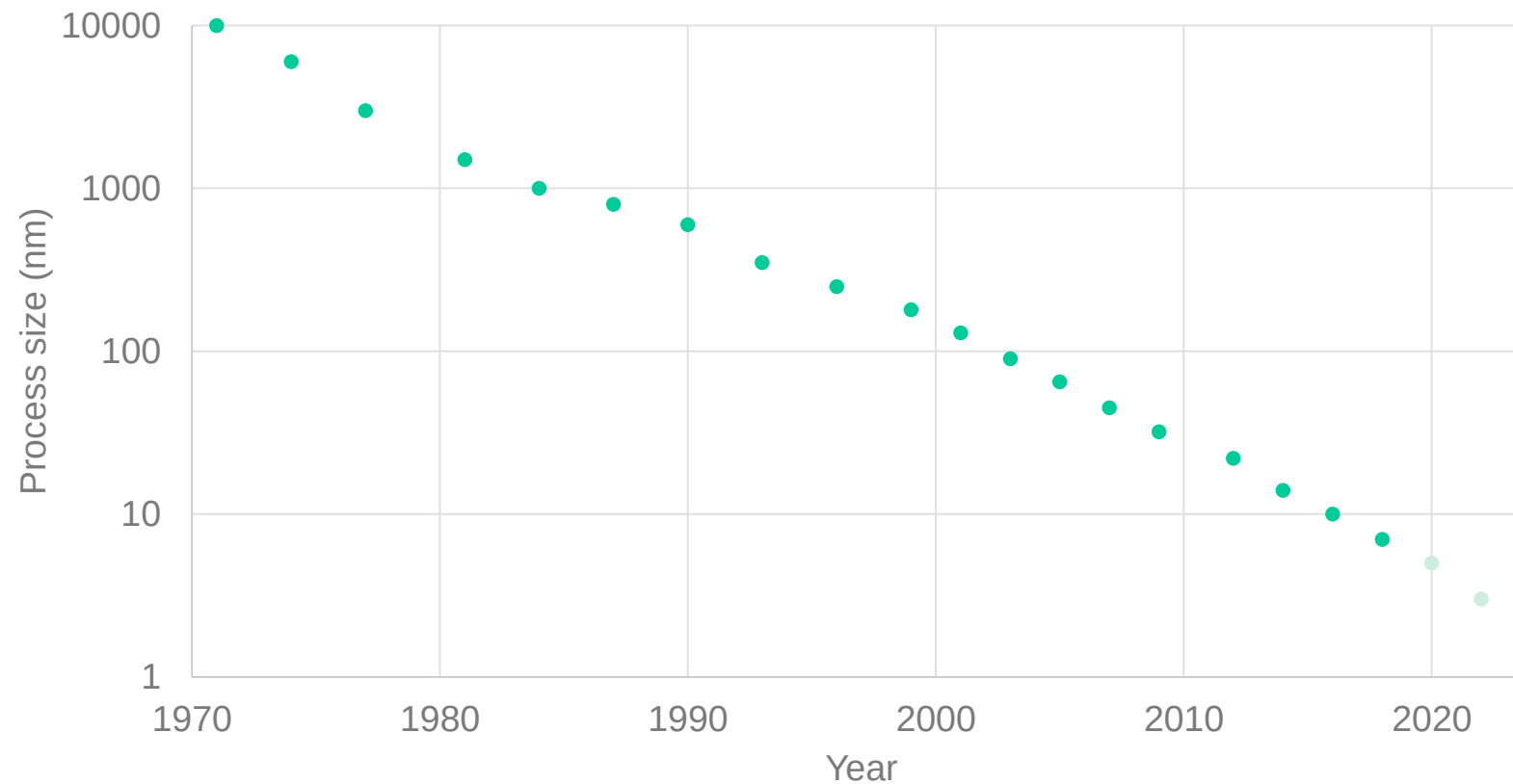
- ▶ Computer chips are manufactured with wafers
 - Flat pieces of sliced silicon crystals
 - Patterns of chemicals are placed on the wafer in multiple steps
 - Creating conductors, transistors, insulators, ...
 - Wafer is diced in pieces, so called dies (chips)
 - Not all dies are flawless
 - Percentage of flawless dies is called “yield”



- ▶ What about the costs?
 - $\text{Cost per die} = \frac{\text{Cost per wafer}}{\text{Number of dies} \cdot \text{yield}}$
 - Straightforward simple math
 - $\text{Number of dies} \approx \frac{\text{Wafer area}}{\text{Die area}}$
 - Approximation (border areas are useless)
 - $\text{Yield} = \frac{1}{\left(1 + \left(\text{Defects per area} \cdot \frac{\text{Die area}}{2}\right)\right)^2}$
 - Empirical observation
 - Costs increase quadratically with die area



[https://commons.wikimedia.org/wiki/Category:Wafers#/media/File:Ghost_in_the_Quantum_Machine_\(25215168597\).jpg](https://commons.wikimedia.org/wiki/Category:Wafers#/media/File:Ghost_in_the_Quantum_Machine_(25215168597).jpg)
https://commons.wikimedia.org/wiki/Category:Wafers#/media/File:Haswell_Chip.jpg



- Smallest realizable size (typically gate length of FETs)
 - By 2020: 7nm (5nm is planned)



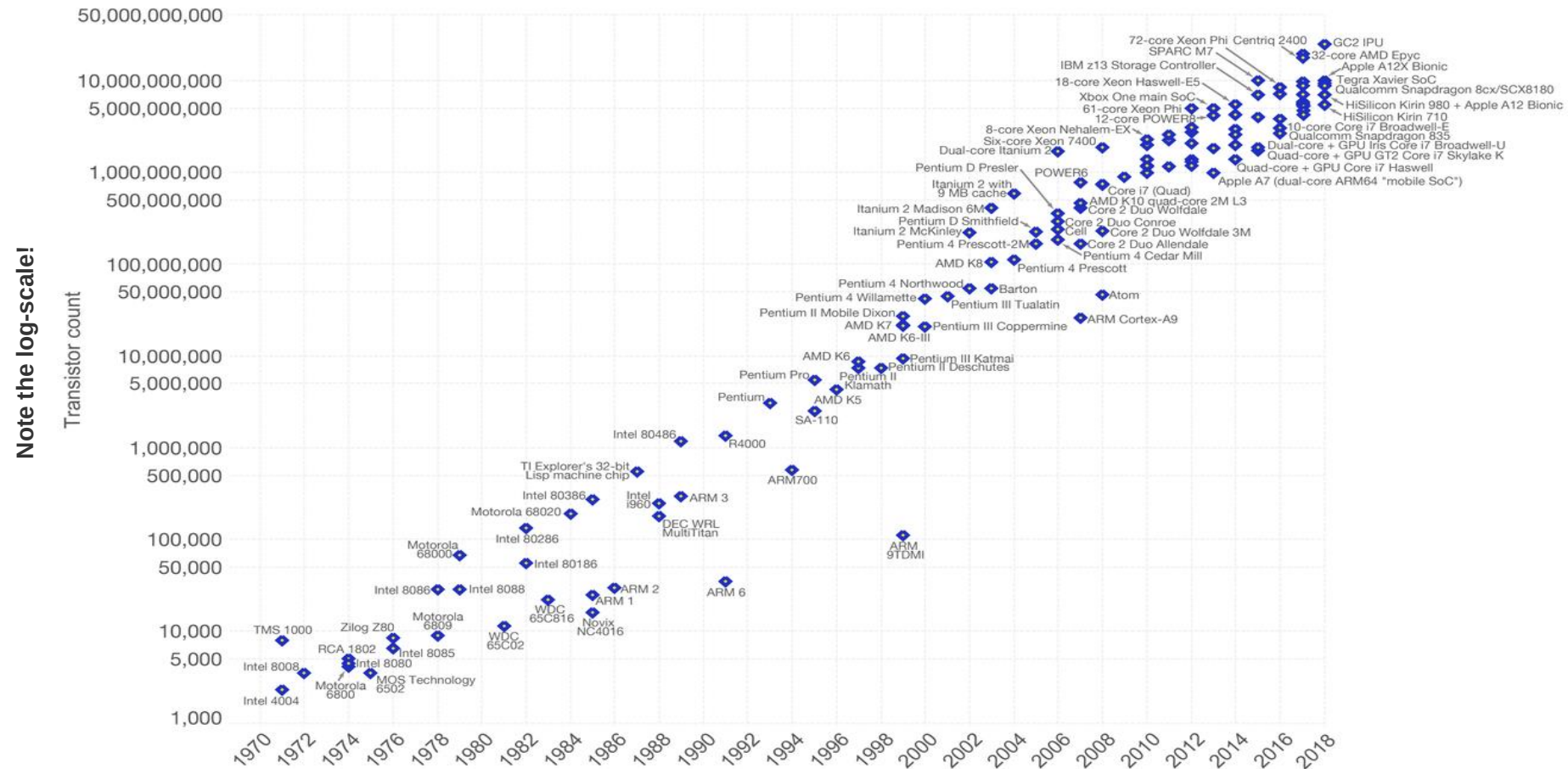
- ▶ Number of transistors in ICs will doubles every two years
 - Forecast by Gordon Moore, co-founder of Intel, in 1965
 - Proved accurate for 50 years!

- ▶ Self-fulfilling prophecy
 - Manufacturers tried to archieve Moore's Law...
 - ...which lead to the fulfilling of the Law

- ▶ Also holds for many other things
 - Storage size, network capacity, pixels per dollar, ...

Moore's Law – The number of transistors on integrated circuit chips (1971-2018)

Moore's law describes the empirical regularity that the number of transistors on integrated circuits doubles approximately every two years. This advancement is important as other aspects of technological progress – such as processing speed or the price of electronic products – are linked to Moore's law.

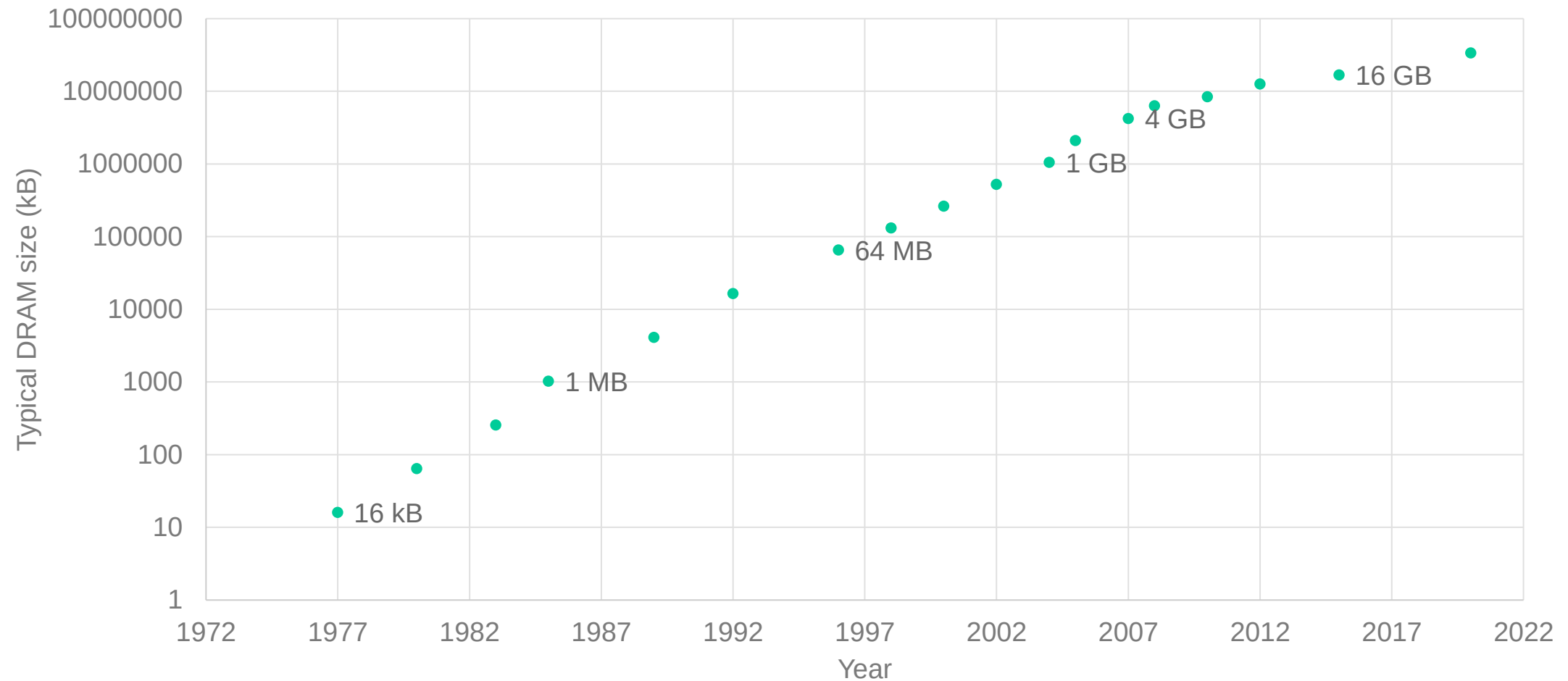


Data source: Wikipedia (https://en.wikipedia.org/wiki/Transistor_count)
The data visualization is available at [OurWorldinData.org](https://ourworldindata.org). There you find more visualizations and research on this topic.

Licensed under CC-BY-SA by the author Max Roser.



► Moores Law also holds for many other things:





► By now

- Smaller process size = better
- More transistors on same space
 - Chip price grows with space
 - More parallelism possible
- Lower capacitance
 - Faster switching of transistors...
 - ...while using less energy

► Necessarily ends by the laws of physics

- There are no infinitesimal small structures
- There is no exponential growth/decrease in real life which can sustain forever!

► Smaller transistors have downsides

- More leakage power while not switching
- More expenses and advanced technology needed
- Soon: Undesired quantum-effects are possible ($< 3\text{nm}$)

► Moore's Law does still hold, but its end was prognosed many times

- Current prognosis: mid 2020s
- New solutions will follow
 - New materials (Graphene, Gallium, Germanium, ...)
 - New process technology (3d ICs)
 - New ideas (Quantum computing)
- Computation power will increase further

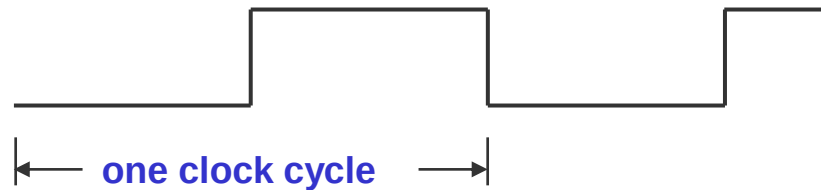


- ▶ Application binary interface
- ▶ Components of a computer
- ▶ Classes of computers
- ▶ Technological evolution
- ▶ Performance considerations



► Machines proceed in clock cycles (CCs)

- Given in ns, ps



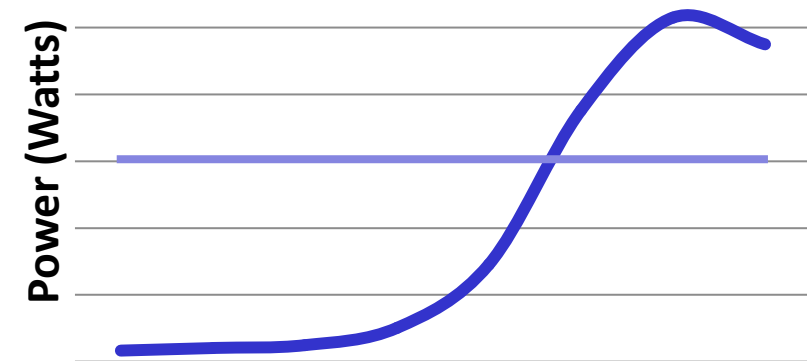
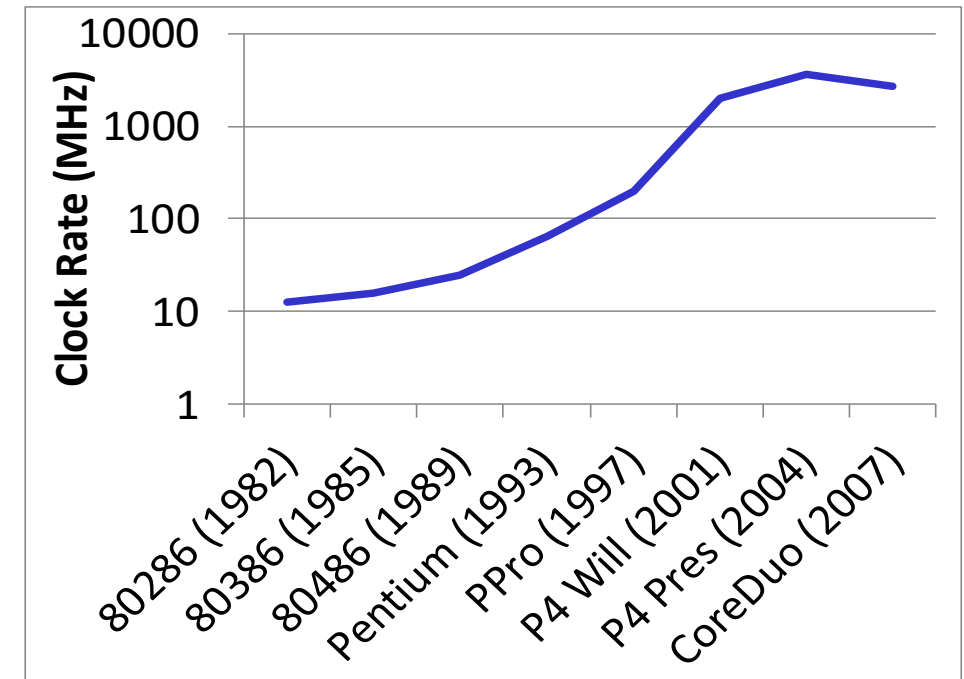
► Clock rate = $1 / \text{CC}$

- Given in MHz, GHz

► Clock rates hit a “power wall”

- Order of GHz not exceeded for several years for energy reasons

Machine Clock Rate





Product	AMD Barcelona	Intel Nehalem	AMD Zen 2	Intel Coffee Lake
Cores per chip	4	4	4-16	4-8
Clock rate	1,8-2.5 GHz	2,6-3,6 GHz	3.5-4.7 GHz	3.0-5.0 GHz
Power	79-137 W	130 W	65-105 W	65-127 W

- ▶ The power challenge has forced a change in the design of microprocessors
 - Higher clock rates lead to high temperatures, not possible with current semi-conductor technology
- ▶ Other approaches needed
 - Number of cores per chip is increased every generation
 - Better design approaches for instruction processing
 - Dynamic voltage/clock rate used



► Instruction set (IS)

- Same IS may be supported by different machines
- Performance may vary

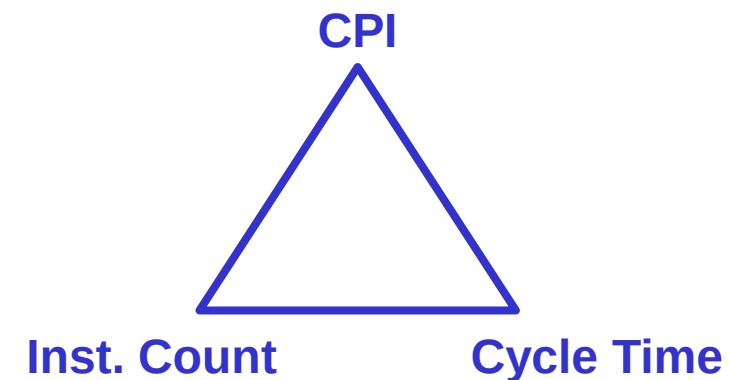
► Clock cycles per instruction (CPI)

- Depends on how IS is implemented on a machine
- Complex instructions have large CPI
 - CPI is instruction-specific
 - Define instruction classes with similar CPI

	Instruction class		
	A	B	C
CPI (machine 1)	1	2	3
CPI (machine 2)	2	3	5



- ▶ Given: programm (instructions)
 - IC : instruction count
 - IC_i : instruction count per CPI class i
- ▶ Instruction mix
 - Influences IC_i
 - Depends on program
- ▶ Effective CPI
 - n : number of CPI classes
 - $CPI_{eff} = \frac{\sum_{i=1}^n CPI_i \cdot IC_i}{IC}$
- ▶ CPU time
 - $t_{CPU} = \frac{\sum_{i=1}^n CPI_i \cdot IC_i}{clock\ rate}$
 - Three factors influenceing CPU time





Factors for Instruction Count, CPI_{eff} , Clock Rate

	Instruction count	CPI_{eff}	Clock rate
Algorithm	X	X	
Compiler	X	X	
ISA	X	X	
Core organization	X	X	X
Technology		X	X



- ▶ Execution speed (CPU time), does not reflect idle times
 - Time to wait for I/O
 - Time to wait for CPU access
 - Etc.

- ▶ Still we define the performance of machine X for program P as
 - $Performance(X, P) = \frac{1}{t_{CPU}(X, P)}$
 - Faster is better
 - Comparison of two machines X and Y for specific program P
 - $\frac{Performance(X, P)}{Performance(Y, P)} = \frac{t_{CPU}(Y, P)}{t_{CPU}(X, P)} = n$ means
 - Performance of X is n times larger than the one of Y
 - X is n times faster than Y



► Benchmark

- A set of programs that form a “workload” specifically chosen to measure performance

► SPEC (System Performance Evaluation Cooperative)

- Creates standard sets of benchmarks
- www.spec.org



- ▶ Response time (overall execution time)
 - Time between the start and the completion of a task
 - Important to individual users

- ▶ Throughput
 - Total amount of work done in a given time
 - Important to data center managers

- ▶ Energy efficiency
 - Power consumption especially important for power-limited applications, e.g., in the embedded market where battery life matters



Vielen Dank für eure Aufmerksamkeit!

Bis nächste Woche!