#1 : Background

Computer Architecture 2019/2020

Ricardo Rocha

Computer Science Department, Faculty of Sciences, University of Porto

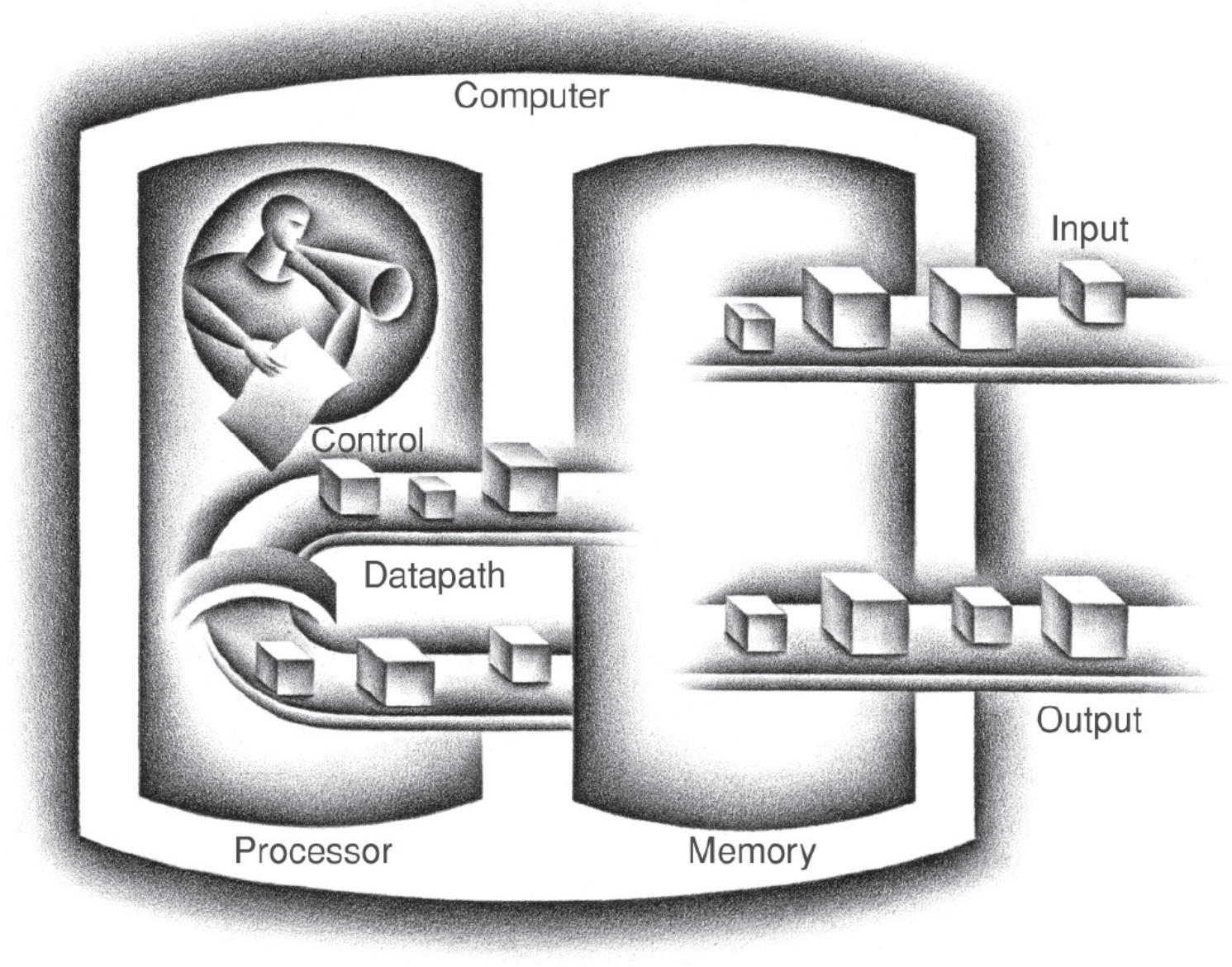
Slides based on the book

'Computer Organization and Design, The Hardware/Software Interface, 5th Edition

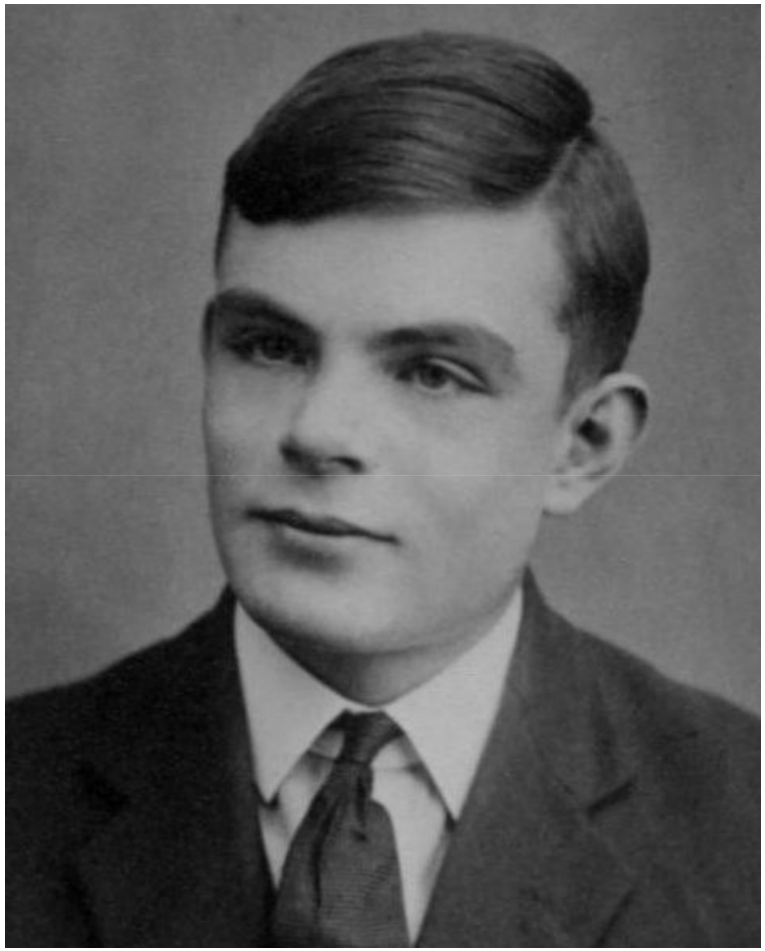
David Patterson and John Hennessy, Morgan Kaufmann'

Sections 1.1 – 1.5 and 1.8

The Five Classic Components in a Modern Computer



Advent of the Digital Computer



Alan Turing

John von Neumann

Advent of the Digital Computer

The principle of the **modern computer** was first described by computer scientist **Alan Turing**, who set out the idea in his seminal 1936 paper, '*On Computable Numbers*'.

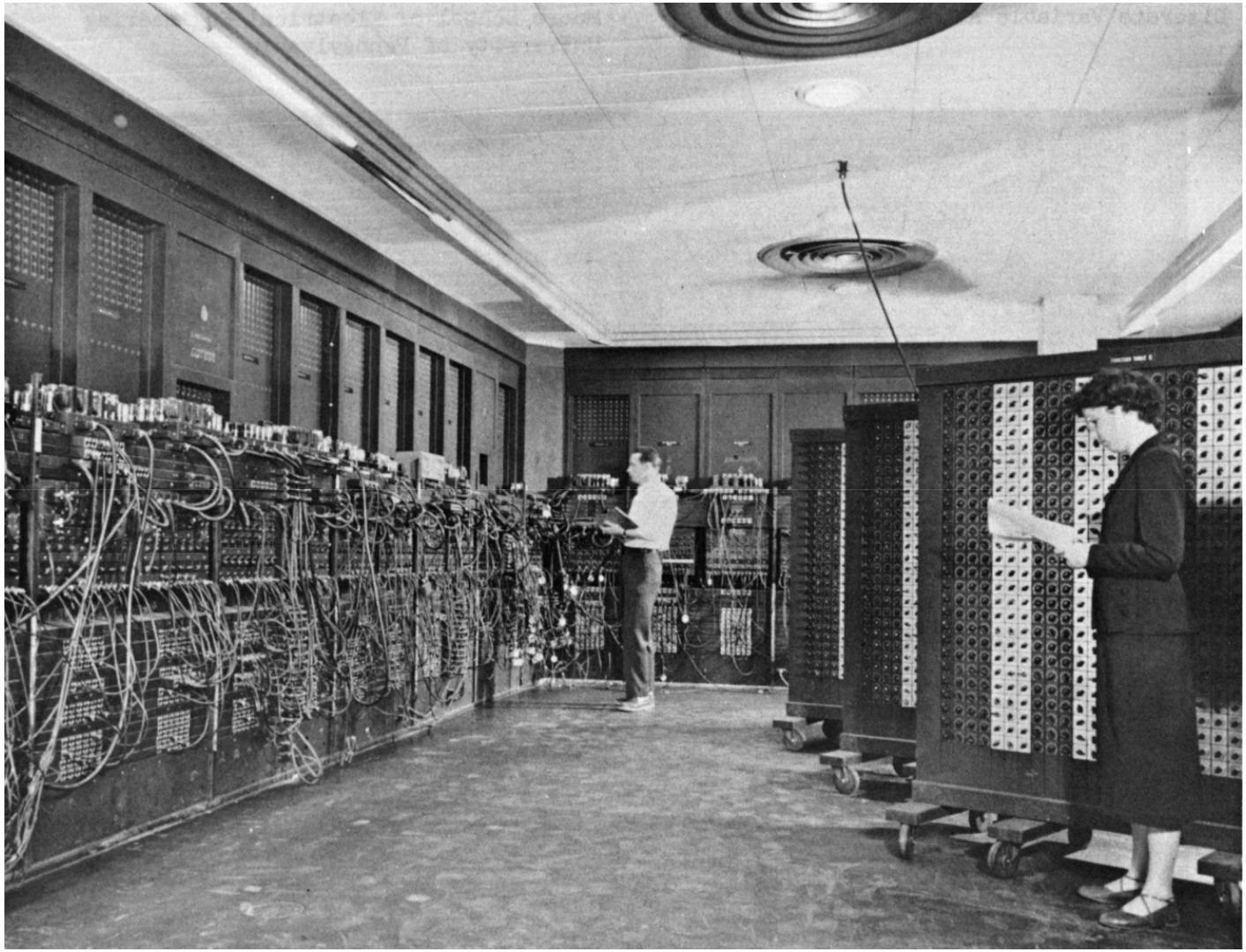
He proved that a **Turing machine** would be capable of performing any conceivable **mathematical computation** if it were **representable as an algorithm**.

Advent of the Digital Computer

Alan Turing also introduced the notion of a **universal Turing machine**, with the idea that such a machine could perform the tasks of any other machine, or in other words, it is provably capable of computing anything that is computable by executing a program stored on tape, allowing the **machine to be programmable**.

Except for the **limitations imposed by their finite memory stores**, modern computers are said to be **Turing-complete**, which is to say, they have algorithm execution capability equivalent to a universal Turing machine.

Advent of the Digital Computer



ENIAC (Electronic Numerical Integrator and Computer)
First general purpose (Turing-complete) computer in the US, 1946

Advent of the Digital Computer

ENIAC combined the high speed of electronics (vacuum tubes, resistors and capacitors) with the **ability to be programmed**. It could add or subtract 5000 times a second, 1000 times faster than any other machine. It also had modules to multiply, divide, and square root. It could compute any problem that would fit into its memory.

A program on the ENIAC was defined by the states of its patch cables and switches. Once a program was written, it had to be **mechanically set into the machine with manual resetting of plugs and switches**. **Reprogramming** was a laborious process, starting with engineers working out flowcharts, designing the new setup, and then the often-exacting process of physically re-wiring patch panels.

Stored-Program Computer

A **stored-program computer** is a computer that **stores program instructions in electronic memory**. This contrasts with machines where the program instructions are stored on plugboards or similar mechanisms.

The **theoretical basis for the stored-program computer had been proposed by Alan Turing** in his 1936 paper. In 1945 Turing joined the National Physical Laboratory and began his work on developing an electronic stored-program digital computer. His 1945 report '*Proposed Electronic Calculator*' was the first specification for such a device.

Stored-Program Computer

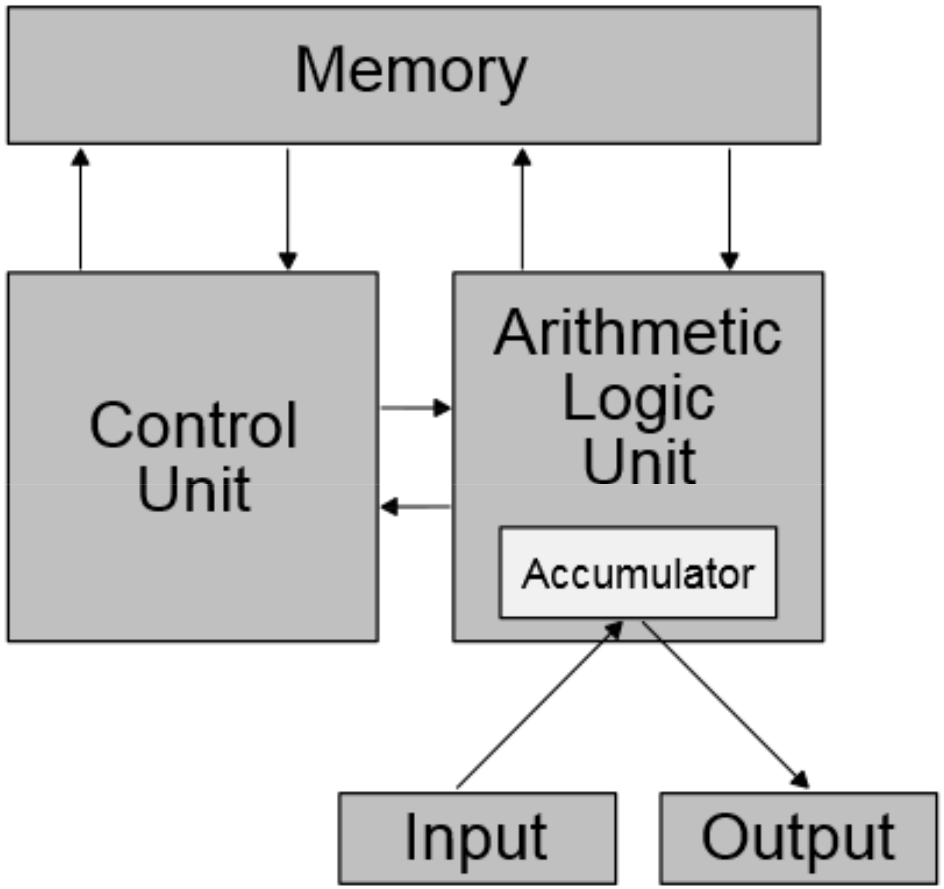
Alongside, John von Neumann circulated his '*First Draft of a Report on the EDVAC*' in 1945. Although substantially similar to Turing's design and containing comparatively little engineering detail, its proposal became known as the **von Neumann architecture**.

Turing presented a more detailed paper in 1946, giving the first reasonably complete design of a stored-program computer. However, the **better-known EDVAC design of von Neumann**, who knew of Turing's theoretical work, **received more publicity**, despite its incomplete nature and questionable lack of attribution of the sources of some of the ideas.

Stored-Program Computer

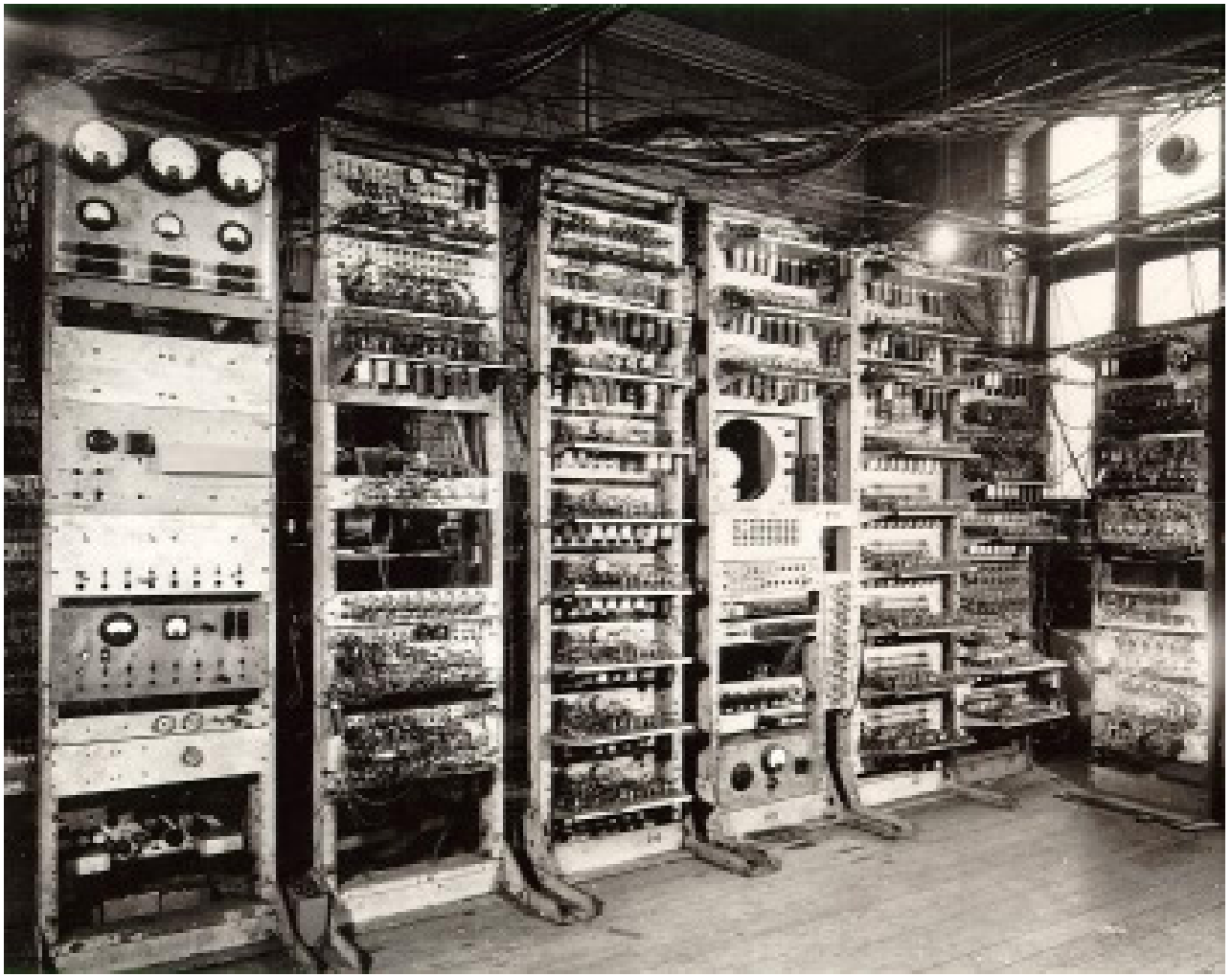
John von Neumann's document describes a design architecture for an electronic digital computer with these components:

- **Processing unit** containing an **arithmetic logic unit** and **processor registers**
- **Control unit** containing a **program counter** and an **instruction register**
- **Memory** that stores **data** and **instructions**
- **Input** and **output** mechanisms



von Neumann architecture, 1945

Stored-Program Computer



Manchester Mark I

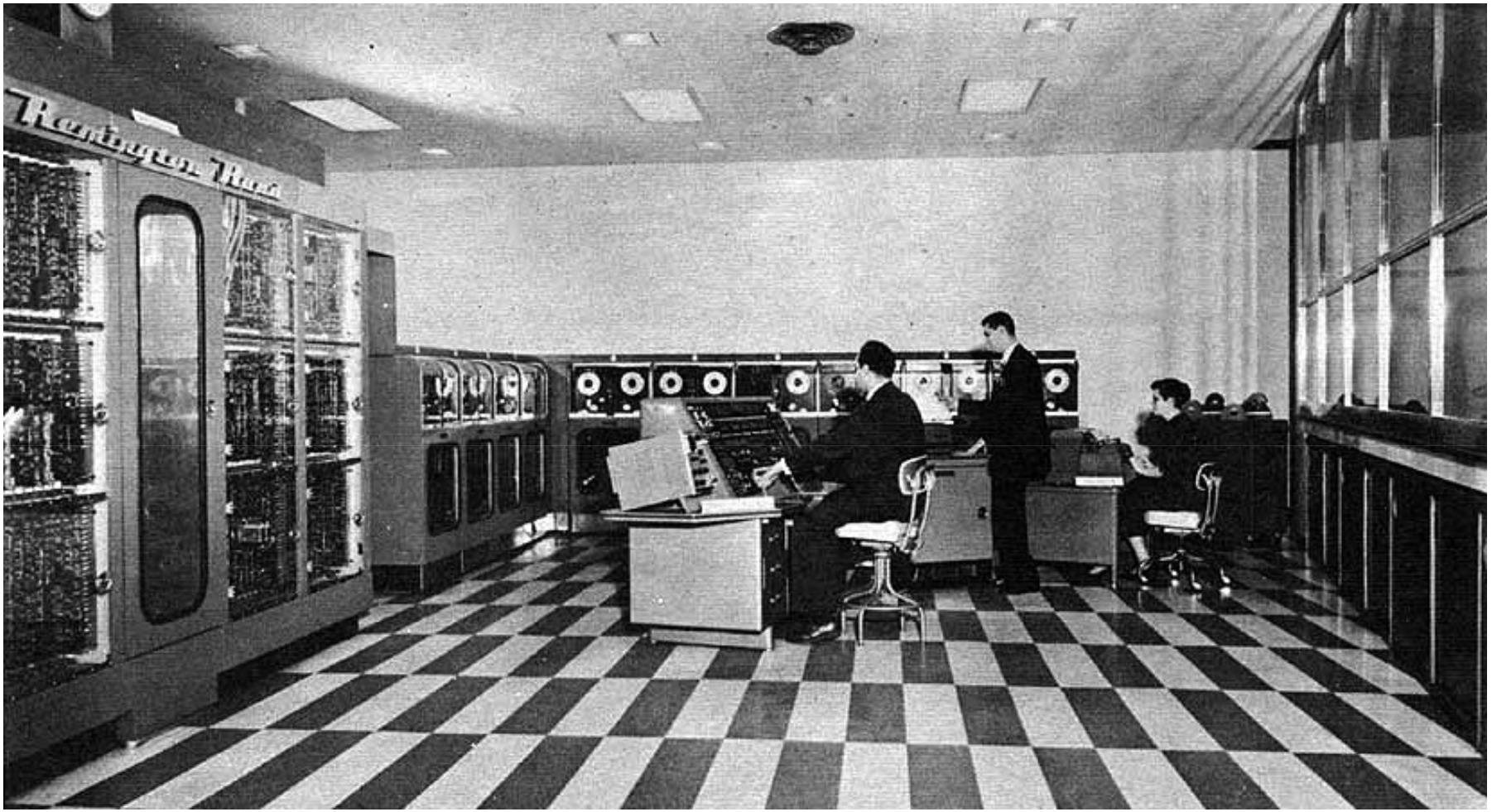
One of the first stored-program computers, 1949

Stored-Program Computer

Ferranti Mark I

First commercial computer, delivered to the University of Manchester, 1951

Stored-Program Computer



UNIVAC I (UNIVersal Automatic Computer I)

First mass produced computer, 1951

46 machines sold at more than US$1 million each ($9.65 million as of 2019)

Transistor Computers

A **transistor** is simply an **on/off switch controlled by electricity**. Compared to vacuum tubes, transistors are **smaller** and **require less power** than vacuum tubes, so give off **less heat**.

Transistorized computers could contain **tens of thousands of binary logic circuits in a relatively compact space**. Transistors greatly reduced computers' size, initial cost, and operating costs. From 1955 onward, transistors replaced vacuum tubes in computer designs giving rise to the **second generation of computers**.

Transistor Computers

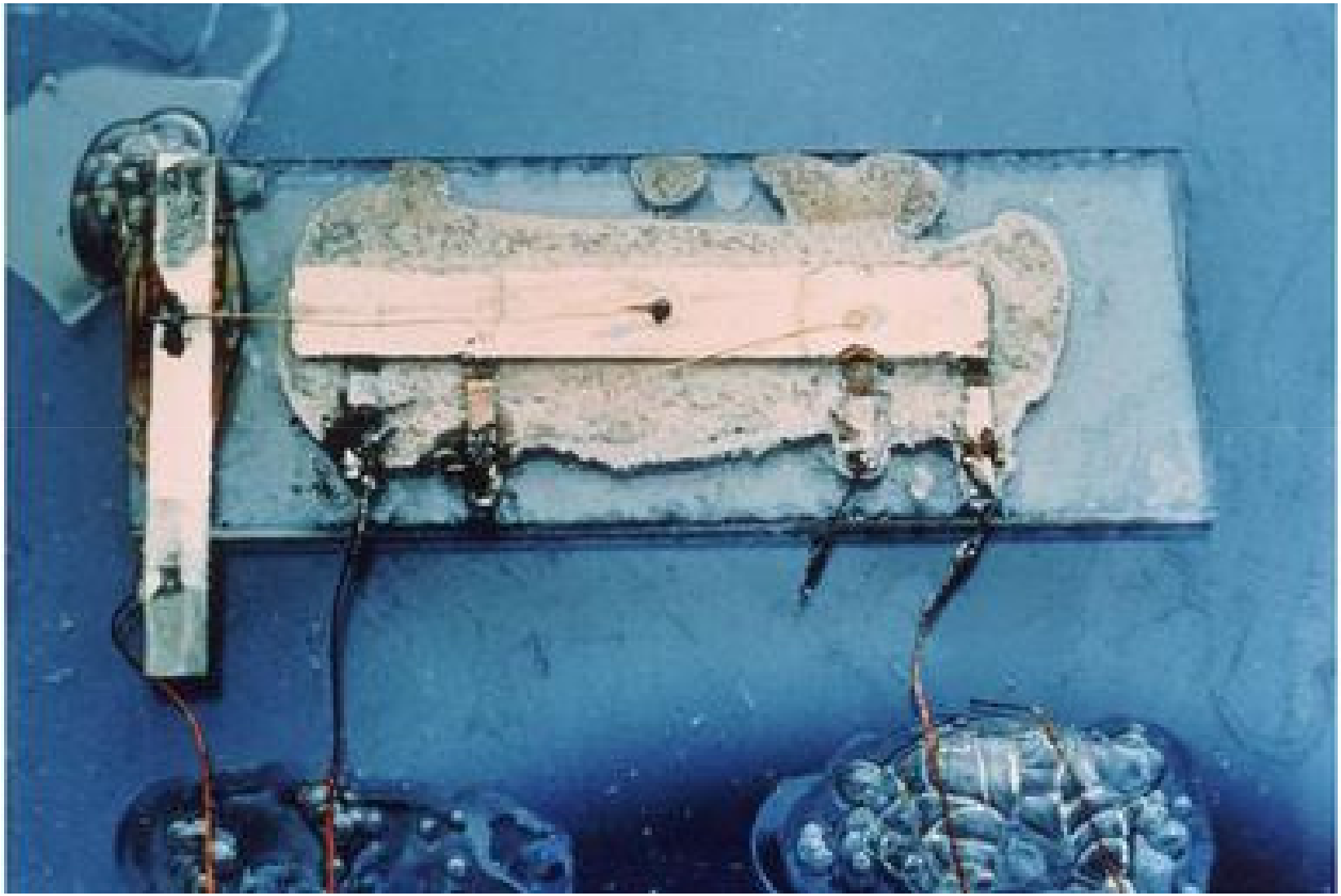
Replica of the first working transistor, 1947

Integrated Circuit Computers

An **integrated circuit (IC)** combines a set of transistors into **one small flat piece, or chip**. The circuit elements are inseparably associated and electrically interconnected so that it is considered to be **indivisible for the purposes of construction and commerce**.

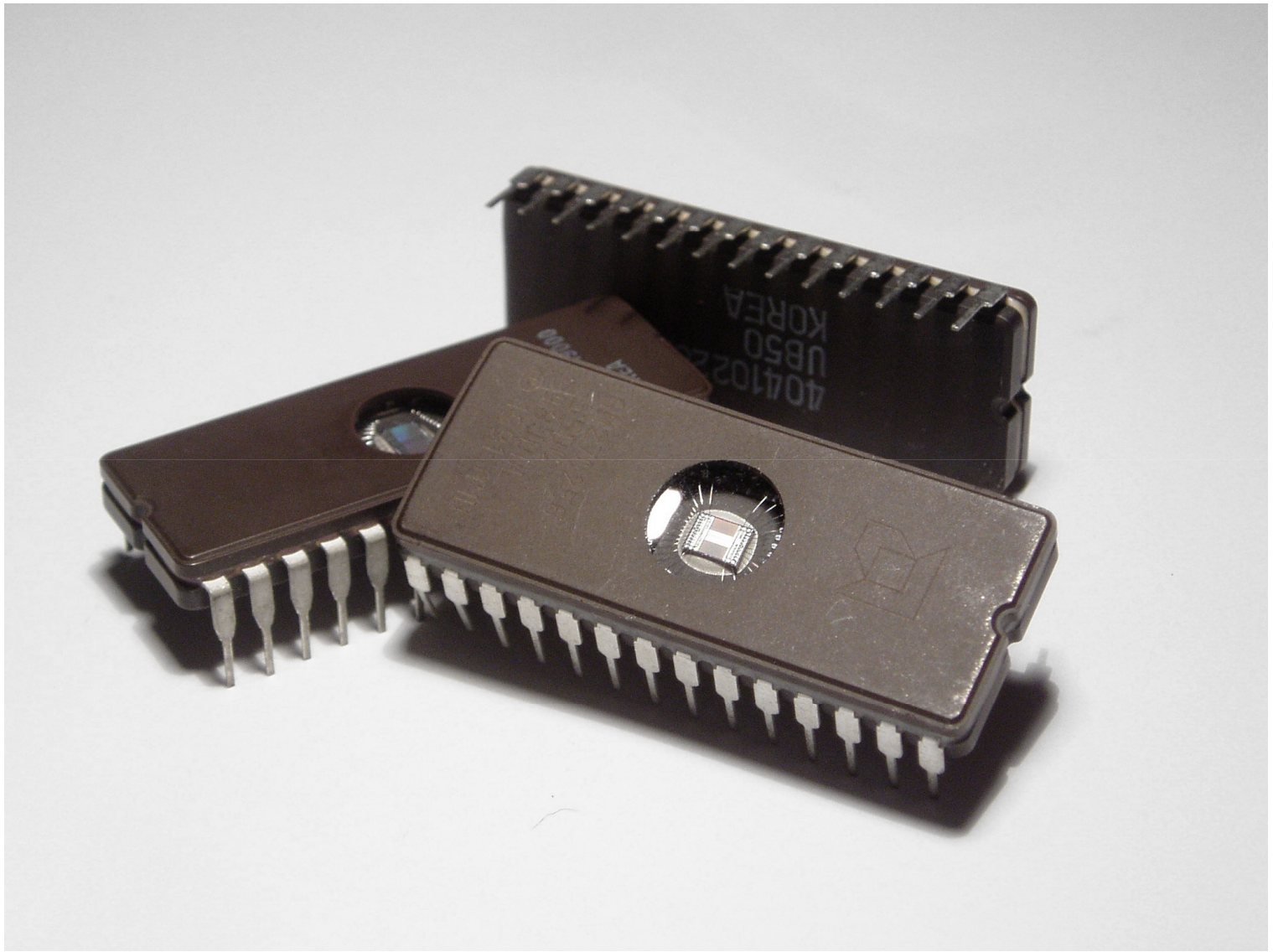
The **third-generation of computers** used integrated circuits as the basis of their logic. Third generation computers first appeared in the early 1960s in computers developed for government purposes, and then in commercial computers beginning in the middle 1960s.

Integrated Circuit Computers



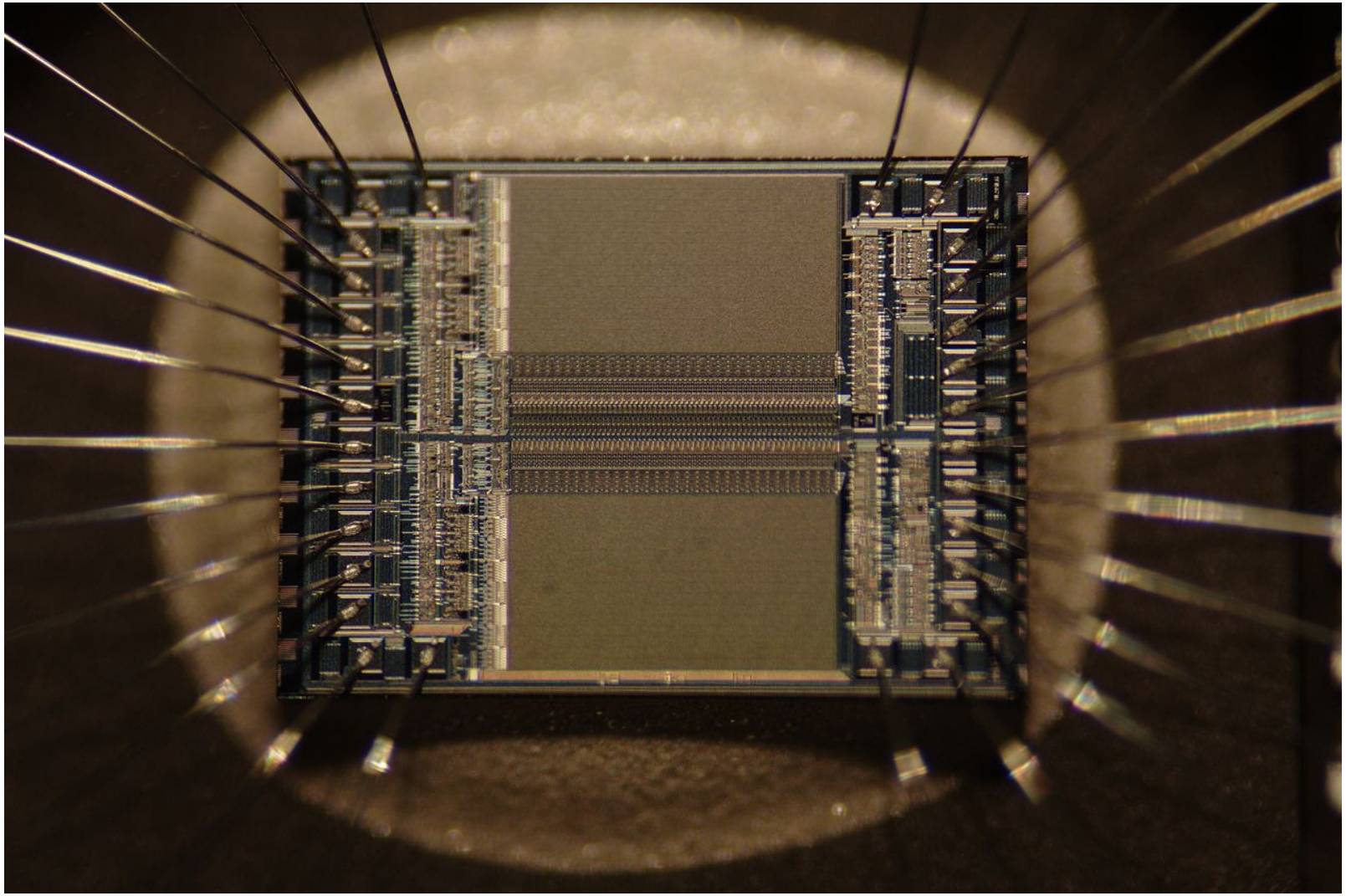
Jack Kilby's original integrated circuit, 1959

Integrated Circuit Computers



EPROM (Erasable Programmable Read-Only Memory) Integrated Circuit

Integrated Circuit Computers



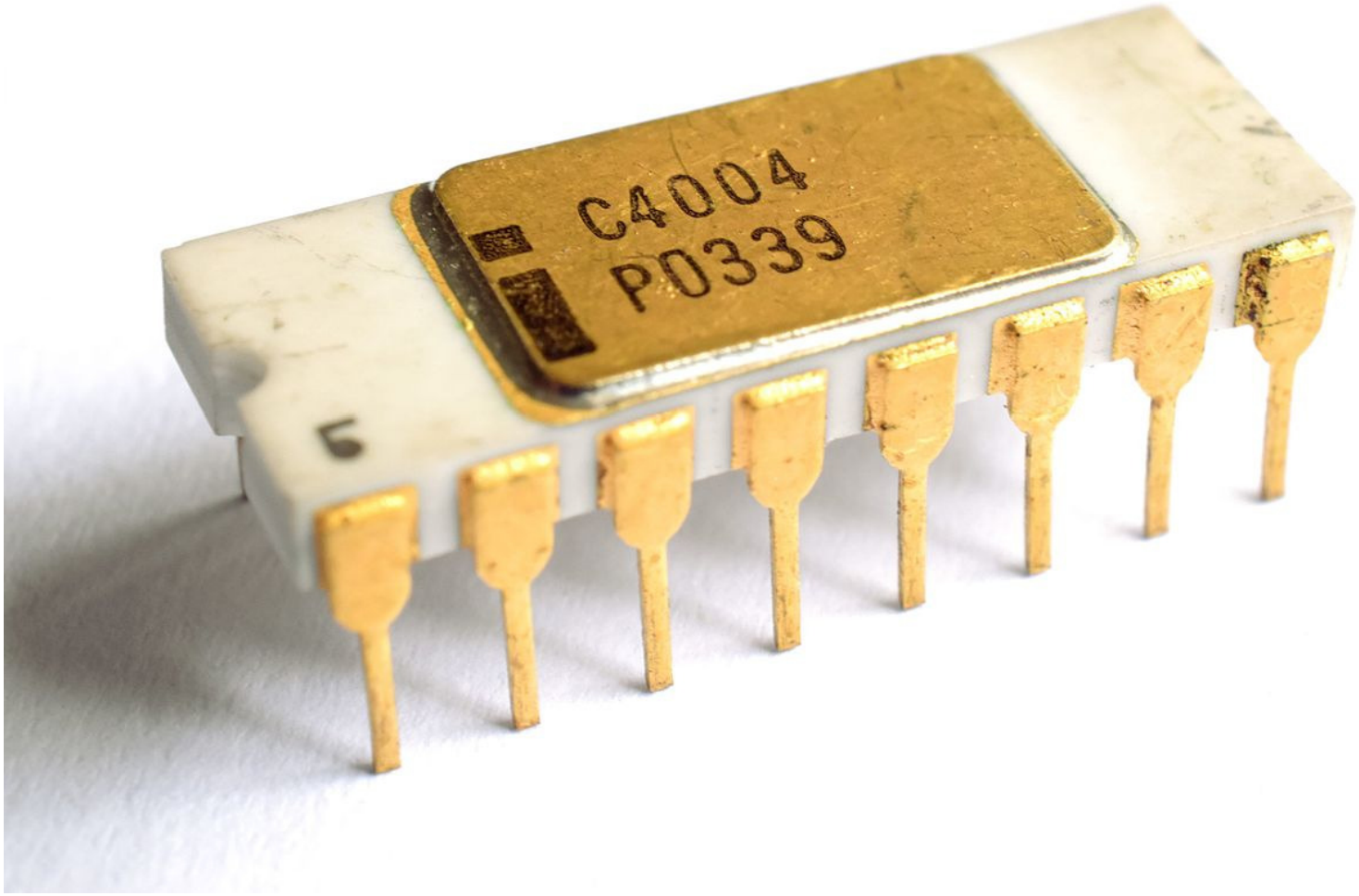
EPROM Integrated Circuit
High-magnification view

Microprocessor Computers

A microprocessor **incorporates the functions of a central processing unit on a single IC** or at most a few ICs. The **earliest microprocessors contained only the processor**. Later, **additional features were added to the processor architecture** – more on-chip registers speed up programs, complex instructions could be used to make more compact programs; integration of the floating point unit as part of the same microprocessor chip speed up floating point calculations.

The **fourth-generation of computers used microprocessors** as the basis of their logic. It is largely undisputed that the **first single-chip microprocessor was the Intel 4004**.

Microprocessor Computers



Intel 4004 – 4-bit CPU

First commercially available microprocessor by Intel, 1971

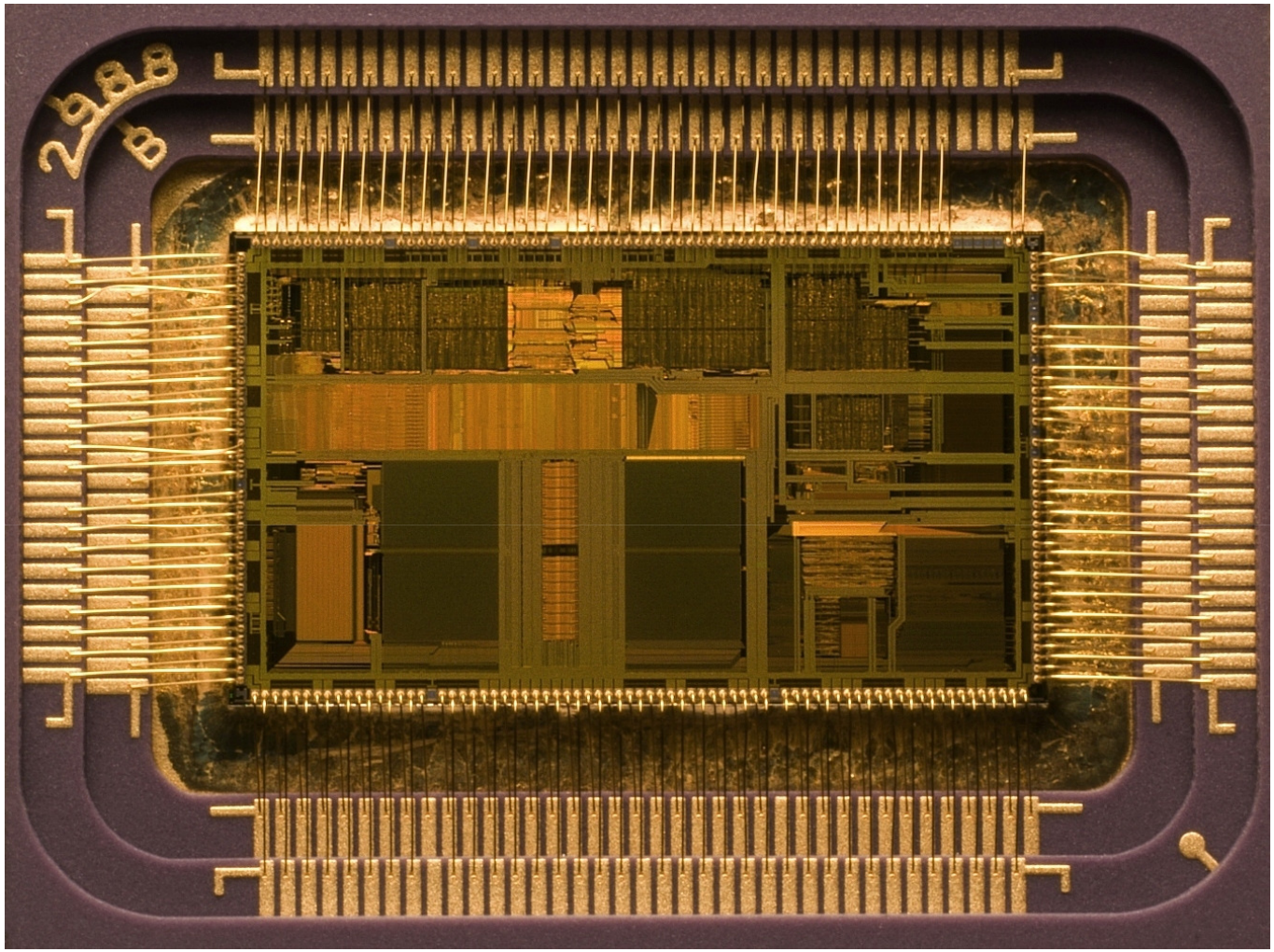
Microprocessor Computers

Intel 80486 (i486) Microprocessor

First x86 chip to use more than a million transitors,1989

Large on-chip cache and an integrated floating-point unit

Microprocessor Computers



Intel 80486DX2 Microprocessor

First chip to use clock doubling, 1992

The processor runs two internal logic clock cycles per external bus cycle

Technology Trends

To describe the tremendous increase in the number of transistors from hundreds to millions/billions, the adjectives **very large-scale** and **ultra large-scale** were added, creating the abbreviations **VLSI and ULSI**.

Year	Technology used in computers	Relative performance/unit cost
1951	Vacuum tube	1
1965	Transistor	35
1975	Integrated circuit	900
1995	Very large-scale integrated circuit	2,400,000
2013	Ultra large-scale integrated circuit	250,000,000,000

Technology Trends

Advancing technology makes **more complex and powerful chips feasible to manufacture**. The complexity of an integrated circuit is bounded by:

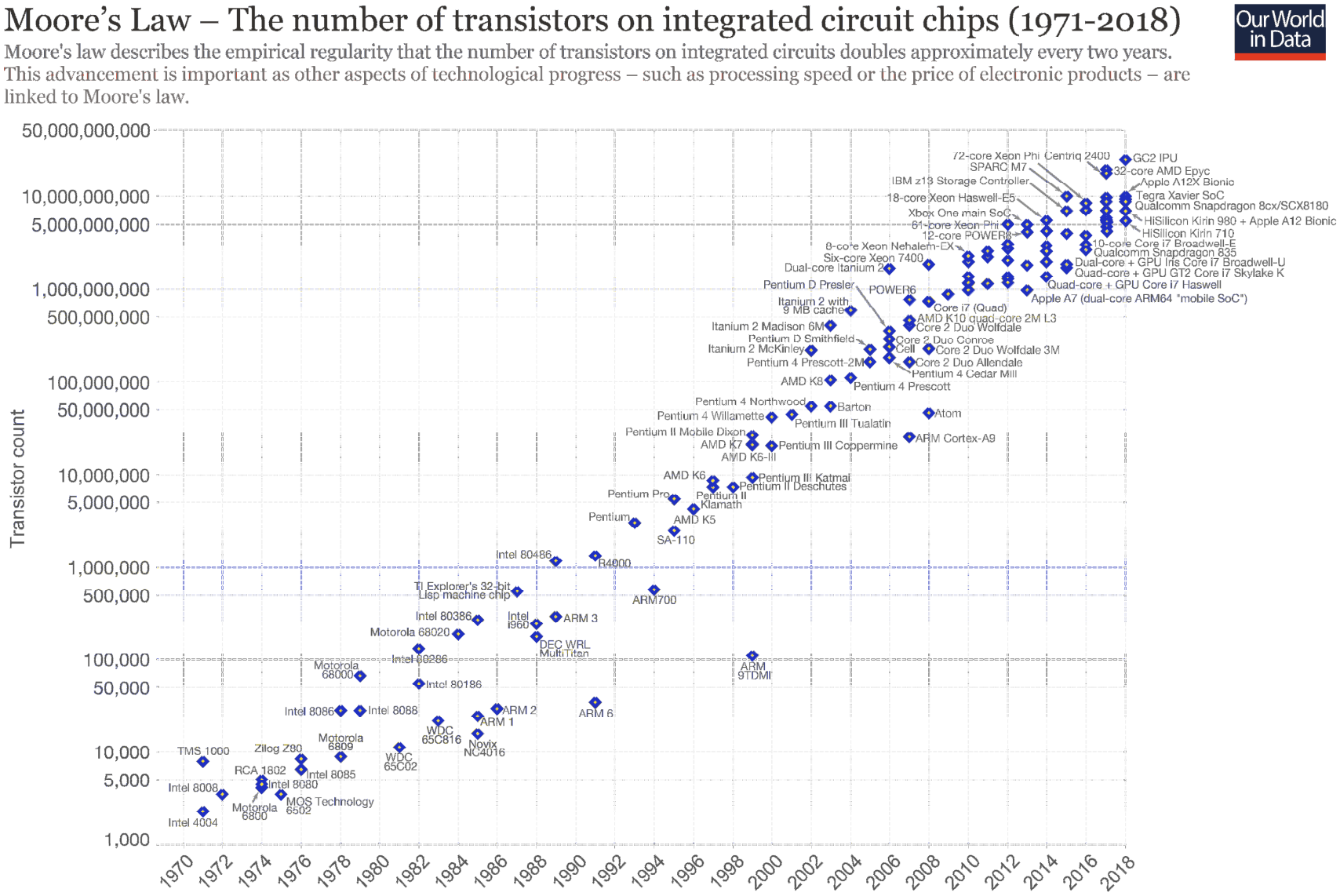
- Physical limitations on the number of transistors that can be put onto one chip
- The number of package terminations that can connect the processor to other parts of the system
- The number of interconnections it is possible to make on the chip
- The heat that the chip can dissipate

Moore's Law

Moore's law is the observation that the **number of transistors in a dense integrated circuit doubles approximately every 18 months**. Moore's law is an **observation and projection of a historical trend** and **not a physical or natural law**.

The observation is named after Gordon Moore, CEO of Intel, whose 1965 paper described a doubling every year in the number of components per integrated circuit, and projected this rate of growth would continue for at least another decade. In 1975, looking forward to the next decade, he revised the forecast to doubling every two years. In 2015 Gordon Moore foresaw that the **rate of progress would reach saturation**: "I see Moore's law dying here in the next decade or so".

Moore’s Law


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.
Our World in Data
Transistor count
50,000,000,000
10,000,000,000
5,000,000,000
1,000,000,000
500,000,000
100,000,000
50,000,000
10,000,000
5,000,000
1,000,000
500,000
100,000
50,000
10,000
5,000
1,000
1970
1972
1974
1976
1978
1980
1982
1984
1986
1988
1990
1992
1994
1996
1998
2000
2002
2004
2006
2008
2010
2012
2014
2016
2018
72-core Xeon Phi Centriq 2400
SPARC M7
IBM z13 Storage Controller
18-core Xeon Haswell-E5
Xbox One main SoC
61-core Xeon Phi
12-core POWER8
8-core Xeon Nehalem-EX
Six-core Xeon 7400
Dual-core Itanium 2
Pentium D Presler
POWER6
Itanium 2 with 9 MB cache
Itanium 2 Madison 6M
Pentium D Smithfield
Itanium 2 McKinley
Pentium 4 Prescott-2M
AMD K8
Pentium 4 Northwood
Pentium 4 Willamette
Pentium II Mobile Dixon
AMD K7
AMD K6-III
AMD K6
Pentium Pro
Pentium
Intel 80486
TI Explorer's 32-bit Lisp machine chip
Intel 80386
Motorola 68020
Intel 80286
Motorola 68000
Intel 80186
Intel 8086
Intel 8088
TMS 1000
Zilog Z80
RCA 1802
Intel 8085
Intel 8080
Intel 8008
Motorola 6800
MOS Technology 6502
Intel 4004
WDC 65C816
WDC 65C02
Motorola 6809
Novix NC4016
ARM 1
ARM 2
ARM 3
Intel i960
DEC WRL MultiTitan
R4000
ARM700
ARM 6
SA-110
AMD K5
Pentium II Klamath
Pentium II Deschutes
Pentium III Katmai
ARM 9TDMI
Core 2 Duo Wolfdale 3M
Core 2 Duo Allendale
Pentium 4 Cedar Mill
Pentium 4 Prescott
Core 2 Duo Conroe
Cell
AMD K10 quad-core 2M L3
Core 2 Duo Wolfdale
Core i7 (Quad)
Apple A7 (dual-core ARM64 "mobile SoC")
Quad-core + GPU Core i7 Haswell
Quad-core + GPU GT2 Core i7 Skylake K
Dual-core + GPU Iris Core i7 Broadwell-U
10-core Core i7 Broadwell-E
Qualcomm Snapdragon 835
HiSilicon Kirin 710
HiSilicon Kirin 980 + Apple A12 Bionic
Qualcomm Snapdragon 8cx/SCX8180
Tegra Xavier SoC
Apple A12X Bionic
32-core AMD Epyc
GC2 IPU
Barton
Atom
Pentium III Tualatin
Pentium III Coppermine
ARM Cortex-A9
Data source: Wikipedia (https://en.wikipedia.org/wiki/Transistor_count)
The data visualization is available at OurWorldinData.org. There you find more visualizations and research on this topic.
Licensed under CC-BY-SA by the author Max Roser.

Classes of Computers

Personal computers (PCs)

- General purpose, variety of software
- Subject to cost/performance tradeoff

Server computers

- High capacity, performance, reliability, network based
- Range from small servers to building sized

Supercomputers

- High-end scientific and engineering calculations
- Highest capability but represent a small fraction of the overall market

Embedded computers

- Hidden as components of systems
- Stringent power/performance/cost constraints

The New Era

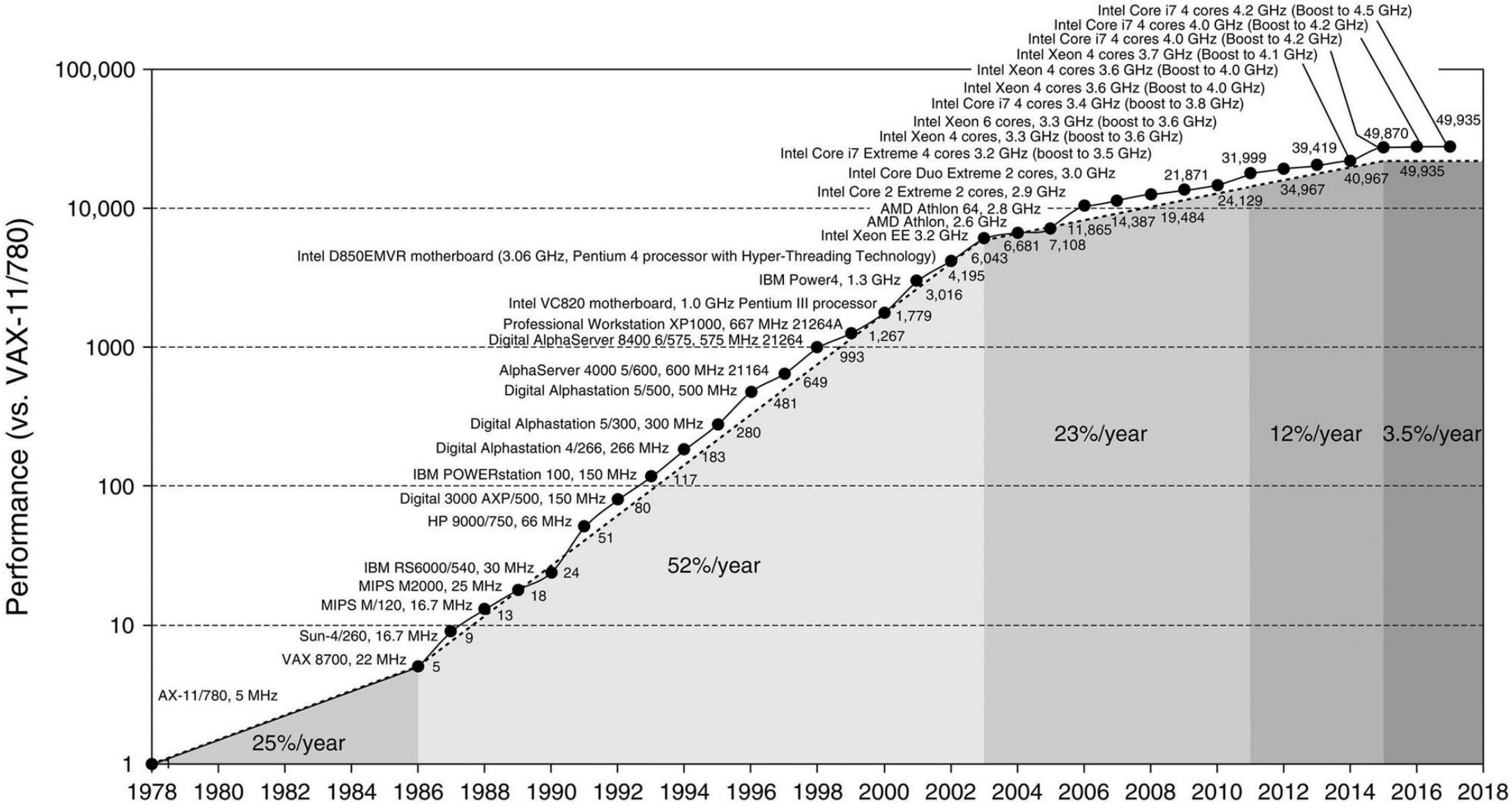
From personal computers to **personal mobile devices (PMDs)**

- Battery operated
- Wireless connectivity to the Internet
- Cost hundreds of dollars
- Users can download apps
- Smart phones, smart watches, tablets and electronic glasses

From servers to **cloud computing**

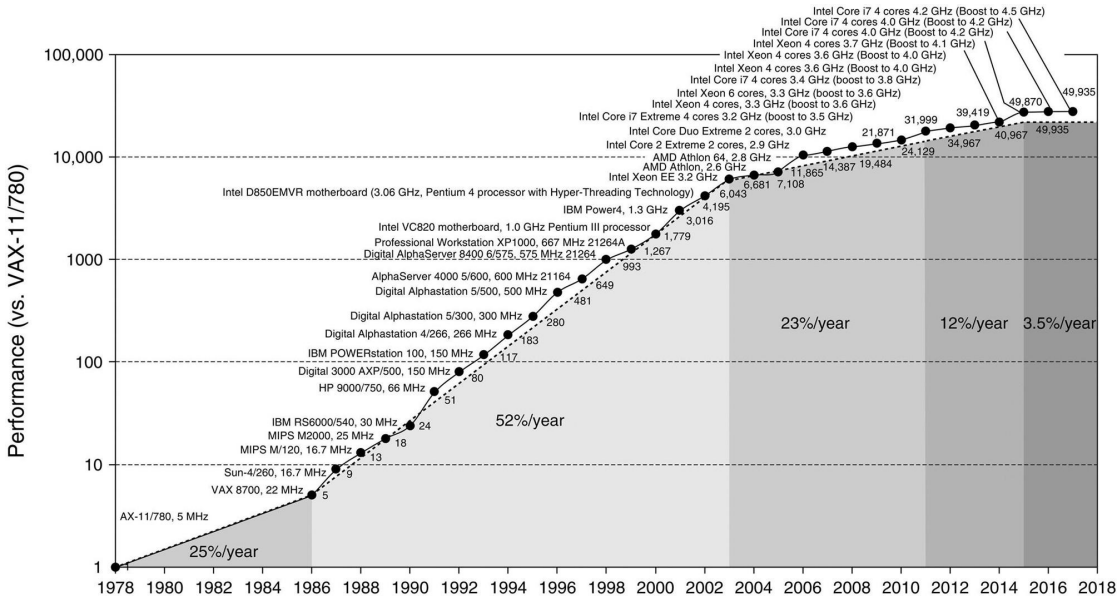
- Giant warehouse scale computers (WSCs)
- Amazon and Google have WSCs with more than 100,000 servers
- Software as a Service (SaaS) deployed via the cloud
- Portion of software runs on the PMD and another portion runs in the Cloud

Growth in Processor Performance



Growth in Processor Performance

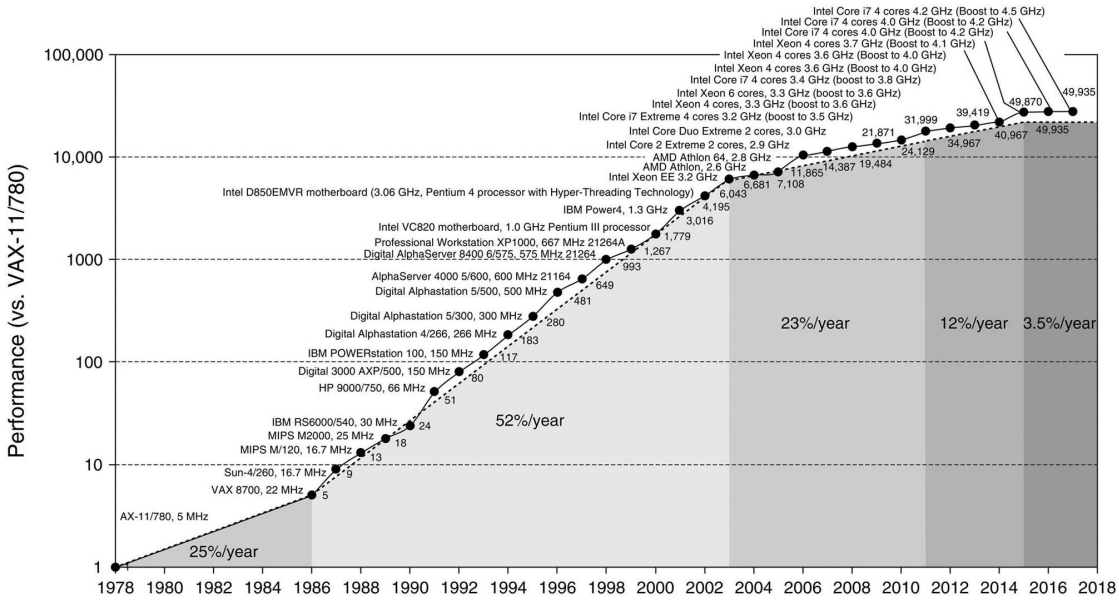
Prior to the mid-1980s, processor performance growth was largely **technology-driven** and averaged about 25% per year (i.e., doubling performance every 3.5 years).



Starting in 1986, the increase in growth to about 52% (doubling every 2 years) is attributable to more advanced architectural and organizational ideas typified in **RISC architectures** (e.g., use of caches and the exploitation of instruction-level parallelism).

Growth in Processor Performance

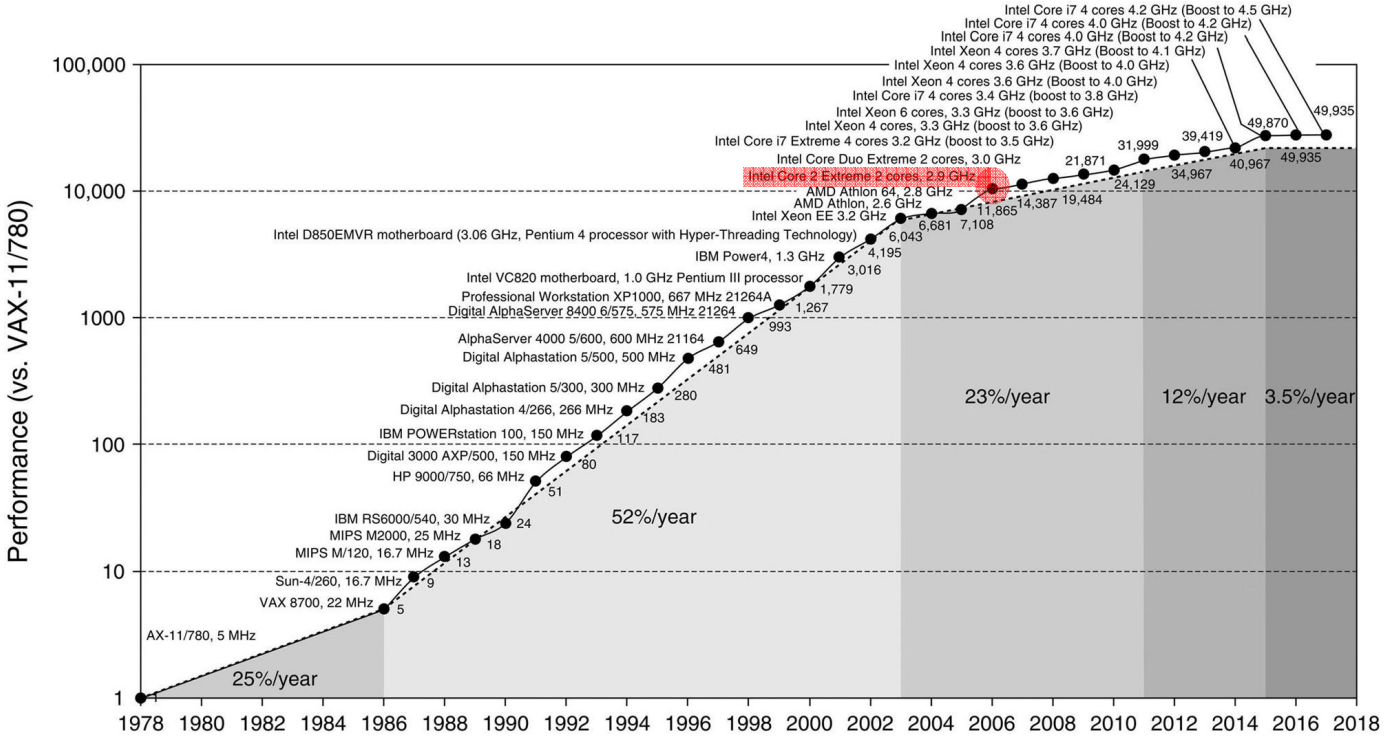
Since 2003, the **limits of power, available instruction-level parallelism, and long memory latency** slowed uniprocessor performance to about 23% per year until 2011.



From 2011 to 2015, the improvement was less than 12% (doubling every 8 years) in part due to the **limits of parallelism of Amdahl's Law**. Since 2015, improvement has been just 3.5% (doubling every 20 years!).

From Uniprocessors to Multiprocessors

Rather than continuing to decrease the response time of a single program running on the single processor, as of 2006 all desktop and server companies are shipping **microprocessors with multiple processors per chip**, generically called **multicore microprocessors**, where the benefit is often more on throughput than on response time.



From Uniprocessors to Multiprocessors

In the past, programmers could rely on innovations in hardware, architecture and compilers to double performance of their programs every 18 months **without having to change a line of code**.

Nowadays, for programmers to get significant improvement in response time, they need to rewrite their programs to take **explicit advantage of multiple processors**.

In the future, programmers will have to **continue to improve performance of their code as the number of cores increases**.

From Uniprocessors to Multiprocessors

Why is so hard for programmers to write explicitly parallel programs?

One reason is that **parallel programming is by definition performance programming** – not only does the program need to be correct, solve an important problem, and provide a useful interface, it **must also be fast**.

Another reason is that fast for parallel hardware means that the programmer must divide an application so that **each processor has roughly the same amount of work to do at the same time (load balancing) without increasing the potential overhead of coordination (which includes scheduling, synchronization and communication between the parties)**.

What We Will Learn

How programs are translated into the machine language and how the hardware executes them:

- The hardware/software interface (instruction set architecture)
- Hierarchical layers of abstraction in both hardware and software

What determines program performance and how it can be improved:

- How hardware designers improve performance
- Execution time as the best performance measure
- Power is a limiting factor, use parallelism to improve performance

Eight Great Ideas in Computer Architecture

Design for **Moore's Law**

Use **abstraction** to simplify design

Make the **common case fast**

Performance via **parallelism**

Performance via **pipelining**

Performance via **prediction**

Hierarchy of memories

Dependability via redundancy

External References

Wikipedia: History of Computing Hardware

- https://en.wikipedia.org/wiki/History_of_computing_hardware

Computer History Museum: Timeline of Computer History

- https://www.computerhistory.org/timeline/computers

Singularity Prosperity: The History of Computing

- https://www.youtube.com/watch?v=-M6lANfzFsM