



Foto: Thomas Josek

# Basisinformationstechnologie I

Wintersemester 2022/23. Rechnerentwicklung, Schaltalgebra / Digitaltechnik. *Basierend auf Jan Wieners' Folien*

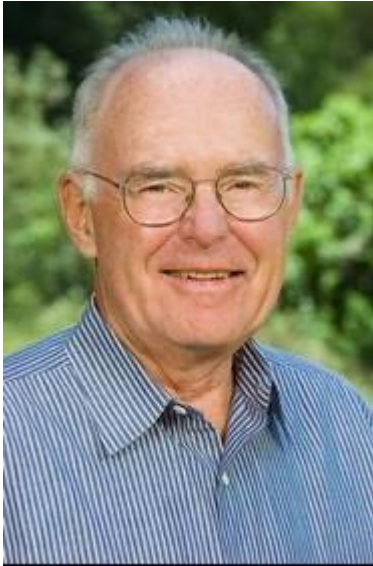
# Themenüberblick „Rechnertechnologie, Schaltalgebra“

## Überblick: Rechner-/Computerentwicklung

- Moore, Leibniz, Babbage, Turing, Weizenbaum
  - von Neumann
  - Die von Neumann Rechnerarchitektur
  - Konzept: Universalrechner
  - Caching
- 
- Von Neumann Architektur, Universalrechner
  - (Logik)Gatter
    - Transistoren
    - Integrierte Schaltkreise
    - Integrationsgrad
    - Gattertypen
  - Digitaltechnik, boolesche- / Schaltalgebra

Rechner-/Computerentwicklung

# Rechner-/Computerentwicklung



„Wenn sich die Luftfahrtindustrie genauso schnell wie die Computertechnologie entwickelt hätte, würde ein Flugzeug 500 Dollar kosten und könnte die Erde in 20 Minuten mit 5 Gallonen Treibstoff umrunden. Allerdings hätte es die Größe eines Schuhkartons.“

(Gordon Moore, zitiert nach: Tanenbaum, Andrew S.: Computerarchitektur, S. 43.)

- Kostenreduzierung
- Leistungsfähigkeit (Performance)
- Ressourcenverbrauch
- Strukturreduzierung



Integrationsgrad: absolute Anzahl von Transistoren in einem Integrierten Schaltkreis

Größenordnungen:

- SSI – Small Scale Integration: 1 bis 10 Gatter
- MSI – Medium Scale Integration: 10 bis 100 Gatter
- **LSI** – Large Scale Integration: 100 bis 100.000 G.
- **VLSI** – Very Large Scale Integration: > 100.000 G.

(vgl.: Tanenbaum: Computerarchitektur. Strukturen – Konzepte – Grundlagen. 2006, 5.Auflage. S. 167.)

„Die Anzahl der Transistoren, die auf einem Chip (IC, Integrated Circuit) untergebracht werden können, verdoppelt sich alle 2 Jahre.“

# Das Mooresche Gesetz (Moore's Law)

„Die Anzahl der Transistoren, die auf einem Chip (IC, Integrated Circuit) untergebracht werden können, verdoppelt sich alle 2 Jahre.“

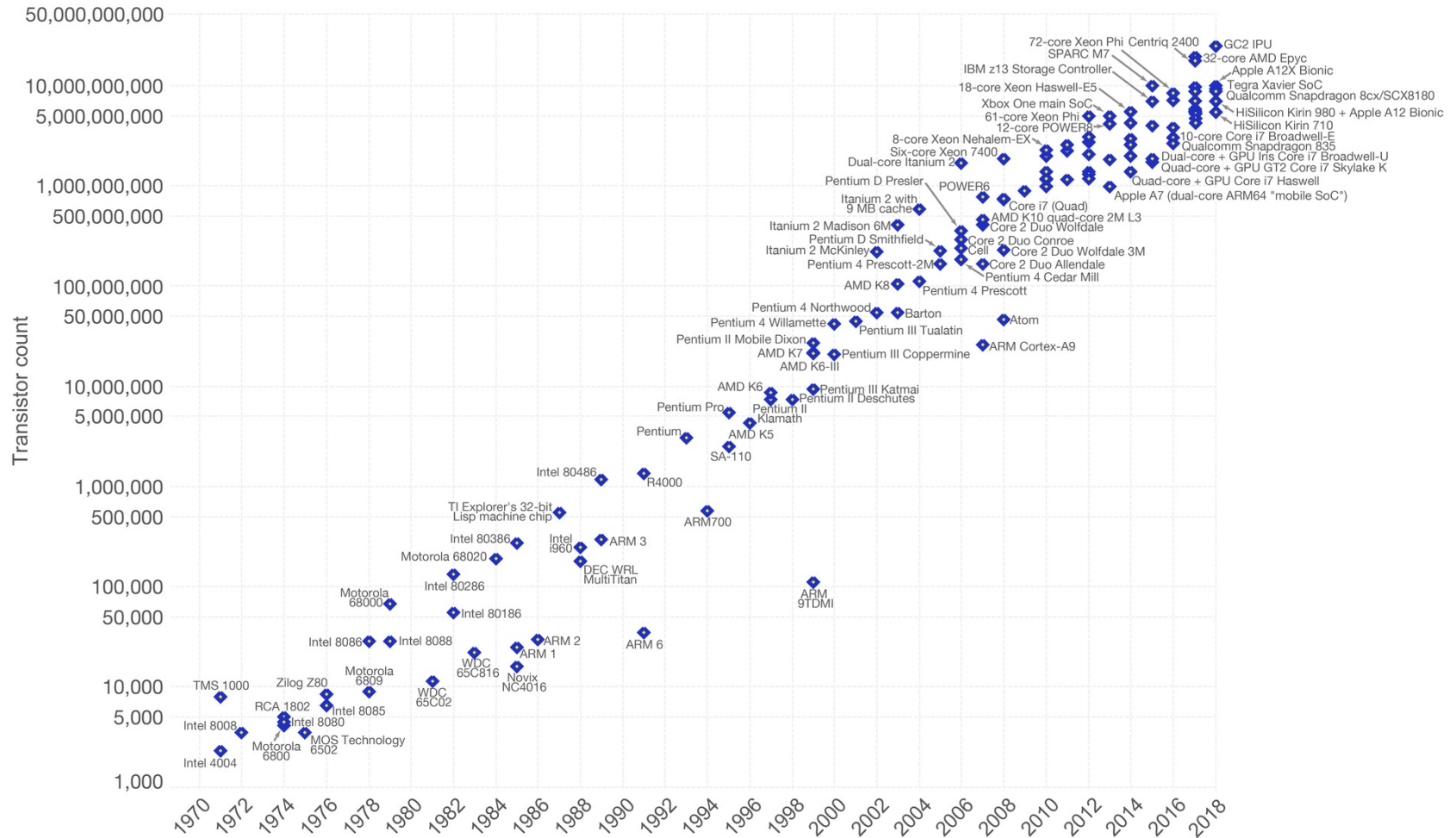
Entwicklungsstufen des Integrationsgrads von integrierten Schaltkreisen

Abk.	Bezeichnung	Komplexität (Gatteräquivalente)		
		typische Interpretation	Tanenbaum <sup>[1]</sup>	Texas Instruments <sup>[2]</sup>
SSI	small scale integration	10	1–10	unter 12
MSI	medium scale integration	100	10–100	12–99
LSI	large scale integration	1.000	100–100.000	100–999
VLSI	very large scale integration	10.000–100.000	ab 100.000	ab 1.000
ULSI	ultra large scale integration	100.000–1.000.000	—	—
SLSI	super large scale integration	1.000.000–10.000.000	—	—
ELSI	extra large scale integration	10.000.000–100.000.000	—	—
GLSI	giant large scale integration	> 100.000.000	—	—

# Das Mooresche Gesetz (Moore's Law)

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

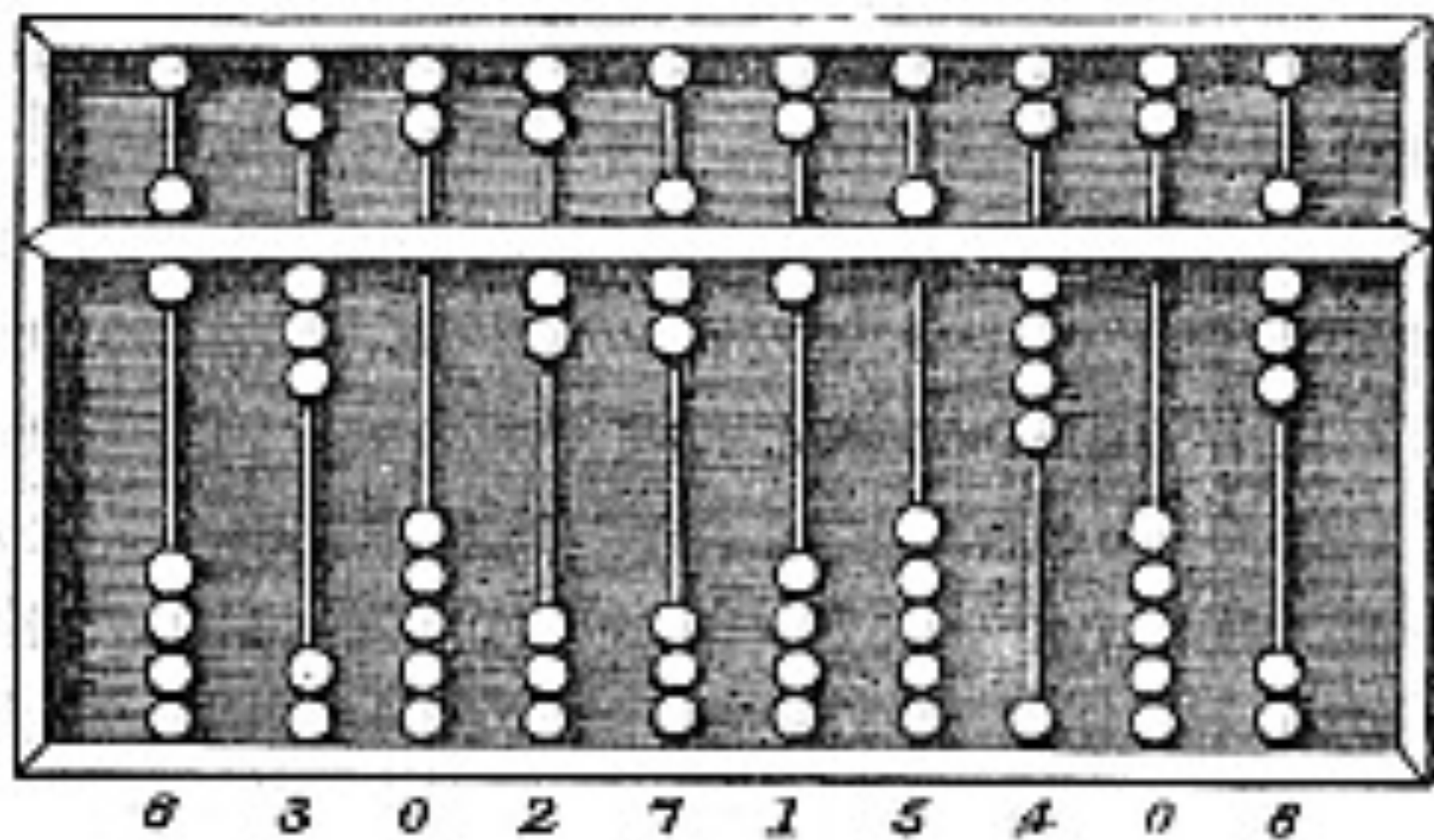
Moore's law describes the empirical regularity that the number of transistors on integrated circuits doubles approximately every two years. This advancement is important as other aspects of technological progress – such as processing speed or the price of electronic products – are linked to Moore's law.



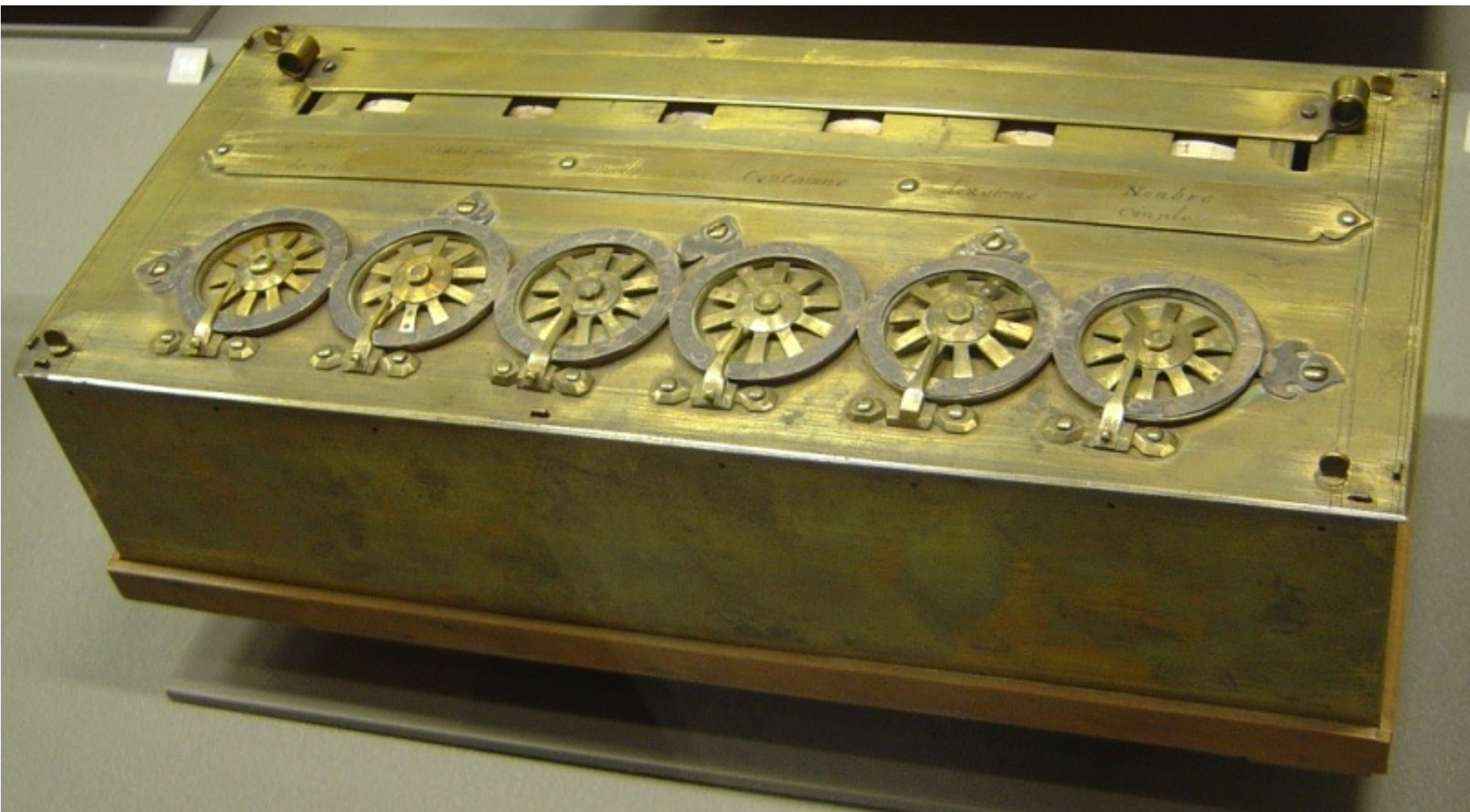
Data source: Wikipedia ([https://en.wikipedia.org/wiki/Transistor\\_count](https://en.wikipedia.org/wiki/Transistor_count))

The data visualization is available at [OurWorldinData.org](https://www.ourworldindata.org). There you find more visualizations and research on this topic.

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



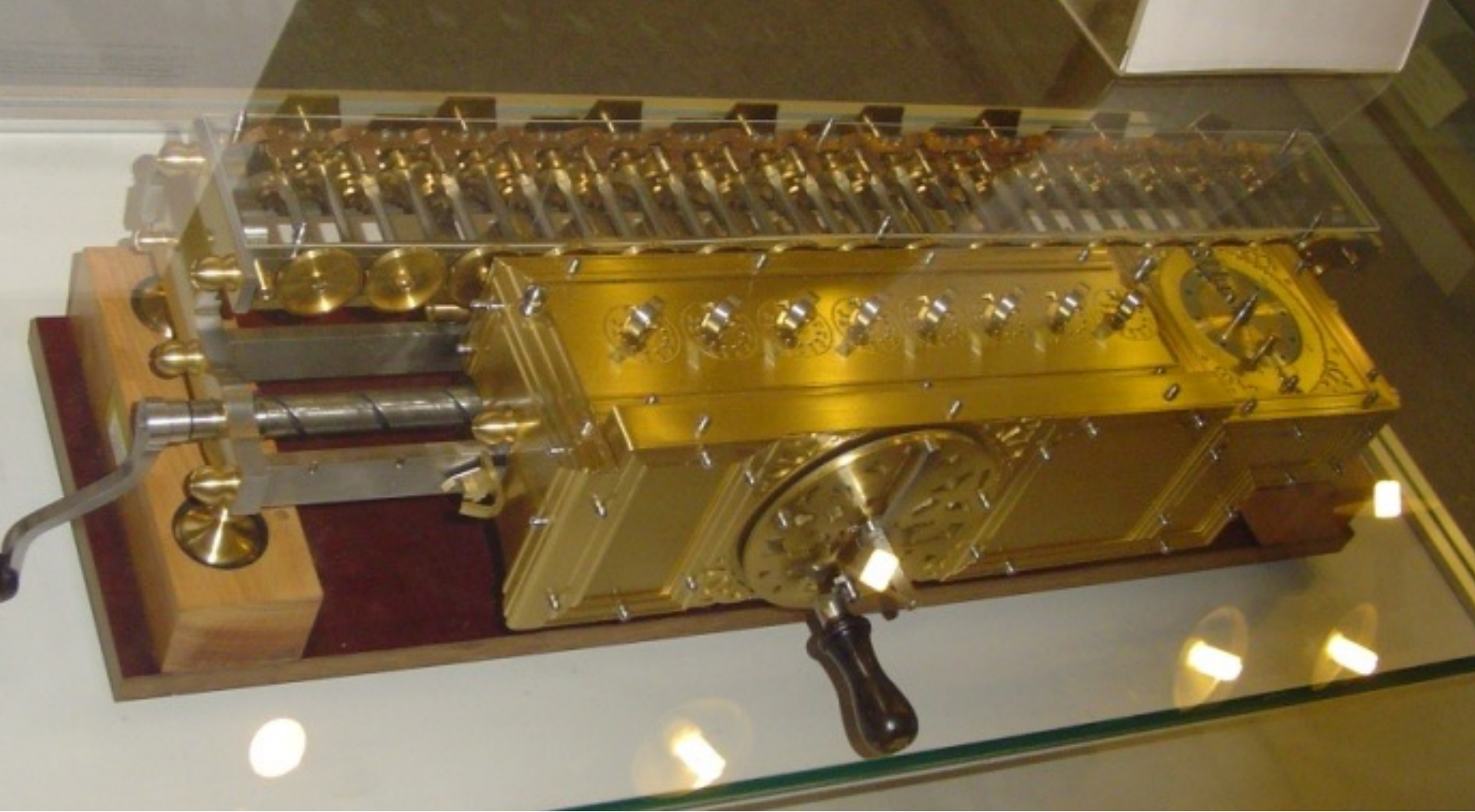




1

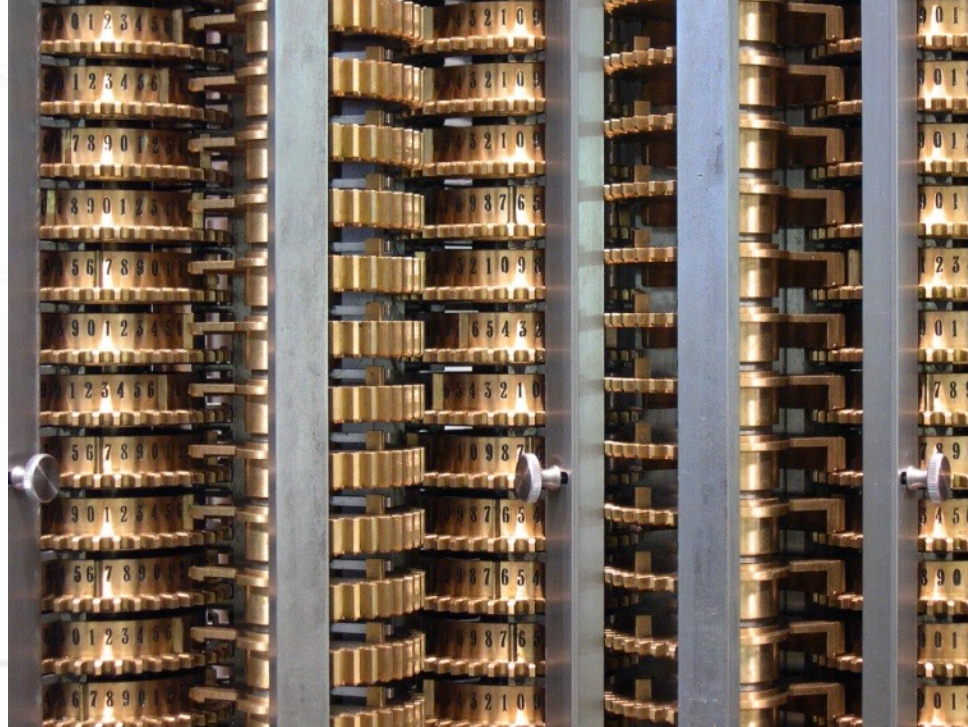
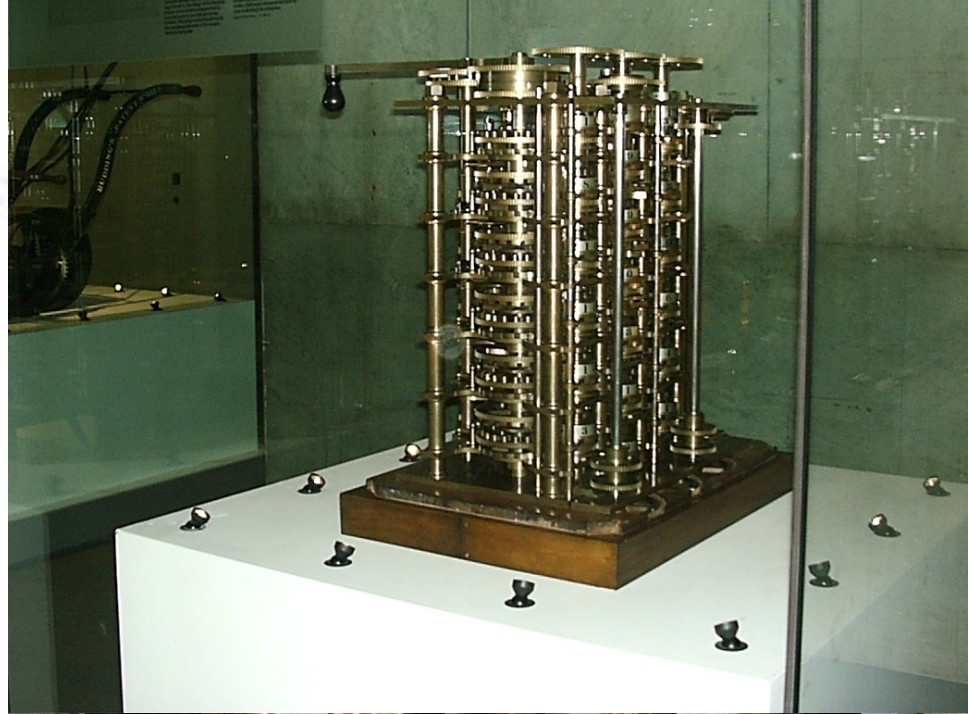
1

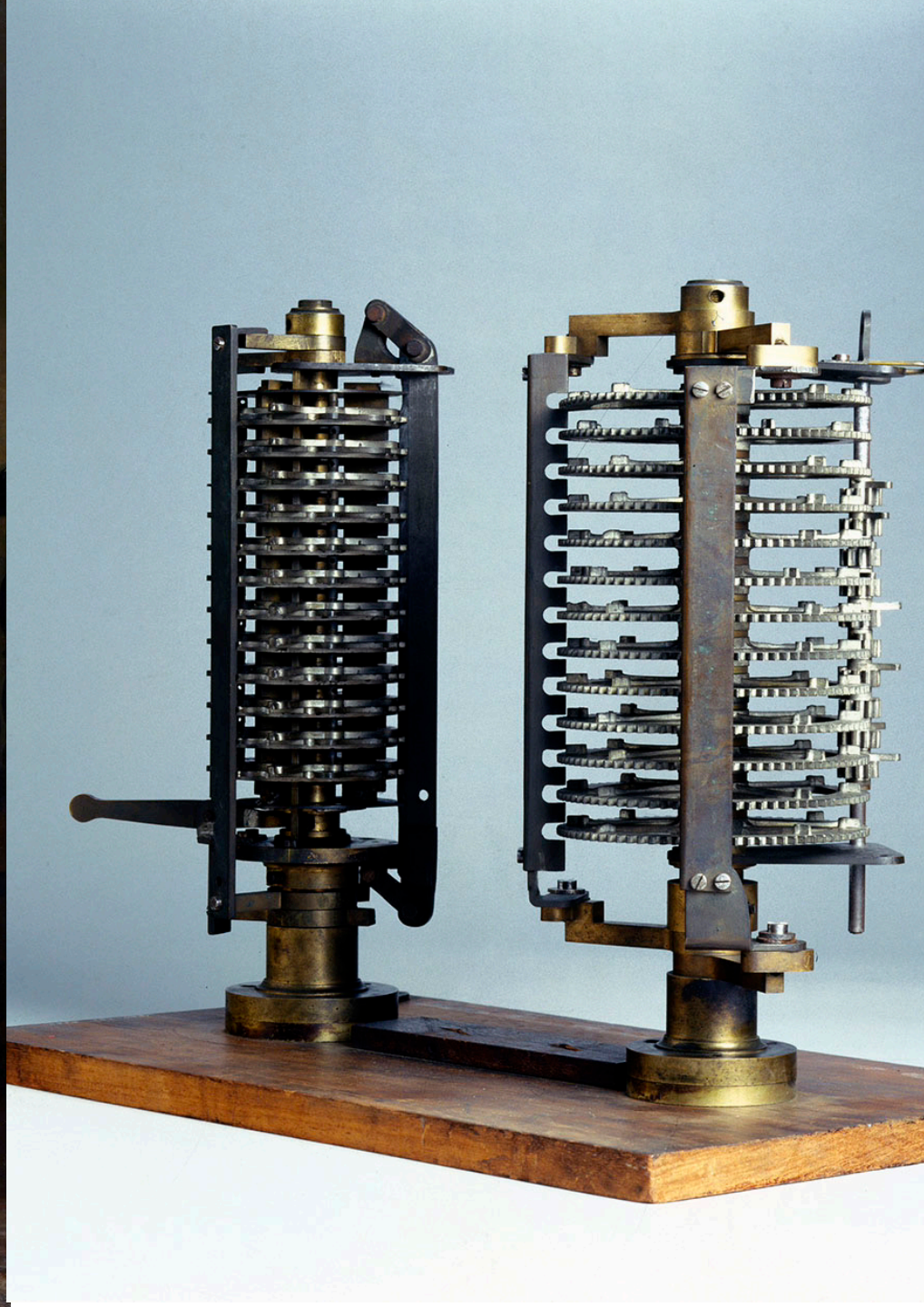
*Cylindre*      *Nombre couple*

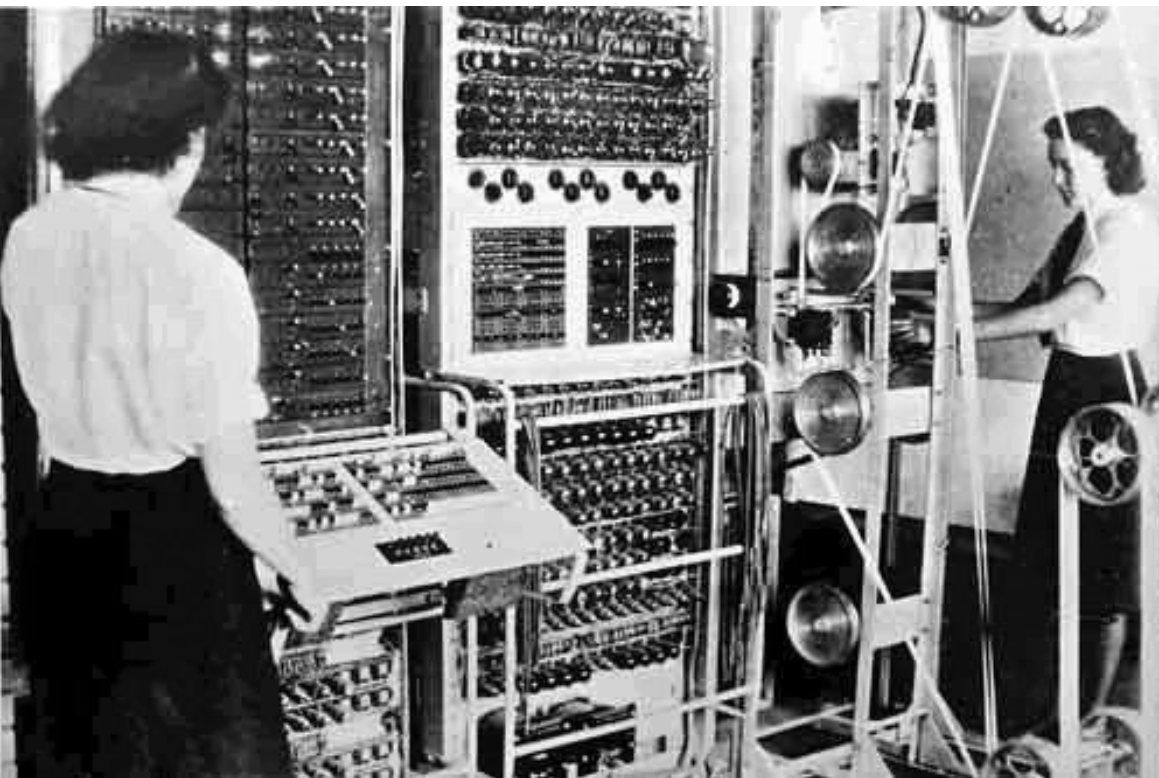




*Charles Babbage*

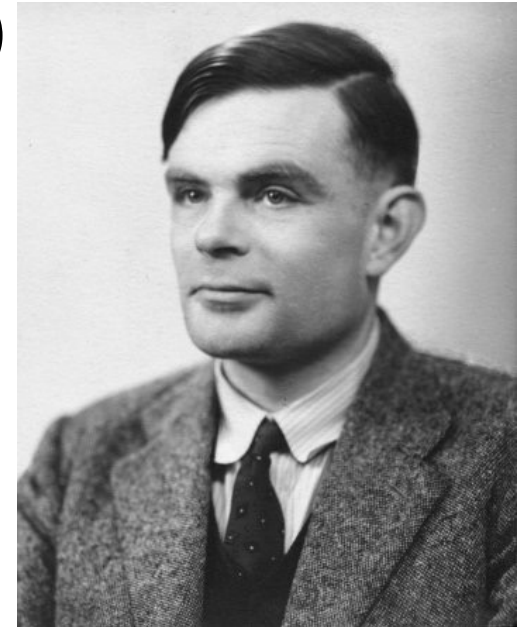


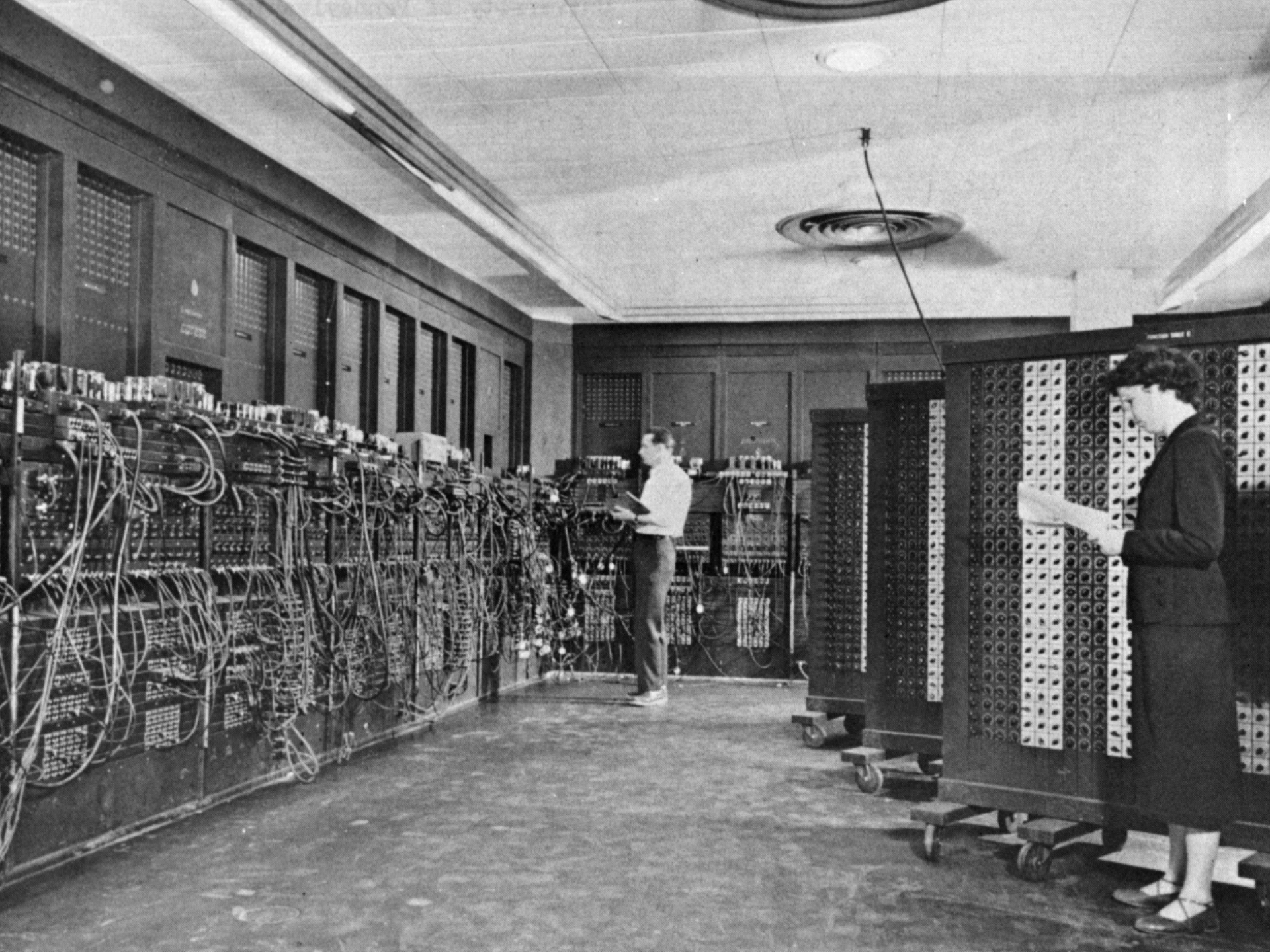




# Alan Turing

- 1912 (London) – 1954 (Wilmslow, Cheshire)
- 1936: On Computable Numbers, with an Application to the “Entscheidungsproblem”  
→ **Turingmaschine**
- 2. Weltkrieg: „Code Knacker“, u.a. „Turing-Bombe“, „Colossus“
- 1945-1948: National Physical Laboratory: ACE (Automatic Computing Engine), Referenz zu Babbages „Analytical Engine“
- 1948-1949: „Manchester Mark I“, Röhrencomputer
- 1950: „Computing machinery and intelligence“  
→ **Turingtest**

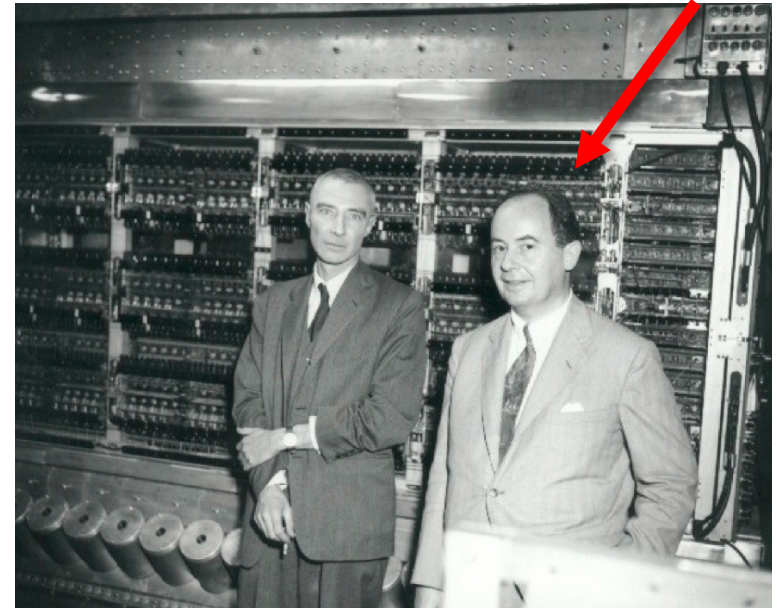




# John von Neumann

## John von Neumann

- \* 1903 (Budapest) als János von Neumann zu Margitta, † 1957 (Washington D.C.)
- U.a. Arbeiten zur Quantenmechanik, Spieltheorie, Manhattan-Projekt (mit Oppenheimer)
- Informatik: **Von-Neumann-Architektur** bzw. Von-Neumann-Rechner: Rechner, in dem Daten und Programm binär codiert in einem Speicher liegen.





A black and white photograph of chess pieces on a board, with a hand moving a piece. The image is used as a background for the book cover.

von Neumann and Morgenstern

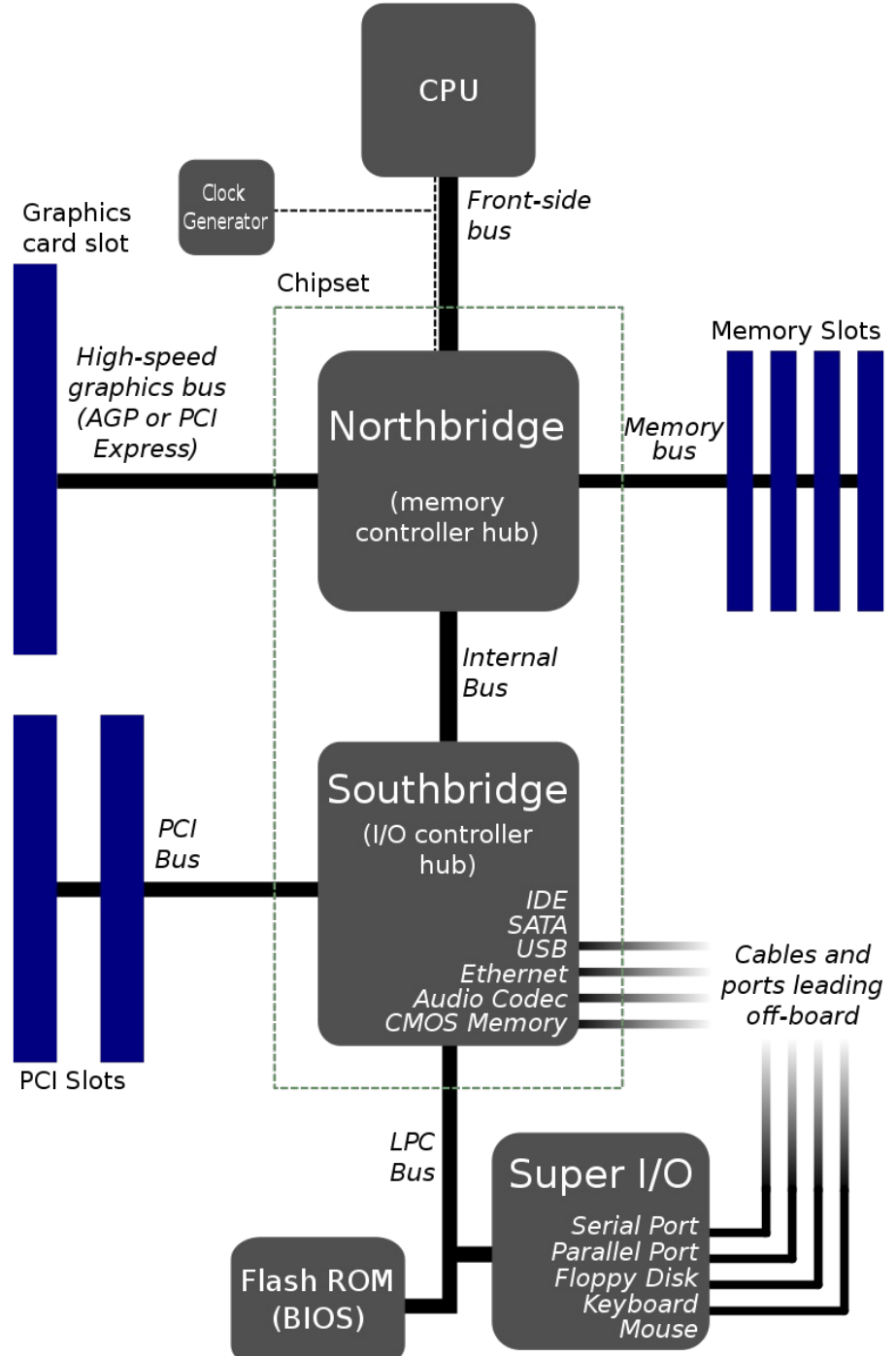
**Theory  
of Games  
and  
Economic  
Behavior**

SIXTIETH-ANNIVERSARY EDITION

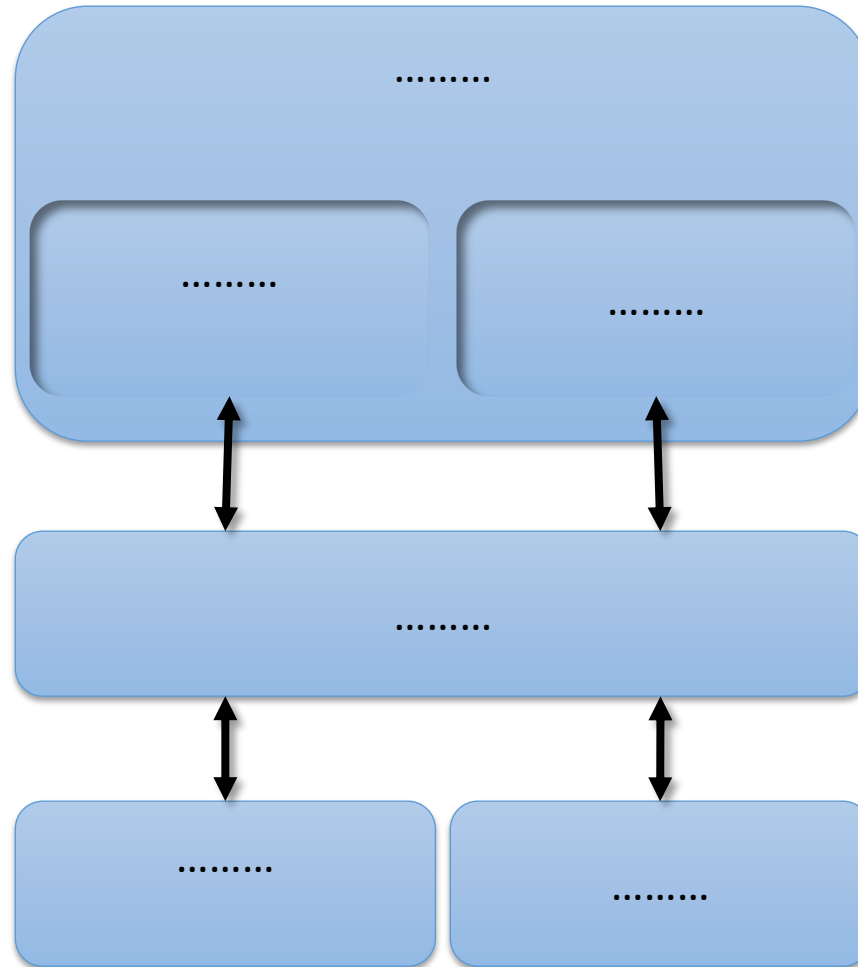
# Die Von-Neuman-Architektur



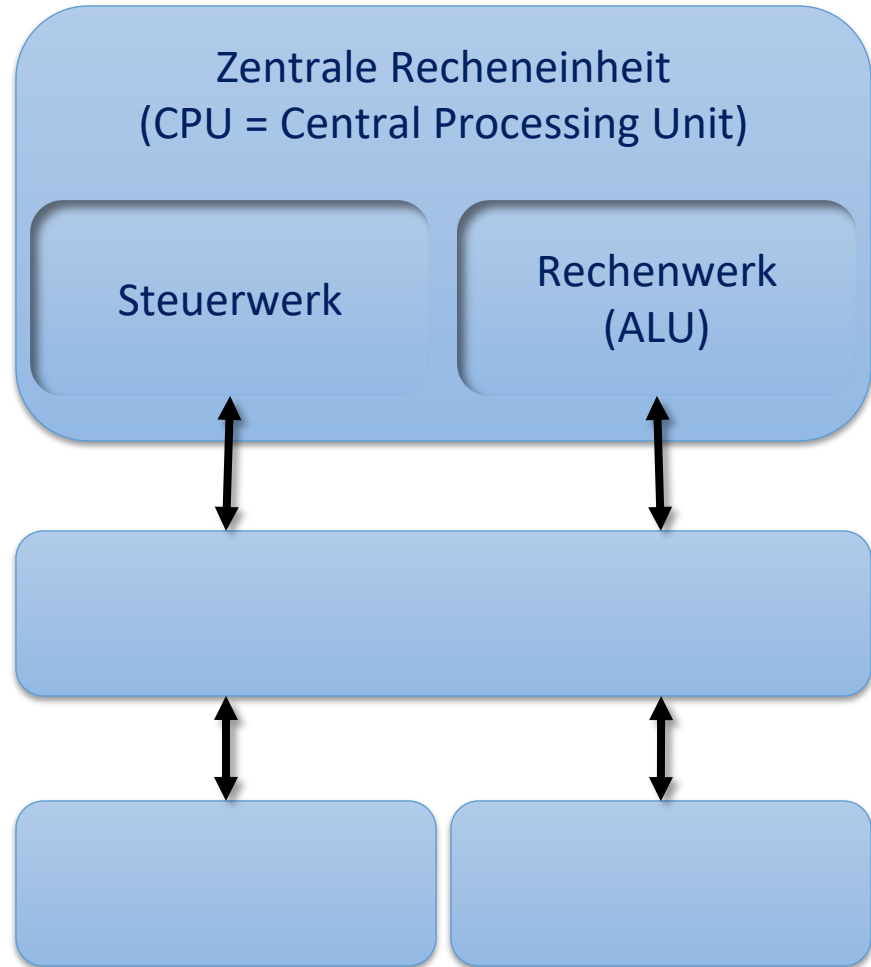




# Von-Neumann-Architektur



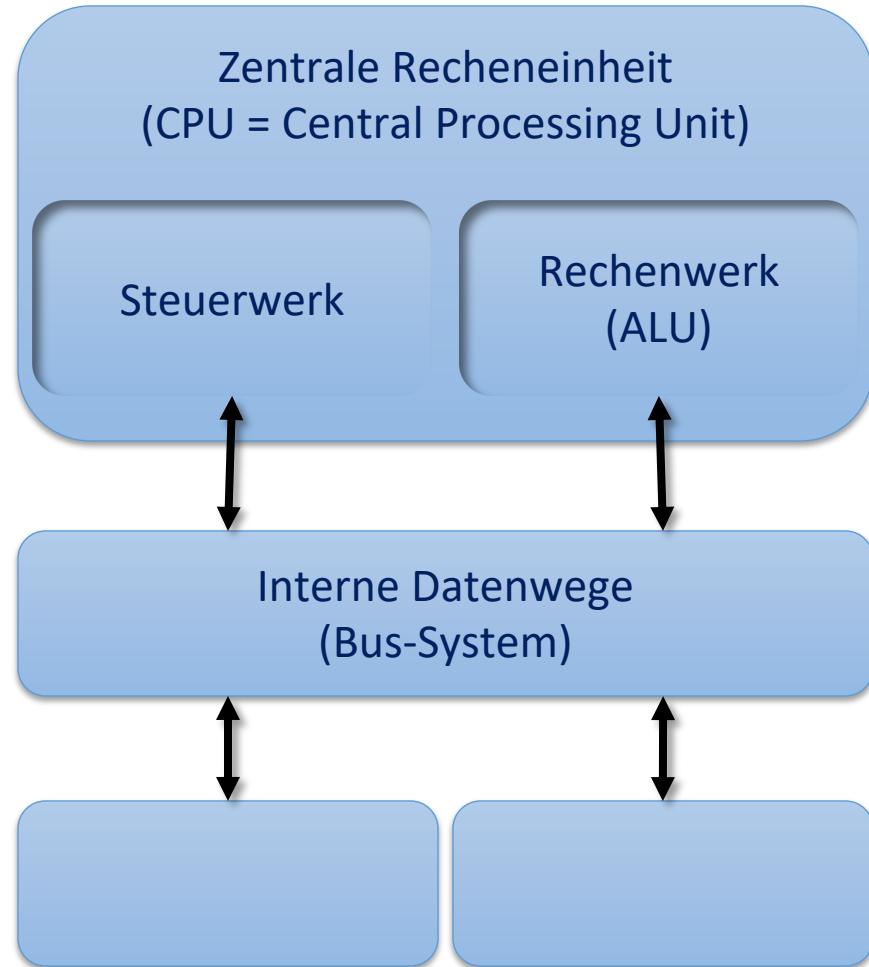
# Von-Neumann-Architektur



# Von-Neumann-Architektur

## Funktionsweise & Eigenschaften

- Zahlen werden im Rechner **binär** dargestellt → **Universalrechner**
- Programme und Daten werden in einem **gemeinsamen Speicher** abgelegt
- Befehle geben nur die **Speicheradresse** an, wo die Daten abgelegt sind, nicht die Daten selbst

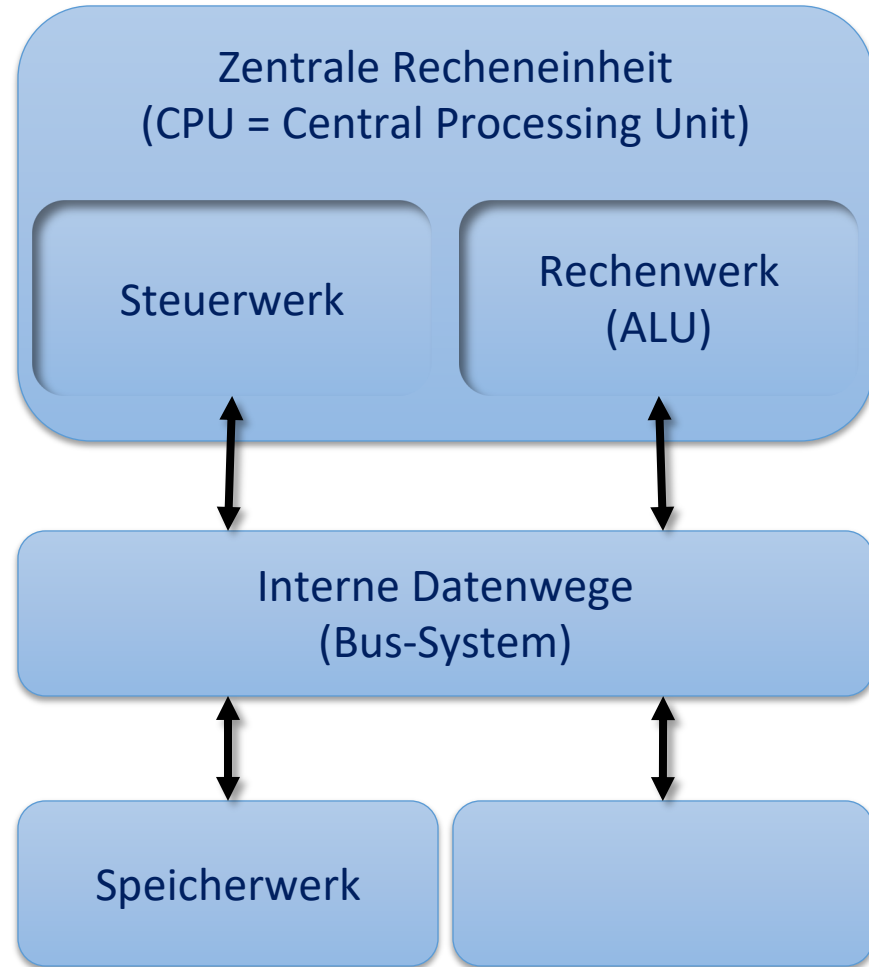




# Von-Neumann-Architektur

## Funktionsweise & Eigenschaften

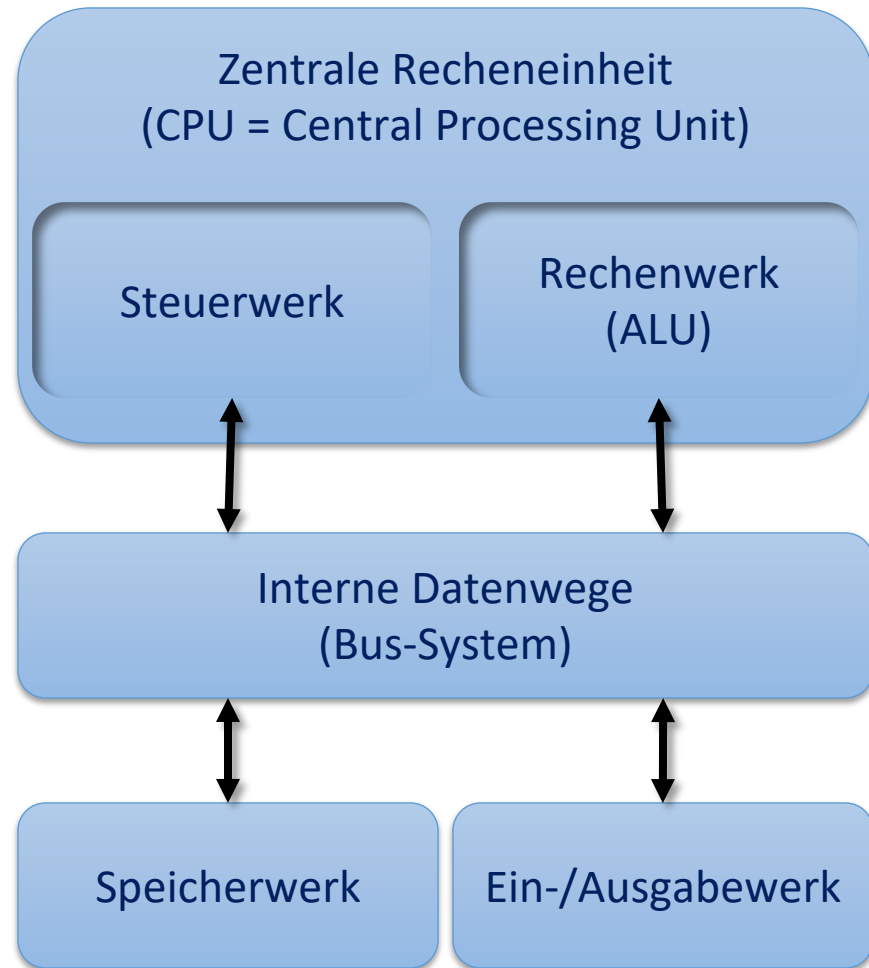
- Zahlen werden im Rechner **binär** dargestellt → **Universalrechner**
- Programme und Daten werden in einem **gemeinsamen Speicher** abgelegt
- Befehle geben nur die **Speicheradresse** an, wo die Daten abgelegt sind, nicht die Daten selbst



# Von-Neumann-Architektur

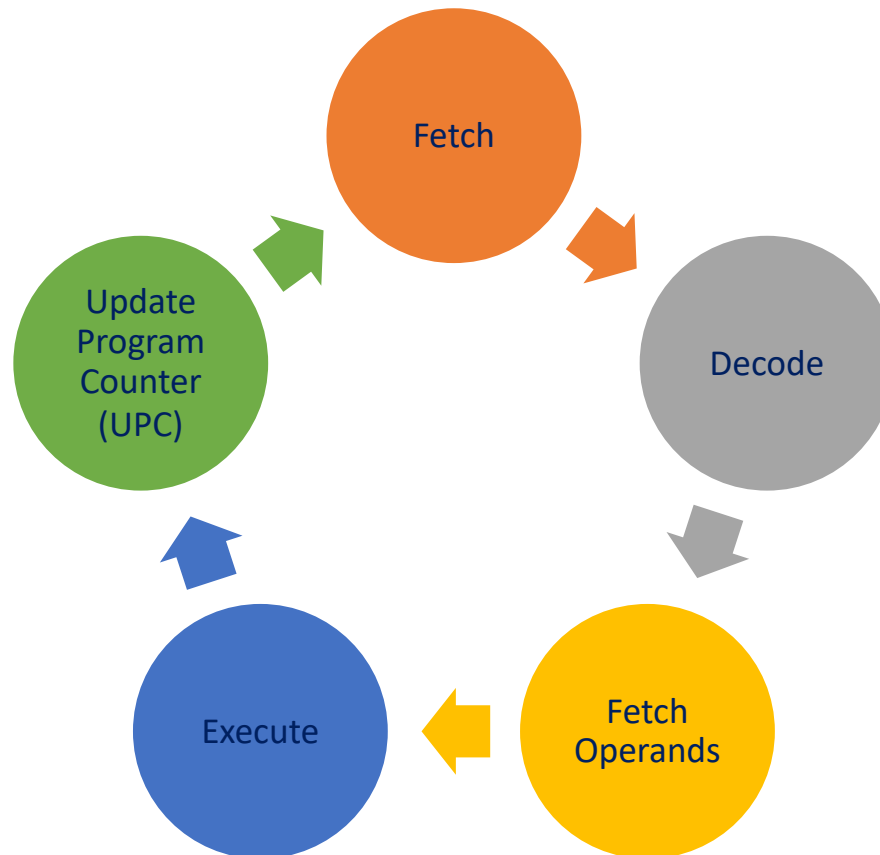
## Funktionsweise & Eigenschaften

- Zahlen werden im Rechner **binär** dargestellt → **Universalrechner**
- Programme und Daten werden in einem **gemeinsamen Speicher** abgelegt
- Befehle geben nur die **Speicheradresse** an, wo die Daten abgelegt sind, nicht die Daten selbst



# Von-Neumann-Architektur

Befehlsverarbeitung → **Von-Neumann-Zyklus** in fünf Teilschritten:



FETCH: Laden des nächsten zu bearbeitenden Befehls in das Befehlsregister (bildet gemeinsam mit Steuerwerk und Rechenwerk die CPU).

FETCH: Laden des nächsten zu bearbeitenden Befehls in das Befehlsregister (bildet gemeinsam mit Steuerwerk und Rechenwerk die CPU).

**DECODE: Befehl wird durch Steuerwerk in Schaltinstruktionen für das Rechenwerk übersetzt.**

FETCH: Laden des nächsten zu bearbeitenden Befehls in das Befehlsregister (bildet gemeinsam mit Steuerwerk und Rechenwerk die CPU).

DECODE: Befehl wird durch Steuerwerk in Schaltinstruktionen für das Rechenwerk übersetzt.

**FETCH OPERANDS: Operanden holen, die durch den Befehl verändert werden sollen**

FETCH: Laden des nächsten zu bearbeitenden Befehls in das Befehlsregister (bildet gemeinsam mit Steuerwerk und Rechenwerk die CPU).

DECODE: Befehl wird durch Steuerwerk in Schaltinstruktionen für das Rechenwerk übersetzt.

FETCH OPERANDS: Operanden holen, die durch den Befehl verändert werden sollen

**EXECUTE: Rechenwerk führt die Operation aus**

FETCH: Laden des nächsten zu bearbeitenden Befehls in das Befehlsregister (bildet gemeinsam mit Steuerwerk und Rechenwerk die CPU).

DECODE: Befehl wird durch Steuerwerk in Schaltinstruktionen für das Rechenwerk übersetzt.

FETCH OPERANDS: Operanden holen, die durch den Befehl verändert werden sollen

EXECUTE: Rechenwerk führt die Operation aus

**UPC: Erhöhung des Befehlszählers, damit der Rechner weiß, an welcher Stelle des Programms er sich gerade befindet. Geschieht parallel zu DECODE und FETCH OPERANDS**



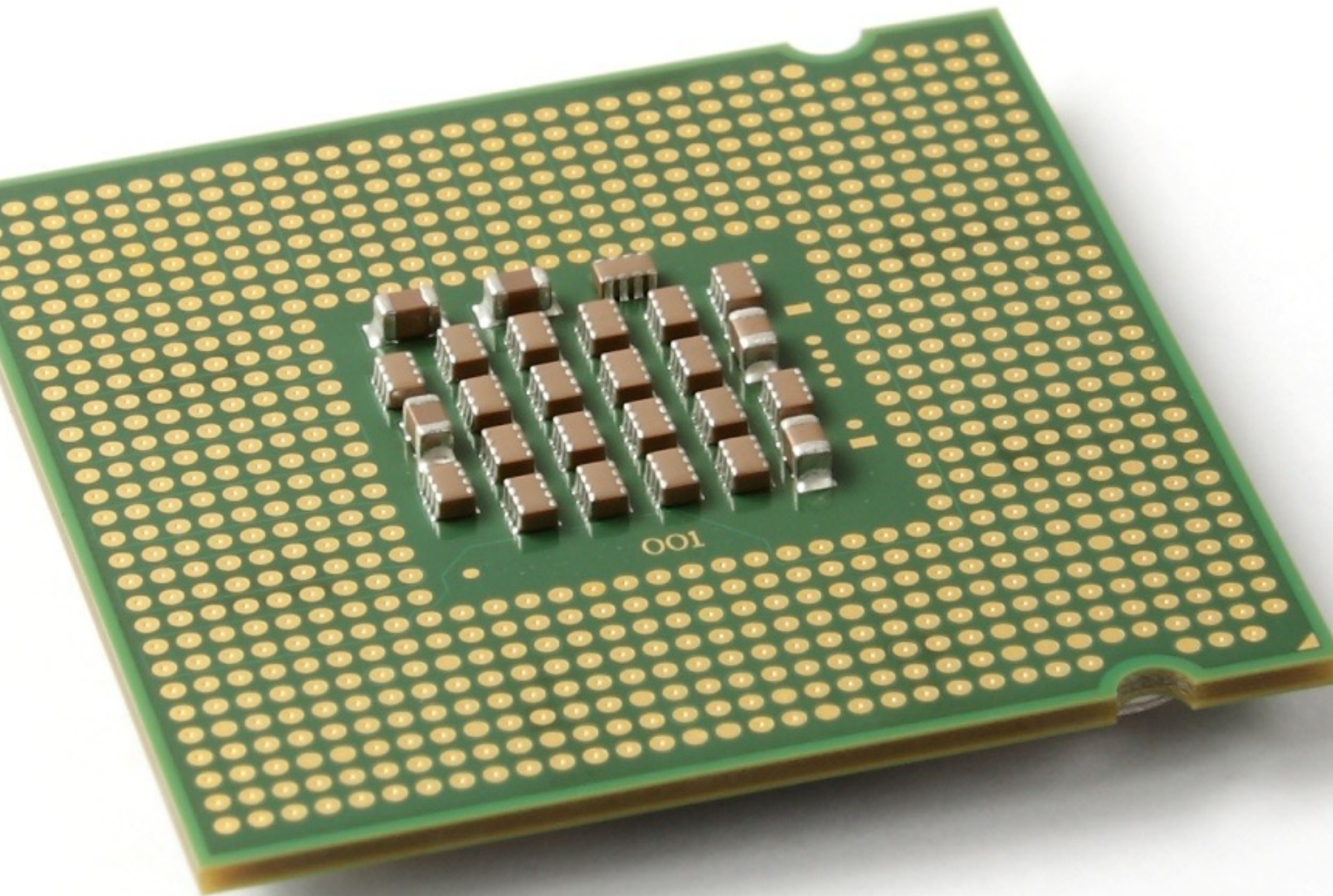


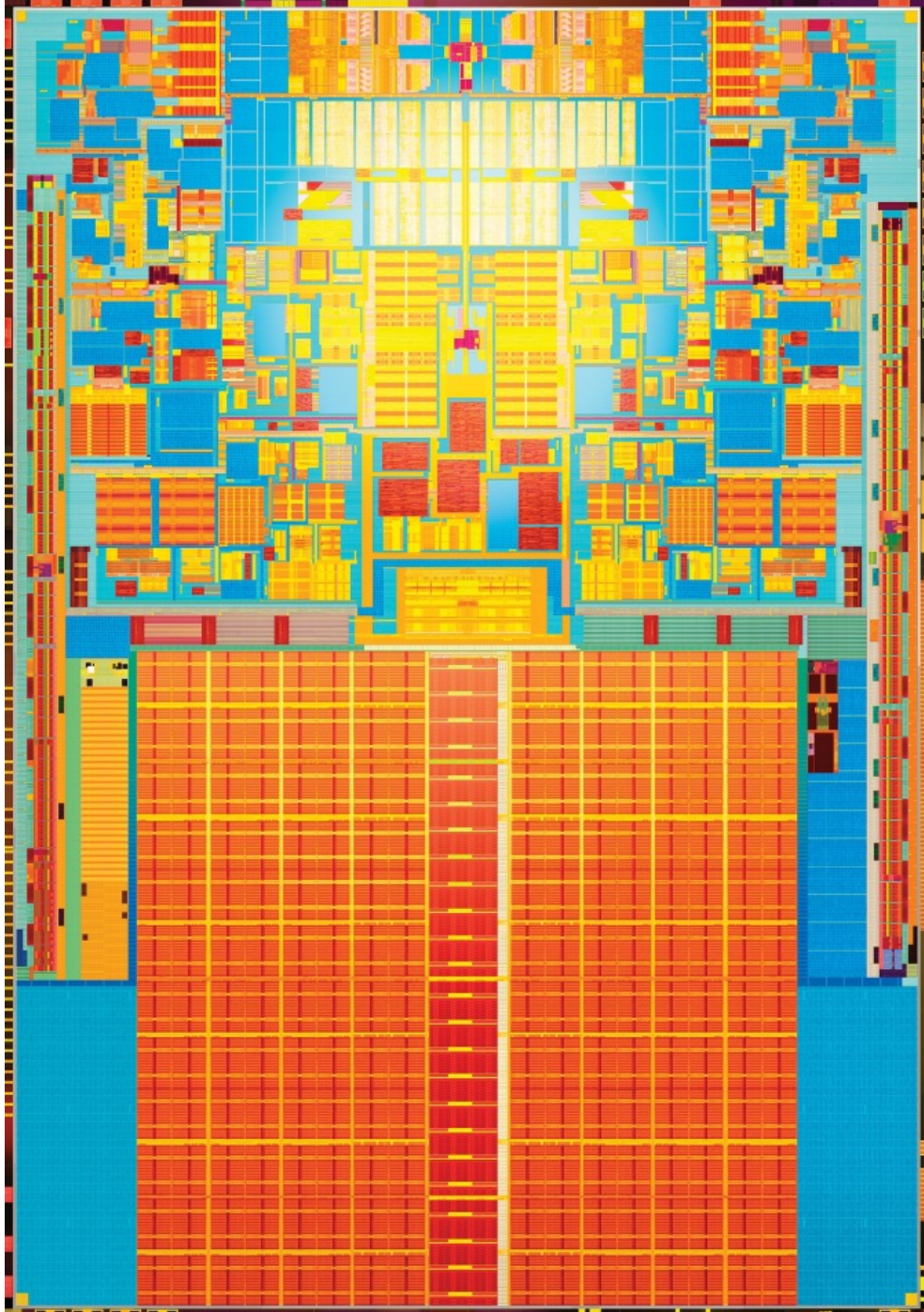
**Paradigmawechsel:** Übergang  
vom starren Programmablauf  
zur flexiblen  
Programmsteuerung bzw. von  
der Rechenmaschine zur  
Datenverarbeitungsmaschine



**Von-Neumann-Flaschenhals**

Caching



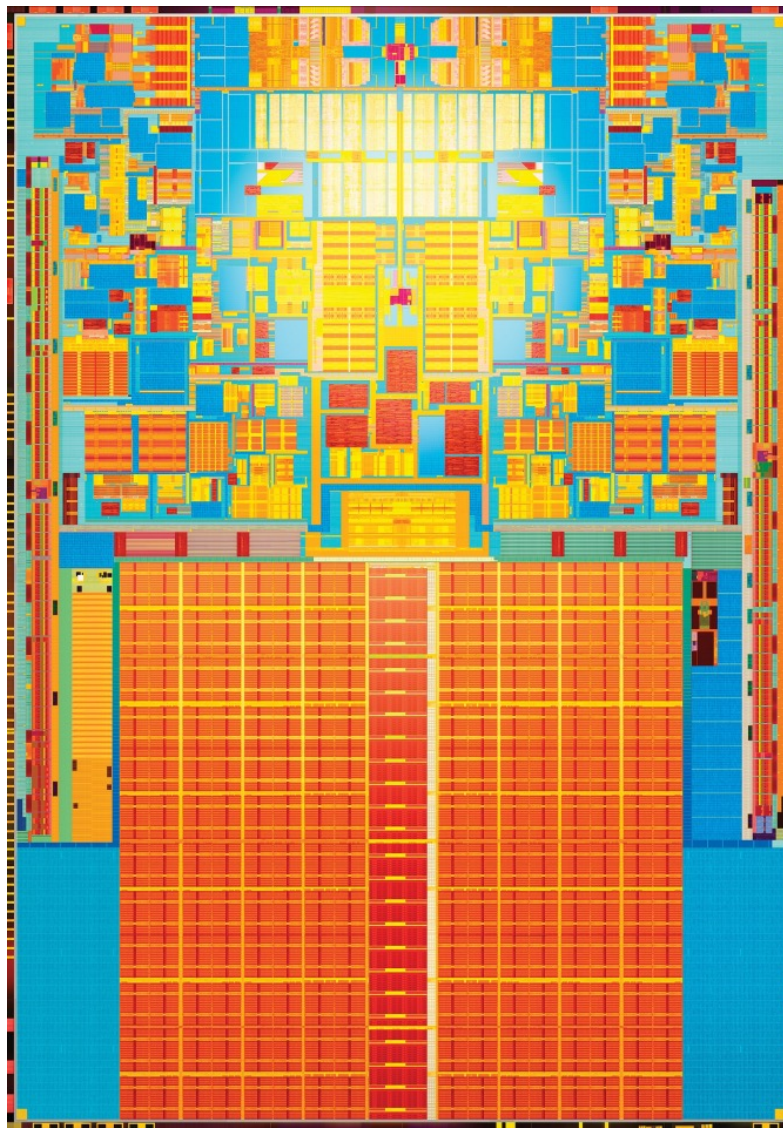




Cache-Hit: Datum / Instruktion befindet sich im Cache.



Cache-Miss: Datum / Instruktion befindet sich nicht im Cache.  
→ Worst Case: Laden von Daten aus dem deutlich langsameren Hauptspeicher.



L1



L2



L3



Ln

# Strukturierte Computerorganisation



**Ebene 5**

Problemorientierte Sprache

**Ebene 4**

Assemblersprache

**Ebene 3**

Betriebssystemmaschine

**Ebene 2**

Befehlssatzarchitektur (ISA)

**Ebene 1**

Mikroarchitektur

**Ebene 0**

Digitale Logik





Digitaltechnik  
boolesche- / Schaltalgebra

# Werner-von-Siemens-Schule Köln

Das Berufskolleg für Elektrotechnik in Köln-Deutz

[Unsere Schule](#) [Kontakt](#) [Termine](#) [Bildungsgänge](#) [Flyer](#) [EU-Austausch](#) [Schüler](#) [Lehrer](#)

## INFORMATIONSVANSTALTUNGEN DER KÖLNER BERUFKOLLEGS 2021



Zur Orientierung und Information bieten die Kölner Berufskollegs und die Bildungsberatung der Stadt Köln über 30 Informationsveranstaltungen an den Berufskollegs an. Zwischen dem 12. und 27. November können Sie sich direkt über die Angebote der Kölner Berufskollegs vor Ort informieren. Am 13. und 18. November stehen wir für Sie zur Verfügung. Eine vorherige Anmeldung für jede einzelne Person ist unbedingt erforderlich....

[weiterlesen](#)

## SCHÜLERINNEN UND SCHÜLER DER KLASSEN 3125 UND 1021 HELFE BETROFFENEN WINZERN IM AHRTAL!



Auch 12 Wochen nach der Flutkatastrophe am 14. Juli 2021 gibt es in den betroffenen Gebieten in NRW und Rheinland-Pfalz noch viel zu tun. Die Menschen sind auf jede helfende Hand angewiesen und unendlich dankbar für jede Hilfe. Auch die Winzer im Ahrtal hat es schwer getroffen. Viele Weingüter und Weinstöcke sind zerstört. Ein Großteil der Klassen 3125 und 1021 fahren...

[weiterlesen](#)



CAMPUS DEUTZ



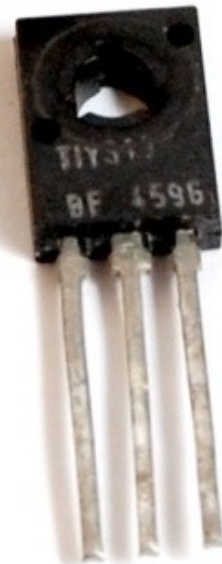
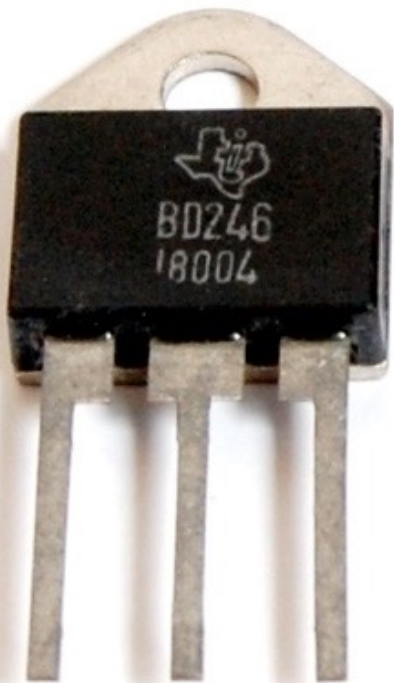
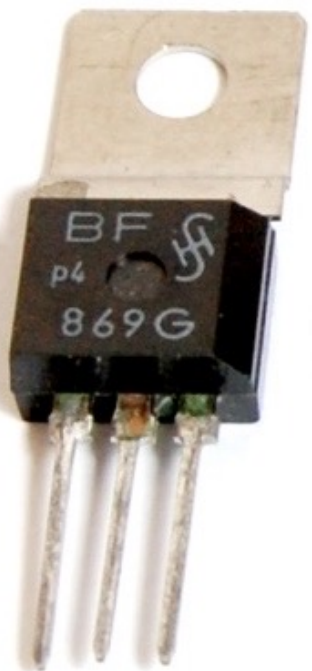
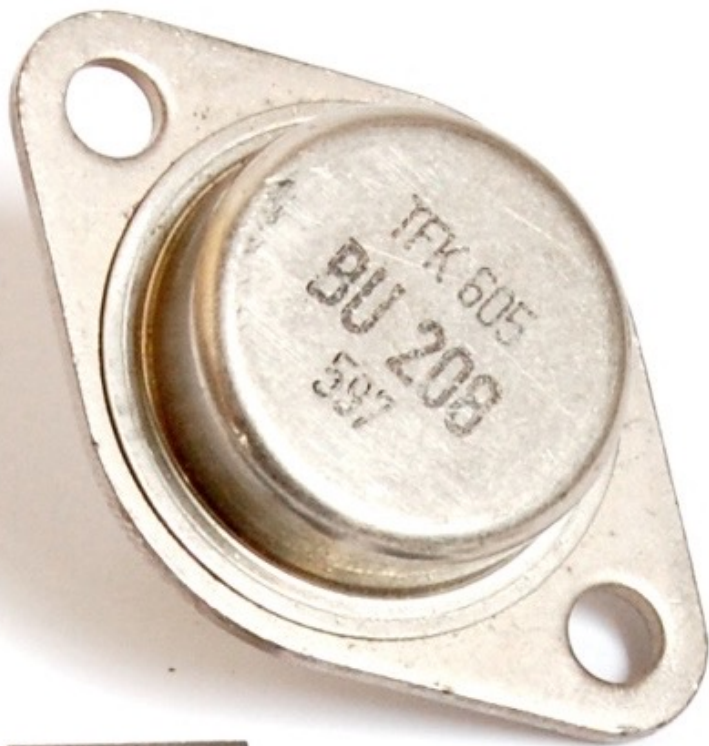
### Werner-von-Siemens-Schule

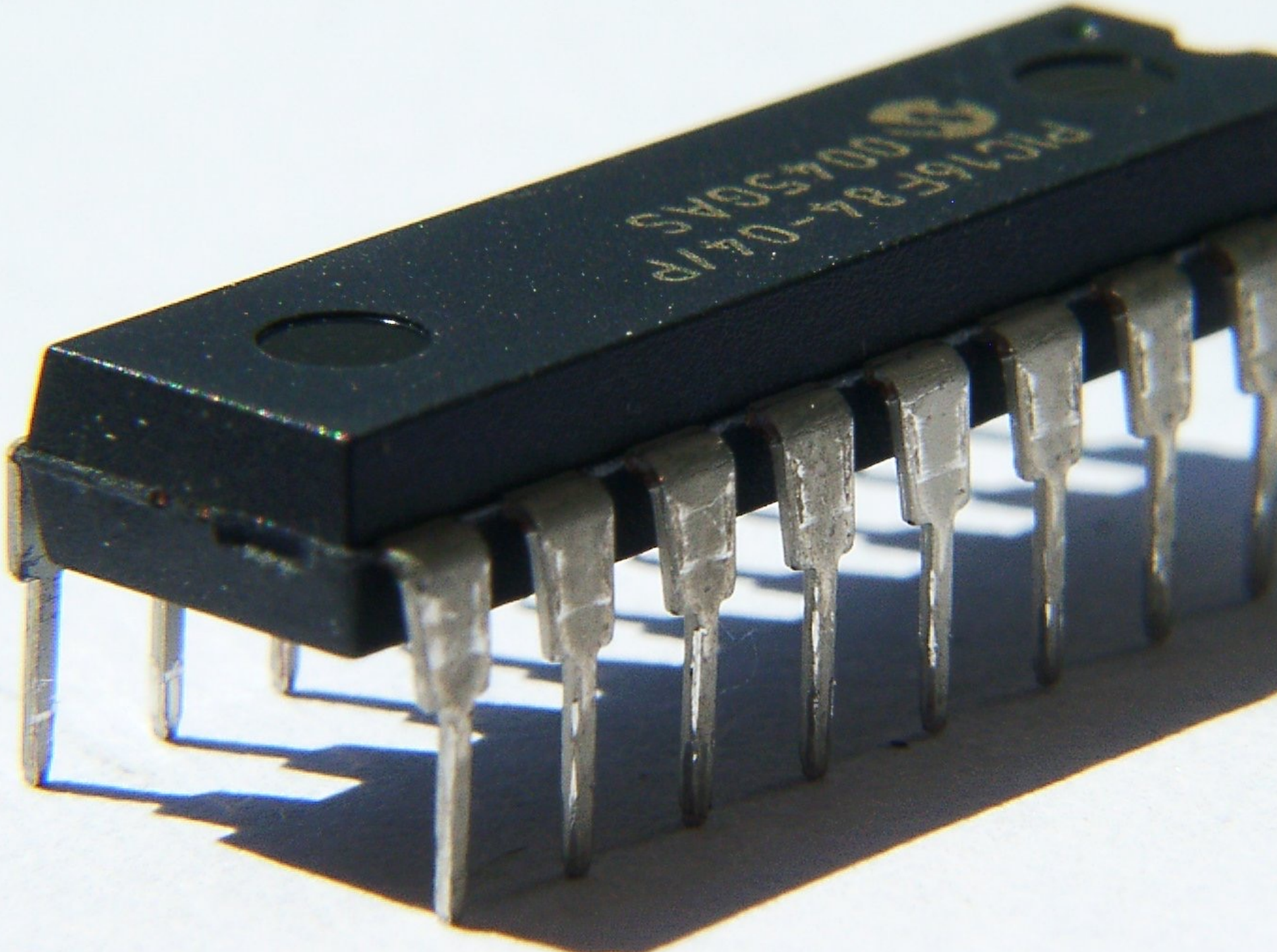
Berufskolleg für Elektrotechnik  
Sekundarstufe II  
Berufliches Gymnasium

### Aktuelle Informationen

- Die Unterrichtszeiten haben sich geändert. Die erste Stunde beginnt jetzt um 8.15 Uhr. Alle Uhrzeiten finden Sie auf der Seite "[Unterrichtszeiten](#)".







74LS163  
005040

# (Logik)Gatter

Vereinfacht: Blackbox mit n Eingängen und einem Ausgang



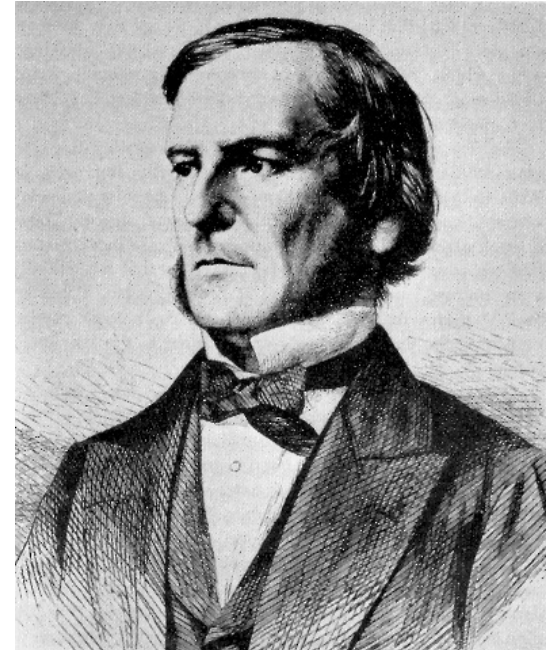
Eingänge / Ausgang: Spannungszustände, i.e. 0 Volt für 0 und 5 Volt für 1

# Schaltalgebra

# Boolesche Algebra / Schaltalgebra

Beschreibung von Schaltungen, die sich durch Kombination von Gattern aufbauen lassen über **Boolesche Algebra**:

- George Boole (1815-1864)
- Variablen und Funktionen können nur die Werte 1 (wahr, TRUE) und 0 (falsch, FALSE) annehmen bzw. zurückgeben.
- Z.B. Datentyp *bool* in C++
- Vollständige Beschreibung der Booleschen Fkt. über Tabelle mit  $2^n$  Zeilen, wobei  $n$  gleich Anzahl der Eingangsvariablen / -werte → **Wahrheitstabelle**
- **Schaltalgebra** kennt zwei Konstanten: 0 (Schalter geschlossen / Leitung unterbrochen) und 1 (Schalter offen / Leitung durchgeschaltet)





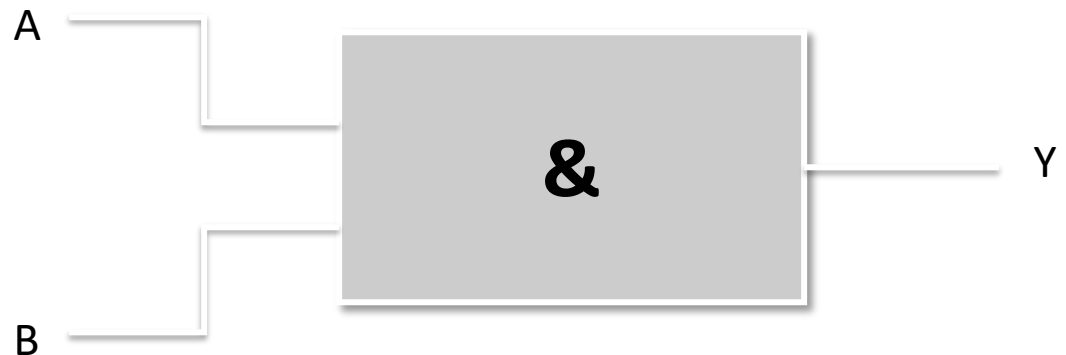
# Wahrheitstabelle

Für zwei Eingänge (A, B):  $2^2=4$  Tabellenzeilen

0  $\triangleq$  falsch

1  $\triangleq$  wahr

A	B	Y
0	0	
0	1	
1	0	
1	1	



# Wahrheitstabelle

Für zwei Eingänge (A, B):  $2^2=4$  Tabellenzeilen

0  $\triangleq$  falsch

1  $\triangleq$  wahr

A	B	Y
f	f	
f	t	
t	f	
t	t	



# Wahrheitstabelle

Für drei Eingänge (A, B, C):  $2^3=8$  Tabellenzeilen

A	B	C	Y
0	0	0	
0	0	1	
0	1	0	
0	1	1	
1	0	0	
1	0	1	
1	1	0	
1	1	1	

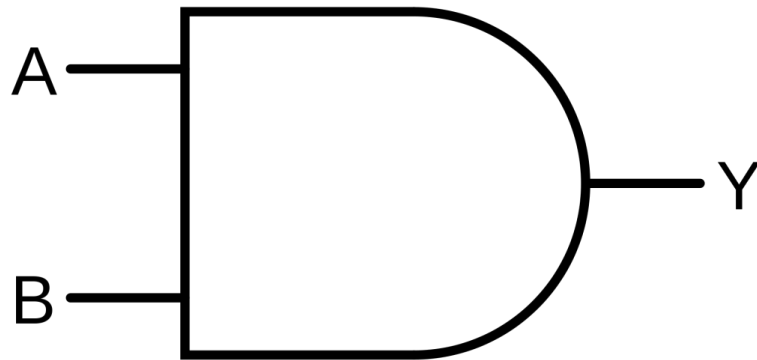
# Gattertypen / Verknüpfungsarten

Verschiedene **Gattertypen**, d.h. Arten, Eingangssignale miteinander zu **verknüpfen**:

- UND (AND)
- ODER (OR)
- NICHT (NOT)
- NICHT UND (NAND)
- ...

# Gattertypen: UND / AND –Gatter → Konjunktion

**Symbol** (nach US ANSI 91-1984)



**Funktion**

$$Y = A \wedge B$$

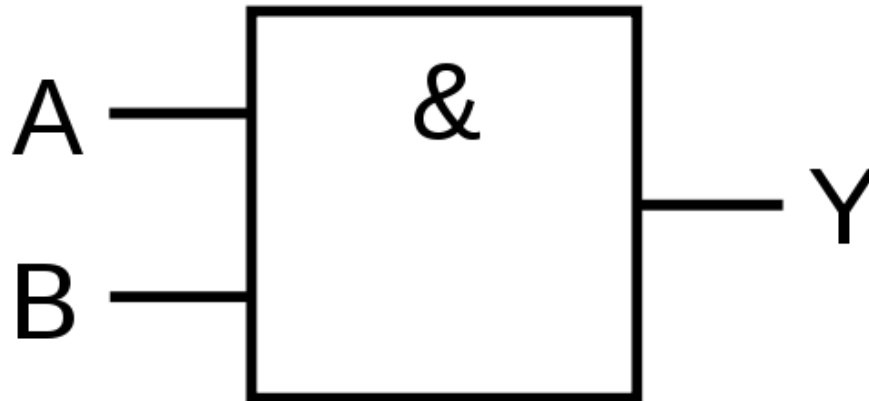
**Wahrheitstabelle**

A	B	Y
0	0	0
0	1	0
1	0	0
1	1	1

# Gattertypen: UND / AND –Gatter → Konjunktion

**Symbol** (nach IEC 60617-12)

IEC: International Electrotechnical Commission



**Funktion**

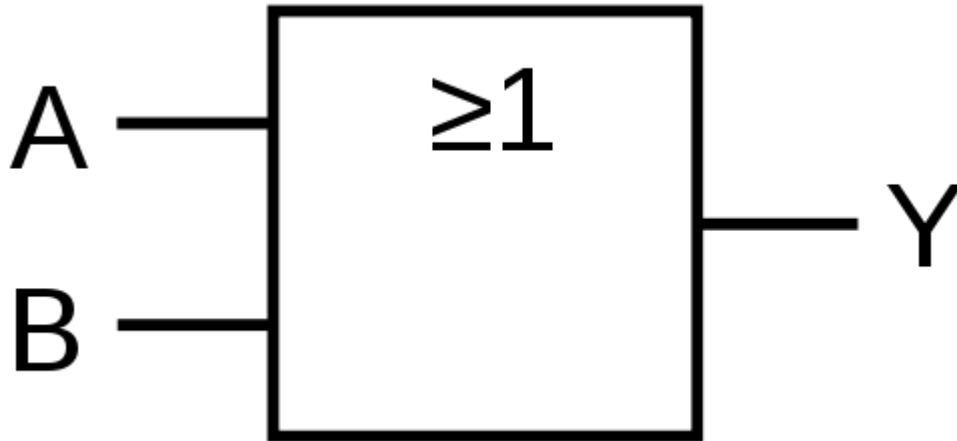
$$Y = A \wedge B$$

**Wahrheitstabelle**

A	B	Y
0	0	0
0	1	0
1	0	0
1	1	1

# Gattertypen: ODER / OR –Gatter → Disjunktion

Symbol



Funktion

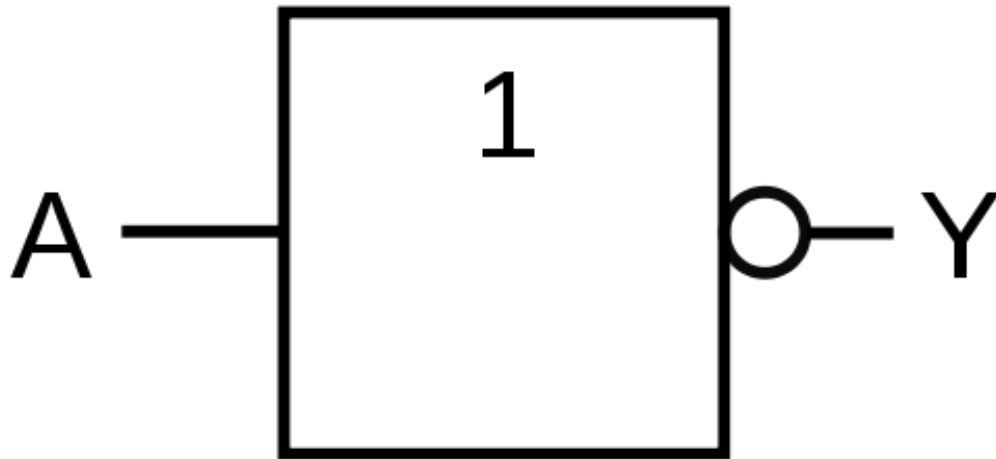
$$Y = A \vee B$$

Wahrheitstabelle

A	B	Y
0	0	0
0	1	1
1	0	1
1	1	1

# Gattertypen: NICHT / NOT –Gatter → Negation

Symbol



Funktion  
 $Y = \neg A$

Wahrheitstabelle

oder

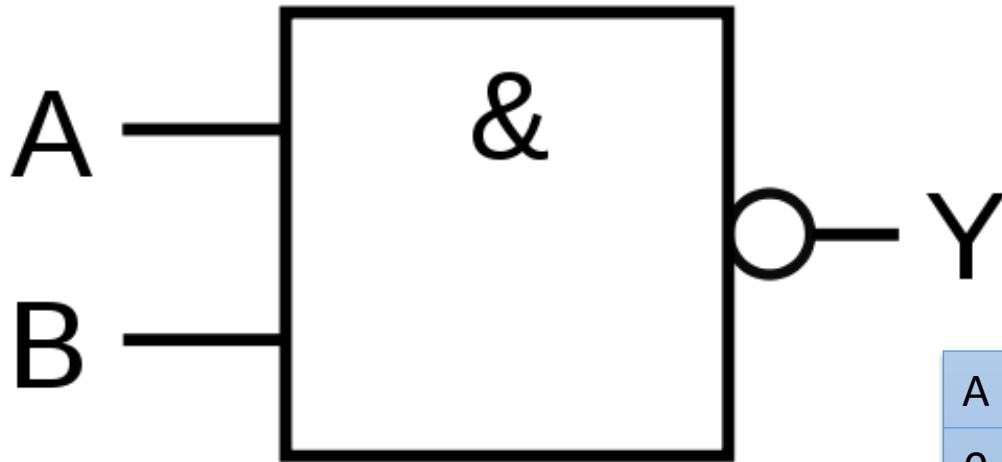
$Y = \overline{A}$

A	Y
0	1
1	0



# Übung 1

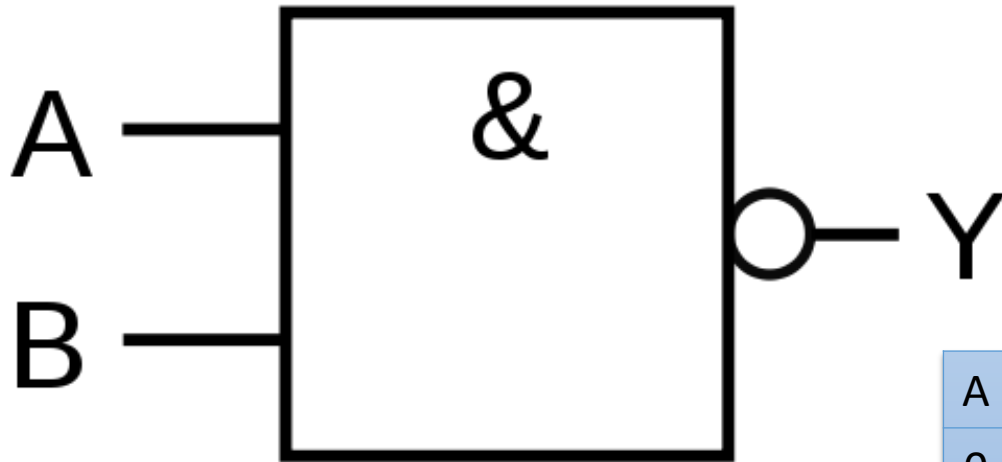
Bestimmen Sie die Wahrheitstabelle für das folgende Gatter:



A	B	$A \wedge B$	$Y = \overline{A \wedge B}$
0	0	0	
0	1	0	
1	0	0	
1	1	1	

# Übung 1

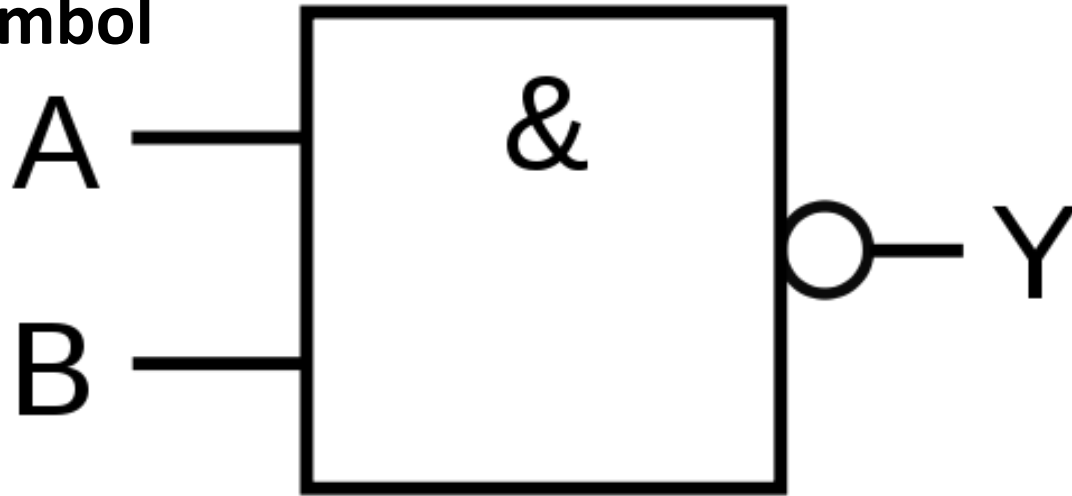
Bestimmen Sie die Wahrheitstabelle für das folgende Gatter:



A	B	$A \wedge B$	$Y = \overline{A \wedge B}$
0	0	0	1
0	1	0	1
1	0	0	1
1	1	1	0

# Gattertypen: NICHT UND / NAND Gatter

Symbol



Funktion

$$Y = A \wedge B$$

oder

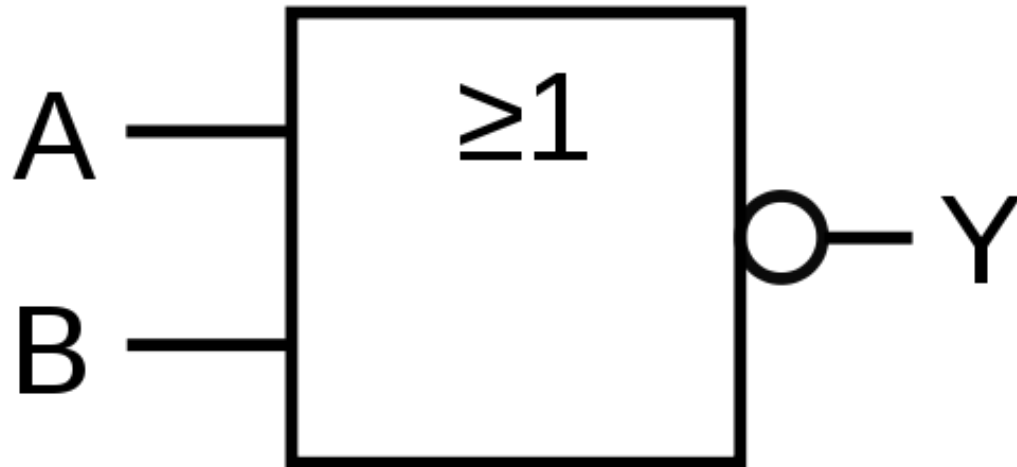
$$Y = \neg(A \wedge B)$$

Wahrheitstabelle

A	B	Y
0	0	1
0	1	1
1	0	1
1	1	0

# Gattertypen: NICHT ODER / NOR Gatter

Symbol



Funktion

$$Y = A \vee B$$

oder

$$Y = \neg(A \vee B)$$

Wahrheitstabelle

A	B	Y
0	0	1
0	1	0
1	0	0
1	1	0

## Übung 2

Bestimmen Sie die vollständigen Wahrheitstabellen für die folgenden Funktionsgleichungen:

- $Y = (A \vee B) \wedge \neg A$
- $Y = (A \wedge B) \wedge \neg (B \vee A)$
- $Y = (A \wedge B) \vee (A \wedge C)$
- $C = A \wedge B$   
 $Y = C \wedge C$  (Eingänge des Gatters kurzgeschlossen)

## Übung 2: Schritt 1

Bestimmen Sie die vollständigen Wahrheitstabellen für die folgende Funktionsgleichung:

$$Y = (A \vee B) \wedge \neg A$$

A	B
0	0
0	1
1	0
1	1

## Übung 2: Schritt 2

Bestimmen Sie die vollständigen Wahrheitstabellen für die folgende Funktionsgleichung:

$$Y = (A \vee B) \wedge \neg A$$

A	B	$A \vee B$
0	0	0
0	1	1
1	0	1
1	1	1

## Übung 2: Schritt 3

Bestimmen Sie die vollständigen Wahrheitstabellen für die folgende Funktionsgleichung:

$$Y = (A \vee B) \wedge \neg A$$

A	B	$A \vee B$	$\neg A$
0	0	0	1
0	1	1	1
1	0	1	0
1	1	1	0



## Übung 2: Schritt 4

Bestimmen Sie die vollständigen Wahrheitstabellen für die folgende Funktionsgleichung:

$$Y = (A \vee B) \wedge \neg A$$

A	B	$A \vee B$	$\neg A$	Y
0	0	0	1	<b>0</b>
0	1	1	1	<b>1</b>
1	0	1	0	<b>0</b>
1	1	1	0	<b>0</b>

## Übung 2

Bestimmen Sie die vollständigen Wahrheitstabellen für die folgende Funktionsgleichung:

$$Y = (A \wedge B) \wedge \neg (B \vee A)$$

A	B	$A \wedge B$	$B \vee A$	$\neg (B \vee A)$	$Y = (A \wedge B) \wedge \neg (B \vee A)$
0	0	0	0	1	<b>0</b>
0	1	0	1	0	<b>0</b>
1	0	0	1	0	<b>0</b>
1	1	1	1	0	<b>0</b>

# Übung 2

Bestimmen Sie die vollständigen Wahrheitstabellen für die folgende Funktionsgleichung:

$$Y = (A \wedge B) \vee (A \wedge C)$$

A	B	C	$A \wedge B$	$A \wedge C$	$Y = (A \wedge B) \vee (A \wedge C)$
0	0	0	0	0	<b>0</b>
0	0	1	0	0	<b>0</b>
0	1	0	0	0	<b>0</b>
0	1	1	0	0	<b>0</b>
1	0	0	0	0	<b>0</b>
1	0	1	0	1	<b>1</b>
1	1	0	1	0	<b>1</b>
1	1	1	1	1	<b>1</b>

## Übung 2

Bestimmen Sie die vollständigen Wahrheitstabellen für die folgende Funktionsgleichung:

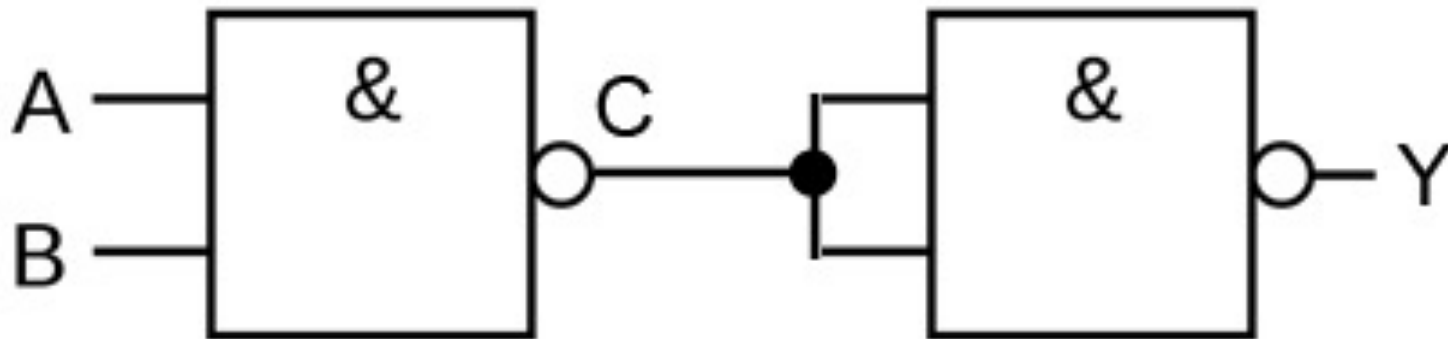
$$C = A \wedge B$$

$$Y = C \wedge C \text{ (Eingänge des Gatters kurzgeschlossen)}$$

A	B	$C = A \wedge B$	$Y = C \wedge C$
0	0	0	<b>0</b>
0	1	0	<b>0</b>
1	0	0	<b>0</b>
1	1	1	<b>1</b>

# Übung 3

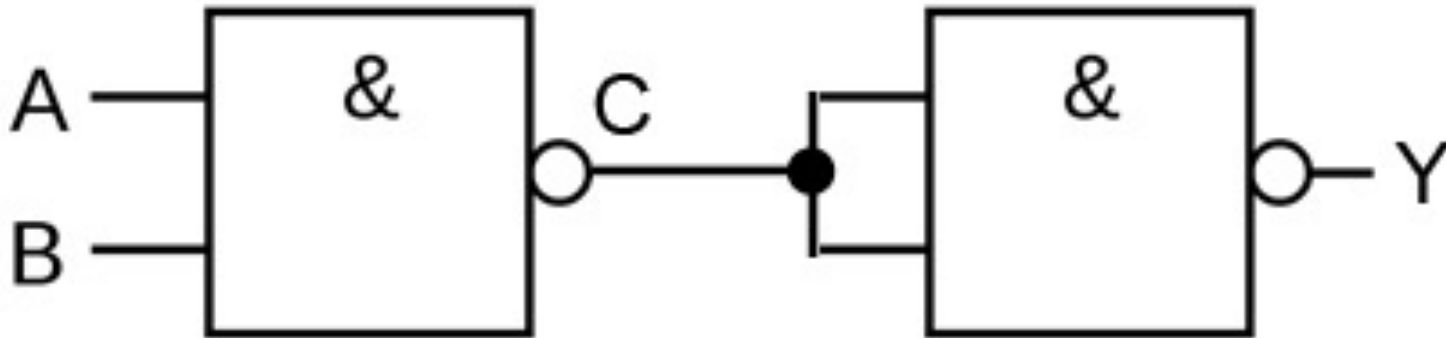
Bestimmen Sie die Wahrheitstabelle für die folgende Schaltung:



A	B	C	Y

# Übung 3

Bestimmen Sie die Wahrheitstabelle für die folgende Schaltung:



A	B	C	Y
0	0	1	0
0	1	1	0
1	0	1	0
1	1	0	1

/

# Bildnachweise

- <http://www.reactiongifs.com>
- John von Neumann: Von LANL - <http://www.lanl.gov/history/atomicbomb/images/NeumannL.GIF> (Archivkopie), Attribution, <https://commons.wikimedia.org/w/index.php?curid=3429594>
- [https://commons.wikimedia.org/wiki/File:Hauptverwaltung\\_1%261\\_Versatel\\_.jpg](https://commons.wikimedia.org/wiki/File:Hauptverwaltung_1%261_Versatel_.jpg)