



max planck institut  
informatik

# Ideen und Konzepte der Informatik

## Rechner

**Kurt Mehlhorn**



# Übersicht

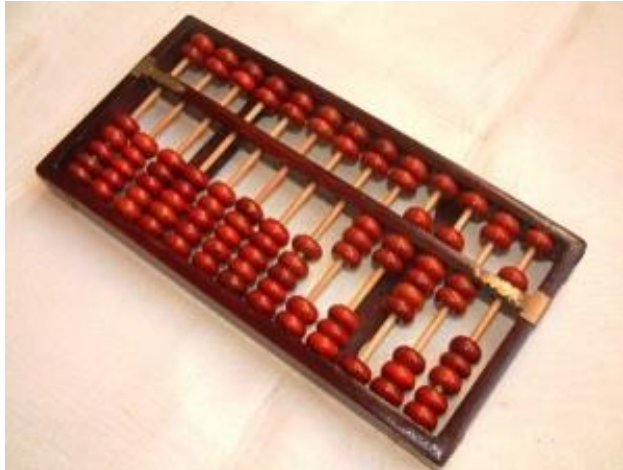
1. Geschichtlicher Rückblick
2. Wie funktionieren Computer?
  - Der Von-Neumann-Rechner
3. Universalität von Rechnern (Basis für Siegeszug der Informatik)
  - Was bedeutet Universalität ?
4. Laufzeit, Rechenzeit
5. Alan Turing
  - Person / Turingmaschine / Turingthese
6. Quantencomputer

# Geschichtlicher Rückblick

- Charles Babbage (1791 – 1871) + Ada Lovelace: Maschine zur Auswertung von Polynomen, Logarithmentafeln, nie fertig
- Alan Turing (1936) entwirft einfachen universellen Rechner als Gedankenexperiment.
- Konrad Zuse (1941): erster funktionierender programmierbarer Rechner
- Mauchly und Presper (43/44) bauen ENIAC
- Grace Hopper (53): erste Programmiersprache (Cobol)



# Frühe “analoge” Rechner



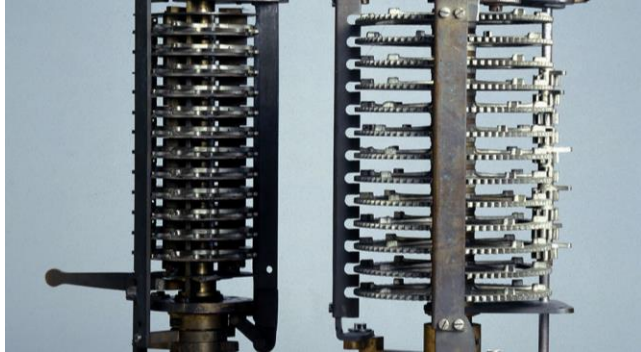
Abacus 2400 BC



Mechanismus von Antikythera 100 BC

Astronomische Uhr

# Programmierbare Rechner



**Charles Babbage (1791-1871), Ada Lovelace (1815-1852)**

„Analytical Machine“

Maschine zur Auswertung von Polynomen, Logarithmentafeln  
nie fertig geworden

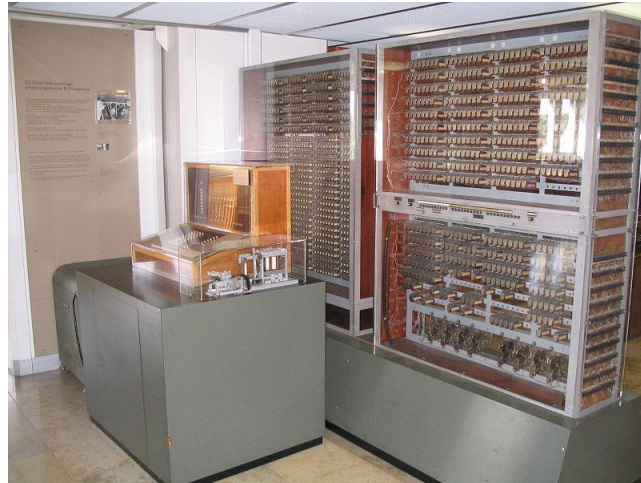


**Konrad Zuse (1910-1995)**

**Z1, 1938**

Mechanisch

# Frühe Computer (Konrad Zuse)



Zuse Z3 (1941)



Zuse Z4 (1942 - 45)

- Z3 und Z4 arbeiten mit Relais (elektromagnetische Schalter)
- Z3 und Z4 sind programmierbar (Programm extern) und universell

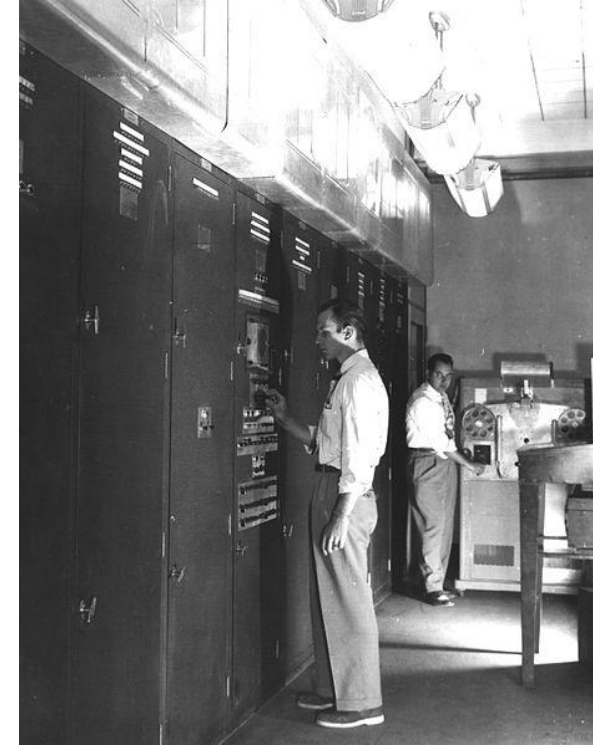
ENIAC (1946)

# EDVAC (Electronic Discrete Variable Automatic Computer)

“First Draft of a Report on the EDVAC”  
by John von Neumann, 1945

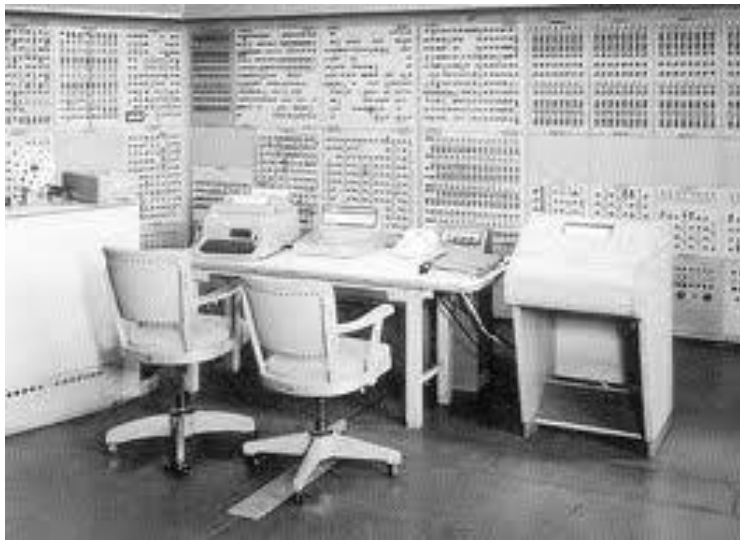
EDVAC, fertiggestellt im Jahr 1951

- Programme im Speicher
- Speicher: 5,5 kilobytes = 44,000 Bits
- Eine Multiplikation in 2,9 Millisekunden
- 6000 Vakuumröhren
- Stromverbrauch 56 kW
- 45,5 m<sup>2</sup> Bodenfläche und 7850 kg Gewicht
- Betriebspersonal 30 Personen für jede 8-Stunden-Schicht
- Kosten: 500000 Dollar (entspricht etwa 7 Millionen in 2018)



**Das Vorbild für alle modernen Rechner**

# Spätere Computer





---

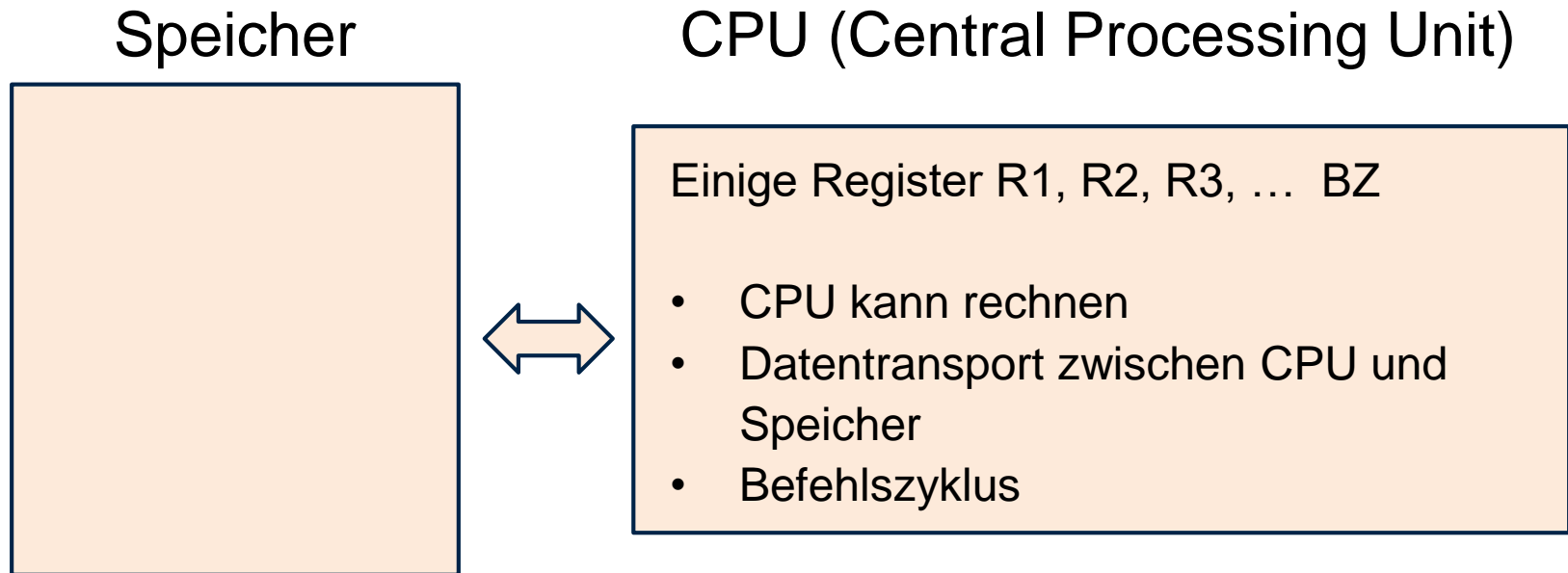
# Aufbau von Rechnern



# Bitstrings und Speicher

- Bitstring = Folge von Nullen und Einsen, z.B. 00110110
- Bitstring kann man interpretieren als
  - Zahl, z. B.  $1010 \equiv 1 \times 8 + 0 \times 4 + 1 \times 2 + 0 \times 1 = 10$
  - Buchstabe, z. B.  $01000000 \equiv @$
  - $01100001 \equiv a$
- Speicher besteht aus Speicherzellen
  - sind nummeriert: 0, 1, 2, 3, 4, 5 ...
  - speichern Bitstring der Länge 64, früher: 32, 16

# Von-Neumann-Rechner



- Rechner = Speicher + CPU
- Speicherzellen in der CPU heißen Register
- CPU kann rechnen (Befehlsumfang = billiger Taschenrechner)
- Speicher enthält Daten und Programm
- Befehlszyklus sorgt für die Ausführung des Programms,  
BZ = Befehlszähler

# Typische Befehle

<b>Transport</b>	$R3 \leftarrow M[5]$ $M[5] \leftarrow R2$	$R1 \leftarrow M[R4]$ $M[R2] \leftarrow R1$
<b>Rechnen</b>	$R1 \leftarrow 0$	$R1 \leftarrow R2 + R3$
<b>Sprung</b>	$BZ \leftarrow 7$	$BZ \leftarrow R1$
<b>Bed. Sprung</b>	if $R1 > 0$ , $BZ \leftarrow n$ , else $BZ \leftarrow BZ + 1$	
<b>Stop</b>	STOP	

# Programme und Befehlszyklus

Programm ist eine Folge von Befehlen

Befehl 1

Befehl 2

Befehl 3

Befehl 4

Befehl 5

Befehl 6

**Befehlszyklus,**

**BZ = Befehlszähler**

1.  $BZ \leftarrow 1$
2. Führe Befehl mit der Nummer BZ aus. Falls STOP, halte an.
3. Erhöhe BZ um eins (außer bei Sprungbefehl, der BZ setzt)
4. Gehe nach 2.

In  $M[1]$  steht eine Zahl  $n \geq 1$ , berechne  $1 + \dots + n$ .

1.  $R1 \leftarrow 0$
2.  $R2 \leftarrow M[1]$
3.  $R1 \leftarrow R1 + R2$
4.  $R2 \leftarrow R2 - 1$
5. IF  $R2 > 0$ ,  $BZ \leftarrow 3$
6. STOP

Ausführung für  $n = 4$

BZ

R1

R2

$M[1]$

Laufzeit =

Am Anfang steht in  $M[1]$  eine natürliche Zahl  $n \geq 1$ .

Wenn das Programm stoppt, dann steht in R1 die Summe  $1 + \dots + n$ .

# Korrektheit des Programs

1.  $R1 \leftarrow 0$
2.  $R2 \leftarrow M[1]$
3.  $R1 \leftarrow R1 + R2$
4.  $R2 \leftarrow R2 - 1$
5. IF  $R2 > 0$ ,  $BZ \leftarrow 3$
6. STOP

Jedes Mal bevor Befehl 3 ausgeführt wird, gilt:

- In R2 steht eine natürliche Zahl  $i$  mit  $n \geq i \geq 1$ .
- In R1 steht  $n + \dots + (i + 1)$

Am Anfang steht in  $M[1]$  eine natürliche Zahl  $n \geq 1$ .

Wenn das Programm stoppt, dann steht in R1 die Summe  $1 + \dots + n$ .

# Höhere Programmiersprachen

1.  $R1 \leftarrow 0$
2.  $R2 \leftarrow M[1]$
3.  $R1 \leftarrow R1 + R2$
4.  $R2 \leftarrow R2 - 1$
5. IF  $R2 > 0$ ,  $BZ \leftarrow 3$
6. STOP

```
sum ← 0;  
i ← n;  
while (i > 0)  
    sum ← sum + i;  
    i ← i - 1;
```

Produktivitätsgewinn

Java, C, C++, Python,

Compiler übersetzen



# Hardware

# Software

- Der Speicher, die CPU (Central Processing Unit), die Peripherie (Bildschirm, Tastatur, Maus, Netzanbindung, ...)
- Führt Befehle aus und realisiert den Befehlszyklus.
- Reagiert auf Peripherie.
- Kauft man im Laden.
- Kann jedes Programm ausführen.

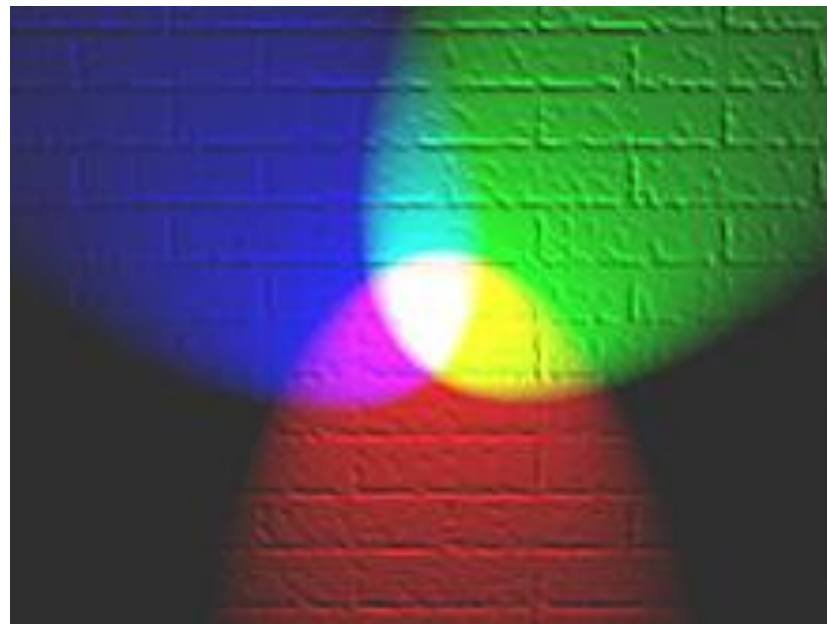
- Die Summe der installierten Programme.
- Programme sind im Speicher abgelegt und werden durch die Hardware ausgeführt.
- Der Fantasie sind kaum Grenzen gesetzt.
- Lädt man aus dem Netz.
- Erstellen von guter Software ist teuer.
- Vervielfältigen ist billig, Grenzkosten  $\approx$  Null

# Kenngrößen und Neuerungen

- Hauptspeicher:  $10^9$  Worte a 64 Bit
- Befehlszyklus:  $10^9$  Befehle pro Sekunde
- Eine Million mal leistungsfähiger (Geschwindigkeit, Speicher, Größe), Tausend mal billiger als 1950
- Neuerungen seit 1950
  - Interrupts (Unterbrechungen), mehrere Programme gleichzeitig
  - Speicherhierarchie: Cache, Main, Disk
  - Bildschirme, Grafik, Maus, Sound, Touch, Mikro
  - Netze
  - Preis und Leistung
  - **Software, Nutzerfreundlichkeit**

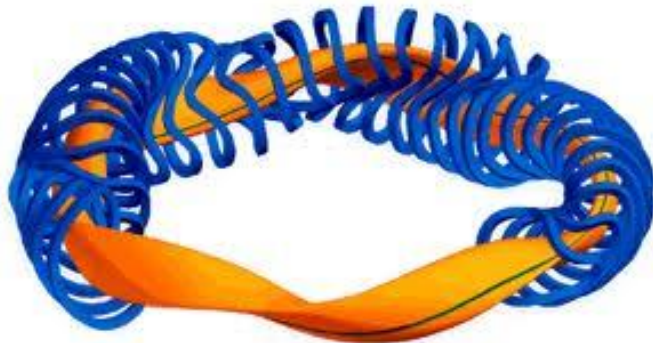
# Bildschirme und Graphik

- Dieser Schirm: 1920 x 1080 Bildpunkte (Pixels)
- Die CPU kann für jedes Pixel Farbe und Helligkeit einstellen



# Supercomputer

- 72 Schränke
- 73000 PowerPCs mit je 2 Gbyte RAM
- Simulation: Physik, Klima, Chemie, Strömung
- 13 Mio Euro



---

# Universalität von Rechnern



# Universalität von Rechnern

- Rechner sind **universell**, d.h., sie können **jedes Programm ausführen**, sofern es nur in ihre Maschinsprache übersetzt ist und es die Ressourcen des Rechners nicht sprengt.
- Universalität von Rechnern ist wesentlich für den Erfolg der Informatik:
  - Das gleiche Programm auf vielen Rechnern.
  - Viele Programme auf einem Rechner.

# Rechner sind universell

- Normale Werkzeuge sind nicht universell: Hammer, Feile, Zange, Auto, ....
- Viele Programme auf einem Rechner: **Ein Smartphone ist Telefon, aktiver Kalender, Fitnessstrainer, Bankterminal, Wetterauskunft, Browser, Suchmaschine, Musik- und Filmspieler, Spielzeug, ...**
- Ein Programm auf vielen Rechnern: **Office läuft auf Rechnern von IBM, Lenovo, Toshiba, Samsung, Apple, ...**

---

# Laufzeit von Programmen

## Effiziente Programme





# Laufzeit von Programmen

- Ausführungszeit in Sekunden
  - $n = 10^8$ , 0.19 sec       $n = 10^9$ , 1.23 sec
- Für theoretische Überlegungen: Anzahl der ausgeführten Befehle/Operationen
- Unser Summenprogramm: Laufzeit =  $3 + 3n = O(n)$
- $O( )$  = Landausymbol für asymptotisches Wachstum: gibt nur den am schnellsten wachsenden Anteil wieder und ignoriert konstante Faktoren

# Effiziente Programme

- Ein Programm heißt **effizient**, wenn seine Laufzeit an Eingaben der Größe  $n$  beschränkt ist durch  $Cn^k$  für Konstanten  $C$  und  $k$ .
- Eingabe der Größe  $n$ :
  - Graph mit  $n$  Knoten und Kanten
  - Zahl mit  $n$  Ziffern
- $P$  = Menge aller Probleme für die es ein effizientes Programm gibt (polynomzeitberechenbar).

# Beispiele

Kein effizientes Programm bekannt:

- Primfaktorzerlegung
- Problem des Handlungsreisenden
- 3-Färbung von Graphen
- Erfüllbarkeitsproblem der Aussagenlogik

Effizientes Programm bekannt:

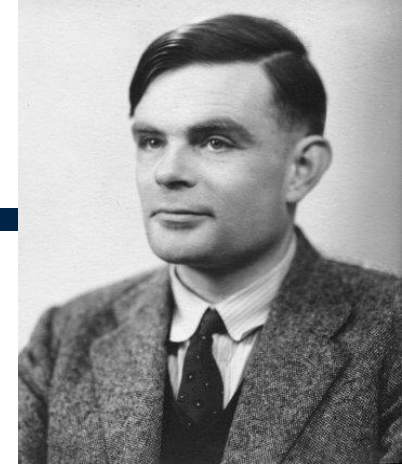
- Sortieren
  - Suchen
  - Kürzeste Wege
  - Multiplikation von Zahlen
  - Lösen von linearen Gleichungen
  - Test, ob eine Zahl Primzahl ist.
- Obige Probleme sind in P.

---

# Alan Turing und die Turingmaschine

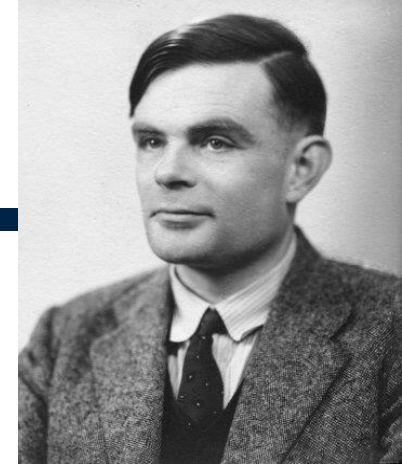


# Alan Turing (1912 – 1952)



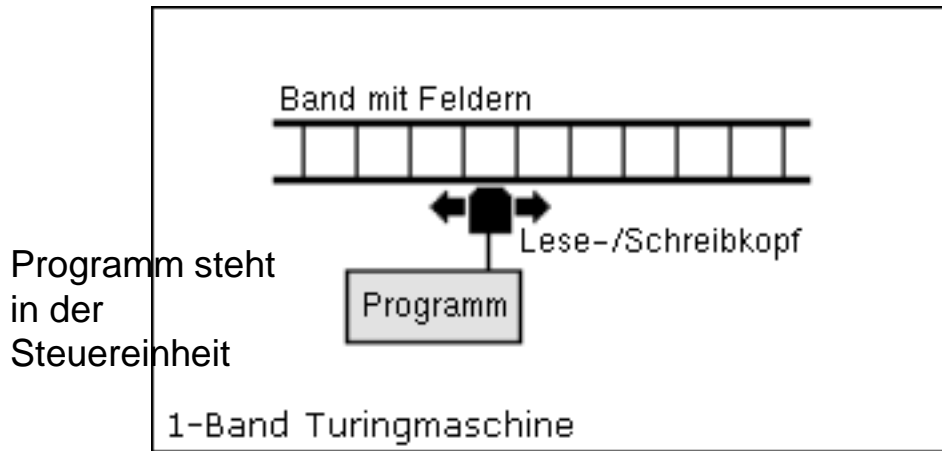
- Britischer Mathematiker
- „On Computable Numbers, with an Application to the Entscheidungsproblem“, 1936
- David Hilbert (1928): kann Mathematik mechanisiert werden?
- Kurt Goedel (1931): NEIN (Parallelen zum „Lügner Paradox“: Dieser Satz ist falsch.)
- Turings Arbeit von 1936 vereinfacht den Beweis wesentlich und führt Turingmaschine ein.

# Alan Turing (1912 – 1952)



- Kryptanalyse: Churchill: „Alan Turing made the single biggest contribution to Allied victory in the war against Nazi Germany.“
- Muster in der Natur, Fell eines Zebras
- Sehr guter Sportler, Marathon in 2:46 (Olympiasieger in 1948, Zeit 2:35).
- Verurteilung wegen Homosexualität in 1952, chemische Kastration.
- Selbstmord durch Blausäure in 1954.

# Die Turingmaschine



Auf jedem Bandquadrat steht ein Buchstabe (Symbol, Zeichen) in A, ..., Z, a, ..., z, 0, ..., 9, \$, §, ..., leer

**Endliches** Alphabet

Steuereinheit befindet sich in einem von **endlich** vielen Zuständen  $p, q, q_0, q_1, q_2, \dots$

Turingbefehl	Zustand	Zeichen	neuer Zustand	neues Zeichen	Bewegung
	$q_1$	a	$q_2$	b	R

Wenn du im Zustand  $q_1$  ein a liest, dann gehe in den Zustand  $q_2$  über, drucke ein b und bewege den Kopf nach rechts

Turingprogramm = Menge (Folge) von Turingbefehlen, je zwei unterscheiden sich in den ersten Spalten

# Beispiel einer Turingmaschine

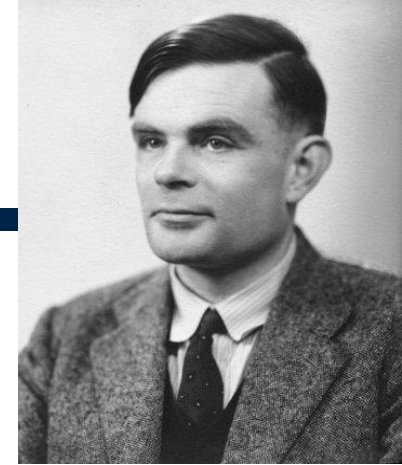
Turingbefehl	Zustand	Zeichen	neuer Zustand	neues Zeichen	Bewegung
	q1	a	q2	b	R

Am Anfang sei das Band mit ....00000BBBBBB..... beschriftet, der Kopf stehe auf der rechtesten Null und die Maschine sei in Zustand q1

q1 0 q2 1 S  
q1 1 q1 0 L  
q2 0 q2 0 R  
q2 1 q2 1 R  
q2 B q1 B L



# Turing-These (1936)



- Turing hat TM im Jahr 1936 eingeführt.
- Turing-These: **Turing Maschine fasst den Begriff „nach Regeln berechenbar“.**
- 4 Argumente
  - Menschliche Rechner, siehe nächste Folie
  - Beispiele: Zählen, Dezimaldarstellung von  $\pi$
  - Universelle Turingmaschine
  - Äquivalenz zur Formalisierung von Church
- These ist allgemein akzeptiert

# AEG Rechnerraum (1920)



# Zusammenfassung

- Rechner sind (im Prinzip) recht einfach aufgebaut: Recheneinheit und Speicher
- Befehlzyklus: wiederhole bis STOP-Befehl
  - Führe Befehl aus und erhöhe Befehlszähler um 1
  - Bei Sprungbefehl setze BZ auf die genannte Adresse
- Übersetzung von höherer Programmiersprache in Maschinensprache erfolgt maschinell
- Rechner sind universell: viele Programme auf einem Rechner, das gleiche Programm auf vielen Rechnern
- Zukunft??: Quantenrechner