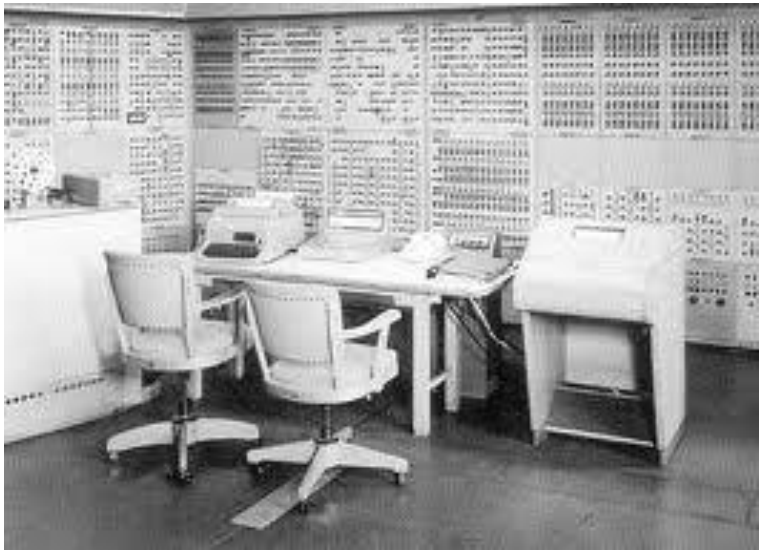
max planck institut
informatik

Ideen der Informatik

Was ist ein Computer?
Was ist ein Programm?
Können Computer Alles?

Kurt Mehlhorn und Adrian Neumann

Was ist ein Computer?



Übersicht

- Was ist ein Computer, ein Programm?
  - Turings Antwort von 1936
  - Moderne Rechner sehen anders aus, sie können aber nicht mehr.
- Universalität von Rechnern: Basis für Siegeszug der Informatik
- Können Computer alles? Schon Turing zeigte: die Antwort ist NEIN
- Die Person Alan Turing

Aufgaben

- Wann hat Alan Turing sein Modell eines Computers vorgestellt?
- Beispiele für Nichtuniversalität.
- In welchen Sinn sind Computer universell?
- Nenne Aufgaben, von denen vor 50 Jahren nur wenige geglaubt haben, dass Computer sie je könnten, dies aber inzwischen der Fall ist.

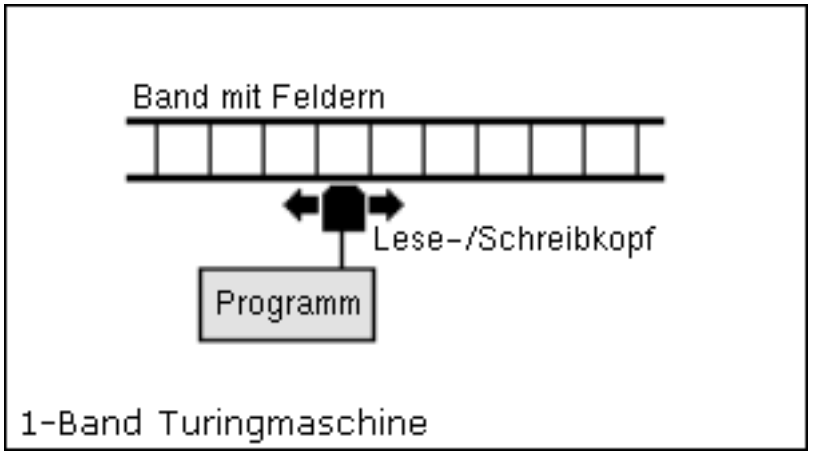
max planck institut
informatik

Was ist ein Computer?

Was ist ein Programm?

Die Antwort von Alan Turing in 1936

Die Turing Maschine



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

Endliches Alphabet

Steuereinheit ist in einem von **endlichen** vielen Zuständen p, q, q0,q1,q2,….

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

Wenn du im Zustand q1 ein a liest, dann gehe in den Zustand q2 über, drucke ein b und bewege den Kopf nach rechts

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

Erstes Beispiel

- Befehle q 0 q 1 L und q 1 q 0 L
- Nur ein Zustand, nämlich q
- Alphabet: 0 und 1
- Rechnung an Eingabe 0010

Berechnung

- Anfangszustand: Zustand = q0,
- Eingabe: ein Wort w
- Kopf steht auf rechtestem Zeichen von w, alles anderen Quadrate sind leer
- Maschine führt Befehle aus, solange einer anwendbar ist.
- Ergebnis: Bandinhalt, wenn Maschine anhält
- Rechnung kann unendlich lang sein

Ein Programm, das zählt

p	0	q	1	S
p	leer	q	1	S
p	1	p	0	L
q	0	q	0	R
q	1	q	1	R
q	leer	p	leer	L

Turing These

Alles was man je mit einer Maschine berechnen kann, kann man auch mit einer Turingmaschine berechnen.

TM fassen den Begriff: berechenbar mit einer Maschine.

- Welche Gründe hatte er für diese kühne Behauptung?
- Hat er Recht?
- Die Person Alan Turing

Aufgabe

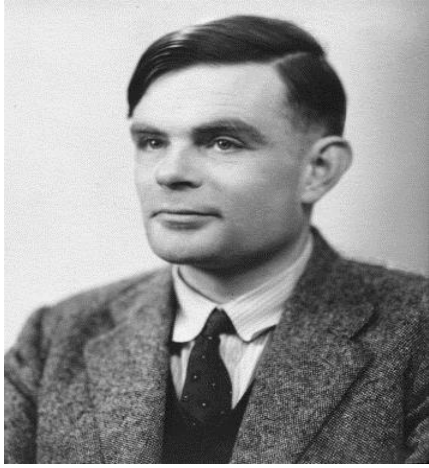
- Was macht folgende Maschine?
- Nur ein Zustand, nämlich q
- Bandalphabet 0, 1, und B (für blank (leer))
- Befehle q 0 q 1 L, q 1 q 1 L, q B q 1 L
- Anfangskonfiguration

max planck institut
informatik

Die Person Alan Turing

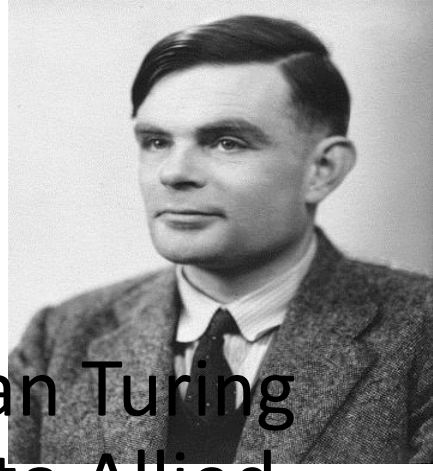
Turing These: Turingmaschinen fassen den Begriff "berechenbar mit Hilfe einer Maschine„ alles was mit irgendeiner Maschine berechnet werden kann, kann auch mit einer TM berechnet werden.

Alan Turing (1912 – 1952)



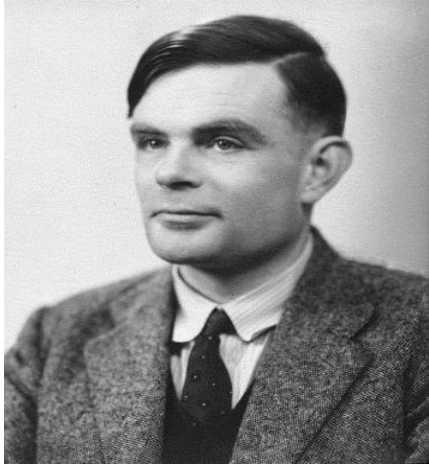
- Britischer Mathematiker
- On Computable Numbers, with an Application to the Entscheidungsproblem
- David Hilbert (1928): kann Mathematik mechanisiert werden?
- Kurt Goedel (1931): NEIN
- Turings Arbeit von 36 vereinfacht den Beweis ganz wesentlich und führt TM ein

Alan Turing (1912 – 1952)



- Kryptanalyse: Churchill stated that Alan Turing made the single biggest contribution to Allied victory in the war against Nazi Germany
- Muster in der Natur
- Sehr guter Sportler, Marathon in 2:46 (Olympiasieger in 48, 2:35)
- Verurteilung wegen Homosexualität in 52, chemische Kastration
- Selbstmord durch Blausäure in 54

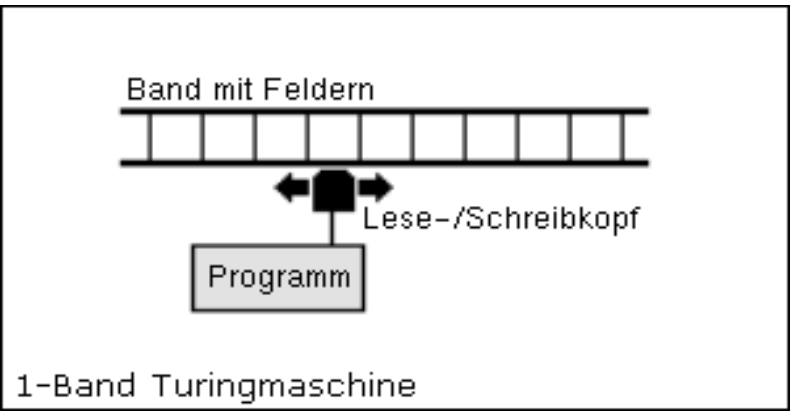
Turingthese (1936)



- Turing hat TM in 1936 eingeführt
- 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

Universelle Turingmaschine

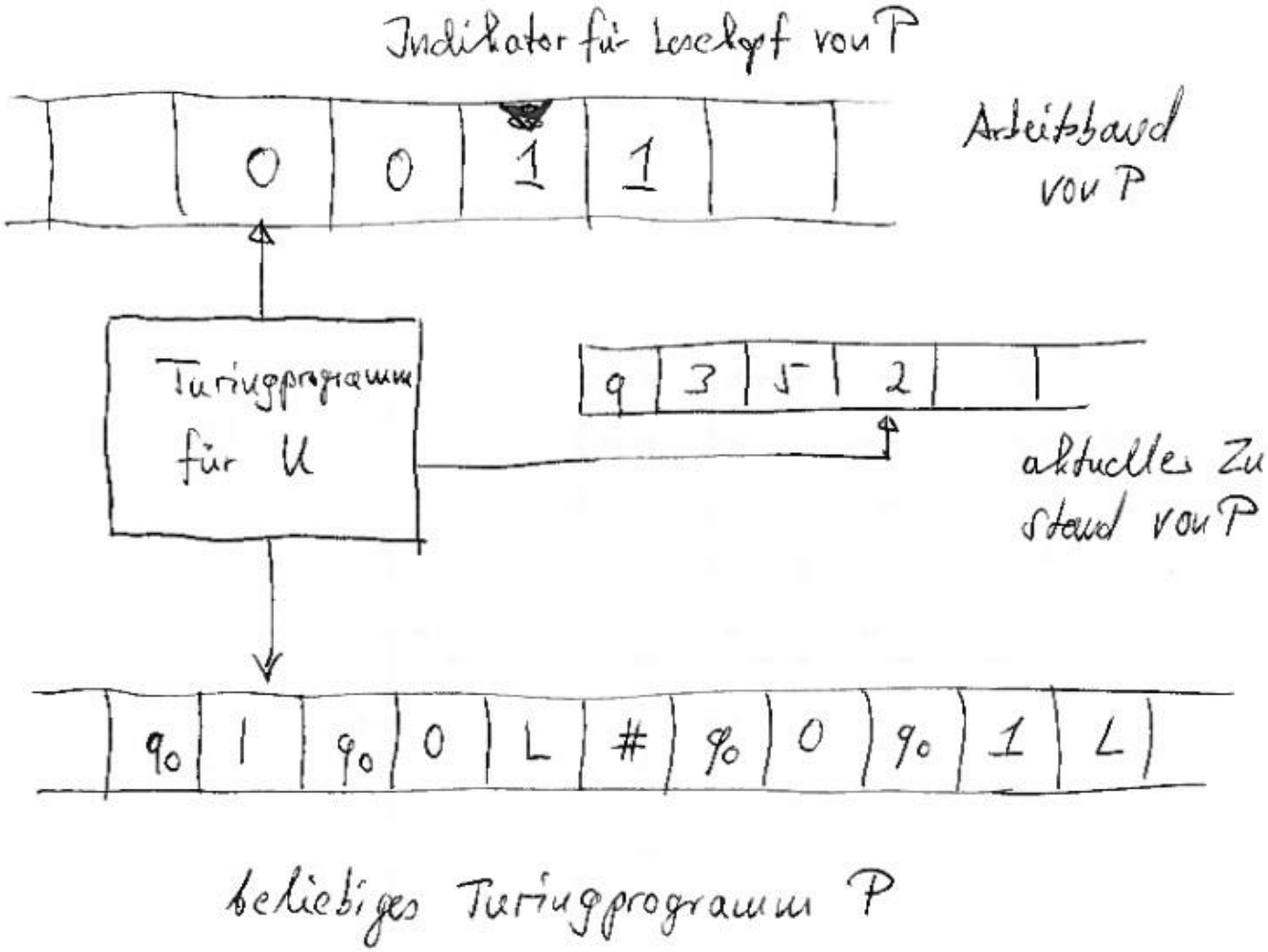


Steuereinheit enthält Programm; für jedes Programm eine eigene Maschine

Beim Ausführen unserer Beispielprogramme sind wir mechanisch vorgegangen: **Programm U**

- Aktueller Zustand und aktuelle Zeichen
- Bestimmung des anwendbaren Befehls
- Ausführung des Befehls, neuer Zustand, neuer Zeichen, Bewegung des Kopfes

Universelles U simuliert Programm P



Universelle Turingmaschine

Eingabe: ein Programm P und eine Eingabe w für P

Eigenschaft: tut, was immer P mit Eingabe w tut

Diese Maschine kann jedes Programm ausführen, sie ist universell. Wenn man universelle Maschinen baut, dann muss man sich nicht darum kümmern, was sie nachher tun sollen.

PCs, Smartphones, … sind universelle Maschinen

Nichtuniverseller Computer: Steuerung ihrer Waschmaschine

Universelle Maschine

Normale Werkzeuge sind nicht universell: Hammer, Feile, Zange, ….

Mein Smartphone ist Telefon, aktiver Kalender, Fitnesstrainer, Bankterminal, Browser, Wetterauskunft, Browser, Suchmaschine, Musikspieler, Spielzeug, …

Universalität von Rechnern ist wesentlich für Siegeszug der Informatik

Aufgaben

- Was ist mit Universalität von Rechnern gemeint?
- Wieso ist sie so wichtig für den Erfolg der Informatik
- Wie lautet Turings These?
- Mit welchen vier Gründen hat er die These gestützt?

max planck institut
informatik

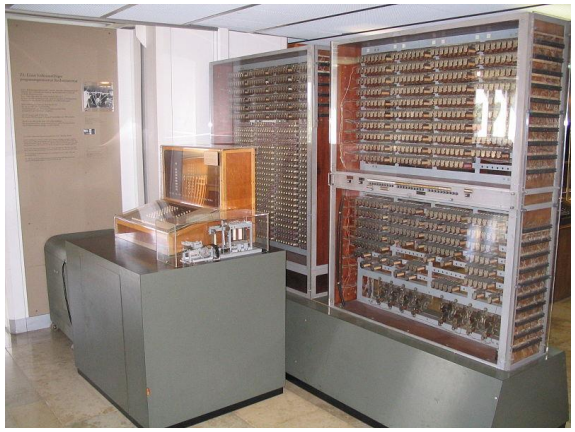
Moderne Rechner

- Von Neumann, 1950
- Heute

Ein Programm für einen modernen Rechner

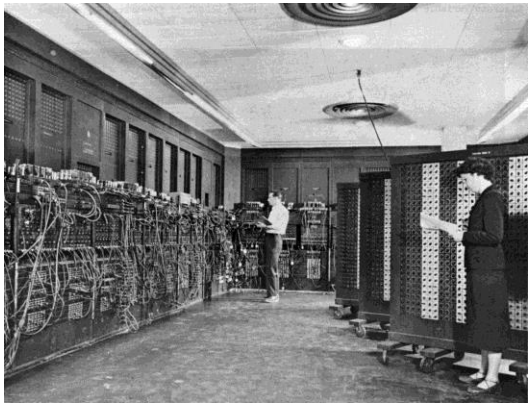
Frühe Computer

Zuse Z1 (1937)



Z3 (1941)

Z4 (1945)

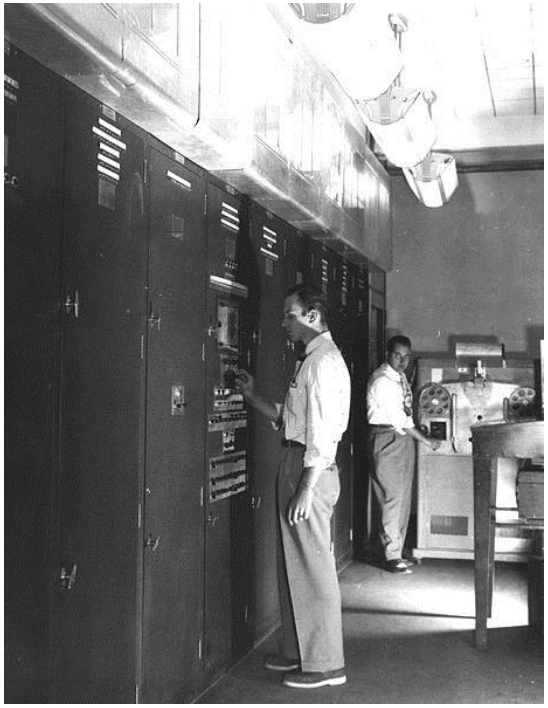


ENIAC (1946)

Z3, Z4 und ENIAC sind programmierbar (Programm extern) und Turingmächtig

Z3 und Z4 arbeiten mit Relais, ENIAC arbeitet mir Röhren

EDVAC (Electronic …. Computer)



First Draft of a Report on the EDVAC
by John von Neumann,
June 30, 1945

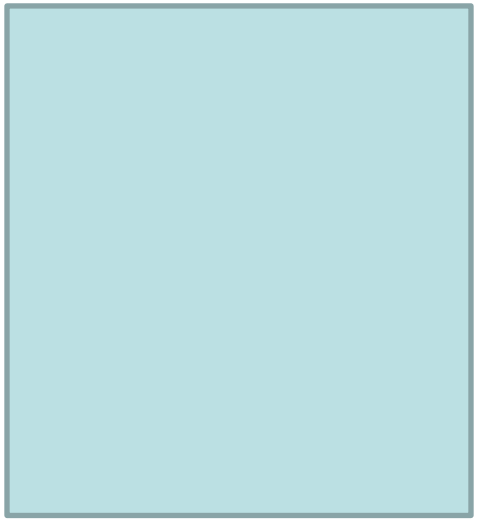
EDVAC, fertiggestellt in 1951

- Stored program
- Speicher, 5.5 kilobytes
- multiplication time 2.9 milliseconds
- 6,000 vacuum tubes
- consumed 56 kW of power
- 45.5 m^2 of floor space and weighed 17,300 lb (7,850 kg)
- operating personnel was thirty people for each eight-hour shift
- Kosten 500,000 Dollar (entspricht etwa 6 Millionen in 2010)

Das Vorbild für alle modernen Rechner

Von Neumann Rechner

Speicher



Speicherzellen
sind nummeriert:
0,1,2,3,4
Kann Bitstring der Länge
64 speichern

CPU (Central Processing Unit)

Akkumulator (ACC)

Befehlszähler

Im ACC wird gerechnet

- Speicher enthält Daten und Programm
- Befehlszyklus
  1. Führe Befehl mit Nummer BZ aus
  2. Erhöhe BZ um eins
  3. Gehe nach 1.

Typische Befehle

- ACC ⇦ 0
- ACC ⇦ M[n]
- M[n] ⇦ ACC
- ACC ⇦ ACC + M[n] auch -, x, /
- ACC ⇦ ACC - 1
- if ACC > 0, BZ ⇦ n, else BZ ⇦ BZ + 1
- STOP

In M[1] steht eine Zahl n, bilde 1 + … + n

1. ACC ⇦ 0
2. M[2] ← ACC
3. ACC ← M[2]
4. ACC ← ACC + M[1]
5. M[2] ← ACC
6. ACC ← M[1]
7. ACC ← ACC - 1
8. M[1] ← ACC
9. IF ACC > 0, BZ ← 3
10. STOP

Ausführung für n = 4

BZ

ACC

M[1]

M[2]

Turing Maschine vs. Von Neumann

- Nur Zeichen unter Kopf kann gelesen werden
- Zelle enthält ein Zeichen
- Befehle sind primitiv, ersetze ein Zeichen durch ein anderes und wechsle Zustand
- Auswahl Befehl durch Zustand und Zeichen

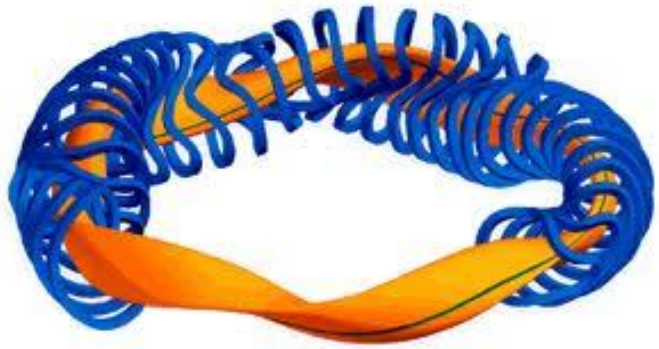
- Speicherzellen sind adressierbar
- Zellen enthalten 32 Bits
- Umfangreicher Befehlssatz, insbesondere Arithmetik mit Zahlen aus 32 Bits
- Auswahl Befehl durch Befehlszähler

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, Graphik, Maus, Sound, Touch, Mikro
  - Netze
  - Preis und Leistung
  - **Software, Nutzerfreundlichkeit**

Supercompute

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



Bildschirme und Graphik

- Dieser Schirm: 1366 x 768 Pixels
- Für die CPU: für jedes der Pixel kann man Farbe und Helligkeit einstellen
- Graphikkarte zeigt die Werte auf dem Bildschirm an.

Aufgaben

- Was sind die wesentlichen Unterschiede zwischen Turing Maschine und von Neumann Rechner?
- Wie funktioniert ein Graphikbildschirm?
- Ändern sie das Programm so ab, dass es die Summe n + n-2 + n-4 + … + 2 (1) berechnet. Der letzte Summand ist 2 falls n gerade ist und 1 falls n ungerade.

Hardware --- Software

Höhere Programmiersprachen

Laufzeit von Programmen

Hardware

- Der Speicher, die CPU (central processing unit), die Peripherie (Bildschirm, Tastatur, Maus, Netzanbindung, …)
- Führt Befehle aus und realisiert den Befehls-zyklus
- Reagiert auf Peripherie
- Kauft man im Laden
- Kann jedes Programm ausführen

Software

- 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

Höhere Programmiersprachen

1. ACC ⇦ 0
2. M[2] ← ACC
3. ACC ← M[2]
4. ACC ← ACC + M[1]
5. M[2] ← ACC
6. ACC ← M[1]
7. ACC ← ACC - 1
8. M[1] ← ACC
9. IF ACC > 0, BZ ← 3
10. STOP

```
int sum = 0;
int i = n;
while ( i > 0 )
    sum = sum + i;
    i = i – 1;
```

Produktivitätsgewinn

Java, C, C++, Python,

max planck institut informatik

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
- Hier: 3 + 7n = O(n)
- O() = Landausymbol für asymptotisches Wachstum: gibt nur den am schnellsten wachsenden Anteil wieder und ignoriert konstante Faktoren

max planck institut
informatik

Korrektheit von Programmen

Korrektheit von Programmen

- Wir haben uns überzeugt, dass das Programm für n = 4 das richtige Ergebnis liefert
- Auch für beliebige n?
- Dazu muss man beweisen:

  Für beliebiges n gilt: das Programm hält dem Wert 1 + … + n in M[2]

In M[1] steht eine Zahl n, bilde 1 + … + n

1. ACC ⇦ 0
2. M[2] ← ACC
3. ACC ← M[2]
4. ACC ← ACC + M[1]
5. M[2] ← ACC
6. ACC ← M[1]
7. ACC ← ACC - 1
8. M[1] ← ACC
9. IF ACC > 0, BZ ← 3
10. STOP

Vor Ausführung von 3 gilt:
in M[1] steht eine Zahl i mit $1 \leq i \leq n$ und in M[2] steht $i + 1 + \ldots + n$.

max planck institut
informatik

Können Computer alles?

Nein

Können Computer alles?

NEIN

Turing: das Halteproblem kann nicht durch eine Maschine gelöst werden

Halteproblem

- Gibt es das folgende Turingprogramm?
  - Eingabe P$w, P = Turingprogramm, w = Wort ohne $
  - Eigenschaften
    - Hält immer
    - Hält im Zustand q1, falls P mit Eingabe w anhält
    - Hält im Zustand q2, falls P mit Eingabe w nicht anhält
    - Falls die Eingabe nicht die richtige Form hat, dann darf das Programm irgendwas tun

Turing: ein solches Programm gibt es nicht

Beweis

- Annahme, Programm existiert, nenne es H
- Dann gibt es auch das folgende Programm Q
  - Eingabe, Turingprogramm P
  - Verhalten:
    - falls H mit Eingabe P$P in q1 hält, dann läuft Q immer weiter
    - Falls H mit Eingabe P$P in q2 hält, dann hält Q an
- Was macht Q an der Eingabe Q? Hält es an oder läuft es für immer?

Turings große Leistungen

- Präzise Fassung des Begriffs berechenbar durch die Definition der Turingmaschine
- Unentscheidbarkeit des Halteproblems
- Konstruktion der universellen Turingmaschine