

Rechnerarchitektur

Frank Slanck

Organisationsökologie

Vorlesung: Frank Stomke

frank.stomke@uni-ulm.de

027 / 3
Nord

Übung: Felix Heintmann

felix.heintmann@uni-ulm.de

Tutorien

Klausur

Projekt: Marcel Ripß

Praktiken

Labo

10 Versuche
z Nachholtermin

Alle Versuche sind
Voraussetzung für die Klausur

Was ist Rechnerarchitektur?

(Architektura: lat. Baukunst, aus gr. griechisch)

Bau-Typologien:

Romanisch

Gotisch

Baroque

In der Vorlesung geht es um die Hardware des Computers

Es soll aber von der schaltungstechnischen Realisierung abstrahiert werden.

1. Konzept Rechnerarchitektur

Universal, programmierbarer Rechner

Von Automaten zur Turingmaschine

Registermaschine

Das von Neumann-Prinzip

Rechnerarchitektur (Von Neumann)

Rechnerstrukturen oder Mikroarchitektur (Struktur)

Der Einfluss von Technologie auf die Architektur

2. Architektur und Mikroarchitektur

Verhalten vs. Struktur:

Komponenten der Rechnerarchitektur

Einführung und Verhalten

Befehlszypern

Operandenadressierung

Speicheradressierung

Mikroprogrammierung

3. Die Rechengeschwindigkeit : Performance

Der Befehlsteil

Rechenleistung

Fließbandverarbeitung

CISC vs RISC

4. Speicherarchitektur

Was Speicherarchitektur?

Die Harvardarchitektur

Der Adressraum

Die Leistung (Performance des Speichers)

Architektur vs. Technologie

Caches:

Struktur

Architektur

Zugriffsprotokolle

Speicherbaudiszipline

Die Speicherkette

Zugriffsstufen, Banks und Verschränkung

Speicherzugriff (Memory Controller)

Speicherverwaltung

(Schnittstelle zu Betriebssystem)

Overlays

Segmentverwaltung

Seitenverwaltung

Speicherverwaltungshardware (Memory Controller)

5. Beispiele RISC-Prozessoren

Befehlssatz MIPS

Befehlswort

Arithmetisch-Logische Befehle

Transferebefehle

Verzweigungen und Sprünge

Befehlssatz RISC

Befehlswort

Arithmetisch-Logische Befehle

Transferebefehle

Verzweigungen und Sprünge

Ableitung des Mikroarchitektur aus dem Befehlssatz des RISC V

6. Assemblersprachen

Weiterungen zur Programmierung

Von der Hochsprache zur Maschine

Entscheidungen (IF - CASE)

Schleifen (FOR - WHILE)

Listen (ARRAY) und Strukturen (STRUCT)

Unterprogramme (Prozeduren und Funktionen)

Rekurrenz und : Die Fibonacci-Zahlen

Werkzeuge um transzendente Funktionen

7. Hemmisse in der Fließbandverarbeitung

Datenabhängigkeiten

Echte Datenabhängigkeiten (Read After Write)

Ausgabeabhängigkeiten (Write After Write)

Gegendependenzen (Write After Read)

Vermeidung von Datenabhängigkeiten

Kontrollflussabhängigkeiten

Vorläufiger Sprung (Branch Delay Slot)

Statische Sprungvorhersage (Branch Prediction)

Dynamische Sprungvorhersage

Hochdynamische Sprungvorhersage

8. Fahrer- und superhalbe Architektur

Fahrerarchiv (VL11W) (SIMD)

Superhalbe Architektur

Tomograph (Reservation Stations)

Schedule und Reorderbuffer

Branch- & Value-Prediction

Milieuprogramme

9. Multithreaded und Mehrkernarchitekturen

Hart (Hardwarethread)

Spezifikationskonistent

Multicore und Cache koherent

10. Ausnahmebehandlung, Multicore und GPU



Konzept

Rechnerarchitektur

Konzept Rechnerarchitektur

Universal, programmierbarer Rechner

Von Automaten zu Turingmaschine

Rechnerarchitektur

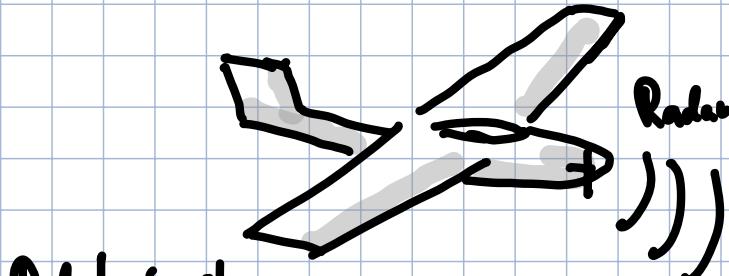
Das von Neumann-Prinzip

Rechnerarchitektur (Vierfach)

Rechnerstruktur oder Mikroarchitektur (Struktur)

Der Einfluss von Technologie auf die Architektur

Rechentechnik: Voraussetzungen



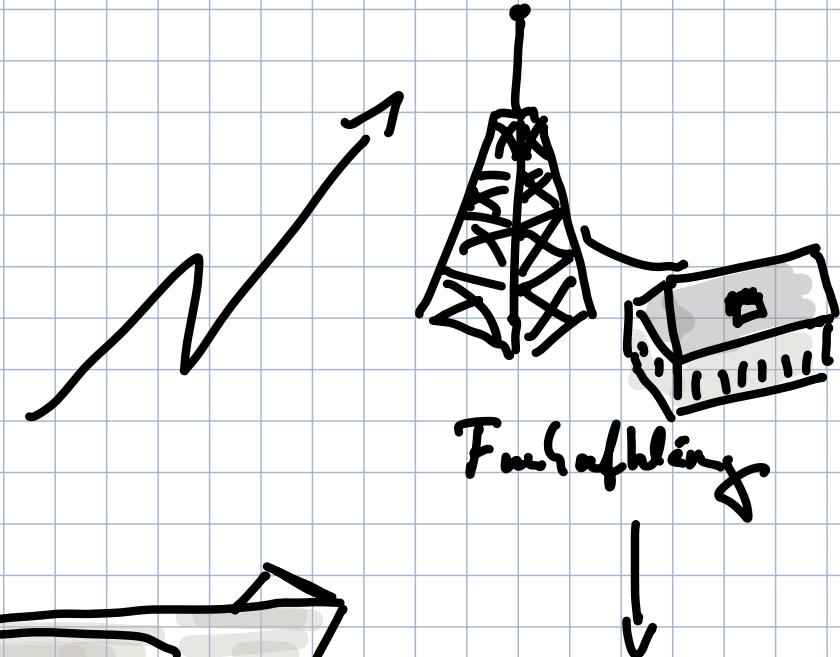
Pulsbreite:



Serielle Kodierung der Zeichen
in frühen Rechnern

Hochfrequenz, Impulse

Problem: Synchronisations Schaltungen
(Flip-Flop)



|00000|

Turing Bomber

Elektromechanische
Kombination

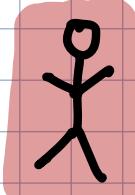
Rechner

Turing

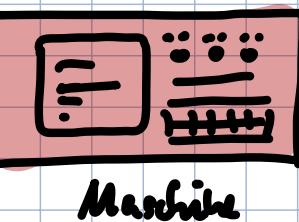
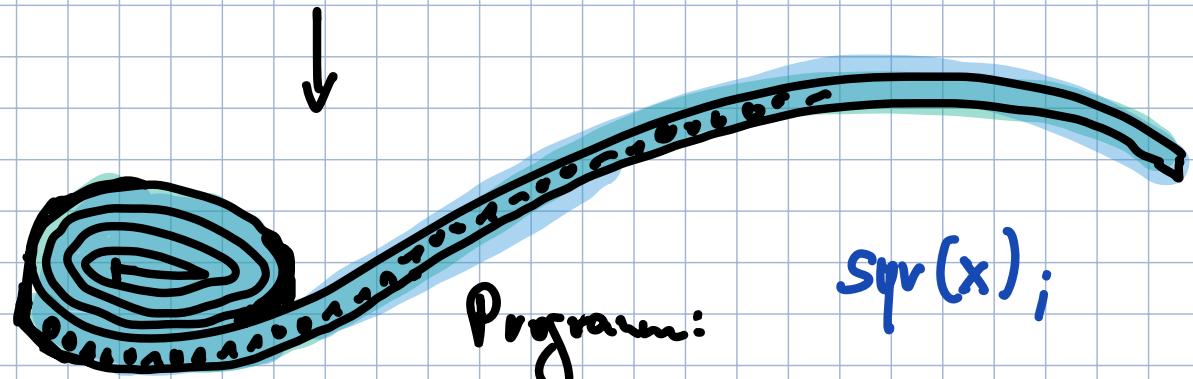
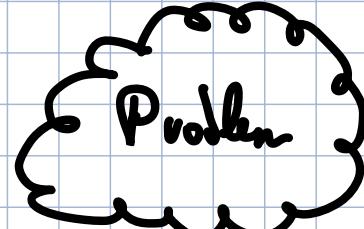
Automatisierung des Rechnens

Informatik
Information - Automatisierung

Computer:



al-Charit mi
Handlungsanweisung zur
Lösung eines Problems



Maschine

Beispiel:

Wurzelziehen

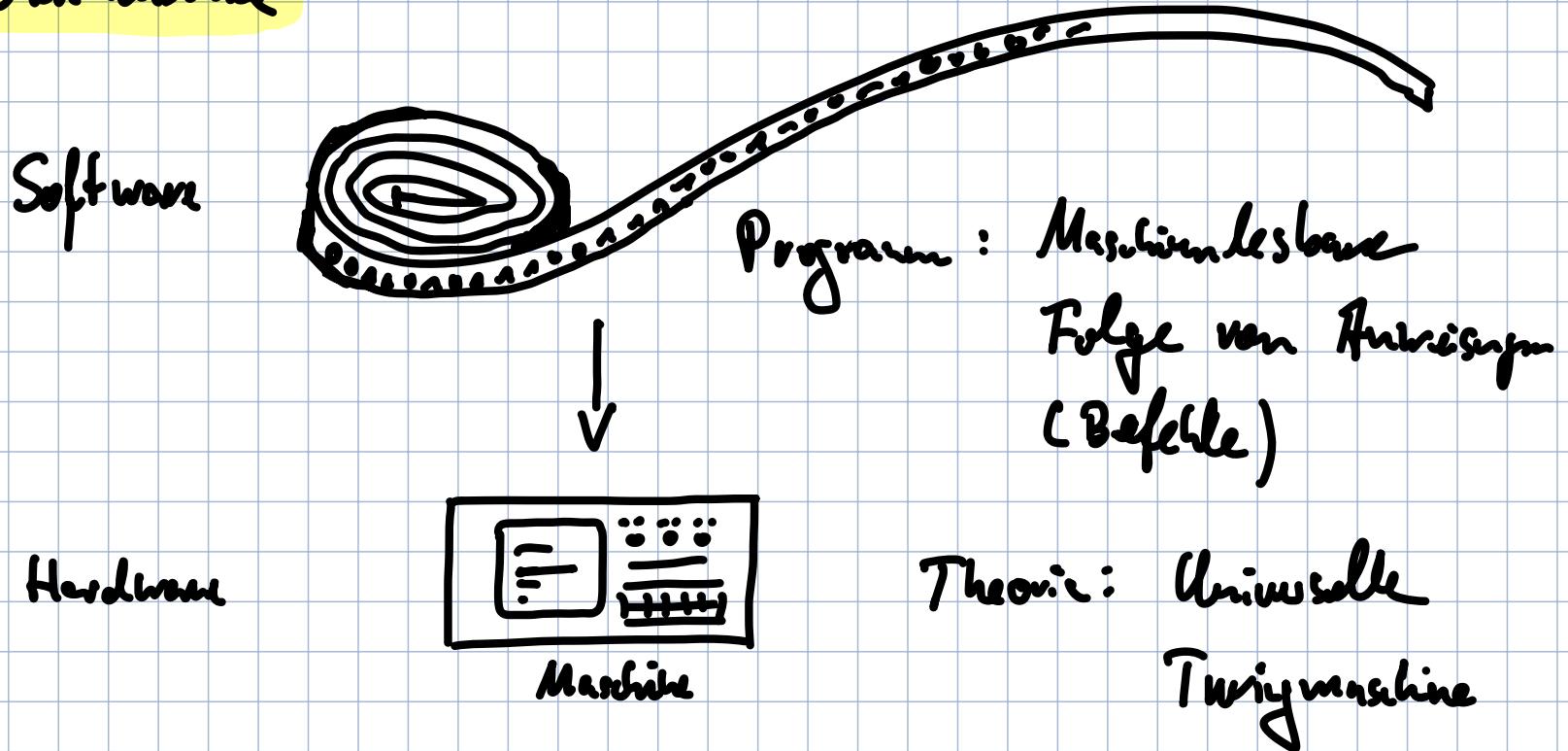
Newton -
Raphson - Verfahren

$\text{Sqr}(x)$:

Folge von Anweisungen

(gv. Programme:
Vorchrift, Vorgeordnetes)

Universelle Rechenmaschine



Maschine ist universell!

Die Maschine ist in der Lage jedes beliebige Problem (Algorithmus) zu lösen, da sie programmiert werden kann.



Die Theorie: Konzepte

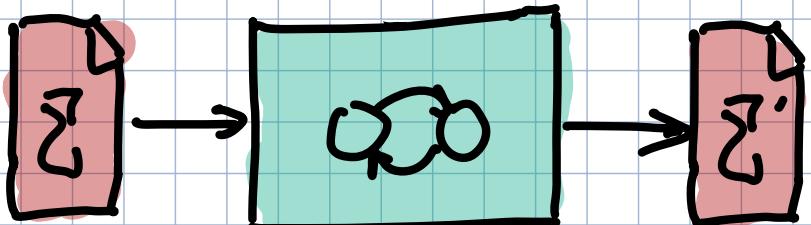
Der endliche Automat

$$\Sigma \in \mathbb{E}$$

$$S := \{s_0, \dots, s_n\}$$

$$\delta := S \times \Sigma \rightarrow (S, \Sigma')$$

Eingabealphabet
Transitionsmenge
Ausgangszustand

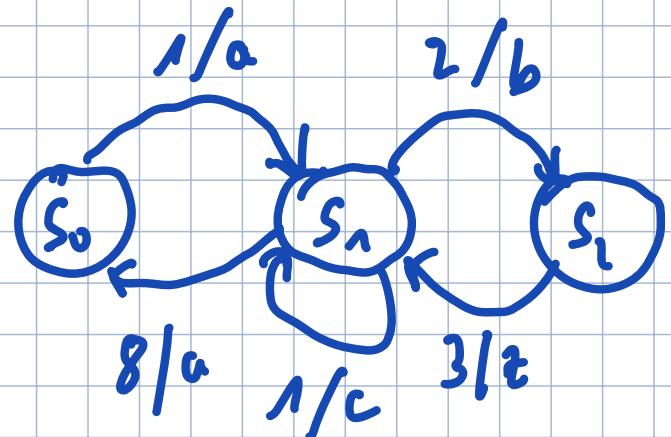


Eingabe

Ablauf

Ausgabe

Beispiel:



$$\Sigma \in \mathbb{N} \cup \{a, b, c, \dots, z\}$$

$$S := \{s_0, s_1, s_2\}$$

$$\delta := \begin{pmatrix} s_0 & 1 & a & s_1 \\ s_1 & 2 & b & s_2 \\ s_2 & 3 & c & s_0 \\ s_0 & 8 & a & s_0 \\ s_1 & 1 & c & s_1 \\ s_2 & 2 & b & s_2 \end{pmatrix}$$

Der endliche Automat: Technisch

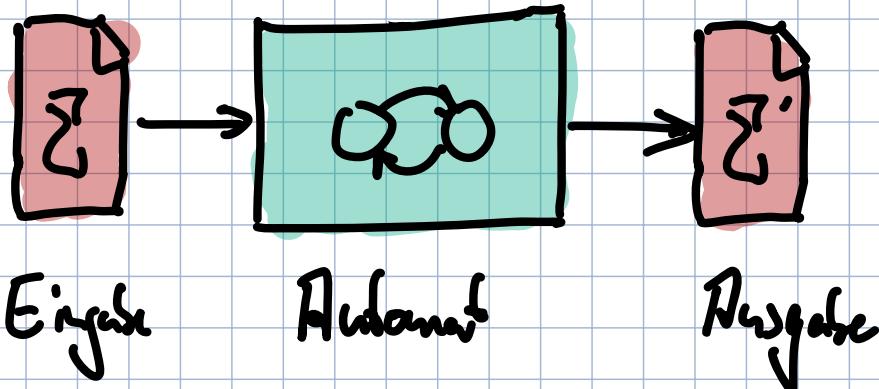
$$\Sigma \in \mathbb{E}$$

$$S := \{\delta_0, \dots, \delta_n\}$$

$$\delta := S \times \Sigma \rightarrow (S, \Sigma')$$

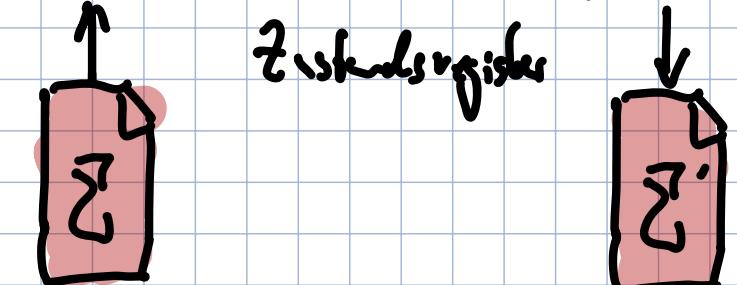
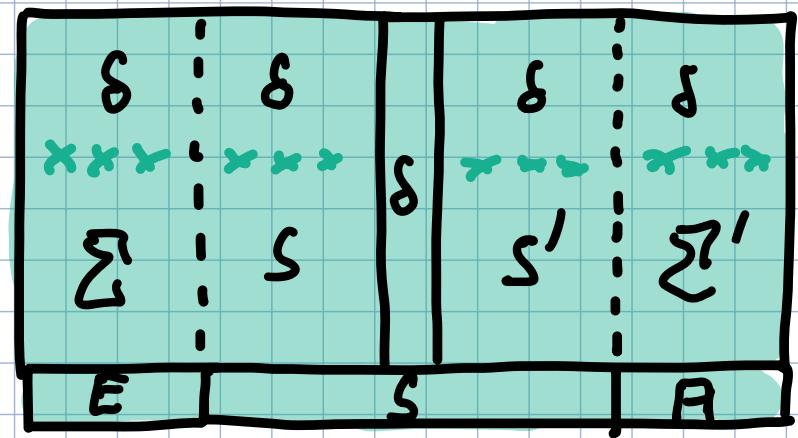
Eingabealphabet
Zustandsmenge

Zustandsübergangsfonction

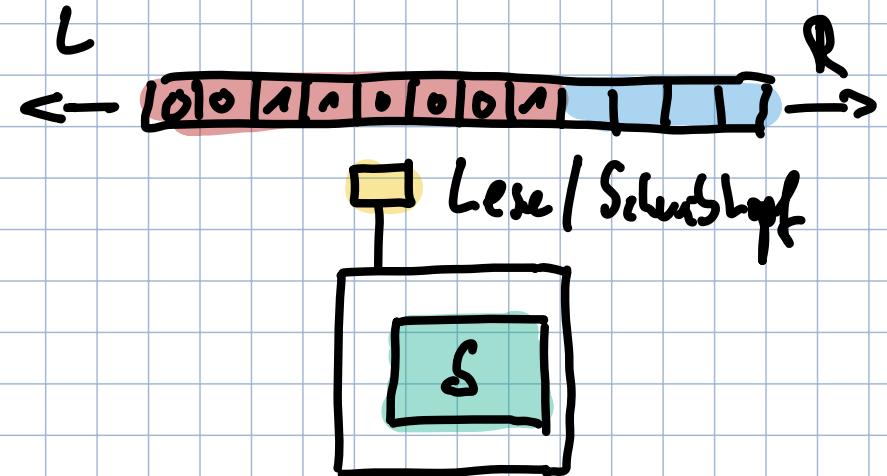
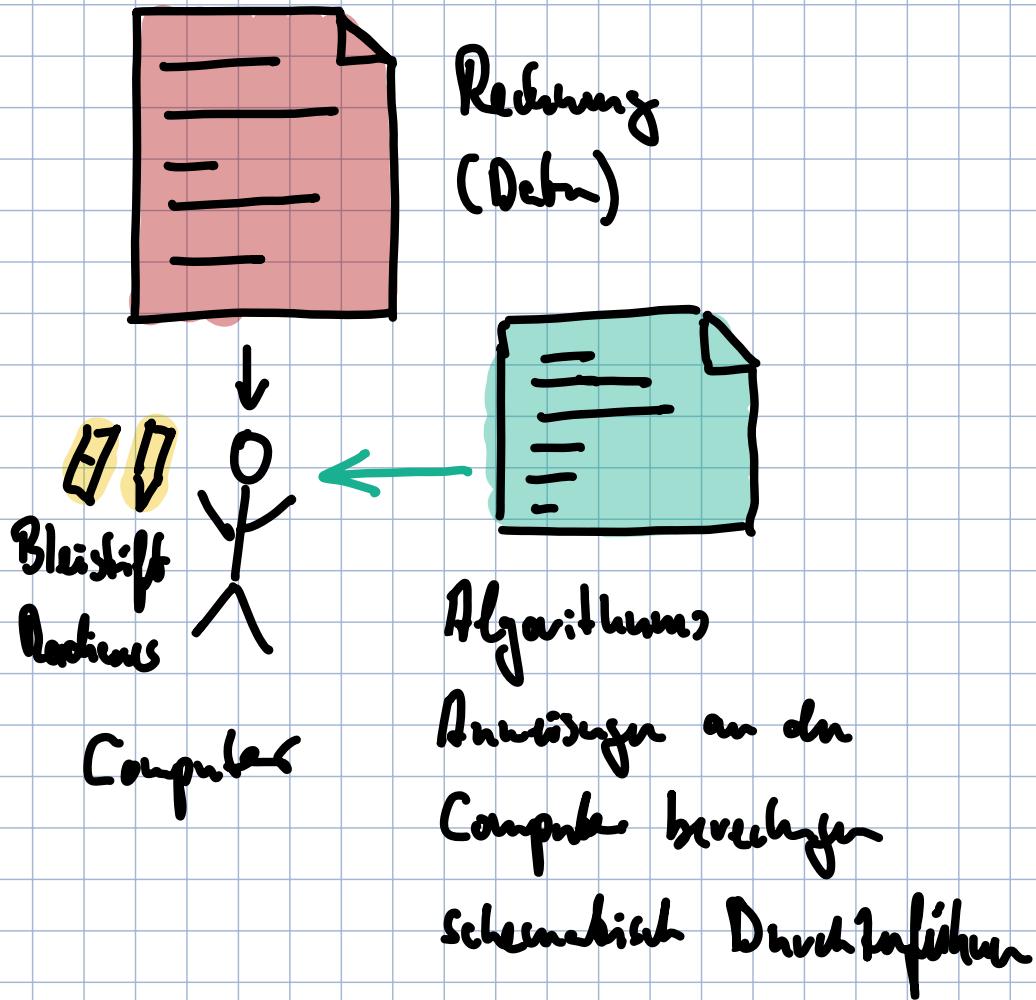


Programmable Logic Array

PLA



Die Turing-Maschine



$$\Sigma := \{0, 1\}$$

Eingabealphabet

$$\Gamma := \{0, 1, \square\}$$

Bandalphabet

$$S := \{S_0, \dots, S_n\}$$

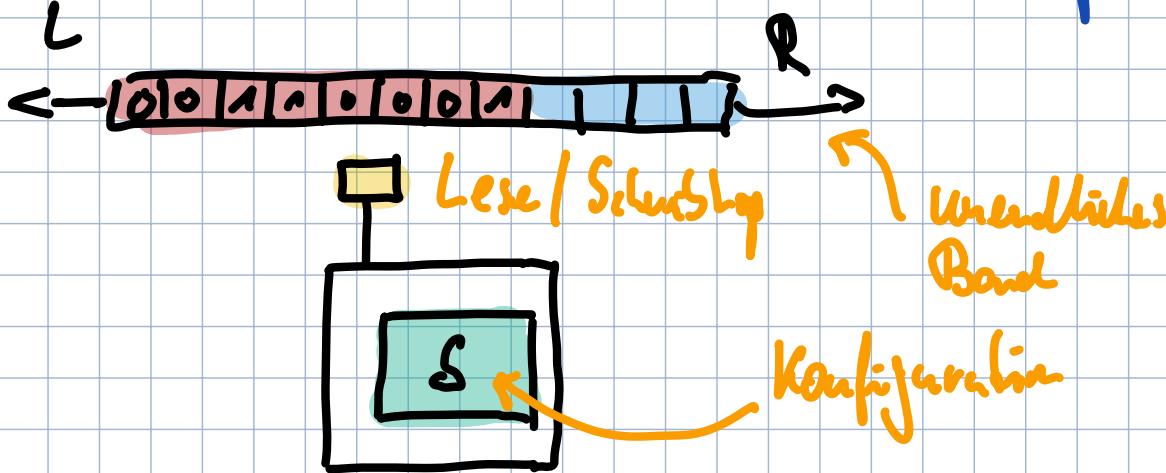
Zustandsmenge

$$s_0 = s \in S$$

Aufgangsstand

$$\delta := S \times \Gamma \times \{L, N, R\}$$

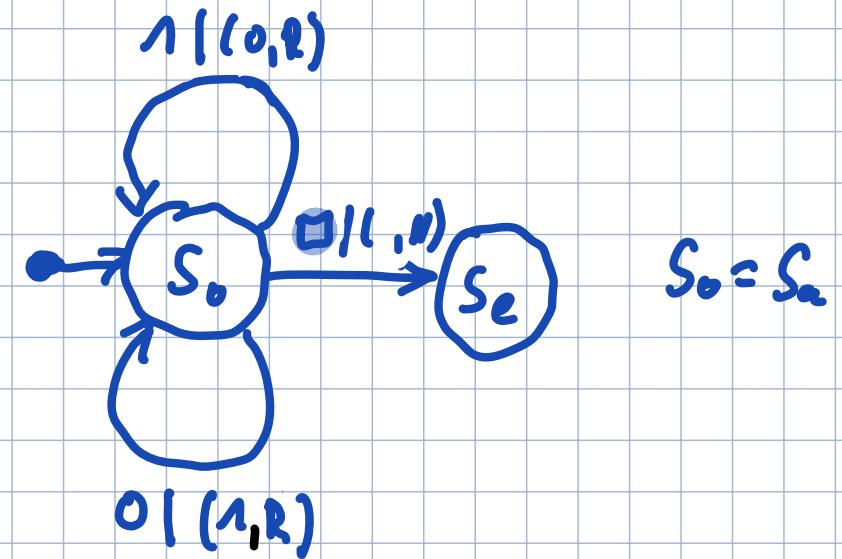
Die Turing-Maschine



Beispiel:

$$1 | (0, R) - 0 \rightarrow (1, R)$$

wenn 0 dann (1,R)



$$S_0 = S_e$$

$$\Sigma := \{0, 1\}$$

$$\Gamma := \{0, 1, 0\}$$

$$S := \{S_0, \dots, S_n\}$$

$$S_a = S \setminus S$$

$$S_r = S \setminus S$$

$$\delta := S \times \Gamma \times \{L, N, R\}$$

Eigeralphabet

Bandalphabet

Zustandsmenge

Aufgabenstellung

Erlaubtesband

zustandsübergangs-

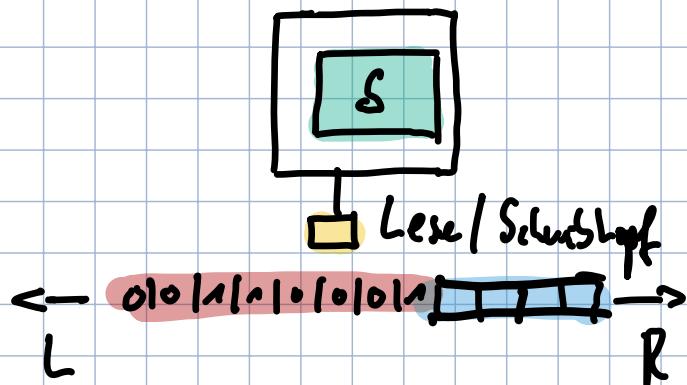
funktion (Konfiguration)

$$\delta := \begin{pmatrix} S_0 & 1 & 0 & R & S_0 \\ S_0 & 0 & 1 & R & S_0 \\ S_0 & \square & N & S_e \end{pmatrix}$$

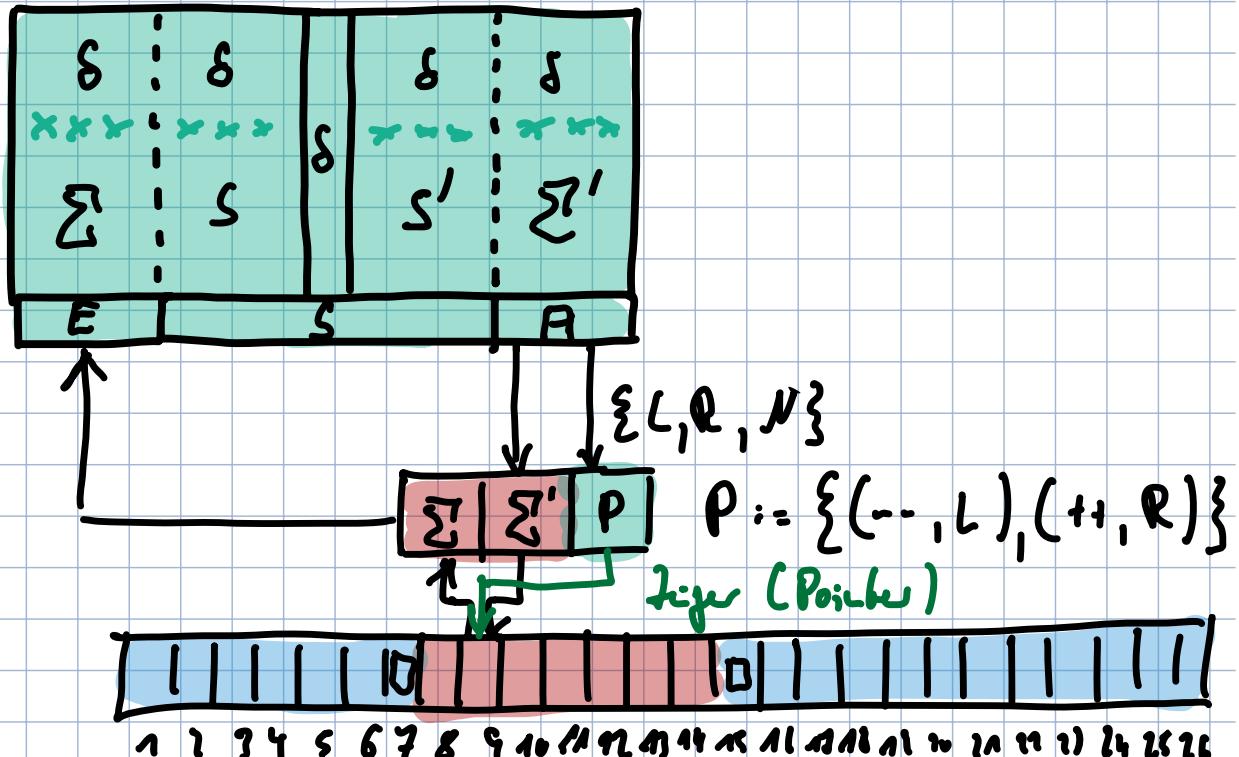
1 0 0 1 1 1 0 0 0 1 0 1 0 □

0 1 1 0 0 0 1 1 1 0 1 0 1 □

Die Turing-Maschine : Technisch



Programmable Logic Array PLA



$$\Sigma := \{0, 1\}$$

$$S_0 = S \in S$$

$$\Gamma := \{0, 1, 0\}$$

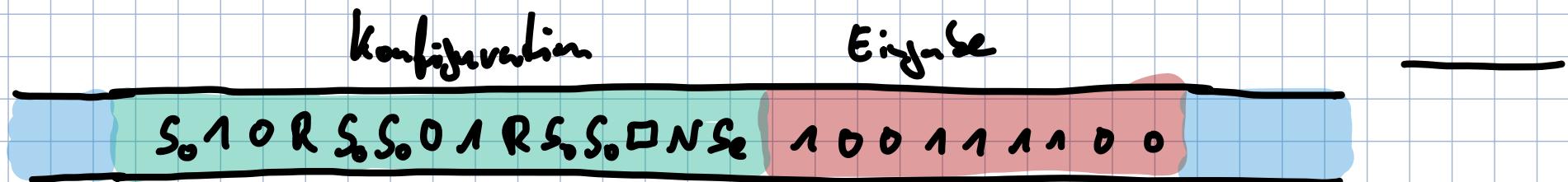
$$S_1 = S \in S$$

$$S := \{S_0, \dots, S_n\}$$

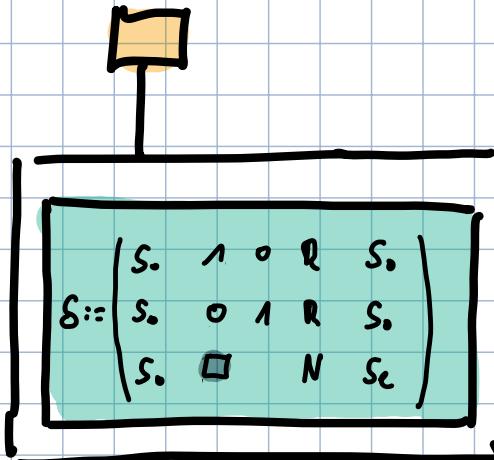
$$\delta := S \times \Gamma \times \{L, N, R\} \rightarrow S', (\Sigma', \{\{L, N, R\}\}')$$

Initialisierter Speicher (Wahlweise leere Zelle)

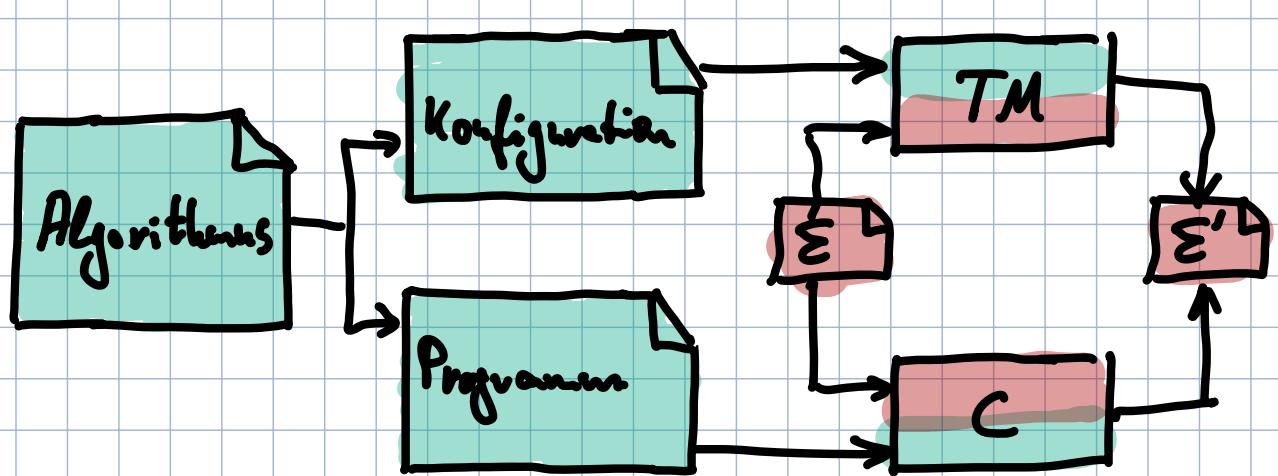
Universelle Turing-Maschine



① Laden der Konfiguration
von Band



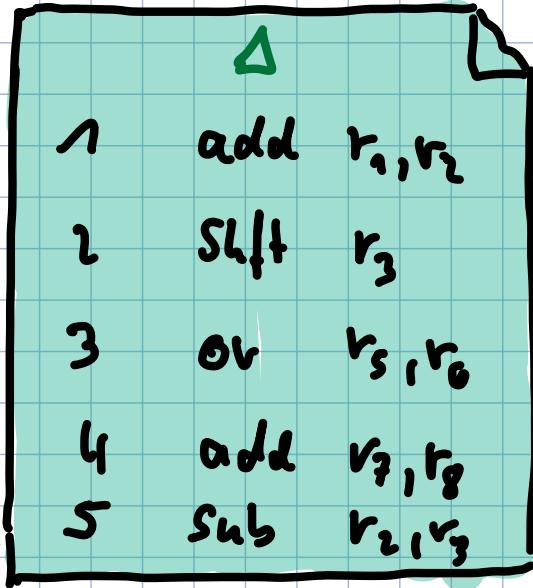
② Konfiguration auf
Eingabe anwenden und
Rückgabewerte.



Registersmaschine

Befehlszähler

5



Programm mit als Folge
durchnumzierte Befehle

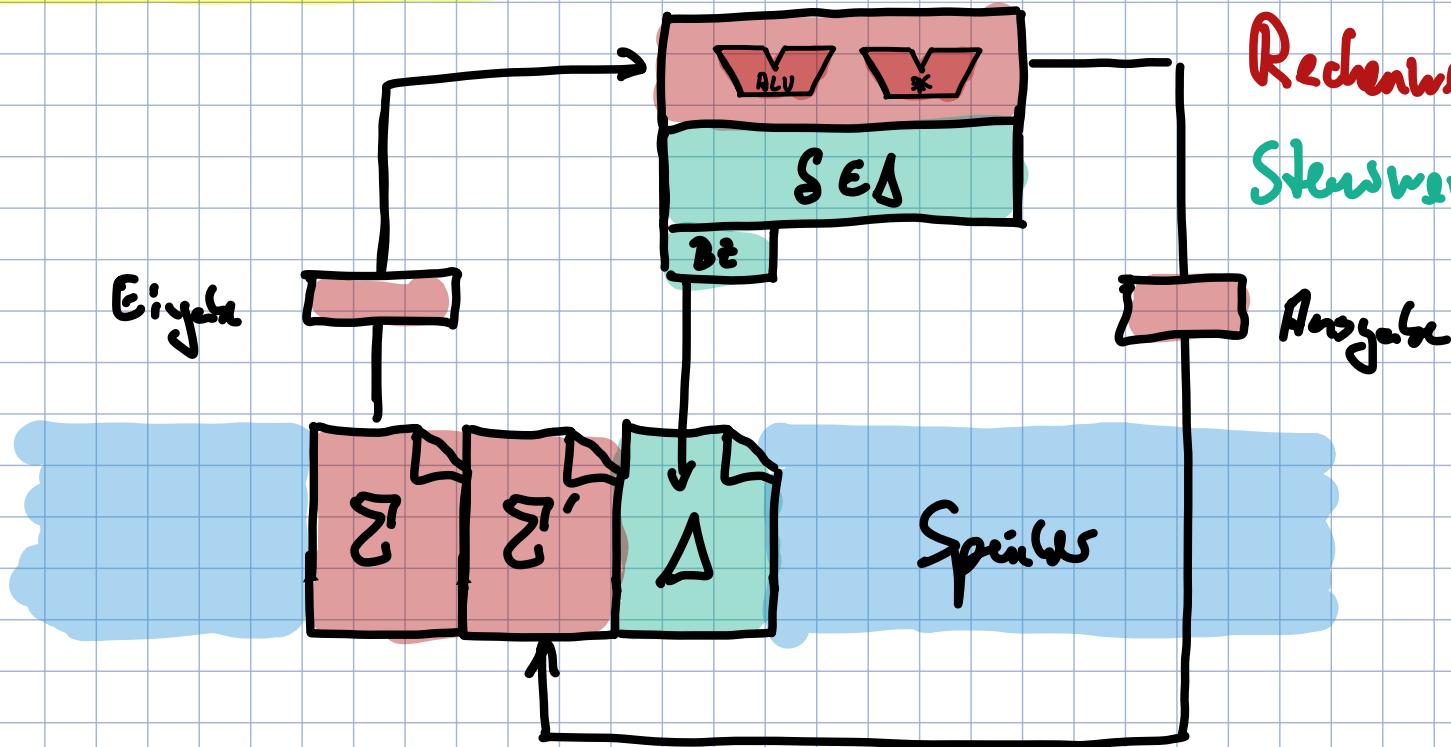
Im Speicher (Register File) liegen
unendlich viele natürliche Zahlen

i62 de IN

⋮	
-5	
-4	
-3	4
-2	3
-1	4
0	9
1	6
2	7
3	8
4	10
5	25
⋮	

Register als durchnumzierte
Speicherzellen

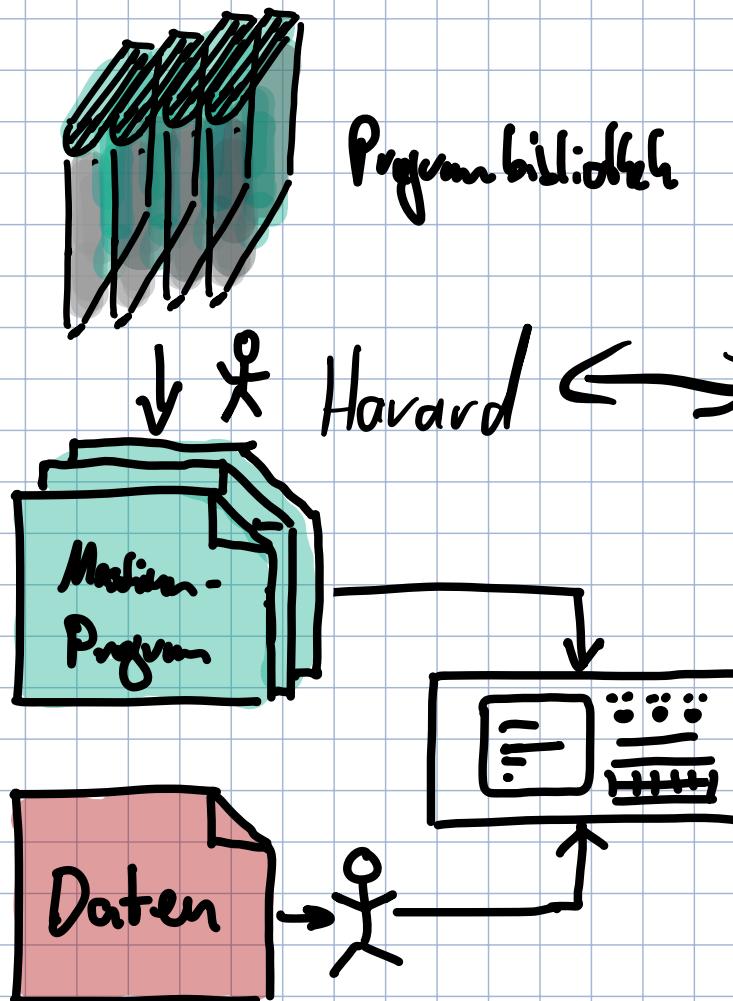
Das Prinzip von Neumanns



- 1) Befehl holen
(Instruction Fetch IF)
- 2) Befehl entschlüsseln
(Instruction Decode ID)
- 3) Eingabe holen
(Operand Fetch OF)
- 4) Befehl ausführen
(Execute EX)
- 5) Ausgabe schreiben
(Operand Store OS)

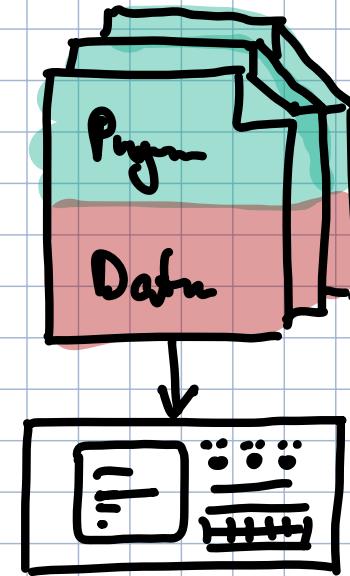
Befehlszyklus

Prinzip → Architektur



von-Neumann-Prinzip

KEINE TRENNUNG ZWISCHEN
PROGRAMM UND DATEN



implementiert Universelle Turingmaschine

Programme und Daten liegen
getrennt vor.

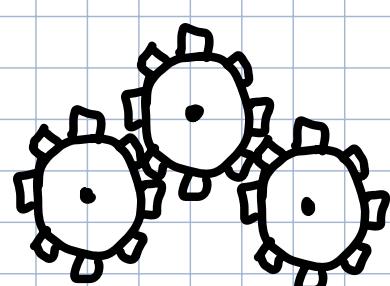
Programme können auch als
Daten aufgefasst werden

Das Prinzip von Neumanns

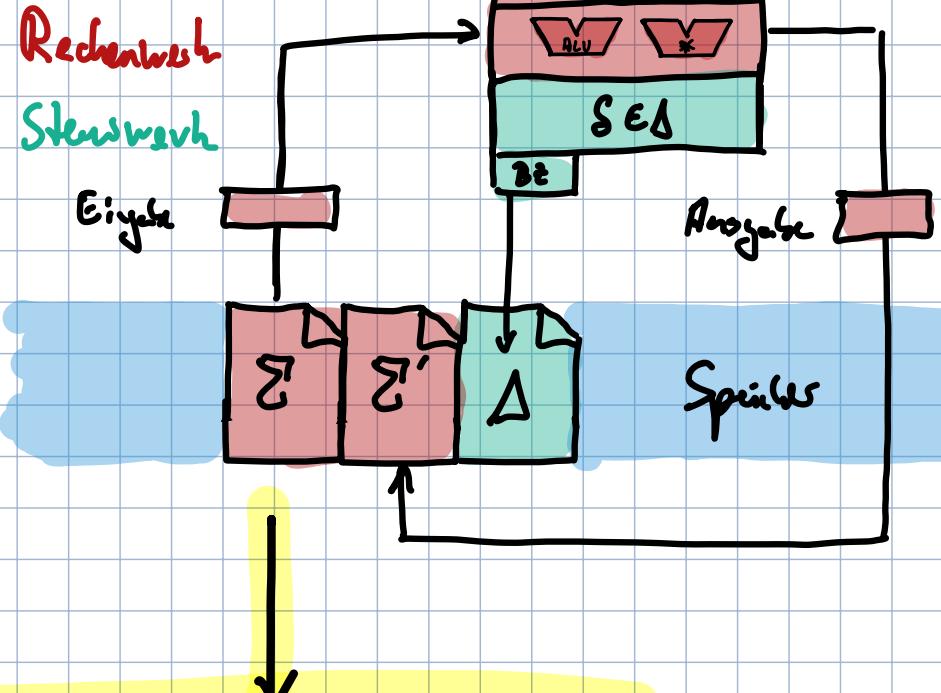
Prinzip: allgemeingültig

Sind wir fertig? Vorlesung?

Aber: Technologie führt zu Einschränkungen



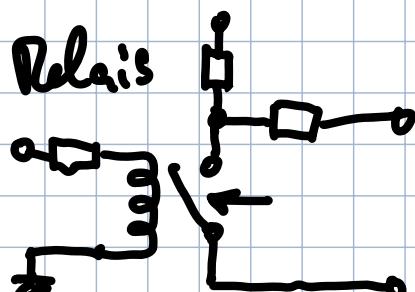
Analytical Machine



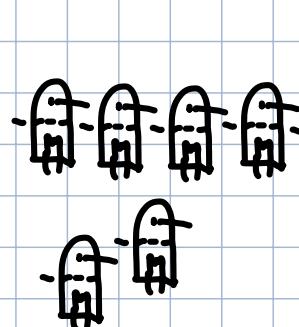
Mechanisch

Elektro-Mechanisch

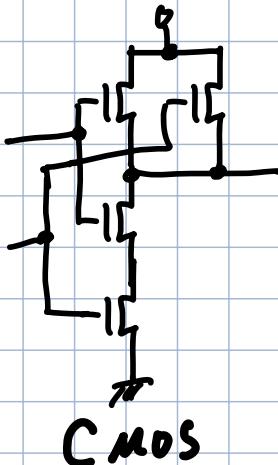
Elektronisch



Zuse T4



Röhren



CMOS

Funktion



Röhre

Transistor

InTEGRATED Schaltung

RTL

DTL TTL PMOS

NMOS

CMOS

18M760

PDP

4004

8086

80186

80286

80486

~600 kHz

~5 MHz

8 MHz

20 MHz

30 MHz

1 GHz

1 ps

1:10

~1 ns

750 ns

1600 ns

2 ns

11 ns

2.5 ns

1:1

~50 ns

1 MB/ \square



1.5B/ \square

~1 kB/ \square

~100 kB/ \square

1 kB/ \square

1 MB/ \square

Laufzeit
Williamsröhre

Magnetbremsspeicher

InTEGRATED Schaltung (DRAM)

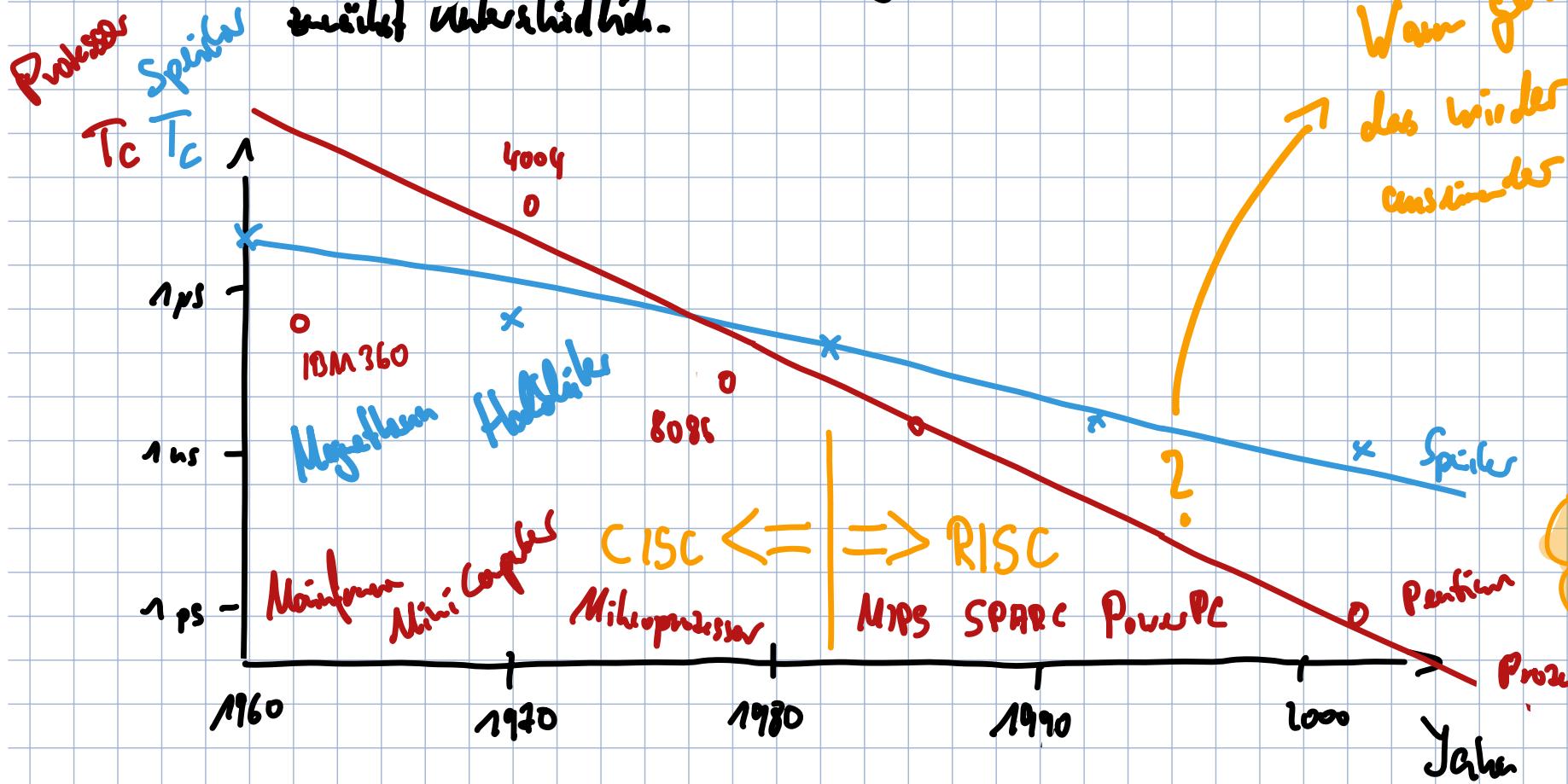
SRAM

DRAM



Die Speicherlücke:

Speicher und Prozesse sind technologisch
sehr viel unterschiedlich.



Lernziele:

Prinzipien erste Art:

Architektur

↑
hoher Einfluss
auf Prinzip

Befehlsatz

Strukturelles Konzept

Befehlausführung

Programming

Prinzipien zweiter Art:

Microarchitektur

Sind technologisch bedingt

Befehle effizient aus dem Speicher holen

Komplexität der Speicherausführung verbergen

Schnelle und effiziente Befehlausführung

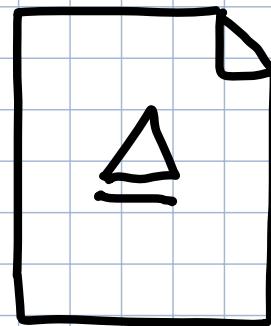
Komplexität der effizienten Befehlausführung verbergen

Beschreibung von Architektur und Microarchitektur,
von Verhalten und Struktur

Von Verhalten zur Maschine:

Architekten:

Verhalten

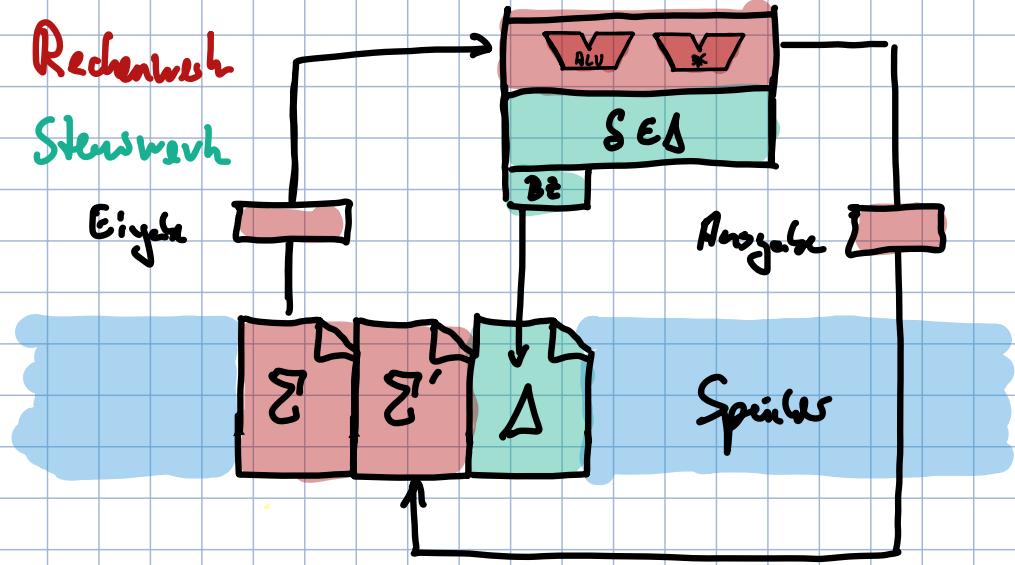


Befehlsatz

Architectural Reference Manual

Prinzip

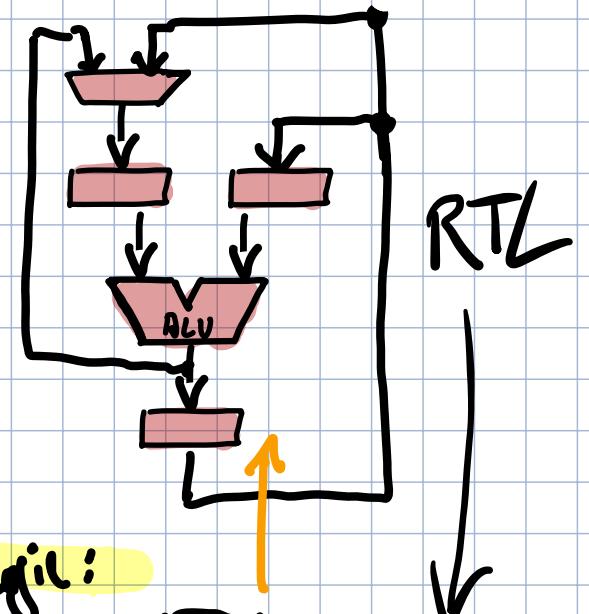
Rechenwerk
Steuermeth.
Eigene



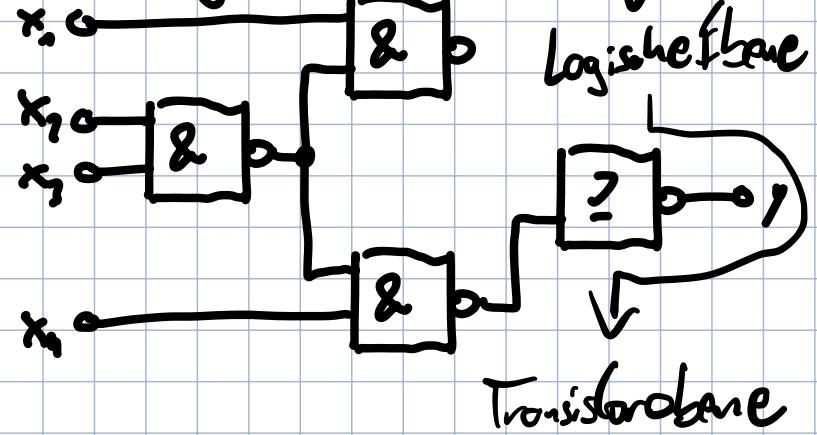
Mikroarchitektur:

Rechenstruktur:

Schreiber



Technologie:



Register architecture

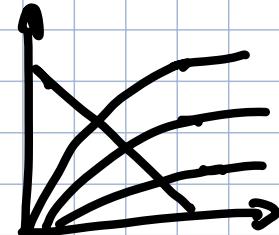
Vorwärts:

$$Acc[15 \text{ darb. } s] \leftarrow Reg[15 \text{ darb. } s]$$

Gesetz der Booleschen Algebra

$$y = (x_1 \text{ und } x_2 \text{ und } x_3) \text{ oder } (x_4 \text{ und } x_5 \text{ und } x_6)$$

$$y = \overline{\overline{x_1 x_2 x_3}} + \overline{\overline{x_4 x_5 x_6}}$$

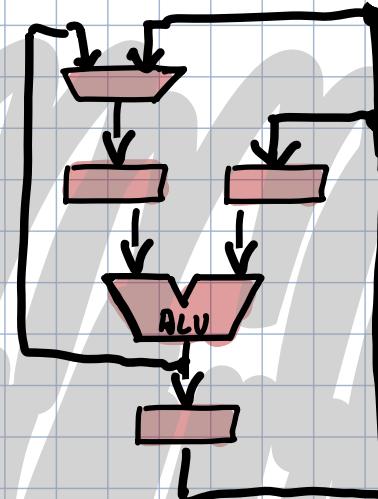


$$i = C \frac{dh}{ds}$$

Schmittschwelle

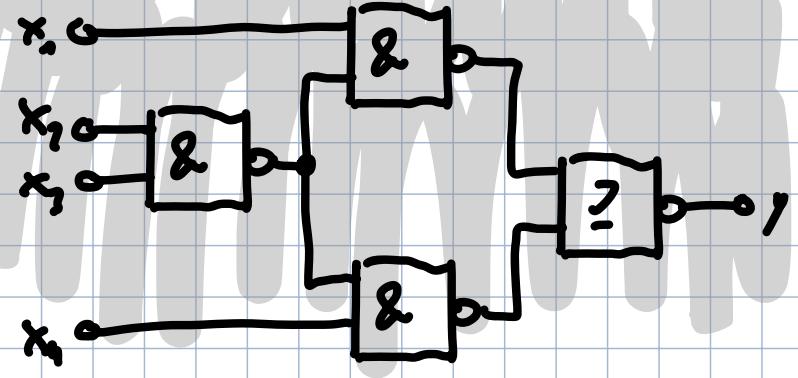
Register Transfer Element

Struktur:

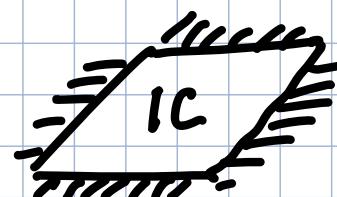


VHDL

Logische Element



Schaltung



Architectuur

Intel 8086

MC 68000

Parcels PC

Architectuur abstracteert van de Hardware

IBM 360

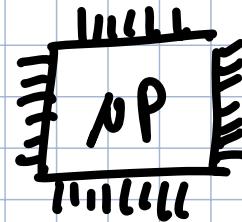
(Befehlssatz) Architectuur

IBM 304

Jede einzelne Maschine benötigt
ein eigens geschriebenes Programm

Mikroprocessor

Hardware



1975

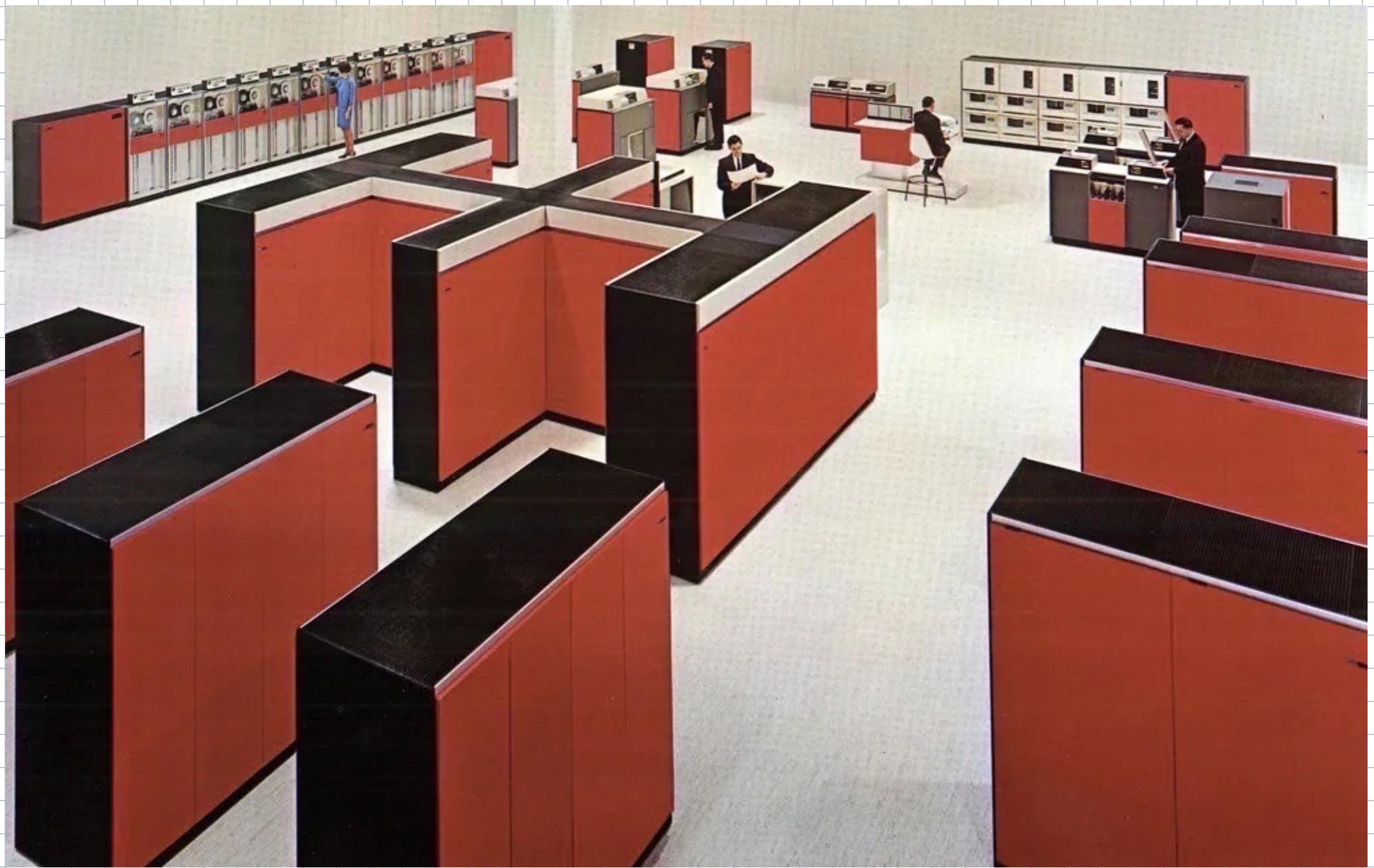


1963



1960

Mainframe-Computer : IBM/360 : The first architecture



Mini-Computers: The Computer went to ~~Inventor~~

Digital

DEC

PDP-11

Kenyon &
Richie



Micro - Computer : The computer went home



Apple Macintosh

IBM PC



Rechnerarchitektur (Befehlssatz)

Befehlstyp

Aritmetisch-Logisch
Transport
Verzweigen und Sprünge
Übertragungen
(Ein- und Ausgabe)

Technologiebestimmt

Operatorengriff

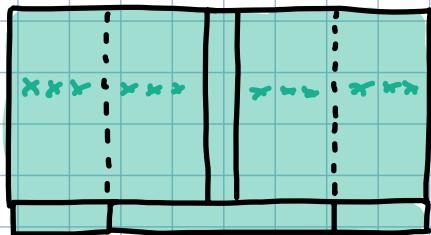
0 - Operandmaschine
n - Operandmaschine
n+1 - Operandmaschine

Spezialgriff

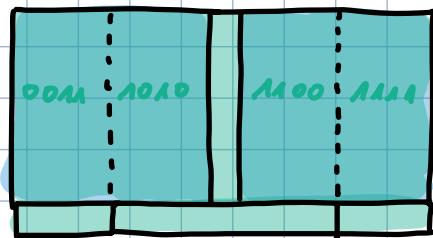
Implizite Addressing
Direkte Addressing
Indirekte Addressing
Indizierbare Addressing
Relativire Addressing

Komponenten des Rechnerarchitekturen (beleuchtet aus Sichtlage der Technischen Informatik) (Microarchitekturen)

Ablaufzähler (Steuerworte)



PLA

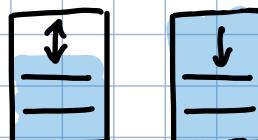


Microprogramm Steuerworte

Arithmetische Schaltungen

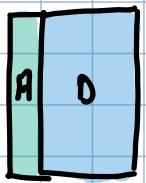


Speicher

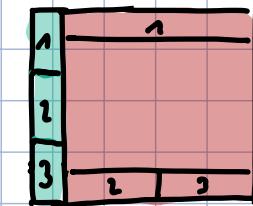


LIFO FIFO

Stack

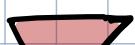


Register

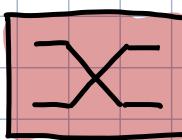


Mehrstufig-Register-Satz

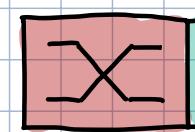
Vereinigungen



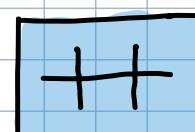
Multiplizierer



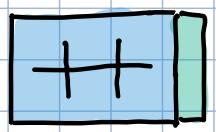
Interconnect



Interconnect mit DMA

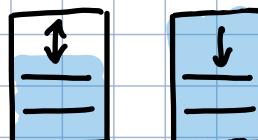


Bus



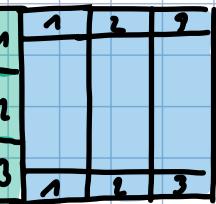
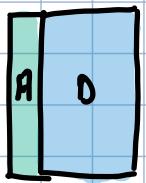
Bus mit DMA

Speicher



LIFO FIFO

Stack



Mehrstufig-Speicher