Gliederung

Brockhaus-Enzyklopädie: „*Informatik*“

Die Wissenschaft von der systematischen Verarbeitung von Informationen, besonders der automatischen Verarbeitung mit Hilfe von Digitalrechnern (→ Computer). . . .

Brockhaus-Enzyklopädie: „*Informatik*“

Die Wissenschaft von der systematischen Verarbeitung von Informationen, besonders der automatischen Verarbeitung mit Hilfe von Digitalrechnern (→ Computer). . . .

system. Verarbeitung: von-Neumann Paradigma

- Wie löst eine Folge elementarer Befehle (Programm) ein Problem?

⇒ Softwareentwicklung, Programmierung

Brockhaus-Enzyklopädie: „*Informatik*“

Die Wissenschaft von der systematischen Verarbeitung von Informationen, besonders der automatischen Verarbeitung mit Hilfe von Digitalrechnern (→ Computer). . . .

system. Verarbeitung: von-Neumann Paradigma

- Wie löst eine Folge elementarer Befehle (Programm) ein Problem?

⇒ Softwareentwicklung, Programmierung

Digitalrechner: das technische System dazu (Rechnerarchitektur)

- Wie wird Information (Zahlen, Zeichen) repräsentiert/codiert?
- Wie arbeiten technische Schaltungen (Hardware) Befehle ab?

⇒ Hardwareentwicklung

Trennung von Software und Hardware?

- seit 80er Jahren: unterschiedliche, getrennte Paradigmen
  - SW
    - Hardware ist vorgegeben
    - Abstraktion von der Hardware möglich
    - Programmierung in Hochsprachen (Produktivität)
  - HW
    - technische Werte als Optimierungsziel (Taktfrequenz, Latenz, Durchsatz, Leistungsaufnahme etc.)
    - getrieben von technischer Entwicklung (*Moore's Law*)
    - Maschinenbefehl wird auf Hardwarearchitektur ausgeführt
    - Kontext aus SW und Betriebssystem wird nicht beachtet
- Trend: technischer Fortschritt langsamer
  Leistungssteigerungen durch neue Architekturkonzepte

Trennung von Software und Hardware?

- seit 80er Jahren: unterschiedliche, getrennte Paradigmen
  - SW
    - Hardware ist vorgegeben
    - Abstraktion von der Hardware möglich
    - Programmierung in Hochsprachen (Produktivität)
  - HW
    - technische Werte als Optimierungsziel (Taktfrequenz, Latenz, Durchsatz, Leistungsaufnahme etc.)
    - getrieben von technischer Entwicklung (*Moore's Law*)
    - Maschinenbefehl wird auf Hardwarearchitektur ausgeführt
    - Kontext aus SW und Betriebssystem wird nicht beachtet
- Trend: technischer Fortschritt langsamer
  Leistungssteigerungen durch neue Architekturkonzepte

$\Rightarrow$ dies funktioniert seit Jahren!

Trennung von Software und Hardware? (cont.)

Konsequenz

verschiedene Sichtweisen funktionieren nicht mehr!

Programmierer: Grundverständnis techn. Funktionsweise und Rechnerarchitektur

Hardwaredesigner: Programmabläufe und Betriebssysteme wichtig!

Trennung von Software und Hardware? (cont.)

Konsequenz

verschiedene Sichtweisen funktionieren nicht mehr!

Programmierer: Grundverständnis techn. Funktionsweise und Rechnerarchitektur

Hardwaredesigner: Programmabläufe und Betriebssysteme wichtig!

Motivation für Rechnerstrukturen und Betriebssysteme

Trennung von Software und Hardware? (cont.)

Konsequenz

verschiedene Sichtweisen funktionieren nicht mehr!

Programmierer: Grundverständnis techn. Funktionsweise und Rechnerarchitektur

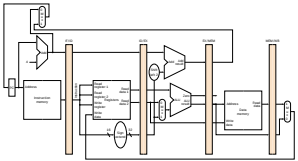
Hardwaredesigner: Programmabläufe und Betriebssysteme wichtig!

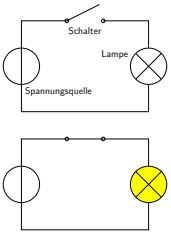
Motivation für Rechnerstrukturen und Betriebssysteme

⇒ Interaktion von SW und HW

```c
#include <stdio.h>

int main( int argc, char ** argv )
{ printf( "Hello, world!\n" );
  return 0;
}
```





- „performante“, „sichere“ Software programmieren
- Systemsicht / Variantenvielfalt von Mikroprozessorsystemen
- Bewertung von Trends und Perspektiven

Trennung von Software und Hardware? (cont.)

Konsequenz

verschiedene Sichtweisen funktionieren nicht mehr!

Programmierer: Grundverständnis techn. Funktionsweise und Rechnerarchitektur

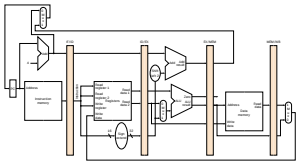
Hardwaredesigner: Programmabläufe und Betriebssysteme wichtig!

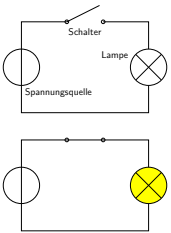
Motivation für Rechnerstrukturen und Betriebssysteme

⇒ Interaktion von SW und HW

```
#include <stdio.h>

int main( int argc, char ** argv )
{ printf( "Hello, world!\n" );
  return 0;
}
```





- „performante“, „sichere“ Software programmieren
- Systemsicht / Variantenvielfalt von Mikroprozessorsystemen
- Bewertung von Trends und Perspektiven

⇒ *Wie funktioniert ein Computer?*

Programmierbeispiel: Fakultät

- Formel $n! = \prod_{n=1}^{n} n \qquad 0! = 1$
- C Code

```c
/* Fakultät... - nicht rekursiv */

#include <stdio.h>
#include <stdlib.h>
#include <string.h>

unsigned int fak(unsigned int n)
{ unsigned int res = 1;
  while (n > 1)
  { res = n * res;
    n   = n - 1;
  }
  return res;
}

int main(int argc, char **argv)
{ int arg;

  if ((argc != 2) || ((arg = atoi(argv[1])) < 0))
  { printf ("usage: \"fak <n>\", n >= 0\n");
    exit(1);
  }

  printf ("%d! = %d\n", arg, fak(arg));
  exit(0);
}
```

Wie funktioniert ein Computer?

```
unsigned int fak(unsigned int n)
{ unsigned int res = 1;
  while (n > 1)
  { res = n * res;
    n   = n - 1;
  }
  return res;
}
```

Ablauf 4!

n =	
res =	

Wie funktioniert ein Computer?

```
unsigned int fak(unsigned int n)
{ unsigned int res = 1;
  while (n > 1)
  { res = n * res;
    n   = n - 1;
  }
  return res;
}
```

Ablauf 4!

n = 4

res =

Wie funktioniert ein Computer?

```
unsigned int fak(unsigned int n)
{ unsigned int res = 1;
  while (n > 1)
  { res = n * res;
    n   = n - 1;
  }
  return res;
}
```

Ablauf 4!

n = 4

res = 1

Wie funktioniert ein Computer?

```
unsigned int fak(unsigned int n)
{ unsigned int res = 1;
  while (n > 1)
  { res = n * res;
    n   = n - 1;
  }
  return res;
}
```

Ablauf 4!

n = 4
res = 1

Wie funktioniert ein Computer?

```
unsigned int fak(unsigned int n)
{ unsigned int res = 1;
  while (n > 1)
  { res = n * res;
    n   = n - 1;
  }
  return res;
}
```

Ablauf 4!

n = 4	
res = 1	4

Wie funktioniert ein Computer?

```
unsigned int fak(unsigned int n)
{ unsigned int res = 1;
  while (n > 1)
  { res = n * res;
    n   = n - 1;
  }
  return res;
}
```

Ablauf 4!

n =	4	3
res =	1	4

Wie funktioniert ein Computer?

```
unsigned int fak(unsigned int n)
{ unsigned int res = 1;
  while (n > 1)
  { res = n * res;
    n   = n - 1;
  }
  return res;
}
```

Ablauf 4!

n =	4	3
res =	1	4

Wie funktioniert ein Computer?


```
unsigned int fak(unsigned int n)
{ unsigned int res = 1;
  while (n > 1)
  { res = n * res;
    n   = n - 1;
  }
  return res;
}
```

Ablauf 4!

n =	4	3	
res =	1	4	12

Wie funktioniert ein Computer?

```
unsigned int fak(unsigned int n)
{ unsigned int res = 1;
  while (n > 1)
  { res = n * res;
    n   = n - 1;
  }
  return res;
}
```

Ablauf 4!

n =	4	3	2
res =	1	4	12

Wie funktioniert ein Computer?

```
unsigned int fak(unsigned int n)
{ unsigned int res = 1;
  while (n > 1)
  { res = n * res;
    n   = n - 1;
  }
  return res;
}
```

Ablauf 4!

n =	4	3	2
res =	1	4	12

Wie funktioniert ein Computer?

```
unsigned int fak(unsigned int n)
{ unsigned int res = 1;
  while (n > 1)
  { res = n * res;
    n   = n - 1;
  }
  return res;
}
```

Ablauf 4!

n =	4	3	2	
res =	1	4	12	24

Wie funktioniert ein Computer?

```
unsigned int fak(unsigned int n)
{ unsigned int res = 1;
  while (n > 1)
  { res = n * res;
    n   = n - 1;
  }
  return res;
}
```

Ablauf 4!

n =	4	3	2	1
res =	1	4	12	24

Wie funktioniert ein Computer?

```
unsigned int fak(unsigned int n)
{ unsigned int res = 1;
  while (n > 1)
  { res = n * res;
    n   = n - 1;
  }
  return res;
}
```

Ablauf 4!

n =	4	3	2	1
res =	1	4	12	24

Wie funktioniert ein Computer?

```
unsigned int fak(unsigned int n)
{ unsigned int res = 1;
  while (n > 1)
  { res = n * res;
    n   = n - 1;
  }
  return res;
}
```

- *Wie werden Programmanweisungen schrittweise abgearbeitet?*

Wie funktioniert ein Computer?

```
unsigned int fak(unsigned int n)
{ unsigned int res = 1;
  while (n > 1)
  { res = n * res;
    n   = n - 1;
  }
  return res;
}
```

- *Wie werden Programmanweisungen schrittweise abgearbeitet?*
- *Wie wird Information (hier Zahlen) technisch dargestellt?*

Wie funktioniert ein Computer?


```
unsigned int fak(unsigned int n)
{ unsigned int res = 1;
  while (n > 1)
  { res = n * res;
    n   = n - 1;
  }
  return res;
}
```

- *Wie werden Programmanweisungen schrittweise abgearbeitet?*
- *Wie wird Information (hier Zahlen) technisch dargestellt?*
- *Welche technischen Komponenten werden benötigt?*

  Wie funktionieren sie? Wie kann damit gerechnet werden?

Wie funktioniert ein Computer?

```
unsigned int fak(unsigned int n)
{ unsigned int res = 1;
  while (n > 1)
  { res = n * res;
    n   = n - 1;
  }
  return res;
}
```

- *Wie werden Programmanweisungen schrittweise abgearbeitet?*
- *Wie wird Information (hier Zahlen) technisch dargestellt?*
- *Welche technischen Komponenten werden benötigt?*
  Wie funktionieren sie? Wie kann damit gerechnet werden?
- *Welche Mechanismen sorgen dafür, dass mehrere Anwendungen (und mehrere Benutzer) „gleichzeitig“ arbeiten?*

Fortschritt

1. ständige technische Fortschritte in Mikro- und Optoelektronik mit einem weiterhin *exponentiellen* Wachstum (50 % . . . 100 % pro Jahr)
   - Rechenleistung von Prozessoren / „Performanz“
   - Speicherkapazität Hauptspeicher (DRAM, SRAM, FLASH)
   - Speicherkapazität Langzeitspeicher (Festplatten, FLASH)
   - Übertragungsraten / Bandbreite (Netzwerke)
2. neue Entwurfsparadigmen und -werkzeuge

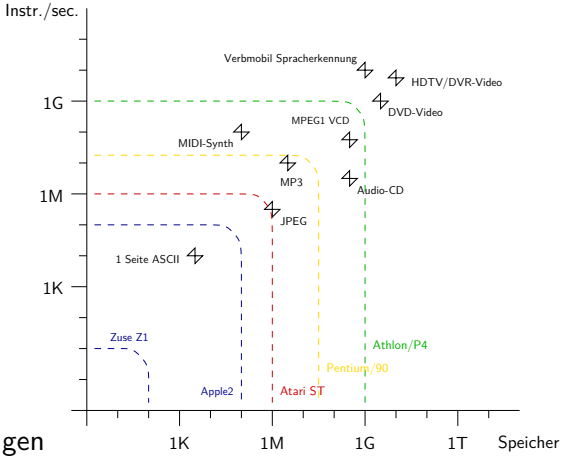
$\Rightarrow$ Möglichkeiten und Anwendungsfelder

$\Rightarrow$ Produkte und Techniken

Fortschritt (cont.)

Kriterien / Maßgrößen

- Rechenleistung: MIPS
- MBytes (RAM, HDD)
- Mbps
- MPixel



⇒ jede Rechnergeneration erlaubt neue Anwendungen

Beispiel: technischer Fortschritt

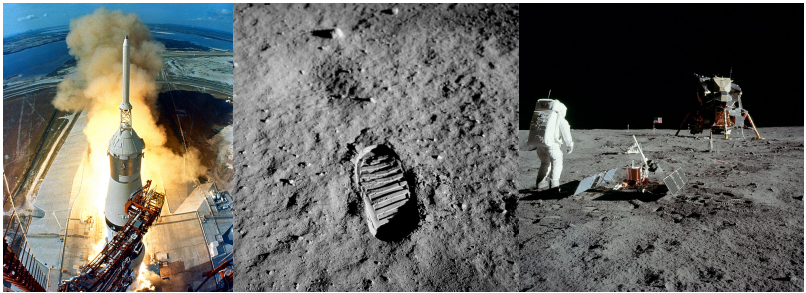
Hardware für den Mondflug

Beispiel: technischer Fortschritt

Hardware für den Mondflug

. . . aber nicht 2024

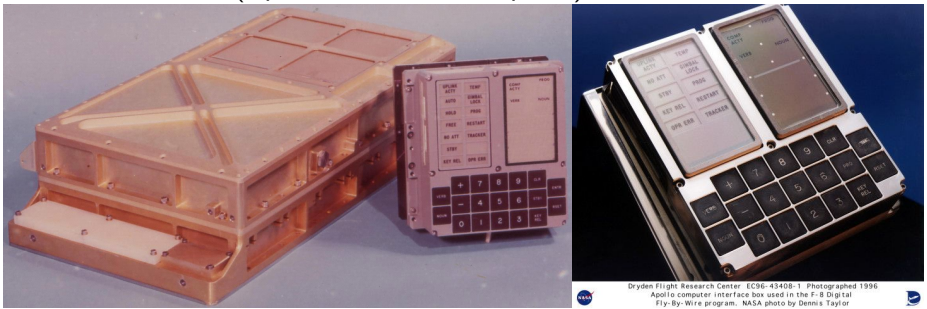
Beispiel: Apollo 11 (1969)



- bernd-leitenberger.de/computer-raumfahrt1.shtml
- history.nasa.gov/computers/contents.html
- en.wikipedia.org/wiki/Apollo_Guidance_Computer
- en.wikipedia.org/wiki/IBM_System/360
  www.computerhistory.org/revolution/mainframe-computers/7

vor 55 Jahren (cont.)

1. Bordrechner: AGC (Apollo Guidance Computer)



- Dimension $61 \times 32 \times 15{,}0\,\text{cm}$ $31{,}7\,\text{kg}$
  $20 \times 20 \times 17{,}5\,\text{cm}$ $8{,}0\,\text{kg}$
- Taktfrequenz: 1,024 MHz
- 16-bit Worte, nur Festkomma
- Speicher ROM 36 KWorte 72 KByte
  RAM 2 KWorte 4 KByte

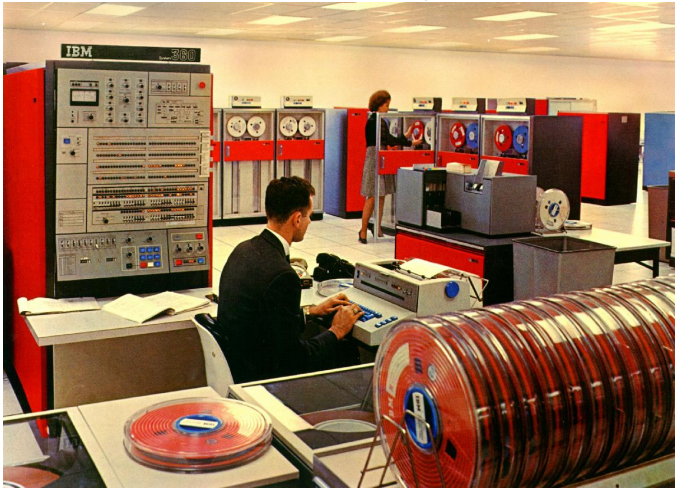
Addition: $\approx 20\,\mu\text{s}$, mehrere Takte

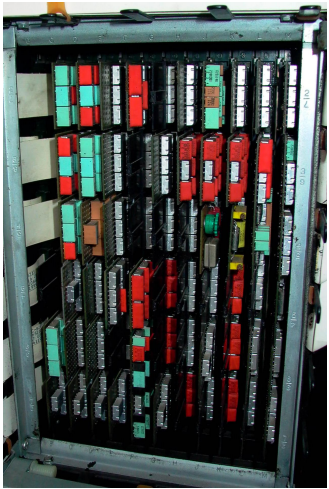
Zykluszeit: $11{,}7\,\mu\text{s}$, 12 Takte, 85,3 KHz

vor 55 Jahren (cont.)

2. mehrere Großrechner: IBM System/360 Model 75s

YouTube Video





vor 55 Jahren (cont.)

- je nach Ausstattung: Anzahl der „Schränke“
- Taktfrequenz: bis 5 MHz
- 32-bit Worte, 24-bit Adressraum (16 MByte)
- Speicherhierarchie: bis 1 MByte Hauptspeicher (1,3 MHz Zykluszeit)
- (eigene) Fließkomma Formate
- Rechenleistung: 0,7 Dhrystone MIPS

vor 55 Jahren (cont.)

- je nach Ausstattung: Anzahl der „Schränke“
- Taktfrequenz: bis 5 MHz
- 32-bit Worte, 24-bit Adressraum (16 MByte)
- Speicherhierarchie: bis 1 MByte Hauptspeicher (1,3 MHz Zykluszeit)
- (eigene) Fließkomma Formate
- Rechenleistung: 0,7 Dhrystone MIPS

- . . . und 2016 [1]

	CPU	Cores	[DMIPS]	F_{clk}[GHz]	
Smartphone	Exynos 8890	8	47 840	2,3	2016
Desktop PC	Core i7 6950X	10	317 900	3,0	

[1] Daten aktueller Systeme sind nicht verfügbar – Benchmark ist überholt

vor 55 Jahren (cont.)

- je nach Ausstattung: Anzahl der „Schränke“
- Taktfrequenz: bis 5 MHz
- 32-bit Worte, 24-bit Adressraum (16 MByte)
- Speicherhierarchie: bis 1 MByte Hauptspeicher (1,3 MHz Zykluszeit)
- (eigene) Fließkomma Formate
- Rechenleistung: 0,7 Dhrystone MIPS

- . . . und 2020 [1]

	CPU	Cores	[DMIPS]	F_{clk}[GHz]	
Smartphone	Exynos 8890	8	47 840	2,3	2016
Desktop PC	Core i7 6950X	10	317 900	3,0	
Workstation	Ryzen Threadripper 3990X	64	2 356 230	4,35	2020

[1] Daten aktueller Systeme sind nicht verfügbar – Benchmark ist überholt

vor 55 Jahren (cont.)

- je nach Ausstattung: Anzahl der „Schränke“
- Taktfrequenz: bis 5 MHz
- 32-bit Worte, 24-bit Adressraum (16 MByte)
- Speicherhierarchie: bis 1 MByte Hauptspeicher (1,3 MHz Zykluszeit)
- (eigene) Fließkomma Formate
- Rechenleistung: 0,7 Dhrystone MIPS

- . . . und 2020 [1]

	CPU	Cores	[DMIPS]	F_{clk}[GHz]	
Smartphone	Exynos 8890	8	47 840	2,3	2016
Desktop PC	Core i7 6950X	10	317 900	3,0	
Workstation	Ryzen Threadripper 3990X	64	2 356 230	4,35	2020

⇒ Moore's Law

[1] Daten aktueller Systeme sind nicht verfügbar – Benchmark ist überholt

Timeline: Rechnergeschichte

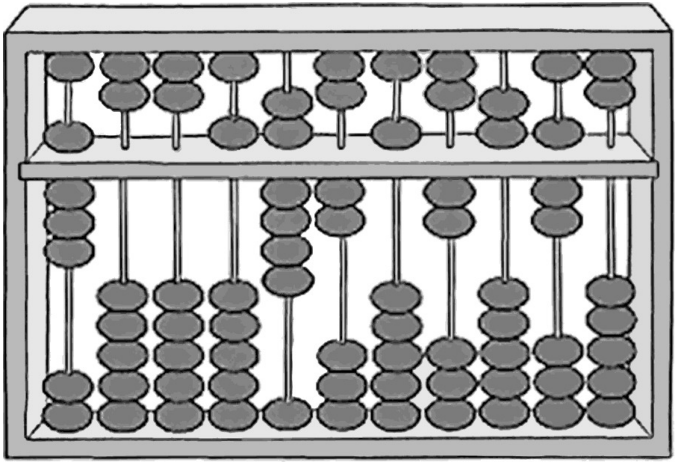
????	Abakus als erste Rechenhilfe
1642	Pascal: Addierer/Subtrahierer
1671	Leibniz: Vier-Operationen-Rechenmaschine
1837	Babbage: Analytical Engine
1937	Zuse: Z1 (mechanisch)
1939	Zuse: Z3 (Relais, Gleitkomma)
1941	Atanasoff & Berry: ABC (Röhren, Magnettrommel)
1944	Mc-Culloch Pitts (Neuronenmodell)
1946	Eckert & Mauchly: ENIAC (Röhren)
1949	Eckert, Mauchly, von Neumann: EDVAC (erster speicherprogrammierter Rechner)
1949	Manchester Mark-1 (Indexregister)

Abakus

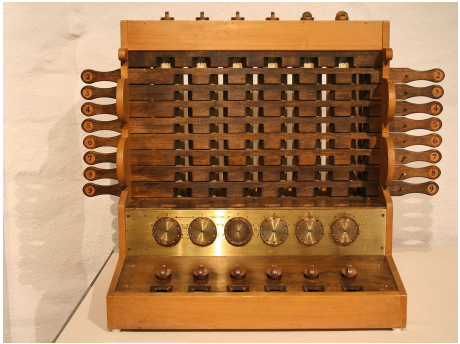
Wert in Spalte	8	0	0	5	14	2	5	2	10	7	0
Zehnerpotenz der Spalte	10^{10}	10^{9}	10^{8}	10^{7}	10^{6}	10^{5}	10^{4}	10^{3}	10^{2}	10^{1}	10^{0}

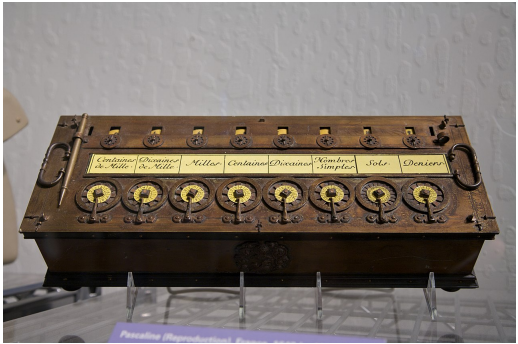
Kugel = 5

Kugel = 1



Mechanische Rechenmaschinen





1623 Schickard: Sprossenrad, Addierer/Subtrahierer

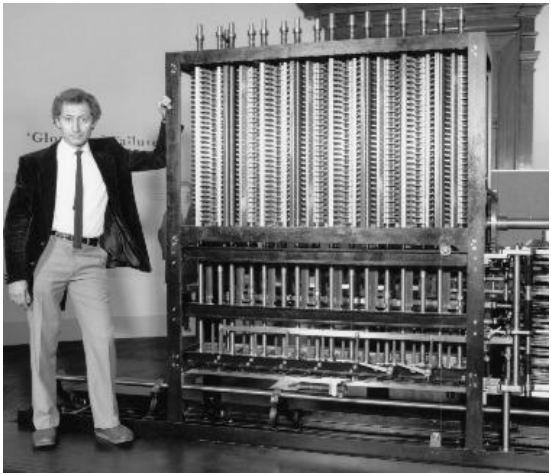
1642 Pascal: „Pascalene“

1673 Leibniz: Staffelwalze, Multiplikation/Division

1774 Philipp Matthäus Hahn: erste gebrauchsfähige ‘4-Spezies“-Maschine

Difference Engine

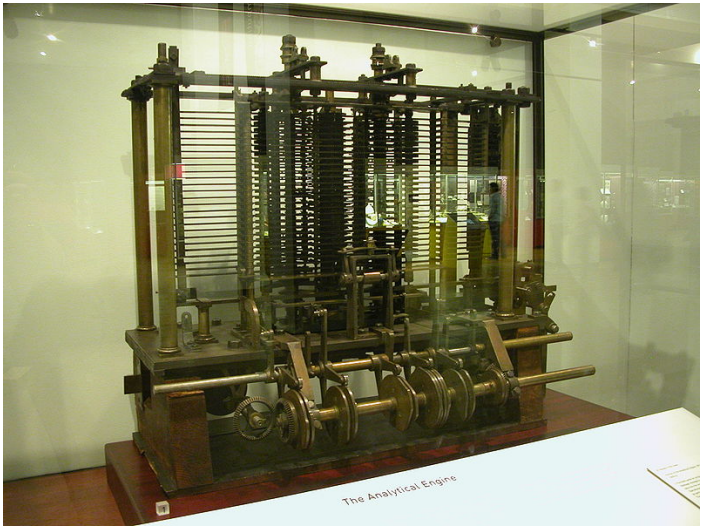
Charles Babbage 1822: Berechnung nautischer Tabellen



Original von 1832 und Nachbau von 1989, London Science Museum

Analytical Engine

Charles Babbage 1837-1871: frei programmierbar, Lochkarten, unvollendet



Programmierbarkeit

- nicht mehr eine Maschine für eine Aufgabe / ein Problem sondern feste Hardware

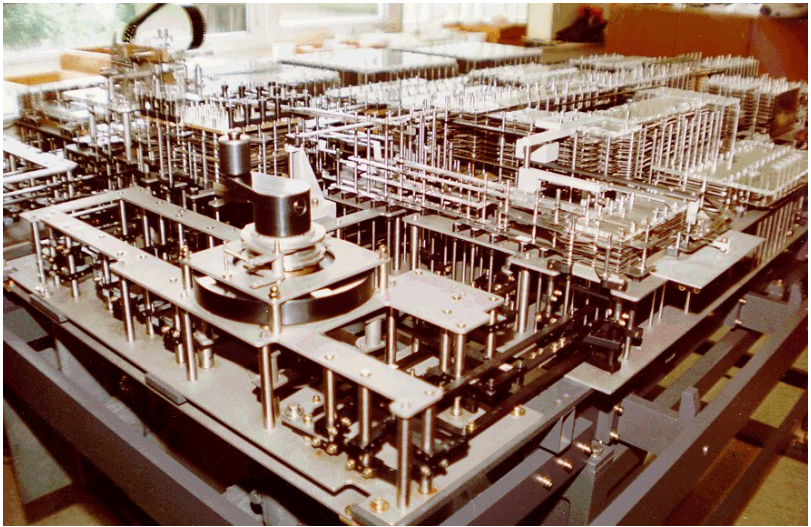
\+ problemspezifisch wird ein flexibles Programm entwickelt

- schrittweises Problemlösen $\Rightarrow$ Algorithmen
- zentrale Paradigmen der Informatik

$\Rightarrow$ von-Neumann Konzept

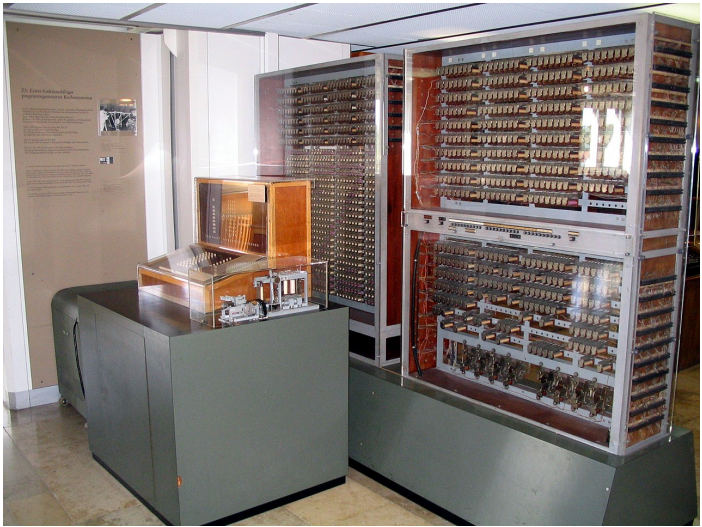
Zuse Z1

Konrad Zuse 1937: 64 Register, 22-bit, mechanisch, Lochfilm



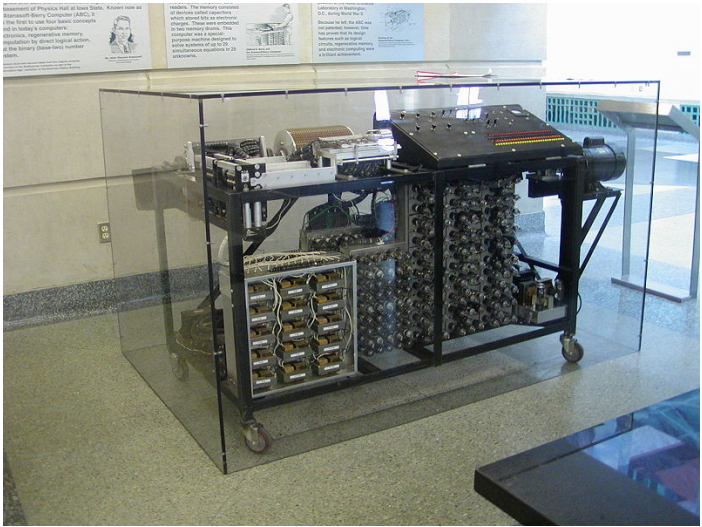
Zuse Z3

Konrad Zuse 1941, 64 Register, 22-bit, 2000 Relays, Lochfilm



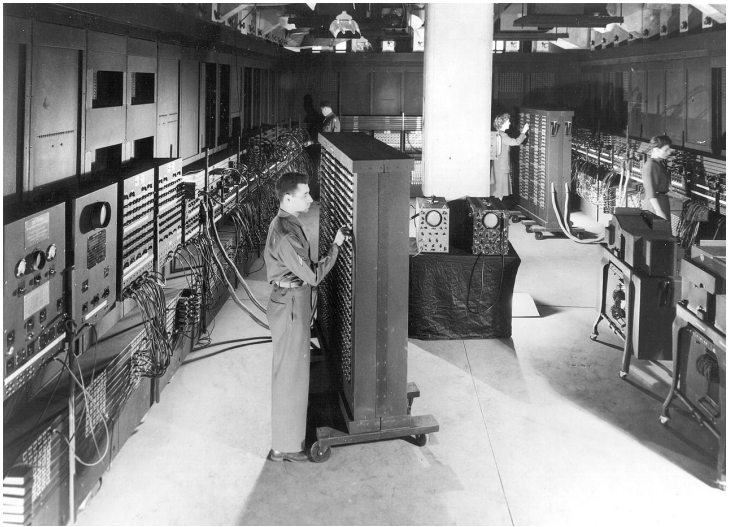
Atanasoff-Berry Computer (ABC)

J.V. Atanasoff 1942: 50-bit Festkomma, Röhren und Trommelspeicher, fest programmiert, erste ALU

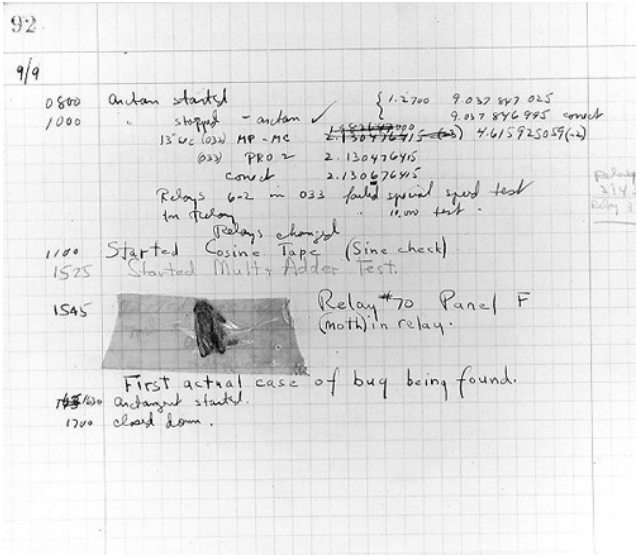


ENIAC – Electronic Numerical Integrator and Computer

J. Mauchly & J.P. Eckert, 1946: Röhren, Steckbrett-Programm

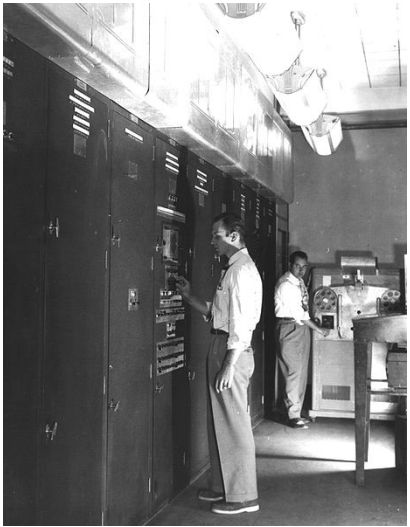


First computer bug



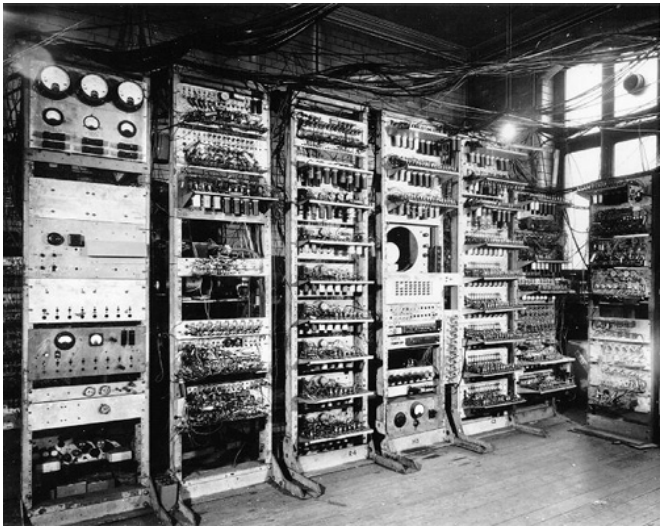
EDVAC

J. Mauchly, J.P. Eckert & J. von Neumann, 1949: Röhren, speicherprogrammiert



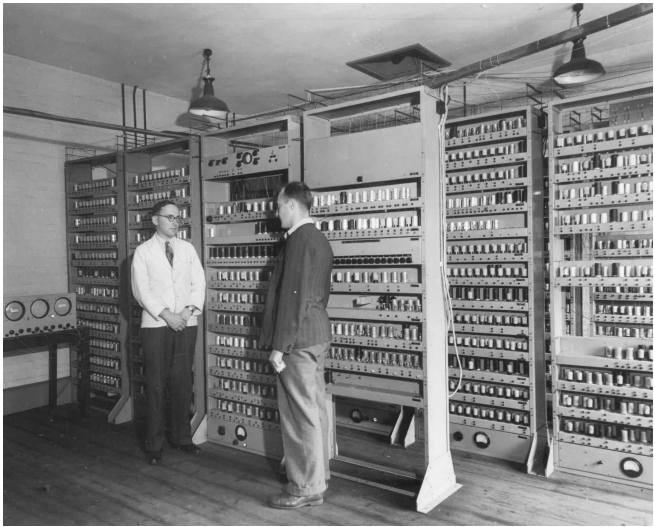
Manchester Mark-1

F.C. Williams & T. Kilburn, 1949: Trommelspeicher, Indexregister

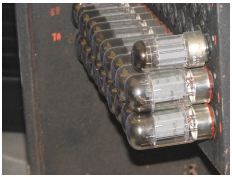


Manchester EDSAC

M. Wilkes 1951: Mikroprogrammierung, Unterprogramme, speicherprogrammiert



erste Computer, ca. 1950



- zunächst noch kaum Softwareunterstützung
- nur zwei Schichten:

1. Programmierung in elementarer Maschinensprache (ISA level)
2. Hardware in Röhrentechnik (device logic level)
   - Hardware kompliziert und unzuverlässig

Mikroprogrammierung (Maurice Wilkes, Cambridge, 1951):

- Programmierung in komfortabler Maschinensprache
- Mikroprogramm-Steuerwerk (Interpreter)
- einfache, zuverlässigere Hardware
- Grundidee der **CISC**-Rechner: VAX, 68000, 8086 etc.
  (**C**omplex **I**nstruction **S**et **C**omputer)

erste Betriebssysteme

- erste Rechner jeweils nur von einer Person benutzt
- Anwender = Programmierer = Operator
- Programm laden, ausführen, Fehler suchen usw.

$\Rightarrow$ Maschine wird nicht gut ausgelastet

$\Rightarrow$ Anwender mit lästigen Details überfordert

Einführung von **Betriebssystemen**

- „system calls“
- Batch-Modus: Programm abschicken, warten
- Resultate am nächsten Tag abholen

zweite Generation: Transistoren

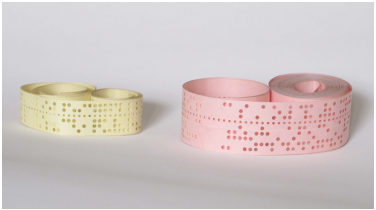
- Erfindung des Transistors 1948 (J. Bardeen, W. Brattain, W. Shockley)
- schneller, zuverlässiger, sparsamer als Röhren
- Miniaturisierung und dramatische Kostensenkung

- Beispiel: Digital Equipment Corporation PDP-1 (1961)
  - 4Ki Speicher (4096 Worte à 18-bit)
  - 200 KHz Taktfrequenz
  - 120 000 $
  - Grafikdisplay: erste Computerspiele
- Nachfolger PDP-8: 16 000 $
  - erstes Bussystem
  - 50 000 Stück verkauft

Festplatten

Massenspeicher bei frühen Computern

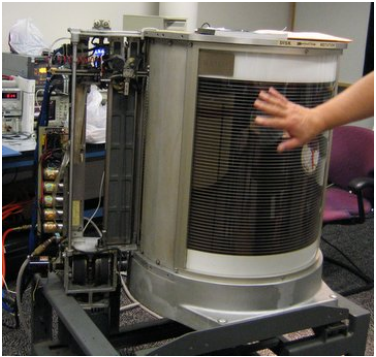
- Lochkarten
- Lochstreifen
- Magnetband



- Magnettrommel
- Festplatte

  IBM 350 RAMAC (1956)

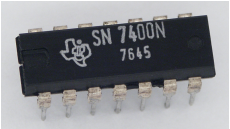
  5 MByte, 600 ms Zugriffszeit



de.wikibooks.org/wiki/Computerhardware_für_Anfänger

dritte Generation: ICs

- Erfindung der integrierten Schaltung 1958 (R. Noyce, J. Kilby)
- Dutzende... Hunderte... Tausende Transistoren auf einem Chip

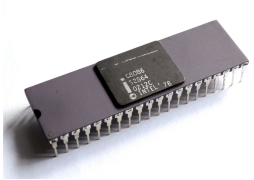


- IBM Serie-360: viele Maschinen, ein einheitlicher Befehlssatz
- volle Softwarekompatibilität

Eigenschaft	**Model 30**	**Model 40**	**Model 50**	**Model 65**
Rel. Leistung [Model 30]	1	3,5	10	21
Zykluszeit [ns]	1 000	625	500	250
Max. Speicher [KiB]	64	256	256	512
Pro Zyklus gelesene Byte	1	2	4	16
Max. Anzahl von Datenkanälen	3	3	4	6

vierte Generation: VLSI

- VLSI = *Very Large Scale Integration*
- ab 10 000 Transistoren pro Chip

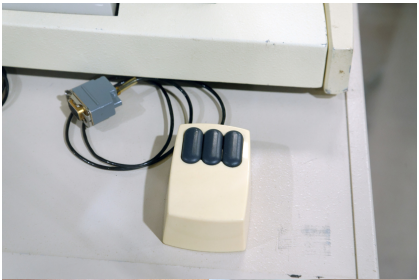


- gesamter Prozessor passt auf einen Chip
- steigende Integrationsdichte erlaubt immer mehr Funktionen

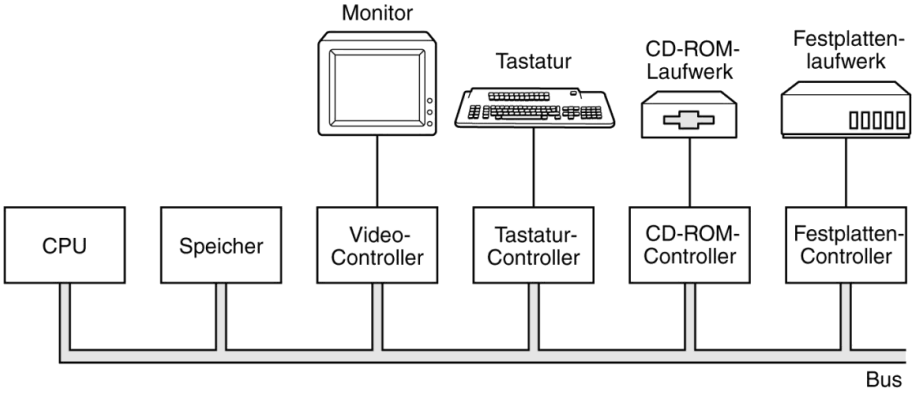
1972 Intel 4004: erster Mikroprozessor
1975 Intel 8080, Motorola 6800, MOS 6502 ...
1981 IBM PC („personal computer“) mit Intel 8088
...

- Massenfertigung erlaubt billige Prozessoren (< 1\$)
- Miniaturisierung ermöglicht mobile Geräte

Xerox Alto: first workstation (1973)



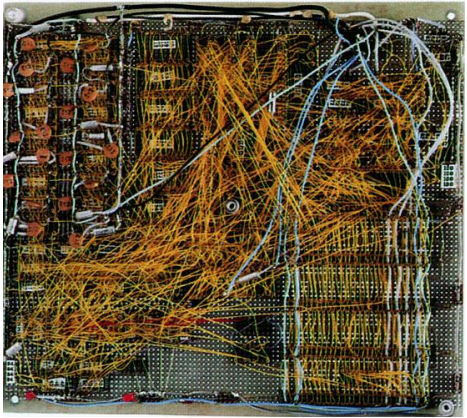
Personal Computer: Aufbau des IBM PC (1981)

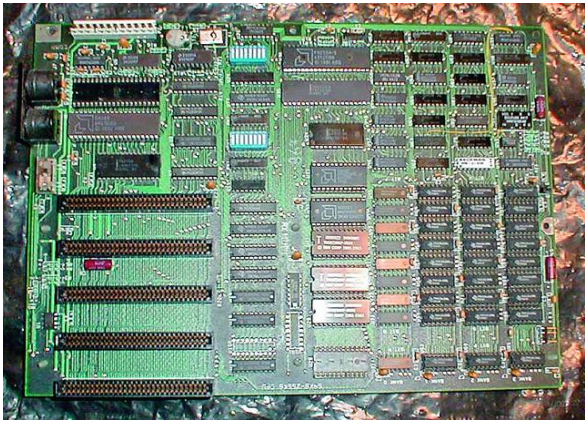


[6]

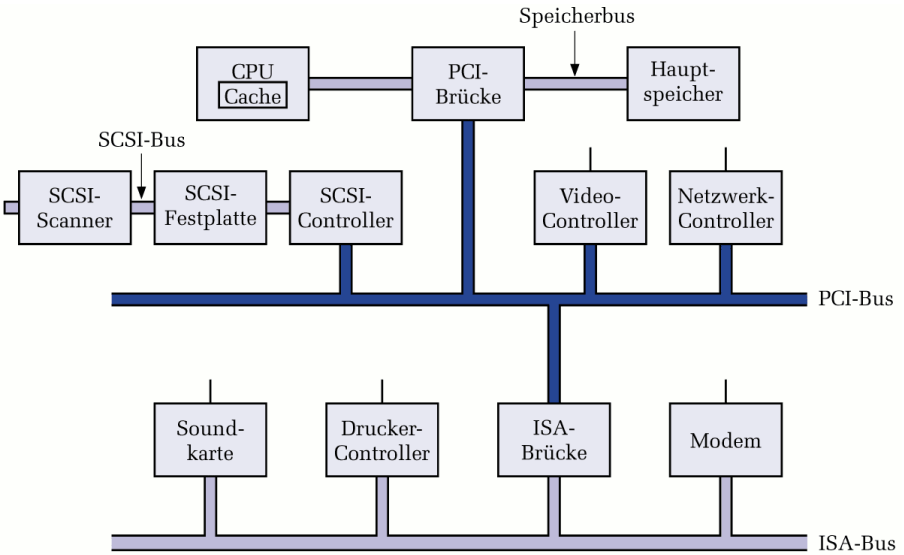
- Intel 8086/8088, 512 KByte RAM, Betriebssystem MS-DOS
- alle Komponenten über den zentralen (ISA-) Bus verbunden
- Erweiterung über Einsteckkarten

PC Prototyp (1981) und Hauptplatine



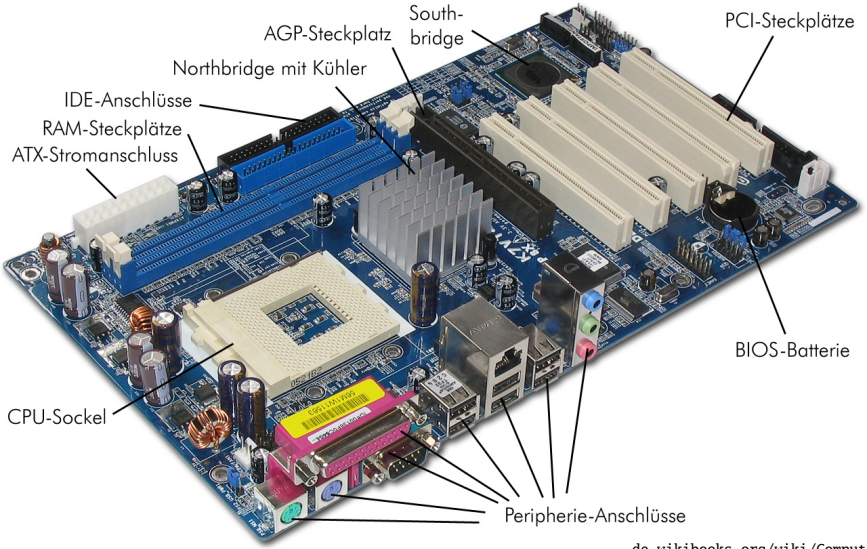


Aufbau mit PCI-Bus (2000)



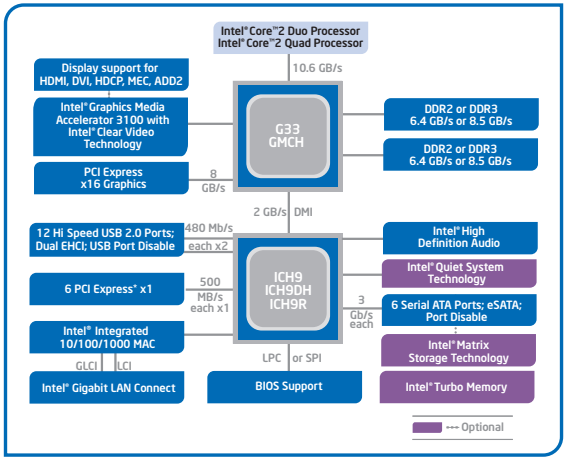
[6]

Hauptplatine (2005)



de.wikibooks.org/wiki/Computerhardware_für_Anfänger

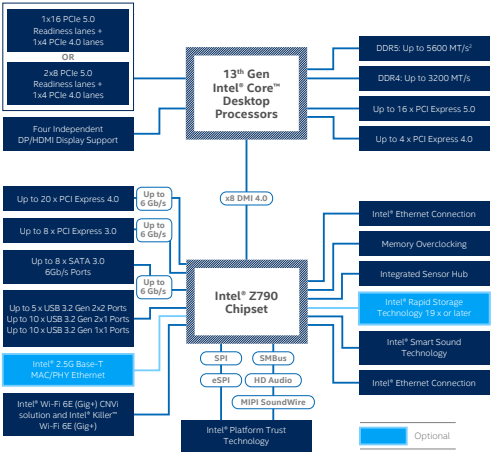
Aufbau (2010)



Intel ark.intel.com

- Mehrkern-Prozessoren („dual-/quad-/octa-core“)
- schnelle serielle Direktverbindungen statt PCI/ISA Bus

Aufbau (2022)



Intel ark.intel.com

- Speichercontroller und externe Anbindung (PCI Express) in CPU
- Grafikprozessor in CPU

- Anzahl an Systemen / Prozessoren – weltweit

System	Anzahl (geschätzt!)
PCs, Workstation, Server	2 Milliarden
Tablets	1,3 Milliarden
Smartphones	4,8 Milliarden
„Embedded Systems“	75-100 Milliarden

- Preis des Prozessors

Typ	Preis [$]	Beispielanwendung
Wegwerfcomputer	0,5	Glückwunschkarten
Mikrocontroller	5	Uhren, Geräte, Autos
Mobile Computer und Spielkonsolen	50	Smartphones, Tablets, Heimvideospiele
Personalcomputer	500	Desktop- oder Notebook-Computer
Server	5 000	Netzwerkserver
Workstation Verbund	50 000 – 500 000	Abteilungsrechner (Minisupercomp.)
Großrechner (Mainframe)	5 Millionen	Batch-Verarbeitung in einer Bank
Supercomputer	$>$ 50 Millionen	Klimamodelle, Simulationen

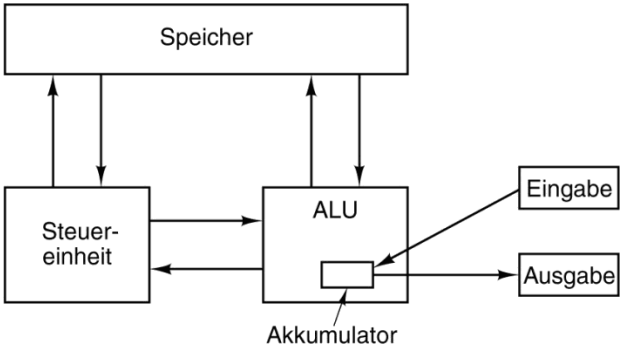
von-Neumann Konzept

- J. Mauchly, J.P. Eckert, J. von-Neumann 1945
- Abstrakte Maschine mit minimalem Hardwareaufwand
  - System mit Prozessor, Speicher, Peripheriegeräten
  - die Struktur ist unabhängig von dem Problem, das Problem wird durch austauschbaren Speicherinhalt (Programm) beschrieben
- gemeinsamer Speicher für Programme und Daten
  - fortlaufend adressiert
  - Programme können wie Daten manipuliert werden
  - Daten können als Programm ausgeführt werden
- Befehlszyklus: Befehl holen, decodieren, ausführen

$\Rightarrow$ enorm flexibel

- **alle** aktuellen Rechner basieren auf diesem Prinzip
- aber vielfältige Architekturvarianten, Befehlssätze usw.

von-Neumann Rechner

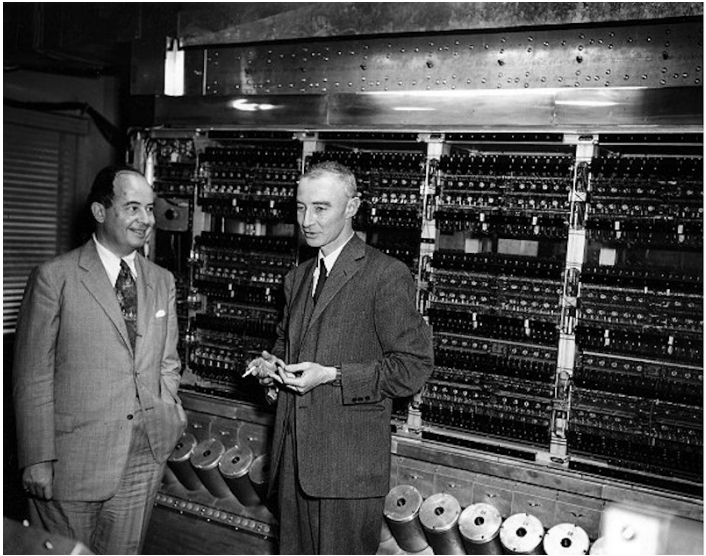


[6]

Fünf zentrale Komponenten:

- Prozessor mit **Steuerwerk** und **Rechenwerk** (ALU, Register)
- **Speicher**, gemeinsam genutzt für Programme und Daten
- **Eingabe**- und **Ausgabewerke**
- verbunden durch Bussystem

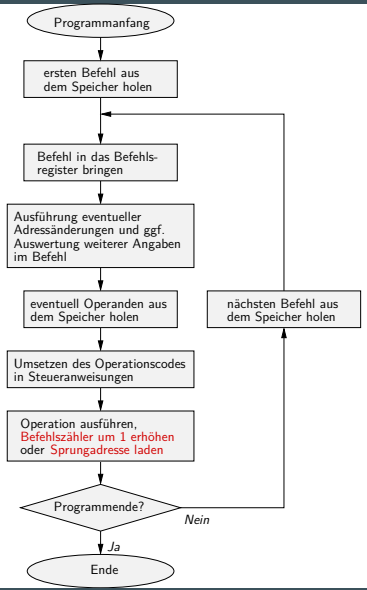
von-Neumann Rechner: IAS Computer



John von Neumann, R. J. Oppenheimer,
IAS Computer Princeton,
5kB Speicher (1024x40)
40-bit Worte
10-bit Befehle und Adressen
`computerhistory.org`

Programmverarbeitung

- Programm als Sequenz elementarer Anweisungen (Befehle)
- als Bitvektoren im Speicher codiert
- Interpretation (Operanden, Befehle und Adressen) ergibt sich aus dem Kontext (der Adresse)
- zeitsequenzielle Ausführung der Instruktionen



Beispiel: PRIMA (die primitive Maschine)

- Hauptspeicher mit 256 Worten a 8-bit
- Prozessor mit vier Registern a 8-bit
  - PC: Program Counter — Adresse für Befehl holen
  - BR: Befehlsregister — aktueller Befehl
  - AR: Adressregister — Adresse für Datenoperationen
  - AKKU: Akkumulator — logische/arith. Operation
- ALU: Arithmetic/Logical Unit — das Rechenwerk
- SW: Steuerwerk zur Ablaufsteuerung
- Input/Output:
  - SW: ein Schalter, direkt ans Steuerwerk angeschlossen

PRIMA: Befehlszyklus

- externes Taktsignal, z.B. 1 MHz
- jeder Befehl benötigt drei Takte: erst den Befehl holen, dann die Datenadresse holen, dann Befehl ausführen:

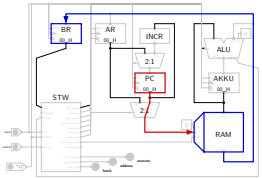
1. BR := MEM[PC]	Befehl holen
PC := PC + 1	und PC um 1 erhöhen
2. AR := MEM[PC]	Adresse holen
PC := PC + 1	und PC um 1 erhöhen
3. switch(BR):	Befehl dekodieren
case 0: break; i	no operation
case 1: AKKU := 0	AKKU löschen
case 2: AKKU := MEM[AR]	Daten laden
...	
case 255: ...	

PRIMA: Befehlssatz (Ausschnitt)

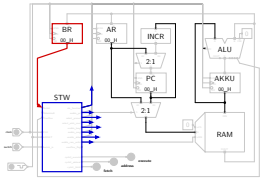
Wert	Mnemonic	Operation
0	**nop**	no operation, do nothing
1	clear	AKKU := 0
2	load	AKKU := MEM[AR]
3	store	MEM[AR] := AKKU
16	incr	AKKU := AKKU + 1
17	decr	AKKU := AKKU - 1
18	**add**	AKKU := AKKU + MEM[AR]
19	**sub**	AKKU := AKKU - MEM[AR]
32	**neg**	AKKU := ~ AKKU
33	**and**	AKKU := AKKU & MEM[AR]
34	**or**	AKKU := AKKU | MEM[AR]
64	**jmp**	PC := AR
65	beq	**if** (AKKU == 0) PC := AR
66	bgt	**if** (AKKU >= 0) PC := AR
255	halt	machine stop

- Verschaltung der Hardwarekomponenten für alle nötigen Datentransfers
- abhängig vom Befehl werden nur bestimmte Pfade aktiv
- Ausführungszyklus

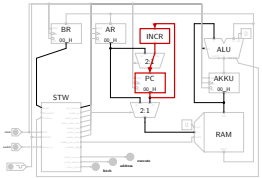
Befehl holen

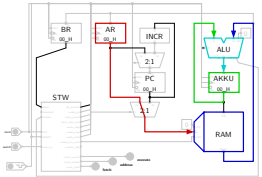


decodieren

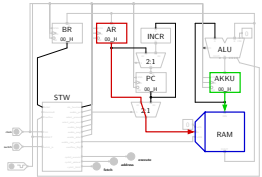


PC inkrementieren

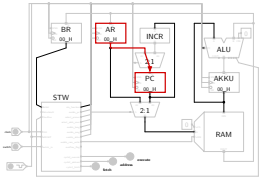




rechnen



speichern

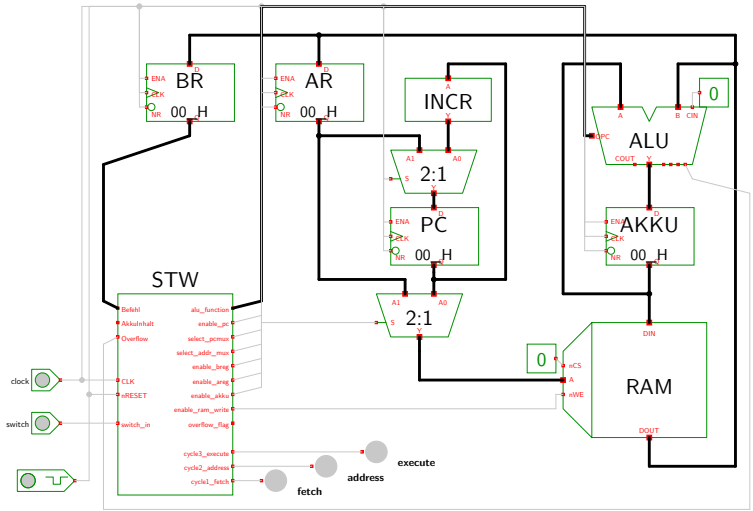


springen

Beispiel: PRIMA (die primitive Maschine)

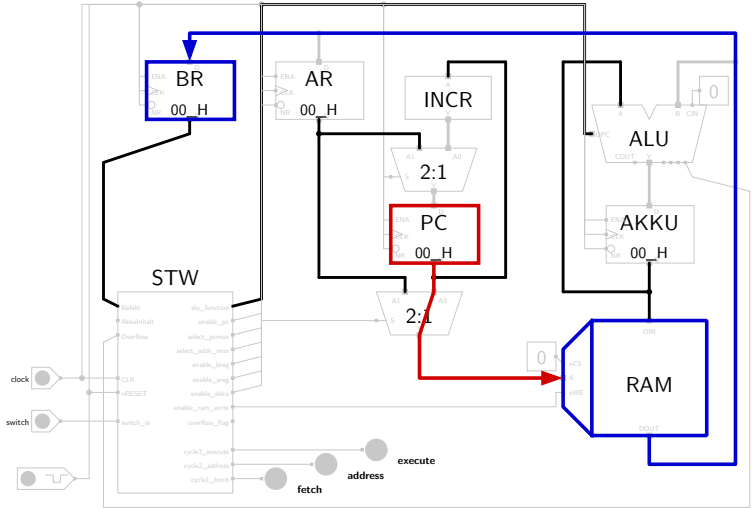
- ein (minimaler) 8-bit von-Neumann Rechner

[2] Hades Demo: `50-rtlib/90-prima/prima`



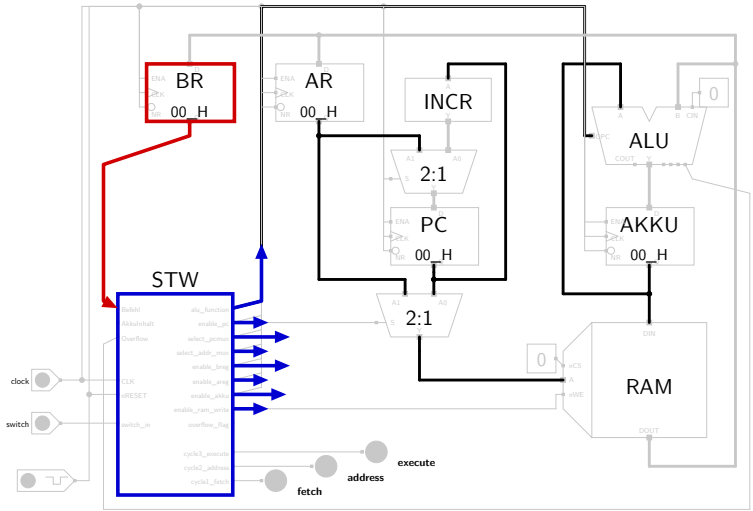
PRIMA: Befehl holen

- BR = RAM[PC]



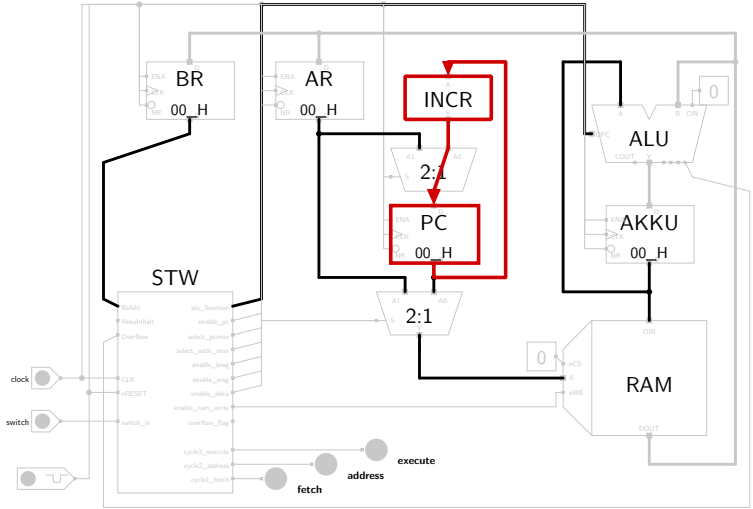
PRIMA: decodieren

- Steuersignale = decode(BR)

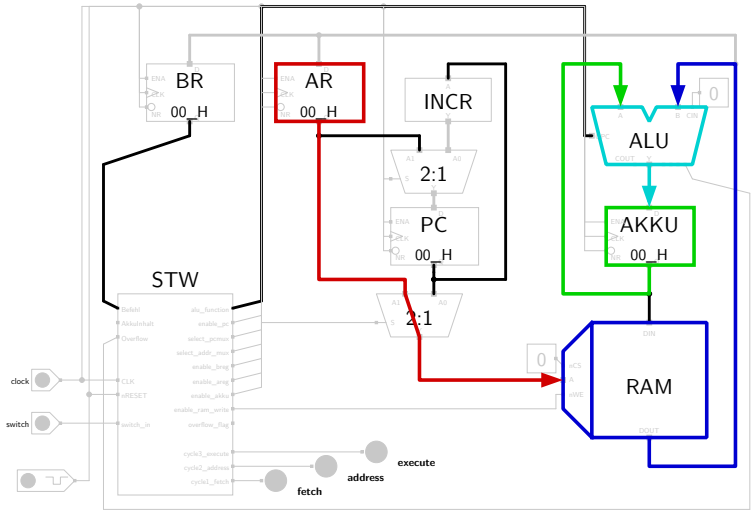


PRIMA: PC inkrementieren

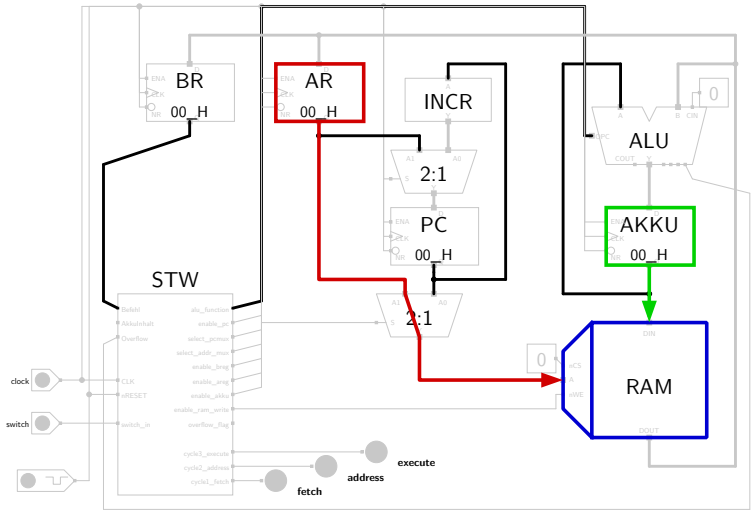
- PC = PC+1



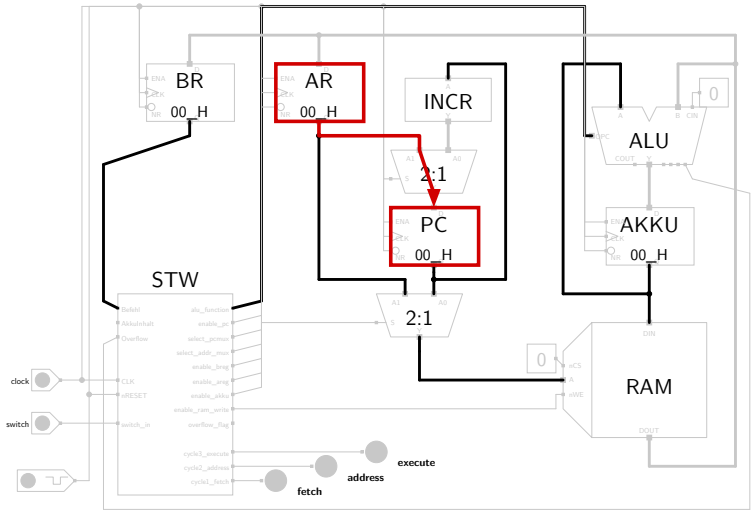
- Akku = Akku + RAM[AR]



- RAM[AR] = Akku

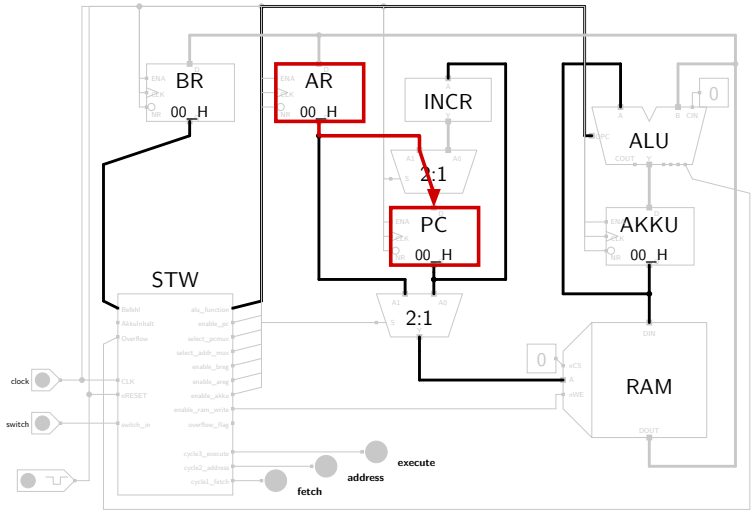

BR
00_H
AR
00_H
INCR
2:1
PC
00_H
2:1
ALU
AKKU
00_H
STW
RAM
clock
switch
execute
address
fetch

- PC = AR


BR
00_H
AR
00_H
INCR
ALU
2:1
PC
00_H
AKKU
00_H
STW
2:1
RAM
clock
switch
fetch
address
execute

- PC = AR

später dazu mehr...



PRIMA: Maschinencode für Zähler in Endlosschleife


```
Label    Adresse   Speicher   Mnemo      Kommentar
start:         0          1   clear      akku = 0
               1          *              unbenutzt
               2          3   store      MEM[253] = 0
               3        253
loop:          4         16   incr       akku = akku + 1
               5          *              unbenutzt
               6          3   store
               7        253              MEM[253] = akku
               8         64   jump
               9          4              jump loop
             ...
counter:     253        <n>
             ...
```

PRIMA: Maschinencode für Multiplikation (etc.)

- Übungsaufgabe

- mit Papier und Bleistift
- mit virtueller Maschine (Simulator)
- mit echter Hardware

PRIMA: Interpreter als Virtuelle Maschine

```
uint8_t MEM[256] = { 1,0,2,253,16,0,3,254,64,4 };
uint8_t PC, AR, BR;      // control/address registers
uint8_t AKKU;              // accumulator (w/o carry)

int main( int argc, char** argv ) { // simulate PRIMA
  PC = 0;            // execution starts at address 0
  while(BR != 255) {
    BR = MEM[ PC ]; PC = PC + 1;
    AR = MEM[ PC ]; PC = PC + 1;
    switch( BR ) {
      case 0: break;                          // nop
      case 1: AKKU = 0; break;               // clear
      case 2: AKKU = MEM[ AR ]; break;        // load
      case 3: MEM[ AR ] = AKKU; break;       // store
      ...
      case 255: break;                        // halt
    }
  }
}
```

PRIMA: Offensichtliche Probleme

- 8-bit Wortbreite: ein bisserl knapp
- 256 Speicheradressen: definitiv viel zu wenig

- selbst einfache Operationen dauern viele Takte
- Befehlssatz: zusätzliche Befehle wären nett
- nur absolute Adressen: sehr unbequem zu programmieren
- Funktionen / Unterprogramme?
- „Semantic Gap“

- Startup: wie wird der Speicher initialisiert?
- Programm schrittweise eingeben? Programm extern laden?

Moore's Law

- bessere Technologie ermöglicht immer kleinere Transistoren
- Kosten (Material/Entwurf/Fertigung) sind proportional zur Chipfläche

$\Rightarrow$ bei gleicher Funktion kleinere und billigere Chips

$\Rightarrow$ bei gleicher Größe leistungsfähigere Chips

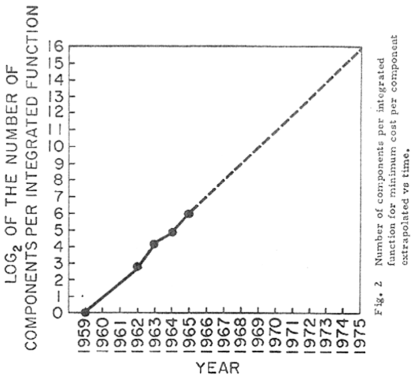
Moore's Law — Gordon Moore, Mitgründer von Intel, 1965

Speicherkapazität von ICs vervierfacht sich alle drei Jahre

$\Rightarrow$ schnelles exponentielles Wachstum

- klares Kostenoptimum bei hoher Integrationsdichte
- trifft auch auf Prozessoren zu

Moore's Law (cont.)

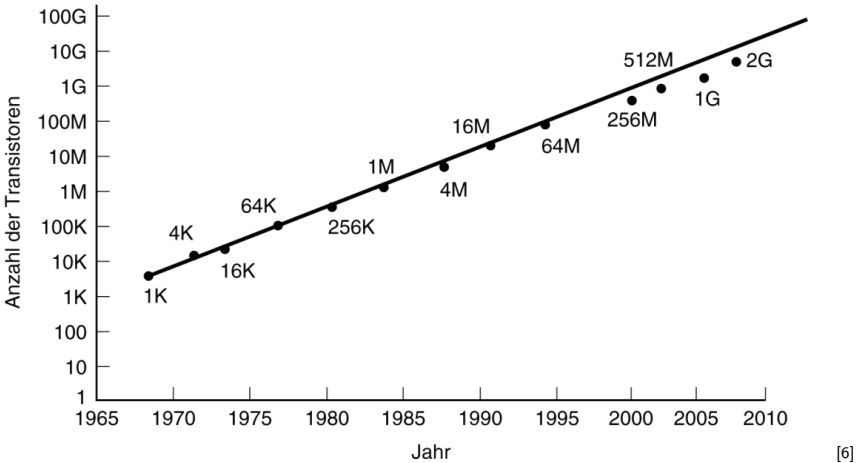


Gordon Moore, 1965, [4]:
Cramming more components onto integrated circuits

Wird das so weitergehen?

- Vorhersage gilt immer noch
- „*IRDS*“ Prognosen bis zum Jahr 2037 [**?**]

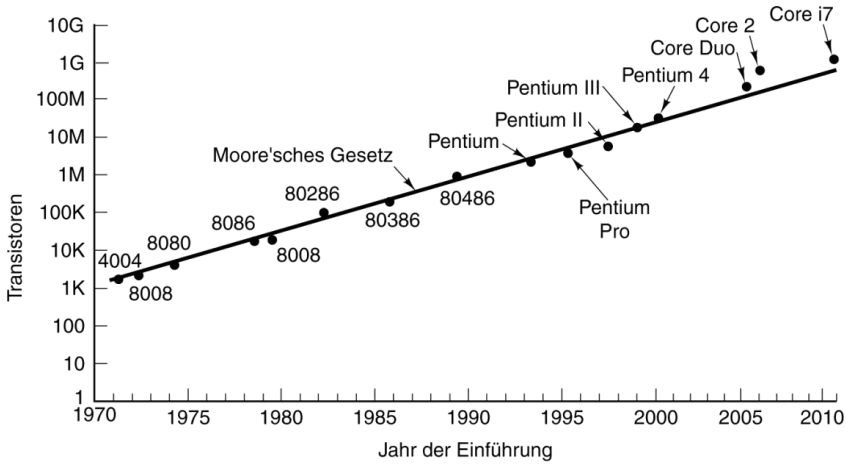
Moore's Law: Transistoren pro Speicherchip



[6]

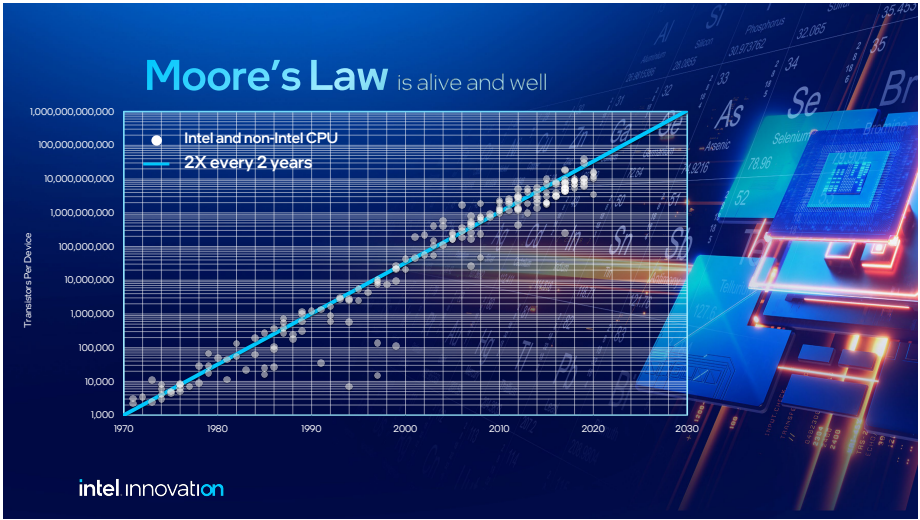
- Vorhersage: 60% jährliches Wachstum der Transistoranzahl pro IC
  Verdopplung alle 18 Monate (12...24 Monate)

Moore's Law: Evolution der Prozessoren



[6]

Moore's Law: Evolution der Prozessoren (cont.)



[3] Intel Innovation 2023

Moore's Law: Evolution der Prozessoren (cont.)

Transistoren pro IC (monolithisch)

Modell		Typ	Jahr	Trans. [Mrd.]
M2 Max	Apple	CPU	2023	67,0

Moore's Law: Evolution der Prozessoren (cont.)

Transistoren pro IC (monolithisch)

Modell		Typ	Jahr	Trans. [Mrd.]
M2 Max	Apple	CPU	2023	67,0
A17	Apple	SOC	2023	19,0

Moore's Law: Evolution der Prozessoren (cont.)

Transistoren pro IC (monolithisch)

Modell		Typ	Jahr	Trans. [Mrd.]
M2 Max	Apple	CPU	2023	67,0
A17	Apple	SOC	2023	19,0
MI300X	AMD	GPU	2023	153,0
GB200	Nvidia	GPU	2024	208,0

Moore's Law: Evolution der Prozessoren (cont.)

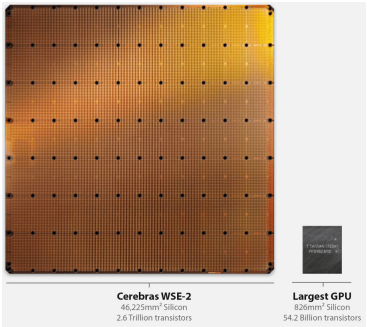
Transistoren pro IC (monolithisch)

Modell		Typ	Jahr	Trans. [Mrd.]
M2 Max	Apple	CPU	2023	67,0
A17	Apple	SOC	2023	19,0
MI300X	AMD	GPU	2023	153,0
GB200	Nvidia	GPU	2024	208,0
Stratix 10	Intel (Altera)	FPGA	2019	43,3
VP 1802	AMD (Xilinx)	FPGA	2021	92,0

Moore's Law: Evolution der Prozessoren (cont.)

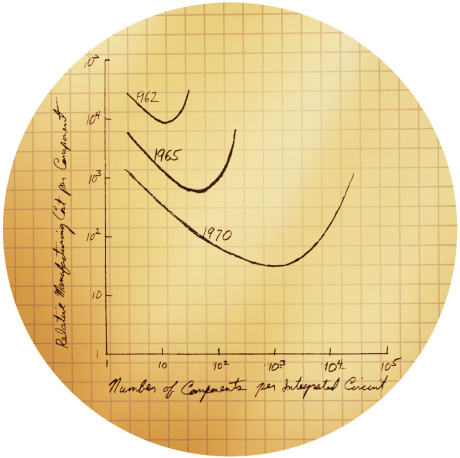
Transistoren pro IC (monolithisch)

Modell		Typ	Jahr	Trans. [Mrd.]
M2 Max	Apple	CPU	2023	67,0
A17	Apple	SOC	2023	19,0
MI300X	AMD	GPU	2023	153,0
GB200	Nvidia	GPU	2024	208,0
Stratix 10	Intel (Altera)	FPGA	2019	43,3
VP 1802	AMD (Xilinx)	FPGA	2021	92,0
WSE-2	Cerebras	WSc	2021	2 600,0



[www.cerebras.net]

Moore's Law: Kosten pro Komponente



Originalskizze von G. Moore [3]

Moore's Law: Formel und Beispiele

$$L(t) = L(0) \cdot 2^{t/18}$$

mit: $L(t)$ = Leistung zum Zeitpunkt t,
$L(0)$ = Leistung zum Zeitpunkt 0,
und Zeit t in Monaten.

Einige Formelwerte:

Jahr 1:	1,5874
Jahr 2:	2,51984
Jahr 3:	4
Jahr 5:	10,0794
Jahr 6:	16
Jahr 7:	25,3984
Jahr 8:	40,3175

Leistungssteigerung der Spitzenrechner seit 1993

www.top500.org de.wikipedia.org/wiki/Supercomputer

Jahr	Rechner	CPU	Linpack [TFlop/s]	Prozessoren
1993	TMC CM-5/1024	(SuperSparc 32MHz)	0,0597	1 024
1994	Intel XP/S140	(80860 50MHz)	0,1434	3 680
1995	Fujitsu NWT	(105 MHz)	0,17	140
1996	Hitachi SR2201/1024	(HARP-1E 120MHz)	0,2204	1 024
1997	Intel ASCI Red	(Pentium Pro 200MHz)	1,068	7 264
1999	Intel ASCI Red	(Pentium Pro 333MHz)	2,121	9 472
2001	IBM ASCI White	(Power3 375MHz)	7,226	8 192
2002	NEC Earth Simulator	(NEC 1GHz)	35,86	5 120
2005	IBM BlueGene/L	(PowerPC 440 2C 700MHz)	136,8	65 536
2006	IBM BlueGene/L	(PowerPC 440 2C 700MHz)	280,6	131 072
2008	IBM Roadrunner	(Opteron 2C 1,8GHz + IBM Cell 9C 3,2 GHz)	1 026,0	122 400
2010	Cray XT5-HE Jaguar	(Opteron 6C 2,6GHz)	1 759,0	224 162
2011	Fujitsu K computer	(SPARC64 VIIIfx 2.0GHz)	8 162,0	548 352
2012	IBM BlueGene/Q Sequoia	(Power BQC 16C 1,6GHz)	16 324,8	1 572 864
2013	NUDT Tianhe-2	(Xeon E5-2692 12C 2,2 GHz + Phi 31S1P)	33 862,7	3 120 000
2016	Sunway TaihuLight	(Sunway SW26010 260C 1,45 GHz)	93 014,6	10 649 600
2018	Summit	(IBM Power 9 22C 3,07 GHz + NVIDIA GV100)	143 500,0	2 397 824
2021	Fugaku	(Fujitsu A64FX 48C 2,2 GHz)	442 010,0	7 630 848
2024	JUWELS [D #21]	(Epyc 7402 24C 2,8 GHz + NVIDIA A100)	44 120,0	449 280
2024	Frontier	(AMD Epyc 64C 2,0 GHz + Instinct)	1 206 000,0	8 699 904

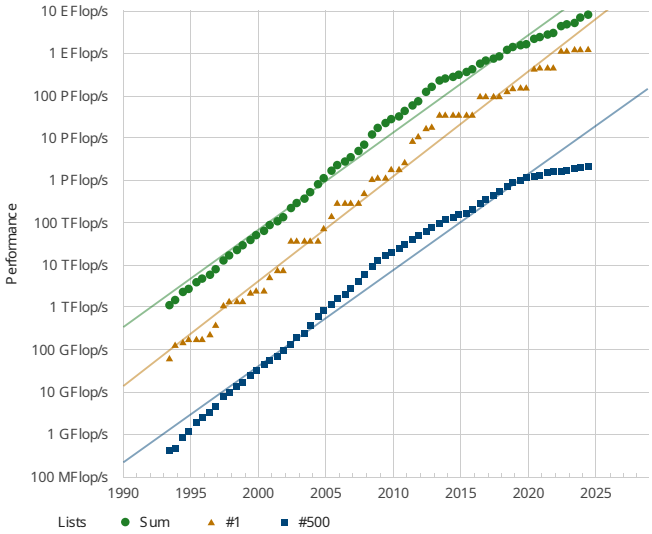
Leistungssteigerung der Spitzenrechner seit 1993

www.top500.org de.wikipedia.org/wiki/Supercomputer

Jahr	Rechner	CPU	Linpack [TFlop/s]	Prozessoren	Power [KW]
1993	TMC CM-5/1024	(SuperSparc 32MHz)	0,0597	1 024	
1994	Intel XP/S140	(80860 50MHz)	0,1434	3 680	
1995	Fujitsu NWT	(105 MHz)	0,17	140	
1996	Hitachi SR2201/1024	(HARP-1E 120MHz)	0,2204	1 024	
1997	Intel ASCI Red	(Pentium Pro 200MHz)	1,068	7 264	
1999	Intel ASCI Red	(Pentium Pro 333MHz)	2,121	9 472	
2001	IBM ASCI White	(Power3 375MHz)	7,226	8 192	
2002	NEC Earth Simulator	(NEC 1GHz)	35,86	5 120	3 200
2005	IBM BlueGene/L	(PowerPC 440 2C 700MHz)	136,8	65 536	716
2006	IBM BlueGene/L	(PowerPC 440 2C 700MHz)	280,6	131 072	1 433
2008	IBM Roadrunner	(Opteron 2C 1,8GHz + IBM Cell 9C 3,2 GHz)	1 026,0	122 400	2 345
2010	Cray XT5-HE Jaguar	(Opteron 6C 2,6GHz)	1 759,0	224 162	6 950
2011	Fujitsu K computer	(SPARC64 VIIIfx 2.0GHz)	8 162,0	548 352	9 899
2012	IBM BlueGene/Q Sequoia	(Power BQC 16C 1,6GHz)	16 324,8	1 572 864	7 890
2013	NUDT Tianhe-2	(Xeon E5-2692 12C 2,2 GHz + Phi 31S1P)	33 862,7	3 120 000	17 808
2016	Sunway TaihuLight	(Sunway SW26010 260C 1,45 GHz)	93 014,6	10 649 600	15 371
2018	Summit	(IBM Power 9 22C 3,07 GHz + NVIDIA GV100)	143 500,0	2 397 824	9 783
2021	Fugaku	(Fujitsu A64FX 48C 2,2 GHz)	442 010,0	7 630 848	29 899
2024	JUWELS [D #21]	(Epyc 7402 24C 2,8 GHz + NVIDIA A100)	44 120,0	449 280	1 764
2024	Frontier	(AMD Epyc 64C 2,0 GHz + Instinct)	1 206 000,0	8 699 904	22 786

Leistungssteigerung der Spitzenrechner seit 1993 (cont.)

www.top500.org de.wikipedia.org/wiki/Supercomputer



Moore's Law: Aktuelle Trends

- Miniaturisierung schreitet weiter fort
- Taktraten physikalisch limitiert
  - Technologie / Strukturgrößen
  - Leistungsaufnahme $\Rightarrow$ Spannungsversorgung + Kühlung

Entwicklungen

- seit 2011: CPU plus Grafikeinheit
- Integration mehrerer CPUs auf einem Chip (2-...128-Cores)
- Cache Speicher (SRAM als schneller Zwischenspeicher) auf dem Die
- Integration von Peripheriegeräten (Speicherinterface, PCIe, ...)
- heterogene Architekturen: „performance“ / „efficiency“
- multi-Chip Module (CPU, IO, Cache) + 3D Stapel
- **SoC**: „System on a chip“

SoC: System on a chip

Gesamtes System auf einem Chip integriert:

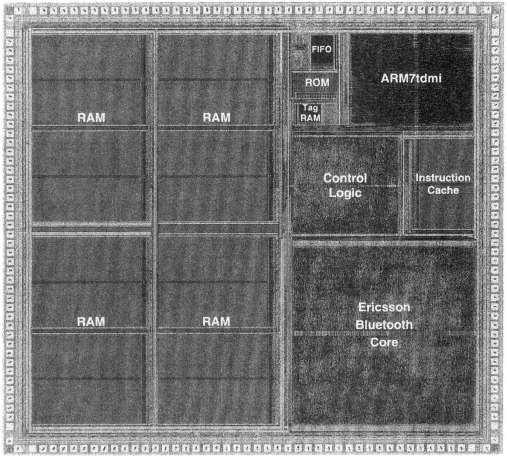
- ein oder mehrere Prozessoren, z.T. verschiedene Typen
  - hohe Rechenleistung
  - energieeffizient

  $\Rightarrow$ z.B. ARM mit *big.LITTLE* Konzept; aktuell Laptop, PC: Intel, AMD
- Cache Hierarchie: 1st Level (D+I pro CPU) / 2nd (pro CPU) / 3rd (alle Kerne)
- dedizierte Prozessoren: Grafik, Video(de)codierung, DSP, AI ...
- Peripherieschnittstellen: Bussysteme, Speichercontroller
- Hauptspeicher (speziell „Embedded“), Grafikspeicher
- weitere Speicher für Medien/Netzwerkoperationen

SoC: System on a chip (cont.)

- Peripherieblöcke nach Kundenwunsch konfiguriert:
  - Displayansteuerung: DP, HDMI, USB-C ...
  - A/V-Schnittstellen: Kamera, Mikrofone, Audio ...
  - serielle und parallele Schnittstellen, SPI, I/O-Pins ...
  - Feldbusse: I^2C, CAN ...
  - PC-like: USB, Firewire, SATA ...
  - Netzwerk kabelgebunden (Ethernet)
  - Funkschnittstellen: WLAN, Bluetooth, 5G ...
- Smartphones, Tablet-Computer, Medien-/DVD-Player, WLAN-Router, NAS-/Home-Server ...

- Bluetooth-Controller (2000)

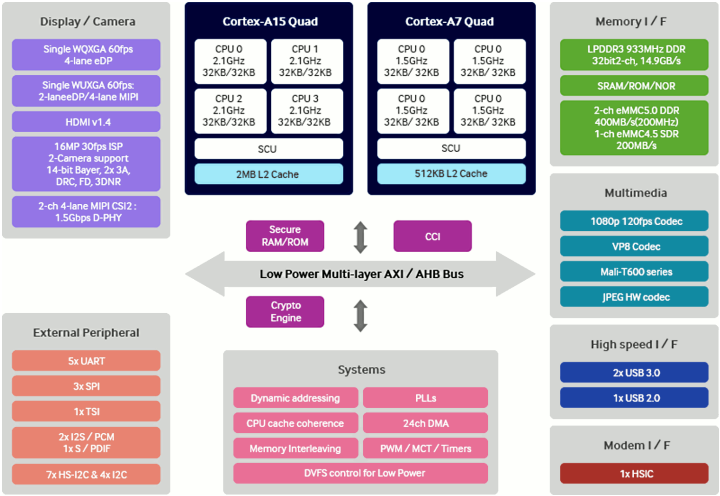


[1]

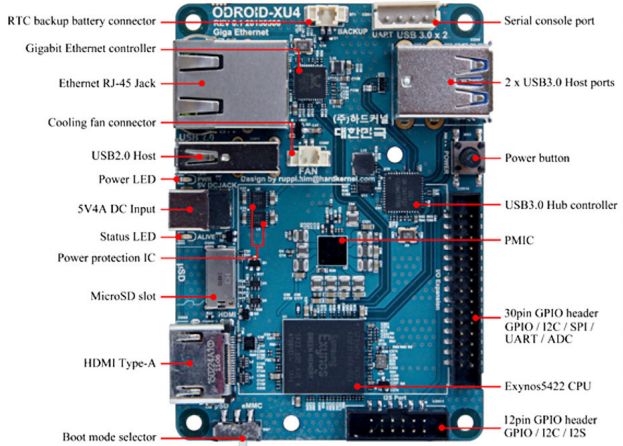
Prozess	0,25 μm
Metall	3-Layer
V_{DD}	2,5 V
Transistoren	4,3 Mill.
Chipfläche	20 mm^2
Taktrate	0 . . . 13 MHz
MIPS	12
Power	75 mW
MIPS/W	160

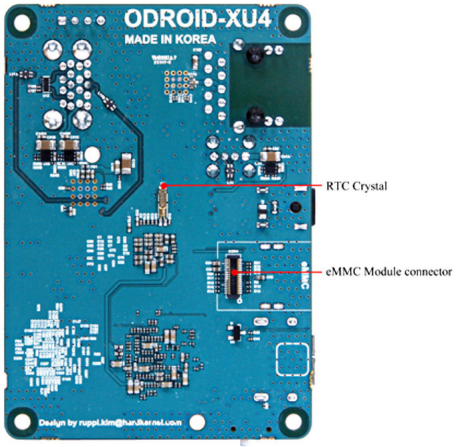
SoC Beispiele (cont.)

- Samsung Exynos-5422 (2014) [5]



- Beispiel: Odroid XU4 (2015) [?]
  - vollständiger Mikrocomputer (32-bit): 4P-Kerne 2,1 GHz + 4E-Kerne 1,4 GHz
  - Betriebssystem: Android oder Linux



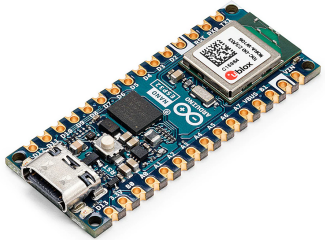


aktuelle Mikrorechner

ArduinoNano ESP 32 (2023)

32-bit kein Betriebssystem
2-Kerne ≤ 240 MHz
512 KB SRAM
16 MB Flash

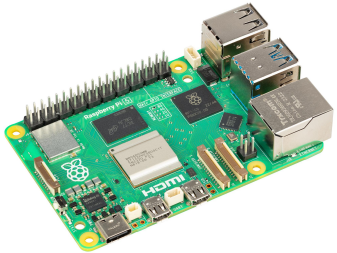
Wi-Fi, Bluetooth, USB
22 €



[?]

Raspberry Pi 5 (2023)

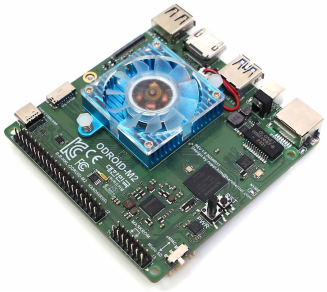
64-bit Linux (+spezialisiert), Windows
4-Kerne $\leq 2{,}4$ GHz
2/4/8 GB SDRAM, L1...L3 Cache
microSD Karte
GPU, 4Kp60 HEVC Decoder, PCIe
2 MIPI cam./display, 2 HDMI, Ethernet,
Wi-Fi, Bluetooth, USB
53/66/87 €



[?]

Odroid M2 (2024)

64-bit Linux (+spezialisiert), Android
4P-Kerne $\leq 2{,}3$ GHz + 4E-Kerne 1,8 GHz
8/16 GB SDRAM, L1...L3 Cache
64GB eMMC, M.2 SSD, microSD Karte
NPU, GPU, 4Kp60 HEVC Decoder, PCIe
MIPI cam./display, HDMI, DP, Ethernet,
USB
115/145 $



[?]

Grenzen des Wachstums

- Jeder exponentielle Verlauf stößt irgendwann an natürliche oder wirtschaftliche Grenzen
- Beispiel: physikalische Limits
  - Eine DRAM-Speicherzelle speichert etwa 200 Elektronen (2012)
    Skalierung: es werden mit jeder neuen Technologiestufe weniger
  - Offensichtlich ist die Grenze spätestens dann erreicht, wenn nur noch ein einziges Elektron für 1-bit gespeichert wird
  - Ab diesem Zeitpunkt gibt es bessere Performanz nur noch durch bessere Algorithmen / Architekturen!

$\Rightarrow$ Annahme: 50 % Skalierung pro Jahr, 200 Elektronen/Speicherzelle
gesucht: $x \widehat{=}$ Jahre Fortschritt

$\Rightarrow$ $200/(1{,}5^x) \geq 1$

$x = \ln(200)/\ln(1{,}5) \approx 13$ Jahre

$a^b = \exp(b \cdot \ln a)$

Roadmap: IRDS

IEEE **I**nternational **R**oadmap for **D**evices and **S**ystems
`irds.ieee.org/editions/2024`

- IEEE: Institute of Electrical and Electronics Engineers
- Beteiligung von
  - Halbleiterherstellern
  - Geräte-Herstellern
  - Universitäten und Forschungsinstituten
  - Fachverbänden aus USA, Europa, Asien
- Publikation von langjährigen Vorhersagen
- Zukünftige Entwicklung der Halbleitertechnologie
- Prognosen zu Fertigungsprozessen, Modellierung, Simulation, Entwurf etc.
- für Chips (Speicher, Prozessoren, SoC ...) und Systeme
- neue Technologien: Quantencomputing etc.

Roadmap: IRDS (cont.)

Table MM01 – More Moore – Logic Core Device Technology Roadmap (Ausschnitt, 2017)

YEAR OF PRODUCTION	2017	2019	2021	2024	2027	2030	2033
	P54M36	P48M28	P42M24	P36M21	P32M14	P32M14T2	P32M14T4
Logic industry "Node Range" Labeling (nm)	"10"	"7"	"5"	"3"	"2.1"	"1.5"	"1.0"
IDM-Foundry node labeling	i10-f7	i7-f5	i5-f3	i3-f2.1	i2.1-f1.5	i1.5-f1.0	i1.0-f0.7
Logic device structure options	finFET FDSOI	finFET LGAA	LGAA finFET	LGAA VGAA	LGAA VGAA	VGAA, LGAA 3DVLSI	VGAA, LGAA 3DVLSI
Logic device mainstream device	finFET	finFET	LGAA	LGAA	LGAA	VGAA	VGAA
DEVICE STRUCTURES							
							
LOGIC TECHNOLOGY ANCHORS							
Patterning technology inflection for Mx interconnect	193i, EUV	193i, EUV DP	193i, EUV DP	193i, High-NA EUV	193i, High-NA EUV+(DSA)	193i, High-NA EUV+(DSA)	193i, High-NA EUV+(DSA)
Channel material technology inflection	Si	SiGe25%	SiGe50%	Ge, IIIV (TFET?), 2D Mat	Ge, IIIV (TFET?), 2D Mat	Ge, IIIV (TFET?), 2D Mat	Ge, IIIV (TFET?), 2D Mat
Process technogy inflection	Conformal deposition	Conformal Doping, Contact	Channel, RMG	Stacked-device Non-Cu Mx	Stacked-device Non-Cu Mx	Steep-SS, 3D	Steep-SS, 3D
Stacking generation inflection	2D	2D	3D-stacking: W2W D2W	3D-device: P-over-N Hetero	3D-device: Mem-on-Logic Hetero	3D-device: Mem-on-Logic Hetero	3D-device: Logic-on-Logic Hetero
LOGIC TECHNOLOGY INTEGRATION CAPACITY							
Design scaling factor for standard cell	-	0,98	1,09	0,96	1,03	2,00	1,00
Design scaling factor for SRAM (111) bitcell	-	1,00	1,00	1,00	1,00	1,25	1,00
▪ ▪ ▪							
POWER AND PERFORMANCE SCALING FACTORS							
Vdd (V)	0,75	0,70	0,65	0,65	0,65	0,60	0,55
Physical gate length for HP Logic (nm)	20,0	18,0	16,0	14,0	12,0	12,0	12,0
Datapath speed improvement at Vdd - relative	1,00	1,19	1,21	1,34	1,56	1,60	1,70
▪ ▪ ▪							
Power density of logic path cube at fmax - relative	1,00	1,20	1,21	1,82	2,69	4,49	8,00
fmax of a single CPU core at Vdd (GHz)	2,5	3,0	3,0	3,3	3,9	4,0	4,2
favg at constant power density and Vdd (GHz)	2,50	2,48	2,51	1,84	1,45	0,89	0,53
CPU SiP throughput at fmax (TFLOPS/sec)	0,16	0,27	0,46	0,79	1,34	2,27	3,86
▪ ▪ ▪							
INTERCONNECT TECHNOLOGY							
Conductor	Cu, non-Cu	Cu, non-Cu	Cu, non-Cu	Cu, non-Cu	Cu, non-Cu	Cu, non-Cu	Cu, non-Cu
Number of wiring layers	14	16	18	20	20	20	20
▪ ▪ ▪							

Angenommen die Lösung einer Rechenaufgabe dauert derzeit vier Jahre und die Rechenleistung wächst jedes Jahr um 60 %.

Wie lösen wir das Problem ?

Angenommen die Lösung einer Rechenaufgabe dauert derzeit vier Jahre und die Rechenleistung wächst jedes Jahr um 60 %.

Ein mögliches Vorgehen ist dann das folgende:

- Wir warten drei Jahre, kaufen dann einen neuen Rechner und erledigen die Rechenaufgabe in einem Jahr.
- *Wie das ?*

Angenommen die Lösung einer Rechenaufgabe dauert derzeit vier Jahre und die Rechenleistung wächst jedes Jahr um 60 %.

Ein mögliches Vorgehen ist dann das folgende:

- Wir warten drei Jahre, kaufen dann einen neuen Rechner und erledigen die Rechenaufgabe in einem Jahr.

$\Rightarrow$ Nach einem Jahr können wir einen Rechner kaufen, der um den Faktor 1,6 Mal schneller ist, nach zwei Jahren bereits $1{,}6 \cdot 1{,}6$ Mal schneller, und nach drei Jahren (also am Beginn des vierten Jahres) gilt $(1 + 60\,\%)^3 = 4{,}096$.

- Wir sind also sogar ein bisschen schneller fertig, als wenn wir den jetzigen Rechner die ganze Zeit durchlaufen lassen.

Wie geht es jetzt weiter?

Ab jetzt erst mal ein *bottom-up* Vorgehen:

Start mit grundlegenden Aspekten

- Informationsverarbeitung und -repräsentation
- Darstellung von Zahlen und Zeichen
- arithmetische und logische Operationen
- Schaltnetze, Schaltwerke, endliche Automaten

dann Kennenlernen aller Basiskomponenten des Digitalrechners

- Gatter, Flipflops ...
- Register, ALU, Speicher ...

und Konstruktion eines Rechners (HW) mit seinen Betriebsmitteln (SW)

- Befehlssatz, -abarbeitung, Assembler
- Pipelining, Speicherhierarchie
- Prozesskontrolle, Locking, Interrupts, Scheduling
- virtueller Speicher, Dateisystem, Ein- / Ausgabe
- ...

Literatur

[1] Steve Furber.
ARM System-on-Chip Architecture.
Pearson Education Limited, Harlow, second edition, 2000.

[2] Norman Hendrich.
Hades — hamburg design system.
Lehrmaterial, Universität Hamburg, Fachbereich Informatik, AB TAMS.

[3] Santa Clara, CA.

[4] Gordon E. Moore.
Cramming more components onto integrated circuits.
Electronics, 38(8), April 19 1965.

[5] Suwon, Südkorea.

Literatur (cont.)

[6] Andrew S. Tanenbaum and Todd Austin.
Rechnerarchitektur – Von der digitalen Logik zum Parallelrechner.
Pearson Deutschland GmbH, Hallbergmoos, sixth edition, 2014.