64-040 Modul InfB-RS: Rechnerstrukturen

https://tams.informatik.uni-hamburg.de/lectures/2015ws/vorlesung/rs

– Kapitel 1 –

Norman Hendrich

Universität Hamburg
Fakultät für Mathematik, Informatik und Naturwissenschaften
Fachbereich Informatik

Technische Aspekte Multimodaler Systeme

Wintersemester 2015/2016

Kapitel 1

Informatik

Brockhaus-Enzyklopädie: „*Informatik*“

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

Informatik

Brockhaus-Enzyklopädie: „*Informatik*“

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

Informatik

Brockhaus-Enzyklopädie: „*Informatik*“

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

Thema in Rechnerstrukturen: *Wie funktioniert ein Digitalrechner?*

- Wie wird Information (Zahlen, Zeichen) repräsentiert / codiert
- technisches Grundverständnis der Funktionskomponenten

Inhalt und Lernziele

Kennenlernen der Themen

- Prinzip des von-Neumann-Rechners
- Zahldarstellung, Rechnerarithmetik, Codierung
- Abstraktionsebenen, Hardware/Software-Schnittstelle
- Befehlssätze und Maschinenprogrammierung (Assembler)
- Befehlsabarbeitung in Prozessoren, Pipelining
- Adressierungsarten, Speicherhierarchie und -verwaltung

$\Rightarrow$ Informatik Basiswissen

$\Rightarrow$ Bewertung von Trends und Perspektiven

$\Rightarrow$ Fähigkeit zum Einschätzen zukünftiger Entwicklungen

$\Rightarrow$ Chancen und Grenzen der Miniaturisierung

Motivation

Wie funktioniert ein Digitalrechner?

Warum ist das überhaupt wichtig?

- Informatik ohne Digitalrechner undenkbar
- Grundverständnis der Interaktion von SW und HW
- zum Beispiel für „performante“ Software
- Variantenvielfalt von Mikroprozessorsystemen
  - Supercomputer, Server, Workstations, PCs, ...
  - Medienverarbeitung, Mobile Geräte, ...
  - RFID-Tags, Wegwerfcomputer, ...

Fortschritt

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

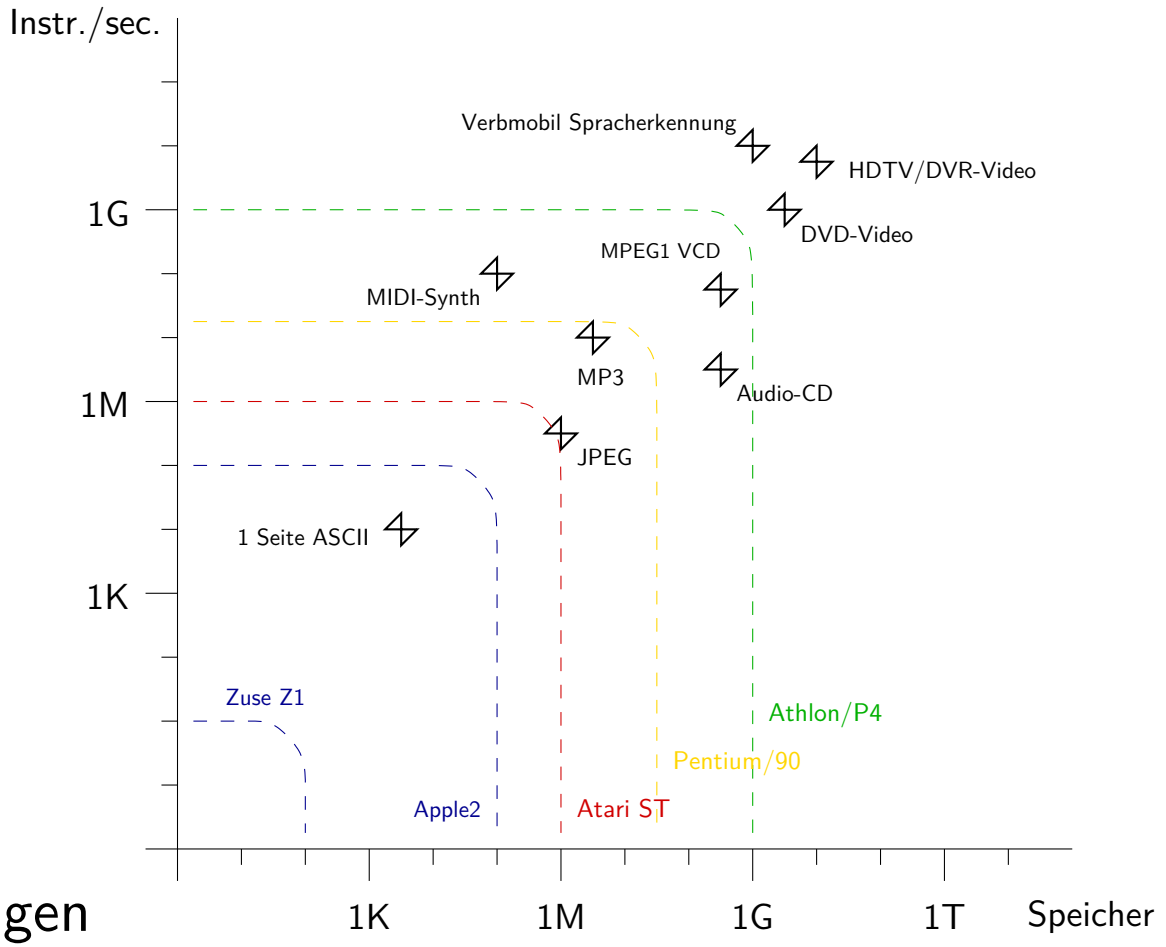
$\Rightarrow$ Möglichkeiten und Anwendungsfelder

$\Rightarrow$ Produkte und Techniken

Fortschritt (cont.)

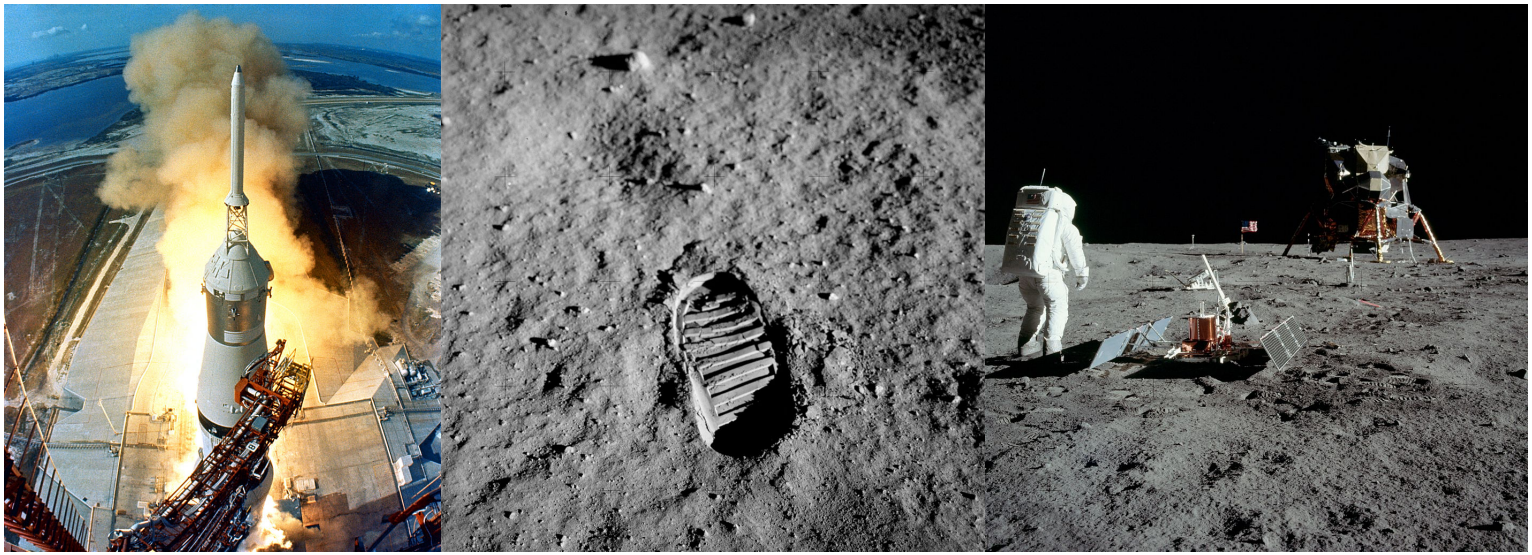
Kriterien / Maßgrößen

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



⇒ jede Rechnergeneration erlaubt neue Anwendungen

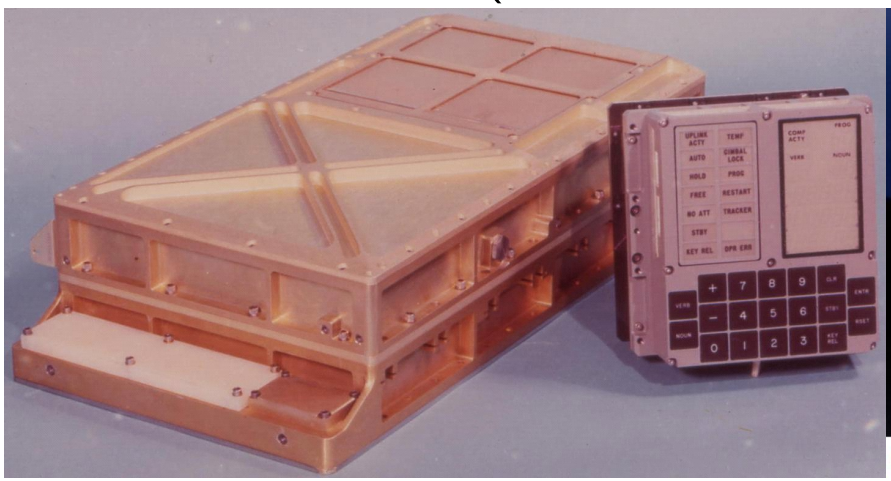
Beispiel: Apollo 11 (1969)



- www.bernd-leitenberger.de/computer-raumfahrt1.shtml
- www.hq.nasa.gov/office/pao/History/computers/Compspace.html
- en.wikipedia.org/wiki/Apollo_Guidance_Computer
- en.wikipedia.org/wiki/IBM_System/360

Beispiel: Apollo 11 (1969) (cont.)

1. Bordrechner: AGC (Apollo Guidance Computer)

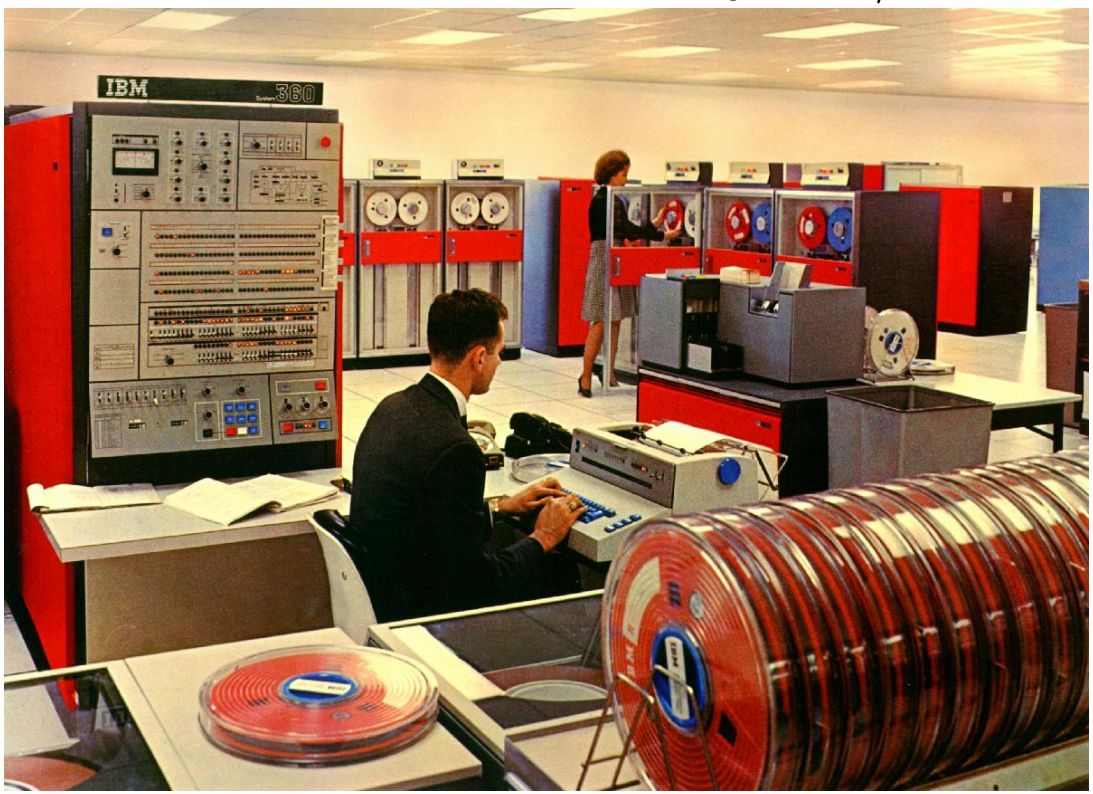


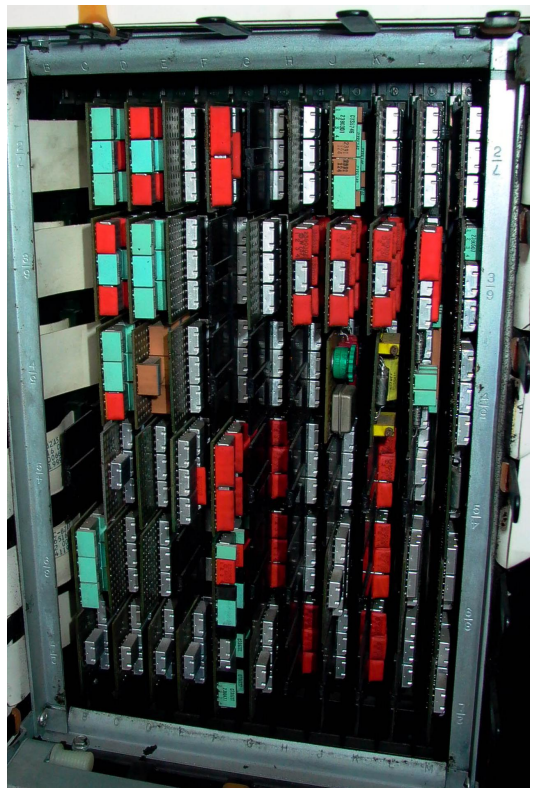
Dryden Flight Research Center EC96-43408-1 Photographed 1996
Apollo computer interface box used in the F-8 Digital
Fly-By-Wire program. NASA photo by Dennis Taylor

- 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: $1{,}024\,\text{MHz}$
- Addition $20\,\mu\text{s}$
- 16-bit Worte, nur Festkomma
- Speicher ROM $36\,\text{KWorte}$ $72\,\text{KByte}$ Zykluszeit $11{,}7\,\text{ms}$ ($85\,\text{Hz}$)
  RAM $2\,\text{KWorte}$ $4\,\text{KByte}$

Beispiel: Apollo 11 (1969) (cont.)

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





Beispiel: Apollo 11 (1969) (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

- Heute. . .

	CPU	Cores	[MIPS]	F_{clk}[GHz]
Smartphone	Exynos 5410	quad	14 600	1,6
Desktop PC	Core i7 4770K	quad	127 273	3,9

Konsequenzen

- wegen technischer Entwicklung: kein „stationärer Zustand“
- Perspektiven/Roadmaps derzeit bis über 2025 hinaus. . .

- Details zu Rechnerorganisation veralten schnell
  aber die Konzepte bleiben gültig (!)

- Schwerpunkt der Vorlesung auf dem „Warum“
  Ziel: ein Gefühl für Größenordnungen entwickeln

- Software entwickelt sich teilweise viel langsamer:
  LISP seit 1958, Prolog 1972, Smalltalk/OO 1972, usw.

Kapitel 1

Definition: Digitalrechner

Tanenbaum, Austin: *Rechnerarchitektur* [TA14]

Ein Computer oder Digitalrechner ist eine Maschine, die Probleme für den Menschen lösen kann, indem sie die ihr gegebenen Befehle ausführt. Eine Befehlssequenz, die beschreibt, wie eine bestimmte Aufgabe auszuführen ist, nennt man ***Programm****.*
Die elektronischen Schaltungen eines Computers verstehen eine begrenzte Menge einfacher Befehle, in die alle Programme konvertiert werden müssen, bevor sie sich ausführen lassen. . . .

- Probleme lösen: durch Abarbeiten einfacher **Befehle**
- Abfolge solcher Befehle ist ein **Programm**
- Maschine versteht nur ihre eigene **Maschinensprache**

Befehlssatz und Semantic Gap

. . . verstehen eine begrenzte Menge einfacher Befehle . . .

Typische Beispiele für solche Befehle:

- addiere die zwei Zahlen in Register R1 und R2
- überprüfe, ob das Resultat Null ist
- kopiere ein Datenwort von Adresse 13 ins Register R4

$\Rightarrow$ extrem niedriges Abstraktionsniveau

- natürliche Sprache immer mit Kontextwissen
  Beispiel: „vereinbaren Sie einen Termin mit dem Steuerberater“

- **Semantic gap**:
  Diskrepanz zu einfachen elementaren Anweisungen
- Vermittlung zwischen Mensch und Computer erfordert zusätzliche Abstraktionsebenen und Software

Rechnerarchitektur bzw. -organisation

- Definition solcher Abstraktionsebenen bzw. Schichten
- mit möglichst einfachen und sauberen Schnittstellen
- jede Ebene definiert eine neue (mächtigere) **Sprache**

- diverse Optimierungs-Kriterien/Möglichkeiten:
  - Performance, Hardwarekosten, Softwarekosten, ...
  - Wartungsfreundlichkeit, Stromverbrauch, ...

Achtung / Vorsicht:

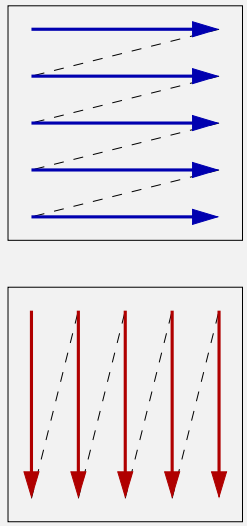
- Gesamtverständnis erfordert Kenntnisse auf allen Ebenen
- häufig Rückwirkung von unteren auf obere Ebenen

Rückwirkung von unteren Ebenen: Arithmetik

```
public class Overflow {
  ...
  public static void main( String[] args ) {
    printInt( 0 );                      //  0
    printInt( 1 );                      //  1
    printInt( -1 );                     // -1
    printInt( 2+(3*4) );                // 14
    printInt( 100*200*300 );            // 6000000
    printInt( 100*200*300*400 );        // -1894967296          (!)
    printDouble( 1.0 );                 // 1.0
    printDouble( 0.3 );                 // 0.3
    printDouble( 0.1 + 0.1 + 0.1 );     // 0.30000000000000004  (!)
    printDouble( (0.3) - (0.1+0.1+0.1) ); // -5.5E-17           (!)
  }
}
```

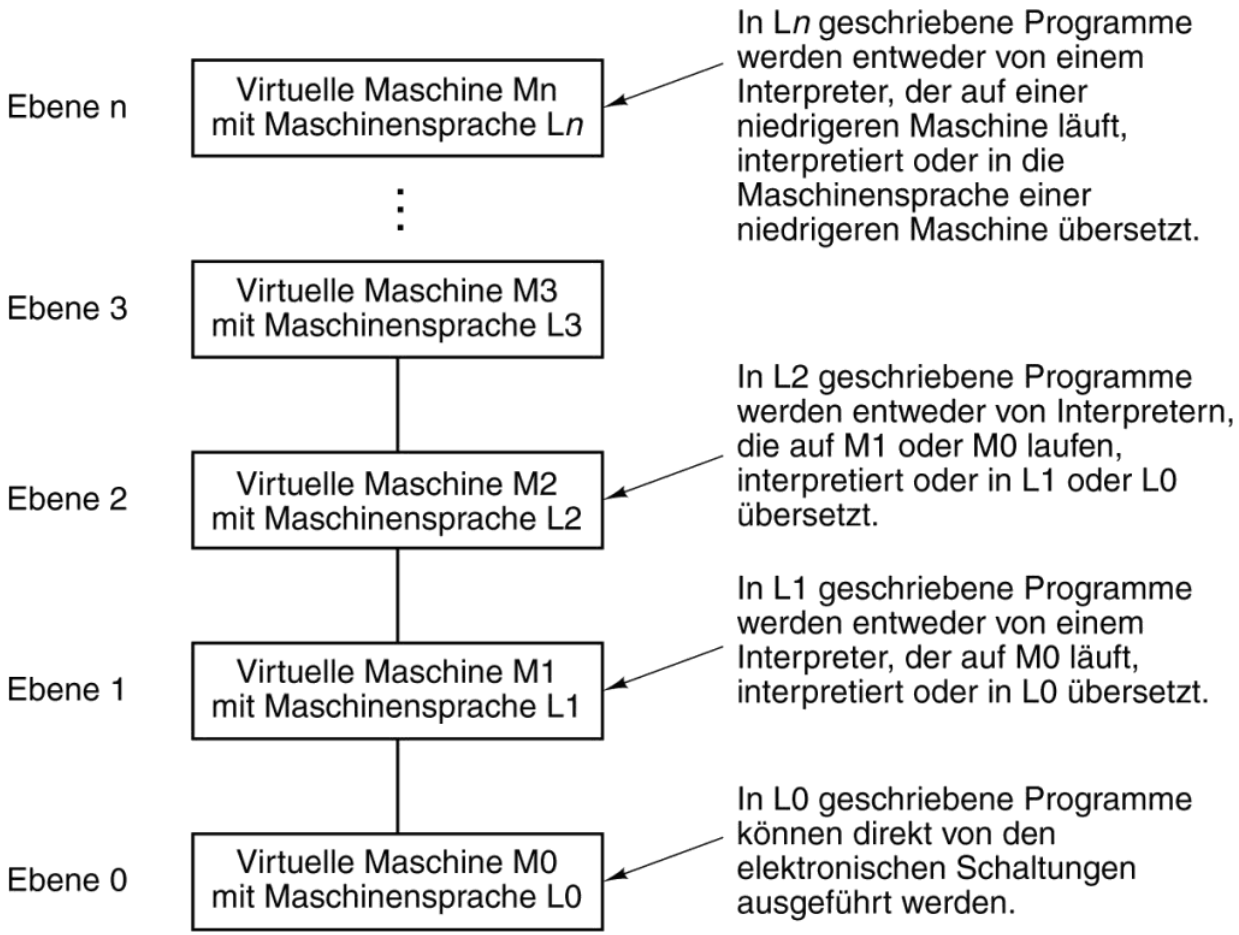
Rückwirkung von unteren Ebenen: Performance

```
public static double sumRowCol( double[][] matrix ) {
  int rows = matrix.length;
  int cols = matrix[0].length;
  double sum = 0.0;
  for( int r = 0; r < rows; r++ ) {
    for( int c = 0; c < cols; c++ ) {
      sum += matrix[r][c];
    }
  }
  return sum;
}
```



Matrix creation (5000×5000)	2105 msec.	
Matrix row-col summation	75 msec.	
Matrix col-row summation	383 msec.	⇒ 5x langsamer
Sum = 600,8473695346258 / 600,8473695342268		⇒ andere Werte

Maschine mit mehreren Ebenen



Tanenbaum: *Structured Computer Organization* [TA14]

Abstraktionsebenen und Sprachen

- jede Ebene definiert eine neue (mächtigere) Sprache
- Abstraktionsebene $\Longleftrightarrow$ Sprache
- L0 $<$ L1 $<$ L2 $<$ L3 $<$ ...

Software zur Übersetzung zwischen den Ebenen

- **Compiler**:
  Erzeugen eines neuen Programms, in dem jeder L1 Befehl durch eine zugehörige Folge von L0 Befehlen ersetzt wird
- **Interpreter**:
  direkte Ausführung der L0 Befehlsfolgen zu jedem L1 Befehl

Virtuelle Maschine

- für einen Interpreter sind L1 Befehle einfach nur Daten
- die dann in die zugehörigen L0 Befehle umgesetzt werden

$\Rightarrow$ dies ist gleichwertig mit einer:

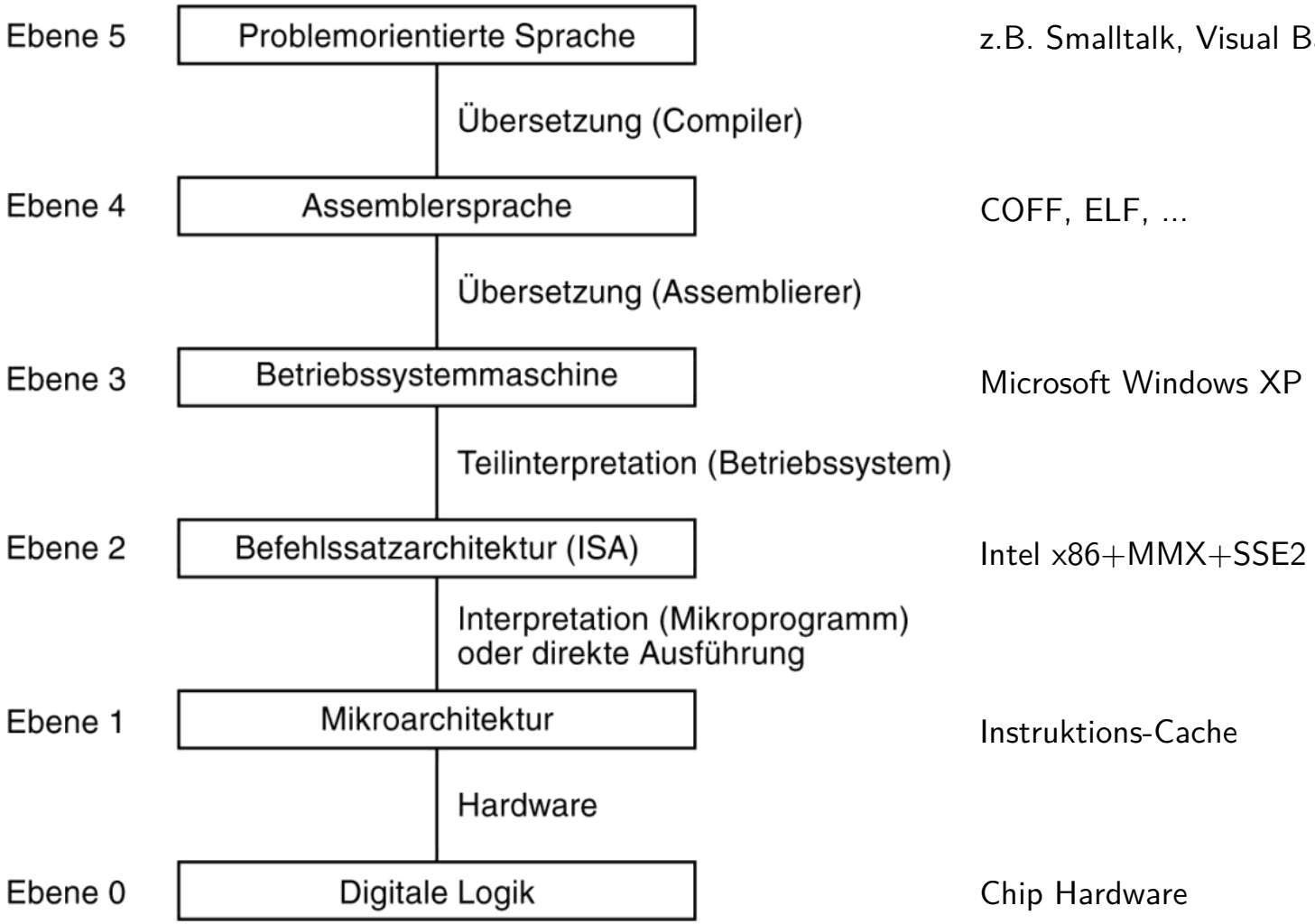
Virtuellen Maschine M1 für die Sprache L1

- ein Interpreter erlaubt es, jede beliebige Maschine zu simulieren
- und zwar auf jeder beliebigen (einfacheren) Maschine M0
- Programmierer muss sich nicht um untere Schichten kümmern
- Nachteil: die virtuelle Maschine ist meistens langsamer als die echte Maschine M1
- Maschine M0 kann wiederum eine virtuelle Maschine sein (!)
- unterste Schicht ist jeweils die Hardware

Übliche Einteilung der Ebenen

Anwendungsebene	Hochsprachen (Java, Smalltalk, ...)
Assemblerebene	low-level Anwendungsprogrammierung
Betriebssystemebene	Betriebssystem, Systemprogrammierung
Rechnerarchitektur	Schnittstelle zwischen SW und HW, Befehlssatz, Datentypen
Mikroarchitektur	Steuerwerk und Operationswerk: Register, ALU, Speicher, ...
Logikebene	Grundschaltungen: Gatter, Flipflops, ...
Transistorebene	Transistoren, Chip-Layout
Physikalische Ebene	Elektrotechnik, Geometrien

Beispiel: Sechs Ebenen



Hinweis: Ebenen vs. Vorlesungen im BSc-Studiengang

Anwendungsebene:	SE1+SE2, AD, ...
Assemblerebene:	RS
Betriebssystemebene:	GSS
Rechnerarchitektur:	RS
Mikroarchitektur:	RS
Logikebene:	RS
Device-Level:	–

HelloWorld: Anwendungsebene Quellcode

```
/* HelloWorld.c - print a welcome message */

#include <stdio.h>

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

Übersetzung

```
gcc -S HelloWorld.c
gcc -c HelloWorld.c
gcc -o HelloWorld.exe HelloWorld.c
```

HelloWorld: Assemblerebene `cat HelloWorld.s`

```
main:
  leal   4(%esp), %ecx
  andl   $-16, %esp
  pushl  -4(%ecx)
  pushl  %ebp
  movl   %esp, %ebp
  pushl  %ecx
  subl   $4, %esp
  movl   $.LC0, (%esp)
  call   puts
  movl   $0, %eax
  addl   $4, %esp
  popl   %ecx
  popl   %ebp
  leal   -4(%ecx), %esp
  ret
```

HelloWorld: Objectcode `od -x HelloWorld.o`

```
0000000 457f 464c 0101 0001 0000 0000 0000 0000
0000020 0001 0003 0001 0000 0000 0000 0000 0000
0000040 00f4 0000 0000 0000 0034 0000 0000 0028
0000060 000b 0008 4c8d 0424 e483 fff0 fc71 8955
0000100 51e5 ec83 c704 2404 0000 0000 fce8 ffff
0000120 b8ff 0000 0000 c483 5904 8d5d fc61 00c3
0000140 6548 6c6c 2c6f 7720 726f 646c 0021 4700
0000160 4343 203a 4728 554e 2029 2e34 2e31 2032
0000200 3032 3630 3131 3531 2820 7270 7265 6c65
0000220 6165 6573 2029 5328 5355 2045 694c 756e
0000240 2978 0000 732e 6d79 6174 0062 732e 7274
0000260 6174 0062 732e 7368 7274 6174 0062 722e
0000300 6c65 742e 7865 0074 642e 7461 0061 622e
0000320 7373 2e00 6f72 6164 6174 2e00 6f63 6d6d
0000340 6e65 0074 6e2e 746f 2e65 4e47 2d55 7473
...
```

HelloWorld: Disassemblieren `objdump -d HelloWorld.o`

```
HelloWorld.o:     file format elf32-i386
Disassembly of section .text:
00000000 <main>:
   0:   8d 4c 24 04             lea    0x4(%esp),%ecx
   4:   83 e4 f0                and    $0xfffffff0,%esp
   7:   ff 71 fc                pushl  0xfffffffc(%ecx)
   a:   55                      push   %ebp
   b:   89 e5                   mov    %esp,%ebp
   d:   51                      push   %ecx
   e:   83 ec 04                sub    $0x4,%esp
  11:   c7 04 24 00 00 00 00    movl   $0x0,(%esp)
  18:   e8 fc ff ff ff          call   19 <main+0x19>
  1d:   b8 00 00 00 00          mov    $0x0,%eax
  22:   83 c4 04                add    $0x4,%esp
...
```

HelloWorld: Maschinencode `od -x HelloWorld.exe`

```
0000000 457f 464c 0101 0001 0000 0000 0000 0000
0000020 0002 0003 0001 0000 8310 0804 0034 0000
0000040 126c 0000 0000 0000 0034 0020 0009 0028
0000060 001c 001b 0006 0000 0034 0000 8034 0804
0000100 8034 0804 0120 0000 0120 0000 0005 0000
0000120 0004 0000 0003 0000 0154 0000 8154 0804
0000140 8154 0804 0013 0000 0013 0000 0004 0000
0000160 0001 0000 0001 0000 0000 0000 8000 0804
0000200 8000 0804 04c4 0000 04c4 0000 0005 0000
0000220 1000 0000 0001 0000 0f14 0000 9f14 0804
0000240 9f14 0804 0104 0000 0108 0000 0006 0000
0000260 1000 0000 0002 0000 0f28 0000 9f28 0804
...
```

Hardware: „*Versteinerte Software*“

- eine virtuelle Maschine führt L1 Software aus
- und wird mit Software oder Hardware realisiert

$\Rightarrow$ Software und Hardware sind logisch äquivalent
„Hardware is just petrified Software“ K.P. Lentz
– jedenfalls in Bezug auf L1 Programmausführung

Entscheidung für Software- oder Hardwarerealisierung?

- abhängig von vielen Faktoren, u.a.
- Kosten, Performance, Zuverlässigkeit
- Anzahl der (vermuteten) Änderungen und Updates
- Sicherheit gegen (Raub-) Kopieren, . . .

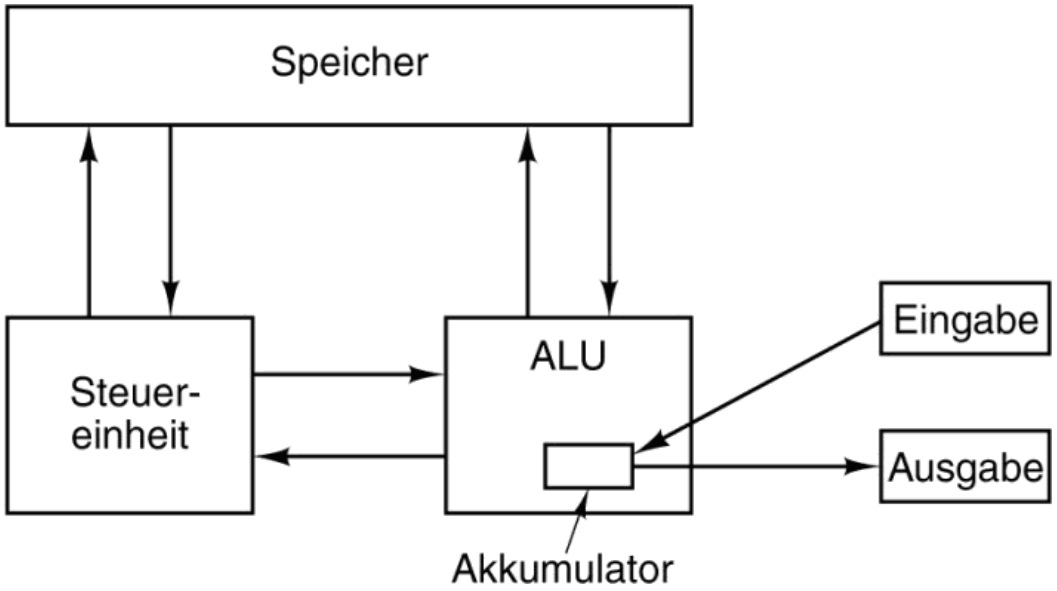
von-Neumann Konzept

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

$\Rightarrow$ enorm flexibel

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

von-Neumann Rechner



[TA14]

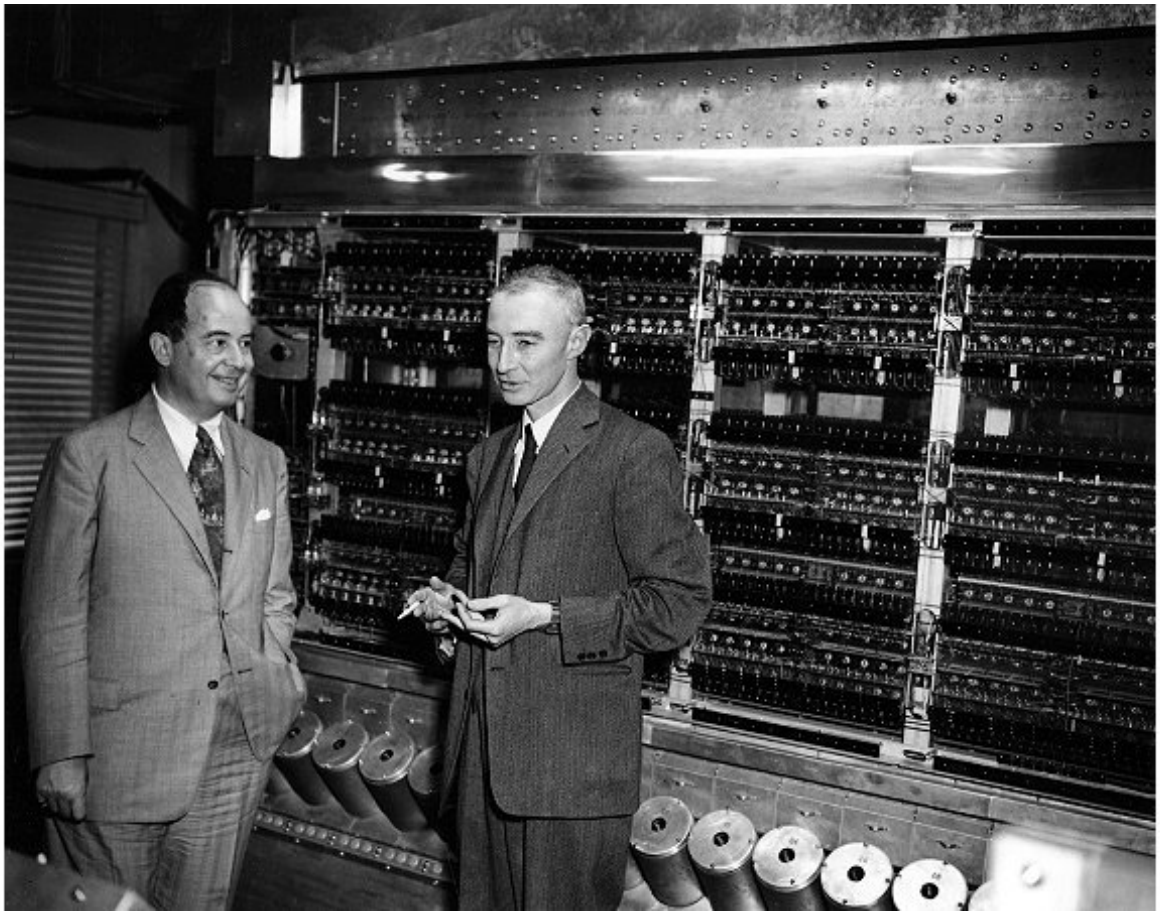
Fünf zentrale Komponenten:

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

von-Neumann Rechner (cont.)

- Steuerwerk: zwei zentrale Register
  - Befehlszähler (*program counter PC*)
  - Befehlsregister (*instruction register IR*)
  - evtl. Adressregister
- Operationswerk (Datenpfad, *data-path*)
  - Rechenwerk (*arithmetic-logic unit ALU*)
  - Universalregister (mind. 1 *Akkumulator*, typisch 8..64 Register)
  - evtl. Register mit Spezialaufgaben
- Speicher (*memory*)
  - Hauptspeicher/RAM: *random-access memory*
  - Hauptspeicher/ROM: *read-only memory* zum Booten
  - Externspeicher: Festplatten, CD/DVD, Magnetbänder
- Peripheriegeräte (Eingabe/Ausgabe, *I/O*)

von-Neumann Rechner: IAS Computer



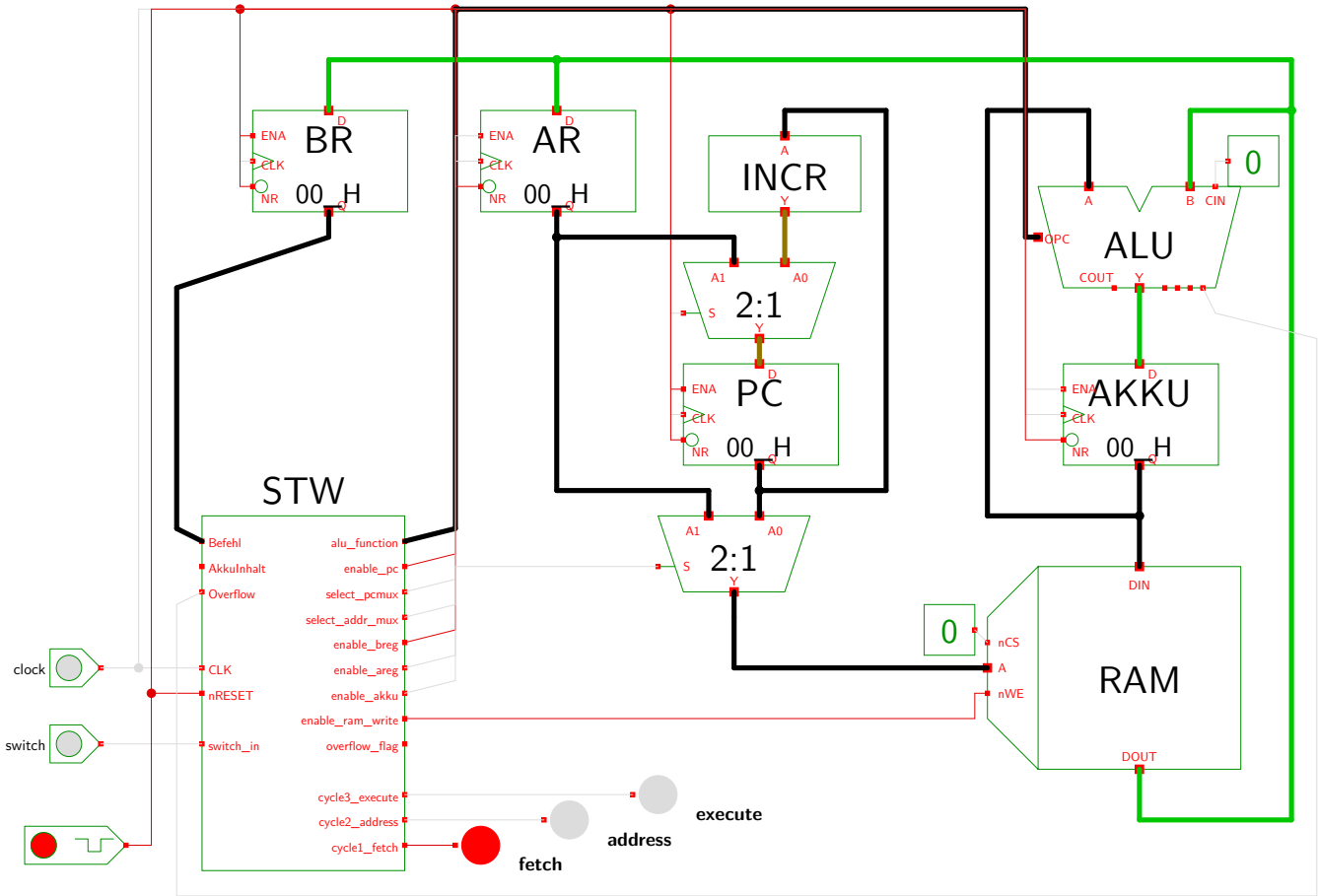
John von Neumann, R. J. Oppenheimer, IAS Computer Princeton `www.computerhistory.org`

PRIMA: die Primitive Maschine

ein (minimaler) 8-bit von-Neumann Rechner

- RAM: Hauptspeicher 256 Worte à 8-bit
- vier 8-bit Register:
  - PC: program-counter
  - BR: instruction register („Befehlsregister“)
  - AR: address register (Speicheradressen und Sprungbefehle)
  - AKKU: accumulator (arithmetische Operationen)
- eine ALU für Addition, Inkrement, Shift-Operationen
- ein Schalter als Eingabegerät
- sehr einfacher Befehlssatz
- Demo: `http://tams.informatik.uni-hamburg.de/applets/hades/webdemos/50-rtlib/90-prima/chapter.html`

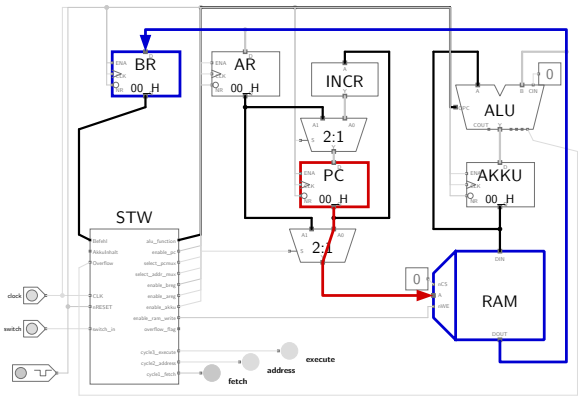
PRIMA: die Primitive Maschine



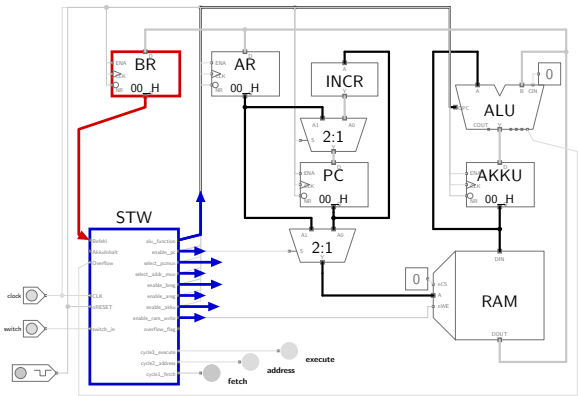
[HenHA] Hades Demo: `50-rtlib/90-prima/prima`

PRIMA: die Zyklen

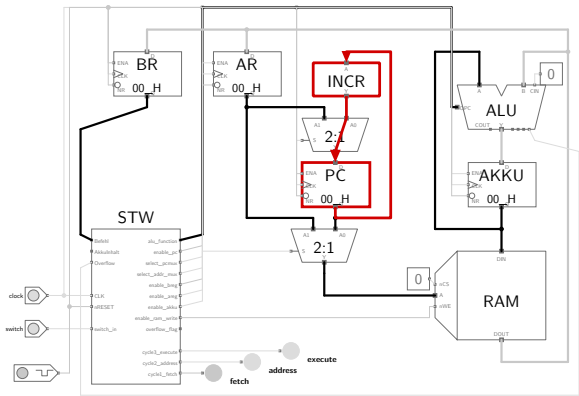
Befehl holen

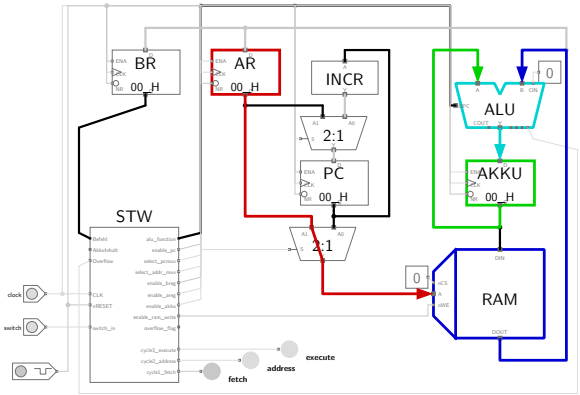


decodieren

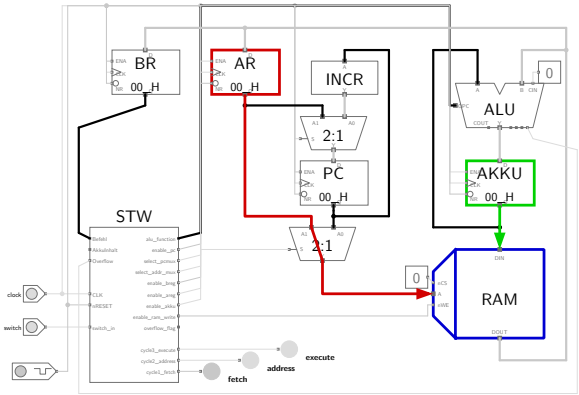


PC inkrementieren

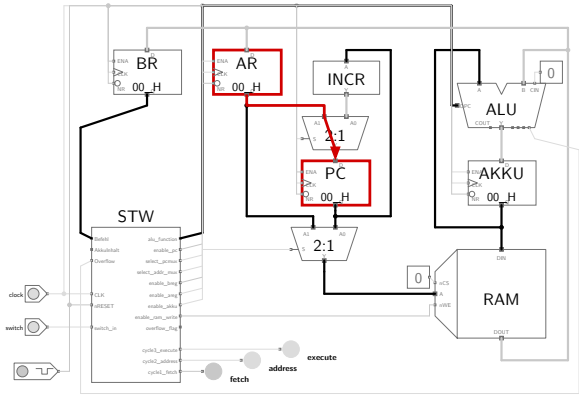




rechnen



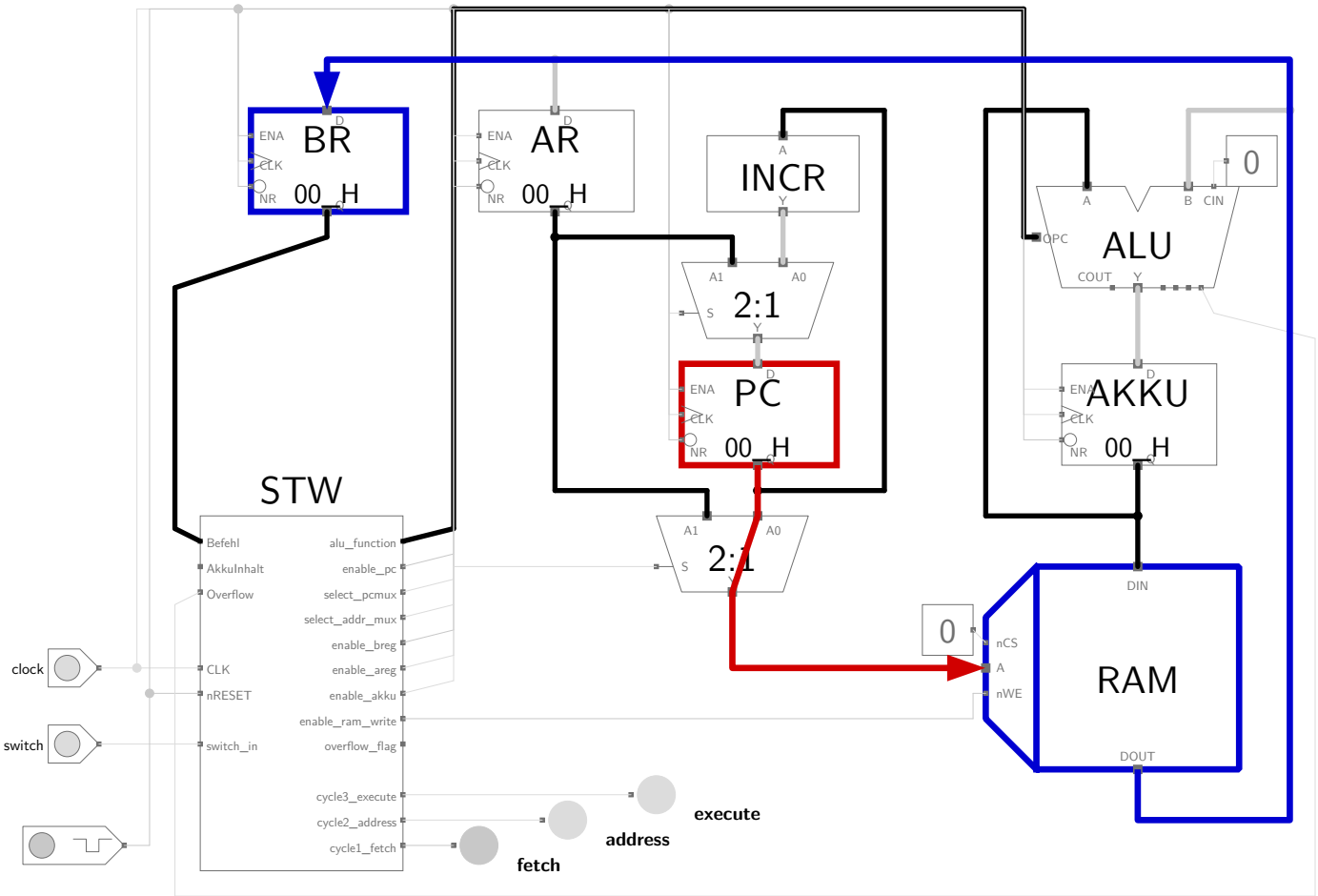
speichern



springen

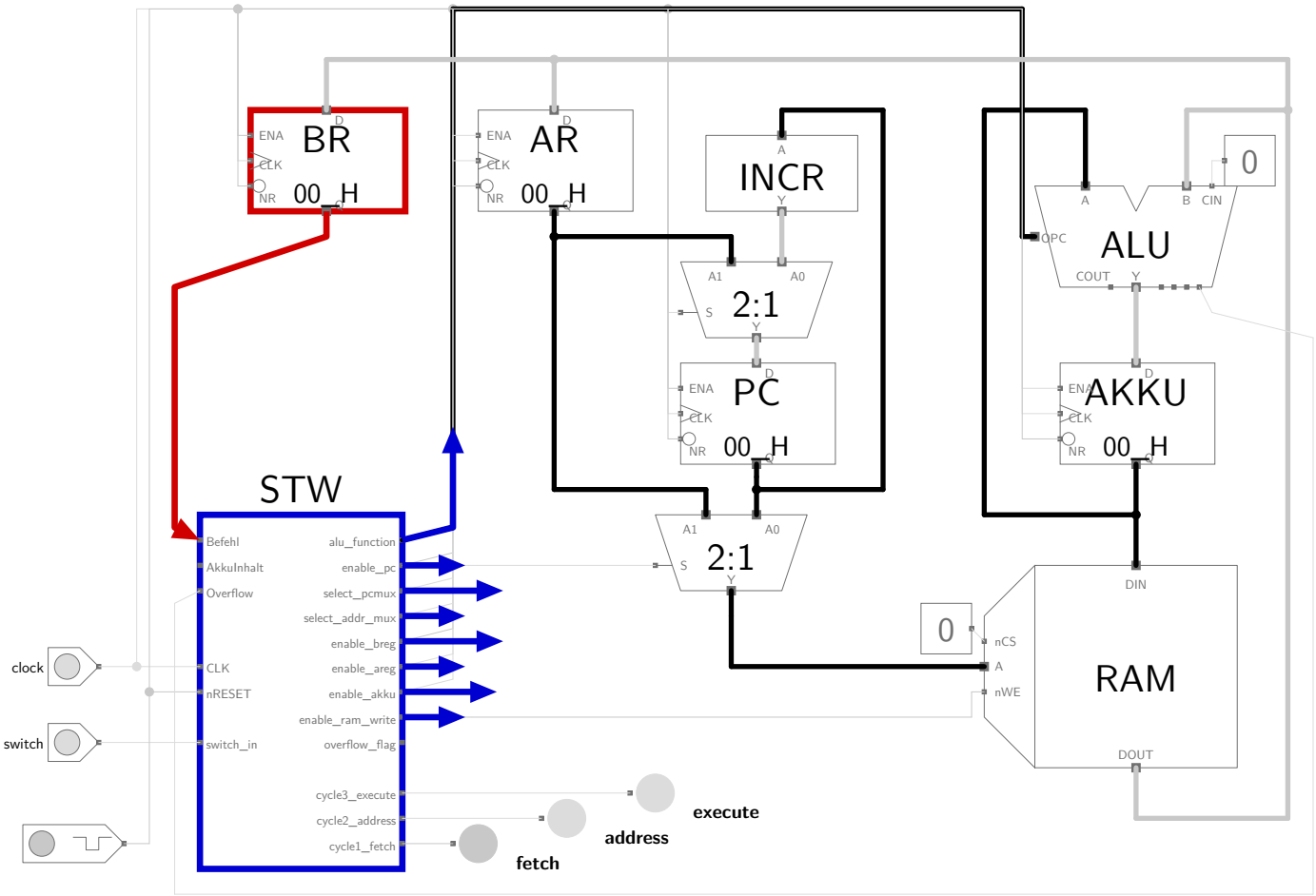
PRIMA: Befehl holen

BR = RAM[PC]



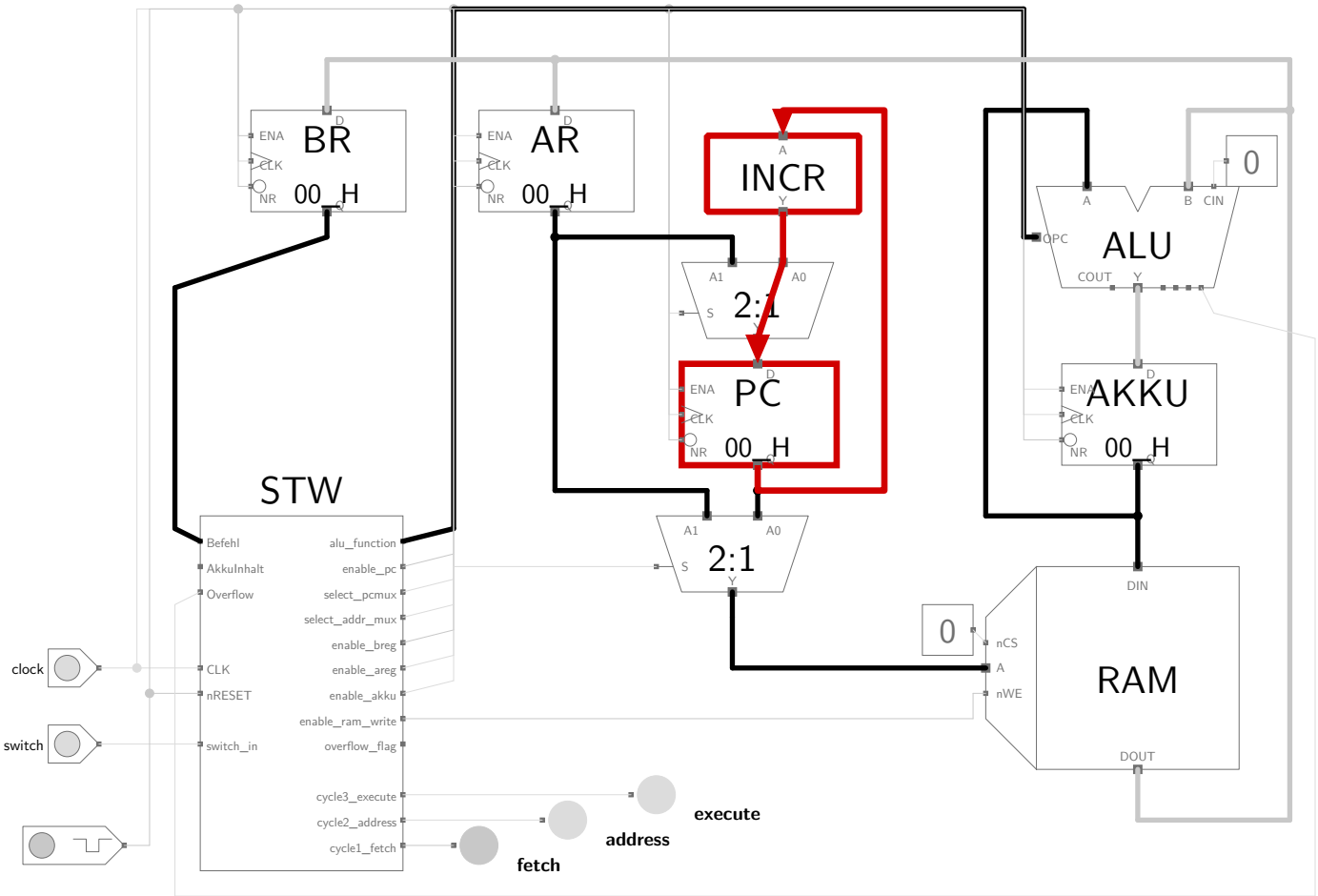
PRIMA: decodieren

Steuersignale = decode(BR)



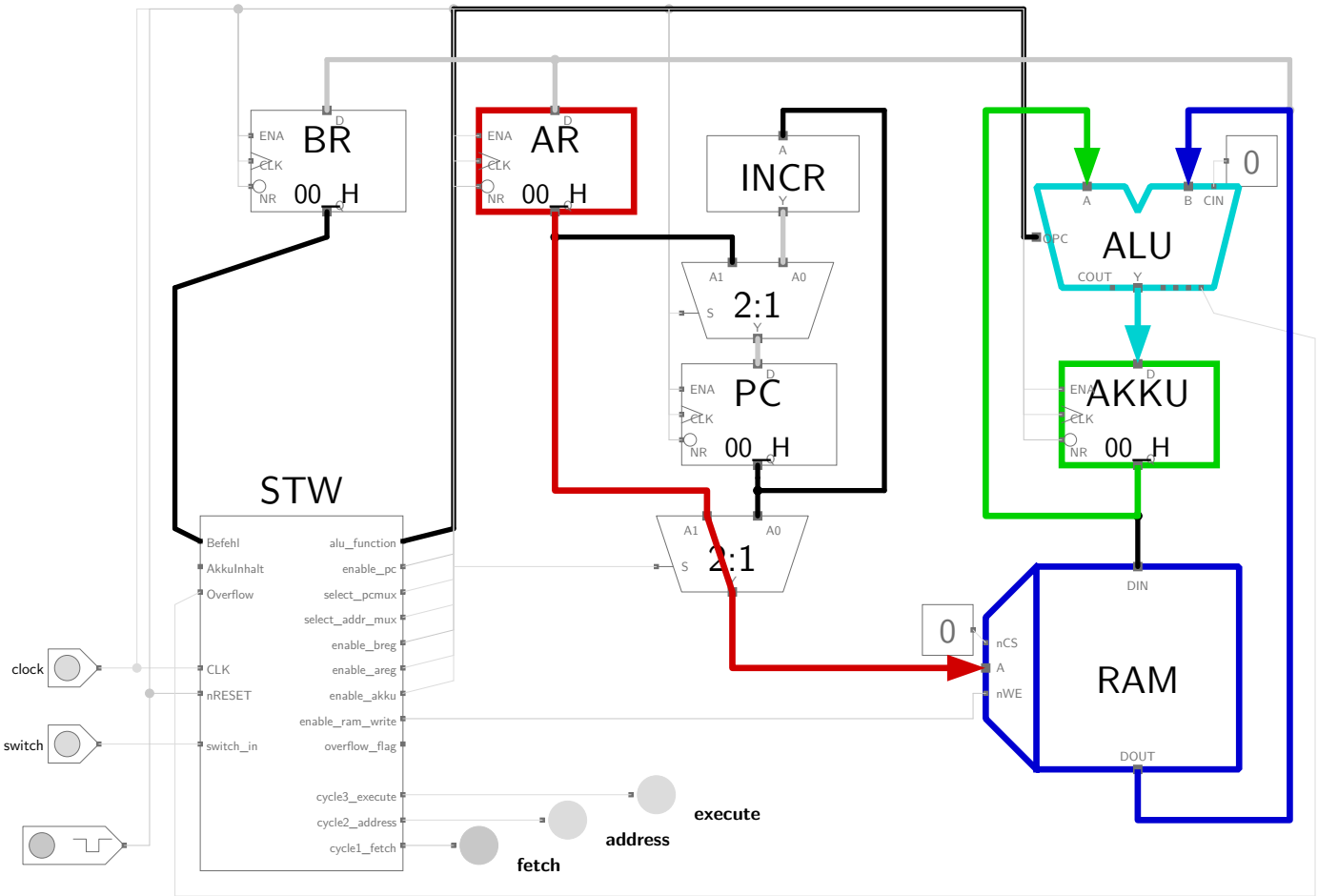
PRIMA: PC inkrementieren

PC = PC+1



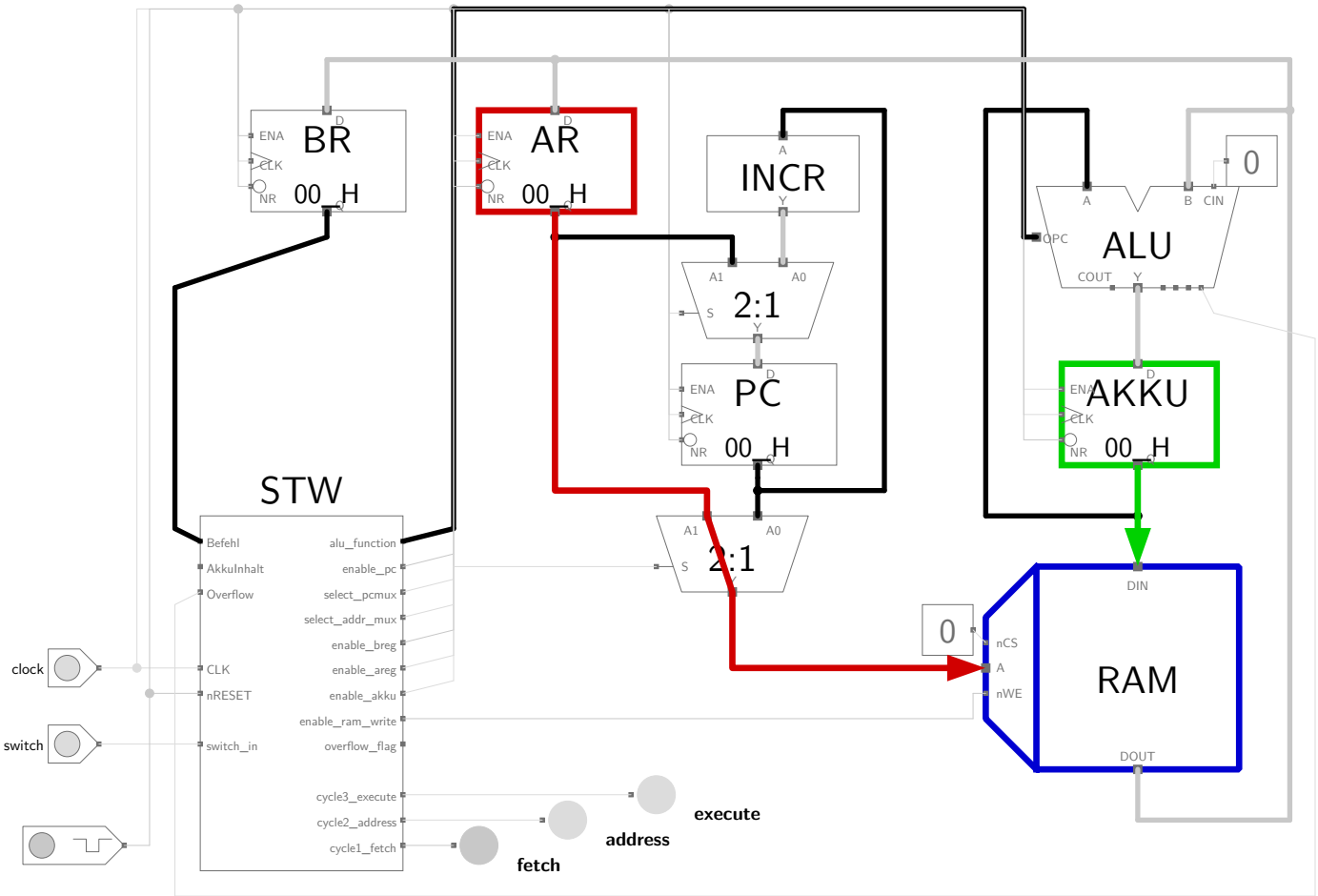
PRIMA: rechnen

Akku = Akku + RAM[AR]



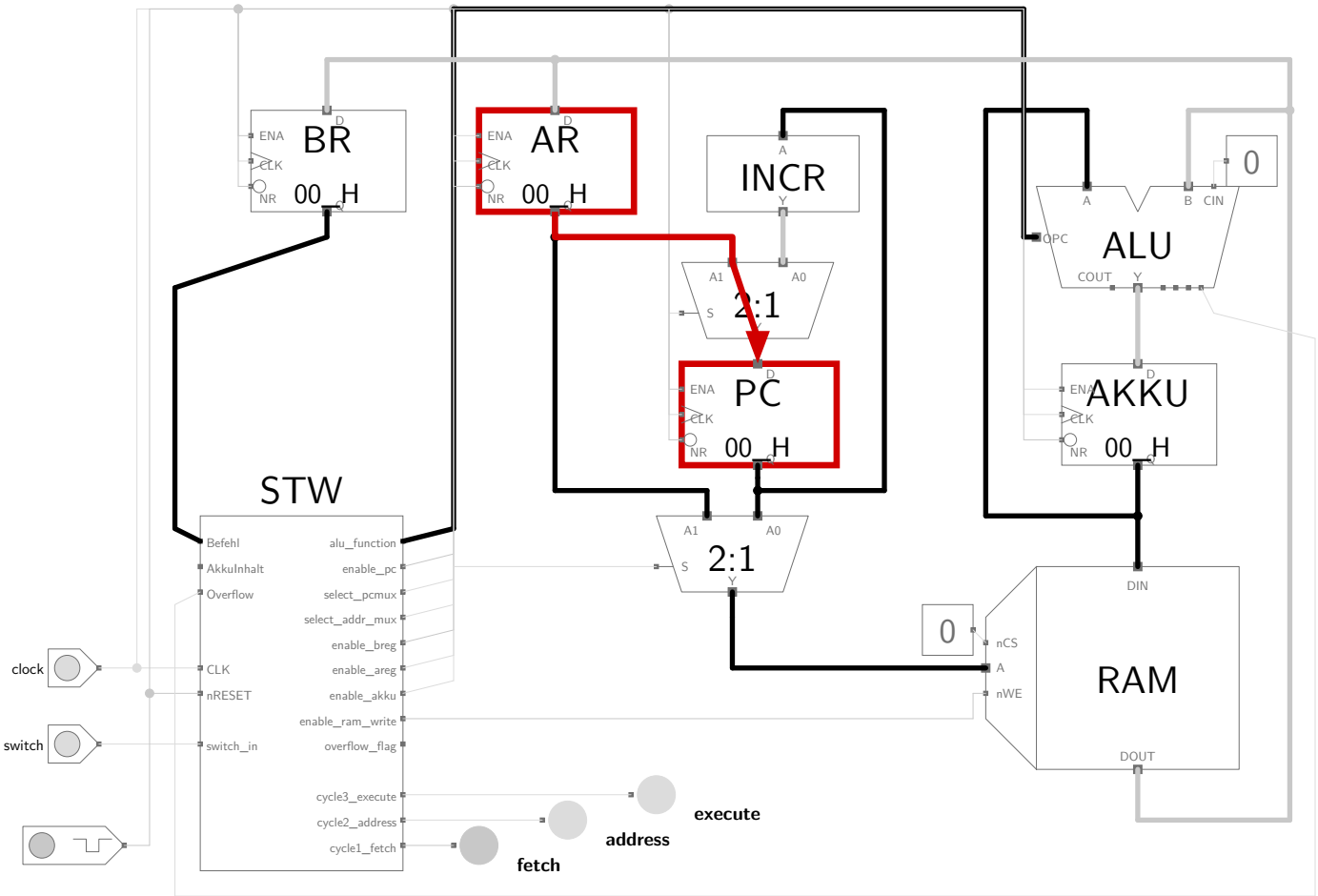
PRIMA: speichern

RAM[AR] = Akku

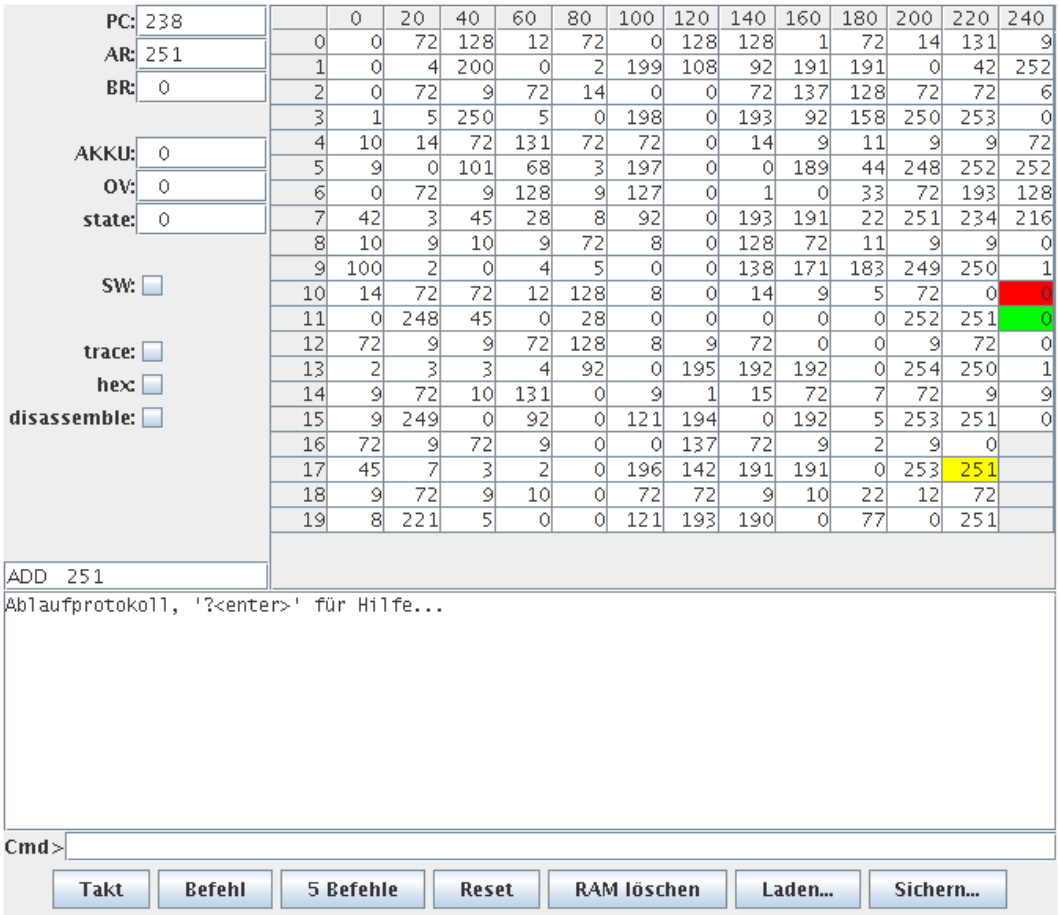


PRIMA: springen

PC = AR



PRIMA: Simulator



	0	20	40	60	80	100	120	140	160	180	200	220	240
0	0	72	128	12	72	0	128	128	1	72	14	131	9
1	0	4	200	0	2	199	108	92	191	191	0	42	252
2	0	72	9	72	14	0	0	72	137	128	72	72	6
3	1	5	250	5	0	198	0	193	92	158	250	253	0
4	10	14	72	131	72	72	0	14	9	11	9	9	72
5	9	0	101	68	3	197	0	0	189	44	248	252	252
6	0	72	9	128	9	127	0	1	0	33	72	193	128
7	42	3	45	28	8	92	0	193	191	22	251	234	216
8	10	9	10	9	72	8	0	128	72	11	9	9	0
9	100	2	0	4	5	0	0	138	171	183	249	250	1
10	14	72	72	12	128	8	0	14	9	5	72	0	0
11	0	248	45	0	28	0	0	0	0	0	252	251	0
12	72	9	9	72	128	8	9	72	0	0	9	72	0
13	2	3	3	4	92	0	195	192	192	0	254	250	1
14	9	72	10	131	0	9	1	15	72	7	72	9	9
15	9	249	0	92	0	121	194	0	192	5	253	251	0
16	72	9	72	9	0	0	137	72	9	2	9	0	
17	45	7	3	2	0	196	142	191	191	0	253	251	
18	9	72	9	10	0	72	72	9	10	22	12	72	
19	8	221	5	0	0	121	193	190	0	77	0	251	

http://tams.informatik.uni-hamburg.de/applets/jython/prima.html

PRIMA: Interpreter: Konzept

```
public class Prima {
  private int   PC, AR, BR;  // program counter, address and instruction register
  private int   AKKU;        // accumulator
  private int   OV, SW;      // overflow flag, input switch
  private int[] RAM;         // 256 bytes main memory
  private int   time, state; // extra variables for simulator state
  ...

  public Prima() {
    time = 0; state = S0;
    PC = 0x00; AR = 0x00; BR = 0x00; AKKU = 0x00; OV = 0; SW = 0;
    RAM = new int[256];
    initializeRAM();
  }

  public void clock() { ... }           // execute one clock cycle
  public void intializeRAM() { ... }    // first time init of RAM
  public void loadRAM( File f ) { ... } // load program/data into simulation
  public void getALU() { ... }          // calculate ALU output
  public void visualize() { ... }       // graphical user interface update
  ...
}
```

PRIMA: Interpreter: Befehlszyklus

```
public void clock()
  if (state == S0) {
     BR = RAM[ PC ];              // load BR register
     PC = (PC + 1) & 0xff;        // increment PC register
  }
  else if (state == S1) {
     AR = RAM[ PC ];              // load AR register
     PC = (PC + 1) & 0xff;        // increment PC register
  }
  else if (state == S2) {         // execute instruction in BR/AR
    if (isBitSet(BR,5)) OV = 0;   // handle *-instructions: reset OV flag

    if (isJump()) {               // true if BR holds a jump opcode
      handleJmpInstruction();     // sets new PC
    }
    else if (isWrite()) {
      RAM[ AR ] = (AKKU & 0xff); // write AKKU content to RAM
    }
    else { // arithmetic op
      AKKU       = getALU();     // function calculates ALU result
      if (!isBitSet(AKKU,8) == getC6()) OV = 1; // update OV flag
      else                              OV = 0;
    }
  } // end if state
} // end clock
```

PRIMA: Multiplikation

- Hardware unterstützt Multiplikation nicht direkt
- Realisierung als Unterprogramm
- fest gewählte Adressen:
  - Codebeginn: 200, Ende: 247
  - Operand 1: 248 (op1)
  - Operand 2: 249 (op2)
  - Resultat: 250 (s)
  - Hilfsvariablen: 251-253 (x, y, h)
  - Konstante '9': 254
  - Rücksprungadresse: 221

PRIMA: Maschinencode für Multiplikation

```
; procedure PRODUKT
;
200  14 LD0          ; AKKU = 0
201   0     dummy
202  72 ST           ; tmp variable s=0
203 250     s
204   9 LD           ; AKKU = op1
205 248     op1
206  72 ST           ; x = op1
207 251     x
208   9 LD           ; AKKU = op2
209 249     op2
210  72 ST           ; y = op2
211 252     y
212   9 LD           ; neun = 9 (number of bits+1)
213 254     9
214  72 ST           ; h = 9
215 253     h
...
```

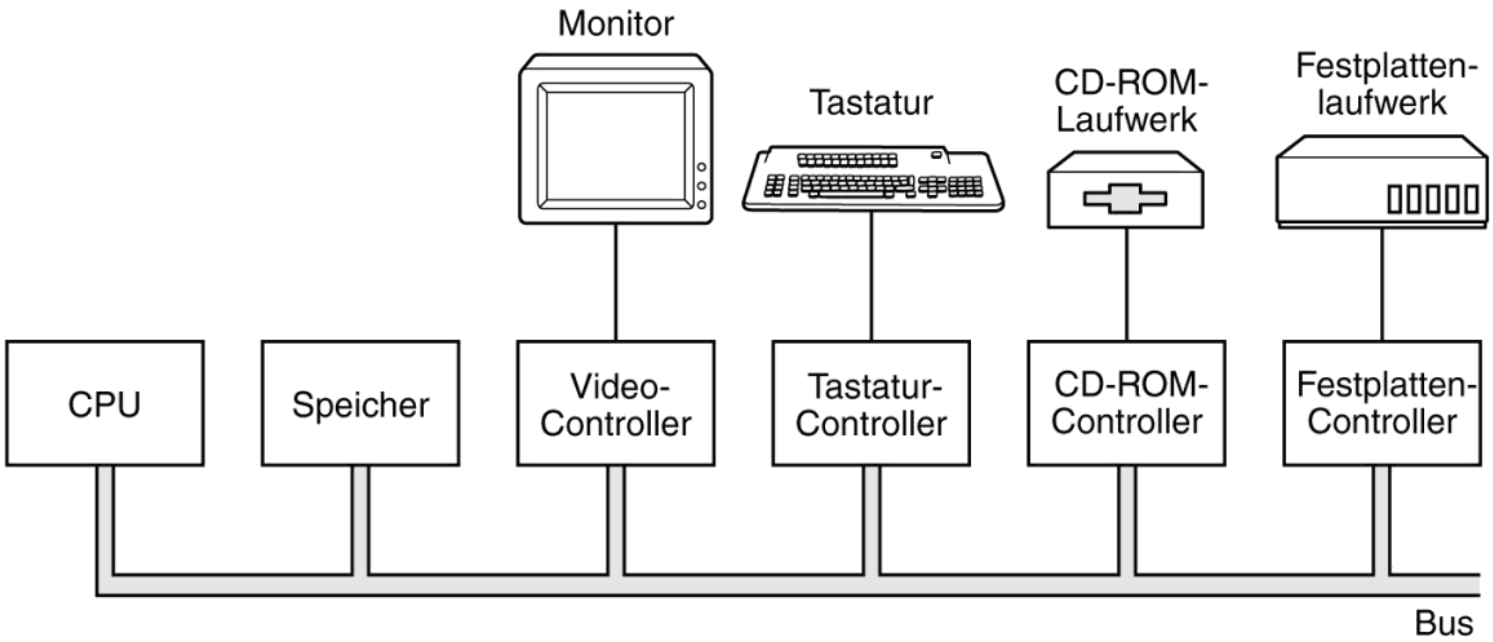
PRIMA: Maschinencode für Multiplikation (cont.)

```
; label 216: check end
216   9              ; AKKU = h
217 253
218  12 SB1          ; AKKU = AKKU - 1
219   0     dummy
220 131 BZ           ; branch-if-zero
221   0     return   ; caller has to set return address here
222  72 ST           ; h = h - 1
223 253     h
224   9 LD           ; AKKU = y
225 252     y
226 193 BEV          ; branch-if-even, if yes goto 234
227 234     label 234
228   9 LD           ; AKKU = s
229 250     s
230   0 ADD          ; AKKU = s + x;
231 251     x
232  72 ST           ; s = s + x
233 250
```

PRIMA: Maschinencode für Multiplikation (cont.)

```
; label 234
234   9 LD          ; AKKU = x
235 251
236   0 ADD         ; AKKU = x + x = 2 * x
237 251     x
238  72 ST          ; x = 2 * x
239 251     x
240   9 LD          ; AKKU = y
241 252     y
242   6 SR          ; shift-right AKKU
243   0     dummy
244  72 ST          ; y = y / 2
245 252
246 128 BU          ; goto 216
247 216    label 216
; data
248  -1             ; op1
249  -1             ; op2
250  -1             ; s, result
251  -1             ; tmp x
252  -1             ; tmp y
253  -1             ; tmp h
254   9             ; constant 9 (8 bits + 1)
```

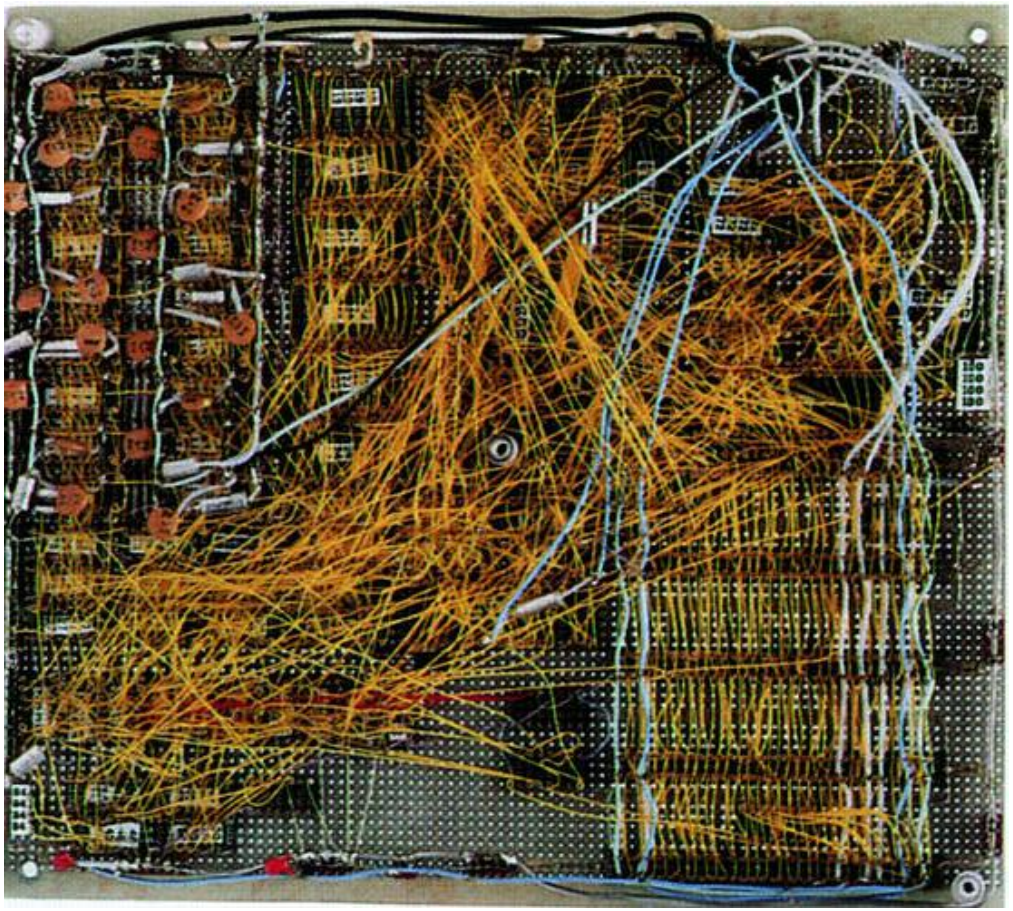
Personal Computer: Aufbau des IBM PC (1981)

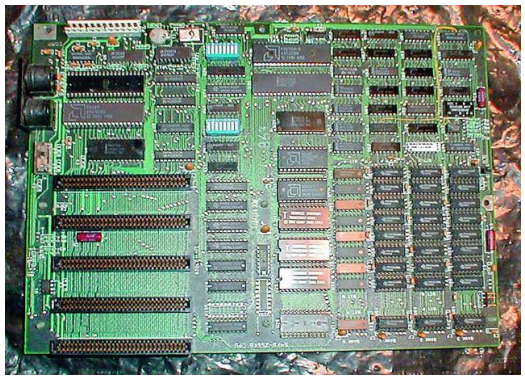


[TA14]

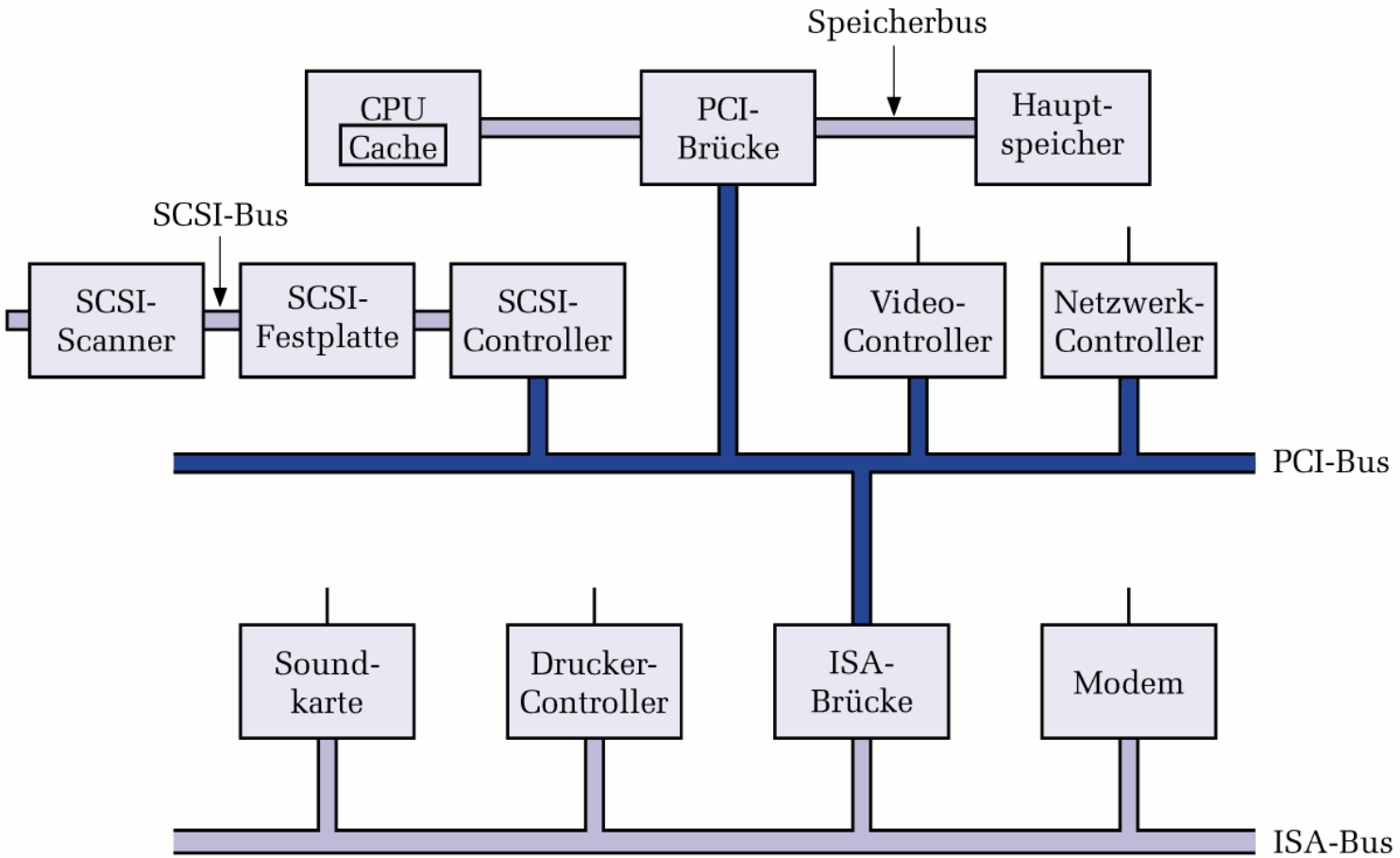
- Intel 8086/8088, 512 KByte RAM, Betriebssystem MS-DOS
- alle Komponenten über den zentralen („ISA“-) Bus verbunden
- Erweiterung über Einsteckkarten

Personal Computer: Prototyp (1981) und Hauptplatine



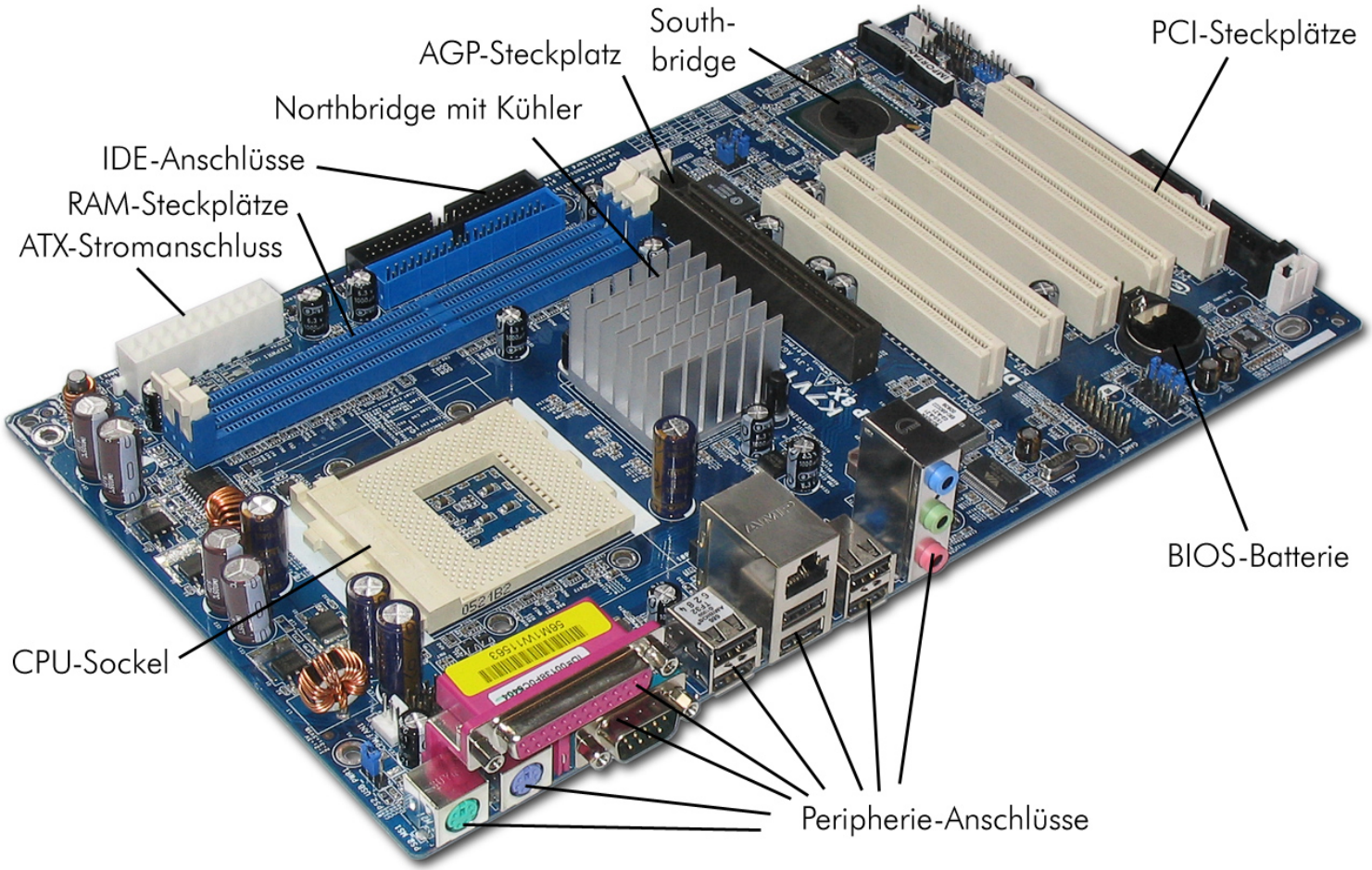


Personal Computer: Aufbau mit PCI-Bus (2000)



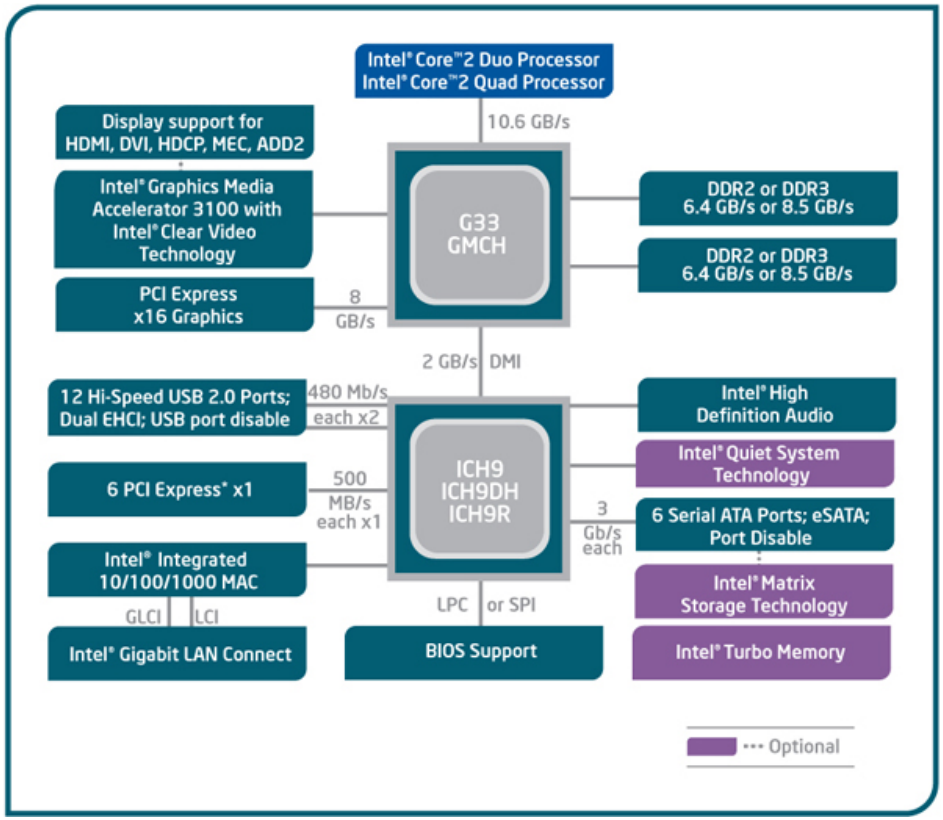
[TA14]

Personal Computer: Hauptplatine (2005)



de.wikibooks.org/wiki/Computerhardware_für_Anfänger

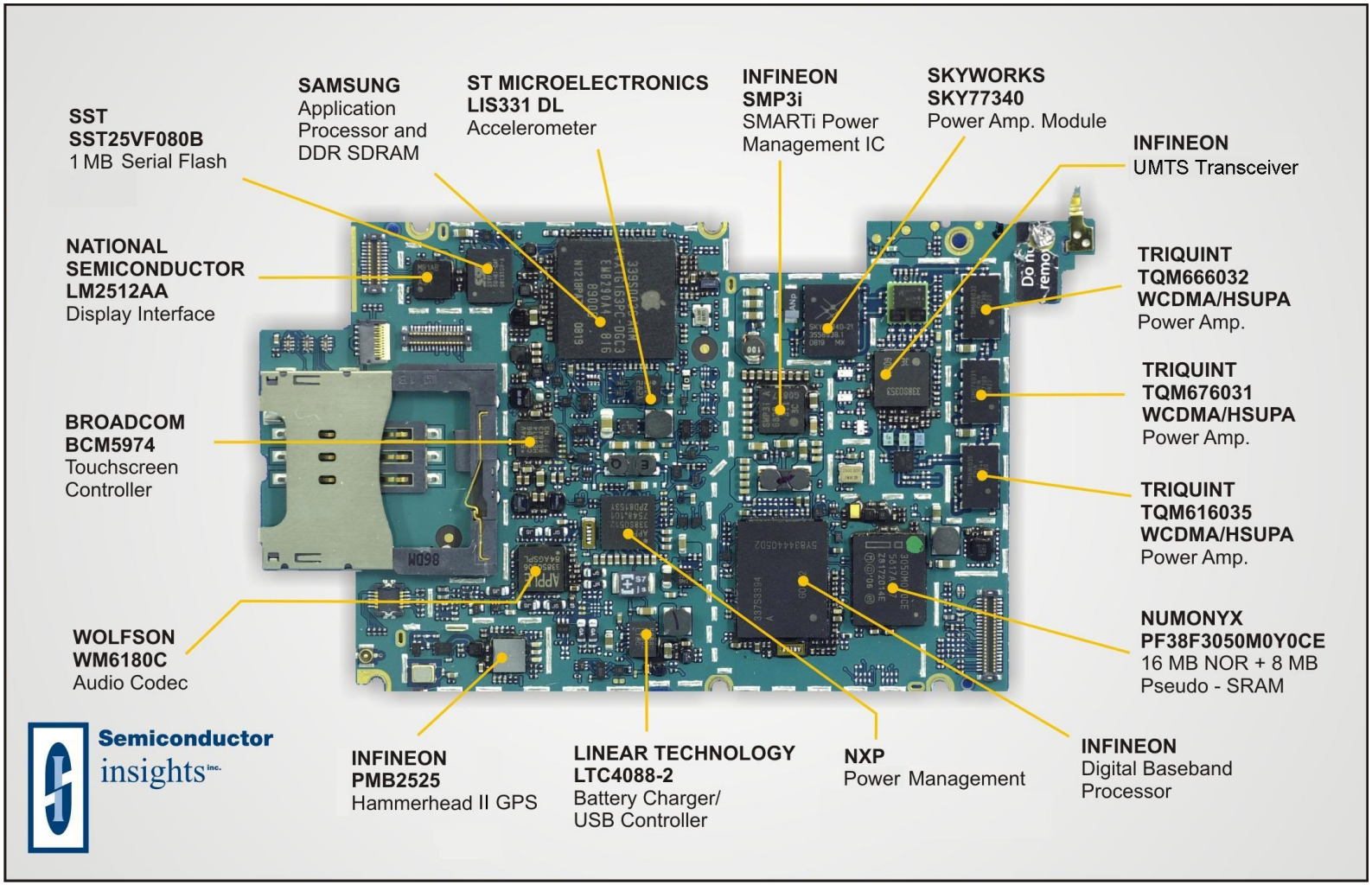
Personal Computer: Aufbau (2010)



Intel `ark.intel.com`

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

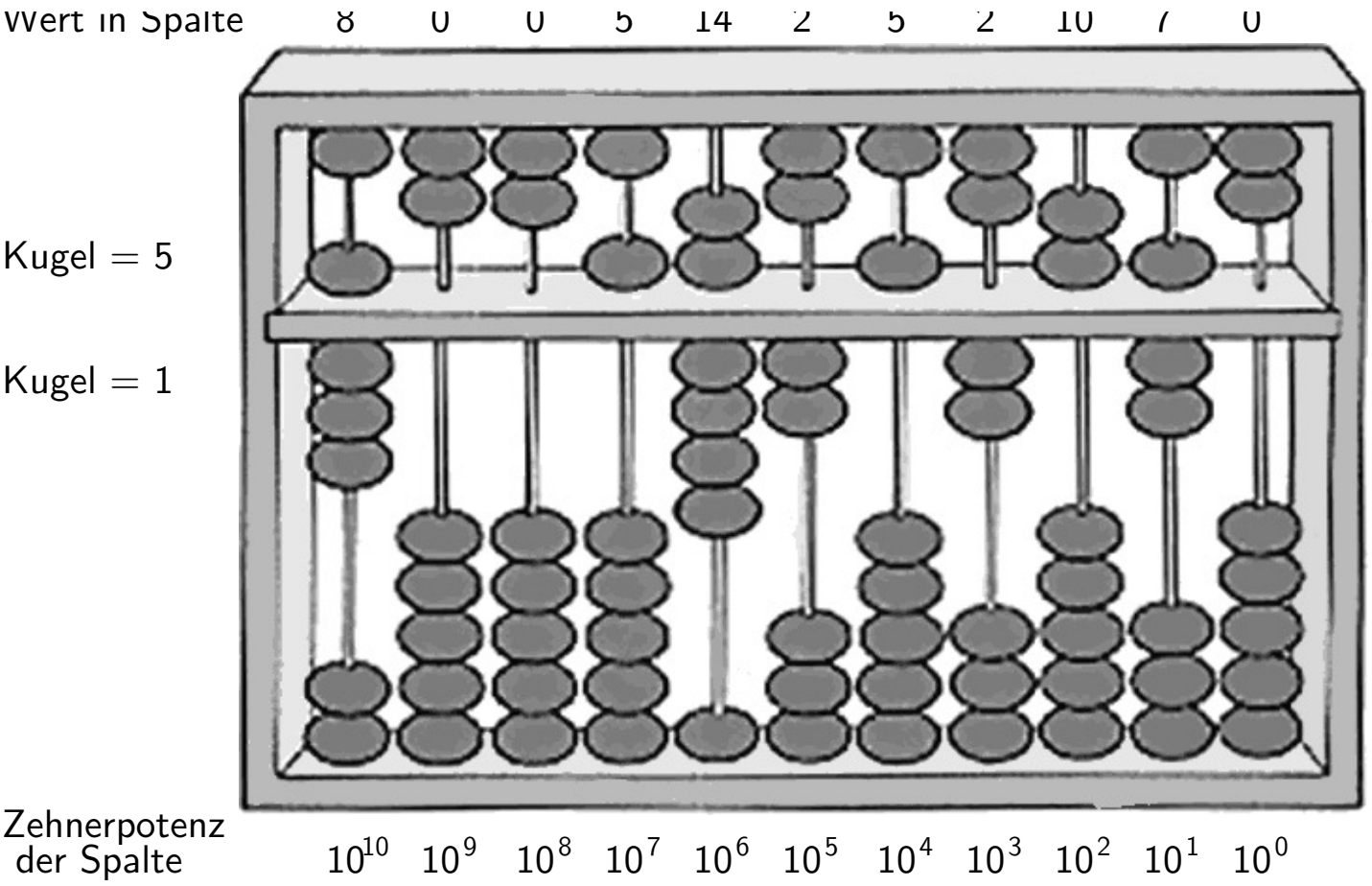
Mobilgeräte: Smartphone (2010)



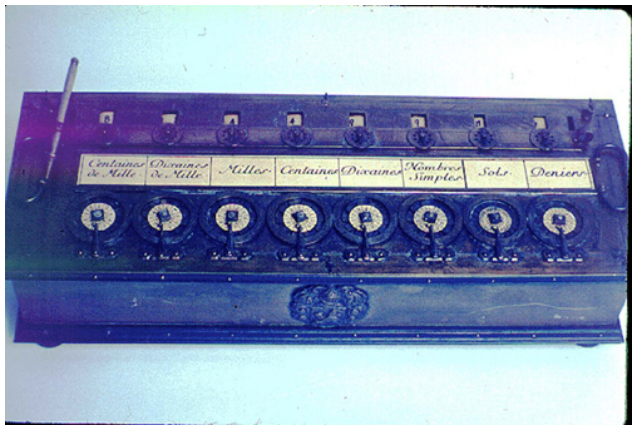
Timeline: Vorgeschichte

????	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
Kugel = 5
Kugel = 1
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}

Mechanische Rechenmaschinen



1623 Schickard: Sprossenrad, Addierer/Subtrahierer

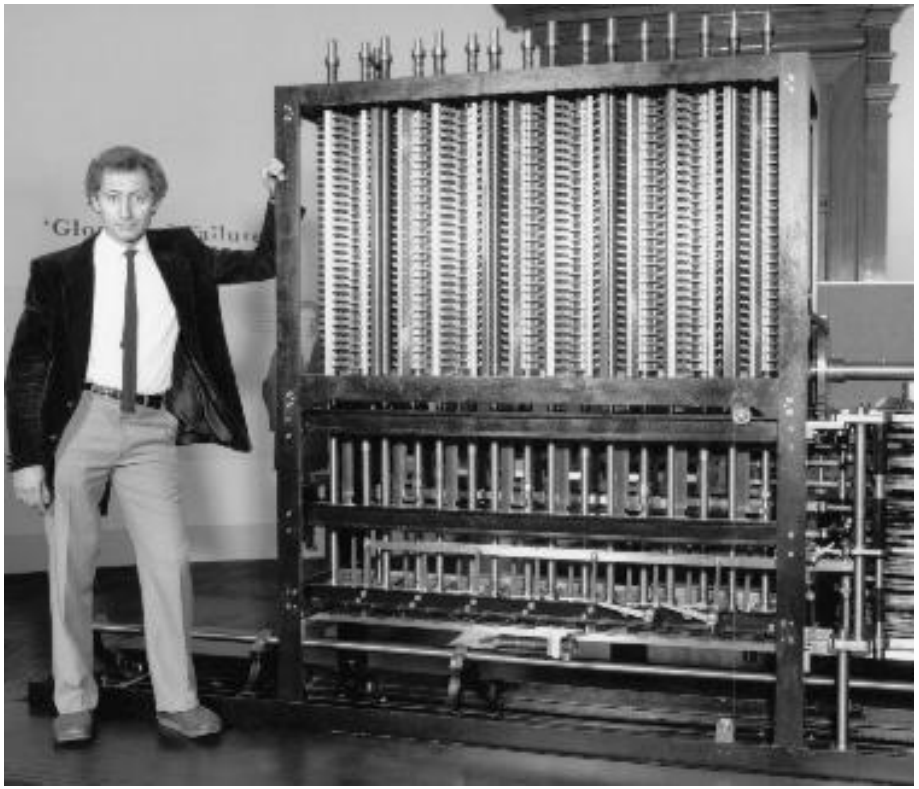
1642 Pascal: „Pascalene“

1673 Leibniz: Staffelwalze, Multiplikation/Division

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

Difference Engine

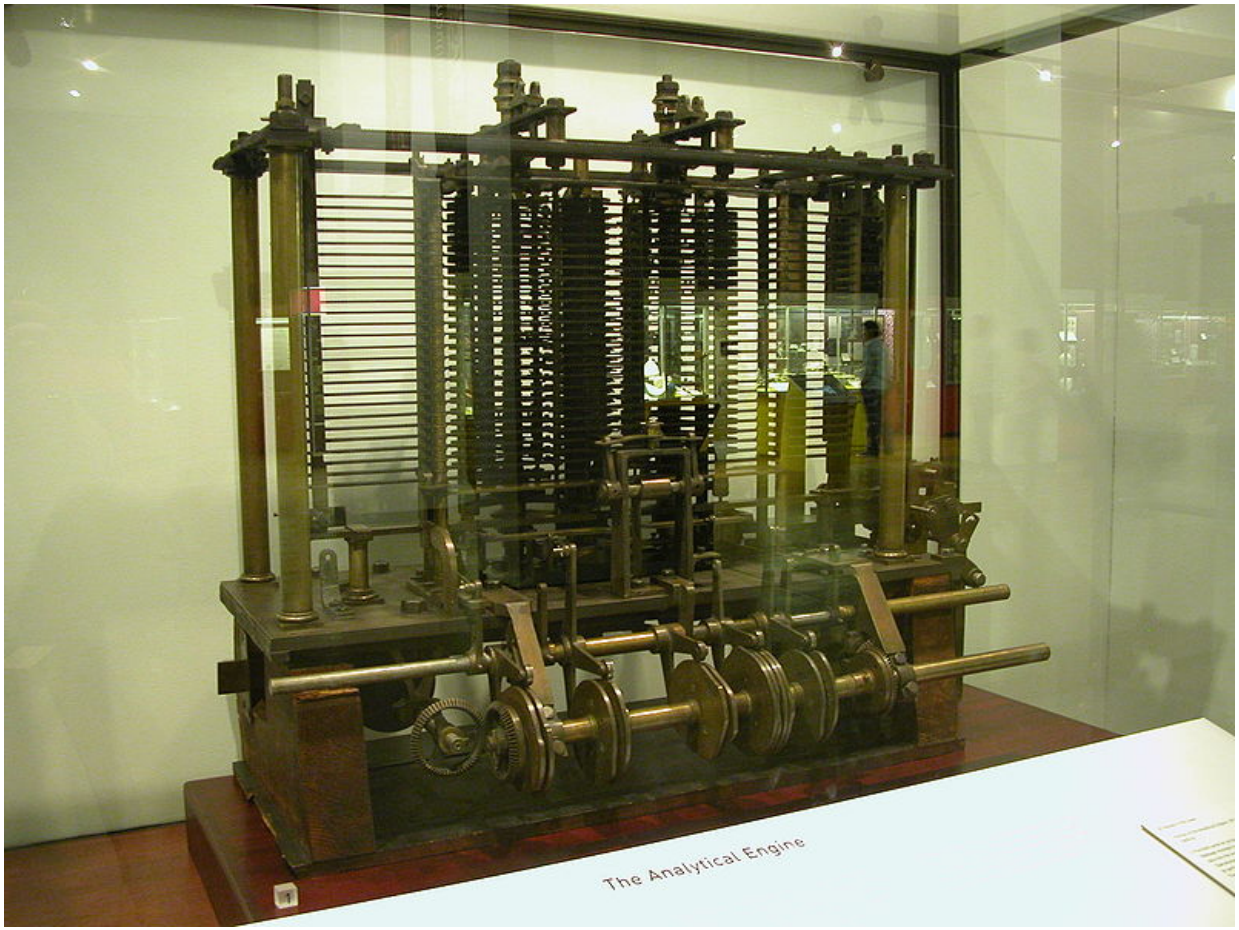
Charles Babbage 1822: Berechnung nautischer Tabellen



Original von 1832 und Nachbau von 1989, London Science Museum

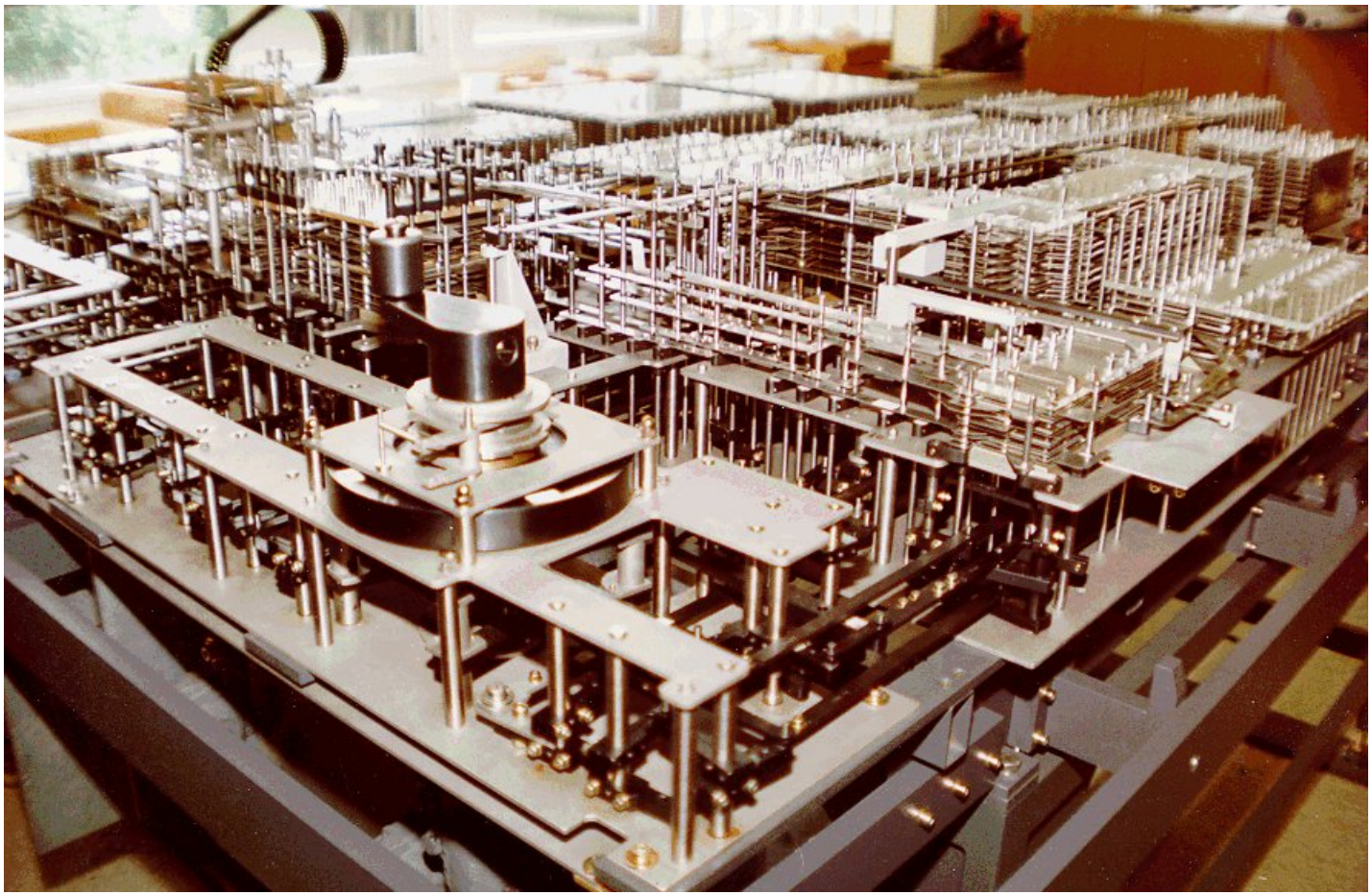
Analytical Engine

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



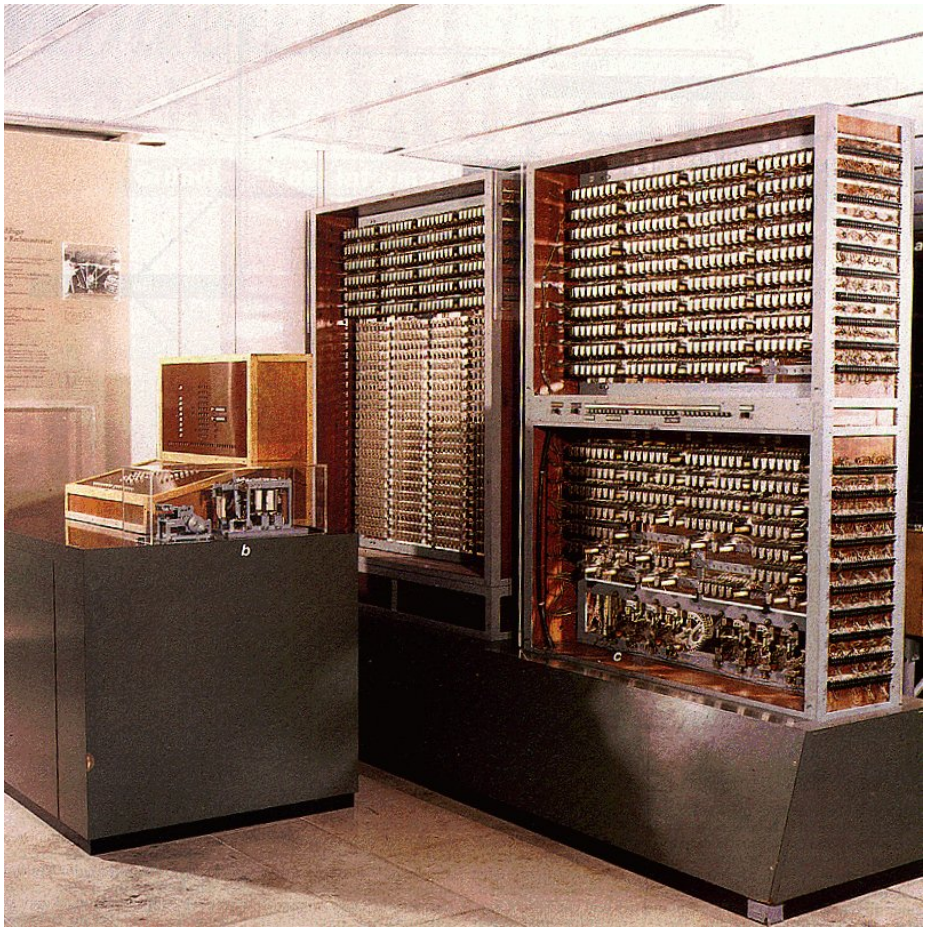
Zuse Z1

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



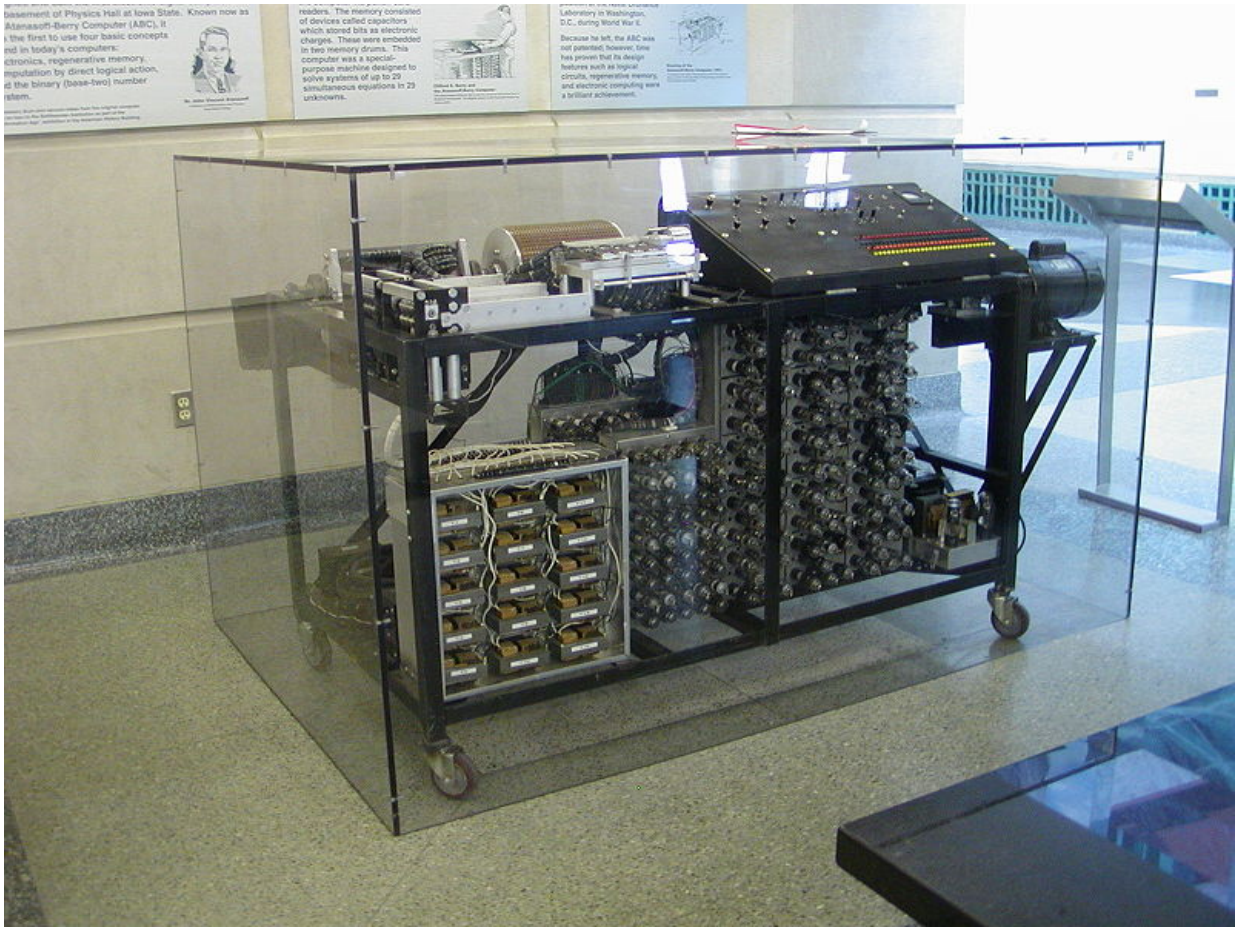
Zuse Z3

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



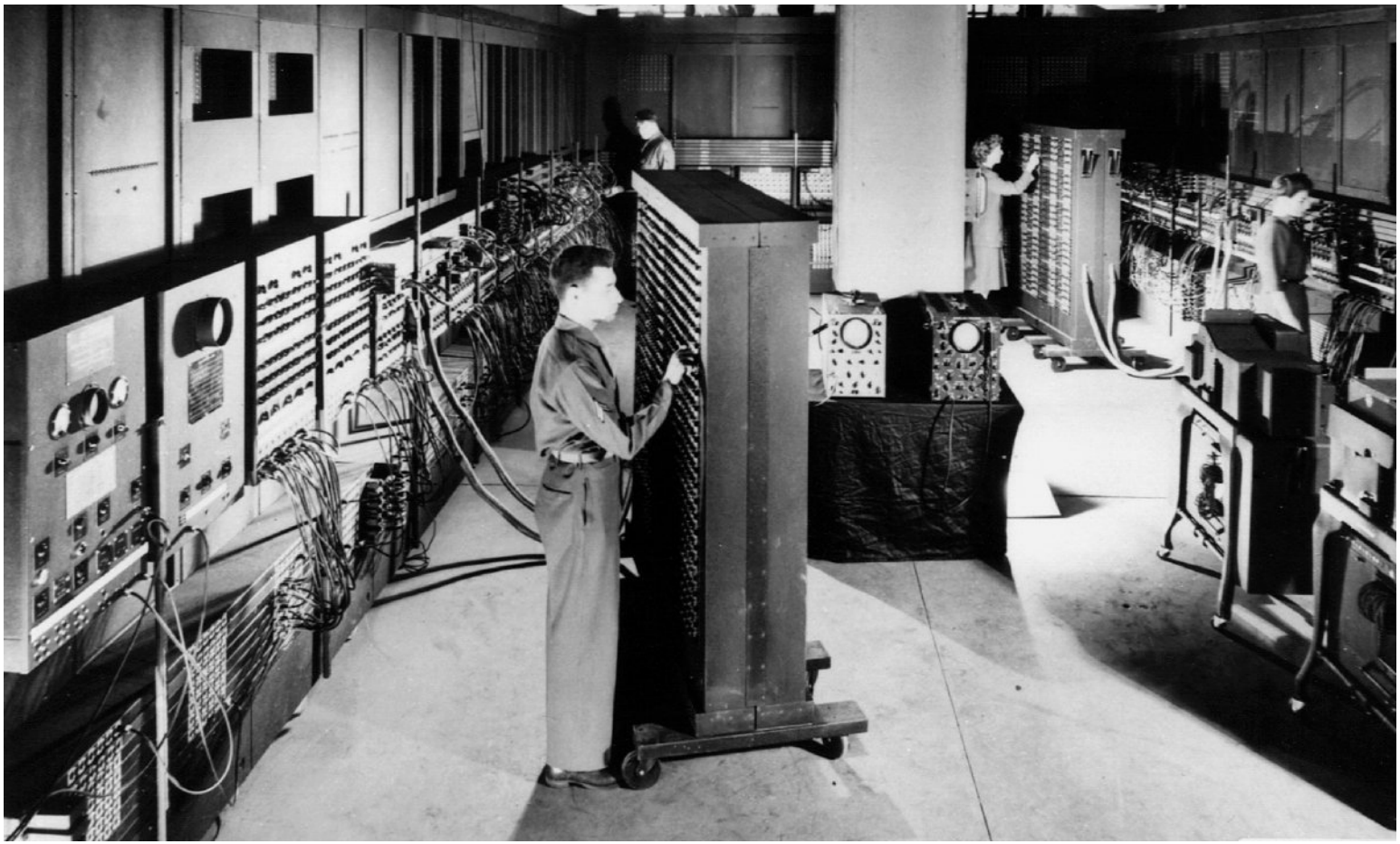
Atanasoff-Berry Computer (ABC)

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

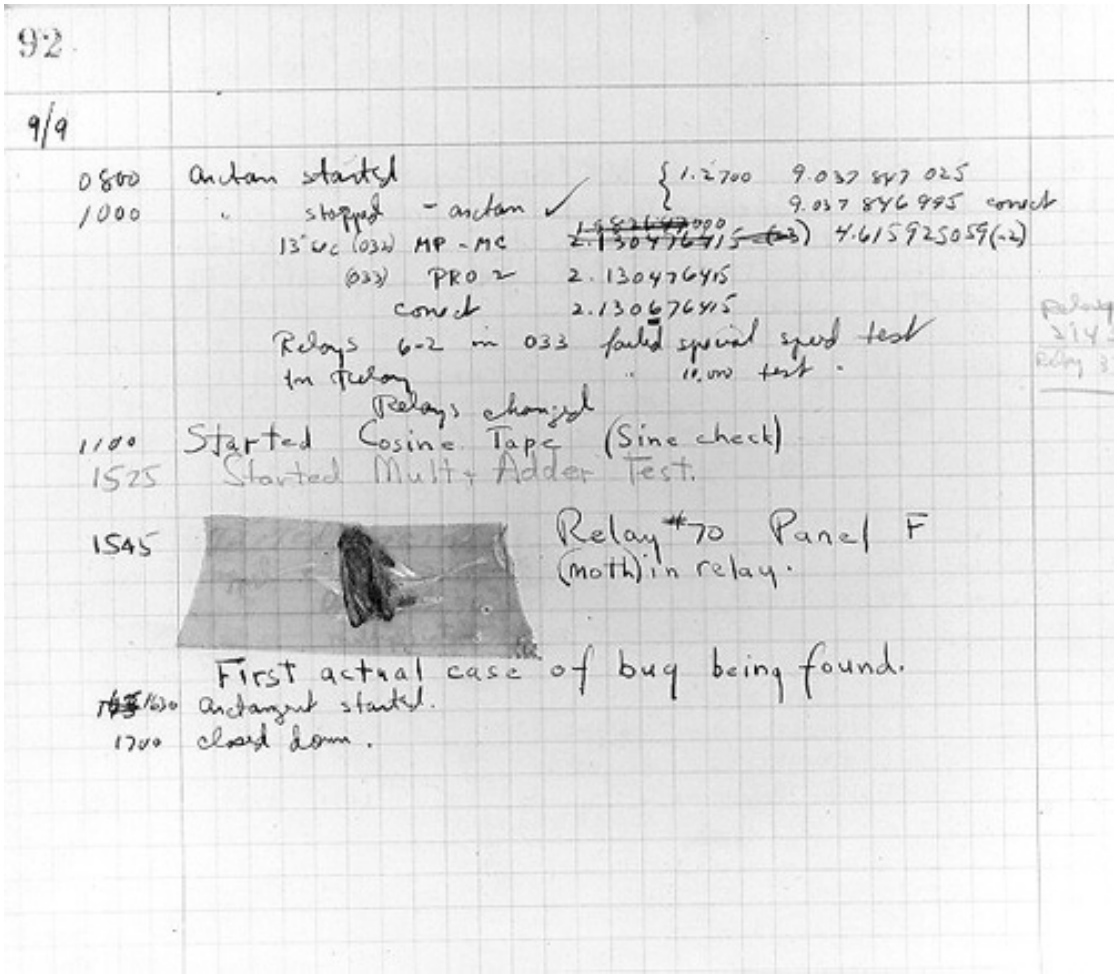


ENIAC — Electronic Numerical Integrator and Computer

Mauchly & Eckert, 1946: Röhren, Steckbrett-Programm

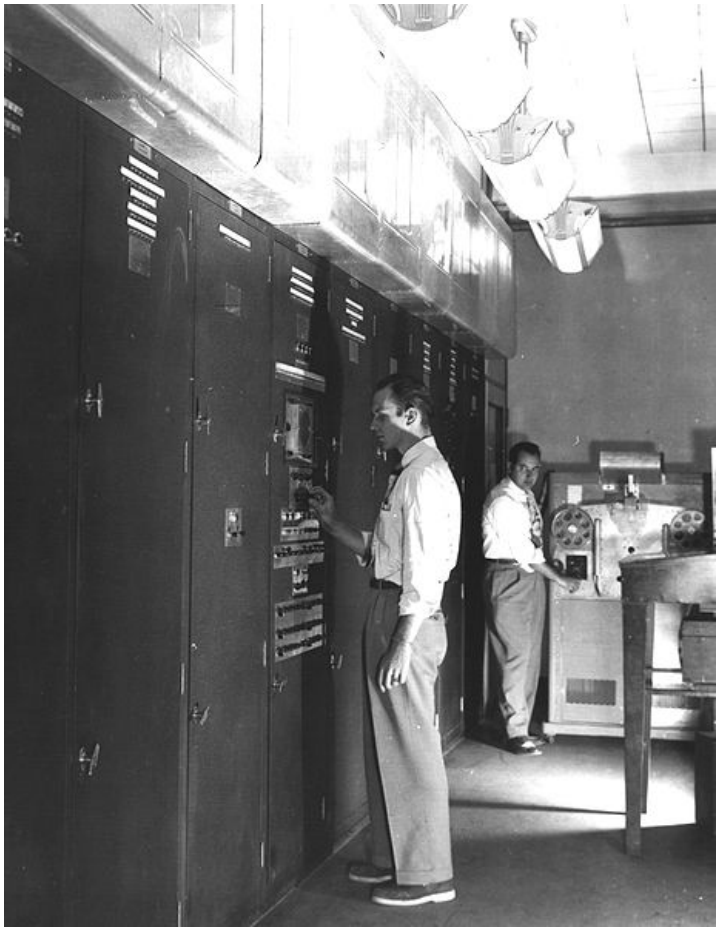


First computer bug



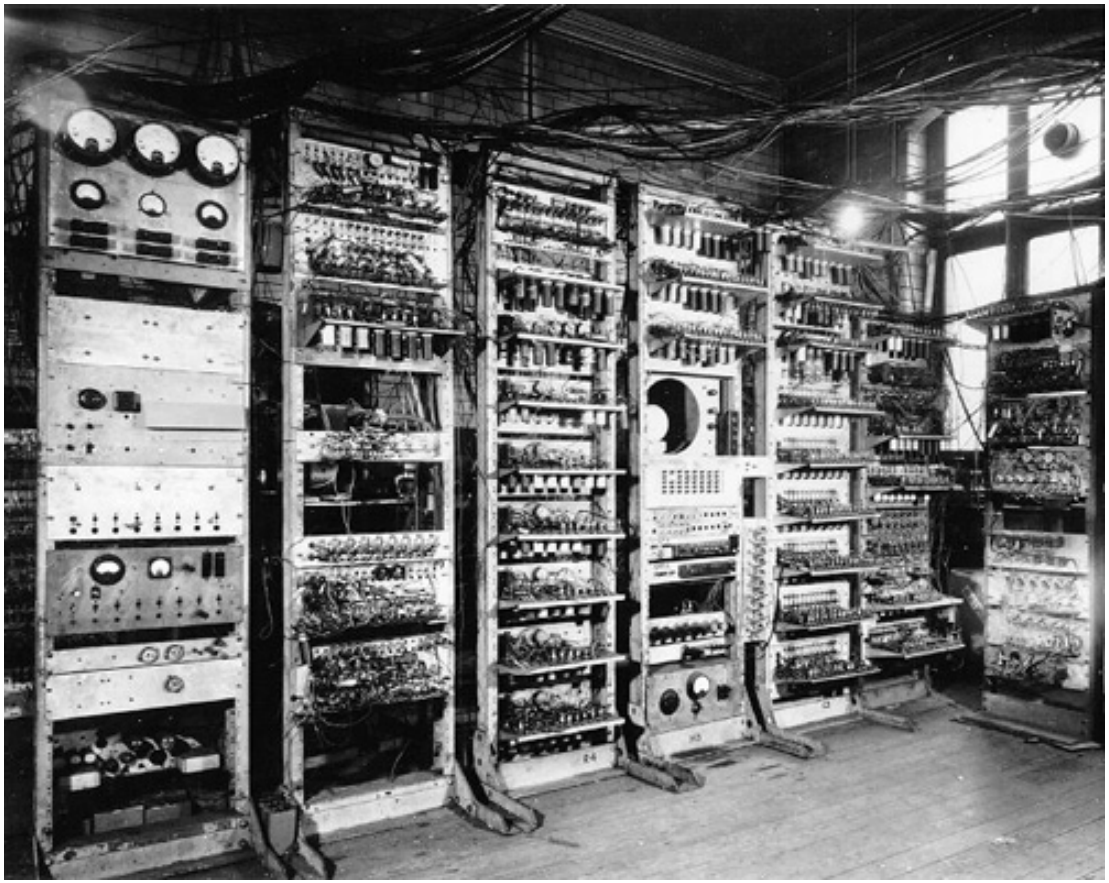
EDVAC

Mauchly, Eckert & von Neumann, 1949: Röhren, speicherprogrammiert



Manchester Mark-1

Williams & Kilburn, 1949: Trommelspeicher, Indexregister



Manchester EDSAC

Wilkes 1951: Mikroprogrammierung, Unterprogramme, speicherprogrammiert

Timeline: Verbesserungen

1952: IBM 701	Pipeline
1964: IBM S/360	Rechnerfamilie, software-kompatibel
1971: Intel 4004	4-bit Mikroprozessor
1972: Intel 8008	8-bit Mikrocomputer-System
1978: Intel 8086	16-bit Mikroprozessor
1979: Motorola 68000	16/32-bit Mikroprozessor
1980: Intel 8087	Gleitkomma-Koprozessor
1981: Intel 8088	8/16-bit für IBM PC
1984: Motorola 68020	32-bit, Pipeline, on-chip Cache
1992: DEC Alpha AXP	64-bit RISC-Mikroprozessor
1997: Intel MMX	MultiMedia eXtension Befehlssatz
2006: Sony Playstation 3	1+8 Kern-Multiprozessor
2006: Intel-VT / AMD-V	Virtualisierung

...

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 sog. **CISC**-Rechner (68000, 8086, VAX)

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

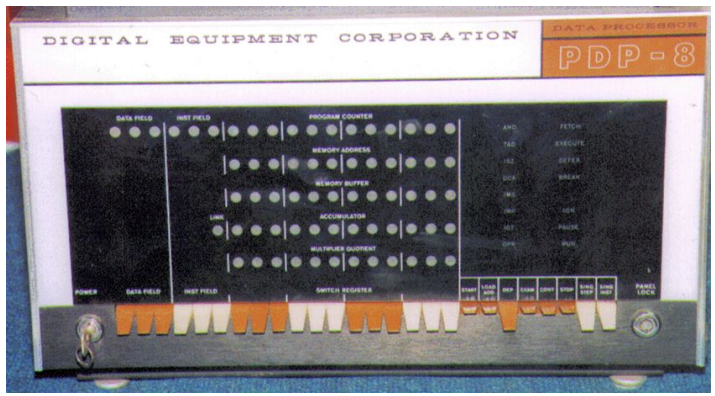
Einführung von **Betriebssystemen**

- „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 Digial Equipment Corporation PDP-1 (1961)
  - 4K Speicher (4096 Worte á 18-bit)
  - 200 kHz Taktfrequenz
  - 120 000 $
  - Grafikdisplay: erste Computerspiele
- Nachfolger PDP-8: 16 000 $
  - erstes Bussystem
  - 50 000 Stück verkauft



Festplatten

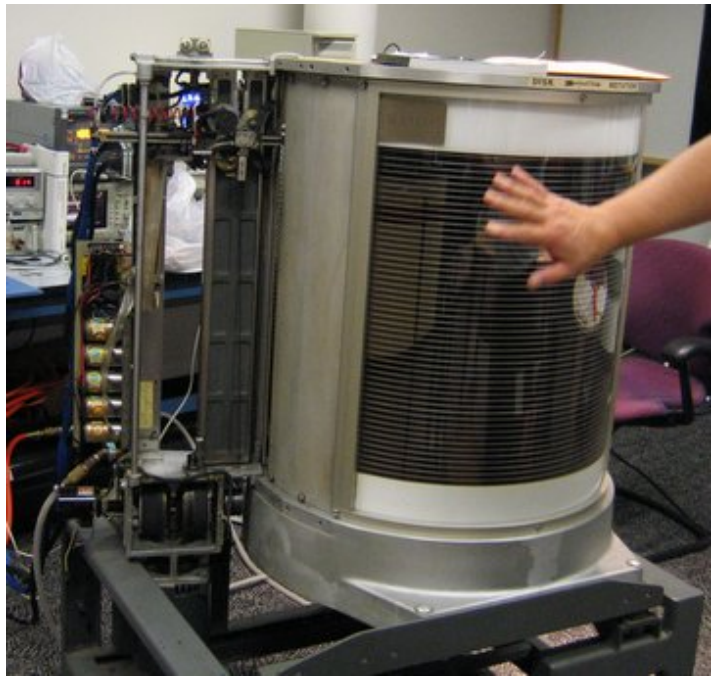
Massenspeicher bei frühen Computern:

- Lochkarten
- Lochstreifen
- Magnetband

- Magnettrommel
- Festplatte

  IBM 350 RAMAC (1956)

  5 MByte, 600 ms Zugriffszeit



http://de.wikibooks.org/wiki/Computerhardware_für_Anfänger

dritte Generation: ICs

- Erfindung der integrierten Schaltung 1958 (Noyce, 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

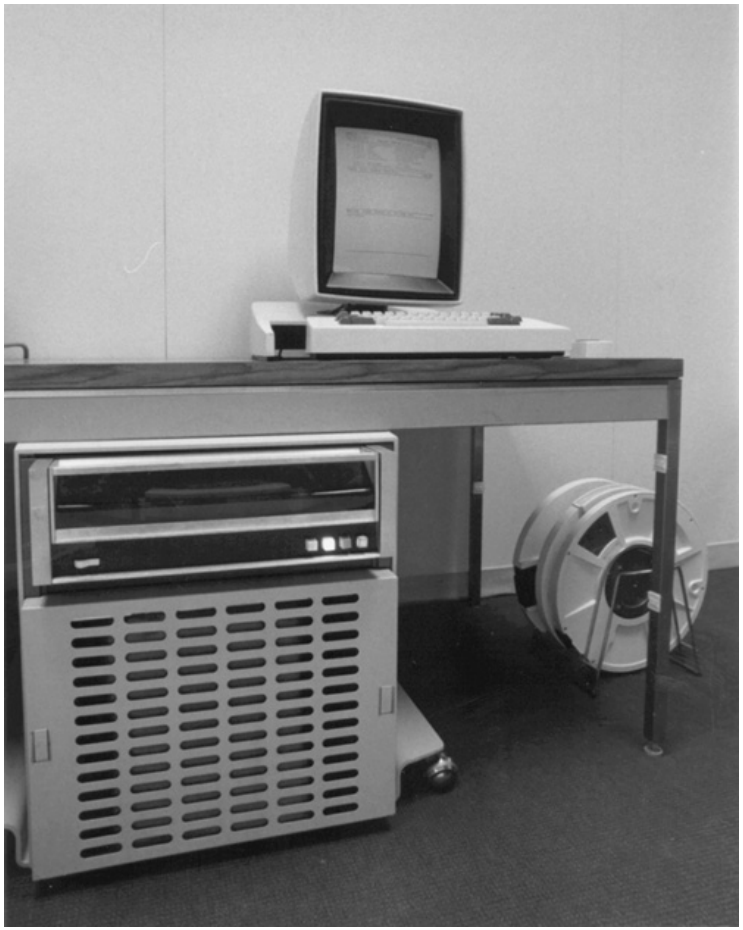
- VLSI = *Very Large Scale Integration*
- ab 10 000 Transistoren pro Chip

- 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



Rechner-Spektrum

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

Literatur

[TA14] A.S. Tanenbaum, T. Austin: *Rechnerarchitektur – Von der digitalen Logik zum Parallelrechner*.
6. Auflage, Pearson Deutschland GmbH, 2014.
ISBN 978–3–86894–238–5

[HenHA] N. Hendrich: *HADES — HAmburg DEsign System*.
Universität Hamburg, FB Informatik, Lehrmaterial.
`tams.informatik.uni-hamburg.de/applets/hades`

Kapitel 1

Moore's Law

- bessere Technologie ermöglicht immer kleinere Transistoren
- Materialkosten sind proportional zur Chipfläche

$\Rightarrow$ bei gleicher Funktion kleinere und billigere Chips

$\Rightarrow$ bei gleicher Größe leistungsfähigere Chips

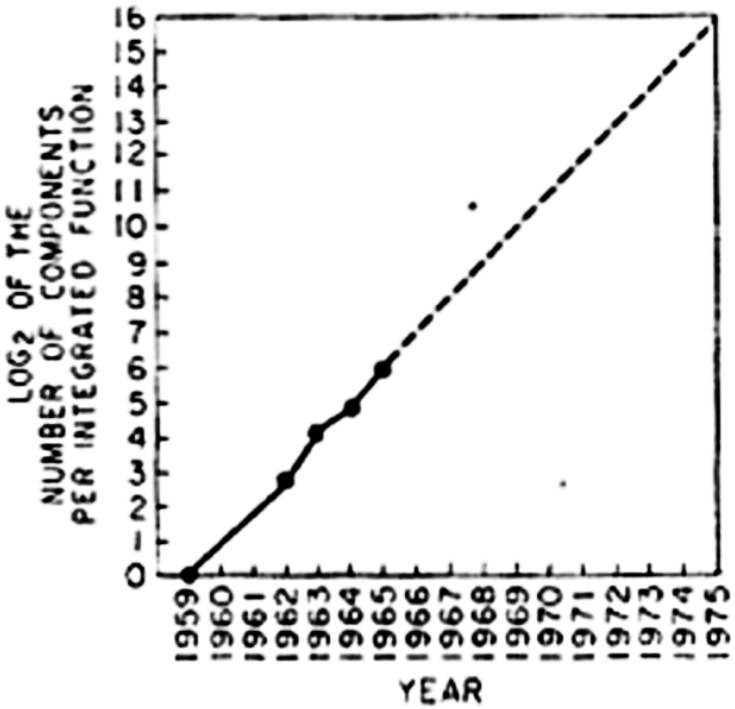
Moore's Law — Gordon Moore, Mitgründer von Intel, 1965

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.)

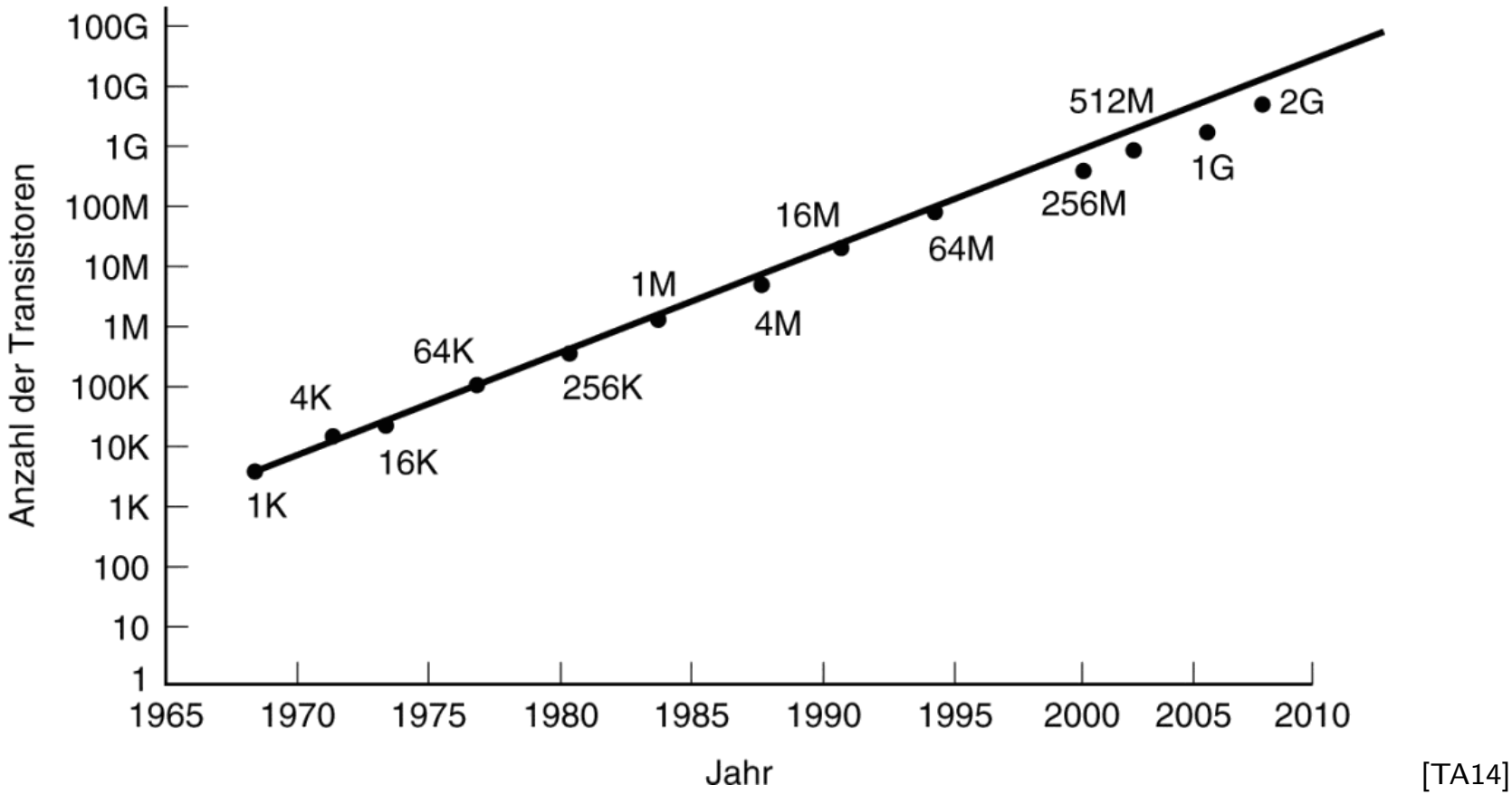


Gordon Moore, 1965, [Moo65]:
Cramming more components onto integrated circuits

Wird das so weitergehen?

- Vorhersage gilt immer noch
- „ITRS" Prognose bis über Jahr 2025 hinaus [ITRS13]

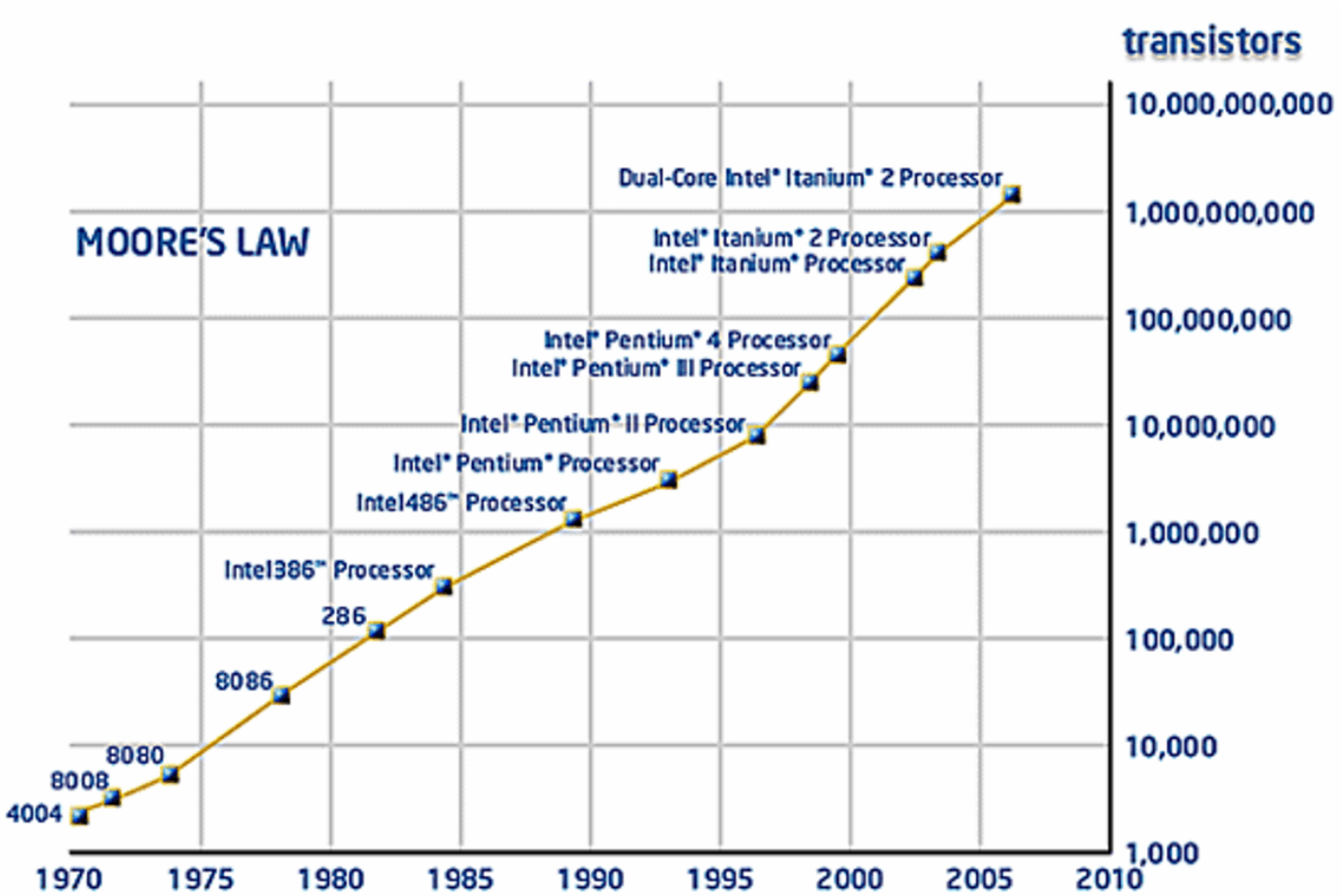
Moore's Law: Transistoren pro Speicherchip



[TA14]

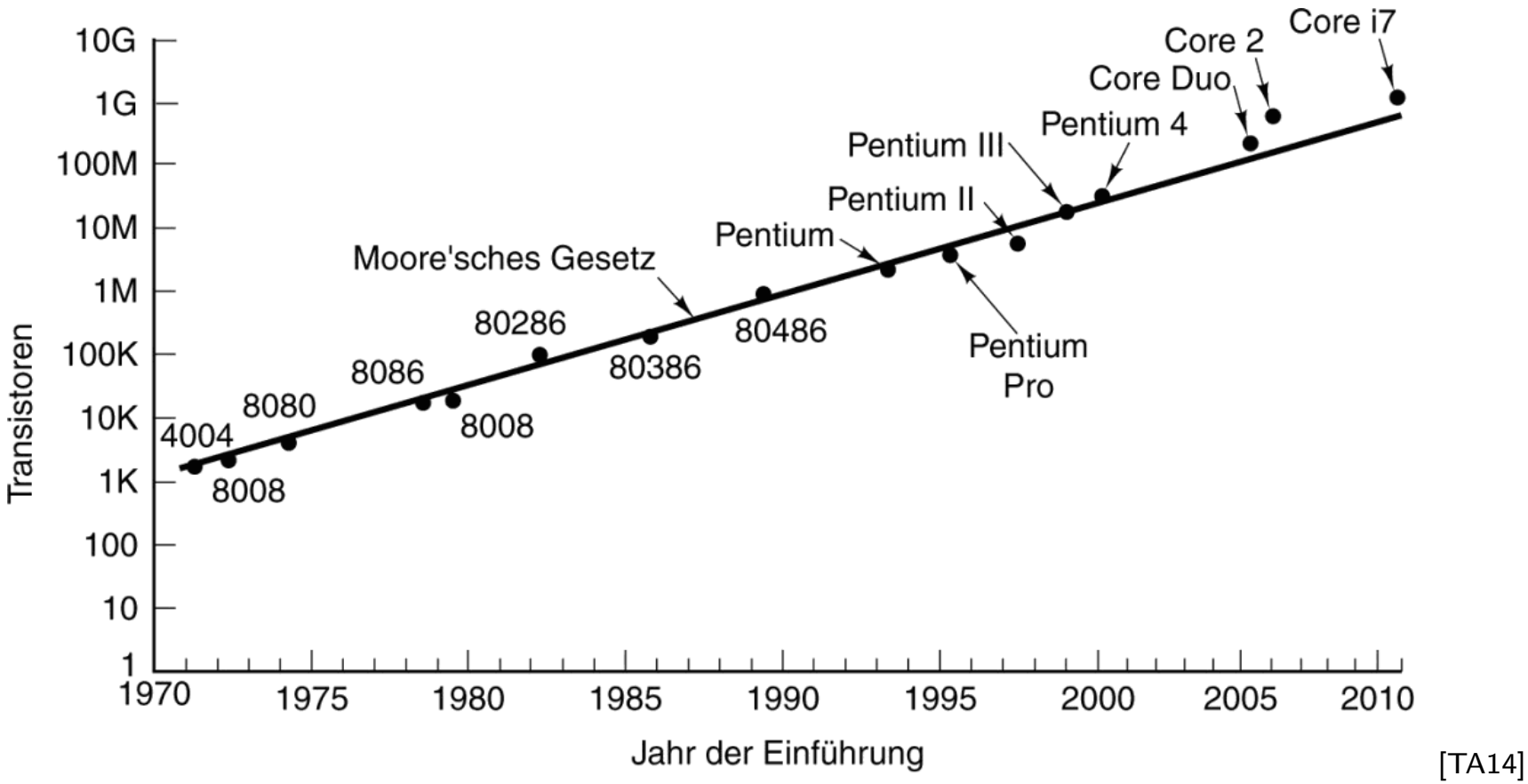
- Vorhersage: 60% jährliches Wachstum der Transistoranzahl pro IC

Moore's Law: Evolution des Intel x86 (bis 2010)



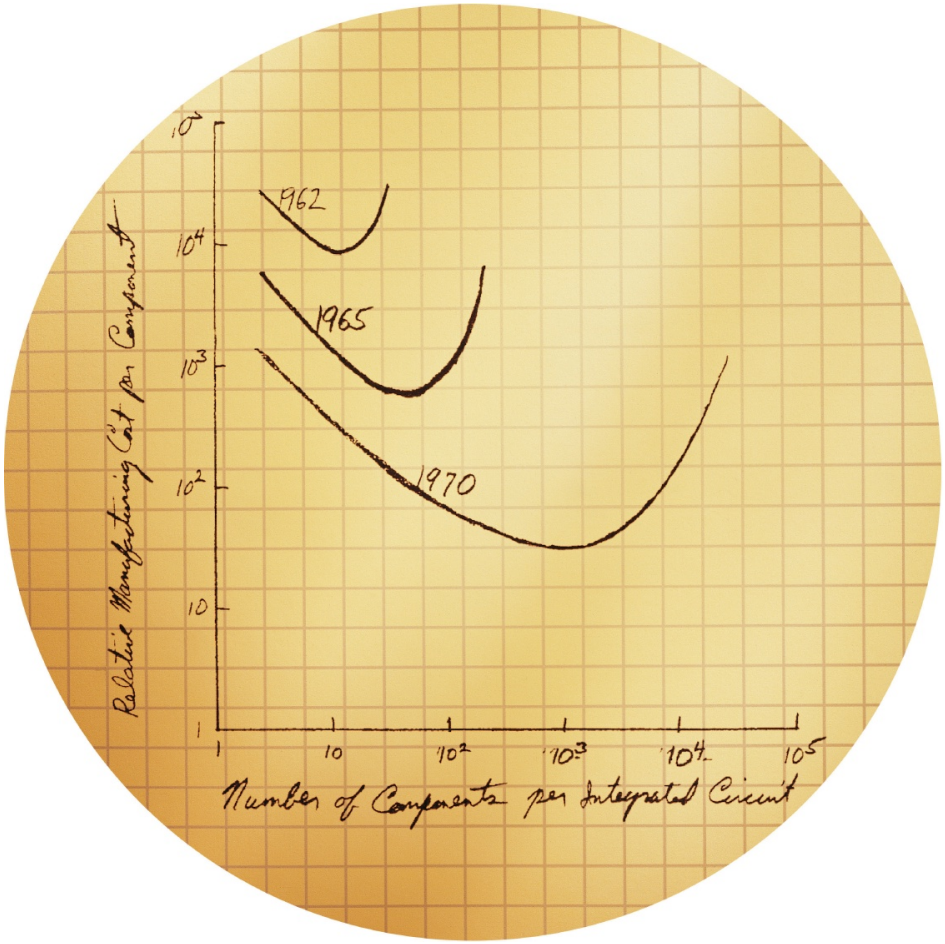
[Intel]

Moore's Law: Evolution des Intel x86 (bis 2010) (cont.)



[TA14]

Moore's Law: Kosten pro Komponente



Originalskizze von G. Moore [Intel]

Moore's Law: Formel und Beispiele

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

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/list 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 Super MUC	(Xeon E5-2680 8C 2,7GHz)	2 897,0	147 456
2012	IBM BlueGene/Q Sequoia	(Power BQC 16C 1,6GHz)	16 324,8	1 572 864
2013	IBM BlueGene/Q JUQUEEN	(Power BQC 16C 1,6GHz)	5 008,9	458 752
2013	NUDT Tianhe-2	(Xeon E5-2692 12C 2,2 GHz + Xeon Phi 31S1P)	33 862,7	3 120 000

Leistungssteigerung der Spitzenrechner seit 1993

www.top500.org/list 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 Super MUC	(Xeon E5-2680 8C 2,7GHz)	2 897,0	147 456	3 423
2012	IBM BlueGene/Q Sequoia	(Power BQC 16C 1,6GHz)	16 324,8	1 572 864	7 890
2013	IBM BlueGene/Q JUQUEEN	(Power BQC 16C 1,6GHz)	5 008,9	458 752	2 301
2013	NUDT Tianhe-2	(Xeon E5-2692 12C 2,2 GHz + Xeon Phi 31S1P)	33 862,7	3 120 000	17 808

Aktuelle Spitzenrechner

TOP500® JUNE 2014

PRESENTED BY

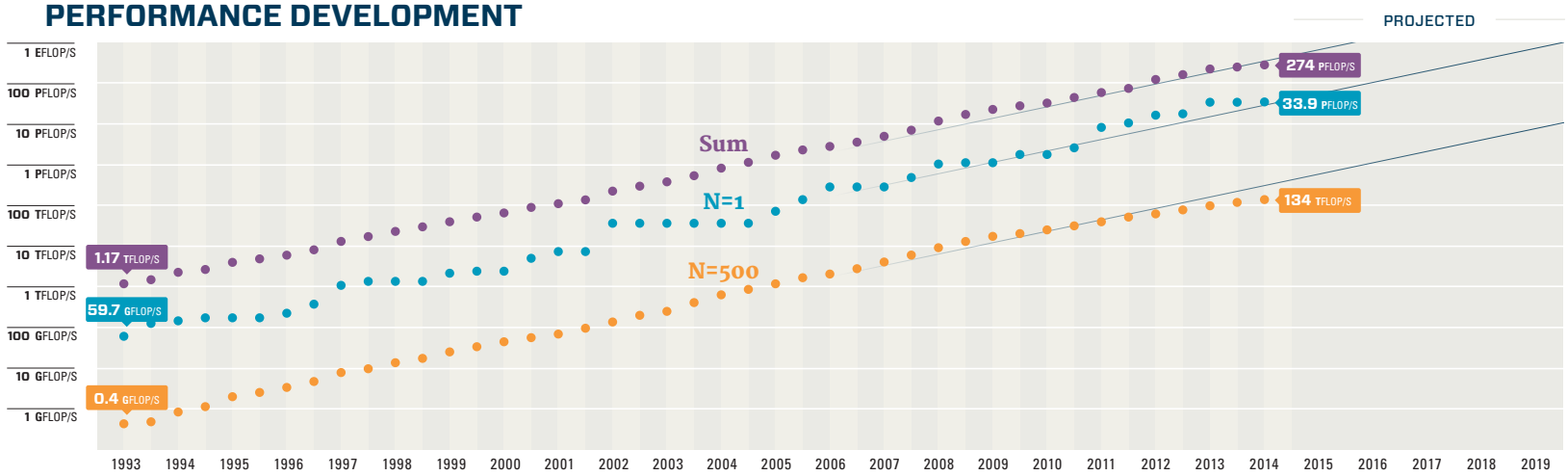
ICL UT INNOVATIVE COMPUTING LABORATORY the UNIVERSITY of TENNESSEE

BERKELEY LAB Lawrence Berkeley National Laboratory

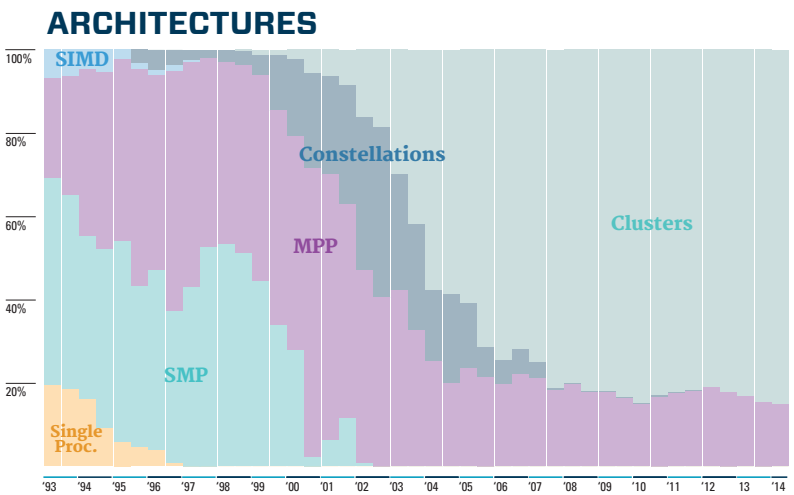
PROMETEUS PROFESSOR MEUER TECHNOLOGIEBERATUNG

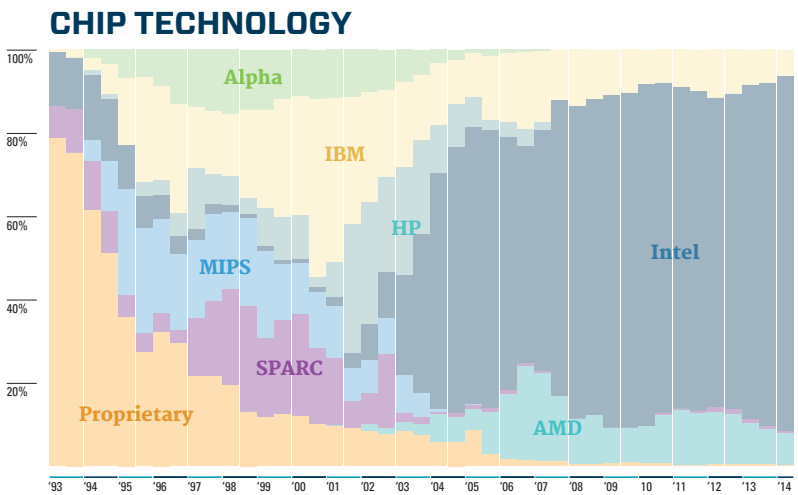
FIND OUT MORE AT www.top500.org

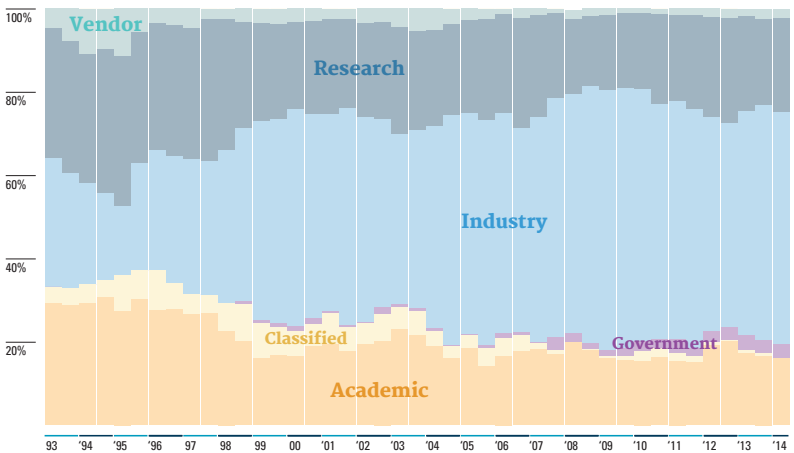
	NAME	SPECS	SITE	COUNTRY	CORES	RMAX PFLOP/S	POWER MW
1	Tianhe-2 (Milkyway-2)	NUDT, Intel Ivy Bridge (12C, 2.2 GHz) & Xeon Phi (57C, 1.1 GHz), Custom interconnect	NSCC Guangzhou	China	3,120,000	33.9	17.8
2	Titan	Cray XK7, Operon 6274 (16C 2.2 GHz) + Nvidia Kepler GPU, Custom interconnect	DOE/SC/ORNL	USA	560,640	17.6	8.2
3	Sequoia	IBM BlueGene/Q, Power BQC (16C 1.60 GHz), Custom interconnect	DOE/NNSA/LLNL	USA	1,572,864	17.2	7.9
4	K computer	Fujitsu SPARC64 VIIIfx (8C, 2.0GHz), Custom interconnect	RIKEN AICS	Japan	705,024	10.5	12.7
5	Mira	IBM BlueGene/Q, Power BQC (16C, 1.60 GHz), Custom interconnect	DOE/SC/ANL	USA	786,432	8.59	3.95

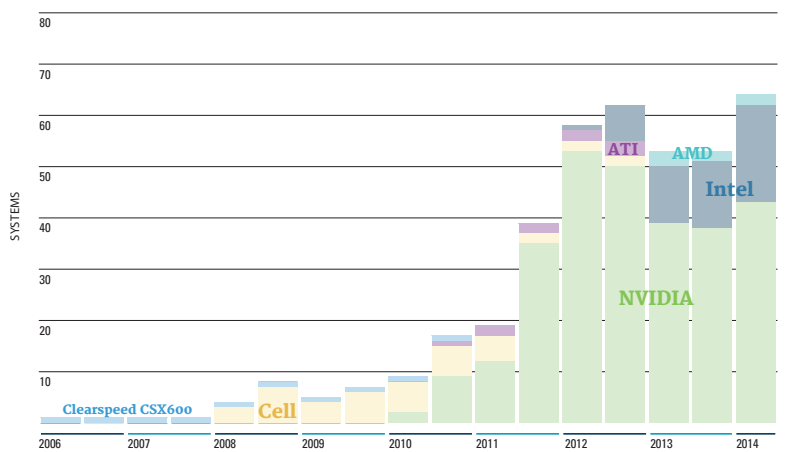


Aktuelle Spitzenrechner (cont.)









Moore's Law: Trends

- Miniaturisierung schreitet weiter fort
- aber Taktraten erreichen physikalisches Limit
- steigender Stromverbrauch, Leckströme

Entwicklungen

- 4 GByte Hauptspeicher (und mehr) sind Standard
- Übergang von 32-bit auf 64-bit Adressierung

$\Rightarrow$ Integration mehrerer CPUs auf einem Chip (Dual-/Quad-Core)

$\Rightarrow$ zunehmende Integration von Peripheriegeräten

$\Rightarrow$ seit 2011: CPU plus leistungsfähiger Grafikchip

$\Rightarrow$ **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

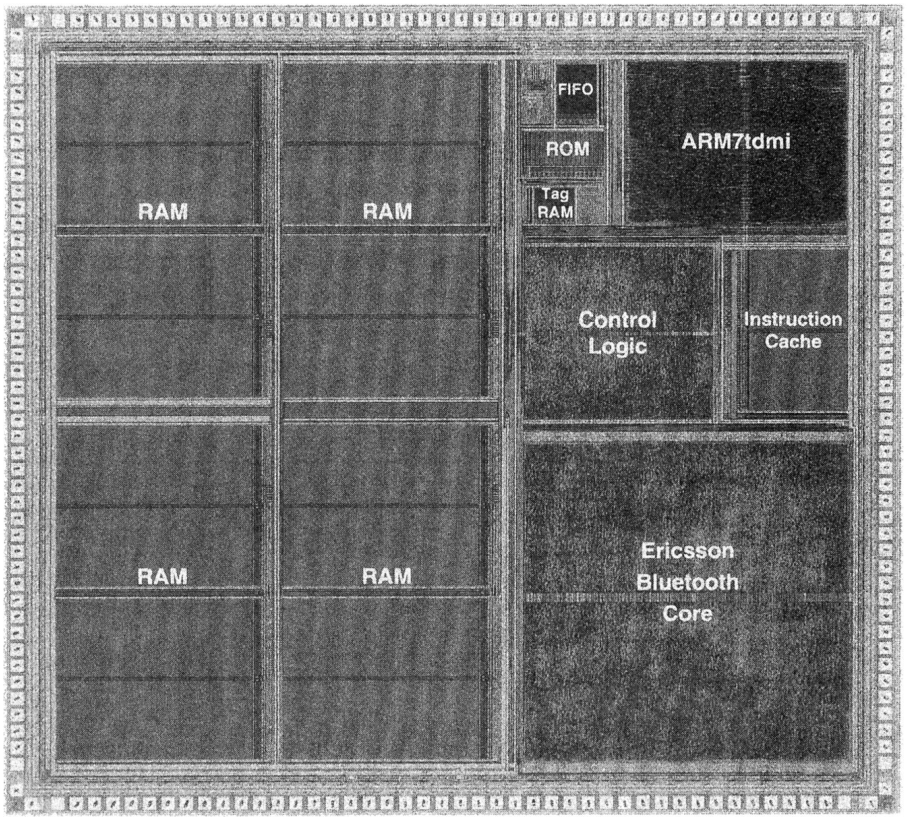
  $\Rightarrow$ z.B. ARM mit *big.LITTLE* Konzept
- Cache Hierarchie: 1-Level D- und I-Cache / 2-Level
- dedizierte Prozessoren: Grafik, Video(de)kodierung, DSP ...
- Hauptspeicher (evtl. auch extern), Speichercontroller
- weitere Speicher für Medien/Netzwerkoperationen

SoC: System on a chip (cont.)

- Peripherieblöcke nach Kundenwunsch konfiguriert:
  - Displayansteuerung: DP, HDMI ...
  - 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, 4G ...
- Smartphones, Tablet-Computer, Medien-/DVD-Player, WLAN-Router, NAS-/Home-Server ...

SoC Beispiele

- Bluetooth-Controller (2000)

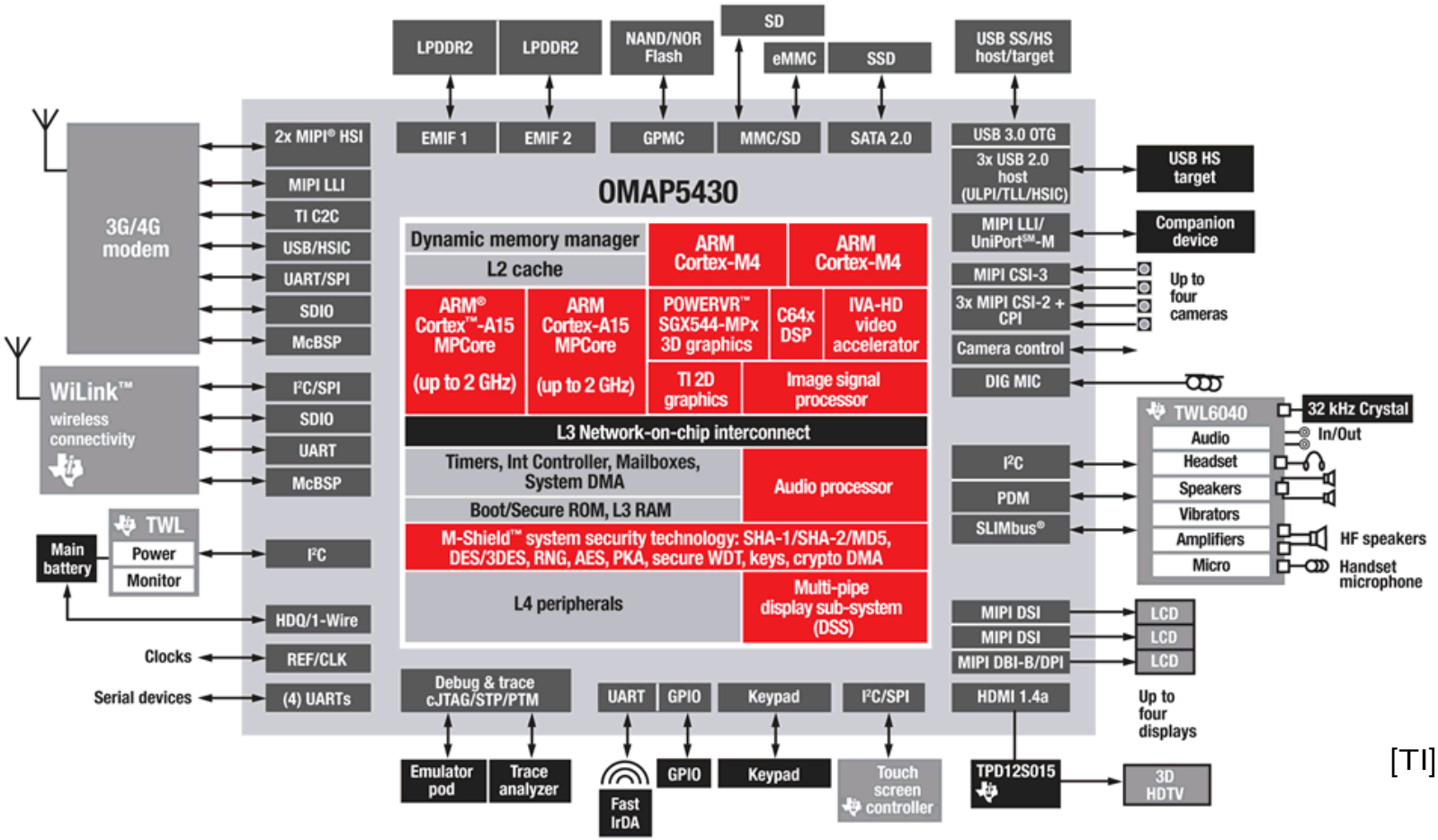


[Fur00]

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.)

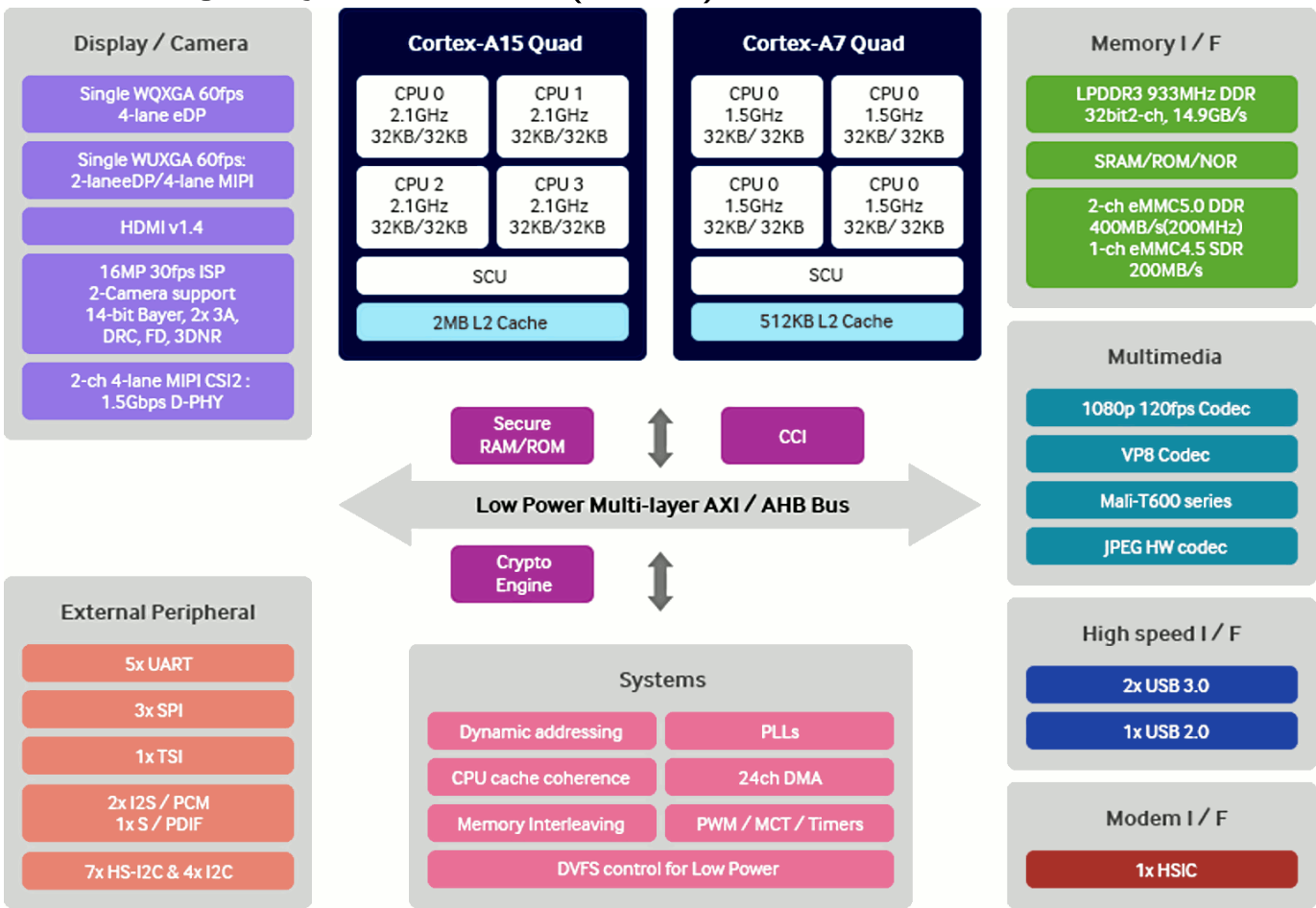
- Texas Instruments OMAP 5430 (2011)



[TI]

SoC Beispiele (cont.)

- Samsung Exynos-5422 (2014)



[Samsung]

Smart Dust

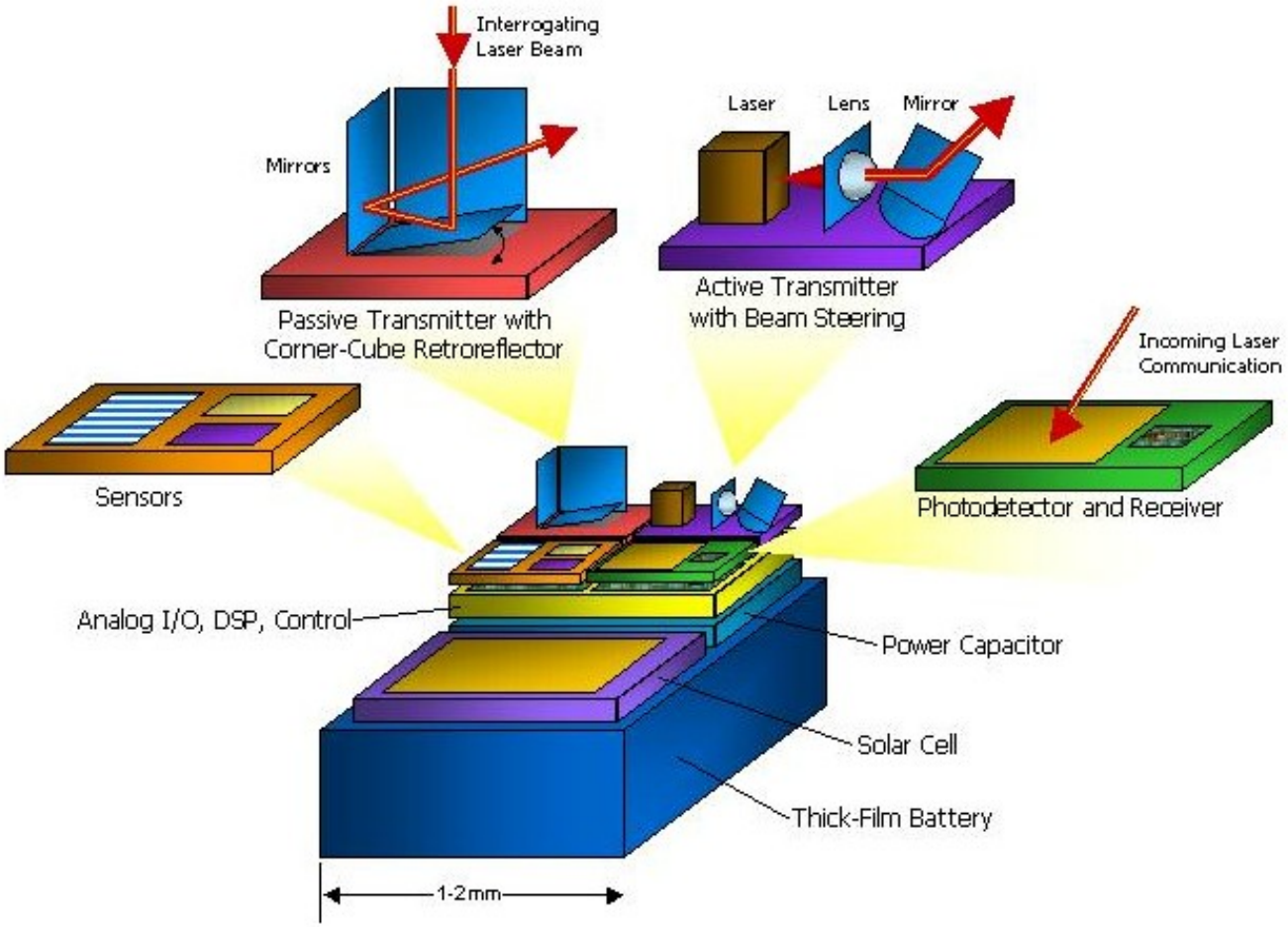
Wie klein kann man Computer bauen?

- Berkeley Projekt: **Smart Dust** 1997-2002
- Integration kompletter Rechensysteme auf 1 mm^3
  - vollständiger Digitalrechner CPU, Speicher, I/O
  - Sensoren Photodioden, Kompass, Gyro
  - Kommunikation Funk, optisch
  - Stromversorgung Photozellen, Batterie, Vibration, Mikroturbine
  - Echtzeit-Betriebssystem Tiny OS
  - inklusive autonome Vernetzung

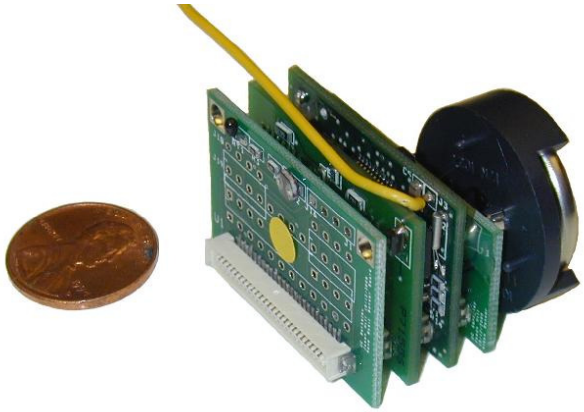
- Massenfertigung? Tausende autonome Mikrorechner
- „Ausstreuen“ in der Umgebung
- vielfältige Anwendungen

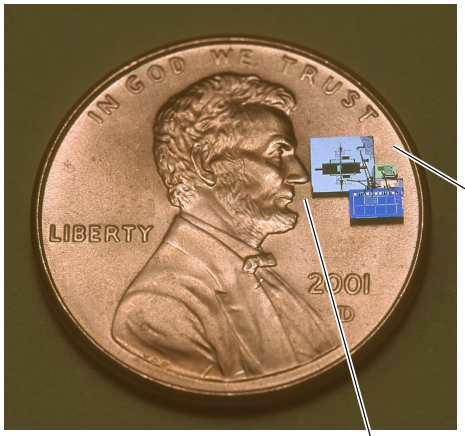
Berkeley Sensor & Actuator Center, `robotics.eecs.berkeley.edu/~pister/SmartDust`

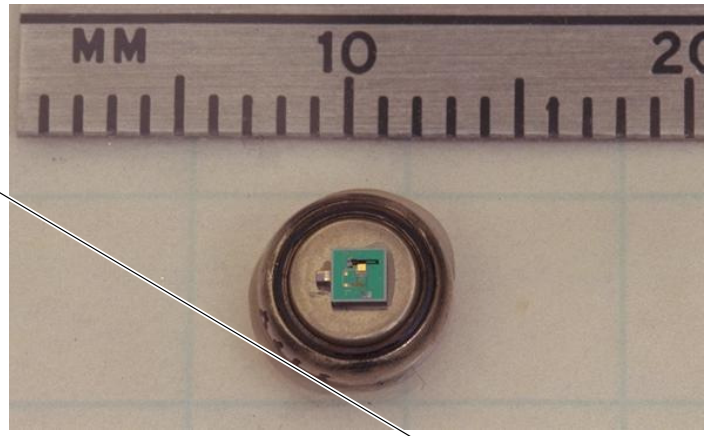
Smart Dust: Konzept

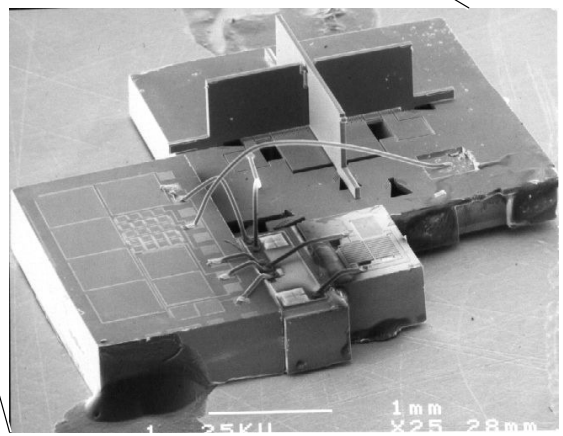


Smart Dust: Prototypen







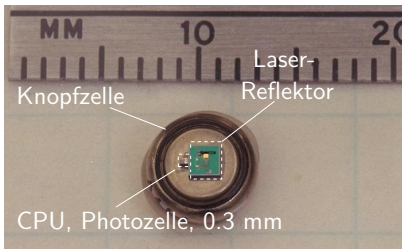


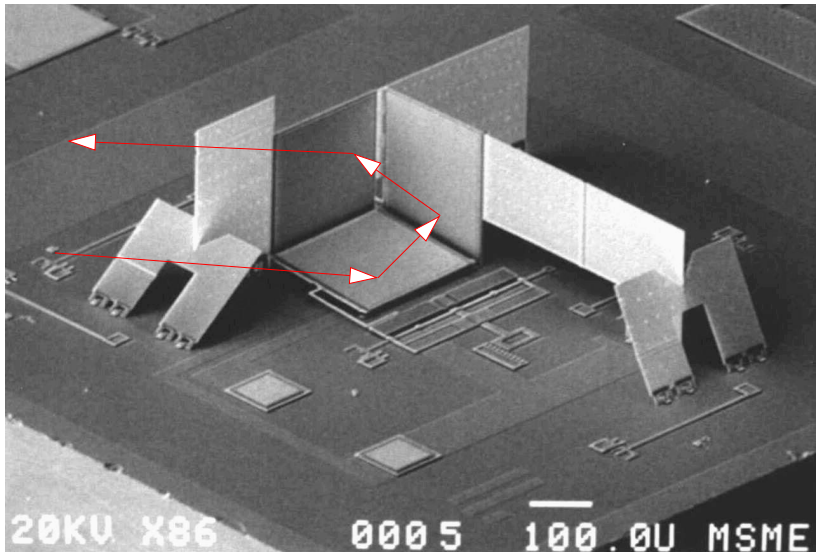
diverse Prototypen:

- vollwertige CPU / Sensoren / RF
- "out-door"-tauglich

- MEMS-"CCR" für opt. Kommunikation

Smart Dust: Corner-cube reflector („Katzenauge“)



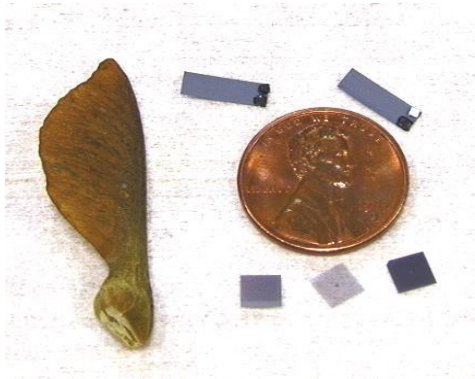


- CCR: seitlich zwei starre Spiegel, Gold auf Silizium
- untere Spiegelfläche beweglich (elektrostatisch, ca. 30 V)
- gezielte Modulation von eingestrahltem Laserlicht
- Reichweiten > 100 m demonstriert

robotics.eecs.berkeley.edu/~pister/SmartDust

Smart Dust: Energieverbrauch

Miniatur-Solarzellen
Wirkungsgrad ca. 3%
26 µW/mm in vollem Sonnenlicht



Batterien:	~ 1J/mm³		
Kondensatoren:	~10 mJ/mm³		
Solarzellen:	~ 0.1 mW/mm	~ 1J/mm /day	(außen,Sonne)
	~ 10 µW/mm	~ 10mJ/mm /day	(innen)
Digitalschaltung	1 nJ/instruction	(StrongArm SA1100)	
Analoger Sensor	1 nJ/sample		
Kommunikation	1 nJ/bit	(passive transmitter, s.u.)	
opt. digitale ASICs:	~ 5 pJ/bit	(LFSR Demonstrator, 1.4V)	

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 gespeichert würde
  - Ab diesem Zeitpunkt gibt es bessere Performance 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: ITRS

International **T**echnology **R**oadmap for **S**emiconductors

`http://www.itrs.net/reports.html`

- non-profit Organisation
- diverse Fördermitglieder
  - Halbleiterhersteller
  - Geräte-Hersteller
  - Unis, Forschungsinstitute
  - Fachverbände aus USA, Europa, Asien
- Jährliche Publikation einer langjährigen Vorhersage
- Zukünftige Entwicklung der Halbleitertechnologie
- Komplexität typischer Chips (Speicher, Prozessoren, SoC, ...)
- Modellierung, Simulation, Entwurfssoftware

Roadmap: ITRS (cont.)

Table ORTC-2D High-Performance MPU and ASIC Product Generations and Chip Size Model

Year of Production	2011	2012	2013	2014	2015	2016	2017	2018	2019	2020	2021	2022	2023	2024	2025	2026
Flash ½ Pitch (nm) (un-contacted Poly)	22	20	18	17	15	14,2	13,0	11,9	10,9	10,0	8,9	8,0	8,0	8,0	8,0	8,0
DRAM ½ Pitch (nm) (contacted)	36	32	28	25	23	20,0	17,9	15,9	14,2	12,6	11,3	10,0	8,9	8,0	7,1	6,3
MPU/ASIC Metal 1 (M1) ½ Pitch (nm)	38	32	27	24	21	18,9	16,9	15,0	13,4	11,9	10,6	9,5	8,4	7,5	6,7	6,0
MPU High-Performance Printed Gate Length (nm)	35	31	28	25	22	19,8	17,7	15,7	14,0	12,5	11,1	9,9	8,8	7,9	6,79	5,87
MPU High-Performance Physical Gate Length (nm)	24	22	20	18	17	15,3	14,0	12,8	11,7	10,6	9,7	8,9	8,1	7,4	6,6	5,9
Logic (Low-volume Microprocessor) High-performance																
Generation at Introduction	p13h	p13h	p16h	p16h	p16h	p19h	p19h	p19h	p22h	p22h	p22h	p25h	p25h	p25h	p28h	p28h
Functions per chip at introduction (million transistors)	8.848	8.848	17.696	17.696	17.696	35.391	35.391	35.391	70.782	70.782	70.782	141.564	141.564	141.564	283.128	283.128
Chip size at introduction (mm^2)	520	368	520	413	328	520	413	328	520	413	328	520	413	328	520	413
Generation at production	p11h	p11h	p13h	p13h	p13h	p16h	p16h	p16h	p19h	p19h	p19h	p22h	p22h	p22h	p25h	p25h
Functions per chip at production (million transistors)	4.424	4.424	8.848	8.848	8.848	17.696	17.696	17.696	35.391	35.391	35.391	70.782	70.782	70.782	141.564	141.564
Chip size at production (mm^2)	260	184	260	206	164	260	206	164	260	206	164	260	206	164	260	206
OH % of Total Chip Area	29,5%	29,5%	29,5%	29,5%	29,5%	29,5%	29,5%	29,5%	29,5%	29,5%	29,5%	29,5%	29,5%	29,5%	29,5%	29,5%
Logic Core+SRAM (Without OH Average Density (Mt/cm2)	2.414	3.414	4.828	6.083	7.664	9.656	12.166	15.328	19.312	24.332	30.656	38.625	48.664	61.313	77.249	97.328
High-performance MPU Mtransistors/cm^2 (including on-chip SRAM)	1.701	2.406	3.403	4.287	5.402	6.806	8.575	10.804	13.612	17.150	21.608	27.224	34.300	43.215	54.448	68.600
ASIC																
ASIC usable Mtransistors/cm^2 (auto layout)	1.701	2.406	3.403	4.287	5.402	6.806	8.575	10.804	13.612	17.150	21.608	27.224	34.300	43.215	54.448	68.600
ASIC max chip size (mm^2) (max. lithographic field size)	858	858	858	858	858	858	858	858	858	858	858	858	858	858	858	858
ASIC max. functions per chip (Mtransistors/chip) (fit in litho.	14.599	20.646	29.198	36.787	46.348	58.395	73.573	92.697	116.790	147.147	185.393	233.581	294.293	370.786	467.162	588.587

Moore's Law

Beispiel für die Auswirkung von Moore's Law

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

Wie lösen wir das Problem ?

Moore's Law: Schöpferische Pause

Beispiel für die Auswirkung von Moore's Law

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 ?*

Moore's Law: Schöpferische Pause

Beispiel für die Auswirkung von Moore's Law

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 $1{,}6 \times 1{,}6$ 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.

Alternativen zum Digitalrechner?

- Analogrechner
- Neuronale Netzwerke (Vorbild: Gehirn)
- Quantencomputer

Analogrechner

- Berechnung durch Kombination mechanischer oder elektrischer Komponenten
- kontinuierliche (analoge) Repräsentation
- parallele Verarbeitung aller Komponenten: schnell
- aber geringe Genauigkeit (Rauschen, Justierung, Drift)

- nicht frei programmierbar
- sondern für spezifische Aufgabe *verdrahtet*

- keine praktische Bedeutung mehr
- statt dessen: digitale Berechnung mit Signalprozessoren (DSP)
- Regelungstechnik / Systemtheorie

Neuronale Netzwerke

- Struktur („Netzwerk“) aus Komponenten („Neuronen“)
- die Verbindungen über einstellbare Kopplungen („Synapsen“)
- Verbindungsstärken nicht programmiert sondern gelernt
- unzählige Varianten
  - ein, zwei, drei, oder mehr Schichten („layers“)
  - mit oder ohne Rückkopplungen der Schichten
  - Neuronen als abstrakte Modelle oder detailliert bestimmten Nervenzellen nachempfunden
  - Vielzahl verschiedener Lernalgorithmen
- Aktueller Hype „Deep Learning“
  - mehrschichtige Netzwerke auf großen Datenmengen trainiert
  - beste bekannte Lösung für viele Klassifikations-Benchmarks

Quantencomputer

- Auswahl eines Quantensystems: Atome, Photonen
- System versucht seine Energie zu minimieren
- dabei *parallele* Berechnung über alle möglichen Zustände
- im Prinzip auch bei exponentieller Anzahl der Zustände (!)
- theoretisch extrem mächtiges Konzept
- aber Experimente derzeit bis ca. 5 QBits (= 32 Zustände)
- (noch) keine Skalierung / Massenproduktion

- erste Anwendungen für sichere Kryptographie
  - Präparation *verschränkter* Photonen
  - Glasfaserübertragung
  - erste Satellitenexperimente

Wie geht es jetzt weiter?

Ab jetzt erstmal ein *bottom-up* Vorgehen: Start mit grundlegenden Aspekten, dann Kennenlernen aller Komponenten des Digitalrechners und Konstruktion eines vollwertigen Rechners.

- Grundlagen der Repräsentation von Information
- Darstellung von Zahlen und Zeichen
- arithmetische und logische Operationen
- ...

- Vorkenntnisse nicht nötig (aber hilfreich)

Literatur

[TA14] A.S. Tanenbaum, T. Austin: *Rechnerarchitektur – Von der digitalen Logik zum Parallelrechner*. 6. Auflage, Pearson Deutschland GmbH, 2014. ISBN 978–3–86894–238–5

[Moo65] G.E. Moore: *Cramming More Components Onto Integrated Circuits*. in: *Electronics* 38 (1965), April 19, Nr. 8

[ITRS13] *International Technology Roadmap for Semiconductors – 2013 Edition*. Semiconductor Industry Association, 2013. `www.itrs.net/Links/2013ITRS/Summary2013.htm`

Literatur (cont.)

[Fur00] S. Furber: *ARM System-on-Chip Architecture*.
2nd edition, Pearson Education Limited, 2000.
ISBN 978–0–201–67519–1

[Intel] Intel Corp.; Santa Clara, CA.
`www.intel.com`
`www.intel.com/content/www/us/en/history/museum-gordon-moore-law.html`

[Samsung] Samsung Electronics Co., Ltd.; Suwon, Südkorea.
`www.samsung.com`

[TI] Texas Instruments Inc.; Dallas, TX.
`www.texasinstruments.com`