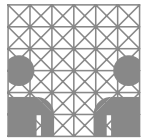


64-040 Modul InfB-RS: Rechnerstrukturen

https://tams.informatik.uni-hamburg.de/lectures/2016ws/vorlesung/rs

– Kapitel 2 –

Andreas Mäder

Universität Hamburg
Fakultät für Mathematik, Informatik und Naturwissenschaften
Fachbereich Informatik

Technische Aspekte Multimodaler Systeme

Wintersemester 2016/2017

Kapitel 2

Digitalrechner

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:

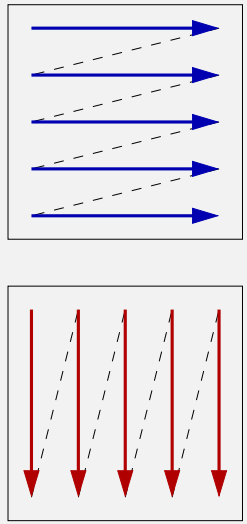
- Gesamtverständnis erfordert Kenntnisse auf allen Ebenen
- häufig Rückwirkung von unteren auf obere Ebenen

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           (!)
  }
}
```

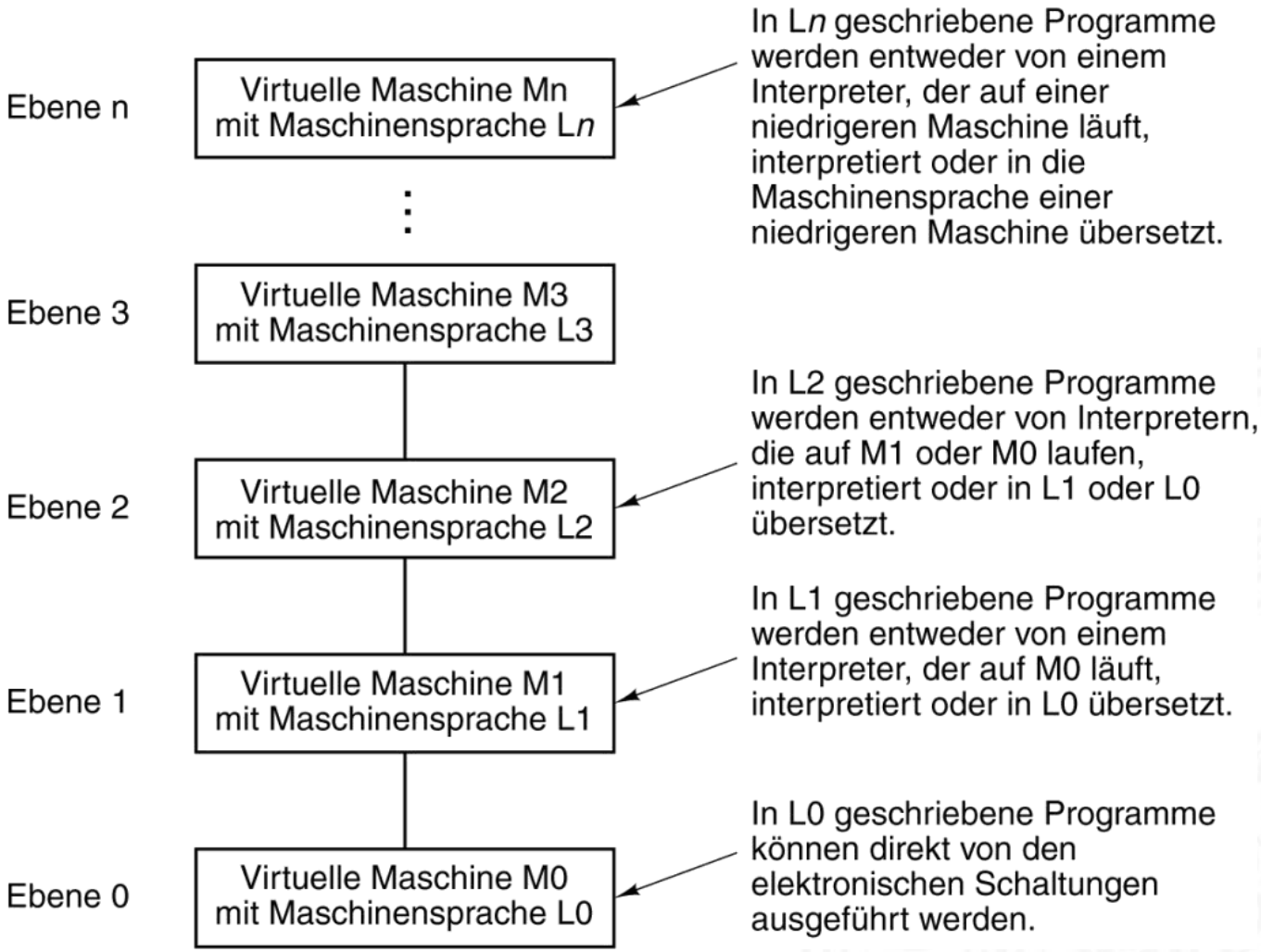
Rückwirkung von unteren Ebenen: Performance


```
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 ms	
Matrix row-col summation	75 ms	
Matrix col-row summation	383 ms	⇒ 5 × 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:

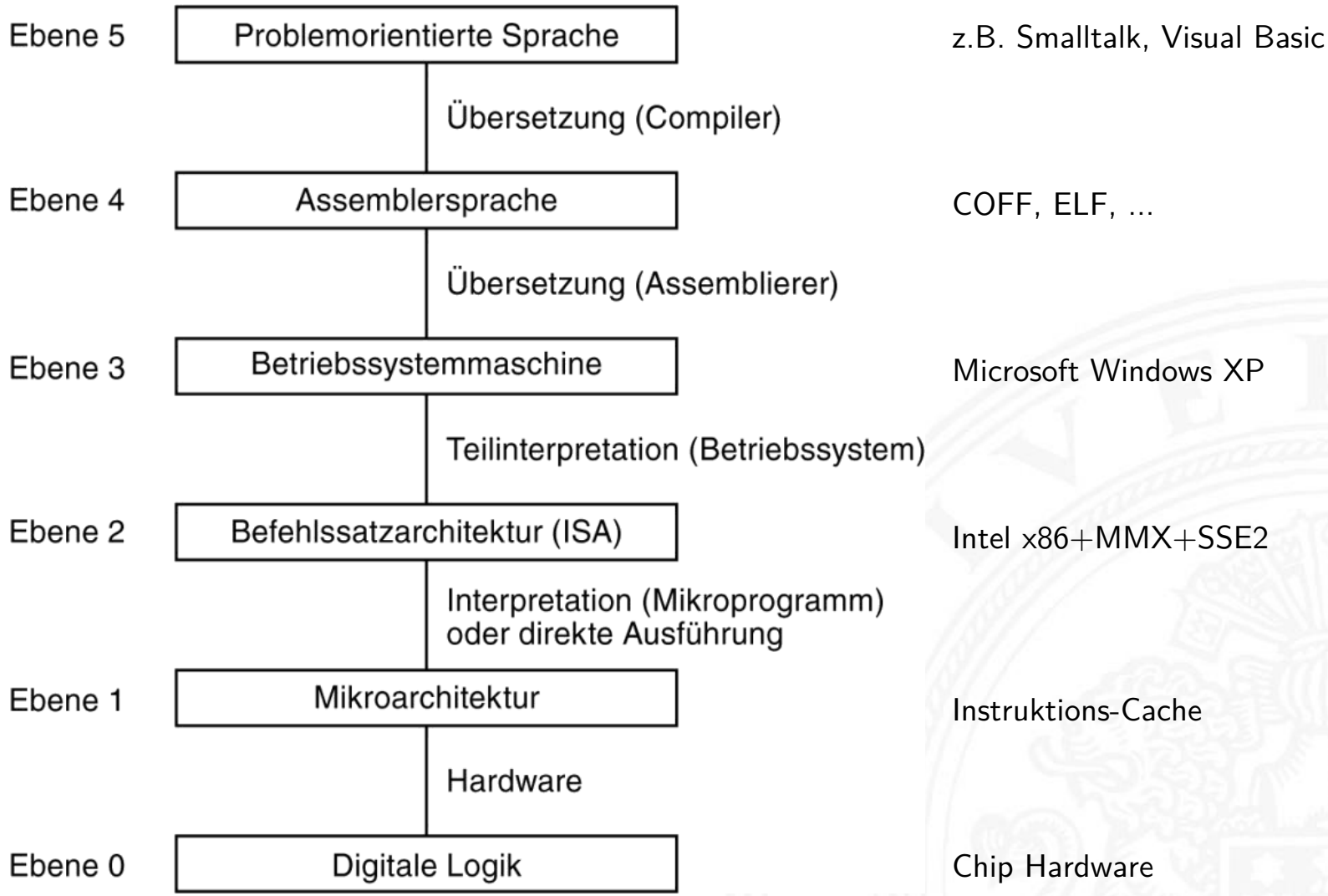
Virtuellen Maschine M1 für die Sprache L1

- 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

```c
/* 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 Kopieren, ...

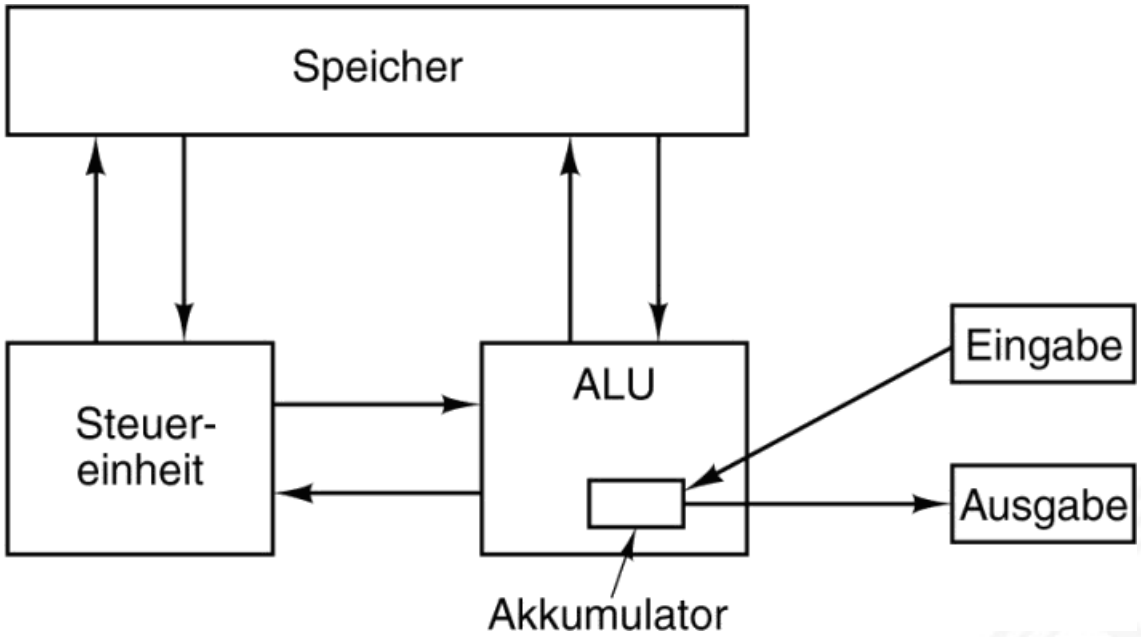
von-Neumann Konzept

- 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



[TA14]

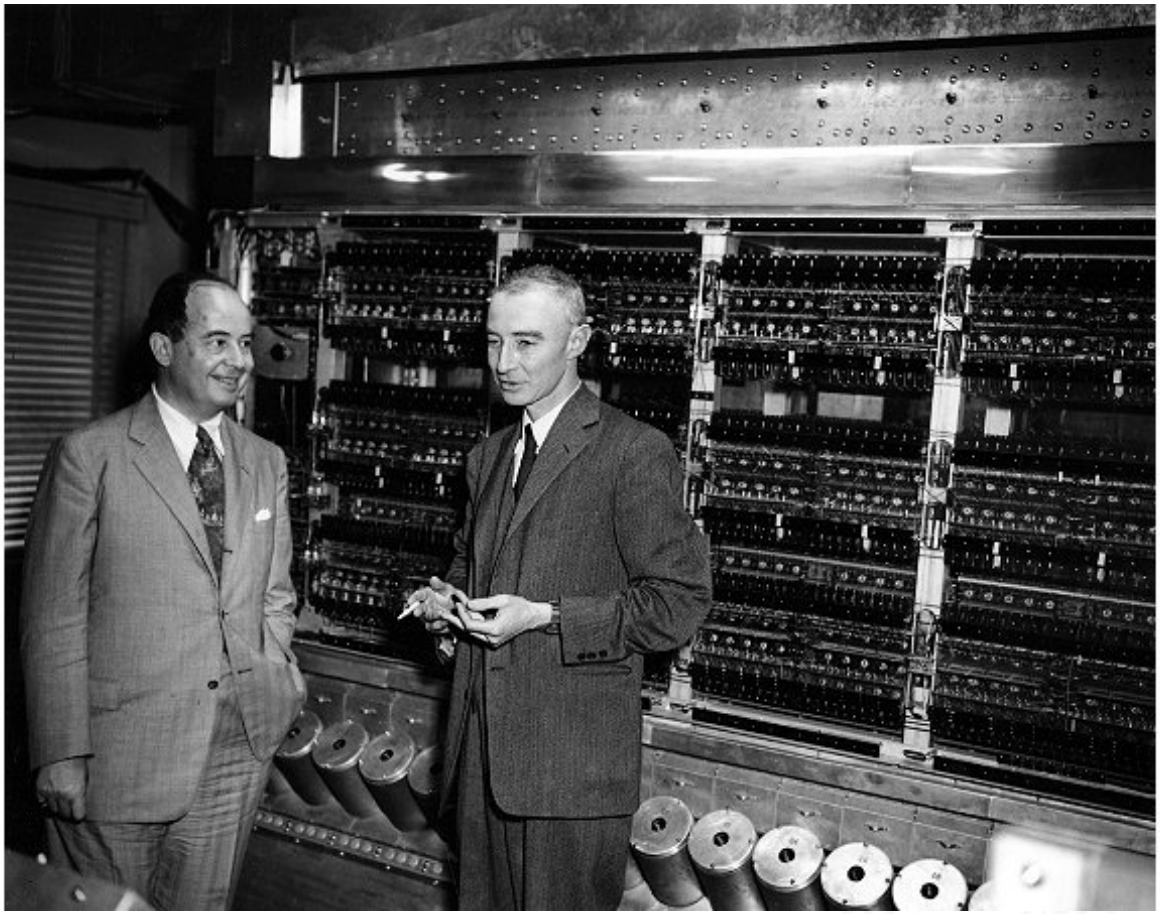
Fünf zentrale Komponenten:

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

von-Neumann Rechner (cont.)

- Prozessor (CPU) = Steuerwerk + Operationswerk
- Steuerwerk: zwei zentrale Register
  - Befehlszähler (*program counter PC*)
  - Befehlsregister (*instruction register IR*)
- 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



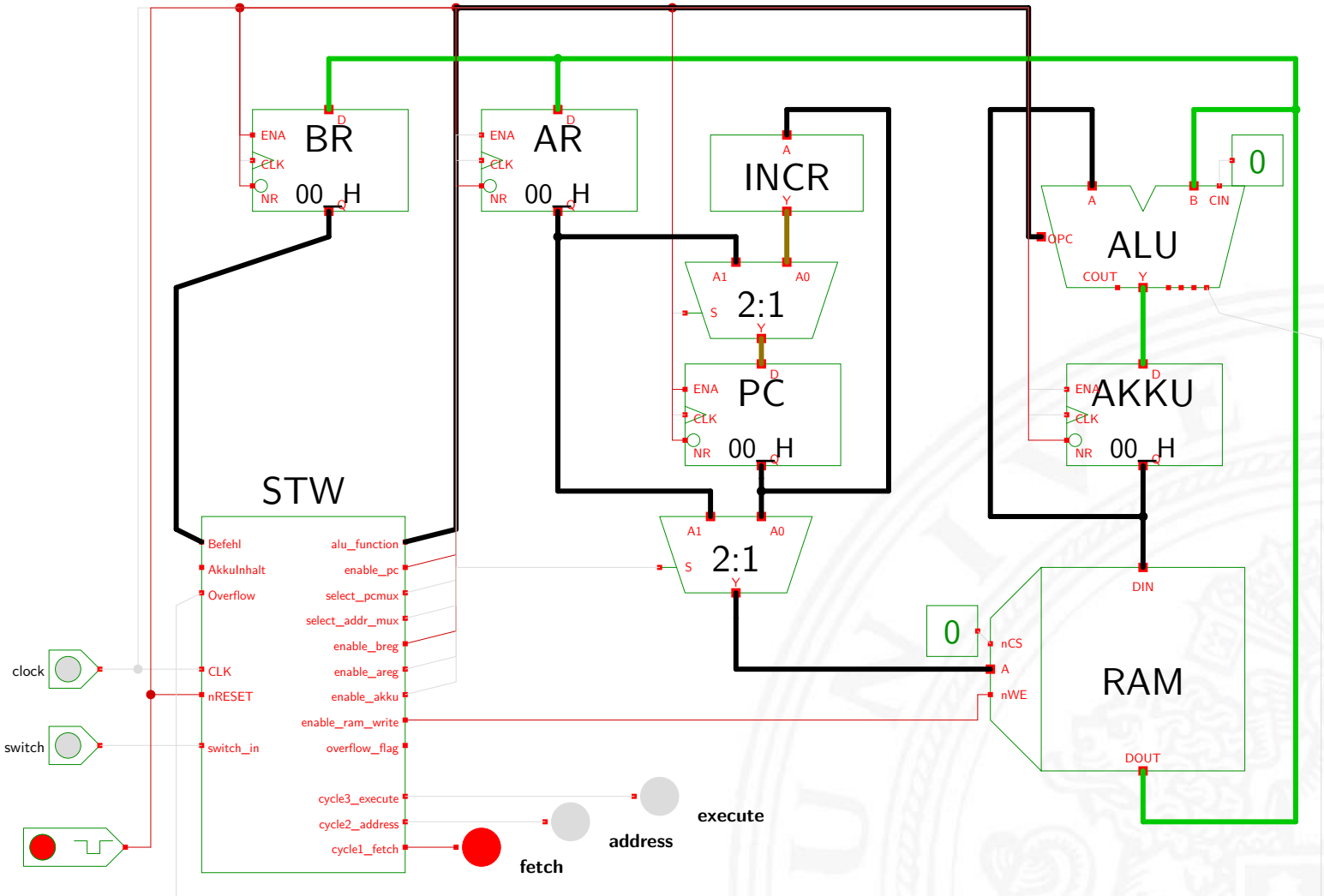
John von Neumann, R. J. Oppenheimer, IAS Computer Princeton `www.computerhistory.org`

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: `https://tams.informatik.uni-hamburg.de/applets/hades/webdemos/50-rtlib/90-prima/chapter.html`

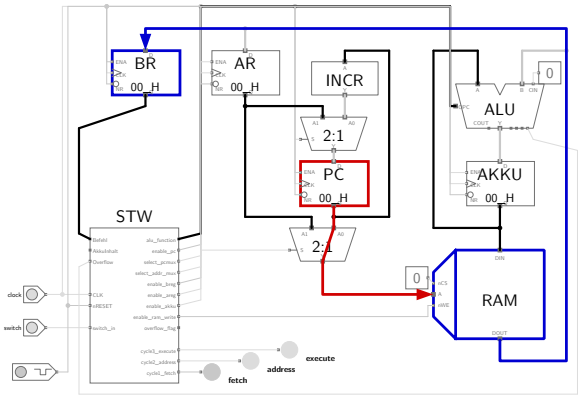
PRIMA: die Primitive Maschine



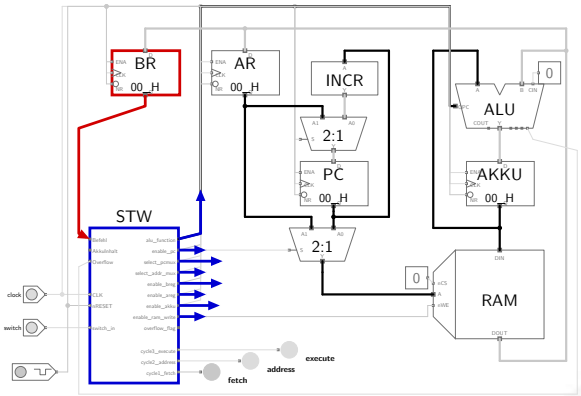
[HenHA] Hades Demo: `50-rtlib/90-prima/prima`

PRIMA: die Zyklen

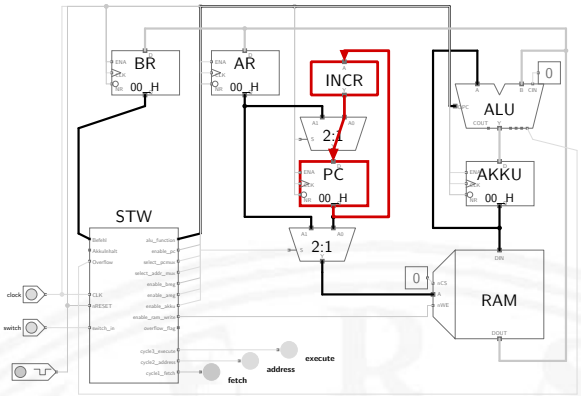
Befehl holen

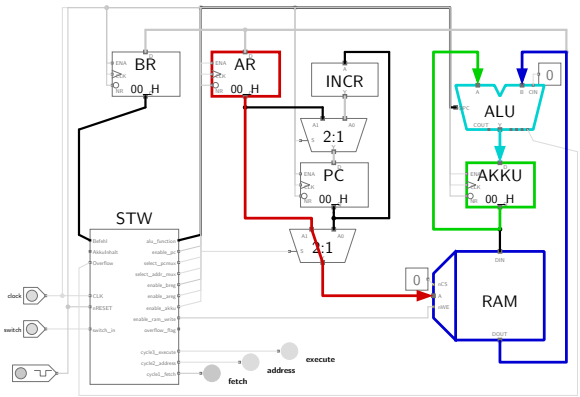


decodieren

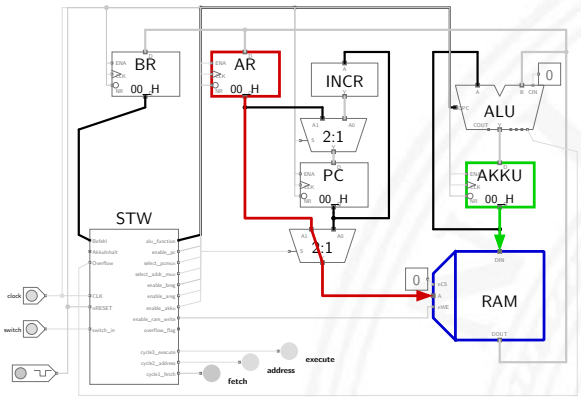


PC inkrementieren

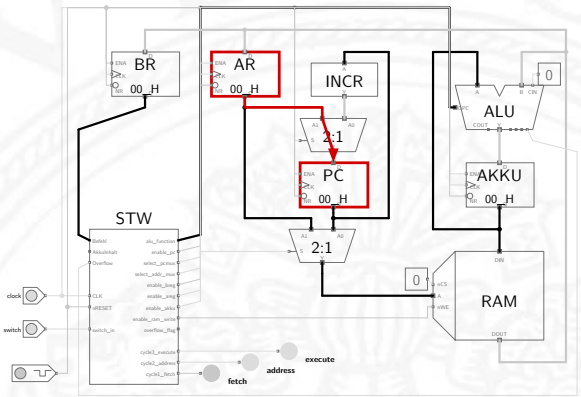




rechnen



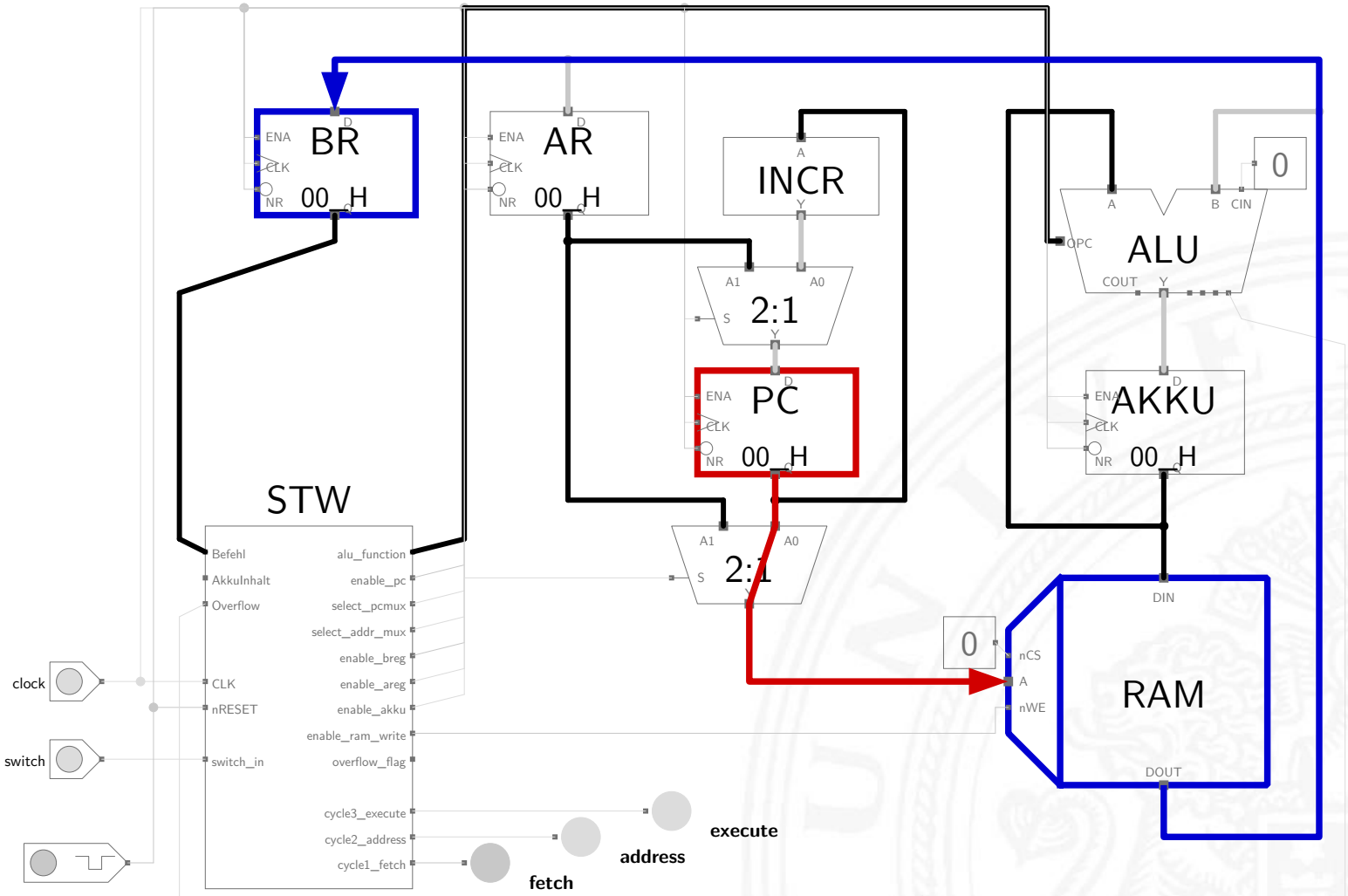
speichern



springen

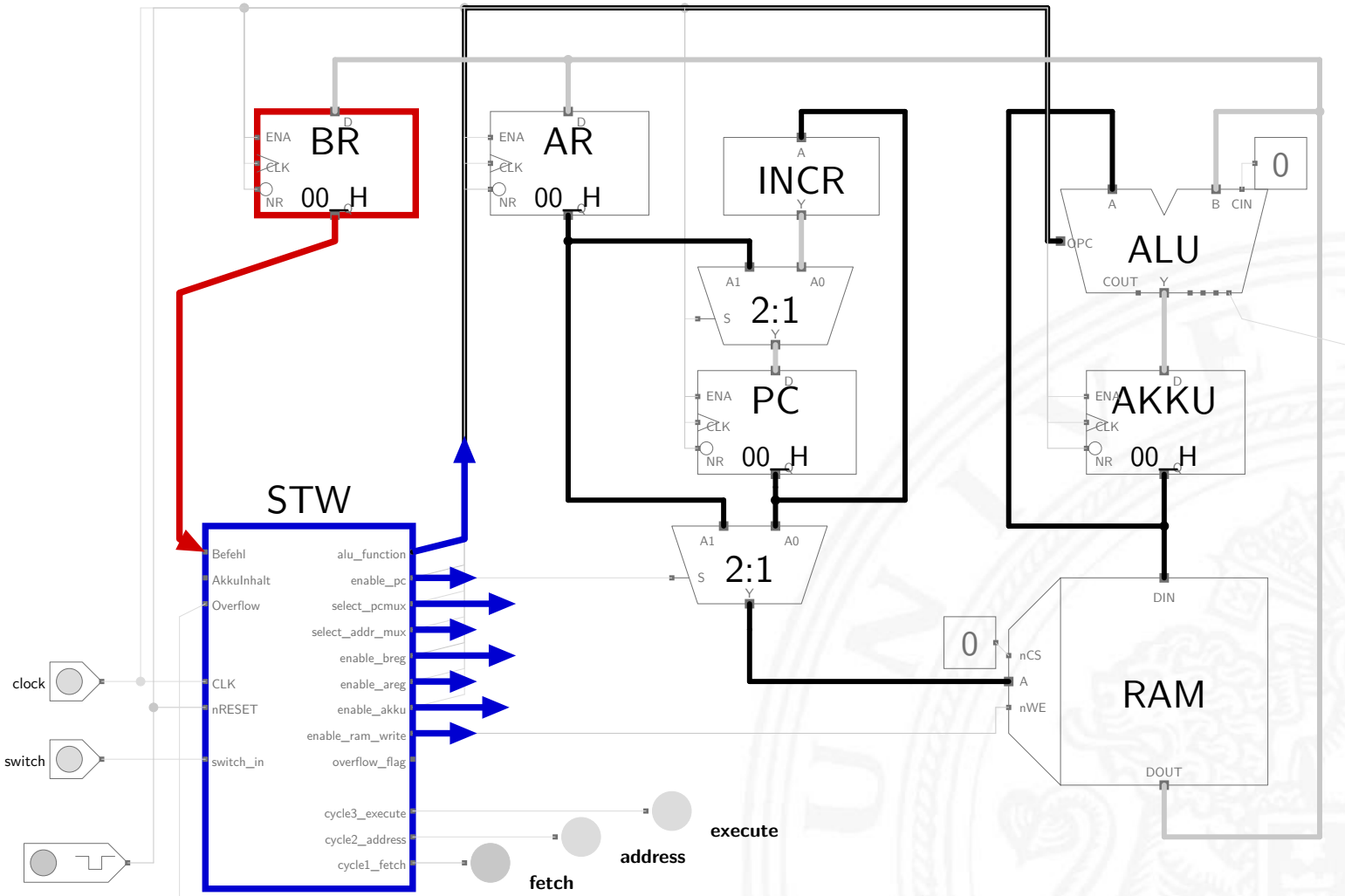
PRIMA: Befehl holen

BR = RAM[PC]



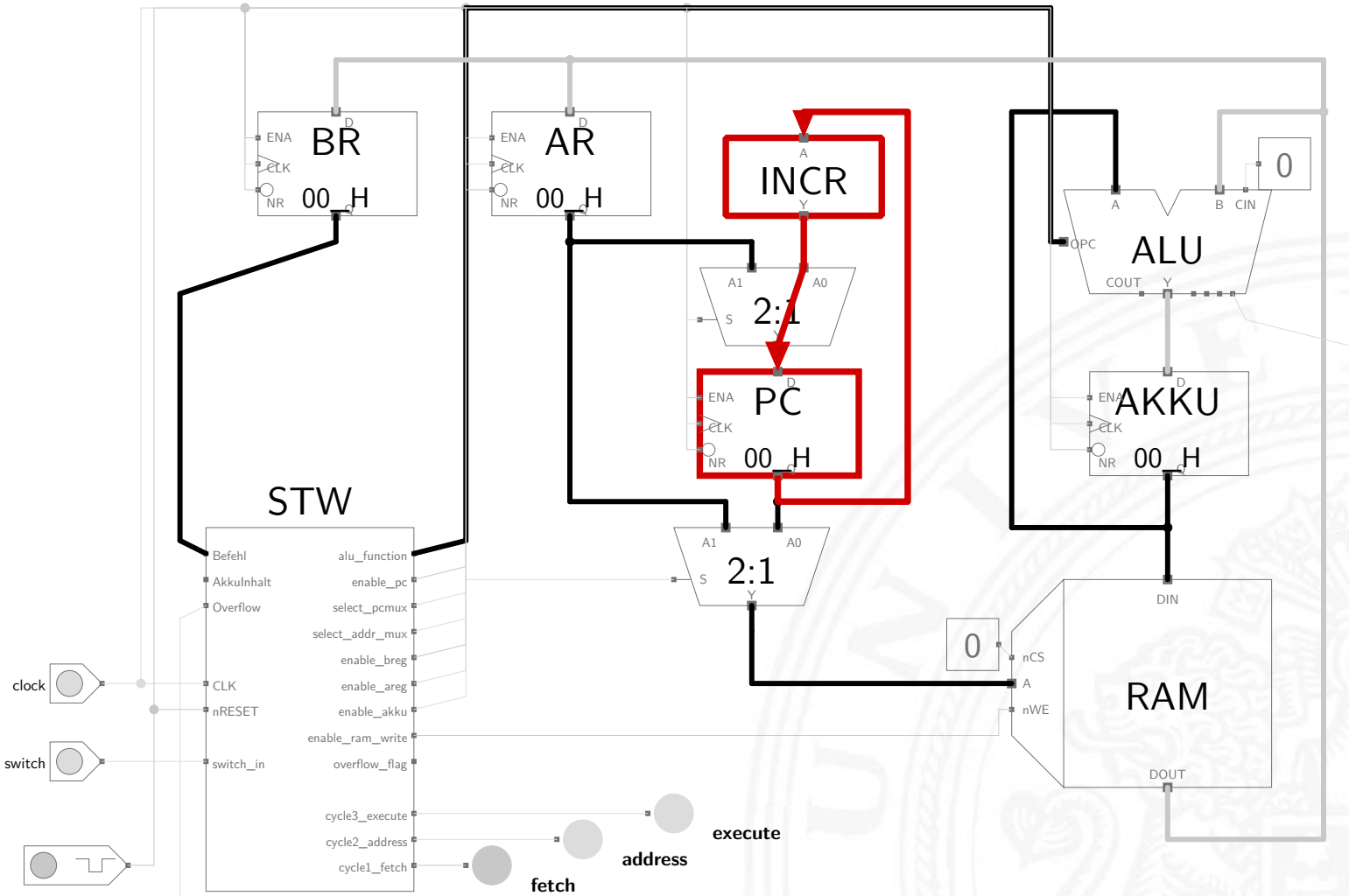
PRIMA: decodieren

Steuersignale = decode(BR)



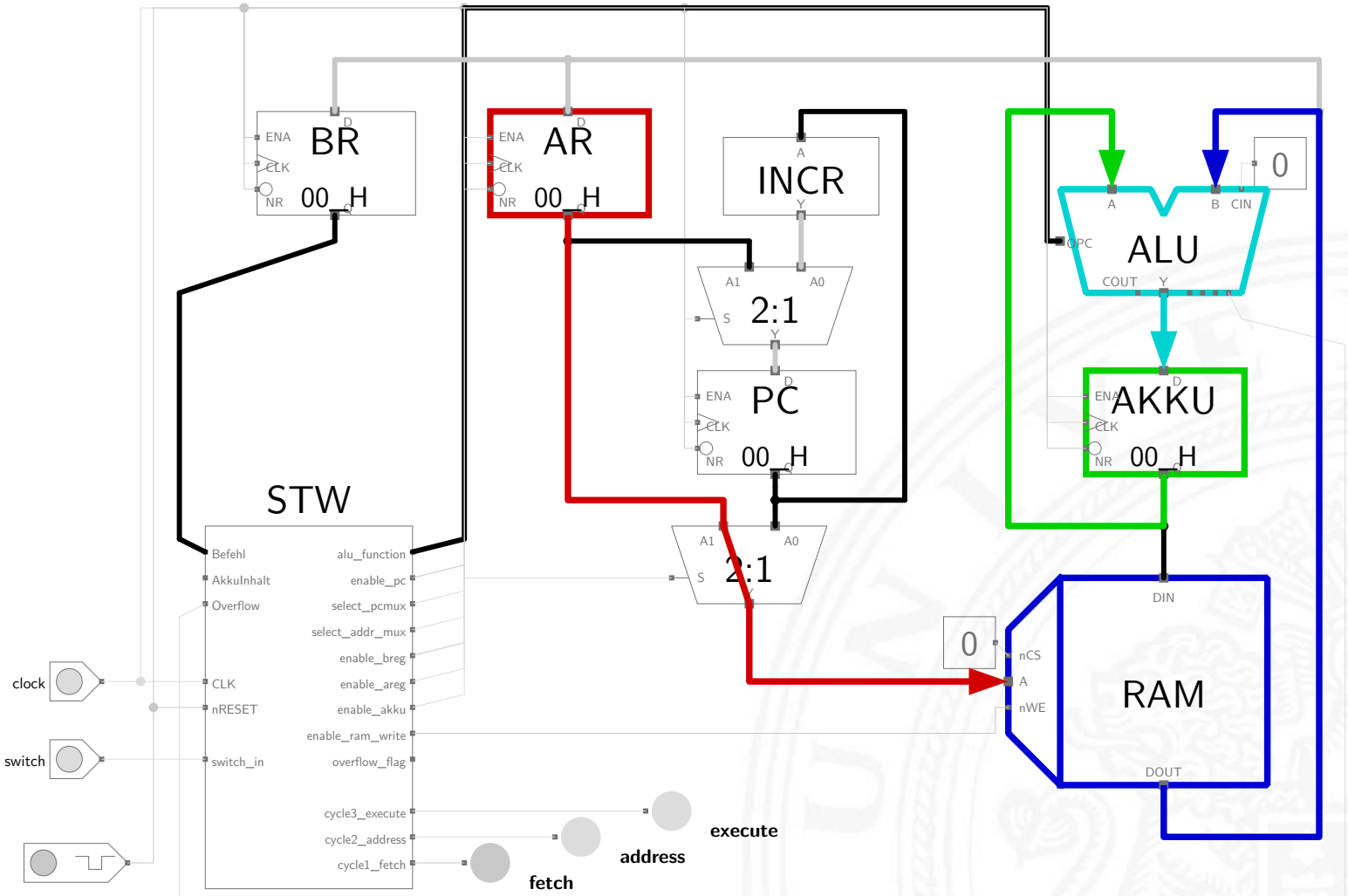
PRIMA: PC inkrementieren

PC = PC+1

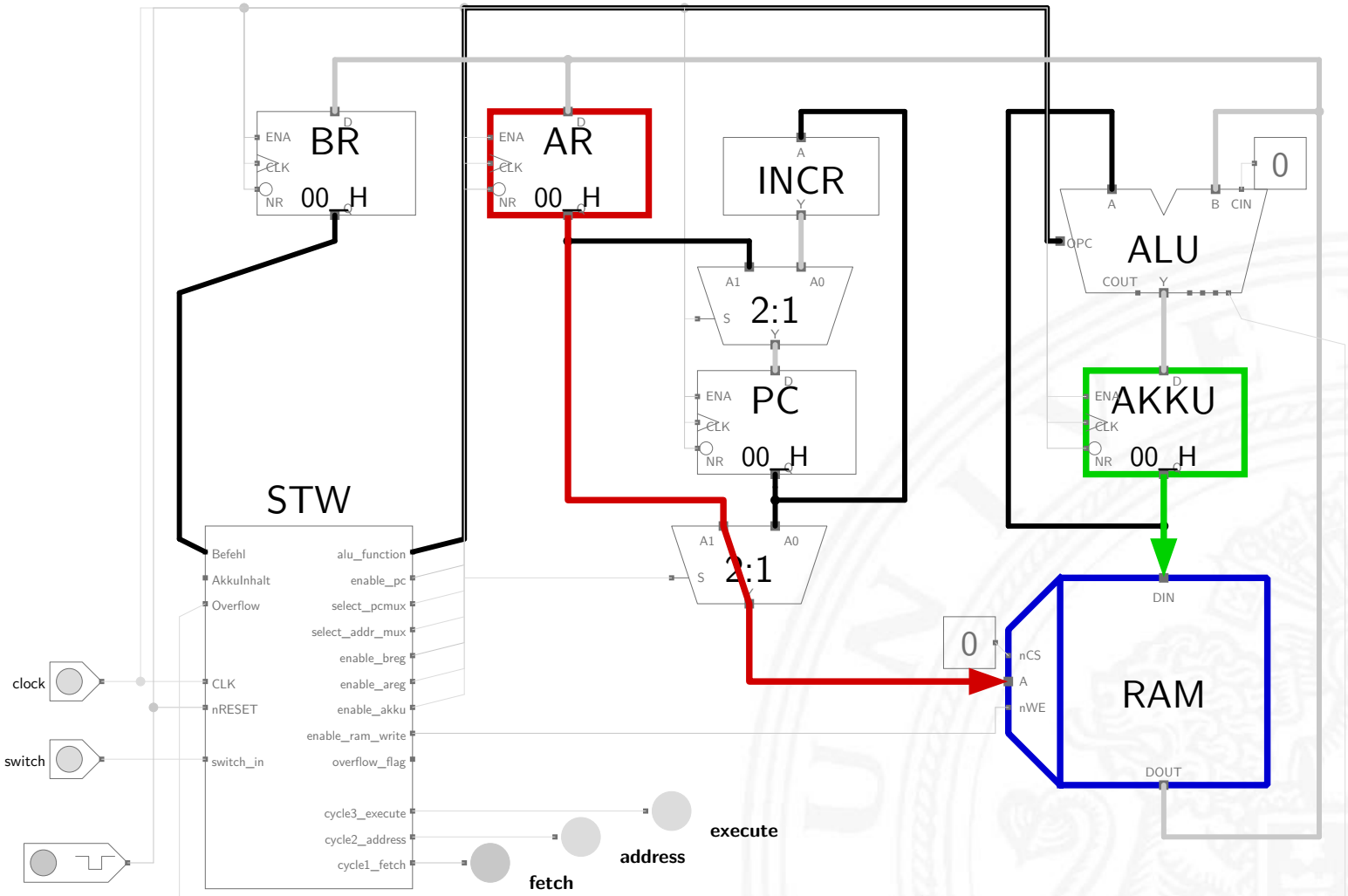


PRIMA: rechnen

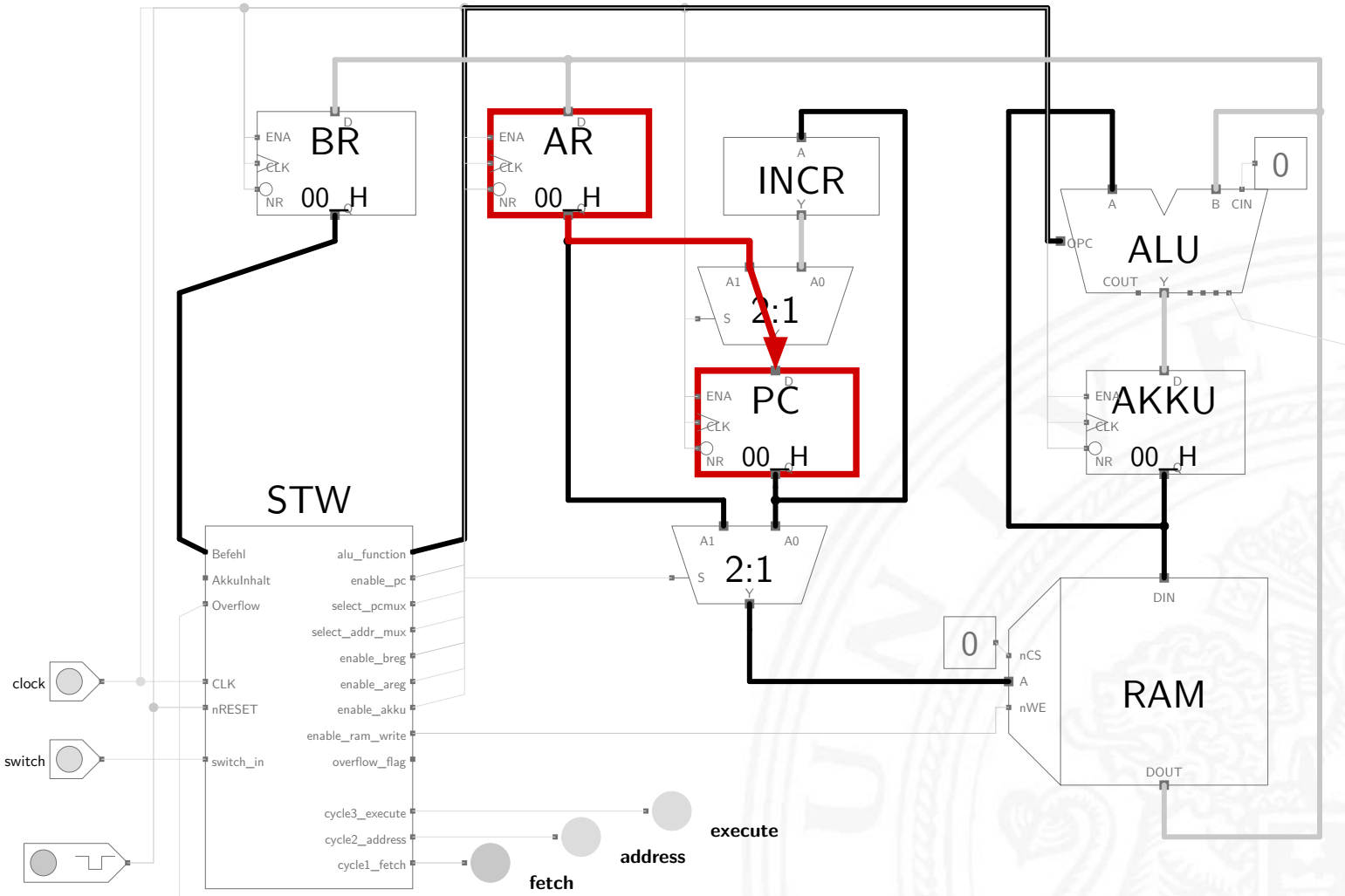
Akku = Akku + RAM[AR]



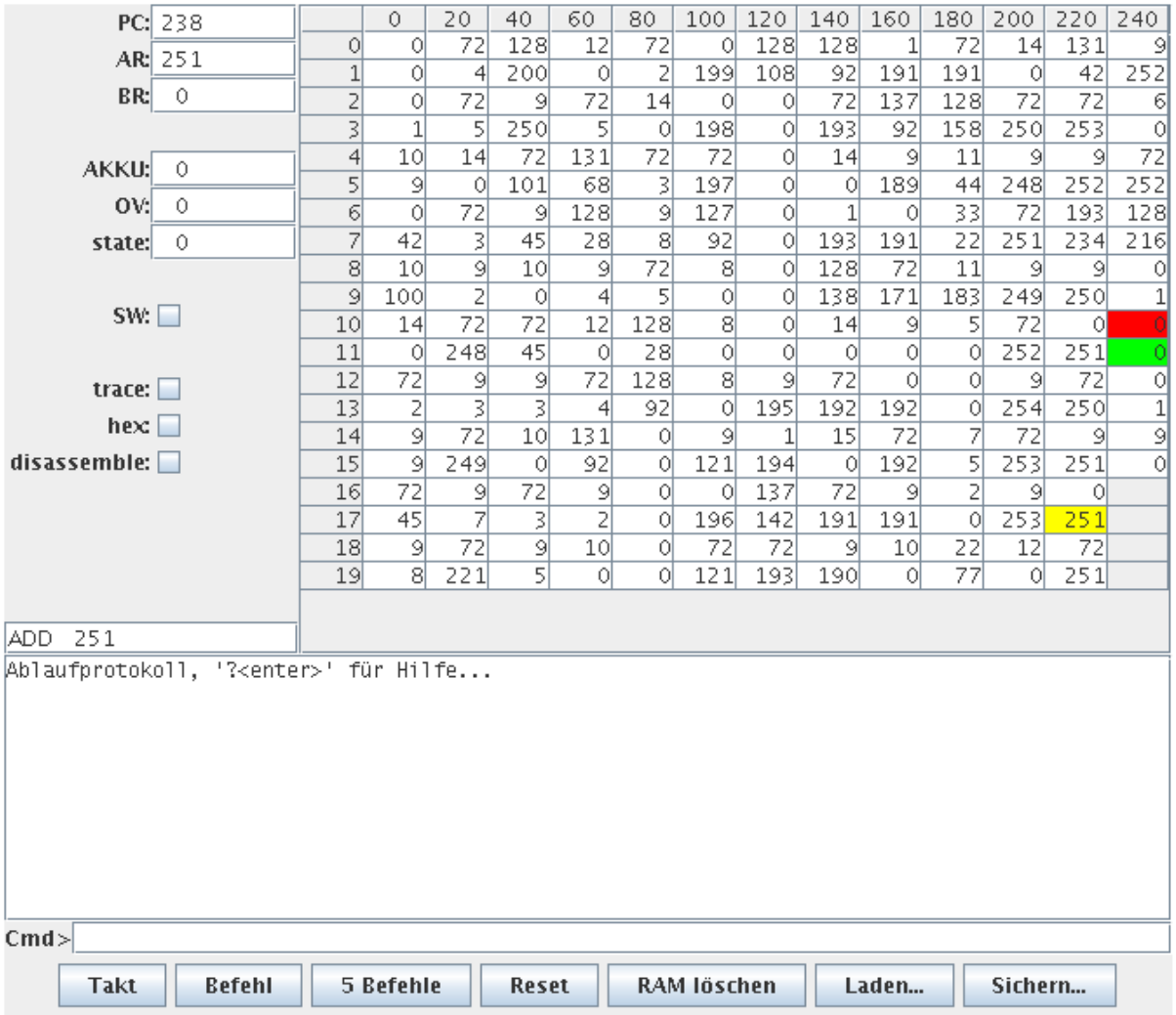
PRIMA: speichern
RAM[AR] = Akku



PRIMA: springen
PC = AR



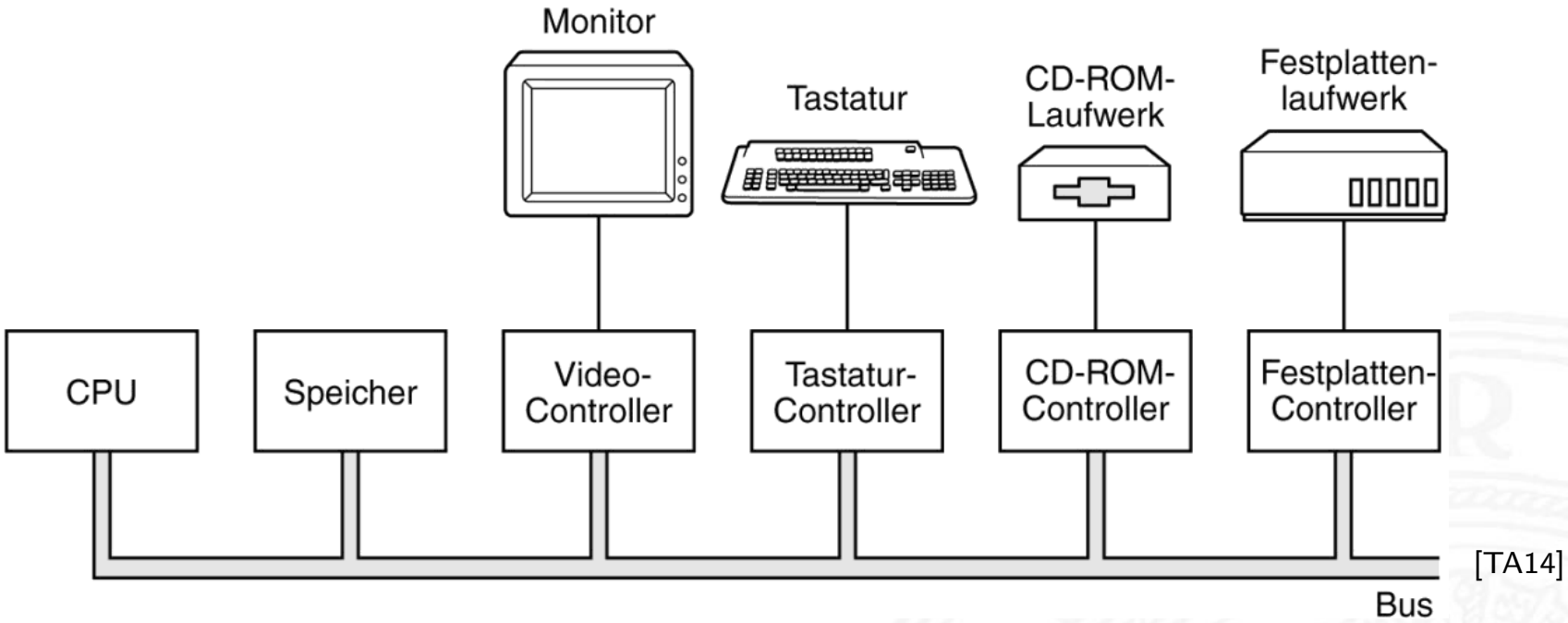
PRIMA: Simulator



	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	

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

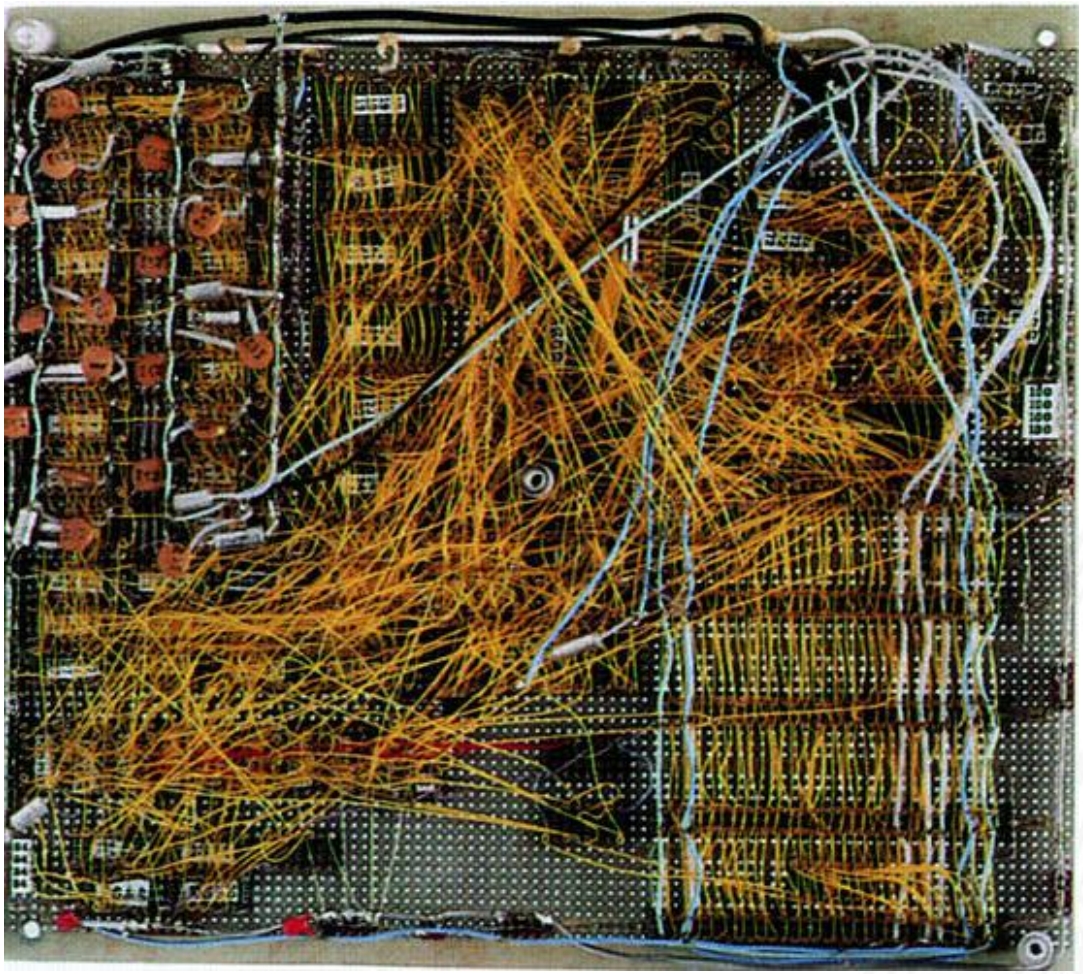
Personal Computer: Aufbau des IBM PC (1981)

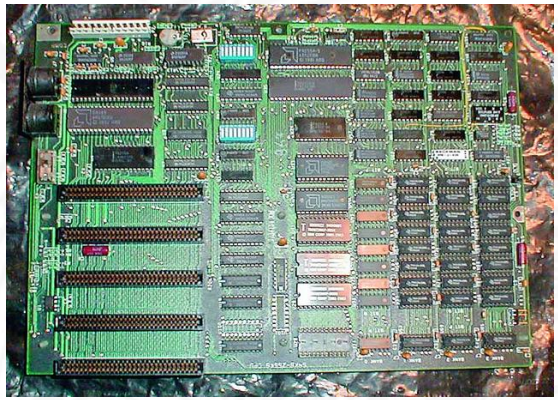


[TA14]

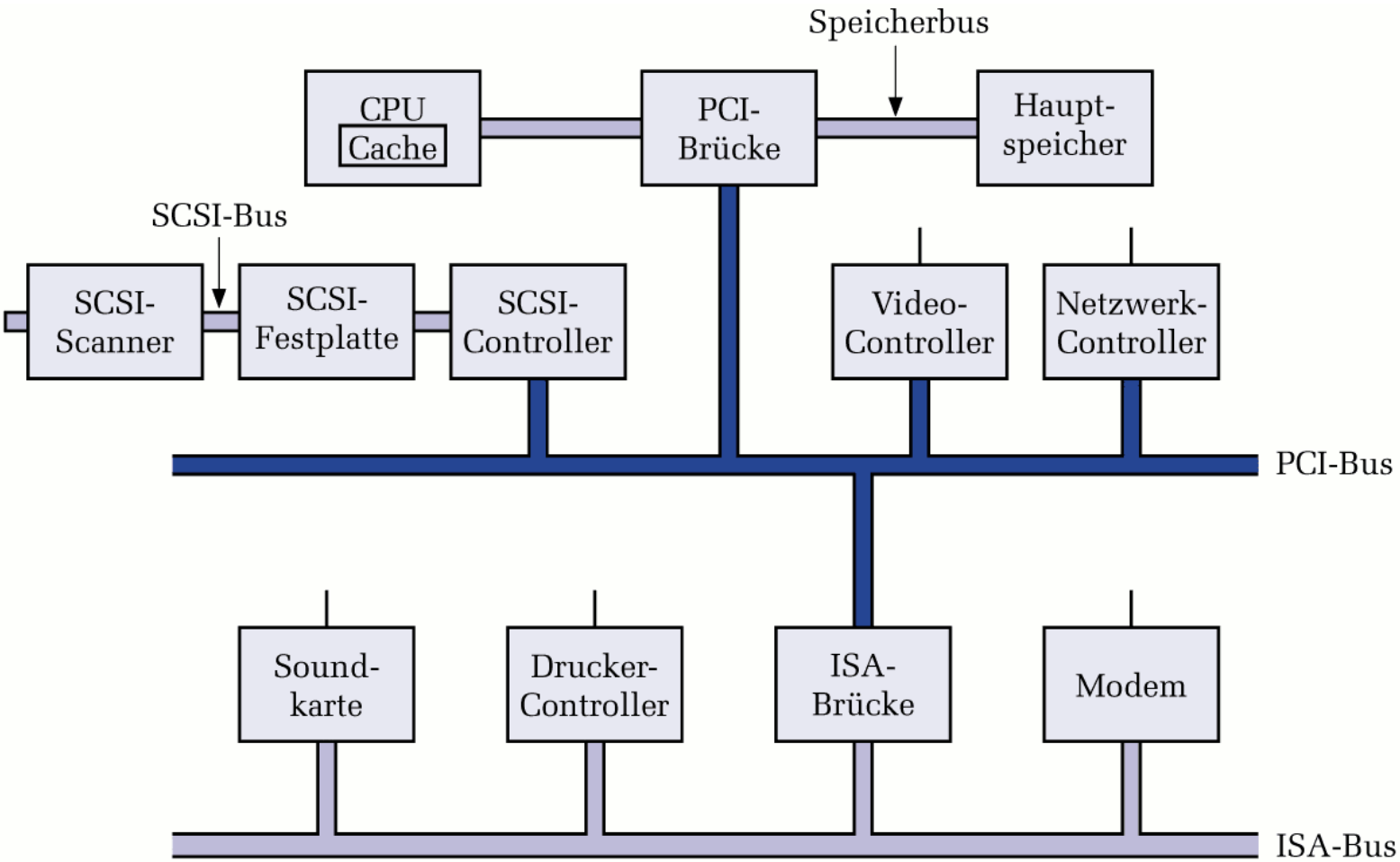
- Intel 8086/8088, 512 KByte RAM, Betriebssystem MS-DOS
- alle Komponenten über den zentralen („ISA“-) Bus verbunden
- Erweiterung über Einsteckkarten

Personal Computer: Prototyp (1981) und Hauptplatine



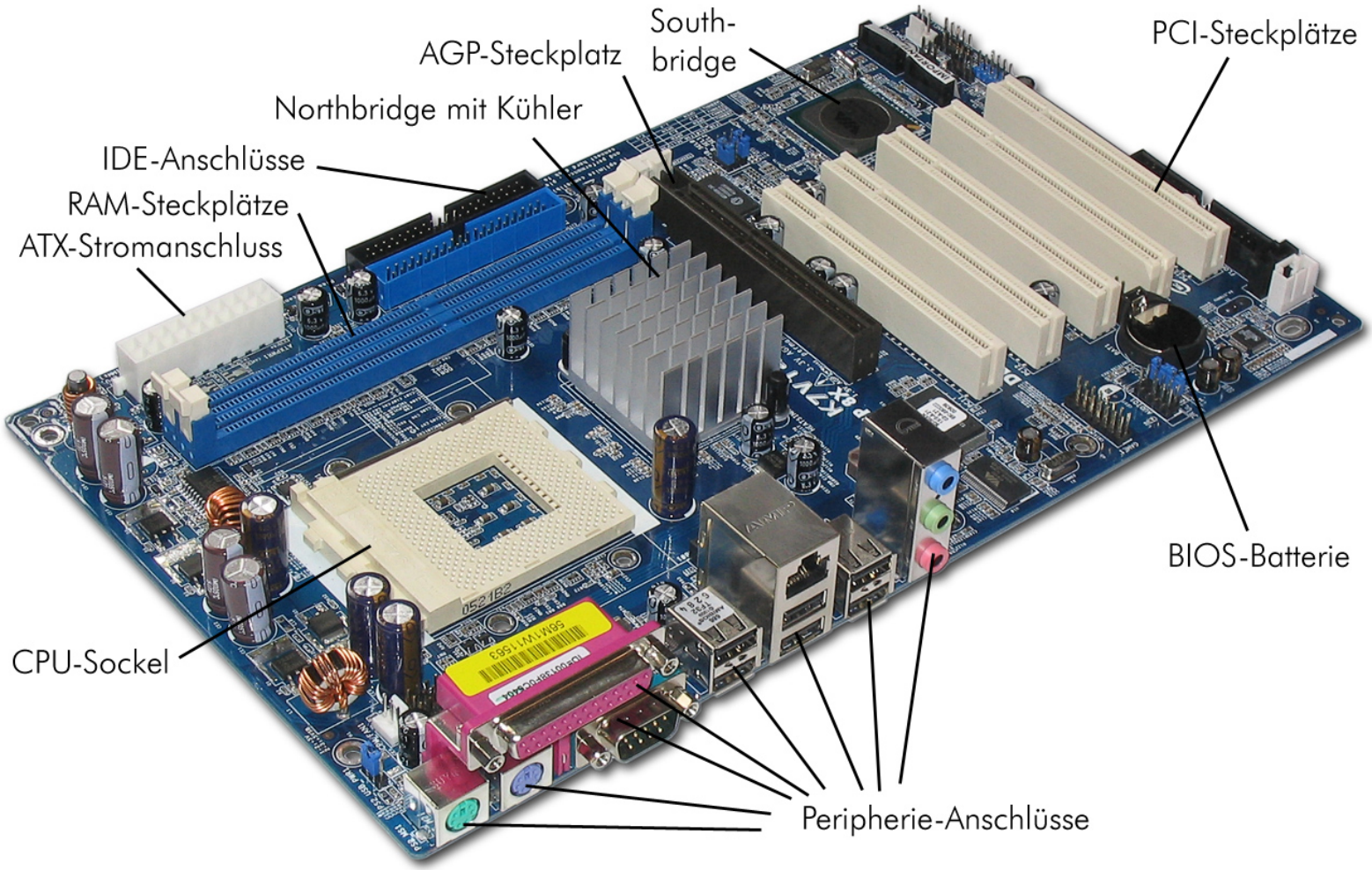


Personal Computer: Aufbau mit PCI-Bus (2000)



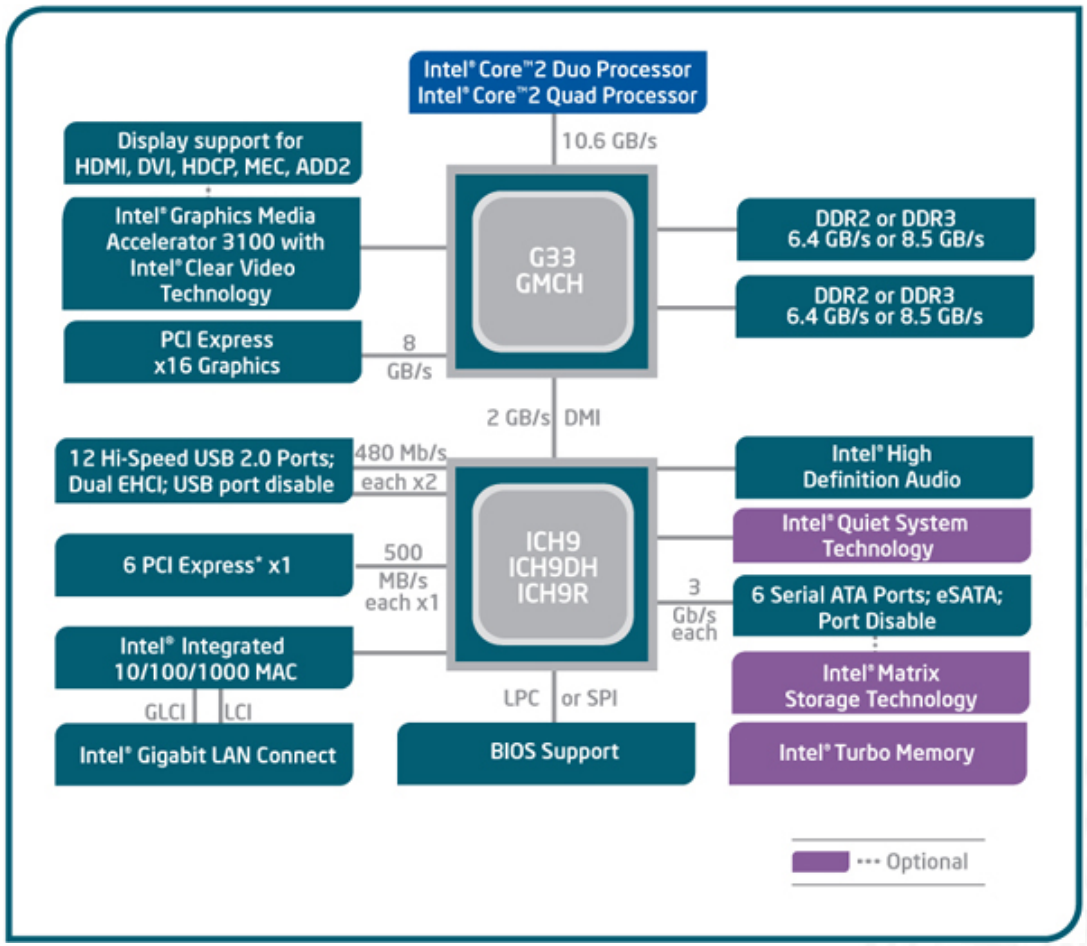
[TA14]

Personal Computer: Hauptplatine (2005)



de.wikibooks.org/wiki/Computerhardware_für_Anfänger

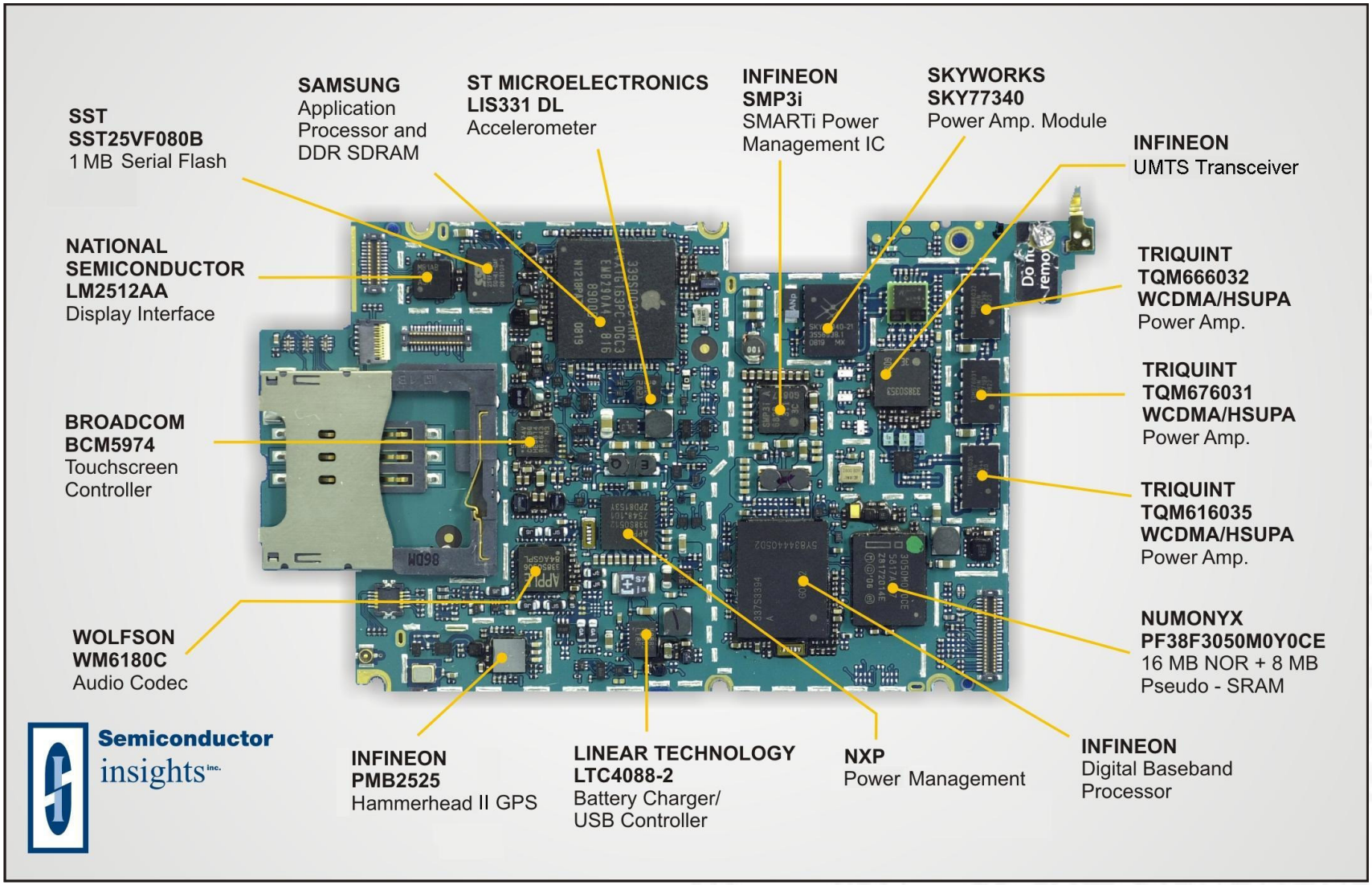
Personal Computer: Aufbau (2010)



Intel `ark.intel.com`

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

Mobilgeräte: Smartphone (2010)

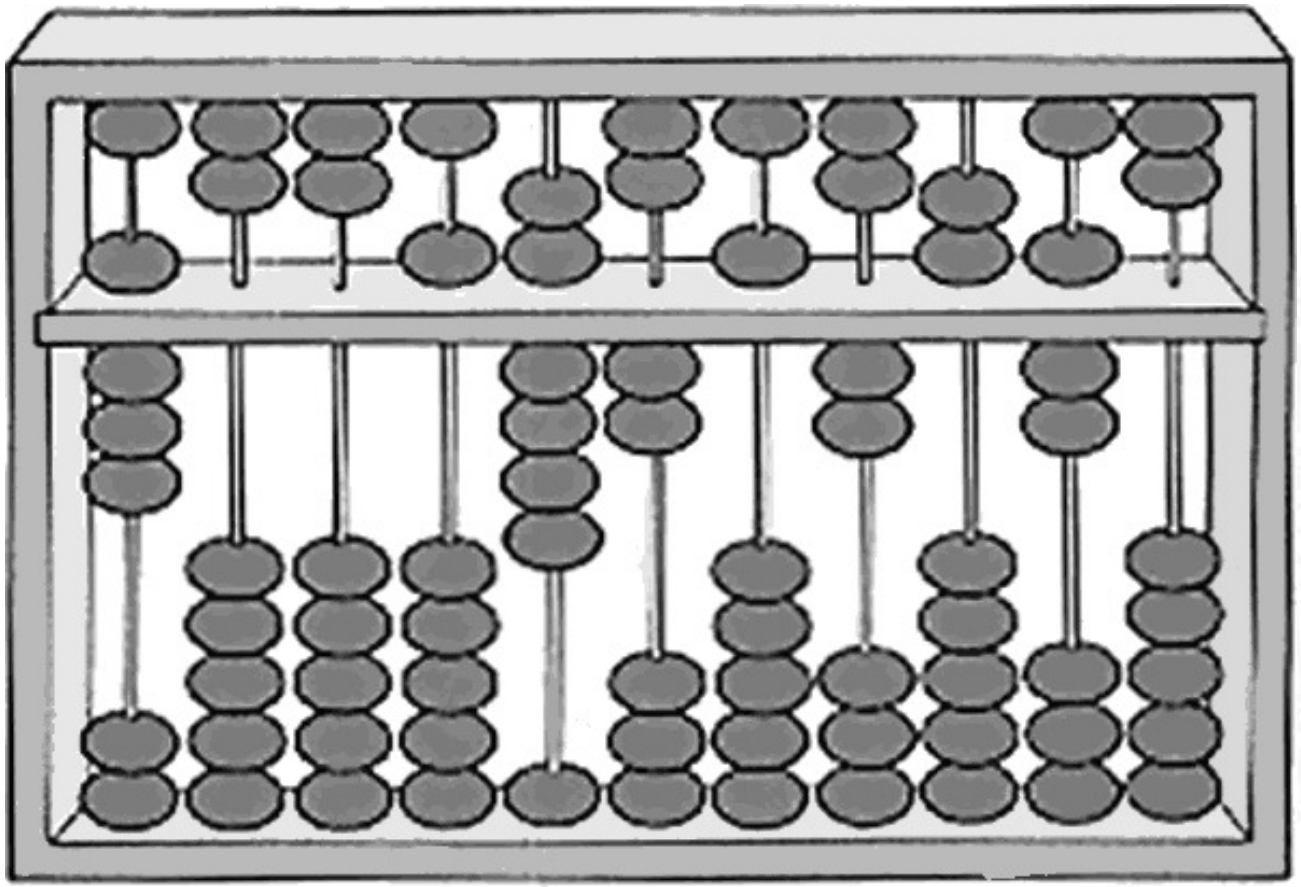


Timeline: Vorgeschichte

????	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

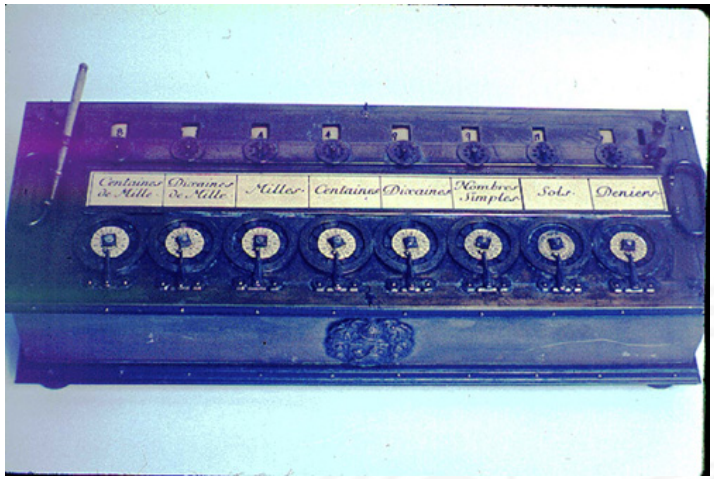


Kugel = 5

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

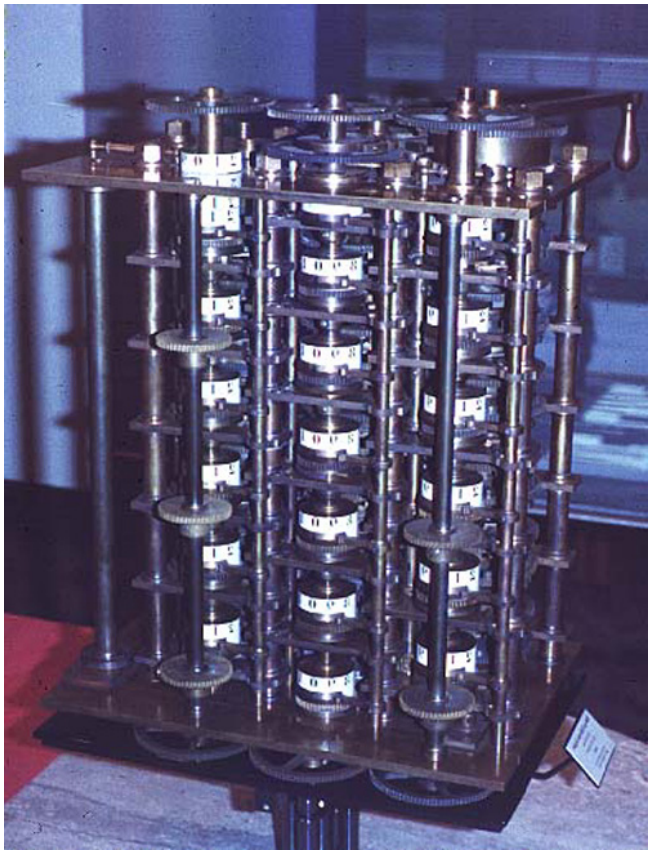
1642 Pascal: „Pascalene“

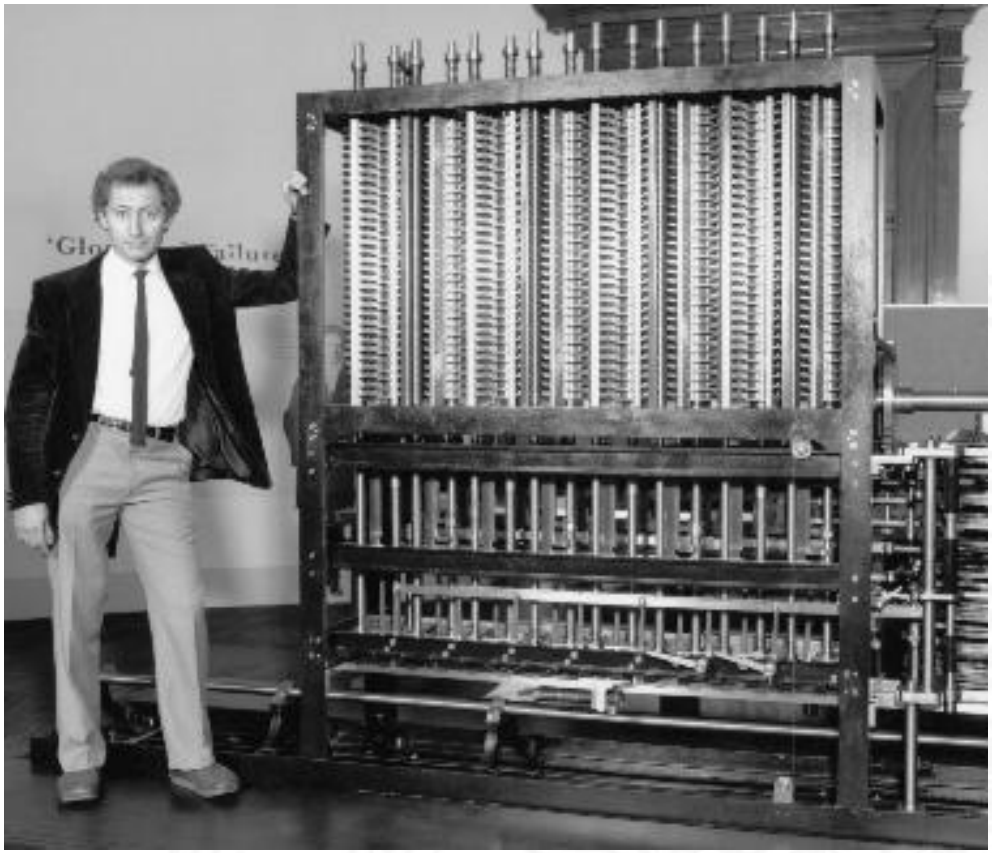
1673 Leibniz: Staffelwalze, Multiplikation/Division

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

Difference Engine

Charles Babbage 1822: Berechnung nautischer Tabellen

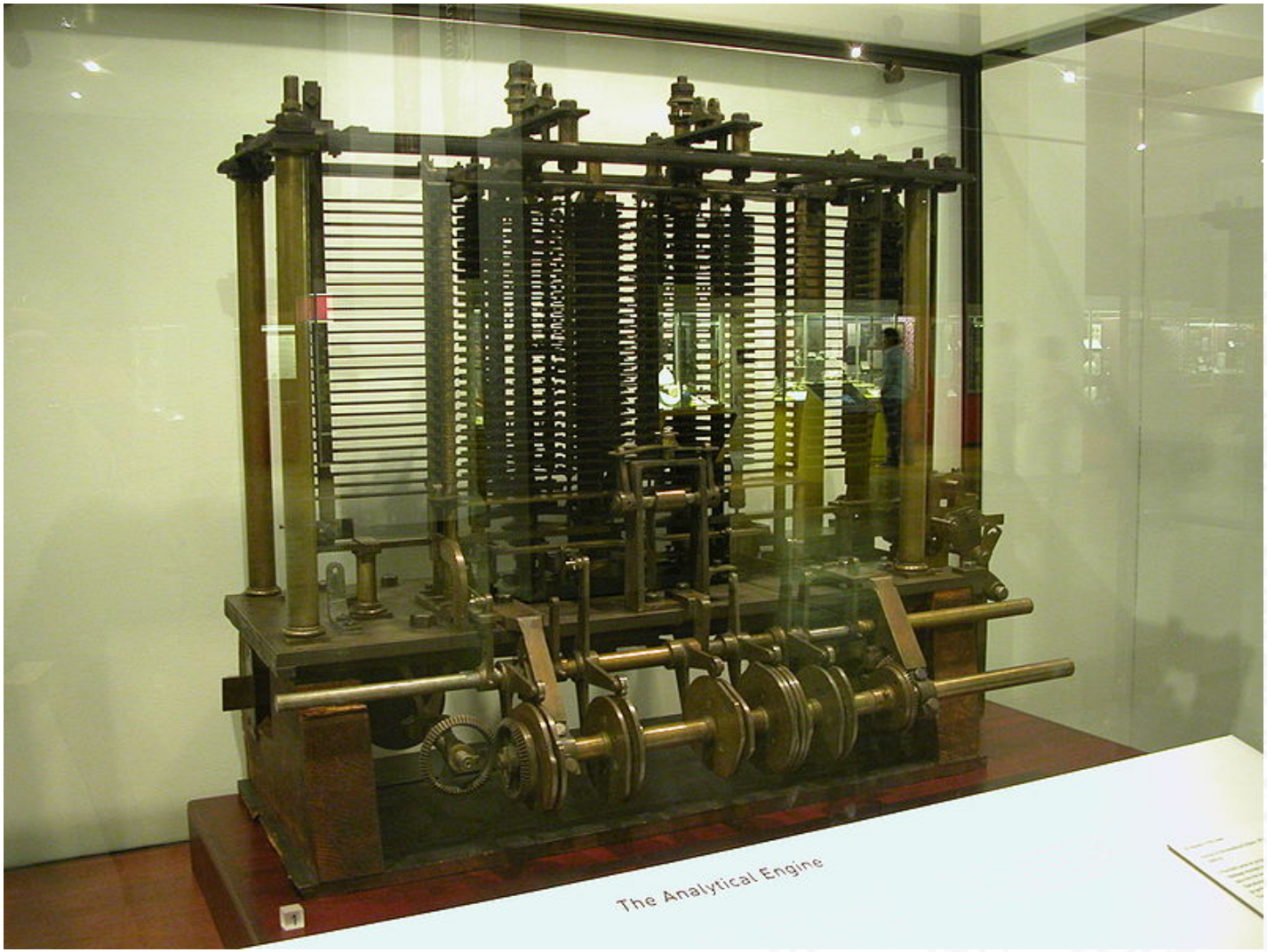




Original von 1832 und Nachbau von 1989, London Science Museum

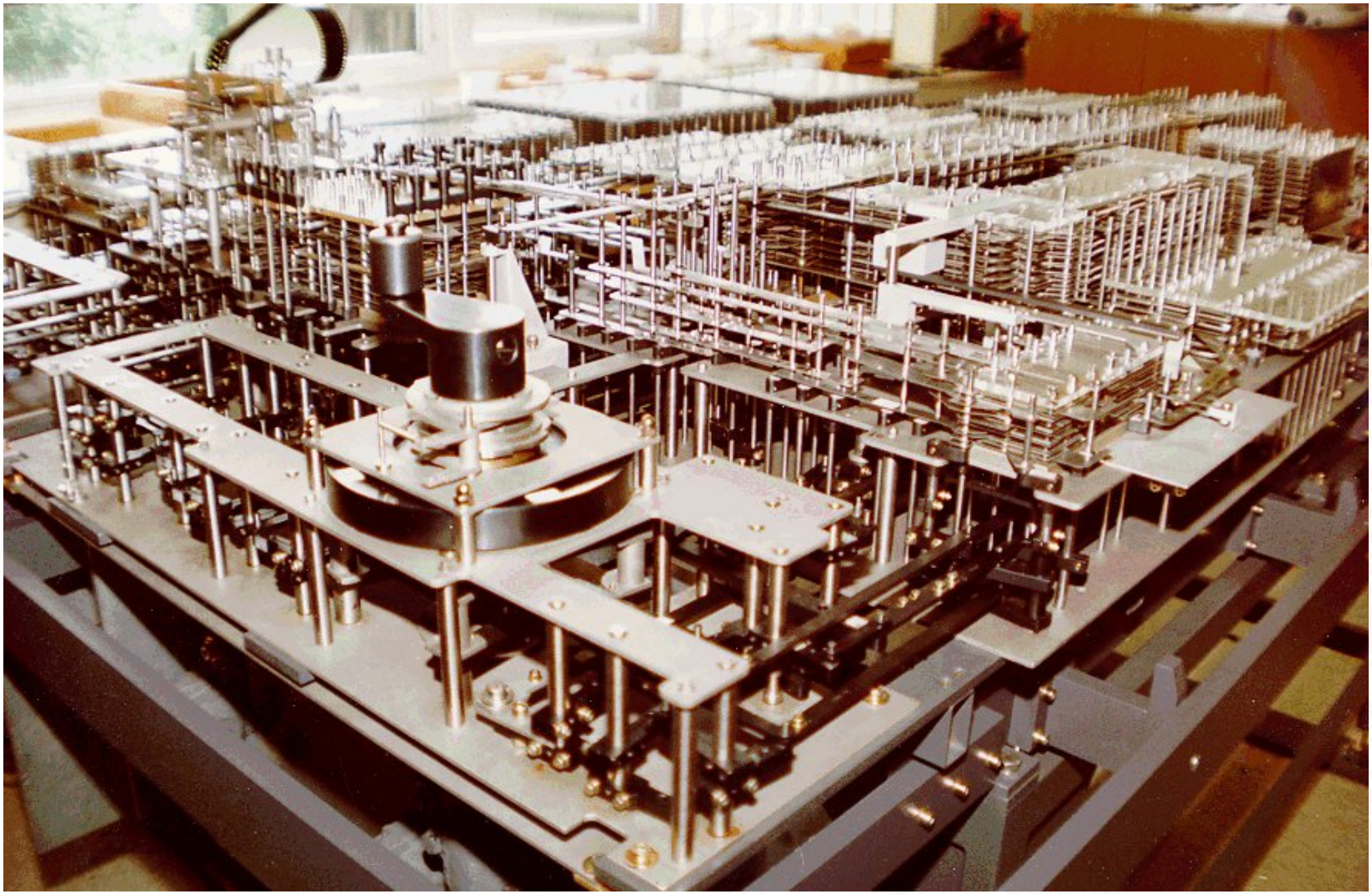
Analytical Engine

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



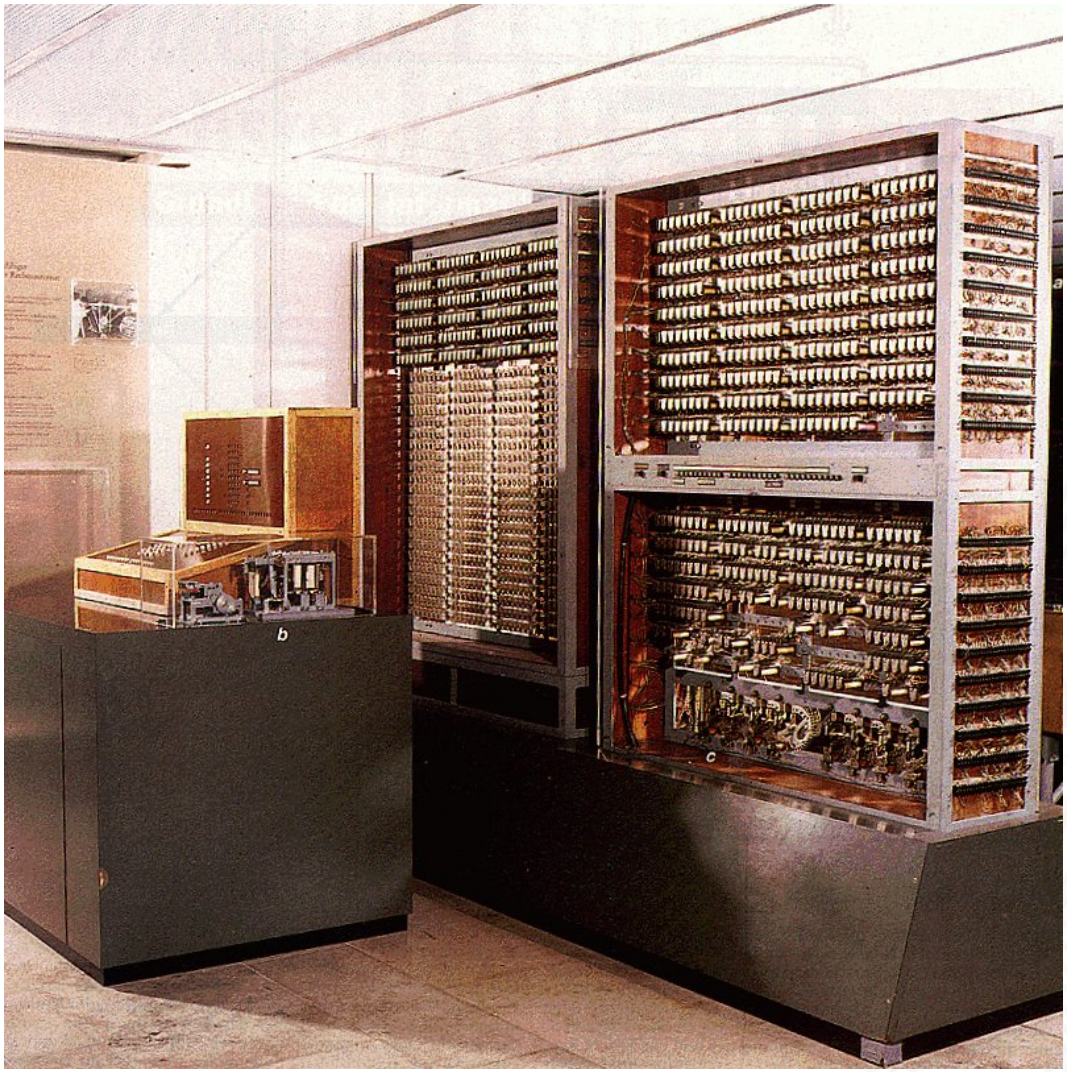
Zuse Z1

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



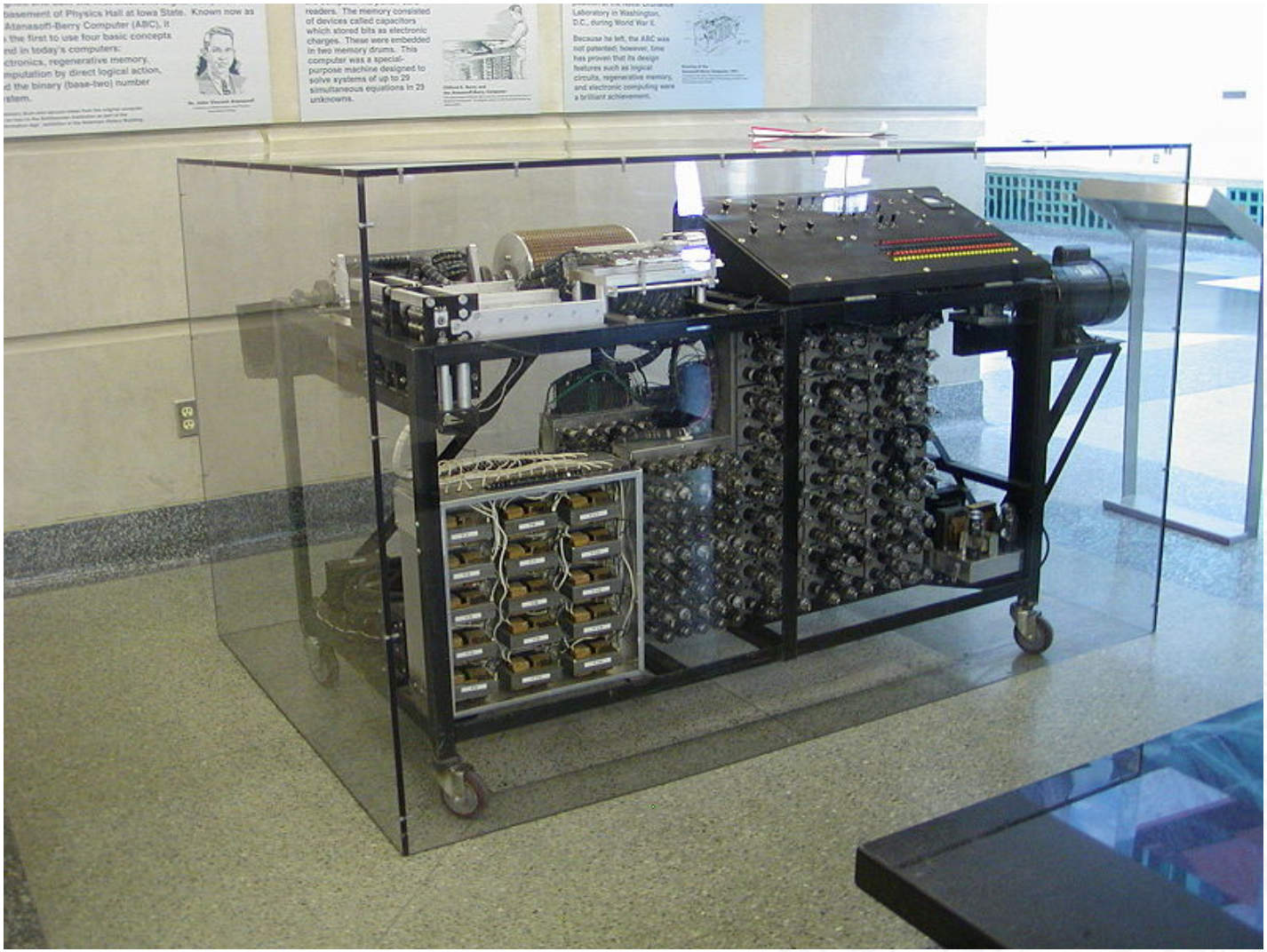
Zuse Z3

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



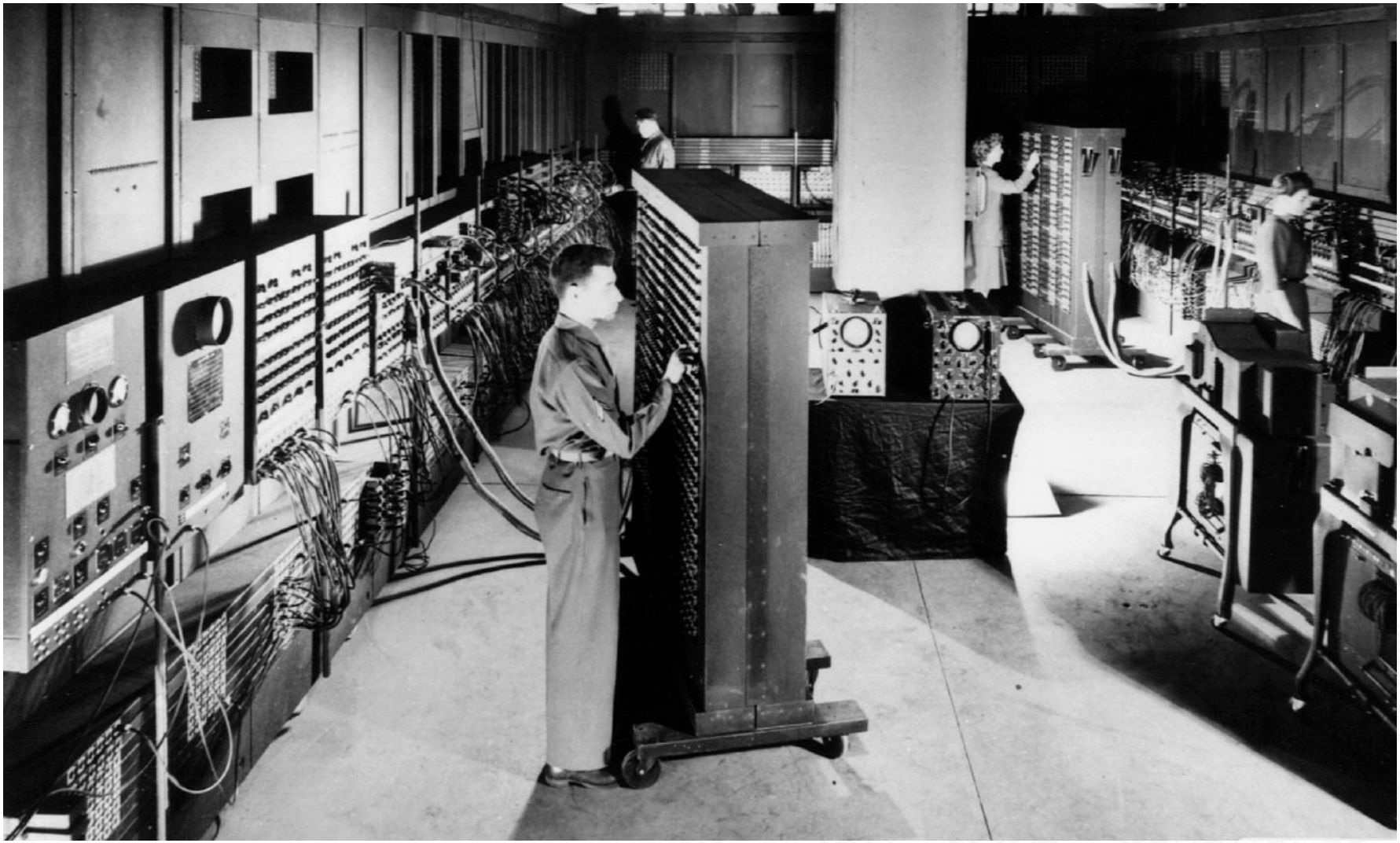
Atanasoff-Berry Computer (ABC)

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

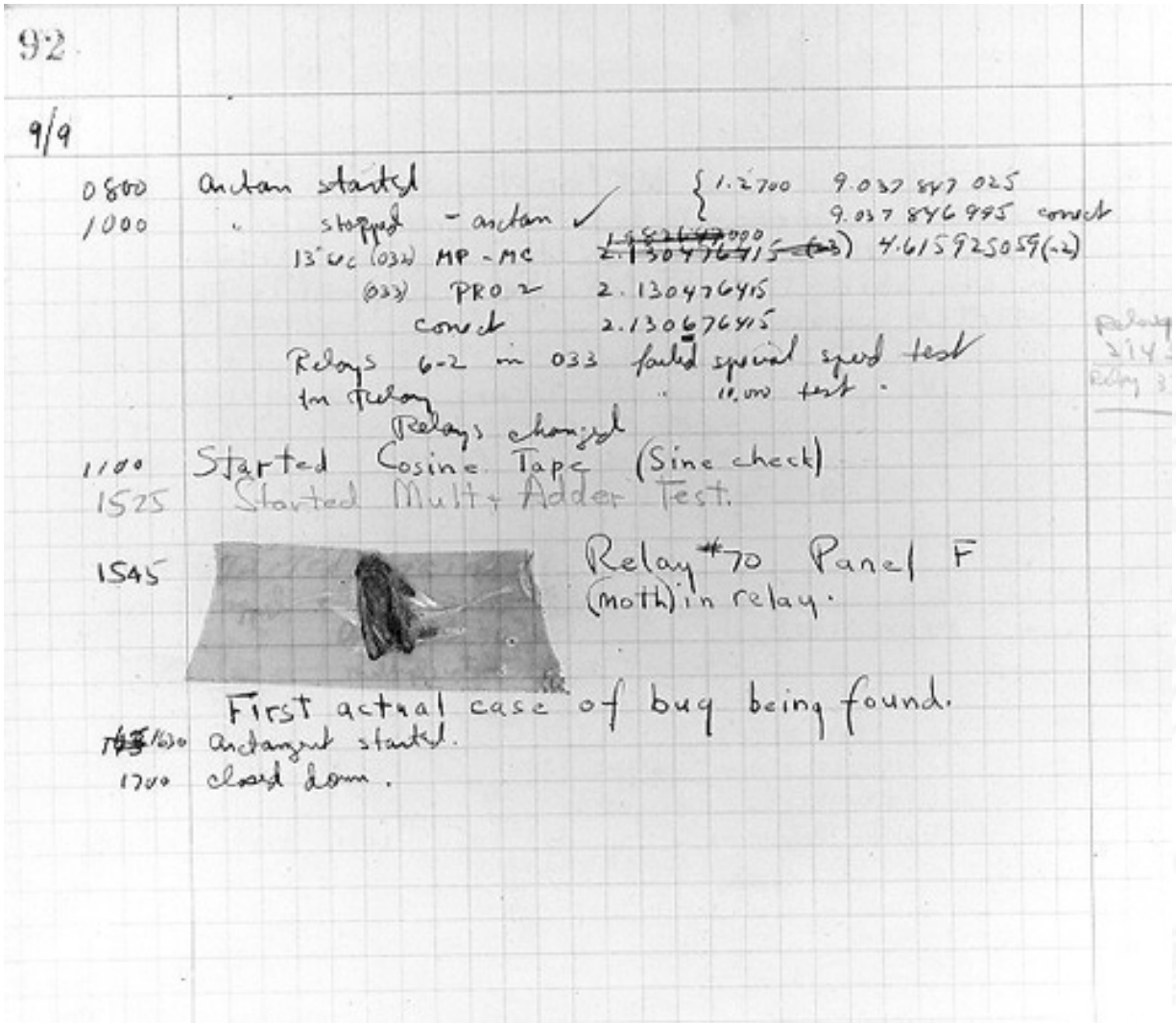


ENIAC – Electronic Numerical Integrator and Computer

Mauchly & Eckert, 1946: Röhren, Steckbrett-Programm



First computer bug



92

9/9

0800 Arctan started

1000 " stopped - arctan ✓ { 1.2700 9.037 847 025
9.037 846 995 correct

13"uc (032) MP - MC ~~2.130476415 (-2)~~ 4.615925059(-2)

(033) PRO 2 2.130476415

correct 2.130676415

Relays 6-2 in 033 failed special speed test
tm relay
" 11.00 test.
Relays changed

1100 Started Cosine Tape (Sine check)

1525 Started Mult + Adder Test.

1545 Relay #70 Panel F
(moth) in relay.

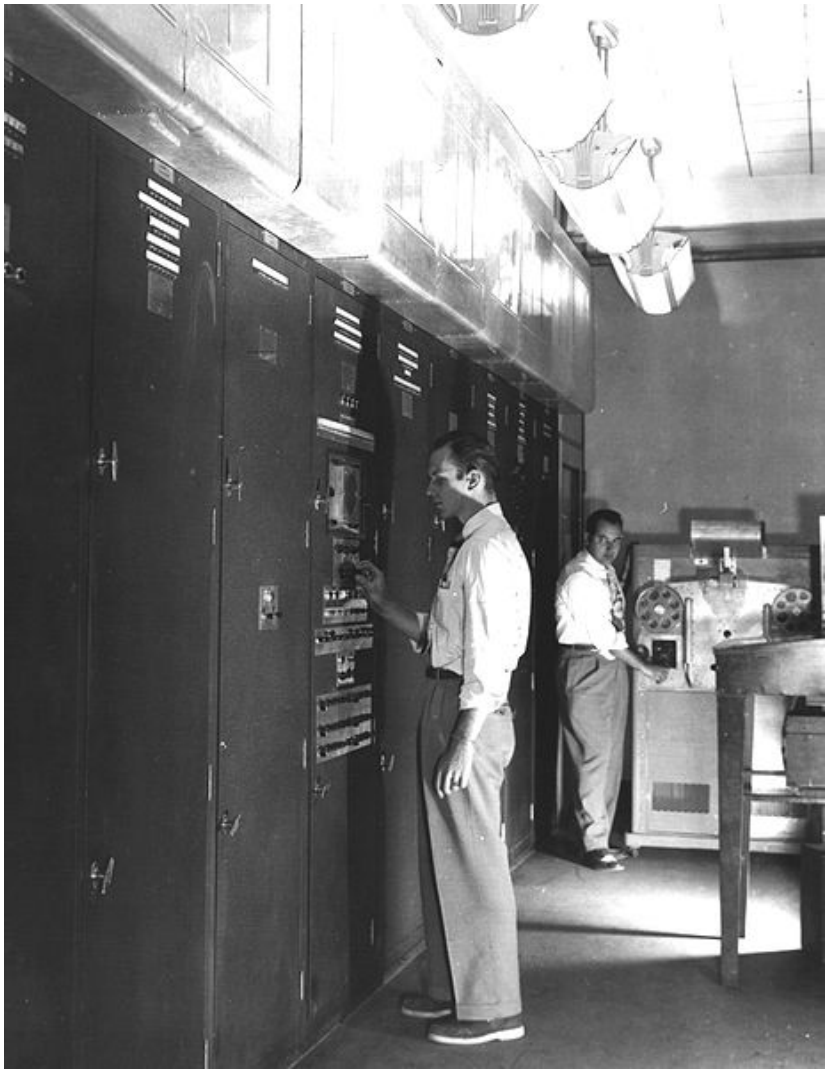
First actual case of bug being found.

~~1655~~ 1700 Arctangent started.

1700 closed down.

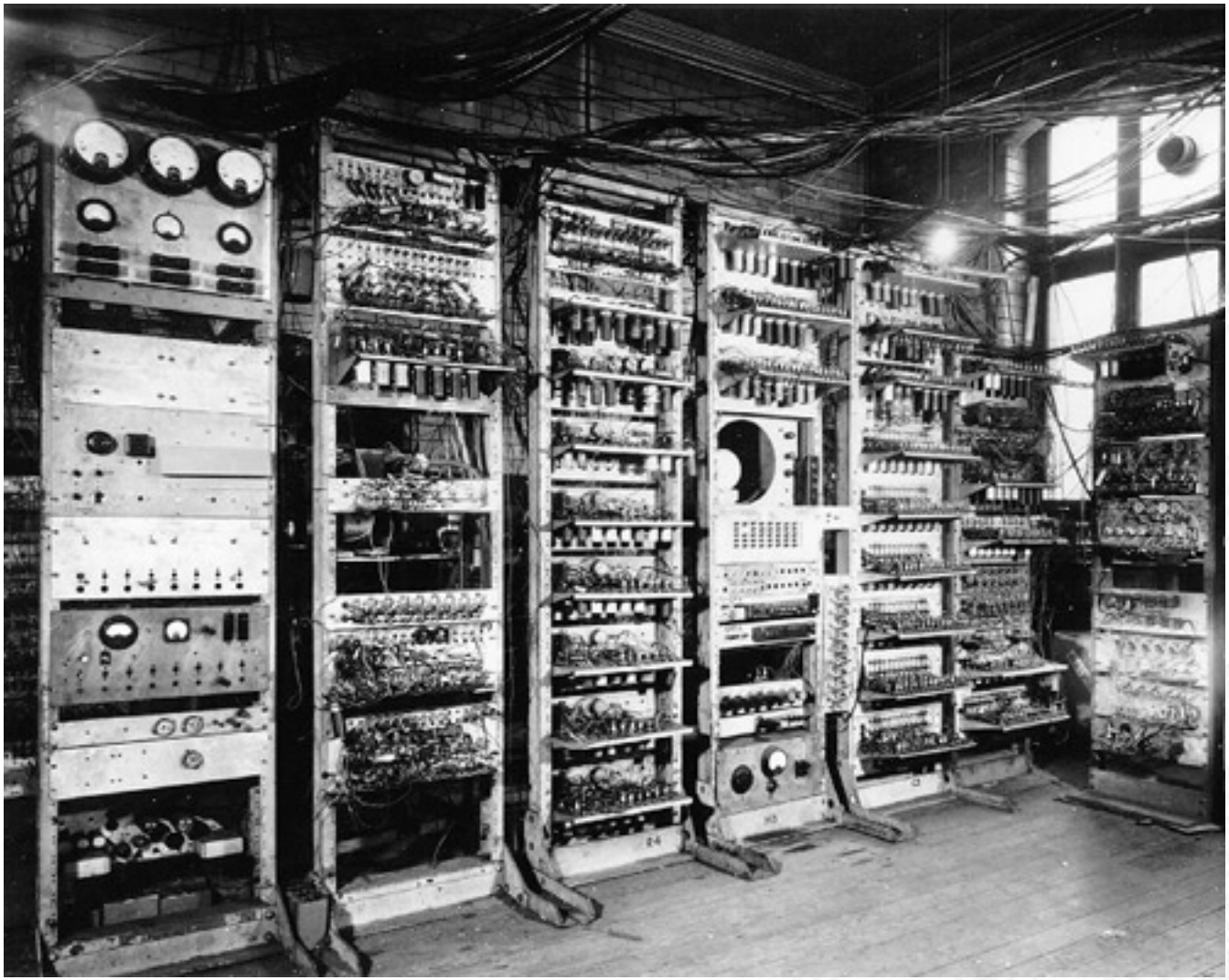
EDVAC

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
2004: AMD Athlon	64-bit, MMX/SSE
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

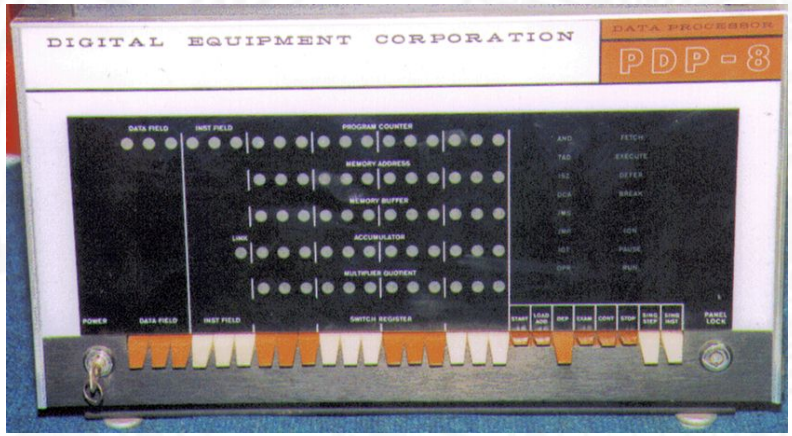
Einführung von **Betriebssystemen**

- „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

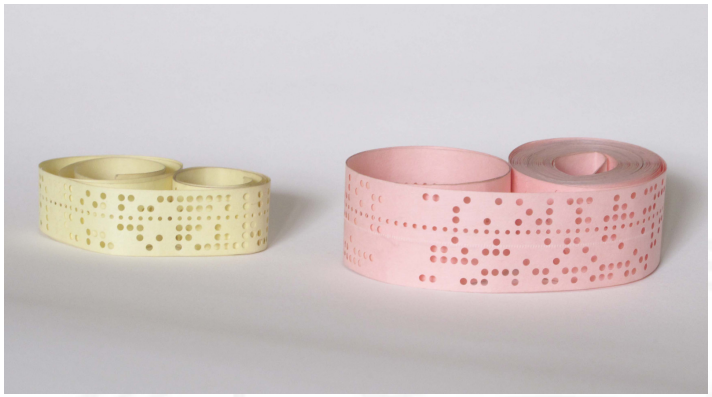
- Beispiel Digial Equipment Corporation PDP-1 (1961)
  - 4K Speicher (4096 Worte á 18-bit)
  - 200 kHz Taktfrequenz
  - 120 000 $
  - Grafikdisplay: erste Computerspiele
- Nachfolger PDP-8: 16 000 $
  - erstes Bussystem
  - 50 000 Stück verkauft



Festplatten

Massenspeicher bei frühen Computern

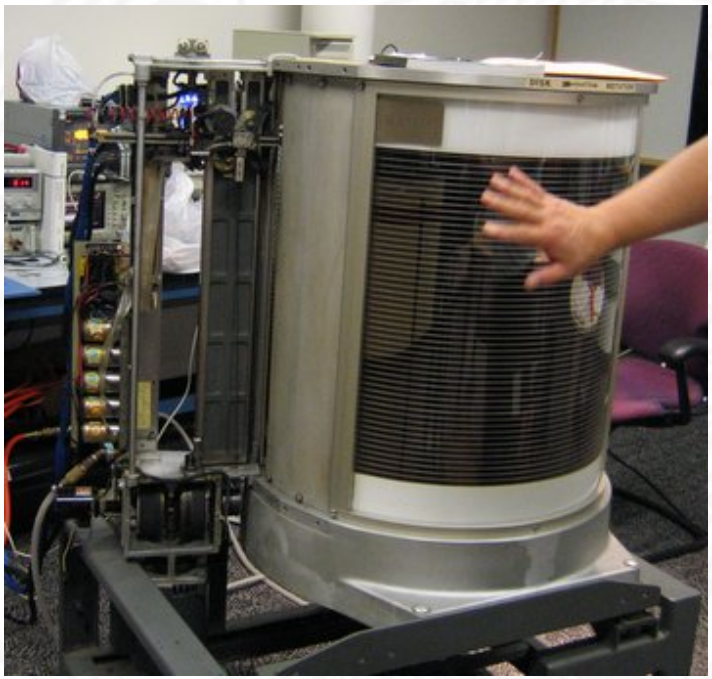
- Lochkarten
- Lochstreifen
- Magnetband



- Magnettrommel
- Festplatte

  IBM 350 RAMAC (1956)

  5 MByte, 600 ms Zugriffszeit



https://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

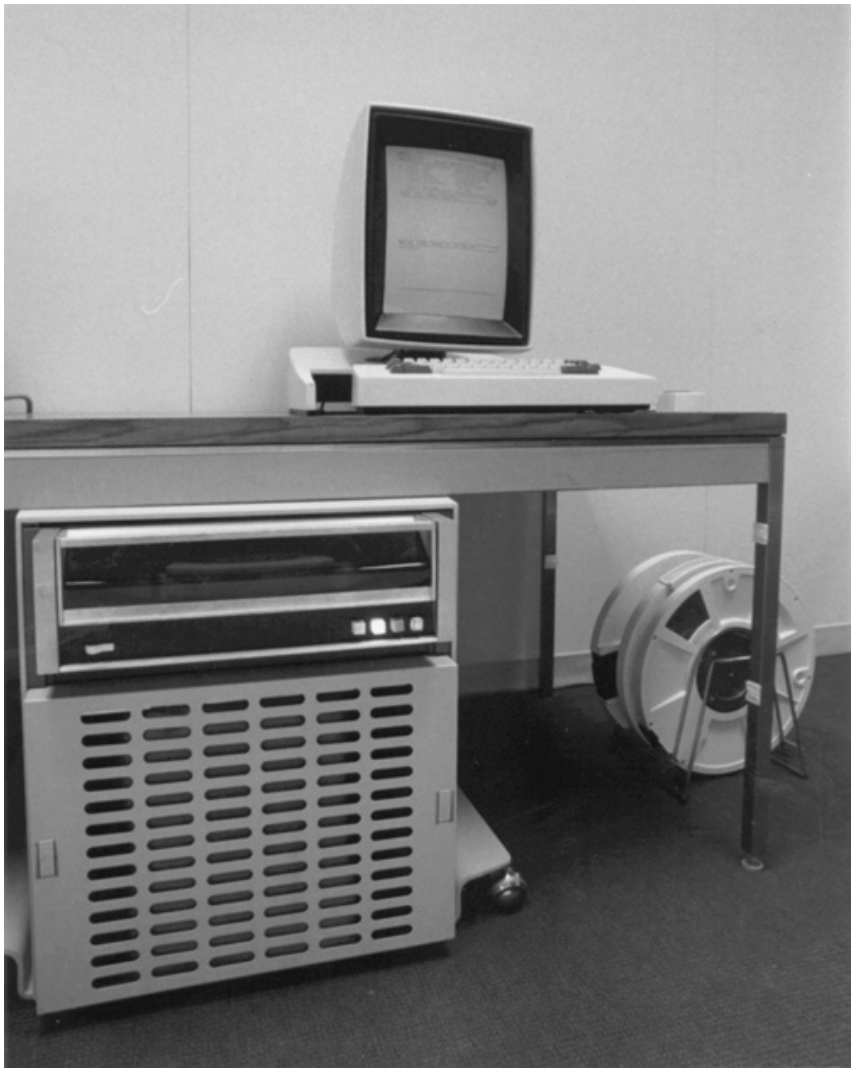
1975 Intel 8080, Motorola 6800, MOS 6502, ...

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`