64-040 Modul IP7: Rechnerstrukturen

1. Einführung, von-Neumann Rechner

Norman Hendrich

Universität Hamburg
MIN Fakultät, Department Informatik
Vogt-Kölln-Str. 30, D-22527 Hamburg
hendrich@informatik.uni-hamburg.de

WS 2013/2014

Inhalt

Inhalt und Lernziele

- Wie funktioniert ein Digitalrechner?
- Warum Mikroprozessoren?

Kennenlernen der Themen:

- Prinzip des von-Neumann-Rechners
- Abstraktionsebenen, Hardware/Software-Schnittstelle
- Rechnerarithmetik, Zahldarstellung, Kodierung
- Prozessor mit Steuerwerk und Operationswerk
- Speicher und -ansteuerung, Adressierungsarten
- Befehlssätze, Maschinenprogrammierung
- Assemblerprogrammierung, Speicherverwaltung

- Fähigkeit zum Einschätzen zukünftiger Entwicklungen
- Chancen und Grenzen der Miniaturisierung

Motivation

- Wie funktioniert ein Digitalrechner?
- Mikroprozessoren?

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, ...
- Bewertung von Trends und Perspektiven

Motivation

- ständige Fortschritte in Mikroelektronik und Optoelektronik
- und zwar weiterhin *exponentielles* Wachstum (50 %. . . 100 % pro Jahr)
  - Rechenleistung von Prozessoren („Performance“)
  - Speicherkapazität (DRAM, SRAM, FLASH)
  - Speicherkapazität (Festplatten)
  - Bandbreite (Netzwerke)
- ständig neue Möglichkeiten und Anwendungen
- ständig neue Produkte und Techniken
- und ganz gewiss kein „stationärer Zustand“
- Roadmaps derzeit bis über 2020 hinaus. . .

Technologie-Fortschritt

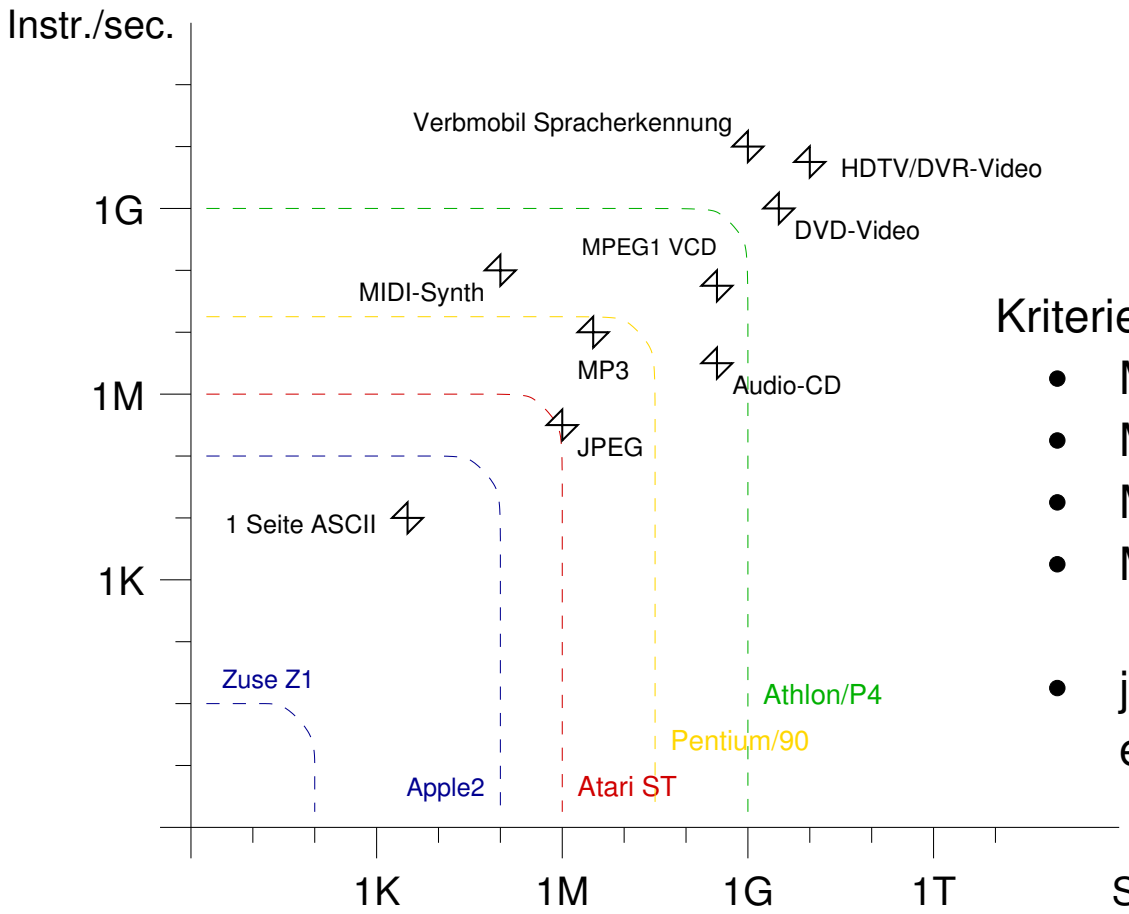
- exponentielles Wachstum, typisch 50 % pro Jahr
- ständig neue Möglichkeiten und Anwendungsfelder
- ständig neue Produkte und Techniken

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

- Schwerpunkt der Vorlesung auf dem „Warum“
- bitte ein Gefühl für Größenordnungen entwicklen

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

Technologie-Fortschritt: neue Anwendungsfelder



Kriterien:

- MIPS
- MBytes (RAM, Platte)
- Mbps
- MPixel

- jede Rechnergeneration erlaubt neue Anwendungen

Neue Anwendungsfelder: Beispiel ReBirth

- Techno per Software: Echtzeit-Software-Emulation der legendären Roland Synthesizer TB-303 TR-808 TR-909 auf einem PC

(Propellerheads ReBirth 1996, www.rebirthmuseum.com)

Neue Anwendungsfelder: Beispiel Autotune

Sie sehen gut aus, aber Ihr Gesang ist lausig?

(Antares Autotune 1999)

Themen heute

- Geschichte der Datenverarbeitung
- Wichtige Beispiele

- Technologie-Fortschritt, Skalierung
- Moore's Gesetz, ITRS-Roadmap
- Grenzen der Miniaturisierung: Smart-Dust

- Grundprinzip des von-Neumann-Rechners
- Aufbau, Befehlszyklus, Befehlssatz

Definition: Digitalrechner

A digital computer is a machine that can solve problems for people by carrying out instructions given to it.
A sequence of instructions describing how to perform a certain task is called a program.
The electronic circuits of each computer can recognize and directly execute a limited set of simple instructions into which all its programs must be converted before they can be executed.

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

(Tanenbaum: Structured Computer Organization)

Befehlssatz und Semantic Gap

- . . . *directly execute a limited set of simple instructions*. . .

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

- extrem niedriges Abstraktionsniveau
- vgl.: „vereinbaren Sie einen Termin mit dem Steuerberater.“
- das sogenannte **semantic gap**

- 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:

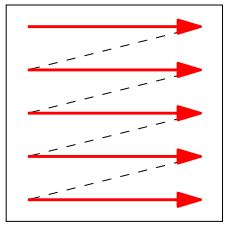
- Gesamtverständnis erfordert Kenntnisse auf allen Ebenen
- häufig Rückwirkung von unteren auf obere Ebenen

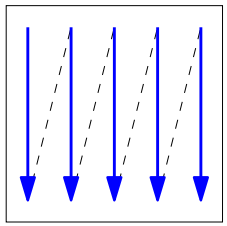
Rückwirkung von unteren Ebenen: Arithmetik

```
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 (5000,5000) creation took 2105 msec.
Matrix row-col summation took 75 msec.
Matrix col-row summation took 383 msec.          (5x langsamer)
Sum is 600.8473695346258 600.8473695342268       (andere Werte)
```

Maschine mit mehreren Ebenen

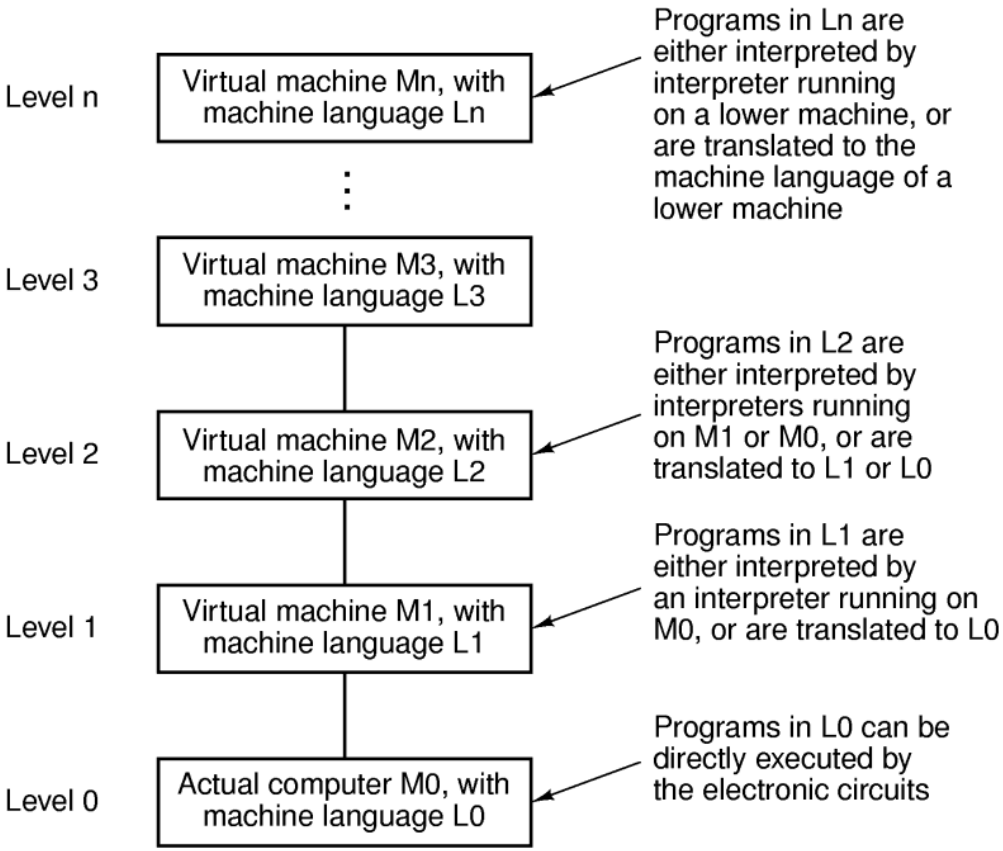


Figure 1-1. A multilevel machine.

(Tanenbaum: Structured Computer Organization)

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

dies ist gleichwertig mit einer:

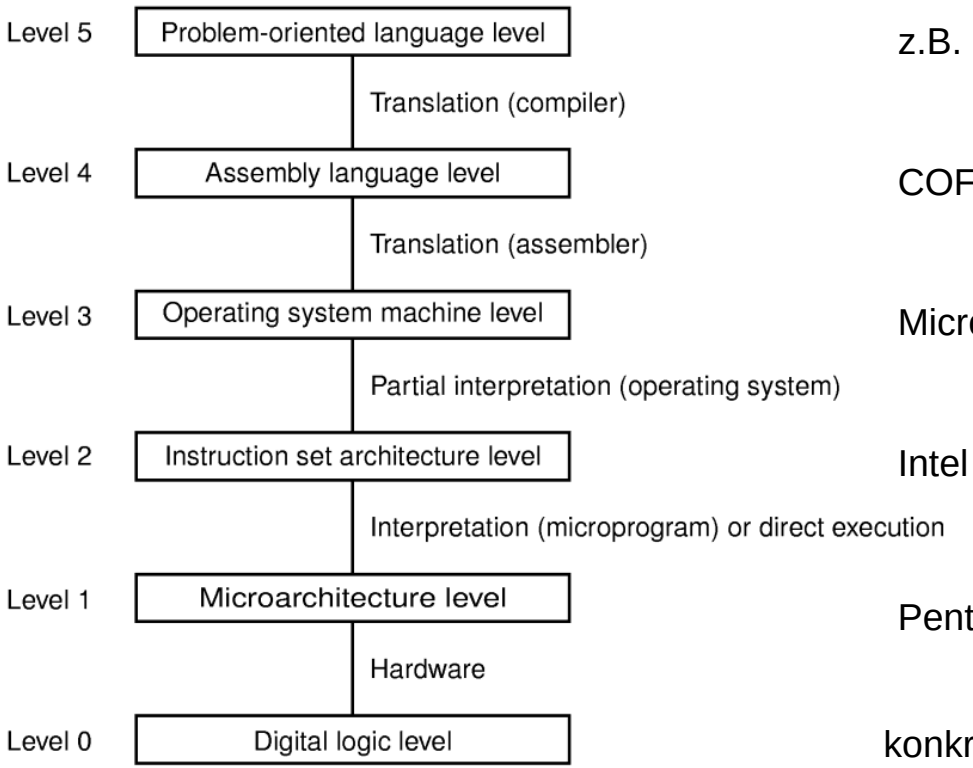
Virtuellen Maschine M1 für die Sprache L1

- 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



z.B. Smalltalk, Visual Basic

COFF, ELF, ...

Microsoft Windows XP

Intel x86+MMX+SSE2

Pentium-IV 512KB Cache

konkreter Pentium-IV Chip

Figure 1-2. A six-level computer. The support method for each level is supported is indicated below it (along with the name of the supporting program).

Hinweis: Ebenen vs. Vorlesungen im BSc-Studiengang

Anwendungsebene:	SE1..SE3, AD, ...
Assemblerebene:	RS
Betriebssystemebene:	GSS ES
Rechnerarchitektur:	RS, ES, RAM
Mikroarchitektur:	RS, RAM
Logikebene:	RS, RAM
Device-Level:	RAM

HelloWorld: Anwendungsebene: Quellcode

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

#include <stdio.h>

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

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

- 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
- System mit Prozessor, Speicher, Peripheriegeräten

- gemeinsamer Speicher für Programme und Daten
- Programme können wie Daten manipuliert werden
- Daten können als Programm ausgeführt werden

- Befehlszyklus: Befehl holen, dekodieren, ausführen
- enorm flexibel

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

von-Neumann Rechner

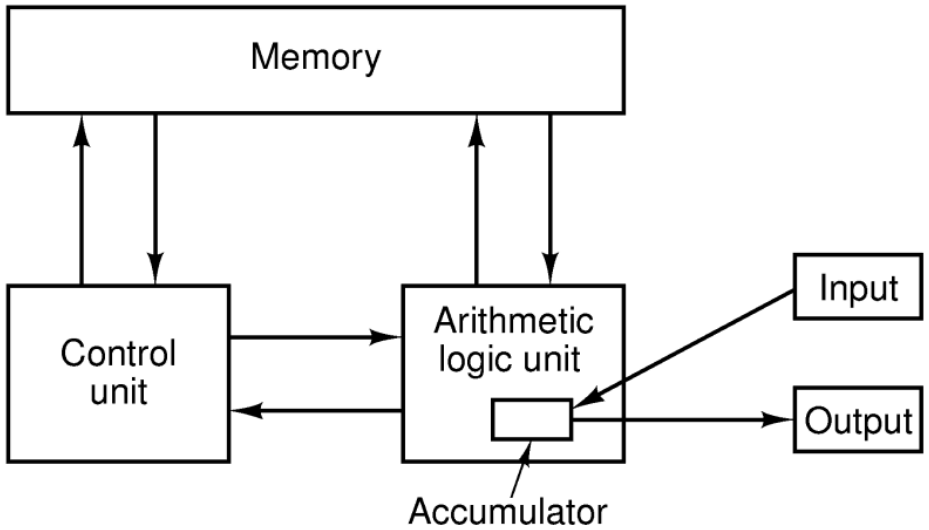


Figure 1-5. The original von Neumann machine.

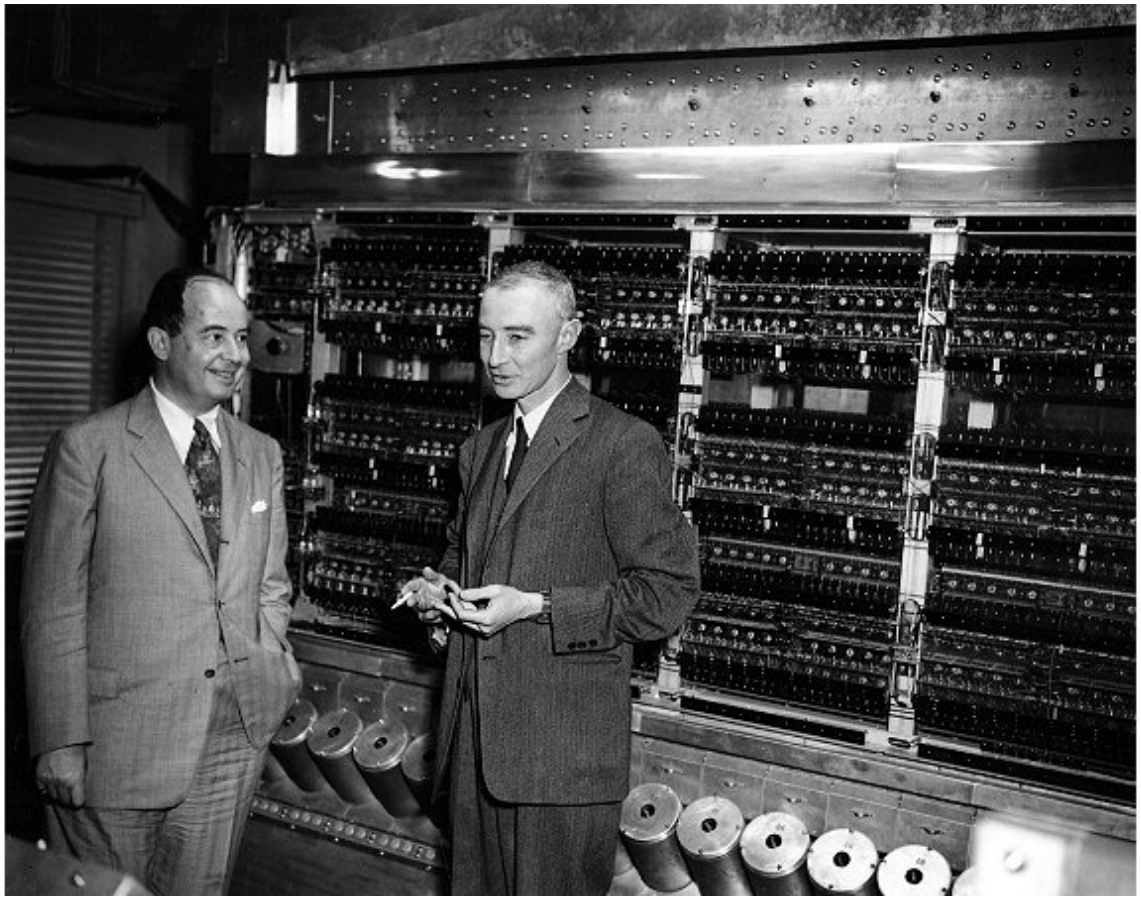
Fünf zentrale Komponenten:

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

von-Neumann Rechner

- Steuerwerk: zwei essentielle Register
  - Befehlszähler (*program counter PC*)
  - Befehlsregister (*instruction register IR*)
- Operationswerk (Datenpfad, *data-path*)
  - Rechenwerk (*arithmetic-logic unit ALU*)
  - Universalregister (mindestens 1 *Akkumulator*, typisch 8..64)
  - 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, computerhistory.org

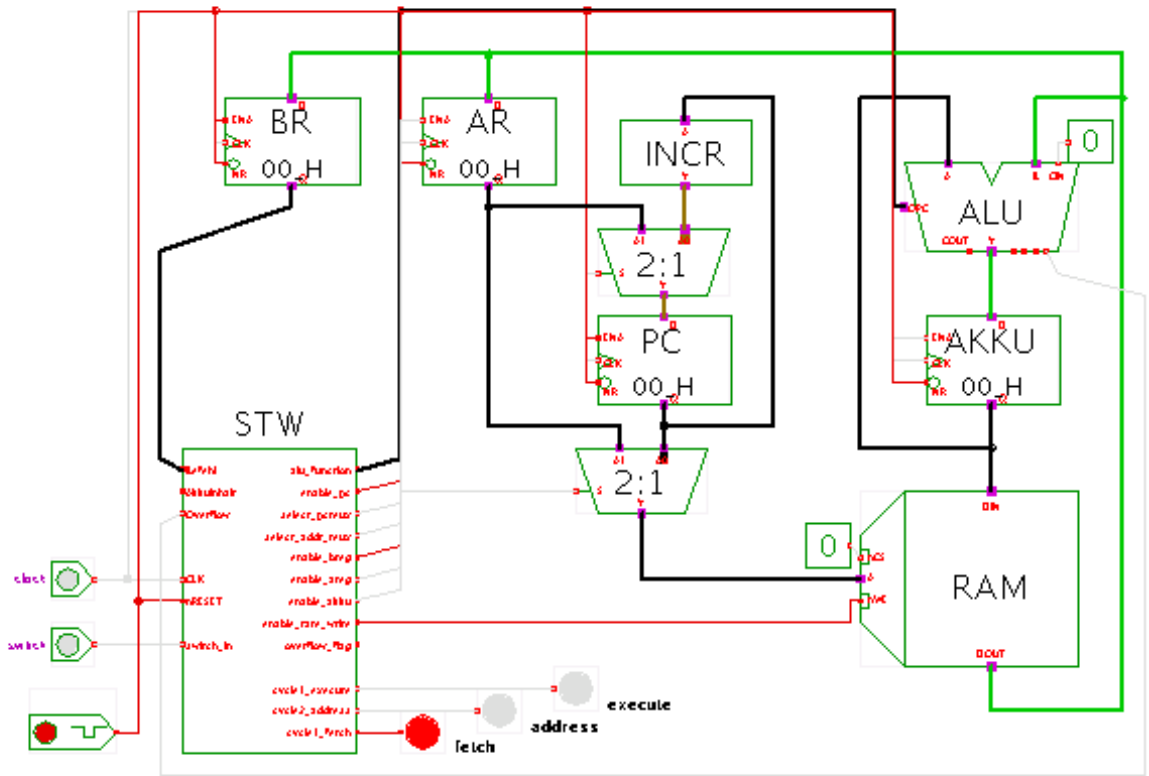
PRIMA: die Primitive Maschine

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-www.informatik.uni-hamburg.de/applets/hades/webdemos/50-rtlib/90-prima/chapter.html

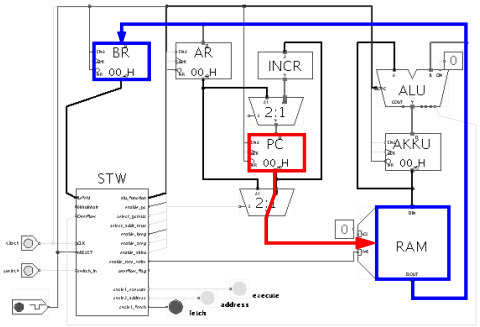
PRIMA: die Primitive Maschine



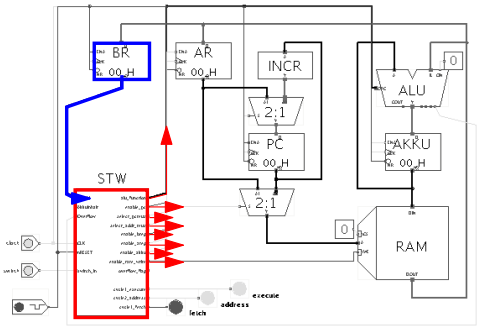
http://tams-www.informatik.uni-hamburg.de/applets/hades/webdemos/50-rtlib/90-prima/chapter.html

PRIMA: die Zyklen

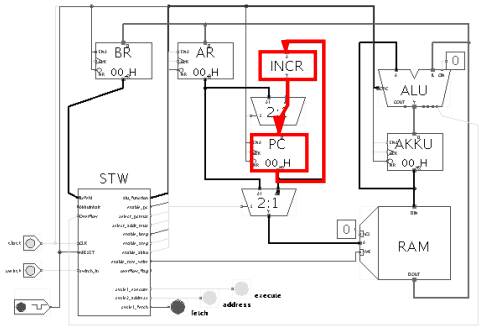
Befehl holen



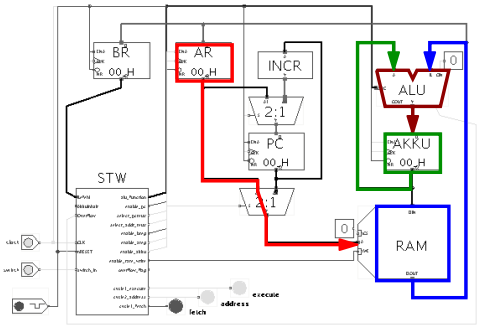
dekodieren



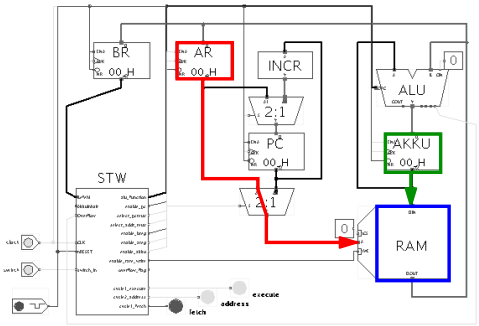
PC inkrementieren



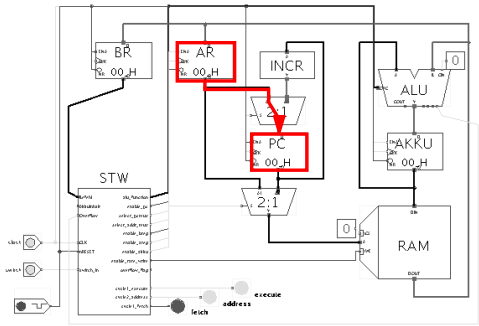
rechnen



speichern

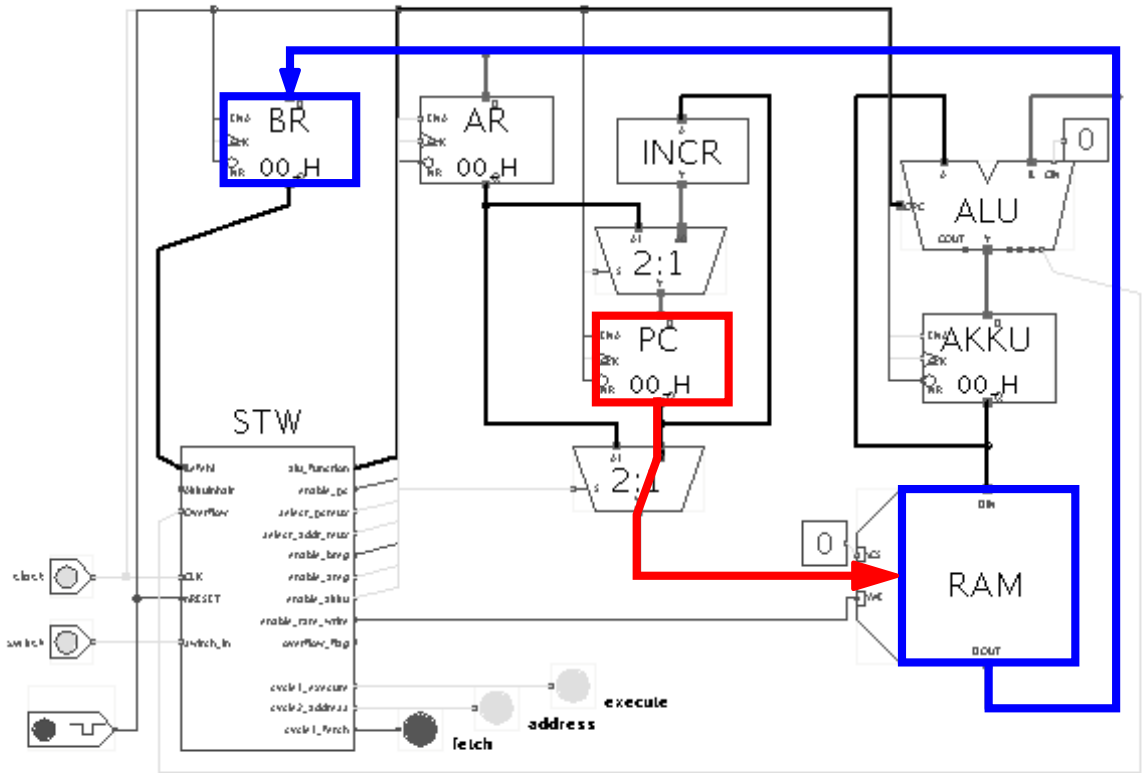


springen



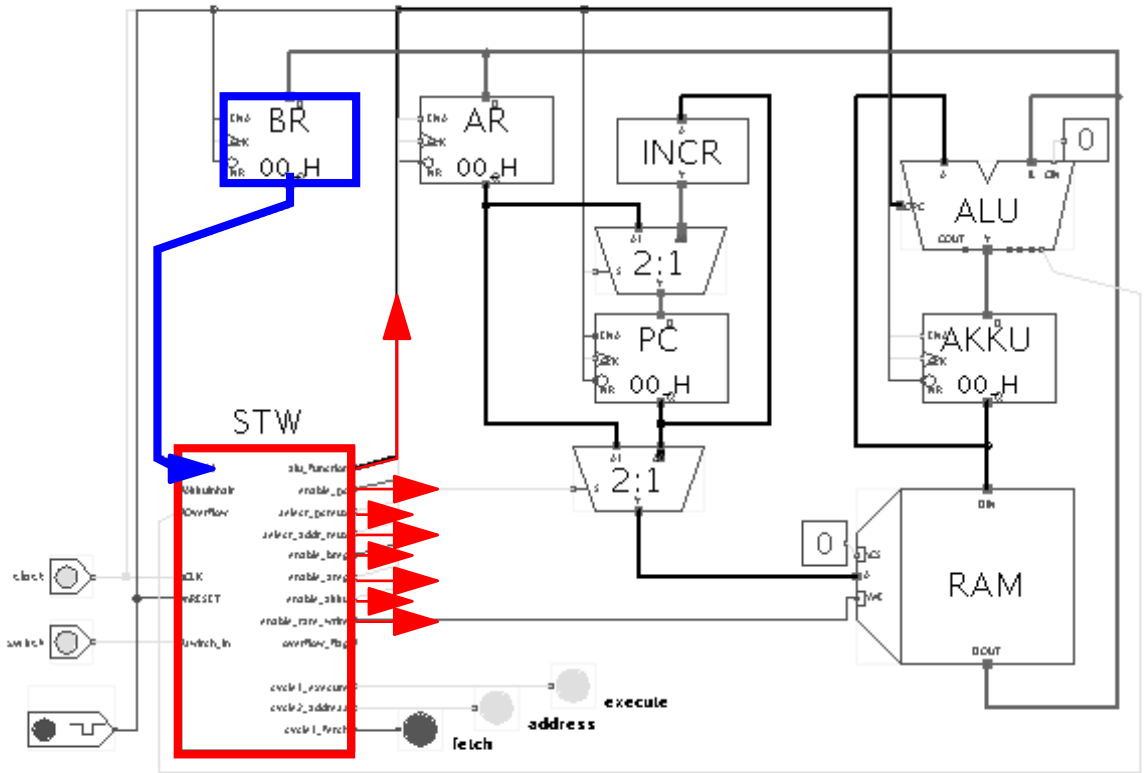
PRIMA: Befehl holen

BR = RAM[PC]



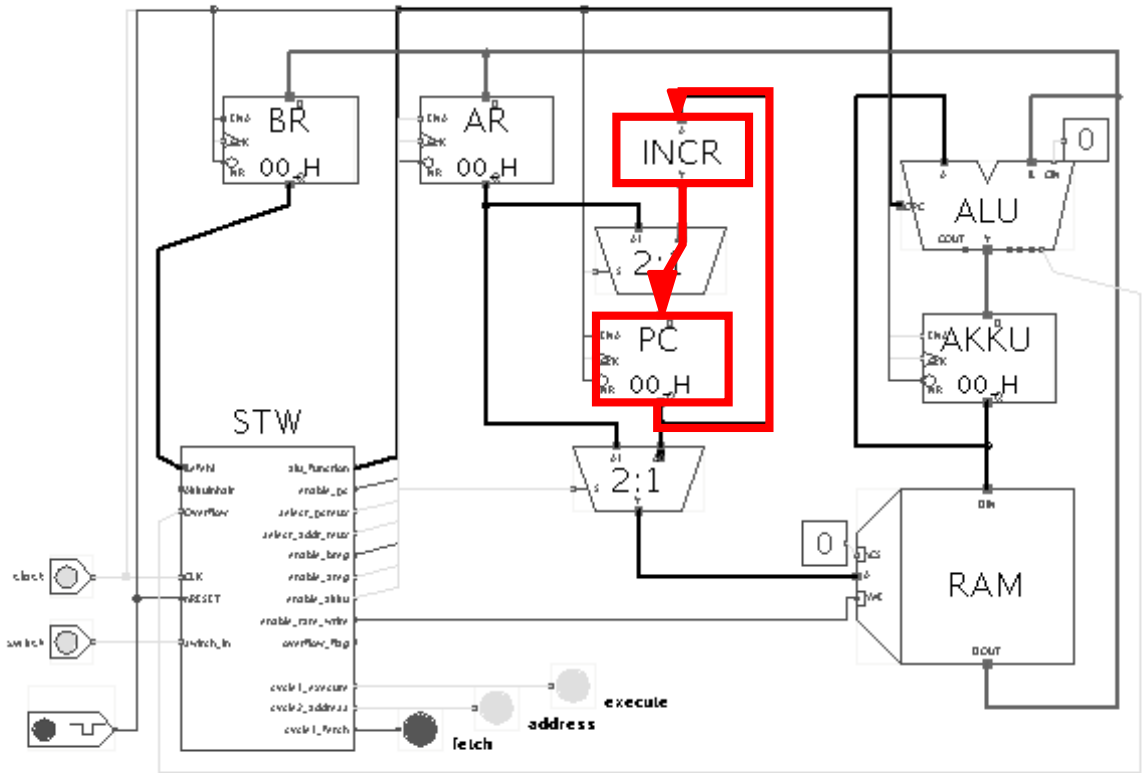
PRIMA: dekodieren

Steuersignale = decode(BR)



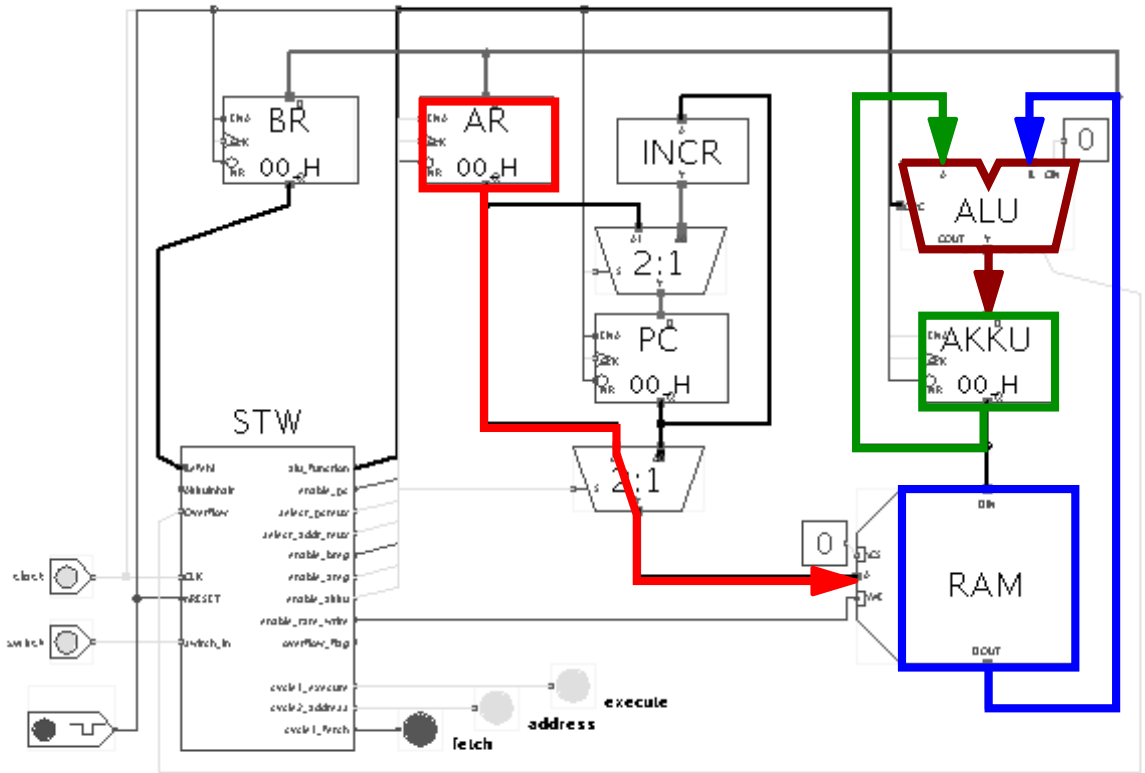
PRIMA: PC inkrementieren

PC = PC+1



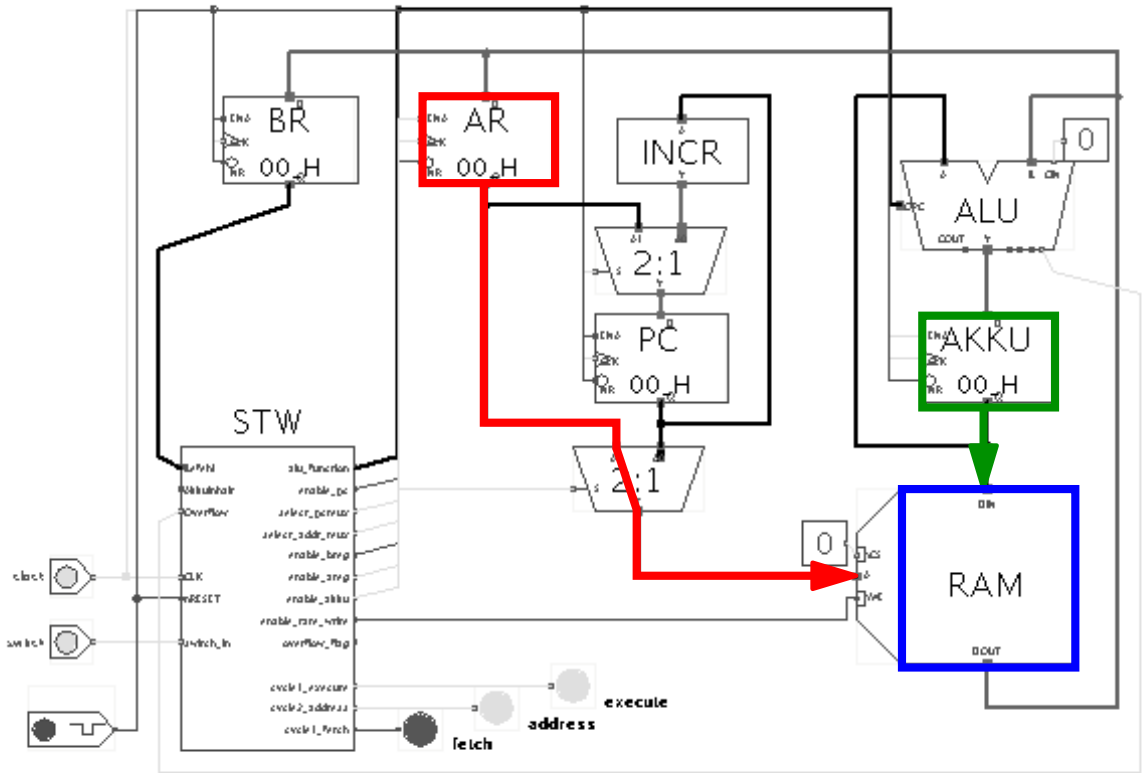
PRIMA: rechnen

Akku = Akku + RAM[AR]



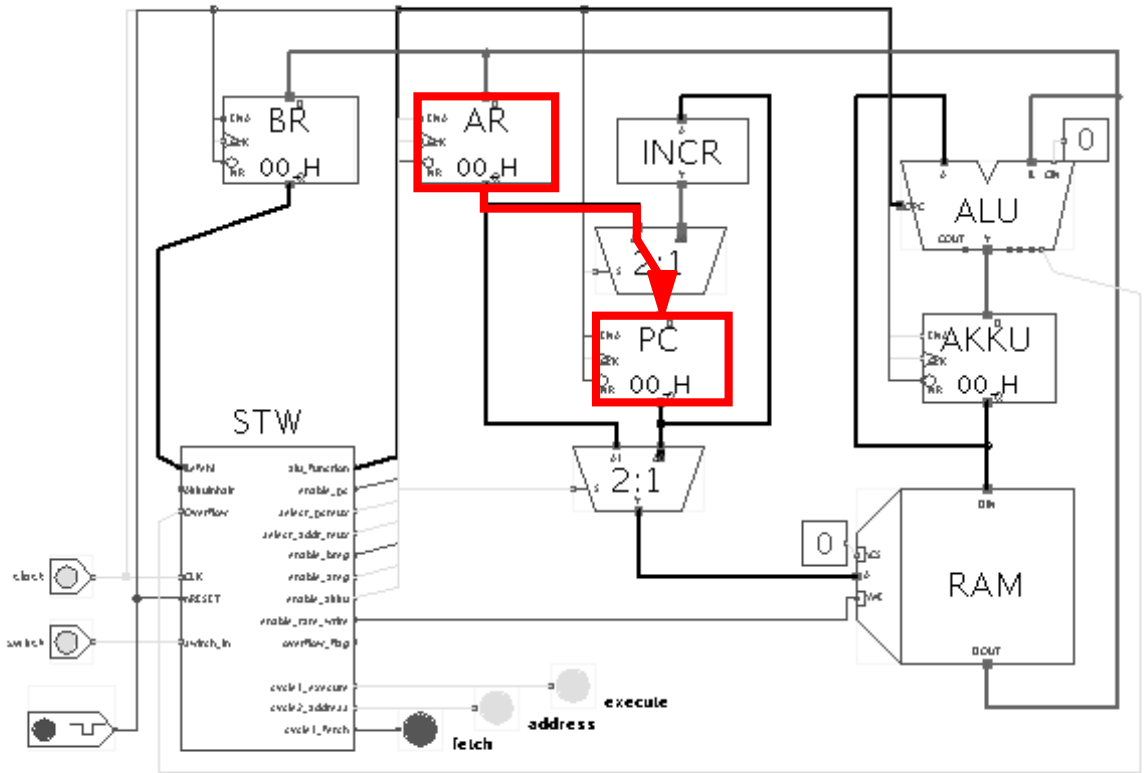
PRIMA: speichern

RAM[AR] = Akku

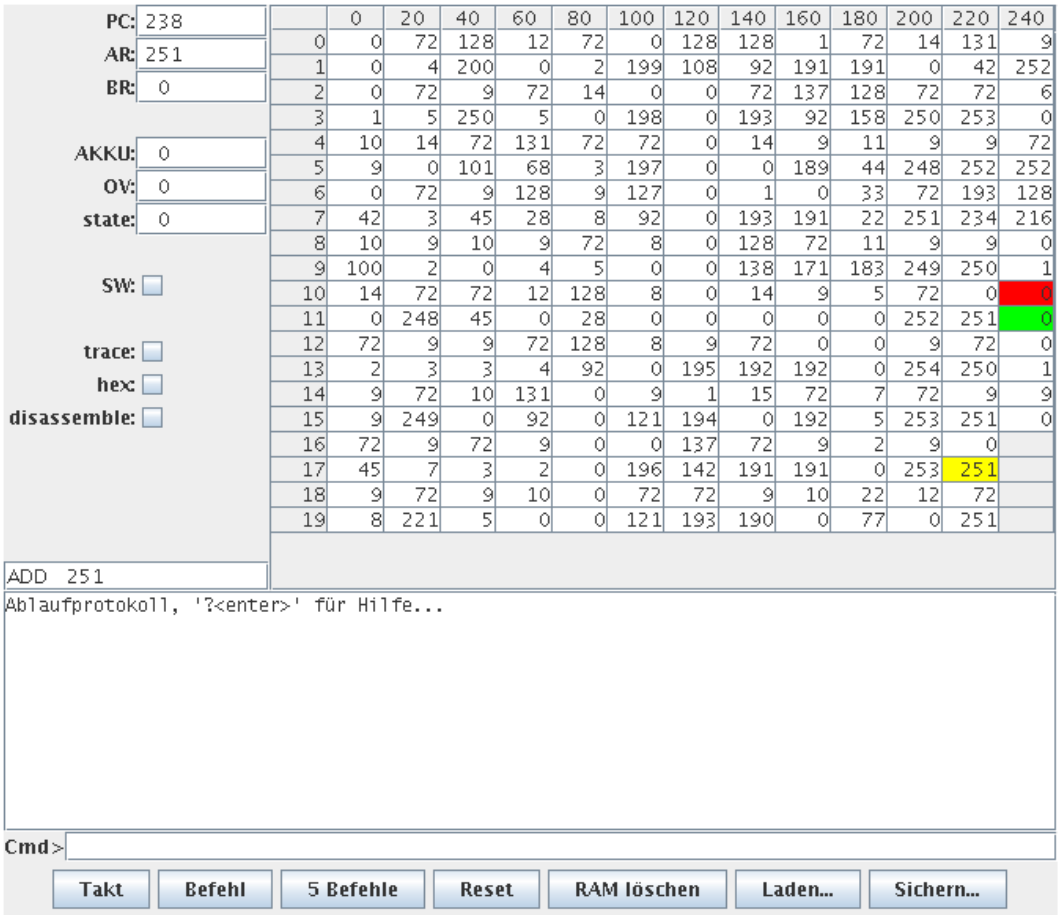


PRIMA: springen

PC = AR



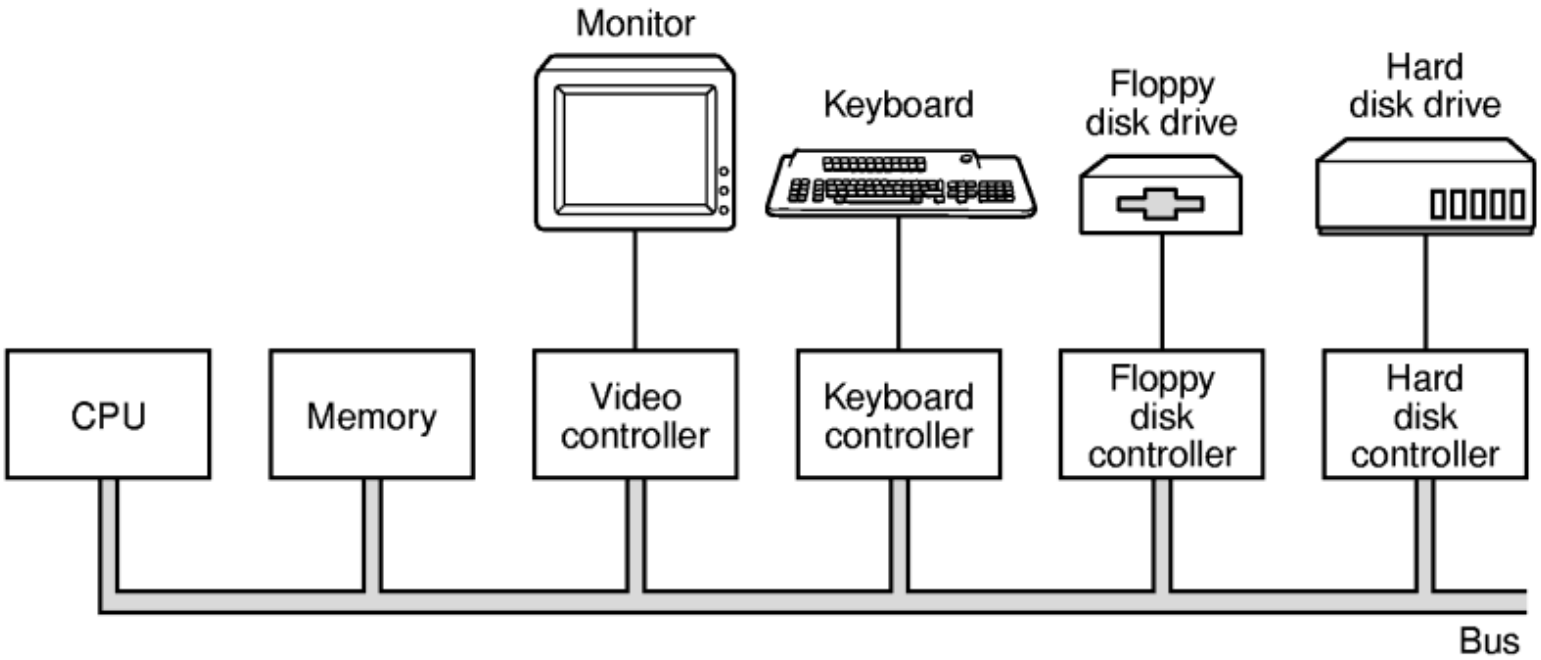
PRIMA: Simulator



	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	

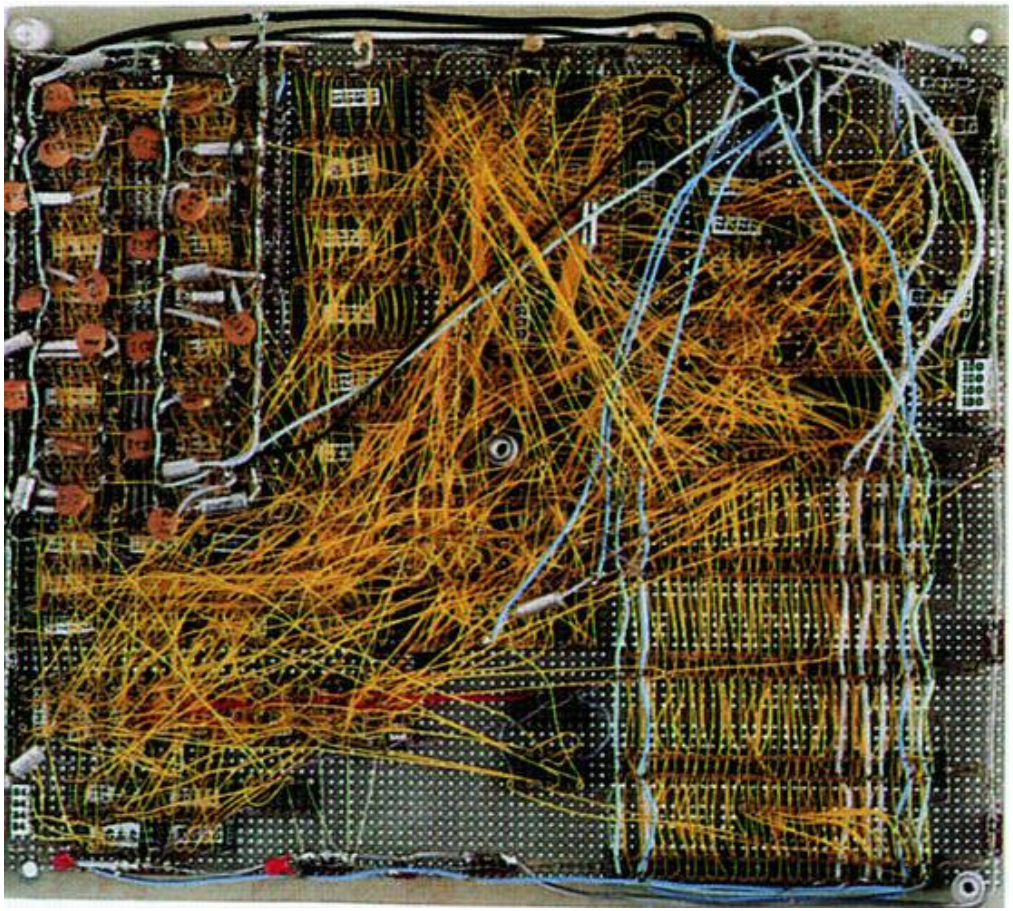
http://tams-www.informatik.uni-hamburg.de/applets/jython/prima.html

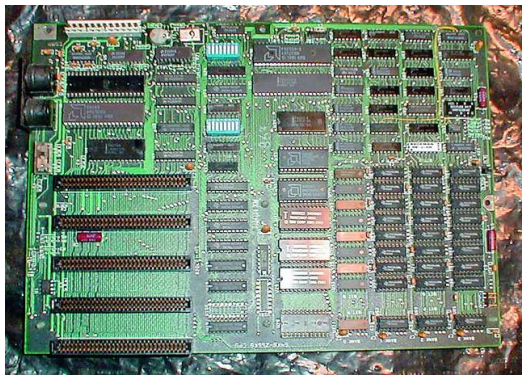
Personal Computer: Aufbau des IBM PC (1981)



- 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)

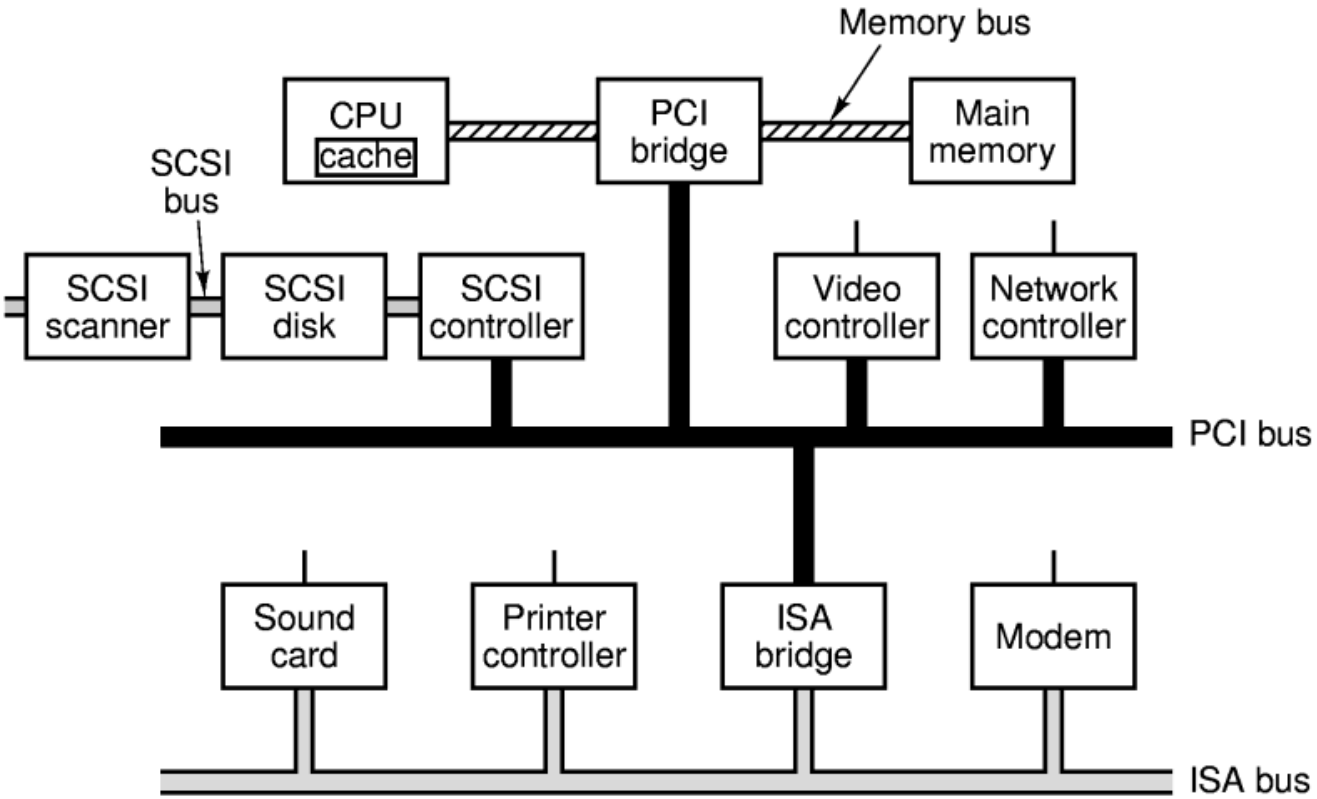
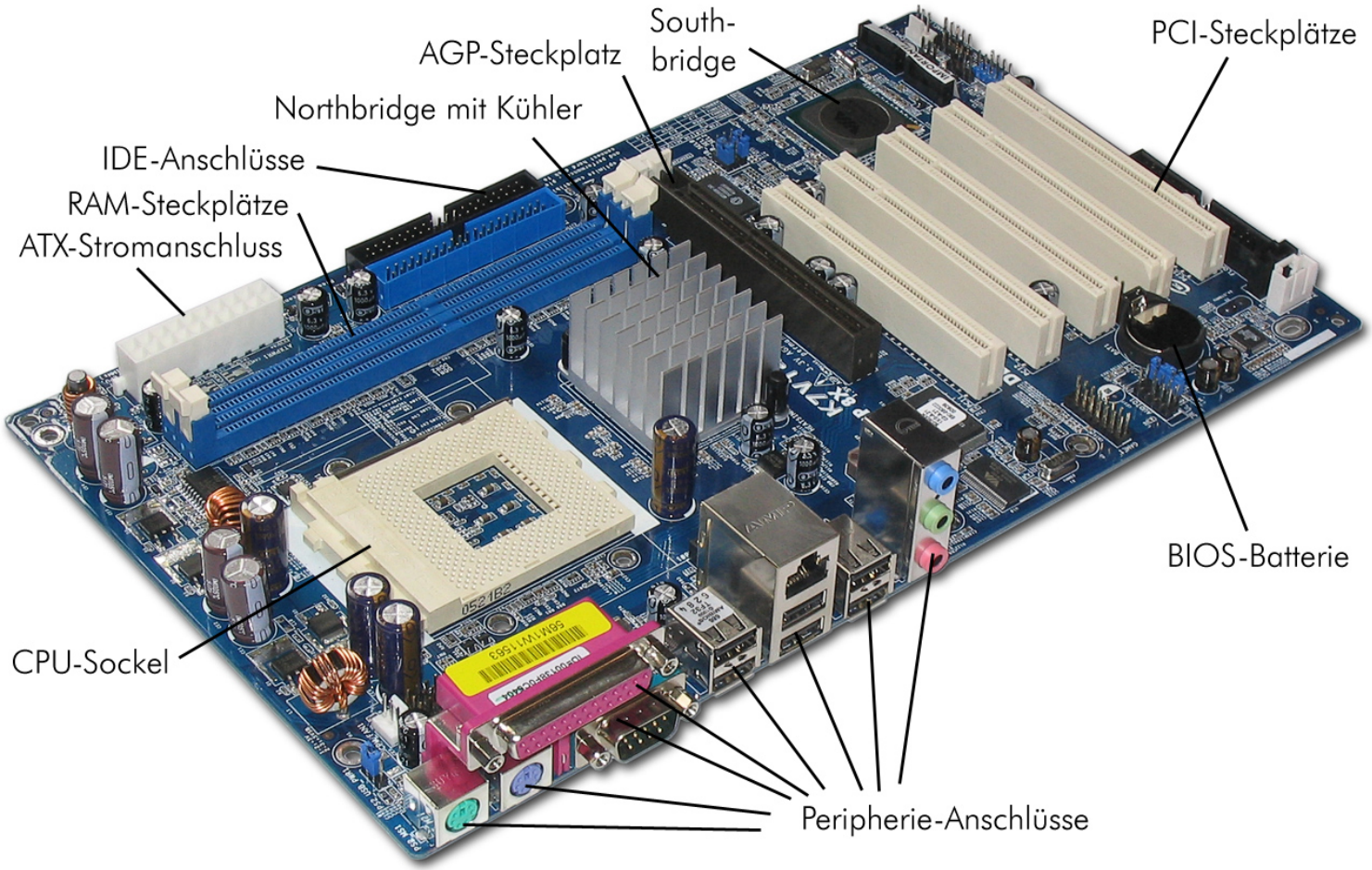


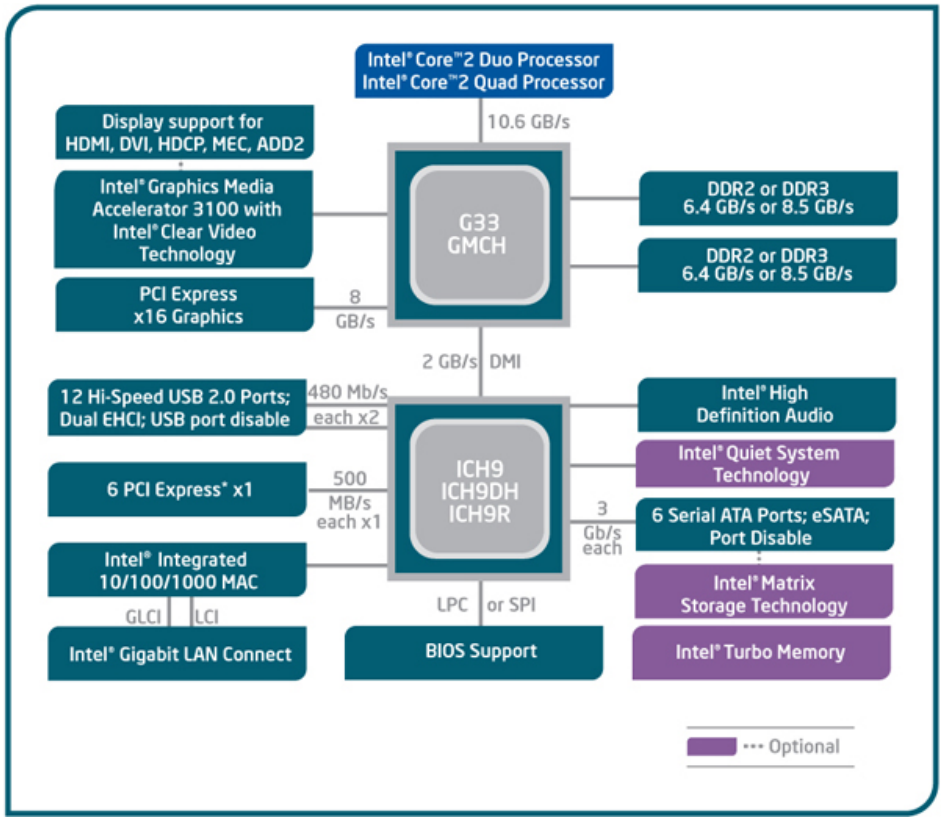
Figure 2-30. A typical modern PC with a PCI bus and an ISA bus. The modem and sound card are ISA devices; the SCSI controller is a PCI device.

Personal Computer: Hauptplatine (2005)



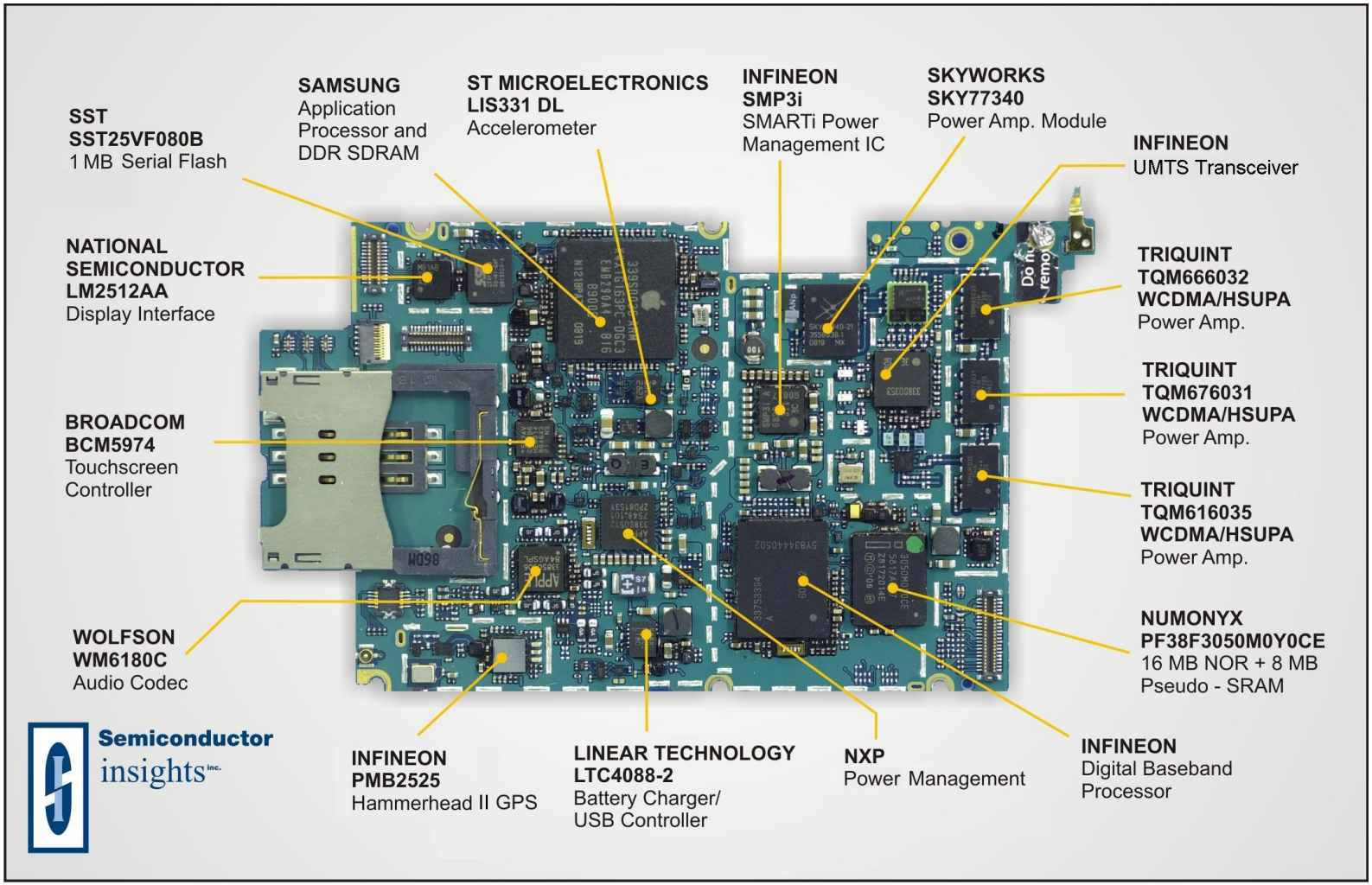
de.wikibooks.org/wiki/Computerhardware_für_Anfänger

Personal Computer: Aufbau (2010)



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

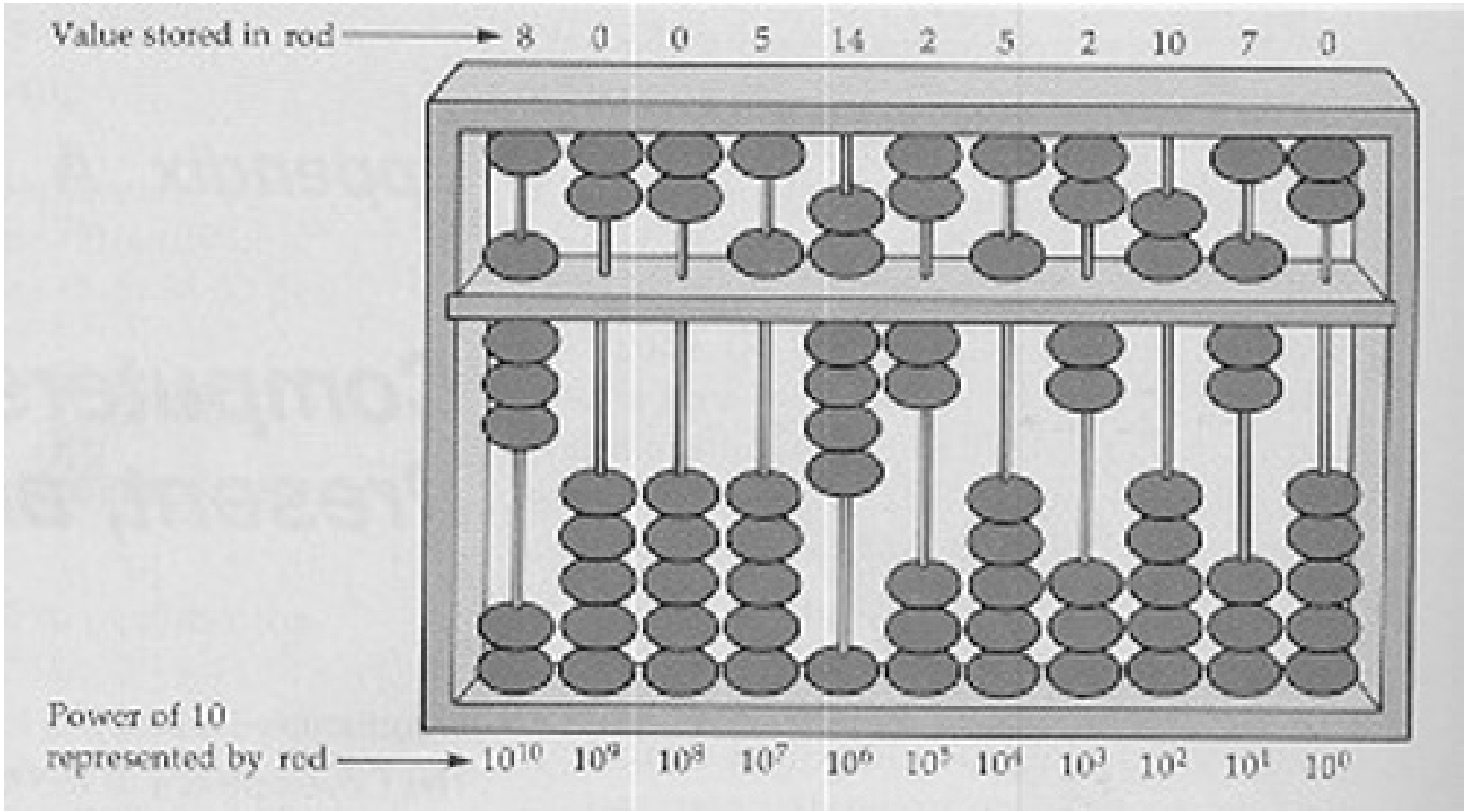
Mobilgeräte: Smartphone (2010)



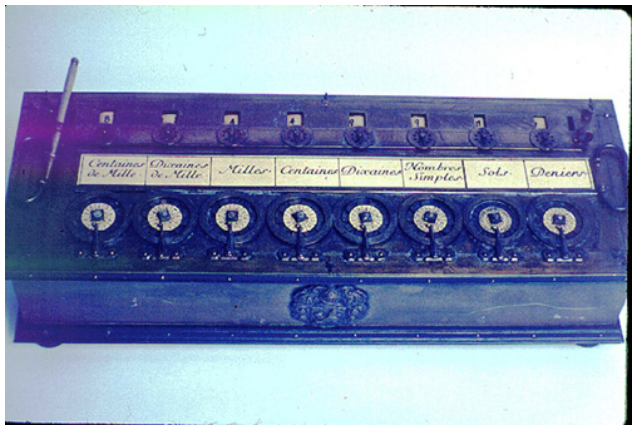
Timeline: Vorgeschichte

????	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


Value stored in rod
8 0 0 5 14 2 5 2 10 7 0
Power of 10
represented by rod
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

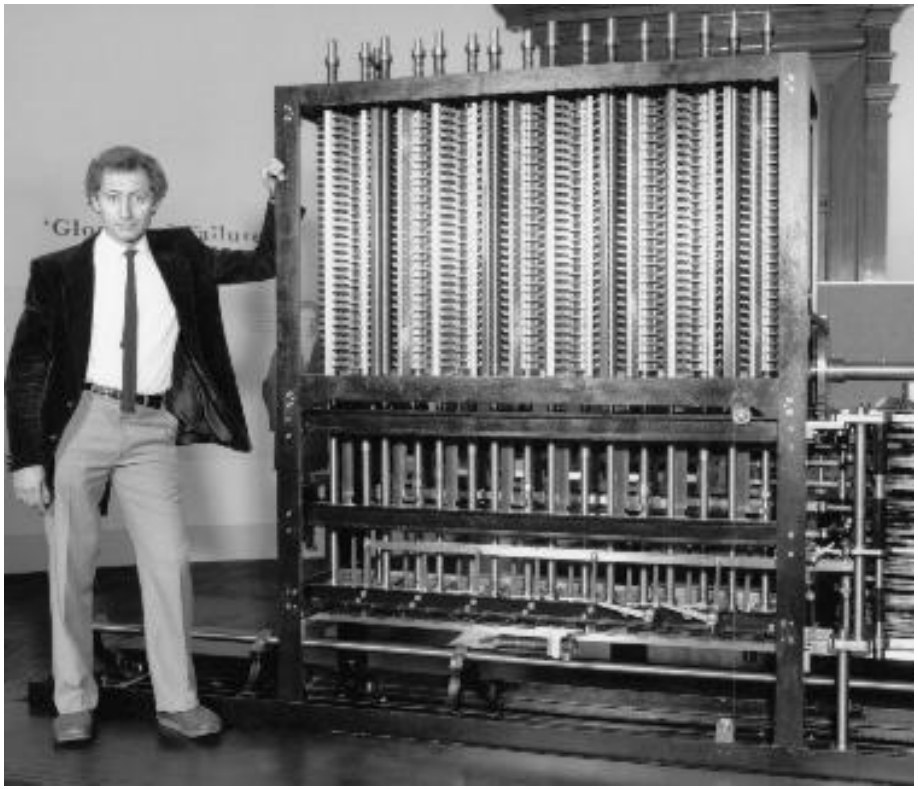
1642 Pascal: „Pascalene“

1673 Leibniz: Staffelwalze, Multiplikation/Division

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

Difference Engine

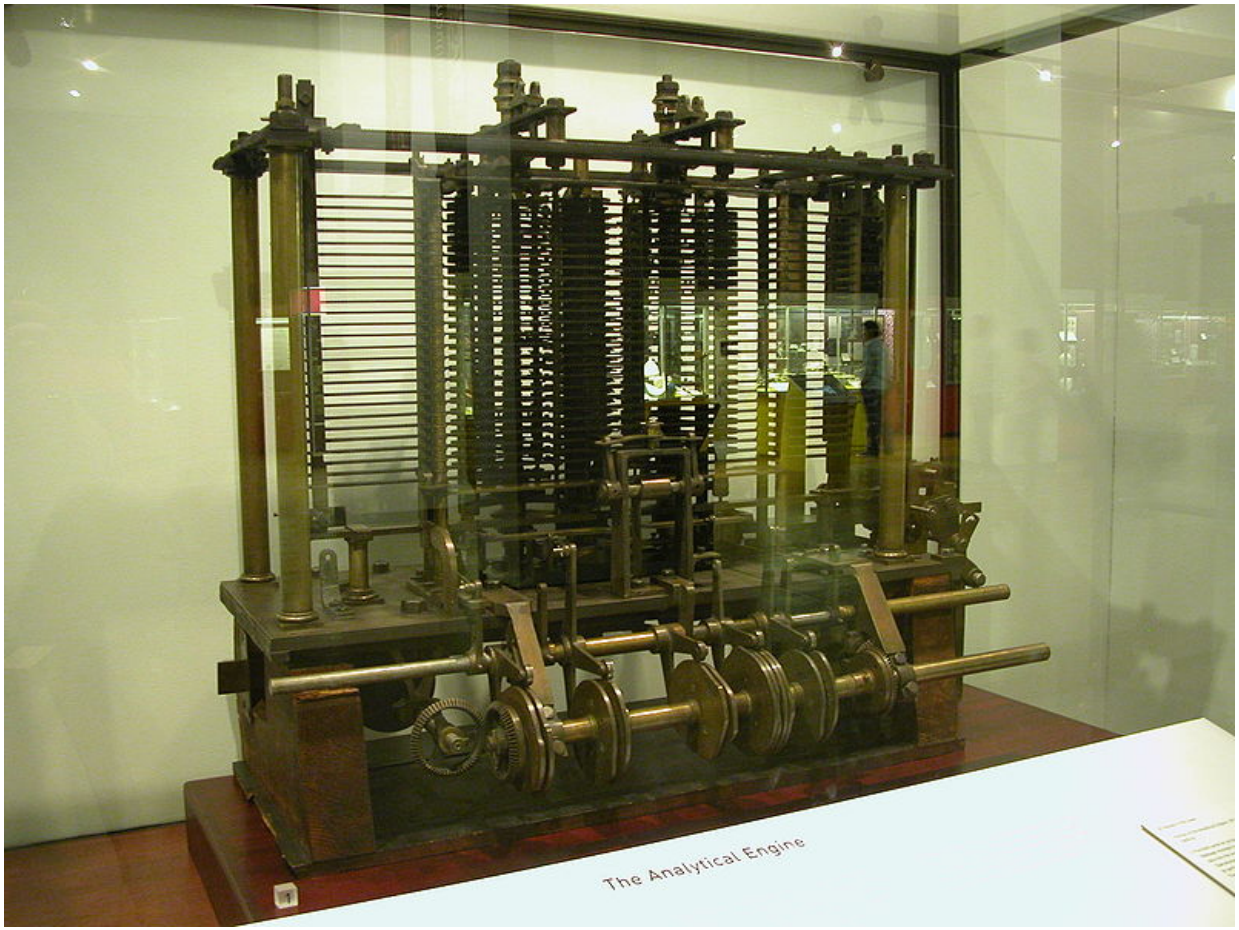
Charles Babbage 1822: Berechnung nautischer Tabellen



(Original von 1832 und Nachbau von 1989, London Science Museum)

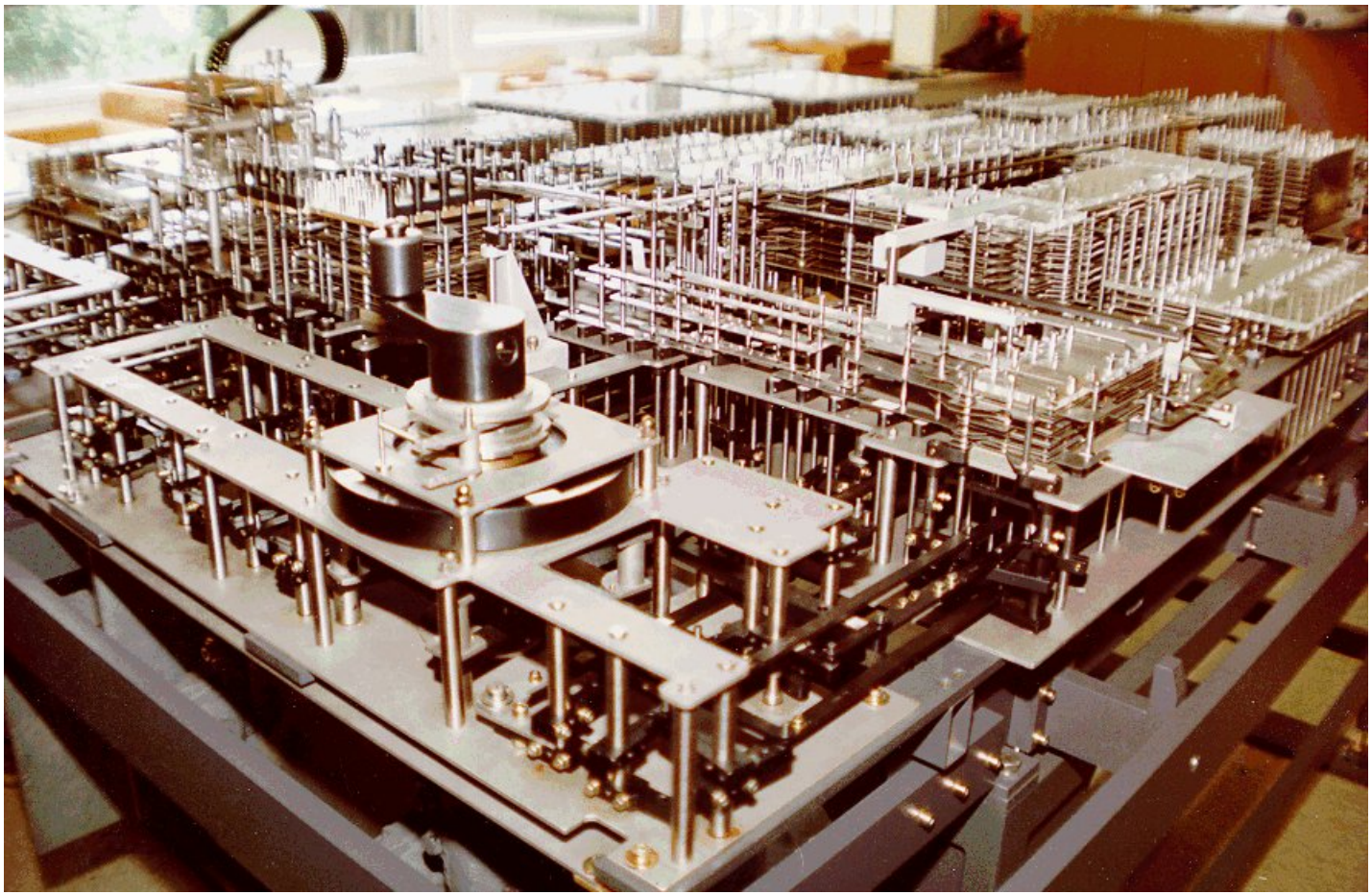
Analytical Engine

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



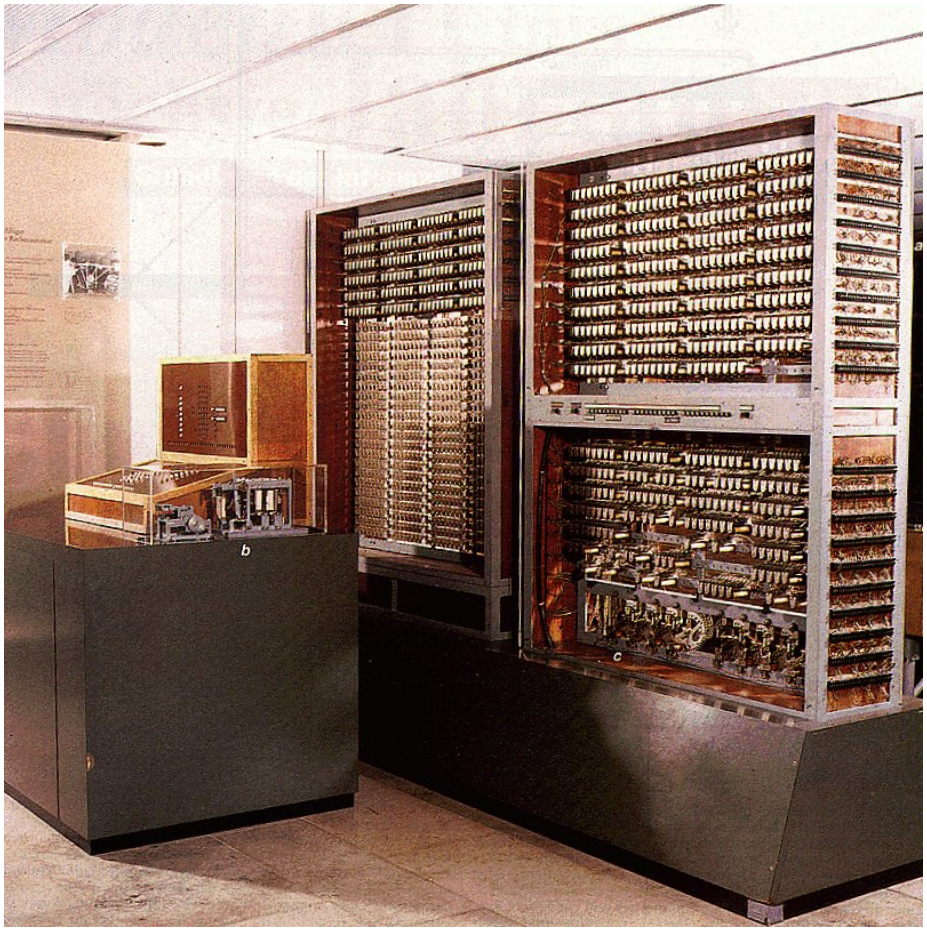
Zuse Z1

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



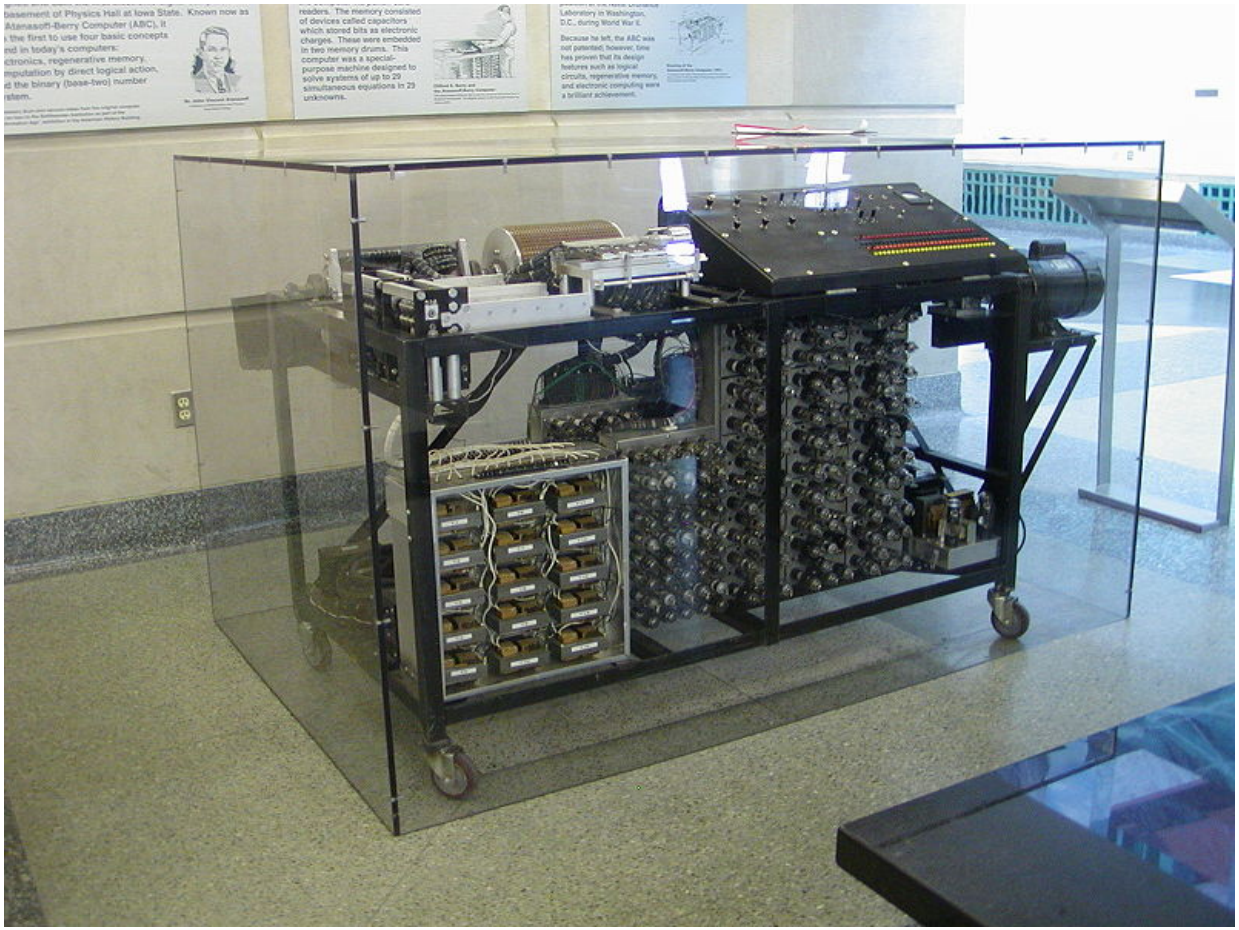
Zuse Z3

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



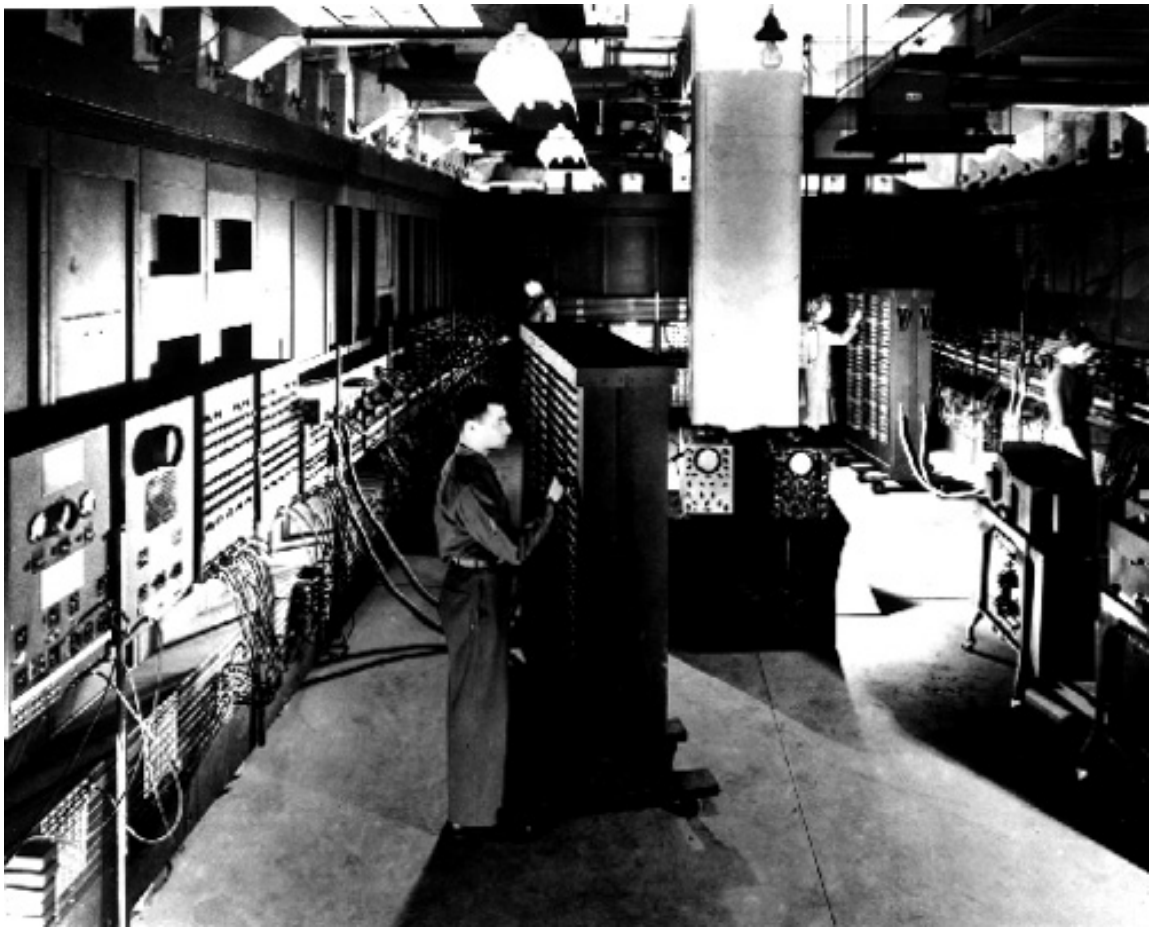
Atanasoff-Berry Computer (ABC)

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

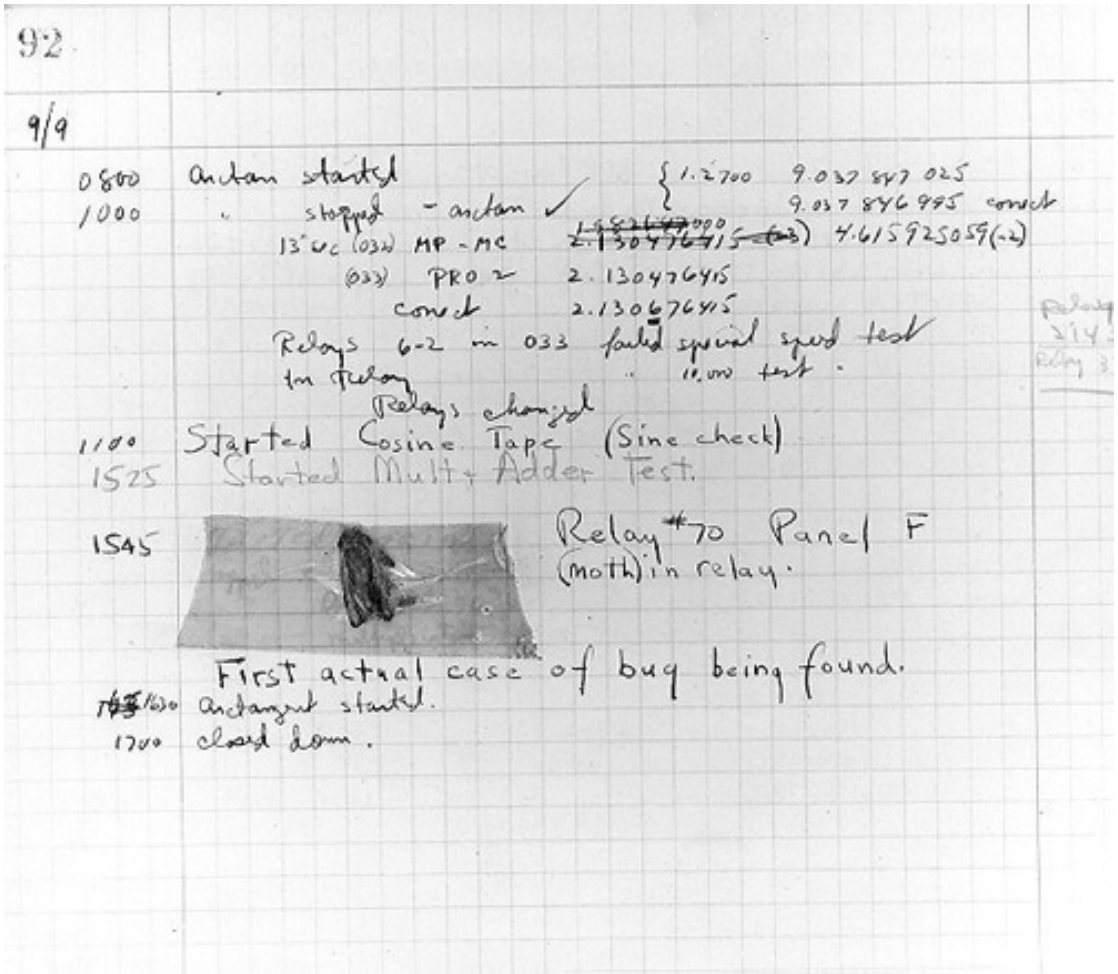


ENIAC — Electronic Numerical Integrator and Computer

Mauchly & Eckert, 1946: Röhren, Steckbrett-Programm

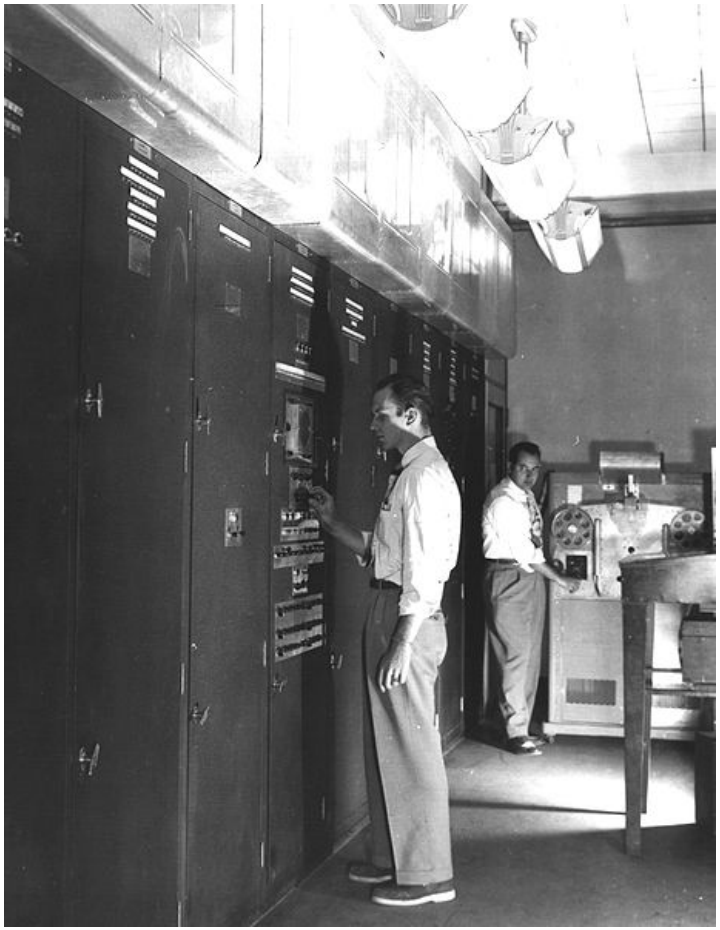


First computer bug



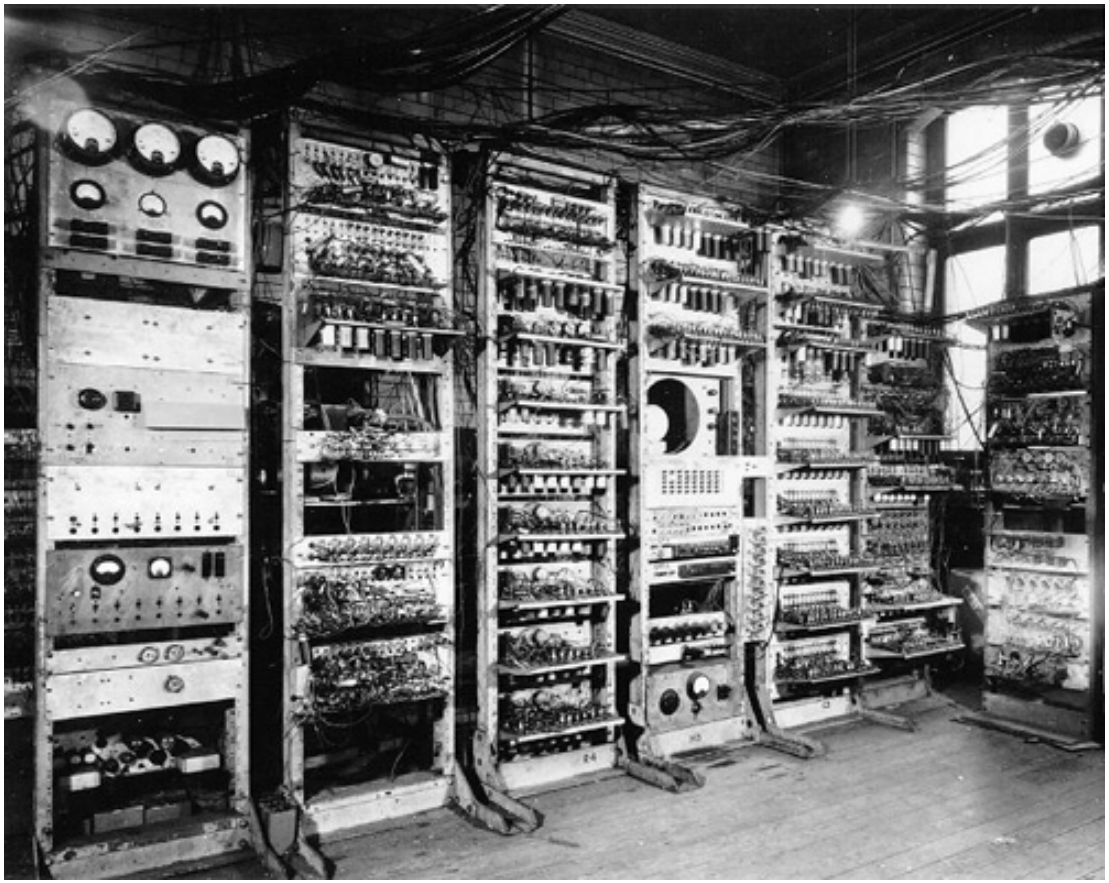
EDVAC

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



Manchester Mark-1

Williams & Kilburn, 1949: Trommelspeicher, Indexregister



Manchester EDSAC

Wilkes 1951: Mikroprogrammierung, Unterprogramme, speicherprogrammiert

Meilensteine

1952: IBM 701	(Pipeline)
1964: IBM S/360	(Rechnerfamilie, software-kompatibel)
1971: Intel 4004	(4-bit Mikroprozessor)
1972: Intel 8008	(8-bit Mikrocomputer-System)
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:

- Programmierung in elementarer Maschinensprache (ISA level)
- Hardware in Röhrentechnik (device logic level)
  - aber: 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.

- Maschine wird nicht gut ausgelastet
- Anwender mit lästigen Details überfordert

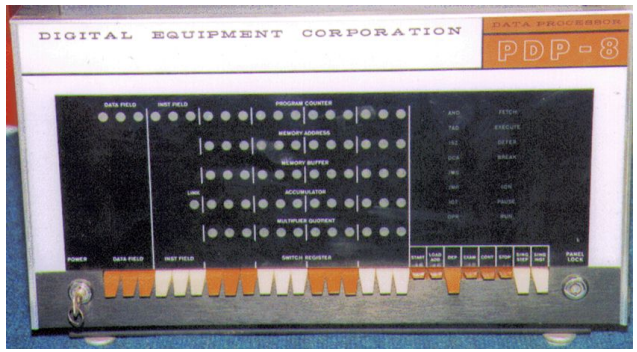
Einführung von **Betriebssystemen**

- „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 $
  - Graphikdisplay: erste Computerspiele
- Nachfolger PDP-8: 16.000 $
  - erstes Bussystem
  - 50.000 Stück verkauft



Festplatten

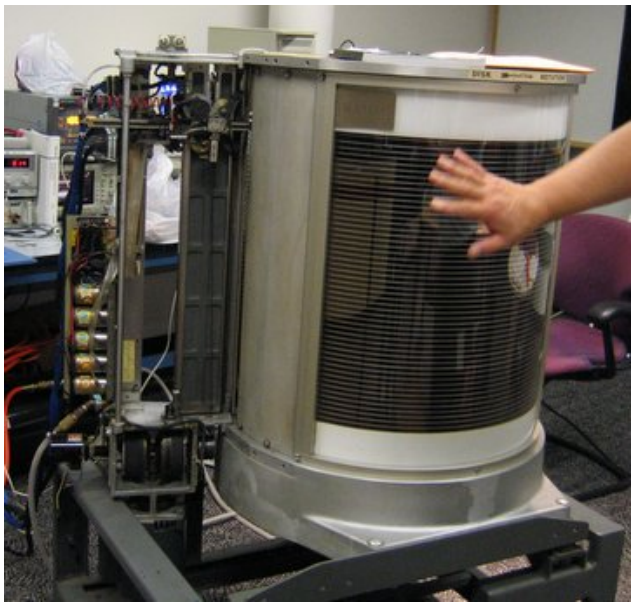
Massenspeicher bei frühen Computern:

- 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 (Noyce, Kilby)
- Dutzende. . . Hunderte. . . Tausende Transistoren auf einem Chip

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

Property	Model 30	Model 40	Model 50	Model 65
Relative performance	1	3.5	10	21
Cycle time (nsec)	1000	625	500	250
Maximum memory (KB)	64	256	256	512
Bytes fetched per cycle	1	2	4	16
Maximum number of data channels	3	3	4	6

Figure 1-7. The initial offering of the IBM 360 product line.

vierte Generation: VLSI

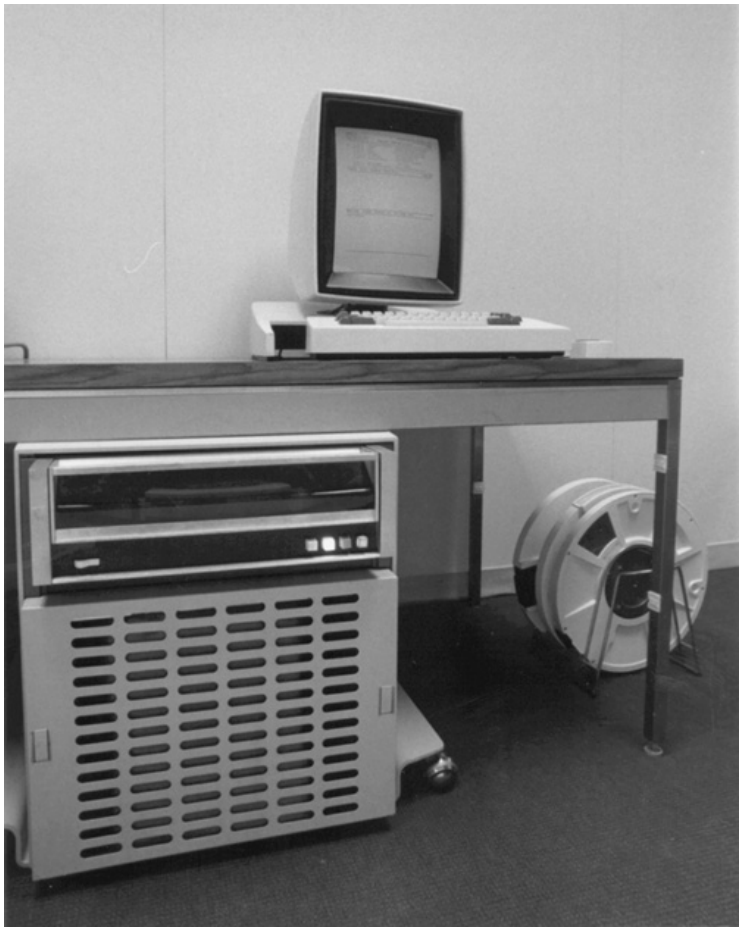
- VLSI = *very large scale integration*
- ab 10.000+ Transistoren pro Chip

- 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

Type	Price ($)	Example application
Disposable computer	1	Greeting cards
Embedded computer	10	Watches, cars, appliances
Game computer	100	Home video games
Personal computer	1K	Desktop or portable computer
Server	10K	Network server
Collection of Workstations	100K	Departmental minisupercomputer
Mainframe	1M	Batch data processing in a bank
Supercomputer	10M	Long range weather prediction

Figure 1-9. The current spectrum of computers available. The prices should be taken with a grain (or better yet, a metric ton) of salt.

Moore's Law

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

- bei gleicher Funktion kleinere und billigere Chips
- bei gleicher Größe leistungsfähigere Chips

Moore's Law: (Gordon Moore, Mitgründer von Intel, 1965)

- Speicherkapazität von ICs vervierfacht sich alle drei Jahre
- schnelles exponentielles Wachstum (!)
- klares Kostenoptimum bei hoher Integrationsdichte
- Vorhersage gilt immer noch, trifft auch auf Prozessoren zu
- „ITRS-Roadmap“ bis über Jahr 2020 hinaus

Moore's Law: Transistoren pro Speicherchip

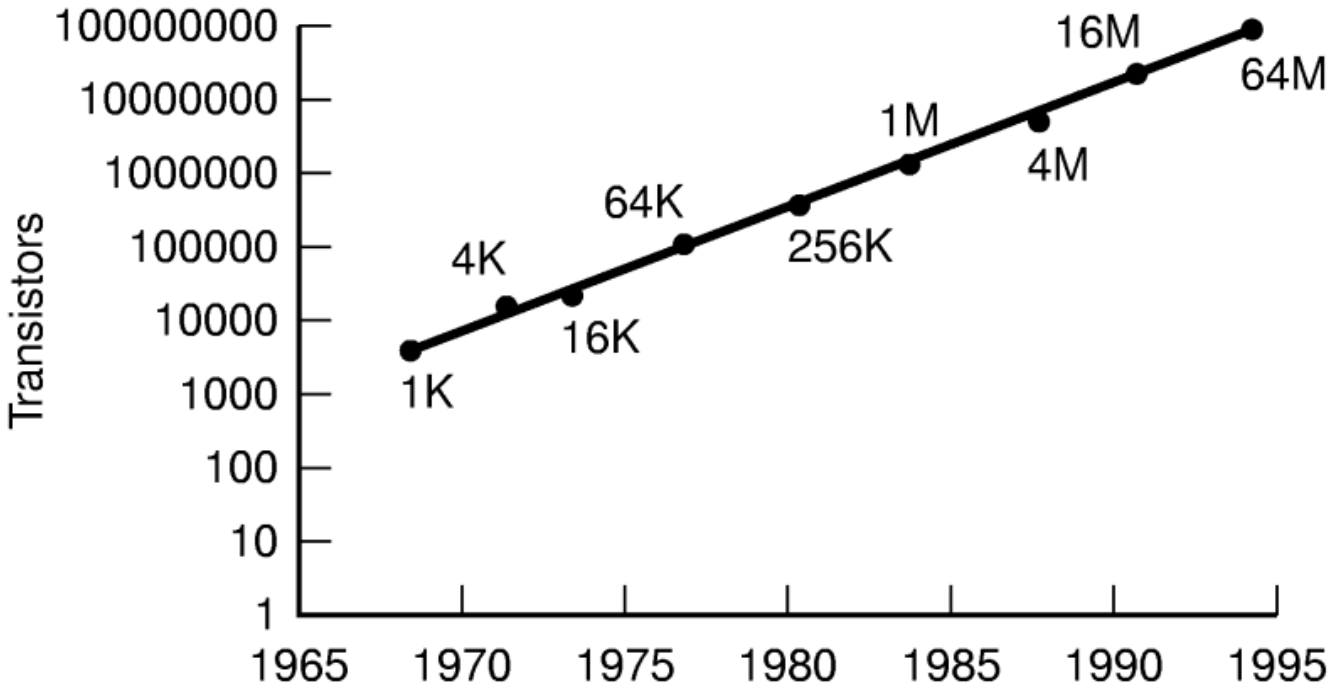
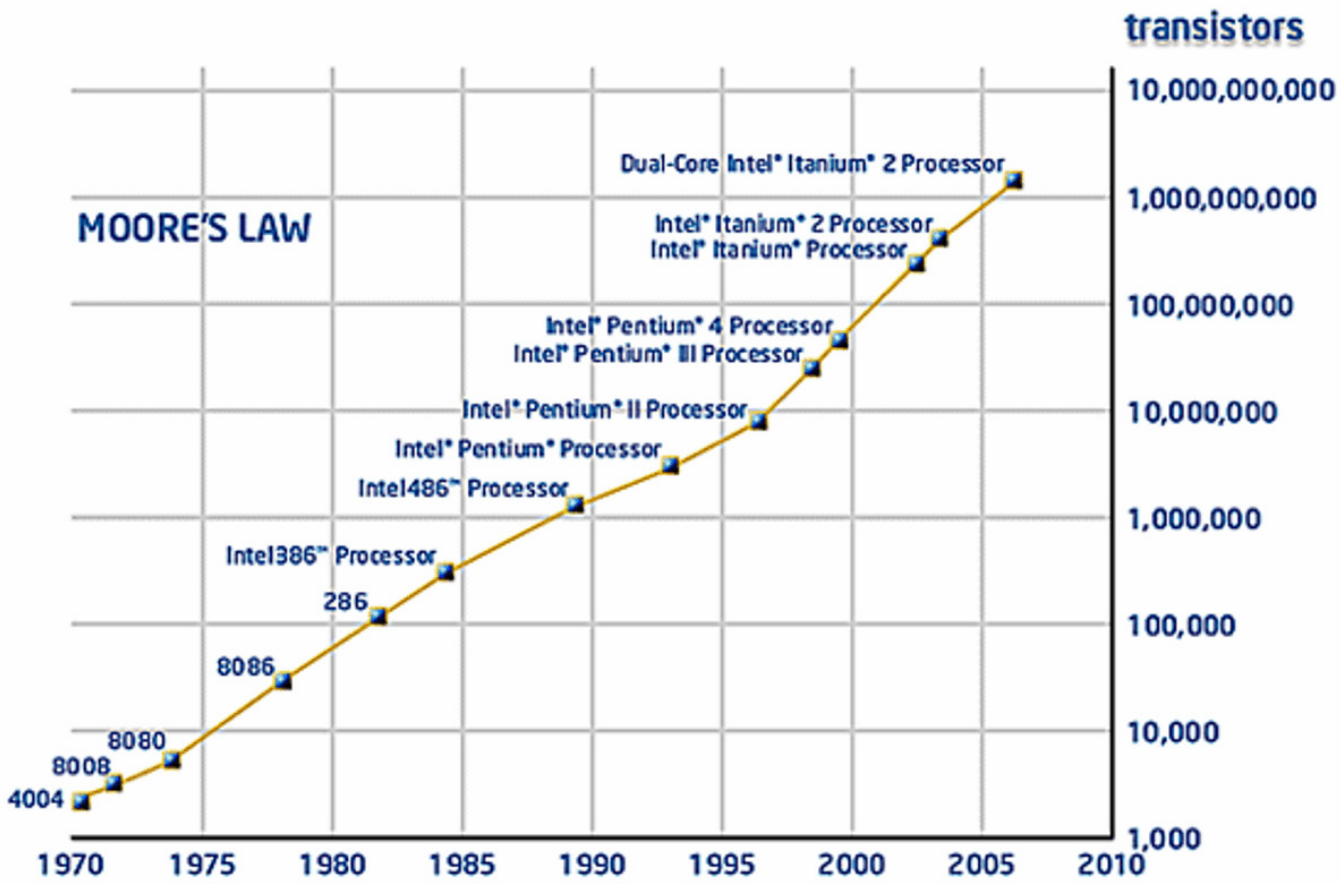


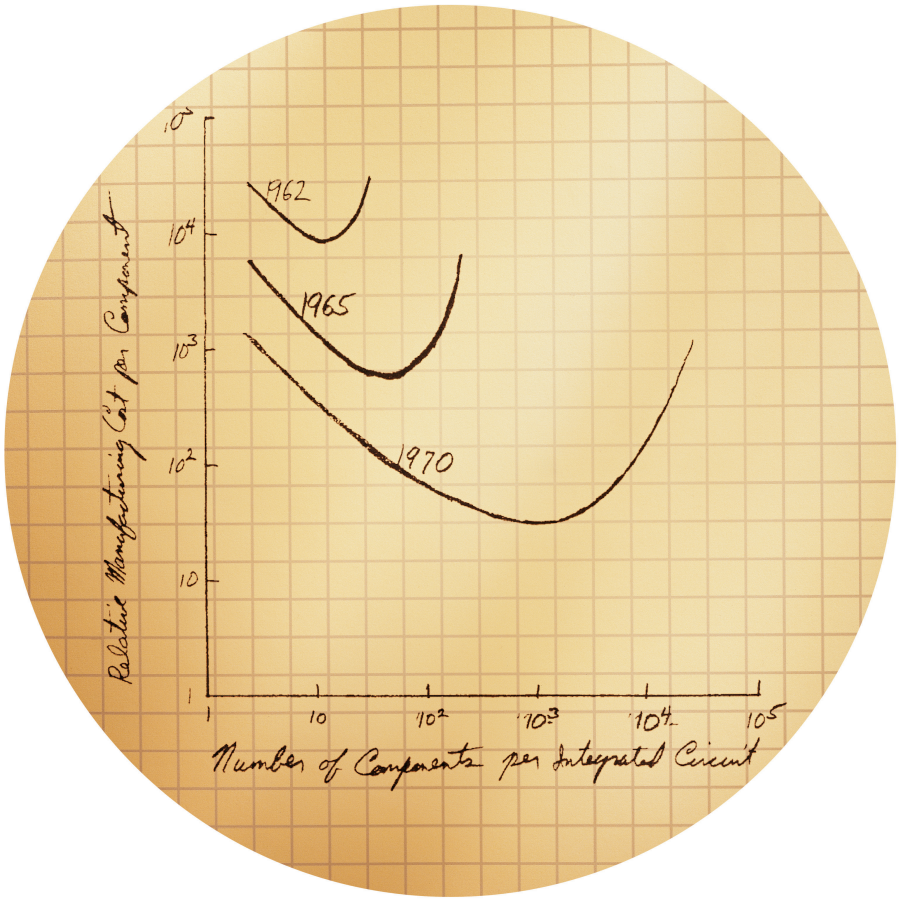
Figure 1-8. Moore's law predicts a 60 percent annual increase in the number of transistors that can be put on a chip. The data points given in this figure are memory sizes, in bits.

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



(Intel)

Moore's Law: Kosten pro Komponente



(Originalskizze von G. Moore, www.intel.com)

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/2012/06/100 de.wikipedia.org/wiki/Supercomputer

Jahr	Rechner	Linpack [GFlop]	Zahl der Prozessoren
1993	Fujitsu NWT	124	140
1994	Intel Paragon XP/S MP	281	6 768
1996	Hitachi CP-PACS	368	2 048
1997	Intel ASCI Red (200 MHz Pentium Pro)	1 338	9 152
1998	ASCI Blue-Pacific (IBM SP 640E)	2 144	5 808
1999	ASCI Intel Red (Pentium II Xeon)	2 379	9 632
2000	ASCI White, IBM (SP Power 3)	4 903	7 424
2002	Earth Simulator, NEC	35 610	5 104
2006	JUBL	45 600	16 384
2008	IBM Roadrunner (Opteron 2c + IBM Cell)	1 105 000	124 400
2009	Cray XK6 Jaguar (Opteron 16c + NVIDIA)	1 941 000	298 592
2012	Super MUC, Leibnitz Rechenz. (Xeon 8/10c)	2 897 000	147 456
2012	Sequoia (Power BQC, 16 cores)	16 324 750	1 572 864

Moore's Law: Aktuelle Trends

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

- 4 GByte Hauptspeicher (und mehr) wird bezahlbar
- Übergang von 32-bit auf 64-bit Adressierung

- Integration mehrerer CPUs auf einem Chip (Dual-/Quad-Core)
- zunehmende Integration von Peripheriegeräten
- ab 2011: CPU plus leistungsfähiger Graphikchip
- **SoC**: „System on a chip“

SoC: System on a chip

Gesamtes System auf einem Chip integriert:

- ein oder mehrere Prozessoren
- Befehls- und Daten-Caches für die Prozessoren
- Hauptspeicher (dieser evtl. auch extern)
- weitere Speicher für Medien/Netzwerkoperationen
- Peripherieblöcke nach Kundenwunsch konfiguriert:
  - serielle und parallele Schnittstellen, I/O-Pins
  - Displayansteuerung
  - USB, Firewire, SATA
  - Netzwerk kabelgebunden (Ethernet)
  - Funkschnittstellen: WLAN, Bluetooth, GSM/UMTS
  - Feldbusse: I2C, CAN, ...
- Handy, Medien-/DVD-Player, WLAN-Router, usw.

SoC: Beispiel: Bluetooth-Controller – Chiplayout

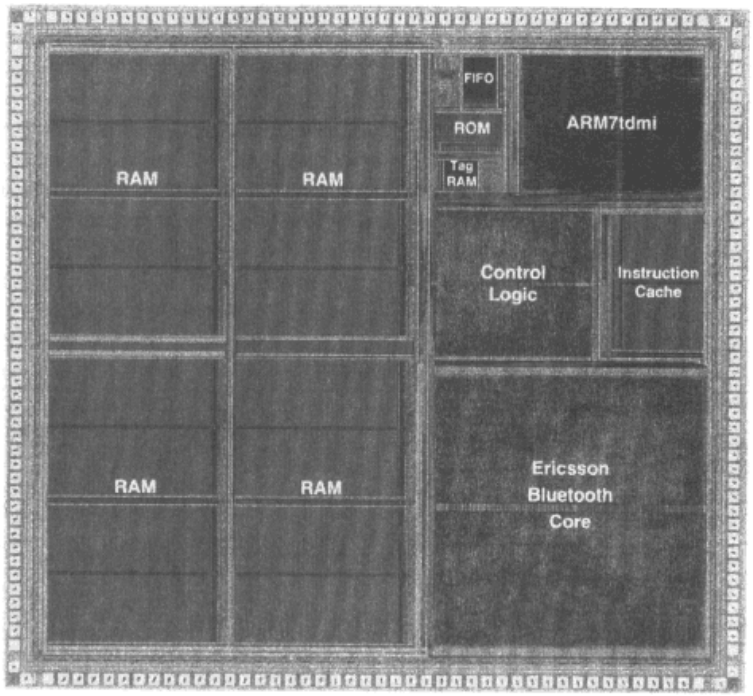


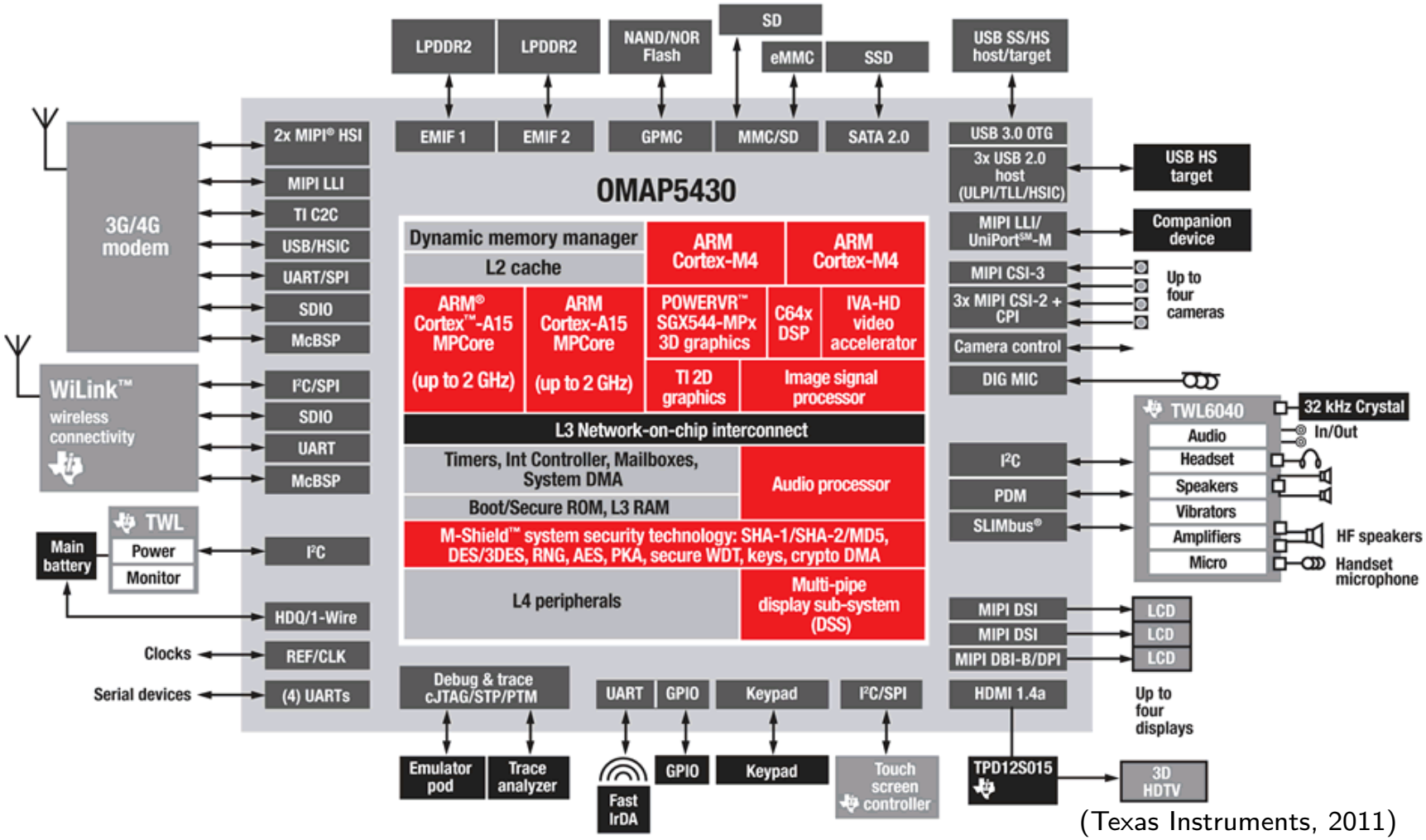
Figure 13.9 Bluetooth Baseband Controller die photograph.

Table 13.1 Bluetooth characteristics.

Process	0.25 µm	**Transistors**	4,300,000	**MIPS**	12
Metal layers	3	**Die area**	20 mm^2	**Power**	75 mW
Vdd (typical)	2.5 V	**Clock**	0–13 MHz	**MIPS/W**	160

(S. Furber, *ARM System-on-Chip Architecture*, 2000)

SoC: Beispiel: OMAP 5430



(Texas Instruments, 2011)

Smart Dust

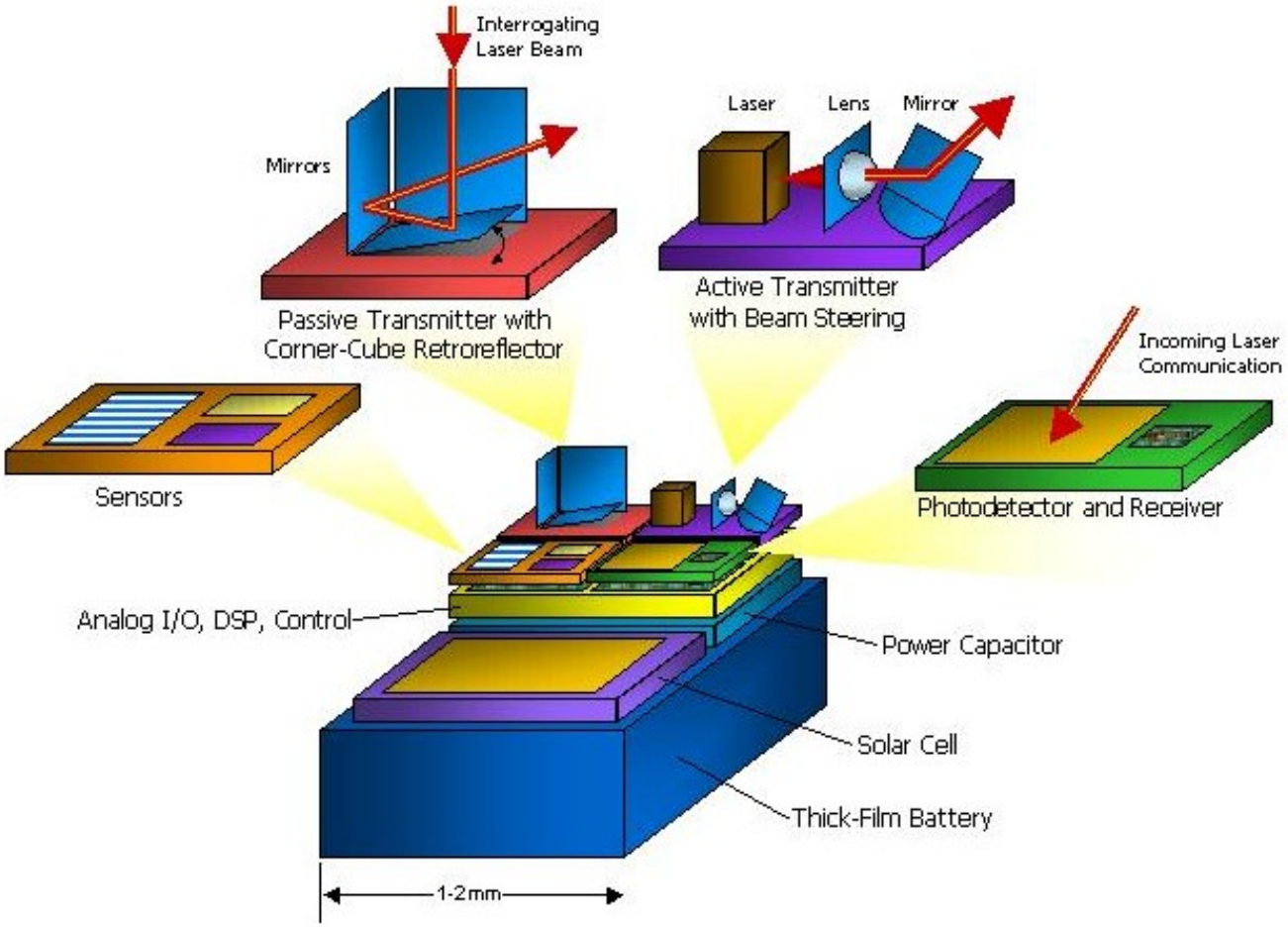
Wie klein kann man Computer bauen?

- Berkeley Projekt: **Smart Dust** (2002-2006)
- Integration kompletter Rechensysteme auf $1\,\mathrm{mm}^3$
  - 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

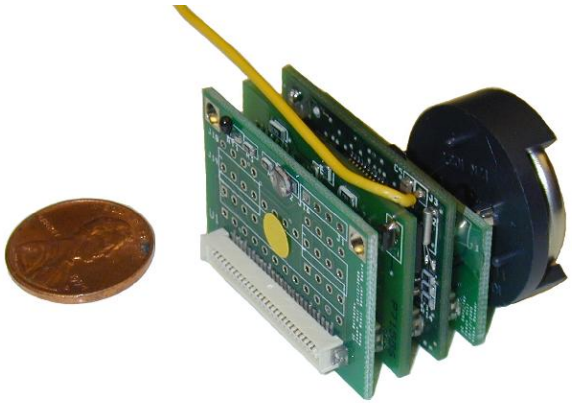
- Massenfertigung? Tausende autonome Mikrorechner
- „Ausstreuen“ in der Umgebung
- vielfältige Anwendungen

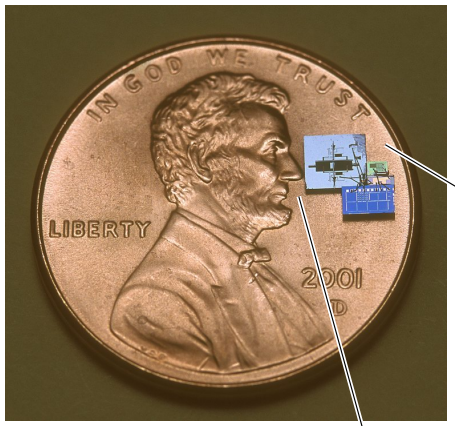
(Berkeley Sensor & Actuator Center, eecs.berkeley.edu)

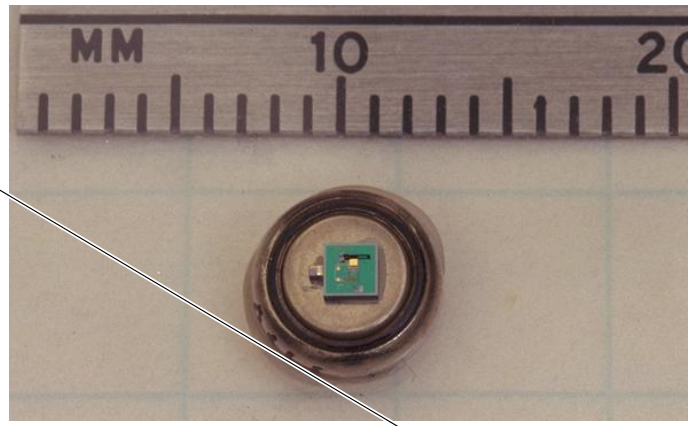
Smart Dust: Konzept

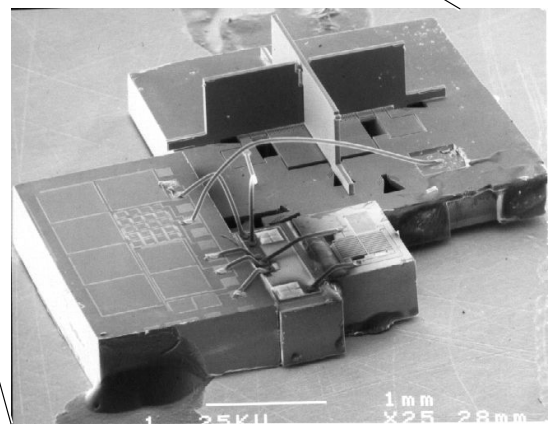


Smart Dust: Prototypen







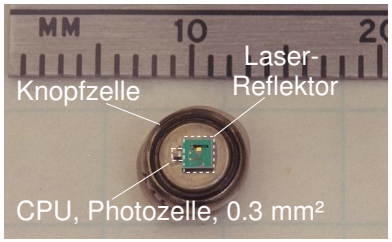


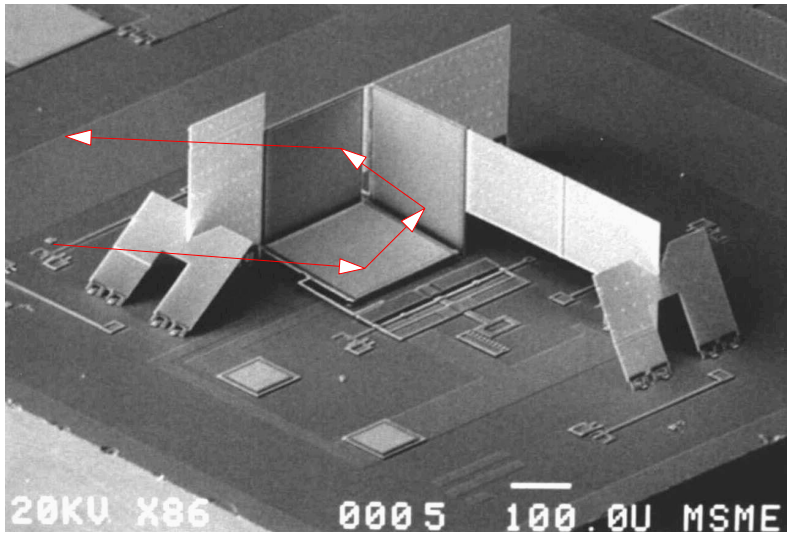
diverse Prototypen:

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

- MEMS-"CCR" für opt. Kommunikation

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



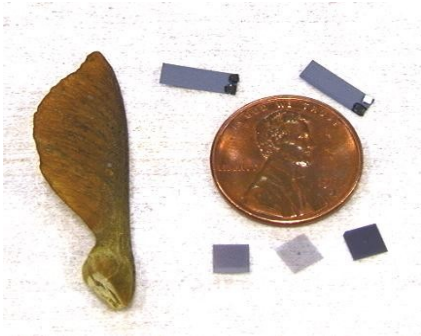


- CCR: seitlich zwei starre Spiegel, Gold auf Silizium
- untere Spiegelfläche beweglich (elektrostatisch, ca. 30 V)
- gezielte Modulation von eingestrahltem Laserlicht
- Reichweiten $> 100\,\mathrm{m}$ demonstriert

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

Smart Dust: Energieverbrauch

Miniatur-Solarzellen
Wirkungsgrad ca. 3%
26 µW/mm² in vollem Sonnenlicht



Batterien:	~ 1J/mm³		
Kondensatoren:	~10 mJ/mm³		
Solarzellen:	~ 0.1 mW/mm²	~ 1J/mm²/day	(außen,Sonne)
	~ 10 µW/mm²	~ 10mJ/mm²/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: eine DRAM-Speicherzelle speichert derzeit etwa 100.000 Elektronen. Durch die Verkleinerung werden es mit jeder neuen Technologiestufe weniger.
- Offensichtlich ist die Grenze spätestens erreicht, wenn nur noch ein einziges Elektron gespeichert würde.
- Ab diesem Zeitpunkt gibt es bessere Performance nur noch durch bessere Algorithmen / Architekturen

- Annahme: 50 % Wachstum pro Jahr, $a^b = \exp(b \cdot \ln a)$
- Elektronen pro Speicherzelle: $100000/(1.5^{x/\text{Jahre}}) \geq 1$.
- $x = \ln(100.000)/\ln(1.5) \approx 28$ Jahre

Roadmap: ITRS

International Technology Roadmap for Semiconductors

- 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

(www.itrs.net/Links/2008ITRS/Home2008.htm)

Moore's Law: Schöpferische Pause

Beispiel für die Auswirkung von Moore's Law. Angenommen die Lösung einer Rechenaufgabe dauert bei gegenwärtiger Rechenleistung 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.

Beweis: 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 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)