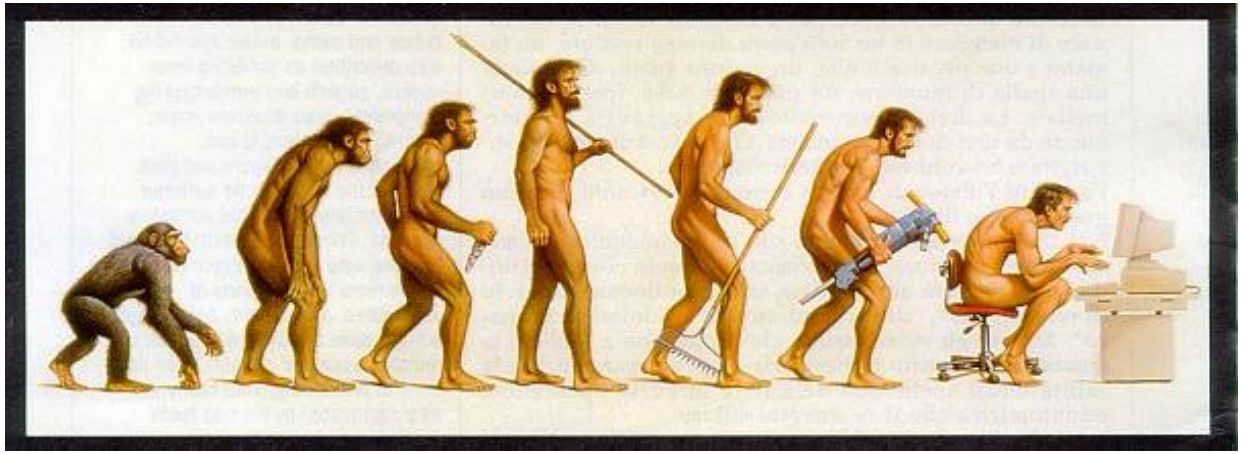
Grundlagen der Programmierung

VL 01: Grundbegriffe

Prof. Dr. Samuel Kounev,
M.Sc. Norbert Schmitt



Danksagung

- Vorlesungsmaterialien von Prof. Dr. Detlef Seese wurden als Basis verwendet
- Unterstützung bei der technischen und inhaltlichen Gestaltung des Vorlesungsmaterials leisteten:

  Jóakim v. Kistowski

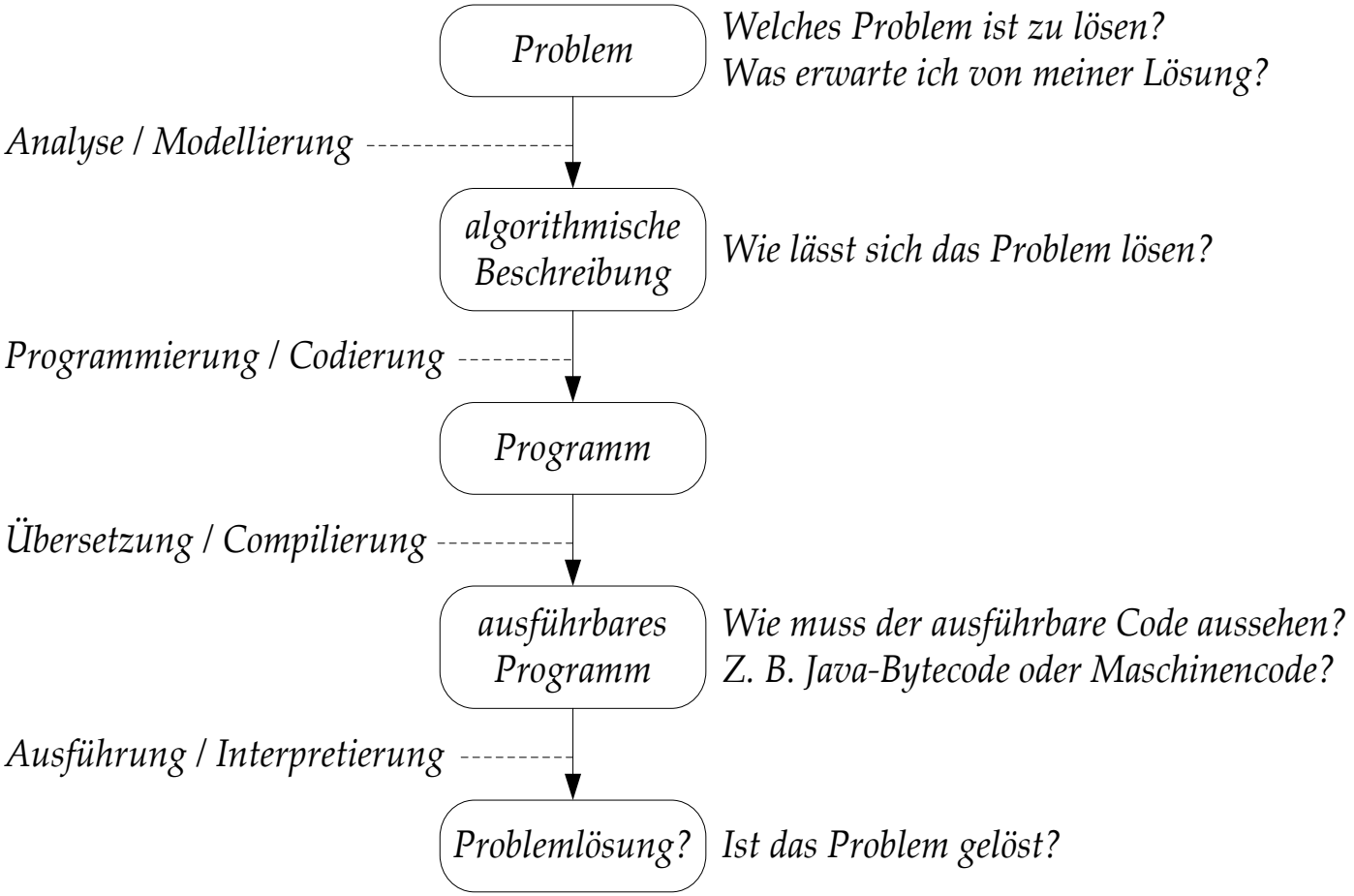
  Dietmar Ratz, Joachim Melcher, Roland Küstermann, Jana Weiner, Hagen Buchwald, Matthes Elstermann, Oliver Schöll, Niklas Kühl, Tobias Diederich

Inhalt

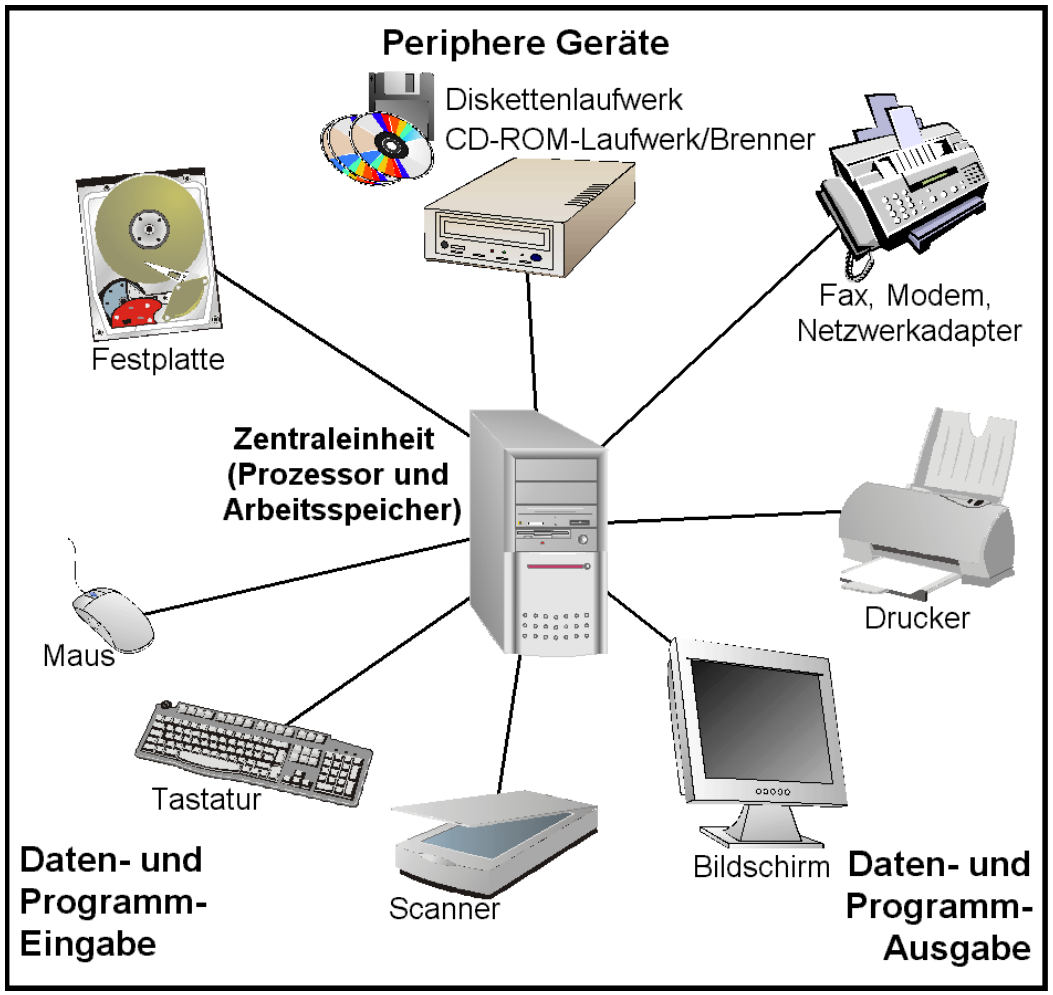
- Was heißt Programmieren?
- Aufbau von Rechenanlagen: Hardware, Software
- Compiler, Interpreter, Zwischensprachen
- Historischer Überblick über Programmiersprachen
- Grundbegriffe der Programmierung
- Anlage
  - Historischer Überblick über die Entwicklung von Computersystemen

Was heißt Programmieren?

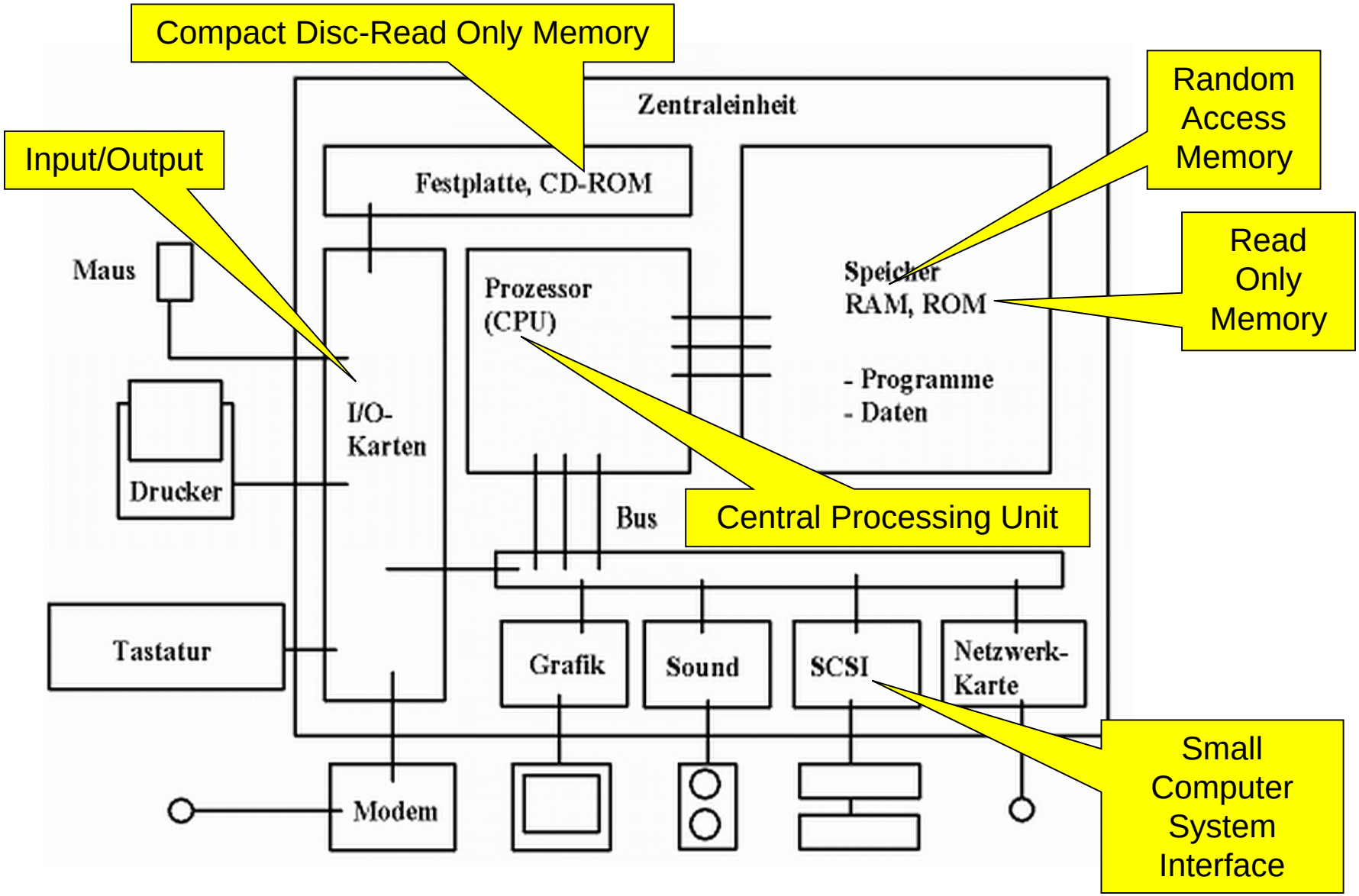
- Umsetzen eines gegebenen Algorithmus in ein lauffähiges Computer-Programm



Umgebung einer "Rechenanlage"

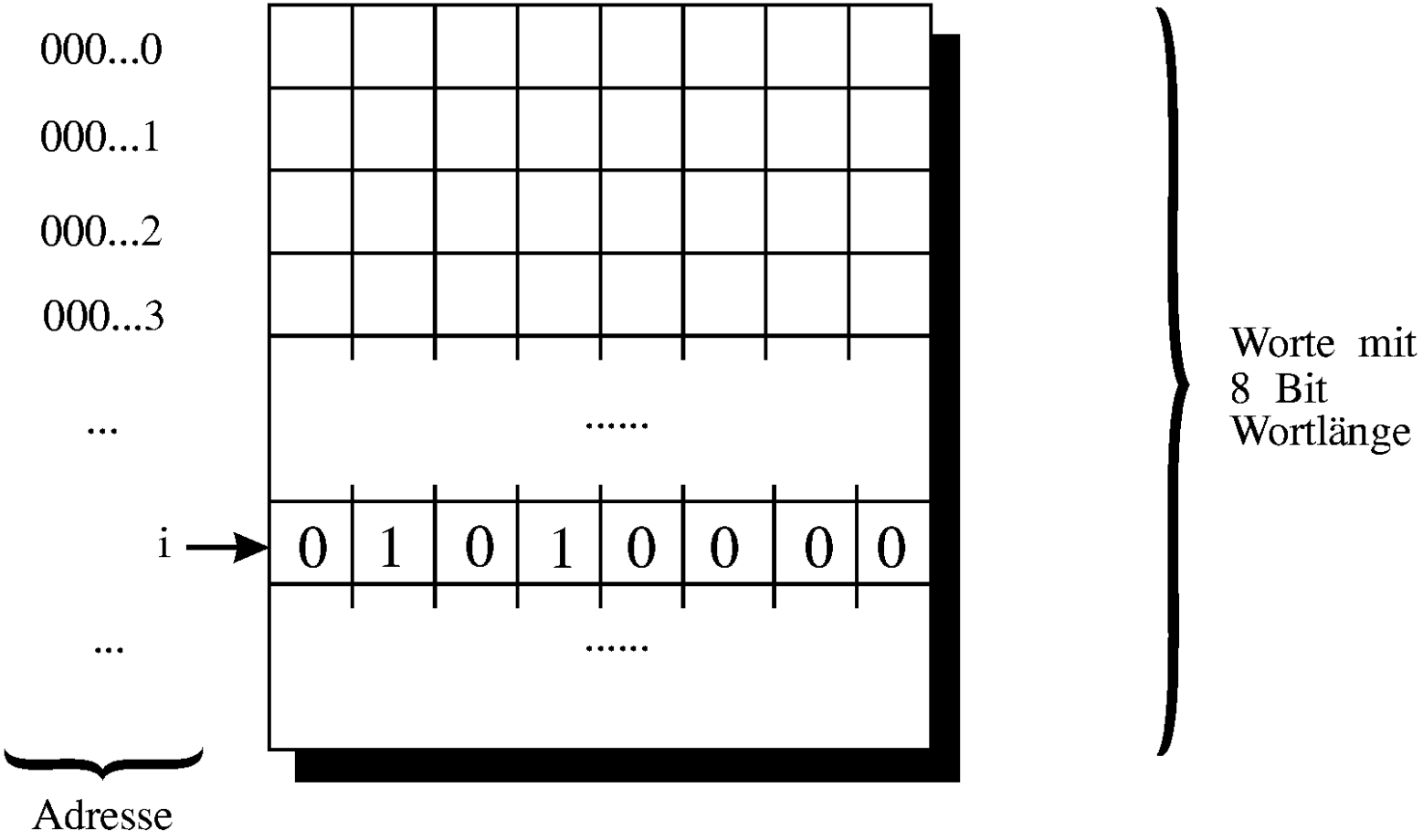


Aufbau von Rechenanlagen



Aufbau von Rechenanlagen (2)

- Einfaches Schema des Arbeitsspeichers



Aufbau von Rechenanlagen (3)

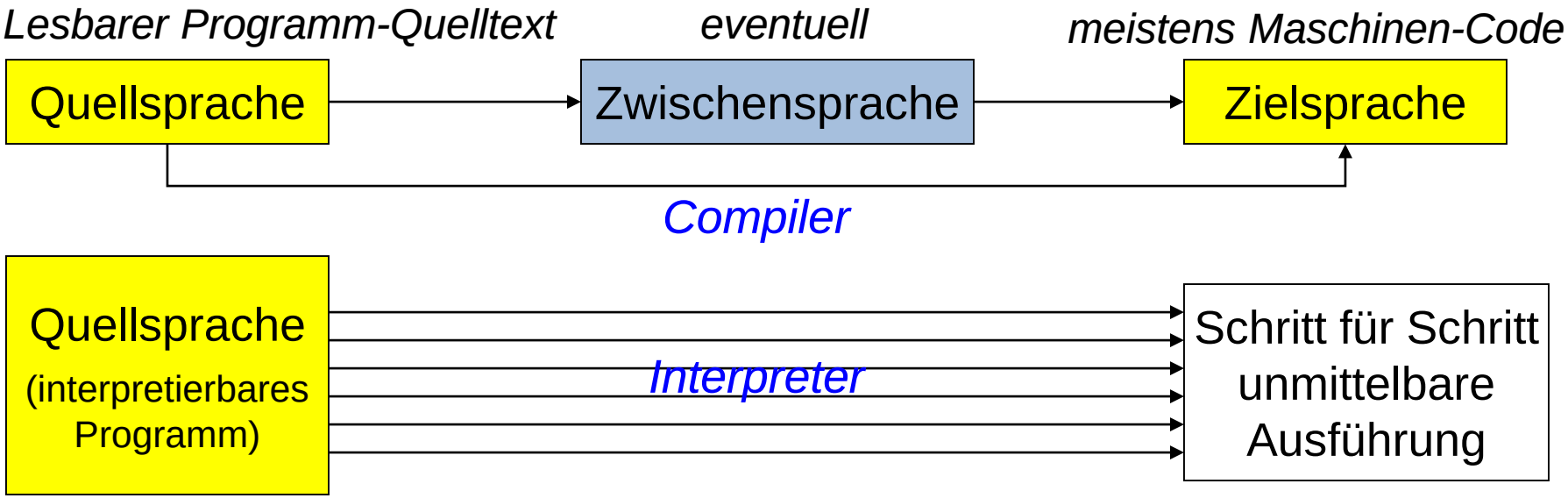
- Datenspeicherung im Arbeitsspeicher
  - einzelne Daten können sich (abhängig von ihrem Typ) aus mehreren Speicherworten zusammensetzen
- Verbindung zu Programmiersprachen
  - Adressen im Speicher erhalten symbolische Namen (z.B. Variablen)
  - Inhalte mehrerer Speicherworte werden kombiniert und ergeben bestimmte Werte (z.B. Werte von Variablen)

Software

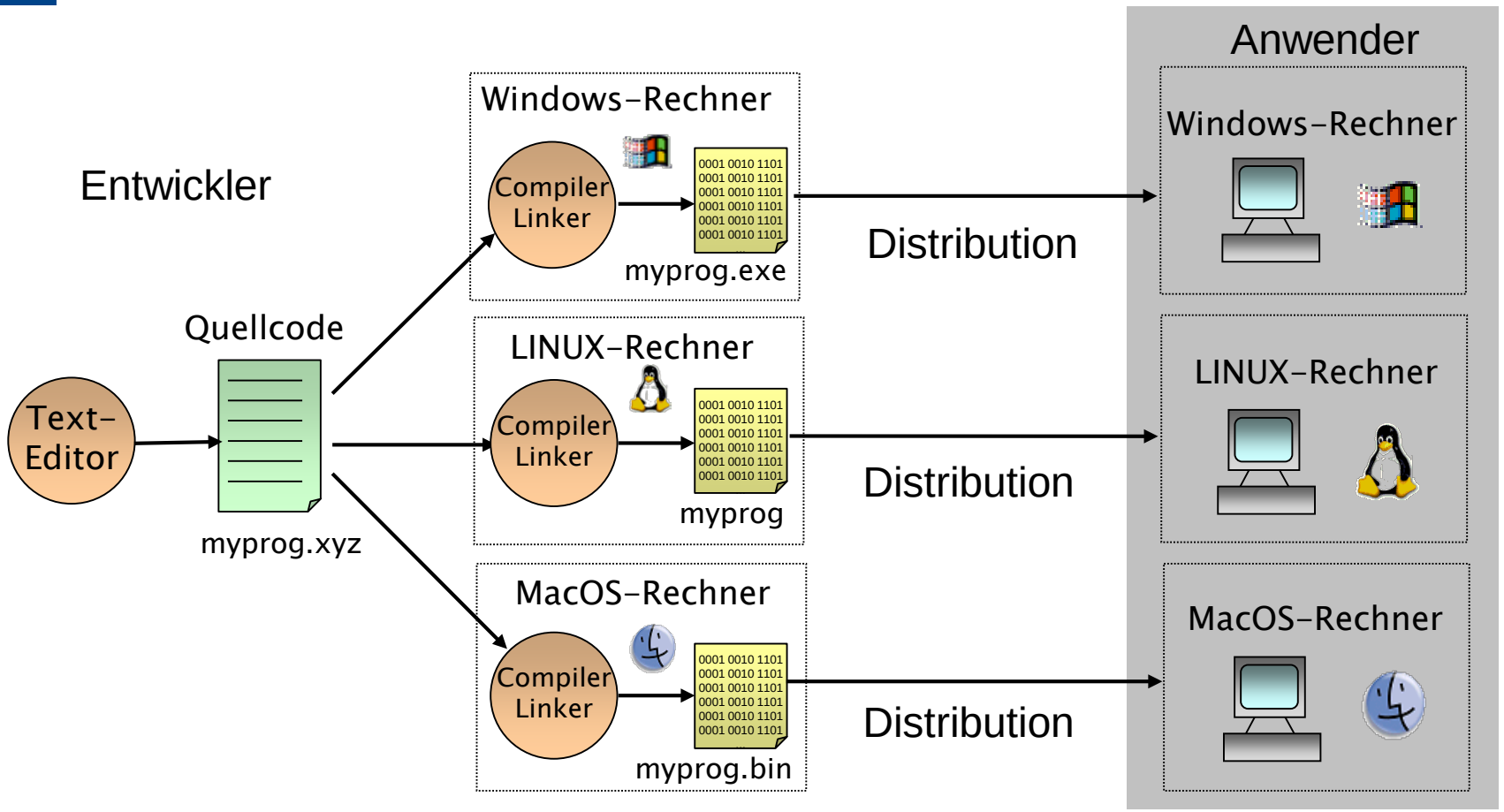
Klassifizierung

- Betriebssysteme (Windows, UNIX, LINUX, MacOS, iOS, Android)
- Dienstprogramme (Editor, Datei-Manager, Mailer, Browser)
- Übersetzer (Assembler, Compiler, Interpreter)
- Anwendungsprogramme (mit Compiler oder Interpreter erzeugt)

Compiler und Interpreter

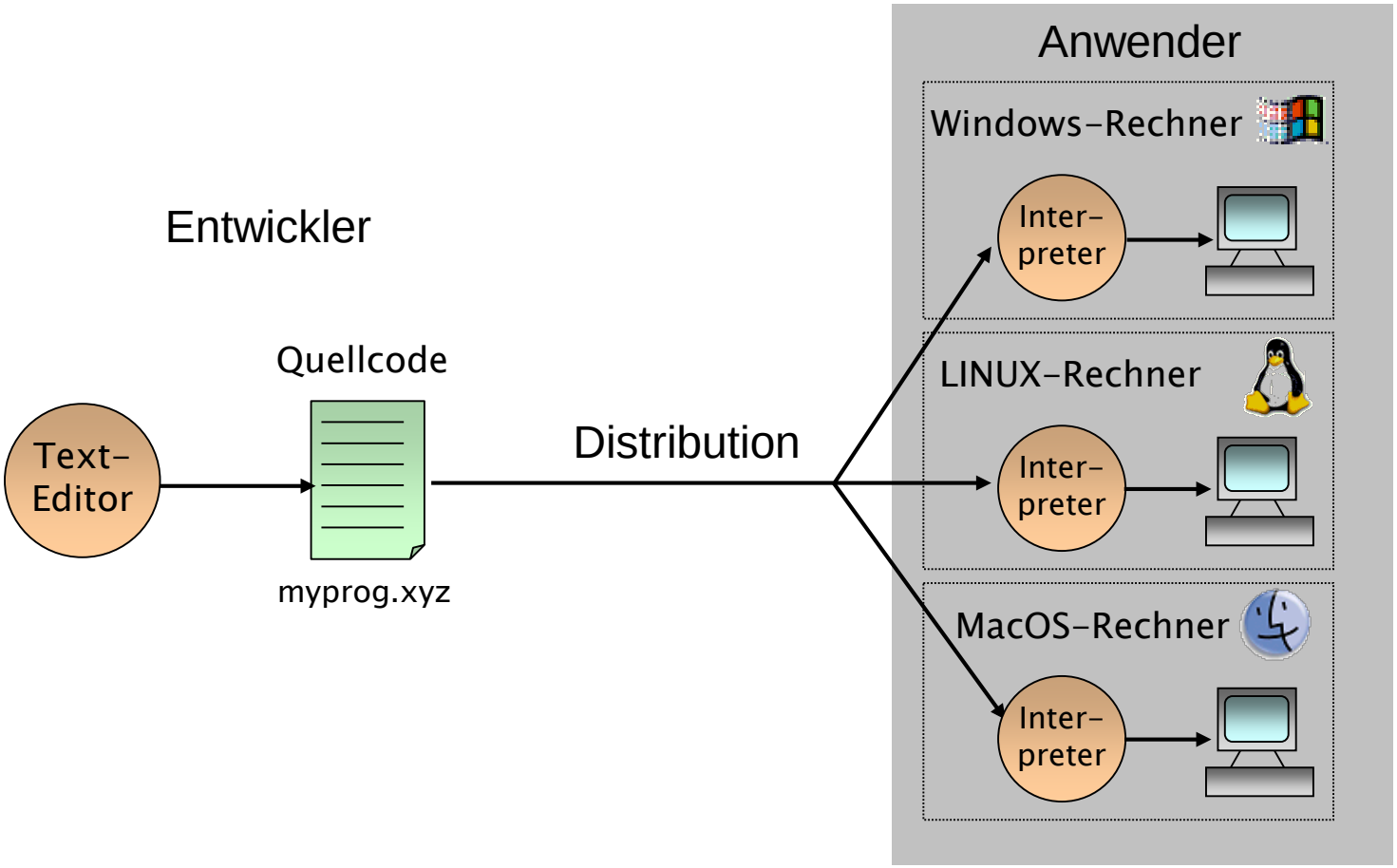


Compiler



- **Vorteile:** Optimale Nutzung der Prozessoreigenschaften, hohe Abarbeitungsgeschwindigkeit, Quellcode geschützt
- **Nachteile:** Programme spezialisiert auf Zielrechner, zusätzlicher Aufwand für Anpassungen (z.B. grafische Oberflächen), verschiedene Compiler notwendig

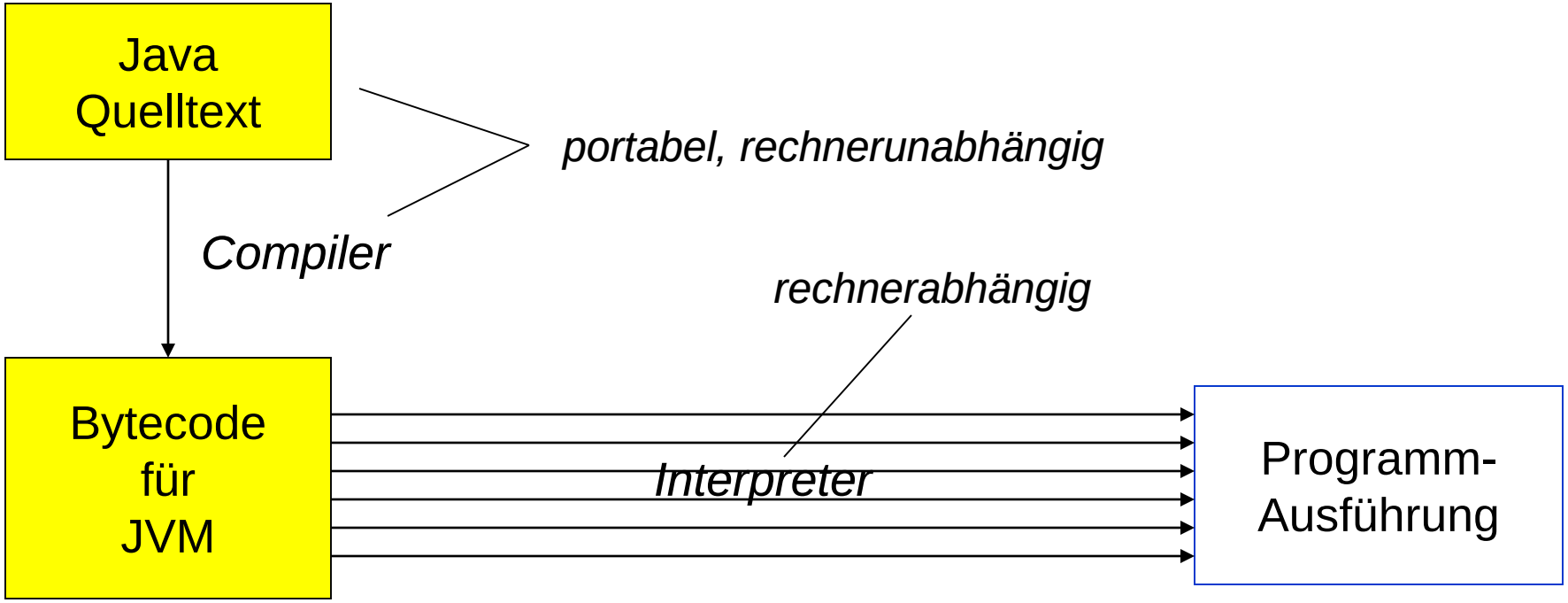
Interpreter



- **Vorteile:** Quellcode direkt ausführbar, falls Interpreter verfügbar
- **Nachteile:** langsame Ausführung, verschiedene Interpreter notwendig, Quellcode lesbar

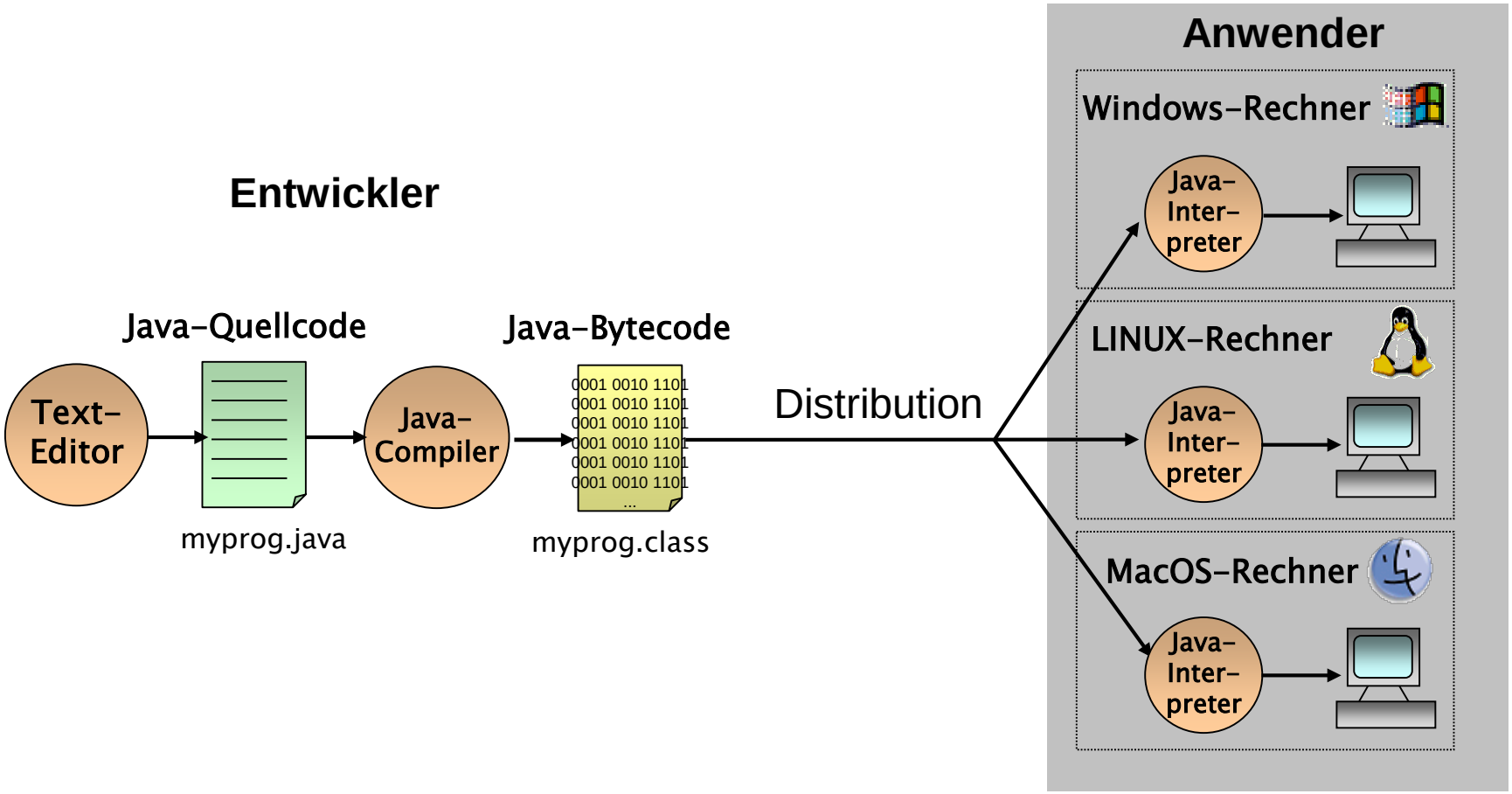
Java

- Kombination von Compiler und Interpreter für Code einer virtuellen Maschine **(Java Virtual Machine - JVM**)



Java: Compiler + Interpreter

Java



- **Vorteile:** nur ein Compiler, Sprache plattformunabhängig, Quellcode teilweise geschützt
- **Nachteile:** langsame Ausführung, verschiedene Interpreter notwendig

Ein klein wenig Geschichte

```
01101001
00001001
10100101
```

Maschinensprache

```
LOAD R1 23
MOVE R7 R2
ADD
```

Assembler

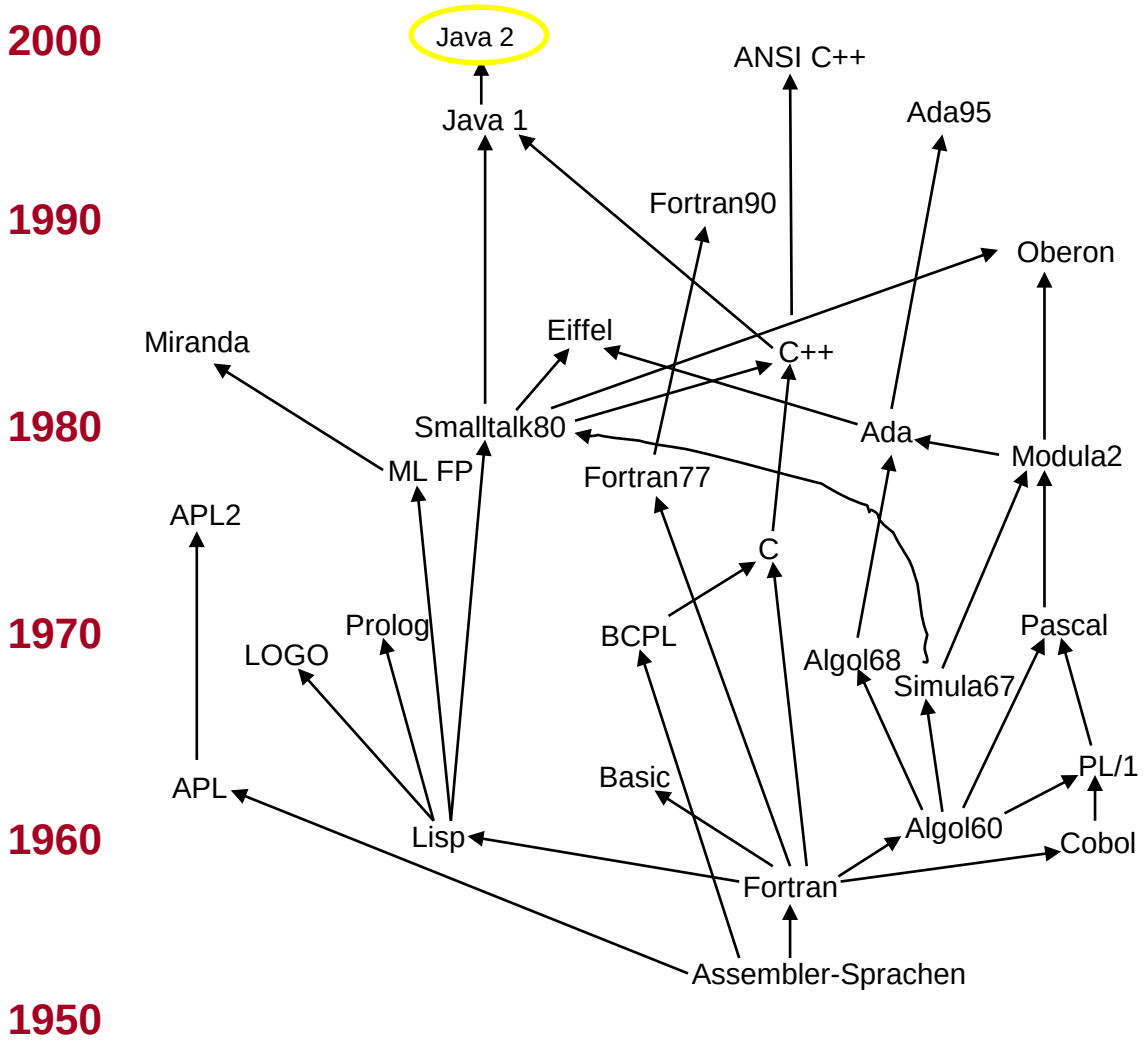
```
10 PRINT "Hello"
20 SET A = 7
30
```

Frühe höhere Programmiersprache (BASIC)

```
public class Hallo
{
  public static void main(String[] args)
  { System.out.println("Hallo"); }
}
```

Moderne höhere Programmiersprache (Java)

Ein klein wenig Geschichte (2)



Ein klein wenig Geschichte (3)

- **FORTRAN** (FORmula TRANslation, 1954)
  - J. W. Backus, IBM. FORTRAN 77, FORTRAN 90, FORTRAN 95
- **COBOL** (COmmon Business Oriented Language, ca. 1958)
  - für kommerzielle Datenverarbeitung
- **BASIC** (Beginners All-purpose Symbolic Instruction Code, 1960)
  - für Kleinrechner, vereinfachtes FORTRAN
- **ALGOL 60** (ALGOrithmic Language, 1960)
  - Vorläufer für viele moderne Programmiersprachen, Dinosaurier
- **PASCAL** (nach Blaise Pascal (1623-1662) benannt, ca. 1970)
  - Jensen und Wirth, standardisiert, viele Dialekte
- **ADA** (ca. 1977)
  - US Verteidigungsministerium, sehr umfangreich
- **Modula 2**, Oberon 2 (1978 und später)
  - Wirth, Weiterentwicklung von Pascal, Modulkonzept
- **C, C++** (ca. 1979 bzw. 1986)
  - immer verbreiteter, systemnah bzw. objektorientiert, teils kryptisch.
- **Prolog, LISP** (ca. 1970, ca. 1960)
  - Sprachen der künstlichen Intelligenz (KI)
- **Java** (1996, Oak 1993)
  - James Gosling, Sun Microsystems Inc., Internet-Programmiersprache

Algorithmen und ihre Darstellung

- Beispiel: Euklidischer Algorithmus zur Berechnung des größten gemeinsamen Teilers zweier natürlicher Zahlen *a* und *b*
  - Beispiel als Text:

```
solange b ≠ 0
        falls a > b
                a := a - b
        andernfalls
                b := b - a
gib a zurück
```

Beispielausführung:

a = 3 b = 4

b:= b-a = 4-3 = 1

a=3 b = 1

a = a-b = 3-1 = 2

a = 2 b = 1

a = a-b = 2–1 = 1

a = 1 b = 1

b = b-a = 1-1 = 0

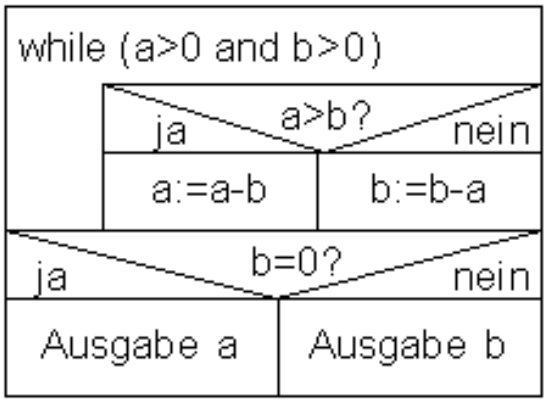
a = 1 b = 0

Algorithmen und ihre Darstellung

- Beispiel: Euklidischer Algorithmus Berechnung des größten gemeinsamen Teilers zweier natürlicher Zahlen *a* und *b*
  - Beispiel als Text:

```
solange b ≠ 0
        falls a > b
                a := a - b
        andernfalls
                b := b - a
gib a zurück
```

  - Beispiel als Struktogramm (Nassi-Shneiderman-Diagramm):



Problem

a = 0 b = 1

b:= b-a = 1-0 = 1

a=0 b = 1

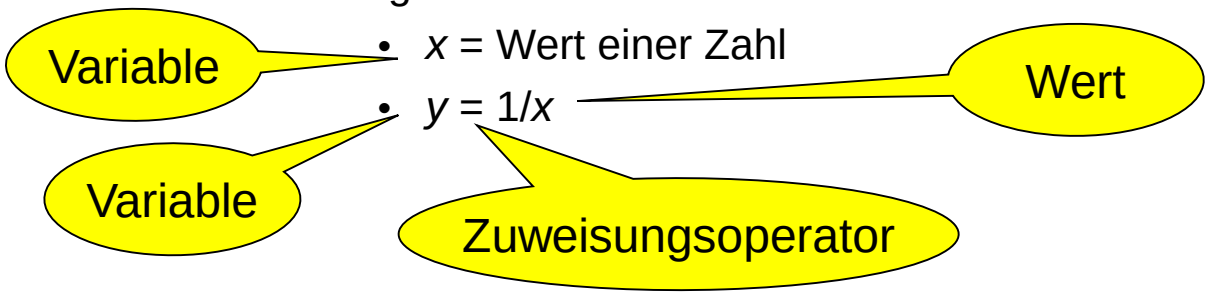
b = b-a = 1-0 = 1

unendliche Schleife

Der im Struktogramm angegebene Algorithmus löst dieses Problem.

Grundbegriffe der Programmierung

- **Variable (variable)**
  - "Platzhalter" für einen Wert eines bestimmten Typs
    - mit einer Adresse im Hauptspeicher assoziiert
  - Beispiel:
    - Algorithmus A: "Kehrwert von 4"
      - dividiere 1 durch 4
    - Algorithmus B: "Kehrwert einer Zahl"
      - setze *x* auf den Wert der Zahl
      - dividiere 1 durch *x*
- **Zuweisung (Wertzuweisung) (assignment)**
  - weist einer Variable einen Wert zu
  - Beispiel:
    - Algorithmus B': "Kehrwert einer Zahl"
      - *x* = Wert einer Zahl
      - *y* = 1/*x*



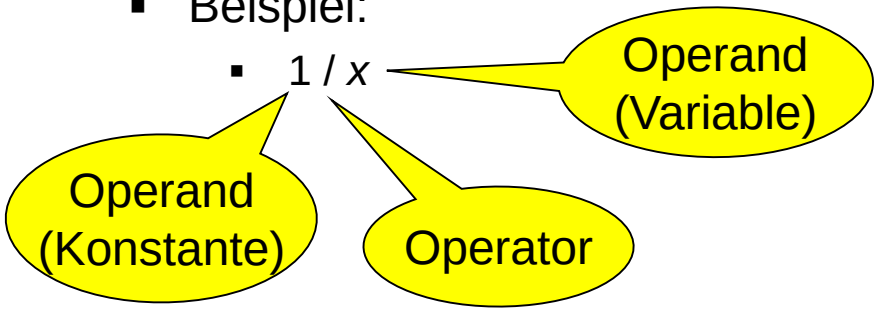
Grundbegriffe der Programmierung (2)

- **Datentyp (Typ) (data type)**
  - Bauplan für Daten (Werte von Variablen oder Konstanten), der festlegt,
    - wie die Darstellung der Werte im Speicher erfolgt,
    - welche Operationen für die Werte erlaubt sind,
    - welche Standardwerte (Default-Werte) festgelegt sind.
  - Beispiele (Variablen im Speicher – hier nur Idee (ungenau)):

Name	Adresse	Wert (Inhalt)	Typ
x	001...0100	0 1 0 0 0 1 0 1 0 0 0 1 0 1 0 0	ganze Zahl (z.B. **`short`**, 16 Bit)
y	010...0010	0 1 0 1 0 1 0 1 1 1 0 0 0 1 0 1 1 1 1 0 0 1 0 1 0 1 0 0 0 1 0 1	Gleitkomma-Zahl (z.B. **`float`**, 32 Bit)
z	011...1101	0 1 0 0 0 1 0 1 1 0 0 1 0 0 1 0	Zeichen (Typ **`char`**, 16 Bit)

Grundbegriffe der Programmierung (3)

- **Ausdruck (expression)**
  - Kombination von Operanden und Operatoren als "Vorschrift" zur Berechnung eines Werts
  - liefert immer einen Wert (Ergebniswert) ab
  - Beispiel:
    - 1 / *x*



- **Anweisung (statement)**
  - Kombination von Ausdrücken und Methoden als "Vorschrift" zur Ausführung einer Aktion
  - Beispiele:
    - *x* = 5 Wertzuweisung
    - *y* = 1 / *x* Wertzuweisung
    - print(*x*) Ausgabeanweisung (Methodenaufruf "Drucke *x*")

Grundbegriffe der Programmierung (4)

Java

- Deklaration:
  - `boolean a;`
  - `int b;`
- Zuweisung:
  - `a = true;`
  - `b = 5;`
- Anweisung (Ausdruck + Zuweisung):
  - `b = b + 2;`
- Deklaration + Anweisung:
  - `int c = b / 2;`

C/C++

- Deklaration:
  - `bool a; // nur in C++`
  - `int b;`
- Zuweisung:
  - `a = true; // nur in C++`
  - `b = 5;`
- Anweisung (Ausdruck + Zuweisung):
  - `b = b + 2;`
- Deklaration + Anweisung:
  - `int c = b / 2;`

In C gibt es generell keinen Typ „boolean“ für Wahrheitswerte.
Es wird meistens integer verwendet (0 = false, 1 = true).

Grundbegriffe der Programmierung (5)

- **Anweisungs-Sequenz (sequence of statements)**
  - mehrere Anweisungen, die nacheinander ausgeführt werden
- **Anweisungs-Block (block)**
  - logisch zusammengefasste Anweisungen bzw. Programmteile, die als *eine* Anweisung aufgefasst werden können
- **Bedingte Anweisung / Entscheidungsanweisung (conditional statement)**
  - Anweisung mit mehreren Alternativen
- **Wiederholungsanweisung / Schleife (loop)**
  - mehrfach ausgeführter Anweisungsteil (Block)

Grundbegriffe der Programmierung (6)

Java

- Schleife mit Block:

```
int a = 1;
while (a < 256) {
   a = a * 2
}
```

- Bedingte Anweisung:

```
if (a > 5) {
  //do something
}
else {
   //do something else
}
```

C/C++

- Schleife mit Block:

```
int a = 1;
while (a < 256) {
   a = a * 2
}
```

- Bedingte Anweisung:

```
if (a > 5) {
  //do something
}
else {
   //do something else
}
```

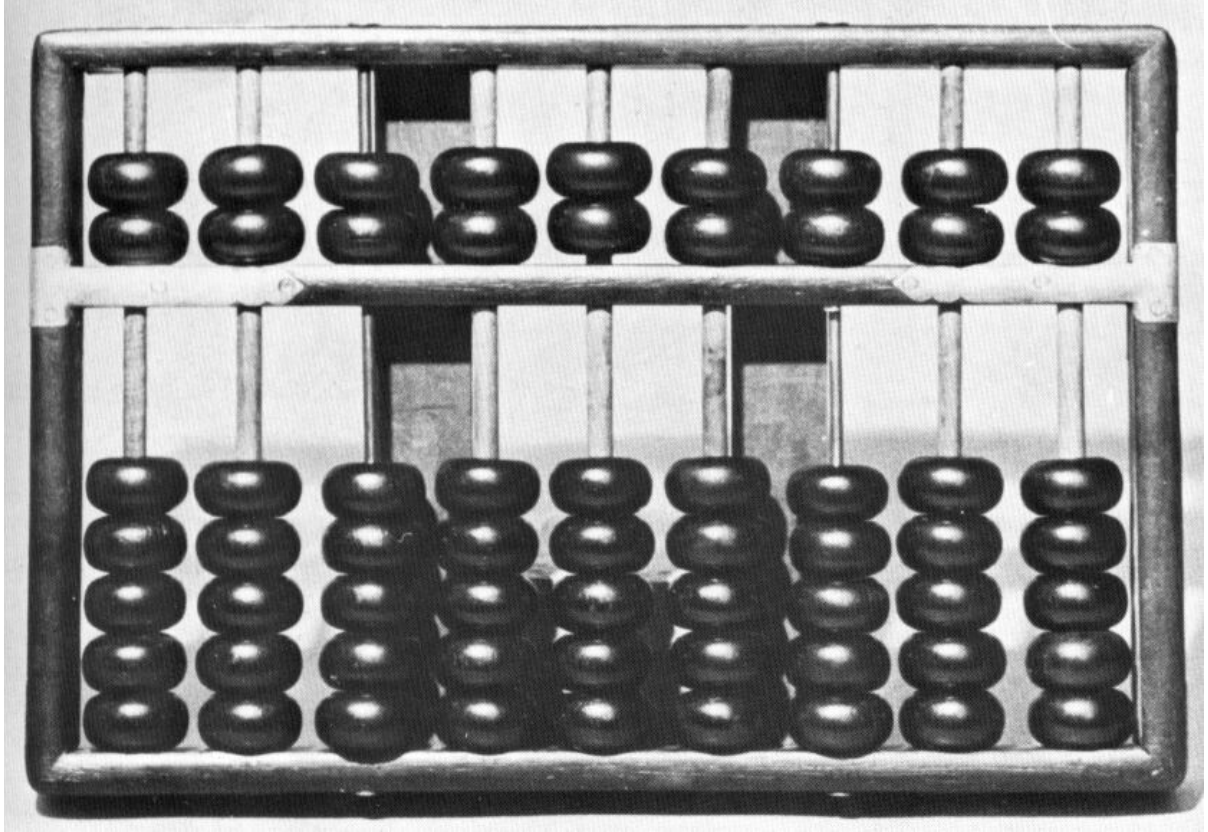
Fragen?


DIE NEUESTE RECHNERGENERATION: 16 FARBEN, AUF FESTPLATTE MONTIERT, INKLUSIVE MAUS...
IHR SOLLTET NOCH WARTEN – IN SECHS MONATEN KOSTET ER NUR NOCH DIE HÄLFTE...
STEIN

Anlagen

Entwicklung von Computersystemen

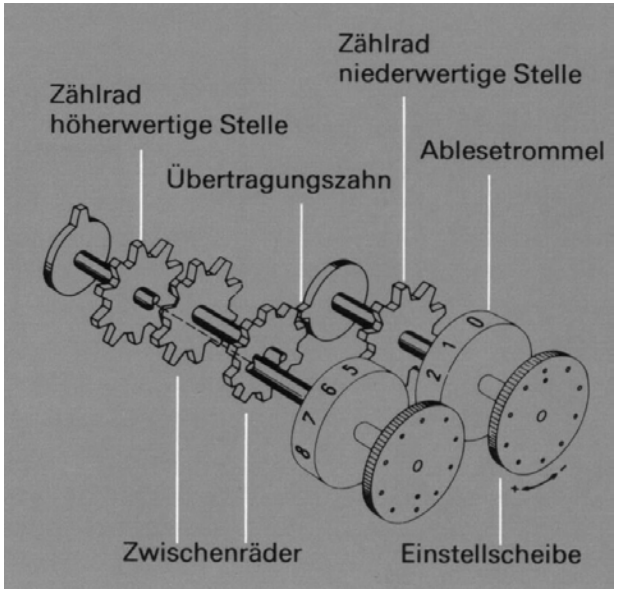
- Abakus (ca. 1100 v. Chr.)
  - In den 80er Jahren noch gebräuchlich im asiatischen Raum



Entwicklung von Computersystemen (2)

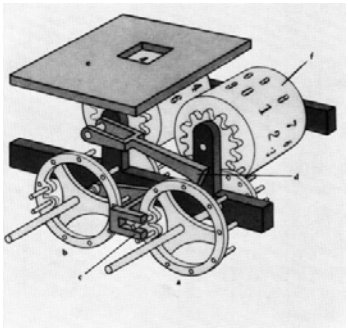
- Erste Rechenmaschine von Wilhelm Schickard (1623)
  - mit Zählrädern und Zehnerübertragung

Nachbau der Rechenmaschine
von W. Schickard
Quelle: http://de.wikipedia.org/wiki/Wilhelm_Schickard



Entwicklung von Computersystemen (3)

- Rechenmaschine von Blaise Pascal (1642)
  - konnte zunächst nur addieren, gebaut zur Erleichterung der Berechnung von Steuereinnahmen in der Normandie



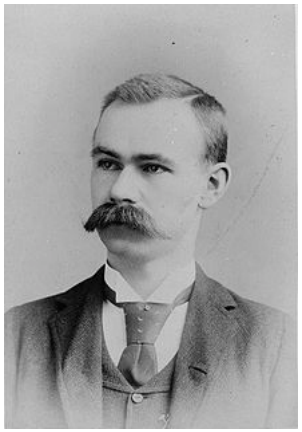
Quelle: http: //de.wikipedia.org/wiki/Blaise_Pascal

- Rechenmaschine von Gottfried Wilhelm Leibniz (1673)
  - mit vier Grundrechenarten

Quelle: http://de.wikipedia.org/wiki/Gottfried_Wilhelm_Leibniz

Entwicklung von Computersystemen (4)

- Elektrische Zähl- und Registriermaschine von Hermann Hollerith (1890)
  - US-Bergwerksingenieur, Sohn deutscher Auswanderer aus der Pfalz
  - erstmals maschinelle Auszählung bei Volkszählung 1890 möglich
  - erstmals Lochkarten zur Datenspeicherung

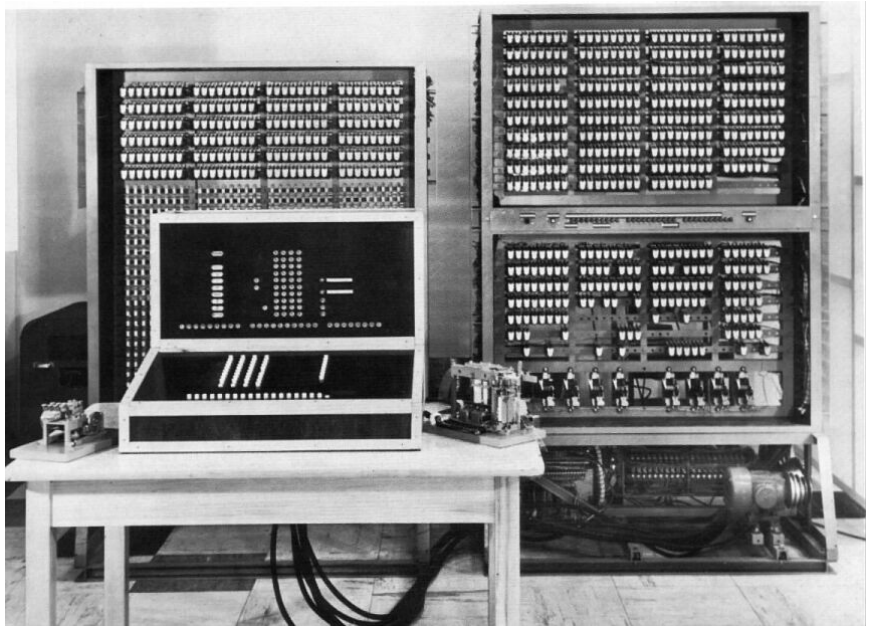


Quelle: http://de.wikipedia.org/wiki/Herman_Hollerith

Entwicklung von Computersystemen (5)

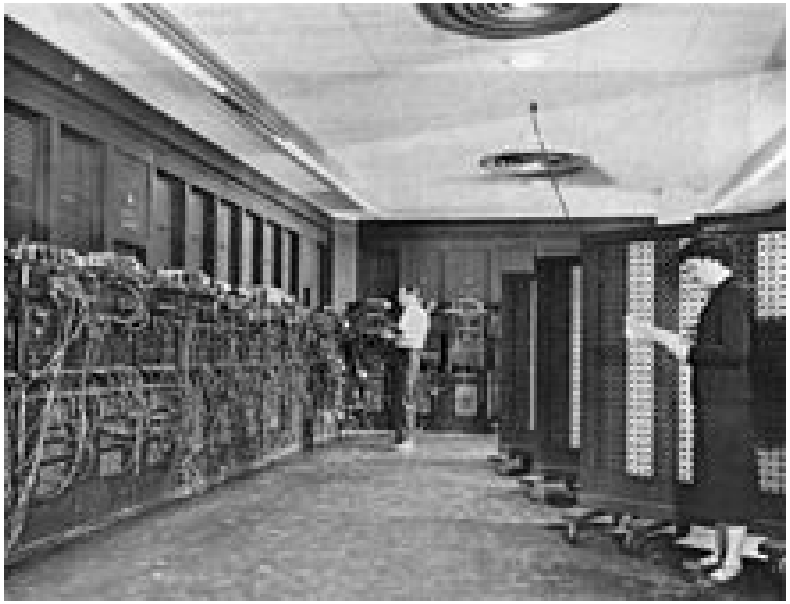
- Erstes programmgesteuertes Rechengerät Z3, Konrad Zuse (1941)
  - Speicherkapazität: 64 Zahlen mit 7 Dezimal- bzw. 22 Dual-Stellen
  - Lampenfeld (für 4 Dezimalen) und Tastatur
  - Lineares Programm (ohne Verzweigungen)
  - Kinofilm als Lochstreifen
  - 2 Additionen pro Sekunde

Quelle: http://www.konrad-zuse.de



Entwicklung von Computersystemen (6)

- Erster elektronischer Rechner der Welt ENIAC (1946)
  - 18 000 Röhren (auf 25% gefahren, dadurch nur 2 bis 3 Ausfälle pro Woche), 1500 Relais, Leistungsaufnahme 150 000 Watt
  - Gewicht 30 Tonnen, Stellfläche 135 qm
  - Schaltungen statt Programm



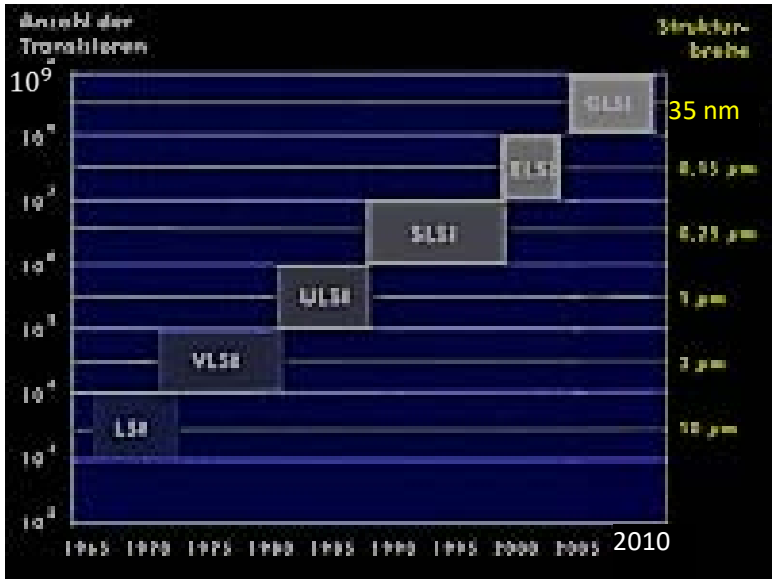
Quelle: http://de.wikipedia.org/wiki/ENIAC

John von Neumann entwickelte die grundlegende Architektur solcher Rechner

siehe Quelle
http://de.wikipedia.org/wiki/John_von_Neumann

Entwicklung von Computersystemen (7)

- 1946: J. v. Neumann schlägt Bau speicherprogr. Rechenanlagen vor
- ab 1950: Klassifizierung der technischen Entwicklung in Generationen
- 1. Generation (bis 1957)
  - ENIAC (1946), UNIVAC (1951), IBM 701 (1952)
- 2. Generation (1958 bis 1964)
  - volltransistorisiert
  - SSI = Small Scale Integration, 100 T
- 3. Generation (1965 bis 1970)
  - integr. Schaltkreise (IC)
  - MSI = Medium SI, 4000 T
- 4. Generation (1971 bis 1975)
  - LSI = Large SI, 30 000 T
  - Disketten!
- 5. Generation (1976 bis ?)
  - VLSI = Very Large Scale Integration, 1980 150 000 T
- Mittlerweile:
  - ULSI = Ultra Large SI (> 10^5 T), SLSI = Super Large SI (> 10^6 T),
  - ELSI = Extra Large SI (> 10^7 T), GLSI = Giga Large SI (> 10^8 T)



Entwicklung von Computersystemen (8)

- Heutzutage
  - schnelle Prozessoren mit über 8 GHz Taktfrequenz
  - Verschiedene Rechnerarten
    - PCs
    - Workstations (HP, SUN)
    - Vektor- und Parallelrechner, Supercomputer
      - z. B. ASCI Red mit ca. 10 000 Pentium-Knotenrechner
  - Rechengeschwindigkeiten
    - in FLOPS (Floating Point Operations Per Second)
    - Intel Pentium mit mehr als 2 GFLOPS (2•109 FLOPS)
    - ASCI Red mit mehr als 10 TFLOPS (2•1012 FLOPS)
    - Die 515.000 Computer der Berkley Open Infrastructure for Network Computing bringen es derzeit (Stand April 2011) auf eine Leistung von bis zu 8 PetaFLOPS (PFLOPS 2•1015 FLOPS)
    - 2. Nov. 2011, Japans Rechner "K" erreicht 10,51 PFLOPS
    - 2012, Sequoia, USA 16,32 PFLOPS
    - 2013, National Supercomputer Center in Guangzou, China 33,8 PFLOPS