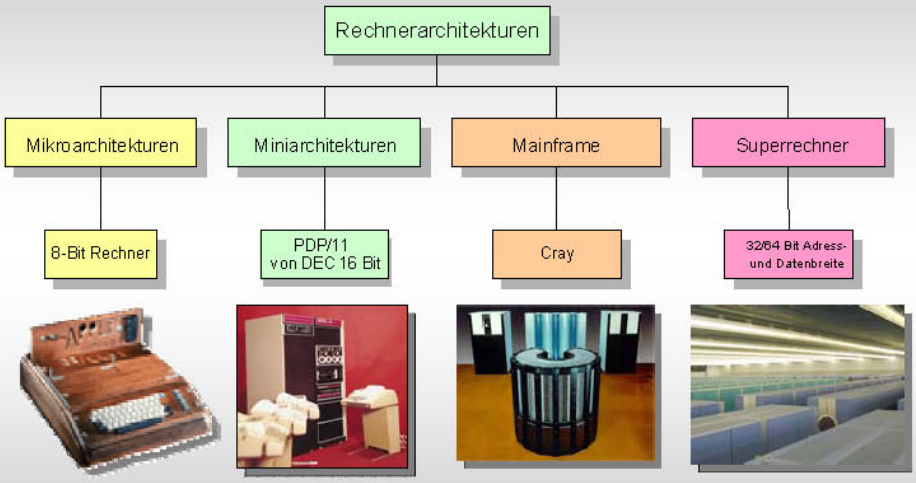
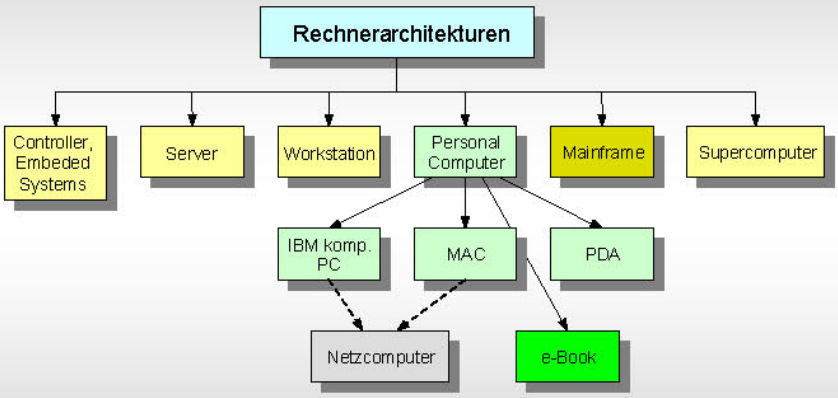
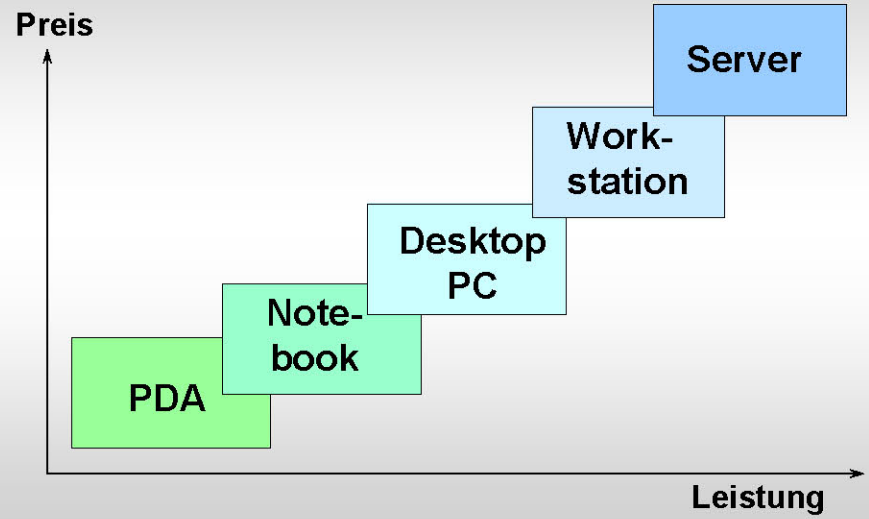
Klassifikation der 70er und 80er Jahre



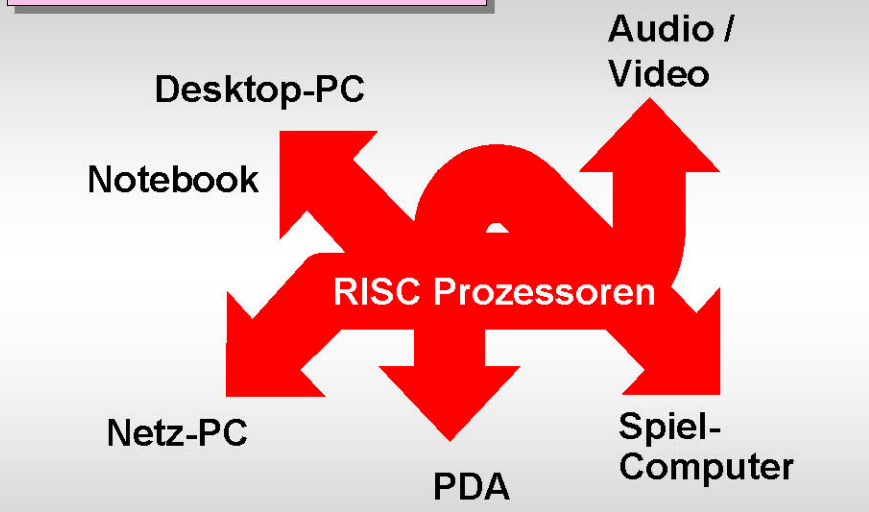
Klassifikation nach Komplexität



Klassifikation der 90er Jahre



Klassifikation im Jahr 2000 und danach



Klassifikation nach M. J. Flynn

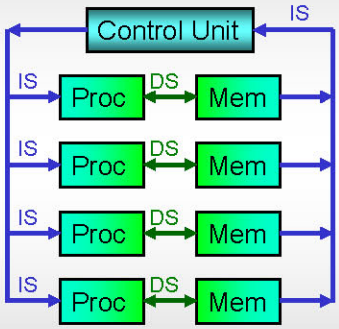
M.J. Flynn: Very High-Speed Computing Systems, Proc. IEEE, Vol.54, 1966, pp.1901-1909

Klassifikation nach **Befehls- (Instruction-)** und **Datenstrom (Datastream)**

Datenstrom \ Befehlsstrom	single	multiple
multiple	**SIMD** (Single Instruction Multiple Data) • n Verarbeitungseinheiten (Prozessoren) bearbeiten gleichzeitig n verschiedene Daten mit einem Befehl • Vektorrechner, MMX, SSE	**MIMD** (Multiple Instruction Multiple Data) • n Datenelemente werden mit m Verarbeitungseinheiten (Prozessoren) gleichzeitig bearbeitet • Multiprozessorrechner (Parallelrechner mit zentralem oder verteiltem Speicher)
single	**SISD** (Single Instruction Single Data) • Jeder Befehl verarbeitet nur ein Datenelement (Sequentielle Abarbeitung von Behlen und Daten) • Klassischer von Neumann- Rechner, Einprozessor-Rechner	**MISD** (Multiple Instruction Single Data) • Ein Datenelement wird nach dem Fließprinzip (Pipelining) nacheinander mit mehreren Verarbeitungseinheiten bearbeitet • Pipeline-Rechner, Grafikprozessoren

Single Instruction Multiple Data (SIMD)

n Verarbeitungseinheiten (Processing Units) bearbeiten gleichzeitig
n verschiedene Daten mit einem Befehl



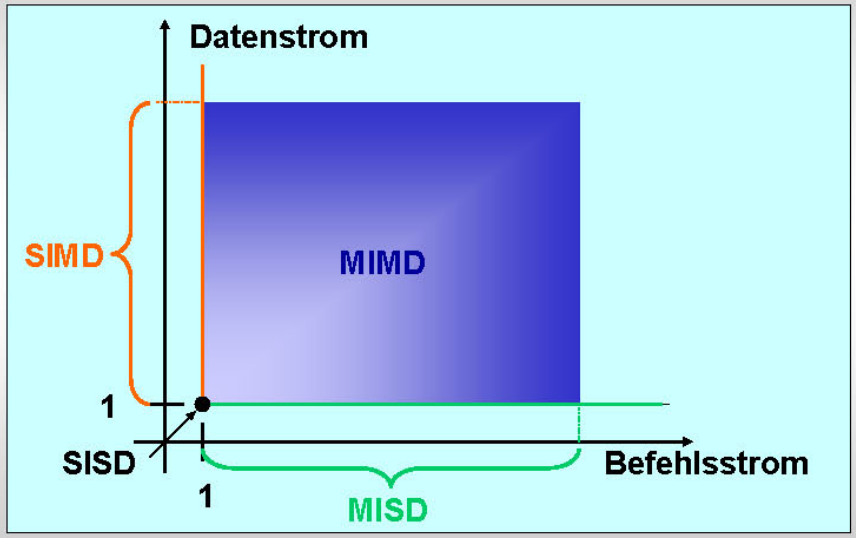
Anwendungsbeispiel:

$$\begin{pmatrix} P0 \\ P1 \\ P2 \\ P3 \end{pmatrix} + \begin{pmatrix} Q0 \\ Q1 \\ Q2 \\ Q3 \end{pmatrix}$$

Addition zweier Matrizen
(4 verschiedene Daten mit einem Befehl)

IS – Instruction Stream
DS – Data Stream

Klassifikation nach Flynn



Parallelitätsebenen

Ebene	Merkmale
4	Programmebenenparallelität (application level parallelism) • parallele Ausführung mehrerer Programme, Prozesse mit Multitasking–Umgebung und mehreren Prozessoren = Multiprocessing (MIMD)
3	Thread-Ebenen-Parallelität (thread level parallelism) • parallele Ausführung mehrer Programmfäden (Threads) innerhalb eines Programms oder einer Multitask-Umgebung) • die CPU muss hierzu mehrere Registersätze und ein komplexes Steuerwerk besitzen = MultiThread Architecture (MTA)
2	Befehlsebenenparallelität (instruction level parallelism) • parallele oder überlappte Ausführung von Befehlen (superskalare Architektur) • mehrere Pipelines parallel; mehr als 1 Befehl pro Takt
1	Bitebenenparallelität (bit level parallelism) • parallele Verarbeitung von mehreren Einzelbitoperationen/operanden (MMX, SSE)

Maschinentypen

- 3-Adressen-Maschine
  - Befehlsform: Operation, Operand 1, Operand 2, Ergebnis
- 2-Adressen-Maschine
  - Befehlsform: Operation, Operand 1, Operand 2
  - eine Operanden-Adresse wird als Ergebnisadresse verwendet
- 1-Adressen-Maschine (Akku-Maschine)
  - Befehlsform: Operation, Operand
  - Spezialregister Akkumulator (für 1. Operand und Ergebnis)
- 0-Adressen-Maschine (Stack-Maschine)
  - Befehlsform: Operation
  - Stapelspeicher (stack, LIFO)
  - als Transfer-Befehle werden noch 1-Adressen-Befehle benötigt

3-Adressmaschinen, 3-Adressbefehle

IBM-Bezeichnung: "SSS„
Befehle bewirken Transfer der Form:

```
Speicher[s1] := Speicher[s2] o Speicher[s3]
```

mit **s1,s2,s3:** Speicheradressen,
o: 2-stellige Operation (+, -, *, /,...).

mögliches Befehlsformat:

Opcode	s1	s2	s3

Befehlswortbreite: 4x 32=128 Bit

Ausführung der Anweisung: D= (A+B)*C:

(Mit a, b, c = Adresse von A, B, C und t1 = Adresse des Zwischenspeichers)

```
add t1, a,b   #Speicher[t1]:=Speicher[a]+Speicher[b]

mult d,t1,c   #Speicher[d]:=Speicher[t1]*Speicher[c]
```


2-Adressmaschinen, 2-Adressbefehle

Verkürzung des Befehlswortes durch Überschreiben eines Operanden mit dem Ergebnis.
IBM-Bezeichnung "SS"
Befehle bewirken Transfer der Form:

```
Speicher[s1] := Speicher[s1] o Speicher[s2]
```

mit **s1,s2:** Speicheradressen,
o: 2-stellige Operation (+, -, *, /,...).

mögliches Befehlsformat:

Opcode	s1	s2

Befehlswortbreite: 3x 32=96 Bit

Ausführung der Anweisung: D= (A+B)*C:

```
move t1,a     #Speicher[t1]:=Speicher[a]
add  t1,b     #Speicher[t1]:=Speicher[t1]+Speicher[b]
mult t1,c     #Speicher[t1]:=Speicher[t1]*Speicher[c]
move d,t1     #Speicher[d] :=Speicher[t1]
```


1½ -Adressmaschinen, 1½ -Adressbefehle

Weitere Verkürzung des Befehlswortes durch Nutzung von Registerspeichern.
IBM-Bezeichnung: "RS„ (beinhaltet auch „RR“)
Befehle bewirken Transfer der Form:

```
Reg[r]:= Reg[r] o Speicher[s]
```

mit **s:** Speicheradresse, **r**: Registernummer
o: 2-stellige Operation (+, -, *, /,...).

mögliches Befehlsformat:

Opcode + Reg.Nr.	s

Befehlswortbreite: 2x 32=64 Bit

Ausführung der Anweisung: D= (A+B)*C:

```
lw,   $8, a    # Reg[8]:=Speicher[a]
add   $8, b    # Reg[8]:=Reg[8] + Speicher[b]
mult  $8, c    # Reg[8]:=Reg[8] * Speicher[c]
sw     d, $8   # Speicher[d]:=Reg[8]
```


1-Adressmaschinen, 1-Adressbefehle

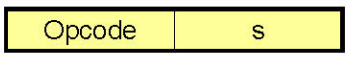
Sonderfall der Nutzung von nur einem Register ("Akkumulator")

Befehle bewirken Transfer der Form:

```
accu:= accu o Speicher[s]
```

mit **s:** Speicheradresse
o: 2-stellige Operation (+, -, *, /,...).

mögliches Befehlsformat:



Befehlswortbreite: 2x 32=64 Bit (i.d.R. kürzere Wortlänge)

Ausführung der Anweisung: D= (A+B)*C:

```
lw,   a    # accu :=Speicher[a]
add   b    # accu:=accu + Speicher[b]
mult  c    # accu:=accu * Speicher[c]
sw    d    # Speicher[d]:=accu
```


0-Adressmaschinen, 0-Adressbefehle

Alle arithmetischen Operationen werden auf einem Stapel ohne explizite Angabe der Operanden ausgeführt. Es werden die beiden obersten Stapelelemente verknüpft und durch das Ergebnis der Operation ersetzt. (Stack-Maschine)

3 Arten von Befehlen:

`push a` : oberstes Stapelelement:= Speicher[a]

`add` : Addiere die beiden obersten Stapelelemente und ersetze sie durch die Summe. Ähnlich für -,*,/

`pop a` : Speicher[a]:= oberstes Stapelelement; entferne dieses Element vom Stapel.

Arithmetik-Befehle: 1 Byte; push, pop: 1 Byte + Adresse (4 Byte) = 5 Byte

Ausführung der Anweisung D= (A+B)*C:

Verwendung der postfix-Notation (umgekehrt polnische Notation): AB+C*

push a → push b → add → push c → mult →pop d

0 – Adressmaschine (Stack-Maschinen)

Beispiel:

```
push a
push b
add
push c
mult
pop d      ←
```

Stapel

Speicher

10	a
20	b
5	c
150	d

- Programmgröße: 22 Byte = 176 Bit
  2 x Arithmetik-Befehle (1 Byte) = 2 Byte + 4 x push, pop (5 Byte) = 20 Byte
- Speicherzugriffe: 4

Gegenüberstellung der Maschinentypen

Gegenüberstellung der Maschinentypen bzgl. Programmlänge und Zahl der Speicherzugriffe für die Gleichung D= (A+B)*C

Maschinentyp	Programmlänge	Speicherzugriffe
3-Adressmaschine	2*128 = 256 Bit	2*3 = 6
2-Adressmaschine	4*96 = 384 Bit	2*3+2*2 = 10
1 ½ -Adressmaschine	4*64 = 256 Bit	1*4 = 4
1-Adressmaschine	4*64 = 256 Bit	1*4 = 4
0-Adressmaschine	4*40+2*8 = 176 Bit	1*4 = 4