RISC-Prozessoren (1)

1) 8 Befehlsklassen und ihre mittlere Ausführungshäufigkeit (Fairclough):

• Zuweisung bzw. Datenbewegung	45,28%
• Programmablauf	28,73%
• Arithmetik	10,75%
• Vergleich	5,92%
• Logik	3,91%
• Shift	2,93%
• Bitmanipulation	2,05%
• Input/Output und verschiedenes	0,44%

2) Bereichsintervalle für literale Operanden (Tanenbaum):

• -15 bis +15	56%
• -511 bis +511	98%

---> reserviertes 5-bit-Feld in einem Befehlsformat deckt 50% des Auftretens von Literalen ab.

RISC-Prozessoren (2)

3) Speicherplatzbedarf für Parameter und lokale Variable bei Prozeduraufruf:

12 Worte in 95% aller Fälle (Tanenbaum, Halbert und Kessler)

---> mit ca. 12 on-chip-Registern ist eine Subroutine ohne Hauptspeicherzugriff zu bearbeiten

---> - Reduktion des Busverkehrs Prozessor - Speicher
- erlaubt kleine Adresssfelder im Befehlsformat

---> - Jeder Subroutine sollten ca. 12 on-chip-Register zugeordnet werden (anstatt komplexe Adressberechnungen zu unterstützen)
- Der Transfer von Daten zwischen Registern sollte sehr schnell sein

RISC-Prozessoren (3)

wünschenswerte Eigenschaften einer RISC-Architektur:

- **genügend on-chip Speicher (Register)**

- **effiziente Parameterübertragung zwischen Subroutinen (Registerfenster)**

- **Im Schnitt sollte ein Befehl in einem Taktzyklus ausgeführt werden.**

- **Keine Implementierung von eher weniger häufigen, aber komplexen Maschinenbefehlen**
  - **Verschwendung von Silizium und Konflikt mit Punkt darüber**
  - **höherer Zeitaufwand für Entwurf, Fabrikation und Test eine Prozessors**

- **keine mikroprogrammierte Architektur**
  - **Im Grenzfall ist ein RISC-Prozessor eine mikroprogrammierte Architektur, in der die Unterscheidng zwischen Maschinenbefehlszyklus und Mikrobefehlszyklus hinfällig ist**

- **Implementierung eines Pipelining-Mechanismus (Abkehr vom skalaren Prozessormodell)**
  - **erlaubt die überlappende Ausführung von Befehlen**

- **möglichst wenige Befehlsformate (am besten nur 1)**
  - **Dekodierung eines Befehls in seine Komponentenfelder kommt mit einem Minimum an Dekodierlogik aus ---> weniger Bits für Op-Code - Feld, da z.B nur 1 Add - Befehl für ein Befehlsformat und nicht 3 verschiedene ---> schnellere Dekodierphase**