# Technische Informatik 1

**Prof. Dr. Rolf Drechsler**

**Christina Plump**

# Überblick

## Teil 1: Der Rechneraufbau (Kapitel 2-5)

- Rechner im Überblick
- **Pipelining**
  - **Befehlssätze**
  - **Pipelining**
- Speicherhierarchie
- Parallelverarbeitung

## Teil 2: Der Funktionalitätsaufbau (Kapitel 6-12)

- Kodierung von Zeichen und Zahlen
- Grundbegriffe, Boolesche Funktionen
- Darstellungsmöglichkeiten
- Schaltkreise, Synthese, spezielle Schaltkreise

# Kapitel 3: Pipelining

Befehlssätze

Pipelining

# Lernziele

- CISC und RISC Rechner kennen und in ihrer Charakterisierung verstehen und unterscheiden können
- Vor- und Nachteile für RISC /CISC Rechner kennen und benennen können
- Gründe für Marktverbreitung von RISC Rechnern kennen

# 3.1 Befehlssatz

Die Extreme:

CISC-Rechner und RISC-Rechner

**Begriffe:**

- CISC = Complex Instruction Set Computer
- RISC = Reduced Instruction Set Computer
- Vergleich der beiden Prinzipien

# CISC – Complex Instruction Set Computer

- Charakterisierung eines CISC-Rechners
  - großer Satz von Maschinenbefehlen, zum Teil auch Maschinenbefehle, die komplexe Operationen auszuführen haben
  - sehr unterschiedliche Ausführungszeiten der verschiedenen Maschinenbefehle
  - Steuerung nicht durch Hardware, sondern durch ein Mikroprogramm, das in einem ROM auf dem Prozessor abgespeichert ist.
- Erster moderner Rechner (IBM 369) war ein CISC-Rechner

# Argumente für CISC

- Komplexe Befehle schließen die semantische Lücke zwischen den verwendeten Hochsprachen und der Hardware

   Maschine, die „direkt“ Hochsprache verarbeiten kann

- Komplexe Befehle verringern den Hauptspeicherbedarf für den Programmcode
- Je kürzer das Maschinenprogramm, desto schneller läuft es ab, da weniger auf den Hauptspeicher zugegriffen werden muss
- Komplexe Befehle vereinfachen den Compiler
- Je mehr Funktionalität unterhalb der Maschinensprache, desto zuverlässiger ein Rechner (übliche Ansicht: „Hardware ist sicher, Software enthält Fehler“)

# Argumente gegen CISC

- Viele Zugriffe auf den Programmspeicher können aufgrund von „Cache-Strategien“ (siehe nächstes Kapitel) schnell durchgeführt werden
  **Grund:** Lokalitätseigenschaften der Codeausführung
- Hardware ist nicht zuverlässiger als Software  Hardware-Fehler sind schwieriger zu korrigieren als Software-Fehler
- Die komplexen Befehle, die eigentlich die Compiler vereinfachen sollten, werden kaum von den Compilern benutzt

# Statistik für CISC-Rechner (ca. 1990)

| Befehlstypen | Ausführungshäufigkeit |
|---|---|
| Einfache Befehle (move, add register, branch, call, return) | 84% |
| Boolesche Operationen auf Wörtern | 7% |
| Gleitkommaoperationen | 4% |

Einfache Befehle

- Mehr als 90% der ausgeführten Befehle sind einfach, könnten schnell ausgeführt werden!
- Die Unterstützung der restlichen 10% verlangsamt in der Regel die Ausführungsgeschwindigkeit der obigen 90%!

# RISC – Reduced Instruction Set Computer

- Charakterisierung
  - einfache Befehle, kleine Befehlssätze
  - in einem Verarbeitungsschritt ausführbar
  - großer, interner Registersatz
  - leistungsfähige Speicherhierarchie
  - Einsparung des Mikrobefehlssatzes, fest verdrahtetes Steuerwerk

- Beispiel:
  - MIPS-R2000/3000 (1982-89)
  - PowerPC

# Vergleich RISC versus CISC (1)

| | RISC | CISC |
|---|---|---|
| Ausführungszeit | 1 Datenpfadzyklus | > 1 Datenpfadzyklus |
| Instruktionsanzahl | klein | groß |
| Instruktionsformat | einfach/einheitlich | variabel |
| Steuerung über | Hardware | Mikroprogramm |
| Hauptspeicherzugriffe | LOAD/STORE-Architektur | Keine Einschränkungen |
| Pipelining | Möglich | Kaum möglich |
| Verlagerung d. Komplexität | Compiler | Hardware |

# Vergleich RISC versus CISC (2)

## RISC - Rechner

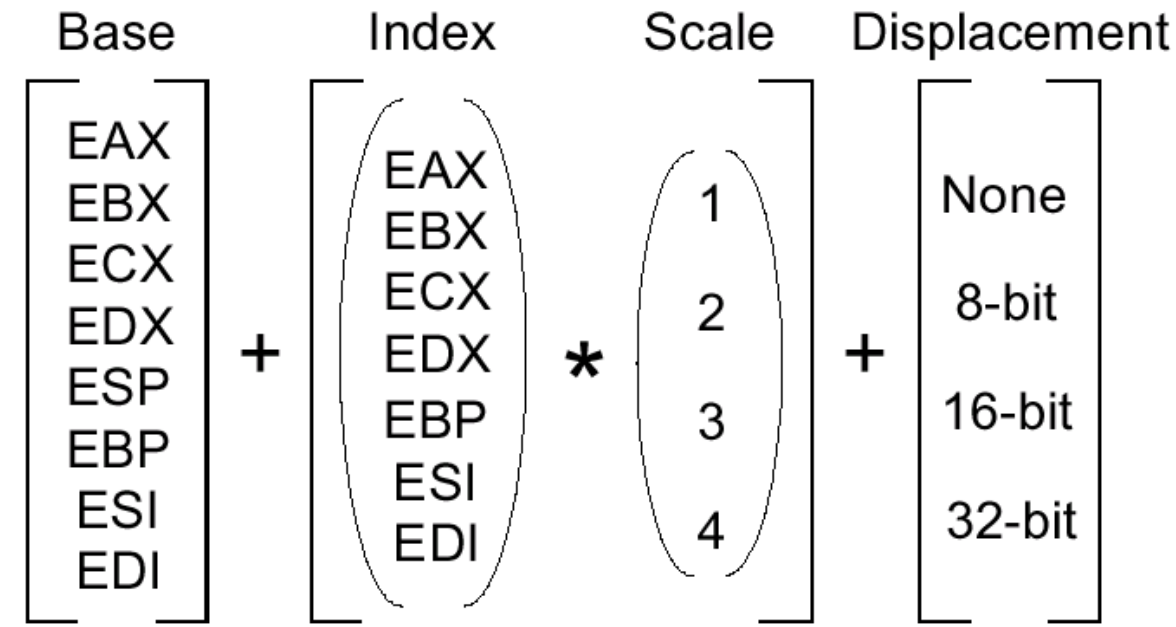
- PowerPC (IBM, RISC)
  - Operanden aus dem Hauptspeicher kommen nur in **`load`**-, **`store`**-, **`branch`**- oder **`cache`**-Befehlen vor
  - Drei Modi der Adressberechnung:
    - Registerinhalt + Konstante
    - Summe zweier Register
    - Registerinhalt

## CISC - Rechner

- Pentium 4 (Intel, CISC)
  - Mögliche Operanden (für bel. Befehle):
    - Konstante (natürlich nicht für Ergebnisse)
    - Registerinhalt
    - Speicherplatz
    - I/O-Port
  - Adressberechnung:
    - Speicheradresse= Segment + Offset
    - Segment steht in einem speziellen Register
    - Offset wird entsprechend der folg. Tab. berechnet

# Vergleich RISC versus CISC (3)

- Offsetberechnung im Pentium 4:




Offset = Base + (Index ∗ Scale) + Displacement

# Designprinzipien für Rechner

- Was sind die Schlüsseloperationen für diesen speziellen Rechnertyp?
- Entwurf des Datenpfades (Rechenwerk mit Verbindungsstruktur) für diese Schlüsseloperationen
- Entwurf der Maschinenbefehle mit dem Ziel, dass möglichst jeder Befehl in einem Datenpfadzyklus (Fetch-Decode-Execute) ausführbar ist (kein Mikroprogramm)
- Erweiterung des Befehlssatzes nur dann, wenn dadurch die Maschine nicht langsamer wird