Organisatorisch:
Bonusklausur in der Semestermitte, im HS, Anrechnung 10%

## 3.3. CISC-Prozessor

CISC: Complex Instruction Set Computer (Rechner mit komplexem Befehlsatz)

Grundidee: sehr viele Befehle, von einfach bis complex

Ziele (u.a.): Befehle auf dem Niveau einer höheren Programmiersprache oder nahe daran

Bsp.: Pico-Java-Prozessor (SUN)

- ➔ Direkte Ausführung von Java-Bytecode
- ➔ Ausgestorben, nicht erfolgreich am Markt

Typische Realisierung der komplexen Ablaufsteuerung durch Mikroprogrammsteuerung ($\mu$ Sequenzer, F4):

- Jeder CISC-Befehl wird in eine sequentielle Folge von Mikrobefehlen umgesetzt. (einschließlich Sprünge)
- Mikrobefehle werden aus dem Mikroprogrammspeicher gelesen
- Der aktuell in Bearbeitung befindliche Mikrobefehl steuert ALU, Register und Bus durch vom aktuelle Mikrobefehl erzeugte Steuersignale
- Der Mikroprogrammstatus aus Rechenwerk kann Verzweigungen und Schleifen im Mikroprogramm steuern
- Mikrobefehlszähler zeigt auf den gerade aktuellen Mikrobefehl
- Makrobefehl stellt zu Beginn die Adresse im Mikroprogrammspeicher des ersten Befehls des ihm zugeordneten Mikroprogramms im Mikrobefehlszähler ein

CISC trennt nicht streng zwischen ausführenden Befehlen und Speichertransport und erzeugt dabei auch variable Operandenadressierung:(F5)

- 0-Operandenadressierung: alle E- und A-Operanden sind auf festen Plätzen, z.B. an der Spitze des Stacks (wie z.B. in der jetzt nicht mehr gebräuchlichen Sprache FORTH)
- 1-Operandenadressierung: 1 Operand und Ergebnis sind in dem gleichen festen Register, der zweite Operand ist adressierbar (siehe Prozessorgrundstruktur in RA1)
- 2-Operandenadressierung: beide Eingangoperanden sind adressierbar, Ergenis überschreibt den ersten Operanden (die meisten Befehle vom Prozessor in der Übung verwenden dieses Prinzip)
- Alle E-Operanden und Ergebnis sind frei adressierbar (relativ selten)

Bei CISC können die Operanden auch im Speicher liegen

Gegenüberstellung von RISC und CISC:

| | |
|---|---|
| Einfache Befehle | Einfache bis komplexe Befehle |
| Für komplexere Funktionen mehrere Maschinenbefehle | Für komplexere Funktionen je einen Befehl |
| Programmcode wird größer | Kleinerer Programmcode |
| Optimierung auf der Folge der einfachen Befehle für komplexere Funktionen | Optimierung nur möglich auf dem Niveau der komplexeren Maschinenbefehle, auf Mikroprogrammniveau nicht |
| Einfache Ablaufsteuerung und Logik | Mikrosequenzer und komplexe Logik |
| Weniger Chipfläche<br>Geringerer elektrischer Leistungbedarf<br>Schnellere Logik | Mehr Chipfläche<br>Höherer Leisungsbedarf<br>Langsamere Logik |
| Regelmäßige logische und | Unregelmäßige logische und |

| | |
|---|---|
| zeitliche Ausführung der Befehle regelmäßig -> günstig für Mikroparallelität | zeitliche Ausführung -> ungünstig für Mikroparallelität |