Lernzettel

Maschinenbefehlsbearbeitung auf Registertransfer-Ebene: Mikrooperationen, Datenpfade und Steuerlogik

Universität: Technische Universität Berlin
Kurs/Modul: Rechnerorganisation
Erstellungsdatum: September 20, 2025

Zielorientierte Lerninhalte, kostenlos!
Entdecke zugeschnittene Materialien für deine Kurse:

https://study.AllWeCanLearn.com

Rechnerorganisation

Lernzettel: Rechnerorganisation

Maschinenbefehlsbearbeitung auf Registertransfer-Ebene: Mikrooperationen, Datenpfade und Steuerlogik

(1) Grundlagen und Ziel der RTL-Benutzung. Auf der Registertransfer-Ebene (RTL) wird die Ausführung eines Maschinenbefehls als Folge von Mikrooperationen beschrieben, die direkt zwischen Registern, dem Datenpfad und der Steuerlogik stattfinden. Der zentrale Gedanke ist: Aus einem Befehl werden Sequenzen von kleinen Operationen generiert, z. B. Werte von Registern in andere Register transferieren, arithmetische/logische Operationen in der ALU durchführen oder Speicherzugriffe ausführen. Der Datenpfad besteht aus Registern, Busse, einem Arithmetic-Logic Unit (ALU), Multiplexern, Verschiebern und Speichernkomponenten wie MAR, MDR, RAM. Die Steuerlogik steuert diese Mikrooperationen zeitlich synchron durch Taktzyklen.

(2) Mikrooperationen. Mikrooperationen sind elementare Schrittoperationen, die auf Register oder Speicher wirken. Typische Typen:

- Datentransfer zwischen Registern: $\mathrm{R_{dst}} \leftarrow \mathrm{R_{src}}$
- ALU-Operationen: $\mathrm{R_{acc}} \leftarrow \mathrm{ALU}(\mathrm{R}_a, \mathrm{R}_b, \mathrm{Ctrl})$
- Speicher-zu/ Speichernutzung: $\mathrm{MDR} \leftarrow \mathrm{Mem[MAR]}$, $\mathrm{RAM[MAR]} \leftarrow \mathrm{MDR}$
- Adressberechnung: $\mathrm{EA} \leftarrow \mathrm{Base} + \mathrm{Index}$ (oder ähnliche Adressierungsmodi)
- Programmzählung: $\mathrm{PC} \leftarrow \mathrm{PC} + 1$ oder $\mathrm{PC} \leftarrow$ Zieladresse

Diese Mikrooperationen werden durch die Steuerlogik in aufeinanderfolgende Takte umgesetzt. Beispiele typischer Abläufe:

$$\text{Fetch}: \quad \mathrm{MDR} \leftarrow \mathrm{Mem[PC]}; \quad \mathrm{IR} \leftarrow \mathrm{MDR}; \quad \mathrm{PC} \leftarrow \mathrm{PC} + 1$$

$$\text{Operate}: \quad \mathrm{Operand1} \leftarrow \mathrm{R}_1; \quad \mathrm{Operand2} \leftarrow \mathrm{R}_2; \quad \mathrm{R}_{dest} \leftarrow \mathrm{ALU}(\mathrm{Operand1}, \mathrm{Operand2})$$

(3) Datenpfad (Datapath). Zentrale Komponenten eines typischen RTL-Datenpfades:

- Registerbusse und Registerdateien (R0, R1, ..., Rn) zur Speicherung temporärer Werte
- Rechenwerk (ALU) mit Steuerbits, das arithmetische/logische Operationen durchführt
- Multiplexer (MUX) zur Auswahl von Operanden oder Zielregistern
- Shifter/Rotator für Verschiebungsoperationen
- Speicherzugriffe über MAR (Memory Address Register) und MDR (Memory Data Register)
- Befehlsregister (IR) zur Dekodierung des nächsten Mikroprogramms

Der Fluss der Daten erfolgt typischerweise über Bussysteme, wobei zu jedem Takt die passenden Signale gesetzt werden, damit ein Mikrooperationen-Satz innerhalb eines Taktzyklus ausgeführt wird. Beispielhafter Ablauf innerhalb eines Zugriff-Zyklus:

$$\text{Takt } T_0: \ \text{PC} \to \text{MAR}, \ \text{PC} \leftarrow \text{PC} + 1; \quad \text{Takt } T_1: \ \text{MDR} \leftarrow \text{Mem[MAR]}, \ \text{IR} \leftarrow \text{MDR}$$

(4) Steuerlogik. Die Steuerlogik koordiniert Mikrooperationen; zweiGrundtypen sind verbreitet:

- Hardwired-Control: Signale werden durch fest verdrahtete Logik in Abhängigkeit von Takt und Zustandsbits erzeugt.
- Mikroprogrammierte Control: Eine Microinstruction-Store (Control Store) legt pro Takt die Signale fest. Die Sequenz der Mikroinstructionen definiert den Ablauf eines Maschinenbefehlssatzes.

Typische Felder einer Mikroinstruction (fiktives Beispiel, zur Orientierung):

$$\text{Fetch-Field}: \quad \text{PC} \to \text{MAR}, \ \text{Mem[MAR]} \to \text{MDR}$$

$$\text{Decode-Field}: \quad \text{IR} \to \text{Dekoder}, \ \text{Operand-Adressen} \to \text{Registerdatei}$$

$$\text{Execute-Field}: \quad \text{ALU-Op} \to \text{ALU}, \ \text{Rdest} \leftarrow \text{ALU-Result}$$

In der Praxis wird der Mikrocode so gestaltet, dass häufig wiederkehrende Muster effizient ablaufen, z. B. Fetch-Decode-Execute, oder spezialisierte Mikrofolgen für Speicherzugriffe und Adressberechnungen.

(5) Einfaches Beispiel eines RTL-Datennutzungszyklus. Betrachte einen einfachen Befehl ADD R1, R2, wobei der Wert von R1 zu R2 addiert und das Ergebnis in R1 gespeichert wird:

$$\begin{aligned}
&\text{T0}: \quad \text{PC} \leftarrow \text{PC} + 1, \ \text{MAR} \leftarrow \text{PC};\\
&\text{T1}: \quad \text{MDR} \leftarrow \text{Mem[MAR]}; \ \text{IR} \leftarrow \text{MDR};\\
&\text{T2}: \quad \text{A} \leftarrow \text{R1}, \ \text{B} \leftarrow \text{R2};\\
&\text{T3}: \quad \text{R1} \leftarrow \text{ALU(A, B)}; \quad \text{fertig}
\end{aligned}$$

Hier wird der Datenpfad genutzt, um Werte zu holen, zu addieren und das Ergebnis zurück in das Zielregister zu schreiben. In realen Prozessoren werden solche Abläufe in mehreren Mikrooperationen pro Befehlssatz umgesetzt und je nach Architektur weiter optimiert (Cache-Zugriffe, Pipeline, Speichermanagement).

(6) Timing, Synchronisation und Pipeline-Überlegungen. Mikrooperationen sind zeitlich durch Taktzyklen gesteuert. Typische Eigenschaften:

- Synchronisation: Alle Datenwechsel erfolgen zu Taktflanken, um Stabilität zu gewährleisten.
- Hazard-Vermeidung: Manchmal sind Wartezyklen nötig, wenn Datenabhängigkeiten bestehen.

- Pipeline-Grundidee: Mehrere Befehle überlappen sich in aufeinanderfolgenden Stufen (Fetch, Decode, Execute). Pipeline-Konflikte (Hazards) erfordern Stalls oder Forwarding.

(7) Übungsnotizen und Reflexion.

- Wie würden Hardwired- versus Mikrocode-Steuerlogik die Komplexität eines Befehlssatzes beeinflussen?
- Welche Mikrooperationen sind für Speicherzugriffe am kritischsten (z. B. Adressbildung, Cache-Kohärenz)?
- Welche Vorteile bietet ein sauberer Datenträgerpfad bei der Fehlersuche in einem RTL-Design?