6.1 Die von-Neumann-Architektur

ZUSATZ

> ZUSATZ
>
> **Dieses Kapitel steht in keinem der beiden Lehrpläne von 2019** — weder im Grundkurs noch im Leistungskurs. Es ist damit nicht verbindlich und kein Gegenstand einer Kursarbeit, solange es nicht ausdrücklich zum Unterrichtsgegenstand erklärt wird. Aufgenommen ist es, weil die Registermaschine zeigt, was hinter jeder Zuweisung und jeder Schleife steckt — und weil der Befehlszyklus die Brücke zwischen Kapitel 1 und der wirklichen Maschine schlägt.

Als der **ENIAC** 1946 an der Universität von Pennsylvania der Öffentlichkeit vorgestellt wurde — 17 468 Elektronenröhren und rund 1500 Relais auf 27 Tonnen Masse — konnte er wie seine Vorgänger Z3 und Mark I nur rechnen. Für eine freie Programmierung war er nicht geeignet: Ein neues Problem bedeutete, Kabel umzustecken und Schalter neu zu setzen, was Tage dauern konnte.

Schon während der ENIAC gebaut wurde, beschrieb der Mathematiker **John von Neumann** (1903–1957) in seinem *First Draft of a Report on the EDVAC* (1945) die Grundstruktur eines frei programmierbaren Rechners. Der entscheidende Gedanke: Das Programm wird nicht verdrahtet, sondern wie die Daten im Speicher abgelegt. Wer die Maschine umprogrammieren will, tauscht nur noch Speicherinhalt aus. Diese Struktur heißt bis heute **von-Neumann-Architektur**.

6.1.1 Funktionseinheiten

- **Zentraleinheit** mit **Steuerwerk**, **Rechenwerk** und **Arbeitsspeicher**
- **Ein- und Ausgabegeräte**

Steuerwerk und Rechenwerk sitzen gemeinsam auf einem Chip, dem **Prozessor** (CPU). Verbunden werden alle Einheiten über den **Systembus**, der aus drei Teilen besteht:

Bus	Aufgabe
Datenbus	transportiert die eigentlichen Daten
Adressbus	legt fest, welche Speicherzelle gemeint ist
Steuerbus	überträgt Signale wie „lesen” oder „schreiben”

Speicher. Der Speicher ist eine lineare Folge gleich großer Zellen. Jede Zelle hat eine Nummer, ihre **Adresse**, und kann einen Wert aufnehmen — einen Datenwert oder einen Befehl, die Maschine sieht da keinen Unterschied. Für die Modellmaschine in diesem Kapitel teilen wir den Speicher trotzdem in einen **Programmspeicher** (die Befehle) und einen **Datenspeicher** (die Werte) auf. Das ist eine didaktische Vereinfachung: Sie erspart uns das Mitzählen von Adressen und macht die Abläufe lesbar.

Steuerwerk. Es organisiert die Programmausführung: Befehle in der richtigen Reihenfolge holen, decodieren, interpretieren und die beteiligten Einheiten mit Steuersignalen versorgen.

Rechenwerk. Hier werden arithmetische und logische Verknüpfungen ausgeführt, daher der Name **ALU** (Arithmetic Logic Unit). Die Operanden bekommt sie vom Steuerwerk zugeteilt.

Register. Register sind besonders schnelle Speicherplätze im Prozessor selbst. Drei sind für uns wichtig:

- Der **Akkumulator** nimmt Werte aus dem Datenspeicher auf und sammelt Rechenergebnisse (von lat. *accumulare*, anhäufen). Vor einer Addition steht ein Operand im Akkumulator, der andere im Speicher; danach steht das Ergebnis im Akkumulator.
- Das **Befehlsregister** enthält den gerade auszuführenden Befehl.

- Der **Befehlszähler** (Befehlsadressregister) enthält die Adresse des Befehls, der als nächstes geholt wird. Solange ein Befehl noch bearbeitet wird, steht dort also dessen *eigene* Adresse; erst der letzte Schritt des Befehlszyklus rückt ihn weiter.

6.1.2 Die fünf Prinzipien

1. **Ein gemeinsamer, adressierbarer Speicher.** Der Speicher besteht aus gleich großen Zellen mit fortlaufenden Adressen, und **Befehle und Daten liegen im selben Speicher**. Genau das ist der Kerngedanke: Ein Programm ist für die Maschine nichts anderes als Daten und kann deshalb geladen, verändert und ausgetauscht werden.
2. **Programmgesteuertes Arbeiten.** Der Aufbau des Rechners ist unabhängig vom Problem; was er tut, bestimmt allein das Programm.
3. **Sequentielle Befehlsabarbeitung.** Die Befehle werden der Reihe nach ausgeführt. Sie enthalten nicht die Werte, sondern die *Adressen* der Werte.
4. **Sprungbefehle.** Von der Reihenfolge darf abgewichen werden; der Sprung kann von gespeicherten Werten abhängen. Erst dadurch entstehen Verzweigung und Schleife.
5. **Binärcodierung.** Sämtliche Daten und Befehle werden binär dargestellt.

Merke

Punkt 3 ist der Grund für Variablen: Ein Programm rechnet nicht mit Zahlen, sondern mit Adressen, hinter denen Zahlen stehen. Punkt 4 ist der Grund dafür, dass überhaupt etwas anderes als eine starre Rechenvorschrift möglich ist.

Stolperstein

Unsere Modellmaschine hat zwei getrennte Speicher — das widerspricht Prinzip 1. Trennt eine *echte* Maschine Programm- und Datenspeicher samt ihrer Busse, heißt das nicht mehr von-Neumann-, sondern **Harvard-Architektur**; viele Mikrocontroller sind so gebaut. Der Unterschied ist keine Spitzfindigkeit: Nur im gemeinsamen Speicher kann ein Programm ein anderes Programm als Daten behandeln — und genau darauf beruhen Compiler, Betriebssysteme und Virenscanner ebenso wie ein Teil der Sicherheitslücken, die Sie in Kapitel 8 wiedersehen.

Arbeitsauftrag 1

1. Ordnen Sie zu, welche Funktionseinheit zuständig ist: eine Zahl addieren; den nächsten Befehl holen; einen Wert dauerhaft ablegen; entscheiden, ob gesprungen wird.
2. Erklären Sie, warum ein Rechner drei getrennte Busse braucht und nicht mit einem auskommt.
3. Was wäre nicht möglich, wenn es Prinzip 4 nicht gäbe? Nennen Sie zwei Beispiele aus Ihren bisherigen Programmen.
4. Recherchieren Sie den Unterschied zwischen von-Neumann- und Harvard-Architektur und nennen Sie je einen Einsatzbereich.