# I GRUNDLAGENWISSEN

S. BARTELS, 17.4.2018

## 1. Abstrakte von-Neumman-Rechner

Um die Funktionsweise von Computern, die im Folgenden auch als Rechner bezeichnet werden, zu verstehen, ist es sinnvoll, sie auf ihre wesentlichen Bestandteile zu reduziere. Das Modell des *von-Neumann-Rechners* beschreibt einen Computer durch die Komponenten *Prozessor*, *Hauptspeicher* und *Ein- und Ausgabeeinheiten* sowie einen *Datenbus*, die in Abbildung 1 skizziert sind.

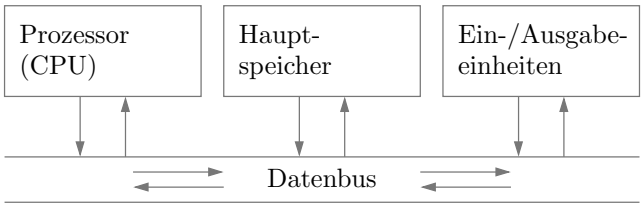


Abbildung 1. Aufbau des von-Neumann-Rechners.

Herzstück eines von-Neumann-Rechners ist der Prozessor, der auch als *Central Processing Unit (CPU)* bezeichnet wird und sämtliche Rechnungen durchführt. Im Hauptspeicher werden Programme und Daten abgelegt. Ein- und Ausgabeeinheiten wie Tastatur, Monitor, Drucker, Maus und weitere Speichermedien dienen der Interaktion mit dem Benutzer. Der Datenbus sorgt für den Transfer von Informationen zwischen den Komponenten.

## 2. Rechnen mit logischen Ausdrücken

Sämtliche Daten auf Computern liegen im Binärformat vor, das heißt sie werden geeignet durch die Informationen 0 und 1 beziehungsweise entsprechende elektrische Signale realisiert. Eine einzelne solche 0/1-Information bezeichnet man als *Bit*. Auch das eigentliche Rechnen, also die Durchführung arithmetischer Operationen, erfolgt mit Hilfe von Bits. Dabei nutzt man eine Kombination der folgenden drei Varianten des Arbeitens mit den Werten 0 und 1:

(i) Repräsentation von Dezimalzahlen im Dual- bzw. Binärsystem
(ii) Darstellung arithmetischer Operationen durch logische Ausdrücke
(iii) Technische Umsetzung mit Transistorschaltungen

Für die Herleitung geeigneter logischer Ausdrücke werden die Gesetze der *Booleschen Algebra* genutzt. Dabei stehen die Werte 0 und 1 für die Wahrheit einer Aussage und repräsentieren die Bewertungen falsch und wahr. Beispiele für Aussagen sind *"heute scheint die Sonne"* und *"morgen ist Dienstag"*, die abhängig vom Ort und Zeitpunkt ihrer Auswertung falsch oder wahr sein können. Besonders interessant sind Verknüpfungen von Aussagen. Sind $A$ und $B$ zwei Aussagen, so ist beispielweise die UND-Verknüpfung $A \wedge B$ eine Aussage, die dann und nur dann wahr ist, wenn $A$ und $B$ gleichzeitig wahr sind. Weitere sind das nicht-ausschließende ODER, in Zeichen $A \vee B$, sowie die Implikation $A \implies B$. Die in Tabelle 1 dargestellte *Wahrheitstabelle* gibt an, wie diese Verknüpfungen definiert sind.

| $A$ | $B$ | $A \wedge B$ | $A \vee B$ | $A \implies B$ | $\neg A$ |
|---|---|---|---|---|---|
| 0 | 0 | 0 | 0 | 1 | 1 |
| 0 | 1 | 0 | 1 | 1 | 1 |
| 1 | 0 | 0 | 1 | 0 | 0 |
| 1 | 1 | 1 | 1 | 1 | 0 |

TABELLE 1. Wertetabelle zur Definition der logischen Verknüpfungen UND, ODER, IMPLIZIERT und NEGATION. Für verschiedene Belegungen der booleschen Variablen $A$ und $B$ sind die Ausdrücke $A \wedge B$, $A \vee B$, $A \implies B$ und $\neg A$ entweder falsch oder wahr beziehungsweise haben den Wert 0 oder 1.

Während die ODER-Verknüpfung gut nachvollziehbar ist, da sie genau dann wahr ist, wenn mindestens eine der beiden Aussagen wahr ist, ist die Festlegung der Implikation zunächst überraschend, insbesondere, dass aus einer falschen Aussage eine wahre folgen darf. Entscheidend für die Gültigkeit der Implikation $A \implies B$ ist aber, dass $B$ immer dann wahr ist, wenn $A$ wahr ist. Dies macht man sich am besten an einfachen Aussagen wie $A =$ *"es regnet"* und $B =$ *"es ist bewölkt"* und der Implikation $(A \implies B) =$ *"wenn es regnet, dann ist es bewölkt"* klar.
Die Verknüpfungen $\wedge$ und $\vee$ erfüllen Kommutativ-, Assoziativ- und Distributivgesetze sowie unter Hinzunahme der Negation, die durch $\neg A$ dargestellt wird, die so genannten *DeMorganschen Gesetze*

$$\neg(A \wedge B) = \neg A \vee \neg B, \quad \neg(A \vee B) = \neg A \wedge \neg B.$$

Dabei bedeutet Gleichheit zweier Ausdrücke, dass sie für jede Belegung der beteiligten Variablen denselben Wahrheitswert ergeben. Mit den Konstanten 0 und 1 bezeichnet man in Ausdrücken die Aussagen, die stets falsch beziehungsweise wahr sind. Mit ihnen gilt beispielsweise $A \wedge 1 = A$ und $A \vee 0 = A$ sowie $A \vee \neg A = 1$.

## 3. Programmiersprachen und Algorithmen

Da Prozessoren nur Folgen von Nullen und Einsen verarbeiten können, ist die direkte Programmierung eines Prozessors kaum praktikabel. Programmiersprachen erlauben die automatisierte Übersetzung von verständlichen Kommandos in Maschinenbefehle. Eine Folge von Kommandos einer Programmiersprache bezeichnet man als *Programm*. Damit dies übersetzt werden kann, muss es gewissen *syntaktischen* und *semantischen* Bedingungen genügen. Als Syntax einer Programmiersprache bezeichnet man dabei gewissermaßen die durch sie vorgegebene Rechtschreibung und Grammatik, während unter Semantik die inhaltlich korrekte Verwendung von Kommandos bezeichnet wird. Im Kontext der deutschen Sprache ist beispielsweise die Aussage *"die Straße ist eckig"* syntaktisch korrekt nicht jedoch semantisch, wohingegen die Aussage *"die Straße ist breit"* unabhängig vom Wahrheitsgehalt beiden Anforderungen genügt.
Programme sind Realisierungen von *Algorithmen*. Als Algorithmus bezeichnen wir dabei eine Folge von elementaren Operationen ähnlich einem Kochrezept. Wie detailliert die einzelnen Schritte angegeben werden, hängt sehr vom Kontext ab und kann stark variieren. Die $pq$-Formel zur Lösung einer quadratischen Gleichung $x^2 + px + q = 0$ also

$$x_{1,2} = -p/2 \pm \left(p^2/4 - q\right)^{1/2}$$

kann für sich schon als Algorithmus angesehen werden, allerdings lässt sie sich in dieser Form nicht direkt in einer Programmiersprache angeben. Etwas konkreter wäre folgende Form.

**Algorithmus 3.1** ($pq$-Formel)**.** Eingabe: *Reelle Zahlen $p$ und $q$.*

(1) *Setze $r = p^2/4 - q$.*
(2) *Ist $r < 0$, so generiere Fehlermeldung* "nicht lösbar" *und stoppe.*
(3) *Berechne $s = \sqrt{r}$.*
(4) *Definiere $x_1 = -p/2 + s$ und $x_2 = -p/2 - s$.*

Ausgabe: *Lösungen $x_1$ und $x_2$.*

Auch dieser Algorithmus ist nicht unbedingt direkt umsetzbar, da zum Beispiel die Quadratwurzel geeignet berechnet werden muss. Dies kann effizient mit dem *Heronschen Verfahren* geschehen, der Approximationen von $\sqrt{r}$ mittels der Initialisierung $s_0 = 1$ und der Iteration $s_{k+1} = (s_k + r/s_k)/2$ berechnet. Diese Vorschrift wird so lange angewendet, bis ein Abbruchkriterium erfüllt ist.

**Algorithmus 3.2** (Heronsches Verfahren)**.** Eingabe: *Reelle Zahl $r \geq 0$ und Toleranz $\delta > 0$.*

(1) *Definiere $s_{neu} = 1$ und $s_{alt} = r$.*
(2) *Gilt $|s_{neu} - s_{alt}| < \delta$, so stoppe.*
(3) *Setze $s_{alt} = s_{neu}$ und anschließend $s_{neu} = (s_{alt} + r/s_{alt})/2$.*
(4) *Fahre fort mit Schritt (2).*

Ausgabe: *Approximation* $s_{neu}$ *von* $s = \sqrt{r}$.

In diesem Algorithmus wird dieselbe Anweisung in Schritt (3) so lange wiederholt, bis eine Abbruchbedingung erfüllt ist. Man könnte alternativ auch die Iterationsvorschrift $N$-mal anwenden und $s_N$ als Approximation verwenden, allerdings ist a priori nicht klar, was eine hinreichend große Zahl $N$ ist.

## 4. Wissenschaftliche Fragestellungen

Das wissenschaftliche Fachgebiet der Informatik ist mit der Untersuchung und Verwendung von Rechnern befasst und lässt sich in die Bereiche der technischen, praktischen und theoretischen Informatik untergliedern, deren Arbeitsbereiche sich auszugshaft folgendermaßen beschreiben lassen:
*Technische Informatik:* Die technische Informatik ist zum Beispiel mit der Entwicklung von Prozessoren und Speichermedien und allgemeiner mit der sogenannten *Hardware* befasst.
*Praktische Informatik:* Gegenstand der praktischen Informatik ist beispielsweise die Konzeption von Algorithmen und deren Umsetzung in Programme also die Entwicklung von *Software*.
*Theoretische Informatik:* Die theoretische Informatik untersucht unter anderem, ob gewisse Problemstellungen berechen- beziehungsweise entscheidbar sind, und entwickelt Kalküle für die Verifizierung der Korrektheit von Programmen.