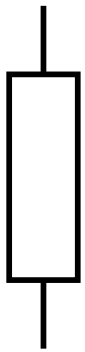
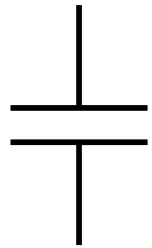
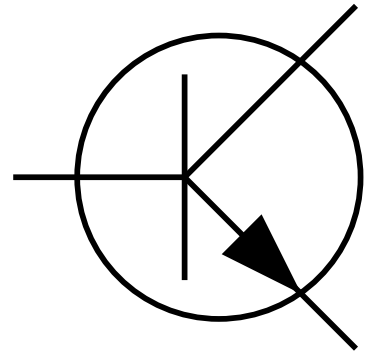
Computergrundlagen
Boolesche Logik, Zahlensysteme und Arithmetik

Axel Arnold

Institut für Computerphysik
Universität Stuttgart

Wintersemester 2011/12

Wie rechnet ein Computer?

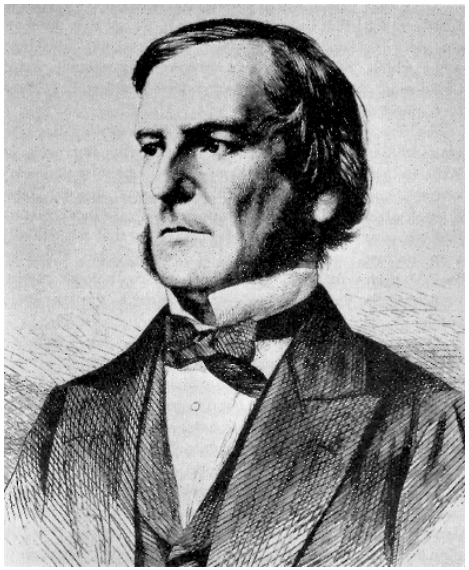




Ein Mikroprozessor

- ist ein Netz von Transistoren, Widerständen und Kondensatoren
- Leitungen kennen nur zwei Zustände: Spannung oder nicht
- Interpretation als ja/nein, 0/1, an/aus, ...
- Schaltungen entsprechen logischen Operationen („Spannung an Pin a und Spannung an Pin b $\Rightarrow$ Spannung am Ausgang“)

Wie kann ich damit rechnen?

Aussagenlogik



G. Boole,
1815 - 1864

- erlaubt formale Beweise über logische Aussagen
- Grundlage der Computerlogik
- kann mathematisch als Algebra formalisiert werden
  – als eine *boolesche Algebra*

Die zweielementige boolesche Algebra

Wir betrachten eine Menge

- mit zwei Elementen 1 („wahr“) und 0 („falsch“)
- mit zwei Verknüpfungen $\vee$ („oder“) und $\wedge$ („und“)
- mit einer einstelligen Verknüpfung $\neg$ („nicht“, Negation)

Ferner gilt für beliebige $a, b, c \in \{0, 1\}$:

1. $a \vee (b \vee c) = (a \vee b) \vee c$,
   $a \wedge (b \wedge c) = (a \wedge b) \wedge c$ (Assoziativität)
2. $a \vee (b \wedge c) = (a \vee b) \wedge (a \vee c)$,
   $a \wedge (b \vee c) = (a \wedge b) \vee (a \wedge c)$ (Distributivität)
3. $a \vee b = b \vee a$, $a \wedge b = b \wedge a$ (Kommutativität)
4. $a \vee (a \wedge b) = a$, $a \wedge (a \vee b) = a$ (Adsorption)
5. $a \vee \neg a = 1$, $a \wedge \neg a = 0$ (Komplemente)

Abgeleitete Gesetze

- Neutralität:
  $a \vee 0 = a, \quad a \wedge 1 = a$
- Idempotenz:
  $a \vee a = a, \quad a \wedge a = a$
- Extremalgesetze:
  $a \vee 1 = 1, \quad a \wedge 0 = 0$
- Doppelnegation: $\neg(\neg a) = a$
- Dualität:
  $\neg 0 = 1, \quad \neg 1 = 0$
- De Morgansche Gesetze:
  $\neg(a \vee b) = \neg a \wedge \neg b, \quad \neg(a \wedge b) = \neg a \vee \neg b$

Die zweielementige boolesche Algebra entspricht genau der Aussagenlogik – und unserem Verständnis von Logik.

Zahlensysteme

- Wie kann ich mit nur zwei Elementen Zahlen darstellen?

Sei $B > 0$ eine natürliche Zahl. Dann kann jede natürliche Zahl z *eindeutig* dargestellt werden als

$$z = \sum_{i=0}^{\infty} B^n z_k,$$

wobei für alle k $0 \leq z_k < B$ und nur endliche viele $z_k \neq 0$.

Beispiel

$B = 10$ entspricht unserem Dezimalsystem:

$$1042 = 10^0 \cdot 2 + 10^1 \cdot 4 + 10^3 \cdot 1 = 1042\text{d}$$

$B = 8$ ergibt das Oktalsystem:

$$1042 = 8^0 \cdot 2 + 8^1 \cdot 2 + 8^3 \cdot 2 = 2022\text{o}$$

Binärsystem

- Wie kann ich mit nur zwei Elementen Zahlen darstellen?

Wir benutzen das *Binärsystem* mit $B = 2$ und Ziffern 0 und 1 (Bits)

Beispiele

$$42 = 2^5 + 2^3 + 2^1 = 101010\text{b}$$

- Umrechnung von Binär- auf Dezimalzahlen ist umständlich
- Oktal ist es einfach:

$$1042 = 2^{10} + 2^4 + 2^1 = \overset{2\,1\;\;4\,2\,1\;\;4\,2\,1\;\;4\,2\,1}{10.000.010.010\text{b}} = 2022\text{o}$$

- oder in *Hexadezimalzahlen* ($B = 16$, Ziffern 1–9, A–F):

$$\overset{4\,2\,1\;\;8\,4\,2\,1\;\;8\,4\,2\,1}{100.0001.0010\text{b}} = 812\text{h}, \quad 1010.1111.1111.1110\text{b} = \text{AFFEh}$$

Addieren/Subtrahieren im Binärsystem

Genau wie im Dezimalsystem:

	101010	(Summand a)
+	1111	(Summand b)
	1110	(Übertrag c)
=	111001	(Ergebnis e)

	101010	(1. Summand a)
-	1111	(2. Summand b)
	11111	(geborgt c)
=	11011	(Ergebnis e)

Betrachten wir die Ziffern als Wahrheitswerte, so gilt:

$$
\begin{aligned}
e_0 &= (a_0 \vee b_0) \wedge \neg(a_0 \wedge b_0) =: a_0 \oplus b_0 \\
c_1 &= a_0 \wedge b_0 \\
e_i &= (a_i \oplus b_i) \oplus c_i \\
c_{i+1} &= (a_i \wedge b_i) \vee (a_i \wedge c_i) \vee (b_i \wedge c_i)
\end{aligned}
$$

Damit kann die Addition als Schleifen-Schaltkreis realisiert werden!

Komplementdarstellung negativer Zahlen

Was ist z.B. -5 = 0 - 5?

```
          0
-       101
    ...11111
-------------
=   ...111011
```

- im Prinzip unendlich viele führende 1er
- in der Praxis endliche Darstellung (z.B. 32 Bit)
- Bei n Bit-Darstellung wird $-z$ als $2^n - z$ dargestellt
- Bei 8 Bit z.B. ist $-5 = 256 - 5 = 251 = 11111011\text{b}$
- Addition und Subtraktion funktionieren ohne Änderung, solange module 2^n gerechnet wird

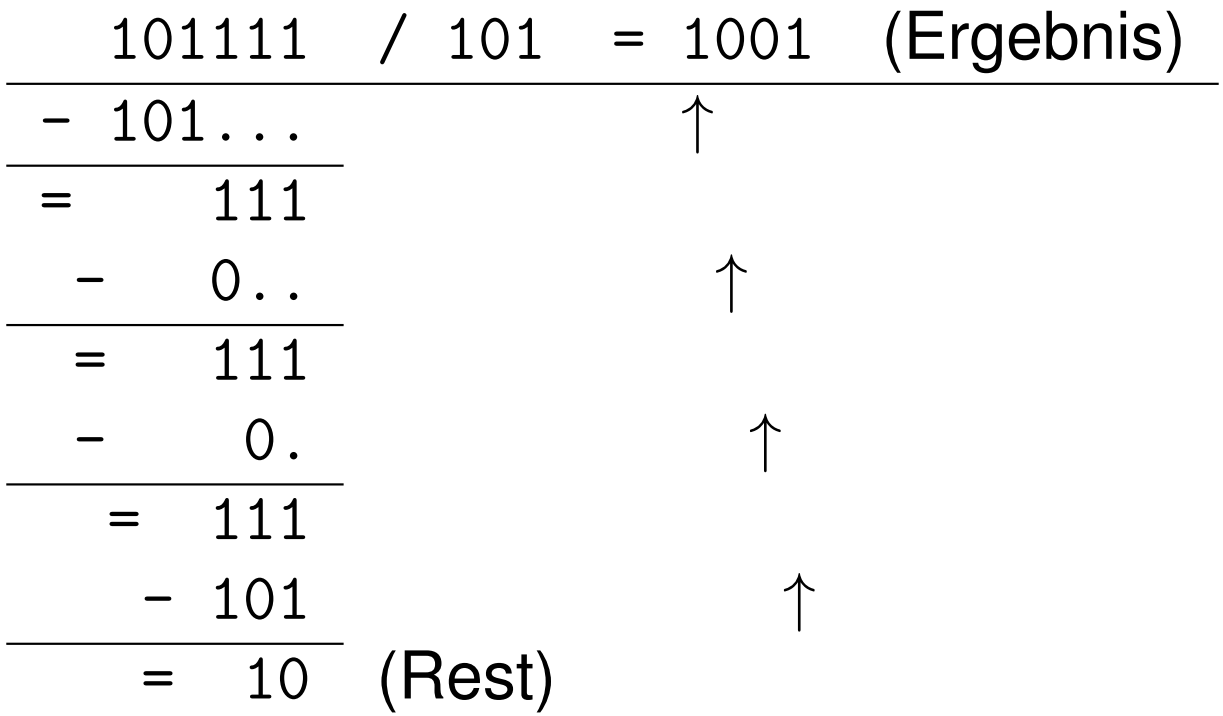
Multiplizieren im Binärsystem


```
      101010 * 1010
-------------------
             0    ↓
    +   101010.   ↓
    +      0..  ↓
    + 101010... ↓
-------------------
 =    110100100
```

Binäre Multiplikation ist sehr einfach:

- Eine Zahl wird bitweise nach links geschoben (Multiplikation mit 2)
- ist das entsprechende Bit der andere Zahl gesetzt, wird die erste addiert, sonst nicht

Dividieren im Binärsystem



Binäre Division analog zur Multiplikation:

- Der Divisor wird zunächst ganz nach links geschoben
- Und dann bitweise wieder nach links
- Wenn der Rest größer als der Divisor ist, abziehen, und das entsprechende Bit des Ergebnisses setzen