

Grundbegriffe der Informatik

Einheit 9: Speicher

Thomas Worsch

Karlsruher Institut für Technologie, Fakultät für Informatik

Wintersemester 2009/2010

Speicher

- Bit und Byte

- Speicher als Tabellen und Abbildungen

- Binäre und dezimale Größenpräfixe

Speicher

Bit und Byte

Speicher als Tabellen und Abbildungen

Binäre und dezimale Größenpräfixe

- ▶ Das Wort „Bit“ hat verschiedene Bedeutungen.
 - ▶ die erste:
Ein **Bit** ist ein Zeichen des Alphabetes $\{0, 1\}$.
 - ▶ die zweite:
in der Vorlesung „Theoretische Grundlagen“ im 3. Semester
- ▶ **Byte**: heute üblicherweise ein Wort aus acht Bits
(früher war das anders)
- ▶ genauer: **Octet**
- ▶ Abkürzungen
 - ▶ für Bit: möglichst „**bit**“, denn „b“ wird schon für die Flächeneinheit „barn“ benutzt
 - ▶ für Byte: meist „**B**“, obwohl das auch schon eine andere Bedeutung hat (Bel)
 - ▶ für Octet: „**o**“

Speicher

Bit und Byte

Speicher als Tabellen und Abbildungen

Binäre und dezimale Größenpräfixe

- ▶ Formalisierung von
 - ▶ Speicher
 - ▶ Lesen aus dem Speicher
 - ▶ Schreiben in den Speicher
- ▶ Diskussion, wozu diese Formalisierungen

- ▶ aktueller Gesamtzustand eines Speichers:
 - ▶ für jede Adresse, zu der etwas gespeichert ist:
 - ▶ welcher Wert ist unter dieser Adresse abgelegt
- ▶ Vorstellung: Tabelle mit zwei Spalten:
 - ▶ links alle Adressen
 - ▶ rechts die zugehörigen Werte

allgemein		Halbleiterspeicher	
Adresse 1	Wert 1	000	10110101
Adresse 2	Wert 2	001	10101101
Adresse 3	Wert 3	010	10011101
Adresse 4	Wert 4	011	01110110
		100	00111110
:	:	101	10101101
		110	00101011
Adresse n	Wert n	111	10101001

- ▶ aktueller Gesamtzustand eines Speichers:
 - ▶ für jede Adresse, zu der etwas gespeichert ist:
 - ▶ welcher Wert ist unter dieser Adresse abgelegt
- ▶ Vorstellung: Tabelle mit zwei Spalten:
 - ▶ links alle Adressen
 - ▶ rechts die zugehörigen Werte

allgemein		Halbleiterspeicher	
Adresse 1	Wert 1	000	10110101
Adresse 2	Wert 2	001	10101101
Adresse 3	Wert 3	010	10011101
Adresse 4	Wert 4	011	01110110
		100	00111110
:	:	101	10101101
		110	00101011
Adresse n	Wert n	111	10101001

- ▶ Tabelle: Abbildung von Adressen auf Werte

$$m : \text{Adr} \rightarrow \text{Val}$$

- ▶ Z. B. Halbleiterspeicher in einem PC mit „4 Gigabyte“:

$$m : \{0, 1\}^{32} \rightarrow \{0, 1\}^8$$

- ▶ Speicher im Zustand m hat an Adresse $a \in \text{Adr}$ den Wert $m(a) \in \text{Val}$ gespeichert.
- ▶ bei Hauptspeicher:
 - ▶ Menge der Adressen ist fest
 - ▶ bezeichnet einen physikalischen Ort auf dem Chip
 - ▶ entspricht einer Angabe wie
„Am Fasanengarten 5, 76131 Karlsruhe“
auf einem Brief

- ▶ Formalisierung als Abbildung `memread`
 - ▶ Argumente:
 - ▶ der gesamte Speicherinhalt m des Speichers und
 - ▶ die Adresse a aus der ausgelesen wird
 - ▶ Resultat ist der in m an Adresse a gespeicherte Wert. Also:

$$\begin{aligned}\text{memread} : \text{Mem} \times \text{Adr} &\rightarrow \text{Val} \\ (m, a) &\mapsto m(a)\end{aligned}$$

- ▶ Dabei sei Mem die Menge aller möglichen Speicherzustände, also die Menge aller Abbildungen von Adr nach Val .
- ▶ auf das „Warum überhaupt so eine Formalisierung?“ kommen wir noch zu sprechen

- ▶ nicht verwirren lassen:
 - ▶ Funktion `memread` bekommt als Argument eine (andere) Funktion m
 - ▶ Beispiel zeigt: kein Problem; man denke an Tabellen
- ▶ die Menge aller Abbildungen der Form $f : A \rightarrow B$
 - ▶ so etwas kommt noch öfter vor
 - ▶ Notation: B^A
 - ▶ beachte Reihenfolge
 - ▶ für endliche Mengen A und B gilt: $|B^A| = |B|^{|A|}$.
- ▶ hätten also auch schreiben können:
 - ▶ $\text{Mem} = \text{Val}^{\text{Adr}}$ und
 - ▶ $\text{memread} : \text{Val}^{\text{Adr}} \times \text{Adr} \rightarrow \text{Val}$
 $(m, a) \mapsto m(a)$

sieht ein wenig komplizierter aus als beim Lesen:

- ▶ $\text{memwrite} : \text{Val}^{\text{Adr}} \times \text{Adr} \times \text{Val} \rightarrow \text{Val}^{\text{Adr}}$

$$(m, a, v) \mapsto m'$$

- ▶ wobei m' festgelegt durch die Forderung, dass für alle $a' \in \text{Adr}$ gilt:

$$m'(a') = \begin{cases} v & \text{falls } a' = a \\ m(a') & \text{falls } a' \neq a \end{cases}$$

- ▶ Das klingt plausibel ...
 - ▶ für Hauptspeicher ist es das auch,
 - ▶ aber es gibt auch Speicher, die anders arbeiten, z. B. sogenannte Cache-Speicher (siehe Vorlesungen über Technische Informatik)

- ▶ Was ist „das wesentliche“ an Speicher?
- ▶ Der allerwichtigste Aspekt überhaupt:
Für alle $m \in \text{Mem}$, $a, a' \in \text{Adr}$ mit $a' \neq a$ und $v' \in \text{Val}$ gilt:

$$\text{memread}(\text{memwrite}(m, a, v), a) = v$$

- ▶ reicht das als Spezifikation von Speicher?
- ▶ Für die oben definierten Funktionen gilt auch:
Für alle $m \in \text{Mem}$, $a, a' \in \text{Adr}$ mit $a' \neq a$ und $v' \in \text{Val}$ gilt:

$$\text{memread}(m, a) = \text{memread}(\text{memwrite}(m, a', v'), a)$$

Wozu diese Formalisierungen?

- ▶ eine Möglichkeit für den *Spezifizierer*, zu sagen
 - ▶ „wie sich Speicher verhalten soll“
 - ▶ algebraische Spezifikation
- ▶ auch eine Möglichkeit für den *Implementierer*, sich über Testfälle klar zu werden
 - ▶ Testen kann nicht Korrektheit einer Implementierung beweisen,
 - ▶ aber immerhin, dass sie falsch ist.

Das sollten Sie mitnehmen:

- ▶ es gibt nicht nur Hauptspeicher
- ▶ Adressen sind manchmal etwas anderes als „physikalische Koordinaten“
- ▶ Abbildungen kann man sich manchmal am besten als Tabelle vorstellen

Das sollten Sie üben:

- ▶ Gewöhnung an Abbildungen, die Abbildungen auf Abbildungen abbilden

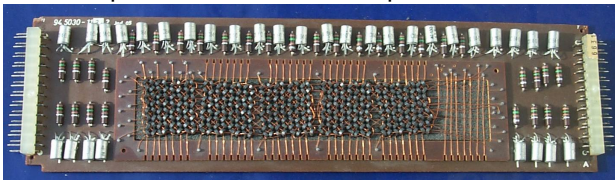
Speicher

Bit und Byte

Speicher als Tabellen und Abbildungen

Binäre und dezimale Größenpräfixe

- ▶ früher: Speicher klein, z. B. ein paar Hundert Bits



Bildquelle: <http://de.wikipedia.org/w/index.php?title=Bild:Kernspeicher1.jpg>

- ▶ heute: groß, z. B.
 - ▶ Hauptspeicher: $2^{32} = 4\,294\,967\,296$ Bytes
 - ▶ Festplatten: so was wie 1 000 000 000 000 Bytes
- ▶ Zahlen nur noch schlecht zu lesen
- ▶ Benutzung von **Präfixen** für kompaktere Notation:
Kilometer, **Mikrosekunde**, **Megawatt**, usw.

10^{-3}	10^{-6}	10^{-9}	10^{-12}	10^{-15}	10^{-18}
1000^{-1}	1000^{-2}	1000^{-3}	1000^{-4}	1000^{-5}	1000^{-6}
milli	mikro	nano	pico	femto	atto
m	μ	n	p	f	a
10^3	10^6	10^9	10^{12}	10^{15}	10^{18}
1000^1	1000^2	1000^3	1000^4	1000^5	1000^6
kilo	mega	giga	tera	peta	exa
k	M	G	T	P	E

- ▶ In Rechnern häufig Größen, die Potenzen von 2 oder 2^{10} sind, und *nicht* Potenzen von 10 bzw. 1000.
- ▶ Die *International Electrotechnical Commission* hat 1999 Präfixe eingeführt, die für Potenzen von $2^{10} = 1024$ stehen.
- ▶ motiviert durch Kunstworte „kilobinary“, „megabinary“, usw.
- ▶ Präfixe *kibi*, *mebi*, *gibi*, usw.
- ▶ abgekürzt Ki, Mi, Gi, usw.

2^{10}	2^{20}	2^{30}	2^{40}	2^{50}	2^{60}
1024^1	1024^2	1024^3	1024^4	1024^5	1024^6
kibi	mebi	gibi	tebi	pebi	exbi
Ki	Mi	Gi	Ti	Pi	Ei

Das sollten Sie mitnehmen:

- ▶ Bit und Byte / Octet
- ▶ binäre Größenpräfixe

Das sollten Sie üben:

- ▶ Rechnen mit binären Größenpräfixen