Grundbegriffe der Informatik

Einheit 9: Speicher

Thomas Worsch

Karlsruher Institut für Technologie, Fakultät für Informatik

Wintersemester 2009/2010

Überblick

Speicher

- Bit und Byte
- Speicher als Tabellen und Abbildungen
- Binäre und dezimale Größenpräfixe

Überblick

Speicher

- Bit und Byte
- Speicher als Tabellen und Abbildungen
- Binäre und dezimale Größenpräfixe

Überblick

Bit und Byte

- 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“

Überblick

im folgenden

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

Gesamtzustand eines Speichers

- 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	
Adresse 1	Wert 1
Adresse 2	Wert 2
Adresse 3	Wert 3
Adresse 4	Wert 4
⋮	⋮
Adresse n	Wert n

Halbleiterspeicher	
000	10110101
001	10101101
010	10011101
011	01110110
100	00111110
101	10101101
110	00101011
111	10101001

Gesamtzustand eines Speichers

- 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	
Adresse 1	Wert 1
Adresse 2	Wert 2
Adresse 3	Wert 3
Adresse 4	Wert 4
⋮	⋮
Adresse n	Wert n

Halbleiterspeicher	
000	10110101
001	10101101
010	10011101
011	01110110
100	00111110
101	10101101
110	00101011
111	10101001

Formalisierung von Speicher

- Tabelle: Abbildung von Adressen auf Werte

$$m : \mathrm{Adr} \to \mathrm{Val}$$

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

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

- Speicher im Zustand m hat an Adresse $a \in \mathrm{Adr}$ den Wert $m(a) \in \mathrm{Val}$ 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 von Lesen aus dem Speicher

- Formalisierung als Abbildung $\mathrm{memread}$
  - 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:

$$\mathrm{memread} : \mathrm{Mem} \times \mathrm{Adr} \to \mathrm{Val}$$
$$(m, a) \mapsto m(a)$$

- 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

Bemerkungen zu $\mathrm{memread}$

- nicht verwirren lassen:
  - Funktion $\mathrm{memread}$ bekommt als Argument eine (andere) Funktion m
  - Beispiel zeigt: kein Problem; man denke an Tabellen
- die Menge aller Abbildungen der Form $f : A \to B$
  - 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:
  - $\mathrm{Mem} = \mathrm{Val}^{\mathrm{Adr}}$ und
  - $$\mathrm{memread} : \mathrm{Val}^{\mathrm{Adr}} \times \mathrm{Adr} \to \mathrm{Val}$$
    $$(m, a) \mapsto m(a)$$

Formalisierung von Schreiben in den Speicher

sieht ein wenig komplizierter aus als beim Lesen:

- $\mathrm{memwrite} : \mathrm{Val}^{\mathrm{Adr}} \times \mathrm{Adr} \times \mathrm{Val} \to \mathrm{Val}^{\mathrm{Adr}}$

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

  - wobei m' festgelegt durch die Forderung, dass für alle $a' \in \mathrm{Adr}$ 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)

Eigenschaften von Speicher

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

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

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

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

Wozu diese Formalisierungen?

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

Was ist wichtig

Das sollten Sie mitnehmen:

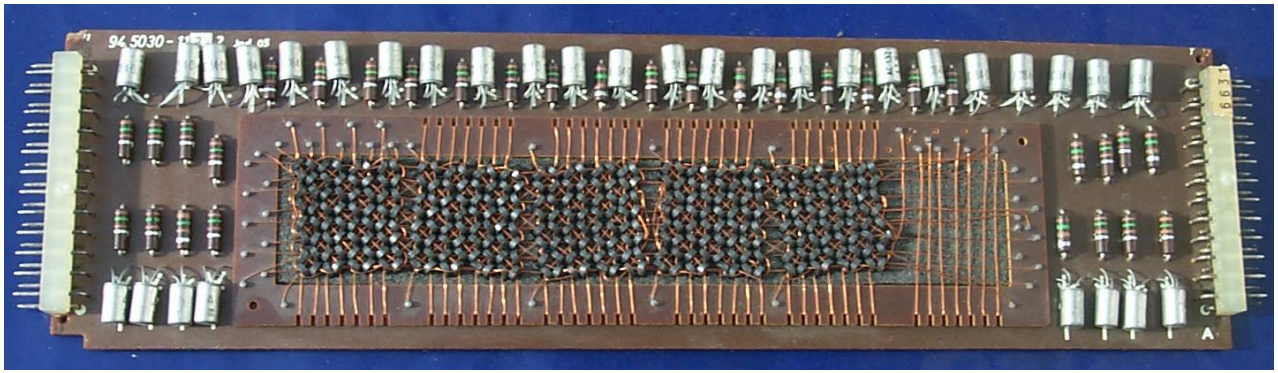
- es gibt nicht nur Hauptspeicher
- Adressen sind manchmal etwas anderes als „physikalische Koordinaten“
- Abbildungen kann man sich manchmal am besten als Tabelle vorstellen

Das sollten Sie üben:

- Gewöhung an Abbildungen, die Abbildungen auf Abbildungen abbilden

Klein und groß

- 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.

Dezimale Größenpräfixe

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

Binäre Größenpräfixe

- 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

Wir halten fest

Das sollten Sie mitnehmen:

- Bit und Byte / Octet
- binäre Größenpräfixe

Das sollten Sie üben:

- Rechnen mit binären Größenpräfixen