Grundbegriffe der Informatik

Einheit 9: Speicher

Thomas Worsch

KIT, Institut für Theoretische Informatik

Wintersemester 2015/2016

Überblick

Bit und Byte

Binäre und dezimale Größenpräfixe

Speicher als Tabellen und Abbildungen

Wo sind wir?

Bit und Byte

Binäre und dezimale Größenpräfixe

Speicher als Tabellen und Abbildungen

Bit und Byte

Das Wort «Bit» hat verschiedene Bedeutungen.

1. Ein Bit ist ein Zeichen des Alphabetes $\{0, 1\}$.
2. siehe Vorlesung «Theoretische Grundlagen» im 3. Sem.

Byte

- heute üblicherweise ein Wort aus acht Bits
- früher war das anders

genauer: Octet

Abkürzungen

- für Bit: «bit», («b» für Flächeneinheit «barn» benutzt)
- für Byte: «B» (obwohl auch schon andere Bedeutung hat (Bel)
- für Octet: «o»

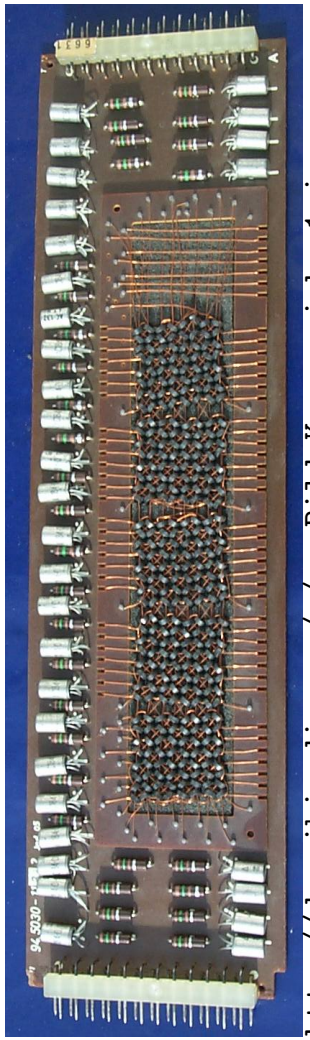
Wo sind wir?

Bit und Byte

Binäre und dezimale Größenpräfixe

Speicher als Tabellen und Abbildungen

Kleiner und großer Speicher



http://de.wikipedia.org/w/...Bild:Kernspeicher1.jpg

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

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

meine Lieblingslängeneinheit: 1 attoparsec

- Wie lang ist das?

Binäre Größenpräfixe

In Rechnern häufig Potenzen von 2 oder 2^{10}

- *nicht* Potenzen von 10 bzw. 1000

Präfixe für Potenzen von $2^{10} = 1024$

- *International Electrotechnical Commission* 1999

motiviert durch «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

Wo sind wir?

Bit und Byte

Binäre und dezimale Größenpräfixe

Speicher als Tabellen und Abbildungen

Formalisierungen sind Spezifikationen — auch im Zusammenhang mit Speicher …

was wird formalisiert?

- Speicher
- Lesen aus Speicher
- Schreiben in Speicher

wozu diese Spezifikationen?

- gut für Tester
- man findet Lücken
- Semantik von Programmiersprachen

- Sie üben funktionales Denken ;-)

Gesamtzustand eines Speichers

zu jedem Zeitpunkt

- für jede *Adresse*
- welcher *Wert* ist dort

Gesamtzustand eines Speichers

zu jedem Zeitpunkt

- für jede *Adresse*
- welcher *Wert* ist dort

allgemein	
Adresse 1	Wert 1
Adresse 2	Wert 2
Adresse 3	Wert 3
⋮	⋮
Adresse n	Wert n

Vorstellung: Tabelle mit zwei Spalten:

- links alle Adressen
- rechts die zugehörigen Werte

Gesamtzustand eines Speichers

zu jedem Zeitpunkt

- für jede *Adresse*
- welcher *Wert* ist dort

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

Vorstellung: Tabelle mit zwei Spalten:

- links alle Adressen
- rechts die zugehörigen Werte
- beides oft Bitfolgen

Formalisierung von Speicher

Tabelle: Abbildung von Adressen auf Werte

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

Halbleiterspeicher in PC mit 4 Gibibyte

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

- in Speicherzustand $m : \text{Adr} \to \text{Val}$
- an Adresse $a \in \text{Adr}$
- 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»

Lesen aus dem Speicher — 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:

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

Mem

- Menge aller möglichen Speicherzustände, hier also
- Menge aller Abbildungen von Adr nach Val

Bemerkungen zu memread

Funktion memread bekommt als Argument eine Funktion m

- kein Problem; man denke an Tabellen

Menge aller Abbildungen der Form $f : A \to B$

- Notation: B^A
- beachte Reihenfolge
- für endliche Mengen gilt: $|B^A| = |B|^{|A|}$.

hätten also auch schreiben können:

- $\mathsf{Mem} = \mathsf{Val}^{\mathsf{Adr}}$ und

- $\mathsf{memread} : \mathsf{Val}^{\mathsf{Adr}} \times \mathsf{Adr} \to \mathsf{Val}$

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

Schreiben in den Speicher — ein wenig komplizierter

Schreiben in den Speicher — ein wenig komplizierter

$$\text{memwrite} : \text{Val}^{\text{Adr}} \times \text{Adr} \times \text{Val} \to \text{Val}^{\text{Adr}}$$
$$(m, a, v) \mapsto m'$$

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

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

Schreiben in den Speicher — ein wenig komplizierter

$$\text{memwrite} : \text{Val}^{\text{Adr}} \times \text{Adr} \times \text{Val} \to \text{Val}^{\text{Adr}}$$
$$(m, a, v) \mapsto m'$$

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

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

- hätten kürzer schreiben können

$$\text{memwrite} : \text{Val}^{\text{Adr}} \times \text{Adr} \times \text{Val} \to \text{Val}^{\text{Adr}}$$
$$(m, a, v) \mapsto \left(a' \mapsto \begin{cases} v & \text{falls } a' = a \\ m(a') & \text{falls } a' \neq a \end{cases} \right)$$

Schreiben in den Speicher — ein wenig komplizierter

$$\text{memwrite} : \text{Val}^{\text{Adr}} \times \text{Adr} \times \text{Val} \to \text{Val}^{\text{Adr}}$$
$$(m, a, v) \mapsto m'$$

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

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

- plausibel … (?) …
  - für Hauptspeicher ist es das
  - aber es gibt auch Speicher, die anders arbeiten, z. B. sogenannte Caches (siehe Technische Informatik)

Eigenschaften von Speicher

Eigenschaften von Speicher

Was ist «das Wesentliche» an Speicher?

Für jedes $m \in \text{Mem}$, $a \in \text{Adr}$, $v \in \text{Val}$ gilt:

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

Reicht das als Spezifikation von Speicher?

Eigenschaften von Speicher

Was ist «das Wesentliche» an Speicher?

Für jedes $m \in \text{Mem}$, $a \in \text{Adr}$, $v \in \text{Val}$ 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:

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

Will man das wirklich immer … ? …

Wozu diese Formalisierungen?

eine Möglichkeit für den *Spezifizierer*, zu sagen

- «wie sich Speicher verhalten soll»
- algebraische Spezifikation

eine Möglichkeit für den *Implementierer*,

- sich über Testfälle klar zu werden
- das kann nicht Korrektheit einer Implementierung beweisen,
- aber immerhin, dass sie falsch ist.

eine mögliche Grundlage für die Festlegung der *Bedeutung* von Programmen

Was ist wichtig

Das sollten Sie mitnehmen:

- es gibt nicht nur Hauptspeicher
- Adressen sind manchmal etwas anderes als «physikalische Koordinaten»
- Abbildungen kann man sich gut als Tabelle vorstellen

Das sollten Sie üben:

- Abbildungen, die Abbildungen auf Abbildungen abbilden