Übungen zur Vorlesung
„Grundlagen der Informatik 2“
Studiengang E-Technik

AG Betriebssysteme, FB 3
Jan Bredereke

SoSe 2002

Übung 3 am 2.5.2002

Wegen des Feiertages Himmelfahrt ist die nächste Übung erst am 16.5.2002. Dann ist auch die Abgabe des Blattes 1.

Die Zulassungskriterien zur Klausur sind geklärt: Alle müssen wie im letzten Semester an einem kurzen Fachgespräch teilnehmen, oder sie müssen erfolgreich vorrechnen.

Zum Vorrechnen kann Aufgabe 2 auf Blatt 1 (der Beweis) gerne auf zwei Leute aufgeteilt werden.

Eigenschaften von Darstellungen ganzer Zahlen

Vorzeichenlose ($\rho(x)$), Zweierkomplement- ($\widetilde{\rho}(x)$), Einerkomplement- ($\widehat{\rho}(x)$) und Vorzeichen/Betrag-Interpretation ($\rho_{\mathrm{v}}(x)$) des Bitmusters:

x_2	x_1	x_0	$\rho(x)$	$\widetilde{\rho}(x)$	$\widehat{\rho}(x)$	$\rho_{\mathrm{v}}(x)$
1	0	0	4	-4	-3	0
1	0	1	5	-3	-2	-1
1	1	0	6	-2	-1	-2
1	1	1	7	-1	0	-3
0	0	0	0	0	0	0
0	0	1	1	1	1	1
0	1	0	2	2	2	2
0	1	1	3	3	3	3
1	0	0	4	-4	-3	0
1	0	1	5	-3	-2	-1
1	1	0	6	-2	-1	-2
1	1	1	7	-1	0	-3
...			...	...	...	...

Eine Addition von 2 bedeutet zwei Schritte nach unten, eine Subtraktion von 2 zwei Schritte nach oben. Dies gilt in der vorzeichenlosen und in der Zweierkomplement-Interpretation. Es gilt nicht in der Vorzeichen/Betrag-Interpretation, die Schrittrichtung hängt dort vom Vorzeichen ab.

Bei den ersten zwei Interpretationen muß man aufpassen, daß man keinen Überlauf 7/0 bzw. 3/-4 bekommt.

Bei der Einerkomplement- und der Vorzeichen/Betrag-Interpretation muß man auf zwei Arten von Überläufen aufpassen: 3/-3 wie bisher, und zusätzlich der Vorzeichenwechsel bei -0/0.

Die erste Art von Überlauf kann man vermeiden, wenn man nur genügend lange Bitvektoren nimmt. Die zweite Art kann man nicht vermeiden, da man häufig mit „kleinen“ Zahlen um die Null herum rechnen muß. Daher sind die letzten beiden Zahlendarstellungen für die Addition und die Subtraktion mit „normalen“ Zahlen nicht so gut geeignet.

Für die Multiplikation und die Division ist dagegen die Vorzeichen/Betrag- (und natürlich die vorzeichenlose) Interpretation besonders geeignet: Der Betrag des Ergebnisses ergibt sich in diesem Falle durch die Multiplikation (bzw. Division) der Beträge der Operanden, und das Vorzeichen des Ergebnisses läßt sich direkt aus den Vorzeichen der Operanden berechnen, ganz ohne Bezug auf die Beträge. Die Zweier- und die Einerkomplementinterpretation benötigen für die Multiplikation und die Division erst eine Umwandlung in die Vorzeichen/Betrag-Interpretation, oder aber zusätzliche Hardware.

Darstellung von Gleitkommazahlen

Eine ausführliche Beschreibung findet sich im Buch von Oberschelp/Vossen in Kapitel 5.2. Die folgende Beschreibung ist dieser Quelle entnommen.

- Nicht-ganze Zahlen: als Dezimalbruch geschrieben.
  42,57
- Festkomma-Darstellung:
  42,57
  1,50
  20,00
  0,00
- Interpretation von $(x_{n-1} \ \ldots \ x_1 \ x_0 \ x_{-1} \ \ldots \ x_{-m+1} \ x_{-m})$ als Festkommazahl:

  $$x = \sum_{i=-m}^{n-1} 2^i x_i$$

- Negative Zahlen: Ein Bit für das Vorzeichen reservieren oder eine der Komplementdarstellungen.
- Für Addition und Subtraktion ist wichtig, daß alle Operanden das Komma an der gleichen Stelle stehen haben. Anderenfalls muß einer der Operanden transformiert werden, indem die Bits geeignet verschoben werden.
- Gleitkomma-Darstellung:

  $$r = m \cdot 2^d$$

  Dabei ist m in Festkommadarstellung und $d \in \mathbb{Z}$.

  m heißt *Mantisse* und d heißt *Exponent*.
- Beispiele:
  $42{,}57 \cdot 2^0$
  $1{,}50 \cdot 2^{63}$
  $20{,}00 \cdot 2^{-51}$
  $0{,}00 \cdot 2^0$
- Vorteil: Viel größerer Wertebereich als bei ganzen Zahlen.

- Achtung: Darstellung ist nicht eindeutig. Verschiedene feste Komma-Positionen sind denkbar. Daher *normalisiert* man die Darstellung, indem man dafür sorgt, daß gilt:

  $$\frac{1}{2} \leq |m| < 1$$

  Das heißt, daß das Komma unmittelbar links von der linkesten Nicht-Null-Stelle der Mantisse steht:
  $0,5107 \cdot 2^7$

- Darstellung einer Gleitkommazahl als Bitvektor:
  Bei 32-Bit z.B.:
  - 1 Bit als Vorzeichen der Zahl
  - 23 Bits für die Mantisse, normalisiert
  - 8 Bits für den Exponenten

- Darstellbarer Wertebereich:
  $0,5 \cdot 2^{-128} \leq r \leq (1 - 2^{-23}) \cdot 2^{127}$
  und
  $-(1 - 2^{-23}) \cdot 2^{127} \leq r \leq -0,5 \cdot 2^{-128}$

- Um die Null herum ist ein „Loch“ im Wertebereich, das auch die Null selbst umfaßt.

  Konvention: Positives Vorzeichen und Exponent = 0 sowie eine beliebige Mantisse bedeuten „Null“.

- Beobachtung: Das höchstwertige Bit der Mantisse ist immer 1, außer bei der Null.

  Nachdem wir diesen Sonderfall separat behandelt haben, brauchen wir daher dieses 1-Bit gar nicht mehr abzuspeichern und haben ein Bit mehr Speicherplatz zur Verfügung, was die Genauigkeit erhöht.

- Wie hoch ist die Genauigkeit?
  Wir haben nur 23 Bits für die Mantisse, das entspricht etwa sieben Dezimalstellen.

  Dies ist die Genauigkeit, die wir von `float`-Variablen in C erwarten können. `double`-Variablen haben doppelt so viele Bits, nämlich 64 Bits insgesamt.

- Darstellung des Exponenten als Bitvektor:
  Häufig in Excess-Darstellung, einer weiteren Variante zur Darstellung von vorzeichenbehafteten Zahlen.

  Bei 8 Bits: Excess-128-Darstellung
  $\rho_x(0\ 0\ \ldots\ 0\ 0) = -128$
  $\rho_x(0\ 0\ \ldots\ 0\ 1) = -127$
  $\rho_x(0\ 1\ \ldots\ 1\ 1) = -1$
  $\rho_x(1\ 0\ \ldots\ 0\ 0) = 0$
  $\rho_x(1\ 0\ \ldots\ 0\ 1) = 1$
  $\rho_x(1\ 1\ \ldots\ 1\ 1) = 127$

  Entspricht der Zweierkomplementdarstellung mit invertiertem Vorzeichen.

  Vorteil: leichterer Größenvergleich von Exponenten.

- Der darstellbare Wertebereich schrumpft durch die besondere Definition der „Null“ auf:
  $0,5 \cdot 2^{\mathbf{-127}} \leq r \leq (1 - 2^{-23}) \cdot 2^{127}$ und $-(1 - 2^{-23}) \cdot 2^{127} \leq r \leq -0,5 \cdot 2^{-128}$
  Man kann auch sagen, daß die kleinsten darstellbaren positiven Zahlen uminterpretiert werden als „Null“.