Übungen zur Vorlesung
„Technische Informatik I“, SS 2002
Hauck / Guenkova-Luy / Prager / Chen
Übungblatt 4 - „Rechnerarithmetik“

Aufgabe 1

a) Bestimmen Sie die Darstellung der Zahl 113_{10} zur Basis 7.

b) Wenden Sie die Berechnungsmethode „sukzessive Division mit Rest“ in aller Ausführlichkeit auf eine beliebige Zahl $z = z_2 \cdot 7^2 + z_1 \cdot 7^1 + z_0 \cdot 7^0$ an. Berechnen Sie die Darstellung zur Basis 7 in Abhängigkeit von z_2, z_1, z_0.

c) Bestimmen Sie Darstellungen der Zahl 113_{10} im Hexadezimalsystem (Basis $b = 16$), im Binärsystem (Basis $b = 2$), im Oktalsystem (Basis $b = 8$) sowie im 12er-System. Wie kann die Umwandlung zwischen den Darstellungen der Basen $2, 8, 16$ vereinfacht werden?

d) Bestimmen Sie die 6-stellige 2-er-Komplement Binärdarstellung der Zahl -17_{10}. Wie sieht die 8-stellige Darstellung dieser Zahl aus?

e) Bestimmen Sie die Binärdarstellung der Zahlen 0.375_{10} und 0.1_{10}. Wieviele Nachkommastellen werden zur exakten Darstellung jeweils benötigt?

f) Wie lautet die 2-er-Komplement Binärdarstellung der Zahl -4.375_{10} bei Verwendung von $k = 3$ Vorkommastellen und $m = 4$ Nachkommastellen?

Aufgabe 2

Forscher des SETI-Projekts haben aus den Tiefen des Universums eine Botschaft intelligenten Lebens empfangen. Die Wesen haben offensichtlich 3 Hände mit jeweils 5, 2 und 5 Fingern pro Hand. Zur Darstellung nichtnegativer ganzer Zahlen $\{0, 1, ..\}$ verwenden Sie die positionale Notation $(z_2, z_1, z_0)_\Psi$, welche sich hervorragend zum Zählen mit den Fingern eignet. Die Forscher kamen zu dem Schluß, daß gilt:

$$z = z_0 + 6 \cdot z_1 + b_2 \cdot z_2 \quad \text{mit} \quad z_0, z_2 \in \{0, ..5\}, z_1 \in \{0, 1, 2\},$$

Allein die Zahl b_2 vermochten Sie nicht zu bestimmen.

a) Wie wird die Zahl b_2 von den Wesen sinnvollerweise gewählt worden sein, um mit ihren Fingern einem möglichst großen, lückenlosen Wertebereich zählen zu können?

b) Welchen Zahlenbereich kann man mit dieser Kodierung abdecken?

c) Berechnen Sie $(z_0, z_1, z_2)_\Psi = 315_\Psi + 113_\Psi$. Eignet sich diese Zahlendarstellung zur Berechnung von Summen?

Aufgabe 3

Gegeben sei ein 3x3-Bit Multiplizierer für vorzeichenlose Zahlen. Entwerfen Sie eine Schaltung, die unter Zuhilfenahme dieses Multiplizierers sowie zusätzlichen Gattern vorzeichenbehaftete 4-Bit Zahlen multiplizieren kann. Sowohl Eingabe als auch Ausgabe sollen im 1-er Komplement dargestellt sein. Welche Wortbreite sollten Sie für die Ausgabe mindestens vorsehen, um Fehler durch Überläufe zu vermeiden?

Aufgabe 4

Eine IEEE 754 32-Bit Gleitkommazahl hat folgendes Format: $x = (-1)^s \cdot 1.f \cdot 2^{e-127}$ wobei die einzelnen Bits $(b_{31}..b_0)$ in einem Rechner in folgender Reihenfolge abgelegt werden: $(s, e_7, ..e_0, f_{-1}, ..f_{-23})$. Details zu diesem Format finden Sie auf den Vorlesungsfolien.

a) Welche Zahlen werden durch die Bitmuster 0|10000101|00000010000000000000000 und 1|10000100|10010000000000000000000 dargestellt? (Die senkrechten Striche dienen hierbei lediglich der Übersicht.)

b) Bestimmen Sie die Gleitkommadarstellung der Zahlen 1, -5 und $6.25 \cdot 10^{-3}$.

c) Warum wird zur Darstellung der Zahl 0 eine Ausnahmeregel benötigt?

d) Warum ist es in Programmiersprachen möglich, die Zahl $1 \cdot 10^{-41}$ erfolgreich in einer 32-Bit Gleitkommazahl abzulegen, wohingegen das Zuweisen von $1 \cdot 10^{41}$ vom Übersetzer zurückgewiesen wird?

e) Berechnen Sie die Summe der Zahlen aus Aufgabe a). Beachten Sie dabei, daß die Eingaben sowie das Ergebnis in normalisierter Form vorliegen.

f) Mit welchen der 5 Ausnahmezustände ($\pm\infty$, ± 0, NaN) würden Sie die Ergebnise der Rechnungen $\sqrt{-13}$, $\log_{10}(0)$ und $\log_{10}(-1)$ codieren?

Aufgabe 5

Die Programmiersprache Java kennt 2 (primitive) Typen von Gleitkommazahlen gemäß dem IEEE 754-Standard: `float` („Single Precision“) und `double` („Double Precision“). Ein verzweifelter Programmierer legt Ihnen folgendes Fragment eines Java-Programms vor, welches die Werte einer Funktion `some_function()` in einem Zahlenbereich aufsummieren soll:

```
float integrate() {
  float sum = 0;
  for (float x = 20000000; x < 20000010; x = x + 1) {
    sum = sum + some_function(x);
  }
  return sum;
}
```

Der Programmierer beklagt sich, daß sich das Programm grundsätzlich bei einem Aufruf der Methode `integrate` „aufhängt“, d.h. in eine Endlosschleife gerät. Können Sie ihm weiterhelfen?

a) Bestimmen Sie die kleinste natürliche Zahl l_{float}, die im IEEE Single Precision Format **nicht** darstellbar ist.

b) Warum funktioniert das Programmfragment nicht wie erwartet?

c) Wie können Sie obiges Programm modifizieren, damit es die gewünschte Funktionalität bietet? Innerhalb welcher Grenzen für die Laufvariable `x` funktioniert das Programm nun korrekt?