Westfälische Wilhelms-Universität Münster

Informatik 4 (Rechnerstrukturen)
Sommersemester 2012 - D. Lammers, D. Meiländer

Blatt 4 – Abgabe: 07.05.2012, 14 Uhr

Hinweis: *Lösungen bitte auf Papier in Briefkasten 65-67 abgeben. Programmcode bitte zusätzlich per Mail an den Übungsleiter senden. Abgabe in Gruppen mit 2 Teilnehmern erwünscht. Die Zusammensetzung der Gruppen soll für die Bearbeitung der Übungsaufgaben beibehalten werden.*

Aufgabe 1 (3 + 3 + 3 + 1 Punkte):

Angenommen ein Computer stelle Gleitkomma-Zahlen wie folgt dar:

(Vorzeichen, Mantisse, Exponent)

Vorzeichen (der Mantisse): 1 Bit;
Mantisse: 10 Bits (ohne Vorzeichen), Vorzeichenbit-Darstellung, normalisiert (d.h. $\frac{1}{b} \leq m < 1$, für Basis b und Mantisse m);
Exponent: 5 Bits, Zweier-Komplement-Darstellung;
Basis: 8.

Geben Sie die interne Darstellung folgender Dezimalzahlen in hexadezimaler Notation an:

a) +12.25

b) -0.55

c) -0.001

d) 0.0

Aufgabe 2 (2 + 2 + 2 + 2 + 2 Punkte): Erläutern Sie für jede der folgenden IEEE Single Gleitkommazahlen, um welche Zahlenart es sich handelt (normalisiert/denormalisiert/unendlich/Null/NaN). Wenn es sich um eine endliche Zahl handelt, geben Sie den numerischen Wert an.

a) 0111 1111 1000 1111 0000 1111 0000 0000

b) 0000 0000 0000 0000 0000 0000 0000 0000

c) 0100 0010 0100 0000 0000 0000 0000 0000

d) 1000 0000 0100 0000 0000 0000 0000 0000

e) 1111 1111 1000 0000 0000 0000 0000 0000

Aufgabe 3 (12 Punkte): Implementieren Sie die folgende Java-Klasse:

```
public class IEEE {
 /*
  * Konvertiert x in einen Bitstring der Länge 32 mit der
  * normalisierten Darstellung von x entsprechend IEEE 754.
  */
 public static String ieeeFromFloat(float x) { ... }

 /*
  * Konvertiert einen String s mit der internen Bitdarstellung
  * einer normalisierten float-Zahl x in diese Zahl.
  * Bei nicht konvertierbarer Eingabe wird eine Exception erzeugt.
  */
 public static float floatFromIeee(String s) throws Exception { ... }
}
```

Aufgabe 4 (8 Punkte): Die C-Funktion `inplace_swap` dient zum Vertauschen von zwei `int`-Variablen ohne den sonst notwendigen zusätzlichen Speicherplatz:

```
void inplace_swap (int *x, int *y) {
 *x = *x ^ *y; /* Schritt 1 */
 *y = *x ^ *y; /* Schritt 2 */
 *x = *x ^ *y; /* Schritt 3 */
}
```

Der binäre Operator `^` ist wie in Java das bitweise XOR. Begründen Sie die Korrektheit der Funktion, indem Sie den Wert von `*x` und `*y` nach jedem Schritt angeben. Schreiben Sie eine `main`-Funktion dazu, übersetzen und testen Sie das Programm.