64-041 Übung Rechnerstrukturen und Betriebssysteme WS26/27



Aufgabenblatt 5 Ausgabe: 31.10., Abgabe bis 11.11.2026 24:00

Gruppe	
Name(n)	Matrikelnummer(n)

Aufgabe 5.1 (Punkte 5+5+5)

Gleitkommazahlen: Normalisieren Sie die folgenden Gleitkommazahlen unter Beibehaltung der jeweiligen Basis, wobei die erste von Null verschiedene Stelle der Mantisse unmittelbar links vom Komma stehen soll (in den Klammern ist jeweils die Mantisse, dann der Exponent angegeben):

(a) $(169,31 \mid 3)_{10}$

(b) $(-1\,1011,00 \mid -110)_2$

(c) $(-0,00CA3 \mid 12)_{16}$

Aufgabe 5.2 (Punkte 5+5)

IEEE 754 Gleitkommazahlen: Geben Sie für die folgenden dualcodierten gemischten Zahlen den Dezimalwert und die IEEE 754 Gleitkommadarstellung (einfache Genauigkeit, 32-bit) an:

(a) $101\,1000_2$

(b) $-1000\,1011,001_2$

Aufgabe 5.3 (Punkte 5+5+5+5)

IEEE 754 Gleitkommazahlen: Wandeln Sie folgenden Dezimalzahlen in Gleitkommazahlen einfacher Genauigkeit im IEEE 754 Format um. Es genügt dabei, wenn Sie die acht höchstwertigen Bit der Mantisse angeben:

(a) -254

(b) $40,25$

(c) $4,875$

(d) $-3,14159$

Aufgabe 5.4 (Punkte 10+10+10)

Arithmetische Operationen mit Gleitkommazahlen: Gegeben seien zwei IEEE 754 Zahlen A und B (32-bit). Von den 23-bit der Mantisse sind hier nur die oberen 8-bit angegeben, alle anderen Stellen sind 0. Die Zahlen sind folgendermaßen formatiert: *s eeee eeee mmmm mmmm*.

$A = 1\,1000\,0001\,1110\,0000$ und
$B = 0\,1000\,0000\,1100\,0000$

Berechnen Sie ohne Umwandlung in das Dezimalsystem die folgenden Ausdrücke (natürlich kann es helfen, die Rechnungen zur Kontrolle auch dezimal durchzuführen). Alle Ergebnisse sollen wieder als IEEE 754 Zahlen (wie oben) dargestellt werden. Geben Sie dabei immer auch die einzelnen Rechenschritte an.

(a) $A + B$

(b) $A - B$

(c) $A \cdot B$

Aufgabe 5.5 (Punkte 10+10+5)

ULP: Um zu untersuchen, wie sich Rundungsfehler bei der Gleitkomma-Arithmetik einschleichen und auswirken, betrachten wir folgendes Programm, das in einer Schleife immer wieder den Wert 1.0 aufaddiert.

```
1  // Test floating point arithmetic...
2  public class aufg05_5 {
3    public static void main( String args[] ) {
4      // calculate the sum of many (precise) numbers
5      long n = 0;
6      float sum = 0; float limit = 5.0E7f;
7      for( sum = 0.0f; sum < limit; sum += 1.0f ) {
8        // if ((n % 100000) == 0) {
9        //   System.out.println("n="+n + " sum="+sum + " target="+(n*1.0));
10       // }
11       n++;
12     }
13     System.out.println("sum is " + sum + " compared to " + (n*1.0));
14   }
15 }
```

Laden Sie dazu den Quellcode (`aufg05_5.java`) und die Class-Datei (`auf05_5.class`) vom Moodle oder der RSB-Webseite herunter. Bei installierter Java Runtime (JDK, JRE) kann das Programm dann mit

```
javac aufg05_5.java                    (Java Compiler)
java aufg05                            (Programm ausführen)
```

compiliert und gestartet werden.

(a) Beschreiben Sie kurz, was das Programm tut. Was erwarten Sie (ungefähr) als Ausgabewert des Programms? Wie viele Iterationen sollten am Ende in der `for`-Schleife durchlaufen worden sein?

(b) Was passiert tatsächlich? Warum?

Tipp: Es kann helfen, die auskommentierten Codezeilen 8 . . . 10 wieder zu aktivieren, neu zu compilieren, und neu zu starten.

(c) Schreiben Sie das Programm so um, dass es den ursprünglich angedachten Zweck erfüllt. Erläutern Sie Ihre Änderungen.