64-041 Übung Rechnerstrukturen und Betriebssysteme

Aufgabenblatt 5 Ausgabe: 02.12., Abgabe: 09.12. 24:00

Gruppe	
Name(n)	Matrikelnummer(n)

Aufgabe 5.1 (Punkte 10+10+10+10)

Arithmetische Operationen mit Gleitkommazahlen: Gegeben seien die beiden folgenden einfach genauen Gleitkommazahlen gemäß IEEE 754. Von der Mantisse sind jeweils nur die oberen acht Bit angegeben, alle anderen Bits sind 0. Zur besseren Lesbarkeit wird das Zeichen „|“ als Feldtrenner in dem Bitstring benutzt (*s eeee eeee mmmm mmmm*):

$A = (1 \mid 1000\,0001 \mid 1110\,0000)_2$ und
$B = (0 \mid 1000\,0000 \mid 1100\,0000)_2$

Berechnen Sie ohne Umwandlung in das Dezimalsystem die folgenden Ausdrücke. 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) $A \cdot B$

(d) $(A - B) \,/\, (A + B)$

Aufgabe 5.2 (Punkte 5+10+5)

Zeichensatzcodierung: Entschlüsseln Sie mit Hilfe der Vorlesungsunterlagen den folgenden hexadezimal codierten Text.

```
41 75 66 67 61 62 65 20 35 2E 32 20 69 73 74 20 65 69 6E 20 55 54 46 2D
38 20 54 65 78 74 0A 09 55 6D 6C 61 75 74 65 3A 20 C3 A4 20 73 74 61 74
74 20 61 65 0A 09 53 79 6D 62 6F 6C 65 3A 20 E2 87 92 2C 20 E2 80 B0 20
E2 80 A6 0A
```

(a) Um was für eine Zeichensatzcodierung handelt es sich?

(b) Wie sieht der Text aus? Notieren Sie dazu die Textdarstellung (mit Steuerzeichen).

(c) Was verrät Ihnen der Text über den Rechner mit dem er erstellt worden ist?

Aufgabe 5.3 (Punkte 10)

Base-64 Codierung: Wie in der Vorlesung skizziert, werden bei der Base-64 Codierung jeweils drei 8-bit Eingangswerte durch vier 6-bit Ausgangswerte ersetzt, die dann zur Datenübertragung in (7-bit) ASCII-Zeichen codiert werden.

Beschreiben Sie durch Logische- und Schiebe-Operationen, wie aus den drei Eingabezeichen `a1...a3`, die vier 6-bit Ausgangswerte `b1...b4` berechnet werden. Vervollständigen Sie dazu die Ausdrücke `b..` im nachfolgenden Java-Code.

```
1  int a1, a2, a3;                    // drei Zeichen, Wertebereich je 0..255
2
3  int b1 = ?
4  int b2 = ?
5  int b3 = ?
6  int b4 = ?
7
8  char[] base64table = new char[] {  // Tabelle ersetzt b_i durch Zeichen
9    'A', 'B', ... 'Z',               // nicht vollständig ...
10   'a', 'b', ... 'z',
11   '0', '1', ... '9', '+', '/' };
12
13 String base64out =                 // 4 Zeichen String als Ausgabe
14   base64table[ b1 ] +
15   base64table[ b2 ] +
16   base64table[ b3 ] +
17   base64table[ b4 ];
18 ...
```

Aufgabe 5.4 (Punkte 5+5+5)

Shift-Operationen statt Multiplikation: Ersetzen Sie die folgenden Berechnungen *möglichst effizient* durch eine Folge von Operationen: <<, +, -. Nehmen Sie für die Variablen x und y den Datentyp `int` (32-bit Zweierkomplementzahl) an.

(a) $y = 6 \cdot x$

(b) $y = -20 \cdot x$

(c) $y = 30 \cdot (x + 2)$

Aufgabe 5.5 (Punkte 15)

Rotate-Operationen: In der Vorlesung wurden die rotate-left und rotate-right Operationen vorgestellt, die aber in Java und C nicht als Operatoren definiert sind. Schreiben Sie daher eine Java-Klasse, die die rotate-Operationen als Methoden zur Verfügung stellt:

```
6
7  public class RotateLeftRight {
8
9    public static int rotateLeft( int i, int distance ) {
10     // hier steht der eigene Code
11     return ...;
12   }
13
14   public static int rotateRight( int i, int distance ) {
15     // hier steht der eigene Code
16     return ...;
17   }
18
```

Versuchen Sie, diese beiden Funktionen in Ihrer Klasse mit logischen und Shift-Operationen zu implementieren. Als Vorlage inklusive Selbsttest finden Sie dazu in dem Moodle die Java-Klasse `RotateLeftRight`. Hinweis: die beiden obigen Methoden sind natürlich in der Java-Klassenbibliothek in `java.lang.Integer` enthalten. Hier geht es darum, deren Funktion zu verstehen.