Praktische Informatik I WS 2008/09

Übungsblatt 6

Abgabe: 8.12.2008

Aufgabe 1 Vom Algorithmus zur Assembler-Ausführung

Aufgabe 1.1 Der Algorithmus (10%)

> Wenn CD aber AB nicht misst, und man nimmt bei AB, CD abwechselnd immer das kleinere vom grösseren weg, dann muss (schliesslich) eine Zahl übrig bleiben, die die vorangehende misst .

[Aus Euklid, Die Elemente, Herausgegeben von Clemens Thaer, Wissenschaftliche Buchgesellschaft Darmstadt, VII Buch, §2]

Dies ist wohl eine der ältesten Algorithmenbeschreibungen der Welt (ca. 300 v.u.Z.) mit der der *grösste gemeinsame Teiler* (ggT) zweier positiver ganzer Zahlen ermittelt wird. Dabei gelten die folgenden Axiome:

1. $ggT(x, x) = x$.
2. $ggT(x, y) = ggT(y, x)$.
3. $ggT(x, y) = ggT(x, y - x)$ für $x < y$.

Schreibt eine Java-Methode, die dieses Verfahren implementiert.

Aufgabe 1.2 Java oder Assembler? - Egal (15%)

Gegeben ist folgende Liste vom Assemblerbefehlen.

LOAD x	Lade den Inhalt von Adresse x in den Akkumulator
LOADA	Lade den Inhalt der Adresse, deren Wert im Akkumulator steht, in den Akkumulator
STORE x	Speichere den Inhalt des Akkumulators in der Speicherzelle mit der Adresse x
ADD x	Addiere den Wert an Adresse x zum Inhalt des Akkumulators
SUB x	Subtrahiere den Wert von Adresse x vom Inhalt des Akkumulators
MULT x	Multipliziert den Wert von Adresse x mit dem Inhalt des Akkumulators
JMPNEG x	Springe zur Marke x, wenn der Inhalt des Akkumulators < 0
JMPEQ x	Springe zur Marke x, wenn der Inhalt des Akkumulators $= 0$
JLE x	Springe zur Marke x, wenn der Inhalt des Akkumulators $<= 0$
JMP x	Springe zur Marke x
NOT	logische Negation - bitweises Komplement des Wertes im Akkumulator
NEG	unäres Minus angewendet auf den Wert im Akkumulator
HALT	Beendet das Programm

Schreibt ein Assembler-Programm zur Berechnung des grössten gemeinsamen Teilers zweier natürlicher Zahlen.

Aufgabe 1.3 Von Neumann Simulation (40%)

Schreibt ein JAVA Programm, mit dem ein Assemblerprogramm, das den obigen Befehlssatz verwendet, ausgeführt werden kann.

Die einzelnen Schlüsselwörter sollen dabei als int-Konstanten definiert werden, der Programmspeicher als `int [][]` (als Array von Paaren (*Befehl*, *Operand*)), und der separate Datenspeicher als `int []` Array.

Aufgabe 1.4 Einbinden des Assemblerprogramms (15%)

Schreibt eine Methode, die zu zwei natürlichen Zahlen der ggt berechnet, indem das unter Aufgabe 1.2 erstellte Programm in Programm- und Datenspeicher für Aufgabe 1.3 bereitgestellt wird, die Eingabeparameter richtig gesetzt werden, der Simulator aufgerufen wird, und danach das Resultat aus dem Datenspeicher zurückgegeben wird.

Aufgabe 1.5 Vergleich (20%)

Schreibt einen *back-to-back*-Test, der das Verhalten des Ursprungsprogramms aus Aufgabe 1.1 mit der Ausführung der Simulation vergleicht. Angenommen, ein solcher Test würde fehlschlagen:
Welche Ursachen können zu diesem Fehlschlag führen, und welche Massnahmen müssen jeweils zur Korrektur vorgenommen werden?