11. Übungsblatt: Programmierpraktikum II (SS 2003)

Abgabe: 21. Juli 2003

Bemerkungen:

- Alle nachfolgenden Aufgaben sind für den M68000-Assembler zu lösen. Die Abgaben müssen sich mit dem BSVC-Emulator im PI-Pool ohne Fehlermeldungen und Warnungen assemblieren lassen.
- Die Abgabe ist ausführlich zu kommentieren; unzureichend kommentierte Assembler-Programme werden nicht korrigiert.

Aufgabe 1: Euklidischer Algorithmus

(10 Punkte, Abgabedatei: euklid.s)

Der größte gemeinsame Teiler (ggT) zweier positiver ganzer Zahlen a und b ist die größte ganze Zahl g, die sowohl a als auch b teilt. Während der ggT für kleine Zahlen (z.B. $a = 70$ und $b = 36$) noch leicht im Kopf auszurechnen ist, verwendet man für große Zahlen (z.B. $a = 42.277.256$ und $b = 55.578.403$) den sogenannten Euklidischen Algorithmus. Dieser berechnet den ggT rekursiv wie folgt:

```
ggT(a,b)
1. if b = 0
2.   return a
3. else
4.   return ggT(b, a mod b)
```

Schreibe ein vollständiges Assembler-Programm zur Berechnung des ggT mit Hilfe des Euklidischen Algorithmus. Dabei sollen die Eingabewerte a und b als Langwörter in den Registern `D0` und `D1` und die Lösung als Langwort in Register `D2` stehen.

Aufgabe 2: Binärbaum durchsuchen

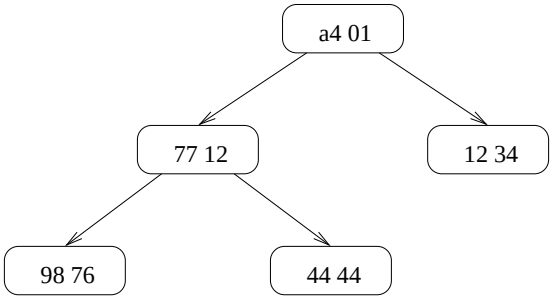
(10 Punkte, Abgabedatei: treesearch.s)

Wir betrachten einen Binärbaum, dessen Knoten jeweils aus einer 16-Bit-Zahl, der 16-Bit-Adresse des linken Nachfolgers und der 16-Bit-Adresse des rechten Nachfolgers besteht. Falls ein Nachfolger nicht existiert, wird `$0000` als Adresse angegeben.

Bsp.: Betrachte den folgenden Speicherinhalt:

```
007000: a4 01 70 06 70 0C 77 12 70 12 70 18 12 34 00 00
007010: 00 00 98 76 00 00 00 00 44 44 00 00 00 00
```

Steht die Wurzel des Binärbaumes ab Adresse `$7000`, so besitzt dieser Baum die folgende Gestalt:



Schreibe ein vollständiges Assembler-Programm, das einen solchen Binärbaum per Tiefensuche durchläuft und dabei einen Wert a sucht. Die Wurzeladresse des Baumes soll in Register `A0` stehen, der gesuchte Wert in Register `D0`. Nach Ende des Programmes soll in Register `D1` stehen, wie oft der gesuchte Wert gefunden wurde.

Zur Erinnerung: Die Tiefensuche in einem Binärbaum mit Wurzelknoten p ist rekursiv definiert wie folgt:

```
dfs(p)
1. if(linkerSohn(p) != 0)
2.   dfs(linkerSohn(p))
3. if(rechterSohn(p) != 0)
4.   dfs(rechterSohn(p))
5. return
```