PHILIPPS-UNIVERSITÄT MARBURG
Fachbereich Mathematik und Informatik
Prof. Dr. R. Loogen

D-35032 Marburg
Hans Meerwein Straße
Lahnberge
13. Januar 2003

Übungen zur „Technischen Informatik I“, WS 2002/03

Nr. 11, Besprechung bzw. Abgabe: 22.1. bis 24.1. in den Übungsgruppen

---

Hinweis zum Microcodesimulator

Der von Dr. Martin Perner entwickelte Mikrocodesimulator liegt als selbst entpackendes Archiv auf der Internetseite zur Vorlesung sowie entpackt zum Kopieren auf dem NT-Server Bangkok unter `M:\2002WS\Technische Informatik I\Tools\MicroSim2000` bereit.

A. Mündliche Aufgaben

52. Mikroprogrammiertes Maschinenspracheprogramm

    Entwickeln Sie analog zur Vorgehensweise der Vorlesung (Folien 273 – 277) ein mikroprogrammiertes Maschinenspracheprogramm für das folgende Programm (in Pascal-Syntax):

```
VAR n, m: Integer;
VAR b, i, aux: Integer;
BEGIN
  read(n);    read(m);  b := n;    i := 2;   aux := n-1;
  WHILE i <= m DO
    BEGIN
      b := (b*aux) div i;
      i := i+1; aux := aux-1;
    END;  write(b)
END.
```

    Was berechnet das Programm?

B. Hausaufgaben

Die Abgabe der Hausaufgaben ist in Gruppen bis zu 3 Personen erlaubt.

53. Quersummenbestimmung **4 Punkte**

    Schreiben Sie ein ausführlich kommentiertes Mikroprogramm, das in Register R1 die Anzahl der Einsen im Register R0 (Quersumme von R0) bestimmt.

54. Mikrobefehle **4 Punkte**

    Wie kann man die folgenden Tests und Zuweisungen mit Mikrobefehlen realisieren? Geben Sie jeweils kommentierte Mikrobefehle bzw. -befehlsfolgen an.

    Der Parameter i mit $0 \leq i \leq 31$ sei jeweils im Register R1 gegeben. Dabei bezeichne $i = 0$ die niedrigstwertige und $i = 31$ die höchstwertige Position in einem 32-Bit-Wort.

(a) setze Bit i in Register R7 auf 0, lasse restliche Bits unverändert

(b) Sprung an Adresse 64, wenn Bit i in Register R0 gleich Eins ist

55. Maschinenbefehle **4 Punkte**

Entwickeln Sie für die folgenden Assemblerbefehle Mikroprogramme, die in den Load-Increment-Execute-Zyklus eingebunden werden können:

(a) ADD <adr1>, <adr2>, <adr3> addiert die an den Adressen <adr1> und <adr2> gespeicherten 4-Byte-Werte und schreibt das Ergebnis an die Adresse <adr3>.

(b) JLEQ <adr> springt an die Adresse <adr>, falls die vorherige (Vergleichs-) Operation das Ergebnis $<=$ geliefert hat.