



Prof. Dr. Rolf Drechsler, drechsler@informatik.uni-bremen.de, MZH 3510  
Dr. Robert Wille, rwille@informatik.uni-bremen.de, MZH 3485  
Oliver Keszöcze, keszocze@informatik.uni-bremen.de, MZH 3440

2. Übungsblatt zur Vorlesung

# Technische Informatik 1

## Aufgabe 1



(1 + 1 Punkte)

- Beschreibe den Unterschied zwischen synchroner und asynchroner Datenübertragung auf einem Bussystem.
- Welche Vorteile und Nachteile haben die jeweiligen Übertragungs-Schemata?

## Aufgabe 2

(3 + 3 Punkte)

Die Arbeitsgruppe Rechnerarchitektur (AGRA) möchte ihr Akronym mit Hilfe von Leuchtstoffröhren über den gesamten Campus erstrahlen lassen. Hierzu wurden die Röhren, wie in Abbildung 1 dargestellt, angeordnet. Aus Platzgründen kann leider immer nur ein Buchstabe gleichzeitig angezeigt werden.

Die Buchstaben haben folgende Form: „A“ = , „G“ =  und „R“ = .

Die Anordnung soll nun nacheinander die Buchstaben „A“, „G“, „R“ und „A“ anzeigen. Die einzelnen Röhren werden entsprechend einem Dualzähler geschaltet, der damit die Funktion eines Taktgebers übernimmt. Der Zählerstand wird durch die Variablen  $x_3, x_2, x_1, x_0$  repräsentiert, die Röhren durch die Variablen  $a_i$ , wobei  $a_i = 1$  bedeutet, dass diese Röhre leuchtet.

Alle Buchstaben sollen gleich lang (und exakt einmal) zu sehen sein. Zwischen den Buchstaben soll keine Pause entstehen, d. h. es leuchtet immer mindestens eine Röhre.

Eine Besonderheit gibt es bei dem Buchstaben „A“. Dieser soll wie folgt animiert werden: Zunächst wird das linke „Bein“ ( $a_9, a_{11}$  und  $a_4$ ) des Buchstabens erleuchtet, dann zusätzlich das rechte „Bein“ ( $a_5$ ) und schließlich der Querbalken in der Mitte ( $a_{10}$  und  $a_6$ ). Jeder Schritt dieser Animation dauert genau einen Zeittakt (die Animation gilt hierbei als das Anzeigen des Buchstaben). Das „A“ am Ende des Akronyms soll exakt umgekehrt animiert werden, d. h. am Ende der Animation leuchtet nur das linke „Bein“.

Gib eine Boolesche Funktion  $f : \mathbb{B}^4 \rightarrow \mathbb{B}^{12}$  durch eine Wahrheitstabelle und durch Boolesche Ausdrücke an, die die Steuerung der Röhren realisiert.

## Aufgabe 3

(2 + 2 Punkte)

Es soll ein neuer Rechner für ein spezielles Einsatzgebiet entworfen werden. Die Rechengenauigkeit aller Operationen soll 32 Bit sein. Aufgrund der vorkommenden Operationen werden 166 Instruktionen benötigt. Der Rechner hat 42 Register bei einem Adressraum von maximal 750K. Es wird eine Load/Store-Architektur verwendet: Befehle, die auf den Speicher zugreifen, haben zwei Register als Operanden (Befehl 1:  $R_{\text{dest}} := \text{Mem}[R_{\text{src}}]$ , Befehl 2:  $\text{Mem}[R_{\text{src}}] := R_{\text{dest}}$ ). Kein Befehl hat mehr als drei Operanden, mindestens zwei Operanden sind Register, der dritte kann eine 16-Bit-Konstante oder ein Register sein.

- Wie breit müssen die Register mindestens sein?
- Wie viele Bit werden benötigt, um ein Maschinenwort zu codieren?

Begründe deine Überlegungen.

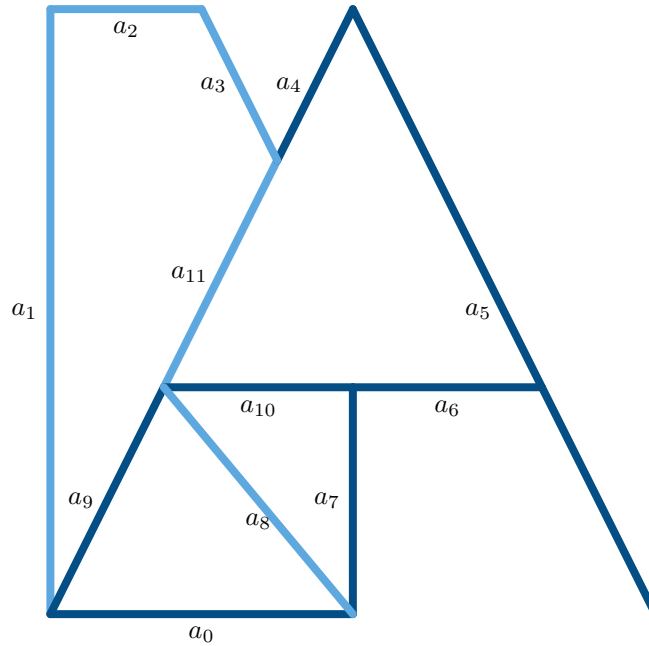


Abbildung 1: Anordnung der Leuchtstoffröhren mit leuchtendem "R" ( $a_1 = a_2 = a_3 = a_8 = 1$ )

#### Aufgabe 4

(1 + 1 Punkte)

Zeige (z.B. in MIPS-Assembler), dass es möglich ist, eine Zuweisung mittels

- indirekter Adressierung ( $R_{\text{dest}} := \text{Mem}[\text{Mem}[R_{\text{src}}]]$ )
- indizierter Adressierung ( $R_{\text{dest}} := \text{Mem}[R_{\text{src}} + \text{index}]$ )

nur unter Verwendung von absoluter Adressierung ( $R_{\text{dest}} := \text{Mem}[R_{\text{src}}]$ ) zu realisieren.

#### Aufgabe 5

(6 Punkte)

Das *Collatz-Problem* (auch  $(3n + 1)$ -Vermutung genannt) ist gegeben durch die Zahlenfolge

$$a_0 \in \mathbb{N}^+$$

$$a_n := \begin{cases} \frac{a_{n-1}}{2} & a_{n-1} \text{ gerade} \\ 3 \cdot a_{n-1} + 1 & a_{n-1} \text{ ungerade} \end{cases}$$

Die Vermutung lautet, dass für jedes  $a_0 \in \mathbb{N}^+$ , die Folge  $(a_n)$  in den Zyklus 4, 2, 1 mündet. Startet man z. B. mit  $a_0 = 7$  so ergibt sich die Zahlenfolge

$$(a_0, a_1, a_2, a_3, a_4, a_5, a_6, a_7, a_8, a_9, a_{10}, a_{11}, a_{12}, a_{13}, a_{14}, a_{15}, a_{16}, a_{17}, a_{18}, \dots)$$

$$(7, 22, 11, 34, 17, 52, 26, 13, 40, 20, 10, 5, 16, 8, 4, 2, 1, 4, \dots)$$

Wir wollen nun den Zykluseintritts-Index  $n^*$  bestimmen, für den zum ersten Mal gilt  $a_{n^*} = 4$  (formal:  $n^* := \min\{n \in \mathbb{N}^+ | a_n = 4\}$ ).

Um dieses Problem zu lösen, haben wir folgendes Programm geschrieben:

```
int collatz(int a_0) {
    int a_n=a_0;
    int nStar=0;
    while (a_n != 4) {
        if (a_n % 2 == 0) {
            a_n=a_n / 2;
        }
        else {
            a_n=3*a_n+1;
        }
        nStar=nStar+1;
    }
    return nStar;
}
```

Gib ein MIPS-R2000-Assembler-Programm an, das  $n^*$  berechnet. Der Parameter  $a_0$  steht zu Beginn in dem Register  $\$a0$  zur Verfügung, das Ergebnis soll in  $\$v0$  gespeichert werden.

Implementiere auch die allgemeine Modulo-Operation. Diese muss also Ausdrücke der Form  $a\%b$  berechnen können. Zur Vereinfachung gehen wir davon aus, dass  $a, b > 0$  gilt. Die MIPS-Befehle `rem` und `div` dürfen zur Implementierung der Modulo-Operation *nicht* verwendet werden

Kommentiere dein Programm ausführlich. Unkommentierte Programme werden nicht bewertet!

Das Assembler-Programm ist elektronisch an die Tutoren zu schicken. Hierbei gilt die gleiche Frist, wie für den restlichen Übungszettel.

Den Befehlssatz des MIPS R2000 findest du z. B. unter folgender URL:

[http://www.iss.tu-darmstadt.de/student\\_area/tgdi/uebungen/mips-r2000.pdf](http://www.iss.tu-darmstadt.de/student_area/tgdi/uebungen/mips-r2000.pdf)

Ein kleines Tutorial, welches auch den `syscall`-Befehl behandelt, findest du unter

<http://logos.cs.uic.edu/366/notes/mips%20quick%20tutorial.htm>

Um dein Programm zu testen, kannst du einen MIPS Simulator verwenden, z. B.:

**MARS:** <http://courses.missouristate.edu/KenVollmar/MARS/index.htm>

**SPIM:** <http://spimsimulator.sourceforge.net/>

**Abgabetermin:** vor Beginn der Vorlesung am **25. April 2013**