Aufgabenblatt 5: Assembler-Programmierung 1

In dieser Übung sollen für den in der Vorlesung vorgestellten Mips-Prozessor Assembler-Programme geschrieben werden. Aus der Dokumentation zu SPIM, die auf den GRA-Webseiten zu finden ist, sollten Ihnen dazu folgende Anweisungen bekannt sein:

Speichersegmente

```
.data   dynamischer Datenbereich
.text   Programm
```

Speicherplatzreservierungen

```
.byte    b1,b2, ...   Speichert Bytes b1,b2,...
.half    h1,h2, ...   Speichert 16-Bit Worte h1,h2,...
.word    w1,w2, ...   Speichert 32-Bit Worte w1,w2,...
.ascii   ''abc''      Speichert String abc,der nicht Null-terminiert wird
.asciiz  ''abc''      Speichert String abc,der zusätzlich Null-terminiert wird
.align   n            Speicherplatzzuweisung auf der nächsten Grenze 2^n
.space   n            Anzahl leerer Speicherplätze
```

Zahlen

```
ZZ    Dezimalzahl
0xZZ  Hexadezimalzahl
```

Es ist zu beachten, dass der MIPS-Prozessor eine **big-endian**-Architektur ist. Der Unterschied zwischen little-endian und big-endian ist die Anordnung der einzelnen Bytes eines 16-Bit bzw. 32-Bit Wortes. Man stelle sich folgende Situation vor:

Adr. 0x00-0x03:	0x01	0x02	0x03	0x04

Im little-endian Modus würde das 32-Bit Wort den Wert `0x04030201`, und im big-endian Modus den Wert `0x01020304` erhalten.

Aufgabe 1: Speicherzuordnung

Folgendes Assembler-Programm ist gegeben:

```
        .data
item:   .byte 1,2,33,0xff
        .word 0x34
        .half 0x23
        .byte 0x12
        .space 5
        .ascii "a"

        .text
        .globl main
main:   lw $t0,item
```

a) Geben Sie an, was folgender Programmabschnitt tut.

b) Tragen Sie in die Tabelle die Speicherbelegung nach Ausführung des Programms ein.

Speicherausschnitt:

	0x00	0x01	0x02	0x03
Adr. 0x00:				
Adr. 0x04:				
Adr. 0x08:				
Adr. 0x0c:				
Adr. 0x10:				

c) Berechnen Sie den Inhalt von `$t0` nach Ablauf des Programms (dezimal).

Aufgabe 2: Datenstrukturen

Sei folgende Java-Klasse gegeben:

```
class Item
{
        int Flag;
        int Index;
        int Length;
        char[] Text = new char[16];
        int EndFlag;
};
```

a) Formen Sie eine Instanz der oben genannten Java-Klasse in Assemberanweisungen um, die die Speicherbelegung im ".data"-Bereich vornehmen.

b) Schreiben Sie eine Assembler-Sequenz, die auf das Feld Length zugreift.

Hinweis: Die Länge von `int` beträgt 32 Bit; `char` wird mit 8 Bit angesetzt.

Aufgabe 3: Sortieren

Schreiben Sie einen Bubble-Sort-Algorithmus, der ein Feld mit 8 Wörtern sortiert. Verwenden sie den folgenden Programmablaufplan als Hilfestellung:

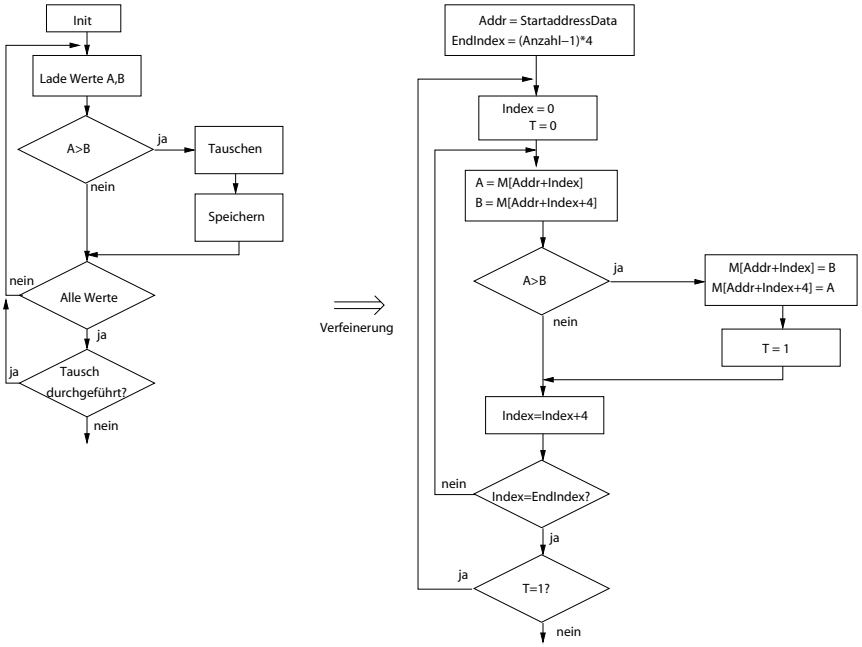

Abbildung 1: Programmablaufplan in 2 Stufen der Verfeinerung

a) Erklären Sie die Funktionsweise des gegebenen Programmablaufplans

b) Realisieren Sie ein Assembler-Programm mit Hilfe des Programmablaufplans

Hinweis: Bei der Realisierung des Assembler-Programmes können Sie die Prozessor-Register `$t0`-`$t7`, `$s0`-`$s7` und das Register `$zero` (Dieses Register enthält immer den Wert `0` und kann nicht beschrieben werden) benutzen.