Musterlösung Übungsblatt 5: Assembler-Programmierung 1

Aufgabe 1: Speicherzuordnung

a) Das Programm füllt den Speicher mit den gegebenen Werten und ließt dann den Inhalt von *item* in das Prozessor-Register `$t0`.

b) Die Speicherbelegung, die das Assembler-Programm vornimmt, ist folgender Tabelle zu entnehmen:

	0x00	0x01	0x02	0x03	
Adr. 0x00	0x01	0x02	0x33	0xff	`.byte 1,2,33,0xff`
Adr. 0x04	0x00	0x00	0x00	0x34	`.word 0x34`
Adr. 0x08	0x00	0x23	0x12	X	`.half 0x23 .byte 0x12`
Adr. 0x0c	X	X	X	X	`.space 5`
Adr. 0x10	"a"				`.ascii "a"`
Adr. 0x14					

c) Um nun den Wert, der in `$t0` steht zu berechnen, müssen folgende Zahlen mit der richtigen Wertigkeit addiert werden:

`$t0` $= 1 \cdot 2^{24} + 2 \cdot 2^{16} + 33 \cdot 2^8 + 255 \cdot 2^0 = 16.916.991$

Es steht also nach Beendigung des Programmes in `$t0` der Wert $(16.916.991)_{10}$.

Aufgabe 2: Datenstrukturen

a) Die Instanz der Java-Klasse kann folgendermaßen in Assembler realisiert werden:

```
        .data
Item:
Flag:   .word 0     # int Flag
Index:  .word 0     # int Index
Length: .word 0     # int Length
Text:   .byte 0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0
                    # char[] text = new char[16]
Endflag: .word 0    # int EndFlag

        .text
        .global main
main:   lw $t0, Length
```

b) Die Anweisung `lw $t0, Length` greift auf das Feld *Length* zu.

Aufgabe 3: Bubble-Sort

a) Wichtig bei dem angegebenen Algorithmus ist, dass für die Berechnung der Speicheradressen aus den Zählvariablen diese mit 4 multipliziert werden müssen, da auf 32-Bit Wörtern gearbeitet wird. Es muss also immer 4 Byte (= 1 Wort) weitergezählt werden um auf das nächste Datum zugreifen zu können.

b) Folgendes Assembler-Programm realisiert den Bubble-Sort auf 8 Wörtern. Hierbei ist zu beachten, dass das Programm schon an einigen Stellen optimiert ist, so dass manche Anweisungen auf den ersten Blick nicht unbedingt einleuchtend sind.

```
# bubble sort for words
# data to sort begins at 'items', number of data is defined at 'length'
# length must be greater then 1
# (c) Thomas Lehmann, AG Rammig, Universitaet-GH Paderborn
#
        .data
items:  .word 0x00
        .word 0x001
        .word 0x012
        .word 0x003
        .word 0x004
        .word 0x005
        .word 0x006
        .word 0x007
        .word 0x008
        .word 0x009
        .word 0x008
        .word 0x010
data:   .word items
length: .word 8                 # number of words to sort
        .text
        .globl main

main:   lw $t6, length          # get number of words
        li $t0,1
        sub $t6,$t6,$t0         # length-1
        add $t6,$t6,$t6
        add $t6,$t6,$t6         # multiply by 4
        lw $t0, data
        add $t1,$zero,$t0       # copy start of data to t1
        add $t5,$t6,$t0         # calculate address of data end
loop:   add $t4,$zero,$zero     # reset flag t4
load:   lw $t6,0($t1)           # load data
        lw $t7,4($t1)
        slt $t2,$t6,$t7         # compare
        bne $t2,$zero, next
change: ori $t4,1               # set flag something chaned
        sw $t6,4($t1)           # change by store
        sw $t7,0($t1)
next:   addi $t1,4              # inc address
        bne $t1,$t5, load       # not the end
        add $t1,$zero,$t0
        bne $t4,$zero,loop
exit:
```