Allgemeiner Hinweis: Beachten Sie die Links zu MIPS im TUWEL-Forum!

Aufgabe 1: Instruktionsformate

a) Sie haben folgendes Bitmuster gegeben:

```
0000 0010 0001 0001 1000 0000 0010 0010
```

Welcher MIPS-Assemblerbefehl entspricht diesem Bitmuster?

b) Sie haben folgende Befehle gegeben:

```
addi $t0, $s6, 8
add $t9, $s7, $zero
sw $t1, 24($t0)
```

Geben Sie für jeden Befehl den Typ und die Werte der einzelnen Felder (als dezimale Zahlen) im entsprechenden Instruktionsformat für diesen Typ an!

Aufgabe 2: Logische Operationen

Gehen Sie für diese Aufgabe von folgenden Registerinhalten aus:

```
$s0 = 0xAAAAAAAA
$s1 = 0xBBBBBBBB
```

Sie haben folgende MIPS-Sequenz gegeben:

```
and $t0, $s0, $s1
or $t1, $s0, $s1
sll $s1, $s1, 1
and $t2, $s0, $s1
or $t3, $s0, $s1
```

Welche Werte beinhalten die Register $t0, $t1, $t2 und $t3 nach der Ausführung der obigen Sequenz und wie ergeben sich diese Werte?

Aufgabe 3: Verzweigungen

a) Gehen Sie davon aus, dass das Register $t0 am Anfang den Wert 0x00000010 enthält. Welchen Wert enthält $t2 nach dem Ausführen der folgenden MIPS-Sequenz? Wie entsteht dieser Wert?

```
        slt $t2, $zero, $t0
        bne $t2, $zero, ELSE
        j DONE
ELSE:   addi $t2, $t2, 2
DONE:   addi $t2, $t2, 1
```

b) Welchen Wert enthält $t2, wenn das Register $t0 am Anfang den Wert 0xFFFFFFF0 enthält? Wie entsteht dieser Wert?

c) Wie kann man den Code kürzer (weniger Zeilen) schreiben und trotzdem das gleiche Ergebnis erzielen?

Aufgabe 4: Schleifen

a) Gehen Sie davon aus, dass am Anfang das Register $t1 den dezimalen Wert 20 und das Register $s2 den dezimalen Wert 0 enthält. Welchen Wert enthält $s2 nach dem Ausführen der folgenden MIPS-Sequenz:

```
LOOP:   slt $t2, $zero, $t1
        beq $t2, $zero, DONE
        addi $t1, $t1, -2
        addi $s2, $s2, 3
        j LOOP
DONE:
```

b) Welcher Code in einer Hochsprache (z.B. Java oder C) entspricht ungefähr dieser MIPS-Sequenz? Sie dürfen wieder beliebige Bezeichner verwenden.

Aufgabe 5: Unterprogramm

Sie haben folgendes Unterprogramm gegeben:

```
func:   add $v0, $a0, $zero
        bgt $a0, $a1, endif
        add $v0, $a1, $zero
endif:  jr $ra
```

Beantworten Sie bitte folgende Fragen:

- Wie viele Parameter werden laut Konvention übergeben?
- Wo wird der Rückgabewert abgelegt?
- Was macht dieses Unterprogramm?
- Wird hier der Stack benutzt?
- Welcher Wert wird in $ra abgelegt, wenn das Unterprogramm aufgerufen wird?