Aufgabe 4.1
Berechnen Sie die Instruktionswörter für folgende Instruktionen „per Hand“! Nehmen Sie dafür die Instruktionsbeschreibungen aus dem „Instruction Set Manual“ zu Hilfe, und vergleichen Sie Ihr Ergebnis mit den vom Assembler generierten Instruktionswörtern!

```
addi  $8,$0,0
addi  $8,$8,1
ori   $8,$0,0xffff
addi  $9,$0,-16384
add   $8,$8,$9
```

Aufgabe 4.2
Schreiben Sie ein Programm, das einige Register mit den Werten gemäß nebenstehender Tabelle initialisiert! Es dürfen keine Pseudoinstruktionen benutzt werden, also im MARS Menüpunkt “Settings” muss die Markierung bei “Permit extended (pseudo) instructions and formats” entfernt sein.

Führen Sie Ihr Programm schrittweise aus, und beobachten Sie, wie sich die Registerinhalte ändern!

GPR	Wert
8	0
9	11
10	0x1000
11	-1
12	-0x8000
13	0x8000
14	0xffff0000
15	0x7fffffff
24	5322
25	75

Aufgabe 4.3
Erweitern Sie Ihr Programm aus Aufgabe 4.2, so dass nach der Initialisierung der Register die nebenstehenden Operationen ausgeführt werden! Die Werte in den Quellregistern sollen sich dabei nicht ändern.

Es dürfen keine Pseudoinstruktionen benutzt werden, also im MARS Menüpunkt “Settings” muss die Markierung bei “Permit extended (pseudo) instructions and formats” entfernt sein. Nehmen Sie die MIPS32 Handbücher zu Hilfe! Eine gute Referenz ist auch die Hilfe-Funktion des MARS, die über den Menüpunkt *Help* oder den entsprechenden Button aufgerufen werden kann.

Wenn Runtime-Fehler auftreten, erklären Sie warum! Machen Sie sich den Unterschied zwischen den Operationen `addi` und `addiu` klar!

Ziel	Operation
GPR[2]	GPR[10] + GPR[9]
GPR[3]	GPR[10] - GPR[9]
GPR[4]	NOT(GPR[9]) + GPR[10] + 1
GPR[5]	GPR[13] « 5
GPR[6]	GPR[13] » 5 (arithmetic)
GPR[7]	GPR[12] » 5 (arithmetic)
GPR[16]	GPR[12] » 5 (logical)
GPR[17]	GPR[13] » GPR[9] (logical)
GPR[18]	GPR[24] / GPR[25]
GPR[19]	GPR[24] % GPR[25]
GPR[20]	GPR[15] + 1 (unsigned)
GPR[21]	GPR[15] + 1 (signed)
GPR[22]	count_leading_ones(GPR[11])
GPR[23]	count_leading_ones(GPR[14])

Aufgabe 4.4
Schreiben Sie folgende Werte in die *data section*:

a) `0xa0` (byte)

b) `0xb1b0` (short, 2 byte)

c) `0xc3c2c1c0` (long, 4 byte)

d) `Das Ende` (ASCII)