

### ***Das zugrundeliegende Programm:***

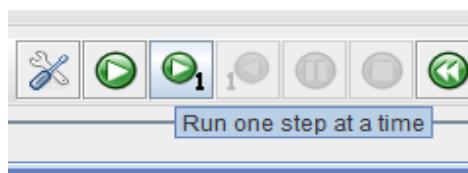
#### ***Addition zweier Integer***

Wie im HowTo beschrieben, wird *Simulation07.asm* im MARS geöffnet.

```
.data
    # Deklaration der Variablen
    number1: .word 3
    number2: .word 11
```

Die Beispielzahlen 3 und 11 werden im .data Teil des Codes hinterlegt.

Empfehlenswert ist es, nach dem Assemblieren Schritt für Schritt durch das Programm zu gehen, das geschieht durch Klicks auf den „Run one step at a time“ Button:



```
.text
    # Laden der Werte in die temporären Register
    lw $t0, number1($zero)
    lw $t1, number2($zero)
```

Im Textsegment werden zuerst die Worte in die temporären Register *\$t0* und *\$t1* geladen, die als Variablen *number1* und *number2* hinterlegt wurden, nach Ausführen dieser Codezeilen kann man also eine  $3_{10}$  bzw.  $11_{10}$  in diesen Registern sehen:

\$t0	8	0x00000003	
\$t1	9	0x0000000b	
\$t2	10	0x00000000	

```
# Addition t2 = t0 + t1
add $t2, $t0, $t1
```

Da nun die notwendigen Parameter bereitstehen, kann der eigentliche Additionsbefehl ausgeführt werden, hier wird die Summe der Inhalte der Register *\$t0* und *\$t1* in das Register *\$t2* geschrieben.

```
# Addition t2 = t0 + t1
add $t2, $t0, $t1

# e add $t1,$t2,$t3      Addition with overflow : set $t1 to ($t2 plus $t3)
li  add $t1,$t2,-100    ADDition : set $t1 to ($t2 plus 16-bit immediate)
sys add $t1,$t2,100000  ADDition : set $t1 to ($t2 plus 32-bit immediate)
```

Nach Ausführen des *add* Befehls ist die Summe, hier im Beispiel also  $14_{10}$ , in Register *\$t2* zu finden:

\$t0	8	0x00000003
\$t1	9	0x0000000b
\$t2	10	0x0000000e

Da die Aufgabenstellung keine Ausgabe erforderte, bleibt nun nur noch das Beenden des Programms, was wie in den anderen Simulationen dieser Reihe *Simulationen mit dem MARS Simulator* auch, über den Wert 10: *terminate execution* für den *syscall* passiert:

```
# exit
li $v0, 10      # der Wert 10 für den syscall bedeutet: exit (terminate execution)
syscall
```