

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

#### ***Subtraktion zweier Integer mit Eingabe der Zahlen und Ausgabe der Differenz***

Wie im HowTo beschrieben, wird *Simulation10.asm* im MARS geöffnet. Es sollen zwei Integerzahlen eingegeben werden, deren Differenz dann ausgegeben wird.

```
.data
prompt1: .asciz "\nBitte die erste Zahl eingeben; x = "
prompt2: .asciz "\nBitte die zweite Zahl eingeben, y = "
message: .asciz "\nDas Ergebnis der Subtraktion ist x - y = "
```

Im *.data* Teil des Codes werden Strings hinterlegt, die einerseits zur Eingabe der Summanden auffordern und andererseits die Ausgabe begleiten sollen.

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:



Der *.text* Teil beginnt mit der Aufforderung, die erste Zahl, also x, einzugeben:

```
.text
# Ausgabe der ersten Nachricht: prompt1
li $v0, 4 # der Wert 4 für den syscall bedeutet: print string
la $a0, prompt1 # lädt die Adresse des ersten Strings in $a0
syscall

# erste Zahl einlesen und im temporären Register $t0 ablegen
li $v0, 5 # der Wert 5 für den syscall bedeutet: read integer
syscall
move $t0, $v0
```

Der Wert 4 für den *syscall* bedeutet *print string*, und die Adresse dieses Strings muss dafür in Register *\$a0* geladen werden. Nach Ausgabe der Nachricht *prompt1* kann dann *x* eingelesen werden, dazu dient der Wert 5: *read integer* für den *syscall*. Wird dann eine Zahl eingegeben und mit *Enter* bestätigt, liegt sie in *\$v0* vor und wird hier, zur weiteren Verwendung, in das temporäre Register *\$t0* kopiert.

Ebenso wird für  $y$  verfahren, sodass nach Ausführen dieser Codezeilen  $x$  und  $y$  in den Registern  $\$t0$  und  $\$t1$  vorliegen und mit dem Subtraktionsbefehl fortgefahrene werden kann.

Hier zu sehen mit der beispielhaften Eingabe  $x = 5$  und  $y = 6$ :

reg		
$\$t0$	8	0x00000005
$\$t1$	9	0x00000006
$\$t2$	10	0x00000000

Der Subtraktionsbefehl schreibt nun die Differenz der Inhalte von  $\$t0$  und  $\$t1$  in das Register  $\$t2$ :

```
# Subtraktion
sub $t2, $t0, $t1
```

```
# Subtraktion
sub $t2, $t0, $t1

# A sub $t1,$t2,$t3      Subtraction with overflow : set $t1 to ($t2 minus $t3)
li    sub $t1,$t2,-100   SUBtraction : set $t1 to ($t2 minus 16-bit immediate)
mov   sub $t1,$t2,100000  SUBtraction : set $t1 to ($t2 minus 32-bit immediate)
...
```

Nun muss noch das Ergebnis ausgegeben werden, die Ausgabe wird vom String *message* begleitet:

```
# Ausgabe der dritten Nachricht: message
li $v0, 4
la $a0, message
syscall
```

Wie schon bei der Ausgabe der Aufforderungen *prompt1* und *prompt2* steht hier der Wert *4* für den *syscall* für *print string* und die Adresse des auszugebenden Strings muss in *\$a0* geladen werden.

```
# Wert ausgeben:
li $v0, 1          # der Wert 1 für den syscall bedeutet: $a0 = integer to print
move $a0, $t2       # legt den Wert aus $t2 im Register $a0 für Parameterübergabe ab
syscall
```

Der Wert *1* für den *syscall* bedeutet, dass im Übertragungsregister *\$a0* eine Integerzahl steht, die ausgegeben werden soll. Entsprechend wird die Summe aus *\$t2* nach *\$a0* kopiert, dann der *syscall* initiiert.

Nun bleibt 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
```

Die Ausgabe ist im Fenster *Run I/O* unterhalb des *Data Segments* im *execute* Fenster zu finden:

