Aufgabe 41

```
# Aufgabe 41: Arbeit mit der Gleitkommaeinheit
#
# von: Stephan Brumme
#
# zuletzt geaendert: 22.Juni 2000

        .text 0x400000
main:

# Aufgabe a)
        l.s     $f2,float1            # zwei Single-Precision-Floats laden
        l.s     $f3,float2

        l.d     $f4,double1           # zwei Double-Precision-Floats laden
        l.d     $f6,double2


# Aufgabe b)
        mul.s   $f2,$f2,$f3           # $f2 *= $f3
        neg.s   $f3,$f3               # $f3 = -$f3

        sub.d   $f4,$f6,$f4           # $f4 = $f6-$f4
        abs.d   $f6,$f6               # $f6 = |$f6|

        s.s     $f2,float1            # Ergebnisse wieder speichern
        s.s     $f3,float2

        s.d     $f4,double1
        s.d     $f6,double2


        li      $v0,10                # Programm beenden
        syscall


        .data

float1: .float 13.13
float2: .float 42.42
double1:.double 1.23456
double2:.double -9.8764321
```

Aufgabe 42

Aus dem verwendeten Programmfragment aus Aufgabe 41 wurden die Rechen- und Speicheroperationen entfernt, so dass lediglich der Datenbereich und die Ladebefehle unverändert blieben.

```
# Aufgabe 42: Arbeit mit der Gleitkommaeinheit
#
# von: Stephan Brumme
#
# zuletzt geaendert: 22.Juni 2000

        .text 0x400000
main:

        l.s     $f2,float1            # zwei Single-Precision-Floats laden
        l.s     $f3,float2

        l.d     $f4,double1           # zwei Double-Precision-Floats laden
        l.d     $f6,double2

# Aufgabe a)
        mfc1.s  $t0,$f2               # Single aus $f2 nach $t0 laden
        mfc1.s  $t1,$f3               # Single aus $f3 nach $t1 laden
        mfc1.d  $t2,$f4               # Double aus $f4 nach $t2 laden
        mfc1.d  $t4,$f6               # Double aus $f6 nach $t4 laden

# Aufgabe b)
        mov.s   $f22,$f2              # Single aus $f2 nach $f22 kopieren
        mov.s   $f23,$f3              # Single aus $f2 nach $f23 kopieren
```

```
        mov.d   $f24,$f4              # Double aus $f4 nach $f24 kopieren
        mov.d   $f26,$f6              # Double aus $f6 nach $f26 kopieren
        mov.d   $f29,$f6              # Double aus $f6 nach $f29 kopieren (Misalignment !)

        li      $v0,10                # Programm beenden
        syscall

        .data

float1: .float 13.13
float2: .float 42.42
double1:.double 1.23456
double2:.double -9.8764321
```

SPIM ignoriert Double-Zugriffe auf ungerade Register.

Aufgabe 43

Wiederum benutze ich das gleiche Programmfragment aus Aufgabe 41:

```
# Aufgabe 43: Arbeit mit der Gleitkommaeinheit
#
# von: Stephan Brumme
#
# zuletzt geaendert: 22.Juni 2000

        .text 0x400000
main:

        l.s     $f2,float1            # zwei Single-Precision-Floats laden
        l.s     $f3,float2

        l.d     $f4,double1           # zwei Double-Precision-Floats laden
        l.d     $f6,double2

# Aufgabe a)
        c.eq.s  $f2,$f3               # $f2=$f3 ?
        bc1f    if_else               # wenn falsch, dann springe zu if_else
                                      # (Vergleich mit den gegebenen Werten immer falsch)
if_then:                              # if-then-Zweig
        div.s   $f2,$f2,$f3
        j       if_end
if_else:                              # else-Zweig
        mul.s   $f2,$f2,$f3
if_end:                               # hier endet die if-then-else-Konstruktion


# Aufgabe b)
        mov.s   $f12,$f2
        jal     print_float
        mov.d   $f12,$f4
        jal     print_double


        li      $v0,10                # Programm beenden
        syscall

# ##############################################################################################
# PRINT_FLOAT:
# UP, das eine Single-Precision-Float ausgibt
#
# Parameter:   $f12 = auszugebene Zahl
print_float:
        li      $v0,2
        syscall
        jr      $ra


# ##############################################################################################
# PRINT_DOUBLE:
# UP, das eine Double-Precision-Float ausgibt
#
# Parameter:   $f12/$f13 = auszugebene Zahl
```

```
print_double:
        li      $v0,3
        syscall
        jr      $ra


        .data

float1: .float 13.13
float2: .float 42.42
double1:.double 1.23456
double2:.double -9.8764321
```

Aufgabe 44

Mein Programm berechnet die Quadratwurzel von natürlichen Zahlen, d.h. ohne die Null, da es sonst Probleme bei der Berechnung von `wurzel_neu` gibt (Division durch Null nicht erlaubt).

```
# Aufgabe 44: Arbeit mit der Gleitkommaeinheit
#
# von: Stephan Brumme
#
# zuletzt geaendert: 22.Juni 2000

        .data
request_number:       .asciiz "Bitte geben Sie die natuerliche Zahl ein, aus der die
Quadratwurzel berechnet werden soll: "
result_part1:  .asciiz "\nDas Programm hat die Wurzel von "
result_part2:  .asciiz " in "
result_part3:  .asciiz " Iterationen berechnet. Die Wurzel von "
result_part4:  .asciiz " ist "
result_part5:  .asciiz ".\n"
error:         .asciiz "Die Wurzel kann nur von positiven Zahlen berechnet werden !\n"
einhalb:       .double 0.5
null:          .double 0.0

        .text 0x400000
main:
        li      $v0,4                 # Eingabeaufforderung
        la      $a0,request_number
        syscall

        li      $v0,5                 # Integerzahl einlesen
        syscall

        blez    $v0,negativ_error     # nur Zahlen >0 sind erlaubt

# ##############################################################################
# Registerbelegung (Variablenbezeichnungen gemaess Aufgabenstellung:
#
#     Integer:
#       $s0   -   n (Zahl, aus der die Wurzel gezogen wird)
#       $s1   -   Anzahl Iterationen
#
#     Double:
#       $f2/$f3 - n
#       $f4/$f5 - wurzel_alt
#       $f6/$f7 - wurzel_neu
#       $f8/$f9 - differenz
#       $f20/$f21, $f22/$f23 - Hilfsregister

        mtc1    $v0,$f2               # n in den Coprozessor kopieren
        cvt.d.w $f2,$f2               # die Integerzahl in Double umwandeln

        move    $s0,$v0               # n fuer spaetere Ausgabe aufbewahren
        li      $s1,0                 # Anzahl Iterationen feststellen

        l.d     $f20,einhalb          # Hilfsvariable laden (Konstante 0,5)
        l.d     $f22,null             # Hilfsvariable laden (Konstante 0,0)
        mov.d   $f4,$f2               # wurzel_alt=n
iteration:
        div.d   $f6,$f2,$f4           # wurzel_neu := n/wurzel_alt
        add.d   $f6,$f4,$f6           # wurzel_neu := wurzel_alt+n/wurzel_alt
```

```
        mul.d   $f6,$f20,$f6          # wurzel_neu := 0,5*(wurzel_alt+n/wurzel_alt)

        sub.d   $f8,$f4,$f6           # differenz := wurzel_alt-wurzel_neu

        mov.d   $f4,$f6               # wurzel_alt := wurzel_neu

        addi    $s1,$s1,1             # Anzahl Iterationen um 1 erhoehen

        c.le.d  $f8,$f22              # ist Differenz <=0 ?
        bc1f    iteration             # wenn nicht, dann naechste Iteration

# Ende Wurzelberechnung
# ###########################################################################

        li      $v0,4                 # 1.Text-Teil der Ausgabe
        la      $a0,result_part1
        syscall

        li      $v0,1                 # n als Integer ausgeben
        move    $a0,$s0
        syscall

        li      $v0,4                 # 2.Text-Teil der Ausgabe
        la      $a0,result_part2
        syscall

        li      $v0,1                 # Anzahl Iterationen als Integer ausgeben
        move    $a0,$s1
        syscall

        li      $v0,4                 # 3.Text-Teil der Ausgabe
        la      $a0,result_part3
        syscall

        li      $v0,1                 # n als Integer ausgeben
        move    $a0,$s0
        syscall

        li      $v0,4                 # 4.Text-Teil der Ausgabe
        la      $a0,result_part4
        syscall

        li      $v0,3                 # Wurzel von n als Double ausgeben
        mov.d   $f12,$f6
        syscall

        li      $v0,4                 # 5.Text-Teil der Ausgabe
        la      $a0,result_part5
        syscall

        li      $v0,10                # Programm beenden
        syscall

negativ_error:
        li      $v0,4                 # Fehlermeldung bei nicht-positiver eingegebener Zahl
        la      $a0,error
        syscall

        li      $v0,10                # Programm beenden
        syscall
```