



Technische Informatik II im SS 2007

## Musterlösungen zum 5. Übungsblatt

Prof. Dr. J. Henkel

Am Zirkel 2, Geb. 20.20  
D-76131 Karlsruhe

Dr.-Ing. T. Asfour

Telefon: +49-721-608-7379  
Fax: +49-721-608-8270  
Email: [asfour@ira.uka.de](mailto:asfour@ira.uka.de)  
<http://ti.ira.uka.de>

### Lösung 1

# Fliesskomma-Arithmetik

```
.data
cr_string: .asciiz "\n" # Sonderzeichen "neue Zeile"
eingabeA:  .asciiz "Fließkomma-Zahl A: "
eingabeB:  .asciiz "Fließkomma-Zahl B: "
eingabeC:  .asciiz "Fließkomma-Zahl C: "
result_sum: .asciiz "A + B + C = "
result_dif: .asciiz "A - B - C = "
result_mul: .asciiz "(A * B) + 16*C = "
result_div: .asciiz "(A / B) * 256 = "
error_str:  .asciiz "Division durch Null nicht definiert!\n"
```

```
.text
```

# Prozedur: Ausgabe eine Fließkomma-Zahl mit CR

```
print_dbl:  li $v0, 3
            syscall
            la $a0, cr_string
            li $v0, 4
            syscall
            jr $ra
```

# Prozedur: Ausgabe eines Strings

```
print_str:  li $v0, 4
            syscall
            jr $ra
```

# Beginn des Hauptprogrammes

```
.globl main
main:      subu $sp, $sp, 8          # Stack Frame ist 8 Bytes
           sw $ra, 0($sp)           # Sichern der Ruecksprungsadresse
           sw $fp, 4($sp)           # Sichern des alten Frame-Pointers
```

```
    addu $fp, $sp, 8           # neuen Frame-Pointer definieren

    la $a0, eingabeA          # Fliesskomma-Zahl A holen
    jal print_str
    li $v0, 7
    syscall
    mov.d $f2, $f0            # A in $f2 sichern

    la $a0, eingabeB          # Fliesskomma-Zahl B holen
    jal print_str
    li $v0, 7
    syscall
    mov.d $f4, $f0            # B in $f4 sichern

    la $a0, eingabeC          # Fliesskomma-Zahl C holen
    jal print_str
    li $v0, 7
    syscall
    mov.d $f6, $f0            # B in $f6 sichern

    la $a0, result_sum         # Ausgabe A + B + C
    jal print_str
    add.d $f8, $f2, $f4
    add.d $f12, $f6, $f8
    jal print_dbl

    la $a0, result_dif         # Ausgabe A - B - C
    jal print_str
    sub.d $f8, $f2, $f4
    sub.d $f12, $f8, $f6
    jal print_dbl

    la $a0, result_mul         # Ausgabe A * B + C
    jal print_str
    mul.d $f8, $f2, $f4
    add.d $f12, $f6, $f8
    jal print_dbl

    li.d $f8, 0.0
    c.eq.d $f4, $f8
    bclt error
    la $a0, result_div         # Ausgabe A / B
    jal print_str
    div.d $f12, $f2, $f4
    jal print_dbl
    b fertig

error:    la $a0, error_str     # Division durch Null
          jal print_str
```

```
fertig:    lw $ra, 0($sp)      # Ruecksprungadresse wiederherstellen
           lw $fp, 4($sp)     # Frame-Pointer wiederherstellen
           addu $sp, $sp, 8    # Stack-Frame loeschen
           jr $ra
```

## Lösung 2

1. Laden von 1111 0000 0011 1101 0000 1001 0000 1001 ins Register \$s0:

```
lui  $s0, 1111 0000 0011 1101  # load upper immediate
ori  $s0, 0000 1001 0000 1001
```

oder auch

```
lui  $s0, 1111 0000 0011 1101
addi $s0, $s0, 0000 1001 0000 1001
```

2. Die 2 niedrigstwertigen Bits einer Wortadresse haben den Wert 0
3. Register- und Speicherinhalte nach der Ausführung:

Registersatz		Hauptspeicher	
Register	Inhalt	Adresse	Inhalt
\$t0	0x10	\$0x20	0x22
\$t1	<b>0x40</b>	\$0x24	0x30
\$t2	0x16	\$0x28	0x40
\$t3	<b>0x20</b>	\$0x2C	0x50
\$t4	<b>0x30</b>	\$0x30	<b>0x30</b>

## Lösung 3

Das Programm ermittelt das maximale Element im Array a[36, 20, 27, 15, 1, 62, 41]. Zur Ermittlung des maximalen Elements wird eine if-then-Schleife benutzt.

```
.data
a:      .word 36, 20, 27, 15, 1, 62, 41
n:      .word 7
max:    .word 0

.text
.globl main

main:   li $t0, 0          # Array-Index i=0 und in $t0 speichern.
        li $s0, 0          # max=0 und in $s0 ablegen
        lw $s1, n          # Anzahl der Array-Elemente in $s1

m1:     bge $t0, $s1, m3
        mul $t1, $t0, 4     # i auf Wortgrenze skalieren
```

```

        lw $t2, a($t1)           # Lade a[i] ins Register $t2
        ble $t2, $s0, m2         # if a[i] <= max, dann "then-Teil"
        move $s0, $t2           # "then-Teil": max = a[i]
m2:     addi $t0, $t0, 1         # Array-Index i inkrementieren
        b m1
m3:     move $a0, $s0           # Ende der Schleife
        li $v0, 1
        syscall
        li $v0, 10
        syscall

```

#### Lösung 4

1. (a) Funktion des Programmstücks:

Addiert alle ungeraden Zahlen, die kleiner oder gleich  $n$  sind.

- (b) Wert im Register  $\$v0$  nach Abarbeitung des Programmstücks:

Wenn  $\$a0 = 9$  dann  $\$v0 = 25$

Wenn  $\$a0 = 10$  dann  $\$v0 = 25$

2.

Pseudobefehle	Echte Befehle
move \$t5, \$t3	addi \$t5, \$t3, 0
clear \$t5	add \$t5, \$zero, \$zero
bgt \$t5, \$t3, marke	slt \$at, \$t3, \$t5 bne \$at, \$zero, marke
bge \$t5, \$t3, marke	slt \$at, \$t3, \$t5 bne \$at, \$zero, marke beq \$t5, \$t3, marke

3. (a) `lw $s1, 100($s2)`:

Laden des Wortes (32-Bit) mit der Adresse  $(100 + \text{Inhalt des Registers } \$s2)$  ins Register  $\$s1$

- (b) `sw $s1, 100($s2)`:

Speichern des Wortes im Register  $\$s1$  an der Adresse  $(100 + \text{Inhalt des Registers } \$s2)$

- (c) `jal mystery`:

Unbedingter Sprung zur Marke `mystery` und Speicherung der Adresse des nächsten Befehls (Rücksprungadresse) im Register  $\$ra$