



Technische Informatik II im SS 2005

Musterlösungen zum 3. Übungsblatt

Prof. Dr. U. Brinkschulte

Geb. 40.28, D-76131 Karlsruhe

Email: brinks@ira.uka.de

Dr.-Ing. T. Asfour

Telefon: +49-721-608-7379

Fax: +49-721-608-8270

Email: asfour@ira.uka.de

<http://i61www.ira.uka.de/users/asfour/TI>

Lösung 1

1.

LDIV:

7. Takt: IR \rightarrow SAR; R = 1
8. Takt: R = 1
9. Takt: R = 1
10. Takt: SDR \rightarrow SAR; R = 1
11. Takt: R = 1
12. Takt: R = 1
13. Takt: SDR \rightarrow Akku

STIV:

7. Takt: IR \rightarrow SAR; R = 1
8. Takt: R = 1
9. Takt: R = 1
10. Takt: SDR \rightarrow SAR
11. Takt: Akku \rightarrow SDR; W = 1
12. Takt: W = 1
13. Takt: W = 1

2. ; Index of last item.

Last = \$00040

* = \$00080

; Fixed -1.

Neg1 DS -1

; Address of current item.

CurrentAdr DS

; Address of the current item belonging to the outer loop.

OuterAdr DS

; Address of the greatest number in the inner loop.

GreatestAdr DS

; Actual greatest number in the inner loop.

GreatestVal DS

* = \$00100

; Do a selection sort on the numbers.

Sort LDV Last

STV OuterAdr

```
; The outer loop, which loops through all items.
SortOuterLoop    STV  GreatestAdr
                  ADD  Neg1
                  JMN  SortOuterLoopEnd
                  STV  CurrentAdr
                  LDIV GreatestAdr
                  STV  GreatestVal

; The inner loop, which loops through the unfinished items (0..OuterAdr).
SortInnerLoop    ; Subtract the greatest value from the current value.
                  LDIV CurrentAdr
                  NOT
                  ADD  GreatestVal
                  NOT
                  ; If current value was less or equal, continue.
                  JMN  Next
                  ; The current value is greater. Update the greatest
                  ; value and index.
                  LDV  CurrentAdr
                  STV  GreatestAdr
                  LDIV CurrentAdr
                  STV  GreatestVal
Next             LDV  CurrentAdr
                  ; Decrement the current address.
                  ADD  Neg1
                  STV  CurrentAdr
                  ; Loop until the address is negative.
                  NOT
                  JMN  SortInnerLoop
SortInnerLoopEnd ; Now we have a greatest value, which we need to
                  ; exchange with the current item.
                  LDIV OuterAdr
                  STIV GreatestAdr
                  LDV  GreatestVal
                  STIV OuterAdr
                  ; Decrement the outer address.
                  LDV  OuterAdr
                  ADD  Neg1
                  STV  OuterAdr
                  ; Check is at the beginning of the loop.
                  JMP  SortOuterLoop
SortOuterLoopEnd HALT
```

Lösung 2

Fliesskomma-Arithmetik

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

main:      .globl main
           subu $sp, $sp, 8           # Stack Frame ist 8 Bytes
           sw $ra, 0($sp)             # Sichern der Ruecksprungadresse
           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
    bc1t 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 3

1. (a) bne \$s4, \$s3, label
 add \$s5, \$s4, \$s3
 label: ...

- (b) beq \$s4, \$s3, label1
 add \$s5, \$s4, \$s3
 j label2
 label1: sub \$s5, \$s4, \$s3
 label2:

- (c) slt \$s5, \$s3, \$s4