

1. Klausur: Technische Grundlagen der Informatik

Beachten Sie bitte folgende Hinweise:

- Es sind **keine** Hilfsmittel erlaubt (auch kein Taschenrechner!).
- Verwenden Sie zur Bearbeitung der Aufgaben nur das ausgeteilte Arbeitspapier. Versehen Sie **jedes Arbeitspapier** mit Ihrem **Namen** und **Vornamen** sowie Ihrer **Matrikelnummer** gemäß Studentenausweis.
- Verwenden Sie für **jede** Aufgabe einen **separaten** Bogen.
- Schreiben Sie **keine** Lösung auf die Angabe! Diese werden **nicht** gewertet.
- Schreiben Sie nicht mit roter oder grüner Farbe und nicht mit Bleistift.
- Die angegebenen Punkte für die einzelnen Aufgaben dienen als Richtschnur und sind nur vorläufig.
- **Geben** Sie die **vorliegende Angabe** am Ende der Klausur mit Ihren Lösungen **ab**!
- Legen Sie Ihren Personal- und Studentenausweis gut sichtbar neben sich.
- Vergessen Sie nicht, Ihre Programme ausreichend zu kommentieren.
- Falls nicht anders angegeben, besitzen alle atomaren Datentypen **Wort-**Format.

Bearbeitungszeit: 60 Minuten

Name:												
Vorname:												
Matrikelnummer:												

Bewertung

Aufgabe 1	max. 10 Pkt.	Pkt.	
Aufgabe 2	max. 10 Pkt.	Pkt.	
Aufgabe 3	max. 10 Pkt.	Pkt.	
Summe max. 30 Pkt.		Pkt.	

Aufgabe 1: Zahlendarstellung

(1.5+4.5+4 Pkt.)

a. Darstellung ganzer Zahlen im 2er-Komplement

Geben Sie die größte und die kleinste Zahl sowie die Null im 8-bit 2er-Komplement (Dualdarstellung!) an.

b. Addition im 2er-Komplement

Folgende Dualzahlen (8-bit) im 2er-Komplement sind gegeben:

10011100 und 01110110

- Addieren Sie die beiden Zahlen (in 2er-Komplement Darstellung).
- Hat bei der Addition ein Überlauf (overflow) stattgefunden? Begründen Sie **kurz** Ihre Antwort.
- Falls kein Überlauf stattgefunden hat, so stellen Sie das Ergebnis als Dezimalzahl dar.

c. Gleitpunktdarstellung

Nach dem IEEE 754 Standard ist die Berechnungsformel einer Gleitkommazahl bei einfacher Genauigkeit wie folgt:

$$(-1)^S \times (1 + \text{Significand}) \times 2^{(\text{Exponent}-127)}$$

wobei die 32-bit Darstellung folgendermaßen ist:

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
S	Exponent										Significand																				

- Wandeln Sie folgende rationale Zahl $(-\frac{3}{8})_{(10)}$ in Gleitkommadarstellung um.
- Welche der beiden folgenden Zahlen x und y ist größer? Begründen Sie **kurz** Ihre Antwort.

Zahl x

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0	1	1	0	1	0	0	1	1	1	1	0	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
S	Exponent								Significand																						

Zahl y

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0	1	1	1	0	0	1	1	1	1	0	1	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
S	Exponent								Significand																						

Antwort:

a. Darstellung ganzer Zahlen im 2er-Komplement

Die kleinste Zahl ist: 10000000

Die größte Zahl ist: 01111111

Die Null ist: 00000000

b. Addition im 2er-Komplement

- ```

10011100
+01110110
Übertrag 111111000

Ergebnis (1)00010010
^
-- Übertrag wird weggelassen

```

- (ii) Es hat kein Überlauf sondern nur ein Übertrag stattgefunden. Das Ergebnis bleibt im darstellbaren Bereich, da eine negative und eine positive Zahl addiert wurde (und das darstellbare Intervall bis auf eins symmetrisch zur Null ist).

(iii)  $00010010 \Rightarrow -(0) \times 2^7 + 0 \times 2^6 + 0 \times 2^5 + 1 \times 2^4 + 0 \times 2^3 + 0 \times 2^2 + 1 \times 2^1 + 0 \times 2^0 = 18$

c. **Gleitkommadarstellung:**

(i)  $(-\frac{3}{8})_{(10)} = -\frac{(11)_{(2)}}{2^3} = -(1.10)_{(2)} \times 2^{-2}$   
 $\Rightarrow S = 1, \text{Exponent} = -2 = 125 - 127$

|    |          |    |    |    |    |    |    |    |    |    |             |    |    |    |    |    |    |    |    |    |    |   |   |   |   |   |   |   |   |   |   |
|----|----------|----|----|----|----|----|----|----|----|----|-------------|----|----|----|----|----|----|----|----|----|----|---|---|---|---|---|---|---|---|---|---|
| 31 | 30       | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20          | 19 | 18 | 17 | 16 | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
| 1  | 0        | 1  | 1  | 1  | 1  | 1  | 0  | 1  | 1  | 0  | 0           | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| S  | Exponent |    |    |    |    |    |    |    |    |    | Significand |    |    |    |    |    |    |    |    |    |    |   |   |   |   |   |   |   |   |   |   |

- (ii) Beide Zahlen haben das gleiche Vorzeichen. Daraus folgt, man muss den Exponenten betrachten. Da der Exponent der Zahl y größer ist (sieht man einfach bei der Bias-Notation) als von x folgt:  
 $x < y$

**Aufgabe 2: Karnaugh-Verfahren**

(3+2+2+2+1 Pkt.)

Gegeben ist Funktion  $f : B_4 \rightarrow B$  mit folgender Funktionstabelle:

| a | b | c | d | f |
|---|---|---|---|---|
| 0 | 0 | 0 | 0 | 0 |
| 0 | 0 | 0 | 1 | 0 |
| 0 | 0 | 1 | 0 | 1 |
| 0 | 0 | 1 | 1 | 1 |
| 0 | 1 | 0 | 0 | 0 |
| 0 | 1 | 0 | 1 | 0 |
| 0 | 1 | 1 | 0 | 0 |
| 0 | 1 | 1 | 1 | 0 |
| 1 | 0 | 0 | 0 | 0 |
| 1 | 0 | 0 | 1 | 0 |
| 1 | 0 | 1 | 0 | 1 |
| 1 | 0 | 1 | 1 | 1 |
| 1 | 1 | 0 | 0 | 0 |
| 1 | 1 | 0 | 1 | 0 |
| 1 | 1 | 1 | 0 | 1 |
| 1 | 1 | 1 | 1 | 1 |

- Zeichnen Sie das zugehörige Karnaugh-Diagramm.
- Minimieren Sie durch Blockbildung unter Beachtung folgender Kriterien: die Anzahl der Blöcke soll möglichst klein sein und die Blöcke möglichst groß.
- Geben Sie die minimierte Funktion an.
- Wodurch unterscheiden sich zyklisch benachbarte Terme eines Karnaugh-Diagramms?
- Welches Gesetz erlaubt das mehrfache benutzen eines Terms (z.B. zum Minimieren)?

**Antwort:**

a.

|    |    |    |    |    |    |
|----|----|----|----|----|----|
|    |    | ab |    |    |    |
|    |    | 00 | 01 | 11 | 10 |
| cd | 00 |    |    |    |    |
|    | 01 |    |    |    |    |
|    | 11 | 1  |    | 1  | 1  |
|    | 10 | 1  |    | 1  | 1  |

b.

- $f(a, b, c, d) = ac + \bar{b}c$  oder  
 $f(a, b, c, d) = c(a + \bar{b})$

- Zyklisch benachbarte Terme unterscheiden sich nur dadurch, dass genau eine Variable in einem Term negiert ist und im anderen Term nicht negiert ist.
- Das Verschmelzungs- oder auch Idempotenzgesetz:  $x = x + x$ .

**Aufgabe 3: SPIM Programmieraufgabe**

(2+7+1 Pkt.)

Es soll ein **vollständiges** SPIM-Programm erstellt werden, dass einen eingelesenen ASCII-Zeichenstring in eine Integer-Zahl umwandelt.

Gehen Sie hierbei folgendermaßen vor:

- a. *Eingabe:* Es wird ein max. 10 Zeichen langer ASCII-String eingelesen. Der Anwender soll mit einem entsprechenden Anweisungstext zur Eingabe einer Zahl aufgefordert werden.
- b. *Hauptteil:* Der ASCII-Zeichenstring wird in eine Integerzahl umgewandelt. Beachten Sie hierzu folgendes:
  - Gehen Sie davon aus, dass der String nur Zahlen enthält.
  - Gehen Sie davon aus, dass das temporäre Register `$t2` die Anzahl der Ziffern im ASCII-String enthält.
  - Gehen Sie davon aus, dass das temporäre Register `$t0` die Adresse des letzten Zeichens im ASCII-String enthält (dies entspricht der Einer-Stelle der damit dargestellten Zahl).
  - Die ASCII-Codierung des Zeichens "0" beträgt 48; ab dieser Codierung sind aufsteigend die restlichen Ziffern 1 bis 9 abgelegt.
  - Legen Sie das Ergebnis der Konvertierung in Register `$s0` ab.
- c. *Ausgabe:* Das Ergebnis wird als Integer wieder ausgegeben.

**Hinweise:**

- Vergessen Sie nicht, notwendigen Speicher zu reservieren!
- Verwenden Sie weitere Register und definieren Sie notwendige Marken.
- **Kommentieren** Sie ihr Programm ausreichend!
- Auf der nächsten Seite befindet sich eine Liste von SPIM-Befehlen.

## SPIM Assemblerbefehle

| Befehl  | Argumente       | Wirkung                                 |
|---------|-----------------|-----------------------------------------|
| add     | Rd, Rs1, Rs2    | $Rd := Rs1 + Rs2$                       |
| addu    | Rd, Rs1, Rs2    | $Rd := Rs1 + Rs2$                       |
| addi    | Rd, Rs1, Imm    | $Rd := Rs1 + Imm$                       |
| addiu   | Rd, Rs1, Imm    | $Rd := Rs1 + Imm$                       |
| sub     | Rd, Rs1, Rs2    | $Rd := Rs1 - Rs2$                       |
| sub     | Rd, Rs1, Imm    | $Rd := Rs1 - Imm$                       |
| mul     | Rd, Rs1, Rs2    | $Rd := Rs1 \times Rs2$                  |
| mul     | Rd, Rs1, Imm    | $Rd := Rs1 \times Imm$                  |
| b       | label           | Sprung nach label                       |
| j       | label           | Sprung nach label                       |
| jr      | Rs              | unbedingter Sprung an die Adresse in Rs |
| beq     | Rs1, Rs2, label | Sprung, falls $Rs1 = Rs2$               |
| beqz    | Rs, label       | Sprung, falls $Rs = 0$                  |
| bne     | Rs1, Rs2, label | Sprung, falls $Rs1 \neq Rs2$            |
| bnez    | Rs1, label      | Sprung, falls $Rs1 \neq 0$              |
| bge     | Rs1, Rs2, label | Sprung, falls $Rs1 \geq Rs2$            |
| bgeu    | Rs1, Rs2, label | Sprung, falls $Rs1 \geq Rs2$            |
| bge     | Rs, label       | Sprung, falls $Rs \geq 0$               |
| bgt     | Rs1, Rs2, label | Sprung, falls $Rs1 > Rs2$               |
| bgtu    | Rs1, Rs2, label | Sprung, falls $Rs1 > Rs2$               |
| bgtz    | Rs, label       | Sprung, falls $Rs > 0$                  |
| ble     | Rs1, Rs2, label | Sprung, falls $Rs1 \leq Rs2$            |
| bleu    | Rs1, Rs2, label | Sprung, falls $Rs1 \leq Rs2$            |
| blez    | Rs, label       | Sprung, falls $Rs \leq 0$               |
| blt     | Rs1, Rs2, label | Sprung, falls $Rs1 < Rs2$               |
| bltu    | Rs1, Rs2, label | Sprung, falls $Rs1 < Rs2$               |
| bltz    | Rs, label       | Sprung, falls $Rs < 0$                  |
| syscall |                 | führt Systemfunktion aus                |
| move    | Rd, Rs          | $Rd := Rs$                              |
| la      | Rd, label       | Adresse des Labels wird in Rd geladen   |
| lb      | Rd, Adr         | $Rd := \text{MEM}[\text{Adr}]$          |
| lw      | Rd, Adr         | $Rd := \text{MEM}[\text{Adr}]$          |
| li      | Rd, Imm         | $Rd := Imm$                             |
| sw      | Rs, Adr         | $\text{MEM}[\text{Adr}] := Rs$          |

| Funktion     | Code in \$v0 | Funktion    | Code in \$v0 |
|--------------|--------------|-------------|--------------|
| print_int    | 1            | read_float  | 6            |
| print_float  | 2            | read_double | 7            |
| print_double | 3            | read_string | 8            |
| print_string | 4            | sbrk        | 9            |
| read_int     | 5            | exit        | 10           |

**Antwort:** Siehe nächste Seite!

```

.data
2 string: .space 11 # Reserviere 11 Bytes (= Laenge + Nulltermination)
 prompt: .asciiz "Geben_Sie_den_Zeichenstring_ein:_"
4 erg: .asciiz "Das_Ergebnis_als_Integer_lautet:_"

6 # Registerbelegungen:
 # $t0 = Adresse des eingelesenen ASCII-Strings (Pointer)
8 # $t1 = enthaelt immer ein Zeichen des Strings
 # $t2 = Zeichen-Zaehler
10 # $t3 = temp. Variable, enthaelt die Dezimal-Stelle
 # $t4 = extrahierte Ziffer
12 # $t5 = errechnete ganze Dezimalstelle
 # $s0 = umgewandelter Integer (Ergebnis)
14

.text
16 main:
 #####
18 ## Einlesen des ASCII-Strings
 #####
20
22 li $v0, 4 # 4: print_str
 la $a0, prompt # Adresse der 1. auszugebenden Zeile nach $a0
 syscall
24
26 la $a0, string # Zieladresse
 li $a1, 10 # Maximale Laenge des Strings 10
 li $v0, 8 # 8: read_str
 syscall # einlesen
28
30 #####
32 ## Umwandlung in Integer
 #####
34
36 # Folgendes musste NICHT programmiert werden
 # Dient der Feststellung der Laenge des Strings
 la $t0, string # Adresse des Strings nach $t0
 li $t2, 0 # $t2 ist der Zeichenzaehler (mit 0 init.)
38 nextCh:
 lb $t1, ($t0) # Ein Zeichen/Byte des Strings einlesen
40 beqz $t1, atoi # Null heisst Ende des Strings erreicht
 addi $t2, $t2, 1 # Zeichen-Zaehler erhoehen
42 addi $t0, $t0, 1 # Pointer auf naechstes Zeichen setzen
 j nextCh # Sprung zum Schleifenanfang
44 atoi:
 sub $t0, $t0, 2 # Pointer auf letztes Zeichen=erste Ziffer setzen
46 sub $t2, $t2, 1 # Zeichen-Zaehler entsprechend anpassen
 #----- Hauptteil, der zu programmieren ist -----
48 li $t3, 1 # temp. Variable mit 1 initialisieren, wird fuer
 # die Dezimalstelle (1,10,100,1000,etc.) verwendet
50 li $s0, 0 # $s0 ist Ergebnisregister (mit 0 init.)
 loop:
52 beqz $t2, ausg # Falls Zeichen-Zaehler=0, dann Ende
 lb $t4, ($t0) # Zeichen nach $t4
54 sub $t4, $t4, 48 # "Ziffer" extrahieren, indem die ASCII-Stelle
 # fuer das Zeichen "0" abgezogen wird
56 mul $t5, $t4, $t3 # Zahl mit der Dezimal-Stelle multipl. (also z.B.
 # 1, 10, 100, 1000, etc.) und in $t5 ablegen
58 add $s0, $s0, $t5 # zum bisherigen Wert addieren

```

```
 mul $t3, $t3, 10 # naechste Dez.stelle errechnen
60 sub $t2, $t2, 1 # Zeichen-Zaehler erniedrigen
 sub $t0, $t0, 1 # Pointer auf naechste Stelle setzen
62 j loop
#----- Ende Hauptteil -----
64 #####
 ## Ausgabe des Ergebnisses
66 #####

68 ausg:
 # Folgende Ausgabe eines Strings musste NICHT programmiert werden!
70 li $v0, 4 # 4: print_str
 la $a0, erg # Adresse der 1. auszugebenden Zeile nach $a0
72 syscall # ausgeben

74 li $v0, 1 # 1: print_int
 move $a0, $s0 # Adresse des Ergebnis-Texts nach $a0
76 syscall # ausgeben

78 li $v0, 10 # Systemaufrufnr. 10 = EXIT
 syscall
```