**Platznummer hier eintragen!** ⇒

LUDWIG-MAXIMILIANS-UNIVERSITÄT MÜNCHEN
INSTITUT FÜR INFORMATIK
LEHRSTUHL FÜR MOBILE UND VERTEILTE SYSTEME
PROF. DR. CLAUDIA LINNHOFF-POPIEN

Sommersemester 2011
Klausur
26.07.2011

# Klausur zur Vorlesung Rechnerarchitektur

Beachten Sie die folgenden Hinweise:

- Tragen Sie oben rechts Ihre **Platznummer** ein! Tragen Sie unten Ihre persönlichen Daten ein.
- Zur Klausur sind **keine** Hilfsmittel erlaubt, bei Schwierigkeiten mit der deutschen Sprache darf ein Wörterbuch verwendet werden.
- Tragen Sie Ihre Lösungen **direkt in dieses Klausurheft** ein. Schreiben Sie nicht mit roter oder grüner Farbe und nicht mit Bleistift. Lösungswege und Rechnungen auf den karrierten Bögen (Schmierpapier) werden **nicht berücksichtigt**!
- Die Punkte für die einzelnen Aufgaben dienen nur als vorläufige Richtlinie.
- **Geben** Sie am Ende der Klausur Ihr Klausurheft, Ihre karierten Bögen und Ihre Platznummer **ab**!
- Um Ihre Klausur zu entwerten, streichen Sie bitte deutlich das **Deckblatt und alle Seiten des Klausurenheftes vor der Abgabe durch**. Die Klausur wird dann nicht korrigiert und auch nicht als Prüfungsversuch gewertet.
- Legen Sie Ihren amtlichen Lichtbild– und Studentenausweis gut sichtbar neben sich.

## Bearbeitungszeit: 120 Minuten

| | |
|---|---|
| **Name:** | |
| **Vorname:** | |
| **Matrikelnummer:** | |

Bewertung

| | | | |
|---|---|---|---|
| Aufgabe 1 | max. 40 Pkt. | Pkt. | |
| Aufgabe 2 | max. 15 Pkt. | Pkt. | |
| Aufgabe 3 | max. 25 Pkt. | Pkt. | |
| Aufgabe 4 | max. 12 Pkt. | Pkt. | |
| Aufgabe 5 | max. 28 Pkt. | Pkt. | |
| Summe max. 120 Pkt. | | Pkt. | |

**Hinweise zu den Multiple Choice Aufgaben:**

- Pro Aussage müssen Sie entscheiden, ob die Aussage **richtig** oder **falsch** ist. Das entsprechende Kästchen markieren Sie bitte **deutlich** mit einem **X**.
- Es werden lediglich die Aussagen/Zeilen gewertet, die *genau ein* **X** enthalten. Für eine richtige Antwort gibt es dabei einen Punkt, für eine falsche Antwort wird ein Punkt abgezogen.
- Aussagen, bei denen entweder keine oder beide Alternativen gekennzeichnet sind, werden mit 0 Punkten gewertet.
- Undeutliche Kennzeichnungen werden im Zweifelsfall zu Ihrem Nachteil gewertet.
- Alle Aussagen sind innerhalb eines Blocks gleich gewichtet und werden jeweils mit 1 Punkt gewertet.
- Man kann auf einen Block im schlechtesten Fall insgesamt 0 Punkte erhalten.
- Ein Block von Aussagen kann *keine*, *eine*, oder *mehrere* richtige Aussagen enthalten.

Fragebeispiel – Hauptstädte:

| Sind folgende Aussagen zum Thema **Hauptstädte** richtig (r) oder falsch (f)? | | **r** | **f** |
|---|---|---|---|
| a | Berlin liegt in Deutschland. | | |
| b | Paris liegt in Deutschland. | | |
| c | London liegt nicht in Europa. | | |
| d | Rom liegt in Spanien. | | |
| e | Paris ist eine Hauptstadt | | |

Antwortbeispiel:

| Sind folgende Aussagen zum Thema **Hauptstädte** richtig (r) oder falsch (f)? | | **r** | **f** | Bewertung |
|---|---|---|---|---|
| a | Berlin liegt in Deutschland. | X | | (korrekt $\rightarrow$ 1 Punkt) |
| b | Paris liegt in Deutschland. | X | | (nicht korrekt $\rightarrow -1$ Punkt) |
| c | London liegt nicht in Europa. | | | (nicht bearbeitet $\rightarrow$ 0 Punkte) |
| d | Rom liegt in Spanien. | X | X | (falsch bearbeitet $\rightarrow$ 0 Punkte) |
| e | Paris ist eine Hauptstadt | X | | (korrekt $\rightarrow$ 1 Punkt) |

Das Ergebnis wäre in diesem Fall: $(+1 - 1 + 0 + 0 + 1 =)$ 1 Punkt (von maximal möglichen 5).

Die **korrekte** Lösung lautet:

| Sind folgende Aussagen zum Thema **Hauptstädte** richtig (r) oder falsch (f)? | | **r** | **f** | Bewertung |
|---|---|---|---|---|
| a | Berlin liegt in Deutschland. | X | | (korrekt $\rightarrow$ 1 Punkt) |
| b | Paris liegt in Deutschland. | | X | (korrekt $\rightarrow$ 1 Punkt) |
| c | London liegt nicht in Europa. | | X | (korrekt $\rightarrow$ 1 Punkt) |
| d | Rom liegt in Spanien. | | X | (korrekt $\rightarrow$ 1 Punkt) |
| e | Paris ist eine Hauptstadt | X | | (korrekt $\rightarrow$ 1 Punkt) |

# **Aufgabe 1:** Multiple Choice

(35 Pkt.)

| Sind folgende Aussagen zum Thema **Komplementdarstellungen** richtig (r) oder falsch (f)? | | **r** | **f** |
|---|---|---|---|
| a | Mit einer 2-Bit-Zahl in Einerkomplement-Darstellung kann man genau 3 verschiedene ganze Zahlen darstellen. | | |
| b | 1000000 ist die größte positive Zahl in 8-bit 2er-Komplementdarstellung. | | |
| c | Für jede im 2er-Komplement dargestellt n-stellige Ganzzahl $z$ gilt, dass auch $-z$ mit n Stellen dargestellt werden kann. | | |
| d | Beim Zweierkomplement geht das höchstwertige Bit $x_{n-1}$ einer n-stelligen Dualzahl mit dem Wert $-x_{n-1} * 2^{n-1}$ in den Wert der Zahl ein. | | |
| e | Es gibt mindestens eine Zahl, welche in 1er- und 2er-Komplementdarstellung durch dieselbe Bitfolge repräsentiert wird. | | |

| Sind folgende Aussagen zum Thema **Darstellung von Informationen** richtig (r) oder falsch (f)? | | **r** | **f** |
|---|---|---|---|
| a | Um ein Schwarz/Weiß-Bild mit der Abmessung $m * n$ Pixel unkomprimiert zu speichern, benötigt man $\lfloor \log_2(m * n) \rfloor$ Bit.[1] | | |
| b | Unicode ist abwärtskompatibel zu ASCII. | | |
| c | Jede vier Bit lange Binärzahl lässt sich durch eine Oktalziffer ausdrücken. | | |
| d | Zeichen in Unicode sind immer 16 bit lang. | | |
| e | Eine Gleitkommazahl der Basis b heißt normalisiert, wenn gilt $1 <= \|m\| <= 2$ (m bezeichnet die Mantisse) | | |

| Sind folgende Aussagen zum Thema **Boolschen Algebra** richtig (r) oder falsch (f)? | | **r** | **f** |
|---|---|---|---|
| a | Es gilt die boolesche Umformung: $A.B + -A. - B = -(-A.B + A. - B)$ | | |
| b | $\{NAND\}$ ist funktional vollständig. | | |
| c | Die Menge $\{+, -\}$ von Boole'schen Funktionen ist vollständig. | | |
| d | Es gibt genau fünf verschiedene zweistellige Boolesche Operationen. | | |
| e | Eine disjunktive Normalform hat die Gestalt $(a_1^1 + ... + a_1^{n_1}) \cdot ... \cdot (a_k^1 + ... + a_k^{n_k})$ | | |

| Sind folgende Aussagen zum Thema **SPIM** richtig (r) oder falsch (f)? | | **r** | **f** |
|---|---|---|---|
| a | Der Stack wächst mit geringer werdenden Hauptspeicheradressen in Richtung der Adresse 0. | | |
| b | Programmcode und Stack teilen sich einen gemeinsamen Speicher. | | |
| c | Der Stackpointer $sp zeigt auf das Wort, das zuletzt in den Stack geladen wurde. | | |
| d | Wegen seiner dynamischen Zellbreite eignet sich der Stack zur Speicherung von Daten, für die die statische Breite der Register (32 Bit) nicht ausreicht. | | |
| e | Beim Speichern auf dem Stack wird der Stackpointer automatisch verschoben. | | |

[1] Für eine reelle Zahl x ist $\lfloor x \rfloor$ die größte ganze Zahl, die kleiner oder gleich x ist.

| Sind folgende Aussagen zum Thema **Speicher** richtig (r) oder falsch (f)? | | **r** | **f** |
|---|---|---|---|
| a | Als Cache-Hit bezeichnet man einen besonders effektiven Cache-Algorithmus. | | |
| b | Um mit einem parallelen Adressbus die vierfache Datenmenge adressieren zu können, werden zwei weitere Adressleitungen benötigt. | | |
| c | Die Speicherhierarchie beschreibt, wer welchen Speicher benutzen darf. | | |
| d | Ein „kalter“ Cache hat viele Misses. | | |
| e | Die elementare Speicherzelle eines DRAM-Bausteins besteht aus einem Transistor und einem Kondensator. | | |

| Sind folgende Aussagen zum Thema **Prozessorarchitekturen** richtig (r) oder falsch (f)? | | **r** | **f** |
|---|---|---|---|
| a | Amdahls Gesetz über die Verbesserung der Rechenzeit durch Verwendung von Caches besagt: $\mathtt{VerbesserteRechenzeit} = \frac{\mathtt{Rechenzeit}}{\mathtt{Verbesserungsfaktor}}$ | | |
| b | Eine unbedingte Sprunganweisung wird ausgeführt, indem die Adresse aus dem rechten Teil des CIR in das SCR kopiert wird. | | |
| c | Bei der von-Neumann-Architektur gibt es keine Trennung von Daten und Programmen. | | |
| d | Vektorrechner werden als SIMD (Single-Instruction-Multiple-Data) Rechersysteme klassifiziert. | | |
| e | Jedes Programm läuft auf einem Mehrkernprozessor immer schneller, als auf einem Einkernprozessor. | | |

| Sind folgende Aussagen zum Thema **Pipelining** richtig (r) oder falsch (f)? | | **r** | **f** |
|---|---|---|---|
| a | Um Pipelining bei der Auswertung von Instruktionen verwenden zu können, müssen sich Instruktionen in mehrere Stufen zerlegen lassen. | | |
| b | Control-Hazards (Kontrollflußkonflikte) können entstehen, wenn während der Ausführung eines Sprungbefehls schon weitere nachfolgende Befehle in die Pipeline geladen wurden. | | |
| c | Pipelining erhöht die Geschwindigkeit der Ausführung einer Instruktion (= Assemblerbefehl) | | |
| d | Forwarding (Bypassing) lässt sich zwischen jeder Zweierkombination von Pipelinestufen verschiedener Instruktionen realisieren. | | |
| e | Da MIPS-Prozessoren kein Forwarding (Bypassing) unterstützen, können bei der MIPS-Architektur keine Strukturkonflikte (Structural Hazards) entstehen. | | |

| Sind folgende Aussagen zum Thema **Vermischtes** richtig (r) oder falsch (f)? | | **r** | **f** |
|---|---|---|---|
| a | Nach IEEE-754 hat der Significand in Double Precision 52 Bit | | |
| b | Die Dezimalzahl 0.1 lässt sich als binäre Zahl nicht exakt darstellen. | | |
| c | Hamming-Codes dienen zur Komprimierung von Daten. | | |
| d | Bei einer KKNF hat jeder Faktor gleich viele Summanden. | | |
| e | Ein Multiplexer mit n Eingangsleitungen und einer Ausgangsleitung benötigt $\lfloor \log(n) \rfloor$ Auswahlleitungen.[2] | | |

[2] Für eine reelle Zahl x ist $\lfloor x \rfloor$ die größte ganze Zahl, die kleiner oder gleich x ist.

## **Aufgabe 2:** Zahlendarstellung

(15 Pkt.)

Beantworten Sie folgende Fragen im Bezug auf die Dualdarstellung von Ganzzahlen und Gleitkommazahlen:

a. Erläutern Sie den Unterschied zwischen den Begriffen Zweierkomplement und Zweierkomplementdarstellung.

b. Geben Sie jeweils die Zweierkomplementdarstellung der beiden Ganzzahlen $x = -103$ und $y = -18$ an. Verwenden Sie zur Darstellung jeweils 8 Bit.

c. Berechnen Sie die Zweierkomplementdarstellung der Zahl $z = x - y$ ($x$ und $y$ mit den Werten aus Aufgabe b). Der Rechenweg muss ersichtlich sein!

d. Kann man feststellen, ob bei der Berechnung $z = x - y$ aus Aufgabe c) ein Überlauf (Overflow) stattfinden wird, bevor man die Berechnung ausführt? Begründen Sie kurz Ihre Antwort. Hat bei der Berechnung ein Überlauf (Overflow) stattgefunden?

e. Erläutern Sie kurz, warum man bei der Darstellung einer Gleitkommazahl nach dem Standard IEEE 754 die Bias-Notation verwendet?

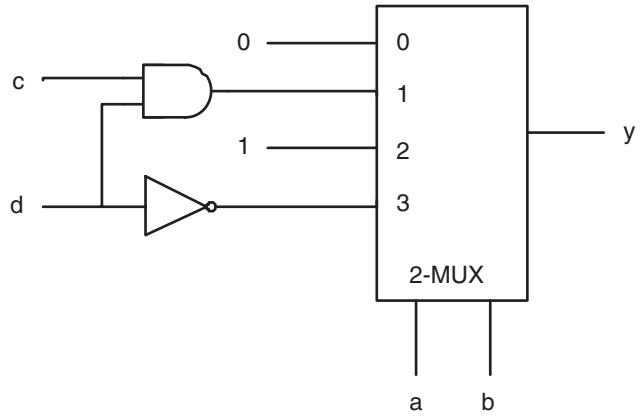
f. Wandeln Sie folgende Zahl, die in Gleitkommadarstellung (IEEE 754) gegeben ist, in ihre Dezimaldarstellung um. Sie dürfen das Ergebnis auch in Bruchdarstellung angeben.

| 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 | 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 | | | | | | | | | | | | | | | | | | | | | | |

## **Aufgabe 3:** Boolesche Algebra und Schaltungsentwurf

(25 Pkt.)

a. Gegeben sei das folgende Multiplexer-Schaltnetz:




(i) Ermitteln Sie die Funktionswerte für alle möglichen Kombinationen von Eingangsbelegungen und vervollständigen Sie damit die folgende Wahrheitstabelle:

| a | b | c | d | y = f(a,b,c,d) |
|---|---|---|---|---|
| 0 | 0 | 0 | 0 | |
| 0 | 0 | 0 | 1 | |
| 0 | 0 | 1 | 0 | |
| 0 | 0 | 1 | 1 | |
| 0 | 1 | 0 | 0 | |
| 0 | 1 | 0 | 1 | |
| 0 | 1 | 1 | 0 | |
| 0 | 1 | 1 | 1 | |
| 1 | 0 | 0 | 0 | |
| 1 | 0 | 0 | 1 | |
| 1 | 0 | 1 | 0 | |
| 1 | 0 | 1 | 1 | |
| 1 | 1 | 0 | 0 | |
| 1 | 1 | 0 | 1 | |
| 1 | 1 | 1 | 0 | |
| 1 | 1 | 1 | 1 | |

(ii) Geben Sie die Funktion `y = f(a,b,c,d)` in kanonischer disjunktiver Normalform (KDNF) an.

(iii) Vereinfachen Sie den Funktionsterm aus Aufgabe aii) soweit wie möglich. Geben Sie dabei **jeden** Vereinfachungsschritt an und kennzeichnen Sie, welche Klauseln Sie zur Vereinfachung verwendet haben. Nummerieren Sie dazu Ihre Klauseln! Halten Sie sich an das Schema aus dem folgenden Beispiel!

**Beispiel:** Vereinfachung der Formel `a.b.c + a.-b.c + -a.-b. c`

```
 1:  a. b. c
 2:  a.-b. c
 3: -a.-b. c
----------------------------------------
 4:  a.    c (Vereinfachung aus 1 und 2)
 5:    -b. c (Vereinfachung aus 2 und 3)
----------------------------------------
Ergebnis: (a.c) + (-b.c)
```

b. Betrachten Sie nun die Boolsche Formel

```
-a.-b.d + b.c.d + a.c.d
```

Diese Formel lässt sich mittels der deduktiven Vereinfachung zu einer einzigen Klausel vereinfachen. Geben Sie diese Klausel an und zeigen Sie mit einem Beweis durch Widerspruch, dass diese Vereinfachung gilt.

**Achtung:** Gehen Sie bei Ihrem Beweis schematisch nach dem Beispiel aus Aufgabe aiii) vor, d.h. nummerieren Sie die einzelnen Klauseln und kennzeichnen Sie bei jeder Vereinfachung, welche Klauseln Sie dazu verwendet haben!

c. Was versteht man im Rahmen der Booleschen Algebra unter dem Dualitätsprinzip?

## **Aufgabe 4:** Pipelining und Konflikte

(12 Pkt.)

Gehen Sie für alle Teilaufgaben von einer 5–stufigen Pipeline mit einer Ausführungszeit von 2ns pro Stufe aus. In der Folgende Tabelle sind die Befehle und deren Ausführungszeiten gegeben, aus denen sich die entsprechenden Assemblerinstruktionen zusammensetzen:

| Befehl | IF | ID | EX | MEM | WB |
|---|---|---|---|---|---|
| load word (lw) | 2 ns | 1 ns | 2 ns | 2 ns | 1 ns |
| store word (sw) | 2 ns | 1 ns | 2 ns | 2 ns | — |
| add, sub | 2 ns | 1 ns | 2 ns | — | 1 ns |

Eine Pipeline besitzt also die Stufen:

**IF:** Instruction Fetch (Instruktion holen)

**ID:** Instruction Decode (Register lesen)

**EX:** EXecution or Address Calculation (ALU Operation ausführen)

**MEM:** Data MEMory Access (Datenzugriff ausüben)

**WB:** Write Back (Register Schreiben)

Beantworten Sie nun unter Annahme dieser Charakteristik einer Pipeline die folgenden Fragen.

a. Wie bestimmt man die Länge einer Pipelinestufe, d.h. wie lang muss das Zeitintervall für eine Stufe mindestens sein?

b. Warum müssen die Pipelinestufen eine gleich lange Ausführungszeit besitzen?

c. Von welchen zwei Eigenschaften hängt der Leistungsgewinn einer idealen Pipeline ab (also ohne Berücksichtigung von Konflikten)?

d. Gegeben ist folgendes Programmfragment:

```
1 lw      $2, 100($5)
2 add     $3, $3, $4
3 add     $1, $4, $5
```

(i) Wie lang dauert die Ausführung dieses Programmfragments **ohne** Pipelining?

(ii) Wie lang dauert die Ausführung dieses Programmfragments **mit** Pipelining?

e. Gegeben sei folgendes Programmfragment:

```
1 add     $2, $3, $4
2 add     $7, $3, $6
3 sub     $5, $2, $6
```

Benennen und beschreiben Sie den Konflikt, der hier auftreten kann. Benennen Sie zudem die Stufe in der Pipeline der jeweiligen Instruktion, die den Konflikt jeweils provoziert.

f. Geben Sie zwei Möglichkeiten an, Wie man den Konflikt aus Aufgabe e) lösen kann. Ihre Lösung soll weiterhin Pipelining verwenden!

g. Gegeben sei folgendes Programmfragment:

```
1 sw      $7, 100($1)
2 add     $2, $3, $6
3 lw      $8, 100($1)
```

Falls hier ein Konflikt möglich ist, dann benennen und erläutern Sie den Konflikt. Benennen Sie auch wieder die Stufe in der Pipeline der jeweiligen Instruktion, die den Konflikt provoziert. Falls hier kein Konflikt möglich ist, dann erläutern Sie kurz, worin sich die Situation von der aus Aufgabe e) unterscheidet?

## **Aufgabe 5:** SPIM Programmierung

(28 Pkt.)

Für die Bearbeitung dieser Aufgabe finden Sie am Ende dieses Klausurheftes einen reduzierte Befehlsreferenz. Die Bearbeitung der Teilaufgaben a) – d) beziehen sich alle auf das folgende Assembler Programm.

```
1          .data
2  in1:    .space  41
3  in2:    .space  2
4  prmpt1: .asciiz "Input1: "
5  prmpt2: .asciiz "Input2: "
6
7          .text
8  main:
9          li      $v0, 4
10         la      $a0, prmpt1
11         syscall
12
13         li      $v0, 8
14         la      $a0, in1
15         li      $a1, 40
16         syscall
17
18         li      $v0, 4
19         la      $a0, prmpt2
20         syscall
21
22         li      $v0, 8
23         la      $a0, in2
24         li      $a1, 2
25         syscall
26
27         la      $t0, in2
28         lb      $s0, 0($t0)
29
30         la      $s1, in1
31 loop:
32         lb      $t0, 0($s1)
33         beqz    $t0, go_on
34
35
36         xor     $a0, $t0, $s0
37         sb      $a0, 0($s1)
38
39         addi    $s1, $s1, 1
40         j       loop
41 go_on:
42         li      $v0, 4
43         la      $a0, in1
44         syscall
45 exit:
46         li      $v0, 10
47         syscall
```

a. Kommentieren Sie in der nachfolgende Tabelle ausführlich, was die Ausführung der entsprechenden referenzierten Codezeilen genau bewirkt.
Formulieren Sie Ihre Kommentare so, dass klar wird, mit welchem Ziel der Programmierer diese Zeilen geschrieben hat!

| Codezeile(n) | Kommentar |
|---|---|
| 9 – 11 | |
| 13 – 16 | |
| 24 | |
| 27 | |
| 28 | |

| Codezeile(n) | Kommentar |
|---|---|
| 32 | |
| 33 | |
| 36 | |
| 37 | |
| 39 | |
| 42 – 43 | |
| 46 – 47 | |

b. Beschreiben Sie in wenigen Sätzen, was das Programm bewirkt.

c. Erläutern Sie, warum in Zeile 24 als letztes Argument eine 2 angegeben wird?

d. Es ist möglich, dass man eine frühere Eingabe des Programms rekonstruiert, wenn man die dazugehörige Ausgabe des Programms kennt. Welche zwei Bedingungen müssen dazu erfüllt sein? Welche boolsche Gesetzmäßigeit macht man sich hier zu Nutze?

e. Das nachfolgende Assembler Programm soll die Länge der im Datensegment abgelegten Zeichenkette “hello world” berechnen.

Vervollständigen Sie den gekennzeichneten Programmteil so, dass das Programm das gewünschte Ergebnis liefert. Verwenden Sie dazu **ausschließlich** die Befehle, die auf der Befehlsreferenz am Ende dieses Klausurhefts angegeben sind! Kommentieren Sie **jede** Zeile ihrer Lösung!

**Achtung:** Verwenden Sie das Register `$t0` zum Zwischenspeichern eines eingelesenen Zeichens. Verwenden Sie das Register `$t1` zum Zwischenspeichern der Anzahl gelesener Zeichen. Fügen Sie keine neuen Sprungmarken ein!

```
1          .data
2  str:    .asciiz "hello world"
3  ans:    .asciiz "Length is "
4  endl:   .asciiz "\n"
5
6          # t0 - Zum Zwischenspeichern eines Zeichens
7          # t1 - Zum Zwischenspeichern der Anzahl gelesener Zeichen
8
9          .text
10 main:
11         la      $t2,str
12         li      $t1,0
13 nextCh:
14         ################ Beginn Ihrer Lösung ################
15
16
17
18
19
20
21
22
23
24
25
26
27
28
29
30
31
32
33         ################ Ende Ihrer Lösung ##################
34 strEnd: la      $a0,ans
35         li      $v0,4
36         syscall
37
38         move    $a0,$t1
39         li      $v0,1
40         syscall
41
42         la      $a0,endl
43         li      $v0,4
44         syscall
45
46         li      $v0,10
47         syscall
```

**Überblick über die wichtigsten SPIM Assemblerbefehle**

| Befehl | Argumente | Wirkung |
|---|---|---|
| add | Rd, Rs1, Rs2 | Rd := Rs1 + Rs2 (mit Überlauf) |
| sub | Rd, Rs1, Rs2 | Rd := Rs1 - Rs2 (mit Überlauf) |
| addu | Rd, Rs1, Rs2 | Rd := Rs1 + Rs2 (ohne Überlauf) |
| subu | Rd, Rs1, Rs2 | Rd := Rs1 - Rs2 (ohne Überlauf) |
| addi | Rd, Rs1, Imm | Rd := Rs1 + Imm |
| addiu | Rd, Rs1, Imm | Rd := Rs1 + Imm (ohne Überlauf) |
| div | Rd, Rs1, Rs2 | Rd := Rs1 DIV Rs2 |
| rem | Rd, Rs1, Rs2 | Rd := Rs1 MOD Rs2 |
| mul | Rd, Rs1, Rs2 | Rd := Rs1 $\times$ Rs2 |
| b | label | unbedingter Sprung nach label |
| j | label | unbedingter Sprung nach label |
| jal | label | unbed.Sprung nach label, Adresse des nächsten Befehls in $ra |
| 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 |
| bgez | 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 |
| not | Rd, Rs1 | Rd := $\neg$ Rs1 (bitweise Negation) |
| and | Rd, Rs1, Rs2 | Rd := Rs1 & Rs2 (bitweises UND) |
| or | Rd, Rs1, Rs2 | Rd := Rs1 \| Rs2 (bitweises ODER) |
| xori | Rd, Rs1, Imm | Rd := Rs1 $\nleftrightarrow$ Imm (bitweises XOR) |
| syscall | | führt Systemfunktion aus |
| move | Rd, Rs | Rd := Rs |
| la | Rd, label | Adresse des Labels wird in Rd geladen |
| lb | Rd, Adr | Rd := MEM[Adr] |
| lw | Rd, Adr | Rd := MEM[Adr] |
| li | Rd, Imm | Rd := Imm |
| sw | Rs, Adr | MEM[Adr] := Rs (Speichere ein Wort) |
| sh | Rs, Adr | MEM[Adr] MOD $2^{16}$ := Rs (Speichere ein Halbwort) |
| sb | Rs, Adr | MEM[Adr] MOD 256 := Rs (Speichere ein Byte) |

| 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 |