# Vorlesungsprüfung aus
# Rechnerstrukturen
## 28. Juni 2018

---

Die Arbeitszeit beträgt 90 Minuten. Als Hilfsmittel sind ausnahmslos Schreibzeug, Lineal und (nicht programmierbarer) Taschenrechner erlaubt. Schreiben Sie Ihre Antworten und Lösungen (inkl. Lösungsweg!) mit Füllfeder oder Kugelschreiber (nicht rot, KEIN Bleistift!) und streichen Sie alles durch, was nicht zur Beurteilung herangezogen werden soll. Ein Abbruch der Prüfung nach Erhalt der Angaben führt in jedem Fall zu einer Beurteilung.

Tragen Sie Namen, Kennzahl und Matrikelnummer **zu Beginn** der Prüfung in die Tabelle ein und beschriften Sie jedes Blatt, das Sie abgeben möchten rechts oben mit Namen und Matrikelnummer.

| Familienname: | Vorname: |
|---|---|
| Kennzahl: | Matrikelnummer: |

Viel Erfolg!

**Die nachfolgende Tabelle nicht beschriften!**

| Beispiel | | Mögliche Punkte | Erhaltene Punkte |
|---|---|---|---|
| Theoriefragen | | 48 | |
| Praxisbeispiele | 1 | 18 | |
| | 2 | 20 | |
| | 3 | 10 | |
| Gesamt | | 96 | |

# Theoriefragen (je 6 Punkte)

**Frage 1:**

Beschreiben Sie die Fertigungsschritte eines Chips.

**Frage 2:**

Wofür steht die Abkürzung MIPS (im Kontext von Performanzmetriken)? Wie wird die Metrik berechnet? Wie gut ist sie als Metrik zum allgemeinen Performanzvergleich geeignet? Begründen Sie Ihre Antwort.

**Frage 3:**

Skizzieren Sie den schematischen Aufbau eines MIPS-Prozessors mit Pipeline (ohne Control Unit, ohne Forwarding).

**Frage 4:**

Nennen Sie jeweils die grundlegenden Eigenschaften der zwei Arten der Sprungvorhersage (statische/dynamische Sprungvorhersage) bei bedingten Sprüngen. Auf welche Annahmen stützen diese sich jeweils? Welches Problem muss außer der Entscheidung ob gesprungen wird oder nicht (branch taken/not taken) noch gelöst werden und wie kann dies beschleunigt werden?

**Frage 5:**

Beschreiben Sie die Unterschiede zwischen „Static multiple issue“ und „Dynamic multiple issue“.

**Frage 6:**

Was ist die grundlegende Idee von „Virtual Memory“? Skizzieren Sie die Übersetzung einer virtuellen in eine physikalische Adresse. Welche Komponenten sind involviert und wie wird die Adresse aufgeteilt?

**Frage 7:**

Welche Typen von Cache-Misses kennen Sie? Warum treten diese jeweils auf?

**Frage 8:**

Was versteht man unter MTTF, MTBF, MTTR und Verfügbarkeit (Availability)? Beschreiben Sie jedes dieser Maße kurz. Wie hängen diese zusammen?

# Praxisbeispiele

## Beispiel 1

Gegeben sei folgender MIPS Assemblercode mit unvollständiger Übersetzung in Maschinencode:

(a) Vervollständigen Sie die Übersetzung des MIPS Assemblercodes in Maschinencode. (4)

```
400778:   _____   or     $v0,$zero,$zero
40077c:   10800004  beqz   $a0,0x400790
400780:   _____   addiu  $v1,$a0,-1
400784:   24420001  addiu  $v0,$v0,1
400788:   _____   beqz   $zero,0x40077c
40078c:   00832024  and    $a0,$a0,$v1
400790:   _____   jr     $ra
400794:   00000000  nop
```

(b) Wie ist ein „nop“ realisiert, d.h. welche Instruktion wird ausgeführt? (1)

(c) Beschreiben Sie so kurz und präzise wie möglich welche Funktionalität der Code aus Aufgabenteil (a) implementiert. $a0 enthält das Argument der Funktion, $v0 enthält den Rückgabewert. (Beachten Sie gegebenenfalls Load-/Branch-Delay-Slots) (4)

(d) Gegeben sei folgender Ausschnitt aus einem MIPS Assembler Programm, welches ohne wesentliche Optimierungen für einen Prozessor mit Standard-MIPS Pipeline ***ohne*** Forwarding implementiert wurde.

```
 (1)          addiu $a3,$a0,16
 (2)          move  $v0,$zero
 (3)  label:  lw    $v1,0($a0)
 (4)          lw    $a2,0($a1)
 (5)          nop
 (6)          nop
 (7)          nop
 (8)          subu  $v1,$v1,$a2
 (9)          nop
(10)          nop
(11)          nop
(12)          sra   $a2,$v1,0x1f
(13)          nop
(14)          nop
(15)          nop
(16)          xor   $v1,$a2,$v1
(17)          nop
(18)          nop
(19)          nop
(20)          subu  $v1,$v1,$a2
(21)          nop
(22)          nop
(23)          nop
(24)          addu  $v0,$v0,$v1
(25)          addiu $a1,$a1,4
(26)          addiu $a0,$a0,4
(27)          nop
(28)          nop
(29)          nop
(30)          bne   $a3,$a0,label
(31)          nop
(32)          nop
(33)          nop
(34)          jr    $ra
(35)          nop
(36)          nop
(37)          nop
```

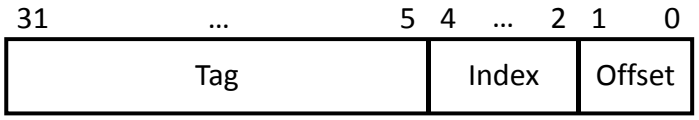
Optimieren Sie das Programm für einen verbesserten Prozessor, ***der über Forwarding verfügt***. Es soll im optimierten Code explizit berücksichtigt werden, dass der Load Befehl in der vierten Stufe ausgeführt wird (ein Zyklus Load Delay Slot) und der Sprungbefehl einen Branch Delay Slot verwendet. Es dürfen Instruktionen und Label umsortiert und entfernt, jedoch keine

hinzugefügt oder verändert werden. Als Lösung ist es ausreichend die Zeilennummern der Instruktionen zu verwenden, d.h. die Instruktionen selbst müssen nicht ausgeschrieben werden. „nops“ sollen, falls erforderlich, jedoch explizit angegeben werden. (7)

Kennzeichnen Sie Load Delay Slots und Branch Delay Slots im optimierten Programm. (2)

## Beispiel 2

Gegeben sei ein Prozessorsystem mit einem 2-Wege satzassoziativem Cache mit Byteadressierung. Die Unterteilung der Hauptspeicheradresse sieht folgendermaßen aus:




(a) Geben Sie die Blockgröße, die Anzahl der Blöcke, die Anzahl der Sets, die Kapazität (nur Daten) und die Gesamtgröße des Caches an. (5)

(b) Nennen Sie drei mögliche alternative Cacheorganisationen mit derselben Cachegröße (unter Beibehaltung der Blockgröße). Geben Sie zudem jeweils die Unterteilung der Hauptspeicheradresse an. (6)

(c) Wie viele alternative Cacheorganisationen gibt es insgesamt (unter Beibehaltung der Blockgröße) inklusive der unter (a) genannten? (1)

(d) Sie haben ein Prozessorsystem mit einem zwei-Wege satzassoziativem Cache mit LRU Ersetzungsstrategie. Ein Cacheblock umfasst zwei Worte und es gibt vier Sets. Tragen Sie für die angegebenen Byteadressen ein: (A) wie die Blockadresse aussieht; (B) in welches Set sie gespeichert werden; (C) wie das zugehörige Tag aussieht; (D) ob es sich um einen Cache Hit oder einen Miss handelt; (E) gegebenenfalls welcher Eintrag (Tag) verdrängt wird. (8)

| Byteadresse | Blockadresse | Set | Tag | Hit/Miss | Verdrängt |
|---|---|---|---|---|---|
| $64_{10}$ | | | | | |
| $128_{10}$ | | | | | |
| $0_{10}$ | | | | | |
| $16_{10}$ | | | | | |
| $512_{10}$ | | | | | |
| $4_{10}$ | | | | | |
| $68_{10}$ | | | | | |
| $42_{10}$ | | | | | |

## Beispiel 3

Ein Computerprogramm verwendet 5% seiner Befehle für Floatingpoint Multiplikationen, 15% für Floatingpoint Divisionen und 30% für Floatingpoint Additionen. Um die Ausführung des Programms zu beschleunigen werden von der Designabteilung folgende Vorschläge gemacht:

a) Addition um den Faktor 4 beschleunigen (2)
b) Multiplikation um den Faktor 8 beschleunigen (2)
c) Addition und Division jeweils um den Faktor 1.5 beschleunigen (2)
d) Multiplikation und Division jeweils um den Faktor 2 beschleunigen (2)

Berechnen Sie alle vier Varianten.

e) Welche Variante ist die beste? (2)