Klausur zur Vorlesung

Rechnerarchitektur

Prof. Christian Plessl
Fachgebiet Hochleistungs-IT-Systeme
Universität Paderborn

19.02.2019

- Die Bearbeitungsdauer beträgt für alle Studenten **90 Minuten.** Es sind **alle 5 Aufgaben** zu bearbeiten.
- Es sind keine Hilfsmittel zugelassen.
- Schreiben Sie nicht mit Bleistift oder Rotstift.
- Verwenden Sie kein eigenes Papier. Bei Bedarf bekommen Sie Papier bei der Klausuraufsicht.
- Schreiben Sie auf jedes Blatt (auch auf das Konzeptpapier) in Blockschrift Ihren Namen und Ihre Matrikelnummer.
- Bei mehreren präsentierten Lösungen wird die Aufgabe nicht gewertet! Streichen Sie daher bei Angabe mehrerer Lösungsansätze die nicht zu bewertenden Lösungen durch! Verwenden Sie kein Tipp-Ex.
- Abschreiben und abschreiben lassen oder Hilfe Dritter führt zum Nichtbestehen der Klausur.

Nachname: _____

Vorname: _____

Matrikelnummer: _____

Studiengang: _____

Aufkleber

Aufgabe	**1**	**2**	**3**	**4**	**5**	$\sum$
Punkte	**14**	**25**	**18**	**15**	**18**	90
Erreicht						

Aufgabe 1 (Multiple Choice) **[14 Punkte]**

Bei den folgenden Fragen können eine oder mehrere Antworten richtig sein. Kreuzen Sie die richtigen Antworten deutlich an. Für jede vollständig korrekt beantwortete Frage gibt es 2 Punkte, ansonsten 0 Punkte.

(a) Der MIPS-Instruktionssatz hat zusätzlich zur `add` eine `addu` Instruktion: (2 Punkte)

- ☐ Weil die ALU aufgrund der Zweierkomplementdarstellung unterschiedliche arithmetische Schaltungen für vorzeichenbehaftete (signed) und nicht vorzeichenbehaftete (unsigned) Zahlen verwenden muss
- ☐ Um explizit zu bestimmen, ob bei einem arithmetischen Überlauf eine Exception ausgelöst werden soll (`add`) oder nicht (`addu`)
- ☐ Um zwischen einer Addition mit zwei Operanden (binär) und einem Operanden (unär) zu unterscheiden
- ☐ Um zwischen Ganzzahl und Floating-Point Additionen zu unterscheiden

(b) Ein Programm hat einen Anteil von 25% Floating-Point Instruktionen. Welche Beschleunigung (Speedup) kann laut Amdahls-Gesetz maximal erzielt werden, wenn die Floating-Point Instruktionen durch Optimierung der Implementierung rascher ausgeführt werden: (2 Punkte)

- ☐ Speedup = 4/3
- ☐ Speedup = 5/4
- ☐ Speedup = 4
- ☐ keines der obengenannten Resultate ist korrekt

(c) Methoden zur Vermeidung von Pipeline-Hazards sind: (2 Punkte)

- ☐ Halbtakt-Zugriff auf Registerfile
- ☐ Verwendung einer Von-Neumann-Architektur statt einer Harvard-Architektur
- ☐ Forwarding von Zwischenresultaten
- ☐ Mehrfachzuordnung von Instruktionen (Multiple Issue)

(d) Für einen Cache mit Write-Back-Architektur gilt: (2 Punkte)

- ☐ Die Inhalte von Cache und Hauptspeicher sind stets konsistent
- ☐ Der Busverkehr auf dem Cache-Speicher-Bus ist niedriger als bei einer Write-Through-Architektur
- ☐ Die Miss-Rate ist niedriger als bei Write-Through-Architektur
- ☐ Die Cache-Metadaten müssen um ein Dirty-Bit erweitert werden

(e) Für Page Tables gilt: (2 Punkte)

- ☐ Es gibt immer zwei Page Tables: eine Page Table für das Betriebssystem und eine Page Table für alle Benutzerprozesse
- ☐ Die Größe der Page Table hängt von der Größe des physikalisch installierten Speichers ab
- ☐ Je größer die Pages sind, desto kleiner ist der Speicherbedarf der Page Table
- ☐ Im Translation Lookaside Buffer (TLB) werden kürzlich benutzte Adressumsetzungen aus der Page Table gecached

(f) Für I/O Datentransfers mit DMA gilt: (2 Punkte)

- ☐ Es kann auf Interrupts verzichtet werden
- ☐ Der DMA Controller muss mit physikalischen Adressen arbeiten
- ☐ Während des DMA Transfers dürfen keine Veränderungen der betroffenen Page Table Einträge vorgenommen werden
- ☐ Bis zum Ende des DMA Transfers kann die CPU keine Speicherzugriffe durchführen, da die DMA Einheit den Bus komplett belegt

(g) Für die asynchrone Übertragung auf I/O Bussen gilt: (2 Punkte)

- ☐ Es ist nicht möglich die Adress- und Datenleitungen zu multiplexen
- ☐ Ein gemeinsames Taktsignals ist überflüssig
- ☐ Um keine Signaländerungen zu verpassen, muss der Prozessor mindestens mit der doppelten Taktrate wie das I/O Gerät betrieben werden
- ☐ Die Kommunikation von Prozessor und I/O Gerät funktioniert unabhängig von der Signallaufzeit korrekt

Aufgabe 2 (Leistungsbewertung) **[25 Punkte]**

Ein Hersteller von anwendungsspezifischen Schaltungen hat einen Prozessor im Angebot, der speziell für die schnelle Ausführung eines Programms entworfen wurde, was folgende Instruktionsverteilung aufweist:

Instruktionsklasse	Anteil am Programm
Fließkommaarithmetik	15%
Ganzzahlarithmetik	44%
Speicherzugriffe	20%
Sonstige	21%

Die erste Version des Prozessors kann das Programm in 100 Sekunden ausführen. Bei der Erstellung einer zweiten Version des Prozessors wird ein anderes Fertigungsverfahren eingesetzt, so dass bei der Planung noch eine große Restfläche auf dem Chip zur Verfügung steht. Dem Hersteller stehen nun verschiedene Verbesserungen zur Verfügung, die diese Restfläche unterschiedlich ausnutzen (die Größe gibt also an, wie viel Prozent der Restfläche belegt würden). Dabei können zwei Verbesserungen unterschiedlicher Kategorien auch gleichzeitig zum Einsatz kommen, falls die Fläche ausreicht. Die möglichen Verbesserungen sind:

Name	Verbesserung von	Speedup	Größe
FV1	Fließkommaarithmetik	1,25	25%
FV2	Fließkommaarithmetik	5	65%
FV3	Fließkommaarithmetik	6	100%
IV1	Ganzzahlarithmetik	1,1	30%
IV2	Ganzzahlarithmetik	1,375	70%
IV3	Ganzzahlarithmetik	1,408	95%

Beispiele: Die Verbesserung FV3 nutzt die Restfläche alleine komplett aus. Die Verbesserungen FV1 und IV1 nutzen zusammen 55% der Restfläche aus. Die Verbesserungen IV1 und IV2 können nicht zusammen verwendet werden, da sie beide die Ganzzahlarithmetik verändern.

(a) Um die Verbesserungen miteinander vergleichen zu können, muss die Auswirkung einer Verbesserung auf das Gesamtprogramm untersucht werden. Wie heißt das Prinzip, nach dem man die neue Ausführungszeit, und somit den Gesamtspeedup berechnen kann? (1 Punkt)

(b) Die Verbesserungen FV3 und IV3 sind beide so groß, dass sie jeweils nur exklusiv eingesetzt werden können. Wie hoch sind die verbesserten Laufzeiten $T^{IV3}_{EXE\_neu}$ und $T^{FV3}_{EXE\_neu}$? (7 Punkte)

(c) Bei den Verbesserungen FV1, FV2, IV1 und IV2 würde ein exklusiver Einsatz Fläche verschwenden, weswegen sie kombiniert werden sollten. Berechnen Sie für die gültigen Kombinationen jeweils die verbesserte Laufzeit. Markieren Sie Ergebnisse deutlich als solche! (14 Punkte)

(d) Welche Verbesserung(en) sollte der Hersteller für die zweite Version des Prozessors also wählen? (1 Punkt)

(e) Wie hoch ist damit der erzielte Speedup insgesamt? Zwei Nachkommastellen reichen aus. (2 Punkte)

Aufgabe 3 (Branch Delay Slots) **[18 Punkte]**

Gegeben sei ein MIPS-Prozessor mit einem Branch Delay Slot. Verändern Sie den vorgegebenen Assemblercode so, dass die Ausführung durch Nutzung des Delay Slots in einem oder mehreren Programmpfaden optimiert wird. Stellen Sie sicher, dass die Register am Ende dieselben Werte wie in der vorliegenden Variante aufweisen.

Ersatz
ungültige Lösung streichen!

(a) (4 Punkte)

```
1:  add $4, $2, $6      1: _________________   1: _________________
2:  add $3, $1, $2      2: _________________   2: _________________
3:  add $6, $6, $1      3: _________________   3: _________________
4:  add $5, $4, $2      4: _________________   4: _________________
5:  beq $5, $4, L1      5: _________________   5: _________________
6:  nop                 6: _________________   6: _________________
...                        _________________      _________________
  L1: ...                  _________________      _________________
```

(b) (4 Punkte)

```
1:    add $1, $2, $3    1: _________________   1: _________________
2:    bne $1, $4, L1    2: _________________   2: _________________
3:    nop               3: _________________   3: _________________
4:    add $4, $4, $2    4: _________________   4: _________________
5:    add $5, $4, $3    5: _________________   5: _________________
6: L1: addi $4, $1, 1   6: _________________   6: _________________
                           _________________      _________________
```

(c) (4 Punkte)

```
1:   addi $1, $zero, 0  1: _________________   1: _________________
2:   addi $2, $zero, 10 2: _________________   2: _________________
3: L1: addi $1, $1, 1   3: _________________   3: _________________
4:   bne $1, $2, L1     4: _________________   4: _________________
5:   nop                5: _________________   5: _________________
                           _________________      _________________
```

(d) Geben Sie die Ausführungsreihenfolge der Instruktionen sowie die Werte der Register $1 und $2 nach Ausführung für den Fall eines vorhandenen und eines nicht vorhandenen Branch Delay Slots an. (6 Punkte)

```
1:      addi $1, $zero, 21
2:      addi $2, $zero, 10
3:      bne  $1, $2, L1
4:      addi $1, $zero, 42
5:      subi $1, $1, 1
6: L1: add  $2, $1, $2
```

Branch Delay Slot	Instruktionsabfolge	$t1	$t2
vorhanden			
nicht vorhanden			

Ersatz - ungültige Lösung streichen!

Branch Delay Slot	Instruktionsabfolge	$t1	$t2
vorhanden			
nicht vorhanden			

Aufgabe 4 (Pipelining) **[15 Punkte]**

Basierend auf einem MIPS-Prozessor mit Pipelining seien folgende Eigenschaften gegeben:

- Die Pipeline bestehe aus 5 Stufen: **IF, ID, EX, MEM, WB**.
- Die Registerzugriffe erfolgen im Halbtaktverfahren.
- Harvard-Architektur sei realisiert.

Betrachten Sie folgende MIPS-Assembler Programmsequenz:

```
01: lw $1, 0($0)
02: sub $3, $1, $6
03: lw $2, 0($6)
04: add $5, $3, $4
05: sw $5, 0($2)
06: sw $4, 0($1)
```

(a) Geben Sie die Anzahl der für die Ausführung obiger Programmsequenz benötigten Taktzyklen an, für den Fall, dass keine Forwarding-Unit verwendet wird. Veranschaulichen Sie dazu die Pipelinebelegung mit Hilfe des nachfolgenden Diagramms. (6 Punkte)

Anzahl Taktzyklen: _____

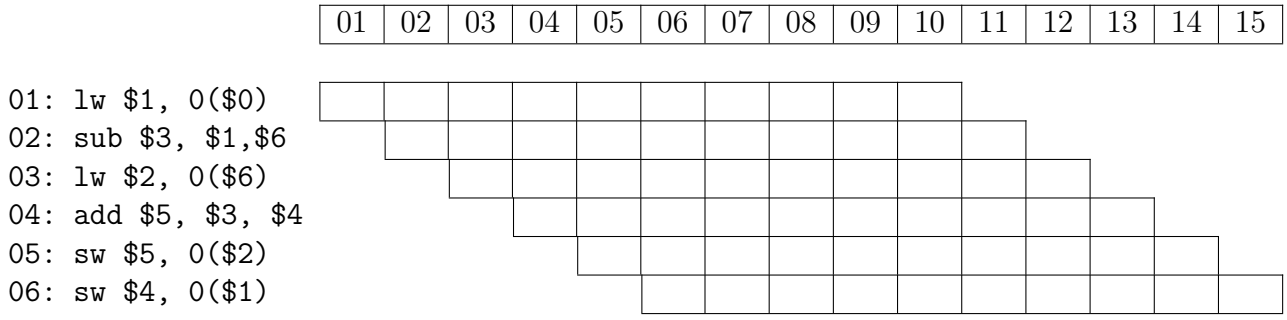
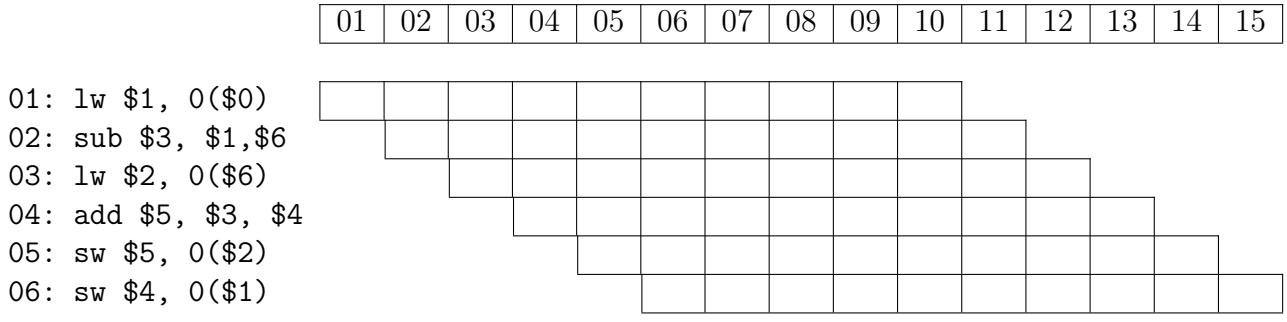


Diagramm zu (a)



Ersatzdiagramm zu (a)
(ungültige Lösung streichen!)

(b) Geben Sie wiederum die Anzahl der für die Ausführung obiger Programmsequenz benötigten Taktzyklen an. Betrachten Sie hierbei den Fall, dass eine Forwarding Unit verwendet wird. Veranschaulichen Sie dazu wiederum die Pipelinebelegung mit Hilfe des nachfolgenden Diagramms.

(6 Punkte)

Anzahl Taktzyklen: _____

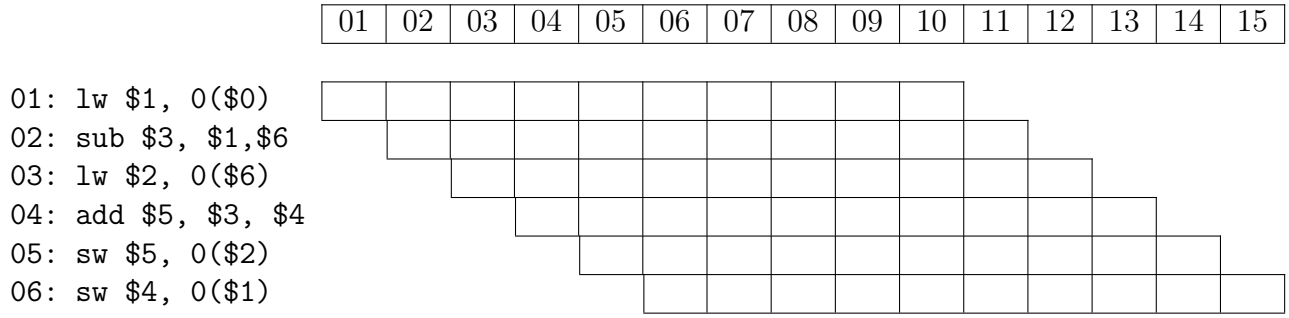
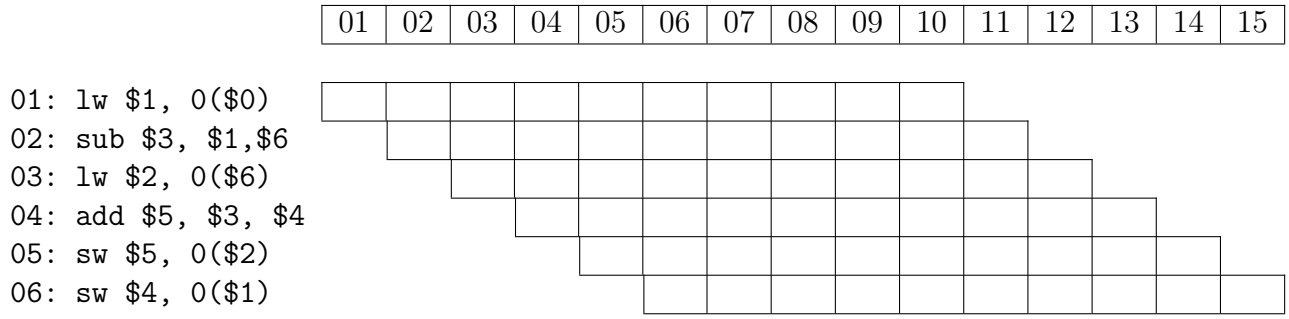


Diagramm zu (b)



Ersatzdiagramm zu (b)
(ungültige Lösung streichen!)

(c) Kann durch Umsortieren der Programmsequenz die Ausführung weiter beschleunigt werden? Begründen Sie Ihre Antwort. (3 Punkte)

Aufgabe 5 (Caches) **[18 Punkte]**

Die aktuelle Belegung eines satzassoziativen Caches ist in Tabelle 1 dargestellt. Als Verdrängungsstrategie wird LRU verwendet. Speicherzugriffe erfolgen auf Wortadressen. M[i] steht für den Inhalt des Hauptspeichers an der Wortadresse i. Ein Wort sind 4 Bytes.

Index	V	Tag	Data_0	Data_1	Data_2	Data_3	V	Tag	Data_0	Data_1	Data_2	Data_3
0	0	0000	M[0]	M[1]	M[2]	M[3]	1	1110	M[224]	M[225]	M[226]	M[227]
1	1	0100	M[72]	M[73]	M[74]	M[75]	0	1100	M[200]	M[201]	M[202]	M[203]
2	0	1001	M[144]	M[145]	M[146]	M[147]	1	1111	M[240]	M[241]	M[242]	M[243]
3	1	1011	M[184]	M[185]	M[186]	M[187]	0	0011	M[56]	M[57]	M[58]	M[59]

Tabelle 1: Aktuelle Cache-Belegung.

(a) Bestimmen Sie die Größen von Tag, Index, Blockoffset und ggf. Byteoffset. Zeichnen Sie die Grenzen der Bitfelder in die Abbildung 1 durch senkrechte Striche ein und beschriften Sie die entstandenen Felder. (2 Punkte)

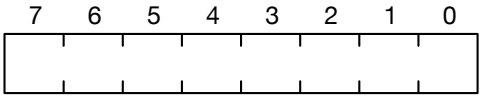


Abbildung 1: Adressaufteilung.

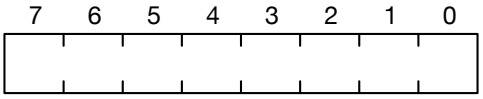


Abbildung 2: Adressaufteilung (Ersatz).

(b) Ein MIPS-Programm führt Speicherzugriffe auf den Cache aus Tabelle 1 aus. Kreuzen Sie jeweils an, ob es sich um Hits oder Misses handelt und geben Sie die resultierende Cache-Belegung in Tabelle 2 an. Nicht veränderte Werte müssen nicht übertragen werden. (10 Punkte)

Hinweis: Die Instruktion `lw rt, offset(base)` holt ein Wort aus `M[base+offset]` in das Register `rt`.

`lw $t0,` 205_{10}`($0)`	`#` 205_{10} `=` 11001101_2	☐ Hit	☐ Miss
`lw $t0,` 224_{10}`($0)`	`#` 224_{10} `=` 11100000_2	☐ Hit	☐ Miss
`lw $t0,` 3_{10}`($0)`	`#` 3_{10} `=` 00000011_2	☐ Hit	☐ Miss
`lw $t0,` 171_{10}`($0)`	`#` 171_{10} `=` 10101011_2	☐ Hit	☐ Miss
`lw $t0,` 210_{10}`($0)`	`#` 210_{10} `=` 11010010_2	☐ Hit	☐ Miss

Index	V	Tag	Data_0	Data_1	Data_2	Data_3	V	Tag	Data_0	Data_1	Data_2	Data_3
0												
1												
2												
3												

Tabelle 2: Resultierende Cache-Belegung.

Ersatz:

```
lw $t0, 205_10($0)      # 205_10 = 11001101_2     ☐ Hit  ☐ Miss
lw $t0, 224_10($0)      # 224_10 = 11100000_2     ☐ Hit  ☐ Miss
lw $t0,   3_10($0)      #   3_10 = 00000011_2     ☐ Hit  ☐ Miss
lw $t0, 171_10($0)      # 171_10 = 10101011_2     ☐ Hit  ☐ Miss
lw $t0, 210_10($0)      # 210_10 = 11010010_2     ☐ Hit  ☐ Miss
```

Index	V	Tag	$Data_0$	$Data_1$	$Data_2$	$Data_3$	V	Tag	$Data_0$	$Data_1$	$Data_2$	$Data_3$
0												
1												
2												
3												

Tabelle 3: Resultierende Cache-Belegung (Ersatz).

(c) Angenommen, Speicherzugriffe auf Byteadressen seien erlaubt. Entsprechend wird die Adressierung um einen Byteoffset erweitert, der Rest bleibt unverändert. Übertragen Sie zunächst die Größen von Tag, Index und Blockoffset. (6 Punkte)

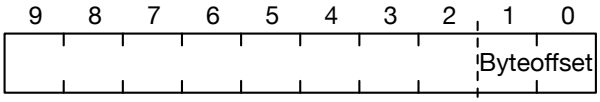


Abbildung 3: Adressaufteilung.

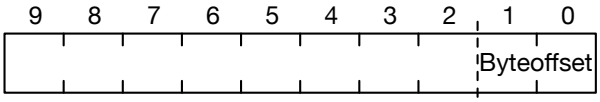


Abbildung 4: Adressaufteilung (Ersatz).

Ein neues MIPS-Programm führt folgenden Speicherzugriff aus. Die Instruktion `lb rt, offset(base)` holt ein Byte aus `M[base+offset]` in das Register `rt`. Bei der Adresse handelt es sich um eine *Byteadresse*.

```
lb $t0, 9_10($0)        # 9_10 = 0000001001_2
```

Kreuzen Sie in Abbildung 5 das adressierte Byte an und markieren Sie eindeutig alle weiteren Bytes, die durch den Speicherzugriff in den Cache geladen werden.

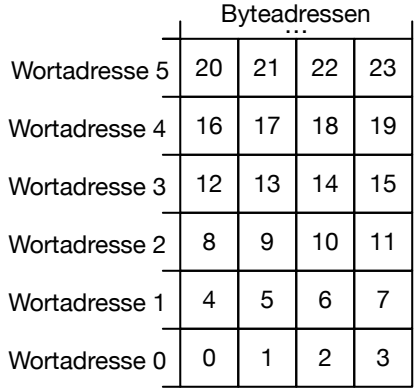


Abbildung 5: Adressaufteilung.

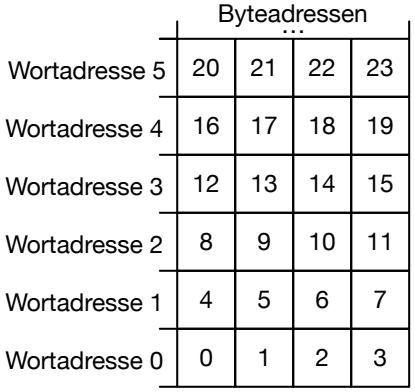


Abbildung 6: Adressaufteilung (Ersatz).

Konzeptpapier: Falls der Platz unter den einzelnen Aufgaben nicht ausreicht, können Sie diese Seiten für Zwischenrechnungen nutzen. Bitte Lösung und Lösungsweg eindeutig mit der Aufgabennummer markieren!