Klausur zur Vorlesung

Grundlagen der Rechnerarchitektur (GRA)

Prof. Marco Platzner
Fachgebiet Technische Informatik
Universität Paderborn

24.02.2017

- 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	**15**	**10**	**20**	**25**	**20**	90
Erreicht						

Aufgabe 1 (Multiple Choice) **[15 Punkte]**

Bei den folgenden Fragen können keine, eine oder mehrere Antworten richtig sein. Kreuzen Sie die richtigen Antworten deutlich an.

(a) Welche Aussagen zu MIPS Adressierungsarten sind korrekt?

- ☐ Die `j` Instruktion verwendet PC-relative Adressierung
- ☐ Bei pseudodirekter Adressierung wird eine 26-Bit Adresse mit den oberen 4 Bits des PC Registers konkateniert
- ☐ Bei PC-relativer Adressierung steht ein Operand im Hauptspeicher
- ☐ Die `beq` Instruktion verwendet Basisadressierung

(b) Zu den Aufgaben des Linkers zählen...

- ☐ Bestimmung der Speicherbereiche, die durch die einzelnen Programmteile belegt werden
- ☐ Auflösen der Querverweise zwischen den Programmteilen auf Basis der relocation information
- ☐ Kopieren der Befehle und Daten aus der Programmdatei in den Hauptspeicher
- ☐ Festlegen eines ausreichend großen Adressbereichs im Hauptspeicher für Programmtext und Daten

(c) Welche Aussagen zu Caches sind korrekt?

- ☐ Die Größe des Caches ist abhängig vom Adressraum des Prozessors
- ☐ Der Datencache wird ausschließlich für die Speicherhierarchie verwendet
- ☐ Um den Prozessor beim Schreiben nicht durch den Hauptspeicher zu bremsen, wird bei write-back ein Pufferspeicher (write buffer) verwendet
- ☐ Einem page fault geht immer ein TLB miss voraus

(d) Welche Aussagen zu Pipelining und Mehrfachzuordnung sind korrekt?

- ☐ Durch Umordnen von Instruktionen können Data Hazards reduziert werden
- ☐ Dynamisches Scheduling vereinfacht den Compilerentwurf im Vergleich zu statischem Scheduling
- ☐ Code für VLIW Prozessoren ist binärkompatibel zu anderen Prozessoren derselben ISA
- ☐ Bei VLIW Prozessoren wird die Zuordnung zu den Ausführungseinheiten vom Compiler durchgeführt

(e) Welche Aussagen zu Arbitrierung sind korrekt?

- ☐ Bei zentraler Arbitrierung kann Fairness garantiert werden
- ☐ Wenn es mehrere Slaves gibt, muss der Bus arbitriert werden
- ☐ Arbitrierung durch Selbstselektion ist ein verteiltes Arbitrierungsverfahren
- ☐ Daisy-Chain Arbitrierung ist ein verteiltes Arbitrierungsverfahren

Aufgabe 2 (Amdahl's Law) **[10 Punkte]**

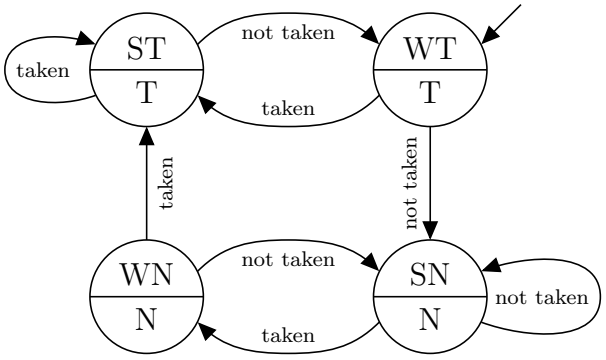
Ein Unternehmen hat eine neuere Version eines Prozessors auf den Markt gebracht. Ein Kunde möchte auf einem Server die Vorgängerversion des Prozessors durch die neuere Version austauschen. Von dem neuen Prozessor ist bekannt, dass Load-/Store-Instruktionen drei mal schneller geworden sind. Zusätzlich soll es eine Verbesserung der Integerberechnungen geben. Auf dem Server wird hauptsächlich ein Programm ausgeführt, dessen Instruktionen zu 60% aus Load/Store-Instruktionen, zu 30% aus Integerberechnungen und zu 10% aus sonstigen Instruktionen bestehen. Die neue Gesamtausführungszeit des Programmes beträgt 36 Sekunden, was einen Speedup von 2 bedeutet.

(a) Wie groß ist die Verbesserung durch die Integerberechnungen?

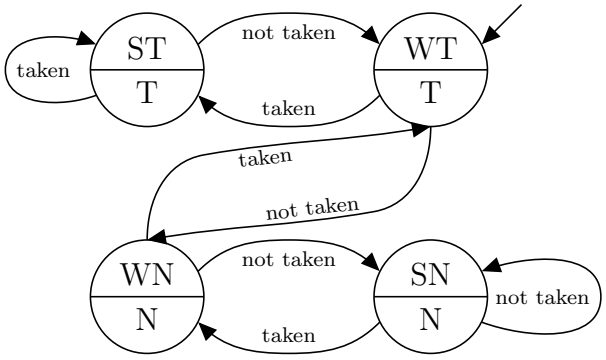
(b) Wie groß wäre der Speedup gewesen, wenn das Unternehmen auf die Verbesserung der Integerberechnung verzichtet hätte und wie lange würde das Programm in dem Fall ausgeführt werden?

Aufgabe 3 (Pipelining-Sprungvorhersage) **[20 Punkte]**

Gegeben sind folgende zwei Automatengraphen für 2-bit Sprungprädiktoren:



Prädiktor P1



Prädiktor P2

Hierbei soll bedeuten:
ST $\widehat{=}$ taken „strong“, **WT** $\widehat{=}$ taken „weak“,
WN $\widehat{=}$ not taken „weak“, **SN** $\widehat{=}$ not taken „strong“.
In den „taken“ Zuständen sagen die Prädiktoren voraus, dass dem bedingten Sprung gefolgt wird (T), in den „not taken“ Zuständen hingegen, dass das Programm ohne Sprung fortgesetzt wird (N).

(a) Füllen Sie folgende Tabellen aus. Ersatztabellen finden Sie auf der nächsten Seite.

Branch		1	2	3	4	5	6	7	8	9	10	Treffer
B1	Tatsächlich	T	N	T	T	N	T	T	N	N	N	—
	Zustand P1	WT										
	Zustand P2	WT										

B2	Tatsächlich	N	T	T	N	N	T	T	N	N	T	—
	Zustand P1	WT										
	Zustand P2	WT										

(b) Konstruieren Sie nun selbst eine Sequenz von tatsächlichen Sprungentscheidungen für einen bedingten Sprung B3, bei der P1 mindestens 2 Treffer mehr als P2 erzielt. Eine Ersatztabelle finden Sie auf der nächsten Seite.

Branch		1	2	3	4	5	6	7	8	9	10	Treffer
B3	Tatsächlich											—
	Zustand P1	WT										
	Zustand P2	WT										

(c) Generalisieren Sie ihre Lösung zu Aufgabenteil **(b)**. Unter welchen Umständen ist P1 besser als P2, beziehungsweise wie muss eine Sprungentscheidungssequenz beschaffen sein, damit P2 eine schlechte Trefferquote hat, P1 aber nicht?

Ersatztabellen für Teil **(a)**. **Ungültige Lösungen streichen!**

Branch		1	2	3	4	5	6	7	8	9	10	Treffer
B1	Tatsächlich	T	N	T	T	N	T	T	N	N	N	—
	Zustand P1	WT										
	Zustand P2	WT										

B2	Tatsächlich	N	T	T	N	N	T	T	N	N	T	—
	Zustand P1	WT										
	Zustand P2	WT										

Ersatztabelle für Teil **(b)**. **Ungültige Lösungen streichen!**

Branch		1	2	3	4	5	6	7	8	9	10	Treffer
B3	Tatsächlich											—
	Zustand P1	WT										
	Zustand P2	WT										

Aufgabe 4 (Caches) **[25 Punkte]**

(a) Gegeben sei eine byte-adressierbare Architektur mit einem 32-bit Adressraum. Die jeweils 16 kB großen L1 Instruktions- und Datencaches sind 2-fach satzassoziativ und haben eine Blockgröße von vier Worten. Jedes Wort besteht aus 4 Bytes. Bitte ergänzen Sie:

Der Blockindex umfasst die Bits _____ bis _____

Der Index umfasst die Bits _____ bis _____

Der Tag umfasst die Bits _____ bis _____

(b) Wie viele Speicherbits braucht man für die Implementierung beider Caches aus **(a)**, wenn neben dem valid-Bit auch zwei Bits für die Ersetzungsstrategie abspeichert werden? Sie brauchen den Ausdruck nicht auszurechnen.

(c) Die Hit-Time des Instruktionscaches aus **(a)** beträgt einen Taktzyklus bei einer Miss-Rate von 2%. Der Datencache wird von 30% aller Instruktionen angesprochen und besitzt eine Hit-Time von einem Taktzyklus bei einer Hit-Rate von 92%. Die Miss-Penalty beträgt für beide Caches 500 Taktzyklen. Wie hoch ist die mittlere Zugriffszeit pro Instruktion bei einer 3 GHz schnellen CPU?

(d) In dieser Aufgabe sollen für den Instruktionscache aus der Aufgabe **(a)** die **Least** und **Most** Recently Used Ersetzungsstrategien verglichen werden. Der Cache ist nun nicht mehr 16 kB sondern 128 Byte groß. Weiterhin ist dazu die folgende Adresssequenz für Lesezugriffe gegeben:

$$120, 90, 40, 170, 130, 144, 180, 60$$

Bitte ergänzen Sie die Tabellen zu Cachebelegungen sowie Hits und Misses für beide Ersetzungsstrategien. Welche Ersetzungsstrategie hat eine bessere Hit-Rate?

Bitte beachten Sie, dass das Feld Time den Zeitpunkt des Zugriffs angibt. Sollte ein Block überschrieben werden, schreiben Sie bitte die Speicheradressen der neuen Werte mit einem Komma getrennt neben die alten Werte. "m[x:y]" steht dabei für den Inhalt des Hauptspeichers von Byteadresse x bis y.

Least Recently Used:

Time	5	6	7	8	9	10	11	12
Adresse	120	90	40	170	130	144	180	60
Hit/Miss								

Index	V	Data	Time	V	Data	Time
00-15	0	m[128:143]	1	1	m[00:15]	1
16-31	0	m[80:95]	6	1	m[80:95]	0
32-47	1	m[160:175]	2	1	m[96:111]	3
48-63	0	m[112:127]	5	1	m[48:63]	4

Most Recently Used:

Time	5	6	7	8	9	10	11	12
Adresse	120	90	40	170	130	144	180	60
Hit/Miss								

Index	V	Data	Time	V	Data	Time
00-15	0	m[128:143]	1	1	m[00:15]	1
16-31	0	m[80:95]	6	1	m[80:95]	0
32-47	1	m[160:175]	2	1	m[96:111]	3
48-63	0	m[112:127]	5	1	m[48:63]	4

Ersatztabellen: _____

Time	5	6	7	8	9	10	11	12
Adresse	120	90	40	170	130	144	180	60
Hit/Miss								

Index	V	Data	Time	V	Data	Time
00-15	0	m[128:143]	1	1	m[00:15]	1
16-31	0	m[80:95]	6	1	m[80:95]	0
32-47	1	m[160:175]	2	1	m[96:111]	3
48-63	0	m[112:127]	5	1	m[48:63]	4

Ersatztabellen: _____

Time	5	6	7	8	9	10	11	12
Adresse	120	90	40	170	130	144	180	60
Hit/Miss								

Index	V	Data	Time	V	Data	Time
00-15	0	m[128:143]	1	1	m[00:15]	1
16-31	0	m[80:95]	6	1	m[80:95]	0
32-47	1	m[160:175]	2	1	m[96:111]	3
48-63	0	m[112:127]	5	1	m[48:63]	4

Aufgabe 5 (Statische Mehrfachzuordnung) [**20 Punkte**]

Abbildung 1 zeigt eine `for`-Schleife und Abbildung 2 den für den Schleifenrumpf erzeugten MIPS-Code. Register `$s1` wird vor der Ausführung des Schleifenrumpfs mit der Adresse von `x[0]`, Register `$s2` mit der Adresse von `y[0]` und der Schleifenzähler `$s3` mit `512` initialisiert.

```
for (i=0; i<512; i++)
    x[i] = x[i]*8 + y[i];
```

Abbildung 1: `for`-Schleife

```
L1:   lw   $t0, 0($s1)
      sll  $t0, $t0, 3
      lw   $t1, 0($s2)
      add  $t1, $t1, $t0
      sw   $t1, 0($s1)
      addi $s1, $s1, 4
      addi $s2, $s2, 4
      addi $s3, $s3, -1
      bne  $s3, $zero, L1
```

Abbildung 2: MIPS-Code

Die Schleife wird auf eine MIPS-Architektur mit statischer Mehrfachzuordnung abgebildet. Die Architektur besitzt vier Ausführungseinheiten, zwei `ALU/BR`-Einheiten für arithmetische-logische Instruktionen sowie bedingte und unbedingte Sprünge und zwei `LD/ST`-Einheiten für Datentransfer-Instruktionen.

(a) Geben Sie den idealen CPI-Wert an, der mit dieser MIPS-Architektur erreichbar ist.

$\text{CPI}_{\text{ideal}} =$ _____

(b) Bilden Sie den Code in Abbildung 2 durch statisches Pipelinescheduling auf die Architektur ab. Vervollständigen Sie dazu Tabelle 1. In dieser Tabelle wird jeder Ausführungseinheit eine Spalte zugeordnet und jede Zeile entspricht einem Taktzyklus. Eine Instruktion in einer Zelle der Tabelle bedeutet, dass die Instruktion in dem entsprechenden Taktzyklus in der Ausführungseinheit bearbeitet wird. Die Architektur benutzt *forwarding*, dh. Instruktionen mit Datenabhängigkeiten können direkt aufeinander folgen, außer bei einem *load-use* Konflikt, der einen *stall cycle* erfordert. Nehmen Sie an, dass die Sprungvorhersage perfekt funktioniert und dass es keine *branch-delay slots* gibt.

	ALU/BR #1	ALU/BR #2	LD/ST #1	LD/ST #2
L1:		`addi $s2, $s2, 4`	`lw $t0, 0($s1)`	
	`bne $s3, $zero, L1`			

Tabelle 1: Code für statische 4-fach Zuordnung

	ALU/BR #1	ALU/BR #2	LD/ST #1	LD/ST #2
L1:		`addi $s2, $s2, 4`	`lw $t0, 0($s1)`	
	`bne $s3, $zero, L1`			

Tabelle 2: Code für statische 4-fach Zuordnung (**Ersatztabelle**)

Geben Sie den CPI-Wert an, den der Code in Tabelle 1 erreicht.

$\text{CPI}_{\text{Code-Tabelle-1}} =$ _____

(c) Entrollen Sie die Schleife 2-fach, so dass der neue Schleifenrumpf zwei Iterationen der ursprünglichen Schleife berechnet. Führen Sie dazu das *loop unrolling* und das *register renaming* durch und geben Sie den resultierenden MIPS-Code in Abbildung 3 an.

```
L1:    lw    $t0, 0($s1)
```

```
       bne   $s3, $zero, L1
```

Abbildung 3: MIPS-Code, Schleife 2-fach entrollt

(d) Bilden Sie den Code aus Aufgabenteil **(c)** durch statisches Pipelinescheduling auf die Architektur ab, indem Sie Tabelle 3 vervollständigen.

	ALU/BR #1	ALU/BR #2	LD/ST #1	LD/ST #2
`L1:`				

Tabelle 3: Code für statische 4-fach Zuordnung, Schleife 2-fach entrollt

	ALU/BR #1	ALU/BR #2	LD/ST #1	LD/ST #2
`L1:`				

Tabelle 4: Code für statische 4-fach Zuordnung, Schleife 2-fach entrollt (**Ersatztabelle**)

Geben Sie den CPI-Wert an, den der Code in Tabelle 3 erreicht.

$\text{CPI}_{\text{Code-Tabelle-3}} =$ _____

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!