Klausur zur Vorlesung

Grundlagen der Rechnerarchitektur / Technische Informatik (GRA/TI)

Prof. Marco Platzner
Fachgebiet Technische Informatik
Universität Paderborn

27.02.2015

- 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**	**20**	**15**	**20**	**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) Was berechnet folgendes Assemblerprogramm:

```
# x in $t0, y in $t1
xor $t1, $t0, $t1
xor $t0, $t1, $t0
xor $t1, $t1, $t0
```

- ☐ `x in $t0, y in $t1`
- ☐ `x in $t0, x in $t1`
- ☐ `0 in $t0, 0 in $t1`
- ☐ `y in $t0, x in $t1`

(b) Was sind Charakteristika einer RISC Instruktionssatzarchitektur?

- ☐ feste Instruktionslänge
- ☐ viele, komplexe Instruktionsarten
- ☐ load/store Architektur
- ☐ Register/Speicher Operationen

(c) Welche Aussagen sind für einen Write-Back Cache korrekt?

- ☐ Cache und Hauptspeicher sind immer konsistent.
- ☐ Jeder Cacheblock benötigt ein Dirty-Bit.
- ☐ Write-Back funktioniert bei Caches mit beliebigem Assoziativitätsgrad.

(d) Wie gross ist die durchschnittliche Rotationslatenz bei einer Festplatte mit 15000 Umdrehungen pro Minute?

- ☐ 0.5 ms
- ☐ 1 ms
- ☐ 2 ms
- ☐ 3 ms

(e) Welche Gemeinsamkeiten haben Superskalar- und VLIW-Prozessoren?

- ☐ Beide sind Prozessoren mit Mehrfachzuordnung.
- ☐ Bei beiden führt der Compiler die Zurodnung von Instruktionen zu Ausführungseinheiten durch.
- ☐ Beide haben einen idealen CPI-Wert von kleiner eins.
- ☐ Beide können von Instruktions-Caches profitieren.

Aufgabe 2 (Assembler und Leistungsbewertung) **[20 Punkte]**

a) In die MIPS-Assembler-Implementierung der Ackermannfunktion $a(n, m)$ haben sich einige Fehler eingeschlichen. Identifizieren und markieren Sie die Fehler und geben Sie korrekte Versionen der entsprechenden Codezeilen in den zugehörigen Kommentarfeldern an. Das Korrigieren richtigen Codes führt zu Punktabzug. Die Ackermannfunktion:

$$\begin{aligned} a(0, m) &= m + 1 \\ a(n + 1, 0) &= a(n, 1) \\ a(n + 1, m + 1) &= a(n, a(n + 1, m)) \end{aligned}$$

Die Parameter n und m werden in den Registern $a0 und $a1 übergeben. Das Resultat befindet sich im Register $v0.

```
 1  ackermann:
 2          sw      $ra,        -4($sp)                 ____________________
 3          sw      $a1,        -8($sp)                 ____________________
 4          sw      $a0,        -12($sp)                ____________________
 5          subi    $sp,        $sp,          12        ____________________
 6          slt     $t0,        $zero,        $a0       ____________________
 7          beq     $t0,        $zero,        NNichtNull ____________________
 8          addi    $v0,        $a1,          1         ____________________
 9          jal     return                              ____________________
10                                                      ____________________
11  NNichtNull:                                         ____________________
12          slt     $t0,        $zero,        $a0       ____________________
13          bne     $t0,        $zero,        MNichtNull ____________________
14          addi    $a0,        $a0,         -1         ____________________
15          addi    $a1,        $zero,        1         ____________________
16          j       ackermann                           ____________________
17          jal     return                              ____________________
18                                                      ____________________
19  MNichtNull:                                         ____________________
20          addi    $a1,        $a1,        -1          ____________________
21          jal     ackermann                           ____________________
22          xor     $a1,        $v0,        $zero       ____________________
23          addi    $a0,        $a0,        -1          ____________________
24          j       ackermann                           ____________________
25                                                      ____________________
26  return:                                             ____________________
27          subi    $sp,        $sp,        -12         ____________________
28          lw      $ra,        -4($sp)                 ____________________
29          lw      $a1,        -8($sp)                 ____________________
30          lw      $a0,        -12($sp)                ____________________
31          jr      $ra                                 ____________________
```

b) Die Funktion `ackermann` wird mit den Parametern $n = 1$ und $m = 0$ ausgeführt. Welches Ergebnis wird berechnet?

Notieren Sie die Werte der Register $a0, $a1 und $sp bei jedem Aufruf der Ackermannfunktion und zwar nach dem Abarbeiten der fünften Zeile. Der Stackpointer wird mit 0x100 vor dem ersten Aufruf der Ackermannfunktion initialisiert.

$sp										
$a0										
$a1										

Ersatztabelle

$sp										
$a0										
$a1										

c) Gegeben sei ein 2 GHz schneller Prozessor in Mehrzyklenimplementierung. Berechnen Sie die Ausführungszeit der Ackermannfunktion für $n = 1$ und $m = 0$, indem Sie für jeden Aufruf der Ackermannfunktion die Anzahl der Instruktionen einer bestimmten Instruktionsklasse in der unten gegebenen Tabelle notieren und dann die Gesamtzahl der Zyklen pro Instruktionsklasse sowie über alle Instruktionsklassen bilden. Bestimmen Sie dann den durchschnittlichen CPI Wert.

Instruktionsklasse$_i$	ack(1,0)	ack(,)			$\sum$	CPI$_i$	$\sum$ Zyklen
R-typ						4	
load word						5	
store word						4	
branch						3	
jump / return						2	
						Gesamtanzahl Zyklen:	

Ersatztabelle

Instruktionsklasse$_i$	ack(1,0)	ack(,)			$\sum$	CPI$_i$	$\sum$ Zyklen
R-typ						4	
load word						5	
store word						4	
branch						3	
jump / return						2	
						Gesamtanzahl Zyklen:	

`CPI=_____________________`

Aufgabe 3 (Leistungsbewertung) **[15 Punkte]**

Ein Prozessor mit einer wie in der Vorlesung besprochenen Mehrzyklenimplementierung soll nur eine begrenzte Auswahl von Programmen ausführen. Von daher ist man in der Lage, einen typischen Instruktionsmix anzugeben:

Instruktion	Häufigkeit	CPI-Wert
Arithmetik	70%	4
Load	10%	5
Store	15%	4
Jump / Branch	5%	3

Der Prozessor habe eine Taktfrequenz von 500 MHz.

(a) Berechnen Sie den Speedup für den Fall, dass die Arithmetik-Instruktionen doppelt so schnell ausgeführt werden können.

(b) Anstatt die Arithmetik-Instruktionen zu beschleunigen, wird nun überlegt, den Prozessor mit einer Pipeline auszustatten, um den Instruktionsdurchsatz zu erhöhen. Für den gegebenen Instruktionsmix stellt man dabei fest, dass 60% der Load-Instruktionen zu einem Datenkonflikt führen, welche zu einem Pipeline-Stall von jeweils 2 Taktzyklen führen. Außerdem führen die Jump-Instruktionen zu einem Pipeline-Stall von jeweils 1 weiteren Taktzyklus. 20% der Arithmetik-Instruktionen stehen ebenfalls im Konflikt. Wie viele Pipeline-Stall-Zyklen darf im Schnitt ein Konflikt bei den Arithmetik-Instruktionen maximal haben, damit ein Speedup von mindestens 3 gegenüber der ursprünglichen Implementierung ohne Verbesserungen erreicht wird?

(c) Als eine weitere Optimierungsmöglichkeit kann man getrennte Caches für Daten und Instruktionen einführen, um die Zugriffszeit auf den Hauptspeicher zu optimieren. Die Caches unterscheiden sich dabei nur in ihrer Miss-Rate. Durch die Caches kann die Taktperiodendauer des Prozessors auf $0,4$ ns gesenkt werden, so dass bei einem Cache-Hit nur ein Taktzyklus benötigt wird. Ein Cache-Miss führt nun allerdings dazu, dass der Prozessor 4 weitere Taktzyklen warten muss, um das Ergebnis aus dem Hauptspeicher bereit zu stellen. Der Instruktions-Cache habe eine Miss-Rate von 10%. Wie groß darf die Miss-Rate des Daten-Caches maximal sein, damit ein Speedup von $4,5$ gegenüber der ursprünglichen Implementierung ohne Verbesserungen erreicht werden kann?

Aufgabe 4 (Multi-level Cache) **[20 Punkte]**

Ein 2 GHz Prozessor hat vier Caches, wie in Abbildung 1 dargestellt, je einen Level 1 Cache für Instruktionen und Daten, einen gemeinsamen Level 2 und einen gemeinsamen Level 3 Cache. Die Anbindung zum Hauptspeicher ist mit einer einfachen Speicherorganisation realisiert, so dass die Worte nacheinander aus dem Speicher geholt und übertragen werden. Dabei muss, für zusammenhängende Speicherblöcke, nur einmal die Adresse übertragen werden. Der Speicherbus hat eine Taktfrequenz von 200 MHz, also eine Taktperiode von 5 ns. Im Folgenden bezeichnet cycles_P Taktzyklen des Prozessors und cycles_M Taktzyklen des Speicherbusses.

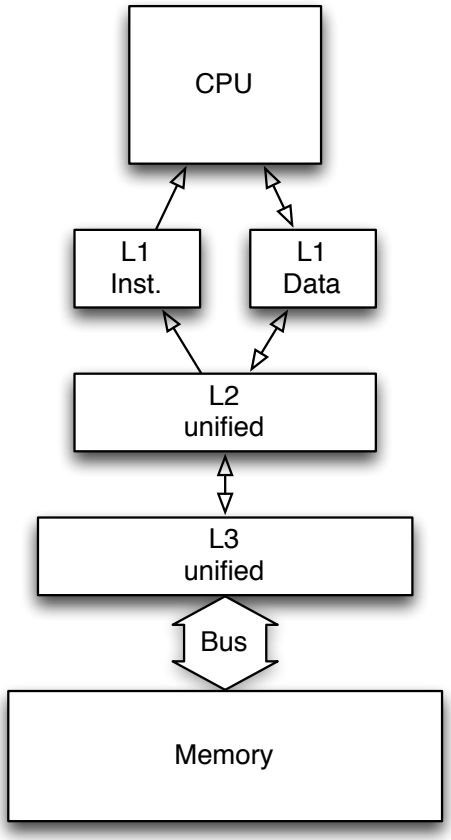

Abbildung 1: Skizze der Speicherorganisation

Die Caches haben folgende Charakteristika:

Cache	Größe	Blockgröße	hit-time	miss-rate
Level 3	6 MB	16 Worte	35 cycles_P	1 %
Level 2	1 MB	4 Worte	8 cycles_P	5 %
Level 1 Daten	64 KB	4 Worte	1 cycle_P	10 %
Level 1 Instruktionen	64 KB	4 Worte	1 cycle_P	2 %

Das Speichersystem hat folgende Charakteristika:

Takt	Speicher-breite	Bus-breite	Adresse senden	Speicherzugriff	Datentransfer
200 MHz	1 Wort	1 Wort	2 cycles_M	26 cycles_M	2 cycles_M

(a) Wie hoch ist die miss-penalty des Level 3 Caches im System aus Abbildung 1, also wie lange braucht das System zur Übertragung eines Cacheblocks aus dem Hauptspeicher? Benennen Sie die dabei Komponenten der miss-penalty Formel und geben Sie das Ergebnis in Nanosekunden an!

(b) Berechnen Sie die mittlere Zugriffszeit für Instruktionen und die für Daten am jeweiligen Level 1 Cache. Arbeiten Sie sich dafür vom L3 Cache hoch und geben Sie jeweils die mittlere Zugriffszeit der Stufe an.

(c) Statt einer Speicherbank haben Sie nun 8 Speicherbänke zur Verfügung, sodass Sie eine Speicherbreite von 8 realisieren, also 8 Worte gleichzeitig aus dem Speicher laden können. Entwerfen Sie damit eine Speicherorganisation, die die mittlere Zugriffszeit am Level 3 Cache auf unter 45 cycles_P senkt. Beschreiben Sie Ihre Speicherorganisation und nennen Sie Vor- und Nachteile im Vergleich zur einfachen.

Aufgabe 5 (Busarbitrierung) **[20 Punkte]**

Abbildung 2 zeigt vier Einheiten (E1...E4) und ein Memory, die an einen synchronen Bus angeschlossen sind. Es wird das Daisy-Chain Arbitrierungsschema verwendet. Zusätzlich gilt folgende Timeout-Strategie: Wenn eine Einheit zum Zeitpunkt t das `Request` Signal setzt und 5 Takte lang kein `Grant` erhält, setzt sie zum Zeitpunkt $t+6$ das `Request` Signal erneut.

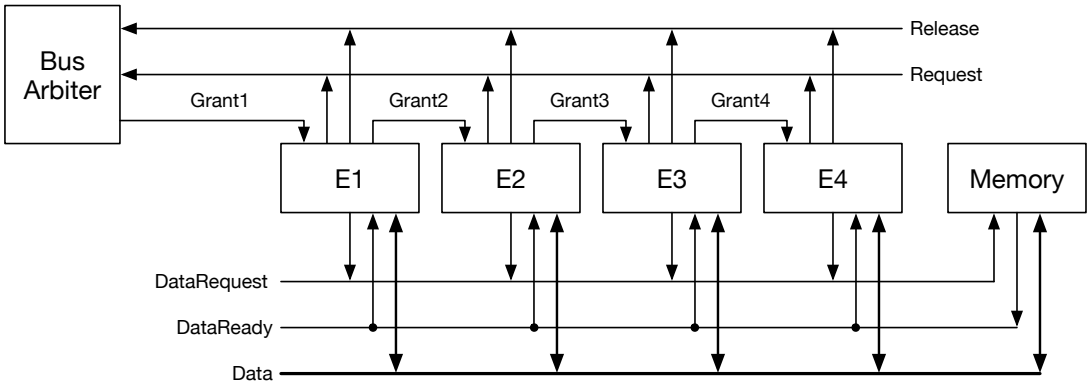

Abbildung 2: Daisy Chain Architektur

Das Senden von Speicheradresse und Daten erfolgt gemultiplext und dauert jeweils 2 Takte, dazwischen gibt es ein Delay von 1 Takt. Die Signale `Release`, `Request` und `Grant` sind nur für einen Takt auf 1 gesetzt.

(a) Füllen Sie das Timing Diagramm in Abbildung 3 für den Fall aus, dass E1, E4 und E3 zu den Zeitpunkten 1, 3 und 11 von dem Memory lesen wollen. Geben Sie bei `Data` „A“ an, wenn eine Adresse gesendet wird und „D“ wenn Daten gesendet werden.

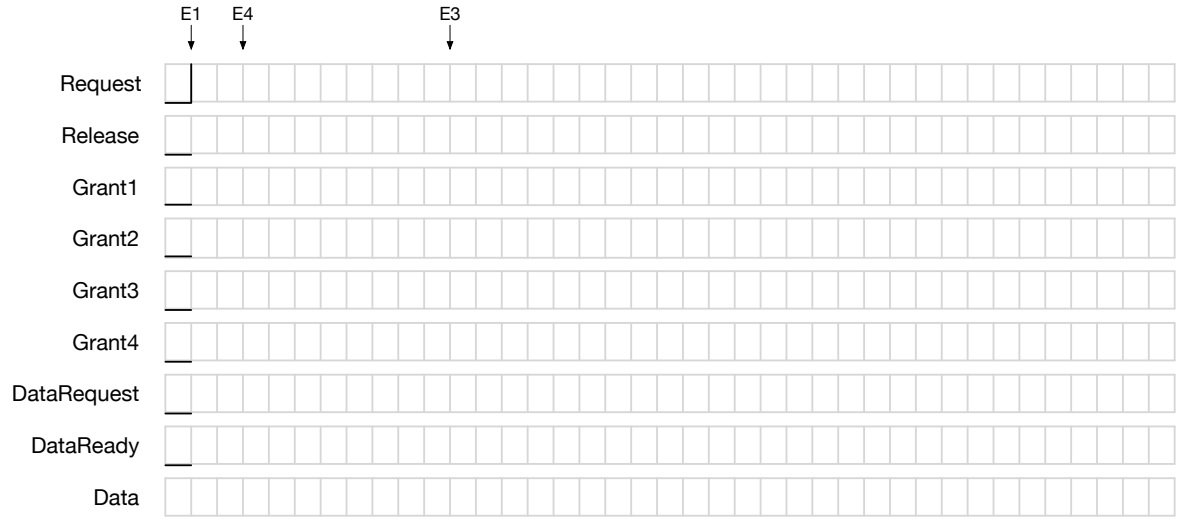

Abbildung 3: Timing-Diagramm

(b) Entwerfen Sie den Controller von E1 wie in Abbildung 4 dargestellt als Moore-Automaten. Gehen Sie davon aus, dass zusätzlich folgende Signale für Eingänge zur Verfügung stehen:

`Requested` → Logic der Einheit will `Request` setzen
`Timeout` → Einheit hat 5 Takte erfolglos auf `Grant` gewartet

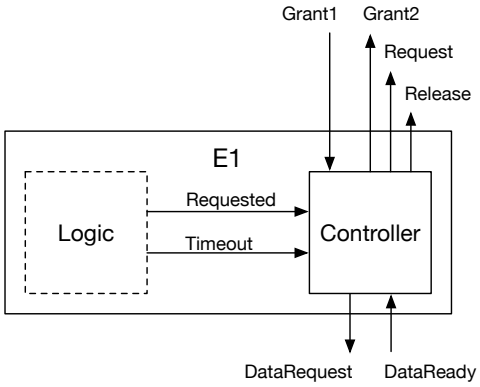

Abbildung 4: E1 Controller

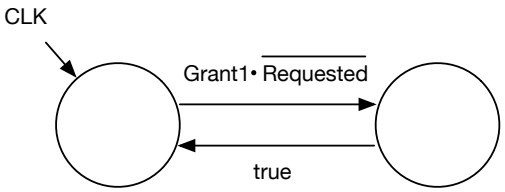

Abbildung 5: Moore-Automat

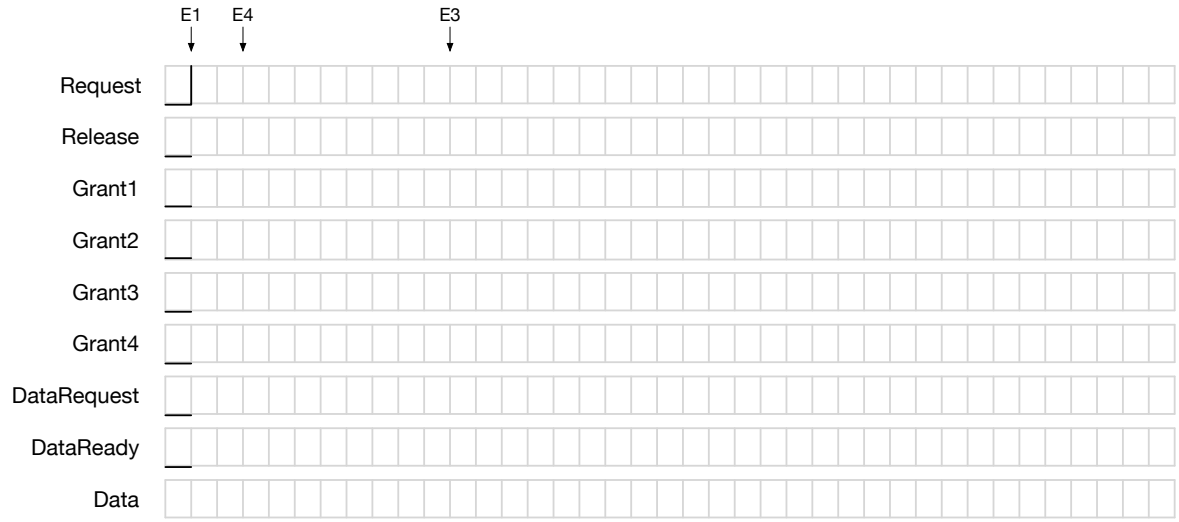

Abbildung 6: Timing-Diagramm (Ersatzlösung, falsche Lösung streichen)

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!

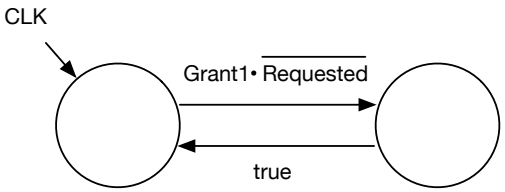

Abbildung 7: Moore-Automat (Ersatzlösung, falsche Lösung streichen)