

# Übungsblatt 11

(10 Punkte)

**Besprechung ab Montag, 29. Juni 2015**

## 11.1 Kohärenz versus Konsistenz (1 Punkt)

Beschreiben Sie jeweils mit Ihren eigenen Worten, was mit *Cachekohärenz* und *Speicherkonsistenz* gemeint ist und verdeutlichen Sie dies jeweils anhand eines einfachen Beispiels. Gibt es einen Zusammenhang zwischen Speicherkonsistenz und Cachekohärenz? Wenn ja, welchen?

## 11.2 Sequentielle Speicherkonsistenz (2 Punkte)

Es wird angenommen, dass drei Prozessoren  $P_1, P_2, P_3$  eines Mehrprozessorsystems auf einen gemeinsamen Speicher zugreifen und dass das System sequentiell konsistent ist. Vor Ausführung des Programmcodes gilt  $A = B = 0$ . Die drei Prozessoren führen nun eine Reihe von Operationen gemäss der unten stehenden Tabelle aus. Dabei ist  $o_{i1}$  die erste Operation, die Prozessor  $P_i$  ausführt, und  $o_{i2}$  die zweite.

$P_1$	$P_2$	$P_3$
$o_{11} : A = 1$	$o_{21} : u = A$ $o_{22} : B = 1$	$o_{31} : v = B$ $o_{32} : w = A$

Das Ergebnis der Ausführung kann durch das Zahlentupel  $(u, v, w)$  beschrieben werden. Dabei kann das Ergebnis variieren und zwar abhängig von der Reihenfolge, in der die Operationen der einzelnen Prozessoren global sichtbar gemacht werden. Einige der in der Tabelle unten gezeigten Tupel sind auf einem sequentiell konsistenten System nicht möglich. Füllen Sie die Tabelle folgendermassen aus (siehe auch das Beispiel in Zeile 2): Geben Sie an, ob das Tupel in der jeweiligen Zeile auf einem sequentiell konsistenten System als Ergebnis auftauchen kann und falls ja, nennen Sie eine mögliche Reihenfolge, in der die Operationen der einzelnen Prozessoren global sichtbar gemacht wurden.

$u$	$v$	$w$	konsistent?	Reihenfolge
0	0	0		
0	0	1	Ja	$o_{21}, o_{31}, o_{11}, o_{22}, o_{32}$
0	1	0		
0	1	1		
1	0	0		
1	0	1		
1	1	0		
1	1	1		

### 11.3 Speicherkonsistenzmodelle (4 Punkte)

Neben der sequentiellen Konsistenz wurden in der Vorlesung auch eingeschränktere Modelle von Speicherkonsistenz eingeführt. Betrachten Sie das folgende MIPS-Programmstück, das von zwei Out-Of-Order Prozessoren ausgeführt wird.

	$P_1$	$P_2$
	li \$a, 0	li \$e, 1
$o_{11}$ :	sw \$a, A	$o_{21}$ : sw \$e, E
$o_{12}$ :	lw \$c, C	$o_{22}$ : acquire L
$o_{13}$ :	lw \$d, D	$o_{23}$ : lw \$f, F
	add \$x, \$c, \$d	$o_{24}$ : sw \$f, R
$o_{14}$ :	sw \$x, X	$o_{25}$ : release L
		li \$g, 3
		$o_{26}$ : sw \$g, G

Im Beispiel sind alle Großbuchstaben feste, jeweils verschiedene Speicheradressen. Nehmen Sie an, dass *acquire* einer *swap*-Anweisung entspricht (atomare Lese- und Schreibzugriff auf *L*) und dass *release* nur einen Schreibzugriff auf *L* durchführt. Für diese Aufgabe sind folgende Konsistenzmodelle relevant:

- Sequentielle Konsistenz (SK)
- Total Store Order (TSO)
- Partial Store Order (PSO)
- Weak Ordering (WO)
- Release Consistency (RC)

a) Geben Sie für jede der folgenden Anordnungen *alle* der genannten Konsistenzmodelle an, denen sie genügt. Hierbei ist es irrelevant, ob eine Ausführung der Instruktionen in dieser Reihenfolge die Programmsemantik verletzt, es geht nur darum, ob die Sequenz dem jeweiligen Konsistenzmodell genügt.

Nr	Sequenz									
1	$o_{11}$	$o_{12}$	$o_{21}$	$o_{13}$	$o_{22}$	$o_{23}$	$o_{14}$	$o_{24}$	$o_{25}$	$o_{26}$
2	$o_{11}$	$o_{22}$	$o_{23}$	$o_{24}$	$o_{25}$	$o_{21}$	$o_{12}$	$o_{13}$	$o_{14}$	$o_{26}$
3	$o_{14}$	$o_{12}$	$o_{13}$	$o_{22}$	$o_{24}$	$o_{23}$	$o_{21}$	$o_{25}$	$o_{26}$	$o_{11}$
4	$o_{12}$	$o_{13}$	$o_{21}$	$o_{11}$	$o_{22}$	$o_{23}$	$o_{14}$	$o_{24}$	$o_{25}$	$o_{26}$
5	$o_{14}$	$o_{12}$	$o_{13}$	$o_{21}$	$o_{22}$	$o_{23}$	$o_{24}$	$o_{25}$	$o_{26}$	$o_{11}$
6	$o_{14}$	$o_{12}$	$o_{13}$	$o_{22}$	$o_{23}$	$o_{21}$	$o_{24}$	$o_{25}$	$o_{26}$	$o_{11}$

**Hinweis:** Es ist hierbei hilfreich zuerst einen Graphen mit der im jeweiligen Modell zu erhaltenden Reihenfolge-Relation zu zeichnen, in dem Kanten für zu erhaltende Abhängigkeiten stehen (Analog zur Vorlesung). Transitive Kanten sollten der Übersicht halber nicht eingezeichnet werden.

b) Welche Reihenfolgen erhalten die Semantik des Programms?

## 11.4 Locks und Synchronisation (3 Punkte)

In der Vorlesung wurden verschiedene Methoden zur hardwareunterstützten Realisierung eines Locks gezeigt.

- Test & Set
- Test & Set mit exponentiellen Backoff
- Test & Test & Set
- Load-Linked/Store-Conditional
- Fetch & Increment

Skizzieren Sie das Vorgehen der einzelnen Verfahren in Pseudocode und stellen Sie dabei die Eigenschaften der Verfahren gegenüber.

**Allgemeine Hinweise:** Die Übungstermine und weitere Informationen finden Sie unter <http://ls12-www.cs.tu-dortmund.de/daes/de/lehre/lehrveranstaltungen/sommersemester-2015/rechnerarchitektur.html>. Die Übungszettel werden zum Semesterbeginn online gestellt und sollen eigenständig bis zum jeweiligen Stichtag gelöst werden. Die Lösungen werden in den Gruppen besprochen. Auf Wunsch kann für diese Veranstaltung ein Übungsschein ausgestellt werden. Hierzu müssen die selbst erstellten Lösungen jeweils vor der Besprechung der Aufgaben beim Übungsgruppenleiter abgegeben werden. Dabei müssen 45% der Gesamtpunkte bei den Übungszetteln erreicht und eigene Lösungen in der Übungsgruppe präsentiert werden. Für die Teilnahme an der Klausur nach BPO 2013 / der Fachprüfung nach DPO 2001 ist der Übungsschein *nicht* erforderlich.