lea.schoenberger [☺] tu-dortmund.de
kuan-hsun.chen [☺] tu-dortmund.de
nils.hoelscher [☺] tu-dortmund.de

Übung zur Vorlesung
Rechnerarchitektur
Sommersemester 2019

Übungsblatt 4

Bearbeitung ab Mittwoch, 22. Mai 2019

4.1 Klassen von Multiprozessorsystemen

Ordnen Sie die vier Klassen von Multiprozessorsystemen nach der Klassifikation von Flynn in die folgende Tabelle und beschreiben Sie eine wichtige Eigenschaft jeder der vier Klassen kurz.

		Befehlsströme	
		1	>1
Datenströme	1		
	>1		

Klasse: _____
Eigenschaft:

Klasse: _____
Eigenschaft:

Klasse: _____
Eigenschaft:

Klasse: _____
Eigenschaft:

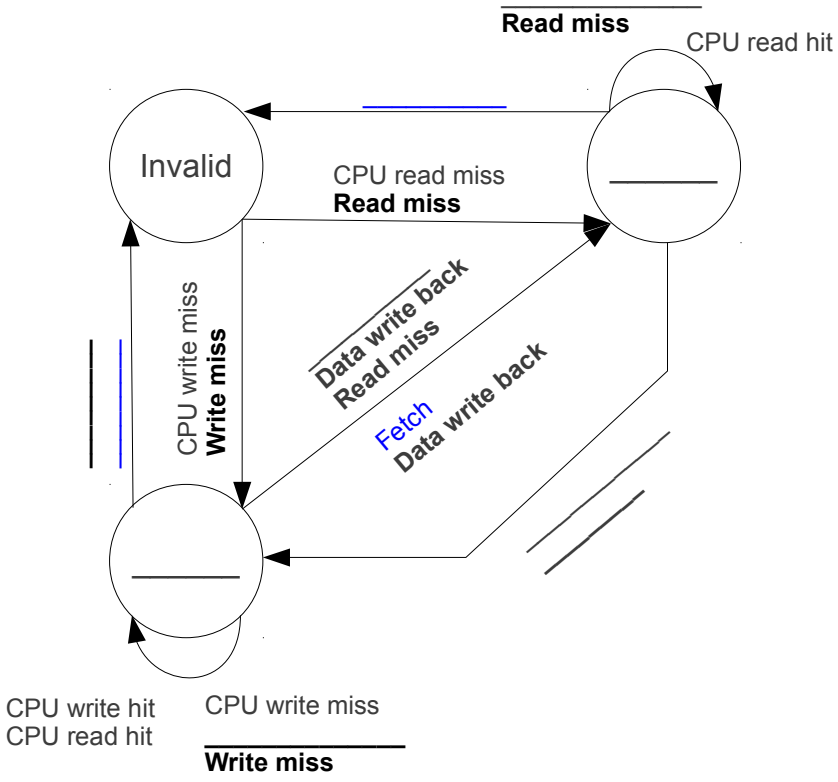
4.2 Cachekohärenz

(a) Geben Sie eine möglichst kompakte Definition von *Cachekohärenz* an.

(b) Nennen Sie die beiden grundlegenden Handlungsmöglichkeiten, die Cachekohärenzprotokolle unabhängig von der Art ihrer Umsetzung (bus- oder verzeichnisbasiert) haben, um Kohärenz in einem System aufrechtzuerhalten. Es ist keine Erklärung der Funktionsweise der Verfahren nötig.

(c) Cachekohärenz für Write-Back-Caches kann mithilfe des in der Vorlesung vorgestellten verzeichnisbasierten MSI-Protokoll erzielt werden. Die folgende Illustration stellt das Zustandsdiagramm eines Cache-Blocks in diesem Protokoll dar. Ergänzen Sie die fehlenden Beschriftungen sowohl für die *Zustände* als auch die *Zustandsübergänge*. Das Eingabe-/Ausgabealphabet des Zustandsautomaten ist nachfolgend angegeben.

Eingaben	Ausgaben
CPU read miss	Read Miss
CPU read hit	Write Miss
CPU write miss	Data Write Back
CPU write hit	
Invalidate	
Fetch	
Fetch/Invalidate	



4.3 Parallelisierung von Conways Spiel des Lebens (Game of Life) mit OpenMP

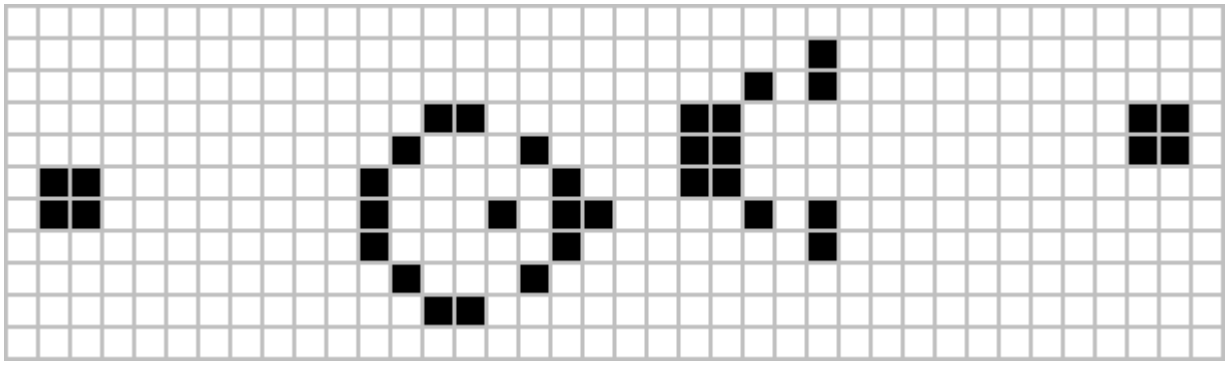


Abbildung 1: Spiel des Lebens Beispiel - Gun Slider Muster [www.wikipedia.org]

Das Spielfeld besteht aus $v \times h$ Elementen, wie in Abbildung 1 zu sehen ist. Jede Zelle heißt dabei lebendig, wenn ihr Wert 1 (schwarzes Feld) ist, sonst tot (Wert 0, weißes Feld). Die Belegung des Spielfelds ändert sich iterativ, d.h., beginnend von einer Startbelegung wird in einer Iteration das gesamte Feld aktualisiert. Nach der Aktualisierung ist die aktuelle Iteration abgeschlossen und der Vorgang wiederholt sich bis zum Erreichen einer festgelegten Grenze. Es gibt Regeln, welche beschreiben, wie sich die Belegung einer Zelle in Abhängigkeit von ihren direkten Nachbarzellen verändert.
Aktualisierungsregeln einer Zelle:

- Eine tote Zelle mit genau drei lebenden Nachbarn wird in der folgenden Iteration neu geboren.
- Lebende Zellen mit weniger als zwei lebenden Nachbarn sterben in der folgenden Iteration an Einsamkeit.
- Eine lebende Zelle mit zwei oder drei lebenden Nachbarn bleibt in der folgenden Iteration am Leben.
- Lebende Zellen mit mehr als drei lebenden Nachbarn sterben in der folgenden Iteration an Überbevölkerung.

Analysieren Sie den Sourcecode des Game of Life im SVN-Repository (“04_2.c” oder “04_2.cpp”).
Optional: Implementieren Sie zu Hause Ihre eigene Version vom Spiel des Lebens in C oder C++.

a. Stellen Sie sicher, dass jedes parallelisierte Programm semantisch korrekt ist und dasselbe Ergebnis wie die sequentielle Version liefert.

b. Führen Sie die Beispiele RA und BIG (in der Datei “muster.txt” im SVN-Repository) mit unterschiedlichen Anzahlen paralleler Threads aus und vergleichen Sie.

Allgemeine Hinweise: Informationen zur Veranstaltung finden Sie unter https://ls12-www.cs.tu-dortmund.de/daes/lehre/lehrveranstaltungen/sommersemester-2019/rechnerarchitektur-ss19.html. Eine Abgabe der Übungsblätter ist nicht erforderlich.