Rechnerarchitektur SS 2017

Exercises: Scoreboarding

Jian-Jia Chen

TU Dortmund

to be discussed on June, 06, 2017

Scoreboardings

Im nächste Folien finden Sie ein Beispiel zu den gespeicherten Informationen innerhalb der Scoreboarding-Datenstrukturen, für einen aus der Vorlesung bekannten Beispielcode. Konkret ist in den Tabellen folgende Situation festgehalten:

- erste Lade-Instruktion ist vollständig verarbeitet und hat ihr Ergebnis in der *write result*-Phase geschrieben.
- zweite Lade-Instruktion hat die Ausführungsphase abgeschlossen und wartet auf das Schreiben ihres Ergebnisses.
- restliche Instruktionen befinden sich vor bzw. in der *issue*-Phase.

Exercise - Scoreboardings

Scoreboarding: Takt +0

Befehls-Status

Befehl		*Issue*	*Read Op.*	*Exec. complete*	*Write*
`L.D`	`F6,32(R2)`	+	+	+	+
`L.D`	`F2,96(R3)`	+	+	+	
`MUL.D`	`F0,F2,F4`	+			
`SUB.D`	`F8,F6,F2`	+			
`DIV.D`	`F10,F0,F6`	+			
`ADD.D`	`F6,F8,F2`				

Status der Funktionseinheiten

Name	*Busy*	*Op*	*Fi*	*Fj*	*Fk*	*Qj*	*Qk*	*Rj*	*Rk*
Integer	ja	Load	F2	R3				nein	
Mult1	ja	Mult	F0	F2	F4	Int.		nein	ja
Mult2	nein								
Add	ja	Sub	F8	F6	F2		Int.	ja	nein
Divide	ja	Div	F10	F0	F6	Mult1		nein	ja

Register-Ergebnis-Status

	F0	*F2*	*F4*	*F6*	*F8*	*F10*
Einheit	Mult1	Int			Add	Div

Exercise - Scoreboardings

Führen Sie die Programmausführung für die nächsten 10 Takte fort und stellen Sie für jeden Takt den aktuellen Stand der Scoreboarding-Datenstrukturen am Taktende dar. Benutzen Sie hierfür die Vorlage *scoreboarding.ppt* von der Webseite der Übung. Verwenden Sie für diese Aufgabe folgende Latenzzeiten der *execute*-Phase:

- Addition: 2 Taktzyklen
- Multiplikation: 5 Taktzyklen
- Division: 8 Taktzyklen

Die Latenzzeiten sind so zu verstehen, daß die Execute-Phase exakt die angegebene Anzahl Takte benötigt um zu terminieren, d.h. wenn beispielsweise eine Addition in Takt x gestartet wird endet sie am Ende von Takt $x + 1$. *Forwarding* von Ergebnissen wird nicht unterstützt, d.h. Ergebnisse die in der Write-Results-Phase geschrieben werden, können erst im nächsten Takt in der Read-Operands-Phase gelesen werden. Außerdem werden keine Operanden gepuffert, d.h. wenn ein Operand verfügbar ist, ein anderer aber noch nicht, werden beide erst gelesen wenn auch beide verfügbar sind.