University of the Saarland
Department 6.2 - Informatik
Prof. Dr. W.J. Paul

Computer Architecture I - WS 02/03

(bis: Montag 27.01.2003)

Aufgabe 1: (Forwarding von Stage 3) **(10 Punkte + 5 Bonuspunkte)**

Im Fall einer Load Instruktion ist Forwarding von Stage 3 mit der Forwarding Engine aus der Vorlesung nicht möglich. Der Grund dafür ist, dass die hit_i Signale aktiv sind, falls eine Instruktion in der entsprechenden Stage ist ($full_i = 1$) und das richtige Register als Zielregister hat. Im Fall einer Load Instruktion liegt das richtige Ergebnis aber erst in Stage 4 (nach dem Shift4load) vor.

- Ändern Sie die Konstruktion der DLX so ab, dass Ihre Variante die Induktionsvorraussetzung über Load Befehle (wenn I_i Register $GPR(r)$ liest, dann sind I_{i-1} und I_{i-2} keine Load-I struktionen in Register $GPR(r)$) nicht mehr nötig ist. Dazu müssen sie die Definition der Hazard-Signale anpassen.
- Wo ändert sich dadurch der Korrektheitsbeweis?

Aufgabe 2: (Formalisierung) **(2 + 2 Punkte)**

Benutzen Sie die Scheduling-Funktion $I_\pi(k, T)$ um die folgenden Sachverhalte auszudrücken:

- Instruktionen in der Pipeline überholen sich nicht gegenseitig.
- Es sind nie zwei Instruktionen gleichzeitig in der gleichen Stufe.

Aufgabe 3: (Schnelle Etall Engine) **(6 Punkte)**

Die Stall Engine aus der Vorlesung hat ein Delay von $O(n)$ (n ist die Anzahl der Stufen). Der Grund hierfür liegt in der Defintion der Stall-Signale: k: $stall_k = full_k \wedge (hazard_k \vee stall_{k+1})$. Dadurch hängt $stall_0$ von allen anderen Stall-Signalen ab und hat damit in der offensichtlichen Konstruktion ein Delay von $O(n)$.

Entwickeln Sie eine Stall Engine, die die $stall_i$ Signale mit einem Delay von $O(\log n)$ berechnet.