Übungsblatt 5

5.1 Multi-Cycle Datenpfad: lw-Instruktion

Erläutern Sie die Ausführung der lw-Instruktion (load word) für den Multi-Cycle-Datenpfad. Welche Vorteile bietet der Multi-Cycle-Datenpfad gegenüber dem Single-Cycle-Datenpfad? Diskutieren Sie dies anhand dieses Befehls.

5.2 Grundlagen Pipelining

Gegeben seien vier unterschiedliche Prozessoren, die sich in der Anzahl der Pipelinestufen und der Taktrate unterscheiden:

Prozessor	Pipelinestufen	Taktrate
Prozessor A	1	100 MHz
Prozessor B	4	800 MHz
Prozessor C	12	1,5 GHz
Prozessor D	20	3,2 GHz

(a) Bestimmen Sie für jeden Prozessor die Latenz der einzelnen Instruktionen.

(b) Wie lange dauert die Ausführung von 400.000 voneinander unabhängigen Instruktionen auf jedem der angeführten Prozessoren? Bestimmen Sie die Performance und den Speedup verglichen mit Prozessor A ohne Pipelining. (Sie können annehmen, dass es keine Stalls gibt.)

5.3 Pipelining: Graphische Darstellung

Beantworten Sie folgende Fragen anhand der Beispiel-Pipeline-Architektur der VO (Kapitel 3.2).

(a) Bestimmen Sie die Anzahl der Pipelinestufen, die Taktdauer und die Taktfrequenz der Beispiel-Pipeline unter Annahme der Angaben auf VO-Folie 3-44 (Ausführungszeiten der Funktionseinheiten). Wie lange dauert die Ausführung eines einzigen Befehls auf der Beispiel-Pipeline?

(b) Angenommen es treten keine Leertakte (stalls) auf, welchen Speedup erreicht die Beispiel-Pipeline aus a) gegenüber einem Single-Cycle Datenpfad, der aus den gleichen Funktionsregistern besteht?

(c) Auf der Pipeline wird folgende Befehlssequenz ausgeführt:

```
and     $10, $2, $3
sw      $11, 4($3)
```

Stellen Sie die Ausführung der oben angeführten Befehlssequenz durch die Beispiel-Pipeline wie auf VO-Folie 3-45 grafisch dar (untere Abbildung). Achten Sie insbesondere auf die zeitliche Anordnung der Zugriffe auf die Registereinheit! Wie lange dauert die Ausführung der Befehlssequenz?

5.4 Pipelining: Daten- und Kontrollabhängigkeiten

Gegeben sei folgendes Code-Fragment:

```
        addi $t5, $t5, 42
        addi $t0, $a0, 516
loop:   addi $t0, $t0, -4
        lw   $t1, 0($t0)
        add  $t2, $t1, $t5
        sw   $t2, 0($t0)
        bne  $t0, $a0, loop
        nop
```

(a) Identifizieren Sie alle Daten- und Kontrollabhängigkeiten, die Leerzyklen (Stalls) verursachen (Annahme: ohne Forwarding-Einheit). Wie viele Taktzyklen werden für die Ausführung des gesamten Codes bzw. pro Ergebniselement (d.h. nur die Schleife) benötigt?

(b) Welche aus den Datenabhängigkeiten resultierenden Pipeline-Konflikte können durch eine Forwarding-Einheit gelöst werden? Wie viele Taktzyklen benötigt die Ausführung der Befehlssequenz mit Forwarding-Einheit für die Ausführung des gesamten Codes bzw. pro Ergebniselement (d.h. nur die Schleife)? Wo und warum muss die Hardware Leerzyklen (d.h. Stalls) einfügen?

(c) Ordnen Sie den Code so um, dass er auf einer modifizierten Beispiel-Pipeline mit Unterstützung für „Delayed Branching“ (d.h. Branch-Delay-Slot) möglichst rasch ausgeführt wird (und die Semantik erhalten bleibt). Wie viele Taktzyklen werden in diesem Fall für die Ausführung des gesamten Codes bzw. pro Ergebniselement (d.h. nur die Schleife) benötigt? Wie hoch ist der erzielte Speedup relativ zu bzw. b)?