



Übungen zur Vorlesung

Technische Informatik II, WS 2003/2004

{kaiser,hubert.piontek}@informatik.uni-ulm.de

Lösungsskizze zum Übungsblatt 8

Aufgabe 1: CISC vs. RISC

Erkennungsmerkmale der Architekturen:

RISC	CISC
Single Cycle Instruktionen	meist deutlich mehr als 1 Cycle/Instruktion
wenige Instruktionen fest verdrahtete Instruktionen wenige Adressierungsarten einfache Befehle ⇒ optimierender Hochsprachencompiler	vielfältige Instruktionen Mikroprogrammierung vielfältige Adressierungsarten komplexe Operationen in einem Befehl möglich ⇒ sogar Assemblerprogrammierung ist „bequem“
sehr kleine Anzahl an Befehlsformaten	Befehlsformate können vielfältig sein
Load/Store Architektur	Reg/Mem oder Mem/Mem Befehle möglich
Pipelining	

„Viermal so schnell“:

Sie kennen aus dem Skript die Zusammenhänge

$$CPUtime = Taktzyklen \cdot Taktdauer$$

und

$$CPI = \frac{Taktzyklen}{Instruktionsanzahl}$$

bzw. zusammengefasst:

$$CPUtime = Instruktionsanzahl \cdot CPI \cdot Taktdauer$$

Setzen wir an:

$$RISC : 1 = 1,3 \cdot IC \cdot 1,4 \cdot \frac{1}{f_{RISC}}$$

und

$$\text{CISC} : 4 = 1 \cdot IC \cdot 8 \cdot \frac{1}{100}$$

oder

$$IC = \frac{4 \cdot 100}{8}$$

bzw.

$$f_{RISC} = 1,3 \cdot IC \cdot 1,4 = 1,3 \cdot \frac{4 \cdot 100}{8} \cdot 1,4 = 91$$

Der beschriebene RISC-Prozessor muss also mit 91MHz getaktet werden, um viermal so schnell zu sein wie der beschriebene CISC-Prozessor mit 100MHz.

Werbung über den Prozessortakt sollte folglich ignoriert werden, da dieser alleine nicht sehr viel über die relative Leistung zu Prozessoren anderer Architektur aussagt.

Aufgabe 2: Pipelining

Hauptphasen der Befehlsausführung:

1. IF (Instruction Fetch) die nächste Instruktion wird geladen
2. ID (Instruction Decode and Operand Fetching) der Opcode und die Operanden werden decodiert und die passenden Steuersignale erzeugt
3. EX (Execution Phase) Operation wird ausgeführt. Falls ein Speicherzugriff nötig ist, wird die Effektive Adresse erzeugt.
4. ME (Memory Access) Daten werden aus dem Speicher gelesen oder in diesen geschrieben.
5. WB (Write Back) Ergebnis einer Berechnung wird in das Zielregister geschrieben.

Mögliche Probleme und deren Lösung/Minderung

Problem	Lösung
Datenabhängigkeiten	Compiler verhindert Datenabhängigkeiten durch Einfügen von NOPs oder Erkennung in Hardware und Verzögerung der weiteren Ausführung und evtl. Einführung von Bypass-Kanälen
Kontrollabhängigkeiten (Sprünge)	Compiler kann Anzahl an Sprüngen vermindern (Loop unrolling etc.); in Hardware: Verzögern bis der Sprung ausgeführt wurde, branch prediction und delayed branch