Kristina Koshlets
RISC und CISC

**Maximieren der Rate, mit der Instruktionen ausgegeben werden.**

Moderne Computer bedienen sich mancher Tricks, um Ihre Leistung zu maximieren. Einer der häufigsten ist es, möglichst viele Instruktionen pro Sekunde auszugeben. Wenn man 500 Millionen Instruktionen pro Sekunde starten kann, hat man immerhin einen 500-MIPS-Prozessor gebaut, ungeachtet dessen, wie lange die Ausführung dieser Instruktion dauert. (MIPS steht für Millionen Instruktionen pro Sekunde). Dieses Prinzip legt nahe, dass Parallelität bei der Verbesserung des Leistungsverhaltens eine wichtige Rolle spielen kann, da man nur dann eine große Zahl von langsamen Instruktionen in kurzer Zeit starten kann, wenn mehrere davon gleichzeitig ausgeführt werden.

**Die Befehle müssen leicht zu dekodieren sein.**

Eine kritische Grenze für die Rate, mit der Instruktionen gestartet werden können, setzt die Dekodierung der einzelnen Instruktionen, um die von ihnen benötigten Ressourcen zu bestimmen. Alles, was diesen Vorgang unterstützt, ist dienlich. Dazu gehört, die Instruktionen regelgerecht auszulegen und ihnen eine feste Länge mit möglichst weniger Feldern zu geben. Je weniger verschiedene Formate es für die Instruktionen gibt, um so besser.

**Nur Lade- und Speichervorgänge sollen auf den Speicher verweisen.**

Zu den einfachsten Möglichkeiten, Operationen in Einzelschritte aufzulösen, gehört die Forderung ,dass die Operanden für die meisten Instruktionen aus Register kommen und dort wieder abgelegt werden. Operanden aus dem Speicher in Register zu verschieben, ist ein Vorgang ,der in separaten Instruktionen erfolgen kann. Da Speicherzugriffe viel zeit beanspruchen können und die Verzögerung unvorhersehbar sind, eignen sie die entsprechenden Instruktionen besonders gut für dass Überlappen mit anderen Instruktionen ,wenn sie weiter nichts tun, als Operanden zwischen Register und dem Speicher zu bewegen. dies bedeutet, dass nur LOAD- und STORE- Instruktionen auf den Speicher verweisen sollen.

**Parallelität auf Instruktionsebene.**

Die Computer Architekten streben ständig danach, die Leistung der von Ihnen entwickelten Maschinen zu steigern. Ein Weg die Chips zu beschleunigen, besteht in der Erhöhung der Taktfrequenz. Doch bei dem neuen Design gibt es eine Grenze dafür, was für jeweiligen Zeit durch schiere Gewalt gerade noch möglich ist. Daher greifen die meisten Computer Architekten zur Parallelität (lassen also zwei oder mehr Dinge gleichzeitig tun )als weg ,um bei gegebener Taktfrequenz noch mehr Leistung aus ihrem design herauszuholen. Parallelität erscheint in zwei Hauptformen: auf Instruktionsebene und auf Prozessorebene. Im ersten Fall wird die Parallelität innerhalb der einzelnen Instruktionen genutzt, um mehr Instruktionen pro Sekunde aus Maschine herauszuholen. Im zweiten Fall arbeiten mehrere CPUs zusammen am gleichen Problem.

In 2.4 zeigt eine Pipeline mit fünf Einheiten oder Stufen (Stages). Stufe 1 ruft die Instruktion aus dem Speicher ab und stellt sie in einer Puffer, bis sie gebraucht wird. Stufe 2 dekodiert die Instruktion und bestimmt dabei den Typ und die benötigten Operanden. Stufe 3 sucht die Operanden und ruft sie entweder aus Register oder aus dem Speicher ab. Stufe 4 fuhrt die

eigentliche Abarbeitung der Instruktion aus, die normalerweise darin besteht, die Operanden durch den Datenweg zu schleusen. Schließlich schreibt stufe  5 das Ergebnis in das dafür vorgesehene Register.

**Superskalare Architekturen.**

Wenn eine Pipeline gut ist, sind zwei Pipelines sicherlich noch besser. Zeigt ein mögliches Design für eine CPU mit zwei Pipelines auf der Basis von. Hier ruft eine Instruktionsabrufeinheit gleich ein Instruktionspaar ab und platziert jede Instruktion in eine eigene Pipeline, die zwecks paralleler Verarbeitung auch über eine eigene ALU verfügt.
Sie dürfen auch nicht von dem Ergebnis der jeweils anderen abhängen. Wie bei einer einzigen Pipeline muss entweder der Compiler dafür sorgen, dass dies jederzeit gewährleistet ist ( d.h., die Hardware führt keine Prüfung durch und liefert falsche Ergebnisse, wenn die Instruktionen nicht Kompatibel sind), oder Konflikte werden erkannt und mittels zusätzlicher Hardware bei der Ausführung eliminiert.
Zwar werden – einzelne oder doppelte – Pipelines meist bei RISC-Maschinen verwendet, Intel begann aber ab dem 486 damit, Pipelines in ihre CPU einzubauen (der 386 und seine Vorgänger haben keine).
Die Grundidee dabei ist, nur mit einer einzigen Pipeline zu arbeiten, dieser jedoch mehrere Funktionseinheiten zur Verfügung zu stellen.