Constantin Lazari, Marco Wettstein 7. Oktober 2013

1. Gegeben sei ein Prozessor mit einer Taktzykluszeit von 1.25 GHz und einem CPI-Wert von 1.45 (der Prozessor verfügt über keine Pipeline). Ein Programm benötigt zur Ausführung 150 000 Befehle.

   (a) Wie lang ist die ungefähre Ausführungszeit des Programms?

   Lösung:

   $$\text{Taktzyklen: } z = 150\,000 \text{ Befehl} \cdot 1.45 \frac{\text{Taktzyklus}}{\text{Befehl}} = 217\,500 \text{ Taktzyklus}$$

   $$\text{Zeit: } t = \frac{217\,500 \text{ Taktzyklus}}{1.25 \cdot 10^9 \frac{\text{Taktzyklus}}{\text{s}}} = 0.000174 \text{ s} = 174 \ \mu\text{s}$$

   Die Ausführungszeit beträgt 174 μs

   (b) Wieso ist der berechnete Wert nur ein Näherungswert?

   Lösung:
   Der CPI-Wert ist ein geschätzter Mittelwert. Er kann je nach Komplixität der verwendeten Befehle grösser oder kleiner sein. Der CPI-Wert ist also nicht exakt $\Rightarrow$ berechnete Werte sind ebenfalls exakt.

   (c) Der Prozessor wird durch einen leistungsfähigeren Prozessor mit 0.4 ns Taktzykluszeit und einem CPI-Wert von 1.8 ersetzt. Wie lang ist nun die Ausführungszeit des Programms?

   Lösung:

   $$\text{Taktzyklen: } z = 150\,000 \text{ Befehl} \cdot 1.8 \frac{\text{Taktzyklus}}{\text{Befehl}} = 270\,000 \text{ Taktzyklus}$$

   $$\text{Zeit: } t = 270\,000 \text{ Taktzyklus} \cdot 0.4 \cdot 10^{-9} \frac{\text{s}}{\text{Taktzyklus}} = 0.000108 \text{ s} = 108 \ \mu\text{s}$$

   Die Ausführungszeit beträgt 108 μs

   (d) Der Prozessor (von c) wird um 10% übertaktet („overclocking“). Die erzielte Leistungssteigerung beträgt in der Realität aber nur knapp 5%. Wieso?

   Lösung:
   Die Rechengeschwindigkeit hängt nicht allein von der Taktrate ab. Insbesondere die Datenübertragungsbusse spielen eine wichtige Rolle. Sie werden aber nicht mit-übertaktet.

2. Gegeben sei ein einfacher Prozessor ohne Pipelining mit einer Wortbreite von 2 Byte (für Daten und Befehle).

   (a) Welchen Wert beinhaltet der Befehlszähler jeweils nach Ausführung der jeweiligen Befehle der folgenden Befehlssequenz (der Initialwert sei 24 048 für den ersten Befehl): Ladebefehl, Ladebefehl, Addition, unbedingter Sprung um -12, Speicherbefehl, unbedingter Sprung um +8, Addition . . . ?

Lösung:

Befehl	**Befehlszähler**	**Kommentar**
Ladebefehl	24 048	Initialwert
Ladebefehl	24 050	+2
Additionsbefehl	24 052	+2
Sprungbefehl	24 054	+2
Sprung	24 042	−12
Speicherbefehl	24 042	+2
Sprungbefehl	24 044	+2
Sprung	24 052	+8
Additionsbefehl	24 052	+2
. . .	24 054	+2

   (b) Was sehen Sie als Informatiker sofort?

Lösung:
An Stelle 24 054 steht wieder der unbedingte erste Sprungbefehl (zurück auf Feld 24 042). Das Programm wird also über diese Zeile nicht hinaus kommen. Es ist in einer Endlosschleife gefangen.

3. Gegeben sei ein Prozessor mit 4-stufiger Pipeline (die vier Stufen, wie in der Vorlesung angegeben) und folgender Ausschnitt einer Programmabfolge:

   ..., Load, Sprung, Addition, ODER-Operation, Store, Subtraktion, Sprung, AND-Operation, ...

   (a) Skizzieren Sie graphisch eine (mögliche) Ausführungsabfolge, unter der Annahme, das beim 1. Sprung zu einer nicht vorhergesehenen Adresse gesprungen wird („branch prediction“ war falsch).

Lösung:

Op. sei Operation

Zyklus	**Pipeline**			
	Stufe 1	**Stufe 2**	**Stufe 3**	**Stufe 4**
1	Load			
2	Sprung	Load		
3	Addition	Sprung	Load	
4	ODER-Op.	Addition	Sprung	Load
5	Store	ODER-Op.	Addition	Sprung
		Pipeline Flush		
6	Addition			
7	ODER-Op.	Addition		
8	Store	ODER-Op.	Addition	
9	Subtraktion	Store	ODER-Op.	Addition
10	Sprung	Subtraktion	Store	ODER-Op.
11	AND-Op.	Sprung	Subtraktion	Store
12		AND-Op.	Sprung	Subtraktion
13			AND-Op.	Sprung
14				AND-Op.

   (b) Beschreiben Sie in Ihren Worten, was ein „pipeline flush“ bedeutet.

Lösung:

Bei einem Pipeline Flush werden Ergebnisse von abgearbeiteten Teilschritten in der Pipeline verworfen, die Pipeline wird geleert und nach dem letzten gültigen Befehl wieder gefüllt.

4. Eine effektive Möglichkeit der Leistungssteigerung bei Prozessoren ist Pipelining.

   (a) Begründen Sie, warum eine n-stufige Pipeline nicht automatisch zu einer n-fachen Leistungssteigerung führt, selbst wenn es gelingt, die Zykluszeit auf $1/n$ zu reduzieren („perfekte Gleichverteilung“ der Stufen – in der Praxis eigentlich nicht realisierbar).

   Lösung:

   Eine Pipeline würde nur zu einer n-fachen Leistungsteigerung führen, wenn in jedem Teilschritt der Pipeline zu jeder Zeit eine Operation der Befehlskette gemacht werden kann und die Pipeline nie geleert wird.

   In der Praxis kommt es aber einerseits zu Löchern in der Pipeline, wenn eine bestimmte Befehlsausführung länger dauert als normal und anderseits zum Leeren der Pipeline („Pipeline Flush“) durch Konflikte, insbesondere Kontrollflusskonflikte. In diesem Fall reduziert sich der Durchsatz der Pipeline.

5. Gegeben sei ein Prozessor ohne Pipeline mit der „bekannten“ Befehlsabarbeitung (siehe Vorlesung) und einer Zykluszeit von 20 MHz. Ein Analyse hat ergeben, dass die einzelnen Teilschritte sehr unterschiedliche Zeit erfordern:

   z. B. „Befehl laden“ $\leq$ 10 ns, „Register lesen“ $\leq$ 3 ns, „Rechenoperation durchführen“ $\leq$ 5 ns, „Speicherzugriff“ $\leq$ 20 ns und „Register schreiben“ $\leq$ 5 ns, ...

   Sie implementieren denselben Prozessor mit einer 5-stufigen Pipeline (die bisherigen Teilschritte erfordern gleich viel Zeit).

   (a) Wie gross ist die Zykluszeit des neuen Prozessors?

   Lösung:
   100 MHz, pro Zyklus wird aber nur ein Teilschritt des Befehls abgerbeitet.

   (b) Um wie viel schneller wird nun ein Befehl maximal ausgeführt?

   Lösung:
   Ein Befehl wird gleich schnell ausgeführt.

   (c) Um wie viel schneller wird ein Programm maximal ausgeführt?

   Lösung:
   Wenn ein Programm n Befehle hat, ist das Programm auf der CPU mit der 5-stufigen Pipeline maximal $\frac{5+n-1}{5 \cdot n}$ mal schneller als auf der CPU ohne Pipeline.

   (d) Wie könnte eine „bessere“ Pipeline-Struktur entwickelt werden?

   Lösung:
   Lange Pipelines sind einerseits komplexer im Aufbau, anderseits anfällig für Pipeline Konflikte (z. B. Control Hazards). In einem solchen Fall muss die Pipeline geleert werden, was die Ausführung verlangsamt.

   Dem kann man einerseits mit besseren Sprungvorhersagen und anderseits kürzeren Pipelines begegnen.

   Weniger und ähnlich lange Befehle sind eine weitere Optimierung, wie sie in RISC CPUs gemacht werden.