Karlsruher Institut für Technologie
Institut für Technische Informatik
Lehrstuhl für Rechnerarchitektur und Parallelverarbeitung
Prof. Dr. rer. nat. Wolfgang Karl

Klausur Rechnerstrukturen Wintersemester 2016/17 Musterlösung

Voraussichtliche Bekanntgabe der vorläufigen Ergebnisse:
April 2017

Aufgabe 1: Sprungvorhersage und Verbindungsstrukturen — 10 P

Sprungvorhersage — 4 P

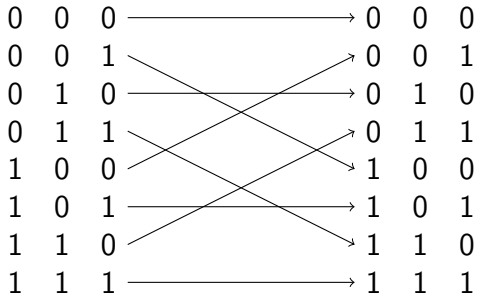
a) Der Vorteil ist, dass das Flattern zwischen den Weakly-Zuständen vermieden wird. *1 P*

b) *3 P*

	Globaler Prädiktor	S1		Globaler Prädiktor	S2	
		Vhs.	Sprung		Vhs.	Sprung
1	(**WT**, WT)	**T**	T	(ST, **WT**)	T	NT
2	(**ST**, SNT)	T	NT	(**WT**, SNT)	**T**	T
3	(ST, **SNT**)	NT	T	(ST, **WNT**)	**NT**	NT
4	(**ST**, SNT)	**T**	T	(ST, **SNT**)	NT	T
5	(ST, **WNT**)	NT	T	(ST, **ST**)	T	NT
6	(**ST**, WT)	T	NT	(**WT**, WT)	**T**	T

Verbindungsstrukturen — 6 P

c) Kreuzpermutation: *1 P*



d) Das Switching-Banyan-Netzwerk verwendet die Kreuzpermutation als Grundlage. *1 P*

e) Die Umkehrpermutation ist für $n = 2$ und $n = 3$ Bits identisch mit der Kreuzpermutation. *0,5 P*

f) Die Steuerung des Verbindungsaufbaus ist bei dynamischen Verbindungsstrukturen im Schaltnetz konzentriert. *0,5 P*

g) Phit ist die Abkürzung für physical transfer unit. Ein Phit ist die Informationseinheit, die in einem Zyklus auf einem Link übertragen wird. *1 P*

h) *2 P*

- Sender overhead
- Time of flight
- Transmission time
- Routing time
- Receiver overhead

Aufgabe 2: Vektorrechner und Tomasulo 10 P

Vektorrechner 5 P

a) Unter einem Vektorrechner versteht man einen Rechner mit pipelineartig aufgebautem/n Rechenwerk/en zur Verarbeitung von Arrays von Gleitpunktzahlen. *1 P*

b) Zur Umsetzung von IF-Anweisungen wird das Vektor-Mask-Register eingesetzt, das ermöglicht nur auf den auf 1 gesetzten Einträgen zu arbeiten. *1 P*

c) Der erste Zugriff hat eine Latenz von 12 Zyklen. Nach insgesamt 8 Zugriffen erfolgt wieder ein Zugriff auf die erste Speicherbank (mit einem Stride von 10 erfolgt der 9. Zugriff auf Adresse 80. $80 \mod 16 = 0$, somit Zugriff auf die 0-te Speicherbank), auf welche bereits zugegriffen wurde, was somit eine Wartezeit von $12 - 7 = 5$ Zyklen erzeugt. Dieser Ablauf wiederholt sich für die komplette Zugriffsfolge. Die Latenz der restlichen Zugriffe kann versteckt werden. Somit ergibt sich insgesamt:
$12\ Zyklen + 7\ Zyklen + 7 \cdot (5\ Zyklen + 7\ Zyklen) = 103\ Zyklen$ *2 P*

d) Die Ergebnisse einer Pipeline werden sofort der nächsten Pipeline zur Verfügung gestellt, wodurch die Gesamtausführungszeit der Vektoroperationen reduziert wird. *1 P*

Tomasulo 5 P

e) *5 P*

Feld	R1	R2	R3	R4	R5
Value				(R1+R5)	
Valid	0	0	0	1	0
RS	Div 1	Add/Sub 2	Mul 1		Add/Sub 1

Unit	Empty	InFU	Op	Dest	Src1	Vld1	RS1	Src2	Vld2	RS2
Add/Sub 1	0	0	add	R5		0	Mul 1	(R5)	1	
Add/Sub 2	0	0	add	R2		0	Mul 1	(R1 + R5)	1	
Mul 1	0	1	mul	R3	(R2)	1		(R4)	1	
Div 1	0	1	div	R1	(R1 + R5)	1		(R5)	1	

Aufgabe 3: Caches **10 P**

a) *3 P*

Private Cache-Organisation: Missrate $r_{Miss-L1} = 100\%$, da jeder Zugriff ein Miss ist aufgrund der begrenzten Kapazität und somit $r_{Hit-L1} = 0\%$. $t_{L1} = 1$ Zyklus und $t_{mem} = 100$ Zyklen.

$t_{pp} = r_{Hit-L1} \cdot t_{L1} + r_{Miss-L1} \cdot (t_{L1} + t_{mem}) = 1\ Zyklus + 100\ Zyklen = 101\ Zyklen$

Geteilte Cache-Organisation: Missrate $r_{Miss-L1} = 50\%$, da beim 2. Zugriff die Daten noch im Cache vorhanden sind. $t_{L1} = 2\ Zyklen$ und $t_{mem} = 100\ Zyklen$.

$t_{ps} = r_{Hit-L1} \cdot t_{L1} + r_{Miss-L1} \cdot (t_{L1} + t_{mem}) = 0.5 \cdot 2\ Zyklen + 0.5 \cdot 102\ Zyklen = 52\ Zyklen$

Faktor: Die Zugriffszeit der geteilten Cache-Organisation ist um den Faktor 1.94 besser.

b) *2 P*

Private Cache-Organisation: Gleiches Ergebnis wie zuvor, $t_{pp} = 101\ Zyklen$.

Geteilte Cache-Organisation: Der jeweils erste Zugriff auf einen Block ist ein Miss. Angenommen Prozessor 1 verursacht diesen jedes mal, dann ist seine mittlere Zugriffszeit wie zuvor $t_{ps_1} = 52\ Zyklen$. Für die anderen gilt $r_{Miss-L1} = 0\%$ und somit

$t_{ps_i} = 1 \cdot 2\ Zyklen + 0 \cdot 102\ Zyklen = 2\ Zyklen,\ i = 2, 3, 4.$

Insgesamt ist die mittlere Zugriffszeit

$t_{ps} = (52\ Zyklen + 3 \cdot 2\ Zyklen)/4 = 14.5\ Zyklen.$

Faktor: Die Zugriffszeit der geteilten Cache-Organisation ist um den Faktor 7.25 besser.

c) *2 P*

Private Cache-Organisation: Der Schreibzugriff führt zu einer Invalidierung mittels des Cache-Protokolls und der folgende Lesezugriff führt zu einem Kohärenzmiss. Beide Aktionen kosten jeweils 10 Zyklen. Somit ist die mittlere Zugriffszeit $t_{pp} = 1\ Zyklus + 10\ Zyklen = 11\ Zyklen$.

Geteilte Cache-Organisation: Aufgrund des geteilten Caches werden keine Cache-Kohärenzzugriffe benötigt und daher ist die mittlere Zugriffszeit

$t_{ps} = 100\% \cdot 2\ Zyklen + 0 = 2\ Zyklen.$

Die Zugriffszeit der geteilten Cache-Organisation ist um den Faktor 5.5 besser.

d) *3 P*

Proz.	Aktion	Proz. 1		Proz. 2		Proz. 3	
		Zeile 1	Zeile 2	Zeile 1	Zeile 2	Zeile 1	Zeile 2
-	init	-	-	-	-	-	-
2	rd 4			4/E			
1	rd 4	4/S		4/S			
3	wr 4	4/I		4/I		4/M	
2	rd 4			4/S		4/S	
1	rd 1	1/E					
1	wr 2		2/M				
2	wr 2		2/I		2/M		
2	rd 3			3/E			

Aufgabe 4: Hardware-Entwurf und VHDL 8 P

Hardware-Entwurf 6 P

a) Formel für Amdahls Gesetz: *3 P*

$$T(n) = \underbrace{\frac{1}{n} \cdot T(1) \cdot (1-a)}_{\text{parallel ausführbarer Programmteil}} + \underbrace{T(1) \cdot a}_{\text{sequentieller Programmteil}}$$

mit $a =$ der ausschließlich sequentiell ausführbare Programmanteil und $n =$ der Beschleunigungsfaktor (beispielsweise die Anzahl der Prozessoren).

Daraus lässt sich die Beschleunigung folgendermaßen berechnen:

$$\frac{T(1)}{T(n)} = \frac{T(1)}{\frac{1}{n} \cdot T(1) \cdot (1-a) + T(1) \cdot a} = \frac{1}{a + \frac{1-a}{n}}$$

Mit $n = 10$ und $a = 1 - F_{fp}$ bzw. $n = 2$ und $a = 1 - F_{ls}$ lässt sich somit das Verhältnis folgendermaßen angeben:

$$\frac{1}{1 - F_{fp} + \frac{F_{fp}}{10}} > \frac{1}{1 - F_{ls} + \frac{F_{ls}}{2}}$$

$$1 - F_{ls} + \frac{F_{ls}}{2} > 1 - F_{fp} + \frac{F_{fp}}{10}$$

$$-F_{ls} + \frac{F_{ls}}{2} > -F_{fp} + \frac{F_{fp}}{10}$$

$$F_{ls} \cdot (-1 + \frac{1}{2}) > F_{fp} \cdot (-1 + \frac{1}{10})$$

$$(-1 + \frac{1}{2}) > \frac{F_{fp}}{F_{ls}} \cdot (-1 + \frac{1}{10})$$

$$\frac{F_{fp}}{F_{ls}} > \frac{\frac{1}{2}}{\frac{9}{10}} = \frac{5}{9}$$

b) *3 P*

i) Ja, da $\frac{F_{fp}}{F_{ls}}$ dem Verhältnis zwischen der Ausführungszeit der Fließkommaoperationen $T_{exe,fp}$ und der Ausführungszeit der Speicherbefehle $T_{exe,ls}$ entspricht. Dieses Verhältnis kann mit den angebenden Werten ermittelt werden, da $T_{exe} = IC \cdot CPI \cdot T_c$ entspricht. Somit ist

$$\frac{F_{fp}}{F_{ls}} = \frac{T_{exe,fp}}{T_{exe,ls}} = \frac{IC_{fp} \cdot CPI_{fp} \cdot T_c}{IC_{ls} \cdot CPI_{ls} \cdot T_c}$$

ii) Nein, da die gesamte Ausführungszeit oder der durchschnittliche CPI-Wert aller Instruktionen nicht gegeben ist und wir somit auch nicht die Anteile der Ausführungszeit F_{fp} oder F_{ls} bekommen können.

VHDL **2 P**

c) **Schritte:** *2 P*

1. High-Level Synthese
2. Logik-Synthese
3. Layout-Synthese
4. Fertigung

Aufgabe 5: Low-Power-Entwurf und Leistungsbewertung — 10 P

Low-Power-Entwurf — 2 P

a) *2 P*

- Schaltwahrscheinlichkeit: $\mathbb{P}_{schalt} = 2 * \mathbb{P}(1) * (1 - \mathbb{P}(1))$ (0,5 Punkte)
- Leistungsaufnahme: $P_{total} = P_{switching} + P_{shortcircuit} + P_{static} + P_{leakage}$ (0,5 Punkte)
- Bestandteil: $P_{leakage}$ (0,5 Punkte)
- Begründung: Zunehmende Integrationsdichte (0,5 Punkte)

Leistungsbewertung — 8 P

b) Berechnung: *2 P*
Sei I die Anzahl der Programmbefehle. Dann:

$$\text{Zyklen}_A = I \cdot 2,0$$
$$\text{Zyklen}_B = I \cdot 1,2 \qquad (0,5 \text{ Punkte})$$

$$\text{Zeit}_A = I \cdot 2,0 \cdot 250ps = 500\ I\ ps$$
$$\text{Zeit}_B = I \cdot 1,2 \cdot 500ps = 600\ I\ ps \qquad (0,5 \text{ Punkte})$$

$$\frac{\text{Zeit}_B}{\text{Zeit}_A} = \frac{600\ I\ ps}{500\ I\ ps} = 1,2 \qquad (0,5 \text{ Punkte})$$

Somit ist Computer A $1,2$ mal schneller als Computer B. (0,5 Punkte)

c) Antwort: *3 P*

(c.1) Falsch (0,5 Punkte), Begründung: Leistungsbewertung anhand von Taktfrequenzen ist nicht aussagekräftig. (0,5 Punkte)

(c.2) Falsch (0,5 Punkte), Begründung: Leistungsbewertung anhand von Taktfrequenzen ist auch bei ähnlichen Architekturen und gleichen Compilern nicht aussagekräftig. (0,5 Punkte)

(c.3) Wahr (0,5 Punkte), Begründung: Leistungsbewertung kann nur anhand geeigneter Benchmarks erfolgen. Diese Benchmarks müssen dem späteren Einsatzzweck möglichst nahe kommen um ein aussagekräftiges Ergebnis zu erhalten. (0,5 Punkte)

d) Berechnung: *3 P*

$$\text{Zyklen} = \sum_{i=1}^{n} (\text{CPI}_i \cdot C_i) \qquad (0,5 \text{ Punkte})$$

$$\begin{aligned} \text{Zyklen}_1 &= (5 \cdot 1 + 1 \cdot 2 + 1 \cdot 3) \cdot 10^9 = 10 \cdot 10^9 \\ \text{Zyklen}_2 &= (10 \cdot 1 + 1 \cdot 2 + 1 \cdot 3) \cdot 10^9 = 15 \cdot 10^9 \qquad (0,5 \text{ Punkte}) \end{aligned}$$

$$\begin{aligned} \text{Zeit}_1 &= \frac{10 \cdot 10^9}{4 \cdot 10^9} = 2,5s \\ \text{Zeit}_2 &= \frac{15 \cdot 10^9}{4 \cdot 10^9} = 3,75s \qquad (0,5 \text{ Punkte}) \end{aligned}$$

$$\text{MIPS} = \frac{\text{Befehlszahl}}{\text{Zeit} \cdot 10^6} \qquad (0,5 \text{ Punkte})$$

$$\begin{aligned} \text{MIPS}_1 &= \frac{(5 + 1 + 1) \cdot 10^9}{2,5 \cdot 10^6} = 2800 \\ \text{MIPS}_2 &= \frac{(10 + 1 + 1) \cdot 10^9}{3,75 \cdot 10^6} = 3200 \qquad (0,5 \text{ Punkte}) \end{aligned}$$

Das Programm weist mit Compiler 2 also einen höheren MIPS-Wert auf, mit Compiler 1 wird es jedoch schneller ausgeführt. (0,5 Punkte)

Aufgabe 6: Parallelverarbeitung und Pipelining 12 P

Parallelverarbeitung 4 P

a) Ebenen der Parallelität: *4 P*

- Programmebene (0,5 Punkte): Parallele Verarbeitung verschiedener Programme als vollständig unabhängige Einheiten, beispielsweise auf Betriebssystemebene. (0,5 Punkte)
- Prozess- oder Task-Ebene (0,5 Punkte): Parallele Verarbeitung von Prozessen eines Programms, die typischerweise jeweils aus vielen sequentiellen Befehlen mit eigenem Datenbereich bestehen. (0,5 Punkte)
- Blockebene (0,5 Punkte): Parallele Verarbeitung von Blöcken aus sequentiellen Befehlen, die sich typischerweise einen Adressraum teilen. (0,5 Punkte)
- Anweisungs- oder Befehlsebene (0,5 Punkte): Parallele Ausführung einzelner Maschinenbefehle oder elementarer Anweisungen.(0,5 Punkte)
- Alternative: Suboperationsebene (0,5 Punkte): Parallele Verarbeitung von Suboperationen elementarer Anweisungen, beispielsweise in Vektorrechnern. (0,5 Punkte)

Pipelining 8 P

b) Konflikte: *6 P*

- Strukturkonflikte (tritt nicht auf) (1 Punkt)
- Datenkonflikte (tritt auf, jeweils zwischen den *add*-Befehlen und dem zugehörigen unmittelbar davor stehenden *lw*-Befehl) (1 Punkt)
- Steuerkonflikte (tritt nicht auf) (1 Punkt)

Umsortiertes Codesegment: (1 Punkt)

```
1        lw   $t1, 0($t0)
2        lw   $t2, 4($t0)
3        lw   $t4, 8($t0)
4        add  $t3, $t1, $t2
5        sw   $t3, 12($t0)
6        add  $t5, $t1, $t4
7        sw   $t5, 16($t0)
```

Weitere Lösungen:

- Pipeline Stall (0,5 Punkte): Anhalten der Pipeline bis der Operand ausgeführt werden kann. (0,5 Punkte)
- Pipeline Bubble (0,5 Punkte): Einfügen von Lehrzyklen in die Pipeline. (0,5 Punkte)
- Alternativ: Pipeline Flush (0,5 Punkte): Leeren und neu befüllen der Pipeline. (0,5 Punkte)

c) Weitere Eigenschaften: *2 P*

- Load-Forwarding (0,5 Punkte): Daten können aus dem Memory oder Cache direkt einer ALU zur Verfügung gestellt werden, ohne vorher in ein Register geschrieben zu werden. (0,5 Punkte)
- Result-Forwarding (0,5 Punkte): Das Ergebnis des Operanden kann anderen Operanden zur Verfügung gestellt werden noch bevor es in das Register zurückgeschrieben ist. (0,5 Punkte)