Karlsruher Institut für Technologie
Institut für Technische Informatik
Prof. Dr. Wolfgang Karl

Klausur Rechnerstrukturen
Wintersemester 2013/14 – 26. Feb. 2014
Musterlösung

Aushang der Ergebnisse: vorauss. Mitte/Ende März 2014

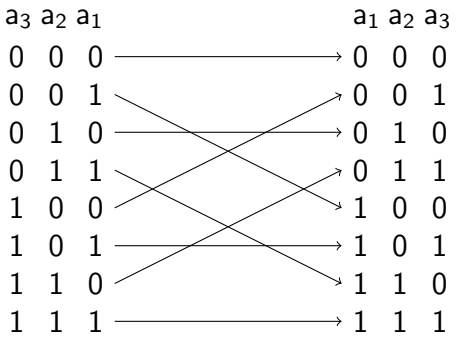
Musterlösung 1: Verbindungsstrukturen & Vektorverarbeitung **10P**

Verbindungsstrukturen **5P**

a) Mögliche Antworten: *1P*

- Fähigkeit, die wesentlichen Eigenschaften des Verbindungsnetzes auch bei beliebiger Erhöhung der Knotenzahl beizubehalten.
- Vergrößerung möglich ohne die wesentlichen Eigenschaften des Netzwerks zu verlieren.

b) Kreuzpermutation: *1P*



c) *1P*

- Durchmesser: 6
- Minimale Bisektionsbreite: 4

d) Bei einem fehlertoleranten Netz muss zwischen jedem Paar von Knoten (0,5P) mindestens ein weiterer, redundanter Weg vorhanden sein (0,5P) *1P*

e) *1P*

- Anzahl der Knoten $N = K^n$
- Verbindungsgrad: 2n

Vektorverarbeitung: **5P**

f) *4P*

```
MOV    R1, 64       # R1 mit 64 initialisieren
MTC1   VLR, R1      # vector-length register := 64
LV     V1, Ra       # a[64] in V1 laden
LV     V2, Rb       # b[64] in V2 laden
MOV    R2, 0        # R2 mit 0 initialisieren
ADDVS  V3, V1, R2   # Kopiere V1 in V3 (V3 = V1 + 0)
MOV    R1, 0xff     # R1 mit 0xff initialisieren
SEQVS  V1, R1       # Vergleich mittels 'equals':
                    # if V1[i] == 0xff: VMR[i] = 1 else VMR[i] = 0
SUBV   V3, V3, V3   # if VMR[i] == 1: V3[i] = V3[i] - V3[i] = 0
ADDV   V3, V3, V2   # if VMR[i] == 1: V3[i] = V3[i] + V2[i] = V2[i]
CVM                 # Clear Vektor Mask; for each i: VMR[i] = 1
SV     Rc, V3       # V3 in c[64] speichern
```

g) *1P*

- Stride von 1: Das nächste Element ist immer in der nächsten Speicherbank. Da die Latenz kleiner als die Anzahl an Speicherbänken ist, kann die Zugriffslatenz für die ersten 31 Elemente maskiert werden und nur für das letzten Element muss die gesamte Latenz abgewartet werden. Daher benötigt man $31 + 5 = 36$ Zyklen.
- Stride von 8: Da der Stride gleich der Anzahl der Bänke ist, handelt es sich hier um den schlechtesten Fall. Jeder Speicherzugriff geht auf die gleiche Speicherbank und kollidiert mit dem vorhergehenden. Damit verursacht jeder Speicherzugriff die volle Latenz von 5 Takten und man benötigt insgesamt $5 * 32 = 160$ Takte.

Musterlösung 2: Low-Power-Entwurf & Rechnerbewertung **8P**

Low-Power-Entwurf **3,5P**

a) $S(f_1)$ ist größer als $S(f_2)$ *1P*

$D(f_1)$ ist kleiner als $D(f_2)$

b) *2,5P*

- Statisch: $P_{static}, P_{leakage}$
- Dynamisch: $P_{switching}, P_{shortcircuit}$
- Antwort: Aufgrund der Miniaturisierung haben Leckströme $P_{leakage}$ einen wesentlichen Einfluss.

Leistungsbewertung **4,5P**

c) SPECint ($\frac{1}{2}$P): Es handelt sich hierbei um eine per Benchmark (genauer: SPEC ist eine Sammlung von Benchmark-Programmen) ermittelte Leistungsquantifizierung ($\frac{1}{2}$P), die durch den Bezug auf eine Referenzmaschine einen fairen Vergleich zwischen Systemen erlaubt. **1P**

d) $MIPS = \frac{f}{CPI*10^6}$ **0,5P**

e) Zugriffszeit t_z und Übertragungszeit $t_{ü}$ *($\frac{1}{2}$P für vollständige Bezeichnungen).* **1,5P**

Hieraus leiten sich Bedienzeit X_i und maximale Auslastung D_{imax} wie folgt ab:

($\frac{1}{2}$P) $X_i = t_z + t_{ü}$

($\frac{1}{2}$P) $D_{imax} = \frac{1}{X_i} = \frac{1}{t_z + t_{ü}}$

f) Prozessor 2($\frac{1}{2}$P): Der niedrigere MIPS-Wert resultiert in kompakterem Code (weniger Speicherverbrauch, $\frac{1}{2}$P) sowie niedrigere Schaltfrequenzen und damit niedrigeren Energieverbrauch ($\frac{1}{2}$P). **1,5P**

Musterlösung 3: Prozessorarchitektur 11P

Sprungvorhersage 8P

a) *4P*

Befehl	Inhalt r0	Inhalt r1	Prädiktor 1 alt	Prädiktor 1 neu	Prädiktor 1 Vorhersage	Prädiktor 2 alt	Prädiktor 2 neu	Prädiktor 2 Vorhersage
Init	1	3	-	T	-	-	T	-
BRNEQ loop2	1	2	-	-	-	T	T	richtig
BRNEQ loop2	1	1	-	-	-	T	T	richtig
BRNEQ loop2	1	0	-	-	-	T	NT	falsch
BREQ loop1	0	0	T	T	richtig	-	-	-
BRNEQ loop2	0	2	-	-	-	NT	T	falsch
BRNEQ loop2	0	1	-	-	-	T	T	richtig
BRNEQ loop2	0	0	-	-	-	T	NT	falsch
BREQ loop1	-1	0	T	NT	falsch	-	-	-

b) Nein, da mit der Initialisierung auf NT jeweils der erste Sprung falsch vorhergesagt wird. *1P*

c) Ein (m,n)-Korrelationsprädiktor nutzt das Verhalten der letzten m Sprünge für die Auswahl aus 2^m Prädiktoren. Jeder Prädiktor stellt einen n-Bit Prädiktor für den jeweils einzelnen Sprung dar. Die globale Vergangenheit der letzten m Sprünge kann in einem m-Bit Schieberegister, dem Sprungverlaufsregister/Branch History Register (BHR), gespeichert werden. Der Inhalt des BHR wird als Adresse benutzt, um eine Sprungsverlaufstabelle (Pattern History Table, PHT) zu selektieren. Hierdurch kann die Sprunghistorie, d.h. Sprungmuster erkannt, werden. *3P*

Für $m = 0$ verhält sich ein Korrelationsprädiktor wie der einfache n-Bit Prädiktor.

Pipelining 3P

d) Formeln: *1,5P*

- $T_{ohnePipeline}(n,k) = n * k$
- $T_{mitPipeline}(n,k) = n + k - 1$
- $S(n,k) = \frac{T_{ohnePipeline}(n,k)}{T_{mitPipeline}(n,k)} = \frac{n*k}{n+k-1}$

e) 5-stufige Pipeline mit Konfliktbehandlung: *1,5P*

- Ausführungszeit (*1P*):
  $T = 12000 + 5 - 1 + \frac{12000}{4} + 2 * \frac{12000}{10} = 12004 + 3000 + 2400 = 17404$
- Speedup ($\frac{1}{2}$P): $S(n,k) = \frac{T_{ohnePipeline}(n,k)}{T_{mitPipeline}(n,k)} = \frac{12000*5}{17404} = \frac{60000}{17404} \approx 3,45 < 4$

Musterlösung 4: Speicherhierarchie **10P**

Cache-Kohärenzprotokoll MESI **6P**

a) *4P*

Zeile	Prozessor	Aktion	Prozessor 1		Prozessor 2		Prozessor 3	
			Line 1	Line 2	Line 1	Line 2	Line 1	Line 2
0.		init	-	-	-	-	-	-
1.	1	wr 1	1/M					
2.	3	rd 2					2/E	
3.	2	wr 1	1/I		1/M			
4.	1	rd 5	5/E					
5.	2	rd 5	5/S			5/S		
6.	3	rd 1			1/S			1/S
7.	2	rd 2				2/S	2/S	
8.	3	wr 4						4/M
9.	2	rd 1			1/S			
10.	3	rd 1					1/S	

b) Das MOESI-Protokoll würde zu einer beschleunigten Abarbeitung führen: in Zeile 6 werden würden die Daten direkt durch einen Cache-zu-Cache-Transfer übertragen. Dies würde zwei Speicherzugriffe sparen (1 lesend/1 schreibend). *1P*

c) In diesem Beispiel gleichen sich die Vorteile durch den Einfluss der LRU-Strategie auf MESI-Zustandsänderungen aus: während die Ersetzung in Zeile 7 von Prozessor 2 einen Hit in Zeile 9 bedingt, verhindert die Ersetzung der Zeile 8 in Prozessor 3 einen Hit in Zeile 10. Somit wiegen sich positive wie negative Effekte dieser Berücksichtigung auf. *1P*

Cache-Leistung **4P**

d) Alternative A: $0,8 * 4\,ns + 0,2 * (0,8 * 20\,ns + 0,2 * 100\,ns) = 3,2\,ns + 0,2 * (16\,ns + 20\,ns) = 3,2\,ns + 0,2 * 36\,ns = 3,2\,ns + 7,2\,ns = 10,4\,ns$, *3P*

Alternative B: $0,8 * 6\,ns + 0,2 * (0,75 * 25\,ns + 0,25 * 125\,ns) = 4,8\,ns + 0,2 * (\frac{3*25}{4}\,ns + \frac{125}{4}\,ns) = 4,8\,ns + 0,2 * \frac{200}{4}\,ns = 4,8\,ns + 10\,ns = 14,8\,ns$

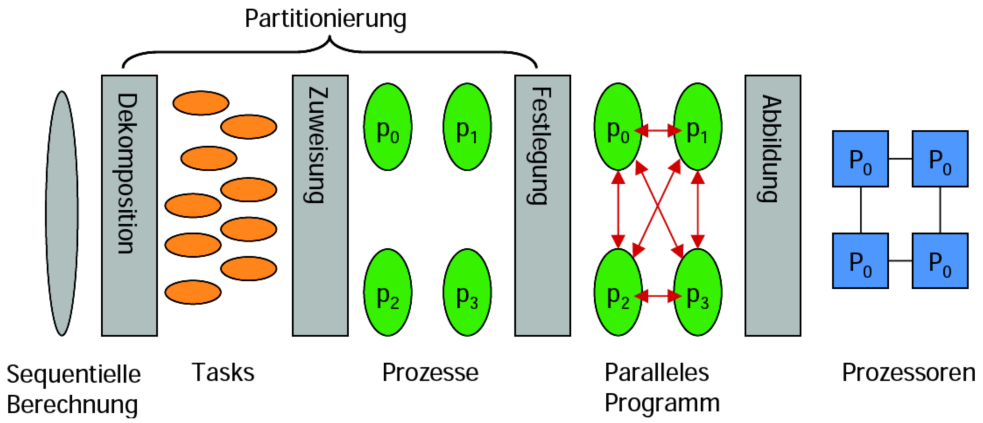
Damit erreicht Alternative B die langsamere Antwortzeit, weshalb ein Wechsel der Cachehierarchie nicht zu empfehlen ist, da dies eine Verschlechterung bedeuten würde.

e) Für eine Multiprozessorsystem mit gemeinsamen Speicher lohnt es sich diese Bewertung zu überdenken, da bei der Variante A im Fall eines Cache-Hits in den höheren Ebenen Anfragen an den Hauptspeicher gestartet werden, die dann später wieder abgebrochen werden. Diese blockieren den Bus und verhindern so, dass notwendige Daten zu anderen Prozessoren transferiert werden. Somit kann man argumentieren, dass die längere Antwortzeit der Variante B toleriert werden kann, da hier keine unnötigen Anfragen an den Hauptspeicher auftreten. *1P*

Musterlösung 5: Parallele Architekturen und Hardwareentwurf 11P

Parallele Architekturen 6P

a) *2P*



b) Das Problem selbst muss skalieren (Problemgröße, Parallelisierbarkeit). *1P*

c) *3P*

	Gemeinsamer Speicher	Physikalisch verteilter Speicher
Globaler Adressraum	SMP Symmetrischer Multiprozessor UMA: Uniform Memory Access	DSM Distributed-shared-memory Multiprozessor NUMA: Non-Uniform Memory Access
Getrennte Adressräume	leer	Nachrichtengekoppelter (Shared-nothing) Multiprozessor / Cluster NORMA: No Remote Memory Access

Hardware-Entwurf **5P**

d) *2P*

Vorteile:

- Flexibilität
- Weniger Fehleranfällig

Nachteile:

- Randbedingungen können nur schwer eingehalten werden
- Syntheseergebnisse oft schlechter als manueller Entwurf

Hinweis: In der Vorlesung wurden noch weitere Vor- und Nachteile genannt.

e) *3P*

Syntheseschritte:

- High-Level-Synthese
- Logik-Synthese
- Layout-Synthese

Beschreibungen:

- RT-Beschreibung
- Gatter-Beschreibung
- Geometrie-Beschreibung

Musterlösung 6: Parallelismus auf Befehlsebene 10P

Parallelismus auf Befehlsebene **6P**

a) Dynamische Parallelisierung mittels Hardware (Superskalartechnik) vs. statische Parallelisierung durch den Compiler (VLIW-Technik). *1P*

b) *5P*

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

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

c) VLIW-Prozessoren *4P*

Slot 1	Slot 2
3) ld r4, [r1]	4) ld r5, [r2]
1) fpdiv f3, f1, f2	5) add r7, r4, r5
6) ld r6, [r7]	2) fpadd f5, f3, f1
7) sub r8, r6, r7	8) st [r5], r7

System A

Integer	Gleitkomma	Load/Store
	1) fpdiv f3, f1, f2	3) ld r4, [r1]
	2) fpadd f5, f3, f1	4) ld r5, [r2]
5) add r7, r4, r5		
		6) ld r6, [r7]
7) sub r8, r6, r7		8) st [r5], r7

System B

Entscheidung: Für System A, da das Programm trotz weniger Funktionseinheiten schneller abgearbeitet wird.