Übungsklausur Rechnerarchitekturen

Klausurvariante 3

July 10, 2025

Prüfungsfach:	Rechnerarchitekturen
Semester:	Platzhalter-Semester
Prüfungsdauer:	90 Minuten
Maximale Punktzahl:	90 Punkte
Erlaubte Hilfsmittel:	Taschenrechner

Name: .....

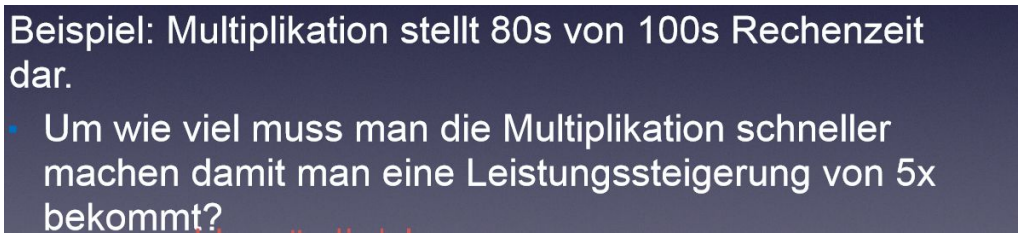
Matrikelnummer: .....

Unterschrift: .....

Viel Erfolg!

Aufgabe 1: Leistung, Compiler und CPI

a)



Aufgabe 2: MIPS

a)

Aufgabe 2.21

[5] <2.6> Geben Sie einen minimalen Satz von MIPS-Befehlen an, der verwendet werden kann, um den folgenden Pseudobefehl zu implementieren:

```
not $t1, $t2    // bitweises Invertieren
```

b)

Aufgabe 2.30

[5] <2.7> Schreiben Sie die Schleife aus Aufgabe 2.29 so um, dass die Anzahl der ausgeführten MIPS-Befehle reduziert wird.

c)

Aufgabe 2.46

Für einen gegebenen Prozessor sei der CPI-Wert von arithmetischen Befehlen 1, der CPI-Wert für Lade-/Speicherbefehle sei 10 und der CPI-Wert von Sprungbefehlen sei 3. Ein Programm habe die folgende Befehlsaufteilung: 500 Millionen arithmetische Befehle, 300 Millionen Lade-/Speicherbefehle, 100 Millionen Sprungbefehle.

2.46.1 [5] <2.19> Angenommen, es werden neue, leistungsfähigere arithmetische Befehle zum Befehlssatz hinzugefügt. Durch die Verwendung dieser leistungsfähigeren arithmetischen Befehle kann die Anzahl der arithmetischen Befehle, die zur Ausführung eines Programms nötig sind, im Mittel um 25 % reduziert werden. Die Kosten aufgrund der Erhöhung der Taktfrequenz steigen nur um 10 %. Ist das eine gute Wahl des Designs? Begründen Sie Ihre Antwort.

2.46.2 [5] <2.19> Angenommen, wir haben eine Möglichkeit gefunden, die Performanz der arithmetischen Befehle zu verdoppeln. Wie groß ist die Beschleunigung unserer Maschine insgesamt? Was ist, wenn wir eine Möglichkeit finden, die Performanz der arithmetischen Befehle zu verzehnfachen?

d)

Aufgabe 2.3

[5] <2.2, 2.3> Wie lautet der zu der folgenden C-Anweisung gehörende MIPS-Assemblercode? Nehmen Sie an, dass die Variablen `f`, `g`, `h`, `i` und `j` den Registern `$s0`, `$s1`, `$s2`, `$s3$` bzw. `$s4` zugewiesen sind. Nehmen Sie außerdem an, dass sich die Basisadressen der Felder `A` und `B` in den Registern `$s6` bzw. `$s7` befinden.

```
B[8] = A[i-j];
```

Aufgabe 3: Arithmetik

a)

Aufgabe 3.42

[10] <3.5> Was erhalten Sie, wenn Sie $-1/4$ viermal mit sich selbst addieren? Was ist $-1/4 \times 4$? Sind die Ergebnisse gleich? Wie sollten sie sein?

b)

Aufgabe 3.9

[10] <3.2> Nehmen Sie an, dass 151 und 214 vorzeichenbehaftete dezimale 8-Bit-Ganzzahlen sind, die im Zweierkomplement-Format gespeichert werden. Berechnen Sie 151 + 214 mittels Sättigungsarithmetik. Schreiben Sie das Ergebnis in Dezimaldarstellung.

Aufgabe 4: Pipelining

a)

Aufgabe 4.18

In dieser Aufgabe vergleichen wir die Leistung von Prozessoren mit Einfach- und Zweifachzuordnung, wobei wir die Möglichkeit von Programmtransformationen berücksichtigen, die die Ausführung mit Einfachzuordnung optimieren. Die Teilaufgaben beziehen sich auf die folgende C-Schleife:

```
for ( i = 0; i != j; i += 2)
  b[i] = a[i] - a[i+1];
```

Wenn Sie im Folgenden MIPS-Code schreiben, dann nehmen Sie an, dass Variablen wie in der Tabelle angegeben in Registern gehalten werden und dass alle Register außer jenen, die als frei gekennzeichnet sind, benutzt werden, um verschiedene Variablen zu halten, so dass sie nicht für irgendetwas anderes benutzt werden können.

j	j	a	b	c	frei
R5	R6	R1	R2	R3	R10, R11, R12

4.18.1 [10] <4.10> Übersetzen Sie den gegebenen C-Code in MIPS-Befehle. Ihre Übersetzung sollte direkt sein, d. h. ohne Umordnungsbefehle, mit denen eine bessere Performanz erreicht werden soll.

4.18.2 [10] <4.10> Angenommen, die Schleife wird nach Ausführen von nur zwei Iterationen beendet. Zeichnen Sie ein Pipeline-Diagramm für Ihren MIPS-Code aus Aufgabe 4.18.1, der auf einem Prozessor mit Zweifachzuordnung wie dem in Abbildung 4.58 gezeigten ausgeführt wird. Nehmen Sie an, dass der Prozessor eine perfekte Sprungvorhersage hat und zwei beliebige Befehle (auch nicht aufeinanderfolgende) im gleichen Takt holen kann.

4.18.3 [10] <4.10> Ordnen Sie den Code aus Aufgabe 4.18.1 um mit dem Ziel, auf einem Prozessor mit Zweifachzuordnung und statischem Scheduling (Abbildung 4.58) eine bessere Performanz zu erreichen.

4.18.4 [10] <4.10> Wiederholen Sie 4.18.2, jedoch diesmal mit Ihrem MIPS-Code aus 4.18.3.

4.18.5 [10] <4.10> Um wie viel schneller wird die Ausführung, wenn von einem Prozessor mit Einfachzuordnung zu einem Prozessor mit Zweifachzuordnung wie in Abbildung 4.58 übergegangen wird? Verwenden Sie für beide Prozessorvarianten Ihren Code aus Aufgabe 4.18.1 und nehmen Sie an, dass 1 000 000 Iterationen der Schleife ausgeführt werden. Nehmen Sie wie in Aufgabe 4.18.2 an, dass der Prozessor eine perfekte Sprungvorhersage hat und dass ein Prozessor mit Zweifachzuordnung innerhalb eines Taktes zwei beliebige Befehle holen kann.

4.18.6 [10] <4.10> Wiederholen Sie 4.18.5, nehmen Sie nun jedoch an, dass in dem Prozessor mit Zweifachzuordnung einer der in einem Takt auszuführenden Befehle von beliebigem Typ sein kann, während der andere kein Speicherbefehl sein darf.

Aufgabe 5: Caching

a)

Aufgabe 5.13

In dieser Aufgabe betrachten wir, wie sich die Ersetzungsstrategien auf die Fehlzugriffsrate auswirken. Gehen Sie von einem zweifach satzassoziativen Cache mit vier Blöcken aus. Für die Lösung der Teilaufgaben könnte es praktisch sein, eine Tabelle wie die folgende anzulegen, in der das Vorgehen für die Adressfolge 0, 1, 2, 3, 4 demonstriert wird.

Adresse des Speicherblocks	Treffer oder Fehlzugriffe	Entfernter Block	Inhalt der Cache-Blöcke nach dem Zugriff			
			Satz 0	Satz 0	Satz 1	Satz 1
0	Fehlzugriff		Mem[0]			
1	Fehlzugriff		Mem[0]		Mem[1]	
2	Fehlzugriff		Mem[0]	Mem[2]	Mem[1]	
3	Fehlzugriff		Mem[0]	Mem[2]	Mem[1]	Mem[3]
4	Fehlzugriff	0	Mem[4]	Mem[2]	Mem[1]	Mem[3]
...						

Betrachten Sie die Adressfolge `0, 2, 4, 8, 10, 12, 14, 16, 0`

5.13.1 [5] <5.4, 5.8> Wie viele Treffer erzeugt diese Adressfolge, wenn eine LRU-Ersetzungsstrategie vorausgesetzt wird?

5.13.2 [5] <5.4, 5.8> Wie viele Treffer erzeugt diese Adressfolge, wenn eine MRU-Ersetzungsstrategie (Most Recently Used) vorausgesetzt wird?

5.13.3 [5] <5.4, 5.8> Simulieren Sie eine zufällige Ersetzungsstrategie, indem Sie eine Münze werfen. Kopf bedeutet beispielsweise, den ersten Block in einem Satz zu entfernen, und Zahl bedeutet, den zweiten Block in einem Satz zu entfernen. Wie viele Treffer erzeugt diese Adressfolge?

5.13.4 [10] <5.4, 5.8> Welche Adressen müssen bei jedem Ersatz entfernt werden, um die Trefferzahl zu maximieren? Wie viele Treffer weist diese Adressfolge auf, wenn Sie dieser „optimalen“ Strategie folgen?

5.13.5 [10] <5.4, 5.8> Begründen Sie, warum es schwierig ist, eine Cache-Ersetzungsstrategie zu implementieren, die für alle Adressfolgen gleichermaßen optimal ist.

5.13.6 [10] <5.4, 5.8> Angenommen, Sie könnten für jeden Speicherzugriff entscheiden, ob die angeforderte Adresse in den Cache gestellt werden soll. Welchen Einfluss hätte dies auf die Fehlzugriffsrate?