

Übungsklausur Rechnerarchitekturen

Klausurvariante 5

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 1.9

Wir betrachten einen Prozessor mit den CPI-Werten 1, 12 und 5 für arithmetische Befehle, Lade-/Speicherbefehle bzw. Sprungbefehle. Außerdem nehmen wir an, dass ein auf einem einzelnen Prozessor laufendes Programm die Ausführung von $2,56 \times 10^9$ arithmetischen Befehlen, $1,28 \times 10^9$ Lade-/Speicherbefehlen und 256 Millionen Sprungbefehlen erfordert. Jeder Prozessor habe eine Taktfrequenz von 2 GHz. Das Programm wird parallelisiert, so dass es auf mehreren Kernen läuft, wobei wir annehmen, dass die Anzahl der arithmetischen Befehle und der Lade- und Speicherbefehle je Prozessor durch $0,5 \times p$ geteilt wird (p ist die Anzahl der Prozessoren), während die Anzahl der Sprungbefehle auf jedem Prozessor gleich bleibt.

1.9.1 [5] <1.7> Bestimmen Sie die Gesamtausführungszeit für dieses Programm auf 1, 2, 4 bzw. 8 Prozessoren. Um wie viel wird die Ausführung schneller, wenn statt einem einzigen Prozessor 2, 4 bzw. 8 Prozessoren verwendet werden?

1.9.2 [10] <1.6, 1.8> Welchen Einfluss hätte es auf die Ausführungszeit des Programms auf 1, 2, 4 bzw. 8 Prozessoren, wenn der CPI der arithmetischen Befehle verdoppelt würde?

1.9.3 [10] <1.6, 1.8> Auf welchen Wert müsste der CPI-Wert der Lade- und Speicherbefehle reduziert werden, um mit einem einzelnen Prozessor die Leistung von vier Prozessoren mit den ursprünglichen CPI-Werten zu erreichen?

Aufgabe 2: MIPS

a)

Aufgabe 2.23

[5] <2.7> Angenommen, \$t0 enthält den Wert 0x00101000. Was ist der Wert von \$t2 nach den folgenden Befehlen?

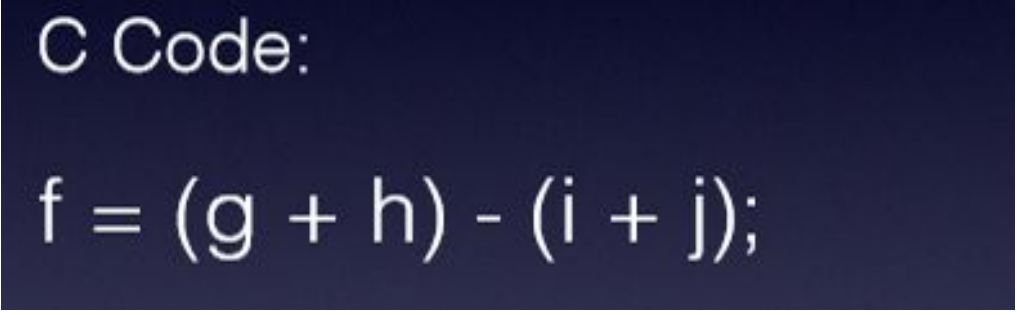
```
    slt    $t2, $0,    $t0
    bne    $t2, $0,    ELSE
    j      DONE
ELSE: addi  $t2, $t2, 2
DONE:
```

b)

Aufgabe 2.37

[5] <2.9> Schreiben Sie ein Programm in MIPS-Assemblersprache, das eine Kette aus ASCII-Ziffern mit positiven und negativen Ganzzahl-Dezimalstrings in eine Ganzzahl konvertiert. Ihr Programm sollte bewirken, dass im Register \$a0\$ die Adresse einer nullterminierten Zeichenkette steht, die eine Kombination der Ziffern 0 bis 9 enthält. Ihr Programm sollte den Integerwert berechnen, der zu dieser Zeichenkette äquivalent ist, und dann diese Zahl im Register \$v0 platzieren. Wenn irgendwo in der Kette ein Zeichen auftaucht, das keine Ziffer ist, dann sollte Ihr Programm mit dem Wert -1 in Register \$v0 anhalten. Wenn das Register \$a0 zum Beispiel auf eine Folge von drei Bytes wie 50_D, 52_D, 0_D (die nullterminierte Zeichenkette 24) verweist, dann sollte das Register \$v0 beim Anhalten den Wert 24_D enthalten.

c)



C Code:

$$f = (g + h) - (i + j);$$

d)

Aufgabe 2.5

[5] <2.2, 2.3> Schreiben Sie den MIPS-Assemblercode aus Aufgabe 2.4 so um, dass die Anzahl der MIPS-Befehle minimiert wird, aber noch immer die gleiche Funktion ausgeführt wird.

Aufgabe 3: Arithmetik

a)

Aufgabe 3.3

[10] <3.2> Konvertieren Sie 5ED4 in eine Binärzahl. Was macht die Basis 16 (Hexadezimaldarstellung) zu einem geeigneten Zahlensystem für die Darstellung von Zahlen in Computern?

b)

Aufgabe 3.5

[5] <3.2> Was ist $4365 - 3412$, wenn diese Werte vorzeichenbehaftete 12-Bit-Oktalzahlen darstellen, die im Vorzeichen-Betrag-Format gespeichert werden? Schreiben Sie das Ergebnis in Oktaldarstellung.

Aufgabe 4: Pipelining

a)

Aufgabe 4.9

In dieser Aufgabe untersuchen wir, wie Datenabhängigkeiten die Ausführung in der einfachen fünfstufigen Pipeline beeinflussen, die wir in Abschnitt 4.5 beschrieben haben. Die Teilaufgaben beziehen sich auf die folgende Befehlssequenz:

```
or r1, r2, r3
or r2, r1, r4
or r1, r1, r2
```

Außerdem setzen wir die folgenden Taktzeiten für die verschiedenen Optionen beim Forwarding voraus: ohne Forwarding 250 ps, mit vollständigem Forwarding 300 ps, nur ALU-ALU-Forwarding 290 ps.

4.9.1 [10] <4.5> Identifizieren Sie Abhängigkeiten und ihre jeweiligen Typen.

4.9.2 [10] <4.5> Nehmen Sie an, dass es bei diesem Prozessor mit Pipelining kein Forwarding gibt. Identifizieren Sie Konflikte und fügen Sie NOP-Befehle hinzu, um sie zu eliminieren.

4.9.3 [10] <4.5> Nehmen Sie an, dass es vollständiges Forwarding gibt. Identifizieren Sie Konflikte und fügen Sie NOP-Befehle hinzu, um sie zu eliminieren.

4.9.4 [10] <4.5> Wie lang ist die Ausführungszeit dieser Befehlssequenz insgesamt für den Fall ohne Forwarding und für den Fall mit vollständigem Forwarding? Um wie viel wird eine Pipeline, die ursprünglich kein Forwarding hat, durch das Hinzufügen von vollständigem Forwarding schneller?

4.9.5 [10] <4.5> Fügen Sie zu diesem Code NOP-Befehle zum Beseitigen von Konflikten hinzu, wenn es nur ALU-ALU-Forwarding gibt (kein Forwarding von der MEM- zur EX-Stufe).

4.9.6 [10] <4.5> Wie lang ist die Ausführungszeit dieser Befehlssequenz insgesamt, wenn es nur ALU-ALU-Forwarding gibt? Um wie viel ist diese Lösung gegenüber einer Pipeline ohne Forwarding schneller?

Aufgabe 5: Caching

a)

Aufgabe 1: Caching (19 Punkte)

/ 19

Für einen direkt abgebildeten byte-adressierten Cache mit 32 Bit-Adressen werden die folgenden Bits der Adresse für den Cache Zugriff verwendet:

Tag	Index	Offset
31...11	10...6	5...0