Übungsklausur Rechnerarchitekturen

Klausurvariante 2

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)

- Zwei verschieden kompilierte Codeversionen die Befehle der Klassen A, B und C verwenden

Klasse	A	B	C
CPI	1	2	3
Befehlszahl Version1	2	1	2
Befehlszahl Version 2	4	1	1

b)

Aufgabe 1.5

[4] <1.6> Betrachten Sie drei verschiedene Prozessoren, P1, P2 und P3, die alle den gleichen Befehlssatz ausführen. P1 hat eine Taktfrequenz von 3 GHz und einen CPI von 1,5. P2 hat eine Taktfrequenz von 2,5 GHz und einen CPI von 1,0. P3 hat eine Taktfrequenz von 4,0 GHz und einen CPI von 2,2.

a. Welcher Prozessor hat die größte Leistung, ausgedrückt in Befehlen pro Sekunde?

b. Angenommen, jeder Prozessor führt ein Programm in 10 Sekunden aus. Bestimmen Sie jeweils die Anzahl der Zyklen und die Anzahl der Befehle.

c. Wir versuchen, die Ausführungszeit um 30 % zu reduzieren, doch das führt zu einem Anstieg des CPI-Werts um 20 %. Welche Taktfrequenz haben wir also, wenn wir diese Zeitverkürzung erreichen?

Aufgabe 2: MIPS

a)

Aufgabe 2.19

Gegeben seien die folgenden Registerinhalte:

```
$t0=0xAAAAAAAA, $t1=0x12345678
```

2.19.1 [5] <2.6> Welchen Wert von `$t2` liefern die oben angegebenen Registerwerte für die folgende Befehlssequenz?

```
sll $t2, $t0, 44
or $t2, $t2, $t1
```

2.19.2 [5] <2.6> Welchen Wert von `$t2` liefern die oben angegebenen Registerwerte für die folgende Befehlssequenz?

```
sll $t2, $t0, 4
andi $t2, $t2, -1
```

2.19.3 [5] <2.6> Welchen Wert von `$t2` liefern die oben angegebenen Registerwerte für die folgende Befehlssequenz?

```
srl $t2, $t0, 3
andi $t2, $t2, 0xFFEF
```

b)

```
        addi $t1, $0, $0
LOOP:   lw   $s1, 0($s0)
        add  $s2, $s2, $s1
        addi $s0, $s0, 4
        addi $t1, $t1, 1
        slti $t2, $t1, 100
        bne  $t2, $s0, LOOP
```

Aufgabe 2.29

[5] <2.7> Übersetzen Sie die folgende Schleife in C. Nehmen Sie an, dass die C-Integervariable `i` im Register `$t1` steht, dass `$s2` die C-Integervariable `result` enthält und dass `$s0` die Basisadresse der Integervariable `MemArray` enthält.

c)

Aufgabe 2.42

[5] <2.6, 2.10> Der aktuelle Wert des Befehlszählers sei `0x1FFFF000`. Können Sie mithilfe eines einzelnen Sprungbefehls die Befehlszähleradresse aus Aufgabe 2.39 erhalten?

$0010\ 0000\ 0000\ 0001\ 0100\ 1001\ 0010\ 0100_B$

d)

Aufgabe 2.1

[5] <2.2> Wie lautet der zu der folgenden C-Anweisung gehörende MIPS-Assemblercode? Nehmen Sie an, dass die Variablen `f`, `g`, `h` und `i` gegeben sind und als 32-Bit-Ganzzahlen betrachtet werden können, die wie üblich in einem C-Programm deklariert sind. Verwenden Sie in Ihrem MIPS-Assemblercode eine minimale Anzahl von Befehlen.

```
f = g + (h - 5);
```

Aufgabe 3: Arithmetik

a)

Aufgabe 3.13

[20] <3.3> Verwenden Sie eine ähnliche Tabelle wie Tabelle 3.2, um das Produkt der vorzeichenlosen hexadezimalen 6-Bit-Ganzzahlen 62 und 12 mit der in Abbildung 3.2 beschriebenen Hardware zu berechnen. Zeigen Sie für jeden Schritt die Registerinhalte.

b)

Aufgabe 3.8

[5] <3.2> Nehmen Sie an, dass 185 und 122 vorzeichenbehaftete dezimale 8-Bit-Ganzzahlen sind, die im Vorzeichen-Betrag-Format gespeichert werden. Berechnen Sie 185 – 122. Gibt es dabei einen Überlauf, einen Unterlauf oder nichts von beidem?

Tab. 3.2: Beispiel für eine Multiplikation mit dem Algorithmus aus Abbildung 3.3.
Das Bit, das den nächsten Schritt bestimmt, ist jeweils unterstrichen.

Iteration	Schritt		Multiplikator	Multiplikand	Produkt
0	Anfangswerte		0011	0000 0010	0000 0000
1	1a:	1 ⇒ Produkt = Produkt + Multiplikand	0011	0000 0010	0000 0010
	2:	schiebe Multiplikand nach links	0011	0000 0100	0000 0010
	3:	schiebe Multiplikator nach rechts	0001	0000 0100	0000 0010
2	1a:	1 ⇒ Produkt = Produkt + Multiplikand	0001	0000 0100	0000 0110
	2:	schiebe Multiplikand nach links	0001	0000 1000	0000 0110
	3:	schiebe Multiplikator nach rechts	0000	0000 1000	0000 0110
3	1:	0 ⇒ keine Operation	0000	0000 1000	0000 0110
	2:	schiebe Multiplikand nach links	0000	0001 0000	0000 0110
	3:	schiebe Multiplikator nach rechts	0000	0001 0000	0000 0110
4	1:	0 ⇒ keine Operation	0000	0001 0000	0000 0110
	2:	schiebe Multiplikand nach links	0000	0010 0000	0000 0110
	3:	schiebe Multiplikator nach rechts	0000	0010 0000	0000 0110

Aufgabe 4: Pipelining

a)

Aufgabe 4.16

In dieser Aufgabe wird die Genauigkeit verschiedener Sprungvorhersagen für das folgende, sich wiederholende (z. B. innerhalb einer Schleife) Muster von Sprungergebnissen: `V, NV, V, V, NV`.

4.16.1 [5] <4.8> Wie gut ist die Genauigkeit des immer-verzweigt-Prädiktors und des nie-verzweigt-Prädiktors für diese Sequenz von Sprungergebnissen?

4.16.2 [5] <4.8> Wie gut ist die Genauigkeit des 2-Bit-Prädiktors für die ersten vier Sprünge in diesem Muster, wenn wir annehmen, dass der Prädiktor im linken unteren Zustand von Abbildung 4.53 (nicht verzweigt) startet?

4.16.3 [10] <4.8> Wie gut ist die Genauigkeit des 2-Bit-Prädiktors, wenn dieses Muster endlos wiederholt wird?

4.16.4 [30] <4.8> Entwerfen Sie einen Prädiktor, der perfekte Genauigkeit erreichen würde, wenn dieses Muster endlos wiederholt würde. Ihr Prädiktor sollte ein Schaltwerk mit einem Ausgang sein, der eine Vorhersage liefert (1 für verzweigt, 0 für nicht verzweigt) und keine Eingaben außer dem Takt und dem Steuersignal hat, das anzeigt, dass der Befehl ein bedingter Sprung ist.

4.16.5 [10] <4.8> Wie gut ist die Genauigkeit Ihres Prädiktors aus 14.6.4, wenn das genaue Gegenteil des bisher betrachteten Musters ist?

4.16.6 [20] <4.8> Wiederholen Sie 4.16.4, wobei Ihr Prädiktor diesmal am Ende in der Lage sein soll (nach einer Aufwärmperiode, während der er falsche Vorhersagen machen kann), sowohl das gegebene Muster als auch sein Gegenteil perfekt vorherzusagen. Ihr Prädiktor sollte eine Eingabe haben, der ihm mitteilt, was die tatsächliche Ausgabe war. Hinweis: Diese Eingabe erlaubt es Ihrem Prädiktor festzustellen, welches der beiden Muster gegeben ist.

Aufgabe 5: Caching

a)

Aufgabe 5.11

Wie in Abschnitt 5.7 beschrieben, verwendet der virtuelle Speicher eine Seitentabelle, um die Abbildung der virtuellen Adressen auf physikalische Adressen zu verfolgen. Bei dieser Aufgabe wird untersucht, wie diese Tabelle beim Zugriff auf Adressen aktualisiert werden muss. Die folgende Tabelle zeigt eine Folge virtueller Adressen auf einem System. Gehen Sie von einer Seitengröße von 4 KiB aus, einem vollständig assoziativen TLB mit vier Einträgen und echter LRU-Ersetzung. Wenn Seiten von der Festplatte geladen werden müssen, wird die nächstgrößte Seitennummer inkrementiert.

```
4669, 2227, 13916, 34587, 48870, 12608, 49225
```

TLB

Gültig	Tag	Physische Seitennummer
1	11	12
1	7	4
1	3	6
0	4	9

Seitentabelle

Gültig	Physische Seite oder auf der Festplatte
1	5
0	Festplatte
0	Festplatte
1	6
1	9
1	11
0	Festplatte
1	4
0	Festplatte
0	Festplatte
1	3
1	12

5.11.1 [10] <5.7> Gegeben sei die Adressfolge der Tabelle und der Anfangszustand des TLB und der Seitentabelle. Zeigen Sie den endgültigen Zustand des Systems. Geben Sie für jeden Zugriff an, ob es sich um einen Treffer im TLB, einen Treffer in der Seitentabelle oder einen Seitenfehler handelt.

5.11.2 [15] <5.7> Wiederholen Sie Aufgabe 5.11.1, aber verwenden Sie jetzt 16 KiB große Seiten anstatt 4 KiB große Seiten. Nennen Sie einige Vorteile einer größeren Seitengröße. Nennen Sie einige Nachteile.

5.11.3 [15] <5.4, 5.7> Zeigen Sie den endgültigen Inhalt des TLB, wenn er zweifach satzassoziativ ist. Zeigen Sie auch den Inhalt des TLB, wenn er direkt abgebildet ist. Diskutieren Sie, was es bedeutet, einen hochleistungsfähigen TLB zu haben. Wie würden Zugriffe auf den virtuellen Speicher verarbeitet, wenn es keinen TLB gäbe?

Mehrere Parameter beeinflussen die Gesamtgröße der Seitentabelle. Nachfolgend sind einige wichtige Seitentabellenparameter aufgelistet.

Virtuelle Adressgröße	Seitengröße	Seitentabelleneintragsgröße
32 Bit	8 KiB	4 Byte

5.11.4 [5] <5.7> Berechnen Sie für die Parameter in der obigen Tabelle die Gesamtseitentabellengröße für ein System, auf dem fünf Applikationen ausgeführt werden und das die Hälfte des verfügbaren Speichers nutzt.

5.11.5 [10] <5.7> Berechnen Sie anhand der in der obigen Tabelle vorgegebenen Parameter die Gesamtseitentabellengröße für ein System, auf dem fünf Applikationen ausgeführt werden, und das die Hälfte des verfügbaren Speichers nutzt. Gehen Sie dabei von einem zweistufigen Seitentabellenansatz mit 256 Einträgen aus. Gehen Sie davon aus, dass jeder Eintrag in der Hauptseitentabelle 6 Byte groß ist. Berechnen Sie die minimale und maximale benötigte Speichergröße.

5.11.6 [10] <5.7> Ein Cache-Entwickler will die Größe eines virtuell indizierten, physisch getagten 4 KiB-Cache erhöhen. Gehen Sie von der in der obigen Tabelle beschriebenen Seitengröße aus. Ist es möglich, einen direkt abgebildeten 16 KiB-Cache mit zwei Wörtern pro Block zu erstellen? Wie würde der Entwickler die Datengröße des Cache erhöhen?