Klausur zur Vorlesung

Grundlagen der Rechnerarchitektur / Technische Informatik (GRA/TI)

Prof. Marco Platzner
Fachgebiet Technische Informatik
Universität Paderborn

07.10.2009

- Die Bearbeitungsdauer beträgt für alle Studenten **90 Minuten.** Es sind **alle 6 Aufgaben** zu bearbeiten.
- Es sind keine Hilfsmittel zugelassen.
- Schreiben Sie nicht mit Bleistift oder Rotstift.
- Verwenden Sie kein eigenes Papier. Bei Bedarf bekommen Sie Papier bei der Klausuraufsicht.
- Schreiben Sie auf jedes Blatt (auch auf das Konzeptpapier) in Blockschrift Ihren Namen und Ihre Matrikelnummer.
- Bei mehreren präsentierten Lösungen wird die Aufgabe nicht gewertet! Streichen Sie daher bei Angabe mehrerer Lösungsansätze die nicht zu bewertenden Lösungen durch! Verwenden Sie kein Tipp-Ex.
- Abschreiben und abschreiben lassen oder Hilfe Dritter führt zum Nichtbestehen der Klausur.

Nachname: _____

Vorname: _____

Matrikelnummer: _____

Studiengang: _____

Aufkleber

Aufgabe	**1**	**2**	**3**	**4**	**5**	**6**	$\sum$
Punkte	**15**	**15**	**20**	**10**	**10**	**20**	90
Erreicht							

Aufgabe 1 (Multiple Choice) **[15 Punkte]**

Bei den folgenden Fragen können keine, eine oder mehrere Antworten richtig sein. Kreuzen Sie die richtigen Antworten deutlich an.

(a) Wie nennt man die Konvention, nach der das höchstwertigste Byte eines Wortes an der niedrigsten Byte-Adresse im Speicher abgelegt wird?

- ☐ Von-Neumann Rechnerarchitektur
- ☐ Big-Endian
- ☐ Little-Endian
- ☐ Stack-Architektur

(b) Was sind Eigenschaften der Einzyklenimplementierung?

- ☐ Jede Instruktion wird in einem Taktzyklus ausgeführt.
- ☐ Der CPI-Wert ist kleiner als 1.
- ☐ Es muss getrennte Speicher für Daten und Instruktionen geben.

(c) Was sind Gründe für die Verwendung von virtuellem Speicher?

- ☐ Man kann leichter verschiedenste Speichermodule anschliessen.
- ☐ Einem Prozess kann mehr Speicher zur Verfügung gestellt werden, als physikalisch vorhanden ist.
- ☐ Jeder Prozess kann seinen eigenen Adressraum haben.
- ☐ Schutzmechanismen lassen sich gut implementieren.

(d) Welche der folgenden Verfahren sind verteilte Arbitrierungsverfahren?

- ☐ Arbitrierung durch Selbstselektion
- ☐ Daisy-Chain Arbitrierung
- ☐ zentrale Arbitrierung
- ☐ Arbitrierung durch Kollisionserkennung

(e) Durch welche Prozessorarchitektur kann man einen CPI-Wert kleiner 1 erreichen?

- ☐ tiefes Pipelining
- ☐ Mehrzyklenimplementierung
- ☐ Mehrfachzuordnung mit Pipelining
- ☐ Mehrfachzuordnung mit tiefem Pipelining

Aufgabe 2 (Assembler) **[15 Punkte]**

Die rekursive Funktion m hat die Eingabeparameter $a0 und $a1, die auf Folgen ganzzahliger Zahlen zeigen, terminiert durch eine 0, sowie den Ausgabeparameter $a2, der auf eine leere Folge der Länge |$a0|+|$a1| zeigt.

```
 1  m:      addi    $sp,    $sp,    -4      _____________________________
 2          sw      $ra,    0($sp)          _____________________________
 3
 4          lw      $t0,    0($a0)          _____________________________
 5          lw      $t1,    0($a1)          _____________________________
 6
 7          bne     $t0,    $zero,  X       _____________________________
 8          beq     $t1,    $zero,  Y4      _____________________________
 9          sw      $t1,    0($a2)          _____________________________
10          addi    $a1,    $a1,    4       _____________________________
11          j       Y2                      _____________________________
12
13  X:      slt     $t1,    $t1,    $t0     _____________________________
14          beq     $t1,    $zero,  Y1      _____________________________
15
16          add     $t0,    $a0,    $zero   _____________________________
17          add     $a0,    $a1,    $zero   _____________________________
18          add     $a1,    $t0,    $zero   _____________________________
19          j       Y3                      _____________________________
20
21  Y1:     sw      $t0,    0($a2)          _____________________________
22          addi    $a0,    $a0,    4       _____________________________
23  Y2:     addi    $a2,    $a2,    4       _____________________________
24  Y3:     jal     m                       _____________________________
25
26  Y4:     lw      $ra,    0($sp)          _____________________________
27          addi    $sp,    $sp,    4       _____________________________
28          jr      $ra                     _____________________________
```

(a) Die Funktion m wird ausgeführt mit den Parametern: $a0=0x100, $a1=0x108 und $a2=0x114. Berechnen Sie die Ausgabe von m. Notieren Sie die Werte der Register $sp, $a0, $a1 und $a2 bei jedem Aufruf der Funktion. Vervollständigen Sie weiterhin die Tabelle mit der Speicherbelegung.

Adresse	0x100	0x104	0x108	0x10C	0x110	0x114	0x118	0x11C
Inhlat	3	0	1	4	0			

Tabelle 1: Speicherbelegung

$sp	0x200								
$a0	0x100								
$a1	0x108								
$a2	0x114								

Tabelle 2: Registerinhalte

(b) Welche Voraussetzungen müssen die Eingabefolgen erfüllen, damit die Ausgabefolge sortiert ist?

Antwort: _____

Adresse	0x100	0x104	0x108	0x10C	0x110	0x114	0x118	0x11C
Inhlat	3	0	1	4	0			

Tabelle 3: Speicherbelegung (Ersatz)

$sp	0x200								
$a0	0x100								
$a1	0x108								
$a2	0x114								

Tabelle 4: Registerinhalte (Ersatz)

Aufgabe 3 (Instruktionssatzarchitekturen) **[20 Punkte]**
Es sind zwei Codefragmente für unterschiedliche Rechnerarchitekturen gegeben, die beide das selbe berechnen. Untersuchen Sie die Rechnerarchitekturen anhand dieser Codefragmente bezüglich der Parameter **Speicherbedarf** und **Speicherbandbreite**.
Für beide Rechnerarchitekturen soll gelten:

- Der Adressbus hat eine Breite von 32 Bit.
- der Datenbus hat eine Breite von 32 Bit.
- Die Daten im Speicher sind 32 Bit breit.
- Alle hier benutzen bedingten Sprünge sind relativ und benutzen einen 16 Bit Sprungoffset.

Beachten Sie, dass das Programm Schleifen besitzt. Berücksichtigen Sie dies bei der Berechnung der **Speicherbandbreite**.

Register-Register-Architektur (MIPS)

```
main: add  $s0, $zero, $zero # Register $s0 mit 0 initialisieren
      add  $s1, $zero, $zero
      lui  $s1, 0x0020       # Adresse 0x2000 in $s1 laden
      add  $s2, $zero, $zero
      lui  $s2, 0x0030       # Adresse 0x3000 in $s2 laden
      addi $s3, $zero, 40    # Register $s3 mit 10 initialisieren
      lw   $t2, 0($s2)       # Wort von Speicherstelle $s2 laden
loop: add  $t0, $s1, $s0     # Adresse (0x2000+x) in $t0 berechnen
      lw   $t1, 0($t0)       # Wort von Speicherstelle $t0 laden
      add  $t2, $t1, $t2     # Beide Werte addieren
      addi $t0, $zero, 3     # $t0 mit 3 laden
wait: addi $t0, $t0, -1      # $t0 um 1 dekrementieren
      bne  $t0, $zero, wait  # Springe zu wait: falls $t0 != 0
      addi $s0, $s0, 4       # $s0 um 4 inkrementieren
      bne  $s0, $s3, loop    # springe zu loop: falls $s0 != $s3
      sw   $t2, 0($s2)       # $t1 an Speicherstelle $s2 schreiben
```

(a) Schreiben Sie den Speicherbedarf und die Speicherbandbreite (für das **gesamte ausgeführte Programm**) in **Bytes** in die nachfolgende Tabelle:

Code			Speicherbedarf	Speicherbandbreite
main:	add	$s0, $zero, $zero		
	add	$s1, $zero, $zero		
	lui	$s1, 0x0020		
	add	$s2, $zero, $zero		
	lui	$s2, 0x0030		
	addi	$s3, $zero, 40		
	lw	$t2, 0($s2)		
loop:	add	$t0, $s1, $s0		
	lw	$t1, 0($t0)		
	add	$t2, $t1, $t2		
	addi	$t0, $zero, 3		
wait:	addi	$t0, $t0, -1		
	bne	$t0, $zero, wait		
	addi	$s0, $s0, 4		
	bne	$s0, $s3, loop		
	sw	$t2, 0($s2)		
		Summe (Bytes):		

Akkumulator-Architektur (angelehnt an MOS 6510)

Hinweise:

- Die Architektur hat eine variable Instruktionslänge von 1-5 Byte.
- x ist ein (Index-)Register im Prozessor
- Der Akkumulator ist 32 Bit breit, das Index-Register x ist 16 Bit breit.
- Vergleiche setzen Condition-Codes, die von bedingten Sprüngen ausgewertet werden.

```
main: ldx 0x0000          # Lade Register x mit 0
loop: lda 0x00200000,x    # Lade das Wort von Adresse (0x00200000+x) in den Akku
      add 0x00300000      # Addiere das Wort von Adresse 0x00300000 zum Akku
      sta 0x00300000      # Speichere den Akku nach Adresse 0x00300000
      lda 0x0003          # Lade den Akku mit 3
wait: dec                 # Dekrementiere den Akku um 1
      cmp 0x0000          # Vergleiche den Akku mit 0
      bne wait            # Ist das Register y nicht 0 springe zu wait:
      inx 0x0004          # Inkrementiere das Register x um 4
      cpx 0x0028          # Vergleiche das Register x mit 40
      bne loop            # Ist Register x nicht 40 springe zu loop:
```

(b) Schreiben Sie den Speicherbedarf und die Speicherbandbreite (für das **gesamte ausgeführte Programm**) in **Bytes** in die nachfolgende Tabelle:

Code			Speicherbedarf	Speicherbandbreite
main:	ldx	0x0000		
loop:	lda	0x00200000,x		
	add	0x00300000		
	sta	0x00300000		
	lda	0x0003		
wait:	dec			
	cmp	0x0000		
	bne	wait		
	inx	0x0004		
	cpx	0x0028		
	bne	loop		
		Summe (Bytes):		

Aufgabe 4 (Pipelining) **[10 Punkte]**

Basierend auf einem MIPS-Prozessor mit Pipelining seien folgende Eigenschaften gegeben:

- Die Pipeline bestehe aus 5 Stufen: IF, ID, EX, MEM, WB.
- Die Registerzugriffe erfolgen im Halbtaktverfahren.
- Harvard-Architektur sei realisiert.

Betrachten Sie folgende MIPS-Assembler Programmsequenz:

```
01: lw $8, 0($4)
02: add $4, $0, $5
03: lw $9, 0($8)
04: add $10, $9, $8
05: add $2, $9, $0
06: sw $10, 0($9)
```

(a) Geben Sie die Anzahl der für die Ausführung obiger Programmsequenz benötigten Taktzyklen an, für den Fall, dass keine Forwarding-Unit verwendet wird. Veranschaulichen Sie dazu die Pipelinebelegung mit Hilfe des nachfolgenden Diagramms.

Anzahl Taktzyklen:_____

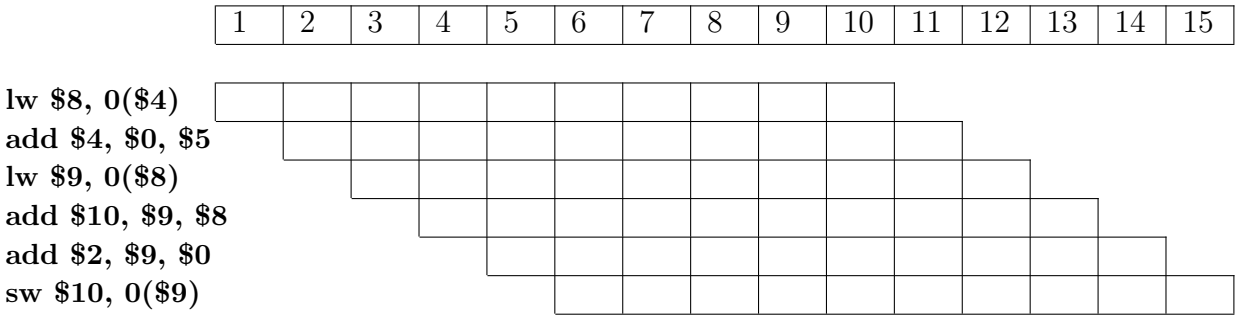
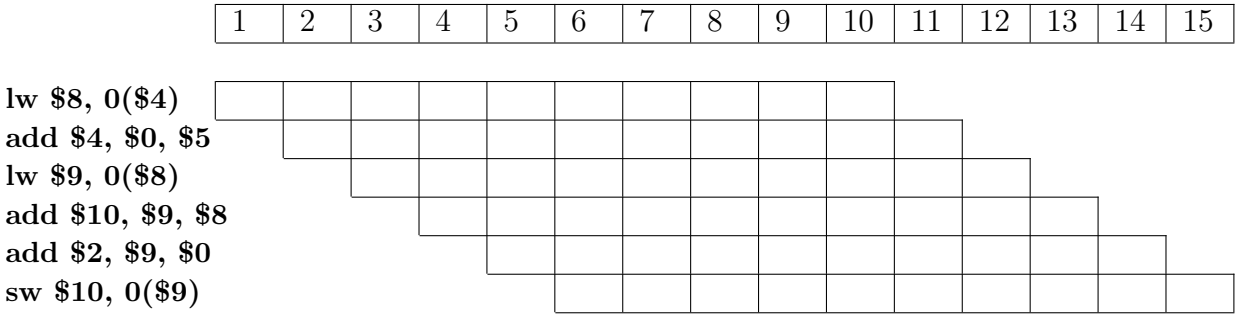


Diagramm zu (a)



Ersatzdiagramm zu (a)
(ungültige Lösung streichen!)

(b) Geben Sie wiederum die Anzahl der für die Ausführung obiger Programmsequenz benötigten Taktzyklen an. Betrachten Sie hierbei den Fall, dass eine Forwarding-Unit verwendet wird. Veranschaulichen Sie dazu wiederum die Pipelinebelegung mit Hilfe des nachfolgenden Diagramms.

Anzahl Taktzyklen:_____

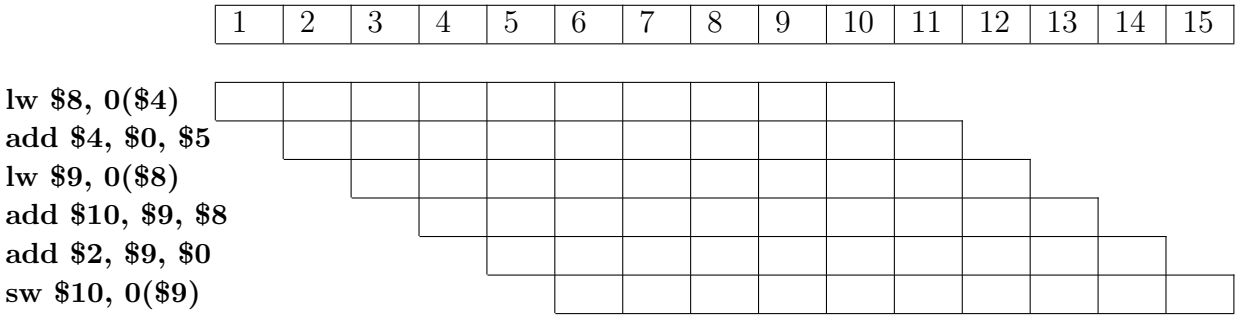
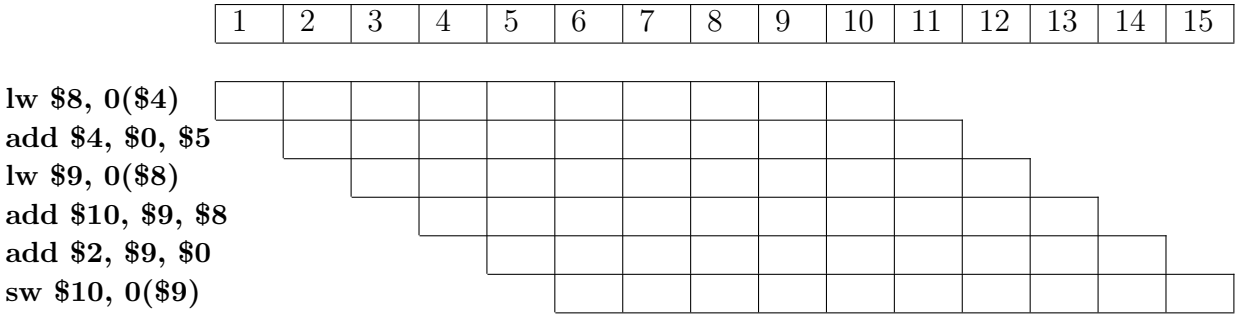


Diagramm zu (b)



Ersatzdiagramm zu (b)
(ungültige Lösung streichen!)

(c) Kann durch Umsortieren der Programmsequenz die Ausführung weiter beschleunigt werden? Begründen Sie Ihre Antwort.

Aufgabe 5 (Multilevel Cache) **[10 Punkte]**

Ein 2 GHz Prozessor hat einen CPI-Wert ohne memory stall cycles von 1,0. Die miss-rate des L1 Caches beträgt 4%, das Bereitstellen der Daten aus dem Hauptspeicher dauert für einen Cacheblock 250ns.

(a) Wie hoch ist die mittlere Zugriffszeit (in Taktzyklen)?

(b) Es wird ein zweiter Cache (L2-Cache) mit einer hit-time von 25ns hinzugefügt. Die miss-penalty für diesen Cache beträgt 250ns. Welche miss-rate muss der neue L2 Cache haben, damit die Prozessorperformance um den Faktor 6 steigt? (Verwenden Sie auch hier die mittlere Zugriffszeit als Metrik!)

Aufgabe 6 (Prozessorperformance) **[20 Punkte]**

In Tabelle 5 werden drei Prozessoren verglichen, die sich nur durch ihre Cacheorganisationen unterscheiden. Für einen bestimmten Workload, bei dem ein Drittel aller Instruktionen Datenzugriffe sind, wurden die angegebenen miss-rates ermittelt. Die Cache miss-penalty beträgt für alle Systeme 8+(Blockgröße [Worte]) Zyklen (Blockgröße = 2^m). An Prozessor P1 wurde ein CPI von 2,66 gemessen.

Prozessor	T	Cacheorganisation	instruction miss-rate	data miss-rate
P1	400 ps	direkte Abbildung, $m = 0$	5 %	7 %
P2	300 ps	direkte Abbildung, $m = 3$	2 %	6 %
P3	400 ps	4-fach satzassoziativ, $m = 2$	3 %	3 %

Tabelle 5: Organisation und miss-rates dreier Prozessoren bei gleichem Workload

(a) Wie lange müssen die einzelnen Prozessoren bei diesem Workload im Schnitt pro Instruktion auf den Speicher warten?

(b) Welcher Prozessor benötigt für den gegebenen Workload die kürzeste Ausführungszeit?

(c) An einem weiteren Prozessor P4 (T=300 ps, 2-fach satzassoziativ, $m = 2$, data miss-rate = 6%), der ebenfalls bis auf die Cacheorganisation mit den anderen Prozessoren identisch ist, wird bei gleichem Workload ein CPI von 2,72 gemessen. Welche instruction miss-rate ergibt sich daraus?

Konzeptpapier: Falls der Platz unter den einzelnen Aufgaben nicht ausreicht, können Sie diese Seiten für Zwischenrechnungen nutzen. Bitte Lösung und Lösungsweg eindeutig mit der Aufgabennummer markieren!

Konzeptpapier: Falls der Platz unter den einzelnen Aufgaben nicht ausreicht, können Sie diese Seiten für Zwischenrechnungen nutzen. Bitte Lösung und Lösungsweg eindeutig mit der Aufgabennummer markieren!

Konzeptpapier: Falls der Platz unter den einzelnen Aufgaben nicht ausreicht, können Sie diese Seiten für Zwischenrechnungen nutzen. Bitte Lösung und Lösungsweg eindeutig mit der Aufgabennummer markieren!

Konzeptpapier: Falls der Platz unter den einzelnen Aufgaben nicht ausreicht, können Sie diese Seiten für Zwischenrechnungen nutzen. Bitte Lösung und Lösungsweg eindeutig mit der Aufgabennummer markieren!