Universität Paderborn

Institut *Elektrotechnik und Informationstechnik*

Fachgebiet *Datentechnik*

Prof. Sybille Hellebrand

Klausur

Technische Informatik / Grundlagen der Rechnerarchitektur

23. Februar 2016

Punkteverteilung						
Aufgabe	1	2	3	4	5	Σ
maximale Punkte	13	20	18	19	20	90
erreichte Punkte						

Note:	

Aufkleber

Name:	
Matrikelnummer:	
Studienrichtung:	

Hinweise:
Für die Lösung der Klausuraufgaben sind ausschließlich die Aufgabenblätter zu verwenden. Lösungsangaben außerhalb der Aufgabenblätter („Schmierzettel“, etc.) werden bei der Bewertung nicht berücksichtigt!

Beschriften Sie jede Doppelseite mit Ihrer Matrikelnummer!

Mit Bleistift oder der Korrekturfarbe rot angefertigte Lösungen werden nicht bewertet!

Die Verwendung von „Tipp-Ex“ oder „Tintenkiller“ ist untersagt.

Es ist ein handgeschriebener DIN-A4 Zettel als Hilfsmittel zugelassen!

Es sind keine weiteren Hilfsmittel zugelassen!

Aufgabe 1: (Leistungsbewertung) 13 Punkte

a) Auf einem Prozessor wird der SPEC2000 Floating-Point Benchmark ausgeführt. Während der Programmlaufzeit werden die in Tabelle 1 aufgelisteten MIPS-Instruktionen mit der entsprechenden Häufigkeit ausgeführt. Die durchschnittliche Anzahl von Taktzyklen pro Instruktion (CPI) ist für die in Tabelle 2 gezeigten Instruktionskategorien bekannt. Berechnen Sie die Anzahl der benötigten Taktzyklen pro Instruktion für den SPEC2000 Floating-Point Benchmark auf dem verwendeten Prozessor.

(5 Punkte)

Core MIPS	**Name**	**Häufigkeit**
add unsigned	addu	24%
subtract unsigned	subu	3%
or	or	3%
nor	nor	2%
load word	lw	15%
store word	sw	7%
branch on equal	beq	2%
branch on not equal	bne	2%
FP add double	add.d	8%
FP subtract double	sub.d	4%
FP multiply double	mul.d	8%
load FP double	l.d	15%
store FP double	s.d	7%

Tabelle 1: Häufigkeit der MIPS-Instruktionen im SPEC2000 Floating Point Benchmark

Instruktions-kategorien	**CPI**
ALU	1
load and store	1,5
conditional branch	1,5
floating-point add/sub	1,0
floating-point multiply	3,0

Tabelle 2: CPI für MIPS-Instruktionskategorien

b) In Aufgabenteil **a)** wurden keine Speicherfehlgriffe berücksichtigt. Die CPU muss 40 Taktzyklen auf die benötigten Daten warten, falls ein Fehlgriff auftritt. Um wie viele Taktzyklen verschlechtert sich die durchschnittliche Anzahl der Taktzyklen pro Instruktion, wenn während der Befehlsspeicherzugriffe eine Fehlgriffsrate von 10% auftritt und die Datenspeicherzugriffe eine Fehlgriffsrate von 7,5% aufweisen?

Hinweis: Nehmen Sie an, dass 50% der ausgeführten Befehle Speicherzugriffe sind.

(2 Punkte)

Ein Prozessorhersteller steht vor der Entscheidung einen Krypto-Core in der neuen Prozessorgeneration zu integrieren. Mit Hilfe dieses Hardwaremoduls können Verschlüsselungsalgorithmen 5 mal schneller ausgeführt werden. Für die Ausführung eines typischen Benchmarks benötigt die bisherige Prozessorgeneration 15 Sekunden.

c) Um wie viel mal ist die neue Prozessorgeneration schneller, wenn der bisherige Prozessor während der Hälfte der Ausführungszeit des Benchmarks mit der Berechnung von Verschlüsselungsalgorithmen beschäftigt ist.

(2 Punkte)

d) Welchen Anteil der Ausführungszeit muss der bisherige Prozessor mit den Verschlüsselungsalgorithmen beschäftigt sein, damit der Benchmark auf der neuen Prozessorgeneration 3 mal schneller ausgeführt werden kann?

(4 Punkte)

Aufgabe 2: (Assembler Programmierung) 20 Punkte

a) Die Funktion `teilersumme` ermittelt die Summe der Teiler einer Zahl n in einem Intervall $[a : b]$, indem es die Summe der Teiler von n in Teilintervallen rekursiv berechnet. Die Zahl n steht im Register `$s0`, und die jeweiligen Intervallgrenzen sind in `$a0` und `$a1` abgelegt. Nehmen Sie an, dass die Funktion mit den Registerwerten `$s0=6`, `$s1=0`, `$a0=1` und `$a1=4` aufgerufen wurde.Vervollständigen Sie die Funktion! (4 Punkte)

```
teilersumme:
01      subi    $sp, $sp, 16            # Schaffe Platz fuer 4 Worte
02      sw      $ra, 12($sp)            # Sichere Ruecksprungadresse
03      sw      $s1, 8($sp)             # Sichere Partialsumme
04      sw      $a0, 4($sp)             # Sichere untere Intervallgrenze
05      sw      $a1, 0($sp)             # Sichere obere  Intervallgrenze

06      addi    $t1, $a0, 1                # Intervall weiter zerlegen
07      _______ $t1, $a1, intervallteilung # falls Intervalllaenge > 1

08      add     $v0, $zero, $a0         # Ergebnis = a0
09      rem     $t1, $s0, $a0           # t1 = Rest von s0/a0
10      _______ $t1, $zero, return      # t1 == 0?
11      addi    $v0, $zero, 0           # Ergebnis = 0
12      _______ return

intervallteilung:
13      add     $a1, $a0, $a1           # Neue obere Intervallgrenze
14      addi    $a1, $a1, 1
15      srl     $a1, $a1, 1
16      ____________________________    # Rekursion in Teilintervall
17      add     $s1, $zero, $v0         # Zwischenergebnis merken

18      add     $a0, $zero, $a1         # Neue untere Intervallgrenze
19      lw      $a1, 0($sp)             # Alte obere  Intervallgrenze
20      ____________________________    # Rekursion in Teilintervall
21      add     $v0, $v0, $s1           # Beide Ergebnisse addieren

return:
22      lw      $a1, 0($sp)             # Vom Stack: obere  Intervallgrenze
23      lw      $a0, 4($sp)             # Vom Stack: untere Intervallgrenze
24      lw      $s1, 8($sp)             # Vom Stack: Partialsumme
25      lw      $ra, 12($sp)            # Vom Stack: Ruecksprungadresse
26      addi    $sp, $sp, 16            # Entferne 4 Worte vom Stack
27      jr      $ra                     # Return
```

b) Wieso muss die Funktion den Inhalt von `$s1` auf den Stack sichern? (2 Punkte)

c) Welche rekursiven Aufrufe arbeitet die Funktion bis zur Ermittlung der Lösung ab? Füllen Sie die folgende Tabelle aus, indem Sie jeweils eine Spalte anfügen, wenn Zeile `05` des Programms erreicht wird. Schreiben Sie in das Register `$ra` statt einer Speicheradresse einfach die entsprechende Zeilennummer des Programms, an die gesprungen werden soll. Der Stackpointer `$sp` soll zu Beginn des Programms auf Adresse 0x1000FFA0 zeigen. Beim Ausfüllen dürfen sie das Präfix „0x1000“ weglassen.

(12 Punkte)

Zeile	—	05	05	...				
`$sp`	FFA0							
`$a0`	1							
`$a1`	4							
`$s0`	6							
`$s1`	0							
`$ra`	—							

Tabelle für Registerwerte.

Zeile	—	05	05	...				
`$sp`	FFA0							
`$a0`	1							
`$a1`	4							
`$s0`	6							
`$s1`	0							
`$ra`	—							

Ersatztabelle, **ungültige Lösungen streichen!**

d) Veranschaulichen Sie die Aufrufstruktur aus c) mit Hilfe eines Aufrufbaumes wie in Abbildung 1: Schreiben Sie in die Knoten die Intervallgrenzen und an die Kanten die von unten zurückgelieferten Teilsummen eines Funktionsaufrufs. Was ist das Endergebnis?

(2 Punkte)

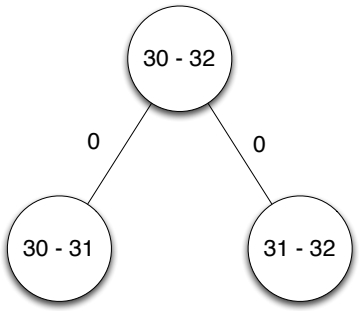


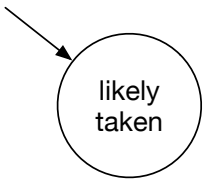
Abbildung 1: Ausschnitt eines Aufrufbaums in dem kein Teiler der Zahl vorhanden ist.

Ergebnis: _____

Aufgabe 3: (Pipelining) 18 Punkte

a) In einem Programm hat eine bestimmte Branch-Instruktion nach einem Schleifendurchlauf das Muster `T, NT, T, T, NT` (mit `T` = taken, `NT` = not taken). Geben Sie an, welche Vorhersagen die aus der Vorlesung bekannte dynamische 2-bit Vorhersagemethode für diese Instruktion ausgeben würde. Nehmen Sie an, der Prädiktor sei mit `likely taken` initialisiert. Zeichnen Sie den Automatengraphen und berechnen Sie die Vorhersagegenauigkeit für diesen Fall.

(5 Punkte)



b) Gegeben sei ein Programm, das auf der aus der Vorlesung bekannten fünfstufigen Pipelinesteuerung ausgeführt wird. Die Befehlsaufteilung ist in folgender Tabelle angegeben:

R-Typ	beq	jmp	lw	sw
40 %	20 %	10 %	25 %	5 %

Für das Programm sollen folgende Methoden zur Sprungvorhersage mit ihren jeweiligen Vorhersagegenauigkeiten untersucht werden:

Always-taken	Always-not-taken	2-bit dynamisch
45 %	55 %	80 %

Durch falsch vorhergesagte Branch-Instruktionen vergrößert sich der CPI. Berechnen Sie den resultierenden CPI für die Methoden `Always-Taken` und `2-bit dynamisch`. Bei der verwendeten Hardware werden sowohl die Adresse des Folgebefehls als auch die Sprungadresse in der IF-Phase berechnet. Die Sprungentscheidung steht nach der EX-Phase fest. Nehmen Sie an, dass keine Data-Hazards existieren, keine Branch Delay Slots verwendet werden und beim Zurücksetzen der Pipeline keine Strafe entsteht.

(5 Punkte)

c) Nehmen Sie an, eine Verbesserung erlaubt, die Hälfte aller Branch-Instruktionen durch jeweils eine ALU Instruktion (R-Typ) zu ersetzen. Korrekt und inkorrekt vorhergesagte Instruktionen haben die gleiche Wahrscheinlichkeit, ersetzt zu werden. Berechnen Sie den aus der Verbesserung resultierenden Speedup, wenn in beiden Fällen die dynamische 2-bit Vorhersagemethode verwendet wird.

(5 Punkte)

d) Manche Branch-Instruktionen sind einfacher vorherzusagen, als andere. Nehmen Sie an, 75% aller ausgeführten Branch-Instruktionen sind einfach vorherzusagende Loop-Back Branches, die immer korrekt vorhergesagt werden. Wie hoch muss die Vorhersagegenauigkeit der dynamischen 2-bit Vorhersage für die schwierig vorherzusagenden Non-Loop-Back Branch Instruktionen sein, damit wie in **b)** insgesamt eine Vorhersagegenauigkeit von 80% erreicht wird?

(3 Punkte)

Aufgabe 4: (Datenpfad) 19 Punkte

a) Benennen Sie in Abbildung 2 die drei MIPS Befehlstypen und die Bereiche der Befehlseinteilung (RS, RT, ...). Geben Sie außerdem die entsprechenden Bit-Grenzen an (gestrichelte Kästchen).

(2 Punkte)

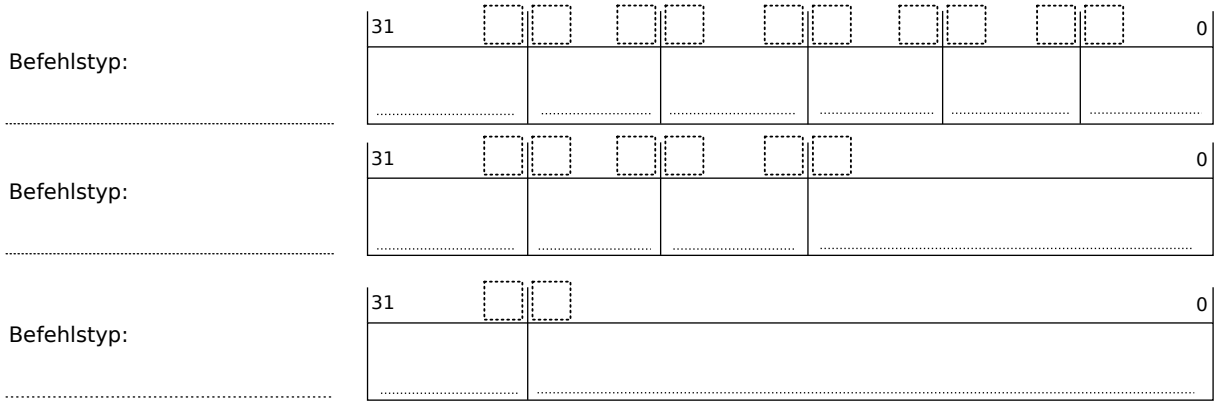


Abbildung 2: Bitgrenzen der Befehlstypen

b) Wie groß ist das Register File bei dem in der Vorlesung behandelten MIPS Prozessor?

(1 Punkt)

---

c) Tragen Sie in die gestrichelten Kästchen von Abbildung 3 die Busbreiten der jeweiligen Leitungen ein.

(1 Punkt)

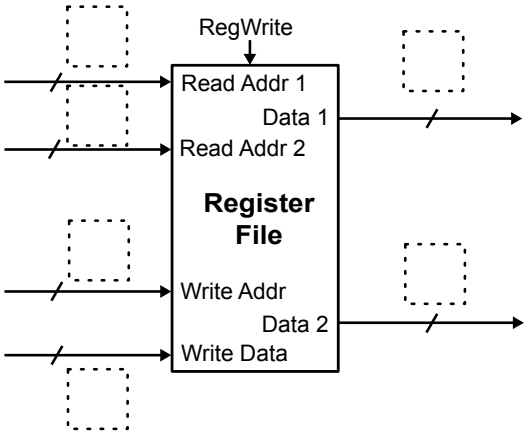


Abbildung 3: Register File

d) Bei bestimmten Befehlsfolgen kann es zu *Daten Hazards* kommen. Diese sollen durch den Datenpfad mit *Forwarding* in Abbildung 4 aufgelöst werden. Markieren Sie die für das *Forwarding* notwendigen Datenleitungen. Die Steuerleitung der *Forwarding Unit* sollen nicht betrachtet werden (Ersatzabbildung auf Seite 15, **ungültige Lösung streichen!**). (2 Punkte)

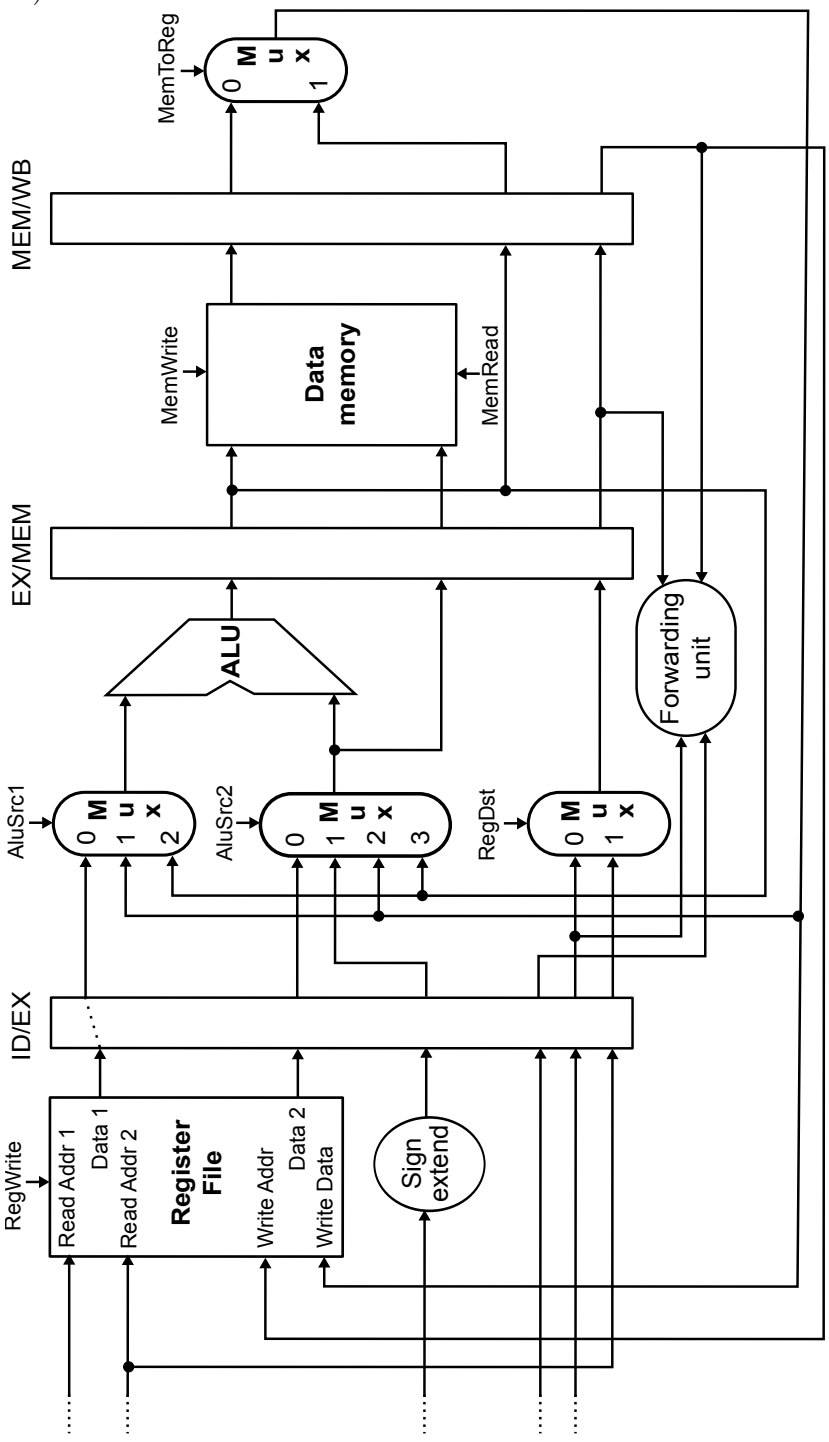


Abbildung 4: Teil des MIPS Datenpfads

e) Gegeben sei folgende Befehlsfolge *addi*, *subi* (Abbildung 5). Zum Takt T_1 befindet sich der erste Befehl in der IF-Phase.

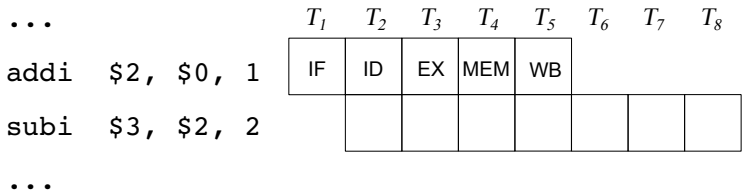


Abbildung 5: Befehlsfolge mit Daten Hazard

Vervollständigen Sie die Phasen für den zweiten Befehl und zeichnen Sie das *Forwarding* in Abbildung 5 ein.

Setzen Sie die Steuerleitungen des Datenpfads aus Abbildung 4 in Tabelle 3.1 so, dass die Befehlsfolge korrekt ausgeführt wird. Markieren Sie alle nicht relevanten Steuerleitungen mit einem **X**.

(5 Punkte)

	T_1	T_2	T_3	T_4	T_5	T_6
AluSrc1						
AluSrc2						
RegDst						
MemToReg						

3.1: Tabelle der Steuerleitungen

	T_1	T_2	T_3	T_4	T_5	T_6
AluSrc1						
AluSrc2						
RegDst						
MemToReg						

3.2: Ersatztabelle der Steuerleitungen, **ungültige Lösung streichen!**

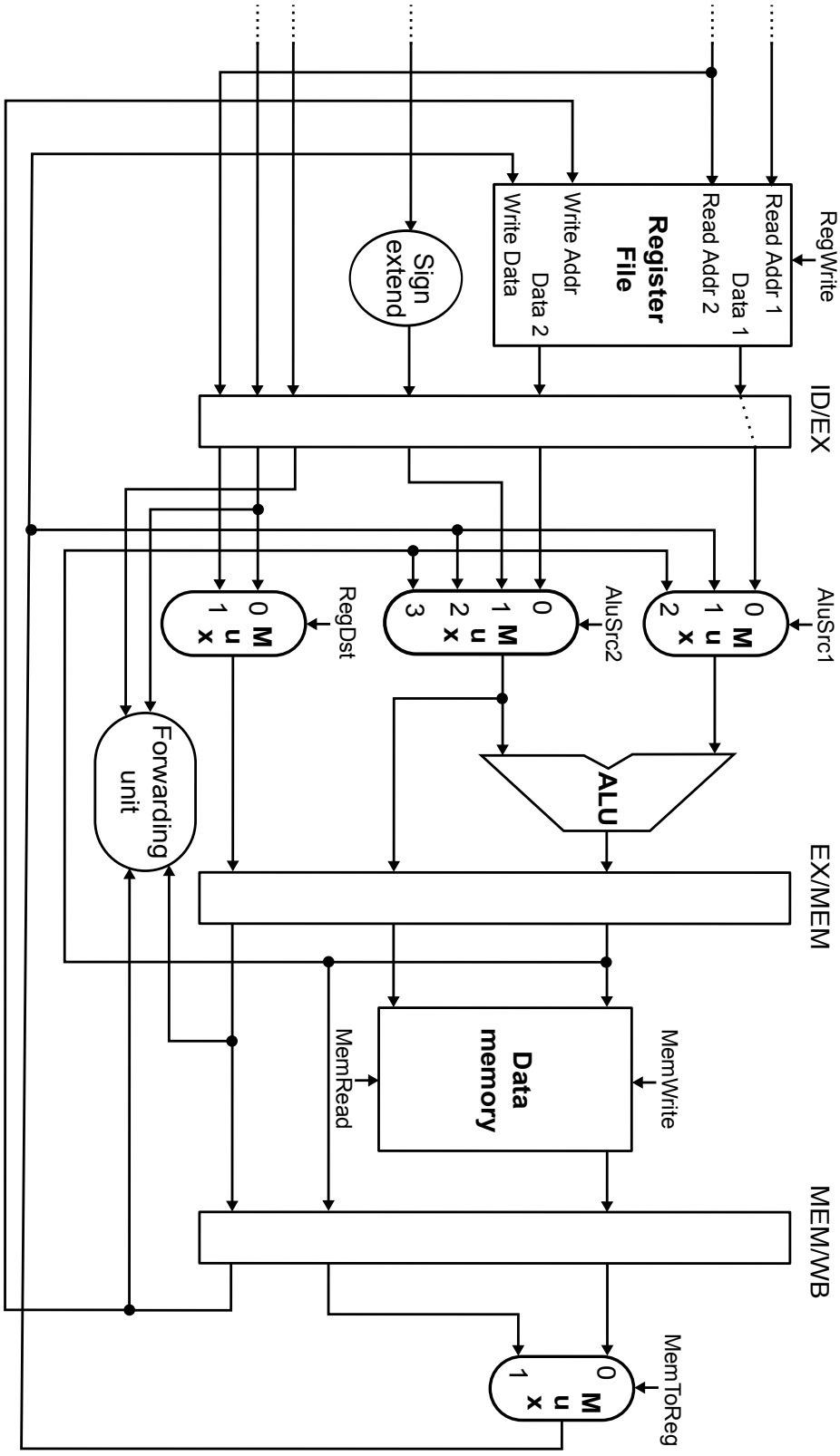


Abbildung 6: Ersatz Teil des MIPS Datenpfads, **ungültige Lösung streichen!**

f) Gegeben sei nachfolgende Befehlsfolge, bei der sich der erste Befehl zum Takt T_1 in der IF-Phase befindet:

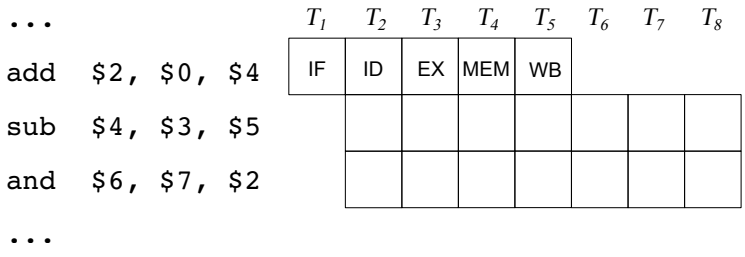


Abbildung 7: Befehlsfolge mit Daten Hazard

Vervollständigen Sie die Phasen für die zwei nächsten Befehle und zeichnen Sie das *Forwarding* in Abbildung 7 ein.

Setzen Sie die Steuerleitungen des Datenpfads aus Abbildung 4 in Tabelle 4 so, dass die Befehlsfolge korrekt ausgeführt wird. Markieren Sie alle nicht relevanten Steuerleitungen mit einem **X**.

(6 Punkte)

	T_1	T_2	T_3	T_4	T_5	T_6
AluSrc1						
AluSrc2						
RegDst						
MemToReg						

Tabelle 4: Tabelle der Steuerleitungen

	T_1	T_2	T_3	T_4	T_5	T_6
AluSrc1						
AluSrc2						
RegDst						
MemToReg						

Tabelle 5: Ersatztabelle der Steuerleitungen, **ungültige Lösung streichen!**

g) Reicht diese Art des Forwardings aus, um den Daten Hazard der nachfolgenden Befehlsabfolge aufzulösen? Begründen Sie Ihre Antwort und geben Sie, falls nötig, eine Maßnahme an. (2 Punkte)

```
...
lw  $2, 100($5)
and  $4, $2, $5
...
```

Aufgabe 5: (Cache) 20 Punkte

Um die Fehlgriffe auf einen Cache zu reduzieren, können unterschiedliche Strategien eingesetzt werden, wie zum Beispiel die Erhöhung der Assoziativität.

a) Geben Sie einen Vorteil des vollassoziativen Caches gegenüber einem Cache mit geringerer Assoziativität an. Begründen Sie Ihre Antwort kurz.

(1 Punkt)

b) Welche zwei Nachteile ergeben sich aus einer Erhöhung der Assoziativität? Begründen Sie Ihre Antwort kurz.

(1 Punkt)

c) Schätzen Sie die Fehlgriffsrate für einen vollassoziativen Cache ab, der als Befehlscache eingesetzt wird und n Rahmen enthält, wobei jeder Rahmen m Worte aufnehmen kann. Das Programm besteht aus rein arithmetischen Befehlen. Vergleichen Sie die Fehlgriffsrate mit einem direkt abgebildeten Cache mit den selben Spezifikationen und begründen Sie, für welchen Cache Sie sich entscheiden würden.

(3 Punkte)

d) Benennen Sie eine weitere Strategie zur Reduzierung der Fehlgriffe und geben Sie kurz deren Vor- und Nachteile an.

(2 Punkte)

Gehen Sie im Folgenden von einem Computer aus, der zur Beschleunigung der Speicherzugriffe auf den byteadressierten Arbeitsspeicher mit einem Adressraum von 8 GB einen Cache mit 8 Rahmen vorsieht, wobei der Datenbereich jedes Blocks 8 Worte zu je 32 Bit umfasst. Der Cache sei als 4-fach mengenassoziativer Cache (4-way-set-associative) organisiert. Als Ersetzungsstrategie wird **LRU** (Least Recently Used) eingesetzt und für das Rückschreiben in den Hauptspeicher die **write-back**-Strategie. Hierzu wird jedem Cache Rahmen ein **Dirty Bit** (D) hinzugefügt, welches auf 1 gesetzt wird, wenn Daten im entsprechenden Block geschrieben wurden, und so lange 1 bleibt, bis dieser Block zurückgeschrieben wird, sonst 0.

e) Geben Sie die Breite des Adressbusses sowie die Anzahl an Bits für Tag, Index und Offsets an.

(3 Punkte)

Adressbus: _____

Tag: _____

Index: _____

Wortoffset: _____

Byteoffset: _____

f) Gegeben sei folgende Programmabfolge, die sequentiell abgearbeitet wird (führende Nullen werden abgeschnitten).

```
t=1:  lw $t0, 0...110100010111
t=2:  sw $t1, 0...111110001001
t=3:  lw $t2, 0...110101011010
t=4:  sb $a0, 0...110100110001
t=5:  sb $t6, 0...110100010011
t=6:  lw $t3, 0...111001011100
t=7:  lw $t0, 0...111000100011
t=8:  lb $t0, 0...110001000111
t=9:  sb $v0, 0...111111001110
t=10: lw $v1, 0...111110010100
```

Der Cache sei zu Beginn leer. Geben Sie nach jedem Befehl den Inhalt des Caches an, ob es sich um einen Hit oder einen Fehlgriff handelt und ob eine Hauptspeicheraktualisierung nötig ist (Ersatztabellen auf Seite 23, **ungültige Lösungen streichen!**).

Hinweis: Geben Sie nur die Bereiche der Tabellen an, die sich ändern.

(10 Punkte)

t=1: lw $t0, 0...110100010111

Index	D	Tag
0		
1		

Hit? ☐ ja ☐ nein

Hauptspeicheraktualisierung? ☐ ja ☐ nein

t=2: sw $t1, 0...111110001001

Index	D	Tag
0		
1		

Hit? ☐ ja ☐ nein

Hauptspeicheraktualisierung? ☐ ja ☐ nein

t=3: lw $t2, 0...110101011010

Index	D	Tag
0		
1		

Hit? ☐ ja ☐ nein

Hauptspeicher-aktualisierung? ☐ ja ☐ nein

t=4: sb $a0, 0...110100110001

Index	D	Tag
0		
1		

Hit? ☐ ja ☐ nein

Hauptspeicher-aktualisierung? ☐ ja ☐ nein

t=5: sb $t6, 0...110100010011

Index	D	Tag
0		
1		

Hit? ☐ ja ☐ nein

Hauptspeicher-aktualisierung? ☐ ja ☐ nein

t=6: lw $t3, 0...111001011100

Index	D	Tag
0		
1		

Hit? ☐ ja ☐ nein

Hauptspeicher-aktualisierung? ☐ ja ☐ nein

t=7: lw $t0, 0...111000100011

Index	D	Tag
0		
1		

Hit? ☐ ja ☐ nein

Hauptspeicher-aktualisierung? ☐ ja ☐ nein

t=8: lb $t0, 0...110001000111

Index	D	Tag
0		
1		

Hit? ☐ ja ☐ nein

Hauptspeicher-aktualisierung? ☐ ja ☐ nein

t=9: sb $v0, 0...111111001110

Index	D	Tag
0		
1		

Hit? ☐ ja ☐ nein

Hauptspeicher-aktualisierung? ☐ ja ☐ nein

t=10: lw $v1, 0...111110010100

Index	D	Tag
0		
1		

Hit? ☐ ja ☐ nein

Hauptspeicher-aktualisierung? ☐ ja ☐ nein

Ersatz: t=________________

Index	D	Tag
0		
1		

Hit?	☐	ja
	☐	nein

Hauptspeicheraktualisierung?	☐	ja
	☐	nein

Ersatz: t=________________

Index	D	Tag
0		
1		

Hit?	☐	ja
	☐	nein

Hauptspeicheraktualisierung?	☐	ja
	☐	nein

Ersatz: t=________________

Index	D	Tag
0		
1		

Hit?	☐	ja
	☐	nein

Hauptspeicheraktualisierung?	☐	ja
	☐	nein

Ersatz: t=________________

Index	D	Tag
0		
1		

Hit?	☐	ja
	☐	nein

Hauptspeicheraktualisierung?	☐	ja
	☐	nein