Universität Paderborn

Institut *Elektrotechnik und Informationstechnik*

Fachgebiet *Datentechnik*

Prof. Sybille Hellebrand

Klausur

Technische Informatik / Grundlagen der Rechnerarchitektur

26. Februar 2010

Punkteverteilung						
Aufgabe	1	2	3	4	5	Σ
maximale Punkte	20	20	10	20	20	90
erreichte Punkte						

Note:	

Aufkleber

Name:	
Matrikelnummer:	
Studienrichtung:	

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

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

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

Es sind keine Hilfsmittel zugelassen!

Aufgabe 1: (Cache) 20 Punkte

a) Ein Rechner habe einen byteadressierten Arbeitsspeicher mit einer Kapazität von 128 MB. Um die Speicherzugriffe beschleunigen zu können, sei ein Cache mit 4 Rahmen zur Verwendung vorgesehen. Jeder Rahmen fasse 2 Worte zu je 32 Bit.

Nehmen Sie an, der Cache sei als 2-wege-mengenassoziativer (2-way-set-associative) Cache organisiert. Bestimmen Sie die benötigte Anzahl von Bits für Tag, Index und Offsets. (4 Punkte)

Adressbus:_____

Tag:_____

Index:_____

Wordoffset:_____

Byteoffset:_____

b) Wieviele Bit werden mindestens benötigt, um den Cache-Speicher aus a) zu implementieren? Begründen Sie Ihr Ergebnis (Rechenweg)! (3 Punkte)

c) Gegeben sei folgende MIPS-Assembler Codesequenz; das Register s0 sei mit der Adresse ...1000001000 belegt: (13 Punkte)

```
t= 1: lw $t0,  0($s0)
t= 2: lb $t1,  5($s0)
t= 3: lb $t2, 19($s0)
t= 4: lw $t3,  8($s0)
t= 5: lw $t4, 44($s0)
t= 6: lw $t5,  4($s0)
t= 7: lw $t6, 12($s0)
t= 8: lw $t7, 20($s0)
t= 9: lb $t8, 41($s0)
t=10: lb $t9, 39($s0)
t=11: lh $s1, 10($s0)
t=12: lw $s2, 24($s0)
```

(Hinweis: lw := load word, lh := load halfword, lb := load byte)

Geben Sie unter der Annahme, der Cache sei zu Programmbeginn leer, die Cachebelegung für den Cache aus a) nach jedem Lesezugriff an. Wird der Inhalt eines Cacherahmens nicht verändert, so nehmen Sie keine Eintragung vor. Verwenden Sie hierzu die auf der nächsten Seite abgebildete Tabelle. Markieren Sie alle „Hits“. Geben Sie an, welche Ersetzungsstrategie Sie verwenden und erläutern Sie diese kurz.

Die Speicherbelegung der zu lesenden Adressen ist in folgender Tabelle aufgelistet. (Hinweis: Die Adressen sind ohne führende Nullen angegeben.)

Adresse	Data
1000001000	ff
1000001001	a7
1000001010	aa
1000001011	c7
1000001100	df
1000001101	f0
1000001110	bb
1000001111	6a
1000010000	2b
1000010001	09
1000010010	73
1000010011	a8

Adresse	Data
1000010100	11
1000010101	d2
1000010110	ca
1000010111	a6
1000011000	12
1000011001	10
1000011010	bc
1000011011	de
1000011100	00
1000011101	99
1000011110	fa
1000011111	4b

Adresse	Data
1000100000	1a
1000100001	3c
1000100010	94
1000100011	cd
1000100100	42
1000100101	e9
1000100110	13
1000100111	7f
1000101000	3a
1000101001	f7
1000101010	19
1000101011	bf

Adresse	Data
1000101100	04
1000101101	31
1000101110	7a
1000101111	15
1000110000	ee
1000110001	ab
1000110010	f9
1000110011	21
1000110100	e0
1000110101	d6
1000110110	18
1000110111	22

(Hinweis: Die Ziffern in den linken Tabellenspalten der unteren Tabelle repräsentieren die vier Cacherahmen.)

t=1	Adresse=...1000001000
0	
1	
2	
3	

t=2	
0	
1	
2	
3	

t=3	
0	
1	
2	
3	

t=4	
0	
1	
2	
3	

t=5	
0	
1	
2	
3	

t=6	
0	
1	
2	
3	

t=7	
0	
1	
2	
3	

t=8	
0	
1	
2	
3	

t=9	
0	
1	
2	
3	

t=10	
0	
1	
2	
3	

t=11	
0	
1	
2	
3	

t=12	
0	
1	
2	
3	

Ersetzungsstrategie:

Adresse	Data
1000001000	ff
1000001001	a7
1000001010	aa
1000001011	c7
1000001100	df
1000001101	f0
1000001110	bb
1000001111	6a
1000010000	2b
1000010001	09
1000010010	73
1000010011	a8

Adresse	Data
1000010100	11
1000010101	d2
1000010110	ca
1000010111	a6
1000011000	12
1000011001	10
1000011010	bc
1000011011	de
1000011100	00
1000011101	99
1000011110	fa
1000011111	4b

Adresse	Data
1000100000	1a
1000100001	3c
1000100010	94
1000100011	cd
1000100100	42
1000100101	e9
1000100110	13
1000100111	7f
1000101000	3a
1000101001	f7
1000101010	19
1000101011	bf

Adresse	Data
1000101100	04
1000101101	31
1000101110	7a
1000101111	15
1000110000	ee
1000110001	ab
1000110010	f9
1000110011	21
1000110100	e0
1000110101	d6
1000110110	18
1000110111	22

↓(Ersatztabelle, ungültige Lösung streichen!)↓

t=1	Adresse=...1000001000
0	
1	
2	
3	

t=2	
0	
1	
2	
3	

t=3	
0	
1	
2	
3	

t=4	
0	
1	
2	
3	

t=5	
0	
1	
2	
3	

t=6	
0	
1	
2	
3	

t=7	
0	
1	
2	
3	

t=8	
0	
1	
2	
3	

t=9	
0	
1	
2	
3	

t=10	
0	
1	
2	
3	

t=11	
0	
1	
2	
3	

t=12	
0	
1	
2	
3	

Ersetzungsstrategie:

↑(Ersatztabelle, ungültige Lösung streichen!)↑

Aufgabe 2: (Datenpfad) 20 Punkte

Bei einer Mehrzyklenimplementierung eines MIPS Prozessors soll der R-Typ Befehl `jalr rs, rd` (Jump And Link Register) überarbeitet und als I-Typ realisiert werden. Bei dem Befehl wird ein unbedingter Sprung zur Instruktion ausgeführt, deren Adresse im Register rs gespeichert ist. Außerdem wird die Adresse der nächsten Instruktion, die nach dem `jalr` Befehl folgt, im Register rd gespeichert. (rd = PC + 4; PC = rs)
Der Daten- und Kontrollpfad für eine Mehrzyklenimplementierung ist in Abbildung 1 dargestellt.

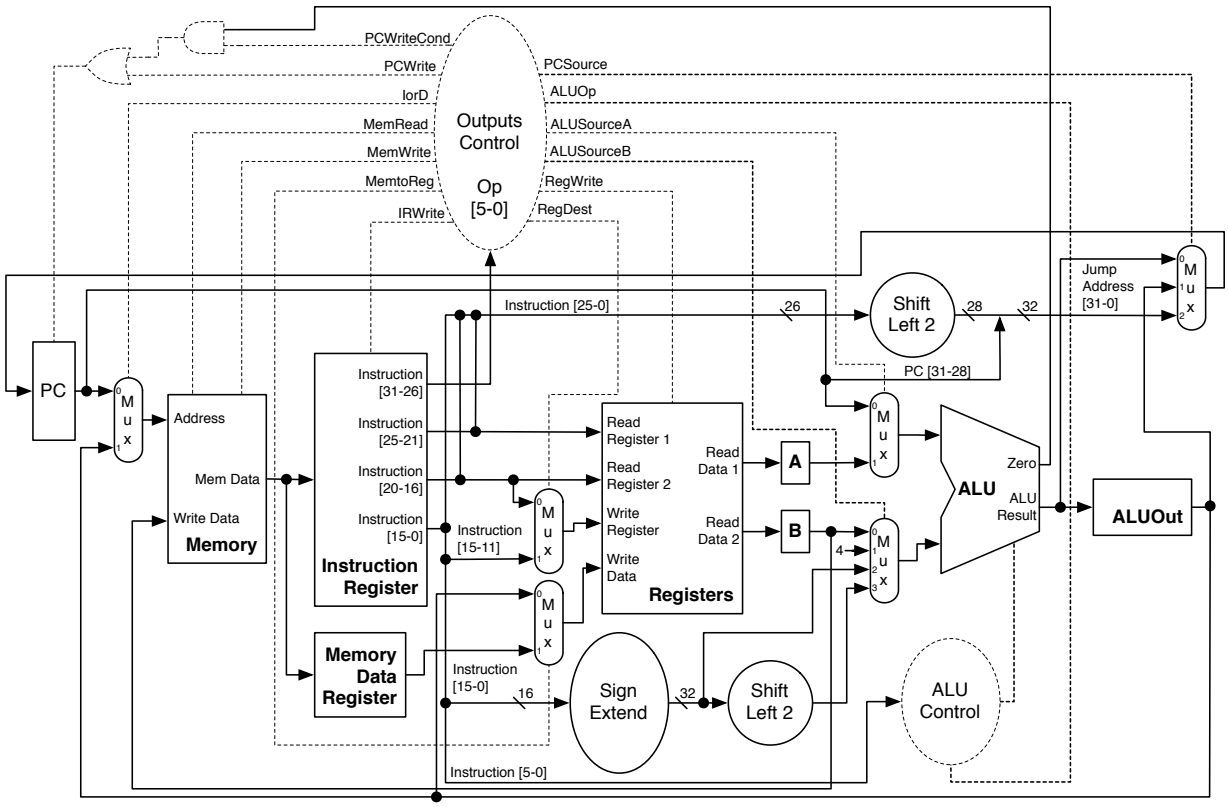


Abbildung 1: Daten- und Kontrollpfad der Mehrzyklenimplementierung

Das bisherige Instruktionsformat für `jalr` sieht folgendermaßen aus (Abb. 2):

31 26	25 21	20 16	15 11	10 6	5 0
op	rs	0	rd	0	9

Abbildung 2: Instruktionsformat von `jalr` bisher

a) Zeichnen Sie das neue Instruktionsformat für den I-Typ in Abbildung 3 ein.

(4 Punkte)

31 26	25 0
op=jalr	

Abbildung 3: „jalr“ im I-Format

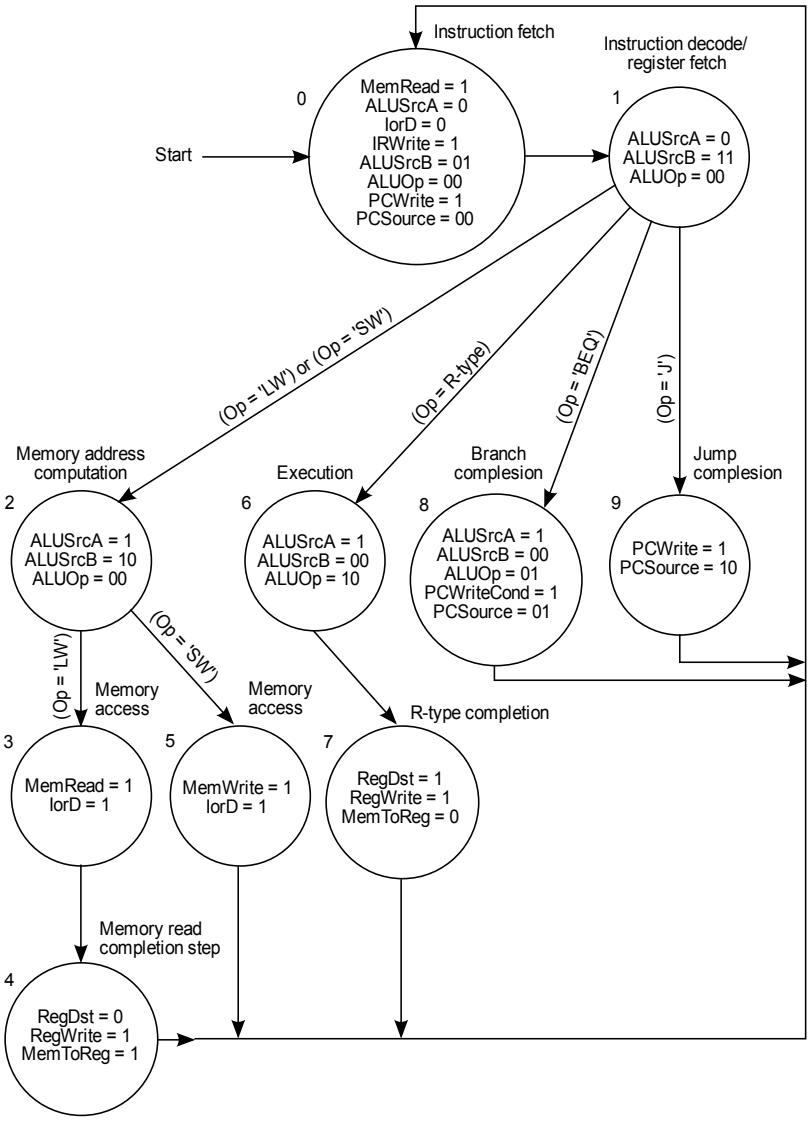
31 26	25 0
op=jalr	

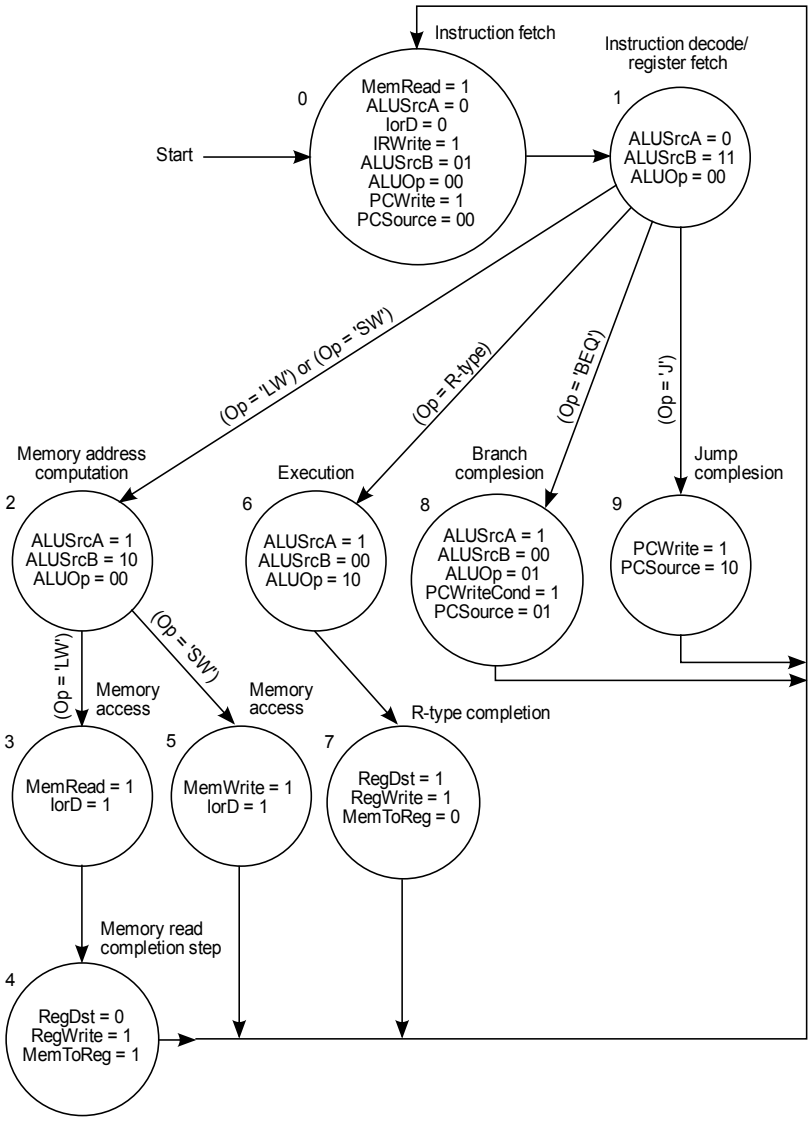
(Ersatzabbildung: ungültige Lösung streichen!)

b) Modifzieren bzw. erweitern Sie die Steuerung, so dass der Befehl `jalr` als I-Typ unterstützt wird.

(10 Punkte)

(Hinweis: Eine Änderung des Datenpfads ist nicht notwendig.)





(Ersatzabbildung: ungültige Lösung streichen!)

c) Was passiert, wenn die angegebenen Register des Befehls die gleichen Adressen haben. Funktioniert die Implementierung dann noch? Begründen Sie Ihre Antwort.

(2 Punkte)

d) In der folgenden Tabelle sehen Sie die Häufigkeiten der Befehlstypen, die innerhalb eines typischen Programms auftreten. (1 Punkt)

Tragen Sie die Anzahl der Taktzyklen für die jeweiligen Instruktionstypen in die Tabelle ein.

Instruktion	Relative Häufigkeit	Takte
R-Type	40%	
LW	20%	
SW	10%	
J	30%	

e) Durch die Konvertierung des `jalr` Befehls zum I-Typ kann idealerweise die Ausführungsdauer auf 3 Takte reduziert werden. Wieviel Prozent der R-Typ Befehle müssen `jalr` Befehle sein, damit bei einem Programm mit 10.000 Instruktionen 500 Takte eingespart werden können. Geben Sie die Berechnung an. (3 Punkte)

Aufgabe 3: (Leistungsbewertung) 10 Punkte

a) Die Firma Slow Computers Inc. möchte die verwendeten Prozessoren verbessern. Ein Benchmarkprogramm benötigt auf der alten CPU, die mit 100 MHz getaktet wird, eine Ausführungszeit von 10 Sekunden. Die Ausführungszeit auf der neuen CPU soll nur noch 6 Sekunden betragen. Das Design Team ist in der Lage eine CPU zu entwerfen, die schneller getaktet werden kann. Allerdings werden für das Programm dann 1,2 mal so viele Taktzyklen benötigt. Mit welcher Frequenz muss die neue CPU getaktet werden, damit die gewünschte Ausführungszeit von 6 Sekunden erreicht wird?

Kreuzen Sie in der Tabelle die richtige Lösung an. (2 Punkte)

- ☐ 120 MHz
- ☐ 600 MHz
- ☐ 200 MHz
- ☐ 167 MHz
- ☐ keine Lösung ist richtig

b) Bei der Konkurrenzfirma Murx Electronics wird ebenfalls an einem neuen Design gearbeitet. Für den Befehlssatz der neuen CPU werden zwei verschiedene Implementierungen diskutiert. Implementierung A erlaubt eine Zykluszeit von 10 ns und erreicht 2 CPI für ein Benchmarkprogramm. Implementierung B kommt bei einer Zykluszeit von 20 ns auf 1,2 CPI. Welches Verhältnis ergibt sich für die Ausführungszeiten des Benchmarkprogramms?

Kreuzen Sie in der Tabelle den richtigen Wert für $\frac{\text{Ausführungszeit}_B}{\text{Ausführungszeit}_A}$ an. (2 Punkte)

- ☐ 1,67
- ☐ 2
- ☐ 0,6
- ☐ 1,2
- ☐ keine Lösung ist richtig

c) Nachdem das CPU Design bei Murx Electronics feststeht, werden verschiedene Compiler getestet. Zur Bewertung stehen folgende Daten zur Verfügung:

Der Befehlssatz lässt sich in drei Klassen gemäß nachstehender Tabelle einteilen:

Befehlsklasse	CPI für die Befehlsklasse
A	1
B	2
C	3

Für den untersuchten Beispielcode erzeugen die beiden Compiler Befehlssequenzen mit den folgenden Eigenschaften:

Compiler	Anzahl der Befehle in Klasse A	B	C
I	2	1	2
II	4	1	1

Geben Sie für beide Compiler jeweils die Codelänge, die Ausführungszeit für den Beispielcode und den CPI Wert an. (3 Punkte)

Anzahl der Befehle für Compiler I: _____

Anzahl der Befehle für Compiler II: _____

Ausführungszeit (# Taktzyklen) für Compiler I: _____

Ausführungszeit (# Taktzyklen) für Compiler II: _____

CPI Wert für Compiler I: _____

CPI Wert für Compiler II: _____

d) Murx Electronics möchte außerdem eine Prozessorvariante mit Pipelining auf den Markt bringen. Der Befehlsablauf wird ähnlich wie beim MIPS-Prozessor in 5 Pipelinestufen aufgeteilt. Zur Abarbeitung einer Pipelinestufe wird 1 CPU-Zyklus benötigt. Die Analyse von Benchmarkprogrammen zeigt, dass die Wahrscheinlichkeit für einen Konflikt 25% beträgt. Bei einem Konflikt müssen im Durchschnitt 2 Wartezyklen eingefügt werden. Wie hoch ist der asymptotische Speedup (unendliche viele Befehle) der Pipelineimplementierung gegenüber einer Mehrzyklenimplementierung mit durchschnittlich 5 CPI.

Kreuzen Sie in der Tabelle die richtige Lösung an: (3 Punkte)

- ☐ 5
- ☐ 0,25
- ☐ 4
- ☐ 2,5
- ☐ keine Lösung ist richtig

Aufgabe 4: (Assembler) 20 Punkte

Gegeben ist das folgende MIPS-Assemblerprogramm.

```
01:  main:   addi    $a0, $zero, 107
02:          jal     funct
03:          add     $a0, $zero, $v0
04:          addi    $v0, $zero, 1    #Integer Ausgabe
05:          syscall                  #Wert Ausgeben
06:          addi    $v0, $zero, 10
07:          syscall                  #Programm beenden
08:  funct:  addi    $sp, $sp, -8
09:          sw      $ra, 4($sp)
10:          sw      $a0, 0($sp)
11:          add     $s0, $zero, $zero
12:          slt     $t0, $zero, $a0
13:          beq     $t0, $zero, Y
14:          srl     $a0, $a0, 2      #2x rechts schieben
15:          jal     funct
16:          lw      $a0, 0($sp)
17:          andi    $a0, $a0, 3
18:          addi    $t0, $zero, 10
19:          mul     $v0, $v0, $t0    #$v0=$v0*$t0
20:          add     $v0, $v0, $a0
21:          j       Z
22:  Y:      add     $v0, $zero, $zero
23:  Z:      lw      $a0, 0($sp)
24:          lw      $ra, 4($sp)
25:          addi    $sp, $sp, 8
26:          jr      $ra
```

a) Wie lauten die Inhalte der Register $sp, $a0, $v0, $s0, $ra nach jedem Aufruf der Funktion funct. Der Stackpointer ist beim Start des Programms mit 0x7FFFEFFC initialisiert. Der Übersicht halber können die ersten 16 Bit der Stackadresse weggelassen werden. Für Spungadressen ist es ausreichend die Zeilennummer des entsprechenden Befehls anzugeben. Machen Sie nicht initialisierte Werte mit ‚XX' kenntlich. (12 Punkte)

Aufruf 1

$sp	
$a0	
$v0	
$s0	
$ra	

Aufruf 2

$sp	
$a0	
$v0	
$s0	
$ra	

Aufruf 3

$sp	
$a0	
$v0	
$s0	
$ra	

Aufruf 4

$sp	
$a0	
$v0	
$s0	
$ra	

Aufruf 5

$sp	
$a0	
$v0	
$s0	
$ra	

Aufruf 6

$sp	
$a0	
$v0	
$s0	
$ra	

Ersatz (streichen Sie ungültige Lösungen unmissverständlich durch):

Aufruf 1

$sp	
$a0	
$v0	
$s0	
$ra	

Aufruf 2

$sp	
$a0	
$v0	
$s0	
$ra	

Aufruf 3

$sp	
$a0	
$v0	
$s0	
$ra	

Aufruf 4

$sp	
$a0	
$v0	
$s0	
$ra	

Aufruf 5

$sp	
$a0	
$v0	
$s0	
$ra	

Aufruf 6

$sp	
$a0	
$v0	
$s0	
$ra	

b) Wie lautet die Ausgabe des oben aufgeführten Programms? (2 Punkte)

_____

c) Die Instruktionen **clear, xmov, neg, subi, andi, bgt** sind Pseudocode und können von einem MIPS-Prozessor nicht nativ ausgeführt werden. Ersetzen Sie die Instruktion durch äquivalente Befehle/Befehlsfolgen. Falls Zwischenergebnisse gespeichert werden müssen, benutzen Sie $1. Des Weiteren dürfen nur folgende Befehle verwendet werden: (6 Punkte)

add	addi	and	andi
beq	bne	j	jal
jr	nor	or	ori
slt	slti	sll	srl
sub	lui		

clear $2 (setze $2 auf 0):

subi $2, $3, 7:

xmov $2, $3 (vertausche $3 und $2):

andi $2, $3, 0xDEADBEEF:

neg $2 : ($2 = -$2)

bgt $2, $3, label: (branch if greater than)

Ersatz (streichen Sie ungültige Lösungen unmissverständlich durch):

clear $2 (setze $2 auf 0):

subi $2, $3, 7:

xmov $2, $3 (vertausche $3 und $2):

andi $2, $3, 0xDEADBEEF:

neg $2 : ($2 = -$2)

bgt $2, $3, label: (branch if greater than)

Aufgabe 5: (Pipelining) 20 Punkte

a) Geben Sie Rechenzeit, Speed-Up und Effizienz einer Pipeline mit 10 homogen verteilten Stufen für 991 Instruktionen an! Die sequentielle Abarbeitung eines Befehls betrage 0.1 ms. (2 Punkte)

Rechenzeit sequentiell:

Rechenzeit Pipeline:

Speed-Up:

Effizienz:

b) Geben Sie Rechenzeit, Speed-Up und Effizienz einer Pipeline mit 10 inhomogen verteilten Stufen für 991 Instruktionen an! Die sequentielle Abarbeitung eines Befehls betrage 0.1 ms. Die Verzögerung der Stufen ist in nachfolgender Tabelle angegeben. (4 Punkte)

Stufe	0	1	2	3	4
Verzögerung (ms)	5/1000	1/100	2/100	1/100	1/100
Stufe	5	6	7	8	9
Verzögerung (ms)	1/100	1/100	1/100	1/100	5/1000

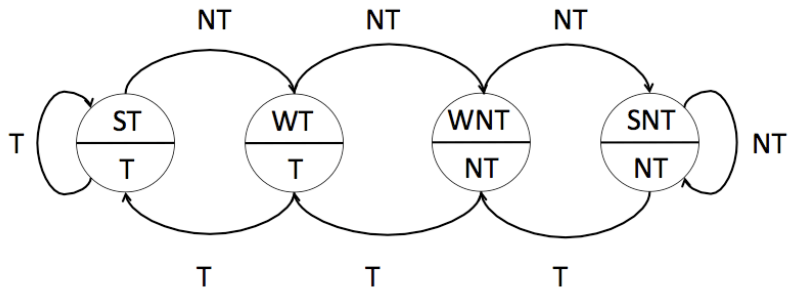
Rechenzeit sequentiell:

Rechenzeit Pipeline:

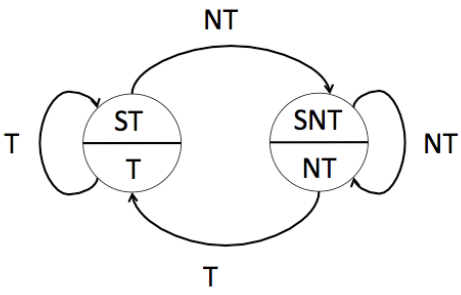
Speed-Up:

Effizienz:

c) Geben Sie für die Sprungfolge $T\ NT\ NT\ NT\ T\ T\ T\ NT\ T\ T\ T$ die Ausgabe und die Anzahl korrekt vorhergesagter Ereignisse der statischen Sprungvorhersage *always taken* und der Moore-Automaten A und B mit Eingabe- und Ausgabealphabet $\{T, NT\}$ und Zuständen $\{ST, WT, WNT, SNT\}$ bzw. $\{ST, SNT\}$ an. Die Zustandsüberführungs- und Ausgabefunktionen können den nachfolgenden Abbildungen entnommen werden. Weiterhin seien die Prädiktoren A und B mit ST initialisiert.
(6 Punkte)



A



B

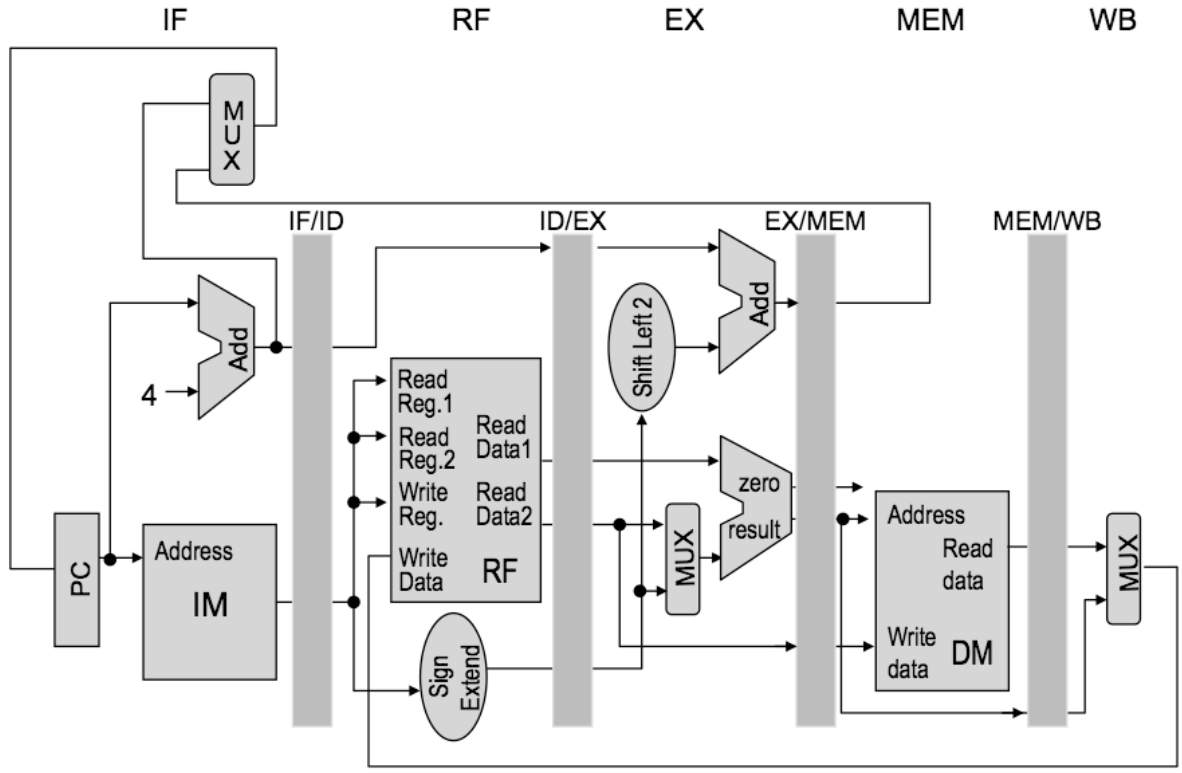
Eingabe	T	NT	NT	NT	T	T	T	NT	T	T	T
Ausgabe *always taken*:											
Ausgabe Prädiktor A:											
Ausgabe Prädiktor B:											

Anzahl der Treffer

***always taken*:**

Prädiktor A:

Prädiktor B:



d) Untersuchen Sie die Pipeline des in der Abbildung dargestellten MIPS-Prozessors! Für alle Befehle gelte die Registerreihenfolge: **destination source$_1$ source$_2$**. Beachten Sie weiter, dass MIPS eine Harvard-Struktur darstellt. Geben Sie für die Programme P1-P3 an, ob ein Daten-, Kontroll-, Strukturhazard oder kein Hazard vorliegt! Begründen Sie Ihre Entscheidungen! (6 Punkte)

P1

```
A:  add  $t1  $t2  $t3
B:  add  $t2  $t3  $t4
C:  add  $t3  $t4  $t1
D:  add  $t4  $t5  $t6
E:  add  $t5  $t6  $t7
F:  add  $t6  $t7  $t8
G:  add  $t7  $t8  $t9
```

P1:

```
P2  A:  add  $t1  $t2  $t3
    B:  lw   $t2  0x99($t3)
    C:  add  $t3  $t4  $t5
    D:  add  $t4  $t5  $t6
    E:  add  $t5  $t6  $t7
```

P2:

```
P3  A:  add  $t1  $t2  $t3
    B:  beq  $t2  $t3  F
    C:  add  $t3  $t4  $t5
    D:  add  $t4  $t5  $t6
    E:  add  $t5  $t6  $t7
    F:  add  $t6  $t7  $t8
```

P3:

e) Ändern Sie die Befehlsreihenfolge des Programms P1, sodass für die Pipeline aus c) etwaige Hazards aufgelöst werden ohne die Semantik des Programms zu verändern! Begründen Sie Ihre Entscheidungen!

Hinweis: Die Reihenfolge der Befehle kann über Label angegeben werden.

(2 Punkte)

P1: