Universität Paderborn

Institut *Elektrotechnik und Informationstechnik*

Fachgebiet *Datentechnik*

Prof. Sybille Hellebrand

Klausur

Technische Informatik / Grundlagen der Rechnerarchitektur

20. September 2016

Punkteverteilung						
Aufgabe	1	2	3	4	5	Σ
maximale Punkte	15	20	20	20	15	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) 15 Punkte

a) Für ein gegebenes CPU Design, sollen verschiedene Compiler getestet werden. Zur Bewertung stehen folgende Daten zur Verfügung:

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

Befehlsklasse	CPI für die Befehlsklasse
A	3
B	2
C	1

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

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

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: _____

b) Ein Rechner benötigt zur Ausführung dreier Programme folgende Ausführungszeiten:

Programm	Instruktionen	Ausführungszeit
1	$81000 \cdot 10^6$	9 s
2	$144000 \cdot 10^6$	12 s
3	$120000 \cdot 10^6$	10 s

Wie hoch ist die durchschnittliche MIPS-Rate des Rechners?

(4 Punkte)

c) Ein Mikroprozessorhersteller entwirft einen Mikroprozessor mit 20 Pipelinestufen. Zur Leistungsbewertung wird ein typisches Anwendungsprogramm mit 10^6 Befehlen verwendet. Die ideale Ausführungszeit einer Pipelinestufe beträgt 100 ps.

(4 Punkte)

1. Wie lange würde der gleiche Mikroprozessor ohne Pipelinestufen zur Berechnung des Anwendungsprogramms benötigen?

   Bitte kreuzen Sie an!

   - ☐ $100\,\mu s$
   - ☐ $200\,\mu s$
   - ☐ $2\,ms$
   - ☐ Keine Lösung ist richtig

2. Wie groß ist der asymptotische Speedup des Mikroprozessors mit Pipelinestufen im Vergleich zum Mikroprozessor ohne Pipelinestufen, wenn wir annehmen, dass es sich um eine ideale Pipeline handelt und keine Pipelinehindernisse (Hazards) auftreten?

   SpeedUp: _____

3. In der Realität ist eine Pipeline nicht perfekt, so treten z.B. Pipelinehindernisse (Hazards) auf. Wirken Sich diese Pipelinehindernisse auf die Befehlslatenz, den Befehlsdurchsatz oder beides aus?

   Bitte kreuzen Sie an!

   - ☐ Befehlslatenz
   - ☐ Befehlsdurchsatz
   - ☐ Beides
   - ☐ Keine Lösung ist richtig

d) Ein Rechner benötigt zur Ausführung eines Programms 200 s. Durch die Verwendung eines neuen Prozessors können Fließkomma Instruktionen sechs mal schneller verarbeitet werden. Wie groß muss der Anteil der Fließkomma Instruktionen sein, damit ein Speedup von zwei erzielt werden kann?

(4 Punkte)

Bitte kreuzen Sie an!

- ☐ $\frac{3}{5}$
- ☐ $\frac{4}{5}$
- ☐ $\frac{1}{3}$
- ☐ $\frac{3}{4}$
- ☐ Keine Lösung ist richtig

Aufgabe 2: (Cache) 20 Punkte

Gehen Sie von einem Computer aus, der zur Beschleunigung der Speicherzugriffe auf den byteadressierten Arbeitsspeicher mit einem Adressraum von 1 GB einen Cache mit 8 Rahmen vorsieht, wobei der Datenbereich jedes Blocks 2 Worte zu je 32 Bit umfasst. Der Cache sei als 2-fach mengenassoziativer Cache (2-way-set-associative) organisiert.

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

(3 Punkte)

Adressbus: _____

Tag: _____

Index: _____

Wortoffset: _____

Byteoffset: _____

b) Wie viel Speicher wird für die Implementierung eines solchen Cache (inkl. Overhead) benötigt? Geben Sie den Rechenweg und das Ergebnis in Byte an.

(3 Punkte)

Cache-Größe (in Byte): _____

c) Der Prozessor lädt nun sequentiell die Daten der folgenden Byteadressen (führende Nullen werden nicht mit angegeben):

```
t=10:  0x45
t=11:  0x1d
t=12:  0xf7
t=13:  0xe5
t=14:  0xe8
t=15:  0x04
t=16:  0xf0
t=17:  0x95
```

Als Ersetzungsstrategie wird **FIFO** (First In First Out) eingesetzt. Gehen Sie von folgendem Zustand des Cache zum Zeitpunkt $t = 9$ aus (die Spalte t gibt an, zu welchem Zeitpunkt die Daten in den Cache übertragen wurden):

Index	**V**	**t**	**Tag**	**Daten**
0	1	7	...010	MEM[0x40-0x47]
	1	6	...101	MEM[0xa0-0xa7]
1	0	1	...110	MEM [0x00-0x07]
	0	3	...111	MEM [0xe8-0xef]
2	0	6	...010	MEM [0x00-0x07]
	0	2	...101	MEM [0x00-0x07]
3	1	4	...000	MEM [0x18-0x1f]
	0	7	...111	MEM [0x00-0x07]

Geben Sie nach jedem Speicherzugriff den Zustand des Caches an. Füllen Sie hierzu die nächsten Zeilen aus. Geben Sie zum Schluss die Hitrate an.
Hinweis: Beachten Sie das Valid-Bit (V)!

(12 Punkte)

	Index	**V**	**t**	**Tag**	**Daten**
t=10: 0x45	_____	_____	_____	_____	MEM[_____]
		_____	_____	_____	MEM[_____]

t=11:
0x1d

Index	V	t	Tag	Daten
_____	_____	_____	_____	MEM[_____]
	_____	_____	_____	MEM[_____]

t=12:
0xf7

Index	V	t	Tag	Daten
_____	_____	_____	_____	MEM[_____]
	_____	_____	_____	MEM[_____]

t=13:
0xe5

Index	V	t	Tag	Daten
_____	_____	_____	_____	MEM[_____]
	_____	_____	_____	MEM[_____]

t=14:
0xe8

Index	V	t	Tag	Daten
_____	_____	_____	_____	MEM[_____]
	_____	_____	_____	MEM[_____]

t=15:
0x04

Index	V	t	Tag	Daten
_____	_____	_____	_____	MEM[_____]
	_____	_____	_____	MEM[_____]

t=16:
0xf0

Index	V	t	Tag	Daten
_____	_____	_____	_____	MEM[_____]
	_____	_____	_____	MEM[_____]

t=17:
0x95

Index	V	t	Tag	Daten
_____	_____	_____	_____	MEM[_____]
	_____	_____	_____	MEM[_____]

Hitrate: _____

_____

Index	V	t	Tag	Daten
_____	_____	_____	_____	_____
	_____	_____	_____	_____

Ersatzzeile, **ungültige Lösung streichen!**

_____

Index	V	t	Tag	Daten
_____	_____	_____	_____	_____
	_____	_____	_____	_____

Ersatzzeile, **ungültige Lösung streichen!**

	Index	V	t	Tag	Daten
_____	_____	_____	_____	_____	_____
		_____	_____	_____	_____

Ersatzzeile, **ungültige Lösung streichen!**

	Index	V	t	Tag	Daten
_____	_____	_____	_____	_____	_____
		_____	_____	_____	_____

Ersatzzeile, **ungültige Lösung streichen!**

d) Wie hoch ist die durchschnittliche Zugriffszeit, ausgehend von der Hitrate des Cache aus Aufgabenteil **c)**, wenn der Zugriff auf den Cache 10ns und der Zugriff auf den Hauptspeicher 100ns beträgt? Geben Sie auch den Rechenweg an.

(2 Punkte)

Durchschnittliche Zugriffszeit: _____

Aufgabe 3: (Assembler) 20 Punkte

a) Vervollständigen Sie folgendes Programm, das als Parameter in `$a0` eine Zahl und in `$a1` eine Speicheradresse übergeben bekommt. (4 Punkte)

```
01      addi   $t0, $a1,   0   # initialize pointer into v1
02      addi   $t1, $zero, 4   #
03      mul    $t1, $t1, $a0   # t1 = 4 * dim
04      add    $t1, $t1, $a1   # initialize pointer into v2
05      addi   $t8, $zero, 0   # initialize partial sum
06      addi   $t9, $zero, 0   # initialize loop counter

loop:                          # load word from address stored
07      ______________________ # in $t0 into register $t2
                               # load word from address stored
08      ______________________ # in $t1 into register $t3
09      mul    $t4, $t2, $t3   # multiply values

10      ______________________ # add result to partial sum
11      addi   $t0, $t0, 4     # next value in v1
12      addi   $t1, $t1, 4     # next value in v2

13      ______________________ # increment loop counter

14      blt_________________   # loop if counter < dim
15      addi   $v0, $t8, 0     # save result
```

b) Das Programms enthält Pseudoinstruktionen, zum Beispiel in Zeile `14`: `blt` (branch on less than). Drücken Sie Zeile `14` in regulären MIPS Instruktionen aus. Sie dürfen dafür die Befehle `slt`, `beq`, `bne` und das Hilfsregister `$at` für temporäre Hilfsvariablen verwenden. (3 Punkte)

c) Gegeben sei folgender Inhalt der Register und des Speichers eines MIPS Systems. Wenn das Programm aus **a)** in diesem Zustand aufgerufen wird, was berechnet es dann? Füllen Sie untenstehende Tabelle aus, indem Sie jeweils eine neue Spalte ausfüllen, wenn der Programmablauf Zeile `7` (Marke *loop*) erreicht, und eine extra Spalte nach Ablauf des Programms (nach Zeile `15`)! Bei den Speicheradressen können Sie das Präfix `1001` weglassen, aus `0x10010004` würde beispielsweise `0x0004`. (12 Punkte)

Register	**Wert**
`$v0`	42
`$v1`	-1
`$a0`	3
`$a1`	`0x1001000C`
`$a2`	123
`$a3`	8128
`$t0`	0
`$t1`	12
`$t2`	81
`$t3`	9999
`$t4`	-245
`$t5`	4192
`$t6`	`0x10010004`
`$t7`	`0x10010028`
`$t8`	13
`$t9`	-1

Speicher-adresse	**Wert**
`0x10010000`	24
`0x10010004`	255
`0x10010008`	3
`0x1001000C`	2
`0x10010010`	123
`0x10010014`	5
`0x10010018`	5
`0x1001001C`	0
`0x10010020`	5
`0x10010024`	1
`0x10010028`	25
`0x1001002C`	0
`0x10010030`	`0x10010000`
`0x10010034`	99374
`0x10010038`	0
`0x1001003C`	16535

Register- und Speicherinhalt zu Beginn der Berechnung.

Register	Wert	Wert	...			
`$v0`	42					
`$t0`	0					
`$t1`	12					
`$t2`	81					
`$t3`	9999					
`$t8`	13					
`$t9`	-1					

Tabelle für Registerwerte.

d) Was berechnet die Funktion in **c)**? (1 Punkt)

Register	Wert	Wert	...			
`$v0`	42					
`$t0`	0					
`$t1`	12					
`$t2`	81					
`$t3`	9999					
`$t8`	13					
`$t9`	-1					

Ersatztabelle für Registerwerte, **ungültige Lösungen streichen!**

Aufgabe 4: (Datenpfad) 20 Punkte

Bei der Mehrzyklenimplementierung eines MIPS Prozessors (Abbildung 3) soll der neue I-Typ Befehl **Jump And Link Register Immediate** (`jalri $rs(imm)`) realisiert werden. Bei dem Befehl wird ein unbedingter Sprung zur Instruktion ausgeführt, deren Basisadresse im Register $rs gespeichert ist und durch einen Offset aus dem Immediate Wert ergänzt wird(PC = rs+imm). Außerdem wird die Rücksprung-Adresse im Register $ra ($31) gespeichert. (ra = PC + 4)

a) Zeichnen Sie das neue Instruktionsformat für den I-Typ in Abbildung 1 ein. Abschnitte im Instruktionsformat, die für die nicht benötigt werden, kennzeichnen Sie mit einem X.

(3 Punkte)

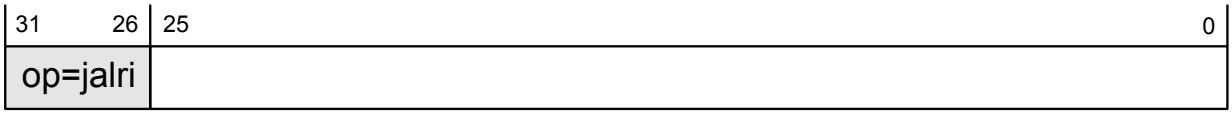

Abbildung 1: neues Instruktionsformat

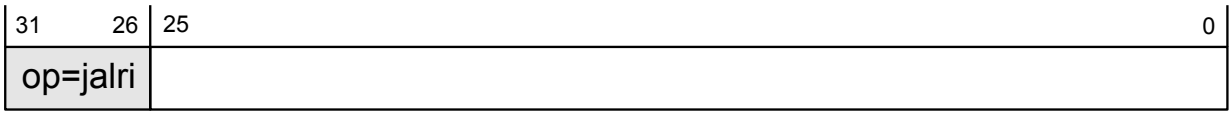

Abbildung 2: Ersatz, **ungültige Lösung streichen!**

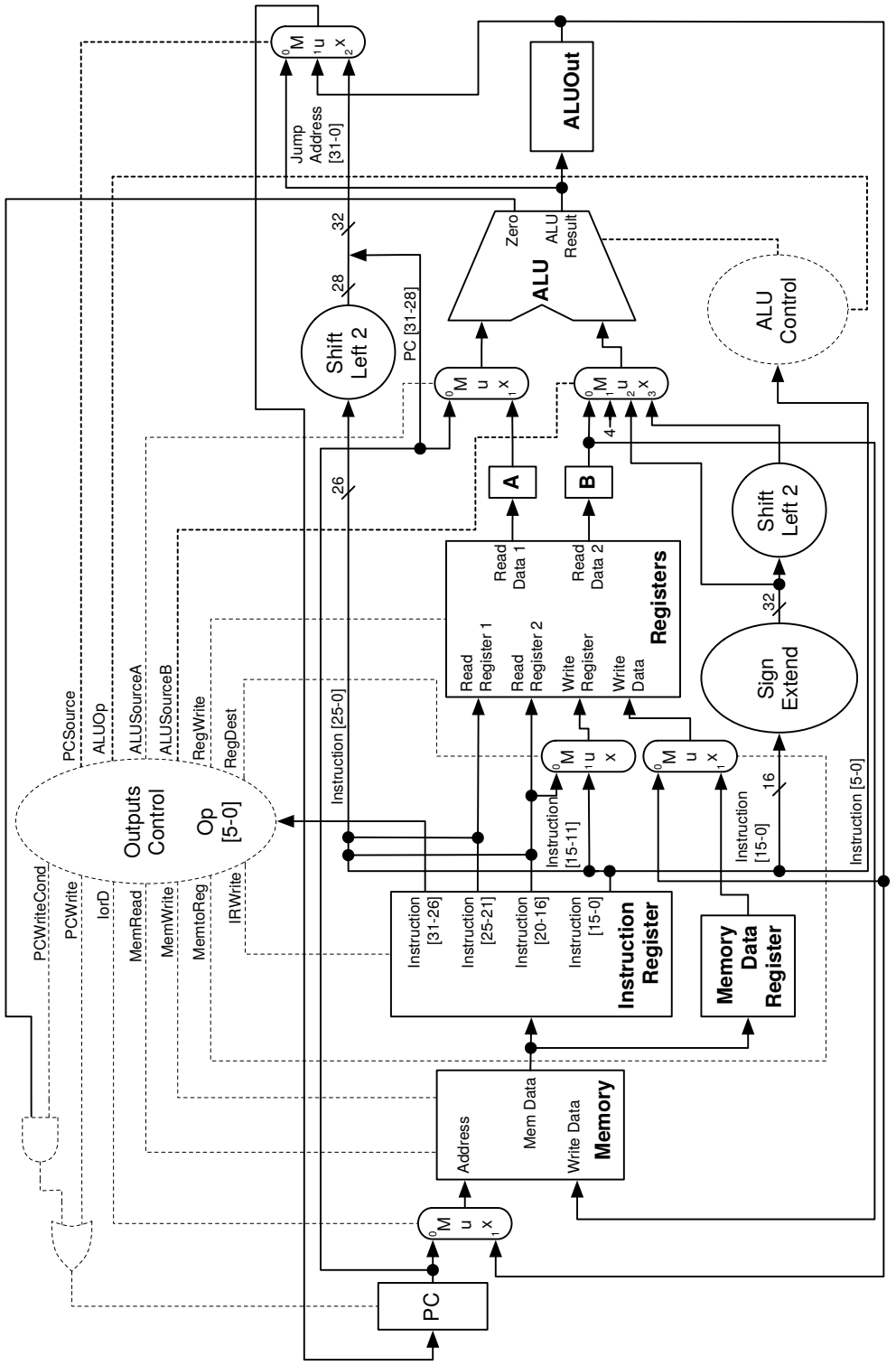

Abbildung 3: Daten- und Kontrollpfad der Mehrzyklenimplementierung, Ersatzbild auf Seite 19.

b) Modifizieren Sie, wenn nötig, den Daten- und Kontrollpfad in Abbildung 3 so, dass der neue Befehl `jalri` realsisiert werden kann und begründen Sie Ihre Entscheidung. **Ohne Begründung gibt es keine Punkte!**

(3 Punkte)

c) Modifzieren bzw. erweitern Sie die Steuerung in Abbildung 4, so dass der Befehl `jalri` unterstützt wird.

(10 Punkte)

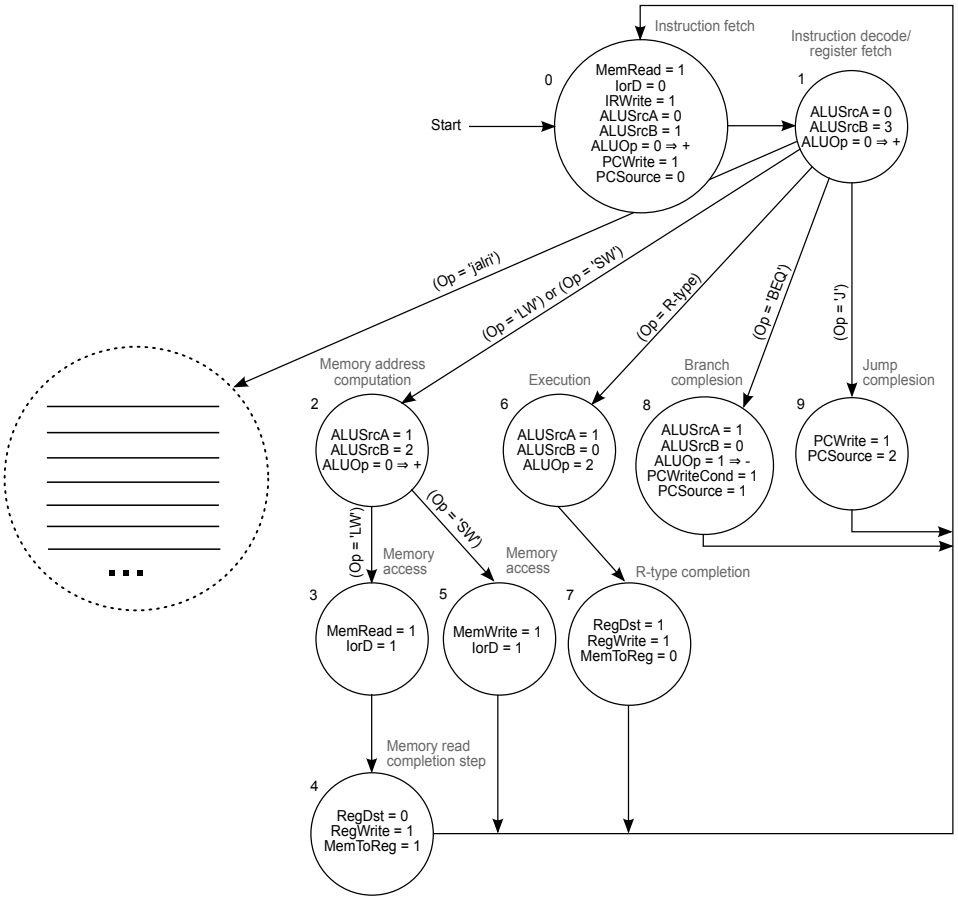

Abbildung 4: Steuerung als endlicher Automat, Ersatzbild auf Seite 20.

In der folgenden Tabelle sehen Sie die Häufigkeiten der Befehlstypen, die innerhalb eines typischen Programms auftreten.

Instruktion	Relative Häufigkeit	Takte
R-Type	20%	4
LW	20%	5
SW	10%	4
J	30%	3
other	20%	4

In einem Programm mit 10.000 Instruktionen sind von den 20 % R-Type Befehlen 25% `jalr` Befehle, welche zusätzlich ein Offset auf die Sprungadresse im Register $j benötigen:

```
...
addi $t1, $j, offset
jalr $t1
...
```

d) Wie viele Takte können durch den neuen Befehl `jalri` eingespart werden? Hinweis: Der addi Befehl ist bei der Instruktion **other** einzuordnen.

(4 Punkte)

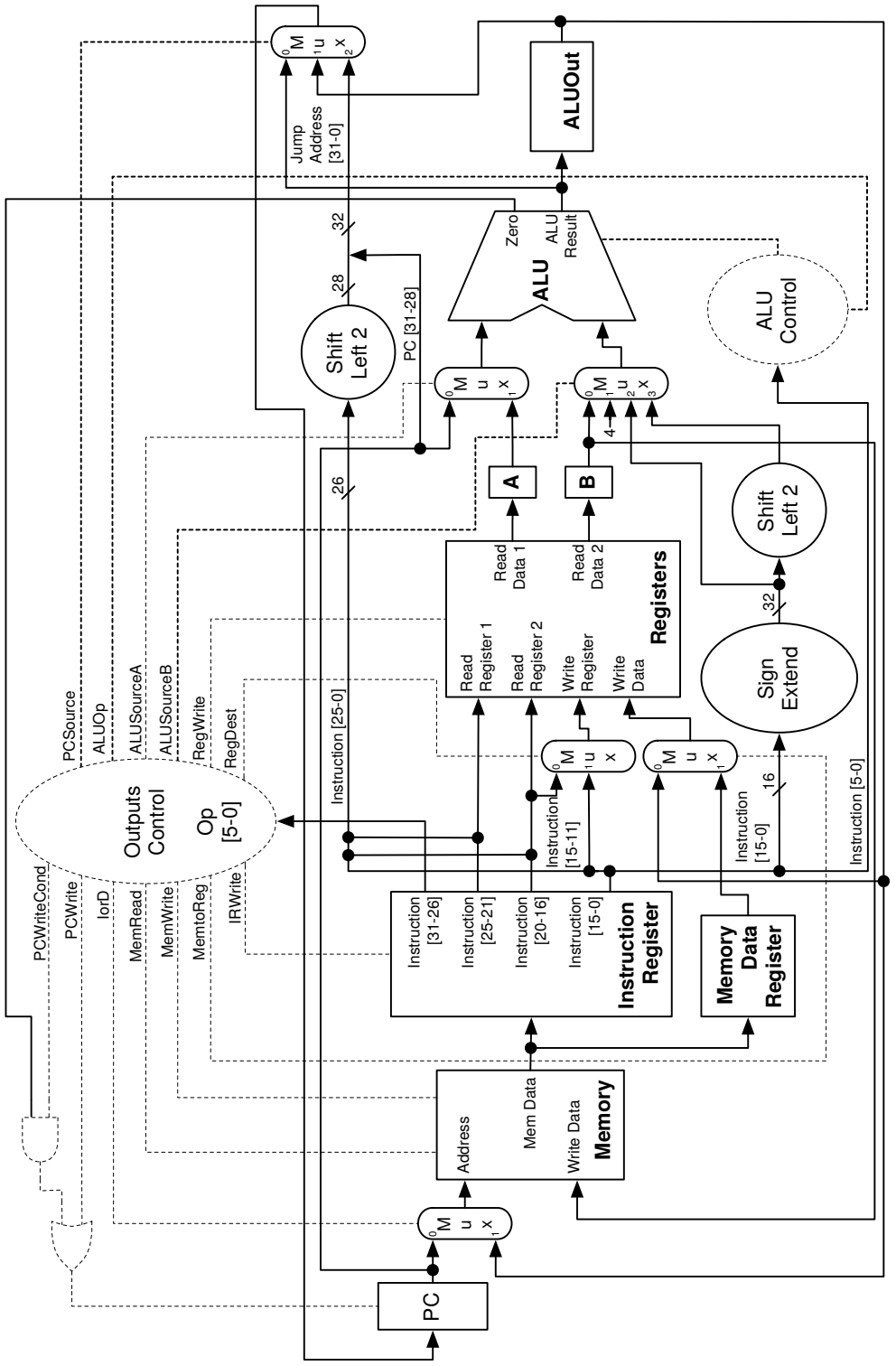

Abbildung 5: Ersatz Mehrzyklenimplementierung des Daten- und Kontrollpfads, **ungültige Lösung streichen!**

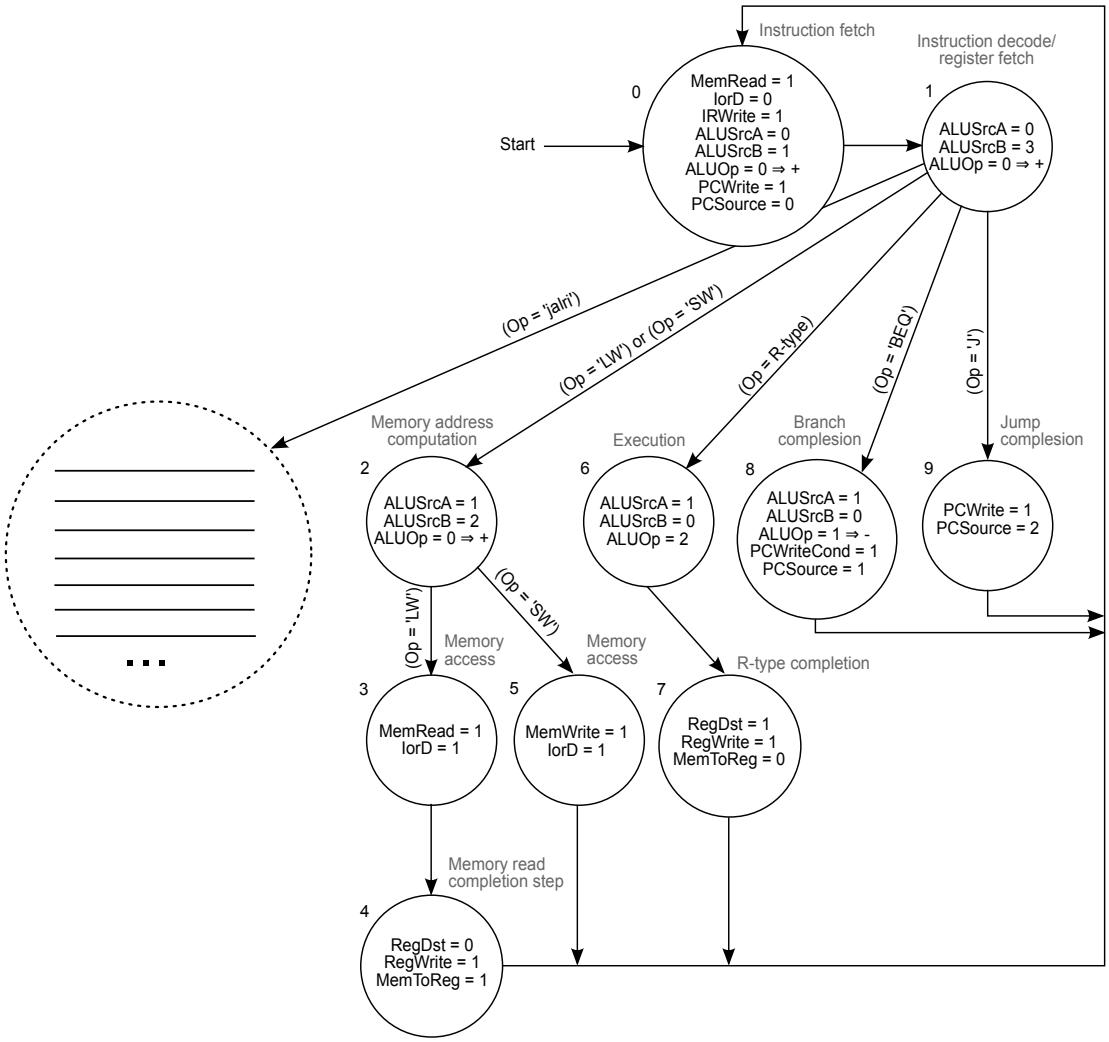

Abbildung 6: Ersatz der Steuerung, **ungültige Lösung streichen!**

Aufgabe 5: (Pipelining) 15 Punkte

Ein Prozessor verwendet eine aus der Vorlesung bekannte 5-stufige MIPS Pipeline (IF, ID, EX, ME, WB). Es wird kein Forwarding unterstützt. Das folgende Assemblerprogramm wird auf dem Prozessor ausgeführt. Gehen Sie davon aus, dass die Sprungbedingung in Zeile 3 nicht erfüllt wird.

```
1       lw  $10, 4($8)
2       add $3,  $4,  $10
3       beq $3,  $10, L1
4       add $4,  $3,  $8
5       sll $8,  $8,  $2
6   L1: add $5,  $4,  $3
7       sw  $9,  0($4)
```

a) Lösen Sie die Konflikte, indem die Pipeline entsprechend angehalten wird. Erstellen Sie ein Diagramm, aus dem die Pipelinebelegung ersichtlich wird. Tritt bei einer Codezeile ein Hazard auf, tragen Sie in Tabelle 1 ein **X** bei dem entsprechenden Hazard ein. (6 Punkte)

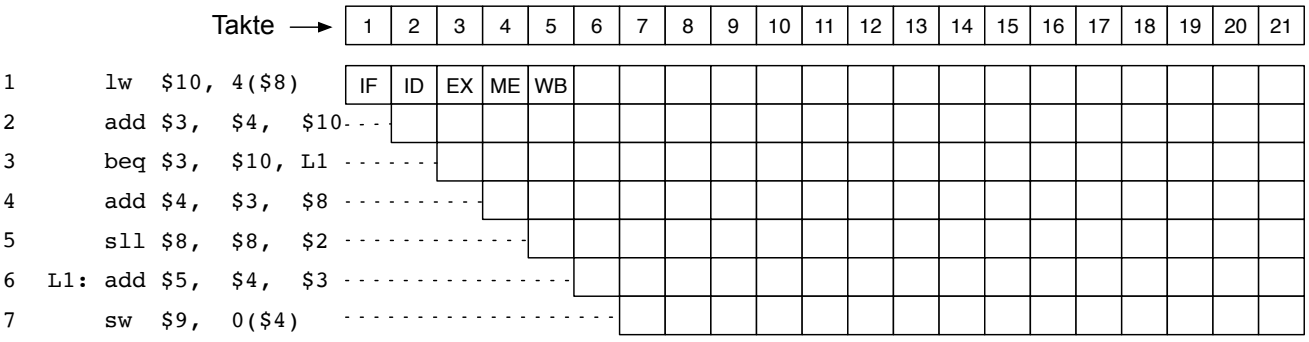

Abbildung 7: Pipelinebelegung zu **a)**

Tabelle 1: Hazards bei Bearbeitung von **a)**

Codezeile	1	2	3	4	5	6	7
Daten-Hazard							
Kontroll-Hazard							
Struktur-Hazard							

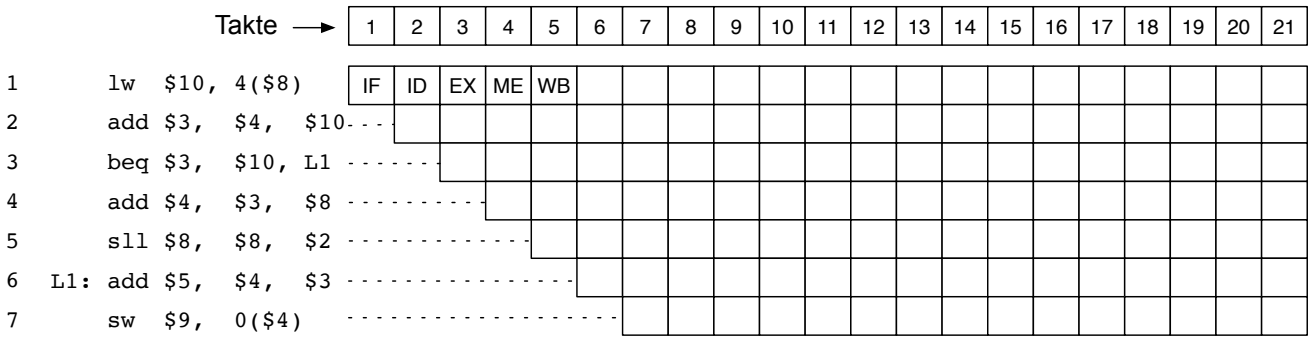

Abbildung 8: Ersatz Pipelinebelegung zu **a)**, **ungültige Lösung streichen!**

b) Der Prozessor aus **a)** unterstützt jetzt zusätzlich jedes mögliche Forwarding. Im Fall eines durch das Forwarding nicht auflösbaren Konflikts wird die Pipeline angehalten. Geben Sie die Pipelinebelegung mithilfe des folgenden Diagramms an und machen Sie das Forwarding mit einem Pfeil kenntlich. Tragen Sie auftretende Hazards in Tabelle 2 ein. (5 Punkte)

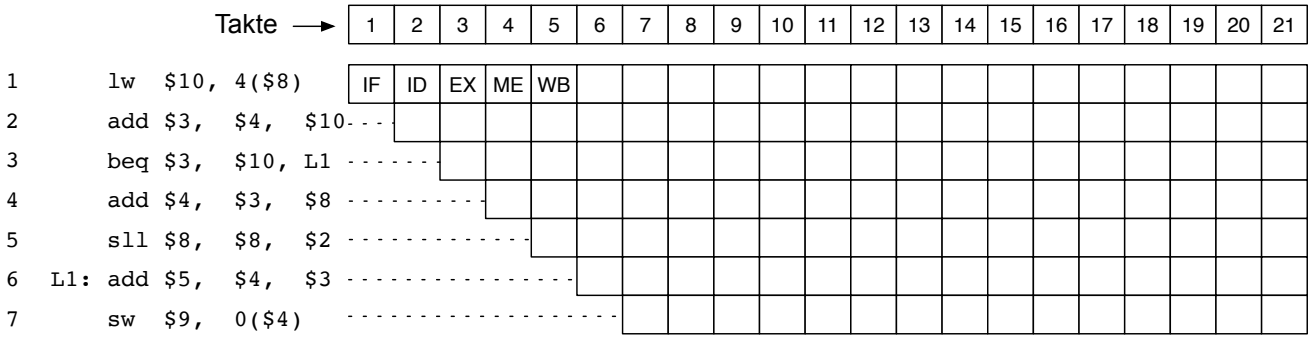

Abbildung 9: Pipelinebelegung zu **b)**

Tabelle 2: Hazards bei Bearbeitung von **b)**

Codezeile	1	2	3	4	5	6	7
Daten-Hazard							
Kontroll-Hazard							
Struktur-Hazard							

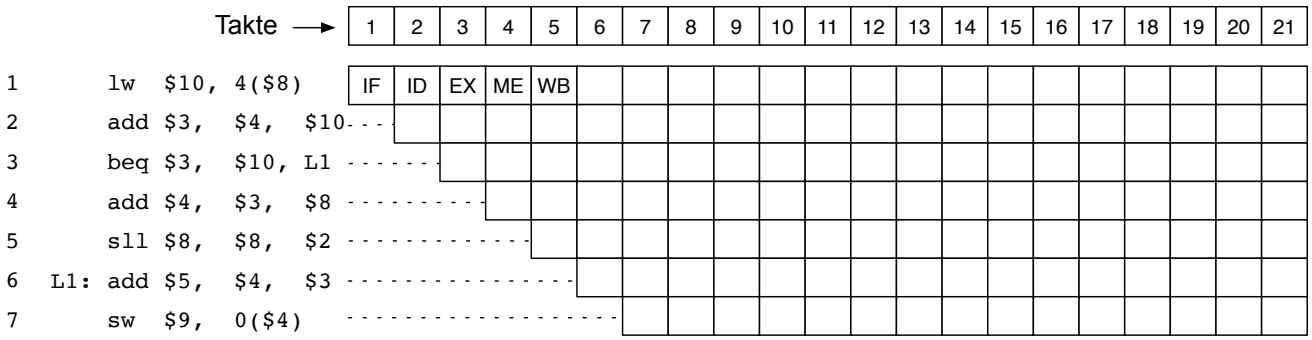

Abbildung 10: Ersatz Pipelinebelegung zu **b)**, **ungültige Lösung streichen!**

c) Berechnen Sie den Speedup des in **b)** modifizierten Prozessors. (1 Punkt)

d) Nennen Sie 3 Möglichkeiten, Kontroll-Hazards aufzulösen. (3 Punkte)

- _____
- _____
- _____