Universität Paderborn
Institut *Elektrotechnik und Informationstechnik*
Fachgebiet *Datentechnik*
Prof. Sybille Hellebrand

Klausur

Grundlagen der Rechnerarchitektur

24. Februar 2014

Punkteverteilung						
Aufgabe	1	2	3	4	5	Σ
maximale Punkte	15	18	21	16	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) 15 Punkte

a) Die Firma Limited Performance Corp. 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 3 Sekunden betragen. Das Design Team ist in der Lage eine CPU zu entwerfen, die schneller getaktet werden kann. Allerdings werden für das Programm dann 1,8 mal so viele Taktzyklen benötigt. Mit welcher Frequenz muss die neue CPU getaktet werden, damit die gewünschte Ausführungszeit von 3 Sekunden erreicht wird?

Bitte kreuzen Sie alle richtigen Antworten an.

(3 Punkte)

- ☐ 60 MHz
- ☐ 180 MHz
- ☐ 540 MHz
- ☐ 600 MHz
- ☐ keine Lösung ist richtig

b) Der Chef-Ingenieur von Limited Performance schlägt eine Verbesserung der Gleitkomma-Einheit vor. Bisher benötigen einfache Befehle (alle Befehle außer Gleitkommabefehlen) im Durchschnitt 2 Taktzyklen und Gleitkommabefehle 10 Taktzyklen. Durch die Verbesserung werden für Gleitkommabefehle nur noch 6 Taktzyklen benötigt, einfache Befehle brauchen dann jedoch 3 Taktzyklen. Außerdem muss die Zykluszeit von 40 ns auf 60 ns erhöht werden. Wie groß muss der Anteil der Gleitkommabefehle mindestens sein, damit sich die Verbesserung lohnt (d.h. insgesamt ein Speedup > 1 erreicht wird)?

Bitte kreuzen Sie alle richtigen Antworten an.

(4 Punkte)

- ☐ größer als 4/7
- ☐ größer als 5/7
- ☐ größer als 1/2
- ☐ größer als 12/25
- ☐ keine Lösung ist richtig

c) 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?

Bitte kreuzen Sie alle richtigen Antworten für $\frac{\text{Ausführungszeit}_B}{\text{Ausführungszeit}_A}$ an.

(2 Punkte)

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

d) Murx Electronics möchte außerdem eine Prozessor-Variante 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 muss im Durchschnitt 1 Wartezyklus eingefügt werden. Wie hoch ist der asymptotische Speedup (unendliche viele Befehle) der Pipeline-Implementierung gegenüber einer Mehrzyklen-Implementierung?

Bitte kreuzen Sie alle richtigen Antworten an.

(3 Punkte)

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

e) Die Konkurrenzfirma Murx Electronics hat eine neue CPU entwickelt, die mit 100 MHz getaktet werden kann. Die Befehle lassen sich grob in drei Klassen einteilen (vgl. Tabelle).

Befehlsklasse	CPI für die Befehlsklasse
A	1
B	2
C	3

Bei einem Testlauf mit einem Beispielprogramm werden 70 MIPS erreicht. In der folgenden Tabelle sind Benchmarkprogramme mit verschiedenen Verteilungen der Befehle auf die einzelnen Befehlsklassen gezeigt. Welche davon können für den Testlauf verwendet worden sein?

Bitte kreuzen Sie alle richtigen Antworten an.

(3 Punkte)

	Programm	Anzahl der Befehle in Klasse		
		A	B	C
☐	I	$4 * 10^9$	$3 * 10^9$	0
☐	II	$10 * 10^9$	$1 * 10^9$	$1 * 10^9$
☐	III	$9 * 10^9$	$3 * 10^9$	0
☐	IV	$10 * 10^9$	$2 * 10^9$	$2 * 10^9$
☐	keines der Programme wurde verwendet			

Aufgabe 2: (Pipelining) 18 Punkte

Gegeben sei der aus der Vorlesung bekannte MIPS-Prozessor mit 5-stufiger Pipeline (IF, ID, EX, MEM, WB) und einer statischen Sprungvorhersage **"branch not taken"**. Die Pipeline nutzt alle Möglichkeiten für Forwarding aus.

a) Folgendes Programm soll ausgeführt werden

```
1:  main:   addi $t3,   $zero, 4
2:          addi $t1,   $zero, 6
3:          addi $t4,   $zero, 2
4:          add  $t0,   $t3,   $t4
5:          lw   $t3,   0($a0)
6:          add  $t7,   $t0,   $t3
7:          sub  $s1,   $t0,   $t1
8:          beq  $s1,   $zero, func
9:          addi $s1,   $s1,   1
10:         j exit
11: func:   ...
```

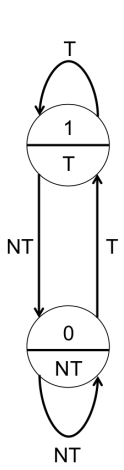
Liegt die Sprungvorhersage des Prozessors hier richtig? Begründen Sie kurz Ihre Antwort.

(2 Punkte)

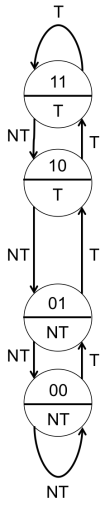
b) Kann das Programm in Aufgabenteil **a)** durch Umsortieren der Befehle schneller ausgeführt werden? Begründen Sie kurz Ihre Antwort und geben Sie, falls möglich, eine bessere Anordnung an.

(3 Punkte)

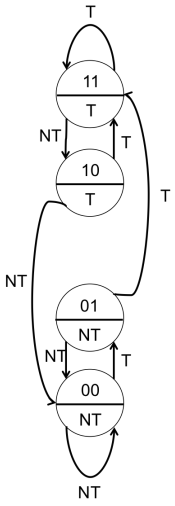
c) Zur Effizienzsteigerung sollen dynamische Prädiktoren eingesetzt werden. Hierzu stehen die Prädiktoren "1-bit Prädiktor", "2-bit Prädiktor weak" und "2-bit Prädiktor strong" zur Verfügung, siehe Abbildung 1.



(a) 1-bit Prädiktor



(b) 2-bit Prädiktor weak



(c) 2-bit Prädiktor strong

Abbildung 1: Dynamische Prädiktoren

Für die Sprungfolge NT-T-NT-NT-T-T-T-NT sollen die Präditkoren miteinander verglichen werden. Füllen Sie dazu folgende Tabelle aus und geben Sie die Vorhersagegenauigkeit in Prozent an (Ersatz- Tabellen ab Seite 9 , **ungültige Lösung streichen!**).

(13 Punkte)

1-bit Prädiktor					
Zeitpunkt	Sprung	Aktueller Zustand	Vorhersage	Richtig oder Falsch?	Folgender Zustand
0	NT	0			
1	T				
2	NT				
3	NT				
4	T				
5	T				
6	T				
7	NT				

2-bit Prädiktor weak					
Zeitpunkt	Sprung	Aktueller Zustand	Vorhersage	Richtig oder Falsch?	Folgender Zustand
0	NT	01			
1	T				
2	NT				
3	NT				
4	T				
5	T				
6	T				
7	NT				

2-bit Prädiktor strong					
Zeitpunkt	Sprung	Aktueller Zustand	Vorhersage	Richtig oder Falsch?	Folgender Zustand
0	NT	01			
1	T				
2	NT				
3	NT				
4	T				
5	T				
6	T				
7	NT				

Vorhersagegenauigkeit 1-bit Prädiktor: _____

Vorhersagegenauigkeit 2-bit Prädiktor weak: _____

Vorhersagegenauigkeit 2-bit Prädiktor strong: _____

Ersatz- Tabellen, **ungültige Lösung streichen!**

1-bit Prädiktor					
Zeitpunkt	Sprung	Aktueller Zustand	Vorhersage	Richtig oder Falsch?	Folgender Zustand
0	NT	0			
1	T				
2	NT				
3	NT				
4	T				
5	T				
6	T				
7	NT				

2-bit Prädiktor weak					
Zeitpunkt	Sprung	Aktueller Zustand	Vorhersage	Richtig oder Falsch?	Folgender Zustand
0	NT	01			
1	T				
2	NT				
3	NT				
4	T				
5	T				
6	T				
7	NT				

2-bit Prädiktor strong					
Zeitpunkt	Sprung	Aktueller Zustand	Vorhersage	Richtig oder Falsch?	Folgender Zustand
0	NT	01			
1	T				
2	NT				
3	NT				
4	T				
5	T				
6	T				
7	NT				

Vorhersagegenauigkeit 1-bit Prädiktor: _____

Vorhersagegenauigkeit 2-bit Prädiktor weak: _____

Vorhersagegenauigkeit 2-bit Prädiktor strong: _____

Aufgabe 3: (Assembler) 21 Punkte

Gegeben ist die folgende MIPS-Assembler-Funktion. Die Funktion bekommt 3 Werte als Parameter übergeben (eine Adresse für ein *aufsteigend sortiertes* Feld mit Ganzzahlen ($a0), die Anzahl der Elemente im Feld ($a1) und eine weitere Ganzzahl x ($a2)) und liefert 2 Werte zurück.

```
1: func:   addi  $s1,   $a1,    -1
2:         sll   $s1,   $s1,    2
3:         add   $s1,   $a0,    $s1
4:         add   $s0,   $a0,    $zero
5:         add   $v0,   $zero,  $zero
6:         add   $v1,   $zero,  $zero
7: jump1:  beq   $s0,   $s1,    jump4
8:         lw    $t0,   0($s0)
9:         lw    $t1,   0($s1)
10:        add   $t1,   $t0,    $t1
11:        beq   $t1,   $a2,    jump3
12:        slt   $t0,   $t1,    $a2
13:        beq   $t0,   $zero,  jump2
14:        addi  $s0,   $s0,    4
15:        j     jump1
16:jump2:  addi  $s1, $s1, -4
17:        j     jump1
18:
19:jump3:  sub   $v0, $s0, $a0
20:        sub   $v1, $s1, $a0
21:        srl   $v0, $v0, 2
22:        srl   $v1, $v1, 2
23:jump4:  jr    $ra
```

a) Was muss an der Funktion geändert werden, damit sie problemlos von anderen Programmierern verwendet werden kann?

(1 Punkt)

b) Wie lautet die Rückgabe der Funktion `funct`, wenn Speicher und Register zum Zeitpunkt des Aufrufs mit den Werten aus Abbildungen 2 und 3 belegt sind?

(12 Punkte)

0x10000140	10
0x1000013C	43
0x10000138	37
0x10000134	23
0x10000130	51
0x1000012C	43
0x10000128	41
0x10000124	37
0x10000120	35
0x1000011C	31
0x10000118	28
0x10000114	25
0x10000110	19
0x1000010C	9
0x10000108	5
0x10000104	3
0x10000100	2

Abbildung 2: Speicherabbild

$zero	0x00000000
$at	0x00120000
$v0	0x0011FFFF
$v1	0x0011FFFE
$v1	0x0011FFFE
$a0	0x10000104
$a1	0x0000000B
$a2	0x00000025
$a3	0x10000100
$t0	0x00110000
$t1	0x00110001
$t2	0x00110002
$t3	0x00110003
$t4	0x00110004
$t5	0x0011FFFE
$t6	0x0011FFFC
⋮	⋮

Abbildung 3: Registerabbild

c) Beschreiben Sie in 1-2 Sätzen was die Funktion `funct` berechnet?

(2 Punkte)

d) Die unten aufgeführten Instruktionen sind Pseudobefehle 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 $at. Des Weiteren dürfen nur folgende Befehle verwendet werden:

(6 Punkte)

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

Ersatzfelder sind auf der folgenden Seite.

not $t1 (invertiere $t1 bitweise):

subi $t1, $t2, 7:

andi $t1, 0xACDC ($t1=$t1 & 0xACDC):

ori $t1, $t2, 0xACDCACDC (32 Bit):

lw $t1, 100 ($t1 = MEM[100]):

ble $t1, $t2, label: (branch if lower or equal)

Ersatz, **ungültige Lösung streichen!**:

not $t1 (invertiere $t1 bitweise):

subi $t1, $t2, 7:

andi $t1, 0xACDC ($t1=$t1 & 0xACDC):

ori $t1, $t2, 0xACDCACDC (32 Bit):

lw $t1, 100 ($t1 = MEM[100]):

ble $t1, $t2, label: (branch if lower or equal)

Aufgabe 4: (Datenpfad) 16 Punkte

a) Geben Sie für die 3 MIPS Befehlstypen die jeweiligen Bit Grenzen an. Benennen Sie dazu in Abbildung 4 die Befehlstypen und geben Sie die noch fehlenden Grenzen (gestrichelt Kästchen) an.

(3 Punkte)

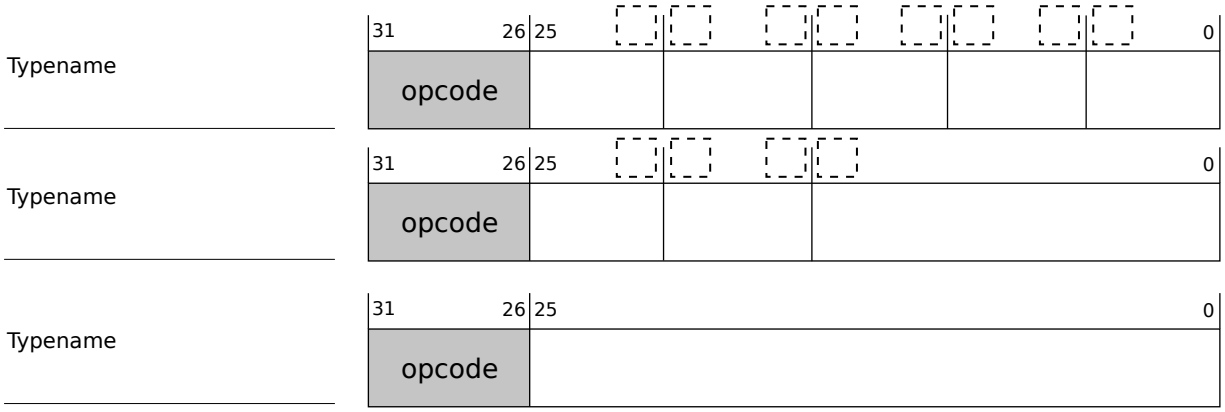


Abbildung 4: Bitgrenzen der Befehlstypen

b) Bei der Fertigung von Chips kann es passieren, dass Leitungen permanent mit '0' verbunden werden (logische *Null*). Beschreiben Sie kurz die Auswirkung dieses Defekts für die Steuerleitungen *RegWrite*, *RegDest*, *MemRead* und *MemToReg* des Datenpfads in Abbildung 6).

Beispiel: *Branch*: Es können keine Branch-Befehle ausgeführt werden.

(2 Punkte)

RegWrite: _____

_____

RegDest: _____

_____

MemRead: _____

_____

MemToReg: _____

_____

c) Markieren sie die Steuersignal-Instruktions-Kombinationen in der Tabelle, bei der die Instruktionen mit dem oben beschriebenen Defekt **nicht** mehr korrekt funktionieren. (4 Punkte)

	add	addi	sll	beq	bne	j	lw	nop
RegWrite=0								
RegDest=0								
MemRead=0								
MemToReg=0								

Tabelle 1: Steuersignal-Instruktions-Kombinationen

	add	addi	sll	beq	bne	j	lw	nop
RegWrite=0								
RegDest=0								
MemRead=0								
MemToReg=0								

Tabelle 2: Ersatz Steuersignal-Instruktions-Kombinationen, **ungültige Lösung streichen!**

d) Die Einzyklenimplementierung des MIPS Datenpfads (Abbildung 6) soll um die Instruktion `addui rt rs imm` (add upper immediate) ergänzt werden. Dieser Befehl besetzt die 16 höherwertigen Bits des Registers *rt* durch die Summe von der im Befehl angegebenen Konstanten *imm* und der 16 höherwertigen Bits von *rs*. Die niederwertigen Bits von *rt* werden mit den niederwertigen Bits von *rs* beschrieben. ($rt[31..16] = rs[31..16] + imm$, $rt[15..0] = rs[15..0]$)

Ein Beispiel zur Verdeutlichung: Der Befehl *addui $t1 $t0* 0x00ff besitzt folgende Bitinterpretation:

opcode	rs	rt	imm
001111	01010	01001	00000000 11111111 (0x00ff)

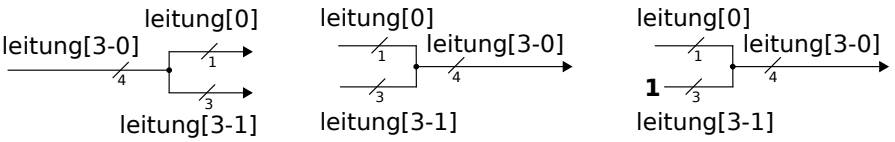
Inhalt von *$t0*:	00100010 00111101 10101101 11111111
Konstante 0x00ff:	00000000 11111111
	⇓
$t1 **nach** Ausführung des Befehls:	00100011 00111100 10101101 11111111

Modifizieren Sie die Einzyklenimplementierung in Abbildung 6 so, dass der Befehl *addui* ausgeführt werden kann.

Als zusätzliche Bauteile stehen Ihnen nur Multiplexer zur Verfügung. Benutzen Sie für das Auftrennen bzw. Zusammenführen von Leitungsbündeln (Bussen) die Notation aus Abbildung 5. Beim Zusammenführen können Leitungen auch auf logisch *Null* oder *Eins* festgelegt werden (Abbildung 5c)

Gehen Sie davon aus, dass die ALU die korrekte Operation ausführt.

(6 Punkte)



(a) Auftrennen (b) Zusammenführen (c) mit fester *Eins*

Abbildung 5: Auftrennen bzw. Zusammenführen von Leitungsbündeln (Bussen)

e) Muss auch der Steuerpfad bzw. die Steuerung verändert werden? Bergünden Sie Ihre Antwort!

(1 Punkt)

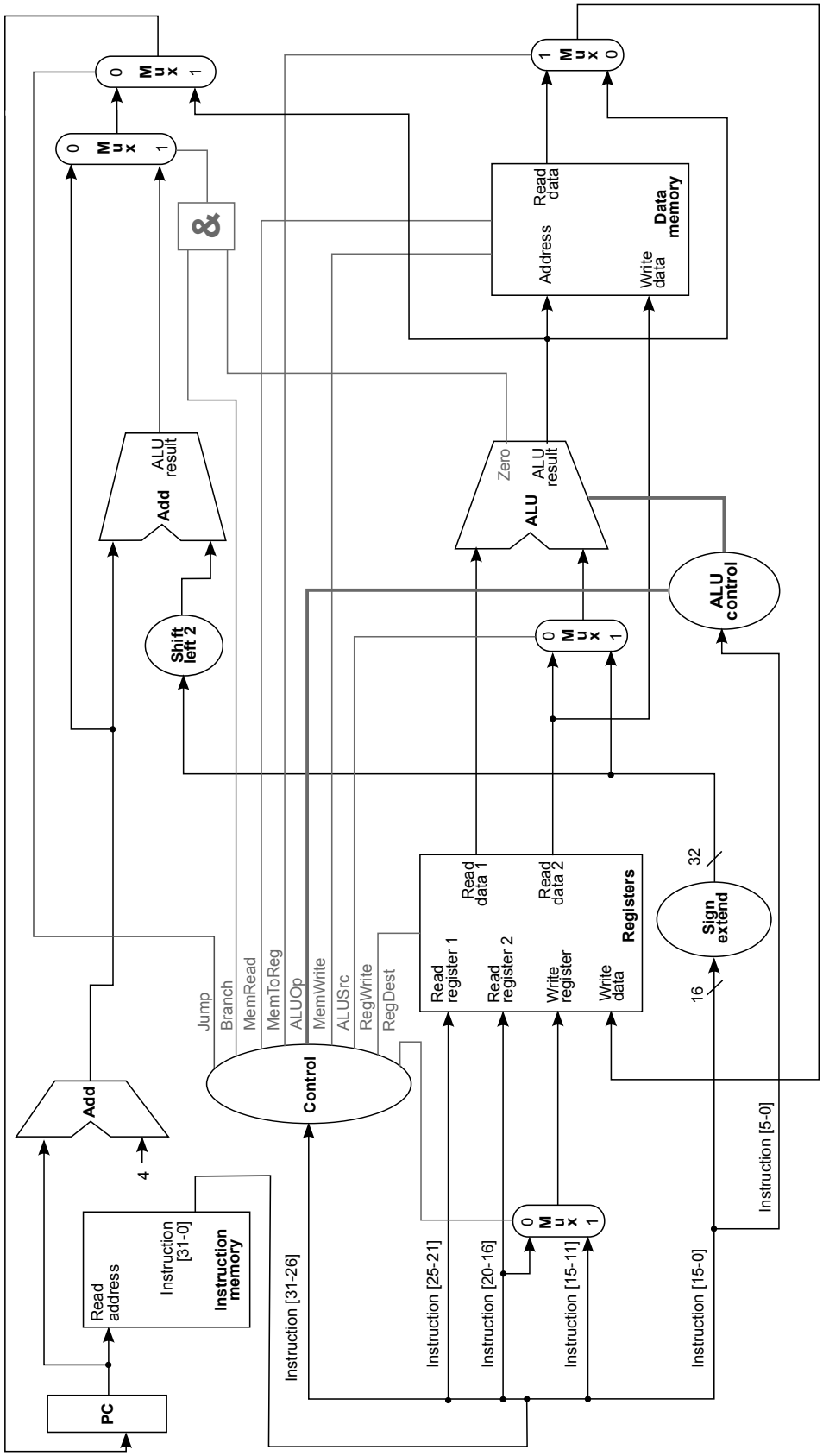


Abbildung 6: Einzyklenimplementierung des MIPS Datenpfads

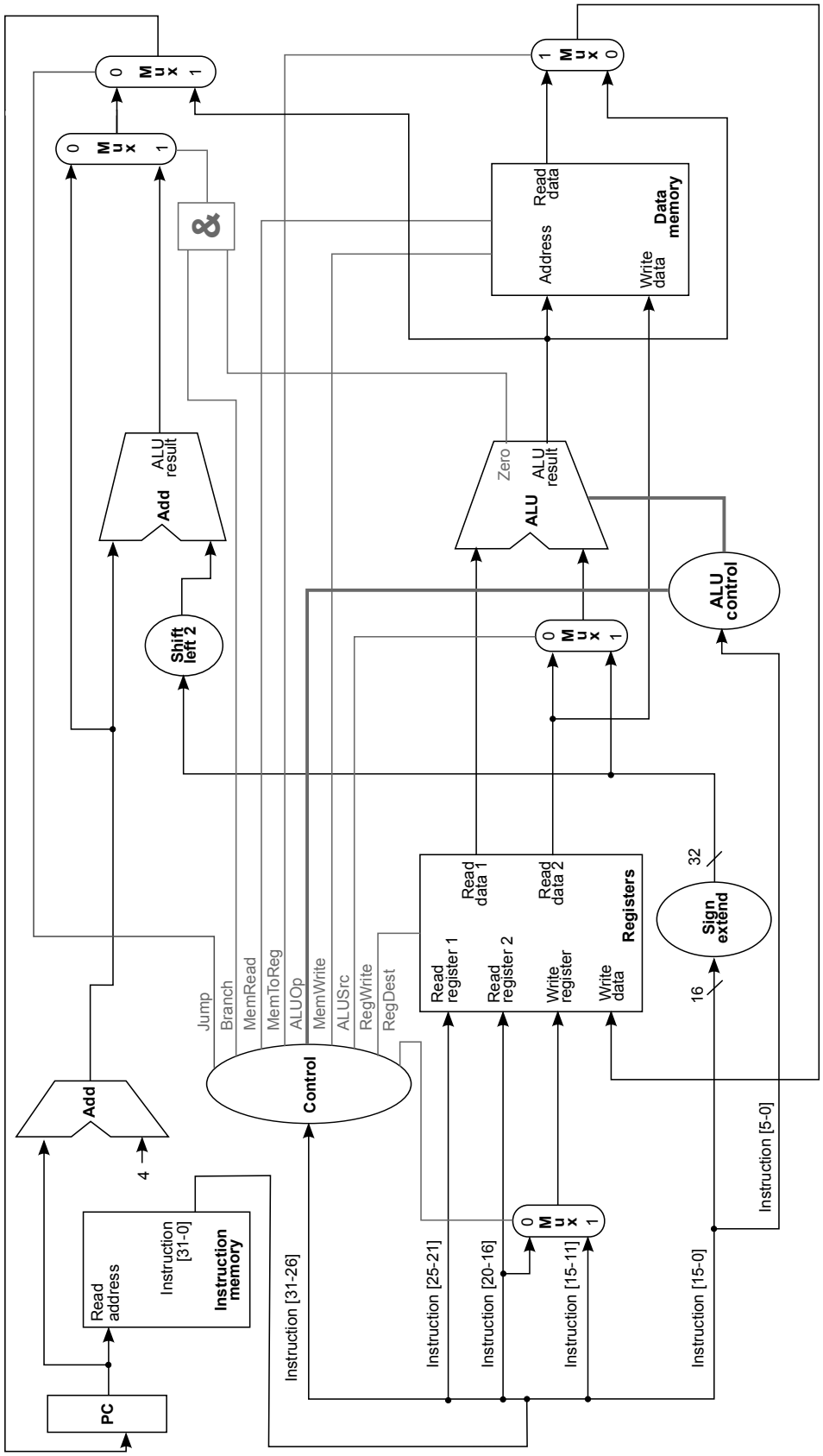


Abbildung 7: Ersatz Einzyklenimplementierung des MIPS Datenpfads. **ungültige Lösung streichen!**

Aufgabe 5: (Cache) 20 Punkte

Zur Beschleunigung der Speicherzugriffe auf den byteadressierten Hauptspeicher mit einer Kapazität von 4GB sei ein Computersystem mit einem Instuktions- und einem Datencache ausgestattet und über einen Bus mit dem Hauptspeicher verbunden, siehe Abbildung 8.

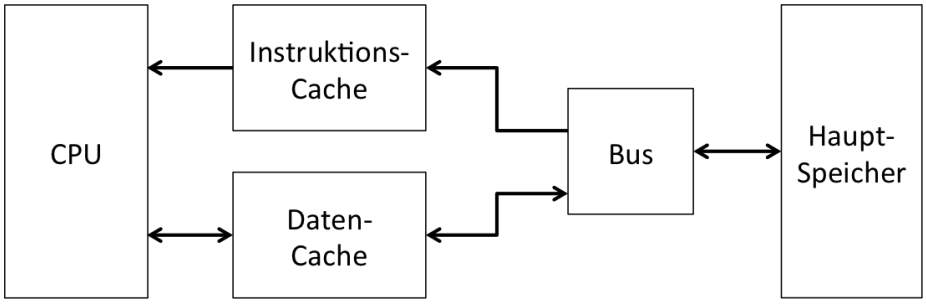


Abbildung 8: Speicherorganisation

Der Instuktionscache ist als direkt abgebildeter Cache (*direct mapping*) mit einer Kapazität von 2KB implementiert, wobei jeder Rahmen 32 Byte aufnehmen kann. Der Datencache hingegen ist 2-way-set-associative mit 4 Rahmen, wobei jeder Rahmen 2 Worte umfasst. Ein Wort ist jeweils 32 bit.

Der Prozessor ist mit 1GHz getaktet und die Kommunikation zwischen Prozessor und Caches benötigt keine zusätzliche Zeit, während der Bus eine Taktrate von 100MHz und eine Breite von 32 bit hat. Hauptspeicherzugriffe benötigen für das erste Wort 7 Buszyklen und für jedes weitere Wort 1 Buszyklus pro Blockübertragung. Es soll nun von der CPU ein Programm mit 256 Instruktionen à 32 Bit ausgeführt werden, welches sequentiell ab Adresse $0x...0000$ im Hauptspeicher angeordnet ist. Bei den Instruktionen handelt es sich nur um arithmetische und *load*- Befehle.

a) Geben Sie, ausgehend für eine CPU mit einem CPI von 2, die CPU-Zeit an, die zur Ausführung des Programms benötigt wird. Gehen Sie davon aus, dass der Befehlscache zu Beginn leer sei und dass sich alle erforderlichen Daten im Datencache befinden.

(6 Punkte)

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

(3 Punkte)

Adressbus: _____

Tag: _____

Index: _____

Wortoffset: _____

Byteoffset: _____

c) Gegeben sei folgender Programmbeginn, wobei das Register `$s0` mit der Adresse $0x...28$ vorgeladen sei :

```
0x...0000: lw $t0, 0($s0)
0x...0004: lb $t1, 7($s0)
0x...0008: lb $t2, -1($s0)
0x...000c: lh $t3, 8($s0)
0x...0010: lb $t4, 1($s0)
0x...0014: lw $t5, 16($s0)
0x...0018: lb $t6, 0($s0)
```

Als Ersetzungsstrategie wird *FIFO* (first-in first-out) eingesetzt. Gehen Sie im Folgenden davon aus, dass der Datencache zu Beginn leer sei.

Geben Sie für jede Adresse an, ob es sich im Datencache um einen *miss* oder einen *hit* handelt. Geben Sie bei einem hit weiterhin an, zu welchem Zeitpunkt t die Daten in den Cache geschrieben wurden. Füllen Sie dazu folgende Tabelle aus (Ersatz-Tabelle auf der nächsten Seite, **ungültige Lösung streichen**):

(8 Punkte)

	hit/miss?	Bei hit: t=?
t=1: 0x...0000: lw $t0, 0($s0)		
t=2: 0x...0004: lb $t1, 7($s0)		
t=3: 0x...0008: lb $t2, -1($s0)		
t=4: 0x...000c: lh $t3, 8($s0)		
t=5: 0x...0010: lb $t4, 1($s0)		
t=6: 0x...0014: lw $t5, 16($s0)		
t=6: 0x...0018: lb $t6, 0($s0)		

Ersatz- Tabelle, **ungültige Lösung streichen!**

	hit/miss?	Bei hit: t=?
`t=1: 0x...0000: lw $t0, 0($s0)`		
`t=2: 0x...0004: lb $t1, 7($s0)`		
`t=3: 0x...0008: lb $t2, -1($s0)`		
`t=4: 0x...000c: lh $t3, 8($s0)`		
`t=5: 0x...0010: lb $t4, 1($s0)`		
`t=6: 0x...0014: lw $t5, 16($s0)`		
`t=6: 0x...0018: lb $t6, 0($s0)`		

d) Der Datencache soll nun als 4-way-set-associative Cache implementiert werden. Ändert sich hierdurch etwas an der hit-Rate für das Programm aus Aufgabenteil **c)**? Begründen Sie Ihre Antwort!

(3 Punkte)