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

Klausur

Grundlagen der Rechnerarchitektur

13. September 2018

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

a) Auf einem Rechner mit einer Taktfrequenz von 3,2 GHz wird das Programm A ausgeführt, welches aus 3 verschiedenen Instruktionstypen besteht:

Typ 1: CPI=12, Häufigkeit=50%

Typ 2: CPI=8, Häufigkeit=25%

Typ 3: CPI=16, Häufigkeit=25%

Wie lange benötigt der Rechner zur Ausführung eines $2^6 \cdot 10^9$ Instruktionen umfassenden Programms?

(4 Punkte)

- ☐ 240 s
- ☐ 42 s
- ☐ 600 s
- ☐ Keine Lösung ist richtig.

b) Wie hoch ist die MIPS-Rate eines Prozessors mit einer Taktrate von 1500 MHz, der innerhalb von 10 Taktzyklen 4 Instruktionen verarbeitet?

(3 Punkte)

- ☐ 3750 MIPS
- ☐ 4000 MIPS
- ☐ 600 MIPS
- ☐ Keine Lösung ist richtig.

c) Gegeben seien folgende Durchlaufzeiten für die Stufen einer 5-stufigen Pipeline eines Prozessors:

Stufe 1:	$3 \cdot 10^{-10}$ s
Stufe 2:	$5 \cdot 10^{-10}$ s
Stufe 3:	$5 \cdot 10^{-10}$ s
Stufe 4:	$4 \cdot 10^{-10}$ s
Stufe 5:	$10 \cdot 10^{-10}$ s

Berechnen Sie den Speedup, den der Prozessor mit Pipeline gegenüber einer äquivalenten Einzyklenimplementierung für ein Programm mit 10^9 Instruktionen erreichen kann.

(3 Punkte)

- ☐ ≈ 10
- ☐ ≈ 100
- ☐ ≈ 1000
- ☐ Keine Lösung ist richtig.

Aufgabe 2: (Assembler) 20 Punkte

a) Die Funktion $fun(n)$ sei wie folgt rekursiv definiert:

$$fun(0) = 5$$

$$fun(n) = fun(n-1) + 4n + 3$$

Der Wert der Variablen n steht zu Beginn des Programms an der ersten Adresse des globalen Datensegments im Speicher. Diese Adresse ist per Konvention durch den *global Pointer* (Register *gp*) gespeichert. Nach Abarbeitung des Programms soll der Wert $fun(n)$ unmittelbar auf n folgend im Speicher abgelegt werden.

Ergänzen Sie das nachstehende Assemblerprogramm zur Berechnung von $fun(n)$ entsprechend (auch die teilweise fehlenden Kommentare). Halten Sie sich an die in der Vorlesung vorgestellten Konventionen bzgl. des MIPS-Prozessors.

(15 Punkte)

```
# Hier beginnt das Codefragement
      lw  ..................  # Lade n in ein temporäres Register
      add ..................  # Kopiere n für Parameterübergabe
      jal fun                 # ..............................
      sw  ..................  # Schreibe Ergebnis in den Speicher
      ......................  # Springe zu end
  fun:
      subu .................  # Schaffe Platz für ..... Elemente
                              # Wo? ............................
      sw ...................  # Sichere entsprechende Register
      ......................
      ......................
      ......................
      bnez $a0, label1        # Falls n > 0: gehe zu label1
      addi .................  # Sonst: return 5
      ......................  # Stack um 2 Elemente erhöhen
      ......................  # Verlasse fun()
  label1:
      addi .................  # n := n - 1
      jal fun                 # Rufe 'fun' mit 'n-1' auf
      lw ...................  # Hole Argument n wieder
      ......................  # Hole Rücksprungadresse
      ......................  # Stack um 2 Elemente erhöhen
      ......................  # Mult. mit 4 durch logisches Schieben
      addi ............, 3    # Berechnung von (4n + 3)
      add ..................  # Berechnung von fun(........) + 4n + 3
      ......................  # Rücksprung
  end:
# Ende des Codefragments
```

Codefragment

```
# Hier beginnt das Codefragement
      lw  ..................  # Lade n in ein temporäres Register
      add ..................  # Kopiere n für Parameterübergabe
      jal fun                 # ..............................
      sw  ..................  # Schreibe Ergebnis in den Speicher
      ......................  # Springe zu end
  fun:
      subu .................  # Schaffe Platz für ..... Elemente
                              # Wo? ............................
      sw ...................  # Sichere entsprechende Register
      ......................
      ......................
      ......................
      bnez $a0, label1        # Falls n > 0: gehe zu label1
      addi .................  # Sonst: return 5
      ......................  # Stack um 2 Elemente erhöhen
      ......................  # Verlasse fun()
  label1:
      addi .................  # n := n - 1
      jal fun                 # Rufe 'fun' mit 'n-1' auf
      lw ...................  # Hole Argument n wieder
      ......................  # Hole Rücksprungadresse
      ......................  # Stack um 2 Elemente erhöhen
      ......................  # Mult. mit 4 durch logisches Schieben
      addi ............, 3    # Berechnung von (4n + 3)
      add ..................  # Berechnung von fun(........) + 4n + 3
      ......................  # Rücksprung
  end:
# Ende des Codefragments
```

Ersatz Codefragment. **Ungültige Lösung streichen!**

b) Der in Aufgabe **a)** verwendete Befehl `bnez $a0, label1` ist ein sogenannter Pseudobefehl. Für den Vergleich bzw. für die Ausführung von bedingten Sprüngen sind im MIPS-Befehlssatz die Befehle `bne, beq, slt, slti, sltu, sltiu` implementiert.

Geben Sie MIPS-Assemblercode für folgende bedingte Sprünge an. Verwenden Sie nur die gegebenen Befehle des MIPS-Befehlssatz (**keine Pseudobefehle**). x sei in Register $\$t1$ und y in Register $\$t2$ gespeichert. Als Hilfsregister steht das Register $\$t3$ zur Verfügung.

(5 Punkte)

```
if (x = y) goto label2
```

```
if (x > y) goto label2
```

```
if (x >= y) goto label2
```

Aufgabe 3: (Datenpfad) 20 Punkte

a) Definieren Sie die drei MIPS Befehlstypen. Benennen Sie dazu in Abbildung 1 die Befehlstypen und Feldnamen und geben Sie die noch fehlenden Bitgrenzen (gestrichelte Kästchen) an.

(3 Punkte)

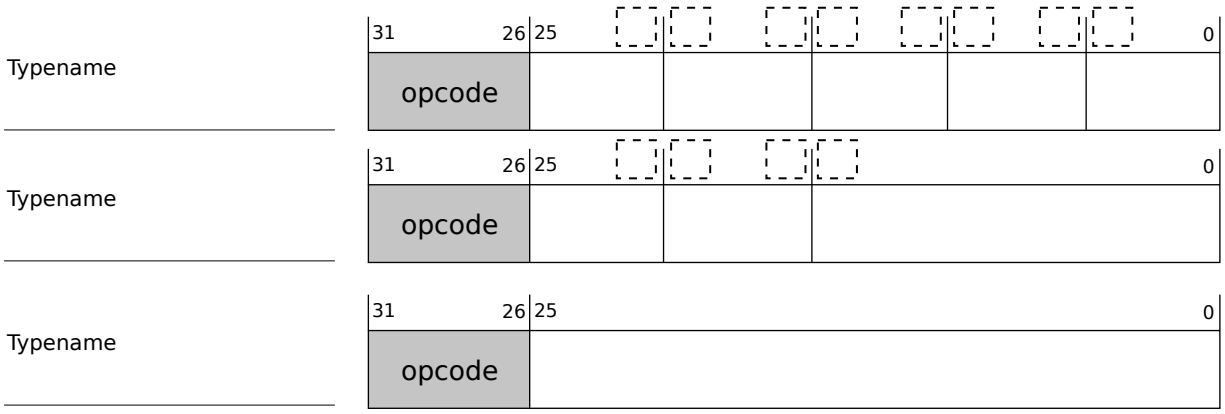


Abbildung 1: Bitgrenzen der Befehlstypen

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 2).

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

(4 Punkte)

RegWrite: _____

_____

RegDest: _____

_____

MemRead: _____

_____

MemToReg: _____

_____

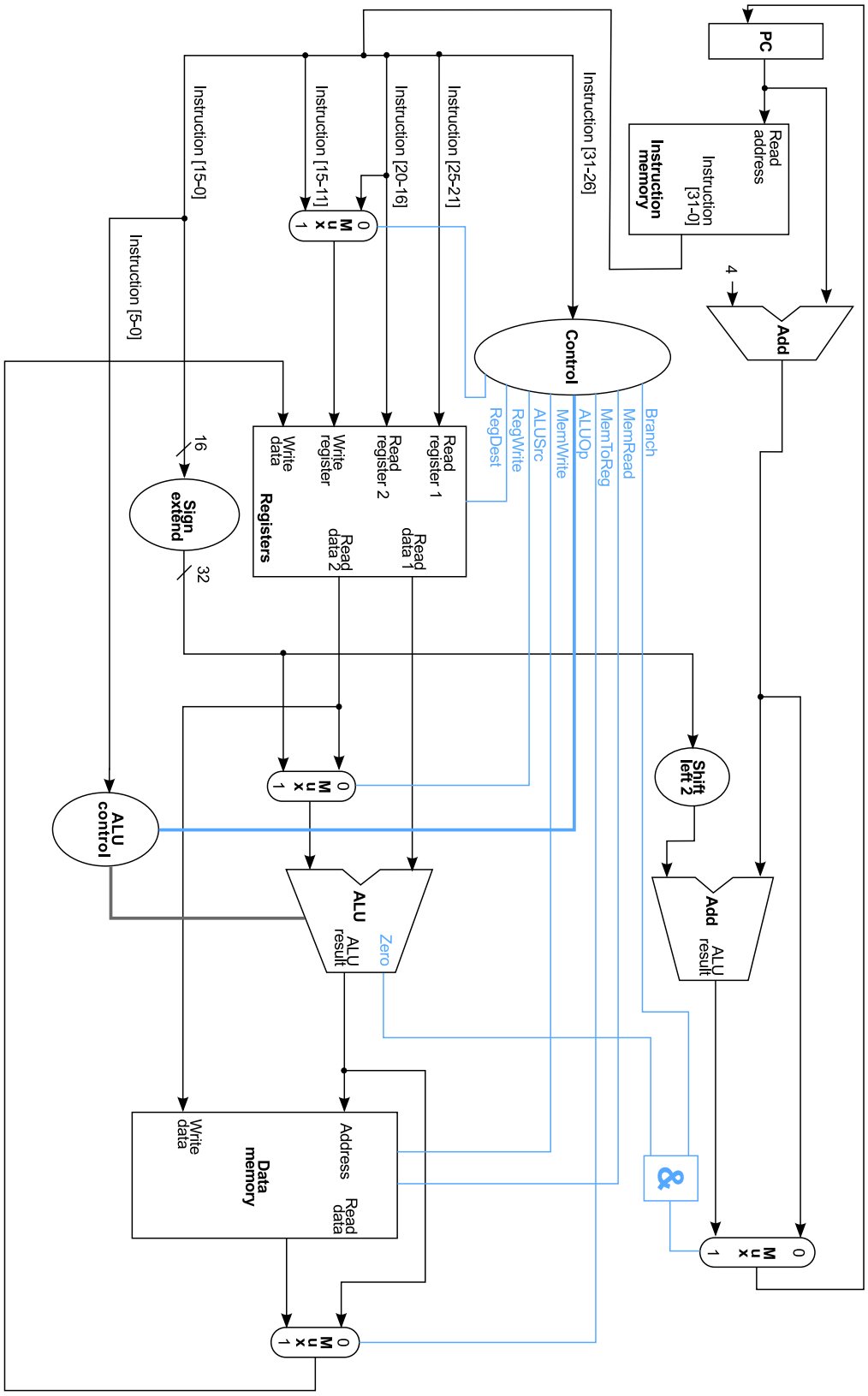


Abbildung 2: Einzyklenimplementierung des MIPS Datenpfads

c) Markieren Sie die Steuersignal-Instruktions-Kombinationen in der Tabelle 1, 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!

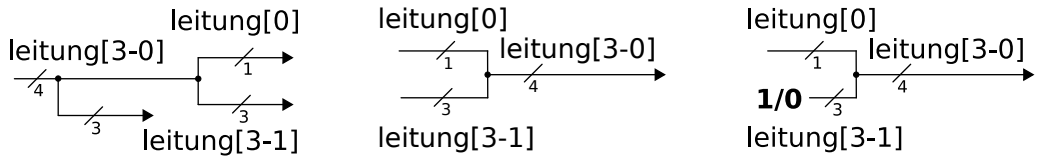
d) Die Einzyklenimplementierung des MIPS Datenpfads (Abbildung 4) soll um die Instruktion `addui rt rs imm` (add upper immediate) ergänzt werden. Dieser Befehl belegt die 16 höherwertigen Bits des Registers rt mit der Summe aus der im Befehl angegebenen Konstanten imm und der 16 höherwertigen Bits vom Register rs. Die niederwertigen Bits von Register rt werden nicht verändert.

$$rt[31..16] = rs[31..16] + imm, \quad rt[15..0] = rt[15..0]$$

Modifizieren Sie die Einzyklenimplementierung in Abbildung 4 so, dass der Befehl *addui* ausgeführt werden kann.

Als zusätzliche Bauteile stehen Ihnen nur Multiplexer zur Verfügung. Benutzen Sie für das Auftrennen/Abzweigen (Abb 3a) bzw. Zusammenführen (Abb 3b) die Notation aus Abbildung 3. Beim Zusammenführen können Leitungen auch auf logisch *Null* oder *Eins* festgelegt werden (Abbildung 3c).

Gehen Sie davon aus, dass die ALU die korrekte Operation ausführt. (8 Punkte)



(a) (b) (c)

Abbildung 3: Abzweigen/Auftrennen (a), Zusammenführen von Leitungsbündeln (Bussen)(b), Einzelne Leitungen auf festen Wert (1/0) legen (c)

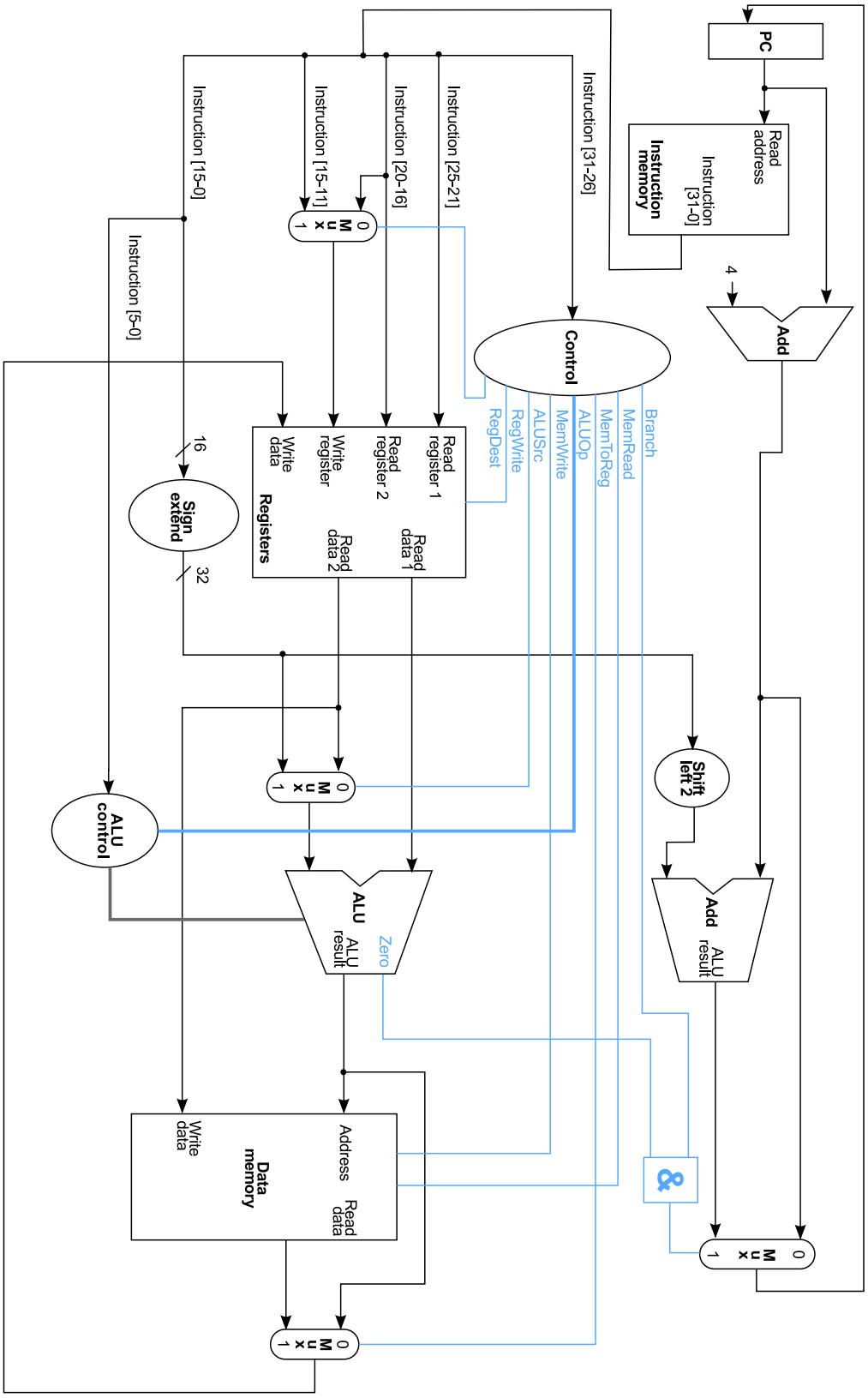


Abbildung 4: Ersatz Einzyklenimplementierung des MIPS Datenpfads
Ungültige Lösung streichen!

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

(1 Punkt)

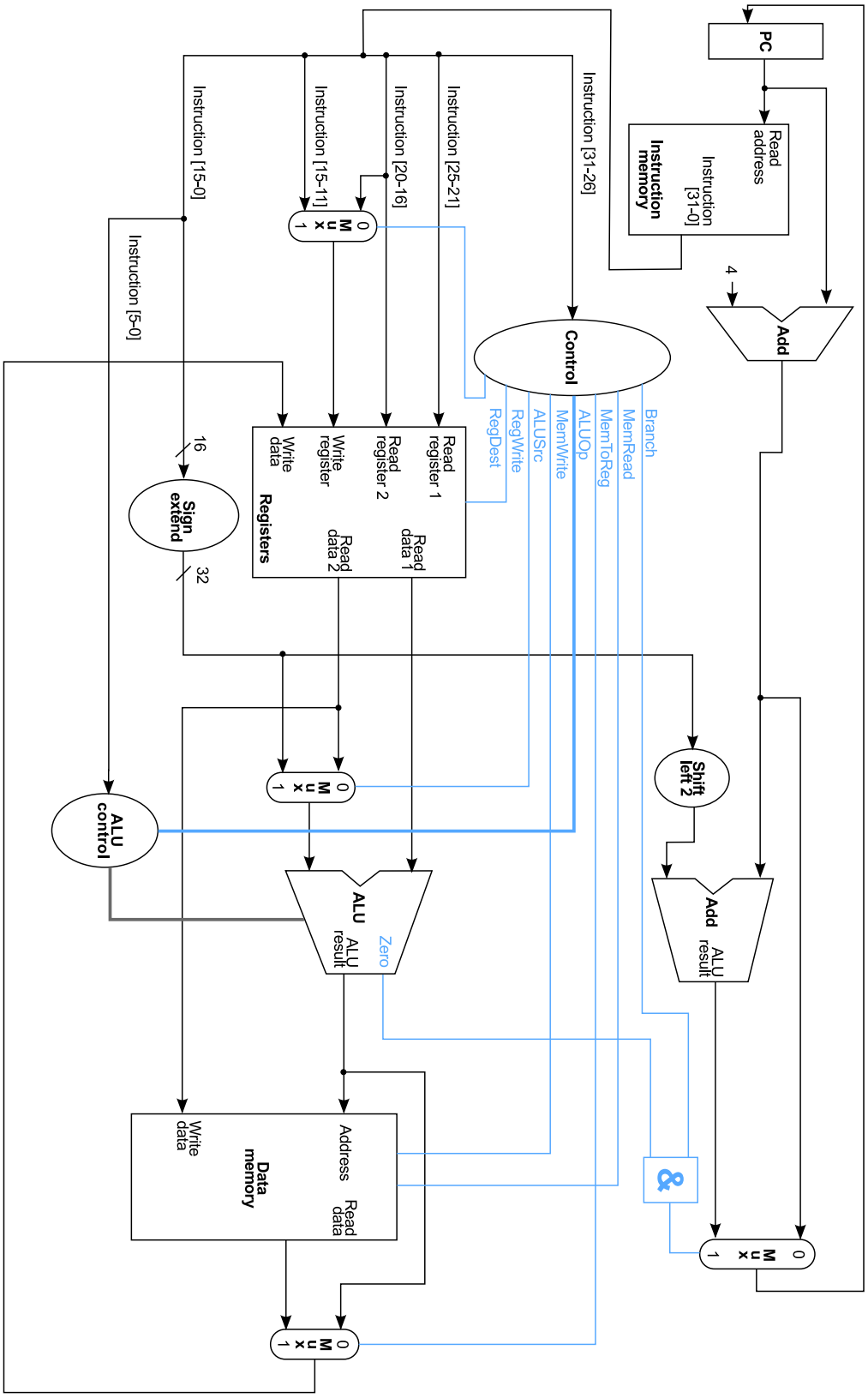


Abbildung 5: Ersatz Einzyklenimplementierung des MIPS Datenpfads
Ungültige Lösung streichen!

Aufgabe 4: (Cache) 20 Punkte

Gegeben sei ein Computer, der zur Beschleunigung der Speicherzugriffe auf den byte-adressierten Arbeitsspeicher mit einem Adressraum von 2 GB einen Cache mit 8 Rahmen vorsieht, wobei jeder Rahmen 4 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 Adressbusses sowie die Anzahl an Bits für Tag, Index, Wortoffset und Byteoffset.

(3 Punkte)

Adressbus: _____

Tag: _____

Index: _____

Wortoffset: _____

Byteoffset: _____

b) In der Vorlesung haben Sie verschiedene Cache-Strukturen kennengelernt. Nennen Sie einen Vor- und einen Nachteil, den (mehrfach) assoziative Caches gegenüber Caches mit direkter Abbildung haben.

(2 Punkte)

Vorteil: _____

Nachteil: _____

c) Nehmen Sie an, dass die CPU eine Taktrate von 2 GHz (0,5 ns pro Takt) mit einem idealen CPI-Wert von 2 hat. Nehmen Sie weiter an, dass die Fehlgriffsrate (miss-rate) bei Befehlen 2% und die Fehlgriffsrate bei Daten 4% beträgt, wobei eine Fehlgriffszeit (miss-penalty) von 100 ns entsteht. Die Häufigkeit von Datenzugriffen beträgt 25%. Bestimmen Sie die mittlere CPU-Zeit (in ns) für einen Befehl. Geben Sie auch den Rechenweg an.

(4 Punkte)

CPU-Zeit: _____

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

```
t=10:   0x49d  = 010010 01 1101
t=11:   0x696  = 011010 01 0110
t=12:   0x3a9  = 001110 10 1001
t=13:   0x245  = 001001 00 0101
t=14:   0x3a2  = 001110 10 0010
```

Gehen Sie von folgendem Zustand des Cache zum Zeitpunkt $t = 9$ aus (Spalte **t** gibt die letzte Zugriffszeit an):

Index	**Valid**	**t**	**Tag**	**Datenbereich**
0	0	0	000000	0x000-0x00f
	0	1	001001	0x240-0x24f
1	1	8	011011	0x6d0-0x6df
	1	9	011010	0x690-0x69f
2	1	6	000010	0x0a0-0x0af
	1	3	011101	0x760-0x76f
3	1	7	101011	0xaf0-0xaff
	0	2	000000	0xab0-0xabf

Geben Sie nach jedem Speicherzugriff den Zustand des Caches an. Füllen Sie hierzu die nachfolgenden Tabellen aus. Als Ersetzungsstrategie nutzen Sie LRU (least-recently-used). Geben Sie zum Schluss die Hitrate an.

Hinweis: Sie dürfen unveränderte Einträge mit „unv.“ markieren statt sie erneut auszufüllen. Das Valid-Bit muss nicht eingetragen werden.

(11 Punkte)

t=10:

Index	**t**	**Tag**	**Datenbereich**
0x49d			

t=11:

Index	**t**	**Tag**	**Datenbereich**
0x696			

t=12: 0x3a9

Index	**t**	**Tag**	**Datenbereich**

t=13: 0x245

Index	**t**	**Tag**	**Datenbereich**

t=14: 0x3a2

Index	**t**	**Tag**	**Datenbereich**

Hitrate: _____

Ersatztabellen. **Ungültige Tabellen streichen!**:

t=____:

Index	**t**	**Tag**	**Datenbereich**

t=____:

Index	**t**	**Tag**	**Datenbereich**

t=____:

Index	**t**	**Tag**	**Datenbereich**

Aufgabe 5: (Pipelining) 20 Punkte

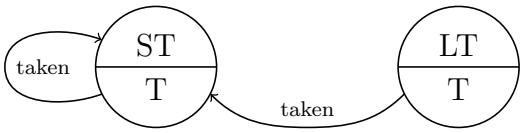
Um das Anhalten der Prozessorpipeline (Pipeline stall) im Falle eines bedingten Sprungs (branch) zu verhindern, kommt oft eine dynamische Sprungvorhersage zum Einsatz, die versucht vorherzusagen, ob die Verzweigung eintritt (taken/T), oder nicht (not taken/N).

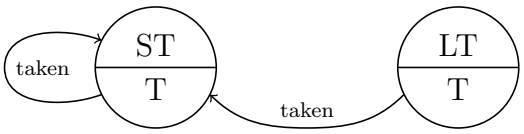
a) In der nachfolgenden Tabelle finden Sie die Aufzeichnung der Sprungvorhersagen eines 2-bit Prädiktors für eine ausgewählte Branch-Anweisung innerhalb eines Programmablaufs. Wie muss der Prädiktorautomat aussehen, um die Vorhersagen wie angegeben zu treffen? Ergänzen Sie in der Tabelle die Zustände des Prädiktors zum Zeitpunkt des Aufrufs, und vervollständigen Sie das untenstehende Automatendiagramm! (10 Punkte)

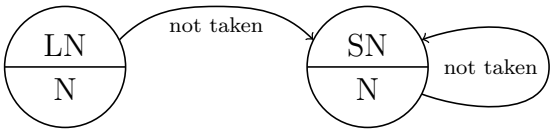
Aufruf	1	2	3	4	5	6	7	8	9	10	11	12	13	14
Zustand	LT													
Vorhersage	T	T	T	T	T	N	T	N	N	N	N	N	N	N
Tatsächlich	T	N	T	N	N	T	N	N	N	T	N	N	N	N

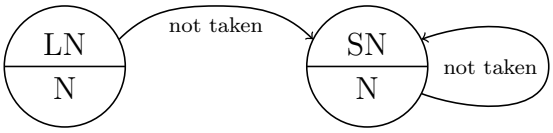
Aufruf	1	2	3	4	5	6	7	8	9	10	11	12	13	14
Zustand	LT													
Vorhersage	T	T	T	T	T	N	T	N	N	N	N	N	N	N
Tatsächlich	T	N	T	N	N	T	N	N	N	T	N	N	N	N

Ersatztabelle. **Ungültige Lösung streichen!**









Prädiktorautomat

Ersatzautomat
Ungültige Lösung streichen!

Hierbei soll bedeuten:
ST $\widehat{=}$ „strongly“ taken, **LT** $\widehat{=}$ „likely“ taken,
LN $\widehat{=}$ „likely“ not taken, **SN** $\widehat{=}$ „strongly“ not taken.

b) Der 2-bit Prädiktor aus Aufgabe **a)** hat bei der angegebenen Sequenz 8 Treffer in der Vorhersage erzielt. Wie gut hätte ein 1-bit Prädiktor die Sprungentscheidungen vorhergesagt? Füllen Sie nachfolgende Tabelle aus und ermitteln Sie die Anzahl der Treffer. (4 Punkte)

Aufruf	1	2	3	4	5	6	7	8	9	10	11	12	13	14
Vorhersage	T													
Tatsächlich	T	N	T	N	N	T	N	N	N	T	N	N	N	N

Aufruf	1	2	3	4	5	6	7	8	9	10	11	12	13	14
Vorhersage	T													
Tatsächlich	T	N	T	N	N	T	N	N	N	T	N	N	N	N

Ersatztabelle. **Ungültige Lösung streichen!**

Anzahl Treffer: _____

c) Neben 2-bit Prädiktoren kommt häufig noch ein Branch Target Buffer (BTB) zum Einsatz, welcher als eine Art Cache für Zieladressen von Verzweigungen verwendet wird. Welche Daten werden in so einem Buffer gespeichert? In einem System mit 32-bit Adressen und 2-bit Prädiktoren, wie viele Einträge könnte ein BTB halten, der 264 Bytes Speicherkapazität hat? (6 Punkte)

In einem Branch Target Buffer werden gespeichert:

Mit 264 Bytes Speicherkapazität, könnte ein BTB _____ Sprungvorhersagen für das beschriebene System vorhalten.