Universität Paderborn

Institut *Elektrotechnik und Informationstechnik*

Fachgebiet *Datentechnik*

Prof. Sybille Hellebrand

Klausur

Technische Informatik / Grundlagen der Rechnerarchitektur

14. September 2012

Punkteverteilung						
Aufgabe	1	2	3	4	5	Σ
maximale Punkte	20	20	16	17	17	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: (Cache) 20 Punkte

a) Der Speicher eines Computers wird oft hierarchisch aufgebaut, um die Speicherzugriffe der Programme zu beschleunigen. Auf Grund welcher beiden Prinzipien funktioniert dieser Aufbau?

(2 Punkte)

Gehen Sie im Folgenden von einem Computer aus, der zur Beschleunigung der Speicherzugriffe auf den byteadressierten Arbeitsspeicher mit einer Kapazität von 2GB einen Cache mit 8 Rahmen vorsieht, wobei jeder Rahmen 2 Worte zu je 32 Bit umfasst. Der Cache sei als 4-way-set-associative Cache organisiert und als Ersetzungsstrategie wird **LRU** (**L**east **R**ecently **U**sed) eingesetzt.

b) Geben Sie die Breite von Daten- und Adressbus sowie die Anzahl an Bits für Tag, Index und Offsets an.

(3 Punkte)

Datenbus: _____

Adressbus: _____

Tag: _____

Index: _____

Wortoffset: _____

Byteoffset: _____

Während einer Programmbearbeitung lädt der Prozessor sequentiell die Daten der folgenden Byteadressen (führende Nullen werden nicht mit angegeben):

```
t=1:  ...01000010
t=2:  ...01000111
t=3:  ...00111011
t=4:  ...01101000
t=5:  ...01011101
t=6:  ...00111100
t=7:  ...01111000
t=8:  ...00011010
t=9:  ...01101001
t=10: ...00111011
```

Gehen Sie davon aus, dass der Cache zu Beginn leer ist.

c) Bestimmen Sie zunächst für die Befehle von *t=1* bis *t=3* die Adressen der zu ladenden Bytes.

(3 Punkte)

`t=1: ...01000010` _____

`t=2: ...01000111` _____

`t=3: ...00111011` _____

d) Bestimmen Sie nun die Hitrate bei der Ausführung des Programms. Geben Sie hierzu die Tags des Caches zum Zeitpunkt t mit Hilfe der folgenden Tabellen an. (Ersatz-Tabellen auf Seiten 7-8, **ungültige Lösung streichen!**).

(10 Punkte)

Adresse=...01000010

t=1	Tag
0	
1	
2	
3	
4	
5	
6	
7	

Adresse=...01000111

t=2	Tag
0	
1	
2	
3	
4	
5	
6	
7	

Adresse=...00111011

t=3	Tag
0	
1	
2	
3	
4	
5	
6	
7	

Adresse=...01101000

t=4	Tag
0	
1	
2	
3	
4	
5	
6	
7	

Adresse=...01011101

t=5	Tag
0	
1	
2	
3	
4	
5	
6	
7	

Adresse=...00111100

t=6	Tag
0	
1	
2	
3	
4	
5	
6	
7	

Adresse=...01111000

t=7	Tag
0	
1	
2	
3	
4	
5	
6	
7	

Adresse=...00011010

t=8	Tag
0	
1	
2	
3	
4	
5	
6	
7	

Adresse=...01101001

t=9	Tag
0	
1	
2	
3	
4	
5	
6	
7	

Adresse=...00111011

t=10	Tag
0	
1	
2	
3	
4	
5	
6	
7	

Hitrate: _____

Ersatz-Tabellen, ungültige Lösung streichen

t=1	Adresse=...01000010
0	
1	
2	
3	
4	
5	
6	
7	

t=2	Adresse=...01000111
0	
1	
2	
3	
4	
5	
6	
7	

t=3	Adresse=...00111011
0	
1	
2	
3	
4	
5	
6	
7	

t=4	Adresse=...01101000
0	
1	
2	
3	
4	
5	
6	
7	

t=5	Adresse=...01011101
0	
1	
2	
3	
4	
5	
6	
7	

t=6	Adresse=...00111100
0	
1	
2	
3	
4	
5	
6	
7	

t=7	Adresse=...01111000
0	
1	
2	
3	
4	
5	
6	
7	

t=8	Adresse=...00011010
0	
1	
2	
3	
4	
5	
6	
7	

t=9	Adresse=...01101001
0	
1	
2	
3	
4	
5	
6	
7	

t=10	Adresse=...00111011
0	
1	
2	
3	
4	
5	
6	
7	

Hitrate: _____

e) Das Programm soll nun zum Zeitpunkt t=4 einen Schreibbefehl ausführen (der Rest bleibt unverändert). Zu welchem Zeitpunkt t wird der Arbeitsspeicher aktualisiert, wenn das

(2 Punkte)

Write-back- Verfahren: _____

Write-through- Verfahren: _____

angewandt wird?

Aufgabe 2: (Assembler) 20 Punkte

a) Gegeben ist der Pseudocode für den Bubble-Sort Algorithmus. Übersetzen Sie die Funktion in MIPS-Assemblerbefehle. Benutzen Sie dabei keine Pseudobefehle und verwenden Sie das Programmgerüst auf der nächsten Seite (Ersatz auf Seite 12, **ungültige Lösung streichen!**).

(10 Punkte)

```
sort(Array A)
  for (n=A.size; n>1; n=n-1)
    for (i=0; i<n-1; i=i+1)
      if (A[i] > A[i+1])
        A.swap(i, i+1)
```

b) Welche Register werden für die Variablen **i** und **n** verwendet?

(1 Punkt)

c) Wie kann die Funktion durch Umsortieren um eine Programmzeile verkleinert werden, ohne die Funktion selbst zu verändern? Verschieben Sie im Programmcode von Aufgabe **a)** eine Zeile so, dass eine andere Zeile überflüssig wird. Welche Zeile muss durch die Verschiebung angepasst werden? Wie sieht die modifizierte Zeile aus?

(4 Punkte)

Assembler- Programmgerüst

```
1  .______ 0x10010000
2  size:   .word 5
3  data:   .word 8
4          .word 82
5          .word 43
6          .word 23
7          .word 74
8
9  .______
10 main:   ____   ____, ________       #Lade Funktionsparameter
11         _____  _____, _____, _____  #Feldposition und Feldgröße
12         ____   ____, ________       #in die Übergaberegister
13
14         ____    ____                #rufe sort auf
15         addi  $v0, $zero, 10        #Systemaufruf, um
16         syscall                     #_______________________
17
18 sort:   add    $s1, $zero, $a1
19 loop1:  _____  _____, _____, _____  #Führe äußere Schleife nur
20         _____  _____, _____, _____  #aus wenn n>1
21         addi   $s2, $zero, 0
22         addi   $t1, $s1, -1
23 loop2:  _____  _____, _____, _____  #wenn(i<n-1)
24         _____  _____, _____, _____
25         _____  _____, _____, _____  #Berechne Adresse A[i] und
26         _____  _____, _____, _____  #speichere sie in $t0
27         lw     $t1, ______          #Lade A[i]
28         lw     $t2, ______          #Lade A[i+1]
29         slt    $t3, $t2, $t1        #überspringe Vertauschen
30         bne    $t3, $zero, noswap   #wenn A[i]>=A[i+1]
31         ____   ____, ________       #tausche A[i] und A[i+1]
32         ____   ____, ________
33 noswap: addi   $s2, $s2, 1
34         j      loop2
35 l2_end: addi   $s1, $s1, -1
36         j      loop1
37 l1_end: _____  _____                 #Rücksprung ins Hauptprogramm
```

Ersatz Assembler- Programmgerüst, **ungültige Lösung streichen!**

```
1  ._____ 0x10010000
2  size:   .word 5
3  data:   .word 8
4          .word 82
5          .word 43
6          .word 23
7          .word 74
8
9  ._____
10 main:   ____   ____, ________       #Lade Funktionsparameter
11         _____  _____, _____, _____  #Feldposition und Feldgröße
12         ____   ____, ________       #in die Übergaberegister
13
14         ____    ____                #rufe sort auf
15         addi  $v0, $zero, 10        #Systemaufruf, um
16         syscall                     #_______________________
17
18 sort:   add    $s1, $zero, $a1
19 loop1:  _____  _____, _____, _____  #Führe äußere Schleife nur
20         _____  _____, _____, _____  #aus wenn n>1
21         addi   $s2, $zero, 0
22         addi   $t1, $s1, -1
23 loop2:  _____  _____, _____, _____  #wenn(i<n-1)
24         _____  _____, _____, _____
25         _____  _____, _____, _____  #Berechne Adresse A[i] und
26         _____  _____, _____, _____  #speichere sie in $t0
27         lw     $t1, ______          #Lade A[i]
28         lw     $t2, ______          #Lade A[i+1]
29         slt    $t3, $t2, $t1        #überspringe Vertauschen
30         bne    $t3, $zero, noswap   #wenn A[i]>=A[i+1]
31         ____   ____, ________       #tausche A[i] und A[i+1]
32         ____   ____, ________
33 noswap: addi   $s2, $s2, 1
34         j      loop2
35 l2_end: addi   $s1, $s1, -1
36         j      loop1
37 l1_end: _____  _____                 #Rücksprung ins Hauptprogramm
```

d) Übersetzten Sie folgende C-Schleife in MIPS Assembler. Verwenden Sie dabei keine Pseudobefehle. Nehmen Sie an, dass das Array **A** an Adresse 0x10004040 abgelegt ist.

(5 Punkte)

```
for(int i=0;i<4;i=i+1)
    A[i]=A[i]*4;
```

Aufgabe 3: (Pipeline) 16 Punkte

a) Nennen Sie die drei Arten von Konflikten, die beim Pipelining auftreten können. Welche Möglichkeiten gibt es, die jeweiligen Konflikte aufzulösen ohne Pipelineleerläufe?

(3 Punkte)

b) Zur Sprungvorhersage werden zwei dynamische Prädiktoren sowie die statische Vorhersage *always taken* eingesetzt. Die dynamischen Prädiktoren seien durch die Moore-Automaten in Abbildung 1 bzw. 2 gegeben. Die Eingaben der Prädiktoren mit dem Eingabealphabet $\{T,NT\}$ entsprechen dem Verhalten des Programms, d.h. ob ein Sprung durchgeführt wurde (T) oder nicht (NT). Die Ausgabe mit dem Ausgabealphabet $\{t,dnt\}$ gibt die Sprungvorhersage an (*take* oder *do not take*). Die Zustände des Automaten sind gegeben durch $\{ST,LT,LNT,SNT\}$ bzw.$\{ST,LT,SNT\}$ und dem Startzustand $\{ST\}$. Vergleichen Sie die drei möglichen Sprungvorhersagen für die Sprungfolge

$$T\ NT\ T\ T\ NT\ NT\ NT\ NT\ T\ NT\ T,$$

in dem Sie den Zustand des Automaten, die vorhergesagten Ausgaben und die Anzahl der richtig vorhergesagten Sprünge angeben.

(6 Punkte)

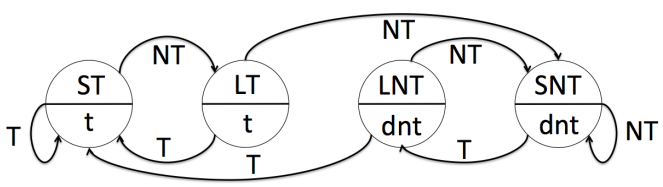

Abbildung 1: Prädiktor A

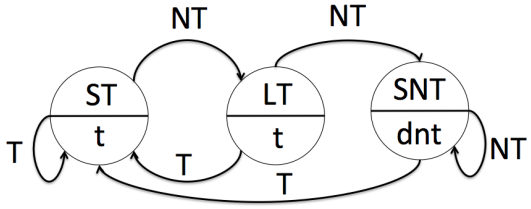

Abbildung 2: Prädiktor B

Sprungfolge	*T*	*NT*	*T*	*T*	*NT*	*NT*	*NT*	*NT*	*T*	*NT*	*T*
Vorhersage *always taken*:											
Zustand Prädiktor A:											
Vorhersage Prädiktor A:											
Zustand Prädiktor B:											
Vorhersage Prädiktor B:											

Anzahl der Treffer bei

always taken: _____

Prädiktor A: _____

Prädiktor B: _____

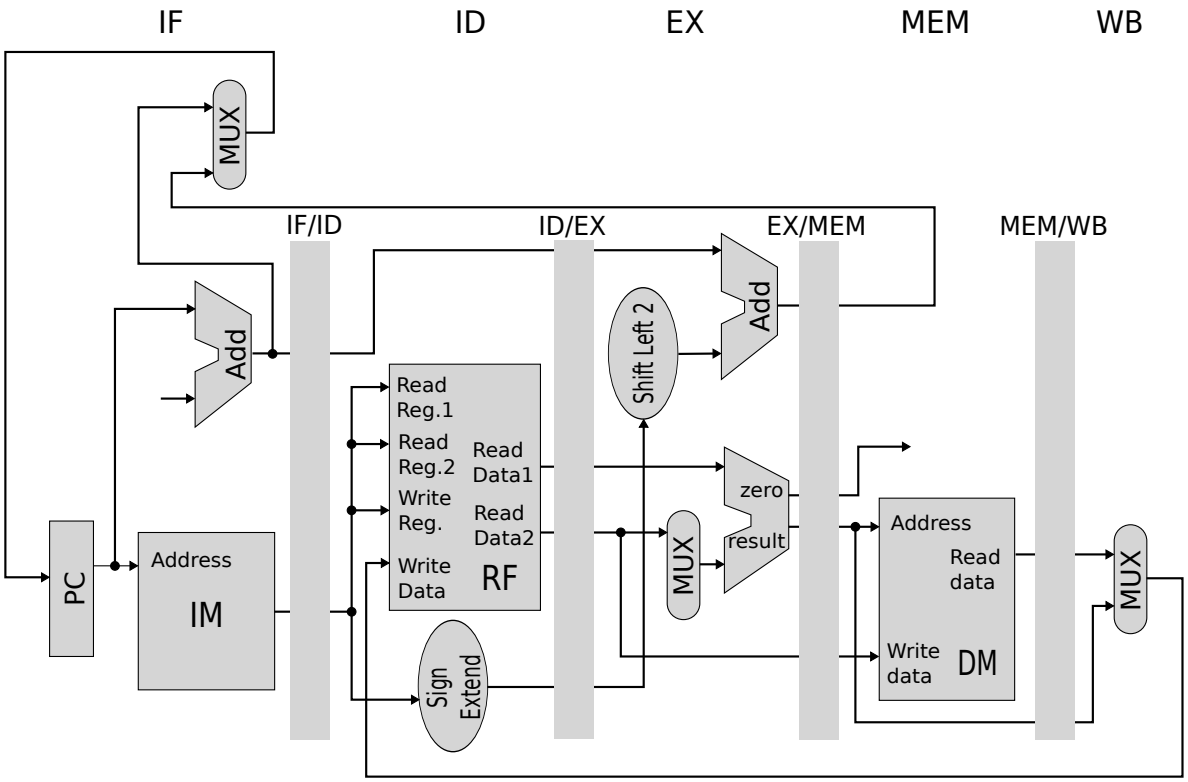

Abbildung 3: MIPS Datenpfad

c) Gegeben sei der MIPS Prozessor aus Abbildung 3 und folgende Sequenz eines Assemblerprogramms:

```
sub $s2, $t1, $t3
add $s4, $s2, $t5
addi $t5, $zero, 6
add $v0, $t5, $zero
sw $s0, 0($s2)
```

Nehmen Sie an, die Hardware sei ohne Forwarding-Unit implementiert und die Befehle vor der angegebenen Sequenz haben keine Stalls der Pipeline erzeugt.

Bestimmen Sie die Anzahl der zur Ausführung der Programmsequenz benötigten Takte. Vervollständigen Sie hierzu die folgende Pipeline-Tabelle:

(3 Punkte)

sub $s2, $t1, $t3	IF	ID	EX	M	WB									
add $s4, $s2, $t5														
addi $t5, $zero, 6														
add $v0, $t5, $zero														
sw $s0, 0($s2)														

Tabelle 1: Pipeline Tabelle

Anzahl der Takte: _____

sub $s2, $t1, $t3	IF	ID	EX	M	WB									
add $s4, $s2, $t5														
addi $t5, $zero, 6														
add $v0, $t5, $zero														
sw $s0, 0($s2)														

Tabelle 2: Ersatz Pipeline Tabelle, **ungültige Lösung streichen**

Anzahl der Takte: _____

d) Wie groß sind Speed-Up und Effizienz der Pipeline für das gegebene Assemblerprogramm gegenüber einer sequentiellen Abarbeitung der Befehle und unter der Annahme von gleichlanger Rechenzeit pro Stufe (Registeroperationen benötigen keine extra Rechenzeit)?

(2 Punkte)

Speed-Up: _____

Effizienz: _____

e) Geben Sie für das gegebene Assemblerprogramm Speed-Up und Effizienz gegenüber einer sequentiellen Abarbeitung der Befehle und unter der Annahme von gleichlanger Rechenzeit pro Stufe an, wenn die Pipeline mit einer Forwarding- Einheit ausgestattet ist (Registeroperationen benötigen keine extra Rechenzeit)!

(2 Punkte)

Speed-Up mit Forwarding: _____

Effizienz mit Forwarding: _____

Aufgabe 4: (Datenpfad) 17 Punkte

Abbildung 5 zeigt die aus der Vorlesung bekannte MIPS Einzyklenimplementierung.

a) Definieren Sie die Steuersignale des Kontrollers für eine Operation des R-Typs und tragen Sie diese in die nachstehende Tabelle ein. Benutzen Sie so viele "don't care"-Werte (X) wie möglich. (4 Punkte)

Instr.	RegDst	ALUSrc	MemToReg	RegWrite
R-Type				

Instr.	MemRead	MemWrite	Branch	ALUOp[1:0]
R-Type				$10\widehat{=}$Addition

b) Erweitern Sie den Datenpfad so, dass die neue Instruktion Jump-Register-Displacement `jrd d(Rs)` (PC=Rd+d_{16}) möglich ist. Der PC enthält nach Ausführung dieses Befehls den Wert des angegeben Registers addiert mit der vorzeichenbehafteten 16 Bit Konstanten d_{16}. Tragen Sie die notwendigen Änderungen in Abbildung 5 ein. (6 Punkte)

c) Geben Sie eine mögliche MIPS Instruktionskodierung für die Instruktion `jrd` in Abbildung 4 an. Begründen Sie Ihre Entscheidung knapp in maximal zwei Sätzen. Ohne Begründung gibt es keine Punkte! (3 Punkte)

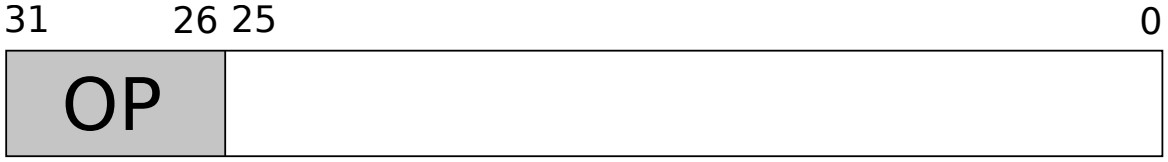

Abbildung 4: Instruktionsformat

d) Definieren Sie nun die Steuersignale des Kontrollers für eine `jrd`-Instruktion und tragen Sie die Belegung in die nachstehende Tabelle ein. Benutzen Sie so viele "don't care"- Werte (X) wie möglich. (4 Punkte)

Instr.	RegDst	ALUSrc	MemToReg	RegWrite	_____
`jrd`					

Instr.	MemRead	MemWrite	Branch	ALUOp[1:0]	_____
`jrd`					

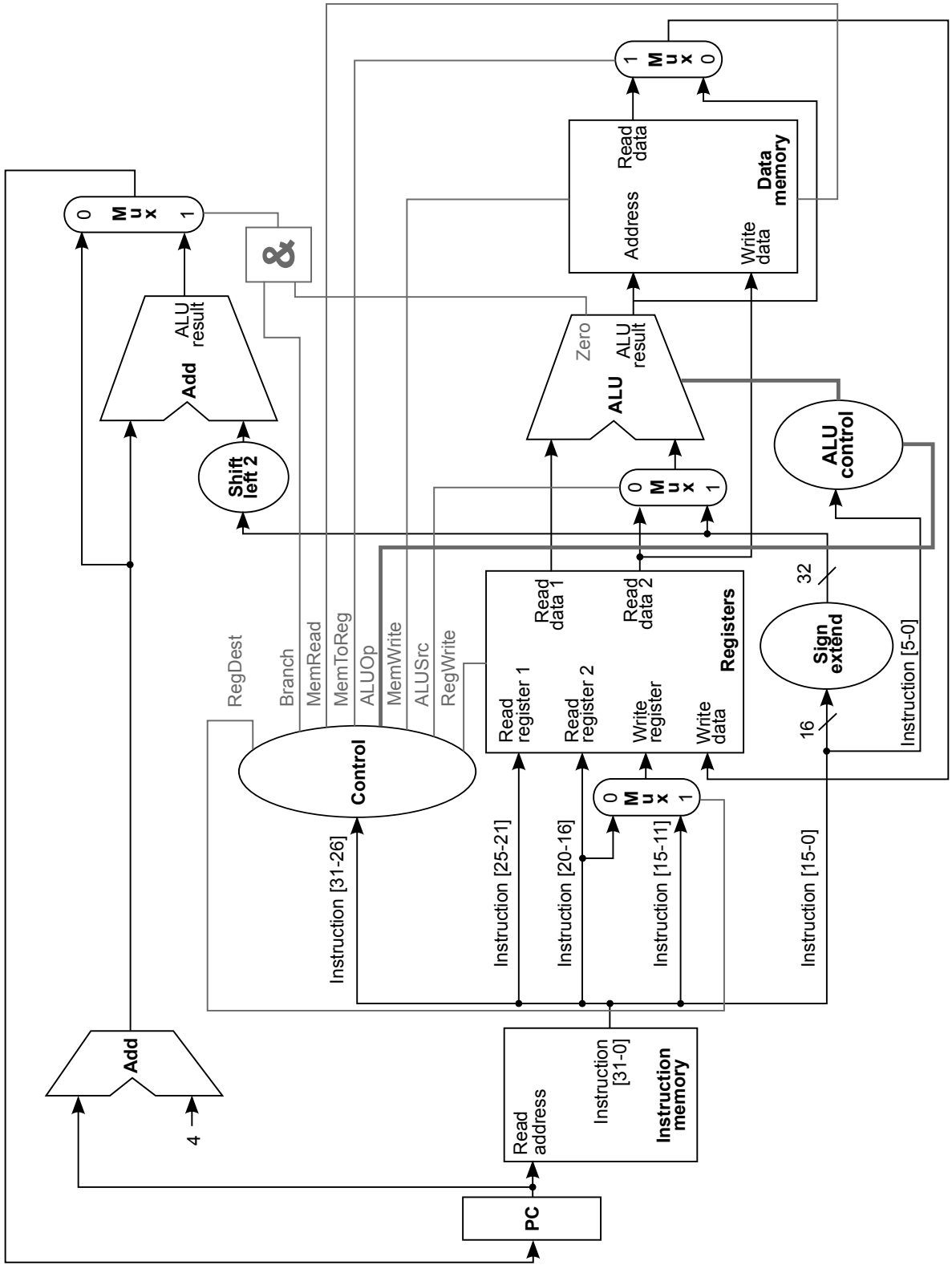

Abbildung 5: Einzyklenimplementierung

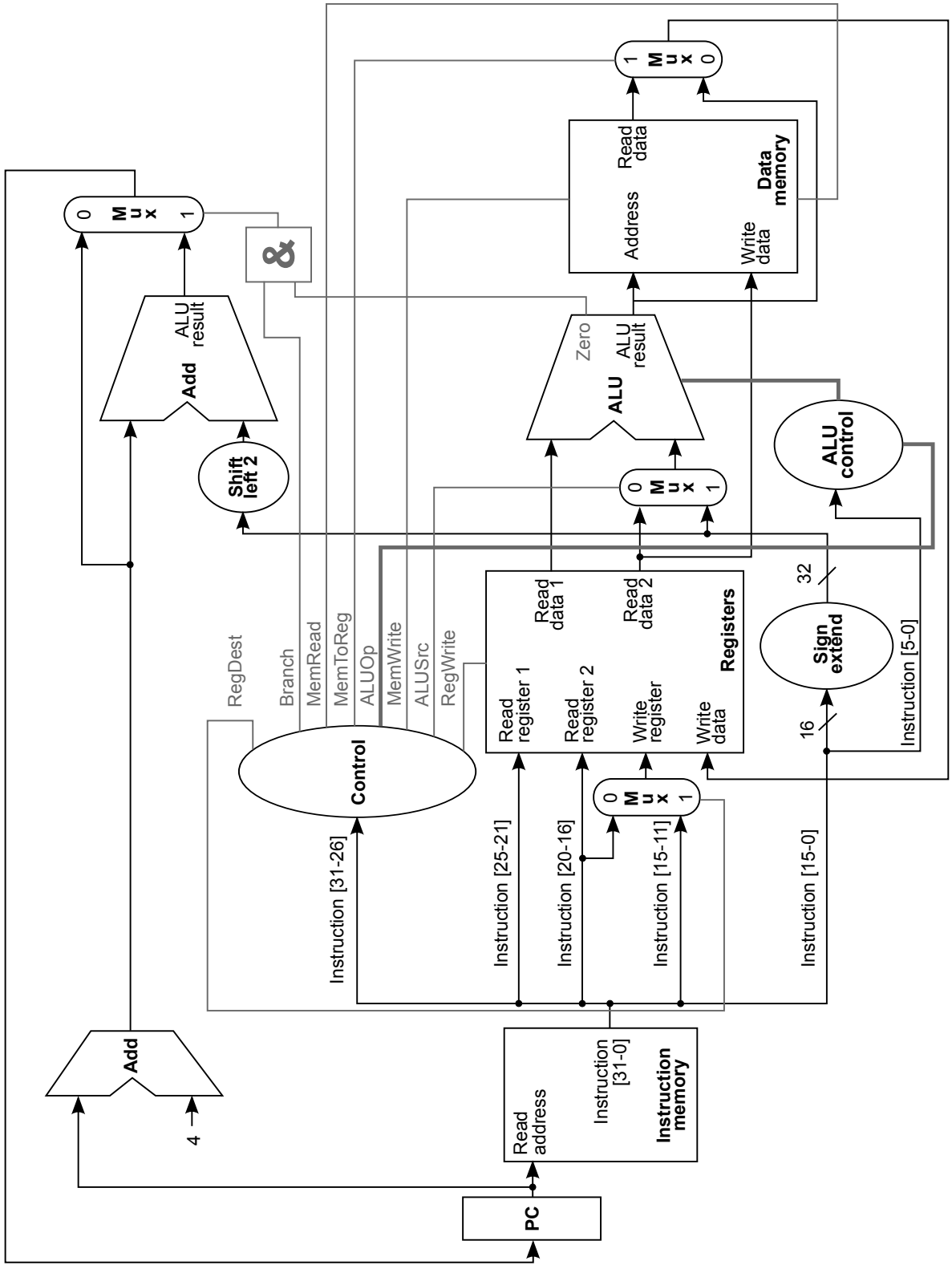

Abbildung 6: Ersatz Einzyklenimplementierung, **ungültige Lösung streichen**

Aufgabe 5: (Leistungsbewertung) 17 Punkte

a) Ein JAVA-Anwendung braucht 15 s auf einem Desktop Computer. Ein neuer Java-Compiler schafft es, die Zahl der Befehle auf 60 % zu reduzieren. Allerdings erhöht sich dann der CPI-Wert um den Faktor 1,1. Wie lange benötigt die Anwendung mit dem neuen JAVA-Compiler?

Bitte kreuzen Sie die richtige Antwort an.

(2 Punkte)

- ☐ $\frac{15 \cdot 0,6}{1,1} s = 8,2s$
- ☐ $15 \cdot 0,6 \cdot 1,1s = 9,9s$
- ☐ $\frac{15 \cdot 1,1}{0,6} s = 27,5s$
- ☐ keine Lösung ist richtig

b) Ein Programm braucht 100 s auf einem Desktop Computer, davon werden 80 s für die Ausführung von Multiplikationsbefehlen benötigt. Um welchen Faktor müsste der Multiplikationsbefehl beschleunigt werden, damit das Programm eine Laufzeit von 20 s erreicht?

Bitte kreuzen Sie die richtige Antwort an.

(2 Punkte)

- ☐ 4
- ☐ 5
- ☐ 20
- ☐ keine Lösung ist richtig

c) Für ein Beispielprogramm wurden auf zwei verschiedenen Rechnern die folgenden Messergebnisse erzielt:

Messung	Rechner A	Rechner B
Anzahl der Befehle	$10 \cdot 10^9$	$8 \cdot 10^9$
Taktrate	4 GHz	4 GHz
CPI	1,0	1,1

Bitte ergänzen Sie die folgenden Aussagen:

(4 Punkte)

Rechner A hat eine MIPS-Rate von _____

Rechner B hat eine MIPS-Rate von _____

Die Laufzeit auf Rechner A beträgt _____

Die Laufzeit auf Rechner B beträgt _____

d) Für einen Prozessor mit einem idealen CPI-Wert von 2 und einer Taktfrequenz von 1 GHz soll der tatsächliche CPI-Wert bei Speicherfehlgriffen ermittelt werden. Die Fehlgriffsraten für den Cache seien 2 % für Befehle und 4 % für Daten. Bei einem Fehlgriff werden 100 ns Strafzeit zusätzlich benötigt („Miss Penalty“). Wie groß ist der tatsächliche CPI-Wert, wenn 40 % der Befehle auf Daten zugreifen?

Bitte kreuzen Sie die richtige Antwort an.

(3 Punkte)

- ☐ 6
- ☐ 8
- ☐ 5,6
- ☐ keine Lösung ist richtig

e) Bei dem Prozessor aus Aufgabe **d)** soll die Taktrate verdoppelt werden. Am Speichersystem wird jedoch nichts geändert (d.h. Fehlgriffsraten und die Strafzeit für einen Fehlgriff bleiben gleich). Wie verhält sich die Laufzeit t_{alt} für ein Programm auf dem alten Prozessor zur Laufzeit t_{neu} auf dem Prozessor mit der doppelten Taktrate?

Bitte kreuzen Sie den richtigen Wert für t_{alt}/t_{neu} an.

(3 Punkte)

- ☐ 5,6/4,6
- ☐ 8/7
- ☐ 2
- ☐ keine Lösung ist richtig

f) Murx Electronics möchte 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 (unendlich viele Befehle) der Pipeline-Implementierung gegenüber einer sequentiellen Abarbeitung der Befehle?

Bitte kreuzen Sie alle richtigen Antworten an.

(3 Punkte)

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