Universität Paderborn

Institut *Elektrotechnik und Informationstechnik*

Fachgebiet *Datentechnik*

Prof. Sybille Hellebrand

Klausur

Technische Informatik / Grundlagen der Rechnerarchitektur

8. Oktober 2010

Bearbeitungszeit: 90 Minuten

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

a) Für die Sprungvorhersage einer Pipeline sind die 3 Vorhersagestrategien always_taken, Prädiktor A und Prädiktor B experimentell zu vergleichen. Die Einheiten A und B sind den Moore-Automaten aus Abbildung 1 zu entnehmen.
(Hinweis: Das Ein-/Ausgabesymbol T bedeutet dabei „taken“, d.h. der Sprung wird ausgeführt, und das Symbol NT bedeutet „not taken“, d.h. der Sprung wird nicht ausgeführt.)

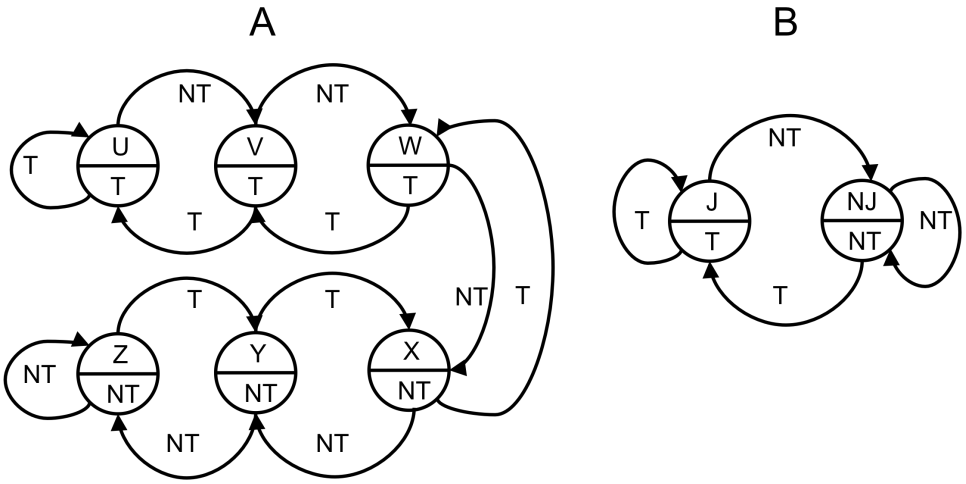


Abbildung 1: Moore-Automaten zur Sprungvorhersage

Für ein Benchmark-Programm sei die Sprungfolge: T T T NT NT T NT NT gegeben. Geben Sie für alle 3 Prädiktoren die vorhergesagte Sprungfolge und die Anzahl korrekt vorhergesagter Sprünge an! Prädiktor A sei hierzu mit U initialisert und B sei initial im Zustand J. (6 Punkte)

Vorhergesagte Sprungfolge für:

always_taken: _____

Prädiktor A: _____

Prädiktor B: _____

Anzahl der korrekt vorhergesagten Sprünge für:

always_taken: _____

Prädiktor A: _____

Prädiktor B: _____

b) Geben Sie eine Sprungfolge der Länge 5 an, für die Prädiktor B (initialisiert mit J) keinen Sprung korrekt vorhersagt! (1 Punkt)

Sequenz: _____

_____

c) Untersuchen Sie die Codesequenz P1 für eine MIPS-Pipeline mit Harvard-Architektur und Forwarding. Geben Sie die Anzahl an Taktzyklen an, die benötigt werden um P1 abzuarbeiten! Geben Sie weiterhin an, zwischen welchen Instruktionen und Pipelinestufen Forwarding benötigt wird! (7 Punkte)

```
P1  A:  lw   $t1,  100($t2)
    B:  add  $t3,  $t1,  $t2
    C:  add  $t7,  $t7,  $t7
    D:  add  $t8,  $t7,  $t6
```

Taktzyklen: _____

Forwarding zwischen Instruktionen: _____

Forwarding zwischen Pipelinestufen: _____

d) Optimieren Sie die Sequenz P1 derart, dass weniger Takte zur Abarbeitung notwendig sind! (2 Punkte)
(Hinweis: Es genügt die Angabe der Label.)

Reihenfolge: _____

_____

Aufgabe 2: (Cache) 20 Punkte

a) Es gibt grundsätzlich zwei verschiedene Strategien, um bei Schreibzugriffen die Daten in Cache und Hauptspeicher konsistent zu halten. Nennen Sie diese beiden Strategien und erläutern Sie kurz die wesentlichen Unterschiede bei einem „Read-Miss“. (4 Punkte)

b) 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 direkt abgebildeter (direct mapped) Cache organisiert. Bestimmen Sie die benötigte Anzahl von Bits für Tag, Index und Offsets.
(4 Punkte)

Adressbus:_____

Tag:_____

Index:_____

Wordoffset:_____

Byteoffset:_____

c) Gegeben sei folgende MIPS-Assembler Codesequenz; das Register s0 sei mit der Adresse ...1000001000 belegt: (12 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 **b)** 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“.

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	

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	

↑(Ersatztabelle, ungültige Lösung streichen!)↑

Aufgabe 3: (Leistungsbewertung) 17 Punkte

a) Die Firma Padertronics möchte die Prozessoren für ihre Netbooks verbessern. Eine Analyse typischer Programme hat die in der Tabelle gezeigte Befehlsverteilung ergeben. (4 Punkte)

Multiplikation	Load/Store	Andere
20 %	50 %	30 %

Es werden folgende Verbesserungen diskutiert:

- Verbesserter Multiplikationsbefehl (Speedup 4 für Multiplikationen).
- Verbesserte LOAD/STORE-Befehle (Speedup 2 für Speicherbefehle).

Die Ausführungszeit für ein Programm mit dem ursprünglichen Prozessor werde mit t_{alt} bezeichnet, die Ausführungszeit bei verbesserter Multiplikation mit t_{mult}, die Ausführungszeit bei verbesserten LOAD/STORE-Befehlen mit t_{LS} und die Ausführungszeit mit beiden Verbesserungen mit $t_{mult/LS}$. Geben sie Werte für t_{mult}, t_{LS} und $t_{mult/LS}$ in Abhängigkeit von t_{alt} an.

t_{mult} = _____ t_{alt}

t_{LS} = _____ t_{alt}

$t_{mult/LS}$ = _____ t_{alt}

b) Wie müsste die Befehlsverteilung aussehen, damit beide Verbesserungen in Aufgabe **a)** zum gleichen Ergebnis führen ($t_{mult} = t_{LS}$). Kreuzen Sie alle richtigen Antworten in der nachfolgenden Tabelle an. (4 Punkte)

	Multiplikation	Load/Store	Andere
☐	30 %	20 %	50 %
☐	20 %	30 %	50 %
☐	30 %	45 %	25 %
☐	keine Verteilung liefert $t_{mult} = t_{LS}$		

c) Der Chefingenieur von Padertronics beauftragt sein Team außerdem, den Einfluss der Speicherorganisation auf die Rechnerleistung zu analysieren. Ausgangspunkt sind die Befehlshäufigkeiten und Befehlszyklen in der folgenden Tabelle. (4 Punkte)

	ALU	Load/Store	Branch
Häufigkeit	50 %	30 %	20 %
Zyklen	4	5	2

Beim Zugriff auf den Befehlscache gibt es mit 5 % Wahrscheinlichkeit einen Konflikt. Auf den Datencache kann nur mit Load/Store-Befehlen zugegriffen werden, wobei 10 % Fehlgriffe zu erwarten sind. In beiden Fällen kommen 40 Zyklen („Strafzyklen“) zur normalen Ausführungszeit hinzu. Berechnen Sie den idealen CPI-Wert CPI_{ideal}, die durchschnittliche Zahl von Strafzyklen pro Befehl sowie den realistischen CPI-Wert mit Fehlgriffen (CPI_{real}).

CPI_{ideal} = _____

durchschnittliche
Strafzyklen pro Befehl = _____

CPI_{real} = _____

d) Die Konkurrenzfirma Murx Electronics arbeitet an einer Prozessor-Variante mit Pipelining. 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 in der Tabelle alle richtigen Antworten an: (5 Punkte)

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

Aufgabe 4: (Assembler) 17 Punkte

Gegeben ist das folgende MIPS-Assemblerprogramm.

```
1   .data
2   str: .asciiz "ein esel lese nie"
3
4   .text
5   main:   la      $a0, str
6           la      $a1, str          # Funktion von loop?
7   loop:   addi    $a1, $a1, 1       #
8           lb      $t0, 0($a1)       #__________________________
9           bne     $t0, $zero, loop  #
10          addi    $a1, $a1, -1      #__________________________
11
12          jal     funct             #__________________________
13          add     $a0, $zero, $v0
14          addi    $v0, $zero, 1     #Integer Ausgabe
15          syscall
16          addi    $v0, $zero, 10
17          syscall                   #Programm beenden
18
19  funct:  ______________________    #Stackspeicher reservieren
20
21          ______________________    #Rücksprungadresse sichern
22          slt     $t0, $a0, $a1
23          beq     $t0, $zero, X
24          lb      $t0, 0($a0)       #load byte
25          lb      $t1, 0($a1)       #load byte
26          bne     $t0, $t1, Y
27
28          ______________________    #Rekursionsaufruf
29                                    #funct($a0+1,$a1-1)
30          ______________________
31
32          ______________________
33          j       Z
34  X:      addi    $v0, $zero, 1   #Rückgabe = 1
35          j       Z
36  Y:      addi    $v0, $zero, 0   #Rückgabe = 0
37  Z:
38          ______________________ #lade Rücksprungadresse
39
40          ______________________ #Stack freigeben
41          jr      $ra
```

a) Ergänzen Sie die fehlenden Teile des Assemblerprogramms (auch die teilweise fehlenden Kommentare). (10 Punkte)

b) Was sind die Abbruchbedingungen für die Rekursion? (2 Punkte)

c) Wie lautet die Ausgabe des Programms? (1 Punkt)

d) Wie muss die an funct übergebene Zeichenkette (abgelegt im Speicher von Adresse $a0 bis $a1) aussehen, um als Rückgabewert eine 1 zu erhalten? (4 Punkte)

Aufgabe 5: (Datenpfad) 20 Punkte

Abbildung 2 zeigt die aus der Vorlesung bekannte MIPS Einzyklenimplementierung.

a) Für eine I-Typ Instruktion wird die Einheit *Vorzeichenerweiterung (Sign extend)* benötigt. Geben Sie für die folgenden zwei Eingangsbelegungen in dezimaler Schreibweise die jeweilige Eingangs- und Ausgangsbelegung der Vorzeichenerweiterung in hexadezimaler Schreibweise an.
(Hinweis: Busbreiten beachten! Zahlendarstellung im Zweierkomplement!)
(2 Punkte)

		Eingang	Ausgang
256_{10}	$\Rightarrow$	0x_____	0x_____
-1_{10}	$\Rightarrow$	0x_____	0x_____

b) Welcher Gattertyp wird für das mit ? in der Abbildung 2 gekennzeichnete Gatter benötigt, damit die beq (branch on equal) Instruktion korrekt ausgeführt wird?
(1 Punkt)

_____

c) Markieren Sie in Abbildung 2 die aktiven (mit relevanten Werten belegten) Leitungen im Datenpfad für eine Operation des R-Typs. (2 Punkte)

d) 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. (5 Punkte)

Instr.	RegDst	ALUSrc	MemtoReg	RegWrite
R-Type				

Instr.	MemRead	MemWrite	Branch	ALUOp[1:0]
R-Type				10

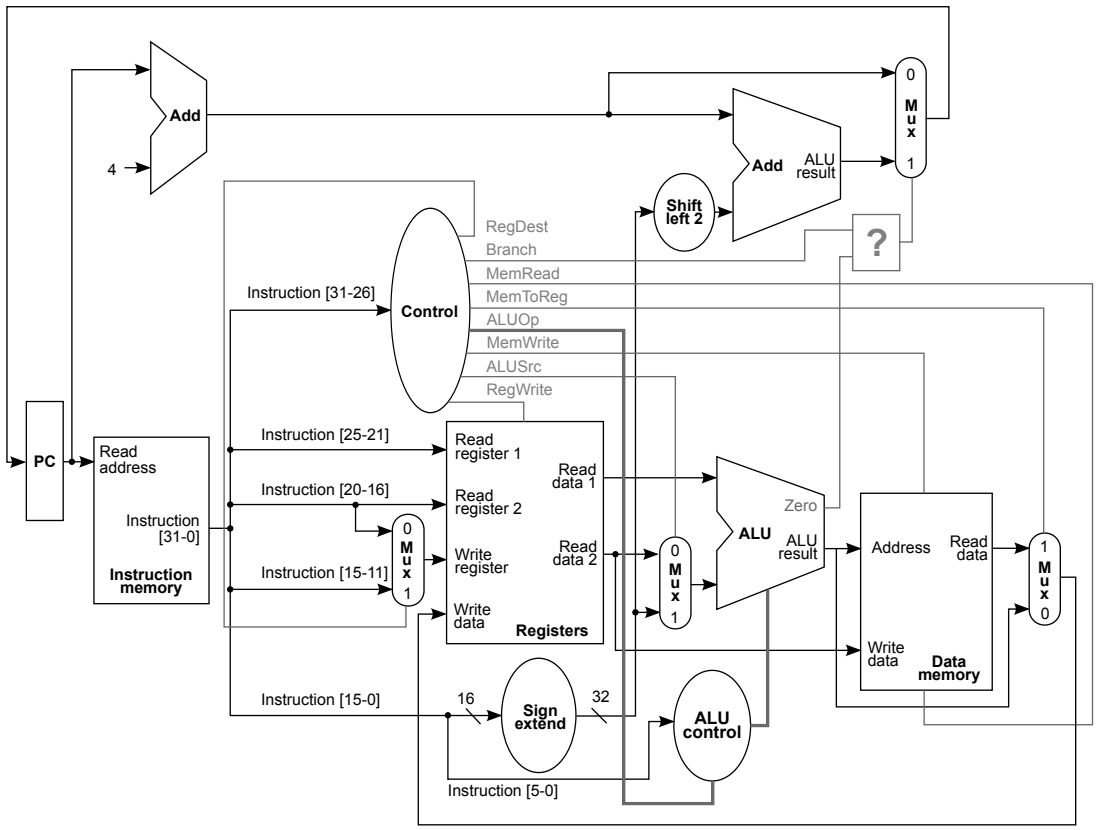
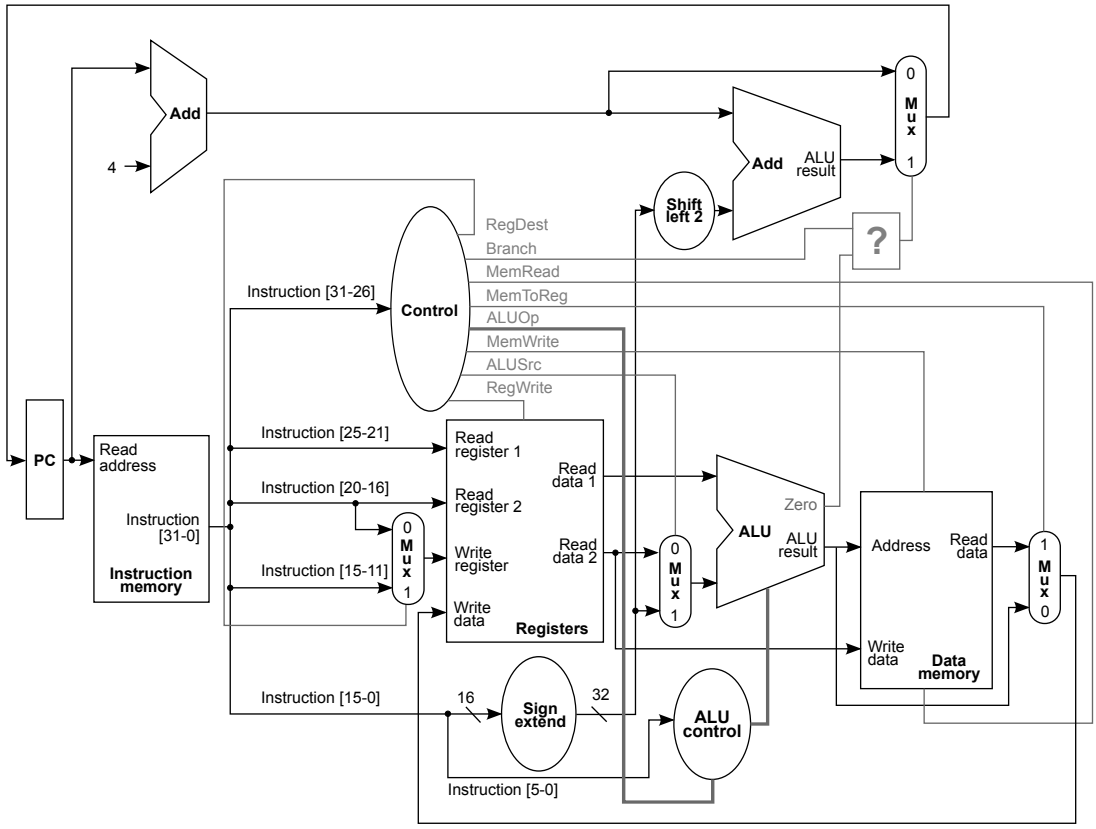


Abbildung 2: Einzyklenimplementierung



(Ersatz zu Abb. 2, ungültige Lösung streichen!)

e) Erweitern Sie den MIPS Einzyklendatenpfad so, dass der Befehl `Jump` realisiert werden kann. Tragen Sie die Erweiterungen in die Abbildung 3 ein.
(Hinweis: PC_{neu} erhält man durch Aneinanderfügen von (PC+4)[31..28] und (Instruction[25..0] << 2).) (5 Punkte)

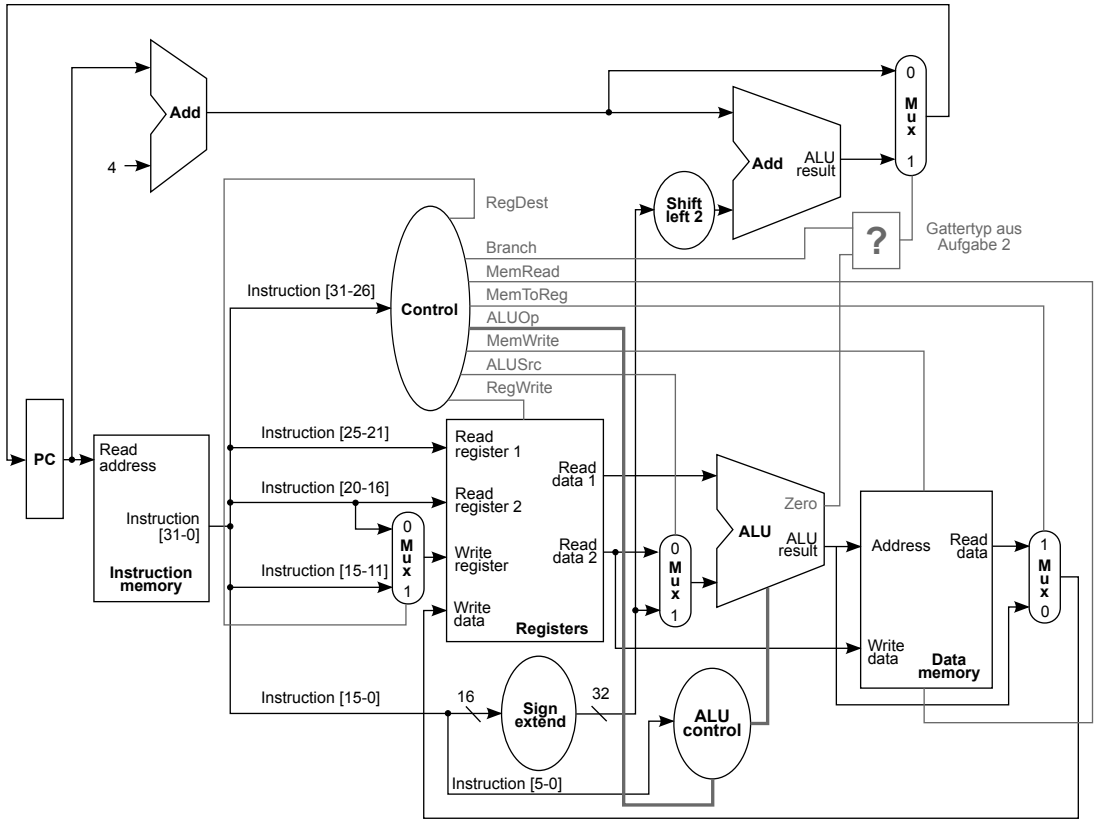
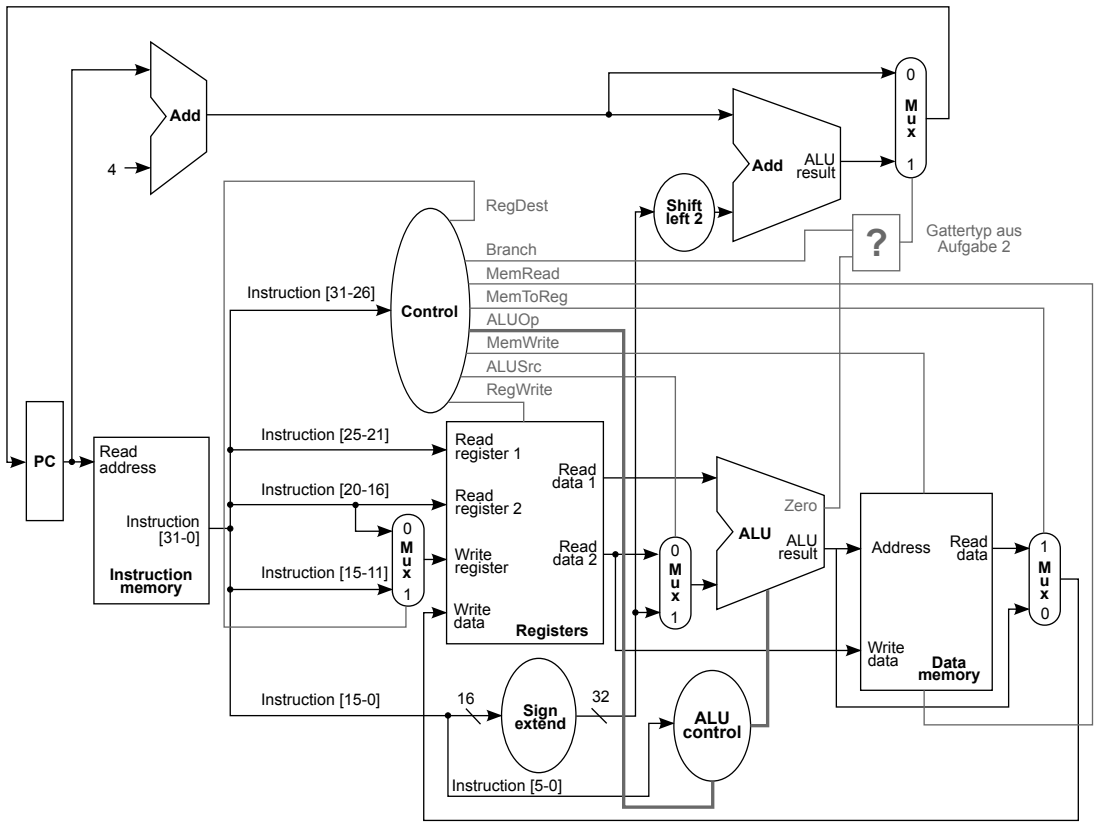


Abbildung 3: Einzyklenimplementierung

f) Definieren Sie nun die Steuersignale des Kontrollers für eine `j`-Instruktion und tragen Sie diese in die nachstehende Tabelle ein. Benutzen Sie so viele "don't care"- Werte (X) wie möglich. (5 Punkte)

Instr.	RegDst	ALUSrc	MemtoReg	RegWrite	
j					

Instr.	MemRead	MemWrite	Branch	ALUOp[1:0]	
j					



(Ersatz zu Abb. 3, ungültige Lösung streichen!)