Klausur zur Vorlesung

Grundlagen der Rechnerarchitektur / Technische Informatik (GRA/TI)

Prof. Marco Platzner
Fachgebiet Technische Informatik
Universität Paderborn

20.02.2013

- Die Bearbeitungsdauer beträgt für alle Studenten **90 Minuten.** Es sind **alle 5 Aufgaben** zu bearbeiten.
- Es sind keine Hilfsmittel zugelassen.
- Schreiben Sie nicht mit Bleistift oder Rotstift.
- Verwenden Sie kein eigenes Papier. Bei Bedarf bekommen Sie Papier bei der Klausuraufsicht.
- Schreiben Sie auf jedes Blatt (auch auf das Konzeptpapier) in Blockschrift Ihren Namen und Ihre Matrikelnummer.
- Bei mehreren präsentierten Lösungen wird die Aufgabe nicht gewertet! Streichen Sie daher bei Angabe mehrerer Lösungsansätze die nicht zu bewertenden Lösungen durch! Verwenden Sie kein Tipp-Ex.
- Abschreiben und abschreiben lassen oder Hilfe Dritter führt zum Nichtbestehen der Klausur.

Nachname: _____

Vorname: _____

Matrikelnummer: _____

Studiengang: _____

Aufkleber

Aufgabe	**1**	**2**	**3**	**4**	**5**	$\sum$
Punkte	**15**	**25**	**15**	**20**	**15**	90
Erreicht						

Aufgabe 1 (Multiple Choice) **[15 Punkte]**

Bei den folgenden Fragen können keine, eine oder mehrere Antworten richtig sein. Kreuzen Sie die richtigen Antworten deutlich an.

(a) Bei welchen Rechnerarchitekturen werden in den Instruktionen Registeroperanden explizit spezifiziert?

- ☐ Load/Store Maschinen
- ☐ Akkumulatormaschinen
- ☐ Stackmaschinen
- ☐ Maschinen mit Register-Speicher Operationen

(b) Der CPI-Wert hängt ab von ...

- ☐ der verwendeten Programmiersprache
- ☐ dem Instruktionssatz
- ☐ der mikroelektronischen Technologie
- ☐ der Rechnerarchitektur

(c) Aufgaben eines Loaders sind ...

- ☐ Festlegen eines ausreichend grossen Adressbereichs im Hauptspeicher für Programmtext und Daten
- ☐ Erzeugen einer Programmdatei im Objektformat
- ☐ Kopieren der Parameter auf den Stack
- ☐ Initialisieren der Register

(d) Ein Victim Cache ...

- ☐ ist ein Cache mit einer miss-rate grösser als 80%
- ☐ reduziert die miss-penalty
- ☐ erhöht die miss-penalty
- ☐ ist ein zusätzlicher Cache, der kürzlich ersetzte Cacheblöcke speichert

(e) Vorteile des dynamischen Pipeline-Scheduling gegenüber dem statischen Pipeline-Scheduling sind ...

- ☐ der Compilerentwurf wird vereinfacht
- ☐ der Code läuft effizient auf jeder Implementierung der Instruktionssatzarchitektur
- ☐ der Hardwareaufwand für den Prozessor wird reduziert

Aufgabe 2 (Assembler) **[25 Punkte]**

```
0x40:   main:    addi    $a1,   $zero, 2
0x44:            addi    $a2,   $zero, 1
0x48:            jal     funct
0x4c:            li      $v0,   10              # Programm beenden
0x50:            syscall                        # Systemaufruf

0x54:   funct:   ____    ____,  ____, ____      # Platz auf Stack schaffen
0x58:            ____    ____,  ____            # und Registerinhalte sichern
0x5c:            add     $t5,   $zero, $a1      # $a1 sichern
0x60:            jal     L1
0x64:            add     $s3,   $zero, ____     # 1. Ergebnis in $s3 sichern
0x68:            add     $t5,   $zero, $a2
0x6c:            jal     L1
0x70:            add     $s4,   $zero, ____     # 2. Ergebnis in $s4 sichern
0x74:            sub     $t5,   $a1,   $a2
0x78:            jal     L1
0x7c:            add     $s5,   $zero, ____     # 3. Ergebnis in $s5 sichern
0x80:            mult    $s4,   $s5
0x84:            mflo    $t3
0x88:            div     $v1,   $s3,  $t3       # $v1:= $s3 DIV $t3
0x8c:            lw      $ra,   0($sp)
0x90:            addi    $sp,   $sp,   4
0x94:            jr      $ra

0x98:   L1:      ____    ____,  ____, ____      # Platz auf Stack schaffen
0x9c:            ____    ____,  ____            # und Registerinhalte sichern
0xa0:            ____    ____,  ____
0xa4:            slti    $t0,   $t5,   1
0xa8:            beq     $t0,   $zero, L2
0xac:            addi    $v1,   $zero, 1
0xb0:            addi    $sp,   $sp,   8
0xb4:            jr      $ra

0xb8:   L2:      subi    $t5,   $t5,   1
0xbc:            jal     L1
0xd0:            ____    ____,  ____            # Registerinhalte vom Stack
0xd4:            ____    ____,  ____            # laden und Stack wieder
0xd8:            ____    ____,  ____, ____      # freigeben
0xdc:            mult    $v1,   $t5
0xe0:            mflo    $v1
0xe4:            jr      $ra
```

Gegeben sei das Assembler- Programm von der vorherigen Seite zur Berechnung des Binomialkoeffizienten

$$\binom{n}{k} = \frac{n!}{k! \cdot (n-k)!},$$

welches an Speicheradresse `0x40` beginnt. Der Stackpointer sei mit `0x0ff8` initalisiert, siehe Abbildung 1. Gehen Sie davon aus, dass der Befehl `div` vom MIPS Prozessor direkt ausführbar ist.

(a) Füllen Sie die Lücken im Assemblercode aus.

(b) Geben Sie die Inhalte der Register `$t5`, `$v1`, `$ra` und `$sp` nach jedem Zeitpunkt t, wenn ein Sprung zu Label L1 durchgeführt wurde, in folgender Tabelle an. Sollte der Inhalt eines Registers unbekannt sein, schreiben Sie ein 'X'.

	t=1	t=2	t=3	t=4	t=5	t=6	t=7
$t5							
$v1							
$ra							
$sp							

Tabelle 1: Registerbelegung

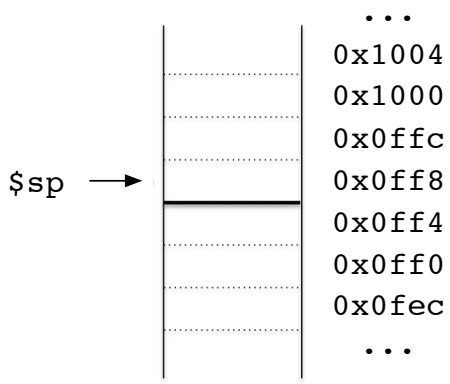


Abbildung 1: Stack

	t=1	t=2	t=3	t=4	t=5	t=6	t=7
$t5							
$v1							
$ra							
$sp							

Tabelle 2: Registerbelegung - Ersatzlösung, **ungültige Lösung bitte streichen**!

(c) Welcher Wert ist im Register `$v1` am Ende des Programms abgespeichert?

Aufgabe 3 (Amdahlsches Gesetz) **[15 Punkte]**

Einem Forscher ist es gelungen einen Computer derart zu beschleunigen, dass ein Testprogramm von diesem nun 2,5 mal schneller ausgeführt wird als bisher. Er hat dafür ausschließlich Änderungen an den Speicherbausteinen und der Fließkommaeinheit vorgenommen. Der Computer führt nun sämtliche Fließkommaoperationen 2 mal schneller und alle Speicheroperationen 4 mal schneller aus. Bei einem Testlauf mit dem verbesserten Computer wird eine Ausführungszeit von 11,5 Sekunden gemessen. Auf dem alten Computer hatte der Forscher damals mit Hilfe eines Profilers gemessen, dass sein Testprogramm zu 5% der Laufzeit Integeroperationen und zu 30% der Laufzeit Fließkommaoperationen ausführt. Leider sind ihm seine Messergebnisse zum Anteil der Speicheroperationen und allen weiteren Operationen abhanden gekommen.

(a) Wie hoch war der Laufzeitanteil der Speicheroperationen des Testprogramms auf dem alten Computer?

(b) Markieren Sie alle Werte in der nachfolgenden Auflistung, die für die Beantwortung von Frage (a) zwingend erforderlich sind:

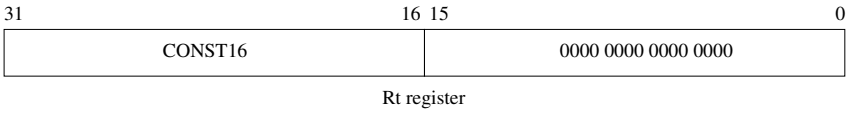
☐	2,5	(Speedup Forschungsprogramm)
☐	2	(Verbesserung Fließkommaoperationen)
☐	4	(Verbesserung Speicheroperationen)
☐	5%	(Laufzeitanteil Integeroperationen)
☐	30%	(Laufzeitanteil Fließkommaoperationen)
☐	11,5 Sek.	(Programmlaufzeit auf erneutertem Computer)

Aufgabe 4 (Erweiterung der Mehrzyklenimplementierung) **[20 Punkte]**

Erweitern Sie die in der Vorlesung besprochene Mehrzyklenimplementierung derart, dass die Instruktion `lui` (load upper immediate) abgearbeitet werden kann.

$$\texttt{lui} \quad \texttt{R}_\texttt{t}, \texttt{CONST16} \qquad \texttt{R}_\texttt{t} = \texttt{CONST16} * 2^{16}$$

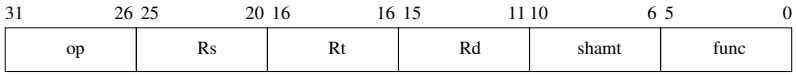
d.h.



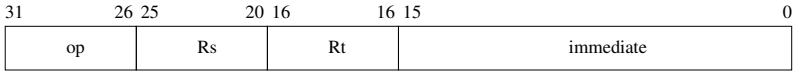
Gehen Sie folgendermaßen vor:

(a) Kreuzen Sie den Instruktionstyp an, der sich für die `lui` Instruktion eignet:

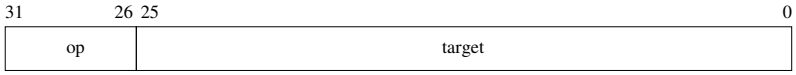
☐ R-Typ



☐ I-Typ



☐ J-Typ



Begründen Sie Ihre Antwort:

(b) Erweitern Sie - falls nötig - den Datenpfad und die Steuersignale und zeichnen Sie diese Erweiterungen in Abbildung 2 ein, damit die Instruktion `lui` abgearbeitet werden kann. Für die Implementierung der logischen Verschiebung nach links nehmen Sie an, dass die ALU um einen barrel shifter erweitert wurde. Ein barrel shifter ist eine kombinatorische Schaltung, die ein Eingangswort um eine variable Anzahl von Stellen, vorgegeben durch einen Kontrolleingang, schiebt. Sie können annehmen, dass die ALU einen zusätzlichen Eingang hat, an dem ein binäres Wort angelegt wird, das die Anzahl der Stellen darstellt, um die der untere ALU-Eingang geschoben wird.

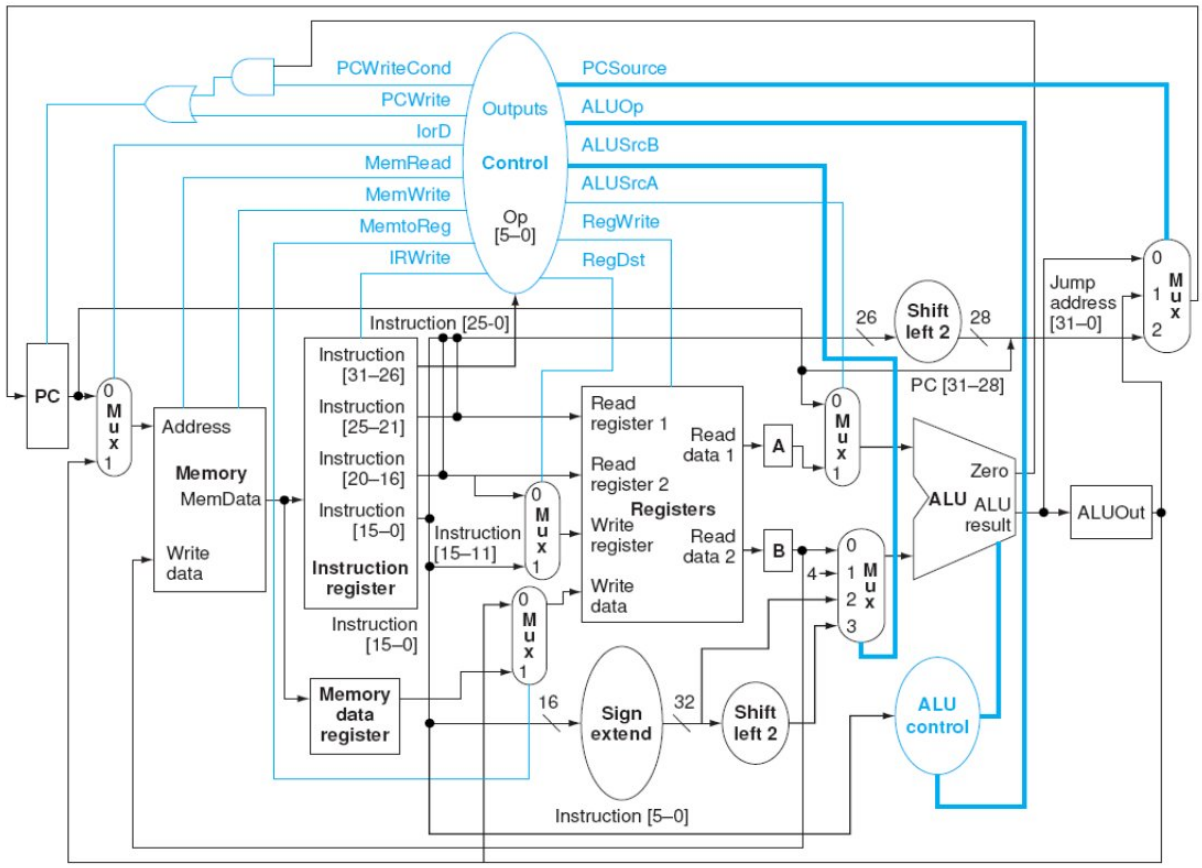


Abbildung 2: Datenpfad und Steuersignale

(c) Vervollständigen Sie die erweiterte ALU-Kontroller Tabelle:

"opcode"	AluOP[1:0]	funct[5:0]	ALUOperation[3:0]	Funktion der ALU
lw	00	XXXXXX	0010	add
sw	00	XXXXXX	0010	add
beq	01	XXXXXX	0110	sub
R-Typ	10	100000	0010	add
R-Typ	10	100010	0110	sub
R-Typ	10	100100	0000	and
R-Typ	10	100101	0001	or
R-Typ	10	101010	0111	slt
lui			1000	sll

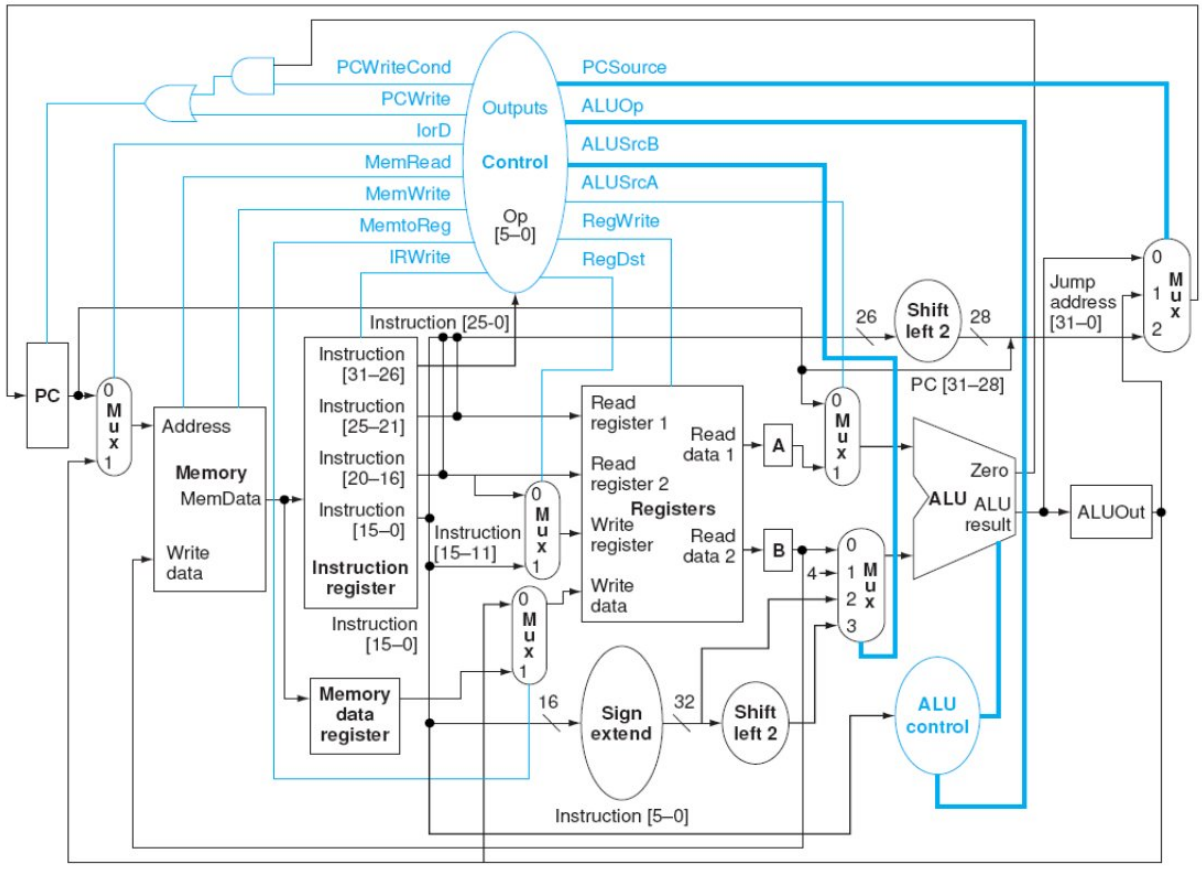


Abbildung 3: Datenpfad und Steuersignale - Ersatzlösung, **ungültige Lösung bitte streichen!**

(d) Beschreiben Sie den Ablauf des Instruktionszyklus für `lui` in der in der Vorlesung verwendeten Pseudocode-Notation. Geben Sie für jeden der Taktschritte `EX`, `MEM` und `WB` die Signalzuweisungen an. `IF` und `ID` bleiben unverändert.

`IF` $\quad$ `IR` $\Leftarrow$ `Memory[PC]`
$\quad$ `PC` $\Leftarrow$ `PC` $+ 4$

`ID` $\quad$ `A` $\Leftarrow$ `Reg[IR[25:21]]`
$\quad$ `B` $\Leftarrow$ `Reg[IR[20:16]]`
$\quad$ `ALUOut` $\Leftarrow$ `PC` $+$ `(sign-extend(IR[15:0])` $\ll 2)$

`EX` ...

`MEM` ...

`WB` ...

(e) Welchen CPI-Wert hat die Instruktion `lui`?

(f) Vervollständigen Sie den Automatengraphen für den um `lui` erweiterten Kontroller.

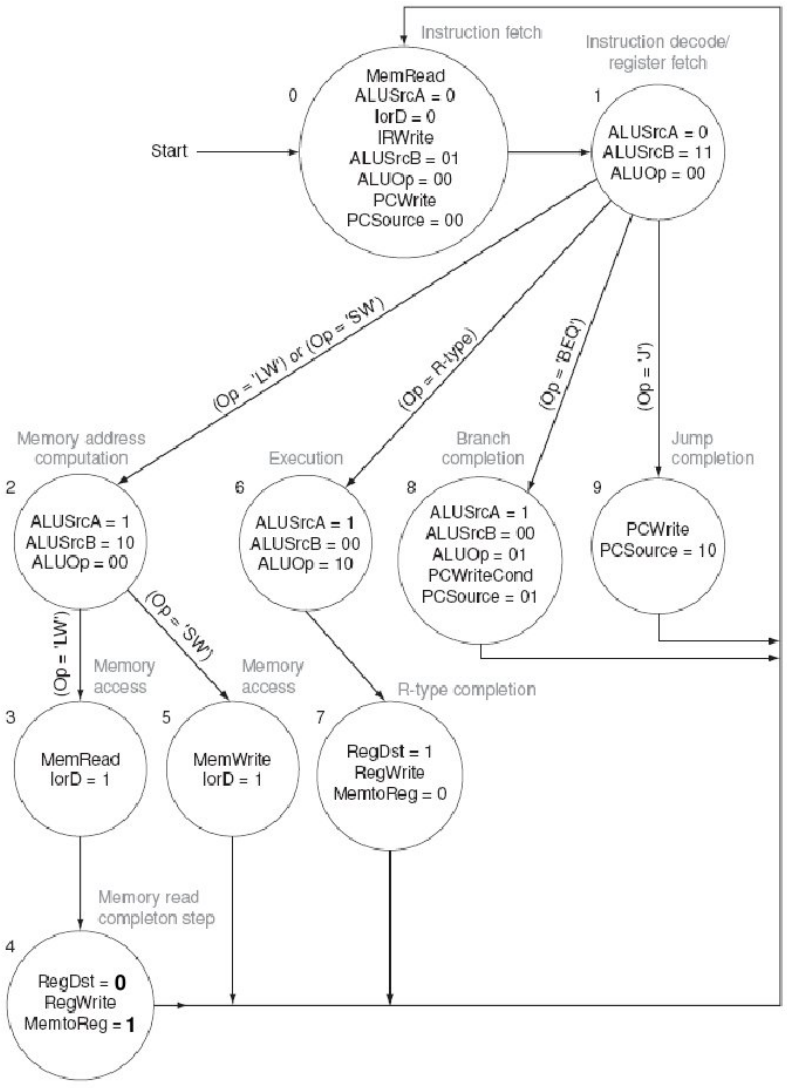


Abbildung 4: Automatengraph

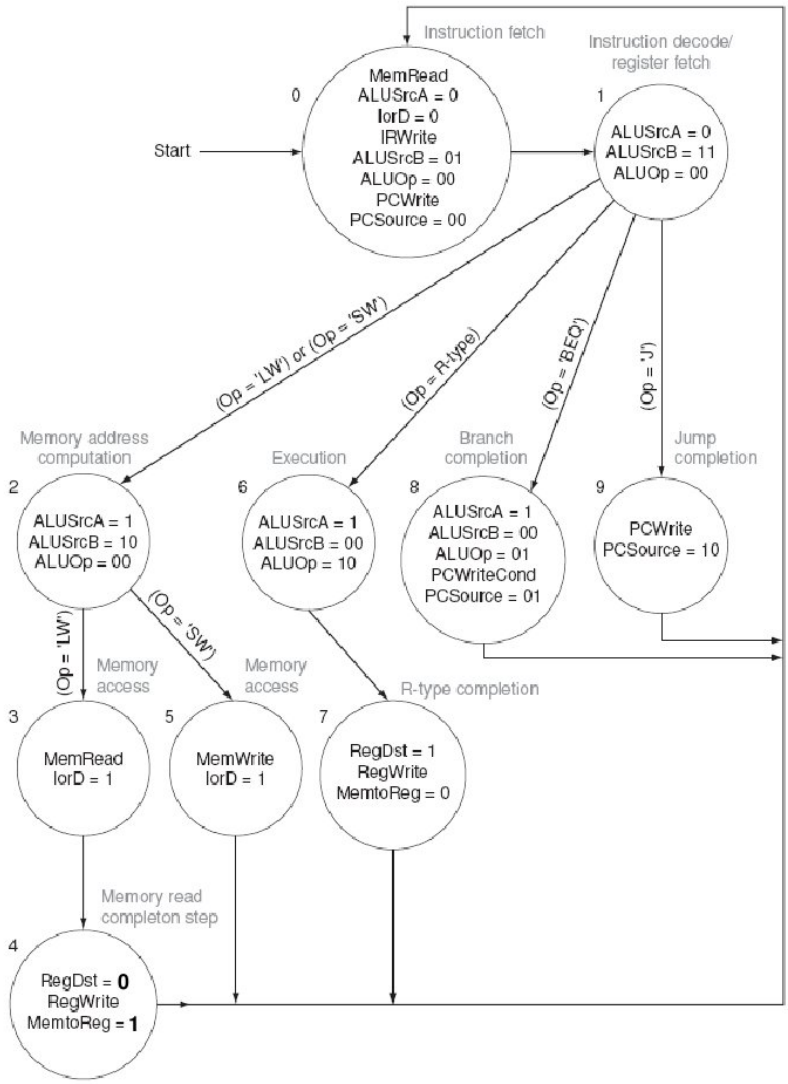


Abbildung 5: Automatengraph - Ersatzlösung, **ungültige Lösung bitte streichen!**

Aufgabe 5 (Multi-Level Caches) **[15 Punkte]**

Gegeben sei ein 2 GHz Prozessor mit 2-stufiger Cachehierarchie. Der L1 Cache besitzt eine Hit-Time von 0,5ns und eine Miss-Rate von 10%. Der L2 Cache besitzt eine Hit-Time von 24 Taktzyklen, eine Hit-Rate von 95% und Miss-Penalty von 200ns. Beide Caches benutzen dieselbe Blockgröße.

(a) Wie hoch ist die mittlere Zugriffszeit auf den 2-stufigen Cache?

(b) Den Entwicklern des Prozessors bietet sich eine Alternative zu dem 2-stufigen Cache. Der neue einstufige Cache besitzt eine Hit-Rate von 99%, eine Miss-Penalty von 200ns und eine Hit-Time von 0,5ns. Wie hoch ist die mittlere Zugriffszeit für diesen Cache? Sollte man diese Alternative vorziehen?

Konzeptpapier: Falls der Platz unter den einzelnen Aufgaben nicht ausreicht, können Sie diese Seiten für Zwischenrechnungen nutzen. Bitte Lösung und Lösungsweg eindeutig mit der Aufgabennummer markieren!