

Prof. L. Thiele

Technische Informatik 1 - HS 2010

Lösungsvorschläge für Übung 6

Datum: 11.11.2010

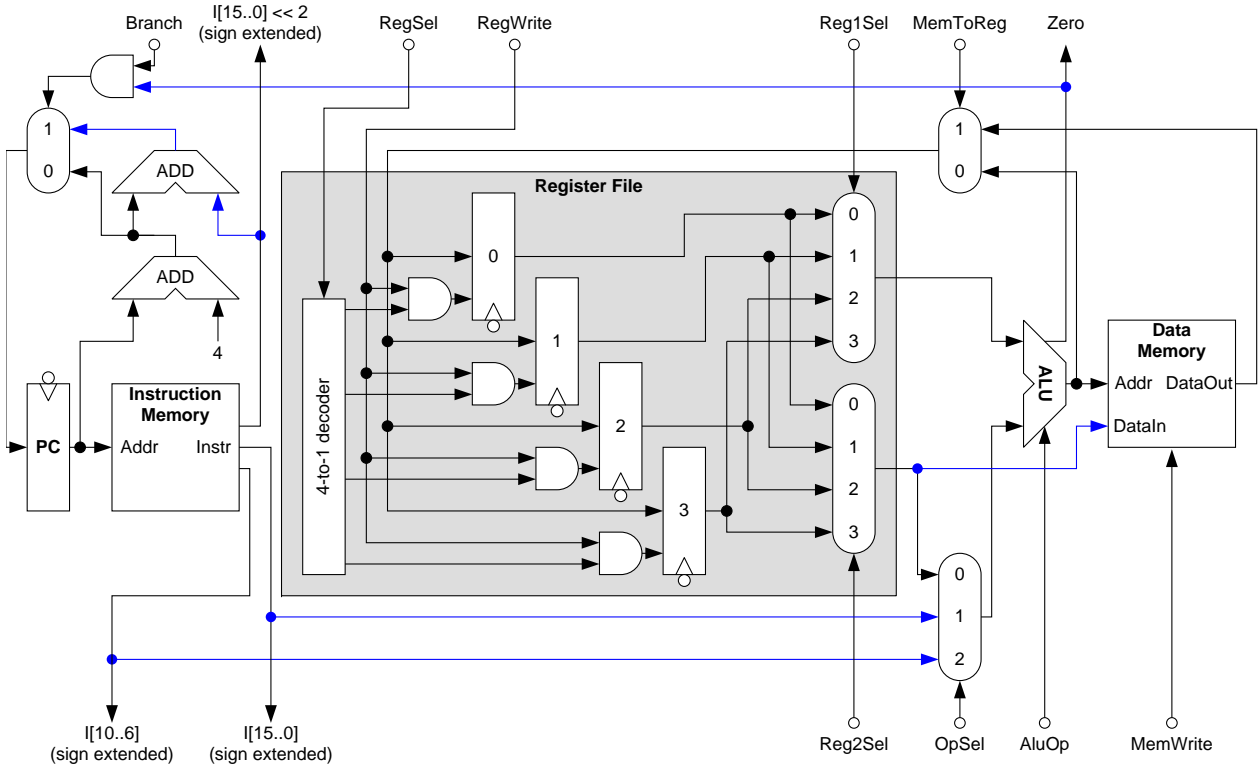
1 MIPS Architektur

Das auf der nächsten Seite abgebildete Blockdiagramm zeigt vereinfacht die wichtigsten Komponenten des Datenpfads der MIPS Architektur. Zur Vereinfachung besteht das Register File aus nur 4 (statt 32) Registern, wobei Register 0 die Konstante 0 enthält. Die Funktion der ALU wird über das Steuersignal `AluOp` definiert, wobei folgende Operationen implementiert sind: "add", "sub", "and", "or", "shift left", und "shift right". Das Zero Flag wird von der ALU auf 1 gesetzt, falls das Ergebnis einer Operation 0 ist.

Zudem ist auf der nächsten Seite ein Teil des MIPS-Instruktionssatzes aufgelistet, der von der dargestellten Architektur implementiert werden soll.

- (a) Vervollständigen Sie den gegebenen Datenpfad, sodass alle gegebenen Instruktionen ausgeführt werden können. Fügen Sie dazu zusätzliche Verbindungen zwischen den gegebenen Hardware Blöcken ein. Es werden keine neuen Hardware Blöcke benötigt.
- (b) Tragen Sie für die gegebenen Instruktionen die Steuersignale der Kontrolllogik in die gegebene Tabelle ein. Gehen Sie von einer active high Logik aus ($\text{HIGH} = 1$, $\text{LOW} = 0$) und markieren Sie Signale, deren Zustand egal ist, explizit mit "—". Als Beispiel sind die Steuersignale für die `add` Instruktion bereits in der Tabelle eingetragen. Die Variablen A , B und C stehen jeweils für eines der vier Register: $A, B, C \in \{0, 1, 2, 3\}$.

Lösung:



R-Format Instruktionen

	31	25	20	15	10	5	0
	6 bits	5 bits	5 bits	5 bits	5 bits	6 bits	
	op	rs	rt	rd	shmt	funct	
add	0	A	B	C	0	32	
sub	0	A	B	C	0	34	
and	0	A	B	C	0	36	
or	0	A	B	C	0	37	
sll	0	0	B	C	α	0	
srl	0	0	B	C	α	2	

Reg1Sel	Reg2Sel	OpSel	MemToReg	MemWrite	RegSel	RegWrite	Branch	AluOp
A	B	0	0	0	C	1	0	add
A	B	0	0	0	C	1	0	sub
A	B	0	0	0	C	1	0	and
A	B	0	0	0	C	1	0	or
B	—	2	0	0	C	1	0	sll
B	—	2	0	0	C	1	0	srl

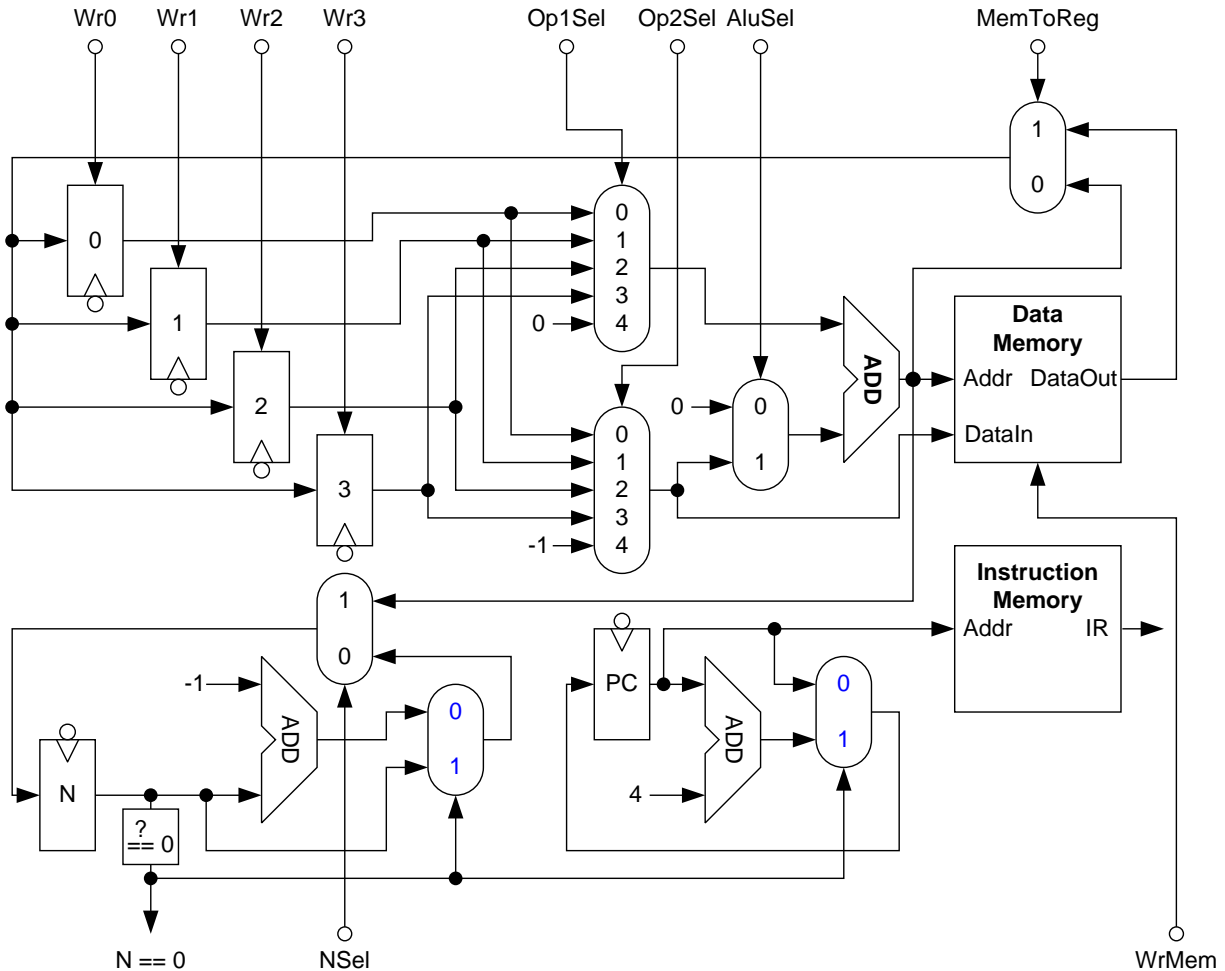
I-Format Instruktionen

	31	25	20	15	10	5	0
	6 bits	5 bits	5 bits	5 bits	5 bits	6 bits	
	op	rs	rt	immediate			
lw	35	A	B	α			
sw	43	A	B	α			
andi	12	A	B	α			
ori	13	A	B	α			
beq	4	A	B	α			

Reg1Sel	Reg2Sel	OpSel	MemToReg	MemWrite	RegSel	RegWrite	Branch	AluOp
A	—	1	1	0	B	1	0	add
A	B	1	—	1	—	0	0	add
A	—	1	0	0	B	1	0	and
A	—	1	0	0	B	1	0	or
A	B	0	—	0	—	0	1	sub

2 Loop Instruktion

In dieser Aufgabe wird der im Blockdiagramm dargestellte Prozessor betrachtet, der den in der Tabelle beschriebenen Instruktionssatz implementiert. Die Variable α steht für eines der vier Register:
 $\alpha \in \{0, 1, 2, 3\}$.



4 bits op	2 bits register	explanation
lw0	α	$\text{Reg}[0] = \text{MEM}[\text{Reg}[\alpha]]$
lw1	α	$\text{Reg}[1] = \text{MEM}[\text{Reg}[\alpha]]$
lw2	α	$\text{Reg}[2] = \text{MEM}[\text{Reg}[\alpha]]$
lw3	α	$\text{Reg}[3] = \text{MEM}[\text{Reg}[\alpha]]$
sw0	α	$\text{MEM}[\text{Reg}[\alpha]] = \text{Reg}[0]$
sw1	α	$\text{MEM}[\text{Reg}[\alpha]] = \text{Reg}[1]$
sw2	α	$\text{MEM}[\text{Reg}[\alpha]] = \text{Reg}[2]$
sw3	α	$\text{MEM}[\text{Reg}[\alpha]] = \text{Reg}[3]$
add0	α	$\text{Reg}[0] = \text{Reg}[0] + \text{Reg}[\alpha]$
add1	α	$\text{Reg}[1] = \text{Reg}[1] + \text{Reg}[\alpha]$
add2	α	$\text{Reg}[2] = \text{Reg}[2] + \text{Reg}[\alpha]$
add3	α	$\text{Reg}[3] = \text{Reg}[3] + \text{Reg}[\alpha]$
loop	α	$N = \text{Reg}[\alpha] - 1$

Op1Sel	Op2Sel	AluSel	NSel	Wr0	Wr1	Wr2	Wr3	WrMem	MemToReg
α	—	0	0	1	0	0	0	0	1
α	—	0	0	0	1	0	0	0	1
α	—	0	0	0	0	1	0	0	1
α	—	0	0	0	0	0	1	0	1
α	0	0	0	0	0	0	0	1	—
α	1	0	0	0	0	0	0	1	—
α	2	0	0	0	0	0	0	1	—
α	3	0	0	0	0	0	0	1	—
α	0	1	0	1	0	0	0	0	0
α	1	1	0	0	1	0	0	0	0
α	2	1	0	0	0	1	0	0	0
α	3	1	0	0	0	0	1	0	0
α	4	1	1	0	0	0	0	0	—

Im Folgenden wird die loop Instruktion näher beschrieben. loop α bewirkt, dass die unmittelbar nachfolgende Instruktion Reg[α] mal ausgeführt wird. (Es wird angenommen, dass die nachfolgende Instruktion keine Verzweigungsinstruktion bzw. keine weitere loop Instruktion ist.) Dazu stellt der Prozessor ein zusätzliches Register N zur Verfügung, das durch Ausführung der loop α Instruktion mit Reg[α] - 1 initialisiert wird. Die Kontrolllogik für das Aktualisieren des Programmzählers PC und des Registers N ist wie folgt definiert:

```
if (N == 0) { PC = PC + 4; N = 0; } else { PC = PC; N = N - 1; }
```

- (a) Indizieren Sie im Blockschaltbild auf Seite 3 die Eingänge der den Registern PC und N zugeordneten Multiplexer so, dass die Schaltung die beschriebene Logik implementiert.
- (b) Bestimmen Sie die Steuerungssignale des Kontrollpfades. Vervollständigen Sie dazu die Tabelle auf Seite 3. Gehen Sie wiederum von einer active high Logik aus und markieren Sie Signale, deren Zustand egal ist, explizit mit "—".
- (c) Implementieren Sie mit dem gegebenen Instruktionssatz unter Zuhilfenahme der loop Instruktion ein Assemblerprogramm mit folgender Funktion: MEM[Reg[2]] = MEM[Reg[0]] · MEM[Reg[1]]. Gehen Sie davon aus, dass Reg[3] zu Beginn den Wert 0 enthält. Die Registerinhalte der Register 0 bis 3 dürfen überschrieben werden.

Lösung:

```
lw0  r0
lw1  r1
loop r0
add3 r1
sw3  r2
```

- (d) (Zusatzaufgabe)
Erweitern Sie das unten abgebildete Petri-Netz, sodass es (1) die Ausführung der loop Instruktion modelliert und (2) die Logik für das Aktualisieren der Register PC und N modelliert. Zusätzlich zur Hilfsvariable NPC, können Sie weitere Hilfsvariablen verwenden.

Lösung:

