

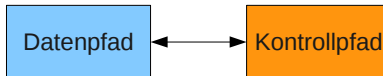
Technische Informatik I

Übung 6 – Datenpfad

David, **Marco**, Michael
zimmerling@tik.ee.ethz.ch

17. November 2011

- Vorlesung Kapitel 8
- Daten- und Kontrollpfad der MIPS Architektur
 - **Datenpfad**: Verarbeitung, Transport und Speicherung von Daten (z. B. Addierer, Bus, Instruktionsspeicher)
 - **Kontrollpfad**: Verarbeitung und Transport von Steuersignalen (z. B. Eingangssignal von Multiplexer und ALU)



- Heute: Einzeltakt-Implementierung
 - Eine Instruktion wird innerhalb *eines* Taktzyklus ausgeführt
 - Zustandsloser Kontrollpfad (kombinatorische Schaltung)
- Übung 7 und 8: Mehrzyklenimplementierung und Pipelining

Aufgabe 1: Unterstützte Instruktionen (Referenz)

	6 bits	5 bits	5 bits	5 bits	5 bits	6 bits	
	op	rs	rt	rd	shmt	funct	explanation
add	0	A	B	C	0	32	Reg[C] = Reg[A] + Reg[B]
sub	0	A	B	C	0	34	Reg[C] = Reg[A] - Reg[B]
and	0	A	B	C	0	36	Reg[C] = Reg[A] & Reg[B]
or	0	A	B	C	0	37	Reg[C] = Reg[A] Reg[B]
sll	0	0	B	C	α	0	Reg[C] = Reg[B] << α
srl	0	0	B	C	α	2	Reg[C] = Reg[B] >> α

	6 bits	5 bits	5 bits	16 bits	
	op	rs	rt	immediate	explanation
lw	35	A	B	α	Reg[B] = MEM[Reg[A] + α]
sw	43	A	B	α	MEM[Reg[A] + α] = Reg[B]
andi	12	A	B	α	Reg[B] = Reg[A] & α
ori	13	A	B	α	Reg[B] = Reg[A] α
beq	4	A	B	α	if (Reg[A] == Reg[B]) goto PC + 4 + 4 * α

Aufgabe 1: Teilaufgaben

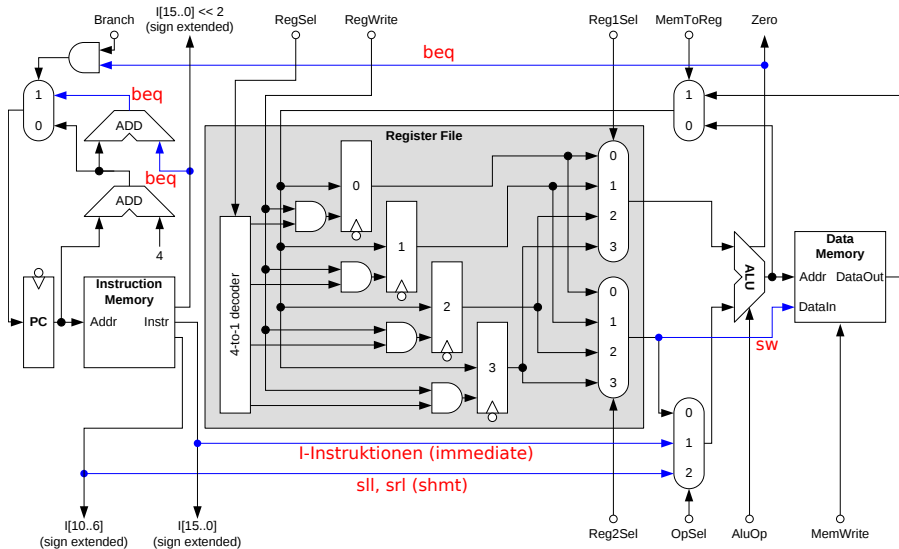
- Vervollständigen Sie den gegebenen Datenpfad (d. h. Verbindungen zwischen Hardware-Blöcken hinzufügen).
- Bestimmen Sie die Steuersignale der Kontrolllogik (active high). Kennzeichnen Sie Signale, deren Zustand egal ist, mit '–'.

Reg1Sel	Reg2Sel	OpSel	MemToReg	MemWrite	RegSel	RegWrite	Branch	AluOp
A	B	0	0	0	C	1	0	add

Vorgehensweise

- Für jede Instruktion den 'Weg' der Daten nachvollziehen (Eingabe-/Ausgabedaten, Register File oder Data Memory, etc.)
 - ⇒ Zusätzliche Verbindung(en) im Datenpfad notwendig?
 - ⇒ Steuersignale? Egal ('–') oder nicht egal?

Aufgabe 1: Lösung Teilaufgabe (a)



Aufgabe 1: Lösung Teilaufgabe (b)

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	shamt	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

Aufgabe 2: Loop Instruktion

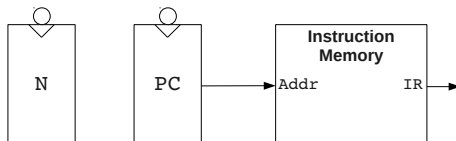
- Die **Instruktion**

loop α

bewirkt, dass die unmittelbar nachfolgende Instruktion $\text{Reg}[\alpha]$ mal ausgeführt wird. ($\alpha \in \{0, 1, 2, 3\}$ steht für eines von vier Registern.)

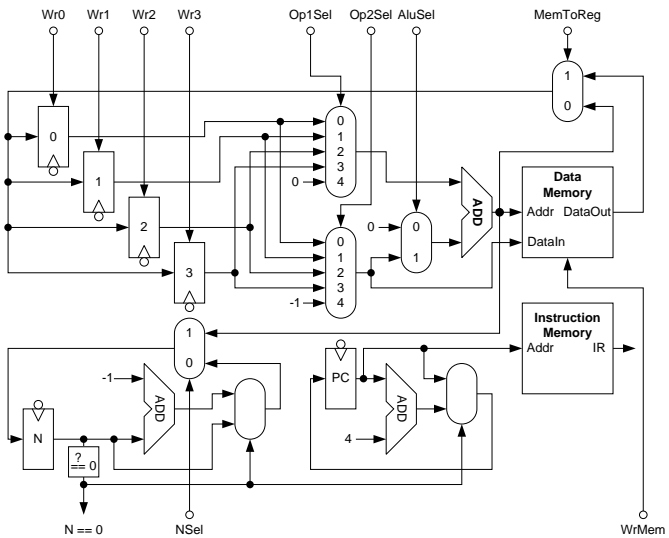
- Dazu stellt der Prozessor ein zusätzliches **Register N** (Schleifenzähler) zur Verfügung, das bei Ausführung der `loop` Instruktion mit $\text{Reg}[\alpha] - 1$ initialisiert wird.
- Logik zur Aktualisierung von PC und N

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



Aufgabe 2: Teilaufgaben

(a) Indizieren Sie die Eingänge der zwei unteren Multiplexer.

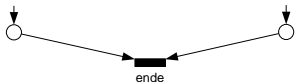
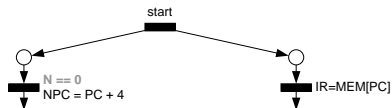


Aufgabe 2: Teilaufgaben (cont.)

- (c) Implementieren Sie mit Hilfe der `loop` Instruktion ein Assemblerprogramm mit folgender Funktion

$$\text{MEM}[\text{Reg}[2]] = \text{MEM}[\text{Reg}[0]] \cdot \text{MEM}[\text{Reg}[1]]$$

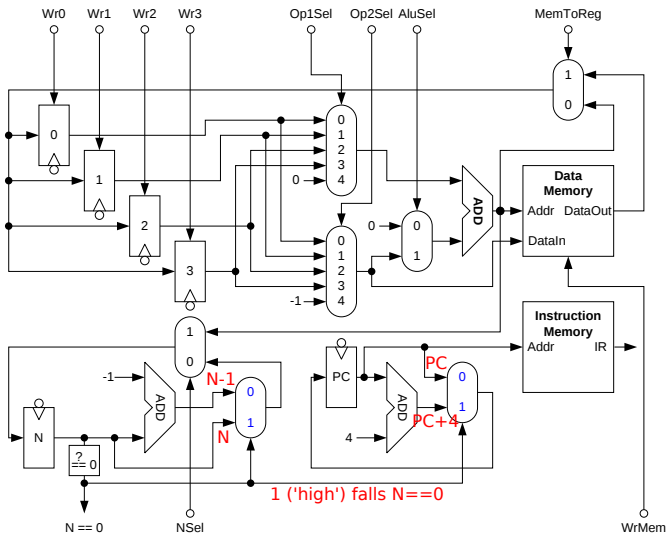
- (d) (Zusatzaufgabe) Erweitern Sie das gegebene Petri-Netz, sodass es die Ausführung der `loop` Instruktion und die Logik zur Aktualisierung von `PC` und `N` modelliert. Zusätzlich zur Hilfsvariablen `NPC` können Sie weitere Hilfsvariablen verwenden.



Vorgehensweise

- Was passiert bei Ausführung der `loop` Instruktion?
(`Op(IR) == loop`)
- Was passiert bei Ausführung der zu wiederholenden Instruktion?
(`Op(IR) != loop`)
- Welchen Wert hat `N` vor Ausführung von `loop` bzw. nach dem letzten Schleifendurchlauf?

Aufgabe 2: Lösung Teilaufgabe (a)



Aufgabe 2: Lösung Teilaufgabe (b)

- Mehrere Lösungen für $lw\{0, 1, 2, 3\}$ und $add\{0, 1, 2, 3\}$, da α sowohl durch $Op1Sel$ als auch durch $Op2Sel$ selektiert werden kann.

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

	Op1Sel	Op2Sel	AluSel	NSel	Wr0	Wr1	Wr2	Wr3	WrMem	MemToReg
lw	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	—
add	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	—
	α	—	—	—	—	—	—	—	—	—

Aufgabe 2: Lösung Teilaufgabe (c)

- Assoziativität der Multiplikation führt prinzipiell zu zwei Lösungen.

$$\text{MEM}[\text{Reg}[2]] = \text{MEM}[\text{Reg}[0]] \cdot \text{MEM}[\text{Reg}[1]]$$

$$\text{MEM}[\text{Reg}[2]] = \text{MEM}[\text{Reg}[1]] \cdot \text{MEM}[\text{Reg}[0]]$$

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

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

Aufgabe 2: Lösung Zusatzaufgabe (d)

